



# CHAPTER 22

**Constructing JSP Pages  
with BC4J Data Tags**

*Hark! Hark! The dogs do bark,  
The beggars are coming to town;  
Some in rags, some in tags;  
And some in velvet gowns.*

—Nursery Rhyme, *Hark! Hark!*



Now that you have been introduced to JSP coding and development in JDeveloper, it is useful to examine how to work with a tag library. Since BC4J is an important feature of JDeveloper, this book concentrates on the BC4J Data Tags Library, which is built to maximize efficiency when programming JSP pages that access BC4J objects. The BC4J Data Tags Library consists of a set of tags that use BC4J objects as their data source. The tags are written so that you can easily bind them to existing BC4J objects. This chapter provides an overview of the tags and gives details about how to work with them in JDeveloper.

The bulk of this chapter is devoted to letting you experience the development methods for these tags in the hands-on practices. The practices create JSP pages that use the higher-level component tags. The JSP pages you develop are similar to those created by the JDeveloper Data Page Wizard.

Chapter 23 discusses other techniques for modifying the defaults and working with the tags. Chapter 24 examines the tags in more detail and concentrates on some of the low-level tags.

## Introduction to the BC4J Data Tags Library

This section briefly introduces the BC4J Data Tags Library. The other tag libraries offer different tags, but the methods for creating applications are similar, as is the support in JDeveloper's wizards, Component Palette, Code Editor, and Embedded OC4J Server. The BC4J Data Tags Library consists of a number of tags that can be categorized into the following types:

**Connection Tags** As with Java applications and applets, the BC4J application module forms the connection point from the user interface components to the BC4J view usages. An `ApplicationModule` tag is used to specify the BC4J application module that will be used for the connection to the database. The `DataSource` tag provides a declaration of the view usage for a particular component.

**Database Operation Tags** These tags call transaction operations in the database (Commit, Rollback, and PostChanges).

**Data Access and Presentation Tags** These tags cause a loop through rows (`RowSetIterate`) or through all attributes in the view object (`AttributeIterate`). Some tags show values of view object attributes on the page (`ShowValue` and `RenderValue`). You can use `SetAttribute` to update values or to insert a value into a new record.

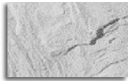
**Form Element Tags** Tags for presenting HTML form elements are connected to view usages to make the form elements data aware. For example, tags such as `InputText`, `InputTextArea`, `InputDate`, and `InputLOVSelect` present an HTML field that is linked to a view object attribute.

**Presentation Form Component Tags** BC4J Component tags connect to a view usage in the application module and present fields or values from a row in that view usage. For example, the DataEdit tag presents an edit form with a field for each attribute in the view object. Components are available for DataTable (a multi-record display) and DataQuery (for a set of query fields). In addition, there are navigation components (DataScroller and DataNavigate).

The components package a large amount of functionality into a single tag. Another file, a *component JSP file*, which is generated when you enter the data component tag, contains the code that implements the logic for these tags. This concept is explained further in the section “Working with Data Tag Component JSP Files” in Chapter 23.

**Other Tags** The library contains tags to help you embed interMedia objects (such as audio, video, or images) into the page that allow you to update and insert these objects in the database.

The WebBean and DataWebBean objects in the Component Palette are available for backwards compatibility with previous versions of JDeveloper. WebBeans present non-data-aware controls or displays. DataWebBeans are connected to BC4J objects and contain the same kind of functionality as the component tags. However, the DataWebBean requires more Java code in the JSP file. Oracle recommends using the component tags for new applications instead of the WebBean and DataWebBean tags, because component tags are simpler to use for default functionality and do not require scriptlet and method calls as do the WebBean and DataWebBean tags.



**NOTE**

*The BC4J data tags are well documented in the JDeveloper online documentation, and you can refer to that source for complete details about each tag. To access this documentation, look for the topic “Reference: BC4J Data Tags Library” after clicking “BC4J data tags” on the Index tab. This topic is also available under the node “Developing Web Applications\Working with JSP Pages for Business Components” in the Contents tab.*

## Development Methods Using the BC4J Data Tags Library

You can develop JSP pages in JDeveloper in several ways by using the BC4J Data Tags Library. Although you can certainly hand-code JSP pages by typing into the Code Editor, this method is not as productive or as easy as starting with a skeleton page, as shown in this chapter. Regardless of the method you use to create the code, you will need to write custom code to fulfill application-specific requirements. It is always good to watch for code that seems to recur throughout the application. That code may be a candidate for inclusion in a common “include” file, custom tag library, or even a modification to the JDeveloper wizards.

### General Development Steps

The general sequence for developing a JSP application in JDeveloper follows:

- I. Create a BC4J project or open an existing BC4J project in a workspace.

2. Add an empty project that will contain the JSP page to the same workspace.
3. On the JSP project node, select New from the right-click menu (or use **File | New**) to run a JDeveloper wizard or dialogs (described later in the section “JSP Wizards and Dialogs”) to create code that you can use as a starting point.
4. Click Save All to save the files.
5. Compile the JSP project using Rebuild <projectname> in the toolbar. Although the next step would compile any uncompiled files by default, fully compiling the project now will allow you to examine warnings and errors in a more controlled way.
6. When the files compile without errors, run the file that starts the JSP application by selecting it in the Navigator (or active Code Editor window) and clicking Run in the toolbar.
7. The Embedded OC4J Server will start, and the page will load in your default browser. Watch the messages that appear in the Log window, indicating the port numbers that are assigned and the URL that the server is calling.

**Additional Information:** The following listing shows an example of these messages with line numbers added. Code without numbers would be written on the same code line as the line above it.

```

1: C:\JDev9i\jdk\bin\javaw.exe -ojvm -classpath C:\JDev9i\j2ee\home\oc4j.jar
   -Doracle.j2ee.dont.use.memory.archive=true com.evermind.server.OC4JServer
   -config C:\JDev9i\jdev\system\oc4j-config\server.xml
2: [Starting OC4J using the following ports: HTTP=8989,RMI=23892,JMS=9228.]
3: [waiting for the server to complete its initialization...]
4: Embedded OC4J startup time: 4927 ms.
5: Oracle9iAS (9.0.3.0.0) Containers for J2EE initialized
6: Target URL --
   http://localhost:8989/DataPageJSP/CountriesView1_StarterPage.jsp

```

Line 1 shows the full runtime command for the OC4J server. Lines 2–5 display the status of the server with the port numbers assigned to it. Once the ports have been initialized, you can access them from any browser session while the Embedded OC4J Server is active, even if it is started outside of JDeveloper or on another machine on the same network.

8. Test the JSP operations and verify that the database actions have been committed or rolled back appropriately. Leave the browser open if you want to make modifications.
9. If necessary, modify the appearance of the JSP page. Use an HTML editor or make changes to the cascading style sheet to address common look-and-feel issues.
10. Add other data tags if needed, and test the file using the Run button again to check the modifications. Alternatively, if the embedded server is still active, you can compile the files and run the files by typing the name in the URL location field in the browser.
11. Add other JSP pages to the project, and provide links between the files.
12. Test the new files by running them using the Run toolbar icon.

13. When all development and testing is complete, prepare and deploy the package as described in Chapter 7.

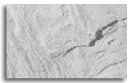
**TIP**

*Changes to the BC4J objects may be cached in the server and might not be available to the project until you restart the Embedded OC4J Server. You can stop the server by selecting Terminate from the right-click menu on the server name in the Run Manager window. If the Run Manager window is not displayed, select **View | Run Manager**.*

## JSP Wizards and Dialogs

The wizards and dialogs that create JSP files are started by selecting items under the Web Tier node in the New gallery. A number of categories under the Web Tier node will create different types of files. The JavaServer Pages (JSP) node contains wizards or dialogs that will create generic JSP files (not specific to any tag library or framework). In that node, the JSP Page item creates a standard JSP page with basic HTML tags and some default text. The JSP Document item creates a JSP 1.2 document that will contain XML-style code. The JSP Page and JSP Document display a dialog that only requires a name. There are also wizards in that node for creating Web Beans and tag libraries.

The BC4J data tags are supported by the JSP for Business Components category under the Web Tier node. This category contains options for creating a full application or for creating data pages. There are similar nodes under the Web Tier for other types of application server code. The BC4J data tag wizards require a bit of explanation.

**CAUTION**

*The exact location and names of these wizards may change over time, so explore the nodes in the New gallery if you do not find the locations or names used in this book.*

### Business Components JSP Application Wizard

The New gallery item Complete JSP Application (in the Web Tier\JSP for Business Components category) runs the Business Components JSP Application Wizard (also called the BC4J JSP Application Wizard). The wizard requires minimal input: the client data model definition that specifies the application module, and view object and view link usages for which you need pages in the application. You can specify whether you want browse, query, and edit pages for each view usage. You can also specify whether you want a page for each view link. The wizard creates a fully functional application, as shown in the hands-on practice in Chapter 1.

For production systems, you will need to go beyond the defaults, and modify the code that the wizard creates. You can modify the cascading style sheet, JSP file component and HTML tags, and the component JSP files to change the behavior and appearance of the application.

The application wizard provides the “quick and dirty” application that may serve to demonstrate BC4J object editing, data administration, and even as a starting point for a more user-friendly application.

## Data Page Wizard

The Data Page Wizard is triggered from the New gallery items Browse & Edit Form, Browse Form, Query Form, and Starter Data Page (in the JSP for Business Components category). The wizard will create a page (or pages) that connects to a single view usage. All modes of the Data Page Wizard copy a number of supporting files (such as the cascading style sheet file) into the project and create complete, working pages. The pages that the Data Page Wizard creates are similar to those that the Business Components JSP Application Wizard creates.

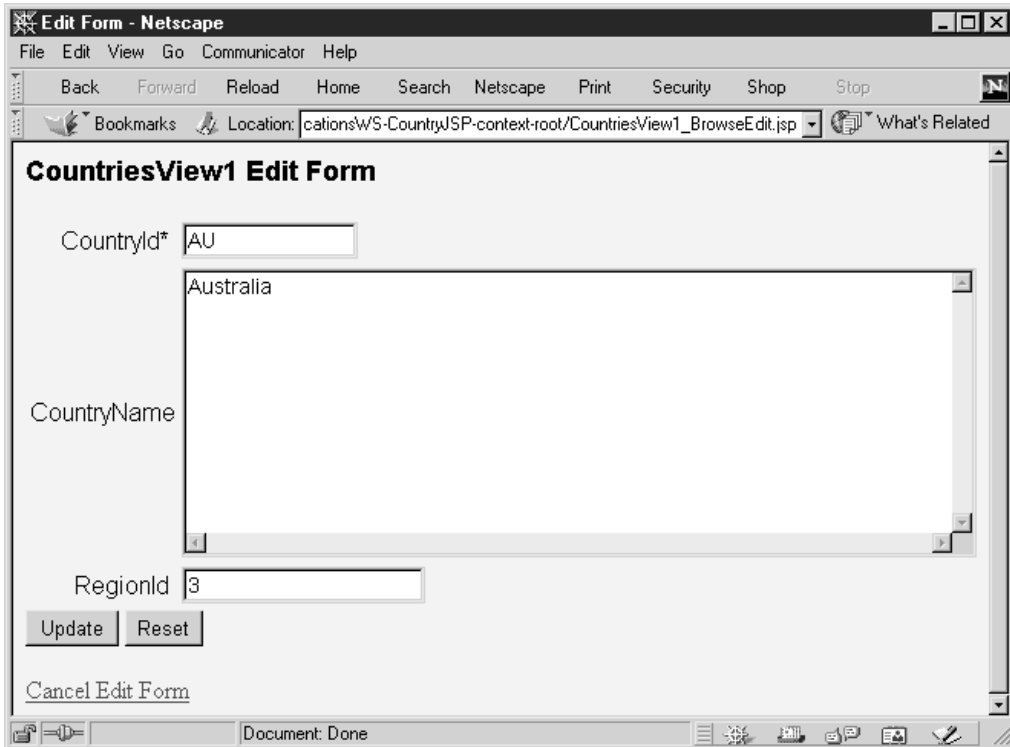
This wizard gives you a more controlled way to build an application. Each page is complete but, as with the Business Components JSP Application Wizard, you will need to modify its look-and-feel to fulfill your application requirements. You can build pages around the output from this wizard, or embed the pages that the wizard creates inside other pages.

Each of the modes of the wizard creates a page with the following characteristics.

**Browse & Edit Form** The Browse & Edit Form selection creates a browse page such as the following that displays rows from a view usage in multi-record, grid table form with scroller links (Previous and Next).



Each row has a link for Delete and Edit. The Delete function deletes the row. The Edit function displays an edit form page that contains editable fields loaded with the values from that row such as the following:



The edit page returns to the browse page, where you can commit or rollback the changes. The browse page also contains a New link that loads the same edit form (with empty fields) where the user can enter a new row.

**Browse Form** The Browse Form selection creates the same browse form without New, Edit, and Delete links. The browse form has scroller links for Previous and Next but no Commit and Rollback links.

**Query Form** The Query Form selection creates a form with fields where the user can enter query criteria and search for matches. After the user clicks Search, a data table appears at the bottom of the page with the results as shown here:

**DepartmentsView1 Query Definition**

[Query Form Usage](#)

DepartmentId

DepartmentName

ManagerId

LocationId

[Delete](#)

**DepartmentsView1 Query Results**

Previous  Next

	New	DepartmentId	DepartmentName	ManagerId	LocationId
Delete	Edit	10	Administration	200	1700
Delete	Edit	110	Accounting	205	1700

By default, there are no Edit, Delete, or New links on the browse area, but, if you create an edit page (or run the Data Page Wizard in Browse & Edit Form mode), you can link that page to the DataTable component by specifying the name of the edit form in the DataTable attribute *edittarget*.

**Starter Data Page** The Starter Data Page selection creates a page containing ApplicationModule and DataSource tags but with no specific data component tag. Therefore, if you run this JSP page, you will see nothing on the page. The purpose of this page is to give you a starting point for entering data component tags that is farther along than the file, which the generic JSP Page dialog creates.

**TIP**

*It is useful to examine the code that the Data Page Wizard and Business Components JSP Application Wizard create to determine how JSP pages are constructed with these tags.*

## Working with BC4J Data Tags in the Code Editor

Although you can type the code for BC4J data tags directly in the Code Editor, JDeveloper offers two tools that help ensure the correct spelling and syntax: the Component Palette and the JSP Data Binding tool.

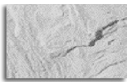
**TIP**

*As mentioned in Chapter 21, the Code Editor offers context-sensitive help for custom JSP tags supplied with JDeveloper. For example, if you place the cursor inside a BC4J tag (such as “<jbo:ApplicationModule>”) and press F1, the help topic for ApplicationModule will be displayed.*

### Component Palette

The Component Palette displays pages appropriate to JSP work when the topmost Code Editor window contains a JSP file. To activate a page, you select it from the pulldown list in the Component Palette bar. If the Component Palette is not visible, select **View | Component Palette**.

When you select a tag, a wizard will appear that steps you through filling out the required properties. When you click Finish, the wizard will enter the tag text at the cursor location in the Code Editor and will add attributes to the tag based on the properties that you completed in the wizard. You cannot run the wizard on an existing tag in the Code Editor. If you run the wizard again, you will have to fill in the tag properties again.

**TIP**

*If you want to replace a tag entry in the Code Editor, select the tag’s text and click the tag in the Component Palette. After you fill out the wizard properties and click Finish, the new tag will overwrite the selected text.*

**Component Palette Placement and Appearance** The Component Palette anchors by default on the right side of the IDE window. If you want more horizontal space in the Code Editor, you can drag the Component Palette to the top of the IDE (for example, under the Document Bar). This frees up the right side of the IDE so that you can expand the Code Editor. The Component Palette will appear with icons and names, as in the following example:



If you select Icon View from the right-click menu on the Component Palette, the names will be hidden, and you will just see the icons for each component, as in the following example:



If you have not memorized the meaning of the icons, this particular view of the Component Palette may slow down your development, so you may want to leave the names displayed. You can switch back to the name and icon display by selecting List View from the right-click menu on the Component Palette.



#### TIP

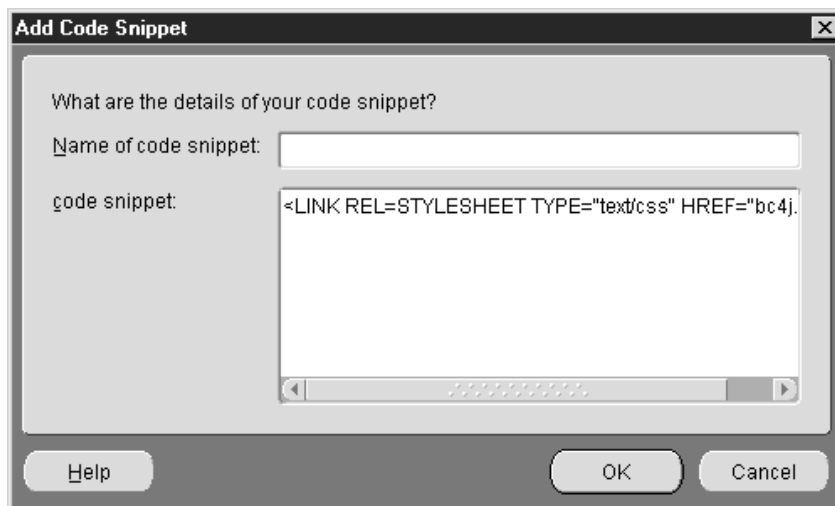
Remember that the Component Palette offers the ability to “pin” the buttons. If you hold the SHIFT key while clicking a button, the tag will stay selected so that you can add more than one without reselecting the button. To unpin the button, click the Pointer button.

**Code Snippets** One of the Component Palette pages available to JSP pages is the Code Snippets page. This page allows you to select code that you have stored and named. For example, although the link to a cascading style sheet is not available on the HTML page of the Component Palette, you can add this to the Code Snippets page using the following steps:

1. Highlight the following line in the Code Editor window for a JSP page that you generated using the Data Page Wizard (or just type the line into an existing JSP page):

```
<LINK REL=STYLESHEET TYPE="text/css" HREF="bc4j.css">
```

2. In the Code Editor, select Add Code Snippet from the right-click menu. The following dialog will appear:



3. Fill in the *Name of code snippet* field with the name, in this case “bc4j.css Reference.” The code snippet will be added to the Component Palette’s Code Snippets page.

To use a code snippet, place the cursor where you want the code to appear in the Code Editor, and click the code snippet’s button in the Component Palette. The code will be added to the file at the cursor position.

You can also create code snippets by selecting Add Component from the right-click menu on the Code Snippets page of the Component Palette. The right-click menu also contains selections for removing a component and displaying the properties of all components in all pages of the Component Palette. The properties dialog also allows you to modify the code in the code snippet.

### JSP Data Binding Tool

The JSP Data Binding tool (shown in Figure 22-1) was new with release 9.0.3. It provides another method for adding BC4J data tags to the JSP file. Since it bypasses the wizard pages, you may find



**FIGURE 22-1.** JSP Data Binding tool

it faster for entering tags once you know the required elements and are familiar with the purpose of the tag. The JSP Data Binding tool appears by default on the right side of the screen when you select **View | JSP Data Binding**. You can close this window using the close window (“X”) icon in the top right corner of the palette window.

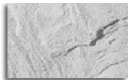
This palette contains the following panes:

- **Client Data Model** This pane provides a hierarchical view of all client data model definitions available to the project. You use this pane to browse the view usages and their attributes. When you select a node in this pane, the Control Type pane is loaded with the components appropriate to that node.
- **Control Type** This pane contains the appropriate component tags for the node selected in the Client Data Model pane. When you select a control type, the Properties pane changes to display the properties of that control.
- **Properties** This pane allows you to set values for the selected control’s properties.

There are buttons in the toolbar for creating a client data model definition, editing the BC4J object selected in the Client Data Model pane, creating view objects and view links, and help.

To add a tag to the JSP file, you click the appropriate object in the Client Data Model pane, select the tag in the Control Type pane, fill out the properties, and drag the component from the Control Type pane to the place in the Code Editor where you want the tag to appear. After you drag the tag into the Code Editor, the properties will remain in the properties window in case you want to add another tag with the same properties.

One advantage of the JSP Data Binding tool over the Component Palette is that it bypasses the wizard, which may save you time. This tool also allows you to interact with the BC4J data tags by starting with the application module element instead of starting with the tag. Another advantage is that you can drag the tag and drop it where you want. The tag does not need to be placed at the cursor location as it does with the Component Palette.



#### NOTE

*Although the Component Palette offers tags from all frameworks and tag libraries, the JSP Data Binding tool only provides support for the BC4J Data Tags Library.*

## Hands-on Practice: Build JSP Pages Using BC4J Data Tags

The hands-on practice in Chapter 1 demonstrates how to generate a complete BC4J JSP application using the JSP Application Wizard. Chapter 3 contains a hands-on practice that introduces the method for creating individual JSP pages using the BC4J Data Tags Library. Chapter 7 contains a brief practice for creating a JSP page using the Data Page Wizard. As discussed earlier in this chapter, those are some of the main ways of creating JSP pages in JDeveloper.

This practice expands on the Chapter 3 practice and digs deeper into the data component tags. It steps through creating browse and edit pages for the LocationsView data. It re-creates most of the functionality of the Browse and Edit Form JSP pages that are generated by the Data

Page Wizard. This practice will give you experience with the functionality of the BC4J Data Tags Library and how you can use them to build an application. It also provides information about modifying the default behavior and appearance of the default tags. The tags that you use in this practice are further described in Chapter 23.

This practice contains the following phases:

- I. Set up the workspace and projects**
- II. Create the browse page using the Component Palette**
  - Create a starter data page
  - Add the interface components
  - Modify the cascading style sheet
- III. Create an edit page using the JSP Data Binding tool**
  - Create a default JSP page
  - Modify the default JSP page

## I. Set Up the Workspace and Projects

This phase creates the workspace and projects in preparation for holding the JSP files. This phase also creates a default BC4J project and an empty JSP project in which you will place JSP files.

1. On the Workspaces node in the Navigator, select New Workspace from the right-click menu.
2. Change the *Directory Name* and the *File Name* fields to “LocationsWS.” Leave the check mark on the *Add a New Empty Project* checkbox, and click OK to create the workspace file.
3. In the New Project dialog, enter the *Directory Name* and *File Name* as “LocJSP.” Click OK to create the project file for the JSP files that you will add in the next phase.
4. Click Save All.
5. On the LocationsWS.jws node in the Navigator, select New Project from the right-click menu.
6. From the General\Project category in the New gallery, double click “Project Containing New Business Components.”
7. Click Next if the Welcome page appears. Fill in the *Directory Name* and *File Name* as “LocBC4J” and click Next.
8. On the Paths page, fill in the *Default Package* as “locbc4j,” and click Next and Finish to create the project. The Business Components Package Wizard will display.
9. Click Next if the Welcome page appears. Be sure that the *Package Name* is “locbc4j.” Click Next to display the Connection page.
10. Be sure the HR connection is selected in the Connection Name field and click Next.

11. On the Business Components page, select COUNTRIES, DEPARTMENTS, and LOCATIONS, and move them to the Selected pane using the right-arrow button. Click Finish. The wizard will create default BC4J objects in the new project.
12. Click Save All.

**What Just Happened?** This phase sets up the workspace and projects that you will use to house the JSP files. The BC4J project provides the application module and view objects that are required by the JSP pages as data sources. If you had a BC4J project with the same objects defined, you have opened that project in the new workspace instead of creating a new BC4J project. This practice modifies the BC4J objects; so, if you were using the project in another workspace, you would need to ensure that the changes you make would not adversely affect the projects in the other workspace.

## II. Create the Browse Page Using the Component Palette

This phase uses the BC4J Data Tags Library to create a page that browses the LocationsView data. It uses the Component Palette to enter the data component tags. The next phase uses the JSP Data Binding tool to enter the tags so that you can experience both techniques.

The main tasks in this phase are adding the data component tags, adding the user interface tags that access data, and modifying some of the styles in the cascading style sheet. These are typical tasks for JSP development.

### Create a Starter Data Page

The first tags in the file need to reference the application module and data source (view object) that the interface components require. The application module is contained in the BC4J project in this workspace. The data source is the locations view object usage within the application module. This phase takes a shortcut offered with release 9.0.3. Instead of using a new JSP Page from the New gallery and adding the data tags as in the Chapter 3 practice, this section uses the Data Page Wizard to create a starter data page that already contains the application module and data source tags. The Data Page Wizard also creates a number of other useful files.

1. On the LocJSP.jsp node, select New from the right-click menu.
2. In the Web Tier\JSP for Business Component category, double click "Starter Data Page." The Starter Data Page Wizard dialog will appear. Click Next if the Welcome page appears.
3. Click New to create a data model definition. Click Next if the Welcome page appears in the BC4J Client Data Model Definition Wizard.

**Additional Information:** The *BC4J client data model definition* (contained in the .cpx file) connects the client interface code (in the JSP page) to the BC4J application module. This definition is discussed further in Chapter 17. This definition is required for JSP as well as Java client applications.

4. Select "LocBC4J.jsp" in the *Business Components Project* pulldown if it is not already selected. Click Next.

5. Click Next in the Definition Name page to accept the default name. Click Finish to create the data definition.
6. Select “LocationsView1” and click Next to display the Finish page. Click Next and Finish to create the starter data page JSP page.

**Additional Information:** Notice that a number of other files appeared under the project along with the starter data page JSP page:

- **bc4j.css** This file contains the cascading style sheet that the starter data page JSP page references. The data component tags reference styles defined in this file.
- **errorpage.jsp** This file supplies error text inside the page that is being displayed so that the user can see the cause of any problems. The file is tailored specifically for BC4J application errors.
- **globalinclude.html** This HTML file contains instructions that will be compiled into the servlet for each JSP page in this project. It allows you to make application-wide changes in one place. By default, this file contains a scriptlet call to set the encoding scheme that defines the number of bytes used for each character sent to the browser. This is important if non-Western character sets need to be supported. You can add or modify this text if you need different instructions for all JSP pages in the project.
- **LocJSP\_jpr\_War.deploy** This file defines the deployment profile for the web application archive (WAR) file—an archive file (like a JAR file) that is generated by the deployment utilities and used to deploy the application to the server.
- **LocJSP.cpx** Many application modules could be defined in this workspace. This is the configuration file that contains the client data model definition. The client data model definition declares which application module is used for this project.
- **ojsp-global-include.xml** This file defines parameters used by the OC4J JSP server to modify the runtime. For example, the default version of this file contains a reference to the globalinclude.html file described earlier.
- **web.xml** This is a standard J2EE file known as the *Web Application Deployment Descriptor*. It provides parameters and other application-specific information to the application server.

Although some of these files are also generated by other methods such as the Business Components JSP Application Wizard, you would normally need to create or copy files such as the error page and cascading style sheet into the project directory. For example, if the Data Page Wizard had not copied the cascading style sheet, you would need to add it using **File | Import** and navigating to the `JDEV_HOME\jdev\system\templates\common\misc` directory (where `JDEV_HOME` is the JDeveloper installation directory, for example, `C:\JDev9i`). The `bc4j.css` file that the JDeveloper wizards copy is housed here. In production installations, you would probably want to use a central location for CSS files so that all JSP pages access the same styles. Similarly, the master `errorpage.jsp` file is located in the

JDEV\_HOME\jdev\system\templates\common>tagcomp directory along with the master component JSP files.

**TIP**

To add a file to the project, use **File | Import** to find and copy or reference the required file.

7. Click Save All. Select LocationsView1\_StarterPage.jsp and click Run.

**Additional Information:** The Embedded OC4J Server will run and display the page in your browser. You will not see anything but a background color and window title, but this test verifies that the basic file works and can read the BC4J application module.

You now need to rename the file that the wizard created. Renaming the file will allow you to run the Data Page Wizard again and not overwrite the customization in the file.

8. Click LocationsView1\_StarterPage.jsp and select **File | Rename**. Fill in the *File name* as "LocBrowse." (The .jsp extension will be added automatically.) Click Save.

**CAUTION**

If the Rename menu item is disabled, select the file again in the Navigator and retry the menu item.

9. Open LocBrowse.jsp in the Code Editor if it is not already open. You will see the application module and data source tags already entered by the Data Page Wizard.
10. Change the title between <title> tags to "Browse Locations."
11. Under the opening <body> tag, add the following heading:

```
<h2>Browse Locations</h2>
```

**Additional Information:** If you pause after typing "<h2>," End Tag Completion will add the ending tag, "</h2>."

12. Change the *rangesize* attribute of the DataSource tag to "6" and the id to "locData" so the tag reads as follows:

```
<jbo:DataSource id="locData" appid="Locbc4jModule" viewobject=
  "LocationsView1" rangesize="6"/>
```

**Additional Information:** The *rangesize* attribute specifies how many records are displayed in multi-record components such as DataTable. The id for this and other tags identifies the usage so that it can be referenced in other code. The Data Page Wizard defined an id of "ds," which does not help distinguish the view usage. If you have more than one view object in the JSP page, it is important to provide distinguishing ids for each one. If you required more than one view object in this JSP page, you would add a data source tag for each additional view usage.

**NOTE**

You can define more than one usage for a view object in your application module's data model. For example, *DepartmentsView* can appear as a master (top-level) usage as well as a detail usage (linked to a *LocationsView* usage). Each of these usages is different, so you need to define a separate data source for each usage if they will appear together within the same JSP page.

**Add the Interface Components**

The default starter data page contains tags for the application module and data source (view object) that this page will use. In this section, you add user interface components and transaction controls using the Component Palette.

1. The JSP Component Palette appears on the right side of the IDE by default. If the Component Palette is not visible, select **View | Component Palette** to display it.
2. Select the BC4J Component Tags page from the Component Palette pulldown. Click the cursor in a blank line after the `jbo:DataSource` tag, and click *DataHandler* in the Component Palette.

**Additional Information:** The *DataHandler* tag is required to coordinate the processing of events for a component such as a data scroller that has Next and Previous actions. You will add a data scroller in later steps.

3. Select "Locbc4jModule" in the appid pulldown as shown next:

Attribute Name	Value
*appid	Locbc4jModule
relativeUrlPath	DataHandlerComponent.jsp

Attributes prefixed with "\*" are required

4. Click Finish to add the data handler tag.
5. With the cursor in a blank line after the `jbo:DataHandler` tag, click `DataTransaction` in the Component Palette. Select "Locbc4jModule" in the appid pulldown and click Finish. The `DataTransaction` tag will be added to the file.

**Additional Information:** The `DataTransaction` tag adds Commit and Rollback buttons to the page so that the changes can be sent from the cache to the database or undone in the cache, respectively. Notice that a `DataTransactionComponent.jsp` file was also added to the project. If you open that file, you will see code that defines an HTML table with two text buttons. This is the file that actually implements the functionality of the `DataTransaction` tag. It is a partial JSP file that must be included in a complete JSP page so that the HTML tags are complete. Similar files will be added in the next steps as you add data component tags.

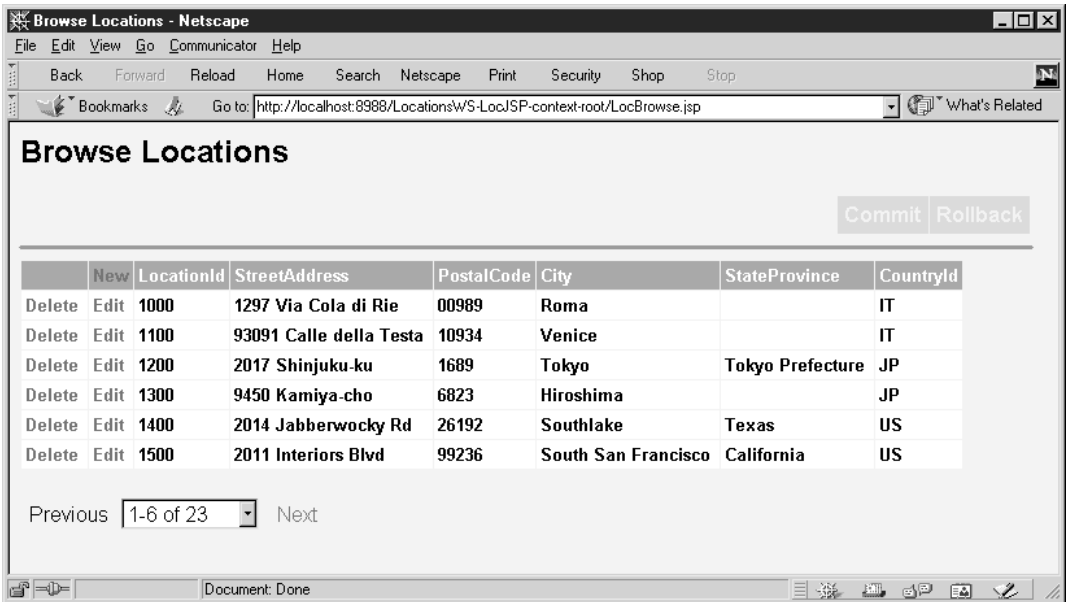
6. With the cursor in a blank line after the `jbo:DataTransaction` tag, click `DataTable` in the Component Palette.
7. In the `DataTable` dialog, select "locData" in the *datasource* field and the *edittarget* as "LocEdit.jsp." The `LocEdit.jsp` file will be built in subsequent phases. Click Finish to add the tag.

**Additional Information:** The `DataTable` tag presents rows from the view usage in an HTML table (rows and columns). The number of records in this table is controlled by the *rangesize* property of the data source tag. A `DataTableComponent.jsp` file will be added to the project.

8. With the cursor in a blank line under the `jbo:DataTable` tag, click `DataScroller` in the Component Palette.
9. Fill in the *datasource* as "locData," and click Finish to write the tag into the file.

**Additional Information:** The `DataScroller` component scrolls back and forth the number of records you specified in the *rangesize* property of the data source tag. You can add a `DataNavigate` component (in the BC4J Data Components pulldown of the Component Palette) that scrolls through the rows one at a time and highlights each row. However, that capability is not required for this JSP page.

10. Click Save All. Run `LocBrowse.jsp`. Something like the following will appear:



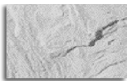
11. Test the scrolling capabilities by selecting from the record pulldown and by clicking the Previous and Next links. The pulldown and links are created by the DataScroller component.

12. Test the Delete link on one of the records, and click Rollback to restore the record.

13. Notice that the data table contains Delete and Edit links on each row. Hold the mouse cursor over an Edit link, and examine the URL in the browser status bar.

**Additional Information:** The page that the Edit link refers to is the text you filled in for the data table *edittarget* property ("LocEdit.jsp"). You will see a similar URL on the New link at the top of the data table.

14. Leave the browser open. You can make changes to the file and refresh the browser to see the effect of the change.

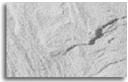


#### TIP

You may need to hold the SHIFT key when clicking Refresh or Reload in the browser to override any cached pages that the browser stores.

## Modify the Cascading Style Sheet

The cascading style sheet that the Data Page Wizard copied to this project (bc4j.css) is also used by other JDeveloper wizards. Suppose you decide that the font for the commit/rollback buttons is too large and that the color of the H2 heading tags should be similar to the heading bar of the data table. The steps that follow demonstrate how to find the style that you need to change and how to change the cascading style sheet.



### TIP

*Special editors are available to assist in creating and modifying cascading style sheets. These editors (such as TopStyle Lite from [www.bradsoft.com](http://www.bradsoft.com)) provide a visual display of the style and property lists that help you enter the correct syntax. Cascading style sheet editing is also available within some HTML editors.*

1. If you closed the browser, run LocBrowse.jsp again.
2. In a blank spot on the browser window (where there is no link text), select View Source from the right-click menu. Another window will appear containing the HTML text displayed in the browser.
3. Find the <h2> tag at the top of the file. Notice that there is no *class* attribute for that tag (that is, the code does not read “<h2 class=classname>”), which means that the style sheet contains an entry that modifies the <h2> tag without requiring the style name in the HTML code. You may have to scroll down in the file to see this text.
4. Search for the word “commit” by using CTRL-F. You will see it in a block of code such as the following that is created by the DataTransactionComponent.jsp:

```
<table class="clsToolBar" cellspacing="1" cellpadding="5"
border="0" width="100%">
<tr>
<td width="100%">&nbsp;</td>
<td class="clsToolBarButton">Commit</td>
<td class="clsToolBarButton">Rollback</td>
</tr>
</table>
```

5. This code defines an HTML table that contains commit and rollback text buttons. Notice that the name of the style that is applied to the “Commit” text is “clsToolBarButton.”
6. Close the view source window. Double click bc4j.css in the Navigator to open it in the Code Editor.
7. Search for “clsToolBarButton” in the file. You will find a block of code such as the following:

```
.clsToolBarButton
{
font-family:Arial, Helvetica, Geneva, sans-serif;
color:#EFEFEF;
font-size:13PT;
font-weight:bold;
```

```
background-color:#CCCCCC;
}
```

**Additional Information:** The period prefix indicates that “clsToolBarButton” is a *global* style, which can be applied to any tag by including an attribute in that tag for “class=clsToolBarButton,” for example, “<body class=clsToolBarButton>.”

8. Change the text “13PT” in the *font-size* property to “10PT.” This will reduce the font size by three points.
9. Search for the definition of the H2 style. It appears at the top of the file as a combined style definition for the H1, H2, H3, and TD tags. Add another style for H2 under this block that specifies the blue color (color number “336699”) used by clsTitleBody.
10. Modify this entry to add the blue color so the style definition appears as follows:

```
H1, H2, H3, TD
{
    font-family:Arial, sans-serif;
}

H2
{
    color: #336699;
}
```

**Additional Information:** Both H2 style entries will be used. The first defines the font family, and the second specifies the color.

11. Click Save All. Examine the changes by clicking Reload (for Netscape) or Refresh (Internet Explorer) to rerun the JSP page. Notice the changed heading color and smaller commit/rollback text.

**Additional Information:** You can also select Refresh or Reload from the View menu to reload the browser.

#### **NOTE**

*You do not need to rebuild the project or JSP pages, because these changes are applied to the cascading style sheet that is not compiled and is read dynamically by the JSP pages.*

12. Do not close the browser.

#### **CAUTION**

*Browsers such as Internet Explorer and Netscape do not always render a JSP or HTML page in the same way. Fonts and spacing may look different in different browsers. If your application must support different browsers, it is important to test the application in those browsers. The sidebar “Testing JSP Pages in a Nondefault Browser” describes how to run JSP Pages in different browsers so that you can test the code in different environments.*

## Testing JSP Pages in a Nondefault Browser

Running a JSP page within the JDeveloper IDE starts the Embedded OC4J Server and opens the browser that is set up as the default in your operating system. You can use the following steps to run another browser session with the same JSP page so that you can test the visual aspects. If you want to permanently change the default browser, follow the steps in the sidebar “Changing Your Default Browser.”

1. Run the JSP page from JDeveloper. Your default browser (such as Internet Explorer) will open with the JSP page loaded.
2. Copy the text from the URL *Address* (or *Location*) text field at the top of the browser.
3. Run a different browser (such as Netscape). It does not matter if the first browser is closed or open.
4. Paste the text you copied in step 2 into the *Address* or *Location* field of the different browser and press ENTER. The application will appear in the browser.

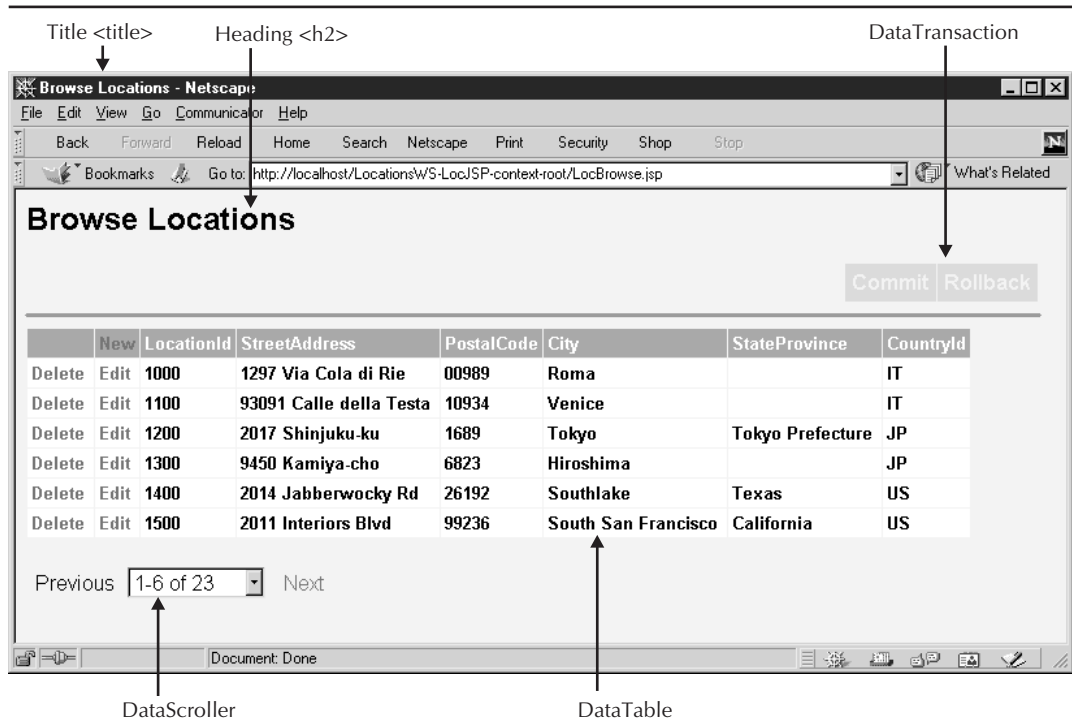
The virtual port (8988) for the OC4J server remains open once you start it. You can verify that it is running by looking in Run Manager (**View | Run Manager**). Run Manager allows you to terminate the server session (from the right-click menu or from **Run | Terminate | Embedded OC4J Server**). Once the server is started and the virtual port is open, you can open or close any browser (from any networked machine) and reconnect using the URL in the *Address* or *Location* field.

**What Just Happened?** You defined a complete browse page with record scrolling, transaction control, and links to an edit page. The pre-built functionality of the tags made this process relatively easy. If you want to check your work, the following should appear in the JSP page between the “</h2>” tag and the “</body>” tag:

```
<jbo:ApplicationModule id="Locbc4jModule" definition="LocJSP.Locbc4JModule"
  releasemode="Stateful" />
<jbo:DataSource id="locData" appid="Locbc4jModule" viewobject=
  "LocationsView1" rangesize="6"/>
<jbo:DataHandler appid="Locbc4jModule" />
<jbo:DataTransaction appid="Locbc4jModule" />
<jbo:DataTable datasource="locData" edittarget="LocEdit.jsp" />
<jbo:DataScroller datasource="locData" />
<jbo:ReleasePageResources />
```

Figure 22-2 shows the main visual elements in the JSP page and where they appear in the browser. In addition to the visible components, the JSP page requires the ApplicationModule, DataSource, and DataHandler component tags to support the other tags.

Your project now contains a dozen files even though you only selected one JSP file from the New gallery. In addition to the files described before that the Data Page Wizard generated, the editor generated separate JSP files for the DataHandler, DataScroller, DataTable, and DataTransaction component tags.

**FIGURE 22-2.** *Tags in the browse page*

### Changing Your Default Browser

If you want to change the browser that JDeveloper uses by default, you can specify the browser in the Preferences dialog (**Tools | Preferences**) Web Browser/Proxy page. Enter the executable name and path in the *Browser Command Line* field, and this browser will be used for all projects.

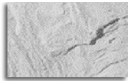
You can also change the browser that the operating system calls by default. In the Windows environments (Windows 95 or 98, Windows NT, Windows XP, and Windows 2000), the default browser is associated with the .html file type. Therefore, changing that association will change the browser that JDeveloper uses to open files in its Embedded OC4J Server if you have not set the preference mentioned before. In Windows operating systems, associating file types with a program is an operation of Windows Explorer (Tools

Options or View Options dialog). Consult the help files to determine how to change the file association.

Note that once the Embedded OC4J Server is started, you can open and close any browser and connect to the URL specified in the Log window. Therefore, if you just want to test another browser, but not change the default browser, you can open the other browser and copy the URL from the default browser or from the Log window into the other browser. You can also run a browser on another machine that is networked to your machine.

### III. Create an Edit Page Using the JSP Data Binding Tool

The browse page is completely functional for browsing and deleting records. You now need a page that will allow users to edit or insert Location records when they click Edit or New in the browse page, respectively. This phase creates an edit page that also allows inserting records. You use the JSP Data Binding tool to enter the tags. The preceding phase uses the Component Palette to enter the tags. At the end of this phase, you will be able to compare the two techniques.



#### NOTE

*The page flow model in BC4J data tag applications relies on HTML links between pages. As of this writing, a new effort called "JavaServer Faces" is under way at Sun Microsystems. JavaServer Faces is aimed at improving ease of development, integration with existing systems using APIs, and flexibility of page flow.*

#### Create a Default JSP Page

You will use the Data Page Wizard to create a starter data page.

1. On the LocJSP.jpr node, select New from the right-click menu. In the Web Tier\JSP for Business Components category, double click Starter Data Page. The Data Page Wizard will appear.
2. Click Next if the Welcome page appears. Select LocationsView1 and click Next and Finish.
3. The Data Page Wizard will repeat the process of creating the starter data page and all of its supporting files. Since the supporting files are already created, you will see one or more confirmation dialogs that ask if you want to overwrite the existing files. Click No in each of these dialogs.

## Modify the Default JSP Page

This section adds the components needed to present the edit page.

1. Select the `LocationsView1_StarterPage.jsp` file and select **File | Rename**. In the *File name* field, enter “LocEdit” (the `.jsp` extension is optional) and click Save.
2. Open `LocEdit.jsp` in a Code Editor window. Change the data source *id* property to “locData.”
3. Change the title to “Edit Location.” Add a heading 2 tag under the `<body>` tag for “Edit Location” (“`<h2>Edit Location</h2>`”).
4. If the JSP Data Binding tool is not open (in a tab next to the Component Palette), open it by selecting **View | JSP Data Binding**.
5. Click `LocationsView1` in the Client Data Model pane. The *Control Type* pane will display a list of tags appropriate to a view object as shown in Figure 22-3.
6. Click the `DataEdit` component in the *Control Type* pane. The properties for the `DataEdit` data component tag will display in the Properties pane as shown in Figure 22-3. Select “locData” in the *datasource* property pulldown. If there is no pulldown, type in “locData.”
7. Drag the `DataEdit` component from the *Control Type* pane to the Code Editor. The cursor will change and show the position that the tag will take when you release the mouse button as shown here:

```

10 <h2>Edit Location</h2>
11 <jbo:ApplicationModule id="am" definition="LocJSP.Locbc4JMo
12 <jbo:DataSource id="locData" appid="am" viewobject="Locatio:
13 |
14 |
15 <jbo:ReleasePageResources />
16 </body>
17 </html>

```

8. Drop the control under the data source tag. The tag will appear in the Code Editor and the `DataEdit` component JSP file added to the project.



### NOTE

If the tag is simple, you can also just type it into the editor. The component JSP file will be added when you complete the tag (with “`>`”).



**FIGURE 22-3.** *JSP Data Binding tool*

9. Click Save All. Click Rebuild LocJSP.jpr.
10. If you closed the browser, click LocBrowse and click Run. If the browser still has LocBrowse running, click Reload or Refresh in the browser toolbar.  
**Additional Information:** The LocBrowse page supplies the parameters to the LocEdit page, so you need to run it first.
11. When the new browse page is displayed, click Edit on Location number 1000. The edit page such as that shown below opens with the record for 1000 loaded. Make a change to this record (by adding a province name, for example).

The screenshot shows a Netscape browser window titled "Edit Location - Netscape". The address bar contains the URL: `Location: 000EF2BB5AB8A&aml=d=am&originURL=/Location/WS-LocJSP-context-root/LocBrowse.jsp`. The main content area displays a form titled "Edit Location" with the following fields and values:

- LocationId\*: 1000
- StreetAddress: 1297 Via Cola di Rie
- PostalCode: 00989
- City\*: Roma
- StateProvince: (empty)
- CountryId: IT

At the bottom of the form, there are two buttons: "Update" and "Reset". The browser's status bar at the bottom indicates "Document: Done".

12. Click Reset to see that the form changes the values back to the retrieved values. Make the change again and click Update. The form returns to the LocBrowse page and refreshes with the modified value.

**Additional Information:** Although you can see the new values in the browse page, they are not yet committed. The Commit and Rollback buttons are now enabled, which indicates a noncommitted change. If you click Rollback, the change you made on the previous page will be reversed. If you click Commit, the change will be saved to the database.

13. Click Rollback to reverse the change.

**What Just Happened?** You defined an edit page using the BC4J Data Component Tags. You started by running the Data Page Wizard to create a starter data page into which you placed the DataEdit component tag. This tag presents a set of edit fields for the attributes in a view object. The edit page is called from the browse page, and context information about which record needs to be edited is passed to it. The page can also be called in insert mode for new records.

## Hands-on Practice: Build Query and Details Pages

The pages that you created in the preceding practice emulate the functionality that the Data Page Wizard creates for Browse and Edit pages. It would also be useful to have a query page so that the user can find a record by entering some query conditions instead of having to scroll through a potentially large number of records.

This practice creates a query page that emulates the page that the Data Page Wizard creates for the Query Form. The practice also creates a link from the query page to a new details page that shows all attributes of the record. This practice uses the same workspace and BC4J project as the preceding practice.

The practice steps through the following phases:

### I. Create a query page

- Create the query fields
- Add a display for the results
- Modify the component JSP file
- Modify the page using BC4J properties

### II. Add a details page

## I. Create a Query Page

A *query page* allows the user to enter values in fields and click a Search button to execute a query that uses the values as query criteria. This phase creates the query page by adding tags to a starter data page. The resulting page emulates the Query Form page that the Data Page Wizard creates. Normally, the Data Page Wizard would be a faster development method, but this phase will show you how this kind of page is built from scratch and give you practice using the BC4J data tags.

### Create the Query Fields

Since you had experience in the preceding practice with adding component tags using the JSP Data Binding tool and the Component Palette, you can choose either of (or both) those techniques for this section. The abbreviated steps here will present the task and allow you to accomplish it using either technique:

1. If you did not finish Phase I of the preceding practice that creates the BC4J project, return to that practice and complete the BC4J project.
2. Use the Data Page Wizard to create a starter data page for LocationsView1 in the LocationsWS workspace and LocJSP project. Rename the starter data page (using **File | Rename**) to "LocQuery.jsp."

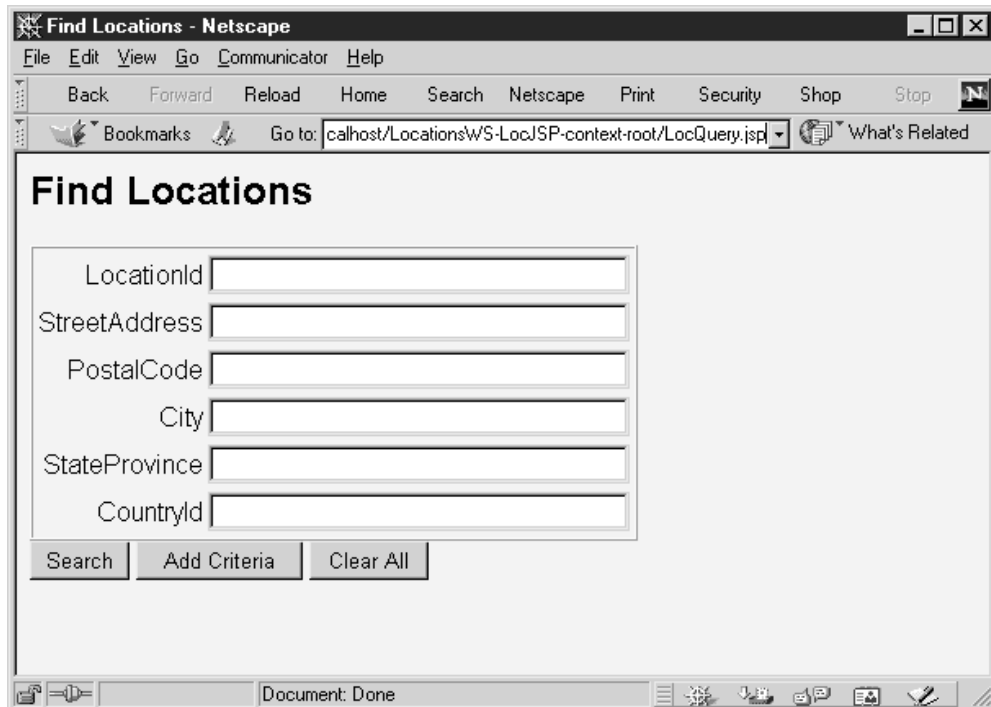
**Additional Information:** The Starter Data Page is an option in the New gallery (**File | New**) Web Tier\JSP for Business Components category. Refer to the steps in the

preceding practice if you need further help with details. Be sure to rename the file and to change the *id* of the data source tag to “locData” to make the data source reference clearer (in case you add other data sources to your JSP page later).

3. Run `LocQuery.jsp` to ensure that your starting file is correct. You will not see any data, but will see a window title.
4. In the Code Editor, change the *rangesize* property of the data source to “6.” This property defines how many rows will be displayed on the screen for a multi-record control such as `DataTable`.
5. Change the title to “Find Locations,” and add heading 2 text at the beginning of the HTML body for “Find Locations.”
6. After the `DataSource` tag, add a `DataHandler` tag and specify “`Locbc4jModule`” for the *appid*.

**Additional Information:** This tag points to the application module instance that was created in the `ApplicationModule` tag. The `DataHandler` is required to process events for database transactions and for scrolling.

7. Add a `DataQuery` tag with a *datasource* property of “locData.” This component presents a form with fields for each view object attribute and buttons for search operations.
8. Make and run this file. You should see something like the following:



**Additional Information:** The buttons in this JSP page will not function until you complete the next section. However, running the JSP page at this stage gives you a sense of what the query form looks like and ensures you that the code compiles.

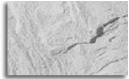
9. Leave the browser open.

### Add a Display for the Results

This section adds code to the component JSP file so that you can display the results of the search on the same page. It adds the same DataTable component that you used in the preceding practice to create the browse page.

1. Under the DataQuery tag, add an OnEvent tag (from the BC4J Events page of the Component Palette). Specify the *datasource* as "locData" and the *list* as "Search, FirstSet, NextSet, PreviousSet, LastSet, GotoSet." If you are using the Component Palette, click Finish to dismiss the dialog and add the tag.

**Additional Information:** The OnEvent tag provides the ability to add or modify elements on the page when an event such as navigation or a database transaction statement occurs. It is a conditional tag that will only execute for the events that you specify in the *list* attribute.



#### CAUTION

*You may have to change the event list if you are using JSP pages created by the Data Page Wizard in earlier versions of JDeveloper 9.0.3, because the GotoSet event is spelled as "Goto."*

2. Move the `</jbo:OnEvent>` closing tag to a new line, and on a blank line before that tag, enter a heading 2 for "Query Results" ("`<h2>Query Results</h2>`"). When an event in the list occurs, this and all other code before the closing OnEvent tag will be processed.
3. Under the `<h2>` tag, add a DataScroller tag (if you are using the Component Palette, this is on the BC4J Component Tags page) with a *datasource* of "locData." This tag provides a Previous and Next link that allows the user to scroll through record sets in the DataTable.
4. Under the DataScroller tag, add a DataTable tag with the following properties:
  - datasource* of "locData"
  - edittarget* of "LocRecord.jsp"
  - relativeUrlPath* of "DataTableQueryComponent.jsp"

**Additional Information:** Specifying the *relativeUrlPath* with a file name that is different from the default will add a file with that name to the project. You can change this new file and not affect other DataTable component JSP files in the same project, because they point to the default file name. Specifying the *edittarget* property adds two columns to the data table grid that contain links—one for Edit and one for Delete. The Edit link runs the file specified in the *edittarget* attribute.

5. Click Finish. Make, save, and run (or refresh the browser for) the `LocQuery.jsp` file. When you click Search, a data table area will appear beneath the query fields as in Figure 22-4. When you click Clear All, the table will disappear and the query fields will clear. Try adding criteria and modifying the query conditions to see what the query mechanism will display.

The following is an example of the code between body tags of your JSP page. The application module name may be different in your case.

```
<h2>Find Locations</h2>
<jbo:ApplicationModule id="Locbc4jModule" definition="LocJSP.LocModule"
  releasemode="Stateful" />
<jbo:DataSource id="locData" appid="Locbc4jModule" viewobject="LocationsView1"
  rangesize="6"/>
<jbo:DataHandler appid="Locbc4jModule" />
<jbo:DataQuery datasource="locData" />
<jbo:OnEvent datasource="locData"
  list="Search, FirstSet, NextSet, PreviousSet, LastSet, GotoSet" >
<h2>Query Results</h2>
<jbo:DataScroller datasource="locData" />
<jbo:DataTable datasource="locData" edittarget="LocRecord.jsp"
  relativeUrlPath="DataTableQueryComponent.jsp" />
</jbo:OnEvent>
<jbo:ReleasePageResources />
```

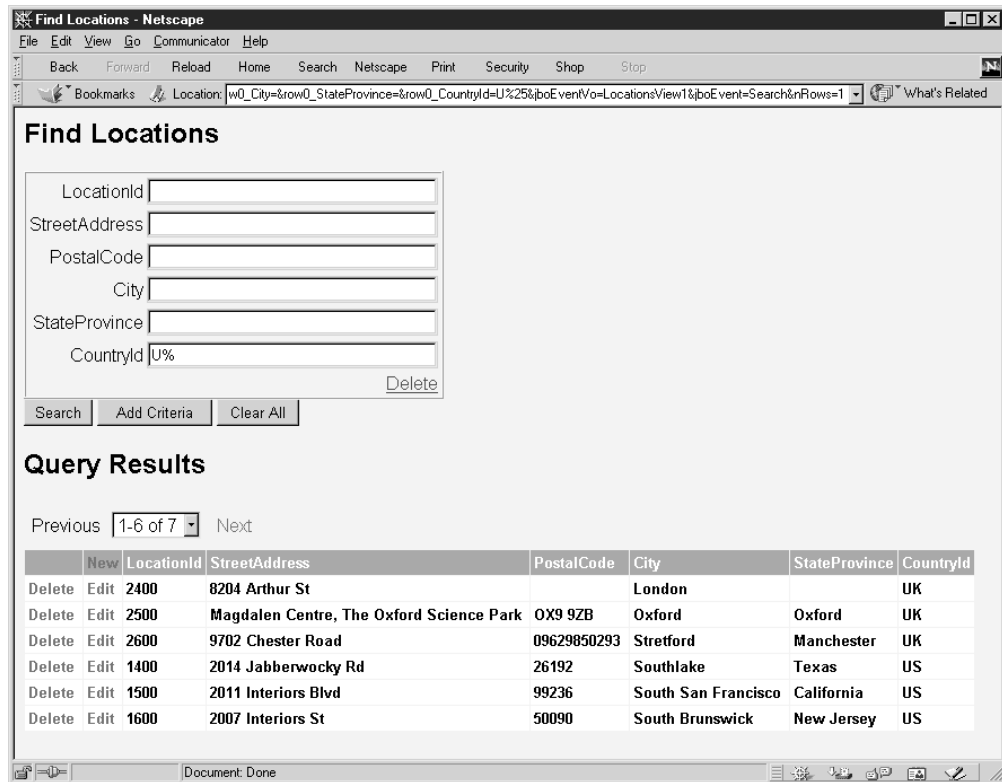
### Modify the Component JSP File

The *component JSP file* is a file created when you add a component tag to a JSP page. For example, adding the `DataQuery` tag to this JSP page added a component JSP file to the project called `DataQueryComponent.jsp`. This file contains low-level functionality that is called by the main JSP page. The file is a copy of a template in one of the JDeveloper install directories. You can modify your local copy to fit your requirements. In this case, the query results should not display the street and postal code attributes. Also, the results table should not contain Delete, New, and Edit links, but does require a Details link that loads a page containing all of the attributes for a particular record.

Making these kinds of changes requires a basic understanding of HTML. The View Source feature of the browser (accessed from the right-click menu of the browser page) will display the underlying HTML. You should be able to find HTML tags in the main JSP page or in the component JSP file that match the HTML in the browser. This will give you a clue about where to make these kinds of changes.

For the requirements of this section, you have studied the HTML source and decide to make the following changes:

- Remove the New link in the second column header.
- Remove the Delete link for each row.
- Specify that the street address and postal code be hidden.
- Change the Edit link to a Details link. You already specified the target as `LocDetails` so that the link will be built with the parameters for presenting the appropriate record.



**FIGURE 22-4.** Query page with results table

The steps here implement these changes:

1. Open the `DataTableQueryComponent.jsp` file in the Code Editor. This file contains the low-level tags that create the HTML table.
2. Find the first table header—a tag starting with “`<th`”. You will find something like the following:

```
<th class="clsTableHeader">&nbsp;&nbsp;&nbsp;</th>
<th class="clsTableHeader"><a href="<jbo:UrlEvent targeturlparam=
'edittarget' event='Create' datasource='dsBrowse' extraparameters=
'<%= "originURL=" + params.getParameter("originURL") %>' />"
New</a></th><&nbsp;&nbsp;&nbsp;
```

**Additional Information:** This code constructs a blank cell for the first column and a New link in the second cell. Since you only need one of these columns and do not need the one with the New link, you can delete that one.

3. Delete the line with the New link (shown on multiple lines in the earlier code) up to the opening scriptlet tag “<%”. The resulting code should appear as follows:

```
<th class="clsTableHeader">&nbsp;&nbsp;&nbsp;</th>
<%
```

4. Look for the `AttributeIterate` starting tag (“<jbo:AttributeIterate ... />”) just below this area that displays the attribute headings. Add a `hideattributes` attribute to this tag so that it looks like the following (on one line):

```
<jbo:AttributeIterate id="df" datasource="dsBrowse"
  hideattributes="StreetAddress, PostalCode">
```

**Additional Information:** The `AttributeIterate` tag normally steps through all BC4J view attributes in the view usage. The `hideattributes` attribute defines the BC4J view attributes that the `AttributeIterate` tag will skip.

5. Add the same `hideattributes` attribute to the other starting `jbo:AttributeIterate` tag that appears toward the end of the file. This tag displays the row values in the table.
6. Make the file and reload the browser (if the browser is still open with the `LocQuery.jsp`), or run the file (if the browser is displaying another page or is not open).

**Additional Information:** You should see a heading row that has three missing column headers. The next steps will remove the row values for the three deleted column headings.

7. Look for the table data (“<td”) code. You will find something like the following:

```
<td> <a href="<jbo:UrlEvent targeturlparam='originURL' event='Delete'
  datasource='dsBrowse' addrowkey='true' />">Delete</a> </td>

<td> <a href="<jbo:UrlEvent targeturlparam='edittarget' event='Edit'
  datasource='dsBrowse' addrowkey='true' extraparameters='
  <%= "originURL=" + params.getParameter("originURL") %>' />">Edit</a>
</td><%
```

8. Delete the first row that contains the “Delete” event. (This spans multiple lines in the example just shown.) This will remove the column for Delete in the table.
9. Change the text at the end of the remaining tag from “>Edit</a>” to “>Details</a>”.

10. Make the file. Run it by refreshing the browser or by clicking Run if you closed the browser. The results area should appear as follows:

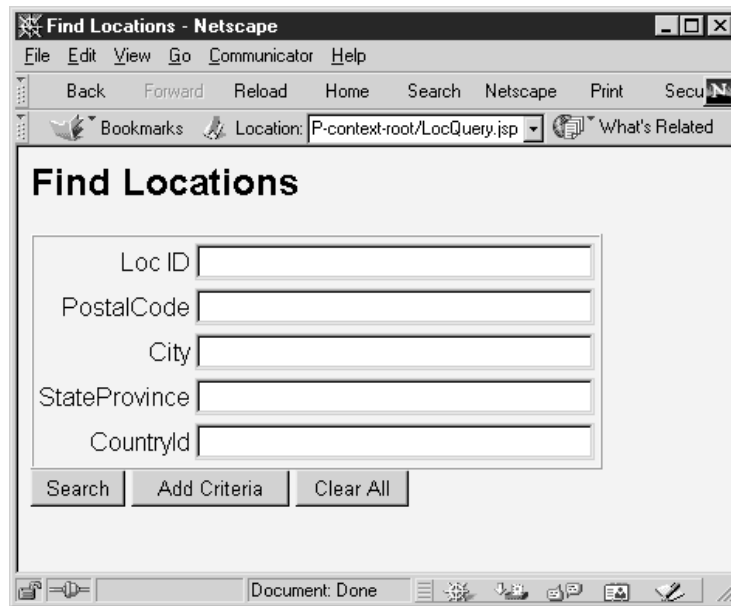
	LocationId	City	StateProvince	CountryId
Details	1000	Roma		IT
Details	1100	Venice		IT
Details	1200	Tokyo	Tokyo Prefecture	JP
Details	1300	Hiroshima		JP
Details	1400	Southlake	Texas	US
Details	1500	South San Francisco	California	US

11. Leave the browser and OC4J server running.

### Modify the Page Using BC4J Properties

Querying the street address attribute could cause slow performance because there is no index for the STREET\_ADDRESS column in the table. Therefore, you do not want to allow the user to search using the address attribute. You also want to change the default label for this field to "Street."

1. Under the LocBC4J.jpr project in the Navigator, select the LocationsView node.
2. In the Structure window, double click the StreetAddress attribute to display the Attribute Wizard.
3. On the View Attribute page, uncheck the *Queryable* checkbox. When this property is not checked, the attribute cannot be used for queries. Therefore, the DataEdit component will not draw it.
4. Click the Control Hints node in the Attribute Wizard navigator, and change the *Label Text* to "Street." This text will be shown instead of the attribute name for data components that read the control hint.
5. Click OK. Open the Attribute Wizard for LocationID and change the Label Text to "LocID."
6. Click Save All and Rebuild LocBC4J.jsp. Close the browser and terminate the Embedded OC4J Server session.
7. Select LocQuery.jsp and click Run. The browser will show the modified query form without the *Street* field and with a modified LocationId prompt as shown in the following image:



**What Just Happened?** You created a complete query page that allows users to search for records based upon values that they enter. You also modified the component JSP code to remove some default columns and to link to a page that you will create next. Modifying the JSP page is a matter of studying the HTML that needs to be changed and finding that HTML in the main JSP page or component JSP file.

The modifications that you made to the code occurred in two general locations: BC4J attribute properties and JSP component code. Another location for visual modification is the cascading style sheet. Chapter 23 discusses other ways to modify BC4J data tags JSP pages.

## II. Add a Details Page

Often, tables contain more columns than you want to display on a query form. The user needs to view data from all columns in the entire row after finding it. This phase uses the `DataRecord` component to display a record with all columns. `DataRecord` shows a form view of a row (instead of the table grid view that `DataTable` displays). As before, the steps for creating the JSP page are abbreviated, and you can refer to the preceding practice if you need more details.

1. Create another starter data page in the `LocJSP` project for the `LocationsView1` view usage. Rename the file to "LocRecord.jsp."
2. Change the `id` property of the `DataSource` object to "locData."

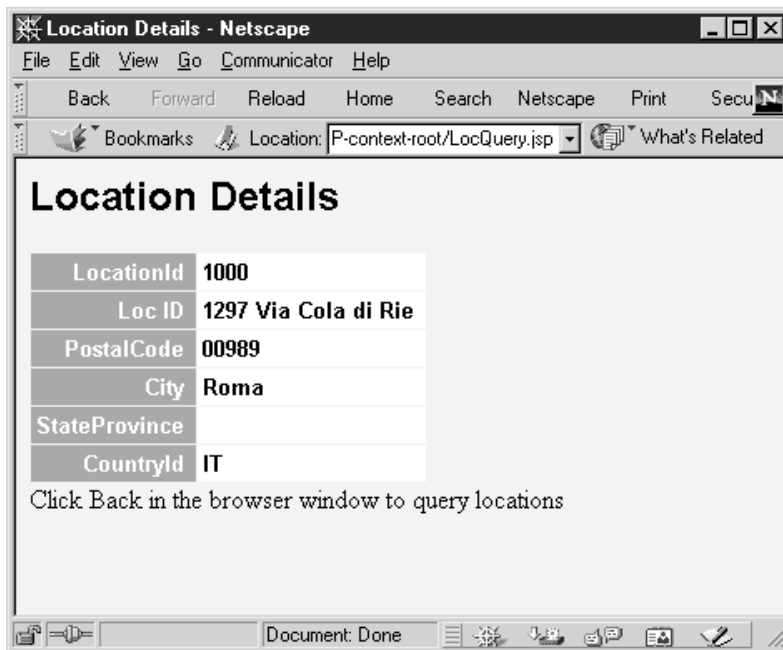
3. Change the title to “Location Details,” and add an h2 header in the HTML body with the text “Location Details.”

**Additional Information:** To try the End Tag Completion feature, pause after typing “<h2>”; the End Tag Completion feature will fill in the “</h2>” tag.

4. Add a DataRecord component (from the BC4J Component Tags page of the Component Palette) under the DataSource tag. Specify a *datasource* of “locData.”
5. Add the following line under the DataRecord tag with the following text. This text informs the user about how to return to the query page:

Click Back in the browser window to query locations

6. Click Save All and run (or refresh the browser for) LocQuery.jsp. Click on a Details link, and you will see something like the following:



This page shows all attributes for a record, but has no insert, update, or delete features.

**What Just Happened?** You used the DataRecord component to construct a page that displays a single record. This kind of page is useful if you have a table with many columns that needs a full page to display all details.