Assignment 2

Servlet, MVC, Sessions,Cookies, EL, JSTL

Points: 100

Due 02/29/16 EOD

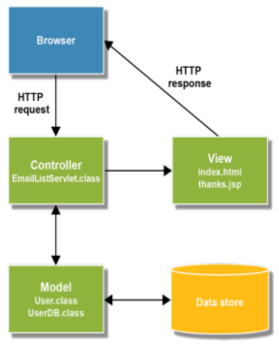
This assignment is to be done by a team of two students. Your first step of the assignment is to ensure that you have documented your team membership to course staff in the manner your instructor has specified. There is a fair amount of work to be done, but there are two people working the assignment together. You should establish a sensible way to share your project development work. You **MUST NOT** make your assignment code available publicly; as this would have to be treated as a violation of UNCC academic integrity policy. So do not use a repository service such as GitHub where all projects are publicly viewable – any such repository must be private.

This assignment is intended to familiarize students with Servlet web development including sessions, cookies, EL and JSTL. The assignment involves the use of servlets for the controller and JavaBeans for the business logic and sessions to maintain state. Please use the attached project as a starting point for your solution; you need to extend this project instead of your own project from assignment 1.

Assignment Description:

In this assignment you will develop Servlet pages and JavaBeans using the MVC pattern, according to the following specifications:

1. Continue working on attached project from the previous assignment.
2. Correct any errors identified in the selected HTML5/JSP prototype before proceeding.
3. All structure, design, and content requirements from the previous assignment are mandatory, unless explicitly updated in this assignment description.
4. Use JavaBeans to implement the business layer of the application (**model**).
5. Use JSP pages to present the **view** to the browser.
6. Use Servlet pages to **control** the flow of the application.



Create JavaBeans for Business Objects (Model)

This assignment will make use of JavaBeans for the Model, in order to represent the main data elements / business objects being used. Remember that a JavaBean is a Java class that (1) provides a zero-argument constructor; (2) provides get and set methods for all of its private instance variables that follow standard Java naming conventions; and (3) implements the Serializable interface. Create JavaBeans for the following Model elements, with the specified instance variables and additional methods. Bean class names MUST use the specified names.

User

- Name
- Email
- Type [Participant | Admin]
- NumCoins
- NumPostedStudies : how many studies have he/she created
- NumParticipation : how many studies have he/she participated in
  1 participation = 1 coin
  1 participant = 1 coin

Study

- StudyName
- StudyCode
- DateCreated
- Email (Creator)
- Question
- String getImageURL() – URL that can be used in your pages, pointing to an image file within the project for your question. Generated from study code.
- Requestedparticipants : How many participants does the creator want
- Numofparitipants : How participants have finished this study thus far.
- Description
- Status
- AnswerType [Text or Numeric]
- List / Collection (your choice) of answer.
- getAverage()
- getMinimum()
- getMaximum()
- getSD() – standard deviation

Answer

- Email : of the participant

- Choice
- SubmissionDate

- **Create Utility Classes for Persistent Data**

For initial development, you will be asked to use a hard-coded "database" to represent sets of users and studies. You must have at least one user in your UserDB and one study.

UserDB

- Hard-Coded set of user details (your choice on how to represent)
- User getUser(String email) – returns a user object based on the email.
- List/Collection<User> getUsers() – returns a set of all the users in the hardcoded "database"

StudyDB

- Hard-Coded set of study details (your choice on how to represent internally, but must be converted into a list/collection of Study beans when required)
- Study getStudy(String studyCode) – returns a Study object based on studyCode.
- List/Collection<Study> getStudies() – returns a set of all the studies in the hardcoded "database"
- List/Collection<Study> getStudies(String email) – returns a set of all the studies in the hardcoded "database" for the passed-in email.
- addStudy(Study study)
- updateStudy(String SCode, Study study)

Create Servlets for Business Logic (Controllers)

- Implement the following Controller servlets to operationalize the business logic. You will need to send parameters as part of the GET or POST http requests from link / button / form submissions as the context information that tells the controller how to proceed. The title [Researchers Exchange Participations] must be a link that points to the UserController servlet with action = "main".

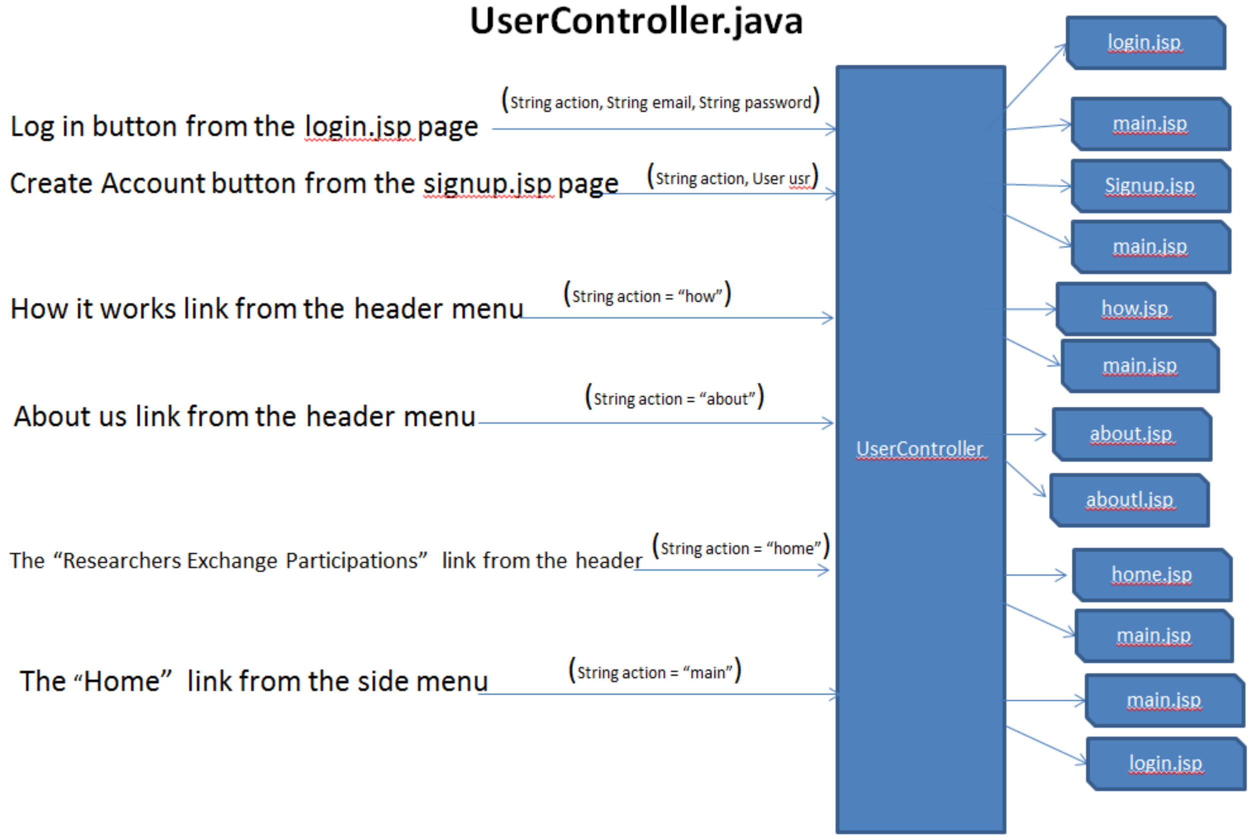
UserController.java

- Checks the http request for a parameter called "action"
  - If there is no action parameter, or if it has an unknown value, dispatches directly to the website's main page (home.jsp).

- If there is an action parameter, validates that its value is either “login”, “create”, “how”, “about”, “home” , “main” or “logout”
    - If action is “login”
        - Checks the http request for parameters called: email and password
        - Validates if the combination exists in the database
        - If they are correct,
            - Check the user type
            - If it is participant,
                - Creates a User bean for the user
                - Adds the User bean to the current session as “theUser”.
                - Dispatches to the main.jsp
            - If it is Admin
                - Creates a User bean for the user
                - Adds the User bean to the current session as “theAdmin”.
                - Dispatches to the admin.jsp
        - If they are not correct,
            - Adds an error message to the http request object , call the parameter “msg”.
            - Dispatches to the login.jsp page.
    - If action is “create” [creating a new account for a participant]
        - Checks the http request for parameters: name, email, type [by default is set to participant], password, and confirm password.
        - Validates the above information for possible errors.
        - If there is any error:
            - Adds an error message to the http request object, call the parameter “msg”.
            - Adds the above information to the http request.
            - Dispatches to the signup.jsp page.
        - If there is no errors
            - Creates a User bean for the user
            - Adds the User bean to the current session as “theUser”
            - dispatches to the main.jsp.
    - If action is “how”
        - Checks the session for “theUser” object,
            - If it exists
                - Dispatches to main.jsp
            - If it does not exist
                - Dispatches to how.jsp
    - If action is “about”
        - Checks the session for “theUser” object,

                - If it exists
                    - Dispatches to about.jsp
                - If it does not exist
                    - Dispatches to aboutl.jsp
    - If action is “home”
        - Checks the session for “theUser” object,
            - If it exists
                - Dispatches to main.jsp
            - If it does not exist
                - Dispatches to home.jsp
    - If action is “main”
        - Checks the session for “theUser” object,
            - If it exists
                - Dispatches to main.jsp
            - If it does not exist
                - Dispatches to login.jsp
    - If action is “logout” [Place the logout link next to the user name].
        - Checks the session for “theUser” or “theAdmin” objects,
            - If exists
                - Destroys the session
                - Dispatches to the home.jsp
            - If it does not exist
                - Dispatches to the home.jsp

UserController.java


Log in button from the login.jsp page
(String action, String email, String password)
Create Account button from the signup.jsp page
(String action, User usr)
How it works link from the header menu
(String action = "how")
About us link from the header menu
(String action = "about")
The "Researchers Exchange Participations" link from the header
(String action = "home")
The "Home" link from the side menu
(String action = "main")
UserController
login.jsp
main.jsp
Signup.jsp
main.jsp
how.jsp
main.jsp
about.jsp
aboutl.jsp
home.jsp
main.jsp
main.jsp
login.jsp

StudyController.java

- Checks the http request for a parameter called "action"
    - If there is no action parameter, or if it has an unknown value
        - Checks the session for "theUser" object,
            - If it exists
                - Dispatches to the website's main page (main.jsp)
            - If it does not exist
                - Checks the session for the "the Admin" object
                    - If it exists
                        - Dispatches to the admin.jsp page
                    - If it does not exist
                        - Dispatches to the website's main page (home.jsp)
    - If action is "participate"
        - Checks the session for "theUser" object,
            - If it exists

      - Checks the http request object for parameter called “StudyCode”
        - If does not exist
          - Retrieves the open studies (status = start) from the DB.
          - Puts them in the http request object.
          - Dispatches to the pariticipate.jsp.
        - If it exists
          - Retrieves **the study record** (question) from the DB.
          - Puts **it** in the http request object.
          - Dispatches to the question.jsp.
    - If it does not exist
      - Dispatches to the login page (login.jsp)

- If action is “edit”
  - Checks the session for “theUser” object,
    - If it exists
      - Checks the http request object for a parameter called “StudyCode”.
      - Pulls **a study record** from the DB based on the study code and creator’s email address.
      - Adds **it** to the http request object.
      - Dispatches to the editstudy.jsp.
    - If it does not exist
      - Dispatches to the login page (login.jsp)
- If action is “report”
  - Checks the session for “theUser” object,
    - If it exists
      - Checks the http request object for parameter called “StudyCode”
        - If does not exist
          - Retrieves all requests that were submitted by the current user from the DB.
          - Adds them to the http request object
          - Dispatches to the reporth.jsp.
        - If it exists
          - Checks the http request object for parameter called “ReporterEmail”
          - Add a new report record into the DB.
          - Dispatches to the confirmrep.jsp

      - If it does not exist
        - Dispatches to the login page (login.jsp)
- If action is “approve”
  - Checks the session for the “theAdmin” object,
    - If it exists
      - Checks the http request object for parameter called “StudyCode”
      - Sets the request [for that study code] status to approve.
      - Retrieves all requests from the DB.
      - Puts **them** in the http request object.
      - Dispatches to the reportques.jsp.
    - If it does not exist
      - Dispatches to the login page (login.jsp)
- If action is “disapprove”
  - Checks the session for the “theAdmin” object,
    - If it exists
      - Checks the http request object for parameter called “StudyCode”
      - Sets the request [for that study code] status to disapprove.
      - Retrieves all requests from the DB.
      - Puts **them** in the http request object.
      - Dispatches to the reportques.jsp.
    - If it does not exist
      - Dispatches to the login page (login.jsp)
- If action is “update”
  - Checks the session for “theUser” object,
    - If it exists
      - Retrieves the study details from the http request
      - Updates the study record in the DB.
      - Retrieves the **user’s studies** from the DB.
      - Puts **them** in the http request object.
      - Dispatches to the studies.jsp
    - If it does not exist
      - Dispatches to the login page (login.jsp)
- If action is “add”
  - Checks the session for “theUser” object,
    - If it exists
      - Retrieves the study details from the http request
      - Adds the **study record** to the DB.
      - Retrieves the **user’s studies** from the DB.
      - Puts **them** in the http request object.

            - Dispatches to the studies.jsp
- If action is “start”
    - Checks the session for “theUser” object.
        - If it exists
            - Checks the http request for parameter called “StudyCode”.
            - Sets the study status to start.
            - Retrieves the **user’s studies** from the DB.
            - Puts **them** in the http request object.
            - Dispatches to the studies.jsp.
        - If it does not exist
            - Dispatches to the login page (login.jsp)

- If action is “stop”
    - Checks the session for “theUser” object,
        - If it exists
            - Checks the http request for parameter called “StudyCode”.
            - Sets the study status to stop.
            - Retrieves the **user’s studies** from the DB.
            - Puts **them** in the http request object.
            - Dispatches to the studies.jsp
        - If it does not exist
            - Dispatches to the login page (login.jsp)

- If action is “answer”
    - Checks the session for “theUser” object,
        - If it exists
            - Checks the http request for parameter called “StudyCode” and “choice”.
            - Adds the answer to the DB.
            - Update the user data (coins and participation in the DB and the session).
            - Retrieves the **open studies** (status = start) from the DB.
            - Puts **them** in the http request object.
            - Dispatches to the pariticipate.jsp.
        - If it does not exist
            - Dispatches to the login page (login.jsp)
- If action is “studies” [in case the user clicks the My Studies link in the side menu]
    - Checks the session for “theUser” object,
        - If it exists
            - Retrieves all studies created by the user—based on the email address from the theUser object—from the database.

        - Puts them in http request object.
        - Dispatches to the studies.jps
    - If it does not exist
        - Dispatches to the login page (login.jsp)

Update JSP Views to Include Dynamic Content From Bean Data

Update the JSP views to replace all of the static placeholder information from the old html files with the dynamic data using JSP functionality to access bean data. (e.g. User Name, Coins, Participants, Participation).

Update JSP Links

Update the JSP links to point into the right servlet and append the needed parameters.

Cookie

Create a cookie that stores the host and port of the first machine to request the home page (home.jsp). Then, display these information in the footer area.

HTML5 Validation/Cross-browser testing

Validate your HTML5 on http://html5.validator.nu/. Also, make sure your app works (and looks right) in both Google chrome and Firefox.

Assignment Submissions

What to submit using Moodle (Email submissions will NOT be accepted):
1**. firstname_assignment2.war** - An archive of the entire web application (project) stored in a standard WAR file. You must ensure that the java source files are included as part of the archive. The WAR file will be imported into Netbeans for grading and may be required to be deployed as part of the submission.

2. **info.pdf** – PDF document with the following assignment information :
a) Explanation of status and stopping point, if incomplete.
b) Explanation of additional features, if any.
c) Discuss the easy and challenging parts of the assignment. How did you overcome all or some of the challenges?
d) Discuss division of labor specifying who did what and why this is a fair and equal split.

3. **Openshift link**. Upload your project into openshift and post the openshift link in the online text session (so we have easy access when grading).

Both team members must submit the assignment (same copy).

Finally, set up a time to demo your assignment to your grader using the process described in the course Moodle site. Students should be prepared to answer questions posed by the grader about their work. Failing to demonstrate the assignment to the grader will result in no credit for all team members.