Modélisation orientée objet

SAP® Sybase® PowerAMC™ 16.5 SP03

Windows

ID DU DOCUMENT : DC31018-01-1653-01
DERNIERE REVISION : Novembre 2013

Table des matières

Partie I

Construction de MOO

Les chapitres de cette partie expliquent comment modéliser vos systèmes d'information dans SAP® Sybase® PowerAMC™.

Chapitre 1

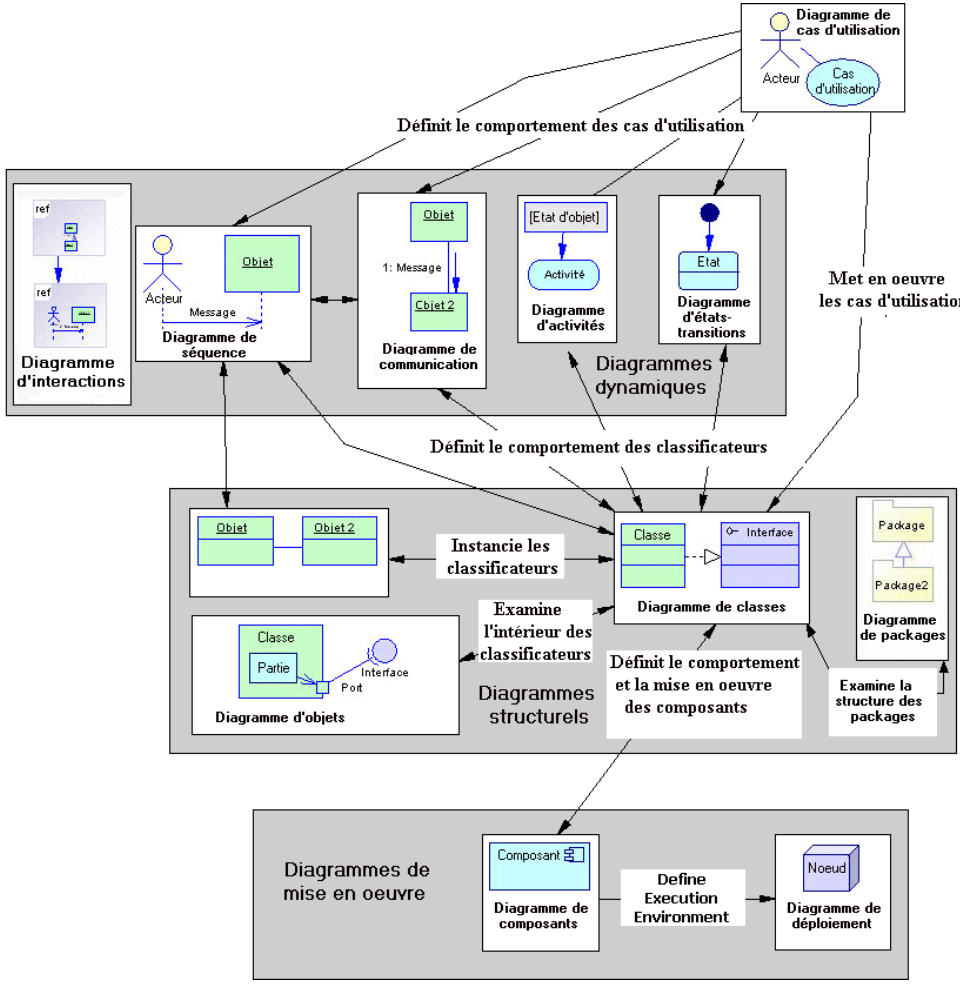
Notions de base relatives à la modélisation orientée objet

Un *modèle orienté objet (MOO)* vous aide à analyser un système d'information par l'intermédiaire de cas d'utilisations, d'analyses structurelles et comportementales, ainsi qu'en termes de déploiement, ce en utilisant le langage UML (Unified Modeling Language (UML) . Vous pouvez modéliser, procéder au reverse engineering et générer pour Java, .NET et d'autres langages.

PowerAMC™ prend en charge les diagrammes UML suivants :

- Diagramme de cas d'utilisation () - voir *Chapitre 2, Diagrammes de cas d'utilisation* à la page 17
- Diagrammes structurels :
  - Diagramme de classes () - voir *Diagrammes de classes* à la page 29
  - Diagramme de structures composites () - voir *Diagrammes de structures composites* à la page 31
  - Diagramme d'objets () - voir *Diagrammes d'objets* à la page 34
  - Diagramme de packages () - voir *Diagrammes de packages* à la page 33
- Diagramme dynamiques :
  - Diagramme de communications () - voir *Diagrammes de communication* à la page 131
  - Diagramme de séquence () - voir *Diagrammes de séquence* à la page 134
  - Diagramme d'activités () - voir *Diagrammes d'activités* à la page 137
  - Diagramme d'états-transitions () - voir *Diagrammes d'états-transitions* à la page 140
  - Diagramme d'interaction () - voir *Diagrammes d'interactions* à la page 143
- Diagrammes de mise en œuvre :
  - Diagramme de composants () - voir *Diagrammes de composants* à la page 219
  - Diagramme de déploiement () - voir *Diagrammes de déploiement* à la page 221

Dans l'illustration ci-dessous, vous pouvez voir de quelle façon les différents diagrammes UML peuvent interagir dans votre modèle :



Bibliographie suggérée

- James Rumbaugh, Ivar Jacobson, Grady Booch – The Unified Modeling Language Reference Manual – Addison Wesley, 1999
- Grady Booch, James Rumbaugh, Ivar Jacobson – The Unified Modeling Language User Guide – Addison Wesley, 1999
- Ivar Jacobson, Grady Booch, James Rumbaugh – The Unified Software Development Process – Addison Wesley, 1999
- Doug Rosenberg, Kendall Scott – Use Case Driven Object Modeling With UML A Practical Approach – Addison Wesley, 1999
- Michael Blaha, William Premerlani – Object-Oriented Modeling and Design for Database Applications – Prentice Hall, 1998

- Geri Schneider, Jason P. Winters, Ivar Jacobson – Applying Use Cases: A Practical Guide – Addison Wesley, 1998
- Pierre-Alain Muller – Instant UML – Wrox Press Inc, 1997
- Bertrand Meyer – Object-Oriented Software Construction – Prentice Hall, 2nd Edition, 1997
- Martin Fowler, Kendall Scott – UML Distilled Applying The Standard Object Modeling Language – Addison Wesley, 1997

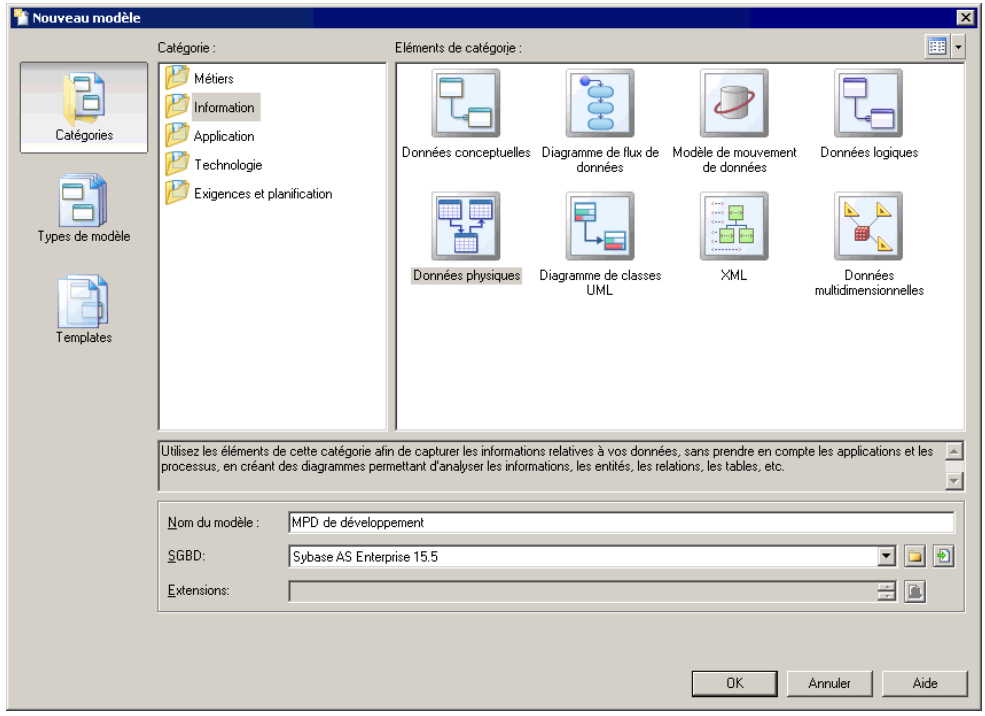
Création d'un MOO

Vous créez un nouveau modèle orienté objet en sélectionnant **Fichier > Nouveau modèle**.

Remarque : Outre le fait que vous pouvez créer de toutes pièces un MOO en utilisant la procédure décrite ci-après, vous pouvez également procéder au reverse engineering de code orienté objet existant dans nouveau modèle (voir *Reverse engineering de fichiers orientés objet dans un MOO* à la page 287).

La boîte de dialogue Nouveau modèle est largement configurable, et votre administrateur peut avoir masqué des options qui ne sont pas pertinentes pour votre travail ou avoir fourni des templates ou des modèles prédéfinis afin de vous guider dans la création d'un modèle. Lorsque vous ouvrez la boîte de dialogue, l'un ou plusieurs des boutons suivants sont disponibles du côté gauche :

- **Catégories** - fournit un jeu de modèles et de diagrammes prédéfinis triés au sein d'une arborescence de catégories configurable.
- **Types de modèle** - fournit la liste classique de types de modèle et de diagramme PowerAMC.
- **Fichiers de template** - fournit un jeu de templates de modèle triés par type de modèle.



1. Sélectionnez **Fichier > Nouveau modèle** pour afficher la boîte de dialogue Nouveau modèle.
2. Cliquez sur un bouton, puis sélectionnez une catégorie ou un type de modèle (**Modèle Orienté Objet**) dans le volet de gauche.
3. Sélectionnez un élément dans le volet de droite. Selon la façon dont votre boîte de dialogue Nouveau modèle est configurée, ces éléments peuvent être les premiers diagrammes ou des templates sur lesquels baser la création de votre modèle.

   Utilisez l'outil **Vues** dans l'angle supérieur droit de la boîte de dialogue afin de contrôler l'affichage des éléments.
4. Saisissez un nom pour le modèle. Le code du modèle, qui est utilisé pour la génération de script ou de code, est dérivé de son nom au moyen des conventions de dénomination.
5. Sélectionnez un langage objet cible, qui personnalise l'environnement d'édition PowerAMC par défaut à l'aide de propriétés, d'objets et de templates de génération spécifiques à la cible.

   Par défaut, PowerAMC créer un lien dans le modèle vers le fichier spécifié. Pour copier le contenu de la ressource et l'enregistrer dans votre modèle, cliquez sur le bouton **Incorporer la ressource dans le modèle** à droite de cette zone. En incorporant un fichier ainsi, vous pouvez effectuer des modifications spécifiques à votre modèle sans affecter les autres modèles qui référencent la ressource partagée.

6. [facultatif] Cliquez sur le bouton **Sélectionner des extensions** et attachez une ou plusieurs extensions à votre modèle.
7. Cliquez sur **OK** pour créer et ouvrir le modèle orienté objet .

Remarque : Des exemples de MOO sont disponibles dans le répertoire Exemples.

Propriétés d'un MOO

Pour afficher la feuille de propriétés d'un modèle, pointez sur le modèle dans l'Explorateur d'objets, cliquez le bouton droit de la souris et sélectionnez **Propriétés**.

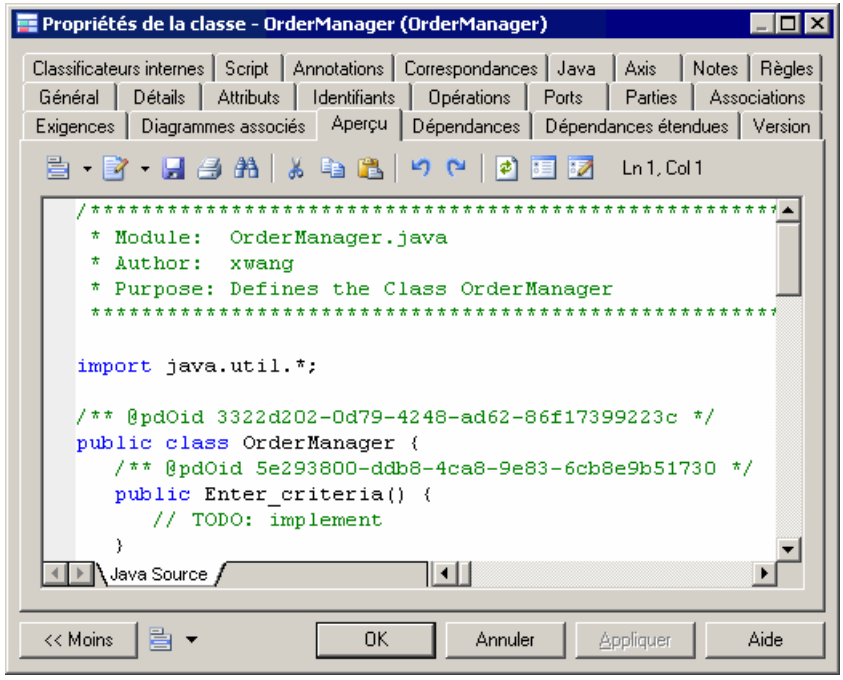
Chaque modèle orienté objet a les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient le modèle. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert le modèle, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré automatiquement à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Chemin du fichier	Spécifie l'emplacement du fichier du modèle. Cette zone est vide si le modèle n'a pas encore été enregistré
Auteur	Spécifie l'auteur du modèle. Si vous laissez cette zone à vide, le champ Auteur des cartouches de titre de diagramme affiche le nom d'utilisateur figurant sur l'onglet Version de la feuille de propriétés du modèle. Si vous saisissez un espace, le champ nom est vide.
Version	Spécifie la version du modèle. Vous pouvez utiliser cette zone pour afficher le numéro de version du référentiel ou un numéro de version personnalisé. Ce paramètre est défini dans les préférences d'affichage.
Langage objet	Spécifie le modèle cible.
Diagramme par défaut	Spécifie le diagramme qui s'affiche par défaut lorsque vous ouvrez le modèle.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Aperçu du code d'un objet

Cliquez sur l'onglet **Aperçu** dans la feuille de propriétés de modèle, package, classe, autre objet de modèle pour afficher le code qui sera généré pour cet objet.

Par exemple, si vous avez créé des composants EJB ou servlets dans Java, la page Aperçu affiche les fichiers de descripteur de déploiement d'EJB ou Web. Si vous avez sélectionné un langage XML, l'aperçu affiche le fichier de schéma qui correspond au fichier XML à générer.



Si vous avez sélectionné l'option **Aperçu modifiable** (disponible sous **Outils > Options du modèle**), vous pouvez modifier le code d'un classificateur directement dans son onglet **Aperçu**. Le code modifié doit être valide et s'applique uniquement au classificateur courant, faute de quoi les modifications seront ignorées. Vous avez la possibilité de créer des liens de généralisation et de réalisation à condition que leurs classificateurs existent déjà dans le modèle, mais vous ne pouvez pas renommer le classificateur ou modifier la déclaration de package pour la déplacer dans un autre package. Vous devez éviter de renommer des attributs et des opérations, sinon les propriétés qui ne sont pas générées (comme la description, l'annotation ou les attributs étendus) seront perdues. Les changements valides sont appliqués lorsque vous quittez l'onglet **Aperçu** ou que vous cliquez sur le bouton **Appliquer**.

Dans un modèle ayant pour cible PowerBuilder®, cette fonctionnalité doit être utilisée pour fournir une vision globale du code d'un objet et de ses fonctions, puisque cette dernière n'est

pas disponible dans PowerBuilder. Vous pouvez afficher l'onglet **Aperçu** pour vérifier où les variables d'instance sont utilisées dans le code. Vous pouvez également modifier le corps d'une fonction ou créer une nouvelle fonction par copier/coller.

Les outils suivants sont disponibles sur la barre d'outils de l'onglet **Aperçu** :

Outils	Description
	**Menu de l'éditeur [Maj+F11]** - Contient les commandes suivantes : • **Nouveau [Ctrl+N]** - Réinitialise la zone en supprimant tout le contenu existant. • **Ouvrir... [Ctrl+O]** - Remplace le contenu du champ par celui du fichier sélectionné. • **Insérer... [Ctrl+I]** - Insère le contenu du fichier sélectionné à l'emplacement du curseur. • **Enregistrer [Ctrl+S]** - Enregistre le contenu de la zone dans le fichier spécifié. • **Enregistrer sous...** - Enregistre le contenu de la zone dans un nouveau fichier. • **Sélectionner tout [Ctrl+A]** - Sélectionne tout le contenu de la zone. • **Rechercher... [Ctrl+F]** - Ouvre une boîte de dialogue afin de rechercher du texte dans la zone. • **Suivant... [F3]** - Trouve l'occurrence suivante du texte recherché. • **Précédent... [Maj+F3]** - Trouve l'occurrence précédente du texte recherché. • **Remplacer... [Ctrl+H]** - Ouvre une boîte de dialogue afin de remplacer du texte dans la zone. • **Aller à la ligne... [Ctrl+G]** - Ouvre une boîte de dialogue permettant d'aller à la ligne spécifiée. • **Activer/désactiver le signet [Ctrl+F2]** Insère et supprime un signet (marque bleue) à l'emplacement du curseur. Notez que les signets ne sont pas imprimables et sont perdus si vous réactualisez l'onglet, ou si vous utilisez l'outil **Afficher les options de génération**. • **Signet précédent [F2]** - Passe au signet suivant. • **Signet suivant [Maj+F2]** - Revient au signet précédent.
	**Editer avec [Ctrl+E]** - Ouvre le code affiché dans un éditeur externe. Cliquez sur la flèche vers le bas pour sélectionner un éditeur particulier ou choisissez **Choisir un programme** afin de spécifier un nouvel éditeur. Les éditeurs spécifiés ici sont ajoutés dans la liste des éditeurs disponible en sélectionnant **Outils > Options générales > Editeurs**.
	**Enregistrer [Ctrl+S]** - Enregistre le contenu de la zone dans le fichier spécifié.
	**Imprimer [Ctrl+P]** - Imprime le contenu de la zone.
	**Rechercher [Ctrl+F]** - Ouvre une boîte de dialogue afin de rechercher un texte.
	**Couper [Ctrl+X]**, **Copier [Ctrl+C]** et **Coller [Ctrl+V]** - Effectue les opérations de Presse-papiers standard.

Outils	Description
	**Annuler [Ctrl+Z]** et **Répéter [Ctrl+Y]** - Annule ou revalide les modifications.
	**Réactualiser [F5]** - Réactualise l'affichage de l'onglet Aperçu. Vous pouvez déboguer les templates du GTL qui génèrent le code affiché dans l'onglet Aperçu. Pour ce faire, ouvrez le fichier de ressource cible ou l'extension, sélectionnez l'option **Activer le suivi**, puis cliquez sur **OK** pour revenir au modèle. Vous pouvez être amené à cliquer sur l'outil **Réactualiser** pour afficher les templates.
	**Sélectionner les cibles de génération [Ctrl+F6]** - Permet de sélectionner des cibles de génération supplémentaires (définies dans des extensions), et ajoute un sous-onglet pour chaque cible sélectionnée. Pour plus d'informations sur les cibles de génération, voir *Personnalisation et extension de PowerAMC > Fichiers d'extension > Fichiers générés (Profile) > Génération de vos fichiers dans une génération standard ou étendue*.
	**Afficher les options de génération [Ctrl+W]** - Affiche la boîte de dialogue Options de génération, afin de vous permettre de modifier les options de génération et de voir leur impact sur le code. Cette fonctionnalité est particulièrement utile lorsque vous travaillez avec Java. Pour les autres langages, les options de génération n'influent pas sur le code.

Personnalisation des scripts de création d'objet

L'onglet Script permet de personnaliser le script de création de l'objet, par exemple en ajoutant des informations descriptives dans ce script.

Exemples

Si un projet de développement archive tous les scripts de création qui sont générés, un en-tête de script peut être inséré avant chaque script de création, indiquant la date, l'heure ainsi que toute autre information appropriée ou, si les scripts générés doivent être archivés en utilisant un système de dénomination qui soit indépendant du nom du script, un en-tête de script peut diriger un script généré de sorte qu'il soit enregistré sous un autre nom.

Vous pouvez insérer des scripts au début (sous-onglet Début) ou à la fin (sous-onglet Fin) d'un script ou bien avant ou après une commande de création de classe ou d'interface (sous-onglet Importations)

Les outils et la touches de raccourcis suivants sont disponibles sur l'onglet Script :

Outil	Description
	[**Maj+F11**] Menu de l'éditeur. Ouvre le menu contextuel de l'éditeur.
	[**Ctrl+E**] Editer avec. Affiche votre éditeur par défaut.

Outil	Description
	Importer un dossier - [sous-onglet Importations] Affiche une fenêtre permettant de sélectionner des packages à importer à l'emplacement du curseur, et le mot clé 'import' est ajouté comme préfixe.
	Importer un classificateur - [sous-onglet Importations] Affiche une fenêtre permettant de sélectionner des classificateurs à importer à l'emplacement du curseur, et le mot clé 'import' est ajouté comme préfixe

Vous pouvez utiliser la syntaxe de mise en forme suivante avec les variables :

Code de format	Format de la valeur de variable dans le script
.L	Caractères minuscules
.T	Suppression des espaces
.U	Caractères majuscules
.c	Initiale majuscule et lettres suivantes en minuscules
.*n*	Longueur maximale (n représentant le nombre de caractères
.*n*J	Justifié à une longueur fixe (n représentant le nombre de caractères)

Vous pouvez incorporer la syntaxe de format dans la syntaxe d'une variable comme suit :

%.*format*:*variable***%**

Personnalisation de votre environnement de modélisation

Le modèle orienté objet PowerAMC met à votre disposition différents moyens pour personnaliser et contrôler votre environnement de modélisation.

Définition des options de modèle de MOO

Vous pouvez définir des options de modèle de MOO en sélectionnant **Outils > Options de modèle** ou bien en pointant sur le fond du diagramme, en cliquant le bouton droit de la souris et en sélectionnant **Options du modèle**. Ces options affectent tous les objets dans le modèle, y compris ceux déjà créés.

Vous pouvez définir les options suivantes :

Option	Définition
Respect de la casse pour le nom/code	Spécifie que la casse des caractères est prise en compte pour les noms et codes de tous les objets, ce qui permet à deux objets d'avoir le même nom ou code, mais avec une casse de caractères différente, dans le même modèle. Si vous changez la prise en compte de la casse lors de la modélisation, nous vous recommandons de lancer une vérification de modèle afin de vous assurer que votre modèle ne contient pas des objets en double.
Activer les liens vers les exigences	Affiche un onglet **Exigences** dans la feuille de propriétés de chaque objet du modèle, ce qui permet d'attacher des exigences aux objets (voir *Modélisation des exigences*).
Afficher les classes comme types de données	Inclut les classes du modèle dans la liste des types de données que vous pouvez définir pour les attributs ou les paramètres, ou les types de résultats définis pour les opérations.
Aperçu modifiable	Concerne le reverse engineering uniquement. Vous pouvez éditer votre code à partir de la page Aperçu d'une classe ou d'une interface si vous avez coché la case Aperçu modifiable. Cette option permet d'effectuer un reverse engineering sur les modifications appliquées à votre code et ce, directement sur la page Aperçu.
Propriétés des raccourcis externes	Spécifie les propriétés qui sont stockées pour les raccourcis externes vers des objets contenus dans d'autres modèles à des fins d'affichage dans les feuilles de propriétés et les symboles. Par défaut **Toutes** les propriétés sont affichées, mais vous pouvez choisir d'afficher uniquement le **Nom/Code** afin de réduire la taille de votre modèle. **Remarque :** Cette option ne contrôle que les propriétés des raccourcis externes vers des modèles de même type (d'un MPD vers un MPD, d'un MAE vers un MAE, etc). Les raccourcis externes vers des objets contenus dans d'autres types de modèle peuvent uniquement afficher les propriétés de raccourci de base.
Types de données par défaut	Spécifie les types de données par défaut pour les attributs, les opérations et les paramètres. Si vous saisissez une valeur de type de données qui n'existe pas dans les listes BasicDataTypes et AdditionalDataTypes du langage objet, c'est la valeur de DefaultDataType qui est utilisée. Pour plus d'informations sur les types de données dans le langage objet, voir *Personnalisation et extension de PowerAMC > Fichiers de définition pour les langage objet, de processus et XML > Catégorie Settings : langage objet*.
Domaine/Attribut : Imposer la cohérence	Spécifie que les attributs attachés à un domaine doivent rester conformes aux propriétés de ce domaine. Vous pouvez spécifier la ou les sous-options suivantes : • Type de données – type de données, longueur et précision • Contrôle – paramètres de contrôle, par exemple valeurs minimum et maximum • Règles – règles de gestion

Option	Définition
Domaine/Attribut : Utiliser le nom complet du type de données	Spécifie que le nom complet du type de données est utilisé, au lieu de son nom abrégé. Cette option s'applique aux types de données d'attribut et permet de disposer d'informations plus lisibles dans la liste des types de données.
Conteneur par défaut d'association	Spécifie un conteneur par défaut pour les association qui ont un rôle avec une multiplicité supérieure à un.
Message : Prise en charge du retard	Spécifie que les messages peuvent avoir une durée (message incliné). Si cette option est désélectionnée, les messages sont instantanés, ou rapides (message horizontal).
Interface/Classe : Mettre en oeuvre automatiquement les interfaces réalisées	Ajoute à la classe qui réalise les méthodes d'une interface réalisée et de ses parents, si elles ne sont pas déjà mises en oeuvre par celle-ci. Le stéréotype <<implement>> est appliqué aux méthodes.
Interface/Classe : Visibilité par défaut d'attribut de classe	Spécifie la visibilité par défaut des attributs de classe.

Remarque : Pour plus d'informations sur la spécification de conventions de dénomination pour vos objets de modèle, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets > Conventions de dénomination*.

Définition des préférences d'affichage de MOO

Les préférences d'affichage de PowerAMC permettent de personnaliser le format des symboles d'objet et les informations qu'ils affichent. Pour définir les préférences d'affichage de modèle orienté objet , sélectionnez **Outils > Préférences d'affichage** ou pointez sur le fond du diagramme, cliquez le bouton droit de la souris, puis sélectionnez **Préférences d'affichage** dans le menu contextuel.

Pour obtenir des informations détaillées sur la personnalisation et le contrôle des attributs et collections affichés sur les symboles d'objet, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Diagrammes, matrices et symboles > Préférences d'affichage*.

Visualisation et édition du fichier de définition du langage objet

Chaque MOO est lié à un fichier de définition qui étend le métamodèle PowerAMC standard afin de proposer des objets, des propriétés, des types de données, des paramètres et templates de génération spécifiques à cette cible. Les fichiers de définition et les autres fichiers de ressources sont des fichiers XML situés dans le dossier `Fichiers de ressources` de votre répertoire d'installation, et peuvent être ouverts et édités dans l'Editeur de ressources de PowerAMC.

Avertissement ! Les fichiers de ressource fournis avec PowerAMC dans le dossier `Program Files` ne peuvent pas être modifiés directement. Pour créer une copie à des fins d'édition, utilisez l'outil **Nouveau** dans la liste de fichiers de ressource, puis enregistrez-la à un autre emplacement. Pour inclure des fichiers de ressource provenant d'autres emplacements afin de les utiliser dans vos modèles, utilisez l'outil **Chemin** dans la liste des fichiers de ressource.

Pour afficher le fichier de définition de votre modèle et examiner ses extensions, sélectionnez **Langage > Editer le langage objet courant**.

Pour obtenir des informations détaillées sur le format de ces fichiers, voir *Personnalisation et extension de PowerAMC > Fichiers de définition pour les langage objet, de processus et XML*.

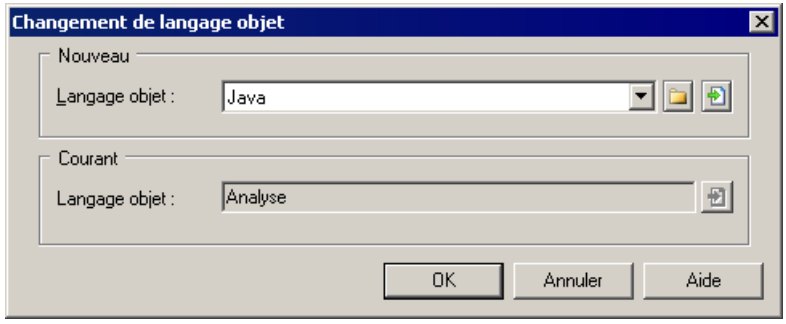
Remarque : Certains fichiers de ressources sont fournis avec la mention "Not certified" dans leur nom. Sybase® s'efforce de procéder à tous les contrôles de validation possibles, toutefois, Sybase n'assure pas la maintenance d'environnements spécifiques permettant la certification complète de ce type de fichiers de ressources. Sybase assure le support de la définition en acceptant les rapports de bogues et fournit les correctifs nécessaires dans le cadre d'une politique standard, mais ne peut être tenu de fournir une validation finale de ces correctifs dans l'environnement concerné. Les utilisateurs sont donc invités à tester ces correctifs fournis par Sybase afin de signaler d'éventuelles incohérences qui pourraient subsister.

Changement du langage objet

Vous pouvez changer le **langage objet** modélisé dans votre MOO à tout moment.

Remarque : Vous pouvez être amené à changer de langage objet si vous ouvrez un modèle et que le fichier de définition associé n'est pas disponible. Les fichiers de définition de langage sont fréquemment mis à jour dans chaque version de PowerAMC, nous vous recommandons donc d'accepter ce changement, faute de quoi vous ne serez pas en mesure de générer pour le langage sélectionné.

1. Sélectionnez **Langage > Changer le langage objet courant** :



2. Sélectionnez un **langage objet** dans la liste.

   Par défaut, PowerAMC créer un lien dans le modèle vers le fichier spécifié. Pour copier le contenu de la ressource et l'enregistrer dans votre modèle, cliquez sur le bouton

Incorporer la ressource dans le modèle à droite de cette zone. En incorporant un fichier ainsi, vous pouvez effectuer des modifications spécifiques à votre modèle sans affecter les autres modèles qui référencent la ressource partagée.

3. Cliquez sur **OK**.

   Une boîte de message s'affiche pour vous indiquer que le langage objet a été modifié.

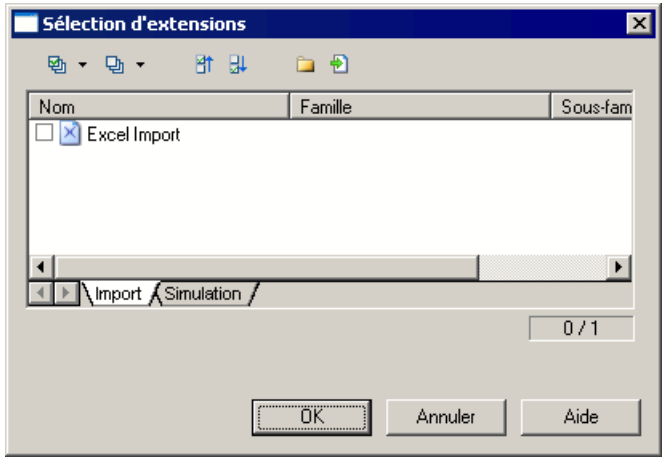
4. Cliquez sur **OK** pour revenir au modèle.

Extension de votre environnement de modélisation

Vous pouvez personnaliser et étendre les métaclasses PowerAMC, les paramètres et la génération de fichiers au moyen d'extensions qui peuvent être stockées comme faisant partie de votre modèle ou sous la forme de fichiers d'extension séparés (fichiers *.xem) afin de les réutiliser avec d'autres modèles.

Pour accéder à l'extension définie dans un fichier `*.xem`, il vous suffit d'attacher le fichier à votre modèle. Vous pouvez réaliser cette opération lors de la création d'un nouveau modèle en cliquant sur le bouton **Sélectionner des extensions** en bas de la boîte de dialogue Nouveau modèle, ou à tout moment en sélectionnant **Modèle > Extensions** pour afficher la boîte de dialogue Liste des extensions et en cliquant sur l'outil **Attacher une extension**.

Dans chacun de ces cas, vous parvenez à la boîte de dialogue Sélection d'extension, qui répertorie les extensions disponibles, réparties sur des sous-onglets, et qui dépendent du type de modèle sur lequel vous travaillez :



Pour en savoir plus sur l'extension des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets > Extension d'objets*. Pour obtenir des informations détaillées sur l'utilisation des extensions, voir *Personnalisation et extension de PowerAMC > Fichiers d'extension*.

Liaison d'objets à l'aide de liens de traçabilité

Vous pouvez créer des liens de traçabilité pour montrer tout type de relation entre deux objets de modèle (y compris entre des objets de modèles différents) via l'onglet **Liens de traçabilité** de la feuille de propriétés de l'objet. Ces liens sont utilisés à des fins de documentation uniquement, et ne sont pas interprétés ou vérifiés par PowerAMC.

Pour plus d'informations sur les liens de traçabilité, voir *Guide des fonctionnalités générales > Liaison et synchronisation de modèles > Notions de base relatives à la liaison et à la synchronisation > Création de liens de traçabilité*.

Chapitre 2 Diagrammes de cas d'utilisation

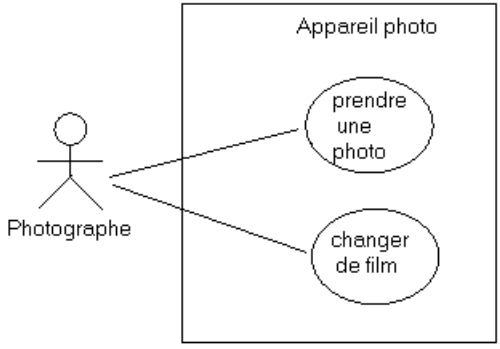
Un *diagramme de cas d'utilisation* est un diagramme UML qui fournit une représentation graphique des exigences de votre système, et vous aide à identifier la façon dont les utilisateurs interagissent avec ce dernier.

Remarque : Pour créer un diagramme cas d'utilisation dans un MOO existant, pointez sur le modèle dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Diagramme de cas d'utilisation**. Pour créer un nouveau modèle, sélectionnez **Fichier > Nouveau modèle**, choisissez Modèle Orienté Objet comme type de modèle et **Diagramme de cas d'utilisation** comme premier diagramme, puis cliquez sur **OK**.

Avec un diagramme de cas d'utilisation, vous disposez d'un aperçu instantané des fonctionnalités du système. Vous pouvez par la suite ajouter plus de détails dans le diagramme si vous le souhaitez afin d'éclaircir certains points relatifs au comportement du système.

Un diagramme de cas d'utilisation est particulièrement approprié pour décrire toutes les tâches pouvant être réalisées à l'aide d'un système de base de données par toutes les personnes susceptibles de l'utiliser. En revanche, il serait peu adapté pour décrire le protocole TCP/IP en raison du nombre de cas d'exception, de branchements et de fonctionnalités conditionnelles (que se passe-t-il lorsque la connexion est coupée, que se passe-t-il lorsqu'un paquet est perdu ?)

L'exemple suivant montre le cas d'utilisation d'un appareil photo, l'acteur "Photographe" effectue deux actions avec son appareil : prendre des clichés et changer de pellicule. L'action de prendre une photo implique ainsi de régler le diaphragme et la mise au point et d'ouvrir et de refermer l'obturateur. Ces activités ne sont toutefois pas d'un niveau suffisant pour être représentées dans un cas d'utilisation.



Objets du diagramme de cas d'utilisation

PowerAMC prend en charge tous les objets nécessaires pour construire des diagrammes de cas d'utilisation.

Objet	Outil	Symbole	Description
Acteur			Utilisé pour représenter une personne externe, un processus ou quelque chose qui interagit avec un système, avec un sous-système ou avec une classe. Voir *Acteurs (MOO)* à la page 21.
Cas d'utilisation			Définit un comportement cohérent dans un système, sans révéler sa structure interne. Voir *Cas d'utilisation (MOO)* à la page 18.
Association			Chemin de communication entre un acteur et un cas d'utilisation auquel il participe. Voir *Associations de cas d'utilisation (MOO)* à la page 25.
Généralisation			Un lien entre un cas d'utilisation général et un cas d'utilisation plus spécifique dont il hérite et qui lui ajoute des fonctionnalités. Voir *Généralisations (MOO)* à la page 101.
Dépendance			Relation entre deux éléments de modélisation, dans laquelle tout changement effectué sur un élément affecte l'autre élément. Voir *Dépendances (MOO)* à la page 105.

Cas d'utilisation (MOO)

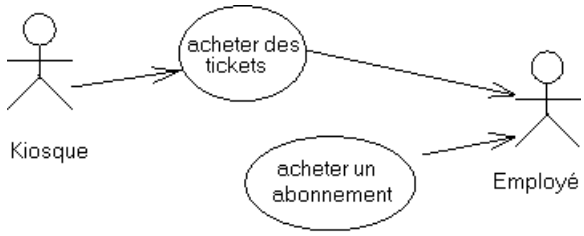
Un *cas d'utilisation* est une interaction entre un utilisateur et un système (ou une partie d'un système). Il définit un but particulier que l'utilisateur souhaite atteindre dans le système, sans révéler la structure interne du système.

Vous pouvez créer un cas d'utilisation dans le diagramme suivant :

- Diagramme de cas d'utilisation

Exemple

Dans cet exemple, acheter des tickets et acheter un abonnement sont des cas d'utilisation.



Création d'un cas d'utilisation

Vous pouvez créer un cas d'utilisation à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Cas d'utilisation** dans la Boîte à outils.
- Sélectionnez **Modèle > Cas d'utilisation** pour afficher la boîte de dialogue Liste des cas d'utilisation, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Cas d'utilisation**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'un cas d'utilisation

Pour visualiser ou modifier les propriétés d'un cas d'utilisation, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.

Propriété	Description
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Onglet Spécification

L'onglet Spécification contient les propriétés suivantes, accessibles via les sous-onglets situés en bas de la boîte de dialogue :

Propriété	Description
Suite d'actions	Spécifie une description sous forme de texte de la séquence normale d'actions associée à un cas d'utilisation. Par exemple : "ouvrir un dossier, attribuer un nouveau numéro et inscrire des indications thérapeutiques" pourrait constituer la suite d'actions pour un cas d'utilisation appelé "inscrire un patient' dans un hôpital".
Points d'extension	Spécifie une description sous forme de texte des actions qui prolongent une séquence d'actions normale. Les extensions sont généralement introduites par une syntaxe "If...then". Par exemple, dans le cas de figure mentionné ci-dessus, une extension de la suite d'actions pourrait être : "Si le patient a déjà un numéro, son dossier personnel est extrait".
Exceptions	Spécifie des signaux émis en réponse à des erreurs lors de l'exécution d'un système.
Pré-conditions	Spécifie des contraintes qui doivent être vérifiées pour qu'une opération puisse être appelée.
Post-conditions	Spécifie des contraintes qui doivent être vérifiées pour qu'une opération se termine correctement.

Onglet Classes de mise en oeuvre

Un cas d'utilisation est le plus souvent une tâche ou un service, représenté par un verbe. Lorsque vous analysez ce qu'un cas d'utilisation doit faire, vous pouvez identifier les classes et interfaces qui doivent être créées pour accomplir cette tâche, puis les associer au cas d'utilisation. L'onglet Classes de mise en oeuvre affiche la liste des classes ou des interfaces utilisées pour mettre en oeuvre un cas d'utilisation.

Outil	Action
	Ajouter des objets – Affiche une boîte de dialogue dans laquelle vous pouvez sélectionner n'importe quelle classe ou interface existant dans le modèle afin de mettre en oeuvre votre cas d'utilisation

Outil	Action
	Créer une nouvelle classe – Crée une nouvelle classe pour mettre en oeuvre le cas d'utilisation.
	Créer une nouvelle interface - Crée une nouvelle interface pour mettre en oeuvre le cas d'utilisation

Par exemple, un cas d'utilisation *envoyer le produit en recommandé* peut être mis en oeuvre par les classes *Expédier*, *Produit* et *Facturation*.

Onglet Diagrammes associés

L'onglet Diagrammes associés vous aide à mieux comprendre le cas d'utilisation. Cliquez sur l'outil **Ajouter des objets** pour ajouter des diagrammes dans la liste à partir de n'importe quel modèle ouvert dans l'espace de travail. Pour plus d'informations, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Diagrammes, matrices et symboles > Diagrammes > Spécification de diagrammes comme diagrammes associés*.

Acteurs (MOO)

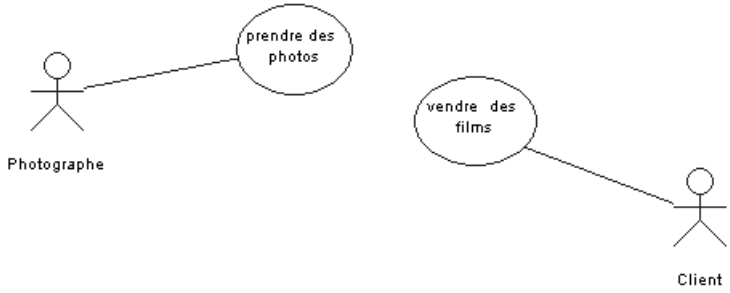
Un *acteur* personnifie un utilisateur ou un groupe d'utilisateurs extérieur qui interagit avec le système. Les acteurs peuvent être des humains ou d'autres systèmes externes. Par exemple, les acteurs d'un réseau informatique peuvent inclure l'administrateur système, un administrateur de base de données et des utilisateurs. Les acteurs sont les entités dont le comportement ne peut pas être contrôlé ou modifié car ils ne font pas partie du système que vous décrivez.

Un acteur peut être créé dans les diagrammes suivants :

- Diagramme de collaboration
- Diagramme de séquence
- Diagramme de cas d'utilisation

Un même acteur peut être utilisé dans des diagrammes de cas d'utilisation, de séquence et de collaboration s'il joue le même rôle dans chaque diagramme. Chaque acteur est disponible pour tous les diagrammes de votre MOO. Un diagramme peut être créé directement dans le type de diagramme de votre choix, ou bien être ajouté par glisser-déposer depuis un autre type de diagramme.

Acteurs dans un diagramme de cas d'utilisation



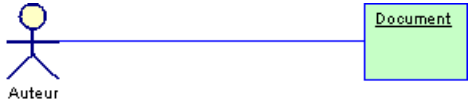
Dans le diagramme de cas d'utilisation, un acteur est un *acteur principal* pour un cas d'utilisation s'il sollicite et/ou déclenche les actions effectuées par ce cas d'utilisation. Les acteurs principaux sont situés à gauche du cas d'utilisation, et l'association qui les lie doit être tracée depuis l'acteur vers le cas d'utilisation.

Un acteur est un *acteur secondaire* pour un cas d'utilisation s'il ne déclenche pas les actions, mais qu'il assiste le cas d'utilisation dans l'accomplissement de ces actions. Une fois une action accomplie, le cas d'utilisation peut donner des résultats, des documents, ou de l'information vers l'extérieur : l'acteur secondaire est alors celui qui les reçoit. Les acteurs secondaires sont situés à droite du cas d'utilisation. Pour indiquer qu'un acteur est un acteur secondaire d'un cas d'utilisation, l'association qui les relie doit être dessinée depuis le cas d'utilisation vers l'acteur.

A une échelle plus globale, un acteur secondaire peut également être l'acteur principal pour un autre cas d'utilisation, situé dans le même diagramme ou dans un autre diagramme.

Acteurs dans un diagramme de collaboration

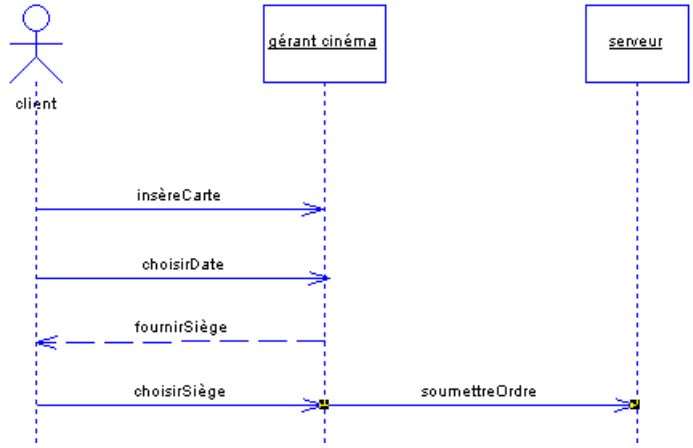
Dans un diagramme de collaboration, un acteur peut être connecté à un objet par un lien entre objets, ou peut envoyer ou recevoir des messages.



Acteurs dans un diagramme de séquence

Dans le diagramme de séquence, un acteur a une ligne de vie qui représente la durée de sa vie. Vous ne pouvez pas dissocier un acteur de sa ligne de vie.

Si un acteur est celui qui appelle une interaction, il est le plus souvent représenté par la première ligne de vie du diagramme (celle située le plus à gauche). Si votre diagramme comporte plusieurs acteurs, efforcez-vous de les placer à gauche ou à droite des lignes de vie existantes car les acteurs sont, par définition, externes au système.



Création d'un acteur

Vous pouvez créer un acteur à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Acteur** dans la Boîte à outils.
- Sélectionnez **Modèle > Acteurs** pour afficher la boîte de dialogue Liste des acteurs, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Acteur**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'un acteur

Pour visualiser ou modifier les propriétés d'un acteur, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Onglet Classes de mise en oeuvre

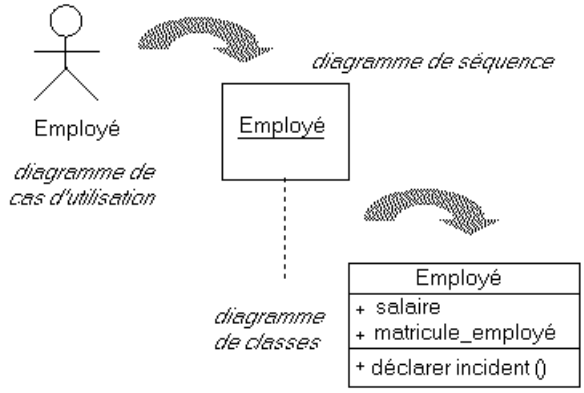
Un acteur peut être un être humain (personne, partenaire), une machine, ou un processus (système automatisé). Lorsque vous analysez ce qu'un acteur doit faire, vous pouvez identifier les classes et interfaces qui doivent être créées pour accomplir cette tâche, puis les associer à l'acteur. L'onglet Classes de mise en oeuvre répertorie les classes et interfaces utilisées pour mettre en oeuvre l'acteur. Les outils suivants sont disponibles :

Outil	Action
	Ajouter des objets – Affiche une boîte de dialogue dans laquelle vous pouvez sélectionner n'importe quelle classe ou interface existant dans le modèle afin de mettre en oeuvre votre acteur
	Créer une nouvelle classe – Crée une nouvelle classe pour mettre en oeuvre l'acteur
	Créer une nouvelle interface - Crée une nouvelle interface pour mettre en oeuvre l'acteur.

Par exemple, un acteur `Voiture` peut être mis en oeuvre par les classes `Moteur` et `Route`.

D'un point de vue conceptuel, vous pouvez lier les éléments encore plus avant. Par exemple, un employé travaillant dans une société d'assurance est représenté sous la forme d'un acteur dans un diagramme de cas d'utilisation, et traite avec des clients qui déclarent un accident de voiture.

L'acteur de l'employé devient un objet dans un diagramme de communication ou de séquence, et reçoit des messages des clients et en envoie à son responsable, qui est une instance de la classe Employé dans un diagramme de classes avec ses attributs et opérations hérités :



Onglet Diagrammes associés

L'onglet Diagrammes associés vous aide à mieux comprendre l'acteur. Cliquez sur l'outil **Ajouter des objets** pour ajouter des diagrammes dans la liste à partir de n'importe quel modèle ouvert dans l'espace de travail. Pour plus d'informations, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Diagrammes, matrices et symboles > Diagrammes > Spécification de diagrammes comme diagrammes associés*.

Réutilisation des acteurs

Un même acteur peut être utilisé dans un diagramme de cas d'utilisation, dans un diagramme de collaboration et dans un diagramme de séquence. Pour réutiliser dans un autre diagramme un acteur créé dans un diagramme particulier :

- Sélectionnez l'acteur dans l'Explorateur d'objets, et faites-le glisser dans le nouveau diagramme.
- Sélectionnez **Symboles > Afficher les symboles** dans le nouveau diagramme afin d'ouvrir la boîte de dialogue Affichage des symboles, cochez la case de l'acteur à afficher, puis cliquez sur OK.

Associations de cas d'utilisation (MOO)

Une *association* est une relation unidirectionnelle qui décrit un lien entre des objets.

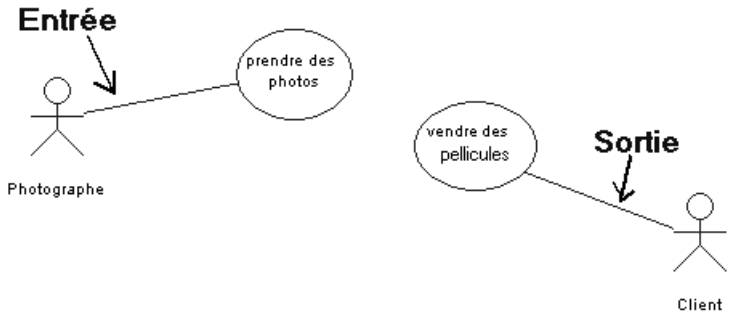
Les associations de cas d'utilisation ne peuvent être créées que dans des diagrammes de cas d'utilisation. Vous pouvez les créer en traçant une association entre :

- Un acteur et un cas d'utilisation - soit une association d'*entrée*
- Un cas d'utilisation et un acteur - soit une association de *sortie*

Le standard UML n'affiche pas de façon explicite l'orientation de l'association, qui est plutôt traduite par la position des actions. Lorsqu'un acteur est placé à gauche du cas d'utilisation, l'association est une entrée, et lorsque l'acteur se trouve à sa droite, il s'agit d'une sortie. Pour

afficher de façon explicite l'orientation de l'association, sélectionnez **Outils > Préférences d'affichage**, sélectionnez **Association de cas d'utilisation** dans l'arborescence Catégorie, puis l'option **Orientation**. Pour obtenir des informations détaillées sur l'utilisation des préférences d'affichage, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Diagrammes, matrices et symboles > Préférences d'affichage*.

Exemple



Création d'une association de cas d'utilisation

Vous pouvez créer une association de cas d'utilisation à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Association** dans la Boîte à outils.
- Sélectionnez **Modèle > Associations** pour afficher la boîte de dialogue Liste des associations, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Association**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'une association de cas d'utilisation

Pour visualiser ou modifier les propriétés d'une association de cas d'utilisation, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Orientation	Définit la direction de l'association. Vous pouvez choisir entre : • Acteur principal – l'association va de l'acteur vers le cas d'utilisation. • Acteur secondaire – l'association va du cas d'utilisation vers l'acteur.
Source	Spécifie l'objet où commence l'association. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Destination	Spécifie l'objet où se termine l'association. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Chapitre 3 Diagrammes structurels

Ces diagrammes abordés dans ce chapitre permettent de modéliser la structure statique de votre système. PowerAMC propose trois types de diagrammes pour la modélisation de cet aspect de votre système, chacun offrant une vue différente de vos objets et de leurs relations.

- Un *diagramme de classes* montre la structure statique des classes qui composent le système. Vous pouvez utiliser un diagramme de classes pour identifier le type des objets qui vont composer votre système, et définir les modalités selon lesquelles ils seront associés. Pour plus d'informations, voir *Diagrammes de classes* à la page 29.
- Un *diagramme de structure composite* permet de définir plus en détails la structure interne de vos classes et les modalités selon lesquelles elles seront associées. Vous pouvez utiliser un diagramme de structure composite en particulier pour modéliser des formes de composition qui seraient particulièrement fastidieuses à modéliser dans un diagramme de classes. Pour plus d'informations, voir *Diagrammes de structures composites* à la page 31.
- Un *diagramme d'objets* est similaire à un diagramme de classes, à ceci près qu'il montre des instances particulières des classes. Vous utilisez un diagramme d'objets pour représenter un instantané des relations entre les instances des classes. Pour plus d'informations, voir *Diagrammes d'objets* à la page 34.
- Un *diagramme de packages* montre la structure des packages qui constituent votre application, et les relations entre elles. Pour plus d'informations, voir *Diagrammes de packages* à la page 33.

Diagrammes de classes

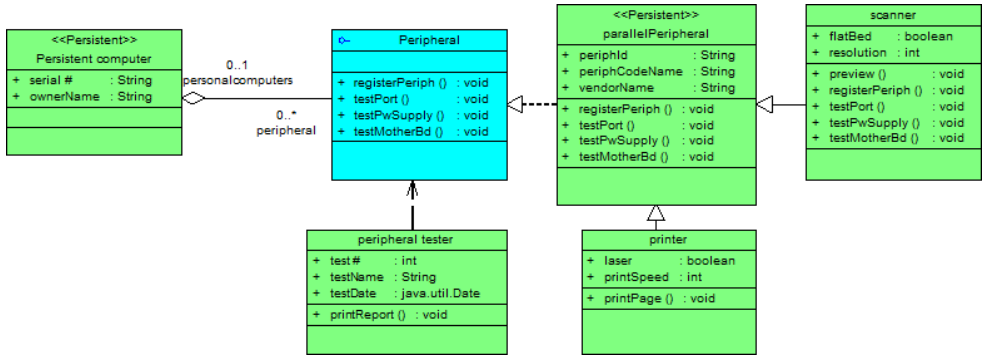
Un *diagramme de classes* est un diagramme UML qui fournit une représentation graphique des classes, interfaces, et packages qui composent un système, ainsi que des relations entre eux.

Remarque : Pour créer un diagramme de classes dans un MOO existant, pointez sur le modèle dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Diagramme de classes**. Pour créer un nouveau modèle, sélectionnez **Fichier > Nouveau modèle**, choisissez Modèle Orienté Objet comme type de modèle et **Diagramme de classes** comme premier diagramme, puis cliquez sur **OK**.

Vous construisez un diagramme de classes pour simplifier l'interaction des objets d'un système que vous êtes en train de modéliser. Les diagrammes de classes expriment la structure statique d'un système en termes de classes et de relations entre ces classes. Une classe décrit un ensemble d'objets et une association décrit un ensemble de liens. Les objets sont les instances d'une classe, et les liens sont les instances d'une association.

Un diagramme de classes n'exprime rien de spécifique concernant les liens d'un objet particulier, mais il décrit, de façon abstraite, le lien potentiel entre un objet et d'autres objets.

L'exemple suivant montre l'analyse de la structure de périphériques dans un diagramme de classes :



Objets du diagramme de classes

PowerAMC prend en charge tous les objets nécessaires pour construire des diagrammes de classes.

Objet	Outil	Symbole	Description
Classe		Compte	Jeu d'objets qui partagent les mêmes attributs, opérations, méthodes et relations. Voir *Classes (MOO)* à la page 37.
Interface		Signal	Descripteur des opérations visibles de l'extérieur pour une classe, un objet ou toute autre spécification de structure interne. Voir *Interfaces (MOO)* à la page 55.
Port			Point d'interaction entre un classificateur et son environnement. Voir *Ports (MOO)* à la page 65.
Généralisation			Lien entre classes montrant que la sous-classe partage la structure ou le comportement d'une ou plusieurs superclasses. Voir *Généralisations (MOO)* à la page 101.
Lien de prérequis			Connecte les classes, les composants, les ports, les parties et les interfaces. Voir *Liens de prérequis (MOO)* à la page 110.
Association			Relation structurelle entre les objets de différentes classes. Voir *Associations (MOO)* à la page 90.

Objet	Outil	Symbole	Description
Agrégation			Forme d'association qui définit une relation tout-partie entre une classe composant et une classe agrégat (par exemple, une voiture a un moteur et des roues). Voir *Associations (MOO)* à la page 90.
Composition			Forme d'agrégation dans laquelle la notion de propriété et la coïncidence sont fortes entre les parties et le tout. Les parties vivent et meurent avec le tout (par exemple, une facture et un élément de cette facture). Voir *Associations (MOO)* à la page 90.
Dépendance			Relation entre deux éléments de modélisation, dans laquelle tout changement intervenant sur l'un des éléments est répercuté sur le second élément. Voir *Dépendances (MOO)* à la page 105.
Réalisation			Relation sémantique entre classificateurs, dans laquelle un classificateur spécifie un contrat que l'autre classificateur s'engage à remplir. Voir *Réalisations (MOO)* à la page 109.
Lien interne			Existe lorsqu'une classe est déclarée au sein d'une autre classe ou interface. Voir *Classificateurs composites et classificateurs internes* à la page 49.
Attribut	(sans objet)	(sans objet)	Propriété nommée d'une classe. Voir *Associations (MOO)* à la page 90.
Opération	(sans objet)	(sans objet)	Service qui peut être demandé par une classe. Voir *Opérations (MOO)* à la page 82.

Diagrammes de structures composites

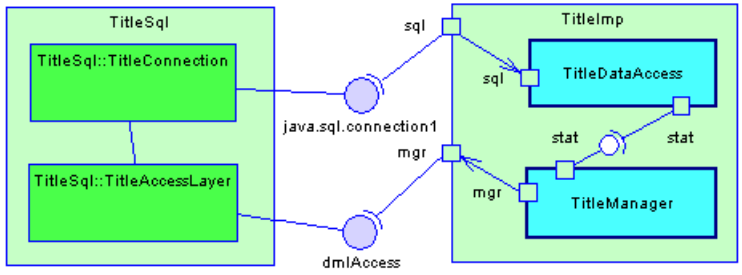
Un *diagramme de structures composites* est un diagramme UML qui fournit une représentation graphique des classes, interfaces et packages qui composent un système, en incluant les ports et parties qui décrivent leurs structures internes.

Remarque : Pour créer un diagramme de structures composites dans un MOO existant, pointez sur le modèle dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Diagramme de structures composites**. Pour créer un nouveau modèle, sélectionnez **Fichier > Nouveau modèle**, choisissez Modèle Orienté Objet comme type de modèle et **Diagramme de structures composites** comme premier diagramme, puis cliquez sur **OK**.

Un diagramme de structure composite joue le même rôle qu'un diagramme de classes, mais permet d'approfondir la description de la structure interne de plusieurs classes et la

représentation des interactions entre ces classes. Vous pouvez représenter de façon graphique les classes internes et les parties et montrer les associations à la fois entre et au sein des classes.

Dans l'exemple suivant, les structures internes des classes TitleSql (qui contient deux classes internes) et TitleImp (qui contient deux parties) sont connectées via les interfaces dmlAccess et java.sql.connection1 :



Objets du diagramme de structure composite

PowerAMC prend en charge tous les objets nécessaires pour construire des diagrammes de structure composite.

Objet	Outil	Symbole	Description
Classe		Compte	Jeu d'objets qui partagent les mêmes attributs, opérations, méthodes et relations. Voir *Classes (MOO)* à la page 37.
Interface		Signal	Descripteur des opérations visibles de l'extérieur pour une classe, un objet ou toute autre spécification de structure interne. Voir *Interfaces (MOO)* à la page 55.
Port			Point d'interaction entre un classificateur et son environnement. Voir *Ports (MOO)* à la page 65.
Partie		Ecriture	Instance de classificateur jouant un rôle particulier dans le contexte d'un autre classificateur. Voir *Parties (MOO)* à la page 63.
Généralisation			Lien entre classes montrant que la sous-classe partage la structure ou le comportement d'une ou plusieurs superclasses. Voir *Généralisations (MOO)* à la page 101.
Lien de prérequis			Connecte les classes, les composants, les ports, les parties et les interfaces. Voir *Liens de prérequis (MOO)* à la page 110.

Objet	Outil	Symbole	Description
Connecteur d'assemblage			Connecte les parties entre elles. Voir *Connecteurs d'assemblage (MOO)* à la page 112.
Connecteur de délégation			Connecte les parties aux ports à l'extérieur des classificateurs. Voir *Connecteurs de délégation (MOO)* à la page 114.
Association			Relation structurelle entre les objets de différentes classes. Voir *Associations (MOO)* à la page 90.
Agrégation			Forme d'association qui définit une relation tout-partie entre une classe composant et une classe agrégat (par exemple, une voiture a un moteur et des roues). Voir *Associations (MOO)* à la page 90.
Composition			Forme d'agrégation dans laquelle la notion de propriété et la coïncidence sont fortes entre les parties et le tout. Les parties vivent et meurent avec le tout (par exemple, une facture et un élément de cette facture). Voir *Associations (MOO)* à la page 90.
Dépendance			Relation entre deux éléments de modélisation, dans laquelle tout changement intervenant sur l'un des éléments est répercuté sur le second élément. Voir *Dépendances (MOO)* à la page 105.
Réalisation			Relation sémantique entre classificateurs, dans laquelle un classificateur spécifie un contrat que l'autre classificateur s'engage à remplir. Voir *Réalisations (MOO)* à la page 109.
Attribut	(sans objet)	(sans objet)	Propriété nommée d'une classe. Voir *Associations (MOO)* à la page 90.
Opération	(sans objet)	(sans objet)	Service qui peut être demandé par une classe. Voir *Opérations (MOO)* à la page 82.

Diagrammes de packages

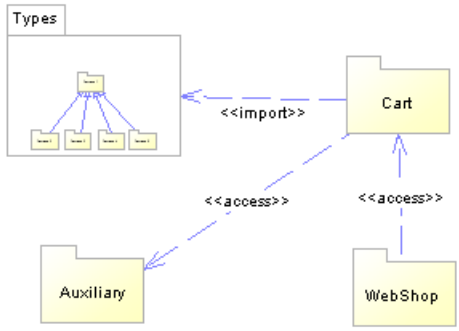
Un *diagramme de packages* est un diagramme UML qui fournit une représentation graphique de haut niveau de l'organisation de votre application, et vous aide à identifier les liens de généralisation et de dépendance entre les packages.

Remarque : Pour créer un diagramme de packages dans un MOO existant, pointez sur le modèle dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Diagramme de packages**. Pour créer un nouveau modèle, sélectionnez **Fichier >**

Nouveau modèle, choisissez Modèle Orienté Objet comme type de modèle et **Diagramme de packages** comme premier diagramme, puis cliquez sur **OK**.

Vous pouvez contrôler le niveau de détail fourni pour chaque package, en alternant les vues standard et composite grâce aux commandes du menu Edition ou des menus contextuels.

Dans l'exemple suivant, le package WebShop importe le package Cart qui, à son tour, importe le package Types et a accès au package Auxiliary. Le package Types est montré en vue composite (sous-diagramme) :



Objets du diagramme de packages

PowerAMC prend en charge tous les objets nécessaires pour construire des diagrammes de packages.

Objet	Outil	Symbole	Description
Package		Package_1	Conteneur permettant d'organiser vos objets de modèle. Voir *Packages (MOO)* à la page 53.
Généralisation			Lien entre packages montrant qu'un sous-package partage la structure ou le comportement défini dans un ou plusieurs packages parent. Voir *Généralisations (MOO)* à la page 101.
Dépendance			Relation entre deux packages, dans laquelle un changement dans un package affecte l'autre package. Voir *Dépendances (MOO)* à la page 105.

Diagrammes d'objets

Un *diagramme d'objets* est un diagramme UML qui fournit une représentation graphique de la structure d'un système via des instances concrètes de classes (objets), d'associations (liens entre objets), et de dépendances.

Remarque : Pour créer un diagramme d'objets dans un MOO existant, pointez sur le modèle dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Diagramme d'objets**. Pour créer un nouveau modèle, sélectionnez **Fichier > Nouveau modèle**, choisissez Modèle Orienté Objet comme type de modèle et **Diagramme d'objets** comme premier diagramme, puis cliquez sur **OK**.

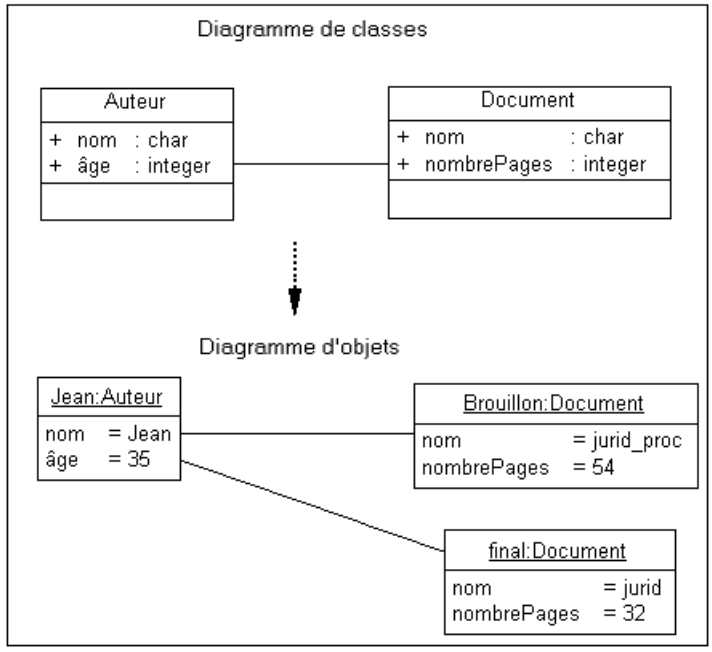
En tant que diagramme d'instances, le diagramme d'objets montre un exemple de structures de données avec des valeurs qui correspondent à une situation détaillée du système à un moment particulier.

Le diagramme d'objets peut être utilisé à des fins d'analyse : les contraintes entre classes qui ne sont pas classiquement représentées dans un diagramme de classes peuvent par contre être représentées dans un diagramme d'objets.

Si vous ne disposez pas encore de connaissances approfondies dans le domaine de la modélisation objet, l'utilisation d'instances est probablement plus "parlante" pour vous que celles de classificateurs, dans la mesure où les classificateurs représentent un certain degré d'abstraction. Le fait de regrouper plusieurs instances sous un même classificateur permet de mieux comprendre ce que sont les classificateurs. En outre, même pour les analystes habitués à l'abstraction, le diagramme d'objets peut permettre une meilleure compréhension de certaines contraintes structurelles qui ne peuvent pas être spécifiées de façon graphique dans un diagramme de classes.

A cet égard, le diagramme d'objets peut être considéré comme une version limitée du diagramme de classes. Dans l'exemple suivant, le diagramme de classes spécifie qu'une classe Auteur est liée à une classe Document.

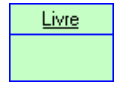
Dans le diagramme d'objets déduit de ce diagramme de classes, l'accent est mis sur les détails suivants : l'objet nommé Jean, instance de la classe Auteur est liée à deux objets différents (Brouillon et Final) qui sont deux instances de la classe Document.



Remarque : Vous pouvez faire glisser des classes et des associations depuis l'Explorateur d'objets dans un diagramme d'objets. Si vous faites glisser des classes, de nouveaux objets constituant des instances de classes sont créés. Si vous faites glisser une association, un nouveau lien entre objets (constituant une instance de l'association) et deux objets sont créés.

Objets du diagramme d'objets

PowerAMC prend en charge tous les objets nécessaires pour construire des diagrammes d'objets.

Objet	Outil	Symbole	Description
Objet			Instance d'une classe. Voir *Objets (MOO)* à la page 58.
Valeurs d'attribut	(sans objet)	(sans objet)	Une valeur d'attribut représente une instance d'un attribut de classe, cet attribut se trouvant la classe liée à l'objet. Voir *Propriétés d'un objet* à la page 60.
Lien entre objets			Lien de communication entre deux objets. Voir *Liens entre objets (MOO)* à la page 120.

Objet	Outil	Symbole	Description
Dépendance			Relation entre deux éléments de modélisation, dans laquelle tout changement intervenant sur l'un des éléments est répercuté sur le second élément. Voir *Dépendances (MOO)* à la page 105.

Classes (MOO)

Une *classe* est une description d'un ensemble d'objets qui ont une structure et un comportement similaires et qui partagent des attributs, opérations, relations et sémantiques.

Une classe peut être créée dans les types de diagramme suivants :

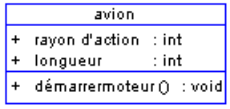
- Diagramme de classes
- Diagramme de structure composite

La structure d'une classe est décrite par ses *attributs* et *associations*, et son comportement est décrit par ses *opérations*.

Les classes, et les relations que vous créez entre elles, forment la structure de base d'un MOO. Une classe définit un concept au sein de l'application modélisée, par exemple :

- un élément physique (une voiture),
- un élément commercial (une commande),
- un élément logique (un planning de diffusion),
- un élément d'application (un bouton OK),
- un élément de comportement (une tâche)

L'exemple suivant montre une classe avion dotée d'attributs (rayon d'action et longueur) et d'opérations (démarrermoteur).



Création d'une classe

Vous pouvez créer une classe à partir d'une interface, ou à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Classe** dans la Boîte à outils.
- Sélectionnez **Modèle > Classes** pour afficher la boîte de dialogue Liste des classes, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Classe**.

- Pointez sur une interface, cliquez le bouton droit de la souris, puis sélectionnez **Créer une classe** dans le menu contextuel (cette méthode permet d'hériter de toutes les opérations de l'interface, y compris les opérations getter et setter, crée un lien de réalisation entre la classe et l'interface, et fait apparaître ce lien dans le sous-onglet **Réalise** de l'onglet **Dépendances** de la feuille de propriétés de la classe).

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'une classe

Pour visualiser ou modifier les propriétés d'une classe, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/ Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Etend	Spécifie la classe parent (la classe à laquelle la classe courante est liée par une généralisation). Cliquez sur l'outil Sélectionner un classificateur à droite de la zone pour spécifier une classe parent et cliquez sur l'outil Propriétés pour accéder à sa feuille de propriétés

Propriété	Description
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension. Les stéréotypes communs suivants sont disponibles par défaut : • <<actor>> - Groupe de rôles cohérent que l'utilisateur joue • <<enumeration>> - Liste de valeurs nommées utilisée comme fourchette pour un type d'attribut • <<exception>> - Classe d'exception. Utilisé principalement en relation avec les messages d'erreur • <<implementationClass>> - Classe dont les instances sont saisies de façon statique. Définit la structure de données physique et les méthodes d'une classe telle qu'utilisée dans les langages de programmation traditionnels • <<process>> - Flux qui peut être exécuté en même temps que d'autres processus • <<signal>> - Spécification d'un stimulus asynchrone communiqué entre instances • <<metaclass>> - Classificateur stéréotypé qui indique que la classe est une métaclasse d'une autre classe • <<powertype>> - Classificateur stéréotypé qui indique que la classe est une métaclasse dont les instances sont des sous-classes d'une autre classe • <<thread>> - Flux léger qui peut être exécuté en même temps que d'autres threads au sein du même processus. Généralement exécuté au sein de l'espace d'adresse du processus dans lequel il est inclus • <<type>> - Classe abstraite utilisée uniquement pour spécifier la structure et le comportement d'un ensemble d'objets, et non leur mise en oeuvre • <<utility>> - Classe dépourvue d'instance D'autres stéréotypes spécifiques à un langage objet peuvent être disponibles s'ils sont spécifiés dans le fichier du langage (voir *Personnalisation et extension de PowerAMC> Fichiers d'extension > Stéréotypes (Profile)*).
Visibilité	Spécifie la visibilité de l'objet, à savoir la façon dont il est perçu hors de son espace de noms. Lorsqu'une classe est visible par un autre objet, elle peut influencer la structure ou le comportement de l'objet, ou être affectée par ce dernier. Vous pouvez choisir l'une des valeurs suivantes : • Private – uniquement par l'objet • Protected – uniquement par l'objet et par ses objets hérités • Package – par tous les objets contenus dans le même package • Public – par tous les objets (option par défaut)

Propriété	Description
Cardinalité	Spécifie le nombre d'instances que peut avoir une classe. Vous pouvez choisir parmi les valeurs suivantes : • 0..1 – Aucune ou une • 0..* – Aucune ou un nombre illimité • 1..1 – Une à une • 1..* – Une à un nombre illimité • * – Nombre illimité
Type	Permet de spécifier qu'une classe est de type générique, ou si elle est liée à un type générique. Vous pouvez choisir parmi les valeurs suivantes : • Classe • Générique • Lié – une liste supplémentaire s'affiche et vous permet de spécifier le type générique auquel la classe est liée. Vous pouvez utiliser les outils à droite de la liste pour créer un objet, parcourir l'arborescence des objets disponibles ou afficher les propriétés de l'objet sélectionné Si vous spécifiez Générique ou Lié, l'onglet Générique s'affiche, permettant de contrôler les variables de type associées. Pour plus d'informations sur les types génériques et la liaison des classes à ces types, voir *Types et méthodes génériques* à la page 45.
Abstrait	Indique que la classe ne peut pas être instanciée et qu'elle n'est donc dotée d'aucune instance directe.
Final	Spécifie que la classe ne peut pas avoir d'objets hérités.
Générer du code	Indique si la classe est automatiquement incluse lorsque vous générez du code à partir du modèle, cela n'affecte en rien la génération intermodèle.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Onglet Détails

L'onglet Détails contient une zone de groupe Persistant qui sert à définir comme persistant le code généré pour une classe lors de la génération d'un MCD ou d'un MPD à partir d'un MOO, et il contient les propriétés suivantes :

Propriété	Description
Persistant	Spécifie que la classe doit être persistante dans un MCD ou un MPD généré. Vous devez sélectionner l'une des options suivantes : • Générer une table - la classe est générée dans une entité ou une table. • Migrer les colonnes - [MPD uniquement] la classe n'est pas générée, et ses attributs associations sont migrés vers la table parent ou enfant • Générer un type de parent abstrait - [MPD uniquement] la classe est générée comme un type de données abstrait (voir *Modélisation des données > Construction de modèles de données > Diagrammes physiques > Types de données abstraits (MPD)*). • Type de valeur – la classe n'est pas générée, et ses attributs sont générés dans leurs types référents. Pour plus d'informations, voir *Gestion de la persistance des objets lors de la génération de modèles de données* à la page 295.
Code	Spécifie le code de la table ou de l'entité qui sera générée à partir de la classe courante dans un MCD ou un MPD. Les codes persistants sont utilisés pour l'ingénierie par va-et-vient : une même classe génère toujours la même entité ou table avec un code conforme au SGBD cible. Exemple : pour générer une classe Acheteur dans une table ACHT, saisissez *ACHT* dans la zone Code.
Interne à	Spécifie le nom de la classe ou de l'interface à laquelle la classe courante appartient en tant que classificateur interne.
Classe d'association	Spécifie le nom de l'association liée à la classe pour former une classe d'association. Les attributs et opérations de la classe courante sont utilisés pour compléter la définition de l'association.

Les onglets suivants sont également disponibles :

- Attributs - répertorie et permet d'ajouter et de créer les attributs (y compris les accesseurs) associés à la classe (voir *Attributs (MOO)* à la page 69). Cliquez sur le bouton Héritées pour passer en revue les attributs publics et protégés hérités d'une classe parent.
- Identificateurs - répertorie et permet de créer les identificateurs associés à la classe (voir *Identifiants (MOO)* à la page 78).
- Opérations - répertorie et permet d'ajouter et de créer les opérations associées à la classe (voir *Opérations (MOO)* à la page 82).
- Générique - permet de spécifier les paramètres de type d'une classe générique ou des valeurs pour les paramètres de type requis pour une classe liée à un type générique (voir *Types et méthodes génériques* à la page 45

- Ports - répertorie et permet de créer les ports associés à la classe (voir *Ports (MOO)* à la page 65).
- Parties - répertorie et permet de créer les associées à la classe (voir *Parties (MOO)* à la page 63).
- Associations - répertorie et permet de créer les associations associées à la classe (voir *Associations (MOO)* à la page 90).
- Classificateurs internes - répertorie et permet de créer les classes et les interfaces internes associées à la classe (voir *Classificateurs composites et classificateurs internes* à la page 49).
- Diagrammes associés - répertorie et permet d'ajouter des diagrammes de modèles liés à la classe (voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Diagrammes, matrices et symboles > Diagrammes > Spécification de diagrammes comme diagrammes associés*).
- Script - permet de personnaliser le script de création de la classe (voir *Personnalisation des scripts de création d'objet* à la page 10)
- Aperçu - permet d'afficher le code à générer pour la classe (voir *Aperçu du code d'un objet* à la page 8)

Remarque : Si la classe est une classe de mise en oeuvre de service Web, voir aussi *Propriétés d'une classe de mise en oeuvre de service Web* à la page 252.

Création de classes Java BeanInfo

Si vous utilisez le langage objet Java, vous pouvez créer des classes Java BeanInfo à partir de n'importe quelle classe ayant le type "JavaBean".

Un JavaBean est un composant logiciel réutilisable qui peut être manipulé visuellement dans un outil de développement. Une classe Java BeanInfo est utilisée comme représentation standard d'un Bean. Chaque JavaBean peut mettre en oeuvre une classe BeanInfo. Les développeurs ayant recours à des Bean peuvent être amenés à fournir des informations explicites sur les méthodes, les propriétés et les événements d'un Bean en fournissant une classe Java BeanInfo.

La classe BeanInfo est générée avec un attribut, ainsi qu'avec les opérations suivantes :

- constructor
- getPropertyDescriptors();
- getMethodDescriptors();

Vous pouvez visualiser l'intégralité du code en cliquant sur l'onglet Aperçu dans la feuille de propriétés de la classe BeanInfo.

Attribut créé

L'attribut a le code suivant :

```
private static final Class <codeClasse>Class = <codeClasse>.class;
```

Opérations créées

Le code du constructeur est le suivant :

```
<codeClasse>BeanInfo()
{
   super();
 }
```

L'opération getPropertyDescriptors() a le code suivant :

```
public PropertyDescriptor[] getPropertyDescriptors ()
{
 // Déclaration du tableau de propriétés
 PropertyDescriptor properties[] = null;

 // Définition des propriétés
 try
 {
  // Création du tableau
  properties = new PropertyDescriptor[<nbPropriétés>];
  // Définition propriété 1
  properties[0] = new
PropertyDescriptor("<codePropriété1>" ,<codeClasse>Class;
  properties[0].setConstrained(false);
  properties[0].setDisplayName("nomPropriété1");
  properties[0].setShortDescription("commentairePropriété1");
  // Définition propriété 2
  properties[1] = new
PropertyDescriptor("<codePropriété2>" ,<codeClasse>Class;
  properties[1].setConstrained(false);
  properties[1].setDisplayName("nomPropriété2");
  properties[1].setShortDescription("commentairePropriété2");

 }
 catch
 {
  // Gestion des erreurs
 }
 return properties;
}
```

L'opération getMethodDescriptors() a le code suivant :

```
public MethodDescriptor[] getMethodDescriptors ()
{
 // Déclaration du tableau de méthode
 MethodDescriptor methods[] = null;
 ParameterDescriptor parameters[] = null;

 // Définition des méthodes
 try
 {
  // Création du tableau
  methods = new MethodDescriptor[<nbMéthodes>];
  // Set method 1
```

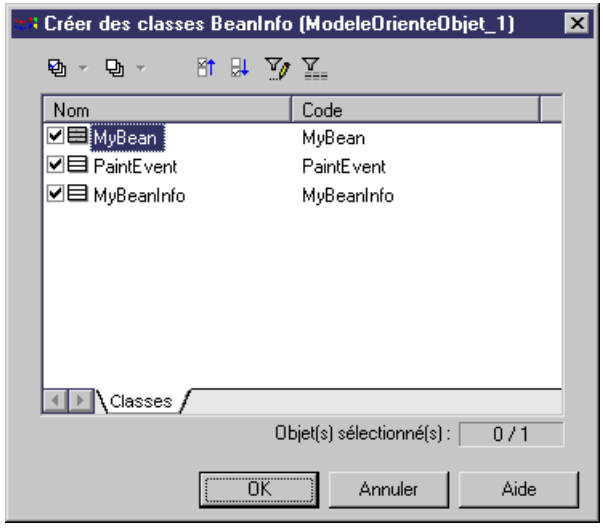
```
 parameters = new ParameterDescriptor[<nbParamètres1>];
 parameters[0] = new ParameterDescriptor();
 parameters[0].setName("codeParamètre1");
 parameters[0].setDisplayName("nomParamètre1");
 parameters[0].setShortDescription("ommentaireParamètre1");
 methods[0] = new MethodDescriptor("<methodCode1>", parameters);
 methods[0].setDisplayName("methodName1");
 methods[0].setShortDescription("methodComment1");
 // Set method 2
 methods[1] = new MethodDescriptor("<codeMéthode2>");
 methods[1].setDisplayName("nomMéthodeName2");
 methods[1].setShortDescription("commentaireMéthode2");

}
catch
{
 // Gestion des erreurs
}
return methods;
}
```

Lorsque vous créez une classe Java BeanInfo, un lien de dépendance est automatiquement créé entre les classes et le stéréotype de la classe Java BeanInfo est défini à <<BeanInfo>>.

Création d'une classe Java BeanInfo à partir du menu Langage

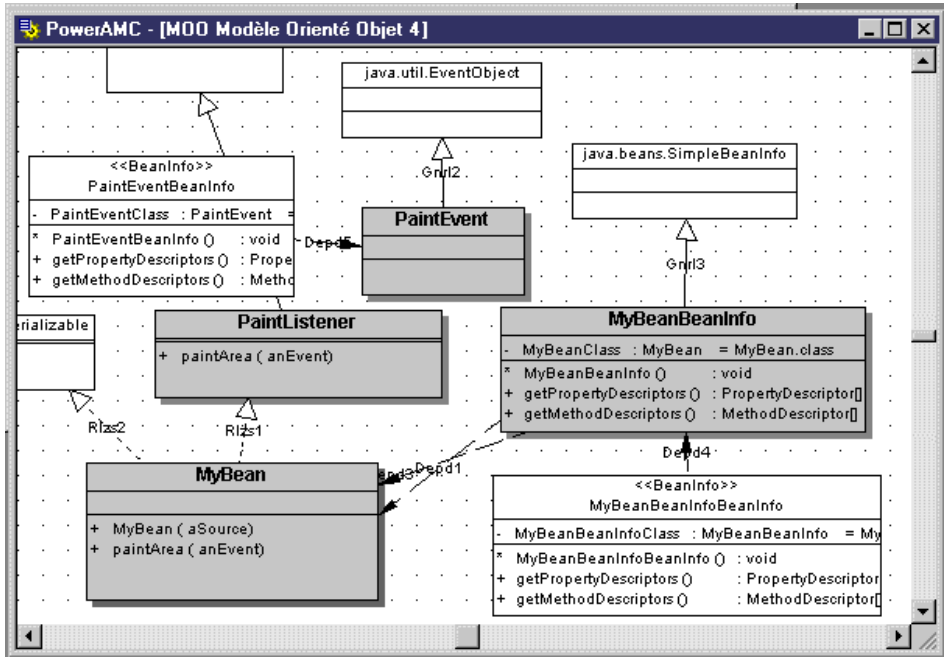
Vous pouvez créer une classe Java BeanInfo à partir du menu Langage.

1. Sélectionnez **Langage > Créer des classes BeanInfo** pour afficher la fenêtre de sélection Créer des classes BeanInfo. Elle contient une liste de toutes classes du modèle qui sont définies avec le type Java Bean.



2. Sélectionnez les classes pour lesquelles vous souhaitez créer des classes Java BeanInfo, puis cliquez sur OK.

Une classe BeanInfo est créée dans le modèle pour chaque classe sélectionnée.



Création d'une classe Java BeanInfo à partir du menu contextuel d'une classe

Vous pouvez créer une classe Java BeanInfo à partir du menu contextuel d'une classe. Pointez sur un symbole de classe dans le diagramme et cliquez le bouton droit de la souris, puis sélectionnez Créer une classe BeanInfo dans le menu contextuel.

Types et méthodes génériques

Les types et méthodes génériques sont une nouvelle fonctionnalité de Java 5.0. Un type générique est une classe ou une interface qui a une ou plusieurs variables de type et une ou plusieurs méthodes qui utilisent une variable de type comme emplacement pour un argument ou un type de résultat.

L'utilisation de types et de méthodes génériques permettent de tirer parti d'une vérification plus poussée des types lors de la compilation. Lorsqu'un type générique est utilisé, un type réel est spécifié pour chaque variable de type. Cette information de type supplémentaire est utilisée par le compilateur pour distribuer automatiquement les valeurs de résultat associées.

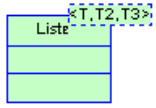
Création de types génériques

PowerAMC permet de définir des classes et des interfaces sous forme de type génériques.

Vous définissez une liste de variables de type qui seront utilisés comme types de données pour les attributs, les paramètres de méthode ou les types de résultats. PowerAMC requiert l'existence d'une classe liée pour créer une généralisation, une réalisation ou une association.

Ensuite, vous liez un classificateur au type générique via cette classe liée intermédiaire, et spécifiez les types réels à utiliser en lieu de place des variables de type requises.

1. Affichez la feuille de propriétés de la classe ou de l'interface, puis sélectionnez Générique dans la liste Type sur l'onglet Général. L'onglet Générique est automatiquement ajouté, et une variable de type est créée dans la liste de cet onglet.
2. Sur l'onglet Générique, et ajoutez des variables de type supplémentaires requises en utilisant l'outil Ajouter une ligne. Vous pouvez également spécifier une contrainte de dérivation sous la forme d'une liste de types.
3. Cliquez sur OK pour revenir au diagramme. Le symbole de classificateur affiche maintenant les variables de type dans son angle supérieur gauche.



Pour que le classificateur devienne un vrai type générique, il doit contenir au moins une méthode générique.

Création de méthodes génériques

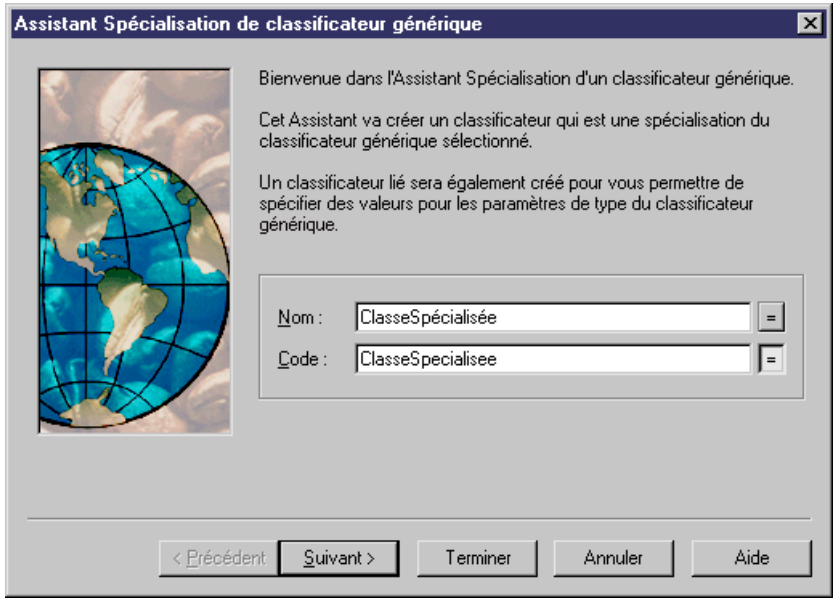
PowerAMC permet de définir des opérations comme méthodes génériques. Les méthodes génériques sont des méthodes qui disposent de leur propre liste de variables de type.

1. Affichez la feuille de propriétés, d'une classe ou d'une interface, puis cliquez sur l'onglet Opérations.
2. Cliquez sur l'outil Ajouter une ligne pour créer une nouvelle opération, puis cliquez sur l'outil Propriétés pour afficher sa feuille de propriétés.
3. Cliquez sur Oui pour confirmer la création de l'opération, puis cochez la case Générique sur l'onglet Général de la feuille de propriétés de la nouvelle opération afin de définir l'opération comme méthode générique. L'onglet Générique est automatiquement ajouté, et une variable de type est créée dans la liste de cet onglet.
4. Les cas échéant, ajoutez les éventuels variables de type supplémentaires en utilisant l'outil Ajouter une ligne et en cliquant sur OK.

Création d'un classificateur spécialisé

Si vous devez créer un classificateur qui va hériter d'un type générique, vous devez créer un classificateur lié intermédiaire. L'Assistant Spécialisation d'un classificateur générique peut effectuer les opérations nécessaires pour vous.

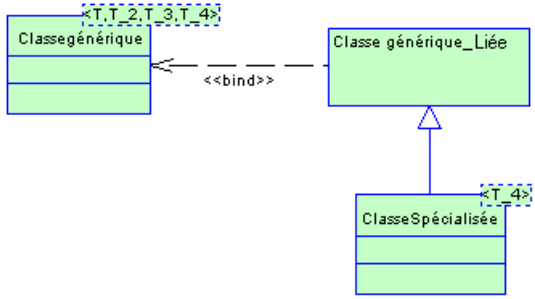
1. Pointez sur une classe ou une interface générique, cliquez le bouton droit de la souris, puis sélectionnez Créer une classe (ou une interface) personnalisée dans le menu contextuel pour lancer l'Assistant Spécialisation d'un classificateur générique :



2. Saisissez un nom et un code pour le classificateur spécialisé, puis cliquez sur Suivant pour passer à l'onglet des paramètres de type.
3. Spécifiez des valeurs pour chacun des paramètres de type dans la liste. Si vous ne spécifiez pas de valeur pour un paramètre de type, ce dernier sera ajouté comme un paramètre de type au nouveau classificateur spécialisé.
4. Cliquez sur Terminer pour revenir au diagramme. L'Assistant a créé le classificateur spécialisé ainsi qu'un classificateur lié qui agit comme un intermédiaire entre le classificateur générique et le classificateur spécialisé, afin de spécifier des valeurs pour les paramètres de type.

   Le classificateur lié est attaché au classificateur générique via une dépendance ayant le stéréotype <<bind>>, et agit comme le parent du classificateur spécialisé, auquel il est lié par le biais d'une généralisation.

   Dans l'exemple ci-dessous, Classe spécialisée hérite de Classe générique via Classe générique_Liée, qui spécifie les paramètres de type pour les types génériques T, T_2 et T_3.



Au moment de la compilation, le classificateur spécialisé peut hériter de méthodes et propriétés du classificateur générique, et les variables de type générique seront remplacées par les types réels. Le compilateur sera alors en mesure de réaliser une vérification plus poussée et une diffusion automatique des valeurs de résultats associées.

Création d'un classificateur lié

Vous pouvez être amené à lier un classificateur à un classificateur générique sans créer de classificateur spécialisé. L'Assistant Création d'un classificateur lié peut effectuer les opérations nécessaires pour vous.

1. Pointez sur une classe ou une interface générique, cliquez le bouton droit de la souris, puis sélectionnez Créer une classe (ou une interface) liée dans le menu contextuel pour lancer l'Assistant Création d'un classificateur lié.
2. L'Assistant crée le classificateur lié, qui est attaché au classificateur générique via une dépendance ayant le stéréotype <<bind>>.

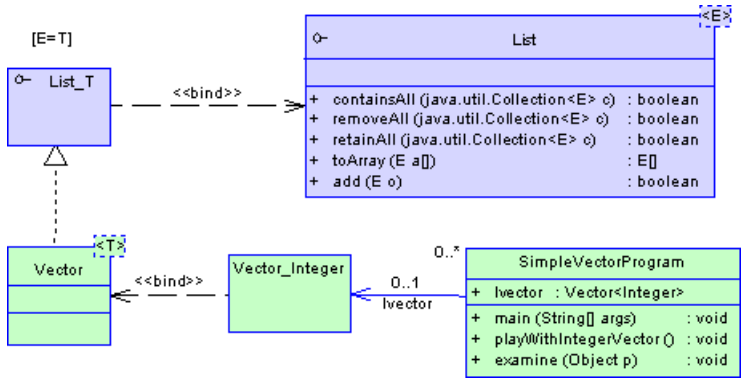
Exemple de type générique

Dans l'exemple ci-dessous, l'interface liée , List_T, spécifie un type 'T' pour le paramètre de type <E> de List.

La classe générique Vector<T> réalise l'interface générique List<E> (via l'interface liée List_T) avec un type <T> (qui est défini dans sa propre définition générique) :

```
public class vector <T> implements List <E>
```

La classe liée Vector_Integer spécifie un type 'Integer' pour le paramètre de type <T> de Vector<T>. La classe SimpleVectorProgram est associée à Vector_Integer, ce qui lui permet d'utiliser un type de données d'attribut que la classe Vector définit comme Integer.

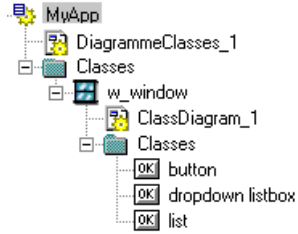


Vous devez créer une classe liée pour une généralisation ou pour une réalisation. Toutefois, on peut avoir spécifié une valeur de paramètre pour le type générique <T> directement (sans créer de classe liée) comme un type de données d'attribut, un type de données de paramètre ou un type de données de résultats, simplement en saisissant l'expression suivante dans la zone Type de SimpleVectorProgram :

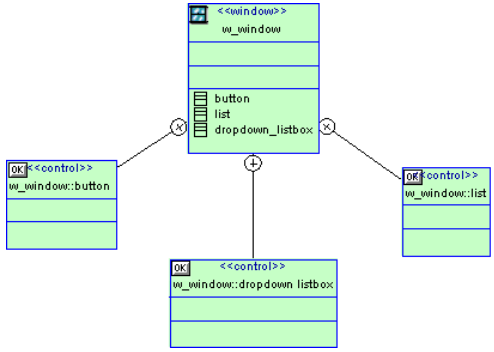
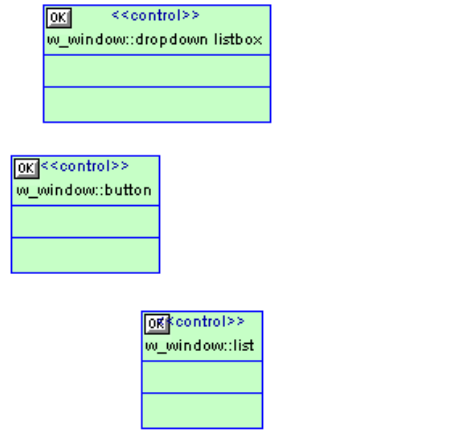
```
Vector<integer>
```

Classificateurs composites et classificateurs internes

Un classificateur composite est une classe ou une interface qui contient d'autres classes ou interfaces (appelées classificateurs internes). Les classificateurs internes sont répertoriés dans l'Explorateur d'objets sous forme d'enfant de leur classificateur composite.



Les classificateurs internes peuvent être affichés directement dans le diagramme de classes principal, ou dans un diagramme de classes supplémentaire spécifique au classificateur composite :

Diagramme principal (avec des liens internes)	Diagramme de classificateur composite
	

Remarque : Vous pouvez afficher plusieurs classificateurs composites dans un diagramme de structures composites (voir *Diagrammes de structures composites* à la page 31).

Remarque : dans les versions précédentes de PowerAMC, les classificateurs composites n'existaient pas et les classificateurs internes apparaissaient au même niveau hiérarchique que leur parent dans l'Explorateur d'objets ainsi que dans la liste des classificateurs.

Création de classificateurs internes

Vous pouvez créer des classificateurs internes dans une classe ou dans une interface.

- Affichez l'onglet **Classificateurs internes** d'une feuille de propriétés de classe ou d'une interface, puis cliquez sur l'outil **Ajouter une classe interne** ou **Ajouter une interface interne**.
- Pointez sur une nouvelle classe ou interface dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Classe** ou **Nouveau > Interface**.
- Sélectionnez l'outil **Lien interne** dans la Boîte à outils, puis tracez un lien entre les deux classes dans le diagramme. La première classe devient une classe composite et la seconde un classificateur interne.
- Créez un diagramme de classificateur composite dédié à la classe (voir la section suivante), puis créez des classes ou des interfaces dans ce diagramme.

Tous les classificateurs internes sont répertoriés en bas du symbole de classe.

Création d'un diagramme de classificateur composite

Vous pouvez être amené à créer un diagramme pour montrer la structure interne d'un classificateur composite. Vous pouvez créer un diagramme de classificateur composite de l'une des façons suivantes :

- Pointez sur une nouvelle classe ou interface dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Diagramme de classes**.
- Maintenez la touche **Ctrl** enfoncée et double-cliquez sur une classe ou interface dans le diagramme.

Le diagramme de classificateur est vide par défaut et ce, même si le classificateur composite comporte déjà des classificateurs internes. Vous pouvez créer des symboles pour les classificateurs internes en sélectionnant **Symbole > Afficher les symboles**, ou en les faisant glisser depuis l'Explorateur d'objets dans le diagramme.

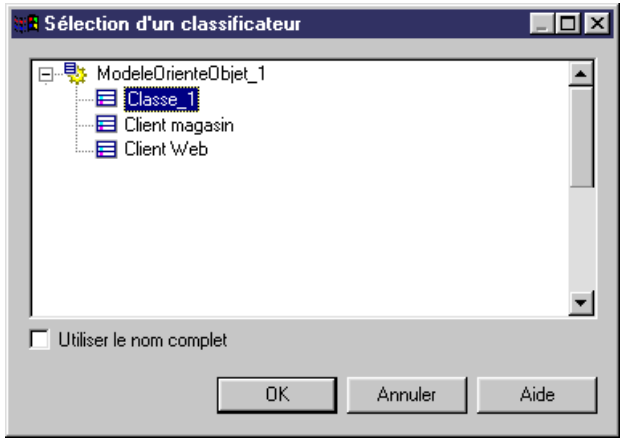
Si vous avez créé un diagramme de classificateur composite, vous pouvez l'afficher dans le diagramme de classes principal. Pour ce faire, pointez sur le symbole de classe, cliquez le bouton droit de la souris, puis sélectionnez **Vue composite > Lecture seule (sous-diagramme)**.

Spécification d'un classificateur comme type de données ou type de résultat

Vous pouvez spécifier une classe ou interface du modèle courant ou d'un autre modèle (incluent un bibliothèque JDK) comme type de données d'attribut ou de paramètre ou comme type de résultat d'opération. Si le classificateur appartient au modèle ou package courant, il est affiché avec les autres classificateurs. S'il appartient à un autre modèle ou package, un raccourci vers ce classificateur est créé dans le package courant.

Remarque : Pour plus d'informations sur la génération des classificateurs liés de cette manière, voir *Gestion de la persistance pour les types de données complexes* à la page 297.

1. Ouvrez la feuille de propriétés de l'objet approprié :
   - Pour spécifier un classificateur sous forme de type de données d'attribut, affichez la feuille de propriétés de l'attribut (voir *Propriétés d'un attribut* à la page 73).
   - Pour spécifier un classificateur sous forme de type de résultat d'opération, affichez la feuille de propriétés de l'opération (voir *Propriétés d'une opération* à la page 85).
   - Pour spécifier un classificateur sous forme de type de données de paramètre d'opération, affichez la feuille de propriétés de l'opération (voir *Paramètres (MOO)* à la page 89).
2. Sur l'onglet **Général**, cliquez sur l'outil **Sélectionner un classificateur** à droite de la zone **Type de données**, puis sélectionnez un classificateur dans la liste, qui répertorie tous les classificateurs disponibles dans tous les modèles ouverts dans l'espace de travail.



Remarque : Sélectionnez l'option **Utiliser le nom complet** pour inclure la hiérarchie de packages menant au classificateur.

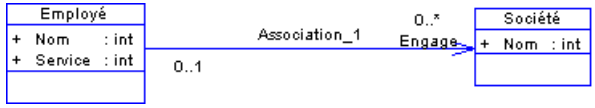
3. Cliquez sur **OK** pour spécifier le classificateur comme type de données.

Remarque : Vous pouvez également saisir le code du classificateur directement dans la zone **Type de données**. Pour saisir un nom complet, utilisez le point comme séparateur (par exemple `Fabrication.Usine.Chaîne`.

Affichage des attributs migrés d'une classe

Les associations navigables migrent les attributs vers les classes lors de la génération du code. Vous avez la possibilité d'afficher ces attributs migrés dans l'onglet Associations d'une feuille de propriétés de classe.

Dans l'exemple suivant, la classe Employé est associée à la classe Société.



Si vous affichez un aperçu du code généré pour la classe Employé, vous voyez les attributs suivants (langage Java) :

```
public class EMPLOYEE
{
 public COMPANY hires[];
 public int NAME;
 public int DEPARTMENT;
}
```

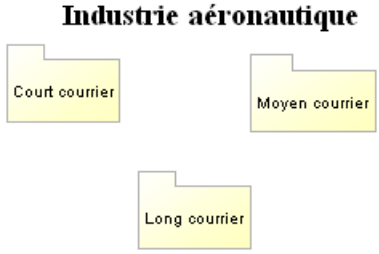
L'association entre Employé et Société est migrée sous la forme de l'attribut `public SOCIETE engage []`.

Vous pouvez utiliser l'onglet **Associations** de la feuille de propriétés d'une classe pour afficher la liste de tous les attributs migrés provenant des associations navigables.

Packages (MOO)

Un package est un mécanisme général permettant d'organiser des éléments en groupes. Il contient des objets de modèle et peut être créé dans tous les diagrammes.

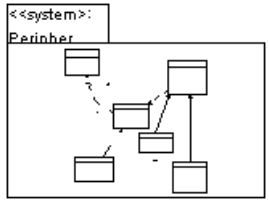
Lorsque vous gérez des modèles de grande taille, il peut être utile de les répartir dans des unités de moindre importance afin d'éviter de manipuler de trop grandes quantités d'éléments. Les packages sont utilisés pour affecter des portions de modèles, représentant différentes tâches ou différents domaines d'activité, à des équipes de développement distinctes.



Vous pouvez créer plusieurs packages au même niveau hiérarchique dans un modèle ou décomposer un package en d'autres packages et ce, sans limitation de niveau de décomposition. Chaque package fait l'objet d'une fenêtre de diagramme par défaut. A chaque niveau de décomposition vous pouvez créer plusieurs diagrammes.

Remarque : Dans les diagrammes d'activités ou d'états-transitions, vous ne créez pas des packages mais décomposez des activités et des états, qui agissent comme des packages dans ce contexte.

Vous pouvez développer un package pour afficher son contenu en en pointant sur son symbole, en cliquant le bouton droit de la souris, puis en sélectionnant **Vue composite > Lecture seule (sous-diagramme)**. Vous pouvez être amené à redimensionner le symbole pour afficher tout son contenu. Double-cliquez sur le symbole composite pour aller dans le diagramme de package.



Pour revenir au symbole standard, pointez sur le symbole, cliquez le bouton droit de la souris, puis sélectionnez **Vue composite > Aucune**.

Propriétés d'un package de MOO

Les packages sont dotés de propriétés qui s'affichent sur leur page de propriétés respective. Tous les packages ont en commun les propriétés suivantes :

Propriété	Description
Nom/Code/ Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Sous-classification dérivée d'un package existant. Les stéréotypes suivants sont disponibles par défaut : • <<archive>> Archive Jar ou archive Bar (Java uniquement). • <<assembly>> – Indique qu'un package produit un fichier exécutable portable (PE), (C# et VB.NET uniquement). • <<CORBAModule>> – Package UML identifié comme étant un module IDL (IDL-CORBA uniquement). • <<facade>> – Le package est une vue d'un autre package. • <<framework>> – Le package est principalement composé de motifs. • <<metamodel>> – Le package est une abstraction d'un autre package. • <<model>> – Spécifie une abstraction sémantiquement fermée d'un système. • <<stub>> – Le package sert de proxy pour le contenu public d'un autre package. • <<subsystem>> – Regroupement d'éléments, dont certains constituent une spécification du comportement offert par les éléments contenus. • <<system>> – Le package représente l'intégralité du système en cours de modélisation. • <<systemModel>> – Le package contient d'autres packages avec le même système physique. Il contient également toutes les relations entre les éléments de modèle contenus dans différents modèles. • <<topLevel>> – Indique le package de plus au niveau dans une hiérarchie.
Diagramme par défaut	Diagramme affiché par défaut lorsque vous ouvrez ce modèle.

Propriété	Description
Utiliser l'espace de noms du parent	[package uniquement] Spécifie que le package ne représente pas un espace de noms distinct de celui de son parent et donc que les objets créés dans ce package doivent avoir un nom unique dans le conteneur parent. Si vous ne sélectionnez pas cette propriété, le package et son package ou modèle parent peuvent tous les deux contenir des classes appelées Classe A.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Définition du type de diagramme d'un nouveau package

Lorsque vous créez un nouveau package, le diagramme par défaut du package est défini en fonction de divers paramètres.

- Si vous avez créé un package à l'aide de l'outil **Package** de la Boîte à outils, le diagramme est du même type que le package ou modèle parent.
- Si vous créez un package à partir de l'Explorateur d'objets, le diagramme est du même type que les diagrammes existants dans le package ou modèle parent, si ces diagrammes sont du même type. Si les diagrammes du package ou modèle parent ne sont pas du même type, vous êtes invité à sélectionner le type de diagramme pour le nouveau package ou sous-package.
- Si vous créez un package à partir de la boîte de dialogue Liste des packages, le diagramme est du même type que celui du diagramme actif.

Interfaces (MOO)

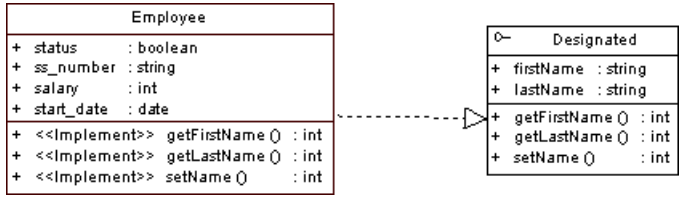
Une interface est similaire à une classe, mais elle est utilisée pour définir la spécification d'un comportement. Une interface est une collection d'opérations utilisée pour spécifier le comportement visible d'une classe. L'interface elle-même n'est pas mise en oeuvre.

Une interface peut être créée dans les types de diagramme suivants :

- Diagramme de classes
- Diagramme structure composite
- Diagramme de composants

Une interface inclut les signatures des opérations. Elle spécifie uniquement une partie limitée du comportement d'une classe. Une classe permet de mettre en oeuvre une ou plusieurs interfaces.

Une classe doit mettre en oeuvre toutes les opérations dans une interface pour réaliser cette interface. L'exemple suivant montre une interface (Designated) réalisée par une classe (Employee).



Création d'une interface

Vous pouvez créer une interface à partir d'une classe, ou à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Sélectionnez l'outil **Interface** dans la Boîte à outils.
- Sélectionnez **Modèle > Interfaces** pour afficher la boîte de dialogue Liste des interfaces, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Interface**.
- Pointez sur une classe, cliquez le bouton droit de la souris, puis sélectionnez **Créer une interface** dans le menu contextuel (cette méthode permet d'hériter de toutes les opérations de la classe, y compris les opérations getter et setter, crée un lien de réalisation entre l'interface et la classe, et montre ce lien dans le sous-onglet **Réalise** de l'onglet **Dépendances** dans la feuille de propriétés de l'interface).

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'une interface

Pour visualiser ou modifier les propriétés d'une interface, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Etend	Indique le nom de la classe ou de l'interface que l'interface courante étend.

Propriété	Description
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension. Les stéréotypes communs suivants sont disponibles par défaut : • <<metaclass>> - interface qui va interagir avec un modèle qui contient des classes ayant des stéréotypes metaclass. • <<powertype>> - métaclasse dont les instances sont des sous-classes d'une autre classe. • <<process>> - flux lourd qui peut être exécuté en même temps que d'autres processus. • <<thread>> - flux léger qui peut être exécuté en même temps que d'autres threads au sein du même processus. Généralement exécuté au sein de l'espace d'adresse du processus dans lequel il est inclus. • <<utility>> - classe dépourvue d'instance.
Visibilité	Spécifie la visibilité de l'objet, à savoir la façon dont il est perçu hors de son espace de noms. Une interface qui est visible d'un autre objet peut influer sur la structure ou le comportement de cet objet et, de la même manière, ses propres propriétés peuvent être affectées par cet objet. Vous pouvez choisir parmi les valeurs suivantes : • Private – uniquement par l'objet. • Protected – uniquement par l'objet et par ses objets hérités. • Package – par tous les objets contenus dans le même package. • Public – par tous les objets (option par défaut).
Interne à	Indique le nom de la classe ou de l'interface à laquelle la classe courante appartient en tant que classificateur interne.
Type	Permet de spécifier qu'une interface est de type générique, ou si elle est liée à un type générique. Vous pouvez choisir parmi les valeurs suivantes : • Classe • Générique • Lié – Si vous sélectionnez cette option, une liste supplémentaire devient disponible à droite, dans laquelle vous pouvez spécifier le type générique auquel l'interface est liée. Si vous spécifiez Générique ou Lié, l'onglet Générique s'affiche, permettant de contrôler les variables de type associées (voir *Types et méthodes génériques* à la page 45).
Générer du code	Indique si l'interface est automatiquement incluses dans les objets générés depuis le modèle lorsque vous générez du code à partir du modèle.

Propriété	Description
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Les onglets suivants répertorient les objets associés à l'interface :

- Attributs - répertorie les attributs associés à l'interface. Vous pouvez créer des attributs directement sur cet onglet, ou bien ajouter des attributs existants. Pour plus d'informations, voir *Attributs (MOO)* à la page 69.
- Opérations - répertorie les opérations associées à l'interface. Vous pouvez créer des opérations directement dans cet onglet, ou ajouter des opérations existantes. Pour plus d'informations, voir *Opérations (MOO)* à la page 82.
- Paramètres génériques - permet de spécifier les paramètres de type d'une interface générique, ou bien les valeurs pour les paramètres de type requis pour une interface liée à un type générique (voir *Types et méthodes génériques* à la page 45
- Classificateurs internes - répertorie les classes et interfaces internes associés à l'interface. Vous pouvez créer des classificateurs internes directement sur cet onglet. Pour plus d'informations, voir *Classificateurs composites et classificateurs internes* à la page 49.
- Diagrammes associés - répertorie et permet d'ajouter des diagrammes de modèles liés à l'interface (voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Diagrammes, matrices et symboles > Diagrammes > Spécification de diagrammes comme diagrammes associés*.

Objets (MOO)

Au niveau conceptuel, un *objet* est un élément défini comme faisant partie du système décrit. Il représente un objet qui n'a pas encore été instancié car à ce stade les classes ne sont pas encore clairement définies.

Si vous devez poursuivre plus avant la mise en oeuvre de votre modèle, l'objet qui a émergé lors de l'analyse se transformera probablement en instance d'une classe définie. Dans ce cas, un objet sera considéré comme une instance d'une classe.

Les trois situations suivantes peuvent se présenter :

- Lorsqu'un objet n'est pas une instance d'une classe - il n'a qu'un nom.
- Lorsqu'un objet est une instance d'une classe - il a un nom et une classe.
- Lorsqu'un objet est une instance d'une classe mais qu'il représente tout ou partie des instances de la classe - il a une classe mais pas de nom.

Un objet peut être créé dans les types de diagramme suivants :

- Diagramme de communication
- Diagramme d'objets

- Diagramme de séquence

Un objet à la même définition dans les diagrammes d'objet, de séquence et de communication. Vous pouvez soit le créer directement dans le type de diagramme de votre choix, soit le faire glisser d'un type de diagramme à l'autre.

Définition de multiples

Un *multiple* définit un jeu d'instances. Il s'agit d'une représentation graphique d'un objet qui représente plusieurs instances. Un objet peut communiquer avec un autre objet qui est multiple. Cette fonctionnalité est principalement utilisée dans le diagramme de communication mais peut également être utilisée dans le diagramme d'objets.

Un employé gère une liste de documents : c'est la liste des documents qui représente un objet multiple.

Lorsque la case Multiple est cochée dans la feuille de propriétés de l'objet, un symbole spécifique représentant deux rectangles superposés s'affiche.

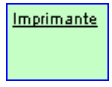
Objets dans un diagramme d'objets

Dans le diagramme d'objets, un objet instance d'une classe peut afficher les valeurs des attributs définis sur la classe. Lorsque la classe est supprimée, les objets associés ne sont pas automatiquement supprimés.

Objets dans un diagramme de communication

Dans un diagramme de communication, un *objet* est une instance d'une classe. Il peut être persistant ou transitoire : un objet persistant est un objet qui continue d'exister une fois que le processus qui l'a créé est terminé, un objet transitoire est un objet qui cesse d'exister une fois que le processus qui l'a créé est terminé.

Le nom de l'objet s'affiche souligné. Le soulignement indique traditionnellement qu'un élément est une instance d'un autre élément.



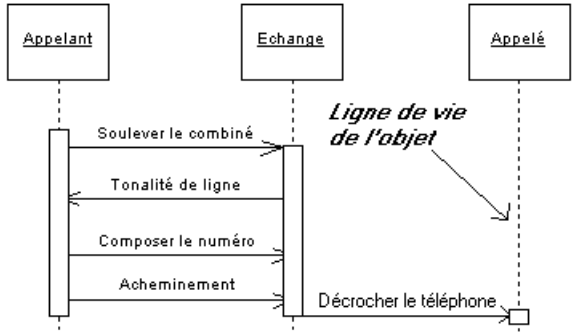
Objets dans un diagramme de séquence

Dans un diagramme de séquence, un objet a une *ligne de vie d'objet* qui est représentée par une ligne verticale en tirets sous le symbole de l'objet. Le temps s'écoule toujours vers le bas de la page. La ligne de vie d'objet indique la période durant laquelle un objet existe. Vous ne pouvez pas séparer un objet de sa ligne de vie.

Si l'objet est créé ou supprimé lors de la période représentée sur le diagramme, sa ligne de vie commence ou s'arrête au point correspondant.

Les objets s'affichent en haut du diagramme. Ils échangent des messages entre eux.

Un objet qui existe quand une transaction ou un message commence est affiché en haut du diagramme, au-dessus de la première flèche de message. La ligne de vie d'un objet qui existe encore après la fin de la transaction se poursuit au-delà de la flèche de fin du message.



Création d'un objet

Vous pouvez créer un objet à partir de l'Explorateur d'objets ou du menu **Modèle**.

- Sélectionnez l'outil **Objet** dans la Boîte à outils.
- Sélectionnez **Modèle > Objets** pour afficher la boîte de dialogue Liste des objets, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Objet**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'un objet

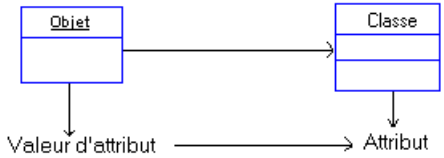
Pour visualiser ou modifier les propriétés d'un objet, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**. Cette propriété n'est pas obligatoire puisque vous pouvez faire en sorte qu'un objet représente une instance non nommée d'une classe ou d'une interface, mais si la zone nom ou code est vide, vous devez spécifier le nom de la classe ou de l'interface dans la zone **Classificateur**. Les noms doivent être uniques pour un classificateur.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Classificateur	Spécifie la classe ou l'interface dont l'objet est une instance. Vous pouvez lier un objet à une classe ou à une interface, ou bien créer une classe à l'aide de l'outil Créer une classe situé en regard de la zone (voir *Liaison d'un classificateur à un objet* à la page 62).
Multiple	Spécifie que l'objet représente plusieurs instances.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

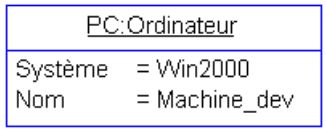
Onglet Valeurs d'attribut

Une valeur d'attribut est une instance d'un attribut de classe provenant de la classe dont l'objet est une instance, ou d'un attribut hérité d'un parent de la classe.



Vous pouvez ajouter des attributs de classe à l'objet et leur affecter des valeurs sur l'onglet Valeurs d'attribut en utilisant l'outil **Ajouter des valeurs d'attribut**, qui affiche une boîte de dialogue qui répertorie tous les attributs de la classe de l'objet, y compris les attributs hérités des classes dont cette classe hérite. Une fois l'attribut ajouté à l'objet, il est possible de définir sa valeur dans la colonne Valeur de la liste. Toutes les autres colonnes sont en lecture seule.

Vous pouvez contrôler l'affichage des valeurs d'attribut sur les symboles d'objet en utilisant les préférences d'affichage (**Outils > Préférences d'affichage**):



Liaison d'un classificateur à un objet

Le diagramme d'objets représente les instances d'une classe ou d'une interface, le diagramme de séquence représente le comportement dynamique d'une classe ou d'une interface et le diagramme de communication représente ces instances dans un mode de communication. Pour toutes ces raisons, vous pouvez lier une classe ou une interface à un objet dans un MOO.

Dans la feuille de propriétés de l'objet vous pouvez :

- Lier l'objet à une classe ou à une interface existante
- Créer une classe

1. Sélectionnez une classe ou une interface dans la liste Classificateur située dans la feuille de propriétés de l'objet.

   ou

   Cliquez sur l'outil Créer une classe en regard de la zone de liste Classe pour créer une classe et afficher sa feuille de propriétés.

   Définissez les propriétés de la nouvelle classe puis cliquez sur OK.

   Le nom de la classe ou de l'interface s'affiche dans la liste Classificateur.
2. Cliquez sur OK.

   Le nom de l'objet s'affiche alors dans le diagramme de séquence suivi du signe deux points et du nom de la classe ou de l'interface sélectionnée.

   Vous pouvez également afficher le nom de l'objet dans la feuille de propriétés de la classe ou de l'interface. Pour ce faire, cliquez sur l'onglet Dépendances, puis sur le sous-onglet Objets. Le nom de l'objet est automatiquement ajouté dans l'onglet Objets.

Remarque : Vous pouvez faire glisser le noeud d'une classe ou d'une interface depuis l'Explorateur d'objets dans un diagramme de séquence, de communication ou d'objets. Vous avez également la possibilité de copier une classe ou une interface puis de la coller, ou de la coller en tant que raccourci, dans l'un de ces diagrammes. Vous créez alors automatiquement un objet qui constitue une instance de la classe ou de l'interface.

Parties (MOO)

Une partie permet de définir une zone bien délimitée à l'intérieur d'une classe ou d'un composant. Les parties peuvent être connectées aux autres parties ou ports, soit directement, soit via un port situé à l'extérieur de la partie.

Vous connectez les parties entre elles via des connecteurs d'assemblage. Vous connectez une partie à un port situé à l'extérieur d'une classe ou d'un composant en utilisant un connecteur de délégation.

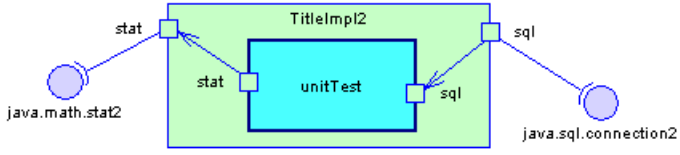
Une partie peut être créée dans les types de diagramme suivants :

- Diagramme de structure composite (au sein d'une classe)
- Diagramme de composants (au sein d'un composant)

Vous ne pouvez créer une partie qu'au sein d'une classe ou d'un composant. Si vous tentez de faire glisser une partie hors de son classificateur, ce dernier est agrandi de façon à continuer à contenir cette partie.

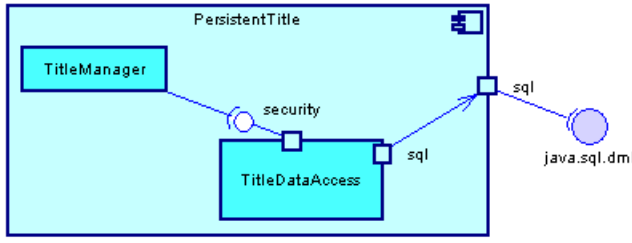
Parties dans un diagramme de structure composite

Dans l'exemple ci-dessous, la classe TitleImpl2 contient une partie appelée unitTest :



Parties dans un diagramme de composants

Dans l'exemple ci-dessous, le composant PersistentTitle contient deux parties, TitleManager et TitleDataAccess :



Création d'une partie

Vous pouvez créer une partie à partir de la Boîte à outils, ou depuis l'onglet **Parties** de la feuille de propriétés d'une classe ou d'un composant.

- Utilisez l'outil **Partie** dans la Boîte à outils.
- Affichez l'onglet **Parties** de la feuille de propriétés d'une classe ou d'un composant, puis cliquez sur l'outil **Ajouter une ligne**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'une partie

Pour visualiser ou modifier les propriétés d'une partie, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Parent	Spécifie l'objet parent.
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Visibilité	Spécifie la visibilité de l'objet, à savoir la façon dont il est perçu hors de son espace de noms. Vous pouvez choisir parmi les valeurs suivantes : • Private – uniquement par l'objet • Protected – uniquement par l'objet et par ses objets hérités • Package – par tous les objets contenus dans le même package • Public – par tous les objets (option par défaut)
Type de données	Spécifie un classificateur comme type de données.

Propriété	Description
Multiplicité	Spécifie le nombre d'instances de la partie. Si la multiplicité est une plage de valeurs, cela signifie que le nombre de parties peut varier au moment de l'exécution. Vous pouvez choisir parmi les valeurs suivantes : • * – Une à un nombre illimité • 0..* – Aucune ou un nombre illimité • 0..1 – Aucune ou une • 1..* – Une à un nombre illimité • 1..1 – Une à une
Composition	Spécifie la nature de l'association avec l'objet parent. Si cette option est sélectionnée, il s'agit d'une composition, dans le cas contraire, il s'agit d'une agrégation.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Les onglets suivants sont également disponibles :

- Ports - répertorie les ports associés à la partie. Vous pouvez créer des ports directement dans cet onglet. Pour plus d'informations, voir *Ports (MOO)* à la page 65.

Ports (MOO)

Un port est créé à l'extérieur d'un classificateur et spécifie un point d'interaction distinct entre le classificateur et son environnement ou bien entre le (comportement du) classificateur et ses parties internes.

Les ports peuvent être connectés à :

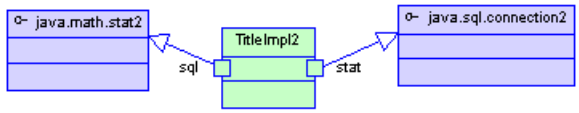
- Une partie via un connecteur de délégation, par le biais duquel les demandes peuvent être effectuées pour appeler des fonctionnalités de comportement d'un classificateur.
- Une interface via un lien de prérequis, par le biais duquel le port peut spécifier les services qu'un classificateur fournit (offre) à son environnement ainsi que les services qu'un classificateur attend (requiert) de son environnement.

Un port peut être créé dans les types de diagramme suivants :

- Diagramme de classes (sur une classe)
- Diagramme de structure composite (sur une classe, une partie, ou une interface)
- Diagramme de composants (sur un composant ou sur une partie)

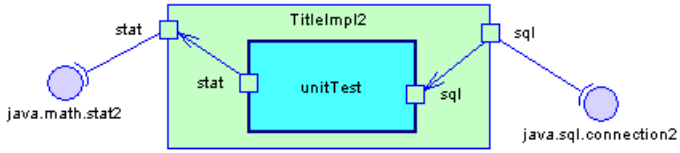
Ports dans un diagramme de classes

Dans l'exemple ci-dessous, la classe TitleImpl2 contient les ports sql et stat, qui sont connectés par des liens de prérequis aux interfaces java.math.stat2 et java.sql.connection2 :



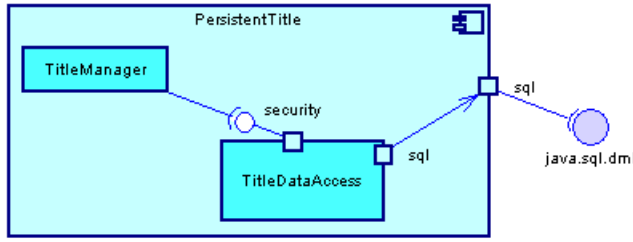
Ports dans un diagramme de structure composite

Dans l'exemple ci-dessous, la structure interne de la classe TitleImpl2 est affichée plus en détails, et montre comment les ports peuvent être utilisés pour spécifier des points d'interaction entre une partie et le classificateur qui la contient :



Ports dans un diagramme de composants

L'exemple ci-dessous illustre l'utilisation des ports pour connecter des parties :



Création d'un port

Vous pouvez créer un port à partir de la Boîte à outils ou de l'onglet **Ports** de la feuille de propriétés d'une classe, d'une partie ou d'un composant.

- Utilisez l'outil **Port** dans la Boîte à outils.
- Affichez l'onglet **Ports** de la feuille de propriétés d'une classe, d'une partie ou d'un composant, puis cliquez sur l'outil **Ajouter une ligne**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'un port

Pour visualiser ou modifier les propriétés d'un port, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les

onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

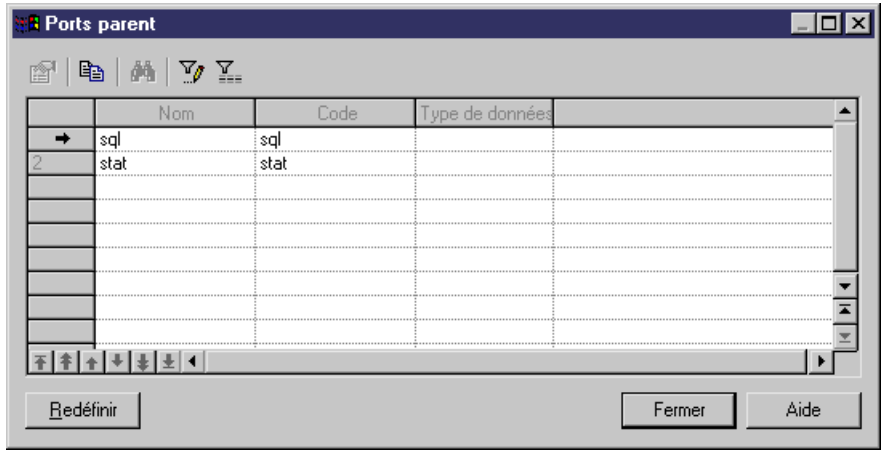
Propriété	Description
Parent	Spécifie le classificateur parent.
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Visibilité	Spécifie la visibilité de l'objet, à savoir la façon dont il est perçu hors de son espace de noms. Vous pouvez choisir parmi les valeurs suivantes : • Private – uniquement par l'objet • Protected – uniquement par l'objet et par ses objets hérités • Package – par tous les objets contenus dans le même package • Public – par tous les objets (option par défaut)
Type de données	Spécifie un classificateur comme type de données.
Multiplicité	Spécifie le nombre d'instances du port. Si la multiplicité est une plage de valeurs, cela signifie que le nombre de ports peut varier au moment de l'exécution. Vous pouvez choisir parmi les valeurs suivantes : • * – Un à un nombre illimité. • 0..* – Aucun ou un nombre illimité. • 0..1 – Aucun ou un. • 1..* – Un à un nombre illimité. • 1..1 – Un à un.
Redéfinit	Un port peut être redéfini lorsque le classificateur qui le contient est spécialisé. Le port qui redéfinit peut avoir des interfaces supplémentaires à celles qui étaient associées au port redéfini ou peut remplacer une interface par l'un de ses sous-types.

Propriété	Description
Service	Spécifie que ce port est utilisé pour fournir la fonctionnalité publiée d'un classificateur (valeur par défaut). Si cette case est décochée, le port est utilisé pour mettre en oeuvre le classificateur mais ne fait pas partie de la fonctionnalité perceptible depuis l'extérieur du classificateur. Il peut toutefois être modifié ou supprimé avec la mise en oeuvre interne du classificateur et les autres propriétés qui sont considérées comme faisant partie de sa mise en oeuvre.
Comportement	Spécifie que ce port est "port de comportement", et les requêtes arrivant à ce port sont envoyées au comportement de classificateur du classificateur. Tout appel d'une fonctionnalité de comportement ciblant un port de comportement peut être géré par les instances du classificateur propriétaire lui-même, plutôt que par des instances que ce classificateur peut contenir.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Redéfinition des ports parent

Un classificateur connecté à un parent par le biais d'une généralisation peut redéfinir les ports du parent.

1. Affichez la feuille de propriétés d'une classe, d'une interface ou d'un composant, puis cliquez sur l'onglet Ports.
2. Cliquez sur le bouton Redéfinir en bas de l'onglet pour afficher la boîte de dialogue Ports parent, qui affiche une liste de parents qui appartiennent au classificateur parent.



1. Sélectionnez un port, puis cliquez sur Redéfinir pour le faire redéfinir par le classificateur enfant.

2. Cliquez sur Fermer pour revenir à la feuille de propriétés de l'enfant. Le port redéfini s'affiche maintenant dans la liste de l'onglet Ports.

Attributs (MOO)

Un attribut est une propriété nommée d'une classe (ou d'une interface) et qui décrit ses caractéristiques.

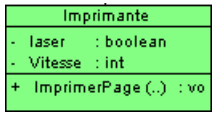
Un attribut peut être créé pour une classe ou une interface dans les types de diagramme suivants :

- Diagramme de classes
- Diagramme de structure composite
- Diagramme de composants

Une classe ou interface peut avoir un ou plusieurs attributs. Tous les objets d'une classe peuvent avoir les mêmes attributs, mais pas avec les mêmes valeurs.

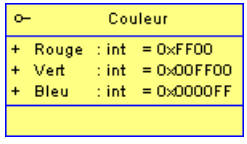
Les noms d'attribut doivent être uniques au sein d'une classe. Vous pouvez affecter le même nom à plusieurs attributs, à condition qu'ils soient placés dans des classes différentes.

Dans l'exemple suivant, la classe Imprimante contient deux attributs nommés Vitesse et Laser :



Attribut d'interface

Un attribut d'interface est différent d'un attribut de classe car une interface ne peut avoir que des attributs constants (statique et figé). Par exemple, considérons une interface nommée Couleur avec trois attributs ROUGE, VERT et BLEU. Ces attributs sont à la fois statiques, finaux et figés.



Si vous générez cette interface en Java, vous obtenez le code suivant :

```
public interface Color
{
 public static final int RED = 0xFF0000;
 public static final int GREEN = 0x00FF00;
 public static final int BLUE = 0x0000FF;
}
```

Ces attributs sont constants car statiques (indépendants de leurs instances), finaux (ne peuvent pas être surchargés), et figés (leur valeur n'est pas modifiable).

Vous pouvez utiliser les attributs des autres interfaces ou classes et les ajouter à l'interface courante.

Création d'un attribut

Vous pouvez créer un attribut à partir de la feuille de propriétés d'une classe, d'un identificateur ou d'une interface.

Affichez la feuille de propriétés d'un classificateur, cliquez sur l'onglet **Attributs**, puis cliquez sur l'un des outils suivants :

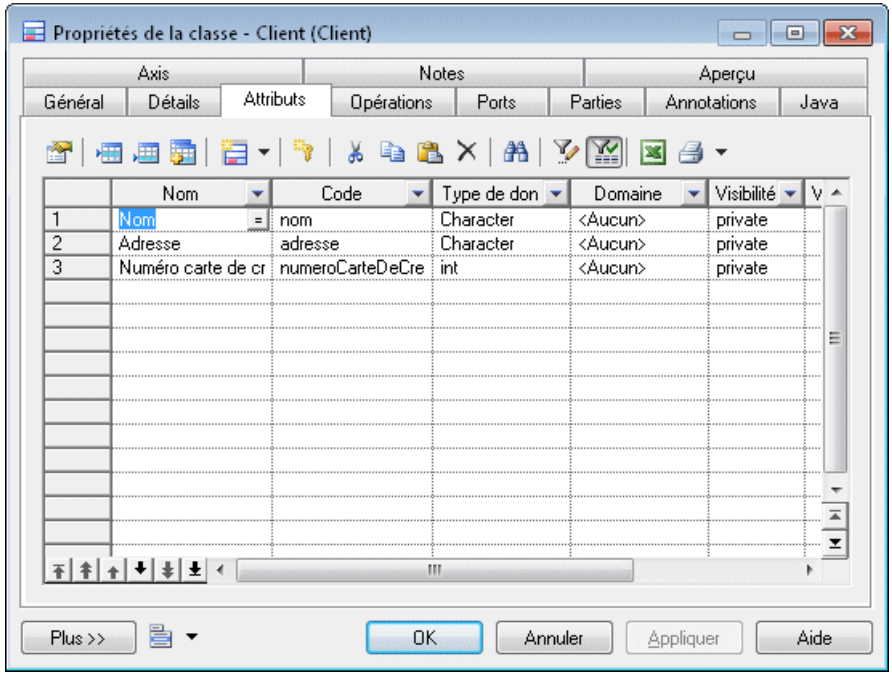
Outil	Description
	**Ajouter une ligne** / **Insérer une ligne** - Saisissez un nom et les autre propriétés appropriées. Vous pouvez également pointer sur une classe ou une interface dans l'Explorateur d'objets, cliquer le bouton droit de la souris, puis sélectionner **Nouveau** > **Attribut**.
	**Ajouter des attributs** - Sélectionnez les attributs existants pour les ajouter au classificateur (voir *Copie d'un attribut dans une classe, interface ou dans un identifiant* à la page 70).
	[PowerAMC] **Redéfinir des attributs hérités** - Sélectionnez les attributs hérités d'un classificateur parent à redéfinir (voir *Redéfinition d'un attribut dans PowerBuilder* à la page 72).
	**Ajouter...** - Cliquez sur la flèche à droite de l'outil, puis sélectionnez le type d'opération standard approprié (comme des constructeurs/destructeurs ou des initialisateurs dans la liste (voir *Ajout d'opération Getter et Setter dans un classificateur* à la page 72).

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

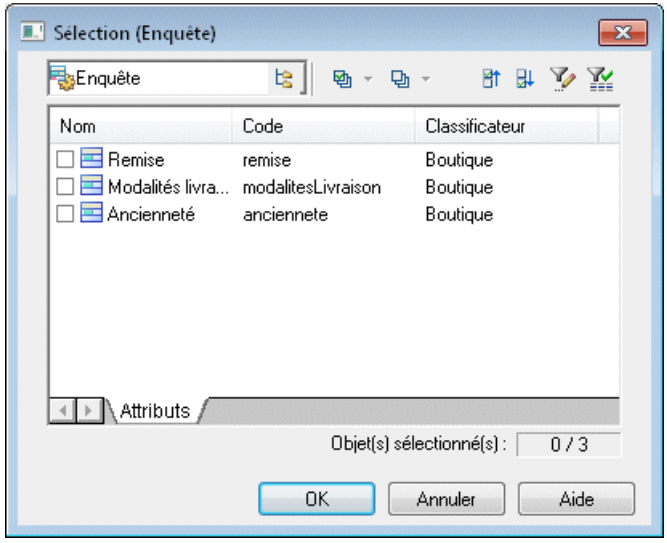
Copie d'un attribut dans une classe, interface ou dans un identifiant

Vous pouvez dupliquer un attribut d'un classificateur dans un autre. Si le classificateur de destination contient déjà un attribut portant le même nom ou code que l'attribut copié, ce dernier est renommé. Par exemple, un attribut nommé PERIPHLD est renommé PERIPHLD2 lorsque vous le copiez dans une classe contenant déjà un attribut nommé PERIPHLD.

1. Affichez la feuille de propriétés d'une classe, d'une interface ou d'un identifiant, puis cliquez sur l'onglet Attributs.



2. Cliquez sur l'outil Ajouter des attributs pour afficher la fenêtre Sélection. Cette fenêtre contient la liste de tous les attributs contenus dans le modèle, à l'exception de ceux qui appartiennent déjà à l'objet.



3. Sélectionnez les attributs à ajouter à l'objet.

   ou

Cliquez sur l'outil Sélectionnez tout pour ajouter les attributs contenus dans la liste à l'objet.

4. Cliquez sur OK pour ajouter les attributs sélectionnés à l'objet courant.

Redéfinition d'un attribut dans PowerBuilder

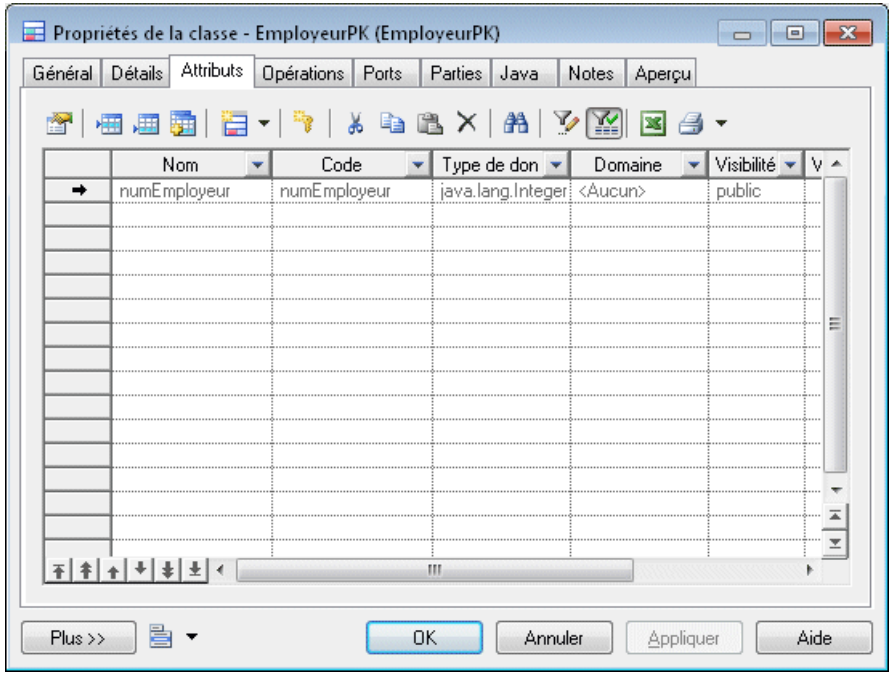
Lorsque vous modélisez pour PowerBuilder, vous pouvez redéfinir les attributs hérités d'une classe parent en les redéfinissant dans la classe enfant. Vous pouvez uniquement modifier la valeur initiale de l'attribut, pas les autres propriétés.

1. Double-cliquez sur une classe qui est liée à une classe parent dans le diagramme pour afficher sa feuille de propriétés, puis cliquez sur l'onglet **Attributs**.
2. Cliquez sur l'outil **Redéfinir les attributs hérités** pour afficher une fenêtre de sélection qui répertorie les attributs qui appartiennent à toutes les classes parent de la classe.
3. Sélectionnez un attribut, puis cliquez sur **OK**. Une copie de l'attribut est ajoutée dans la liste des attributs de la classe enfant. Elle est grisée pour indiquer que ses propriétés ne peuvent pas être modifiées, et son stéréotype est défini à <<Override>>.
4. Cliquez sur **OK**.

Ajout d'opération Getter et Setter dans un classificateur

PowerAMC vous permet de créer rapidement des opérations Getter et Setter pour vos attributs sur l'onglet **Attributs** de votre classificateur.

1. Affichez la feuille de propriétés de votre classificateur, puis cliquez sur l'onglet **Attributs**.



2. Sélectionnez un ou plusieurs attributs, cliquez sur le bouton **Ajouter...** en bas de l'onglet, puis sélectionnez l'action à effectuer. Selon le langage cible, certaines des actions suivantes peuvent être disponibles :
   - Opérations Get/Set - Crée des opérations get et set sur l'onglet **Opérations** pour les attributs sélectionnés.
   - Propriété - [C#/VB.NET uniquement] Crée une propriété sur l'onglet **Attributs** ainsi que des opérations get et set sur l'onglet **Opérations** afin d'accéder à l'attribut original via la propriété.
   - Indexeur - [C# uniquement] Crée un indexeur sur l'onglet **Attributs** ainsi que des opérations get et set sur l'onglet **Opérations** afin d'accéder à l'attribut original via l'indexeur.
   - Opérations d'événement - [C#/VB.NET uniquement, pour les attributs ayant le stéréotype Event] Crée et supprime des opérations sur l'onglet **Opérations** pour l'événement.
3. [facultatif] Cliquez sur l'onglet **Opérations** pour afficher les opérations que vous venez de créer. Certaines valeurs, telles que les noms, ne peuvent pas être modifiées.
4. Cliquez sur **OK** pour fermer la feuille de propriétés et revenir au modèle.

Propriétés d'un attribut

Pour visualiser ou modifier les propriétés d'un attribut, double-cliquez sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de

propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Parent	Spécifie le classificateur auquel l'attribut appartient.
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Visibilité	Spécifie la visibilité de l'objet, à savoir la façon dont il est perçu hors de son espace de noms. Lorsqu'une classe est visible par un autre objet, elle peut influencer la structure ou le comportement de l'objet, ou être affectée par ce dernier. Vous pouvez choisir parmi les valeurs suivantes : • Private – uniquement par la classe à laquelle il appartient • Protected – uniquement par la classe et de ses objets dérivés • Package – par tous les objets contenus dans le même package. • Public – par tous les objets (option par défaut)
Type de données	Jeu d'instances partageant les mêmes opérations, attributs abstraits, relations, et sémantiques

Multiplicité	Spécifie la plage de valeurs admises pour l'attribut. Vous pouvez choisir l'une des valeurs suivantes :
	• 0..1 – zéro ou une
	• 0..* – zéro à un nombre illimité
	• 1..1 – exactement une
	• 1..* – un à un nombre illimité
	• * – aucune ou un nombre illimité
	Vous pouvez changer le format par défaut de la multiplicité depuis le Registre.
	`HKEY_CURRENT_USER\Software\Sybase\PowerAMC 16\ModelOptions\Cld\`
	`MultiplicityNotation = 1 (0..1) or 2 (0,1)`

Propriété	Description
Taille du tableau	Spécifie la multiplicité dans la syntaxe d'un langage donné, lorsque la multiplicité de l'attribut ne peut pas l'exprimer. Par exemple, vous pouvez définir une taille de tableau de [4,6,8] pour obtenir la syntaxe PowerBuilder int n[4,6,8] ou définir une taille de tableau [,,] pour obtenir la syntaxe c# int[,,] n; Selon le langage objet choisi pour le modèle, les valeurs suivantes sont générées : • Java, C# et C++ – [2][4][6] • PowerBuilder – [2,4,6] • VB .NET – (2,4,6)
Classe d'énumération	[Java 5.0 et versions supérieures] Spécifie une classe anonyme pour un EnumConstant. Utilisez les outils à droit de cette zone pour créer une classe, sélectionner une autre classe ou afficher les propriétés de la classe sélectionnées.
Statique	L'attribut est associé à la classe, par conséquence les attributs statiques sont partagés par toutes les instances de la classe et ont toujours la même valeur parmi les instances
Dérivé	Indique que l'attribut peut être calculé à partir d'un autre attribut. La formule de dérivation peut être définie dans l'onglet de description de l'attribut, et n'influe par sur la génération de code.
Obligatoire	Attribut calculé booléen sélectionné si la multiplicité minimale est supérieure à zéro.
Volatile	Indique que l'attribut n'est pas membre de la classe. Défini uniquement par les opérations getter et setter. Dans C#, remplace l'ancien attribut étendu Volatile. Pour plus d'informations sur l'ajout d'opérations dans une classe, voir *Ajout d'opération Getter et Setter dans un classificateur* à la page 72.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Onglet Détails

L'onglet **Détails** contient les propriétés suivantes :

Propriétés	Description
Valeur initiale	Spécifie la valeur initiale affectée à l'attribut lors de sa création.

Propriétés	Description
Modifiable	Spécifie si la valeur de l'attribut peut être modifiée une fois que l'objet a été initialisé. Vous pouvez choisir parmi les valeurs suivantes : • Modifiable – La valeur peut être modifiée • Lecture seule – Empêche la création d'une opération setter (une telle opération est créée dans la méthode contenue dans la classe) • Figé – Constante • Ajout uniquement – Permet uniquement la création d'une nouvelle valeur
Domaine	Spécifie un domaine (voir *Domaines (MOO)* à la page 124), qui va définir le type de données et les autres caractéristiques de données pour l'attribut et peut également indiquer des paramètres de contrôle ainsi que des règles de gestion. Sélectionnez un domaine dans la liste, ou cliquez sur le bouton Points de suspension à droite de la zone pour créer un nouveau domaine dans la liste.
Identifiant primaire	[attributs de classe] Spécifie que l'attribut fait partie d'un identifiant primaire. Les identifiants primaires sont convertis en clés primaires lors de la génération d'un MPD à partir d'un MOO. Valable uniquement pour les classes.
Migré depuis	Spécifie l'attribut redéfini (PowerAMC uniquement) ou l'association migrée (voir *Migration des rôles d'association dans un diagramme de classes* à la page 99). Cliquez sur l'outil **Propriétés** à droite de la zone pour afficher la feuille de propriétés de l'objet référencé.
Persistant	[attributs de classe] Spécifie que l'attribut sera persistant et stocké dans une base de données (voir *Gestion de la persistance des objets lors de la génération de modèles de données* à la page 295).
Code	Spécifie le code de la table ou de l'entité qui sera générée dans un MCD ou un MPD persistant
Type de données	Spécifie un type de données persistant utilisé dans la génération d'un modèle persistant, qu'il s'agisse d'un MCD ou d'un MPD. Le type de données persistant est défini à partir des types de données conceptuels PowerAMC
Longueur	Spécifie le nombre maximal de caractères du type de données persistant.
Précision	Spécifie le nombre de décimales pour les valeurs de type de données persistant qui comportent des chiffres après la virgule

Définition de contraintes de profilage de données

PowerAMC prend en charge le profilage des données dans le modèle physique de données (MPD) et les onglets **Contrôles standard** et **Contrôles supplémentaires** sont disponibles

dans le feuilles de propriétés d'attribut et de domaine de MOO uniquement pour retenir ces informations lorsque vous utilisez la liaison et la synchronisation entre un MOO et un MPD.

Les contraintes suivantes sont disponibles sur l'onglet **Contrôles standard** des feuilles de propriétés d'attributs et de domaines de MOO :

Propriété	Description
Valeurs	Spécifie la plage des valeurs acceptables. Vous pouvez définir un : • Minimum - Valeur numérique la plus basse admise • Maximum - Valeur numérique la plus élevée admise • Défaut - Valeur affectée en l'absence d'une valeur expressément saisie.
Caractéristiques	Ces propriétés sont à des fins de documentation uniquement, et ne seront pas générées. Vous pouvez choisir : • Format - Nombre de formats standard sont fournis dans la liste. Vous pouvez saisir un nouveau format directement dans la zone ou utiliser les outils à droite de la zone pour créer un format de données à réutiliser ailleurs. • Unité - Unité de mesure standard. • Sans espace - Les espaces ne sont pas admis. • Non modifiable - La valeur ne peut pas être modifiée après son initialisation.
Casse des caractères	Spécifie la casse acceptable pour les données. Vous pouvez choisir l'une des valeurs suivantes : • Mixte [défaut] • Majuscules • Minuscules • Première initiale en majuscule • Toutes les initiales en majuscules
Liste des valeurs	Spécifie les diverses valeurs admises. Cochez la case **Complet** en bas de la liste pour exclure toute autre valeur n'apparaissant pas dans la liste.

Création de formats de données à réutiliser

Vous pouvez créer des formats de données à réutiliser dans des contraintes pour plusieurs objets en cliquant sur le bouton **Nouveau** à droite de la zone **Format** sur l'onglet **Contrôles standard**. Les formats de données sont fournis à titre d'information uniquement, et ne sont pas générés sous forme de contraintes.

Remarque : Pour créer plusieurs formats de données, utilisez la boîte de dialogue Liste des formats de données, disponible en sélectionnant **Modèle > Formats de données**.

Propriétés d'un format de données

Pour visualiser ou modifier les propriétés d'un format de données, double-cliquez sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/ Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Type	Spécifie le type du format. Vous pouvez choisir l'une des valeurs suivantes : • Date/Heure • Chaîne • Expression régulière
Expression	Spécifie la forme des données à stocker dans la colonne. Par exemple, 9999.99 représente un nombre de quatre chiffres avec deux décimales.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Spécification de contraintes avancées

L'onglet **Contrôles supplémentaires** est utilisé dans le modèle physique de données (MPD) afin de spécifier des contraintes de colonne complexes, et est disponible dans les feuilles de propriétés d'attribut et de domaine de MOO uniquement pour retenir ces informations lorsque vous utilisez la liaison et la synchronisation entre un MOO et un MPD.

Identifiants (MOO)

Un identifiant est un attribut de classe, ou une combinaison d'attributs de classe, dont les valeurs identifient de façon unique chaque occurrence d'une classe. Il est utilisé lors de la

génération intermodèle, lorsque vous générez un MOO depuis un MCD ou un MPD. Les identifiants de MCD ou les clés primaires ou alternatives de MPD sont transformés en identifiants dans le MOO résultant.

Un identifiant peut être créé pour une classe dans les types de diagramme suivants :

- Diagramme de classes
- Diagramme de structure composite

Chaque classe peut être dotée d'au moins un identifiant. Parmi les identifiants, l'identifiant primaire est le principal identifiant de la classe. Cet identifiant correspond à la clé primaire dans un MPD.

Lorsque vous créez un identifiant, vous pouvez lui attacher des attributs ou des règles de gestion. Vous avez également la possibilité de définir un ou plusieurs attributs comme identificateurs primaires d'une classe.

Par exemple, le numéro de sécurité sociale pourrait être utilisé comme identifiant primaire d'une classe Employé.

Création d'un identifiant

Vous pouvez créer un identifiant à partir de la feuille de propriétés d'une classe ou d'une interface.

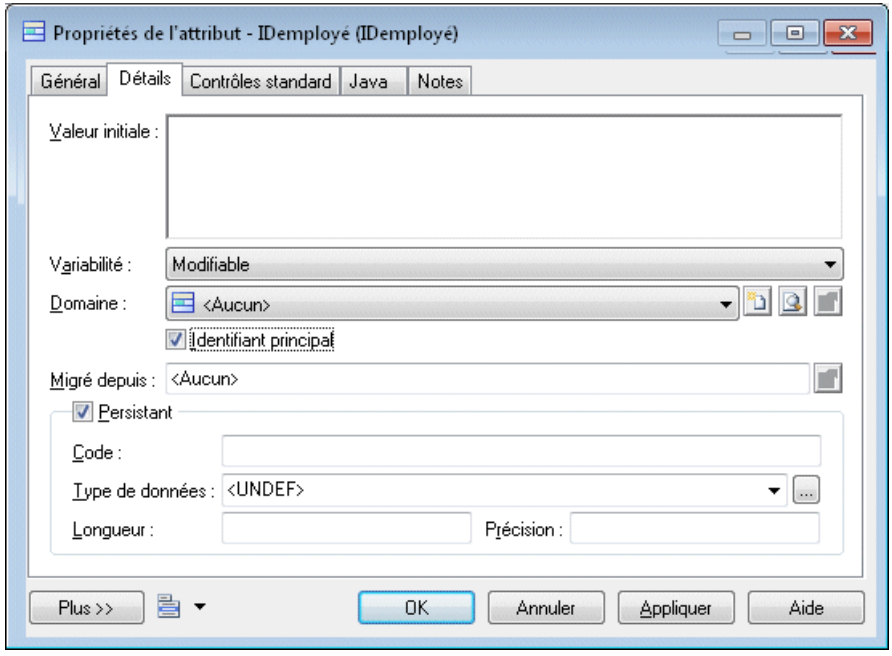
- Affichez l'onglet **Identifiants** dans la feuille de propriétés d'une classe ou d'une interface, puis cliquez sur l'outil **Ajouter une ligne**.
- Affichez l'onglet **Attributs** dans la feuille de propriétés d'une classe ou d'une interface, puis cochez la case **Identifiant primaire** lorsque vous créez un attribut.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Création d'un identifiant primaire lors de la création des attributs de classe

Vous pouvez créer un identifiant primaire lorsque vous créez des attributs pour une classe.

1. Double-cliquez sur le symbole d'une classe dans le diagramme pour afficher sa feuille de propriétés, puis cliquez sur l'onglet Attributs.
2. Double-cliquez sur un attribut dans la liste pour afficher la feuille de propriétés correspondante, puis cliquez sur l'onglet Détails.
3. Cochez la case Identifiant primaire, puis cliquez sur OK pour revenir à l'onglet Attributs de la feuille de propriétés de la classe.



4. Cliquez sur l'onglet Identifiants pour afficher le nouvel identifiant dans la liste.
5. Cliquez sur OK.

Définition d'un identifiant primaire à partir de la liste des identifiants

Vous pouvez définir un identifiant primaire à partir de la boîte de dialogue Liste des identifiants.

1. Sélectionnez **Modèle > Identifiants** pour afficher la liste des identifiants.
2. Double-cliquez sur un identifiant dans la liste pour afficher sa feuille de propriétés.
3. Cochez la case Identifiant primaire.
4. Cliquez sur OK dans les boîtes de dialogue successives.

Propriétés d'un identifiant

Pour visualiser ou modifier les propriétés d'un identifiant, double-cliquez sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

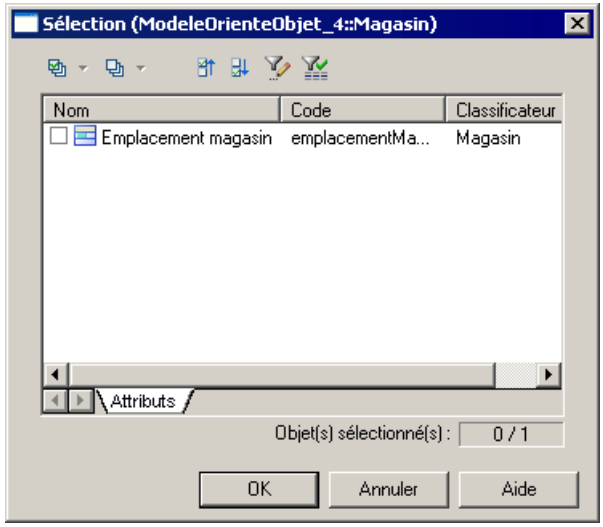
Propriété	Description
Parent	Spécifie la classe à laquelle l'identifiant appartient.

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Identifiant primaire	Spécifie que l'identifiant est un identifiant primaire.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Ajout d'attributs à un identifiant

Un identifiant peut comporter plusieurs attributs. Vous pouvez ajouter des attributs à un identifiant afin de mieux caractériser ce dernier.

1. Sélectionnez un identifiant dans la boîte de dialogue Liste des identifiants ou dans l'onglet Identifiants de la feuille de propriétés d'une classe, puis cliquez sur l'outil Propriétés pour afficher sa feuille de propriétés.
2. Cliquez sur l'onglet Attributs, puis cliquez sur l'outil Ajouter des attributs afin d'afficher la liste des attributs pour la classe.



3. Cochez les cases correspondant aux attributs que vous souhaitez ajouter à l'identifiant.
4. Cliquez sur OK dans les boîtes de dialogue successives.

Opérations (MOO)

Une opération est une spécification nommée d'un service qui peut être requis par n'importe quel objet d'une classe afin de modifier son comportement. Il s'agit de la spécification d'une requête qu'un objet peut être appelé à exécuter.

Une opération peut être créée pour une classe ou une interface dans les types de diagramme suivants :

- Diagramme de classes
- Diagramme de structure composite
- Diagramme de composants

Une classe peut comporter un nombre illimité d'opérations, ou bien n'en comporter aucune.

Dans l'exemple suivant, la classe Voiture comporte 3 opérations : démarrer, freiner et accélérer.

Les opérations doivent être dotées d'un nom et d'une liste de paramètres. Plusieurs opérations peuvent porter le même nom au sein d'une classe, à condition toutefois que leurs paramètres soient différents.

Pour plus d'informations sur *opérations d'EJB*, voir *Définition d'opérations pour un EJB* à la page 377.

Création d'une opération

Vous pouvez créer une opération à partir de la feuille de propriétés ou du noeud d'Explorateur d'objets d'une classe ou d'une interface.

Affichez la feuille de propriétés d'un classificateur, cliquez sur l'onglet **Opérations**, puis cliquez sur l'un des outils suivants :

Outil	Description
	**Ajouter une ligne** / **Insérer une ligne** - Saisissez un nom et les autres propriétés appropriées. Vous pouvez également pointer sur une classe ou interface dans l'Explorateur d'objets, cliquer le bouton droit de la souris, puis sélectionner **Nouveau** > **Opération**.
	**Ajouter des opérations** - Sélectionnez des opérations existantes à ajouter au classificateur (voir *Copie d'une opération depuis un autre classificateur* à la page 83).

Outil	Description
	**Redéfinir des opérations héritées** - Sélectionnez des opérations héritées d'un classificateur parent à redéfinir (voir *Héritage et redéfinition d'opérations à partir de classificateurs parent* à la page 83).
	**Ajouter...** - Cliquez sur la flèche à droite de l'outil, puis sélectionnez le type d'opération standard approprié (comme constructeurs/destructeurs ou initialisateurs) dans la liste (voir *Création d'une opération standard* à la page 84).
	**Opérations non réalisées** - Sélectionnez des opérations à réaliser à partir d'une interface à laquelle la classe courante est liée par un lien de réalisation (voir *Réalisation d'opération à partir d'une interface* à la page 85).

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Copie d'une opération depuis un autre classificateur

Vous pouvez copier une opération d'un classificateur à un autre. Si le classificateur contient déjà une opération portant le même nom ou code que l'opération copiée, cette dernière est renommée.

1. Affichez la feuille de propriétés d'un classificateur, puis cliquez sur l'onglet **Opérations**.
2. Cliquez sur l'outil **Ajouter des opérations** pour afficher une boîte de dialogue de sélection qui répertorie toutes les opérations disponibles dans le modèle.
3. Sélectionnez une ou plusieurs opérations dans la liste, puis cliquez sur **OK** pour créer l'opération et l'ajouter dans le classificateur.

Héritage et redéfinition d'opérations à partir de classificateurs parent

PowerAMC permet de visualiser et de redéfinir des opérations héritées de classes parent à partir de l'onglet **Opérations** de la feuille de propriétés de la classe enfant.

Pour hériter et être en mesure de redéfinir des opérations héritées, votre classe doit être liée à un ou plusieurs classificateurs parent pour lesquels des opérations ont été définies.

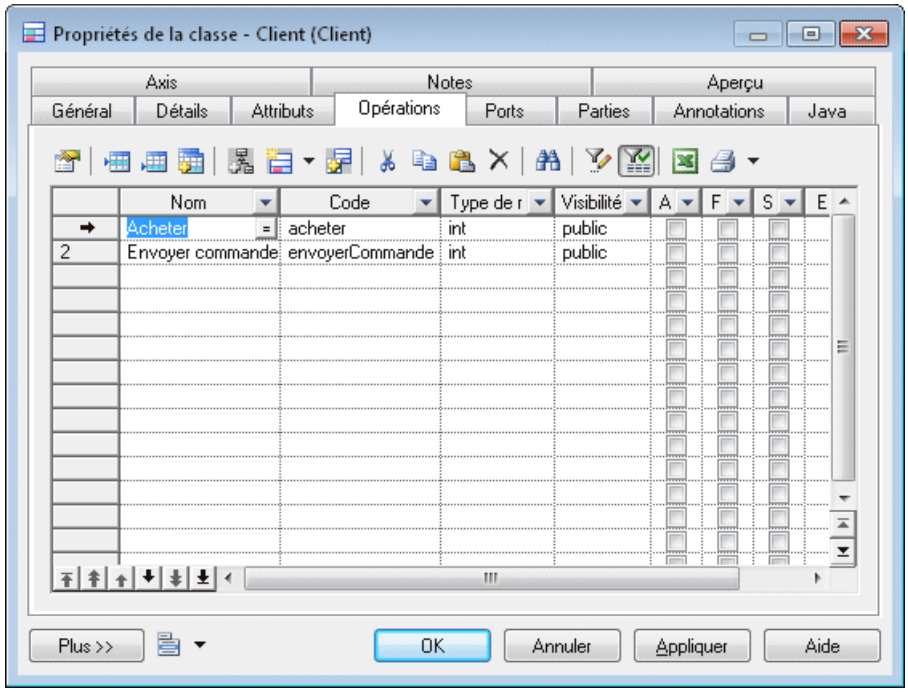
1. Affichez la feuille de propriétés d'une classe qui est liée par un lien de généralisation à un ou plusieurs parents, puis cliquez sur l'onglet **Opérations**.
2. Cliquez sur le bouton **Redéfinir les opérations héritées** afin d'afficher une boîte de dialogue de sélection, qui répertorie les opérations dont les classificateurs hérite de ses parents.
3. Sélectionnez une ou plusieurs opérations, puis cliquez sur **OK** afin de les copier dans la liste des opérations de la classe enfant avec leur stéréotype défini à `<<Override>>`.

   Chaque opération redéfinie a la même signature (nom et paramètres) que l'opération d'origine, et vous ne pouvez modifier que le code spécifié sur l'onglet **Mise en oeuvre** de sa feuille de propriétés.

Création d'une opération standard

PowerAMC permet de créer des opérations standard pour vos classificateurs en utilisant les outils situés sur l'onglet **Opérations** de la feuille de propriétés du classificateur.

1. Affichez la feuille de propriétés du classificateur, puis cliquez sur l'onglet **Opérations**.



2. Cliquez sur le bouton **Ajouter...**, puis sélectionnez le type d'opération standard que vous souhaitez ajouter :
   - *Constructeur/Destructeur par défaut* - pour initialiser/supprimer des classificateurs. Vous pouvez ajouter les paramètres ultérieurement.
   - *Constructeur par copie* - pour copier les attributs d'une instance de classe afin d'initialiser une autre instance.
   - *Initialisateur/initialisateur statique*- [Java uniquement] pour initialiser une classe avant tout constructeur.
   - *Opération de duplication* - pour créer et initialiser une instance d'une classe dans la classe.
   - *Activate/Deactivate* - [PowerBuilder uniquement]

   L'opération est ajoutée dans la liste. Certaines de ses propriétés, voire toutes, sont affichées en grisé pour indiquer qu'elles ne sont pas modifiables.

3. [facultatif] Sélectionnez l'opération, puis cliquez sur l'outil **Propriétés** afin d'ajouter des paramètres à cette opération ou pour compléter sa définition.
4. Ajoutez d'autres opérations si nécessaire, ou bien cliquez sur **OK** pour fermer la feuille de propriétés et revenir à votre modèle.

Réalisation d'opération à partir d'une interface

Lorsque vous créez un lien de réalisation entre une classe et une interface, la classe doit mettre en oeuvre toutes les opérations de l'interface.

Remarque : Pour créer automatiquement les opérations nécessaires dans votre classe, sélectionnez **Outil > Options du modèle** pour afficher la boîte de dialogue Options du modèle, puis sélectionnez **Mettre en oeuvre automatiquement les interfaces réalisées**. Si cette option n'est pas sélectionnée, vous pouvez mettre en oeuvre les opérations manuellement.

1. Affichez la feuille de propriétés d'une classé liée par un ou plusieurs liens de réalisation, puis cliquez sur l'onglet **Opérations**.
2. Cliquez sur l'outil **Opérations non réalisées** pour afficher une boîte de dialogue de sélection, qui répertorie toutes les opérations qui attendent d'être réalisées dans la classe.
3. Sélectionnez une ou plusieurs opérations, puis cliquez sur **OK** afin de les copier dans la liste des opérations de la classe avec leur stéréotype défini à `<<Implement>>`.

   Chaque opération réalisée a la même signature (nom et paramètres) que l'opération originale, et vous ne pouvez modifier son code que sur l'onglet **Mise en oeuvre**.

Propriétés d'une opération

Pour visualiser ou modifier les propriétés d'une opération, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Parent	Spécifie le classificateur parent auquel l'opération appartient.
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.

Propriété	Description
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension. Les stéréotypes communs suivants sont disponibles par défaut : • <<constructor>> - Opération appelée lors de l'instanciation d'un objet et qui crée une instance d'une classe • <<create>> - Opération utilisée par une classe lorsqu'elle instancie un objet. • <<destroy>> - Opération utilisée par une classe et qui détruit une instance d'une classe • <<storedProcedure>> - Opération qui deviendra une procédure stockée dans le MPD généré • <<storedFunction>> - Opération qui deviendra une fonction stockée dans le MPD généré • <<EJBCreateMethod>> - CreateMethod spécifique aux EJB • <<EJBFinderMethod>> - FinderMethod spécifique aux EJB. • <<EJBSelectMethod>> - SelectMethod spécifique aux EJB. Pour plus d'informations sur les méthodes spécifiques aux EJB, voir *Définition d'opérations pour un EJB* à la page 377.
Type de résultat	Une liste de valeurs renvoyées par un appel de l'opération. Si l'opération ne renvoie aucune valeur, la valeur Type de résultat est NULL
Visibilité	[opérateurs de classe] Visibilité de l'opération, dont les valeurs indiquent comment elle est perçue hors de son espace de noms. N'existe que dans les classes : • Private - Uniquement par la classe à laquelle elle appartient • Protected - Uniquement par la classe et ses objets dérivés. • Package - Par tous les objets du même package. • Public - Par tous les objets
Evénement de langage	Lorsque les classes représentent des éléments d'interfaces, cette zone permet de montrer une opération comme déclenchée par une occurrence significative d'un événement.
Statique	L'opération est associée à la classe, les opérations statiques sont donc partagées par toutes les instances de la classe et ont la même valeur pour toutes ces instances.

Propriété	Description
Tableau	Indicateur définissant le type de résultat de l'opération. A la valeur "true" si la valeur renvoyée est un tableau.
Abstrait	Indique que l'opération ne peut pas être instanciée et qu'elle n'est donc dotée d'aucune instance directe.
Final	Indique que l'opération ne peut pas être redéfinie
Lecture seule	Opération dont l'exécution ne change pas l'instance de la classe.
Méthode de service Web	Si affichée et sélectionnée, implique que l'opération est utilisée comme méthode de service Web.
Objet influent	Spécifie l'opération sur laquelle est basée l'opération courante. En général, il s'agit soit d'une opération parent qui est redéfinie via un lien de généralisation, soit d'une opération d'interface qui est mise en oeuvre via un lien de généralisation.
Générique	Spécifie que l'opération est une méthode générique (voir *Types et méthodes génériques* à la page 45).
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Onglet Paramètres

L'onglet Paramètres répertorie les paramètres de votre opération. Chaque paramètre est une variable qui peut être modifiée, transmise ou renvoyée. Les propriétés d'un paramètre sont les suivantes :

Propriété	Description
Parent	Spécifie l'opération à laquelle le paramètre appartient.
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.

Propriété	Description
Type de données	Série d'instances qui partagent les mêmes opérations, attributs abstraits, relations et sémantique
Tableau	Lorsque sélectionné, permet de présenter les attributs sous forme de tableau.
Taille de tableau	Spécifie une taille de tableau précise lorsque la multiplicité est supérieure à 1.
Argument variable	Spécifie que la méthode peut prendre un nombre variable de paramètres pour un argument donné. Vous ne pouvez sélectionner cette propriété que si le paramètre est le dernier de la liste.
Type de paramètre	Direction du flux d'information du paramètre. Indique le type de valeur renvoyé lorsque le paramètre est appelé par l'opération lors de la procédure d'exécution. Vous pouvez définir les valeurs suivantes pour le type de paramètre : • Entrée - Paramètre d'entrée passé par une valeur. La valeur finale ne peut pas être modifiée et l'information n'est pas disponible pour l'appelant • Entrée/Sortie - Paramètre d'entrée qui peut être modifié. La valeur finale peut être modifiée pour communiquer l'information à l'appelant • Sortie - Paramètre de sortie. La valeur finale peut être modifiée pour communiquer l'information à l'appelant
Valeur par défaut	Valeur par défaut lorsqu'un paramètre est omis. Par exemple : Vous utilisez une opération `oper(string param1, integer param2)`, et spécifiez deux arguments oper(val1, val2) lors de l'invocation. Certains langages, par exemple C++, permettent de définir une valeur par défaut qui est ensuite mémorisée lorsque le paramètre est omis lors de l'invocation. Si la déclaration de la méthode est oper(string param1, integer param2 = default), l'invocation oper(val1) équivaut à oper(val1, default).
Type de données WSDL	Uniquement disponible avec les services Web. Définit le type XML Schema/SOAP utilisé lors de l'appel d'une méthode Web (en utilisant HTTP ou Soap)
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Onglet Mise en oeuvre

L'onglet Mise en oeuvre permet de spécifier le code qui sera utilisé pour mettre en oeuvre l'opération, et contient les sous-onglets suivants, en bas de la boîte de dialogue :

Elément	Description
Corps	Code de la mise en oeuvre

Elément	Description
Exceptions	Signal émis en réponse à une erreur de comportement lors de l'exécution du système. Utilisez l'outil **Ajouter une exception** pour sélectionner un classificateur d'exception à ajouter à l'emplacement du curseur.
Pré-conditions	Contrainte qui doit être vérifiée lorsqu'une opération est appelée.
Post-conditions	Contrainte qui doit être vérifiée à la fin d'une opération.
Spécification	Similaire à un pseudo code, il s'agit de la description d'une séquence normale d'actions.

Les onglets suivants sont également disponibles :

- Paramètres - répertorie les paramètres de l'opération. Chaque paramètre est une variable qui peut être changée, passée ou renvoyée (voir *Paramètres (MOO)* à la page 89).
- Paramètres génériques - permet de spécifier les paramètres de type d'une méthode générique (voir *Types et méthodes génériques* à la page 45).
- Diagrammes associés - répertorie et permet d'ajouter des diagrammes de modèles liés à l'opération (voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Diagrammes, matrices et symboles > Diagrammes > Spécification de diagrammes comme diagrammes associés*).

Paramètres (MOO)

Un paramètre est une variable qui peut être modifiée, passée ou renvoyée. Pour visualiser ou modifier les propriétés d'un paramètre, sélectionnez-le sur l'onglet Paramètres de la feuille de propriétés d'une opération ou d'un événement, puis cliquez sur l'outil **Propriétés**.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Parent	Spécifie l'opération ou l'élément auquel le paramètre appartient.
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.

Propriété	Description
Type de données	Série d'instances qui partagent les mêmes opérations, attributs abstraits, relations et sémantique.
Tableau	Spécifie que le type de données est un format de tableau.
Taille de tableau	Spécifie une taille de tableau précise lorsque la multiplicité est supérieure à 1.
Argument variable	Spécifie que la méthode peut prendre un nombre variable de paramètres pour un argument donné. Vous ne pouvez sélectionner cette propriété que si le paramètre est le dernier de la liste.
Type de paramètre	Direction du flux d'information du paramètre. Vous pouvez définir les valeurs suivantes : • Entrée - Paramètre d'entrée passé par une valeur. La valeur finale ne peut pas être modifiée et l'information n'est pas disponible pour l'appelant • Entrée/Sortie - Paramètre d'entrée qui peut être modifié. La valeur finale peut être modifiée pour communiquer l'information à l'appelant • Sortie - Paramètre de sortie. La valeur finale peut être modifiée pour communiquer l'information à l'appelant
Valeur par défaut	Valeur par défaut lorsqu'un paramètre est omis. Par exemple : Vous utilisez une opération `oper(string param1, integer param2)`, et spécifiez deux arguments oper(val1, val2) lors de l'invocation. Certains langages, par exemple C++, permettent de définir une valeur par défaut qui est ensuite mémorisée lorsque le paramètre est omis lors de l'invocation. Si la déclaration de la méthode est oper(string param1, integer param2 = default), l'invocation oper(val1) équivaut à oper(val1, default).
Type de données WSDL	Uniquement disponible avec les services Web. Définit le type XML Schema/SOAP utilisé lors de l'appel d'une méthode Web (en utilisant HTTP ou Soap)
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Associations (MOO)

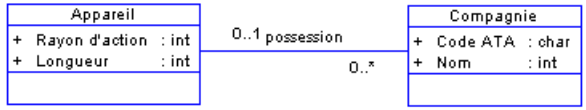
Une *association* représente une relation structurelle entre des classes ou entre une classe et une interface.

Une association peut être créée dans les types de diagramme suivants :

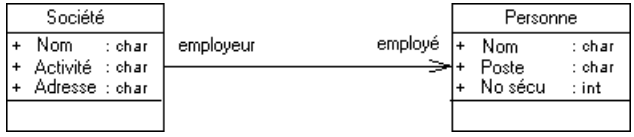
- Diagramme de classes

- Diagramme de structures composites

Une association est représentée sous la forme d'un trait plein entre deux symboles.



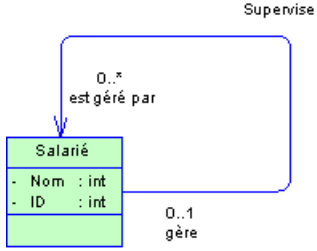
Vous pouvez non seulement nommer l'association elle-même, mais aussi spécifier un nom de rôle pour chaque extrémité de celle-ci afin de décrire la fonction d'une classe du point de vue de la classe opposée. Par exemple, une personne considère la société qui l'emploie comme un employeur et la compagnie considère cette personne comme un employé.



Association réflexive

Une association réflexive est une association entre une classe et elle-même.

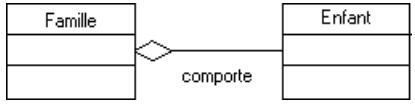
Dans l'exemple suivant, l'association Supervise exprime le fait qu'un employé peut être à la fois gestionnaire et géré.



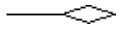
Dans l'onglet **Dépendances** de la feuille de propriétés d'une classe, vous pouvez voir deux occurrences identiques de l'association, ce afin d'indiquer que l'association est réflexive et sert à la fois d'origine et de destination pour le lien.

Agrégation

Une *agrégation* est un type d'association dans lequel une classe représente un ensemble (un tout) composé d'éléments (les parties). Cette association spéciale est un lien de type "comporte un". Ceci signifie qu'un objet du tout comporte des objets de la partie. Dans l'exemple suivant, la famille est le tout et elle peut comporter des enfants.

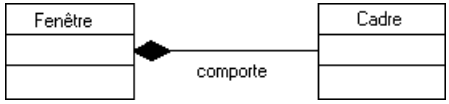


Vous pouvez créer une agrégation directement en utilisant l'outil **Agrégation** dans la Boîte à outils. Le symbole d'agrégation dans un diagramme se présente comme suit :



Composition

Une *composition* est un type particulier d'agrégation dans lequel les parties sont fortement liées au tout. Dans une composition, un objet ne peut être partie que d'un seul composite à la fois, et le composite gère la création et la destruction de ses parties. Dans l'exemple suivant, le cadre est une partie d'une fenêtre. Si vous supprimez la fenêtre, le cadre disparaît également.



Vous pouvez créer une composition directement en utilisant l'outil **Composition** dans la Boîte à outils. Le symbole de composition dans un diagramme se présente comme suit :

Vous pouvez définir l'un des rôles d'une association comme étant une agrégation ou une composition. La propriété Conteneur doit être définie pour spécifier lequel des deux rôles est une agrégation ou une composition.

Création d'une association

Vous pouvez créer un association à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Association**, **Agrégation** ou **Composition** dans la Boîte à outils.
- Sélectionnez **Modèle > Associations** pour afficher la boîte de dialogue Liste des associations, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Association**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'une association

Pour visualiser ou modifier les propriétés d'une association, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Classe A/Classe B	Spécifie les classes à chaque extrémité de l'association. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Type	Spécifie le type de l'association. Vous pouvez choisir parmi les valeurs suivantes : • Association. • Agrégation – relation tout-partie entre une classe et une classe agrégée • Composition – forme d'agrégation avec une forte notion de propriété entre les parties et le tout et des cycles de vie coïncidents.
Conteneur	Si l'association est une agrégation ou une composition, les boutons radio Conteneur permettent de spécifier quelle classe contient l'autre dans l'association.
Classe d'association	Classe reliée à l'association courante et qui complète la définition de l'association.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Onglet Détails

Chaque extrémité d'une association est appelée un *rôle*. Vous pouvez définir sa multiplicité, sa persistance, son ordre et son caractère modifiable. Vous avez également la possibilité de définir sa mise en oeuvre.

Propriété	Description
Nom de rôle	Nom de la fonction de la classe du point de vue de la classe opposée

Propriété	Description
Visibilité	Spécifie la visibilité du rôle de l'association, dont la valeur indique comment elle est perçue hors de son espace de noms. Une association dont le rôle est visible par un autre objet peut influer sur la structure ou le comportement de cet objet. De même, ses propres propriétés peuvent être affectées par cet objet. Vous pouvez choisir parmi les valeurs suivantes : • Private – uniquement par l'objet. • Protected – uniquement par l'objet et par ses objets hérités • Package – par tous les objets contenus dans le même package • Public – par tous les objets (option par défaut)
Multiplicité	La cardinalité de chacun des deux rôles d'une association est appelée multiplicité. La multiplicité indique les nombres maximal et minimal de valeurs qu'un rôle peut avoir. Vous pouvez choisir parmi les valeurs suivantes : • 0..1 – Aucune ou une • 0..* – D'aucune à l'infini • 1..1 – Exactement une • 1..* – D'une à l'infini. • * – Aucune ou infini. Chaque rôle d'une association fait l'objet d'un attribut étendu. Cet attribut étendu permet de choisir les modalités de mise en oeuvre de l'association. Ils sont disponibles dans votre langage objet courant, dans la catégorie `Profile\Association\ExtendedAttributes`, sous les noms `roleAContainer` et `roleBContainer`. Ces attributs étendus sont pertinents uniquement dans le cas d'une multiplicité de type "plusieurs" (représentée par *), ils fournissent une définition des collections d'associations. Pour plus d'informations, voir *Personnalisation et extension de PowerAMC > Fichiers de définition pour les langage objet, de processus et XML*.
Taille de tableau	Spécifie une taille de tableau précise lorsque la multiplicité est supérieure à 1.
Modifiable	Spécifie si un jeu de liens associés à un objet peuvent changer une fois l'objet initialisé. Vous pouvez choisir parmi les valeurs suivantes : • Modifiable – Vous pouvez ajouter, supprimer et modifier les associations sans contrainte • Lecture seule – Vous n'êtes pas autorisé à modifier l'association • Figé – Association constante • Ajout uniquement – De nouvelles associations peuvent être ajoutées à partir d'une classe sur l'extrémité opposée de l'association

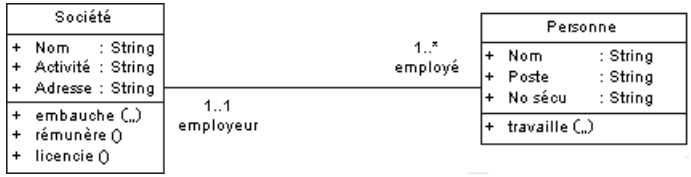
Propriété	Description
Ordre	L'association est incluse dans l'ordre qui trie la liste des associations en fonction de leur ordre de création. Vous pouvez choisir parmi les valeurs suivantes : • Trié – Les objets situés à l'extrémité d'une association sont arrangés en fonction de la façon dont ils sont définis dans le modèle • Ordonné – Les objets situés à l'extrémité d'une association sont arrangés en fonction d'un ordre spécifique • Non ordonné – Les objets situés à l'extrémité d'une association ne sont ni triés, ni ordonnés
Valeur initiale	Spécifie une instruction pour initialiser les attributs migrés, par exemple 'new client ()'.
Navigable	Spécifie que l'information peut être transmise entre les deux objets liés par la relation.
Persistante	Spécifie que l'instance de l'association est conservée après la disparition du processus qui l'a créé
Volatile	Spécifie que les attributs migrés correspondants ne sont pas membres de la classe. Est défini uniquement par les opérations getter et setter.
Type de conteneur	Spécifie une collection de conteneurs pour les attributs migrés de types complexes.
Classe de mise en oeuvre	Spécifie la mise en oeuvre de conteneur (voir *Mise en oeuvre d'une association* à la page 95).
Attribut migré	Spécifie le nom du rôle d'association migré.

Mise en oeuvre d'une association

Les associations décrivent des relations structurelles entre des classes, qui deviennent des liens entre les instances de ces classes. Ces liens représentent une navigation interobjet, c'est-à-dire le fait qu'une instance puisse extraire une autre instance via le lien navigable.

Lorsque l'extrémité d'une association est navigable, cela signifie que vous souhaitez pouvoir extraire l'instance de la classe à laquelle cette extrémité est liée, cette instance s'affiche sous forme d'*attribut migré* dans l'instance courante d'une classe. Le nom de rôle de cette extrémité peut être utilisé pour clarifier la structure utilisée pour représenter le lien.

Par exemple, considérons une association entre la classe Société et la classe Personne. La navigation est possible dans les deux directions pour permettre à Société d'obtenir la liste des employés, et à chaque employé d'obtenir le nom de sa société.



PowerAMC prend en charge différents modes de mise en oeuvre des associations dans chaque langage objet.

Mise en oeuvre par défaut

Par défaut, les attributs migrés utilisent la classe dont ils proviennent comme type.

Lorsque la multiplicité d'association est supérieure à un, le type est le plus souvent un tableau de la classe, affiché avec des signes [ ]. Dans notre exemple, l'attribut Employé de la classe Société est de type Personne et a un tableau de valeurs. Lorsque vous instanciez la classe Société, vous obtenez une liste d'employés à stocker pour chaque société.

```
public class Company
{
 public String Name;
 public String Catalog;
 public String Address;

 public Person[] employee;
```

Selon le type de langage sur lequel vous travaillez, vous pouvez mettre en oeuvre des attributs migrés de façons différentes. PowerAMC permet de choisir une mise en oeuvre dans l'onglet Détails de la feuille de propriétés d'associations.

Type de conteneur et classe de mise en oeuvre

Un*conteneur* est un type de collection d'objets qui stocke des éléments. La structure de conteneur est plus complexe que celle du type de tableau et fournit plus de méthodes pour accéder aux éléments (tester l'existence de l'élément, insérer l'élément dans la collection, etc.) et pour gérer l'allocation de mémoire de façon dynamique.

Vous pouvez sélectionner un type de conteneur à partir de la liste *Type de conteneur*. Ce type sera utilisé par les attributs migrés. Le code d'un attribut migré qui utilise un type de conteneur contient des fonctions getter et setter utilisées pour définir la mise en oeuvre de l'association. Ces fonctions sont visibles dans l'onglet Aperçu du code, mais ne s'affichent pas dans la liste des opérations.

Lorsque vous utilisez la fonctionnalité de migration de rôle pour visualiser des attributs migrés et sélectionner un type de conteneur, le code généré est identique.

Selon le langage et les bibliothèques que vous utilisez, le type de conteneur peut être associé à une *classe de mise en oeuvre*. Dans ce cas, le type de conteneur est utilisé comme une interface pour déclarer les fonctionnalités de la collection, et la classe de mise en oeuvre développe cette collection. Par exemple, si vous sélectionnez les type de conteneur java.util.Set, vous savez

que cette collection ne contient aucun élément en double. Vous pouvez alors sélectionner une classe de mise en oeuvre parmi les suivantes : HashSet, LinkedHashSet ou TreeSet.

Pour plus d'informations sur les types de conteneur et les classes de mise en oeuvre, reportez-vous à la documentation relative au langage approprié.

Plus d'informations sur la classe de mise en oeuvre

Le mécanisme de mise en oeuvre par défaut est défini dans le fichier de ressources situé sous la catégorie Profile\Association. Ce mécanisme utilise des templates pour définir la syntaxe générée pour l'attribut migré, les opérations à générer, et les autres détails relatifs à l'association. Par exemple, le template *roleAMigratedAttribute* permet de récupérer la visibilité et la valeur initiale du rôle d'association. Vous pouvez utiliser l'interface de l'éditeur de ressources pour modifier les détails de mise en oeuvre.

Pour plus d'informations sur la définition de template, voir *Personnalisation et extension de PowerAMC > Personnalisation de la génération à l'aide du langage de génération par template*

Notions de base relatives au code généré

Lorsque vous définissez une classe de mise en oeuvre, les détails de la mise en oeuvre de l'association sont toujours générés dans les classes d'origine et/ou de destination de l'association navigable.

Les détails de la mise en oeuvre d'une association ne sont pas générés dans le code de l'association, en revanche la génération utilise des balises de documentation spécifiques pour stocker les informations relatives à l'association. Ces balises de commentaire rassemblent tous les détails requis pour recréer l'association à l'issue d'un reverse engineering. Les balises de commentaires de documentation sont traitées lors du reverse engineering afin de rendre possible l'ingénierie par va-et-vient.

Les balises de documentation suivantes sont utilisées :

- *pdRoleInfo* permet d'extraire le nom du classificateur, le type de conteneur, la classe de mise en oeuvre, la multiplicité et le type d'association.
- *pdGenerated* est utilisé pour signaler les fonctions générées automatiquement liées à la mise en oeuvre de l'association. Ces fonctions ne doivent pas faire l'objet d'un reverse engineering, faute de quoi les modèles générés et récupérés via reverse engineering seront différents.

Avertissement ! Prenez garde à ne pas modifier ces balises afin de préserver vos possibilités d'ingénierie par va-et-vient.

Dans Java

La syntaxe de tag javadoc est utilisée */**@valeur balise*/*.

Dans l'exemple suivant, la balise @pdRoleInfo est utilisée pour stocker des détails de la mise en oeuvre d'association, et @pdGenerated est utilisée pour indiquer que la méthode getter est automatiquement générée et ne doit pas faire l'objet d'un reverse engineering.

```
   /**@pdRoleInfo name=Person coll=java.util.Collection
impl=java.util.LinkedList mult=1..* */
   public java.util.Collection employee;
   /** @pdGenerated default getter */
 public java.util.Collection getEmployee()
   {
      if (employee == null)
         employee = new java.util.HashSet();
      return employee;
   }
...
```

Dans C#

La balise de documentation *///<valeur balise />* est utilisée.

Dans l'exemple suivant, la balise <pdRoleInfo> est utilisée pour stocker les détails de mise en oeuvre d'association, et <pdGenerated> est utilisée pour indiquer que la méthode getter est automatiquement générée et ne doit pas faire l'objet d'un reverse engineering.

```
   ///<pdRoleInfo name='Person' coll='System.CollectionsArrayList'
impl='java.util.LinkedList' mult='1..*' type='composition'/>
   public java.util.Collection employee;
   ///<pdGenerated> default getter </pdGenerated>
...
```

Dans VB .NET

La balise de documentation *"<valeur balise />* est utilisée.

Dans l'exemple suivant, la balise <pdRoleInfo> est utilisée pour stocker les détails de la mise en oeuvre d'association, et <pdGenerated> est utilisée pour indiquer que la méthode getter est automatiquement générée et ne doit pas faire l'objet d'un reverse engineering.

```
   "<pdRoleInfo name='Person' coll='System.CollectionsArrayList'
impl='java.util.LinkedList' mult='1..*' type='composition'/>
   public java.util.Collection employee;
   "<pdGenerated> default getter </pdGenerated>
...
```

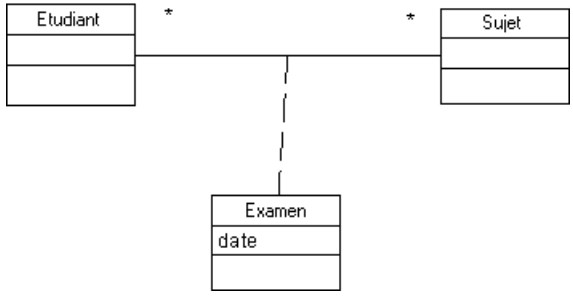
Création d'une classe d'association

Vous pouvez ajouter des propriétés à une association liant des classes ou liant une classe et une interface en créant une *classe d'association*, celle-ci permet d'enrichir les propriétés d'une association en lui ajoutant des attributs et opérations.

Une classe d'association est une association qui a les propriétés d'une classe ou une classe qui a des propriétés d'association. Dans le diagramme, le symbole d'une classe d'association est une connexion entre une association et une classe. Les classes d'association doivent se trouver dans le même package que l'association parent. Vous ne pouvez pas utiliser un raccourci de classe pour créer une classe d'association.

La classe utilisée pour créer une classe d'association ne peut pas être réutilisée pour une autre classe d'association. Vous pouvez en revanche créer d'autres types de liens depuis et vers cette classe.

Dans l'exemple suivant, les classes Etudiant et Sujet sont liées par l'association Examen. En revanche, cette association ne spécifie pas la date de l'examen. Vous pouvez alors créer une classe d'association appelée Examen qui va indiquer les informations supplémentaires relatives à l'association.



1. Pointez sur l'association, cliquez le bouton droit de la souris, puis sélectionnez Ajouter une association de classe dans le menu contextuel.
2. Double-cliquez sur l'association pour afficher sa feuille de propriétés, puis cliquez sur le bouton Créer à droite de la liste Classe d'association.

Une ligne discontinue est automatiquement ajoutée entre la classe et l'association.

Migration des rôles d'association dans un diagramme de classes

Il est possible de migrer des rôles d'association et de créer des attributs avant la génération. Cette fonctionnalité permet entre autres de personnaliser les types de données et de modifier l'ordre des attributs dans les listes d'attributs, ce qui peut se révéler particulièrement utile pour XML.

Quelle que soit la navigabilité, la migration crée un attribut et définit ses propriétés comme suit :

- Nom et code de l'attribut : le rôle de l'association s'il est déjà défini, dans le cas contraire, utilise le nom de l'association.
- Type de données : le code du classificateur lié par l'association.
- Multiplicité : la multiplicité du rôle.
- Visibilité : la visibilité du rôle.

Règles de migration

Les règles suivantes s'appliquent lors de la migration de rôles d'association :

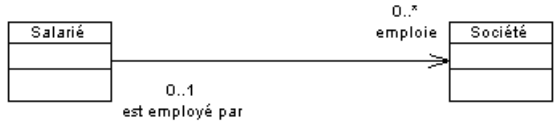
- Si le nom de l'attribut migré est identique au nom du rôle, toute modification du nom de rôle est répercutée sur le nom de l'attribut migré.
- Si le type de données de l'attribut migré est identique au classificateur du rôle, toute modification de la multiplicité du rôle est répercutée sur la multiplicité de l'attribut migré.
- Si le code du classificateur, lié par l'association, change, le type de données de l'attribut migré est automatiquement synchronisé.
- Si vous changez manuellement l'attribut migré, la synchronisation ne fonctionnera pas et le rôle de l'association ne sera pas modifié.
- L'attribut migré est automatiquement supprimé si l'association est supprimée.

A l'issue de la migration, la feuille de propriétés du nouvel attribut affiche le nom de l'association dans la zone Migré depuis l'onglet Détails.

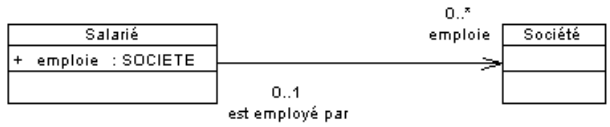
Migration des rôles navigables

Vous pouvez migrer les rôles navigables d'une association et la transformer en attribut :

1. Pointez sur un lien d'association dans le diagramme et cliquez le bouton droit de la souris.



2. Sélectionnez **Migrer > Migrer les rôles navigables** dans le menu contextuel de l'association.



Un attribut est créé et nommé d'après le rôle navigable de l'association, il est suivi du nom du classificateur.

Régénération des liens de type de données

Si un type de données de classificateur n'est pas lié à son classificateur d'origine, vous pouvez utiliser la fonctionnalité de régénération des liens de type de données pour restaurer ce lien. Cette fonctionnalité recherche tous les classificateurs du modèle courant et les lie, si nécessaire, au classificateur d'origine.

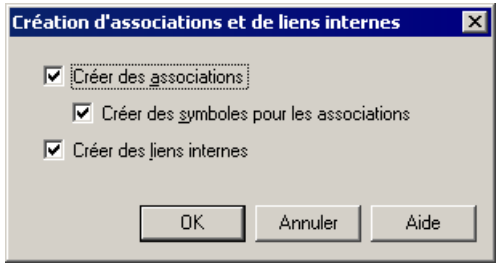
La régénération des liens de type de données examine les types de données suivants :

- Type de données d'attribut : une association est créée et l'attribut est signalé comme étant un attribut migré.
- Type de données de paramètre : une association est créée et liée au classificateur d'origine.
- Type de résultat d'opération : une association est créée et liée au classificateur d'origine.

Dans certains cas, en particulier pour C++, cette fonctionnalité est très utile pour conserver la synchronisation d'un lien même après un changement de type de données lorsqu'il fait toujours référence à la classe d'origine.

La fonctionnalité de régénération des liens de type de données comporte les options suivantes :

1. Sélectionnez **Outils > Régénérer les liens de types de données** pour afficher la fenêtre Création d'associations et de liens internes.



2. Définissez les options suivantes :

Option	Description
Créer des associations	Recherche des attributs dont les types de données correspondent à un classificateur et lie les attributs à la nouvelle association en tant qu'attributs migrés.
Créer des symboles pour les associations	Crée un symbole de la nouvelle association.
Créer des liens internes	Crée un lien entre le type de résultat ou le type de données de paramètre et le classificateur auquel il fait référence.

3. Cliquez sur OK.

Liaison d'une association à un lien entre objets

Vous pouvez faire glisser un noeud d'association depuis l'Explorateur d'objets dans un diagramme de communication ou d'objets. Vous créez alors automatiquement deux objets, ainsi qu'un lien entre eux. Ces deux objets sont des instances des classes ou d'interfaces, et le lien entre objets est une instance de l'association.

Généralisations (MOO)

Une *généralisation* est une relation entre un élément général (le parent) et un élément plus spécifique (l'enfant). L'élément plus spécifique est entièrement cohérent avec l'élément plus général et comporte des informations supplémentaires.

Une généralisation peut être créée dans les types de diagramme suivants :

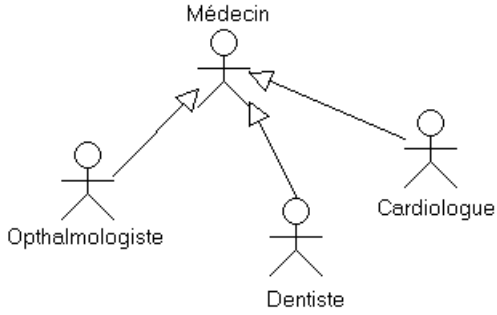
- Diagramme de classes
- Diagramme de structures composites
- Diagramme de composants
- Diagrammes de cas d'utilisation

Vous créez une relation de généralisation lorsque plusieurs objets ont des comportements en commun. Vous avez également la possibilité de créer une généralisation entre un raccourci d'objet et un objet mais, si le lien a un sens, seul l'objet parent peut être le raccourci.

Généralisations ans un diagramme de cas d'utilisation
Dans un diagramme de cas d'utilisation, vous pouvez créer une généralisation entre :

- Deux acteurs
- Deux cas d'utilisation

Par exemple, plusieurs acteurs peuvent présenter des points communs, et communiquer de façon identique avec le même groupe de cas d'utilisation. Cette similarité est exprimée en termes de généralisation vers un autre acteur. Dans ce cas, les acteurs enfant héritent à la fois des rôles et des relations avec les cas d'utilisation définis pour l'acteur parent. Un acteur enfant inclut les attributs et opérations de son ancêtre.

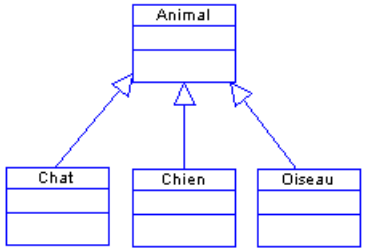


Généralisations dans un diagramme de classes ou de structure composite
Dans un diagramme de classes, vous pouvez créez une généralisation entre :

- Deux classes
- Deux interfaces

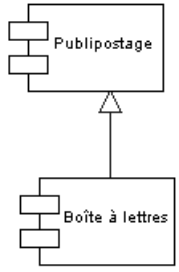
Par exemple, un animal est un concept plus général qu'un chat, un chien ou un oiseau. A l'inverse, un chat est un concept plus spécifique qu'un animal.

Animal est une superclasse. Chat, Chien et Oiseau sont des sous-classes de la superclasse.



Généralisations dans un diagramme de composants

Dans un diagramme de composants, vous pouvez créer une généralisation entre deux composants, comme indiqué ci-dessous :



Création d'une généralisation

Vous pouvez créer une généralisation à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Généralisation** dans la Boîte à outils.
- Sélectionnez **Modèle > Généralisations** pour afficher la boîte de dialogue Liste des généralisations, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Généralisation**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'une généralisation

Pour visualiser ou modifier les propriétés d'une généralisation, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Parent	Spécifie l'objet parent. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Enfant	Spécifie l'objet enfant. Cliquez sur l'outil Propriétés à droite de cette zone pour afficher les propriétés de l'objet sélectionné.
Visibilité	Spécifie la visibilité de l'objet, à savoir la façon dont il est perçu hors de son espace de noms. Vous pouvez choisir parmi les valeurs suivantes : • Private – Uniquement par la généralisation elle-même. • Protected – Uniquement par la généralisation elle-même et par ses objets hérités. • Package – Par tous les objets contenus dans le même package. • Public – Par tous les objets (option par défaut).
Générer la classe parent sous forme de table	Sélectionne l'option de persistance "Générer une table" dans l'onglet Détails de la feuille de propriétés de la classe parent. Si cette option n'est pas sélectionnée, l'option de persistance "Migrer les colonnes" de la classe parent est sélectionnée.
Générer la classe enfant sous forme de table	Sélectionne l'option de persistance "Générer une table" dans l'onglet Détails de la feuille de propriétés de la classe enfant. Si cette option n'est pas sélectionnée, l'option de persistance "Migrer les colonnes" de la classe enfant est sélectionnée.
Attribut discriminant	Spécifie un attribut persistant (avec le stéréotype <<specifying>>) dans la table parent. Cliquez sur l'outil Nouveau pour créer un nouvel attribut. Cet attribut ne sera généré que si la table enfant n'est pas générée.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Si la généralisation est créée dans un diagramme de cas d'utilisation, vous ne pouvez pas changer le type des objets liés par la généralisation. Par exemple, vous ne pouvez pas attacher la dépendance provenant d'un cas d'utilisation à une classe ou à une interface.

Dépendances (MOO)

Une *dépendance* est une relation sémantique entre deux objets dans laquelle toute modification effectuée sur un objet (l'élément influent) affecte l'autre objet (élément dépendant).

Une dépendance peut être créée dans les types de diagramme suivants :

- Diagramme de classes
- Diagramme de structures composites
- Diagramme d'objets
- Diagrammes de cas d'utilisation
- Diagramme de composants
- Diagramme de déploiement

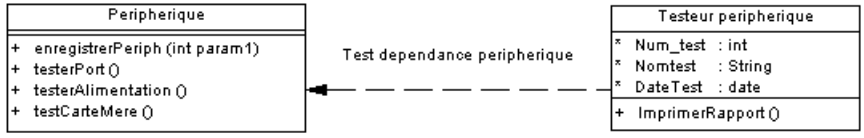
La relation de dépendance indique qu'un objet dans un diagramme utilise les services ou fonctionnalités d'un autre objet. Vous pouvez également définir des dépendances entre un package et un élément de modélisation.

Dépendances dans un diagramme de classes ou de structure composite

Dans un diagramme de classes, vous pouvez créer une dépendance entre :

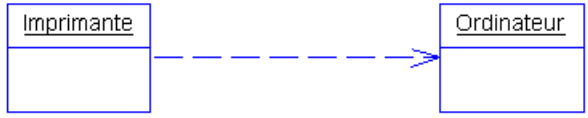
- Une classe et une interface (et réciproquement)
- Deux classes
- Deux interfaces

Par exemple :



Dépendances dans un diagramme d'objets

Dans un diagramme d'objets, vous pouvez créer une dépendance entre deux objets comme suit :

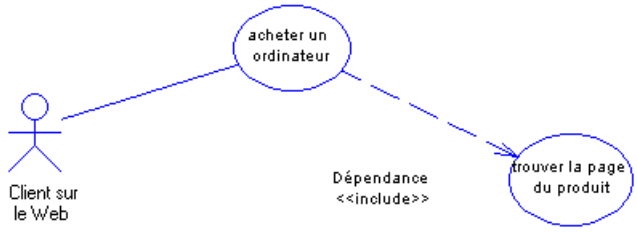


Dépendances dans un diagramme de cas d'utilisation

Dans un diagramme de cas d'utilisation, vous pouvez créer une dépendance entre :

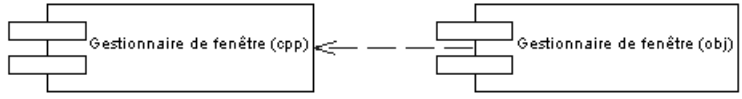
- Un acteur et un cas d'utilisation (et réciproquement)
- Deux acteurs
- Deux cas d'utilisation

Par exemple, l'achat d'un ordinateur via un site Internet implique de trouver l'onglet correspondant au produit sur le site du vendeur :



Dépendances dans un diagramme de composants

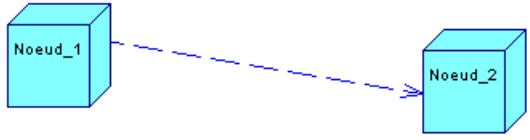
Dans un diagramme de composants, vous pouvez créer une dépendance entre deux composants comme illustré ci-dessous. Vous ne pouvez pas créer une dépendance entre un composant et une interface.



Lorsque vous utilisez une dépendance, vous pouvez imbriquer deux composants en utilisant un stéréotype.

Dépendances dans un diagramme de déploiement

Dans un diagramme de déploiement, vous pouvez créer une dépendance entre des noeuds, ainsi qu'entre des instances de composant, comme illustré ci-dessous :



Création d'une dépendance

Vous pouvez créer une dépendance à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Dépendance** dans la Boîte à outils.
- Sélectionnez **Modèle > Dépendances** pour afficher la boîte de dialogue Liste des dépendances, puis cliquez sur l'outil **Ajouter une ligne**.

- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Dépendance**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'une dépendance

Pour visualiser ou modifier les propriétés d'une dépendance, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.

Propriété	Description
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension. Les stéréotypes communs suivants sont fournis par défaut : • << Access >> - Contenu public du package cible auquel le package source peut accéder. • << Bind >> - Objet source qui instancie le template cible à l'aide des paramètres fournis. • << Call>> - Opération source qui appelle l'opération cible. • << Derive >> - Objet source qui peut être calculé à partir de la cible. • << Extend >> - (Cas d'utilisation/Classe) L'objet cible enrichit le comportement de l'objet source au point d'extension spécifié. • << Friend>> - Objet source doté d'une visibilité spéciale vis-à-vis de la cible. • << Import >> - Tout ce qui est déclaré comme public dans l'objet cible devient visible pour l'objet source, comme si cela faisait partie de la définition de l'objet source. • << Include >> - (Cas d'utilisation/Classe) Inclusion du comportement d'un premier objet dans le comportement de l'objet client, sous le contrôle du premier objet. • << Instantiate >> - Spécifie que les opérations sur la classe source créent des instances de la classe cible. • << Refine >> - Le degré d'abstraction de l'objet source est meilleur que celui de l'objet cible. • << Trace >> - Indique qu'il existe un lien historique entre l'objet source et l'objet cible. • << Use >> - Spécifie que la sémantique de l'objet source dépend de la sémantique de la partie publique de l'objet cible. • << ejb-ref >> - (Java uniquement) Utilisé dans la génération Java pour créer des références aux EJB (beans d'entité et beans de session) pour générer le descripteur de déploiement. • << sameFile >> - (Java uniquement) Utilisé dans la génération Java pour générer des classes Java ayant la visibilité protected ou private au sein d'un fichier correspondant à une classe de visibilité public.
Influent	Le cas d'utilisation ou acteur sélectionné influe sur l'objet dépendant. Les changements apportés sur l'objet influent affectent l'objet dépendant. Cliquez sur l'outil Propriétés à droite de cette zone pour afficher la feuille de propriétés de l'objet.
Dépendant	Le cas d'utilisateur ou acteur sélectionné dépend de l'objet influent. changements apportés sur l'objet dépendant n'affectent pas l'objet influent. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.

Propriété	Description
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Si la dépendance est créée dans un diagramme de cas d'utilisation, vous ne pouvez pas changer les objets liés par la dépendance. Par exemple, vous ne pouvez pas attacher les dépendances provenant d'un cas d'utilisation à une classe ou à une interface.

Réalisations (MOO)

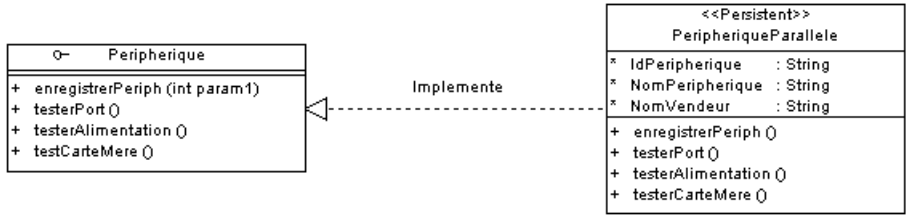
Une réalisation est une relation entre une classe ou un composant et une interface.

Une réalisation peut être créée dans les types de diagramme suivants :

- Diagramme de classes
- Diagramme de composants
- Diagramme de structure composite

Dans une réalisation, la classe met en oeuvre les méthodes spécifiées dans l'interface. L'interface est appelée élément de spécification et la classe est appelée élément de mise en oeuvre. Vous avez également la possibilité de créer une réalisation entre le raccourci d'une interface et une classe. Quand le lien est orienté, seul l'objet parent peut être le raccourci.

La pointe de la flèche située à l'une des extrémités de la réalisation pointe toujours vers l'interface.



Création d'une réalisation

Vous pouvez créer une réalisation à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Réalisation** dans la Boîte à outils.
- Sélectionnez **Modèle > Réalisations** pour afficher la boîte de dialogue Liste des réalisations, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Réalisations**

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'une réalisation

Pour visualiser ou modifier les propriétés d'une réalisation, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Interface	Nom de l'interface qui effectue la réalisation. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Classe	Nom de la classe pour laquelle la réalisation est effectuée.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

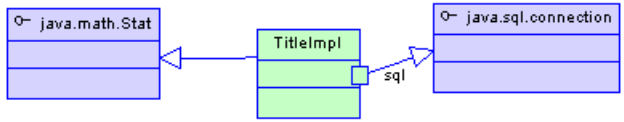
Liens de prérequis (MOO)

Les liens de prérequis connectent les classificateurs et les interfaces. Un *lien de prérequis* peut connecter une classe, un composant ou un port situé à l'extérieur de l'un de ces classificateurs à une classe.

Un lien de prérequis peut être créé dans les types de diagramme suivants :

- Diagramme de classes
- Diagramme de composants
- Diagramme de structure composite

Liens de prérequis dans un diagramme de classes
Dans l'exemple ci-dessous, les liens de prérequis connectent la classe TitleImpl aux interfaces java.math.stat et java.sql.connection. Notez comment un lien de prérequis peut provenir d'un port ou directement de la classe :



Création d'un lien de prérequis

Vous pouvez créer un lien de prérequis à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Lien de prérequis/Connecteur** dans la Boîte à outils.
- Sélectionnez **Modèle > Liens de prérequis** pour afficher la boîte de dialogue Liste des liens de prérequis, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Lien de prérequis**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'un lien de prérequis

Pour visualiser ou modifier les propriétés d'un lien de prérequis, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Interface	Spécifie l'interface à laquelle lier. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Client	Spécifie la classe, le port ou le composant à lier.

Propriété	Description
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

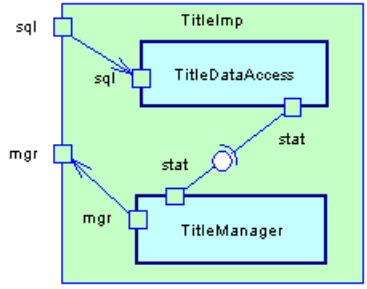
Connecteurs d'assemblage (MOO)

Les connecteurs d'assemblage représentent les chemins de communication par le biais desquels vos classificateurs se demandent et se fournissent des services.

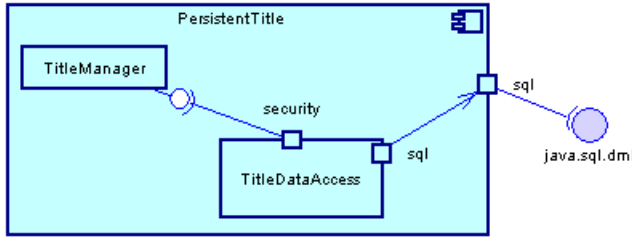
Un connecteur d'assemblage peut être créé dans les types de diagramme suivants :

- Diagramme de composants
- Diagramme de structure composite

Connecteurs d'assemblage dans un diagramme de structure composite
Dans l'exemple ci-dessous, un connecteur d'assemblage connecte la partie fournisseur "TitleDataAccess" à la partie client "TitleManager".



Connecteurs d'assemblage dans un diagramme de composant
Dans l'exemple ci-dessous, un connecteur d'assemblage connecte la partie fournisseur "TitleDataAccess" à la partie client "TitleManager".



Création d'un connecteur d'assemblage

Vous pouvez créer un connecteur d'assemblage en utilisant l'outil **Lien de prérequis/ Connecteur** dans la Boîte à outils.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'un connecteur d'assemblage

Pour visualiser ou modifier les propriétés d'un connecteur d'assemblage, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Fournisseur	Spécifie la partie qui fournit le service. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Client	Spécifie la partie qui demande le service. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Interface	Spécifie l'interface que la partie fournisseur utilise pour fournir le service. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Connecteurs de délégation (MOO)

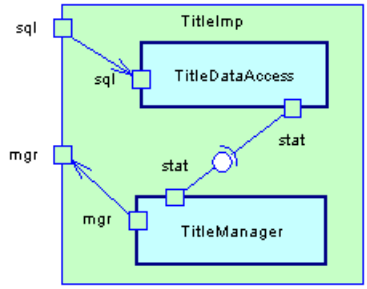
Les connecteurs de délégation représentent les chemins de communication par le biais desquels les parties situées à l'intérieur d'un classificateur sont connectées aux ports situés à l'extérieur de ce classificateur et se demandent et se fournissent des services.

Un connecteur de délégation peut être créé dans les types de diagramme suivants :

- Diagramme de composants
- Diagramme de structure composite

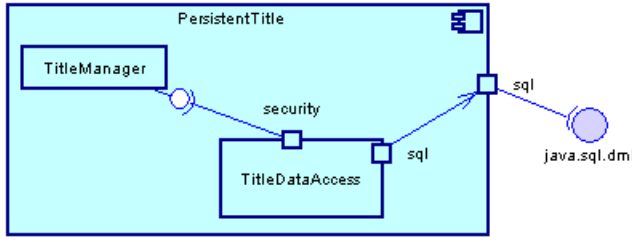
Connecteurs de délégation dans un diagramme de structure composite

Dans l'exemple ci-dessous, un connecteur de délégation connecte le port fournisseur "sql" à l'extérieur de la classe "TitleImp" à la partie client "TitleDataAccess" via un port "sql". Un second connecteur de délégation connecte la partie fournisseur "TitleManager" via le port "mgr" au port "mgr" à l'extérieur de la classe "TitleImp".



Connecteurs de délégation dans un diagramme de composants

Dans l'exemple ci-dessous un connecteur de délégation connecte la partie fournisseur "TitleDataAccess" via le port "sql" au port client "sql" à l'extérieur du composant "PersistentTitle".



Création d'un connecteur de délégation

Vous pouvez créer un connecteur de délégation en utilisant l'outil **Lien de prérequis/ Connecteur** dans la Boîte à outils.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'un connecteur de délégation

Pour visualiser ou modifier les propriétés d'un connecteur de délégation, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Fournisseur	Spécifie la partie ou le port qui fournit le service. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Client	Spécifie la partie ou le port qui demande le service. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Annotations (MOO)

Les langages Java 5.0 et . NET (C# 2.0 et VB 2005) fournissent des méthodes permettant d'ajouter des métadonnées dans le code. PowerAMC prend en charge complètement les

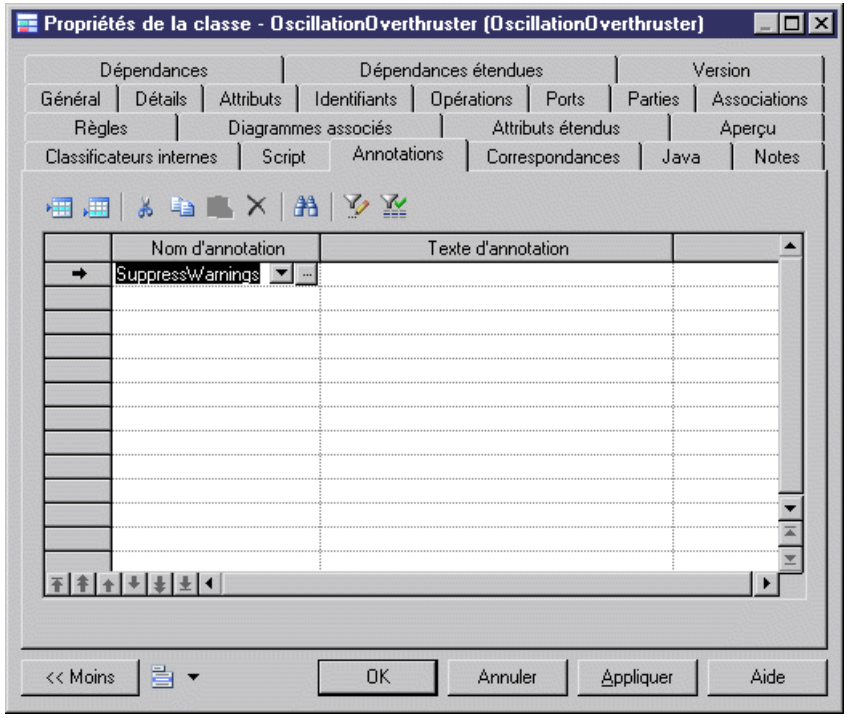
annotations Java et les attributs personnalisés .NET. En ce qui concerne les informations particulières à chaque langage, veuillez vous reporter au chapitre approprié dans la seconde partie de ce manuel.

Ces métadonnées sont accessibles aux outils de post-traitement ou lors de l'exécution afin de faire varier le comportement du système.

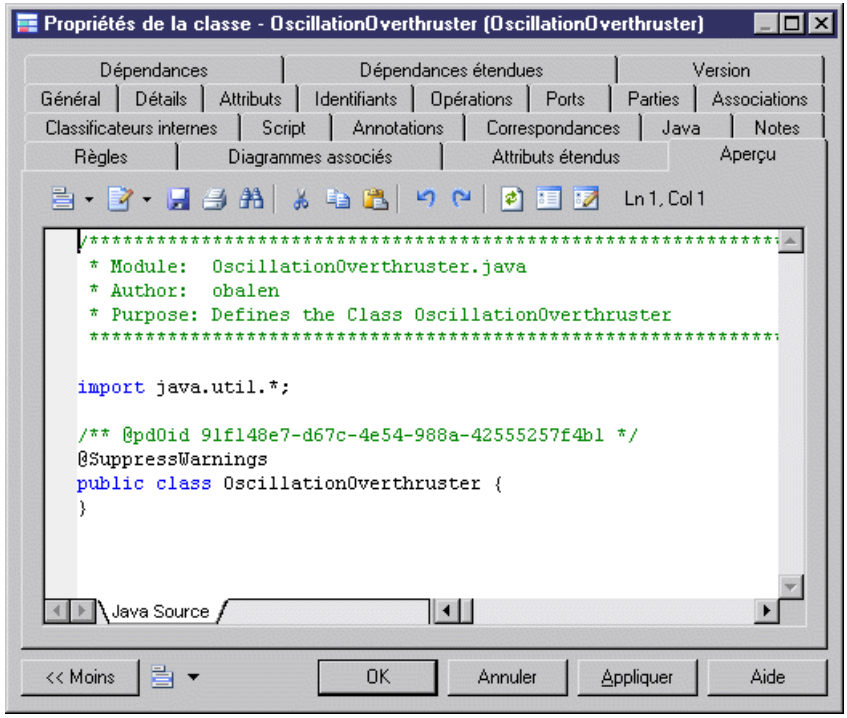
Affectation d'une annotation à un objet de modèle

PowerAMC prend en charge les annotations intégrées de Java 5.0, les attributs personnalisés .NET 2.0 et permet, que ce soit pour Java 5.0 et .NET 2.0, de créer les vôtres. Vous pouvez affecter des annotations à des types et à d'autres objets de modèle :

1. Double-cliquez sur une classe ou un autre objet pour afficher sa feuille de propriétés, puis cliquez sur l'onglet Annotations.
2. Cliquez sur la colonne Nom d'annotation, puis sélectionnez une annotation dans la liste.



3. Si l'annotation nécessite des paramètres, vous pouvez les saisir directement dans la colonne Texte de l'annotation, ou bien cliquer sur le bouton Points de suspension pour afficher l'Editeur d'annotations.
4. [facultatif] Cliquez sur l'onglet Aperçu pour voir le code qui sera généré pour la classe, avec sa déclaration précédée par l'annotation :

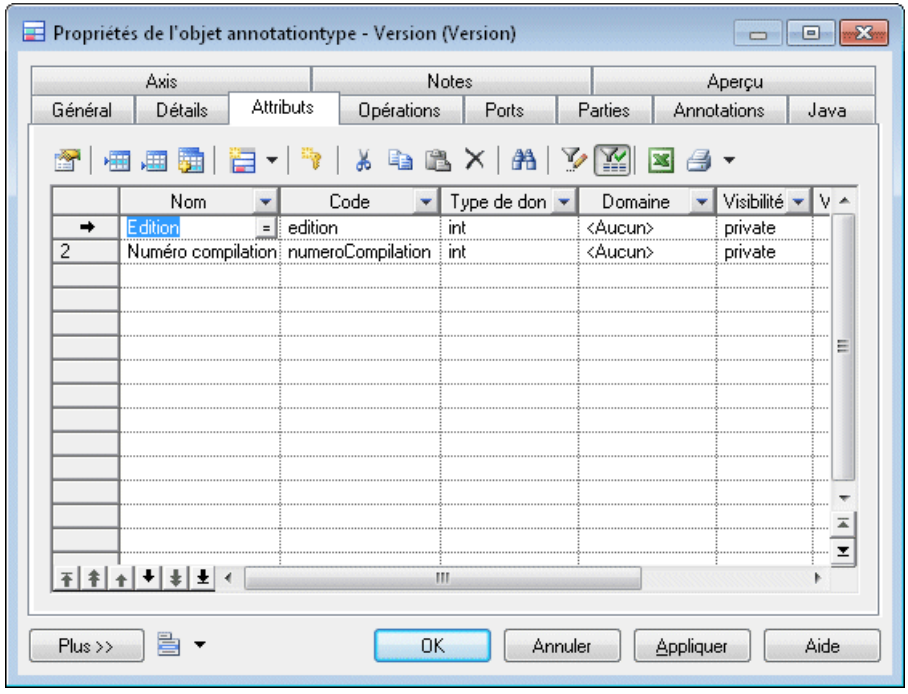


5. Cliquez sur OK pour revenir au diagramme.

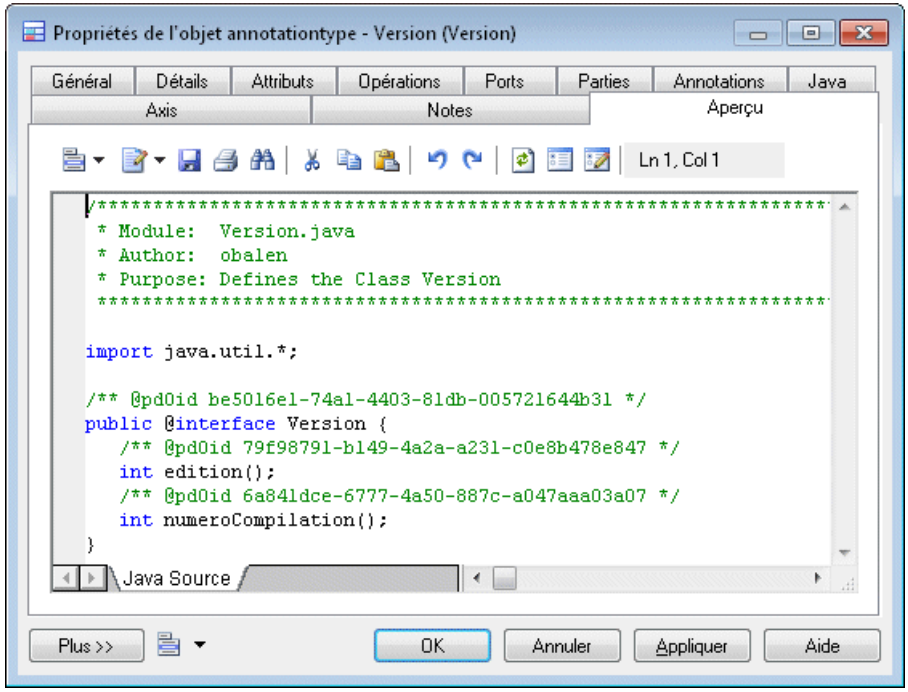
Création d'un nouveau type d'annotation

Vous pouvez créer des nouveaux types d'annotation à rattacher à vos objets.

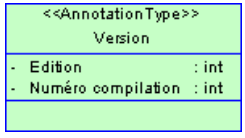
1. Créez une classe, puis double-cliquez sur cette classe pour afficher sa feuille de propriétés.
2. Sur l'onglet Général, dans la liste Stéréotype :
   - Pour Java 5.0 - sélectionnez AnnotationType.
   - Pour .NET 2.0 - sélectionnez AttributeType.
3. Cliquez sur l'onglet Attributs, et ajoutez un attribut pour chaque paramètre accepté par le type d'annotation.



4. [facultatif] Cliquez sur l'onglet Aperçu pour voir le code à générer pour le type d'annotation :

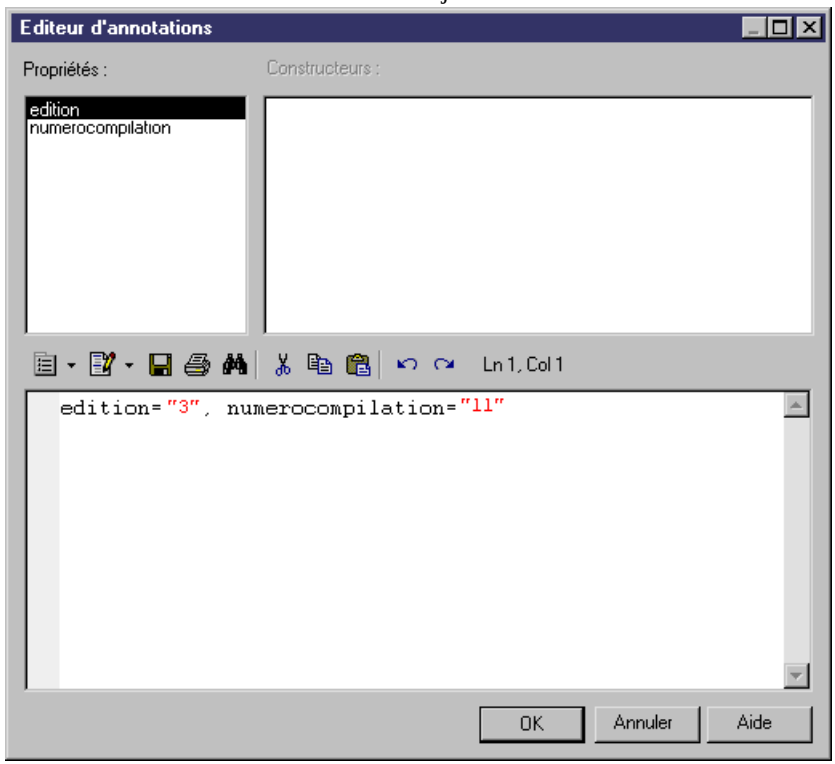


5. Cliquez sur OK pour revenir au diagramme. Le type d'annotation sera présenté comme suit :



Utilisation de l'Editeur d'annotations

L'Editeur d'annotations vous aide à spécifier les valeurs pour les paramètres d'annotation. Pour y accéder, cliquez sur le bouton Points de suspension dans la colonne Texte de l'annotation d'une classe ou d'un autre objet :



Les volets supérieurs fournissent des listes propriétés et constructeurs disponibles, sur lesquels vous pouvez double-cliquer pour les ajouter dans le volet du bas, qui sert à l'édition.

Liens entre objets (MOO)

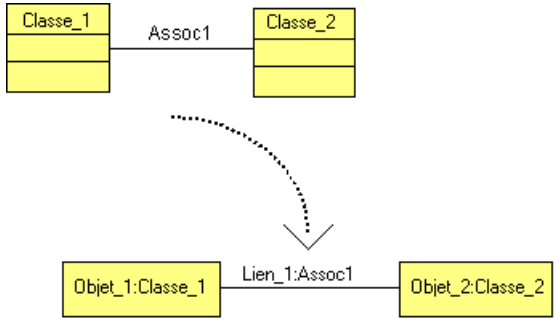
Un *lien entre objets* représente une connexion entre deux objets. Il s'affiche sous la forme d'un trait plein entre deux objets.

Un lien entre objets peut être créé dans les types de diagramme suivants :

- Diagramme de communication
- Diagramme d'objets

Lien entre objets dans un diagramme d'objets

Les liens entre objets ont une relation forte avec les associations du diagramme de classes : les associations entre classes, ou les associations entre une classe et une interface peuvent devenir des liens entre objets (instances d'associations) entre objets dans le diagramme d'objets. En outre, le symbole du lien entre objets dans le diagramme d'objets est similaire à celui de l'association dans le diagramme de classes, à ceci près qu'il est dépourvu de cardinalités.

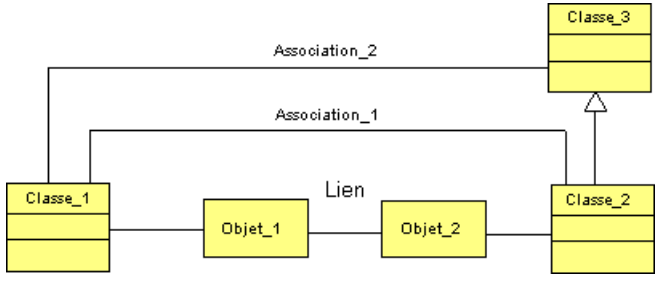


Les rôles du lien entre objets sont dupliqués depuis les rôles de l'association. Un lien entre objets peut donc être une agrégation ou une composition, tout comme une association du diagramme de classes. Si tel est le cas, le symbole de composition ou d'agrégation s'affiche sur le symbole du lien entre objets. Les rôles de l'association sont également affichés sur le symbole du lien entre objets si vous avez activé l'option Noms de rôle de l'association dans les préférences d'affichage des liens entre objets.

Exemple

L'illustration suivante montre Objet_1 comme instance de Classe_1, et Objet_2 comme instance de Classe_2. Ces objets sont liés par un lien entre objets. Classe_1 et Classe_2 sont liées par une association. En outre, attendu que Classe_2 est associée à Classe_1 et hérite de Classe_3, Classe_1 et Classe_3 sont par conséquent liées par une association.

Le lien entre objets entre Objet_1 et Objet_2 dans l'illustration représente Association_1 ou Association_2.



Vous pouvez également utiliser des raccourcis d'associations, à condition toutefois que le modèle auquel ils renvoient soit ouvert dans l'espace de travail.

Comportement des liens entre objets

Les règles suivantes s'appliquent aux liens entre objets :

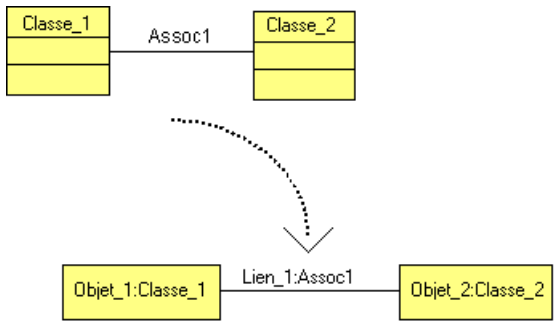
- Lorsqu'une association entre classes devient un lien entre objets, les deux classes liées par l'association et les deux classes des objets liés par le lien entre objets doivent correspondre (ou bien la classe de l'objet doit hériter des classes parent liées par l'association). Ceci est également valable concernant une association entre une classe et une interface.
- Deux liens entre objets peuvent être définis entre les mêmes objets source et destination (*liens entre objets parallèles*). Si vous fusionnez les deux modèles, la fonctionnalité de fusion différencie les liens entre objets parallèles en fonction des associations de diagramme de classes auxquelles ils correspondent.
- Vous pouvez lier un objet à lui-même via un lien réflexif (même objet source et destination)

Lien entre objets dans un diagramme de communication

Un lien entre objets représente une connexion entre deux objets, il permet de mettre en exergue la collaboration entre objets, d'où le nom de diagramme de communication. Il est représenté sous forme d'un trait plein entre :

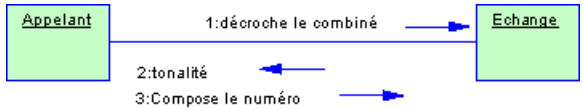
- Deux objets
- Un objet et un acteur (ou vice-versa)

Un lien entre objets peut être une instance d'association entre classes, ou une association entre une classe et une interface.



Le rôle du lien entre objets provient de l'association. Le nom d'un lien entre objets inclut les noms des objets situés à ses extrémités, ainsi que celui de l'association.

Le symbole du lien entre objets peut inclure plusieurs symboles de message associés.



Les liens entre objets contiennent une liste de messages ordonnée : le numéro d'ordre qui spécifie l'ordre dans lequel les messages sont échangés entre objets. Pour plus d'informations, voir *Messages (MOO)* à la page 144.

Comportement des liens entre objets

Les règles suivantes s'appliquent aux liens entre objets :

- Vous pouvez utiliser un lien entre objets récursif (même objet source et destination).
- Deux liens entre objets peuvent être définis entre les mêmes objets source et destination (*liens entre objets parallèles*).
- Lorsque vous supprimez un lien entre objets, ses messages sont aussi supprimés si aucun diagramme de séquence ne les utilise.
- Lorsqu'une association entre classes est transformée en lien entre objets, les deux classes liées par l'association, et les deux classes des objets liés par le lien entre objets doivent correspondre (ou bien la classe de l'objet doit hériter des classes parent liées par l'association). Ceci est également valable concernant l'association entre une classe et une interface.
- Si vous changez une extrémité d'une association, le lien entre objets qui vient de l'association est détaché.
- Lorsque vous copiez et collez, ou que vous déplacez, un lien entre objets, ses messages sont automatiquement copiés.
- Lorsque les extrémités du message changent, le message est détaché du lien entre objets.
- Si vous utilisez la fonctionnalité Afficher les symboles pour afficher le symbole d'un lien entre objets, tous les messages attachés au lien entre objets sont affichés.

Création d'un lien entre objets

Vous pouvez créer un lien entre objets à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Lien entre objets** dans la Boîte à outils.
- Sélectionnez **Modèle > Liens entre objets** pour afficher la boîte de dialogue Liste des liens entre objets, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Lien entre objets**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Glisser-déposer d'associations dans un diagramme d'objets

Vous pouvez faire glisser une association depuis l'Explorateur d'objets dans un diagramme de communication. Vous créez ainsi un lien entre objets en tant qu'instance de l'association, et deux objets constituant des instances de ces classes ou interfaces.

Propriétés d'un lien entre objets

Pour visualiser ou modifier les propriétés d'un lien entre objets, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom et le code sont en lecture seule. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Objet A	Nom de l'objet à une extrémité du lien entre objets. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Objet B	Nom de l'objet à l'autre extrémité du lien entre objets. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Association	Association entre classes (ou association entre une classe et une interface) que le lien entre objets utilise pour communiquer entre les objets de ces classes. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Domaines (MOO)

Les domaines vous aident à identifier le type des informations contenues dans votre projet. Ils définissent un ensemble de valeurs pour lesquelles un attribut est correct. L'utilisation de domaines sur des attributs permet d'uniformiser les caractéristiques de données des attributs contenus dans différentes classes.

Un domaine peut être créé dans les types de diagramme suivants :

- Diagramme de classes

Dans un MOO, vous pouvez définir les informations suivantes en affectant un domaine :

- Type de données

- Paramètres de contrôle
- Règles de gestion

Création d'un domaine

Vous pouvez créer un domaine à partir de l'Explorateur d'objets ou du menu **Modèle**.

- Sélectionnez **Modèle > Domaines** pour afficher la boîte de dialogue Liste des domaines, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Domaine**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'un domaine

Pour visualiser ou modifier les propriétés d'un domaine, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet Général contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Type de données	Format des données correspondant au domaine (par exemple, numérique, alphanumérique, booléen, etc)

Propriété	Description
Multiplicité	Spécification de la plage des nombres de valeurs admises que peut prendre un attribut utilisant ce domaine. La multiplicité d'un domaine est utile lorsque vous travaillez par exemple avec un attribut multiple. La multiplicité fait partie du type de données et la multiplicité et le type de données peuvent être fournis par le domaine. Vous pouvez choisir parmi les valeurs suivantes : • 0..1 – Zéro ou une. • 0..* – Aucune à un nombre illimité. • 1..1 – Une et une seule. • 1..* – Une à un nombre illimité. • * – Aucune ou un nombre illimité.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Onglet Détails

L'onglet Détails contient une zone de groupe Persistant dont l'objet est d'améliorer la génération des codes et des types de données lors de la génération d'un modèle MCD ou d'un MPD à partir d'un MPD, et il contient les propriétés suivantes :

Propriété	Description
Persistant	Zone de groupe pour la génération de modèles MCD ou MPD persistants. Définit un modèle comme persistant (voir *Gestion de la persistance des objets lors de la génération de modèles de données* à la page 295).
Type de données	Spécifie un type de données persistant utilisé dans la génération d'un modèle persistant, soit un MCD soit un MPD. Le type de données persistant est défini à partir des types de données conceptuels par défaut de PowerAMC.
Longueur	Spécifie le nombre maximal de caractères du type de données persistant.
Précision	Spécifie le nombre de décimales pour les valeurs de type de données persistant qui comportent des chiffres après la virgule.

Les onglets suivants sont également disponibles :

- Contrôles standard - contient les vérifications qui contrôlent les valeurs permises pour le domaine (voir *Définition de contraintes de profilage de données* à la page 76)
- Contrôles supplémentaires - permet de spécifier des contraintes supplémentaires (non définies par les paramètres de contrôle standard) pour le domaine.
- Règles - répertorie les règles de gestion associées au domaine (voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets > Règles de gestion*).

Les tableaux suivants fournissent des détails sur les types de données disponibles :

Types de données numériques

Type de données	Contenu	Longueur	Précision obligatoire
Entier	Entier 32 bits	—	—
Entier court	Entier 16 bits	—	—
Entier long	Entier 32 bits	—	—
Octet	Valeurs comprises entre 1 et 256	—	—
Numérique	Nombre à décimale fixe	Fixe	
Décimal	Nombre à décimale fixe	Fixe	
Réel	Nombre en virgule flottante 32 bits	Fixe	—
Réel court	Nombre en virgule flottante de moins de 32 bits	—	—
Réel long	Nombre en virgule flottante de 64 bits	—	—
Monnaie	Nombre à décimale fixe	Fixe	
Séquentiel	Nombre incrémenté automatiquement	Fixe	—
Booléen	Deux valeurs opposées (vrai/faux ; oui/non ; 1/0)	—	—

Chaînes de caractères

Type de données	Contenu	Longueur
Caractère alpha	Chaînes de caractères	Fixe
Caractère variable	Chaînes de caractères	Maximum
Caractère long	Chaînes de caractères	Maximum
Caractère long var.	Chaînes de caractères	Maximum
Texte	Chaînes de caractères	Maximum
Multibyte	Chaînes de caractères sur plusieurs octets	Fixe
Multibyte variable	Chaînes de caractères sur plusieurs octets	Maximum

Date et heure

Type de données	Contenu
Date	Jour, mois et année
Heure	Heure, minute et seconde
Date et heure	Date et heure
Date système	Date et heure système

Autres types de données

Type de données	Contenu	Longueur
Binaire	Chaînes binaires	Maximum
Binaire long	Chaînes binaires	Maximum
Bitmap	Images au format bitmap (BMP)	Maximum
Image	Images	Maximum
OLE	Liaisons OLE	Maximum
Autre	Types de données définis par l'utilisateur	—
Non défini	Type de données non encore défini	—

Mise à jour d'attributs à l'aide d'un domaine dans un MOO

Lorsque vous modifiez un domaine, vous pouvez choisir de mettre à jour automatiquement les propriétés suivantes pour les colonnes qui utilisent ce domaine :

- Type de données
- Paramètres de contrôle
- Règles de gestion

1. Sélectionnez **Modèle > Domaines** pour afficher la boîte de dialogue Liste des domaines.
2. Cliquez sur un domaine dans la liste, puis cliquez sur l'outil Propriétés pour afficher sa feuille de propriétés.

   Remarque : Vous pouvez également accéder à la feuille de propriétés d'un domaine en double-cliquant sur le domaine approprié dans l'Explorateur d'objets.
3. Saisissez ou sélectionnez les propriétés de domaine appropriées sur les différents onglets, puis cliquez sur OK. Si le domaine est utilisé par un ou plusieurs attributs, une boîte de

confirmation s'affiche pour vous demander si vous souhaitez modifier les propriétés des attributs qui utilisent ce domaine.

4. Sélectionnez les propriétés que vous souhaitez voir mises à jour pour les attributs utilisant le domaine, puis cliquez sur Oui.

Chapitre 4 Diagrammes dynamiques

Les diagrammes décrits dans ce chapitre permettent de modéliser le comportement dynamique de votre système, et d'indiquer comment ses objets interagissent au moment de l'exécution. PowerAMC propose quatre types de diagrammes pour la modélisation de cet aspect de votre système :

- Un *diagramme de communication* qui représente un cas d'utilisation particulier ou une fonctionnalité du système en termes d'interactions entre des objets, tout en se focalisant sur la structure de l'objet. Pour plus d'informations, voir *Diagrammes de communication* à la page 131.
- Un *diagramme de séquence* qui représente un cas d'utilisation particulier ou une fonctionnalité du système en termes d'interactions entre des objets, tout en se focalisant sur l'ordre chronologique des messages échangés. Pour plus d'informations, voir *Diagrammes de séquence* à la page 134.
- Un *diagramme d'activités* qui représente un cas d'utilisation particulier ou une fonctionnalité du système en termes d'actions ou d'activités effectuées et de transitions déclenchées par l'accomplissement de ces actions. Il permet également de représenter des branches conditionnelles. Pour plus d'informations, voir *Diagrammes d'activités* à la page 137.
- Un *diagramme d'états-transitions* qui représente un cas d'utilisation particulier ou une fonctionnalité du système en utilisant les états par lesquels passe un classificateur et les transitions entre ces états. Pour plus d'informations, voir *Diagrammes d'états-transitions* à la page 140.
- Un *diagramme d'interactions* qui fournit une représentation de haut niveau des interactions se produisant dans votre système. Pour plus d'informations, voir *Diagrammes d'interactions* à la page 143.

Diagrammes de communication

Un *diagramme de communication* est un diagramme UML qui fournit une représentation graphique des interactions entre les objets d'un scénario de cas d'utilisation, l'exécution d'une opération, ou une interaction entre des classes, en mettant l'accent sur la structure du système.

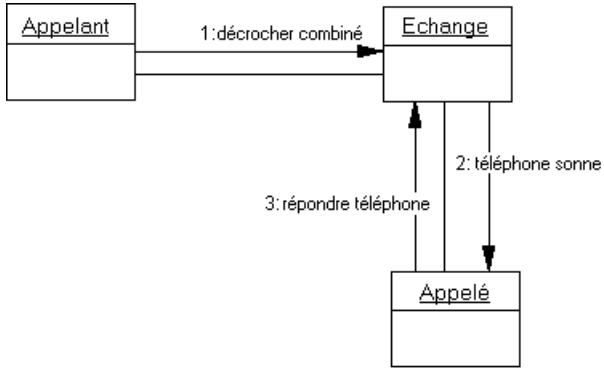
Remarque : Pour créer un diagramme de communication dans un MOO existant, pointez sur le modèle dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Diagramme de communication**. Pour créer un nouveau modèle, sélectionnez **Fichier > Nouveau modèle**, choisissez Modèle Orienté Objet comme type de modèle et **Diagramme de communication** comme premier diagramme, puis cliquez sur **OK**. Pour créer un diagramme de communication qui réutilise les objets et messages d'un diagramme de séquence existant (et créer des liens entre les objets qui communiquent par le biais de

messages, en mettant met également à jour les numéros d'ordre des messages, en se basant sur la position relative de ces messages sur la ligne de vie), pointez sur le diagramme de séquence, cliquez le bouton droit de la souris puis sélectionnez **Créer un diagramme de communication par défaut**, ou bien sélectionnez **Outils > Créer un diagramme de communication par défaut**. Notez que ces deux diagrammes ne restent pas synchronisés, les changements effectués dans un diagramme ne seront pas répercutés dans l'autre.

Vous pouvez utiliser un ou plusieurs diagrammes de communication pour mettre en scène un cas d'utilisation ou pour identifier toutes les possibilités d'un comportement complexe.

Un diagramme de communication véhicule le même genre d'information qu'un diagramme de séquence, à ceci près qu'il se concentre sur la structure des objets au lieu de la chronologie des messages qu'ils échangent.

Un diagramme de communication montre des acteurs, des objets (instances de classes) et leurs liens de communication (appelés liens entre objets), ainsi que les messages qu'ils échangent. Les messages sont définis sur des liens entre objets qui correspondent à un lien de communication entre deux objets qui interagissent. L'ordre dans lequel les messages sont échangés est représenté par les numéros d'ordre.



Analyse d'un cas d'utilisation

Un diagramme de communication peut être utilisé pour modéliser le comportement d'un cas d'utilisation, pour affiner la description d'un cas d'utilisation ou enrichir la définition d'un diagramme de classes en utilisant une approche itérative. Cette approche est utile lors de l'analyse des besoins car elle peut aider à identifier des classes et associations qui ne s'étaient pas imposées immédiatement.

Vous pouvez formaliser l'association entre le cas d'utilisation et le diagramme de communication en ajoutant le diagramme dans l'onglet Diagrammes associés de la feuille de propriétés du cas d'utilisation.

Il est souvent nécessaire de créer plusieurs diagrammes afin de décrire tous les scénarios possibles d'un cas d'utilisation. Dans ce cas de figure, il peut être utile d'utiliser les diagrammes de communication pour découvrir tous les objets pertinents avant de tenter

d'identifier les classes qui vont les instancier. Une fois ces classes identifiées, vous pouvez ensuite déduire les associations entre elles grâce aux liens entre objets.

La principale difficulté de cette approche consiste à identifier les objets appropriés afin de transcrire les pas d'action du cas d'utilisation. Une extension d'UML, "Robustness Analysis" peut simplifier ce processus. Cette méthode préconise de séparer les objets en 3 types :

- Les objets Boundary sont utilisés par les acteurs lorsqu'ils communiquent avec le système, il peut s'agir de fenêtres, d'écrans, de boîtes de dialogue ou de menus.
- Les objets Entity représentent des données stockées telles qu'une base de données, des tables de base de données ou tout type d'objet temporaire tel qu'un résultat de recherche.
- Les objets Control sont utilisés pour contrôler les objets Boundary et Entity, et qui représentent un transfert d'informations.

PowerAMC prend en charge l'extension Robustness Analysis via un fichier d'extension (voir *Personnalisation et extension de PowerAMC > Fichiers d'extension > Exemple : Création d'extensions de diagramme de robustesse*.

Analyse d'un diagramme de classe

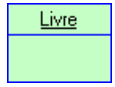
La construction d'un diagramme de communication peut également être l'opportunité de tester un modèle statique pendant la conception ; il peut représenter un scénario dans lequel les classes du diagramme de classes sont instanciées pour créer les objets nécessaires à l'exécution du scénario.

Il complète le diagramme de classes qui représente la structure statique d'un système en spécifiant le comportement des classes et interfaces ainsi que l'utilisation possible de leurs opérations.

Vous pouvez créer les objets et les liens entre objets nécessaires automatiquement en sélectionnant les classes et associations appropriées dans un diagramme de classes, puis en appuyant sur **Ctrl+Maj** et en les faisant glisser dans un diagramme de communication vide. Il vous suffit ensuite d'ajouter les messages nécessaires.

Objets du diagramme de communication

PowerAMC prend en charge tous les objets nécessaires pour construire des diagramme de communication.

Objet	Outil	Symbole	Description
Acteur			Personne externe, un processus ou quelque chose interagissant avec un système, un sous-système ou une classe. Voir *Acteurs (MOO)* à la page 21.
Objet			Instance d'une classe. Voir *Objets (MOO)* à la page 58.

Objet	Outil	Symbole	Description
Lien entre objets			Lien de communication entre deux objets. Voir *Liens entre objets (MOO)* à la page 120.
Message			Interaction qui convoie des informations avec l'espoir qu'une action en découle. Crée un lien entre objets par défaut lorsqu'il n'en existe pas. Voir *Messages (MOO)* à la page 144.

Diagrammes de séquence

Un *diagramme de séquence* est un diagramme UML qui fournit une représentation graphique de la technologie d'échange de messages entre des objets et des acteurs pour un cas d'utilisation, l'exécution d'une opération, ou une interaction des classes, en mettant l'accent sur leur chronologie.

Remarque : Pour créer un diagramme de séquence dans un MOO existant, pointez sur le modèle dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Diagramme de séquence**. Pour créer un nouveau modèle, sélectionnez **Fichier > Nouveau modèle**, choisissez Modèle Orienté Objet comme type de modèle et **Diagramme de séquence** comme premier diagramme, puis cliquez sur **OK**. Pour créer un diagramme de séquence qui réutilise les objets et messages d'un diagramme de communication existant, pointez sur le diagramme de communication, cliquez le bouton droit de la souris, puis sélectionnez **Créer un diagramme de séquence par défaut**, ou bien sélectionnez **Outils > Créer un diagramme de séquence par défaut**. Notez que les deux diagrammes ne restent pas synchronisés – les changements effectués dans un diagramme ne seront pas répercutés dans l'autre.

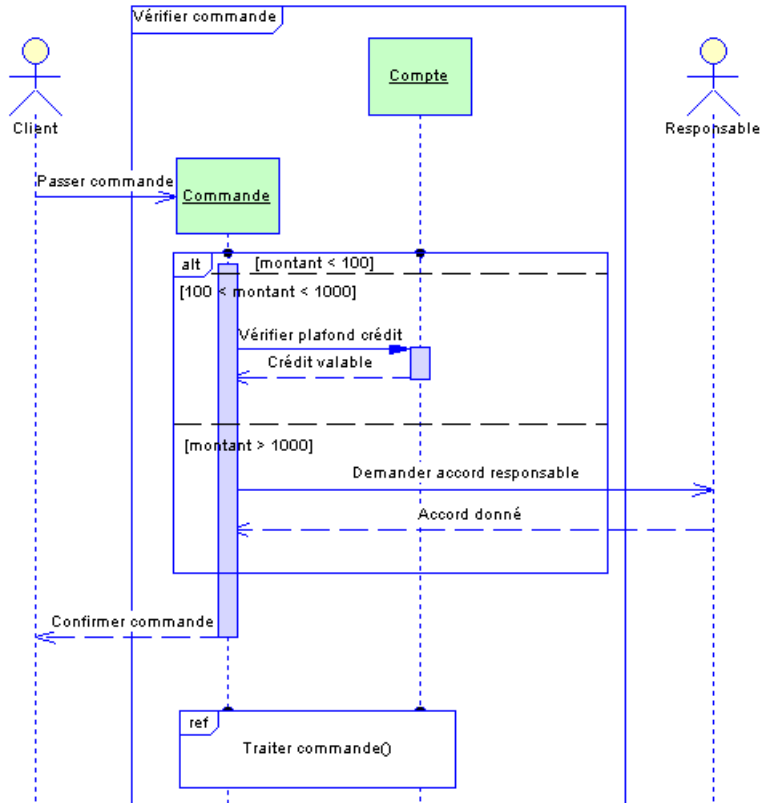
Vous pouvez utiliser un ou plusieurs diagrammes de séquence pour mettre en scène un cas d'utilisation ou pour identifier toutes les possibilités d'un comportement complexe.

Un diagramme de séquence montre des acteurs, des objets (instances de classes) et les messages qu'ils échangent. Il véhicule le même genre d'informations qu'un diagramme de communication, à ceci près qu'il se concentre sur la chronologie des messages échangés entre les objets plutôt que sur la structure des objets.

Par défaut, PowerAMC fournit un "cadre d'interaction", qui entoure les objets dans le diagramme et agit comme la limite extérieure du système (ou d'une partie de ce système) en cours de modélisation. Les messages peuvent avoir comme origine ou destination n'importe quel endroit de ce cadre d'interaction, et ces *portes* peuvent être utilisées à la place des acteurs (voir *Messages et portes* à la page 155). Vous pouvez supprimer le cadre en sélectionnant **Outils > Préférences d'affichage**, en sélectionnant la catégorie **Cadre d'interaction** et en décochant l'option **Symbole d'interaction**. Pour obtenir des informations détaillées sur l'utilisation des préférences d'affichage *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Diagrammes, matrices et symboles > Préférences d'affichage*.

L'un des principaux avantages d'un diagramme de séquence sur un diagramme de communication est qu'il permet de référencer des interactions courantes et de spécifier facilement des scénarios alternatifs ou parallèles en utilisant des fragments d'interaction. Ainsi, vous pouvez décrire dans un seul diagramme de séquence un nombre d'interactions liées qui nécessiterait plusieurs diagrammes de communication.

Dans l'exemple ci-dessus, l'acteur Client passe une commande. Le message Passer commande crée un objet Commande. Un fragment d'interaction gère diverses possibilités de vérifier la commande. L'objet Compte et l'auteur Responsable peuvent interagir avec la commande en fonction de sa taille. Une fois le message Confirmer commande envoyé, l'interaction Traiter commande est initiée. Cette interaction est stockée dans un autre diagramme de séquence, et est représentée ici par une référence d'interaction :



Analyse d'un cas d'utilisation

Un diagramme de séquence peut être utilisé pour affiner le comportement ou la description d'un cas d'utilisation. Cette approche est utile lors de l'analyse des besoins car elle peut aider à identifier des classes et associations qui ne s'étaient pas imposées immédiatement.

Vous pouvez formaliser l'association entre le cas d'utilisation et le diagramme de séquence en ajoutant le diagramme dans l'onglet Diagrammes associés de la feuille de propriétés du cas d'utilisation.

Il est souvent nécessaire de créer plusieurs diagrammes afin de décrire tous les scénarios possibles d'un cas d'utilisation. Dans ce cas de figure, il peut être utile d'utiliser les diagrammes de séquence pour découvrir tous les objets pertinents avant de tenter d'identifier les classes qui vont les instancier. Une fois ces classes identifiées, vous pouvez ensuite déduire les associations entre elles grâce aux liens entre objets.

Analyse d'un diagramme de classe

La construction d'un diagramme de séquence peut également être l'opportunité de tester un modèle statique pendant la conception ; il peut représenter un scénario dans lequel les classes du diagramme de classes sont instanciées pour créer les objets nécessaires à l'exécution du scénario.

Il complète le diagramme de classes qui représente la structure statique d'un système en spécifiant le comportement des classes et interfaces ainsi que l'utilisation possible de leurs opérations.

Un diagramme de séquence permet d'analyser des opérations de classe avec plus de précision qu'un diagramme de communication. Vous pouvez créer une opération dans la classe d'un objet qui reçoit un message par le biais de la feuille de propriétés du message. Vous pouvez également effectuer une telle tâche dans un diagramme de communication, mais il y a plus d'espace dans un diagramme de séquence pour afficher des informations détaillées (arguments, valeur de résultat, etc.) relatives à l'opération.

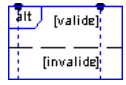
Remarque : Les fonctionnalités Disposition automatique, Aligner et Grouper les symboles ne sont pas disponibles dans le diagramme de séquence.

Lorsque vous utilisez la fonctionnalité de fusion de modèles sur des diagrammes de séquence, les symboles de tous les éléments du diagramme de séquence sont fusionnés sans comparaison. Vous avez le choix d'accepter toutes les modifications sur tous les symboles ou de refuser toute modification.

Objets du diagramme de séquence

PowerAMC prend en charge tous les objets nécessaires pour construire des diagrammes de séquence.

Objet	Outil	Symbole	Description
Acteur			Personne externe, un processus ou quelque chose interagissant avec un système, un sous-système ou une classe. Voir *Acteurs (MOO)* à la page 21.

Objet	Outil	Symbole	Description
Objet			Instance d'une classe. Voir *Objets (MOO)* à la page 58.
Activation			Exécution d'une procédure, y compris le temps nécessaire à l'exécution des procédures qui y sont imbriquées. Voir *Activation (MOO)* à la page 159.
Référence d'interaction			Référence à un autre diagramme de séquence. Voir *Références d'interaction et activités d'interaction (MOO)* à la page 163.
Fragment d'interaction			Collection de messages associés. Voir *Fragments d'interaction (MOO)* à la page 165.
Message			Communication qui convoie des informations avec l'espoir qu'une action en découle. Voir *Messages (MOO)* à la page 144.
Message réflexif			Message récursif ayant le même objet comme émetteur et récepteur. Voir *Messages (MOO)* à la page 144.
Message d'appel de procédure			Message d'appel de procédure avec une activation par défaut. Voir *Messages (MOO)* à la page 144.
Message d'appel réflexif			Message récursif d'appel de procédure avec une activation par défaut. Voir *Messages (MOO)* à la page 144.
Message de retour			Spécifie la fin d'une procédure. Généralement associé à un appel de procédure, le message de retour peut être omis car il est implicite à la fin d'une activation. Voir *Messages (MOO)* à la page 144.
Message de retour réflexif			Message récursif avec un type de flux de contrôle Retour. Voir *Messages (MOO)* à la page 144.

Diagrammes d'activités

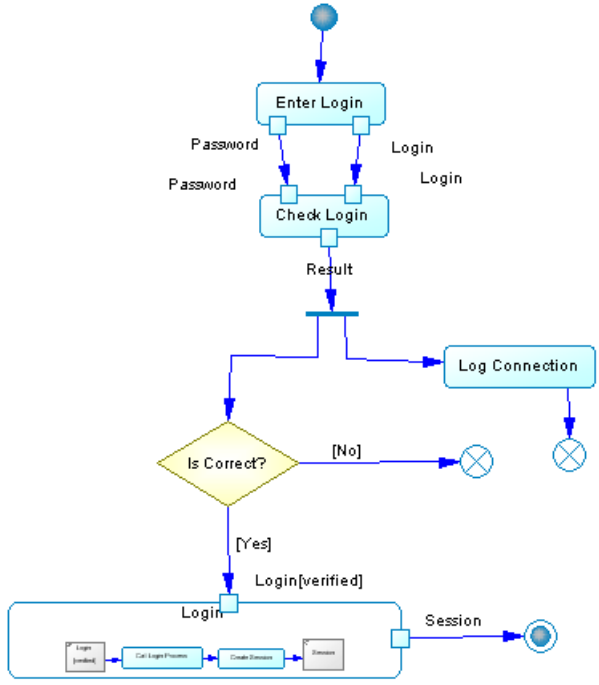
Un *diagramme d'activités* est un diagramme UML qui fournit une représentation graphique du comportement d'un système, et vous aide à le décomposer de façon fonctionnelle afin d'analyser sa mise en oeuvre.

Remarque : Pour créer un diagramme d'activités dans un MOO existant, pointez sur le modèle dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau** > **Diagramme d'activités**. Pour créer un nouveau modèle, sélectionnez **Fichier** >

Nouveau modèle, choisissez Modèle Orienté Objet comme type de modèle et **Diagramme d'activités** comme premier diagramme, puis cliquez sur **OK**.

Alors qu'un diagramme d'états-transitions se focalise sur la mise en oeuvre des opérations dans lesquelles la plupart des événements correspondent précisément à la fin de l'activité précédente, le diagramme d'activités ne fait pas la distinction entre les états, les activités et les événements.

Le diagramme d'activités donne une représentation simplifiée d'un processus, en montrant les flux de contrôle (appelés transitions) entre les actions effectuées dans le système (appelées activités). Ces flux représentent le comportement interne d'un élément de modèle (cas d'utilisation, package, classificateur ou opération) depuis un point de début jusqu'à plusieurs points de fin potentiels.



Vous pouvez créer plusieurs diagrammes d'activités dans un package ou dans un modèle. Chacun de ces diagrammes est indépendant et définit un contexte isolé dans lequel l'intégrité des éléments peut être vérifiée.

Analyse d'un cas d'utilisation

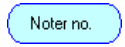
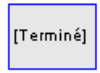
Un diagramme d'activités est fréquemment utilisé pour décrire de façon graphique un cas d'utilisation. Chaque activité correspond à une étape d'action et les points d'extension peuvent être représentés comme des branches conditionnelles.

Analyse d'un processus métiers
Au-delà de la modélisation orientée objets, les diagrammes d'activité sont de plus en plus souvent utilisés pour modéliser les aspects métiers de l'entreprise. Ce type de modélisation se produit avant la modélisation UML classique d'une application, et permet l'identification des processus importants de l'entreprise et les domaines de responsabilité pour chaque unité organisationnelle au sein de l'entreprise.

Pour plus d'informations sur la modélisation des processus métiers à l'aide de PowerAMC, voir *Modélisation des processus métiers*.

Objets du diagramme d'activités

PowerAMC prend en charge tous les objets nécessaires pour construire des diagrammes d'activités.

Objet	Outil	Symbole	Description
Début			Point de départ des activités représentées dans le diagramme d'activités. Voir *Débuts et fins (MOO)* à la page 190.
Activité			Appel d'une action. Voir *Activités (MOO)* à la page 169.
Activité décomposée	—		Activité complexe décomposée pour être détaillée. Voir *Activités (MOO)* à la page 169.
Noeud d'objet			Représente un état particulier d'une activité. Voir *Noeuds d'objet (MOO)* à la page 200.
Unité d'organisation			Société, système, service, organisation, utilisateur ou rôle. Voir *Unités d'organisation (MOO)* à la page 183.
Flux			Chemin emprunté par le flux de contrôle pour joindre les activités. Voir *Flux (MOO)* à la page 197.
Décision			Décision que le flux de contrôle doit prendre lorsque plusieurs itinéraires de transition sont possibles. Voir *Décisions (MOO)* à la page 192.
Synchronisation			Permet la synchronisation du contrôle entre plusieurs actions concurrentes. Voir *Synchronisations (MOO)* à la page 195.
Fin			Point de fin des activités représentées dans le diagramme d'activités. Voir *Débuts et fins (MOO)* à la page 190.

Diagrammes d'états-transitions

Un *diagramme d'états-transitions* est un diagramme UML qui fournit une représentation graphique d'une State Machine, le comportement public d'un classificateur (composant ou classe), sous la forme des changements de l'état du classificateur et des événements qui permettent la transition d'un état à l'autre.

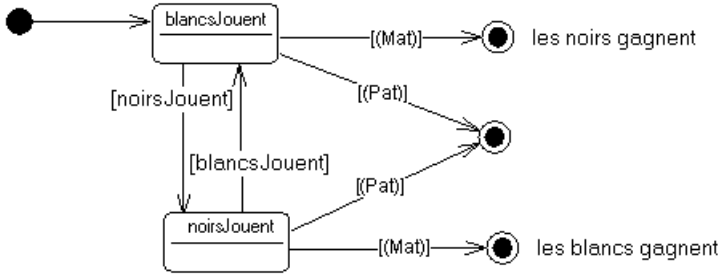
Remarque : Pour créer un diagramme d'états-transitions dans un MOO existant, pointez sur le modèle dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau** > **Diagramme d'états-transitions**. Pour créer un nouveau modèle, sélectionnez **Fichier** > **Nouveau modèle**, choisissez Modèle Orienté Objet comme type de modèle et **Diagramme d'états-transitions** comme premier diagramme, puis cliquez sur **OK**.

Le classificateur est censé avoir déjà été identifié dans un autre diagramme et tous ses états doivent pouvoir être répertoriés.

A la différence des autres diagrammes d'interaction, le diagramme d'états-transitions peut représenter une spécification complète des scénarios possibles appartenant au classificateur. A un moment donné, l'objet doit être dans l'un des états définis.

Vous pouvez créer plusieurs diagrammes d'états-transitions pour le même classificateur, mais alors les états et les transitions représentés doivent être liés à différents aspects de son évolution. Par exemple, une personne peut être considérée d'un côté comme se déplaçant entre les états étudiant, travailleur, sans emploi et en retraite, et de l'autre étant successivement célibataire, fiancé, marié et divorcé.

Les diagrammes d'états-transitions montrent le comportement du classificateur via des règles d'exécution qui expliquent précisément de quelle façon les actions sont exécutées lors des transitions entre les différents états ; ces états correspondent aux différentes situations se produisant lors de la vie du classificateur.



L'exemple ci-dessus montre les états d'un jeu d'échecs.

La première étape dans la création d'un diagramme d'états-transitions consiste à définir les états initial et final et le jeu d'états possibles entre eux. Vous liez ensuite les états à l'aide de

transitions, en notant sur chacune d'entre elles l'événement qui déclenche le passage d'un état à l'autre.

Vous pouvez également définir une action qui s'exécute au moment de la transition. De même, l'entrée ou la sortie d'un état peut provoquer l'exécution d'une action. Il est même possible de définir les événements internes qui ne changent pas l'état. Les actions peuvent être associées aux opérations du classificateur décrit par le diagramme.

Il est également possible de décomposer les états complexes en sous-états, qui sont représentés dans des sous-diagrammes d'états-transitions.

Un diagramme d'états-transitions requiert l'identification préalable d'un classificateur. Il peut être utilisé pour décrire le comportement du classificateur, et vous aide également à découvrir ses opérations via la spécification des actions associées aux événements d'états-transitions.

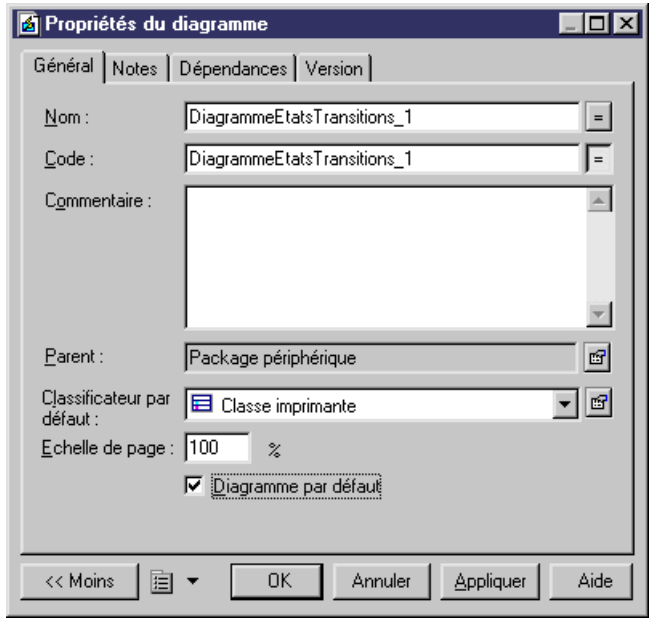
Vous pouvez également utiliser les transitions identifiées afin d'établir l'ordre dans lequel les opérations peuvent être appelées. Ce type de diagramme est appelé "protocol state machine".

La spécification de l'interface utilisateur graphique (GUI, Graphic User Interface) peut être une autre utilisation. Les états sont les différents écrans disponibles avec les transitions entre eux, le tout dépendant des événements de clavier et de souris produits par l'utilisateur.

Définition d'un classificateur par défaut dans le diagramme d'états-transitions

Vous pouvez définir le classificateur d'un état en utilisant la liste Classificateur dans la feuille de propriétés de l'état. Cette opération permet de lier l'état à un cas d'utilisation, à un composant ou à une classe.

Au niveau du diagramme, vous avez également la possibilité de spécifier le contexte d'un état en renseignant la zone Classificateur par défaut dans la feuille de propriétés du diagramme d'états-transitions. Chaque état qui est alors créé dans un diagramme à l'aide de l'outil Etat est automatiquement associé au classificateur par défaut spécifié dans la feuille de propriétés du diagramme d'états-transitions.



Les nouveaux diagrammes sont créés par défaut avec une valeur vide dans la liste Classificateur par défaut, sauf les diagrammes de sous-état qui partagent automatiquement la valeur Classificateur définie pour l'état composite parent. La valeur Classificateur par défaut est une valeur facultative dans le diagramme d'états-transitions.

Objets du diagramme d'états-transitions

PowerAMC prend en charge tous les objets nécessaires pour construire des diagrammes d'états-transitions.

Objet	Outil	Symbole	Description
Début			Point de départ des états représentés dans le diagramme d'états-transitions. Voir *Débuts et fins (MOO)* à la page 190.
Etat		Vérification	Situation d'un élément de modèle dans l'attente d'événements. Voir *Etats (MOO)* à la page 202.
Action	—	—	Spécification d'une instruction pouvant être exécutée. Voir *Actions (MOO)* à la page 213.
Evénement	—	—	Occurrence d'un élément observable, convoie des informations spécifiées par des paramètres. Voir *Evénements (MOO)* à la page 210.
Transition			Chemin emprunté par le flux de contrôle pour joindre les états. Voir *Transitions (MOO)* à la page 207.

Objet	Outil	Symbole	Description
Point de jonction			Divise une transition entre états. Utilisé particulièrement lorsque vous spécifiez des conditions mutuellement exclusives. Voir *Points de jonction (MOO)* à la page 216.
Synchronisation			Permet de scinder ou de synchroniser le contrôle entre deux états concurrents. Voir *Synchronisations (MOO)* à la page 195.
Fin			Point de fin des états représentés dans le diagramme d'états-transitions. Voir *Débuts et fins (MOO)* à la page 190.

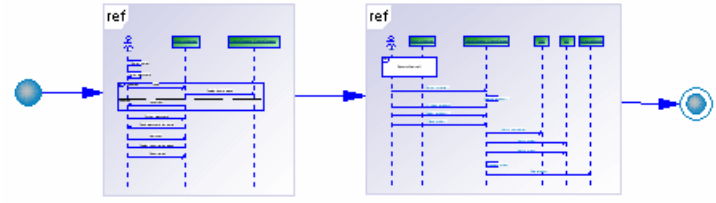
Diagrammes d'interactions

Un *diagramme d'interactions* est un diagramme UML qui fournit une représentation graphique de haut niveau du flux de contrôle de votre système alors que ce dernier est décomposé en diagrammes de séquence et autres diagrammes d'interactions.

Remarque : Pour créer un diagramme d'interactions dans un MOO existant, pointez sur le modèle dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Diagramme d'interactions**. Pour créer un nouveau modèle, sélectionnez **Fichier > Nouveau modèle**, choisissez Modèle Orienté Objet comme type de modèle et **Diagramme d'interactions** comme premier diagramme, puis cliquez sur **OK**.

Vous pouvez inclure des références vers des diagrammes de séquence, des diagrammes de communication et d'autres diagrammes d'interactions.

Dans l'exemple suivant, un flux de contrôle lie deux diagrammes de séquence :



Objets du diagramme d'interactions

PowerAMC prend en charge tous les objets nécessaires pour construire des diagrammes d'interactions.

Objet	Outil	Symbole	Description
Début			Point de départ des interactions représentées dans le diagramme. Voir *Débuts et fins (MOO)* à la page 190.
Activité d'interaction			Référence à un diagramme de séquence, à un diagramme de communication ou à un diagramme d'interactions. Voir *Références d'interaction et activités d'interaction (MOO)* à la page 163.
Flux			Flux de contrôle entre deux interactions. Voir *Flux (MOO)* à la page 197.
Décision			Décision que le flux de contrôle doit prendre lorsque plusieurs itinéraires sont possibles. Voir *Décisions (MOO)* à la page 192.
Synchronisation			Permet la scission ou la synchronisation du contrôle entre plusieurs flux. Voir *Synchronisations (MOO)* à la page 195.
Fin			Point de fin des interactions représentées dans le diagramme. Voir *Débuts et fins (MOO)* à la page 190.

Messages (MOO)

Un *message* est une communication entre objets. La réception d'un message produit généralement un résultat.

Un message peut être créé dans les diagrammes suivants :

- Diagramme de communication
- Diagramme de séquence

Les objets peuvent coopérer à l'aide de différents types de requêtes (envoi d'un signal, appel d'une opération, création d'un objet, suppression d'un objet existant, etc.). L'envoi d'un signal permet de déclencher une réaction de la part du récepteur de façon asynchrone et sans qu'une réponse ne soit nécessaire. L'appel d'une opération va appliquer une opération à un objet de façon synchrone ou asynchrone, et peut requérir une réponse de la part de son récepteur. Toutes ces requêtes constituent des *messages*. Ces messages correspondent aux stimulus dans le langage UML.

Un message a un émetteur, un récepteur et une action. L'*émetteur* est l'objet ou l'acteur qui envoie le message. Le *récepteur* est l'objet ou l'acteur qui reçoit le message. L'action est exécutée sur le récepteur. Vous pouvez également créer des messages récursifs, c'est-à-dire des messages dont l'émetteur est en même temps le récepteur.

Le symbole d'un message est une flèche qui montre son sens, et qui peut également afficher les informations suivantes :

- Un numéro d'ordre qui indique l'ordre dans lequel les messages sont échangés (voir *Numéros d'ordre* à la page 156)
- Le nom du message (ou nom de l'opération associée).
- La condition.
- La valeur de résultat.
- L'argument.

Réutilisation des messages

Le même message peut être utilisé à la fois dans un diagramme de séquence et de communication ou dans plusieurs autres diagrammes de l'un ou l'autre de ces types de diagramme. Lorsque vous faites glisser un message d'un diagramme vers un autre, vous récupérez en même temps ses deux extrémités si elles n'existent pas (dans un diagramme de communication), et ce message est attaché à un lien entre objets par défaut.

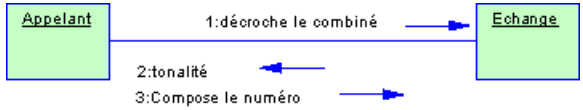
Le numéro d'ordre attaché à un message est identique dans tous les diagrammes si le message est réutilisé.

Lorsque vous copiez un message, son nom ne change pas. Vous pouvez conserver son nom ou bien le modifier à l'issue de la copie.

Toute modification apportée à la valeur Action ou Flux de contrôle d'un message sera répercutée dans tous les diagrammes. Cependant, si le changement que vous souhaitez apporter n'est pas valide, il ne sera pas effectué. Par exemple, vous n'êtes pas autorisé à déplacer un message Création si un message Création existe déjà entre l'émetteur et le récepteur.

Messages dans un diagramme de communication

Dans un diagramme de communication, chaque message est associé à un lien entre objets. Un lien entre objets peut avoir plusieurs messages associés, mais chacun de ces messages ne peut être associés qu'à un seul lien entre objets. La destruction d'un lien entre objets détruit tous les messages qui lui sont associés.



Messages dans un diagramme de séquence

Un message est affiché sous la forme d'une flèche horizontale pleine partant de la ligne de vie d'un objet ou d'un acteur pour aller à la ligne de vie d'un autre objet ou acteur. La flèche est

légendée par le nom du message. Vous pouvez également définir un *type de flux de contrôle* qui représente à la fois la relation entre une action et les actions qui le précèdent et le suivent, ainsi que la sémantique d'attente entre eux.

Dans un diagramme de séquence, vous pouvez choisir parmi les types de message suivants :

- Message
- Message récursif
- Message d'appel de procédure
- Message d'appel réflexif
- Message de retour
- Message de retour réflexif

Vous pouvez créer des activations sur la ligne de vie d'un objet pour représenter la période pendant laquelle il effectue une action.

Un message peut être tracé depuis un acteur vers un objet, ou inversement. Il est également possible de créer un message entre deux acteurs, mais il sera détecté et provoquera un avertissement lors du processus de vérification de modèle.

Remarque : Si vous devez décrire un peu plus un message ou lui adjoindre un libellé, vous pouvez rédiger une note à l'aide de l'outil Note, puis la placer à proximité du message.

Création d'un message

Vous pouvez créer un message à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Message** (ou, dans un diagramme de séquence, sur l'outil **Message récursif**, **Message d'appel de procédure**, ou **Message d'appel réflexif**) dans la Boîte à outils.
- Sélectionnez **Modèle > Messages** pour afficher la boîte de dialogue Liste des messages, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Message**.
- Affichez la feuille de propriétés d'un lien entre objets (dans un diagramme de communication), cliquez sur l'onglet **Messages**, puis cliquez sur l'outil **Créer un nouveau message**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'un message

Pour visualiser ou modifier les propriétés d'un message, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Emetteur	Objet d'origine du message. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Récepteur	Objet sur lequel le message se termine. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné. Vous pouvez également inverser le sens du message. **Remarque :** Vous pouvez pointer sur un message dans le diagramme, cliquer le bouton droit de la souris et sélectionner **Inverser** pour inverser les sens du message. Vous ne pouvez pas inverser le sens d'un message Création ou Destruction.
Numéro d'ordre	Permet d'ajouter manuellement un numéro d'ordre au message. Principalement utilisé dans des diagrammes de communication pour décrire l'ordre des messages, mais peut également être utilisé dans des diagrammes de séquence.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Onglet Détails

L'onglet **Détails** inclut les propriétés suivantes :

Propriété	Description
Action	Spécifie le type d'action du message. Vous pouvez choisir parmi les valeurs suivantes : • Création – l'objet émetteur instancie et initialise l'objet récepteur. Un message avec une action Création est le premier message entre un émetteur et un récepteur • Destruction – l'objet émetteur détruit l'objet récepteur. Un grand X est affiché sur la ligne de vie de l'objet récepteur. Un message avec une action Destruction est le dernier message entre un émetteur et un récepteur. • Auto-destruction – (disponible uniquement si la propriété Flux de contrôle est définie à "Retour") l'objet émetteur avertit l'objet récepteur qu'il se détruit. Un grand X est affiché sur la ligne de vie de l'objet émetteur. Un message avec une action Auto-destruction est le dernier message entre un émetteur et un récepteur
Flux de contrôle	Spécifie le mode d'envoi des messages. Vous pouvez choisir parmi les valeurs suivantes : • Asynchrone – l'objet émetteur n'attend pas un résultat et peut donc faire autre chose en parallèle. *No-wait semantics* • Appel de procédure – Appel d'une procédure. La séquence est terminée avant que la séquence suivante ne commence. L'émetteur doit attendre une réponse ou la fin de l'activation. *Wait semantics* • Retour – Généralement associé à un Appel de procédure. La flèche Retour peut être omise puisqu'elle est implicite à la fin d'une activation • Non défini – Aucun flux de contrôle n'est défini

Propriété	Description
Opération	Lie le message à une opération d'une classe. Si le récepteur d'un message est un objet, et si cet objet a une classe, le message, en tant que flux dynamique d'information, appelle une opération. Vous pouvez donc lier un message à une opération existante d'une classe mais également aux opérations définies sur les classes parent, ou bien vous pouvez créer une nouvelle opération à partir de la liste Opération dans la feuille de propriétés du message. Si une opération est liée à un message, vous pouvez remplacer le nom du message par le nom de la méthode qu'un objet demande à l'autre d'appeler. Cette façon de procéder peut s'avérer très utile lors de la phase de mise en oeuvre. Pour afficher le nom de l'opération au lieu du nom du message, sélectionnez l'option Remplacer par le nom de l'opération dans les préférences d'affichage de message. Vous avez même la possibilité de lier un message Création à une opération Constructeur d'une classe si vous souhaitez détailler encore plus avant la relation entre un message et une opération. Vous n'êtes toutefois pas autorisé à lier un message avec un flux de contrôle Retour à une opération. Si vous modifiez la généralisation qui existe entre les classes, l'opération qui est liée au message peut ne plus être disponible. Dans ce cas, l'opération est automatiquement détachée du message. Il en va de même si vous inversez le sens du message, à moins toutefois que le nouvel objet récepteur ait la même classe.
Arguments	Arguments de l'opération.
Valeur de résultat	Valeur de résultat de fonction stockée dans une variable et qui peut être utilisée par d'autres fonctions.
Liste des prédécesseurs	Composée d'une liste de numéros d'ordre suivie de '/ ', la liste des prédécesseurs définit quels messages doivent être échangés avant que le message courant ne puisse être envoyé. Exemple : numéros de séquence 1, 2, 4 avant 3 = '1,2,4/ 3'.
Condition	Condition associée au message. Peut être spécifiée en plaçant une expression booléenne entre crochets sur le diagramme. Exemple : Condition pour le minutage : [durée de numérotation < 30 sec].
Heure de début	Alias d'heure défini par l'utilisateur, utilisé pour définir des contraintes. Exemple : Heure de début=t1, Heure de fin=t2, contrainte={t2 - t1 < 30 sec}.
Heure de fin	Alias d'heure défini par l'utilisateur, utilisé pour définir des contraintes.

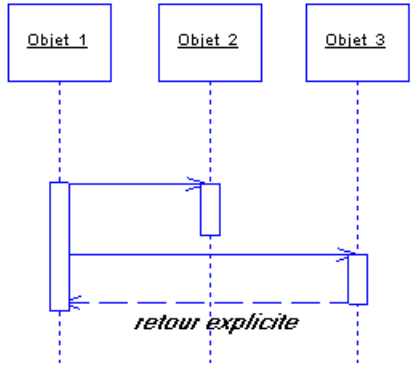
Propriété	Description
Pris en charge du retard	Spécifie que le message peut avoir une durée. Le symbole du message sera incliné, et toujours orienté vers le bas. Si vous avez désélectionné cette option, le message est instantané, ou rapide, et le symbole de message sera toujours horizontal dans le diagramme Vous pouvez sélectionner également l'option Prise en charge du retard comme option par défaut dans la boîte de dialogue Options du modèle. L'option Prise en charge du retard n'est pas disponible avec un message récursif, cette option apparaît alors sélectionnée et grisée.

Flux de contrôle

Un message a par défaut un flux de contrôle Non défini

Si vous souhaitez rendre un diagramme plus lisible, vous pouvez tracer une flèche Retour pour illustrer le moment exact auquel l'action est renvoyée à l'émetteur. Il s'agit d'un retour explicite qui se traduit par le renvoi d'une valeur à son point d'origine.

Dans l'exemple ci-dessous, le retour explicite provoque le renvoi des valeurs à leur activation d'origine.



Vous pouvez combiner des flux de contrôle de message et des actions de message en fonction du tableau ci-après :

Flux de contrôle	Symbole	Aucune action	Création	Destruction	Auto-destruction
Asynchrone					Oui
Appel de procédure					Oui
Retour			Oui	Oui	

Flux de contrôle	Symbole	Aucune action	Création	Destruction	Auto-destruction
Non défini	→				Oui

Remarque : Vous pouvez accéder aux valeurs Action et Flux de contrôle d'un message. Pour ce faire, pointez sur le symbole du message dans le diagramme et cliquez le bouton droit de la souris, puis sélectionnez Action/Flux de contrôle dans le menu contextuel

Création de messages Création et Destruction dans un diagramme de séquence

Les messages Création et Destruction sont spécifiés via la propriété Action située sur l'onglet Détails de leur feuille de propriétés.

Création d'un message Création

Un message peut créer un objet s'il s'agit du premier message reçu par l'objet et que vous avez spécifié la valeur "Création" pour la propriété Action.

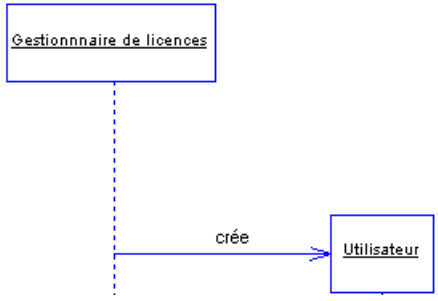
Vous ne pouvez pas créer un acteur ou utiliser l'action Création avec un message récursif.

Lorsqu'un message crée un objet, ce message est tracé avec sa flèche sur l'objet ; l'objet et le message sont situés au même niveau.

Dans un diagramme de séquence, vous pouvez également créer un message Création comme suit :

1. Sélectionnez l'outil **Message** dans la Boîte à outils.
2. Cliquez sur la ligne de vie de l'émetteur et maintenez le bouton de la souris enfoncé. Faites glisser le curseur diagonalement sur le symbole du récepteur.
3. Relâchez le bouton de la souris au-dessus du symbole du récepteur (et non sur sa ligne de vie).

   Si le message est le premier message a être reçu par l'objet récepteur, le symbole de l'objet récepteur se déplace vers le bas et vient s'aligner sur la flèche du message.



Création d'un message Destruction

Un message peut détruire un objet s'il s'agit du dernier message reçu par l'objet et si vous spécifiez la valeur Destruction à sa propriété Action.

Vous ne pouvez pas détruire un acteur ou utiliser l'action Destruction avec un message récursif.

En cas de destruction, un grand X s'affiche au point d'intersection entre la ligne de vie de l'objet et le message. L'action Destruction termine à la fois l'activation et la ligne de vie de l'objet au même point.

L'action Destruction ne détruit pas l'objet, elle se contente de représenter la destruction de l'objet dans le diagramme. La ligne de vie de l'objet s'interrompt à un moment précis dans le temps ; il n'est pas possible de prolonger sa ligne de vie vers le bas.

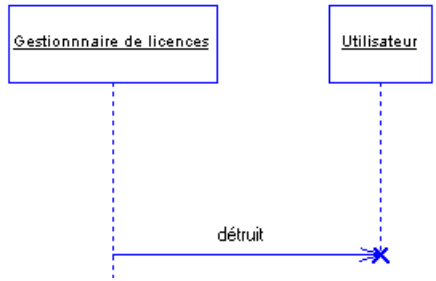
Il existe deux types de destruction :

- Destruction
- Auto-destruction

Création d'un message Destruction

Vous pouvez créer un message Destruction à partir de la Boîte à outils.

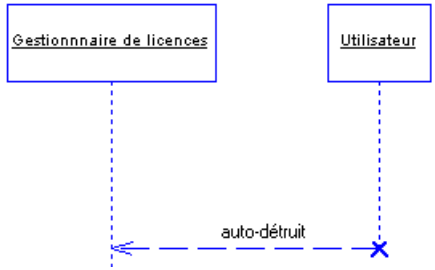
1. Sélectionnez l'outil **Message** dans la Boîte à outils.
2. Cliquez sur la ligne de vie de l'émetteur et maintenez le bouton de la souris enfoncé. Faites glisser le curseur sur la ligne de vie du récepteur.
3. Relâchez le bouton de la souris sur la ligne de vie du récepteur, puis double-cliquez sur le symbole du nouveau message pour afficher sa feuille de propriétés.
4. Sélectionnez **Destruction** dans la liste **Action** de l'onglet **Détails** (cette action n'est pas disponible si le message n'est pas le dernier de la ligne de vie du récepteur).
5. Cliquez sur **OK**. La ligne de vie de l'objet est marquée par un grand X, au point d'intersection avec la flèche du message Destruction.



Création d'un message Auto-destruction

Vous pouvez créer un message Auto-destruction à partir de la Boîte à outils.

1. Sélectionnez l'outil **Message** dans la Boîte à outils.
2. Cliquez sur la ligne de vie de l'émetteur et maintenez le bouton de la souris enfoncé. Faites glisser le curseur sur la ligne de vie du récepteur.
3. Relâchez le bouton de la souris sur la ligne de vie du récepteur, puis double-cliquez sur le symbole du nouveau message pour afficher sa feuille de propriétés.
4. Sélectionnez **Auto-Destruction** dans la liste **Action** de l'onglet **Détails**.
5. Sélectionnez Retour dans la liste Flux de contrôle.
6. Cliquez sur OK. La ligne de vie de l'objet auto-détruit est marquée par un grand X.



Création d'un message récursif dans un diagramme de séquence

Un message peut être récursif lorsque l'objet s'envoie un message à lui-même. Dans ce cas, la flèche commence et se termine sur la ligne de vie du même objet.

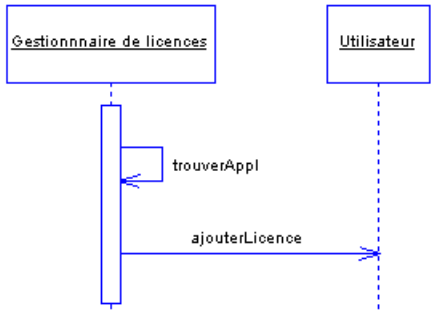
Vous n'êtes pas autorisé à utiliser l'action Création ou Auto-destruction, ni l'option Prise en charge du retard, avec un message récursif.

Lorsque vous créez des messages récursifs Non défini ou Retour à partir de la Boîte à outils, la valeur du flux de contrôle est déjà sélectionnée :

Type de message	Outil
Message récursif Non défini	
Message récursif Retour	

Vous pouvez également créer un message récursif Non défini, puis changer ensuite sa valeur de flux de contrôle.

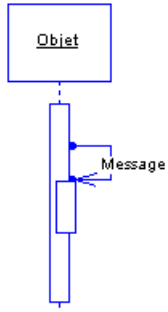
Exemple



Vous pouvez choisir de créer un message récursif avec ou sans activation en utilisant les outils de la Boîte à outils.

Lorsque vous créez un message récursif avec activation, ce dernier est automatiquement attaché à une activation et sa valeur de flux de contrôle est un Appel de procédure qui, par défaut, démarre l'activation.

Les symboles d'activation sont ensuite automatiquement créés sur la ligne de vie de l'objet comme illustré ci-après :



Création d'un message récursif sans activation

Vous pouvez créer un message récursif sans activation à partir de la Boîte à outils.

1. Cliquez sur l'outil **Message récursif** dans la Boîte à outils.
2. Cliquez sur la ligne de vie de l'objet pour créer un message récursif.

Création d'un message récursif avec activation

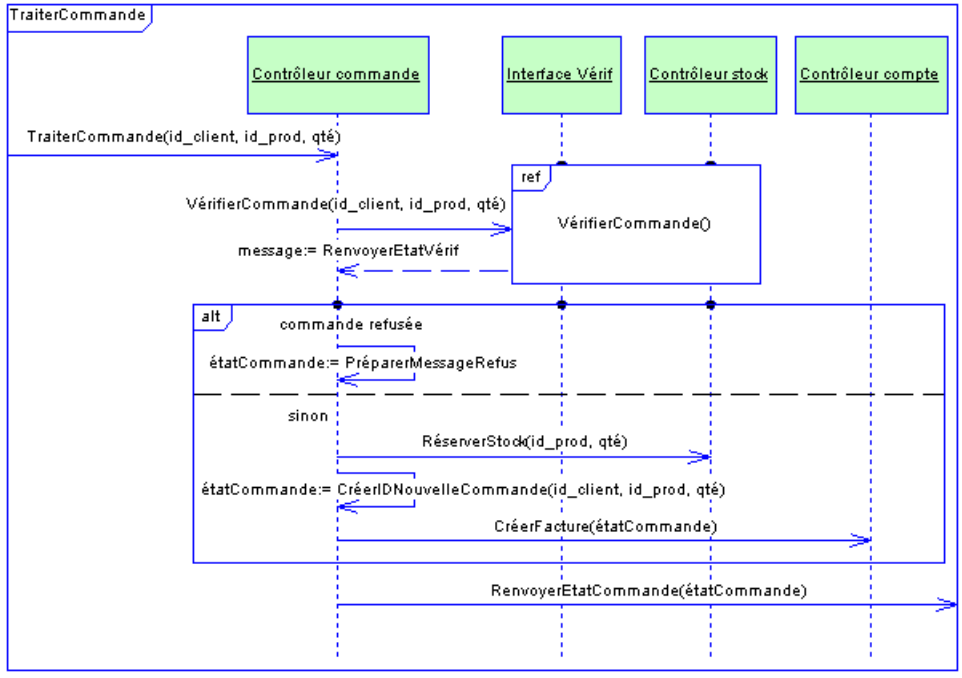
Vous pouvez créer un message récursif avec activation à partir de la Boîte à outils.

1. Sélectionnez l'outil **Message récursif** dans la Boîte à outils.
2. Cliquez sur la ligne de vie de l'objet pour créer un message récursif avec activation.

Messages et portes

Dans UML 2, vous pouvez envoyer des messages vers et depuis le cadre d'interaction qui entoure votre diagramme de séquence. Le cadre représente la limite externe du système (ou de la partie du système) en cours de modélisation et peut être utilisé à la place d'un acteur (les acteurs ne sont plus utilisés dans les diagrammes de séquence UML 2, mais continuent à être pris en charge pour des raisons de rétrocompatibilité dans PowerAMC). Un message provenant d'un endroit du cadre est envoyé depuis une *porte d'entrée*, tandis qu'un message parvenant à cet endroit est reçu par une *porte de sortie*.

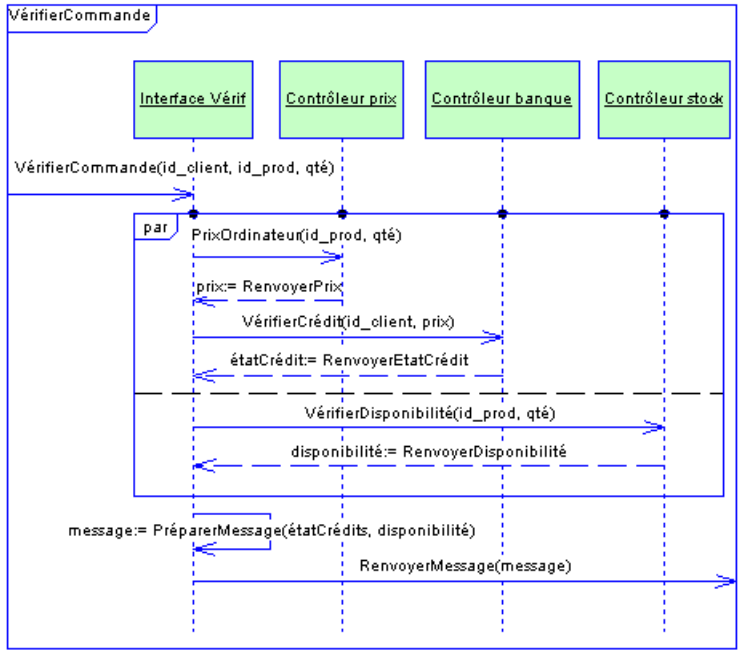
Dans l'exemple ci-dessous, un diagramme de séquence de haut niveau, TraiterCommande, montre une série de communications entre un utilisateur et un système de ventes :



Le message TraiterCommande part d'une porte d'entrée sur le cadre d'interaction TraiterCommande, et est reçu sous la forme d'un message d'entrée par l'objet Contrôleur commande. Une fois le traitement de la commande terminé, le message RenvoyerEtatCommande est reçu par une porte de sortie sur le cadre d'interaction TraiterCommande.

Le message VérifierCommande provient de l'objet Contrôleur commande et est reçu comme un message d'entrée par une porte d'entrée sur le cadre d'interaction VérifierCommande. Une fois la vérification de la commande terminée, le message RenvoyerEtatVérif est envoyé depuis une porte de sortie sur le cadre de référence d'interaction et est reçu par l'objet Contrôleur commande.

Le diagramme suivant montre le diagramme de séquence VérifierCommande qui illustre le processus de vérification d'une commande :



Ici, le message VérifierCommande provient d'une porte d'entrée sur le cadre d'interaction VérifierCommande, et est reçu comme message d'entrée par l'objet Interface Vérif. Une fois le traitement de la commande terminé, le message RenvoyerMessage est reçu par une porte de sortie sur le cadre d'interaction VérifierCommande.

Remarque : PowerAMC permet d'utiliser des acteurs et des cadres d'interaction dans vos diagrammes afin de mettre à votre disposition un choix de styles et d'assurer une compatibilité avec les versions antérieures. Toutefois, ces deux concepts représentant des objets extérieurs au système modélisé, nous vous déconseillons d'utiliser à la fois des acteurs et des cadres d'interaction au sein du même diagramme. Vous ne pouvez pas envoyer des messages entre un acteur et un cadre d'interaction.

Numéros d'ordre

Vous pouvez attribuer des numéros d'ordre dans les diagrammes de communication et diagrammes de séquence, mais c'est dans les diagrammes de communication qu'ils jouent un rôle plus important.

Lorsque vous créez un message dans un diagramme de communication, la valeur par défaut du numéro de séquence est calculée par rapport au message créé ou modifié le plus récemment. Le premier numéro d'ordre créé est 1.

La succession des numéros d'ordre est construite en partant du dernier numéro de séquence utilisé, incrémenté de 1. Par exemple, 3 + 1 => 4, ou 2.1 + 1 => 2.2

La création des numéros de séquence respecte la syntaxe des numéros déjà utilisés dans le diagramme (1, 2, 3, etc. ou 1.1, 1.2, etc.).

Par convention, l'ajout de lettres aux numéros d'ordre signifie que les messages sont parallèles. Par exemple, les messages ayant un numéro d'ordre 3.1a et 3.1b sont envoyés simultanément.

Si vous devez changer les numéros d'ordre manuellement, vous pouvez déplacer ou insérer des messages dans le diagramme ou bien augmenter ou diminuer les numéros d'ordre.

Déplacement de numéros d'ordre

Vous pouvez déplacer et insérer des numéros d'ordre dans un diagramme de communication.

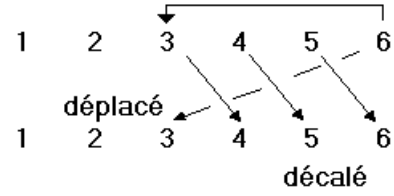
Lorsque vous déplacez un numéro d'ordre existant et l'attachez à un autre message, les numéros d'ordre sont recalculés en fonction des règles suivantes :

- Pour un numéro 'x', tous les numéros supérieurs ou égaux à 'x' sont modifiés.
- Tout intervalle libre dans la numérotation est occupé par le numéro de séquence suivant à l'issue du déplacement.

Exemple 1

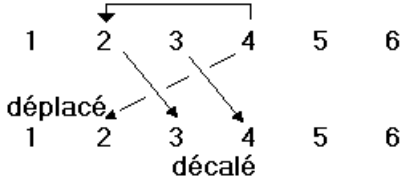
Les numéros d'ordre dans un diagramme de communication sont 1, 2, 3, 4, 5, et 6.

Lorsque vous faites passer le numéro d'ordre 6 à la troisième position, le numéro 6 devient le 3, tous les nombres entre 3 et 6 sont alors modifiés comme suit :

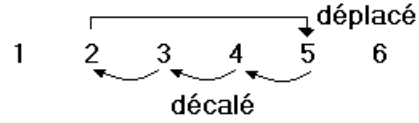


Exemple 2

Lorsque vous faites passer le numéro d'ordre 4 à la seconde position, il devient alors le numéro 4 et tous les numéros compris entre 2 et quatre inclus changent alors. Les numéros 5 et 6 restent inchangés :



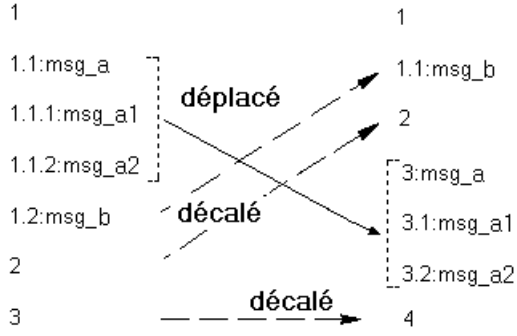
Exemple 3
Lorsque vous faites passer le numéro d'ordre 2 à la cinquième position, il devient le numéro 5 : tous les numéros de 2 à 5 sont modifiés comme suit :



Exemple 4
Les numéros d'ordre dans un diagramme de communication se présentent comme suit :

- 1
- 1.1:msg_a
- 1.1.1:msg_a1
- 1.1.2:msg_a2
- 1.2:msg_b
- 2
- 3

Si vous déplacez le numéro 1.1:msg_a en troisième position 3, les changements suivants se produisent :



Remarque : Vous pouvez utiliser la fonction Annuler chaque fois que nécessaire au cours de la manipulation de ces éléments dans des diagrammes de communication.

Insertion de numéros d'ordre

Lorsque vous insérez un nouveau message doté d'un nouveau numéro d'ordre dans un diagramme de communication existant, les numéros d'ordre sont recalculés en fonction de la règle suivante : tous les numéros situés après le point d'insertion sont incrémentés d'une unité (+1).

De même, un numéro parent change ses numéros enfant. Par exemple, si on incrémente d'une unité des numéros tels que 1.1, 1.2 ou 1.3 on obtient le résultat du type : 1.1 + 1 => 1.2.

La syntaxe des numéros de séquence utilisée dans le diagramme est respectée ; ainsi les numéros de séquence sont incrémentés d'une unité quelle que soit leur syntaxe (1, 2, 3... ou 1.1, 1.2, 1.3...).

Incrémentation de numéros d'ordre dans un diagramme de communication

Vous pouvez incrémenter un numéro d'ordre de l'une des façons suivantes :

- Pointez sur le message dans le diagramme, cliquez le bouton droit de la souris, puis sélectionnez Incrémenter le numéro dans le menu contextuel.

  ou

  Sélectionnez le message dans le diagramme, puis maintenez la touche **Ctrl** enfoncée et appuyez sur le signe **+** sur le pavé numérique afin d'incrément le numéro de 1.

Décrémentation de numéros d'ordre dans un diagramme de communication

Vous pouvez décrémenter un numéro d'ordre de l'une des façons suivantes :

- Pointez sur le message dans le diagramme, cliquez le bouton droit de la souris, puis sélectionnez Décrémenter le numéro dans le menu contextuel.

  ou

  Sélectionner le numéro d'ordre dans le digramme, puis appuyez sur **Ctrl** et sur le signe **+** pour décrémenter le numéro d'une unité.

Activation (MOO)

Les activations sont des symboles facultatifs qui représentent le temps requis pour qu'une action soit effectuée. Ils sont créés sur la ligne de vie d'un objet. Les activations sont uniquement des symboles et elles sont dépourvues de feuille de propriétés.

Une activation peut être créée dans les types de diagramme suivants :

- Diagramme de séquence

Dans un diagramme de communication, les messages qui sont transmis lors de la période représentée par une activation se voient attribuer des sous-numéros. Ainsi, une activation créée par le message 1 peut donner lieu à la création de messages 1.1 et 1.2.

Vous pouvez attacher un message à une activation ou l'en détacher. Vous pouvez également déplacer et redimensionner une activation ou lui faire chevaucher d'autres activations.

Création d'une activation

Vous pouvez créer une activation lorsque vous créez un message Appel de procédure ou après avoir créé tout type de message. Un message Appel de procédure démarre une activation, c'est pourquoi le message est toujours attaché à une activation.

Création d'activations avec des messages Appel de procédure

Les activations sont automatiquement créées lorsque vous créez des messages Appel de procédure dans le diagramme.

1. Sélectionnez l'outil **Appel de procédure** dans la Boîte à outils.
2. Cliquez sur la ligne de vie de l'émetteur et maintenez le bouton de la souris enfoncé. Faites glisser le curseur sur la ligne de vie du récepteur.
3. Relâchez le bouton de la souris sur la ligne de vie du récepteur.

   Le message est créé, avec les activations sur les lignes de vie de l'émetteur et du récepteur (notez que les activations ne sont pas créées sur les lignes de vie des acteurs).

Création d'une activation à partir d'un diagramme

Vous pouvez créer une activation à partir d'un diagramme en utilisant l'outil Activation.

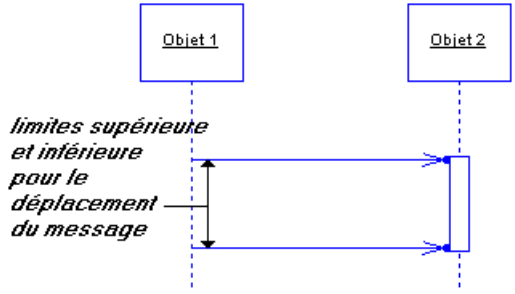
1. Sélectionnez l'outil **Activation** dans la Boîte à outils.
2. Cliquez sur la ligne de vie d'un objet pour créer un symbole d'activation là où vous avez cliqué. Si vous créez l'activation au-dessus de messages existants, ceux-ci sont alors automatiquement attachés à cette activation.

Attachement d'un message à une activation

Un message est attaché à une activation lorsque son point de début ou de fin se trouve sur le symbole d'activation et non pas sur la ligne de vie de l'objet. Les symboles d'attachement sont situés aux extrémités de la flèche du message. Si les symboles d'attachement ne sont pas affichés dans le diagramme, sélectionnez **Outils > Préférences d'affichage**, puis sélectionnez Point d'attache des activations dans la catégorie Message.

Vous pouvez attacher un message existant à une activation en faisant glisser le message dans l'activation en maintenant la touche **Ctrl** enfoncée.

Lorsqu'un message est attaché à une activation, vous ne pouvez pas le déplacer hors des limites du symbole d'activation comme illustré ci-dessous :



Si vous supprimez une activation à laquelle un message est attaché, le message est détaché de l'activation mais n'est pas supprimé.

Flux de contrôle de message et activation
Lorsqu'un message est attaché à une activation, la valeur du flux de contrôle influe sur la position de l'activation vis-à-vis du message :

- Appel de procédure - Un message Appel de procédure attaché à une activation démarre l'activation sur la ligne de vie du récepteur, c'est-à-dire que le point d'arrivée du message est situé au sommet de l'activation.
- Retour - Un message Retour attaché à une activation termine l'activation sur la ligne de vie de l'émetteur, c'est-à-dire que le point de départ du message est situé à la base de l'activation.

Les messages Appel de procédure et Retour sont les seuls messages définis à un endroit particulier dans l'activation : un message Appel de procédure se trouve au sommet de l'activation, un message Retour se trouve à la base de l'activation. Les autres messages attachés à l'activation peuvent être déplacés sans contrainte particulière au sein de l'activation.

Détachement d'un message d'une activation

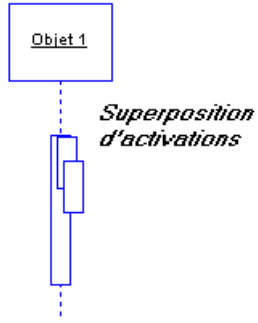
Vous pouvez détacher un message d'une activation en faisant glisser le message hors de l'activation, tout en maintenant la touche **Ctrl** enfoncée.

Activations superposées

Une activation peut se superposer à d'autres activations existantes.

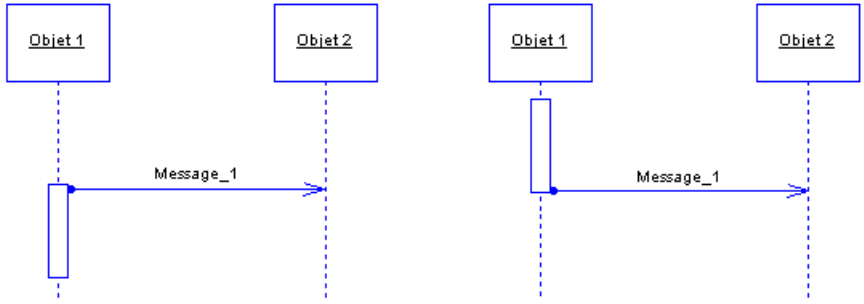
Par exemple, vous pouvez souhaiter qu'une activation se superpose à une autre pour représenter une action dans une boucle. L'action est répétée jusqu'à ce qu'elle atteigne son but, cette boucle peut commencer et se terminer en même temps que l'autre activation représente une autre action.

Vous pouvez superposer ainsi les activations pour illustrer des activités simultanées.

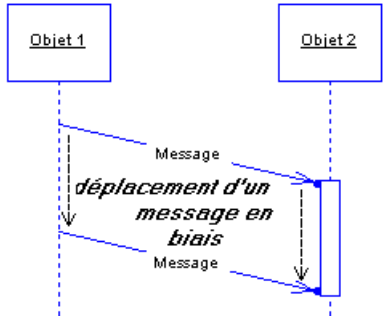


Déplacement d'une activation

Si vous déplacez une activation à laquelle un message est attaché, il est possible de la déplacer vers le haut et vers le bas jusqu'à ce que son sommet ou sa base atteigne le niveau du message. Le message lui-même ne se déplace pas :



Si vous déplacez le point de fin d'un message pour lequel l'option Prise en charge du retard est activée, l'angle d'inclinaison du message est préservé comme illustré ci-dessous :



Redimensionnement d'une activation

Vous pouvez redimensionner une activation en étirant le sommet ou la base de son symbole. Vous redimensionnez alors l'activation verticalement.

Lorsque vous redimensionnez une activation, les règles suivantes s'appliquent :

- Un message Appel de procédure est toujours attaché au sommet de l'activation sur la ligne de vie du récepteur. Le message Appel de procédure reste au sommet de l'activation si cette dernière est déplacée vers le haut, ou redimensionnée vers le bas.
- Un message Retour est toujours attaché à la base de l'activation sur la ligne de vie de l'émetteur. Le message Retour reste à la base de l'activation si cette dernière est déplacée vers le bas.
- Les messages qui sont couverts par l'activation à l'issue du redimensionnement ne sont pas automatiquement attachés à l'activation.

Pour changer l'activation d'un message, maintenez la touche **Ctrl** enfoncée et cliquez pour sélectionner l'une des extrémités du message et la faire glisser sur une autre activation.

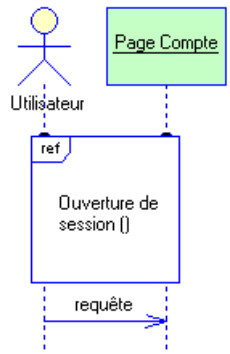
Références d'interaction et activités d'interaction (MOO)

Une référence d'interaction est utilisée pour représenter un diagramme de séquence dans le corps d'un autre. Cette fonctionnalité vous permet de modulariser et de réutiliser des interactions couramment utilisées sur une plage de diagrammes de séquence. Les activités d'interaction sont similaires, mais permettent de représenter un diagramme de séquence, un diagramme de communication ou un diagramme d'interactions.

Les références d'interaction et activités d'interaction peuvent être créées dans des diagrammes de séquence.

Pour une activité d'interaction, pointez sur une activité, cliquez le bouton droit de la souris, puis sélectionnez **Vue composite > Lecture seule (sous-diagramme)** pour afficher le diagramme référencé dans le symbole. Sélectionnez **Ajuster à la vue en lecture seule** dans le menu contextuel afin de redimensionner automatiquement le symbole afin d'optimiser l'affichage du diagramme référencé.

Exemple : référence d'interaction dans un diagramme d'activité
Dans l'exemple ci-dessous, l'utilisateur doit ouvrir une session avant de transmettre une requête sur la page de compte d'un site web. Comme le processus de connexion fera partie des nombreuses interactions avec le site, il a été représenté dans un autre diagramme de séquence appelé "Ouverture de session", et est représenté ici sous forme d'une référence d'interaction.



Création d'une référence d'interaction

Vous pouvez créer une référence d'interaction à partir de la Boîte à outils ou de l'Explorateur d'objets.

- Utilisez l'outil **Référence** dans la Boîte à outils. Vous pouvez soit cliquer à proximité de la ligne de vie d'un objet pour créer une référence d'interaction attachée à cette la ligne de vie, soit tracer un rectangle qui va englober et attacher plusieurs lignes de vie.

La boîte de dialogue **Sélection d'un diagramme de séquence** s'affiche pour vous permettre de spécifier le diagramme de séquence auquel la référence se réfère. Sélectionnez un diagramme existant ou un nouveau diagramme, cliquez sur **OK**.

- Faites glisser un autre diagramme de séquence depuis l'Explorateur d'objets dans le diagramme de séquence courant.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Création d'une activité d'interaction

Vous pouvez créer une activité d'interaction à partir de la Boîte à outils ou de l'Explorateur d'objets.

- Utilisez l'outil **Activité d'interaction** dans la Boîte à outils.
  La boîte de dialogue **Sélection d'un diagramme** s'affiche pour vous permettre de spécifier le diagramme auquel l'activité fait référence. Sélectionnez un diagramme existant ou un nouveau diagramme, puis cliquez sur **OK**.
- Faites glisser un diagramme de séquence, un diagramme de communication ou un diagramme d'interactions depuis l'Explorateur d'objets dans le diagramme d'interactions.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'une référence d'interaction ou d'une activité d'interaction

Vous pouvez modifier les propriétés d'un objet à partir de sa feuille de propriétés. Pour afficher la feuille de propriétés d'une référence ou activité d'interaction, double-cliquez sur son symbole dans le diagramme, en prenant soin de pointer sur l'étiquette d'opérateur, dans l'angle supérieur gauche.

L'onglet Général contient les propriétés suivantes :

Propriété	Description
Diagramme référencé	Spécifie le diagramme de séquence qui sera représenté dans le diagramme courant par la référence d'interaction. Vous pouvez cliquer sur l'outil Créer à droite de la zone pour créer un nouveau diagramme de séquence.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML.
Arguments	Spécifie les arguments qui doivent être transmis au premier message dans le diagramme référencé.
Valeur de retour	Spécifie la valeur à renvoyer par le dernier message dans le diagramme référencé.

Manipulation des références d'interaction

Vous pouvez déplacer les références d'interaction, les redimensionner et généralement les manipuler librement. Lorsque vous faites en sorte que le symbole se superpose à la ligne de vie d'un objet, il s'y attache immédiatement, et cet attachement est représenté par une petite

pastille sur la partie supérieure du symbole, là où il rencontre la ligne de vie. Si vous faites glisser ou redimensionnez le symbole de telle sorte qu'il ne se superpose plus à la ligne de vie, il s'en détache automatiquement.

Si vous déplacez un objet attaché à une interaction, le symbole se redimensionne automatiquement pour rester attaché à la ligne de vie de l'objet.

Vous pouvez contrôler manuellement si les lignes de vies sont attachées à une référence d'interaction qui les chevauche en cliquant sur le point d'attachement.

Notez qu'une référence d'interaction ne peut pas être copiée ou réutilisée dans un autre diagramme. Toutefois, plusieurs références au même diagramme peuvent être créées.

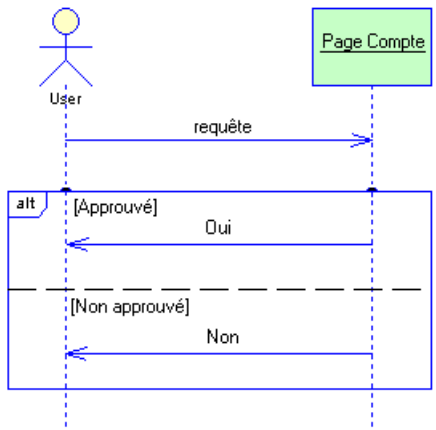
Fragments d'interaction (MOO)

Un fragment d'interaction permet de regrouper des messages liés au sein d'un diagramme de séquence. Il existe différents types de fragment prédéfinis qui permettent de spécifier des trajets alternatifs, des messages parallèles ou des boucles.

Un fragment d'interaction peut être créé dans les types de diagramme suivants :

- Diagramme de séquence

Dans l'exemple ci-dessous, Utilisateur envoie une requête à la page Compte. Deux réponses possibles et les conditions dont elles dépendent sont contenues dans un fragment d'interaction.



Création d'un fragment d'interaction

Vous pouvez créer un fragment d'interaction à partir de la Boîte à outils.

- Utilisez l'outil **Fragment d'interaction** dans la Boîte à outils. Vous pouvez soit cliquer à proximité de la ligne de vie d'un objet pour créer un fragment d'interaction attaché à cette ligne de vie, soit tracer un rectangle qui englobe et attache plusieurs lignes de vie.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'un fragment d'interaction

Vous pouvez modifier les propriétés d'un objet à partir de sa feuille de propriétés. Pour afficher une feuille de propriétés de fragment d'interaction, double-cliquez sur son symbole dans le diagramme, dans l'étiquette d'opérateur située dans la partie supérieure gauche de ce symbole. Les sections suivantes détaillent les onglets de la feuille de propriétés qui contiennent les propriétés les plus utilisées pour les fragments d'interaction.

L'onglet Général contient les propriétés suivantes :

Propriété	Description
Opérateur	Spécifie le type de fragment. Vous pouvez choisir parmi les valeurs suivantes : • Alternative (alt) – le fragment est scindé en plusieurs régions mutuellement exclusives, chacune dotée d'une condition de garde. Seuls les messages provenant de l'une de ces régions seront exécutés au moment de l'exécution. • Assertion (assert) – l'interaction doit se produire exactement tel qu'indiqué sous peine d'être invalide. • Break (break) – si la condition associée est remplie, l'interaction parent se termine à la fin du fragment. • Consider (consider) – seuls les messages affichés sont significatifs. • Critical Region (critical) – aucun autre message ne peut intervenir avant que ces messages ne soient terminés. • Ignore (ignore) – certains messages non significatifs ne sont pas affichés. • Loop (loop) – le fragment d'interaction sera répété à plusieurs reprises. • Negative (neg) – l'interaction est invalide et ne peut pas se produire. • Option (opt) – l'interaction se produit uniquement si la condition de garde est satisfaite. • Parallel (par) – le fragment est scindé en plusieurs régions, toutes seront exécutées en parallèle lors de l'exécution. • Strict Sequencing (strict) – l'ordre des messages est forcé. • Weak Sequencing (seq) – l'ordre des messages est forcé sur chaque ligne de vie, mais pas entre les lignes de vie. Le type d'opérateur est affiché dans l'angle supérieur gauche du symbole de fragment d'interaction.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML.
Condition	Spécifie toute condition associée au fragment. Il peut s'agir de l'évaluation d'une variable, telle que : X > 3 Ou, dans le cas d'un fragment de boucle, la spécification du nombre minimum et (éventuellement) du nombre maximum de fois que la boucle est parcourue. Exemple : 1,10 Dans le cas des opérateurs Consider ou Ignore, ce champ répertorie les messages associés. Ce champ n'est pas disponible si le fragment ne prend pas en charge les conditions.

Onglet Sous-régions
L'onglet Sous-régions répertorie les régions contenues dans le fragment. Il n'est affiché que si vous sélectionnez un opérateur qui requiert plus d'une région. Vous pouvez ajouter ou supprimer des régions et (le cas échéant) spécifier les conditions correspondantes.

Manipulation de fragments d'interaction

Vous pouvez manipuler des fragments d'interaction, avec toutefois certaines limitations.

Déplacement et redimensionnement de fragments
Vous pouvez déplacer des fragments d'interaction, les redimensionner, et généralement les manipuler librement. Lorsque vous faites en sorte que le symbole se superpose à la ligne de vie d'un objet, il s'y attache immédiatement, et cet attachement est représenté par une petite pastille sur la partie supérieure du symbole, là où il rencontre la ligne de vie. Si vous faites glisser ou redimensionnez le symbole de telle sorte qu'il ne se superpose plus à la ligne de vie, il s'en détache automatiquement.

Si vous déplacez un objet attaché à un fragment d'interaction, le symbole se redimensionne automatiquement pour rester attaché à la ligne de vie de l'objet.

Vous pouvez contrôler manuellement si les lignes de vies sont attachées à une référence d'interaction qui les chevauche en cliquant sur le point d'attachement.

Déplacement de messages
Chaque message qui est entièrement contenu dans un fragment d'interaction se déplace avec le fragment vers le haut et le bas sur la ligne de vie de l'objet auquel il est attaché. Toutefois, si vous éloignez le fragment de la ligne de vie auquel il est attaché, le message sera détaché du fragment et ne sera pas déplacé.

Vous pouvez déplacer les messages librement vers l'extérieur ou l'intérieur des fragments d'interaction. Si vous déplacez un message de sorte qu'il soit inclus entièrement dans un fragment, il est attaché à ce fragment. Si vous déplacez un message de sorte qu'une ou l'autre de ses extrémités se retrouve hors du fragment, ce message est détaché du fragment.

Messages et régions
Lorsqu'un fragment est scindé en plusieurs régions, vous pouvez déplacer les messages librement entre les régions. Toutefois, vous ne pouvez pas déplacer la ligne de séparation située au-dessous. Un tel redimensionnement va affecter la taille totale du fragment. Pour redimensionner la dernière région, située en bas du fragment, vous devez sélectionner et déplacer le bord inférieur du fragment.

Si vous supprimez une région, l'espace qu'elle occupait et les messages qu'elle contenait seront fusionnés avec ceux de la région située au-dessus.

Notez qu'un fragment d'interaction ne peut pas être copié comme raccourci dans un autre diagramme.

Activités (MOO)

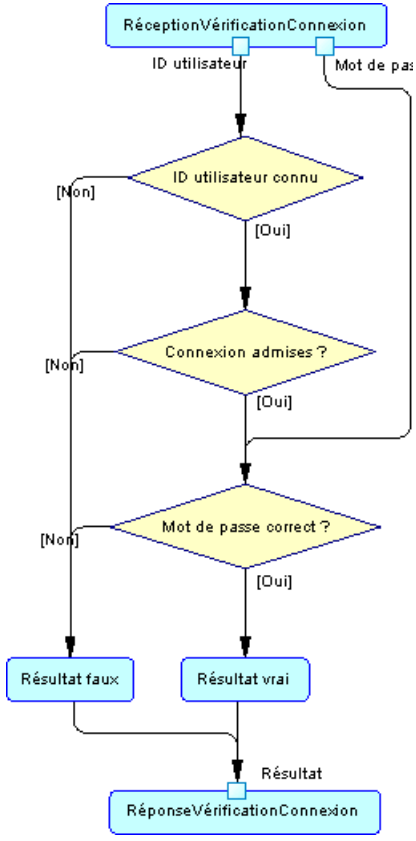
Une *activité* est l'appel d'une action manuelle ou automatisée, telle que "envoyer un courrier", ou "incrémenter un compteur". Lorsqu'une activité prend le contrôle, elle effectue l'action puis, selon le résultat de cette action, la transition (flux de contrôle) est transmise à une autre activité.

Une activité peut être créée dans les types de diagramme suivants :

- Diagramme d'activités

La prise en charge de UML 2 par PowerAMC permet de disposer d'une grande souplesse et d'un grand niveau de détail dans vos diagrammes d'activité. Vous pouvez simplement lier les activités entre elles afin de montrer un flux de contrôle de haut niveau, ou affiner votre modèle en spécifiant :

- les paramètres passés entre les activités (voir *Spécification des paramètres d'activité* à la page 173)
- le type d'action et l'activité et l'associer avec d'autres objets de modèle (voir *Spécification des types d'action* à la page 174)



Dans l'exemple ci-dessus, l'activité RéceptionVérificationConnexion a comme type d'action "Accepter appel" (voir *Spécification des types d'action* à la page 174), et passe les deux paramètres de sortie "ID utilisateur" et "mot de passe" (voir *Spécification des paramètres d'activité* à la page 173) à une série de décisions qui conduisent à RéponseVérificationConnexion. Cette dernière activité a un paramètre d'entrée appelé "Résultat" et le type d'action "Répondre appel".

Activités atomiques et décomposées

Une activité peut être atomique ou décomposée. Les activités décomposées contiennent des sous-activités, qui sont représentées dans un sous-diagramme. Pour plus d'informations, voir *Activités décomposées et sous-activités* à la page 180.

Une activité PowerAMC équivaut à une activité UML (ActionState ou SubactivityState) et à un graphe d'activité. Dans UML, un ActionState représente l'exécution d'une action atomique, et un SubactivityState est l'exécution d'un graphe d'activité (qui est, à son tour, la description d'une action complexe représentée par des sous-activités).

Le tableau suivant montre les correspondances entre les concepts et la terminologie UML et PowerAMC :

Objet UML	Objets PowerAMC
ActionState	Activité
SubactivityState	Activité composite
Activity Graph	Activité composite

PowerAMC combine une SubactivityState et un graphe d'activités (activity graph) dans une activité décomposée de sorte que vous pouvez définir des sous-activités directement sous l'activité parent et ce, sans avoir à définir d'objet supplémentaire. Si vous devez mettre en exergue la différence, vous pouvez créer des activités directement sous le modèle ou le package, et utiliser des raccourcis d'activité pour détailler la mise en oeuvre des activités, ainsi SubactivityState correspond au raccourci d'une activité décomposée.

Création d'une activité

Vous pouvez créer un activité à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Activité** dans la Boîte à outils.
- Sélectionnez **Modèle > Activités** pour afficher la boîte de dialogue Liste des activités, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle, un package ou sur une activité décomposée dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Activité**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'une activité

Pour visualiser ou modifier les propriétés d'une activité, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Unité d'organisation	Spécifie l'unité d'organisation (voir *Unités d'organisation (MOO)* à la page 183) liée à l'activité et permet également d'affecter la valeur Activité communautaire (voir *Affichage d'une activité communautaire* à la page 186) à une activité décomposée afin de représenter de façon graphique les liens qui existent entre les unités d'organisation représentées sous forme de couloirs et les sous-activités. Une activité communautaire est une activité réalisée par plusieurs unités d'organisation. Vous pouvez cliquer sur le bouton Points de suspension en regard de la zone Unité d'organisation pour créer une nouvelle unité d'organisation, ou bien cliquer sur l'outil Propriétés afin d'afficher la feuille de propriétés de l'unité d'organisation.
Etat composite	Spécifie si l'activité est décomposée en sous-activités. Vous pouvez choisir parmi les valeurs suivantes : • Activité atomique – (valeur par défaut) l'activité ne contient aucune sous-activité. • Activité décomposée – l'activité peut contenir des sous-activités. Un onglet Sous-activités est affiché dans la feuille de propriétés afin de répertorier ces sous-activités, et un sous-diagramme est créé sous l'activité dans l'Explorateur d'objets pour les afficher. Si vous repassez de Activité décomposée à Activité atomique, les sous-activités que vous avez créées seront supprimées.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Onglet Action

L'onglet**Action** définit la nature, le type et la durée d'une action qu'une activité accomplit. Il contient les propriétés suivantes :

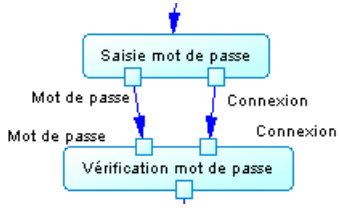
Propriété	Description
Type d'action	Spécifie la catégorie d'action que l'activité exécute. Pour plus d'informations, voir *Spécification des types d'action* à la page 174.
[objet de l'action]	Selon le type d'action que vous sélectionnez, une zone supplémentaire peut être affichée, permettant de spécifier une activité, un classificateur, un attribut, un événement, une expression, une opération ou une variable sur lequel/laquelle l'action est accomplie. Vous pouvez utiliser les outils à droite de la liste pour créer un objet, parcourir la liste des objets disponibles ou afficher la feuille de propriétés de l'objet sélectionné
Pré-conditions/ Actions /Post-conditions	Ces sous-onglets fournissent des informations contextuelles sur les modalités d'exécution de l'action. Par exemple, vous pouvez rédiger du pseudo code ou de l'information sur le programme à exécuter
Durée	Spécifie la durée estimée ou statistique nécessaire à l'exécution de l'action. Cette information est utilisée à des fins de documentation uniquement ; l'estimation de la durée globale n'est pas calculée.
Dépassement de délai	Zéro par défaut. Si la valeur n'est pas fixée à zéro, une erreur de dépassement de délai se produit si l'exécution de l'activation prend plus de temps que la limite spécifiée. Vous pouvez saisir la valeur alphanumérique de votre choix dans la zone Dépassement de délai (par exemple : 20 secondes).

Onglets Paramètre d'entrée et Paramètres de sortie
Ces onglets contiennent les paramètres d'entrée et de sortie requis par l'activité (voir *Spécification des paramètres d'activité* à la page 173).

Onglet Sous-activités
Cet onglet n'est affiché que si la propriété Etat composite de l'activité a pour valeur Décomposé, et répertorie les sous-activités de l'activité

Spécification des paramètres d'activité

Les paramètres d'activité sont des valeurs passées entre les activités. Elles sont représentées sous la forme de petits carrés sur le bord des symboles d'activité, les paramètres ID utilisateur et mot de passe sont passés de l'activité Saisie mot de passe à l'activité Vérification mot de passe.



1. Affichez la feuille de propriétés d'une activité et cliquez sur l'onglet Paramètres d'entrée ou Paramètres de sortie.
2. Utilisez les outils pour ajouter les paramètres existants ou créer un nouveau paramètre.

Remarque : Vous pouvez également créer des paramètres dans le cadre de la spécification d'un type d'action d'activité. Voir *Spécification des types d'action* à la page 174.

Les paramètres d'activité peuvent avoir les propriétés suivantes :

Propriété	Description
Parent	Spécifie l'activité parent.
Nom/Code/ Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Type de données	Spécifie le type de données du paramètre. Vous pouvez choisir un type de données standard ou spécifier un classificateur. Vous pouvez utiliser les outils situés à droite de la liste pour créer un classificateur, passer en revue les classificateurs disponibles ou afficher les propriétés du classificateur sélectionné.
Etat	Spécifie l'état d'objet lié au paramètre. Vous pouvez saisir du texte libre dans la zone, ou utiliser les outils à droite de la liste afin de créer un état, passer en revue les états disponibles ou afficher les propriétés de l'état sélectionné.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Spécification des types d'action

Vous pouvez ajouter des détails supplémentaires à vos modélisations d'activités en spécifiant le type d'action effectué et, dans certains cas, en l'associant à un modèle objet particulier sur lequel elle agit, ainsi que les paramètres qu'elle passe.

1. Affichez la feuille de propriétés d'une activité, puis cliquez sur l'onglet **Action**.
2. Sélectionnez un type d'action. La liste suivante détaille les types d'action disponibles, et spécifie, le cas échéant, l'objet requis pour l'action :

- *<Undefined>* [aucun objet] - valeur par défaut. Aucune action définie.
- *Reusable Activity* [aucun objet] – conteneur racine.
- *Call* [opération ou activité] – appelle une opération ou activité. Voir *Exemple: Utilisation du type d'action Call* à la page 175
- *Accept Call* [opération ou activité] – attend qu'une opération ou activité soit appelée.
- *Reply Call* [opération ou activité] – suit une action Accept Call, et répond à une opération ou activité.
- *Generate Event* [événement] – génère un événement. Peut être utilisé pour soulever une exception.
- *Accept Event* [événement] – attend qu'un événement se produise.
- *Create Object* [classificateur] – crée une nouvelle instance d'un classificateur.
- *Destroy Object* [classificateur] – supprime une instance d'un classificateur.
- *Read Attribute* [attribut de classificateur] – lit une valeur d'attribut dans une instance d'un classificateur.
- *Write Attribute* [attribut de classificateur] – écrit une valeur d'attribut dans une instance d'un classificateur
- *Read Variable* [variable] – écrit une valeur dans une variable locale. La variable peut être utilisée pour stocker un paramètre de sortie fourni par une action à réutiliser ultérieurement dans le diagramme. Voir *Propriétés d'une variable* à la page 179.
- *Write Variable* [variable] - lit une valeur depuis une variable locale. Voir *Propriétés d'une variable* à la page 179.
- *Evaluate Expression* [texte d'expression] – évalue une expression et envoie la valeur sous la forme d'un paramètre de sortie.
- *Unmarshall* [aucun objet] – scinde les instances d'un objet entrant en plusieurs sorties calculées à partir de ce dernier.
- *Region* [aucun objet] – activité composite qui isole une partie du graphe. Equivaut à l'objet UML Interruptible Activity Region.
- *For Each* [aucun objet] – boucle une collection d'entrée pour exécuter un jeu d'actions spécifiées dans l'activité décomposée. Equivaut à l'objet UML Expansion Region.
- *Loop Node* [texte d'expression] – texte d'expression.

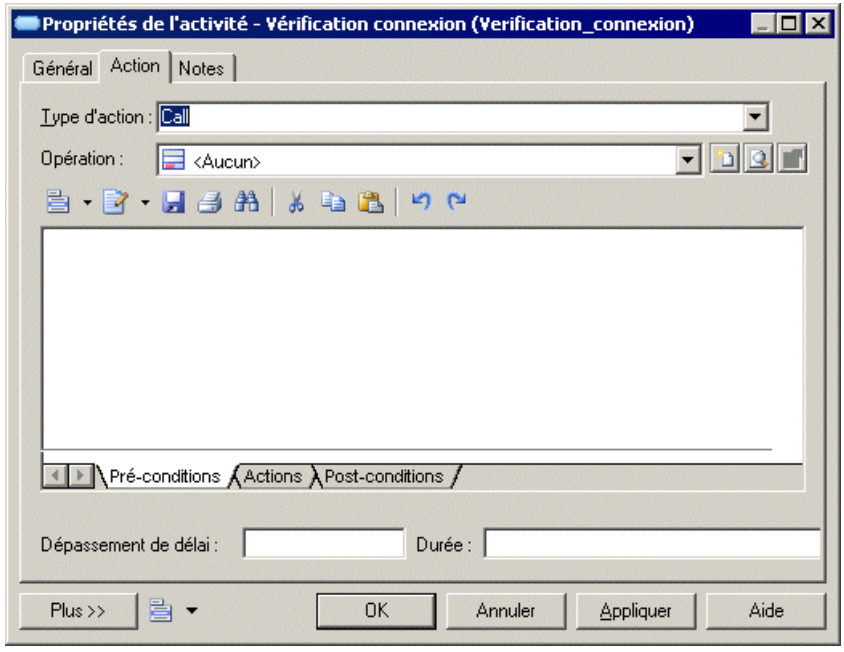
3. Si le type d'action requiert un objet, une zone supplémentaire s'affiche directement sous la liste Type d'action, vous permettant de spécifier une activité, un classificateur, un attribut, un événement, une expression, une opération ou une variable sur lequel/laquelle l'action agit. Vous pouvez utiliser les outils à droite de la liste pour créer un objet, parcourir l'arborescence des objets disponibles ou afficher les propriétés de l'objet sélectionné.
4. Cliquez sur **OK** pour enregistrer vos modifications et revenir au diagramme..

Exemple: Utilisation du type d'action Call

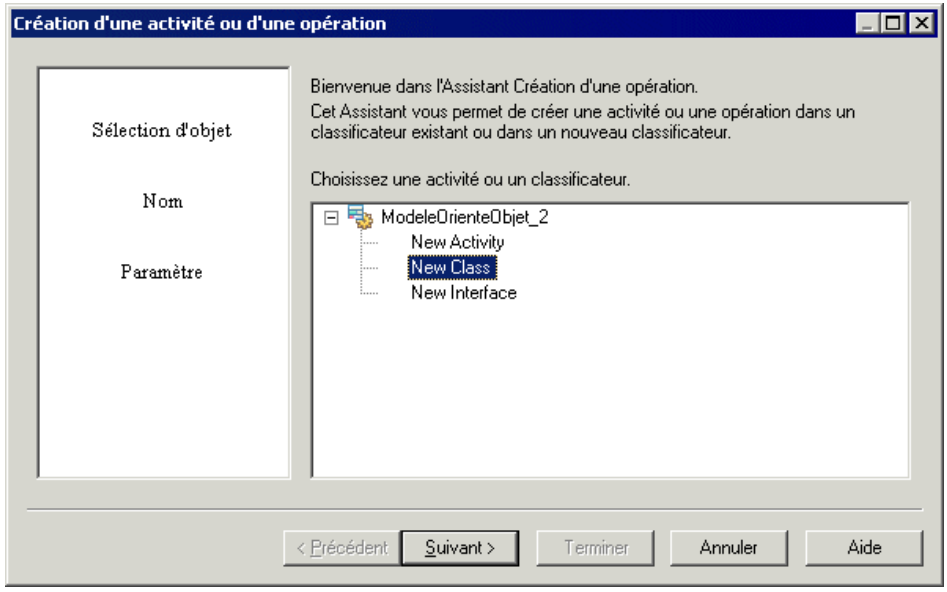
Un des types d'action les plus couramment utilisés est Call, qui permet à une activité d'appeler une opération de classificateur (ou une autre activité).

1. Créez une activité et appelez-là Vérification connexion.

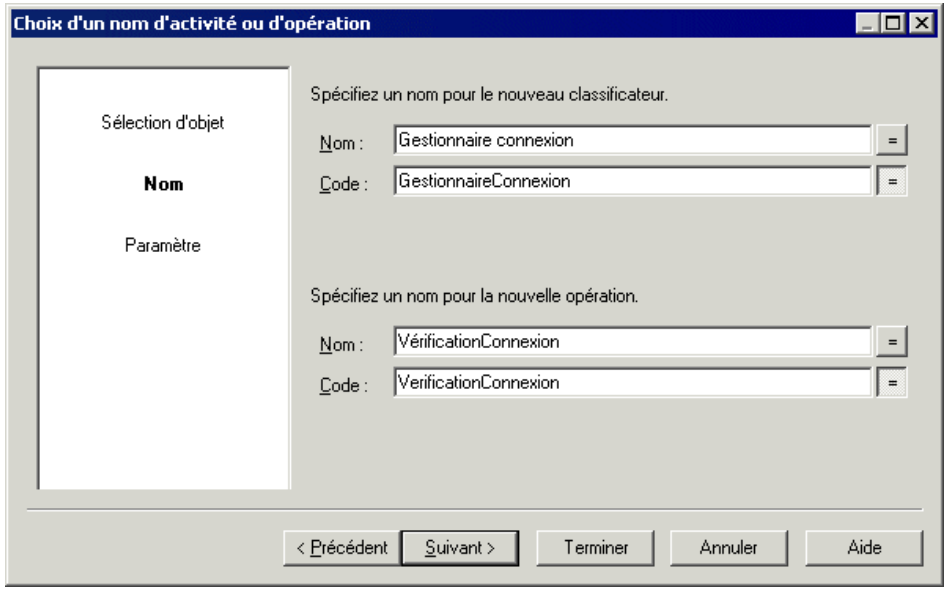
2. Affichez sa feuille de propriétés, cliquez sur l'onglet Action, puis sélectionnez Call dans la liste Type d'action. La zone Opération s'affiche :



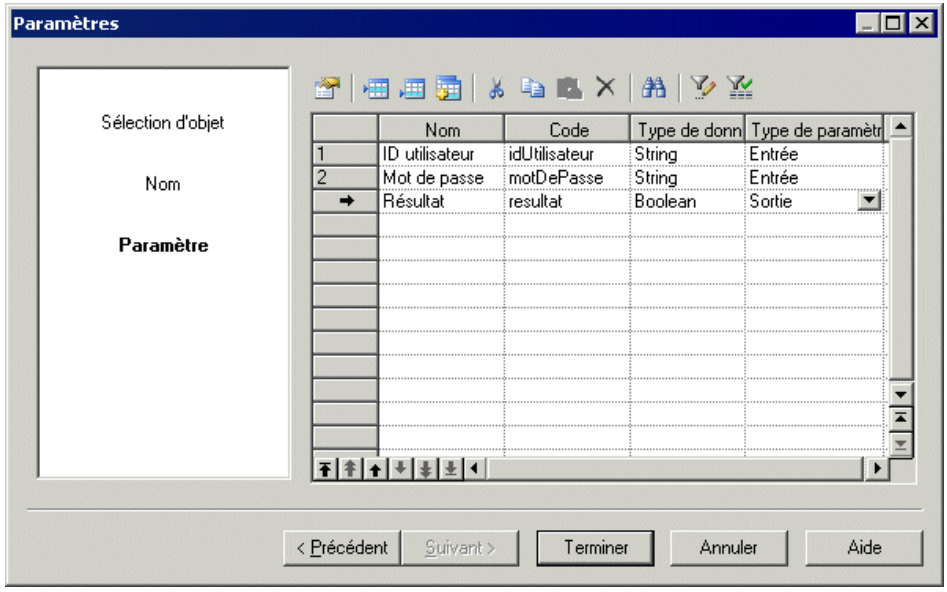
3. Cliquez sur l'outil Créer à droite de la nouvelle zone pour afficher un Assistant permettant de choisir une opération :



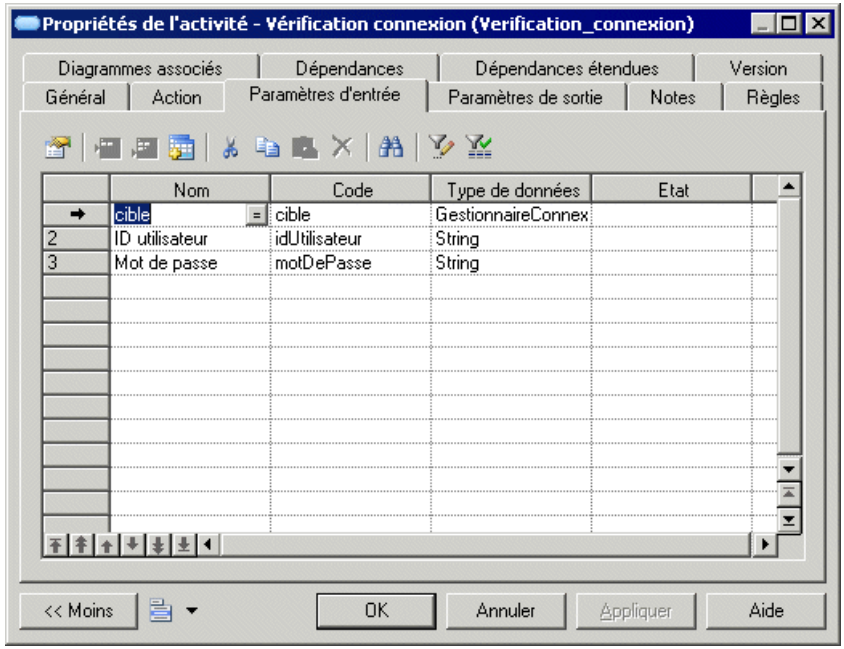
4. Vous pouvez choisir un classificateur ou une activité existant, ou bien choisir d'en créer. Sélectionnez New Class, puis cliquez sur Suivant :



5. Spécifiez un nom pour la classe et pour l'opération que vous souhaitez créer, puis cliquez sur Suivant :



6. Créez deux paramètres d'entrée et un paramètre de sortie pour la nouvelle opération, puis cliquez sur Terminer. La feuille de propriétés de la nouvelle opération s'affiche pour permettre d'affiner sa définition. Une fois les informations appropriées spécifiez, cliquez sur OK pour revenir à la feuille de propriétés de l'activité, puis cliquez sur l'onglet Paramètres d'entrée pour visualiser les paramètres que vous avez créés :



Notez que, en plus de vos deux paramètres d'entrée, PowerAMC en a créé un troisième, appelé "cible", avec le type de la nouvelle classe.

7. Cliquez sur OK pour enregistrer les modifications et revenir au diagramme :

L'activité affiche maintenant deux paramètres d'entrée et un paramètre de sortie (le paramètre cible est caché par défaut). La classe et l'opération que vous avez créées sont disponibles dans l'Explorateur d'objets pour utilisation ultérieure.

Exemple : Lecture et écriture de variables

Les variables contiennent des valeurs temporaires qui peuvent être passées d'une activité à l'autre. Vous pouvez créer des variables et y accéder en utilisant les types d'action Write Variable et Read Variables.

1. Affichez la feuille de propriétés d'une activité, puis cliquez sur l'onglet **Action**.
2. Sélectionnez le type d'action approprié :
   - Read Variable - puis cliquez sur l'outil **Créer** ou **Sélectionner un objet** à droite de a zone de variable pour créer ou sélectionner la variable à lire.
   - Write Variable - puis cliquez sur l'outil **Créer** ou **Sélectionner un objet** à droite de a zone de variable pour créer ou sélectionner la variable dans laquelle écrire.
3. Spécifiez le nom et les autres propriétés, puis cliquez sur **OK** pour revenir à la feuille de propriétés d'activité.

Propriétés d'une variable

Pour visualiser ou modifier les propriétés d'une variable, double-cliquez sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/ Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.

Propriété	Description
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Type de données	Spécifie le type de données de la variable. Vous pouvez choisir un type de données standard ou spécifier un classificateur. Vous pouvez utiliser les outils situés à droite de la liste pour créer un classificateur, passer en revue les classificateurs disponibles ou afficher les propriétés du classificateur sélectionné.
Multiplicité	Spécifie le nombre d'instances de la variable. Si la multiplicité est une plage de valeurs, cela signifie que le nombre de variables peut varier au moment de l'exécution. Vous pouvez choisir une des valeurs suivantes : • * – de aucune à un nombre illimitée. • 0..* – de zéro à un nombre illimitée. • 0..1 – de zéro à une. • 1..* – de une à un nombre illimitée. • 1..1 – exactement une.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Activités décomposées et sous-activités

Une activité décomposée est une activité qui contient des sous-activités. Elle équivaut à la description d'un SubactivityState et d'un graphe d'activité dans UML. L'activité décomposée se comporte comme un package spécialisé ou un conteneur. Une sous-activité peut elle-même être décomposée en sous-activités, et ainsi de suite.

Remarque : Pour afficher toutes les activités situées dans des activités décomposées du modèle, cliquez sur l'outil Inclure les activités décomposées dans la liste des activités accessible depuis le menu Modèle.

Vous pouvez décomposer des activités soit directement dans le diagramme en utilisant une vue composite modifiable ou en utilisant des sous-diagrammes. Les sous-objets créés dans l'un de ces modes peuvent être affichés dans les deux modes, mais les deux modes ne sont pas automatiquement synchronisés. La vue composite **Modifiable** permet de rapidement décomposer des activités et de montrer les liens directs entre les activités et les sous-activités, tandis que le mode **Lecture seule (sous-diagramme)** favorise une décomposition plus formelle et est plus approprié si vous utilisez de nombreux niveaux de décomposition.

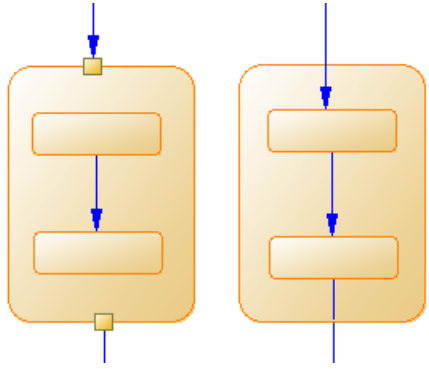
Vous pouvez choisir le mode d'affichage des activités composites objet par objet. Pour ce faire, pointez sur un symbole, cliquez le bouton droit de la souris, puis sélectionnez le mode souhaité dans le menu **Vue composite**.

Vous ne pouvez pas créer un package ou tout autre type de diagramme UML au sein d'une activité décomposée, mais vous pouvez utiliser des raccourcis vers des packages.

Utilisation du mode de vue composite modifiable

Vous pouvez décomposer une activité et créer des sous-activités dans cette dernière tout simplement en créant ou faisant glisser d'autres activités sur son symbole. Vous pouvez redimensionner le symbole parent si besoin est et y créer autant de sous-activités que vous le souhaitez. Vous pouvez décomposer une sous-activité en créant ou en faisant glisser sur son symbole une autre activité, et ainsi de suite.

Les flux peuvent lier des activités situées au même niveau, ou bien lier des activités dans le diagramme parent à des sous-activités contenues dans la vue composite directe :

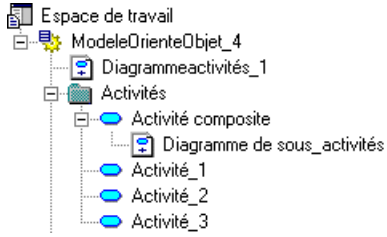


Conversion d'un activité atomique en activité décomposée

Vous pouvez convertir une activité atomique en activité décomposée de l'une des façons suivantes :

- Maintenez la touche **Ctrl** enfoncée et double-cliquez sur le symbole de l'activité (vous affichez directement la sous-activité)
- Affichez la feuille de propriétés de l'activité et, sur l'onglet Général, sélectionnez l'option **Activité décomposée**
- Pointez sur l'activité, cliquez le bouton droit de la souris, puis sélectionnez **Décomposer l'activité** dans le menu contextuel

Lorsque vous créez une activité décomposée, un diagramme de sous-activité, initialement vide, est ajouté au-dessous de son entrée dans l'Explorateur d'objets :



Pour ouvrir un diagramme de sous-activité, maintenez la touche **Ctrl** enfoncée et double-cliquez sur le symbole de l'activité décomposée, ou bien double-cliquez sur le diagramme approprié dans l'Explorateur d'objets.

Vous pouvez ajouter des objets dans un diagramme de sous-activité de la même façon que vous les ajoutez dans un diagramme d'activités. Toute activité que vous ajoutez à un diagramme de sous-activité fera partie de son activité décomposée parent et sera répertoriée sous l'activité décomposée dans l'Explorateur d'objets.

Vous pouvez créer plusieurs diagrammes de sous-activité dans une activité décomposée, mais nous vous conseillons de n'en créer qu'un seul, à moins que vous ne soyez amené à concevoir des cas d'exception, par exemple pour gérer des cas d'erreur.

Remarque : Vous pouvez localiser n'importe quel objet ou diagramme dans l'Explorateur d'objets à partir de la fenêtre de diagramme courante. Pour ce faire, pointez sur le symbole de l'objet, ou bien sur le fond du diagramme, cliquez le bouton droit de la souris, puis sélectionnez **Edition > Rechercher dans l'Explorateur d'objets**.

Conversion d'un diagramme d'activités en activité décomposée

Vous pouvez convertir un diagramme d'activités en activité décomposée en utilisant l'Assistant Convertir en activité décomposée. Vous ne pouvez effectuer cette opération que sur un diagramme qui contient des objets. En convertissant un diagramme en activité décomposée, vous pouvez ensuite utiliser l'activité décomposée dans un autre diagramme d'activités.

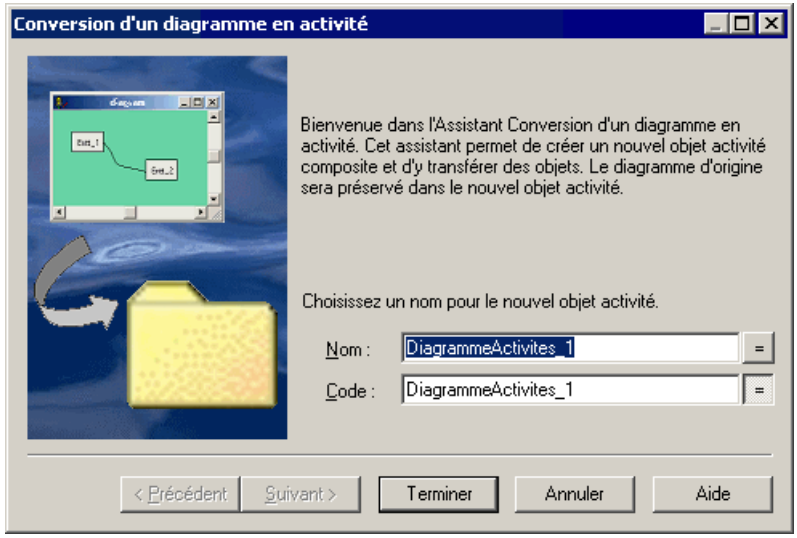
1. Pointez sur le noeud du diagramme dans l'Explorateur d'objets, cliquez le bouton droit de la souris et sélectionnez Convertir en activité décomposée dans le menu contextuel.

   ou

   Pointez sur le fond du diagramme, cliquez sur le bouton droit de la souris et sélectionnez **Diagramme > Convertir en activité décomposée** dans le menu contextuel.

   ou

   Sélectionnez **Outils > Convertir en activité décomposée**.



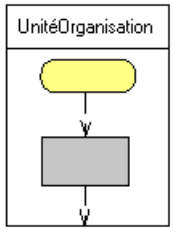
2. Saisissez un nom et un code dans la page Conversion d'un diagramme en activité décomposée, puis cliquez sur Suivant pour afficher la page Sélection des objets à déplacer.
3. Sélectionnez les activités que vous souhaitez déplacer dans le nouveau diagramme d'activité décomposée. Les activités que vous sélectionnez seront déplacées dans l'Explorateur d'objets sous la nouvelle activité décomposée. Celles que vous ne sélectionnez pas restent à leur emplacement actuel dans l'Explorateur d'objets et seront représentées dans le nouveau diagramme de sous-activité sous la forme de raccourcis.
4. Cliquez sur Terminer pour quitter l'Assistant. La nouvelle activité décomposée et son diagramme de sous-activité sont créés, et tous les objets sélectionnés pour être déplacés s'affichent maintenant sous l'objet décomposé dans l'Explorateur d'objets.

Unités d'organisation (MOO)

Une *unité d'organisation* peut représenter une société, un système, un service, une organisation, un utilisateur ou un rôle, qui est responsable d'une activité. Dans UML, une unité d'organisation est appelée un couloir, tandis que dans le MOO, le terme "couloir" fait référence au symbole de l'unité d'organisation.

Remarque : Pour activer l'affichage des couloirs d'unité d'organisation, sélectionnez **Outils > Préférences d'affichage**, puis cochez la case **Couloir d'unité d'organisation** sur la page **Général**, ou bien pointez sur le fond du diagramme, cliquez le bouton droit de la souris, puis sélectionnez **Activer le mode Couloir**.

Une unité d'organisation peut être créée dans un diagramme d'activités et peut contenir n'importe quels autres objets d'un diagramme d'activités :



Création d'une unité d'organisation

Vous créez une unité d'organisation afin de montrer le participant responsable de l'exécution des activités.

Pour pouvoir ajouter des couloirs d'unités d'organisation dans vos diagrammes, vous devez sélectionner **Outils > Préférences d'affichage**, puis cocher la case **Couloirs d'unité d'organisation**.

- Utilisez l'outil **Couloir d'unité d'organisation** dans la Boîte à outils. Cliquez à proximité ou sur un couloir ou pool de couloirs existant pour ajouter un couloir au pool. Cliquez à l'écart de tout couloir pour créer un nouveau pool.
- Sélectionnez **Modèle > Unités d'organisation** pour afficher la boîte de dialogue Liste des unités d'organisation, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Unité d'organisation.**

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'une unité d'organisation

Pour visualiser ou modifier les propriétés d'une unité d'organisation, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.

Propriété	Description
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension. Une unité d'organisation peut avoir les stéréotypes prédéfinis suivants : • Rôle. Définit un rôle que l'utilisateur joue. • Utilisateur. Définit un utilisateur. • Groupe. Définit un groupe d'utilisateurs. • Société. Définit une société. • Organisation. Définit une organisation comme un ensemble. • Division. Définit une division au sein d'une structure globale. • Service. Définit un service au sein d'une structure globale.
Organisation parent	Spécifie une autre unité d'organisation comme unité d'organisation parent de celle-ci. Par exemple, vous pouvez souhaiter décrire une hiérarchie organisationnelle entre un service Serv1 et un chef de service ChefServ1, avec ChefServ1 comme unité d'organisation parent de Serv1. La relation entre les unités d'organisation parent et enfant peut être utilisée pour regrouper les couloirs ayant le même parent (voir *Regroupement de couloirs* à la page 189).
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Attachement d'activités à des unités d'organisation

Vous attachez des activités à des unités d'organisation afin de rendre ces dernières responsables de ces activités. Lorsque des activités sont attachées à une unité d'organisation affichée sous forme de couloir, le nom de l'unité d'organisation s'affiche dans la liste Unité d'organisation de leur feuille de propriétés.

Vous attachez des activités à des unités organisation en les créant dans le couloir approprié (ou en y déplaçant des activités existantes). Vous pouvez également sélectionner un nom d'unité d'organisation dans la liste Unité d'organisation de la feuille de propriétés d'activité, puis cliquer sur OK pour l'attacher.

Pour détacher des activités d'une unité d'organisation, faites-les glisser hors du couloir correspondant, ou bien sélectionnez <Aucune> dans la liste Unité d'organisation de la feuille de propriétés de chaque activité.

Affichage d'une activité communautaire

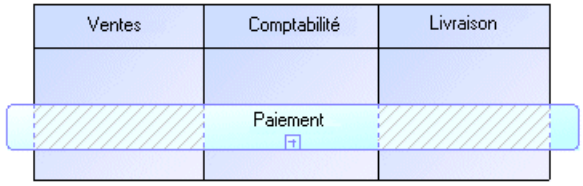
Une activité communautaire est une activité décomposée dont les sous-activités sont gérées par plusieurs unités d'organisation.

1. Affichez la feuille de propriétés d'une activité décomposée.
2. Sélectionnez **Activité communautaire** dans la liste Unité d'organisation, puis cliquez sur **OK**.

   Cette valeur est disponible uniquement pour les activités décomposées.
3. Dans le diagramme, redimensionnez le symbole d'activité décomposée de façon à recouvrir les couloirs appropriés.

   La couleur de fond du symbole change sur les couloirs selon que chacun d'entre eux est ou non responsable de sous-activités.

Dans l'exemple suivant, toutes les sous-activités de Paiement sont gérées dans l'unité d'organisation Comptabilité :



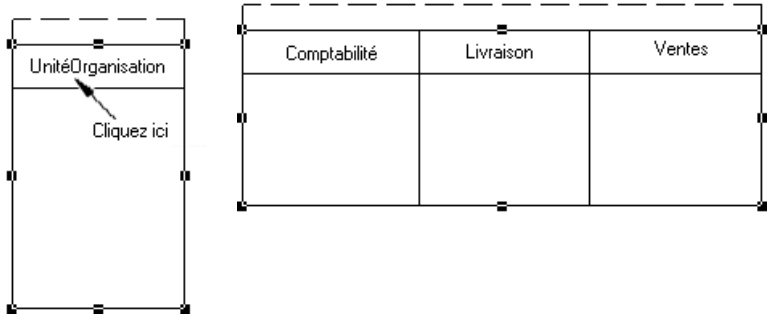
Le fond du symbole de l'activité communautaire est plus clair et hachuré sur Ventes et Livraison car :

- Elles ne gèrent pas de sous-activités
- Elles n'ont pas de symbole dans le diagramme de sous-activité

Notez que cet affichage n'est pas possible en mode de vue composite.

Déplacement, redimensionnement, copie et collage de couloirs

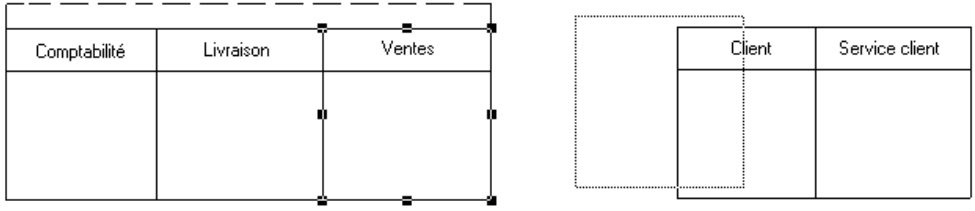
Chaque groupe d'un ou de plusieurs couloirs forme un pool. Vous pouvez créer plusieurs pools dans un diagramme, et chaque pool est généralement utilisé pour représenter une organisation distincte. Pour sélectionner un couloir individuel dans un pool, cliquez sur son en-tête. Pour sélectionner un pool, cliquez sur l'un de ses couloirs ou placez le curseur au-dessus du pool, jusqu'à ce qu'apparaisse une flèche verticale pointant vers le cadre, puis cliquez pour afficher le cadre de sélection.



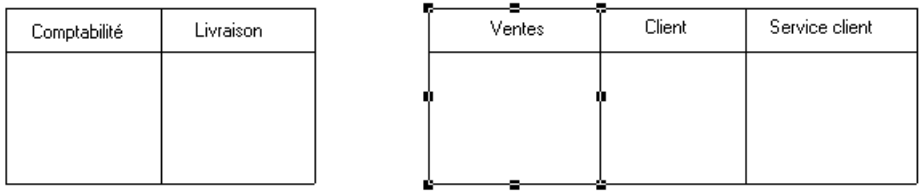
Remarque : La fonctionnalité de disposition automatique n'est pas disponible avec les unités d'organisation affichées sous la forme de couloirs.

Lorsque vous déplacez un couloir ou un pool au sein d'un même diagramme, tous les symboles contenus dans les couloirs sont déplacés simultanément (et ce, même si certains éléments ne sont pas attachés de façon formelle). Lorsque vous déplacez un couloir ou un pool vers un autre dossier ou diagramme, les symboles contenus dans le ou les couloirs ne sont pas copiés.

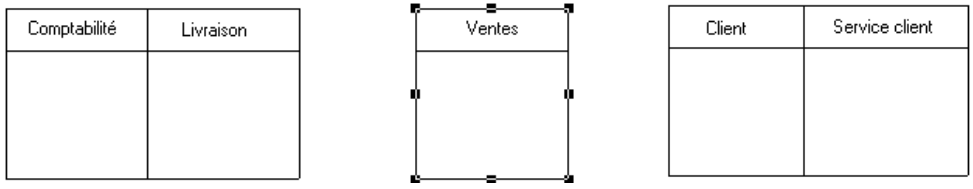
Si un couloir est déposé sur un autre couloir ou pool ou à sa proximité immédiate, il rejoint le pool. Dans l'exemple suivant, Ventes forme un pool avec Comptabilité et Livraison :



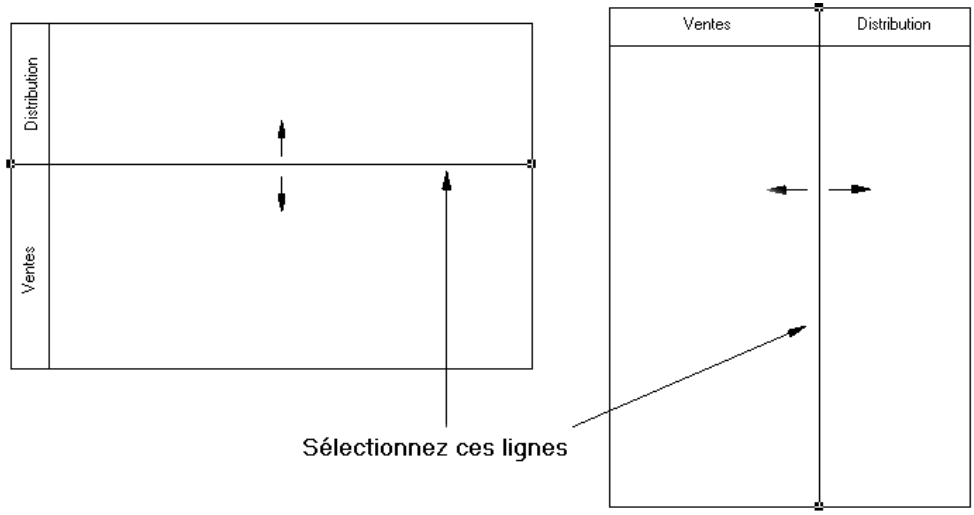
Ventes est transféré dans un autre pool contenant Client et Service client :



Si le couloir déplacé est déposé à l'écart de tout autre couloir ou pool, il forme de lui-même un nouveau pool :



Vous pouvez redimensionner des couloirs au sein d'un pool en cliquant sur la ligne qui les sépare et en la faisant glisser. Vous pouvez redimensionner un pool en sélectionnant l'une des poignées située autour du pool, puis en la faisant glisser dans la direction de votre choix. Les éventuels autres pools de votre diagramme peuvent également être redimensionnés afin de préserver la disposition générale du diagramme.



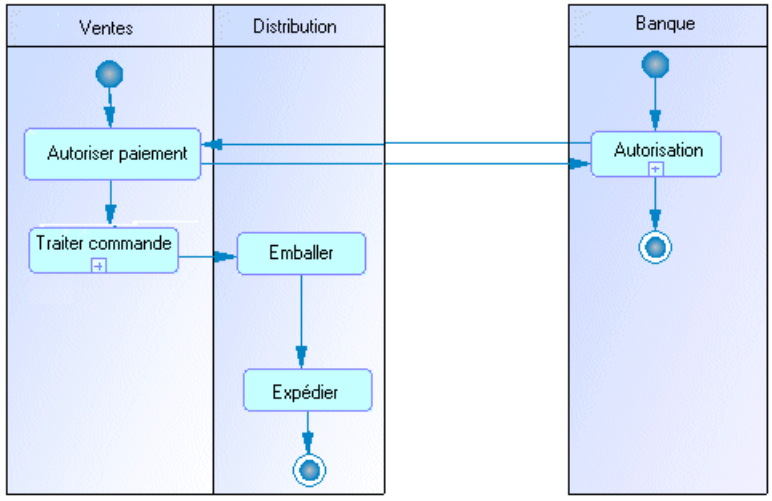
Si vous changez la largeur ou la hauteur d'un couloir individuel, tous les symboles d'activité attachés à ce couloir conservent leur position.

Création de liens entre des pools de couloirs

Vous créez des liens entre des pools ou entre des activités contenues dans des pools distincts afin de représenter les liens entre elles.

Pour créer des liens entre des pools de couloirs, cliquez sur l'outil **Flux** dans la Boîte à outils et tracez un flux depuis une activité contenue dans pool vers une autre activité contenue dans un autre pool, ou bien d'un pool à l'autre.

Dans l'exemple suivant, les flux passent de Autoriser paiement dans le couloir Ventes à Autorisation dans le couloir Banque contenu dans un autre pool :



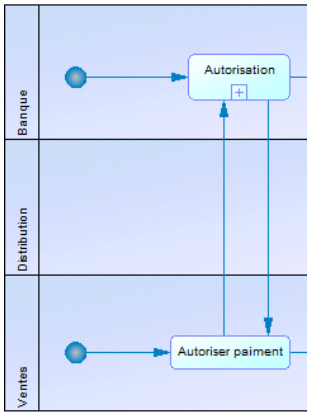
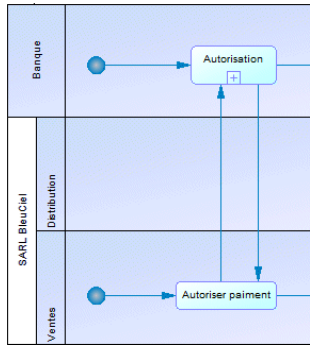
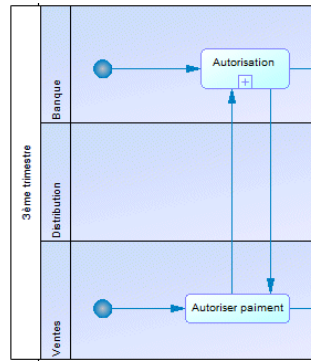
Remarque : De tels liens entre des activités contenues dans des pools distincts ne sont pas visibles lorsque les couloirs ne sont pas en mode de vue composite.

Regroupement de couloirs

Vous groupez des couloirs d'unité d'organisation au sein d'un pool afin de les organiser sous un parent commun ou sous un nom personnalisé.

Pour grouper des couloirs au sein d'un pool, pointez sur ce pool, cliquez le bouton droit de la souris, puis sélectionnez **Type de groupe de couloirs**, puis :

- **Par parent** - pour affecter le nom du dernier parent commun au groupe
- **Personnalisé** - pour affecter le nom de votre choix au groupe. Ensuite, vous devez sélectionner au moins deux couloirs attachés, puis sélectionner **Symbole > Grouper les symboles** dans la barre de menus afin d'afficher un nom par défaut que vous pouvez modifier.

Aucun groupe	Groupe parent	Groupe défini par l'utilisateur
Les trois couloirs sont situés dans un pool, sans regroupement : 	Ventes et Distribution sont regroupés par leur parent : 	Le pool est affecté à un groupe défini par l'utilisateur nommé 3ème trimestre : 

Pour dissocier des couloirs, sélectionnez **Séparer les symboles** dans le menu contextuel du pool, ou bien sélectionnez **Symbole > Séparer les symboles**.

Changement de l'orientation et du format des couloirs

Vous pouvez choisir d'orienter vos couloirs verticalement de haut en bas, ou horizontalement de gauche à droite. Tous les couloirs doivent être orientés de la même manière.

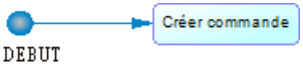
Sélectionnez **Outils > Préférences d'affichage**, sélectionnez l'option appropriée dans la zone de groupe **Couloir d'unité d'organisation**, puis cliquez sur **OK**.

Débuts et fins (MOO)

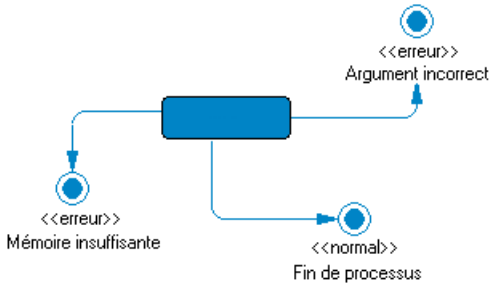
Un *début* est le point de départ d'un flux représenté dans le diagramme, et la *fin* est un point de terminaison du flux.

Vous pouvez créer un début dans les types de diagrammes suivants :

- Diagramme d'activités - un ou plusieurs par diagramme
- Diagramme d'états-transitions - un ou plusieurs par diagramme
- Diagramme d'interactions - un seul par diagramme



Vous pouvez créer plusieurs fins au sein d'un même diagramme si vous souhaitez représenter plusieurs cas de fins différents, par exemple des scénarios d'erreur :



Vous ne devez pas utiliser le même début dans deux diagrammes, et vous ne pouvez pas créer des raccourcis pour des débuts ou des fins.

En l'absence de fin, le diagramme contient une activité sans fin. En revanche, une activité décomposée doit systématiquement contenir au moins une fin.

Remarque : Le début est comparé et fusionné dans la fonctionnalité de fusion de modèles, qui s'assure qu'il n'y a qu'un seul début par activité décomposée.

Création d'un début ou d'une fin

Vous pouvez créer un début ou une fin à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil Début ou **Fin** dans la Boîte à outils.
- Sélectionnez **Modèle > Débuts** ou **Modèle > Fins**, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Début** ou **Nouveau > Fin**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'un début ou d'une fin

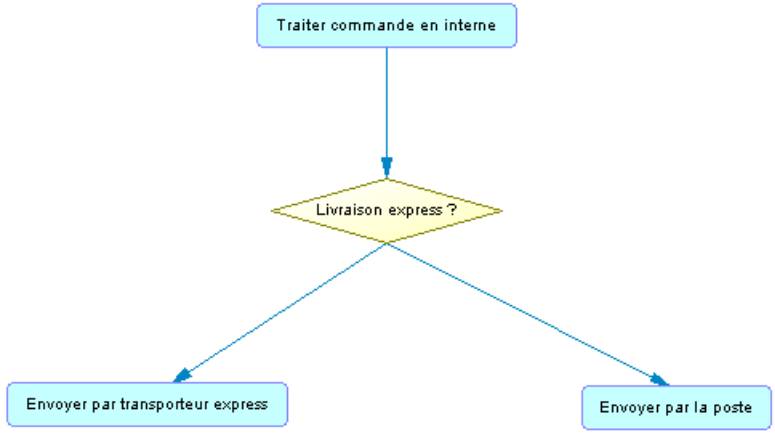
Pour visualiser ou modifier les propriétés d'un début ou d'une fin, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/ Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Terminaison	[fins uniquement] Spécifie si la fin est la terminaison de l'activité entière ou simplement celle d'un des flux possibles.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Décisions (MOO)

Une *décision* spécifie quel chemin emprunter lorsqu'un choix parmi plusieurs options est possible. Une décision peut avoir une ou plusieurs transitions entrantes et une ou plusieurs transitions sortantes, chacune dotée d'une *condition de garde* distincte, qui doit être satisfaite pour qu'un flux associé exécute une action. Vos conditions de garde doivent éviter toute ambiguïté en ne se recoupant pas, mais doivent couvrir toutes les possibilités afin d'éviter un gel du processus.

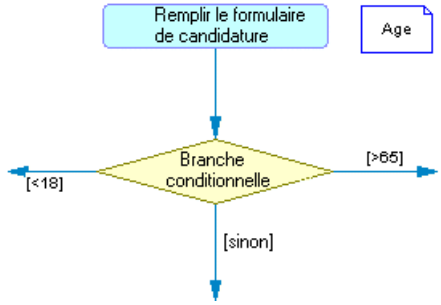
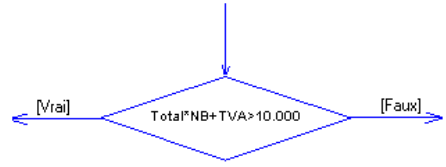


Une décision peut être créée dans les types de diagramme suivants :

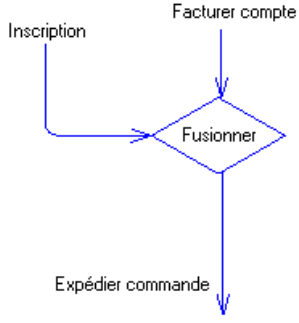
- Diagramme d'activités
- Diagramme d'interactions

Une décision peut représenter :

- Une branche conditionnelle : un flux d'entrée et plusieurs flux de sortie. Vous pouvez afficher une condition sur le symbole de décision afin de factoriser les conditions attachées aux flux :

Sans condition sur le symbole	Avec condition sur le symbole
Dans l'exemple suivant, le flux de gauche contrôle si l'âge spécifié dans le formulaire de candidature est inférieur à 18 ans, et à droite si l'âge est supérieur à 65 ans, et prend une autre route si l'âge n'est pas mentionné :	Dans l'exemple suivant, la condition Total * NB + TVA > 10.000 est saisie dans l'onglet Condition de la feuille de propriétés d'une décision, et True et False sont saisis dans les onglets Condition des flux :
	

- Une fusion : plusieurs flux entrants et un seul flux sortant. Dans l'exemple suivant, les flux Inscription et Facturer compte fusionnent pour devenir le flux Expédier commande :



Une décision permet de créer des flux complexes de type :

- if ... then ... else ...
- switch ... case ...
- do ... while ...
- loop

- for ... next ...

Remarque : Il n'est pas possible d'attacher deux flux de directions opposées au même angle d'un symbole de décision.

Création d'une décision

Vous pouvez créer une décision à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Décision** dans la Boîte à outils.
- Sélectionnez **Modèle > Décisions** pour afficher la boîte de dialogue Liste des décisions, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Décision**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'une décision

Pour visualiser ou modifier les propriétés d'une décision, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/ Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.

Propriété	Description
Type	Valeur calculée en lecture seule qui montre le type de la décision : • Incomplète - Pas de flux d'entrée ni de sortie, ou un seul flux d'entrée et un seul flux de sortie. • Branche conditionnelle : Un flux d'entrée et plusieurs flux de sortie. • Fusion : plusieurs flux entrants et un seul flux sortant.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Les onglets suivants sont également disponibles :

- **Condition** - contient les propriétés suivantes :

Propriété	Description
Alias	Nom abrégé de la décision, à afficher en regard de son symbole dans le diagramme.
Condition (zone de texte)	Spécifie une condition à évaluer afin de déterminer comme la décision doit être traversée. Vous pouvez saisir les informations appropriées directement dans cette zone, mais aussi ouvrir, insérer et enregistrer des fichiers de texte. Vous pouvez ouvrir directement l'onglet Condition en pointant sur le symbole de décision, cliquant le bouton droit de la souris et sélectionnant Condition dans le menu contextuel.

Synchronisations (MOO)

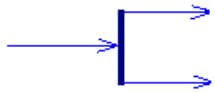
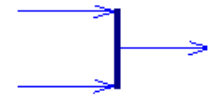
Une *synchronisation* permet de scinder ou de synchroniser le contrôle entre plusieurs actions concurrentes.

Une synchronisation peut être créée dans les diagrammes suivants :

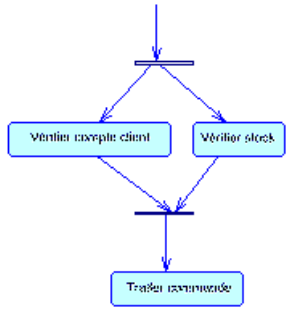
- Diagramme d'activité
- Diagramme d'états-transitions
- Diagramme d'interactions

Les synchronisations sont représentées sous la forme de traits verticaux ou horizontaux. Vous pouvez changer l'orientation du symbole en pointant sur ce dernier, en cliquant le bouton droit de la souris, puis en sélectionnant **Afficher verticalement** ou **Afficher horizontalement**.

Une synchronisation peut être une :

Fourche	Jointure
Scinde un flux entrant en plusieurs flux sortants indépendants exécutés en parallèle : 	Fusionne plusieurs flux entrants en un seul flux sortant. Tous les flux entrants doivent atteindre la jointure avant que le flux sortant unique ne puisse poursuivre : 

Dans l'exemple suivant, le flux provenant de la première synchronisation est scindé en deux flux séparés entrant dans les processus Vérifier compte client et Vérifier stock. Les deux flux sont ensuite fusionnés en une autre synchronisation qui produit un flux unique, ce dernier aboutissant au processus Traiter commande :



Création d'une synchronisation

Vous pouvez créer une synchronisation à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisation l'outil **Synchronisation** dans la boîte à outils.
- Sélectionnez **Modèle > Synchronisations** pour afficher la boîte de dialogue Liste des synchronisations, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Synchronisation**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'une synchronisation

Pour visualiser ou modifier les propriétés d'une synchronisation, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/ Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Dépassement de délai	Définit un délai d'attente maximal avant que toutes les transitions se terminent. Vide lorsque la valeur est égale à zéro.
Type	[lecture seule] Calcule la forme de la synchronisation : • Incomplet - Aucune transition entrante ou sortante, ou une transition entrante et une transition sortante. • Branche conditionnelle – Une transition entrante et plusieurs transitions sortantes. • Fusion - Plusieurs transitions entrantes et une transition sortante.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Flux (MOO)

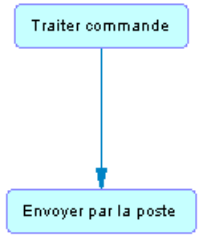
Un *flux* est la route que le flux de contrôle emprunte pour transiter entre des objets. L'acheminement du flux s'effectue via l'utilisation de conditions de garde définies sur les flux. Si la condition est remplie, le contrôle est passé à l'objet suivant.

Un flux peut être créé dans les types de diagramme suivants :

- Diagramme d'activités
- Diagramme d'interactions

Un flux entre une activité et un noeud d'objet indique que l'exécution de l'activité met un objet dans un état particulier. Lorsqu'un événement particulier se produit ou que des conditions particulières sont remplies, le flux de contrôle passe de l'activité au noeud d'objet. Un flux entre un noeud d'objet et une activité signifie que l'activité utilise cet état particulier dans son exécution. Dans les deux cas, le flux est représenté sous la forme d'une simple flèche.

Dans l'exemple suivant, le flux lie le processus Traiter commande au processus Envoyer par la poste :



Un flux peut lier des raccourcis. Un flux accepte les raccourcis aux deux extrémités afin d'empêcher son déplacement automatique lorsqu'un processus est déplacé. Dans ce cas, le processus est déplacé et laisse un raccourci, mais contrairement aux autres liens, le flux n'est pas déplacé. Les raccourcis de flux n'existent pas, et les flux restent en place dans tous les cas.

Les règles suivantes s'appliquent :

- Les *flux réflexifs* (le processus source est en même temps le processus de destination) sont admis sur les processus.
- Deux flux distincts peuvent être définis entre la même paire d'objets source et destination, on parle alors de *flux parallèles*.

Remarque : Lorsque des flux sont comparés et fusionnés par la fonctionnalité de fusion de modèles, ils sont mis en correspondance d'abord par événement déclencheur, puis par leur nom calculé. Lorsque deux flux correspondent, les actions de déclenchement sont automatiquement mises en correspondance car il ne peut y avoir plusieurs actions de déclenchement.

Création d'un flux

Vous pouvez créer un flux à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Flux/Flux de ressource** dans la Boîte à outils.
- Sélectionnez **Modèle > Flux** pour afficher la boîte de dialogue Liste des flux, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Flux**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'un flux

Pour visualiser ou modifier les propriétés d'un flux, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les

onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom et le code sont en lecture seule. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Source / Destination	Spécifie l'objet dont part ou bien auquel aboutit le flux. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné. Vous pouvez également afficher la feuille de propriétés des objets source et destination en cliquant sur le bouton approprié dans la partie supérieure de la feuille de propriétés du flux.
Type de flux	Vous pouvez créer vos propres types de flux dans la liste, ou bien sélectionner l'une des valeurs suivantes : • Succès. Définit un flux terminé avec succès. • Dépassement de délai. Définit l'occurrence d'un dépassement de délai. • Erreur technique. • Erreur de gestion. • Compensation. Définit un flux de compensation. Le type de flux n'est pas disponible si vous associez un événement au flux sur l'onglet Condition.
Poids	Spécifie le nombre d'objets consommés à chaque traversée.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Remarque : Vous pouvez afficher plusieurs flux entrants et sortants d'un processus à partir de sa feuille de propriétés en cliquant sur les sous-onglets **Flux entrants** et **Flux sortants** de son onglet **Dépendances**.

Les onglets suivants sont également disponibles :

- **Paramètres** - répertorie les paramètres qui sont transmis via le flux. La liste est automatiquement renseignée si vous tracez le flux entre deux paramètres d'activité.
- **Transformation** - spécifie une transformation de données à appliquer aux jetons d'entrée. Par exemple, elle peut extraire une seule valeur d'attribut d'un objet d'entrée.

Noeuds d'objet (MOO)

Un *noeud d'objet* est l'association entre un objet (instance d'une classe) et un état. Il représente un objet dans un état particulier.

Le symbole du noeud d'objet est un rectangle, comme illustré ci-dessous :

Un noeud d'objet peut être créé dans les types de diagramme suivants :

- Diagramme d'activités

Dans le diagramme d'activités, un même objet peut évoluer après que plusieurs actions définies par des activités aient été exécutées. Par exemple, un document peut évoluer en partant de son état initial vers un brouillon, puis vers une version corrigée avant de passer finalement à l'état de document approuvé.

Vous pouvez tracer une lien allant d'une activité à un noeud d'objet, et inversement :

- Flux allant d'une activité à un noeud d'objet - signifie que l'exécution de l'activité met l'objet dans un état spécifique. Représente alors le résultat d'une activité.
- Flux allant du noeud d'objet à une activité - signifie que l'activité utilise cet état spécifique dans son exécution. Représente un flux de données entre ces deux éléments.

Lorsqu'une activité met un objet dans un état et que cet objet est immédiatement réutilisé par une autre activité, cela dénote une transition entre deux activités avec un échange de données, et le noeud d'objet représente cet échange de données.

Par exemple, les noeuds d'objet Commande approuvée et Facture éditée sont liés aux classes Commande et Facture, qui sont représentées dans un diagramme de classe distinct :

Création d'un noeud d'objet

Vous pouvez créer un noeud d'objet à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Noeud d'objet** dans la Boîte à outils.
- Sélectionnez **Modèle > Noeuds d'objet** pour afficher la boîte de dialogue Liste des noeuds d'objet, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Noeud d'objet**.

- Faites glisser un classificateur depuis l'Explorateur d'objets dans un diagramme d'activités. Le nouveau noeud d'objet sera lié au nom du classificateur et affichera son nom.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'un noeud d'objet

Pour visualiser ou modifier les propriétés d'un noeud d'objet, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Type de données	Spécifie le type de données du noeud d'objet. Vous pouvez utiliser les outils à droite de la liste pour créer un classificateur, parcourir l'arborescence des classificateurs disponibles ou afficher les propriétés du classificateur sélectionné.
Etat	Spécifie l'état du noeud d'objet. Vous pouvez saisir le nom d'un état dans cette zone ou, si un classificateur a été spécifié comme type de données, sélectionner un de ses états dans la liste. Vous pouvez utiliser les outils à droite de la liste pour créer un état, parcourir l'arborescence des états disponibles ou afficher les propriétés de l'état sélectionné.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Etats (MOO)

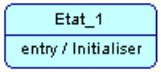
Un *état* représente une situation lors de la vie d'un classificateur qui est généralement spécifiée par des conditions. Il peut également être défini comme la situation d'un classificateur en attente d'événements. La stabilité et la durée sont deux caractéristiques d'un état.

Un état peut être créé dans les types de diagramme suivants :

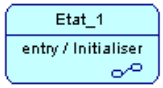
- Diagramme d'états-transitions

Un état peut être atomique ou décomposé :

- Un état atomique ne contient pas de sous-état, et son symbole se présente comme suit :



- Un état décomposé contient des sous-états, qui sont représentés par un sous-diagramme, et ont un symbole qui se présente comme suit :



Pour plus d'informations sur les états décomposés, voir *Etats décomposés et sous-états* à la page 205.

Plusieurs états d'un diagramme d'états-transitions correspondent à plusieurs situations lors de la vie du classificateur.

Les événements ou conditions définis sur les transitions sortantes définissent la stabilité d'un état. Certaines actions peuvent être associées à un état, tout particulièrement lorsque l'objet entre dans l'état ou le quitte. Certaines actions peuvent également être effectuées lorsque des événements se produisent dans l'état ; ces actions sont appelées transitions internes et ne provoquent pas de changement de l'état.

Vous ne pouvez pas utiliser des raccourcis vers des états.

Faire glisser un cas d'utilisation, un composant, ou une classe dans un diagramme d'états-transitions

Le diagramme d'états-transitions décrit le comportement d'un classificateur. Pour mettre en exergue la relation entre un classificateur et un état, vous pouvez définir le classificateur de contexte d'un état à l'aide de la liste Classificateur dans la feuille de propriétés de l'état. Vous liez ainsi un état à un cas d'utilisation, à un composant ou à une classe.

Vous avez également la possibilité de déplacer, copier/coller ou bien glisser-déposer une classe, un cas d'utilisation ou un composant dans un diagramme d'états-transitions pour créer automatiquement un état associé à l'élément déplacé.

Création d'un état

Vous pouvez créer un état à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Etat** dans la Boîte à outils.
- Sélectionnez **Modèle > Etats** pour afficher la boîte de dialogue Liste des états, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Etat**.
- Faites glisser une classe, un cas d'utilisation ou un composant depuis l'Explorateur d'objets dans le diagramme.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'un état

Pour visualiser ou modifier les propriétés d'un état, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/ Code/ Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Classificateur	Classificateur lié à l'état. Il peut s'agir d'un cas d'utilisation, d'une classe ou d'un composant. Lorsqu'un classificateur est sélectionné, il s'affiche entre crochets après le nom de l'état dans l'Explorateur d'objets. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.

Propriété	Description
Etat composite	Si vous sélectionnez l'option Etat décomposé, l'état devient un état décomposé. Si vous sélectionnez l'option Etat atomique, l'état devient un état atomique et ses objets enfant sont supprimés.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Onglet Actions

Vous pouvez spécifier un jeu d'actions internes sur un état atomique ou décomposé à partir de l'onglet **Actions** dans la feuille de propriétés de l'état. Elles représentent les actions exécutées dans l'état lorsque certains événements se produisent. Vous pouvez créer les actions et définir leurs propriétés à partir de l'onglet **Actions**, ou bien double-cliquer sur la flèche au début de la ligne pour afficher la feuille de propriétés d'une action particulière.

Remarque : Vous pouvez ouvrir l'onglet **Actions** en pointant sur le symbole d'état dans le diagramme, en cliquant le bouton droit de la souris et en sélectionnant **Actions** dans le menu contextuel.

Pour plus d'informations sur les actions, voir *Actions (MOO)* à la page 213.

Onglet Evénements différés

L'onglet **Evénements différés** contient un outil **Ajouter des objets** qui permet d'ajouter des événements existants, mais pas d'en créer de nouveaux. Cette liste est similaire à la liste des règles de gestion qui permet de réutiliser des éléments, mais pas de les créer.

La différence entre un événement et un *événement différé* est qu'un événement est toujours instantané et géré de façon dynamique par un état, alors qu'un événement différé est un événement qui se produit lors d'un état particulier dans le cycle de vie d'un objet mais qui n'est pas directement utilisé par cet objet.

Un événement différé se produit dans un état spécifique, est géré dans une file d'attente, puis déclenché plus tard par un autre état du même classificateur

Onglet Sous-états

L'onglet **Sous-états** s'affiche lorsque l'état courant est décomposé afin d'afficher une liste des états enfant. Vous pouvez utiliser les outils **Ajouter une ligne** et **Supprimer** pour modifier la liste des états enfant. L'onglet **Sous-états** disparaît lorsque vous convertissez l'état courant en état atomique, car cette opération a pour effet de supprimer les enfants de l'état.

Etats décomposés et sous-états

Un état décomposé est un état qui contient des sous-états. L'état décomposé se comporte comme un package spécialisé ou un conteneur. Un sous-état peut lui-même être décomposé en sous-états supplémentaires, et ainsi de suite.

Remarque : Pour afficher tous les états décomposés du modèle, cliquez sur l'outil Inclure les états décomposés dans la liste des états accessible depuis le menu Modèle.

Vous pouvez décomposer des états soit directement dans le diagramme en utilisant une vue composite modifiable ou en utilisant des sous-diagrammes. Les sous-objets créés dans l'un de ces modes peuvent être affichés dans les deux modes, mais les deux modes ne sont pas automatiquement synchronisés. La vue composite **Modifiable** permet de rapidement décomposer des états et de montrer les liens directs entre les états et les sous-états, tandis que le mode **Lecture seule (sous-diagramme)** favorise une décomposition plus formelle et est plus approprié si vous utilisez de nombreux niveaux de décomposition.

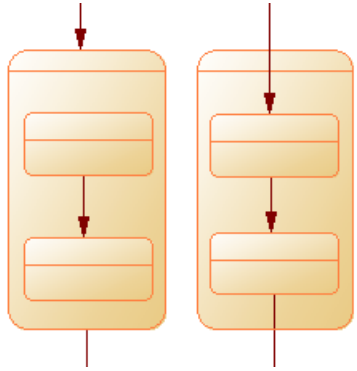
Vous pouvez choisir comment afficher les états composites objet par objet, en pointant sur un symbole, en cliquant le bouton droit de la souris, puis en sélectionnant le mode désiré dans le menu **Vue composite**.

Vous ne pouvez pas créer un package ou tout autre type de diagramme UML au sein d'un état décomposé, mais vous pouvez créer des raccourcis vers les packages.

Utilisation du mode de vue composite modifiable

Vous pouvez décomposer un état et créer des sous-états dans ce dernier tout simplement en créant ou faisant glisser d'autres états sur son symbole. Vous pouvez redimensionner le symbole parent si besoin est et y créer autant de sous-états que vous le souhaitez. Vous pouvez décomposer un sous-état en créant ou en faisant glisser sur son symbole un autre état, et ainsi de suite.

Les transitions peuvent lier deux états au même niveau, ou bien lier des états contenus dans le diagramme parent avec des sous-états dans le mode Vue composite modifiable :

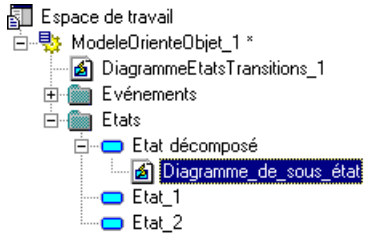


Gestion des diagrammes de sous-état

Vous pouvez convertir un état atomique en état décomposé de l'une des façons suivantes

- Maintenez la touche **Ctrl** enfoncée et double-cliquez sur le symbole d'état (vous ouvrez ainsi directement le sous-état)
- Affichez la feuille de propriétés d'un état puis, sur l'onglet Général, sélectionnez l'option **Etat décomposé**
- Pointez sur l'état et cliquez le bouton droit de la souris, puis sélectionnez **Décomposer l'état**

Lorsque vous créez un état décomposé, un diagramme de sous-état, initialement vide, est ajouté au-dessous de son entrée dans l'Explorateur d'objets :



Pour ouvrir un diagramme de sous-état, maintenez la touche **Ctrl** enfoncée et double-cliquez sur le symbole de l'état décomposé, ou bien double-cliquez sur le diagramme approprié dans l'Explorateur d'objets.

Vous pouvez ajouter des objets dans un diagramme de sous-état de la même façon que vous les ajoutez dans un diagramme d'états. Tout état que vous ajoutez à un diagramme de sous-états fera partie de son état décomposé parent et sera répertoriée sous l'état décomposé dans l'Explorateur d'objets.

Vous pouvez créer plusieurs diagrammes de sous-état dans un état décomposé, mais nous vous conseillons de n'en créer qu'un seul, à moins que vous ne soyez amené à concevoir des cas d'exception, par exemple pour gérer des cas d'erreur.

Remarque : Vous pouvez localiser n'importe quel objet ou diagramme dans l'Explorateur d'objets à partir de la fenêtre de diagramme courante. Pour ce faire, pointez sur le symbole de l'objet, ou bien sur le fond du diagramme, cliquez le bouton droit de la souris, puis sélectionnez **Edition > Rechercher dans l'Explorateur d'objets**.

Conversion d'un diagramme d'états-transitions en état décomposé

Vous pouvez convertir un diagramme d'états-transitions en état décomposé en utilisant l'Assistant Conversion d'un diagramme en état. L'option de conversion n'est disponible que si le diagramme contient déjà des objets. Lorsque vous convertissez un diagramme d'états-transitions en état décomposé, vous pouvez ensuite utiliser l'état décomposé dans un autre diagramme d'états-transitions.

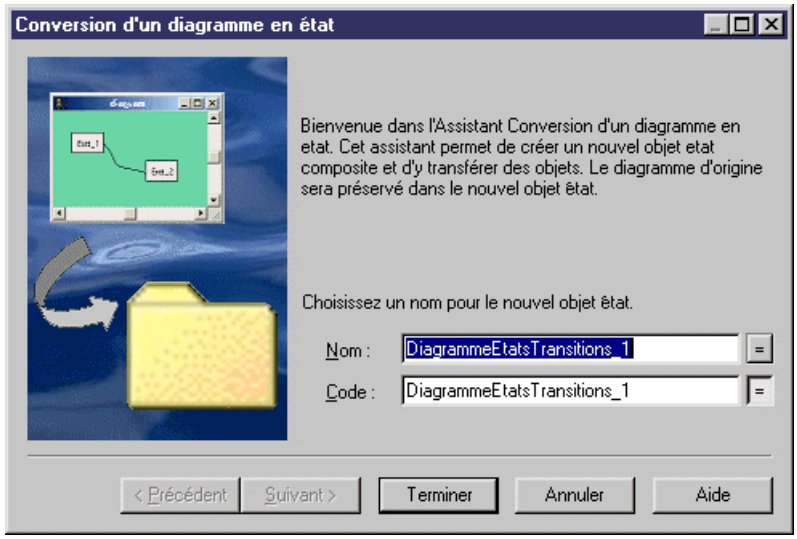
1. Pointez sur le noeud du diagramme dans l'Explorateur d'objets, cliquez le bouton droit de la souris et sélectionnez Convertir en état décomposé dans le menu contextuel.

   ou

   Pointez sur le fond du diagramme, cliquez sur le bouton droit de la souris et sélectionnez **Diagramme > Convertir en état décomposé** dans le menu contextuel.

   ou

   Sélectionnez **Outils > Convertir en état décomposé**.



2. Saisissez un nom et un code dans la page Conversion d'un diagramme en état, puis cliquez sur Suivant pour afficher la page Sélection des objets à déplacer.
3. Sélectionnez les états que vous souhaitez déplacer dans le nouveau diagramme de l'état décomposé. Les états que vous sélectionnez seront déplacés dans l'Explorateur d'objets sous le nouvel état décomposé. Ceux que vous ne sélectionnez pas restent à leur emplacement dans l'Explorateur d'objets et seront représentés par des raccourcis dans le nouveau diagramme de sous-état.
4. Cliquez sur Terminer pour quitter l'Assistant. Le nouvel état décomposé et son diagramme de sous-état sont créés, et les objets sélectionnés pour le déplacement apparaissent alors sous l'objet décomposé dans l'Explorateur d'objets.

Transitions (MOO)

Une *transition* est un lien orienté entre deux états ou activités, qu'un élément dans un état ou dans une activité peut passer à un autre état ou à une autre activité lorsqu'un événement se

produit (si une condition de garde est satisfaite, lorsqu'il en existe une). On dit alors que la transition est déclenchée.

Une transition peut être créée dans les types de diagramme suivants :

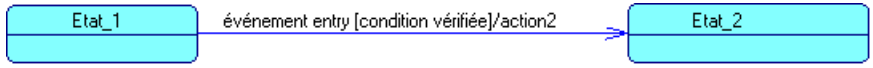
- Diagramme d'états-transitions

La transition dans le diagramme d'états-transitions est relativement similaire au flux qui se trouve dans le diagramme d'activités, avec quelques propriétés supplémentaires :

- Un événement déclencheur : il s'agit de l'événement qui déclenche la transition (lorsque vous copiez une transition, l'événement déclencheur est également copié).
- Une action de déclenchement : elle spécifie l'action à exécuter lorsque la transition est déclenchée.

Le diagramme d'activités est une version simplifiée du diagramme d'états-transitions dans laquelle les états n'ont qu'une action et où la transition est dotée d'un événement déclenché correspondant à la fin de l'action.

Le lien de transition est représenté sous la forme d'une simple flèche. L'événement associé, la condition et l'action à exécuter sont affichés au-dessus du symbole.



Les règles suivantes s'appliquent aux transitions :

- Les *transitions réflexives* n'existent que sur les activités.
- Un événement déclencheur ne peut être défini que si la source est un début ou un état.
- Deux transitions ne peuvent pas être définies entre les mêmes objets source et destination (*transitions parallèles*). La fonctionnalité de fusion de modèles interdit la création de transitions parallèles.

Remarque : Lorsque des transitions sont comparées et fusionnées par la fonctionnalité de fusion de modèles, elles sont mises en correspondance d'abord par événement déclencheur, puis par leur nom calculé. Lorsque deux transitions correspondent, les actions de déclenchement sont automatiquement mises en correspondance car il ne peut y avoir plusieurs actions de déclenchement.

Création d'une transition

Vous pouvez créer une transition à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Transition** dans la Boîte à outils.
- Sélectionnez **Modèle > Transitions** pour afficher la boîte de dialogue Liste des transitions, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Transition**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'une transition

Pour visualiser ou modifier les propriétés d'une transition, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/ Commentaire	Identifient l'objet. Le nom et le code sont en lecture seule. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Source	Point de départ de la transition. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Destination	Terminaison de la transition. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Type de flux	Représente une condition qui peut être attachée à la transition. Vous pouvez choisir parmi les types par défaut suivants ou créer votre propre type : • Succès – Définit un flux s'étant terminé avec succès. • Dépassement de délai – Définit un délai maximal. • Exception – Représente un cas d'exception.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Onglet Déclencheur

L'onglet **Déclencheur** contient les propriétés suivantes :

Propriété	Description
Evénement déclencheur	Spécifie l'événement (voir *Evénements (MOO)* à la page 210) qui déclenche la transition. Vous pouvez cliquer sur l'outil Propriétés en regard de cette zone pour afficher la feuille de propriétés de l'événement. Il est disponible uniquement pour les transitions provenant d'un état ou d'un début et ne peut être modifié dans les autres cas de figure. Lorsque vous définissez un événement déclencheur, la relation inverse est affichée dans la page Objets déclenchés de la feuille de propriétés de l'événement correspondant. L'onglet Objets déclenchés répertorie les transitions que l'événement peut déclencher.
Arguments d'événement	Spécifie une listes d'arguments d'événement (arg1, arg2,...) délimitée par des virgule.
Action	Spécifie l'action à exécuter lorsque la transition est déclenchée.
Opération	Liste en lecture seule qui répertorie les opérations du classificateur associé à l'état enfant de la transition. Permet de spécifier la mise en oeuvre de l'action à l'aide d'une opération. Cette zone est grisée et vide lorsque le classificateur n'est pas une classe
Arguments d'opération	Arguments d'un événement défini sur une opération

Onglet Condition

L'onglet **Condition** contient les propriétés suivantes :

Propriété	Description
Alias	Nom abrégé de la condition, à afficher en regard de son symbole dans le diagramme
Condition (zone de texte)	Spécifie une condition à évaluer pour déterminer si la transition doit être traversée. Vous pouvez spécifier toute information appropriée dans cette zone, mais aussi y ouvrir, y insérer et y enregistrer des fichiers de texte.

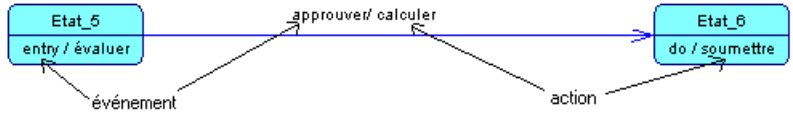
Evénements (MOO)

Un *événement* est l'occurrence de quelque chose d'observable. L'occurrence est supposée être instantanée et donc dépourvue de durée.

Un événement peut être créé dans les types de diagramme suivants :

- Diagramme d'états-transitions

Les événements véhiculent des informations spécifiées par des paramètres. Ils sont utilisés dans le diagramme d'états-transitions en association avec les transitions : ils sont ainsi attachés aux transitions pour spécifier quel événement déclenche la transition. Il sont également utilisés en association avec les actions : l'événement peut déclencher le changement d'état d'un classificateur ou l'exécution d'une action interne sur un état.



Un même événement peut être partagé entre plusieurs transitions et actions. Il est réutilisable par nature car il ne dépend pas du contexte.

L'icône de l'événement dans l'Explorateur d'objets se présente comme suit :

Evénements prédéfinis

Vous pouvez sélectionner un événement de la liste Evénement déclencheur dans les feuilles de propriétés d'action ou de transition. Vous pouvez également sélectionner une valeur d'événement prédéfinie dans la liste Evénement déclencheur si vous définissez l'événement sur une action.

La liste des événements contient les valeurs prédéfinies suivantes :

- Entry : l'action est exécutée lors de l'entrée dans l'état.
- Do : un ensemble d'actions est exécuté après l'action entry.
- Exit : l'action est exécutée au moment ou l'état est abandonné.

Exemples

Exemples d'événement :

- Une expression booléenne devenant vraie.
- La réception d'un signal.
- L'invocation d'une opération.
- Un événement temporel, par exemple un dépassement de délai ou une date atteinte.

Vous pouvez afficher les arguments d'un événement dans le diagramme d'états-transitions.

Pour plus d'informations sur les arguments d'un événement, voir *Définition d'un argument d'événement* à la page 212.

Création d'un événement

Vous pouvez créer un événement à partir de l'Explorateur d'objets ou du menu **Modèle**, ou bien à partir de la feuille de propriétés d'une transition.

- Sélectionnez **Modèle > Evénements** pour afficher la boîte de dialogue Liste des événements, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Evénement**.

- Double-cliquez sur une transition pour afficher sa feuille de propriétés, cliquez sur l'onglet **Déclencheur**, puis sur l'outil **Créer** en regard de la zone **Evénement déclencheur**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'un événement

Pour visualiser ou modifier les propriétés d'un événement, double-cliquez sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Les onglets suivants sont également disponibles :

- Paramètres - répertorie les paramètres de l'événement correspondant à la signature de l'événement (voir *Paramètres (MOO)* à la page 89).
- Dépendances - contient un sous-onglet Objets déclenchés qui affiche les actions sur les états et transitions déclenchés par cet événement.

Définition d'un argument d'événement

Les arguments d'événement sont légèrement différents des paramètres d'événement. Les arguments d'événement sont définis sur l'action ou sur la transition qui reçoit l'événement, ils dépendent d'un contexte particulier qui fait suite à cette réception.

Il s'agit d'une zone de texte définie sur l'action ou sur la transition. Vous pouvez l'éditer et séparer les arguments par une virgule (exemple : arg1, arg2). PowerAMC ne contrôle pas la cohérence entre les paramètres d'événement et les arguments d'événement.

Exemple
Un événement peut avoir un paramètre 'personne' qui peut être par exemple une personne envoyant une requête. Dans le contexte d'une transition déclenchée par cet événement, vous pouvez savoir clairement que ce paramètre est un client et donc l'appeler 'client' au lieu de 'personne'.

Actions (MOO)

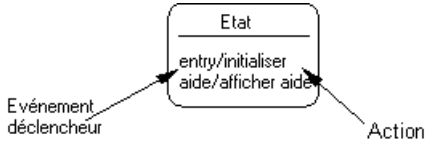
Une *action* est la spécification d'une instruction pouvant être calculée. Elle se produit dans une situation spécifique et peut comprendre plusieurs événements prédéfinis (entry, do et exit) ainsi que des transitions internes.

Une action peut être créée dans les types de diagramme suivants :

- Diagramme d'états-transitions

Les transitions internes peuvent être définies sur un état, elles sont internes pour cet état et ne provoquent pas de changement d'état ; elles effectuent des actions lorsqu'elles sont déclenchées par des événements. Les transitions internes ne doivent pas être confondues avec les transitions réflexives sur l'état car les valeurs entry et exit ne sont pas exécutées lorsque l'événement interne se produit.

Une action comporte une propriété *Evénement déclencheur* qui contient la spécification de l'événement qui déclenche l'action.



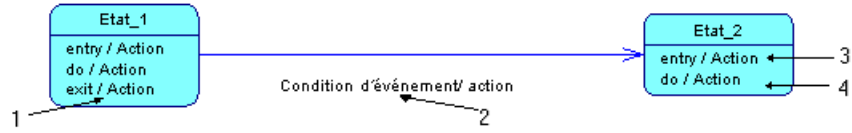
Pour plus d'informations sur les événements, voir *Evénements (MOO)* à la page 210.

Action sur l'état et sur la transition
Dans un MOO, une action est utilisée dans le diagramme d'états-transitions en association avec des états : l'action est exécutée dans l'état lors du traitement de entry ou exit. L'action est aussi utilisée en association avec des transitions : l'action est exécutée lorsque la transition est déclenchée.

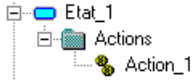
Dans UML, la différence repose sur le fait qu'une action apparaît dans les diagrammes d'interaction (en association avec les messages) et dans les diagrammes d'états-transitions.

Lorsque vous définissez une action sur un état, le nombre d'actions que vous pouvez définir n'est pas limité. En revanche, vous ne pouvez définir qu'*une seule* action sur une transition, car la transition ne peut exécuter qu'une seule action. Une action définie sur un état peut contenir l'événement qui la déclenche ; la feuille de propriétés de l'action contient la feuille de propriétés de l'événement. Une action définie sur une transition ne contient pas l'événement qui la déclenche : vous ne pouvez que saisir l'action dans une zone de texte.

Dans l'illustration suivante, vous pouvez voir des actions définies sur des états, et des actions définies sur des transitions avec l'ordre d'exécution des actions :



L'icône d'action dans l'Explorateur d'objets est un symbole formé de deux roues, il est défini sous un état mais ne s'affiche pas sous une transition.



Création d'une action

Vous pouvez créer une action à partir de la feuille de propriétés d'un été ou d'une transition.

- Affichez l'onglet **Actions** de la feuille de propriétés d'un état, puis cliquez sur l'outil **Ajouter une ligne**.
- Affichez l'onglet **Déclencheur** dans la feuille de propriétés d'une transition, puis saisissez le nom de l'action dans la zone **Action de déclenchement**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'une action

Pour visualiser ou modifier les propriétés d'une action, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.

Propriété	Description
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Evénement déclencheur	Spécifie le rôle joué par une action pour un état ou l'événement qui déclenche son exécution. Vous pouvez : • Ajouter un événement à une action en le choisissant dans la liste. • Ajouter plusieurs événements en cliquant sur le bouton Points de suspension en regard de la liste. • Créer un nouvel événement en cliquant sur l'outil Créer. • Sélectionner un événement créé dans le modèle courant ou dans d'autres modèles en cliquant sur l'outil Sélectionner l'événement déclencheur. Vous pouvez cliquer sur l'outil Propriétés en regard de cette zone pour afficher la feuille de propriétés de l'événement. Lorsqu'un événement déclencheur est défini sur une action, la relation inverse est affichée dans la feuille de propriétés de l'événement (voir *Evénements (MOO)* à la page 210).
Arguments d'événement	Arguments d'un événement défini sur un état. Les arguments sont des instances de paramètres ou des noms donnés aux paramètres dans le contexte de l'exécution d'un événement. Vous pouvez spécifier une liste d'arguments d'événement (arg1, arg2,...) dans cette zone.
Opération	Liste en lecture seule qui répertorie les opérations du classificateur associé à l'état. Elle permet de spécifier les modalités de mise en oeuvre de l'action à l'aide d'une opération. Grisée et vide lorsque le classificateur n'est pas une classe.
Arguments d'opération	Arguments d'un événement défini sur une opération.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

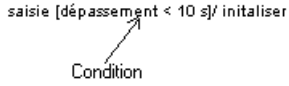
Onglet Condition

L'onglet **Condition** est disponible pour les actions définies sur des états. Vous pouvez spécifier une condition supplémentaire sur l'exécution d'une action lorsque l'événement spécifié par l'événement du trigger se produit.

La zone Alias permet de saisir une condition attachée à une action. Vous pouvez également utiliser la zone de texte dans laquelle vous détaillez la condition. Par exemple, vous pouvez rédiger des informations relatives aux conditions à exécuter, mais aussi ouvrir, insérer et enregistrer des fichiers de texte contenant des informations appropriées.

Il est recommandé d'écrire un alias (expression courte) lorsque vous utilisez une condition longue afin d'afficher cet alias plutôt que la condition dans le diagramme.

La condition est affichée entre crochets :



Points de jonction (MOO)

Un *point de jonction* permet de fusionner et scinder des transitions dans un diagramme d'états-transitions. Il est similaire à une décision dans le diagramme d'activités.

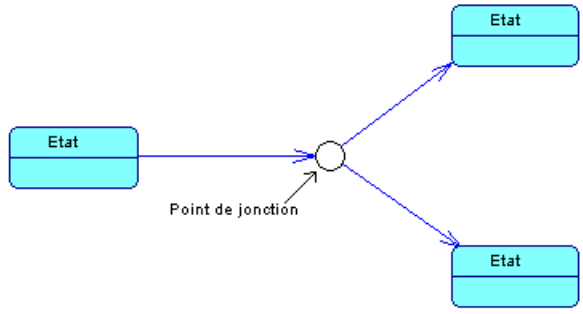
Un point de jonction peut être créé dans les types de diagramme suivants :

- Diagramme d'états-transitions

Vous ne pouvez pas utiliser des raccourcis vers un point de jonction. Un point de jonction peut dépendre de paramètres d'événement si les paramètres incluent des variables de scission ou de fusion, par exemple.

Vous pouvez attacher deux transitions de direction opposées au même symbole de point de jonction.

Le symbole d'un point de jonction est un cercle vide :



Création d'un point de jonction

Vous pouvez créer un point de jonction à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Point de jonction** dans la Boîte à outils.
- Sélectionnez **Modèle > Points de jonction** pour afficher la boîte de dialogue Liste des points de jonction, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Point de jonction**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'un point de jonction

Pour visualiser ou modifier les propriétés d'un point de jonction, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Chapitre 5 Diagrammes de mise en oeuvre

Les diagrammes décrits dans ce chapitre permettent de modéliser l'environnement physique de votre système, et de spécifier les modalités de développement de ses composants. PowerAMC fournit deux types de diagramme pour modéliser cet aspect de votre système.

- Un *diagramme de composants* représente votre système décomposé en composants auto-contenus ou sous-systèmes. Il peut montrer les classificateurs qui constituent ces systèmes avec les artefacts qui les mettent en oeuvre, et exposer les interfaces proposées ou requises par chaque composant, et les dépendances entre ces interfaces. Pour plus d'informations, voir *Diagrammes de composants* à la page 219.
- Un *diagramme de déploiement* permet de présenter l'environnement d'exécution pour un projet. Il décrit le matériel sur lequel vos composants seront exécutés et indique de quelle façon les différentes parties de ce matériel sont connectées. Pour plus d'informations, voir *Diagrammes de déploiement* à la page 221.

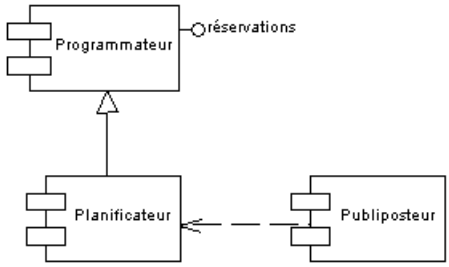
Diagrammes de composants

Un *diagramme de composants* est un diagramme UML qui fournit une représentation graphique des dépendances et des généralisations entre composants logiciels, en incluant les composants de code source, les composants de code binaire et les composants exécutables.

Remarque : Pour créer un diagramme de composants dans un MOO existant, pointez sur le modèle dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau** > **Diagramme de composants**. Pour créer un nouveau modèle, sélectionnez **Fichier** > **Nouveau modèle**, choisissez Modèle Orienté Objet comme type de modèle et **Diagramme de composants** comme premier diagramme, puis cliquez sur **OK**.

Pour plus d'informations sur les composants spécifiques à Java et .NET, voir *Chapitre 12, Java* à la page 359 et *Chapitre 16, VB .NET* à la page 449.

L'exemple suivant représente les relations entre les composants dans un système de réservation de salle de conférence :



Les diagrammes de composants sont utilisés pour définir les dépendances et relations entre objets à un niveau supérieur à celui du diagramme de classes.

Les composants doivent être conçus de façon à pouvoir être réutilisés pour plusieurs applications distinctes, ils doivent être étendus sans mettre en péril la pérennité des applications existantes.

Vous pouvez utiliser les diagrammes de composants pour modéliser la structure du logiciel, et pour montrer les dépendances entre le code source, le code binaire et les composants exécutables, de sorte que l'impact d'une modification puisse être évalué.

Un diagramme de composants est très utile lors de l'analyse et de la conception. Il permet aux analystes et aux chefs de projet de spécifier les composants dont ils ont besoin avant de les faire développer et mettre en oeuvre. Le diagramme de composants met à votre disposition une représentation des composants et simplifie leur conception, leur développement et leur maintenance, mais facilite également le déploiement, l'identification et la recherche des composants du côté serveur.

Objets du diagramme de composants

PowerAMC prend en charge tous les objets nécessaires pour construire des diagrammes de composants.

Objet	Outil	Symbole	Description
Composant		Connexion	Représente un élément partageable d'un système. Voir *Composants (MOO)* à la page 223.
Interface			Descripteur des opérations visibles de l'extérieur pour une classe, un objet ou toute autre entité dépourvue de spécification de structure interne. Voir *Interfaces (MOO)* à la page 55.
Port			Point d'interaction entre un classificateur et son environnement. Voir *Ports (MOO)* à la page 65.

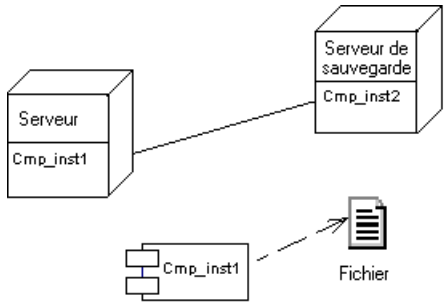
Objet	Outil	Symbole	Description
Partie			Instance de classificateur jouant un rôle particulier dans le contexte d'un autre classificateur. Voir *Parties (MOO)* à la page 63.
Généralisation			Lien entre un composant général et un composant plus spécifique qui hérite de lui et qui en enrichit les fonctionnalités. Voir *Généralisations (MOO)* à la page 101.
Réalisation			Relation sémantique entre classificateurs, dans laquelle un classificateur spécifie un contrat que l'autre classificateur s'engage à remplir. Voir *Réalisations (MOO)* à la page 109.
Lien de prérequis			Connecte composants et les interfaces. Voir *Liens de prérequis (MOO)* à la page 110.
Connecteur d'assemblage			Connecte les parties entre elles. Voir *Connecteurs d'assemblage (MOO)* à la page 112.
Connecteur de délégation			Connecte les parties aux ports situés à l'extérieur des composants. Voir *Connecteurs de délégation (MOO)* à la page 114.
Dépendance			Relation entre deux éléments de modélisation, dans laquelle tout changement intervenant sur l'un des éléments est répercuté sur le second élément. Voir *Dépendances (MOO)* à la page 105.

Diagrammes de déploiement

Un *diagramme de déploiement* est un diagramme UML qui fournit une représentation graphique de la configuration physique des éléments d'exécution de votre système.

Remarque : Pour créer un diagramme de déploiement dans un MOO existant, pointez sur le modèle dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Diagramme de déploiement**. Pour créer un nouveau modèle, sélectionnez **Fichier > Nouveau modèle**, choisissez Modèle Orienté Objet comme type de modèle et **Diagramme de déploiement** comme premier diagramme, puis cliquez sur **OK**.

Le diagramme de déploiement fournit une vue des noeuds reliés par des liens de communication. Ce diagramme permet de modéliser des noeud, des objets fichier associés aux noeuds qui sont utilisés pour le déploiement, et les relations entre les noeuds. Les noeuds contiennent des instances de composant qui peuvent être déployées dans et exécuté sur des serveurs de base de données, des serveurs d'applications ou des serveurs Web.



Les diagrammes de déploiement sont utilisés pour le déploiement effectif des composants sur des serveurs. Un déploiement représente la possibilité d'utiliser des instances.

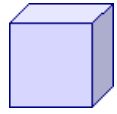
Vous pouvez utiliser le diagramme de déploiement pour établir le lien avec l'architecture physique. Il est particulièrement approprié pour modéliser les topologies en réseau, par exemple.

Vous pouvez construire un diagramme de déploiement pour représenter les types de vues suivants, depuis une architecture de haut niveau qui décrit les ressources matérielles ainsi que la distribution du logiciel dans ces ressources jusqu'au déploiement complet sur un serveur :

- Identification de l'architecture système : utiliser les noeuds et associations de noeuds uniquement.
- Identification du lien entre le logiciel et le matériel : utiliser les instances de composant, gérer leur acheminement, identifier et sélectionner les serveurs.
- Déploiement des composants sur des serveurs : inclure certains détails et ajouter des paramètres physiques.

Objets du diagramme de déploiement

PowerAMC prend en charge tous les objets nécessaires pour construire des diagrammes de déploiement.

Objet	Outil	Symbole	Description
Noeud			Elément physique qui représente une ressource de traitement, une unité physique (ordinateur, imprimante ou autre matériel). Voir *Noeuds (MOO)* à la page 230.
Instance de composant			Instance d'un composant déployable qui peut s'exécuter sur un noeud. Voir *Instances de composant (MOO)* à la page 233.
Association de noeuds			Une association entre deux noeuds signifie que les noeuds communiquent entre eux. Voir *Associations de noeuds (MOO)* à la page 237.

Objet	Outil	Symbole	Description
Dépendance			Relation entre deux éléments de modélisation, dans laquelle tout changement intervenant sur l'un des éléments est répercuté sur le second élément. Voir *Dépendances (MOO)* à la page 105.

Composants (MOO)

Un *composant* est une partie physique et remplaçable d'un système qui inclut les modalités de sa mise en oeuvre, se conforme à un jeu d'interfaces et fournit leur réalisation. Ce composant peut représenter une partie physique de la mise en oeuvre d'un système, par exemple du code informatique (source, binaire ou exécutable), des scripts ou des fichiers de commande. Il s'agit d'un élément indépendant d'un logiciel développé pour une fonction spécifique, mais pas pour une application spécifique. Il peut être constitué à partir du diagramme de classes et rédigé de toutes pièces pour le nouveau système, ou bien provenir d'autres projets ou fournisseurs.

Un composant peut être créé dans les diagrammes suivants :

- Diagramme de composants

Le symbole du composant se présente comme suit :

Un composant constitue une sorte de 'boîte noire' pour la construction de logiciels. Ainsi, du point de vue extérieur, le composant fait apparaître deux interfaces qui le décrivent, alors de que l'intérieur, le composant reflète à la fois les interfaces réalisées par la classe, les opérations des interfaces étant les opérations de la classe.

Un développeur de composant a une vue interne du composant—ses interfaces et classes de mise en oeuvre—alors que celui qui assemble les composants pour construire un nouveau composant ou une application ne dispose que d'une représentation externe (les interfaces) de ces composants.

Un composant peut être mis en oeuvre dans n'importe quel langage. Dans du code Java, vous pouvez faire intervenir des composants EJB, servlets, JSP par exemple.

Pour plus d'informations sur les autres types de composants : EJB, servlets, JSP et ASP.NET, voir *Chapitre 12, Java* à la page 359 and *Chapitre 16, VB .NET* à la page 449.

Si vous commencez à développer un composant avec des classes et des interfaces dans un MOO et que vous souhaitez par la suite les stocker dans une base de données, vous avez la possibilité de créer manuellement une table de correspondance entre objets de sorte que les objets de MOO puissent correspondre aux objets d'un MPD. De même, si vous disposez à la fois d'un MPD et d'un MOO et que ces deux modèles doivent être préservés ; vous pouvez

gérer le lien entre l'environnement orienté objet et la base de données via la *correspondance objet-relationnel*. Grâce à cette correspondance, vous pouvez faire communiquer vos composants et évoluer dans un environnement objet, tout en extrayant des données stockées dans une base de données.

Pour plus d'informations sur les correspondances O/R, voir *Chapitre 19, Modélisation des correspondances objet/relationnel (O/R)* à la page 507.

Création d'un composant

Vous pouvez créer un composant à l'aide d'un Assistant ou à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Composant** dans la Boîte à outils.
- Sélectionnez **Modèle > Composants** pour afficher la boîte de dialogue Liste des composants, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Composant**.
- Sélectionnez **Outils > Créer un composant** pour afficher l'Assistant de création de composant standard.

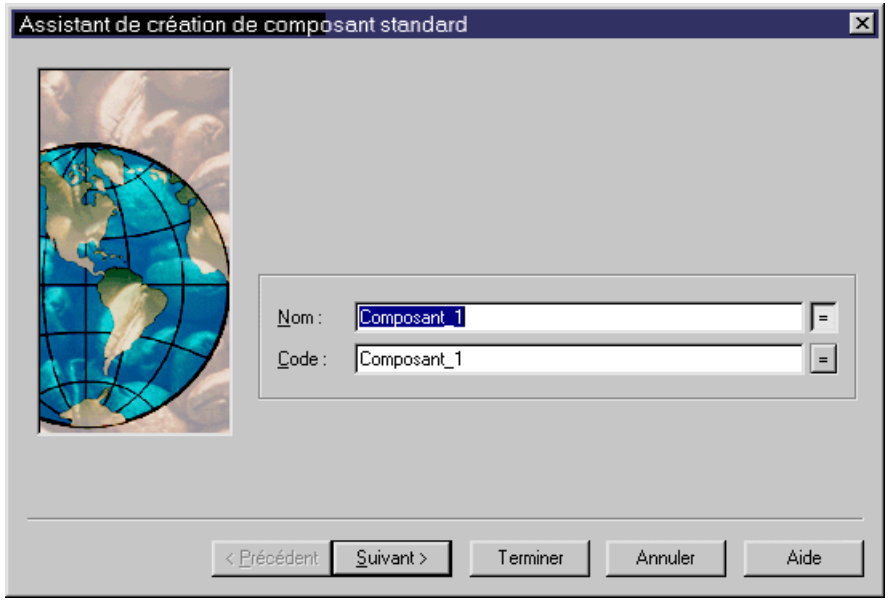
Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Pour plus d'informations sur les autres types de composants, tels que EJB, servlets, JSP et ASP.NET, voir *Chapitre 12, Java* à la page 359 et *Chapitre 16, VB .NET* à la page 449.

Utilisation de l'Assistant de création de composant standard

PowerAMC met à votre disposition un Assistant pour vous aider à créer des composants à partir des classes.

1. Ouvrez un diagramme de classes ou un diagramme de structure composite, puis sélectionner la ou les classes que vous souhaitez inclure dans le nouveau composant.
2. Sélectionnez **Outils > Créer un composant** pour afficher l'Assistant de création de composant standard.



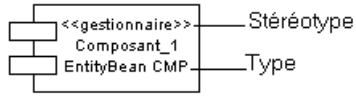
3. Saisissez un nom et un code pour le composant, puis cliquez sur Suivant.
4. Si vous souhaitez que le composant ait un symbole et soit affiché dans un diagramme, cochez la case Créer le symbole dans, puis spécifiez le diagramme dans lequel vous souhaitez voir apparaître ce symbole (vous pouvez choisir de créer un nouveau diagramme). Si vous ne cochez pas cette case, le composant est créé et visible dans l'Explorateur d'objets, mais il est dépourvu de symbole.
5. Si vous souhaitez créer un nouveau diagramme de classes pour y regrouper les classes sélectionnées, cochez la case Créer un diagramme de classes pour les classificateurs de composant.

Propriétés d'un composant

Pour visualiser ou modifier les propriétés d'un composant, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension. Les stéréotypes standard suivants sont disponibles par défaut : • <<Document>> - Fichier générique qui n'est ni un fichier source, ni un exécutable. • <<Executable>> - Fichier de programme qui peut être exécuté sur un ordinateur. • <<File>> - Fichier physique dans le contexte du système développé. • <<Library>> - Fichier de bibliothèque statique ou dynamique. • <<Table>> - Table de base de données. Vous pouvez modifier un stéréotype existant ou créer un nouveau stéréotype dans un langage objet ou dans un fichier d'extension.

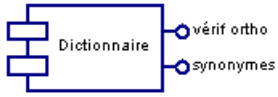
Propriété	Description
Type	Spécifie le type du composant. Vous pouvez choisir un composant standard (si aucune mise en oeuvre particulière n'est définie) ou bien un composant spécifique, comme EJB, JSP, Servlet ou ASP.NET (voir *Chapitre 6, Services Web* à la page 241). Pour afficher le type d'un composant, sélectionnez **Outils > Préférences d'affichage** et sélectionnez l'option Type dans la catégorie Composant.  Lorsque vous changez le type d'un composant après sa création, la modification provoque une conversion d'un type à l'autre : toutes les interfaces et classes concernées ainsi que toutes les dépendances sont automatiquement créées et initialisées. Un tel changement affecte automatiquement certaines feuilles de propriétés, de même que la fonctionnalité de vérification de modèle et la génération de code. Par exemple, si vous convertissez un composant standard en bean d'entité EJB, ce composant génère automatiquement une classe Bean et une classe de clé primaire de l'EJB, de même que des interfaces de composant et interfaces Home. Si vous convertissez un EJB en composant standard, les classes et interfaces de l'EJB sont préservées dans le modèle.
Transaction	Utilisé pour un composant ayant un comportement transactionnel.
Diagramme de classes	Spécifie un diagramme contenant des classes et interfaces liées au composant, qui est automatiquement créé et mis à jour (voir *Création d'un diagramme de classes pour un composant* à la page 229).
Service Web	Indique que le composant est un service Web.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Onglets Interfaces

Chaque composant utilise une ou plusieurs interfaces. Il utilise ou requiert également des interfaces d'autres composants. Ces interfaces sont des points d'entrée et des services visibles qu'un composant met à la disposition des autres composants et classes. Si les dépendances entre composants prennent leur origine dans les interfaces, ces composants peuvent être remplacés par d'autres composants utilisant les mêmes interfaces.

L'onglet **Interfaces** répertorie les interfaces exposées et mises en oeuvre par le composant. Utilisez l'outil **Ajouter des objets** pour ajouter des interfaces existantes ou l'outil **Créer un objet** pour créer de nouvelles interfaces pour le composant.

Les interfaces de composant sont affichées sous forme de cercles liés au bord latéral du composant par un trait horizontal ou vertical :



Le symbole d'une interface est visible si vous avez sélectionné la préférence d'affichage de composant Symboles d'interface **Outils > Préférences d'affichage**. Le symbole d'une interface peut être déplacé autour du symbole de composant, et le lien entre le composant et l'interface peut être étendu.

Si vous utilisez des EJB, certaines des interfaces ont une signification particulière (interface Local, interface Remote, etc.). Pour plus d'informations, voir *Définition d'interfaces et de classes pour les EJB* à la page 375.

Onglet Classes

Un composant utilise généralement une classe de mise en ouvre comme classe principale, tandis que d'autres classes sont utilisées pour mettre en oeuvre les fonctions du composant. En règle générale, un composant est constitué de nombreuses classes internes et de packages de classes, mais il peut être également formé d'une collection de composants plus petits.

L'onglet **Classes** répertorie les classes contenues par le composant. Utilisez l'outil **Ajouter des objets** pour ajouter des classes existantes ou l'outil **Créer un objet** pour créer de nouvelles classes pour le composant.

Les classes ne sont pas visibles dans le diagramme de composants.

Les onglets suivants sont également disponibles :

- Composants - répertorie les composants enfant du composant courant. Vous pouvez créer des composants directement sur cet onglet.
- Opérations - répertorie les opérations contenues dans les interfaces associées au composant. Utilisez le filtre dans la barre d'outils afin de filtrer en fonction d'interfaces particulières.
- Ports - répertorie les ports associés au composant. Vous pouvez créer des ports directement dans cet onglet (voir *Ports (MOO)* à la page 65).
- Parties - répertorie les parties associées au composant. Vous pouvez créer des parties directement dans cet onglet (voir *Parties (MOO)* à la page 63).
- Fichiers - répertorie les fichiers associés au composant. Si des fichiers sont attachés à un composant, ils sont déployés sur le serveur avec le composant (voir *Objets fichier (MOO)* à la page 235).
- Diagrammes associés - répertorie et permet d'associer des diagrammes de modèle associés au composant (voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC*

> Diagrammes, matrices et symboles > Diagrammes > Spécification de diagrammes comme diagrammes associés).

Création d'un diagramme de classes pour un composant

Vous pouvez créer un diagramme de classes pour un composant sélectionné afin de disposer d'une vue d'ensemble des classes et interfaces associées à ce composant. Vous ne pouvez créer qu'un seul diagramme de classes par composant.

La fonctionnalité Créer/Mettre à jour le diagramme de classes, accessible via le menu contextuel du composant, accomplit les tâches suivantes :

- Crée un diagramme de classes s'il n'en existe pas déjà un.
- Associe un diagramme de classes à un composant.
- Ajoute les symboles de toutes les interfaces et classes associées dans le diagramme de classes.
- Complète les liens dans le diagramme.

Cette fonctionnalité permet également de mettre à jour un diagramme de classes après avoir modifié un composant.

La fonctionnalité Ouvrir le diagramme de classes, disponible à partir du menu contextuel du composant, permet d'ouvrir le diagramme de classes correspondant, s'il existe, ou de créer un diagramme de classes par défaut.

Dans le cas des composants EJB, par exemple, la fonctionnalité Ouvrir le diagramme de classes ouvre le diagramme de classes dans lequel la classe Bean du composant est définie.

Si vous supprimez un composant qui est associé à un diagramme de classes, le diagramme de classes est également supprimé. En outre, les symboles de classes et d'interfaces sont supprimés dans le diagramme de classes, mais les classes et interfaces restent présentes en tant qu'objets dans le modèle.

Pointez sur le composant dans le diagramme de composants, cliquez le bouton droit de la souris et sélectionnez Créer/Mettre à jour le diagramme de classes dans le menu contextuel.

Un nouveau diagramme de classes, spécifique au composant, est affiché dans la fenêtre de diagramme et le noeud correspondant s'affiche au-dessous dans l'Explorateur d'objets. Vous pouvez créer d'autres objets pour le composant dans le diagramme de classes.

Remarque : Pour ouvrir le diagramme de classes pour un composant, pointez sur le composant dans le diagramme, cliquez le bouton droit de la souris, puis sélectionnez Ouvrir le diagramme de classe dans le menu contextuel ou maintenez la touche **Ctrl** enfoncée et double-cliquez sur le composant.

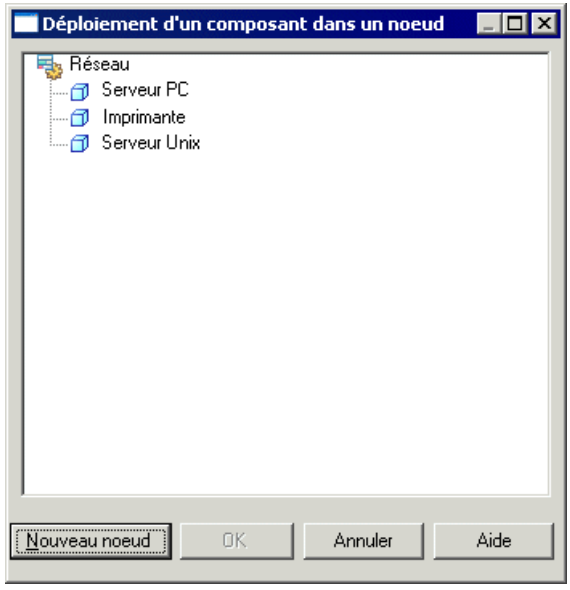
Déploiement d'un composant dans un noeud

Le fait de déployer un composant dans un noeud permet de définir une instance du composant au sein du noeud. Vous pouvez déployer un composant à partir du diagramme de composant ou depuis l'Explorateur d'objets. Une fois le composant déployé, un raccourci vers le

composant et une instance de composant sont créés et placés au sein de son noeud de déploiement.

Vous ne pouvez sélectionner qu'un seul composant à la fois pour le déploiement.

1. Pointez sur le symbole de composant, cliquez le bouton droit de la souris, puis sélectionnez Déployer le composant dans un noeud pour afficher la fenêtre Déploiement d'un composant dans un noeud :



2. Sélectionnez un noeud existant dans lequel déployer le composant, ou bien cliquez sur le bouton Nouveau noeud pour créer un nouveau noeud et y déployer le composant.
3. Cliquez sur OK pour créer une nouvelle instance de composant à l'intérieur du noeud sélectionné.

Noeuds (MOO)

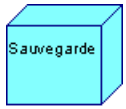
Un noeud est l'élément principal d'un diagramme de déploiement. Il constitue un élément physique qui représente une ressource de traitement, une unité physique concrète ou un emplacement physique de déploiement (ordinateur, imprimante ou tout autre périphérique matériel).

Dans UML, un noeud est défini comme Type ou Instance. Ainsi, vous pouvez définir 'MachineSauvegarde' comme étant un noeud Type, et définir 'Serveur:MachineSauvegarde' comme Instance. Pour simplifier, PowerAMC ne gère qu'un seul élément, appelé noeud, qui représente en fait une instance de noeud. Si vous devez spécifier le type, vous pouvez utiliser un stéréotype, par exemple.

Un noeud peut être créé dans les diagrammes suivants :

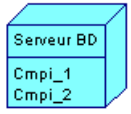
- Diagramme de déploiement

Le symbole d'un noeud est un cube :



Un noeud *ne peut pas* contenir un autre noeud. Il peut en revanche contenir des instances de composant et des objets fichier : les instances de composant logiciels et/ou objets fichier associés sont exécutés au sein des noeuds. Vous pouvez également utiliser des raccourcis de composants.

Vous pouvez ajouter une instance de composant à partir de la feuille de propriétés du noeud. Vous pouvez également afficher la liste des instances de composant dans le noeud du symbole. Pour ce faire, sélectionnez l'option Instances de composant dans les préférences d'affichage relatives aux noeuds.



Vue composite

Vous pouvez ajouter des instances de composant et des objets fichier dans un noeud en les faisant glisser sur le symbole du noeud. Par défaut, les sous-objets sont affichés à l'intérieur du symbole. Pour désactiver l'affichage de ces sous-objets, pointez sur le symbole de noeud, cliquez le bouton droit de la souris, puis sélectionnez **Vue composite > Aucune**. Pour les afficher à nouveau, sélectionnez **Vue composite > Modifiable**.

Création d'un noeud

Vous pouvez créer un noeud à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Noeud** dans la Boîte à outils.
- Sélectionnez **Modèle > Noeuds** pour afficher la boîte de dialogue Liste des noeuds, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Noeud**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'un noeud

Pour visualiser ou modifier les propriétés d'un noeud, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les

onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/ Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Cardinalité	Nombre spécifique d'instances qu'un noeud peut avoir, par exemple : 0...1.
Adresse réseau	Adresse ou nom de machine.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Onglet Instances de composant

L'onglet **Instances de composant** répertorie toutes les instances des composants qui peuvent être exécutées sur le noeud courant (voir *Instances de composant (MOO)* à la page 233). Vous pouvez spécifier des instances de composant directement sur cet onglet et elles seront affichées dans le symbole du noeud.

Diagrammes de noeud

Vous pouvez créer des diagrammes de déploiement avec un noeud afin de visualiser les instances de composant et les objets fichier qu'il contient.

Pour créer un diagramme de noeud, maintenez la touche **Ctrl** enfoncée et double-cliquez sur le symbole du noeud dans le diagramme de déploiement, ou bien pointez sur le noeud dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Diagramme de déploiement**. Le diagramme est crée sous le noeud dans l'Explorateur d'objets et s'affiche dans la fenêtre de diagramme.

Pour ouvrir un diagramme de noeud à partir du symbole du noeud dans un diagramme de déploiement, maintenez la touche **Ctrl** enfoncée et double-cliquez sur le symbole du noeud ou bien pointez sur le symbole du noeud, cliquez le bouton droit de la souris, puis sélectionnez **Ouvrir le diagramme**.

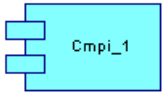
Instances de composant (MOO)

Une instance de composant est une instance d'un composant qui peut être exécuté sur un noeud. Chaque fois qu'un composant est exécuté sur un noeud, une instance de ce composant est créée. L'instance du composant joue un rôle important dans la mesure où elle contient le paramètre de déploiement sur un serveur.

Une instance de composant peut être créée dans les diagrammes suivants :

- Diagramme de déploiement

Le symbole de l'instance de composant est identique au symbole d'un composant dans un diagramme de composants.



La relation entre l'instance de composant et le noeud s'apparente à une composition ; il s'agit d'une relation forte, alors que la relation entre l'objet fichier et le noeud est différente car plusieurs noeuds peuvent utiliser le même fichier en fonction des besoins en terme de déploiement.

Glisser-déposer d'un composant dans un diagramme de déploiement
Vous pouvez faire glisser un composant depuis l'Explorateur d'objets dans un diagramme de déploiement et créer ainsi automatiquement une instance de composant liée au composant.

L'instance de composant qui hérite du composant hérite automatiquement de son type : le type du composant est affiché dans la feuille de propriétés de l'instance de composant.

Déploiement d'un composant dans un noeud depuis le diagramme de composant
Vous pouvez créer une instance de composant à partir d'un composant. Pour ce faire, utilisez la commande Déployer le composant dans un noeud. Cette commande est disponible dans le menu contextuel d'un noeud de composant (l'Explorateur d'objets) ou d'un symbole de composant (diagramme). Cette opération a pour effet de créer une instance de composant et de l'associer à un noeud. Si vous affichez le symbole du noeud dans un diagramme de déploiement, le nom de l'instance de composant est indiqué dans le symbole du noeud auquel il est attaché.

Pour plus d'informations, voir *Déploiement d'un composant dans un noeud* à la page 229.

Création d'une instance de composant

Vous pouvez créer une instance de composant à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Instance de composant** dans la Boîte à outils.

- Sélectionnez **Modèle > Instances de composant** pour afficher la boîte de dialogue Liste des instances de composant, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Instance de composant**.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'une instance de composant

Pour visualiser ou modifier les propriétés d'une instance de composant, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/ Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Cardinalité	Nombre spécifique d'occurrences qu'une instance de composant peut avoir, par exemple : 0...1.
Composant	Composant dont l'instance de composant constitue une instance. Si vous changez le nom de composant dans cette zone, le nom de l'instance de composant est mis à jour dans le modèle.
Type de composant	Zone en lecture seule qui affiche le type du composant d'origine de l'instance de composant.
Service Web	Indique que l'instance de composant est une instance d'un composant de service Web.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Si vous souhaitez dresser la liste de toutes les instances de composant d'un composant, cliquez sur l'onglet Instances de composant dans la page Dépendances de la feuille de propriétés du composant.

Objets fichier (MOO)

Un objet fichier peut être un fichier bitmap utilisé à des fins de documentation, ou bien un fichier contenant du texte utilisé pour le déploiement sur un serveur.

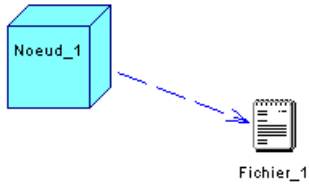
Un objet fichier peut être créé dans les diagrammes suivants :

- Tous les diagrammes

Le symbole d'un objet fichier se présente comme suit :

L'objet fichier peut avoir une fonction particulière dans un diagramme de déploiement, dans lequel il est spécifié comme artefact (en sélectionnant la propriété Artefact) et généré lors du processus de génération.

Lorsque vous souhaitez associer un objet fichier à un noeud, vous pouvez tracer une dépendance entre l'objet fichier et le noeud :



Vous pouvez également maintenir la touche **Ctrl** enfoncée et double-cliquer sur le symbole parent, puis créer l'objet fichier dans le diagramme de noeuds.

Vous pouvez éditer un objet fichier à partir du diagramme de déploiement. Pour ce faire, pointez sur son symbole, cliquez le bouton droit de la souris et sélectionnez **Ouvrir le document** ou **Ouvrir avec >** ***l'éditeur de votre choix***.

Création d'un objet fichier

Vous pouvez créer un objet fichier par glisser-déposer ou à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Fichier** dans la Boîte à outils.
- Sélectionnez **Modèle > Fichiers** pour afficher la boîte de dialogue Liste des fichiers, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Fichier**.

- Faites glisser un fichier depuis votre Explorateur Windows dans le diagramme ou dans l'Explorateur d'objets de PowerAMC.

Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés de l'objet fichier

Pour visualiser ou modifier les propriétés d'un objet fichier, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

Propriété	Description
Nom/Code/ Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Type d'emplacement	Spécifie la nature de l'objet. Vous pouvez choisir l'une des valeurs suivantes : • Fichier incorporé – fichier est stocké dans le modèle et est enregistré lorsque vous enregistrez ce dernier. Si vous changez par la suite le type en Externe, vous serez averti que le contenu existant sera perdu. • Fichier externe – le fichier est stocké dans le système de fichiers Windows, et vous devez spécifier son chemin d'accès dans la zone Emplacement. Si vous changez par la suite le type en Incorporé, vous serez invité à importer le contenu du fichier dans le modèle. • URL– le fichier se trouve sur le web et vous devez spécifier son URL dans la zone Emplacement
Emplacement	[Externe et URL uniquement] Spécifie le chemin d'accès ou l'URL du fichier.
Suffixe	Suffixe du nom de l'objet fichier, qui est utilisé pour l'associer à un éditeur. Le suffixe par défaut est `txt`.
Générer	Spécifie que l'objet fichier est généré lorsque vous générez le contenu du modèle dans un autre modèle.

Propriété	Description
Artefact	Spécifie que l'objet fichier n'est pas un simple élément de documentation, mais fait partie intégrante de l'application. Si un artefact a un suffixe qui est spécifié dans la page Editeurs de la boîte de dialogue Options générales et lié à l'éditeur *<internal>*, un onglet Contenu est affiché dans la feuille de propriétés de l'artefact, qui permet d'éditer le fichier d'artefact dans l'éditeur de texte de PowerAMC.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Associations de noeuds (MOO)

Vous pouvez créer des associations entre les noeuds, on les appelle alors associations de noeuds. Elles sont définies avec un nom de rôle et une multiplicité à chaque extrémité. Une association entre deux noeuds implique que les noeuds communiquent entre eux, par exemple lorsqu'un serveur envoie des données à un serveur de sauvegarde.

Une association de noeuds peut être créée dans les diagrammes suivants :

- Diagramme de déploiement

Le symbole d'une association de noeuds se présente comme suit :

Création d'une association de noeuds

Vous pouvez créer une association de noeuds à partir de la Boîte à outils, de l'Explorateur d'objets ou du menu **Modèle**.

- Utilisez l'outil **Association de noeuds** dans la Boîte à outils.
- Sélectionnez **Modèle > Association de noeuds** pour afficher la boîte de dialogue Liste des associations de noeuds, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Association de noeuds**.

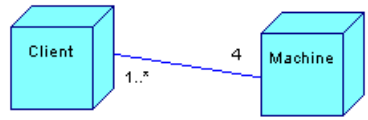
Pour obtenir des informations générales sur la création des objets, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets*.

Propriétés d'une association de noeuds

Pour visualiser ou modifier les propriétés d'une association de noeuds, double-cliquez sur son symbole dans le diagramme ou sur l'entrée correspondante dans l'Explorateur d'objets ou dans

une liste. Les onglets de feuille de propriétés et zones répertoriés ici sont ceux disponibles par défaut, avant toute personnalisation de l'interface par vous ou par un administrateur.

L'onglet **Général** contient les propriétés suivantes :

Propriété	Description
Nom/Code/Commentaire	Identifient l'objet. Le nom doit permettre à des utilisateurs non spécialistes de savoir à quoi sert l'objet, tandis que le code, qui est utilisé afin de générer du code ou des scripts, peut être abrégé, et ne doit normalement contenir aucun espace. Vous pouvez également spécifier un commentaire afin de fournir des informations plus détaillées sur l'objet. Par défaut, le code est généré à partir du nom en appliquant les conventions de dénomination spécifiées dans les options du modèle. Pour supprimer la synchronisation du nom et du code, cliquez sur le bouton = en regard de la zone **Code**.
Stéréotype	Etend la sémantique de l'objet au-delà de la définition UML. Vous pouvez saisir un stéréotype directement dans cette zone, ou bien ajouter des stéréotypes dans la liste en les spécifiant dans un fichier d'extension.
Rôle A	Une des extrémités de l'association de noeuds. Chaque rôle peut avoir un nom et une cardinalité, et être navigable. Vous pouvez utiliser les outils à droite de la liste pour créer un objet, parcourir l'arborescence des objets disponibles ou afficher les propriétés de l'objet sélectionné.
Noeud A	Nom du noeud à une extrémité de l'association de noeuds. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Multiplicité A	La multiplicité indique les nombre maximum et minimum d'instances de l'association de noeuds. Vous pouvez choisir l'une des valeurs suivantes : • 0..1 – Aucune ou une. • 0..* – D'aucune à l'infini. • 1..1 – Exactement une. • 1..* – D'une à l'infini. • * – D'aucune à l'infini. Par exemple, dans un environnement informatique, vous pouvez avoir 100 clients et 100 machines mais une contrainte peut spécifier qu'une machine ne peut accepter plus de 4 clients simultanément. Ainsi, le nombre d'instances est fixé à 4 dans la zone Multiplicité du côté de la machine : 
Rôle B	Nom du rôle situé à une extrémité de l'association de noeuds. Chaque rôle peut avoir un nom et une cardinalité et peut être navigable.

Propriété	Description
Noeud B	Nom du noeud situé à une extrémité de l'association de noeuds. Vous pouvez utiliser les outils à droite de la liste pour créer un objet, parcourir l'arborescence des objets disponibles ou afficher les propriétés de l'objet sélectionné.
Multiplicité B	Nombres minimal et maximal d'instances de l'association de noeuds. Pour plus de détails voir la description de Multiplicité A, ci-dessus.
Mots clés	Permet de grouper de façon informelle des objets. Pour saisir plusieurs mots clés, séparez-les de virgules.

Chapitre 6 Services Web

Un service Web est un service proposé via le Web. Le principe de fonctionnement d'un service Web est le suivant : une application envoie une requête à un service situé à une adresse URL particulière. Cette requête peut utiliser le protocole SOAP via HTTP. Le service reçoit la requête, traite cette requête et renvoie une réponse. Un service de cotation boursière dans lequel une requête demande le cours d'une action et dans lequel la réponse serait le cours de l'action constitue un exemple du fonctionnement d'un service Web.

Dans un MOO, vous concevez un service Web comme un *composant* (un EJB, un servlet ou un composant standard) qui inclut une classe de mise en oeuvre d'un service Web.

Lorsque vous travaillez sur les services Web dans un MOO, vous utilisez les diagrammes de classe, de composants et de déploiement. Ces diagrammes vous permettent d'effectuer les opérations suivantes :

- Créer un nouveau composant de service Web
- Procéder au reverse engineering de WSDL pour créer un composant de service Web
- Parcourir UDDI pour chercher un WSDL
- Générer un WSDL pour une définition de composant de service Web
- Générer du code de services Web côté serveur pour Java (AXIS, JAXM, JAX-RPC, Web Services for J2EE) ou pour .NET (C# et VB .NET)
- Générer un proxy client pour Java ou .NET
- Procéder au reverse engineering de Java et .NET

Vous devez disposer d'un compilateur Java, C# ou Visual Basic .NET pour utiliser les services Web.

Pour Java, vous devez également disposer d'un outil WSDL-vers-Java et d'un outil Java-vers-WSDL pour générer le code de proxy Java ainsi que le code côté serveur compatible JAX-RPC. Les outils WSDL-vers-Java et Java-vers-WSDL sont utilisés pour le fichier d'extension WSDL for Java. Par exemple, le WSDP (Web Service Developer Pack) fournit un outil XRPCC, Apache AXIS fournit un outil wsdl2java et un outil java2wsdl (qui peuvent être téléchargés depuis l'adresse suivante : *http://www.oracle.com/technetwork/java/index.html*). Apache AXIS peut être téléchargé depuis l'adresse suivante : *http://ws.apache.org/axis*.

Pour pouvoir générer du code de proxy client pour .NET, vous allez devoir utiliser le fichier WSDL.exe inclus dans Visual Studio .NET et déclarer le chemin d'accès vers le fichier WSDL.exe dans la boîte de dialogue Options générale (**Outills > Options générales**) lorsque vous créez les variables d'environnement WSDL.

Définition des outils de services Web

Un service Web est une interface qui décrit une collection d'opérations qui sont accessibles sur le réseau via des messages SOAP.

L'utilisation de services Web via Internet ou au sein d'un intranet d'entreprise s'apparente à la recherche d'un site Web : soit vous saisissez l'adresse du site (URL), soit vous utilisez un moteur de recherche pour localiser ce dernier. Soit vous connaissez l'adresse de ce service Web ou celle de son interface (WSDL, par exemple), soit vous devez chercher le service en lançant une requête sur le registre des services Web. La spécification UDDI définit un mécanisme standard pour la publication et la localisation des sociétés et des services qu'elles proposent.

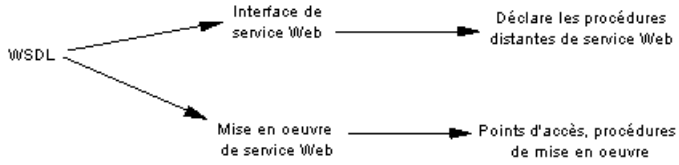
WSDL est un langage qui décrit ce dont un service Web est capable et comment un client peut localiser et invoquer un tel service. Le document Web Services Description Language (WSDL) 1.1, disponible à l'adresse *http://www.w3.org/TR/wsdl*, décrit les documents WSDL comme suit :

"Un document WSDL définit des services comme des collections de points d'accès réseau, également appelés ports. Dans WSDL, la définition abstraite des points d'accès et des messages est séparée de leur mis en oeuvre concrète dans le réseau. Cette procédure vous permet de réutiliser les définitions abstraites : les *messages* sont des descriptions abstraites des données échangées, et les *types de port* sont des collections abstraites d'opérations. Le protocole concret et les spécifications de format de données pour un type de port particulier constituent un lien réutilisable. Vous définissez un port en associant une adresse réseau avec un lien réutilisable, et une collection de ports définit un service. En outre, un document WSDL utilise les éléments suivants dans la définition des services réseau :

- *Types* : un conteneur pour les définitions de type de données utilisant un système de types (par exemple, XSD)
- *Message* : une définition abstraite typée de la donnée communiquée
- *Opération* : une description abstraite d'une action prise en charge par le service
- *Type de port* : un jeu d'opérations abstrait pris en charge par un ou plusieurs points d'accès
- *Lien (binding)* : protocole concret et spécification de format de données pour un type de port particulier
- *Port* : un point d'arrêt unique défini comme une combinaison d'un lien et d'une adresse réseau
- *Service* : collection de points d'accès liés"

Interface et mise en oeuvre

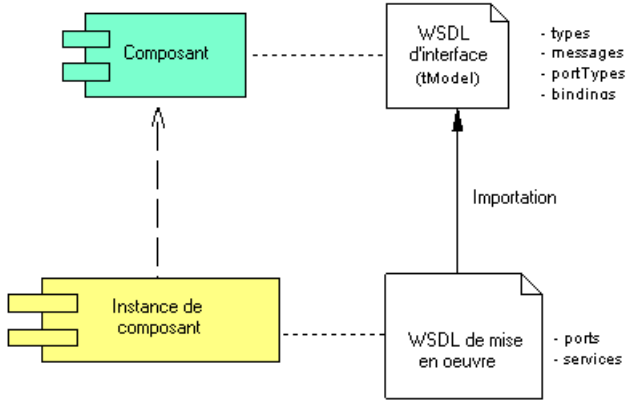
WSDL est utilisé pour définir l'interface du service Web, la mise en oeuvre du service Web, ou les deux. Il est donc possible d'utiliser deux fichiers WSDL, l'un pour l'interface et l'autre pour la mise en oeuvre.



Dans un *WSDL d'interface*, vous déclarez les procédures qui permettent de créer un service Web.

Dans un *WSDL de mise en oeuvre*, vous définissez les modalités de mise en oeuvre de ces procédures via les services et ports (URL de points d'accès).

Dans un MOO, un WSDL d'interface est associé à un composant, et un WSDL de mise en oeuvre est associé à une instance de composant. Vous pouvez enregistrer les deux fichiers WSDL dans le modèle.



Pour obtenir des informations détaillées sur WSDL, reportez-vous à la page suivante : *http://www.w3.org/2002/ws/desc*.

SOAP (Simple Object Access Protocol)

SOAP est un protocole basé sur XML pour l'échange d'informations dans un environnement distribué. Il représente le mécanisme d'invocation au sein d'une architecture de services Web. WSDL permet à un utilisateur de mieux appréhender quel format du message SOAP doit être envoyé pour appeler le service et quel est le format de message de retour attendu.

UDDI (Universal Description Discovery and Integration)

UDDI est un registre international de sociétés au format XML. Ce registre répertorie tous les services Web disponibles sur Internet et gère leurs adresses.

Dans UDDI, une organisation ou une société, appelée *businessEntity* publie généralement un WSDL pour décrire ses interfaces de service Web comme *tModel*. Une autre société peut la

mettre en oeuvre, puis publier dans l'UDDI les éléments suivants qui indiquent comment invoquer le service Web :

- La société, appelée *businessEntity*
- Le service, appelé *businessService*
- Les points d'accès, appelés *bindingTemplate*
- La spécification, appelée *tModel*

Services Web pris en charge

Dans le cas des services Web mis en oeuvre à l'aide de .NET, PowerAMC génère des fichiers .asmx pour C# ou VB .NET, des fichiers WSDL et des proxies client dans C# ou VB .NET.

Dans le cas des services Web mis en oeuvre à l'aide de Java, PowerAMC permet d'utiliser l'un des types de modèle suivants : AXIS, JAXM, JAX-RPC et Web Services for J2EE (Stateless Session Bean).

Définition des cibles de service Web

Les langages de MOO Java et .NET (C# et Visual Basic .NET) prennent en charge les services Web.

En général, le WSDL peut être généré par le serveur sur lequel le service Web est déployé. Le WSDL généré par le serveur contient donc à la fois la définition de l'interface et celle de la mise en oeuvre.

Lorsque vous travaillez avec Java, C# ou VB.NET dans un MOO, un fichier d'extension est automatiquement attaché au modèle. Il est utilisé pour compléter la définition de ces langages dans le contexte des services Web

Définition d'un composant de service Web

Un service Web est représenté comme un composant que vous pouvez afficher dans un diagramme de composants. Dans un diagramme de composants, vous pouvez afficher le code d'interface et de mise en oeuvre d'un service Web, vous pouvez également déployer les composants de service Web dans des noeuds si vous souhaitez décrire le déploiement des composants sur des serveurs.

Pour plus d'informations sur le déploiement des composants, voir *Déploiement d'un composant dans un noeud* à la page 229.

Un composant peut être une interface de service Web ou un type de mise en oeuvre de service Web. Vous devez cocher la case Service Web dans la feuille de propriétés du composant pour déclarer ce dernier comme service Web.

Les types de service Web suivants sont pris en charge pour le langage Java :

- *Java Web Service* : expose une classe Java ayant le suffixe .jws comme un service Web en utilisant Apache Axis.
- *Axis RPC* : expose une classe Java comme service Web à l'aide du modèle Apache Axis RP.
- *Axis EJB* : expose un bean de session sans état (Stateless Session Bean) comme service Web à l'aide du modèle Apache Axis EJB.
- *JAX-RPC* : utilise une classe Java et une interface pour mettre en oeuvre le modèle JAX-RPC.
- *JAXM* : expose un Servlet comme service Web en utilisant JAXM.
- *Web Service for J2EE* : expose un bean de session sans état (Stateless Session Bean) comme service Web en utilisant le modèle Web Service for J2EE (JSR109).

Propriétés d'un composant de service Web

Les feuilles de propriétés de composant de service Web contiennent toutes les propriétés d'un composant standard, avec des onglets supplémentaires.

Onglet Service Web

L'onglet Composant de service Web inclut les propriétés suivantes

Propriété	Description
Classe de service Web	Spécifie le nom de classe de service Web. Utilisez les outils à droite de cette liste pour créer ou sélectionner, ou pour afficher les propriétés de la classe sélectionnée. Si le composant de service Web est un bean de session sans état (Stateless Session Bean), la classe de service Web est également la classe bean
Espace de noms de l'application	Spécifie l'espace de noms de l'application, qui est utilisé pour générer l'URL pour le service Web sur le serveur. Par défaut, il utilise le code du package du composant ou le code du modèle, vous pouvez changer cette valeur.
URL WSDL	Spécifie où le WSDL est publié sur le Web.
Type de service Web	Spécifie le type de service Web. Interface est un composant qui définit l'interface de service uniquement. Implementation fait référence à un composant qui met en oeuvre l'interface du service.
Utiliser WSDL externe	Spécifie que le WSDL est publié sur une URL particulière et que le WSDL d'origine est préservé. Lorsque vous importez un WSDL, cette option est sélectionnée par défaut.

Onglet WSDL

L'onglet WSDL inclut les propriétés suivantes

Propriété	Description
Espace de noms cible	Spécifie une URL liée à une méthode qui assure l'unicité du service Web et évite les conflits avec d'autres services Web du même nom. Par défaut, il s'agit de `http://tempuri.org` pour .NET et de `urn:%Code%Interface` pour Java, mais nous vous recommandons de changer cette valeur pour préserver l'unicité du nom de service
Préfixe	Spécifie de définir un préfixe d'espace de noms cible
Style de codage	Spécifie le type de codage, soit SOAP (soap:xxx) soit XML-Schema (xsd:xxx) pour le WSDL
Commentaire	Fournit une description du fichier WSDL.
Editeur WSDL	Permet d'éditer le contenu du WSDL. Si vous effectuez des modifications, le bouton Défini par l'utilisateur est enfoncé.

Onglet Schéma WSDL

L'onglet Schéma WSDL inclut une zone de texte qui contient certaines définitions de schéma partagé provenant du schéma WSDL. Cette partie du schéma définit les types de données utilisés par les messages d'entrée, de sortie et d'erreur.

L'autre partie du schéma est définie au sein des différentes méthodes Web (opérations) sous forme de type de données de message d'entrée SOAP, de types de données, de message de sortie SOAP et/ou de types de données d'erreur SOAP (voir *Définition des types de données SOAP du schéma WSDL* à la page 260).

Onglet UDDI/Attributs étendus

Ces onglets incluent les propriétés suivantes :

Nom	Description
Style de binding SOAP	Définit le style de binding SOAP. Il peut s'agir d'un document ou d'un rpc Nom dans le script : SoapBindingStyle
URI de transport du SOAP binding	Définit l'URI de transport du SOAP binding Nom dans le script : SoapBindingTransport
Espace de noms XML schema pour les types de données	Définit l'espace de noms XML schema pour les types de données dans le WSDL Nom dans le script : SoapBodyNamespace
Nom de société	Stocke le nom de la société trouvée dans un registre UDDI Nom dans le script : BusinessName

Nom	Description
Description de société	Stocke la description de la société proposant le service Web trouvée dans un registre UDDI. Nom dans le script : BusinessDescription
Clé de société	Stocke la clé de la société trouvée dans un registre UDDI Nom dans le script : BusinessKey
Espaces de noms définis dans les sections de définition	Stocke des espaces de noms supplémentaires qui ne sont pas automatiquement identifiés par PowerAMC Nom dans le script : Namespaces
Nom du service	Stocke le nom du service trouvé dans un registre UDDI Nom dans le script : ServiceName
Description du service	Stocke la description du service Web trouvée dans un registre UDDI. Nom dans le script : ServiceDescription
Clé du service	Stocke la clé du service trouvée dans un registre UDDI Nom dans le script : ServiceKey
Nom tModel	Stocke le nom du tModel trouvé dans un registre UDDI Nom dans le script : tModelName
Clé tModel	Stocke la clé du tModel trouvée dans un registre UDDI Nom dans le script : tModelKey
URL tModel	Stocke l'URL du tModel trouvée dans un registre UDDI. Elle permet d'extraire le WSDL Nom dans le script : tModelURL
URL de l'opérateur UDDI utilisé pour la recherche	Stocke l'URL de l'opérateur de registre UDDI utilisée pour trouver le WSDL Nom dans le script : UDDIOperatorURL

Vous pouvez utiliser l'onglet Aperçu du composant pour afficher un aperçu du code que vous allez générer pour le composant de service Web.

Création d'un service Web à l'aide de l'Assistant

Vous pouvez créer un service Web à l'aide de l'*Assistant* qui vous guide au travers des différentes étapes de création du composant. L'Assistant est appelé depuis un *diagramme de classes* et n'est disponible que si vous utilisez la famille de langage Java ou .NET.

Vous pouvez créer un service Web sans sélectionner de classe, ou bien utiliser l'approche standard qui consiste à sélectionner une classe existante, puis à lancer l'Assistant à partir du menu contextuel de la classe.

Vous pouvez également créer plusieurs services Web du même type en sélectionnant plusieurs classes à la fois. L'Assistant créera automatiquement un service Web par classe. Les classes que vous avez sélectionnées dans le diagramme de classes deviennent des classes de service Web, elles sont renommées pour correspondre aux conventions de dénomination et sont liées au nouveau composant de service Web.

Remarque : Vous devez créer le composant de service Web au sein d'un package de sorte que ce package agisse comme un espace de noms.

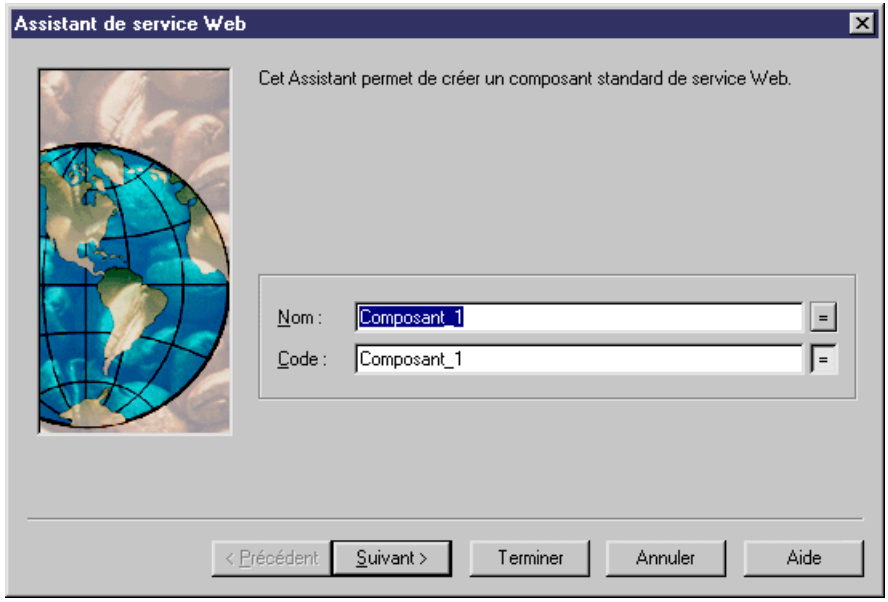
L'Assistant de création des services Web permet de définir les paramètres suivants :

Propriété	Description
Nom	Nom du composant de service Web
Code	Code du composant de service Web
Type de service Web	Interface ou Implementation. Interface fait référence à un composant qui définit l'interface de service Web uniquement. Implementation fait référence à un composant qui met en oeuvre une interface de service Web
Type de composant	Le type de composant dépend du type de service Web. Vous pouvez sélectionner le type dans une liste de protocoles Interface ou Implementation
Classe de mise en oeuvre de service Web	Définit la classe qui est utilisée pour mettre en oeuvre le service Web
Créer un symbole dans	Crée un symbole de composant dans le diagramme spécifié en regard de la case à cocher. S'il existe déjà un diagramme de composants, vous pouvez le sélectionner dans la liste. Vous pouvez également afficher la feuille de propriétés du diagramme sélectionné en cliquant sur l'outil Propriétés

Propriété	Description
Créer un diagramme de classes pour les classificateurs de composant	Disponible uniquement pour les beans de session sans état et les servlets. Crée un diagramme de classes avec un symbole pour chaque classe et interface. Si vous avez sélectionné des classes et interfaces avant de démarrer l'Assistant, elles sont utilisées pour créer le composant. Cette option permet d'afficher ces classes et interfaces dans un diagramme

1. Sélectionnez **Outils > Créer un composant de service Web** depuis un diagramme de classes.

   La boîte de dialogue Assistant de service Web s'affiche.



2. Sélectionnez un nom et un code pour le composant de service Web, puis cliquez sur Suivant.

   Remarque : Si vous avez sélectionné plusieurs classes avant de démarrer l'Assistant, certaines étapes ne s'affichent pas car les noms sont créés par défaut à partir du nom des classes sélectionnées.

   Lorsque vous créez un service Web à l'aide de l'Assistant, vous devez sélectionner le type de service Web et le type de composant qu'il utilise comme source. Le tableau suivant établit la correspondance entre les types de composant disponibles et les types de service Web dans Java :

Type de composant	Interface de service Java	Mise en oeuvre de service Java	Utilisé avec
Standard		—	Classe Java
Axis RPC	—		Classe Java
Axis EJB	—		Bean de session sans état d'EJB
Java Web Service (JWS)	—		Classe Java avec le suffixe .jws
JAXM	—		Servlet Java
JAX-RPC	—		Classe Java
Web Service for J2EE	—		Bean de session sans état d'EJB

= admis

— = non admis

3. Sélectionnez un type de service Web et un type de composant, puis cliquez sur Suivant.
4. Sélectionnez une classe de mise en oeuvre de service Web, puis cliquez sur Suivant.
5. A la fin de l'Assistant, vous devez définir la création des symboles.

Une fois que vous avez fini d'utiliser l'Assistant, les actions suivantes sont exécutées :

- Un composant de service Web est créé
- Une classe de mise en oeuvre de service Web est créée et visible dans l'Explorateur d'objets. Elle est nommée d'après la classe d'origine si vous avez sélectionné une classe avant de démarrer l'Assistant. Si vous n'avez pas sélectionné de classe avant de démarrer l'Assistant, le nom de la classe prend comme préfixe celui du composant par défaut d'origine afin de préserver la cohérence
- Une opération par défaut est créée avec 'indicateur "Web method"
- Selon le type de composant, les interfaces requises associées au composant sont ajoutées

Création d'un service Web à partir du diagramme de composants

Vous pouvez également créer un service Web à partir du diagramme de composants.

1. Cliquez sur l'outil **Composant** dans la Boîte à outils, puis cliquez dans le diagramme pour créer un composant.
2. Cliquez sur l'outil **Pointeur** ou cliquez le bouton droit de la souris pour libérer l'outil **Composant**.
3. Double-cliquez sur le symbole du composant pour afficher sa feuille de propriétés.
4. Cochez la case **Service Web** sur l'onglet **Général**.

5. Cliquez sur **OK**.

Définitions de types de données pour WSDL

WSDL utilise XML Schema pour définir les types de données pour les structures de message.

Correspondances de types de données WSDL

Pour générer du WSDL, il est nécessaire d'établir des correspondances entre les types de données Java ou .NET et les types XML.

.NET et les types XML. Dans un MOO, il existe trois tables de correspondances de types de donnée définies dans le fichier d'extension WSDL.

- WSDL2Local - convertit les types de données WSDL en types Java ou .NET
- Local2SOAP - convertit les types de données WSDL Java ou .NET en types SOAP pour le codage SOAP
- Local2XSD - convertit les types de données Java ou .NET en types XML Schema pour le codage XML Schema

Sélection des types de données WSDL

Si vous devez utiliser un type de données spécifique, vous pouvez sélectionner le type de données WSDL pour le type de résultat d'une opération ou pour le type de paramètre.

Vous pouvez sélectionner un type de données WSDL dans la liste des feuilles de propriétés d'opération et de paramètre. Cette zone inclut les types de données de base pour le codage XML Schema ou pour le codage SOAP.

Vous pouvez également cliquer sur le bouton Propriétés en regard du type de données WSDL afin d'ouvrir l'onglet Schéma WSDL de la feuille de propriétés du composant. Cette page affiche le contenu du schéma WSDL.

Tant que le type de données WSDL n'est pas modifié manuellement, il est synchronisé avec le type de données Java ou .NET. Vous pouvez utiliser l'onglet Aperçu dans la feuille de propriétés de la classe pour vérifier le code à tout moment.

1. Dans l'onglet Méthode Web de la feuille de propriétés de l'opération, spécifiez un nouveau type de données dans la zone Type de données WSDL.

   ou

   Dans la feuille de propriétés de paramètre, spécifiez un nouveau type de données dans la zone Type de données WSDL.
2. Cliquez sur Appliquer.

Déclaration des types de données dans le WSDL

Les classes utilisées comme types de données sont déclarées comme Complex Types dans la section <types> du WSDL.

Propriétés d'une classe de mise en oeuvre de service Web

Un service Web requiert une classe de mise en oeuvre. Une classe de mise en oeuvre ne peut être associée qu'à un composant de service Web. Dans les langages .NET, la classe de mise en oeuvre peut être générée dans le fichier .asmx ou en dehors de celui-ci. Dans Java, une classe de service Web peut avoir une classe de *sérialisation* et une classe de *désérialisation*.

Les feuilles de propriétés des classes de mise en oeuvre de service Web comportent les propriétés supplémentaires suivantes sur l'onglet Détails :

Propriété	Description
Composant de service Web	Spécifie le composant de service Web lié à la classe de mise en oeuvre de service Web. Cliquez sur l'outil Propriétés à droite de cette zone pour afficher la feuille de propriétés du composant.
Classe de sérialisation	Spécifie la classe utilisée pour convertir un objet au format texte ou binaire. Cliquez sur l'outil Propriétés à droite de cette zone pour afficher la feuille de propriétés de la classe.
Classe de désérialisation	Spécifie la classe utilisée pour convertir un format texte, XML ou binaire en objet. Cliquez sur l'outil Propriétés à droite de cette zone pour afficher la feuille de propriétés de la classe.

Cliquez sur l'onglet Aperçu pour afficher un aperçu des éléments suivants :

- Classe de mise en oeuvre du service Web dans Java
- Le fichier .ASMX dans .NET
- Le WSDL d'interface généré à partir du composant et de la classe de mise en oeuvre dans Java et .NET (en lecture seule)

Gestion de méthodes de services Web

Vous pouvez définir une ou plusieurs méthodes comme faisant partie d'un service Web. Dans PowerAMC, vous utilisez des opérations pour créer des méthodes de service Web.

Création d'une méthode de service Web

Une méthode de service Web est une opération dont la propriété Méthode de service Web est sélectionnée.

Une méthode de service Web peut appeler d'autres méthodes qui ne sont pas exposées comme méthodes de service Web. Dans ce cas, ces méthodes internes ne sont pas générées dans le WSDL.

Les méthodes de service Web peuvent appartenir à une classe de mise en oeuvre de composant ou à des interfaces de composant.

Les interfaces liées à un composant de service Web peuvent être utilisées pour concevoir différents groupes de méthodes représentant différents types de port.

Une interface de composant contenant au moins une opération ayant la propriété Méthode de service Web est considérée comme un type de port.

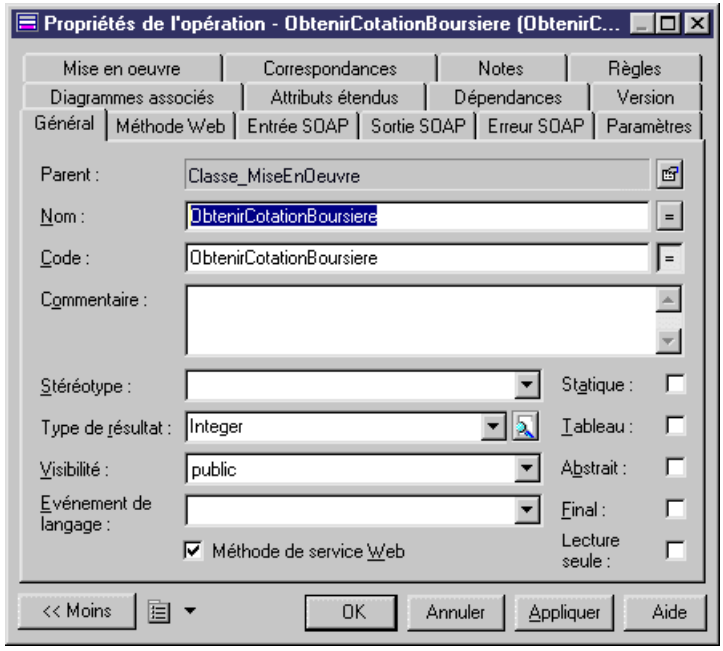
Les méthodes d'interface utilisent les mêmes attributs étendus que les méthodes de classes pour la personnalisation du WSDL, comme expliqué dans la section *Définition des attributs étendus d'une méthode de service Web* à la page 259.

Trois attributs étendus permettent de décider du type de port à générer : SOAPPortType, HttpGetPortType et HttpPostPortType. Si une méthode de service Web est créée dans une interface, seul l'attribut SOAPPortType est défini comme True. Cette méthode est automatiquement ajoutée dans la classe de mise en oeuvre du composant.

Dans le cas d'un composant de service Web JAXM, la mise en oeuvre du service Web doit être effectuée dans la méthode onMessage(). Pour pouvoir générer le WSDL approprié, vous devez déclarer une méthode de service Web sans mise en oeuvre afin de définir le message d'entrée SOAP.

Pour plus d'informations sur la mise en oeuvre des méthodes, voir *Mise en oeuvre d'une méthode de service Web dans Java* à la page 255 et *Mise en oeuvre d'une méthode de service Web dans .NET* à la page 259.

1. Affichez la feuille de propriétés de la classe ou de l'interface de service Web.
2. Cliquez sur l'onglet **Opérations**, puis sur l'outil **Insérer une ligne** pour créer une nouvelle opération.
3. Cliquez sur **Appliquer**, puis sur l'outil **Propriétés** pour afficher la feuille de propriétés de l'opération.
4. Cochez la case Méthode de service Web dans l'onglet **Général**.



5. Cliquez sur **OK**.

Propriétés d'une méthode de service Web

Lorsque vous cochez la case Méthode de service Web dans la feuille de propriétés de l'opération, des onglets supplémentaires s'affichent.

L'onglet Méthode de service Web de la feuille de propriétés d'opération inclut les propriétés suivantes :

Propriété	Description
Classe Soap extension	Utilisé pour .NET. A la création de la classe, de nouvelles fonctions par défaut sont ajoutées. Dans .NET, une méthode peut avoir une classe SOAP extension pour gérer la sérialisation et la désérialisation pour la méthode et contrôler les sécurité des autres fonctionnalités SOAP extension. Utilisez les outils à droite de la liste pour créer ou sélectionner un objet, ou pour afficher les propriétés de l'objet sélectionné.
Type de données WSDL	Type de données pour le type de résultat. Inclut les types de données de base du langage objet ainsi que les types de données complexes du schéma WSDL. Vous pouvez cliquer sur l'outil Propriétés en regard de cette zone pour afficher l'onglet Schéma WSDL de la feuille de propriétés du composant. Cette page affiche le contenu du schéma WSDL

Les onglets suivants sont également affichés :

- Entrée SOAP - définit le nom et le schéma du message d'entrée SOAP.
- Sortie SOAP - définit le nom et le schéma du message de sortie SOAP.
- Erreur SOAP - définit le nom et le schéma du message d'erreur SOAP.

Mise en oeuvre d'une méthode de service Web dans Java

Pour mettre en oeuvre une méthode de service Web, vous devez définir les éléments suivants :

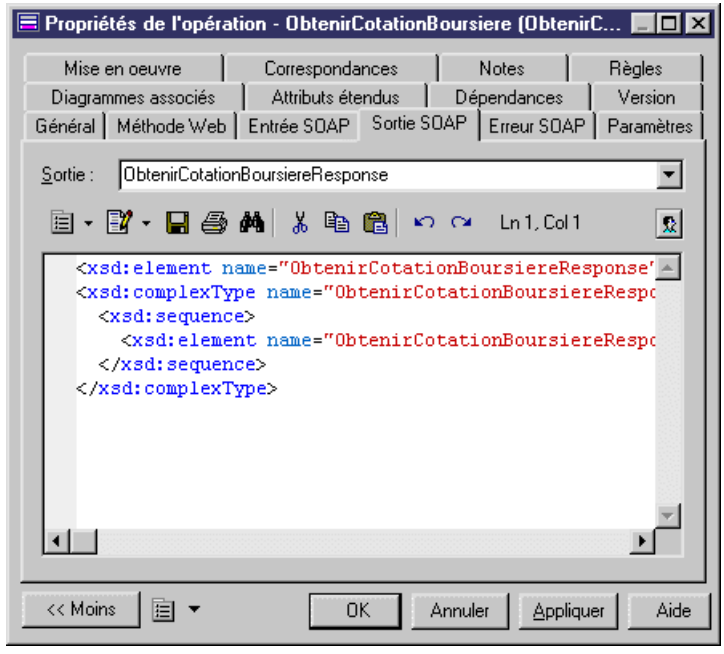
- Le type de résultat de l'opération
- Les paramètres de l'opération
- La mise en oeuvre de l'opération

Définition du type de résultat d'une opération

Pour définir le type de résultat d'une opération, vous devez spécifier les éléments suivants :

- *Le type de résultat pour la méthode Java* : dans l'onglet Général de la feuille de propriétés de l'opération. Si la valeur de résultat dans Java est une classe Java ou un tableau de classes Java, PowerAMC va générer un message de sortie basé sur la structure de classe de résultat.
- *Le type de message de sortie pour WSDL* : dans le cas des valeurs de résultat simples, le message de sortie pour le WSDL peut être défini dans l'onglet Méthode Web de la feuille de propriétés de l'opération.

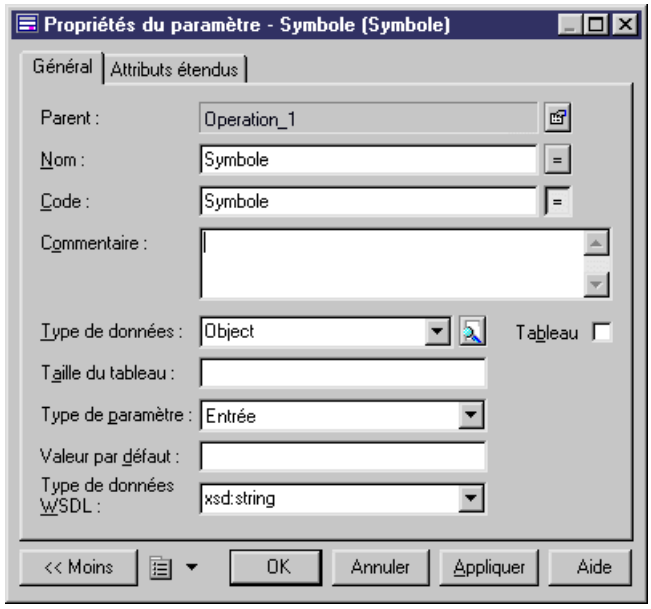
Dans le cas de types de données plus complexes, vous pouvez définir manuellement le message de sortie en utilisant l'onglet Sortie SOAP de la feuille de propriétés de l'opération.



Définition des paramètres d'une opération

Pour définir les paramètres d'une opération, vous devez spécifier :

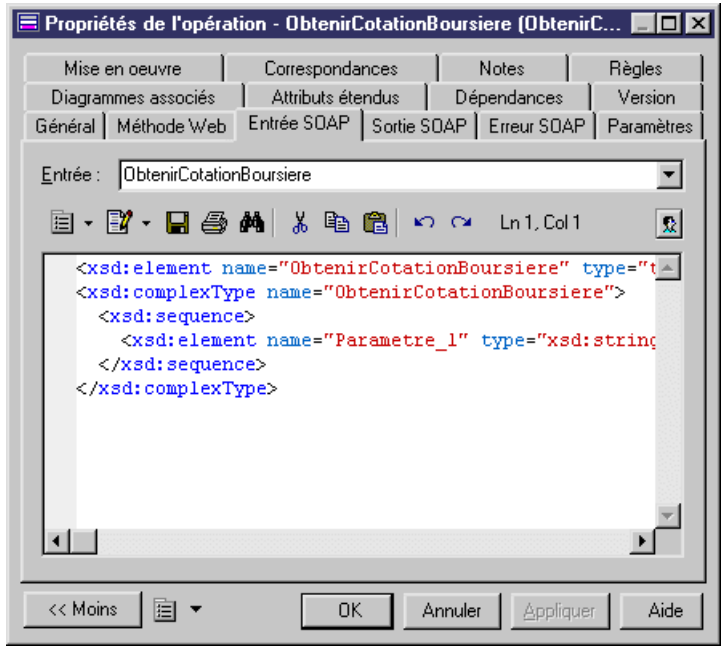
- Le *type de paramètre* pour la méthode Java en utilisant la liste Type de données dans la feuille de propriétés du paramètre
- Le *type de message d'entrée* pour WSDL en utilisant la liste Type de données WSDL dans la feuille de propriétés du paramètre. Cette liste affiche les types de données de base à partir du langage objet et des types de données complexes du schéma WSDL. Vous pouvez cliquer sur l'outil Propriétés en regard de cette zone pour afficher l'onglet Schéma WSDL de la feuille de propriétés du composant. Cette page affiche le contenu du schéma WSDL



Dans le cas des valeurs de paramètre simples, le type de message d'entrée est généré depuis les types WSDL des paramètres. Si un paramètre est une classe ou un tableau de classes, PowerAMC génère uniquement un lien SOAP.

Pour plus d'informations sur SOAP, voir *http://www.w3.org/2000/xp/Group*.

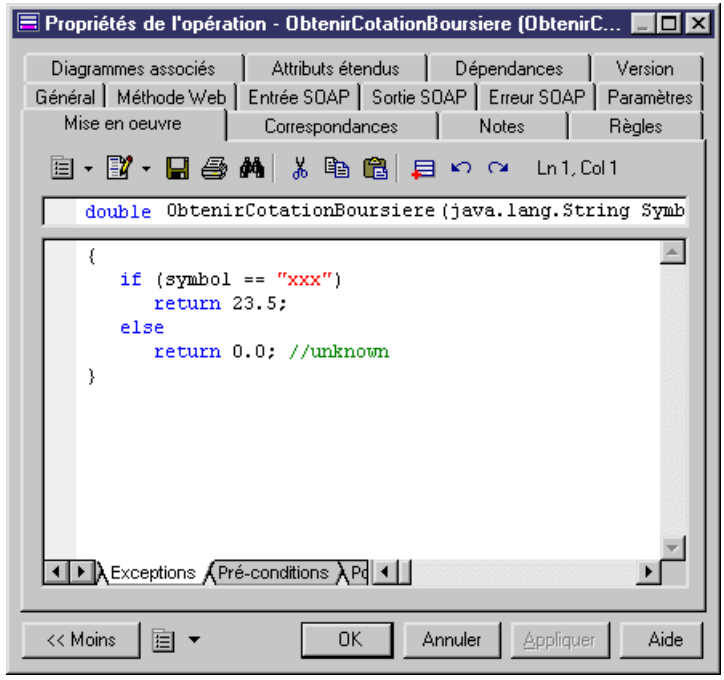
Dans le cas des types de paramètres complexes et des types de message d'entrée, vous pouvez définir manuellement le schéma de message d'entrée en utilisant l'onglet Entrée SOAP dans la feuille de propriétés de l'opération.



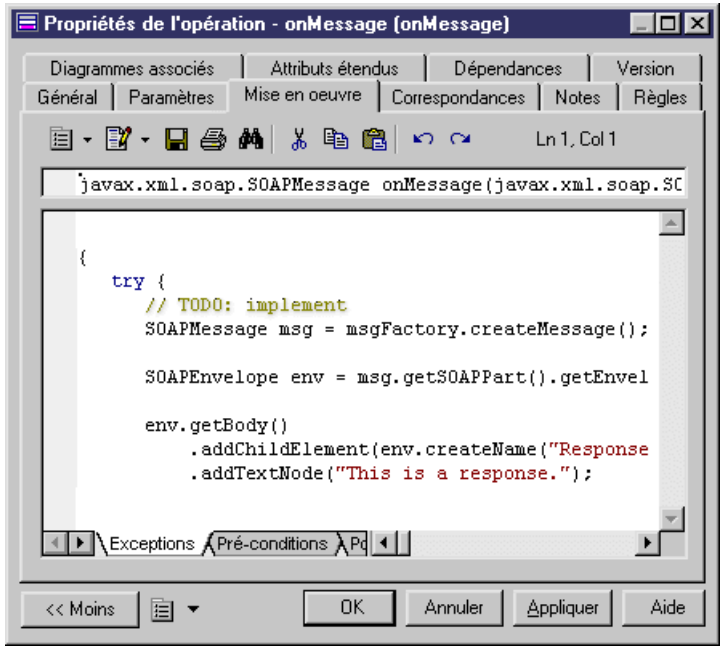
Mise en oeuvre de l'opération

Vous mettez en oeuvre une méthode de service Web comme une méthode Java normale.

L'exemple suivant montre la mise en oeuvre de la méthode de service Web ObtenirCotationBoursière.



Dans le cas d'une méthode de service Web contenue dans un service Web JAXM, vous devez mettre en oeuvre la méthode onMessage(). Vous devez pour ce faire traiter le message d'entrée SOAP, générer un message de sortie SOAP et renvoyer le message de sortie.



Mise en oeuvre d'une méthode de service Web dans .NET

Pour mettre en oeuvre une méthode de service Web dans .NET, vous devez définir des paramètres d'entrée et un type de résultat.

Ces procédures sont décrites dans *Mise en oeuvre d'une méthode de service Web dans Java* à la page 255.

Par défaut, PowerAMC génère le classe de service Web C# ou VB .NET dans le fichier .asmx. Si vous souhaitez générer la classe C# ou VB .NET dans un fichier distinct et utiliser le mode CodeBehind pour le fichier .asmx, vous devez modifier une option de génération : dans l'onglet Options de la boîte de dialogue de génération, définissez "false" comme valeur pour Génération de code de service Web C# dans un fichier .asmx.

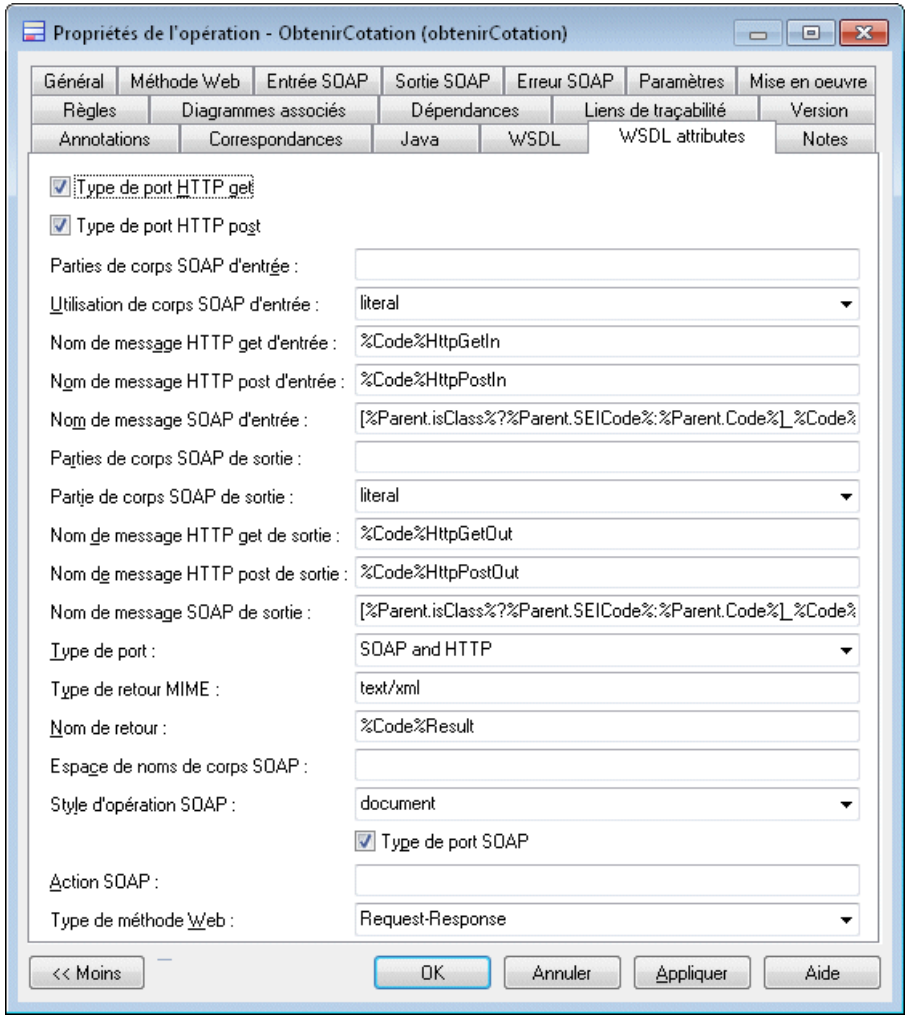
Vous pouvez afficher un aperçu du code du fichier .asmx et du code du WSDL dans l'onglet Aperçu de la feuille de propriétés de classe.

Définition des attributs étendus d'une méthode de service Web

Vous pouvez personnaliser la génération WSDL en utilisant des attributs étendus. Les attributs étendus permettent de modifier le nom du message d'entrée, le nom du message de sortie, le style d'opération SOAP, les types de port à générer, l'action SOAP, le type de méthode Web, etc.

1. Affichez la feuille de propriétés de l'opération.

2. Cliquez sur l'onglet WSDL attributes et modifiez les valeurs attributs étendus.



Définition des types de données SOAP du schéma WSDL

Chaque méthode Web a un type de données d'*entrée*, de *sortie* et d'*erreur* que vous devez définir. Un type de données a un nom et une définition de schéma.

Dans le cas des services Web récupérés par reverse engineering, les types de données d'entrée, de sortie et d'erreur sont définis avec la valeur trouvée dans le schéma WSDL récupéré. Les types de données qui ne sont pas associés avec une entrée, une sortie ou une erreur sont considérés comme des définitions de schéma partagé et sont disponibles dans le composant. Ils sont affichés dans l'onglet Schéma WSDL de la feuille de propriétés de composant.

Dans le cas d'opérations nouvellement créées, les types de données d'entrée et de sortie sont automatiquement définis avec une valeur par défaut, et sont synchronisés avec des changements de paramètre. Le nom et le schéma par défaut du type de données sont définis dans la définition étendue de modèle WSDL et peuvent être personnalisés. Toutefois, une fois modifié, un type de données devient type de données utilisateur et ne peut plus être synchronisé. Les types de données d'erreur sont toujours des types de données utilisateur.

Vous pouvez réutiliser un type de données défini dans une autre opération. Une vérification est disponible sur les composants pour vérifier qu'aucun type de données n'a des noms différents dans un même composant (voir *Chapitre 9, Vérification d' un MOO* à la page 303).

Lorsque vous procédez à la *génération*, le schéma de WSDL est composé des définitions de schémas partagés standard provenant du composant, et d'une combinaison calculée de toutes les définitions d'entrée, de sortie et d'erreur SOAP provenant des opérations.

Vous pouvez saisir les noms des types de données d'entrée, de sortie et d'erreur SOAP dans les pages appropriées de la feuille de propriétés d'opération. Chaque page contient une zone de texte dans laquelle vous pouvez éditer la définition du type de données provenant du schéma de WSDL.

Définition d'une instance de composant de service Web

Le diagramme de déploiement est utilisé avec les services Web pour modéliser leur déploiement. Ce diagramme est très utile si vous souhaitez déployer un service Web sur un ou plusieurs serveurs, et si vous souhaitez savoir quel service Web sera déployé à un emplacement particulier, les diagrammes de déploiement permettant en effet d'afficher des adresses réseau, des points d'accès et des types d'accès.

Une instance de composant définit les ports, le type d'accès et les points d'accès qui constituent l'URL complète permettant d'invoquer le service Web.

Lorsque la case Service Web est cochée dans la feuille de propriétés de l'instance de composant, cette instance de composant est une instance de composant de service Web. Une instance de composant qui hérite d'un composant hérite également de son type : le type du composant est affiché dans la feuille de propriétés de l'instance de composant.

Lorsque vous cochez la case Service Web, une page Service Web et une page WSDL sont automatiquement ajoutées dans la feuille de propriétés de l'instance de composant.

Pour plus d'informations sur le diagramme de déploiement, voir *Chapitre 5, Diagrammes de mise en oeuvre* à la page 219.

Onglet Service Web de l'instance de composant

La page Service Web de la feuille de propriétés de l'instance de composant inclut les propriétés suivantes :

Propriété	Description
URL du point d'accès	Affiche l'URL complète permettant d'invoquer le service. Il s'agit d'une valeur calculée qui utilise l'adresse réseau située dans le noeud. Vous pouvez également saisir votre propre URL en utilisant l'outil Défini par l'utilisateur situé en regard de la zone. Une fois que vous cliquez sur cet outil, vous pouvez remplacer l'URL indiquée
URL WSDL	Indique où le WSDL doit être publié sur le Web. Vous pouvez également saisir votre propre URL en utilisant l'outil Défini par l'utilisateur situé en regard de la zone. Une fois que vous cliquez sur cet outil, vous pouvez remplacer l'URL indiquée
Utiliser WSDL externe	Indique que le WSDL est publié sur une URL particulière

Lorsqu'un WSDL n'est pas défini par l'utilisateur, il est régénéré à chaque fois et peut être affiché dans l'onglet Aperçu de la feuille de propriétés de l'instance de composant. Un WSDL externe a un texte de WSDL défini par l'utilisateur.

Exemples d'URL du point d'accès

Voici des exemples de la syntaxe utilisée dans .NET et Java :

Pour .NET, la syntaxe par défaut est la suivante :

```
accesstype://machine_networkaddress:port/application_namespace/
webservice_code.asmx
```

Par exemple : http://doc.sybase.com:8080/WebService1/StockQuote.asmx.

Pour Java, la syntaxe par défaut est la suivante :

```
accesstype://machine_networkaddress:port/application_namespace/
webservice_code
```

Par exemple : http://doc.sybase.com/WebService1/StockQuote.

Les attributs calculés AccessType et PortNumber pour le générateur de code & VBScript sont calculés à partir de l'URL de point d'accès. Par exemple : http, https.

Exemples d'URL WSDL

Pour .NET, la syntaxe par défaut est la suivante :

```
accesstype://machine_networkaddress:port/application_namespace/
Webservice_code.asmx?WSDL
```

Pour Java, la syntaxe par défaut est la suivante :

```
accesstype://machine_networkaddress:port/application_namespace/
wsdl_file
```

Onglet WSDL de l'instance de composant

La page WSDL de la feuille de propriétés de l'instance de composant inclut les propriétés suivantes :

Propriété	Description
Espace de noms cible	URL liée à une méthode qui vérifie l'unicité du service Web et évite les conflits avec d'autres services Web de même nom. Par défaut, la valeur est : http://tempuri.org/ pour .NET, et urn:%Component.targetNamespace% pour Java
Importer le WSDL d'interface	Lorsque cette option est sélectionnée, signifie que le WSDL de mise en oeuvre importe le WSDL d'interface existant
Commentaire	Utilisé comme description du fichier WSDL lors de la génération WSDL et de la publication de WSDL dans UDDI
Editeur WSDL	Vous pouvez également utiliser un éditeur de texte situé sous la zone Commentaire pour afficher le contenu du WSDL. Lorsque vous cliquez sur l'outil Défini par l'utilisateur vous spécifiez le contenu de cette zone comme défini par l'utilisateur. Une fois que vous cliquez sur cet outil, vous pouvez remplacer le contenu de la zone

Remarque : Vous pouvez afficher directement certaines pages de la feuille de propriétés d'une instance de composant. Pour ce faire, pointez sur un symbole d'instance de composant dans le diagramme, cliquez le bouton droit de la souris, puis sélectionnez la commande appropriée dans le menu contextuel.

Utilisation des propriétés de noeud

La feuille de propriétés de noeud inclut la propriété suivante, spécifique aux services Web :

Propriété	Description
Adresse réseau	Adresse ou nom de machine. Par exemple : doc.sybase.com, ou 123.456.78.9

Puisqu'une machine peut être utilisée pour plusieurs services et que chaque service peut avoir un type d'accès, un numéro de port et un chemin différents, l'adresse réseau de la machine n'est utilisée que comme valeur par défaut. Vous pouvez redéfinir à tout moment l'URL effective de chaque instance de composant dans la feuille de propriétés de cette dernière.

Génération de services Web pour Java

Vous pouvez générer des classes de mise en oeuvre, interfaces, descripteur de déploiement, JAR, WAR, ou EAR côté client ou côté serveur en utilisant la commande Générer du code langage objet accessible dans le menu Langage.

La génération de code pour les services Web *côté serveur* implique les opérations suivantes :

- Générer des classes de mise en oeuvre et des interfaces de services Web (classe Java, bean de session sans état, Servlet, etc.)
- Générer un descripteur de déploiement de services Web pour Java en utilisant la spécification JSR109. Le descripteur de déploiement est un fichier XML qui doit être inclus dans chaque archive WAR, il s'appelle WEB.xml par convention et contient des informations nécessaires au déploiement d'un service Web
- Générer des fichiers de WSDL d'interface et de WSDL de mise en oeuvre (les fichiers WSDL peuvent être générés séparément car ils peuvent être utilisés pour des applications UDDI ou des applications client)

En règle générale, lorsqu'un service Web est déployé, le serveur est capable de générer un WSDL de mise en oeuvre.

La génération de code pour les services Web *côté client* consiste à générer des classes de proxy.

PowerAMC prend en charge les types de service Web Java JAXM, JAX-RPC, Web Service for J2EE (JSR 109), AXIS RPC, EJB et Java Web Service (JWS).

Génération de services Web JAXM

JAXM est une API Java pour messagerie XML qui fournit un moyen standard d'envoyer des documents XML via Internet depuis la plateforme Java.

Si le type de mise en oeuvre de service Web est JAXM, PowerAMC utilise le modèle JAXM pour la mise en oeuvre. Les composants de service Web JAXM permettent de gérer simplement des formats de message complexes.

La classe Java JAXM utilise la méthode onMessage() pour obtenir le message d'entrée SOAP et renvoyer le message de sortie SOAP. Pour pouvoir générer le WSDL approprié, vous devez définir une méthode de service Web ayant le nom, le format de message d'entrée et le format de sortie appropriés, mais sans mise en oeuvre. La méthode onMessage() ne doit pas être définie comme une méthode de service Web.

Pour pouvoir utiliser JAXM, vous pouvez utiliser le Java Web Services Developer Pack (JWSDP) 1.1 ou version supérieure (disponible à l'adresse *http://www.oracle.com/technetwork/java/index.html*) ou un conteneur de servlet ou serveur J2EE prenant en charge JAXM.

Pour compiler des composants de service Web JAXM, vous devez avoir les fichiers jaxm-api.jar, jaxp-api.jar et saaj-api.jar dans le chemin défini par la variable d'environnement CLASSPATH. Vous devez également définir une variable JAVACLASSPATH dans PowerAMC pour définir le classpath spécifique à PowerAMC, auquel cas la variable JAVACLASSPATH remplace la variable d'environnement CLASSPATH.

Vous avez besoin d'un serveur d'application ou d'un conteneur Servlet qui prenne en charge JAXM. JWSDP est fourni avec Apache Tomcat qui prend en charge JAXM.

1. Sélectionnez **Langage > Générer du code Java** pour afficher la boîte de dialogue Génération.
2. Spécifiez le répertoire dans lequel les fichiers doivent être générés, et indiquez si vous souhaitez effectuer une vérification de modèle (voir *Chapitre 9, Vérification d'un MOO* à la page 303).
3. Sur l'onglet **Tâches**, sélectionnez la commande Java : Conditionnement de l'application J2EE dans un fichier EAR. Cette commande va créer un fichier .WAR et un fichier .EAR.
4. Cliquez sur **OK** pour lancer la génération.

Génération de services Web JAX-RPC

JAX-RPC est une API Java pour le RPC (protocole d'appel de procédure distante) XML. Elle facilite le RPC via Internet en permettant à des paramètres au format XML d'être passés à des services distants et en permettant le renvoi de valeurs au format XML.

Si le type de mise en oeuvre du service Web est JAX-RPC, PowerAMC utilise le modèle JAX-RPC pour la mise en oeuvre. JAX-RPC définit un appel de type RPC pour les services Web, mais est limité aux formats de message simples. Vous pouvez utiliser une mise en correspondance objets/XML très complexe.

L'utilisation du modèle JAX-RPC implique d'effectuer les opérations suivantes :

- Générer le code de la classe et de l'interface Java du service Web
- Compiler la classe et d'interface Java du service Web
- Exécuter un outil JAX-RPC pour générer les artefacts côté serveur et le proxy côté client afin de gérer le service Web
- Placer tout le code compilé, le WSDL et le descripteur de déploiement dans un fichier .WAR
- Déployer le fichier .WAR sur un serveur qui prend en charge JAX-RPC

Pour pouvoir utiliser JAX-RPC, vous pouvez utiliser le Java Web Services Developer Pack (JWSDP) 1.1 ou version supérieure (disponible à l'adresse *http://www.oracle.com/technetwork/java/index.html*) ou un autre serveur d'application prenant en charge le modèle JAX-RPC.

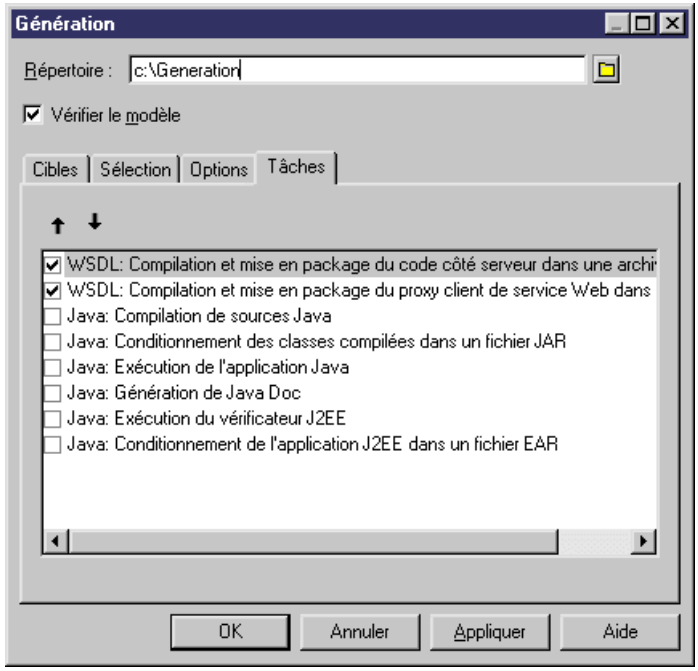
Pour générer le code côté serveur et le proxy client pour JAX-RPC, si vous utilisez JWSDP, vous pouvez utiliser l'outil wscompile.bat. Pour les autres mises en oeuvre compatibles JAX-RPC, veuillez vous reporter à la documentation. Pour appeler l'outil wscompile.bat à partir de

PowerAMC, vous devez définir une variable d'environnement WSCOMPILE dans la catégorie Variables située dans la boîte de dialogue Options générales (**Outils > Options générales**). La variable WSCOMPILE doit indiquer le chemin d'accès du fichier wscompile.bat. Pour pouvoir exécuter wscompile.bat, le fichier jaxrpc-api.jar doit se trouver sur le chemin de la variable d'environnement CLASSPATH. Vous pouvez également définir une variable JAVACLASSPATH dans PowerAMC pour définir le classpath spécifique à PowerAMC, auquel cas la variable JAVACLASSPATH remplace la variable d'environnement CLASSPATH.

Pour déployer des composants de service Web JAX-RPC, vous aurez besoin d'un serveur d'application ou d'un conteneur de Servlet qui prend en charge JAX-RPC. JWSDP est fourni avec Apache Tomcat qui prend en charge JAX-RPC

1. Sélectionnez **Langage > Générer du code Java** pour afficher la boîte de dialogue Génération.
2. Spécifiez le répertoire dans lequel les fichiers doivent être générés, et indiquez si vous souhaitez effectuer une vérification de modèle (voir *Chapitre 9, Vérification d'un MOO* à la page 303).
3. Sur l'onglet **Tâches**, pour générer du code côté serveur, sélectionnez la commande WSDL: Compilation et mise en package du code côté serveur dans une archive. Pour générer un proxy client, sélectionnez la commande WSDL: Compilation et mise en package du proxy client de service Web dans une archive.

   Ces commandes vont compiler les classes Java générées par PowerAMC, exécuter l'outil WSCOMPILE et créer un fichier .WAR.



4. Cliquez sur **OK** pour lancer la génération.

Génération de beans de session sans état de service Web

PowerAMC prend en charge la spécification Web Services for J2EE qui définit le modèle de programmation et l'architecture d'exécution pour la mise en oeuvre des services Web.

Dans Java, les services Web peuvent être mis en oeuvre soit via les points d'arrêt JAX-RPC, soit via les composants de bean de session sans état d'EJB. Ces deux types de mise en oeuvre exposent leurs méthodes Web via une SEI (Service Endpoint Interface).

Les points d'arrêt JAX-RPC sont considérés comme des composants de service Web, ils sont représentés sous forme de *servlets* dans le MOO et sont placés dans un fichier WAR, alors que les *beans de sessions sans état* d'EJB sont placés dans un fichier JAR d'EJB. Dans les deux cas, les fichiers WSDL, et les descripteurs de déploiement requis doivent être inclus dans les répertoires WEB-INF ou META-INF. Vous pouvez vous référer aux chapitres 5 et 7 de la spécification J2EE pour les services Web afin de trouver des informations complémentaires.

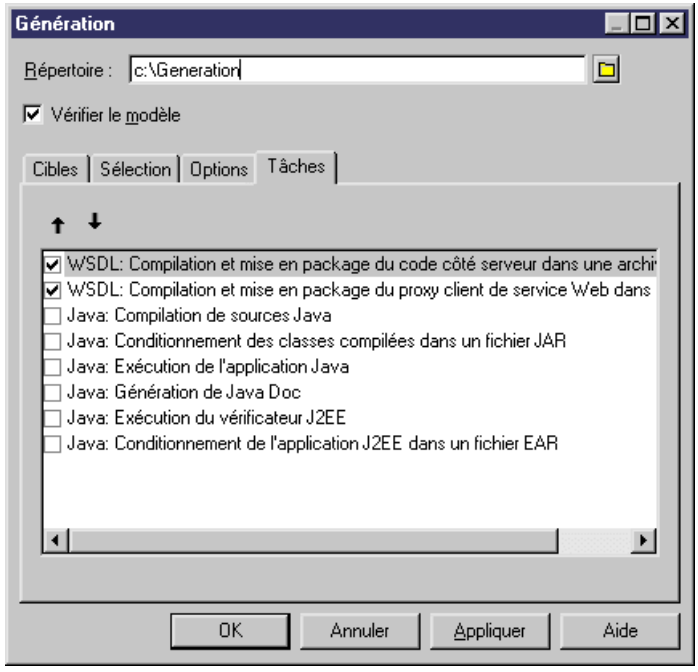
Si le type de mise en oeuvre de service Web est EJB-Stateless Session Bean, PowerAMC utilise la spécification Web Services for J2EE pour la mise en oeuvre.

Pour développer un bean de session sans état sous forme de service Web, vous procédez comme pour JAX-RPC : vous utilisez une classe Bean au lieu d'une classe Java normale. Tout comme dans JAX-RPC, cette technique ne s'applique qu'aux formats de message simples.

Pour plus d'informations sur JAX-RPC, voir *Génération de services Web JAX-RPC* à la page 265.

1. Sélectionnez **Langage > Générer du code Java** pour afficher la boîte de dialogue Génération.
2. Spécifiez le répertoire dans lequel les fichiers doivent être générés, et indiquez si vous souhaitez effectuer une vérification de modèle (voir *Chapitre 9, Vérification d'un MOO* à la page 303).
3. Dans l'onglet Tâches, pour générer du code côté serveur, sélectionnez la commande WSDL : Compilation et mise en package du code côté serveur dans une archive. Pour générer le proxy côté client, sélectionnez la commande WSDL : Compilation et mise en package du proxy client de service Web dans une archive.

   Ces commandes vont compiler les classes Java générées par PowerAMC, exécuter l'outil WSCOMPILE et créer un fichier .WAR.



4. Cliquez sur **OK** pour lancer la génération.

Génération de services Web AXIS RPC

Si le type de mise en oeuvre de service Web est AXIS RPC, PowerAMC utilise une classe Java pour la mise en oeuvre et Apache Axis pour le déploiement.

Le type de fournisseur pris en charge est Java:RPC. Les styles de fournisseur pris en charge sont "RPC", "document" et "wrapped". Si le fournisseur est <Default>, PowerAMC va automatiquement sélectionner le meilleur style de fournisseur. Pour sélectionner le style de

fournisseur, vous pouvez changer l'attribut étendu AxisProviderStyle du composant de service Web.

Pour personnaliser la génération de descripteur de déploiement Axis, vous pouvez changer les attributs étendus spécifiques à Axis dans la feuille de propriétés du composant de service Web.

Un fichier deploy.wsdd et un fichier undeploy.wsdd sont générés à partir du modèle ou du package qui contient des composants de service Web. Un seul fichier deploy.wsdd et un seul fichier undeploy.wsdd sont générés pour tous les composants de service Web du modèle ou package.

1. Sélectionnez **Langage > Générer du code Java** pour afficher la boîte de dialogue Génération.
2. Spécifiez le répertoire dans lequel les fichiers doivent être générés, et indiquez si vous souhaitez effectuer une vérification de modèle (voir *Chapitre 9, Vérification d'un MOO* à la page 303).
3. [facultatif] Cliquez sur l'onglet **Sélection** et spécifiez les objets à partir desquels vous souhaitez générer. Par défaut, tous les objets sont générés.
4. [facultatif] Cliquez sur l'onglet **Options** et sélectionnez les options de génération appropriées :
5. [facultatif] Cliquez sur l'onglet **Tâches** et spécifiez les tâches de génération à effectuer :
6. Cliquez sur **OK** pour lancer la génération.

   Une fois la génération terminée, la boîte de dialogue Fichiers générés s'affiche et répertorie les fichiers générés dans le répertoire spécifié. Sélectionnez un fichier dans liste, puis cliquez sur **Editer** pour l'ouvrir dans votre éditeur associé, ou bien cliquez sur **Fermer** pour quitter la boîte de dialogue.

Génération de services Web d'EJB AXIS

Si le type de mise en oeuvre de service Web est AXIS EJB, PowerAMC utilise un bean de session sans état pour la mise en oeuvre, un serveur d'applications pour le déploiement d'EJB et Apache Axis pour l'exposition de l'EJB comme service Web.

Pour personnaliser la génération du descripteur de déploiement Axis, vous pouvez changer plusieurs attributs étendus spécifiques à Axis dans la feuille de propriétés de composant de service Web.

Un fichier deploy.wsdd et un fichier undeploy.wsdd sont générés à partir du modèle ou du package qui contient des composants de service Web. Un seul fichier deploy.wsdd et un seul fichier undeploy.wsdd sont générés pour tous les composants de service Web du modèle ou package.

Pour exposer un bean de session sans état comme service Web à l'aide d'Axis, vous devez effectuer les opérations suivantes :

- Générer du code d'EJB
- Compiler et packager l'EJB
- Déployer l'EJB sur un serveur J2EE
- Exposer l'EJB comme un service Web à l'aide d'Axis

1. Sélectionnez **Langage > Générer du code Java** pour afficher la boîte de dialogue Génération.
2. Spécifiez le répertoire dans lequel les fichiers doivent être générés, et indiquez si vous souhaitez effectuer une vérification de modèle (voir *Chapitre 9, Vérification d'un MOO* à la page 303).
3. Dans l'onglet Options, vous pouvez modifier les options de génération et de déploiement.
4. Dans l'onglet Tâches, vous pouvez sélectionner les commandes dans l'ordre suivant : Conditionnement de l'application J2EE dans un fichier EAR, Déploiement d'un fichier EAR d'application J2EE, Expositions des EJB sous forme de services Web.

Génération de services Web Java (JWS)

Si le type de mise en oeuvre de service Web est Java Web Service (JWS), PowerAMC utilise une classe Java pour la mise en oeuvre. La classe Java aura alors le suffixe .jws.

Pour déployer la classe Java, vous pouvez simplement copier le fichier .jws sur le serveur qui prend en charge le format de service Web Java. Par exemple, dans le cas de Apache Axis, vous pouvez copier le fichier .jws dans le répertoire webapps\axis.

Test des services Web pour Java

Vous pouvez tester un service Web Java de l'une des façons suivantes :

- Envoyer un message SOAP. Vous pouvez écrire un programme Java pour envoyer un message SOAP à un service Web et traiter le message SOAP renvoyé à l'aide de l'API SAAJ
- Utiliser l'appel dynamique (Dynamic Invocation). Vous pouvez utiliser la méthode d'appel dynamique définie par JAX-RPC
- Utiliser un proxy dynamique. Vous pouvez utiliser la méthode Dynamic Proxy définie par JAX-RPC
- Utiliser le proxy client. Vous pouvez utiliser un client proxy pour appeler plus facilement un service Web. Si vous utilisez le JWSDP, vous pouvez utiliser l'outil wscompile.bat pour générer un proxy client. Si vous utilisez Apache Axis, vous pouvez utiliser la classe Java java.org.apache.axis.wsdl.WSDL2Java

Génération des services Web pour .NET

Les éléments suivants sont générés dans les langages .NET (C# et VB.NET) :

- Une classe de mise en oeuvre (C# ou VB.NET) avec une superclasse spéciale et une propriété WebMethod pour les méthodes. Si vous désactivez l'option Génération de code de service Web C# dans un fichier .asmx, une classe C# ou VB .NET sera également générée pour chaque service Web
- Un fichier .ASMX

Remarque
Il n'est pas nécessaire de définir la superclasse (également appelée WebService) pour les classes de service Web ; si la superclasse WebService n'est pas définie, le générateur de code l'ajoute dans le code.

Lorsque vous générez du code côté serveur, vous pouvez utiliser les options par défaut et les tâches suivantes pour vous aider à démarrer automatiquement la génération de caractéristiques prédéfinies.

Définition des options de génération de services Web dans .NET

Vous pouvez sélectionner des options de génération de services Web disponibles pour les langages .NET en sélectionnant la commande Générer du code langage objet dans le menu Langage.

Options de génération C#
Les options suivantes sont disponibles depuis l'onglet Options de la boîte de dialogue de génération pour C# :

Option	Description
Génération de code de service Web C# dans un fichier .asmx	Génère le code C# dans le fichier .ASMX

Options de génération VB.NET
Les options suivantes sont disponibles dans l'onglet Options de la boîte de dialogue de génération pour VB.NET :

Option	Description
Génération de code de service Web dans un fichier .asmx	Génère le code Visual Basic.NET dans le fichier .ASMX

Définition des tâches de génération de service Web dans .NET

Vous pouvez sélectionner des tâches de génération de service Web disponibles pour les langages .NET en utilisant la commande **Langage > Générer du code langage objet**.

Les tâches suivantes sont disponibles sur l'onglet Tâches de la boîte de dialogue de génération pour C# et VB.NET :

Option	Description
Compilation de fichiers source Visual Basic .NET	Compile le code généré
Génération de code de proxy de service Web	Génère la classe de proxy de service Web pour une instance de composant de service Web. Vous devez définir une instance de composant pour l'URL de déploiement de service Web
Ouverture de la solution dans Visual Studio .NET	Si vous avez sélectionné l'option Génération de fichiers de projet Visual Studio .NET, cette tâche permet d'ouvrir la solution dans l'environnement de développement .NET de Visual Studio

Génération de services Web dans .NET

Le déploiement d'un service Web est la phase qui consiste à copier la classe de mise en oeuvre générée et le fichier .ASMX dans le dossier virtuel du serveur Web.

Le fichier .ASMX est un fichier ASP.NET, il contient le code de la classe de service Web C# ou VB.NET.

1. Sélectionnez **Langage > Générer du code C#** ou **Générer du code VB .NET**.
2. Sélectionnez le répertoire de génération. Vous pouvez sélectionner un répertoire Microsoft Internet Information Server (IIS) pour la génération, par exemple, C:\Inetpub \wwwroot\StockQuote. Si vous avez défini votre service Web au sein d'un package, vous pouvez générer le code des services Web dans le répertoire C:\Inetpub\wwwroot. Un sous-répertoire est créé pour chaque package.
3. Définissez l'option Génération de code de service Web C# dans un fichier .asmx comme False sur l'onglet **Options** si vous souhaitez générer la classe C# ou VB .NET hors d'un fichier .asmx.
4. Sélectionnez la commande Compilation des fichiers source C# ou Compilation de fichiers source Visual Basic .NET sur l'onglet **Tâches** si vous générez la classe de service Web C# ou VB .NET en dehors d'un fichier .asmx.
5. Cliquez sur **OK** pour lancer la génération.

   Le processus de génération de code crée un sous-répertoire sous wwwroot en utilisant le nom du package, et crée un fichier <NomServiceWeb>.ASMX dans ce sous-répertoire.

Génération d'une classe de proxy .NET pour un service Web

PowerAMC peut également générer une classe de proxy client pour simplifier l'appel du service Web. Pour générer la classe de proxy client, PowerAMC utilise le programme wsdl.exe fourni avec Visual Studio .NET. Vous devez définir une variable WSDL pour indiquer où le programme wsdl.exe est situé.

Définition de la variable WSDL

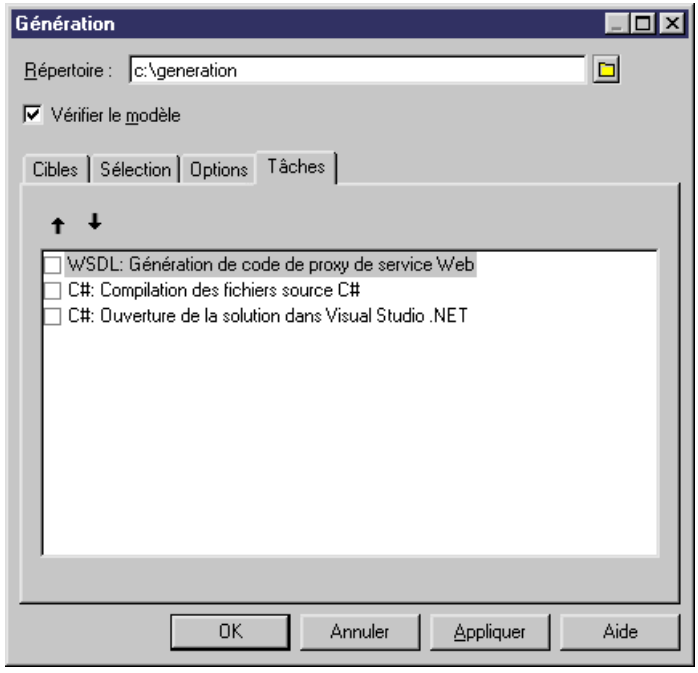
Pour définir une variable WSDL :

1. Sélectionnez **Outils > Options générales**.
2. Sélectionnez la catégorie Variables.
3. Ajoutez une variable WSDL dans la colonne Nom.
4. Spécifiez le chemin d'accès complet du fichier wsdl.exe dans la colonne Valeur.
5. Cliquez sur OK.

Génération des classes de proxy client

Pour générer les classes de proxy client :

1. Sélectionnez **Languge > Générer du code C#** ou **Générer du code VB .NET**.
2. Cliquez sur l'onglet **Tâches**.
3. Sélectionnez la commande WSDL : Génération de code de proxy de service Web.



4. Cliquez sur **OK** pour lancer la génération.

Déploiement de services Web dans .NET

Pour déployer et tester un service Web pour .NET, vous devez installer IIS (Microsoft Internet Information Server). Sur la machine sur laquelle vous installez IIS, vous devez également installer le .NET Framework. Vous pouvez télécharger .NET Framework ou .NET Framework SDK depuis le site de Microsoft.

Pour déployer le code de service Web, il vous suffit de copier le fichier .asmx et les classes C# ou VB .NET sous le répertoire IIS C:\Inetpub\wwwroot\<NomPackage>, où <NomPackage> représente le nom du package. Par exemple : C:\Inetpub\wwwroot\StockQuote.

Test de services Web pour .NET

Pour tester le service Web, vous devez spécifier l'URL du service Web dans l'explorateur : http://[NomHôte]/[NomPackage]/[NomService].asmx. Par exemple : http://localhost/StockQuote/StockQuote.asmx.

Le serveur IIS Web va générer une page de test pour vous permettre de tester le service Web déployé si les paramètres d'entrée et la valeur de retour utilisent des types de données simples.

Pour tester les services Web avec des types de données complexes, vous devez créer un programme de test avec un proxy de service Web ou utiliser un outil pour envoyer un message SOAP au service Web.

Génération de services Web pour Sybase WorkSpace

Sybase WorkSpace fournit un environnement de développement intégré permettant de développer, de tester et de déployer des services Web. PowerAMC permet de générer des services Web Java et EJB et d'utiliser les utilitaires Sybase WorkSpace pour affiner leur mise en oeuvre.

Vous pouvez modéliser vos services Web en utilisant l'environnement PowerAMC standard ou en utilisant le plugin PowerAMC pour Eclipse qui fonctionne dans l'environnement WorkSpace (voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Le plugin PowerAMC pour Eclipse*).

PowerAMC vous aider à effectuer rapidement les tâches suivantes :

- Créer un service Web avec la classe de mise en oeuvre appropriée
- Définir le package de classe Java
- Générer le service Web et l'ouvrir dans l'éditeur dans Workspace Java Service Editor

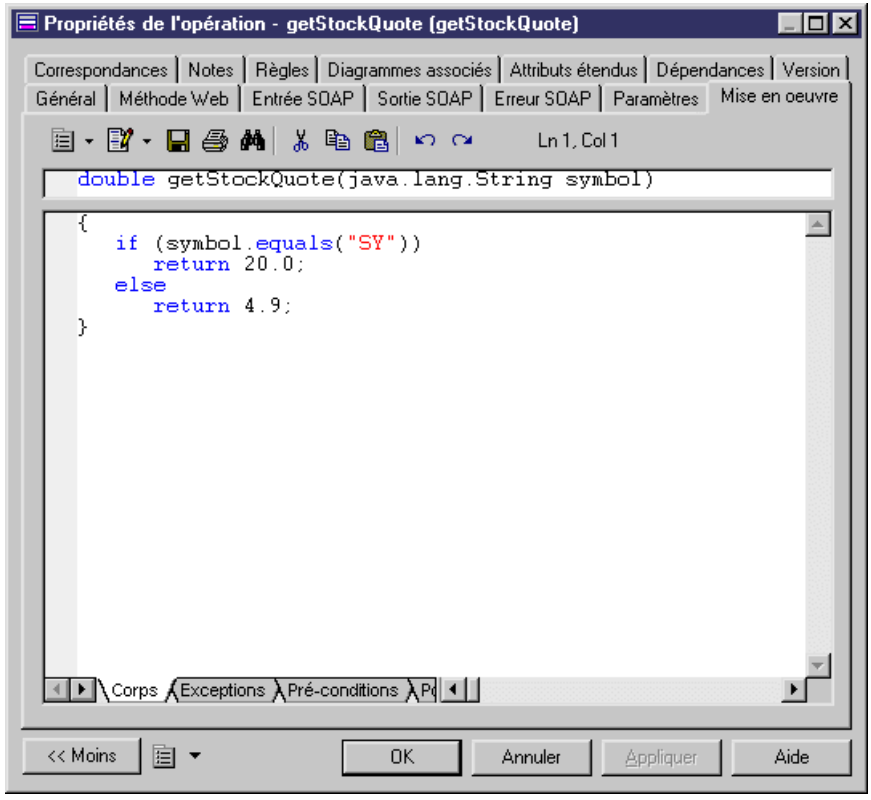
Création d'un service Web Java ou EJB pour Sybase WorkSpace

PowerAMC assure la prise en charge complète de la création d'un service Web Java pour Sybase WorkSpace.

1. Sélectionnez **Fichier > Nouveau modèle** pour afficher la boîte de dialogue Nouveau modèle, puis sélectionnez Modèle Orienté Objet dans la liste **Type de modèle**.
2. Sélectionnez Java dans la liste **Langage objet**, puis sélectionnez Diagramme de classes dans la liste **Diagramme**.
3. Cliquez sur le bouton **Sélectionner des extensions**, cliquez sur l'onglet **IDE**, sélectionnez la définition étendue de modèle Sybase WorkSpace, puis cliquez sur **OK** pour revenir à la boîte de dialogue Nouveau modèle.
4. Cliquez sur **OK** pour créer le MOO.
5. Sélectionnez **Outils > Créer un composant de service Web** pour afficher l'Assistant de service Web.
6. Saisissez un nom et un code pour le composant, puis cliquez sur **Suivant**.
7. Sélectionnez Implementation dans la liste **Type de service Web**, puis sélectionnez l'une des valeurs suivantes dans la liste **Type de composant** :
   - Java Web Service (JWS)
   - Axis EJB (Stateless Session Bean)
8. Cliquez sur **Suivant**, sélectionnez une classe de mise en oeuvre de service Web, puis cliquez sur **Terminer**.

Le composant de service Web Java est créé avec la classe de mise en oeuvre et la méthode Web correspondante.

9. Double-cliquez sur la classe de mise en oeuvre dans le diagramme, puis affichez l'onglet **Opérations** dans la feuille de propriétés de la classe.
10. Double-cliquez sur l'opération WebMethod créée par défaut. La case à cocher **Méthode de service Web** est cochée. Vous pouvez renommer l'opération si nécessaire.
11. Cliquez sur l'onglet **Mise en oeuvre** et saisissez la mise en oeuvre du service Web :



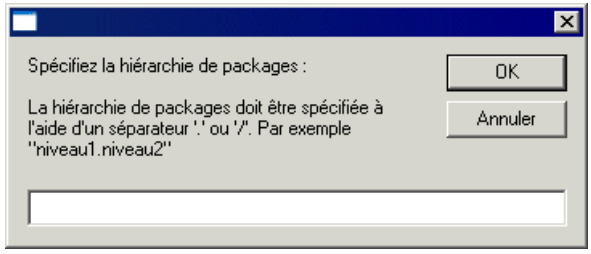
12. Cliquez sur **OK** dans les boîtes de dialogue successives.

Vous pouvez afficher un aperçu du code généré pour le service Web en cliquant sur l'onglet **Aperçu** de la classe ou du composant associé. Vous pouvez personnaliser le service Web en éditant les propriétés du composant.

Définition du package de classe Java

Lorsque vous créez un service Web en utilisant l'Assistant service Web, la classe Java et le composant de service Web sont tous les deux créés au niveau du modèle. Vous pouvez, si nécessaire, définir un nom de package pour votre classe Java et le composant.

1. Pointez sur le modèle dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Ajouter une hiérarchie de packages** dans le menu contextuel.



2. Saisissez la hiérarchie de packages appropriée et cliquez sur **OK** pour la créer dans le modèle.

Génération de services Web Java ou EJB pour Sybase WorkSpace

PowerAMC peut générer tous les fichiers nécessaires pour définir un service Web dans le Workspace Java Service Editor.

1. Sélectionnez **Langage > Générer du code Java** pour afficher la boîte de dialogue Génération.
2. Spécifiez le répertoire dans lequel les fichiers doivent être générés, et indiquez si vous souhaitez effectuer une vérification de modèle (voir *Chapitre 9, Vérification d'un MOO* à la page 303).
3. Vérifiez que Sybase WorkSpace IDE est sélectionné sur l'onglet **Cibles**.
4. Cliquez sur **OK** pour lancer la génération.
   - Un fichier svc_java ou svc_ejb qui définit le service Web
   - Un fichier .java file pour la mise en oeuvre du service Web
   - Un fichier .project pour la création d'un projet Java s'il n'existe pas
   - Un fichier .classpath pour définir des bibliothèques Java au sein d'un projet Java

   Lorsque vous générez en utilisant le plugin PowerAMC pour Eclipse, un projet Eclipse est créé ou, si vous générez dans un projet existant, le projet est réactualisé pour montrer les nouveaux fichiers.
5. Dans Workspace, pointez sur le fichier .svc_java dans la fenêtre Navigateur, cliquez le bouton droit de la souris, puis sélectionnez **Ouvrir avec > Java Service Editor**.
6. Cliquez sur le sous-onglet **Interface** et poursuivez la mise en oeuvre du service.

Pour plus d'informations sur Java Service Editor, voir votre documentation Sybase Workspace.

Notions de base relatives au .svc_java ou .svc_ejb

PowerAMC génère un fichier svc_java ou svc_ejb pour chaque service Web.

Chaque fichier contient les champs suivants :

Champ	Propriété de composant
serviceName	Code du composant de service Web
serviceComment	Commentaire du composant de service
projectName	[svc_java uniquement] Nom du dernier répertoire sur le chemin d'accès complet de la génération
projectPath	[svc_ejb uniquement] Répertoire du projet et nom du dossier
serviceFilename	%serviceName%.svc_java
authorName	Nom de modificateur de composant de service Web
dateCreated	Date et heure de génération au format : Maa jj, aaa hh:mm:ss AM/PM
ejbFullName	[svc_ejb uniquement] <PackageCode>.<ComponentCode>
operationName	Code de la méthode Web
operationComment	Commentaire de la méthode Web
static	[svc_java uniquement] Utilise "METHOD_TYPE_STATIC" si la méthode Web est statique
inputMessage	Nom du message d'entrée
outputMessage	Nom du message de sortie
returnType	Type de résultat de l'opération
parameterName	Code du paramètre d'opération
parameterComment	Commentaire du paramètre d'opération
dataType	Type de données du paramètre d'opération
javaServiceParamType	[svc_java uniquement] Type de paramètre d'opération Web
classFullPath	%projectName %/ %qualifiedClassName%
qualifiedClassName	Nom de fichier complet de classe Java
endpointName	[svc_ejb uniquement] Nom de point d'arrêt

Champ	Propriété de composant
connectionName	[svc_ejb uniquement] Nom de profil de connexion au serveur d'application.
EJBComponentURI	[svc_ejb uniquement] URI de composant EJB
jndiProviderURL	[svc_ejb uniquement] URL de fournisseur JNDI
initialContextFactory	[svc_ejb uniquement] Nom de classe de la factory de contexte initiale
jndiName	[svc_ejb uniquement] Nom JNDI d'EJB
clientJAR	[svc_ejb uniquement] Chemin d'accès du fichiers JAR client
ejbRemoteInterface	[svc_ejb uniquement] Nom qualifié de l'interface Remote d'EJB
ejbHomeInterface	[svc_ejb uniquement] Nom qualifié de l'interface Home d'EJB

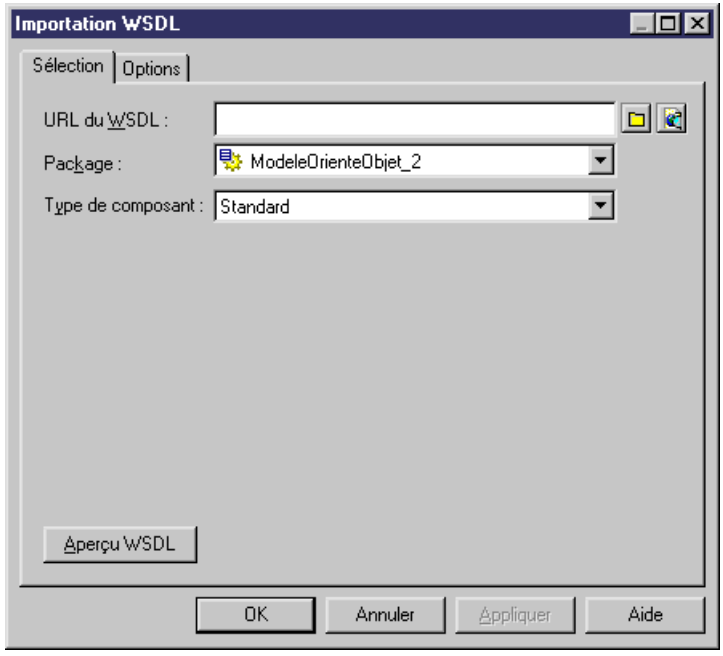
Importation de fichiers WSDL

PowerAMC peut importer des fichiers WSDL pour .NET et Java.

1. Sélectionnez **Langage > Importer du code WSDL** pour afficher la boîte de dialogue Importation WSDL.
2. Sur l'onglet **Sélection**, renseignez les zones suivantes :

Elément	Description
URL du WSDL	Indique l'emplacement du fichier de WSDL. Vous pouvez renseigner cette zone en : • Saisissant directement l'emplacement dans la zone • Cliquant sur l'outil **Rerchercher le fichier** pour sélectionner le fichier dans votre système de fichiers local • En cliquant sur l'outil Parcourir l'UDDI pour rechercher sur un serveur UDDI (voir *Recherche de fichiers WSDL via UDDI* à la page 281)
Package	Spécifie le package et l'espace de noms dans lesquels le composant et la classe de service Web seront créés.
Type de composant	[Java uniquement] Spécifie le type du composant à créer.

3. Sélectionnez les services Web et types de port à importer.



Chaque service Web sélectionné sera importé sous la forme d'un composant et d'une classe de mise en oeuvre. Chaque type de port sélectionné dans un service Web sélectionné est généré sous la forme d'une interface.

4. [facultatif] Cliquez sur l'onglet **Aperçu WSDL** pour afficher un aperçu du code WSDL ainsi que la clé unique utilisée pour localiser l'UDDI.
5. [facultatif] Cliquez sur l'onglet **Options**, qui permet de spécifier dans quels diagrammes PowerAMC doit créer les symboles pour les objets importés. Si vous désélectionnez une option, vous supprimez la création d'un symbole, mais l'objet reste importé.
6. Cliquez sur **OK** pour commencer l'importation.

   Une boîte de progression s'affiche. Si le modèle dans lequel vous effectuez le reverse engineering contient déjà des données, la boîte de dialogue Fusion de modèles vous permet de sélectionner la façon dont les objets importés seront fusionnés avec votre modèle.

   Pour obtenir des informations détaillées sur la fusion des modèles, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Comparaison et fusion de modèles*.

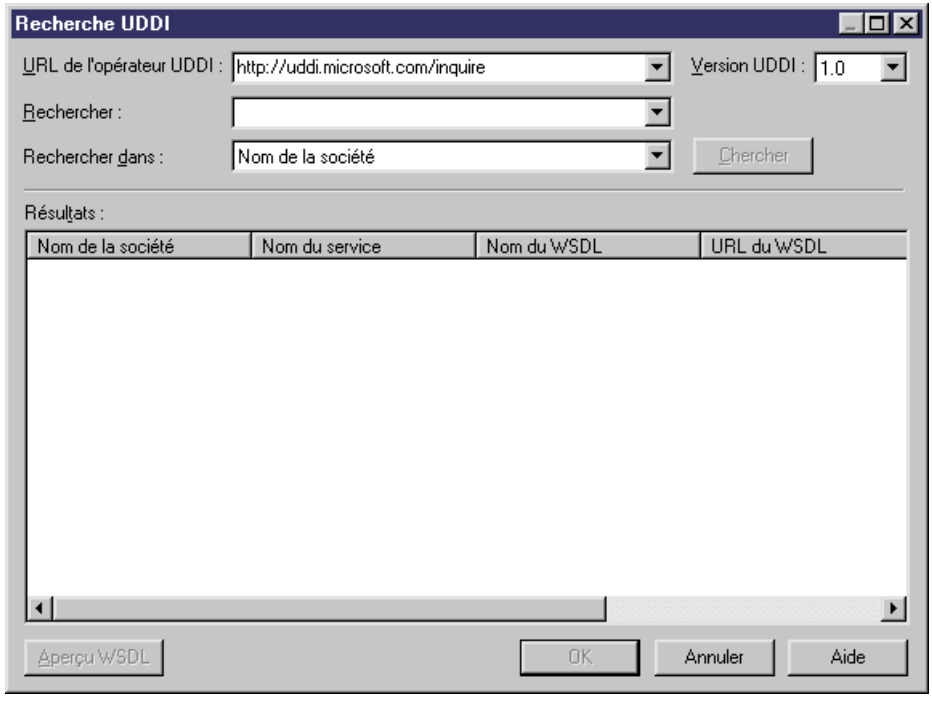
   Chaque service Web sélectionné sera importé sous la forme d'un composant et d'une classe de mise en oeuvre. Chaque type de port sélectionné dans un service Web sélectionné génère une interface.

Remarque : Si le WSDL contient une section préfixée par <!-- service -->, une instance de composant est créée. Cette section est affichée dans l'onglet WSDL de la feuille de propriétés de l'instance de composant.

Recherche de fichiers WSDL via UDDI

PowerAMC fournit une interface pour recherche un WSDL directement sur un serveur UDDI.

1. Cliquez sur l'outil **Parcourir UDDI** en regard de la zone URL du WSDL sur l'onglet Sélection de la boîte de dialogue Importation WSDL.



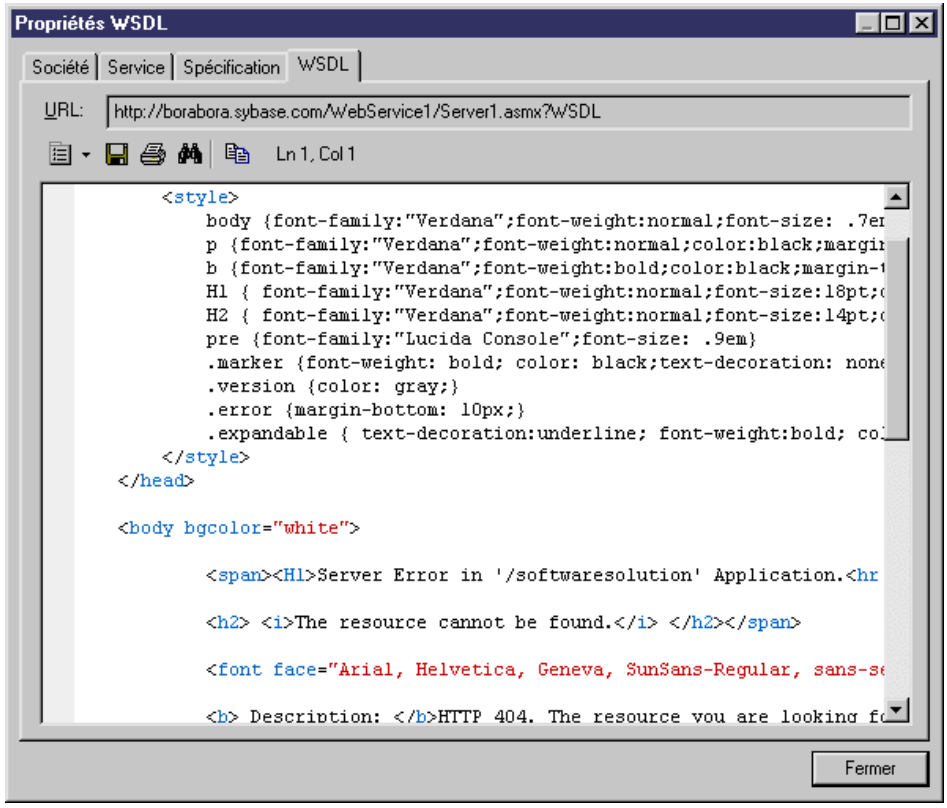
2. Renseignez les zones suivantes pour spécifier vos critères de recherche :

Item	Description
URL de l'opérateur UDDI	Choisissez parmi une liste d'URL d'opérateurs UDDI, ou saisissez votre propre URL.
Version UDDI	Spécifiez la version d'UDDI correspondant à l'URL.
Rechercher	Spécifiez le nom de l'élément que vous cherchez.
Rechercher dans	Spécifiez si vous devez faire porter la recherche sur l'entité métiers (nom de la société), le nom de service Web, ou le nom de WSDL.

3. Cliquez sur le bouton **Chercher**.

Le résultat s'affiche dans la fenêtre Résultats.

4. [facultatif] Cliquez sur le bouton **Aperçu WSDL** pour afficher la feuille de propriétés du WSDL, qui contient différents onglets permettant de visualiser les informations relatives à l'entité et au service métiers, la spécification et le code WSDL :



5. Cliquez sur **Fermer** pour revenir à la boîte de dialogue Parcourir UDDI.
6. Cliquez sur **OK** pour revenir à la boîte de dialogue Importation WSDL pour terminer l'importation.

Chapitre 7

Génération et reverse engineering de fichiers source orientés objet

PowerAMC est capable de générer des fichiers source à partir d'un MOO et d'effectuer un reverse engineering depuis ces fichiers.

Génération de fichiers source OO à partir d'un MOO

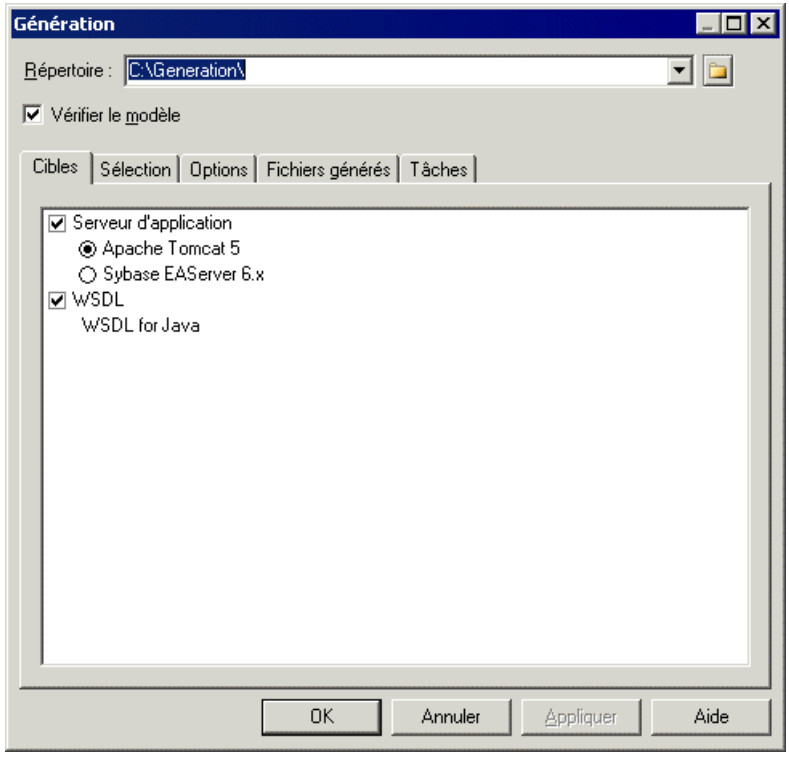
PowerAMC fournit une interface standard pour générer des fichiers source pour tous les langages orientés objet pris en charge. Pour plus de détails sur les options spécifiques au langage et sur les tâches de génération, voir le chapitre du langage approprié.

Par défaut, PowerAMC prend en charge la génération pour les types d'objet suivants pour le langage objet du MOO :

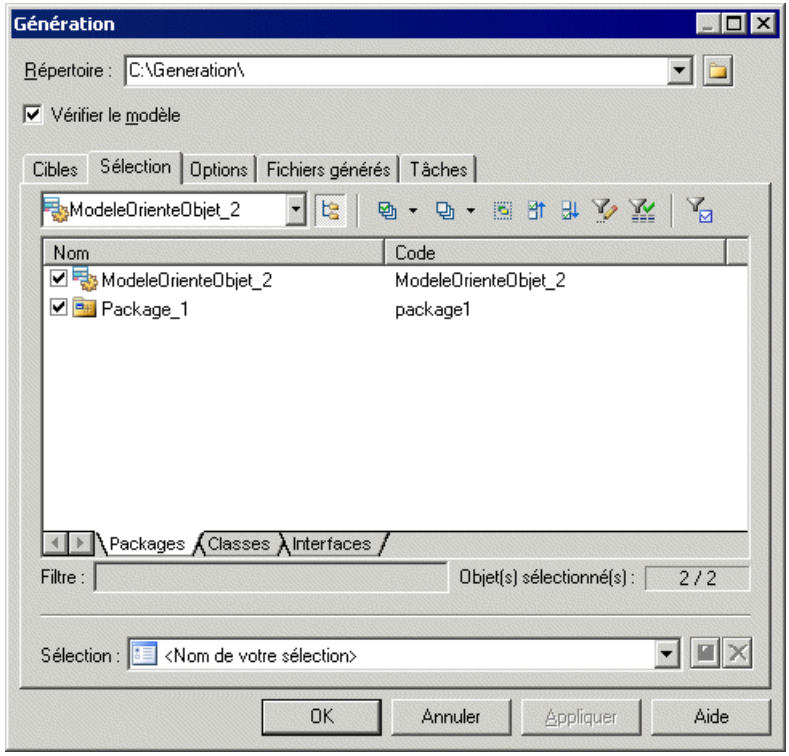
Langage objet	Ce qui est généré
Analysis	Aucun fichier n'est généré car ce langage est utilisé principalement à des fins de modélisation
C#	Fichiers de définition .CS
C++	Fichiers de définition C++ (.h et .cpp)
IDL-CORBA	Fichiers de définition IDL-CORBA
Java	Fichiers Java à partir des classes et interfaces du modèle. Inclut la prise en charge de EJB et de J2EE
PowerBuilder	Fichiers d'application .PBL ou fichiers .SRU à partir des classes du modèle
Visual Basic.Net	Fichiers .VB
XML–DTD	Fichiers .DTD
XML–Schema	Fichiers .XSD. Inclut les propriétés de langage XML standard

Remarque : Le système de génération de PowerAMC est extrêmement personnalisable grâce à l'utilisation des extensions (voir *Extension de votre environnement de modélisation* à la page 15). Pour obtenir des informations détaillées sur la personnalisation de la génération, notamment sur l'ajout de cibles, d'options et de tâches de génération, voir *Personnalisation et extension de PowerAMC > Fichiers d'extension*.

1. Sélectionnez **Langage** > **Générer du code *langage*** pour afficher la boîte de dialogue de génération :



2. Spécifiez le répertoire dans lequel les fichiers doivent être générés, et indiquez si vous souhaitez effectuer une vérification de modèle (voir *Chapitre 9, Vérification d'un MOO* à la page 303).
3. [facultatif] Sélectionnez des cibles supplémentaires pour la génération. Ces cibles sont définies par les extensions que vous pouvez attacher au modèle (voir *Gestion des cibles de génération* à la page 286).
4. [facultatif] Cliquez sur l'onglet **Sélection** et spécifiez les objets à partir desquels vous souhaitez générer. Par défaut, tous les objets sont générés, et PowerAMC se souvient des changements apportés pour les prochaines générations.



5. [facultatif] Cliquez sur l'onglet **Options** et spécifiez les options de génération appropriées. Pour plus d'informations sur ces options, voir le chapitre de langage approprié.

Remarque : Pour plus d'informations sur la modification des options qui s'affichent sur cet onglet et sur l'onglet **Tâches** ainsi que sur l'ajout de vos propres options et tâches, voir *Personnalisation et extension de PowerAMC > Fichiers de définition pour les langage objet, de processus et XML > Catégorie Generation*.

---

6. [facultatif] Cliquez sur l'onglet **Fichiers générés** et spécifiez les fichiers à générer. Par défaut, tous les fichiers sont générés.

   Pour plus d'informations sur la personnalisation des fichiers qui seront générés, voir *Personnalisation et extension de PowerAMC > Fichiers d'extension > Fichiers générés (Profile)*.

7. [facultatif] Cliquez sur l'onglet **Tâches** et spécifiez les tâches de génération spécifiques au langage à exécuter.
8. Cliquez sur **OK** pour lancer la génération.

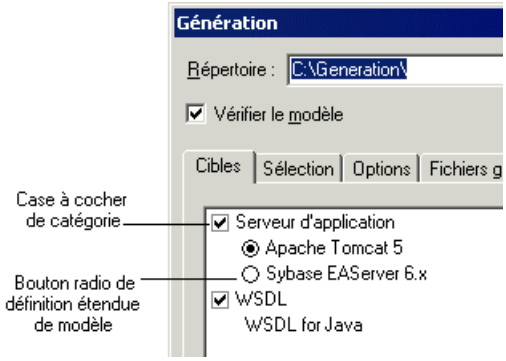
   Une fois la génération terminée, la boîte de dialogue Fichiers générés s'affiche et répertorie les fichiers générés dans le répertoire spécifié. Sélectionnez un fichier dans liste, puis cliquez sur **Editer** pour l'ouvrir dans votre éditeur associé, ou bien cliquez sur **Fermer** pour quitter la boîte de dialogue.

Gestion des cibles de génération

L'onglet Cible de la boîte de dialogue de génération permet de spécifier des cibles de génération supplémentaires, qui sont définies par les fichiers d'extension.

PowerAMC fournit de nombreuses extensions, qui peuvent étendre le langage objet pour l'utilisation avec un serveur ou un environnement particulier, etc. Vous pouvez modifier ces extensions ou créer vos propres extensions. Pour plus d'informations sur l'attachement d'extensions à votre modèle, voir *Extension de votre environnement de modélisation* à la page 15.

L'onglet cibles de la boîte de dialogue de génération regroupe les cibles par catégorie. Pour chaque catégorie, il est possible de sélectionner une ou plusieurs extensions à la fois.



Pour obtenir des informations détaillées sur l'édition et la création d'extensions, voir *Personnalisation et extension de PowerAMC > Fichiers d'extension*.

Définition du package de code source

Dans le cas des langages qui prennent en charge les concepts de packages et/ou d'espaces de noms, les classes doivent être générées dans les packages qui sont utilisés comme espaces de nom qualifiants. Vous pouvez définir ces packages qualifiants en utilisant la commande Ajouter une hiérarchie de packages.

1. Pointez sur le modèle dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez Ajouter une hiérarchie de packages dans le menu contextuel.
2. Saisissez une hiérarchie de packages dans la zone de texte, en utilisant des points ou des barres obliques pour séparer les packages. Par exemple :

```
com.masociete.monproduit.moo
```

ou

```
com/masociete/monproduit/moo
```

La hiérarchie de packages correspondante sera créée dans l'Explorateur d'objets. Tous les diagrammes et objets (à l'exception des objets globaux) existant dans le modèle seront déplacés dans le package situé au niveau le plus bas dans la hiérarchie.

Reverse engineering de fichiers orientés objet dans un MOO

Le reverse engineering est le processus qui consiste à extraire des données ou du code source à partir d'un fichier, puis à l'utiliser pour construire ou mettre à jour un MOO. Vous pouvez procéder au reverse engineering d'objets en les transférant dans un nouveau modèle, ou bien dans un modèle existant.

Vous pouvez faire procéder au reverse engineering des types de fichier suivants vers un MOO :

- IDL
- Java
- Objets PowerBuilder
- XML - PowerAMC utilise un analyseur syntaxique développé par Apache Software Foundation (*http://www.apache.org*).
- C#
- VB
- VB.NET

Classificateurs internes

Lorsque vous procédez au reverse engineering vers un MOO d'un langage qui comporte un ou plusieurs classificateurs internes (voir *Classificateurs composites et classificateurs internes* à la page 49) une classe est créée pour la classe externe et une classe est créée pour chaque classificateur interne, de plus un lien interne est créé entre chaque classificateur interne et la classe externe.

Création de symbole

Si vous sélectionnez l'option de reverse engineering Créer des symboles, la disposition des symboles dans le diagramme est définie automatiquement. Si le reverse engineering porte sur un grand nombre d'objets avec des interactions complexes, la fonctionnalité de disposition automatique peut être amenée à créer des synonymes d'objets pour améliorer la lisibilité du diagramme.

Reverse engineering de fichiers orientés objet dans un nouveau MOO

Vous pouvez procéder au reverse engineering de fichiers de langage objet pour créer un nouveau MOO.

1. Sélectionnez **File > Reverse engineering > Langage objet** pour afficher la boîte de dialogue Nouveau modèle orienté objet.

2. Sélectionnez un langage objet dans la liste, puis cliquez sur le bouton radio **Partager**.
3. [facultatif] Cliquez sur l'onglet **Sélection d'extensions**, puis sélectionnez les extensions que vous souhaitez attacher au nouveau modèle.

   Pour plus d'informations sur l'utilisation des extensions, voir *Personnalisation et extension de PowerAMC > Fichiers d'extension*.
4. Cliquez **OK** pour afficher la fenêtre de reverse engineering spécifique au langage appropriée. Pour obtenir des informations détaillées, reportez-vous au chapitre relatif à ce langage.
5. Sélectionnez le fichiers sur lesquels vous souhaitez faire porter le reverse engineering, définissez les options appropriées, puis cliquez sur **OK** pour lancer le reverse engineering.

   Une boîte de progression s'affiche. Les classes sont automatiquement ajoutées dans votre modèle.

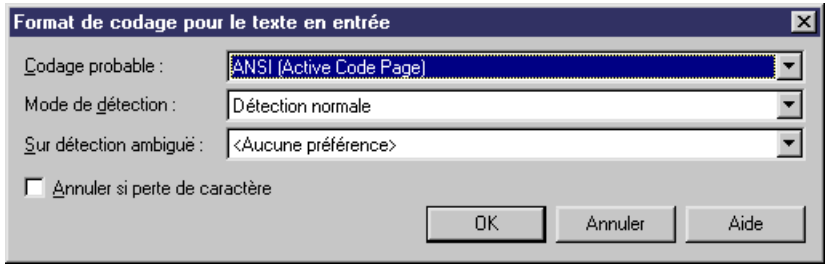
Remarque : Ce produit inclut le logiciel XML4C 3.0.1 développé par Apache Software Foundation (*http://www.apache.org*)

Copyright (c) 1999 The Apache Software Foundation. Tous droits réservés. LE LOGICIEL XML4C 3.0.1 (LE "LOGICIEL") EST FOURNI "EN L'ÉTAT" ; AUCUNE GARANTIE EXPRESSE OU IMPLICITE N'EST DONNÉE, Y COMPRIS, MAIS SANS LIMITATION, LES GARANTIES IMPLICITES DE QUALITÉ MARCHANDE ET D'ADÉQUATION À UN OBJECTIF PARTICULIER. EN AUCUN CAS THE APACHE SOFTWARE FOUNDATION OU SES DONATEURS NE SAURAIENT ÊTRE TENUS RESPONSABLES POUR TOUT DOMMAGE DIRECT, INDIRECT, IMMATÉRIEL OU CONSÉCUTIF (Y COMPRIS, MAIS SANS LIMITATION, LA FOURNITURE D'UN PRODUIT OU D'UNE PRESTATION DE SUBSTITUTION ; LA PERTE D'EXPLOITATION, DE DONNÉES OU DE BÉNÉFICES ; L'INTERRUPTION DE L'ACTIVITÉ COMMERCIALE), QU'ELLE QU'EN SOIT LA CAUSE ET QUE LA RESPONSABILITÉ ENGAGÉE SOIT CONTRACTUELLE, DÉLICTUELLE OU QUASI-DÉLICTUELLE (Y COMPRIS PAR NÉGLIGENCE OU POUR TOUT AUTRE MOTIF), ET SURVENANT DE QUELQUE MANIÈRE QUE CE SOIT DU FAIT DE L'UTILISATION DU LOGICIEL, MÊME EN AYANT ÉTÉ INFORMÉ DE LA POSSIBILITÉ DE LA SURVENANCE D'UN TEL DOMMAGE.

Reverse engineering de format de codage

Si les applications Java sur lesquelles vous souhaitez effectuer un reverse engineering contiennent des fichiers sources écrits avec Unicode ou MBCS (Multibyte Character Set), vous devez utiliser les paramètres de codage mis à votre disposition dans la zone Codage de fichier.

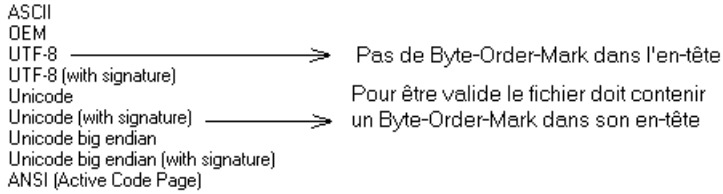
Si vous souhaitez changer ces paramètres car vous savez quel codage est utilisé dans les sources, cliquez sur le bouton Points de suspension en regard de la zone Codage de fichier pour sélectionner le paramètre de codage approprié. Vous affichez ainsi la boîte de dialogue Format de codage pour le texte en entrée qui permet de sélectionner le type de codage de votre choix.



La boîte de dialogue Format de codage pour le texte en entrée inclut les propriétés suivantes :

Propriété	Description
Codage probable	Format de codage à utiliser comme codage probable lors du reverse engineering du fichier
Mode de détection	Indique si la détection de codage de texte doit être tentée et spécifie la quantité de chaque fichier qui doit être analysée. Vous pouvez sélectionner l'une des options suivantes : • Aucune détection - Désactive la fonctionnalité de détection. Sélectionnez cette valeur si vous connaissez le codage de format • Détection rapide - Analyse un fragment limité de la mémoire tampon dans le cadre de la détection. Sélectionnez cette option lorsque vous pensez que le format de codage sera facile à identifier • Détection complète - Analyse la totalité du fichier dans le cadre de la détection. Sélectionnez cette option lorsque vous pensez que le nombre de caractères qui peuvent déterminer le format de codage est très restreint
Sur détection ambiguë	Spécifie le type d'action à entreprendre en cas d'ambiguïté. Vous pouvez sélectionner l'une des options suivantes : • Utiliser le codage spécifié et afficher un avertissement - le codage probable est utilisé et un message d'avertissement s'affiche dans la page Reverse de la fenêtre Résultats • Utiliser le codage spécifié - utilise le format de codage sélectionné dans la zone Codage probable. Aucun message d'avertissement n'est affiché • Utiliser le codage détecté - Utilise le format de codage détecté par PowerAMC
Annuler si perte de caractère	Permet d'arrêter le reverse engineering si des caractères ne peuvent pas être identifiés et risquent d'être perdus lors du codage de fichier

Voici un exemple de lecture de formats de codage dans la liste :



Reverse engineering dans un MOO existant

Vous pouvez également effectuer un reverse engineering de fichiers sources vers un MOO existant.

1. Sélectionnez **Langage > Reverse engineering** pour afficher la boîte de dialogue de reverse engineering.
2. Sélectionnez l'option de reverse engineering de fichiers ou de répertoires dans la liste Reverse engineering.
3. Cliquez sur le bouton Ajouter dans l'onglet Sélection afin d'afficher une boîte de dialogue d'ouverture de fichier standard.
4. Sélectionnez les fichiers ou le répertoire sur lesquels effectuer le reverse engineering, puis cliquez sur Ouvrir.
5. Cliquez sur OK pour lancer le reverse engineering.

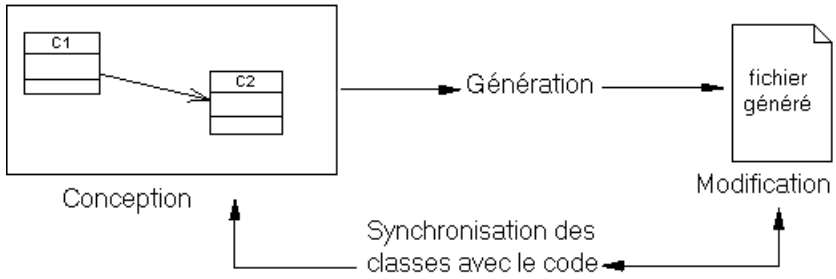
   La fenêtre résultats affiche un message pour indiquer que le fichier spécifié a subi le reverse engineering et la boîte de dialogue Fusion de modèles s'affiche.
6. Passez en revue les objets que vous allez importer, ainsi que les changements qu'ils provoqueront dans le modèle.

   Pour plus d'informations sur la fusion de modèles, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Comparaison et fusion de modèles*.
7. Cliquez sur OK pour fusionner les changements sélectionnés dans votre modèle.

Synchronisation d'un modèle avec des fichiers générés

Vous pouvez modéliser votre système dans PowerAMC, utiliser le processus de génération, puis visualiser et modifier le fichier généré dans l'éditeur de code, synchroniser les

classificateurs avec le code source, puis revenir au modèle. Vous effectuez ainsi vos modifications dans le fichier qui sera utilisé comme source pour le reverse engineering.



La synchronisation affiche une fenêtre de reverse engineering dans laquelle l'option appropriée est présélectionnée et remplit la liste des classificateurs au moyen de classificateurs sélectionnés dans le diagramme de classes.

Vous pouvez ensuite facilement localiser les fichiers qui doivent être pris en compte pour la synchronisation. Si aucun classificateur n'est sélectionné, le reverse engineering présélectionne les répertoires et ajoute le répertoire courant dans la liste.

1. Sélectionnez **Langage > Synchroniser avec les fichiers générés** pour afficher la boîte de dialogue Reverse engineering de Java.

   La boîte de dialogue s'affiche à la page Sélection.

2. Sélectionnez l'option de reverse engineering de fichiers ou de répertoires dans la liste Reverse engineering.
3. Cliquez sur le bouton Ajouter pour afficher la boîte de dialogue de sélection de dossier.
4. Sélectionnez le répertoire approprié, puis cliquez sur OK pour afficher la boîte de dialogue Reverse engineering de Java appropriée.
5. Cliquez sur OK pour lancer la synchronisation.

   Une boîte de progression s'affiche, suivie d'une boîte de dialogue de fusion.

   Remarque : La boîte de dialogue Fusion de modèles affiche le modèle Depuis (répertoire source) dans le volet gauche et le modèle Vers (modèle courant) dans le volet droit. Vous pouvez développer les noeuds dans le volet droit pour vérifier que les actions de fusion sélectionnées correspondent aux opérations que vous souhaitez effectuer.
6. Passez en revue les objets que vous allez importer, ainsi que les changements qu'ils provoqueront dans le modèle, puis cliquez sur OK.

   La page Reverse de la fenêtre Résultats affiche les modifications apportées lors de la synchronisation et la fenêtre de diagramme affiche le modèle synchronisé.

Chapitre 8

Génération d'autres modèles à partir d'un MOO

Vous pouvez générer des modèles conceptuels et physiques de données et des modèles XML à partir d'un MOO.

Le tableau suivant détaille comment les objets de MOO sont générés dans d'autres modèles :

MOO	MCD	MPD	MSX
Domaine	Domaine	Domaine	Type de simple
Classe (cases Persistante et Générer cochées).	Entité	Table (la cardinalité d'une classe de vient le nombre d'enregistrement d'une table.)	Elément
Classe abstraite	Entité	Table	Type complexe
Attribut (case Persistante cochée)	Attribut	Colonne	Attribut ou élément (voir les options de génération)
Identifiant	Identifiant	Identifiant	Clé
Composition	-	-	Nouveau niveau dans la hiérarchie des éléments
Opération avec le stéréotype <<storedProcedure>> (classe parent générée comme une table)	-	Procédure stockée attachée à la table, avec le corps de l'opération comme corps dans la définition de procédure.	-
Opération avec le stéréotype <<storedFunction>> (classe parent générée comme une table)	-	Fonction stockée attachée à la table, avec le corps de l'opération comme corps dans la définition de fonction.	-

MOO	MCD	MPD	MSX
Association	Relation ou association	Table (si multiplicité - plusieurs-plusieurs) ou référence. Les noms de rôle deviennent des clés étrangères migrées.	Contraintes KeyRef
Classe d'association	Notation Entités/Relations : entité avec deux associations. Notation Merise : association et deux liens d'association	Une table et deux associations entre les extrémités de la classe d'association	-
Dépendance	-	-	-
Réalisation	-	-	-
Généralisation	Héritage	Référence	Dérivation de type complexe (XSD) ou migration d'attribut vers un élément enfant (DTD)

1. Sélectionnez **Outils**, puis l'une des commandes suivantes pour ouvrir la fenêtre Options de génération de modèle appropriée :
   - **Générer un Modèle Conceptuel de Données... Ctrl+Maj+C** - Par exemple, pour convertir les classe de MOO en entités de MCD. Vous serez alors en mesure d'affiner la conception de votre modèle et, si vous le souhaitez, de générer un Modèle Physique de Données (MPD) à partir de ce MCD
   - **Générer un Modèle Physique de Données... Ctrl+Maj+P** - Par exemple, pour convertir la modélisation de votre système dans votre base de données. L'avantage de ce processus est qu'il permet de modéliser les objets dans l'environnement dans lequel ils seront utilisés, et d'automatiser la conversion en tables et colonnes de base de données.
   - **Générer un Modèle Orienté Objet... Ctrl+Maj+O** - Par exemple, pour transformer un MOO analytique (modélisé à l'aide du langage objet Analysis) en MOO de mise en oeuvre modélisés pour Java, C#, et les autres langages objet pris en charge par PowerAMC.
   - **Générer un Modèle XML... Ctrl+Maj+M** - Par exemple, pour générer un format de message à partir de votre structure de classes.

2. Sur l'onglet **Général**, sélectionnez une option permettant de choisir de générer un nouveau modèle ou de mettre à jour un modèles existant, puis spécifiez les options appropriées.
3. [facultatif] Cliquez sur l'onglet **Détails** puis définissez les options appropriées. Nous vous recommandons de cocher la case Vérifier le modèle pour rechercher les erreurs ou avertissements éventuels avant de procéder à la génération.
4. [facultatif] Cliquez sur l'onglet **Modèles cible** et spécifiez les modèles cibles pour les éventuels raccourcis générés.
5. [facultatif] Cliquez sur l'onglet **Sélection** puis sélectionnez ou désélectionnez les objets à générer.
6. Cliquez sur **OK** pour lancer la génération.

Remarque : Pour obtenir des informations détaillées sur les options disponibles sur les différents onglets de la fenêtre de génération, voir *Guide des fonctionnalités générales > Liaison et synchronisation de modèles > Génération de modèles et d'objets de modèle*.

Gestion de la persistance des objets lors de la génération de modèles de données

Les développeurs utilisent les langages orientés objet pour développer des objets métiers qui seront stockés dans une base de données. PowerAMC fournit diverses propriétés pour vous permettre de contrôler avec précision la génération des objets persistants dans un modèle de données.

Il peut arriver que les codes des classes et attributs dans les langages de programmation orientés objet (spécifiés dans la zone **Code** sous la zone **Nom** dans l'onglet **Général** de leur feuille de propriétés) soient différents des codes utilisés pour les tables et les colonnes dans le modèle de données qui représente la base de données.

Dans ces cas, vous pouvez spécifier un code alternatif persistant dans la zone **Code** en utilisant la zone de groupe **Persistant** sur l'onglet **Détails** de la feuille de propriétés des classes et attributs. Ces codes seront utilisés à la place des codes standard lors de la génération d'un modèle de données et faciliteront l'ingénierie par va-et-vient en permettant de récupérer les codes d'objets dans la base de données.

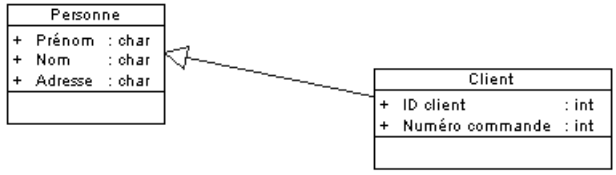
Les autres propriétés de ces zones de groupe **Persistant** vous aident à contrôler la façon dont les classes seront générées dans les modèles de données (voir *Propriétés d'une classe* à la page 38) et les types de données qui seront utilisés pour les attributs (voir *Propriétés d'un attribut* à la page 73)

Remarque : Vous pouvez également créer des correspondances objet-relationnel entre des objets de MOO et de MCD ou MPD en utilisant l'éditeur de correspondances (voir *Guide des fonctionnalités générales > Liaison et synchronisation de modèles > Mise en correspondance d'objets*).

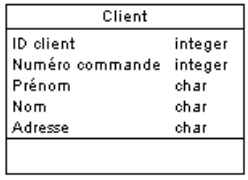
Gestion de la persistance pour les généralisations

Vous pouvez contrôler la génération des classes connectées par un lien de généralisation en entités de MCD ou tables de MPD à l'aide des options `Générer une table` et `Migrer les colonnes` dans la zone de groupe **Persistant** située sur l'onglet **Détails**.

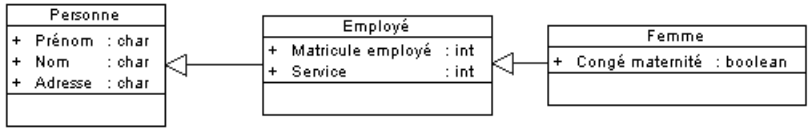
Dans l'exemple suivant, `Client` est défini à `Générer une table` et hérite, via un lien de généralisation, de `Personne`, qui est défini à `Migrer les colonnes` :



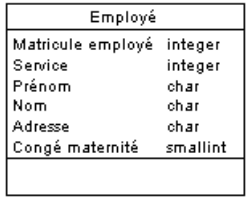
Client hérite des attributs de la classe parent dans le MPD généré :



Les classes dérivées sont créées pour améliorer la lisibilité d'un modèle, mais elles n'ajoutent aucune information sémantique et ne sont pas générées dans un MPD, leurs attributs étant migrés vers le parent. Dans l'exemple suivant, `Femme` et `Personne` sont tous les deux définis à `Migrer les colonnes`, tandis que `Employé` est défini à `Générer une table` :



Dans le MPD généré, Employé hérite à la fois de sa classe parent et de la classe dérivée :

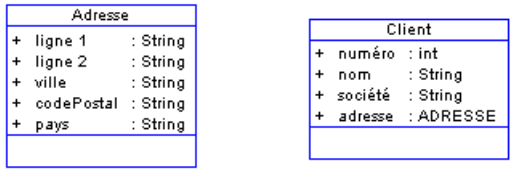


Pour plus d'informations, voir *Propriétés d'une classe* à la page 38 et *Généralisations (MOO)* à la page 101.

Gestion de la persistance pour les types de données complexes

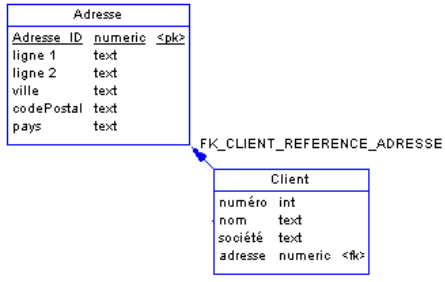
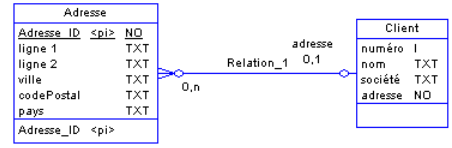
Lorsque vous spécifiez une classe comme type de données d'un attribut, vous pouvez contrôler sa génération vers un MCD ou un MPD en utilisant les options `Générer une table`, `Type de valeur` et `Générer un type de données abstrait` (MPD uniquement) dans la zone de groupe **Persistant** située sur l'onglet **Détails**.

Dans l'exemple suivant, `Client` contient un attribut, `Adresse`, pour lequel la classe `Adresse` a été sélectionnée comme type de données (voir *Spécification d'un classificateur comme type de données ou type de résultat* à la page 51) :



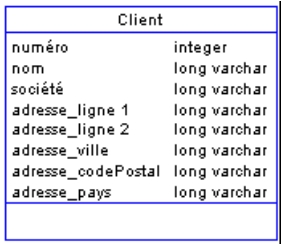
Client est spécifié comme persistant, et l'option `Générer une table` est sélectionnée. Vous pouvez générer la classe `Adresse` de l'une des façons suivantes :

- Sous forme de classe persistante - en sélectionnant `Générer une table` dans la zone de groupe **Persistant** située sur l'onglet **Détails** (voir *Propriétés d'une classe* à la page 38) :

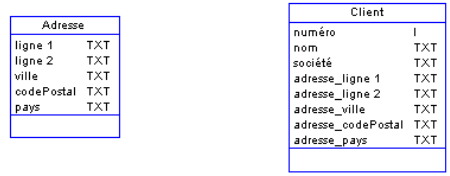
Dans un MPD, les deux classes sont générées sous forme de tables liées par une référence qui relie la colonne de clé primaire `Adresse ID` (créée lors de la génération) dans la table parent `Adresse` et la colonne de clé étrangère `Adresse` dans la table enfant `Client` : 	Dans un MCD, deux classes sont générées sous forme d'entités liées par une relation un-plusieurs qui relie l'identifiant primaire `Adresse ID` (créé lors de la génération) dans l'entité `Adresse` et l'attribue `Adresse` dans l'entité `Client` : 

- Sous la forme d'une classe incorporée - en sélectionnant `Type de valeur` :

Dans un MPD, `Client` est généré comme une table, avec tous les attributs de `Adresse` incorporés comme colonnes préfixées par `adresse_` :

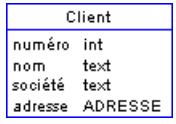


Dans un MCD, les deux classes sont générées comme entités, et `Client` inclut tous les attributs de `Adresse` incorporés comme attributs préfixés par `adresse_` :



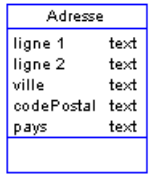
- Sous forme de classe de type de données abstrait (MPD uniquement) - en sélectionnant `Générer un type de données abstrait` :

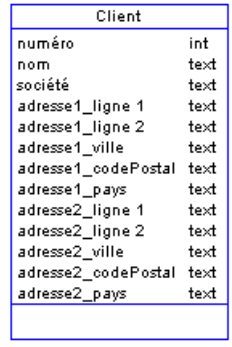
Dans un MPD, `Client` est généré comme une table et `Adresse` est généré comme un type de données abstrait (qui n'a pas de symbole), qui est référencé par la colonne `Adresse` :



Remarque : Si vous spécifiez une multiplicité (voir *Propriétés d'un attribut* à la page 73) pour l'attribut utilisant un type de données complexe, lorsque vous générez :

- Une classe persistante - la multiplicité est générée sous forme de cardinalité sur la référence entre les tables.
- Une classe incorporée - les attributs sont générés le nombre maximum de fois requis par la multiplicité, mais uniquement si le maximum est défini à une valeur fixe. Dans l'exemple suivant, la multiplicité d'attribut est 0..2, de sorte que les attributs seront incorporés deux fois :





- Une classe de type de données abstrait - pour les SGBD qui prennent en charge ARRAY et LIST pour les types de données abstraits, la multiplicité affecte la génération comme suit :

- 0..n ou 1..n - générer un type de données abstrait de type LIST (exemple : TABLE pour Oracle).
- 1..2 ou 0..10 - générer un type de données abstrait de type ARRAY (exemple : VARRAY pour Oracle).
- 0..1 ou 1..1 - générer un type de données abstrait de type OBJECT.

Personnalisation de la génération MSX pour les objets individuels

Lorsque vous générez un MSX à partir d'un MPD ou d'un MOO, vous pouvez spécifier des options de génération globales afin de générer des tables/classes sous la forme des éléments avec ou sans types complexes et des colonnes/attributs sous forme d'éléments ou d'attributs. Vous pouvez passer outre ces options pour les objets individuels en attachant les extensions `PDM XML Generation` ou `OOM XML Generation` à votre modèle source et en sélectionnant les options de génération XML de ces extensions.

Remarque : L'extension fournit de nouveaux onglets de feuille de propriétés permettant de définir des options de génération pour des objets individuels, mais vous pouvez également définir ces options avec ou sans l'extension en sélectionnant **Modèle > *objets*** pour ouvrir la liste d'objets appropriée, cliquer sur l'outil **Personnaliser les colonnes et filtre**, et choisir d'afficher la colonne `Mode de génération XML`.

Par exemple, si vous souhaitez générer la plupart de vos colonnes de table dans un MSX sous la forme d'attributs XML, mais que vous voulez générez certaines colonnes sous forme d'éléments, procédez comme suit :

- Modifiez les options de génération XML pour les colonnes que vous souhaitez générer comme éléments.
- Sélectionnez l'option de génération des colonnes sous forme d'attributs sur l'onglet **Détails** de la boîte de dialogue Options de génération.

1. Sélectionnez **Modèle > Extensions** pour afficher la Liste des extensions, puis cliquez sur l'outil **Attacher une extension**.
2. Sur l'onglet **Général**, sélectionnez `PDM XML Generation` ou `OOM XML Generation` puis cliquez sur **OK** afin d'attacher l'extension à votre modèle, puis sur **OK** pour fermer la Liste des extensions.

   Ces fichiers d'extension activent l'affichage de l'onglet **XML** dans toutes les feuilles de propriétés de table et de colonne.
3. Ouvrez la feuille de propriétés de la table, colonnes, classe ou attribut dont vous souhaitez personnaliser la génération, puis cliquez sur **XML**.
4. Utilisez les options pour spécifier comment vous voulez générer les objets dans le MSX.
   - Pour les tables et les classes, vous pouvez spécifier qu'elles doivent être générées en tant que :

    - Eléments - chaque table/classe est générée comme un élément non typé directement lié à ses colonnes/attributs générés comme attributs ou sous-éléments.
    - Eléments avec des types complexes - la table/classe est générée comme un élément typé par un type complexe, généré en parallèle, afin de contenir les colonnes/attributs.
    - Défaut - la génération de la table/classe est contrôlée par l'option sélectionnée dans la zone de groupe **Génération XML** sur l'onglet **Détails** de la boîte de dialogue Options de génération.
- Pour les tables, vous pouvez également spécifier la génération des clés sous forme de :
    - Clé - [défaut] Les colonnes de clé primaire sont générées, de même que KEY et KEYREF chaque fois que la table est référencée.
    - Attribut d'ID - Les colonnes de clé primaire ne sont pas générées, et un attribut d'ID, `id`, est généré pour les remplacer. Chaque fois que la table est référencée, l'attribut IDREF est généré pour référencer l'élément approprié. Si le nom de rôle de référence est assigné, l'attribut prend ce nom. Dans le cas contraire, le nom de la table référencée est utilisé et le mécanisme de changement de nom standard est utilisé automatiquement.
    - Clé et attribut d'ID - Le plus souvent, les colonnes de clé contiennent des données significatives et vous souhaitez les générer, de même qu'un attribut d'ID. Dans ce cas, un attribut d'ID est généré pour l'élément, et IDREF est systématiquement utilisé pour toutes référence à la table :

    Les règles suivantes s'appliquent à la génération des clés :
    - Si une table génère un ID, toutes ses tables enfant vont générer un attribut d'ID.
    - Si une table génère des colonnes de clé, toutes ses tables enfant vont générer des colonnes de clé.
    - Si une table enfant est censée générer une clé primaire uniquement, l'attribut d'ID sera automatiquement généré.
    - Si une table génère un attribut d'ID, ni Key ni KeyRef ne sera généré, et TOUTES les références vont générer un attribut IDREF.. (même si la table génère également des colonnes de clé)
    - Si une table génère UNIQUEMENT un attribut d'ID, toutes les colonnes de clé étrangère qui référencent ses colonnes de clé seront systématiquement supprimées et remplacées par un attribut IDREF
- S'agissant des colonnes et des attributs, vous pouvez spécifier une génération sous forme de :
    - Eléments - [défaut] la colonne/l'attribut est généré(e) comme sous-élément de l'élément ou du type complexe de sa table/classe.
    - Attributs - la colonne/l'attribut est généré comme attribut de l'élément ou du type complexe de sa table/classe.

- Défaut - la génération de la colonne/de l'attribut est contrôlée par l'option sélectionnée dans la zone de groupe **Génération XML** dans l'onglet **Détail** de la boîte de dialogue Options de génération.

5. Modifiez les options de génération XML pour tout autre objet que vous souhaitez générer de façon différente.
6. Sélectionnez **Outils > Générer un modèle XML**, assurez-vous d'avoir correctement défini les options dans la zone de groupe **Génération XML** située sur l'onglet **Détails** de la boîte de dialogue Options de génération, puis lancez votre génération.

Chapitre 9 Vérification d' un MOO

Le modèle orienté objet est un outil très souple, qui vous permet de développer votre modèle rapidement et sans contrainte. Vous pouvez vérifier la validité de votre MOO à tout moment.

Un MOO valide doit respecter les types de règles suivants :

- Chaque nom d'objet doit être unique dans un MOO
- Chaque classe dans un MOO doit avoir au moins un attribut et une opération
- Chaque début ou fin doit être lié à un objet du diagramme

Remarque : Il est recommandé de procéder à la vérification de la validité du modèle orienté objet avant de générer du code ou un autre modèle à partir de ce modèle . Si une erreur est détectée, la génération est interrompue. L'option **Vérifier le modèle** est activée par défaut dans la boîte de dialogue de génération.

Vous pouvez vérifier votre modèle de l'une des façons suivantes :

- Appuyez sur F4, ou
- Sélectionnez **Outils > Vérifier le modèle**, ou
- Pointez sur le fond du diagramme, cliquez le bouton droit de la souris, puis sélectionnez Vérifier le modèle dans le menu contextuel

La boîte de dialogue Paramètres de vérification de modèle s'affiche, et vous permet de spécifier le type de vérifications à effectuer, ainsi que les objets sur lesquels vous souhaitez faire porter ces vérifications. Les sections suivantes documentent les vérifications spécifiques au MOO disponibles par défaut. Pour plus d'informations sur les vérifications effectuées sur des objets génériques disponibles dans tous les types de modèles et pour des informations détaillées sur l'utilisation de la boîte de dialogue Paramètres de vérification de modèle, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Objets > Vérification de modèles*.

Vérification des domaines

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des domaines.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Incohérence entre les valeurs par défaut et les paramètres de contrôle	Les valeurs spécifiées dans l'onglet Paramètres de contrôle ne sont pas cohérentes pour les types de données numériques et de chaîne : la valeur de défaut n'est pas compatible avec les valeurs minimum et maximum, elle n'appartient pas à la liste des valeurs spécifiée ou bien la valeur minimale est supérieure à la valeur maximale. Les paramètres de contrôle doivent avoir été définis de façon cohérente. • Correction manuelle : Modifiez la valeur de défaut, les valeurs mininum/maximum ou la liste des valeurs sur l'onglet de paramètres de contrôle • Correction automatique : Aucune

Vérification des sources de données

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des sources de données.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Existence du modèle	Une source de données doit comporter au moins un modèle physique de données dans sa définition. • Correction manuelle : Ajoutez un modèle physique de données via l'onglet Modèles de la feuille de propriétés de source de données • Correction automatique : Supprime la source de données dépourvue de modèle physique de données
Source de données contenant des modèles ayant des SGBD différents	Les modèles qui composent une source de données représentent une seule et même base de données. Les différents modèles de cette source de données doivent utiliser le même SGBD. • Correction manuelle : Supprimez les modèles dont le SGBD est différent ou modifiez le SGBD des modèles inclus dans la source de données • Correction automatique : Aucune

Vérification des packages

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des packages.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Héritage circulaire	Les objets dépendent les uns des autres. Les liens circulaires doivent être détectés. • Correction manuelle : Supprimez les liens de généralisation circulaires • Correction automatique : Aucune
Dépendance circulaire	Les classes dépendent les unes des autres via les liens des associations ou classes d'association et/ou des généralisations. Les liens circulaires doivent être détectés. • Correction manuelle : Supprimez les liens circulaires • Correction automatique : Aucune
Unicité de code de raccourci	Un même espace de nom ne peut pas contenir deux raccourcis au code identique. • Correction manuelle : Modifiez le code en double • Correction automatique : Ajoute un numéro au code en double
Raccourci risquant d'être généré sous forme de table enfant d'une référence	Le package ne doit pas contenir d'association avec un raccourci externe sous forme de table enfant. Ce type de structure peut être toléré dans le MOO, mais l'association ne pourra pas être générée dans un MPD si le raccourci externe est généré sous forme de raccourci. • Correction manuelle : Modifier votre modèle afin de créer l'association dans le package dans lequel la classe enfant est définie • Correction automatique : Aucune

Vérification des acteurs et des cas d'utilisation

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des acteurs et des cas d'utilisation.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Acteur/cas d'utilisation lié à aucun objet	Un acteur ou un cas d'utilisation doit être lié à au moins un objet dans le modèle. • Correction manuelle : Créez un lien entre l'acteur et un cas d'utilisation, ou un objet • Correction automatique : Aucune

Vérification des classes

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des classes.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune

Vérification	Description et correction
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Classificateur vide	Il manque des attributs et des opérations. • Correction manuelle : Ajoutez des attributs et des opérations au classificateur • Correction automatique : Aucune
Classe persistante sans attribut persistant	Une classe persistante ne peut pas avoir uniquement des attributs non persistants. • Correction manuelle : Définissez au moins un attribut comme persistant • Correction automatique : Aucune
Classe d'association avec identifiant	Une classe d'association ne peut pas avoir d'identifiant. • Correction manuelle : Supprimez l'identifiant • Correction automatique : Aucune
Visibilité du classificateur	Un classificateur privé ou protégé doit être interne à un autre classificateur. • Correction manuelle : Définissez une visibilité Public ou Package pour le classificateur • Correction automatique : Change la valeur de visibilité en Public ou Package
Type de résultat de constructeur de classe	Un constructeur ne doit pas avoir de type de résultat. • Correction manuelle : Supprimez le type de résultat • Correction automatique : Supprime le type de résultat
Modificateurs de constructeur de classe	Un constructeur ne peut pas être statique, abstrait ou final. • Correction manuelle : Supprimez la propriété Statique, Abstrait ou Final • Correction automatique : Supprime la propriété Statique, Abstrait ou Final

Vérification	Description et correction
Mise en oeuvre d'opération	Lorsqu'il existe une réalisation entre une classe et une interface, vous devez mettre en oeuvre les opérations de l'interface au sein de la classe. Pour ce faire, cliquez sur l'onglet Opérations dans la feuille de propriétés de la classe et sélectionnez le bouton A réaliser en bas de l'onglet pour mettre en oeuvre les opérations manquantes. • Correction manuelle : Réalisez les opérations de l'interface dans la classe • Correction automatique : Aucune
Affectation de nom de rôle	Un rôle navigable sera migré en tant qu'attribut dans une classe. Si le rôle n'a pas de nom, c'est le code de l'association qui est utilisé. • Correction manuelle : Affectez un nom au rôle de l'association • Correction automatique : Aucune
Unicité des noms de rôle	Le nom du rôle est utilisé par un autre rôle ou attribut. • Correction manuelle : Modifiez le nom/code en double • Correction automatique : Ajoute un numéro au nom/code en double
Classe JavaBean sans classe BeanInfo	Les Bean implementors qui fournissent des informations explicites à propos de leurs beans doivent spécifier une classe BeanInfo. • Correction manuelle : Créez une classe BeanInfo • Correction automatique : Aucune
Classe BeanInfo sans classe JavaBean	Un BeanInfo doit dépendre d'une classe JavaBean. • Correction manuelle : Créez une classe JavaBean et recréez son BeanInfo, ou bien supprimez le BeanInfo • Correction automatique : Aucune
Parent de type d'énumération	Un énumération ne peut pas avoir d'enfant. • Correction manuelle : Supprimez les liens vers les classes enfant. • Correction automatique : Aucune
Définition de classe Bean	La classe Bean doit être définie comme publique. Vous devez définir un constructeur public qui ne prend aucun argument et qui ne peut pas définir la méthode finalize(). Il doit être abstrait pour les beans d'entité CMP mais ne peut pas être abstrait ou final pour les beans d'entité CMP, les beans de session et beans commandés par message. • Correction manuelle : Changez la visibilité de la classe en public, définissez un constructeur public sans argument, ne définissez pas la méthode finalize() • Correction automatique : Change la visibilité de la classe en public, définit un constructeur public sans argument et supprime la méthode finalize(). Définit final = false et abstract = false

Vérification	Description et correction
Mise en oeuvre de méthodes de gestion de classe Bean	A chaque méthode définie dans un ou plusieurs composants d'interface doit correspondre une méthode dans la classe Bean avec le même nom, le même type de résultat ainsi que le même nombre et le même type d'arguments. • Correction manuelle : Ajoutez une méthode avec les mêmes nom, numéro, type de résultat et type d'arguments dans la classe Bean • Correction automatique : Ajoute une méthode avec les valeurs appropriées dans la classe Bean
Classe beans : mise en oeuvre de méthodes d'interface Home	A chaque méthode create<METHOD> de la ou des interfaces Home du bean doit correspondre une méthode ejbCreate<METHOD> dans la classe Bean avec les mêmes arguments. A chaque méthode home de la ou des interfaces Home doit correspondre une méthode ebjHome<METHOD> dans la classe Bean avec le même type de résultat ainsi que le même nombre et le même type d'arguments. La vérification suivante s'applique aux *beans d'entité uniquement*. A chaque méthode ejbCreate<METHOD> de la classe Bean, doit correspondre une méthode ejbPostCreate<METHOD> dans la classe Bean avec le même nombre et le même type d'arguments. • Correction manuelle : Ajoutez une méthode avec les mêmes nom et type d'arguments dans la classe Bean • Correction automatique : Ajoute une méthode avec les valeurs appropriées dans la classe Bean La vérification suivante s'applique aux *beans d'entité BMP uniquement*. A chaque méthode finder find<METHOD> définie dans la ou les interfaces Home du bean doit correspondre une méthode ejbFind<METHOD> avec le même type de résultat ainsi que le même nombre et le même type d'arguments. • Correction manuelle : Ajoutez une méthode avec les mêmes nom, numéro, type de résultat et type d'arguments dans la ou les interfaces Home du Bean • Correction automatique : Ajoute une méthode avec les valeurs appropriées dans la ou les interfaces Home du Bean

Vérification	Description et correction
Méthodes ejb-Create de classe Bean	Les méthodes ejbCreate<METHOD> doivent être définies comme public, elles ne peuvent être ni final ni static. La vérification suivante s'applique aux *beans d'entité uniquement*. Le type de résultat d'une méthode ejbCreate() doit être le type clé primaire. • Correction manuelle : Sélectionnez la clé primaire dans la liste Type de résultat de la feuille de propriétés de l'opération • Correction automatique : Sélectionne la clé primaire comme type de résultat La vérification suivante s'applique aux *Session Beans* et aux *Message Driven Beans*. Le type de résultat d'une méthode ejbCreate() doit être void. • Correction manuelle : Sélectionnez void dans la liste Type de résultat de la feuille de propriétés de l'opération • Correction automatique : Change le type de résultat au profit de la valeur void La vérification suivante s'applique aux *Message Driven Beans* uniquement. La classe Bean doit définir une méthode ejbCreate() qui ne prend aucun argument. • Correction manuelle : Ajoutez une méthode sans argument dans la classe Bean • Correction automatique : Ajoute une méthode sans argument dans la classe Bean
Méthodes ejb-PostCreate de classe Bean	La vérification suivante s'applique aux *beans d'entité* uniquement. Les méthodes ejbPostCreate<METHOD> doivent être définies comme publiques et ne peuvent être ni final ni static. Leur type de résultat doit être void. • Correction manuelle : Changez la visibilité de la méthode en public, décochez les cases Final et Statique et sélectionnez void dans la liste Type de résultat dans la feuille de propriétés de l'opération • Correction automatique : Change la visibilité de la méthode en public, supprime les paramètres final et static et change le type de résultat en void

Vérification	Description et correction
Méthodes ejbFind de classe Bean	(Bean d'entité BMP) Les méthodes ejbFind<METHOD> doivent être définies comme publiques et doivent pas être statiques ou finales. Leur type de résultat peut être le type de clé primaire du bean ou une collection de clés primaires. • Correction manuelle : Changez la visibilité de la méthode en public et décochez la case Statique • Correction automatique : Change la visibilité de la méthode en public et supprime les paramètres static et final. Impose le type de clé primaire de comme type de résultat pour ejbFind<METHOD>
Méthodes ejbHome de classe Bean	Les méthodes ejbHome<METHOD> doivent être définies comme publiques et doivent pas être statiques. • Correction manuelle : Changez la visibilité de la méthode en public et décochez la case Statique • Correction automatique : Change la visibilité de la méthode en public et supprime le paramètre static
Méthodes ejbSelect de classe Bean	La vérification suivante s'applique aux *beans d'entité CMP uniquement*. Les méthodes EjbSelect <METHOD> doivent être définies comme publiques et abstraites. Leur clause throws doit inclure l'exception javax.ejb.FinderException. • Correction manuelle : Changez la visibilité de la méthode en public, cochez la case Abstrait et incluez l'exception javax.ejb.FinderException • Correction automatique : Change la visibilité de la méthode en public, spécifie le paramètre abstract et inclut l'exception javax.ejb.FinderException
Définition de classe de clé primaire	La vérification suivante s'applique aux *beans d'entité* uniquement. La classe de clé primaire doit être déclarée comme publique et vous devez définir un constructeur public qui ne prend aucun argument. • Correction manuelle : Changez la visibilité de la méthode en public et ajoutez un constructeur par défaut dans la classe de clé primaire • Correction automatique : Change la visibilité de la méthode en public et ajoute un constructeur par défaut dans la classe de clé primaire

Vérification	Description et correction
Attributs de classe de clé primaire	Tous les attributs de classe de clé primaire doivent être déclarés comme public. En outre, chaque classe de clé primaire avoir un champ cmp correspondant dans la classe Bean. • Correction manuelle : Changez la visibilité en public, et créez un champ cmp dans la classe Bean avec le même nom et le même type de données que l'attribut de la classe de clé primaire • Correction automatique : Change la visibilité en public, et crée un champ cmp dans la classe Bean avec le même nom et le même type de données que l'attribut de la classe de clé primaire
Existence de classe de clé primaire	Si la classe bean comporte plusieurs attributs de clé primaire, une classe de clé primaire doit exister. Si la classe bean ne comporte qu'un seul attribut de clé primaire, elle ne peut pas avoir un type de données standard, mais doit avoir un type de données objet (par exemple : java.lang.Long). • Correction manuelle : S'il existe plusieurs attributs de clé primaire, créez une classe de clé primaire. S'il n'existe qu'un seul attribut de clé primaire, sélectionnez un type de données objet/classificateur • Correction automatique : Crée une classe de clé primaire, ou bien sélectionne le type de données objet/classificateur approprié
Correspondance de classe non définie	La classe doit être mise en correspondance avec des tables ou des vues dans la source de données. • Correction manuelle : Définissez la correspondance sur l'onglet Correspondances de la feuille de propriétés de classe (onglet Sources de la classe), ou supprimez la source de données • Correction automatique : Supprime la source de données de la liste Correspond à dans l'onglet Correspondances de la classe Pour plus d'informations sur les correspondances O/R, voir *Guide des fonctionnalités générales > Liaison et synchronisation de modèles > Mise en correspondance d'objets > Correspondances objet-relationnel (O/R)*.
Correspondance d'attribut non définie	L'attribut doit être mis en correspondance avec des colonnes dans la source de données. • Correction manuelle : Définissez la correspondance à partir de l'onglet Correspondance de la feuille de propriétés de classe (onglet Correspondances d'attribut), ou bien supprimez la source de données • Correction automatique : Supprime la source de données dans la liste Correspond à dans l'onglet Correspondances de la feuille de propriétés de classe Pour plus d'informations sur les correspondances O/R, voir *Guide des fonctionnalités générales > Liaison et synchronisation de modèles > Mise en correspondance d'objets > Correspondances objet-relationnel (O/R)*.

Vérification	Description et correction
Classificateur lié incomplet	Un classificateur de type "Lié" doit être lié à un classificateur générique. • Correction manuelle : Spécifiez un classificateur générique dans la zone située à droite de la liste Type sur l'onglet Général de la feuille de propriétés du classificateur lié. Vous pouvez également le connecter à un classificateur générique par le biais d'une dépendance ayant le stéréotype <<bind>>. • Correction automatique : Aucune
Mode de génération invalide	Si le mode de persistance d'une classe est défini comme Migrer les colonnes, elle doit avoir un parent ou enfant persistant vers lequel migrer les colonnes. • Correction manuelle : Liez la classe à un parent ou enfant persistant, ou changez son mode de persistance sur l'onglet Détails de sa feuille de propriétés. • Correction automatique : Aucune

Vérification des identifiants

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des identifiants.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.

Vérification	Description et correction
Existence d'attribut	Les identifiants doivent avoir au moins un attribut. • Correction manuelle : Ajoutez un attribut à l'identifiant, ou supprimez l'identifiant • Correction automatique : Aucune
Inclusion d'identifiant	Deux identifiants ne doit pas utiliser le même attribut. • Correction manuelle : Supprimez l'identifiant superflu • Correction automatique : Aucune

Vérification des interfaces

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des interfaces.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Classificateur vide	Les attributs et opérations sont manquants pour ce classificateur. • Correction manuelle : Ajouter des attributs ou des opérations à ce classificateur • Correction automatique : Aucune

Vérification	Description et correction
Visibilité de classificateur	Un classificateur ayant la visibilité Private ou Protected doit être interne à un autre classificateur. • Correction manuelle : Changez la visibilité du classificateur en Public ou Package • Correction automatique : Change la visibilité du classificateur en Public ou Package
Constructeur d'interface	Une interface ne peut pas être instanciée, vous ne devez donc pas définir de constructeur pour une interface. • Correction manuelle : Supprimez le constructeur • Correction automatique : Aucune
Navigabilité d'interface	La navigation n'est pas admise pour une interface. • Correction manuelle : Supprimez la navigabilité du côté classe de l'association • Correction automatique : Supprime la navigabilité du côté classe de l'association
Méthodes create d'interface Home	Le type de résultat pour les méthodes create<METHOD> doit être le type d'interface de composant du bean. La clause throws doit inclure l'exception javax.ejb.CreateException ainsi que toutes les exceptions définies dans la clause throws de la méthode ejbCreate<METHOD> et des méthodes ejbPostCreate<METHOD> correspondantes de la classe Bean. • Correction manuelle : Incluez javax.ejb.CreateException et toutes les exceptions définies dans la clause throws de la méthode ejbCreate<METHOD> et des méthodes ejbPostCreate<METHOD> correspondantes de la classe Bean, ou supprimez les exceptions dans la méthode ejbPostCreate<METHOD> • Correction automatique : Inclut javax.ejb.CreateException et toutes les exceptions définies dans la clause throws de la méthode ejbCreate<METHOD> et des méthodes ejbPostCreate<METHOD> correspondantes de la classe Bean

Vérification	Description et correction
Méthodes finder d'interface Home	Le type de résultat pour les méthodes find<METHOD> doit être le type d'interface de composant du bean (pour un finder portant sur un seul objet) ou une collection de clés primaires (pour un finder portant sur plusieurs objets). La clause throws doit inclure javax.ejb.FinderException. • Correction manuelle : Incluez javax.ejb.FinderException dans la clause throws • Correction automatique : Inclut javax.ejb.FinderException dans la clause throws et définit le Type de résultat comme type d'interface du composant La vérification suivante s'applique aux *Beans d'entité BPM uniquement*. La clause throws doit inclure toutes les exceptions définies dans la clause throws des méthodes ejbFind<METHOD> correspondantes de la classe Bean. • Correction manuelle : Incluez toutes les exceptions définies dans la clause throws des méthodes ejbFind<METHOD> de la classe Bean, ou supprimez les exceptions de la méthode ejbFind<METHOD> • Correction automatique : Inclut toutes les exceptions définies dans la clause throws des méthodes ejbFind<METHOD> de la classe Bean
Méthodes d'interface Remote Home	La clause throws des méthodes de l'interface Remote Home doit inclure java.rmi.RemoteException. • Correction manuelle : Incluez java.rmi.RemoteException • Correction automatique : Inclut java.rmi.RemoteException
Méthodes de gestion d'interface de composant	La clause throws des méthodes de gestion d'interface de composant doit inclure toutes les exceptions définies dans les clauses throws de la méthode correspondante dans la classe Bean. La clause throws des méthodes d'interface Remote doit inclure java.rmi.RemoteException. • Correction manuelle : Incluez java.rmi.RemoteException • Correction automatique : Inclure java.rmi.RemoteException
Classificateur lié incomplet	Un classificateur de type "Lié" doit être lié à un classificateur générique. • Correction manuelle : Spécifiez un classificateur générique dans la zone située à droite de la liste Type sur l'onglet Général de la feuille de propriétés du classificateur lié. Vous pouvez également le connecter à un classificateur générique par le biais d'une dépendance ayant le stéréotype <<bind>>. • Correction automatique : Aucune

Vérification des attributs de classe et d'interface

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des des attributs de classe et d'interface.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Détection d'incohérences entre les paramètres de contrôle	Les valeurs spécifiées dans l'onglet Paramètres de contrôle ne sont pas cohérentes pour les types de données numériques et de chaîne : la valeur de défaut n'est pas compatible avec les valeurs minimum et maximum, elle n'appartient pas à la liste des valeurs spécifiée ou bien la valeur minimale est supérieure à la valeur maximale. Les paramètres de contrôle doivent avoir été définis de façon cohérente. • Correction manuelle : Modifiez les valeurs de défaut ou les valeurs min max pour restaurer la cohérence • Correction automatique : Aucune
Affectation d'un type de données	Le type de données d'un attribut doit être défini. En outre, un attribut ne doit pas être VOID. • Correction manuelle : Affectez un type de données valide à l'attribut • Correction automatique : Aucune
Valeur initiale pour un attribut final	L'attribut final d'un classificateur doit être initialisé. • Correction manuelle : Définissez une valeur par défaut pour l'attribut final • Correction automatique : Aucune

Vérification	Description et correction
Divergence vis-à-vis du domaine	Une définition d'attribut n'est pas conforme à la définition du domaine qui lui est attaché. • Correction manuelle : Modifiez le type d'attribut de façon à respecter la définition du domaine • Correction automatique : Corrige le type d'attribut pour éviter une divergence vis-à-vis du domaine Pour plus d'informations sur les divergences vis-à-vis des domaines, voir *Définition des options de modèle de MOO* à la page 11.
Type de données de paramètre d'événement	[VB 2005] Un attribut d'interface ayant un stéréotype Event doit avoir un délégué comme type de données. • Correction manuelle : Définissez le délégué approprié comme type de données • Correction automatique : Aucune

Vérification des opérations de classe et d'interface

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des opérations de classe et d'interface.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.

Vérification	Description et correction
Affectation d'un type de résultat	Le type de résultat d'une opération doit être défini. • Correction manuelle : Affectez un type de résultat à l'opération • Correction automatique : Affecte un type de résultat 'void' à l'opération
Affectation de type de données à un paramètre	Le type de données d'un paramètre doit être défini. En outre, un paramètre ne doit pas être VOID. • Correction manuelle : Choisissez un type de données valide pour le paramètre • Correction automatique : Aucune
Corps d'opération abstraite	[classes] Les opérations abstraites contenues dans une classe ne doivent pas être mises en oeuvre. • Correction manuelle : Supprimez le corps ou la propriété abstraite pour l'opération • Correction automatique : Aucune
Opération abstraite dans une classe non abstraite	[classes] Les opérations abstraites doivent être déclarées uniquement dans des classes abstraites. • Correction manuelle : Définissez la classe comme abstraite, ou supprimez la propriété Abstrait de l'opération • Correction automatique : Supprime la propriété Abstrait dans la feuille de propriétés de l'opération
Surcharge de signature d'opération	[classes] Des opérations surchargées ayant le même nom et le même type de données de paramètres ne peuvent pas avoir un type de résultat différent dans une classe. La surcharge d'opération consiste à utiliser le même nom de méthode en effectuant des opérations différentes en fonction du nombre ou du type de paramètres. • Correction manuelle : Changez le nom de l'opération, le type de données de paramètre ou le type de résultat • Correction automatique : Aucune
Redéfinition d'opérations	[classes] Lorsque vous redéfinissez une opération parent dans une classe, vous ne pouvez pas changer ses modificateurs. La redéfinition d'une opération signifie qu'une opération définie dans une classe particulière est redéfinie dans une classe enfant, dans ce cas l'opération de la classe parent est dite redéfinie. • Correction manuelle : Conservez les mêmes modificateurs pour l'opération enfant • Correction automatique : Aucune

Vérification	Description et correction
Enum : les constantes doivent surcharger une méthode abstraite	[classes] Vous pouvez donner un comportement différent à chaque constante enum en déclarant une méthode abstraite dans le type enum et en la surchargeant avec une méthode concrète pour chaque constante. Dans ce cas, chaque constante doit surcharger la méthode abstraite. • Correction manuelle : Assurez-vous que chaque constante soit associée à une méthode concrète qui surcharge la méthode abstraite. • Correction automatique : Aucune

Vérification des réalisations

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des réalisations.

Vérification	Description et correction
Réalisations redondantes	Une seule réalisation est nécessaire pour une instance de deux objets. • Correction manuelle : Supprimez les réalisations redondantes • Correction automatique : Aucune
Paramètres de type enfant manquants pour une réalisation générique	Un enfant d'un classificateur générique doit résoudre tous les paramètres de type définis par son parent. • Correction manuelle : Résolvez les paramètres de type manquants. • Correction automatique : Aucun.
L'enfant ne peut être lié pour une réalisation générique	Un classificateur lié ne peut pas être l'enfant d'un classificateur autre que son parent générique. • Correction manuelle : Supprimez les liens supplémentaires. • Correction automatique : Aucune.

Vérification des généralisations

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des généralisations.

Vérification	Description et correction
Généralisations redondantes	Deux classes ou interfaces ne peuvent pas être liées par plusieurs généralisations. • Correction manuelle : Supprimez les généralisations redondantes • Correction automatique : Aucune

Vérification	Description et correction
Héritages multiples de classes	La vérification suivante ne concerne que *Java* et *PowerBuilder*. UML admet les héritages multiples, mais Java ne les admet pas. • Correction manuelle : Ne conservez qu'un seul héritage • Correction automatique : Aucune
Extension de classe finale	Une classe finale ne peut pas être étendue. • Correction manuelle : Supprimez le lien de généralisation, ou retirez la propriété Final dans une classe parent • Correction automatique : Aucune
Attribut discriminant non persistant	Si une généralisation a un attribut discriminant, l'attribut doit être marqué comme persistant. • Correction manuelle : Cochez la case Persistant sur l'onglet Détails de la feuille de propriétés de l'attribut discriminant. • Correction automatique : Aucune
Générique : Paramètres de type enfant manquants	Un enfant d'un classificateur générique doit résoudre tous les paramètres de type définis par son parent. • Correction manuelle : Résolvez les paramètres de type manquants. • Correction automatique : Aucun.
Générique : L'enfant ne peut pas être lié	Un classificateur lié ne peut pas être l'enfant d'un classificateur autre que son parent générique. • Correction manuelle : Supprimez les liens supplémentaires. • Correction automatique : Aucune.

Vérification des objets

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des objets.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune

Vérification	Description et correction
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Objet isolé	Un objet ne doit pas être isolé dans le modèle. • Correction manuelle : Créez une relation vers ou depuis l'objet. Il peut s'agir d'un message, d'un lien entre objets ou d'une dépendance *ou* Liez l'objet à un noeud d'objet dans le diagramme d'activités • Correction automatique : Aucune Remarque : la fonctionnalité de vérification de modèle prend en compte l'objet, mais pas le symbole d'objet pour effectuer cette vérification. Si l'objet est déjà associé à un lien d'instance ou à un noeud d'objet dans votre modèle, la vérification ne renvoie pas de message d'erreur.

Vérification des liens entre objets

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des liens entre objets.

Vérification	Description et correction
Liens entre objets redondants	Deux liens entre objets entre les deux mêmes objets ne doivent pas avoir la même association. • Correction manuelle : Supprimez l'un des liens entre objets redondants • Correction automatique : Aucune

Vérification des messages

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des messages.

Vérification	Description et correction
Message dépourvu de numéro d'ordre	Un message doit être doté d'un numéro d'ordre. • Correction manuelle : Spécifiez un numéro d'ordre pour le message • Correction automatique : Aucune
Message utilisé par plusieurs liens entre objets	Un message ne doit pas être attaché à plusieurs liens entre objets. • Correction manuelle : Détachez le message du lien entre objets • Correction automatique : Aucune
Message entre acteurs	Un acteur ne peut pas envoyer un message à un autre acteur dans le modèle. Les messages ne sont admis qu'entre objets, et entre des objets et des acteurs. • Correction manuelle : Créez un message entre deux objets ou bien entre un acteur et un objet • Correction automatique : Aucune

Vérification des états

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des états.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.

Vérification	Description et correction
Unicité du nom/ code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Transition entrante manquante	Chaque état doit être doté d'au moins une transition entrante. Un état sans transition entrante ne peut pas être atteint. • Correction manuelle : Ajoutez une transition liée à l'état • Correction automatique : Aucune
Etat composite dépourvu de début	Un état composite détaille le comportement d'un état dans un diagramme de sous-état. Pour être complet, ce diagramme de sous-état doit comporter un début. • Correction manuelle : Ajoutez un début dans le diagramme de sous-état, ou désélectionnez la case Composite dans la feuille de propriétés de l'état • Correction automatique : Aucune
Ordre des actions incorrect	Les événements entry doivent être les premiers dans la liste des actions d'un état. Les événements exit doivent se trouver à la fin de la liste. L'emplacement des autres actions est libre. • Correction manuelle : Déplacez les événements entry en haut de la liste et les événements exit en fin de liste • Correction automatique : Déplace les événements entry en haut de la liste et les événements exit en fin de liste

Vérification des actions d'état

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des actions d'état.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune

Vérification	Description et correction
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Evénement déclencheur non spécifié	Vous devez spécifier un événement déclencheur pour chaque action portant sur un état. Cet événement déclencheur indique si l'action est exécutée. Remarquez que cette vérification ne s'applique pas aux actions définies sur les transitions car ces dernières sont dotées d'un événement implicite correspondance à la fin de l'exécution des actions internes (de l'état source). • Correction manuelle : Spécifiez un événement déclencheur dans la feuille de propriétés de l'action • Correction automatique : Aucune
Occurrence dupliquée	Deux actions distinctes d'un même état ne doivent pas se produire simultanément. L'occurrence d'une action est définie par la combinaison d'un événement déclencheur et d'une condition. • Correction manuelle : Changez l'événement déclencheur ou la condition de l'action • Correction automatique : Aucune

Vérification des événements

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des événements.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune

Vérification	Description et correction
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Evénement non utilisé	Un événement permet de déclencher une action définie sur un état ou sur une transition. Un événement isolé est inutile. • Correction manuelle : Supprimez l'événement ou utilisez-le dans une action sur un état ou une transition • Correction automatique : Supprime l'événement

Vérification des points de jonction

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des points de jonction.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.

Vérification	Description et correction
Point de jonction incomplet	Un point de jonction représente une séparation ou une fusion de chemins de transition. Il doit donc compter au moins une transition entrante et une transition sortante. • Correction manuelle : Ajouter les transitions manquantes au point de jonction • Correction automatique : Aucune

Vérification des activités

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des activités.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Transition entrante ou sortante absente	Chaque activité doit comporter au moins une transition entrante et une transition sortante. • Correction manuelle : Ajoutez une transition liée à l'activité • Correction automatique : Aucune

Vérification	Description et correction
Activité composite dépourvue de début	Une activité composite détaille l'exécution de l'activité dans un diagramme de sous-activités. Pour être complet, ce diagramme de sous-activité doit comporter un début relié à d'autres activités, ou doit commencer par un début. • Correction manuelle : Ajoutez un début dans le diagramme de sous-activité, ou désélectionnez la case Composite dans la feuille de propriétés de l'activité • Correction automatique : Aucune
Appels d'activité non-réutilisable	Seules les activités ayant un type d'action <undefined> ou Reusable activity peuvent être réutilisés par d'autres activités ayant le type d'action Call, Accept Call ou Reply Call. • Correction manuelle : Changez le type d'action de l'activité référencée, ou supprimer toutes référence à cette activité. • Correction automatique : Aucune

Vérification des décisions

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des décisions.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.

Vérification	Description et correction
Décision incomplète	Une décision représente une branche conditionnelle dans laquelle une transition entrante unique est scindée en plusieurs transitions sortantes, ou représente une fusion lorsque plusieurs transitions entrantes sont fusionnées en une transition sortante unique. Une décision doit donc comporter plusieurs transitions entrantes ou plusieurs transitions sortantes. • Correction manuelle : Ajoutez les transitions manquantes sur la décision • Correction automatique : Aucune

Vérification des noeuds d'objet

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des noeuds d'objet.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Objet non défini pour un noeud d'objet	Un noeud d'objet représente un état particulier d'un objet. Cet noeud d'objet doit donc être lié à un objet. • Correction manuelle : Dans la feuille de propriétés du noeud d'objet, sélectionnez ou créez un objet dans la liste Objet • Correction automatique : Aucune

Vérification	Description et correction
Type données de noeud d'objet	Un noeud d'objet ne transporte aucune information s'il est dépourvu de type de données. • Correction manuelle: Sélectionnez un type de données dans la feuille de propriétés du noeud d'objet. • Correction automatique : Aucune

Vérification des unités d'organisation

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des unités d'organisation.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Dépendance circulaire	L'unité d'organisation ne peut pas être son propre parent ou être parent d'une autre unité d'organisation. • Correction manuelle : Changez d'unité d'organisation dans la zone Parent de la feuille de propriétés de l'unité d'organisation • Correction automatique : Aucune

Vérification des débuts et des fins

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des débuts et des fins.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Transition manquante	Un début ou une fin doit être lié à un objet du diagramme d'activités ou d'états-transitions. • Correction manuelle : Créez une transition entre partir du début/depuis la fin • Correction automatique : Aucune

Vérification des synchronisations

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des synchronisations.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Synchronisation incomplète	Une synchronisation représente un embranchement dans lequel une transition entrante unique est scindée en plusieurs transitions sortantes exécutées en parallèle, ou bien la jonction de plusieurs transitions entrantes qui se rejoignent et attendent que toutes aient atteint le point de jonction avant de poursuivre sous forme d'une seule transition sortante. Une synchronisation doit donc comporter plusieurs transitions entrantes ou bien plusieurs transitions sortantes. • Correction manuelle : Ajoutez les transitions manquantes à la synchronisation • Correction automatique : Aucune

Vérification des transitions et des flux

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des transitions et des flux.

Vérification	Description et correction
Transition/flux dépourvu(e) de source ou de destination	La transition ou le flux ne comporte pas de source ou de destination. Sélectionnez ou créez un objet servant de source ou de destination. • Correction manuelle : Affectez une source ou une destination à la transition/au flux • Correction automatique : Aucune
Condition inutile	S'il n'existe qu'une seule transition sortante ou un seul flux sortant, l'existence d'une condition ou d'un type sur cette transition ou ce flux n'est pas justifiée. • Correction manuelle : Supprimez la condition ou le type de flux inutile ou créez une autre transition avec une autre condition ou type de flux • Correction automatique : Aucune
Condition manquante	Si un objet a plusieurs transitions/flux sortant(e)s, ou si la transition ou le flux est de type réflexif, chaque transition/flux doit comporter une condition. Dans un diagramme d'états-transitions, une transition doit comporter un événement ou une condition. • Correction manuelle : Définissez une condition sur la transition, ou bien créez une synchronisation pour spécifier une exécution en parallèle • Correction automatique : Aucune
Transition dupliquée entre états/ Flux dupliqué entre activités	Deux transitions parallèles (ayant les mêmes extrémités) ne peuvent se produire simultanément, mais doivent plutôt être gouvernées par des conditions (et, dans le cas des transitions, d'événements déclencheur). • Correction manuelle : Changez l'événement déclencheur (diagramme d'états-transitions uniquement) ou la condition de l'une des transition • Correction automatique : Aucune

Vérification des composants

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des composants.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Composant isolé	Un composant ne doit pas être isolé dans le modèle. Il doit être lié à une classe ou à une interface. • Correction manuelle : Attachez certaines classes ou interface à ce composant • Correction automatique : Aucune
Classificateurs attachés à un composant EJB	Les beans d'entité et de session doivent fournir une vue client Remote ou local, ou les deux. • Correction manuelle : Complétez la ou les vues existantes, ou bien créez une vue distante si aucune interface n'a été exposée • Correction automatique : Complète la ou les vues existantes, ou bien crée une vue distante si aucune interface n'a été exposée

Vérification	Description et correction
Redéfinition de message SOAP	Vous ne pouvez pas avoir le même type de données d'entrée et de sortie SOAP au sein d'un même composant. • Correction manuelle : Changez le nom du type de données d'entrée SOAP ou celui du type de données de sortie SOAP dans l'onglet Entrée SOAP ou Sortie SOAP de la feuille de propriétés de l'opération • Correction automatique : Aucune La définition des types de données SOAP est disponible dans la partie schéma du WSDL que vous pouvez afficher dans l'onglet WSDL de la feuille de propriétés d'un composant.

Vérification des noeuds

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des noeuds.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Noeud vide	Un noeud est considéré comme vide lorsqu'il ne contient aucune instance de composant : • Correction manuelle : Ajoutez au moins une instance de composant au noeud • Correction automatique : Aucune

Vérification des formats de données

PowerAMC fournit des vérifications par défaut afin de contrôler la validité des formats de données.

Vérification	Description and Correction
Expression vide	Les formats de données doivent avoir une valeur dans la zone **Expression**. • Correction manuelle : Spécifiez une expression pour le format de données. • Correction automatique : Aucune

Vérification des instances de composant

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des instances de composant.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Instance de composant dépourvue de composant	Une instance de composant a été créée mais aucun composant ne lui est attaché. Vous devez l'attacher à un composant. • Correction manuelle : Attachez un composant existant à une instance de composant, ou bien créez un nouveau composant en sélectionnant le bouton Créer dans la feuille de propriétés de l'instance de composant • Correction automatique : Aucune

Vérification	Description et correction
Instances de composant en double	Un noeud contient plusieurs instances du même composant. • Correction manuelle : Supprimez l'instance de composant en double ou attachez-la au composant approprié • Correction automatique : Aucune
Instance de composant isolée	Une instance de composant ne doit pas être créée hors d'un noeud. Elle ne sera pas déployée. • Correction manuelle : Attachez-la à un noeud • Correction automatique : Aucune

Vérification des références d'interaction

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des références d'interaction.

Vérification	Description et correction
Diagramme référencé manquant	Une référence d'interaction doit faire référence à un diagramme de séquence. • Correction manuelle : Afficher la feuille de propriétés de la référence d'interaction et spécifiez le diagramme de séquence à référencer. • Correction automatique : Aucune.

Vérification	Description et correction
Cohérence des lignes de vie attachées	Le symbole de référence d'interaction a une liste de lignes de vie attachées, qui correspondent aux acteurs et aux objets. Ces acteurs et objets doivent correspondre à un sous-ensemble de ceux affichés dans le diagramme de séquence référencé. Cela correspond à un sous-ensemble car certaines lignes de vies dans le diagramme référencé ne peuvent pas être affichées dans le diagramme de la référence d'interaction. La vérification courante s'assure du respect des règles de cohérence suivantes : Le nombre de lignes de vie attachées ne peut pas être supérieur au nombre de lignes de vies existant dans le diagramme référencé Si une ligne de vie attachée correspond à un objet, et si cet objet a une métaclasse associée, le diagramme de séquence référencé doit contenir au moins un objet associé à cette métaclasse • Correction manuelle : Changez la liste des lignes de vies attachées pour la référence d'interaction. Pour ce faire, il vous suffit de redimensionner le symbole de la référence d'interaction ou de cliquer avec le pointeur sur l'intersection du symbole de la référence d'interaction et de la ligne de vie. Le curseur de l'outil change sur cette zone et vous permet de détacher la ligne de vie de la référence d'interaction (ou de l'attacher à cette dernière). • Correction automatique : Aucune.
Trop de messages entrants pour la référence	La référence d'interaction a plus de messages entrants que de messages sortants. • Correction manuelle : Supprimez des messages entrants ou ajoutez des messages sortants jusqu'à ce que leur nombre s'équilibre. • Correction automatique : Aucune
Trop de messages sortants pour la référence	La référence d'interaction a plus de messages sortants que de messages entrants. • Correction manuelle : Supprimez des messages sortants ou ajoutez des messages entrants jusqu'à ce que leur nombre s'équilibre. • Correction automatique : Aucune

Vérification des parties de classe

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des parties de classe.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Type de classificateur de partie de classe	Une partie de classe doit avoir un type de données qui est un classificateur lié à son classificateur propriétaire par le biais d'une association. • Correction manuelle : Spécifiez un type de données pour la partie et connectez le classificateur approprié à son classificateur propriétaire. • Correction automatique : Aucune
Type d'association de partie de classe	La propriété Composition d'une partie doit correspondre au type de l'association entre son propriétaire et son type de données. • Correction manuelle : Activez ou désactivez la propriété Composition. • Correction automatique : La propriété Composition est activée ou désactivée.

Vérification des ports de classe et de composant

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des ports de classe et de composant.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Ports de classe ou de composant isolés	Les ports de classe et de composant doivent avoir un type de données, ou avoir une interface fournie ou requise. • Correction manuelle : Spécifiez un type de données ou une interface. • Correction automatique : Aucune

Vérification des connecteurs d'assemblage de classe ou de composant

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des connecteurs d'assemblage de classe ou de composant.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.
Unicité du nom/code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.
Connecteur d'interface nul pour un connecteur d'assemblage de composant	Une interface doit être définie pour chaque connecteur d'assemblage. • Correction manuelle : Définissez une interface pour le connecteur d'assemblage. • Correction automatique : Aucune.
Interfaces pour un connecteur d'assemblage de composant	L'interface définie pour un connecteur d'assemblage doit être fournie par le fournisseur et requise par le client. • Correction manuelle : Assurez-vous que le fournisseur et le client soient correctement définis. • Correction automatique : Aucune.

Vérification des associations

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des associations.

Vérification	Description et correction
Générique : Paramètres de type enfant maquants	Dans une association navigable, si le parent est générique, l'enfant doit redéfinir tous les paramètres de type du parent. Si le parent est un classificateur partiellement lié (dans lequel certains paramètres de type ne sont pas résolus) l'enfant doit redéfinir tous les paramètres de type non résolus. • Correction manuelle : Résolvez les paramètres de type manquants. • Correction automatique : Aucune.
Générique : L'enfant ne peut pas être lié	Un classificateur lié ne peut pas être l'enfant d'une association navigable autre que son parent générique. • Correction manuelle : Supprimez les liens supplémentaires. • Correction automatique : Aucune.

Vérification des paramètres d'entrée et de sortie d'activité

PowerAMC fournit des vérifications de modèle par défaut afin de contrôler la validité des paramètres d'entrée et de sortie d'activité.

Vérification	Description et correction
Le nom/code contient des termes qui ne figurent pas dans le glossaire	[si le glossaire est activé] Les noms et les codes ne doivent contenir que des termes approuvés tirés du glossaire. • Correction manuelle - Modifiez le nom ou le code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Aucune
Le nom/code contient des synonymes de termes de glossaire	[si le glossaire est activé] Les noms et les codes ne doivent pas contenir de synonymes de termes de glossaire. • Correction manuelle - Modifiez le nom ou code de sorte qu'il ne contienne que des termes du glossaire. • Correction automatique - Remplace les synonymes par les termes de glossaire qui leur sont associés.

Vérification	Description et correction
Unicité du nom/ code	Les noms d'objet doivent être uniques dans l'espace de noms. • Correction manuelle - Modifiez le nom ou code en double. • Correction automatique - Ajoute un numéro au nom ou code en double.

Chapitre 10

Importation d'un modèle Rational Rose dans un MOO

Vous pouvez importer des modèles Rational Rose (.MDL) créés à l'aide des version 98, 2000 et 2002 dans un MOO.

Remarque : Un modèle Rose peut prendre en charge plusieurs langages tandis qu'un MOO PowerAMC ne peut avoir qu'un seul langage objet. Lorsque vous importez un modèle Rose multi-langage dans un MOO, le MOO n'a qu'un seul des langages objet du modèle Rose. Le tableau suivant montre comment les langages Rose sont convertis en langages PowerAMC :

Langage Rose	Langage PowerAMC
CORBA	IDL-CORBA
Java	Java
C++	C++
VC++	C++
XML-DTD	XML-DTD
Visual Basic	Visual Basic 6
Analysis, Oracle 8, Ada, COM, Web Modeler, et tous les autres langage	Analysis

1. Sélectionnez **Fichier > Importer > Fichier Rational Rose**, puis affichez le répertoire qui contient le fichier Rose.
2. Sélectionnez Fichiers Rational Rose (*.MDL) dans la liste Type de fichier, puis sélectionnez le fichier à importer.
3. Cliquez sur **Ouvrir**.

   Le processus d'importation commence, et le diagramme par défaut du modèle s'affiche dans la zone de travail. Les objets généraux Rose sont importés comme suit :

Objet Rose	Objet de MOO
Package	Package **Remarque :** La propriété Global n'est pas importée.

Objet Rose	Objet de MOO
Diagram	Diagramme
Note	Note
Note link	Lien de note
Text	Texte
File	Fichier

Seules les propriétés suivantes sont importées pour ces objets :

Propriété Rose	Propriété de MOO
Documentation	Commentaire
Zoom	Echelle de page
Stereotype	Stéréotype

Importation de diagrammes de cas d'utilisation (use case diagrams) Rational Rose

PowerAMC peut importer les objets les plus importants des diagrammes de diagrammes de cas d'utilisation (use case diagrams) Rational Rose.

Seules les propriétés répertoriées sont importées :

Objet Rose	Objet de MOO
Class avec un stéréotype <<actor>>, <<business actor>> ou <<business worker>>	Classe de mise en oeuvre contenue par un acteur
Classe avec stéréotype <<boundary>>, <<business entity>>, <<control>>, <<entity>> ou <<table>>	Classe (sans symbole, car les classes ne sont pas admises dans les diagrammes de cas d'utilisation du MOO)
Use case : • Diagrams	Cas d'utilisation : • Diagrammes associés
Association : • Navigable	Association : • Orientation

Remarque : Notez que les dépendances entre un cas d'utilisation (use case) et un acteur (actor) ne sont pas importées.

Importation de diagrammes de classes (class diagrams) Rational Rose

PowerAMC peut importer les objets les plus importants des diagrammes de diagrammes de classes (class diagrams) Rational Rose.

Seules les propriétés répertoriées sont importées :

Objet Rose	Objet de MOO
Class : • 'Class utility' Type • Export Control • Implementation • Cardinality: 0..n, 1..n • Nested class • Persistence • Abstract	Classe : • Type 'Class' • Visibilité • Package • Cardinalité : 0..*, 1..* • Classificateur interne • Persistance • Abstrait
Interface : • Export Control • Implementation • Nested class	Interface : • Visibilité • Package • Classificateur interne
Attribute : • Export Control • Implementation • Initial value • Static • Derived	Attribut : • Visibilité • Package • Valeur initiale • Statique • Dérivé
Operation : • Export Control • Implementation	Opération : • Visibilité • Package

Objet Rose	Objet de MOO
Generalization : • Export Control • Implementation • Virtual inheritance • Multi inheritance	Généralisation : • Visibilité • Package • Attribut étendu • Héritage multiple
Association : • Role name • Export Control • Navigable • Cardinality • Aggregate (class A or B) • Aggregate (by reference) • Aggregate (by value)	Association : • Nom de rôle • Visibilité • Navigable • Multiplicité • Conteneur • Agrégation • Composition
Dependency	Dépendance

Importation de diagrammes de collaboration (collaboration diagrams) Rational Rose

PowerAMC peut importer les objets les plus importants des diagrammes de diagrammes de collaboration (collaboration diagrams) Rational Rose.

Seules les propriétés répertoriées sont importées :

Objet Rose	Objet de MOO
Object ou class instance : • Class • Multiple instances	Objet : • Classe ou interface • Multiple
Link ou object link : • Assoc • Messages list	Lien d'instance : • Association • Messages
Actor	Acteur

Objet Rose	Objet de MOO
Message : • Simple stereotype • Synchronous stereotype • Asynchronous stereotype • Balking	Message : • Indéfini • Appel de procédure • Asynchrone • Condition

Importation de diagrammes de séquence (sequence diagrams) Rational Rose

PowerAMC peut importer les objets les plus importants dans les diagrammes de séquence Rose.

Seules les propriétés répertoriées sont importées :

Objet Rose	Objet de MOO
Object ou class instance : • Persistence • Multiple instances	Objet : • Persistance • Multiple
Actor	Acteur
Message : • Simple stereotype • Synchronous stereotype • Asynchronous stereotype • Balking • Return message • Destruction marker	Message : • Indéfini • Appel de procédure • Asynchrone • Condition • Retour • Message récursif avec une action Destruction

Importation de diagrammes d'états-transitions (statechart diagrams) Rational Rose

PowerAMC peut importer les objets les plus importants des diagrammes de diagrammes d'états-transitions (statechart diagrams) Rational Rose.

Dans Rose, les diagrammes d'activités et d'états-transitions sont créés dans la Use Case View ou la Logical View :

- Au niveau racine
- Dans une activité
- Dans un état

Une UML State Machine est automatiquement créée : elle contient des diagrammes "statechart" (états-transitions) "activity" (diagrammes d'activités) avec leurs objets.

Dans PowerAMC, les diagrammes d'états-transitions sont créés au niveau du modèle ou dans un état composite : le package parent ou le modèle est considéré comme la State Machine UML.

Les diagrammes d'états-transitions Rose qui se trouvent au niveau racine ou dans un état sont importés, mais ceux qui se trouvent dans une activité ne sont pas importés.

Seules les propriétés répertoriées sont importées :

Objet Rose	Objet de MOO
State : • When action • OnEntry action • OnExit action • Do action • OnEvent action • Event action • Event arguments	Etat ou noeud d'objet : • Evénement déclencheur • Entry • Exit • Do • Event • Evénement déclencheur • Arguments d'événement
State transition : • <No name> • <No code> • Event • Arguments • Guard condition • Action	Transition : • Nom calculé • Code calculé • Evénement déclencheur • Arguments d'événement • Condition • Action déclenchante

Importation de diagrammes d'activités (activity diagrams) Rational Rose

PowerAMC peut importer les objets les plus importants des diagrammes de diagrammes d'activités (activity diagrams) Rational Rose.

Seules les propriétés répertoriées sont importées :

Objet Rose	Objet de MOO
Activity : • Actions	Activité : • Action
Object (associé avec un state)	Noeud d'objet
State	Etat (aucun symbole dans le diagramme d'activité)
Start state	Début
Self Transition ou Object Flow	Transition
Synchronization	Synchronisation
Decision	Décision
End state	Fin
Swimlane	Unité d'organisation/couloir

Remarque :

- PowerAMC ne prend pas en charge les actions multiples sur une même activité. A l'issue de l'importation, l'onglet Action de la feuille de propriétés d'une activité dans le MOO affiche <<Undefined>> et la zone de texte reproduit la liste des actions importées.
- PowerAMC ne gère pas les objets subunits de Rose sous forme de fichiers distincts, le programme importe les fichiers *.CAT *.SUB dans le modèle qui leur faire référence. Si un fichier .CAT ou .SUB ne se trouve pas sur le chemin spécifié, PowerAMC recherche dans le répertoire où se trouve le fichier contenant le modèle.
- Dans Rose, vous pouvez associer un Object (instance d'une classe) avec un State. L'Object Rose est importé comme un objet dépourvu de symbole. Si l'Object Rose est associé à un State, un état d'objet avec symbole est créé dans le MOO et contient le nom, le stéréotype et le commentaire du State. Si le diagramme Rose qui contient le symbole de l'Object est une activité composite, PowerAMC crée un raccourci de l'objet dans l'activité composite car le noeud d'objet ne peut pas être décomposé dans un MOO.

Importation de diagrammes de composants (component diagrams) Rational Rose

PowerAMC peut importer les objets les plus importants des diagrammes de diagrammes de composants (component diagrams) Rational Rose.

Seules les propriétés répertoriées sont importées :

Objet Rose	Objet de MOO
Component : • Interface of realize • Class of realize • File • URL file • Declaration	Composant : • Interface • Classe • Fichier externe dans des dépendances étendues • Fichier URL dans les dépendances étendues • Description dans l'onglet Notes

Les types de composants suivants, qui ont des stéréotypes Rose prédéfinis avec des symboles différents, sont importés. Les stéréotypes sont préservés, mais chacun d'entre eux aura un symbole de composant de MOO standard :

- Active X
- Applet
- Application
- DLL
- EXE
- Generic Package
- Generic Subprogram
- Main Program
- Package Body
- Package Specification
- Subprogram Body
- Subprogram Specification
- Task Body
- Task Specification

Importation de diagrammes de déploiement (deployment diagrams) Rational Rose

PowerAMC peut importer les objets les plus importants des diagrammes de déploiement Rational Rose.

Seules les propriétés répertoriées sont importées :

Objet Rose	Objet de MOO
Node : • Device • Processor • Device characteristics • Processor characteristics	Noeud : • Noeud • Noeud • Description • Description
File objects : • File • URL definition	Objets fichier: • Objet fichier lié au package • Objet fichier
Node association : • Connection • Characteristics	Association de noeud : • Association de noeud • Description

Chapitre 11 Importation et exportation d'un MOO dans un format XMI

PowerAMC prend en charge l'importation et l'exportation de fichiers XML Metadata Interchange (XMI) UML v2.x, un format ouvert qui permet de transférer des objets UML entre différents outils. Tous les objets du MOO peuvent être importés et exportés.

Importation de fichiers XMI

PowerAMC prend en charge l'importation d'un fichier XMI dans un MOO. XMI prenant uniquement en charge le transfert des objets, PowerAMC alloue des symboles par défaut aux objets importés et les affecte à des diagrammes par défaut.

1. Sélectionnez **Fichier > Importer > Fichier XMI** pour afficher la boîte de dialogue Nouveau modèle.
2. Sélectionnez un langage objet, spécifiez le premier diagramme du modèle, puis cliquez sur OK.
3. Sélectionnez le fichier .XML ou .XMI à importer, puis cliquez sur **Ouvrir**.

   L'onglet Général de la fenêtre Résultats montre les objets qui sont importés. Une fois l'importation terminée, le premier diagramme spécifié s'affiche dans la fenêtre de diagramme.

Exportation de fichiers XMI

PowerAMC prend en charge l'importation d'un MOO dans un fichier XMI. XMI prenant uniquement en charge le transfert des objets, les objets et symboles PowerAMC associés ne sont pas conservés lors de l'exportation.

PowerAMC exporte vers des objets conteneur UML comme suit :

Conteneur UML	Objet PowerAMC	Diagramme PowerAMC
Activité	Activité composite	Diagramme d'activités
State Machine	Etat composite	Diagramme d'états

Conteneur UML	Objet PowerAMC	Diagramme PowerAMC
Interaction	Package	Diagramme de séquence, Diagramme de communication, ou Diagramme d'interactions

1. Sélectionnez **Fichier > Exporter > Fichier XMI** pour afficher une boîte de dialogue Enregistrer sous standard.
2. Saisissez un nom pour le fichier à exporter, sélectionnez le type approprié, puis cliquez sur Enregistrer

   L'onglet Général de la fenêtre Résultats montre les objets qui sont exportés. Vous pouvez ouvrir le fichier XMI résultant avec n'importe quel outil de modélisation qui prend en charge ce format d'échange ou un générateur de code tel que Java, CORBA, ou C++.

Partie II

Référence des définitions de langages objet

Les chapitres de cette partie fournissent des informations spécifiques aux langages objet pris en charge par PowerAMC.

Chapitre 12 Java

PowerAMC permet la prise en charge pour la modélisation de Java, y compris l'ingénierie par va-et-vient.

Remarque : Pour modéliser pour Java à partir de la v5, sélectionnez Java comme langage cible. La prise en charge des versiond antérieures, via Java 1.x, a été abandonnée.

Pour obtenir des informations spécifiques à la modélisation pour Java dans l'environnement Eclipse, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Le plugin PowerAMC pour Eclipse*.

Classes publiques Java

Java permet de créer plusieurs classes dans un même fichier, mais l'une de ces classes, et une seule, doit être publique.

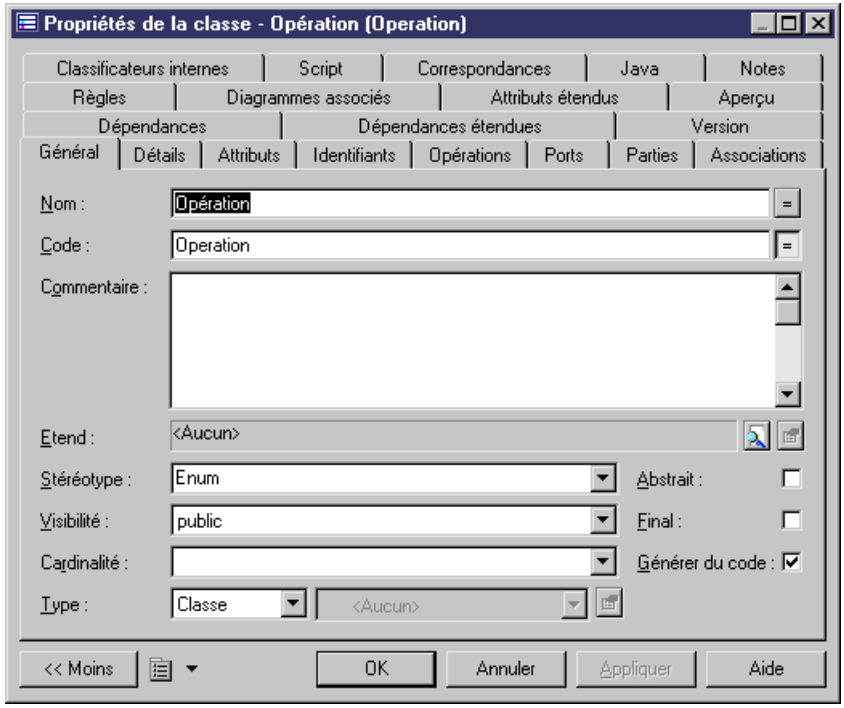
Dans PowerAMC, vous devez créer une classe publique et plusieurs classes dépendantes et tracer des liens de dépendance avec le stéréotype <<sameFile>> entre elles. Ce type de lien est géré lors de la génération et le reverse engineering.

Types énumérés (enums) Java

Java 5 prend en charge les types énumérés. Ces derniers remplacent l'ancienne notion de typesafe, et sont beaucoup plus compacts et plus faciles à maintenir. Ils peuvent être utilisés pour répertorier des collections de valeurs telles que les jours de la semaine ou les suite d'un jeu de cartes, ou tout jeu de constantes fixe, tels que les commandes d'un menu.

Un enum est représenté par une classe ayant le stéréotype <<enum>>.

1. Créez une classe dans un diagramme de classes ou dans un diagramme de structure composite, puis double-cliquez sur le symbole de cette classe pour afficher sa feuille de propriétés.
2. Sur l'onglet Général, sélectionnez <<enum>> dans la liste Stéréotype.

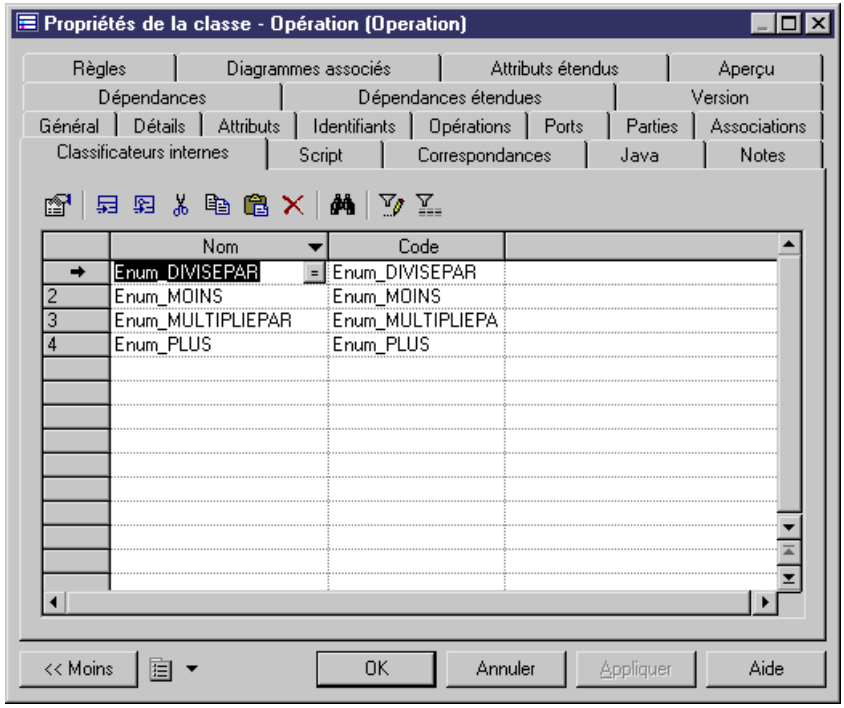


3. Cliquez sur l'onglet Attributs, puis ajoutez autant d'attributs que nécessaire. Ces attributs ont par défaut le type de données EnumConstant. Par exemple, pour créer un type d'enum contenant des opérations mathématiques standard, vous pouvez créer quatre attributs EnumConstant portant les noms "PLUS", "MOINS", "FOIS" et "DIVISEPAR".

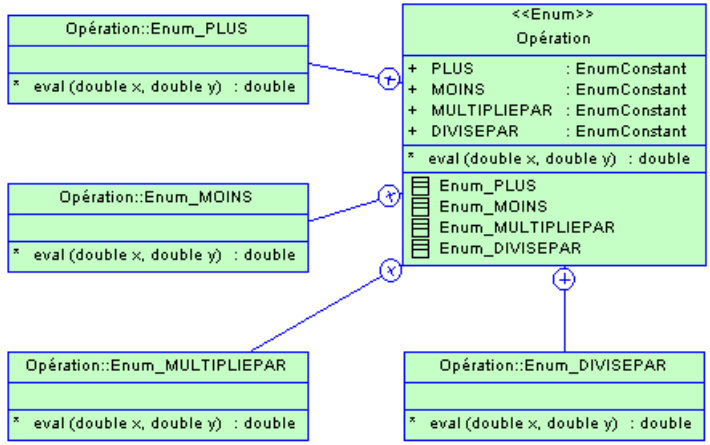
   Remarquez que, compte tenu du fait qu'un enum Java est une classe à part entière, vous pouvez également ajouter d'autres types d'attributs en cliquant sur la colonne Type de données et en sélectionnant un autre type dans la liste.

4. [facultatif] Vous pouvez créer une classe anonyme pour un EnumConstant en sélectionnant la ligne correspondante sur l'onglet Attributs et en cliquant sur l'outil Propriétés pour afficher sa feuille de propriétés. Sur l'onglet Général, cliquez sur l'outil Créer en regard de la classe Enum afin de créer la classe interne.

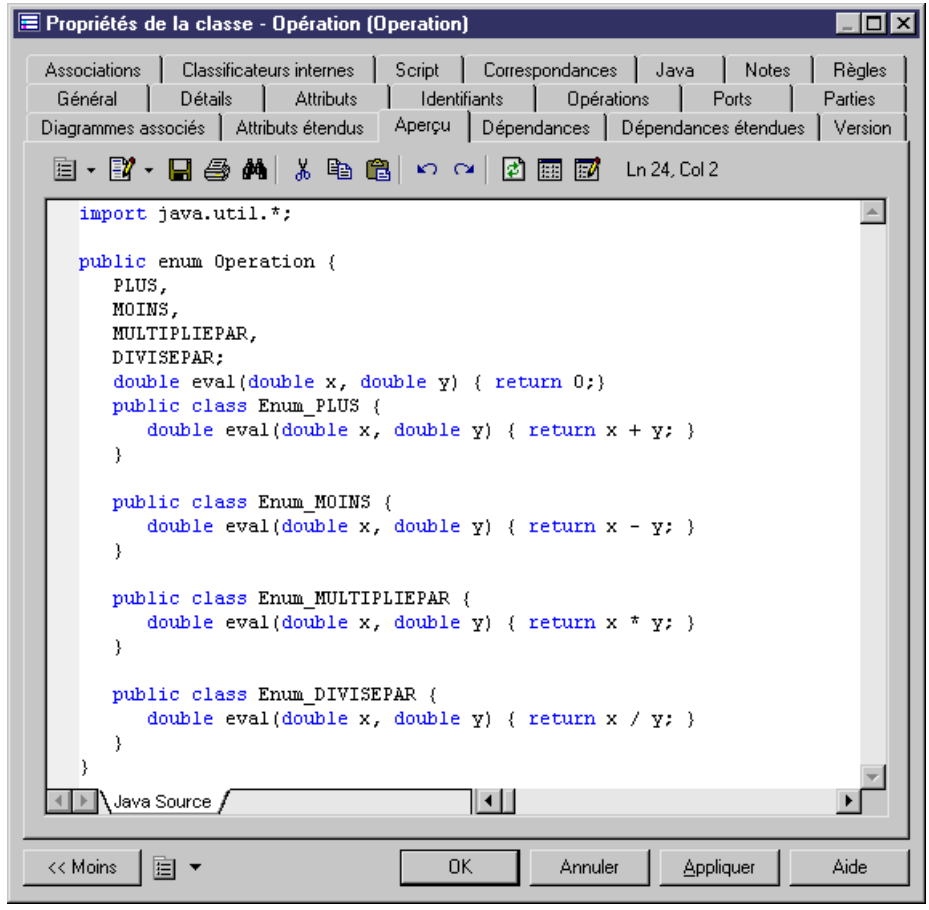
   Ces classes anonymes seront affichées sur l'onglet Classificateurs internes de la feuille de propriétés de classe Enum :



La classe enum Opération, avec une classe anonyme pour chaque EnumConstant afin de permettre différentes opérations arithmétiques, peut être représentée dans un diagrammes de classes comme suit :



Le code équivalent se présente comme suit :



Commentaires Javadoc

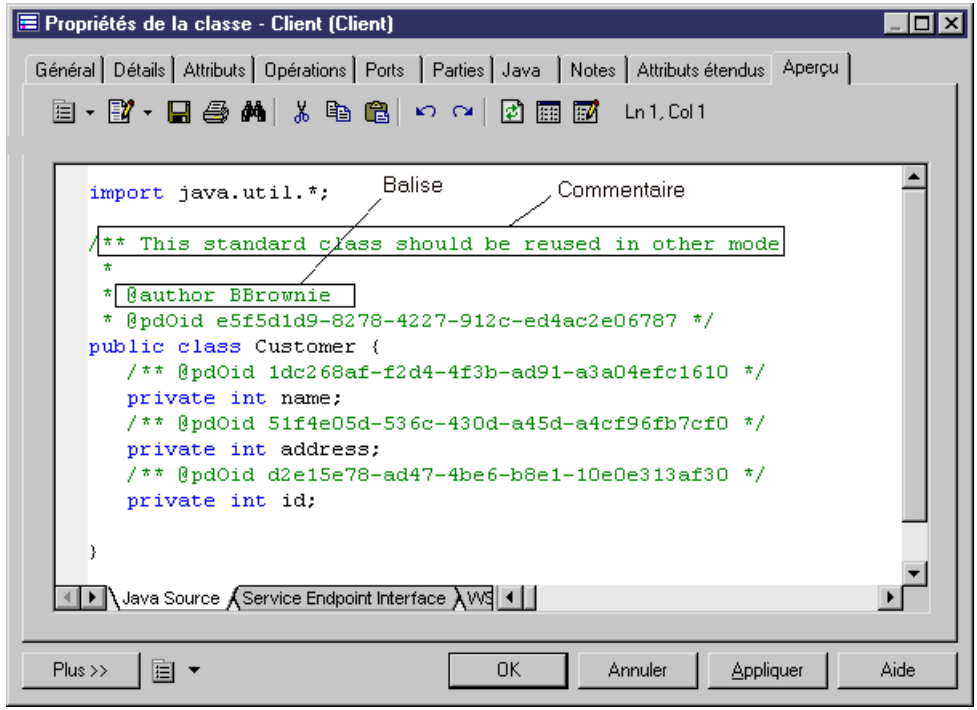
Javadoc est un outil fourni dans le JDK qui analyse les déclarations et commentaires de documentation dans un jeu de fichiers source Java et produit le jeu correspondant de pages HTML décrivant les objets du modèle.

Les commentaires Javadoc sont inclus dans le code source d'un objet, immédiatement avant la déclaration d'un objet, entre /** et */.

Un commentaire Javadoc peut contenir :

- Une description après le délimiteur de début /**. Cette description correspond à un commentaire dans un objet de MOO
- Des balises préfixés par le caractère @

Par exemple, dans l'onglet d'aperçu de code suivant, vous pouvez lire la balise @author, et le commentaire inséré depuis la zone Commentaire de l'onglet Général de la feuille de propriétés de classe.



Le tableau suivant résume la prise en charge des commentaires Javadoc dans PowerAMC :

Javadoc	Description	S'applique à	Attribut étendu correspondant
%comment%	Zone de commentaire. Si les commentaires Javadoc sont introuvables, les commentaires standard sont récupérés à la place.	Classe, interface, opération, attribut	—
@since	Ajoute un en-tête "Since" avec le texte since spécifié dans la documentation générée.	Classe, interface, opération, attribut	Javadoc@since

Javadoc	Description	S'applique à	Attribut étendu correspondant
@deprecated	Ajoute un commentaire indiquant que cette API n'est plus utilisable.	Classe, interface, opération, attribut	Javadoc@deprecated
@author	Ajoute une entrée Author.	Classe, interface	Javadoc@author. Si la balise n'est pas définie, le nom d'utilisateur de l'onglet Version est utilisé. Dans le cas contraire, la valeur définie pour la balise est affichée.
@version	Ajoute une entrée Version, faisant le plus souvent référence à la version du logiciel.	Classe, interface	Javadoc@version
@see	Ajoute un en-tête "See Also" avec un lien ou une entrée de texte qui pointe vers une référence.	Classe, interface, opération, attribut	Javadoc@see
@return	Ajoute une section "Returns" avec le texte descriptif.	Opération	Javadoc@misc
@throws	Ajoute une sous-section "Throws" à la documentation générée.	Opération	Javadoc@misc. Vous pouvez déclarer des exceptions d'opération.
@exception	Ajoute une sous-section "Exception" à la documentation générée.	Opération	Javadoc@misc. Vous pouvez déclarer des exceptions d'opération.
@serialData	Documente les types et l'ordre des données dans le formulaire sérialisé.	Opération	Javadoc@misc

Javadoc	Description	S'applique à	Attribut étendu correspondant
@serialField	Documente un objet ObjectStreamField d'un membre serialPersistentFields d'une classe sérialisable.	Attribut	Javadoc@misc
@serial	Utilisé dans le commentaire pour un champ sérialisable par défaut.	Attribut	Javadoc@misc
@param	Ajoute un paramètre à la section Paramètres.	Opération	Javadoc@misc

Définition de valeurs pour les étiquettes Javadoc

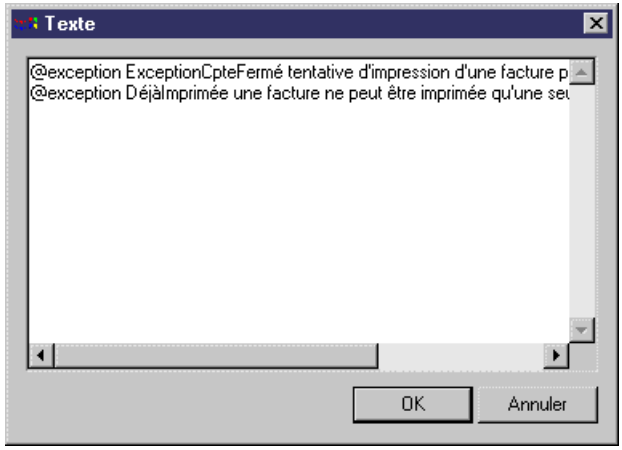
Vous pouvez définir des valeurs pour les étiquettes Javadoc à partir de l'onglet Attributs étendus de la feuille de propriétés d'un objet.

Pour ce faire, vous devez sélectionner une étiquette Javadoc dans la liste des attributs étendus, puis cliquer sur le bouton Points de suspension dans la Colonne Valeur. La boîte de dialogue de saisie qui s'affiche vous permet de créer des valeurs pour l'étiquette Javadoc sélectionnée. Par exemple, si le type de données a pour valeur (Color), vous pouvez sélectionner une autre couleur dans la boîte de dialogue Couleur en cliquant sur le bouton Points de suspension dans la colonne valeur.

Remarque : vous définissez des valeurs pour les commentaires @return, @exception, @throws, @serialData, @serialField, @serial JavaDoc et @param dans l'attribut étendu Javadoc@misc.

Aucune étiquette Javadoc n'est générée si vous ne spécifiez pas de valeur pour les attributs étendus.

N'oubliez pas de répéter l'étiquette Javadoc avant chaque nouvelle valeur, certaines valeurs pouvant comporter plusieurs lignes. Si vous ne répétez pas l'étiquette, les valeurs ne seront pas générées.



Lorsque vous affectez une valeur à une étiquette Javadoc, l'étiquette et sa valeur s'affichent dans l'onglet d'aperçu du code de l'objet.

@author

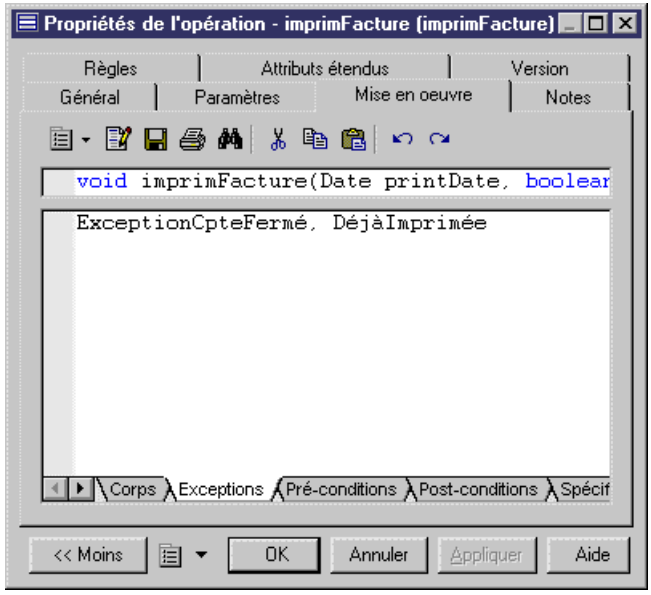
@author n'est pas généré si l'attribut étendu n'a aucune valeur.

@exceptions et @throws

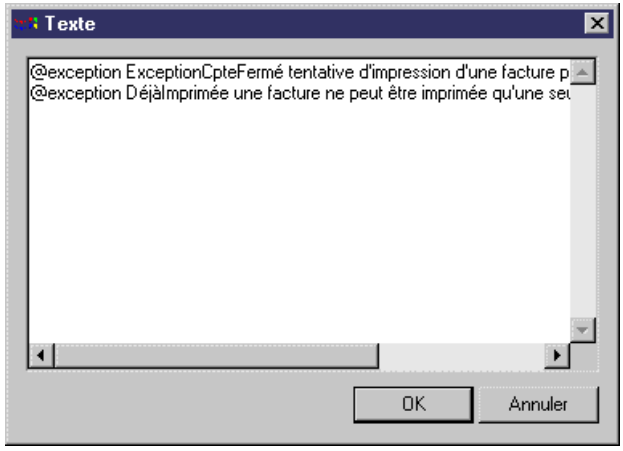
@exceptions et @throws sont des étiquettes Javadoc synonymes utilisées pour définir les exceptions qui peuvent être provoquées par une opération.

Pour utiliser ces étiquettes, procédez comme suit :

- Dans la feuille de propriétés de l'opération, cliquez sur l'onglet Mise en oeuvre, puis sur l'onglet Exceptions pour afficher l'onglet Exceptions. Vous pouvez saisir des exceptions dans cette page.

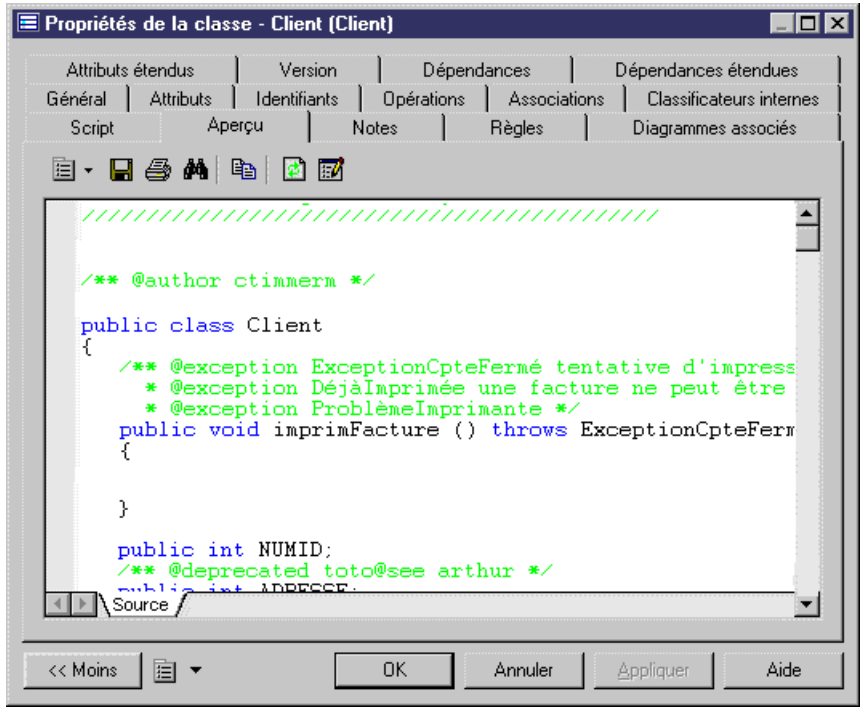


- Dans la même feuille de propriétés, cliquez sur l'onglet Attributs étendus, sélectionnez la ligne Javadoc@exception dans la liste, puis cliquez sur le bouton Points de suspension dans la colonne Valeur. Saisissez des valeurs pour chaque exception déclarée, sans oublier de répéter @exception ou @throws pour chaque exception.



Toutefois, il est également possible de saisir des valeurs directement après les étiquettes @exception ou @throws dans l'onglet Attributs étendus. Ces commentaires décrivent les exceptions qui ne sont pas répertoriées dans les exceptions d'opération, et qui ne sont pas affichées après le paramètre `throws` dans l'onglet d'aperçu du code de la classe.

Lorsque vous affichez un aperçu du code généré, chaque exception s'affiche avec sa valeur :



Génération et reverse engineering de commentaires JavaDoc

Vous récupérez les commentaires Javadoc dans les classes, interfaces, opérations et attributs lors du reverse engineering. Les commentaires Javadoc ne sont récupérés par le reverse engineering que s'ils existent dans le code source.

La fonctionnalité de reverse engineering des commentaires Javadoc est très utile pour l'*ingénierie par va-et-vient* : vous conservez les commentaires Javadoc lors du reverse engineering et vous pouvez régénérer le code en utilisant les commentaires Javadoc préservés. Le processus de génération génère des commentaires d'objet conformes à Javadoc

Pour plus d'informations sur la génération des commentaires Javadoc, voir *Génération et reverse engineering de commentaires JavaDoc* à la page 368.

Annotations Java 5.0

PowerAMC assure la prise en charge complète des annotations Java 5.0, qui permettent d'ajouter des métadonnées dans votre code. Ces métadonnées sont accessibles aux outils de post-traitement ou au moment de l'exécution afin de faire varier le comportement du système.

Vous pouvez utiliser des annotations intégrées, telles que celles répertoriées ci-après, et aussi créer vos propres annotations, à appliquer à vos types.

Il existe trois types d'annotations :

- Annotations normales – elles peuvent accepter plusieurs arguments
- Annotations à membre unique – elles n'acceptent qu'un seul argument, qui a une syntaxe plus compacte
- Annotations marqueur – elles n'utilisent pas de paramètre et sont utilisées pour indiquer au compilateur Java les modalités de traitement spécifique d'un élément

PowerAMC supporte les sept annotations intégrées de Java 5.0 :

- java.lang.Override - spécifie qu'une déclaration de méthode va redéfinir une déclaration de méthode effectuée dans une classe parent, et va générer une erreur à la compilation dans le cas contraire.
- java.lang.Deprecated – spécifie qu'un élément est déprécié, et génère un avertissement à la compilation si elle est utilisée dans du code non déprécié.
- java.lang.SuppressWarning – spécifie les avertissements à la compilation qui doivent être supprimés pour l'élément.
- java.lang.annotation.Documented – spécifie que les annotations avec une déclaration de type doivent être documentées par javadoc et des outils similaires par défaut pour faire partie de l'API publique des éléments annotés.
- java.lang.annotation.Inherited – spécifie qu'un type d'annotation est automatiquement hérité de la classe parent
- java.lang.annotation.Retention – spécifie jusqu'à quel point les annotations sont conservées lors du traitement. Les valeurs possibles sont les suivantes :
  - SOURCE – les annotations sont supprimées à la compilation
  - CLASS – [défaut] les annotations sont retenues par le compilateur, mais supprimées à l'exécution
  - RUNTIME – les annotations sont retenues par la machine virtuelle lors de l'exécution
- java.lang.annotation.Target – restreint le type d'élément de programme auquel une annotation peut être appliquée et génère des erreurs au moment de la compilation. Peut avoir l'une des valeurs suivantes :
  - TYPE – classe, interface, ou déclaration d'enum
  - FIELD – y compris les constantes d'enum
  - METHOD
  - PARAMETER
  - CONSTRUCTOR
  - LOCAL_VARIABLE
  - PACKAGE

Pour obtenir des informations générales sur la modélisation de cette forme de métadonnée dans PowerAMC, voir *Attributs (MOO)* à la page 69.

Mot clé strictfp Java

Un processeur de virgule flottante peut effectuer des calculs avec plus de précision et une plus grande plage de valeurs que ce que requiert la spécification Java. Le mot clé strictfp peut être utilisé avec des classes ou des opérations afin de spécifier la conformité avec la spécification Java (IEEE-754). Si ce mot clé n'est pas défini, les calculs à virgule flottante peuvent varier d'un environnement à l'autre.

Pour activer le mot clé strictfp :

1. Double-cliquez sur une classe pour afficher sa feuille de propriétés.
2. Cliquez sur l'onglet Attributs étendus, puis sur le sous-onglet Java.
3. Localisez l'entrée strictfp dans la colonne Nom, puis définissez la valeur à "true".
4. Cliquez sur OK. La classe sera générée avec le mot clé strictfp, et effectuera des opérations conformément aux spécifications Java concernant la virgule flottante

Enterprise Java Beans (EJB) v2

La plate-forme Java TM 2, Enterprise Edition (*J2EE* TM) est une plate-forme Java qui définit le standard pour le développement d'applications professionnelles multi-tiers. J2EE simplifie le développement des applications en les basant sur des composants standardisés, réutilisables et modulaires, fournit un jeu complet de services pour ces composants, et gère automatiquement sans recours à une programmation complexe de nombreux détails du comportement de l'application.

PowerAMC prend en charge la spécification EJB 2.0 avec un accent particulier mis sur les beans d'entité (CMP et BMP), et permet de profiter pleinement de l'intégration entre le MPD et le MOO.

Pour travailler avec EJB 2.0, vous devez disposer de Java 2 SDK Standard Edition (J2SE TM) 1.3 (édition finale), Java 2 SDK Enterprise Edition (J2EE TM) 1.3 (édition finale), d'un Java IDE ou d'un éditeur de texte, ainsi qu'un serveur d'application J2EE prenant en charge EJB 2.0.

Nous vous recommandons d'ajouter les variables système `JAVA_HOME` et `J2EE_HOME` dans votre environnement de la façon suivante :

Dans `CLASSPATH` :

```
%JAVA_HOME%\lib;%J2EE_HOME%\lib\j2ee.jar;%J2EE_HOME%\lib
```

Dans `Path` :

```
%JAVA_HOME%\bin;%J2EE_HOME%\bin
```

Pour plus d'informations sur les EJB, reportez-vous au site Web Java d'Oracle, à l'adresse suivante : *http://www.oracle.com/technetwork/java/index.html*.

Utilisation des types d'EJB

Vous pouvez définir les différents types de composants EJB suivants :

Type	Définition
Beans d'entité	Conçus pour représenter des données dans la base de données ; ils donnent aux données une sémantique objet de métier et lisent et mettent à jour les données automatiquement. Ils incluent : *CMP (Container Managed Persistence, persistance gérée par conteneur)* Dans le cas des beans d'entité CMP, la persistance est gérée par le serveur de composants (également appelé conteneur) *BMP (Bean Managed Persistence, persistance gérée par bean)* Dans le cas des beans d'entité BMP, la gestion de la persistance est laissée au développeur du bean
Beans de session (avec et sans état)	Encapsule la logique métier et fournit un point d'entrée unique pour les utilisateurs du client. En règle générale, ces beans gèrent et fournissent un accès indirect à plusieurs beans d'entité, ce qui permet de réduire le trafic réseau de façon significative. Il peut s'agir de beans avec ou sans état (voir ci-dessous)
Beans commandés par message	Beans anonymes qui ne peuvent pas être référencés par un client particulier, mais répondent à des messages asynchrones JMS. Comme les beans de session, ils permettent d'encapsuler la logique métier du côté serveur

Beans d'entité

Les beans d'entité sont utilisés pour représenter des objets sous-jacents. L'application la plus courante pour les beans d'entité est leur représentation des données dans les bases de données relationnelles. Un simple bean d'entité peut être défini pour représenter une table de base de données où chaque instance du bean représente une ligne spécifique. Des beans d'entité plus complexes peuvent représenter des vues de tables jointes dans une base de données. Une instance peut alors représenter un client spécifique et toutes les commandes et les articles commandés par ce client.

Le code généré est différent selon le type de bean d'entité : persistance gérée par conteneur (CMP, Container Managed Persistence) ou persistance gérée par bean (BMP, Bean Managed Persistence).

Beans de session avec ou sans état

Un bean de session est un EJB dans lequel chaque instance d'un bean de session est créée via son interface Home et est privée du point de vue de la connexion client. L'instance de bean de session ne peut pas être facilement partagée par d'autres clients, ce qui permet au bean de session de maintenir l'état du client. La relation entre le client et l'instance de bean de session est de type un-un.

Les beans de session *avec état (stateful)* gèrent un état conversationnel lorsqu'ils sont utilisés par un client. Un état conversationnel n'est pas écrit dans une base de données, il s'agit d'un état conservé en mémoire pendant qu'un client utilise une session.

Les beans de session *sans état (stateless)* ne gèrent pas d'état conversationnel. Chaque méthode est indépendante et n'utilise que les données transmises dans ses paramètres.

Beans commandés par message (Message Driven Beans)

Il s'agit de composants côté serveur, dépourvus d'état et ayant un comportement transactionnel qui traite les messages asynchrones délivrés par JMS (Java Message Service, service de messagerie Java). Les applications utilisent une messagerie asynchrone pour communiquer en échangeant des messages qui permettent aux émetteurs de rester indépendants des récepteurs.

Propriétés d'un EJB

L'onglet **EJB** de la feuille de propriétés d'un composant fournit des propriétés supplémentaires.

Propriété	Description
Interface Remote Home	Définit les méthodes et opérations utilisées dans une vue client distante. Etend l'interface javax.ejb.EJBHome
Interface Remote	Fournit la vue client distante. Etend l'interface javax.ejb.EJBObject
Interface Local Home	Définit les méthodes et opérations utilisées localement dans une vue client locale. Etend l'interface javax.ejb.EJBLocal-Home
Interface Local	Permet au bean d'être étroitement lié à son client et d'être accessible directement. Etend l'interface javax.ejb.EJBLocalObject
Classe Bean	Classe qui met en oeuvre les méthodes de gestion du bean
Classe de clé primaire	Classe fournissant un pointeur dans la base de données. Cette classe est liée à la classe Bean. Applicable uniquement aux beans d'entité

Remarque : Vous pouvez afficher la page EJB en pointant sur un symbole de composant EJB, en cliquant le bouton droit de la souris, puis un sélectionnant **EJB**.

Pour plus d'informations sur les méthodes d'interface et les méthodes de mise en oeuvre, voir *Notions de base relatives à la synchronisation des opérations* à la page 379.

Affichage de l'aperçu du code d'un composant

Vous pouvez voir la relation entre un EJB et ses classes et interfaces à partir de l'onglet **Aperçu** sans avoir à générer de fichier. Pour afficher un aperçu du code d'un EJB, cliquez sur l'onglet Aperçu dans la feuille de propriétés d'un composant (voir *Aperçu du code d'un objet* à la page 8). Les différents sous-onglets montrent le code pour chaque interface et classe de l'EJB. Dans la feuille de propriétés de modèle ou de package, la page Aperçu décrit le fichier descripteur de déploiement EJB avec le nom de l'EJB généré et ses méthodes.

Création d'un EJB à l'aide de l'Assistant

Vous pouvez utiliser l'*Assistant* pour créer un composant EJB. Cet Assistant vous guide tout au long des différentes étapes de création du composant et est disponible uniquement si le langage du modèle est Java.

L'Assistant peut être lancé à partir d'un *diagramme de classes*. Vous pouvez soit créer un EJB sans sélectionner de classe, soit commencer par sélectionner une classe, puis lancer l'Assistant à partir du menu contextuel du symbole de classe.

Vous avez également la possibilité de créer plusieurs EJB du même type en sélectionnant plusieurs classe simultanément. L'Assistant crée alors automatiquement un EJB par classe. Les classes que vous avez sélectionnées dans le diagramme de classes deviennent des classes Bean. Elle sont renommées pour être en conformité avec les conventions de dénomination standard et liées à leur composant.

Si vous avez sélectionné des classes ou des interfaces avant de démarrer l'Assistant, ces classes sont automatiquement liées au nouveau composant EJB.

Lorsqu'une interface ou une classe est déjà dotée d'un stéréotype, par exemple <<EJBEntity>>, elle est principalement utilisée pour être l'interface ou la classe de l'EJB.

Pour plus d'informations sur la classe ou l'interface d'EJB stéréotypée, voir la section *Définition d'interfaces et de classes pour les EJB* à la page 375.

L'Assistant de création d'un EJB permet de définir les paramètres suivants :

Zone de l'Assistant	Description
Nom	Nom du composant EJB
Code	Code du composant EJB
Type de composant	Entity Bean CMP, Entity Bean BMP, Message Driven Bean, Session Bean Stateful ou Session Bean Stateless. Pour plus d'informations sur les différents types d'EJB, reportez-vous à la section *Utilisation des types d'EJB* à la page 371
Classe Bean	Nom de la classe qui met en oeuvre les méthodes de gestion du bean
Interface Remote	Etend l'interface javax.ejb.EJBObject et fournit une vue client distante
Interface Remote Home	Définit les méthodes et opérations utilisées dans une vue client distante. Etend l'interface javax.ejb.EJBHome
Interface Local	Etend l'interface javax.ejb.EJBLocalObject. Permet aux beans d'être étroitement liés à leurs clients et d'être directement accessibles
Interface Local Home	Définit les méthodes et opérations utilisées localement dans une vue client locale. Etend l'interface javax.ejb.EJBLocal-Home

Zone de l'Assistant	Description
Classe de clé primaire	Nom de la classe qui fournit un pointeur vers la base de données. Applicable uniquement aux beans d'entité
Transaction	Définit le support de transaction utilisé pour le composant. La transaction est importante pour la distribution via un réseau depuis un serveur. La valeur de support de transaction s'affiche dans le descripteur de déploiement. Cette information est fournie par le descripteur de déploiement au serveur lorsque vous générez le composant
Créer un symbole dans	Crée un symbole de composant dans le diagramme de composants spécifié dans la liste. S'il existe déjà un diagramme de composants, vous pouvez le sélectionner dans la liste. Vous pouvez également afficher la feuille de propriétés du diagramme sélectionné en cliquant sur l'outil Propriétés
Créer un diagramme de classes pour les classificateurs du composant	Crée un diagramme de classes avec un symbole pour chaque classe et interface associée au composant. Si vous avez sélectionné des classes et interfaces avant de démarrer l'Assistant, elles sont utilisées pour créer le composant. Cette option permet d'afficher ces classes et interfaces dans un diagramme

La zone de groupe Transaction contient les valeurs suivantes, conformes à la spécification Enterprise JavaBeans 2.0 :

Valeur	Description
Non supportée	Le composant ne prend pas en charge les transactions, elles ne lui sont pas nécessaires. S'il en existe une, il l'ignore
Supportée	Le composant attend une transaction, il l'utilise
Requise	S'il n'existe aucune transaction, elle est créée
Nouvelle requise	Le composant doit disposer d'une nouvelle transaction à sa création, le serveur doit lui fournir cette nouvelle transaction
Obligatoire	En l'absence de transaction, une exception est renvoyée
Jamais	Une transaction n'est pas nécessaire

Le descripteur de déploiement d'EJB prend en charge le type de transaction pour chaque méthode : vous pouvez donc spécifier un type de transaction pour chaque méthode d'interface Remote ou Local d'EJB.

Vous pouvez définir le type de transaction pour chaque méthode à l'aide d'un attribut étendu dans le dossier Profile/Operation/Extended Attributes/EJB du langage objet Java. Si le type de

transaction de l'opération n'est pas spécifié (vide), le type de transaction par défaut défini dans le composant est utilisé.

1. Sélectionnez **Outils > Créer un Enterprise JavaBean** à partir d'un diagramme de classes pour afficher la boîte de dialogue Assistant EJB (Enterprise JavaBean).

   Remarque : Si vous avez sélectionné des classes avant de lancer l'Assistant, certaines des étapes suivantes peuvent être omises car les différents noms sont créés par défaut en fonction des noms des classes sélectionnées.

2. Saisissez un nom et un code pour le composant, puis cliquez sur **Suivant**.
3. Sélectionnez le type de composant, puis cliquez sur **Suivant**.
4. Sélectionnez le nom de classe Bean, puis cliquez sur **Suivant**.
5. Sélectionnez les noms d'interface Remote et Remote Home, puis cliquez sur **Suivant**.
6. Sélectionnez les noms d'interface Local et Local Home, puis cliquez sur **Suivant**.
7. Sélectionnez le nom de classe de clé primaire, puis cliquez sur **Suivant**.
8. Sélectionnez le type de support de transaction, puis cliquez sur **Suivant**.
9. A la fin de l'Assistant, vous devez définir la création des symboles et diagrammes.

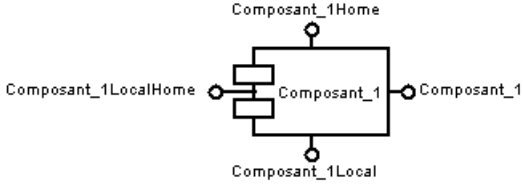
Une fois que vous avez fini d'utiliser l'Assistant, les actions suivantes sont exécutées :

- Un composant EJB est créé
- Les classes et interfaces sont associées au composant, et les éventuelles classes ou interfaces manquantes associées au composant sont créées
- Les diagrammes associés au composant sont mis à jour ou créés
- En fonction du type d'EJB choisi, la classe de clé primaire d'EJB, ses interfaces et dépendances sont automatiquement créées et visibles dans l'Explorateur d'objets. En outre, toutes les dépendances entre les interfaces Remote, les interfaces Local et la classe Bean du composant sont créées
- L'EJB créé est nommé d'après la classe d'origine si vous avez sélectionné une classe avant de démarrer l'Assistant. Les classes et interfaces ont également un préfixe faisant référence au nom de la classe d'origine afin de préserver la cohérence

Définition d'interfaces et de classes pour les EJB

Un EJB comprend un certain nombre d'interfaces et de classes de mise en oeuvre spécifiques. Les interfaces d'un EJB sont toujours exposées, vous définissez une interface publique et l'exposez. Vous ne pouvez associer une interface ou une classe qu'à un seul EJB à la fois

Les interfaces de composant d'EJB sont illustrées sous forme de cercles reliés au côté du composant EJB par une ligne horizontale ou verticale :



Les interfaces fournissent une vue distante (Interface Remote Home/Interface Remote), ou une vue locale (Interface Local Home/Interface Local).

Les classes n'ont pas de symbole dans le diagramme de composants, mais la relation entre la classe et le composant EJB est visible dans l'onglet Classes de la feuille de propriétés du composant EJB, ainsi que l'onglet Composants dans l'onglet Dépendances de la feuille de propriétés de classe.

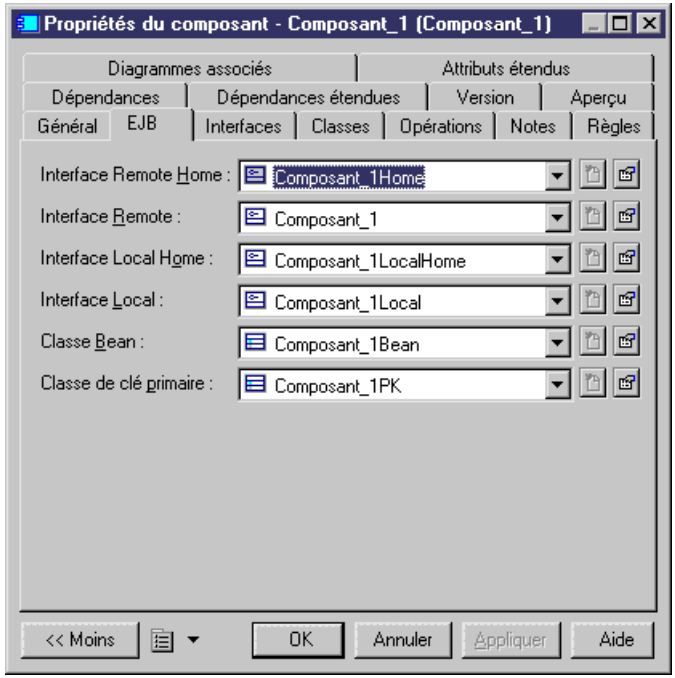
Le tableau suivant affiche les stéréotypes utilisés pour identifier automatiquement les interfaces et classes d'EJB :

Stéréotype	Décrit
<<EJBRemoteHome>>	L'interface Remote Home
<<EJBRemote>>	L'interface Remote
<<EJBLocalHome>>	L'interface Local Home
<<EJBLocal>>	L'interface Local
<<EJBEntity>>	La classe Bean du bean d'entité
<<EJBSession>>	La classe Bean du bean de session
<<EJBMessageDriven>>	La classe Bean du bean commandé par message
<<EJBPrimaryKey>>	La classe de clé primaire d'un bean d'entité

Les noms de template sont instanciés conformément au composant correspondant et affectés aux objets créés. Si une interface ou classe non associée correspondant à un nom ou à un type de classificateur donné existe dans le modèle, elle est automatiquement associée à l'EJB.

1. Pointez sur le composant EJB dans le diagramme, cliquez le bouton droit de la souris, puis sélectionnez EJB dans le menu contextuel.

   La feuille de propriétés du composant s'affiche à l'onglet EJB. Les interfaces et classes sont créées et associées à l'EJB.



Vous pouvez utiliser le bouton Créer en regard du nom de la classe ou de l'interface pour recréer une interface ou une classe si elle est définie à la valeur <Aucun>.

2. Cliquez sur le bouton Propriétés en regard du nom de l'interface ou de la classe que vous souhaitez définir.

   La feuille de propriétés de l'interface ou de la classe s'affiche.

3. Sélectionnez les propriétés appropriées.

   Les définitions d'interfaces et de classes sont ajoutées dans la définition du composant EJB.

Définition d'opérations pour un EJB

Vous pouvez créer les types d'opération suivants pour un EJB à partir de la feuille de propriétés de la classe Bean ou des interfaces d'EJB :

- EJB Business Method (local)
- EJB Business Method (remote)
- EJB Create Method (local)
- EJB Create Method (remote)
- EJB Finder Method (local)
- EJB Finder Method (remote)
- EJB Select Method

Remarque : Vous ne pouvez pas créer une opération à partir de l'onglet Opérations de la feuille de propriétés du composant car cette dernière permet uniquement de visualiser les opérations du composant EJB. Pour afficher les opérations d'un EJB, vous devez ouvrir l'onglet Opérations de la feuille de propriétés du composant

Stéréotypes

Les stéréotypes standard suivants, définis pour les EJB, sont affectés à ces opérations :

- <<EJBCreateMethod>>
- <<EJBFinderMethod>>
- <<EJBSelectMethod>>

Beans d'entité CMP

Vous pouvez créer les opérations suivantes pour les beans d'entité CMP uniquement :

- EJB Create(...) Method (local)
- EJB Create(...) Method (remote)

Lorsque vous créez une opération EJB Create(...) Method (local), la méthode est créée dans l'interface local home avec tous les attributs persistants comme paramètres. Lorsque vous créez une opération EJB Create(...) Method (remote), la méthode est créée dans l'interface remote home avec tous les attributs persistants comme paramètres.

En outre, toutes les méthodes liées ejbCreate(...) et ejbPostCreate(...) sont créées automatiquement dans la classe Bean avec tous les attributs persistants comme paramètres. Les paramètres sont synchronisés chaque fois qu'une modification est appliquée.

Remarque : Si vous devez modifier les méthodes d'une interface, vous pouvez le faire à partir de l'onglet Opérations de la feuille de propriétés d'interface.

Ajout d'une opération dans une classe Bean

A l'issue de la création de la classe Bean, vous pouvez être amené à créer une méthode qui fait défaut à ce stade du processus, par exemple une méthode interne.

1. Double-cliquez sur la classe Bean pour afficher sa feuille de propriétés.
2. Cliquez sur l'onglet Opérations, puis cliquez sur une ligne vide dans la liste.

   Une flèche s'affiche au début de la ligne.
3. Double-cliquez sur la flèche au début de la ligne.

   Une boîte de confirmation vous demande de confirmer la création de l'objet.
4. Cliquez sur Oui.

   La feuille de propriétés de l'opération s'affiche.
5. Saisissez un nom et un code pour votre opération.
6. Cliquez sur l'onglet Mise en oeuvre.

L'onglet Mise en oeuvre s'affiche sur le sous-onglet Corps.

7. Ajoutez le code de la méthode sur ce sous-onglet.
8. Cliquez sur OK.

   Vous revenez à la feuille de propriétés de classe.
9. Cliquez sur l'onglet Aperçu pour afficher l'onglet correspondant.

   Vous pouvez maintenant valider le code Java à générer pour la classe Bean.
10. Cliquez sur OK.

Création d'une opération dans une interface d'EJB

Vous pouvez ajouter une opération dans une interface d'EJB à partir de la feuille de propriétés de l'interface ou de la feuille de propriétés de la classe Bean, en utilisant le bouton Ajouter de l'onglet Opérations.

Lorsque vous ajoutez une opération à une interface, une opération de réalisation est automatiquement créée dans la classe Bean car l'opération de l'interface a une méthode liée. Ceci assure la bonne synchronisation des opérations (voir *Notions de base relatives à la synchronisation des opérations* à la page 379).

1. Affichez la feuille de propriétés de la classe Bean, puis cliquez sur l'onglet **Opérations**.
2. Cliquez sur l'outil **Ajouter...** puis sélectionnez l'opération EJB requise dans la liste.

   L'opération demandée est créée à la fin de la liste des opérations dans la feuille de propriétés de la classe Bean, et vous pouvez également vérifier que la nouvelle opération s'affiche dans la liste des opérations d'interface.

Notions de base relatives à la synchronisation des opérations

La synchronisation permet de préserver la cohérence générale du modèle chaque fois qu'une modification est appliquée aux opérations, aux attributs et aux exceptions. Elle se produit au fur et à mesure que vous modifiez le modèle.

Synchronisation d'opération

La synchronisation se produit de l'interface vers la classe Bean. Les opérations d'interface sont dotées de *méthodes liées* dans la classe Bean, avec le nom/code, le type de résultat et les paramètres synchronisés avec l'opération d'interface. Lorsque vous ajoutez une opération à l'interface, vous avez la possibilité de vérifier que la méthode liée correspondante est créée dans la classe Bean (grisée dans la liste). En revanche aucune opération n'est créée dans une interface si vous ajoutez une opération à une classe Bean.

Par exemple, double-cliquez sur la classe Bean d'un composant, cliquez sur l'onglet Opérations, cliquez sur le bouton Ajouter en bas de l'onglet Opérations, puis sélectionnez EJB Create method (local) : PowerAMC ajoute cette opération à l'interface et crée automatiquement les opérations ejbCreate et ejbPostCreate dans la classe Bean.

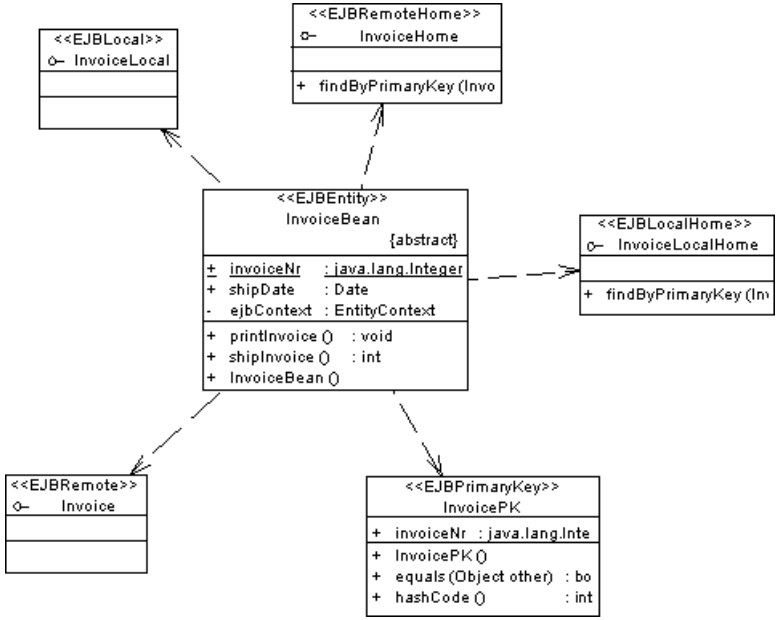
Synchronisation d'exception
Les exceptions sont synchronisées de la classe Bean vers les interfaces. La liste des exceptions de l'interface de la méthode Create de l'interface home est un sur-ensemble de l'union des listes d'exceptions des opérations à réaliser ejbCreate et ejbPostCreate correspondantes dans la classe Bean.

Les attributs d'exception d'interface sont par conséquent mis à jour chaque fois que la liste des exceptions de la méthode à réaliser de la classe Bean est modifiée.

Notions de base relatives à la prise en charge des EJB dans un MOO

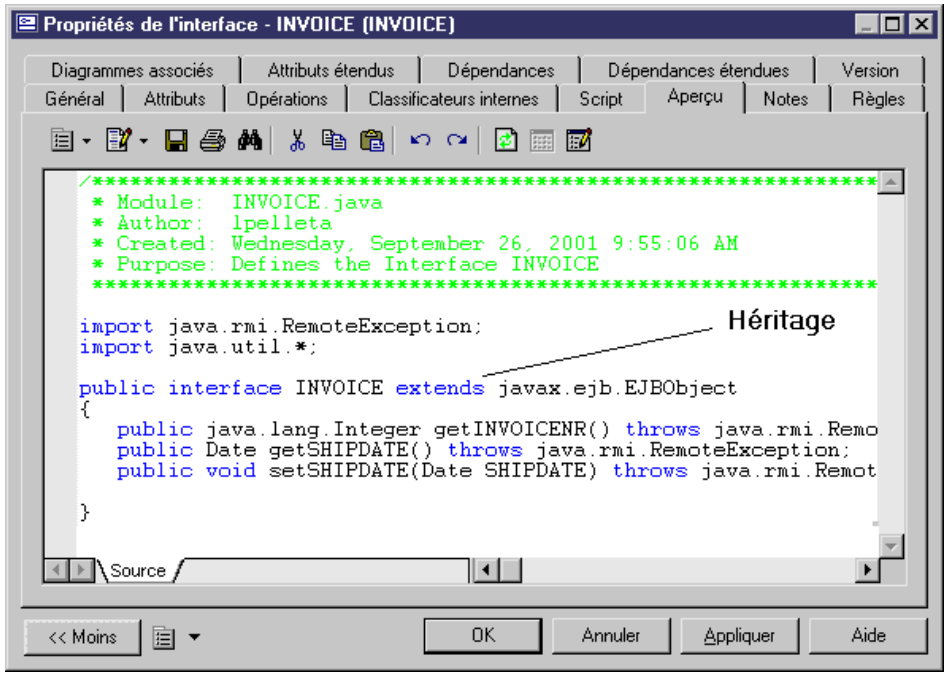
PowerAMC simplifie le processus de développement en gérant de façon transparente les concepts EJB et en assurant le respect du contrat de programmation d'EJB.

- Initialisation automatique : - Lorsque vous créez un EJB, le rôle du processus d'initialisation est d'initialiser les classes et les interfaces et leurs méthodes conformément au contrat de programmation. PowerAMC accomplit automatiquement cette tâche chaque fois qu'une classe ou interface est attachée à un EJB



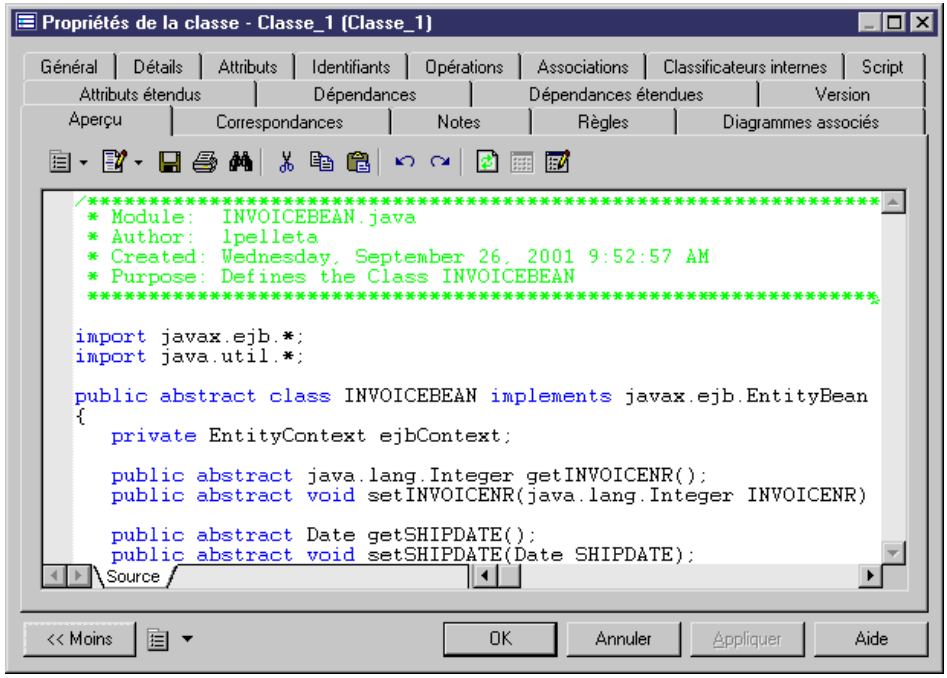
- Synchronisation - assure la cohérence du modèle tout entier dès qu'une modification intervient :
  - Opérations - La synchronisation s'effectue de l'interface vers la classe Bean avec des méthodes liées (*Notions de base relatives à la synchronisation des opérations* à la page 379).
  - Exceptions - La synchronisation s'effectue de la classe Bean vers l'interface (*Notions de base relatives à la synchronisation des opérations* à la page 379).

    - Attribut d'identifiant primaire - La synchronisation s'effectue de la classe Bean vers la classe de clé primaire. Lorsque l'attribut est un identifiant primaire dans la classe Bean, il est automatiquement migré vers la classe de clé primaire.
- Vérifications de modèle : la fonctionnalité de vérification de modèle permet de valider un modèle et de compléter la synchronisation en proposant des corrections automatiques. Vous pouvez vérifier votre modèle à tout moment en utilisant la commande Vérifier le modèle (menu Outils) (voir *Chapitre 9, Vérification d' un MOO* à la page 303).
- Génération de code basé sur des templates - des templates spécifiques pour EJB sont disponibles dans la catégorie Profile/Component/Templates du langage objet Java. La génération de classes et d'interfaces EJB crée les liens d'héritage suivants (pour les beans d'entité CMP) :
    - L'interface Local Home hérite dejavax.ejb.EJBLocalHome
    - L'interface Local hérite de javax.ejb.EJBLocalObject
    - L'interface Remote Home hérite de javax.ejb.EJBHome
    - L'interface Remote hérite de javax.ejb.EJBObject

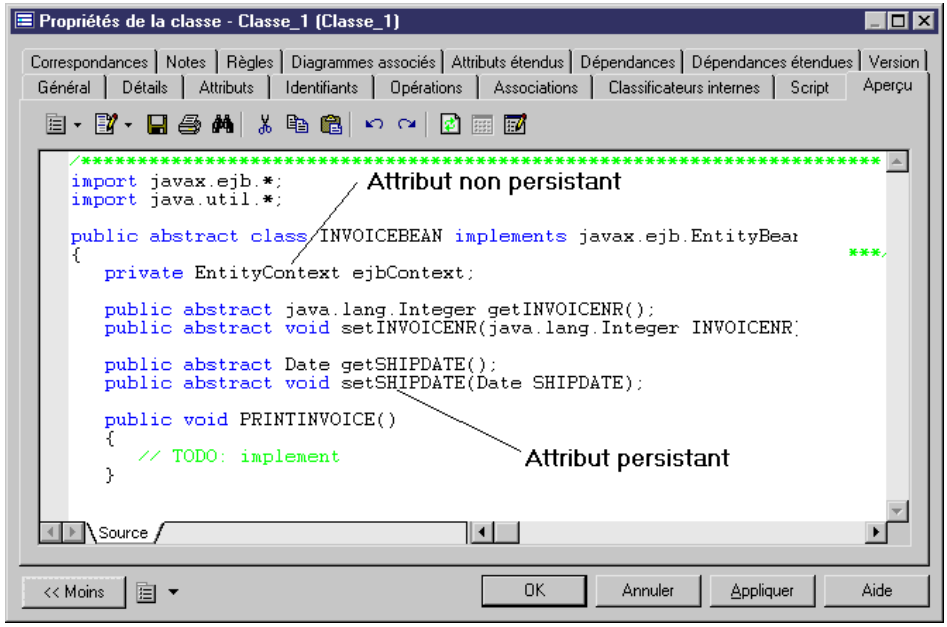


Il crée les liens de réalisation suivants :

- La classe de clé primaire met en oeuvrejava.io.Serializable
- La classe Bean met en oeuvre javax.ejb.EntityBean



Il transforme les champs CMP (attributs marqués comme persistants) et les champs CMR (attributs migrés depuis de associations) dans des méthodes getter et setter :



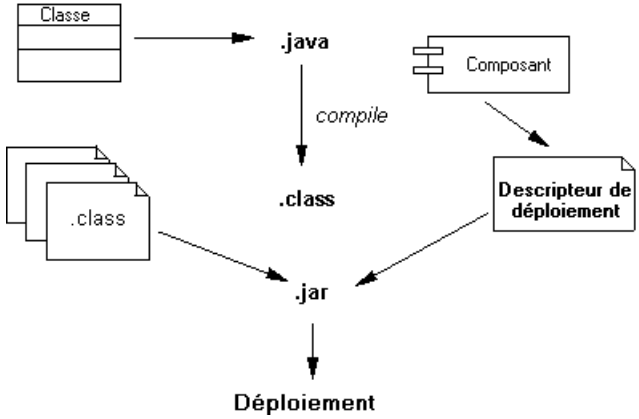
A un haut niveau, PowerAMC prend en charge différentes approches pour vous aider dans le processus de développement de composant, notamment :

- Ingénierie standard : depuis un MOO vers un MPD. Permet de créer et de récupérer par reverse engineering des EJBs dans un MOO, de générer le MPD correspondant, d'établir une correspondance O/R et de générer du code
- Reverse engineering : depuis un MPD (base de données) vers un MOO. Permet de créer et de récupérer par reverse engineeering des tables dans un MPD, de générer les classes correspondantes, de créer un EJB à partir de classes données et de générer du code

Affichage de l'aperçu d'un descripteur de déploiement d'EJB

Un descripteur de déploiement d'EJB décrit la structure et les propriétés d'un ou de plusieurs EJB dans un fichier au format XML. Le descripteur de déploiement est utilisé pour déployer l'EJB dans le serveur d'application. Il déclare les propriétés des EJB, les relations et les dépendances entre les EJB. Un descripteur de déploiement est automatiquement généré pour chaque package ou modèle, il décrit les EJB définis dans ce package ou modèle.

Le rôle du descripteur de déploiement au sein du processus global est illustré ci-dessous :



Le descripteur de déploiement d'EJB et les classes Java compilées des EJB doivent être inclus dans un fichier JAR.

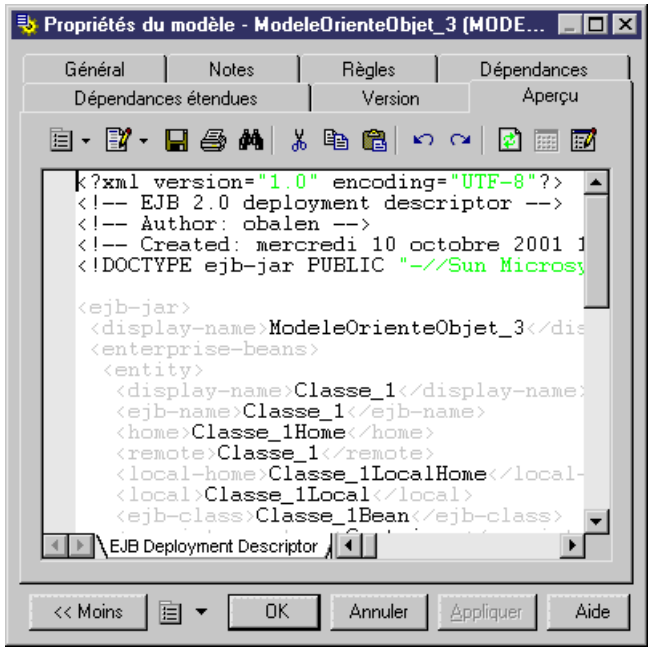
L'outil de déploiement d'EJB du serveur d'applications importe les classes Java à partir du fichier JAR et configure les EJB dans le serveur d'applications en fonction de la description des EJB contenue dans le descripteur de déploiement d'EJB.

Vous pouvez afficher le descripteur de déploiement sur l'onglet Aperçu de la feuille de propriétés du package ou du modèle.

Vous pouvez personnaliser le descripteur de déploiement d'EJB en modifiant les templates dans le fichier du langage objet Java.

Pour plus d'informations sur la personnalisation du langage objet, voir *Personnalisation et extension de PowerAMC > Fichiers de définition pour les langage objet, de processus et XML*.

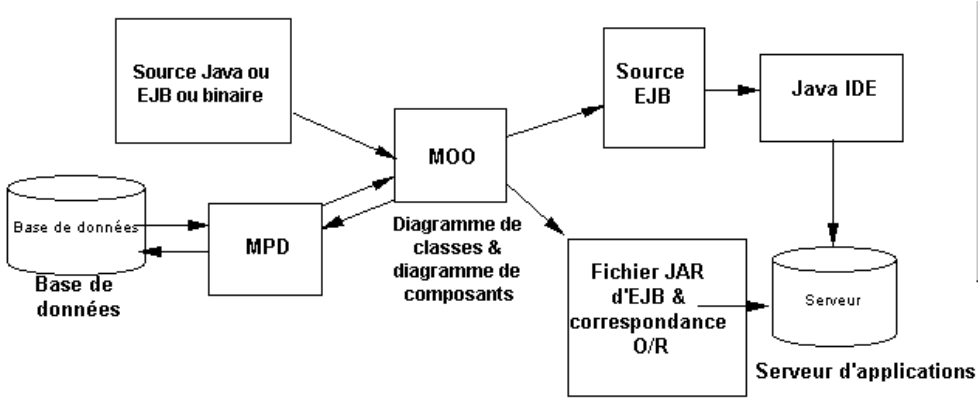
1. Pointez sur le modèle dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Propriétés** pour afficher la feuille de propriétés du modèle.
2. Cliquez sur l'onglet **Aperçu**.



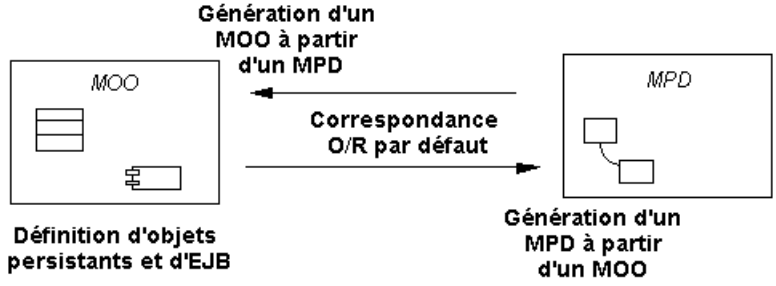
Génération des EJB

Le processus de génération des EJB permet de générer du code source EJB compatible J2EE 1.3 (EJB 2.0). Le générateur de code (Java, EJB, XML, etc...) est basé sur des templates et des macros. Vous pouvez personnaliser le code généré en éditant le langage objet Java (sélectionnez **Outils > Ressources > Langages objet**.

L'illustration suivante montre le processus de développement des EJB.



L'illustration suivante se concentre sur la partie concernant PowerAMC, et met en exergue le rôle de la génération de correspondances O/R. La correspondance O/R peut être créée lorsque vous générez un MPD à partir d'un MOO ou que vous générez un MOO à partir d'un MPD.



Pour plus d'informations sur la mise en correspondance d'objets, voir *Chapitre 19, Modélisation des correspondances objet/relationnel (O/R)* à la page 507.

Quel type de génération utiliser ?

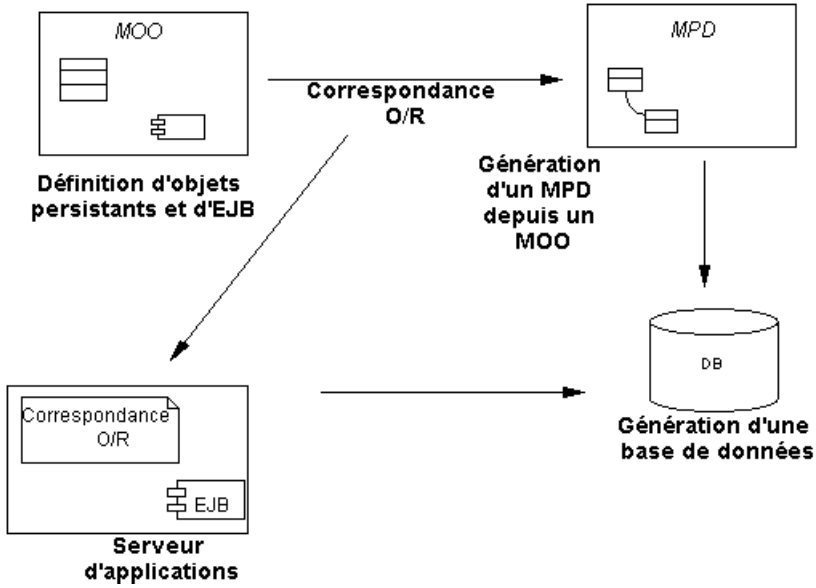
Vous pouvez être confronté à deux types de situation lorsque vous générez du code source EJB :

- Vous ne disposez pas d'une base de données
- Vous disposez déjà d'une base de données

Vous pouvez souhaiter générer du code source EJB lorsque vous créez une base de données. Si vous ne disposez pas encore d'une base de données, le processus de développement se déroule comme suit :

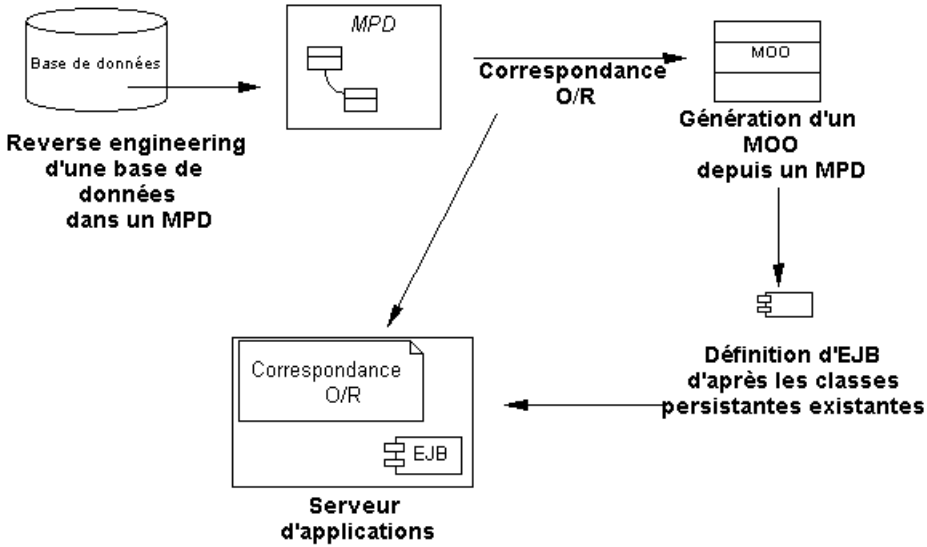
- Vous définissez des objets persistants dans le diagramme de classes, et définissez des EJB dans le diagramme de composants
- Vous générez le MPD, en créant la correspondance O/R lors de la génération
- Vous créez la base de données

- Vous générez le code source d'EJB et le descripteur de déploiement pour le serveur d'applications



Vous pouvez être amené à générer du code source EJB pour une base de données existante. Si la base de données existe déjà, vous pouvez être amené à englober la base de données existante dans des EJB et à utiliser ces EJB pour accéder à cette dernière. Le processus de développement se déroule alors comme suit :

- Vous procédez au reverse engineering du MPD dans un MOO, et vous identifiez les correspondances O/R lors de cette opération
- Vous définissez des EJB dans le MOO en fonction des classes persistantes
- Vous générez le code source d'EJB et le descripteur de déploiement pour le déploiement du serveur d'applications



Vous pouvez également générer le code source d'EJB lorsque vous gérez la persistance des EJB avec une base de données existante. Par exemple, il peut être nécessaire de gérer la persistance d'un EJB déjà défini avec une base de données existante. Etant donné que vous ne pouvez pas modifier la définition de l'EJB ou la structure de la base de données, vous devez utiliser une mise en correspondance O/R manuelle.

Pour plus d'informations sur la mise en correspondance d'objets, voir *Chapitre 19, Modélisation des correspondances objet/relationnel (O/R)* à la page 507.

Notions de base relatives au source EJB et à la persistance

Vous générez les méthodes de gestion de la persistance en fonction du langage objet.

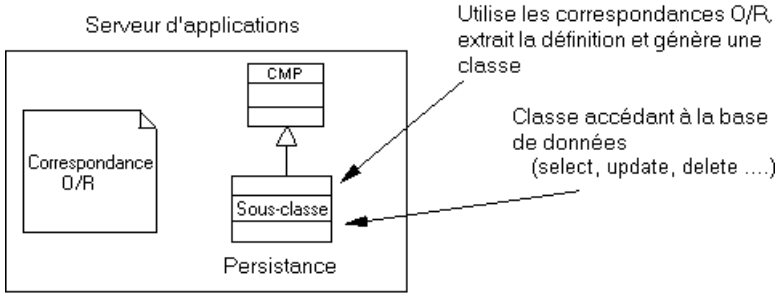
Selon que l'EJB est de type CMP ou BMP, le fichier de descripteur de déploiement a un contenu différent :

- Un CMP met en jeu le serveur d'applications. Il inclut l'EJB et le descripteur de correspondance O/R (.XML). Le serveur extrait à la fois l'EJB et le descripteur de correspondance O/R pour générer le code

  Si le serveur d'applications ne prend pas en charge le descripteur de correspondance O/R, la mise en correspondance doit être effectuée manuellement. Si le descripteur de correspondance O/R de votre serveur d'applications n'est pas encore pris en charge, vous pouvez créer votre propre descripteur en créant un nouveau fichier d'extension. Pour plus d'informations sur l'utilisation des fichiers d'extension, voir *Personnalisation et extension de PowerAMC > Fichiers d'extension*.
- Un BMP met en jeu un processus manuel. Il inclut la source de l'EJB, sans descripteur de correspondance O/R (la correspondance O/R est facultative). Le développeur de BMP doit mettre en oeuvre lui-même la gestion de la persistance à l'aide des méthodes ejbStore() et

ejbLoad(), le serveur d'applications ne prend en charge que ses fonctions. Une classe de mise en oeuvre hérite de la classe Bean BMP, gère les données de persistance et communique avec la base de données.

- Vous pouvez également définir un EJB comme CMP, puis le générer comme BMP lorsque vous générez le code. Le générateur de code génère une classe de mise en oeuvre (sous-classe) pour la classe Bean qui contient ses méthodes, et utilise une correspondance O/R et un template persistant pour mettre en oeuvre la persistance.



Pour plus d'informations sur la mise en correspondance d'objets, voir *Chapitre 19, Modélisation des correspondances objet/relationnel (O/R)* à la page 507.

Vous pouvez utiliser différentes méthodes pour générer un EJB CMP dans un EJB BMP. Vous pouvez soit copier le langage objet Java fourni dans PowerAMC comme référence à un nouveau langage objet, puis décrire comment doivent être générées les classes de mise en oeuvre de l'EJB CMP dans votre propre langage objet, ou bien créer un fichier d'extension qui inclut ces classes de mise en oeuvre.

Vous pouvez également rédiger du code VB script pour convertir l'EJB CMP en EJB BMP. Pour ce faire, vous devez générer l'EJB comme CMP, puis lancer le script VB qui passera en revue tous les objets du modèle et générera une classe de mise en oeuvre pour chaque classe identifiée.

Génération du code source et du descripteur de déploiement d'un EJB

Lorsque vous générez un EJB, les classes et interfaces de l'EJB sont directement sélectionnées. La génération récupère les classes utilisées par le composant, ainsi que les interfaces associées à ces classes.

1. Sélectionnez **Langage > Générer du code Java** pour afficher la boîte de dialogue Génération.
2. Spécifiez le répertoire dans lequel les fichiers doivent être générés, et indiquez si vous souhaitez effectuer une vérification de modèle (voir *Chapitre 9, Vérification d'un MOO* à la page 303).
3. [facultatif] Cliquez sur l'onglet **Sélection** et sélectionnez les objets que vous souhaitez générer. Par défaut, tous les objets sont générés.

4. [facultatif] Cliquez sur l'onglet **Options** et sélectionnez les options appropriées (voir *Génération de fichiers Java* à la page 416).
5. [facultatif] Cliquez sur l'onglet **Tâches** et sélectionnez les tâches à effectuer lors de la génération (voir *Génération de fichiers Java* à la page 416).
6. Cliquez sur **OK** pour lancer la génération.

   Une boîte de progression s'affiche, suivie par une liste Résultats. Vous pouvez utiliser le bouton **Editer** dans la liste Résultats pour éditer chaque fichier généré individuellement.
7. Cliquez sur **Fermer**.

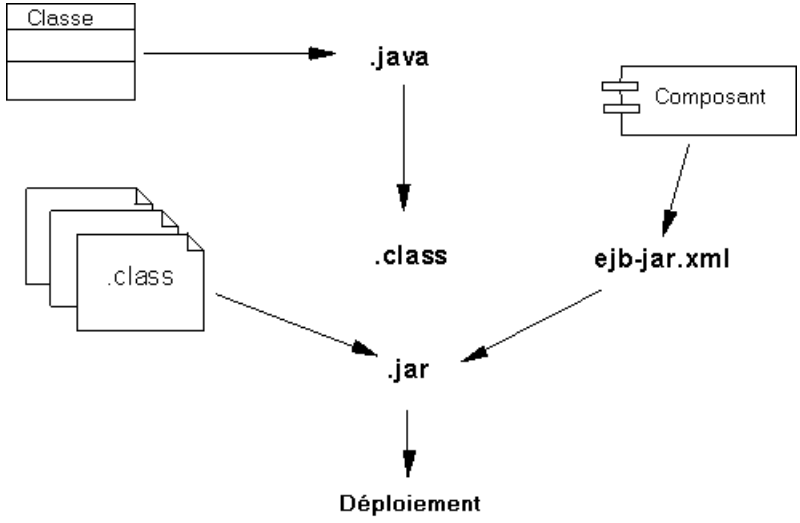
   Le descripteur de déploiement ejb-jar.xml est créé dans le répertoire META-INF et tous les fichiers sont générés dans le répertoire de génération.

Génération de fichiers .JAR

Pour conserver ensemble les différents éléments de l'EJB, les classes Bean, les interfaces et le descripteur de déploiement sont placés dans un fichier .JAR. Ce processus est commun à tous les types de composants EJB.

Vous pouvez générer des fichiers .JAR en utilisant l'onglet Tâches de la boîte de dialogue de génération (**Langage > Générer du code Java**).

Par exemple, une de ces tâches permet de compiler les fichiers .JAVA à l'aide d'un compilateur et de créer un fichier .JAR qui inclut les classes Java compilées, puis de le compléter en y ajoutant le descripteur de déploiement et les icônes.



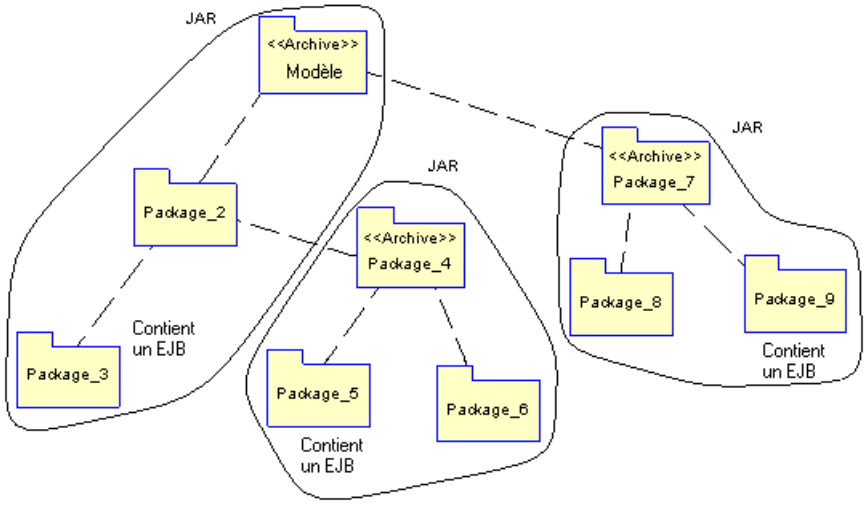
Avertissement ! Vous devez définir les valeurs des commandes utilisées lors de la génération afin de les activer. Pour ce faire, vous devez utiliser la section Variables de la boîte de dialogue Options générales. Par exemple, vous pouvez définir les exécutables javac.exe et jar.exe via cette boîte de dialogue.

Pour plus d'informations sur la définition de ces variables, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Personnalisation de votre environnement de modélisation > Options générales > Variables d'environnement*.

Il n'existe pas de contrainte portant sur la génération d'un fichier JAR par package. Seuls les packages ayant le stéréotype <<archive>> donnent lieu à la génération d'un fichier JAR s'ils contiennent (ou si l'un de leurs packages descendants n'ayant pas comme stéréotype <<archive>> contient) un EJB.

Le nouveau fichier archive créé contient le package et tous ses descendants non stéréotypés. Le package racine (qui est le modèle) est toujours considéré comme stéréotypé <<archive>>.

Par exemple, si un modèle contient plusieurs composants EJB situés dans différents sous-packages mais qu'aucun de ces packages n'est stéréotypé <<archive>>, un seul fichier JAR est créé et il contient tous les packages.



Reverse engineering de composants EJB

PowerAMC peut procéder au reverse engineering de composants EJB situés dans des fichiers .JAR.

1. Sélectionnez **Langage > Reverse engineering Java** pour afficher la boîte de dialogue de reverse engineering.
2. Sur l'onglet **Sélection**, sélectionnez `Archives` dans la liste **Reverse**.
3. Cliquez sur le bouton **Ajouter**, sélectionnez les objets sur lesquels vous souhaitez faire porter le reverse engineering, puis cliquez sur **Ouvrir** pour les ajouter dans la liste.
4. Cliquez sur l'onglet **Options**, puis cochez la case **Reverse engineering de descripteur de déploiement**.

5. Cliquez sur **OK**.

   Une boîte de progression s'affiche. Si le modèle dans lequel vous effectuez un reverse engineering contient déjà des données, une boîte de dialogue de fusion s'affiche.

   Pour plus d'informations sur la fusion de modèles, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Comparaison et fusion de modèles*.

6. Cliquez sur OK.

   L'onglet Reverse de la fenêtre Résultats affiche les changements effectués lors du reverse engineering et le modèle mis à jour est affiché dans la fenêtre de diagramme.

Enterprise Java Beans (EJB) v3

La spécification EJB 3.0 tente de simplifier l'architecture d'EJB 2.1 en limitant le nombre d'artefacts de programmation que les développeurs doivent fournir, réduisant le nombre de méthode callback requises et simplifiant le modèle de programmation de bean d'entité et le modèle de correspondance O/R.

Les deux principaux changements dans la proposition de spécification EJB 3.0 sont les suivants :

- *Un modèle de programmation d'EJB basé sur les annotations* - tous les types d'enterprise beans sont des plain old Java objects (POJOs) ayant les annotations appropriées. Une approche de configuration par exception utilise les valeurs par défaut intuitives pour présumer les principaux paramètres. Les annotations sont utilisées pour définir l'interface métier du bean, les correspondances O/R, les références de ressources ainsi que d'autres informations précédemment définies via les descripteurs de déploiement ou interfaces. Les descripteurs de déploiement ne sont pas requis ; l'interface Home n'est plus utilisée et il n'est plus nécessaire de mettre en oeuvre une interface métier (le conteneur peut la générer pour vous).

  Par exemple, vous déclarez un bean de session sans état en utilisant l'annotation @Stateless. Dans le cas de bean avec état, l'annotation @Remove est marquée sur une méthode particulière afin d'indiquer que l'instance de bean doit être supprimée après traitement de l'appel de la méthode marquée.
- *Le nouveau modèle de persistance pour les beans d'entité* - Les nouveaux beans d'entité sont également des POJOs avec quelques annotations et ne sont pas par essence des entités persistantes. Une instance d'entité devient persistante une fois qu'elle est associée à un EntityManager et devient une partie d'un contexte de persistance, qui est plus ou moins synonyme d'un contexte de transaction et coexiste de façon implicite avec la portée d'une transaction.

  Les relations de l'entité et les correspondances O/R sont définies via les annotations, en utilisant l'environnement open source Hibernate (voir *Chapitre 20, Génération d'objet persistants pour Java et de pages JSF* à la page 537).

Il y a également plusieurs effets de bord liés à ces propositions, telles que le nouveau modèle de programmation client, l'utilisation d'interfaces métiers et le cycle de vie d'un bean d'entité.

Remarque : Le modèle de programmation EJB 2.1 (avec descripteurs de déploiement et interfaces Remote/Home) reste valide (et pris en charge par PowerAMC). Le nouveau modèle simplifié, qui est uniquement disponible avec Java 5.0, ne remplace pas entièrement le modèle EJB 2.1

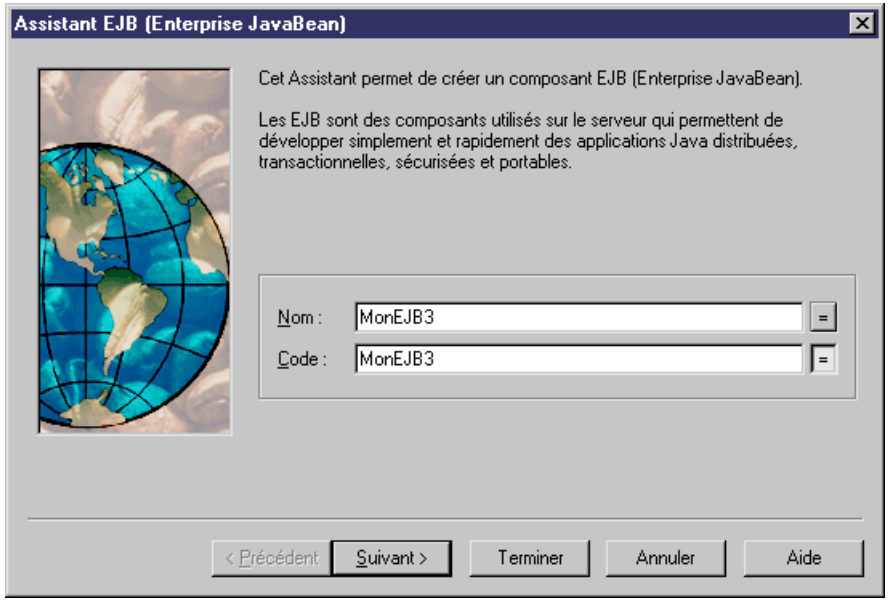
Création d'un EJB 3.0 avec l'Assistant Enterprise JavaBean

Pour créer un EJB3, lancez l'Assistant Enterprise JavaBean à partir d'un diagramme de classes.

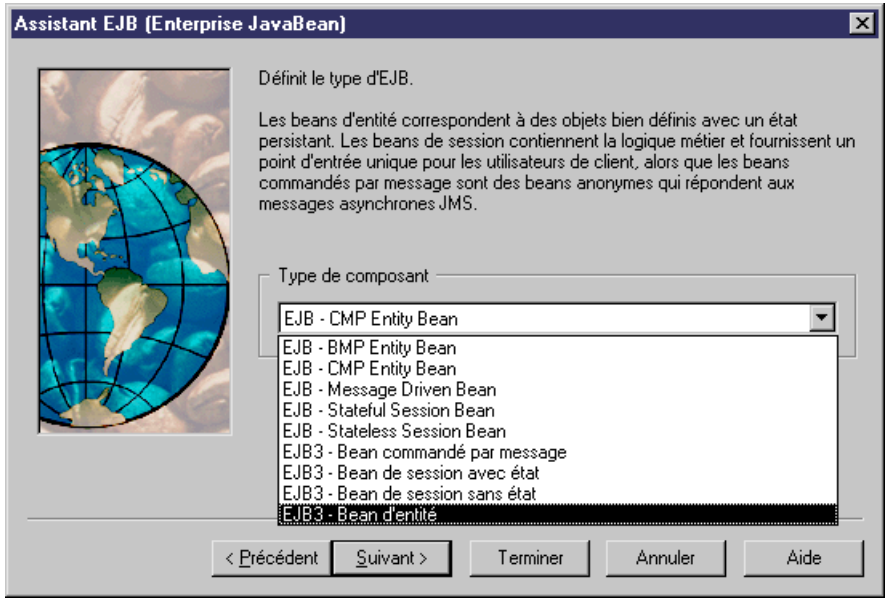
Les types de beans EJB3 sont disponibles :

- Bean d'entité – généré avec une annotation @Entity
- Bean commandés par message – généré avec une annotation @MessageDriven
- Bean de session avec état – généré avec une annotation @Stateful
- Bean de session sans état – généré avec une annotation @Stateless

1. Si vous avez déjà créé une classe devant servir comme classe Bean, pointez sur cette classe, cliquez le bouton droit de la souris, puis sélectionnez Créer un EJB (Enterprise JavaBean) dans le menu contextuel. Dans le cas contraire, pour créer un EJB 3.0 avec une nouvelle classe Bean, sélectionnez **Outils > Créer un EJB (Enterprise JavaBean)**. Dans les deux cas, l'Assistant EJB s'affiche :

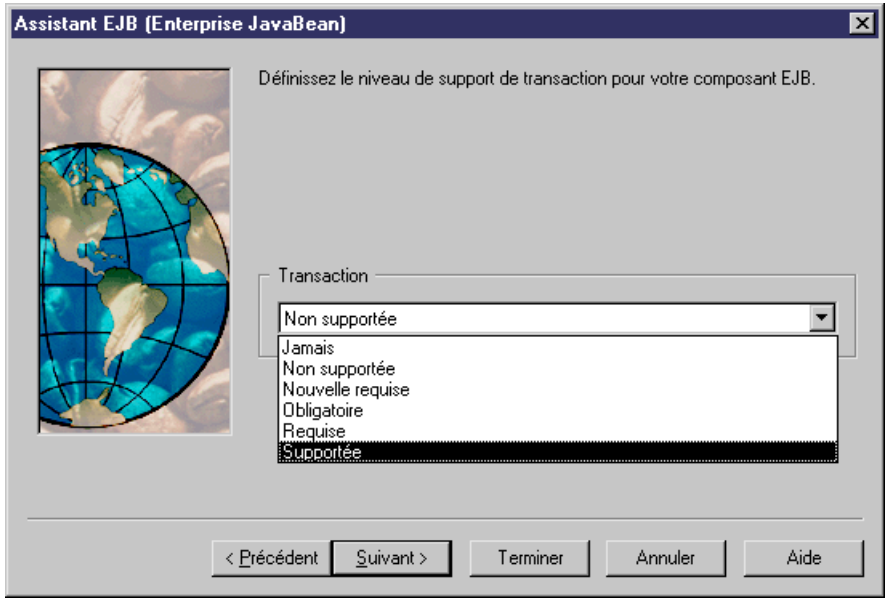


2. Spécifiez un nom pour l'EJB, puis cliquez sur Suivant pour afficher l'écran suivant :



3. Choisissez un type d'EJB3, puis cliquez sur Suivant pour afficher l'écran suivant :

4. Choisissez une classe Bean. Si vous n'avez pas sélectionné de classe avant de lancer l'Assistant, l'Assistant propose de créer une classe par défaut portant le même nom que le composant. Cliquez sur Suivant pour afficher l'écran suivant :



5. Choisissez le niveau de prise en charge de transaction souhaité, puis cliquez sur Suivant pour afficher l'écran suivant :

6. Cochez les cases appropriées si vous souhaitez créer des diagrammes pour le composant et/ou pour les classificateurs de composant, puis cliquez sur Terminer pour que PowerAMC les créé.

Propriétés d'une classe Bean EJB 3.0

Une classe de bean est la classe principale contenue dans un composant EJB 3.0. Les feuilles de propriétés de classe Bean EJB 3.0 contiennent tous les onglets d'une classe standard, ainsi que l'onglet EJB3.

L'onglet EJB3 contient les propriétés suivantes :

Propriété	Description
Gestion des transactions	Spécifie la méthode de gestion des transactions pour un bean de session ou un bean commandé par message. Vous pouvez choisir entre : • Bean • Container Généré sous la forme d'une annotation @TransactionManagement.
Type d'attribut de transaction	Type d'attribut de transaction pour la classe Bean Spécifie le type d'attribut de transaction pour un bean de session ou un bean commandé par message. Vous pouvez choisir entre : • Not Supported • Supports • Required • Requires New • Mandatory • Never Généré sous la forme d'une annotation @TransactionAttribute.
Exclure les intercepteurs par défaut	Spécifie que l'appel des méthodes d'intercepteur par défaut est exclu. Généré sous la forme d'une annotation@ExcludeDefaultInterceptors.
Exclure les listeners de classe parent	Spécifie que l'appel des méthodes de listeners de classe parent est exclu. Généré sous la forme d'une annotation @ExcludeSuperclassListeners.
Nom mis en correspondance	Spécifie un nom spécifique au produit. Généré sous la forme d'une annotation @MappedName.
Run-As	Spécifie la propriété run-as du bean (rôle de sécurité). Généré sous la forme d'une annotation @RunAs.
Rôles déclarés	Spécifie des références aux rôles de sécurité. Généré sous la forme d'une annotation @DeclareRoles.

Propriété	Description
Rôles admis	Spécifie les rôles admis pour toutes les méthodes de bean. Généré sous la forme d'une annotation @RolesAllowed.
Permettre tout	Spécifie que tous les rôles sont admis pour toutes les méthodes métiers de bean. Généré sous la forme d'une annotation @PermitAll.

Propriétés d'un composant EJB 3.0

Les feuilles de propriétés de composant EJB 3.0 contiennent tous les onglets de propriétés d'un composant standard, avec en plus l'onglet EJB.

L'onglet EJB contient les propriétés suivantes :

Propriété	Description
Classe Bean	Spécifie la classe bean associée.
Interface Remote Home	[beans de session uniquement] Spécifie une interface Remote Home facultative (pour les clients EJB plus anciens).
Interface Local Home	[beans de session uniquement] Spécifie l'interface Local Home (pour les clients EJB plus anciens).

Ajout d'interfaces et de classes supplémentaires à l'EJB

Outre la classe bean et les interfaces Remote et Home définies sur l'onglet EJB, vous pouvez lier des classes et interfaces supplémentaires à l'EJB3.

Vous pouvez lier les classes et interfaces supplémentaires à suivantes à l'EJB3 :

- Une collection facultative d'interfaces Remote avec le stéréotype <<EJBRemote>>. Généré sous la forme d'une annotation @Remote.
- Une collection facultative d'interfaces Local avec le stéréotype <<EJBLocal>>. Généré sous la forme d'une annotation @Local.
- Une collection facultative de classes d'interception avec le stéréotype <<EJBInterceptor>>. Généré sous la forme d'une annotation @Interceptors.
- Une collection facultative de classes de listener d'entité <<EJBEntityListener>>. Généré sous la forme d'une annotation @EntityListeners.

Vous pouvez ajouter ces interfaces et classes dans le composant EJB3 via les onglets Interfaces et Classes. Par exemple, vous pouvez ajouter une interface <<EJBInterceptor>> à un EJB3 :

1. Affichez la feuille de propriétés d'un EJB3 et cliquez sur l'onglet Interfaces.
2. Cliquez sur l'outil Créer un nouvel objet pour créer une nouvelle interface et afficher sa feuille de propriétés.

3. Sur l'onglet Général, sélectionnez <<EJBInterceptor>> dans la liste des stéréotypes.
4. Renseignez les propriétés restantes et cliquez sur OK pour revenir à la feuille de propriétés de l'EJB3.

Propriétés d'une opération EJB 3.0

Les feuilles de propriétés d'opération EJB 3.0 contiennent tous les onglets de feuille de propriétés d'opération standard avec en plus l'onglet EJB3.

L'onglet EJB3 contient les propriétés suivantes :

Propriété	Description
Méthode Initialize	Spécifie une méthode initialize. Généré sous la forme d'une annotation @Init.
Méthode Remove	Spécifie une méthode remove. Généré sous la forme d'une annotation @Remove.
Post-Construct	Spécifie une méthode post construct. Généré sous la forme d'une annotation @PostConstruct.
Post-Activate	Spécifie une méthode post activate. Généré sous la forme d'une annotation @PostActivate.
Pre-Passivate	Spécifie une méthode pre passivate. Généré sous la forme d'une annotation @PrePassivate.
Pre-Destroy	Spécifie une méthode pre destroy. Généré sous la forme d'une annotation @PreDestroy.
Méthode interceptrice	Spécifie une méthode interceptrice. Généré sous la forme d'une annotation @AroundInvoke.
Méthode Timeout	Spécifie une méthode timeout. Généré sous la forme d'une annotation @Timeout.
Exclure les intercepteurs par défaut	Exclut l'appel d'intercepteurs par défaut pour la méthode. Généré sous la forme d'une annotation @ExcludeDefaultInterceptors.
Exclure les intercepteurs de classe	Exclut l'appel d'intercepteurs de niveau classe pour la méthode. Généré sous la forme d'une annotation @ExcludeClassInterceptors.
Type d'attribut de transaction	Spécifie un type d'attribut de transaction pour la méthode. Généré sous la forme d'une annotation @TransactionAttribute.

Propriété	Description
Permettre tous les rôles	Spécifie que tous les rôles sont permis pour la méthode. Généré sous la forme d'une annotation @PermitAll.
Refuser tous les rôles	Spécifie que la méthode ne peut être appelée par un rôle de sécurité. Généré sous la forme d'une annotation @DenyAll.
Rôles admis	Spécifie les rôles admis pour la méthode. Généré sous la forme d'une annotation @RolesAllowed.

Servlets Java

Les servlets sont des programmes qui aident à la construction d'applications générant des pages Web dynamiques (HTML, XML). Les servlets sont les équivalents Java des scripts CGI qui sont des homologues côté serveur des applets Java situés côté client.

Les servlets sont des classes Java qui mettent en oeuvre une interface spécifique et produisent du code HTML en réponse à des demandes GET/POST.

Dans un MOO, un servlet est représenté sous forme de composant et est lié à une classe de servlet qui met en oeuvre l'interface requise et assure la mise en oeuvre du servlet.

Lorsque vous choisissez le type Servlet pour un composant, la classe servlet appropriée est automatiquement créée, ou bien attachée si elle existe déjà. La classe servlet est initialisée de sorte que les opérations y soient automatiquement ajoutées.

Page Servlet du composant

Lorsque vous sélectionnez comme type de composant Servlet, l'onglet Servlet est automatiquement affichée dans la feuille de propriétés du composant.

L'onglet Servlet dans la feuille de propriétés du composant inclut les propriétés suivantes :

Propriété	Description
Classe servlet	Classe qui met en oeuvre l'interface requise. Vous pouvez cliquer sur l'outil Propriétés en regard de cette zone pour afficher la feuille de propriétés de la classe, ou bien sur l'outil Créer pour créer une classe
Type de servlet	HttpServlet prend en charge le protocole Http, il s'agit du protocole le plus couramment utilisé. GenericServlet étend la classe générique de servlet. User-defined implique une personnalisation car ce type ne met en oeuvre aucun élément. Les méthodes varient si vous changez la valeur de type de servlet

Définition de classes de servlet

Les classes Servlet sont identifiées à l'aide du stéréotype <<ServletClass>>.

Le nom de classe servlet est synchronisé avec le nom du composant conformément à la convention spécifiée dans l'entrée Settings/Namings/ServletClassName dans la définition du langage objet Java.

Création d'un servlet à l'aide de l'Assistant

Vous pouvez créer un servlet à l'aide de l'*Assistant* qui vous guide au long des différentes étapes nécessaires à la création d'un servlet. Cet Assistant ne peut être appelé qu'à partir d'un *diagramme de classes* et n'est disponible que si le langage objet du modèle est Java.

Vous pouvez créer un servlet sans sélectionner de classe, ou bien commencer par sélectionner une classe puis lancer l'Assistant à partir du menu contextuel de cette classe.

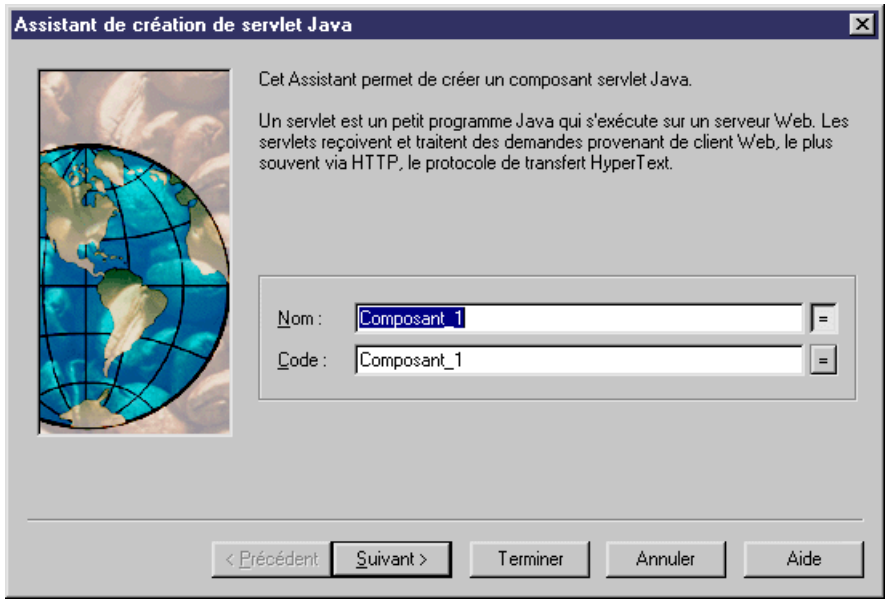
Vous pouvez également créer plusieurs servlets du même type en sélectionnant plusieurs classes à la fois. L'Assistant crée alors un servlet par classe. Les classes que vous avez sélectionnées dans le diagramme de classes deviennent alors des classes servlet. Elles sont renommées pour être mises en conformité avec les conventions de dénomination et sont liées au nouveau composant servlet.

L'Assistant de création de servlet permet de définir les paramètres suivants :

Paramètre de l'Assistant	Description
Nom	Nom du composant servlet
Code	Code du composant servlet
Type de servlet	Vous pouvez sélectionner l'un des types suivants : HttpServlet qui prend en charge le protocole Http (le plus couramment utilisé), GenericServlet qui étend la classe servlet générique, ou User-Defined qui implique une personnalisation car il ne met en oeuvre aucun élément
Classe servlet	Classe qui met en oeuvre le servlet.
Créer un symbole dans	Crée un symbole de composant dans le diagramme de composants spécifié dans la liste. S'il existe déjà un diagramme de composants, vous pouvez le sélectionner dans la liste. Vous pouvez également afficher la feuille de propriétés du diagramme sélectionné en cliquant sur l'outil Propriétés
Créer un diagramme de classes pour les classificateurs du composant	Crée un diagramme de classes avec un symbole pour chaque classe associée au composant. Si vous avez sélectionné des classes avant de démarrer l'Assistant, elles sont utilisées pour créer le composant. Cette option permet d'afficher ces classes dans un diagramme

1. Sélectionnez **Outils > Créer un Servlet** à partir d'un diagramme de classes.

   La boîte de dialogue Assistant de création de Servlet Java s'affiche.



   Remarque : Si vous avez sélectionné plusieurs classes avant de démarrer l'Assistant, certaines étapes ne s'affichent pas car les noms sont créés par défaut à partir du nom des classes sélectionnées.

2. Sélectionnez un nom et un code pour le composant, puis cliquez sur Suivant.
3. Sélectionnez un type de servlet, puis cliquez sur Suivant.
4. Sélectionnez le nom de la classe servlet, puis cliquez sur Suivant.
5. A la fin de l'Assistant, vous devez définir la création des symboles et diagrammes.

Lorsque vous avez fini d'utiliser l'Assistant, les actions suivantes sont exécutées :

- Un composant servlet est créé
- La classe servlet est créée et devient visible dans l'Explorateur d'objets. Elle est nommée d'après la classe d'origine si vous avez sélectionné une classe avant de démarrer l'Assistant
- Si vous n'avez pas sélectionné de classe avant de démarrer l'Assistant, le nom de la classe servlet prend comme préfixe celui du composant par défaut d'origine afin de préserver la cohérence
- Les éventuels diagrammes associés au composant sont créés ou mis à jour

Notions de base relatives à l'initialisation et à la synchronisation d'un servlet

Lorsque vous créez un servlet, le processus d'initialisation instancie la classe servlet ainsi que ses méthodes.

Le rôle de la synchronisation est de préserver la cohérence du modèle lorsqu'une modification est effectuée. Cette synchronisation s'effectue progressivement entre classes déjà attachées à un composant. PowerAMC effectue plusieurs actions pour compléter la synchronisation lorsque le modèle est modifié.

L'initialisation et la synchronisation de la classe servlet fonctionne de façon similaire aux classes Bean commandés par message.

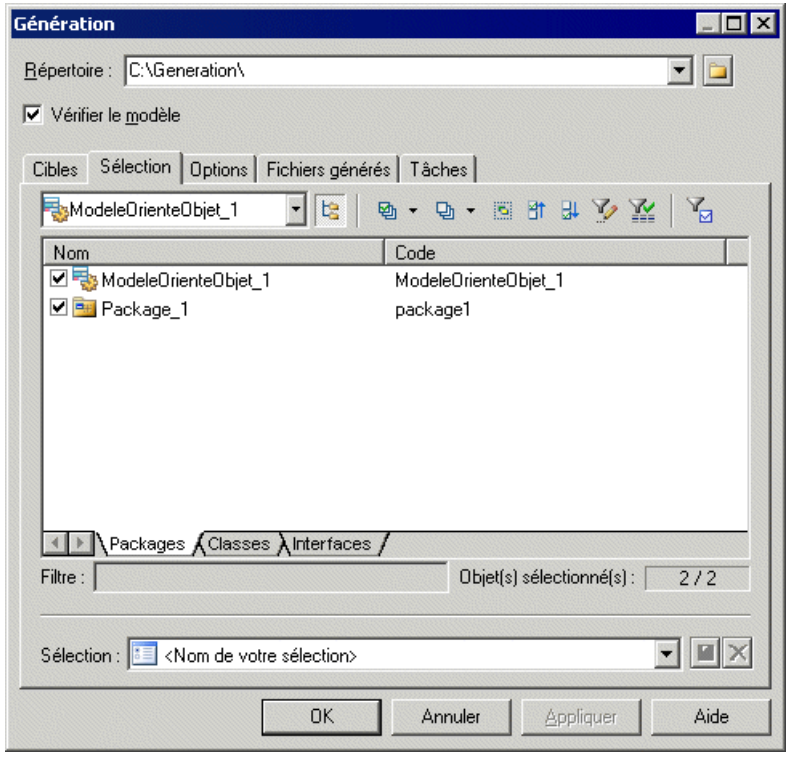
- Lorsque la classe servlet est attachée à un composant servlet, les méthodes de mise en oeuvre pour les opérations définies dans l'interface javax.servlet.Servlet sont ajoutées par défaut. Cette interface constitue l'interface de base pour tous les servlets et peut être ou non incluse dans le code en fonction de la valeur sélectionnée pour le type de servlet. Dans le cas de HttpServlet et de GenericServlet, la classe servlet étant directement dérivée d'une classe qui la met en oeuvre, elle n'a pas besoin de remettre en oeuvre l'interface. A l'inverse, dans le cas de servlets définis par l'utilisateur (type User-defined), cette interface est mise en oeuvre. Vous pouvez le voir dans l'onglet Aperçu de la classe servlet.
- Les méthodes de mise en oeuvre sont supprimées lorsque la classe est détachée si leur corps n'a pas été modifié
- Le jeu de méthodes prédéfinies varie en fonction du type de servlet

Vous pouvez utiliser à tout moment la fonctionnalité de vérification de modèle pour valider votre modèle et compléter la synchronisation. Pour ce faire, sélectionnez **Outils > Vérifier le modèle**.

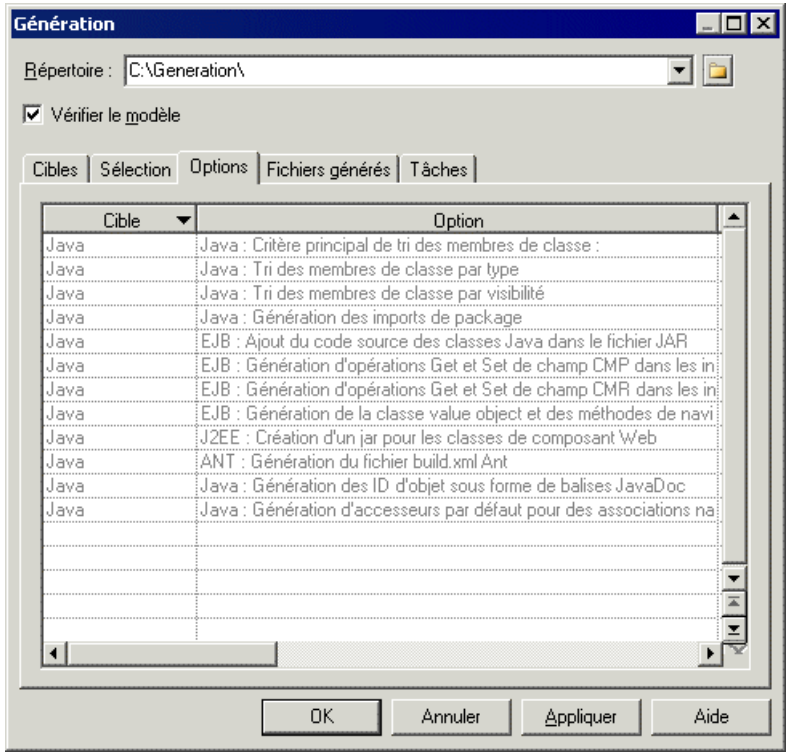
Génération de servlets

Le processus de génération récupère toutes les classes utilisées par les composants servlet pour générer les servlets.

1. Sélectionnez **Langage > Générer du code Java** pour afficher la boîte de dialogue Génération.
2. Spécifiez le répertoire dans lequel les fichiers doivent être générés, et indiquez si vous souhaitez effectuer une vérification de modèle (voir *Chapitre 9, Vérification d'un MOO* à la page 303).
3. Cliquez sur l'onglet **Sélection**, puis sélectionnez les objets appropriés sur les différentes pages.

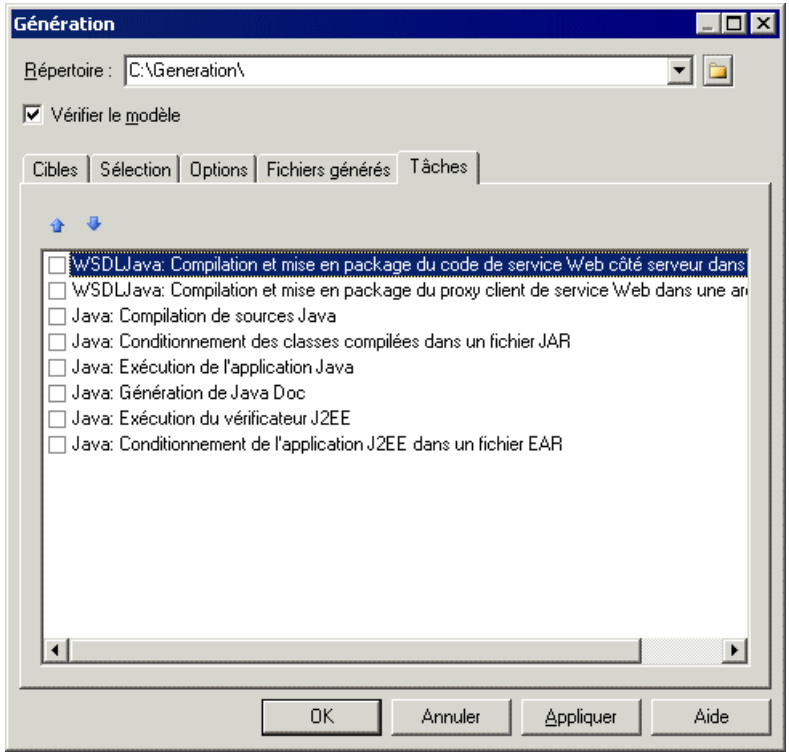


4. Cliquez sur **Appliquer**.
5. Cliquez sur l'onglet **Options**, puis spécifiez vos options de génération.



Pour plus d'informations sur les options de génération, voir *Génération de fichiers Java* à la page 416.

6. Cliquez sur **Appliquer**.
7. Cliquez sur l'onglet **Tâches**, puis sélectionnez les commandes à exécuter lors de la génération.



Pour plus d'informations sur les tâches de génération, voir *Génération de fichiers Java* à la page 416.

Vous devez avoir auparavant défini vos variables d'environnement en sélectionnant **Outils > Options générales > Variables** pour les activer dans cette page.

Pour plus d'informations sur la définition de ces variables, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Personnalisation de votre environnement de modélisation > Options générales > Variables d'environnement*.

8. Cliquez sur **OK** pour fermer la boîte de dialogue.

   Une boîte de progression s'affiche, suivie par une liste de résultats. Vous pouvez utiliser le bouton **Editer** dans la liste Résultats pour éditer chaque fichier généré individuellement.

9. Cliquez sur **OK** pour lancer la génération.

   Le fichier web.XML est créé dans le répertoire WEB-INF et tous les fichiers sont générés dans le répertoire spécifié.

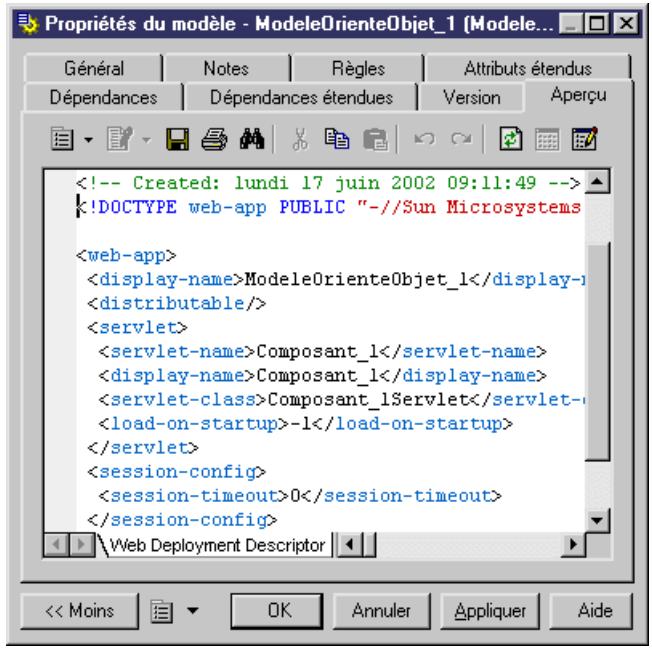
Génération d'un descripteur de déploiement de servlet

Le descripteur de déploiement Web est un fichier XML, appelé web.XML. Il est généré dans le répertoire WEB-INF et est indépendant du serveur d'applications.

Pour plus d'informations sur la génération d'un descripteur de déploiement Web, voir *Génération de servlets* à la page 401.

Le descripteur de déploiement d'application Web est généré pour chaque package. Une commande WAR disponible dans l'onglet Tâches de la boîte de dialogue de génération permet de construire une archive Web contenant le descripteur de déploiement Web, en plus de toutes les classes et interfaces référencées par les composants servlet. Au niveau modèle, une commande EAR est également fournie pour grouper tous les fichiers WAR et JAR générés pour un modèle particulier au sein d'une seule archive. Le EAR contient un descripteur de déploiement supplémentaire généré pour chaque modèle et appelé application.XML.

Le descripteur de déploiement Web contient plusieurs servlets à déployer, il est disponible à partir de l'onglet Aperçu du package, ou de la feuille de propriétés du modèle.



Génération de fichiers WAR

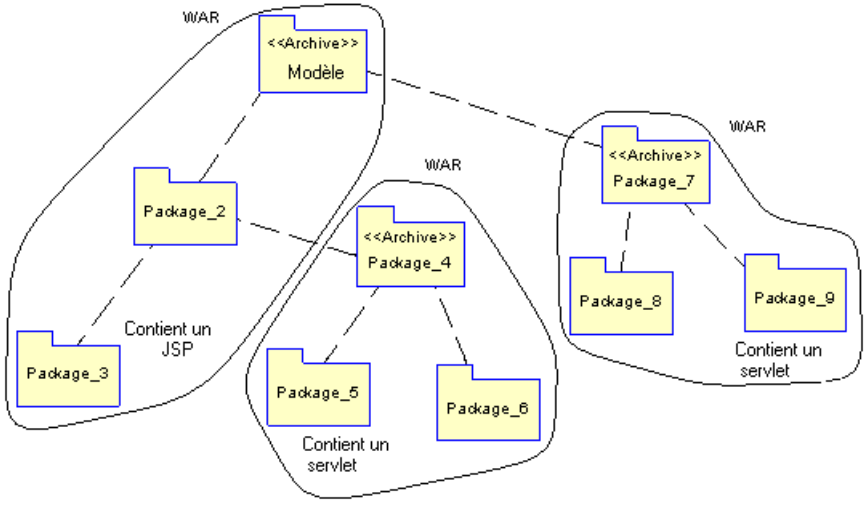
Vous pouvez incorporer des servlets et des JSP dans un fichier .WAR.

Vous pouvez générer des fichiers .WAR à partir de l'onglet **Tâches** de la boîte de dialogue de génération (**Langage > Générer du code Java**).

Il n'existe pas de contrainte portant sur la génération d'un fichier WAR par package. Seuls les packages ayant le stéréotype <<archive>> donnent lieu à la génération d'un fichier WAR s'ils contiennent (ou si l'un de leurs packages descendants n'ayant pas comme stéréotype <<archive>> contient) un JSP.

Le nouveau fichier archive créé contient le package et tous ses descendants non stéréotypés. Le package racine (qui est le modèle) est toujours considéré comme stéréotypé <<archive>>.

Par exemple, si un modèle contient plusieurs composants Web situés dans différents sous-packages mais qu'aucun de ces packages n'est stéréotypé <<archive>>, un seul fichier WAR est créé et il contient tous les packages.



Pour plus d'informations sur la génération des fichiers WAR, voir *Génération de fichiers Java* à la page 416.

Reverse engineering de servlets

Vous pouvez procéder au reverse engineering vers un MOO du code et du descripteur de déploiement d'un servlet. La fonctionnalité de reverse engineering traite la classe Java comme une classe servlet, récupère le servlet comme un composant, et associe la classe servlet à ce composant. La fonctionnalité de reverse engineering récupère également le descripteur de déploiement et tous les fichiers contenus dans le WAR.

Vous commencer les reverse engineering des servlets en sélectionnant **Langage > Reverse engineering Java**.

Sélectionnez ensuite l'un des formats Java suivants dans l'onglet Sélection, puis cochez la case Reverse engineering de descripteur de déploiement dans l'onglet Options :

- Répertoires Java

- Répertoires de classes
- Archive (fichier .JAR)

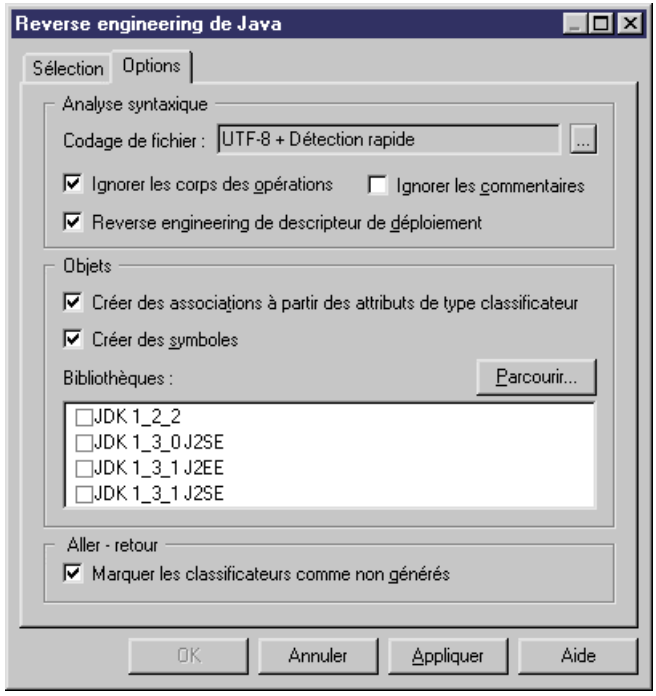
Pour plus d'informations sur les formats Java, voir *Reverse engineering de code Java* à la page 421.

1. Sélectionnez **Langage > Reverse engineering Java**.

   La boîte de dialogue Reverse engineering Java s'affiche.
2. Sélectionnez Archive dans la liste Reverse sur l'onglet **Sélection**.
3. Cliquez sur le bouton **Ajouter**.

   Une boîte de dialogue standard d'ouverture de fichiers s'affiche.
4. Sélectionnez les éléments sur lesquels vous souhaitez faire porter le reverse engeering, puis cliquez sur **Ouvrir**.

   La boîte de dialogue Reverse engineering Java affiche les éléments sélectionnés.
5. Cliquez sur l'onglet **Options**, puis cochez la case **Reverse engineering de descripteur de déploiement**.



6. Cliquez sur **OK**.

   Une boîte de progression s'affiche. Si le modèle vers lequel vous effectuez le reverse engineering contient déjà des données, la boîte de dialogue Fusion de modèle s'affiche.

Pour plus d'informations sur la fusion de modèles, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Comparaison et fusion de modèles*.

7. Cliquez sur **OK**.

L'onglet Reverse de la fenêtre Résultats affiche les changements effectués lors du reverse engineering et le modèle mis à jour est affiché dans la fenêtre de diagramme.

Java Server Pages (JSP)

Un JSP (Java Server) est une page Web HTML qui contient des bits de code supplémentaires qui exécutent une logique d'application afin de générer un contenu dynamique.

Dans PowerAMC, un JSP est représenté sous forme d'objet fichier lié à un composant. Le type de composant JSP (Java Server Page) permet d'identifier ce composant. Les composants JSP sont liés à un seul objet fichier qui définit la page.

Lorsque vous affectez le type JSP à un composant, le fichier JSP approprié est automatiquement créé, ou bien attaché s'il existe déjà. Vous pouvez voir l'objet fichier JSP dans l'onglet Fichiers de la feuille de propriétés du composant.

Onglet JSP de la feuille de propriétés du composant

Lorsque vous affectez le type JSP à un composant, l'onglet JSP s'affiche automatiquement dans la feuille de propriétés du composant.

L'onglet JSP de la feuille de propriétés de composant contient les propriétés suivantes :

Propriété	Description
Fichier JSP	Objet fichier qui définit la page. Vous pouvez cliquer sur l'outil Propriétés en regard de cette zone pour afficher la feuille de propriétés de l'objet fichier, ou bien cliquer sur l'outil Créer pour créer un objet fichier
Template par défaut	Attribut étendu qui permet de sélectionner un template pour la génération. Son contenu peut être défini par l'utilisateur ou fourni par défaut

Pour modifier le contenu par défaut, éditez le langage objet courant en sélectionnant **Langage > Editer le langage objet courant**, puis modifiez l'élément suivant : Profile/FileObject/Criteria/JSP/Templates/DefaultContent<%is(DefaultTemplate)%>. Créez ensuite les templates et renommez-les DefaultContent<%is(<nom>)%> où <nom> correspond au nom de template DefaultContent correspondant.

Pour définir des templates DefaultContent supplémentaire pour les JSP, vous devez modifier le type d'attribut étendu JSPTemplate dans l'entrée Profile/Share/Extended Attribute Types et ajouter de nouvelles valeurs correspondant au nom respectif des nouveaux templates.

Pour plus d'informations sur la propriété Template par défaut, voir la définition de TemplateContent dans *Création d'un JSP à l'aide de l'Assistant* à la page 409.

Définition des objets fichier pour les JSP

Le contenu de l'objet fichier pour les JSP est basé sur un template spécial appelé DefaultContent défini par rapport à la métaclasse FileObject. Il est situé dans la catégorie Profile/FileObject/Criteria/JSP/Templates du langage objet Java. Ce lien vers le template est fourni comme base de travail, si vous éditez l'objet fichier, le lien vers le template est perdu, ce mécanisme s'apparente à celui en vigueur pour les corps d'opération par défaut.

Pour plus d'informations sur la catégorie Criteria, voir *Personnalisation et extension de PowerAMC > Fichiers d'extension > Critères (Profile)*.

Les fichiers Java Server Page sont identifiés par le stéréotype JSPFile. Le nom de page JSP est synchronisé avec le nom de composant JSP conformément aux conventions spécifiées dans la zone Valeur de l'entrée Settings/Namings/JSPFileName du langage objet Java.

Vous pouvez pointer sur un objet fichier, cliquer le bouton droit de la souris et sélectionner **Ouvrir avec > *éditeur de texte*** dans le menu contextuel pour afficher le contenu de ce fichier.

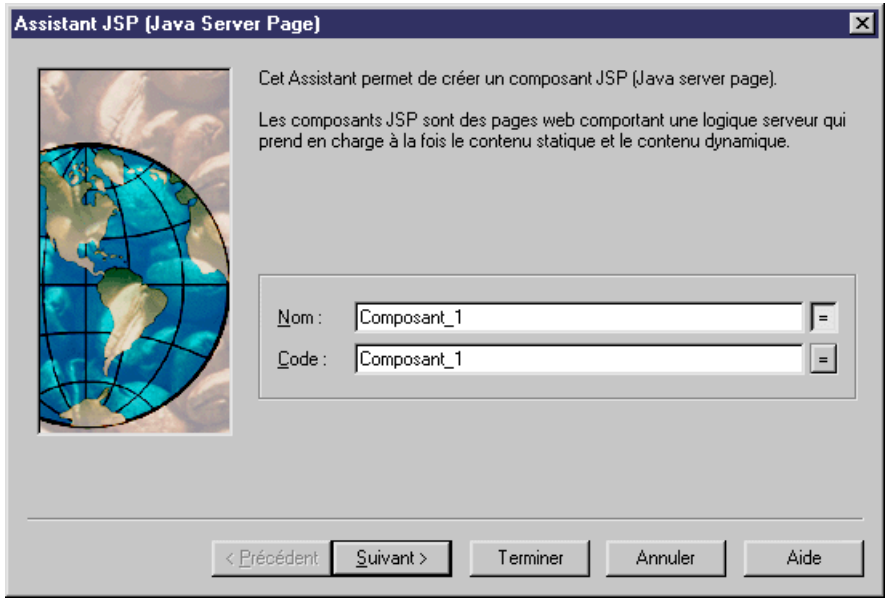
Création d'un JSP à l'aide de l'Assistant

Vous pouvez créer un JSP à l'aide de l'*Assistant* qui vous guide tout au long des différentes étapes de création du composant. Cet Assistant est appelé à partir d'un *diagramme de classes* et n'est disponible que si le langage est Java.

Vous pouvez soit créer un JSP sans sélectionner d'objet fichier, soit commencer par sélectionner un objet fichier, puis lancer l'Assistant à partir du menu **Outils**.

Vous avez également la possibilité de créer plusieurs JSP du même type en sélectionnant plusieurs objets fichier simultanément. L'Assistant crée alors automatiquement un JSP par objet fichier. Les objets fichier que vous avez sélectionnés dans le diagramme de classes deviennent les fichiers .JSP.

1. Sélectionnez **Outils > Créer un JSP** à partir d'un diagramme de classes.



2. Sélectionnez un nom et un code pour le composant JSP, puis cliquez sur **Suivant**.
3. Sélectionnez le template par défaut pour l'objet fichier JSP. `TemplateContent` est un attribut étendu situé dans la catégorie `Profile/Component/Criteria/J2EE-JSP` du langage objet Java. Si vous ne modifiez pas le contenu de l'objet fichier, le contenu par défaut est conservé. Tous les templates sont disponibles dans la catégorie **Profile/FileObject/Criteria/JSP/templates** du langage objet Java.
4. Cliquez sur **Suivant** pour aller à la page de fin de l'Assistant, sur laquelle vous pouvez sélectionnez **Créer un symbole** afin de créer un symbole de composant dans le diagramme spécifié.

Une fois que vous avez fini d'utiliser l'Assistant, les actions suivantes sont exécutées :

- Un composant JSP et un objet fichier avec une extension .JSP sont créés et visibles dans l'Explorateur d'objets. Le nom de l'objet fichier est défini d'après le nom du composant par défaut d'origine afin de préserver la cohérence.
- Si vous affichez la feuille de propriétés de l'objet fichier, vous pouvez voir que la propriété Artefact est sélectionnée.

  Pour plus d'informations sur les fichiers d'artefact, voir *Propriétés de l'objet fichier* à la page 236.
- Vous pouvez éditer l'objet fichier directement dans l'éditeur interne de PowerAMC, si le suffixe de nom de fichier correspond à un suffixe défini dans la page Editeurs de la boîte de dialogue Options générales, et si le mot clé <internal> est défini dans les colonnes Nom de l'éditeur et Commande de l'éditeur pour ce suffixe.

Génération de JSP

Le processus de génération ne prend en compte que les objets fichier dont la propriété Artefact est sélectionné.

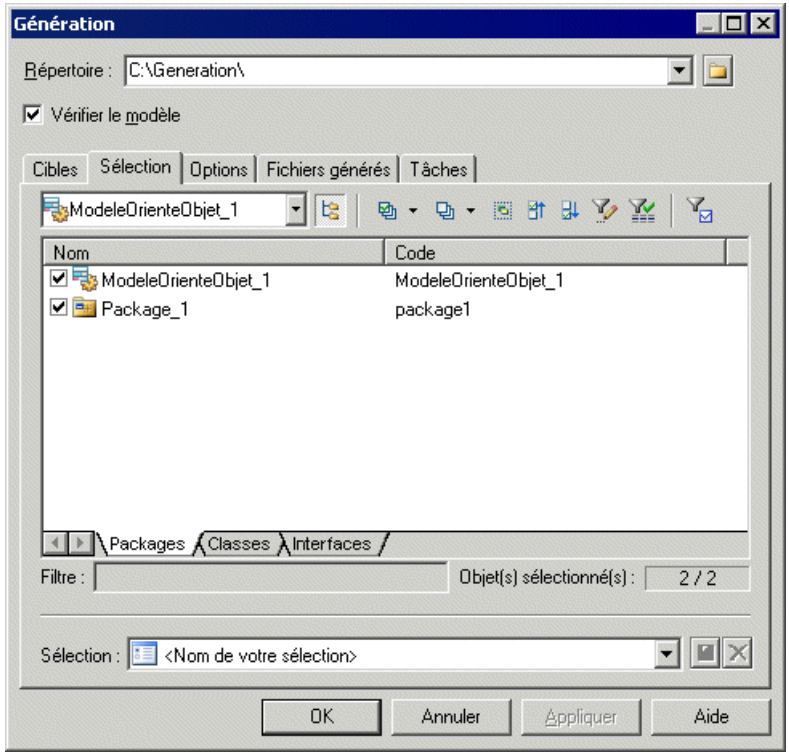
Génération d'un descripteur de déploiement Web de JSP

Le descripteur de déploiement Web est un fichier XML, appelé web.XML. Il est généré dans le répertoire WEB-INF. Il est indépendant du serveur d'applications.

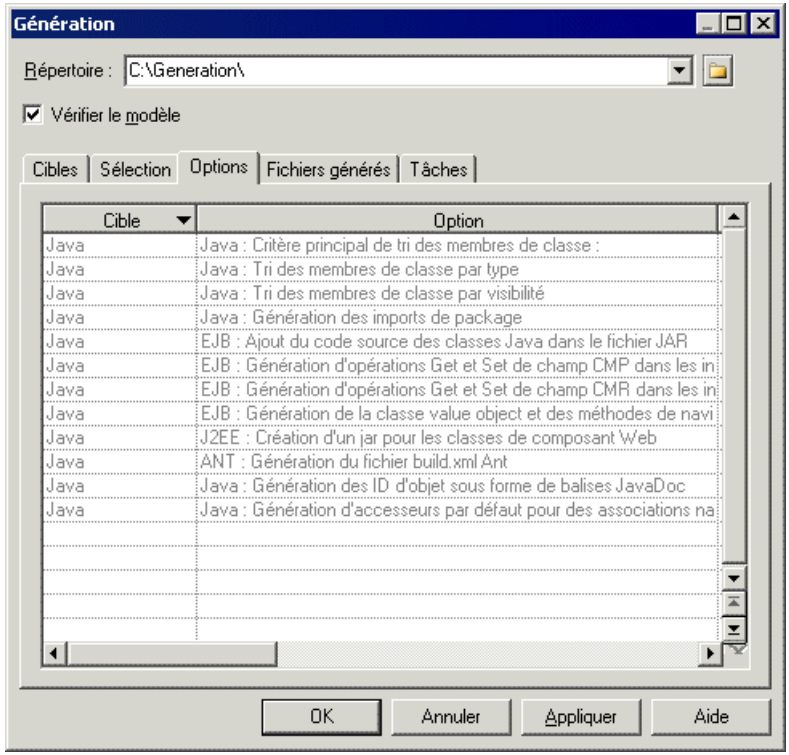
Le descripteur de déploiement d'application Web est généré pour chaque package. Une commande WAR disponible dans l'onglet Tâches de la boîte de dialogue de génération permet de construire une archive Web contenant le descripteur de déploiement Web, en plus de toutes les classes et tous les objets fichier référencés par les composants JSP. Au niveau modèle, une commande EAR fournie permet de grouper tous les fichiers WAR et JAR générés pour un modèle particulier au sein d'un même fichier archive. Le EAR contient un descripteur de déploiement supplémentaire généré pour chaque modèle et nommé application.XML.

Le descripteur de déploiement Web est disponible à partir de l'onglet Aperçu du package, ou de la feuille de propriétés du modèle.

1. Sélectionnez **Langage > Générer du code Java** pour afficher la boîte de dialogue Génération.
2. Spécifiez le répertoire dans lequel les fichiers doivent être générés, et indiquez si vous souhaitez effectuer une vérification de modèle (voir *Chapitre 9, Vérification d'un MOO* à la page 303).
3. Cliquez sur l'onglet **Sélection**, puis sélectionnez les objets appropriés sur les différentes pages.

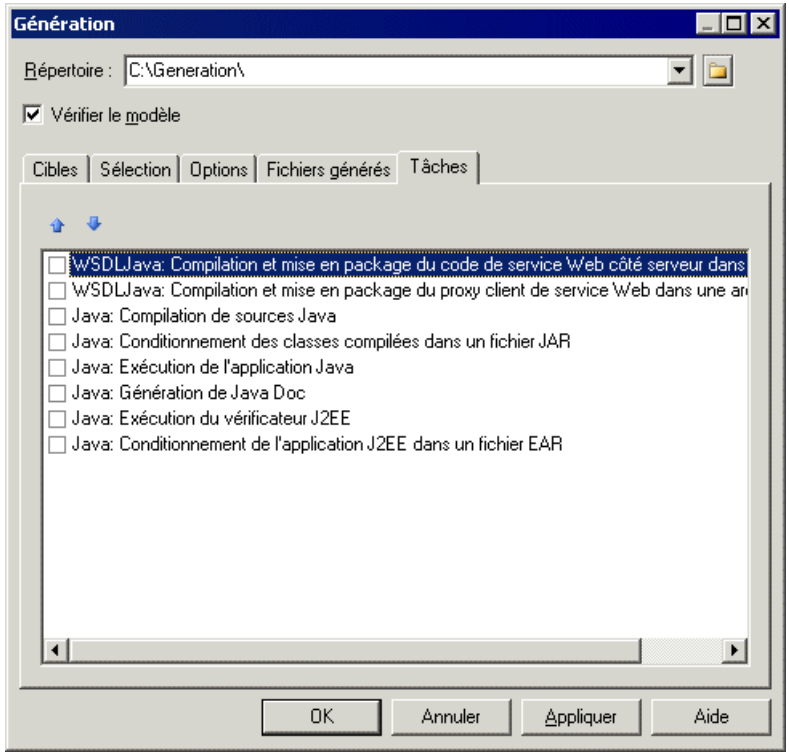


4. Cliquez sur **Appliquer**.
5. Cliquez sur l'onglet **Options**, puis spécifiez les options de génération sur la page **Options**.



Pour plus d'informations sur les options de génération, voir *Génération de fichiers Java* à la page 416.

6. Cliquez sur **Appliquer**.
7. Cliquez sur l'onglet **Tâches** pour l'afficher, puis sélectionnez les commandes à lancer lors de la génération sur la page **Tâches**.



Pour plus d'informations sur les tâches de génération, voir *Génération de fichiers Java* à la page 416.

Vous devez avoir préalablement défini les variables d'environnement via la section Variables de la boîte de dialogue Options générales afin que ces commandes soient activées dans cet onglet.

Pour plus d'informations sur la définition de ces variables, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Personnalisation de votre environnement de modélisation > Options générales > Variables d'environnement*.

8. Cliquez sur **OK** pour lancer la génération.

   Une boîte de progression s'affiche, suivie par une liste Résultats. Vous pouvez utiliser le bouton **Editer** dans la liste Résultats pour éditer chaque fichier généré individuellement.

9. Cliquez sur **Fermer**.

   Le fichier web.XML est créé dans le répertoire WEB-INF, et tous les fichiers sont générés dans le répertoire spécifié pour la génération.

Reverse engineering des JSP

Vous pouvez procéder au reverse engineering vers un MOO du code et du descripteur de déploiement d'un JSP. La fonctionnalité de reverse engineering récupère le contenu des fichiers pour créer des composants JSP et récupère le descripteur de déploiement stocké dans le WAR.

Vous pouvez commencer le reverse engineering de JSP en sélectionnant **Langage > Reverse engineering Java**. Sélectionnez l'un des formats Java suivants sur l'onglet **Sélection**, puis cochez la case **Reverse engineering de decripteur de déploiement** sur l'onglet **Options** :

- Répertoires Java
- Répertoires de classes
- Archive (fichier .JAR)

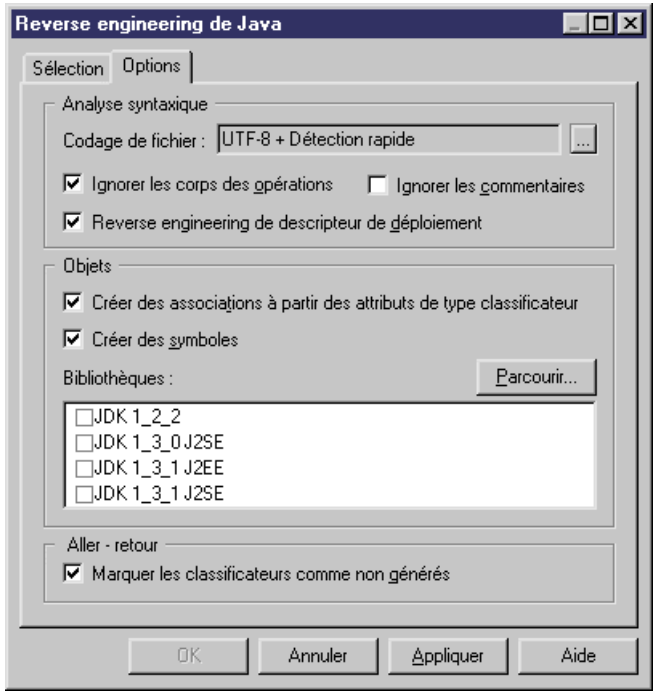
Pour plus d'informations sur les formats Java, voir *Reverse engineering de code Java* à la page 421.

1. Sélectionnez **Langage > Reverse engineering Java**.

   La boîte de dialogue Reverse engineering Java s'affiche.
2. Sélectionnez Archive dans la liste Reverse sur l'onglet **Sélection**.
3. Cliquez sur le bouton **Ajouter**.

   Une boîte de dialogue standard d'ouverture de fichiers s'affiche.
4. Sélectionnez les élément sur lesquels vous souhaitez faire porter le reverse engineering, puis cliquez sur **Ouvrir**.

   La boîte de dialogue Reverse engineering Java affiche les éléments sélectionnés.
5. Cliquez sur l'onglet **Options**, puis cochez la case **Reverse engineering de descripteur de déploiement**.



6. Cliquez sur **OK**.

   Une boîte de progression s'affiche. Si le modèle dans lequel vous effectuez un reverse engineering contient déjà des données, une boîte de dialogue de fusion s'affiche.

   Pour plus d'informations sur la fusion de modèles, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Comparaison et fusion de modèles*.

7. Cliquez sur **OK**.

   L'onglet Reverse de la fenêtre Résultats affiche les changements effectués lors du reverse engineering et le modèle mis à jour est affiché dans la fenêtre de diagramme.

Génération de fichiers Java

Vous générez des fichiers source Java à partir des classes et des interfaces d'un modèle. Un fichier distinct, portant le suffixe de nom de fichier .java, est généré pour chaque classe ou interface sélectionnée dans le modèle, ainsi qu'un fichier journal de génération. Vous ne pouvez générer des fichiers Java que pour un seul modèle à la fois.

Les variables PowerAMC suivantes sont utilisées dans la génération de fichiers source Java :

Variable	Description	Défaut
J2EEVERIF	Programme par lots permettant de vérifier si le JAR de déploiement pour un type EJB est correct	verifier.bat
JAR	Commande d'archivage des fichiers java	jar.exe
JAVA	Commande d'exécution des programmes JAVA	java.exe
JAVAC	Commande de compilation des fichiers source JAVA	javac.exe
JAVADOC	Commande de définition des commentaires Javadoc	javadoc.exe

Pour passer en revue ou éditer ces variables, sélectionnez **Outils > Options générales** puis cliquez sur la catégorie **Variables**. Par exemple, vous pouvez ajouter la variable JAVACLASSPATH de ce tableau afin passer outre la variable d'environnement CLASSPATH de votre système.

1. Sélectionnez **Langage > Générer du code Java** pour afficher la boîte de dialogue Génération.
2. Spécifiez le répertoire dans lequel les fichiers doivent être générés, et indiquez si vous souhaitez effectuer une vérification de modèle (voir *Chapitre 9, Vérification d'un MOO* à la page 303).
3. [facultatif] Sélectionnez des cibles supplémentaires pour la génération. Ces cibles sont définies par les extensions que vous pouvez attacher au modèle (voir *Gestion des cibles de génération* à la page 286).
4. [facultatif] Cliquez sur l'onglet **Sélection** et spécifiez les objets à partir desquels vous souhaitez générer. Par défaut, tous les objets sont générés.
5. [facultatif] Cliquez sur l'onglet **Options** et sélectionnez les options de génération appropriées :

Option	Description
Java : Critère principal de tri des membres de classe	Trie les attributs et les opérations par : • [valeur par défaut] Les attributs et opérations publics sont générés avant les attributs et opérations privés • Type - Les attributs et opérations sont triés par type, quelle que soit leur visibilité

Option	Description
Java : Tri des membres de classe par type	Trie les attributs et opérations dans l'ordre suivant : • Attributes – Operations - Les attributs sont générés avant les opérations • Operations – Attributes - Les opérations sont générées avant les attributs
Java : Tri des membres de classe par visibilité	Trie les attributs et opérations dans l'ordre suivant : • Public – Private - Les attributs et opérations publics sont générés avant les attributs et opérations privés • Private – Public - Les attributs et opérations privés sont générés avant les attributs et opérations publics • None - L'ordre des attributs et opérations n'est pas modifié

Java : Génération des imports de package	Lorsqu'une classe est utilisée par une autre classe, elle est référencée par un import de classe :
	`import package1.package2.class.`
	Cette options permet de déclarer l'importation de la totalité du package, ce qui permet de gagner du temps lorsque plusieurs classes du même package peuvent être référencées :
	`import package1.package2.*;`

Java : Génération des ID d'objet sous forme de balises JavaDoc	Génère les informations utilisées pour le reverse engineering, telles que les identificateurs d'objet (@pdoid), qui sont alors générées sous forme de balises de documentation. Si vous ne souhaitez pas que ces balises soient générées, vous devez définir cette option comme False
Java : Génération d'accesseurs par défaut pour des associations navigables	Génère les méthodes getter et setter pour les associations navigables
ANT : Génération du fichier build.xml Ant	Génère le fichier build.xml. Vous pouvez utiliser ce fichier si vous avez installé Ant
EJB : Génération d'opérations Get et Set de champ CMP dans les interfaces de composant	Génère des opérations Get et Set de champ CMP dans les interfaces EJB
EJB : Génération d'opérations Get et Set de champ CMR dans les interfaces de composant	Génère des opérations Get et Set de champ CMR dans les interfaces EJB
EJB : Ajout du code source des classes Java dans le fichier JAR	Inclut le code des classes Java dans le fichier JAR

Option	Description
EJB : Génération d'opérations Get et Set de champ CMR dans les interfaces de composant	Génère une classe supplémentaire appelée %Component. Code %ValueObject pour chaque de bean CMP et déclare tous les champs CMP en tant qu'attributs publics. En outre, des Get et des Set sont générés dans la classe bean pour chaque relation CMR
J2EE : Création d'un jar pour les classes de composant Web	Archive les classes de composant Web dans un fichier Jar

Remarque : Pour plus d'informations sur la modification des options qui s'affichent sur cet onglet et sur l'onglet **Tâches** ainsi que sur l'ajout de vos propres options et tâches, voir *Personnalisation et extension de PowerAMC > Fichiers de définition pour les langage objet, de processus et XML > Catégorie Generation*.

6. [facultatif] Cliquez sur l'onglet **Fichiers générés** et spécifiez les fichiers à générer. Par défaut, tous les fichiers sont générés.

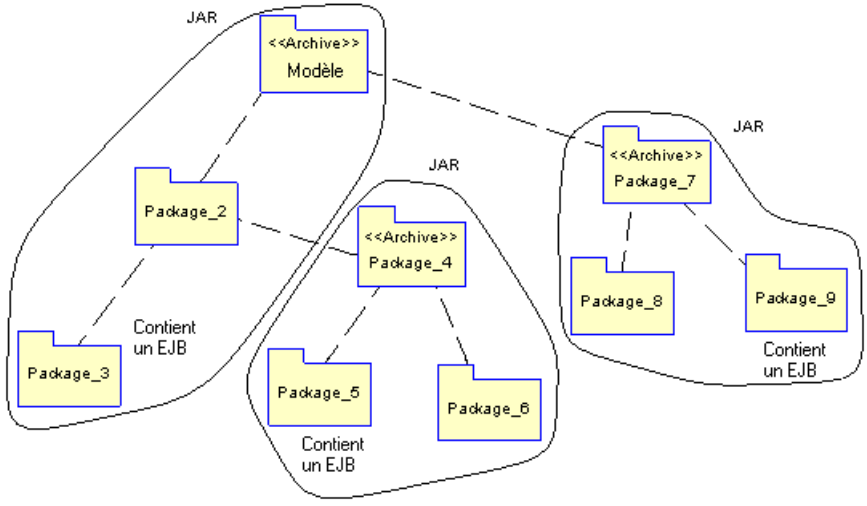
   Pour plus d'informations sur la personnalisation des fichiers qui seront générés, voir *Personnalisation et extension de PowerAMC > Fichiers d'extension > Fichiers générés (Profile)*.

7. [facultatif] Cliquez sur l'onglet **Tâches** et spécifiez les tâches de génération à effectuer :

Tâche	Description
Java : Compilation de sources Java	Démarre un compilateur à l'aide de la commande javac afin de compiler des fichiers source Java.
Java : Conditionnement de classes compilées dans un fichier JAR	Compile les fichiers source et les place dans un fichier JAR
Java : Lancement de l'application Java	Compile les fichiers source et exécute l'application Java à l'aide de la commande java
Java: Génération de Javadoc	Génère Javadoc
Java: Conditionnement de l'application J2EE dans un fichier EAR	Appelle les commandes de construction du source pour le composant EJB, de création d'un fichier JAR pour les classes Java et d'un descripteur de déploiement, de construction du code source pour le composant Web, de création d'un fichier EAR pour les classes de composant Web et d'un descripteur de déploiement, et de création d'un fichier d'archives EAR contenant tous les fichiers JAR/WAR générés

Tâche	Description
Java : Exécution du vérificateur J2EE	Appelle les commandes de construction du code source pour le composant EJB, de création d'un fichier JAR pour les classes Java et d'un descripteur de déploiement, de construction du code source pour le composant Web, de création d'un fichier WAR pour les classes de composant Web et d'un descripteur de déploiement, de création d'un fichier d'archives EAR contenant tous les fichiers JAR/WAR générés et de vérification J2EE des archives générées
WSDL : Compilation et mise en package du code de service Web côté serveur dans une archive	Appelle les commandes qui permettent de compiler le code source d'EJB et de composant Web, en exécutant l'outil de compilation WSCompile, en créant un fichier WAR pour les classe de composant Web et le descripteur de déploiement et en créant un fichier JAR pour les classes Java et le descripteur de déploiement
WSDL : Compilation et mise en package du code de proxy client de service Web dans une archive	Appelle les commandes qui permettent de compiler le code source d'EJB et de composant Web, en exécutant l'outil de compilation WSCompile, en créant un fichier WAR pour les artefacts côté client

Remarque : Les packages ayant le stéréotype <<archive>> génèrent un fichier JAR (quand ils ou un de leurs packages descendants n'ayant pas comme stéréotype <<archive>> contient un EJB) ou un fichier WAR (s'ils contiennent un servlet ou JSP). Chaque archive contient le package et tous ses descendants non-stéréotypés. Le modèle agit comme package racine et est considéré comme ayant le stéréotype <<archive>>.



8. Cliquez sur **OK** pour lancer la génération.

   Une fois la génération terminée, la boîte de dialogue Fichiers générés s'affiche et répertorie les fichiers générés dans le répertoire spécifié. Sélectionnez un fichier dans liste, puis

cliquez sur **Editer** pour l'ouvrir dans votre éditeur associé, ou bien cliquez sur **Fermer** pour quitter la boîte de dialogue.

Reverse engineering de code Java

Vous pouvez procéder au reverse engineering de fichiers contenant des classes Java vers un MOO. Pour chaque classe définie dans un fichier Java, une classe correspondante est créée dans le MOO, sous le même nom et avec les mêmes informations.

Lorsque vous effectuez le reverse engineering d'une classe Java qui existe déjà dans le modèle, vous pouvez choisir dans la fenêtre de fusion de remplacer la classe existante, ou de conserver sa définition dans le modèle.

1. Sélectionnez **Langage > Reverse engineering Java** pour afficher la boîte de dialogue Reverse engineering de Java.

2. Sélectionnez l'un des formats de fichiers suivants dans la liste Reverse engineering :

   - *Fichiers Java* (.java) - Fichiers contenant une ou plusieurs définitions de classe.
   - *Répertoires Java* - Dossiers contenant les fichiers Java. Tous les fichiers .java, y compris ceux contenus dans des sous-répertoires, feront l'objet du reverse engineering. Chaque sous-répertoire devient un package au sein du modèle. Les fichiers Java contenus dans un même répertoire étant souvent interdépendants, si vous ne procédez

pas au reverse engineering de la totalité des fichiers contenus dans le répertoire, votre modèle risque d'être incomplet.

- *Fichiers de classe* (.class) – Fichiers compilés contenant la définition d'une unique classe ayant le même nom que le fichier. Chaque sous-répertoire devient un package au sein du modèle.
- *Répertoires de classe* – Dossiers contenant des fichiers de classe. Tous les fichiers .class, y compris ceux contenus dans des sous-répertoires, font l'objet du reverse engineering.
- *Archives* (.zip, .jar) - Fichiers compressés contenant des définitions d'une ou de plusieurs classes. PowerAMC crée une classe pour chaque définition de classe dans le fichier .jar ou .zip. Les fichiers suivants ne sont pas concernés par le reverse engineering : manifest.mf, web.xml, ejb-jar.xml, et *.jsp. Les autres fichiers sont récupérés sous forme des fichiers avec la propriété Artefact définie à true de sorte qu'ils peuvent être générés ultérieurement. Les fichiers sont récupérés par reverse engineering dans des packages correspondant à la structure des répertoires existant dans l'archive.

3. Cliquez sur le bouton **Ajouter** pour sélectionner les fichiers et répertoires sur lesquels faire porter le reverse engineering, puis cliquez sur **Ouvrir** pour revenir à la boîte de dialogue Reverse engineering de Java, qui affiche maintenant les fichiers sélectionnés.

   Vous pouvez répéter cette étape autant de fois que nécessaire pour sélectionner les fichiers ou répertoires situés à des emplacements différents.

   Vous pouvez pointer sur n'importe quel fichier, cliquer le bouton droit de la souris et sélectionner **Editer** dans le menu contextuel pour afficher son contenu dans un éditeur.

4. [facultatif] Cliquez sur l'onglet **Options** et spécifiez les options de reverse engineering appropriées. Pour plus d'informations sur ces options, voir *Onglet Options de la boîte de dialogue Reverse engineering de Java* à la page 423

   Remarque : Reverse engineering sans le corps du code Vous pouvez effectuer le reverse engineering de fichiers .java sans le corps du code afin de les visualiser ou à des fins de comparaisons, voire pour limiter la taille de votre modèle si vous devez faire porter le reverse engineering sur un grand nombre de classes. Pour ce faire, cochez la case Ignorer les corps des opérations.

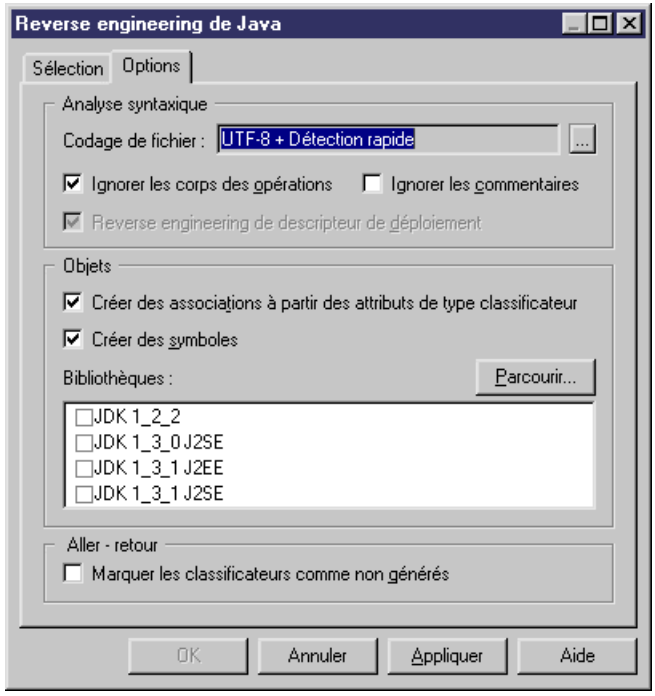
5. Cliquez sur **OK** pour lancer le processus de reverse engineering. Si le modèle dans lequel vous effectuez le reverse engineering contient déjà des données, la boîte de dialogue Fusion de modèles s'affiche pour vous permettre de spécifier si les objets existants doivent être écrasés.

   Pour plus d'informations sur la fusion des modèles, voir*Guide des fonctionnalités générales > Modélisation avec PowerAMC > Comparaison et fusion de modèles*.

   PowerAMC crée une classe dans le modèle pour chaque définition de classe dans les fichiers soumis au reverse engineering. Les classes sont visibles dans l'Explorateur d'objets et, par défaut, les symboles sont créés dans un ou plusieurs diagrammes.

Onglet Options de la boîte de dialogue Reverse engineering de Java

L'onglet Options permet de spécifier diverses options relatives au reverse engineering.



Vous pouvez définir les options de reverse engineering Java suivantes (notez que la disponibilité de certaines options peut varier en fonction du type des fichiers Java sur lesquels porte le reverse engineering) :

Option	Résultat
Codage de fichier	Permet de modifier le codage de fichier par défaut pour les fichiers sur lesquels vous souhaitez faire porter le reverse engineering
Ignorer les corps des opérations	Effectue le reverse engineering sans y inclure le corps du code. Cela peut s'avérer utile pour les visualiser ou à des fins de comparaisons, voire pour limiter la taille de votre modèle si vous devez faire porter le reverse engineering sur un grand nombre de classes
Ignorer les commentaires	Effectue le reverse engineering des classes sans y inclure les commentaires
Reverse engineering de descripteur de déploiement	Procède au reverse engineering des composants avec le descripteur de déploiement. Pour plus d'informations, voir *Reverse engineering de composants EJB* à la page 390, *Reverse engineering de servlets* à la page 406, and *Reverse engineering des JSP* à la page 415.

Option	Résultat
Créer des associations à partir des attributs de type classificateurs	Créer des associations entre les classes et/ou les interfaces
Créer des symboles	Crée un symbole pour chaque objet dans le diagramme. Si vous décochez cette case, les objets récupérés ne sont visibles que dans l'Explorateur d'objets
Bibliothèques	Spécifie une liste de modèles de bibliothèque à utiliser comme référence lors du reverse engineering. Le modèle de destination du reverse engineering peut contenir des raccourcis vers des objets définis dans une bibliothèque. Si vous spécifiez la bibliothèque ici, le lien entre le raccourci et son objet cible (contenu dans la bibliothèque) sera préservé et la bibliothèque sera ajoutée à la liste des modèles cibles dans le modèle de destination du reverse engineering. Vous pouvez faire glisser les bibliothèques dans la liste afin de spécifier une hiérarchie de bibliothèques. PowerAMC tentera de résoudre les raccourcis trouvés dans le modèle de destination du reverse engineering en utilisant tour à tour chacune des bibliothèques spécifiées. Ainsi, si la bibliothèque v1.1 apparaît dans la liste avant la bibliothèque v1.0, PowerAMC tentera d'abord de résoudre les raccourcis vers la bibliothèque v1.1 et analysera uniquement la bibliothèque v1.0 s'il reste des raccourcis non résolus. Vous devez utiliser la liste des modèles cibles pour gérer les bibliothèques liées au modèle de destination du reverse engineering, par exemple, vous pouvez changer la version de bibliothèque (voir *Guide des fonctionnalités générales > Liaison et synchronisation de modèles > Raccourcis et répliques > Utilisation des modèles cible*).
Marquer les classificateurs comme n'étant pas à générer	Les classificateurs récupérés par reverse engineering (classes et interfaces) ne seront ensuite plus générés à partir du modèle. Pour pouvoir les générer, vous devez cochez la case Générer dans leur feuille de propriétés respective

Reverse engineering des commentaires du code Java

Lorsque vous procédez au reverse engineering des fichiers Java, certains commentaires peuvent changer de forme ou de position au sein du code.

Commentaire dans le fichier Java d'origine	A l'issue du reverse engineering
Avant les déclarations d'import	Est transféré dans l'en-tête.
Commence par /*	Commence par //.
A la fin du fichier, tout à la fin du code	Est transféré à la fin du code.

Commentaire dans le fichier Java d'origine	A l'issue du reverse engineering
Au sein d'une classe, mais pas au sein d'une opération	Est attaché à l'attribut ou à l'opération qui le suit directement.

Chapitre 13 Technical Architecture Modeling (TAM)

Technical Achitecture Modeling est le standard interne SAP pour la modélisation d'architecture qui combine des éléments de FMC et de UML.

PowerAMC prend en charge la modélisation des diagrammes TAM suivants :

- Diagramme de cas d'utilisation - voir *Chapitre 2, Diagrammes de cas d'utilisation* à la page 17.
- Diagramme de classes - voir *Diagrammes de classes* à la page 29.
- Diagramme de packages - voir *Diagrammes de packages* à la page 33.
- Diagramme de séquence - voir *Diagrammes de séquence* à la page 134.
- Diagramme d'activités - voir *Diagrammes d'activités* à la page 137.
- Diagramme d'états-transitions - voir *Diagrammes d'états-transitions* à la page 140.
- Diagramme de composants - voir *Diagrammes de composants* à la page 219.
- Diagramme de blocs - voir *Diagrammes de blocs (TAM)* à la page 427.

Pour créer un MOO TAM, utilisez la catégorie Technical Architecture Modeling (TAM) sur l'onglet **Catégories** de la boîte de dialogue Nouveau modèle, ou créez un MOO ciblant le langage objet Technical Architecture Modeling (TAM) sur l'onglet **Types de modèle**.

Diagrammes de blocs (TAM)

Les diagrammes de blocs montrent la structure de composition de tout système traitant des informations et illustrent la façon dont les agents accèdent aux données dans les stockages et communiquent via des canaux. PowerAMC prend en charge les diagrammes de blocs sous la forme de diagrammes de composants dotés d'outils TAM supplémentaires.

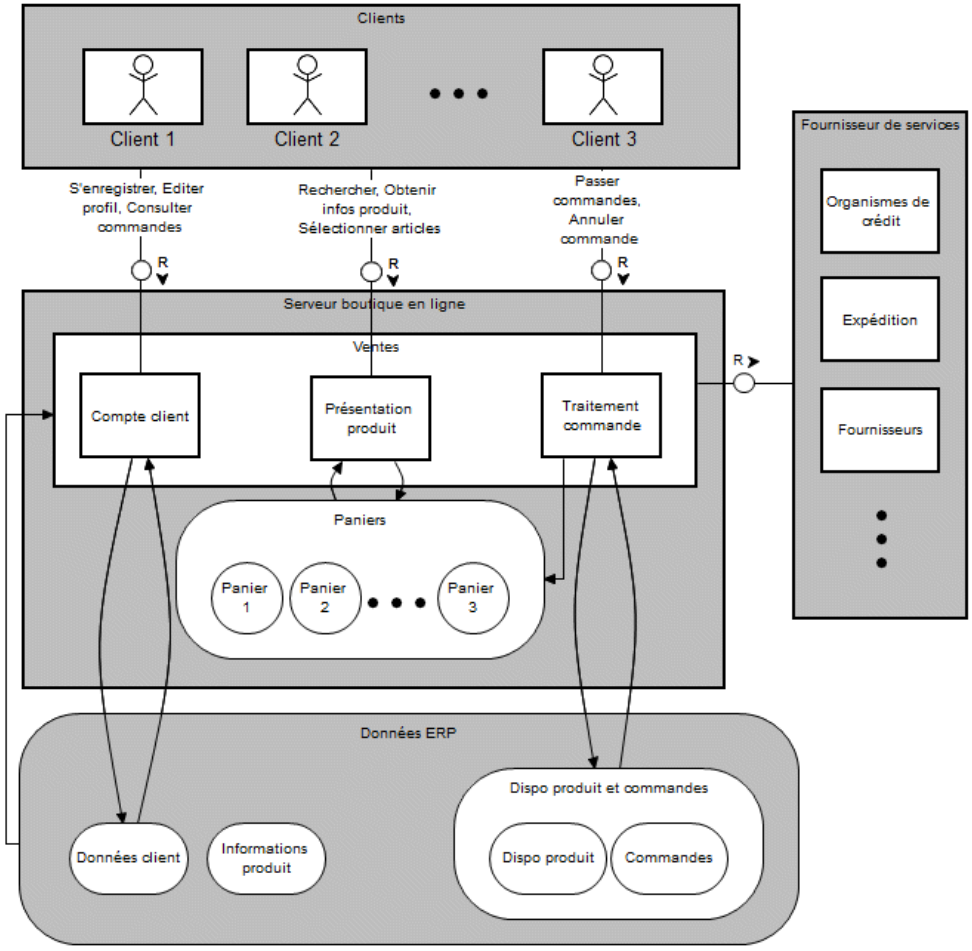
Remarque : Pour créer un diagramme de blocs dans un MOO TAM existant, pointez sur le modèle dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Diagramme de composants**. Pour créer un nouveau modèle, sélectionnez **Fichier > Nouveau modèle**, cliquez sur le bouton Catégories, sélectionnez la catégorie Technical Architecture Modeling (TAM), cliquez sur Diagramme de blocs, spécifiez un nom de modèle, puis cliquez sur **OK**.

Les outils suivants sont disponibles dans la boîte à outils Technical Architecture Modeling (TAM) dans un diagramme de blocs :

Outil	Description
	Agents, agents humains - Représentent les parties actives du système. Seuls les agents peuvent faire quelque chose, les autres éléments d'un diagramme de blocs sont passifs. Vous pouvez imbriquer des agents et des agents humains dans des agents. Les agents humains ne permettent en revanche pas l'imbrication. Vous pouvez utiliser l'imbrication pour montrer la structure interne des agents et des stockages. Vous pouvez grouper des éléments qui appartiennent à un système (partie), qui sont exécutés sur une plateforme, qui sont situés sur la même base de données, etc, afin de réduire le nombre de canaux entre les agents, ou les accès entre agents et stockages.
	Stockages - Contiennent des informations qui peuvent être traitées par un agent. Pour spécifier le type du stockage et le format dans lequel les données sont stockées, affichez sa feuille de propriétés et utilisez les propriétés **Type de stockage** et **Format de stockage**. Vous pouvez sélectionner des valeurs dans la liste ou saisir vos propres valeurs.
	Zones de fonctionnalité commune - Fournissent des espaces logiques pour regrouper tous les autres types d'éléments. Pour ajouter des objets dans une zone, créez-les directement sur celle-ci, ou faites-les glisser dessus. Les zones permettent l'imbrication.
	Accès en lecture, en écriture et en modification - Relient les agents aux stockages. Pour changer le type d'accès après la création, pointez sur l'accès, cliquez le bouton droit de la souris, puis sélectionnez la commande **Changer en...** appropriée, ou affichez sa feuille de propriétés et utilisez la propriété **Type d'accès**. L'accès en modification peut être affiché sous la forme d'une ligne simple ou de d'arcs doubles. Pour contrôler le type de symbole, pointez sur l'accès, cliquez le bouton droit de la souris, puis sélectionnez la commande **Afficher sous forme de ligne simple** ou **Afficher sous forme de double arc**, ou utilisez la propriété **Afficher sous forme de double arc**.

Outil	Description
	Canaux de communication demande/réponse, unidirectionnel et directionnel - Représentent des lignes téléphoniques, des connexions réseau, des tuyaux ou de simples appels de procédures, et connectent plusieurs agents, leur permettant de communiquer. Pour changer le type de canal après la création, pointez sur le canal, cliquez le bouton droit de la souris, puis sélectionnez la commande **Changer en...** appropriée, ou affichez sa feuille de propriétés et utilisez la propriété **Type de communication**. Les flèches indiquent la direction du flux d'information. Un "R" avec une petite flèche représente une paire de canaux demande-réponse, la pointe indiquant la direction de la demande du client au serveur. Pour inverser la direction, pointez sur le canal, cliquez le bouton droit de la souris, puis sélectionnez la commande **Changer de direction de canal**. Pour contrôler l'affichage des flèches, pointez sur le canal, cliquez le bouton droit de la souris, puis sélectionnez la commande **Afficher les flèches** ou **Masquer les flèches**, ou utilisez la propriété **Afficher les flèches**.
	Points de suspension - [dans la Boîte à outils Symboles prédéfinis] Peuvent être placés dans un symbole d'agent ou de stockage pour indiquer plusieurs instances.
	Ligne de limite, limite de protocole - [utilisez l'outil Ligne dans la Boîte à outils Symboles libres] Identifie une limite système. Pour changer le format d'une ligne de limite, pointez dessus, cliquez le bouton droit de la souris, puis sélectionnez la commande **Format**. Par exemple, pour obtenir une ligne en pointillés, sélectionnez le style approprié dans la liste sur l'onglet **Style de ligne**.

L'exemple suivant montre des agents clients qui communiquent avec un serveur de boutique en ligne, sur lequel les composants de ventes accèdent aux stockages des paniers et des données ERP :


Clients
Client 1
Client 2
Client 3
Fournisseur de services
Organismes de crédit
Expédition
Fournisseurs
S'enregistrer, Editer profil, Consulter commandes
Rechercher, Obtenir infos produit, Sélectionner articles
Passer commandes, Annuler commande
R
R
R
R
Serveur boutique en ligne
Ventes
Compte client
Présentation produit
Traitement commande
Paniers
Panier 1
Panier 2
Panier 3
Données ERP
Dispo produit et commandes
Dispo produit
Commandes
Données client
Informations produit

Chapitre 14 Eclipse Modeling Framework (EMF)

PowerAMC permet la modelisation dans le langage EMF, y compris l'ingénierie par va-et-vient.

EMF (Eclipse Modeling Framework) est un environnement de modélisation et un utilitaire de génération de code permettant de construire des outils et d'autres applications basées sur un modèle objet structuré. Un modèle EMF fournit un modèle simple de classes et de données pour une application, et est utilisé comme cadre de définition des métadonnées dans de nombreux outils Eclipse, notamment Sybase DI et Sybase WorkSpace.

Pour plus d'informations sur EMF, reportez-vous à la documentation et aux didacticiels EMF à l'adresse *http://www.eclipse.org/emf*.

Objets EMF

Les objets suivants sont disponibles dans un MOO ayant pour cible EMFF.

EPackages

PowerAMC modélise les EPackages EMF sous la forme de packages standard UML, mais avec des propriétés supplémentaires.

Les feuilles de propriétés d'EPackage contiennent tous les onglets standard d'une feuille de propriétés de package, avec en plus un onglet EMF, dont les propriétés sont répertoriées ci-dessous :

Propriété	Description
Préfixe d'espace de noms	Utilisé lorsque les références aux instances des classes dans ce package sont sérialisées.
URI d'espace de noms	Apparaît dans la balise xmlns pour identifier ce package dans un document XMI.
Nom de base du package	Contient le code généré pour le modèle.

Pour plus d'informations sur la création et l'utilisation des packages, voir *Packages (MOO)* à la page 53.

EClasses, EEnums et EDataTypes

PowerAMC modélise les EClasses EMF sous la forme de classes UML standard, et les EEnums et EDataTypes sous la forme de classes UML standard respectivement avec un stéréotype Enum et EDataType.

Pour plus d'informations sur la création et l'utilisation des classes, voir *Classes (MOO)* à la page 37.

Les feuilles de propriétés d'EClass, EEnum et EDataType contiennent tous les onglets des feuilles de propriétés de classes standard, avec en plus l'onglet EMF, dont les propriétés sont répertoriées ci-dessous :

Propriété	Description
Nom d'instance de la classe	Spécifie le nom de classe d'instance de type de données.

EAnnotations

PowerAMC modélise les EAnnotations EMF sous la forme de classes UML standard, avec un stéréotype AnnotationType.

Pour plus d'informations sur la création et l'utilisation des classes, voir *Classes (MOO)* à la page 37.

Les feuilles de propriétés d'EAnnotation contiennent tous les onglets des feuilles de propriétés de classes standard, avec en plus l'onglet EMF, dont les propriétés sont répertoriées ci-dessous :

Propriété	Description
Références	Spécifie la référence d'annotation EMF.
URI	Spécifie la source d'annotation EMF.

EAttributes et EEnumLiterals

PowerAMC modélise les EAttributes et les EEnumLiterals (EAttributes appartenant aux EEnums) EMF sous la forme d'attributs UML standard, mais avec des propriétés supplémentaires.

Pour plus d'informations sur la création et l'utilisation des attributs, voir *Attributs (MOO)* à la page 69.

Les feuilles de propriétés d'EAttribute et d'EEnumLiteral contiennent tous les onglets d'une feuille de propriétés d'attribut standard, avec en plus des onglets spécifiques à EMF dont les propriétés sont répertoriées dans les sections ci-dessous.

Propriété	Description
Unique	Spécifie que l'attribut ne peut pas avoir de doublons.
Avec annulation de modification (unsettable)	Génère une méthode unset pour annuler l'opération set.
Trié	Spécifie que l'attribut est ordonné.

EReferences

PowerAMC modélise les EReferences EMF sous la forme d'association UML standard, mais avec des propriétés supplémentaires.

Pour plus d'informations sur la création et l'utilisation des associations, voir *Associations (MOO)* à la page 90.

Les feuilles de propriétés d'EReference contiennent tous les onglets standard de feuille de propriétés d'association avec en plus l'onglet EMF, dont les propriétés sont répertoriées ci-dessous :

Propriété	Description
Unique	Spécifie que le rôle sélectionné ne peut pas avoir de doublon.
Résoudre les proxies	Résoud les proxies si le noeud sélectionné se trouve dans un autre fichier.
Non paramétrable	Spécifie que le rôle sélectionné ne peut pas être défini.

EOperations et EParameters

PowerAMC modélise les EOperations et EParameters EMF sous la forme d'opérations et de paramètres d'opération UML standard.

Pour plus d'informations sur la création et l'utilisation des opérations, voir *Opérations (MOO)* à la page 82.

Génération de fichiers EMF

PowerAMC permet de générer des fichiers EMF .ecore et .genmodel.

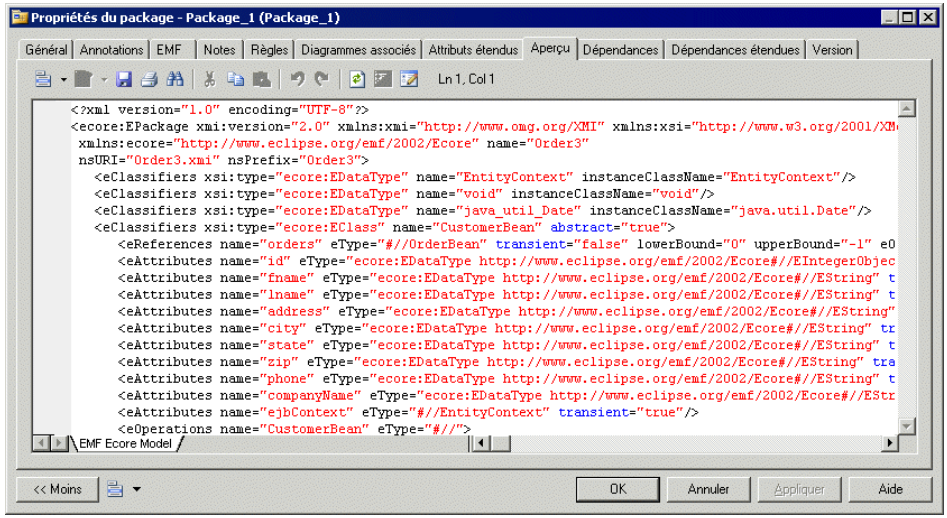
1. Sélectionnez **Langage > Générer du code EMF** pour afficher la boîte de dialogue de génération.
2. Spécifiez le répertoire dans lequel vous souhaitez générer les fichiers et indiquez si vous souhaitez ou non procéder à une vérification de modèle.

3. [facultatif] Sur l'onglet Sélection, sélectionnez les objets à partir desquels vous souhaitez générer. Par défaut, tous les objets sont générés, et PowerAMC se souvent des changements que vous effectuez dans la sélection pour les générations ultérieures.

   Remarque : Bien que vous puissiez créer tous les diagrammes UML standard et leurs objets associés, vous ne pouvez générer que les packages, les classes et les interfaces.

4. [optionnel] Cliquez sur l'onglet Options et spécifiez la version d'EMF pour laquelle vous souhaitez générer.
5. Cliquez sur OK pour lancer la génération.

   Une boîte de progression s'affiche. La fenêtre Liste de résultats affiche les fichiers générés, que vous pouvez éditer. Les résultats sont également affichés dans la fenêtres Résultats, située dans la partie inférieure de la fenêtre principale.



Reverse engineering de fichiers EMF

Pour pouvoir procéder au reverse engineering de fichiers EMF dans un MOO, vous devez utiliser le plugin PowerAMC pour Eclipse, et il faut avoir également installé le plugin EMF.

1. Sélectionnez **Langage > Reverse engineering EMF** pour afficher la boîte de dialogue de reverse engineering.
2. Cliquez sur l'un des boutons suivants pour aller sélectionner les fichiers .ecore, .emof ou .genmodel sur lesquels vous souhaitez faire porter le reverse engineering :
   - Browse File System (Parcourir le système de fichiers)
   - Browse Workspace (Parcourir l'espace de travail)

3. Sélectionnez les fichiers sur lesquels vous souhaitez faire porter le reverse engineering, puis cliquez sur Open (ou OK) afin de les ajouter à la liste Model URIs.
4. Cliquez sur Next pour passer à la page Package Selection, puis sélectionnez les packages sur lesquels vous souhaitez faire porter le reverse engineering.
5. Cliquez sur Terminer pour lancer le reverse engineering.

   Si le modèle dans lequel vous effectuez le reverse engineering contient déjà des données, la boîte de dialogue Fusion de modèles s'affiche pour vous permettre de spécifier si les objets existants doivent être écrasés.

   Pour plus d'informations sur la fusion de modèles, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Comparaison et fusion de modèles*.

   Les packages sont ajoutés dans votre modèle.

Chapitre 15 PowerBuilder

PowerAMC permet la prise en charge pour la modélisation de PowerBuilder, y compris l'ingénierie par va-et-vient.

PowerBuilder est un outil de développement orienté objet. La plupart de ses composants sont conçus comme des objets dotés de propriétés, de méthodes et d'événements qui peuvent être mis en correspondance avec les classes UML définies dans un MOO. PowerAMC prend en charge les objets PowerBuilder stockés dans un fichier .PBL. Les fichiers PBD (Dynamic PowerBuilder Libraire) ne sont pas pris en charge

Remarque : Si vous avez plusieurs versions de PowerBuilder installées sur votre machine, PowerAMC utilise par défaut la plus récente. Si vous souhaitez travailler sur une version antérieure de PowerBuilder, cliquez sur le bouton **Changer** à droite de la zone **Version de PowerBuilder** dans la boîte de dialogue de génération ou de reverse engineering.

Objets PowerBuilder

Cette section décrit les correspondances entre les objets PowerBuilder et objets de MOO PowerAMC.

Applications

Vous modélisez une application PowerBuilder à l'aide d'une classe en utilisant le stéréotype *<<application>>*. Les propriétés d'application sont définies comme suit :

PowerBuilder	PowerAMC
Instance variable	Attribut
Shared variable	Attribut statique
Global variable	Attribut avec le stéréotype <<global>>
Property	Attribut avec le stéréotype <<property>>
External function	Opération avec le stéréotype <<external>>
Function	Opération
Event	Opération avec le stéréotype <<event>> ou opération avec un nom d'événement non-null
Structure in object	Classe interne avec un stéréotype <<structure>>

Structures

Vous modélisez une structure PowerBuilder à l'aide d'une classe ayant le stéréotype *<<structure>>*. Les membres de la structure sont conçus à l'aide d'attributs de classe.

Functions

Vous modélisez une function PowerBuilder à l'aide d'une classe ayant le stéréotype *<<function>>*. Cette classe doit également contenir une opération. Les structures contenues dans une fonction sont modélisées à l'aide de classes internes *<<structure>>* liées à la classe.

User Objects

Vous modélisez un user object PowerBuilder à l'aide d'une classe ayant le stéréotype *<<userObject>>*. Les propriétés d'un user object sont définies comme suit :

PowerBuilder	PowerAMC
Control	Classe interne avec un stéréotype <<control>>
Instance variable	Attribut
Shared variable	Attribut statique
Property	Attribut avec le stéréotype <<property>>
Function	Opération
Event	Opération avec le stéréotype <<event>> ou opération avec un nom d'événement non-null
Structure in object	Classe interne avec un stéréotype <<structure>>

Proxies

Vous modélisez un proxy PowerBuilder à l'aide d'une classe ayant le stéréotype *<<proxyObject>>*. Les instance variables du proxy sont modélisées à l'aide d'attributs de classes, et les functions de proxy sont modélisées à l'aide d'opérations.

Windows

Vous modélisez une window PowerBuilder à l'aide d'une classe ayant le stéréotype *<<window>>*. Les propriétés de window sont définies comme suit :

PowerBuilder	PowerAMC
Control	Classe interne avec un stéréotype <<control>>
Instance variable	Attribut
Shared variable	Attribut statique
Property	Attribut avec le stéréotype <<property>>

PowerBuilder	PowerAMC
Function	Opération
Event	Opération avec le stéréotype <<event>> ou opération avec un nom d'événement non-null
Structure in object	Classe interne avec un stéréotype <<structure>>

Operations

Si l'attribut étendu d'opéation *GenerateHeader* est défini à True, l'en-tête de l'opération sera généré. Cet attribut est défini à True pour toute nouvelle opération. Vous pouvez forcer la génération d'en-tête pour toutes les opérations dans un modèle en définissant l'attribut étendu *ForceOperationHeader* à True. Les en-têtes d'opération sont générés de la façon suivante :

```
//<FuncType>:  <Operation signature>
//Description: <Operation comment line1>
//      <Operation comment line2>
//Access:  <visibility>
//Arguments: <parameter1 name> - <parameter1 comment line1>
//            <parameter1 comment line2>
//      <parameter2 name> - <parameter2 comment>
//Returns:   <Return comment>
//      <Return comment2>
```

Elément d'en-tête	Objet ou propriété PowerAMC
FuncType	Fonction, sous-routine, ou événement
Description	Commentaire saisi dans la feuille de propriétés d'opération
Access	Propriété Visibilité dans la feuille de propriétés d'opération
Arguments	Nom et commentaire de paramètre(s)
Returns	Valeur d'attribut étendu ReturnComment dans la feuille de propriétés d'opération
User-defined comment	Valeur d'attribut étendu UserDefinedComment dans la feuille de propriétés d'opération

Events

Pour pouvoir générer un :

- *Standard event handler* - vous devez créer une opération et sélectionner une valeur d'événement dans la liste déroulante Evénement de langage
- *User-defined event handler* - vous devez créer une opération et sélectionner le stéréotype <<event>>. La liste déroulante Evénement de langage doit rester vide

- *Custom event handler* - vous devez créer une opération et définir une valeur pour l'attribut étendu EventID. Lorsque cet attribut étendu a une valeur, l'opération est générée sous forme de d'événement personnalisé, bien qu'elle ait un nom défini dans la liste déroulante Evénement de langage ou le stéréotype <<event>>.

Autres objets

Ces PowerBuilder sont récupérés par reverse engineering sous la forme de classes ayant le stéréotype PowerBuilder correspondant. Leurs propriétés ne sont pas mises en correspondance avec les propriétés de classe PowerAMC, et leur symbole et une grande icône PowerBuilder.

PowerBuil-der	PowerAMC
Query	Classe <<query>>
Data window	Classe <<dataWindow>>
Menu	Classe <<menu>>
Project	Classe <<project>>
Pipe line	Classe <<pipeLine>>
Binary	Classe <<binary>>

Pour plus d'informations sur le verse engineering PowerBuilder, voir *Reverse engineering de code PowerBuilder* à la page 441.

Génération d'objets PowerBuilder

Vous pouvez générer des objets PowerBuilder dans une application PowerBuilder existante ou sous la forme de fichiers source. Chaque classe ayant un stéréotype est générée sous la forme de l'objet PowerBuilder approprié. Les classes dépourvues de stéréotype sont générées sous la forme de user objects. Les objets qui ne sont pas pleinement pris en charge par PowerAMC voient leurs propriétés retirées et seul leur en-tête est généré.

1. Sélectionnez **Langage > Générer pour PowerBuilder** pour afficher la boîte de dialogue Génération PowerBuilder.
2. [facultatif] Cliquez sur l'onglet **Sélection** et spécifiez les objets à partir desquels vous souhaitez générer. Par défaut, tous les objets sont générés.
3. [facultatif] Cliquez sur l'onglet **Options** et sélectionnez les options de génération appropriées :

Option	Description
Vérifier le modèle	Lance une vérification de modèle avant de procéder à la génération (voir *Chapitre 9, Vérification d' un MOO* à la page 303).
A l'aide de bibliothèques	Ce mode n'est disponible que si vous avez installé PowerBuilder sur votre machine. Spécifiez une version de PowerBuilder et sélectionnez une cible ou une application dans la liste Cible/Application. Les objets sont générés comme suit : • Un package avec un chemin d'accès de bibliothèque (défini dans un attribut étendu lors du reverse engineering) est généré dans la bibliothèque correspondante, sélectionnée dans la liste de cible/application • Un package à la racine du modèle sans chemin d'accès de bibliothèque est généré dans une nouvelle bibliothèque, au même niveau que la bibliothèque cible/application • Un package enfant sans chemin d'accès de bibliothèque est généré dans le package parent • Un objet à la racine du modèle est généré dans la bibliothèque cible/application
A l'aide de fichiers source	Spécifiez un répertoire dans lequel générer les fichiers. Les objets sont générés comme suit : • Les classes définies au niveau du modèle sont générées sous forme de fichiers source dans le répertoire spécifié. • Les classes définies dans des packages sont générées sous forme de fichiers source dans des sous-répertoires. Vous allez donc devoir importer les objets générés dans PowerBuilder.

4. Cliquez sur **OK** pour lancer la génération.

   Les fichiers sont générés dans l'application ou dans le répertoire spécifié.

Reverse engineering de code PowerBuilder

Cette section explique comment les objets PowerBuilder sont récupérés lors du reverse engineering et comment définir les options de reverse engineering pour PowerBuilder.

Objets récupérés par le reverse engineering

Vous pouvez procéder au reverse engineering vers un MOO des objets contenus dans un fichier .PBL ou exportés par PowerBuilder dans des fichiers. Certains des objets ainsi

récupérés prennent en charge une mise en correspondance avec une classe de MOO, d'autres non.

Bibliothèques

Chaque bibliothèque PowerBuilder est récupérée par reverse engineering sous forme de package dans le MOO résultant. Le chemin de la bibliothèque est stocké dans un attribut étendu attaché au package.

Les objets récupérés par reverse engineering depuis une bibliothèque sont créés dans le package correspondant dans PowerAMC.

Mise en correspondance complète

Lors du reverse engineering, de nouvelles classes ayant un stéréotype correspondant à leur objet PowerBuilder d'origine sont créées. Le symbole de ces classes comporte une icône dans l'angle supérieur gauche :

Pour plus d'informations sur les objets PowerBuilder entièrement pris en charge, voir *Objets PowerBuilder* à la page 437.

Mise en correspondance minimale

Les objets PowerBuilder qui ne sont pas entièrement pris en charge dans PowerAMC sont récupérés sous forme de classes ayant le type stéréotype PowerBuilder correspondant. En revanche, leurs propriétés ne sont pas mises en correspondance avec les propriétés de classe PowerAMC et leur symbole est une grosse icône PowerBuilder.

Le code source de ces objets est extrait sans analyse et stocké dans l'en-tête de la classe, comme indiqué dans l'onglet Script\sous-onglet Début de la classe ; il sera utilisé de la même façon lors de la génération.

Pour plus d'informations sur les objets PowerBuilder partiellement pris en charge, voir *Objets PowerBuilder* à la page 437.

En-tête d'opération récupéré

Le reverse engineering traite le premier bloc de commentaire de la fonction entre deux lignes de barres obliques.

```
////////////////////////////////////////////////////////////////////
//////
// <FuncType>:  <Operation signature>
// Description: <Operation comment line1>
```

```
//               <Operation comment line2>
// Access:       <visibility>
// Arguments:    <parameter1 name> - <parameter1 comment line1>
//                                   <parameter1 comment line2>
//               <parameter2 name> - <parameter2 comment>
// Returns:      <Return comment>
//               <Return comment2>
////////////////////////////////////////////////////////////////////
//////
```

Si tous les mots clés générés sont trouvés, le bloc sera supprimé et les attributs appropriés seront définis :

Attribut de mot clé	Attribut d'opération correspondant
FuncType, Subroutine, Event	Nom
Description	Commentaire d'opération
Access	Propriété Visibilité
Arguments	Nom et commentaires de paramètre(s)
Returns	Valeur pour l'attribut étendu ReturnComment
User-defined comment	Valeur pour l'attribut étendu UserDefinedComment
GenerateHeader	Défini à True
Other function comments	Conservé dans le corps de l'opération

Dans le cas contraire, les commentaires de fonction sont conservés dans le corps de l'opération et l'attribut étendu GenerateHeader est défini à false.

Redéfinition d'attributs

Lorsqu'une classe hérite d'une autre classe, les attributs hérités non privés peuvent être définis comme des propriétés de la classe enfant, ce qui permet à l'utilisateur de définir des valeurs initiales dans la classe enfant.

1. Affichez la feuille de propriétés d'une classe enfant, puis cliquez sur l'onglet **Attributs**.
2. Cliquez sur l'outil **Redéfinir les attributs hérités** pour afficher la liste des attributs disponibles dans la classe parent.
3. Sélectionnez un ou plusieurs attributs dans la liste, puis cliquez sur **OK**.

   Les attributs s'affichent dans la liste des attributs de la classe enfant. Vous pouvez modifier leur valeur initiale dans la colonne correspondante.

Processus de reverse engineering PowerBuilder

Lorsque vous procédez au reverse engineering d'objets à partir de PowerBuilder, vous pouvez choisir de faire porter cette opération sur des bibliothèques, sur des fichiers ou sur des répertoires.

Reverse engineering de bibliothèques
Ce mode permet de sélectionner une cible/application PowerBuilder dans la liste déroulante Cible/Application. Lorsqu'une cible ou une application est sélectionnée, les bibliothèques utilisées par la cible ou l'application sont automatiquement affichées dans la liste. Par défaut, tous les objets de toutes les bibliothèques sont sélectionnés. Vous pouvez désélectionner des objets et des bibliothèques avant de lancer le reverse engineering.

Si PowerBuilder n'est pas installé sur votre machine, la liste Cible/Application reste vide.

Reverse engineering de fichiers source
Ce mode permet de sélectionner des fichiers source d'objet PowerBuilder sur lesquels faire porter le reverse engineering. Le suffixe du fichier source détermine le type de l'objet ayant subi un reverse engineering.

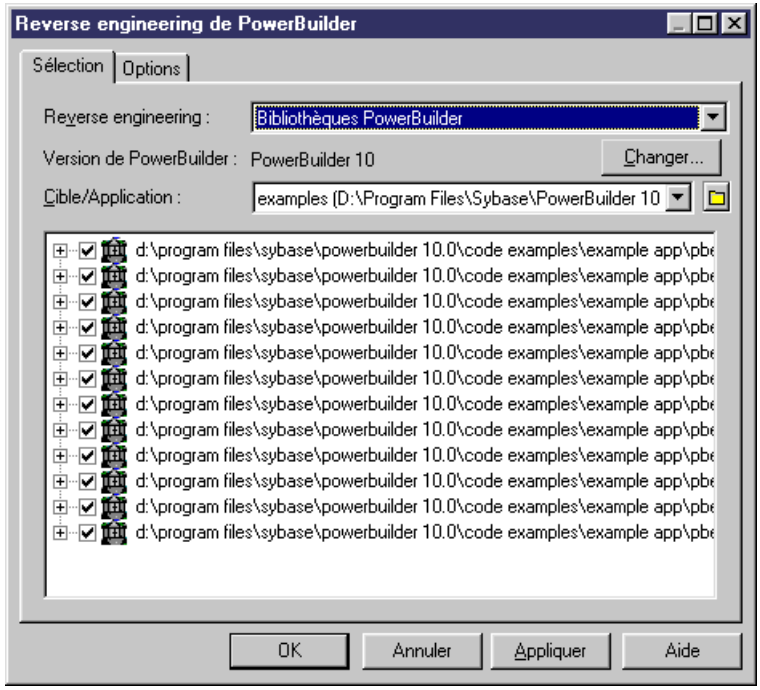
Vous pouvez pointer sur les fichiers sur lesquels vous souhaitez réaliser un reverse engineering, cliquer le bouton droit de la souris et sélectionner la commande Editer pour visualiser le contenu de ces fichiers. Pour pouvoir utiliser cette commande, vous devez avoir préalablement associé le suffixe de nom de fichier avec un éditeur dans la boîte de dialogue Options générale\Editeur.

Reverse engineering de répertoires
Ce mode permet de sélectionner un répertoire PowerBuilder sur lequel faire porter le reverse engineering. Lorsque vous sélectionnez un répertoire, vous ne pouvez pas sélectionner de cible ou d'application individuelle. Utilisez le bouton Changer pour sélectionner un répertoire.

Reverse engineering d'objets PowerBuilder

Vous pouvez procéder au reverse engineering d'objets PowerBuilder en sélectionnant **Langage** > **Reverse engineering PowerBuilder**.

1. Sélectionnez **Langage > Reverse engineering de PowerBuilder** pour afficher la boîte de dialogue Reverse engineering de PowerBuilder.
2. Sélectionnez un fichier, une bibliothèque ou un répertoire dans la zone Reverse engineering.
3. Lorsque disponible, sélectionnez une cible ou une application dans la liste.



4. Cliquez sur l'onglet Options et définissez les options appropriées.

Option	Description
Ignorer le corps des opérations	Effectue le reverse engineering d'objets PowerBuilder sans y inclure le corps du code
Ignorer les commentaires	Effectue le reverse engineering d'objets PowerBuilder sans y inclure les commentaires
Créer des symboles	Crée un symbole dans le diagramme pour chaque objet. Dans le cas contraire, les objets récupérés ne sont visibles que dans l'Explorateur d'objets.
Créer des symboles de classes internes	Crée un symbole dans le diagramme pour chaque classe interne.
Marquer les classificateurs comme n'étant pas à générer	Les classificateurs récupérés par reverse engineering (classes et interfaces) ne seront pas générés à partir du modèle. Pour pouvoir les générer, vous devez cocher la case Générer dans leur feuille de propriétés respective.
Créer des associations	Crée des associations entre les classes et /ou les interfaces.

Option	Description
Bibliothèques	Spécifie une liste de modèles de bibliothèque à utiliser comme référence lors du reverse engineering. Le modèle de destination du reverse engineering peut contenir des raccourcis vers des objets définis dans une bibliothèque. Si vous spécifiez la bibliothèque ici, le lien entre le raccourci et son objet cible (contenu dans la bibliothèque) sera préservé et la bibliothèque sera ajoutée à la liste des modèles cibles dans le modèle de destination du reverse engineering Vous pouvez faire glisser les bibliothèques dans la liste afin de spécifier une hiérarchie de bibliothèques. PowerAMC tentera de résoudre les raccourcis trouvés dans le modèle de destination du reverse engineering en utilisant tour à tour chacune des bibliothèques spécifiées. Ainsi, si la bibliothèque v1.1 apparaît dans la liste avant la bibliothèque v1.0, PowerAMC tentera d'abord de résoudre les raccourcis vers la bibliothèque v1.1 et analysera uniquement la bibliothèque v1.0 s'il reste des raccourcis non résolus. Vous devez utiliser la liste des modèles cibles pour gérer les bibliothèques liées au modèle de destination du reverse engineering, par exemple, vous pouvez changer la version de bibliothèque (voir *Guide des fonctionnalités générales > Liaison et synchronisation de modèles > Raccourcis et répliques > Utilisation des modèles cible*.

5. Cliquez sur **OK**.

   Une boîte de progression s'affiche. Si le modèle vers lequel vous effectuez le reverse engineering contient déjà des données, la boîte de dialogue Fusion de modèles (voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Comparaison et fusion de modèles*) s'affiche.

   Les classes sont automatiquement ajoutées dans votre modèle et sont visibles dans le diagramme et l'Explorateur d'objets. Elle figurent également dans l'onglet Reverse de la fenêtre Résultats, située en bas de la fenêtre principale de PowerAMC.

   Remarque : Certains objets standards tels que les fenêtres ou les structures héritent des classes parent définies dans les bibliothèques système. Si ces bibliothèques ne sont pas chargées dans l'espace de travail, PowerAMC ne crée plus de classe non résolue pour représenter le parent de l'objet standard dans le modèle. Le lien entre l'objet standard et le parent sera recréé à l'issue de la génération grâce au stéréotype de l'objet standard.

Chargement d'un modèle de bibliothèque PowerBuilder dans l'espace de travail

Lorsque vous procédez au reverse engineering de fichiers PowerBuilder, vous pouvez également charger en même temps l'un des modèles PowerBuilder contenant les bibliothèques

de classes pour une version particulière de PowerBuilder. Le programme d'installation copie ces modèles dans le répertoire Bibliothèque de PowerAMC.

Vous pouvez choisir d'effectuer le reverse engineering d'un modèle de bibliothèque PowerBuilder à partir de l'onglet Options de la boîte de dialogue Reverse engineering de PowerBuilder.

Vous pouvez ouvrir un modèle de bibliothèque PowerBuilder dans l'espace de travail à partir du répertoire Bibliothèque de PowerAMC.

1. Sélectionnez **Fichier > Ouvrir** pour afficher une boîte de dialogue d'ouverture de fichiers.
2. Sélectionnez le répertoire de bibliothèque de PowerAMC.

   Les fichiers de bibliothèque disponibles sont répertoriés. Chaque fichier PB correspond à une version particulière de PowerBuilder.
3. Sélectionnez le fichier correspondant à la version souhaitée.

   Ce fichier contient les fichiers de bibliothèques de classes correspondant à la version de PowerBuilder choisie.
4. Cliquez sur Ouvrir.

   Le MOO est ouvert dans l'espace de travail.

Chapitre 16 VB .NET

PowerAMC prend en charge la modélisation de programmes VB .NET, y compris l'ingénierie par va-et-vient.

Héritage et implémentation

Vous pouvez concevoir un *héritage (inheritance)* VB .NET en utilisant un lien de *généralisation* entre des classes.

Vous pouvez concevoir une *implementation* VB .NET à l'aide d'un lien de *réalisation* entre une classe et une interface.

Espace de noms

Vous pouvez définir un *espace de noms* VB .NET en utilisant un package.

PowerAMC modélise les espaces de noms sous la forme de packages standard avec la propriété **Utiliser l'espace de noms du parent** désélectionnée.

Dans l'exemple suivant, la classe Architect est déclarée dans le package Design qui est un sous-package de Factory. La déclaration d'espace de noms se présente comme suit :

```
Namespace Factory.Design
 Public Class Architect
...
...End Class
End Namespace ' Factory.Design
```

Les classificateurs définis directement au niveau du modèle appartiennent à l'espace de noms global de VB.NET.

Projet

Vous pouvez faire porter le reverse engineering sur des projets VB .NET. Pour ce faire, sélectionnez Projets VB .NET dans la liste Reverse engineering de la boîte de dialogue Reverse engineering de VB .NET.

Vous ne devez pas procéder au reverse engineering de plusieurs projets distincts dans un même modèle.

Les propriétés d'assembly sont traitées comme suit par le reverse engineering :

Propriétés d'assembly VB .NET	Equivalent PowerAMC
Title	Nom du modèle
Description	Description du modèle
Company	Attribut étendu AssemblyCompany
Copyright	Attribut étendu AssemblyCopyright
Product	Attribut étendu AssemblyProduct
Trademark	Attribut étendu AssemblyTrademark
Version	Attribut étendu AssemblyVersion
AssemblyInformationalVersion AssemblyFileVersion AssemblyDefaultAlias	Stocké dans l'attribut étendu CustomAttributes

Les propriétés de projet sont récupérées sous forme d'attributs étendus qu'elles aient une valeur ou non. Par exemple, la disposition de page HTML par défaut est enregistrée dans l'attribut étendu DefaultHTMLPageLayout.

Vous pouvez utiliser le bouton Points de suspension dans la colonne Valeur pour modifier la valeur d'attribut étendu, bien qu'il soit conseillé d'être prudent lorsque vous effectuez de telles modifications qui peuvent compromettre la génération de modèle.

Accessibilité

Pour définir l'*accessibilité* d'une classe, d'une interface, d'un attribut ou d'une méthode, vous devez utiliser la propriété *Visibilité* dans PowerAMC.

Les attributs d'accessibilité suivants sont pris en charge dans PowerAMC :

Accessibilité VB .NET	Visibilité PowerAMC
Public (no restriction)	*Public*
Protected (accessible par les classes dérivées)	*Protected*
Friend (accessible au sein d'un programme qui contient la déclaration de la classe)	*Friend*
Protected Friend (accessible par les classes dérivées et au sein d'un programme qui contient la déclaration de la classe)	*Protected Friend*
Private (accessible uniquement par la classe)	*Private*

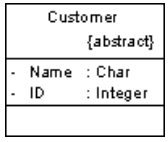
Dans l'exemple suivant, la visibilité de la classe Customer est Friend :

```
Friend Class Customer
```

Classes, interfaces, structures et énumérations

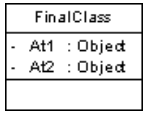
Vous modélisez une *classe* VB .NET en utilisant une classe dans PowerAMC. Les *structures* sont des classes ayant le stéréotype <<*structure*>>, et les *énumérations* des classes ayant le stéréotype <<*enumeration*>>.

- peuvent contenir des événements, des variables, des constantes, des méthodes et des propriétés. Les classes particulières suivantes sont également prises en charge par PowerAMC :
  - Une classe *MustInherit* équivaut à une classe abstraite. Pour concevoir ce type de classe, vous devez créer une classe et cocher la case *Abstrait* dans l'onglet Général de la feuille de propriétés de classe.




```
Public MustInherit Class Cus-
tomer
 Private Name As Char
 Private ID As Integer
End Class
```

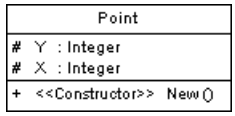
  - Une classe *NotInheritable* équivaut à une classe finale. Pour concevoir ce type de classe, vous devez créer une classe et cocher la case *Final* dans l'onglet Général de la feuille de propriétés de cette classe.




```
Public NotInheritable Class
FinalClass
 Private At1 As Object
 Private At2 As Object
End Class
```

Remarque : Vous modélisez un *type imbriqué* VB .NET à l'aide d'une classe interne ou d'une interface.

- Les *interfaces* VB .NET sont modélisées sous la forme d'interfaces standard. Elles peuvent contenir des événements, des propriétés et des méthodes ; elles ne prennent pas en charge les variables, les constantes ou les constructeurs.
- Les structures peuvent mettre en oeuvre des interfaces, mais ne prennent pas en charge l'héritage ; elles peuvent contenir des événements, des variables, des constantes, des méthodes, des constructeurs et des propriétés. La structure suivante contient deux attributs et une opération de constructeur :




```
...
Public Class Point
 Protected Y As Integer
 Protected X As Integer
 Public Sub New()

 End Sub
End Class
...
```

- Les attributs de classe Enumeration sont utilisés comme valeurs d'énumération. Les éléments suivants doivent être définis :
  - *Type de données* - en utilisant l'attribut étendu *EnumDataType* de l'énumération (par exemple Byte, Short ou Long)
  - *Expression initiale* - en utilisant la zone *Valeur initiale* d'un attribut d'énumération

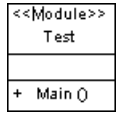
Par exemple :


```
Public Enum Day
 Monday
 Tuesday
 Wednesday
 Thursday
 Friday
 Saturday
 Sunday
 FirstDay = Monday
 LastDay = Sunday
End Enum
```

Module

Vous modélisez un *module* VB .NET en utilisant une classe ayant comme stéréotype *<<module>>* et des attributs, fonctions, subs et événements.

Dans l'exemple suivant, vous définissez un module Test en utilisant une classe ayant le stéréotype <<module>>. Test contient une *fonction*. Pour concevoir cette fonction, vous devez créer une opération appelée Main et supprimer le contenu de la propriété Type de résultat. Vous pouvez ensuite définir le corps de la fonction dans l'onglet Mise en oeuvre de l'opération.

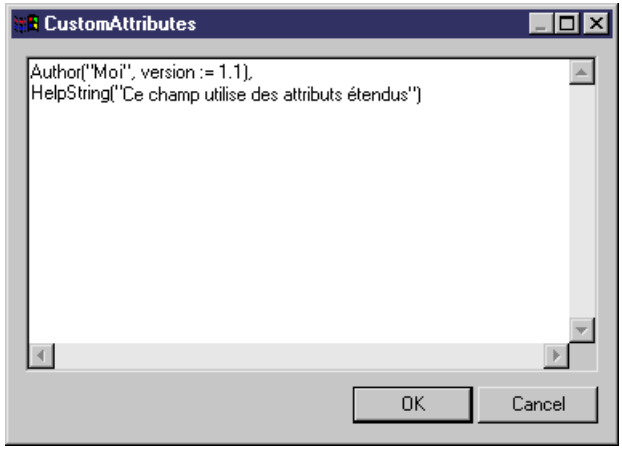



```
...
Public Module Test
  Public Sub Main()
   Dim val1 As Integer = 0
     Dim val1 As Integer = val1
     val2 = 123
```

```
    Dim ref1 As New Class1 ()
    Dim ref1 As Class1 = ref1
    ref2.Value = 123
    Console.WriteLine ("Value: "& val1", "& val2)
    Console.WriteLine ("Refs: "&ref1.Value &", "& ref2.Value)

 End Sub
End Module
...
```

Attributs personnalisés

Pour définir les *attributs personnalisés* d'une classe, interface, variable ou méthode, vous devez utiliser l'attribut étendu *Attributs personnalisés* dans PowerAMC. Vous pouvez utiliser la boîte de saisie de CustomAttributes pour spécifier tous les attributs personnalisés que vous souhaitez ajouter en utilisant la syntaxe VB.NET appropriée.



Attributs étendus pour des types de résultats

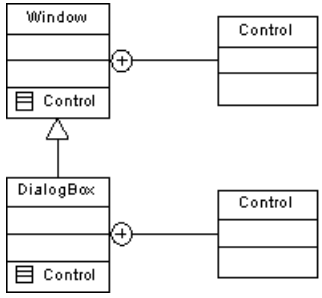
Vous pouvez utiliser l'attribut étendu *Attributs personnalisés de type de résultat* pour définir des attributs étendus personnalisés pour le type de résultats d'un attribut de propriété ou d'une méthode.

Shadows

Shadows indique qu'un élément hérité cache un élément parent de même nom. Pour concevoir une classe, interface, variable ou méthode *Shadows*, vous devez définir l'attribut étendu *Shadows* à true pour la classe ou l'interface.

Dans l'exemple suivant, la classe DialogBox hérite de la classe Window. La classe Window contient un classificateur interne nommé Control, il en va de même pour la classe DialogBox.

Vous ne souhaitez pas que la classe DialogBox hérite du contrôle défini dans Window, pour ce faire vous devez définir l'attribut étendu Shadows à True, dans la classe Control interne à DialogBox :




```
...
Public Class DialogBox
 Inherits Window
 Public Shadows Class Control
 End Class
End Class
...
```

Variables

Vous modélisez une variable VB .NET en utilisant un attribut dans PowerAMC.

Le tableau suivant récapitule les différents types de variables et d'attributs VB .NET pris en charge dans PowerAMC :

Variable VB .NET	Equivalent PowerAMC
Variable *ReadOnly*	Attribut avec la valeur *Lecture seule* dans la zone Modifiable
Variable *Const*	Attribut avec la valeur *Figé* dans la zone Modifiable
Variable *Shared*	Attribut avec la propriété *Statique* sélectionnée
Variable *Shadowing*	Attribut étendu *Shadowing* défini à *Shadows* ou *Overloads*
Variable *WithEvents*	Attribut avec le stéréotype *withEvents*
Variable *Overrides*	Attribut étendu *Overrides* défini à True
Variable *New*	Attribut étendu *AsNew* défini à True

- *Data Type* : Vous définissez le type de données d'une variable en utilisant la propriété Type de données de l'attribut
- *Initial Value* : Vous définissez la valeur initiale d'une variable en utilisant la propriété Valeur initiale de l'attribut
- *Shadowing* : pour définir un mode d'occultation (shadowing) par nom, définissez la valeur Shadows pour l'attribut étendu Shadowing. Pour définir un mode d'occultation par nom et

par signature, spécifiez la valeur Overloads pour l'attribut étendu Shadowing. Voir *Méthode* à la page 456 pour plus de détails sur le shadowing

Propriété

Pour concevoir une *propriété* VB .NET, vous devez créer un attribut ayant le stéréotype <<*Property*>>, un autre attribut avec le stéréotype <<*PropertyImplementation*>> est alors automatiquement créé, il s'affiche avec un trait de soulignement dans la liste des attributs. Les opérations getter et setter correspondantes sont également automatiquement créées.

Vous pouvez supprimer l'attribut de mise en oeuvre.

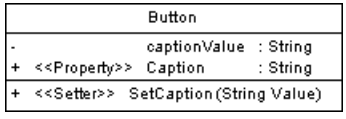
Si vous supprimez l'opération getter, le mot-clé *ReadOnly* est automatiquement généré. Si vous supprimez l'opération setter, le mot clé *WriteOnly* est automatiquement généré. Si vous supprimez à la fois les opérations getter et setter, l'attribut n'a plus le stéréotype <<*Property*>>.

Lorsque vous définissez un attribut <<*Property*>>, le caractère modifiable ou non et les opérations getter/setter sont intimement liés comme indiqué dans le tableau suivant

Opérations	Caractère modifiable de la propriété
Si vous conservez les opérations getter et setter	La propriété est Modifiable
Si vous supprimez l'opération setter d'une propriété modifiable	La propriété devient Lecture seule
Si vous supprimez l'opération getter d'une propriété modifiable	La propriété devient Ecriture seule

D'un autre côté, si vous modifiez le caractère modifiable ou non de la propriété, les opérations refléteront ce changement, par exemple si vous transformez une propriété modifiable en propriété en lecture seule, l'opération setter est automatiquement supprimée.

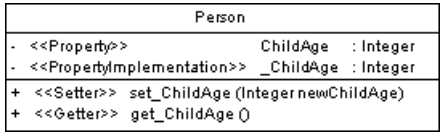
Dans l'exemple suivant, la classe Button contient une propriété Caption. L'opération Getter a été supprimée car elle provoque l'apparition du mot clé WriteOnly dans la ligne de déclaration de la propriété :




```
Public Class Button
 Private captionValue As String
 Public WriteOnly Property Caption() As String
  Set (ByVal Value As String)
   captionValue = value
   Repaint()

  End Set
```

```
 End Property
End Class
```

- *Doit redéfinir* : Définissez l'attribut étendu Doit redéfinir de la propriété à True pour exprimer le fait que la propriété définie dans une classe de base doit être redéfinie dans une classe dérivée avant de pouvoir être utilisée
- *Redéfinissable* : Définissez l'attribut étendu Redéfinissable à True pour exprimer le fait que la propriété peut être redéfinie dans une classe dérivée
- *Redéfinit* : Définissez l'attribut étendu Redéfinit de la propriété a True pour exprimer le fait qu'une propriété redéfinit un membre hérité d'une classe de base
- *Paramètres* : Saisissez une valeur dans la zone *Paramètres de propriété* afin de spécifier quelle valeur de la propriété d'attribut doit être utilisée comme paramètre. Dans l'exemple suivant, la classe Person contient l'attribut de propriété ChildAge. Le paramètre utilisé pour trier la propriété est ChildName :




```
Public Class Person
 Private _ChildAge As Integer

 Private Property ChildAge(ChildName as String) As Integer
  Get
   return _ChildAge
  End Get
  Set (ByVal Value ChildAge As Integer)
   If (_ChildAge <> newChildAge)
    _ChildAge = newChildAge
   End If
  End Set
 End Property
End Class
```

Méthode

Vous créez une *méthode* VB .NET en utilisant une opération. Les méthodes peuvent être des fonctions, ou des subs.

Pour modéliser une *fonction*, utilisez une opération avec une valeur de résultat.

Pour modéliser une *sub*, utilisez une opération avec une valeur de résultat vide.

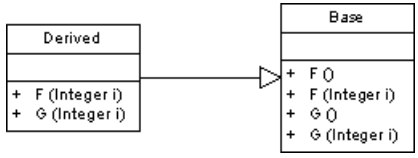
Le tableau suivant récapitule les différentes méthodes prises en charge par PowerAMC :

Méthode VB .NET	Equivalent dans PowerAMC
Shadowing ou *Overloads*	Sélectionnez *Shadows* ou *Overloads* dans la liste *Shadowing* sur l'onglet VB.NET de la feuille de propriétés de l'opération

Méthode VB .NET	Equivalent dans PowerAMC
Shared	Cochez la case *Statique* sur l'onglet Général de la feuille de propriétés de l'opération
NotOverridable	Cochez la case *Final* sur l'onglet Général de la feuille de propriétés de l'opération
Overridable	Cochez la case *Redéfinissable* sur l'onglet VB.NET de la feuille de propriétés de l'opération
MustOverride	Cochez la case *Abstrait* sur l'onglet Général de la feuille de propriétés de l'opération
Overrides	Cochez la case *Redéfinit* sur l'onglet VB.NET de la feuille de propriétés de l'opération

Shadowing

Vous pouvez modéliser l'occultation (shadowing) en sélectionnant *Shadows* dans à la liste *Shadowing* sur l'onglet VB.NET de la feuille de propriétés de l'opération. Pour définir un mode d'occultation par nom et par signature, sélectionnez *Overloads* pour l'attribut étendu Shadowing. Dans l'exemple suivant, la classe Derived hérite de la classe Based.



L'opération F de la classe Derived redéfinit l'opération F dans la classe Base ; et l'opération G dans la classe Derived occulte l'opération G dans la classe Base :

```
Public Class Derived
 Inherits Base
  Public Overloads Sub F(ByVal i As Integer)
  End Sub
  Public Shadows Sub G(ByVal i As Integer)
  End Sub
End Class
```

Paramètres de méthode

Vous devez définir les paramètres de méthode VB .NET en utilisant des paramètres d'opération.

Vous pouvez définir les modificateurs de paramètre suivants dans PowerAMC :

Modificateur VB .NET	Equivalent PowerAMC
ByVal	Sélectionnez *Entrée* dans la zone Type de paramètre de l'onglet Général de la feuille de propriétés d'un paramètre

Modificateur VB .NET	Equivalent PowerAMC
ByRef	Sélectionnez *Entrée/Sortie* ou *Sortie* dans la zone Type de paramètre de l'onglet Général de la feuille de propriétés d'un paramètre
Optional	Définissez l'attribut étendu *Optional* à True sur l'onglet Attributs étendus
ParamArray	Cochez la case *Argument variable* sur l'onglet Général de la feuille de propriétés de paramètre

Réalisation de méthode

Les méthodes de classe sont réalisées par les opérations d'interface correspondantes. Pour définir la réalisation des méthodes d'une classe, vous devez utiliser le bouton A réaliser de l'onglet Opérations d'une feuille de propriétés de classe, puis cliquer sur le bouton Réaliser, et ce pour chaque méthode à réaliser. La méthode s'affiche avec le stéréotype <<*Implement*>>.

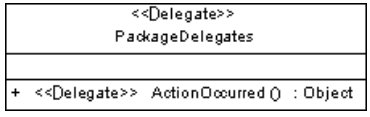
Constructeur & destructeur

Vous modélisez les *constructeurs* et *destructeurs* VB .NET en cliquant sur le bouton **Ajouter > Constructeur / Destructeur par défaut** dans la liste des opérations d'une classe. Vous créez ainsi automatiquement un constructeur appelé New et ayant le stéréotype Constructor, ainsi qu'un destructeur appelé Finalize ayant le stéréotype Destructor. Le constructeur et le destructeur sont grisés dans la liste, ce qui signifie que vous ne pouvez pas modifier leur définition, mais vous pouvez toutefois les supprimer de la liste.

Délégué (delegate)

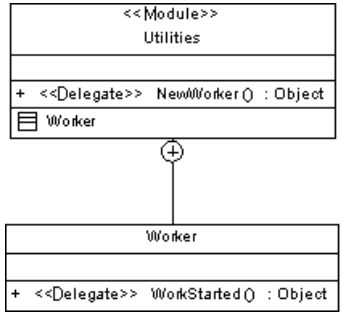
Vous pouvez concevoir l'un des types de *délégué (delegate)* VB .NET suivants :

- Pour créer un délégué au niveau de l'espace de noms, créez une classe avec le stéréotype <<*Delegate*>>, puis ajoutez une opération ayant le stéréotype <<*Delegate*>> à cette classe et définissez la visibilité de cette opération. Cette visibilité devient la visibilité pour ce délégué. Le nom de cette classe n'est pas important dans la mesure où elle ne sera pas générée. Ajoutez une ou plusieurs opérations sans stéréotype à cette classe




```
...
Public Delegate Function ActionOccurred () As Object
...
```

- Pour créer un délégué dans une classe, dans un module ou dans une structure, il vous suffit de créer une opération ayant le stéréotype <<*Delegate*>>. Dans l'exemple suivant, la classe Worker est interne au module Utilities. Ces deux éléments contiennent des délégués internes modélisés comme des opérations ayant le stéréotype <<Delegate>>

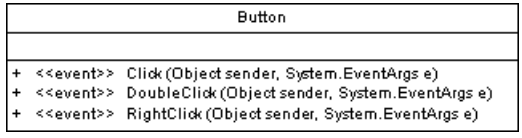



```
...
Public Module Utilities
 Public Delegate Function NewWorker () As Object
 Public Class Worker
  Public Delegate Function WorkStarted () As Object
 End Class
End Module
...
```

Evénement

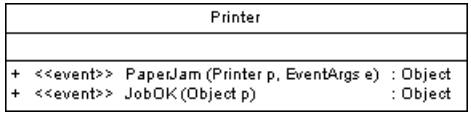
Pour définir un événement dans VB .NET, vous devez déclarer sa signature. Vous pouvez pour ce faire utiliser un délégué comme type pour cet événement, ou bien définir la signature sur l'événement lui-même. Ces deux types de déclarations peuvent être combinées au sein d'une classe.

Le délégué utilisé comme type est représenté par un attribut ayant le stéréotype <<*Event*>>. Vous définissez le nom du délégué à l'aide du type de données de l'attribut.




```
Public Class Printer
 Public PaperJam As EventHandler
 Public OutOfPaper As EventHandler
 Public JobOK As PrinterGoodDelegate
End Class
```

Lorsque vous définissez la signature sur l'événement lui-même, vous devez utiliser l'opération avec le stéréotype <<*Event*>>. La signature de cette opération devient ensuite la signature de l'événement.




```
Public Class Printer
 Public Event PaperJam(ByVal p As Printer, ByVal e As EventArgs)
 Public Event JobOK(ByVal p As Object)
End Class
```

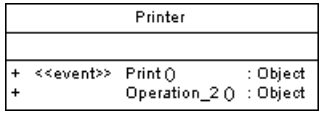
Mise en oeuvre de l'événement

Pour modéliser la clause de mise en oeuvre d'un délégué utilisé comme type, vous devez saisir une clause dans l'attribut étendu *implements* de l'attribut <<Event>>.

Dans le cas d'opérations <<Event>>, vous devez utiliser la fonctionnalité A réaliser dans la liste des opérations de la classe.

Gestionnaire d'événements (Event Handler)

Pour modéliser un *gestionnaire d'événements (event handler)* VB .NET, vous devez avoir créé une opération ayant le stéréotype <<*event*>> dans votre classe. Vous devez ensuite créer une autre opération, et spécifier le nom de l'opération <<event>> dans la zone Valeur de l'attribut étendu *Handles*.




```
...
Public Function Operation_2() As Object Handles Print
 End Function
...
```

Méthode externe

Vous modélisez une méthode externe VB .NET en utilisant une opération ayant le stéréotype <<*External*>>. Les méthodes externes ont les mêmes propriétés que les méthodes standard.

Vous pouvez également définir les propriétés spécifiques suivantes pour une méthode externe :

- *Alias clause* : vous pouvez utiliser l'attribut étendu *Nom d'alias* pour spécifier des ordinaux numériques (préfixés par un caractère @) ou un nom pour une méthode externe
- *Library clause* : vous pouvez utiliser l'attribut étendu *Nom de bibliothèque* pour spécifier le nom du fichier externe qui met en oeuvre la méthode externe
- Les modificateurs *Ansi, Unicode* et *Auto* utilisés pour appeler la méthode externe peuvent être définis à l'aide de l'attribut étendu *Jeu de caractères* de la méthode externe

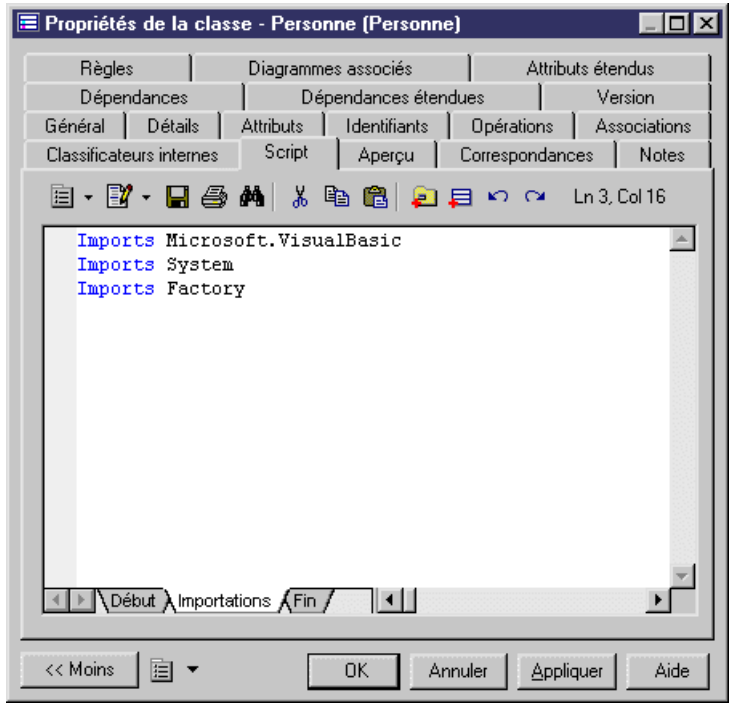
Génération de fichiers VB.NET

Vous générez des fichiers source VB.NET à partir des classes et des interfaces d'un modèle. Un fichier distinct, portant le suffixe de nom de fichier .vb, est généré pour chaque classe ou interface sélectionnée dans le modèle, ainsi qu'un fichier journal de génération.

Lors de la génération VB .NET, chaque objet racine c'est-à-dire chaque classe, interface, module etc. donne lieu à la génération d'un fichier source portant le suffixe .vb. Les classificateurs internes sont générés dans la source de classificateur du conteneur.

La directive *Imports* peut s'afficher au début du script de chaque objet généré.

Vous pouvez définir des imports dans PowerAMC en utilisant l'onglet Script\Importations d'une feuille de propriétés de classe. Vous pouvez saisir l'instruction Import ou bien utiliser l'outil Importer un dossier ou Importer un classificateur.



Les options s'affichent dans l'en-tête du fichier généré. Vous pouvez définir les options suivantes pour les principaux objets :

- *Compare* : saisissez la valeur Text ou Binary dans la zone Valeur pour l'attribut étendu Compare de l'objet généré

- *Explicit* : sélectionnez True ou False dans la zone Valeur pour l'attribut étendu Explicit de l'objet généré
- *Strict* : sélectionnez True ou False dans la zone Valeur pour l'attribut étendu Strict de l'objet généré

Les variables PowerAMC suivantes sont utilisées dans la génération de fichiers source VB.NET :

Variable	Description
VBC	Chemin d'accès complet du compilateur VB .NET. Par exemple, C:\WINDOWS \Microsoft.NET\Framework\v1.0.3705\vbc.exe
WSDL	Chemin d'accès complet du générateur de proxy de service Web. Par exemple, C: \Program Files\Microsoft Visual Studio .NET\FrameworkSDK\Bin\wsdl.exe

Pour passer en revue ou éditer ces variables, sélectionnez **Outils > Options générales** puis cliquez sur la catégorie **Variables**.

1. Sélectionnez **Langage > Générer du code VB.NET** pour afficher la boîte de dialogue de génération VB.NET.
2. Spécifiez le répertoire dans lequel les fichiers doivent être générés, et indiquez si vous souhaitez effectuer une vérification de modèle (voir *Chapitre 9, Vérification d'un MOO* à la page 303).
3. [facultatif] Sélectionnez des cibles supplémentaires pour la génération. Ces cibles sont définies par les extensions que vous pouvez attacher au modèle (voir *Gestion des cibles de génération* à la page 286).
4. [facultatif] Cliquez sur l'onglet **Sélection** et spécifiez les objets à partir desquels vous souhaitez générer. Par défaut, tous les objets sont générés.
5. [facultatif] Cliquez sur l'onglet **Options** et sélectionnez les options de génération appropriées :

Options	Description
Génération de code Visual Basic .NET de service Web dans un fichier .ASMX plutôt que dans un fichier .VB	Génère le code Visual Basic dans le fichier .ASMX
Génération de fichiers de projet Visual Studio .NET	Génère les fichiers du projet Visual Studio .NET. Un fichier de solution est généré avec plusieurs fichiers de projet, chaque projet correspondant à un modèle ou à un package ayant le stéréotype <<Assembly>>

Options	Description
Génération des ID d'objet sous forme de balises de documentation	Génère des informations utilisées pour le reverse engineering, telles que les identificateurs d'objet (@pdoid) qui sont générés sous forme de balises de documentation. Si vous ne souhaitez pas que ces balises soient générées, Vous devez définir cette option à False
Version de Visual Studio .NET	Indique le numéro de version de Visual Studio .NET

Remarque : Pour plus d'informations sur la modification des options qui s'affichent sur cet onglet et sur l'onglet **Tâches** ainsi que sur l'ajout de vos propres options et tâches, voir *Personnalisation et extension de PowerAMC > Fichiers de définition pour les langage objet, de processus et XML > Catégorie Generation*.

6. [facultatif] Cliquez sur l'onglet **Fichiers générés** et spécifiez les fichiers à générer. Par défaut, tous les fichiers sont générés.

   Pour plus d'informations sur la personnalisation des fichiers qui seront générés, voir *Personnalisation et extension de PowerAMC > Fichiers d'extension > Fichiers générés (Profile)*.

7. [facultatif] Cliquez sur l'onglet **Tâches** et spécifiez les tâches de génération à effectuer :

Tâche	Description
Génération du code de proxy de service Web (WSDL)	Génère la classe proxy
Compilation des fichiers source Visual Basic .NET	Compile les fichiers source
Ouverture de la solution dans Visual Studio .NET	Si vous sélectionnez l'option Génération de fichiers de projet Visual Studio .NET, cette tâche permet d'ouvrir la solution dans l'environnement de développement Visual Studio .NET

8. Cliquez sur **OK** pour lancer la génération.

   Une fois la génération terminée, la boîte de dialogue Fichiers générés s'affiche et répertorie les fichiers générés dans le répertoire spécifié. Sélectionnez un fichier dans liste, puis cliquez sur **Editer** pour l'ouvrir dans votre éditeur associé, ou bien cliquez sur **Fermer** pour quitter la boîte de dialogue.

Reverse engineering de code VB .NET

Vous pouvez procéder au reverse engineering de fichiers VB .NET dans un MOO.

Dans l'onglet Sélection, vous pouvez choisir de faire porter le reverse engineering sur des fichiers, sur des répertoires ou sur des projets.

Vous avez également la possibilité de définir un répertoire de base. Le répertoire de base est le répertoire racine commun pour tous les fichiers sur lesquels vous devez faire porter le reverse engineering. Ce répertoire de base sera utilisé lors de la génération pour recréer à l'identique la structure des fichiers sur lesquels vous avez effectué le reverse engineering.

Edition de la source

Vous pouvez pointer sur les fichiers sur lesquels vous souhaitez réaliser un reverse engineering, cliquer le bouton droit de la souris et sélectionner la commande Editer pour visualiser le contenu de ces fichiers. Pour pouvoir utiliser cette commande, vous devez avoir préalablement associé le suffixe de nom de fichier avec un éditeur dans la boîte de dialogue Options générale\Editeur.

Sélection des options de reverse engineering pour VB .NET

Vous pouvez définir les options de reverse engineering pour VB .NET dans la boîte de dialogue Reverse engineering de VB .NET :

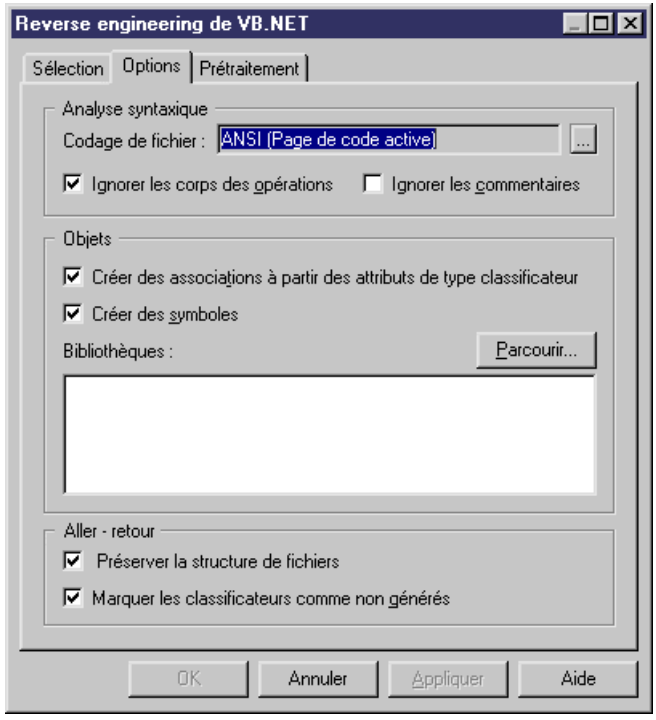
Option	Résultat
Codage de fichier	Permet de modifier le codage de fichier par défaut pour les fichiers sur lesquels vous souhaitez faire porter le reverse engineering
Ignorer les corps des opérations	Effectue le reverse engineering sans y inclure le corps du code
Ignorer les commentaires	Effectue le reverse engineering des classes sans y inclure les commentaires
Créer des associations à partir des attributs de type classificateurs	Créer des associations entre les classes et/ou les interfaces
Créer des symboles	Crée un symbole pour chaque objet dans le diagramme. Si vous décochez cette case, les objets récupérés ne sont visibles que dans l'Explorateur d'objets

Option	Résultat
Bibliothèques	Spécifie une liste de modèles de bibliothèque à utiliser comme référence lors du reverse engineering. Le modèle de destination du reverse engineering peut contenir des raccourcis vers des objets définis dans une bibliothèque. Si vous spécifiez la bibliothèque ici, le lien entre le raccourci et son objet cible (contenu dans la bibliothèque) sera préservé et la bibliothèque sera ajoutée à la liste des modèles cibles dans le modèle de destination du reverse engineering. Vous pouvez faire glisser les bibliothèques dans la liste afin de spécifier une hiérarchie de bibliothèques. PowerAMC tentera de résoudre les raccourcis trouvés dans le modèle de destination du reverse engineering en utilisant tour à tour chacune des bibliothèques spécifiées. Ainsi, si la bibliothèque v1.1 apparaît dans la liste avant la bibliothèque v1.0, PowerAMC tentera d'abord de résoudre les raccourcis vers la bibliothèque v1.1 et analysera uniquement la bibliothèque v1.0 s'il reste des raccourcis non résolus. Vous devez utiliser la liste des modèles cibles pour gérer les bibliothèques liées au modèle de destination du reverse engineering, par exemple, vous pouvez changer la version de bibliothèque (voir *Guide des fonctionnalités générales > Liaison et synchronisation de modèles > Raccourcis et répliques > Utilisation des modèles cible*).
Préserver la structure de fichiers	Crée un artefact lors du reverse engineering afin d'être en mesure de régénérer une structure de fichiers à l'identique
Marquer les classificateurs comme non générés	Les classificateurs récupérés par reverse engineering (classes et interfaces) ne seront ensuite plus générés à partir du modèle. Pour pouvoir les générer, vous devez cochez la case Générer dans leur feuille de propriétés respective

Définition d'options de reverse engineering VB .NET

Pour définir des options de reverse engineering VB .NET :

1. Sélectionnez **Langage > Reverse engineering VB .NET**.
2. Cliquez sur l'onglet Options.



3. Sélectionnez ou désélectionnez les options appropriées.
4. Si nécessaire utilisez le bouton Parcourir pour sélectionner le répertoire des bibliothèques.
5. Cliquez sur Appliquer, puis sur Annuler.

Prétraitement lors du reverse engineering VB .NET

Les fichiers .NET peuvent contenir du code conditionnel qui doit être géré par des *directives de prétraitement* lors du reverse engineering. Une directive de prétraitement est une commande placée au sein du code source qui demande au compilateur d'effectuer une tâche particulière avant que le reste du code ne soit analysé et compilé. La structure d'une directive de prétraitement est la suivante :

```
#directive symbole
```

est suivi du nom de la directive, et le *symbole* est une constante de compilateur conditionnelle utilisée pour sélectionner des sections particulières de code et pour exclure d'autres sections.

Dans VB .NET, les symboles ont des valeurs.

Dans l'exemple suivant, la directive #if est utilisée avec le symboles FrenchVersion et GermanVersion pour produire des versions en allemand ou en français d'une même application à partir du même code source :

```
#if FrenchVersion Then
  '  <code specific to French language version>.
#ElseIf GermanVersion Then
  ' <code specific to French language version>.
#Else
  ' <code specific to other language version>.
#End If
```

Vous pouvez déclarer une liste de symboles pour les directives de prétraitement. Ces symboles sont analysés par des directives de prétraitement : si la condition de la directive est true, l'instruction est conservée ; dans le contraire, l'instruction est supprimée.

Directives de prétraitement VB .NET prises en charge

Les directives suivantes sont prises en charge lors du prétraitement :

Directive	Description
#Const	Définit un symbole
#If	Evalue une condition. Si la condition est vérifiée, l'instruction suivant cette condition est conservée ; dans le cas contraire, elle est ignorée
#Else	Si le test #If précédent échoue, le code source suivant la directive #Else sera inclus
#Else If	Utilisé avec la directive #if, si le précédent test #If échoue, #Else If inclut ou exclut du code source, selon la valeur résultant de sa propre expression ou selon son identifiant
#End If	Ferme le bloc de code conditionnel #If

Note : les directives #Region, #End Region et #ExternalSource sont supprimées du code source.

Définition d'un symbole de prétraitement VB .NET

Vous pouvez redéfinir des symboles de prétraitement VB .NET dans l'onglet prétraitement de la boîte de dialogue de reverse engineering.

Les noms de symbole sont sensibles à la casse et doivent être uniques. Assurez-vous de ne pas saisir des mots réservés tels que true, false, if, do etc. Vous devez systématiquement affecter une valeur à un symbole, cette valeur peut être une chaîne (aucun " " requis), une valeur numérique, une valeur booléenne ou Nothing.

La liste des symboles est enregistrée dans le modèle et sera réutilisée lorsque vous synchroniserez votre modèle avec du code existant en utilisant la commande Synchroniser avec les fichiers générés.

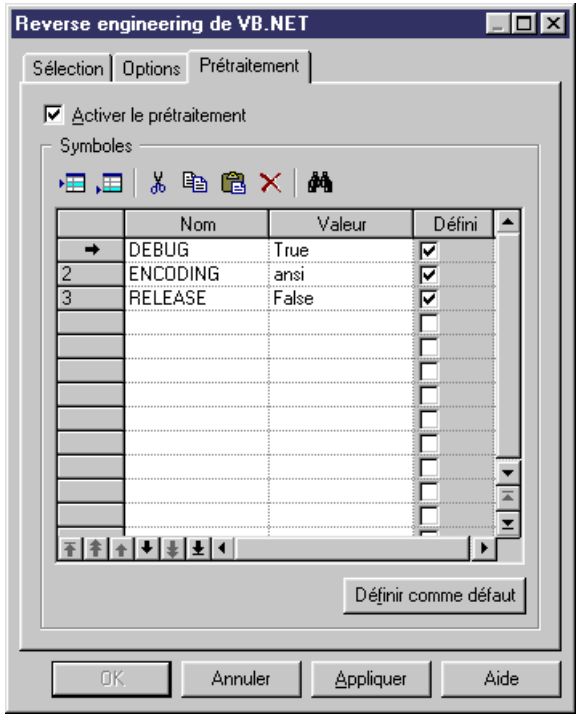
Pour plus d'informations sur la commande Synchroniser avec les fichiers générés, voir *Synchronisation d'un modèle avec des fichiers générés* à la page 290.

Vous pouvez utilisez le bouton Enregistrer comme défaut pour enregistrer la liste des symboles dans le Registre.

1. Sélectionnez **Langage > Reverse engineering VB .NET**.

   La boîte de dialogue Reverse engineering VB .NET s'affiche.
2. Cliquez sur l'onglet Prétraitement, puis cliquez sur l'outil Ajouter une ligne pour insérer une ligne dans la liste.
3. Saisissez les noms de symbole dans la colonne Nom.
4. Saisissez la valeur pour chaque symbole dans la colonne Valeur.

   La case Défini est automatiquement cochée pour chaque symbole afin d'indiquer que le symbole sera pris en compte lors du prétraitement.



5. Cliquez sur Appliquer.

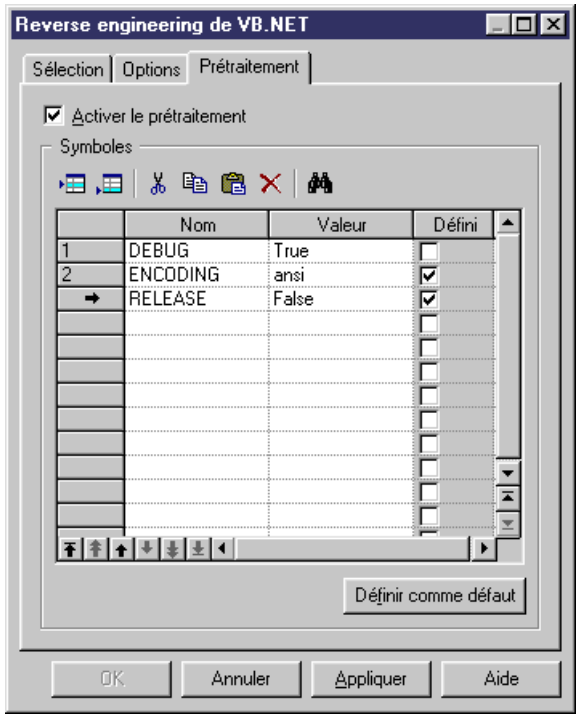
Reverse engineering VB .NET avec prétraitement

Le prétraitement est une option que vous pouvez activer ou désactiver lorsque vous procédez au reverse engineering d'un modèle.

1. Sélectionnez **Langage > Reverse engineering VB** .NET.

   La boîte de dialogue Reverse engineering VB .NET s'affiche.
2. Sélectionnez les fichiers sur lesquels vous souhaitez faire porter le reverse engineering dans l'onglet Sélection.
3. Sélectionnez les options de reverse engineering dans l'onglet Options.

4. Cochez la case Activer le prétraitement dans l'onglet Prétraitement.
5. Sélectionnez les symboles dans la liste des symboles.



6. Cliquez sur OK pour lancer le reverse engineering.

   Lorsque le prétraitement est terminé, le code est passé au reverse engineering.

Reverse engineering de fichiers VB .NET

Vous pouvez procéder au reverse engineering de fichiers VB .NET.

1. Sélectionnez **Langage > Reverse engineering VB .NET** pour afficher la boîte de dialogue Reverse engineering de VB .NET.
2. Sélectionnez Fichiers VB .NET ou Répertoires VB .NET dans la liste Reverse engineering.
3. Cliquez sur le bouton Ajouter dans l'onglet Sélection.

   Une boîte de dialogue standard de sélection de fichiers s'affiche.
4. Sélectionnez les éléments ou répertoires sur lesquels vous souhaitez faire porter le reverse engineering, puis cliquez sur le bouton Ouvrir.

Remarque : Vous pouvez sélectionner plusieurs fichiers à la fois pour le reverse engineering, en utilisant les touches **Ctrl** et **Maj**. Vous ne pouvez pas sélectionner plusieurs répertoires à la fois.

La boîte de dialogue de reverse engineering affiche les fichiers que vous avez sélectionnés.

5. Cliquez sur OK.

   Une boîte de progression s'affiche. Si le modèle dans lequel vous effectuez un reverse engineering contient déjà des données, une boîte de dialogue de fusion s'affiche.

   Pour plus d'informations sur la fusion de modèles, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Comparaison et fusion de modèles*.

   Les classes sont automatiquement ajoutées dans votre modèle et sont visibles dans le diagramme. Elles figurent également dans l'onglet Reverse de la fenêtre Résultats, située en bas de la fenêtre principale.

Utilisation d'ASP.NET

Un ASP (Active Server Page) est une page HTML qui inclut un ou plusieurs scripts (petits programmes incorporés) qui sont interprétés par un interpréteur de script (VBScript ou JScript) et traités sur un serveur Web Microsoft avant que la page ne soit envoyée à l'utilisateur. Un ASP implique des programmes exécutés sur un serveur et qui le plus souvent adaptent une page pour l'utilisateur. Le script contenu dans la page Web située sur le serveur utilise les informations reçues en retour d'une requête émise par l'utilisateur pour que la page accède aux données d'une base de données, puis construit ou personnalise la page à la volée avant de l'envoyer au demandeur.

ASP.NET (également appelé ASP+) représente la prochaine génération de pages ASP (Active Server Page) de Microsoft. ASP et ASP.NET permettent à un concepteur de site Web de construire de façon dynamique des pages Web à la volée en insérant des requêtes dans une base de données relationnelles dans la page Web. ASP.NET diffère de ses prédécesseurs pour les raisons suivantes :

- Il prend en charge le code écrit dans des langages compilés tels que Visual Basic, VB .NET et Perl
- Il inclut des contrôles serveur qui peuvent séparer le code du contenu, permettant une édition WYSIWYG des pages

Les fichiers ASP.NET sont dotés d'un suffixe .ASPX. Dans un MOO, un ASP.NET est représenté par un objet fichier et est lié à un composant (de type ASP.NET). Le type de composant Active Server Page (ASP.NET) permet d'identifier ce composant. Les composants de ce type sont liés à un seul objet fichier qui définit la page.

Lorsque vous affectez le type ASP.NET à un composant, le fichier ASP.NET approprié est automatiquement créé, ou bien attaché s'il existe déjà. Vous pouvez voir l'objet fichier ASP.NET dans l'onglet Fichiers de la feuille de propriétés du composant.

Onglet ASP de la feuille de propriétés du composant

Lorsque vous affectez le type ASP.NET à un composant, l'onglet **ASP** s'affiche automatiquement dans la feuille de propriétés du composant.

L'onglet **ASP** de la feuille de propriétés de composants contient les propriétés suivantes :

Propriété	Description
Fichier ASP	Objet fichier qui définit la page. Vous pouvez cliquer sur l'outil **Propriétés** en regard de cette zone pour afficher la feuille de propriétés de l'objet fichier, ou bien cliquer sur l'outil **Créer** pour créer un objet fichier
Template par défaut	Attribut étendu qui permet de sélectionner un template pour la génération. Son contenu peut être défini par l'utilisateur ou fourni par défaut

Pour modifier le contenu par défaut, éditez le langage objet courant en sélectionnant **Langage > Editer le langage objet courant** puis modifiez l'élément suivant : Profile/FileObject/Criteria/ASP/Templates/DefaultContent<%is(DefaultTemplate)%>. Créez ensuite les templates et renommez-les DefaultContent<%is(<name>)%> où <name> correspond au nom de template DefaultContent correspondant.

Pour définir des templates DefaultContent supplémentaire pour ASP.NET, vous devez modifier le type d'attribut étendu ASPTemplate dans l'entrée Profile/Share/Extended Attribute Types et ajouter de nouvelles valeurs correspondant au nom respectif des nouveaux templates.

Pour plus d'informations sur la propriété Template par défaut, reportez-vous à la définition de TemplateContent dans *Création d'un ASP.NET à l'aide de l'Assistant* à la page 472.

Définition des objets fichier pour les ASP.NET

Le contenu de l'objet fichier pour les ASP est basé sur un template spécial appelé DefaultContent défini par rapport à la métaclasse FileObject. Il est situé dans la catégorie Profile/FileObject/Criteria/ASP/Templates des langages objet C# et VB.NET. Ce lien vers le template est fourni comme base de travail, si vous éditez l'objet fichier, le lien vers le template est perdu, ce mécanisme s'apparente à celui en vigueur pour les corps d'opération par défaut.

Pour plus d'informations sur la catégorie Criteria, voir *Personnalisation et extension de PowerAMC > Fichiers d'extension > Critères (Profile)*.

Les fichiers Active Server Page sont identifiés par le stéréotype ASPFile. Le nom de page ASP est synchronisé avec le nom du composant ASP.NET conformément aux conventions spécifiées dans la zone Valeur de l'entrée Settings/Namings/ASPFileName des langages C# et VB.NET.

Vous pouvez pointer sur un objet fichier, cliquer le bouton droit de la souris et sélectionner **Ouvrir avec > *éditeur de texte*** dans le menu contextuel pour afficher le contenu de ce fichier

Création d'un ASP.NET à l'aide de l'Assistant

Vous pouvez utiliser l'*Assistant* pour créer un ASP.NET. Cet Assistant vous guide tout au long des différentes étapes de création du composant et est disponible uniquement si le langage du modèle est C# ou VB.NET.

Vous pouvez soit créer un ASP.NET sans sélectionner d'objet fichier, soit commencer par sélectionner un objet fichier, puis lancer l'Assistant à partir du menu Outils.

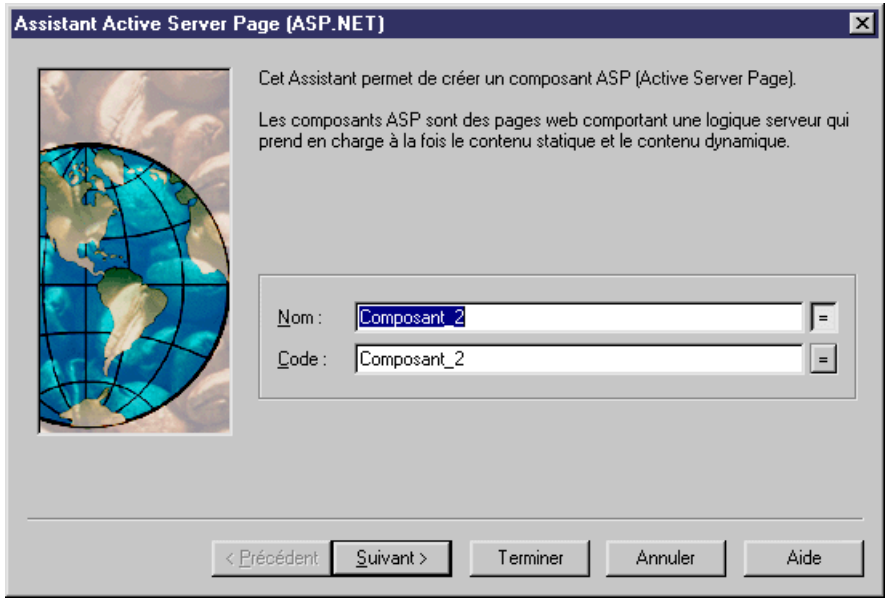
Vous avez également la possibilité de créer plusieurs ASP.NET du même type en sélectionnant plusieurs objets fichier simultanément. L'Assistant crée alors automatiquement un ASP.NET par objet fichier. Les objets fichier que vous avez sélectionnés dans le diagramme de classes deviennent les fichiers .JSP.NET.

L'Assistant de création d'un ASP.NET permet de définir les paramètres suivants :

Page de l'Assistant	Description
Nom	Nom du composant ASP.NET
Code	Code du composant ASP.NET
TemplateContent	Permet de choisir le template par défaut de l'objet fichier ASP.NET. TemplateContent est un attribut étendu situé dans la catégorie Profile/Component/Criteria/ASP.NET des langages objet C# et VB.NET. Si vous ne modifiez pas le contenu de l'objet fichier, le contenu par défaut est conservé (il voit l'onglet Contenu de la feuille de propriétés de l'objet fichier). Tous les templates sont disponible dans la catégorie FileObject/Criteria/ASP/templates du langage objet courant
Créer le symbole dans	Crée un symbole de composant dans le diagramme de composants spécifié dans la liste. S'il existe déjà un diagramme de composants, vous pouvez le sélectionner dans la liste. Vous pouvez également afficher la feuille de propriétés du diagramme sélectionné en cliquant sur l'outil Propriétés

1. Sélectionnez **Outils > Créer un ASP** à partir d'un diagramme de classes.

   La boîte de dialogue Assistant Activer Server Page s'affiche.



2. Sélectionnez un nom et un code pour le composant ASP.NET, puis cliquez sur Suivant.
3. Sélectionnez un template ASP.NET, puis cliquez sur Suivant.
4. A la fin de l'Assistant, vous devez définir la création des symboles et diagrammes.

Une fois que vous avez fini d'utiliser l'Assistant, les actions suivantes sont exécutées :

- Un composant ASP.NET et un objet fichier avec une extension .ASPX sont créés et visibles dans l'Explorateur d'objets. Le nom de l'objet fichier est défini d'après le nom du composant d'origine par défaut pour préserver la cohérence
- Si vous affichez la feuille de propriétés de l'objet fichier, vous pouvez voir que la propriété Artefact est sélectionnée

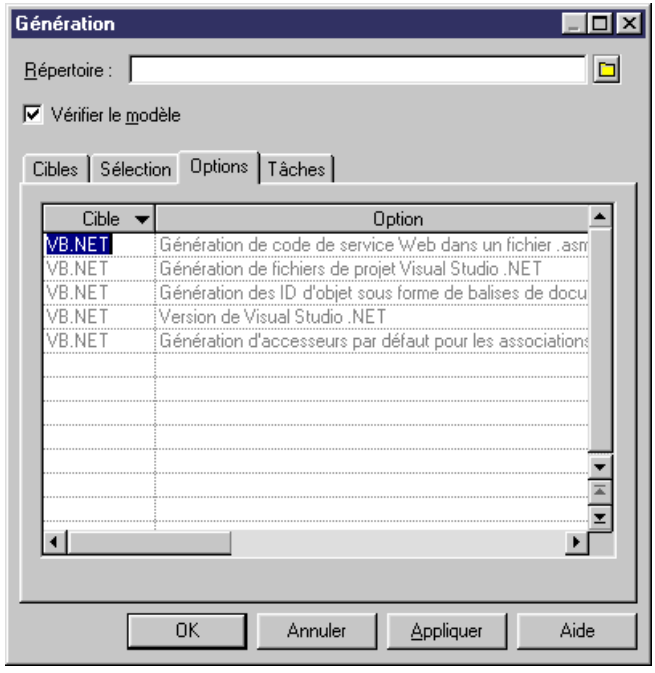
  Pour plus d'informations sur les fichiers d'artefact, voir *Propriétés de l'objet fichier* à la page 236.
- Vous pouvez éditer l'objet fichier directement dans l'éditeur interne de PowerAMC, si le suffixe de nom de fichier correspond à un suffixe défini dans la page Editeurs de la boîte de dialogue Options générales, et si le mot clé <internal> est défini dans les colonnes Nom de l'éditeur et Commande de l'éditeur pour ce suffixe

Génération de ASP.NET

Le processus de génération ne prend en compte que les objets fichier dont la propriété Artefact est sélectionné.

1. Sélectionnez **Langage > Générer du code C# ou VB.NET** pour afficher la boîte de dialogue Génération.

2. Sélectionnez un répertoire qui contiendra les fichiers générés.
3. Cliquez sur l'onglet Sélection, puis sélectionnez les objets appropriés sur les différentes pages.
4. Cliquez sur Appliquer.
5. Cliquez sur l'onglet Options, puis spécifiez les options de génération appropriées dans l'onglet Options.



Pour plus d'informations sur les options de génération, voir *Génération de fichiers VB.NET* à la page 461.

6. Cliquez sur **Appliquer**.
7. Cliquez sur l'onglet **Tâches**.
8. Sélectionnez les commandes que vous souhaitez exécuter lors de la génération sur l'onglet **Tâches**.

Pour plus d'informations sur les tâches de génération, voir *Génération de fichiers VB.NET* à la page 461.

Vous devez avoir préalablement défini les variables d'environnement via la section Variables de la boîte de dialogue Options générales afin que ces commandes soient activées dans cet onglet.

Pour plus d'informations la définition de ces variables, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Personnalisation de votre environnement de modélisation > Options générales > Variables d'environnement*.

9. Cliquez sur **OK**.

Une boîte de progression s'affiche, suivie par une liste Résultats. Vous pouvez utiliser le bouton **Editer** dans la liste Résultats pour éditer chaque fichier généré individuellement.

10. Cliquez sur **Fermer**.

Les fichiers sont générés dans le répertoire spécifié pour la génération.

Chapitre 17 C# 2.0

PowerAMC permet la prise en charge complète pour la modélisation de tous les aspects de C# 2.0, y compris l'ingénierie par va-et-vient.

C# 2.0 est un langage orienté objet moderne de type sécurisé, qui combine les avantages du développement rapide et la puissance de C++.

Outre les palettes standard de PowerAMC, les outils personnalisés suivants sont disponibles pour vous aider à développer rapidement vos diagrammes de classes et de structure composite :

Icône	Outil
	Assembly – collection de fichiers C# (voir *Assemblies C# 2.0* à la page 477).
	Attribut personnalisé – pour ajouter des métadonnées (voir *Attributs personnalisés C# 2.0* à la page 493).
	Delegate – classes de type référence sécurisé (voir *Délégués (delegates) C# 2.0* à la page 485).
	Enum – jeu de contraintes nommées (voir *Enumérations (enums) C# 2.0* à la page 486).
	Struct – types légers (voir *Structures (Structs) C# 2.0* à la page 484).

Assemblies C# 2.0

Un assembly C# est une collection de # fichiers qui forment une DLL ou un exécutable. PowerAMC prend en charge à la fois les modèles mono-assembly (dans lesquels le modèle représente l'assembly) et les modèles multi-assembly (dans lesquels chaque assembly apparaît directement sous le modèle dans l'Explorateur d'objets, et est modélisé sous la forme d'un package UML standard doté d'un stéréotype <<Assembly>>).

Création d'un assembly

PowerAMC prend en charge les modèles mono-assembly et multi-assembly.

Par défaut, lorsque vous créez un MOO C# 2.0, le modèle lui-même représente un assembly. Pour continuer avec un modèle mono-assembly, insérez un type ou un espace de noms dans le diagramme racine. Le modèle sera par défaut un assembly mono-module, la racine du modèle représentant l'assembly.

Pour créer un modèle multi-assembly, insérez un assembly dans le diagramme racine de l'une des façons suivantes :

- Utilisez l'outil **Assembly** dans la Boîte à outils C# 2.0.
- Sélectionnez **Modèle > Objets Assembly** pour afficher la boîte de dialogue Liste des objets Assembly, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou sur un package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Assembly**.

Remarque : Si ces options ne sont pas disponibles, c'est que vous travaillez dans un modèle mono-assembly.

Conversion d'un modèle mono-assembly en modèle multi-assembly

Pour convertir un modèle mono-assembly en modèle multi-assembly, pointez sur le modèle dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Convertir en modèle Multi-Assembly**, saisissez un nom pour l'assembly qui va contenir tous les types de votre modèle dans la boîte de dialogue Création d'un assembly, puis cliquez sur **OK**.

PowerAMC convertit le modèle mono-assembly en modèle multi-assembly en insérant un nouvel assembly directement sous le noeud racine du modèle afin d'y regrouper tous les types présents dans le modèle. Vous pouvez ajouter des assemblies supplémentaires si nécessaire, mais uniquement comme enfant de la racine du modèle.

Propriétés d'un assembly

Les feuilles de propriétés d'assembly contiennent tous les onglets d'une feuille de propriétés de package standard, avec en plus des onglets spécifiques à C#, dont les propriétés sont répertoriées ci-dessous :

L'onglet **Application** contient les propriétés suivantes :

Propriété	Description
Générer le fichier de projet	Spécifie si un fichier de projet Visual Studio 2005 doit être généré pour l'assembly.
Nom de fichier de projet	Spécifie le nom du projet dans Visual Studio. La valeur par défaut est la valeur de la propriété de code de l'assembly.
Nom d'assembly	Spécifie le nom de l'assembly dans Visual Studio. La valeur par défaut est la valeur de la propriété de code de l'assembly.
Espace de noms racine	Spécifie le nom de l'espace de noms racine dans Visual Studio. La valeur par défaut est la valeur de la propriété de code de l'assembly

Propriété	Description
Type de résultat	Spécifie le type de l'application modélisée. Vous pouvez choisir l'une des valeurs suivantes : • Class library • Windows Application • Console Application
GUID du projet	Spécifie un GUID unique pour le projet. Ce champ sera complété automatiquement à la génération.

L'onglet **Assembly** contient les propriétés suivantes :

Propriété	Description
Générer les informations d'assembly	Spécifie si un fichier de manifeste d'assembly doit être généré.
Titre	Spécifie un titre pour le manifeste d'assembly. Ce champ est lié au champ Nom dans l'onglet Général.
Description	Spécifie une description facultative pour le manifeste d'assembly.
Société	Spécifie un nom de société pour le manifeste d'assembly.
Produit	Spécifie un nom de produit pour le manifeste d'assembly.
Copyright	Spécifie une notice de copyright pour le manifeste d'assembly.
Marque déposée	Spécifie une marque déposée pour le manifeste d'assembly.
Culture	Spécifie la culture prise en charge par l'assembly.
Version	Spécifie la version de l'assembly.
Version de fichier	Spécifie un numéro de version qui demande au compilateur d'utiliser une version particulière pour la ressource de version de fichier Win32.
GUID	Spécifie un GUID unique qui identifie l'assembly.
Visibilité d'assembly COM	Spécifie si les types contenus dans l'assembly seront accessibles à COM

Unités de compilation C# 2.0

Par défaut, PowerAMC génère un fichier source pour chaque classe, interface, délégué, ou tout autre type, et base la structure de répertoires source sur les espaces de noms définis dans le modèle.

Vous pouvez souhaiter utiliser plutôt plusieurs classificateurs dans un seul fichier source et/ou construire la structure des répertoires sans la calquer sur celle des espaces de noms.

Une unité de compilation permet de grouper plusieurs types dans un seul fichier source. Elle se compose de zéro ou plus directives using suivies de zéro ou plus attributs globaux ou namespace-member-declarations (déclarations de membre d'espace de noms).

PowerAMC modélise les unités de compilation sous la forme d'artefacts avec un stéréotype <<Source>> et permet de construire une hiérarchie de répertoires source utilisant des dossiers. Les unités de compilation sont dépourvues de symbole dans le diagramme, et ne sont visibles que dans le dossier Artefacts dans l'Explorateur d'objets.

Vous pouvez afficher à tout moment un aperçu du code qui sera généré pour votre unité de compilation, il vous suffit pour ce faire d'afficher sa feuille de propriétés, puis de cliquer sur l'onglet **Aperçu**.

Création d'une unité de compilation

Pour créer une unité de compilation vide depuis l'Explorateur d'objets, pointez sur le noeud du modèle ou le dossier Artefacts, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Source**, saisissez un nom (sans oublier le suffixe .cs), puis cliquez sur **OK**.

Remarque : Vous pouvez créer une unité de compilation et la remplir avec un type à partir de l'onglet **Fichiers générés** de la feuille de propriétés du type en cliquant sur l'outil **Nouveau** dans la colonne **Artefacts**.

Ajout d'un type à une unité de compilation

Vous pouvez ajouter des types à une unité de compilation en procédant de l'une des façons suivantes :

- Faites glisser le symbole du type depuis le diagramme sur l'entrée de l'unité de compilation dans l'Explorateur d'objets.
- Affichez la feuille de propriétés de l'unité de compilation, cliquez sur l'onglet **Objets** et cliquez sur l'outil **Ajouter des objets de production**
- Affichez la feuille de propriétés du type, cliquez sur l'onglet **Fichiers générés**, puis cliquez sur l'outil **Ajouter/Supprimer** dans la colonne **Artefacts**. Les types ajoutés à plusieurs unités de compilation seront générés sous la forme de types partiels et vous pouvez spécifier dans quelle unité de compilation chacun de leurs attributs et méthodes sera généré.

Création d'une structure de dossiers de génération
Vous pouvez contrôler la structure des répertoires dans laquelle vos unités de compilation seront générées en utilisant les dossiers des artefact :

1. Pointez sur le modèle ou sur un dossier existant sous le dossier Artefacts dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Dossier d'artefact**.
2. Spécifiez un nom pour le dossier, puis cliquez sur **OK** pour le créer.
3. Ajoutez des unités de compilation au dossier faisant glisser leur entrée depuis sur l'entrée du dossier dans l'Explorateur d'objets, ou bien en pointant sur le dossier, en cliquant le bouton droit de la souris, puis en sélectionnant **Nouveau > Source**.

Remarque : Les dossiers ne peuvent contenir que des unités de compilation et/ou d'autres dossiers. Ainsi, pour placer un type dans la hiérarchie des dossiers de génération, vous devez commencer par l'ajouter dans une unité de compilation.

Types partiels

Les types partiels sont des types qui peuvent appartenir à plusieurs unités de compilation. Les types partiels sont préfixés du mot clé `partial`.

```
public partial class Server
{
  private int start;
}
```

Dans ce cas, vous pouvez spécifier à quelle unité de compilation chaque champ et méthode sera affecté, en utilisant la zone **Unité de compilation** sur l'onglet **C#** ou **VB** de leur feuille de propriétés.

Lorsqu'un type partiel contient des types internes, vous pouvez préciser l'unité de compilation à laquelle chaque type interne sera affecté de la façon suivante :

1. Ouvrez la feuille de propriétés du type contenant des types internes, et cliquez sur l'onglet **Classificateurs Internes**.
2. Si la colonne **CompilationUnit** n'est pas affichée, cliquez sur l'outil **Personnaliser les colonnes et filtrer**, cochez la case correspondant à cette colonne, puis cliquez sur **OK** pour revenir à l'onglet.
3. Cliquez sur la colonne **CompilationUnit** pour ouvrir une liste d'unités de compilation disponibles, sélectionnez-en une et cliquez sur **OK** pour fermer la feuille de propriétés.

Espaces de nom C# 2.0

Les espaces de nom permettent de limiter la portée du nom d'un objet. Chaque classe ou autre type doit avoir un nom unique à l'intérieur de l'espace de noms auquel il appartient.

PowerAMC modélise les espaces de noms sous la forme de packages standard avec la propriété **Utiliser l'espace de noms du parent** désélectionnée. Pour plus d'informations sur la création et l'utilisation des packages, voir *Packages (MOO)* à la page 53.

Dans l'exemple suivant, la classe Architect est déclarée dans le package Design qui est un sous-package de Factory. La déclaration d'espace de noms se présente comme suit :

```
namespace Factory.Design
{
 public class Architect
 {
 }
}
```

Cette structure, qui fait partie du modèle NewProduct, s'affiche dans l'Explorateur d'objets PowerAMC comme suit :

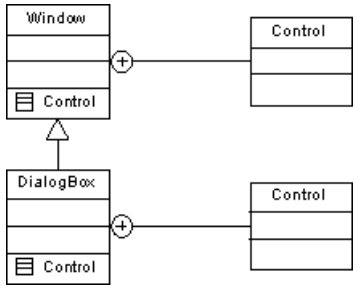
Les classificateurs définis directement au niveau du modèle appartiennent à l'espaces de noms global C#.

Classes C# 2.0

PowerAMC modélise les classes C# 2.0 comme des classes UML standard, mais dotées de propriétés supplémentaires.

Pour plus d'informations sur la création et l'utilisation des classes, voir *Classes (MOO)* à la page 37.

Dans l'exemple suivant, la classe DialogBox hérite de la classe Window, qui contient un classificateur interne Control, tout comme la classe DialogBox :

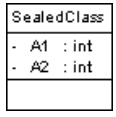



```
{
 public class DialogBox : Window
 {
  public new class Control
  {
  }
 }
}
```

Dans l'exemple suivant, la classe Client est définie comme abstraite en cochant la case **Abstrait** sur l'onglet **Général** de la feuille de propriétés de classe :


```
{
 public abstract class Client
 {
  private int Name;
  private int ID;

 }
}
```

Dans l'exemple suivant, la classe SealedClient est définie comme sealed en cochant la case **Final** sur l'onglet **Général** de la feuille de propriétés de classe :




```
{
 public sealed class SealedClass
 {
  private int A1;
  private int A2;

 }
}
```

Propriétés d'une classe C#

Les feuilles de propriétés de classe C# contiennent tous les onglets d'une feuille de propriétés de classe standard, avec en plus un onglet **C#**, dont les propriétés sont répertoriées ci-dessous :

Propriété	Description
Static	Spécifie le modificateur static pour la déclaration de classe.
Sealed	Spécifie le modificateur sealed pour la déclaration de classe.
New	Spécifie le modificateur new pour la déclaration de classe.
Unsafe	Spécifie le modificateur unsafe pour la déclaration de classe.

Interfaces C# 2.0

PowerAMC modélise les interfaces C# 2.0 sous forme d'interfaces UML standard, dotées de propriétés supplémentaires.

Pour plus d'informations sur la création et l'utilisation des interfaces, voir *Interfaces (MOO)* à la page 55.

Les interfaces C# peuvent contenir des événements, des propriétés, des indexeurs et des méthodes ; elles ne prennent pas en charge les variables, les constantes ni les constructeurs.

Propriétés d'une interface C#

Les feuilles de propriétés d'interface C# contiennent tous les onglets d'une feuille de propriétés d'interface standard, avec en plus un onglet C#, dont les propriétés sont répertoriées ci-dessous :

Propriété	Description
New	Spécifie le modificateur new pour la déclaration d'interface.
Unsafe	Spécifie le modificateur new pour la déclaration d'interface.

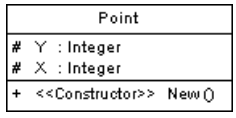
Structures (Structs) C# 2.0

Les structures (structs) sont des types légers qui envoient moins de demandes au système d'exploitation et à la mémoire que les classes conventionnelles. PowerAMC modélise les structures C# 2.0 sous forme de classes ayant le stéréotype <<Structure>>.

Pour plus d'informations sur la création et l'utilisation des classes, voir *Classes (MOO)* à la page 37.

Une structure peut mettre en oeuvre des interfaces mais ne prend pas en charge l'héritage ; elle peut contenir des événements, des variables, des constantes, des méthodes, des constructeurs et des propriétés.

Dans l'exemple suivant, la structure contient deux attributs :




```
{
 public struct Point
 {
  public int New()
  {
   return 0;
  }
  private int x;
  private int y;
 }
}
```

Création d'une structure

Vous pouvez créer une structure :

- Utilisez l'outil Structure dans la Boîte à outils **C# 2.0**.
- Sélectionnez **Modèle > Objets Structure** pour afficher la boîte de dialogue Liste des objets structure, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou sur un package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Structure**.

Propriétés d'une structure

Les feuilles de propriétés de structure contiennent tous les onglets d'une feuille de propriétés de classe standard, avec en plus un onglet **C#**, dont les propriétés sont répertoriées ci-dessous :

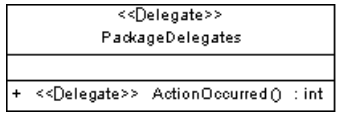
Propriété	Description
New	Spécifie le modificateur new pour la déclaration de structure.
Unsafe	Spécifie le modificateur unsafe pour la déclaration de structure.

Délégués (delegates) C# 2.0

Les délégués (delegates) sont des types référence sécurisés qui jouent un rôle similaire à celui des pointeurs fonction dans d'autres langages. PowerAMC modélise les délégués sous forme de classes ayant le stéréotype <<Delegate>> avec une opération unique portant le nom "<signature>". La visibilité, le nom, le commentaire, les marqueurs et les attributs sont spécifiés sur l'objet classe tandis que le type de résultat et les paramètres sont spécifiés sur l'opération.

Un délégué de niveau type (classe ou structure) est modélisé soit sous la forme d'une opération ayant le stéréotype <<Delegate>>, soit sous la forme d'un délégué au niveau de l'espace de noms dans lequel la classe représentant le délégué est interne au type conteneur.

Pour plus d'informations sur la création et l'utilisation des classes, voir *Classes (MOO)* à la page 37.




```
{
 public delegate int ActionOc-
curred();
}
```

Création d'un délégué

Vous pouvez créer un délégué de l'une des façons suivantes :

- Utilisez l'outil **Delegate** dans la Boîte à outils C# 2.0.
- Sélectionnez **Modèle > Objets Delegate** pour afficher la boîte de dialogue Liste des objets Delegate, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou sur un package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Delegate**.

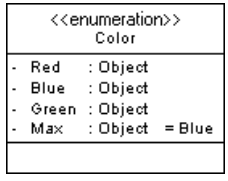
Propriétés d'un délégué

Les feuilles de propriétés de délégué contiennent tous les onglets d'une feuille de propriétés de classe standard, avec en plus un onglet **C#**, dont les propriétés sont répertoriées ci-dessous :

Propriété	Description
New	Spécifie le modificateur new pour la déclaration du délégué.
Unsafe	Spécifie le modificateur unsafe pour la déclaration du délégué.

Enumérations (enums) C# 2.0

Les énumérations (enums) sont des jeux de constantes nommées. PowerAMC modélise les énumérations sous la forme de classes ayant le stéréotype <<Enum>>.




```
{
 public enum Color : colors
 {
  Red,
  Blue,
  Green,
  Max = Blue
 }
}
```

Pour plus d'informations sur la création et l'utilisation des classes, voir *Classes (MOO)* à la page 37.

Création d'une énumération

Vous pouvez créer une énumération de l'une des façons suivantes :

- Utilisez l'outil **Enum** dans la Boîte à outils C# 2.0.

- Sélectionnez **Modèle > Objets Enum** pour afficher la boîte de dialogue Liste des objets Enum, puis cliquez sur l'outil **Ajouter une ligne**.
- Pointez sur le modèle (ou sur un package) dans l'Explorateur d'objets, cliquez le bouton droit de la souris, puis sélectionnez **Nouveau > Enum**.

Propriétés d'une énumération

Les feuilles de propriétés d'énumération C# contiennent tous les onglets d'une feuille de propriétés de classe standard, avec en plus un onglet **C#**, dont les propriétés sont répertoriées ci-dessous :

Propriété	Description
Type intégral de base	Spécifie le type intégral de base pour l'énumération.
New	Spécifie le modificateur new pour la déclaration d'énumération.

Champs C# 2.0

PowerAMC modélise les champs C# sous forme d'attributs UML standard.

Pour plus d'informations sur la création et l'utilisation des attributs, voir *Attributs (MOO)* à la page 69.

Propriétés d'un champ

Les feuilles de propriétés de champ C# contiennent tous les onglets d'une feuille de propriétés d'attribut standard, avec en plus un onglet C#, dont les propriétés sont répertoriées ci-dessous :

Propriété	Description
Unité de compilation	Spécifie l'unité de compilation dans laquelle le champ sera stocké. Disponible uniquement si le type parent est un type partiel (alloué à plusieurs unités de compilation).
New	Spécifie le modificateur new pour la déclaration du champ.
Unsafe	Spécifie le modificateur unsafe pour la déclaration du champ.
Const	Spécifie le modificateur const pour la déclaration du champ.
Readonly	Spécifie le modificateur readonly pour la déclaration du champ.

Méthodes C# 2.0

PowerAMC modélise les méthodes C# sous la forme d'opérations.

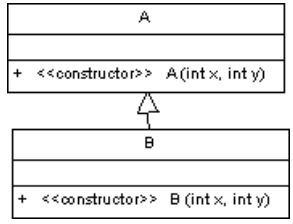
Pour plus d'informations sur la création et l'utilisation des opérations, voir *Opérations (MOO)* à la page 82.

Propriétés d'une méthode

Les feuilles de propriétés de méthode contiennent tous les onglets d'une feuille de propriétés d'opération standard, avec en plus un onglet C#, dont les propriétés sont répertoriées ci-dessous :

Propriété	Description
Unité de compilation	Spécifie l'unité de compilation dans laquelle la méthode sera stockée. Disponible uniquement si le type parent est un type partiel (alloué à plusieurs unités de compilation).
Extern	Spécifie le modificateur extern pour la déclaration de la méthode.
New	Spécifie le modificateur new pour la déclaration de la méthode. Lorsqu'une classe hérite d'une autre classe et contient des méthode avec une signature identique comme classe parent, ce champ est sélectionné automatiquement afin de faire en sorte que la méthode enfant prime sur la méthode parent.
Override	Spécifie le modificateur override pour la déclaration de la méthode.
Unsafe	Spécifie le modificateur unsafe pour la déclaration de la méthode.
Virtual	Spécifie le modificateur virtual pour la déclaration de la méthode.
Scope	Spécifie la portée de la méthode.

Propriété	Description

Initialisateur de base	Crée un initialisateur de constructeur d'instance de la forme de base. Il peut faire en sorte qu'un constructeur d'instance de la classe de base soit appelé.
	
	Dans l'exemple suivant, la classe B hérite de la classe A. Vous définissez un initialisateur de base dans le constructeur de la classe B, qui sera utilisé pour initialiser le constructeur de la classe A :
	
	 
	
	`internal class B : A`
	`{`
	`public B(int x, int y) : base(x + y, x - y)`
	`{}`
	`}`

Initialisateur This	Crée un initialisateur de constructeur d'instance, faisant en sorte qu'un constructeur de la classe elle-même soit appelé.

Constructeurs et destructeurs

Vous modélisez les *constructeurs* et *destructeurs* C# en cliquant sur le bouton **Ajouter Constructeur/Destructeur par défaut** sur l'onglet **Opérations** d'une feuille de propriétés de classe. Vous créez ainsi automatiquement un constructeur ayant le stéréotype Constructor, ainsi qu'un destructeur ayant le stéréotype Destructor. Le constructeur et le destructeur sont grisés dans la liste, ce qui signifie que vous ne pouvez pas modifier leur définition.

Réalisation de méthode

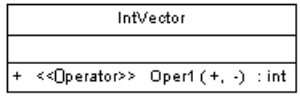
Les méthodes de classe sont réalisées par les opérations d'interface correspondantes. Pour définir la réalisation des méthodes d'une classe, vous devez utiliser le bouton **A réaliser** sur l'onglet **Opérations** de la feuille de propriétés de classe, puis cliquez sur le bouton **Réaliser** chaque méthode à réaliser. La méthode est affichée avec le stéréotype <<*Implement*>>.

Méthode d'opérateur

Vous modélisez un *opérateur* C# en utilisant une opération ayant le stéréotype <<*Operator*>>. Assurez-vous que l'opération <<Operator>> a une visibilité Public et que sa propriété Statique soit cochée.

Pour définir un *opérateur externe*, vous devez définir l'attribut étendu externe de l'opération à True. Les attributs étendus new, virtual et override ne sont pas admis pour les opérateurs.

L'opérateur *token* (par exemple +, -, !, ~, ou ++) est le no de la méthode.

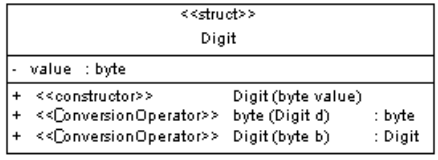


Méthode d'opérateur de conversion

Vous modélisez un *opérateur de conversion* C# en utilisant une opération ayant le stéréotype *<<ConversionOperator>>*.

Vous devez également déclarer l'opérateur de conversion en utilisant les mots-clés *explicit* ou *implicit*. Vous définissez le mot clé de l'opérateur de conversion en sélectionnant la valeur implicit ou explicit de l'attribut étendu *scope*.

Dans l'exemple suivant, la classe Digit contient un opérateur de conversion explicit et un opérateur de conversion implicite :




```
public struct Digit
{
 public Digit(byte value)
  {
   if (value < 0 || value > 9)
throw new ArgumentException();
   this.value = value;
  }
 public static implicit operator
byte(Digit d)
 {
  return d.value;
 }

 public static explicit operator
Digit(byte b)
  {
   return new Digit(b);
  }
 private byte value;
}
```

Evénements, indexeurs et propriétés C# 2.0

PowerAMC représente les événements, indexeurs et propriétés C# sous la forme d'attributs UML standard avec des propriétés supplémentaires.

Pour plus d'informations sur la création et l'utilisation des attributs, voir *Attributs (MOO)* à la page 69.

Création d'un événement, d'un indexeur ou d'une propriété

Pour créer événement, un indexeur ou une propriété, affichez la feuille de propriétés d'un type, cliquez sur l'onglet Attributs, cliquez sur le bouton **Ajouter** en bas de l'onglet, puis sélectionnez l'option appropriée.

Ces objets sont créés comme suit :

- Evénements – stéréotype <<Event>> avec une ou plusieurs opérations liées représentant les gestionnaires add et/ou remove
- Indexeurs – stéréotype <<Indexer>> avec une ou plusieurs opérations liées représentant les accesseurs get et/ou set
- Propriétés – stéréotype <<Property>> avec une ou plusieurs opérations liées représentant les accesseurs get et/ou set. En outre, vous devez noter que :
  - La visibilité de la propriété est définie par l'opération de l'accesseur get si elle existe, dans le cas contraire par celle de l'accesseur set.
  - Lorsqu'un attribut devient une propriété, un attribut de mise en oeuvre est automatiquement créé pour stocker la valeur de la propriété. L'attribut de réalisation n'est pas persistant et a une visibilité private. Il a le stéréotype <<PropertyImplementation>> et porte le même nom que la propriété, mais commence par une minuscule. Si le nom de la propriété commence déjà par un caractère minuscule, son premier caractère sera converti en majuscule.
  - L'attribut de réalisation peut être retiré pour les propriétés qui n'en n'ont pas l'usage (propriétés calculées, par exemple
  - Si l'attribut étendu booléen Extern est défini à true, aucune opération ne doit être liée à cette propriété
  - Lorsqu'une déclaration de propriété inclut un modificateur Extern, la propriété est dite propriété externe. Une déclaration de propriété externe ne fournit aucune réalisation concrète, chacune de ses déclarations d'accesseur consiste alors en un point-virgule

Propriétés d'événement, d'indexeur et de propriété

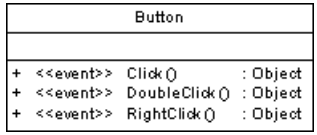
Les feuilles de propriétés d'événement, d'indexeur et de propriété contiennent tous les onglets d'une feuille de propriétés d'attribut standard, avec en plus un onglet C#, dont les propriétés sont répertoriées ci-dessous :

Propriété	Description
Unité de compilation	Spécifie l'unité de compilation dans laquelle l'attribut sera stocké. Disponible uniquement si le type parent est un type partiel (alloué à plusieurs unités de compilation).
New	Spécifie le modificateur new pour la déclaration de l'attribut.
Unsafe	Spécifie le modificateur unsafe pour la déclaration de l'attribut.
Abstract	Spécifie le modificateur abstract pour la déclaration de l'attribut.

Propriété	Description
Extern	Spécifie le modificateur extern pour la déclaration de l'attribut.
Override	Spécifie le modificateur override pour la déclaration de l'attribut.
Sealed	Spécifie le modificateur sealed pour la déclaration de l'attribut.
Virtual	Spécifie le modificateur virtual pour la déclaration de l'attribut.

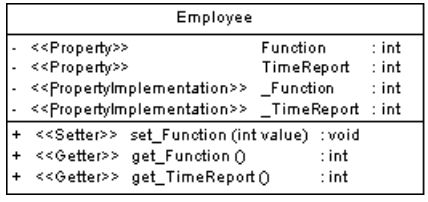
Exemple d'événement

L'exemple suivant montre la classe Button, qui contient trois événements :



Exemple de propriété

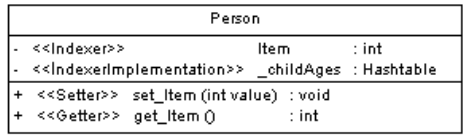
Dans l'exemple suivant, la classe Employee contient deux 2 propriétés. L'opération Setter a été supprimée de la propriété TimeReport :




```
{
 public class Employee
 {
  private int _Function;
  private int _TimeReport;
  // Property Function
  private int Function
  {
   get
   {
     return _Function;
   }
   set
   {
     if (this._Function != value)
       this._Function = value;
    }
  }
  // Property TimeReport
  private int TimeReport
  {
   get
   {
     return _TimeReport;
    }
  }
 }
```

Exemple d'indexeur

Dans l'exemple suivant, la classe Person contient l'attribut indexeur Item. Le paramètres est utilisé pour trier la propriété est String Name :




```
public class Person
{
 private Hashtable _childAges;
 // Indexer Item
 private int this[String name]
 {
  get
  {
   return (int)_ChildAges[name];
  }
  set
  {
    _ChildAges[name] = value;
  }
 }
}
Person someone;
someone ["Alice"] = 3;
someone ["Elvis"] = 5;
```

Héritage et réalisation C# 2.0

PowerAMC modélise les liens d'héritages C# entre types sous la forme de généralisation UML.

Pour plus d'informations sur la création et l'utilisation des généralisations, voir *Généralisations (MOO)* à la page 101.

PowerAMC modélise les liens de réalisation C# entre types et interfaces sous la forme de réalisations UML standard. Pour plus d'informations, voir *Réalisations (MOO)* à la page 109.

Attributs personnalisés C# 2.0

PowerAMC prend en charge entièrement les attributs personnalisés C# 2.0, qui permettent d'ajouter des métadonnées dans votre code. Ces métadonnées peuvent être accessibles par des outils de post-traitement ou au moment de l'exécution pour varier le comportement du système.

Vous pouvez utiliser des attributs personnalisés intégrés, tels que System.Attribute et System.ObsoleteAttribute, et créer également vos propres attributs personnalisés à appliquer à vos types.

Pour plus d'informations sur la modélisation de cette forme de métadonnées dans PowerAMC, voir *Annotations (MOO)* à la page 115.

Génération de fichiers C# 2.0

Vous générez des fichiers source C# 2.0 à partir des classes et interfaces d'un modèle. Un fichier distinct, portant le suffixe .cs, est généré pour chaque classe ou interface que vous sélectionnez dans le modèle, ainsi qu'un fichier journal de génération.

Les variables PowerAMC suivantes sont utilisées dans la génération des fichiers source C# 2.0 :

Variable	Description
CSC	Chemin d'accès complet du compilateur C#. Par exemple, C:\WINDOWS\Microsoft.NET\Framework\v1.0.3705\csc.exe
WSDL	Chemin d'accès complet du générateur de proxy de service Web. Par exemple, C:\Program Files\Microsoft Visual Studio .NET\FrameworkSDK\Bin\wsdl.exe

Pour passer en revue ou éditer ces variables, sélectionnez **Outils > Options générales**, puis cliquez sur la catégorie **Variables**.

1. Sélectionnez **Langage > Générer du code C# 2** pour afficher la boîte de dialogue de génération C# 2.0.
2. Spécifiez le répertoire dans lequel les fichiers doivent être générés, et indiquez si vous souhaitez effectuer une vérification de modèle (voir *Chapitre 9, Vérification d'un MOO* à la page 303).
3. [facultatif] Sélectionnez des cibles supplémentaires pour la génération. Ces cibles sont définies par les extensions que vous pouvez attacher au modèle (voir *Gestion des cibles de génération* à la page 286).
4. [facultatif] Cliquez sur l'onglet **Sélection** et spécifiez les objets à partir desquels vous souhaitez générer. Par défaut, tous les objets sont générés.
5. [facultatif] Cliquez sur l'onglet **Options** et sélectionnez les options de génération appropriées :

Option	Description
Génération des ID d'objet sous forme de balises de documentation	Spécifie si les ID d'objet doivent être générés sous la forme de balises de documentation.

Option	Description
Tri principal des membres de classe par	Spécifie la méthode de tri principale pour les membres de classe : • Visibility • Type
Tri des membres de classe par type	Spécifie l'ordre dans lequel les membres de classe sont triés en fonction de leur type : • Methods – Properties - Fields • Properties – Methods - Fields • Fields – Properties - Methods
Tri des membres de classe par visibilité	Spécifie l'ordre dans lequel les membres de classe sont triés en fonction de leur visibilité : • Public - Private • Private – Public • None
Génération de fichiers de projet Visual Studio 2005	Spécifie si des fichiers de projet à utiliser pour Visual Studio 2005 doivent être générés.
Génération de fichiers d'informations d'assembly	Spécifie si des fichiers d'information doivent être générés pour les assemblies.
Génération du fichier Visual Studio Solution	Spécifie si un fichier de solution à utiliser avec Visual Studio 2005 doit être généré.
Génération de code de service Web dans un fichier .asmx	Spécifie si des services Web doivent être générés dans un fichier .asmx.
Génération d'accesseurs par défaut pour les associations navigables	Spécifie si les accesseurs par défaut doivent être générés pour les associations navigables.

Remarque : Pour plus d'informations sur la modification des options qui s'affichent sur cet onglet et sur l'onglet **Tâches** ainsi que sur l'ajout de vos propres options et tâches, voir *Personnalisation et extension de PowerAMC > Fichiers de définition pour les langage objet, de processus et XML > Catégorie Generation*.

6. [facultatif] Cliquez sur l'onglet **Fichiers générés** et spécifiez les fichiers à générer. Par défaut, tous les fichiers sont générés.

   Pour plus d'informations sur la personnalisation des fichiers qui seront générés, voir *Personnalisation et extension de PowerAMC > Fichiers d'extension > Fichiers générés (Profile)*.

7. [facultatif] Cliquez sur l'onglet **Tâches** et spécifiez les tâches de génération à effectuer :

Tâche	Description
WSDLDotNet : Génération de code de proxy de service Web	Génère la classe de proxy
Compilation des fichiers source	Compile les fichiers source
Ouverture de la solution dans Visual Studio	Dépend de l'option Génération de fichiers de projet Visual Studio 2005. Ouvre le projet généré dans Visual Studio 2005.

8. Cliquez sur **OK** pour lancer la génération.

   Une fois la génération terminée, la boîte de dialogue Fichiers générés s'affiche et répertorie les fichiers générés dans le répertoire spécifié. Sélectionnez un fichier dans liste, puis cliquez sur **Editer** pour l'ouvrir dans votre éditeur associé, ou bien cliquez sur **Fermer** pour quitter la boîte de dialogue.

Reverse engineering de code C# 2.0

Vous pouvez procéder au reverse engineering de fichiers C# dans un MOO.

1. Sélectionnez **Langage > Reverse engineering C#** pour afficher la boîte de dialogue Reverse engineering de C#.
2. Sélectionnez la forme de code sur laquelle vous souhaitez faire porter le reverse engineering. Vous pouvez choisir :
   - Fichiers C# (.cs)
   - Répertoires C#
   - Projets C# (.csproj)
3. Sélectionnez les fichiers, répertoires ou projets sur lesquels effectuer un reverse engineering en cliquant sur le bouton **Ajouter**.

   Remarque : Vous pouvez sélectionner plusieurs fichiers à la fois pour le reverse engineering, en utilisant les touches **Ctrl** et **Maj**. Vous ne pouvez pas sélectionner plusieurs répertoires à la fois.

   Les fichiers ou répertoire sélectionnés sont affichés dans la boîte de dialogue et le répertoire de base est défini sur leur répertoire parent. Vous pouvez changer de répertoire de base en utilisant les boutons situés à droite de la zone.
4. [facultatif] Cliquez sur l'onglet **Options** et définissez les options appropriées. Pour plus d'informations, voir *Onglet Options de la boîte de dialogue Reverse engineering de C#* à la page 497.

5. [facultatif] Cliquez sur l'onglet **Prétraitement** et définissez les symboles de prétraitement appropriés. Pour plus d'informations, voir *Directives de prétraitement pour le reverse engineering C#* à la page 498.
6. Cliquez sur **OK** pour lancer le reverse engineering.

   Une boîte de progression s'affiche. Si le modèle dans lequel vous effectuez un reverse engineering contient déjà des données, une boîte de dialogue de fusion s'affiche.

   Pour plus d'informations sur la fusion des modèles, voir *Guide des fonctionnalités générales > Modélisation avec PowerAMC > Comparaison et fusion de modèles*.

   Les classes sont automatiquement ajoutées dans votre modèle et sont visibles dans le diagramme. Elles figurent également dans l'onglet Reverse de la fenêtre Résultats, située en bas de la fenêtre principale.

Onglet Options de la boîte de dialogue Reverse engineering de C#

Les options suivantes sont disponibles sur cet onglet :

Option	Description
Codage de fichier	Permet de modifier le codage de fichier par défaut pour les fichiers sur lesquels vous souhaitez faire porter le reverse engineering.
Ignorer les corps des opérations	Effectue le reverse engineering sans y inclure le corps du code.
Ignorer les commentaires	Effectue le reverse engineering des classes sans y inclure les commentaires.
Créer des associations à partir des attributs de type classificateurs	Créer des associations entre les classes et/ou les interfaces.
Créer les symboles	Crée un symbole pour chaque objet dans le diagramme, dans le cas contraire, les objets récupérés ne sont visibles que dans l'Explorateur d'objets

Option	Description
Bibliothèques	Spécifie une liste de modèles de bibliothèque à utiliser comme référence lors du reverse engineering. Le modèle de destination du reverse engineering peut contenir des raccourcis vers des objets définis dans une bibliothèque. Si vous spécifiez la bibliothèque ici, le lien entre le raccourci et son objet cible (contenu dans la bibliothèque) sera préservé et la bibliothèque sera ajoutée à la liste des modèles cible dans le modèle de destination du reverse engineering. Vous pouvez faire glisser les bibliothèques dans la liste afin de spécifier une hiérarchie de bibliothèques. PowerAMC tentera de résoudre les raccourcis trouvés dans le modèle de destination du reverse engineering en utilisant tour à tour chacune des bibliothèques spécifiées. Ainsi, si la bibliothèque v1.1 apparaît dans la liste avant la bibliothèque v1.0, PowerAMC tentera d'abord de résoudre les raccourcis vers la bibliothèque v1.1 et analysera uniquement la bibliothèque v1.0 s'il reste des raccourcis non résolus. Vous devez utiliser la liste des modèles cibles pour gérer les bibliothèques liées au modèle de destination du reverse engineering, par exemple, vous pouvez changer la version de bibliothèque (voir *Guide des fonctionnalités générales > Liaison et synchronisation de modèles > Raccourcis et répliques > Utilisation des modèles cible*).
Préserver la structure des fichiers	Crée un artefact lors du reverse engineering afin d'être en mesure de régénérer une structure de fichiers à l'identique.
Marquer les classificateurs comme n'étant pas à générer	Les classificateurs récupérés par reverse engineering (classes et interfaces) ne seront ensuite plus générés à partir du modèle. Pour pouvoir les générer, vous devez cocher la case Générer dans leur feuille de propriétés respective.

Directives de prétraitement pour le reverse engineering C#

Les fichiers C# peuvent contenir du code conditionnel qui doit être géré par des *directives de prétraitement* lors du reverse engineering. Une directive de prétraitement est une commande placée au sein du code source qui demande au compilateur d'effectuer une tâche particulière avant que le reste du code ne soit analysé et compilé. La structure d'une directive de prétraitement est la suivante :

```
#directive symbol
```

est suivi du nom de la directive, et le *symbole* est une constante de compilateur conditionnelle utilisée pour sélectionner des sections particulières de code et pour exclure d'autres sections.

En C#, les symboles sont dépourvus de valeur, ils peuvent être true ou false.

Dans l'exemple suivant, la directive #if est utilisée avec le symbole DEBUG pour générer un message lorsque le symbole DEBUG est vérifié (true), si DEBUG n'est pas vérifié (false), c'est un autre message qui est généré :

```
using System;
public class MyClass
{
   public static void Main()
   {

      #if DEBUG
         Console.WriteLine("DEBUG version");
      #else
         Console.WriteLine("RELEASE version");
      #endif
   }
}
```

Vous pouvez déclarer une liste de symboles pour les directives de prétraitement. Ces symboles sont analysés par des directives de prétraitement : si la condition de la directive est true, l'instruction est conservée ; dans le contraire, l'instruction est supprimée.

Directives de prétraitement C# prises en charge

Les directives suivantes sont prises en charge lors du prétraitement :

Directive	Description
#define	Définit un symbole.
#undefine	Supprime la définition précédente du symbole.
#if	Evalue une condition. Si la condition est vérifiée, l'instruction suivant cette condition est conservée ; dans le cas contraire, elle est ignorée.
#elif	Utilisé avec la directive #if, si le précédent test #if échoue, #elif inclut ou exclut du code source, selon la valeur résultant de sa propre expression ou selon son identifiant.
#endif	Ferme le bloc de code conditionnel #if.
#warning	Affiche un message d'avertissement si la condition est vérifiée.
#error	Affiche un message d'erreur si la condition est vérifiée.

Note : les directives #region, #endregion et #line sont supprimées du code source.

Définition d'un symbole de prétraitement C#

Vous pouvez redéfinir des symboles de prétraitement C# dans l'onglet Prétraitement de la boîte de dialogue de reverse engineering.

Les noms de symbole sont sensibles à la casse et doivent être uniques. Assurez-vous de ne pas saisir des mots réservés tels que true, false, if, do, etc.

La liste des symboles est enregistrée dans le modèle et sera réutilisée lorsque vous synchroniserez votre modèle avec du code existant en utilisant la commande Synchroniser avec les fichiers générés.

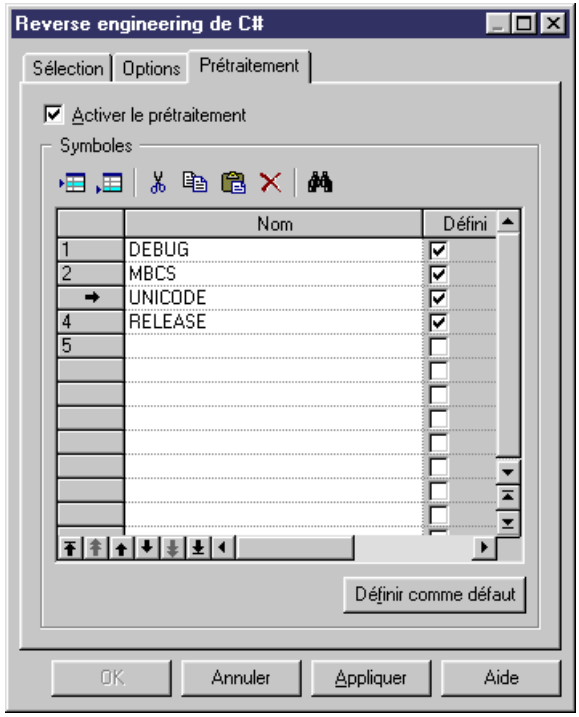
Pour plus d'informations sur la commande Synchroniser avec les fichiers générés, voir *Synchronisation d'un modèle avec des fichiers générés* à la page 290.

Vous pouvez utiliser le bouton Enregistrer comme défaut pour enregistrer la liste des symboles dans le Registre.

1. Sélectionnez **Langage > Reverse engineering C#**.

   La boîte de dialogue Reverse engineering C# s'affiche.
2. Cliquez sur l'onglet Prétraitement pour afficher l'onglet correspondant.
3. Cliquez sur l'outil Ajouter une ligne pour insérer une ligne dans la liste.
4. Saisissez les noms de symbole dans la colonne Nom.

   La case Défini est automatiquement cochée pour chaque symbole afin d'indiquer que le symbole sera pris en compte lors du prétraitement.



5. Cliquez sur Appliquer.

PowerAMC ne prend pas en charge l'espace de noms par défaut dans un projet Visual Studio. Si vous définissez des espaces de noms par défaut dans vos projets, vous devez éviter de faire

porter le reverse engineering sur l'intégralité de la solution. Il est préférable de procéder au reverse engineering de chaque projet séparément.

Chapitre 18 C++

PowerAMC permet la prise en charge de la modélisation de programmes C++, y compris l'ingénierie par va-et-vient.

Modélisation pour C++

Cette section explique comment modéliser les objets C++ dans le modèle orienté objet de PowerAMC.

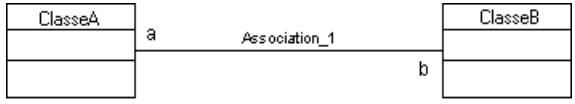
Déclaration d'espace de noms pour les classificateurs

L'attribut étendu *UseNamespace* permet de générer un classificateur au sein d'une déclaration d'espace de noms. Vous devez définir la valeur True pour cet l'attribut étendu.

Gestion des associations bidirectionnelles dans C++

Le problèmes des associations bidirectionnelles est traité par l'utilisation de déclarations anticipées aux lieux de includes.

Considérons une association bidirectionnelle entre ClassA et ClassB.



Le code généré dans A.h serait le suivant :

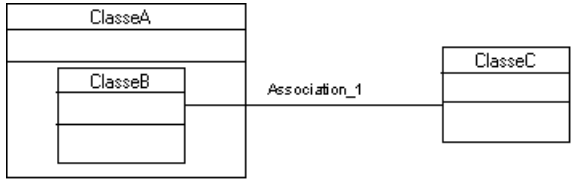
```
#if !defined(__A_h)
#define __A_h

class B;  // forward declaration of class B

class A
{
public:
  B* b;

protected:
private:

};

#endif
```

Le code généré dans B.h serait le suivant :

```
#if !defined(__B_h)
#define __B_h
```

```
class A;  // forward declaration of class A

class B
{
public:
  A* a;

protected:
private:

};

#endif
```

Cette approche ne fonctionne pas si l'une des classes est une classe interne car il n'est pas possible d'utiliser une déclaration anticipée des classes internes dans C++.



Si cette situation se produit, un message d'avertissement s'affiche lors de la génération, et le code correspondant est mis en commentaire.

Fonctionnalités ANSI C++ non prises en charge

PowerAMC ne prend pas en charge les fonctionnalités C++ suivantes :

- Templates
- Enums
- Typedefs
- Inline methods

Génération pour C++

Lorsque vous générez avec C++, les fichiers générés sont générés pour les classes et interfaces.

Un fichier d'en-tête avec le suffixe .h, et un fichier source avec le suffixe .cpp sont générés pour chaque classificateur.

Un fichier journal de la génération est également créé à l'issue de la génération.

1. Sélectionnez **Langage** > **Générer du code C++** pour afficher la boîte de dialogue de génération.
2. Spécifiez un répertoire de destination dans la zone Répertoire.

ou

Cliquez sur le bouton Sélectionner un chemin pour sélectionner un répertoire.

3. Sélectionnez les objets à inclure dans la génération en utilisant les différents onglets en bas de l'onglet Sélection.

Remarque : Par défaut, toutes les classes du modèle, y compris celles contenues dans des packages, sont sélectionnées et affichées dans la liste. Vous pouvez utiliser les outils de sélection situés à droite de la liste pour modifier la sélection. L'outil Inclure les sous-objets permet d'inclure dans votre sélection toutes les classes et interfaces incluses dans des packages.

4. Cliquez sur l'onglet Options.
5. <facultatif> Cochez la case Vérifier le modèle si vous souhaitez vérifier la validité de votre modèle avant la génération.
6. Sélectionnez une valeur pour chaque option requise.
7. Cliquez sur l'onglet Tâches, puis sélectionnez les tâches requises.
8. Cliquez sur OK pour lancer la génération.

Une boîte de progression s'affiche. La fenêtre Liste de résultats affiche les fichiers générés, que vous pouvez éditer. Les résultats sont également affichés dans la fenêtres Résultats, située dans la partie inférieure de la fenêtre principale.

Tous les fichiers C++ sont générés dans le répertoire de destination.

Chapitre 19

Modélisation des correspondances objet/ relationnel (O/R)

PowerAMC prend en charge et peut générer et synchroniser automatiquement les correspondances O/R entre les objets de MOO et de MPD.

Le tableau suivant répertorie les correspondances entre les objets de ces deux types de modèle :

Elément de MOO	Elément de MPD
Domaine	Domaine
Classe (si la case Persistant est cochée et l'option Générer une table sélectionnée)	Table
Colonne d'attribut (si la case Persistant est cochée)	Colonne
Identifiant	Identifiant
Association	Référence ou table
Classe d'association	Table avec deux associations entre les points d'extrémité de la classe d'association
Généralisation	Référence

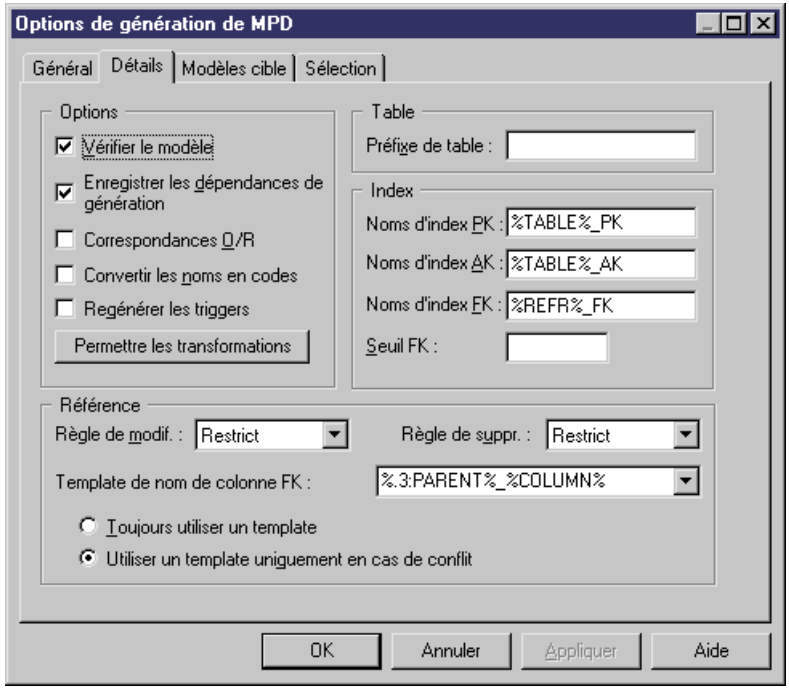
Vous pouvez définir les correspondances entre ces deux types de modèles de l'une des façons suivantes :

- Ingénierie standard – générer des tables et d'autres objets de MPD à partir des classes de MOO
- Rétro-ingénierie – générer des classes et autres objets de MOO à partir des tables de MPD
- Utilisation intermédiaire – définir manuellement les correspondances entre les classes et les tables en utilisant un éditeur de correspondances graphique

Ingénierie standard : Mise en correspondance des classes et des tables

PowerAMC fournit des règles de transformation par défaut pour la génération de modèles physiques de données à partir de modèles orientés objet. Vous pouvez personnaliser ces règles en utilisant les paramètres de persistance et les options de génération.

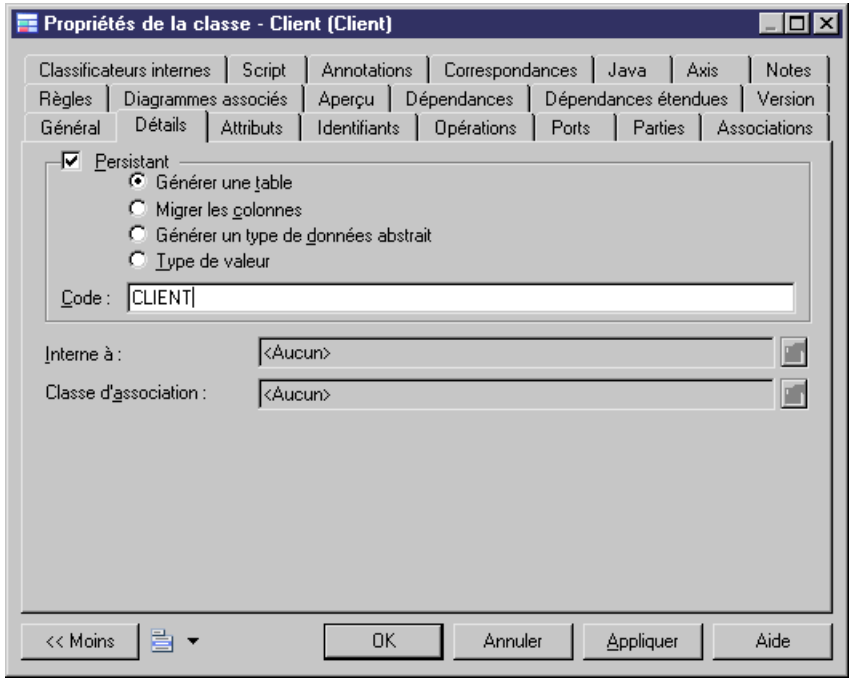
1. Créez votre MOO, remplissez-le de classes persistantes (voir *Transformation de classe d'entité* à la page 509), de liens d'héritage et d'associations etc, pour définir la structure du domaine de votre modèle.
2. Sélectionnez **Outils > Générer un modèle physique de données** pour afficher la boîte de Options de génération de MPD.
3. Sur l'onglet **Général**, spécifiez le type de SGBD, ainsi que le nom et le code du MPD à générer (ou sélectionnez un MPD existant à mettre à jour).
4. Cliquez sur l'onglet **Détails** et cochez la case **Correspondances O/R**. Vous pouvez également spécifier un préfixe de table qui s'appliquera à toutes les tables générées.



5. Cliquez sur l'onglet **Sélection**, puis sélectionnez les objets de MOO que vous souhaitez transformer en objets de MPD.
6. Cliquez sur **OK** pour générer (ou mettre à jour) votre MPD.

Transformation de classe d'entité

Pour transformer une classe en table, vous cochez la case **Persistant** sur l'onglet **Détails** de sa feuille de propriétés, puis définissez le type de persistance dans la zone de groupe **Persistant**.



Les classes persistantes sont des classes avec l'un des types de persistance suivants :

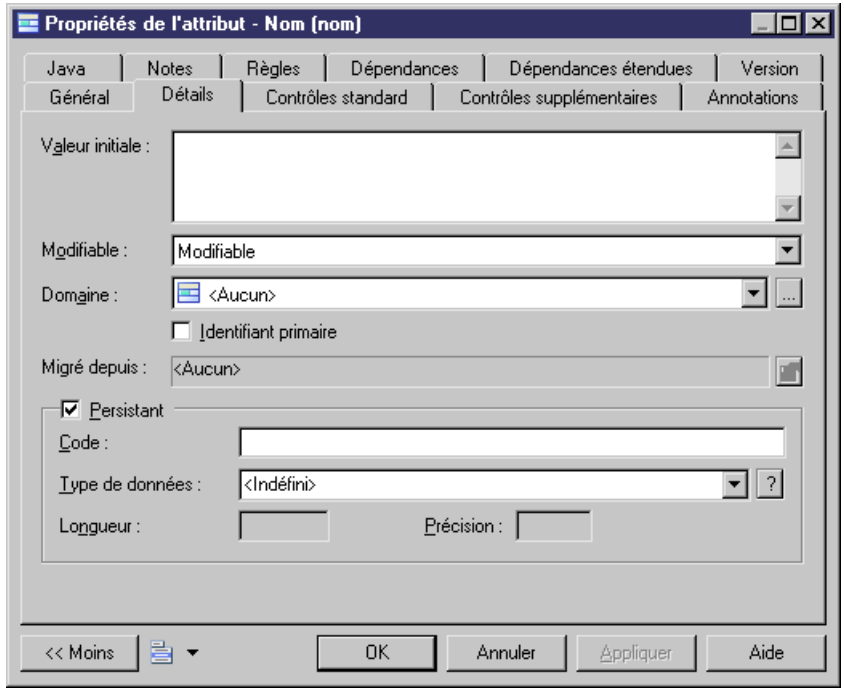
- Générer une table - Ces classes sont appelées classes d'entité, et seront générées sous la forme de tables distinctes. Vous pouvez personnaliser le code des tables générées dans la zone Code de la zone de groupe Persistant. Vous ne pouvez générer qu'une seule table pour chaque classe d'entité de ce type, mais vous pouvez mettre en correspondance une classe d'entité et plusieurs tables (voir *Mise en correspondance de classe d'entité* à la page 520).
- Migrer les colonnes - Ces classes sont appelées classes d'entité, mais aucune table distincte n'est générée pour elles. Ce type de persistance est uniquement utilisé dans la transformation d'héritage, et ses attributs et associations sont migrés dans le parent généré ou la table enfant.
- Générer un type de données abstrait - Ces classes sont générées sous la forme de types de données abstrait, des types de données utilisateur qui peuvent encapsuler une plage de valeurs de données et des fonctions. Cette option n'est pas utilisée lorsque vous définissez les correspondances O/R.

- Type de valeur - Ces classes de type de valeur. Aucune table distincte n'est générée pour la classe ; ses attributs persistants seront transformés en colonnes incorporées dans une ou plusieurs autres tables.

Remarque : Les identifiants de classes persistantes, dans lesquelles le type de génération n'est pas défini à Type de valeur, sont transformés en clés de table. Les identifiants primaires sont transformés en clés primaires ou en partie de clés primaires (voir *Mise en correspondance d'un identifiant primaire* à la page 524). Les attribut persistants contenus dans les identifiant primaires sont transformés en colonnes de clés primaires.

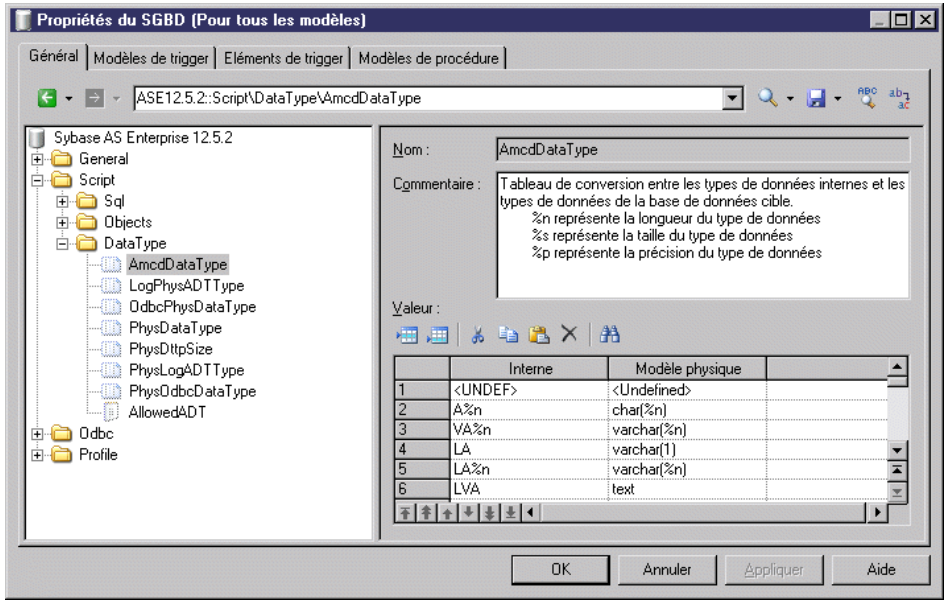
Transformation d'attribut

transformer un attribut en colonne, sélectionnez l'option **Persistant** sur l'onglet **Détails** de sa feuille de propriétés.



Les attributs persistants peuvent avoir des types de données simples ou complexes :

- Type de données simple - par exemple, int, float, String, Date, etc.
  Chaque attribut persistant est transformé en colonne. Son type de données est converti en type de données interne, qui est ensuite mis en correspondance avec le type de données approprié dans le SGBD. Ces transformations sont contrôlées par la table des valeurs dans l'entrée AMCDDataType du dossier Data Type dans la définition du SGBD :



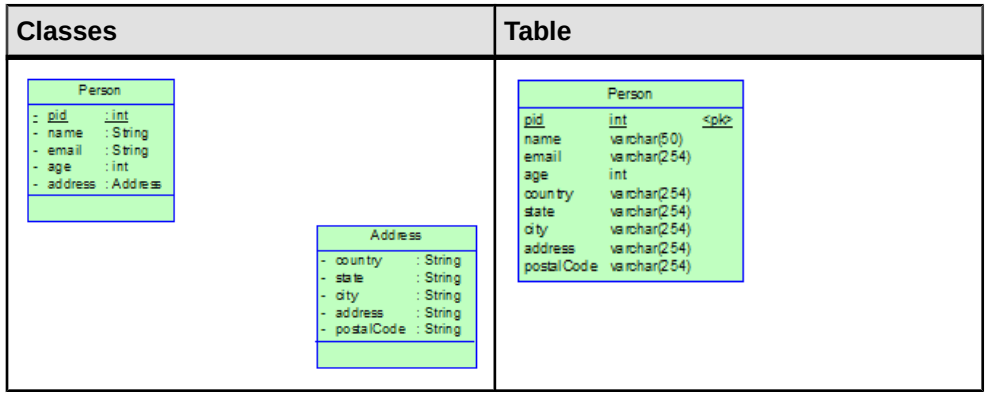
- Type de données complexe - basé sur un classificateur. La transformation dépend des paramètres de persistance du classificateur. Le classificateur est généralement utilisé comme une classe de type de valeur (voir *Transformation de type de valeur* à la page 511).

Vous pouvez personnaliser le code des types de données générés dans la zone Code de la zone de groupe Persistant. Vous pouvez également personnaliser le code des colonnes générées.

Transformation de type de valeur

PowerAMC prend en charge les modèles à persistance fine. Plusieurs classes peuvent être transformées en une table unique.

Envisageons deux classes, *Person* et *Address*, avec la classe *Person* qui contient un attribut *address* dont le type de données est *Address*, les classes peuvent être transformées en une table si le type de transformation de la classe *Address* est définie à Type de valeur. Les colonnes transformées depuis les attributs persistants de la classe *Address* seront incorporées dans la table transformée depuis la classe *Person*.

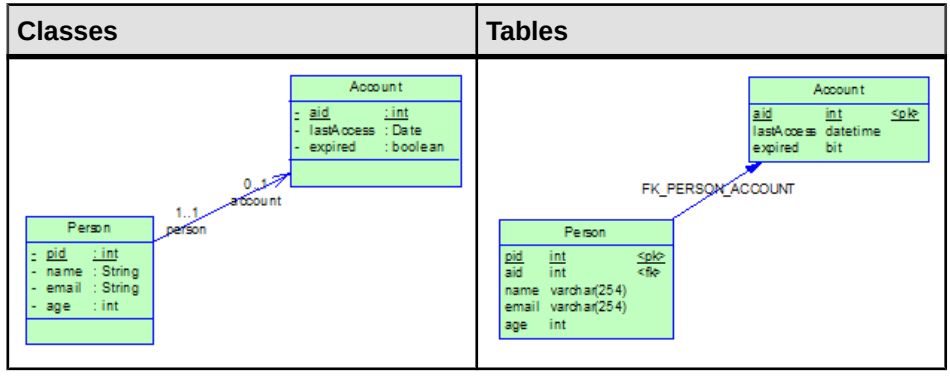


Transformation d'association

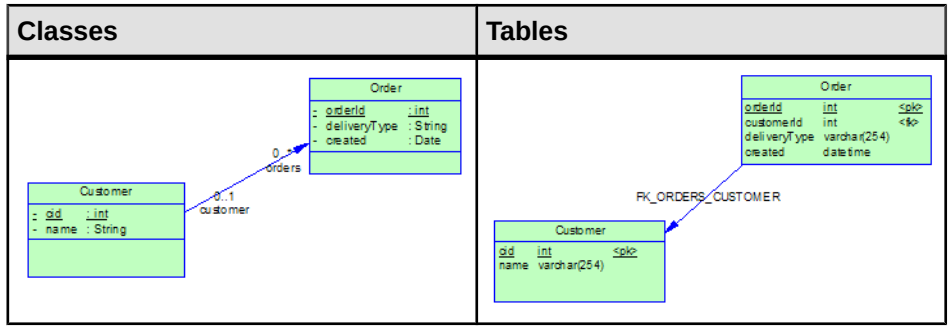
Une association définie entre les classes d'entité sera transformée en clés de référence ou en tables de référence. Les associations ayant des classes de type valeur comme cible ou source sont ignorées.

Les règles de transformation diffèrent en fonction du type d'association :

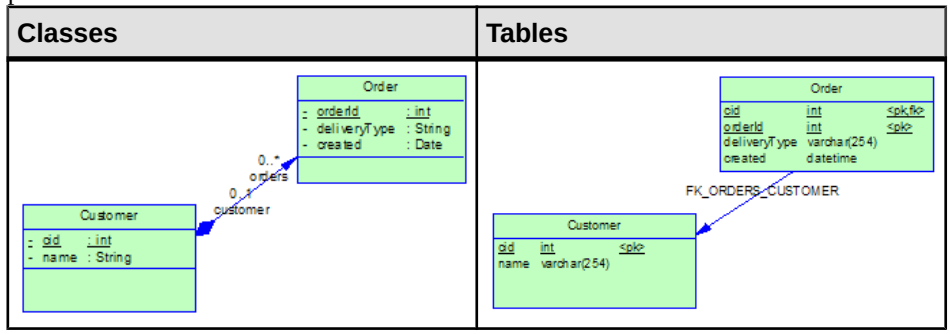
- Association un-un - une clé étrangère est générée avec la même direction que l'association. La clé primaire de la table parent va également migrer vers une table enfant comme sa clé étrangère.



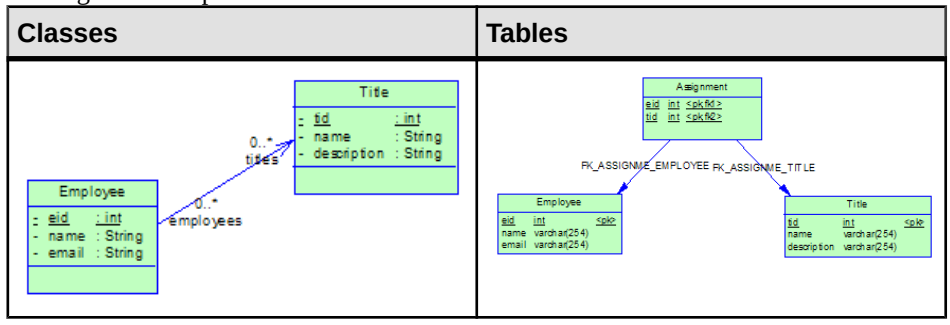
  La clé étrangère générée a la même direction que l'association. Si l'association est bidirectionnelle (si elle peut naviguer dans les deux sens), deux clés étrangères ayant chacune une des directions seront générées car PowerAMC ne sait pas quelle table générée est la table enfant ou la table parent. Vous devez en supprimer une manuellement.
- Association un-plusieurs - une seule clé étrangère sera générée pour chaque association un-plusieurs, quelle que soit sa direction (bidirectionnelle ou unidirectionnelle). La clé de référence navigue de la table générée depuis la classe d'entité du côté plusieurs vers la table générée depuis la classe d'entité du côte un.



- Composition un-plusieurs - PowerAMC peut générer une clé primaire d'une table parent comme faisant partie de la clé primaire de la table enfant si vous définissez l'association comme une composition avec la classe située côté un contenant la classe située côté plusieurs :



- Plusieurs-plusieurs - chaque association plusieurs-plusieurs sera transformée en table intermédiaire et deux clés de référence qui naviguent de la table intermédiaire vers les tables générées depuis les deux classes d'entité.



Dans la plupart des environnements de correspondances O/R, une association unidirectionnelle un-plusieurs (voir *Stratégie de mise en correspondance d'une association un-plusieurs* à la page 528) sera généralement mise en correspondance avec une table intermédiaire et deux références navigant de la table intermédiaire vers les tables mises en correspondance par les deux classes d'entité.

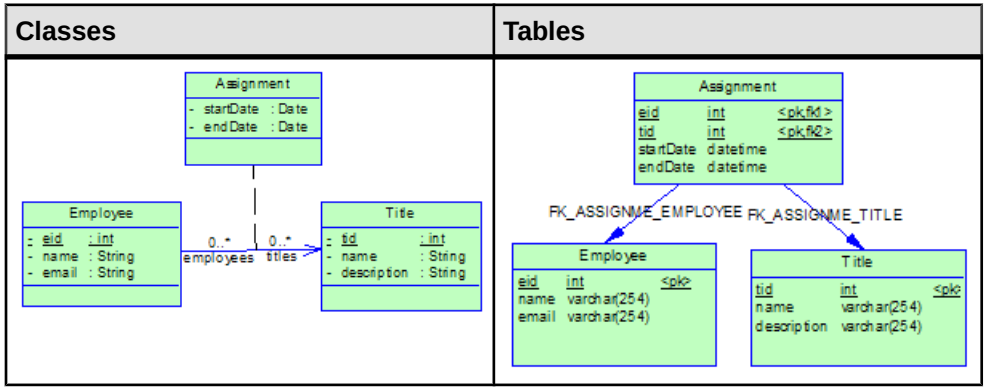
Remarque : La multiplicité minimale des extrémités d'une association peut affecter la propriété Obligatoire des clés de référence générées :

- Dans le cas des associations un-un, si la multiplicité minimale du côté qui est transformé en table parent est supérieure à un, la clé étrangère générée sera obligatoire.
- Dans le cas des associations un-plusieurs, si la multiplicité minimale du côté un est supérieure à un, la clé étrangère générée sera obligatoire.

Transformation d'une classe d'association

En ce qui concerne la correspondance O/R, les classes d'association ne sont pertinentes que dans le cas des associations plusieurs-plusieurs. Les attributs persistants contenus dans la classe d'entité d'association seront transformés en colonnes dans la table intermédiaire.

Dans l'exemple suivant, nous avons défini une classe d'association destinée à contenir informations supplémentaires relatives à l'association :

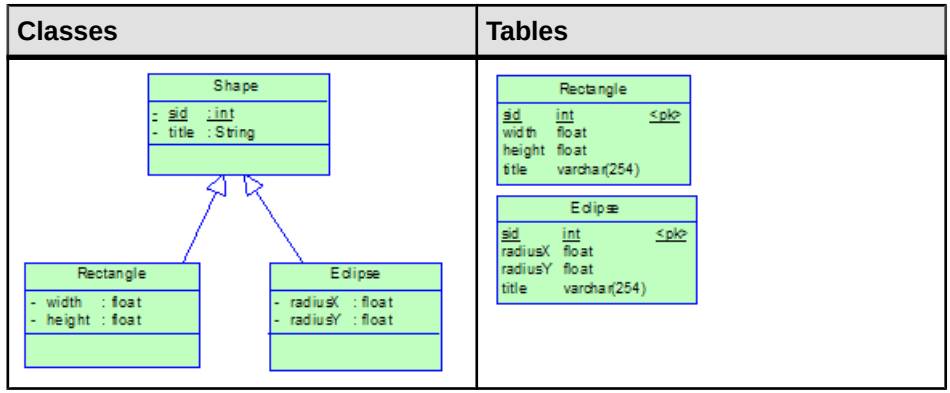


Transformation d'héritage

PowerAMC prend en charge différentes stratégies de mise en correspondance pour la persistance d'héritage. Chaque stratégie a ses avantages et ses inconvénients, et vous devez choisir celle qui est la mieux adaptée à votre cadre de persistance.

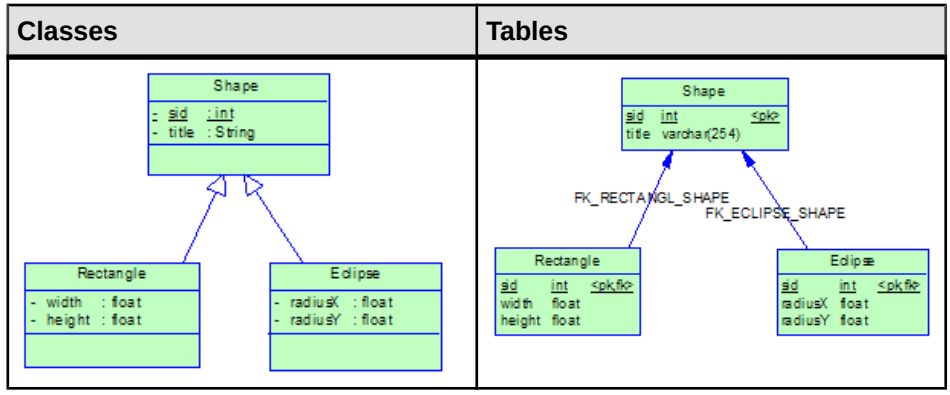
- Une table par hiérarchie de classes. Dans cette stratégie, la totalité de la hiérarchie de classes est mise en correspondance avec une table. La table a une colonne qui sert de "colonne discriminante". La valeur de cette colonne identifie la sous-classe spécifique à laquelle l'instance qui est représentée par la ligne appartient.

  Pour pouvoir appliquer ce genre de stratégie, vous devez définir le type de transformation des classes extrémité à *Générer une table* et le type de transformation des autres classes de la hiérarchie à *Migrer les colonnes*. PowerAMC va uniquement générer les tables pour les classes extrémité. Si vous souhaitez mettre en correspondance des classes avec d'autre classes, vous devez le faire manuellement.



- Stratégie de sous-classe jointe. La classe racine de la hiérarchie de classes est représentée par une table unique. Chaque sous-classe est représentée par une table distincte. Cette table contient les champs qui sont spécifiques à la sous-classe (non hérités de sa classe parent), ainsi que les colonnes qui représentent cette clé primaire. La ou les colonnes de clé primaire de la table de la sous-classe servent de clé étrangère à la clé primaire de la table de la classe parent.

  Pour pouvoir appliquer cette stratégie, vous devez définir le type de transformation de toutes les classes à *Générer une table*. Vous pouvez également définir un discriminant pour ce type de stratégie, bien que cela ne soit pas obligatoire.

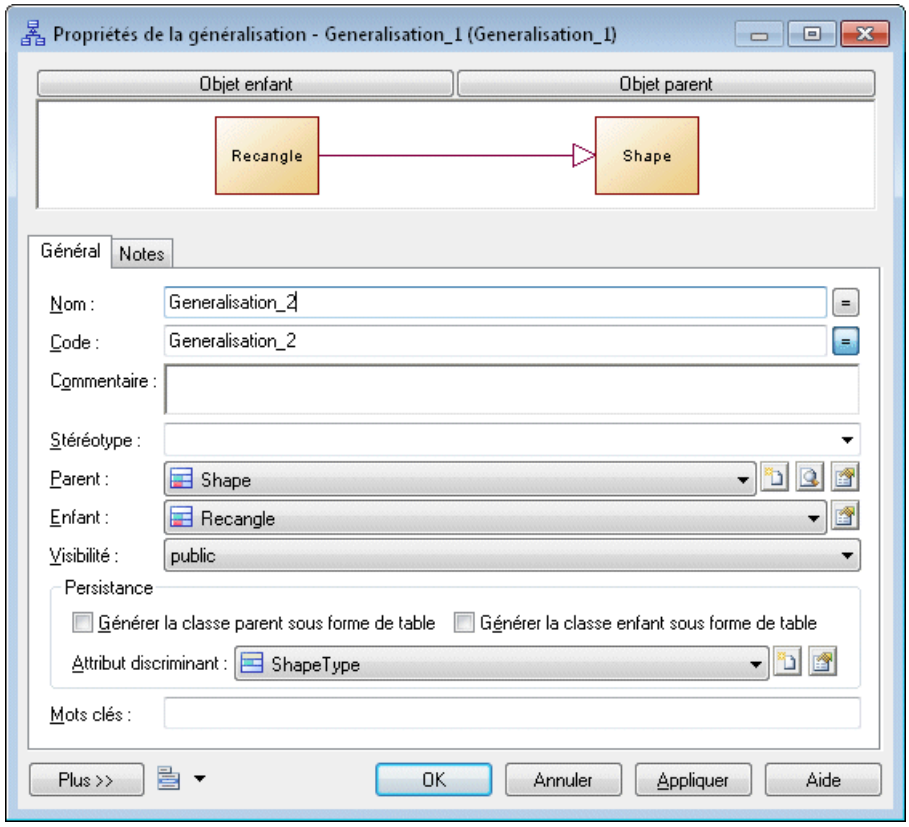


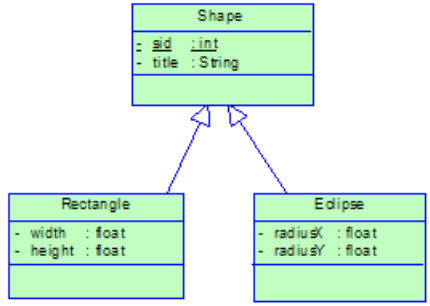
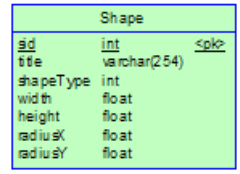
- Une table par classe. Chaque classe est mise en correspondance avec une table distincte. Toutes les propriétés de la classe, y compris les propriétés héritées, sont mises en correspondance avec les colonnes de la table pour la classe.

  Pour pouvoir appliquer ce genre de stratégie, vous devez définir le type de transformation de la classe racine à *Générer une table* et le type de transformation des autres classes de la hiérarchie à *Migrer les colonnes*.

  Pour chaque hiérarchie de classes, un discriminant est nécessaire pour faire la distinction entre différentes instances de classes. Vous devez sélectionner l'un des attributs de la classe racine dans la liste *Attribut discriminant* située dans la feuille de propriétés d'un lien d'héritage enfant de la classe racine. L'attribut sera transformé en colonne discriminante.

Dans l'exemple suivant, on définit un attribut supplémentaire *shapeType* dans *Shape* et on le sélectionne comme attribut discriminant :



Classes	Tables
	

- Stratégie mixte - Vous pouvez appliquer plusieurs stratégies dans la même hiérarchie d'héritage.

  La transformation des classes d'entité en utilisant le type de transformation *Générer une table* ne changera pas, en revanche la transformation de celles définies à *Migrer les*

colonnes sera légèrement différente. Si des classes d'entité définies à *Migrer les colonnes* ont à la fois leur classes parent et enfant ayant un type de transformation *Générer une table*, les colonnes transformées depuis leurs attributs persistants seront migrées dans des tables transformées depuis des sous-classes. La migration vers les sous-classes a une priorité plus élevée.

Rétro-ingénierie : Mise en correspondance des tables et des classes

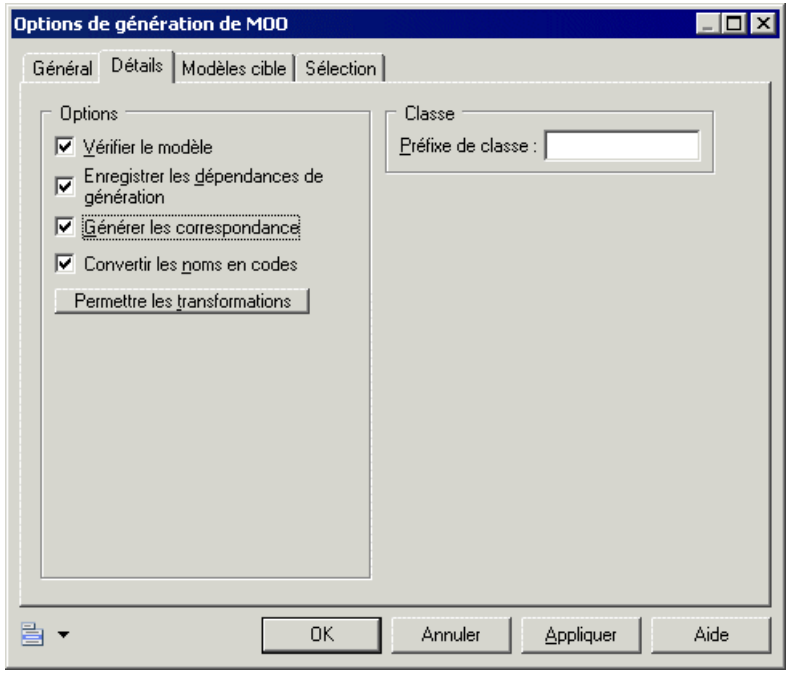
PowerAMC fournit des règles de transformation par défaut pour la génération de modèles orientés objet à partir de modèles physiques de données. Vous pouvez améliorer ces règles manuellement en utilisant l'Editeur de correspondances.

Lorsque vous générez un modèle orienté objet à partir d'un modèle de données :

- Chaque table sélectionnée est transformée en une classe d'entité persistante. En outre :
  - Ses colonnes sont transformées en attributs persistants.
  - Ses clés sont transformées en identifiants.
  - Ses clés primaires sont transformées en identifiants primaires.
- Les clés de référence ont, par défaut, la cardinalité 0..* et seront transformées en associations bidirectionnelles plusieurs-plusieurs. Pour générer une association un-un, vous devez spécifier comme cardinalité maximale 1 (cardinalité 0..1 ou 1..1). Si la clé de référence est obligatoire, la multiplicité minimale de l'autre côté de l'association générée sera 1.

  Vous ne pouvez pas générer des liens d'héritage à partir de clés de référence et de tables.

1. Créez votre MPD (par exemple, en procédant au reverse-engineering d'une base de données) et remplissez-le des tables et références appropriées.
2. Sélectionnez **Outils > Générer un modèle orienté objet** pour afficher la boîte de dialogue Options de génération de MOO.
3. Sur l'onglet Général, spécifiez le type de langage objet ainsi que le nom et le code du MOO à générer (ou bien sélectionnez un MOO à mettre à jour).
4. Cliquez sur l'onglet Détails, puis cochez la case Correspondances O/R. Vous pouvez également spécifier un préfixe de classe qui s'appliquera à toutes les classes générées.



5. Cliquez sur l'onglet Sélection, puis sélectionnez les tables que vous souhaitez transformer en classes.
6. Cliquez sur OK pour générer (ou mettre à jour) votre MOO.

Utilisation intermédiaire : Mise en correspondance manuelle des classes et des tables

Si vous disposez déjà d'un MOO et d'un MPD, vous pouvez définir les correspondances entre les deux manuellement en utilisant l'Editeur de correspondances.

Il n'y a pas de contrainte sur la façon dont vous mettez en correspondance vos classes persistantes. Il existe toutefois des stratégies de correspondance bien définies, qui sont prises en charge par la plupart des technologies de correspondance O/R. Vous devez suivre ces stratégies si vous souhaitez construire correctement des modèles de correspondance O/R. Toutefois, il existe des différences mineures entre elles qui seront traitées ultérieurement.

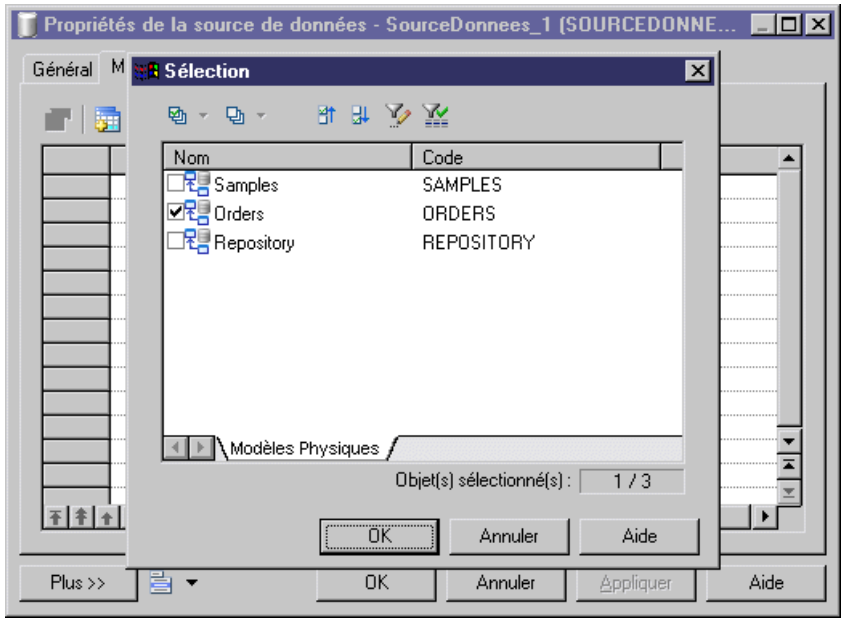
Remarque : lorsque vos modèles de correspondance O/R sont liés à une technologie particulière, par exemple lorsque vous modélisez pour la persistance EJB 3.0, certaines contraintes doivent être respectées et des vérifications de modèles sont disponibles pour vérifier la syntaxe des correspondances que vous avez définies.

Pour définir une correspondance de base, vous devez définir une source de données pour votre MOO. Vous pouvez ensuite définir la correspondance en utilisant l'onglet Correspondance de

l'objet de MOO que vous souhaitez mettre en correspondance avec un objet de MPD ou en utilisant l'Editeur de correspondances.

1. Dans le MOO, sélectionnez **Modèle > Sources de données** pour afficher la liste correspondante.
2. Cliquez sur l'outil Ajouter une ligne pour créer une source de données.

   Vous pouvez plusieurs sources de données dans le modèle.
3. Double-cliquez sur la source de données dans la liste pour afficher sa feuille de propriétés.
4. Sur l'onglet Modèles, cliquez sur l'outil Ajouter des modèles pour sélectionner un ou plusieurs MPD parmi les MPD ouverts afin de les choisir comme modèles source pour la source de données.



5. Définissez les correspondances en utilisant l'onglet Correspondances ou l'Editeur de correspondances.

   L'Editeur de correspondances est plus pratique à utiliser car vous pouvez définir toutes les correspondances à un même endroit par glisser-déposer. Toutefois, il est simple de comprendre les correspondances entre les éléments de MOO et les élément de MPD en utilisant l'onglet Correspondances dans la feuille de propriétés d'un objet. Nous traiterons donc de l'utilisation de l'onglet Correspondances pour définir des correspondances dans les sections suivantes.

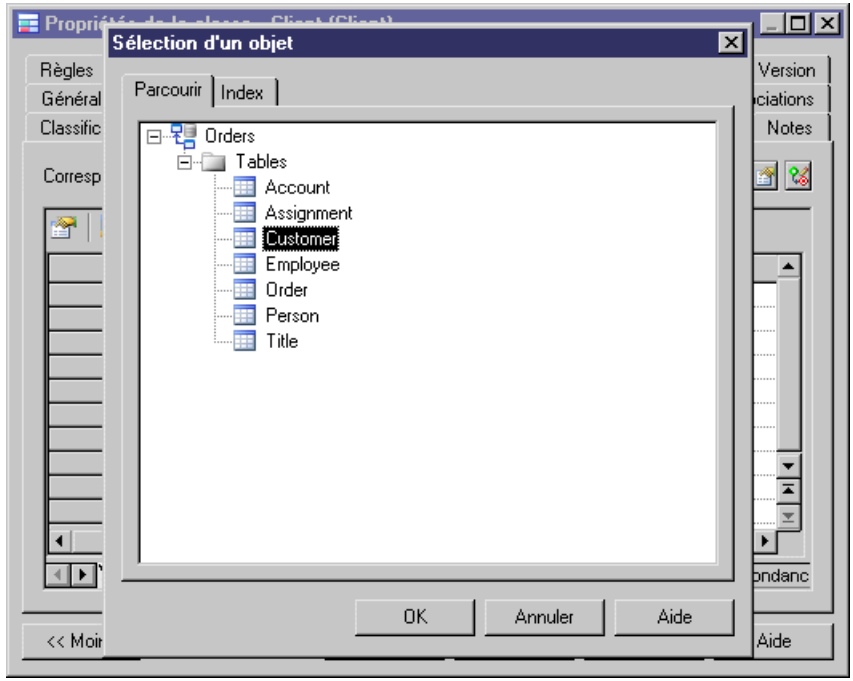
   Lorsque vous serez familiarisé avec les concepts de correspondances O/R, vous pourrez utiliser l'Editeur de correspondances.

Mise en correspondance de classe d'entité

Pour pouvoir définir une correspondances pour les classes d'entité, vous devez effectuer les opérations suivantes :

- Affichez l'onglet Correspondances de la feuille de propriétés d'une classe
- Cliquez sur le bouton Créer une correspondances pour créer une nouvelle correspondances pour la classe
- Sur la boîte de dialogue Sélection d'un objet, ajoutez un objet de modèle de données comme source de la correspondance

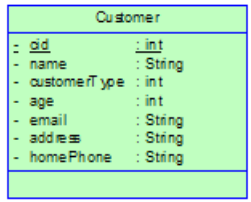
Vous pouvez également cliquer sur l'outil Ajouter des objets dans le sous-onglet Sources situé sur l'onglet Correspondances de la feuille de propriétés de classe après avoir créé la correspondance de classe.



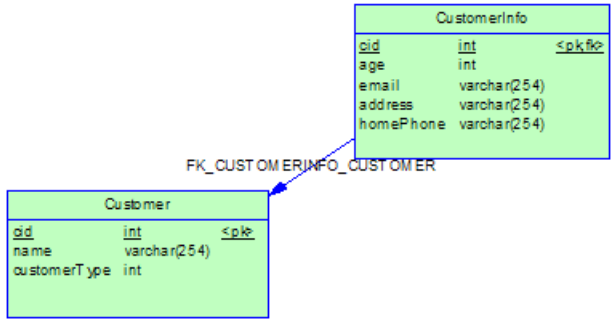
Vous pouvez ajouter des tables, vues et références comme sources de correspondance. Il existe des contraintes concernant l'utilisation de vues comme sources de correspondances, car certaines vues ne peuvent pas être mises à jour. Lorsque vous ajoutez des références comme sources de correspondances, les tables situées aux deux extrémités sont également ajoutées.

Vous pouvez ajouter plusieurs tables comme sources de correspondance. En règle générales, la première table que vous ajoutez est appelée table principale. Les autres tables sont appelées tables secondaires. Chaque table secondaire doit avoir des clés de références qui pointent vers une table primaire, laquelle est jointe par le biais de sa clé primaire.

Considérons la classe *Customer* suivante :

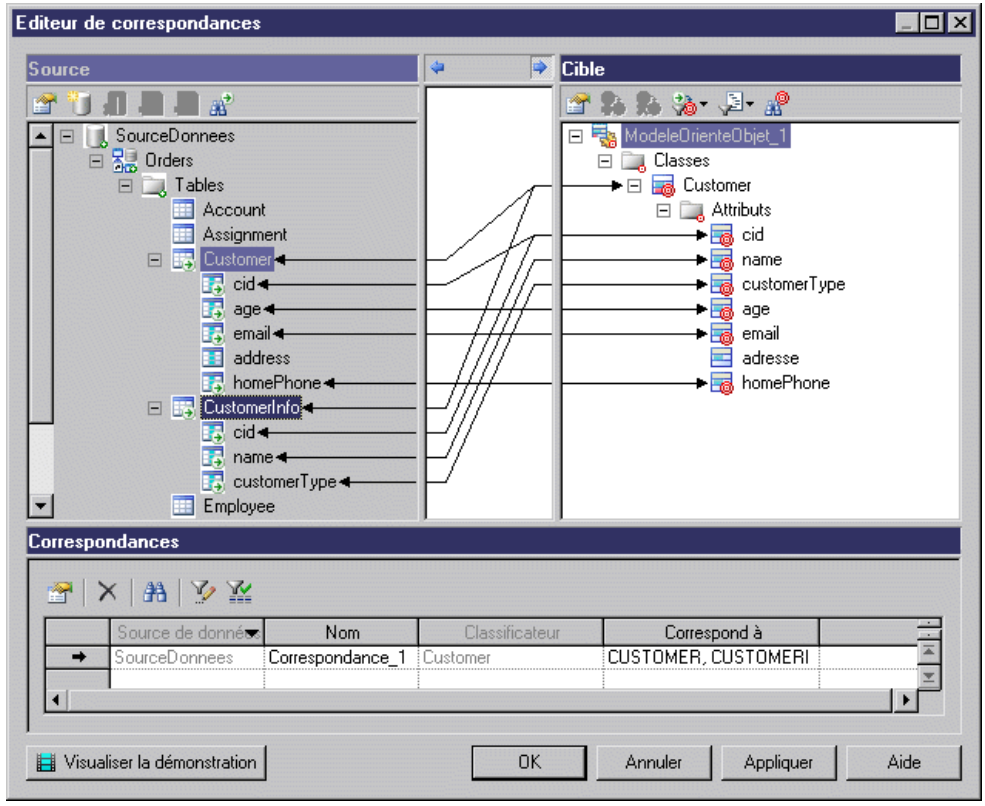


Elle peut être mise en correspondance avec deux tables :



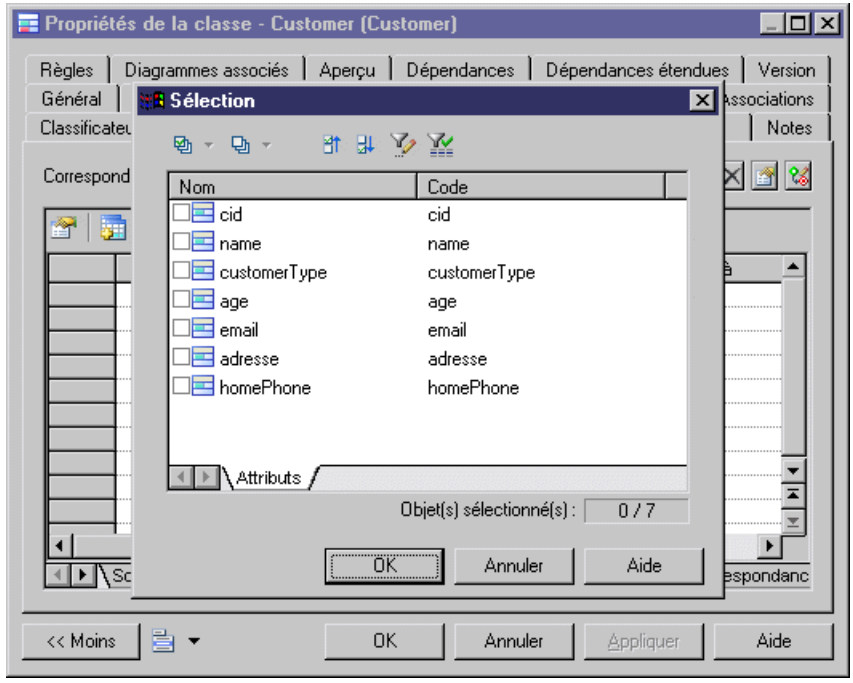
La table *Customer* est la table principale. La table *CutomerInfo* est la table secondaire et a une clé de référence qui pointe sur la table principale, qui est jointe par sa clé primaire.

En utilisant Editeur de correspondances, il vous suffit de faire glisser les deux tables et de les déposer sur la classe *Customer* afin de définir des correspondances de classe.

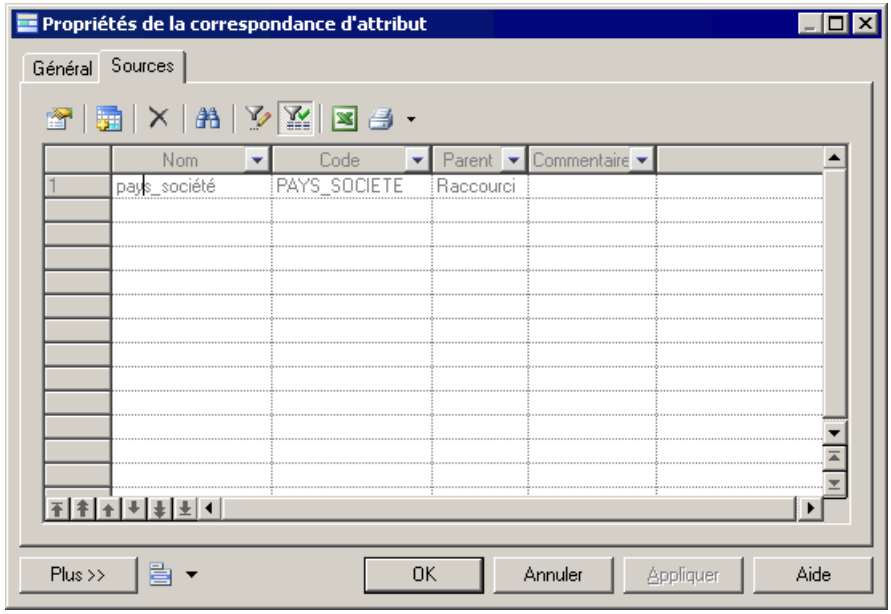


Mise en correspondance d'un attribut

Une fois que vous avez défini la correspondance de classe, vous pouvez définir les correspondances d'attributs pour la classe dans le sous-onglet Correspondances des attributs de l'onglet Correspondances. PowerAMC va générer certaines correspondances d'attribut en faisant correspondre leurs noms avec les noms de colonnes. Cliquez sur l'outil Ajouter des correspondances, puis sélectionnez les attributs que vous souhaitez mettre en correspondance dans la liste.



Pour chaque attribut, vous pouvez sélectionner la colonne à laquelle il correspond dans la liste, en utilisant la colonne Correspond à. En règle générale, il vous suffit de mettre en correspondance chaque attribut et une colonne. Toutefois, vous pouvez être amené à mettre l'attribut en correspondance avec plusieurs colonnes si vous définissez des correspondances d'attribut pour une classe de type valeur, par exemple. Dans ce cas, vous pouvez afficher la feuille de propriétés de correspondance d'attribut et sélectionner l'onglet Sources pour ajouter plusieurs colonnes.



Vous pouvez également mettre en correspondance l'attribut et une expression de formule en la définissant dans la zone Correspond à sur l'onglet Général. Vous pouvez construire la formule en utilisant l'éditeur SQL.

Lorsqu'un attribut a pour type une classe de type valeur, il n'est pas nécessaire de lui définir des correspondances d'attribut. Vous devez plutôt définir une correspondance pour la classe de type valeur.

Mise en correspondance d'un identifiant primaire

Les colonnes de clé primaire doivent être mises en correspondance avec les attributs persistants. Tout comme les clés primaires pour les tables, vous devez définir ces attributs persistants comme identifiants primaires des classes d'entité. Les clés primaires mises en correspondance doivent être des clés primaires de tables principales.

Il existe trois types de mise en correspondance d'identifiant primaire :

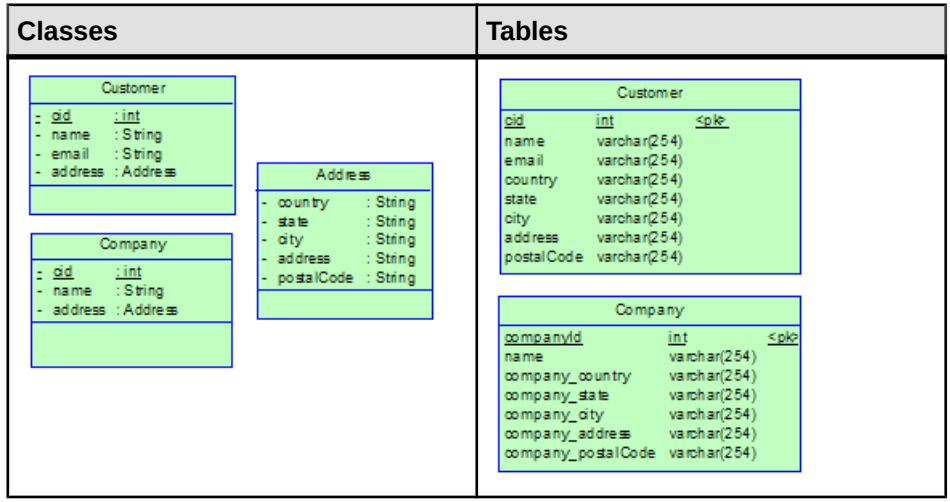
- *Correspondance d'identifiant primaire simple* - la clé primaire est associée à une seule colonne et l'identifiant primaire mis en correspondance a un attribut persistant mis en correspondance avec la colonne.
- *Correspondance d'identifiant primaire composite* - la clé primaire est associée à plusieurs colonnes et l'identifiant primaire mis en correspondance a le même nombre d'attributs persistants mis en correspondance avec les colonnes.

  La ou les colonnes de clé primaire peuvent être mises en correspondance avec des associations (voir *Transformation d'association* à la page 512). Elles sont migrées depuis les clés primaires vers d'autres tables.

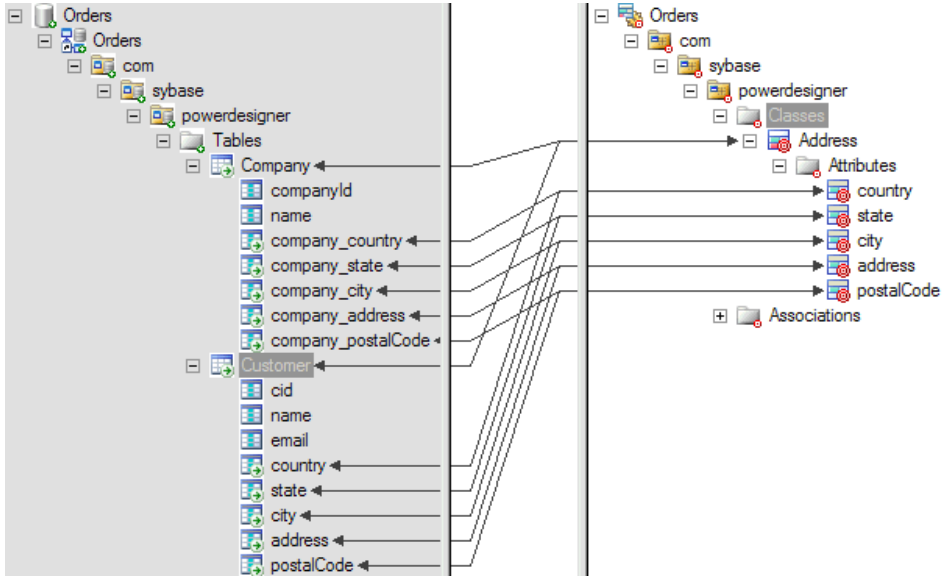
- *Correspondance d'identifiant primaire de composant* - plusieurs attributs persistants sont encapsulés dans une classe Type de valeur, et l'identifiant primaire mis en correspondance contient un attribut dont le type est une classe Type de valeur.

  Les attributs des classes Type de valeur sont mis en correspondance avec les colonnes, qui sont incorporées dans les tables principales mises en correspondance par d'autres classes d'entité. Vous devez donc ajouter les tables primaires des classes conteneur comme sources de correspondance pour les classes Type de valeur. Si la classe Type de valeur est utilisée dans plusieurs classes d'entité, vous devez mettre en correspondance chacun de ses attributs persistants avec plusieurs colonnes de tables de ces classes.

  Par exemple, la classe Type de valeur *Address* utilisée comme type d'attribut pour deux classes, *Product* and et *Customer*. L'attribut de la classe Type de valeur *Address* peut être mis en correspondance avec les colonnes de deux tables, *Company* et *Customer* :



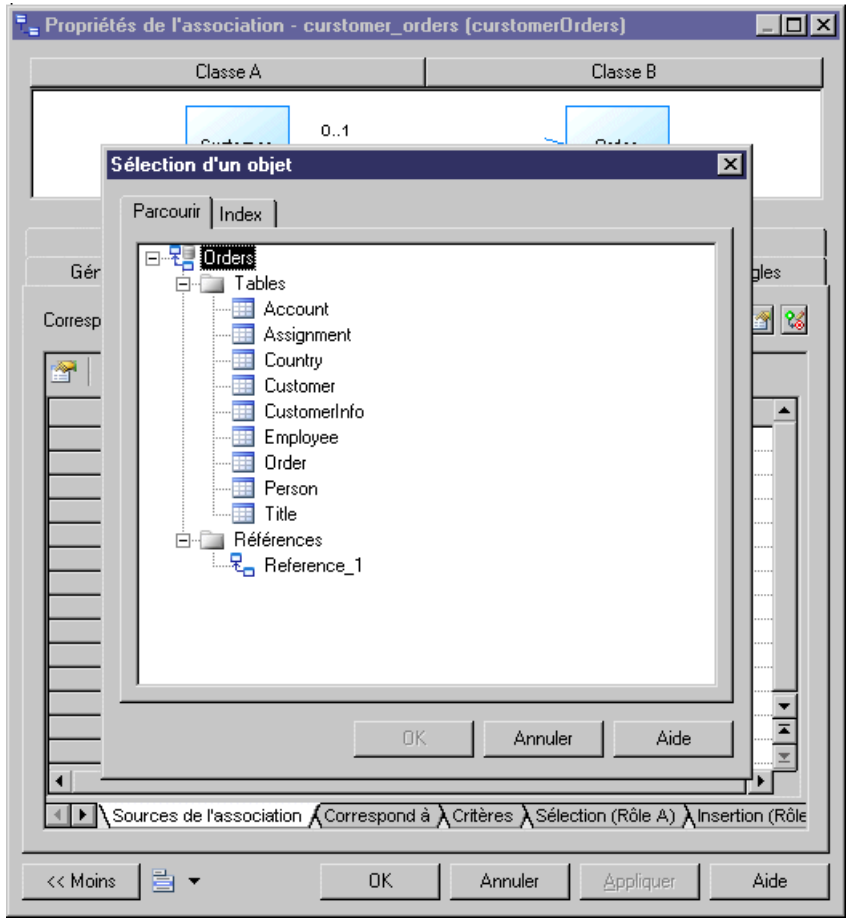
  La mise en correspondances est plus simple à visualiser dans l'Editeur de correspondances.



La correspondance d'identifiant primaire est obligatoire pour les classes d'entité.

Mise en correspondance d'une association

Vous pouvez définir une correspondances d'associations sur l'onglet Correspondances de la feuille de propriétés de l'association, puis sélectionner l'outil Ajouter des objets pour ajouter des source de correspondances.



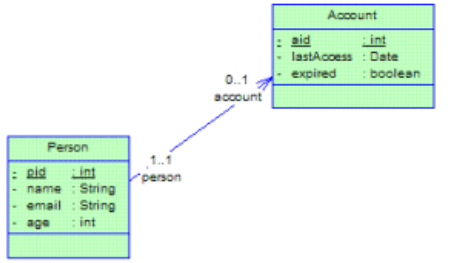
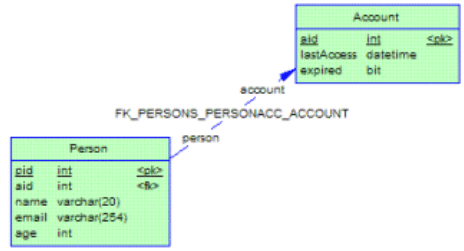
Les associations définies entre les classes d'entité peuvent être mises en correspondance avec des clés de référence ou des tables. Pour pouvoir définir une correspondance d'association, vous devez ajouter les clés de référence ou les tables comme sources de correspondance. Lorsque vous ajoutez des clés de référence, les tables situées à leurs extrémités sont également ajoutées.

Les associations peuvent être classées en un-un, un-plusieurs et plusieurs-plusieurs en fonction des multiplicités de leurs extrémités, elles peuvent être également considérées comme unidirectionnelle et bidirectionnelle en fonction de la navigabilité à leurs deux extrémités. Ces concepts seront abordés en détail dans les sections suivantes.

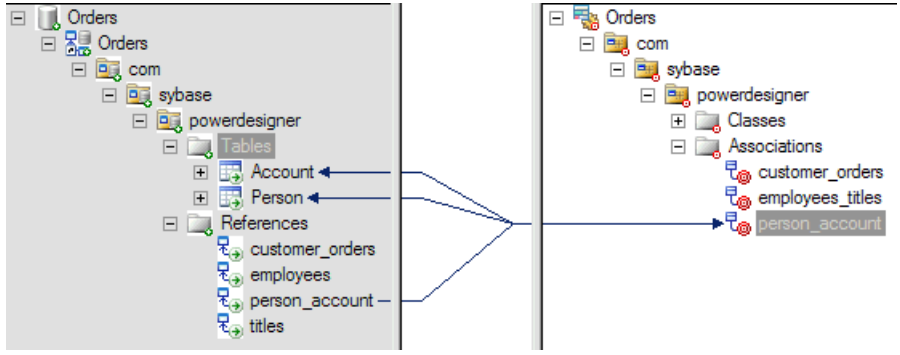
Stratégie de mise en correspondance d'une association un-un

Vous pouvez mettre en correspondance chaque association un-un unidirectionnelle avec une clé de référence. La clé étrangère doit avoir la même direction que l'association.

Dans l'exemples suivant, deux classes d'entité, *Person* et *Account* sont liées par une association un-un. L'association est unidirectionnelle et navigue de la classe d'entité *Person* vers la classe d'entité *Account*.

Classes	Tables
	

L'association et la clé de référence sont liées dans l'Editeur de correspondances.

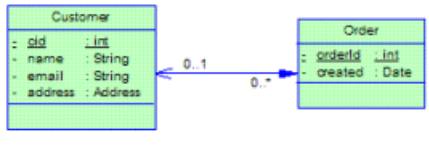
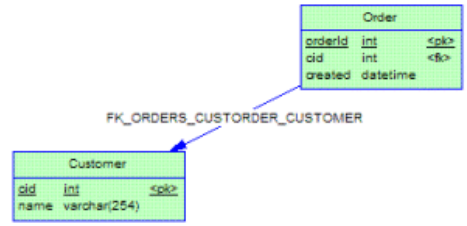


Pour une association un-un bidirectionnelle, vous pouvez également vous contenter de la mettre en correspondance avec une clé de référence. Mais la clé de référence peut naviguer dans les deux directions.

Stratégie de mise en correspondance d'une association un-plusieurs

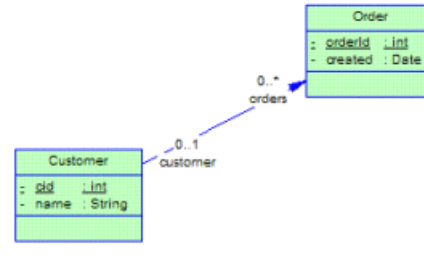
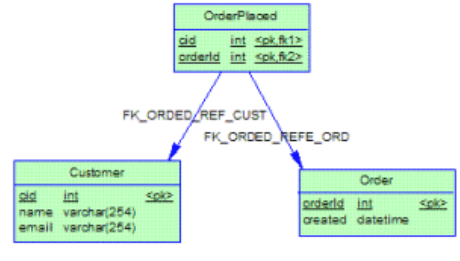
Chaque association un-plusieurs unidirectionnelle est mise en correspondance avec une référence qui a la même direction que l'association.

Dans l'exemple suivant, l'association un-plusieurs unidirectionnelle définie entre les classes *Customer* et *Order* est mise en correspondance avec la clé de référence :

Classes	Tables
	

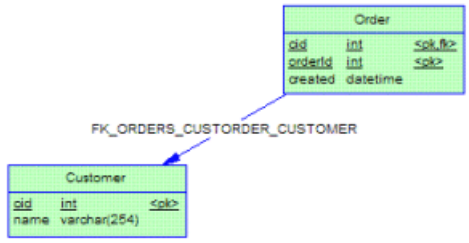
Chaque association un-plusieurs unidirectionnelle doit être mise en correspondance avec une table intermédiaire et deux références pointant vers les tables mises en correspondances avec les classes d'entité situées aux deux extrémités de l'association.

Dans l'exemple suivant, l'association définie entre *Customer* et *Order* est une association unidirectionnelle un-plusieurs mise en correspondance avec une table intermédiaire et des clés de référence :

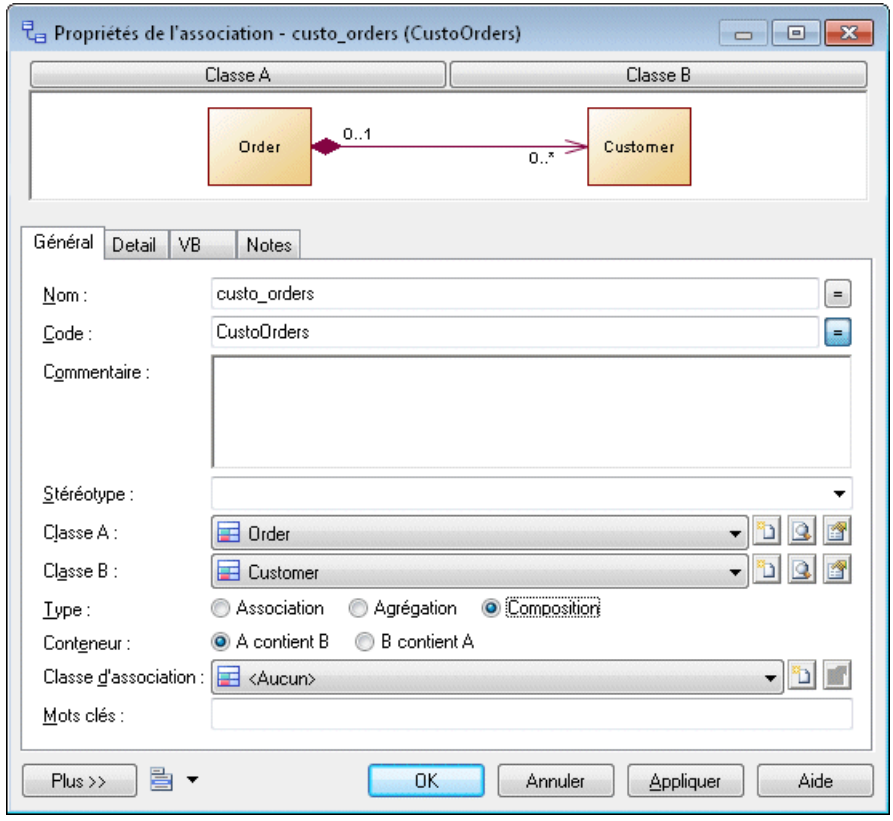
Classes	Tables
	

Vous pouvez mettre en correspondance une association bidirectionnelle un-plusieurs comme une association plusieurs-un unidirectionnelle. La référence ne peut naviguer que depuis la table principale de la classe du côté plusieurs vers la table primaire de la classe du côté un.

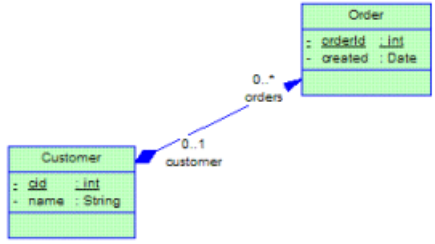
Parfois, il est souhaitable de faire en sorte que la clé primaire de la table parent fasse partie de la clé primaire de la table enfant et de référencer la jointure de clé sur la ou les colonnes migrées. Par exemple, on peut mettre en correspondance *Customer*, *Order* et une association un-plusieurs bidirectionnelle avec des tables et une clé de référence comme suit :



Pour pouvoir définir un tel type de correspondance d'association, vous devez définir l'association comme une composition avec la classe située du côté un contenant la classe située du côté plusieurs.



L'association se présente comme suit :

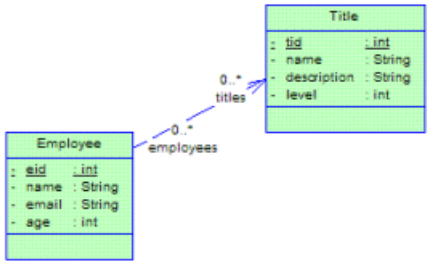
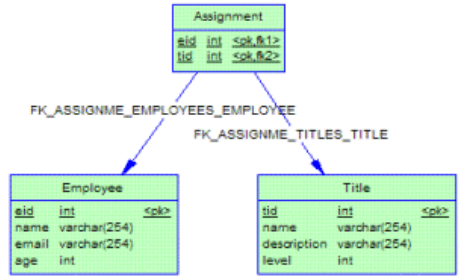


Ajoutez ensuite la référence comme source de correspondance. Vous pouvez définir de la même manière une correspondance d'association pour une association un-plusieurs bidirectionnelle.

Stratégie de mise en correspondance d'une association plusieurs-plusieurs

Chaque association plusieurs-plusieurs doit être mise en correspondance avec une table intermédiaire et deux clés de référence qui pointent vers des tables mises en correspondance avec les classes d'entité situées aux deux extrémités de l'association.

Dans l'exemple suivant, l'association plusieurs-plusieurs définie entre les classes *Employee* et *Title* est mise en correspondance avec une table intermédiaire et des références :

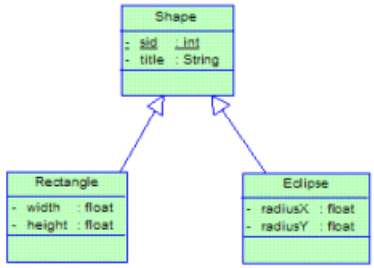
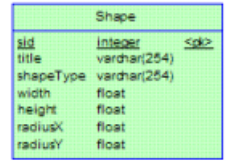
Classes	Tables
	

Définition de la correspondance d'un héritage

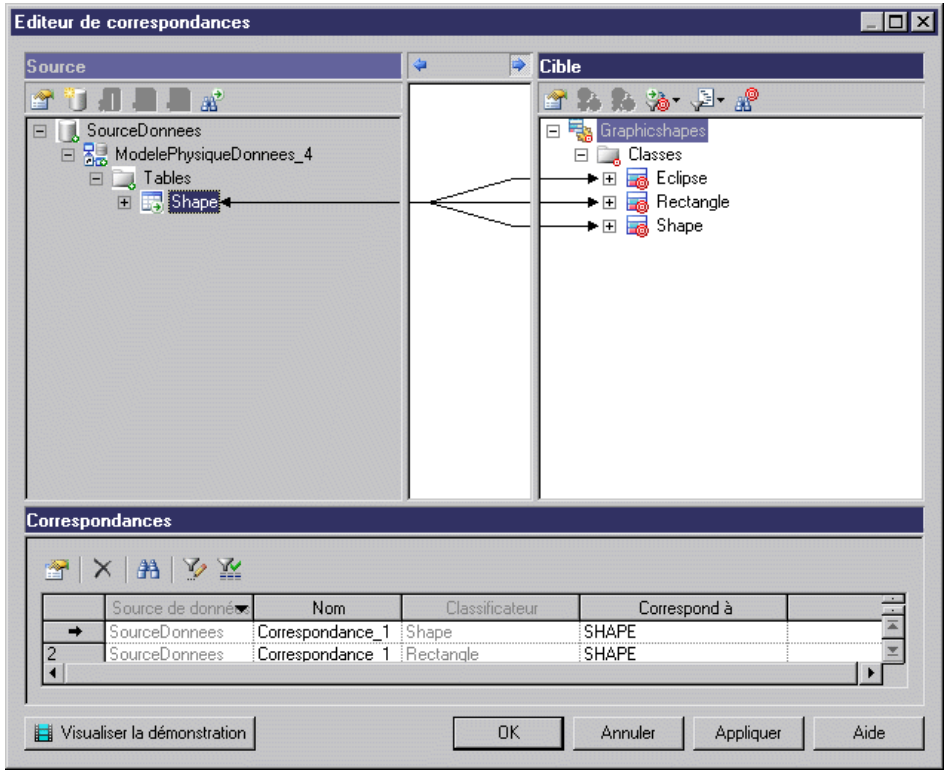
L'héritage peut être mis en correspondance en utilisant une stratégie de correspondance "une table par classe", une stratégie de sous-classe jointe ou une stratégie "une table par hiérarchie de classes". Vous pouvez utiliser l'une ou l'autre de ces stratégies, ou même les mélanger. Vous devez définir les identifiants primaires sur la classe d'entité qui constitue la racine de la hiérarchie d'entités.

Stratégie de mise en correspondance d'héritage "une table par hiérarchie de classes"

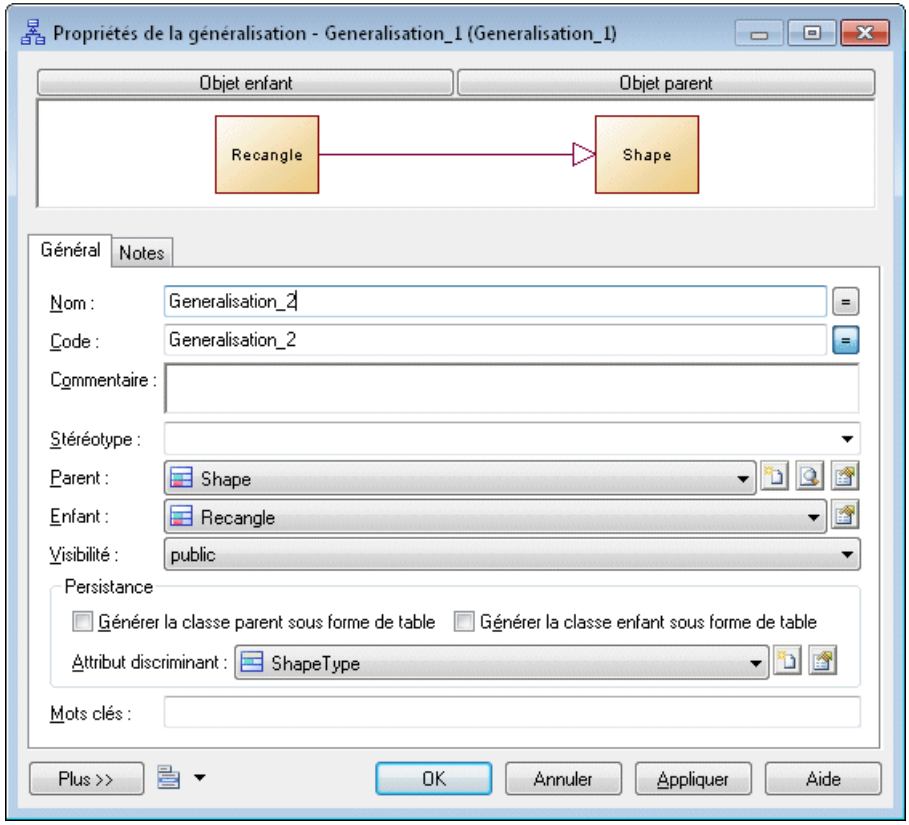
La totalité de la hiérarchie de classes est mise en correspondance avec une table. Une colonne discriminante de type de données caractère ou entier est définie pour distinguer les instances des différentes classes dans la hiérarchie.

Classes	Tables
	

1. Définissez les correspondances de classe pour chaque classe dans la hiérarchie, de sorte que toutes les classes ont la même table principale. Elles peuvent également être mises en correspondance avec des tables secondaires :



2. Définissez une correspondance d'identifiant dans la classe racine.
3. Définissez des correspondances d'attribut ou des correspondances d'association pour chaque classe.
4. Sélectionnez l'un des attributs dans la classe racine, dans la liste Attribut discriminant de la feuille de propriétés d'un des liens d'héritage enfant de la classe racine afin de le spécifier comme colonne discriminante, utilisée pour faire la distinction entre les différentes instances de classe. Dans l'exemple suivant, nous définissez un attribut supplémentaire*shapeType* dans *Shape* et sélectionnez-le comme attribut discriminant :

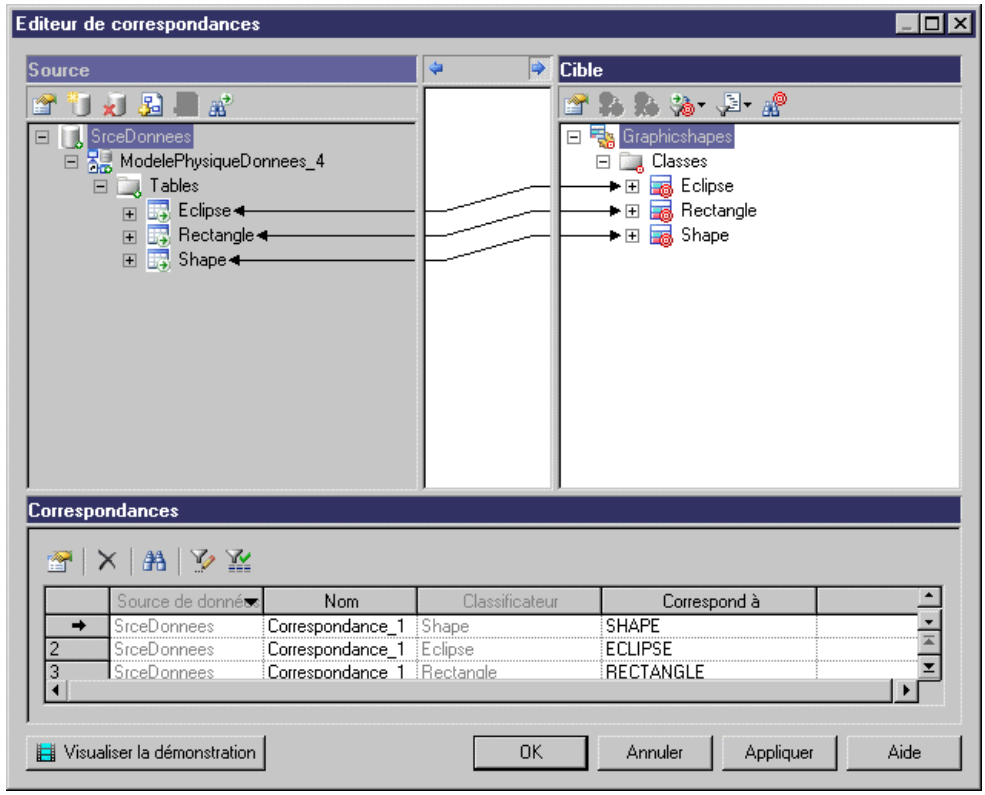


5. Définissez un type de génération de persistance pour chaque classe. Définissez le type de persistance de génération de la classe racine à *Générer une table* et toutes les autres classes à *Migrer les colonnes*.

Stratégie de mise en correspondance d'héritage de sous-classe jointe

Chaque classe est mise en correspondance avec sa propre table principale. Chaque table principale a une clé de référence qui pointe sur une table principale de sa classe parent, sauf en

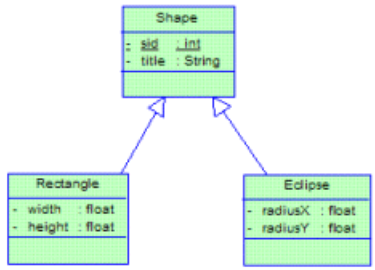
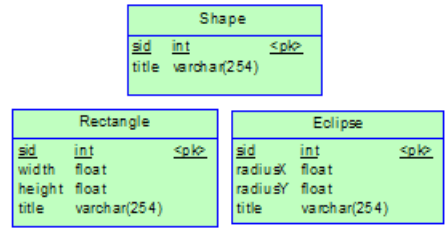
ce qui concerne la table principale de la classe racine. La clé de référence doit effectuer la jointure sur la clé primaire de la table principale.



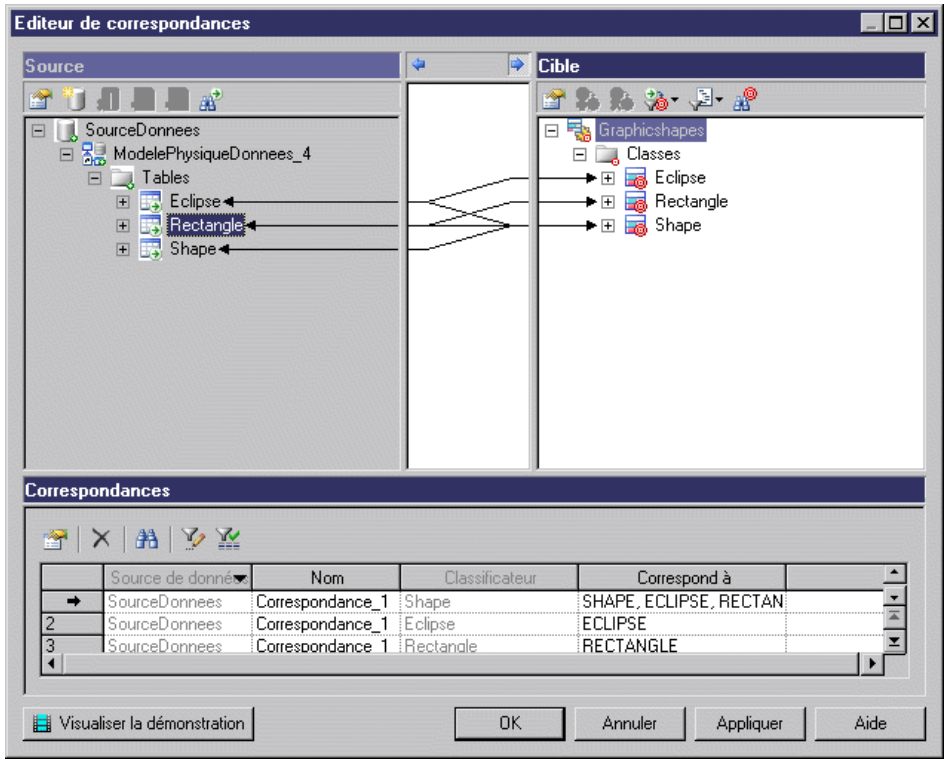
1. Définissez les correspondances de classe pour chaque classe dans la hiérarchie. Chaque classe est mise en correspondance avec sa propre table principale.
2. Définissez une correspondance d'identifiant dans la classe racine.
3. Définissez des correspondances d'attribut ou des correspondances d'association pour chaque classe.
4. Définissez le type de génération de persistance pour chaque classe.
5. Définissez le type de génération de persistance de toutes les classes comme *Générer une table*.

Stratégie de mise en correspondance d'héritage "une table par classe"

Chaque classe est mise en correspondance avec sa propre table principale. Tous les attributs persistants de la classe, y compris les attributs persistants hérités, sont mis en correspondance avec des colonnes de la table.

Classes	Tables
	

1. Définissez des correspondances de classe d'entité pour chaque classe de la hiérarchie, en mettant chaque classe en correspondance avec sa propre table principale :



2. Définissez des correspondances d'attribut et des correspondances d'association pour chaque classe.
3. Définissez une correspondance d'identifiant dans la classe racine.
4. Définissez une type de génération de persistance pour chaque classe.
5. Définissez la type de génération de persistance *Générer une table* pour les classes extrémité et *Migrer les colonnes* pour toutes les autres classes.

Remarque : Les superclasses peuvent également être mises en correspondance avec des tables principales de sous-classes si des attributs persistants hérités sont mis en correspondance de façons différentes pour les sous-classes, par exemple avec des colonnes différentes. L'autre table principale peut être seulement des tables secondaires. PowerAMC générera ces tables secondaires pour les superclasses.

Pour ce type de stratégie, certaines superclasses peuvent n'être en correspondance avec aucune table. Ces classes sont utilisées pour définir des informations d'état et de correspondance qui peuvent être héritées par leurs sous-classes.

Chapitre 20 Génération d'objet persistants pour Java et de pages JSF

PowerAMC prend en charge la génération d'objets persistants pour Hibernate et pour EJB3, ainsi que des JavaServer Faces pour Hibernate.

Génération d'objets persistants Hibernate

Hibernate est un projet open source développé par JBoss, qui fournit une solution puissante et performante de persistance objet/relationnel transparente et d'interrogation par requête pour Java. PowerAMC peut générer des fichiers de correspondances O/R Hibernate pour vos MOO Java.

Pour activer Hibernate dans votre modèle, sélectionnez **Modèle > Extensions**, cliquez sur l'outil **Attacher une extension**, sélectionnez le fichier `Hibernate` (sur l'onglet **Correspondance O/R**), puis cliquez sur **OK** pour l'attacher.

Hibernate permet de développer des objets persistants en utilisant des POJO (Plain Old Java Object). Toutes les idiomes Java communs, y compris les associations, l'héritage, le polymorphisme, la composition et les structures de collections Java sont pris en charge. Hibernate permet d'exprimer des requêtes dans sa propre extension SQL portable (HQL), ainsi qu'en SQL natif, ou au moyen d'objets Criteria et Example Java.

PowerAMC prend en charge la conception de classes Java, d'une structure de base de données et de la correspondance objet/relationnel (O/R). En utilisant ces métadonnées, PowerAMC peut générer des objets persistants Hibernate tels que :

- Classes Java persistantes (objets spécifiques à un domaine)
- Fichier de configuration
- Fichiers de correspondance O/R
- DAO factory
- DAO (Data Access Objects, objets d'accès aux données)
- Classes de test unitaire pour les tests automatisés

Définition des options par défaut Hibernate

Vous pouvez définir ces options dans le modèle ou dans une feuille de propriétés de package.

1. Affichez la feuille de propriétés du modèle ou du package, puis cliquez sur l'onglet **Hibernate Default Options** :
2. Définissez les options par défaut de niveau modèle ou package.

Option	Description
Importation automatique	Spécifie que l'utilisateur peut employer un nom de classe non qualifié dans les requêtes.
Accès par défaut	Spécifie le type d'accès par défaut à l'attribut de classe.
Cascade par défaut	Spécifie le type de cascade par défaut.
Nom du schéma	Spécifie le nom du schéma de base de données par défaut.
Nom de catalogue	Spécifie le nom de catalogue de base de données par défaut.

Définition des paramètres de configuration Hibernate

Hibernate peut prendre en charge plusieurs bases de données. Vous devez définir les paramètres de configuration de base de données. Les paramètres de configuration de base de données sont stockés dans le fichier de configuration, nommé hibernate.cfg.xml.

1. Affichez la feuille de propriétés du modèle et cliquez sur l'onglet **Hibernate Configuration**.
2. Définissez le type de base de données, le pilote JDBC™, l'URL de connexion, le chemin d'accès du fichier JAR de pilote JDBC, le nom d'utilisateur, le mot de passe, etc.

Option	Description
Dialecte	Spécifie le dialecte, et donc le type de base de données. Balise Hibernate : dialect
Classe du pilote JDBC	Spécifie la classe du pilote JDBC. Balise Hibernate : connection.driver_class
URL de connexion	Spécifie la chaîne d'URL de connexion JDBC. Balise Hibernate : connection.url
Fichier jar du pilote JDBC	Spécifie le chemin d'accès du fichier jar du pilote JDBC. Balise Hibernate : N/A
Nom d'utilisateur	Spécifie le nom d'utilisateur de base de données. Balise Hibernate : connection.username
Mot de passe	Spécifie le mot de passe de base de données. Balise Hibernate : connection.password

Option	Description
Affichage du code SQL	Spécifie que les instructions SQL doivent être affichées dans le journal. Balise Hibernate : show_sql
Exportation de schéma automatique	Spécifie le mode de création des tables. Balise Hibernate : hbm2ddl.auto
Préfixe de package	Spécifie un préfixe d'espaces de noms pour tous les packages dans le modèle. Balise Hibernate : Aucun
Taille de pool de connexions	Spécifie le nombre maximal de connexions en pool. Balise Hibernate : connection.pool_size

Vous pouvez vérifier les paramètres de configuration dans l'onglet **Aperçu**.

Définition des correspondances O/R de base Hibernate

Il existe deux types de classes dans Hibernate, les classes d'entité (entity classes) et les classes de type de valeur (value type classes). Les classes d'entité ont leur propres identité de base de données, fichiers de correspondances et cycles de vie, alors que les classes de type de valeur en sont dépourvues. Les classes de type de valeur dépendent des classes d'entité. Les classes de type de valeur sont également appelées classes de composant.

Hibernate utilise des fichiers de correspondances pour définir les métadonnées de correspondance. Chaque fichier de correspondance <Class>.hbm.xml peut contenir des métadonnées pour une ou plusieurs classes. PowerAMC utilise la stratégie de regroupement suivante :

- Un fichier de correspondances distinct est généré pour chaque classe d'entité qui n'est pas une hiérarchie d'héritage.
- Un fichier de correspondances distinct est généré pour chaque hiérarchie d'héritages dotée d'une stratégie de correspondance unique. Toutes les correspondances des sous-classes sont définies dans le fichier de correspondance. Le fichier de correspondance est généré pour la classe racine de la hiérarchie. Voir *Définition des correspondances d'héritage Hibernate* à la page 553 pour plus de détails sur la définition de la stratégie de correspondance.
- Aucun fichier de correspondance n'est généré pour une classe de type valeur unique qui n'appartient pas à une hiérarchie d'héritage. Sa correspondance est définie dans le fichier de correspondance de son propriétaire.

Définition des options de correspondance de classe Hibernate

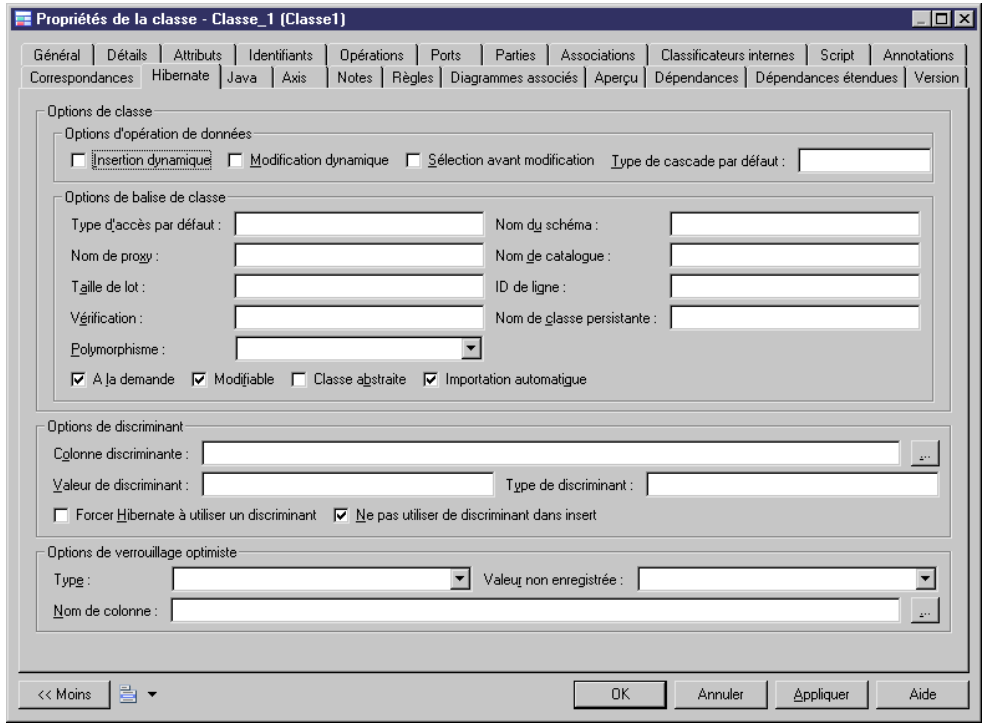
Les classes peuvent être mises en correspondances avec des tables ou des vues. Les vues ayant de nombreuses contraintes et des fonctionnalités limitées (par, exemple elles n'ont ni clé

primaire ni clés de référence), certaines vues ne peuvent pas être mises à jour, et les correspondances peuvent ne pas fonctionner correctement dans certains cas.

Certaines conditions doivent être remplies pour pouvoir générer une correspondance pour une table particulière :

- La source Java peut être générée. Cela n'est pas possible par exemple lorsque la visibilité de la classe est private.
- La classe est persistante.
- Le mode de génération n'est pas défini à Générer un type de données abstrait.
- Si la classe est une classe interne, elle doit être statique et avoir une visibilité public. Hibernate doit alors être en mesure de créer des instances de la classe.

Les options de correspondance de classe spécifiques à NHibernate sont définies dans l'onglet NHibernate de la feuille de propriétés d'une classe :



Option	Description
Insertion dynamique	Spécifie qu'une instruction INSERT SQL doit être générée lors de l'exécution et contiendra uniquement les colonnes dont les valeurs ne sont pas nulles. Balise Hibernate : dynamic-insert

Option	Description
Modification dynamique	Spécifie qu'une instruction UPDATE SQL doit être générée lors de l'exécution et contiendra uniquement les colonnes dont les valeurs ont été modifiées. Balise Hibernate : dynamic-update
Sélection avant modification	Spécifie que Hibernate ne doit jamais procéder à un SQL UPDATE à moins d'être certain qu'un objet soit effectivement modifié. Balise Hibernate : select-before-update
Type de cascade par défaut	Spécifie le style de cascade par défaut. Balise Hibernate : default-cascade
Type d'accès par défaut	Spécifie le type d'accès par défaut (champ ou propriété) Balise Hibernate : default-access
Nom de proxy	Spécifie une interface à utiliser pour l'initialisation de proxy à la demande. Balise Hibernate : proxy
Taille de lot	Spécifie une "taille de lot" pour la récupération des instances de cette classe par identifiant. Balise Hibernate : batch-size
Vérification	Spécifie une expression SQL utilisée pour générer une contrainte de vérification multiligne pour la génération automatique de schéma. Balise Hibernate : check
Polymorphisme	Spécifie si un polymorphisme de requête implicite ou explicite est utilisé. Balise Hibernate : polymorphism
Nom du schéma	Spécifie le nom du schéma de base de données. Balise Hibernate : schema
Nom de catalogue	Spécifie le nom du catalogue de base de données. Balise Hibernate : catalog
ID de ligne	Spécifie que Hibernate peut utiliser la colonne ROWID sur les bases de données qui la prennent en charge (par exemple, Oracle). Balise Hibernate : rowed
Nom de classe persistante	Spécifie une classe de persistance personnalisée. Balise Hibernate : persister
A la demande	Spécifie que la classe doit être à la demande. Balise Hibernate : lazy

Option	Description
Modifiable	Spécifie que les instances de la classe sont modifiables. Balise Hibernate : mutable
Classe abstraite	Spécifie que la classe est abstraite. Balise Hibernate : abstract
Importation automatique	Spécifie qu'un nom de classe non qualifié peut être utilisé dans une requête. Balise Hibernate : Auto-import
Colonne discriminante	Spécifie la colonne ou formule discriminante pour le comportement polymorphique dans une table par stratégie de correspondance hiérarchique. Balise Hibernate : discriminator
Valeur discriminante	Spécifie une valeur qui distingue les sous-classes individuelles, qui sont utilisées pour un comportement polymorphique. Balise Hibernate : discriminator-value
Type discriminant	Spécifie le type discriminant. Balise Hibernate : type
Forcer Hibernate à utiliser un discriminant	Force Hibernate à spécifier des valeurs discriminantes admises et ce, même lors de l'extraction de toutes les instances de la classe racine. Balise Hibernate : force
Ne pas utiliser de discriminant dans insert	Force Hibernate à ne pas inclure la colonne dans les SQL INSERT Balise Hibernate : insert
Type de verrouillage optimiste	Spécifie une stratégie de verrouillage optimiste. Balise Hibernate : optimistic-lock
Nom de colonne (de verrouillage optimiste)	Spécifie la colonne pour le verrouillage optimiste. Un champ est également généré si cette option est définie. Balise Hibernate : version/ timestamp
Valeur non enregistrée (de verrouillage optimiste)	Spécifie si une valeur non enregistrée est nulle ou indéfinie. Balise Hibernate : unsaved-value

Correspondances d'identifiant primaire

La correspondance d'identifiant primaire est obligatoire dans Hibernate. Les identifiants primaires des classes d'entité sont mis en correspondance avec les clés primaires des tables principales dans les source de données. Si vous n'en définissez aucune, une correspondance d'identifiant primaire par défaut est générée, mais risque de ne pas fonctionner correctement.

Il existe trois types de correspondance d'identifiant primaire dans Hibernate :

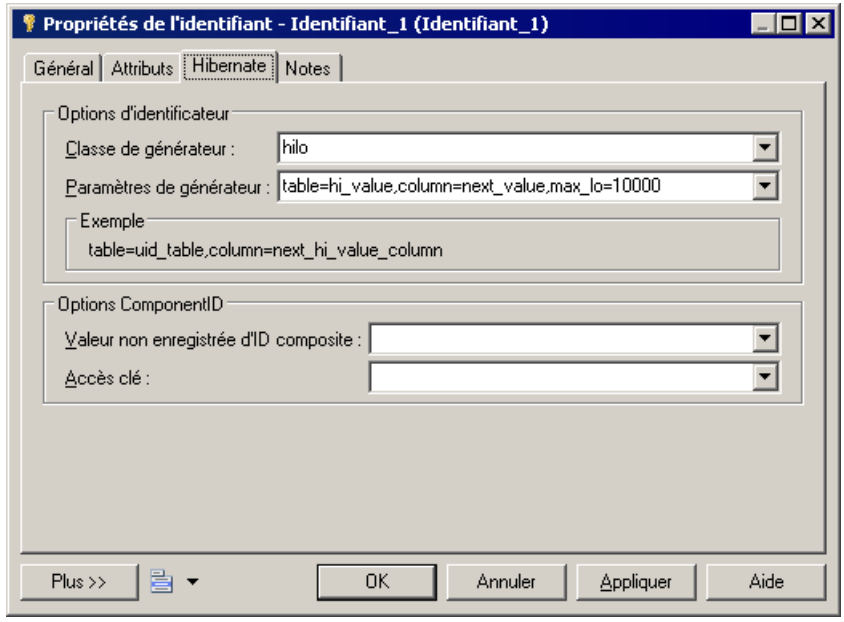
- Correspondance d'identifiant simple
- Correspondance d'identifiant composite
- Correspondance d'identifiant de composant

Les classes mises en correspondance doivent déclarer la colonne de clé primaire de la table de base de données. La plupart des classes auront également une propriété Java-Beans-style qui contiendra l'identificateur unique d'une instance.

Correspondance d'identifiant simple

Si une clé primaire est attachée à une colonne unique, seul un attribut dans l'identifiant primaire peut être mis en correspondance. Ce type de clé primaire peut être généré automatiquement. Vous pouvez définir la classe de générateur ainsi que les paramètres. Il existe de nombreux types de classe de générateur tels que increment, identity, sequence, etc. Chaque type de classe de générateur peut avoir des paramètres qui lui sont propres. Pour plus d'informations, reportez-vous à votre documentation Hibernate.

Vous pouvez définir la classe de générateur et les paramètres dans l'onglet Hibernate de la feuille de propriétés d'un identifiant primaire. Les paramètres ont la forme suivante param1=value1; param2=value2.

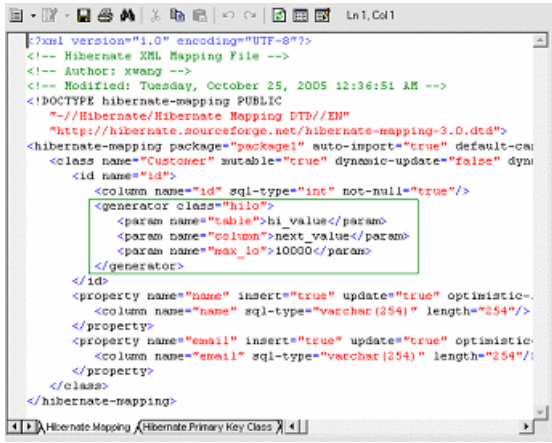


1. Affichez la feuille de propriétés de la classe, puis cliquez sur l'onglet Attributs.
2. Créez un attribut et définissez-le comme attribut Primaire.

3. Cliquez sur l'onglet Identifiants, puis double-cliquez sur l'entrée appropriée pour afficher sa feuille de propriétés.
4. Cliquez sur l'onglet Hibernate, sélectionnez une classe de générateur et définissez ses paramètres.

   Exemples de paramètres :

   - Sélectionnez hilo dans la liste Classe de générateur
   - Saisissez "table=hi_value,column=next_value,max_lo=10000" dans la zone Paramètres de générateur. Vous devez utiliser des virgules pour séparer les paramètres.
5. Vous pouvez visualiser le code dans l'onglet Aperçu :

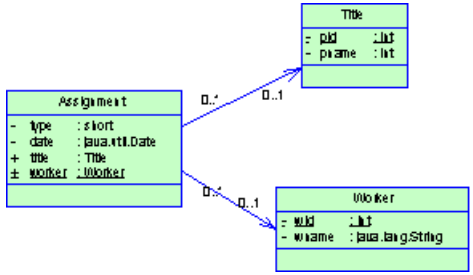


   Notez que, s'il existe plusieurs attributs d'identifiant primaire, le générateur n'est pas utilisé.

Correspondance d'identifiant composite

Si une clé primaire inclut plusieurs colonnes, l'identifiant primaire peut avoir plusieurs attributs mis en correspondance avec ces colonnes. Dans certains cas, la colonne de clé primaire peut également être colonne de clé étrangère.

1. Définissez des correspondances d'associations.
2. Migrez les rôles navigable des associations.
3. Ajoutez ces attribut migrés dans un identifiant primaire. Les attributs migrés n'ont pas besoin d'être mis en correspondance.



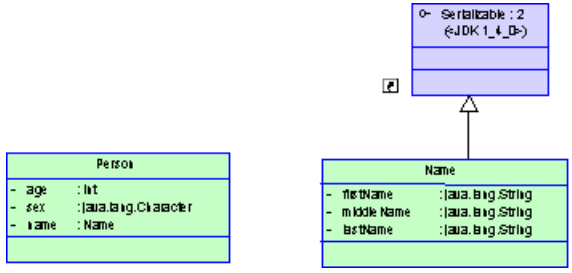
Dans l'exemple ci-dessus, la classe Assignment a un identifiant primaire doté de trois attributs : un attribut de base, et deux attributs migrés. La correspondance d'identifiant se présente comme suit :

```
<composite-id>
  <key-property name="type">
    <column name="type" sql-type="smallint"
    not-null="true"/>
    </key-property>
    <key-many-to-one name="title">
    </key-many-to-one>
    <key-many-to-one name="worker">
    </key-many-to-one>
</composite-id>
```

Correspondance d'identifiant primaire de composant

Pour plus de commodité, un identifiant composite peut être mis en oeuvre via une classe de type valeur séparée. L'identifiant primaire a juste un attribut ayant le type class. La classe séparée doit être définie comme classe de type valeur. La correspondance de classe de composant sera alors générée.

1. Définissez un attribut d'identifiant primaire.
2. Définissez le type de l'attribut comme une classe de type valeur.
3. Définissez la propriété Classe de générateur de l'attribut d'identifiant primaire à Embedded.
4. Définissez le type valeur de la classe d'identifiant primaire à true.
5. Définissez une correspondance pour la classe d'identifiant primaire.



Dans l'exemple ci-dessous, trois attributs liés au nom sont groupés dans une classe distincte Name. Cette classe est mise en correspondance avec la même table que la classe Person. L'identifiant primaire généré se présente comme suit :

```
<composite-id name="name" class="identifier.Name">
  <key-property name="firstName">
    <column name="name_firstName"
    sql-type="text"/>
  </key-property>
  <key-property name="middleName">
    <column name="name_middleName"
    sql-type="text"/>
  </key-property>
  <key-property name="lastName">
    <column name="name_lastName"
    sql-type="text"/>
  </key-property>
</composite-id>
```

Remarque : La classe de type valeur doit réaliser l'interface java.io.Serializable, qui met en oeuvre les méthodes equals() et hashCode().

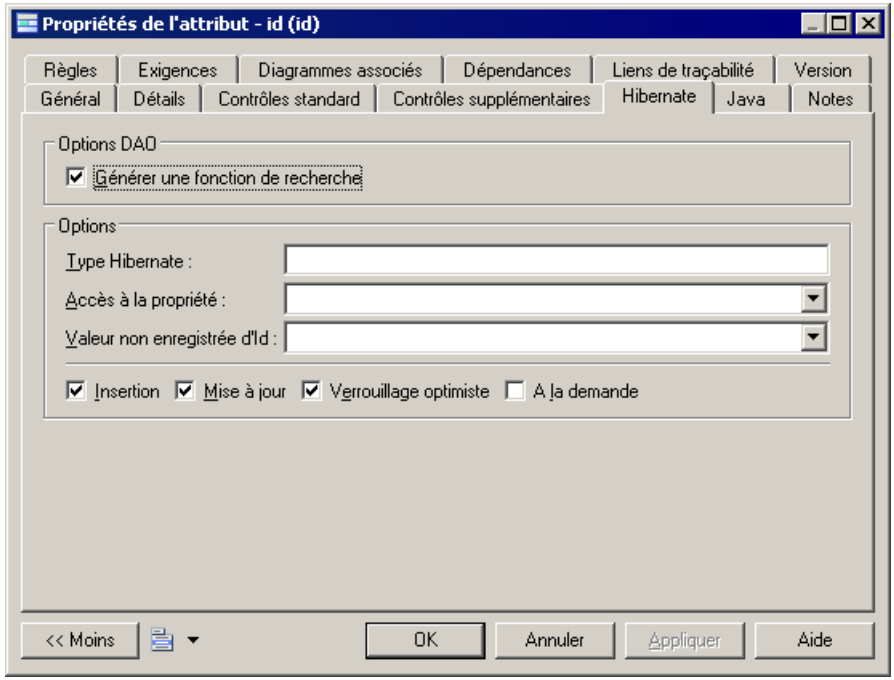
Définition des correspondances d'attributs

Les attributs peuvent être des attributs migrés ou des attributs ordinaires. Les attributs ordinaires peuvent être mis en correspondance avec des colonnes ou des formules. Les attributs migrés ne requièrent pas de correspondance d'attribut.

Les types de correspondances suivants sont possibles :

- Correspondance d'attribut et d'une formule - Lorsque vous mettez en correspondance un attribut et une formule, vous devez vous assurer que la syntaxe est correcte. Il n'y a aucune colonne dans la table source de la correspondance d'attribut.
- Correspondance d'attribut de composant - Une classe de composant peut définir la correspondance d'attribut comme pour une autre classe, à ceci près qu'il n'y pas d'identifiant primaire.
- Correspondance discriminante - Dans la correspondance par héritage avec une stratégie "une table par hiérarchie", le discriminant doit être spécifié dans la classe racine. Vous pouvez définir le discriminant dans l'onglet Hibernate de la feuille de propriétés de classe.

Les options relatives à la correspondance d'attribut Hibernate sont définies dans l'onglet Hibernate de la feuille de propriétés d'attribut.



Option	Description
Générer une fonction de recherche	Génère une fonction de recherche pour l'attribut.
Type Hibernate	Spécifie un nom qui indique le type Hibernate.
Accès à la propriété	Spécifie la stratégie que Hibernate doit utiliser pour accéder à la valeur de la propriété.
Valeur non enregistrée d'ID	Spécifie la valeur d'un ID non enregistré.
Insertion	Spécifie que les colonnes doivent être incluses dans n'importe quelle instruction SQL INSERT.
Mise à jour	Spécifie que les colonnes mises en correspondance doivent être incluses dans n'importe quelle instruction SQL UPDATE.
Verrouillage optimiste	Spécifie que les mises à jour de cette propriété requièrent l'acquisition du verrouillage optimiste.
A la demande	Spécifie que cette propriété doit être chargée à la demande lors du premier accès à la variable d'instance (requiert l'utilisation de bytecode au moment de la compilation).

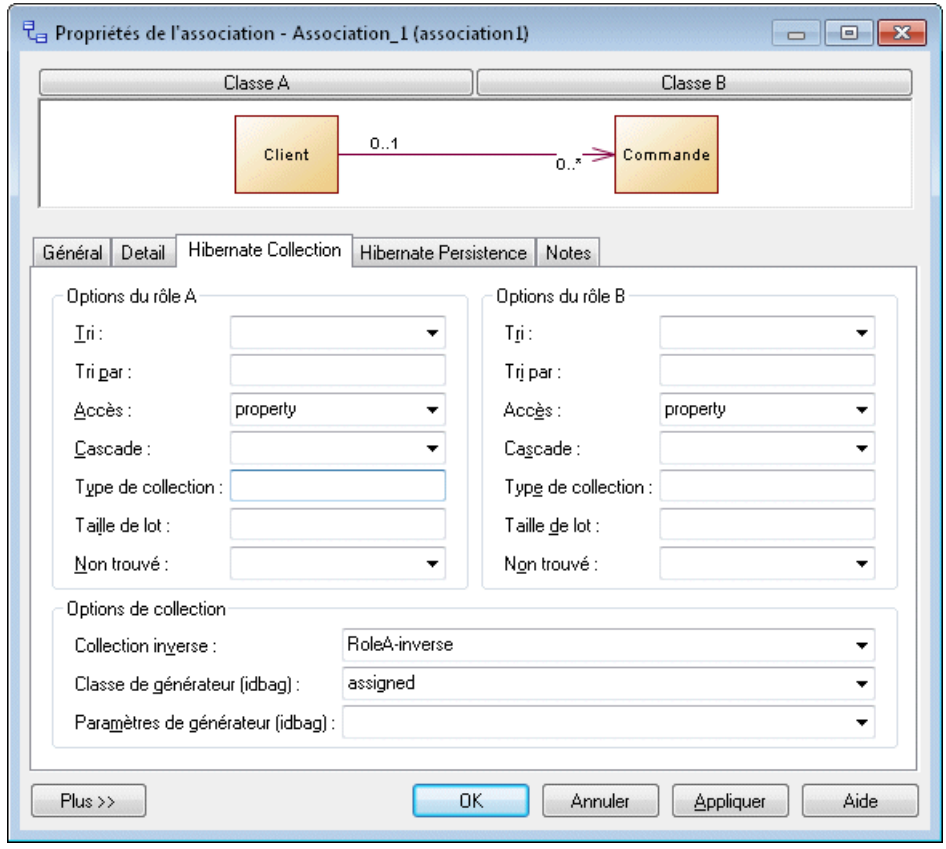
Correspondances d'association Hibernate

Hibernate prend en charge les correspondances d'association un-un, un-plusieurs/plusieurs-un et plusieurs-plusieurs. La modélisation des correspondances s'effectue de la même manière que pour une modélisation de correspondance O/R standard. Toutefois, Hibernate fournit des options particulières afin de définir ses correspondances d'association, qui seront enregistrées dans le fichier de correspondance <Class>.hbm.xml. PowerAMC permet de définir des attributs d'association standard tels que le conteneur, le type, la classe d'association, la navigabilité de rôle et la taille de tableau, ainsi que des attributs étendus spécifiques pour les correspondances d'association Hibernate.

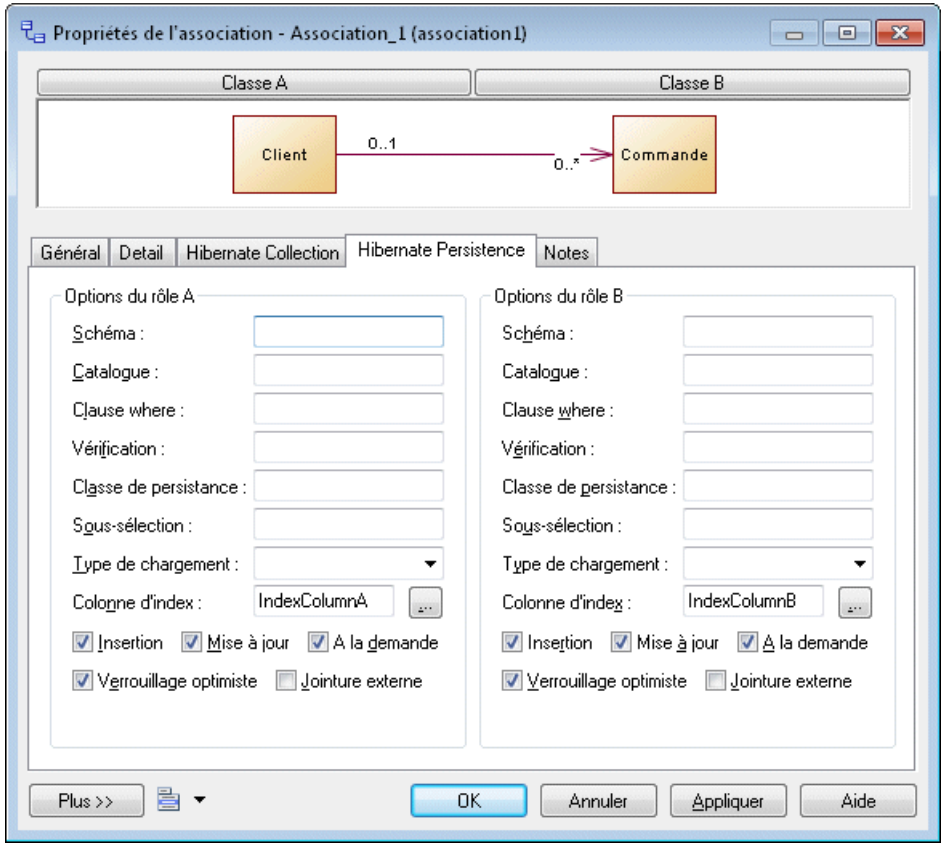
Définition des options de correspondance d'association Hibernate

Vous définissez des options de correspondance d'association Hibernate de la façon suivante :

1. Affichez la feuille de propriétés de l'association et cliquez sur l'onglet Hibernate Collection.
2. Définissez les options de gestion de collection (voir *Options de gestion des collections* à la page 550).



3. Cliquez sur l'onglet Hibernate Persistence. Définissez les options de persistance (voir *Options de persistance* à la page 551).



Options de gestion des collections

Les options suivantes sont disponibles :

Option	Description
Tri	Spécifie une collection triée avec un ordre de tri naturel, ou une classe de comparaison donnée. Balise Hibernate : sort
Tri par	Spécifie une colonne (ou plusieurs colonnes) de table qui définit l'ordre d'itération de Set ou bag, ainsi qu'un asc ou desc facultatif. Balise Hibernate : order-by
Access	Spécifie la stratégie que Hibernate doit utiliser pour accéder à la valeur de la propriété. Balise Hibernate : access

Option	Description
Cascade	Spécifie quelles opérations doivent être transmises en cascade depuis l'objet parent vers l'objet associé. Balise Hibernate : cascade
Collection type	Spécifie un nom qui indique le type Hibernate. Balise Hibernate : type
Taille de lot	Spécifie la taille de lot. Balise Hibernate : batch-size
Non trouvé	Spécifie de quelle façon les clés étrangères qui référencent les lignes manquantes seront gérées : ignore traite une ligne manquante comme une association nulle. Balise Hibernate : not-found
Collection inverse	Spécifie que le rôle est la relation inverse du rôle opposé. Balise Hibernate : inverse

Options de persistance

Les options suivantes sont disponibles :

Option	Description
Schéma	Spécifie le nom du schéma. Balise Hibernate : schema
Catalogue	Spécifie le nom du catalogue. Balise Hibernate : catalog
Clause Where	Spécifie une condition SQL WHERE arbitraire à utiliser pour la récupération des objets de cette classe. Balise Hibernate : where
Vérification	Spécifie une expression SQL utilisée pour générer une contrainte de vérification multiligne pour la génération automatique de schéma. Balise Hibernate : check
Type de chargement	Spécifie un chargement par jointure externe ou par sélection séquentielle. Balise Hibernate : fetch
Classe de persistance	Spécifie une classe de persistance personnalisée. Balise Hibernate : persister

Option	Description
Sous-sélection	Spécifie une entité non modifiable et en lecture seule pour une sous-sélection de base de données. Balise Hibernate : subselect
Colonne d'index	Spécifie le nom de colonne si les utilisateurs utilisent le type de collection list ou array. Balise Hibernate : index
Insertion	Spécifie que les colonnes mises en correspondance doivent être incluses dans n'importe quelle instruction SQL INSERT. Balise Hibernate : insert
Mise à jour	Spécifie que les colonnes mises en correspondance doivent être incluses dans n'importe quelle instruction SQL UPDATE. Balise Hibernate : update
A la demande	Spécifie que cette propriété doit être chargée à la demande lors du premier accès à la variable d'instance. Balise Hibernate : lazy
Verrouillage optimiste	Spécifie si le numéro de version doit être incrémenté si la valeur de cette propriété est modifiée. Balise Hibernate : optimistic-lock
Jointure externe	Spécifie d'une jointure externe doit être utilisée. Balise Hibernate : outer-join

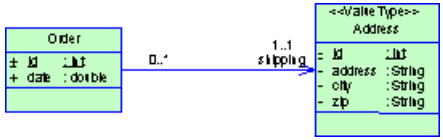
Mise en correspondance de collections de types valeur

S'il y a une classe de type valeur du côté du rôle navigable d'une association ayant la multiplicité un, PowerAMC va incorporer le type valeur dans le type d'entité sous la forme d'un attribut composite.

Mise en correspondance des collections de type valeur

Vous mettez en correspondance les collections de type valeur de la façon suivante :

1. Créez une classe d'entité.
2. Créez une autre classe pour le type valeur.
3. Affichez la feuille de propriétés de la classe de type valeur, cliquez sur l'onglet Détails, puis sélectionnez l'option Type de valeur.
4. Créez une association entre la classe de type valeur et une classe de type entité. Du côté du type valeur, définissez la multiplicité à un et la navigabilité à true.



5. Générez le MPD avec la correspondance O/R.
6. Affichez la feuille de propriétés de la classe d'entité, puis cliquez sur l'onglet Aperçu.
7. Vérifiez le fichier de correspondance.

Une class d'entité composite peut contenir des composants, en utilisant la déclaration <nested-composite-element>.

Définition d'un type de collection pour les associations un-plusieurs ou plusieurs-plusieurs

Vous définissez un type de collection pour les associations un-plusieurs ou plusieurs-plusieurs de la façon suivante :

1. Affichez la feuille de propriétés de l'association puis cliquez sur l'onglet Détails.
2. Spécifiez une valeur dans la zone Multiplicité pour les deux extrémités.
3. Spécifiez une navigabilité unidirectionnelle ou bidirectionnelle.
4. Spécifiez les noms de rôle si nécessaire.
5. Si une association de rôle est navigable et que la multiplicité est plusieurs, vous pouvez définir le type de conteneur de collection et la taille de chargement de lot.
6. Si vous sélectionnez java.util.List ou <none>, cela implique que vous allez utiliser un type de collection array ou list-indexed. Vous devez ensuite définir une colonne d'index pour préserver l'ordre des objets de la collection dans la base de données.

Remarque : le type de conteneur Java collection conditionne le type de collection Hibernate.

Type de conteneur de collection	Type de collection Hibernate
<None>	array
java.util.Collection	bag ou idbag (many-to-many)
java.util.List	list
java.util.Set	set

Définition des correspondances d'héritage Hibernate

Hibernate prend en charge 3 type de stratégies de correspondance d'héritage de base :

- Une table par hiérarchie de classes
- Une table par sous-classe
- Une table par classe concrète

- La définition standard de correspondance d'héritage dans Hibernate de diffère pas particulièrement de la définition de correspondance d'héritage standard dans la modélisation de correspondance OR. Toutefois, un fichier de correspondances distinct est généré pour chaque hiérarchie d'héritages dotée d'une stratégie de correspondance unique. Toutes les correspondances des sous-classes sont définies dans le fichier de correspondance. Le fichier de correspondance est généré pour la classe racine de la hiérarchie.

Génération de code pour Hibernate

Pour pouvoir générer du code pour Hibernate, vous devez :

- Installer Hibernate 3.0 ou une version supérieure.
- Vérifier le modèle.
- Définir des options de génération.

Vérification du modèle

Lorsque vous terminez la définition du modèle, vous devez utiliser la fonctionnalité de vérification de modèle afin de vous assurer qu'il n'y a aucun avertissement ou aucune erreur dans le modèle. S'il subsiste des erreurs, vous devez les corriger avant de générer le code.

Définition des options de génération

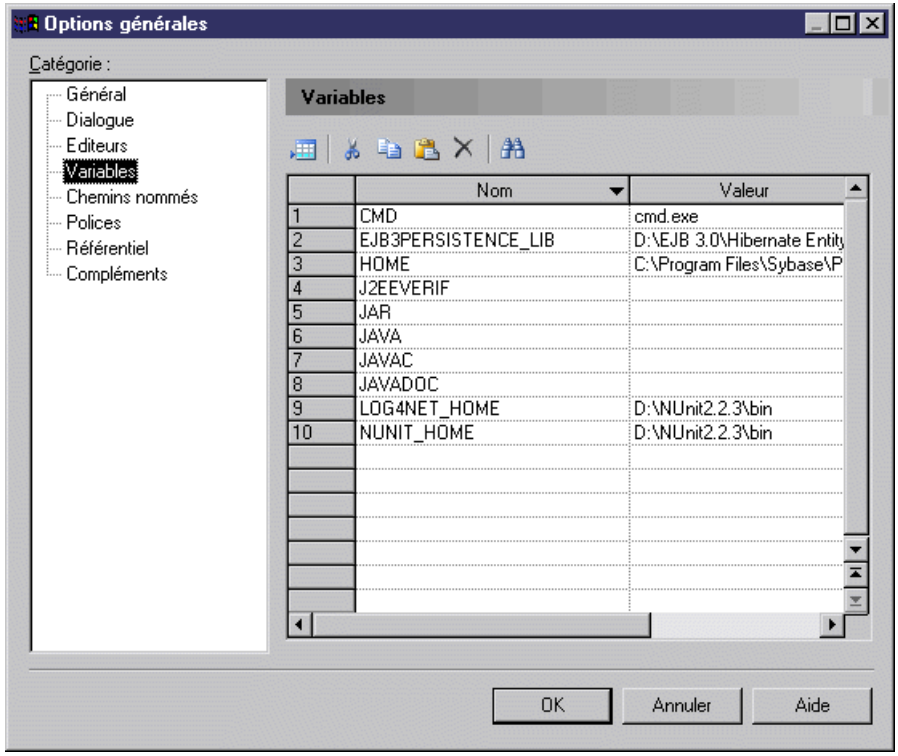
Il existe deux types d'option de génération :

- Variables d'environnement - pour permettre au script de compilation Eclipse ou Ant de trouver les fichiers Jar de bibliothèque Hibernate Jar
- Options de génération

Définition des variables d'environnement

Vous définissez les variables d'environnement de la façon suivante:

1. Sélectionnez **Outils > Options générales**.

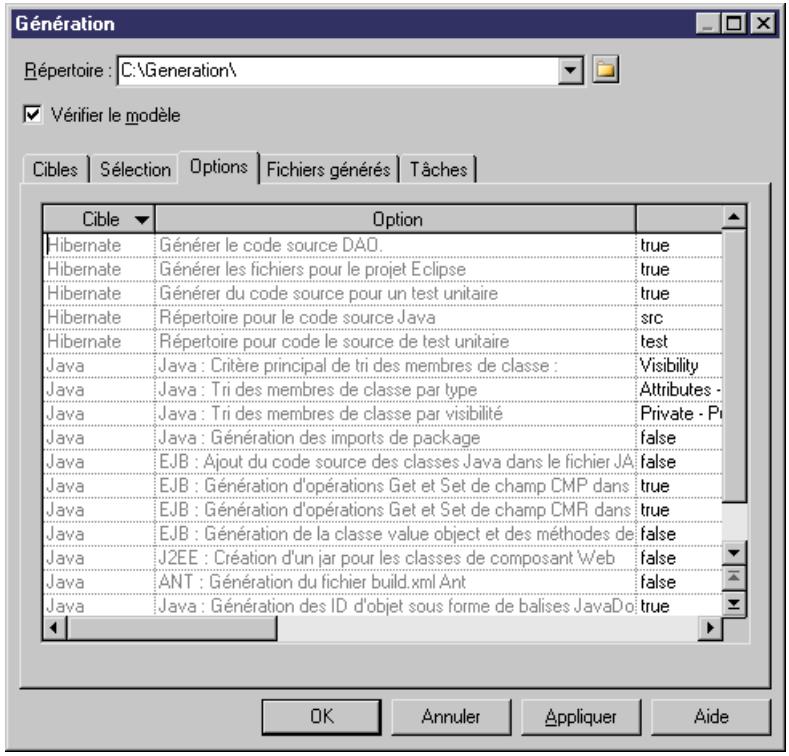


2. Sélectionnez le noeud Variables
3. Ajoutez une variable HIBERNATE_HOME et, dans la colonne Valeur, spécifiez le chemin du répertoire racine d'Hibernate. Par exemple D:\Hibernate-3.0.

Définition des options de génération

Vous définissez les options de génération de la façon suivante :

1. Sélectionnez **Langage > Générer du code Java** pour afficher la boîte de dialogue Génération.
2. Spécifiez le répertoire dans lequel les fichiers doivent être générés, et indiquez si vous souhaitez effectuer une vérification de modèle (voir *Chapitre 9, Vérification d'un MOO* à la page 303).
3. Cliquez sur l'onglet Options.



4. [facultatif] Pour utiliser DAO, définissez l'option Générer le code source DAO à true.
5. [facultatif] Pour utiliser Eclipse afin de compiler et tester les classes Java, définissez l'option Générer les fichiers pour le projet Eclipse à true.
6. [facultatif] Pour utiliser les classes de test unitaire afin de tester les objets persistants Hibernate, définissez l'option Générer du code source pour un test unitaire à true.
7. Cliquez sur OK pour générer le code immédiatement, ou sur Appliquer ou annuler afin d'enregistrer vos modifications pour des générations ultérieures.

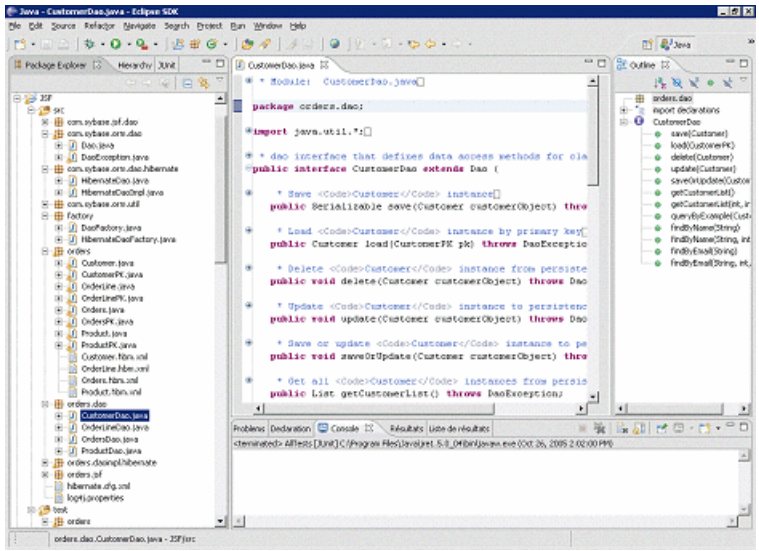
Génération de code pour Hibernate

Une fois que vous avez terminé votre modèle, que vous l'avez vérifié et que vous avez défini vos options de génération, vous pouvez générer le code pour Hibernate.

1. Sélectionnez **Langage > Générer du code Java** pour afficher la boîte de dialogue Génération.
2. Spécifiez le répertoire dans lequel les fichiers doivent être générés, et indiquez si vous souhaitez effectuer une vérification de modèle (voir *Chapitre 9, Vérification d'un MOO* à la page 303).
3. [facultatif] Cliquez sur l'onglet Sélection pour changer la sélection d'objets.

4. [facultatif] Cliquez sur l'onglet Options pour changer les options de génération Hibernate et Java.
5. [facultatif] Cliquez sur l'onglet Fichiers générés pour passer en revue tous les fichiers qui seront générés.
6. Cliquez sur **OK** pour lancer la génération.

Vous pouvez utiliser un IDE tel qu'Eclipse pour modifier le code généré, et le compiler, exécuter le test unitaire et développer votre application.



Utilisation du code Hibernate généré

Pour utiliser Eclipse, vous devez télécharge et installer le SDK Eclipse.

Importation du projet généré dans Eclipse

Si vous avez sélectionné l'option de génération Générer les fichiers pour le projet Eclipse, vous pouvez importer le projet généré dans Eclipse et utiliser Eclipse pour modifier, compiler et exécuter les tests.

Si vous utilisez ensuite le plugin Eclipse pour PowerAMC, après la génération du code, le projet est automatiquement importé ou réactualisé dans Eclipse.

Si vous utilisez la version autonome de PowerAMC, vous devez importer le projet généré, comme suit :

1. Dans Eclipse, Sélectionnez **Fichier > Importer**
2. Dans la liste d'importation, sélectionnez Projets existants dans l'espace de travail. Eclipse va automatiquement compiler toutes les classes Java. En cas d'erreur, vous devez vérifier :

- Que tous les fichiers Jar requis soient inclus dans le fichier .classpath.
- Que la version du JDK soit la version appropriée. Si vous utilisez Java 5.0 comme langage dans votre MOO, vous devez utiliser le JDK 5.0 pour compiler le code.

Réalisation des tests unitaires

Si les classes Java générées sont compilées sans erreur, vous pouvez exécuter les tests au sein d'Eclipse ou en utilisant Ant.

Les tests unitaires génèrent des données aléatoires pour créer et mettre à jour des objets.

Après la création, la mise à jour, la suppression ou la recherche fructueuse d'objets, un test évalue si le résultat obtenu était le résultat attendu.

Si tel est le cas, le test réussit. Dans le cas contraire, il échoue.

Avant d'exécuter les tests unitaires, vous devez effectuer les opérations suivantes :

1. Créer le fichier de base de données
2. Définir une connexion ODBC.
3. Générer la base de données à partir du MPD en utilisant la connexion ODBC.
4. Donner à l'utilisateur du test la permission de se connecter à la base de données.
5. Démarrer la base de données.

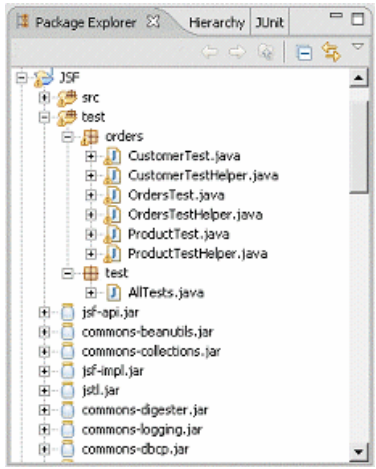
Exécution de tests unitaires dans Eclipse

Eclipse intègre JUnit. Les fichiers Jar JUnit et fichiers d'interface utilisateur JUnit sont fournis.

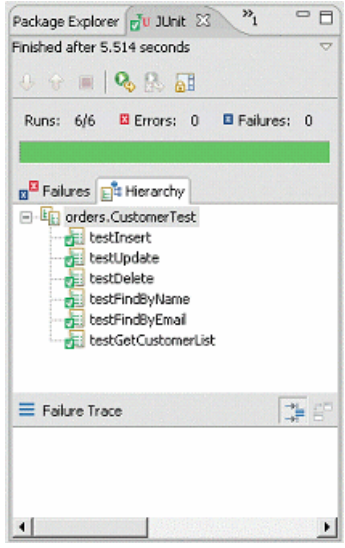
Exécution d'un seul test unitaire

Vous exécutez un seul test unitaire de la façon suivante :

1. Ouvrez la perspective Java.
2. Dans l'explorateur de packages, développez le package de test.



3. Pointez sur un cas de test (par exemple, CustomerTest.java), cliquez le bouton droit, puis sélectionnez **Run As > JUnit Test**.
4. Sélectionnez la vue JUnit pour vérifier le résultat :



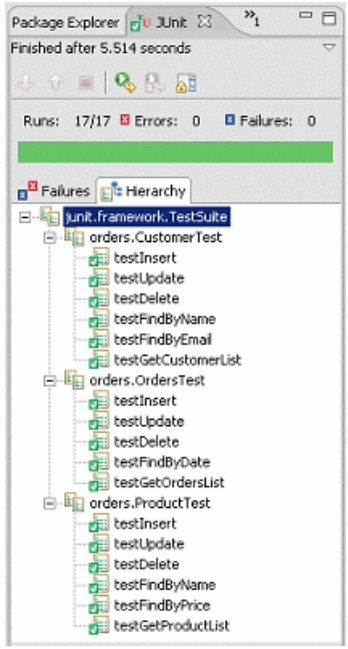
S'il y a 0 erreur, le test a réussi. S'il y a des erreurs, vous devez vérifier la vue Console pour localiser la source de ces erreurs. Il peut s'agir d'un des types de problème suivants :

- La base de données n'est pas démarrée
- Le nom d'utilisateur ou le mot de passe est incorrect.
- La base de données n'est pas générée.
- La correspondance est incorrecte.

Exécution de la suite de test

Vous exécutez la suite de test de la façon suivante :

1. Ouvrez la perspective Java.
2. Dans l'explorateur de packages, développez le package de test.
3. Pointez sur la suite AllTests.java, cliquez le bouton droit, puis sélectionnez **Run As > JUnit Test**.
4. Sélectionnez la vue JUnit pour vérifier le résultat :



Exécution de tests unitaires avec Ant

Pour générer le fichier build.xml de Ant, vous devez sélectionner l'option Génération du fichier build.xml Ant dans la fenêtre de génération de code Java.

Pour pouvoir utiliser Ant, vous devez :

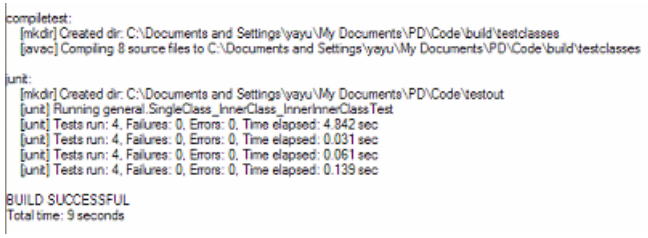
- Le télécharger depuis le site *http://www.apache.org*, et l'installer.
- Définir une variable d'environnement ANT_HOME et la faire pointer vers le répertoire d'installation de Ant.
- Copier junit-3.8.1.jar depuis le répertoire HIBERNATE_HOME/lib dans le répertoire ANT_HOME/lib.
- Vous assurer que les fichiers Jar Hibernate sont définis dans le fichier build.xml ou dans la variable d'environnement CLASSPATH.

- Vous assurer que le fichier Jar de pilote JDBC de votre base de données est définie dans le fichier build.xml ou dans la variable d'environnement CLASSPATH.

Exécution de tests unitaires à l'aide de Ant depuis PowerAMC

Vous exécutez des tests unitaires à l'aide de Ant depuis PowerAMC de la façon suivante :

1. Sélectionnez **Langage > Générer du code Java** pour afficher la boîte de dialogue Génération.
2. Spécifiez le répertoire dans lequel les fichiers doivent être générés, et indiquez si vous souhaitez effectuer une vérification de modèle (voir *Chapitre 9, Vérification d'un MOO* à la page 303).
3. Cliquez sur l'onglet Options.
4. Définissez l'option Génération du fichier build.xml Ant à true.
5. Cliquez sur l'onglet tâches.
6. Sélectionnez la tâche Hibernate : Exécuter les test unitaire générés.
7. Cliquez sur **OK** pour lancer la génération.
8. Une fois que vous avez fermé la liste des fichiers générés, la tâche JUnit s'exécute. Vous pouvez voir le résultat dans la fenêtre Résultats.



Exécution de tests unitaires avec Ant depuis la ligne de commande

Vous exécutez des tests unitaires avec Ant depuis la ligne de commande de la façon suivante :

1. Affichez une fenêtre de ligne de commande.
2. Passez dans le répertoire dans lequel vous avez généré le code.
3. Exécutez la tâche du test JUnit: Ant junit
4. Vérifiez le résultat généré.

Génération d'objets persistants EJB 3

EJB 3.0 introduit une spécification de correspondance O/R standard et le passage à une persistance basé sur les POJO. PowerAMC prend en charge EJB 3 via un fichier d'extension.

Pour activer EJB 3 dans votre modèle, sélectionnez **Modèle > Extensions**, cliquez sur l'outil **Attacher une extension**, sélectionnez le fichier `EJB 3.0` (sur l'onglet **Correspondance O/R**), puis cliquez sur **OK** pour l'attacher.

La persistance EJB 3.0 fournit une solution de persistance légère pour les applications Java. Elle prend en charge une persistance objet/relationnelle puissante et transparent, qui peut être utilisée dans le conteneur et hors du conteneur.

La persistance EJB 3.0 permet de développer des objets persistants qui incluent des POJO (Plain Old Java Object). Les principaux idiomes Java y compris les associations, l'héritage, le polymorphisme, la composition et les structures de collections sont pris en charge. La persistante EJB 3.0 permet de rédiger des requêtes dans sa propre extension SQL (EJBQL), ou en SQL natif.

PowerAMC prend en charge la conception de classes Java, d'une structure de base de données et de la correspondance objet/relationnel (O/R). En utilisant ces métadonnées, PowerAMC peut générer du code pour la persistance EJB 3 :

- Entités EJB persistantes (objets spécifiques au domaine)
- Fichier de configuration
- Fichiers de correspondances O/R (Facultatif))
- DAO factory
- Data Access Objects (DAO)
- Classes de test unitaire pour les tests automatisés

Génération d'entités pour EJB 3.0

Vous générez des entités pour EJB 3.0 de la façon suivante :

1. Créez un MOO et un MPD, puis définissez vos correspondances O/R. Pour obtenir des informations détaillées, voir *Chapitre 19, Modélisation des correspondances objet/relationnel (O/R)* à la page 507.
2. Définissez les paramètres de persistance EJB 3.
3. Générez du code Java.
4. Exécutez les tests unitaires.

Définition de la correspondance O/R de base EJB 3

Il existe trois types de classes persistantes dans EJB 3 :

- Classes d'entité
- Classes incorporables
- Superclasses mises en correspondance

Les exigences suivantes sont applicables aux classes persistantes :

- Elles doivent être définies sous la forme de classes persistantes (voir *Transformation de classe d'entité* à la page 509).
- Elles doivent être des classes racine (et non pas des classes internes).
- Les casses d'entité et les superclasses mises en correspondance doivent avoir le stéréotype EJBEntity.
- Les classes incorporables sont des classes type valeur, i.e. des classes persistantes ayant comme type de persistance un Type de valeur

Les classes qui ne satisfont pas à ces exigences seront ignorées.

Astuce : vous pouvez définir le stéréotype et la persistance de toutes les classes dans un modèle ou package (et sous-packages) en pointant sur le modèle ou package, et cliquant le bouton droit de la souris et en sélectionnant Make Persistent dans le menu contextuel.

Définition des correspondances d'entité

Définissez le stéréotype des classes persistantes pour les transformer en classes d'entité EJB 3.

L'annotation d'entité est générée pour spécifier que la classe est une entité.

```
@Entity
@Table(name="EMPLOYEE")
public class Employee { ... }
```

Pour plus d'informations sur la définition des correspondances de classes d'entité, voir *Chapitre 19, Modélisation des correspondances objet/relationnel (O/R)* à la page 507.

Options de correspondance d'entité EJB 3

Les options correspondance spécifiques à EJB3 suivantes peuvent être définies sur l'onglet EJB 3 Persistence de la feuille de propriétés d'une classe.

Option	Description
Nom d'entité	Spécifie que l'alias de classe peut être utilisé dans EJB QL.
Stratégie d'accès	Spécifie le type d'accès par défaut (FIELD ou PROPERTY)
Nom de schéma	Spécifie le nom du schéma de base de données.

Option	Description
Nom de catalogue	Spécifie le nom du catalogue de base de données.
Type de définition de correspondance	Spécifie ce qui sera généré pour les métadonnées de correspondance : le fichier de correspondances, les annotations, ou les deux.
Valeur de discriminant	Spécifie la valeur discriminante permettant de distinguer les instances de la classe.

Mise en correspondance avec plusieurs tables

Dans EJB 3, les classes d'entité peuvent être mises en correspondance avec plusieurs tables. Pour plus d'informations sur la mise en correspondance d'une classe d'entité et de plusieurs tables, voir *Chapitre 19, Modélisation des correspondances objet/relationnel (O/R)* à la page 507.

Aucune vérification ne permet d'assurer que les tables secondaires ont des clés de référence pointant sur des clés primaires.

L'annotation SecondaryTable est générée pour spécifier une table secondaire pour la classe d'entité annotée. L'annotation SecondaryTables est utilisée lorsqu'il y a plusieurs tables secondaires pour une entité.

Définition de la correspondance d'un identifiant primaire

Trois types de correspondance d'identifiant primaire sont pris en charge dans EJB 3.0 :

- Correspondance d'identifiant simple - Ce type de clé primaire peut être généré automatiquement dans EJB 3. Vous pouvez définir la classe et les paramètres de générateur. Il existe quatre types de classe de générateur : Identity, Sequence, Table et Auto. Les générateurs de table et de séquence nécessitent la définition de certains paramètres. Reportez-vous à la spécification de la persistance EJB 3.0 pour plus de détails.

  Vous pouvez définir la classe et les paramètres de générateur dans l'onglet EJB 3 Persistence de la feuille de propriétés d'un identifiant primaire. Les paramètres prennent la forme param1=value1; param2=value2.

  L'annotation Id générée spécifie la propriété de clé primaire ou le champ d'une entité. L'annotation GeneratedValue fournit la spécification des stratégies de génération pour les valeurs des clés primaires :

```
@Id
@GeneratedValue(strategy=GenerationType.TABLE,
generator="customer_generator")
@TableGenerator(
  name=" customer_generator",
  table="Generator_Table",
  pkColumnName="id",
  valueColumnName="curr_value",
  initialValue=4
)
@Column(name="cid", nullable=false)
```

- Correspondance d'identifiant composite - L'annotation IdClass sera générée pour une classe d'entité ou une superclasse mise en correspondance afin de spécifier une classe de clé primaire composite qui est mise en correspondance avec plusieurs champs ou propriétés de l'entité :

```
@IdClass(com.acme.EmployeePK.class)
@Entity
public class Employee {
  @Id String empName;
  @Id Date birthDay;
 ...
}
```

- Correspondance d'identifiant primaire incorporé - correspond à une correspondance d'identifiant primaire de composant. L'annotation EmbeddedId est générée pour un champ ou une propriété persistante d'une classe d'entité ou d'une classe de superclasse mise en correspondance pour signaler une clé primaire composite qui est une classe incorporable :

```
@EmbeddedId
protected EmployeePK empPK;
```

Définition des correspondances d'attribut

Chaque attribut persistant ayant un type de base peut être mis en correspondance avec une colonne. Suivez les instructions pour définir les correspondances d'attribut pour ce type d'attributs persistants.

Les options de correspondance d'attribut spécifiques à EJB3 sont disponibles sur l'onglet EJB 3 Persistence de la feuille de propriétés de chaque attribut :

Option	Description
Attribut de version	Spécifie si l'attribut est mis en correspondance en tant qu'attribut de version.
Insérable	Spécifie que les colonnes mises en correspondance doivent être incluses dans les instructions SQL INSERT.
Modifiable	Spécifie que les colonnes mises en correspondance doivent être incluses dans les instructions SQL UPDATE.
Charger	Spécifie si l'attribut doit être chargé à la demande.
Générer une méthode de recherche	Génère une fonction de recherche pour l'attribut.

L'annotation Basic est générée pour spécifier le mode de chargement pour l'attribut et la propriété et indiquer si cet attribut ou cette propriété est obligatoire. L'annotation Column est générée pour spécifier une colonne mise en correspondance pour une propriété ou un champ persistant.

```
@Basic
@Column(name="DESC", nullable=false, length=512)
public String getDescription() { return description; }
```

D'autres annotations peuvent également être générés pour indiquer le type de persistance d'un attribut ou d'une propriété. Une annotation Temporal spécifie qu'une propriété ou un attribut persistant doit persister comme un type temporel. L'annotation enumerated existe également pour les types enumerated et Lob pour les types d'objets de grande taille.

Définition de la correspondance de gestion de versions

Utilise la gestion des version pour procéder à un verrouillage optimiste. Si vous souhaitez utiliser ce type de fonctionnalité, vous devez définir un attribut persistant comme attribut Version, en sélectionnant l'option Attribut de version sur l'onglet EJB 3 Persistence. Les types suivants sont pris en charge pour Attribut de version : int, Integer, short, Short, long, Long, Timestamp.

L'attribut Version doit être mis en correspondance avec la table principale pour la classe d'entité. Les applications qui mettent en correspondance la propriété Version avec une table autre que la table principale ne seront pas portables. Un seul attribut Version doit être définir pour chaque classe Entity.

L'annotation Version est générée afin de spécifier l'attribut ou la propriété de version d'une classe d'entité qui est utilisé comme sa valeur de verrouillage optimiste.

```
@Version
@Column(name="OPTLOCK")
protected int getVersionNum() { return versionNum; }
```

Définition de la correspondance de classe incorporable

Les classes incorporables sont des classes de type valeur simples. Suivez les instructions relatives à la définition des correspondances pour les classes de type valeur afin de définir une correspondance de classe Embeddable pour EJB 3.

Dans EJB 3, les classes Embeddable (incorporables) contiennent uniquement des correspondances d'attribut, et ces attributs persistants peuvent avoir uniquement des types de base, i.e. les classes incorporables ne peuvent pas contenir d'autres classes incorporables.

Remarque : la classe Embeddable doit mettre en oeuvre l'interface java.io.Serializable et rédéfinit les méthodes equals() et hashCode().

L'annotation Embeddable est générée pour spécifier une classe dont les instances sont stockées comme faisant partie intrinsèquement de l'entité propriétaire et partagent l'identité de cette entité.

```
@Embeddable
public class Address implements java.io.Serializable {
    @Basic(optional=true)
    @Column(name="address_country")
    public String getCountry() {}
    .....
}
```

Définition des correspondances d'association EJB 3

La persistance EJB 3 permet la prise en charge de la plupart des stratégies de correspondance d'association décrites dans le chapitre Modélisation des correspondances O/R. Ne sont traitées dans cette section que les différences de comportement.

Une association doit être définie entre deux classes Entity ou entre une classe Entity et une superclasse mise en correspondance avant de pouvoir elle-même être mise en correspondance. Une correspondance d'association ayant une superclasse mise en correspondance comme cible sera ignorée. Les classes incorporables peuvent être la source ou la cible des associations.

La correspondance d'une association avec une classe d'association n'est pas prise en charge. Vous devez séparer les différents types d'associations en deux associations équivalentes.

Pour plus d'informations sur les correspondances, voir *Chapitre 19, Modélisation des correspondances objet/relationnel (O/R)* à la page 507.

Mise en correspondances des associations un-un

La persistance EJB 3 prend en charge à la fois la correspondance d'association bidirectionnelle un-un et la correspondance d'association unidirectionnelle un-un.

L'annotation OneToOne est générée pour définir une association a une seule valeur à une autre entité dotée d'une multiplicité un-un. Dans le cas des associations bidirectionnelles un-un, les annotations générées se présenteront comme suit :

```
@OneToOne(cascade=CascadeType.ALL, fetch=FetchType.EAGER)
@JoinColumns({
  @JoinColumn(name="aid", referencedColumnName="aid")
})
public Account getAccount() { ... }

@OneToOne(cascade=CascadeType.PERSIST, mappedBy="account")
public Person getPerson() { ... }
```

Les annotations générées pour les associations unidirectionnelles un-un sont similaires. Une vérification de modèle permet de s'assurer que les correspondances sont correctement définies pour les associations unidirectionnelles un-un. Une association unidirectionnelle ne peut être mise en correspondance qu'avec une référence qui a la même direction que l'association.

Pour plus d'informations sur les correspondances, voir *Chapitre 19, Modélisation des correspondances objet/relationnel (O/R)* à la page 507.

Mise en correspondances des associations un-plusieurs

La persistance EJB 3 prend en charge la correspondance d'association bidirectionnelle un-plusieurs, la correspondance d'association unidirectionnelle un-un, et la correspondance d'association unidirectionnelle un-plusieurs.

Une annotation OneToMany est générée pour définir une association multivaleurs avec une multiplicité un-plusieurs. Une annotation ManyToOne est générée afin de définir une association à une seule valeur avec une autre classe d'entité qui a une multiplicité plusieurs-un. L'annotation JoinColumn est générée pour spécifier une colonne de jointure pour la référence qui associe les tables. Dans le cas des associations bidirectionnelles un-plusieurs, les annotations générées se présenteront comme suit :

```
@OneToMany(fetch=FetchType.EAGER, mappedBy="customer")
public java.util.Collection<Order> getOrder() { ... }
```

```
@ManyToOne
@JoinColumns({
 @JoinColumn(name="cid", referencedColumnName="cid")
})
public Customer getCustomer() { ... }
```

Les annotations générées pour les associations unidirectionnelles plusieurs-un sont similaires. Une vérification de modèle permet de s'assurer que les correspondances pour les associations un-plusieurs et les associations unidirectionnelles plusieurs-un sont correctement définies. Les références peuvent uniquement naviguer depuis les tables principales des classes du côté multivaleur vers les tables principales de classes du côté à une seule valeur.

Pour une association unidirectionnelle un-plusieurs, l'association JoinTable est générée afin de définir une table intermédiaire et des colonnes de jointures pour les deux clés de référence.

```
@OneToMany(fetch=FetchType.EAGER)
@JoinTable(
 name="Customer_Order",
 joinColumns={
  @JoinColumn(name="cid", referencedColumnName="cid")
 },
 inverseJoinColumns={
  @JoinColumn(name="oid",
    referencedColumnName="orderId")
 }
)
public java.util.Collection<Order> getOrder() { ... }
```

Une vérification de modèle est disponible afin de vous assurer que les correspondances pour les associations unidirectionnelles un-plusieurs sont correctement définies. Les tables intermédiaires sont nécessaires pour ce type d'associations un-plusieurs.

Les associations un-plusieurs dans lesquelles la clé primaire est migrée ne sont pas prises en charge dans EJB 3.

Pour plus d'informations sur les correspondances, voir *Chapitre 19, Modélisation des correspondances objet/relationnel (O/R)* à la page 507.

Mise en correspondances des associations plusieurs-plusieurs

La persistance EJB 3 prend en charge EJB 3 à la fois la correspondances d'association bidirectionnelle plusieurs-plusieurs et la correspondances d'association unidirectionnelle plusieurs-plusieurs.

Une annotation ManyToMany est générée afin de définir une association multivaleur avec une multiplicité plusieurs-plusieurs.

```
@ManyToMany(fetch=FetchType.EAGER)
@JoinTable(
  name="Assignment",
  joinColumns={
   @JoinColumn(name="eid", referencedColumnName="eid")
  },
  inverseJoinColumns={
   @JoinColumn(name="tid", referencedColumnName="tid")
  }
)
public java.util.Collection<Title> getTitle() { ... }
```

Une vérification de modèle permet de s'assurer que les correspondances sont correctement définies pour les associations plusieurs-plusieurs. Les tables intermédiaires sont nécessaires pour les correspondances d'association plusieurs-plusieurs.

Pour plus d'informations sur les correspondances, voir *Chapitre 19, Modélisation des correspondances objet/relationnel (O/R)* à la page 507.

Définition des options de correspondance d'association EJB 3

Les options spécifiques à EJB 3 suivantes pour les correspondances d'association sont disponibles sur l'onglet EJB 3 Persistence de la feuille de propriétés d'une association :

Champ	Description
Inverse side	Spécifie quel côté est le côté opposé.
Rôle A - cascade	Spécifie quelle opération doit être effectuée en cascade du côté du rôle A.
Rôle B - cascade	Spécifie quelle opération doit être effectuée en cascade du côté du rôle B.
Rôle A - fetch	Spécifie si les instances du rôle A doivent être chargées à la demande ou forcées.
Rôle B - fetch	Spécifie si les instances du rôle B doivent être chargées à la demande ou forcées.
Rôle A order by	Spécifie l'ordre des éléments du côté du rôle A lorsque ce dernier est récupéré.

Champ	Description
Rôle A - order by	Spécifie l'ordre des éléments du côté du rôle B lorsque ce dernier est récupéré.

Définition des correspondances d'héritage EJB 3

La persistance EJB 3 prend en charge les trois stratégies de correspondances d'héritage populaires, ainsi que des stratégies mixtes.

- Une table par hiérarchie de classes - SINGLE_TABLE
- Sous-classe jointe - JOINED
- Une table par classe concrète - TABLE_PER_CLASS

Toutes les classes dans la hiérarchie de classes doivent être des classes d'entité ou des superclasses mises en correspondance. Pour chaque hiérarchie de classes, l'identifiant primaire doit être défini sur la classe d'entité qui constitue la racine de la hiérarchie, ou sur une superclasses mise en correspondance de la hiérarchie.

Vous avez également la possibilité de définir un attribut Version sur l'entité qui est la racine de la hiérarchie d'entités ou sur une superclasses mise en correspondances de la hiérarchie d'entités.

Superclasses mises en correspondances

Dans EJB 3.0, les superclasses mises en correspondances sont utilisées pour définir les informations d'état et de mise en correspondances qui sont communes à plusieurs classes d'entité. Elles ne sont pas mises en correspondance chacune avec leur propre table. Vous ne pouvez pas définir des superclasses mises en correspondances dans PowerAMC.

Stratégie "Une table par hiérarchie de classes"

Dans cette stratégie, la totalité de la hiérarchie de classes est mise en correspondance avec une table. Vous avez également la possibilité de définir des valeurs discriminantes pour chaque classe d'entité dans la hiérarchie en utilisant l'onglet EJB 3 Persistence de la feuille de propriétés de classe.

Option	Description
Valeur discriminante	Spécifie une valeur qui distingue cette classe des autres classes.

L'annotation Inheritance avec une stratégie SINGLE_TABLE est générée. L'annotation DiscriminatorColumn est générée pour définir la colonne discriminante. L'annotation DiscriminatorValue est générée pour spécifier la valeur de la colonne discriminante pour les entités du type donné si vous la spécifiez pour la classe.

Pour plus d'informations sur les correspondances, voir *Chapitre 19, Modélisation des correspondances objet/relationnel (O/R)* à la page 507.

```
@Entity(name="Shape")
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
```

```
@DiscriminatorColumn(name="shapeType",
discriminatorType=DiscriminatorType.STRING, length=100)
@Table(name="Shape")
public class Shape { ... }

@Entity(name="Rectangle")
@Inheritance(strategy=InheritanceType.SINGLE_TABLE)
@DiscriminatorValue("Rectangle")
@Table(name="Shape")
public class Rectangle extends Shape { ... }
```

Une vérification de modèle est disponible pour s'assurer que les colonnes discriminantes sont correctement définies.

Stratégie de sous-classe jointe

Dans cette stratégie, chaque classe est mise en correspondance avec sa propre table primaire. Les tables primaires des classes enfant ont des clés de référence qui pointent vers les tables primaires des classes parent.

Une annotation Inheritance avec la stratégie JOINED est générée. L'annotation PrimaryKeyJoinColumn est générée pour définir une colonne de jointure qui joint la table primaire d'une sous-classe d'entité à la table primaire de sa superclasse.

Pour plus d'informations sur les correspondances, voir *Chapitre 19, Modélisation des correspondances objet/relationnel (O/R)* à la page 507.

```
@Entity(name="Shape")
@Inheritance(strategy=InheritanceType.JOINED)
@DiscriminatorColumn(name="shapeType")
@Table(name="Shape")
public class Shape { ... }

@Entity(name="Rectangle")
@Inheritance(strategy=InheritanceType.JOINED)
@PrimaryKeyJoinColumns({
 @PrimaryKeyJoinColumn(name="sid", referencedColumnName="sid")
})
@Table(name="Rectangle")
public class Rectangle extends Shape { ... }
```

Une vérification de modèle est disponible pour s'assurer que les tables primaires des classes enfant ont des références de clé pointant vers les tables primaires de leurs classes parent.

Application d'une stratégie "une table par classe"

Dans cette stratégie, chaque classe est mise en correspondance avec une table distincte. Lorsque vous transformez un MOO en MPD, PowerAMC ne génère des tables que pour les classes racine, et suppose que toutes les autres classes ne sont pas mises en correspondance avec une table et ce, même si vous définissez manuellement des correspondances supplémentaires. Les annotations MappedSuperclass sont générées pour ces classes, et l'annotation Inheritance ne sera pas générée pour toutes les classes. Vous devez personnaliser

les annotations générées et créer des tables supplémentaires si vous souhaitez mettre en correspondances des classes autres que les classes racine avec des tables.

```
@MappedSuperclass
public class Shape { .. }

@Entity(name="Rectangle")
@Table(name="Rectangle")
public class Rectangle extends Shape { ... }
```

Définition des options par défaut pour la persistance EJB 3

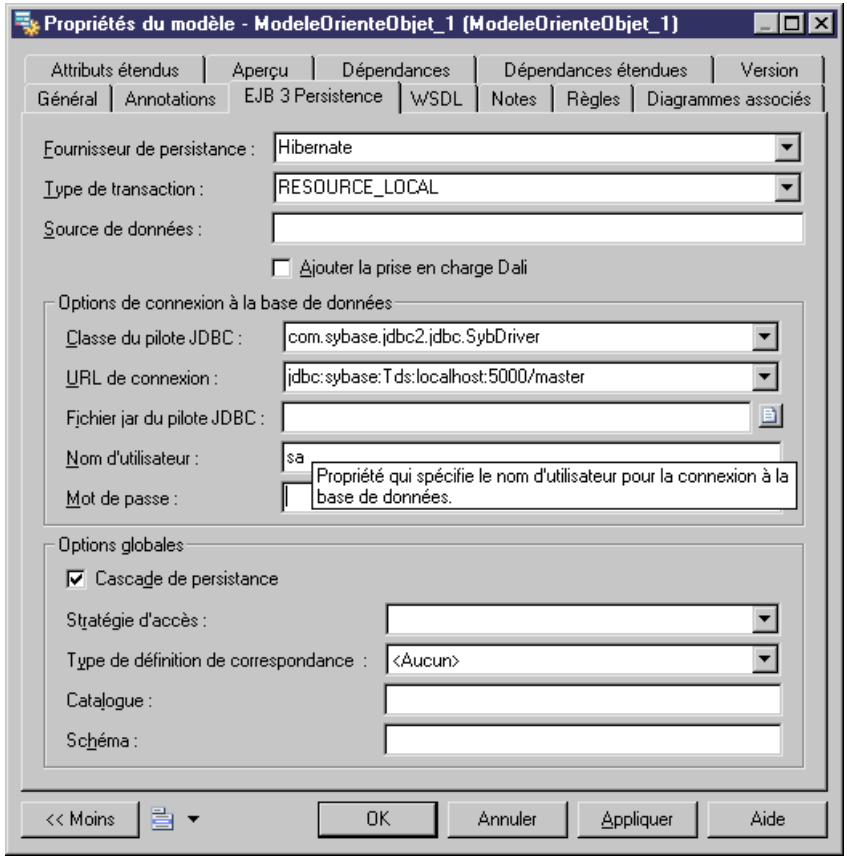
Les options de persistance par défaut suivantes peuvent être générées au niveau modèle, package ou classe :

Option	Description
Accès par défaut	Spécifie une stratégie d'accès.
Type de définition de correspondance	Spécifie le niveau de métadonnées de correspondance à générer.
Nom de catalogue	Spécifie le nom de catalogue pour les classes persistantes.
Nom de schéma	Spécifie le nom de schéma pour les classes persistantes.

Définition de la configuration de persistance EJB 3

Certaines propriétés de persistance sont utilisées pour la connexion à une base de données. Vous devez les définir avant de lancer l'application générée.

1. Affichez la feuille de propriétés d'un modèle, puis cliquez sur l'onglet EJB 3 Persistence.



2. Sélectionnez le fournisseur de persistance que vous souhaitez utiliser. Le choix d'un fournisseur implique des contraintes spécifiques, reportez-vous à la documentation appropriée.
3. Définissez la classe de pilote JDBC, l'URL de connexion, le chemin d'accès du fichier jar de pilote JDBC, le nom d'utilisateur et le mot de passe.

Option	Description
Fournisseur de persistance	Spécifie le fournisseur de persistance à utiliser.
Type de transaction	Spécifie le type de transaction à utiliser.
Source de données	Spécifie le nom de la source de données (si une source de données est utilisée).
Ajouter la prise en charge Dali	Spécifie que le projet généré peut être édité dans Dali.

Option	Description
Classe de pilote JDBC	Spécifie la classe de pilote JDBC.
URL de connexion	Spécifie la chaîne d'URL de connexion JDBC.
Fichier jar de pilote JDBC	Spécifie le chemin d'accès du fichier jar de pilote JDBC.
Nom d'utilisateur	Spécifie le nom d'utilisateur de base de données.
Mot de passe	Spécifie le mot de passe d'utilisateur de base de données.
Cascade de persistance	Spécifie si le style de cascade doit être réglé à PERSIST pour toutes les relations dans l'unité persistante.

Vous pouvez vérifier l'effet des paramètres de configuration dans l'onglet Aperçu. Le fichier de configuration de persistance généré se présente comme suit :

```
<persistence xmlns="http://java.oracle.com/xml/ns/persistence"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://java.oracle.com/xml/ns/persistence
 http://java.oracle.com/xml/ns/persistence/persistence_1_0.xsd"
 version="1.0">
 <persistence-unit name="EJB3_0Model" transaction-
type="RESOURCE_LOCAL">
  <description>
  This is auto generated configuration for persistent unit
EJB3_0Model
  </description>
<provider>org.hibernate.ejb.HibernatePersistence</provider>
  <!-- mapped files -->
  <!--jar-file/-->
  <!-- mapped classes -->
  <class>com.company.orders.Customer</class>
  <class>com.company.orders.Order</class>
  <properties>
   <property
name="hibernate.dialect">org.hibernate.dialect.SybaseDialect</
property>
   <property
name="hibernate.connection.driver_class">com.sybase.jdbc2.jdbc.SybD
river</property>
   <property
name="hibernate.connection.url">jdbc:sybase:Tds:localhost:5000/
Production</property>
   <property name="hibernate.connection.username">sa</property>
   <property name="hibernate.connection.password"></property>
  </properties>
 </persistence-unit>
</persistence>
```

Vérification du modèle

Pour s'assurer de la validité et de la cohérence du modèle et des correspondances, vous devez procéder à une vérification de modèle. Si des erreurs sont détectées, vous devez les corriger. Vous pouvez aussi lancer une vérification de modèle avant de procéder à la génération de code.

Génération de code pour la persistance EJB 3

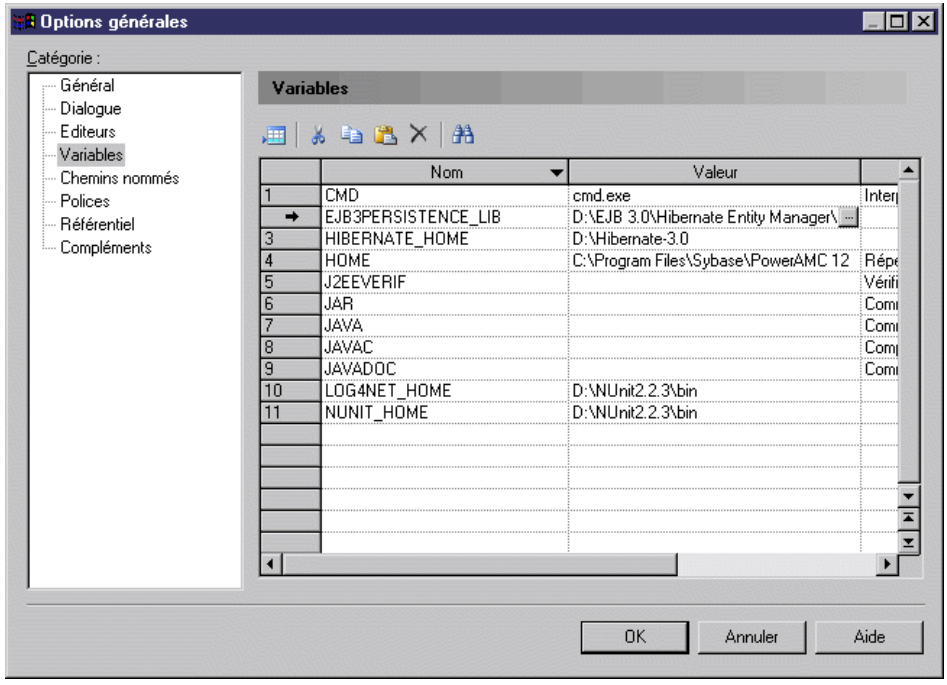
Pour pouvoir générer du code pour la persistance EJB 3, vous devez :

- Télécharger un fournisseur de persistance EJB 3 tel que Hibernate Entity Manager, Kodo, TopLink and GlassFish.
- Définir une variable d'environnement afin de spécifier l'emplacement du répertoire de bibliothèque de persistance.
- Générer du code - définir des options de sélection dans le modèle et de génération et afficher un aperçu des fichiers générés.
- Exécuter des tests unitaires.

Définition de la variable d'environnement

PowerAMC une variable d'environnement pour générer la configuration de bibliothèque pour une projet Eclipse ou un fichier de script de compilation.

1. Sélectionnez **Outils > Options générales**.
2. Sélectionnez le noeud Variables.



3. Ajoutez une variable EJB3PERSISTENCE_LIB et, dans la colonne Valeur, spécifiez le chemin d'accès du répertoire dans lequel vous placez vos bibliothèques de persistance, par exemple D:\EJB 3.0\Hibernate Entity Manager\lib.

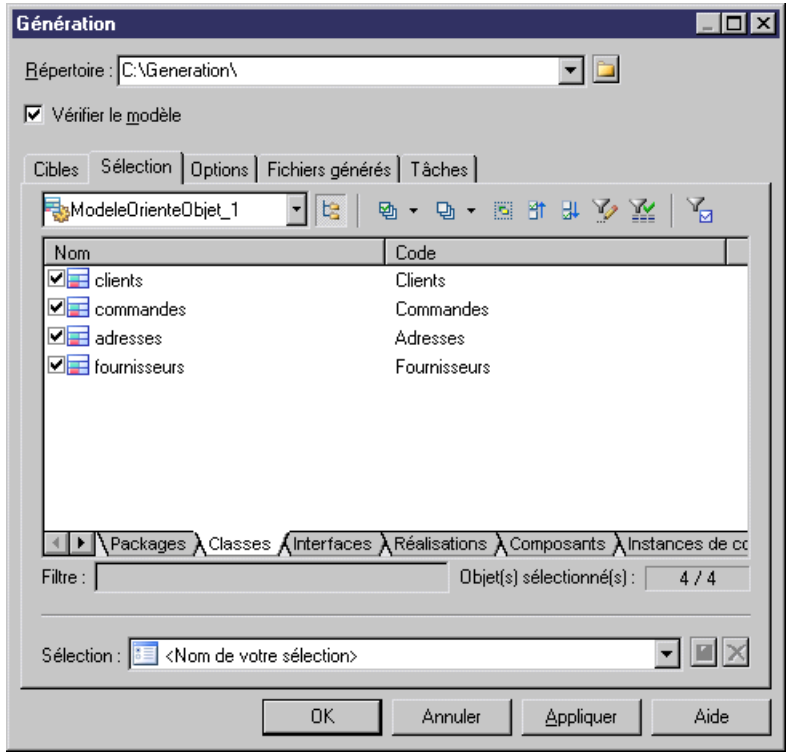
Vous pouvez également définir cette variable comme variable d'environnement système Windows, mais vous devez alors redémarrer PowerAMC pour que ce changement soit prise en compte.

Génération de code

Pour générer du code, sélectionnez **Langage > Générer du code Java** (Ctrl + G). Spécifiez le répertoire dans lequel vous souhaitez générer le code. Sur l'onglet Cibles, assurez-vous que la case Correspondance O/R est cochée.

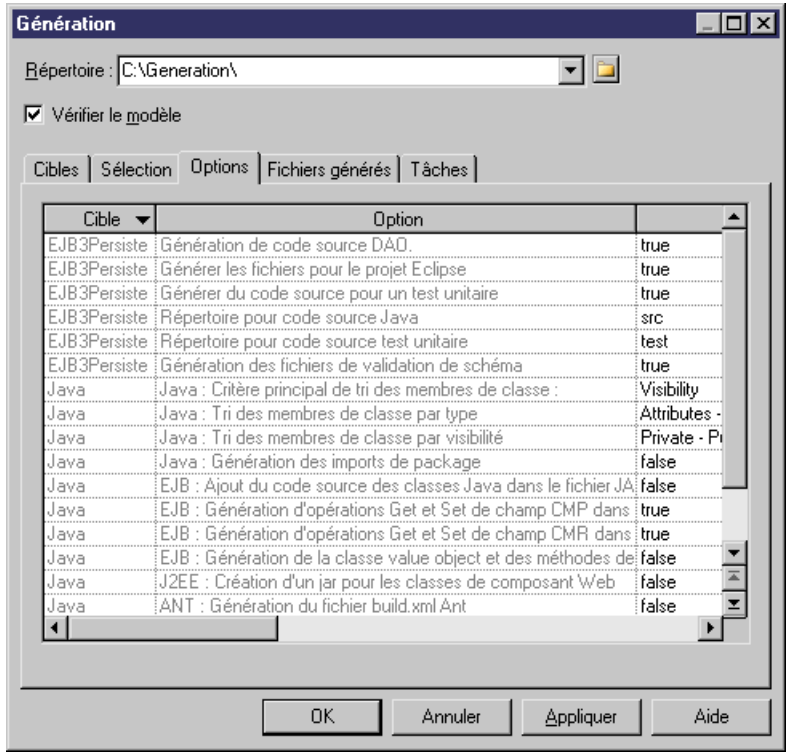
Sélection des éléments de modèle

Sélectionnez les éléments de modèle à générer sur l'onglet Sélection. Le modèle doit être sélectionné car certains artefacts importants sont générés au niveau modèle, tels que le fichier de configuration de persistance, les classes de DAO factory, les classes DAO base, etc.



Définition des options de génération

- Définissez les options de génération sur l'onglet Options.



Option	Description
Génération de code source DAO.	Spécifie si le code source DAO doit être généré.
Générer les fichiers pour le projet Eclipse	Spécifie si le fichier de projet Eclipse et le fichier classpath doivent être générés.
Générer du code source pour un test unitaire	Spécifie si le code source des tests unitaires doit être généré.
Répertoire pour code source Java	Spécifie le répertoire pour les sources Java.
Répertoire pour code source test unitaire	Spécifie le répertoire pour les sources de test unitaire.
Génération des fichiers de validation de schéma	Spécifie si le fichier de schéma et le script de validation doivent être générés.
Génération du fichier Ant build.xml	Spécifie si le fichier Ant build.xml doit être généré.

Affichage d'un aperçu de la liste des fichiers générés

Vous pouvez afficher un aperçu de la liste des fichiers générés sur l'onglet Fichiers générés.

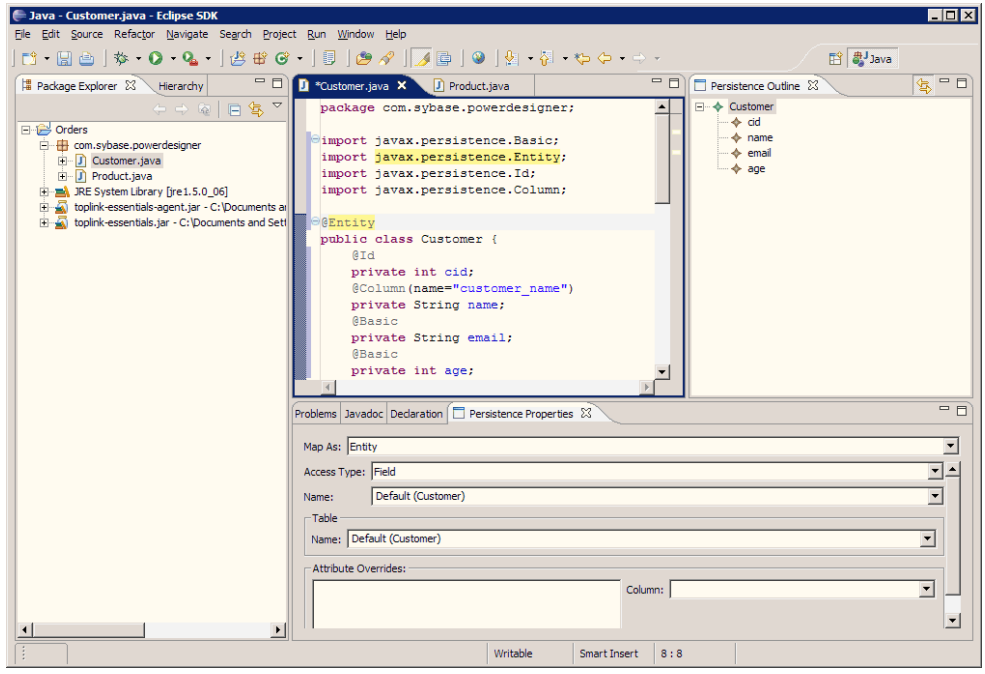
Spécification des tâches post-génération
Vous pouvez spécifier des tâches à effectuer à l'issue de la génération. Sélectionnez ces tâches sur l'onglet Tâches. La tâche la plus utile est la tâche Exécution des tests unitaires générés. Si vous avez sélectionné cette tâche, PowerAMC va exécuter les tests unitaires par script Ant à l'issue de la génération. Vous pouvez également les lancer via une ligne de commande. Il existe des prérequis pour l'exécution de cette tâche. La section suivante traite de l'exécution des tests unitaires.

Edition à l'aide des outils Dali

Les outils Dali JPA permettent de définir, d'éditer et de déployer des correspondances O/R pour les beans d'entité EJB 3.0. La définition et l'édition des correspondances est simplifiée par le biais des fonctionnalités suivantes :

- Assistants de création et d'édition de correspondances
- Assistance intelligente à la définition des correspondances
- Identification dynamique des problèmes

Vous pouvez importer le projet Eclipse généré et l'éditer dans les outils Dali si vous aviez coché la case Ajouter la prise en charge de Dali dans la feuille de propriétés du modèle.



Exécution des tests unitaires

Il existe de nombreuses façons d'exécuter des tests unitaires générés. L'un de ces méthodes consiste à utiliser une tâche Ant, mais vous pouvez également les exécuter dans Eclipse.

Exécution des tests unitaires à l'aide de Ant

Pour générer le fichier build.xml Ant, vous devez sélectionner l'option Génération du fichier build.xml Ant dans l'onglet Options de la boîte de dialogue de génération.

Pour utiliser Ant, vous devez :

- Télécharger Ant depuis le site *http://www.apache.org*, et installez-le.
- Définir une variable d'environnement ANT_HOME et la faire pointer vers le répertoire où vous avez installé Ant.
- Télécharger junit-3.8.1.jar si vous ne disposez pas déjà de ce fichier.
- Copier junit-3.8.1.jar dans le répertoire $ANT_HOME/lib.
- Vérifier que vous avez correctement défini les paramètres de connexion à la base de données et le fichier jar de pilote JDBC.

Exécution de tests unitaires avec Ant depuis PowerAMC

Vous pouvez exécuter des tests unitaires avec Ant depuis PowerAMC.
Sélectionnez la tâche Exécution des tests unitaires générés lorsque vous générez du code.

Exécution de tests unitaires depuis une ligne de commande Ant

Vous pouvez exécuter des tests unitaires depuis une ligne de commande Ant.

1. Affichez une fenêtre de ligne de commande.
2. Passez dans le répertoire dans lequel vous avez généré le code.
3. Exécutez la tâche de test JUnit en tapant la commande suivante : Ant junit
4. Vérifiez le résultats dans ejb3-persistence.log et dans le répertoire testout.

Exécution des tests unitaires dans Eclipse

Pour pouvoir utiliser Eclipse, vous devez télécharger et installer le SDK Eclipse.

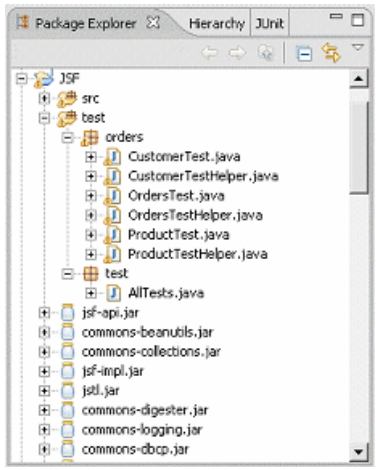
Si vous avez sélectionné l'option Générer les fichiers pour le projet Eclipse, vous pouvez importer le projet généré dans Eclipse puis utiliser Eclipse pour modifier, compiler et exécuter les tests.

Si vous utilisez le plugin PowerAMC pour Eclipse, à l'issue de la génération du code, le projet est automatiquement importé ou réactualisé dans Eclipse.

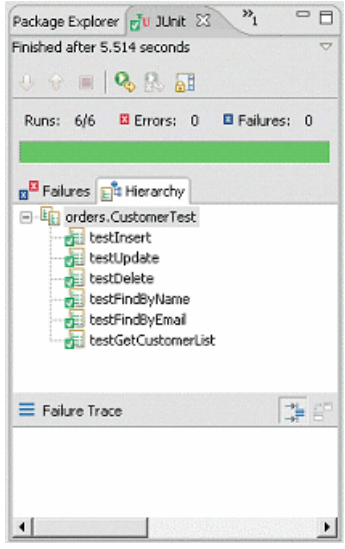
Vous ne pouvez exécuter qu'un seul test à chaque fois ou les exécuter les uns à la suite des autres.

Exécution d'un seul cas de test
Vous pouvez exécuter un seul cas de test.

1. Ouvrez la perspective Java.
2. Dans le Package Navigator, développez le package de test.



3. Sélectionnez une classe de cas de test, par exemple CustomerTest.java, puis exécutez-la comme test unitaire
4. Sélectionnez la vue JUnit pour vérifier le résultat :



Exécution de la suite de tests
Vous pouvez exécuter la suite de tests.

1. Sélectionnez la classe AllTests sous le package de test.
2. Exécutez-la comme application. Tous les tests sont exécutés les uns après les autres.

Liste des fichiers générés

Les fichiers suivants sont générés :

- Fichiers de projets Eclipse - Si vous avez sélectionné l'option de génération Générer les fichiers pour le projet Eclipse, le fichier .project et le fichier .classpath sont générés par PowerAMC. En revanche, si vous régénérez le code, ces deux fichiers ne sont pas à nouveau générés.
- Classes Java persistantes - Si le type de définition de correspondance spécifié inclut Annotation, Default, Annotation ou Mapping File & Annotation, les annotations seront générées dans des sources Java.
- Classes de clé primaire - Les classes de clé primaire sont générées pour faciliter le fonctionnement des recherches par clé primaire, elles sont également obligatoires dans le cas d'une clé primaire composite.
- Fichier de configuration EJB 3 - Le fichier de configuration de persistance EJB 3 persistence.xml est généré dans le sous-répertoire META-INFO d'un répertoire source Java.
- Fichier de configuration Log4J - PowerAMC utilise Log4j comme cadre de consignation par défaut pour la consignation des messages. Le fichier de propriétés log4j.properties est généré dans le répertoire source Java.
- Classe d'utilitaire - La classe Util.java contient certaines fonctionnalités utilitaires qui sont utilisées par les tests unitaires, telles que la comparaison des dates par précision. Elle est définie dans le package com.sybase.orm.util.
- Fichiers de correspondance EJB 3 O/R - Si le type de définition de correspondance spécifié inclut le Mapping file, Mapping File & Annotation ou Mapping file, les fichiers de correspondance O/R EJB 3 seront générés. Ces fichiers de correspondance sont générés dans le même répertoire que la source Java.
- Factory et Data Access Objects - Aident à simplifier le développement de votre application, PowerAMC génère DAO Factory et Data Access Objects (DAO), en utilisant les motifs de conception Factory et DAO.
- Classes de test unitaire - générées pour aider l'utilisateur à effectuer une afin de s'assurer que :
  - Les correspondances sont correctement définies
  - La matrice CLMS (Création, Lecture, Modification et Suppression) fonctionnera correctement
  - Les méthodes de recherche fonctionnent
  - Les navigations fonctionnent

  Les classes de test unitaire incluent :

    - Classes d'aide au tests - contient certaines fonctionnalités utilitaires pour les classes de test unitaire, telles que la création de nouvelles instances, la modification de l'état des instances, l'enregistrement des instances etc.
    - Classes de test unitaire - Pour chaque entité persistante, PowerAMC génère une classe de test unitaire. Les classes de test générées sont les suivantes :
        - Insert test method - pour tester l'insertion d'instance.
        - Update test method - pour tester la modification d'instance.
        - Delete test method - pour tester la suppression d'instance.
        - Property finder test methods - pour tester chaque méthode property finder définie dans Dao.
        - Get all instance list test method - pour tester la méthode de récupération de toutes les instances.
        - Navigation test method - pour tester la correspondance d'association.
        - Inheritance test method - pour tester la correspondance d'héritage.
        - User defined operation test methods - méthodes de test squelette pour des fonctions utilisateur.
- AllTest Class - suite de tests qui exécute toutes les cas de test.
- Fichier de compilation Ant - PowerAMC peut générer un fichier de compilation Ant afin de vous aider à compiler et exécuter des tests unitaires si vous avez défini à true l'option de génération Génération du fichier build.xml Ant. Le fichier build.xml Ant contient des éléments personnalisés pour EJB 3 :
    - Propriétés personnalisées - qui spécifient les répertoires et chemins d'accès de classes.
    - Définitions de cibles personnalisées - pour définir des tâches JUnit.
    - Tâches personnalisées - pour exécuter un test JUnit test ou générer des rapports de test JUnit.

Génération de JavaServer Faces (JSF) pour Hibernate

JavaServer Faces (JSF) est un environnement d'interface utilisateur pour les applications Web en Java. JSF est conçu pour réduire de façon significative la charge de la rédaction et de la maintenance d'applications qui tournent sur un serveur d'applications Java et rendent leurs interfaces sur un client cible.

PowerAMC prend en charge JSF via un fichier d'extension qui fournit des attributs étendus JSF, des vérifications de modèle, des templates JSP, des templates de bean gérés par l'appelant ainsi que des templates de configuration de faces pour vos MOO Java.

Vous pouvez construire rapidement des applications Web sans avoir à écrire du code répétitif, en utilisant PowerAMC afin de générer automatiquement des classes persistantes, des DAO, des beans gérés, la navigations dans des pages, et des pages JSF en fonction de votre environnement persistant Hibernate ou EJB 3.0.

Pour activer JSF dans votre modèle, sélectionnez **Modèle > Extensions**, cliquez sur l'outil **Attacher une extension**, sélectionnez le fichier `JavaServer Faces (JSF)` (sur l'onglet **User Interface**), puis cliquez sur **OK** pour l'attacher.

Remarque : Notez que, dans la mesure où JSF utilise DAO (Data Access) pour accéder aux données provenant de la base de données, vous allez également devoir ajouter une extension de gestion de la persistance telle que Hibernate afin de pouvoir générer pour JSF.

La génération de JSF peut vous aider à tester des objets persistants en utilisant des pages Web avec vos propres données et peut également vous aider à gérer une application Web JSF par défaut. Vous pouvez utiliser un IDE afin d'améliorer les pages JSF générées pour en changer la disposition.

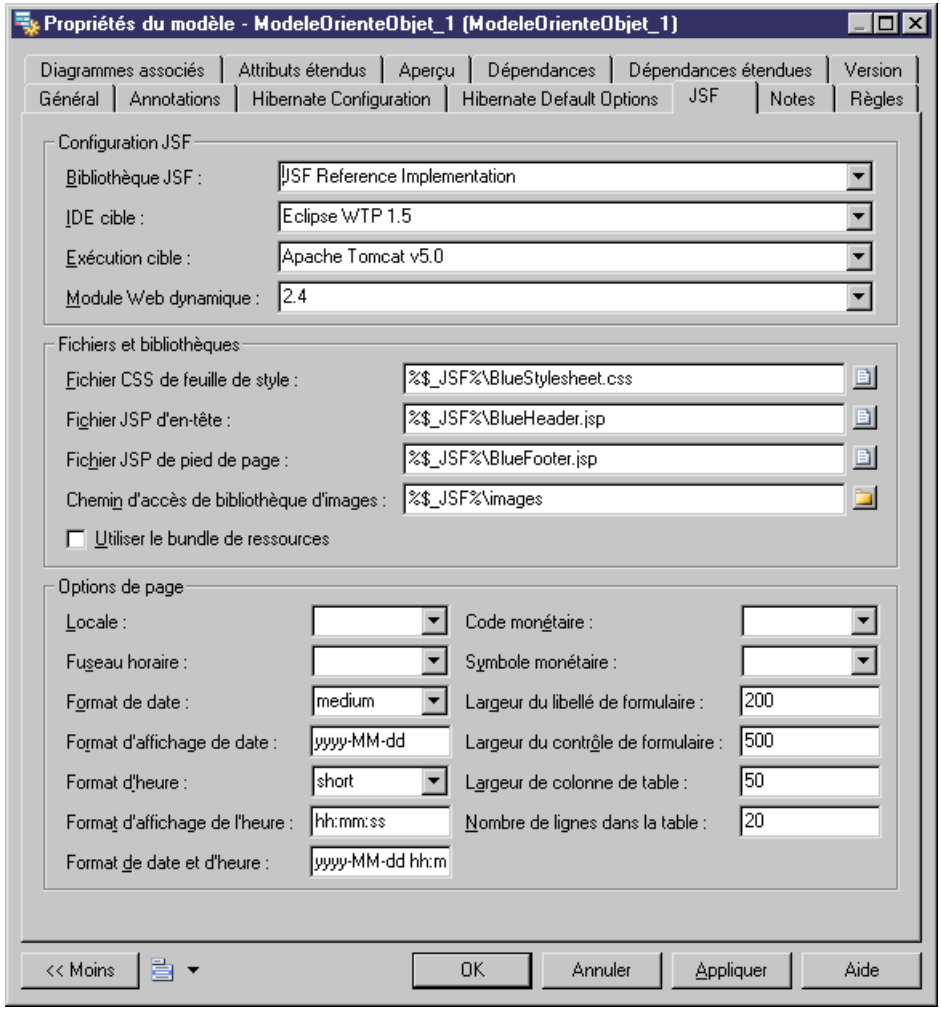
Définition des options globales

Chaque page doit utiliser une feuille de style, un fichier d'en-tête et un fichier de pied de page pour définir sa présentation standard.

PowerAMC fournit des fichiers de feuille de style, d'en-tête et pied de page par défaut.

Vous pouvez également définir des options globales par défaut telles que le format de données, le format d'heure, etc.

1. Affichez la feuille de propriétés de modèle, puis cliquez sur l'onglet JSF :



2. Définissez les fichiers de feuille de style, d'en-tête et de pied de page.
3. Définissez le répertoire dans lequel les images utilisées par la feuille de style, l'en-tête et le pied de page doivent être stockées.
4. Définissez le répertoire des fichiers JAR de bibliothèque JSF.
5. Définissez les options par défaut.

Les options suivantes sont disponibles :

Option	Description
Bibliothèque JSF	Spécifie la bibliothèque JSF. Il peut s'agir de JSF Reference Implementation ou de Apache My Faces.

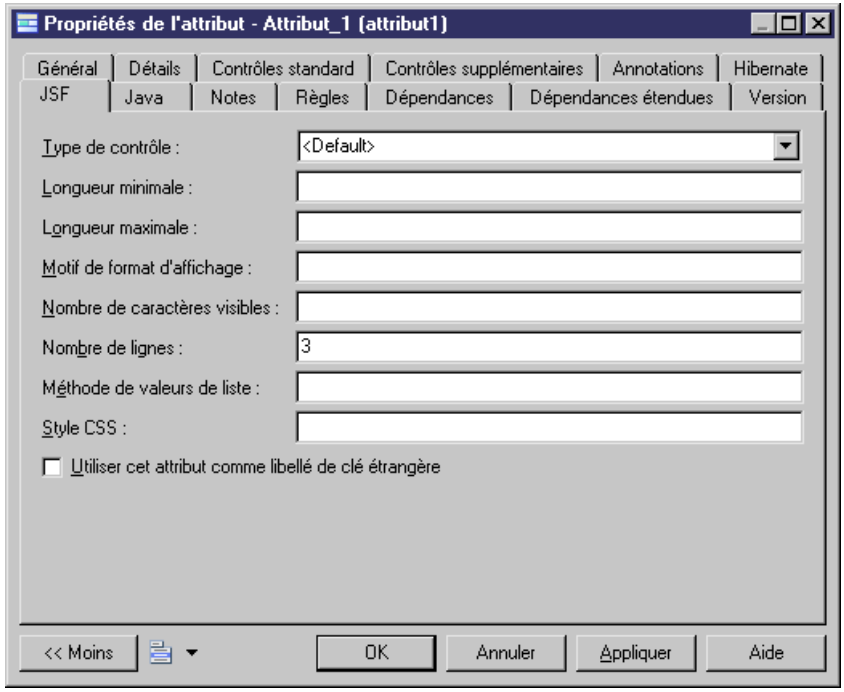
Option	Description
IDE cible	Spécifie l'IDE cible. Liste/Valeurs par défaut : Eclipse WTP, Sybase WorkSpace
Exécution cible	Spécifie l'exécution cible. Liste/Valeurs par défaut : Apache Tomcat v5.0, Sybase EAServer v5.x, etc.
Module Web dynamique	Spécifie le numéro de version du module Web dynamique List/Default Values: 2.2, 2.3, and 2.4.
Fichier CSS de feuille de style	Spécifie le fichier de feuille de style pour les pages JSF. Liste/Valeurs par défaut : %$_JSF%\stylesheet.css
Fichier JSP d'en-tête	Spécifie le fichier d'en-tête pour les pages JSF. Liste/Valeurs par défaut : %$_JSF%\header.jsp
Fichier JSP de pied de page	Spécifie le fichier de pied de page pour les pages JSF. Liste/Valeurs par défaut : %$_JSF%\footer.jsp
Chemin d'accès de bibliothèque d'images	Spécifie le chemin du répertoire qui contient les images pour les pages JSF. Liste/Valeurs par défaut : %$_JSF%\images
Utiliser le bundle de ressources	Spécifie l'utilisation d'un bundle de ressources.
Locale	Spécifie le pays
Fuseau horaire	Spécifie le fuseau horaire
Format de date	Spécifie le format de date. Les valeurs possibles sont default, short, medium, long, full Valeur par défaut : short
Format d'affichage de date	Spécifie le format d'affichage pour la date
Format d'heure	Spécifie le format de date. Les valeurs possibles sont default, short, medium, long, full Valeur par défaut : short
Format d'affichage d'heure	Spécifie le format d'heure
Format de date et heure	Spécifie le format de date et d'heure

Option	Description
Code monétaire	Spécifie le code monétaire
Symbole monétaire	Spécifie le symbole monétaire
Largeur du libellé de formulaire	Spécifie la largeur du libellé de contrôle (en pixels) dans un formulaire Valeur par défaut : 200
Largeur du contrôle de formulaire	Spécifie la largeur du contrôle (en pixels) dans un formulaire Valeur par défaut : 500
Largeur de colonne de table	Spécifie la largeur du contrôle (en pixels) dans une table Valeur par défaut : 50
Nombre de lignes dans la table	Spécifie le nombre de lignes pouvant être affichées dans une table Valeur par défaut : 20

Définition d'options relatives aux attributs

Vous pouvez définir des options de niveau attribut pour la validation ou le style de présentation.

1. Affichez une feuille de propriétés d'attributs, puis cliquez sur l'onglet JSF.
2. Définissez les options d'attribut appropriées.

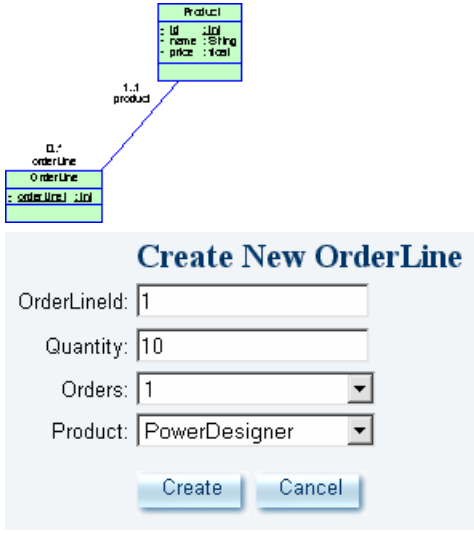


Option	Description
Type de contrôle	Spécifie le type de contrôle. Remarque : Vous devez sélectionner le type qui peut prendre en charge le type Java de l'attribut. • String - EditBox, MultilineEdit • Boolean - CheckBox • Date - Date, Time, DateTime, Year, Month, Day • <Contains List of Value> - ListBox, ComboBox, RadioButtons
Longueur minimale	Spécifie le nombre minimum de caractères
Longueur maximale	Spécifie le nombre maximum de caractères
Motif de format d'affichage	Spécifie le motif de format d'affichage pour l'attribut.
Nombre de caractères visibles	Spécifie le nombre de caractères visibles par ligne.

Option	Description
Nombre de lignes	Spécifie le nombre de lignes pour les zones d'édition multiligne Valeur par défaut : 3
Méthode de valeurs de liste	Spécifie la méthode qui fournit la liste des valeurs pour des contrôles List-Box, ComboBox ou RadioButtons.
Style CSS	Spécifie le style de format CSS
Utiliser cet attribut comme libellé de clé étrangère	Spécifie que la colonne associée à l'attribut sera utilisée comme libellé de clé étrangère pour la sélection de clé étrangère. Si aucun libellé de colonne de clé étrangère n'est défini, PowerAMC choisit la première colonne n'étant ni une colonne de clé étrangère ni une colonne de clé primaire et l'utilise comme colonne par défaut pour le libellé. Valeur par défaut : False

Remarque : Si la case "Utiliser cet attribut comme libellé de clé étrangère" n'est pas cochée et que la table courante contient une clé étrangère, PowerAMC génère par défaut une liste modifiable pour afficher l'ID de clé étrangère. Si vous souhaitez afficher la valeur d'une autre colonne (par exemple, le nom de produit au lieu de l'ID de produit), vous pouvez utiliser l'option "Utiliser cet attribut comme libellé de clé étrangère" pour l'attribut du nom de produit afin d'indiquer qu'il sera utilisé comme libellé de clé étrangère.

N'oubliez pas que si certains attributs spécifient le choix comme true, le libellé de clé étrangère sera généré que pour le premier attribut.



Attributs dérivés

Pour prendre en charge les attributs dérivés dans PowerAMC, vous pouvez :

1. Définir un attribut, et indiquer qu'il n'est pas persistant, et qu'il est dérivé.
2. Générer et mettre en oeuvre un getter.

Lorsque vous générez des pages, PowerAMC va automatiquement inclure des attributs dérivés.

Règles de validation et valeurs par défaut d'attribut

PowerAMC fournit une validation et des valeurs par défaut pour les zones d'édition dans les pages Create et Edit.

1. Affichez la feuille de propriétés d'un attribut, puis cliquez sur l'onglet Contrôles standard.
2. Vous pouvez définir des valeurs minimum et maximum pour contrôler la page de valeurs admise.
3. Vous pouvez définir une valeur par défaut. Une chaîne doit être placée entre apostrophes. Vous pouvez également définir la valeur initiale sur l'onglet Détails.
4. Vous pouvez définir une liste de valeurs qui seront utilisées dans une zone de ligne, dans une liste modifiable ou dans un jeu de boutons radio.

Remarque : Vous pouvez définir une "liste de valeurs" sur l'onglet Contrôles standard de la feuille de propriétés d'un attribut, PowerAMC génère alors une liste modifiable contenant ces valeurs. Dans le cas des règles de validation, vous pouvez définir le domaine personnalisé également, puis sélectionner le domaine que vous souhaitez appliquer dans l'attribut spécifié.

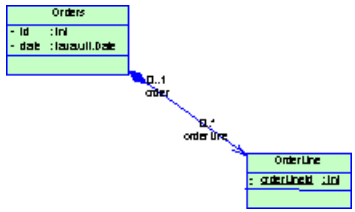
Vous pouvez également sélectionner le type de contrôle à utiliser :

- Combobox
- Listbox
- Radio buttons (s'il y a peu de valeurs). Par exemple, M., Mme.

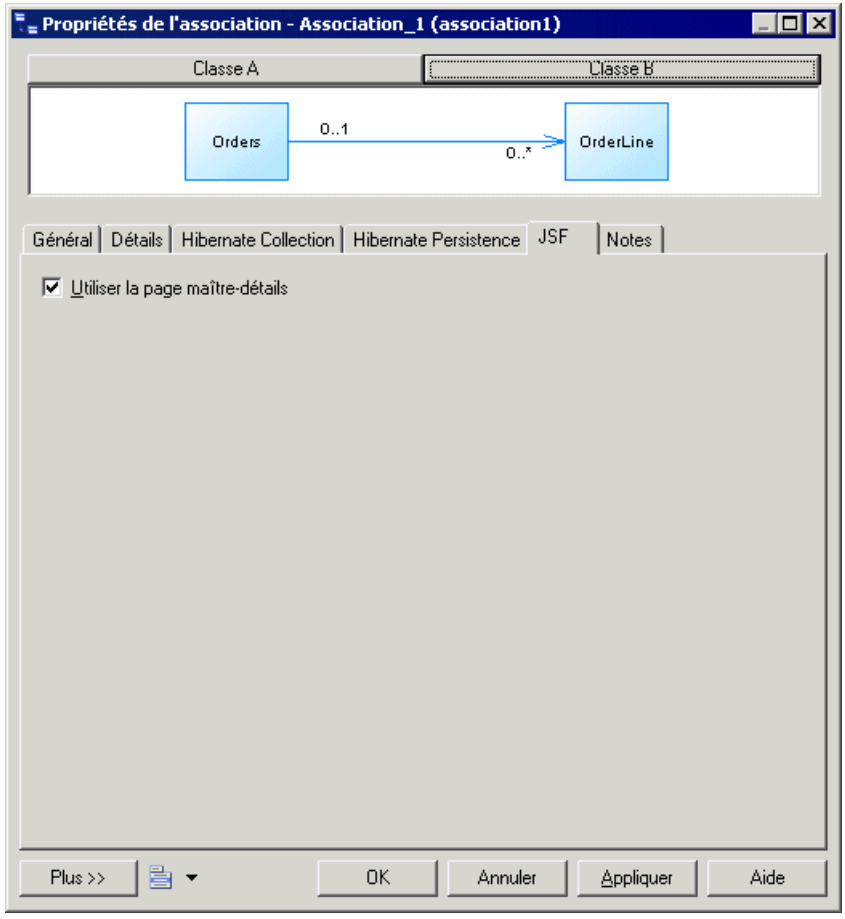
Définition de pages maître-détails

Si deux objets ont une relation maître-détails, PowerAMC les gère (méthode create, update, delete et find) sur la même page. Lorsque vous cliquez sur la colonne contenant le bouton de lien vers les détails dans la vue de la table maître, le contenu de la vue de la page détails dans la même page change de façon dynamique.

Par exemple, la table Orders (table maître) et la table Orderline (table détails). L'association est une composition. Si vous supprimez une commande (order), les articles de la commandes (order lines) doivent être supprimés. Ils seront affichés sur la même page :

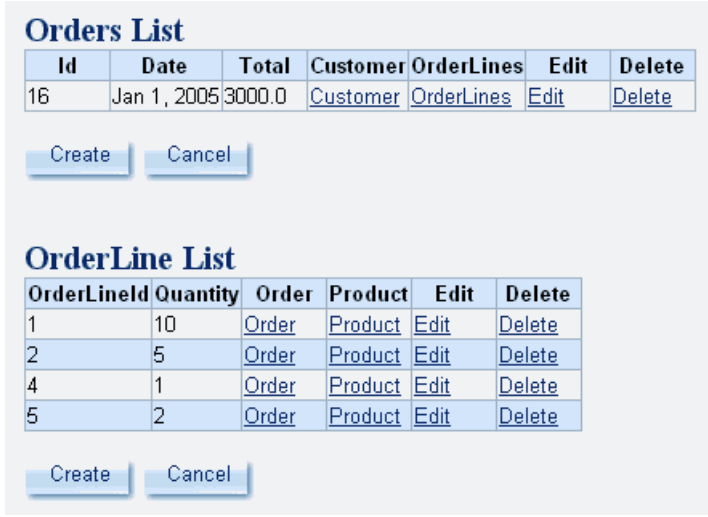


1. Créez une association un-plusieurs, dans laquelle la direction un-plusieurs est navigable.
2. Affichez la feuille de propriétés de l'association, et cliquez sur l'onglet JSF.
3. Cochez la case "Utiliser la page maître-détails" :



Le type d'association doit être Composition ou Agrégation, ce qui signifie que l'une des extrémités de l'association est une référence faible à la classe maître.

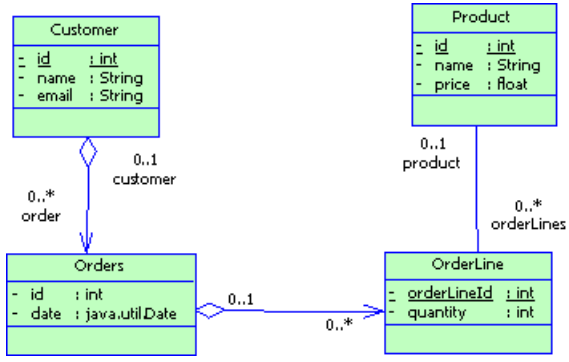
La page JSF maître-détails se présentera comme suit :



Génération de diagrammes de flux de pages

Dans Java Server Faces, un fichier de configuration (xml) est utilisé pour définir les règles de génération entre les différentes pages web, qui constituent le PageFlow (flux de pages). PowerAMC fournit un diagramme de flux de pages de haut niveau pour envisager les différents type de définition, et peut générer des règles de navigation pour l'application web JSF et le bean de page JSF en fonction du diagramme PageFlow.

Vous pouvez générer le diagramme PageFlow de haut niveau dans 3 niveaux : modèle, package et classe.

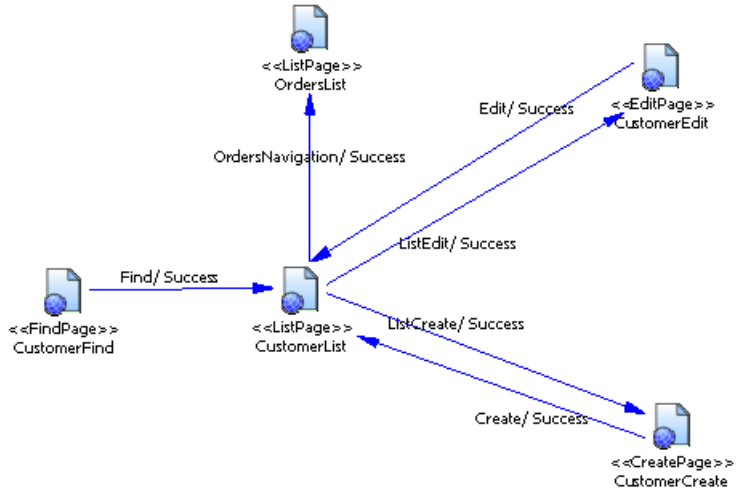


Génération d'un diagramme de flux de pages de niveau classe

Vous générez un diagramme de flux de pages de niveau classe de la façon suivante :

1. Sélectionnez une classe dans le diagramme de classes, par exemple Customer. Pointez sur la classe et cliquez le bouton droit de la souris et sélectionnez "Générer un diagramme de flux de pages" dans le menu contextuel.

2. Un nouveau PageFlow est automatiquement créé : CustomerPageFlow.



Génération d'un diagramme de flux de pages de niveau package

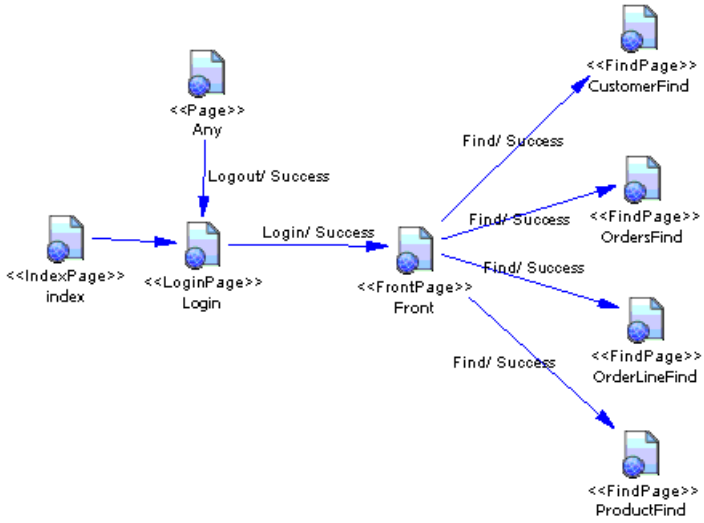
Vous généréz un diagramme de flux de pages de niveau package de la façon suivante :

1. Sélectionnez un package, par exemple "Orders", cliquez le bouton droit de la souris puis sélectionnez "Générer un diagramme de flux de pages" dans le menu contextuel.
2. Le PageFlow sera généré pour chaque classe située dans ce package et ses sous-packages : CustomerPageFlow, OrdersPageFlow, ProductPageFlow, OrderLinePageFlow.

Génération d'un diagramme de flux de pages de niveau modèle

Vous générez un diagramme de flux de pages de niveau modèle de la façon suivante :

1. Pointez sur un modèle, par exemple "JSF", cliquez le bouton droit, puis sélectionnez "Générer un diagramme de flux de pages" dans le menu contextuel.
2. Le PageFlow sera généré : "JSFPageFlow". Un PageFlow est également généré pour chaque classe située dans ce modèle et dans ses packages : CustomerPageFlow, OrdersPageFlow, ProductPageFlow, OrderLinePageFlow.



Modification du diagramme de flux de pages par défaut

Une fois que vous avez généré le diagramme de flux de pages, vous pouvez définir des pages et des flux de page personnalisés dans le flux de page de niveau classe.

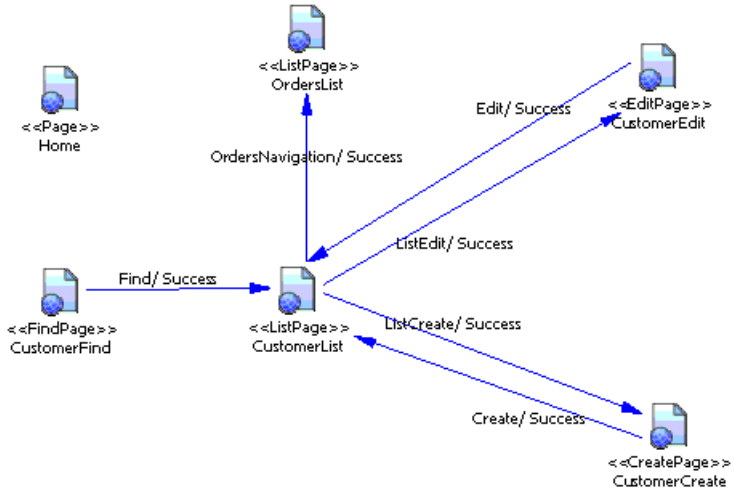
Toutes les pages dans le diagramme de flux de pages par défaut sont dotées d'un stéréotype prédéfini, par exemple le stéréotype pour CustomerFind est "FindPage", celui pour CustomerEdit est "EditPage", etc. Vous pouvez ajouter votre page personnalisée.

Vous pouvez également ajouter de nouveaux flux de page pour lier les pages dans le diagramme de flux de pages, ce qui revient à ajouter une transition dans un diagramme d'états-transitions.

Ajout d'une nouvelle page

Vous ajoutez une nouvelle page à partir de la Boîte à outils.

1. Sélectionnez l'outil **Etat** dans la Boîte à outils, puis cliquez dans le diagramme de flux de pages, vous créez ainsi une nouvelle page portant un nom par défaut.
2. Vous pouvez changer son nom et son stéréotype en Page dans la feuille de propriétés, par exemple changer son nom en "Home". Cette feuille de propriétés est identique à celle d'un état standard.

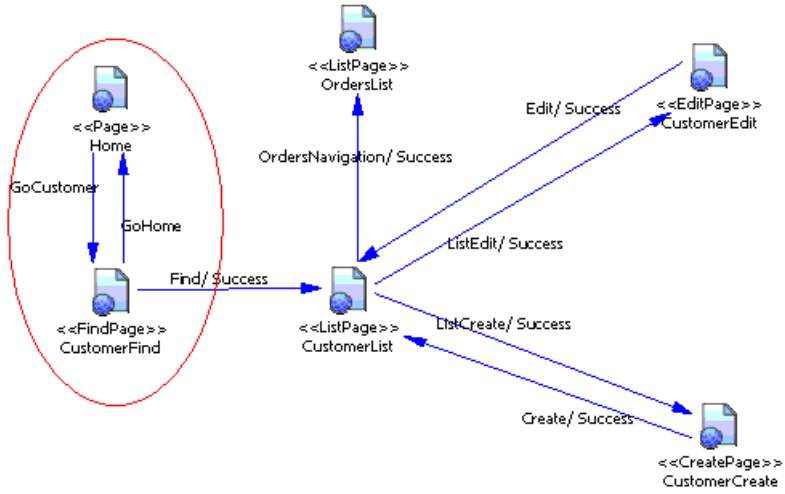


Après avoir créé une nouvelle page, puis la génération de code, une page JSF par défaut et son bean de page sont générés.

Ajout d'un nouveau flux de page

Vous ajoutez un nouveau flux de page à partir de la Boîte à outils.

1. Sélectionnez **Transition** dans Boîte à outils.
2. Tracez une transition entre l'état source, à savoir `Home`, et l'état cible, c'est-à-dire `CustomerFind`.
3. Affichez la feuille de propriétés de la transition, puis cliquez sur l'onglet **Déclencheur**.
4. Cliquez sur l'outil **Créer** en regard de la zone **Evénement déclencheur** pour créer un nouvel événement, saisissez le nom approprié pour l'événement, puis cliquez sur **OK** pour revenir à la feuille de propriétés de la transition.
5. Cliquez sur **OK** pour créer le nouveu flux de page :



Une fois que vous avez modifié le flux de pages par défaut, et généré les codes JSF, les pages JSF par défaut, les beans de page et le fichier faces-config correspondants sont mis à jour.

Installation des fichiers Jar d'exaction Apache MyFaces

Vous pouvez éditer, déployer et tester des pages JSF dans les IDE Eclipse WTP ou Sybase WorkSpace. Si l'IDE n'inclut pas les fichiers Jar d'exécution JSF, vous devez les télécharger et les installer.

1. Téléchargez la version appropriée d'Apache MyFaces JSF depuis la page suivante : *Apache MyFaces Project website*
2. Extrayez le contenu du fichier `myfaces-all.jar` dans le dossier approprié.
3. Téléchargez les fichiers jar de dépendance depuis le même site et copiez les fichiers jar depuis le dossier `myfaces-blank-example\WEB-INF\lib` dans le dossier lib d'Apache MyFaces. Vous devez avoir les fichiers jar suivants :
   - commons-beanutils-1.6.1.jar
   - commons-codec-1.2.jar
   - commons-collections-3.0.jar
   - commons-digester-1.5.jar
   - commons-el.jar
   - commons-logging.jar
   - commons-validator.jar
   - log4j-1.2.8.jar
4. Dans PowerAMC, sélectionnez **Outils > Options générales > Variables**, puis définissez une variable`JSF_LIB` pour indiquer le chemin du dossier lib d'Apache MyFaces.

Configuration pour la génération JSF

Vous configurez pour la génération JSF de la façon suivante :

1. Sélectionnez **Outils > Options générale**.
2. Sélectionnez la catégorie Variables.
3. Ajoutez une nouvelle ligne dans la liste Variables :
   - Nom : JSF_LIB
   - Valeur : sélectionnez le dossier de fichier Jar de bibliothèque JSF.
4. Sélectionnez la catégorie Chemins nommés.
5. S'il n'existe pas de chemin nommé _JSF, ajoutez une ligne dans la liste :
   - Nom : _JSF
   - Valeur : sélectionnez le dossier JSF contenant les feuilles de style JSF, les en-tête, les pieds de page et les images.

Génération de pages JSF

Avant de procéder à la génération, assurez-vous que vous avez attaché le fichier d'extension Hibernate au modèle, et vérifiez l'absence d'erreur dans le modèle

1. Sélectionnez **Langage > Générer du code Java** pour afficher la boîte de dialogue Génération.
2. Spécifiez le répertoire dans lequel les fichiers doivent être générés, et indiquez si vous souhaitez effectuer une vérification de modèle (voir *Chapitre 9, Vérification d'un MOO* à la page 303).
3. Définissez des options de génération.
4. Cliquez sur **OK** pour lancer la génération.

   La génération produit les fichiers suivants :

   - Fichiers de persistance (classes persistantes, DAO, ...)
   - Artefacts de projet Eclipse et Eclipse WTP
   - Une page d'accueil
   - Des pages JSF pour les classes persistantes :
     - Une page de recherche pour chercher des objets
     - Une page de liste pour l'affichage des résultats de recherche
     - Une page de création pour créer de nouveaux objets
     - Une page d'édition pour la mise à jour des objets
   - Des beans gérés
   - Des flux de page (face configuration files)

Test des pages JSF

Vous pouvez déployer une application Web JSF sur un serveur Web ou sur un serveur d'application qui prend en charge JSP. Par exemple, Apache Tomcat, JBoss.

Vous pouvez utiliser un IDE tel que Eclipse pour éditer les classes Java générées, et utiliser le WTP (Web Tools Project) Eclipse pour éditeur les pages JSF et les fichiers face config.

Test de JSF avec Eclipse WTP

Vous testez JSF avec Eclipse WTP de la façon suivante :

1. Installez un serveur Web tel que Tomcat ou un serveur d'application tel que JBoss.
2. Générez du code Java.
3. Importez le projet JSF dans Eclipse. Le projet est compilé.
4. Configurer votre serveur Web ou votre serveur d'application.
5. Démarrez la base de données.
6. Démarrez le serveur Web ou le serveur d'application en utilisant la vue WTP Servers.
7. Pointez sur index.jsp sous le dossier webroot, cliquez le bouton droit de la souris, puis sélectionnez **Run As > Run on Server**.
8. Sélectionnez le serveur que vous souhaitez utiliser dans la liste.

Test de pages JSF avec Apache Tomcat

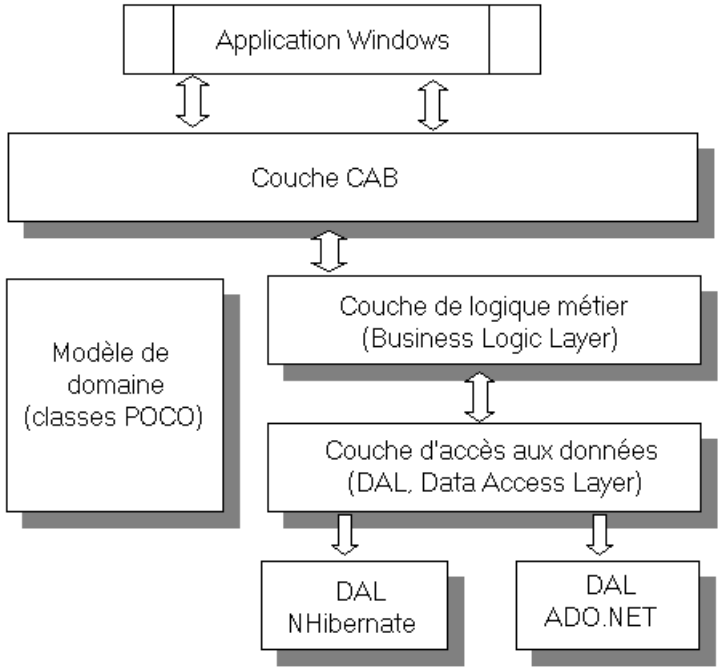
Vous testez des pages JSF avec Apache Tomcat de la façon suivante :

1. Installez un serveur Web tel que Tomcat ou un serveur d'application tel que JBoss.
2. Générez du code Java.
3. Importez le projet JSF dans Eclipse. Le projet est compilé.
4. Copiez le dossier <NomProjet> sous le dossier .deployables dans le dossier webapps de Apache Tomcat. <NomProjet> représente le nom du projet Eclipse.
5. Démarrez la base de données.
6. Démarrez le serveur Apache Tomcat.
7. Exécutez l'application Web en utilisant l'URL suivant : http://<nomhôte>:8080/<NomProjet>/index.jsp. Si Apache Tomcat est installé localement, <nomhôte> est localhost.

Chapitre 21

Génération d'objets persistants .NET 2.0 et d'applications Windows

PowerAMC adopte le mode de fonctionnement et les motifs de modélisation pour produire des applications professionnelles d'architecture n-tiers pour l'environnement .NET, comme illustré ci-après.



PowerAMC peut être utilisé pour générer toutes les couches suivantes :

- Modèle de domaine - contient des POCO (Plain Old CLR Objects) persistants, qui sont similaires aux POJO de Java. Les POCO agissent comme des conteneurs d'information pour l'application et ne contiennent pas de logique métier. Une classe de clé primaire est également générée pour chaque classe persistante afin d'aider la fonction de recherche par clé primaire, tout particulièrement si la classe à un identificateur primaire composite.
- Couche d'accès aux données (Data Access Layer) - suit le motif standard des DAO, et fournit des méthodes CMLS typiques pour chaque classe. Cette couche est divisée en deux parties, l'une contenant les interfaces pour la DAL, et l'autre contenant la mise en oeuvre

pour ces interfaces, en utilisant la technologie ADO.NET pour accéder aux bases de données.

La DAL Helper fournit des fonctionnalités communes utilisées par toutes les mises en oeuvre de DAL, telles que la gestion de connexion et de transaction, et la fourniture de paramètres de commande SQL. Certaines classes communes, telles que Session, Criteria et Exception, sont également définies.

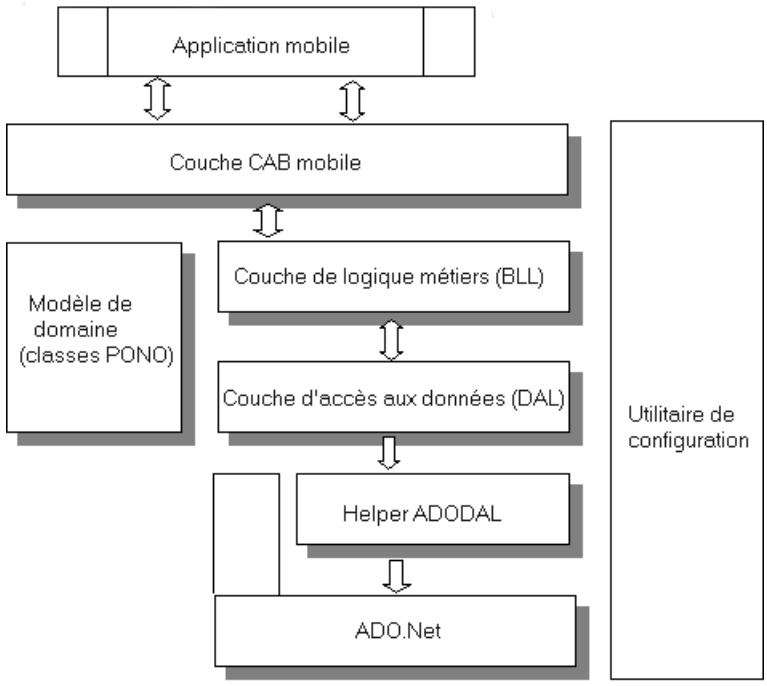
PowerAMC prend en charge deux types de mise en oeuvre de DAL :

- ADO.NET (voir *Génération d'objets ADO.NET et ADO.NET CF persistants* à la page 601)
- Nhibernate (voir *Génération d'objets persistants NHibernate* à la page 610)

- Couche de logique métier (Business Logic Layer) - contient la logique métier définie par l'utilisateur. Elle englobe le DAL, exposant les fonctionnalités CLMS fournies par la DAL sous-jacente. Vous pouvez personnaliser cette couche afin de l'adapter à vos besoins.
- Windows Application - la couche CAB (Composite UI Application Block), permet de construire des applications d'interface utilisateur complexes qui fonctionnent sur Windows. Elle fournit une architecture et une mise en oeuvre qui vous aident à créer des applications en utilisant les motifs communs trouvés dans les applications client professionnelles.

  PowerAMC peut générer des applications Windows orientées données basées sur la couche CAB (voir *Génération d'applications Windows ou Smart Device* à la page 635).

.NET CF (Compact Framework) a une organisation similaire, mais avec une classe de configuration utilitaire qui permet de charger et d'analyser la configuration utilisée dans différentes couches, par exemple la configuration de source de données, la configuration de journal et d'exception, etc :



PowerAMC prend en charge la mise en oeuvre de la DAL ADO.NET pour .NET CF (voir *Génération d'objets ADO.NET et ADO.NET CF persistants* à la page 601)

Génération d'objets ADO.NET et ADO.NET CF persistants

Microsoft ADO.NET et ADO.NET CF sont des API ne dépendant pas d'une base de données particulière.

PowerAMC prend en charge ADO.NET et ADO.NET CF via des fichiers d'extension qui améliorent la prise en charge :

- Cascade d'associations : associations un-un, un-plusieurs, plusieurs-un, et associations plusieurs-plusieurs complexes.
- Héritage : table par classe, table par sous-classe et table par hiérarchies de classes concrètes.
- Types valeur
- Instructions SQL : y compris des instructions SQL complexes telles que Join.

Pour activer ADO.NET ou ADO.NET CF dans votre modèle, sélectionnez **Modèle > Extensions**, cliquez sur l'outil **Attacher une extension**, sélectionnez le fichier `ADO.NET` ou `ADO.NET Compact Framework` (sur l'onglet **Correspondance O/R**), puis cliquez sur **OK** pour l'attacher.

PowerAMC prend également en charge la modélisation de classes .NET de structure de base de données et de correspondance objet/relationnel, et peut utiliser ces métadonnées afin de générer des objets persistants ADO.NET et ADO.NET CF parmi lesquels :

- Classes .NET persistantes (POCO)
- Classes de correspondance O/R ADO.NET (ADODALHelper)
- Factory de DAL
- Objets d'accès aux données (DAL)

Options ADO.NET et ADO.NET CF

Pour définir les paramètres de connexion à la base de données ainsi que les autres options ADO.NET ou ADO.NET CF, double-cliquez sur le nom du modèle dans l'Explorateur d'objets pour afficher sa feuille de propriétés, puis cliquez sur l'onglet ADO.NET ou ADO.NET CF tab.

Option	Description
Périphérique cible	[ADO.NET CF uniquement] Spécifie le système d'exploitation sur lequel l'application sera déployée.
Dossier du fichier de résultats	[ADO.NET CF uniquement] Spécifie l'emplacement sur le périphérique auquel l'application sera déployée. Cliquez sur le bouton Points de suspension pour éditer l'emplacement racine et ajouter les éventuels sous-répertoire appropriés.
Fournisseur de données	Spécifie le fournisseur de données que vous souhaitez utiliser. Pour ADO.NET, vous pouvez choisir entre : • OleDB • SqlClient • ODBC • Oracle Pour ADO.NET CF, vous pouvez choisir entre : • Microsoft SQL Server 2005 Mobile Edition • Sybase ASA Mobile Edition
Chaîne de connexion	Spécifie la chaîne de connexion. Vous pouvez saisir cette valeur ou cliquer sur le bouton Points de suspension à droite de la zone pour utiliser une boîte de dialogue personnalisée. Pour plus d'informations sur les paramètres spécifiques au fournisseur utilisé pour construire la chaîne de connexion, voir *Configuration des chaînes de connexion* à la page 627.
Accès par défaut	Spécifie le type d'accès par défaut pour l'attribut de classe. Cette option, ainsi que les autres options de package, sont valides pour le modèle tout entier. Vous pouvez affiner ces options pour un package individuel en utilisant sa feuille de propriétés.

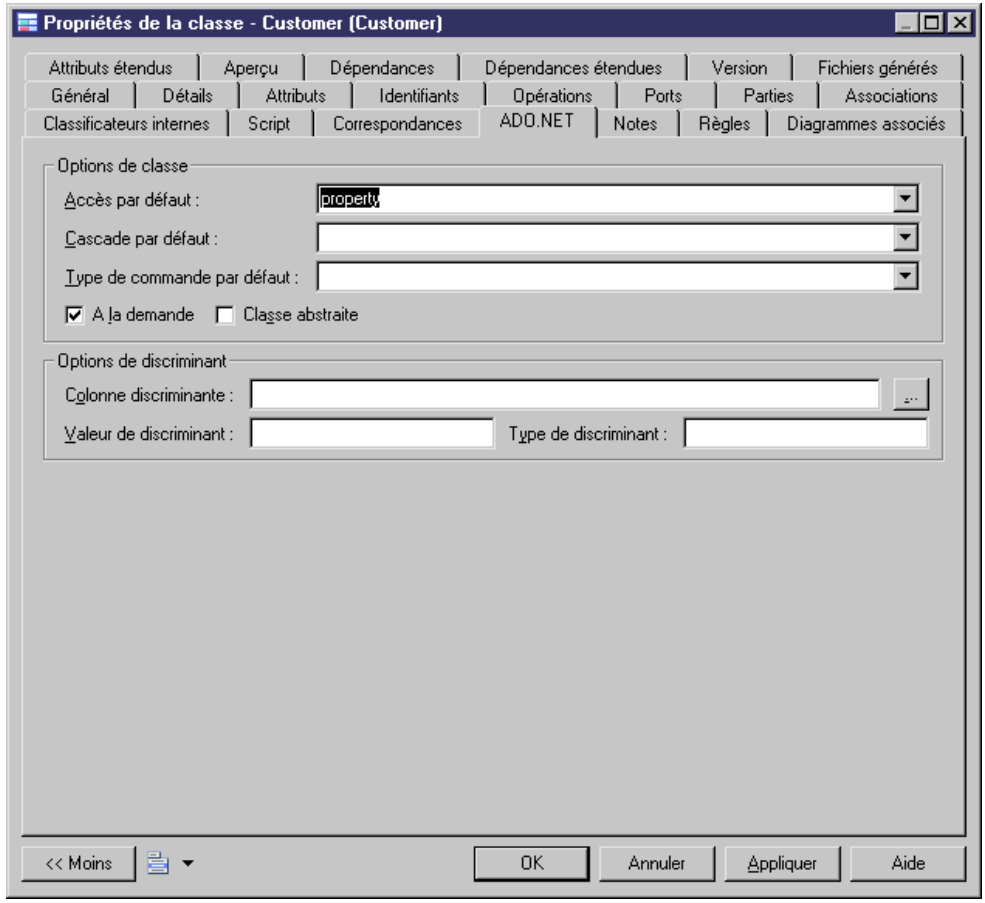
Option	Description
Cascade par défaut	Spécifie le type de cascade par défaut.
Type de commande par défaut	Spécifie le type de commande par défaut, qui peut être redéfini par une classe concrète.
Conteneur DAL	Spécifie le type de collection renvoyé depuis la base de données. Les valeurs possibles sont Generic List Collection et System.Collections.ArrayList.
Type de connexion	[ADO.NET uniquement] Le composant de connexion commun est Log4Net, mais vous pouvez le réutiliser si vous vous disposez de votre propre environnement de connexion. Par défaut, la valeur de connexion est Console type, mais PowerAMC prend en aussi en charge "None" ou Log4Net.

Correspondances de classes

Il existe deux catégories de classes dans ADO.NET et ADO.NET CF :

- Les classes d'entité - elles disposent de leur propre identité de base de données, fichier de correspondance et cycle de vie
- Les classes de type valeur - dépendent des classes d'entité. Egalement appelées classes de composant

Les options de correspondance de classe spécifiques à l'environnement sont définies sur l'onglet ADO.NET ou ADO.NET CF de la feuille de propriétés de classe :



Option	Description
Cascade par défaut	Spécifie le type de cascade par défaut.
Accès par défaut	Spécifie le type d'accès par défaut (field ou property).
Type de commande par défaut	Spécifie le type de commande. Les valeurs possibles sont SQL Text et StoreProcedure.
A la demande	Spécifie que la classe doit être chargée à la demande.
Classe abstraite	Spécifie que la classe est abstraite.
Colonne discriminante	Spécifie la colonne ou formule discriminante pour le comportement polymorphique dans une stratégie de correspondance de type "une table par hiérarchie".
Valeur de discriminant	Spécifie une valeur qui distingue les sous-classes individuelles, qui sont utilisées pour le comportement polymorphique.

Option	Description
Type de discriminant	Spécifie le type de discriminant.

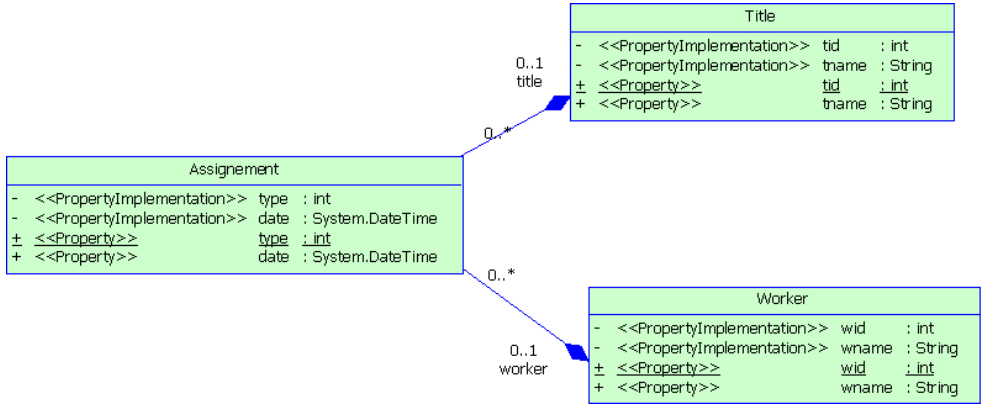
Correspondances d'identifiant primaire

La correspondance d'identifiant primaire est obligatoire dans ADO.NET et ADO.NET CF. Les identifiants primaires des classes d'entité sont mis en correspondance avec les clés primaires des tables maîtres dans les sources de données. Si elle n'est pas définie, une correspondance d'identifiant primaire par défaut sera générée, mais risque de ne pas fonctionner de façon appropriée.

Les classes mises en correspondance doivent déclarer la colonne de clé primaire de la table de base de données. La plupart des classes auront également une propriété qui contient l'identifiant unique d'une instance.

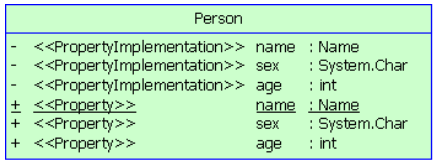
Il existe trois catégories de correspondance d'identifiant primaire dans ADO.NET et ADO.NET CF :

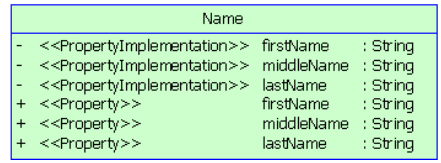
- Correspondance d'identifiant simple - Si une clé primaire est attachée à une colonne unique, seul un attribut dans l'identifiant primaire peut être mis en correspondance. Ce type de clé primaire peut être généré automatiquement. Vous pouvez définir l'incrément, l'identité, la séquence, etc., sur la colonne correspondante dans le MPD.
- Correspondance d'identifiant composite - Si une clé primaire inclut plusieurs colonnes, l'identifiant primaire peut avoir plusieurs attributs mis en correspondance avec ces colonnes. Dans certains cas, la colonne de clé primaire peut également être colonne de clé étrangère. Dans l'exemple suivant, la classe Assignment a un identifiant primaire doté de trois attributs : un attribut de base, et deux attributs migrés :



- Correspondance d'identifiant de composant - Pour plus de commodité, un identifiant composite peut être mis en oeuvre via une classe de type valeur séparée. L'identifiant primaire a juste un attribut ayant le type de classe. La classe séparée doit être définie comme classe de type valeur. La correspondance de classe de composant sera alors générée. Dans l'exemple ci-dessus, trois attributs liés au nom sont groupés dans une classe

distincte Name. Cette classe est mise en correspondance avec la même table que la classe Person.





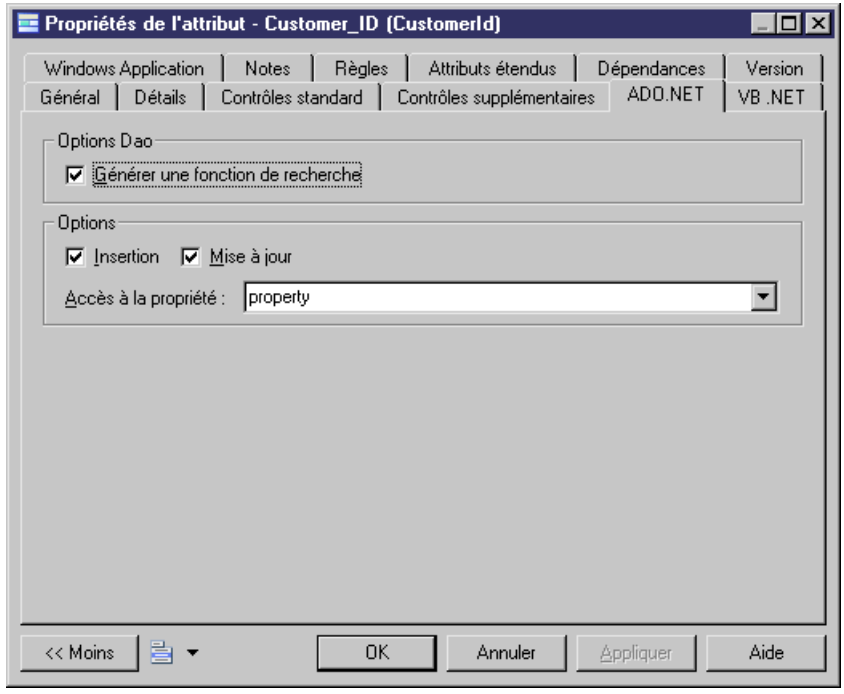
Correspondances d'attributs

Les attributs peuvent être des attributs migrés ou des attributs ordinaires. Les attributs ordinaires peuvent être mis en correspondance avec des colonnes ou des formules. Les attributs migrés ne requièrent pas de correspondance d'attribut.

Les types de correspondances suivants sont possibles :

- Correspondance d'attribut de composant - Une classe de composant peut définir la correspondance d'attributs comme c'est le cas pour les autres classes, à ceci près qu'il n'y a pas d'identifiant primaire.
- Correspondance discriminante - Dans la correspondance par héritage avec une stratégie "une table par hiérarchie", le discriminant doit être spécifié dans la classe racine. Vous pouvez définir le discriminant dans l'onglet ADO.NET ou ADO.NET CF de la feuille de propriétés de classe.

Les options relatives à la correspondance d'attribut de l'environnement sont définies dans l'onglet ADO.NET ou ADO.NET CF de la feuille de propriétés d'attribut.

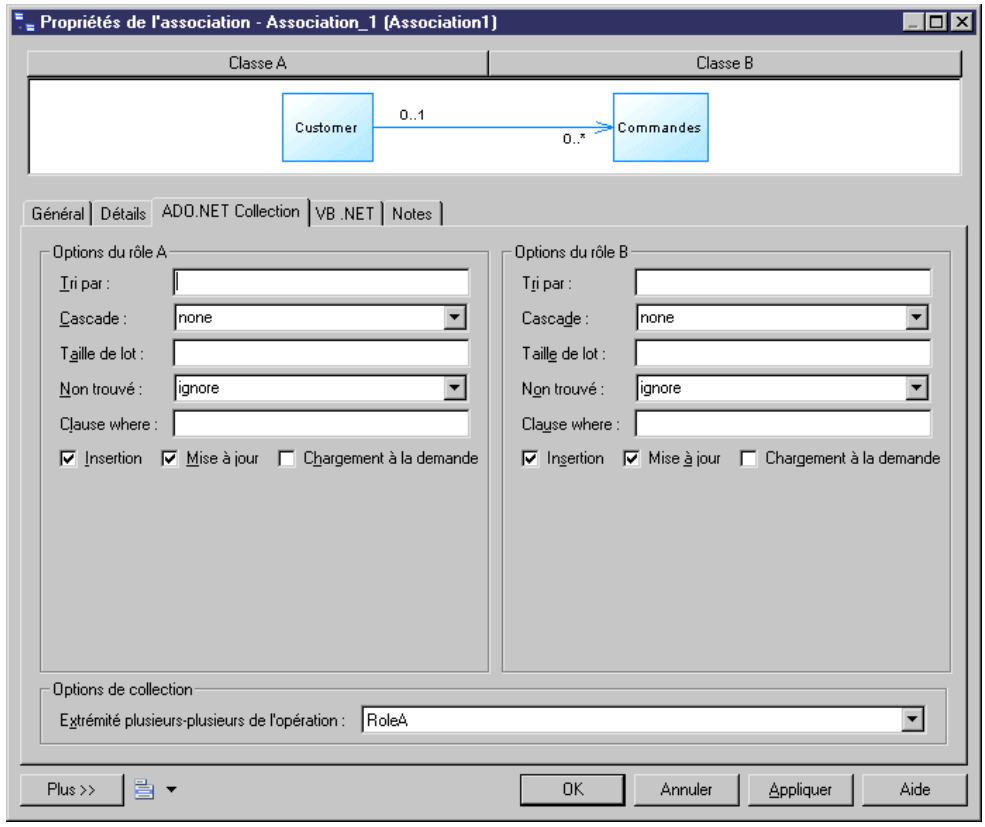


Option	Description
Générer une fonction de recherche	Génère une fonction de recherche pour l'attribut.
Insertion	Spécifie que les colonnes doivent être incluses dans n'importe quelle instruction SQL INSERT.
Mise à jour	Spécifie que les colonnes mises en correspondance doivent être incluses dans n'importe quelle instruction SQL UPDATE.
A la demande	Spécifie que cette propriété doit être chargée à la demande lors du premier accès à la variable d'instance (requiert l'utilisation de bytecode au moment de la compilation).
Accès à la propriété	Spécifie la stratégie à utiliser pour accéder à la valeur de la propriété.

Définition de correspondances d'association

ADO.NET et ADO.NET CF prennent en charge les correspondances d'association un-un, un-plusieurs/plusieurs-un et plusieurs-plusieurs. PowerAMC permet de définir des attributs d'association standard tels que conteneur, le type, la classe d'association, la navigabilité de rôle et la taille de tableau, ainsi que des attributs étendus spécifiques pour les correspondances d'association.

Affichez la feuille de propriétés de l'association et cliquez sur l'onglet ADO.NET ou ADO.NET CF Collection.



1. Définissez les options appropriées, puis cliquez sur OK.

Les options suivantes sont disponibles sur cet onglet :

Option	Description
Tri par	Spécifie une colonne (ou plusieurs colonnes) de table qui définit l'ordre d'itération de Set ou bag, ainsi qu'un asc ou desc facultatif.
Cascade	Spécifie quelles opérations doivent être transmises en cascade depuis l'objet parent vers l'objet associé.
Taille de lot	Spécifie la taille de lot.
Non trouvé	Spécifie de quelle façon les clés étrangères qui référencent les lignes manquantes seront gérées : ignore traite une ligne manquante comme une association nulle.

Option	Description
Insertion	Spécifie que les colonnes doivent être incluses dans n'importe quelle instruction SQL INSERT.
Mise à jour	Spécifie que les colonnes mises en correspondance doivent être incluses dans n'importe quelle instruction SQL UPDATE.
Chargement à la demande	Spécifie que cette propriété doit être chargée à la demande au premier accès à la variable d'instance.
Extrémité plusieurs/plusieurs de l'opération	Spécifie le point d'entrée lors des échanges de données dans une association bidirectionnelle plusieurs-plusieurs. Le résultat ne change pas que RoleA ou RoleB soit spécifié.

Définition des correspondances d'héritage

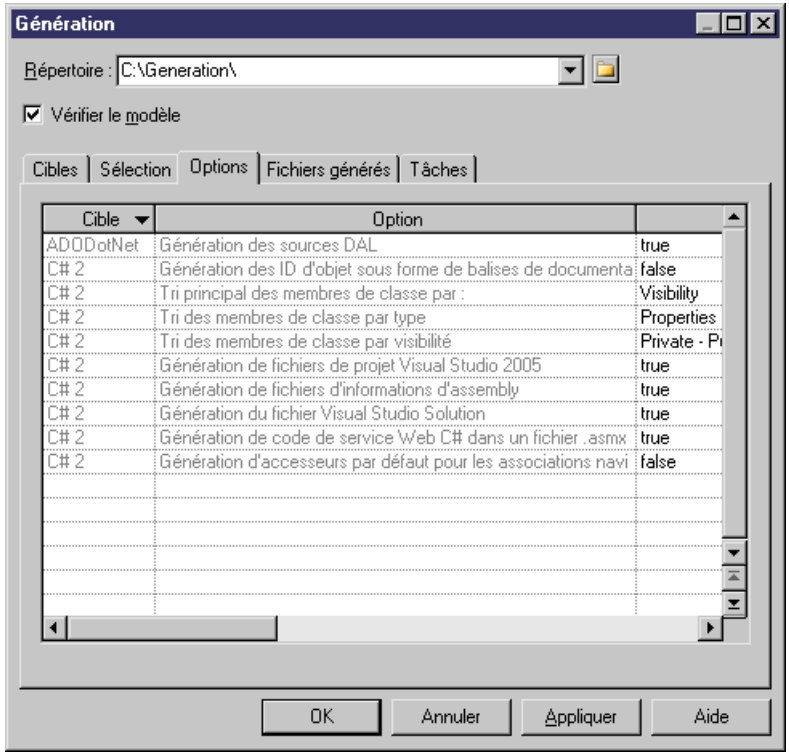
ADO.NET et ADO.NET CF prennent en charge trois stratégies de correspondances d'héritage :

- Une table par hiérarchie de classes
- Une table par sous-classe
- Une table par classe concrète
- Ces stratégies suivent toutes les définitions de correspondance d'héritage standard.

Génération de code pour ADO.NET ou ADO.NET CF

Pour pouvoir générer du code pour ADO.NET ou ADO.NET CF, vous devez avoir installé .NET Framework 2.0 ou Visual Studio.NET 2005 ou une version supérieure.

1. Sélectionnez **Outils > Vérifier le modèle** pour vous assurer que le modèle ne contient ni erreur ni avertissement. Si des erreurs sont détectées, corrigez-les avant de poursuivre avec la génération du code.
2. Sélectionnez **Langage > Générer du code C# 2 Code** ou **Générer du code Visual Basic** pour afficher la boîte de dialogue de génération :
3. Spécifiez le répertoire racine dans lequel vous souhaitez générer le code, puis cliquez sur l'onglet Options :



4. [facultatif] Pour utiliser DAL, définissez l'option Génération des sources DAL à True. Pour plus d'informations sur les options de génération standard C# et VB.NET, voir *Génération de fichiers VB.NET* à la page 461 or *Génération de fichiers C# 2.0* à la page 494.
5. Cliquez sur OK pour générer du code immédiatement ou sur Appliquer puis sur Annuler pour enregistrer vos changements pour une génération ultérieure.

Une fois la génération terminée, vous pouvez utiliser un IDE tel que Visual Studio.NET 2005 pour modifier le code, compiler et développer votre application.

Génération d'objets persistants NHibernate

NHibernate est une partie de Hibernate Core for Java adapté à .NET Framework. Il permet de gérer les POCO (Plain Old CLR Objects) persistants en relation avec une base de données relationnelle sous-jacente.

PowerAMC prend en charge NHibernate via un fichier d'extension qui améliore la prise en charge de tous les idiomes .NET communs, y compris l'association, l'héritage, le polymorphisme, la composition, ainsi que les structures de collections. Hibernate permet d'exprimer des requêtes dans sa propre extension SQL portable (HQL), ainsi qu'en SQL natif, ou au moyen d'objets Criteria et Example Java.

Pour activer NHibernate dans votre modèle, sélectionnez **Modèle > Extensions**, cliquez sur l'outil **Attacher une extension**, sélectionnez le fichier `NHibernate` (sur l'onglet **Correspondance O/R**), puis cliquez sur **OK** pour l'attacher.

PowerAMC prend en charge la conception de classes .NET, d'une structure de base de données et de la correspondance objet/relationnel (O/R). En utilisant ces métadonnées, PowerAMC peut générer des objets persistants Hibernate tels que :

- Classes .NET persistantes (objets spécifiques au domaine)
- Fichier de configuration NHibernate
- Fichiers de correspondances O/R NHibernate
- DAL factory
- Data Access Objects (DAL)

Options NHibernate

Pour définir les paramètres de connexion à la base de données ainsi que les autres options NHibernate, double-cliquez sur le nom du modèle dans l'Explorateur d'objets pour afficher sa feuille de propriétés, puis cliquez sur l'onglet NHibernate.

Option	Description
Dialecte	Spécifie le dialecte, et donc le type de base de données. Balise Hibernate : dialect
Classe du pilote JDBC	Spécifie la classe du pilote JDBC. Balise Hibernate : connection.driver_class
Chaîne de connexion	Spécifie la chaîne de connexion. Vous pouvez saisir cette valeur ou cliquer sur le bouton Points de suspension à droite de la zone pour utiliser une boîte de dialogue personnalisée. Pour plus d'informations sur les paramètres spécifiques au fournisseur utilisé pour construire la chaîne de connexion, voir *Configuration des chaînes de connexion* à la page 627. Balise NHibernate : connection.url
Importation automatique	Spécifie qu'un nom de classe non qualifié peut être utilisé dans une requête.
Accès par défaut	Spécifie le type d'accès par défaut pour l'attribut de classe. Cette option, ainsi que les autres options de package, sont valides pour le modèle tout entier. Vous pouvez affiner ces options pour un package individuel en utilisant sa feuille de propriétés.
Cascade par défaut	Spécifie le type de cascade par défaut.
Nom de schéma	Spécifie le nom de schéma de base de données par défaut.
Nom de catalogue	Spécifie le nom de catalogue de base de données par défaut.

Option	Description
Affichage du code SQL	Spécifie que les instructions SQL doivent être affichées dans le journal. Balise Hibernate : show_sql
Exportation de schéma automatique	Spécifie le mode de création des tables. Balise Hibernate : hbm2ddl.auto

Définition des correspondances de classe

Il existe deux types de classes dans NHibernate :

- Les classes d'entité - elles disposent de leur propre identité de base de données, fichier de correspondance et cycle de vie
- Les classes de type valeur - dépendent des classes d'entité. Egalement appelées classes de composant

NHibernate utilise fichiers de correspondances XML pour définir les correspondances des métadonnées. Chaque fichier de correspondance peut contenir les métadonnées d'une ou de plusieurs classes. PowerAMC utilise les stratégies de regroupement suivantes :

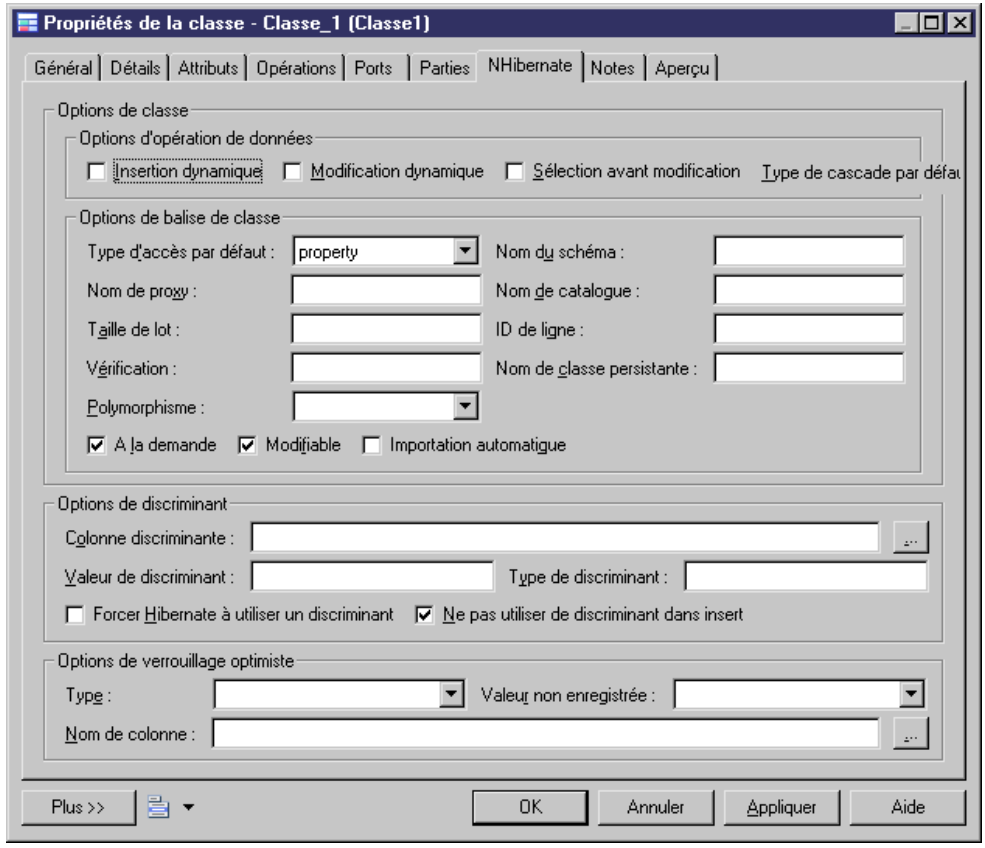
- Un fichier de correspondances distinct est généré pour chaque classe d'entité qui n'est pas une hiérarchie d'héritage.
- Un fichier de correspondances distinct est généré pour chaque hiérarchie d'héritages dotée d'une stratégie de correspondance unique. Toutes les correspondances des sous-classes sont définies dans le fichier de correspondance. Le fichier de correspondance est généré pour la classe racine de la hiérarchie.
- Aucun fichier de correspondance n'est généré pour une classe de type valeur unique qui n'appartient pas à une hiérarchie d'héritage. Sa correspondance est définie dans le fichier de correspondance de son propriétaire.

Les classes peuvent être mises en correspondances avec des tables ou des vues. Les vues ayant de nombreuses contraintes et des fonctionnalités limitées (par exemple, elles n'ont ni clé primaire ni clés de référence), certaines vues ne peuvent pas être mises à jour, et les correspondances peuvent ne pas fonctionner correctement dans certains cas.

Certaines conditions doivent être remplies pour pouvoir générer une correspondance pour une table particulière :

- La source peut être générée. Cela n'est pas possible par exemple lorsque la visibilité de la classe est private.
- La classe est persistante.
- Le mode de génération n'est pas défini à Générer un type de données abstrait et Type de valeur.
- Si la classe est une classe interne, elle doit être statique et avoir une visibilité public. NHibernate doit alors être en mesure de créer des instances de la classe.

Les options de correspondance de classe spécifiques à NHibernate sont définies dans l'onglet NHibernate de la feuille de propriétés d'une classe :



Option	Description
Insertion dynamique	Spécifie qu'une instruction INSERT SQL doit être générée lors de l'exécution et contiendra uniquement les colonnes dont les valeurs ne sont pas nulles. Balise Hibernate : dynamic-insert
Modification dynamique	Spécifie qu'une instruction UPDATE SQL doit être générée lors de l'exécution et contiendra uniquement les colonnes dont les valeurs ont été modifiées. Balise Hibernate : dynamic-update
Sélection avant modification	Spécifie que Hibernate ne doit jamais procéder à un SQL UPDATE à moins d'être certain qu'un objet soit effectivement modifié. Balise Hibernate : select-before-update

Option	Description
Type de cascade par défaut	Spécifie le style de cascade par défaut. Balise Hibernate : default-cascade
Type d'accès par défaut	Spécifie le type d'accès par défaut (champ ou propriété) Balise Hibernate : default-access
Nom de proxy	Spécifie une interface à utiliser pour l'initialisation de proxy à la demande. Balise Hibernate : proxy
Taille de lot	Spécifie une "taille de lot" pour la récupération des instances de cette classe par identifiant. Balise Hibernate : batch-size
Vérification	Spécifie une expression SQL utilisée pour générer une contrainte de vérification multiligne pour la génération automatique de schéma. Balise Hibernate : check
Polymorphisme	Spécifie si un polymorphisme de requête implicite ou explicite est utilisé. Balise Hibernate : polymorphism
Nom du schéma	Spécifie le nom du schéma de base de données. Balise Hibernate : schema
Nom de catalogue	Spécifie le nom du catalogue de base de données. Balise Hibernate : catalog
ID de ligne	Spécifie que Hibernate peut utiliser la colonne ROWID sur les bases de données qui la prennent en charge (par exemple, Oracle). Balise Hibernate : rowed
Nom de classe persistante	Spécifie une classe de persistance personnalisée. Balise Hibernate : persister
A la demande	Spécifie que la classe doit être à la demande. Balise Hibernate : lazy
Modifiable	Spécifie que les instances de la classe sont modifiables. Balise Hibernate : mutable
Classe abstraite	Spécifie que la classe est abstraite. Balise Hibernate : abstract
Importation automatique	Spécifie qu'un nom de classe non qualifié peut être utilisé dans une requête. Balise Hibernate : Auto-import

Option	Description
Colonne discriminante	Spécifie la colonne ou formule discriminante pour le comportement polymorphique dans une table par stratégie de correspondance hiérarchique. Balise Hibernate : discriminator
Valeur discriminante	Spécifie une valeur qui distingue les sous-classes individuelles, qui sont utilisées pour un comportement polymorphique. Balise Hibernate : discriminator-value
Type discriminant	Spécifie le type discriminant. Balise Hibernate : type
Forcer Hibernate à utiliser un discriminant	Force Hibernate à spécifier des valeurs discriminantes admises et ce, même lors de l'extraction de toutes les instances de la classe racine. Balise Hibernate : force
Ne pas utiliser de discriminant dans insert	Force Hibernate à ne pas inclure la colonne dans les SQL INSERT Balise Hibernate : insert
Type de verrouillage optimiste	Spécifie une stratégie de verrouillage optimiste. Balise Hibernate : optimistic-lock
Nom de colonne (de verrouillage optimiste)	Spécifie la colonne pour le verrouillage optimiste. Un champ est également généré si cette option est définie. Balise Hibernate : version/ timestamp
Valeur non enregistrée (de verrouillage optimiste)	Spécifie si une valeur non enregistrée est nulle ou indéfinie. Balise Hibernate : unsaved-value

Correspondances d'identifiant primaire

La correspondance d'identifiant primaire est obligatoire dans NHibernate. Les identifiants primaires des classes d'entité sont mis en correspondance avec les clés primaires des tables principales dans les source de données. Si vous n'en définissez aucune, une correspondance d'identifiant primaire par défaut est générée, mais risque de ne pas fonctionner correctement.

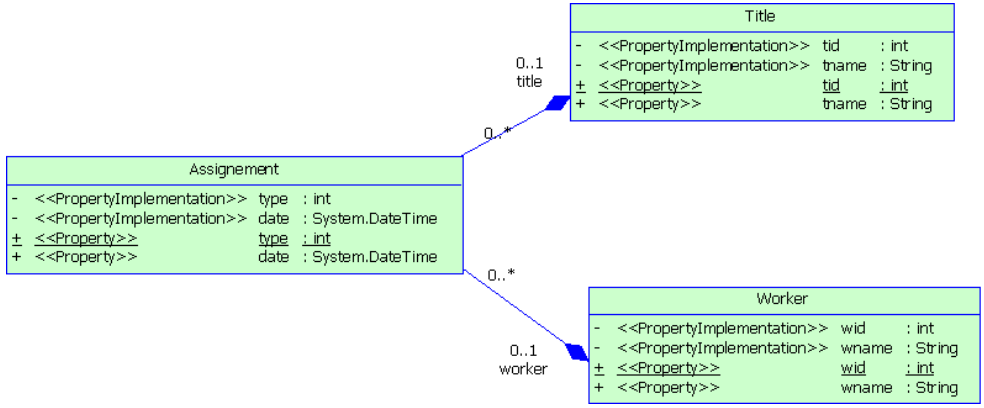
Il existe trois types de correspondance d'identifiant primaire dans NHibernate :

- Correspondance d'identifiant simple
- Correspondance d'identifiant composite
- Correspondance d'identifiant de composant

Les classes mises en correspondance doivent déclarer la colonne de clé primaire de la table de base de données. La plupart des classes auront également une propriété qui contiendra l'identificateur unique d'une instance.

Correspondance d'identifiant composite

Si une clé primaire inclut plusieurs colonnes, l'identifiant primaire peut avoir plusieurs attributs mis en correspondance avec ces colonnes. Dans certains cas, la colonne de clé primaire peut également être colonne de clé étrangère.

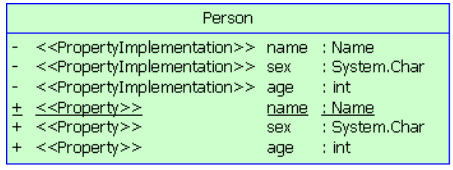


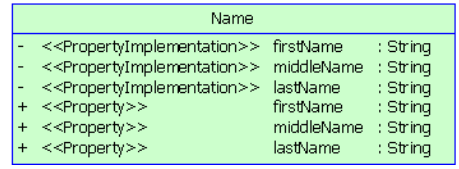
Dans l'exemple ci-dessus, la classe Assignment a un identifiant primaire doté de trois attributs : un attribut de base, et deux attributs migrés. La correspondance d'identifiant se présente comme suit :

```
<composite-id>
  < key-property name="Type" access="property">
    < column name="Type" sql-type="integer"
                not-null="true"/>
  </key-property>
  <key-many-to-one name="title" access="property">
  </key-many-to-one>
  <key-many-to-one name="worker" access="property">
  </key-many-to-one>
</composite-id>
```

Correspondance d'identifiant primaire de composant

Pour plus de commodité, un identifiant composite peut être mis en oeuvre via une classe de type valeur séparée. L'identifiant primaire a juste un attribut ayant le type class. La classe séparée doit être définie comme classe de type valeur. La correspondance de classe de composant sera alors générée.





Dans l'exemple ci-dessous, trois attributs liés au nom sont groupés dans une classe distincte Name. Cette classe est mise en correspondance avec la même table que la classe Person. L'identifiant primaire généré se présente comme suit :

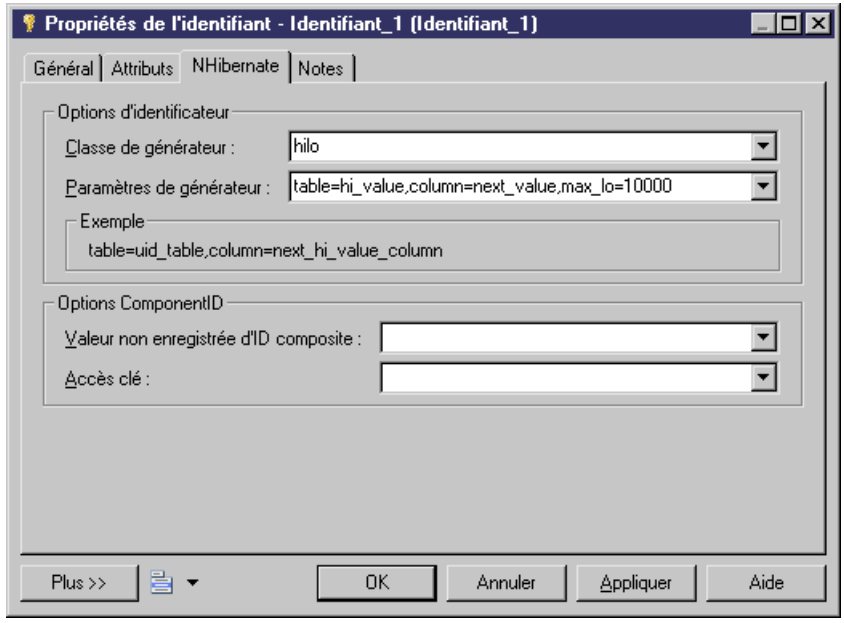
```
<composite-id name="name" class="identifier.Name">
  <key-property name="firstName">
    <column name="name_firstName"
    sql-type="text"/>
  </key-property>
  <key-property name="middleName">
    <column name="name_middleName"
    sql-type="text"/>
  </key-property>
  <key-property name="lastName">
    <column name="name_lastName"
    sql-type="text"/>
  </key-property>
</composite-id>
```

Remarque : La classe de type valeur doit réaliser l'interface java.io.Serializable interface, qui met en oeuvre les méthodes equals() et hashCode().

Correspondance d'identifiant simple

Si une clé primaire est attachée à une colonne unique, seul un attribut dans l'identifiant primaire peut être mis en correspondance. Ce type de clé primaire peut être généré automatiquement. Vous pouvez définir la classe de générateur ainsi que les paramètres. Il existe de nombreux types de classe de générateur tels que increment, identity, sequence, etc. Chaque type de classe de générateur peut avoir des paramètres qui lui sont propres. Pour plus d'informations, reportez-vous à votre documentation NHibernate.

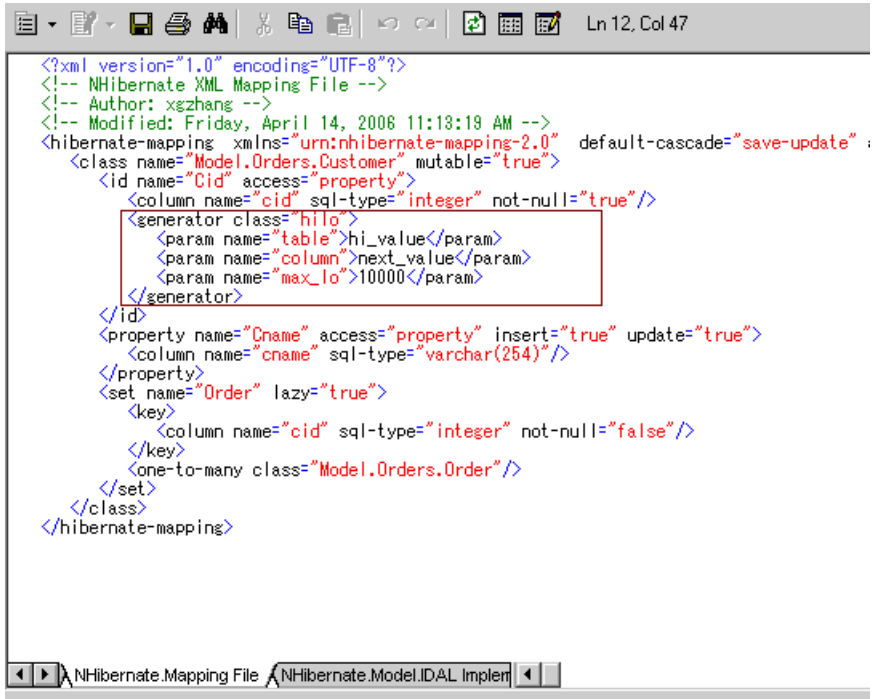
Vous pouvez définir la classe de générateur et les paramètres dans l'onglet NHibernate de la feuille de propriétés d'un identifiant primaire. Les paramètres ont la forme suivante : param1=value1; param2=value2.



1. Afficher la feuille de propriétés de la classe, puis cliquez sur l'onglet Identifiants. Double-cliquez sur l'entrée appropriée pour afficher sa feuille de propriétés.
2. Cliquez sur l'onglet NHibernate, sélectionnez une classe de générateur et définissez ses paramètres.

   Exemples de paramètres :

   - Sélectionnez hilo dans la liste Classe de générateur
   - Saisissez "table=hi_value,column=next_value,max_lo=10000" dans la zone Paramètres de générateur. Vous devez utiliser des virgules pour séparer les paramètres.
3. Vous pouvez visualiser le code dans l'onglet Aperçu :



Notez que, s'il existe plusieurs attributs d'identifiant primaire, le générateur n'est pas utilisé.

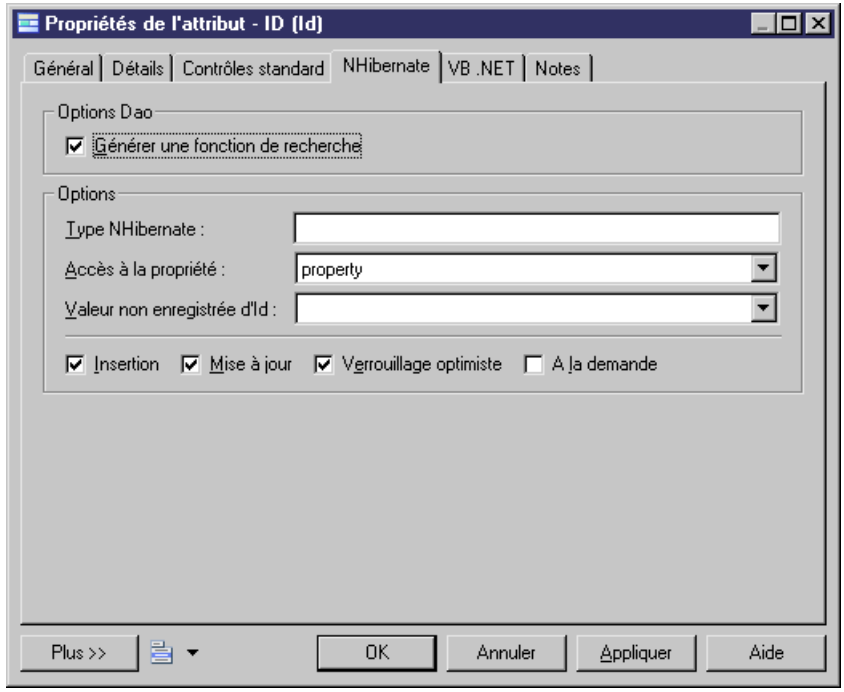
Correspondances d'attributs

Les attributs peuvent être des attributs migrés ou des attributs ordinaires. Les attributs ordinaires peuvent être mis en correspondance avec des colonnes ou des formules. Les attributs migrés ne requièrent pas de correspondance d'attribut.

Les types de correspondances suivants sont possibles :

- Correspondance d'attribut et d'une formule - Lorsque vous mettez en correspondance un attribut et une formule, vous devez vous assurer que la syntaxe est correcte. Il n'y a aucune colonne dans la table source de la correspondance d'attribut.
- Correspondance d'attribut de composant - Une classe de composant peut définir la correspondance d'attribut comme pour une autre classe, à ceci près qu'il n'y pas d'identifiant primaire.
- Correspondance discriminante - Dans la correspondance par héritage avec une stratégie "une table par hiérarchie", le discriminant doit être spécifié dans la classe racine. Vous pouvez définir le discriminant dans l'onglet NHibernate de la feuille de propriétés de classe.

Les options relatives à la correspondance d'attribut NHibernate sont définies dans l'onglet NHibernate de la feuille de propriétés d'attribut.

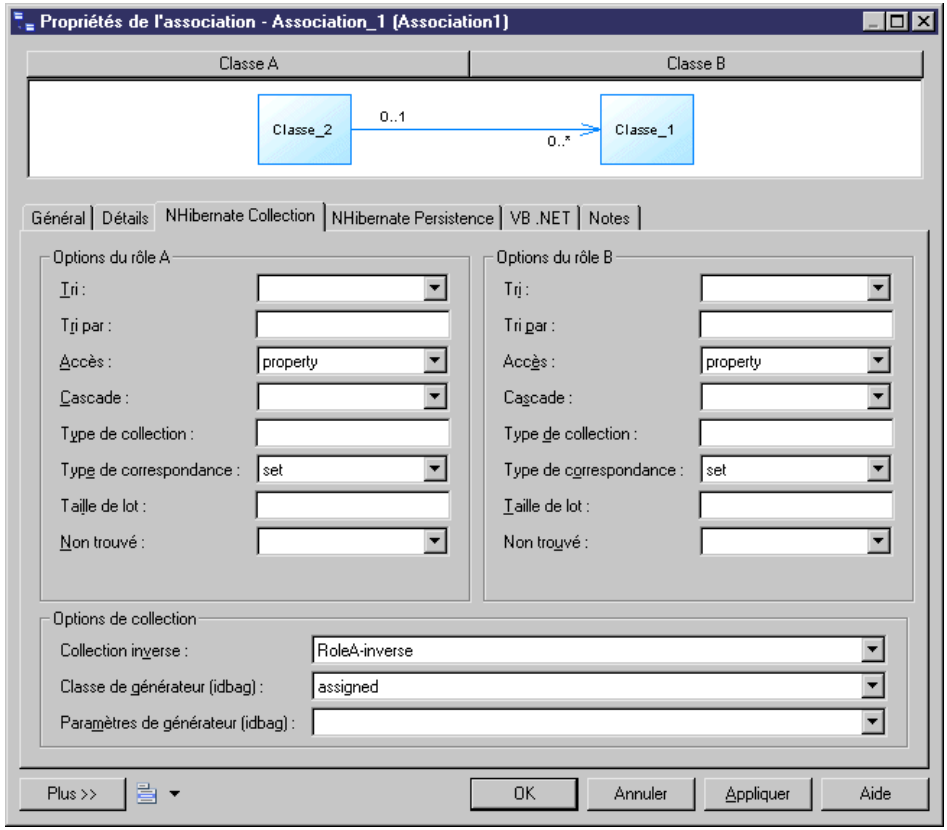


Option	Description
Générer une fonction de recherche	Génère une fonction de recherche pour l'attribut.
Type NHibernate	Spécifie un nom qui indique le type NHibernate.
Accès à la propriété	Spécifie la stratégie que NHibernate doit utiliser pour accéder à la valeur de la propriété.
Valeur non enregistrée d'ID	Spécifie la valeur d'un ID non enregistré.
Insertion	Spécifie que les colonnes doivent être incluses dans n'importe quelle instruction SQL INSERT.
Mise à jour	Spécifie que les colonnes mises en correspondance doivent être incluses dans n'importe quelle instruction SQL UPDATE.
Verrouillage optimiste	Spécifie que les mises à jour de cette propriété requièrent l'acquisition du verrouillage optimiste.
A la demande	Spécifie que cette propriété doit être chargée à la demande lors du premier accès à la variable d'instance (requiert l'utilisation de bytecode au moment de la compilation).

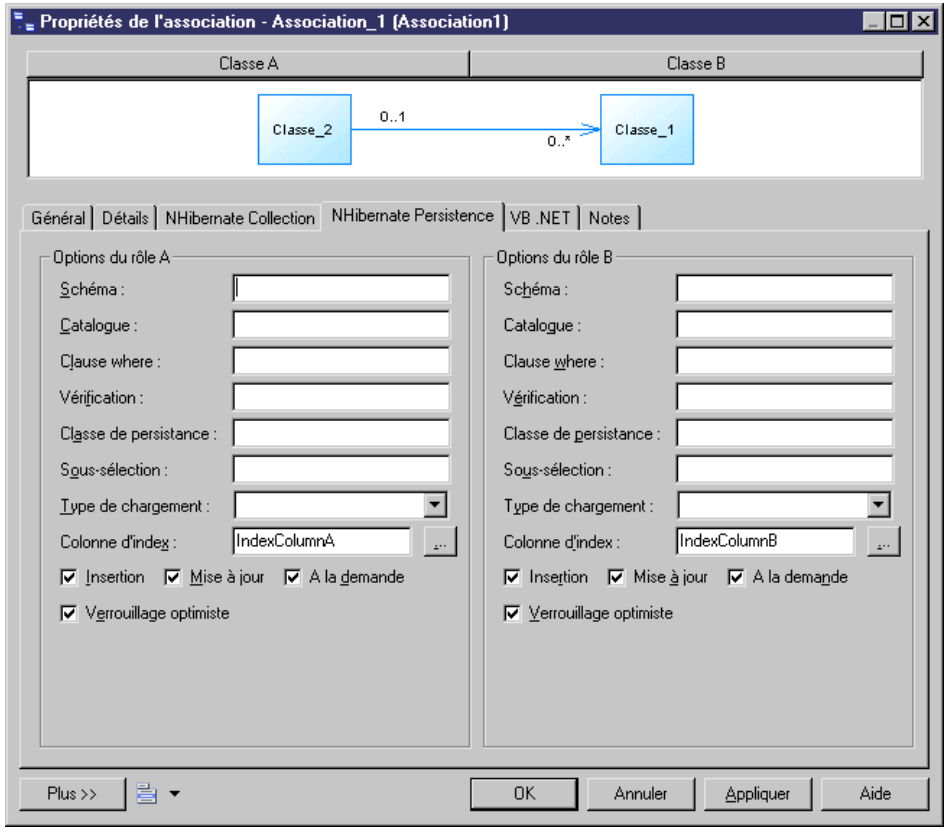
Définition de correspondances d'association

NHibernate prend en charge les correspondances d'association un-un, un-plusieurs/plusieurs-un et plusieurs-plusieurs. La modélisation des correspondances s'effectue de la même manière que pour une modélisation de correspondance O/R standard. Toutefois, NHibernate fournit des options particulières afin de définir ses correspondances d'association, qui seront enregistrées dans le fichier de correspondance <Class>.hbm.xml. PowerAMC permet de définir des attributs d'association standard tels que conteneur, le type, la classe d'association, la navigabilité de rôle et la taille de tableau, ainsi que des attributs étendus spécifiques pour les correspondances d'association NHibernate.

1. Affichez la feuille de propriétés de l'association et cliquez sur l'onglet NHibernate Collection.
2. Définissez les options de gestion de collection (voir *Définition des options de collection NHibernate* à la page 622).



3. Cliquez sur l'onglet NHibernate Persistence, puis définissez les options de persistance (voir *Définition des options de persistance NHibernate* à la page 623).



Définition des options de collection NHibernate

Les options suivantes sont disponibles sur cet onglet :

Option	Description
Tri	Spécifie une collection triée avec un ordre de tri naturel, ou une classe de comparaison donnée. Balise NHibernate : sort
Tri par	Spécifie une colonne (ou plusieurs colonnes) de table qui définit l'ordre d'itération de Set ou bag, ainsi qu'un asc ou desc facultatif. Balise NHibernate : order-by
Access	Spécifie la stratégie que NHibernate doit utiliser pour accéder à la valeur de la propriété. Balise NHibernate : access

Option	Description
Cascade	Spécifie quelles opérations doivent être transmises en cascade depuis l'objet parent vers l'objet associé. Balise NHibernate : cascade
Collection type	Spécifie un nom qui indique le type NHibernate. Balise NHibernate : type
Taille de lot	Spécifie la taille de lot. Balise NHibernate : batch-size
Non trouvé	Spécifie de quelle façon les clés étrangères qui référencent les lignes manquantes seront gérées : ignore traite une ligne manquante comme une association nulle. Balise NHibernate : not-found
Collection inverse	Spécifie que le rôle est la relation inverse du rôle opposé. Balise NHibernate : inverse
Type de correspondance	Spécifie le type de correspondance de la collection. Balise NHibernate : Set, Array, Map ou List.

Définition des options de persistance NHibernate

Les options suivantes sont disponibles sur cet onglet :

Option	Description
Schéma	Spécifie le nom du schéma. Balise NHibernate : schema
Catalogue	Spécifie le nom du catalogue. Balise NHibernate : catalog
Clause Where	Spécifie une condition SQL WHERE arbitraire à utiliser pour la récupération des objets de cette classe. Balise NHibernate : where
Vérification	Spécifie une expression SQL utilisée pour générer une contrainte de vérification multiligne pour la génération automatique de schéma. Balise NHibernate : check
Type de chargement	Spécifie un chargement par jointure externe ou par sélection séquentielle. Balise NHibernate : fetch
Classe de persistance	Spécifie une classe de persistance personnalisée. Balise NHibernate : persister

Option	Description
Sous-sélection	Spécifie une entité non modifiable et en lecture seule pour une sous-sélection de base de données. Balise NHibernate : subselect
Colonne d'index	Spécifie le nom de colonne si les utilisateurs utilisent le type de collection list ou array. Balise NHibernate : index
Insertion	Spécifie que les colonnes mises en correspondance doivent être incluses dans n'importe quelle instruction SQL INSERT. Balise NHibernate : insert
Mise à jour	Spécifie que les colonnes mises en correspondance doivent être incluses dans n'importe quelle instruction SQL UPDATE. Balise NHibernate : update
A la demande	Spécifie que cette propriété doit être chargée à la demande lors du premier accès à la variable d'instance. Balise NHibernate : lazy
Verrouillage optimiste	Spécifie si le numéro de version doit être incrémenté si la valeur de cette propriété est modifiée. Balise NHibernate : optimistic-lock
Jointure externe	Spécifie d'une jointure externe doit être utilisée. Balise NHibernate : outer-join

Définition d'un type de conteneur de collection NHibernate

NHibernate prend en charge les types de correspondance Set, Bag, List, Array et Map, il limite le type de conteneur. PowerAMC ne prend pas en charge le type de correspondance Map.

Type de correspondance de collection	Type de conteneur de collection
Set	Iesi.Collections.ISet
Bag	System.Collections.IList
List	System.Collections.IList
Array	<Aucun>

Vous pouvez spécifier le type de conteneur manuellement, ou bien sélectionner le type de correspondance nécessaire, et PowerAMC va automatiquement sélectionner le type de conteneur approprié.

Définition des correspondances d'héritage

NHibernate prend en charge deux stratégies de correspondances d'héritage de base

- Une table par hiérarchie de classes
- Une table par sous-classe

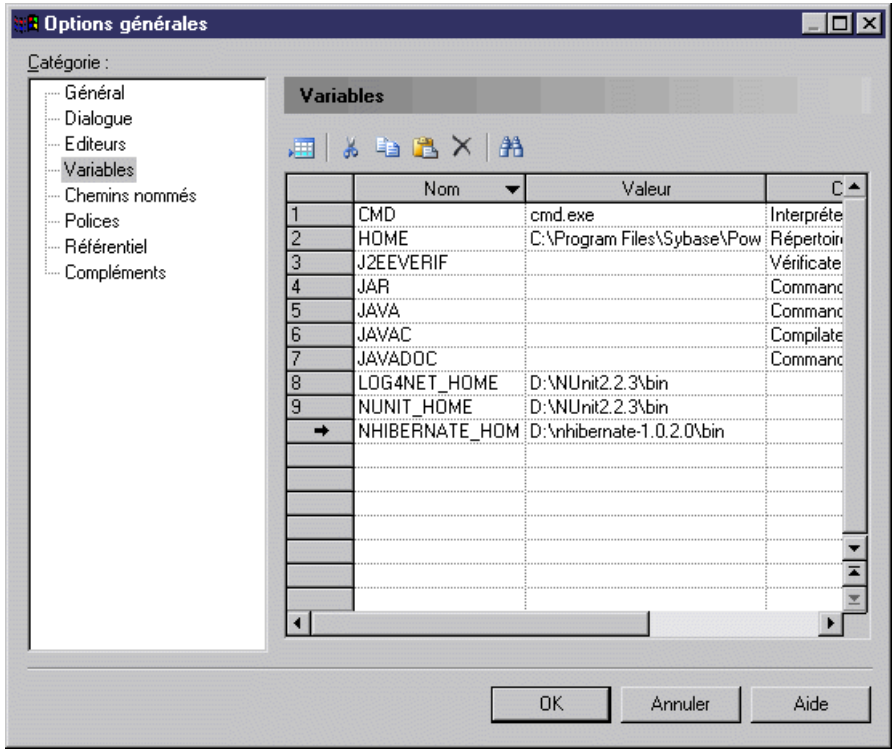
PowerAMC ne prend pas en charge la stratégie de correspondance "Une table par classe concrète".

Ces stratégies suivent toutes les définitions de correspondance d'héritage standard. Toutefois, un fichier de correspondance distinct est généré pour chaque hiérarchie d'héritage dotée d'une stratégie de correspondance unique. Toutes les correspondances des sous-classes sont définies dans le fichier de correspondance. Le fichier de correspondance est généré pour la classe racine de la hiérarchie.

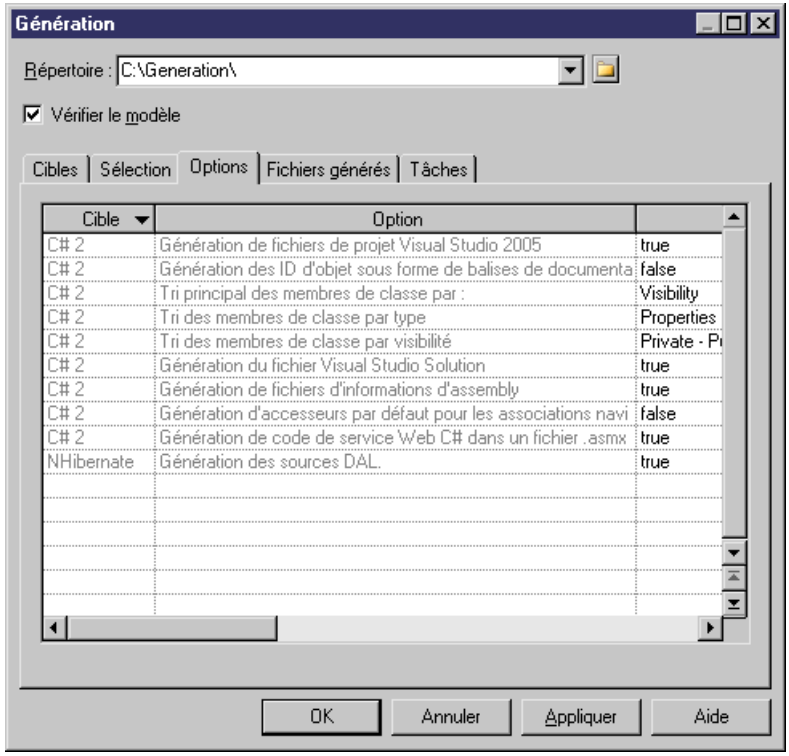
Génération de code pour NHibernate

Avant de générer du code pour NHibernate, vous devez avoir installé NHibernate 1.0.2 ou version supérieure.

1. Sélectionnez **Outils > Vérifier le modèle** afin de vous assurer de l'absence d'erreur dans votre modèle. Si la vérification détecte des erreurs, corrigez ces dernières avant de poursuivre avec la génération de code.
2. Sélectionnez **Outils > Options générales**, puis cliquez sur le noeud Variables.



3. Ajoutez une variable NHIBERNATE_HOME et spécifiez dans la colonne valeur le chemin d'accès du répertoire racine de NHibernate. Par exemple : D:\nhibernate-1.0.2.0\bin.
4. Sélectionnez **Langage > Générer du code C# 2** ou **Générer du code Visual Basic 2005 Code** pour afficher la boîte de dialogue Génération :
5. Spécifiez le répertoire racine dans lequel vous souhaitez générer le code, puis cliquez sur l'onglet Options :



6. [facultatif] Pour utiliser DAL, définissez l'option Génération des sources DAL à true. Pour plus d'informations sur les options de génération standard pour C# et VB.NET, reportez-vous aux chapitres appropriés.
7. Cliquez sur **OK** pour générer du code immédiatement ou sur **Appliquer** puis sur **Annuler** pour enregistrer vos changements pour une génération ultérieure.

Une fois la génération terminée, vous pouvez utiliser un IDE tel que Visual Studio.NET 2005 pour modifier le code, compiler et développer votre application.

Configuration des chaînes de connexion

PowerAMC prend en charge plusieurs types de chaînes de connexion à la base de données avec les différents environnements .NET.

Chaque connexion requiert un jeu de paramètres différent, qui peuvent être spécifiés à la main dans le champ Chaîne de connexion de l'onglet ADO.NET ou NHibernate, ou via des boîtes de dialogue personnalisées accessibles via l'outil Points de suspension situé à droite de cette zone.

Configuration d'une chaîne de connexion à partir de ADO.NET ou ADO.NET CF

Vous configurez une chaîne de connexion à partir de ADO.NET ou ADO.NET CF de la façon suivante :

1. Sélectionnez un fournisseur de données.
2. Cliquez sur le bouton Points de suspension pour afficher une boîte de dialogue spécifique au fournisseur.
3. [ADO.NET uniquement] Spécifiez les paramètres requis et cliquez sur le bouton Tester la connexion pour les valider.
4. Cliquez sur Appliquer à la chaîne de connexion, puis sur Fermer pour revenir à l'onglet ADO.NET ou ADO.NET CF.

Configuration d'une chaîne de connexion à partir de 'onget NHibernate

Vous configurez une chaîne de connexion à partir de NHibernate de la façon suivante :

1. Sélectionnez un dialecte et une classe de pilote.
2. Cliquez sur le bouton Points de suspension pour afficher une boîte de dialogue de chaîne de connexion spécifique au fournisseur.
3. Spécifiez les paramètres requis et cliquez sur le bouton Tester la connexion pour les valider.
4. Si le test de la connexion donne les résultats appropriés, cliquez sur Fermer pour revenir à l'onglet NHibernate.

Paramètres de chaîne de connexion OLEDB

Les paramètres suivants sont requis pour configurer une chaîne de connexion OLEDB :

Option	Description
Fournisseur de données	Spécifie le fournisseur de données depuis la liste.
Nom de serveur ou de fichier	Spécifie le nom de serveur ou de fichier.
Nom d'utilisateur	Spécifie le nom d'utilisateur de base de données.
Mot de passe	Spécifie le mot de passe d'utilisateur de base de données.
Utiliser la sécurité intégrée de Windows NT	Spécifie l'utilisation de la sécurité intégrée de Windows NT.
Permettre l'enregistrement du mot de passe	Spécifie si vous souhaitez autoriser l'enregistrement du mot de passe.

Option	Description
Catalogue initial	Spécifie le catalogue initial de la base de données.

Paramètres de chaîne de connexion ODBC

Les paramètres suivants sont requis pour configurer une chaîne de connexion ODBC :

Option	Description
Source de données ODBC	Spécifie le nom de la source de données ODBC.
Nom d'utilisateur	Spécifie le nom d'utilisateur de base de données.
Mot de passe	Spécifie le mot de passe d'utilisateur de base de données.

Paramètres de chaîne de connexion Microsoft SQL Server et Microsoft SQL Server Mobile Edition

Les paramètres suivants sont requis pour configurer une chaîne de connexion Microsoft SQL Server et Microsoft SQL Server Mobile Edition :

Option	Description
Nom de serveur	Spécifie le nom du serveur.
Nom d'utilisateur	Spécifie le nom d'utilisateur de base de données.
Mot de passe	Spécifie le mot de passe d'utilisateur de base de données.
Type d'authentification	Spécifie type d'authentification, Utilisez SQL Server, ou Windows.
Nom de la base de données	Spécifie le nom de la base de données.
Nom de fichier de la base de données	Spécifie le nom du fichier de la base de données.
Nom logique	Spécifie le nom logique du fichier de la base de données.

Paramètres de chaîne de connexion Oracle

Les paramètres suivants sont requis pour configurer une chaîne de connexion Oracle :

Option	Description
Nom de serveur	Spécifie le nom du serveur.
Nom d'utilisateur	Spécifie le nom d'utilisateur de base de données.
Mot de passe	Spécifie le mot de passe d'utilisateur de base de données.

Génération de code pour UnitTest

Vous pouvez générer les tests à l'aide de NUnit ou de Visual Studio Test System. PowerAMC prend en charge la génération de code de tests unitaires via l'utilisation d'un fichier d'extension.

Si le modèle contient de nombreuses classes persistantes, il peut être difficile de toutes les tester pour s'assurer que :

- Les correspondances sont correctement définies
- Les options CLMS (Création, Lecture, Modification, Suppression) fonctionnent
- Les méthodes de recherche fonctionnent
- La navigations fonctionne

Le plus souvent, les développeurs doivent développer des tests unitaires ou des interfaces utilisateur afin des tester ces objets. PowerAMC permet d'automatiser cette tâche fastidieuse en utilisant NUnit ou Visual Studio Test System (VSTS) pour générer les classes de test unitaire. La plupart du code généré pour ces deux environnements de tests unitaires est très similaire. Les principales différences sont les suivantes :

- Team Test utilise différentes attributs avec NUnit dans une classe de test, tels que [TestClass()] to [TestFixture] and [TestMethod()] to [Test] etc.
- Le fichier AllTests n'est pas généré car tous les tests seront exécutés dans l'interface de Visual Studio ou à l'invite de commande.
- Log4net est remplacé par le fichier de résultats de test .trx qui peut être ouvert dans la fenêtre Résultats des tests dans Visual St.

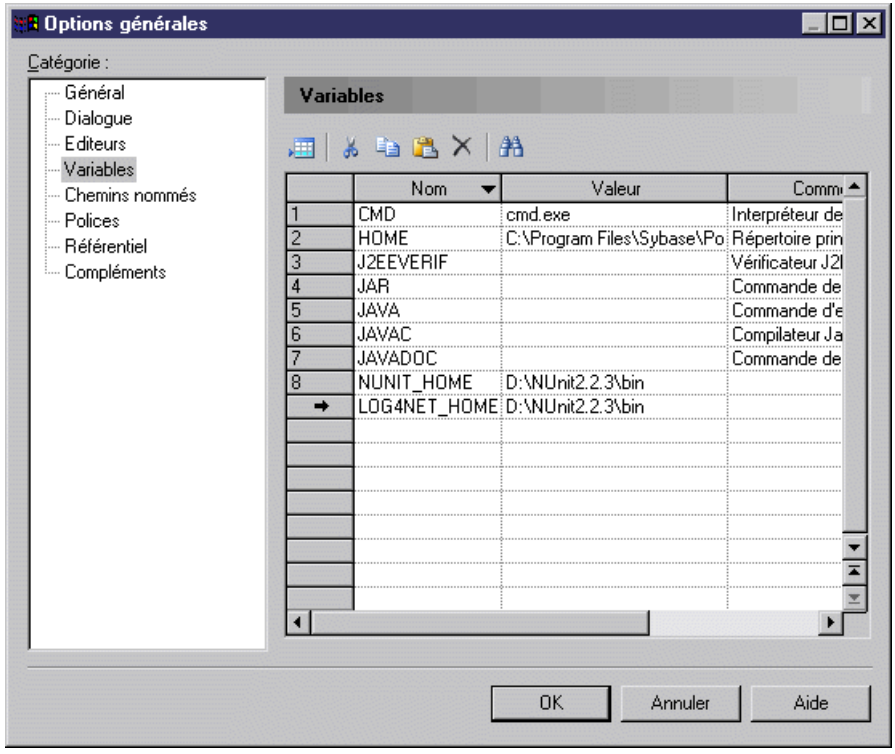
Certaines conditions doivent être remplies pour procéder au test unitaire d'une classe :

- La correspondance d'une classe doit être définie.
- La classe peut être instanciée. Les classes de test unitaire ne peuvent pas être générées pour les classes abstraites.
- La classe n'est pas un type valeur.
- La propriété Modifiable est définie à true. Si Modifiable est définie à false, la classe ne peut être ni modifiée ni supprimée.
- La classe n'a aucune contrainte de clé étrangère non respectée. Si une clé étrangère est obligatoire, la classe parent doit être accessible (navigable du côté de la classe parent) à des fins de test.

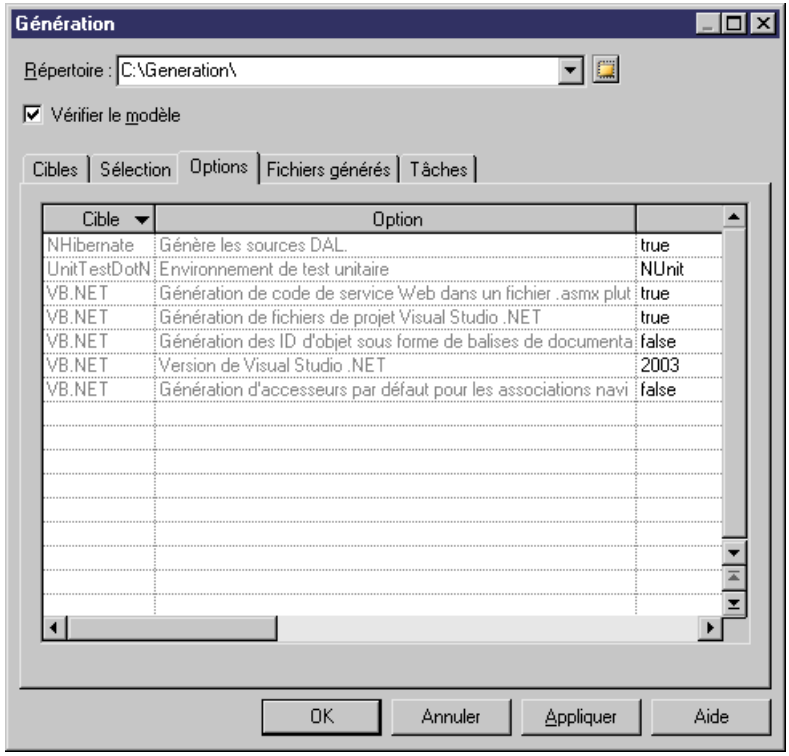
Pour activer UnitTest dans votre modèle, sélectionnez **Modèle > Extensions**, cliquez sur l'outil **Attacher une extension**, sélectionnez le fichier `UnitTest.NET` ou `UnitTest.NET CF` (sur l'onglet **Unit Test**), puis cliquez sur **OK** pour l'attacher.

Avant de générer du code pour UnitTest, vous devez installer NUnit 2.2.3 ou version supérieure (disponible à l'adresse *http://www.nunit.org*).

1. Sélectionnez **Outils > Vérifier le modèle** pour vous assurer que le modèle ne contient ni erreur ni avertissement. Si des erreurs sont détectées, corrigez-les avant de poursuivre avec la génération du code.
2. Sélectionnez **Outils > Options générales**, puis cliquez sur le noeud Variables.
3. Ajoutez la variable NUNIT_HOME et, dans la colonne Valeur, spécifiez le chemin du répertoire racine de NUnit. Par exemple, spécifiez D:\NUnit2.2.3\bin. Ajoutez une variable LOG4NET_HOME de la même façon si log4net doit être utilisé pour la consignation.



4. Sélectionnez **Langage > Générer du code C# 2** ou **Générer du code Visual Basic 2005** pour afficher la boîte de dialogue de génération.
5. Spécifiez le répertoire racine dans lequel vous souhaitez générer le code, puis cliquez sur l'onglet **Options** :



6. Sélectionnez un environnement de test unitaire dans "Environnement de test unitaire". Vous pouvez choisir Nunit ou Visual Studio Team Test.
7. Cliquez sur **OK** pour générer du code immédiatement ou sur **Appliquer**, puis sur **Annuler** pour enregistrer vos modifications pour plus tard.

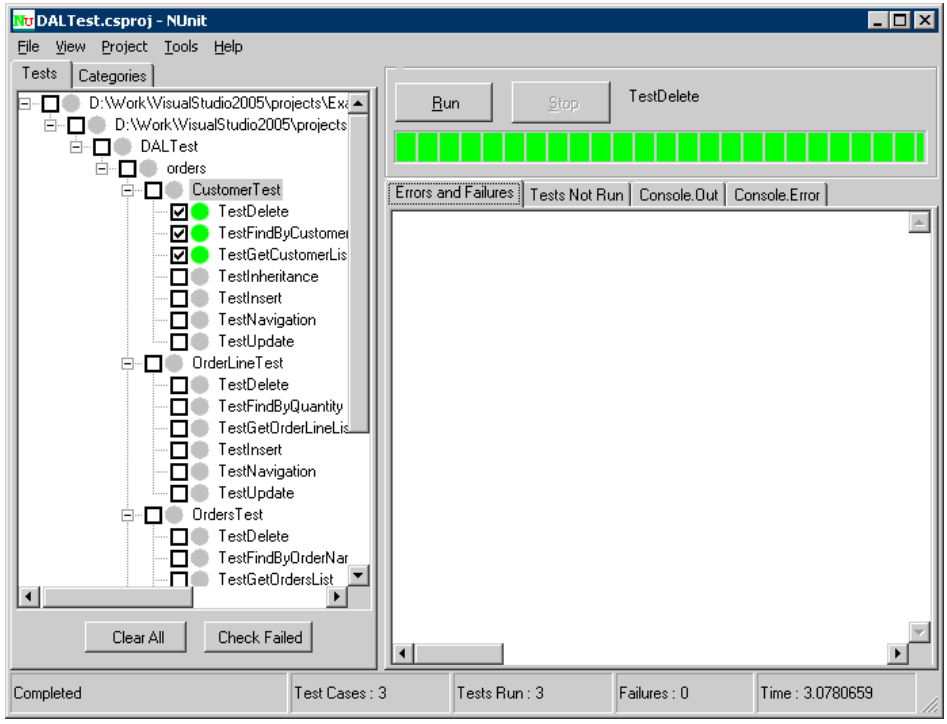
Exécution des tests unitaires NUnit

Après avoir généré votre code de test, vous pouvez l'exécuter de l'une des façons suivantes :

- Exécuter dans Nunit - NUnit peut exécuter de deux façons vos cas de test. L'exécution par console, via nunit-console.exe, est plus rapide à lancer, mais elle n'est pas interactive. L'exécution via l'interface graphique, en utilisant nunit-gui.exe, est une application Windows basée sur des formulaires qui permet de gérer de façon sélective l'exécution des tests et d'obtenir un retour d'information sous forme graphique.

  NUnit met également à votre disposition l'attribut Category qui fournit une alternative aux suites pour la gestion d'un groupe de tests. Les tests ou contextes de test individuels peuvent être identifiés comme appartenant à une catégorie particulière. L'exécution des test via l'interface ou via la console permet de spécifier une liste des catégories à inclure ou à exclure de l'exécution. Lorsque vous utilisez des catégories, seuls les tests de la catégorie sélectionnée seront exécutés. Les tests appartenant à des catégories non sélectionnées ne sont pas signalés.

- Interface graphique Nunit - Le programme nunit-gui.exe est un module d'exécution graphique. Il montre les tests dans une fenêtre de type explorateur et fournit une indication visuelle du succès ou de l'échec de ces tests. Il permet d'exécuter des tests particuliers ou des suites de tests de façon sélective et se recharge automatiquement lorsque vous modifiez et recompilez votre code.



Comme vous pouvez le voir, les tests qui n'ont pas été exécutés sont marqués par un cercle gris, tandis que ceux qui ont été exécutés avec succès sont représentés par un cercle vert. Si un test a échoué, il est représenté par un cercle rouge.

- Console Nunit - Le programme nunit-console.exe est un programme d'exécution en ligne de commande que vous pouvez utiliser lorsque vous souhaitez exécuter tous vos tests et n'avez pas besoin d'une indication graphique du succès ou de l'échec des tests. Il convient à l'automatisation des tests et à l'intégration dans d'autres systèmes. Il enregistre automatiquement ses résultats au format XML, ce qui permet de produire des rapports ou de traiter les résultats.

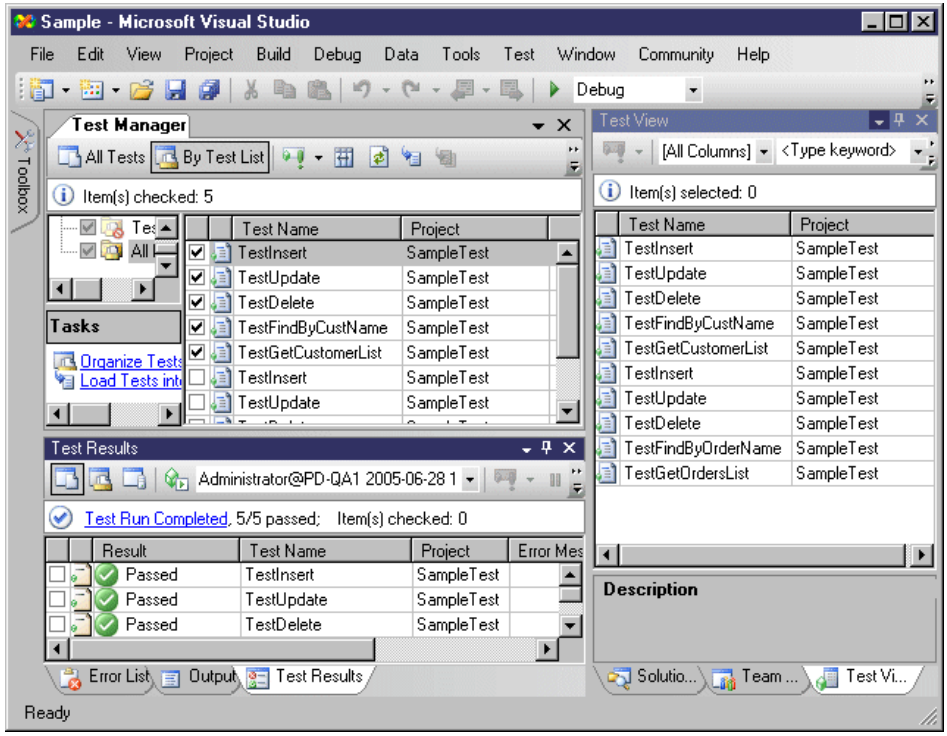
Exécution de test unitaires dans Visual Studio Test System

Les outils de test de Visual Studio Team System Team permettent d'exécuter des tests de différentes façons, à la fois depuis l'environnement de développement intégré (IDE) ou vie une interface de ligne de commande.

Exécution des tests depuis l'IDE Visual Studio.NET 2005 :

Vous exécutez des tests depuis l'IDE Visual Studio.NET 2005 de la façon suivante :

1. Utilisez la fenêtre Gestionnaire de tests ou Affichage de tests. Vous pouvez également relancer l'exécution de tests depuis la fenêtre Résultats des tests.
2. Dans la fenêtre Gestionnaire de tests, sélectionnez les tests en cochant la case correspondante, puis cliquez sur Exécuter les tests activés dans la barre d'outils, ou bien pointez sur les tests sélectionnés, cliquez le bouton droit de la souris, puis cliquez sur Exécuter les tests activés.
3. Dans la fenêtre Affichage de tests, sélectionnez les tests à exécuter, puis cliquez sur Exécuter la sélection dans la barre d'outils Affichage de tests, ou bien pointez sur les tests sélectionnés, cliquez le bouton droit de la souris, puis cliquez sur Exécuter la sélection.



Execution des tests depuis la ligne de commande

Vous exécutez des tests depuis la ligne de commande de la façon suivante :

1. Ouvrez une invite de commandes Visual Studio
2. Passez dans le répertoire de votre solution ou, lorsque vous exécutez le programme MSTest.exe pas-à-pas, spécifiez le chemin d'accès absolu ou relatif vers le fichiers de métadonnées ou le conteneur de test.

3. Exécutez le programme MSTest.exe.

Génération d'applications Windows ou Smart Device

Le CAB (Composite UI Application Block) vous aide à construire des interfaces utilisateur complexes qui fonctionnent sous Windows. Il met à votre disposition une architecture et une mise en oeuvre qui vous aident à construire des applications en utilisant les motifs communs trouvés dans les applications client professionnelles. La version courant de Composite UI Application Block est destiné aux applications Windows Forms qui sont exécutées avec Microsoft .NET Framework 2.0.

PowerAMC prend en charge le développement d'applications Windows et Smart Device à l'aide de fichiers d'extension pour vos MOO C# v2.0 ou Visual Basic 2005.

Vous pouvez construire rapidement des applications Windows et Smart Device en utilisant PowerAMC pour générer des classes persistantes, des fichiers DAL, BLL et fichiers d'interface basés sur CAB en fonction de votre environnement persistant NHibernate ou ADO.NET.

Pour activer Windows ou Smart Device dans votre modèle, sélectionnez **Modèle > Extensions**, cliquez sur l'outil **Attacher une extension**, sélectionnez le fichier `Windows Application` ou `Smart Device Application` sur l'onglet **User Interface**), puis cliquez sur **OK** pour l'attacher.

Remarque : Vous devez également ajouter l'extension de gestion de persistance ADO.NET ou Nhibernate afin d'être en mesure de générer vos fichiers d'application.

En utilisant cette application Windows, vous pouvez tester des objets persistants avec vos propres données. Vous pouvez également améliorer les fichiers générés et changer la disposition comme vous le souhaitez dans Visual Studio.

Spécification d'une bibliothèque d'images

Vos formulaires vont probablement utiliser des images comme icônes. PowerAMC fournit une bibliothèque d'images par défaut, qu'il utilise par défaut pour les applications Windows. Vous pouvez également spécifier votre propre bibliothèque d'images.

1. Affichez la feuille de propriétés du modèle, puis cliquez sur l'onglet Window Application.
2. Spécifiez un chemin vers votre bibliothèque d'images, puis cliquez sur OK pour revenir au modèle.

Contrôle de la DataGridView

Vous pouvez spécifier la longueur des vos contrôles DataGridView pour ADO.NET CF.

1. Ouvrez la feuille de propriétés du modèle, puis cliquez sur l'onglet Smart Device Application.

2. Spécifiez le nombre de lignes, puis cliquez sur OK pour revenir au modèle.

Définition des options d'affichage relatives aux attributs

Pour pouvez définir des options de niveau attribut pour les styles de présentation.

1. Affichez la feuille de propriétés d'attribut, puis sélectionnez l'onglet Windows Application.
2. Définissez les options appropriées, puis cliquez sur OK.

Les options disponibles sont les suivantes :

Option	Description
Type de contrôle	Vous pouvez choisir entre TextBox et ComoboBox comme type de contrôle.
Afficher comme libellé de clé étrangère	Spécifie que l'attribut courant doit être affiché sous la forme d'un libellé de clé étrangère dans les zones de liste modifiables, au lieu de la clé étrangère elle-même. Par exemple, sélectionnez cette option pour afficher le nom du produit au lieu de l'ID du produit.

Définition des règles de validation d'attribut et des valeurs par défaut

PowerAMC fournit la validation et les valeurs par défaut pour les zones d'édition dans les formulaires Create, Find, ListView et DetailView.

1. Affichez la feuille de propriétés d'un attribut, puis cliquez sur l'onglet Contrôles standard.
2. [facultatif] Définissez les valeurs minimum et maximum pour contrôler la plage de valeurs.
3. [facultatif] Définissez une valeur par défaut. Une valeur chaîne doit être placée entre apostrophes. Vous pouvez également définir la valeur initiale dans l'onglet Détails.
4. [facultatif] Définissez une liste de valeurs. PowerAMC va automatiquement générer une zone de liste modifiable qui inclut ces valeurs.
5. Cliquez sur OK pour revenir au modèle.

Génération de code pour une application Windows

Avant de générer du code pour une application Windows, vous devez installer le CAB (disponible sur le site Web de Microsoft).

Vérifiez la présence des fichiers requis suivants dans votre répertoire d'installation :

- Microsoft.Practices.CompositeUI.dll
- Microsoft.Practices.CompositeUI.WinForms.dll
- Microsoft.Practices.ObjectBuilder.dll

1. Sélectionnez **Outils > Vérifier le modèle** afin de vous assurer de l'absence d'erreur dans votre modèle. Si la vérification détecte des erreurs, corrigez ces dernières avant de poursuivre avec la génération de code.
2. Sélectionnez **Outils > Options générales**, puis cliquez sur l'onglet Variables.
3. Ajoutez une variable CAB_PATH avec comme valeur le chemin de votre répertoire d'installation.
4. Sélectionnez **Langage > Générer du code C# 2.0 ou Générer du code Visual Basic 2005** pour afficher la boîte de dialogue Génération.
5. Spécifiez le répertoire de destination, puis cliquez sur OK pour lancer la génération.

   Les fichiers suivants seront générés :

   - Fichiers de domaine (classes persistantes, fichiers de correspondance)
   - Fichiers DAL (fichier de connexion à une base de données et fichiers d'accès aux données)
   - Un fichier de solution et des fichiers de projet
   - Une boîte de dialogue de connexion
   - Des formulaires pour les classes persistantes
   - Des fichiers contrôleurs

Pour chaque classe persistante, PowerAMC génère :

- Une boîte de dialogue de recherche pour chercher des objets
- Une vue sous forme de liste pour l'affichage des résultats de la recherche
- Une vue détaillée pour afficher les informations détaillées relatives à l'objet
- Un formulaire de vue de création pour créer de nouveaux objets

Vous pouvez déployer ou éditer une application Windows dans un IDE, tels que Visual Studio .NET 2005.

Génération de code pour une application Smart Device Application

PowerAMC peut générer du code pour des applications Smart Device.

Avant de générer une application smart device, vous devez :

- Vous assurer que vous avez attaché l'extension ADO.NET Compact Framework (ainsi que, si vous souhaitez générer des tests unitaires, l'extension UnitTest.NET Compact Framework) (voir *Génération d'applications Windows ou Smart Device* à la page 635).
- Définir les propriétés de modèle appropriées sur les onglets ADO.NET CF et Application, y compris une chaîne de connexion fonctionnelle (voir *Options ADO.NET et ADO.NET CF* à la page 602).
- Spécifier les valeurs appropriées pour les variables suivantes (sélectionnez **Outils > Options générales**, puis cliquez sur la catégorie Variables):

- CFUT_HOME – si vous utilisez Microsoft Mobile Client Software Factory CFUnitTester
- ASANET_HOME – si vous utilisez Sybase ASA. Spécifie l'emplacement de iAnywhere.Data.AsaClient.dll.
- SQLSERVERMOBILENET_HOME – si vous utilisez Microsoft SQL Server Mobile Edition. Spécifie l'emplacement de System.Data.SqlServerCe.dll
- ULTRALITENETCE_HOME – si vous utilisez Sybase UltraLite. Spécifie l'emplacement de ulnet10.dll
- ULTRALITENET_HOME – si vous utilisez Sybase UltraLite. Spécifie l'emplacement de iAnywhere.Data.UltraLite.dll et de en\iAnywhere.Data.UltraLite.resources.dll

1. Sélectionnez **Outils > Vérifier le modèle** afin de vérifier l'absence d'erreur dans le modèle. Si des erreurs sont détectées, corrigez-les avant de poursuivre la génération de code.
2. Sélectionnez **Langage > Générer du code C#2 Code ou Générer du code Visual Basic 2005** pour afficher la boîte de dialogue Génération.
3. Spécifiez le répertoire racine dans lequel vous souhaitez générer le code, puis cliquez sur l'onglet **Options**.
4. Spécifiez les options appropriées, puis cliquez sur **OK** pour générer du code immédiatement ou bien sur **Appliquer** puis sur **Annuler** pour sauvegarder vos modifications pour une utilisation ultérieure.

Déploiement du code sur un périphérique smart device

Vous déployez du code sur un périphérique smart device de la façon suivante :

1. Compilez votre code généré dans Visual Studio.
2. Déployez le projet de démarrage, c'est-à-dire le projet <model>Test ou User Interface.
3. Déployez le projet SystemFramework séparément avec le fichier de base de données et les DLL requises (tels que ulnet10.dll pour la prise en charge de UltraLite).

Test de l'application sur le périphérique

Vous testez l'application sur le périphériqued de la façon suivante :

1. Si vous avez généré et déployé les projets d'interface utilisateur sur le périphérique, vous pouvez les exécuter et tester l'application en spécifiant des données.
2. Si vous avez généré pour 'Microsoft Mobile Client Software Factory', vous pouvez exécuter les tests unitaires en cliquant sur GuiTestRunner.exe dans le dossier de déploiement sur le périphérique. Le fichier exe et ses références peuvent être copiés dans le dossier d'installation de Microsoft Mobile Client Software Factory.

Index

B

C

D

E

F

G

H

I

N

R

S

T

U

V

W

X

Z