



Display dynamic information on Web pages with JavaServer Faces

Contents

Display dynamic information on Web pages with JavaServer Faces 1

Display dynamic information on Web pages with JavaServer Faces	1
Module 1: Create Web pages with data connections	4
Lesson 1.1: Import the required resources and setting the target server.	5
Lesson 1.2: Connect to a database and display data on a Web page	6
Lesson 1.3: Test the Web site.	13
Lesson 1.4: Create a new record to display and update the database	14

Lesson 1.5: Program the Submit button	18
Lesson 1.6: Create an update page.	21
Module 1: Summary	25
Module 2: Add advanced features	26
Lesson 2.1: Format a data table	30
Lesson 2.2: Use the file upload component	37
Lesson 2.3: Use navigation rules	39
Lesson 2.4: Use automatic key generation	41
Module 2: Summary	43
Display dynamic information on Web pages with JavaServer Faces.	44

Display dynamic information on Web pages with JavaServer Faces

This tutorial teaches you how to design a dynamic Web site that functions as a newspaper's classified advertising section. The Web site is a complete J2EE Web application that uses JavaServer Faces (JSF) components and Service Data Objects (SDO). This application uses JSF technology to create dynamic Web pages that link to a database that stores classified advertising data. By dragging JSF components onto your pages, you can create a JSF Web application known as a CRUD application because it can create, read, update, and delete records from a database. Site users can use these functions to manage the classified ads in the database.

Learning objectives

This tutorial is divided into two modules, each with its own learning objectives. You can choose to complete either or both modules. Within each module, you must complete the exercises in order for the Web site to work properly.

Module 1: Create Web pages with data connections

This module teaches you how to set up a connection to a database and display the information from the database on a Web page. In this module, you will:

- Connect Web pages to a database.
- Create pages that display, create, edit, and delete records from the database.
- Send data from one page to another

Module 2: Add advanced features

This module teaches you more powerful ways to use data from a database. In this module, you will:

- Format database records on a Web page
- Add a component that allows uploading files to a database from a Web page
- Navigate from page to page automatically
- Automate some database administration tasks

When you are ready, begin Module 1: Creating Web pages with data connections.

Time required

Total: 3 hours and 30 minutes

Module 1: 2 hours

Module 2: 1 hour and 30 minutes

Display dynamic information on Web pages with JavaServer Faces

This tutorial teaches you how to design a dynamic Web site that functions as a newspaper's classified advertising section. The Web site is a complete J2EE Web application that uses JavaServer Faces (JSF) components and Service Data Objects (SDO). This application uses JSF technology to create dynamic Web pages that link to a database that stores classified advertising data. By dragging JSF components onto

your pages, you can create a JSF Web application known as a CRUD application because it can create, read, update, and delete records from a database. Site users can use these functions to manage the classified ads in the database.

This tutorial might require some optionally installable components. To ensure that you have installed the appropriate optional components, see the System requirements list.

Upon completing this tutorial, site users can not only view items for sale, but also add new items, change details about the items (for example the price or description), or search for a specific item. While this tutorial site is simply designed, the principles and technologies it covers are also used in much larger and more complicated Web sites.

Learning objectives

This tutorial is divided into two modules, each with its own learning objectives. You can choose to complete either or both modules. Within each module, you must complete the exercises in order for the Web site to work properly.

Module 1: Create Web pages with data connections

This module teaches you how to set up a connection to a database and display the information from the database on a Web page. In this module, you will:

- Connect Web pages to a database.
- Create pages that display, create, edit, and delete records from the database.
- Send data from one page to another

Module 2: Add advanced features

This module teaches you more powerful ways to use data from a database. In this module, you will:

- Format database records on a Web page
- Add a component that allows uploading files to a database from a Web page
- Navigate from page to page automatically
- Automate some database administration tasks

When you are ready, begin Module 1: Creating Web pages with data connections.

Time required

Total: 3 hours and 30 minutes

Module 1: 2 hours

Module 2: 1 hour and 30 minutes

Skill level

Intermediate

Audience

Web application developers, Web user interface designers

System requirements

To complete this tutorial, you must first install and configure a runtime server. This tutorial has been tested with the following servers:

- WebSphere® Application Server 6.1

For information on deploying SDOs on servers other than WebSphere Application Servers, refer to Help topic: Deploying SDOs on non-WAS servers.

This tutorial might require some capabilities that have not been initiated during the installation of the product. If you cannot find user interface options described in the tutorial, ensure that you have enabled the appropriate capabilities (**Window** → **Preferences** → **Capabilities**). In order to complete this tutorial, you will need the following capabilities:

- Java™ Developer
- Web Developer (typical)
- XML Developer
- Database Developer
- Enterprise Java Developer
- Data

To use this tutorial, you must have an application server installed and configured. To verify that a server runtime environment is available, click **Window** → **Preferences**, expand **Server**, and then click **Installed Runtimes**. You can use this pane to add, remove, or edit installed server runtime definitions. You can also download and install support for a new server.

Prerequisites

To complete this tutorial, you should be familiar with:

- Basic Web design concepts, such as Web sites, Web pages, browsers, and servers.
- How to create a simple static Web page.
- The elements of a Web page, such as tables, hyperlinks, forms, and images.
- Database terms, such as tables, records, columns, and fields.

It will also help if you understand:

- How to use the perspectives and views of the workbench.
- How to edit the HTML code for a Web page.

Expected results

Once the Web application is complete, the home page will look like the picture below. The Web site navigation that is added to the Web pages via the site template, links to pages that allow you to create a new advertisement or view all of the advertisements. The row action in the table displayed below, brings you to an update page where you can change the information associated with a particular advertisement.

[Home](#)

[View all listings »](#)

[Post a listing](#)

View all listings

Title	Details	Category
W/V Passat wagon 	Description: 2000 good condition Price: \$1,000.00 Phone: 416-512-8665	Automotive
Magic Flute 	Description: Imagine it. Price: \$101.00 Phone: 314-159-265	Entertainment
Video Camera 	Description: Great camera, suitable for amateur or professional use. Price: \$900.00 Phone: 534-333-454	Entertainment
VCR 	Description: Old and worn but still working ok, hence low price. Price: \$40.00 Phone: 322-654-044	Entertainment
Flat Screen TV 	Description: Very clear picture, sound excellent too. Reluctant sale. Price: \$700.00 Phone: 456-456-944	Entertainment
1 2 3 4 5		

Related information

[View the PDF version of this tutorial](#)

Module 1: Create Web pages with data connections

In this module, you will learn how to create Web pages that allow you to display, edit, delete, and create information from a database. You will learn about JavaServer Faces (JSF) technology and use it to make complex programming operations simple.

Learning objectives

This module teaches you how to set up a connection to a database and display the information from the database on a Web page. In this module, you will:

- Connect Web pages to a database.
- Create pages that display, create, edit, and delete records from the database.
- Send data from one page to another

Time required

This module should take approximately 2 hours to complete. If you decide to explore other facets of dynamic Web sites while working on the tutorial, it could take longer to finish.

Lesson 1.1: Import the required resources and setting the target server

Before you begin, you need to import the required resources for this tutorial: a set of Web pages and a sample Derby database.

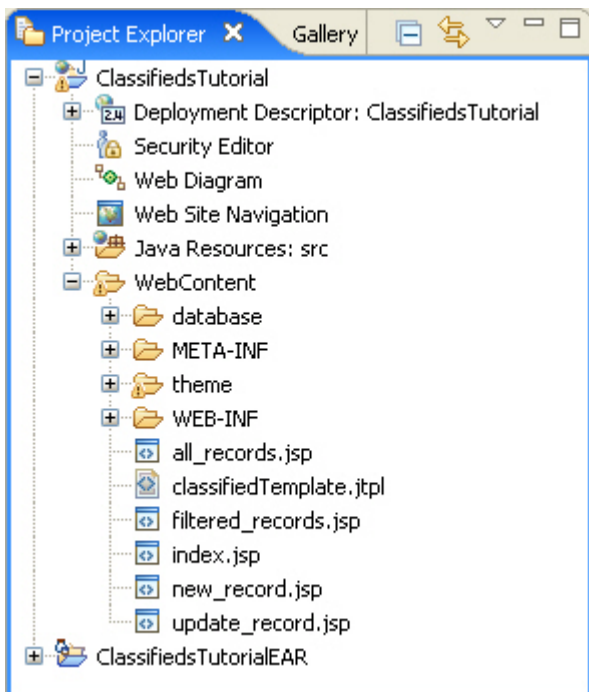
The main purpose of this tutorial is to teach you how to create a Web application that enables users to access and manipulate data in a database. The tutorial does not go into the detail of designing the look and feel of a Web site. Accordingly, the Web site design has been prepared for you already.

To complete this tutorial you will need to access an existing set of JavaServer Faces JSP files and a sample Derby database. These foundational Web pages and sample database are included in a ZIP file. This lesson will take you through the steps of importing the ZIP file so you can use the Web pages and the database. You will also set the target server for the dynamic Web project.

The Web pages and database that you will use for this tutorial are included in a ZIP file. To import the content of the ZIP file:

Import the sample project file

1. Import the project. Switch to the Web perspective (**Window** → **Open Perspective** → **Web**).
2. In the Project Explorer view of the Web perspective, ensure that your ClassifiedsTutorial project looks like the following image:



Set the target server for the dynamic Web project

Setting the target server for the Web project enables you to test the resources that you will be creating in this tutorial.

To set the target server:

1. In the Project Explorer view of the Web perspective, right-click **ClassifiedsTutorial** and select **Properties**.
2. In the properties list, click **Server**.
3. In the Default server list, select the server that you want to use as the default. Click **Apply**.
4. In the properties list, click **Targeted Runtimes**.
5. In the Runtimes list, click the runtime that corresponds with the server that you selected. Click **OK**.

If the server that you want to use is not listed in the target runtime list, close the workbench and install the server that you want to use. Once the server is installed, follow the instructions to set the target server.

Note: If you do not see any servers listed in the Default server list, and you have installed server runtimes, it is possible the server needs to be configured. To configure a server, you can do the following:

1. Right-click **all_records.jsp** file, then select click **Run As → Run on Server**.
2. Choose **Manually define a new server**.
3. Select a server you have installed.
4. Follow the directions in the wizard to configure the server. The first time you run on server you may receive an error. To fix the error set the target server as described above, restart the server in the Servers view, and reload the Web page in the browser.

If you go back through the previously described steps for setting a target server, you will now find the default server is the one you have just configured. If a server is installed but not configured, it will not show up in the list of servers from which you can choose a default target.

For information on deploying SDOs on servers other than WebSphere Application Servers, refer to Help topic: Deploying SDOs on non-WAS servers.

Lesson checkpoint

You have now imported the **ClassifiedsTutorial** dynamic Web project and set the target server.

You can browse the files in the tutorial Web project. To open a file, double-click it in the Project Explorer view. To view a map-like representation of how the pages are related, double-click Web Site Navigation in the Project Explorer.

The majority of your work in this sample will involve the following files:

all_records.jsp

This is the site's home page. It will display every classified ad in the database.

new_record.jsp

This page will create a new classified ad.

update_record.jsp

This page will change the details about an ad in the database or delete it.

classifiedTemplate.jtpl

This is the template for the site pages. It includes elements like the table and the gray "Welcome to the Classifieds" banner that are on every page. This page also has two navigational tabs below the gray banner that lead to the home page and the new classified ads page.

Now you are ready to begin Exercise 1.2: Working with the relational record list and data table components.

Lesson 1.2: Connect to a database and display data on a Web page

In this lesson, you will learn how to connect to a database and display data records on a Web page. You will also learn how to add a relational records list to your Web page.

The Web site in this tutorial uses dynamic Web pages with JavaServer Faces components to access a database and display information on the page. This tutorial uses relational records and relational record lists to represent the data in a database so the data can be displayed on the page in the form of a data table or an ordinary HTML table. Java access beans are used by these components.

Learn more about data access beans:

Data access beans are Java bean representations of enterprise beans. They are typically used in client programs that employ JavaServer Pages (JSP) files, servlets, or enterprise beans that interface with other enterprise beans. Access beans shield you from the complexities of managing enterprise bean life cycles. This means that you can program to enterprise beans as easily as you can program to Java beans, which simplifies your enterprise bean client programs and reduces your overall development time.

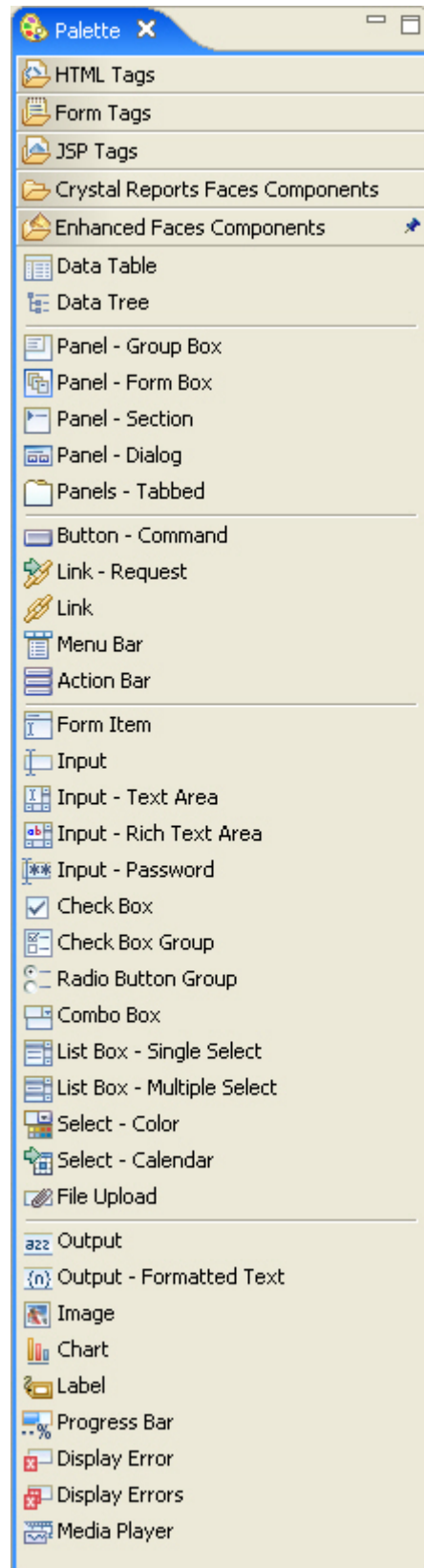
Learn more about JavaServer Faces and Faces components:

JavaServer Faces is a technology that helps you build user interfaces for dynamic Web applications that run on a server. The JavaServer Faces framework manages UI state across server requests and offers a simple model for the development of server-side events that are activated by the client. JavaServer Faces is based on a model-view-controller (MVC) framework. For JavaServer Faces, this means that the controller is a servlet, the model is represented by JavaBeans™, and the view is comprised of JavaServer Faces components with little or no application code. The goal of this model is to separate content from presentation.

Tools such as Faces Components to help you use this technology in your Web applications. Faces Components include a JavaServer Page (JSP) custom tag library for expressing a JavaServer Faces interface within a page. This wizard helps you create JSP files that are enabled to use the Faces components. Faces components let you develop a Web application by dragging components from a Faces drawer in the Palette view and dropping them on the Web pages you are creating.

For example, you can drag an input text field and drop it to a form on the page. Then you can drag and drop a Submit button next to the input text field. Finally, you can connect the input text field to your data source. This will enable end users to enter data from the Web application to your data source.

Another benefit of applications built using Faces components is that the user interface is rendered independently of the underlying program logic. This means that your applications can run and access data on different platforms, such as browsers or handheld devices.



Add a relational records list

In this lesson you will create a relational record list to represent all of the classified ads in the database. Then, you will connect to the database and select the table that holds the information you need in the relational record list. Finally, you will display this relational record list on the page in a data table.

- Relational records connect to only one record from a database. In this case, a relational record represents a single classified ad from the sample database. Using a relational record, you can create a new record, edit an existing record, or delete an old record.
- Relational record lists connect to more than one record from a database. In this case, a relational record list represents anywhere from two to all of the classified ads in the sample database.
- Data tables display the data from a relational record list on the page. Data tables simply designate a place for the record lists; they do not format the data into rows and columns in the same way that an HTML table does.

Learn more about data tables:

The data table is a component in a Faces JSP page that holds data objects such as a relational record list. Though a data table appears to have rows and columns like an HTML table, it does not work like an HTML table. If you wish to format input and output controls as in a table, you must use a Panel - Group Box from the Enhanced Faces Components drawer in the Palette. This is covered in Exercise 2.2: Formatting a Data Table.

Because data tables are Faces components and not HTML components, they are controlled through the Properties view, not through the Page Designer view. Using the Properties view, you can customize a data table in a variety of ways:

- Add, remove, and change the order of columns
- Format text and background
- Add header, footer, and margins
- Add paging for results display

To create a new relational record list:

1. In the Project Explorer view of the Web perspective, expand **ClassifiedsTutorial** → **WebContent**.
2. In the WebContent folder, double-click **all_records.jsp**. The `all_records.jsp` file opens in the editor.
3. Delete the default text Default content of bodyarea.
4. In the Palette view, expand the **Data drawer**.
5. Drag the **Relational Record List** component from the Palette onto the blank content area. The Add Relational Record List window opens.
6. In the Name field, type `all_recordlist`. Relational record list and relational record names must conform to Java standard naming conventions for variable names (for example, they cannot contain any spaces).
7. Make sure that **Add data controls** is selected. When **Add data controls** is selected, the wizard creates a data table to display the record list on the page. Otherwise, the wizard only creates the record list and not a representation of that data on the page. For now, you will have the wizard create the default data table and you will customize it later. The Add Relational Record List window should look like this:

Add Relational Record List

Relational Record List

Enter a reference name for record list.

Name:

Create a name to refer to this record list within the page.

☐ Retrieve an existing record or record list from scope

Scope :

Key :

☐ Reuse metadata definition from an existing record or record list

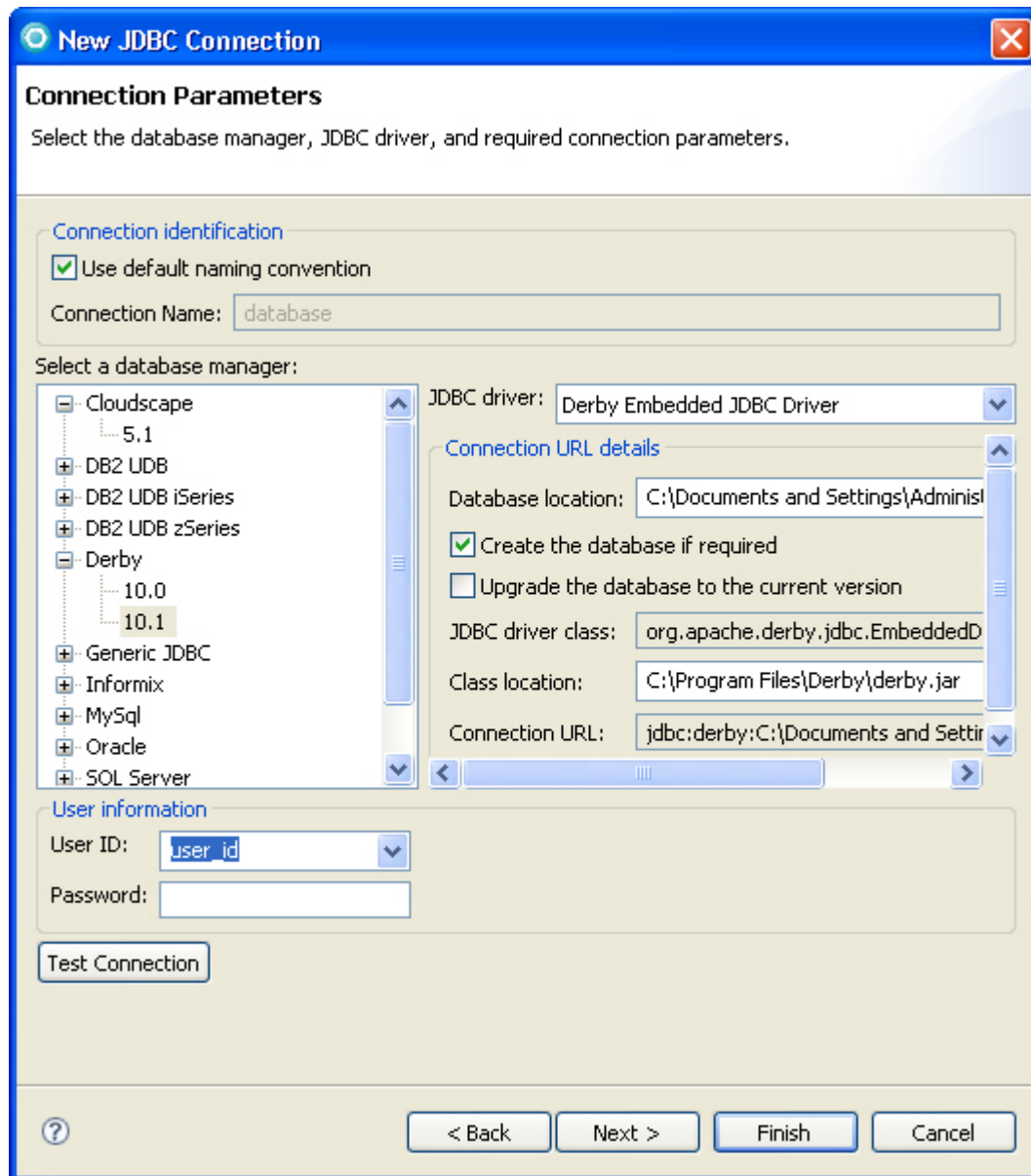
Input file:

You can automatically add data controls to your page to display the list of records (configure details on last page of this wizard).

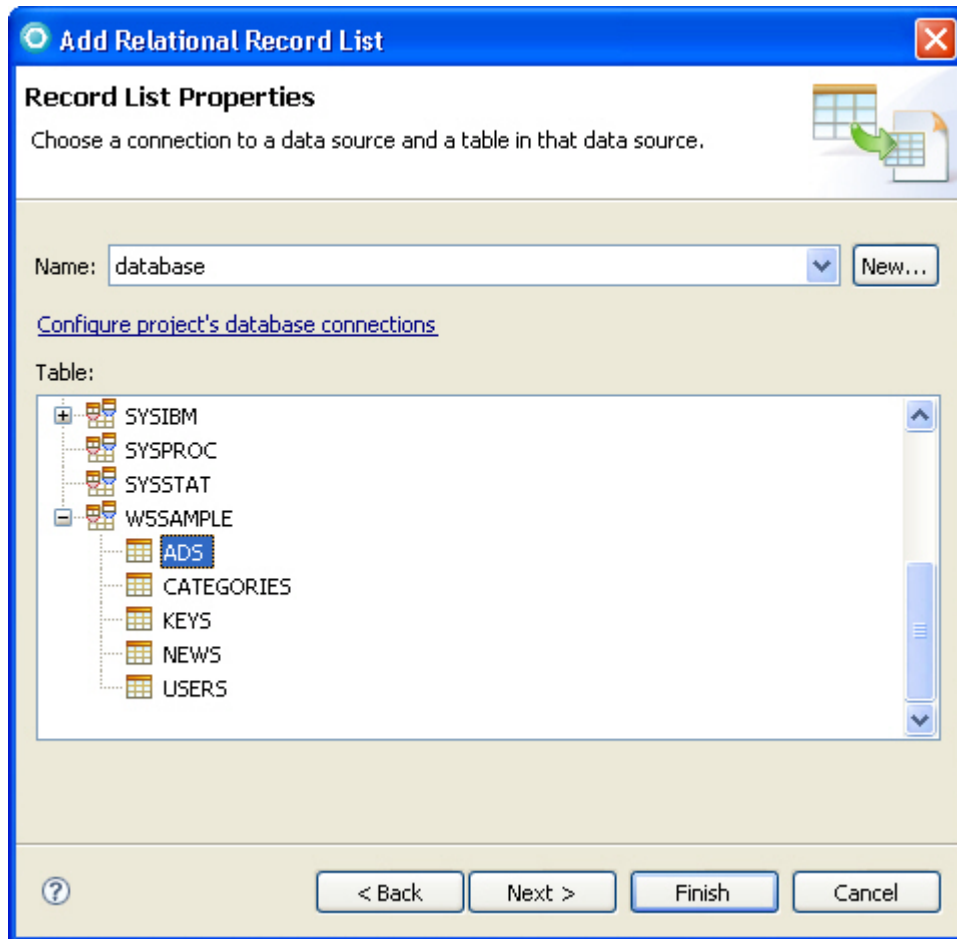
☒ Add data controls

Click **Next**.

8. In the **Connection name** field on the next page of the wizard, click **New** to create a new database connection. The New JDBC Connection dialog opens. Select the **Create a new connection** radio button then click **Next**.
9. In the **Select a database manager** list, expand **Derby**, then select **10.1**.
10. In the Database location field, click **Browse** and select `<workspace_location>\ClassifiedsTutorial\WebContent\\database`, where `<workspace_location>` is the directory of your current workspace. Click **OK** to close the Browse For Folder window.
11. In the **Class location field**, type in the location of the JAR or JARs that implement the JDBC driver class, or click **Browse** to select the location from your file system. Once you have chosen the location of the JAR files, click **Open**.
12. You may need to enter a User ID to access the database. A password is not required. The New Database Connection wizard should look like this:

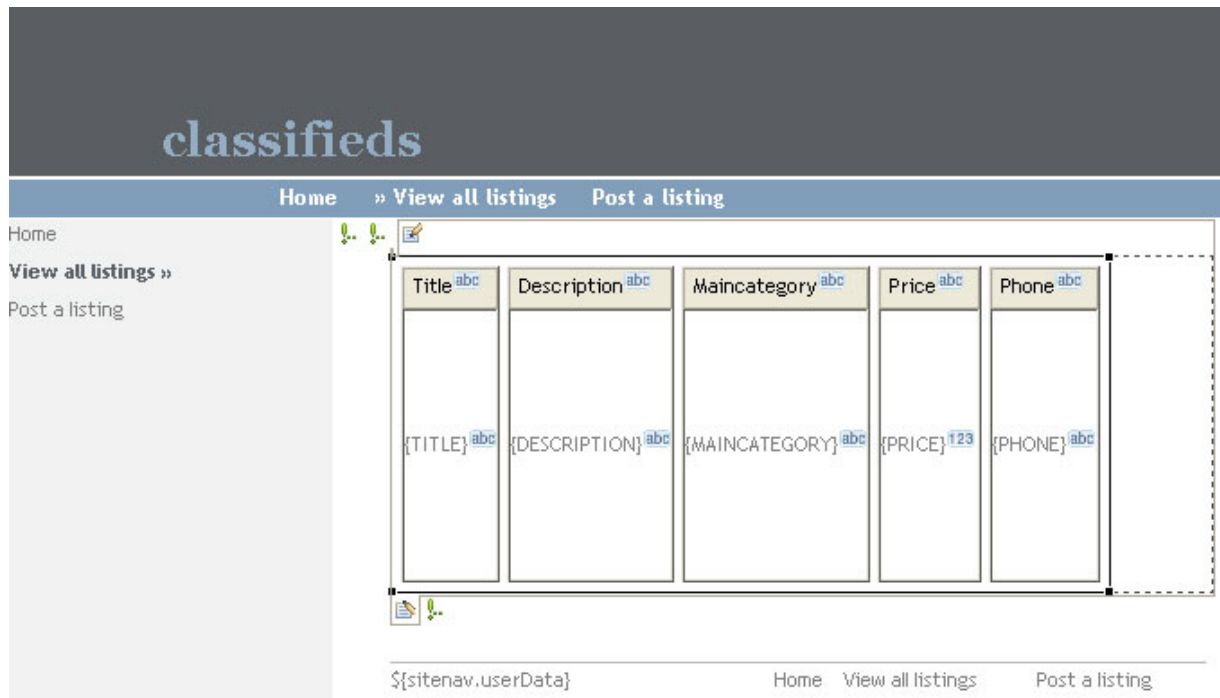


13. In the New JDBC Connection wizard, click **Finish**. You return to the Add Relational Record List wizard.
14. Click **Configure project's database connections**. The JDBC Connections properties window opens.
15. Click **Edit** next to the Runtime connection details section. Select **Use driver manager connection** as the runtime connection type. Click **Finish** then click **OK** to return to the Add Relational Record List dialog. Now that you have created a connection to the Derby database, you need to choose a table or the record list to represent. The Add Relational Record List wizard shows the tables in the database.
16. Expand **W5SAMPLE** and select **ADS**. Click **Next**. The remaining pages in the wizard let you exclude columns from your record list and perform advanced options, such as defining the primary key, adding relationships to other tables, and specifying filter and sort conditions. You will learn more about these pages in later exercises.



17. In the Add Relational Record page of the dialog, you will specify the columns that you want to include in your relational record list. Click **Next**.
18. For this tutorial you need to show only the title, description, category, price, and phone number for each classified ad. By default, all columns are selected; click to clear all but the following columns for display:
 - **MAINCATEGORY: String**
 - **TITLE: String**
 - **DESCRIPTION: String**
 - **PRICE: BigDecimal**
 - **PHONE: String**
19. You can refine the organization of the data table display on your Web page. To reorder the columns, select a column and click the up and down arrows until the column is in the desired position. Arrange the columns in the following order:
 - **TITLE**
 - **DESCRIPTION**
 - **MAINCATEGORY**
 - **PRICE**
 - **PHONE**

Click **Finish**. Your Web page now looks like this:



20. Save the page.

In the next lesson, you will see how this page will look on a real Web server. There are many options for formatting data tables and JavaServer Faces components. Some of these options are covered in the next module. You can also explore the Properties view on your own for the various JavaServer Faces components on the page (for example, the data table and the individual output components).

Lesson checkpoint

You have completed Lesson 1.2. In this lesson, you learned how to connect to a database and display data records on a Web page using a relational records list.

Lesson 1.3: Test the Web site

When you are ready to publish your Web application, you will need a server that will host it so users can access the Web site through the Internet; however, to test your Web site, you can use a test environment.

In this lesson, you will run a JSP in a test environment to simulate what a page will look like when it is actually published. You will also learn how to start and stop a server. At any time during your Web site development, you can open a page in Page Designer and use the **Preview** tab to see how your design will look in a browser. However, the Preview view does not allow you to see the dynamic aspects of your page (such as database connections) as they would display running from a server.

Tip: Once you have started the server, you must stop it before you can continue working with the site.

To use this tutorial, you must have an application server installed and configured. To verify that a server runtime environment is available, click **Window** → **Preferences**, expand **Server**, and then click **Installed Runtimes**. You can use this pane to add, remove, or edit installed server runtime definitions. You can also download and install support for a new server.

To test the Web site on a server:

1. In the Project Explorer view, right-click **all_records.jsp** file, then select click **Run As** → **Run on Server**. The Run On Server window opens

2. You must use the same server as the one you chose to be the target server in Lesson 1.1. Using a different server in this step will create errors.
 - If you want to test the Web site on a previously defined server configuration:
 - a. Select **Choose an existing server**.
 - b. Select the server that you want to use from the list available.
 - If you want to test the Web site on a new server configuration:
 - a. Select **Manually define a new server**.
 - b. Select the server type that you want to use from the list available.
3. Click **Finish**.

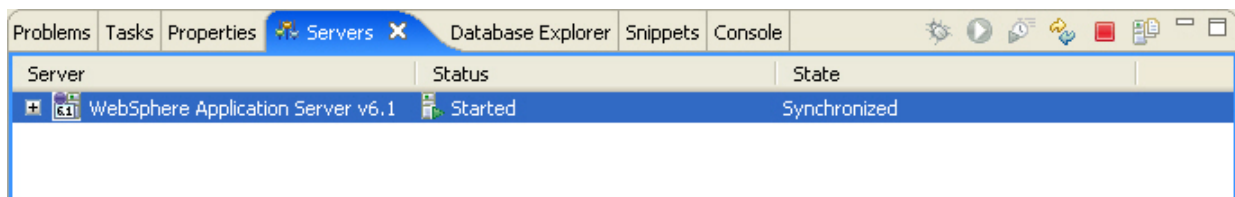
The server tools create the new server, start it, and open the page in the workbench internal Web browser. This may take a moment. In the Console view, you can watch the messages as the server tools start the server.

Once this process is complete, you can preview how the site will look and work once it is actually published to a Web server. Click the links, enter data in the forms, and see any dynamically generated content in the site that may not appear in Preview view. Also, if you wish to see your page in different Web browsers, you can copy the URL from the workbench Web browser into the browser of your choice.

You must stop the server before you can continue with the tutorial. If you leave the server running, you will see error messages when you continue with the tutorial. These errors appear because the server is connected to the sample database, preventing you from connecting to the database to change the information shown on the Web site.

To stop the server:

1. Open the Servers view. This view is usually located at the bottom center of the workbench. If you are not able to find the Servers view, click **Window → Show View → Servers** from the main menu.
2. Select the server that you want to stop. Notice that the server status is listed as *Started*.



3. Click the **Stop the server** button  at the top right of the Servers view. The server **Status** changes to *Stopping*. You will see messages in the Console view while the server shuts down. When the server Status in the Servers view changes to **Stopped**, the server is stopped and you can continue working on the Web site.

Tip: You can test your Web site on the server at any time, but remember to stop the server when you are finished.

Lesson checkpoint

You have completed Lesson 1.3. In this lesson, you tested a JSP in a test environment to simulate what your page will look like when it is actually published. You also learned how to start and stop a server.

Lesson 1.4: Create a new record to display and update the database

In this lesson, you will create a page that allows users to create new classified ads and post them to the database.

First you will create a relational record to represent a new record in the database, and then create a visualization of the record on your page as a set of input fields.

To create a new relational record:

1. Double-click the **new_record.jsp** file in the Project Explorer view to open it.
2. Delete the default text Default content of bodyarea.
3. In the Palette view, click the **Data** drawer to expand it.
4. Drag the **Relational Record** component from the Palette onto the blank content area. The Add Relational Record window opens.

Note: If a warning message appears and says that a connection to your database could not be established, then you left the server running after testing the Web site. If this happens, click **Cancel** in each dialog and stop the server as explained in Lesson 1.3.

5. In the **Name** field, type create_record.
6. Under **Create controls for**, click **Creating a new record**.
7. Make sure **Add input/output controls to display the record on the web page** is selected. The Add Relational Record window should look like this:

Add Relational Record

Relational Record
Enter a reference name for record.

Name:

Create a name to refer to this record within the page.

☐ Retrieve an existing record or record list from scope

Scope :

Key :

☐ Reuse metadata definition from an existing record or record list

Input file:

You can automatically add data controls to your page to work with this record (configure details on last page of this wizard).

☒ Add input/output controls to display the record on the web page

Create controls for:

☐ Displaying an existing record (read-only)

☐ Updating an existing record

☒ Creating a new record

8. Click **Next**.
9. In the Table box, expand **W5SAMPLE** and select **ADS**. Click **Next**.
10. Click **Next** again. The Configure Data Controls page opens.

Tip: The Configure Data Controls page helps you customize the visualization of your relational record. For example, you can change the columns, field labels, and submit button on your input form. After you finish these steps, a fully functional input form will be on the page.

11. In the **Fields to display** section, clear the check box next to every field name except for the ones you want in your input form:
 - DESCRIPTION
 - ID
 - MAINCATEGORY
 - PHONE
 - PRICE
 - TITLE
12. Rearrange the fields in the following order by clicking the Up or Down buttons:
 - ID
 - TITLE
 - DESCRIPTION
 - MAINCATEGORY
 - PRICE
 - PHONE
13. Rename the labels as you like. For example, shorten the "Maincategory:" label to just "Category:" To rename the labels generated for the input fields, click a label from the **Label** column. The mouse icon turns into a cursor so you can type new text.
14. Select **MAINCATEGORY** then click **Options**. The Options window opens.
15. Make sure the **Submit button** option is selected.
16. Type Post New Listing in the **Label** field. Then click **OK**. The Add Relational Record window should now look like this:

Add Relational Record

Configure Data Controls

Specify the columns to display and how to display them

Fields to display:

Field Name	Label	Control Type
☒ ID (java.lang.Integer)	Id:	Input Field
☒ TITLE (java.lang.String)	Title:	Input Field
☒ DESCRIPTION (java.lang.S	Description:	Input Field
☒ MAINCATEGORY (java.lan	Category:	Input Field
☒ PRICE (java.math.BigDecir	Price:	Input Field
☒ PHONE (java.lang.String)	Phone:	Input Field
☐ TYPECODE (java.lang.Sho	Typecode:	Input Field
☐ DATEOPEN (java.util.Date	Dateopen:	Input Field
☐ CATEGORY_ID (java.lang.	Category_id:	Input Field
☐ TYPE (java.lang.String)	Type:	Input Field
☐ EMAIL (java.lang.String)	Email:	Input Field
☐ PHOTO (byte[])	Photo:	File Upload
☐ IMAGETYPE (java.lang.Stri	Imagetype:	Input Field
☐ LOCATION (java.lang.Strir	Location:	Input Field

17. Click **Finish** to generate the input form. It should look like this:

classifieds

Home View all listings » Post a listing

Home
View all listings
Post a listing »

Id:	{ID}	123
Title:	{TITLE}	abc
Description:	{DESCRIPTION}	abc
Category:	{MAINCATEGORY}	abc
Price:	{PRICE}	123
Phone:	{PHONE}	abc

{Error Messages}

Post New Listing

{sitenav.userData} Home View all listings Post a listing

Note: The form has an Error Messages field. This does not mean that your project has errors; this field marks the place where errors will be displayed if there are any when the user submits the form.

18. Save the page.

When your input form is submitted, the page will automatically add the new record to the database. In the next lesson, you will program the **Post New Listing** button to return to the `all_records.jsp` page so that you can immediately view the new record in the database.

Lesson checkpoint

You have completed Lesson 1.4. In this lesson, you created a page that allows users to create new classified ads and post them to the database.

Lesson 1.5: Program the Submit button

When your input form is submitted, the page will automatically add the new record to the database. In this lesson, you will program the **Post New Listing** submit button to return to the `all_records.jsp` page so that you can immediately view the new record in the database.

To program the submit button:

1. In the `new_record.jsp` file, right-click the **Post New Listing** button you created in the form on the Web page and select **Properties**.
2. Click **Add Rule**. The Add Navigation Rule window opens.
3. From the Page drop down, select `all_records.jsp` then click **OK**.
4. Save the page. If you want to try adding a record to the database to verify that you will return to the `all_records.jsp` page, run `new_record.jsp` on your test server.

Prevent duplicate keys

Important:

Since the ID column is a primary key in the ADS table, you cannot add records with an ID value that already exists in the table. In Module 2, you will see how to use automatic key generation to automatically create a new unused key for each new record.

Until then, you must enter an unused ID number in this page to add a new record. The records that come with the database use ID numbers from 1 to 24, so you can enter any number above 24 as a primary key. Be sure not to duplicate keys if you enter more than one record.

Bind input to the relational record (optional)

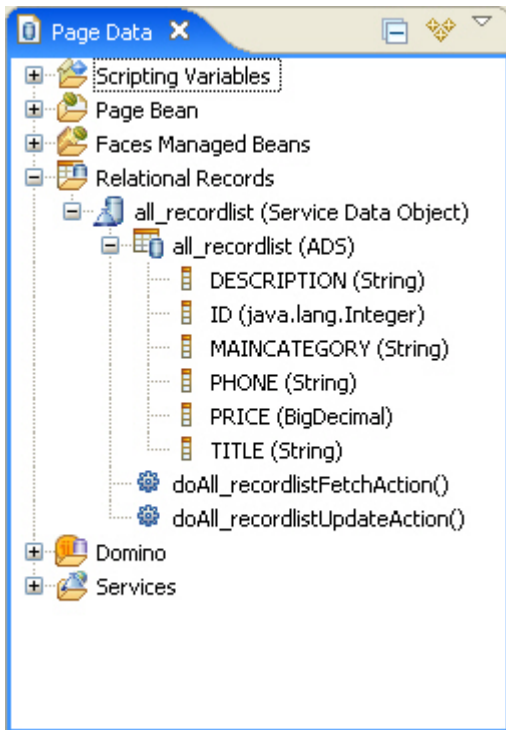
The input form on the page is a set of JavaServer Faces input fields which have been bound to the relational record that you created. Recall that you created the relational record to represent a new record in the ADS table of your database. Binding is a method by which you can link a JavaServer Faces input component to a column in the relational record.

When you created the relational record, the wizard automatically bound all of its columns to the input fields on the page. If you want to make changes manually, you can bind other columns to other input fields. In order to bind a column from your relational record to the input field, drag the column from the Page Data view onto the field. You can experiment with this process by deleting and recreating the Description input field on your form.

Learn more about the Page Data view:

The Page Data view is usually found at the bottom left corner of the workbench. If you can't find it, go to the menu bar and click **Window** → **Show View** → **Page Data**. Generally, the Page Data view is only used when you are creating dynamic Web pages.

The Page Data view stores connections to data sources like databases in the form of data objects such as relational records or relational record lists. These data objects do not represent the data source itself or any component on the page. Instead, these data objects represent a connection between the project and a data source. Once this connection is created, you can drag a data object from the Page Data view onto a Web page to complete the connection and display information from the data source on the Web page. For example, here is a picture of the Page Data view showing a relational record list created during this tutorial:



In this picture, you can see the `all_recordlist (ADS)` relational record list, followed by all of the columns in this relational record list. Once this data object is defined and in the Page Data view, you can drag these columns onto input components on a Web page as many times as you want without reconnecting to the database.

For example, if you wanted to show the value of the `DESCRIPTION` column on a web page, you would first drag a text output field onto the page and then drag the `DESCRIPTION` column from the Page Data view onto that text output field. This is called binding a column to an output field.

Alternately, if you wanted to let the user type in a new value for the `DESCRIPTION` column in the database, you could drag a text input field onto the page. Then, drag the `DESCRIPTION` column onto that text input field.

The following steps are provided to illustrate the concept of the input field and the process of binding; however, walking through these steps is optional in this tutorial. If you don't want to do this, move on to the next lesson.

1. In the `new_record.jsp` file, click the **Description** input field. Press the **Delete** key.
2. In the Palette view, click on the **Enhanced Faces Components** drawer to expand it.
3. Drag an **Input** component from the Palette onto the cell that contained the **Description** input field you just deleted. There is now an input field in this cell, but there is no text such as `{ID}` or `{TITLE}` inside it because this input field is not bound to any column.
4. In the Page Data view, click to expand **Relational Records** → **create_record (Service Data Object)** → **create_record (ADS)** and drag the `DESCRIPTION` column onto the **Input** component you just created. The text inside the **Input** component changes to indicate that it is now bound to the `DESCRIPTION` column, as in this picture:

Id:	{ID} 123
Title:	{TITLE} abc
Description:	{DESCRIPTION} abc
Category:	{MAINCATEGORY} abc
Price:	{PRICE} 123
Phone:	{PHONE} abc
{Error Messages}	
Post New Listing	

5. Save the file and then run the page on the test server if you would like to.

Lesson checkpoint

You have completed Lesson 1.5. In this lesson you learned to program a command button and to bind data to an input field by dragging and dropping a JSF widget from the palette onto a Web page.

Lesson 1.6: Create an update page

At this point, you have created pages for viewing and creating listings for the Web site. In this exercise, you will create a page that allows users to update and delete listings. The update page will look almost exactly like the create page except that on the update page, the input fields will display data from an existing record for the user to change.

First, you will create a relational record, which represents an existing record from the database. Next, you will create a JavaServer Faces update form for this relational record and after a few small changes, your page will be ready to test.

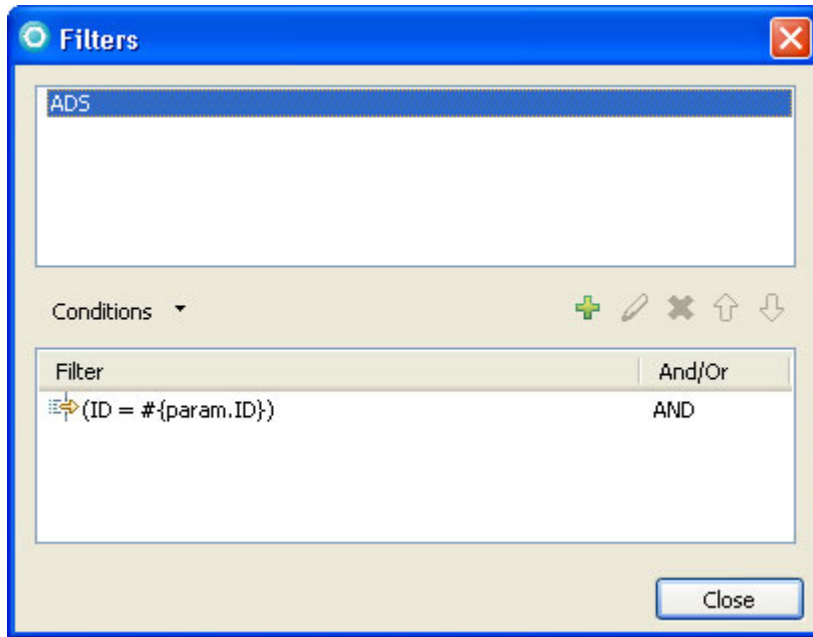
Create the update relational record

1. Open the **update_record.jsp** file by double-clicking it in the Project Explorer view.
2. Delete the default text Default content for bodyarea.
3. In the Palette view, click the **Data** drawer to expand it.
4. Drag the **Relational Record** component from the Palette onto the blank content area. The Add Relational Record window opens.
5. In the Name field, type **update_record**.
6. Under **Create controls for**, click **Updating an existing record**.
7. Make sure the **Add input/output controls to display the Relational Record on the web page** option is selected.
8. Click **Next**.
9. In the Table box, expand **W5SAMPLE** and select **ADS**.
10. Click **Next**. The Add Relational Record page opens.

Filter the results

A relational record can show only one record from the database. Therefore, you must filter the database table so only one record appears for the user to edit. (You didn't need to filter the database in the previous exercise because you created a new record and thus there were no results from the database to filter.) Since each record in the database has a unique ID number, you will filter the results to the one with a given ID number.

1. The Add Relational Record page has a list of task links. Under **Tasks**, click **Filter Results**. The Filters window opens and inserts the default filter condition **ID = #{param.ID}** in the **Filter** column. The Filters window looks like this:



This code filters the records in the database so only the record with the specified ID number will appear in the relational record. You will learn more about this condition in the Add a row action section later in this exercise. Click **Close**.

2. Click **Next**. The Configure Data Controls page opens.
3. In the **Fields to display** section, clear the check box next to every field name except for those you want to display in your input form:
 - ID
 - TITLE
 - DESCRIPTION
 - MAINCATEGORY
 - PRICE
 - PHONE
4. By clicking **Up** or **Down**, reorder the field names from top to bottom as follows:
 - ID
 - TITLE
 - DESCRIPTION
 - MAINCATEGORY
 - PRICE
 - PHONE
5. For the ID field, select **Output field** from the drop-down list in the **Control Type** column. Although you want users to be able to view a record ID number, you do not want them to be able to update it. Making the ID field into an Output field will help you avoid the problem of duplicate IDs.
6. Click **Options**. The Options window opens. Make sure the **Submit button** option is selected. Type Update in the **Label** field. Click **OK**. The Add Relational Record window should look like this:

Add Relational Record

Configure Data Controls

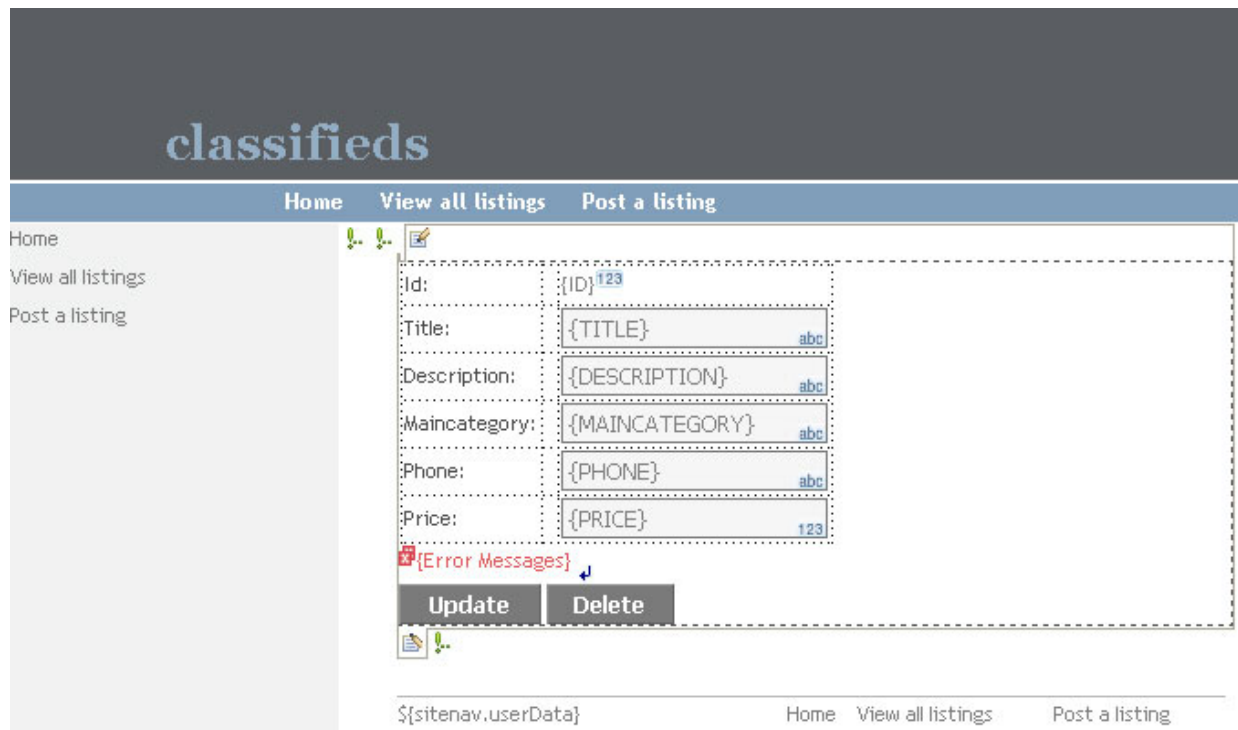
Specify the columns to display and how to display them

Fields to display:

Field Name	Label	Control Type
☒ ID (java.lang.Integer)	Id:	Output Field
☒ TITLE (java.lang.String)	Title:	Input Field
☒ DESCRIPTION (java.lang.String)	Description:	Input Field
☒ MAINCATEGORY (java.lang.String)	Maincategory:	Input Field
☒ PRICE (java.math.BigDecimal)	Price:	Input Field
☒ PHONE (java.lang.String)	Phone:	Input Field
☐ TYPECODE (java.lang.Short)	Typecode:	Input Field
☐ DATEOPEN (java.util.Date)	Dateopen:	Input Field
☐ CATEGORY_ID (java.lang.Integer)	Category_id:	Input Field
☐ TYPE (java.lang.String)	Type:	Input Field
☐ EMAIL (java.lang.String)	Email:	Input Field
☐ PHOTO (byte[])	Photo:	File Upload
☐ IMAGETYPE (java.lang.String)	Imagetype:	Input Field
☐ LOCATION (java.lang.String)	Location:	Input Field

?

- Click **Finish** to generate your update form on the page, as shown below:



Program the update and delete buttons

Again, you will add code to refer the user to the all_records.jsp page to display the changed record along with all the other records.

1. Right-click the **Update** button that you just created on the Web page and select **Properties**.
2. Click **Add Rule**. The Add Navigation Rule window opens.
3. From the Page drop down, select **all_records.jsp**. In the **This rule is used by** box, click **This action only**. Ensure the field contains: `#{pc_Update_record.doUpdate_recordUpdateAction}`. Click **OK**.
4. Repeat steps 1-3 to edit the code in the same way for the Delete button on the Web page. Ensure the **This rule is used by** field contains: `#{pc_Update_record.doUpdate_recordDeleteAction}`
5. Save the page. If you want to try adding a record to the database to verify that you will return to the all_records.jsp page, run new_record.jsp on your test server.

Add a row action

Now, you will add a row action to the table on all_records.jsp so the user can select a database record to update. `#{param.ID}` represents the ID number of the record that the update page will update. When the user clicks a row, that record ID number will be sent to the update_record.jsp page as the `#{param.ID}` parameter. Then, the filtered relational record you just inserted into the update_record.jsp page will display only that record for updating.

1. In the Project Explorer view, double-click the **all_records.jsp** file to open it in the Editor. Click anywhere inside the data table. Open the Properties view.
2. In the Properties view, click the **Row Actions** tab under **hx:dataTableEx** from the list of HTML tags.
3. Click **Add an action that's performed when a row is clicked**. The Configure a RowAction dialog opens. Select **Clicking the row sends a request to the server with row information sent as parameters** then click **OK**. A column is added to the data table.
4. In the Properties view, click the **hx:requestRowAction** tab.
5. Click **Add Rule**. The Add Navigation Rule dialog opens.
6. From the Page drop down, select **update_records.jsp**. In the **This rule is used by** box, select **This action only**. The **this action only** field is automatically populated by `#{pc_All_records.doRowAction1Action}` which was created when you added the row action. Click

OK. If the field is not automatically populated with the above method, type the method in the field and add the following to ClassifiedsTutorial/Java Resources: src/pagecode/All_records.java:

```
public String doRowAction1Action() {  
    return "";
```

7. In the Properties view, click the **Parameter** tab under hx:requestRowAction. Click **Add Parameter** then type ID in the Name field. You need to bind the ID parameter to the ID column. Binding the row action to the ID column of the data list means that when the row is clicked, the request parameter will be the ID of the record in the list.
 - a. Click **Value** then click the **Select Page Data Object**  button.
 - b. In the **Data Object** tab, expand **all_recordlist (Service Data Object)** → **all_recordlist (ADS)** and select **ID (java.lang.Integer)**.
 - c. Click **OK**.

Now when the user clicks a row, the Web site will allow the user to update information about the classified ad. Save the file and test the page if you would like. Remember to open all_records.jsp first, because this is the page that links to update_record.jsp.

Lesson checkpoint

You have completed Lesson 1.6. In this lesson, you learned how to create Web pages that update records in a database.

Module 1: Summary

You have completed Module 1: Create Web pages with data connections.

Lessons learned

You now know how to:

- Display information from databases on Web pages.
- Work with relational records, relational record lists, and data tables.
- Display, edit, create, and delete database records from a Web page.
- Link to a record or set of records using a row action on the data table component.

This module introduced you to JavaServer Faces technology and tooling. There is much more you can do to make your site attractive, usable, and efficient. In the next module, “Module 2: Add advanced features” on page 26, you will learn how to take this simple classified advertising site and turn it into a more visually pleasing and complex application. Continue to the next module if you want to learn to make your site look like this:

Home

View all listings »

Post a listing

View all listings

Title	Details	Category
W/V Passat wagon 	Description: 2000 good condition Price: \$1,000.00 Phone: 416-512-8665	Automotive
Magic Flute 	Description: Imagine it. Price: \$101.00 Phone: 314-159-265	Entertainment
Video Camera 	Description: Great camera, suitable for amateur or professional use. Price: \$900.00 Phone: 534-333-454	Entertainment
VCR 	Description: Old and worn but still working ok, hence low price. Price: \$40.00 Phone: 322-654-044	Entertainment
Flat Screen TV 	Description: Very clear picture, sound excellent too. Reluctant sale. Price: \$700.00 Phone: 456-456-944	Entertainment
1 2 3 4 5		

Module 2: Add advanced features

In this module, you will learn how to use advanced features to modify the format of your Web pages, add a feature that allows users to upload files to the database, set up rules that will automatically send users to a particular page, and automate some administrative tasks such as key generation.

Learning objectives

This module teaches you more powerful ways to use data from a database. In this module, you will:

- Format database records on a Web page
- Add a component that allows uploading files to a database from a Web page
- Navigate from page to page automatically
- Automate some database administration tasks

Time required

This module should take approximately 1 hour and 30 minutes to complete. If you decide to explore other facets of dynamic Web sites while working on the tutorial, it could take longer to finish.

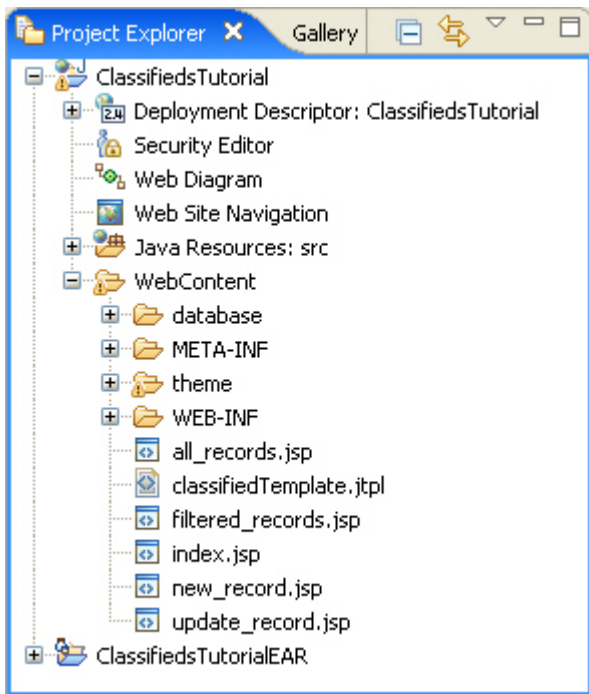
Prerequisites

If you have already completed Module 1: Creating Web pages with data connections, skip the rest of this prerequisites section and begin working on “Lesson 2.1: Format a data table” on page 30.

If you are beginning your work in this tutorial with module 2, without doing module 1 first, you must first import the required resources, set up the target server, and set up the database connection.

To import the required resources:

1. Import the project. Switch to the Web perspective (**Window** → **Open Perspective** → **Web**).
2. In the Project Explorer view of the Web perspective, ensure that your ClassifiedsTutorial project looks like the following image:



To set the target server:

1. In the Project Explorer view of the Web perspective, right-click **ClassifiedsTutorial** and select **Properties**.
2. In the properties list, click **Server**.
3. In the Default server list, select the server that you want to use as the default. Click **Apply**.
4. In the properties list, click **Targeted Runtimes**.
5. In the Runtimes list, click the runtime that corresponds with the server that you selected. Click **OK**.

If the server that you want to use is not listed in the target runtime list, close the workbench and install the server that you want to use. Once the server is installed, follow the instructions to set the target server.

Note: If you do not see any servers listed in the Default server list, and you have installed server runtimes, it is possible the server needs to be configured. To configure a server, you can do the following:

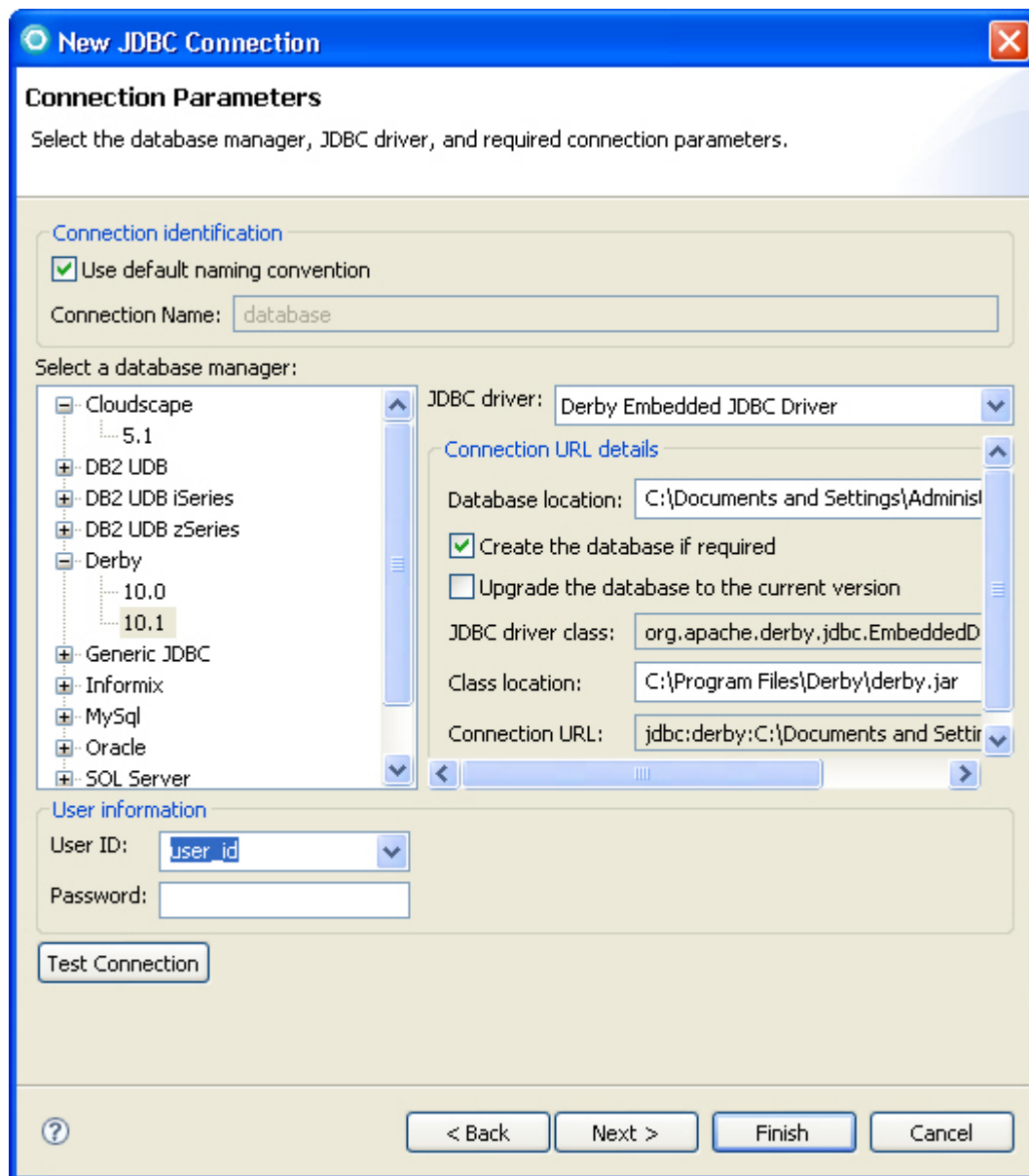
1. Right-click **all_records.jsp** file, then select click **Run As → Run on Server**.
2. Choose **Manually define a new server**.
3. Select a server you have installed.
4. Follow the directions in the wizard to configure the server. The first time you run on server you may receive an error. To fix the error set the target server as described above, restart the server in the Servers view, and reload the Web page in the browser.

If you go back through the previously described steps for setting a target server, you will now find the default server is the one you have just configured. If a server is installed but not configured, it will not show up in the list of servers from which you can choose a default target.

For information on deploying SDOs on servers other than WebSphere Application Servers, refer to Help topic: Deploying SDOs on non-WAS servers.

To set up the database connection:

1. In the Project Explorer view, right-click **ClassifiedsTutorial** and select **Properties**. The Properties for the ClassifiedsTutorial window opens.
2. Click JDBC Connections.
3. In the JDBC Connection properties, click **New**. The New JDBC Connection window opens.
4. Click **Create a new connection** then click **Next**.
5. In the **Select a database manager** list, expand **Derby**, then select **10.1**.
6. In the Database location field, click **Browse** and select `<workspace_location>\ClassifiedsTutorial\WebContent\\database`, where `<workspace_location>` is the directory of your current workspace. Click **OK** to close the Browse For Folder window.
7. In the **Class location field**, type in the location of the JAR or JARs that implement the JDBC driver class, or click **Browse** to select the location from your file system. Once you have chosen the location of the JAR files, click **Open**.
8. You may need to enter a User ID to access the database. A password is not required. The New Database Connection wizard should look like this:



9. In the New JDBC Connection wizard, click **Finish**. You return to the Properties for ClassifiedsTutorial window.
10. Click **Configure project's database connections**. The JDBC Connections properties window opens.
11. Click **Edit** next to the Runtime connection details section. Select **Use driver manager connection** as the runtime connection type. Click **Finish** then click **OK**.

You can browse the files in the tutorial Web project. To open a file, double-click it in the Project Explorer view. To view a map-like representation of how the pages are related, double-click Web Site Navigation in the Project Explorer.

The majority of your work in this sample will involve the following files:

all_records.jsp

This is the site's home page. It will display every classified ad in the database.

new_record.jsp

This page will create a new classified ad.

update_record.jsp

This page will change the details about an ad in the database or delete it.

classifiedTemplate.jtpl

This is the template for the site pages. It includes elements like the table and the gray "Welcome to the Classifieds" banner that are on every page. This page also has two navigational tabs below the gray banner that lead to the home page and the new classified ads page.

Lesson 2.1: Format a data table

The data table on the `all_records.jsp` page has a utilitarian appearance. It displays all of the records in the database and links to the other pages as it was intended to do, but it is not very attractive. In this exercise, you will improve this data table by adding a pager, style rules, and pictures of the items for sale.

Rename column headers

In many cases, you do not want the column headers in your data table to be the exact column names from the database. The following steps show you how to rename data table column headers to something more appropriate.

1. Double-click the `all_records.jsp` page in the Project Explorer view
2. Click the **Maincategory** column header in the data table.
3. In the Properties view, click the **h:outputText** tab and change the **Value** text field from `Maincategory` to `Category`.
4. If you want to, rename the other columns.
5. Save the page.

Format output components

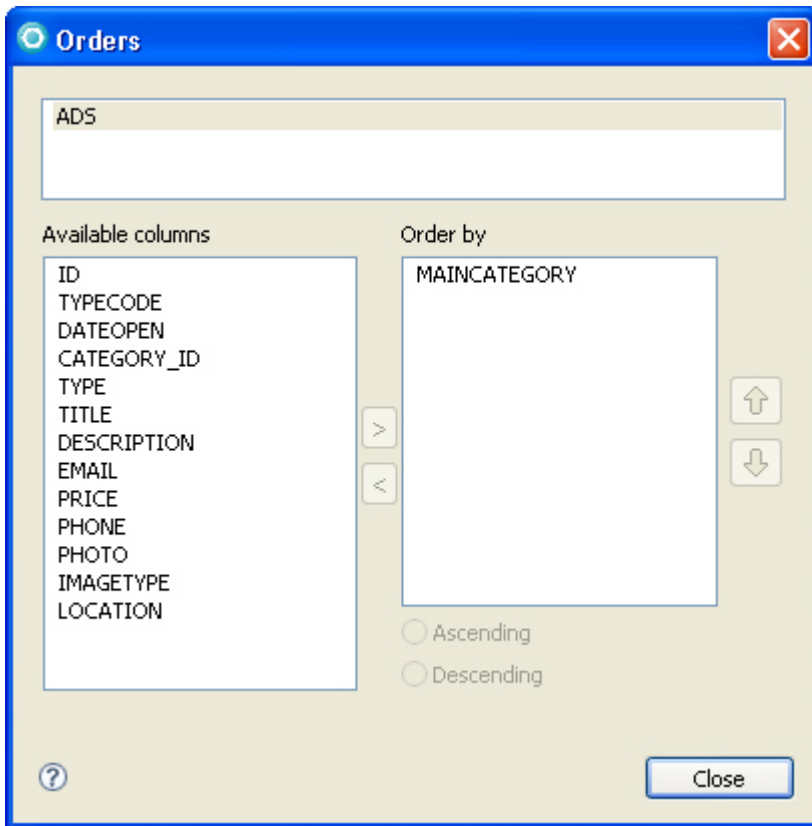
You can also format the output components. In these steps, you will format the `{PRICE}` output component to appear as a currency value instead of as an ordinary number.

1. Click the `{PRICE}` output component on the page.
2. In the Properties view, find the **Type** drop-down list and select **Currency**. Now the price of each item will be shown in the currency style instead of as an ordinary number. Other styles and formats are available for different types of data such as percentages, dates, and times. You can also customize the format of an output component to show a type of data not listed in the Properties view, such as a telephone number.
3. Save the page.

Sort the data

Right now, the data is not listed in any order. To provide the users with a more organized list of ads, order the record list by category so that similar items will be grouped together.

1. Right-click **all_recordlist (ADS)** in the Page Data view (**Window** → **Show View** → **Page Data**) and click **Configure** from the context menu. The Configure Relational Record List window opens. If a warning message appears and says that a connection to your database could not be established, then you left the server running after testing the Web site. If this happens, click cancel in each dialog and stop the server as explained in the section "Stopping the server" in Testing the Web site.
2. Select **Retrieve existing record or record list from scope** and **Reuse metadata definition from an existing record or record list** and then click **Next**.
3. In the next window, click to select **ADS**, then click **Next**.
4. On the right side of the next window, click **Order results**. This opens the Orders window.
5. In the Available columns pane, click **MAINCATEGORY**.
6. Click the **>** button to move the **MAINCATEGORY** column into the Order by pane. The window should look like this:



7. Click **Close** to close the Orders window.
8. Click **Finish** to apply the changes. Now, the data will be ordered by category. You can sort by any column in the database.

Add a pager

Rather than display every record at once in our data table, you can use a pager. A pager automatically splits the records into pages of a set size without creating any new JSP files in your project.

1. Click anywhere inside the data table.
2. In Properties view, click **h:dataTableEx** from the list of HTML tags at the left of the view.
3. Click the **Display options** tab under **h:dataTableEx**.
4. In the **Rows per page** field, enter 5.
5. Click the button next to **Add a web style pager**. A picture of what the pager will look like appears at the bottom of the data table. The visualization of the pager in Page Designer is a placeholder image and does not actually reflect how many pages will be displayed, as this can only be determined when the page loads in a browser.
6. Save the page. You can experiment with the different styles of pagers and the page statistics component to find the right fit for the page if you want.

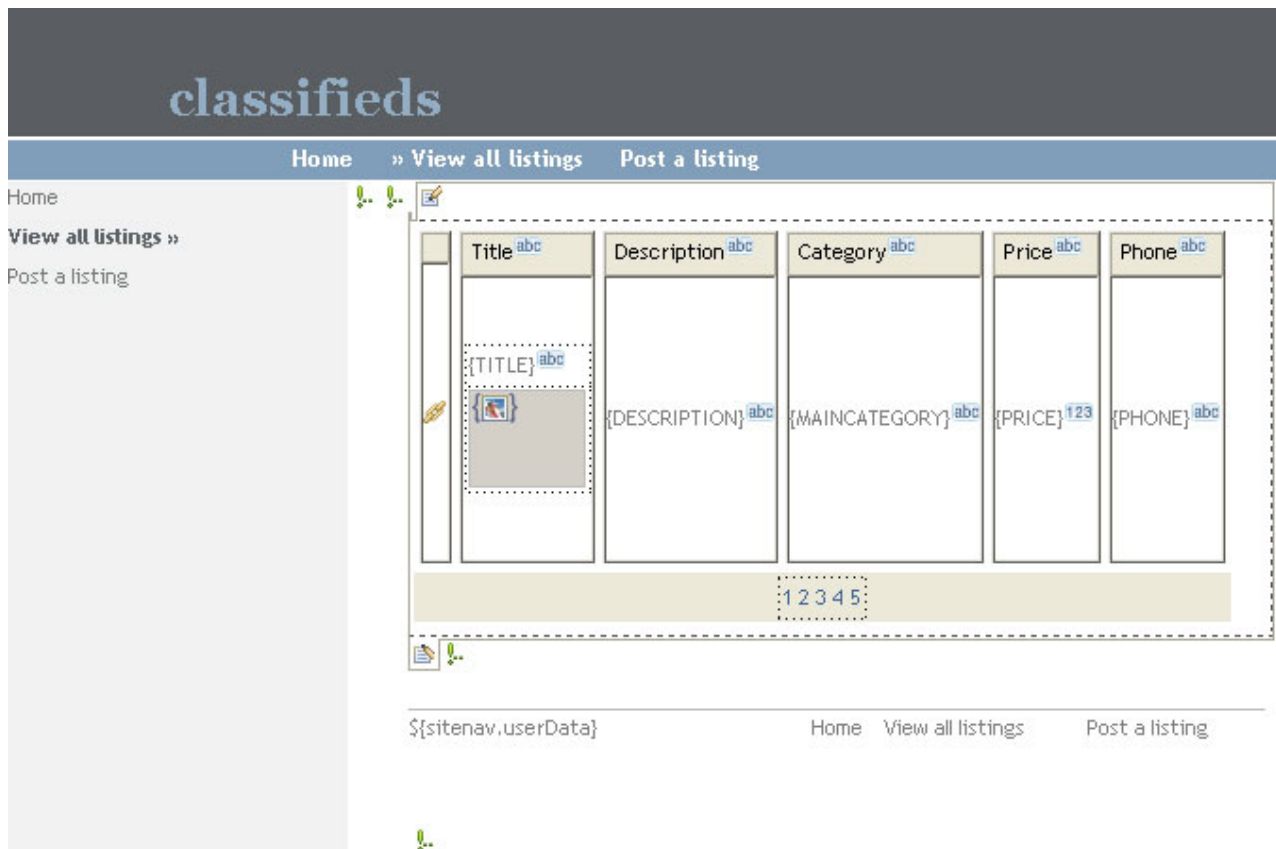
Lay out the components in a group box as a list

Instead of having exactly one output component in each column of the data table, it is possible to combine elements within columns to produce a more attractive layout. You will use a group box to arrange the components, similar to how you would use a hidden HTML table. Specifically, after adding the image component to visually enhance the site, you will take most of the textual information about each ad and place it into a single data table column, which will then be labeled DETAILS.

1. Click the Enhanced Faces Components drawer in the Palette to open it. Drag a **Panel - Group Box** component into the TITLE column of the data table. The Select Type window opens.

2. Choose **Box** as the group box type in the Select Type window and click **OK**. This group box lays out components placed within it in a single column or row and can be oriented either vertically or horizontally.
3. Click the group box to select it.
4. Using the Properties view, change the **Orientation** to **Vertical**.
5. Drag the **{TITLE}** component into the list group box. The instructional text in the group box disappears once you add a component to it.
6. Drag an **Image** component from the Enhanced Faces Components drawer of the Palette into the list group box. It helps to drop the image component onto the bottom edge of the group box in order to ensure that the image will be placed below the title.
7. Click the image component that you just added.
8. Use the **Size** area of the Properties view to set the **Width** to 60 pixels and the **Height** to 50 pixels. This ensures that no matter what size the images are in the database, they will always appear as 60x50 on the page.
9. Bind the image component to the PHOTO column of all_recordlist by dragging the PHOTO column from the Page Data view onto the image component. This will cause the image component to display the image data found in the PHOTO column of each record. If the PHOTO (ByteArray) column is not listed, right-click **all_recordlist (ADS)** in the Page Data view and select **Configure** from the menu. Click **OK** to close the URL window if it opens. In the Configure Relational Record List window, click **Next**. Select ADS and then click **Next**. Click to check the box next to **PHOTO (ByteArray)** and then click **Finish**.
10. Click the graphic you inserted. On the **hx:graphicImageEx** tab of the Properties view, click the **Select Page Data Object**  button next to the **Type** field. The Select Page Data Object window opens.
11. Click to expand **all_recordlist (Service Data Object)** → **all_recordlist (ADS)**.
12. Click **IMAGETYPE (String)**.
13. Click **OK**, then save the page.

Your page should now look like this:

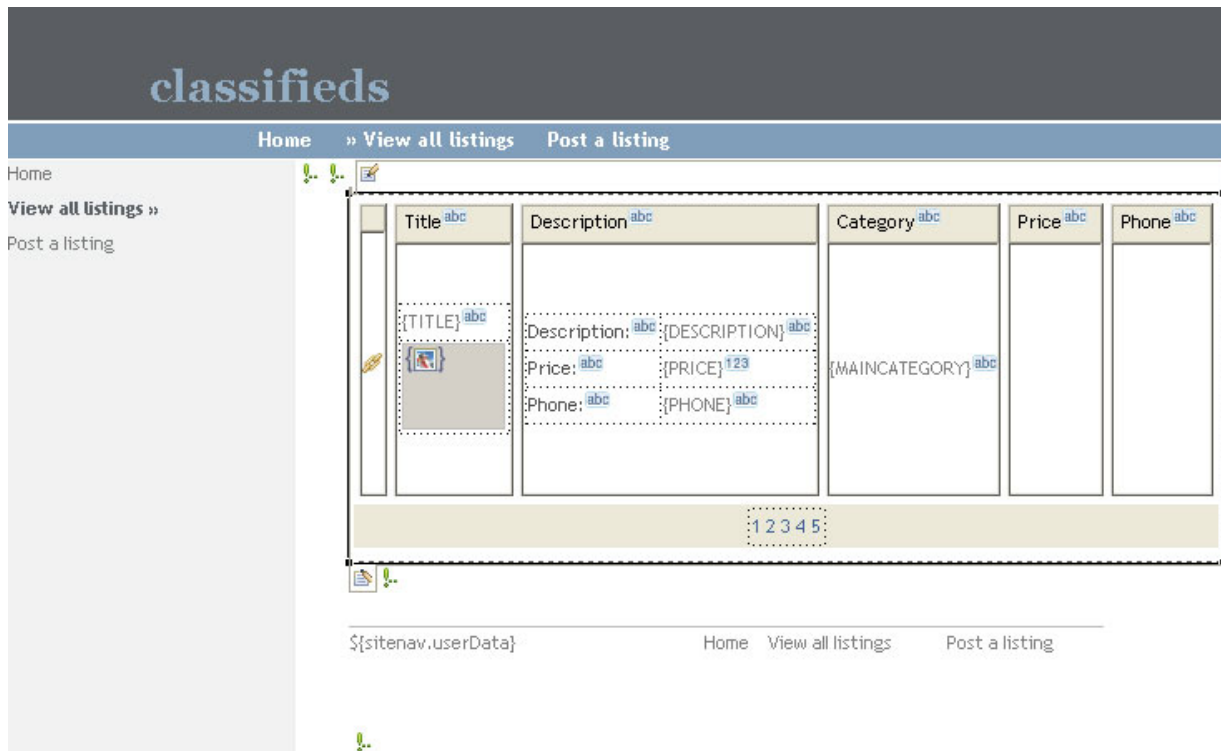


Refine layout with a group box grid

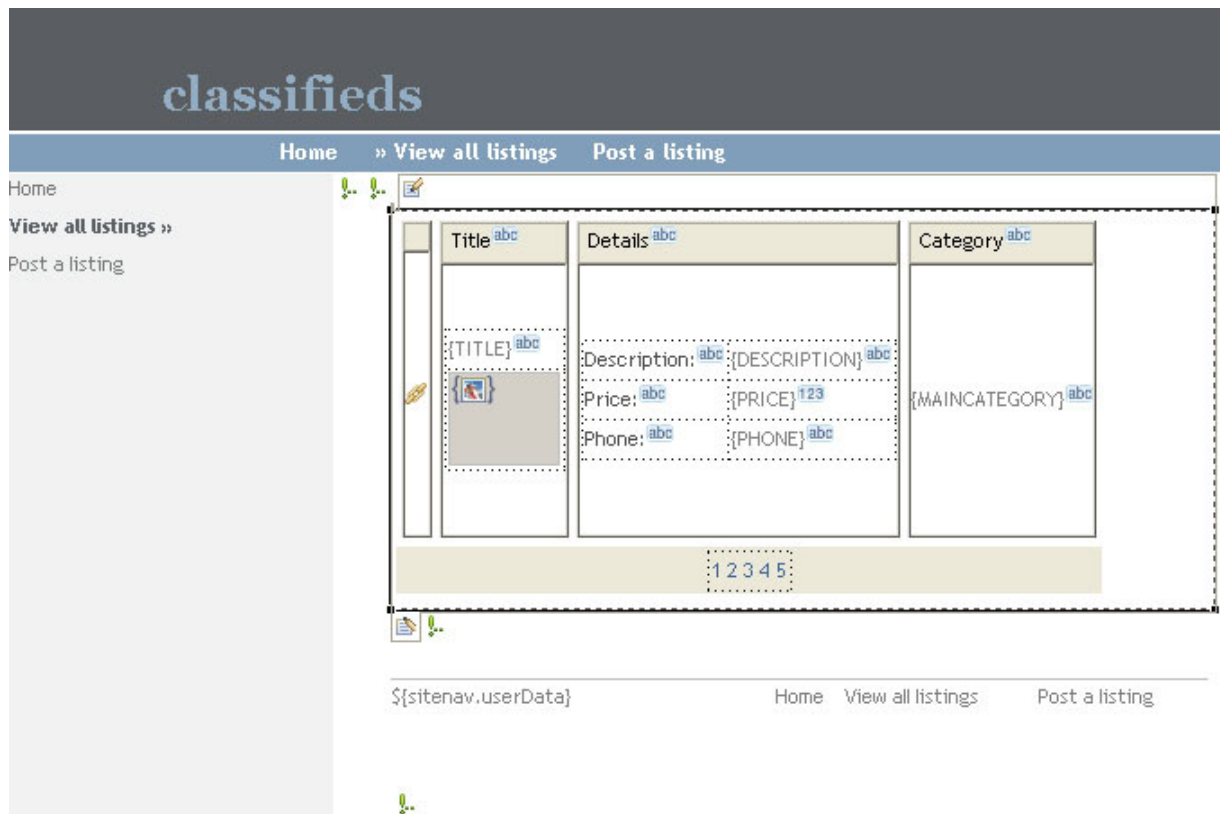
A group box can also organize components in a format like a table. A list type group box can only have one row or column, but a grid type group box can have any number of rows and columns. In these steps, you will move the {PRICE} and {PHONE} components into the DESCRIPTION column of the data table and include labels for them.

1. Drag a **Panel - Group Box** component from the Enhanced Faces Components drawer of the Palette into the DESCRIPTION column of the data table. The Select Type window opens.
2. Click **Grid** as the type of component to add and click **OK**.
3. Click the new grid group box.
4. Use the Properties view to set the number of **Columns** for this group box to be 2.
5. Drop an **Output** component from the Faces Components drawer of the Palette into the grid group box. This output component will be a label for the description of the item for sale. Each output component will have a label like this in the left cell of the table.
6. Click the **Output** component (it appears in Page Designer as **outputText**) and use the Properties view to assign it a **Value** of Description:
7. Drag the existing {DESCRIPTION} component to the bottom edge of the grid group box. If you are having trouble dragging the components to the right place in the group box, try holding the mouse button down and watching the cursor. The location of the cursor in the group box indicates where the component will appear when you let go of the mouse button. For this exercise, you should release the mouse button when the cursor is to the right of the previous component.
8. Drag another **Output** component from the Palette and drop on to the right side of the group box.
9. Use the Properties view to assign it a **Value** of Price:.
10. Drag the existing {PRICE} output component on the right of the Price output text.
11. In the same way, drag a new **Output** component for the label for the {PHONE} component and label it Phone:.

12. Drag the existing {PHONE} component into the grid group box to the right of the Phone output text. Your page should now look like this:



13. Delete the empty PRICE and PHONE columns of the data table. To delete a column, click the column and open Properties view. Then, click **h:dataTable** from the list of HTML tags at the left of the view, select the column you want to delete from the list at the right of the view, and click Remove.
14. Click the header of the Description column and use the Properties view to change its **Value** field to Details. The page should look like this:



15. Save the page and test it. If you are not familiar with running applications on the server, refer to "Lesson 1.3: Test the Web site" on page 13.

When you test the page, it should look like this:

classifieds

[Home](#)
[» View all listings](#)
[Post a listing](#)

[Home](#)
[View all listings »](#)
[Post a listing](#)

View all listings

Title	Details	Category
W/V Passat wagon 	Description: 2000 good condition Price: \$1,000.00 Phone: 416-512-8665	Automotive
Magic Flute 	Description: Imagine it. Price: \$101.00 Phone: 314-159-265	Entertainment
Video Camera 	Description: Great camera, suitable for amateur or professional use. Price: \$900.00 Phone: 534-333-454	Entertainment
VCR 	Description: Old and worn but still working ok, hence low price. Price: \$40.00 Phone: 322-654-044	Entertainment
Flat Screen TV 	Description: Very clear picture, sound excellent too. Reluctant sale. Price: \$700.00 Phone: 456-456-944	Entertainment

1 2 3 4 5

Any ads that do not have an associated image will display a broken link. In the next lesson, you will learn how to add a file upload component to the page to add or change an ad's image.

Lesson checkpoint

You have completed Lesson 2.1. In this lesson you have learned how to format your data table in various ways.

You learned how to:

- Change the column headings.
- Sort the data in your table.
- Add a pager to break the data into several smaller Web pages.
- Use group boxes in various ways to change the layout of the table on the Web page.

36 Display dynamic information on Web pages with JavaServer Faces

Lesson 2.2: Use the file upload component

Use this lesson to learn how to give users the ability to upload pictures for the classified ads. This function is particularly important on the `new_record.jsp` and `update_record.jsp` pages. The file upload component allows users to browse their file system, upload a file to the database, and have this file show up in the database immediately.

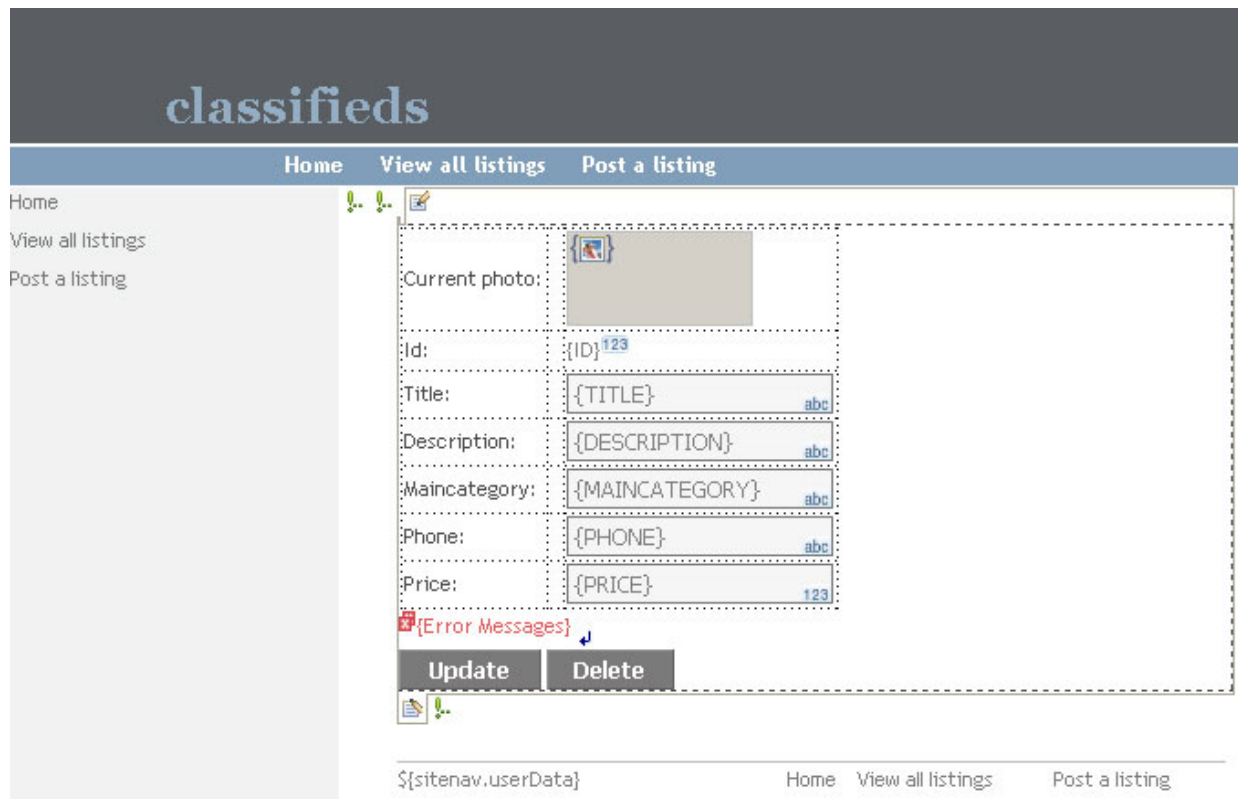
Since the procedure for using the file upload component is similar in both the create and update cases, this exercise will allow the update page to change the current image for any listing. If you want, you can make the same changes to the new record page when you are finished with the update page.

Add the current photo to the page

You will notice that the update form is nothing more than an HTML table with static text and input components bound to columns in the `update_record` relational record. Knowing this, you can add rows and columns to modify the form just like modifying any HTML table. In the following steps, you will add a new row to show the current photo.

Adding rows and columns is more complicated in a data table displaying a relational record list. For information about adding columns to a data table, refer to “Lesson 1.2: Connect to a database and display data on a Web page” on page 6.

1. Double click the `update_record.jsp` page in the Project Explorer view.
2. Place the cursor in the first (top left) cell of the table. This cell is labeled ID.
3. From the main menu, click **Table → Specify and Add**. The Add Rows or Columns dialog opens. Click **Rows** then **Above the selected cell**. This will create a new row at the top of the table that will contain the current photo for the record. Click **OK**.
4. In the left cell of this new row, type `Current photo:` to act as a label.
5. Drag an **Image** component from the Faces Components drawer of the Palette into the right-most cell of the new row. Unlike in the previous exercise, you will allow the user to see the full image without constraining the size, so do not change the width and height in the Properties view.
6. Bind the image component to the `PHOTO` column of `update_record` by dragging the `PHOTO` column from the Page Data view onto the new image component. The image component is now bound to the `PHOTO` column of the database.
7. With the graphic image still selected, go to the Properties view and click the Select Page Data Object button next to the Type field. The Select Page Data Object window opens.
8. Click `update_record (Service Data Object) → update_record (ADS) → IMAGETYPE (String)`.
9. Click **OK** Now, the page shows the current photo for the classified ad, if it has one. Your page should look like this:



Add the file upload component to the page

Next, add a new row to the bottom of the table to contain the file upload component.

1. Place the cursor in the last row by clicking it.
2. From the menu bar, click **Table** → **Add Row Below**.
3. In the first cell of the new row, type New photo: as the label.
4. Drag a **File Upload** component from the Enhanced Faces Components drawer of the Palette into the last cell of the new row.
5. In the same way that you bound the image component, bind the file upload component to the **PHOTO** column of the update_record relational record. **{PHOTO}** is shown in the text field to show that this component is bound to the **PHOTO** column and that the uploaded file will be placed in that column. Your page should look like this:

The screenshot shows a web application titled "classifieds". At the top, there is a navigation bar with links: Home, View all listings, and Post a listing. Below this, on the left, is a sidebar with links: Home, View all listings, and Post a listing. The main content area displays a form for updating a listing. The form is enclosed in a dashed border and contains the following fields:

- Current photo: A placeholder image.
- Id: A text field containing the value 123.
- Title: A text field containing the value abc.
- Description: A text field containing the value abc.
- Maincategory: A text field containing the value abc.
- Phone: A text field containing the value abc.
- Price: A text field containing the value 123.
- New photo: A text field containing the value PHOTO, followed by a Browse... button.

Below the form, there is a section for Error Messages, followed by two buttons: Update and Delete. At the bottom of the page, there is a footer with the text \${sitenav.userData} and the same navigation links: Home, View all listings, and Post a listing.

6. Save the page and test it if you want.

Lesson checkpoint

You have completed Lesson 2.2. In this lesson you learned how to manipulate the table on the update page, add a current photo to the table, and add a file upload component to the table.

Lesson 2.3: Use navigation rules

In this lesson, you will learn how to set up navigation rules to redirect a site user to specific Web pages.

The way the new_record.jsp page is set up, users must be very careful not to enter an ID number that is already in use, because each record in the database must have a unique ID number. This was explained in more detail under the note "Prevent duplicate keys" in "Lesson 1.5: Program the Submit button" on page 18. In this lesson, you will check to see if the ID entered is unique, and if not, send the user to an error page that describes the problem and tells the user how to fix it.

Navigation rules allow you to direct users to the error page or the all_records.jsp page after checking to see if users have entered a duplicate ID number or not. You will assign aliases to these two possible outcomes and then link those two aliases to the correct target pages. In this example, an error on the create_record.jsp page will signal the ERROR_CREATE alias, which will send users to the error page. If users fill out the create_record.jsp page correctly, it will signal the MAIN alias, which will be linked to the all_records.jsp page as usual.

Set up the rules

1. Double-click the **new_record.jsp** page in the Project Navigator.
2. Click the **Post New Listing** button on the page.

3. In the Properties view, click **Add Rule**. The Add Navigation Rule window opens. The first rule will send the user to an error page named `create_error.jsp` if something goes wrong when creating and committing the new record to the database.
4. In the **Page** field, type `error_create.jsp`. This page does not exist, but for the purpose of this tutorial, you can just imagine that it exists.
5. Click the **The outcome named** radio button.
6. Type `ERROR_CREATE` in the text field after the **The outcome named** radio button.
7. Click the **This page only** radio button because there is no other page in the site where the user could trigger this particular error by entering a duplicate ID number. Click **OK**. The next rule will navigate to `all_records.jsp` if the user entered a valid ID number.
8. Click the **Add Rule** button to open the Add Navigation Rule window again.
9. Use the **Page** drop down box to select `all_records.jsp`.
10. Click the **The outcome named** checkbox and then type `MAIN` into the text field after it.
11. Since you might want to reuse this rule in another page (for instance, the `update_record.jsp` page), click the **All pages** radio button under **This rule is used for**. Click **OK**. The two rules are now displayed on the Properties view.

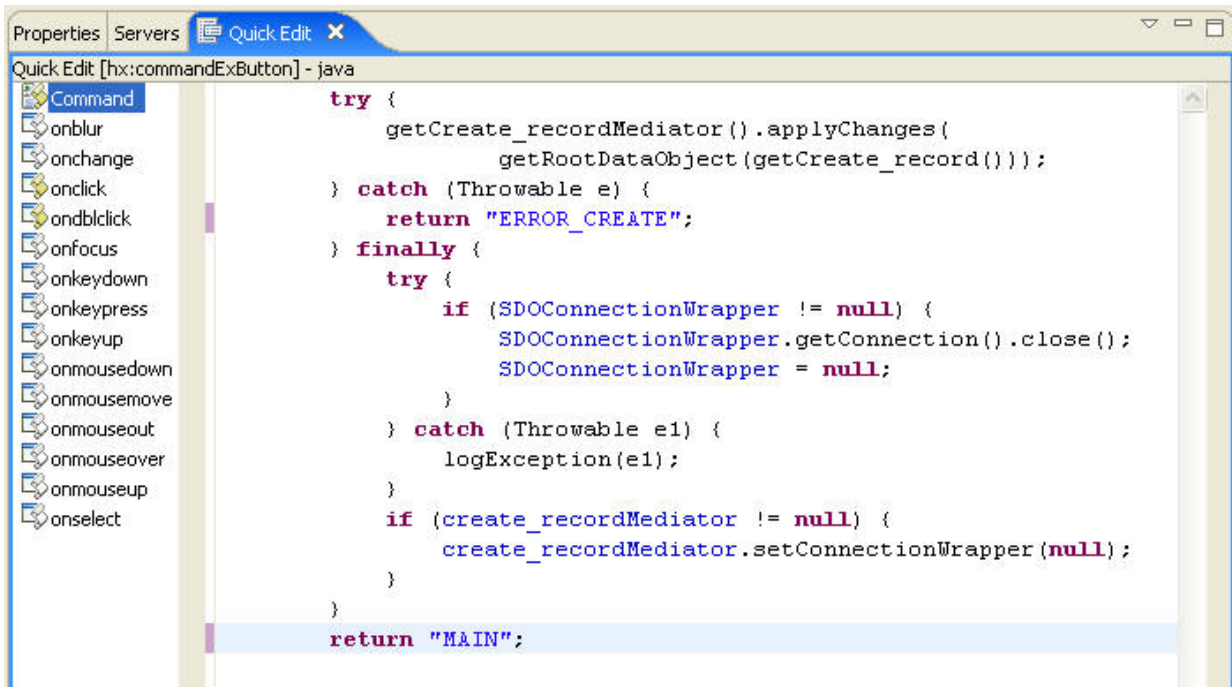
Return aliases from the button action

All that remains now is to put the new navigation rules to work. You will add two return statements into the code for the Post New Listing button. These return statements summon the appropriate alias so the user is sent to the appropriate page as set in the navigation rule.

1. Click the **Post New Listing** button.
2. Open the **Quick Edit** view.
3. In the Quick Edit view, find the line that reads `} catch (Throwable e) {`. This catch function runs if the user has entered a duplicate ID number.
4. Remove all the code between the opening curly brace `{` at the end of this line and the next closing curly brace `}` a few lines down. Do not remove either curly brace.
5. In place of the code between the braces, type this text: `return "ERROR_CREATE";`
6. Next, remove all code below the last closing curly brace `}` and in its place type this text: `return "MAIN";`

This step removes the `gotoPage` action that you added in exercise 1.4. You no longer need this code because the navigation rules do the same thing.

Your button's code should now look like this:



7. Save your page and test it if you want.

Optionally, you can create a simple error page named `create_error.jsp` which explains to your users that an error occurred while creating their listing and to try a different ID value. You can then test these navigation rules by attempting to add a new listing that uses an existing ID value (such as 1).

Lesson checkpoint

You have completed Lesson 2.3. In this lesson you learned how to use navigation rules to create an error if user try to enter an invalid ID value, or to send the user to `all_recordlist.jsp` if the user enters a valid ID.

Lesson 2.4: Use automatic key generation

In the previous lesson, you set up navigation rules to ensure that users entered a unique ID for the new classified ad. This process is frustrating and unrealistic, because you would not want your site users to find a unique ID through trial and error. In this lesson, you will set up automatic key generation so that the database assigns a unique number for each new record in the database automatically.

Automatic key generation is a complex topic, but in short, a database can choose new keys if it has a special table reserved for key generation. This table must have a list of unused keys in one column (the *incrementor* column) and a list of numbers in order starting with 1 in the other column (the *identity* column). When the database needs a new key, it takes the key from the row with the 1 in the identity column and then gets a new key ready for the next time.

Learn more about automatic key generation:

In order to set up automatic key generation, you must know a few things about how it works. To use automatic key generation, the database must have a table set aside for this purpose. This table has two columns:

- The *incrementor* column stores available keys. When the database needs a new unique key, it retrieves one from this column.

- The *identity* column is a list of numbers with only one instance of the number 1. This column tells the database which key to select from the incrementor column. The identity column is the primary key of the key generation table.

When the database needs a new unique key, it finds the row with an identity column value of 1. This row's incrementor column value holds the next available key. The database uses this key and updates the table so a new key will be available next time.

Here is an example of a key generation table. What is the next available key for this database? The answer is below the table.

Table 1. Key generation table

Identity Column	Incrementor Column
3	78
4	3
1	65
2	12

The next available key in this table is 65, because that key (in the incrementor column) is in the same row as the 1 in the identity column.

After the next available key is fetched from the table, the table is updated for the next time a key is needed. The database can also retrieve multiple keys at once by taking the incrementor column value of more than one row.

In short, for automatic key generation to work, you need only have a key generation table set up with two columns: a primary key column for use as an identity column, and a column for storing the next available key. This table must be initialized with one record whose identity column value is 1 and whose incrementor column value is the first available key to use. Once you have this set up, you are ready to use automatic key generation.

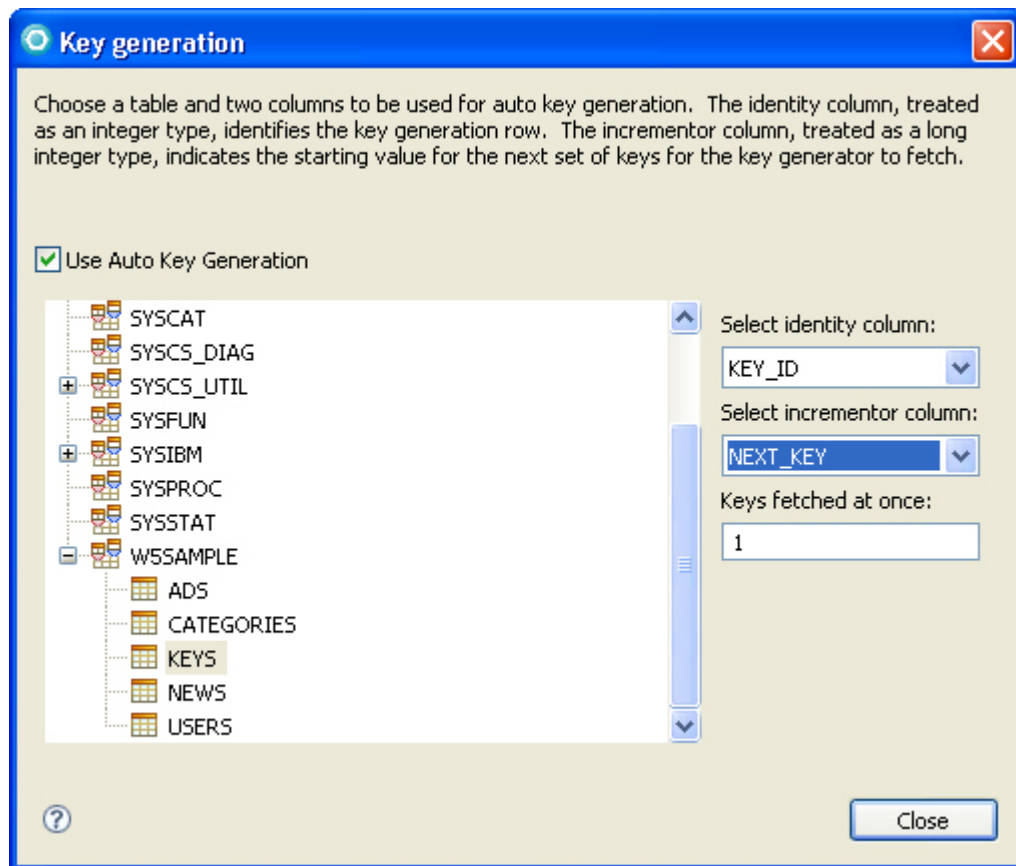
Set up the automatic key generation

The sample database provided with this tutorial has a key generation table named KEYS. Its two columns, as described above, will supply a new ID number for each new record. In these steps, you will set up the `create_record` relational record to take its ID number from the KEYS table.

1. Double-click the **new_record.jsp** page in the Project Explorer view.
2. Double-click the **create_record** relational record in the Page Data view. The Configure Relational Record window opens.
3. Click **Reuse metadata definition from an existing record or record list** and **Fill record with existing data from the database** then click **Next**.
4. Select your database connection from the drop down list, then click **Next**.
5. In the Configure Relational Record window, on the right is a list of links. Click **Auto generate key**. The Key generation window opens.
6. Select **Use Auto Key Generation**.
7. Click to expand the W5SAMPLE list, then click the **KEYS** table.
8. In the **Select identity column** field, click **KEY_ID**.
9. In the **Select incrementor column** field, click **NEXT_KEY**.

Since you only need one key for the ad's ID, you will leave the "Keys fetched at once" setting at 1. If you needed multiple keys, this setting would make the database select all of them at once.

The Configure Data Object window should look like this:



10. Click **Close** to close the Key generation window, then **Finish** to apply the changes. Now the ID field will be automatically generated for each new record. Next you must remove the ID input field so the user cannot enter a value.
11. Place the cursor in the top row of the input form table by clicking on the **Id:** text.
12. Click **Table** → **Delete** → **Delete Row**.
13. Save the page.

Optionally, if you wanted to see what key is being generated for you, you could instead delete the input component for the ID number and replace it with an output component bound to the ID column of `create_record`. Then, the automatically generated key would appear at the top of the form but the user would not be able to change it.

Test the completed tutorial

When you are ready to publish your Web application, you will need a server that will host it so that users can access the Web site through the Internet. However, to test your Web site, you can use an available runtime to simulate a server for testing purposes. To find out how to test the Web site, refer to “Lesson 1.3: Test the Web site” on page 13.

Lesson checkpoint

You have completed Lesson 2.4. In this lesson you learned how to set up automated key generation.

Module 2: Summary

You have completed Module 2: Add advanced features.

Lessons learned

You now know how to:

- Format database records on a Web page
- Add a component that allows users to upload files to a database from a Web page
- Use navigation rules to go from page to page automatically
- Use automatic key generation

Additional resources

For more information about this product or about JavaServer Faces and JSF technology, see the links below.

Related information

JavaServer Faces Technology

JSF in developerWorks

Display dynamic information on Web pages with JavaServer Faces

You have completed the Display dynamic information on Web pages with JavaServer Faces tutorial. This tutorial has taught you how to use JavaServer Faces to create Web pages that can use information from a database.

Lessons learned

By completing this tutorial, you learned about the following concepts and tasks:

- How to connect Web pages to a database.
- How to create pages that can create, read, update, and delete records from the database.
- How to send data from one page to another.
- How to format dynamic content on Web pages.
- About JavaServer Faces and the JSF components

Additional resources

If you want to learn more about the topics covered in this tutorial, consider the following sources:

Related information

JavaServer Faces Technology

JSF in developerWorks