CIS 3308 (Kyvernitis) Delete Lab

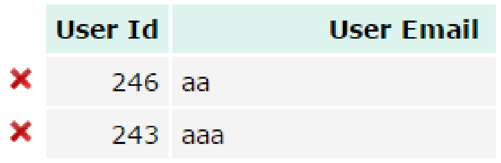
Overview:

In this lab, you will modify each of your three list pages (users.jsp, other.jsp, assoc.jsp) so that every row has a delete icon (in addition to the edit/update icon from the previous lab). When the user clicks on a delete icon, the page posts to itself, deletes the record, then redisplays the data.

Functional Requirements:

other.jsp, users.jsp, and assoc.jsp:

- Each row of data shall have a "delete" icon (that might look like this). (Your page will also have an edit icon for each row.)



	User Id	User Email
✖	246	aa
✖	243	aaa

- Each delete icon shall be surrounded with a link (such as shown below). Each link shall self-invoke the JSP data list page, passing in the id of the row that is to be deleted.

  `<a href='users.jsp?deleteId=246'><img src='icons/delete.png'></a>` for the first row where id = 246
  `<a href='users.jsp?deleteId=243'><img src='icons/delete.png'></a>` for the second row where id=243

- When the user clicks on one of the delete icons, the page shall "call itself" with a delete Id parameter in the url and the record shall be deleted before the page full of records is built/displayed.

Design Specifications

1. **other.jsp, users.jsp, and assoc.jsp** shall
   a. Declare a database connection object, pass that to classes/methods as needed, and then close the database connection object when it is no longer needed (no database connection leaks).
   b. Have as little code as possible (delegating functionality, wherever possible, to reusable java classes). Note: a JSP page is the only place where you can use JSP implicit objects, so the JSP page must handle input (request.getParameter), output (out.print), session (get/put logon information), and security/redirect (response.sendRedirect).
   c. All JSP code shall be before the user interface and the user interface shall have HTML code with <%=xxx%> JSP created string values wherever appropriate.
   d. The delete operation's message (error or confirmation) shall be displayed on the page after every delete.
2. From last lab, your web application has a **model package** with 3 sub-packages, with each sub-package named for one of your database tables (not "assoc", not "other"). For each of your database tables, add a delete method and place it in the same class as your insert and update methods (in the proper package, of course). Your delete method shall return an error message or confirmation message.

3. From previous labs, your web application has a **view package** with classes to display data from the database. For each of your three database tables, you have one or two list methods that return an HTML table full of all the data from the database table. In this lab, you will **create a new list method for each of your 3 database tables** (don't break the other methods, add a new one) that produces an "enhanced" HTML table, one that has the delete and update icons built in. It is VERY BAD DESIGN (and not allowed) for you to have your view method make ANY assumptions about what may or may not be found in the JSP page. This means that your JSP page needs to pass in things like name of the CSS class for the <table> tag, name of the JSP page (for self invoking), name of the query parameter (e.g., deleteId), etc.

Labs Page (Blog)

- Your labs page shall always include a blog for each lab, explaining what you learned and linking to your work.

Programming Style

- Follow what is listed under "Requirements for All Labs and Project" (on the 3308 labs page).
- Any JSP page that accesses your database will need one DbConn object. That JSP page should instantiate the object (constructor opens the connection), use it, and then be sure to close it (for EVERY CODE path). This will prevent your code from having any database connection leaks.

Homework submission

- After getting your code to work locally, publish it to cis-linux2 and test it.
- Submit a zip file of your whole project to blackboard.
- Make sure that you have a link from your labs page to a screen capture of your data model (that you created using mySqlWorkbench).

Suggested Approach

assoc.jsp:

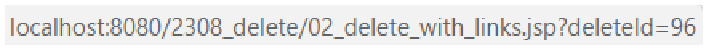
1. Begin by working on assoc.jsp. This is because nothing points to your associative table, so the database should never throw an exception when anyone tries to delete a record. Make a backup copy of assoc.jsp (e.g., assoc_bak.jsp – you can delete it later after everything is working).

2. Modify assoc.jsp so that it first tries to read a parameter, e.g., request.getParameter("deleteId"). If this returns a non-null and non-empty result, then invoke a delete method prior to listing the data. You can test this by using URL hacking (just type "?deleteId=xx" to the right of the URL in the browser). Work on your delete method (in the same class as your insert/update methods). Make sure that your delete message passes back an error or confirmation message. Remember that you are to have "user friendly" database error messages for any error that might occur in bug free code. This includes "database unavailable" and (for users.jsp and other.jsp) foreign constraint error messages like "you can't delete this user because they have purchases" or "you can't delete this product because it has purchase records associated with it".

3. Once you have delete working without the delete links, then start working on your new list method that will generate the linked delete icons for each row of data.

   ```
   <a href='users.jsp?deleteId=246'><img src='icons/delete.png'></a>
   ```
   for the first row where id = 246

4. Check that the code around the delete icon is syntactically correct by right clicking and "Viewing Source" (especially in firefox, chrome is not as good at showing you syntax errors). Another way to check is to hover over each delete icon and see the proper javaScript call in the bottom left of your browser. Each row have a link around the delete icon, like this:

   localhost:8080/2308_delete/02_delete_with_links.jsp?deleteId=96

5. Once you get delete working from the better user interface (and it's working with your layout), make sure that the list method you just wrote does not make ANY assumptions about what may or may not be on the JSP page, for example, name of javaScript function, name of delete icon. Any information like this needs to be passed from the JSP page down to the new list method.

Repeat this process for **other.jsp**.

- The only extra work for other.jsp is that it is very possible for a user to try to delete a record (from your "other" table) that is being referenced by a record from your associative table. The database will throw a foreign key exception and this is what we want – we do not want to delete a record that is being referenced. Replace the technical database exception error message (something about foreign key violation) with a user friendly error message. For example, "You cannot delete this product there are purchase records for this product".

- Many students will ask if they should write code to automatically delete the purchase records in question and then delete the product record. In some cases, this might be the desired functionality, but certainly not before the user is made aware of the (possibly big) mistake they might be making. In real applications (when the user really does want to delete a product that has purchases), the system might mark the product for deletion, have that product not come up in product lists (e.g., for new purchases). At the end of some time period (maybe year end), there may be an archive operation where old purchase records are archived and removed from the operational database and, at that point, products that are marked for deletion and now have no purchases might be physically deleted from the purchase table. Think about an accounting system – you would NEVER delete a purchase record like that. The only way to "delete" a purchase record would be an archive operation.

Repeat this process for **users.jsp**.

- This page also has the possibility of attempting to delete a user that is being referenced by your associative table. Replace the technical database exception error message (something about foreign key violation) with a user friendly error message. For example, "You cannot delete this user because they have made purchases".