Delete Lab

Lab Summary: In a previous lab, you created JSP pages that displayed data from your database (your web_users table and your other table). In this lab, each of your JSP pages that data display will also provide delete functionality, similar to JSP page delete.jsp that is shown below (except you will link a delete icon, not the word “delete”):

<table>
<thead>
<tr>
<th>Delete?</th>
<th>Id</th>
<th>Last Name</th>
<th>Credit Limit</th>
</tr>
</thead>
<tbody>
<tr>
<td>delete</td>
<td>174</td>
<td>James</td>
<td>Bond</td>
</tr>
<tr>
<td>delete</td>
<td>279</td>
<td>Matt</td>
<td>C</td>
</tr>
<tr>
<td>delete</td>
<td>131</td>
<td>santa</td>
<td>claus</td>
</tr>
</tbody>
</table>

If you hover your mouse over the first “delete” link, you will see this URL:

data.jsp?deleteCustId=174

Functional Requirements:
As mentioned in the summary, each of your JSP pages that display data shall now provide user functionality to delete each row.

- Each data row shall have a delete icon that, when clicked, self posts to the JSP page, passing in a URL parameter with the id of the record to be deleted.

- Post back code (in same JSP page) deletes the specified record then redisplays the data from the database – along with a message that either confirms the deletion or explains an error that occurred.

Design Specifications:
In addition to the above functional requirements, here are design specifications that you must also follow.

- The code in your new/modified data display JSP pages shall be short, pushing as much code as possible to reusable java classes.

- Your old view method (that returns HTML table code, that does not support delete functionality) shall remain in your “view” package. Your new view method (that also returns HTML table code, that supports delete functionality) shall be in the same class as your old view method.

- This new view method shall not make any assumptions about what is or is not on the JSP page. (Think of the method as a worker, not a boss. The worker has no say in what the boss does. The JSP page is the boss.) This means that:
  - the JSP page shall pass the name of the delete icon to the method
• Your method for deleting from each table shall be kept in one of your “model” packages, e.g., “modelWebUser”
  o This delete method shall be passed in the id of the record to be deleted.
  o This delete method shall pass back a string that either
    ▪ Is “” (empty string), which means the delete worked as expected -OR-
    ▪ contains an error message.

[You can test this by trying to delete when you are not tunneled in or by creating an error in your SQL delete statement.]

Suggestions About How to work:

(1) **Back up** your project (as always) before starting anything.
(2) **Download the sample code** from the zip file and install it on your PC (as it’s own project). Leave this project untouched so it continues to function properly and so you can reference it for help.
(3) Since you are new to programming, it is very important that you make just a few changes between running and testing your code each iteration.

Suggestions About How to Debug

a. **SQL**: in this lab, your java code will build a SQL statement that deletes a record. The SQL must be syntactically correct and it must reference table names and field names exactly as you defined them in your database. If your SQL is not correct, your java/jsp program will throw an exception when you try to run it. You just have to be sure to display the exception error message, e.getMessage(), either on the JSP page (out.print) or in the glassfish log (System.out.print). Another technique is that you can type your proposed SQL command right into the query area at the top of MySqlWorkbench and try to execute it. MySqlWorkbench provides errors at the bottom, if there are any.

b. **JSP**: thankfully, the NetBeans editor gives you warning about any JSP syntax errors that you may be typing in. However, you can still get unexpected results or runtime errors. As mentioned before, your best option is to add out.print statements into your JSP pages – this will help you determine which code is being run and the values of certain variables. A couple of things to check in your JSP pages:
  i. Check that your page posts to itself (action attribute of form references name of jsp page) – especially if you just changed the name of the jsp page.
  ii. Check that when you do request.getParameter("name_of_input_tag"), you spell the name of the input tag exactly as you have it spelled below inside the <form> section of your HTML. If you misspelled the name of the input tag that you are checking for postback, your page will always act like first rendering.
  iii. If you have a weird java compilation error that confuses you, click on your first left brace { and check to see it’s matching right brace }. Chances are that you are missing a brace or have an extra one. The weird error you are getting is actually in the servlet that is created from your JSP page.
c. **Java:** If you are debugging a java class, you cannot use `out.print()`, but you can use `System.out.print()`. I usually precede the message with stars so I can find the messages which appear at the bottom of the glassfish log in the output pane of NetBeans. For example,

```
System.out.println("exception in ..., msg is: " + e.getMessage());
```

When you no longer need to bebug some code, I recommend that you comment out the debug statement rather than delete it (who knows, you might need to debug again in the same place).

**Submission:** after publishing and testing, submit a zip file to blackboard.