Appeon マイグレーション ガイド

Appeon® 6.0 for PowerBuilder® 日本語版

FOR WINDOWS

ドキュメント ID：DC00388-01-0600-02

改訂：2009 年 02 月 02 日

Appeon Corporation, 1/F, Shell Industrial Building, 12 Lee Chung Street, Chai Wan District, Hong Kong.

目次

1 はじめに

1.1 対象読者

このマニュアルは、Appeon® 6.0 for PowerBuilder®日本語版を使用して新しく Web アプリケーションを開発、または既存の Sybase® PowerBuilder®アプリケーションを Web にマイグレーションする開発者を対象としています。

1.2 このマニュアルの内容

このマニュアルは次に示す 7 章から構成されます。

第 1 章：はじめに

このマニュアルの一般的な内容を説明します。

第 2 章：PowerBuilder と Appeon による Web RAD 開発

PowerBuilder と Appeon for PowerBuilder 日本語版を使用して Web アプリケーションを迅速に開発する方法を紹介します。

第 3 章：マイグレーション プロセス

既存の PowerBuilder アプリケーションを Web アプリケーションに変換する方法を説明します。

第 4 章：マイグレーション FAQ

Appeon 6.0 for PowerBuilder 日本語版を使用して PowerBuilder アプリケーションを Web へマイグレーションする際に、繰り返し質問される事項に対して回答します。

第 5 章：分散アプリケーションの構築とマイグレーション

分散アプリケーションを Web へマイグレーションする特別な手順を説明します。

第 6 章：Web または Appeon 機能によるアプリケーションの拡張

Web に配布されたアプリケーションが効果的に機能するために、PowerBuilder アプリケーションに追加できる拡張機能について説明します。

第 7 章：マイグレーション チュートリアル

実際の操作に沿って、小さな PowerBuilder アプリケーションを Web 化するプロセスを解説しています。

1.3 関連マニュアル

Appeon では Appeon 製品とその機能をより良く理解するために、次のマニュアルを提供しています。

- 『Appeon デモ アプリケーション チュートリアル』（英語）

  Appeon デモ アプリケーションについて説明しています。Appeon デモ アプリケーションには「Appeon Sales Application」、「Appeon Code Examples」、「Appeon ACF Demo」が提供されており、PowerBuilder アプリケーションを Web 化する Appeon の機能を体験することができます。

- 『Appeon Developer ユーザ ガイド』（または Appeon ヘルプにおける『Appeon Developer ユーザ ガイド』）（英語）

  Appeon 6.0 により提供される Appeon Developer ツールバーの使用方法を説明します。

  Appeon ヘルプにおける『Appeon Developer ユーザ ガイド』は『Appeon Developer ユーザ ガイド』の HTML 版です。

- 『Appeon Server コンフィグレーション ガイド』（英語）

  Appeon Server ステータス モニタの設定方法、Appeon Server とデータベース サーバの接続方法、および Appeon Server と Appeon Web アプリケーションを保守するための AEM の設定方法を説明します。

- 『Appeon サポート機能ガイド』（または『Appeon 機能ヘルプ』）（英語）

  Appeon 6.0 でサポートされ、Web アプリケーションに変換可能な PowerBuilder 機能の詳細、および未サポート機能について説明します。

  『Appeon 機能ヘルプ』は『Appeon サポート機能ガイド』の HTML 版です。

- 『Appeon インストレーション ガイド』

  Appeon for PowerBuilder 日本語版のインストール方法について説明しています。

- 『Appeon マイグレーション ガイド』

Appeon による Web 化の手順と、これらの手順を構成するステップに関連する様々なトピックを完全な図解入りで説明しており、また、チュートリアルでは小さな PowerBuilder アプリケーションを Web に配備するプロセスを一通り体験することができます。

- 『Appeon パフォーマンス チューニング ガイド』（英語）

  Appeon により Web 化されたアプリケーションのパフォーマンスを向上させるための、PowerBuilder アプリケーションの修正方法について説明しています。

- 『Appeon トラブルシューティング ガイド』（英語）

  製品のインストレーション、Web 配布、AEM、Web アプリケーションの実行等に関するトラブルシューティングについて説明しています。

- 『Appeon の紹介』（英語）

  Appeon 6.0 for PowerBuilder 日本語版に含まれるすべてのドキュメントについて説明します。

- 『Appeon 新機能ガイド』（英語）

  Appeon 6.0 for PowerBuilder 日本語版の新機能および変更された機能を説明します。

1.4 不明な点があるときは

Appeon ソフトウェアがインストールされているサイトでは、Sybase のサポート契約を結んでいる Sybase サポート センタまたは Sybase に認定されたサポート パートナーとの連絡担当の方（コンタクト パーソン）が決めております。マニュアルやオンライン ヘルプでは解決できない問題があった場合、その担当の方を通して Sybase のサポート センタまたは Sybase に認定されたサポート パートナーまでご連絡ください。Sybase サポート センタの Web サイトには http://www.sybase.com/support でアクセスできます。

2 PowerBuilder と Appeon による RAD Web 開発

2.1 概要

PowerBuilder と Appeon for PowerBuilder 日本語版は、Web アプリケーションの開発において、比類の無い迅速な開発（Web RAD）と高い生産性を実現します。PowerBuilder と Appeon による Web の RAD 開発はソフトウェア開発のライフサイクルと、IT 開発の難しさを大幅に減少し、エンド ユーザのこれまでのソフトウェア開発への投資を最大限に保護します。

2.1.1 Web 開発の伝統的なアプローチ

伝統的な方法で Web 開発を行うと、小さな Web アプリケーションでさえ多くの時間を要し、結果として作成された Web アプリケーションのユーザ インタフェースは単純なもので、表現力豊かな操作性の高いユーザ インタフェースを実現できません。また、Web アプリケーションの開発者は、Microsoft または Java テクノロジーを使用するために、ASP/JSP/PHP、JavaScript、XML、HTML、Java、および J2EE のような多様な新しいスキルをマスターする必要があります。

2.1.2 Appeon による Web RAD 開発のアドバンテージ

強力な Web アプリケーションを開発するために、PowerBuilder と Appeon を使用する方法には、次に示す利点があります。

Web への最速の方法を提供

Appeon for PowerBuilder 日本語版は、アプリケーションの Web への書き換えに伴う Web アプリケーションのデザイン、コーディング、およびテストの伝統的な作業を排除し、プロジェクトのライフサイクルとコストで前例の無い削減を達成します。また、Web アプリケーションの新規開発を目的とする場合、Appeon は最も開発生産性の高いアプローチを提供し、規制の多い Web 標準に従うエンタプライズ クラスのアプリケーションの構築を可能にします。

最適な Web GUI の提供

Appeon の Web アプリケーションは、標準の Microsoft Web ブラウザ内で実行される HTML によって、デスクトップ アプリケーションのユーザ インタフェースを正確に再現します。Appeon のグラフィカルな能力は、利用可能な他のどんな方法よりもエンドユーザにより高い生産性をもたらし、ユーザの再教育時間およびコストをすべて排除します。

ビジネス リスクの最小化

Appeon for PowerBuilder は、十分テストされた PowerBuilder プログラム コードとデータウィンドウが Web アプリケーションのコアとして実行されるため、新しいソフトウェア アプリケーションの構築に関連したビジネス リスクを大幅に減少します。

現行の企業スキルの優位性

Appeon for PowerBuilder 日本語版は：

- アプリケーション開発と保守に PowerBuilder スキルと開発環境のみを使用します。
- 既存のアプリケーション ソースコードとデータベースを再利用することにより、デザイン、アプリケーション UI の構築とテスト、アプリケーション ビジネス ロジックデータ アクセス ロジック、およびデータベースへの投資を抑制します。
- デスクトップと Web 環境への配布を 1 セットのネィティブな PowerBuilder ソースコードで管理できます。

2.2 N-Tier NVO 機能の優位性

Appeon の N-Tier NVO のサポートは、拡張された Web 機能へのアクセスを可能にします。PowerBuilder と Appeon によって Web アプリケーションを新規開発する場合、EAServer 内の N-Tier NVO コンポーネントをブリッジとして使用して、Web 上の多くの機能と Web の制約を超えるいくつかのクライアント サーバ アプリケーション機能にアクセスできます。N-Tier NVO の機能の詳細はセクション 5.2「未サポート機能を N-Tier NVO として Appeon Server へ移行」を参照してください。

2.3 Appeon の Web RAD 開発

PowerBuilder と Appeon による Web RAD（Rapid Application Development）は、新しい PowerBuilder アプリケーションの記述と Appeon for PowerBuilder を使用してそのアプリケーションを Web に変換することにより行われます。

Appeon の Web 開発プロジェクトには、次に示す 3 つのステップがあります。

ステップ 1 ― プロジェクト リクワイアメントの分析。これらのリクワイアメントに沿ってプロジェクトに適合する機能仕様をレイアウトします。

ステップ 2 ー 新しい PowerBuilder アプリケーションを開発し、Appeon の PowerBuilder コーディング 標準に準拠するプログラミングによって機能を実装します。

コーディングの過程では、Appeon Developer の Code Insight ツールを利用して未サポート機能を含まない新しいコードを記述できます。

PowerBuilder アプリケーションの新規開発においては、近年の標準である米国のリハビリテーション法 第 508 条に準拠するように心がけてください。この標準の詳細については http://www.sybase.com/content/1035234/PB_508_Compliance_wp.pdf のホワイトペーパを参照してください。

ステップ 3 ー Appeon for PowerBuilder による PowerBuilder アプリケーションの Web へのマイグレーションは、第 3 章「マイグレーション プロセス」に従って行います。

Appeon の Web RAD 手法では、PowerBuilder 開発環境内でデスクトップ アプリケーションを記述する作業に最も多くの時間と労力が費やされます。

2.3.1 Appeon の PowerBuilder コーディング標準

Appeon の PowerBuilder コーディング標準とは、それが Appeon によって認識され Web 言語（HTML、JavaScript と XML）に変換できる PowerBuilder コードを記述するための推奨方法です。

Appeon の PowerBuilder コーディング標準は、Appeon 機能ヘルプ（HTML にコンパイルされたヘルプ システム）に説明されています。新規 PowerBuilder アプリケーションのコードを記述するときは、このドキュメントを参照して Appeon の PowerBuilder コーディング標準に従います。

新規 PowerBuilder アプリケーションを記述した後で、PowerBuilder 開発環境に統合されている Appeon Developer のツーバーを使用して、PowerBuilder アプリケーションを自動的に Web に変換します。アプリケーションを完全に機能的にするために、PowerBuilder アプリケーションに対しさらにいくつかの修正を行う必要があるかも知れません。最終的に、PowerBuilder アプリケーションを変換し Web アプリケーションをサーバに配布して製品化します。Web への配布手順の詳細は『Appeon Developer ユーザ ガイド』を参照してください。

3 マイグレーション プロセス

3.1 イントロダクション

この章では、マイグレーションの各ステップの詳細手順を提供する代わりに、全体のプロセスを主眼として説明します。各セクションではこのマニュアルの他のセクションまたは他の Appeon マニュアルやサポート情報を参照しています。

新規 PowerBuilder アプリケーションを完全に記述して Web に配布することを目的としている場合、最初に第 2 章「PowerBuilder と Appeon による RAD Web 開発」を参照し、その後、この章の説明を参照してください。

図 3-1 には、既存の PowerBuilder アプリケーションを Web に配布する一般的なプロセスを示します。このプロセスは以下の 2 つのフェーズに分けられます。

- フェーズ 1： 計画 – Web マイグレーション目標達成手段の評価

  ステップ 1 － 必要なソフトウェアのインストール

  ステップ 2 － Appeon の一般的な制約の理解

  ステップ 3 － マイグレーション目標の定義

- フェーズ 2： 実装 –Web マイグレーション目標の実装

  ステップ 4 －オリジナル PowerBuilder アプリケーションのアップグレード

  ステップ 5 － ターゲット PowerBuilder アプリケーションの準備

  ステップ 6 － Web アプリケーションの事前構成

  ステップ 7 － 未サポート機能の修正と対策

  ステップ 8 － Web または Appeon 機能によるアプリケーションの拡張

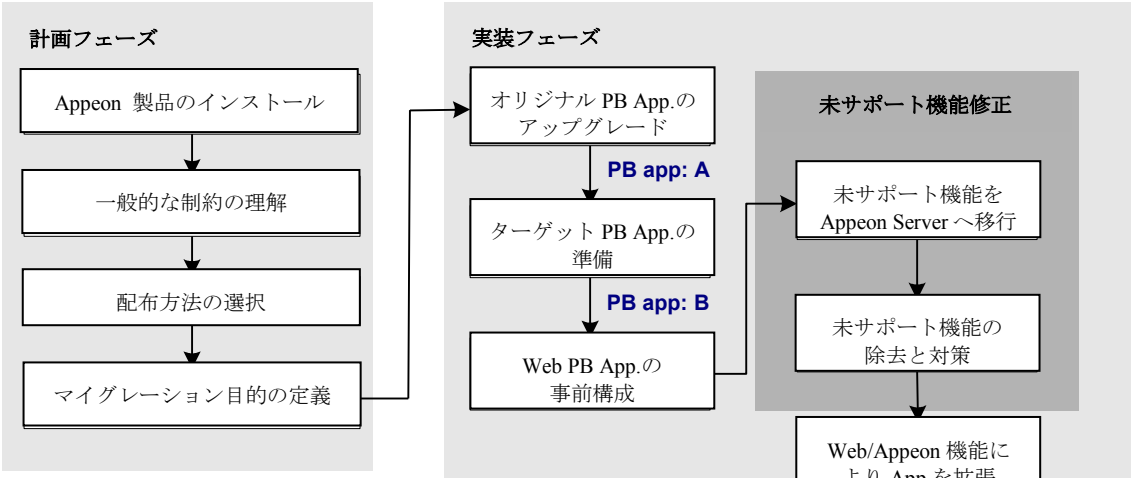
  ステップ 9 － トライアル配布とデバッグ

  ステップ 10 － Web アプリケーションのデバッグ

  ステップ 11 － 実行時パフォーマンスのチューニング

  ステップ 12 － プロダクション配布

図 3-1: Appeon の Web マイグレーション プロセス



PB app. A
適切なバージョンにアップグレードされテストされたオリジナル PB App.

PB app. B
オリジナル PB App.からマイグレートするために導き出されたターゲット PB App.

PB app. C
未サポート機能の削除または対策されたターゲット PB App.

PB app. D
Web マイグレーションのため、フルに機能アップされたターゲット PB App.

PB app. E
Web パフォーマンス、およびプロダクション配布のためファイン チューンされたターゲット PB App.

3.2 必要なソフトウェアのインストール

Web マイグレーションの最初のステップは、ソフトウェア環境をセットアップすることです。

図 3-2: Appeon for PowerBuilder が動作する標準の N-Tier Web アーキテクチャ

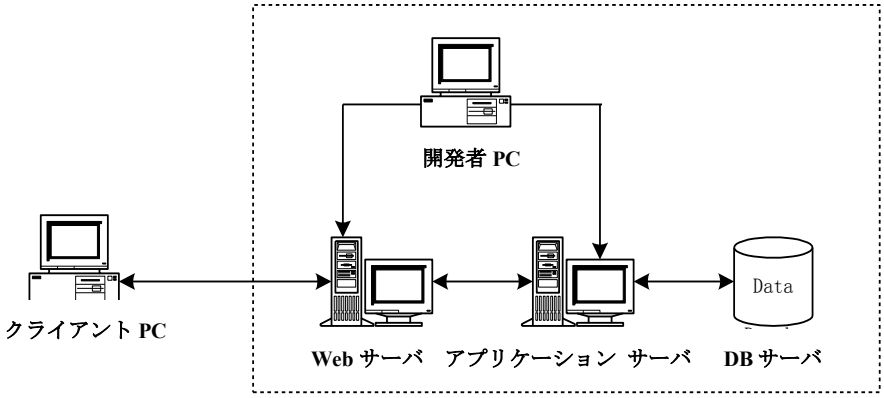


図 3-2 に示すアーキテクチャをサポートするには、次に示すソフトウェアをインストールする必要があります。

- Microsoft Windows 2000、Windows 2003 または Windows XP
- Microsoft Internet Explorer 6.0 またはそれ以上
- Sybase PowerBuilder
- アプリケーションで使用するデータベース
- EAServer
- Appeon for PowerBuilder、これには次のものが含まれます：
    - Appeon Server （Appeon Enterprise Manager、Appeon Server ステータスモニタおよび Appeon Server Web Component が含まれる）
    - Appeon Developer
    - Appeon Server Web Component
    - Appeon ヘルプ
    - Sybase EAServer
    - PDFPrinter

必要なソフトウェアをインストールする手順の詳細については、『Appeon インストレーション ガイド』を参照してください。

3.3 Appeon の一般的な制約の理解

Appeon for PowerBuilder のマイグレーション能力には、いくつかの一般的な制約があります。これらの制約は、PowerBuilder アプリケーションを Appeon による Web マイグレーションに適合させるうえで影響を与えます。Web マイグレーションに先立って、これらの制約を知ることが重要です。

表 3-1：Appeon の一般的な制約

制約事項	制約の内容	対策
コーディング スタイル	汎用化コードのようなコーディング スタイルを使用しない。	Appeon ヘルプの説明に従い、Appeon で正しく動作しないコーディング スタイルを避ける。
データベース	Appeon for PowerBuilder は次のデータベースをサポートします： Sybase ASE、Sybase ASA、Microsoft SQL Server、Oracle、IBM DB2、Sybase IQ、Informix 詳細は『インストール ガイド』を参照してください。	サポートするデータベースを使用する。
未サポート機能	Appeon ヘルプでは、サポートおよび未サポート機能のリストを提供している。	以下の Appeon ワークアラウンド ガイドに従って未サポート機能の対策を行う。 http://www.appeon.com/support/documents/workarounds/6.0/.

制約事項の詳細については、『Appeon 機能ヘルプ』内の「基本要件とアーキテクチャ要件」を参照することを推奨します。

3.4 マイグレーション目標の定義

マイグレーションには、次に示す 3 つの典型的な目標があります。

- 既存の PowerBuilder アプリケーションを Web に変換する。

- 既存の PowerBuilder アプリケーションから、ウィンドウや主要な機能などのような一部の機能を抜き出して、それを Web にマイグレートする。
- オリジナルの PowerBuilder アプリケーションからいくつかのデータウィンドウを Web にマイグレートする。

3.4.1 非 PFC アプリケーションのマイグレーション目標の定義

ステップ 1 ー オリジナルの PowerBuilder アプリケーションの PBL の合計サイズを計算します。

ステップ 2 ー 既存の PowerBuilder アプリケーション内の主要な未サポート機能を識別します。

ステップ 3 ー 既存の PowerBuilder アプリケーション内の未サポート機能の修正に必要な作業量を評価します。評価基準を次に示します。

- より複雑な PowerBuilder アプリケーションは、未サポート機能の修正作業がより難しくなります。複雑性は、深い継承レベルなどを含み高度なコーディング テクニックによって特徴づけられます。

  クライアント サイドのビジネス ロジック量が多く（たとえば、ウィンドウや非表示のコントロール内など）、そのロジックが複雑な場合（たとえば、ビジネス ルールの処理が完了するまでに 20 イベントがトリガされるなど）、ビジネス ロジックを N-Tier NVO に移行することで最も恩恵を受けることができます。これには追加の作業が必要になります。

- より多くの未サポート機能を持つ PowerBuilder アプリケーションには、より多くの修正作業が必要です。

  セクション 3.8「未サポート機能の修正と対策」には、未サポート機能への対策方法が提供されています。

ステップ 4 ー マイグレーション目標を決定します：

- ビジネスを Web に移行することが緊急の課題であるならば、最初のマイグレーション目標、すなわち、すべての PowerBuilder アプリケーションを Web に配布することを選択します。そうでない場合は 2 番目か 3 番目のマイグレーション目標を選択します。
- 前のステップの評価に基づいて、オリジナルの PowerBuilder アプリケーションの未サポート機能を修正するために多くの労力が必要な場合、アプリケー

ションの一部を Web に変換することを考慮します（目標 2）。採られるアプローチにかかわらず、Appeon によって PowerBuilder アプリケーションを Web に配布する作業は、PowerBuilder の高い生産性と特に PowerBuilder チームにとっては、それを J2EE/.NET で書き換えるより、少ない作業で済みます。

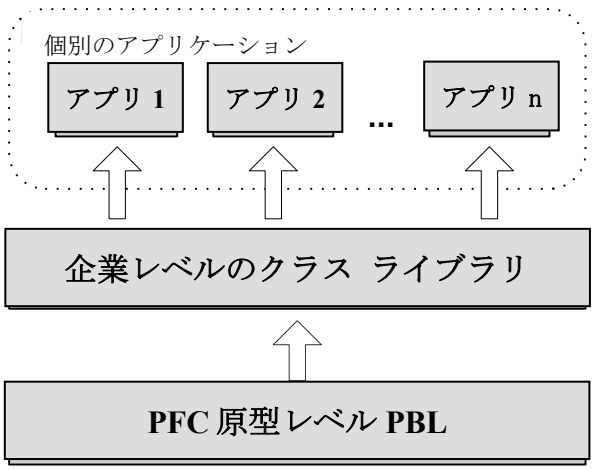
- 3 番目の目標は、アプリケーションのデータウィンドウのみを（アプリケーションから分離して）Web へ配布することです。これには小さな新しい PowerBuilder アプリケーションを作成して既存のデータウィンドウを再利用することも含みます。これは、いくつかの主要なレポート機能やデータを非常に短時間に Web 上で利用する必要がある場合には論理的なソリューションであり、最終的にはすべてのアプリケーションを配布するための重要な取り組みになります。

3.4.2 PFC アプリケーションの マイグレーション目標の定義

3.4.2.a 企業レベルの PFC アーキテクチャ

次の図は、PFC アプリケーションの典型的なアーキテクチャを示しています。

図 3-3: 典型的な PFC アプリケーションのアーキテクチャ



企業レベルのクラス ライブラリでは、PFC の原型のレベルを拡張して機能のカスタマイズと拡張が行われ、多様なアプリケーションによって再利用されます。PFC の拡張レベルの PBL は、企業レベルのクラス ライブラリの中に含まれ、PFC の原型のレベルを拡張するために利用されます。たとえば、企業アナリストは、PFC の原型のレベルと拡張レベル間に拡張レベルを追加するか、または既存の PFC 拡張レベルを使用して拡張します。

ビジネス ニーズと企業の複雑性に依存して、ときには PFC の原型のレベルには企業の部門標準やビジネス ルールが含まれるいくつかのレイヤーが入れられることで拡

張されることがあります。これらのレイヤーはすべて企業レベルのクラス ライブラリに含まれます。

3.4.2.b 段階的なマイグレーション目標の定義

PFC アプリケーションのアーキテクチャが大きな企業レイヤーを持つ場合、Appeon は、小さなアプリケーションまたは既存の PFC アプリケーションの一部を Web にマイグレートして Web 上でフレームワーク動作ができるようにすることを最初の目的にし、やがてはアプリケーションの多くの部分またはアプリケーション全体を配布できるように発展させることを推奨します。この漸進的なアプローチは、同時に発生する多くの問題を防ぎ、一方では、アプリケーションと企業レベルのクラス ライブラリに内在する未サポート機能への対応を可能にします。

ステップ 1 － 次の 3 つのパートから構成される小さな PFC アプリケーションを配布します。

- 単一のアプリケーションから分離した少量のアプリケーション固有のロジック
- 企業レベルのクラス ライブラリ
- PFC 原型レベル

このステップでは、アプリケーション固有のロジックを小さくシンプルにして、PFC 原型レベルと企業レベルのクラス ライブラリの Web へのマイグレートに焦点をあてます。PFC 原型レベルと企業レベルのクラス ライブラリは、すべての企業アプリケーションの基本クラスであり非常に重要です。

ステップ 2 － アプリケーションのサイズと複雑さを次第に増やし、単一アプリケーションの Web への配布に焦点をあてます。

3.5 オリジナル アプリケーションのアップグレード

Appeon 6.0 は PowerBuilder 9、10、10.2 および 11.1 をサポートしています。Web に配布されるどんな PowerBuilder ソースコードも、最初に、必要な PowerBuilder バージョンと 100% 互換を保つようにアップグレードされる必要があります。アプリケーションがアップグレードされない場合、配布した Web アプリケーションの実行時エラーに直面するでしょう。

3.5.1 古い PBL のアップグレード

以前のバージョン（たとえば PowerBuilder 5 など）の PBL を最新のバージョンでロードするとき、多くのエラー メッセージが PowerBuilder のアウトプット ウィンドウに表示されることがあります。前身オブジェクトからその後身オブジェクトすべてへの継承の問題により、実際に 1 つのコードからいくつかのエラーが報告されるかも知れません。

PowerBuilder アプリケーションのアップグレード時に見られる共通のエラーのひとつは文字列の破損です。これらのエラーを訂正するには、ソースコードを編集します（PowerBuilder の出力ウィンドウに表示されたエラー項目上のコンテキスト メニューから［ソースの編集］を選択します）。アプリケーション ターゲットのフル構築を行うと、エラーが表示されなくなります。

3.5.2 PFC アプリケーションのアップグレード

Appeon で Web へマイグレートしようとする PFC アプリケーションは、PFC 9、10、10.2 または 11.1 に準拠するようにアップグレードする必要があります。いくつかのレガシーな PFC アプリケーションでは PFC 5.0 の PBL に基づいており、新しい PFC 9、10、10.2 または 11.1 PBL を使用するためにはこれらのアプリケーションをアップグレードしなければなりません。

PFC の PFCOLD.PBL ライブラリには、Appeon の Web マイグレーションではサポートされない古いオブジェクトが含まれています。オリジナルの PowerBuilder アプリケーションが PFCOLD.PBL のオブジェクトを使用する PFC アプリケーションである場合、このアプリケーションを PFC 9、10、10.2 または 11.1 にアップグレードするときにこれらのオブジェクトを削除するべきです。

3.6 ターゲット アプリケーションの準備

このステップでは、ターゲット アプリケーションを作成するために、オリジナルの PowerBuilder アプリケーションを処理する必要があります。この作業では、*Appeon ヘルプ*内の Appeon アーキテクチャ ガイドラインに沿って、オリジナルの PowerBuilder アプリケーションをターゲット アプリケーションにすることに焦点をあてます。

3.6.1 PFC アプリケーションに必要とする特別な処理

Appeon は大部分の PFC 機能をサポートしています。ほとんどの場合は PFC ソースコードを修正しないで Appeon の環境で利用できます。しかしながら、いくつかの未サポート機能が存在します。これらの未サポート機能について Appeon はワークアラ

ウンドまたは代替案のソリューションを提供しています。PFC フレームワークは CS 環境で実行されているか、Web 環境で実行されているか、自動的に環境を識別する機能を提供しています。CS 環境ならオリジナルのソースを実行し、Web 環境なら修正後のソースが実行されます。これらの未サポート機能を修正後、通常の PowerBuilder アプリケーション同様に PFC アプリケーションを配置することができます。

3.6.1.a 環境の自動識別

PFC アプリケーションを PowerBuilder と Appeon の両方の環境で稼働させるための考え方はアプリケーションが稼働している環境を識別するためのグローバル ファンクションを追加します。グローバル ファンクションはオリジナルのコードを実行するか、Appeon 対応のコードを実行するかを識別します。

以下のグローバル ファンクション appeongetclienttype を pfcmain.pbl に追加します。

```
Global function string appeongetclienttype();
     Return 'PB'
End function
```

3.6.1.b 推奨する PFC の修正方法

Appeon 未サポートおよび PFC の機能を PowerBuilder と Appeon の両方の環境で実行できることを確保するために、PFC コードをどのように修正すればよいのか。以下に詳細な方法を記述します。既存の CS ユーザが Appeon 用に修正した部分の影響を受けないように、コードの修正は絶対必要な状況と環境の自動識別を使用する場合のみ行うことができます。

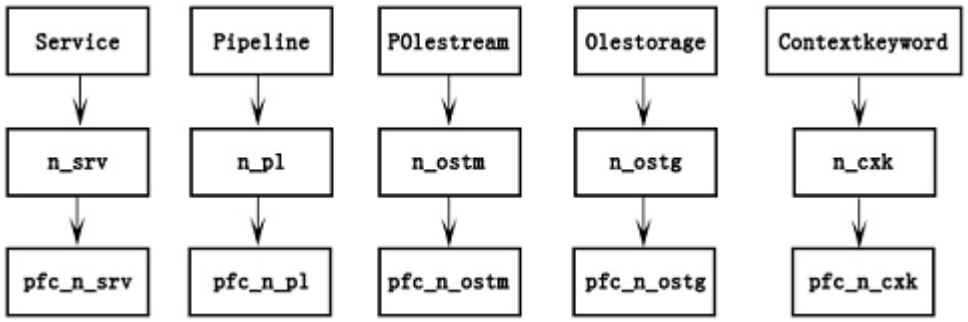
Appeon は以下の方法を提案します。

```
If appeongetclienttype = 'PB' Then
         Original code in PFC
Else
         Appeon compliant code in PFC
End If
```

3.6.1.c 未サポートのユーザオブジェクト

PFC フレームワークは、次の未サポートのユーザオブジェクトが含まれています（最後 2 行）：

図 3-4: PFC にある未サポートのユーザオブジェクト



注意： これらのユーザオブジェクトが既に定義されていますが、PFC フレームワークで利用することがないので、修正しないでください。アプリケーションでこれらのオブジェクトの利用を避けてください。

3.6.1.d 未サポートの標準クラスオブジェクト

次に示す 4 つの標準クラスオブジェクトがサポートされていません：

- Contextinformation
- Classdefinition
- Scriptdefinition
- Variabledefinition

推奨：次のオブジェクト関数への参照を避けることによって、機能への影響を減少させることができます：

- pfcmain.pbl¥pfc_w_master¥of_setresize
- pfcapsrv.pbl¥pfc_n_cst_security¥of_setsecurity
- pfcapsrv.pbl¥pfc_n_cst_metaclass¥of_findmatchingevent
- pfcapsrv.pbl¥pfc_n_cst_metaclass¥of_findmatchingvariable
- pfcapsrv.pbl¥pfc_n_cst_metaclass¥of_getancestorclasses
- pfcapsrv.pbl¥pfc_n_cst_metaclass¥of_isfunctiondefined
- pfcapsrv.pbl¥pfc_n_cst_metaclass¥of_iseventimplemented
- pfcapsrv.pbl¥pfc_n_cst_metaclass¥of_iseventdefined

- pfcapsrv.pbl¥pfc_n_cst_metaclass¥of_isancestorclass

Classdefinition を使用したい場合、次のワークアラウンドを利用してください：

元のスクリプト：

```
lb_defined = inv_metaclass.of_isFunctionDefined &
(lpo_tocheck.ClassDefinition, "of_PostUpdate", ls_args)
```

変更後のスクリプト：

```
If lpo_tocheck.TriggerEvent("pfc_descendant") = 1 Then
        lb_defined = True
Else
        lb_defined = False
End If
```

3.6.1.e 未サポートオブジェクトのプロパティ

テーブル 3-1：未サポートされないオブジェクトのプロパティ

オブジェクト	プロパティ	コール回数	説明
Application	ToolbarUserControl	4	修正不要
	ToolbarText	5	
	ToolbarPopMenuText	3	
	DwMessageTitle	4	
	DdeTimeOut	3	
OLEControl	ClassShortName	1	修正不要
	ClassLongName	1	
Transaction	Lock	3	修正不要
	Sqlreturndata	1	
Message	Returnvalue	1	修正不要
	Number	1	
	Handle	1	
Connection	Trace	2	修正不要
	Options	2	
	Connectstring	2	
GraphicObject	Classdefinition	1	修正不要

テーブル **3-2**：修正必要な未サポートオブジェクトプロパティ

オブジェクト	プロパティ	コール回数	説明

Window	Classdefinition	1	**修正必要**　詳細は以下を参照。
			詳細情報：
			場所: Of_setresize(boolean) function of pfc_w_master in pfcmain.pbl
			Appeon 未サポートコード： Lcd_class = this.ClassDefinition
			行番号： 45
			以下のように修正：
			if AppeonGetClientType( ) = 'PB' then
			classdefinition lcd_class
			lcd_class = this.ClassDefinition
			
			li_vars = UpperBound (lcd_class.VariableList)
			For li_v = 1 to li_vars
			If lcd_class.VariableList[li_v].Name = "width" Then li_origwidth = Integer (lcd_class.VariableList[li_v].InitialValue)
			If lcd_class.VariableList[li_v].Name = "height" Then li_origheight = Integer (lcd_class.VariableList[li_v].InitialValue)
			If li_origwidth > 0 And li_origheight > 0 Then Exit
			Next
			inv_resize.of_SetOrigSize (li_origwidth, li_origheight)
			else
			inv_resize.of_SetOrigSize (this.width, this.height)
			end if

PowerObject	Classdefinition	7	**修正必要**　詳細は以下を参照 詳細情報： **場所 1：** Of_validation(powerobject) function of pfc_n_cst_luw in pfcapsrv.pbl **Appeon 未サポートコード：** Lcd_class = this.ClassDefinition **行番号：** 123 **以下のように修正：** if AppeonGetClientType( ) = 'PB' then

			lb_defined = inv_metaclass.of_isFunctionDefined & (lpo_tocheck.ClassDefinition, "of_Validation", ls_args)
			Else
			lb_defined = True
			End If
			
			場所 2： Of_updatespending(powerobject) function of pfc_n_cst_luw in pfcapsrv.pbl
			Appeon 未サポートコード： Lb_defined = inv_metaclass.of_isFunctionDefined(lpo_tocheck .ClassDefinition, "of_UpdatesPending", ls_args)
			行番号： 130
			以下のように修正：
			if AppeonGetClientType( ) = 'PB' then
			lb_defined = inv_metaclass.of_isFunctionDefined & (lpo_tocheck.ClassDefinition, "of_UpdatesPending", ls_args)
			Else
			lb_defined = True
			End If
			
			場所 3： Of_updateprep(powerobject) function of pfc_n_cst_luw in pfcapsrv.pbl
			Appeon 未サポートコード： Lb_defined = inv_metaclass.of_isFunctionDefined(lpo_tocheck .ClassDefinition, "of_UpdatePrep", ls_args)
			行番号： 122
			以下のように修正：
			if AppeonGetClientType( ) = 'PB' then
			lb_defined = inv_metaclass.of_isFunctionDefined & (lpo_tocheck.ClassDefinition, "of_UpdatePrep", ls_args)
			else
			lb_defined = True
			End If
			
			場所 4：
			Of_update(powerobject,boolean,boolean) function of pfc_n_cst_luw in pfcapsrv.pbl
			Appeon 未サポートコード： Lb_defined = inv_metaclass.of_isFunctionDefined(lpo_tocheck .ClassDefinition, "of_Update", ls_args)

			行番号： 145
			以下のように修正：
			if AppeonGetClientType( ) = 'PB' then
			lb_defined = inv_metaclass.of_isFunctionDefined &
			(lpo_tocheck.ClassDefinition, "of_Update", ls_args)
			Else
			lb_defined = True
			End If
			
			場所 **5**： Of_postupdate(powerobject) function of pfc_n_cst_luw in pfcapsrv.pbl
			Appeon 未サポートコード： Lb_defined = inv_metaclass.of_isFunctionDefined(lpo_tocheck .ClassDefinition, " of_PostUpdate ", ls_args)
			行番号： 123
			以下のように修正：
			if AppeonGetClientType( ) = 'PB' then
			lb_defined = inv_metaclass.of_isFunctionDefined &
			(lpo_tocheck.ClassDefinition, " of_PostUpdate", ls_args)
			else
			lb_defined = True
			End If
			
			場所 **6**： Of_isselfupdatingobject(powerobject) function of pfc_n_cst_luw in pfcapsrv.pbl
			Appeon 未サポートコード： Lb_defined = inv_metaclass.of_isFunctionDefined(apo_control .ClassDefinition, "of_Update", ls_args)
			行番号： 60
			以下のように修正：
			if AppeonGetClientType( ) = 'PB' then
			lb_defined = inv_metaclass.of_isFunctionDefined &
			(apo_control.ClassDefinition, "of_Update", ls_args)
			else
			lb_defined = True
			End If
			
			場所 **7**： Of_accepttext(powerobject,boolean) function of pfc_n_cst_luw in pfcapsrv.pbl

			Appeon 未サポートコード： Lb_defined = inv_metaclass.of_isFunctionDefined(lpo_tocheck .ClassDefinition, "of_AcceptText", ls_args)
			行番号： 128
			以下のように修正：
			if AppeonGetClientType( ) = 'PB' then
			lb_defined = inv_metaclass.of_isFunctionDefined &
			(lpo_tocheck.ClassDefinition, "of_AcceptText", ls_args)
			else
			lb_defined = True
			End If

DataWindow control	Hsplitscroll	1	**修正必要** 詳細は以下を参照。
			場所： Of_position(dragobject,boolean) function of pfc_n_cst_dropdown in pfcapsrv.pbl
			Appeon 未サポートコード： If ldw_object.hsplitscroll
			行番号： 237
			以下のように修正：
			If AppeonGetClientType( ) = 'PB' then
			li_hsplit = Integer (ldw_object.Describe("DataWindow.HorizontalScrollSplit"))
			li_hpos1 = Integer (ldw_object.Describe("DataWindow.HorizontalScrollPosition"))
			li_hpos2 = Integer (ldw_object.Describe("DataWindow.HorizontalScrollPosition2"))
			If ldw_object.hsplitscroll Then
			If li_hsplit > 4 and li_pointerx > li_hsplit Then
			li_hpos = li_hpos2 - li_hsplit - of_GetSystemSetting(DW_HSPLITBAR_WIDTH)
			Else
			li_hpos = li_hpos1
			End If
			Else
			li_hpos = li_hpos1
			End If
			Else
			li_hpos = 0
			End if

3.6.1.f 未サポートのオブジェクトの関数

テーブル 3-3：未サポートのオブジェクトの関数

オブジェクト	関数	コール回数	説明
NonVisualObject	GetContextService	1	修正不要
Window	Settoolbar	5	修正不要
	Gettoolbar	3	
	Gettoolbarpos	2	
	Settoolbarpos	2	
SingleLineEdit	Canundo	1	修正不要
RichTextEdit	Canundo	1	修正不要
OLEControl	UpdateLinksDialog	1	修正不要
	PasteSpecial	1	
	Paste	1	
	Cut	1	
	Copy	1	
EditMask	Canundo	1	修正不要
DataStore	ReselectRow	1	修正不要

3.6.1.g 未サポートのオブジェクトのイベント

テーブル 3-4：未サポートのオブジェクトのイベント

オブジェクト	イベント	説明
RichTextEdit	PrintHeader	修正不要
	FileExists	
ListView	Sort	修正不要
Application	SystemError	修正不要

3.6.1.h 未サポートのシステム関数

テーブル 3-5：未サポートのシステム関数

システム関数	コール回数	説明
ShowHelp	18	修正不要
FindClassDefinition	6	修正不要

3.6.1.i 未サポートの共有変数

テーブル 3-6：未サポートの共有変数

共有変数	コール回数	説明
snv_property	5	修正不要

3.6.1.j その他

テーブル 3-7：その他

関数	コール回数	説明
dwObject.GetChild	1	修正不要

テーブル 3-8：修正必要な未サポート関数

関数	コール回数	説明

dwObject.Print	3	**修正必要**　詳細は以下を参照。
		詳細情報：
		場所 1： Pfc_print() event of pfc_u_dw in pfcmain.pbl
		Appeon 未サポートコード： Li_rc = lds_selection.Print(true, true)
		行番号： 75
		以下のように修正：
		if IsValid (lds_selection) then
		If AppeonGetClientType( ) = 'PB' then
		li_rc = lds_selection.Print (true, true)
		else
		li_rc = lds_selection.Print (true)
		end if
		destroy lds_selection
		else
		If AppeonGetClientType( ) = 'PB' then
		li_rc = this.Print (true, true)
		else
		li_rc = this.Print (true)
		end if
		end if
		
		場所 2： Pfc_print() event of pfc_n_ds in pfcmain.pbl
		Appeon 未サポートコード： Li_rc = lds_selection.Print(true, true)
		行番号： 87
		以下のように修正：
		if IsValid (lds_selection) then
		If AppeonGetClientType( ) = 'PB' then
		li_rc = lds_selection.Print (true, true)
		else
		li_rc = lds_selection.Print (true)
		end if
		destroy lds_selection
		else
		If AppeonGetClientType( ) = 'PB' then
		li_rc = this.Print (true, true)
		else
		li_rc = this.Print (true)
		end if
		end if

3.6.1.k Differently behaved features

オブジェクト		説明

MDI frames	WorkSpaceX
	WorkSpaceY
		修正必要 詳細は以下を参照。
		場所： Of_setposition () function of pfc_w_statusbar in pfcwnsrv.pbl
		行番号： 72
		提案修正方法：
		ステップ 1．以下のようなフォーマットの rect ストラクチャを pfcwnsrv.pbl に追加します。
		global type rect from structure
		long left
		long top
		long right
		long bottom
		end type
		ステップ 2．Pfc_w_statusbar でローカル外部関数を宣言します。
		Function long GetWindowRect (long hwnd , ref rect lpRect) Library "user32"
		ステップ 3．現在のスクリプトを下記のように書き直します。
		long ll_bottompos,ll_rtn,ll_TopPos,ll_LeftPos
		rect lr_WinRect
		ll_rtn = getwindowrect(handle(iw_parentwindow) , lr_WinRect)
		ll_LeftPos = PixelsToUnits(lr_WinRect.Left, XPixelsToUnits!)
		ll_TopPos = PixelsToUnits(lr_WinRect.Top, YPixelsToUnits!)
		ll_bottompos = PixelsToUnits(lr_WinRect.bottom, YPixelsToUnits!)
		If AppeonGetClientType( ) = 'PB' then
		// The Y Position of the Status Bar is the Bottom of the Frame Window
		// minus the MicroHelpHeight minus the MicroHelpBorderHeight.
		ll_microhelp_ypos = &
		(iw_parentwindow.y + iw_parentwindow.height + li_ypos_extra) - &
		(iw_parentwindow.mdi_1.microhelpheight + ii_borderheight)
		// The desired X Position of the Status Bar is the Frame Right End minus
		// the StatusBar window. Also subtract the extra spacing on win95.
		ll_desiredstatubar_xpos = &
		iw_parentwindow.x + iw_parentwindow.workspacewidth() + &

		ii_borderwidth - (ii_winmaxwidth + 12 + li_xpos_extra)
		
		// The Frame X Position.
		ll_frame_xpos = iw_parentwindow.x + (2*ii_borderwidth) + 16
		
		if ll_desiredstatubar_xpos < ll_frame_xpos then
		// Status Bar would extend to the left of the frame.
		this.move(ll_frame_xpos , ll_microhelp_ypos)
		// Make the StatuBar the width of the frame.
		this.width = workspacewidth(iw_parentwindow) - (20 + li_xpos_extra)
		else
		// Normal as large as defined Status Bar window.
		this.move(ll_desiredstatubar_xpos , ll_microhelp_ypos)
		this.width = ii_winmaxwidth
		end if
		
		else
		// The Y Position of the Status Bar is the Bottom of the Frame Window
		// minus the MicroHelpHeight minus the MicroHelpBorderHeight.
		ll_microhelp_ypos = &
		(ll_TopPos + iw_parentwindow.height + li_ypos_extra) - &
		
		(iw_parentwindow.mdi_1.microhelpheight + ii_borderheight)
		// The desired X Position of the Status Bar is the Frame Right End minus
		// the StatusBar window. Also subtract the extra spacing on win95.
		ll_desiredstatubar_xpos = &
		ll_LeftPos + iw_parentwindow.workspacewidth() + &
		ii_borderwidth - (ii_winmaxwidth + 12 + li_xpos_extra)
		ll_frame_xpos = ll_LeftPos + (2*ii_borderwidth) + 16
		
		if ll_desiredstatubar_xpos < ll_frame_xpos then
		// Status Bar would extend to the left of the frame.
		this.move(ll_frame_xpos , ll_bottompos - 70)
		// Make the StatuBar the width of the frame.
		this.width = iw_parentwindow.workspacewidth() - (20 + li_xpos_extra)

		else
		// Normal as large as defined Status Bar window.
		this.move(ll_desiredstatubar_xpos , ll_bottompos - 70)
		this.width = ii_winmaxwidth
		end if
		end if

MDI frames	Resize Service	**修正必要** 詳細は以下を参照。
		ステップ 1．w_examplemain オブジェクトの wintype プロパティを MDI から main へ変更します。
		ステップ 2．appexmfe.pbl の n_exampleappmanager の中の pfc_open イベントを以下のように修正します。
		Current scripts in PFC:
		Open (w_examplemain)
		Appeon compliant scripts in PFC:
		Open (w_frame)
		opensheet(w_examplemain, w_frame, 0, layered!)
		ステップ 3．appexmfe.pbl の w_examplemain オブジェクトのなかの pfc_preopen イベントから exmmain.pbl の w_frame オブジェクトの pfc_preopen イベントへ以下のスクリプトをコピーします。それからイベント内の下記のスクリプトをコメントアウトします。
		// Set the frame window with the application manager.
		gnv_app.of_SetFrame(w_examplemain)
		gnv_app.of_SetMicrohelp(true)
		//// Enable the Status Bar Services
		of_SetStatusBar(true)
		if IsValid(inv_statusbar) then
		inv_statusbar.of_SetBorderType(0)
		inv_statusbar.of_SetGapWidth(40)
		inv_statusbar.of_Register('debugsrv', 'bitmap', 'dbsvc.bmp', 80)
		inv_statusbar.of_Register('sqlspysrv', 'bitmap', 'sssvcoff.bmp', 80)
		inv_statusbar.of_Register('debuglogwin', 'bitmap', 'dlgwnoff.bmp', 80)
		inv_statusbar.of_Register('sqlspywin', 'bitmap', 'sswinoff.bmp', 80)
		if gnv_app.ienv_object.Win16 then
		inv_statusbar.of_SetGDI(true)
		inv_statusbar.of_SetUser(true)
		end if
		inv_statusbar.of_SetTimer(true)
		end if

3.6.2 マイグレーション目標に基づくアプリケーションの処理

次のテーブルには、マイグレーション目標に基づいて行わなければならないタスクが示されています。

表 3-2：マイグレーション目標に基づくアプリケーションの処理

マイグレーション計画	アプリケーションに対して行う処理
オリジナル アプリケーション全体	1. オリジナル PowerBuilder アプリケーションをテストして、機能やユーザ インタフェースの問題を訂正します。 注意: このステップは、レガシーPowerBuilder アプリケーションに内在する問題、またはレガシーPowerBuilder アプリケーションのアップグレードにより引き起こされる問題の検出と除去のため行われます。 2. PowerBuilder 環境で、オリジナル PowerBuilder アプリケーションのフル構築と最適化を行います。
オリジナル アプリケーションの一部	1. オリジナル PowerBuilder アプリケーションから希望する部分を抽出して、新しい PowerBuilder アプリケーション ターゲットに入れます。 2. バグを無くし期待する機能が動作するように、アプリケーションをテストします。 3. PowerBuilder 環境で、この抽出した部分のフル構築と最適化を行います。

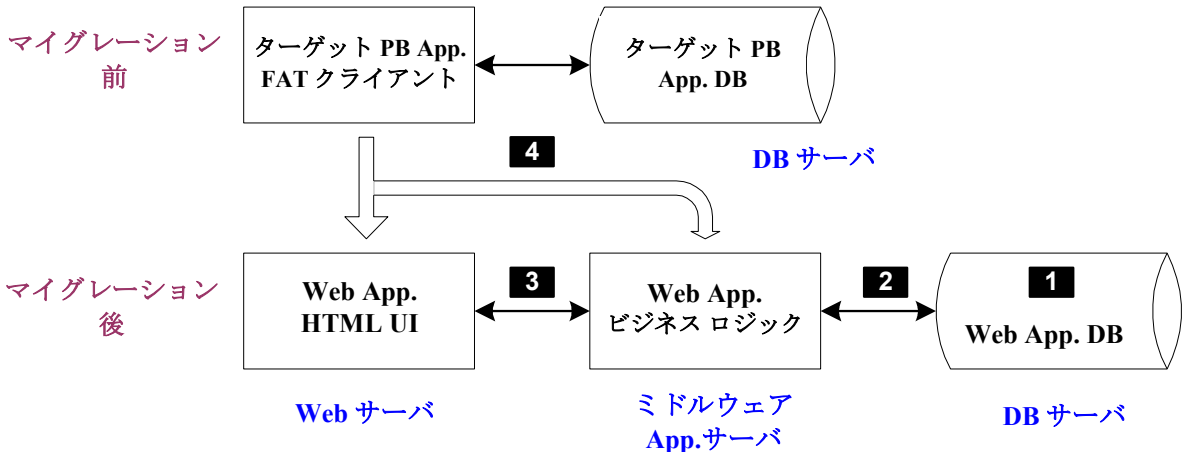
オリジナル アプリケーションのいくつかのデータウィンドウ	1. オリジナル PowerBuilder アプリケーションと同じワークスペース内に、新しい PowerBuilder アプリケーション ターゲットを作成します。 2. 望むデータウィンドウ オブジェクトをオリジナル PowerBuilder アプリケーションから新しい PowerBuilder アプリケーション ターゲットに移動します。 3. 新しい PowerBuilder アプリケーションにウィンドウ、メニュー、および一般的な UI を追加し、簡単なビジネス ロジックを記述して機能するようにします。 4. 新しい PowerBuilder アプリケーションをテストして、それらの機能とユーザ インタフェースの問題を訂正します。 5. PowerBuilder 環境で、この新しい PowerBuilder アプリケーションのフル構築と最適化を行います。

3.7 Web アプリケーションの事前構成

3.7.1 なぜ事前構成が必要か

下図に示す Appeon のマイグレーション ソリューションは、なぜこれらの事前構成が必要かを説明しています。

図 3-5: Appeon Web マイグレーションの前と後



Web マイグレーションの前には、ターゲット PowerBuilder アプリケーションの重いクライアントがデータベース サーバと相互に対話を行います。アプリケーションが分散されている場合、クライアントはまた、ターゲット アプリケーションのホスト

であり重要なビジネス ロジックを実行するアプリケーション サーバと相互に対話を行います。

Web マイグレーションの後には、Web アプリケーションは、ミドルウェアとしてビジネス ロジックを保持する Appeon Server を含む N-Tier の Web アーキテクチャを持ちます。この構造は、Web アプリケーションが Internet Explorer などの Web ブラウザ経由でアクセスされるため、ブラウザ サーバと呼ばれます。

N-Tier の Web 環境内に、期待する Web アプリケーションの機能を作るには、HTML ユーザ インタフェースとビジネス ロジックのホスティングのみならずそれ以上の機能を必要とします。

3.7.2 4 つの事前構成作業

図 3-5 に示される番号は、事前構成に必要な 4 つのステップを表わしています。

タスク#1：Web アプリケーションのデータベース サーバを手作業で設定します (非-Appeon 作業)。このタスクは、PowerBuilder アプリケーションのデータベース サーバの設定と同じです。

タスク#2：バックエンド DB とミドルウェアである Appeon Server 間のコミュニケーションを手作業で設定します。特に、JDBC コネクション キャッシュを設定します（非-Appeon 作業）。

この作業の手順については、『Appeon Server コンフィグレーション ガイド』を参照してください。

タスク#3：Web アプリケーションの UI（Web サーバ上でホストされる）と Web アプリケーションのビジネス ロジック（Appeon Server 上でホストされる ）間のコミュニケーションを設定します。これはアプリケーションのデータベース構成を介して行われます（Appeon 作業）。

この作業の手順については、『Appeon Server コンフィグレーション ガイド』を参照してください。

タスク#4：ターゲット PowerBuilder アプリケーションの変換と配布を自動的に行うように Appeon Developer を構成します（Appeon 作業）。

この作業の手順については、『Appeon Developer ユーザ ガイド』を参照してください。

3.8 未サポート機能の修正と対策

Appeon では、いくつかの PowerBuilder 機能はサポートされていません。その理由は：

1. デスクトップ アプリケーションと Web アプリケーション間には、アーキテクチャ的な相違があります。デスクトップ アプリケーションに対して、Web アプリケーションの機能はしばしば制約されます。

2. Appeon for PowerBuilder の現行バージョンは、すべての PowerBuilder プログラミング機能をサポートしているわけではありません。

未サポート機能が修正されなかった場合は、生成された Web ファイル内でコメントアウトされます。未サポート機能を含んでいるコードとこれらの未サポート機能に依存するコードは動作しません。

このステップでは、アプリケーションの実行上いくつかの機能的な影響を与える未サポート機能の修正を行います。いくつかの外見的な機能、たとえば “Border” プロパティのような機能がアプリケーションに影響を与えない場合は修正プロセスを無視できます。

3.8.1 未サポート機能の識別

以下に示す 4 つの情報源は、PowerBuilder アプリケーション内の Appeon 未サポート機能の識別をガイドしています。

- 未サポート機能解析 (UFA) レポート ― このツールは Appeon Developer のツールバーに含まれています。このツールは PowerBuilder アプリケーションの未サポート機能をスキャンして、未サポート機能の変更を支援します。UFA レポートの使用方法は、『Appeon Developer ユーザ ガイド』を参照してください。

- Code Insight ― このツールもまた Appeon Developer のツールバーに含まれています。このツールは PowerBuilder ペインタ内で直接未サポート機能を識別するうえで役に立ちます。Code Insight の使用方法は、『Appeon Developer ユーザ ガイド』を参照してください。

- 『Appeon 機能ヘルプ』― このヘルプ ファイルは、Appeon Developer のツールバーの［ヘルプ］ボタンをクリックすることにより開始される検索可能な HTML ファイルです。これはサポートおよび未サポート機能の詳細をリストし、Appeon のコーディング標準に沿って PowerBuilder コードを書き換えるための情報を提供します。

3.8.2 機能の修正方法

未サポート コードは、http://www.appeon.com/support/documents/workarounds/6.0/ で利用可能な『Appeon ワークアラウンド ガイド』の手順に従って修正を行います。このガイドはいくつかの共通な未サポート機能のサンプルとこれらの対策を提供します。

次の 2 つの修正方法は『Appeon ワークアラウンド ガイド』に含まれており、その重要性からハイライトをここに示します：

- 未サポート機能を PowerBuilder のノンビジュアル ユーザオブジェクト(NVO)にカプセル化して、この NVO を Appeon Server に配布します。

  この方法は、多くの未サポート機能 (ブラウザの制約と Appeon の制約)の対策になります。この方法の手順は、セクション 5.2「未サポート機能を N-Tier NVO として Appeon Server へ移行」を参照してください。

- 分散データウィンドウへの対策は Appeon の提供する Workaround PBL を使用します。

  Appeon の Workaround PBL は、分散データウィンドウをサポートするために、2 つのオブジェクト appeondatawindow と appeondatastore と 4 つの関数 GetFullState、SetFullState、GetChanges および SetChanges を提供しています。この方法の手順は、セクション 5.4「分散データウィンドウを持つ分散アプリケーションのマイグレーション」を参照してください。

3.9 Web または Appeon 機能によるアプリケーションの拡張

これはオプションのステップです。いくつかの拡張機能が配布されたアプリケーションに自動的に含まれますが、いくつかの機能は最初に PowerBuilder アプリケーション内で変更される必要があります。次のリストは、配布されたアプリケーションを効果的に機能させるために、PowerBuilder アプリケーションに追加されるべき主要な拡張機能です。

- Appeon Server におけるすべてのアプリケーションの実行を管理するために呼出される Appeon Server オープン インタフェース
- クライアント情報を取得するため、または Appeon DataWindow メニューを有効にするために呼出される Appeon クライアント機能
- コマンドライン引数による他のアプリケーションとのインテグレーション、または HyperLink コントロール

第 6 章「Web または Appeon 機能によるアプリケーションの拡張」では、配布されたアプリケーション内に Web または Appeon 機能を実装する手順を提供します。

3.10 トライアル配布とデバッグ

前のステップが完了すると、ターゲット PowerBuilder アプリケーションの最初のトライアル配布の準備ができたことになります。

トライアル配布は、最後のプロダクション配布の前に行う中間的な配布です。Appeon Web アプリケーションの配布と実行の詳細は、『Appeon Developer ユーザ ガイド』を参照してください。

3.10.1 分散アプリケーションの特別な配布ステップ

分散アプリケーションには、分散データウィンドウを含むか含まないかに関わらず、特別な配布ステップが必要です。分散アプリケーションの配布手順は、第 5 章「分散アプリケーションの構築とマイグレーション」を参照してください。

3.10.2 配布したアプリケーションのデバッグ

トライアル配布では問題が発生する可能性があります。たとえば、Web アプリケーションの機能がターゲット PowerBuilder アプリケーションの動作と異なるか、またはまったく動作しないことがあります。これらの問題は、通常、未サポート機能または既知の問題（マニュアル、『Appeon リリース ノート』に説明）によって引き起こされます。UFA レポートはクライアント/サーバ アプリケーションとブラウザ/サーバ アプリケーション間のすべての小さな相違を検出できないため、配布アプリケーションに未サポート機能が含まれる可能性があります。

基本は、Web アプリケーションのデバッグと問題となる原因の発見にあります。Appeon デバッガを利用して Appeon Web アプリケーションをデバッグすることができます。デバッグの手順については、『Appeon Developer ユーザ ガイド』の「Appeon Web アプリケーションのデバッグ」を参照してください。

デバッグの結果に従ってターゲット PowerBuilder アプリケーションを修正後、Web アプリケーションにまだエラーがあるかを見るために、トライアル配布を行います。Web アプリケーションの機能が適切に動作する前は、“トライアル配布 → Web アプリケーションのデバッグ”のプロセスを繰り返して行います。

3.11 実行時パフォーマンスのチューニング

クライアント/サーバ アプリケーションと Web アプリケーション間のアーキテクチャの相違にはそれぞれ利点と欠点があります。一般には、Web アプリケーションは

良いサーバ スケーラビリティを提供しますが、クライアント/サーバ アプリケーションは優れたユーザ操作性（リッチ GUI など）とより良い実行時パフォーマンスを提供します。

Appeon の Web アプリケーションは、N 層 Web アーキテクチャのスケーラビリティと共にリッチな PowerBuilder GUI を提供します。リッチな PowerBuilder GUI は、優れたユーザ操作性と高い生産性を提供します。サーバ スケーラビリティは J2EE アプリケーションと同等かそれ以上ですが、良くないならば、Appeon ではより多くの処理をサーバからクライアントに移したためです。『Appeon パフォーマンス チューニング ガイド』には、Web パフォーマンスを劣化させる PowerBuilder プログラミング機能とコーディング スタイル、その提唱と対策が説明されています。より詳細の情報については、『Appeon パフォーマンス チューニング ガイド』を参照してください。

3.12 プロダクション配布

プロダクション配布は、Web マイグレーション プロセスの最終ステップになります。このステップでは、ターゲット PowerBuilder アプリケーションをプロダクション サーバへ配布して、一般ユーザが Web アプリケーションを利用できるようにします。

プロダクション サーバへの最終的な配布を始める前に、Appeon Developer のアプリケーション プロファイル内で次の設定を行います：

- Web ファイルの生成モードをリリースに設定 － これは Web アプリケーションの JavaScript コードを暗号化し、組織のビジネス ルールと知的資産を保護します。
- すべてのレポート オプションを OFF に設定 － これは Web ファイルのサイズを小さくし、配布した Web アプリケーションの実行時パフォーマンスを高めます。

アプリケーションの配布手順の詳細については、『Appeon Developer ユーザ ガイド』を参照してください。

4 マイグレーション FAQ

4.1 新規 Web アプリケーションを迅速に構築するには？

ステップ 1 － 新しいアプリケーションの作成

PowerBuilder 環境で、PowerBuilder のクライアント/サーバ アプリケーションのワークスペースを新しく作成します。

ステップ 2 － アプリケーションのコードを記述します。

Appeon developer のツールバーに含まれている Code Insight を有効にして、コードを記述します。Code Insight を使用すると、記述するコードが Appeon でサポートされているかどうかを簡単に知ることができます。

ステップ 3 － Appeon 配布ウィザードでアプリケーションを自動的にマイグレーションします。

このステップ後には、PowerBuilder アプリケーションの機能は Appeon が 100%サポートできるものになっていますので、未サポート機能の修正は必要ありません。このステップでは、Appeon Developer の配布ウィザードを使用して、クライアント/サーバ アプリケーションをブラウザ/サーバ アプリケーションに簡単にマイグレートできます。

4.2 Appeon はすべての PowerBuilder 機能をサポートするか？

Appeon の現行バージョンはすべての PowerBuilder 構文をサポートしているわけではありません。未サポートの PowerBuilder 構文は次の二つのカテゴリに分けられます。

カテゴリ 1：クライアント/サーバ アーキテクチャ特有のもので、N-Tier Web アプリケーションには実装できない PowerBuilder 機能。

このカテゴリの未サポート機能を N-Tier アーキテクチャに移すために作業が必要になります。

カテゴリ 2：現在 Appeon では未サポートですが将来サポート可能な PowerBuilder 機能。

セクション 3.8「未サポート機能の修正と対策」では、未サポート機能の修正におけるハイレベルの手順を説明しています。また、『Appeon ワークアラウンド ガイド』では、いくつかの共通な未サポート機能のサンプルとその対策を説明していま

す。『Appeon ワークアラウンド ガイド』は、
http://www.appeon.com/support/documents/workarounds/6.0/ で利用できます。

4.3 Appeon の Web アプリケーションは外部リソースをサポートするか？

Appeon では Web アプリケーション内で外部リソースの使用をサポートします。INI およびイメージ ファイル、OCX と OLE コントロールなど、最も多くの外部リソースの利用をサポートします。

4.4 なぜアプリケーションをタイプ別に分類するか？

PowerBuilder から Web へのプロセスは、マイグレートされるアプリケーションのタイプによって、簡単か複雑かを判断できます。Appeon は異なるタイプのアプリケーションに対するガイドラインを提供しています。一度アプリケーションのタイプを識別すると、特定のガイドラインに従って、比較的早くアプリケーションを Web にできます。

4.5 アプリケーション タイプの違いとは何か？

PowerBuilder アプリケーションは次に示す 3 つのタイプのどれかに分類できます：

タイプ 1 アプリケーション：次のリクワイアメントに適合し自動的に Web に配布できるアプリケーション。

- Web に実装できない機能を含まないアプリケーション
- 未サポート機能を含まないアプリケーション
- サイズが 50 MB 以下のこれらのタイプのアプリケーション（アプリケーションの PBL にフレームワークを含む）

タイプ 2 アプリケーション：いくつかの Web に実装できない機能、未サポート機能を含むアプリケーション。サイズが大きく、相互依存性のあるユーザオブジェクトをもつアプリケーション。アプリケーションのオブジェクトまたはコードを『Appeon 機能ヘルプ』に従って修正します。

タイプ 3 アプリケーション：Web に実装できない関数を含み、クリティカルな機能と PFC アプリケーションのような非常に複雑なフレームワークを持ち、Appeon のサポート機能で調整されていないアプリケーション。ビジネス リクワイアメントとプロジェクトの範囲に応じて、典型的な J2EE または .NET での書き換えより早くアプ

リケーションを Web にするため、リッチな PowerBuilder GUI を維持しつつ PowerBuilder の RAD 能力を利用します。

4.6 異なるアプリケーション タイプの変換への対応は？

異なるアプリケーション タイプへの対応方法を以下に示します。

タイプ 1 アプリケーション：このタイプはすでにマイグレートする準備が整っており、さらに行う必要な作業はありません。Appeon は PowerBuilder の PBL を読み込み N 層 Web アプリケーションを生成します。この Web アプリケーションは、データウィンドウ、ビジネス ロジック、およびオリジナルの PowerBuilder アプリケーションと同様の UI を持つことができます。

タイプ 2 アプリケーション：Web では実装できない未サポート機能とクライアント/サーバ固有の機能をアプリケーションから除去し、Web に配布したアプリケーションが PowerBuilder アプリケーションと同様の振る舞いを行うようにします。このアプリケーションが大きくて複雑な場合、これらを小さなアプリケーションに分けてデバッグと配布を単純化することを推奨します。マイナーで必要のない機能は可能な限り除去します。コードとオブジェクトの相互依存性を単純化してアプリケーションをより簡単にします。データウィンドウおよび PowerBuilder ソースコードの良い部分は利用できます。再利用できる PowerBuilder ソースコードの量は実際のアプリケーションの複雑性とユーザ リクワイアメントに依存します。

タイプ 3 アプリケーション： このタイプのアプリケーションは一般に Web にマイグレートできませんが、データウィンドウおよび他の便利なオブジェクトはエクスポートして、別の PowerBuilder アプリケーションを構築するために使用できます。

このガイドラインの最終的な目的は、Appeon がより良くサポートできるように PowerBuilder アプリケーションの修正を手助けすることです。既存のデータウィンドウも使用できます。アプリケーションのタイプに応じて、PowerBuilder の関数、一部のコード、またはソースコード全体を修正します。

4.7 複雑なアプリケーションを書き換えるリクワイアメントは？

『Appeon 機能ヘルプ』には、マイグレーションに関わるリクワイアメントとその対策が説明されています。より詳細については、『Appeon 機能ヘルプ』の最初の 2 つのセクションを参照してください。基本的には、アプリケーションの一部を書き換えて（たとえば、複雑なオブジェクトの相互依存性を除去し、1 つのウィンドウ内で使用されているデータウィンドウ数やデータストア数を少なくすることで）、アプリケーションのマイグレーションをより簡単に行えるようにします。

4.8 どんなときにアプリケーションをモジュール化するか？

アプリケーションが、前記に説明したタイプ 2 またはタイプ 3 の場合、実際のマイグレーションのリクワイアメントに応じて、アプリケーションを小さなアプリケーションに分割し、それらを別々に配布します。

4.9 アプリケーションをモジュール化するメリットは？

アプリケーションを小さなアプリケーションに分割する利点を以下に示します：

- 小さなアプリケーションは、Web に配布されたとき、より良いパフォーマンスを示します。
- 小さなアプリケーションは、デバッグや未サポート機能の修正作業をより効果的に行うことができます。
- アプリケーションを小さなアプリケーションに分割すると、複数の開発者が 1 つのプロジェクトで同時に作業でき、複数の開発者がそれぞれ特定のモジュールに専任できるため、開発生産性が高くなります。これらのモジュールは個別に Web に配布して、一体化させた HTML インタフェースにリンクできます。

4.10 アプリケーションのモジュール化の基準は？

アプリケーションを小さなアプリケーションに分割 （モジュール化）する基準を以下に示します。

個々の小さなアプリケーションの機能性は、他のアプリケーションに依存しません。配布された各アプリケーションは他の影響を受けずに実行でき、効果的な作業を行うことができます。この基準に従って、小さなアプリケーション内の関連する機能ポイントを統合し、使用されないオブジェクトの参照を除去するのは良い方法です。たとえば、購買管理アプリケーションを小さな部分に分割し、受注管理のロジック（モジュール）を小さなアプリケーションに移し、サプライヤー情報管理のロジックを他の小さなアプリケーションに移します。

最初の基準に基づき、すべての小さなアプリケーションに渡って複雑なオブジェクトを均等にバランスすることを試みます。

アプリケーションは可能な限り小さくするべきです。

4.11 モジュール化プロセスの例は？

アプリケーションのモジュール化プロセスのハイライトを以下の例に示します。

ここでは前述の購買管理アプリケーションを Web に配布します。このアプリケーションは大きくて複雑なアプリケーションであり、高いレベルで次に示す機能を提供しています。

- 受注入力
- 受注管理
- サプライヤー管理
- リソース プランニング
- レポーティング

ビジネス リクワイアメントに応じて、早急に Web 化が必要な最も重要なアプリケーション機能は、受注入力、受注管理、レポーティングです。

上記のビジネス リクワイアメントに基づき、変換プロセスの最初のフェーズはオリジナル アプリケーションから受注入力、受注管理、レポート機能を抜き出し、それらを PowerBuilder にインポートして新しい小さなアプリケーションを作成します。

『Appeon 機能ヘルプ』と Appeon Code Examples を使用して、小さなアプリケーション内の未サポート ソースコードを Web に順応できるように修正して Web に配布します。

5 分散アプリケーションの構築とマイグレーション

5.1 概要

Appeon は、サーバにデータストアが含まれるか否かに関わらず、分散アプリケーションのマイグレーションをサポートしています。アプリケーションが大きく複雑なロジックまたは多くの未サポート機能を含む場合、ビジネス ロジックの一部をサーバに移すことによって分散アプリケーションの構築を考慮します。

分散アプリケーションのマイグレーション プロセスは、基本的に第 3 章「マイグレーション プロセス」と同じです。マイグレーションを成功させるには、この章の手順を注意深く読み、特別な準備または配布プロセスを実行します。次に示すセクションを参照してください。

- セクション 5.2「未サポート機能を N-Tier NVO として Appeon Server へ移行」では、分散アプリケーションにおける N 層 NVO の使用について、利点、制約、およびガイドラインの詳細を説明しています。
- セクション 5.3「分散データウィンドウを持たない分散アプリケーションのマイグレーション」では、分散データウィンドウを含まない分散アプリケーションのマイグレーションに必要な準備作業を説明しています。
- セクション 5.4「分散データウィンドウを持つ分散アプリケーションのマイグレーション」では、分散アプリケーションに対応する特別なテクニック、GetFullState、SetFullState、GetChanges および SetChanges を説明しています。

  注意：ワークアラウンドテクニックは EAServer 版の AppeonServer でのみ利用できます。

5.2 未サポート機能を N-Tier NVO として Appeon Server へ移行

5.2.1 計画

アプリケーションに未サポート機能が含まれる場合、これらを PowerBuilder のノンビジュアル ユーザオブジェクト (NVO) にカプセル化し、この NVO を Appeon Server に配布します。これらの NVO は、他の Web アプリケーションまたは PowerBuilder アプリケーションと同様に、Appeon Web アプリケーションから呼出すことができます。

このN層NVOは、アプリケーション内に強力なPowerBuilder機能をプログラムすることをサポートします。既存のPowerBuilderアプリケーションをWebに配布する場合、このN層NVOテクニックは多くの未サポート機能（ブラウザの制約とAppeonの制約）への対策として役に立ちます。

5.2.2 N層NVOを利用する利点

N層NVOを使用すると、次の対策を行うことができます：

- 未サポートのデータウィンドウ構文（関数、イベント、プロパティ、ドット表記と式）にAppeonの分散テクニック（データウィンドウのGet/SetFullState、Get/SetChangesをサポート）を適用
- 未サポートの非ビジュアルPowerBuilderシステム オブジェクト
- 未サポートのシステム関数

N層NVOはまた、次に示す処理を行うことができます：

- Appeon Server内でPowerBuilder NVOコードを実行することにより、Webブラウザの制約を除去
- DLLとの連携
- Appeon Server/EAServer内の他のPowerBuilder NVOとの連携
- PowerBuilder 9.0 PBNI/EJBのサポートを通じて、BEA WebLogicやIBM WebSphere、および他のJ2EE-準拠のアプリケーション サーバ内のEJBと連携
- リモートWebサービスまたは.NETコンポーネントとの連携
- Appeon Server/EAServer Webサービス ツールキットを使用して、PowerBuilder NVOからのWebサービスの作成とエクスポース
- PBNIを介してC++クラス/DLLとの連携（その逆も可能）
- メッセージ キューイング(JMS、MQSeries、Tibco)を通じてメッセージ システムとの連携
- PowerBuilder 9.0 XMLデータウィンドウまたはPBDOM機能の使用による、他の企業/部門に送信するためのXML結果集合の作成

- PowerBuilder 9.0 XML データウィンドウまたは PBDOM 機能の使用による、他の企業/部門からの XML 結果集合の利用

- PowerBuilder 9.0 データウィンドウの SaveAs (PDF) 関数の使用による、PDF データウィンドウ ファイルの作成およびテキスト ファイルの処理

- クライアントからサーバへのロジックと処理の移行。多くのロジックや処理をサーバへ移行すると、クライアントでの実行がより速くなり Web ファイルは軽く簡単になります。たとえば、データウィンドウの更新に 10 秒以上要する大きくて複雑なアプリケーションでは、データ更新における入力条件チェックを N 層 NVO に集約することで、データウィンドウの更新が 1 秒に改善されます。

5.2.3 N 層 NVO の使用における制約

N 層 NVO には、一般に、アプリケーションのビジュアルな概観に関わらないビジネス ロジックをカプセル化します。PowerBuilder の未サポート機能のすべてを Appeon Server で実行される NVO コンポーネントにカプセル化することはできません。

N 層 NVO の使用についての詳細情報は、『Appeon 機能ヘルプ- Pure-JavaScript 版』または『Appeon 機能ヘルプ － Appeon Xcelerator 版』に「アプリケーション テクニック｜分散アプリケーション サポート｜N-Tier PowerBuilder NVO」をご参照ください。

5.2.4 未サポート機能やビジネス ロジックの Appeon Server への移行ステップ

ステップ 1 － 新しい PowerBuilder NVO オブジェクトを作成し、必要に応じてユーザ関数、イベント、およびプロパティを追加します。未サポートのコードまたはビジネス ロジックをこれらの関数/イベントにカプセル化します。

ステップ 2 － この NVO を、ターゲット PowerBuilder アプリケーションが Web として 配布される Appeon Server に配布します。PowerBuilder NVO を EAServer コンポーネントとして配布するには、PowerBuilder 開発環境で EAServer コンポーネント ウィザードを使用します。

ステップ 3 － 前のステップで配布した NVO コンポーネントのスタブとスケルトンを生成します。

ステップ 4 －［Properties｜Instances］タブ内の各サーバ コンポーネントの Bind Thread プロパティを有効にします。

ステップ 5 ー クライアントのターゲット アプリケーションで、EAServer プロキシ オブジェクトを作成します。プロキシ オブジェクトはサーバに配布された NVO コンポーネントのエージェントとして動作します。EAServer プロキシ オブジェクトを作成するには、PowerBuilder プロジェクト ペインタで EAServer プロキシ オブジェクト ジェネレータを使用します。

ステップ 6 ー クライアントのターゲット アプリケーションで、Connection オブジェクトのプロパティを設定して、NVO コンポーネントが配布されている Appeon Server に接続します。

ステップ 7 ー クライアントのターゲット アプリケーションで、サーバに配布された NVO コンポーネントをインスタンス化して、オリジナルの未サポート コードが配置されたメソッドを呼出します。

ステップ 8 ー クライアントのターゲット アプリケーションを実行およびデバッグして、サーバの NVO コンポーネントが正しく動作することを確認します。その後、PowerBuilder 開発環境でターゲット アプリケーションのフル構築と最適化を行います。

PowerBuilder の NVO を EAServer コンポーネントとして配布して使用するための詳細の情報については、PowerBuilder ヘルプ と Sybase EAServer のドキュメントを参照してください。

5.3 分散データウィンドウを持たない分散アプリケーションのマイグレーション

5.3.1 EAServer でスタブとスケルトンの生成

スタブとスケルトン ファイルは、クライアントとサーバのコンポーネント タイプに関わらず、クライアントと EAServer 内のコンポーネント間のコミュニケーションを可能にします。たとえば、Java または C++ クライアントは EAServer 内の PowerBuilder NVO コンポーネントと対話を行うことができます。

Appeon Web アプリケーションのスタブとスケルトンの関係は、EAServer プロキシ オブジェクトと N 層 PowerBuilder アプリケーション内の EAServer コンポーネントの関係と同様です。N 層 PowerBuilder アプリケーションと Appeon Web アプリケーションは両方とも EAServer 内の NVO コンポーネントを呼出すことができます。プロキシ オブジェクトは、PowerBuilder クライアントが EAServer 内の NVO コンポーネントにアクセスすることを可能にします。スタブは、Appeon Web アプリケーショ

ンが EAServer とのコミュニケーションを確立して Appeon Server 上の NVO コンポーネントへアクセスすることを可能にします。

詳細方法は Appeon Workarounds Guide の「How to deploy NVO to EAServer 6.1」セッションを参照してください。

5.3.2 アプリケーションの配布

N 層 NVO に必要なスタブとスケルトンを生成した後、分散データウィンドウを含まない分散アプリケーションのマイグレーションは通常のクライアント/サーバ アプリケーションと同様になります。分散アプリケーションのクライアント アプリケーション部分のみのアプリケーション プロファイルを設定し、Appeon Developer 配布ウィザードを使用してアプリケーションを配布します。

5.4 分散データウィンドウを持つ分散アプリケーションのマイグレーション

5.4.1 分散データウィンドウを使用する利点

分散データウィンドウとは、分散環境でデータウィンドウ/データストア オブジェクトを使用することをいいます。分散 PowerBuilder アプリケーションでは、クライアント側のデータウィンドウ コントロールが EAServer 内のデータストア オブジェクトに関連付けられます。クライアント側のデータウィンドウ コントロールは、ユーザ操作に伴うデータのビジュアルなプレゼンテーションと処理に責任を持ち、EAServer 内のデータストア オブジェクトはトランザクション処理に責任を持ちます。クライアント側のデータウィンドウ コントロールのステータス情報は、適切なデータウィンドウ関数を使用することにより、EAServer 内のデータストア オブジェクトのステータス情報に同期されます。

Appeon による分散データウィンドウ テクノロジーの使用には、次の 2 つの利点があります。

- ユーザ インタフェースとビジネス ロジックを分離することで、よりスケーラビリティを提供します。
- Appeon が未サポートのデータウィンドウ機能をサーバのデータストア オブジェクトに移行することで、未サポート機能の対策を行うことができます。

5.4.2 分散データウィンドウの使用に必要な対策

分散データウィンドウを使用しているアプリケーションを Web へマイグレーションする場合、Appeon が提供するワークアラウンド(以下に示す分散データウィンドウへの対応策)を使用する必要があります。

1. Appeon が提供する appeondatawindow と appeondatastore を継承して、分散データウィンドウとデータストア オブジェクトを作成します。

2. PowerBuilder の GetFullState、SetFullState、GetChanges と SetChanges 関数を、Appeon が提供する関数に置き換えます。

5.4.2.a なぜワークアラウンドが必要か

PowerBuilder の GetFullState、SetFullState、GetChanges および SetChanges 関数は、データウィンドウまたはデータストア オブジェクト仕様を受け渡すために、BLOB (Binary Large Object) 型の引数を使用します。Appeon は BLOB 型をサポートしますが、BLOB 型のデータウィンドウまたはデータストア オブジェクト仕様を直接解釈することはできません。ワークアラウンドでは、BLOB 型によってデータウィンドウまたはデータストア オブジェクト仕様を解釈する処理を行います。

5.4.2.b 主要なワークアラウンドのステップ

ステップ 1 － Appeon により提供されるワークアラウンド *appeondatawindow* オブジェクトと *appeondatastore* オブジェクトをアプリケーションに適用します。

詳細の手順とサンプルについては、Appeon ヘルプの「Appeon ワークアラウンド ガイド｜一般的なワークアラウンド テクニック｜Appeon 拡張機能の使い方｜Appeon の GetFullState/SetFullState/GetChanges/SetChanges」を参照してください。

ステップ 2 － PowerBuilder で、*appeondatastore* から継承されたデータストアドを Appeon Server に配布します。

appeondatawindow および *appeondatastore* オブジェクトを Appeon Server に配布しなければなりません。

ステップ 3 － セクション 5.3.1「EAServer でスタブとスケルトンの生成」に記述される手順に従って、サーバのデータストアドのスタブとスケルトンを生成します。

5.4.2.c ワークアラウンドの制約

分散データウィンドウの対応策として、appeondatawindow と appeondatastore オブジェクトを使用するとき、Appeon の GetFullState、SetFullState、GetChanges、および SetChanges 関数の使用についていくつかの制約があります。

テーブル 5-1: Appeon ワークアラウンドの制約

制約/相違事項	制約/相違の説明
データウィンドウのスタイル	OLE およびツリービュー以外のスタイルのデータウィンドウまたはデータストアに動作する
関数の戻り値	Appeon の SetFullState 関数の戻り値は PowerBuilder の SetFullState 関数と異なる
	Appeon の GetChanges 関数はエラーのときいつも-1 を戻すが、PowerBuilder の GetChanges 関数のエラーでは -1、-2 および -3 を戻す
	Appeon の SetChanges 関数は -1 と -3 を戻すが 2 と -2 は戻さない
データウィンドウの ImportString 関数	分散データウィンドウ環境で ImportString 関数を使用する場合、クライアントと Appeon Server マシンで同じ date 表示書式を維持するようにする。加えて、AEM の date/time 書式は Appeon Server のシステム date/time と同じ書式に設定する。
データウィンドウ/データストアのステータスの初期化	PowerBuilder では、データウィンドウ/データストアのステータス情報は DataObject プロパティを設定したときに初期化される。しかし、appeondatawindow と appeondatastore を使用する場合、ステータス情報は DataObject プロパティを異なるデータウィンドウ オブジェクトに変更したときに初期化される。
文字の切捨て	Appeon の SetChanges 関数をターゲット データウィンドウ/データストアに適用した場合、ソース データウィンドウ/データストアの Char 型のカラムにターゲット データウィンドウ/データストアのカラムより多くの文字数が定義されたとき、ターゲット カラムの長さの制約を超えたソース カラムの文字は切り捨てられますが、PowerBuilder では維持される。
Un-modified または modified データ	PowerBuilder の GetFullState と GetChanges の呼出しでは、データウィンドウ コントロール内で変更されたデータ（受入れられていない）は un-modified データとして扱われるが、Appeon の appeondatawindow と appeondatastore の使用では、変更されたデータ（受入れられていない）は modified データとして扱われる。

6 Web または Appeon 機能によるアプリケーションの拡張

6.1 概要

Web または Appeon 機能でアプリケーションを拡張する一般的なアイデアを表 6-1 に示します。

表 6-1：配布アプリケーションを拡張する Web/Appeon 機能

機能のタイプ	機能の内容	配布アプリケーションに追加する方法
典型的な Web 機能	• URL または Sybase Enterprise Portal からアプリケーションにアクセス • ファイアウォール互換。Web ファイルは HTTP 80 番ポート上で転送 • SSL およびディジタル認証セキュリティ • 優れたスケーラビリティ 典型的なアプリケーションは 60-80 コンカレント ユーザ/CPU、300-500 ネームド/エンドユーザ/CPU • その他…	これらの機能は特別なコードを記述すること無く、配布アプリケーションに自動的に追加される。 Enterprise Portal にアプリケーションをロードする方法の詳細はセクション 6.4「Sybase Enterprise Portal へのアプリケーションのローディング」を参照のこと。
Appeon Server オープン インタフェース	PowerBuilder コードを使用した Appeon Web アプリケーションを管理するために、Appeon Server オープン インタフェースを使用。	このインタフェースは、配布アプリケーションに効果を与えるため、PowerBuilder アプリケーション内から呼出される。 詳細はセクション 6.2「Appeon Server オープン インタフェース」を参照のこと。
Appeon クライアント関数	Appeon クライアント関数を使用し、クライアントまたはサーバ情報を取得します。またはアプリケーションで Appeon 拡張機能を使用します。	クライアント関数は、配布アプリケーションに効果を与えるため、PowerBuilder アプリケーション内から呼出される。 詳細はセクション 6.3「Appeon クライアント関数」を参照のこ

		と。
Web インテグレーション	• 柔軟性 & オープン Java、.NET、Web サービスの実装を提供 • アプリケーション サーバ上での Java/EJB、PB NVO、C/C++ DLL、COM/ActiveX コンポーネントの利用 • Web サービス、J2EE、.NET の利用 • メッセージ キューイング システム (MQSeries、JMS など) の利用	Appeon Web アプリケーションに他のアプリケーションを実装するには異なる 2 つの方法がある。 詳細はセクション 6.5「シングル サインオン」とセクション 6.6「Appeon Web アプリケーションと JSP/ASP のインテグレーション」を参照のこと。

6.2 Appeon Server オープン インタフェース

6.2.1 概要

Appeon Server オープン インタフェースは、PowerBuilder コードを介して Appeon Server によるサービスを管理する機会を与えます。Appeon 2.8 以前のバージョンでは、AEM システムのみが Appeon の配布した Web アプリケーションを管理していました。Appeon 6.0 では、PowerBuilder コードを使用して Appeon が配布した Web アプリケーションを簡単に管理できます。

Appeon Server は次に示す五つのオープン インタフェースを提供します：

- getSessionCount
- killAllSessions
- rollbackAllTransactions
- getAllClients
- getAllSessions

6.2.2 オープン インタフェースの特徴

Appeon Server オープン インタフェースの構文については、「Appeon 機能ヘルプ | Web 拡張機能と相違」を参照してください。これらのインタフェースを呼出すコードは PowerBuilder アプリケーション内に記述しますが、PowerBuilder アプリケーションには何の影響も与えません。これらは、Appeon が配布した Web アプリケーションでのみ機能します。

6.2.2.a GetSessionCount

GetSessionCount を使用すると、次に示す 3 種類の情報を取得できます：

- 指定された Appeon Server 内の特定のアプリケーションで開かれたアクティブ セッションの合計数
- 指定された Appeon Server 内のアクティブ セッションの合計数
- Appeon Server クラスタ内の特定のアプリケーションで開かれたセッションの合計数

  Appeon Server クラスタを使用する場合は最初に AEM でクラスタを構成する必要があります。

Appeon の GetSessionCount を使用すると、PowerBuilder コードを使用して特定の Appeon Server 内のアクティブ セッションの合計数を簡単に取得でき、その情報を KillSession のような他のオープン インタフェースに適用してセッションを管理できます。たとえば、サーバ内のアクティブ セッション数が 100 以上になったとき、配布アプリケーションから Appeon Server 内のすべてのセッションを削除するには、PowerBuilder アプリケーション内で最初に GetSessionCount を呼出し、その後 KillSession を呼出します。

6.2.2.b KillAllSessions

KillAllSessions は、Appeon Server または Appeon Server クラスタ内のすべてのアクティブ セッションを削除し、関連するすべてのトランザクションをロールバックします。Appeon Server クラスタを使用する場合は最初に AEM でクラスタを構成する必要があります。

6.2.2.c RollbackAllTransactions

RollbackAllTransactions は、Appeon Server または Appeon Server クラスタ内のすべてのトランザクションをロールバックします。Appeon Server クラスタを使用する場合は最初に AEM でクラスタを構成する必要があります。

6.2.2.d getAllClients

getAllClients は、指定された Appeon Server にある指定されたアプリケーションのアクティブ セッションに対応するすべてのクライアント マシンの IP アドレスを返します。

6.2.2.e getAllSessions

getAllSessions は、指定された Appeon Server にある指定されたアプリケーションのアクティブ セッションの詳細情報を XML 形式で返します。

6.2.3 Appeon 配布アプリケーションへの Appeon Server オープン インタフェースの適用

Appeon が配布した Web アプリケーションに Appeon Server オープン インタフェースを適用するには、次に示す 2 つのステップを実行する必要があります。

1. PowerBuilder アプリケーションから Appeon Server オープン インタフェースを呼出します。Appeon Server オープン インタフェースの呼出しの詳細は、Appeon ヘルプの「Appeon 機能ヘルプ｜Web 拡張機能と相違｜アプリケーションの拡張機能と相違」を参照してください。

2. 通常の PowerBuilder アプリケーションと同様の方法で、PowerBuilder アプリケーションを Appeon Server に配布します。

6.3 Appeon クライアント関数

6.3.1 概要

Appeon クライアント関数は、Appeon の workaround PBL にカプセル化された PowerBuilder グローバル関数の集まりです。Appeon クライアント機能を PowerBuilder アプリケーションに追加すると、配布アプリケーションで次に示す目的を達成できます：

- IP アドレス、またはブラウザ タイプのようなクライアント情報を取得します。
- クラスタまたは単一 Appeon Server にあるアクティブなセッションの総数のようなサーバ情報を取得します。
- Appeon DataWindow メニュー、PDF プリント、LDAP ログオンのような Appeon Web 拡張機能を利用できます。

6.3.2 Appeon クライアント関数の特徴

クライアント関数と戻り値を表 6-2 に示します。いくつかの Appeon クライアント関数は、Appeon システムによって提供されるインタフェースであり、空のボディと空の戻り値を持っていることに注目してください。

表 6-2：クライアント関数と戻り値

関数	PowerBuilder で実行された場合の戻り値	Web アプリケーションで実行された場合の戻り値
AppeonGetAppeonUserName()	無し。 この関数は PowerBuilder では機能しません。	Appeon Web ログインダイアログで入力されたユーザ名。
AppeonGetBrowserVersion()	無し。 この関数は PowerBuilder では機能しません。	クライアント使用している Internet Explorer のバージョン情報。
AppeonGetClientID()	PowerBuilder クライアント固有のセッション ID	Internet Explorer クライアント固有のセッション ID
AppeonGetCacheDir()	無し。 この関数は PowerBuilder では機能しません。	現在の Appeon Web アプリケーションで使用されているキャッシュ ディレクトリ名。
AppeonGetClientIP()	PowerBuilder クライアント マシンの IP アドレス	Internet Explorer クライアント マシンの IP アドレス
AppeonGetClientType()	“PB”	“WEB”
AppeonGetHttpInfo(string attribute)	無し。 この関数は PowerBuilder では機能しません。	Internet Explorer クライアント マシンの IP アドレス
AppeonGetIEHandle()	無し。 この関数は PowerBuilder では機能しません。	現在の Appeon Web アプリケーションの Internet explorer ハンドル情報。
AppeonGetIEURL()	無し。 この関数は	現在の Appeon Web アプリケーションの URL 情報。

	PowerBuilder では機能しません。	
AppeonGetOSType()	PowerBuilder クライアント アプリケーションが実行されている OS タイプ	Internet Explorer ブラウザが実行されている OS タイプ
AppeonGetSessionCount	無し。 この関数は PowerBuilder では機能しません。	指定された Appeon Serve 内の指定されたアプリケーションで開かれたアクティブ セッションの合計数
AppeonGetServerType()	無し。 この関数は PowerBuilder では機能しません。	“J2EE”を返します。
AppeonPopMenu(Datawindow adw_dw, Integer nx, Integer ny)	無し。 この関数は PowerBuilder では機能しません。	なし。 関数を実行すると、指定された DataWindow コントロール上で指定された座標に Appeon DataWindow メニューを表示します。 AppeonPopMenu 関数は AppeonPopMenuOn 関数より優先度が高い
AppeonPopMenuOn(Datawindow adw_dw, Boolean ab_show)	無し。 この関数は PowerBuilder では機能しません。	なし。 関数を実行すると、指定された DataWindow を右クリックしたとき、Appeon DataWindow メニューがポップアップされます。
AppeonPrint2PDF(powerobject adw)	無し。 この関数は PowerBuilder では機能しません。	整数。 DataWindow/DataStore が PDF ファイルに正常に出力された場合は 1。それ以外は -1。

AppeonLDAPLogon(string as_username, string as_password)	無し。 この関数は PowerBuilder では機能しません。	LDAP サーバへログオンしている ユーザ名とパスワードを返します。

6.3.3 Appeon クライアント関数の使用方法

Appeon クライアント関数の目的は、Appeon が配布したアプリケーションに機能を実装することにありますが、これを行うには PowerBuilder アプリケーション内で関数を呼出して機能を有効にする必要があります。

Appeon が配布したアプリケーション内で Appeon クライアント関数を使用するには、「Appeon ワークアラウンド ガイド｜Appeon ワークアラウンド PBL リファレンス｜Appeon クライアント関数」を参照してください。

6.4 Sybase Enterprise Portal へのアプリケーションのローディング

6.4.1 概要

Sybase Enterprise Portal (EP) は、エンドユーザには直感的でセキュアなカスタマイズ可能な環境を、開発者には rapid 開発と配布ツールを、管理者には使いやすいコンソールを提供するオープンでスケーラブルなポータル フレームワークです。Sybase Enterprise Portal のより多くの特徴については、http://www.sybase.com/products/mobilesolutions/unwiredaccelerator を参照してください。

Appeon が配布したアプリケーションは Sybase Enterprise Portal または Unwired Accelerator で実行できます。

6.4.2 Enterprise Portal サポートの制約事項

Appeon は Enterprise Portal 6.1 Info エディションおよび Enterprise エディションをサポートします。Enterprise Portal 内に Appeon が配布したアプリケーションをロードする場合、次に示す制約があります：

- 1 EP ページ内には 1 つの Appeon Web アプリケーションのみ表示されます。たとえば、Sales App Demo はある EP ページに表示され、Appeon Code Examples は他の EP ページに表示されます。これら両方のアプリケーションは正しく機能します。
- Appeon Web アプリケーションを終了しても Internet Explorer は閉じられません。

- 1 つの Internet Explorer ブラウザには、1 つのレスポンス ウィンドウのみ表示されます。

6.4.3 アプリケーションを Enterprise Portal へロードするために必要な作業

アプリケーションのマイグレーション後、Appeon が配布したアプリケーションを Enterprise Portal 内にロードするには、次に示す 4 つのタスクを実行する必要があります。

タスク #1：アプリケーションを portlet に追加します。portlet 内に HTML エレメントを作成し、Appeon が配布したアプリケーションの URL を HTML エレメントにアサインします。

タスク #2：アプリケーションをページ に追加。新しいページを作成し、そのページにアプリケーションを追加してアプリケーションを認可します。

タスク#3：ページをページ グループに追加。新しいページ グループを作成し、タスク#2 で作成したページをページ グループに追加します。

タスク#4：アプリケーションを表示するアカウントを作成。アカウントを作成しアプリケーションを実行する権限をアカウントにアサインします。

前述の各タスクを指導するステップ バイ ステップの手順については、Sybase Enterprise Portal 関連のドキュメントを参照してください。このドキュメントは http://www.sybase.com/products/archivedproducts/enterpriseportal/technicalsupport で利用可能です。

6.5 シングル サインオン

次の 2 方法で Appeon により配布された Web アプリケーションにシングル サインオンを行うことができます。

- サーバ コンポーネントを使用してユーザ情報を管理する
- コマンドライン引数を適用する

詳細な手順については、『Appeon ワークアラウンド ガイド』にある「配布済み Web アプリケーションにシングル サインオンを行う方法」セクションを参照してください。

6.6 Appeon Web アプリケーションと JSP/ASP のインテグレーション

次の 3 方法で Appeon により配布された Web アプリケーションを JSP/ASP アプリケーションと統合することができます。

- Appeon CommandParm および Hyperlink 機能の適用
- Internet Explorer フレームの使用
- N 層 サーバ を介して JSP/ASP アプリケーションをインテグレーション

詳細な手順については、『Appeon ワークアラウンド ガイド』にある「Appeon Web アプリケーションを JSP/ASP アプリケーションと統合する方法」セクションを参照してください。

7 マイグレーション チュートリアル

7.1 イントロダクション

7.1.1 概要

このチュートリアルは 、Appeon の PowerBuilder チュートリアル アプリケーションを Web アプリケーションに変換する手順について 6 つのレッスンから構成されています。このチュートリアルを通して、Appeon を使用する Web マイグレーションの貴重な経験を身につけることができます。

このチュートリアルに使用される PowerBuilder チュートリアル アプリケーションはデータベースとの対話機能を持つプログラムです。このアプリケーションは Pure-JavaScript 配布に対してはほんの少しの未サポート機能を持ち、Appeon Xcelerator 配布に対しては未サポート機能を持っていないことに着目してください。このアプリケーションは、チュートリアルの中では完全な Web マイグレーション作業が適用されたターゲット PowerBuilder アプリケーションとして使用され、Pure-JavaScript 配布では、未サポート機能の修正ステップを適用して Web マイグレーション作業が行われます。

このチュートリアルは 、『Appeon マイグレーション ガイド』 の 「第 3 章： マイグレーション プロセス」 に記述される Web マイグレーション方法の簡単で実用的な例を示します。既存 PowerBuilder アプリケーションを変換したい開発者は、このチュートリアルを入門講座として使用してください。このチュートリアルの学習によって、ユーザはより複雑なネットワーク環境内のより複雑な実際の PowerBuilder アプリケーションをマイグレーションすることができます。

このチュートリアルに含むレッスン：

レッスン 1	Appeon チュートリアル アプリケーションを PowerBuilder にロードする
レッスン 2	Appeon Developer を設定する
レッスン 3	チュートリアル アプリケーションの未サポート機能を解析する
レッスン 4	Appeon チュートリアル アプリケーションを Web へ配布する
レッスン 5	Web アプリケーションを実行する

ユーザが習得する内容：

レッスン 1	PowerBuilder ワークスペースの作成、アプリケーションの PBL ファイルのロード、ODBC データソースの設定、PowerBuilder IDE 内でのアプリケーションの実行。
レッスン 2	アプリケーション プロファイル、Appeon Server プロファイル、Web サーバ プロファイル、配布プロファイル、コネクション キャッシュ プロファイル、およびトランザクション オブジェクト マッピングを Appeon Developer 配置に追加。
レッスン 3	未サポート機能の分析、チュートリアルアプリケーションのフル構築と最適化。
レッスン 4	Appeon 配布ウィザードによる Appeon チュートリアル アプリケーションの配布。
レッスン 5	Internet Explorer による配布済み Web アプリケーションの実行。

所要時間

このチュートリアルの学習時間は約 1 時間半かかります。

7.1.2 チュートリアルの準備

このチュートリアルを始める前に、次の準備ができる必要があります。

- １つのワークステーション

  このチュートリアルでは最も簡単なネットワーク環境を想定し、一台のマシンをすべての n-Tier Web アーキテクチャ、すなわちクライアント PC、Web サーバ、AP サーバ、DB サーバ、開発 PC の役割で使用します。

  システムの環境設定については、『Appeon インストレーション ガイド』 をご参照ください。

- 必要なソフトウェアのインストール

Microsoft Windows 2000、2003 または XP、Sybase PowerBuilder 9、10、10.2 または 11.1、EAServer 6.1、Internet Explorer 6.0 以降のバージョン。

注意：

1) EAServer は Sybase EAServer セットアップ プログラムからインストールすることができます。或いは、Appeon Server をインストールする際に、Appeon Server プログラムで自動的にインストールすることもできます。

2) このチュートリアルでは、EAServer を Web サーバと アプリケーション サーバの役割として使用します。

- Appeon for PowerBuilder 日本語版のインストール

  同一マシンに Appeon Developer と Appeon Server をインストールします。

  Appeon デモ アプリケーションをインストールするには、Appeon Developer をインストールする際、フル インストールするか、または“Demo Applications”オプションを選択してカスタム インストールする必要があります。

7.1.3 関連ファイル

このチュートリアルでは、次の 2 つのファイルを使用します。

ファイル名	appeontutor.pbl	appeontutor.db
位置	¥appeondemo¥Tutorial （例、C:¥Program Files¥Appeon¥Developer6.0¥appeondemo¥Tutorial）	
説明	チュートリアルに用いる PowerBuilder ライブラリ（アプリケーションのソース コード）	チュートリアル アプリケーションに用いる ASA データベース ファイル

7.2 チュートリアル アプリケーションのロード

このチュートリアルは appeontutor.pbl と appeontutor.db の 2 つのファイルを使用して開始します。appeontutor.pbl は Appeon の PowerBuilder チュートリアル アプリケーションのソース コードです。最初に、次の手順に従って、PowerBuilder チュートリアル アプリケーションが正しく実行できることを確認してください。

このセクションでは、次の項目について学習します。

- PowerBuilder ワークスペースの新規作成

- ワークスペースに PBL ファイルのロード
- ODBC データ ソースの設定
- チュートリアル アプリケーションの実行

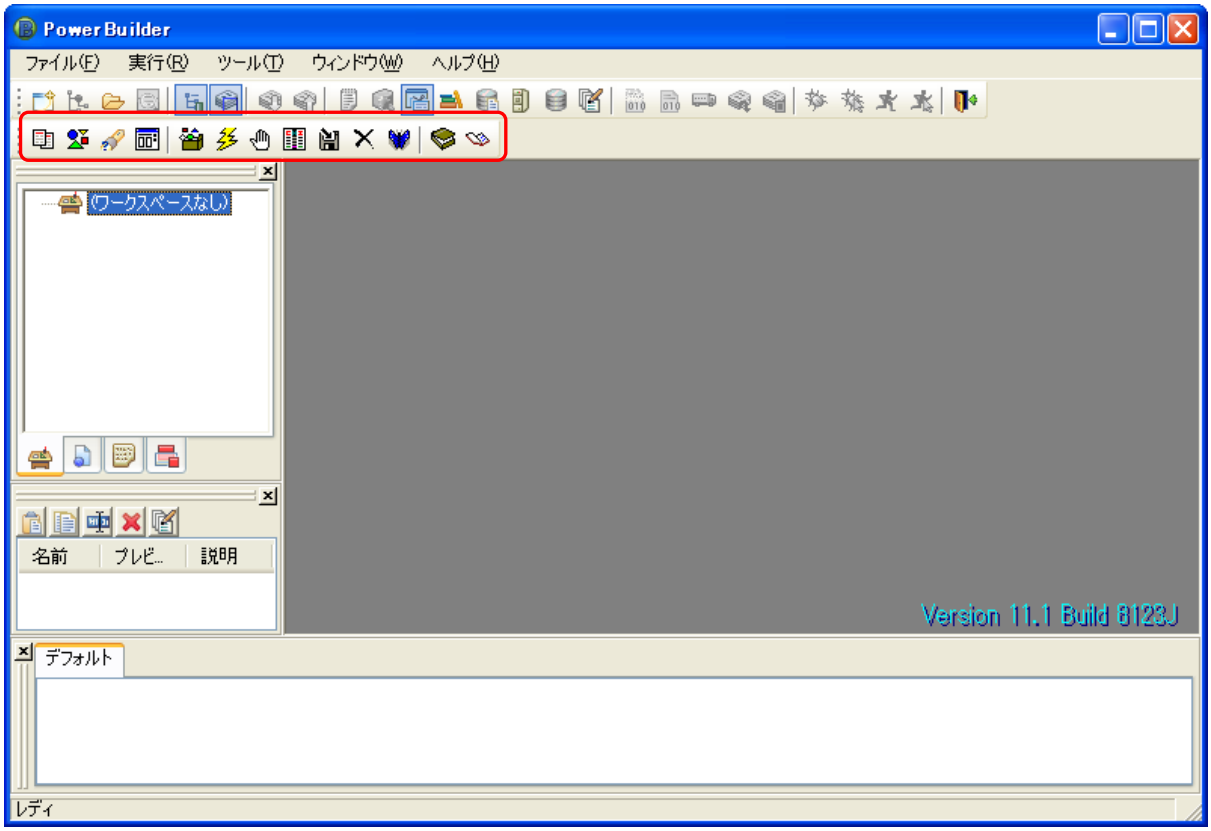
7.2.1 ワークスペースの新規作成

PowerBuilder では同時に 1 つのワークスペースしか開けませんが、ワークスペース内に複数のターゲットを追加し、複数のターゲット内のオブジェクトを開いたり、編集したりすることができます。

PowerBuilder チュートリアル アプリケーション用のワークスペースを作成するには次のステップを実行します：

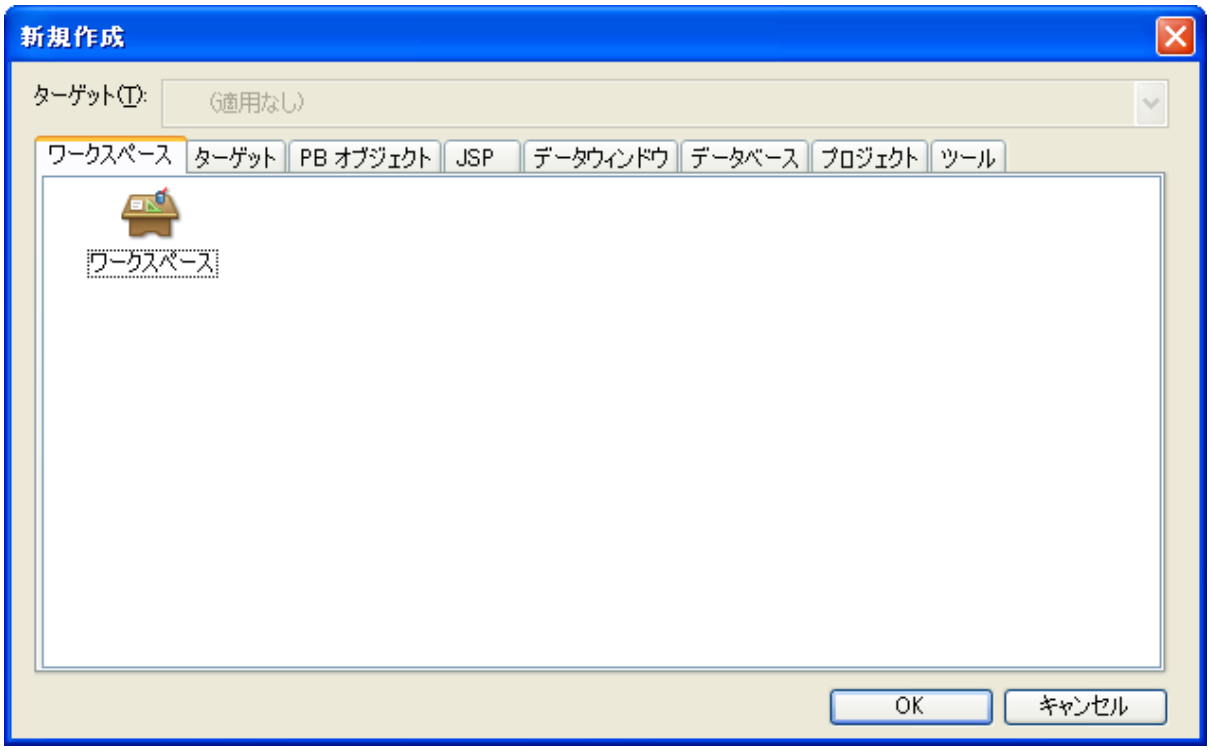
ステップ 1 － PowerBuilder IDE が起動され、Appeon Developer ツールバーがロードされます。

図 7-1：Appeon Developer ツールバー



ステップ 2 － PowerBuilder のメニューから ［ファイル｜新規作成］を選択して、新規作成 ダイアログ ボックスの［ワークスペース］タブを表示します。

図 7-2：ワークスペースの新規作成



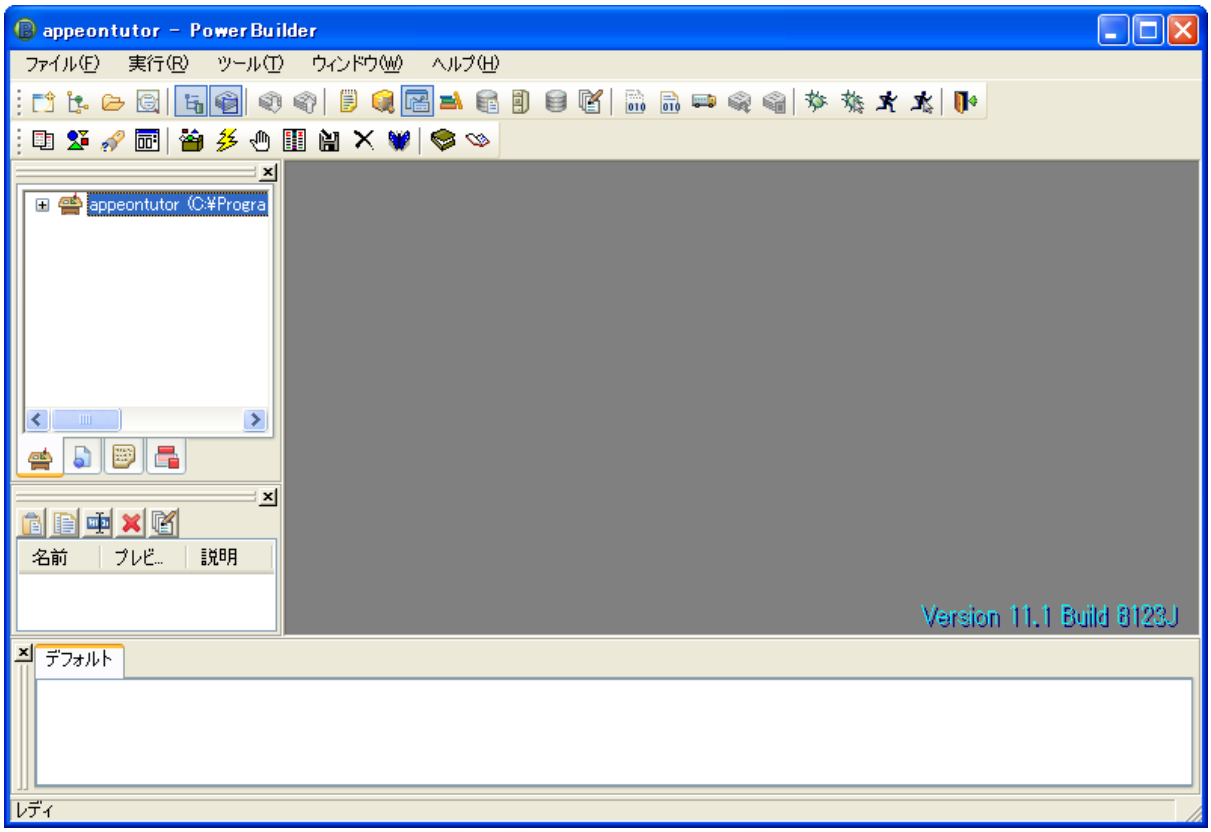
ステップ 3 ― 図 7-2 に示す新規作成 ダイアログ ボックスの［ワークスペース］タブから「ワークスペース」アイコンを選択して［OK］をクリックします。新しいワークスペース ダイアログ ボックスが表示されます。

ステップ 4 ― %Appeon%¥Developer6.0¥appeondemo¥Tutorial フォルダに移動します。［ファイル名］テキストボックスに「appeontutor」と入力し、［保存］をクリックします。

図 7-3：ワークスペースの名前

新しいワークスペース ダイアログ ボックスが閉じられ、作成したワークスペースがシステム ツリー内の最初の項目として表示されます。

図 7-4：新規作成したワークスペース



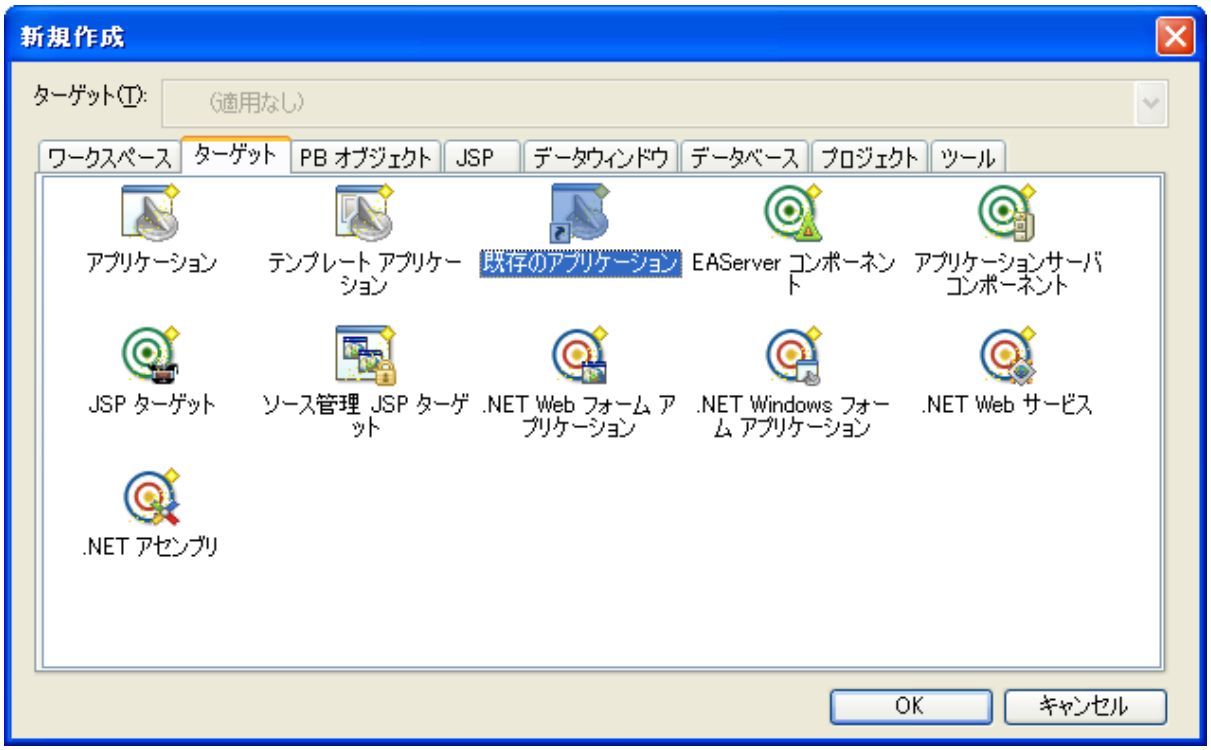
7.2.2 チュートリアル PBL ファイルのロード

PowerBuilder ライブラリ（*.pbl ファイル）はコンパイルしたオブジェクト定義（コンパイル済みのバイナリ形）とソース オブジェクト（スクリプトを含む）の集合であり、これらは同じ場所に格納されています。PowerBuilder ペインタとウィザードは、たとえばアプリケーション、ウィンドウ、データウィンドウ、メニュー、関数、構造体、ユーザ オブジェクトなどの各種のオブジェクトをライブラリに保存します。

チュートリアル アプリケーションの PBL ファイルをワークスペースにロードするには次のステップを実行します：

ステップ 1 ― PowerBuilder のメニューから［ファイル｜新規作成］を選択して、新規作成 ダイアログ ボックスを表示します。［ターゲット］タブをクリックします。

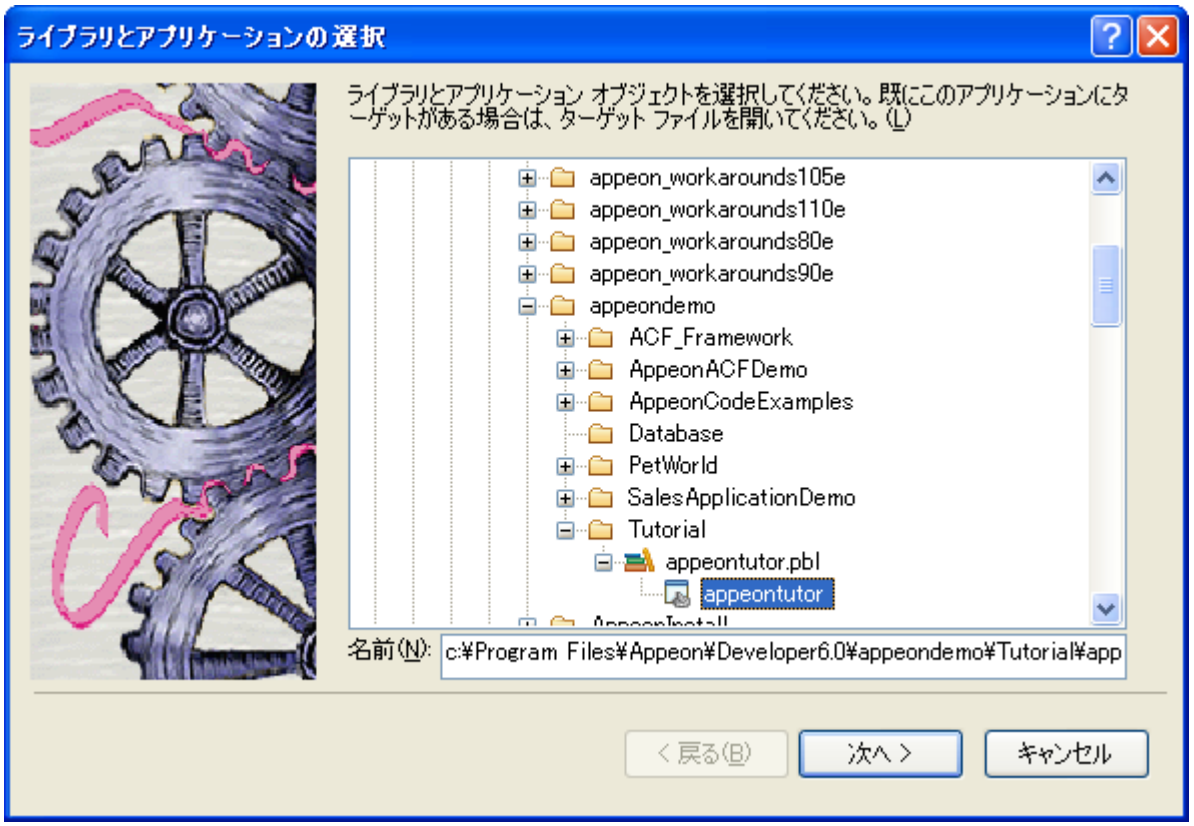
図 7-5：「ターゲット」タブ



ステップ 2 ー「既存のアプリケーション」アイコンを選択して、［OK］をクリックします。［ライブラリとアプリケーションの選択］ダイアログボックスが表示されます。

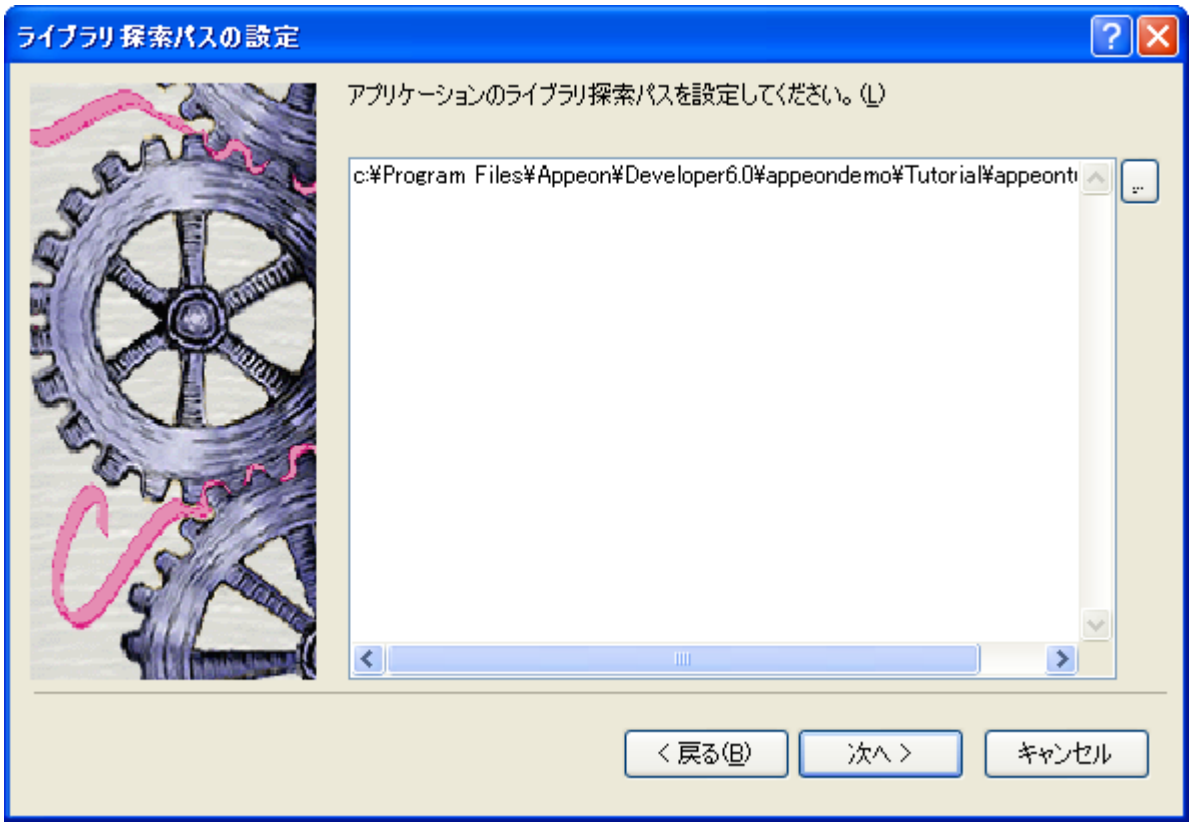
appeontutor.pbl 下の appeontutor アプリケーションを選択して、［次へ］をクリックします。

図 7-6：ライブラリとアプリケーションの選択



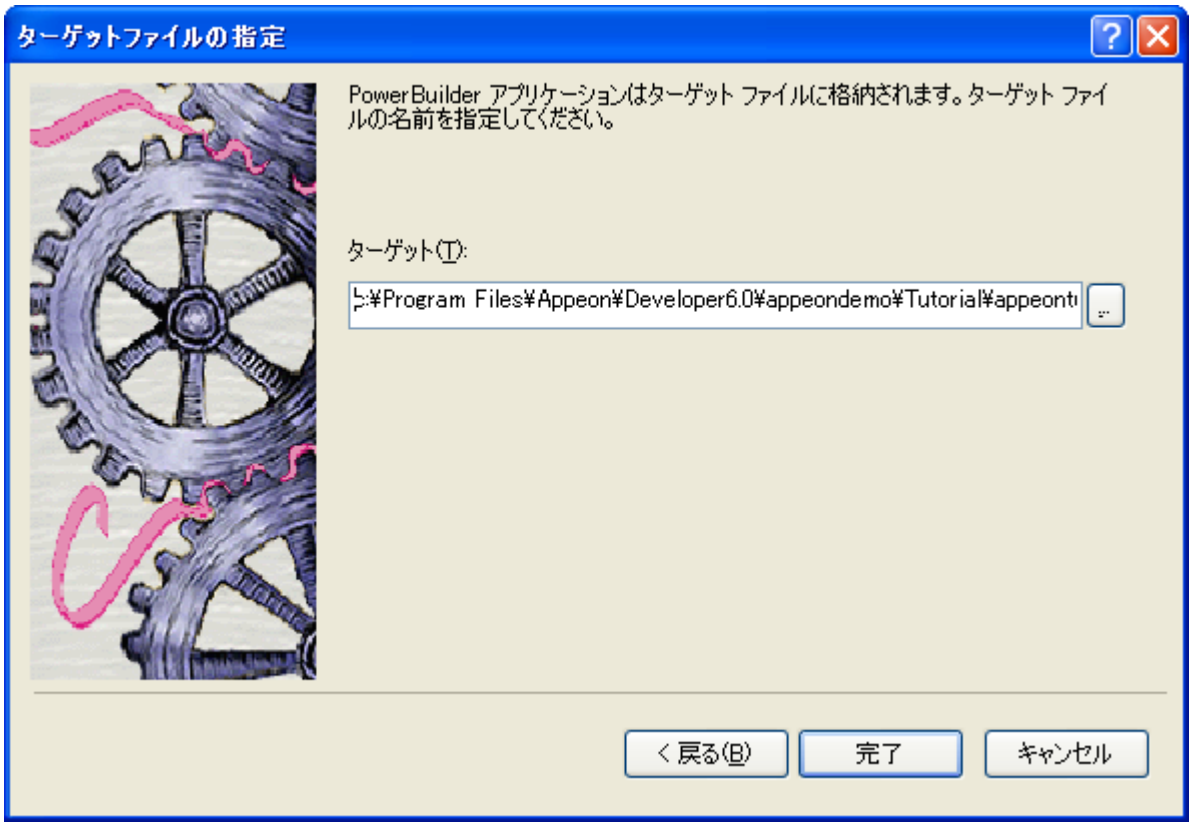
ステップ 3 ー［ライブラリ検索パスの設定］ダイアログボックスが表示されます。［次へ］をクリックします。

図 7-7：ライブラリ探索パスの設定



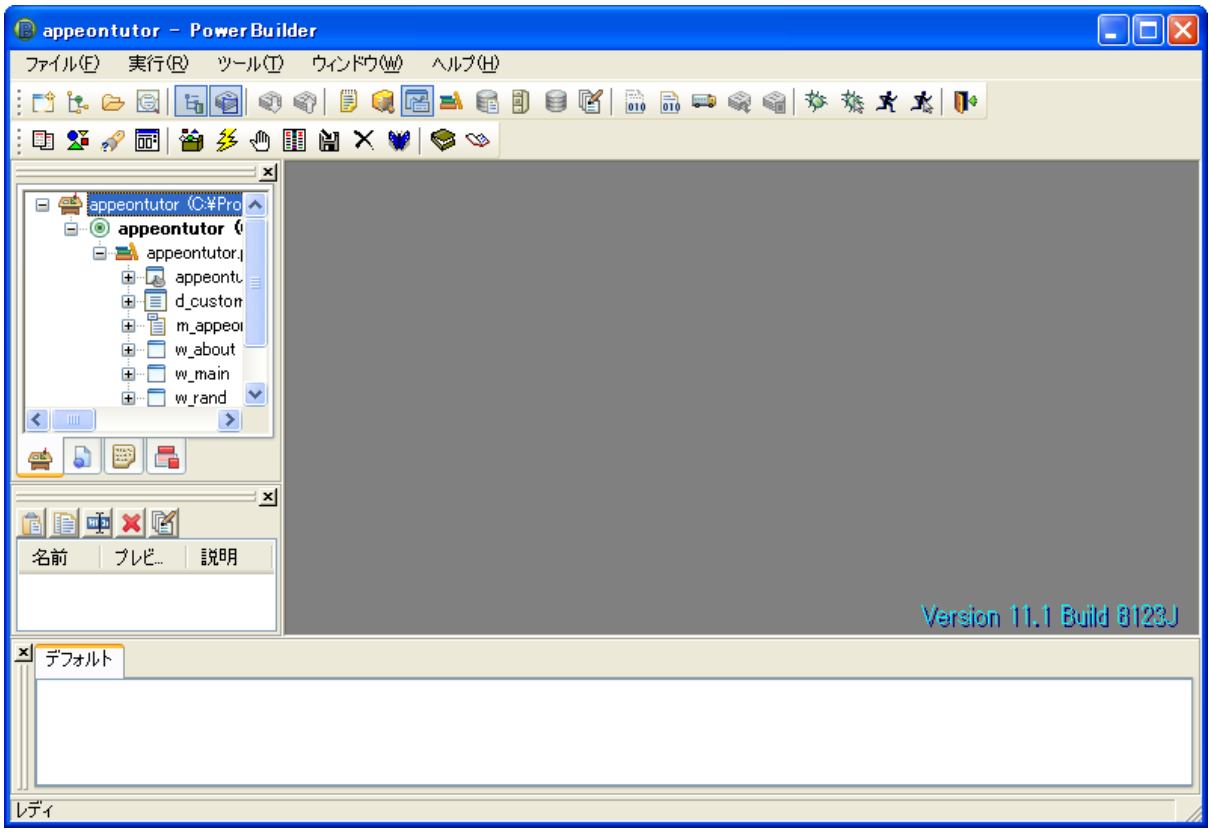
ステップ 4 ー［ターゲットファイルの指定］ダイアログ ボックスが表示されます。［完了］をクリックします。

図 7-8：ターゲット ファイルの指定



ステップ 5 ー Appeon チュートリアル アプリケーションは appeontutor ワークスペースにロードされ、システム ツリーに表示されます。

図 7-9：新規追加された appeontutor ワークスペース



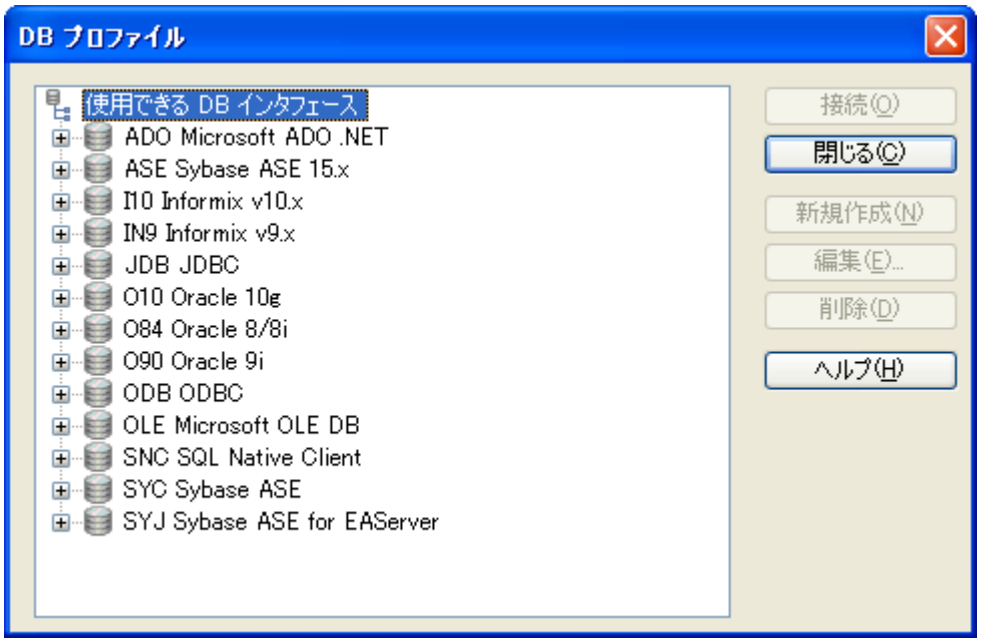
7.2.3 ODBC データ ソースの設定

ODBC データソースは、オープン データベース コネクティビティ（Open Database Connectivity）インタフェースによるデータベース接続に使用するパラメータを格納しています。Appeon チュートリアル PowerBuilder アプリケーションが使用する ASA データベースを識別する ODBC データソースを作成することによって、ODBC データソース名を参照することにより ASA データベースに接続することができます。

チュートリアル アプリケーションのための ODBC データソースを設定するには次のステップを実行します：

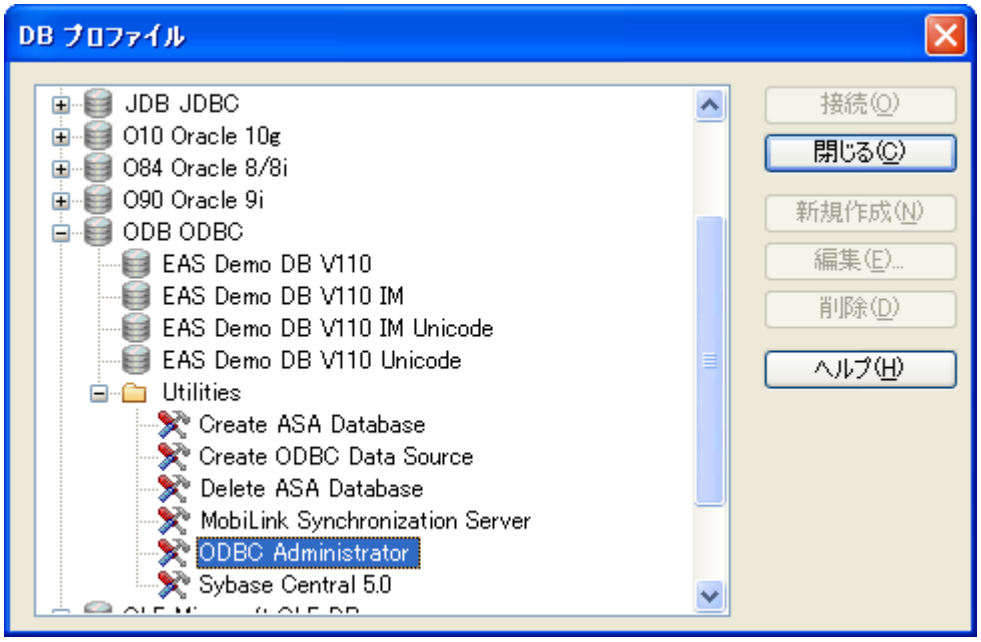
ステップ 1 ー PowerBuilder メニューから［ツール｜DB プロファイル］を選択します。[DB プロファイル] ダイアログ ボックスが表示されます。

図 7-10：データベース プロファイル



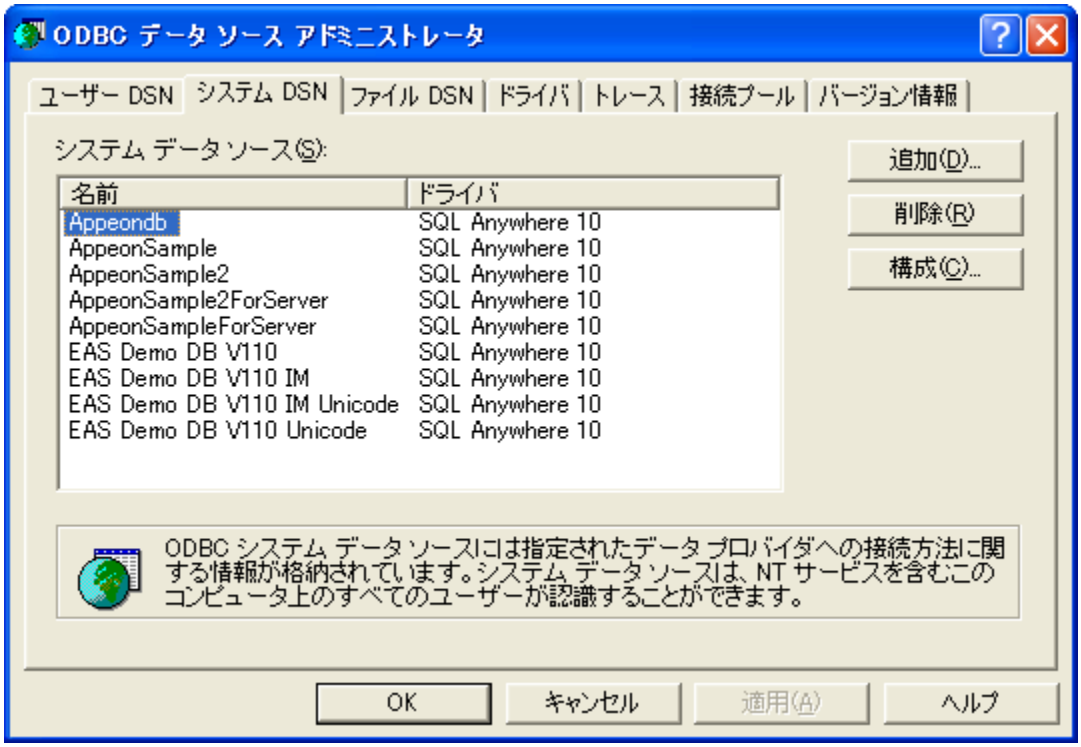
ステップ 2 ー 図 7-11 に示す［ODB ODBC｜Utilities｜ODBC Administrator］をダブルクリックします。

図 7-11：データベース プロファイル



ステップ 3 － ［ODBC データ ソース アドミニストレータ］ダイアログ ボックスが表示されます。［システム DSN］タブをクリックして、［追加］ボタンをクリックします。

図 7-12：[ODBC データ ソース アドミニストレータ] ダイアログ ボックス



ステップ 4 － ［データソースの新規作成］ダイアログ ボックスで EAS 6.0 ASA を選択して、［完了］をクリックします。

ここでは、appeontutor.db データベース ファイルに接続するために EAS 6.0 ASA ドライバを選択します。

図 7-13：データ ソースの新規作成

ステップ 5 － ［SQL Anywhere 9 の ODBC 設定］ダイアログ ボックスが表示されます。必須項目は下表のように入力し、その他の項目はデフォルトにしてください。

表 7-1：データ ソースの設定

タブ名	項目	入力値
ODBC 図 7-14	データソース名	appeontutor
ログイン 図 7-15	ユーザ ID	dba（大文字、小文字を区別する）
	パスワード	sql（大文字、小文字を区別する）
データベース 図 7-16	サーバ名	appeontutor
	データベース名	Appeontutor
	データ ファイル	［参照］をクリックして、appeonsample.db ファイルを選択します。たとえば、C:¥Program Files¥Appeon¥Developer6.0¥Appeondemo¥Database¥appeonsample.db。

図 7-14：[ODBC] タブ

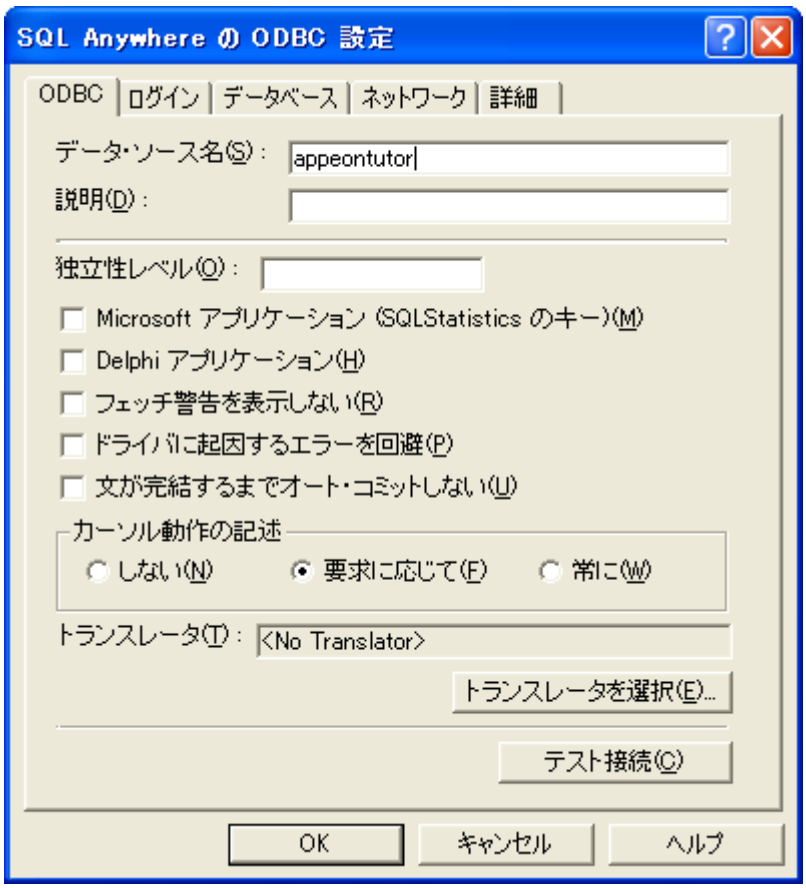

SQL Anywhere の ODBC 設定
ODBC
ログイン
データベース
ネットワーク
詳細
データ・ソース名(S):
appeontutor
説明(D):
独立性レベル(O):
Microsoft アプリケーション (SQLStatistics のキー)(M)
Delphi アプリケーション(H)
フェッチ警告を表示しない(R)
ドライバに起因するエラーを回避(P)
文が完結するまでオート・コミットしない(U)
カーソル動作の記述
しない(N)
要求に応じて(F)
常に(W)
トランスレータ(T):
<No Translator>
トランスレータを選択(E)...
テスト接続(C)
OK
キャンセル
ヘルプ

図 7-15：[ログイン] タブ

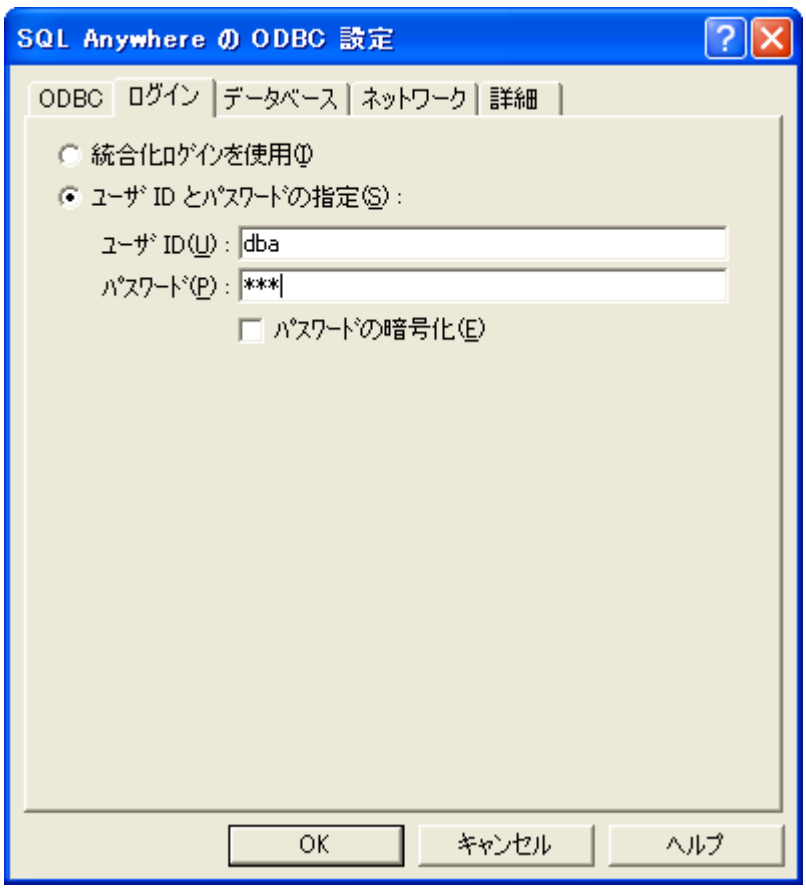
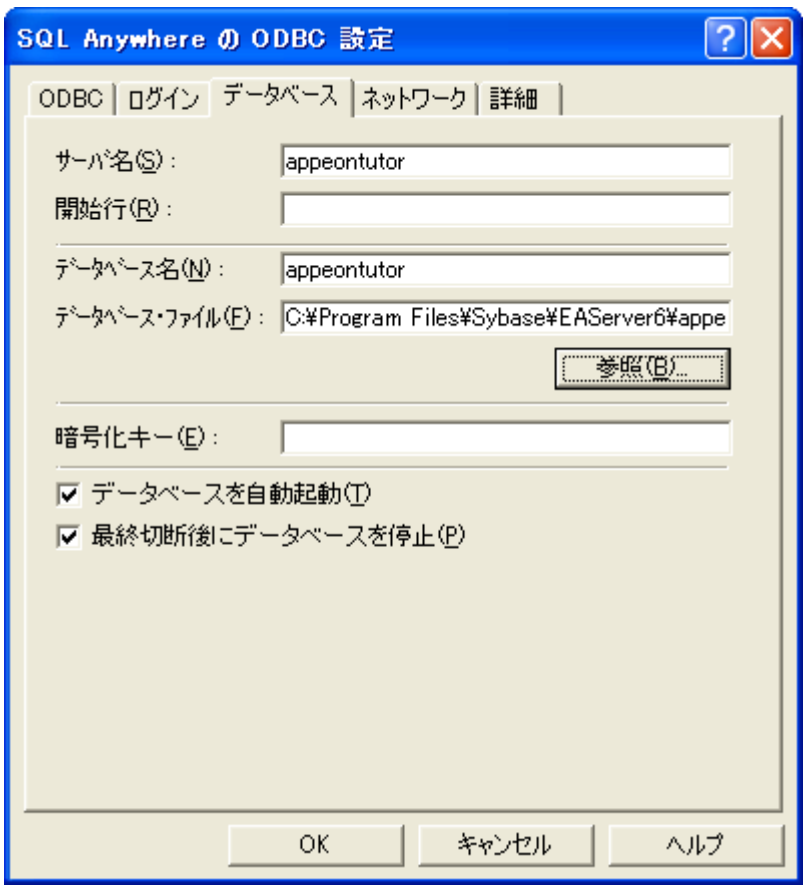

SQL Anywhere の ODBC 設定
ODBC
ログイン
データベース
ネットワーク
詳細
統合化ログインを使用(I)
ユーザ ID とパスワードの指定(S):
ユーザ ID(U):
dba
パスワード(P):
***
パスワードの暗号化(E)
OK
キャンセル
ヘルプ

図 7-16：[データベース] タブ

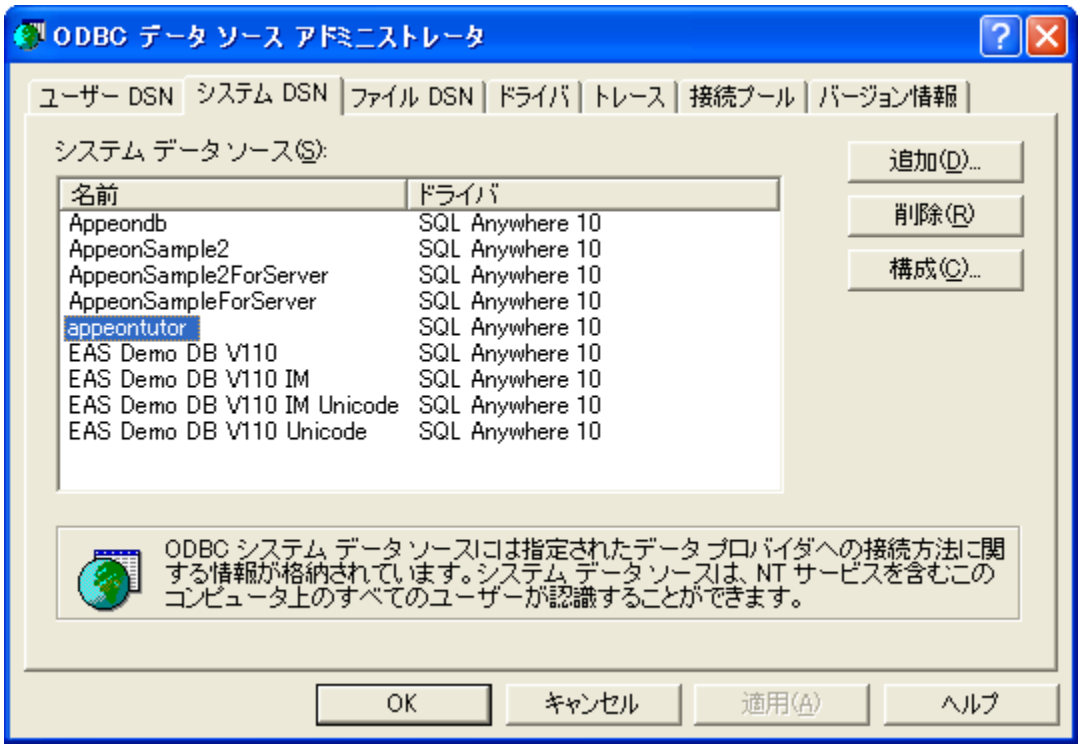


ステップ 6 ー「ODBC」タブに戻り、［テスト接続］をクリックします。接続が成功することを確認します。

ステップ 7 ー［OK］をクリックして、［SQL Anywhere 10 の ODBC 設定］ダイアログ ボックスを閉じます。

appeontutor がシステム データ ソースとして「システム DSN」タブ に追加されます。

図 7-17：[ODBC データソース アドミニストレータ] ダイアログ ボックス



ステップ 8 ― ［OK］をクリックして、［ODBC データソース アドミニストレータ］ダイアログ ボックスを閉じます。

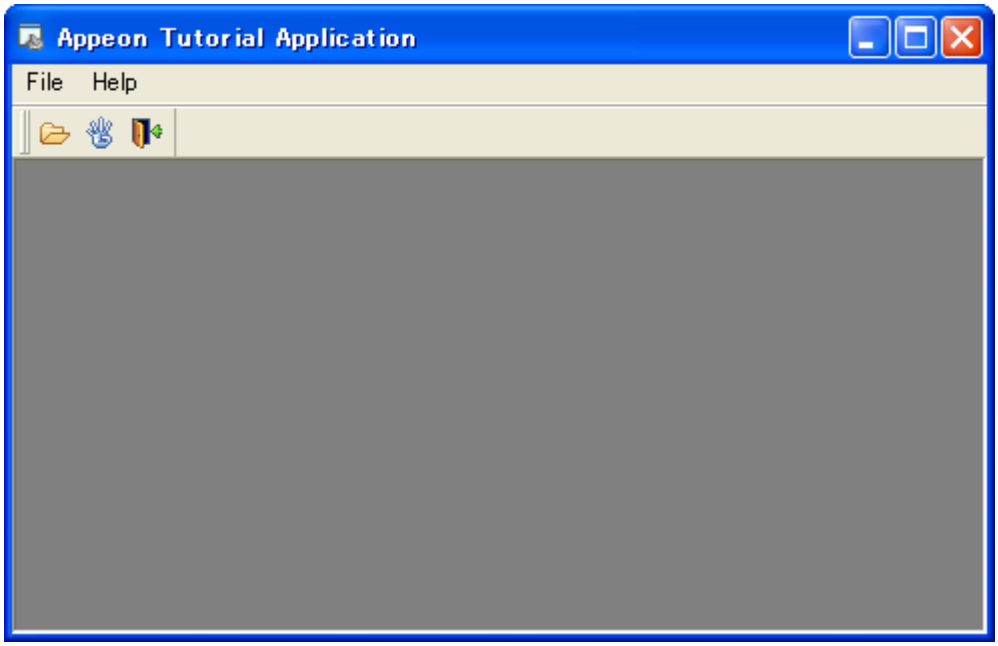
7.2.4 チュートリアル アプリケーションの実行

appeontutor データソースに接続するためのデータベース接続パラメータ（PowerBuilder Transaction プロパティ）は、Appeon チュートリアル PowerBuilder アプリケーション内で既に設定されています。ユーザはそのまま Appeon チュートリアル PowerBuilder アプリケーションを実行することができます。

Appeon チュートリアル PowerBuilder アプリケーションを実行するには次のステップを実行します：

ステップ 1 ― PowerBuilder メニューから ［実行｜実行］を選択するか、またはパワーバーの［実行］ボタンをクリックします。Appeon チュートリアル PowerBuilder アプリケーションが起動されます。

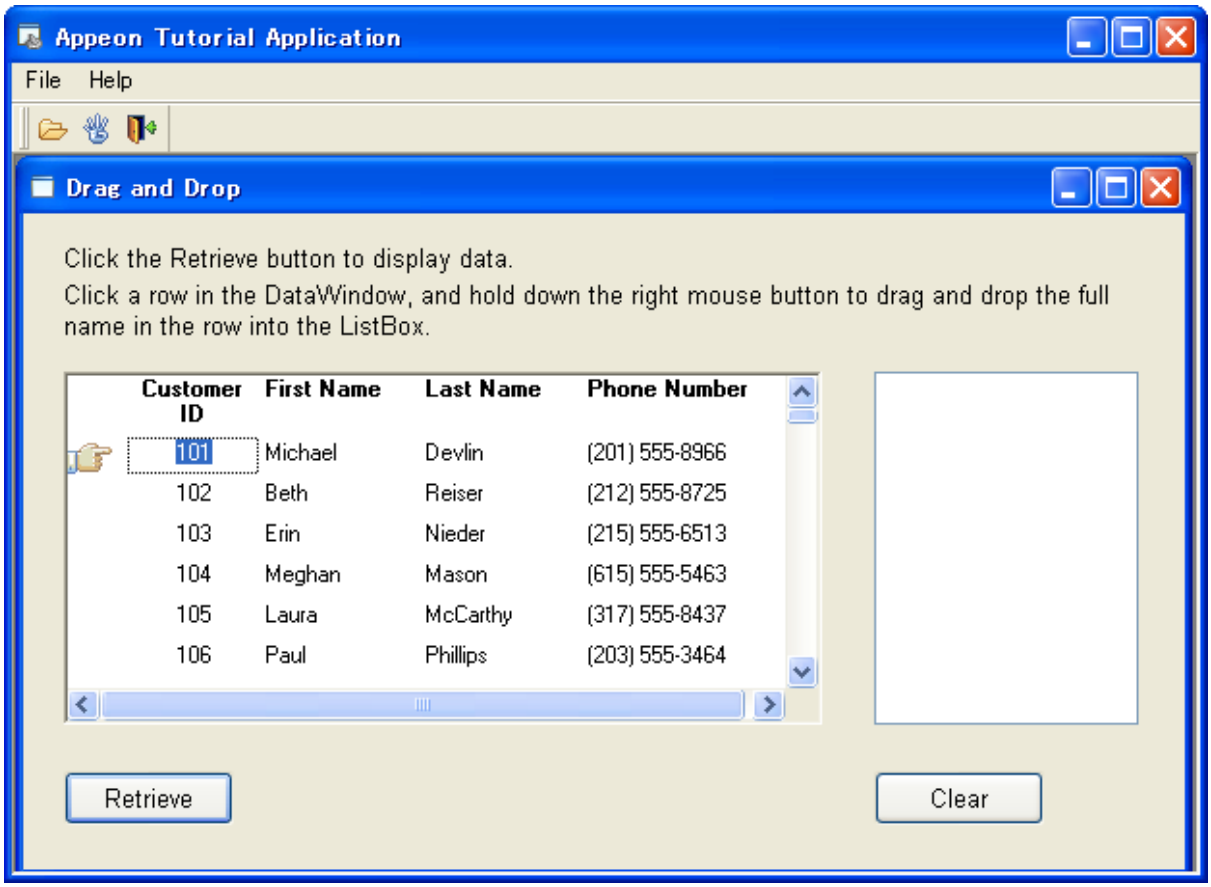
図 7-18：Appeon チュートリアル アプリケーション



ステップ 2 ー Appeon チュートリアル アプリケーションのメニューから［File｜Drag and Drop］を選択して、Drag and Drop ウィンドウを表示します。

Appeon チュートリアル アプリケーションには 1 つの MDI ウィンドウ、2 つのシート ウィンドウ、および 1 つの About ウィンドウがあります。Drag and Drop ウィンドウの［Retrieve］をクリックするとデータが表示されます。データウィンドウの行をクリックしてから、マウスの右ボタンを押しながらドラッグし、右のリストボックス上にドロップすると、フルネームがリストボックスに表示されます。［Clear］をクリックすると、リストボックスがクリアされます。

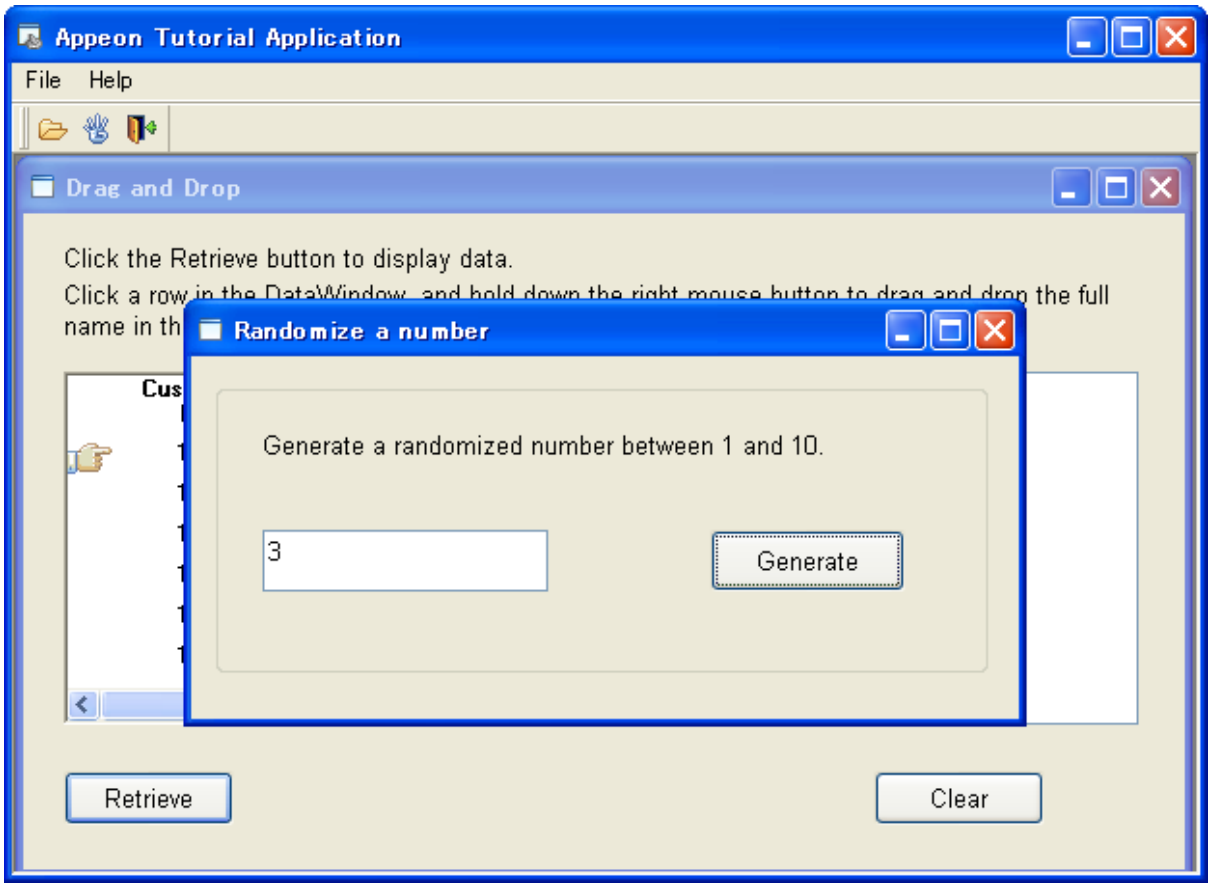
図 7-19：ドラッグ アンド ドロップ



ステップ 3 ー［File｜Randomize］を選択して、Randomize a number ウィンドウを表示します。

このウィンドウは PowerBuilder の Randomize と Rand 関数を使っています。［Generate］をクリックすると、1 から 10 までの範囲にある整数の乱数がウィンドウに表示されます。

図 7-20：整数の乱数



ステップ 4 ー Appeon チュートリアル PowerBuilder アプリケーションが閉じて、PowerBuilder IDE に戻ります。

7.3 Appeon Developer の設定

Appeon Developer は、Appeon for PowerBuilder 日本語版の 1 つのコンポーネントであり、PowerBuilder の機能を拡張し、PowerBuilder の技術だけを使用して、新規または既存の PowerBuilder アプリケーションを Web アプリケーションへ変換することができます。

Appeon Developer は 、PowerBuilder IDE 内で PowerBuilder から Web への変換を可能にする一連のツールを提供しています。これらのツールは PowerBuilder を開く度に自動的にロードされ、PowerBuilder IDE 内のツールバーを介してアクセスできます。

図 7-21：Appeon Developer ツールバー

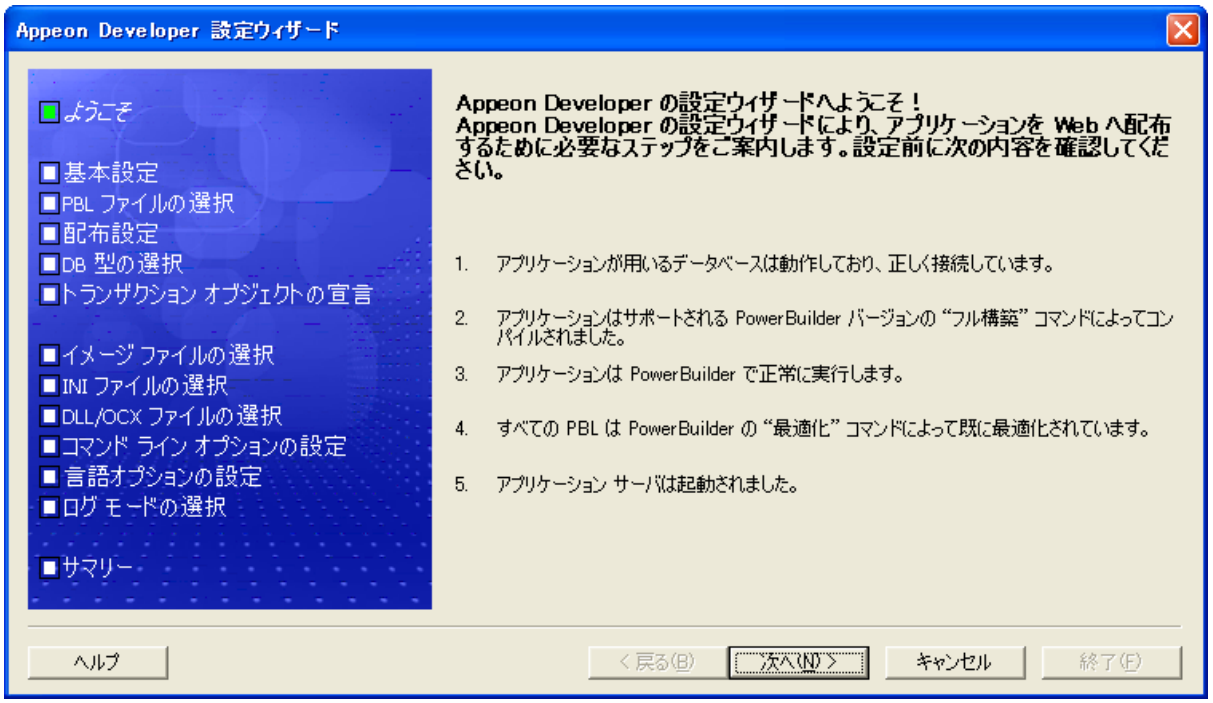
Appeon Developer で Appeon チュートリアル アプリケーションを配布する前に、次の 5 つの設定を正しく行う必要があります。

- 基本設定
- PBL ファイルを選択する
- 配布設定を行う
- DB 型を選択する
- トランザクション オブジェクトを宣言する

すべての設定が Appeon 設定ウィザードにより完成できます。ウィザードを実行するには、Appeon Developer ツールバー上の［設定ウィザード］ボタン（） をクリックしてください。

最初の画面で紹介を読んでから［次へ］をクリックして続けます。

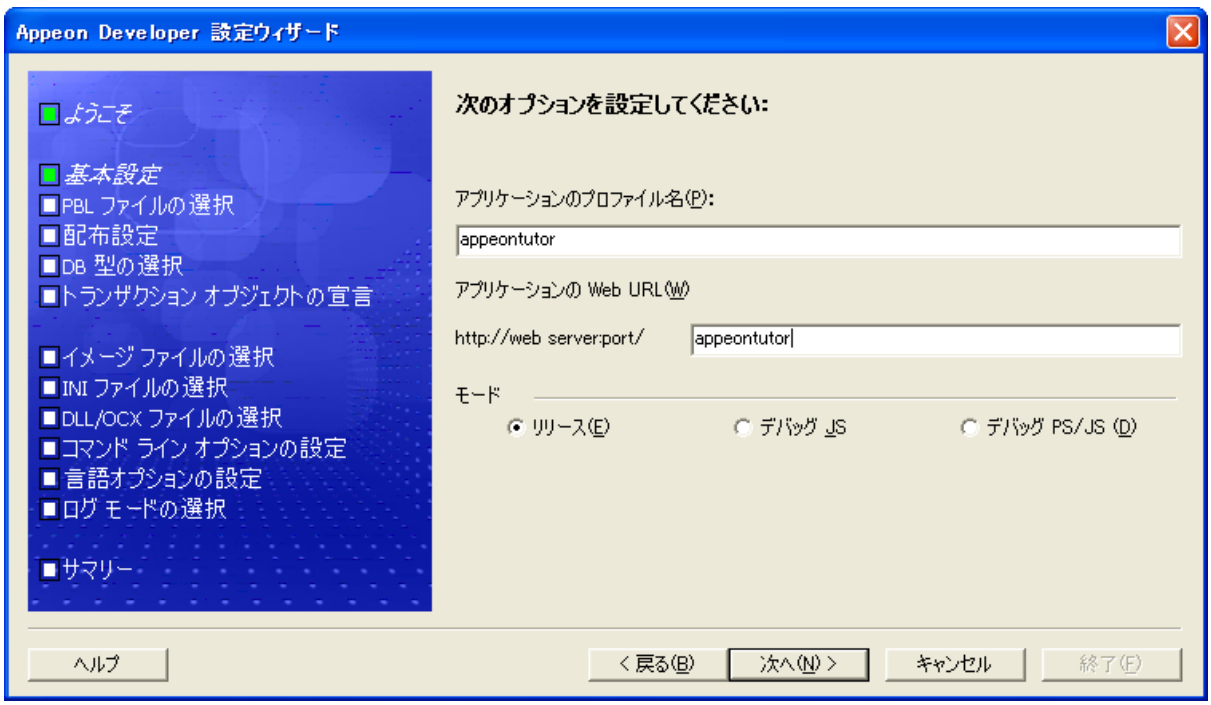
図 7-22：紹介ページ



7.3.1 基本設定

基本設定とは、配布プロセスが行う際にアプリケーションに関する必要な設定です。これは、図 7-23 に示すようにアプリケーション プロファイル名、Web ルートおよびモードを含めています。

図 7-23：基本設定ウィンドウを設定する



ステップ 1 ー 次のように設定します：

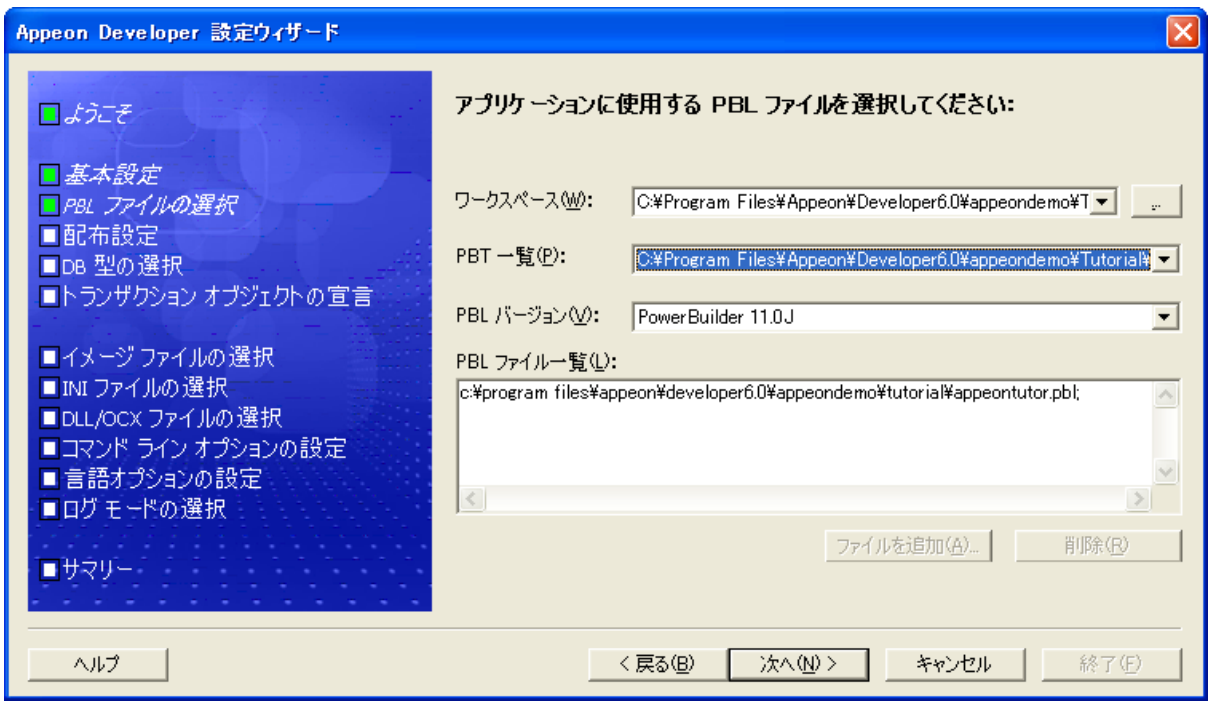
- ［アプリケーション プロファイル名］では、*appeontutor* を入力します。
- ［アプリケーション Web ルート］では、*appeontutor* を入力します。

ステップ 2 ー［次へ］をクリックして続けます。

7.3.2 PBL ファイルを選択する

PowerBuilder アプリケーションのソースコードの場所を指定します。

図 7-24：PBL を選択する



ステップ 1 － PowerBuilder IDE に *appeontutor* が既に開かれていることを確認します。［現在のワークスペース］をクリックして、*appeontutor* を［ワークスペース］ドロップダウン リストに追加します。

appeontutor が選択されると、*appeontutor* の PBT ファイルが［PBT 一覧］に追加されるようになります。

ステップ 2 － *appeontutor* の PBT ファイルが選択されていない場合、これを選択します。

appeontutor の PBT ファイルが選択されると、*appeontutor* の PBL ファイルが［PBL ファイル一覧］に追加されるようになります。

ステップ 3 －［次へ］をクリックして続けます。

7.3.3 配布設定を行う

配布設定とは、アプリケーションが配布される際に必要な Appeon Server および Web サーバに関する設定です。

このセクションに従って、次の内容を学習します：

- Appeon Server および Web サーバを起動する

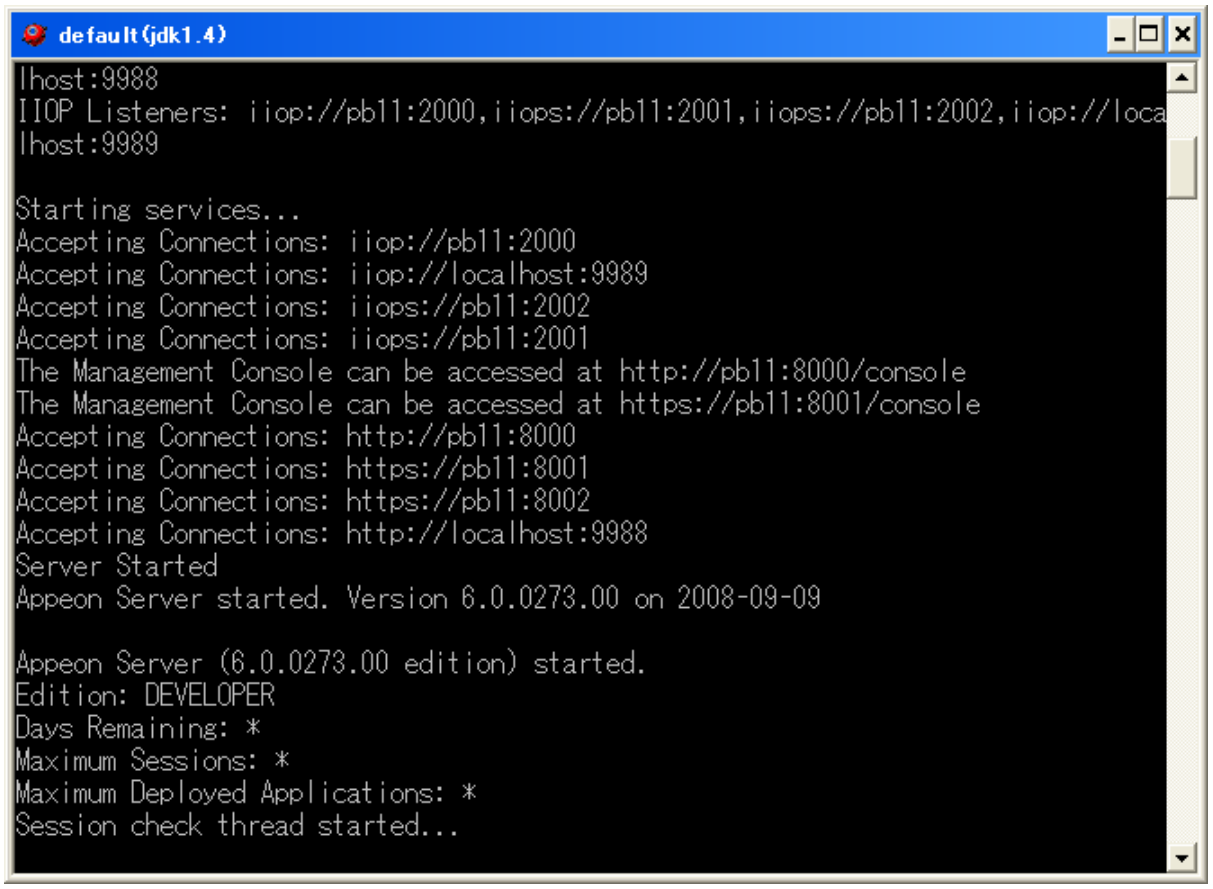
- Appeon Server プロファイルを追加する
- Web サーバ プロファイルを追加する
- 配布プロファイルを追加する

7.3.3.a Appeon Server および Web サーバを起動する

Appeon Server プロファイルと Web サーバ プロファイルを追加する前に、Appeon Server および Web サーバに接続していることを確認する必要があります。このチュートリアルでは、EAServer が Appeon Server と Web サーバとしても使用されているので、EAServer を起動する必要があります。EAServer の起動方法については、次のステップに従ってください。

Windows で［スタート｜プログラム｜Appeon 6.0 for PowerBuilder｜Appeon Server | %InstanceName%］を選択します。図 7-25 に示すように、“Accepting connections”というメッセージが表示するまで待ってください。このメッセージの表示は、Appeon Server および Web サーバが利用できるようになることを示します。

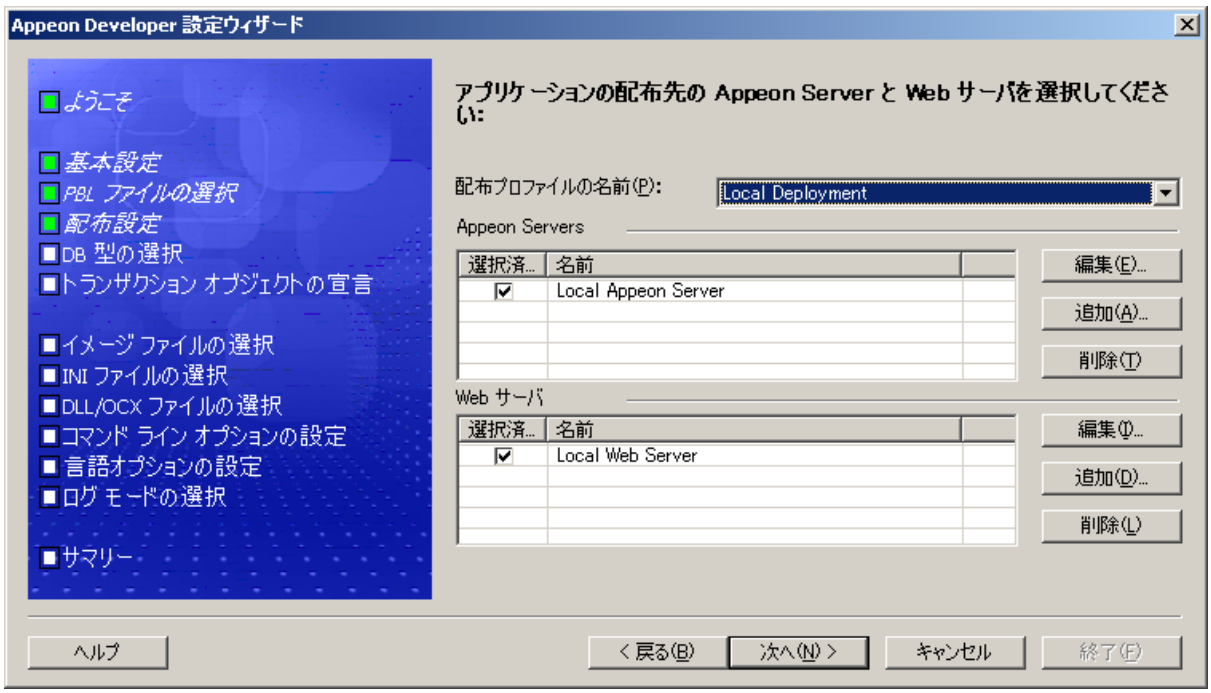
図 7-25：Appeon Server



7.3.3.b Appeon Server プロファイルを追加する

Appeon Server プロファイルには、Appeon 配布に必要な Appeon Server の設定およびアプリケーション サーバの設定を含めています。

図 7-26：Appeon Server プロファイル



チュートリアル アプリケーションの配布に Appeon Server プロファイルを追加するには：

ステップ 1 – Appeon Server グループ ボックス内の［追加］ボタンをクリックします。

図 7-27 に示すように、［追加］ダイアログ ボックスが表示されます。

図 7-27：Appeon Server プロファイルの設定

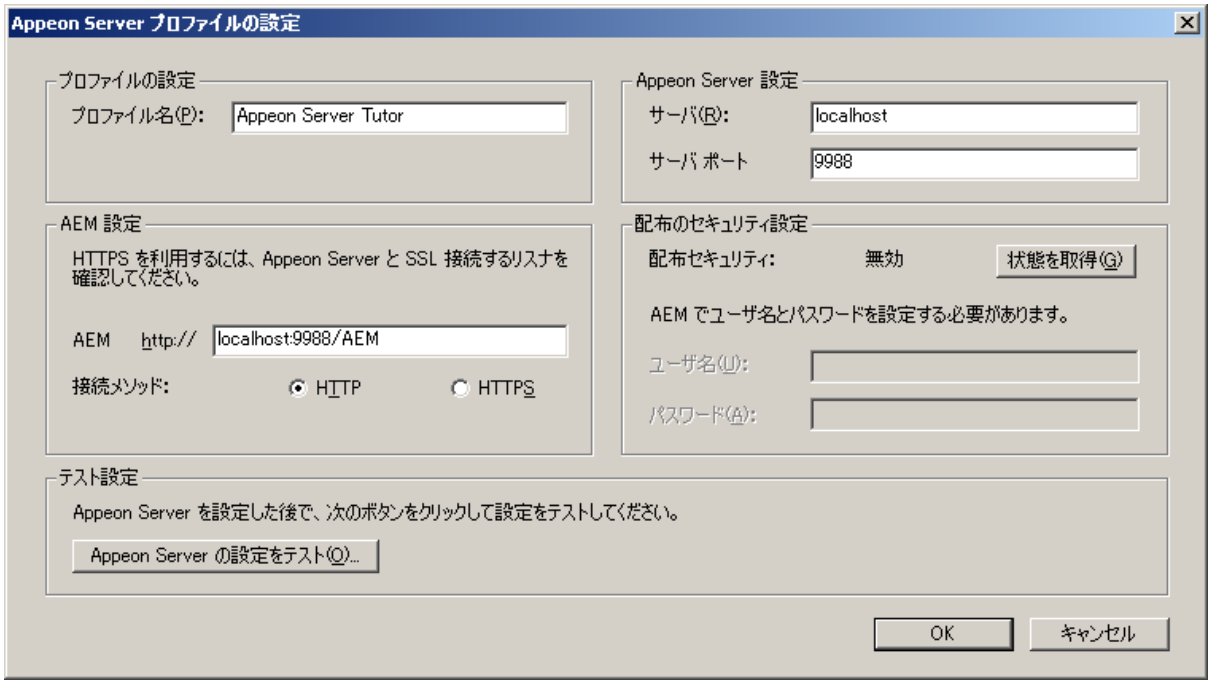


表 7-2 は Appeon Server プロファイルの設定方法を説明しています。

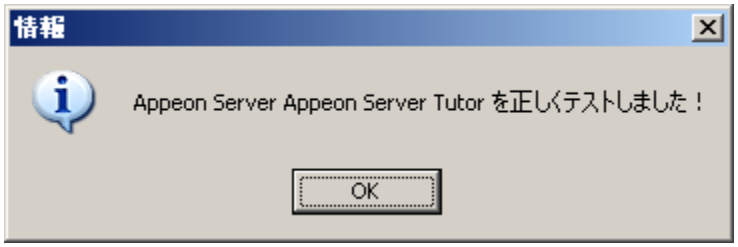
表 7-2：Appeon Server プロファイルの設定説明

プロパティ	設定説明
プロファイル名	“Appeon Server Tutor”を Appeon Server プロファイルの名前として入力します。
サーバ	“localhost”を入力します。
サーバ番号(http)	“9988”を入力します。
AEM URL	［サーバ］と［サーバ番号］を入力し次第、Appeon Enterprise Manager (AEM) の URL が自動的に生成されます。
接続メソッド	そのままデフォルトで使用します。
配布セキュリティ	そのままデフォルトで使用します。
ユーザ名	そのままデフォルトで使用します。
パスワード	そのままデフォルトで使用します。

ステップ 2 ー［テスト］をクリックします。

Appeon Developer は以上の設定によって Appeon Server への接続を試します。図 7-28 に示すように、テストが成功するまで、次のステップに進まないでください。

図 7-28：接続のテスト



ステップ 3 ー［OK］をクリックします。

図 7-29 に示すように、“Appeon Server Tutor”が Appeon Server 一覧に追加されます。

図 7-29：Appeon Server Tutor の追加

7.3.3.c Web サーバ プロファイルを追加する

Web サーバ プロファイルには Appeon 配布に必要な Web サーバの設定を含めています。

Appeon Server Web サーバが Appeon チュートリアル PowerBuilder アプリケーションの配布に使用されるため、Appeon Server が既に実行中の場合に再起動する必要はありません。

チュートリアル アプリケーション配布に Web サーバ プロファイルを追加するには：

ステップ 1 – Web サーバ グループ ボックス内の［追加］ボタンをクリックします。

図 7-30 に示すように、追加ダイアログ ボックスが表示されます。

図 7-30：Web サーバ プロファイルの設定ウィンドウ

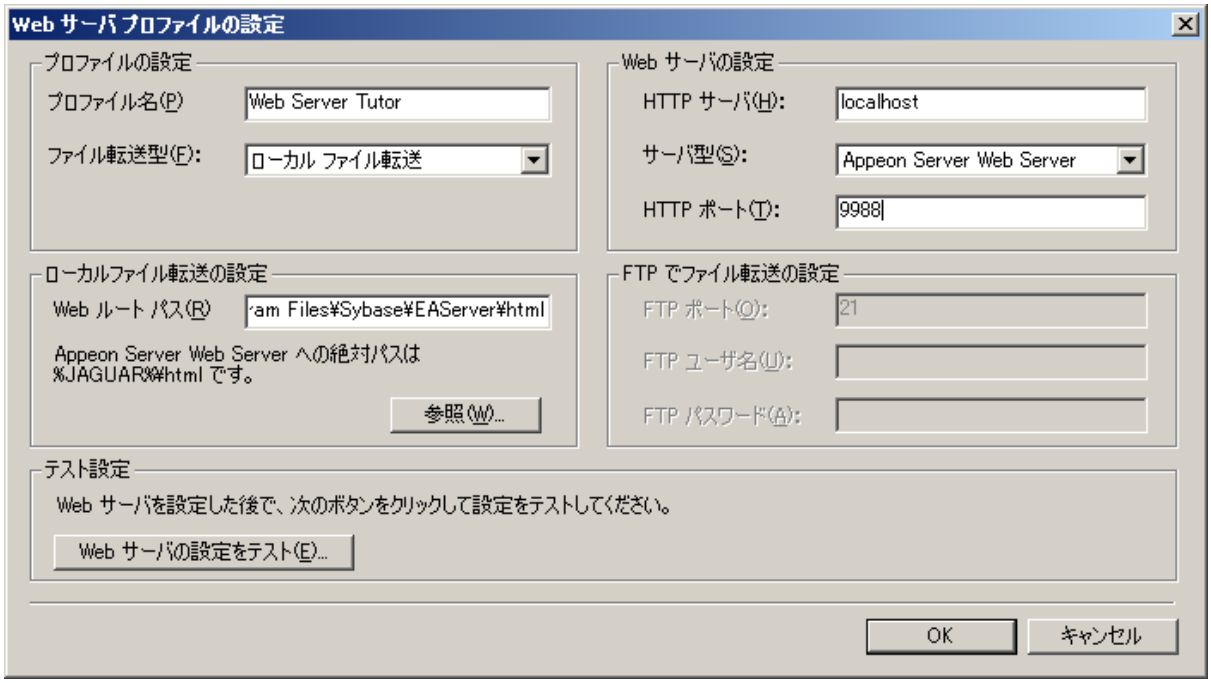


表 7-3 は Web サーバ プロファイルの設定方法を説明しています。

表 7-3：Web サーバ プロファイルの設定説明

プロパティ	説明
プロファイル名	“Web Server Tutor” を Web サーバ プロファイルとして入力します。
ファイルの転送型	“ローカル ファイル転送” を選択します。
HTTP サーバ	“localhost” を入力します。
サーバ型	“Appeon Server Web Server” を選択します。
HTTP ポート	“9988” を入力します。
Web ルート パス	Web ルート パスが%JAGUAR%¥html であることを確認します。ここで、*%JAGUAR%* が EAServer のインストレーション ディレクトリを示します。（たとえば、C:¥Program Files¥Sybase¥EAServer¥html）。

ステップ 2 – ［テスト］をクリックします。

Appeon Developer は IP（localhost）とポート（9988）によって Web サーバへの接続を試します。テストが成功するまで、次のステップに進まないでください。

ステップ 3 － ［OK］をクリックします。

“Web Server Tutor”が Web サーバ一覧に追加されます。（図 7-31）

図 7-31：Web Server Tutor が追加される

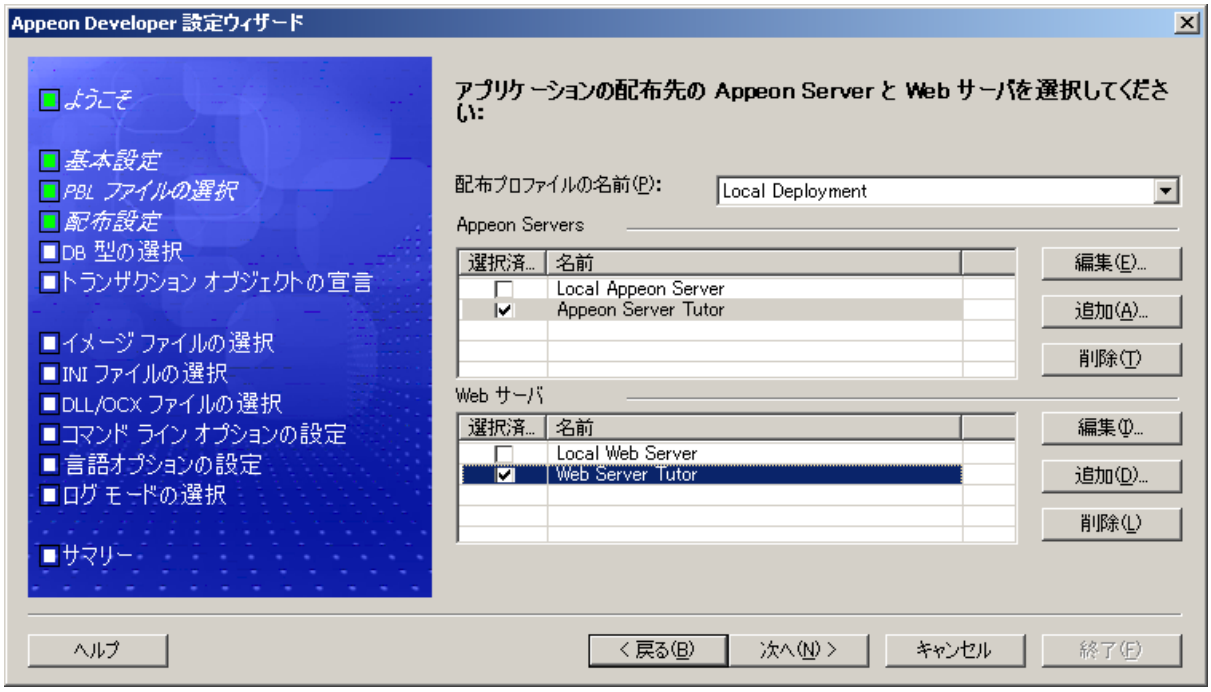
7.3.3.d 配布プロファイルを追加する

配布プロファイルは、指定された Web サーバと Appeon Server を 1 つのグループとして関連付けて Web 配布に使用します。このチュートリアルは、配布プロファイルが Appeon Server Tutor と Web Server Tutor を関連付けることになります。

ステップ 1 － ［配布プロファイル名］に“*Local Deployment for Tutorial*”を入力します。

ステップ 2 － Appeon Server 一覧から“*Appeon Server Tutor*”を選択して、Web サーバ一覧から“*Web Server Tutor*”を選択します。（図 7-32）

図 7-32：配布プロファイル

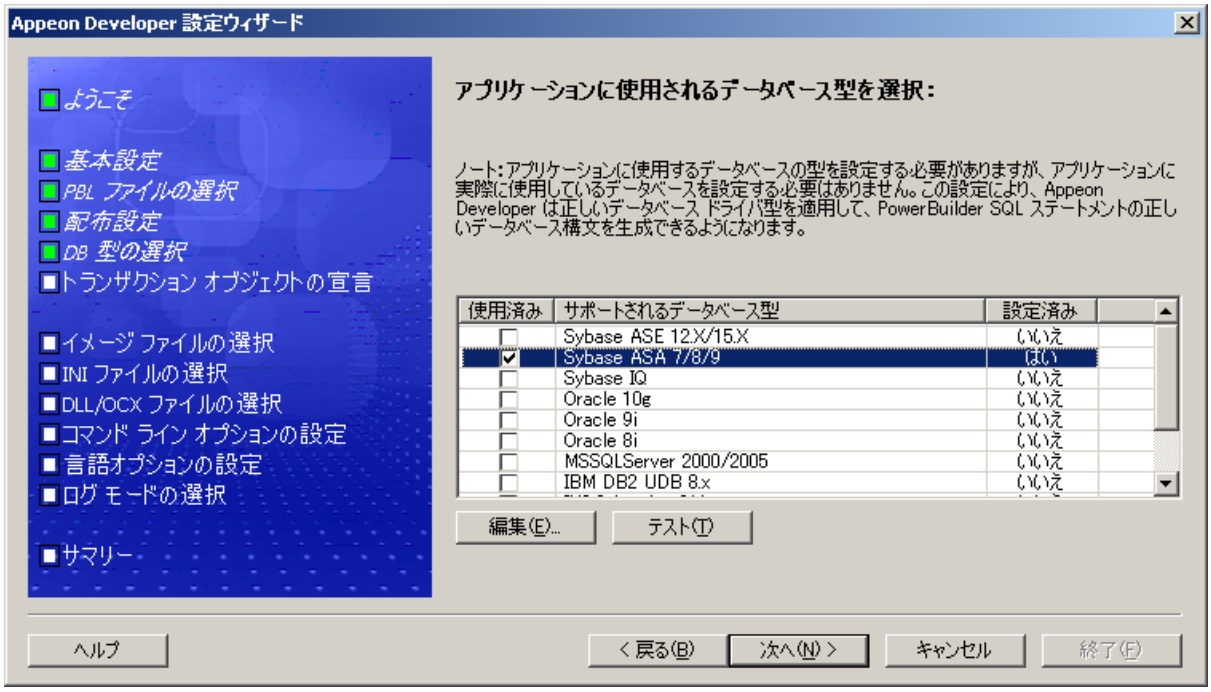


ステップ 3 ー［次へ］をクリックします。

7.3.4 DB 型を選択する

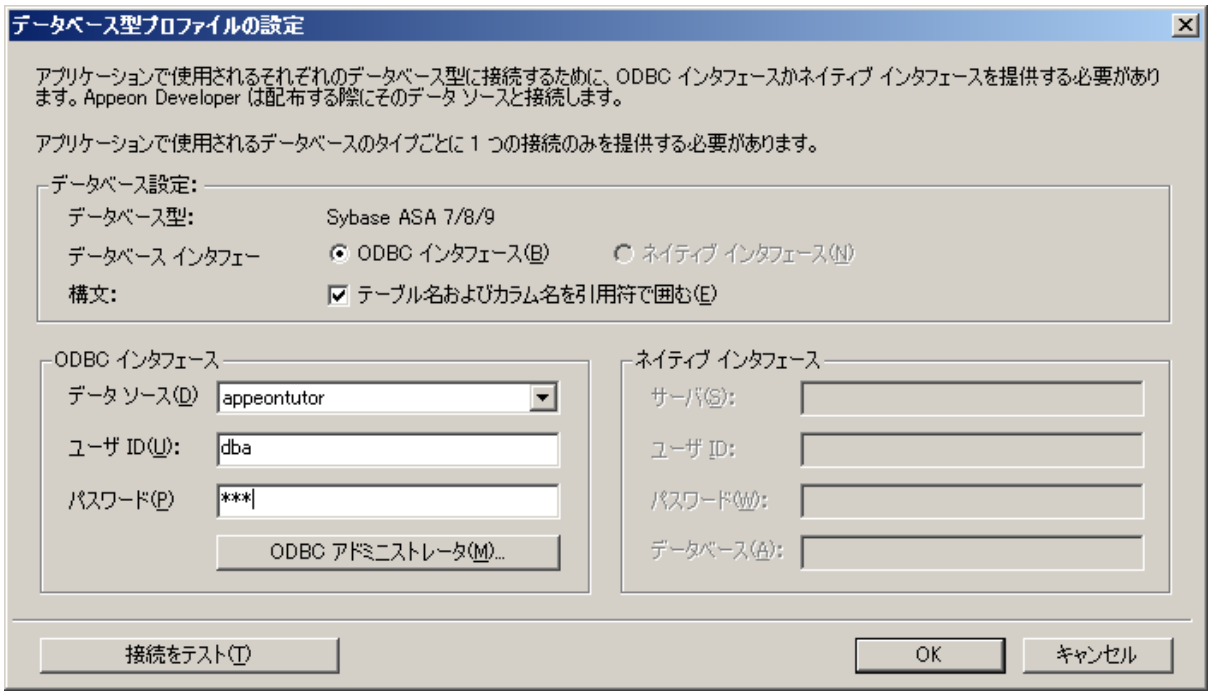
アプリケーションが使用するデータベースの種類を選択します。このチュートリアルで、データベース型は Sybase ASA 7/8/9/10 です。これは既にデフォルトとして選択されました。（図 7-33）

図 7-33：DB 型



Sybase ASA 7/8/9/10 が未設定の場合（設定済みカラムが“いいえ”に示す）、これを選択して［設定］ボタンをクリックしてください。図 7-34 に示すように、表示されたデータベース型のプロファイル設定ウィンドウにプロファイルを作成する必要があります。

図 7-34：データベース型のプロファイル設定ウィンドウ



ステップ 1 ― データベース型のプロファイル設定ウィンドウで、次の表に示すように ODBC インタフェースを設定します。

表 7-4：データベース型のプロファイル設定

項目	設定
ODBC インタフェース	［ODBC インタフェース］ラジオ ボタンを選択します。
データ ソース	ドロップダウン リストから“*appeontutor*”を選択します。 “appeontutor”が利用できない場合は Appeon デモ データベース（*appeonsample*）または他の ASA データソースも選択できます。
ユーザ ID	“*dba*”を入力します。“*appeonsample*”データソースを選択した場合、空白のままにします。
パスワード	“*sql*”を入力します。“*appeonsample*”データソースを選択した場合、空白のままにします。

ステップ 2 ― 図 7-35 に示すように、［テスト］をクリックして接続を確認します。

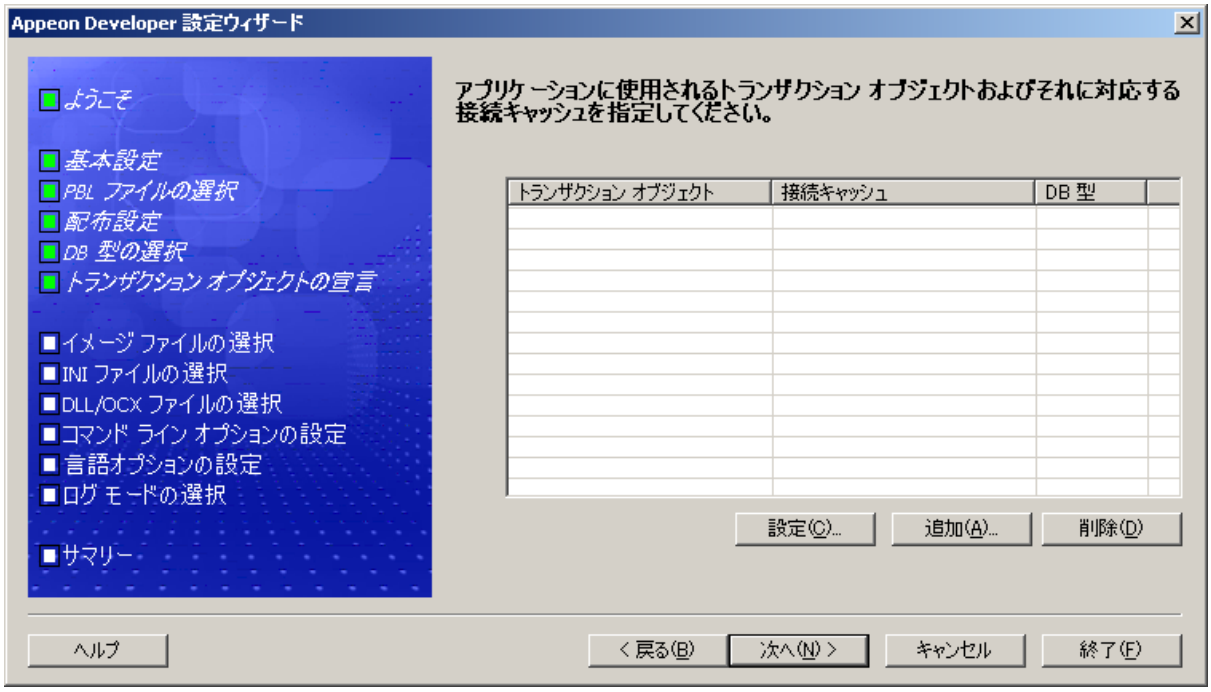
図 7-35：接続のテスト

ステップ 3 ―［OK］をクリックします。

7.3.5 トランザクション オブジェクトを宣言する

チュートリアル PowerBuilder アプリケーションが Web へ配布された後、Appeon Server は PowerBuilder アプリケーションに定義されたトランザクション オブジェクトを使わないで、データベース コネクション キャッシュを使ってデータベースへの接続をハンドルするようになります。そのために、Appeon チュートリアル PowerBuilder アプリケーションに使用されるトランザクション オブジェクトを適切なコネクション キャッシュに関連付ける必要があります。

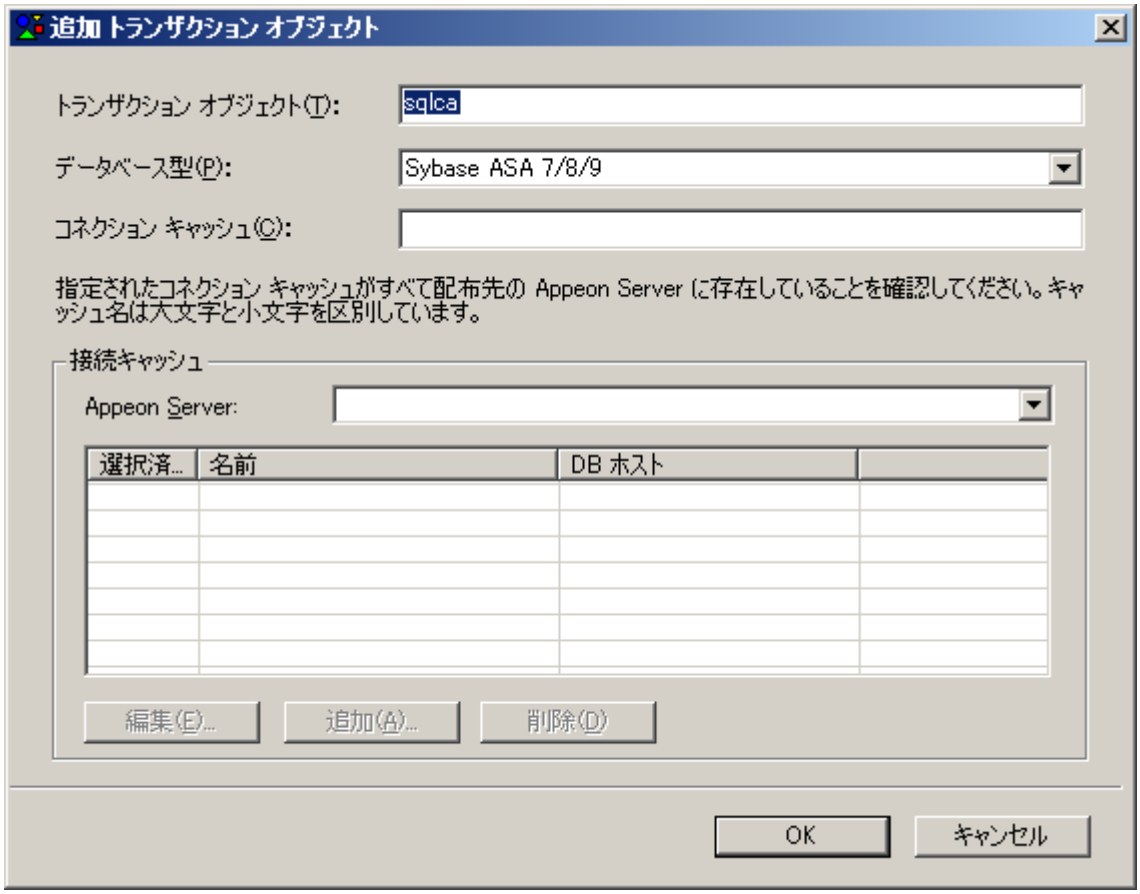
図 7-36：トランザクション オブジェクト マッピング



Appeon チュートリアル アプリケーションは *appeontutor* データベースを接続する SQLCA トランザクション オブジェクトを使用します。次のステップに従って、SQLCA トランザクション オブジェクトを適切なコネクション キャッシュを関連付けることができます：

ステップ 1 ー ［追加］ボタンをクリックします。追加トランザクション オブジェクト ダイアログ ボックスが表示されます。

図 7-37：トランザクション オブジェクトの追加

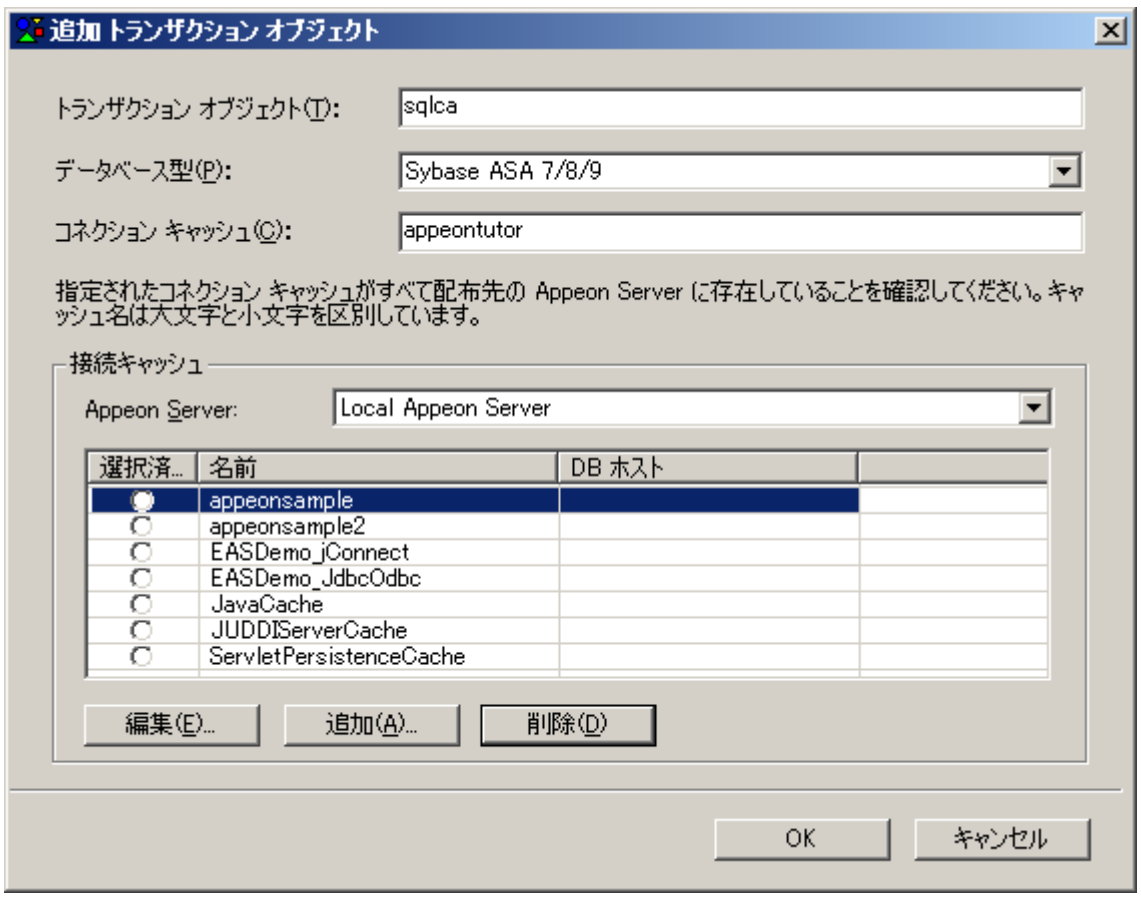


ステップ 2 ー 他のフィールドはそのままデフォルトで使います。

ステップ 3 ー Appeon Server 一覧から“Appeon Server Tutor”を選択します。既存の Appeon Server 内のコネクション キャッシュが一覧表示されます。

注意：このコネクション キャッシュ ツールは EAServer5.x でのみ利用できます。EAServer6.x では利用できません。サーバの管理コンソールからデータソースの設定を行う必要があります。詳細は「Appeon Server Configuration Guide」の Database Connection Setup の章を参照してください。

図 7-38：Appeon Server におけるコネクション キャッシュ



次のステップに従って、ODBC-JDBC ドライバによって appeontutor の ODBC データソースとリンクする EAServer の JDBC コネクション キャッシュを作成できます。

ステップ 5 ー［追加］をクリックして、追加ダイアログ ボックスが表示されます。

図 7-39：コネクション キャッシュの追加

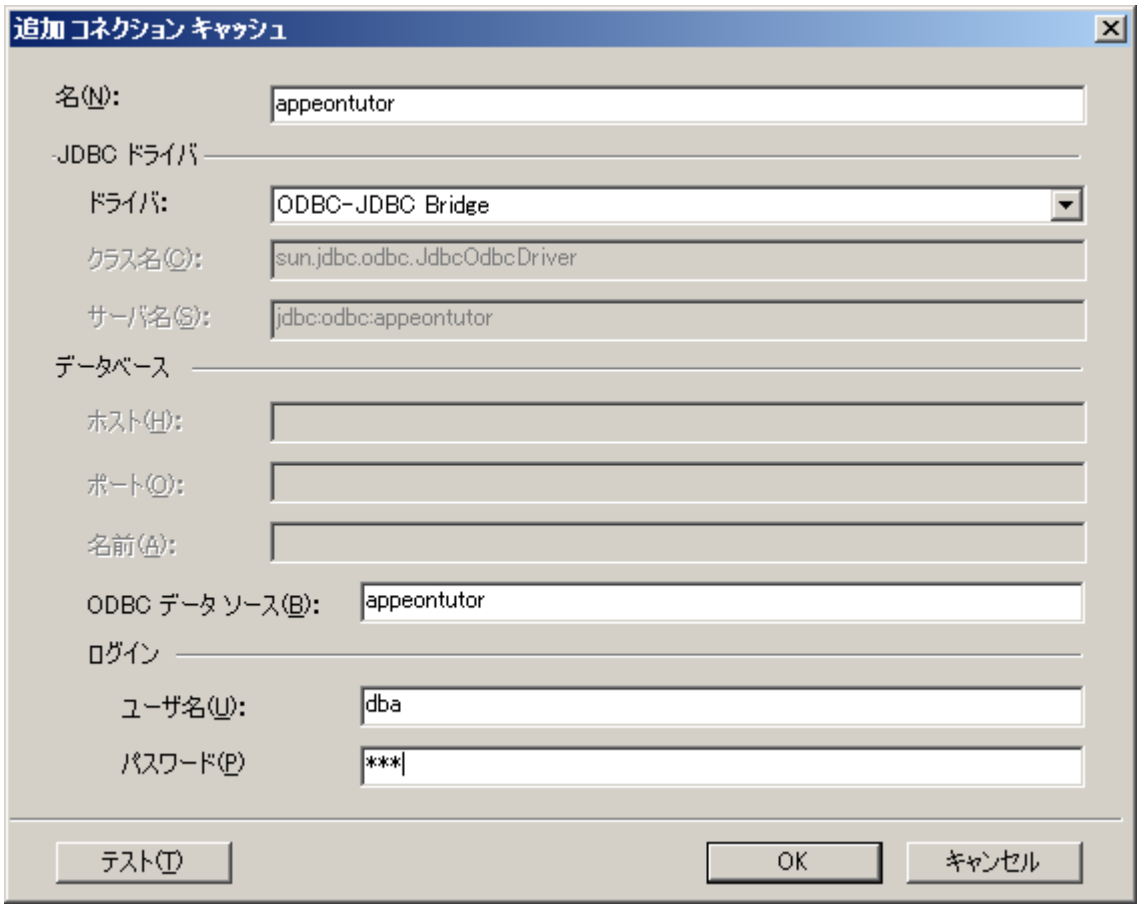


表 7-5 はコネクション キャッシュの設定方法を説明しています。

表 7-5：コネクション キャッシュの設定説明

設定	説明
名前	“*appeontutor*”をコネクション キャッシュの名前として入力します。
ドライバ	“ODBC-JDBC Bridge”をコネクション キャッシュのドライバ型として選択します。
ODBC データ ソース	“*appeontutor*”をデータ ソースの名前として選択します。 これにより、セクション 7.2.3「ODBC データ ソースの設定」で作成された *appeontutor* ODBC データ ソースへ接続できるようになります。
ユーザ名	“*dba*”を入力します。
パスワード	“*sql*”を入力します。

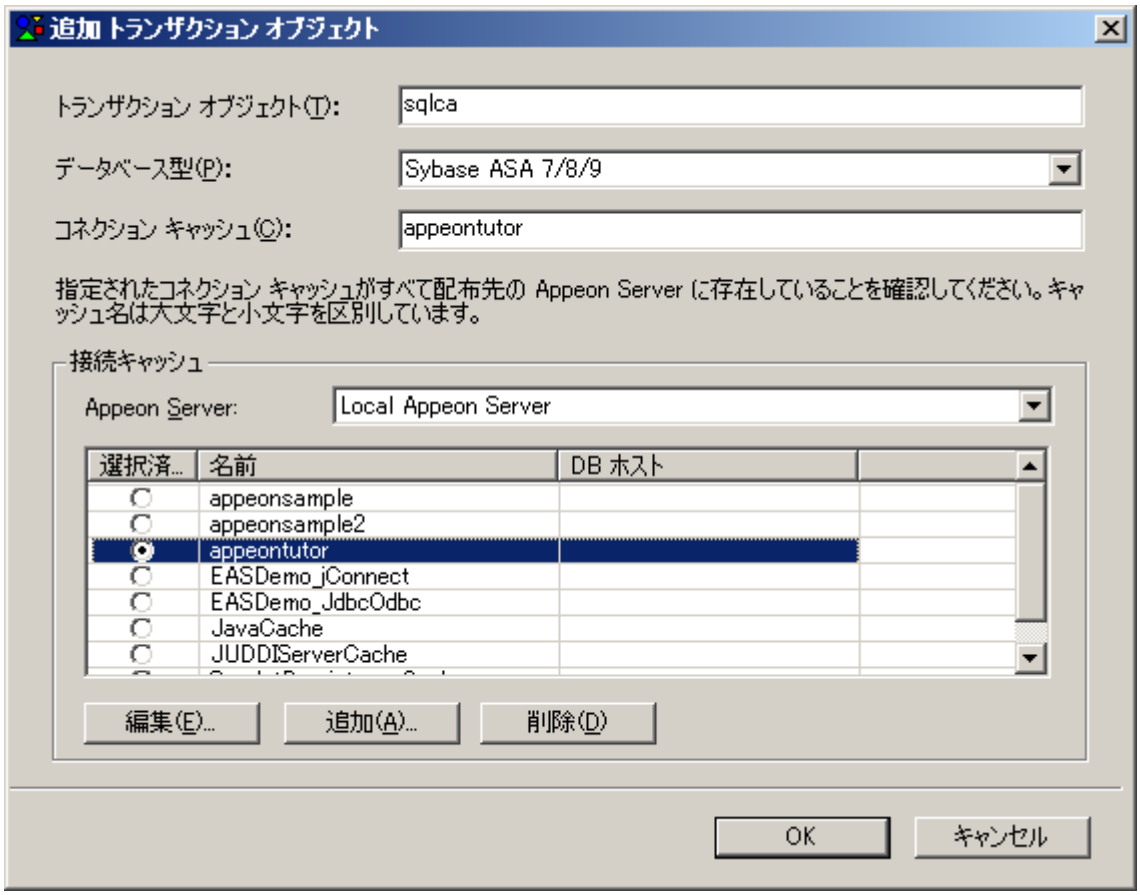
ステップ 6 ー［テスト］をクリックして、*appeontutor* データ ソースへの接続を確認します。

ステップ 7 ー［OK］をクリックして、追加ダイアログ ボックスを閉じます。

図 7-40 に示すように、*appeontutor* コネクション キャッシュが追加されます。

ステップ 8 ー *appeontutor* コネクション キャッシュの［選択済み］ボタンをクリックします。

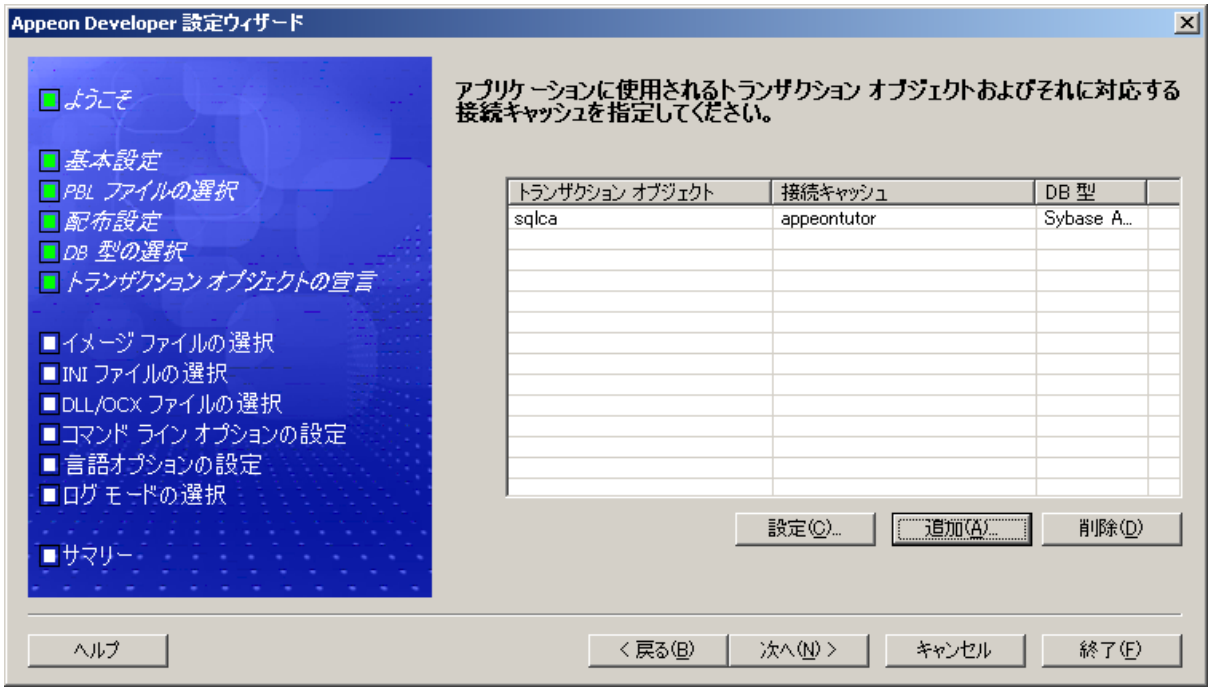
図 7-40：appeontutor が追加済み



ステップ 9 ー ［OK］をクリックして追加ダイアログ ボックスを閉じます。

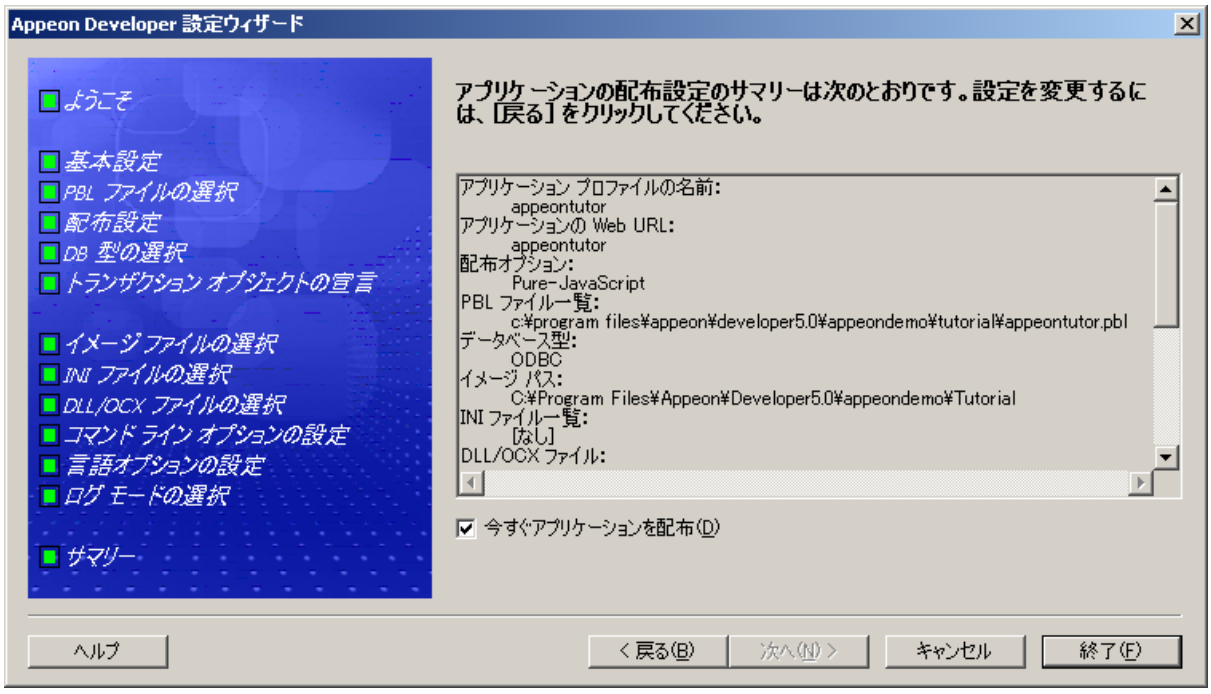
トランザクション オブジェクト sqlca が追加されました（図 7-41）。

図 7-41：sqlca が追加された



ステップ 10 ー サマリー ページが表示するまで［次へ］をクリックします。

図 7-42：サマリー



ステップ 11 ー サマリー ページで［終了］をクリックします。

Appeon チュートリアル アプリケーションが正しく追加されます。

図 7-42 に示すように、サマリー ページで［今すぐアプリケーションを配布する］オプションを選択する場合、Appeon 配布ウィザードが自動的に起動されることができます。

7.4 未サポート機能の解析

この章では、アプリケーションの機能を失うこと無くすべての機能が変換されるように、Appeon チュートリアル PowerBuilder アプリケーションを UFA ツールで未サポート機能を解析し、最適化、再コンパイルする方法について説明します。

ここでは、次の項目について学習します。

- チュートリアル アプリケーションの未サポート機能の解析
- チュートリアル アプリケーションのフル構築と最適化

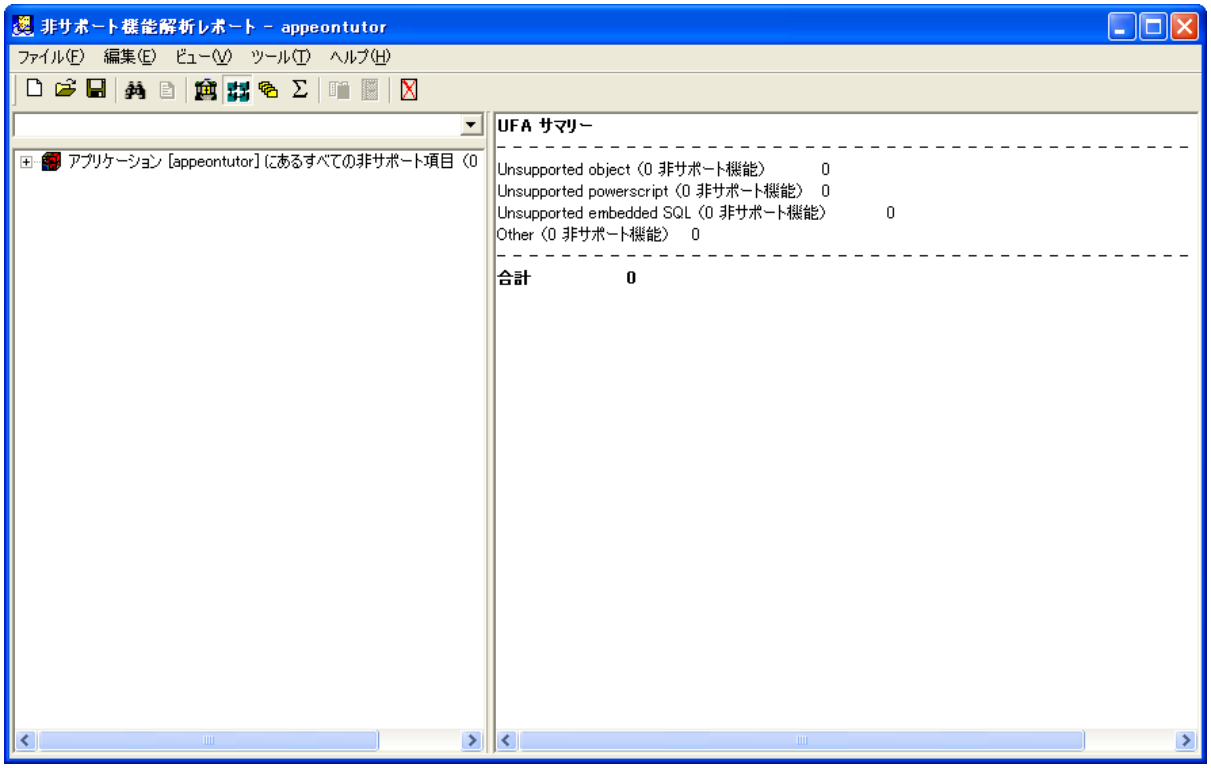
7.4.1 未サポート機能の解析

Appeon は、PowerBuilder アプリケーション内で Appeon によりサポートされないソースコードを検出するため、未サポート機能解析（Unsupported Feature Analysis、UFA と略称）ツールを提供します。

未サポート機能解析を実行するには、次のステップを実行します：

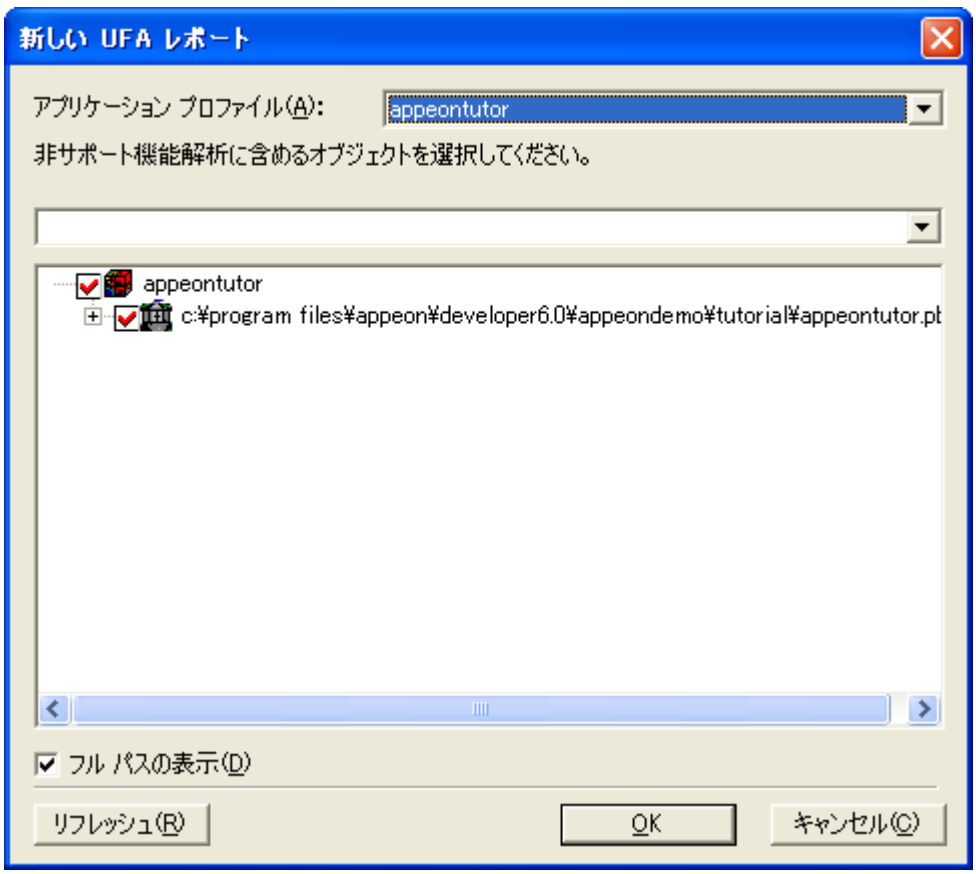
ステップ 1 － PowerBuilder IDE 内で、 Appeon Developer ツールバーの［解析］（  ） ボタンをクリックします。図に示すように、未サポート機能解析レポート ウィンドウ（UFA レポート ウィンドウ）が表示されます。（図 7-43）

図 7-43：UFA レポート ウィンドウ



ステップ 2 ー UFA レポート ウィンドウのメニューから［ファイル｜新規レポート］を選択します。図 7-44 に示すように、［新しい UFA レポート］ダイアログ ボックスが表示されます。

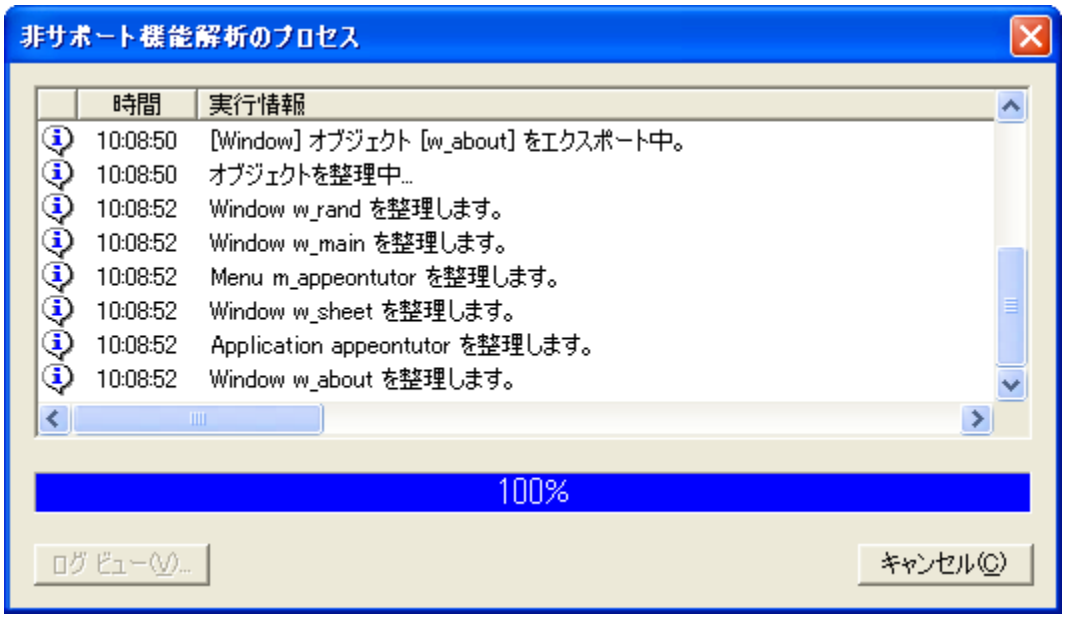
図 7-44：新しい UFA レポート ダイアログ ボックス



ステップ 3 ー［OK］をクリックし、アプリケーションを解析します。

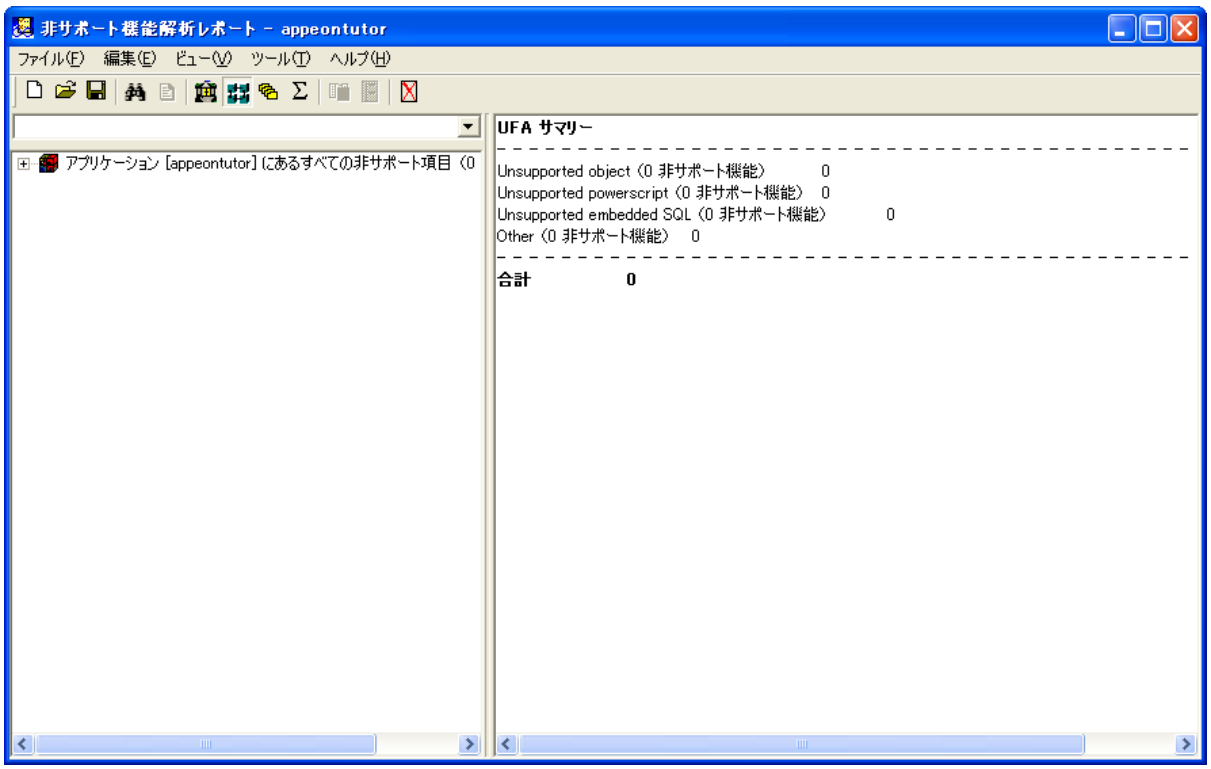
［未サポート機能解析ステータス］ダイアログ ボックスが表示され、図 7-45 に示すように未サポート機能の解析情況が表示されます。

図 7-45：未サポート機能解析ステータス



解析を完了すると、［閉じる］をクリックします。図 7-46 に示す未サポート機能解析レポートが（［未サポート機能解析レポート］ウィンドウ）が表示されます。左のツリー ビューからから未サポート機能を表示し、これを修正できます。

図 7-46：[未サポート機能解析レポート] ウィンドウ



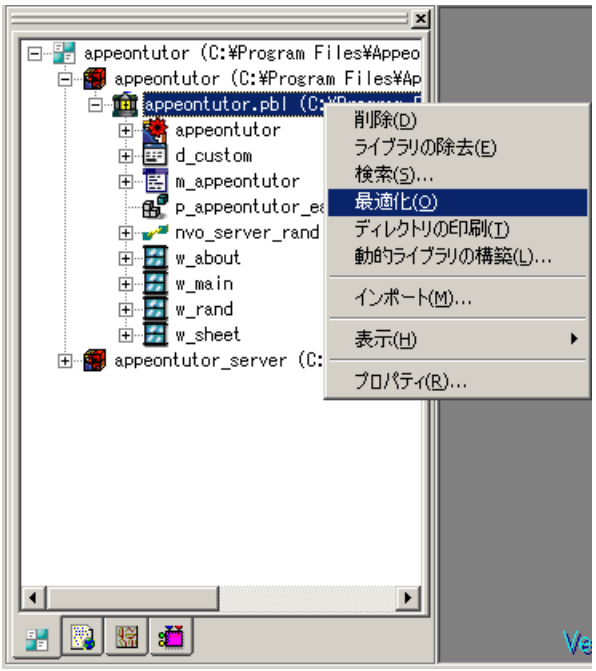
7.4.2 最適化とフル構築

Appeon により PowerBuilder アプリケーションの変換を行う前に、PBL を最適化する必要があります。さらに、PowerBuilder アプリケーションのフル構築を行う必要があります。

チュートリアル アプリケーションの最適化とフル構築を行うには、次のステップを実行します：

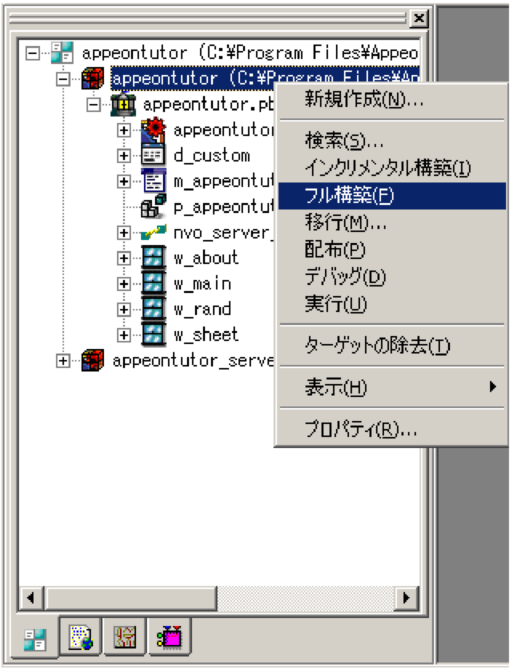
ステップ 1 ― PowerBuilder システム ツリー内で appeontutor.pbl を右クリックして、ポップアップ メニューから［最適化］を選択します。

図 7-47：アプリケーションの最適化



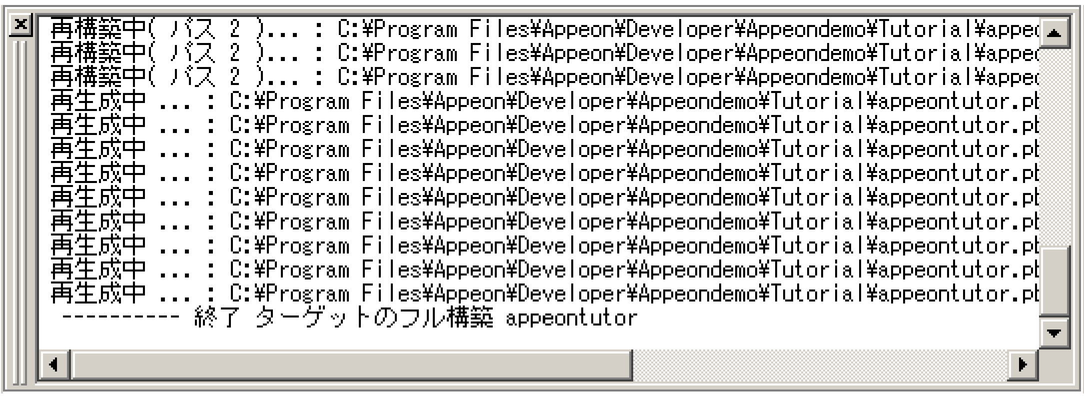
ステップ 2 – システム ツリー内で appeontutor ターゲットを右クリックして、ポップアップ メニューから［フル構築］を選択します。

図 **7-48**：アプリケーションのフル構築



ステップ 3 － チュートリアル アプリケーションがフル構築されます。フル構築された情報が出力ウィンドウに表示されます。フル構築が完了した後にエラーが無いことを確認してください。

図 **7-49**：フル構築終了



7.5 チュートリアル アプリケーションの配布

前のレッスンでは、Appeon チュートリアル PowerBuilder アプリケーションの未サポート機能を分析し、必要な構成を行いました。ここでは、Appeon の配布ウィザードによりチュートリアル アプリケーションを 自動的に Web に変換します。

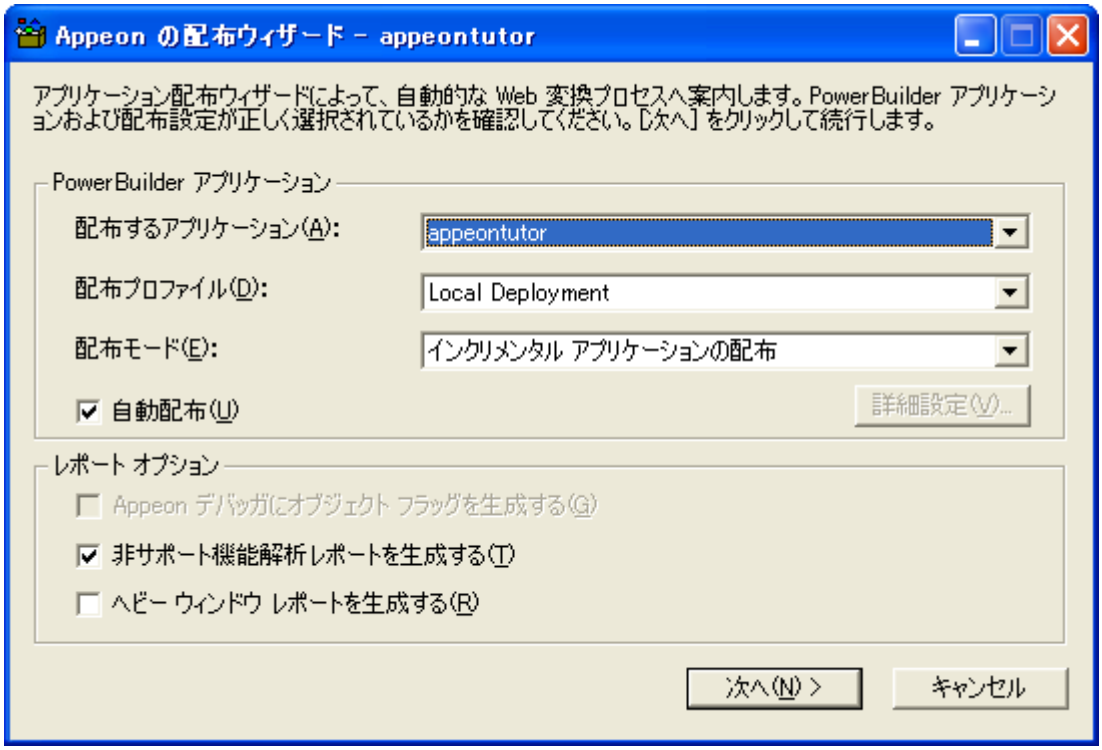
チュートリアル アプリケーションを Web へ自動的に配布するには、次のステップを実行します：

ステップ 1 － Appeon Server が起動されていることを確認します。詳細は「」を参照してください。

Appeon Server を起動するには、Windows の［スタート］メニューから［プログラム｜Appeon 6.0 for PowerBuilder｜Appeon Server｜%InstanceName%］を選択します。Appeon Server が正常に起動されると、「Accepting connections」というメッセージが表示されます。

ステップ 2 － PowerBuilder IDE で Appeon Developer ツール バーの［配布］（）ボタンをクリックし、Appeon の配布ウィザードを表示します。

図 7-50：Appeon の配布ウィザード



ウィザードの開始画面では、次の情報を表示しています：

- 配布するアプリケーション－現行のアプリケーションは、Appeon Developer アプリケーション プロファイル内にデフォルトとして設定されたアプリケーション（appeontutor）です。
- 配布プロファイル－配布プロファイルとして「Local Deployment for Tutorial」プロファイルを選択します。
- 配布モード－選択されたアプリケーションが初めての配布の場合、[フル アプリケーションの配布]がデフォルトとして選択されます。これ以後の配布に対

しては、[インクリメンタル アプリケーションの配布]がデフォルトとして選択されます。

- 自動配布 – この項目を選択すると、全プロセスが自動的に実行されます。

ステップ 3 – ［次へ］をクリックし、Web への配布プロセスを開始します。Appeon の配布ウィザードは、次の 3 つのタスクを実行します。

1. PowerBuilder アプリケーションのソースコードをエクスポートして解析します。
2. PowerBuilder アプリケーションのソースコード（PBL ファイル）を Web アプリケーション ファイルへ変換し、ローカル マシンに格納します。
3. ファイルのコピーまたは FTP を使用して、ローカルの Web ファイルを Web サーバへ転送し、データウィンドウを Appeon Server にアップロードします。

配布が完了すると、次のレポート ダイアログ ボックスが表示されます。

図 7-51：Appeon の配布ウィザード－配布終了

ステップ 4 – ［終了］をクリックします。

7.6 Web アプリケーションの実行

これまでの作業で、Appeon チュートリアル PowerBuilder アプリケーションは、 Web アプリケーションに変換され、適切なデータベースとの接続が設定されました。これで Web アプリケーションとして実行できます。

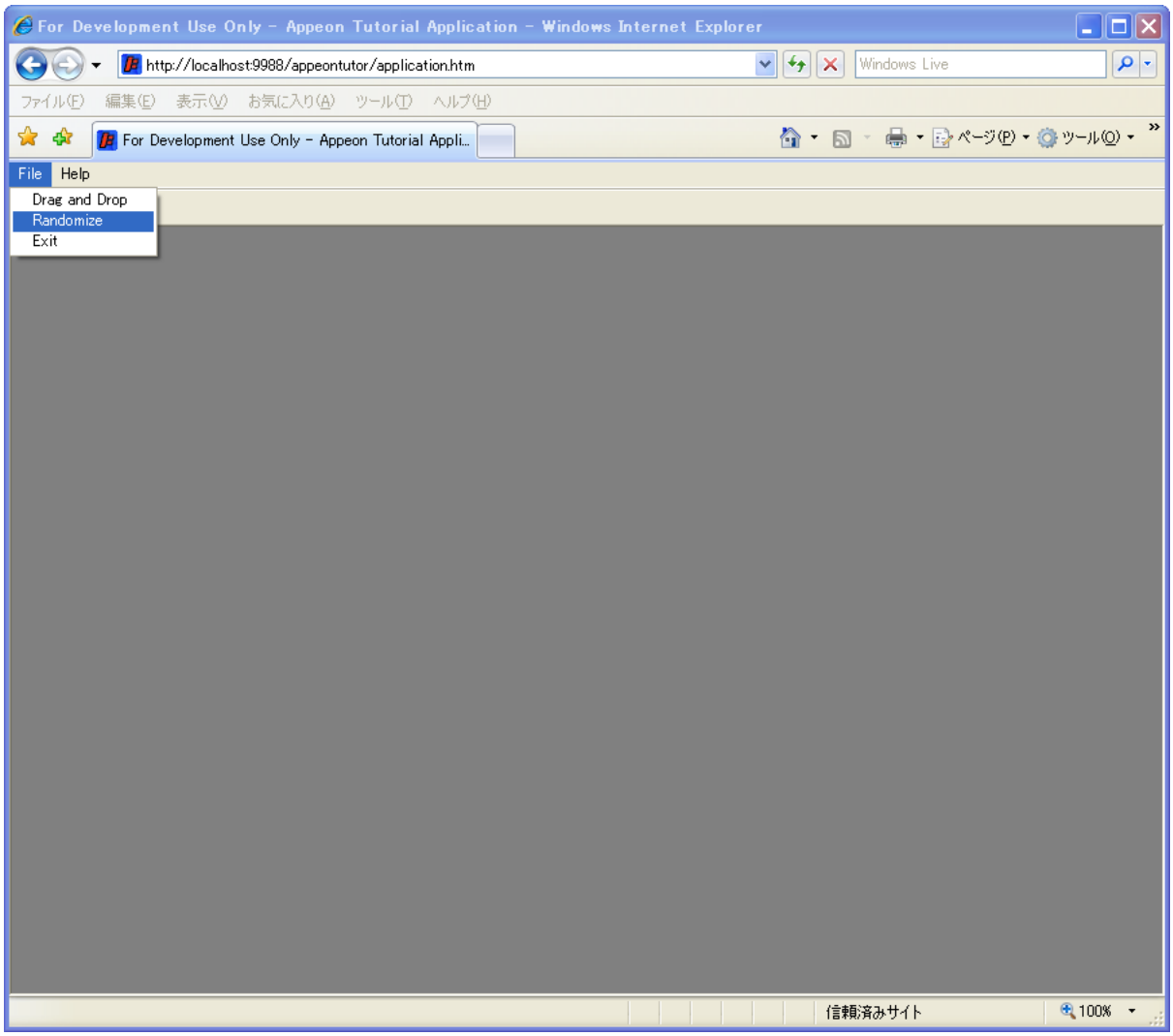
チュートリアル アプリケーションの変換後、Web アプリケーションとして起動するには、次のステップを実行します：

ステップ 1 － Appeon Server が実行されていることを確認します。

ステップ 2 － PowerBuilder IDE から Appeon Developer ツール バーの［実行］（  ）ボタンをクリックしてポップアップ メニューから「*appeontutor*」アプリケーションを選択するか、または IE ブラウザを開いて、アドレスバーに「http://localhost:9988/appeontutor/」と入力して、Enter キーを押します。

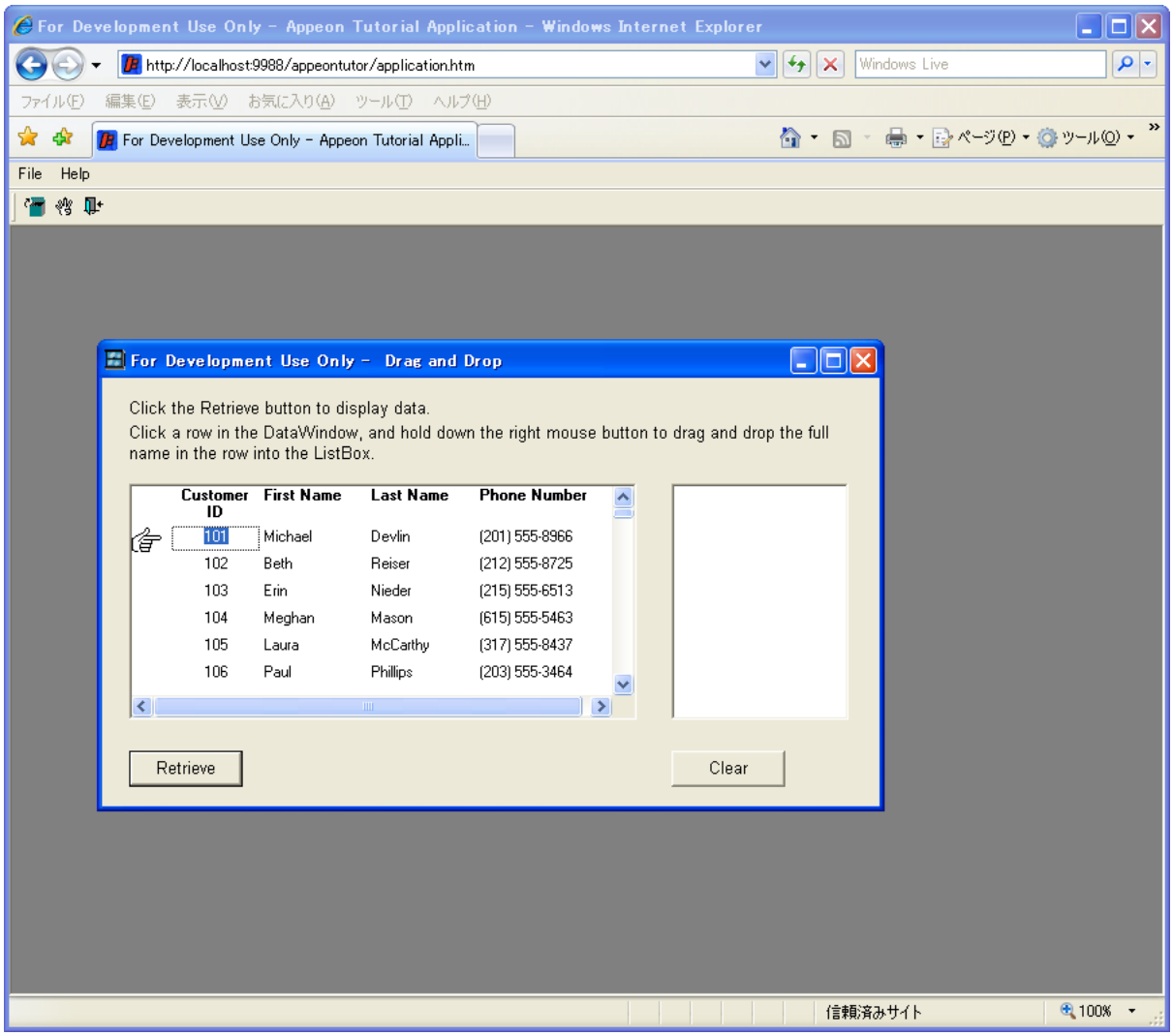
図 7-52 に示すように、Web アプリケーションが起動されます。

図 7-52：Appeon チュートリアル アプリケーション－Randomize



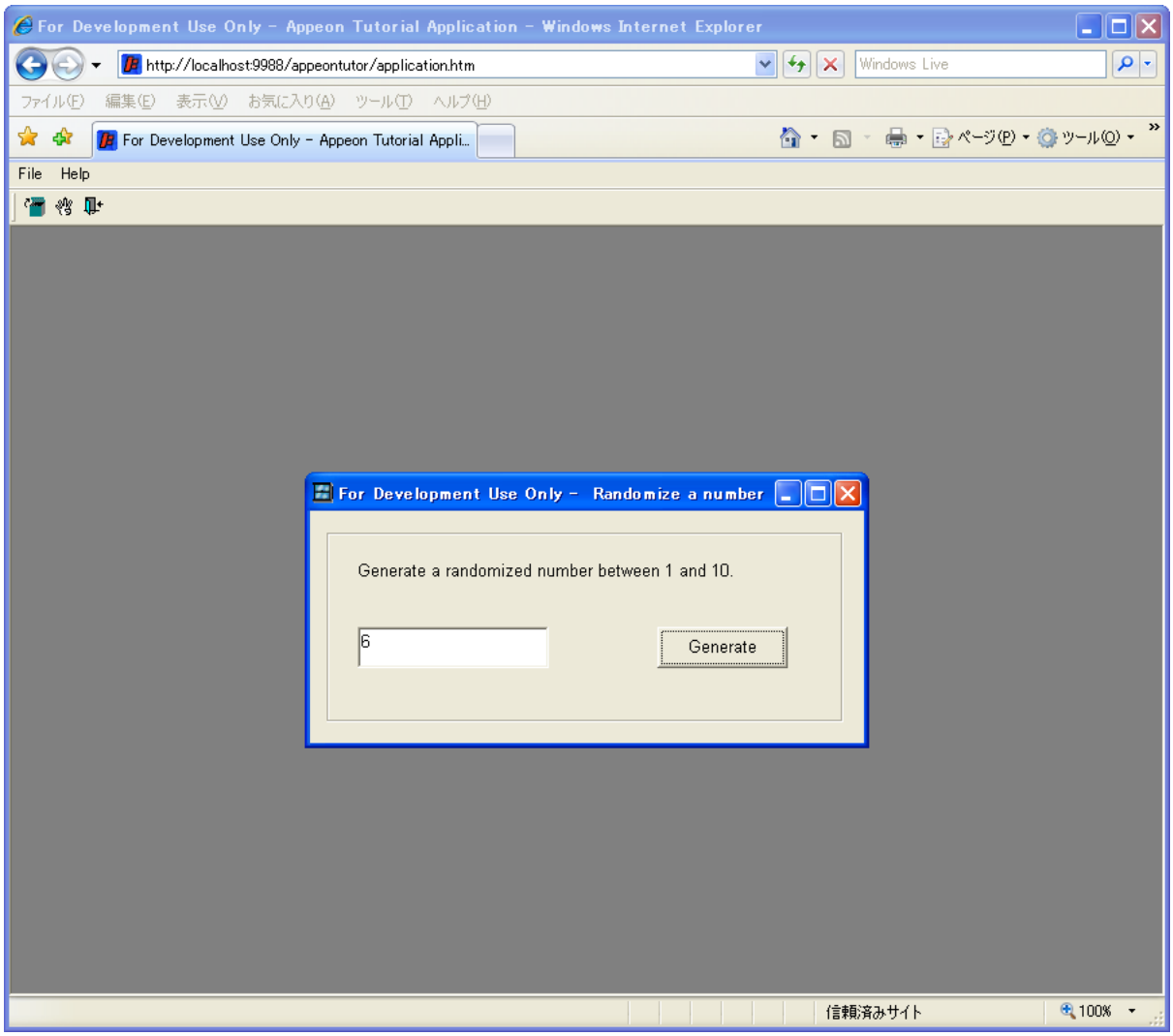
ステップ 3 ー Drag and Drop ウィンドウを開きます。

図 7-53：Appeon チュートリアル アプリケーション



ステップ 4 ー Randomize a number ウィンドウを開き、テストします。

図 7-54：Appeon チュートリアル アプリケーション



索引

N

O

P

R

S

W

あ

か

さ

た

は

ま

ら

わ