

Spring Web MVC Assignments

- 1) Design and develop a Spring MVC web application as follows:
 - a. Create index.jsp page having one hyperlink. When user clicks on that hyperlink, it should call HelloWorldController.
 - b. Design HelloWorldController class that returns a view helloWorld.jsp
 - c. Design a helloWorld.jsp page that displays "Hello World" message.
- 2) Design and develop a Spring MVC web application as follows:
 - a. Design simpleInterest.jsp to capture principal amount, no. of years and rate of interest
 - b. SimpleInterestController will receive these inputs and calculates simple interest. SimpleInterestController should return a view that contains simple interest.
- 3) Design and develop a Spring MVC web application as follows:
 - a. Design a User model class having attributes username, password & email.
 - b. Design login.jsp to capture username and password (Use ModelAttribute)
 - c. UserController should receive the credentials and return "success" view if user is authenticated successfully, else return "error" view.
 - d. Design success.jsp and error.jsp
- 4) Design and develop a Spring MVC web application as follows:
 - a. Design a User model class having attributes username, password & email.
 - b. Design registration.jsp to capture username, password and email. (Use ModelAttribute)
 - c. UserController should receive the details and store into a database. (Use JdbcTemplate). After successful registration, it should return login view.
 - d. Design login.jsp to capture username and password (Use ModelAttribute)
 - e. UserController should receive the credentials and return "success" view if user is authenticated successfully, else return "error" view.
 - f. Design success.jsp and error.jsp
- 5) Develop an "Employee Management System" that manages the information about employees
 1. Add a new employee
 2. Searching for specific employee
 3. Deleting an existing employee
 4. Finding all employees
 5. Editing/updating employee information.
 - a. Create an Employee domain model class having following properties: employeeId, employeeName, employeeDepartment, employeeDesignation, employeeSalary. Employee Id should be generated automatically at database level.

Refer following UI screens

| | |
|------------------------------------|----------------------|
| Employee Name | <input type="text"/> |
| Employee Department | <input type="text"/> |
| Employee Designation | <input type="text"/> |
| Employee Salary | <input type="text"/> |
| <input type="button" value="Add"/> | |

| Employee Id | Employee Name | Employee Department | Employee Designation | Employee salary | Update | Delete |
|-------------|---------------|---------------------|----------------------|-----------------|----------------------|------------------------|
| 101 | John Doe | HR | HR | 23000 | Edit | Delete |
| 102 | Jane Doe | Sales | Sales Manager | 56000 | Edit | Delete |

- b. If user clicks on delete option, record should be deleted from the database.
- c. If user clicks on edit option, a form to be displayed on user screen where user can update Employee information.

| | |
|---------------------------------------|--|
| Employee Id | <input type="text" value="102"/> |
| Employee Name | <input type="text" value="Jane Doe"/> |
| Employee Department | <input type="text" value="Sales"/> |
| Employee Designation | <input type="text" value="Sales Manager"/> |
| Employee Salary | <input type="text" value="56000"/> |
| <input type="button" value="Update"/> | |

Note: Employee Id should be disabled and rest of the fields are editable.

- d. Use Spring JdbcTemplate to develop repository layer logic.
- e. Design appropriate controller, service and repository layer logics.

6) Example on Validation-API

Develop a spring mvc application based on following guidelines.

- Design a class Customer with following attributes: username, password, email, contact, city and zip code.
- Design a registration.jsp page by using spring form tags.

| | |
|---|--|
| Username | <input type="text"/> |
| Password | <input type="password"/> |
| Email | <input type="text"/> |
| Contact | <input type="text"/> |
| City | <input type="text" value="Pune"/> <input type="button" value="v"/> |
| ZipCode | <input type="text"/> |
| <input type="button" value="Register"/> | |

Apply the following server-side validations to the above registration form:

- Username should not be empty or null. Username should be alphanumeric and between 8 to 20 characters. Username should not contain space.
- Password should not be empty or null. Password should contain at least one upper-case letter, lower-case letter, a digit or special character (_ , \$, # , . , @). Password should also be 8 to 20 characters long.
- Email should not be empty or null. Email should be valid.
- City should be selected.
- ZipCode should not be empty or null. It should be 6-digits.

If user does not enter valid inputs, error messages should be shown on the registration form after the input field.

7) Example on Custom validation.

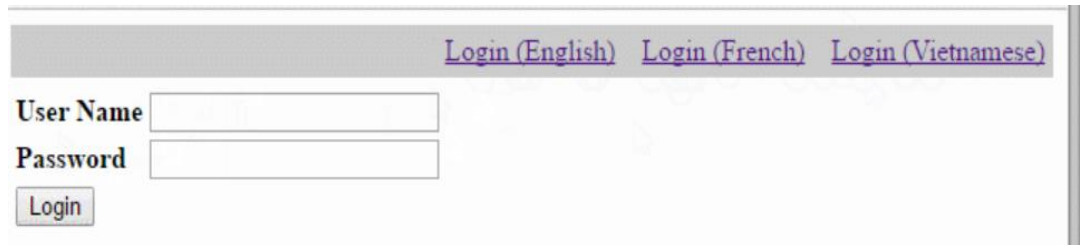
Add custom validations in the above example.

- Contact should be only numeric and exactly 10-digits.
- Once Customer selects city and ZipCode. Validate the zipcode i.e ZipCode belongs to the same city or not. (Use small set of sample data)

8) Example on i18n

Design and develop a Spring MVC web application as follows:

- a. Create a login.jsp page

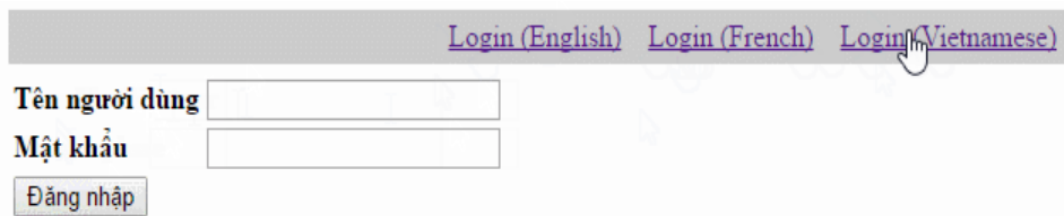


The screenshot shows the English version of the login page. At the top, there are three links: [Login \(English\)](#), [Login \(French\)](#), and [Login \(Vietnamese\)](#). Below these links, there are two input fields: "User Name" and "Password". A "Login" button is positioned below the "Password" field.

- b. When user clicks on the French or Vietnam, page text should be changed to selected Locale.



The screenshot shows the French version of the login page. At the top, there are three links: [Login \(English\)](#), [Login \(French\)](#), and [Login \(Vietnamese\)](#). A mouse cursor is pointing at the "Login (French)" link. Below these links, there are two input fields: "Nom d'utilisateur" and "Mot de passe". A "Connexion" button is positioned below the "Mot de passe" field.



The screenshot shows the Vietnamese version of the login page. At the top, there are three links: [Login \(English\)](#), [Login \(French\)](#), and [Login \(Vietnamese\)](#). A mouse cursor is pointing at the "Login (Vietnamese)" link. Below these links, there are two input fields: "Tên người dùng" and "Mật khẩu". A "Đăng nhập" button is positioned below the "Mật khẩu" field.

9) Design and develop a Spring MVC web application as follows:

- a. Create index.jsp page which contains one hyperlink as shown below



The screenshot shows the index.jsp page. It contains a single hyperlink: [View Employees](#).

- b. Develop EmployeeController class that returns showAllEmployees view which displays only first 5 records and links to other pages that show further employee details.
- c. Create showAllEmployees.jsp
- d. Develop controller, service, repository layers and domain model classes.

Employees List

| Id | Name | Salary |
|----|---------|---------|
| 1 | Amit | 30000.0 |
| 2 | Ajeet | 40000.0 |
| 3 | James | 50000.0 |
| 4 | Sonoo | 60000.0 |
| 5 | Sarfraz | 70000.0 |

[1](#) [2](#) [3](#)

Employees List

| Id | Name | Salary |
|----|--------|---------|
| 6 | Bob | 80000.0 |
| 7 | Rahul | 90000.0 |
| 8 | Rakesh | 25000.0 |
| 9 | Udit | 35000.0 |
| 10 | Jai | 45000.0 |

[1](#) [2](#) [3](#)

Employees List

| Id | Name | Salary |
|----|--------|---------|
| 11 | Nikhil | 55000.0 |
| 12 | Somesh | 65000.0 |
| 13 | Rajesh | 75000.0 |
| 14 | Ankit | 85000.0 |
| 15 | Ratan | 95000.0 |

[1](#) [2](#) [3](#)

10) Design and develop a Spring MVC web application as follows:

- a. Create index.jsp page which contains one hyperlink as shown below



- b. Develop controller, service, repository layers and domain model classes.
- c. EmployeeController class should return all the employee details in excel sheet.

11) Modify the above example to generate employee details in pdf format.

12) Exception handling concept (@ControllerAdvice, @ExceptionHandler etc. will be covered in our regular Banking Application case study.

Exceptions:

- a. LowBalanceException: when customer is not having sufficient fund for transaction
- b. AccountNotFoundException: when payees account does not exists. (fund transfer)