

Lab 5: Sessions

In this lab, you will write a web application that for ordering pizza online. The application will have three “screens” of information:

- A static HTML form (OrderForm.html) that lets the user choose the size of pizza they wish to order
- An HTML form generated by a PizzaOrder JSP that lets the user enter in their delivery address and name
- An HTML page generated by an AddressInfo JSP that confirms the order

To maintain state between the three screens, your web application will use sessions to store an instance of a JavaBean that contains the order information. The PizzaOrder JSP will create the Bean instance at session scope and update the Bean’s pizza-size property. This JSP will then generate the HTML form that lets the user enter delivery information (e.g street address). The AddressInfo JSP will retrieve the Bean instance and update it with the delivery information and then generate the final confirmation page. See Figure 1:

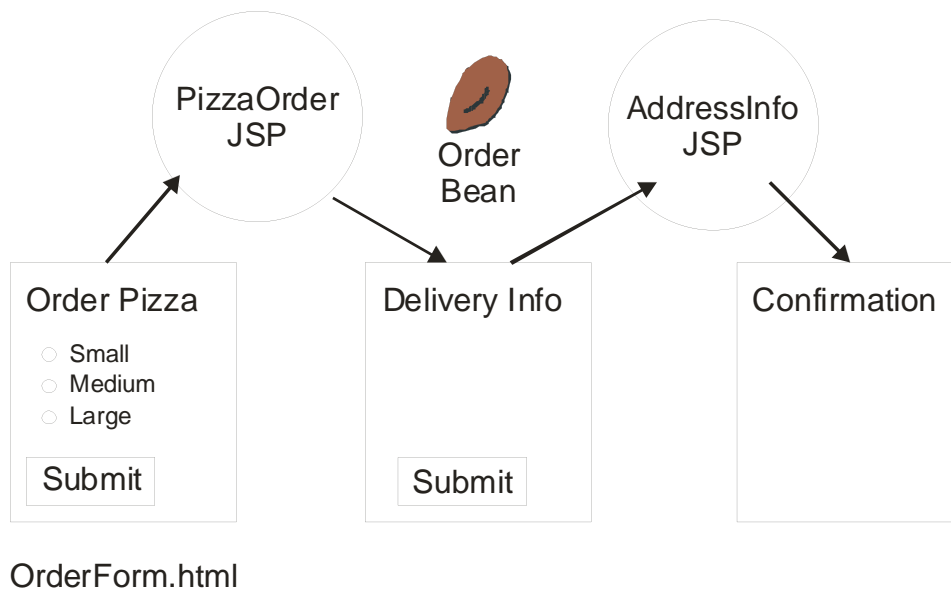


Figure 1: Application Screens

Objectives:

- To write JSP that use session scope Beans to maintain state

Steps:

1. Determine the following information:

HTML Deployment Directory: _____

Lab Installation Directory: _____

Servlet Deployment Directory: _____

2. Open a console command prompt window. (Under Windows, use the Windows Start Menu to open a Command Prompt window.) Change to the *Lab Installation Directory*. Then change to the lab05 directory. For example, under Windows:

```
cd \jspclass\lab05
```

3. Start by examining the provided OrderForm.html static HTML file, which defines an HTML form that lets the user choose the size of pizza. Especially note the form's action attribute and the name of the radio button group. Which JSP does the form invoke? Make sure you understand the flow as described in Figure 1 above. **Note:** if you are not using JRun's internal web server, delete the ":8100" text from the action attribute of the form.
4. Now examine the fully complete Order.java file in the *order* subdirectory. This JavaBean stores information about a pizza order: the pizza size and delivery information (name and address). Note that the application will initialize part of the order information from the PizzaOrder JSP and the rest from the AddressInfo JSP. To maintain state between these two pages, your application will store the Bean at session scope.
5. Next, create the PizzaOrder.jsp file and complete it so that it accomplishes the following:
 - Creates an instance of the Order Bean at session scope
 - Retrieves the *size* parameter from the request
 - Sets the *size* property in the Order Bean
 - Generates an HTML form that invokes the AddressInfo.jsp. This form should have two text input elements: one that lets the user enter their name, the other that lets the user enter a street address.

Note: if you are using JRun's internal web server, be sure that the form's action attribute includes the ":8100" port number in the URL.

6. Next, create the AddressInfo.jsp file. Then complete the JSP so that it:
 - Retrieves a reference to the Order Bean at session scope
 - Retrieves the *name* and *address* parameters from the request
 - Sets the *name* and *address* properties in the Order bean
 - Generates HTML for a order confirmation that displays the pizza size, customer name and address. Your JSP should read all of these properties from the Order Bean

7. Compile the Java code. From the console, in the lab05 directory, type:

```
javac order\*.java
```

Fix any compilation errors before continuing.

8. Next you need to deploy your files.
 - Copy OrderForm.html, PizzaOrder.jsp and AddressInfo.jsp to the **HTML Deployment Directory**
 - Create a subdirectory under the **Servlet Deployment Directory** named **order**
 - Copy Order.class into the newly created **order** directory

If you are using Windows, you can use the Windows Explorer to accomplish these tasks. Otherwise, use command-line commands like the Unix *cp* and *mkdir* commands.

9. Ensure that your application server is running. Check back to Lab One for details.
10. Test your application by starting your browser and entering the URL:

`http://localhost:8100/OrderForm.html`

Omit the “:8100” if you are not using JRun’s internal web server.

Then choose a pizza size and press Submit. Does the address-information page appear correctly? Enter a name and address and press Submit. Does the confirmation page correctly display all of the order information? If so, then your web application’s use of sessions is working properly!

Optional

1. If your browser supports it, disable cookies. (To disable cookies in Internet Explorer 5 and later, choose Tools – Internet Options – Security. Then add <http://localhost> to the list of “Restricted sites”. You can then use the “Custom Level” button to set the browser so it prompts when a site tries to set a cookie – that will let you see the process in more detail. Close and restart the browser to apply the new settings.)

Then try running your program again – it should fail, since the servlet API by default uses cookies to maintain the session. To fix it, re-code the PizzaOrder JSP so that it uses URL rewriting for the URL of the AddressInfo JSP. Then try it again with cookies disabled.

2. Update your web application so that the user can choose pizza ingredients, e.g. cheese, beef, etc. The easiest way to do this is to modify the OrderEntry form. Be sure to update your Order Bean class to contain the new information! You will also need to modify the generated confirmation page to reflect the changes.

