

## Module 3 Assignment Instructions

### Part 1 - The Java Servlets Assignments

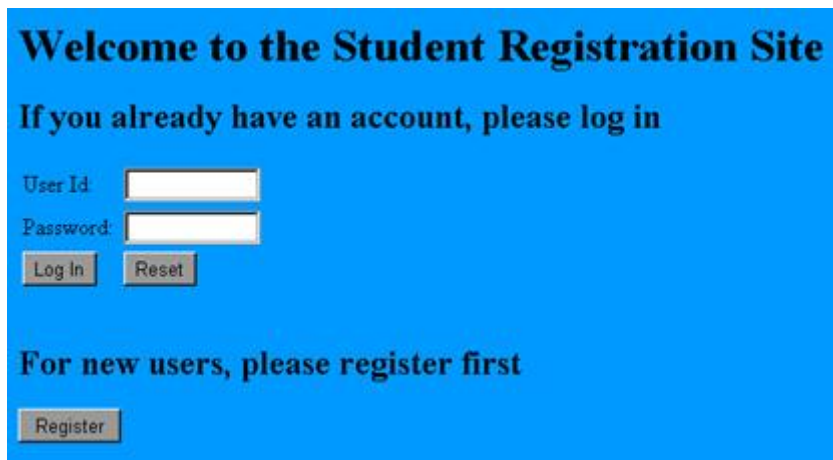
In this assignment students have to design and implement two use cases.

**#Use Case 1** - The "Login" use case which is based on the HTML document, "login.html" for a student to login to the site and supporting servlet, the Login servlet, to process a business logic.

The "Login" use case - to login as previously registered student by providing USERID/PASSWORD and to get information about the courses being offered and available during upcoming semester. If USERID/PASSWORD is invalid (both userid and password have to be 8 characters each and be non-blank), then the system returns an appropriate error message and suggests a user trying to log in again.

**Note:** After N unsuccessful attempts, the system terminates client's session with an appropriate error message.

The following is a screen shot of the Login page:



The screenshot shows a blue background with white text. At the top, it says "Welcome to the Student Registration Site". Below that, it says "If you already have an account, please log in". There are two input fields: "User Id" and "Password". Below the "User Id" field is a "Log In" button, and below the "Password" field is a "Reset" button. At the bottom, it says "For new users, please register first" and there is a "Register" button.

#### Requirements:

- All business logic has to be processed by servlets or JSPs. **No JavaScript code is permitted!!!** This requirement is applicable to all assignments throughout this course.
- All users' requests have to be received only by one servlet, the **RegistrationController** servlet. Then it passes it to the next servlet based on a type of the request. The collaboration between servlets should be implemented using the RequestDispatcher mechanism, either "include" or "forward" as it is appropriate.

The "Login" use case control flow looks like this:

**Login page -> request -> "RegistrationController\_servlet" -> "Login\_servlet" -> database -> response (Welcome page/Error page)**

- During login process, the syntax validation of the 'UserId' and 'Password' parameters has to occur as follows:
  - the length for both should be 8 characters.

- neither one can contain "space".
- Should any syntax error occur, the Login servlet sends a response to a user with an error message and lets the user try logging again.
- Once the number of login attempts exceeds predefined number, user session has to be disconnected with appropriate message.

**Important note:** A predefined number of login attempts must be an initial parameter of the Login servlet provided by the web container as web application configuration parameter (using web.xml file).

- Then, the Login servlet accesses the JHU database (remember, you created it in JDBC assignment a few weeks ago) to validate an existence of the student account.

**Important note:** The SRS system will use DataSource-based method of accessing the JHU database. So, the URL for connecting to WebLogic Server, as well as the DataSource name, have to be provided to the Login\_servlet as session parameters established by the RegistrationController servlet, not be hardcoded. The RegistrationController servlet has to setup those based on initial parameters for this servlet provided by web container.

- If an account does exist (both UserId and Password are valid), the Login servlet sends the welcome HTML page with a message:

**Welcome to the site, Joe Doe**

where *first\_name* and *last\_name* have to be received from the STUDENT database for a given userId.

- If an account doesn't exist, Login Servlet sends a message back:

**Sorry, you don't have an account. You must register first.**

**#Use Case 2** - The "Registration" case which is based on two HTML documents, "Registration Form A" and "Registration Form B", and supporting servlet, the Registration servlet, to process a business logic.

This includes processing and collecting personal information and storing it in the database for future references. During the registration each student will input a userid/password which he/she will use to login to the system later on. The registration information takes two forms, A and B, to be filled in. The following are screen shots of those registration forms A and B:

## Registration Form A

User Id:

Password:

Password (repeat):

First name:

Last name:

Social Security Number:  -  -

Email:

## Registration Form B

Address:

City, State:  --

ZIP/ Postal Code:

"Registration" use case control flow:

**Form A** → request → "RegistrationController\_servlet" → "Registration\_servlet" → response with Form B

**Form B** → request → "RegistrationController\_servlet" → "Registration\_servlet" → database → Welcome page/Error page

- During the registration process the Registration\_servlet has to accumulate information from both Form A and Form B (using session tracking mechanism) and then to store this information in the database.

**Important note:** The URL to connect to WLS and the DataSource name both have to be provided to the Registration\_servlet as session parameters established by RegistrationController servlet.

- To keep track of user session you have to implement a session tracking mechanism.
- As it was already described, you must have three initial parameters defined for your application servlets in web.xml file:
  - the URL to connect to WebLogic Server;
  - the DataSource name;
  - number of attempts allowed to login before terminating a session.

## Deployment

By this time you have learned from the textbook and other sources that, according to the Servlets specification and to the WebLogic Server implementation, the deployment process can be done in two ways: either as one war file or as a set of separate web application components.

For now, we will deploy this portion of application not as one war file, but as a set of individual components. To do that you have to create an **SRS** (Student Registration System) directory somewhere on your local drive. Then you have to define a web.xml file with **all** components (servlets, their initial parameters according to the requirements, database connections, etc.). After that, start WLS, connect to it with the Admin console, and deploy your SRS application through the console.

Execute your SRS application, including all business functions (registration process and login process). Collect screen shots for submission; you will submit them along with those completed in Part 2 of this assignment.

## Part 2 - The JavaServer Pages Assignment

We will continue working on the "Student Registration System" project. In this assignment we will add the "Course" use case to it. The "Course" use case is about registering to a desired course based on the following requirements.

### Requirements

You are going to use the RegistrationController servlet developed in the Part 1 of this week's assignment as one-point-of-entry for the SRS web application. All user requests have to be received only by this Java Servlet and then are passed to the next appropriate processing application component/resource.

- As you remember from Part 1 of this assignment, after a successful registration or login process, a user will get a response as a message:

**Welcome to the site, Joe Doe**

with *first\_name* and *last\_name* filled in from the STUDENT table in the JHU database.

- In this assignment, change this HTML page to look like:

**Welcome to the site, Joe Doe**  
**Select your next action:**

and then add radio box options:

- **Register for the course**
- **Logout**

and then "**Submit**" button.

- As the user selects an option "Register for the course", the request gets passed from RegistrationController servlet to the JSP called **CoursesJSP.jsp** to retrieve a list of known courses from the database and then to dynamically generate the response in the form of a drop-down menu with all the courses. Each item in the menu is a string concatenation of *course\_id* and *course\_name*.
- In order to feed the CoursesJSP.jsp, you have to create two new tables, COURSES and REGISTRAR, in the JHU database.

First table, COURSES, is going to have two columns, '**courseid**' as integer and '**course\_name**' as varchar. To populate this table, you have to develop a Java program and provide an input file for it.

Second table, REGISTRAR, includes two columns, '**courseid**' as integer, and '**number\_students\_registered**' as integer. This table will be maintained based on online processing. Every request to register for a given course will increase the number of students registered by 1.

- As a next step in a process, user selects any course from the drop-down list of courses provided by CourseJSP page. The request with selected course has to be processed by **RegistrarCourse.jsp** JSP page to either register a user for the course, if this course is still available, or to deny registration, if this course is not available. The availability of the course has to be established based on comparison of current number of students registered with a JSP configuration parameter, "CourseCapacity". This parameter has to be declarative and not hardcoded within the JSP code. This way we can change it any time without changing the code.
- You have to develop and use two Java Beans, **CoursesSupportBean** and **RegistrationSupportBean**, in order to support business logic flow and execution CoursesJSP and RegistrarCourse pages respectively.
- In both cases, successful and unsuccessful registration processing, the respective JSP page has to generate a dynamic content response with an appropriate message, either

**"You have been registered to *course\_id* *course\_name*"**

or

**"Sorry, the registration to this course has been closed based on availability"**

Execute your SRS application and collect screen shots for submission; you will submit them along with those completed in Part 1 of this assignment.