

SYBASE®

Component Tutorials: Web Application
Development

Sybase® WorkSpace

1.5

DOCUMENT ID: DC00509-01-0150-01

LAST REVISED: June 2006

Copyright © 2005-2006 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, SYBASE (logo), ADA Workbench, Adaptable Windowing Environment, Adaptive Component Architecture, Adaptive Server, Adaptive Server Anywhere, Adaptive Server Enterprise, Adaptive Server Enterprise Monitor, Adaptive Server Enterprise Replication, Adaptive Server Everywhere, Advantage Database Server, Afaria, Answers Anywhere, Applied Meta, Applied Metacomputing, AppModeler, APT Workbench, APT-Build, APT-Edit, APT-Execute, APT-Translator, APT-Library, ASEP, Avaki, Avaki (Arrow Design), Avaki Data Grid, AvantGo, Backup Server, BayCam, Beyond Connected, Bit-Wise, BizTracker, Certified PowerBuilder Developer, Certified SYBASE Professional, Certified SYBASE Professional Logo, ClearConnect, Client-Library, Client Services, CodeBank, Column Design, ComponentPack, Connection Manager, Convoy/DM, Copernicus, CSP, Data Pipeline, Data Workbench, DataArchitect, Database Analyzer, DataExpress, DataServer, DataWindow, DataWindow .NET, DB-Library, dbQueue, Dejima, Dejima Direct, Developers Workbench, DirectConnect Anywhere, DirectConnect, Distribution Director, Dynamic Mobility Model, e-ADK, E-Anywhere, e-Biz Integrator, E-Whatever, EC Gateway, ECMAP, ECRTP, eFulfillment Accelerator, EII Plus, Electronic Case Management, Embedded SQL, EMS, Enterprise Application Studio, Enterprise Client/Server, Enterprise Connect, Enterprise Data Studio, Enterprise Manager, Enterprise Portal (logo), Enterprise SQL Server Manager, Enterprise Work Architecture, Enterprise Work Designer, Enterprise Work Modeler, eProcurement Accelerator, eremote, Everything Works Better When Everything Works Together, EWA, ExtendedAssist, Extended Systems, ExtendedView, Financial Fusion, Financial Fusion (and design), Financial Fusion Server, Formula One, Fusion Powered e-Finance, Fusion Powered Financial Destinations, Fusion Powered STP, Gateway Manager, GeoPoint, GlobalFIX, iAnywhere, iAnywhere Solutions, ImpactNow, Industry Warehouse Studio, InfoMaker, Information Anywhere, Information Everywhere, InformationConnect, InstaHelp, Intelligent Self-Care, InternetBuilder, iremote, irLite, iScript, Jaguar CTS, jConnect for JDBC, KnowledgeBase, Legion, Logical Memory Manager, M2M Anywhere, Mach Desktop, Mail Anywhere Studio, Mainframe Connect, Maintenance Express, Manage Anywhere Studio, MAP, M-Business Anywhere, M-Business Channel, M-Business Network, M-Business Suite, MDI Access Server, MDI Database Gateway, media.splash, Message Anywhere Server, MetaWorks, MethodSet, mFolio, Mirror Activator, ML Query, MobiCATS, MobileQ, MySupport, Net-Gateway, Net-Library, New Era of Networks, Next Generation Learning, Next Generation Learning Studio, O DEVICE, OASIS, OASIS logo, ObjectConnect, ObjectCycle, OmniConnect, OmniQ, OmniSQL Access Module, OmniSQL Toolkit, OneBridge, Open Biz, Open Business Interchange, Open Client, Open ClientConnect, Open Client/Server, Open Client/Server Interfaces, Open Gateway, Open Server, Open ServerConnect, Open Solutions, Optima++, Partnerships that Work, PB-Gen, PC APT Execute, PC DB-Net, PC Net Library, Pharma Anywhere, PhysicalArchitect, Pocket PowerBuilder, PocketBuilder, Power++, Power Through Knowledge, power.stop, PowerAMC, PowerBuilder, PowerBuilder Foundation Class Library, PowerDesigner, PowerDimensions, PowerDynamo, Powering the New Economy, PowerScript, PowerSite, PowerSocket, Powersoft, PowerStage, PowerStudio, PowerTips, Powersoft Portfolio, Powersoft Professional, PowerWare Desktop, PowerWare Enterprise, ProcessAnalyst, Pylon, Pylon Anywhere, Pylon Application Server, Pylon Conduit, Pylon PIM Server, Pylon Pro, QAnywhere, Rapport, Relational Beans, RemoteWare, RepConnector, Report Workbench, Report-Execute, Replication Agent, Replication Driver, Replication Server, Replication Server Manager, Replication Toolkit, Resource Manager, RFID Anywhere, RW-DisplayLib, RW-Library, SAFE, SAFE/PRO, Sales Anywhere, Search Anywhere, SDF, Search Anywhere, Secure SQL Server, Secure SQL Toolset, Security Guardian, ShareSpool, ShareLink, SKILS, smart.partners, smart.parts, smart.script, SOA Anywhere Trademark, SQL Advantage, SQL Anywhere, SQL Anywhere Studio, SQL Code Checker, SQL Debug, SQL Edit, SQL Edit/TPU, SQL Everywhere, SQL Modeler, SQL Remote, SQL Server, SQL Server Manager, SQL SMART, SQL Toolset, SQL Server/CFT, SQL Server/DBM, SQL Server SNMP SubAgent, SQL Station, SQLJ, Stage III Engineering, Startup.Com, STEP, SupportNow, S.W.I.F.T. Message Format Libraries, Sybase Central, Sybase Client/Server Interfaces, Sybase Development Framework, Sybase Financial Server, Sybase Gateways, Sybase IQ, Sybase Learning Connection, Sybase MPP, Sybase SQL Desktop, Sybase SQL Lifecycle, Sybase SQL Workgroup, Sybase Synergy Program, Sybase Virtual Server Architecture, Sybase User Workbench, SybaseWare, Syber Financial, SyberAssist, SybFlex, SybMD, SyBooks, System 10, System 11, System XI (logo), SystemTools, Tabular Data Stream, The Enterprise Client/Server Company, The Extensible Software Platform, The Future Is Wide Open, The Learning Connection, The Model For Client/Server Solutions, The Online Information Center, The Power of One, TotalFix, TradeForce, Transact-SQL, Translation Toolkit, Turning Imagination Into Reality, UltraLite, UltraLite.NET, UNIBOM, Unilib, Uninull, Unisep, Unistring, URK Runtime Kit for UniCode, Viafone, Viewer, VisualWriter, VQL, WarehouseArchitect, Warehouse Control Center, Warehouse Studio, Warehouse WORKS, Watcom, Watcom SQL, Watcom SQL Server, Web Deployment Kit, Web.PB, Web.SQL, WebSights, WebViewer, WorkGroup SQL Server, XA-Library, XA-Server, XcelleNet, XP Server, XTNDAccess and XTNDConnect are trademarks of Sybase, Inc. or its subsidiaries. 05/06

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

All other company and product names used herein may be trademarks or registered trademarks of their respective companies.

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

About This Book	v	
CHAPTER 1	Getting Started with Web Application Development	1
	Web Application Development environment	1
	Web Application Development perspective	2
	Debug perspective	3
	Web Application Development tools.....	3
	Installing the Web Application Development component	5
	Configuring the runtime application server.....	6
	Using the Web application project template.....	6
CHAPTER 2	Creating a Web Application Development Project.....	7
	Creating a project	7
	Lesson 1: Open Web Application Development perspective	7
	Lesson 2: Create a Web Application Development project	8
	Lesson 3: Indicate the component-support options.....	12
CHAPTER 3	Using Java Managed Beans.....	15
	Creating a Java managed bean	15
	Lesson 1: Create a Java managed bean	15
	Lesson 2: Create a Web page using the bean	23
CHAPTER 4	Using Service Managed Beans	41
	Creating a service managed bean on a Web page	41
	Lesson 1: Create a service managed bean.....	41
	Lesson 2: Create and test a Web page that uses the bean	49
CHAPTER 5	Linking Web Pages	57
	Linking Web pages.....	57
	Lesson 1: Define navigation rules	57
	Lesson 2: Test linked pages.....	61

CHAPTER 6	Using DataWindow Objects	63
	Creating a DataWindow object.....	63
	Lesson 1: Import a DataWindow library	63
	Lesson 2: Create a DataWindow object	72
	Lesson 3: Create Web pages that use DataWindow objects ..	77
	Using DataWindow advanced features	88
	Lesson 1: Add server-side events	89
	Lesson 2: Add client-side events.....	97
CHAPTER 7	Debugging Web Applications.....	101
	Debug perspective	101
	Available debugging tools	106

About This Book

Audience

This guide is intended for users who want to learn how to build service-oriented Web applications using Sybase® WorkSpace integrated development tooling.

How to use this book

This guide contains these chapters:

- Chapter 1, “Getting Started with Web Application Development” introduces Web Application Development, which utilizes the JavaServer Faces (JSF) framework to enable the development, testing, debugging, and deployment of a Web application on a runtime server.
- Chapter 2, “Creating a Web Application Development Project” describes how to use Web Application Development to create a project.
- Chapter 3, “Using Java Managed Beans” shows you how to create a Java managed bean, create a Web page that invokes it, and how to test the Web page by running it on a server.
- Chapter 4, “Using Service Managed Beans” allows you to create a Web page that invokes a SOAP service using a service managed bean.
- Chapter 5, “Linking Web Pages” describes how to link a sequence of Web pages using navigation rules and how to test the linked pages.
- Chapter 6, “Using DataWindow Objects” illustrates how to create a DataWindow® object, how to implement DataWindow objects in a Web page, and how to test the Web page by running it on a server.
- Chapter 7, “Debugging Web Applications” introduces the Debug perspective in Sybase WorkSpace and how to set breakpoints in your Web application code.

Related documents

For more information on Sybase WorkSpace:

Sybase WorkSpace online bookshelf From the Sybase WorkSpace, main menu, select **Help|Help Contents** to view the Sybase WorkSpace and supporting documentation.

The tutorial and sample files and documentation are available for download from Sybase CodeXchange. From the Sybase WorkSpace main menu, select **Help|Tutorials** for more information.

Adaptive Server Enterprise online bookshelf See Product Manuals at <http://sybooks.sybase.com/>.

Sybase WorkSpace Getting Started CD The Sybase WorkSpace Getting Started CD includes the following documents:

Sybase WorkSpace 1.5 Installation Guide

Sybase Developer Edition Servers 1.5 Installation Guide

Sybase WorkSpace 1.5 Release Bulletin

Adaptive Server Enterprise 15.0 Installation Guide

Unwired Accelerator 7.0 Installation Guide

To access the Product Manuals Web site, go to Product Manuals at <http://sybooks.sybase.com/>.

Other sources of information

Use the Sybase Getting Started CD, the SyBooks™ CD, and the Sybase Product Manuals Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.
- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.

Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.

- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

To access the Sybase Product Manuals Web site, go to Product Manuals at <http://sybooks.sybase.com/>.

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ Finding the latest information on product certifications

- 1 Point your Web browser to Technical Documents at <http://www.sybase.com/support/techdocs/>.
- 2 Click Certification Report.
- 3 In the Certification Report filter select a product, platform, and timeframe and then click Go.
- 4 Click a Certification Report title to display the report.

❖ Finding the latest information on component certifications

- 1 Point your Web browser to Availability and Certification Reports at <http://certification.sybase.com/>.
- 2 Either select the product family and product under Search by Base Product; or select the platform and product under Search by Platform.
- 3 Select Search to display the availability and certification report for the selection.

❖ Creating a personalized view of the Sybase Web site (including support pages)

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

- 1 Point your Web browser to Technical Documents at <http://www.sybase.com/support/techdocs/>.
- 2 Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

❖ Finding the latest information on EBFs and software maintenance

- 1 Point your Web browser to the Sybase Support Page at <http://www.sybase.com/support>.
- 2 Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.
- 3 Select a product.
- 4 Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the “Technical Support Contact” role to your MySybase profile.

- 5 Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Conventions

The following formatting conventions are used in this manual:

Formatting example	To indicate
command names and method names	When used in descriptive text, this font indicates keywords such as: <ul style="list-style-type: none">• Command names used in descriptive text• C++ and Java method or class names used in descriptive text• Java package names used in descriptive text
<i>myCounter</i> variable <i>server.log</i> <i>myfile.txt</i>	Italic font indicates: <ul style="list-style-type: none">• Program variables• Parts of input text that must be substituted• Directory and file names
<i>sybase\bin</i>	A backward slash (“\”) indicates cross-platform directory information. A forward slash (“/”) applies to information specific only to UNIX. Directory names appearing in text display in lowercase unless the system is case sensitive.

Formatting example	To indicate
File Save	Menu names and menu items are displayed in plain text. The pipe indicates how to navigate menu selections, such as from the File menu to the Save option.
<code>parse put get</code> <code>Name Address</code>	The vertical bar indicates: <ul style="list-style-type: none"> • Options available within code • Delimiter within message examples
<code>create table</code> <code>table created</code>	Monospace font indicates: <ul style="list-style-type: none"> • Information that you enter on a command line or as program text. • Example output fragments
Type the Name of the attribute. Click Apply .	GUI field or button name that is the recipient of a procedural action.
<code>setup -is:tempdir <full path to alternate temp directory></code>	Information that must be supplied by the user is displayed between brackets.

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.



Getting Started with Web Application Development

Sybase WorkSpace introduces a new set of tooling features for Java Web application development. Web Application Development utilizes the JavaServer Faces (JSF) framework to enable the development, testing, debugging, and deployment of a Web application on a runtime server.

Web Application Development supports the building of JavaServer Pages (JSP) for JSF and HTML applications, enabling both visual and code-oriented development. Additionally, it provides an integrated environment for building Web applications that interface with public SOAP service invocations and for incorporating DataWindow technology.

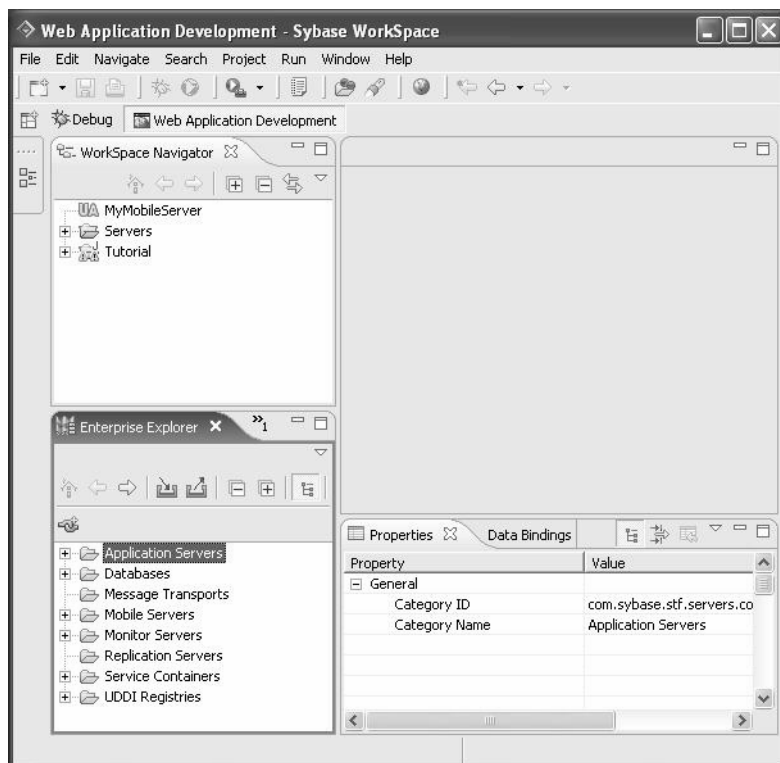
Web Application Development environment

Web Application Development enables rapid Web application development by providing a structured, integrated development environment.

The following perspectives and tools support both visual and code-oriented development.

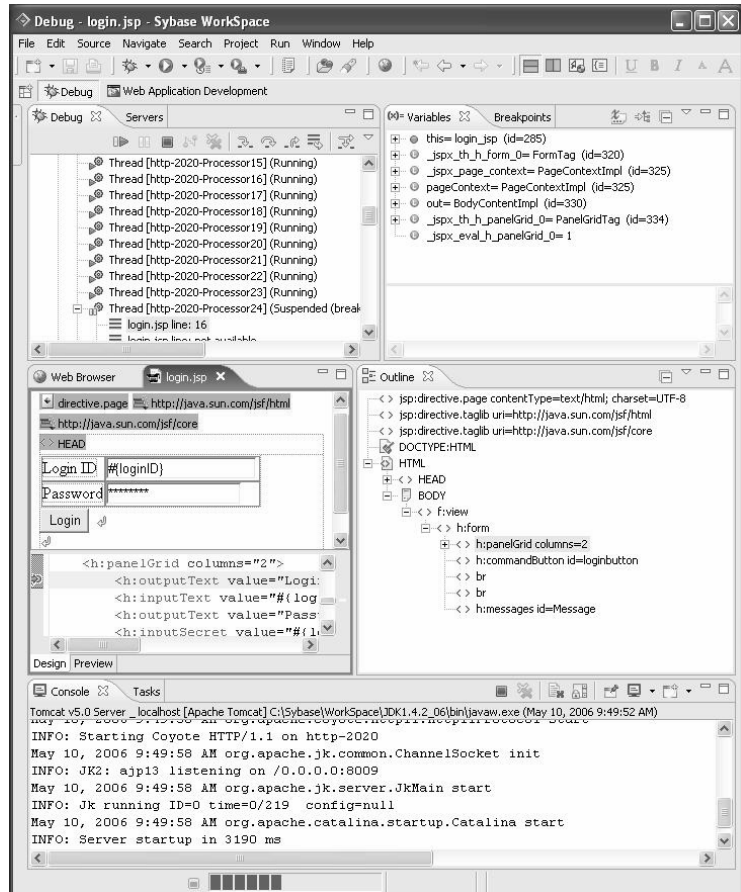
Web Application Development perspective

The Web Application Development perspective provides a set of views, wizards, and editors that help you create managed beans and other resources, design Web pages, and configure and test a Web application.



Debug perspective

The Debug perspective provides a set of views and editors that help you debug and troubleshoot a Java Web application.



Web Application Development tools

Web Application Development adds these features to Sybase WorkSpace.

Automatic code generation

Available for artifacts that invoke public SOAP services in a Web page and for JSP layouts that are based on managed beans.

DataBinding view

Value and method binding to user interface components.

Faces Configuration editor	JSF application configuration files creation and editing: <ul style="list-style-type: none">• Graphical editor to facilitate the linking of Web pages• Form-based editor to easily define and edit the parameters of the <i>faces-config.xml</i> file• Source editor to view the code base of the <i>faces-config.xml</i> file
Form-based resource bundle editor	Resource bundle message file creation and editing.
Multi-tab properties view	Web page control properties and attributes definition and editing.
Object creation wizards	Complete set of wizards to facilitate the creation of Web pages and their components.
Reuse of DataWindow libraries	Importing of existing DataWindow libraries into Sybase WorkSpace.
Testing and debugging tools	Web page validation throughout the development life cycle.
Troubleshooting tools	Assistance with troubleshooting design and development problems: <ul style="list-style-type: none">• Problems view to identify whether the Java source code in a JSP page has compilation errors.• Error markers on the source view page to identify the cause and solution for an error. Moving your mouse over an error marker displays the problem cause, and double-clicking the error marker displays possible resolutions.• Error Logs view to identify design-time errors.
Web page development	Web application development support for: <ul style="list-style-type: none">• DataWindow objects• EJB managed beans• Java managed beans• Service managed beans
Web Page editor	Graphical and code-based design and editing: <ul style="list-style-type: none">• Maximize the design canvas and source view using the resize icons in the main toolbar.• Enter and edit text directly on the design canvas.• Resize all JSF HTML controls on the design canvas.

- Manipulate all objects on the design canvas.
- Undo and redo all commands entered onto the Web Page editor using the Edit menu in the main menu bar.
- Add cascading style sheet (CSS) styles to any object by simply right-clicking the object.
- Drag and drop controls from a palette directly onto the Web Page editor.
- Drag and drop properties and methods from the Data Bindings view onto the Web Page editor to automatically generate a JSP layout and bind business logic to the controls.

XML Web file editor

Creation and editing of the *web.xml* file.

Installing the Web Application Development component

Install Sybase WorkSpace 1.5 with the Web Application Development component. If your installation of Sybase WorkSpace does not include the Web Application Development component, you must install it. See the *Sybase WorkSpace Installation Guide* on the Sybase WorkSpace Getting Started CD or the Sybase Product Manuals Web site at <http://www.sybase.com/support/techdocs>.

If you have not purchased Sybase WorkSpace but would like to perform this tutorial, you can download an evaluation version at <http://eshop.sybase.com/eshop/>. Once you log in to eShop, in the left pane, click **Development & Integration** and then **WorkSpace**.

Configuring the runtime application server

Web Application Development supports both the EAServer and Apache Tomcat server in the runtime environment. Before you can perform the Web Application Development tutorials, verify that you installed the Web Application Development features of Sybase Developer Edition Servers. See the *Sybase Developer Edition Servers Installation Guide* on the Sybase WorkSpace Getting Started CD or the Sybase Product Manuals Web site at <http://www.sybase.com/support/techdocs>.

If you have not purchased Sybase WorkSpace but would like to perform this tutorial, you can download an evaluation version of the Sybase WorkSpace, which includes the Developer Edition Servers, at <http://eshop.sybase.com/eshop/>. Once you log in to eShop, in the left pane, click **Development & Integration** and then **WorkSpace**.

Using the Web application project template

If you do not have time to complete the entire tutorial, you can quickly create a project using the Sybase Web Application Sample template. You can then view the Java files, JSP pages, and other project artifacts that compose the completed Web Application Development component tutorial.

For instructions on how to create the sample project, see “Lesson 2: Create a Web Application Development project” on page 8.

Creating a Web Application Development Project

This chapter describes how to create a project in Web Application Development. To efficiently develop and manage a Web application, you must first create a Web Application Development project and associate it with a runtime server. In this tutorial, you use the Apache Tomcat 5.0 runtime server.

Creating a project

The following lessons illustrate how to use Sybase WorkSpace to create a Web Application Development project:

- Lesson 1: Open Web Application Development perspective
- Lesson 2: Create a Web Application Development project
- Lesson 3: Indicate the component-support options

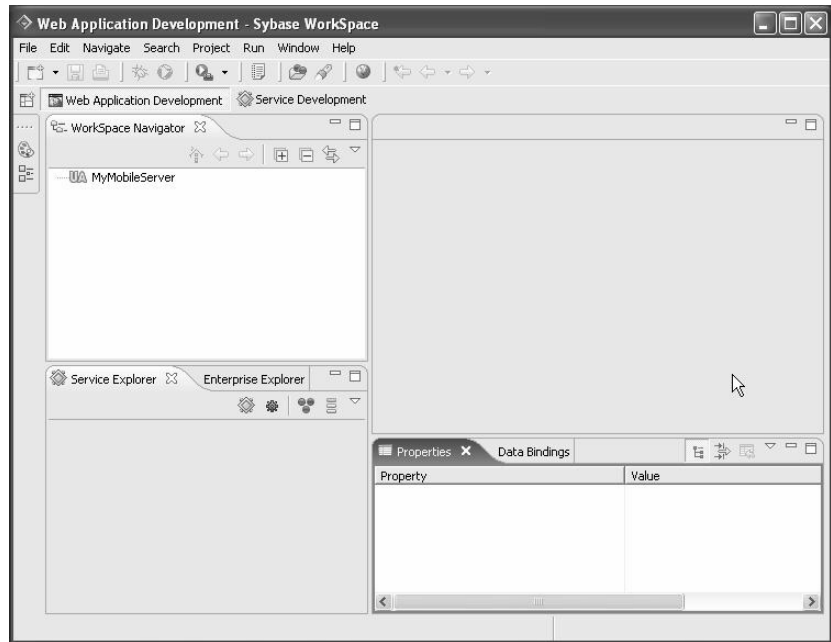
Lesson 1: Open Web Application Development perspective

First, you must launch Sybase WorkSpace and open the Web Application Development perspective.

❖ Opening the Web Application Development perspective

- 1 Select **Start|Programs|Sybase|Sybase WorkSpace|Sybase WorkSpace 1.5** to start Sybase WorkSpace.
- 2 Select **Window|Open Perspective|Web Application Development** to open the Web Application Development perspective.

The Web Application Development perspective displays.

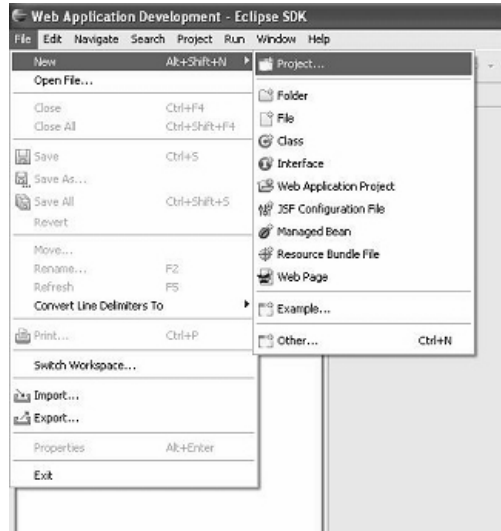


Lesson 2: Create a Web Application Development project

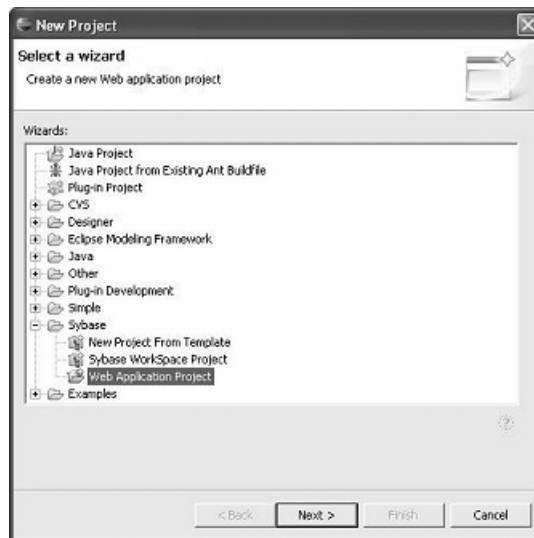
Create the Web Application Development project using the blank project template.

❖ **Creating a Web Application Development project**

- 1 Select **File|New|Project** from the Sybase WorkSpace menu bar.



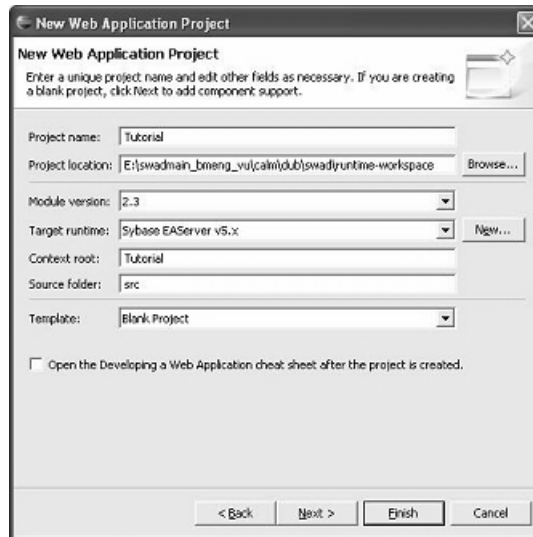
The **New Project** wizard displays.



- 2 Expand the **Sybase** folder, select **Web Application Project**, and click **Next**.

Sybase WorkSpace displays the New Web Application Project wizard.

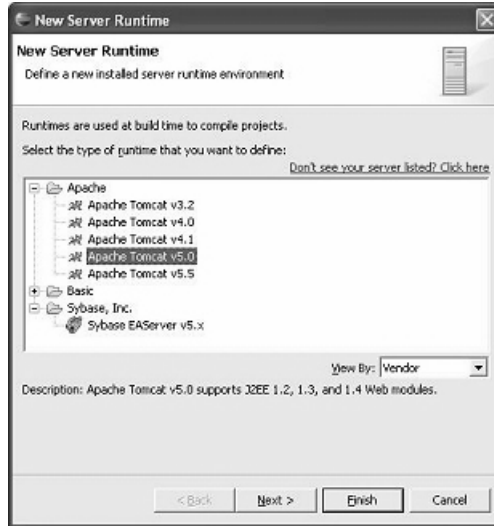
- 3 In the **Project name** field, enter `Tutorial` and click **New**.



Note If you want to see the tutorial project files and artifacts without performing the complete tutorial, select **Sybase Web Application Sample** from the **Template** drop-down list and click **Finish**. In the WorkSpace Navigator, you can now view the Java files, JSP pages, and other project artifacts that compose the completed Web Application Development component tutorial.

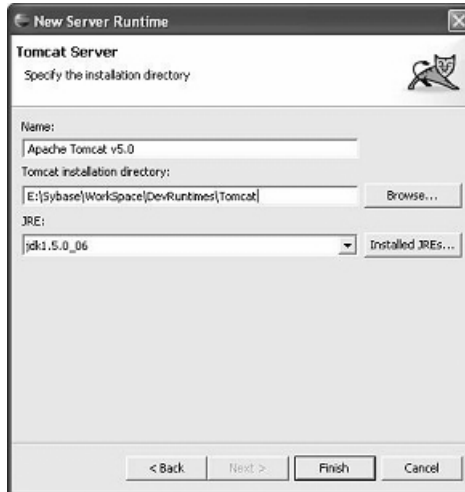
❖ **Configuring the server on which to deploy your Web application**

- 1 In the **New Server Runtime** dialog box, expand the **Apache** folder and select **Apache Tomcat v5.0**.



- 2 Click **Next**.

Sybase WorkSpace displays the **Tomcat Server** page.



- 3 In the **Tomcat installation directory** field, click **Browse** to specify the location of the Tomcat server:
<installation directory>\DevRuntimes\Tomcat
- 4 Click **Finish** to return to the **New Application Project** page.

Lesson 3: Indicate the component-support options

This tutorial demonstrates how to build a Web application using the blank project template. However, if you do not have time to complete the tutorial, you can create a project using the Sybase Web Application Sample template, which creates the Java files, JSP pages, and other project artifacts that compose the completed Web Application Development component tutorial.

❖ Indicating component-support options

- 1 In the **New Web Application Project** wizard, click **Next**.
- 2 In the **Add Component Support** page, click all check boxes to select support for all the Web Application Development components and click **Next**.

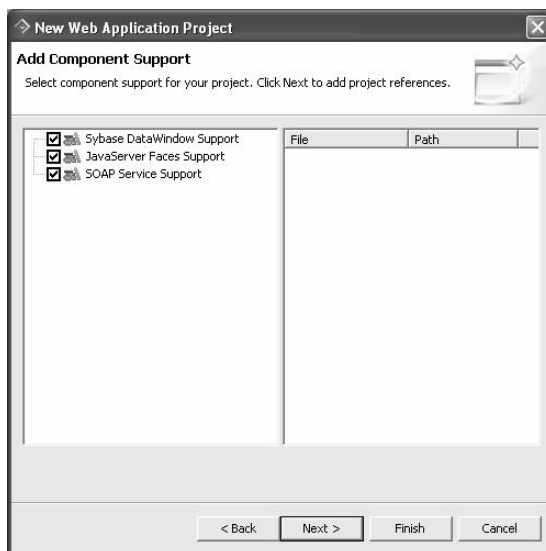
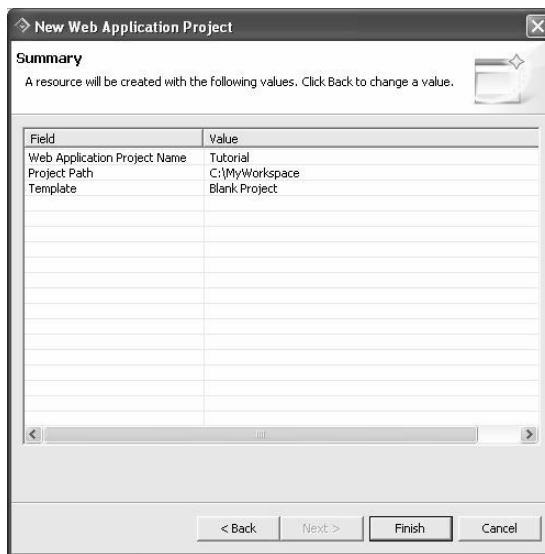


Table 2-1: Selecting Web Application Development support

Type	Description
Sybase DataWindow Support	Enables the use of DataWindow objects in Web application development.
JavaServerFaces Support	Enables the use of JSFs in Web application development.
SOAP Service Support	Enables the use of SOAP services in Web application development.

Sybase WorkSpace displays the Project Reference page.

- 3 In the **Project Reference** page, click **Next**.
- 4 Review the **Summary** page.



- 5 Click **Finish**.

The Tutorial project appears in the Workspace Navigator view. You are now ready to create a Web page with login and password fields using a Java managed bean. See Chapter 3, “Using Java Managed Beans” for step-by-step instructions.

Using Java Managed Beans

This chapter shows you how to create a Java managed bean, create a Web page that invokes it, and how to test the Web page by running it on a server.

The JavaServer Faces (JSF) framework supports the Model-View-Controller (MVC) design paradigm. It allows you to create managed beans, a reusable software component for business objects whose properties are displayed on Web pages, based on Java classes. You can use value binding to bind bean properties and business logic to the user interface components of the JSP page.

Creating a Java managed bean

The following lessons illustrate how to create a Java managed bean that creates login and password fields on a Web page.

Lesson 1: Create a Java managed bean

Lesson 2: Create a Web page using the bean

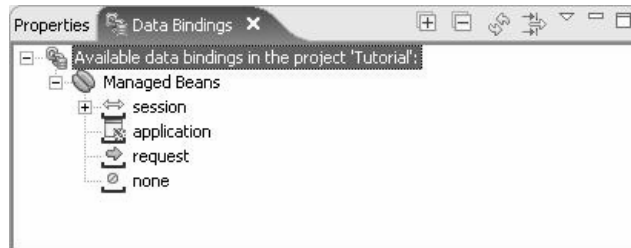
Before you can perform the steps in this tutorial, create a Web Application Development project using the blank project template as described in Chapter 2, “Creating a Web Application Development Project.”

Lesson 1: Create a Java managed bean

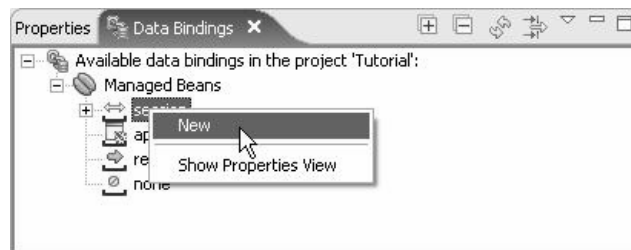
Create a Java managed bean based on a Java class.

❖ **Creating a Java managed bean**

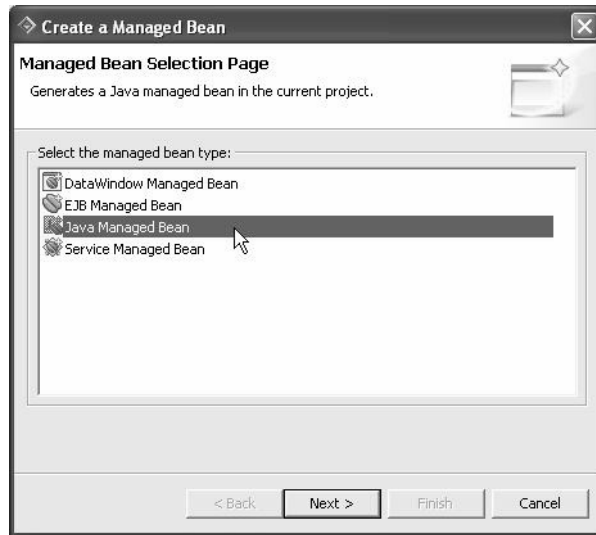
- 1 In the **Web Application Development perspective**, click the **Data Bindings** view at the bottom of the perspective window to display it.



- 2 Right-click **session** and select **New** from the context menu to open the **Create a Managed Bean** wizard.



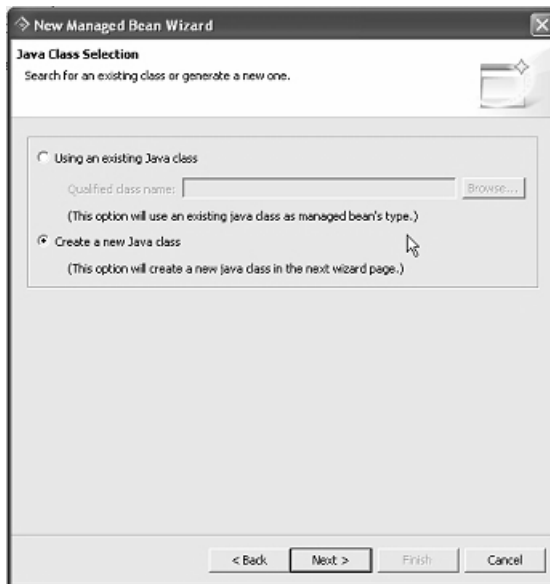
- 3 In the **Create a Managed Bean** wizard, select **Java Managed Bean** and click **Next**.



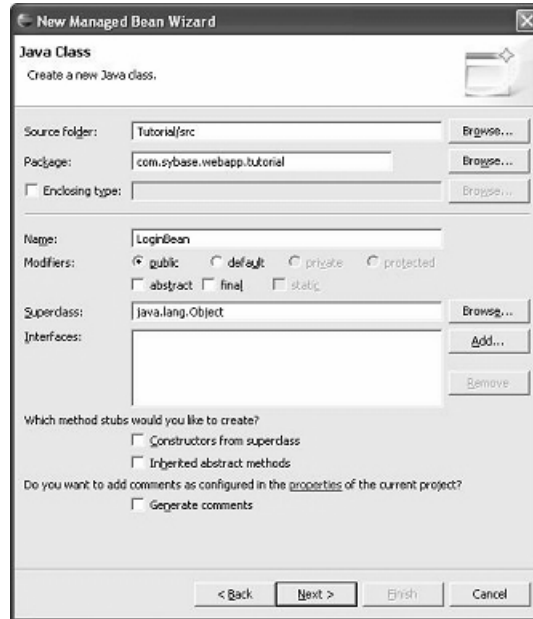
- 4 In the **Faces-Config File** page, accept the default settings and click **Next**.



- 5 In the **Java Class Selection** page, select **Create a new Java class** and click **Next**.



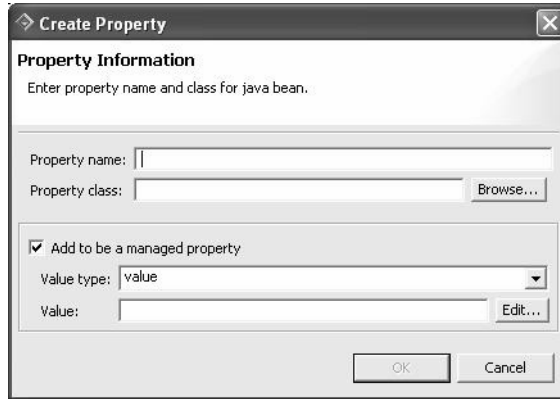
- 6 In the **Java Class** page, define the Java class:
 - In the **Package** field, enter `com.sybase.webapp.tutorial`.
 - In the **Name** field, enter `LoginBean`.



Accept the remaining default values and click **Next**. Sybase WorkSpace displays the Managed Bean Configuration page.

- 7 In the **Managed Bean Configuration** page, click **Add**.

The Create Property dialog box displays.



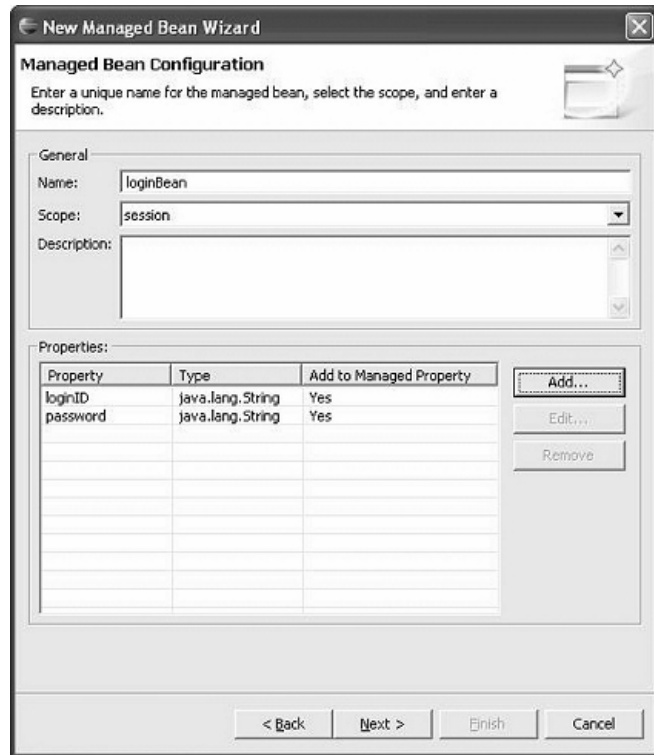
- 8 Enter the property information for the login ID:
 - In the **Property name** field, enter `loginID`.
 - In the **Property class** field, enter `java.lang.String`.
 - In the **Value** field, enter `sybase`.

The Value type field displays **value**. Keep the rest of the defaults and click **OK**.

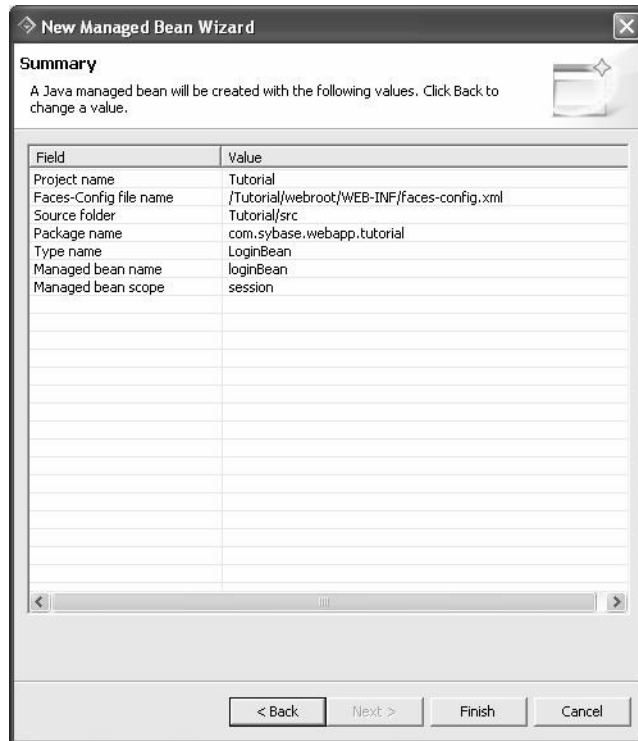
- 9 Click **Add** again to enter the property information for the password.
 - In the **Property name** field, enter `password`.
 - In the **Property class** field, enter `java.lang.String`.
 - In the **Value** field, enter `sybase`.

The Value type field displays **value**. Keep the rest of the defaults and click **OK**.

- 10 In the **New Managed Bean** wizard, click **Next**.



- 11 Review the Summary page.



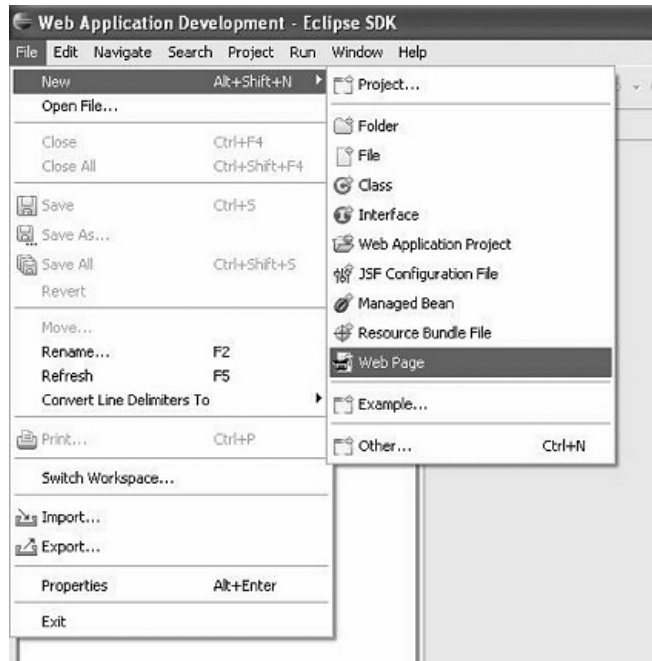
- 12 Click **Finish** to create the Java managed bean.

Lesson 2: Create a Web page using the bean

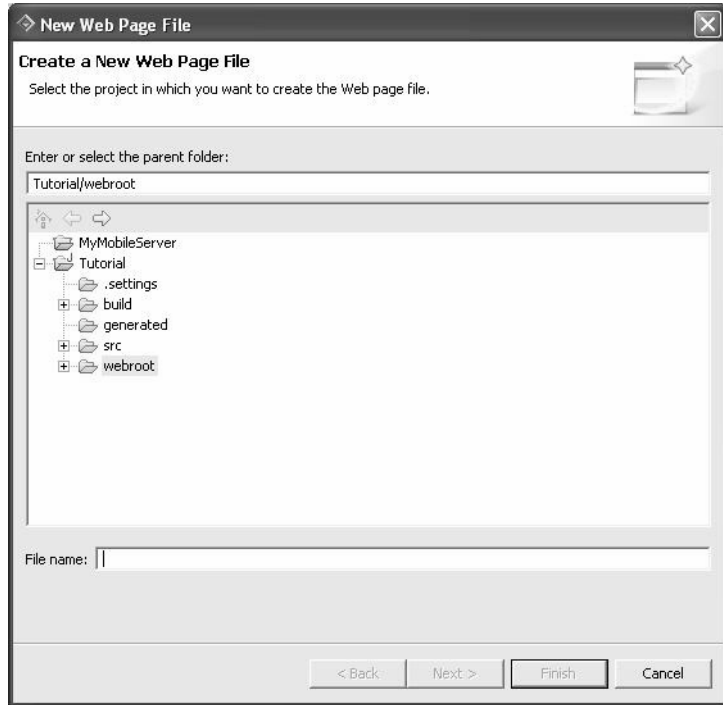
Create a Web page and bind it to the business logic in the Java managed bean you just created. Then, test the Web page by running it on the runtime server.

❖ Creating a Web page

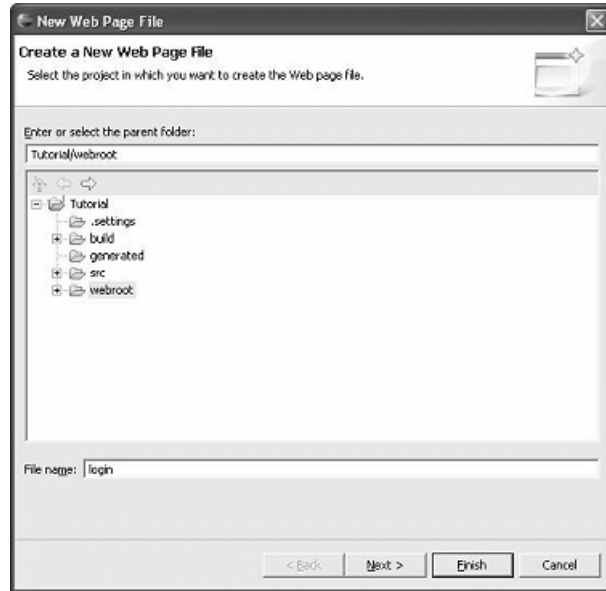
- 1 Select **File|New|Web Page** from the menu bar.



The **New Web Page File** wizard opens.



- 2 In the **File name** field, enter `login`.



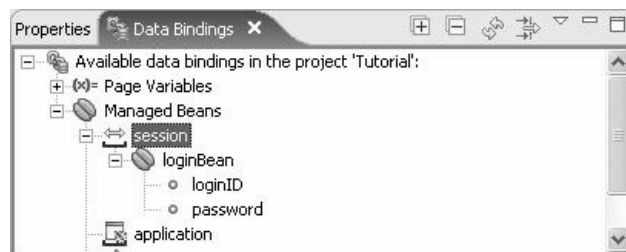
- 3 Click **Finish** to create the `login.jsp` page.

Sybase WorkSpace creates the `login.jsp` page and displays its preliminary contents in an editor.

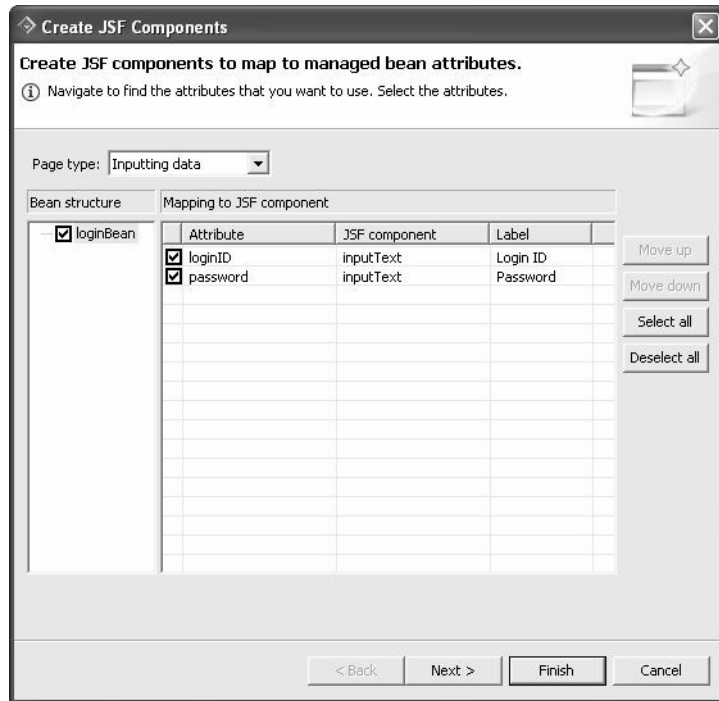
❖ **Generating the page layout**

Generate the layout of the `login.jsp` page, using the login bean that you just created.

- 1 In the **Data Binding** view, expand the tree to see the contents of **loginBean**.

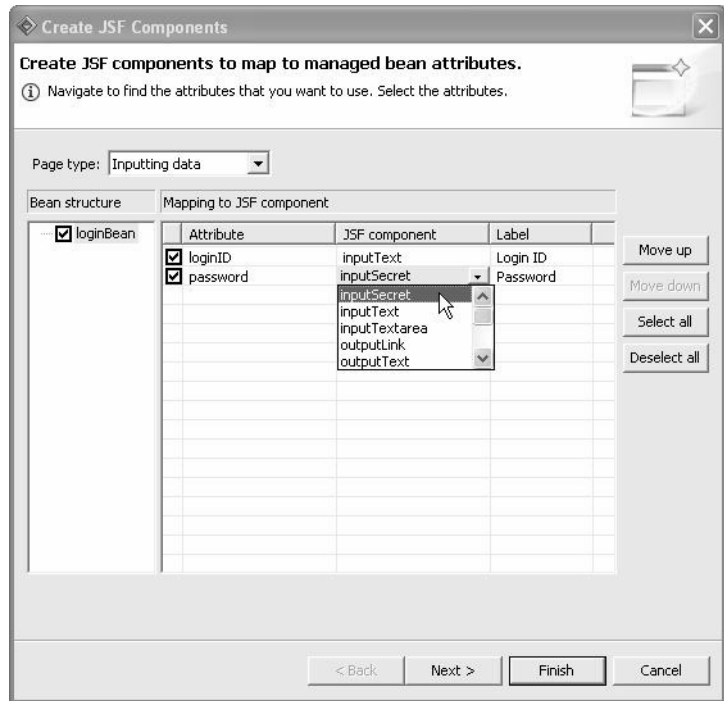


- 2 Drag and drop **loginBean** to the design pane of the *login.jsp* page to open the **Create JSF Components** wizard to create JSF components that map to the managed bean's attributes.

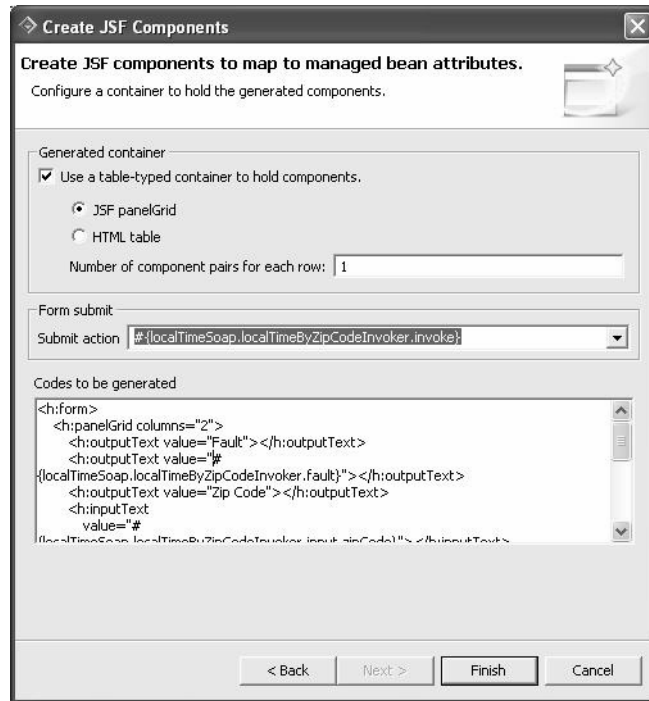


- 3 Accept the default settings for the **loginID** class member.

- To define the JSF component for the **password** class member, click the **JSF component** column in the password row, select **inputSecret** from the drop-down list, and click **Next**.



- 5 On the next page, accept the defaults and click **Finish** to generate a panel grid layout.

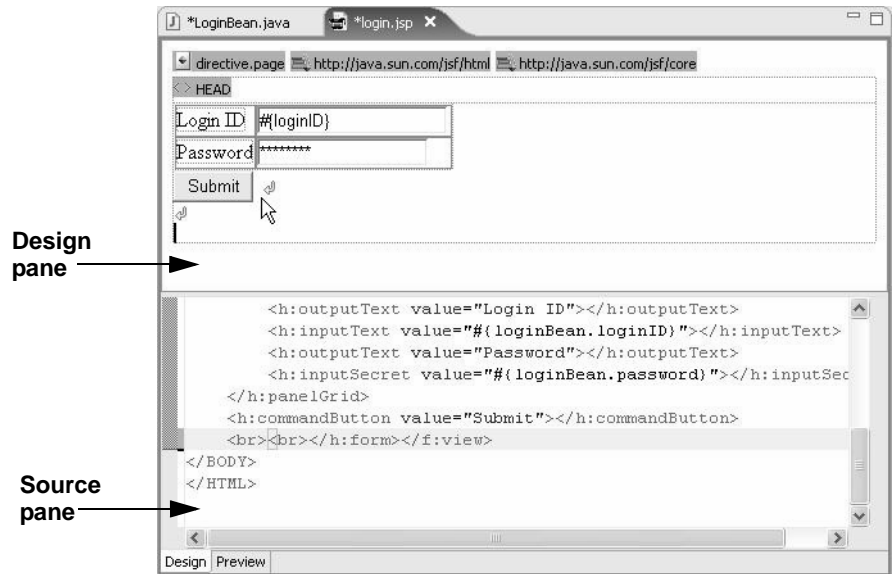


❖ **Adding a messages control**

Add a messages control to the Web page to display a message when the user inputs do not pass the validation of the business logic.

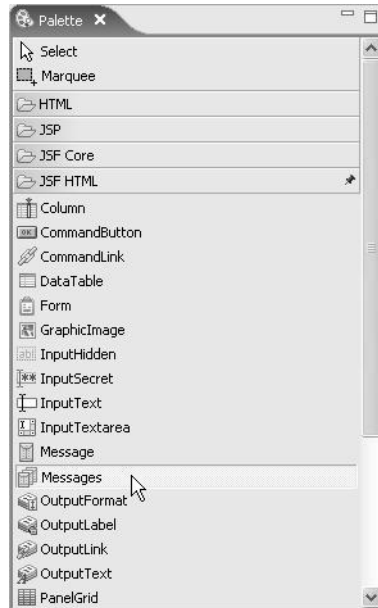
- 1 In the Web Page editor, select the **Design** tab, if necessary.
- 2 Place your cursor in the design pane at the end of the **Submit** button, and press **Enter** twice to add two hard breaks in the Web page.

The line breaks appear as `
` `
` in the Web Page editor's source pane.

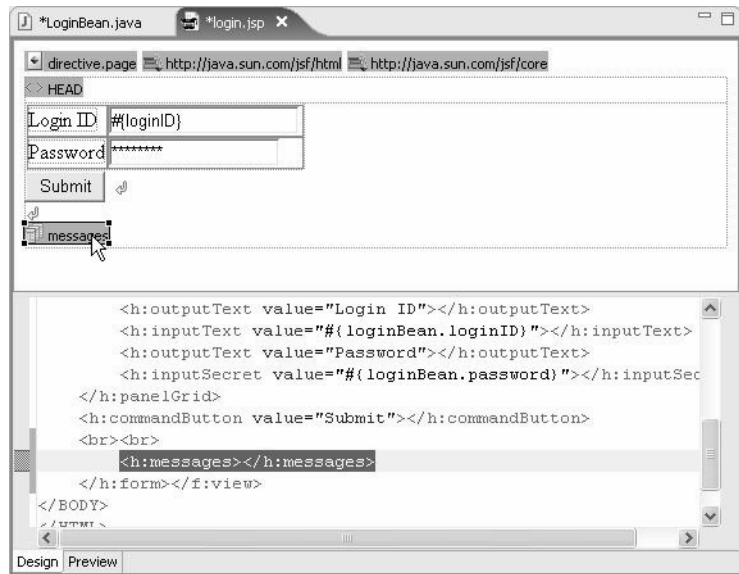


- 3 Select **Window|Show View|Palette** to display the **Palette** view.

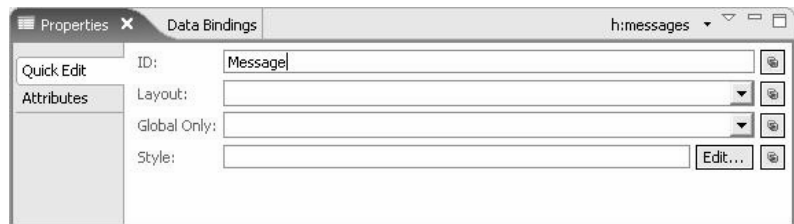
- 4 In the **Palette** view, click to expand the **JSF HTML** folder and drag and drop the **Messages** control in the design pane directly below the hard breaks.



- 5 Double-click the **Messages** control in the design pane to display the **Quick Edit** tab in the **Properties** view.



- 6 In the **ID** field, enter `Message`.



- 7 Next, modify the `loginBean` class, as described in the next section, “Editing the Java file.”

❖ Editing the Java file

Add the `validateLogin()` method to the `loginBean` class.

- 1 In the **Data Bindings** view, right-click **loginBean** and select **Open** to open the `LoginBean.java` file in the editor.
- 2 Add the following import statements under the package statement:

```
import javax.faces.application.FacesMessage;
import javax.faces.context.FacesContext;
```

Ignore the warning markers in the marker bar to the left of the Java editor. The warning messages resolve once you add the code for the `validateLogin()` method.

Note

- If you manually enter the code, the Content Assist tool prompts you with code selection. Double-click the code in the list to select it.
 - If you want to undo or redo a code entry, select Undo or Redo from the Edit menu in the menu bar.
 - If error markers appear in the marker bar to the left of the editor, hover over the marker to identify the error. It is recommended that you resolve all errors before continuing.
-

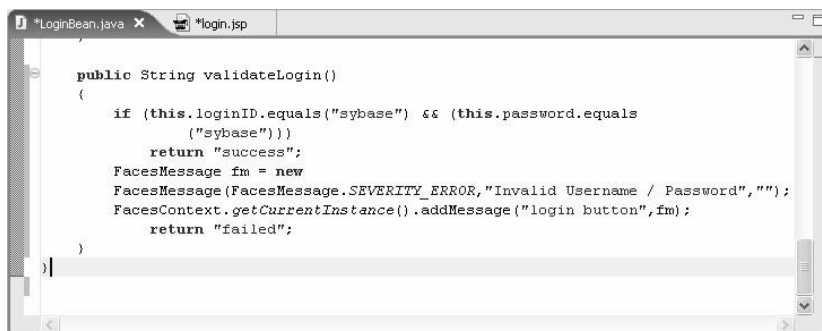
- 3 Add the `validateLogin()` method into the `loginBean` class.

```
public String validateLogin()
{
    if (this.loginID.equals("sybase") &&
        (this.password.equals("sybase")))
        return "success";

    FacesMessage fm = new
    FacesMessage (FacesMessage.SEVERITY_ERROR,
        "Invalid Username / Password", "");
    FacesContext.getCurrentInstance().addMessage
        ("login button", fm);

    return "failed";
}
```

The formatted code looks like this in the Java editor.



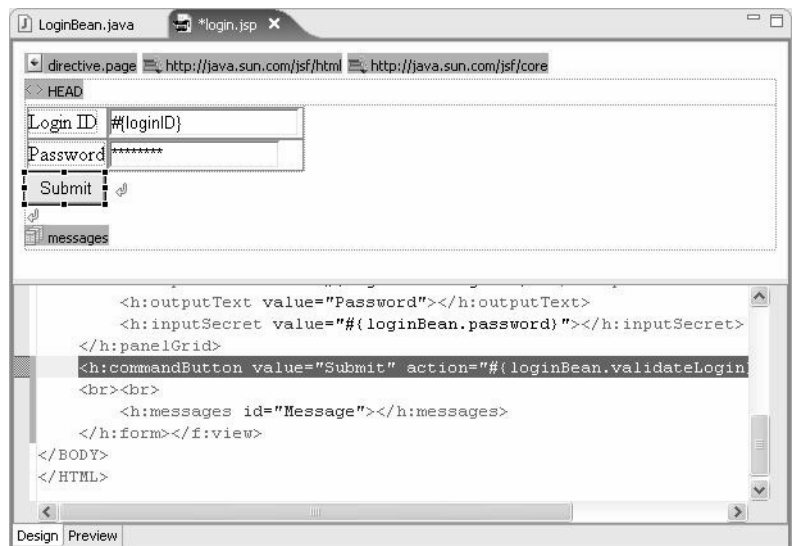
- 4 Select **File|Save** from the menu bar to save the *LoginBean.java* file. The Data Bindings view displays the `validateLogin()` method.



- 5 To associate the `validateLogin()` method with the **Submit** button, return to the *login.jsp* file.
- 6 Expand the **loginBean** session in the **Data Bindings** view and drag and drop the `validateLogin()` method onto the **Submit** button in the design pane. Sybase WorkSpace adds the following code to the JSP page.

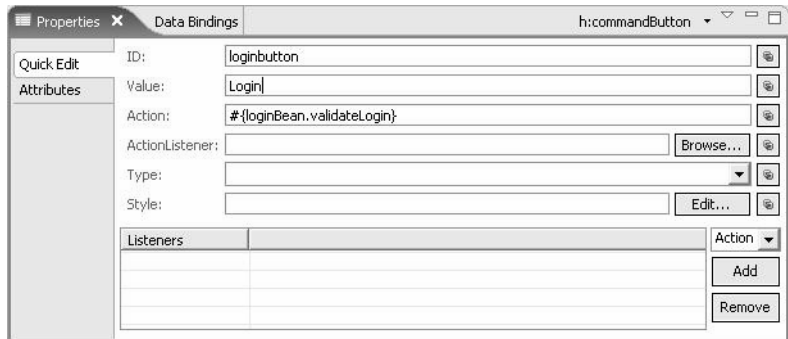
```
<h:commandButton value="Submit"
action="#{loginBean.
validateLogin}" ></h:commandButton>
```

- 7 View the following code change in the Web Page editor's source pane.

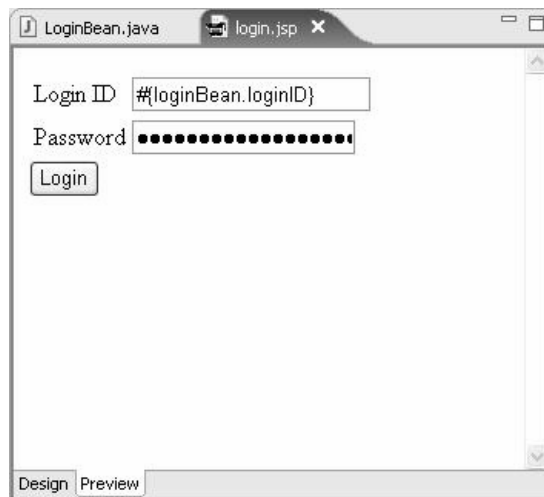


❖ **Editing the attributes for the Submit button**

- 1 Double-click the **Submit** button in the design pane to display the **Quick Edit** tab in the **Properties** view.
- 2 In the **ID** field, enter `loginbutton`.
- 3 In the **Value** field, overwrite the default value, Submit, and enter `Login`.



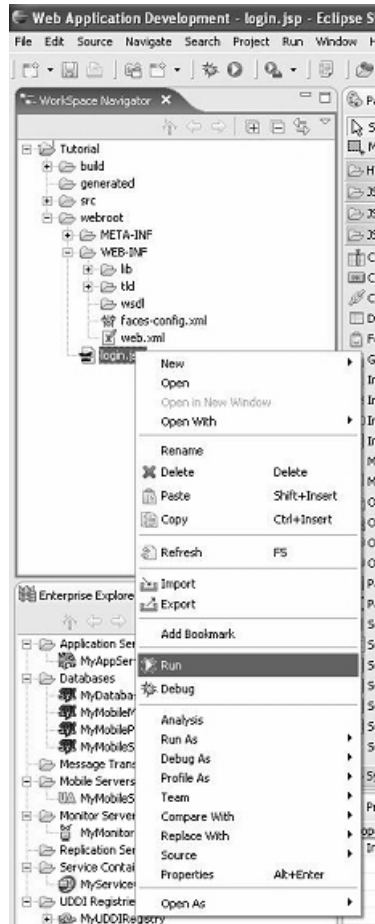
- 4 Select **File|Save** from the menu bar to save the `login.jsp` file.
- 5 Select the **Preview** tab at the bottom of the editor to preview the Web page.



❖ **Testing the login.jsp Web page**

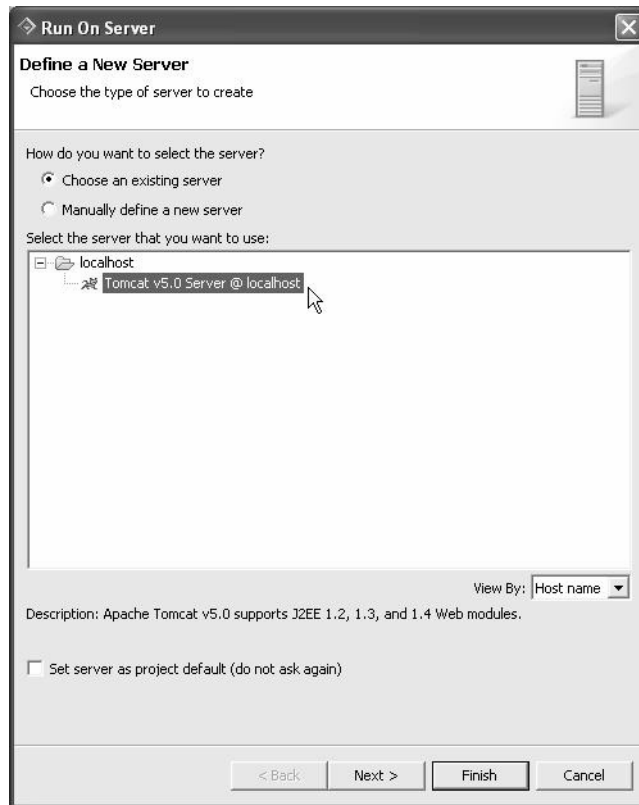
Run the Web page on the Tomcat 5.0 server.

- 1 In the WorkSpace Navigator, expand the **webroot** folder, right-click *login.jsp* and select **Run** from the context menu.



Note If you receive an error, compile the Java code. In the **WorkSpace Navigator** view, right-click **Tutorial** and select **Build** from the context menu.

- 2 In the **Run On Server** wizard, select **Choose an existing server**, and select **Tomcat v5.0 Server @ localhost** from the list.



If you do not see an existing server, you must define one. See “Starting the Tomcat 5.0 server” on page 38 for step-by-step instructions.

- 3 Click **Finish**.

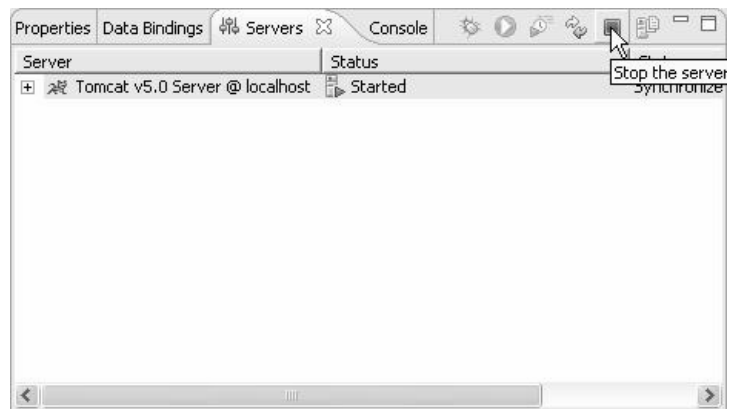
The Apache Tomcat server starts, and the JSF Page Template view displays your Web page.

- 4 To test the Web page in the browser, enter anything in the **Password** field and click **Login**.

The JSF Page Template view displays Invalid Username/Password. Once you create a service managed bean and link its associated JSF page with this one in subsequent lessons in this tutorial, the password functionality works.



- 5 Click the **Stop the server** icon in the **Servers** view to stop the server before continuing.

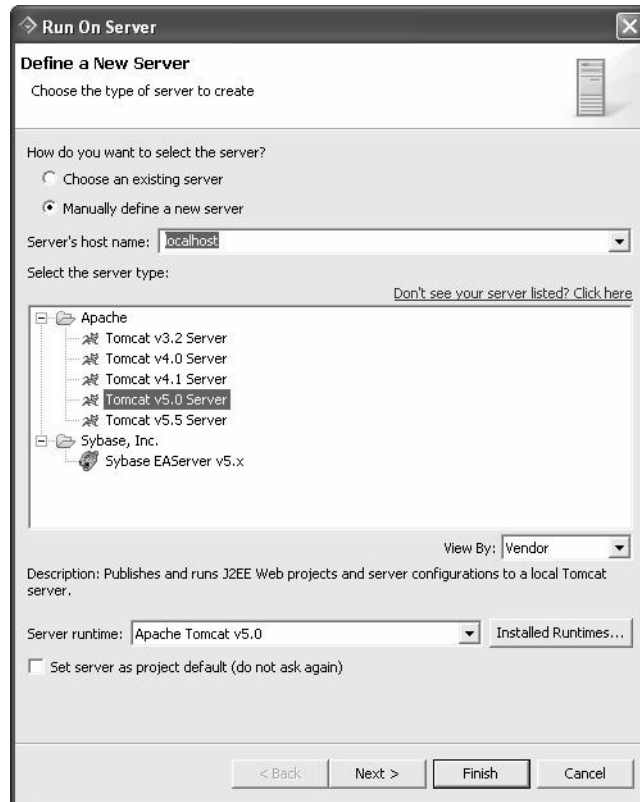


You are now ready to create a service managed bean that displays the local date and time for a specified zip code. Continue to Chapter 4, “Using Service Managed Beans.”

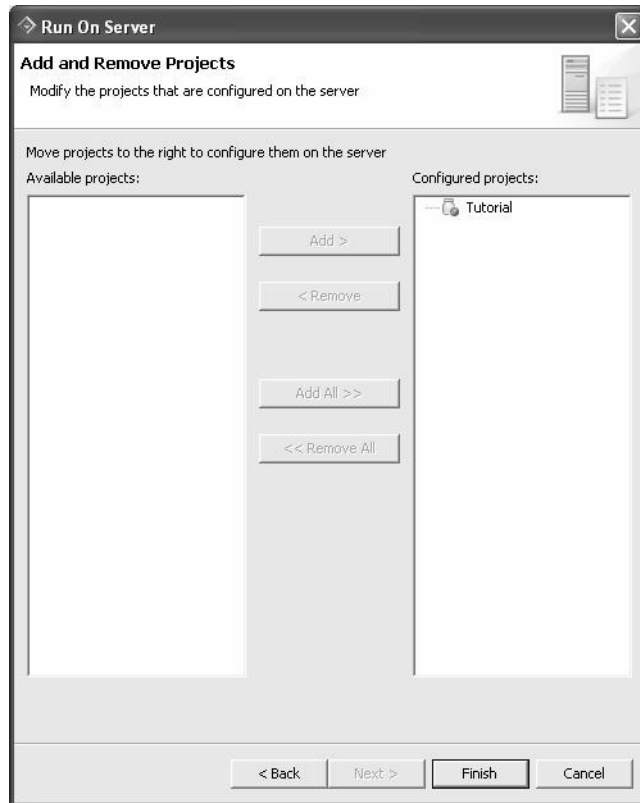
❖ **Starting the Tomcat 5.0 server**

If you try to run a JSP page and notice that you do not have a previously defined server, you can manually define and start the Tomcat 5.0 server necessary to run and test the JSP file.

- 1 In the **Run On Server** wizard, click **Manually define a new server**.
- 2 In the **Apache** folder, click **Tomcat v5.0 Server** and click **Next**.

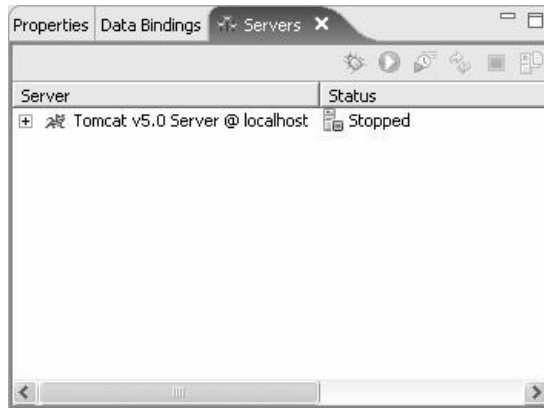


- 3 In the **Available projects** list, select **Tutorial** and click **Add** to add the project to the **Configured projects** list.



- 4 Click **Finish**.

The **Servers** view displays the server



You are now ready to run the JSP file on the server.

- To run the *login.jsp* page, see “Testing the *getTimeService.jsp* Web page” on page 53.
- To run the *getTimeService.jsp* page, see “Testing the *getTimeService.jsp* Web page” on page 53.

Using Service Managed Beans

Web Application Development allows you to create a Web page that invokes a SOAP service using a service managed bean.

This chapter describes how to create a service managed bean, how to create a Web page that invokes the service managed bean, and how to test the Web page by running it on a server.

Creating a service managed bean on a Web page

The following lessons illustrate how to create a service managed bean based on a public WSDL file. In this example, your Web page displays the local date and time for a specified zip code.

Lesson 1: Create a service managed bean

Lesson 2: Create and test a Web page that uses the bean

Before you can perform the steps in this tutorial, you must complete the tutorials in all the previous chapters in this guide.

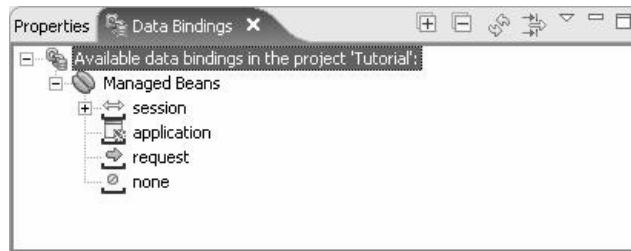
Lesson 1: Create a service managed bean

Create a service managed bean based on a public WSDL file.

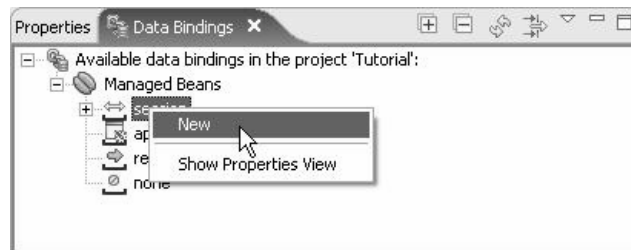
❖ **Creating a service managed bean**

- 1 If necessary, close the **JSF Page Template** view.

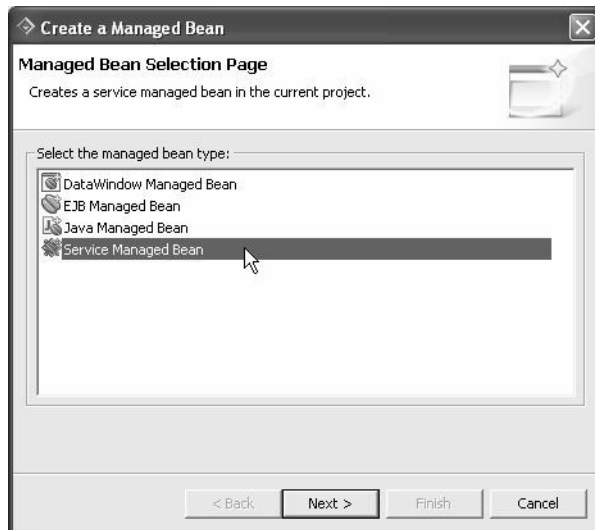
- 2 Click the **Data Bindings** view at the bottom of the perspective window to display it.



- 3 Right-click **session** and select **New** from the context menu to open the **Create a Managed Bean** wizard.



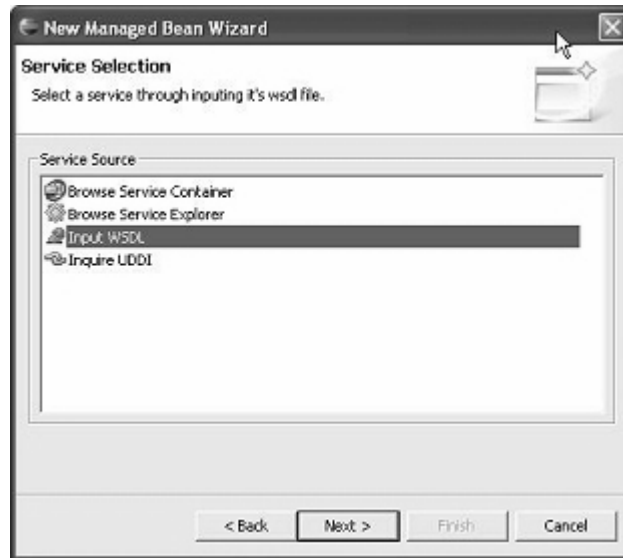
- 4 In the **Create a Managed Bean** wizard, select **Service Managed Bean** and click **Next**.



- 5 In the **Faces-Config File** page, accept the default settings and click **Next**.



- 6 Select **Input WSDL** and click **Next**.

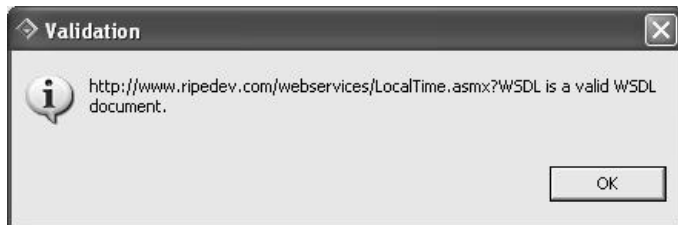


- 7 To define the WSDL file for the service managed bean, select **From URL** and enter the following in the **URL** field:

```
http://www.ripedev.com/webservices  
/LocalTime.asmx?WSDL
```



- 8 Click **Validate WSDL** to test the validity of the WSDL file.



- 9 Click **OK** to close the **Validation** box and then **Next** in the **Service Managed Bean** page to continue.
- 10 On the **Service Selection** page, define the service and port for the service managed bean.
 - In the **Service Name** field, if blank, select **LocalTime** from the drop-down list.
 - In the **Port Name** field, select **LocalTimeSoap** from the drop-down list.

Leave the **Operation Name** field blank and click Next.



The screenshot shows a dialog box titled "New Managed Bean Wizard" with a close button in the top right corner. The main heading is "Service Selection" and the instruction is "Select the service name, port name and operation name." There is a small icon of a document with a star in the top right of the instruction area. Below the instruction are three dropdown menus: "Service Name:" with "LocalTime" selected, "Port Name:" with "LocalTimeSoap" selected, and "Operation Name:" which is currently blank. Below these is a section for authorization: "If a service requires authorization, enter a user ID and password." with "User name:" and "Password:" labels and corresponding text input fields. At the bottom of the dialog are four buttons: "< Back", "Next >", "Finish", and "Cancel".

For this tutorial, the the managed bean is based on the port name; however, you can also create service managed beans based on operation names.

- 11 In the **Managed Bean Configuration** page, accept the default settings and click **Next**.



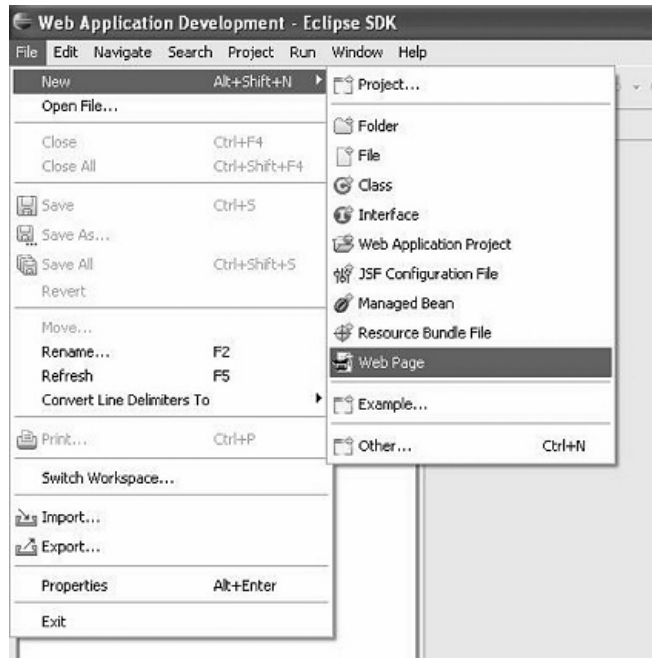
The screenshot shows a dialog box titled "New Managed Bean Wizard" with a close button in the top right corner. The main title is "Managed Bean Configuration". Below the title, there is a small icon of a document with a star and a plus sign, and a text instruction: "Enter a unique name for the managed bean, select the scope, and enter a description." The dialog is divided into a "General" section and a bottom navigation area. The "General" section contains three fields: "Name:" with the text "localTimeSoap", "Scope:" with a dropdown menu showing "session", and "Description:" with an empty text area and vertical scroll bars. The bottom navigation area contains four buttons: "< Back", "Next >", "Finish", and "Cancel".

Lesson 2: Create and test a Web page that uses the bean

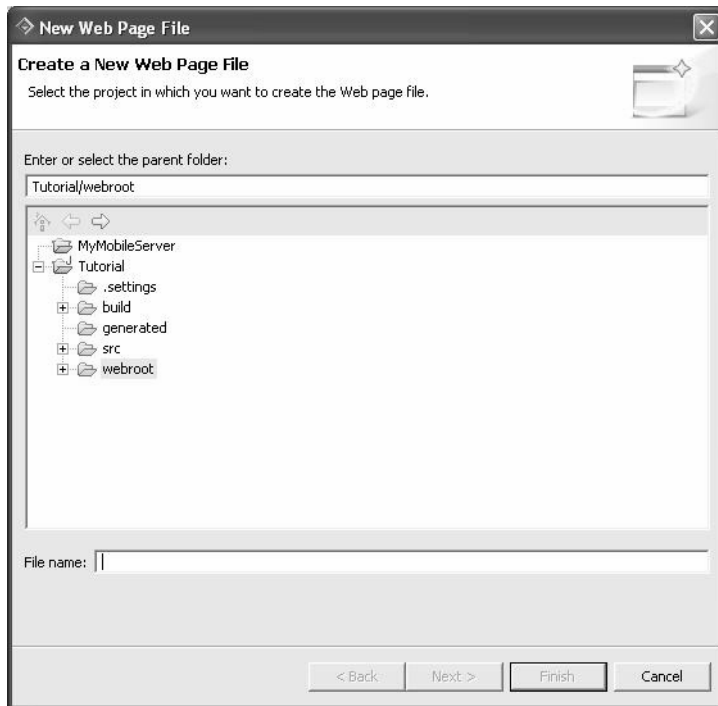
Next, create a Web page, bind it to the business logic in the service managed bean you just created, generate the Web page layout, and then test the Web page by running it on a server.

❖ Creating the Web page

- 1 Select **File|New|Web Page** from the main menu.



The **New Web Page File** wizard opens.



- 2 In the **Create a New Web Page File** page, make sure the parent folder is *Tutorial\webroot*.
- 3 In the **File name** field, enter `getTimeService`.
- 4 Click **Finish** to create the *getTimeService.jsp* page.

Sybase WorkSpace creates a *getTimeService.jsp* file with default JSF contents and opens it in the Web Page editor.

❖ **Mapping the JSF components to the beans attributes**

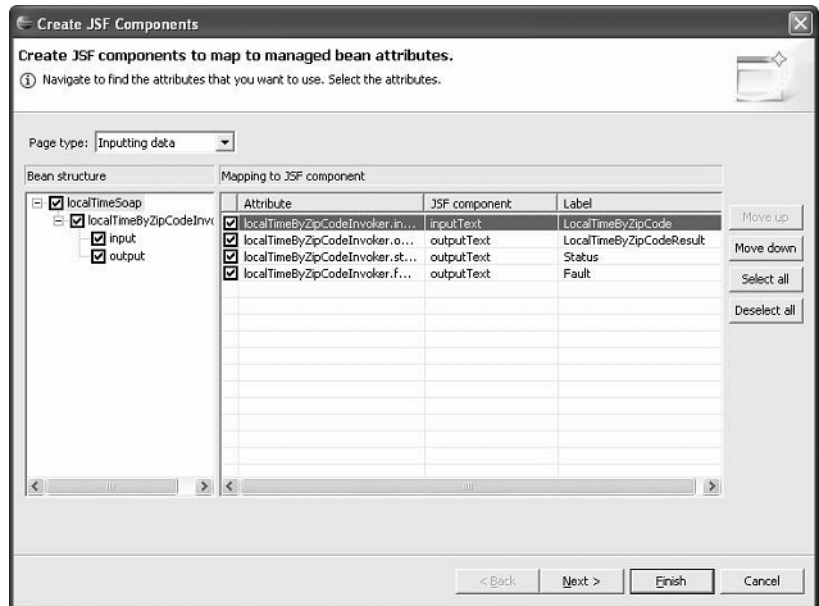
Map the appropriate JSF components to the service managed bean attributes and then generate the layout of the *getTimeService.jsp* page using the service managed bean that you just created..

- 1 In the Data Bindings view, drag and drop the **LocalTimeSoap** bean onto the design pane of the *getTimeService.jsp* page to open the **Create JSF Components** wizard to create JSF components that map to the managed bean's attributes.

As required, expand the wizard and the columns to see the entire attribute name and/or label.

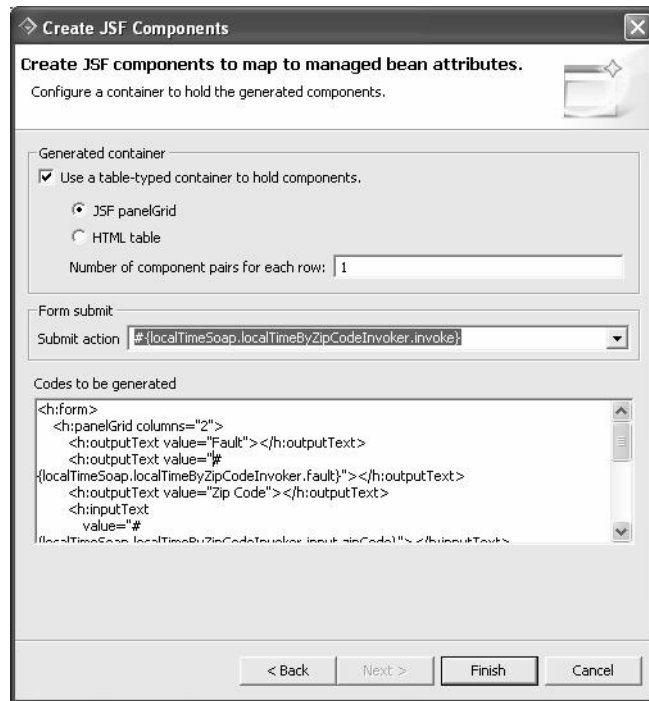
- 2 Move the **Fault** class member to the bottom of the list.

Select the row and click **Move Down** multiple times until the *Fault* class member is at the bottom of the list.



- 3 Accept the default values in the **Label** field and click **Next**.

- 4 Make sure that **JSF panelGrid** is selected as the generated container, accept the remaining default values, and click **Finish**.

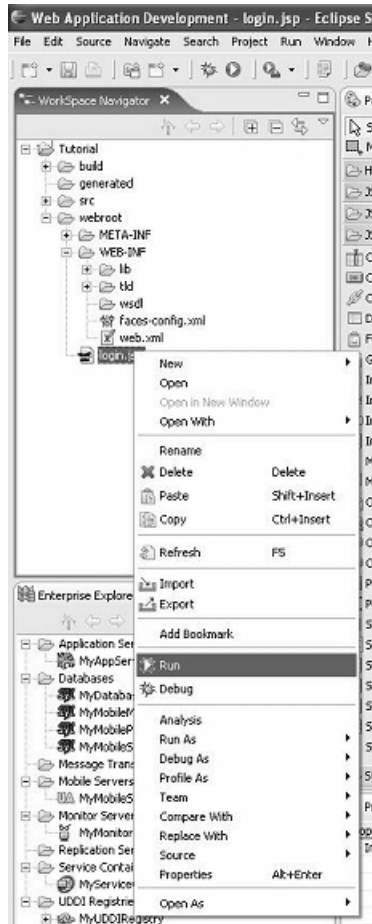


The Create JSF Components wizard generates a Web page grid with ZipCode as the input value and Local Time as the output value.

- 5 Select **File|Save** from the main menu bar to save the *getTimeService.jsp* page.
- 6 Select the **Preview** tab at the bottom of the editor to preview the Web page in the **Web Browser** view.

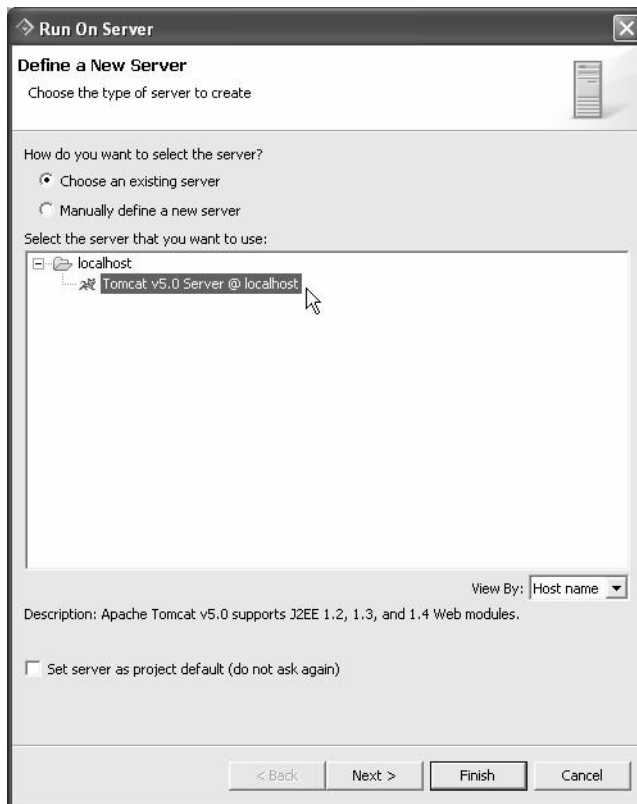
❖ Testing the `getTimeService.jsp` Web page

- 1 In the **Workspace Navigator**, expand the **webroot** folder, right-click the `getTimeService.jsp`, and select **Run** from the context menu.



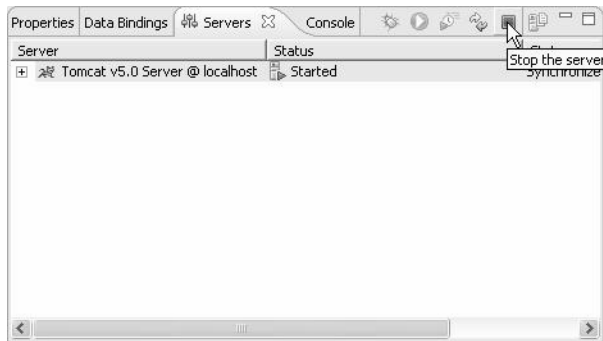
- 2 In the **Run On Server** wizard, select **Choose an existing server**, and select **Tomcat v5.0 Server @ localhost** from the list.

Note If you completed all previous tutorials in this guide, you already defined the Tomcat server. However, if it does not appear on the list, see “Starting the Tomcat 5.0 server” on page 38 for step-by-step instructions.



- 3 Click **Finish**.
The Apache Tomcat server starts, and the JSF Page Template view displays your Web page.
- 4 To test the Web page, enter a zip code in the **Zip Code** field, and click **Submit**.
- 5 View the results in the **Local Time By Zip Code Result** field.

- 6 Click the **Stop the server** icon in the **Servers** view to stop the server before continuing.



You are now ready to link the two Web pages you previously created, *login.jsp* and *getTimeService.jsp*. Continue to Chapter 5, “Linking Web Pages.”

Linking Web Pages

This chapter describes how to link a sequence of Web pages using navigation rules and how to test the linked pages.

Web Application Development allows you to define navigation rules that link a sequence of Web pages together using the JSF Application Configuration editor.

Linking Web pages

In previous tutorials, you created two Web pages: *login.jsp* and *getTimeService.jsp*. Next, define navigation rules that link these two pages in a sequence and then test the link:

Lesson 1: Define navigation rules

Lesson 2: Test linked pages

Before you can perform the steps in this tutorial, you must complete the tutorials in all the previous chapters in this guide.

Lesson 1: Define navigation rules

You define navigation rules in the *faces-config.xml* file.

Note Before you begin, verify that you have stopped the Tomcat server.

❖ Defining navigation rules

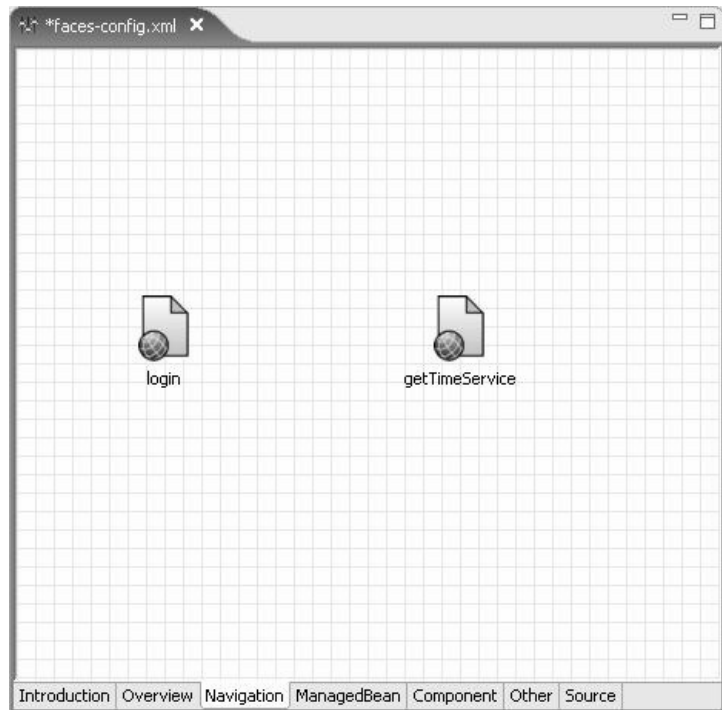
- 1 In the **WorkSpace Navigator**, expand the *Tutorial\webroot\WEB-INF* folders, right-click the *faces-config.xml* file and select **Open** from the context menu.

The XML file displays in the JSF Application Configuration editor.



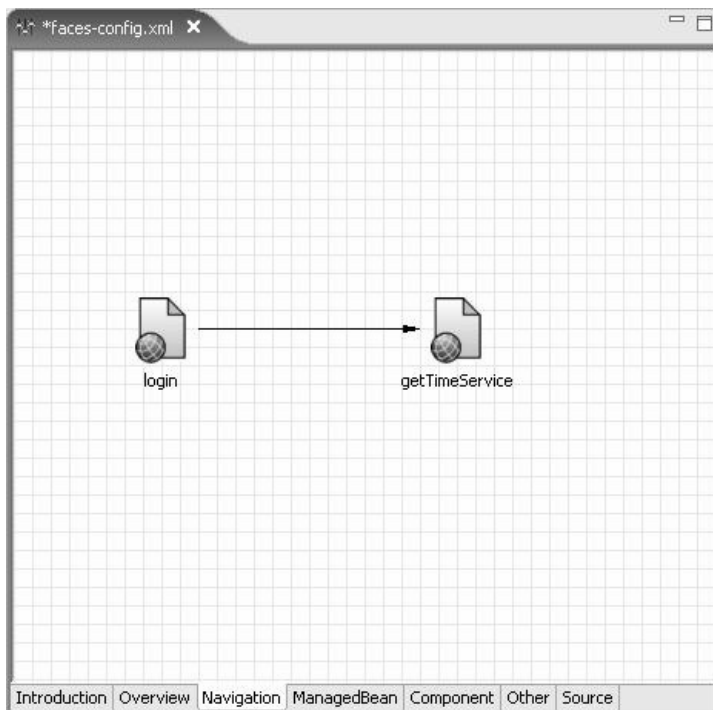
- 2 To select the Web pages to link, select the **Navigation** tab at the bottom of the editor.

- 3 Drag and drop the *login.jsp* and *getTimeService.jsp* files from the **Workspace Navigator** onto *faces-config.xml* file.

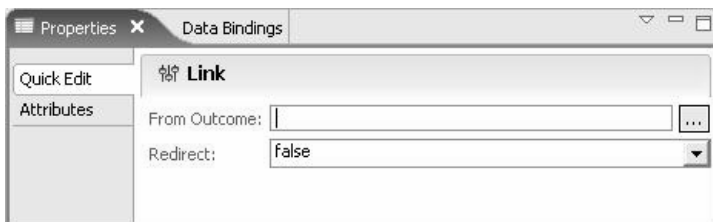


- 4 Select **Window|Show View|Palette** to open the **Palette** view to link the *login.jsp* page to the *getTimeService.jsp* page.
- 5 Select the **Link** control and drag it onto the editor.
- 6 Click the *login.jsp* page and draw a line to the *getTimeService.jsp* page.

- 7 Click *getTimeService.jsp* to release the cursor and set the link arrow in place.



- 8 In the **Palette** view, select the **Select** control and then select the link arrow.
- 9 Click the **Properties** tab at the bottom the perspective to define the link properties.



- 10 In the **From Outcome** field, enter `success`.
- 11 Select **File|Save** from the menu bar to save the *faces-config.xml* file.

Lesson 2: Test linked pages

❖ Testing the linked Web pages

Run the Web pages on the Tomcat 5.0 server.

- 1 In the **WorkSpace Navigator**, right-click the *login.jsp* file and select **Run** from the context menu.
- 2 Select **Choose an existing server** and select **Tomcat v5.0Server@localhost** from the list.
- 3 Click **Finish**.

The Apache Tomcat server starts and the **JSF Page Template** view opens.

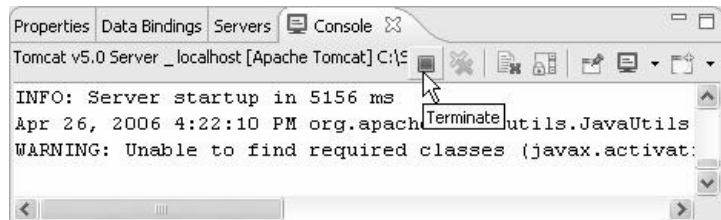
- 4 To test the Web page sequence in the browser, enter *sybase* as the password and click **Login**.

The *getTimeService.jsp* page opens in the browser.

- 5 Enter a zip code and click **Submit**.

The Local Time By Zip Code Result field displays the local date and time.

- 6 Click the **Terminate** icon in the **Console** view to stop the server before continuing.



Using DataWindow Objects

A DataWindow is an object that you use to retrieve, present, and manipulate data from a relational database or other data source. DataWindow objects have knowledge about the data they are retrieving. You can specify display formats, presentation styles, and other data properties so that users can make the most meaningful use of the data.

Web Application Development allows you to create new DataWindow objects and associate them with a Web page.

This chapter describes how to create a DataWindow object, how to implement DataWindow objects in a Web page, and how to test the Web page by running it on a server.

Creating a DataWindow object

The following lessons illustrate how to use Sybase WorkSpace to import a DataWindow library, create a DataWindow object, and use a DataWindow object on a Web page:

Lesson 1: Import a DataWindow library

Lesson 2: Create a DataWindow object

Lesson 3: Create Web pages that use DataWindow objects

Before you can perform the steps in this tutorial, you must complete the tutorials in all the previous chapters in this guide.

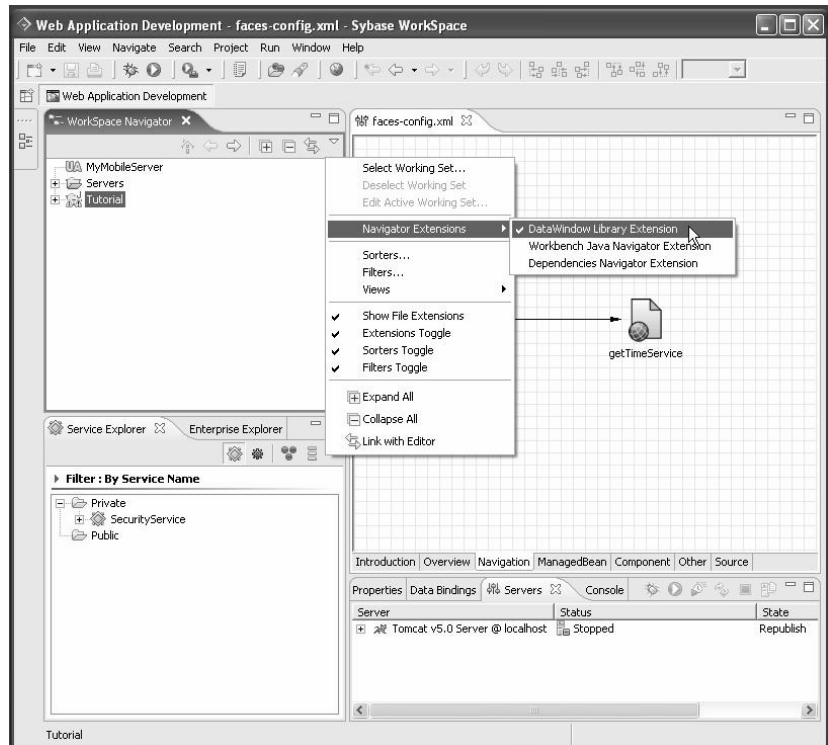
Lesson 1: Import a DataWindow library

You can import DataWindow libraries into Sybase WorkSpace and then reuse any DataWindow object in a Web page.

❖ Importing a DataWindow library

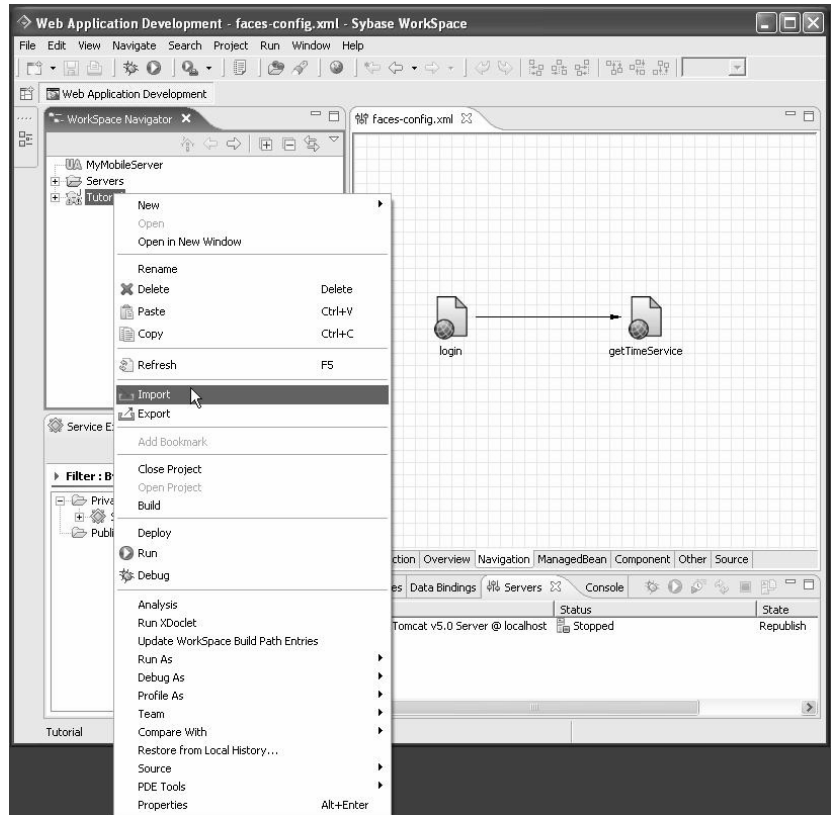
By default, DataWindow libraries extensions display during the import process.

- 1 To verify that the DataWindow library extensions are selected to display in the **WorkSpace Navigator**, click the **Menu** icon (down arrow) and select **Navigator Extensions|DataWindow Library Extension**.



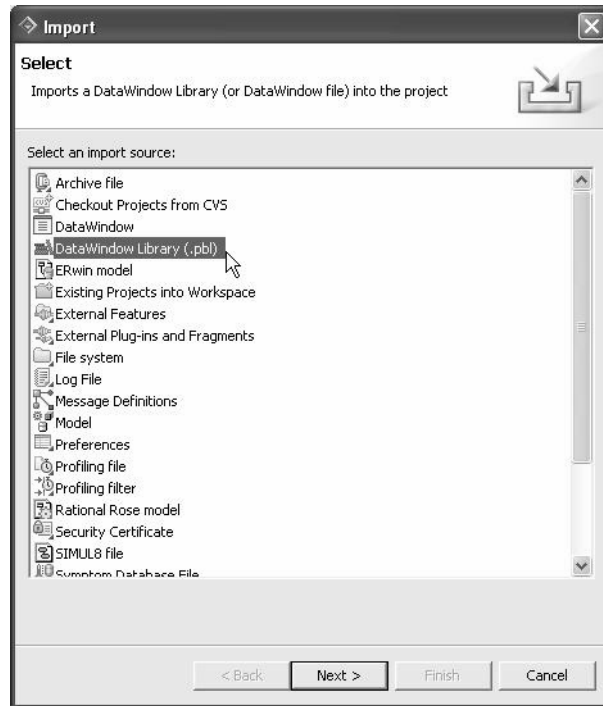
The check mark to the left of the menu item indicates it is already selected. Do not deselect the menu item by clicking on it.

- 2 In the **Workspace Navigator**, right-click **Tutorial** and select **Import** from the context menu to import an existing DataWindow library into your project.



The Import wizard displays.

- 3 Select **DataWindow Library (.pbl)** and click **Next**.

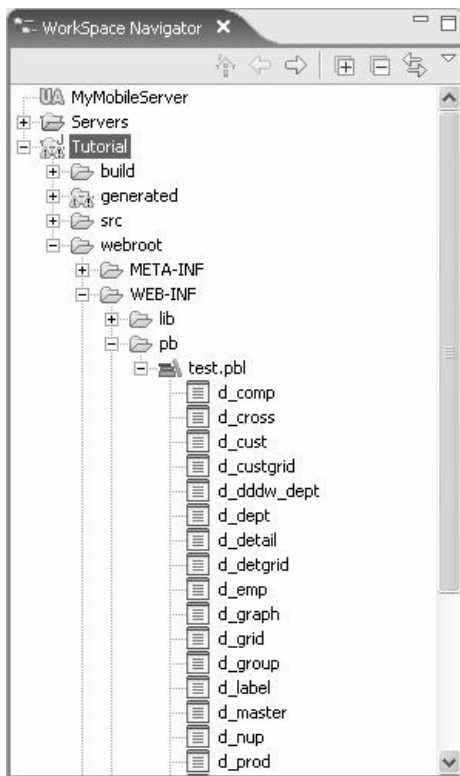


- 4 In the **From directory** field, click **Browse** and select this path:
`<installation directory>\sybase_workspace
\web_development\eclipse\plugins
\com.sybase.stf.jmt.template_1.5.0\tutorial
\webroot\WEB-INF\pb`
- 5 Click the **test.pbl** check box in the right pane.
- 6 Verify that the **Into folder** field reads:
`Tutorial\webroot\WEB-INF\pb`

- 7 Accept the remaining default settings, and click **Finish**.



- 8 Verify that you successfully imported the *test.pbl* file by expanding its contents in the **WorkSpace Navigator**.



❖ **Importing the connection profile and viewing an imported DataWindow**

A connection profile enables connection to servers, message transports, and databases. Before you can open and view a DataWindow, you must connect to the database so you have access to the DataWindow object data sources.

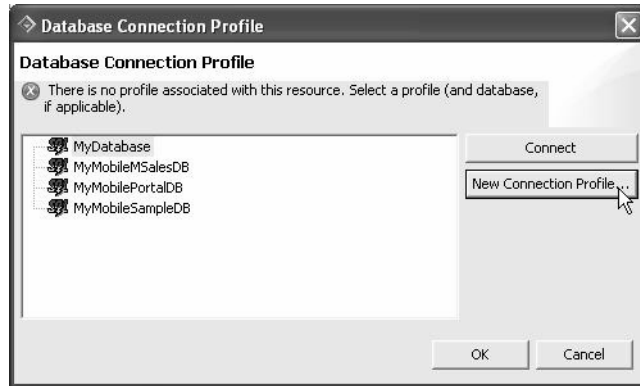
- 1 Double-click the *startdemo.bat* file to start the Adaptive Server Anywhere server and the sample database.

`<installation directory>\DevRuntimes\ASA\startdemo.bat`

Adaptive Server Anywhere successfully starts when you see its icon in the Windows taskbar.



- 2 In the **WorkSpace Navigator**, right-click the **d_emp** DataWindow library and select **Open** from the context menu to create a connection profile for the tutorial database that contains the datasources for this DataWindow object.
- 3 In the **Database Connection Profile** dialog box, click **New Connection Profile**.



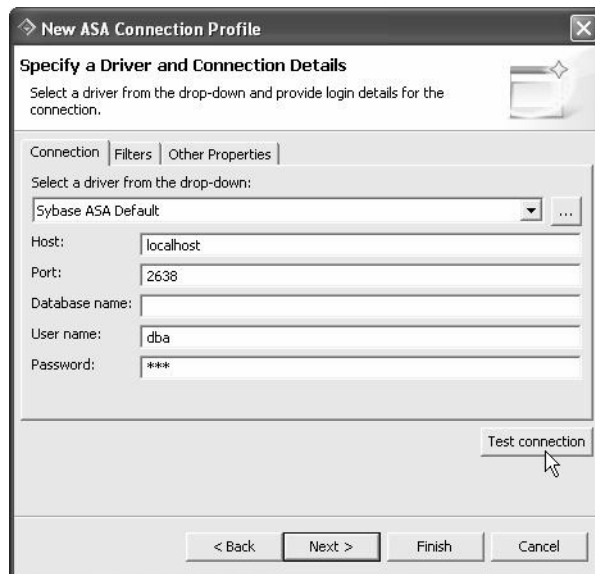
- 4 In the **New Connection Profile** wizard, select **Sybase ASA** and click **Next**.



- 5 In the **Name** field, enter `webapp_db` and click **Next**.



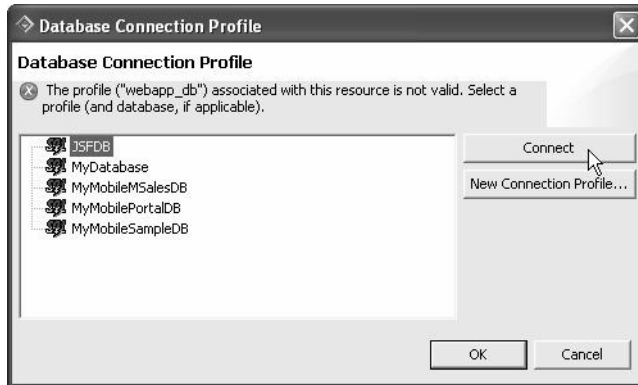
- 6 In the **Specify a Driver and Connection Details** page, click **Test connection** to verify that you can successfully connect to the server.



Do not proceed until the ping succeeds. If the ping does not succeed, see your system administrator for assistance.

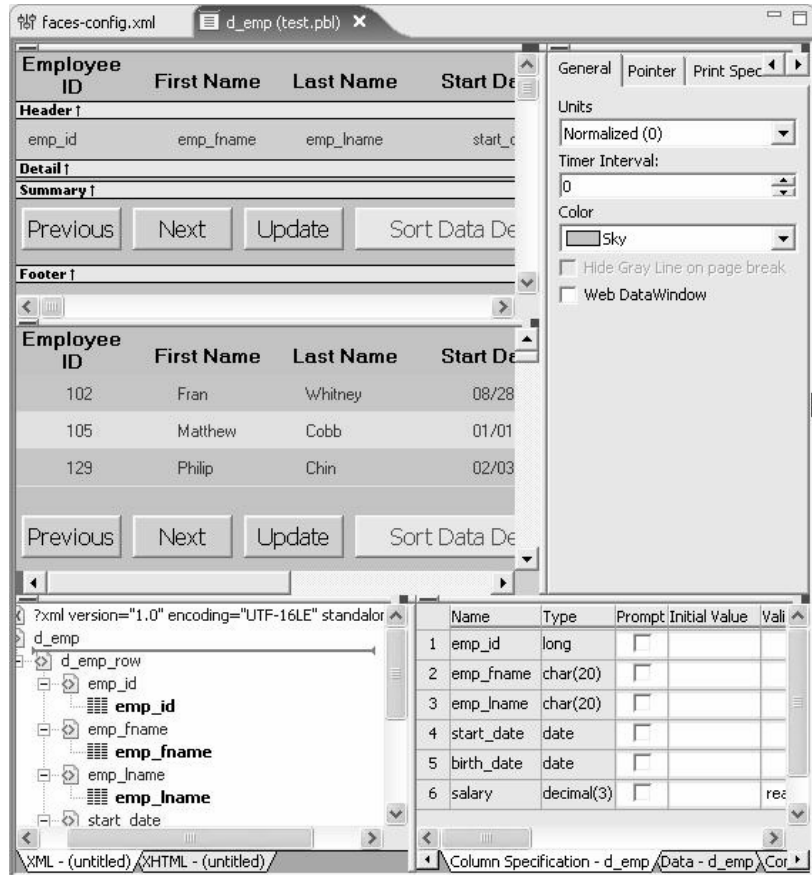
In the **Success** dialog box, click **OK** to continue.

- 7 Click **Finish** to create the connection profile.
- 8 In the **Database Connection Profile** dialog box, click **webapp_db** and then **Connect**.



- 9 Click **OK** to connect to the *asaDemo* database.

The *d_emp* DataWindow opens in the editor.



Lesson 2: Create a DataWindow object

Create a DataWindow object to add it to the DataWindow library.

❖ Creating a new DataWindow object

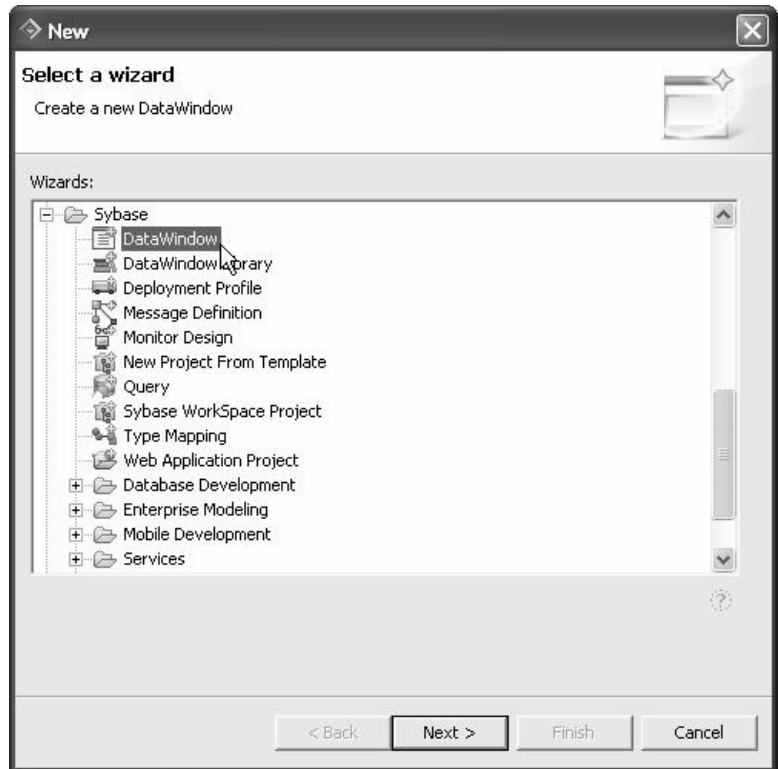
- 1 Verify that the Adaptive Server Anywhere database is running.

Look for the ASA icon in the Windows taskbar.

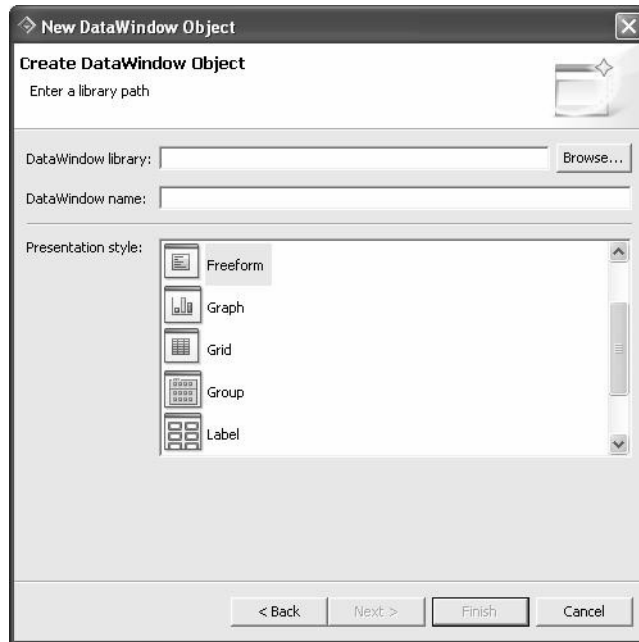


If you need to start Adaptive Server Anywhere, see “Importing the connection profile and viewing an imported DataWindow” on page 68.

- 2 Select **File|New|Other** from the menu bar to open the **New DataWindow Object** wizard.
- 3 Scroll down and expand the **Sybase** folder, select **DataWindow**, and click **Next**.



The New DataWindow Object dialog box displays.

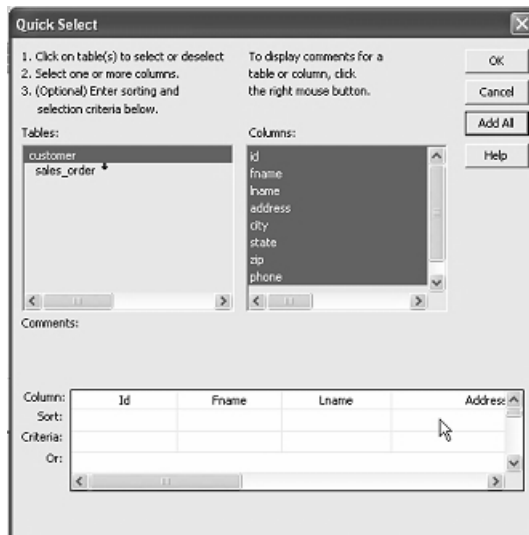


- 4 In the **DataWindow** library field, enter *Tutorial\webroot\WEB-INF\pb\test.pbl*.
- 5 In the **DataWindow** name field, enter *mycustomer*.
- 6 From the **Presentation style** list, select **Freeform** and click **Next**.
- 7 In the **Database Connection Profile** page, select **webapp_db** and click **Next**.

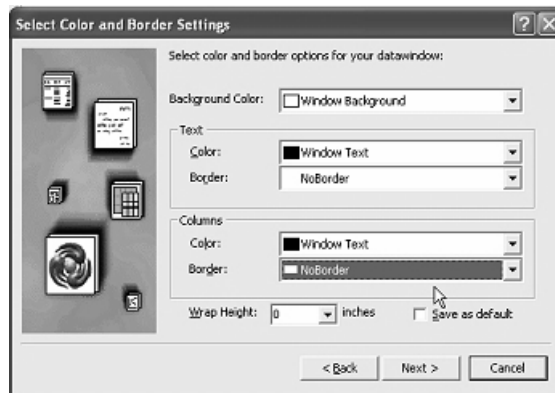
- 8 In the **Choose DataSource for Freeform DataWindow** page, select **Quick Select** as the data source and click **Next**.



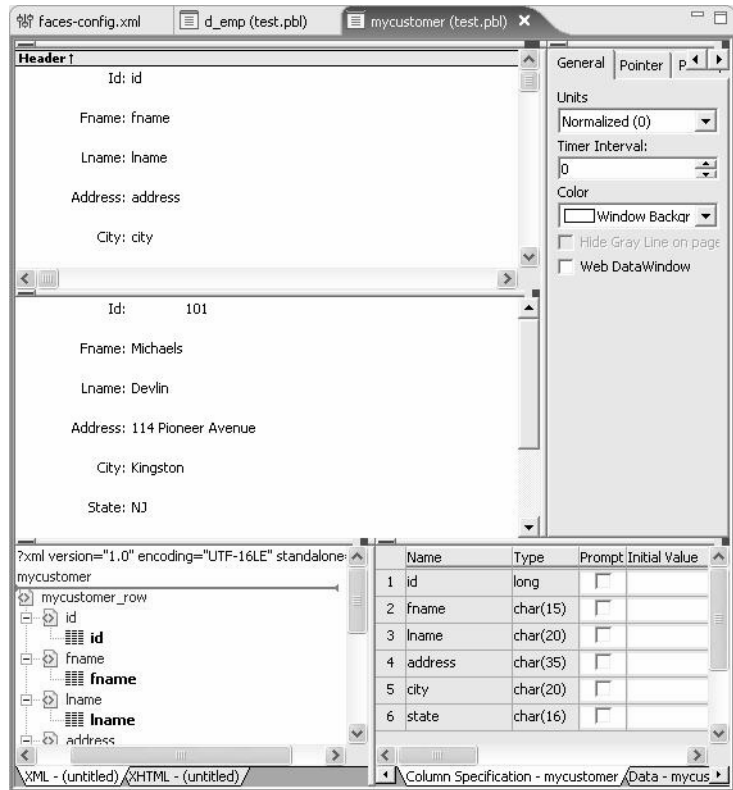
- 9 In the **Quick Select** page, select **customer** from the **Tables** list, click **Add All** to add all columns to the form, and then click **OK**.



- 10 In the **Select Color and Border Settings** wizard, accept the default settings for the color and border settings, and click **Next**.



- 11 Review the **Ready to Create Freeform DataWindow** page and click **Finish** to create the DataWindow object.

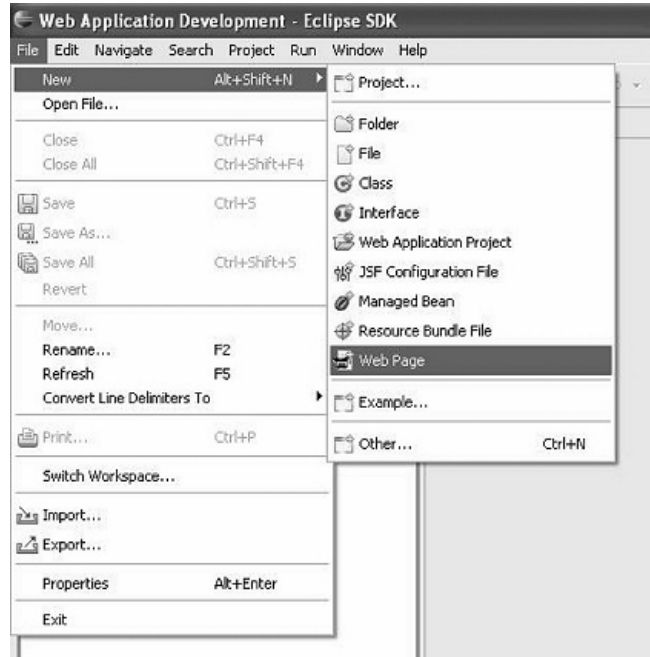
12 View the **mycustomer (test.pbl)** DataWindow in the editor.

Lesson 3: Create Web pages that use DataWindow objects

In this lesson, you return to the *d_master* and *d_detail* DataWindow libraries that you imported earlier and create a Web page using DataWindow objects in these libraries.

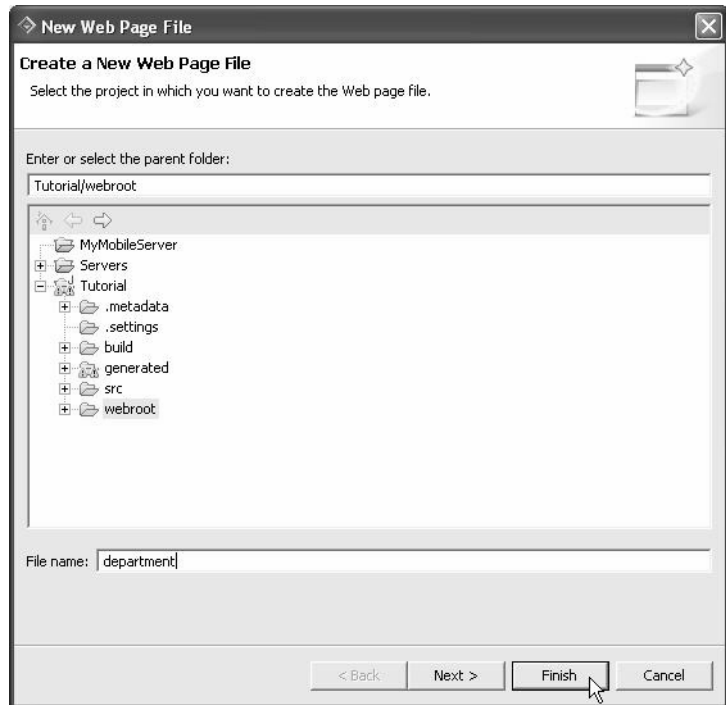
❖ **Creating a Web page using a DataWindow object**

- 1 Select **File|New|Web Page** from the menu bar to open the **New Web Page** wizard.



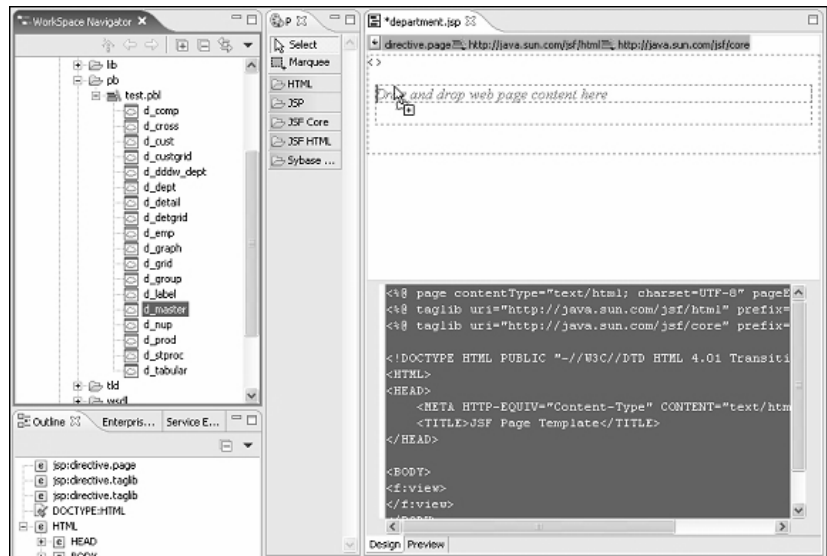
- 2 In the **Create a New Web Page File** page, make sure that parent folder is *Tutorial\webroot*.

- 3 In the **File name** field, enter `department`, and click **Finish**.

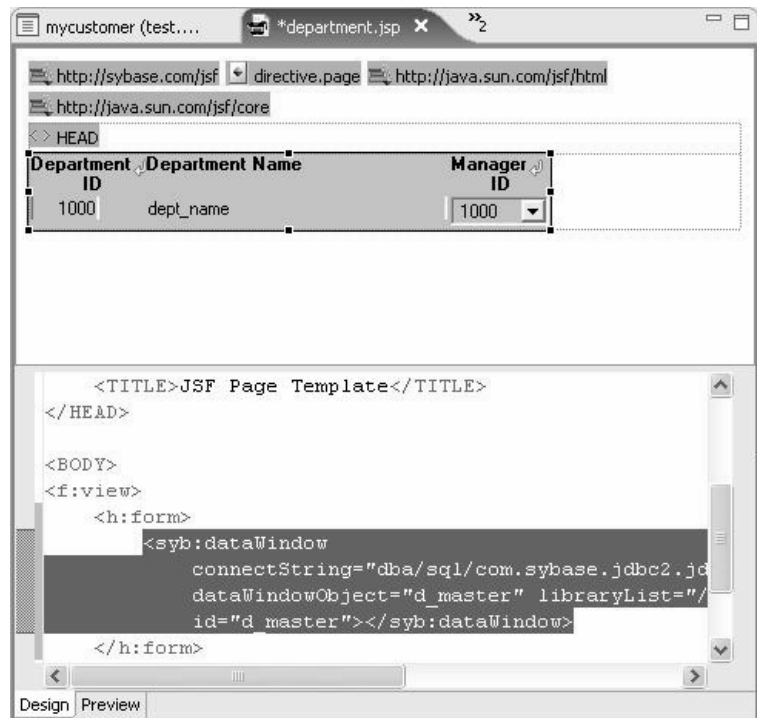


Sybase WorkSpace creates *department.jsp* with the default JSF contents and opens it in a Web Page editor.

- 4 To open a DataWindow object, in WorkSpace Navigator, expand the **Tutorial\webroot\WEB-INF\pb\text.pb1** folder, and drag and drop the **d_master** DataWindow onto the design pane of the *department.jsp* page.

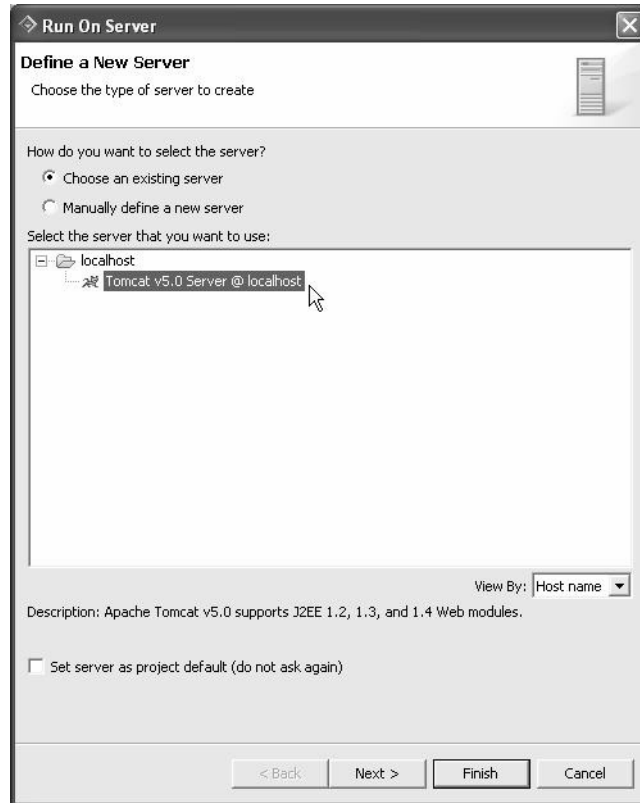


The Web Page editor displays code for that DataWindow and its graphical representation on the design pane.

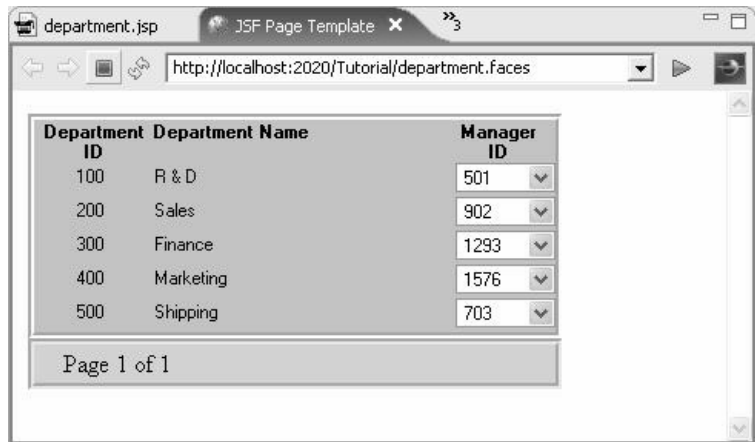


- 5 Select the **Preview** tab in the Web Page editor to preview the Web page.
- 6 Select **File|Save** from the menu bar to save the Web page.
- 7 To test the Web page on the Tomcat 5.0 server, in the **WorkSpace Navigator**, right-click *department.jsp* and select **Run** from the context menu.

- 8 Select **Choose an existing server**, select **Tomcat v5.0Server @ localhost**, and click **Finish**.



The Apache Tomcat server starts, and the JSF Page Template opens.



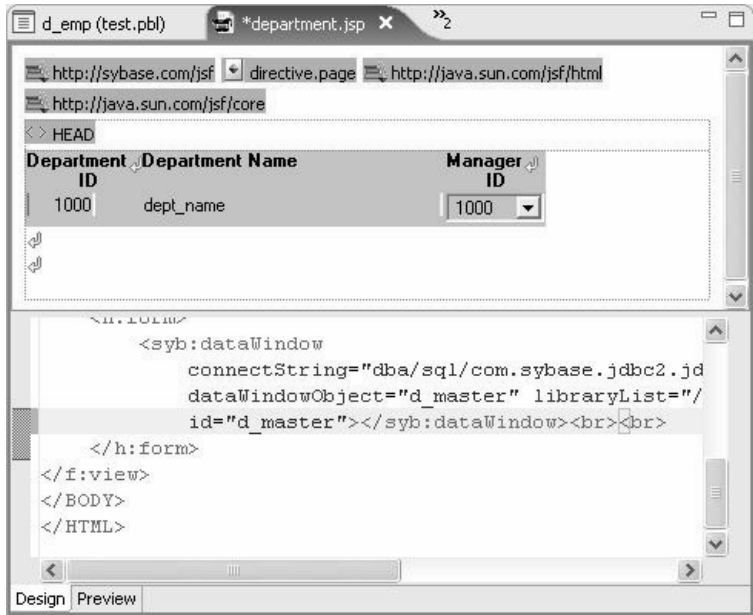
- 9 In the Console view, click the **Terminate** icon to stop the server before continuing.

❖ Adding a detail DataWindow object to a Web page

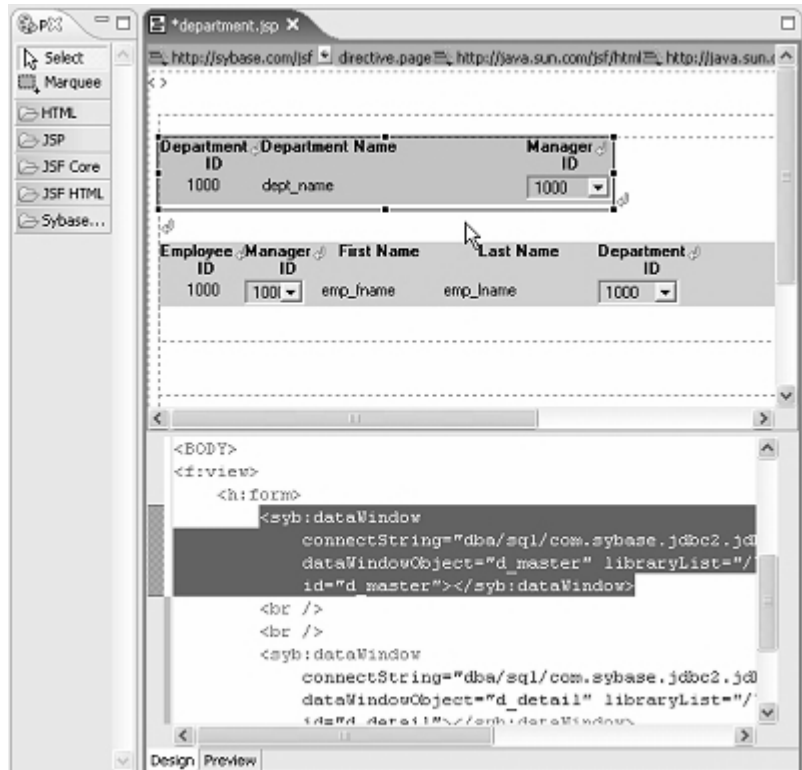
Add a detail DataWindow object to the Web page.

- 1 At the bottom of the *department.jsp* page, click the **Design** tab to switch to the Web Page editor.
- 2 In the design pane, place your cursor at the end of the DataWindow object and press **Enter** twice to add two line break tags.

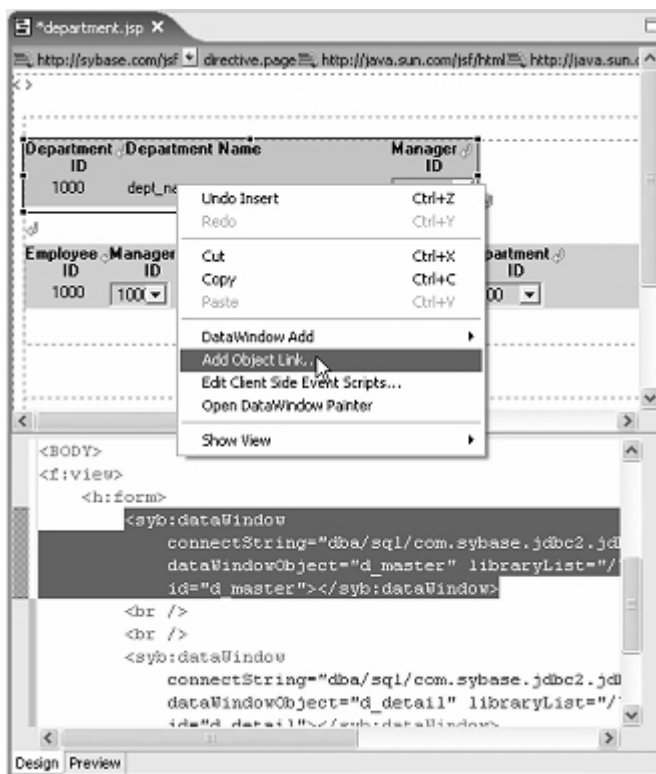
These appear as `
` `
` in the Web Page editor source view.



- 3 In **WorkSpace Navigator**, expand the **Tutorial\webroot\WEB-INF\pb\text.pbl** folder and drag and drop the **d_detail** DataWindow under the second line break on the design pane of the *department.jsp* page.



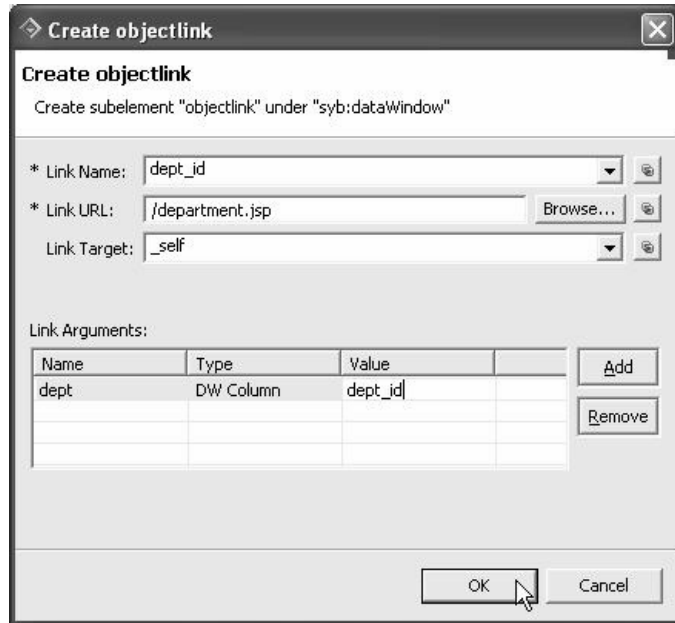
- 4 On the *department.jsp* page, right-click the **Department ID** DataWindow and select **Add Object Link** from the context menu.



The **Create objectlink** dialog box displays.

- 5 In the **Link Name** field, enter `dept_id`.
- 6 In the **Link URL** field, enter `/department.jsp`.
- 7 In the **Link Target** field, enter `_self`.
- 8 In the **Link Arguments** list, click **Add** to define the argument properties and then click **OK**.
 - In the **Name** field, enter `dept`.
 - In the **Type** field, select **DW Column** from the drop-down menu.

- In the **Value** field, enter `dept_id`.

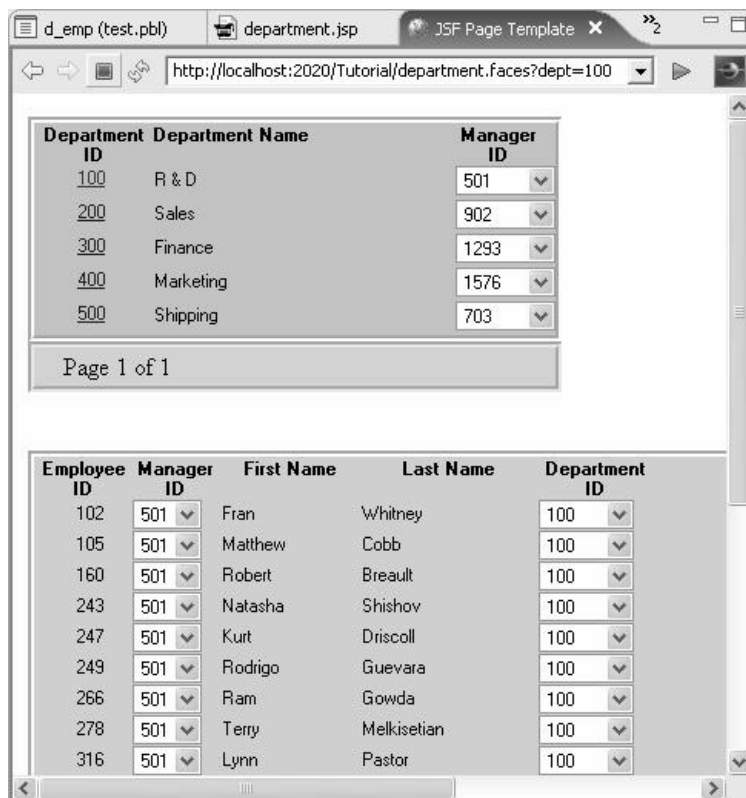


- 9 Select **File|Save** from the menu bar to save the Web page.
- 10 To test the Web page on the server, right-click **department.jsp** and select **Run** from the context menu.

The *department.jsp* page opens in the JSF Page Template.

- 11 Click any link under **Department ID**.

The bottom of the JSF Page Template window displays the employee details below.



Using DataWindow advanced features

DataWindow advanced features support both the creation of server-side and client-side events. You implement server-side events in Java code and client-side events using JavaScript.

This tutorial contains these lessons:

- Lesson 1: Add server-side events
- Lesson 2: Add client-side events

Lesson 1: Add server-side events

Add server-side events, which are executed during runtime, to your DataWindow objects. DataWindow server-side events are implemented in Java code with a defined event handler interface.

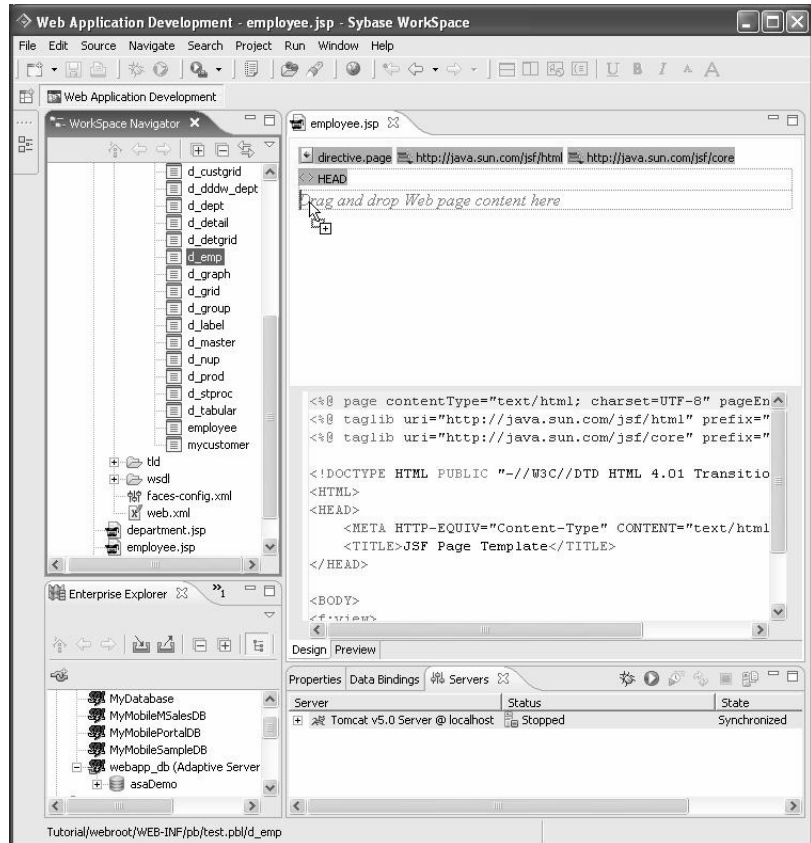
❖ **Creating the Web page and adding the DataWindow object**

In this tutorial, you will create a Web page called *employee.jsp* to which you will then add server-side events to a DataWindow object. The server-side events enable you to page through the employee information retrieved from the server using the **Previous** and **Next** buttons.

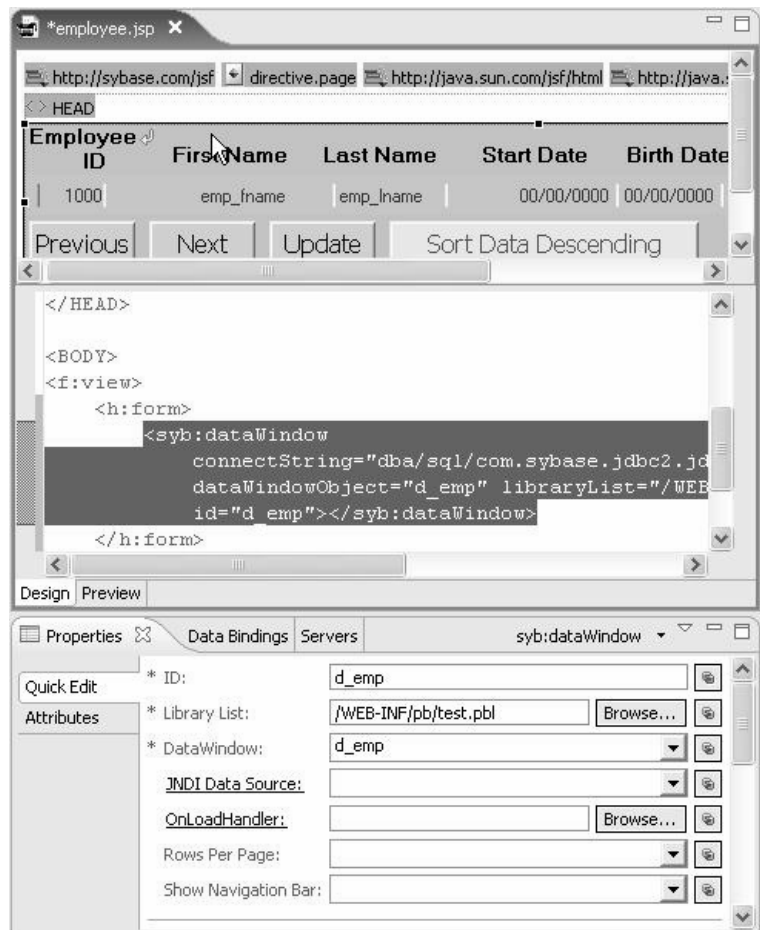
- 1 Select **File|New|Web Page** from the menu bar to open the **New Web Page File** wizard.
- 2 In the **Create a New Web Page File** page, make sure that parent folder is *Tutorial\webroot*.
- 3 In the **File name** field, enter `employee`, and click **Finish**.

Sybase WorkSpace creates the *employee.jsp* with the default JSF contents and opens it in a Web Page editor.

- To add server-side events to the Web page, in **WorkSpace Navigator**, expand the **Tutorial\webroot\WEB-INF\pb\test.pbl** folder and drag and drop the **d_emp** DataWindow onto the design pane of the *employee.jsp* page.

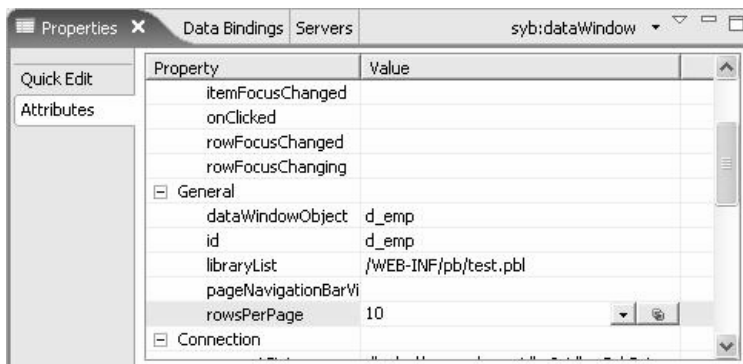


- To display the properties for the Employee ID DataWindow object, click the object in the design pane and then click the **Properties** view.



- In the **Properties** view, select the **Attributes** tab.

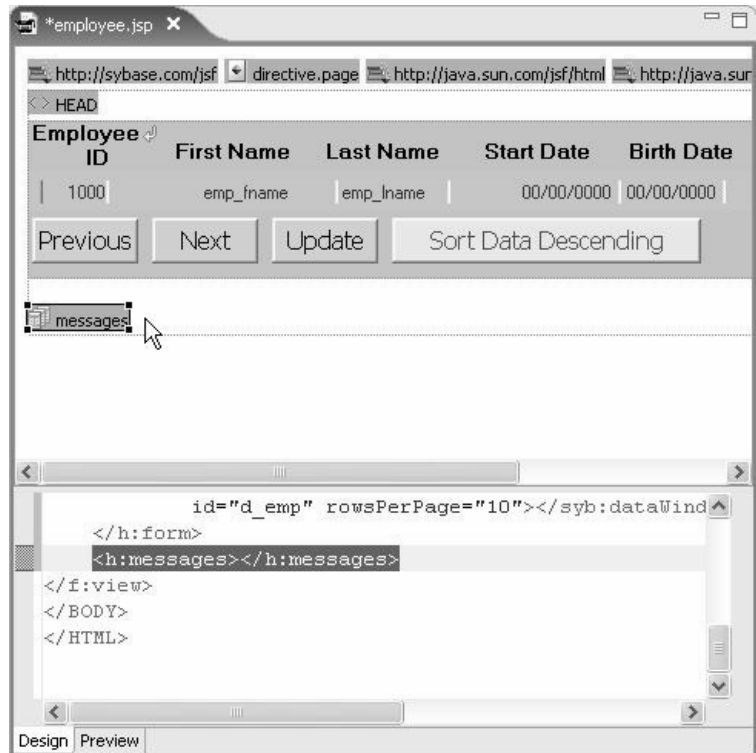
- 7 Locate the **rowsPerPage** attribute under the General category, and enter 10 in the **Value** column to display 10 records per Web page.



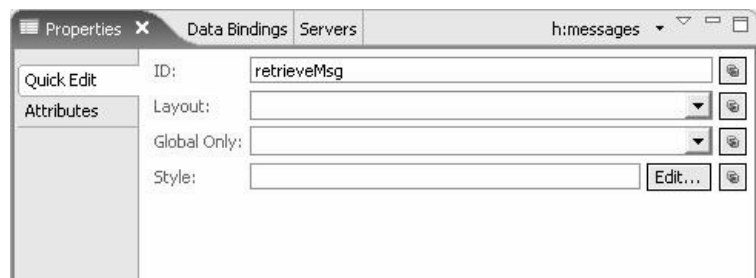
❖ **Adding a messages control to the Web page**

- 1 Select **Window|Show View|Palette** from the main menu to add a messages control to the Web page.

- In the Palette, expand the **JSF HTML** folder, and then drag and drop a messages control directly after the Employee ID DataWindow on the *employee.jsp* page.

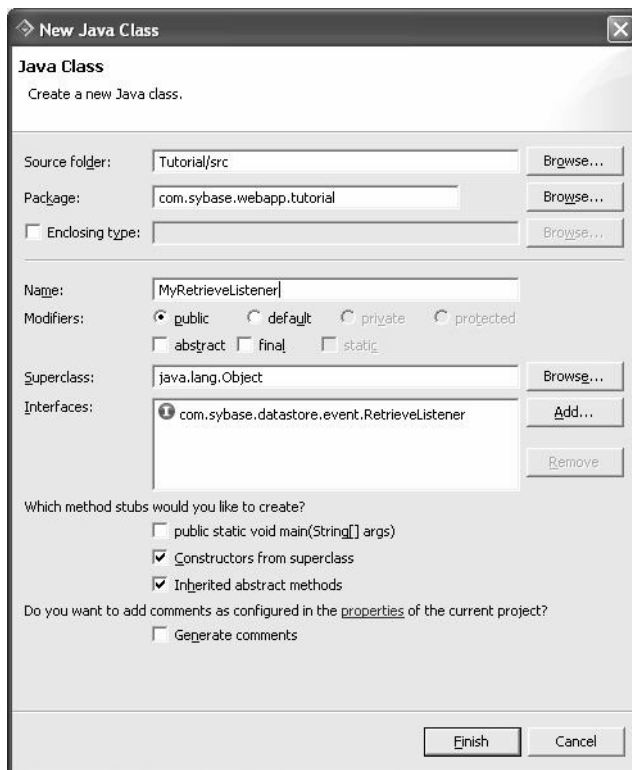


- If not selected, click the **messages** control in the design pane, and in the **Properties** view, select the **Quick Edit** tab.
- In the **ID** field, enter `retrieveMsg`.



- 5 To add a DataWindow object retrieve listener, right-click the Employee ID DataWindow in the *employee.jsp* page, and select **Add Listener|RetrieveListener** from the context menu.
- 6 In the **Create retrieveListener** dialog box, click **Type** to create a new Java class.

The New Java Class dialog box opens.
- 7 In the **Source Folder** field, make sure it reads *Tutorial\src*.
- 8 In the **Package** field, enter `com.sybase.webapp.tutorial`.
- 9 In the **Name** field, enter `MyRetrieveListener`.
- 10 Accept the remaining default settings and click **Finish** to create the Java class and return to the **Create retrieveListener** dialog box.



- 11 In the **Create retrieveListener** dialog box, click **OK** to create the `MyRetrieveListener` interface and add it to the Interfaces list.

Next, you are going to replace the default generated code in the Web Page editor with provided sample code that enables the display of messages in the Web browser as data is retrieved from the database.

- 12 Select **File|Open File** from the main menu bar to open the following file:

```
<installation directory>\sybase_workspace\web_development\eclipse  
\plugins\com.sybase.stf.jmt.template_1.5.0\tutorial\src\com\sybase  
\webapp\tutorial\MyRetrieveListener.java
```

- 13 Copy the contents of the Sybase-provided *MyRetrieveListener.java* file and replace them with the existing contents of the *MyRetrieveListener.java* file in the Web Page editor, and then close the Sybase-provided Java file.
- 14 Select **File|Save** from the menu bar to save the *MyRetrieveListener.java* file.

❖ **Testing the Web page on the Tomcat 5.0 server**

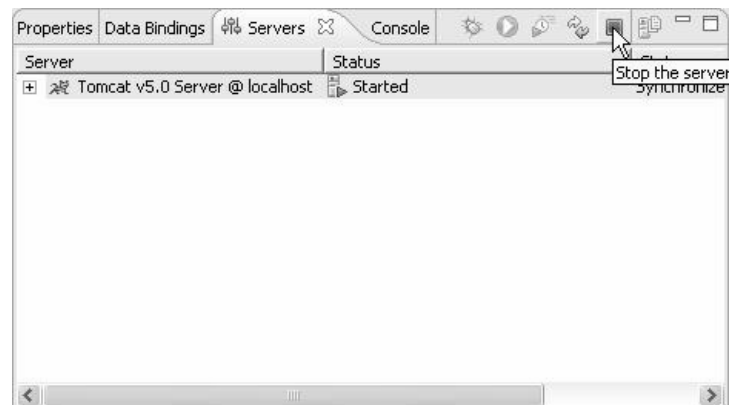
- 1 If necessary, save the *employee.jsp* file.
- 2 In the **WorkSpace Navigator**, right-click *employee.jsp* and select **Run** from the context menu.
- 3 Select **Choose an existing server**, select **Tomcat v.5.0Server @ local host** from the list, and click **Finish**.

The Apache Tomcat server starts, and the JSP Page Template opens.

- To test the Web page, click the **Previous** or **Next** button at the bottom of the window.



- Before continuing, click the **Stop the server** icon in the **Servers** view to stop the server.



Now you are ready to add client-side events to a DataWindow object, as described in “Lesson 2: Add client-side events,” below.

Lesson 2: Add client-side events

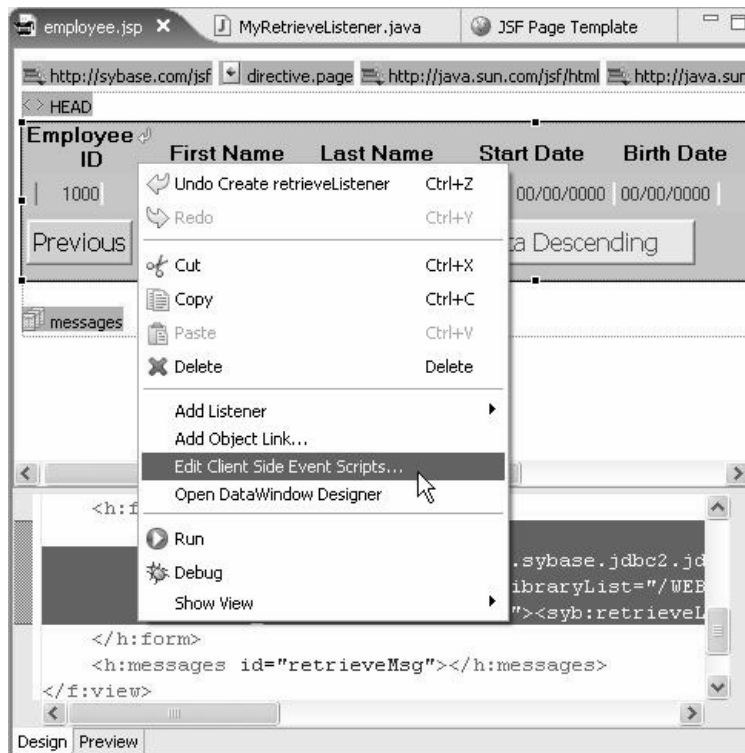
Client-side events and methods enable you to create Web pages with dynamic content without a round trip to the server, which can improve performance. During runtime, the runtime engine executes these events.

In this lesson, add a client-side event to your DataWindow object. This client-side event enables you to sort employee information in descending order by employee ID.

❖ Adding client-side events to a DataWindow object

- 1 Display the *employee.jsp* page in the Web Page editor.

- 2 In the Web Page editor design pane, right-click the Employee ID DataWindow object, and select **Edit Client Side Event Scripts** from the context menu.



Next, edit the client-side event script to sort employee records in descending order when you click the **Sort Data Descending** button on the DataWindow object in the browser.

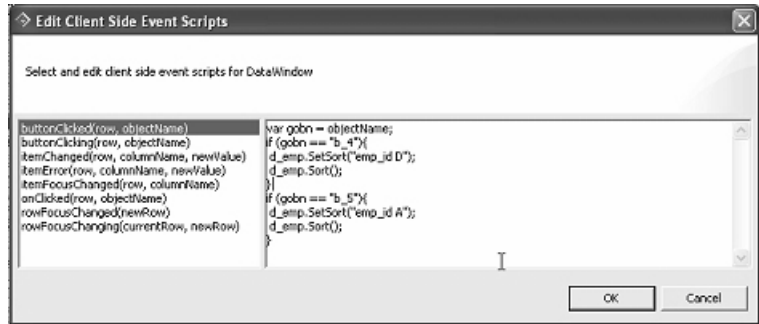
- 3 Select **buttonClicked(row,objectName)**.
- 4 Enter the following JavaScript in the editor box and click **OK**.

```
var gobn = objectName;
if (gobn == "b_4") {
d_emp.SetSort("emp_id D");
d_emp.Sort();
}
if (gobn == "b_5") {
```

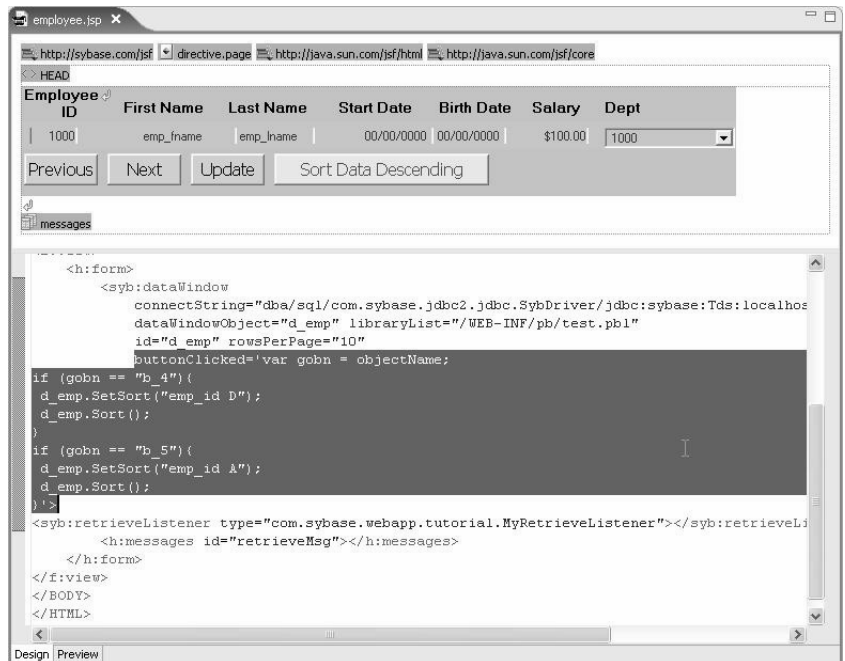
```

d_emp.SetSort ("emp_id A");
d_emp.Sort ();
}

```



The code for the client-side events display in the Web Page editor.



❖ **Testing the employee.jsp Web page on the Tomcat 5.0 server**

- 1 Select **File|Save** from the menu bar to save the *employee.jsp* file.
- 2 Right-click the design pane and select **Run** from the context menu.
- 3 Select **Choose existing server**, select **Tomcat v5.0Server @ localhost** from the list, and click **Finish**.

The Apache Tomcat server starts, and the JSF Page Template opens.

- 4 Click the **Sort Data Descending** button on the Web page to sort the client-side event data in a descending order by Employee ID.

Employee ID	First Name	Last Name	Start Date	Birth Date	Salary	Dept
1751	Alex	Ahmed	07/12/1994	12/12/1963	\$34,992.00	400
1740	Robert	Nielsen	06/24/1994	06/19/1965	\$34,889.00	400
1684	Janet	Hildebrand	03/15/1994	10/31/1955	\$45,829.00	400
1658	Michael	Lynch	02/27/1994	01/18/1973	\$24,903.00	500
1643	Elizabeth	Lambert	12/15/1993	09/12/1968	\$29,984.00	400
1615	Sheila	Romero	11/19/1993	09/12/1972	\$27,500.00	500
1607	Mark	Morris	10/13/1993	01/08/1941	\$61,300.00	400
1596	Catherine	Pickett	08/12/1993	11/18/1959	\$47,653.00	200
1576	Scott	Evans	07/01/1993	11/15/1960	\$68,940.00	400
1570	Anthony	Rebeiro	05/29/1993	04/12/1963	\$34,576.00	500

beginRetrieveEvent.getRetAction() = com.sybase.datastore.enums.BeginRetrieveAction@ee6ad6 #
rowRetrievedEvent.getRowNumber() = 1 # rowRetrievedEvent.getRowNumber() = 2 #
rowRetrievedEvent.getRowNumber() = 3 # rowRetrievedEvent.getRowNumber() = 4 #
rowRetrievedEvent.getRowNumber() = 5 # rowRetrievedEvent.getRowNumber() = 6 #
rowRetrievedEvent.getRowNumber() = 7 # rowRetrievedEvent.getRowNumber() = 8 #
rowRetrievedEvent.getRowNumber() = 9 # rowRetrievedEvent.getRowNumber() = 10 #
rowRetrievedEvent.getRowNumber() = 11 # rowRetrievedEvent.getRowNumber() = 12 #

- 5 Before continuing, click the **Stop the server** icon in the **Servers** view to stop the server.

Debugging Web Applications

The chapter introduces the Debug perspective in Sybase WorkSpace. In this tutorial, you will set breakpoints in your Web application code, which causes Sybase WorkSpace to launch the Debug perspective.

When you create your own Web applications, the Debug perspective and its associated views enables you to step through the breakpoints, identify compilation errors in the Java source code, view error logs, and perform other basic debugging tasks.

Debug perspective

Throughout the development cycle, you will use Web Application Development tooling to debug your Web pages. You can debug code by setting breakpoints on the Web page or in Java managed beans by setting breakpoints in the Java class.

Before you can perform the debugging tutorials, you must complete all the previous tutorials in the guide.

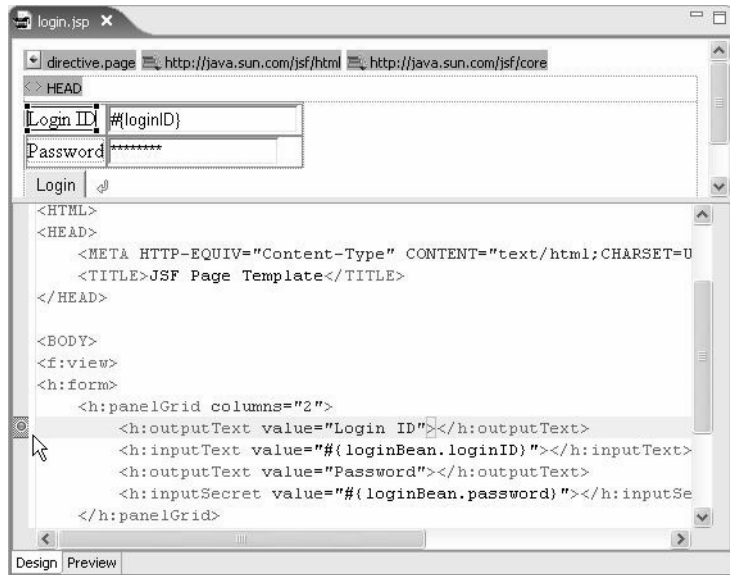
❖ Setting breakpoints and launching the Debug perspective

Note Before you begin the debugging process, be sure that the Tomcat server is stopped.

- 1 In the **WorkSpace Navigator**, double-click the *login.jsp* file to open the file in the Web Page editor.
- 2 To set a breakpoint in the Web Page editor, double-click in the vertical gray border to the left of the line following the `<h:panelGrid>` tag:

```
<h:outputText value="Login ID"></h:outputText>
```

A blue dot in the gray border indicates that you successfully set the breakpoint.

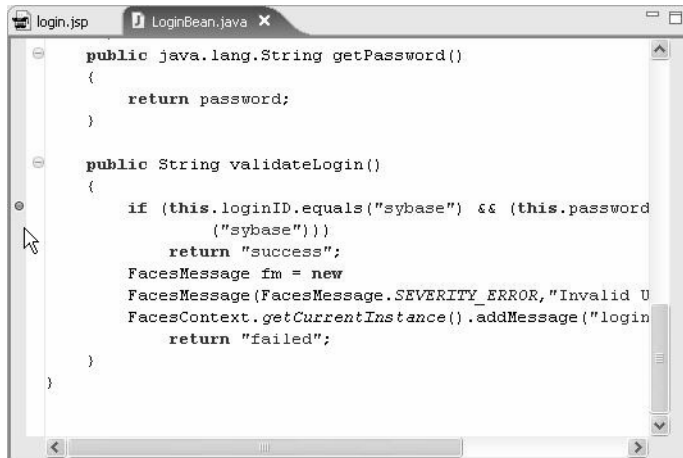


Next, set a breakpoint in the *loginBean*.

- 3 In the **Data Bindings** view, right-click **loginBean** and select **Open** from the context menu.
- 4 In the source view of the Web Page editor, set a breakpoint by double-clicking in the gray border to the left of the line following the *if* statement inside the *validateLogin()* method.

```
if (this.loginID.equals("sybase") &&
    (this.password.equals("sybase")))
```


A blue dot appears in the gray border identifying the breakpoint.

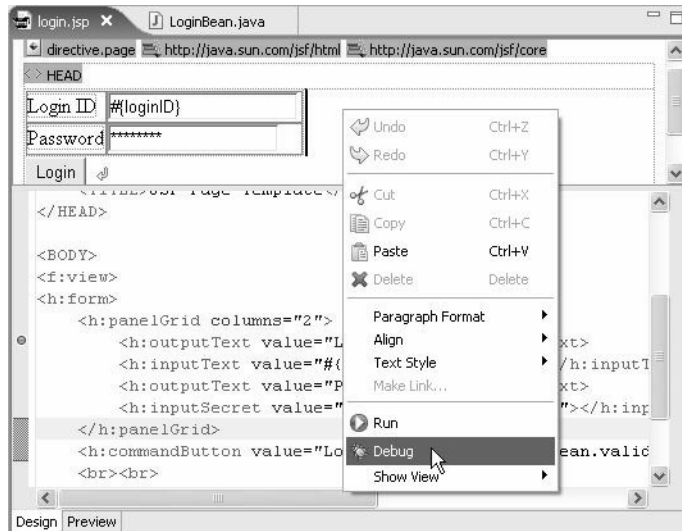


```

login.jsp LoginBean.java
public java.lang.String getPassword()
{
    return password;
}

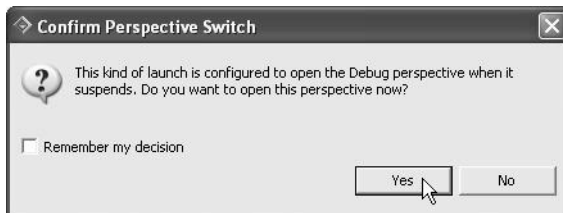
public String validateLogin()
{
    if (this.loginID.equals("sybase") && (this.password
        ("sybase")))
        return "success";
    FacesMessage fm = new
    FacesMessage (FacesMessage.SEVERITY_ERROR, "Invalid U
    FacesContext.getCurrentInstance().addMessage ("login
        return "failed";
}
  
```

- Return to *login.jsp* page, right-click the design pane, and select **Debug** from the context menu to debug the Web page.

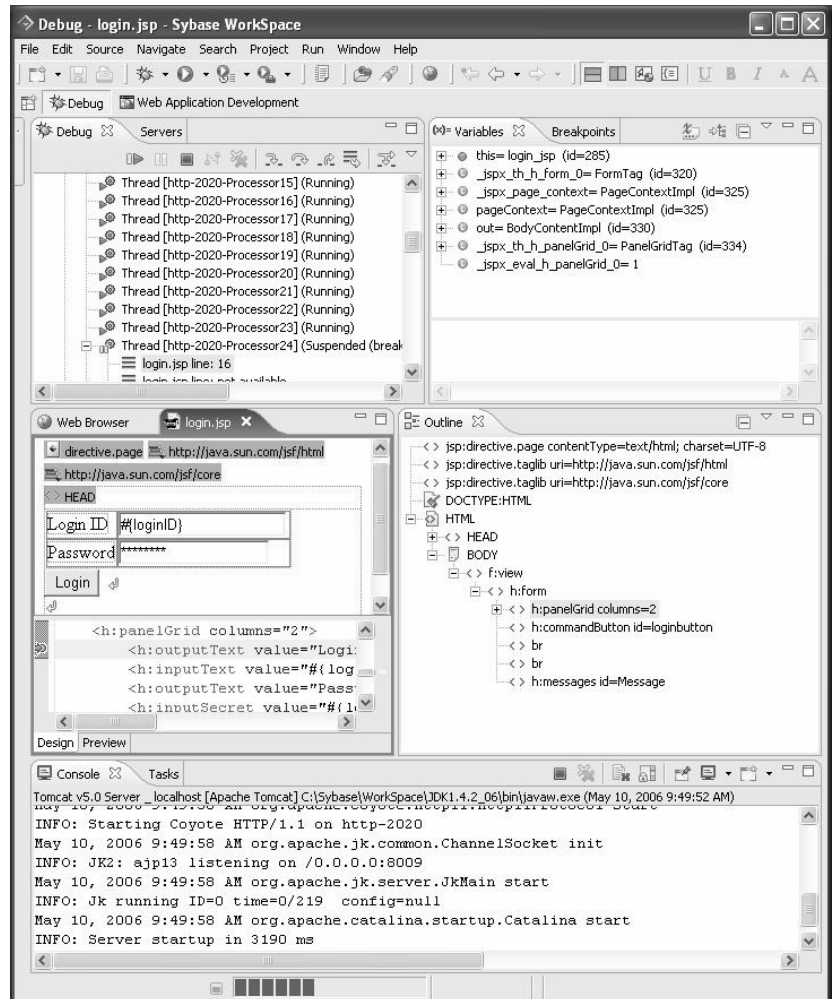


- 6 In the **Debug on Server** wizard, select **Choose an existing server**, select **Tomcat v5.0Server @ localhost**, and click **Finish**.

The Confirm Perspective Switch dialog box displays, which confirms that you want to change to the Debug perspective.



7 Click Yes to open the Debug perspective.



In the Debug perspective, you can run through the breakpoints and perform debugging tasks for your Web applications. You have now completed the Web Application Development component tutorial.

Available debugging tools

These tools can assist you in troubleshooting design and development problems:

- Problems view** Refer to the Problems view to identify compilation errors in the Java source code of a JSP page.
- Errors markers** Use the errors markers that appear on the vertical ruler of the source view in the Web Page editor to identify the cause and solution for an error. Move your mouse over the error marker to display the problem cause; double-click the error marker to display possible resolutions.
- Error Log view** Review error logs to identify design and runtime errors.
- Select **Windows|Show View|Others** from the menu bar, and then select **PDE Runtime|Error Log** from the list.