

Assignment 2: JSP and State Persistence

Overview:

This assignment is intended to expand upon the previous servlet assignment by incorporating more of an MVC approach. The state will be maintained (session management) and transaction functionality will be implemented.

For this assignment, you will need to move the previous html creation accomplished in the servlet and move the html into the jsp. No html is allowed in the servlets unless it is to create dynamic html (e.g., rows in table based on an array). Individual page and servlet requirements are described below.

Assignment Description:

Create a web application that uses servlets as the controller component and jsp pages as the view component for interaction with the user. The following is an outline of the pages and servlets that will make up the web application as a whole.

1. A java class for the session bean
2. A page for user login (use jsp)
3. A servlet that handles login (e.g., LoginControllerServlet.java)
4. A page for new user registration (use jsp)
5. A servlet that handles registration (e.g., RegistrationControllerServlet.java)
6. A servlet to serve as the controller for the cardTrader.jsp (e.g., CardTradingControllerServlet.java)
7. A jsp page for card collection and available cards (e.g., cardTrader.jsp)
8. A servlet to handle the transaction (e.g., TransactionControllerServlet.java)
9. A servlet to handle logout

A few items to note:

1. Java class for the session bean. At a minimum include the following with the appropriate getters and setters:
 - `private String userName = "";`
 - `private String message= "";`
 - `private HashMap<String, String[]> cardCollection = new HashMap<String, String[]>();`
 - `private HashMap<String, String[]> cardsForSale = new HashMap<String, String[]>();`
2. **login.jsp**
 - Create / obtain session bean object with the scope of session
 - User enters a username/password, clicking the submit button on the form should lead to the LoginControllerServlet.java.
 - Clicking the registration button or link should lead to the registration page

- Below the title / above the username, create a placeholder with something like `bean.getMessage()`. This will be used in #3

3. LoginControllerServlet.java

- Obtain the session bean object
 - `HttpSession session = request.getSession(false);`
 - `SessionBean sessionInfo = (SessionBean) session.getAttribute("bean");`
- Obtain the username and password and check that they equal either of the following combos:
 - `username = "pat" password = "pat"`
 - `username = "bob" password = "bob"`
- If one of the combos are equal, put the username into the session bean and forward on to the `CardTradingControllerServlet.java`
- If they do not equal, put an unsuccessful message into the session bean, send the user back to the login page, and have the mssg show up under the title.
 - Once `login.jsp` loads the mssg, set the mssg to ""
 - This can be tested by reloading the page - the mssg should go away

4. registration.jsp

- Create / obtain the session bean object
- Create a simple registration form (username, password, confirm).
- Below the title, create a placeholder with something like `bean.getMessage()`. This will be used in #5
- Clicking the Register button form should lead to `RegistrationControllerServlet.java`.
 - Accomplish this as a form action or with javascript.

5. RegistrationControllerServlet.java

- Create / obtain session bean object
 - HttpSession session = request.getSession(false);
 - SessionBean sessionInfo = (SessionBean) session.getAttribute("bean");
 - The bean should have been created at the login.jsp but this handles the issue if someone navigated directly to registration
- Check that the username is not pat, bob, peg, or null / empty.
 - If their entry was pat, bob, peg, or null / empty:
 - Put their username into the mssg stating the username is taken
 - Send them back to registration.jsp and have the mssg show up under the title.
 - Once registration.jsp loads the mssg, set the mssg to ""
 - This can be tested by reloading the page - the mssg should go away
 - If null or empty - send them back to registration with mssg
 - If not go to the next step
- Check that the password and confirm password match and are not empty.
 - If so - put the username into the session and forward them on to CardTradingControllerServlet.java
 - If not - inform the user of a non-match by putting the mssg into the bean and send them back to registration.jsp where the mssg will be displayed.
 - The mssg should go away when the page reloads

The image shows three screenshots of a web form titled "Registration". Each form has fields for "UserName:", "Password:", and "Confirm:", followed by "Register" and "Back to Login" buttons.

- Left Screenshot:** Displays the message "*** Don't leave empty ***" in blue text above the input fields.
- Middle Screenshot:** Displays the message "*** username taken ***" in blue text above the input fields.
- Right Screenshot:** Displays the message "** Passwords don't match **" in blue text above the input fields.

6. CardTradingControllerServlet.java

- Obtain the session bean object
- If the referer comes from login.jsp or registration.jsp (if not, send them back to login.jsp)
 - Populate the session bean (see step 1 hashmaps) with the following:
 - setCardCollection("1983 Topps Tony Gwynn RC", "482", "75.21", "1");
 - setCardCollection("1984 Fleer Update Roger Clemens", "27", "120.00", "2");
 - setCardCollection("1983 Topps Ryne Sandberg", "83", "20.00", "3");
 - setCardsForSale("1984 Donruss Don Mattingly", "248", "40");
 - setCardsForSale("1984 Donruss Joe Carter RC", "41", "8");
 - setCardsForSale("1984 Donruss Darryl Strawberry", "68", "12");
 - Option - Dynamically build cardTrader forms in servlet and store it in the bean to be sent to cardTrader.jsp (a separate class to handle this issue would work well in this situation).

7. cardTrader.jsp

- Obtain the session bean object
- When the jsp loads, include the username with the title
- The cardTrader.jsp must list the current card collection and the available cards to buy as listed in step #6.
 - Should/could have been built in CardTradingControllerServlet.java or a separate class that specifically handles the creation of forms
 - Note - the forms / form data cannot be made statically.
- The action drop down box on the page should include "Buy" and "Sell".
- Clicking the "Make Transaction" button should lead to the TransactionDetailsServlet.java and send the following:
 - Card name
 - Card number
 - Value
 - Amount owned
 - Action (buy or sell)
 - Quantity
 - Hint the use of hidden fields really helps
 - `<input type="hidden" name="cardName" value="<%= cardName %>">`
- All fields in the image should be included (order/location replication not required)
- Under the title, put messages about incorrect transactions as handled by the TransactionControllerServlet.java - e.g., Sorry, you can't sell more than you own.

Current Card Collection for Pat

Sorry, you can't sell more than you own.

Card	Card Number	Value	Amount Owned	Total Value	Action	Quantity	
1983 Topps Ryne Sandberg	83	\$20.00	3	\$60.00	sell ▼	20	Make Transaction
1983 Topps Tony Gwynn RC	482	\$75.21	1	\$75.21	buy ▼		Make Transaction
1984 Fleer Update Roger Clemens	27	\$120.00	2	\$240.00	buy ▼		Make Transaction

Available Cards to Buy

Card	Card Number	Cost	Quantity	
1984 Donruss Joe Carter RC	41	\$8		Purchase
1984 Donruss Darryl Strawberry	68	\$12		Purchase
1984 Donruss Don Mattingly	248	\$40		Purchase

Logout

8. TransactionControllerServlet.java

- Obtain the: cardName, cardNumber, cardPrice, ownedCards, buySell, quantity

- Check if any of the previous was null or empty - if so, send back with a message (message goes away on refresh of page).
- Check if they are trying to sell a negative quantity or sell more than they own. If so, send back with a message (message goes away on refresh of page).
- Check if IfReferredBy came from the cardTrader.jsp or the transactionDetails.jsp (see code example at end of this document).
 - If from cardTrader.jsp
 - Make appropriate actions/calculations and then redirect to transactionDetails.jsp.
 - If from transactionDetails.jsp
 - Make appropriate actions/calculations/updates (e.g., update the session bean HashMap)
 - Rebuild the form (you might have done this in the CardTraderController.java or you might have created a separate form class.
 - Redirect to cardTrader.jsp
 - From this point, cardTrader.jsp should show the updated purchase or sale.

9. **TransactionDetails.jsp** - convert the html from TransactionDetailsServlet.java

- Provide a summary of the purchase which includes all fields shown in the image.
- Clicking the "Confirm Transaction" submit button should lead to the TransactionControllerServlet and pass the necessary data.

Transaction Details					
Card	Number	Buy/Sell	Quantity	Card Price	Total Cost
1983 Topps Tony Gwynn RC	482	buy	2	\$75.21	\$150.42
Bank Account Information					
Account Holder Name	<input type="text"/>				
Routing Number	<input type="text"/>				
Confirm Transaction		Cancel		Logout	

10. **LogOutServlet.java**

- Used whenever someone clicks the Logout link / button. Redirect user to login page.

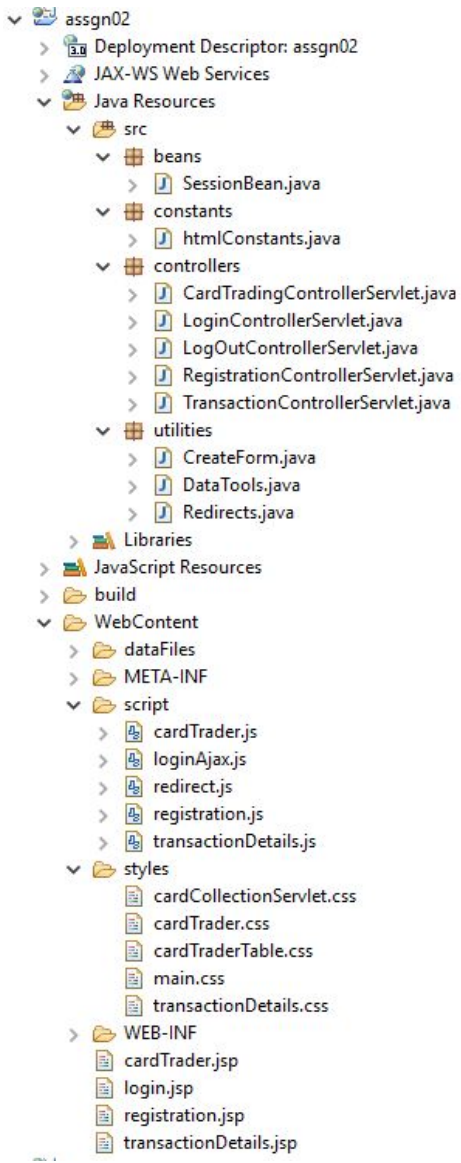
Assignment Submissions:

Your completed application must be loaded and available for public viewing on your AWS server.

What to submit using Blackboard:

1. **HW2.war** - An archive of the entire web application (project) stored in a standard war file. When creating the war file you must include your java source files. This war file will be imported into Eclipse for grading.
 - To create a war file, right click your project > Export > WAR file - **make sure you check Export Source files before clicking finish!!**
2. **info.doc** - Document with the following assignment information:
 - Link to your Login page on your Google server
 - Explanation of status and stopping point, if incomplete.
 - Explanation of additional features, if any.

Workspace and code examples below

 <p>The Project Explorer shows the following structure for 'assgn02':</p> <ul style="list-style-type: none"> Deployment Descriptor: assgn02 JAX-WS Web Services Java Resources <ul style="list-style-type: none"> src <ul style="list-style-type: none"> beans <ul style="list-style-type: none"> SessionBean.java constants <ul style="list-style-type: none"> htmlConstants.java controllers <ul style="list-style-type: none"> CardTradingControllerServlet.java LoginControllerServlet.java LogOutControllerServlet.java RegistrationControllerServlet.java TransactionControllerServlet.java utilities <ul style="list-style-type: none"> CreateForm.java DataTools.java Redirects.java Libraries JavaScript Resources build WebContent <ul style="list-style-type: none"> dataFiles META-INF script <ul style="list-style-type: none"> cardTrader.js loginAjax.js redirect.js registration.js transactionDetails.js styles <ul style="list-style-type: none"> cardCollectionServlet.css cardTrader.css cardTraderTable.css main.css transactionDetails.css WEB-INF <ul style="list-style-type: none"> cardTrader.jsp login.jsp registration.jsp transactionDetails.jsp 	<pre> IfReferredBy(request, "registration.jsp") public static boolean IfReferredBy(HttpServletRequest request, String cameFrom) { if(request.getHeader("Referer")==null) { return false; } else { String Referer = (String)request.getHeader("Referer"); if(Referer.contains(cameFrom)) { return true; } else { return false; } } } public static booleanNullOrEmpty(String input) { if(input==null input.trim().equals("")) { return true; } else { return false; } } </pre>
---	--