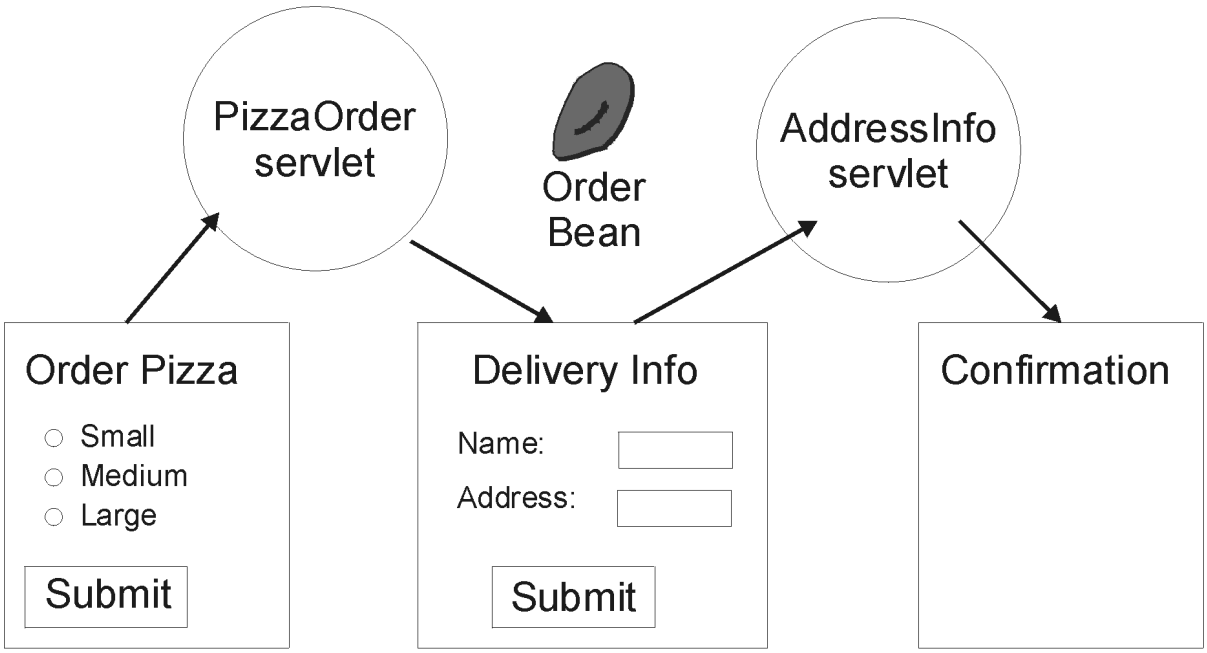
Servlet Lab 4: Sessions

In this lab, you will write a web application 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* servlet that lets the user enter in their delivery address and name
- An HTML page generated by an *AddressInfo* servlet that confirms the order

To maintain state between the three screens, your web application will use the servlet session API to store an instance of a JavaBean that contains the order information. The PizzaOrder servlet will create the Bean instance, update it with information about the order and then store the Bean instance in the session. This servlet will then generate the HTML form that lets the user enter address information. The AddressInfo servlet will retrieve the Bean instance and update it with the delivery information and then generate the final confirmation page. See Figure 1:



OrderForm.html

Figure 1: Application Flow

Objectives:

- To write servlets that maintain state using the servlet session API

Steps:

_1. Ask your instructor for the following information:

BEA Installation directory: _____

BEA Username and password: _____

Lab Installation directory: _____

If your machine has the standard lab setup, the directories will be something like:

```
BEA Installation directory:    c:\java\bea
BEA Username and password:     administrator
Servlet Deployment directory:  c:\j2eeclass\DefaultWebApp\WEB-INF\classes
Lab Installation directory:    c:\j2eeclass
```

_2. Start by examining the provided **{Lab InstallationDirectory}/servlab04/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 servlet does the form invoke? Make sure you understand the flow as described in Figure 1 above.

_3. Now examine the fully completed **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* servlet and the rest from the *AddressInfo* servlet. To maintain state between these two servlets, your application will use the servlet session API.

_4. Next, examine the partially completed **PizzaOrder.java** file. The provided code creates an output stream variable and then generates part of the required HTML. Please examine the provided code carefully so you understand how it generates the next page in the web application. Refer back to Figure 1. Draw a sketch of the resulting HTML page and write down the name of the input variables on a piece of paper.

_5. Complete the *PizzaOrder* servlet so that it accomplishes the following:

- Retrieves the string from the request that indicates the pizza size
- Creates an instance of the Order Bean using the no-argument constructor
- Sets the *size* property in the Order Bean
- Retrieves (creates) a reference to a Session object
- Stores the OrderBean reference as an attribute in the session. You can choose whatever attribute name you want, but write it down for future reference.

_6. Now examine the **AddressInfo.java** file. This servlet responds to the address-information HTML form that was generated by the PizzaOrder servlet. The *AddressInfo* servlet needs to complete the order information by reading the customer's name and address and storing them in the Order object. But first, it must retrieve the Order object reference from the session. Please examine the provided code carefully so you understand how it generates the confirmation page. Draw a sketch of the resulting HTML. Then complete the servlet so that it:

- Retrieves the *name* and *address* strings from the request
- Retrieves a reference to the Session object. Be sure to report an error if the session object does not exist. For example, throw a ServletException.
- Retrieves the Order Bean reference from the session
- Sets the *name* and *address* properties in the Order bean

_7. Your next task is to package your web application into a .war file and deploy it using Ant.

Examine the provided **build.xml** file and note how it defines three *targets*: a *build* target that compiles your Java files, a *makewar* target that uses Ant's *war* task to create a .war file containing your files, and a *deploy* target that cleans up any previous version of the web application and then copies the .war file to the correct directory. Note also that the *deploy* target depends on the *makewar* target -- that means that Ant will run *makewar* first. Likewise, the *makewar* target depends on the *build* target.

_8. To deploy using Ant, open a command prompt window (choose Run from the Start menu and enter "cmd" with no quotes), change to the **{Lab Installation directory}/servlab04** directory and then run Ant. For example, assuming standard lab directories:

```
cd \j2eeclass\servlab04
ant -emacs
```

Ant will execute the commands in the **build.xml** file and report any error messages, which you must correct before continuing. You can then use Windows Explorer to examine the WebLogc directories for the files that Ant just copied for you.

_9. Test your application by starting your browser and entering the URL:

```
http://localhost:7001/servlab04
```

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:

Optional labs are for students that wish to explore topics further with minimal guidance from the lab write-up.

_1. Disable cookies in the browser and run the application again -- it should fail, since the container will not be able to maintain state between requests. Then add code so that the URLs are encoded and retest. **Note:** when you test, set the size to something other than "Large", which is the default setting.

Disabling Cookies in IE Version 6:

- Choose Tools - Internet Options
- Click the General Tab, then press the "Delete cookies" button. Then press Apply.
- Click the Security Tab, then click the "Local Intranet" icon. Then press the "Sites" button. Uncheck all of the checkboxes. Press OK then Apply.
- Click the Privacy Tab, then press the "Advanced" button. Check the "Override automatic cookie handling" box, then select "Prompt" for both settings. Press OK as necessary to close all dialogs.
- Close Internet Explorer and restart it.

Disabling Cookies in IE Version 5:

- Choose Tools - Internet Options
- Click the Security Tab, then add "http://localhost" to the list of "Restricted sites". Then press the "Custom Level" button and adjust the settings so that the browser prompts when a site attempts to set a cookie. Press OK as necessary to close all dialogs.
- Close Internet Explorer and restart it.

_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 class to contain the new information! You will also need to modify the generated confirmation page to reflect the changes.