



**Tutorial: BlackBerry Application
Development using Custom Development**

Sybase Unwired Platform 1.5.2

DOCUMENT ID: DC01214-01-0152-03

LAST REVISED: February 2011

Copyright © 2011 by Sybase, Inc. All rights reserved.

This publication pertains to Sybase software and to any subsequent release until otherwise indicated in new editions or technical notes. Information in this document is subject to change without notice. The software described herein is furnished under a license agreement, and it may be used or copied only in accordance with the terms of that agreement.

To order additional documents, U.S. and Canadian customers should call Customer Fulfillment at (800) 685-8225, fax (617) 229-9845.

Customers in other countries with a U.S. license agreement may contact Customer Fulfillment via the above fax number. All other international customers should contact their Sybase subsidiary or local distributor. Upgrades are provided only at regularly scheduled software release dates. No part of this publication may be reproduced, transmitted, or translated in any form or by any means, electronic, mechanical, manual, optical, or otherwise, without the prior written permission of Sybase, Inc.

Sybase trademarks can be viewed at the Sybase trademarks page at <http://www.sybase.com/detail?id=1011207>. Sybase and the marks listed are trademarks of Sybase, Inc. ® indicates registration in the United States of America.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP AG in Germany and in several other countries all over the world.

Java and all Java-based marks are trademarks or registered trademarks of Sun Microsystems, Inc. in the U.S. and other countries.

Unicode and the Unicode Logo are registered trademarks of Unicode, Inc.

All other company and product names mentioned may be trademarks of the respective companies with which they are associated.

Use, duplication, or disclosure by the government is subject to the restrictions set forth in subparagraph (c)(1)(ii) of DFARS 52.227-7013 for the DOD and as set forth in FAR 52.227-19(a)-(d) for civilian agencies.

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

Introduction to Getting Started Tutorials	1
Overview of Getting Started Tutorials	1
Understanding the Unwired Platform Development Environment	2
Development in Eclipse	2
Understanding Fundamental Mobile Development Concepts	3
Mobile Business Objects	3
Synchronization Methods	4
Application Types	4
Data Sources	4
Switching Between Developer Profiles	4
Task Flow	7
Getting Started	9
Installing Sybase Unwired Platform	9
Starting Unwired Platform Services	9
Starting Sybase Unwired WorkSpace	10
Learning the Basics	10
Developing a BlackBerry Application	13
Installing the BlackBerry Java Plug-in for Eclipse	13
Deploying the Database Mobile Business Objects	14
Generating Java Object API Code	15
Creating the BlackBerry Project	17
Configuring the Generated Customer.java File	19
Creating the User Interface	20
Creating a Launch Configuration for the Project	35
Testing the Device Application on the BlackBerry Simulator	36
Learn More about Sybase Unwired Platform	41
Index	43

Introduction to Getting Started Tutorials

Getting started tutorials enable users of all levels to try Sybase® Unwired Platform with minimal setup. You can also use the tutorials to demonstrate system functionality and train users.

Overview of Getting Started Tutorials

The getting started tutorials demonstrate how to develop, deploy, and test mobile business objects, device applications, and message-based mobile workflow packages.

- Learn mobile business object (MBO) basics, and create a mobile device application:
 - *Tutorial: Mobile Business Object Development*
 - *Tutorial: BlackBerry Application Development using Device Application Designer*
 - *Tutorial: Windows Mobile Device Application Development using Device Application Designer*
- Create native mobile device applications:
 - *Tutorial: BlackBerry Application Development using Custom Development*
 - *Tutorial: iPhone Application Development using Custom Development*
 - *Tutorial: Windows Mobile Application Development using Custom Development*
- Create a mobile workflow package:
 - *Tutorial: Mobile Workflow Package Development*

The getting started tutorials demonstrate a cross section of basic functionality, which includes creating MBOs that can be used in replication-based or message-based synchronization; and using various Sybase Unwired WorkSpace development tools, independent development environments, and device types.

Table 1. Tutorial summary

Tutorials	Mobile business objects (MBOs)	Synchroni- zation types	Development tools	Device types
Tutorial: Mobile Business Object Development	Create new MBOs	Replication-based	Sybase Unwired WorkSpace	N/A
Tutorial: BlackBerry Application Development using Device Application Designer	Reuse MBOs	Replication-based	Device Application Designer	BlackBerry

Tutorials	Mobile business objects (MBOs)	Synchronization types	Development tools	Device types
Tutorial: BlackBerry Application Development using Custom Development	Create new MBOs	Replication-based	Sybase Unwired WorkSpace	BlackBerry
Tutorial: iPhone Application Development using Custom Development	Create new MBOs	Message-based	Sybase Unwired WorkSpace	iPhone
Tutorial: Windows Mobile Application Development using Device Application Designer	Reuse MBOs	Replication-based	Device Application Designer	Windows Mobile
Tutorial: Windows Mobile Device Application Development using Custom Development	Create new MBOs	Message-based	Sybase Unwired WorkSpace	Windows Mobile
Tutorial: Mobile Workflow Package Development	Create new MBOs	Message-based	Mobile Workflow Forms Editor	Windows Mobile

Understanding the Unwired Platform Development Environment

Learn more from the getting started tutorials by understanding basic development environment concepts. Sybase Unwired Platform provides an Eclipse development environment.

Development in Eclipse

Sybase Unwired WorkSpace is a plug-in to your Eclipse development environment that provides tools for creating mobile applications.

Unwired WorkSpace includes back-end integration tools that connect Unwired Server to enterprise data sources, allowing you to create mobile business objects (MBOs) from the back-end business data model.

Developers can perform MBO code generation at any time and use this MBO model code along with the user interface code that users write in a native integrated development environment (IDE). This makes the code available to transition from the Unwired WorkSpace MBO development tool to the fully extensible and open development environment provided for device platforms from third-party vendors. Optionally, use the Device Application

Designer to create prototype device applications for BlackBerry and Windows Mobile devices.

Developers can use the Mobile Workflow Forms Editor to develop message-based mobile workflow packages for Windows Mobile and iOS devices.

Understanding Fundamental Mobile Development Concepts

Learn more from the getting started tutorials by understanding basic mobile development concepts.

Learn more about these concepts:

- *Fundamentals*
- *Sybase Unwired WorkSpace – Mobile Business Object Development*

Mobile Business Objects

Mobile business objects help form the business logic for mobile applications.

A mobile business object (MBO) is derived from a data source (such as a database server, Web service, or SAP® server). MBOs are deployed to Unwired Server, and accessed from mobile device application clients. MBOs include:

- Implementation-level details – metadata columns that include information about the data from a data source.
- Abstract-level details – attributes that correspond to instance-level properties of a programmable object in the mobile client, and map to data source output columns. Parameters correspond to synchronization parameters on the mobile client, and map to data source arguments. For example, output of a SQL SELECT query are mapped as attributes, and the arguments in the WHERE clause are mapped as synchronization parameters, so that the client can pass input to the query.
MBO operations include parameters that map to data source input arguments. Operation parameters determine information a client passes to the enterprise information system (EIS).
- Relationships – defined between MBOs by linking attributes and parameters in one MBO, to attributes and parameters in another MBO.

You can define MBOs using either a top-down approach—first designing attributes and parameters, then binding them to a data source; or a bottom-up approach—first specifying a data source, then automatically generating attributes and parameters from it.

A mobile application package includes MBOs, roles, and data source connection mappings, and other artifacts that are delivered to the Unwired Server during package deployment.

Synchronization Methods

Developers can use either replication-based or message-based synchronization to move data and transactions between device application clients and Unwired Server.

The choice depends on the target device platform, application requirements, target platform, and the nature of data changes and activity between Unwired Server and clients, for example, mobile workflow forms always use message-based synchronization.

Unwired Server manages and maintains data freshness between multiple data sources and device application clients through synchronization.

Application Types

Sybase Unwired Platform supports two choices for application type. First is the native application type, and the other is the container-based business workflow type.

The native application model enables the developer to write custom code (C#, Java, or Objective-C, depending on the platform) to create a device application. The native application model is supported on BlackBerry, iOS, Windows Mobile, and Windows platforms. The choice depends on the functionality desired in the application, and the need to access third-party and platform-provided APIs. Optionally, use the Device Application Designer to create prototype device applications for BlackBerry and Windows Mobile devices.

The business workflow model offers a fast and simple way to build applications that support simple business workflows, such as approvals and requests. The workflow model is supported on iOS, Windows Mobile, and Windows platforms.

Data Sources

A data source is the enterprise information system where data is retrieved from and transactions are executed. A connection profile is a design-time connection to a data source. Connection profiles are created to specific data source by providing connection information such as host, port, login, and password among others. The connection profiles are used to define MBOs and operations, and mapped to existing, or used to create new, server connections when the package is deployed to Unwired Server..

Unwired Platform hides the interaction complexity with datasource-specific protocols, such as JDBC™ for database and SOAP for Web services.

Unwired Platform currently supports multiple EIS connection types. See *Supported Third-Party Software and Hardware* for information.

Switching Between Developer Profiles

Switch between basic and advanced developer profiles in the Mobile Application Diagram.

If you do not see an Unwired Workspace feature (wizard, property, or Workspace Navigator item) that you expect or need, switch to the advanced developer profile, or modify developer

profile settings. To use backend data sources other than those supplied by Sybase Unwired Platform, you must switch to the advanced developer profile to see the Server Connection Mapping page when deploying the Mobile Business Object package.

1. Right-click in the Mobile Application Diagram and select **Switch Developer Profile > Basic/Advanced**.
2. You can also select **Window > Preferences > Sybase, Inc. > Mobile Development > Developer Profile** to directly view or modify the developer profile preference settings. Basic is the default developer profile.

Task Flow

Sybase Unwired WorkSpace Eclipse tutorials explain how to develop, deploy, and run a mobile application.

Table 2. Eclipse tutorials

Task	Goals	Steps required to complete the task
Getting Started	<ul style="list-style-type: none"> • Install all required WorkSpace components and external resources. • Start Unwired Server, then use Sybase Control Center to connect to the server. • Open the Mobile Development perspective, and become familiar with the views of the perspective, the Mobile Application Diagram, and the Device Application Designer. 	<ul style="list-style-type: none"> • <i>Installing Sybase Unwired Platform</i> on page 9 • <i>Starting Unwired Platform Services</i> on page 9 • <i>Starting Sybase Unwired WorkSpace</i> on page 10 • (Optional) <i>Learning the Basics</i> on page 10 <hr/> <p>Note: These steps are prerequisites for the rest of this tutorial. You need to perform them only once.</p>
Developing Database Mobile Business Objects	<ul style="list-style-type: none"> • Create a mobile application project and a connection to the database. • Create two mobile business objects, and create a relationship between them. • Deploy the mobile business objects to Unwired Server. 	Complete the <i>Tutorial: Mobile Business Object Development</i> .

Task Flow

Task	Goals	Steps required to complete the task
Developing a BlackBerry Application	Generate Java code for the BlackBerry platform, create the user interface for the application, and run it on the Simulator.	<ul style="list-style-type: none">• <i>Installing the BlackBerry Java Plugin for Eclipse</i> on page 13• <i>Deploying the Database Mobile Business Objects</i> on page 14• <i>Generating Java Object API Code</i> on page 15• <i>Creating the BlackBerry Project</i> on page 17• <i>Creating the User Interface</i> on page 20• <i>Creating a Launch Configuration for the Project</i> on page 35• <i>Testing the Device Application on the BlackBerry Simulator</i> on page 36

Getting Started

Goal: Install and learn about Sybase Unwired Platform and its associated components.

The following tasks are required, unless otherwise noted, for all tutorials, but you need to perform them only once.

1. *Installing Sybase Unwired Platform* on page 9
2. *Starting Unwired Platform Services* on page 9
3. *Starting Sybase Unwired WorkSpace* on page 10
4. (optional) *Learning the Basics* on page 10

Installing Sybase Unwired Platform

Goal: Install Sybase Unwired Platform.

Install these Sybase Unwired Platform components:

- Data Tier
- Unwired Server
- Unwired WorkSpace
- Device Application Designer
- Windows Mobile .NET components (for developing device applications in Visual Studio)

If Unwired Platform is already installed and any of these components are missing:

1. Start the Sybase Unwired Platform installer.
2. Follow the instructions in the installation wizard.
3. Select the required components, and complete the installation.

For complete installation instructions, see the *Sybase Unwired Platform Installation Guide* and *Release Bulletin*.

Starting Unwired Platform Services

Goal: Start Unwired Server and the sample database.

In Windows, select **Start > Programs > Sybase > Unwired Platform<version> > Start Unwired Platform Services** .

Starting Sybase Unwired WorkSpace

Goal: Start Unwired WorkSpace.

1. In Windows, select **Start > Programs > Sybase > Unwired Platform<version> > Unwired WorkSpace**.

Sybase Unwired WorkSpace opens, and displays the Welcome page with links to product information, and to the product.

2. To read more about Sybase Unwired WorkSpace concepts and tasks, select **Help > Help Contents** from the main menu.

Learning the Basics

Goal: Learn about Sybase Unwired WorkSpace and how to access help.

Prerequisites

Start Unwired WorkSpace.

Task

1. From the Welcome page, select any of the links to familiarize yourself with the Unwired WorkSpace environment.

To close this page, click the **X**. You can reopen this page by selecting **Help > Welcome**.

2. Select **Start Development** to access the Sybase Unwired WorkSpace development environment. Look at the area (window or view) that you will be working in to access, create, define, and update mobile business objects (MBOs).

View	Description
WorkSpace Navigator	This view displays mobile application project folders, each of which contains all project-related resources in subfolders, including MBOs, data source references to which the MBOs are bound, personalization keys, and so on. Use this view to review and modify MBO-related properties.

View	Description
Enterprise Explorer	A window that provides functionality to connect to various enterprise back-end systems; for example, database servers, SAP servers, and Sybase Unwired Server.
Mobile Application Diagram	<p>A graphical editor where you create and define mobile business objects.</p> <p>Use the Mobile Application Diagram to create MBOs (including attributes and operations), then define relationships with other MBOs. You can:</p> <ul style="list-style-type: none"> • Create MBOs in the Mobile Application Diagram using Palette icons and menu selections – either bind or defer binding to a data source, when creating an MBO. For example, you may want to model your MBOs before creating the data sources to which they bind. This is sometimes called the top-down approach. • Drag items from Enterprise Explorer and drop them onto the Mobile Application Diagram to create the MBO – quickly creates the operations and attributes automatically based on the data source being dropped on the Mobile Application Diagram. This is sometimes called the bottom-up approach. <p>Each new mobile application project generates an associated Mobile Application Diagram.</p>
Palette	Access the Palette from the Mobile Application Diagram. It provides controls, such as the ability to create MBOs, add attributes and operations, and define relationships, by dragging and dropping the corresponding icon onto the Mobile Application Diagram or existing MBO.
Properties view	Select an object in the Mobile Application Diagram to display and edit its properties in the Properties view. You cannot create an MBO from the Properties view, but generally, most development and configuration is performed here.

View	Description
Outline view	Displays an outline of the file that is currently open in the editor area, and lists structural elements. The contents are editor-specific.
Problem view	Displays problems, errors, or warnings that you may encounter.

3. To access the online help, select **Help > Help Contents** from the main menu bar.
4. Expand any of the documents that appear in the left pane. Some documents are for Sybase Unwired Platform, while others are for the Eclipse development environment.

Developing a BlackBerry Application

Generate code for the BlackBerry platform, develop a BlackBerry device application with code, and test its functionality.

Prerequisites

Complete these tasks:

- *Getting Started* on page 9.
- Finish the *Tutorial: Developing Database Mobile Business Objects*.
- Install the BlackBerry Java Plug-in for Eclipse.

Task

The device application communicates with the database mobile business objects that are deployed to Unwired Server. Develop the device application:

1. Open the SUP101 Mobile Application Project if it is not already open:
In WorkSpace Navigator, right-click the SUP101 folder and select **Open in Diagram Editor**.
2. *Deploying the Database Mobile Business Objects* on page 14
3. *Generating Java Object API Code* on page 15
4. *Configuring the Generated Customer.java File* on page 19
5. *Creating the User Interface* on page 20
6. *Testing the Device Application on the BlackBerry Simulator* on page 36

Installing the BlackBerry Java Plug-in for Eclipse

The Device Application Designer supports the BlackBerry Java Plug-in for Eclipse, which allows you to generate the device application code using the Device Application Designer code generation wizard, then debug the generated code.

Prerequisites

You must have a BlackBerry developer account to download the BlackBerry Java Plug-in for Eclipse. You may be required to register if you do not already have an account.

Task

Note: To ensure that you are using the supported version of the BlackBerry Java Plug-in for Eclipse, see the topic *Sybase Unwired Platform 1.5.5 > New Features > Supported Hardware and Software*.

1. Shut down Unwired WorkSpace.
2. Go to <http://us.blackberry.com/developers/javaappdev/> and download the BlackBerry Java Plug-in for Eclipse (full installer) to a temporary folder.
3. Double-click the setup application file.
4. On the Introduction page, click **Next**.
5. Accept or decline the terms of the license agreement and click **Next**.
6. Choose <UnwiredPlatform_InstallDir>\UnwiredPlatform\Eclipse as the installation directory and click **Next**.
7. Review the information on the Pre-installation Summary screen and click **Install**.
8. Click **Done**.

Deploying the Database Mobile Business Objects

Goal: Deploy the mobile application project that contains the database mobile business objects to the server.

Prerequisites

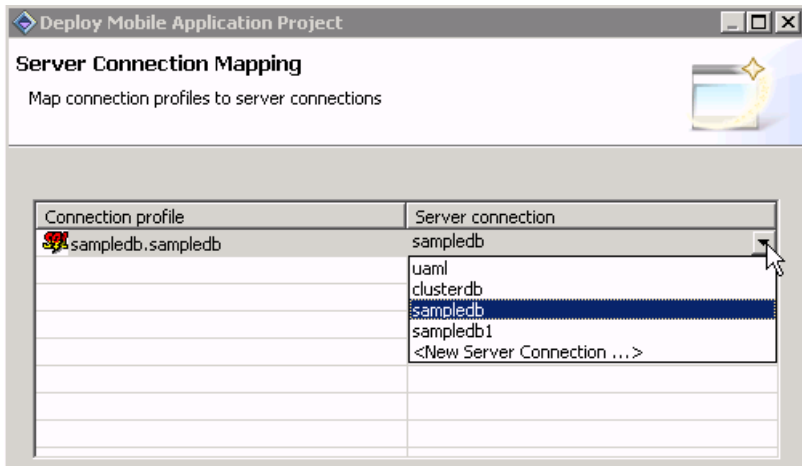
Finish the *Tutorial: Developing Database Mobile Business Objects*. You must be connected to both the sampledb database and Unwired Server.

Task

1. Switch to the Mobile Development perspective.
2. In Workspace Navigator, right-click in the SUP101 Mobile Application Diagram , and select **Deploy Project**.
3. In the first page of the Deploy Mobile Application Project, accept the defaults, select **Replication-based**, and click **Next**.
4. In the Contents page, select the **customer** and **sales_order** MBOs and click **Next**.
5. In the Package Jars page, click **Next**.

Note: This window appears only if you are using the Advanced developer profile.

6. In the Target Server page, from the list of available servers, select **My Unwired Server** and click **Refresh**.
Once connected, accept the default Domain and Security configuration settings, and click **Next**.
7. If you have multiple server connections, the Server Connection Mapping page displays. Select the **sampledb** server connection from the drop-down list and click **Finish**.



The deployment progress window displays.

8. In the Deployment status window, click **OK**.

A status window indicates progress and a successful deployment.

9. To view the deployed project, in Enterprise Explorer, expand **My Unwired Server > Domains > default > Packages**.

The server package *sup101:1.0* into which you deployed the MBOs appears in the Packages folder. The two MBOs appear in the Mobile Business Objects folder.

Generating Java Object API Code

Goal: Generate object API code for the SUP101 mobile application project.

1. Right-click in the SUP101 the Mobile Application Diagram and select **Generate Code**.
2. In the next page, select **Continue without a configuration**, and click **Next**.

Note: This page of the code generation wizard is seen only if you are using the Advanced developer profile.

3. Enter the information for these configuration options:

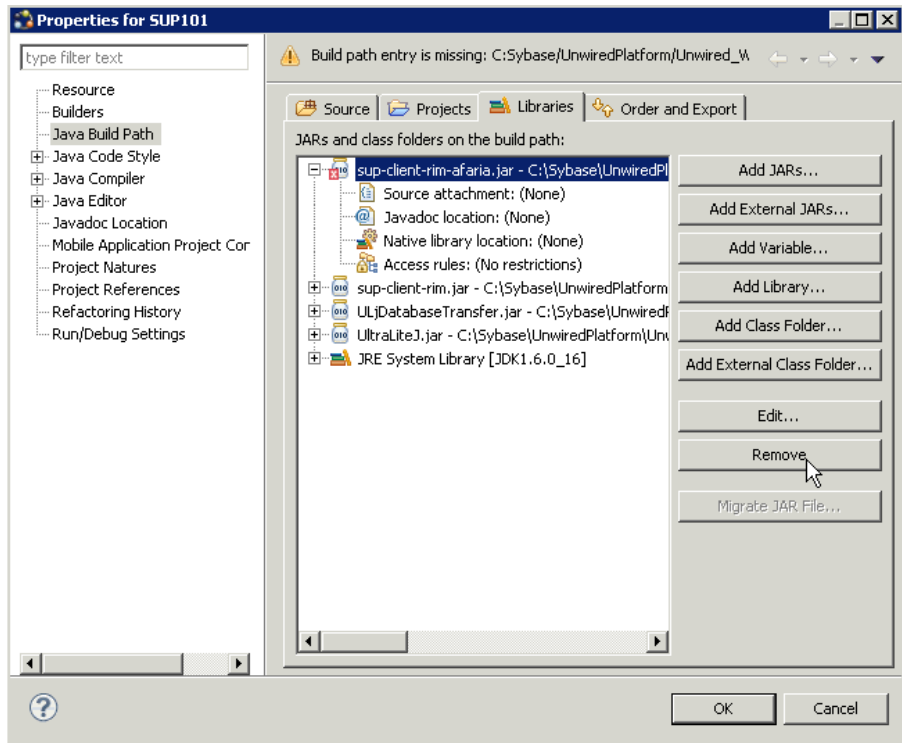
Option	Description
Language	Select Java .
Platform	Select Java ME for BlackBerry .
Unwired Server	Select My Unwired Server .
Server domain	The first domain in the list is chosen by default. Keep the default.

Option	Description
Page size	<p>Leave blank. If you do not set the page size, the default page size is 4k at runtime.</p> <p>The page size should be larger than the sum of all the attributes' max-length and must be valid for the database.</p> <hr/> <p>Note: The page size option is not enabled for message-based applications.</p>
Package	Leave this blank.
Destination	Specify the Project path for the generated device client files: \SUP101\Generated Code.
Replication-based	Select to use replication-based synchronization for the application.
Message-based	This option is not supported for Java applications.
Backward compatible	Leave unchecked.

4. Click **Next**.
5. In Select Mobile Objects, select all the MBOs in the mobile application project or select MBOs under a specific domain, whose references, metadata, and dependencies (referenced MBOs) are included in the generated device code.
Dependent MBOs are automatically added (or removed) from the Dependencies section depending on your selections.

Note: Code generation fails if the server-side (run-time) enterprise information system (EIS) data sources referenced by the MBOs in the project are not running and available to connect to when you generate object API code.

6. (Optional) Select **Generate metadata classes** to generate metadata for the attributes and operations of each generated client object.
7. (Optional) Select **Generate JavaDoc** to generate API documentation from the source code.
8. Click **Finish**.
9. Click **OK** in the Success code generation dialog.
10. An error icon appears next to the SUP101 mobile application project. This appears because the project is looking for a library that does not exist. To fix the error:
 - a) Right-click the **SUP101** mobile application project and select **Properties**.
 - b) Select **Java Build Path** in the left pane, then click the **Libraries** tab.
 - c) Select the `sup.client.rim.afari` build path that shows the error icon, and click **Remove**.



d) Click **OK**.

Creating the BlackBerry Project

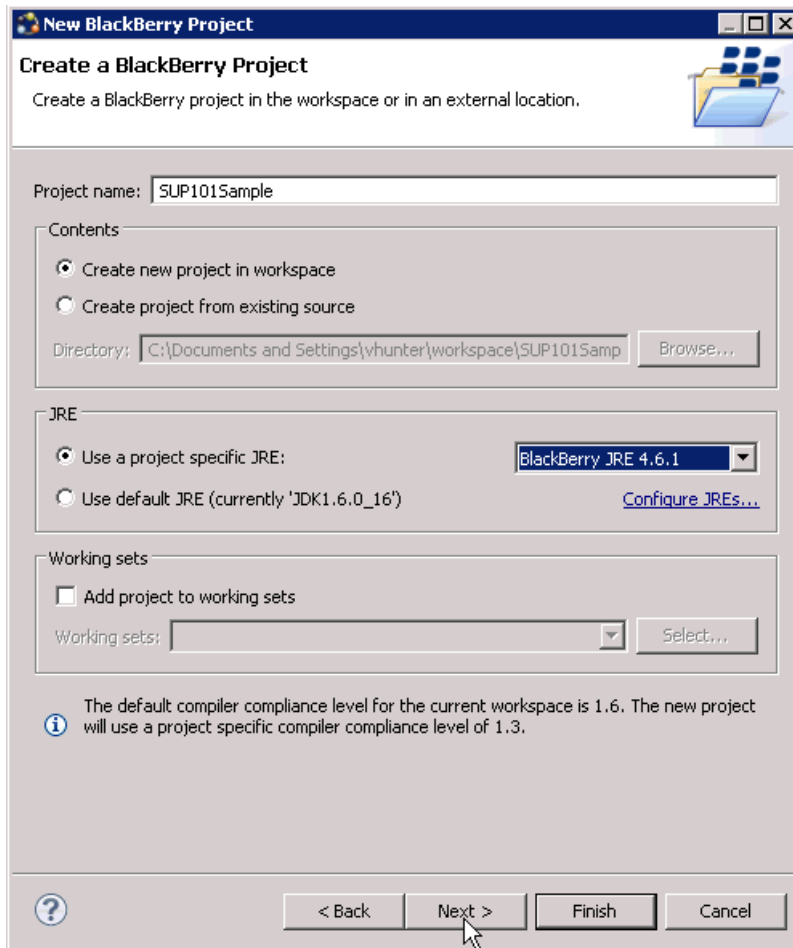
Goal: Create a new BlackBerry project.

Prerequisites

Installing the BlackBerry Java Plug-in for Eclipse on page 13

Task

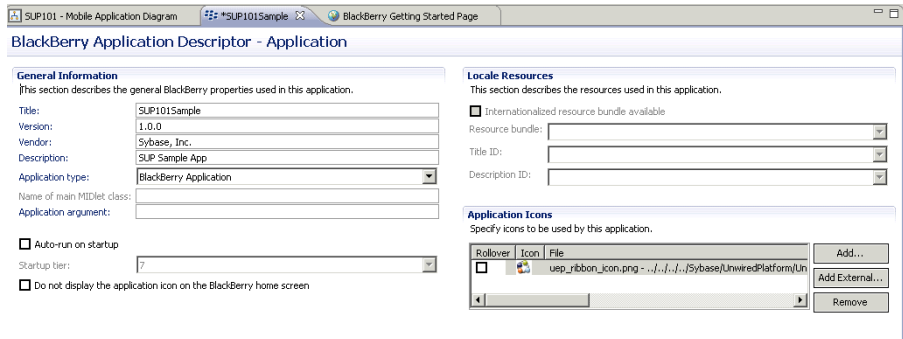
1. From the Eclipse menu, select **New > Other > BlackBerry > BlackBerry Project**.
2. In the New BlackBerry Project wizard, use these values and click **Next**.
 - Name – enter SUP101Sample
 - Use a project specific JRE – select **BlackBerry JRE 4.6.1**



3. In Java Settings, modify the build path to point to the correct location for `sup-client-rim.jar` and `UltraLiteJ.jar`.
 - a) Click the **Libraries** tab.
 - b) Click **Add External Jars**.
 - c) Browse to `sup-client-rim.jar`, located in `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\ClientAPI\java\RIM42`, select it and click **Open**.
 - d) Click **Add External Jars**.
 - e) Browse to `UltraLiteJ.jar` located in `<UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\ClientAPI\UltraliteJ\BlackBerry4.2`, select it and click **Open**.
 - f) Click **Finish**.

4. In the SUP101Sample descriptor tab:

- a) In Title, enter SUP101Sample.
- b) In Application Icons, click **Add External** and browse to uep_ribbon_icon.png, located in <UnwiredPlatform_InstallDir>\UnwiredPlatform\Unwired_WorkSpace\Eclipse\sybase_workspace\mobile\ eclipse\plugins\com.sybase.uep.bob.rim.1.5.2.<version>\generate\blackberry\build-4.6.1\images\uep_ribbon_icon.png, and click **Open** to add the image.



5. Save the SUP101Sample descriptor file.

6. Copy these files to the BlackBerry Simulator directory that is located in the Eclipse plugins' corresponding component package folder, for example, <UnwiredPlatform_InstallDir>\UnwiredPlatform\Eclipse\plugins\net.rim.ejde.componentpack4.6.1_4.6.1.49\components\simulator:

- Copy sup-client-rim.cod from <UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\ClientAPI\java\RIM42 to the BlackBerry Simulator directory.
- Copy UltraLiteJ.cod from <UnwiredPlatform_InstallDir>\UnwiredPlatform\Servers\UnwiredServer\ClientAPI\UltraliteJ\BlackBerry4.2 to the BlackBerry Simulator directory.

7. Copy the SUP101 folder from the SUP101 project path SUP101\Generated Code\src to SUP101Sample\src.

Configuring the Generated Customer.java File

Modify the Customer.java file so that the KeywordFilterField displays the data properly.

1. Open the Java perspective:

- a) From the main menu select **Window > Open Perspective > Other**.
- b) Select **Java** and click **OK**.

2. In Package Explorer, expand the SUP101Sample\src\SUP101 folder.
3. Open the Customer.java file, and add this code to the end of the file, after the line `/** End code region: JSON methods */`:

```
public String toString()
{
    return getFname() + " " + getLname();
}
```

4. Save the Customer.java file.

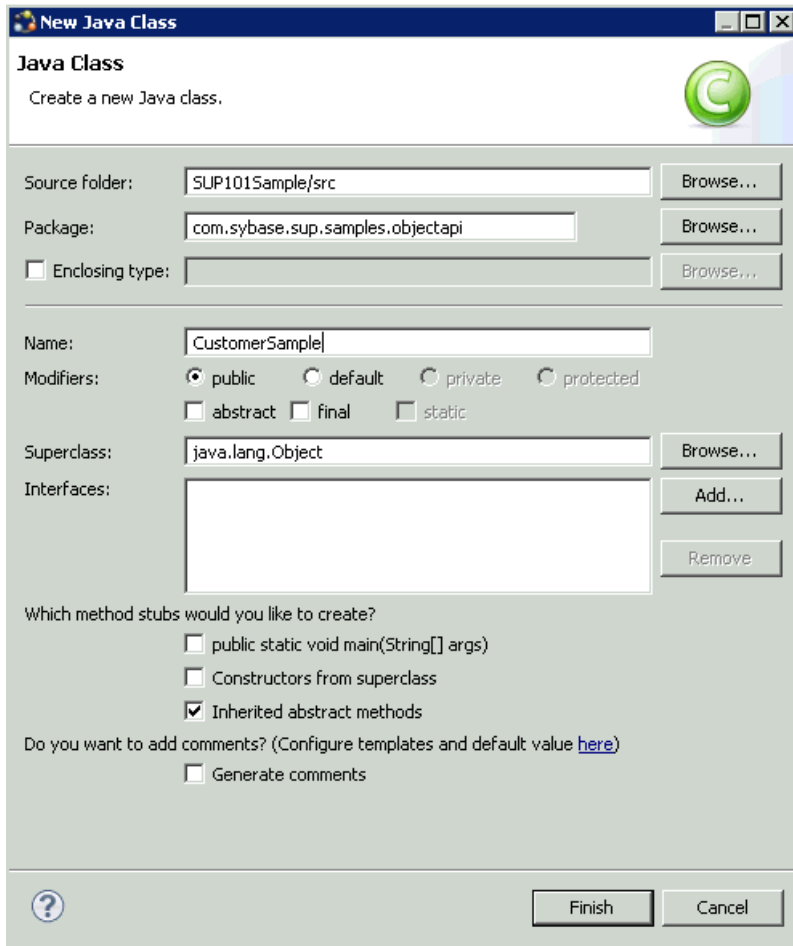
Creating the User Interface

Create the user interface for the SUP101Sample application.

Note: This procedure includes code snippets you can copy and paste for your project. Click [SUP_BB_Custom_Dev_Tutorial_code.zip](#) to access the text files that include the code snippets for the CustomerSample, CustomerList, and CustomerSampleScreen java files.

If you are viewing this guide as a PDF, you can locate the referenced document at <http://infocenter.sybase.com/>. Go to *Sybase Unwired Platform 1.5.2 > Tutorial: BlackBerry Application Development using Custom Development* to launch this PDF.

1. In the Java perspective, right-click **SUP101Sample**, and select **New > Package**.
2. In the New Java Package dialog, in Name, enter `com.sybase.sup.samples.objectapi` and click **Finish**.
3. Right-click the `com.sybase.sup.samples.objectapi` package and select **New > Class**.
4. In the New Java Class wizard, in Name, enter `CustomerSample`, and click **Finish**.



5. In the `CustomerSample.java` file, enter this code after the package `com.sybase.sup.samples.objectapi;` line:

This creates the main Customer application.

```
import java.util.Vector;

import net.rim.device.api.system.Characters;
import net.rim.device.api.ui.Graphics;
import net.rim.device.api.ui.UiApplication;
import net.rim.device.api.ui.XYRect;
import net.rim.device.api.ui.component.BasicEditField;
import net.rim.device.api.ui.component.KeywordFilterField;
import SUP101.Customer;
import SUP101.SUP101DB;

import com.sybase.collections.ObjectList;
```

```
/**
 * @author bdeng
 *
 * Customer sample main application.
 *
 * Note: For the sake of simplicity, this sample application may
not leverage
 * resource bundles and resource strings. However, it is STRONGLY
recommended
 * that application developers make use of the localization
features available
 * within the BlackBerry development platform to ensure a seamless
application
 * experience across a variety of languages and geographies. For
more information
 * on localizing your application, please refer to the BlackBerry
Java Development
 * Environment Development Guide.
 */

public final class CustomerSample extends UiApplication
{
    private KeywordFilterField _keywordFilterField;
    private CustomerList      _customerList;
    private Vector            _customers;

    // sync lock for synchronization
    private static Object     _syncLock = new Object();

    /**
     * Entry point for application.
     *
     * @param args
     *         Command line arguments (not used).
     */
    public static void main(String[] args)
    {
        // login to sync
        SUP101DB.loginToSync("supAdmin", "s3pAdmin");

        // Create a new instance of the application
        CustomerSample app = new CustomerSample();

        // Make the currently running thread the application's
event
        // dispatch thread and begin processing events.
        app.enterEventDispatcher();
    }

    /**
     * Constructor
     */
    public CustomerSample()
    {
        // Create an instance of KeywordFilterField class.
    }
}
```

```

        _keywordFilterField = new KeywordFilterField();

        // Refresh the data
        refreshData();

        // Add our list to a KeywordFilterField object.
        _keywordFilterField.setSourceList(_customerList,
        _customerList);

        // We're providing a customized edit field for
        // the KeywordFilterField.
        CustomKeywordField customSearchField = new
CustomKeywordField();
        _keywordFilterField.setKeywordField(customSearchField);

        // Create main screen.
        CustomerSampleScreen screen = new
CustomerSampleScreen(this);

        // We need to explicitly add the search/title field via
        // mainScreen.setTitle().
        screen.setTitle(_keywordFilterField.getKeywordField());

        // Add our KeywordFilterField to the screen and push the
screen
        // onto the stack.
        screen.add(_keywordFilterField);
        pushScreen(screen);

    }

    /**
     * Refreshes the data.
     */
    public void refreshData()
    {
        // Populate vector with data.
        if ( _customers != null )
        {
            _customers.removeAllElements();
            _customers = null;
        }

        _customers = loadData();

        // Create an instance of our SortedReadableList class.
        if ( _customerList == null )
        {
            _customerList = new CustomerList();
        }

        _customerList.loadFrom(_customers.elements());
    }

    /**
     * Method to access the KeywordFilterField object.

```

```

    *
    * @return This applications KeywordFilterField.
    */
    KeywordFilterField getKeywordFilterField()
    {
        return _keywordFilterField;
    }

    /**
    * Method populates and returns a vector of Customer objects
    containing data
    * read from data source.
    *
    * @return A vector containing Customer objects.
    */
    private Vector loadData()
    {
        ObjectList customers = Customer.findAll();
        Vector customerVec = new Vector(customers.size());

        int size = customers.count();
        for (int i = 0; i < size; i++)
        {
            customerVec.addElement(customers.elementAt(i));
        }

        return customerVec;
    }

    /**
    * Returns the sync lock object
    *
    * @return Object
    */
    public Object getSyncLock()
    {
        return _syncLock;
    }

    /**
    * Inner Class: A custom keyword input field for the
    KeywordFilterField. We
    * want to prevent a save dialog from being presented to the
    user when
    * exiting the application as the ability to persist data is
    not relevant to
    * this application. We are also using the paint() method to
    customize the
    * appearance of the cursor in the input field.
    */
    final static class CustomKeywordField extends BasicEditField
    {
        // Constructor
        CustomKeywordField()
        {
            // Custom style.

```

```

        super(USE_ALL_WIDTH | NON_FOCUSABLE | NO_LEARNING |
NO_NEWLINE);

        setLabel("Search: ");
    }

    /**
     * Intercepts ESCAPE key.
     *
     * @see
net.rim.device.api.ui.component.TextField#keyChar(char,int,int)
     */
    protected boolean keyChar(char ch, int status, int time)
    {
        switch (ch)
        {
            case Characters.ESCAPE:
                // Clear keyword.
                if ( super.getTextLength() > 0 )
                {
                    setText("");
                    return true;
                }
        }
        return super.keyChar(ch, status, time);
    }

    /**
     * Overriding super to add custom painting to our class.
     *
     * @see net.rim.device.api.ui.Field#paint(Graphics)
     */
    protected void paint(Graphics graphics)
    {
        super.paint(graphics);

        // Draw caret.
        getFocusRect(new XYRect());
        drawFocus(graphics, true);
    }
}

```

6. Save the `CustomerSample.java` file.
7. In the Java perspective, right-click the `com.sybase.sup.samples.objectapi` package, and select **New > Class**.
8. In the New Java Class wizard, in **Name**, enter `CustomerList`, and click **Finish**.
9. In the `CustomerList.java` file, enter this code after the package `com.sybase.sup.samples.objectapi` line:

```

import net.rim.device.api.collection.util.SortedReadableList;
import net.rim.device.api.ui.component.KeywordProvider;
import net.rim.device.api.util.Comparator;

```

```

import net.rim.device.api.util.StringUtilities;
import SUP101.Customer;

public class CustomerList extends SortedReadableList
    implements KeywordProvider
{
    /**
     * Creates a customer list based on a Vector of customers.
     */
    public CustomerList()
    {
        super(new CustomerListComparator());
    }

    /**
     * @see
net.rim.device.api.ui.component.KeywordProvider#getKeywords(Object
    element)
     */
    public String[] getKeywords(Object element)
    {
        if ( element instanceof Customer )
        {
            Customer customer = (Customer) element;
            return
StringUtilities.stringToWords(customer.getFname() + " " +
customer.getLname());
        }
        return null;
    }

    /**
     * A Comparator class used for sorting our Customer objects by
name.
     */
    final static class CustomerListComparator
        implements Comparator
    {
        /**
         * Compares two customers by comparing their names' in
alphabetical
         * order.
         *
         * @see net.rim.device.api.util.Comparator#compare(Object,
Object)
         */
        public int compare(Object o1, Object o2)
        {
            if ( o1 == null || o2 == null ) throw new
IllegalArgumentException("Cannot compare null customers");

            return o1.toString().compareTo(o2.toString());
        }
    }
}

```

10. Save the `CustomerList.java` file.
11. In the Java perspective, right-click the `com.sybase.sup.samples.objectapi` package, and select **New > Class**.
12. In the New Java Class wizard, in **Name**, enter `CustomerSampleScreen`, and click **Finish**.
13. In the `CustomerSampleScreen.java` file, enter this code after the package `com.sybase.sup.samples.objectapi`; line:

```
import net.rim.device.api.system.Characters;
import net.rim.device.api.ui.Field;
import net.rim.device.api.ui.Font;
import net.rim.device.api.ui.Graphics;
import net.rim.device.api.ui.MenuItem;
import net.rim.device.api.ui.UiApplication;
import net.rim.device.api.ui.component.BasicEditField;
import net.rim.device.api.ui.component.ButtonField;
import net.rim.device.api.ui.component.GaugeField;
import net.rim.device.api.ui.component.KeywordFilterField;
import net.rim.device.api.ui.container.HorizontalFieldManager;
import net.rim.device.api.ui.container.MainScreen;
import SUP101.Customer;
import SUP101.SUP101DB;

import com.sybase.persistence.ObjectSyncStatusData;
import com.sybase.persistence.SyncStatusListener;
import com.sybase.persistence.SyncStatusState;

/**
 * @author bdeng
 *
 * This class represents the main screen for the Customer Sample
 * application.
 *
 * Note: For the sake of simplicity, this sample application may
 * not leverage
 * resource bundles and resource strings. However, it is STRONGLY
 * recommended
 * that application developers make use of the localization
 * features available
 * within the BlackBerry development platform to ensure a seamless
 * application
 * experience across a variety of languages and geographies. For
 * more information
 * on localizing your application, please refer to the BlackBerry
 * Java Development
 * Environment Development Guide.
 */
public final class CustomerSampleScreen extends MainScreen
    implements SyncStatusListener
{
    private CustomerSample _app;
    private KeywordFilterField _keywordFilterField;
```

```

// last sync status
private int _lastSyncState = -1;
// synchronization indicator
private GaugeField _syncProgressField;
// if a synchronization is happening so that we display an
indicator
private boolean _isSynchronizing;
// whether there are pending changes
private boolean _changed;

/**
 * Creates a new CustomerSampleScreen.
 *
 * @param app
 * The UiApplication creating an instance of this
class.
 */
public CustomerSampleScreen(CustomerSample app)
{
    // A reference to the UiApplication instance for use in this
class.
    _app = app;

    // We need a reference to the UiApplication's
KeywordFilterField.
    _keywordFilterField = _app.getKeywordFilterField();

    // Add menu item to the screen's menu.
    addMenuItem(syncItem);

    if ( !SUP101DB.isSynchronized("default") )
    {
        synchronize();
    }
}

/**
 * Intercepts the ENTER key and displays info screen.
 *
 * @see net.rim.device.api.ui.Screen#keyChar(char,int,int)
 */
protected boolean keyChar(char key, int status, int time)
{
    if ( key == Characters.ENTER )
    {
        displayInfoScreen();
        return true; // We've consumed the event.
    }
    return super.keyChar(key, status, time);
}

/**
 * Handles a trackball click.
 *
 * @see net.rim.device.api.ui.Screen#invokeAction(int)
 */

```



```

public boolean invokeAction(int action)
{
    switch (action)
    {
        case ACTION_INVOKE: // Trackball click.
            displayInfoScreen();
            return true; // We've consumed the event.
    }
    return super.invokeAction(action);
}

/**
 * Creates an InfoScreen instance and pushes it onto the stack
for
 * rendering.
 */
private void displayInfoScreen()
{
    // Retrieve the selected Customer and use it to invoke a new
InfoScreen.
    Customer customer = (Customer)
_keywordFilterField.getSelectedElement();
    if ( customer != null )
    {
        InfoScreen infoScreen = new InfoScreen(this, customer);
        _app.pushScreen(infoScreen);
    }
}

/**
 * Prevent the save dialog from being displayed.
 *
 * @see
net.rim.device.api.ui.container.MainScreen#onSavePrompt()
 */
public boolean onSavePrompt()
{
    return true;
}

// Inner
classes-----
--
/**
 * Synchronizes the mbos
 */
private final MenuItem syncItem = new MenuItem("Synchronize",
0, 0)
{
    public void run()
    {
        synchronize();
    }
};

```

```

/**
 * A MainScreen class to display/update secondary information
for a selected
 * customer.
 */
private final static class InfoScreen extends MainScreen
{
    private CustomerSampleScreen _sampleScreen;
    private Customer             _customer;

    private BasicEditField      _fnameField;
    private BasicEditField      _lnameField;
    private BasicEditField      _companyField;
    private BasicEditField      _addressField;
    private BasicEditField      _stateField;
    private BasicEditField      _cityField;
    private BasicEditField      _phoneField;
    private BasicEditField      _zipField;

    /**
     * Constructs a screen to display
     *
     * @param customer
     *       The customer to display secondary information
about
     */
    InfoScreen(CustomerSampleScreen sampleScreen, Customer
customer)
    {
        _sampleScreen = sampleScreen;
        _customer = customer;

        // Set up and display UI elements.
        setTitle("Update Customer");
        _fnameField = new BasicEditField("First name: ",
customer.getFname(), BasicEditField.DEFAULT_MAXCHARS,
Field.FOCUSABLE);
        _lnameField = new BasicEditField("Last name: ",
customer.getLname(), BasicEditField.DEFAULT_MAXCHARS,
Field.FOCUSABLE);
        _companyField = new BasicEditField("Company: ",
customer.getCompany_name(),
BasicEditField.DEFAULT_MAXCHARS,
Field.FOCUSABLE);
        _addressField = new BasicEditField("Address: ",
customer.getAddress(), BasicEditField.DEFAULT_MAXCHARS,
Field.FOCUSABLE);
        _stateField = new BasicEditField("State: ",
customer.getState(), BasicEditField.DEFAULT_MAXCHARS,
Field.FOCUSABLE);
        _cityField = new BasicEditField("City: ",
customer.getCity(), BasicEditField.DEFAULT_MAXCHARS,
Field.FOCUSABLE);
        _phoneField = new BasicEditField("Phone: ",
customer.getPhone(), BasicEditField.DEFAULT_MAXCHARS,
Field.FOCUSABLE);
    }
}

```

```

        _zipField = new BasicEditField("Zip: ",
customer.getZip(), BasicEditField.DEFAULT_MAXCHARS,
Field.FOCUSABLE);
        add(_fnameField);
        add(_lnameField);
        add(_companyField);
        add(_addressField);
        add(_stateField);
        add(_cityField);
        add(_phoneField);
        add(_zipField);

        HorizontalFieldManager hfm = new
HorizontalFieldManager(Field.FIELD_HCENTER);

        ButtonField submitButton = new ButtonField("submit",
Field.FIELD_RIGHT)
        {
            protected boolean navigationClick(int status, int
time)
            {
                submit();
                return true;
            }
        };

        ButtonField cancelButton = new ButtonField("cancel",
Field.FIELD_LEFT)
        {
            protected boolean navigationClick(int status, int
time)
            {
                close();
                return true;
            }
        };

        hfm.add(submitButton);
        hfm.add(cancelButton);

        add(hfm);
    }

    protected void submit()
    {
        _customer.setFname(_fnameField.getText().trim());
        _customer.setLname(_lnameField.getText().trim());

        _customer.setCompany_name(_companyField.getText().trim());
        _customer.setAddress(_addressField.getText().trim());
        _customer.setState(_stateField.getText().trim());
        _customer.setCity(_cityField.getText().trim());
        _customer.setPhone(_phoneField.getText().trim());
        _customer.setZip(_zipField.getText().trim());
        _customer.save();
        _sampleScreen._changed = true;
    }

```

```

        _sampleScreen._keywordFilterField.updateList();
        close();
    }
}

// synchronizes the mbo
private void synchronize()
{
    Runnable runnable = new Runnable()
    {
        public void run()
        {
            synchronized (_app.getSyncLock())
            {
                setSynchronizing(true);

                try
                {
                    if ( _changed )
                    {
                        _changed = false;
                        SUP101DB.submitPendingOperations();
                    }
                }
            }
        }
    };

    SUP101DB.synchronize(CustomerSampleScreen.this);
    catch (Exception e)
    {
    }
    finally
    {
        setSynchronizing(false);
    }
}

    new Thread(runnable).start();
}

/**
 * Sets sync indicator and updates the sync status message.
 *
 * @param isSynchronizing whether synchronization is
happending
 */
public void setSynchronizing(boolean isSynchronizing)
{
    this._isSynchronizing = isSynchronizing;

    if ( this._isSynchronizing )
    {
        _syncProgressField = new GaugeField("",
SyncStatusState.SYNC_STARTING, SyncStatusState.SYNC_DONE, 0,
GaugeField.NO_TEXT |
GaugeField.LABEL_AS_PROGRESS)

```

```

        {
            protected void paint(Graphics g)
            {
                super.paint(g);
                int indicatorHeigth = 20;
                int fontSize = 12;
                int y = getHeight() - indicatorHeigth;

                // setting up font style
                Font f = Font.getDefault().derive(Font.PLAIN,
fontSize);
                g.setFont(f);
                g.setColor(0x00000000);
                g.setDrawingStyle(Graphics.DRAWSTYLE_AALINES,
true);
                // paint the synchronizing indicator as it can't
be
                // displayed in the progress bar using
                // LABEL_AS_PROGRESS style
                g.drawText(getLabel(), 1, y + (indicatorHeigth -
fontSize) / 2);
                g.setColor(0x00BEBEBE);
                g.drawRect(0, 0, getWidth(), getHeight());
            }
        };

        _app.invokeLater(new Runnable()
        {
            public void run()
            {
                setStatus(_syncProgressField);
            }
        });
    }
    else
    {
        _app.invokeLater(new Runnable()
        {
            public void run()
            {
                _syncProgressField = null;
                setStatus(null);
            }
        });

        _app.refreshData();
    }
}

/**
 * Called when there is information to report about the status
of a
 * synchronization that is taking place.
 *
 * @param data
 * A {@link ObjectSyncStatusData} object that

```

```

contains
    *
        information about the status of the running
synchronization.
    * @return Return true to cancel the synchronization or false
to continue.
    */
    public boolean objectSyncStatus(ObjectSyncStatusData data)
    {
        if ( !_isSynchronizing )
        {
            return false;
        }

        final int syncState = data.getSyncStatusState();

        if ( _lastSyncState != syncState )
        {
            _lastSyncState = syncState;

            String syncMsg = null;
            if ( syncState == SyncStatusState.SYNC_STARTING )
            {
                syncMsg = "Connecting...";
            }
            else if ( syncState ==
SyncStatusState.APPLICATION_DATA_UPLOADING )
            {
                syncMsg = "Metadata Uploading...";
            }
            else if ( syncState ==
SyncStatusState.APPLICATION_DATA_DOWNLOADING )
            {
                syncMsg = "Metadata Downloading...";
            }
            else if ( syncState == SyncStatusState.SYNC_DONE )
            {
                syncMsg = "Completed!";
            }

            if ( syncMsg != null )
            {
                final StringBuffer strLabel = new StringBuffer(256);

                strLabel.append("Retrieving synchronization
group:").append("default").append(" - ").append(syncMsg);

                UiApplication.getUiApplication().invokeLater(new
Runnable()
                {
                    public void run()
                    {
                        _syncProgressField.setValue(syncState);

                        _syncProgressField.setLabel(strLabel.toString());
                    }
                }
            }
        }
    }

```

```

        }
    }
    return false;
}
}

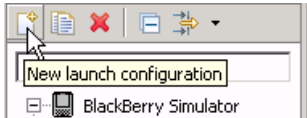
```

14. Save the `CustomerSampleScreen.java` file.

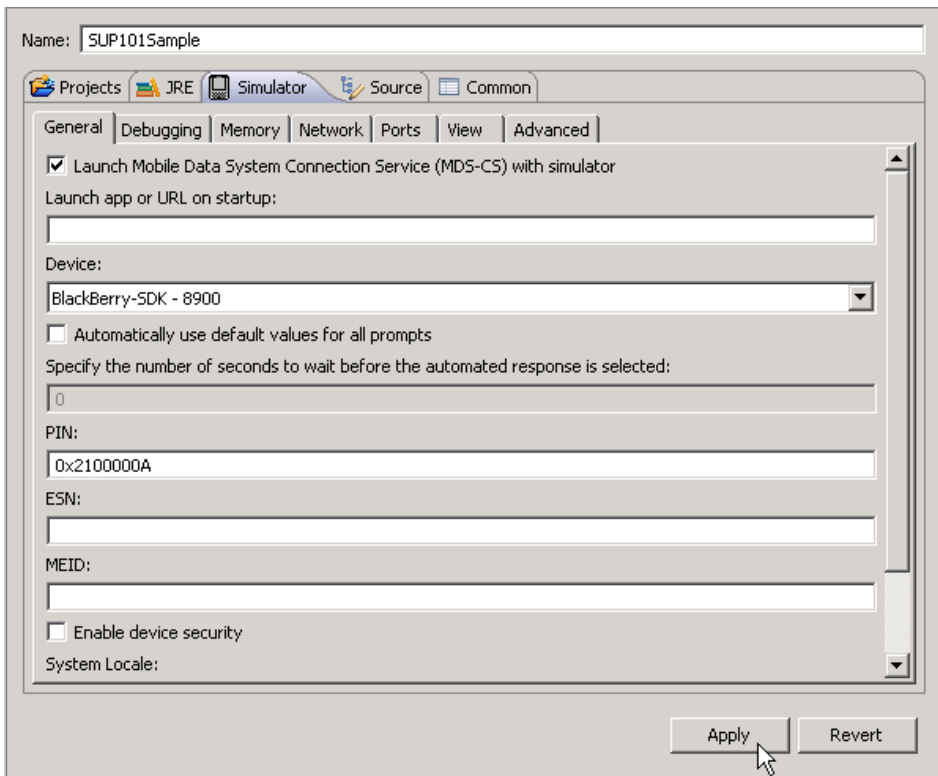
Creating a Launch Configuration for the Project

Goal: Create and configure a new launch configuration for the `SUP101Sample` project.

1. In the Java perspective, right-click the `SUP101Sample` project, and select **Run-as > Run Configurations**.
2. Select **BlackBerry Simulator** in the left pane, click the new launch configuration icon, and select the `SUP101Sample` project in the right pane.



3. In Name, enter `SUP101Sample`.
4. Click the **JRE** tab, and select the JRE to use, in this case **BlackBerry JRE 4.6.1**.
5. Click the **Simulator** tab, and select **Launch Mobile Data System Connection Service (MDS-CS) with simulator**, then select **BlackBerry-SDK - 8900** as the device.



6. Click **Apply**.
7. Click **Close**.

Testing the Device Application on the BlackBerry Simulator

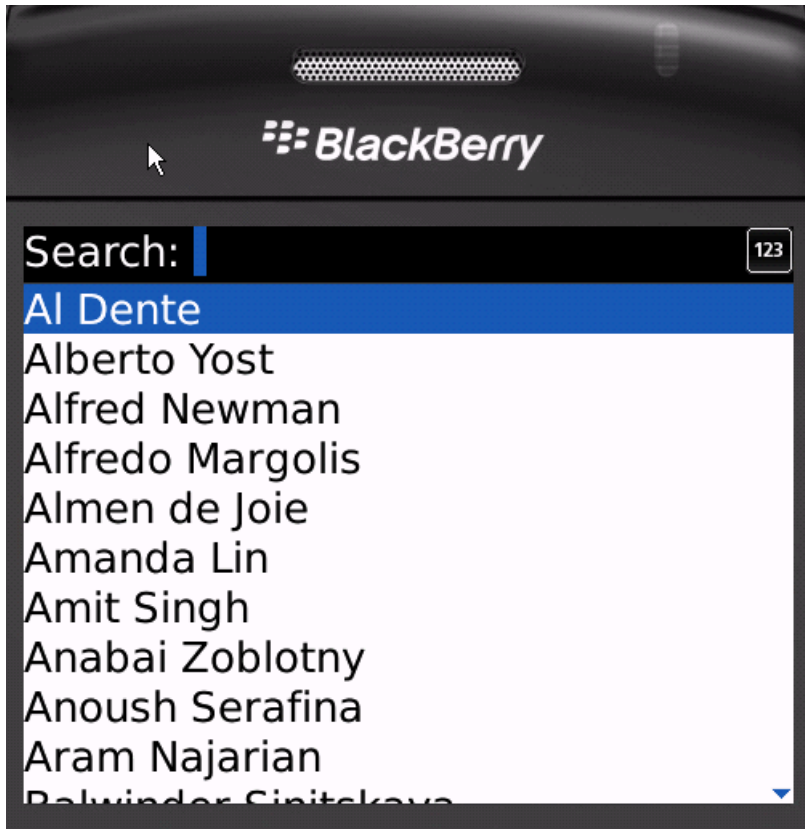
Goal: Run and test the SUP101Sample application on the BlackBerry 8900 Simulator.

This tutorial shows the device application running on the BlackBerry 8900 Simulator.

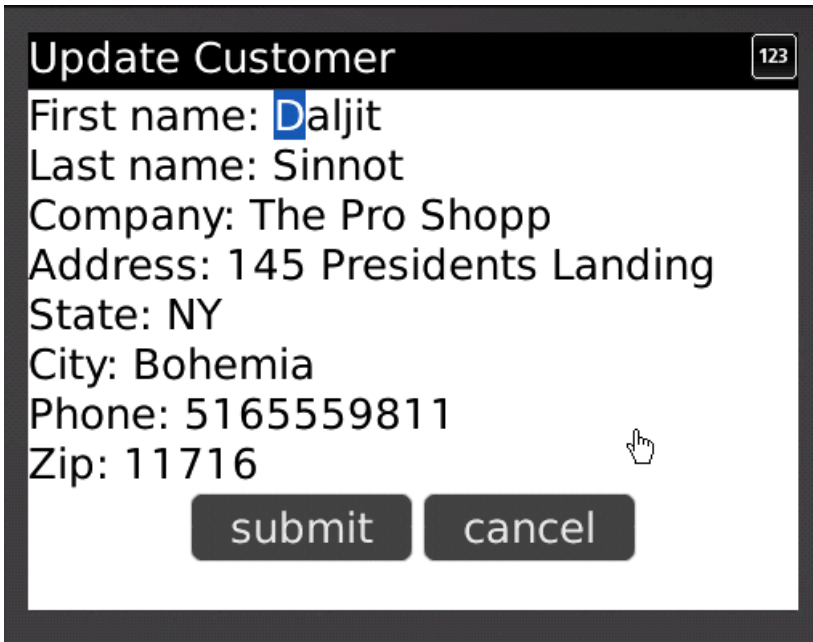
1. In the Java perspective, right-click the **SUP101Sample** project and select **Run As > BlackBerry Simulator**
2. Click the menu key to access the applications screen, and navigate to the **Downloads** folder and open it.



3. Scroll to locate the **SUP101Sample** application, and click the trackball to open it.
4. Click the menu button and select **Synchronize**.
The customer list appears.



5. Focus on the customer list and enter `dal` in the **Search** bar.
The customer list is filtered and only customers with a first or last name beginning with 'dal' are shown, in this case, Daljit Sinnot.
6. Select the customer, **Daljit Sinnot**, and click the trackball.
The detail screen for Daljit Sinnot appears.



Update Customer 123

First name: Daljit
Last name: Sinnot
Company: The Pro Shopp
Address: 145 Presidents Landing
State: NY
City: Bohemia
Phone: 5165559811
Zip: 11716

submit cancel

7. In the customer detail screen, change the first name of the customer to abc and click Submit.

The **Submit** button on the Customer Detail screen is mapped to the **update** operation of the customer mobile business object. When the application is synchronized, any pending operations are uploaded to Unwired Server.

8. To upload the new record to the back-end database `sampledb`, click the Menu key, and select **Synchronize** from the menu.

Learn More about Sybase Unwired Platform

Once you have finished, try some of the other samples or tutorials, or refer to other development documents in the Sybase Unwired Platform documentation set.

Getting Started Tutorials

Try out some of the other getting started tutorials to get a broad view of the development tools available to you.

Advanced Tutorials

Tutorials are available that demonstrate how to use some of Sybase Unwired Platform advanced features.

Check the Sybase Web site regularly for updates. Navigate to *Support > Product Documentation > Sybase Unwired Platform*, then select the most current version of the document.

Samples

Sample applications are fully developed, working applications that demonstrate the features and capabilities of Sybase Unwired Platform.

Check the Sybase Web site regularly for updates. Navigate to the Sybase Web site, then select *Products > Sybase Unwired Platform > Use tab*: <http://www.sybase.com/products/mobileenterprise/sybaseunwiredplatform?htab=USE>.

Online Help

See the online help that is installed with the product, or the Product Documentation Web site.

Check the Sybase Web site regularly for updates. Navigate to *Support > Product Documentation > Sybase Unwired Platform*, then select the most current version of the document.

Developer References

See the Developer References to learn about using the API to custom code device applications using the API.

- *Developer Reference for BlackBerry*
- *Developer Reference for iOS*
- *Developer Reference for Mobile Workflow Packages*
- *Developer Reference for Windows and Windows Mobile*

Check the Sybase Web site regularly for updates. Navigate to *Support > Product Documentation > Sybase Unwired Platform*, then select the most current version of the document.

Learn More about Sybase Unwired Platform

Javadocs are also available in the installation directory.

Programmer References

See the Programmer References to learn how to use the Administration API and Server API to extend functionality.

- *Reference: Administration APIs* – integrate your own administrative tools with Unwired Platform to monitor and manage Unwired Platform.
- *Reference: Custom Development for Unwired Server* – customize some Unwired Server features.

Check the Sybase Web site regularly for updates. Navigate to *Support > Product Documentation > Sybase Unwired Platform*, then select the most current version of the document.

Javadocs are also available in the installation directory.

Index

.cod files

- adding to the Simulator directory 17
- sup-client-rim.cod 17
- UltraLiteJ.cod 17

A

adding

- external JAR files 17

B

basics, learning 10

BlackBerry Java Plug-in for Eclipse 13

BlackBerry project, creating 17

BlackBerry Simulator 36

build path 17

C

creating

- a new Java class 20
- BlackBerry project 17
- user interface 20

Customer.java file 19

CustomerList 20

CustomerSample 20

CustomerSampleScreen 20

D

data sources 4

deploy project 14

deploying

- database mobile business objects 14
- mobile application project 14
- packages 14
- to Unwired Server 14

descriptor file 17

developer profile

- switching 4

developing

- device application 13

- device application, developing 13

developing in Eclipse 2

development environment 2

E

Eclipse Studio Edition

- Sybase Unwired WorkSpace 10

EIS

- data sources 4

Enterprise Explorer, defined 10

G

generate code 15

generating

- Object API code 15

getting started

- Sybase Unwired Platform 9
- Sybase Unwired WorkSpace 10

Getting Started tutorials

- developing a device application 13
- introduction 1
- mobility concepts 3
- overview 1

goals 7

H

help, online 10

I

installing

- Sybase Unwired Platform 9

J

JAR files

- adding 17
- sup-client-rim.jar 17
- UltraLiteJ.jar 17

Java build path 15

Java class, creating 20

Java Object API code, generating 15
Java perspective 19, 20

K

KeywordFilterField 19

L

language 15
launch configuration 35

M

MBOs
 overview 3
message-based synchronization
 factors 4
Mobile Application Diagram 14
Mobile Application Diagram, defined 10

O

Object API code
 generating 15
 Java ME for BlackBerry 15
 language 15
 platform 15
online help, accessing 10

P

page size 15
Palette, defined 10
platform 15
project build path 17
project, deploying 14
Properties view, defined 10

R

replication-based 14
replication-based synchronization

 factors 4
running the application 36

S

server domain 15
servers
 Unwired Server, starting 9
starting
 Sybase Unwired WorkSpace 10
 Unwired Server 9
sup-client-rim.cod 17
sup-client-rim.jar 17
sup.client.rim.afaria 15
SUP101Sample application
 running 36
 testing 36
 updating data 36
 viewing data 36
Sybase Unwired Platform
 getting started 9
 installing 9
Sybase Unwired WorkSpace
 getting started 10
 starting 10

T

task flow 7
testing 36

U

UltraLiteJ.cod 17
UltraLiteJ.jar 17
Unwired Server 9
 deploying to 14
 packages 14
user interface
 creating 20

W

WorkSpace Navigator, defined 10