PROG2240: Java Web Tech

Assignment 4

Winter 2017

Assignment Type: Pair Programming

Due Date: Week 14 (end of class)

Before you begin:

- Backup your previous XXYYClub solution.
- Read the Important Notes section of A1.
- Read and study sample book applications from Chapter 12.
- Review the scripts for creating the ***murach*** and ***music*** databases (see Figure A-8 on page 709 and Figure 3-15 on page 81).
- Do the **Individual Exercise 12-2** on page 419-420. Review the **DBUtil** class. Then, do the **JUnit Lab** and demo to your partner.
- Print **LastPage_PPLog.pdf**. Date and sign Sections B and C of your **Pair Programming (PP) Experience Log.**
  - **Note: Your team will get a 50% penalty if the PP Log is incomplete or missing.**
- In this assignment, you are expected to use JSP standard action/tags, JSP EL and JSTL. *Do not* use scripting elements.

Background: In this assignment, you will perform unit testing and persist the club member information to a MySQL database. You will develop **Member Maintenance** feature that will display and add/update/delete member records using JDBC/Connection Pooling.

Additional Requirements:

- In this assignment, you are expected to design and develop the code with only a few details. Follow the MVC pattern and other best practices you learned from the previous assignments and exercises.

- Within the **XXYYMemberAdminController** servlet (controller), **do not use the processRequest()** method generated by NetBeans. The **doGet()** method must implement actions that does not modify the database table (e.g., display members, display member input form, display confirm delete), while **doPost()** method must implement actions that modify the database table (e.g., insert, update and delete).

Tasks:

1. Setup XXYYClub NetBeans project

a. Add package and Java classes

- Download and unzip A4Files.zip
- Copy Member.java to the existing **club.business** package.
- Copy MemberDB.java, ConnectionPool.java and DBUtil.java to the existing **club.data** package.
    - Study all the code provided. *Note: These classes are similar to classes in Exercise 12-2: ch12_ex2_userAdmin.*
    - **Important:** Do not modify any of these classes.
- Use the above classes and methods as part of your solution.
- You will add **XXYYMemberAdminController** servlet with URL pattern **XXYYMemberAdmin**, under the **club.admin** package (see Step 3 below).

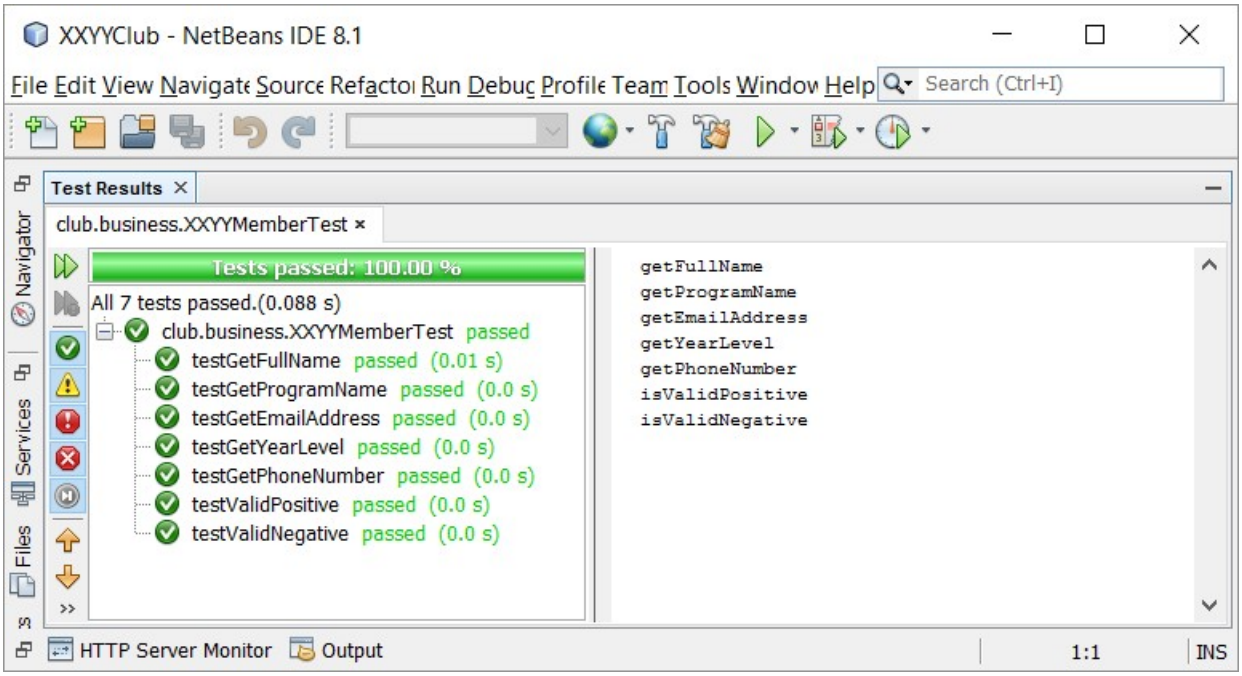
b. Add JSTL 1.2.x Library to your project (if not previously added)

c. Add JUnit Test Library to your project

---

2. JUnit Testing

- Create a JUnit test class named XXYYMemberTest within the club.business package (under the "Test Packages" node).
- Generate test methods for all the getter/setter methods of Member class provided.
    - Each test method must execute the appropriate getter and setter methods.
    - When writing test methods, use non-default / non-blank values, if possible.
- Add one positive test (i.e., method call returns *true*) and one negative test method (i.e., method call returns *false*) for the isValid() method.
    - Name the test methods testValidNegative() and testValidPositive().
    - Hint: For the positive test method, call the constructor with 2 valid arguments.
- Important: Every test methods must have an **assertEquals()** method.
- Run your unit test cases and verify that all test cases passed. A sample JUnit testing output is shown below:



3. Member Maintenance

For this task, you will create a series of pages that allow you to display, add, update, or delete a member of your club.

a. In MySQL Workbench, run the **create_memberdb.sql** script.

b. In NetBeans, add **MySQL JDBC Driver** library to your project.

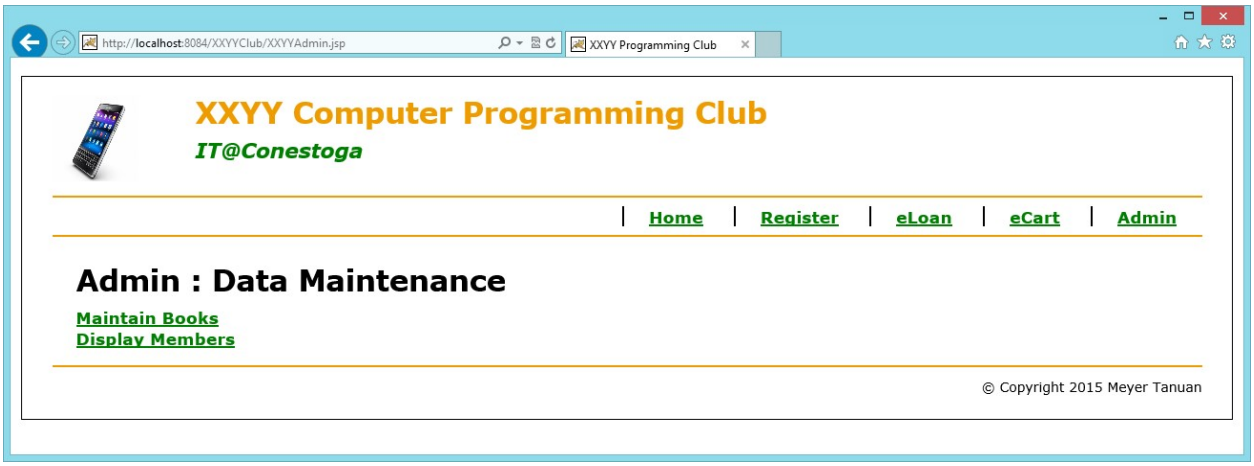
c. Add a <Resource> element to your **context.xml** as follows:

```
<Resource auth="Container"
          driverClassName="com.mysql.jdbc.Driver"
          logAbandoned="true"
          maxActive="100" maxIdle="30" maxWait="10000"
          removeAbandoned="true" removeAbandonedTimeout="60"
          type="javax.sql.DataSource"
          url="jdbc:mysql://localhost:3306/memberdb"
          name="jdbc/memberdb" username="root" password="sesame"
/>
```

*Note: The data source **name** is **jdbc/memberdb** to match the code provided in ConnectionPool.java.*

d. In your existing **XXYYAdmin.jsp** page, add a new hyperlink to display the list of members

- The Display Members link calls the **XXYYMemberAdmin** servlet with an **action** parameter with value **"displayMembers"**.
- Note: Ensure that your links work even when cookies are disabled in the client browser. *Hint: Use the <c:url> tag*

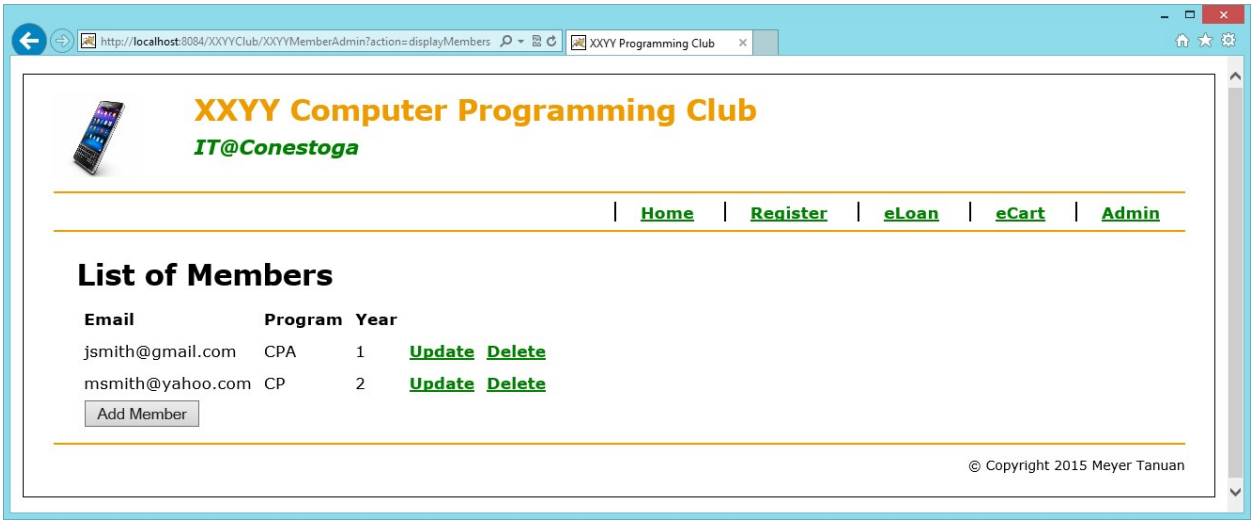


e. Add a new **XXYYMemberAdminController** servlet with URL pattern **XXYYMemberAdmin**.

- This servlet must handle all CRUD actions
- If **action** parameter is not defined, the default **action** is **"displayMembers"**, which will forward control to the **XXYYDisplayMembers.jsp**.

f. **XXYYDisplayMembers.jsp:** Sample screenshot to display all members

URL:
http://localhost:8084/XXYYClub/XXYYMemberAdmin?action=displayMembers



Operations:

- The **Display Members** page (**XXYYDisplayMembers.jsp**) contains a link that leads to the Members page that can be used to add, update, or delete products.
- To add a new member, the user selects the **Add Member** button. Add code to validate input, to ensure a name and valid email address is provided.

  Sample Add Member Link:
  http://localhost:8084/XXYYClub/XXYYMemberAdmin?action=addMember

  Optional: Modify the existing Register link of your XXYYBanner.jsp page to match the above Add Member link.

- To edit an existing member, the user selects the Edit link for the member. This displays the Update Member page with all existing data for the member displayed. Then, the user can edit **any entries** and click on the Update Member button to update the data for the existing member. If the email does not exist, insert a new record. *Hint: See sample screen and code from Exercises 12-2.*

Sample Update Link:
http://localhost:8084/XXYYClub/XXYYMemberAdmin?action=displayMember&email=jsmith@gmail.com

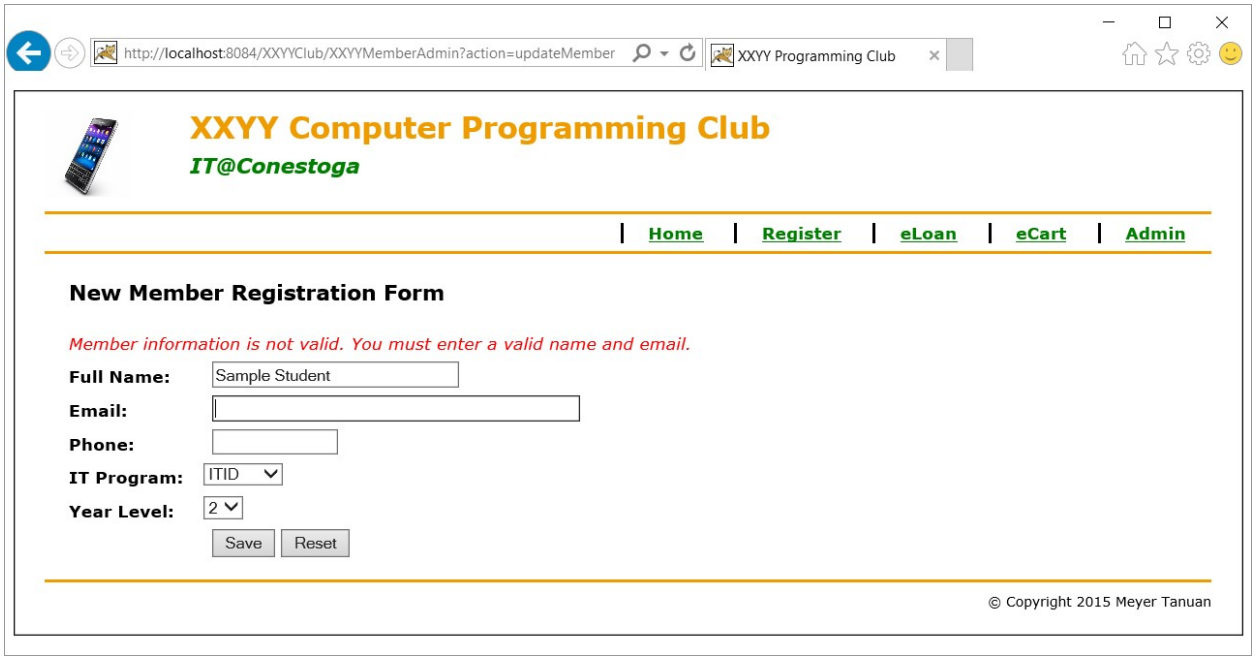
- To delete a member, the user selects the Delete link for the member. This displays the **Confirm Delete** page. Then, if the user confirms the deletion by selecting the **Yes** button, the **member** is deleted and the **Display Members** page is displayed to reflect the new data. If the user selects the **No** button, the **Display Members** page is displayed.

Sample Delete Link:
http://localhost:8084/XXYYClub/XXYYMemberAdmin?action=confirmDeleteMember&email=jsmith@gmail.com

g. **XXYYMember.jsp:** This page will be used for both **Add a Member** and **Update a Member**.

Update Member button:

- Use the **isValid()** method of Member object as part of your input validation. See sample *error message* in the screenshot below.
- If the email does not exist in the database table, insert a new record. Otherwise, update the existing record.



h. **XXYYConfirmMemberDelete.jsp**: Design your own Confirm Delete page with the following content:

- **Header Title**: Do you want to delete this member?
- **Display member information**: Full Name, Email, Phone, Program and Year Level
- **Buttons:**
- **Yes**: This button will permanently remove the record.
  - Sample URL: http://localhost:8084/XXYYClub/XXYYMemberAdmin?action=deleteMember&email=jsmith@gmail.com
- **No**: This button will cancel the delete operation and display all members
  - http://localhost:8084/XXYYClub/XXYYMemberAdmin?action=displayMembers

Using the *Exercise 12-2* as a guide, complete the Member Maintenance with the following specifications:

- Use a **Member** class provided to store the member data.
- Use a **MemberDB** class provided to display the list of members.
- Use a **member** table provided in the **create_memberdb.sql** script as a starting point for the initial members.
- All the actions (display, insert, update, delete) for Member Maintenance is managed by the **XXYYMemberAdminController** servlet**.**
- Use server-side validation to validate the user entries required. The *email* of the Member *must be unique* for the **Member** table.

---

4. Test and submit your Member Registration Web Application

- Print and submit sample screenshots of your **JUnit Test result** and your running application (**Display All, Add, Update, Delete**).

---

5. Submit your entire XXYYClub solution to eConestoga

- Submit your entire XXYYClub solution (**XXYYClub.zip** file) to the eConestoga (**A4_DropBox**) on or before the due date.

Assignment Submissions

Important: You must submit your printouts (Cover Page, screenshots, PP Log) and upload your solution (XXYYClub.zip). Do not print your code.

Sequence	Item
1	A printout of the A4_CoverPage.pdf (a total of 3 pages) with all the relevant sections filled in.
2	A printout of 4 sample screenshots similar to the above screenshots (with the **title** displayed). The screenshots should be large enough for marking (Use Alt-PrtSc to copy and paste screenshot to MS Word, one screenshot per page).
3	A printout of the LastPage_PPLog.pdf with all the relevant sections filled in.
4	Zip and upload your assignment solution XXYYClub.zip (where XXYY is your team initials in upper case letters) to your appropriate section's **A4_Dropbox** on **eConestoga**. Upload using the **driver's** eConestoga login. **Note:** This zip file must be ready to run on NetBeans 8.1 without modifications when unzipped to C:\murach in any Doon IT Lab. *Otherwise you will get 5% penalty per upload-related error.*