BITP 3123 Distributed Application Development

Lab 11
Programming with Java Server Pages

Learning Outcomes

At the end of this lab, the student should be able to:-

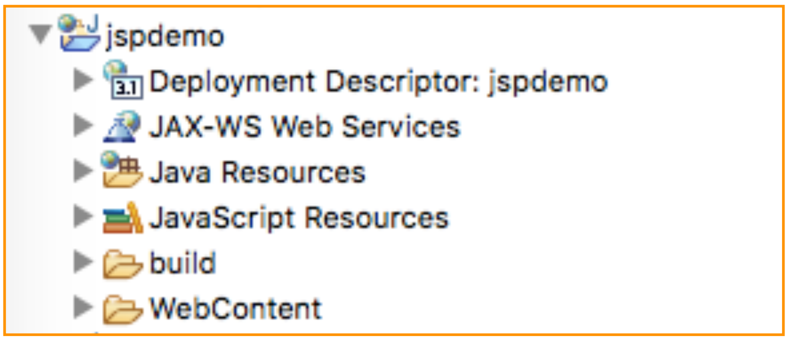
1. Amend simple JSP programs.
2. Create simple JSP programs.
3. Integrate JSP and existing Java class for a simple dynamic web based application.

Table of Content

Exercise 1: Setting-Up Lab 11

Purpose:To create a new project for a dynamic web application that will be using JSP and servlet.

1. Create a new dynamic web project named jspdemo.
2. Expand the project. The structure should be similar as shown Figure 1.



```
▼ jspdemo
    ▶ Deployment Descriptor: jspdemo
    ▶ JAX-WS Web Services
    ▶ Java Resources
    ▶ JavaScript Resources
    ▶ build
    ▶ WebContent
```

Figure 1: Structure of jspdemo

3. Create a new folder named `jsp` in **WebContent**.

Exercise 2: Importing File to Project

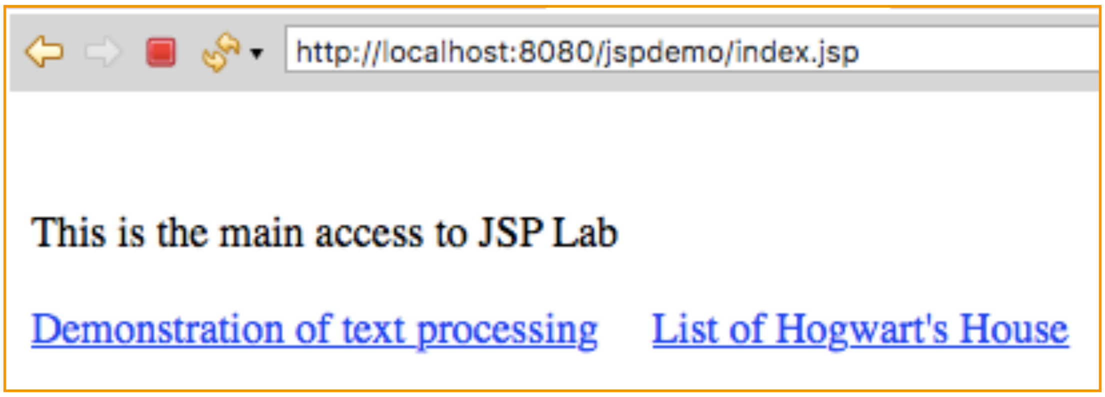
Purpose: To include some of the pre-defined files into the project.

1. Download and extract `Codes for lab11.zip`.
2. Put `index.jsp` in **WebContent** and the other `.jsp` files on `jsp` folder.
3. Put all the `.java` files into an appropriate folder in **Java Resources/src**.

Exercise 3: Testing the Lab 11 Setup

Purpose: To test the setup for implementation of Lab 11.

1. Add **jspdemo** into the current Tomcat server.
2. Start the server.
3. Right click on **jspdemo**. Select **Run As > Run On Server**.
4. The browser should display an output as shown in Figure 2 below.



http://localhost:8080/jspdemo/index.jsp

This is the main access to JSP Lab

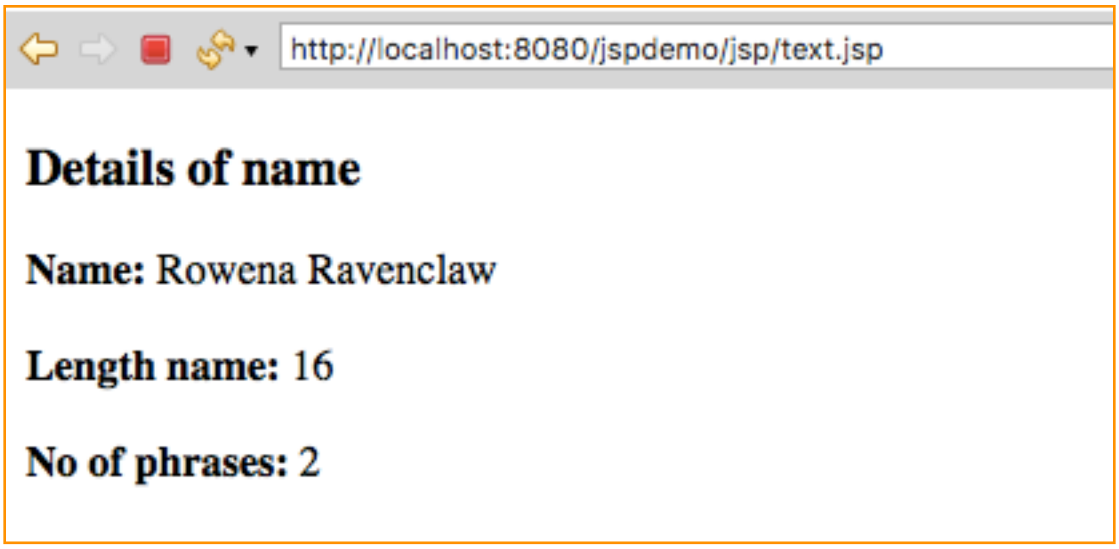
Demonstration of text processing   List of Hogwart's House

Figure 2: Main access to the application

Exercise 4: Amend JSP Code

Purpose: To provide Java codes in a JSP file.

1. Expand **WebContent/jsp**.
2. Double click `text.jsp` to open the file.
3. This file is an unfinished code. Complete the code and make amendments where is is necessary. Use the comments to guide your implementation.
4. Save the code.
5. Run the application.
6. Click the link labelled as **Demonstration of Text Processing.** You should get an output as shown in Figure 3.



```
Details of name

Name: Rowena Ravenclaw

Length name: 16

No of phrases: 2
```

Figure 3: Expected output from `text.jsp`

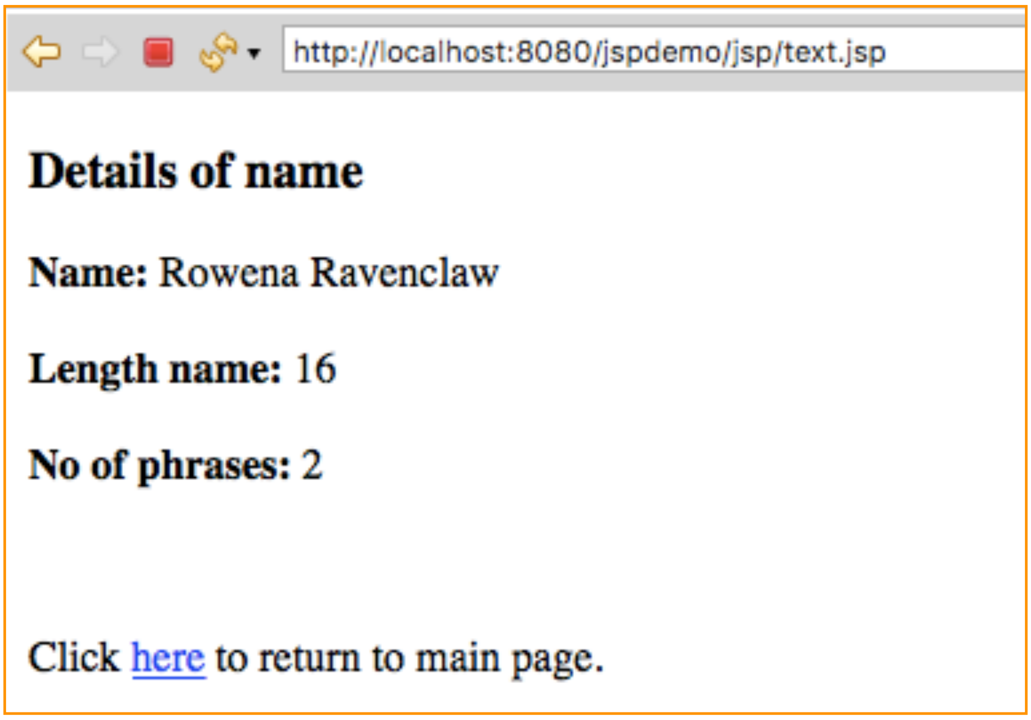
Exercise 5: Including a Footer Menu

Purpose:To use `<jsp:include>` to include a HTML file.

1. Open `text.jsp`.

```
<!-- Include a footer menu -->
<jsp:include page="footerMenu.html" />
```

2. Add the following code the `</body>`.
3. Save the file.
4. Run the application.
5. Click the link labelled as **Demonstration of Text Processing**. You should get an output as shown in Figure 4 below


http://localhost:8080/jspdemo/jsp/text.jsp

Details of name

Name: Rowena Ravenclaw

Length name: 16

No of phrases: 2

Click here to return to main page.

Figure 4:An expected output from `text.jsp` after including `footerMenu.html`

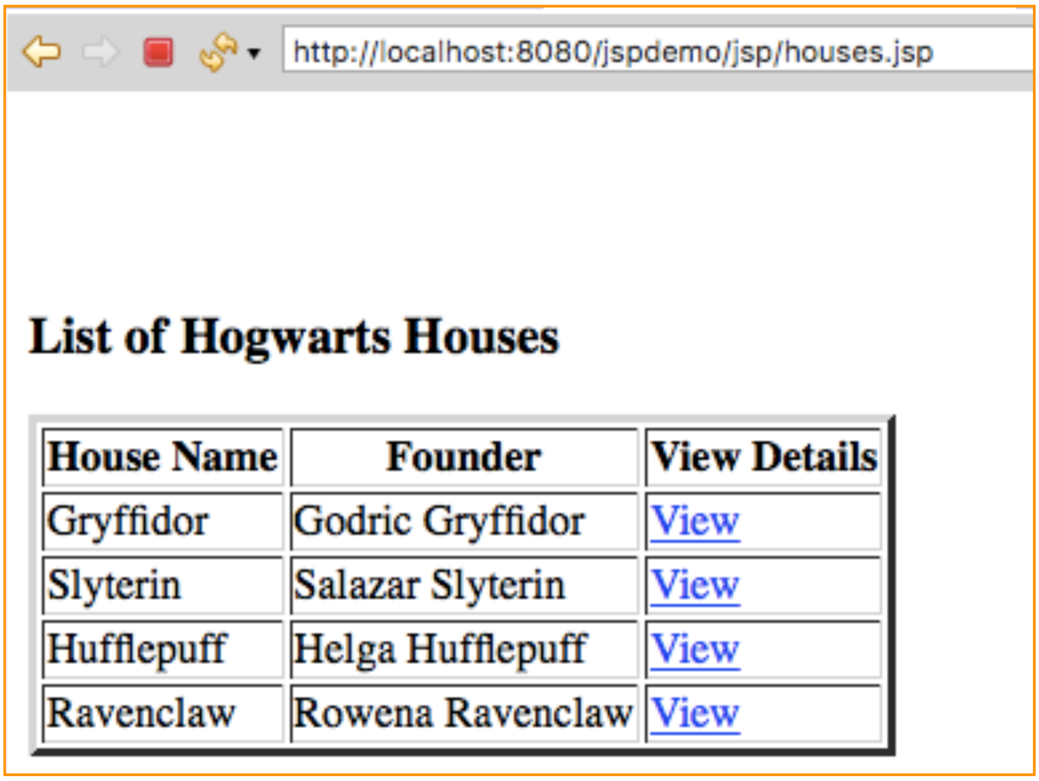
6. Click the link label as **here**. It shall take the browser to the main page as shown in Figure 2 .

Exercise 6: Amend more JSP Codes

Purpose:To integrate JSP with Pre-Defined Java Classes

Display List of Record using Facade Class

1. Open `houses.jsp`. This page is supposed to display a list of houses in Hogwarts.
2. There are three classes that has been created to manage the data related to Hogwarts's houses. These classes were imported in Exercise 2. The classes shall located in package `demo.hogwarts`.
3. Any interaction to manipulate the data either from JSP, servlet or other classes that are not within the same package must be from `HouseFacade.java`.
4. `houses.jsp` is an finished code. Complete the code to display a list of houses in Hogwarts.
5. You are allow to make amendments where is it deem necessary in the JSP. Use the comments to guide your implementation.
6. Import where necessary.
7. Save the code.
8. Run the application.
9. Click the link labelled as List of Hogwarts's Houses. You should get an output as shown in Figure 5 below.


http://localhost:8080/jspdemo/jsp/houses.jsp

List of Hogwarts Houses

House Name	Founder	View Details
Gryffidor	Godric Gryffidor	View
Slyterin	Salazar Slyterin	View
Hufflepuff	Helga Hufflepuff	View
Ravenclaw	Rowena Ravenclaw	View

Figure 5: Expected output from `houses.jsp`

Include Menu to Return to index.jsp

1. Include footerMenu.html. Use `<jsp:include>` as practiced in the previous exercise.
2. Save the code.
3. Run the application again. Similar menu as shown in Figure 4 should appear at the end of the page.
4. Click at the link labelled as here. The browser should return to the main page.

Display Total Number of Records

1. Add a total number of record after the table that displays a list of houses in Hogwarts.
2. Save the code.
3. Run the application again.
4. The output in step 1 shall be correctly displayed.

Exercise 7: Displaying a Selected Record

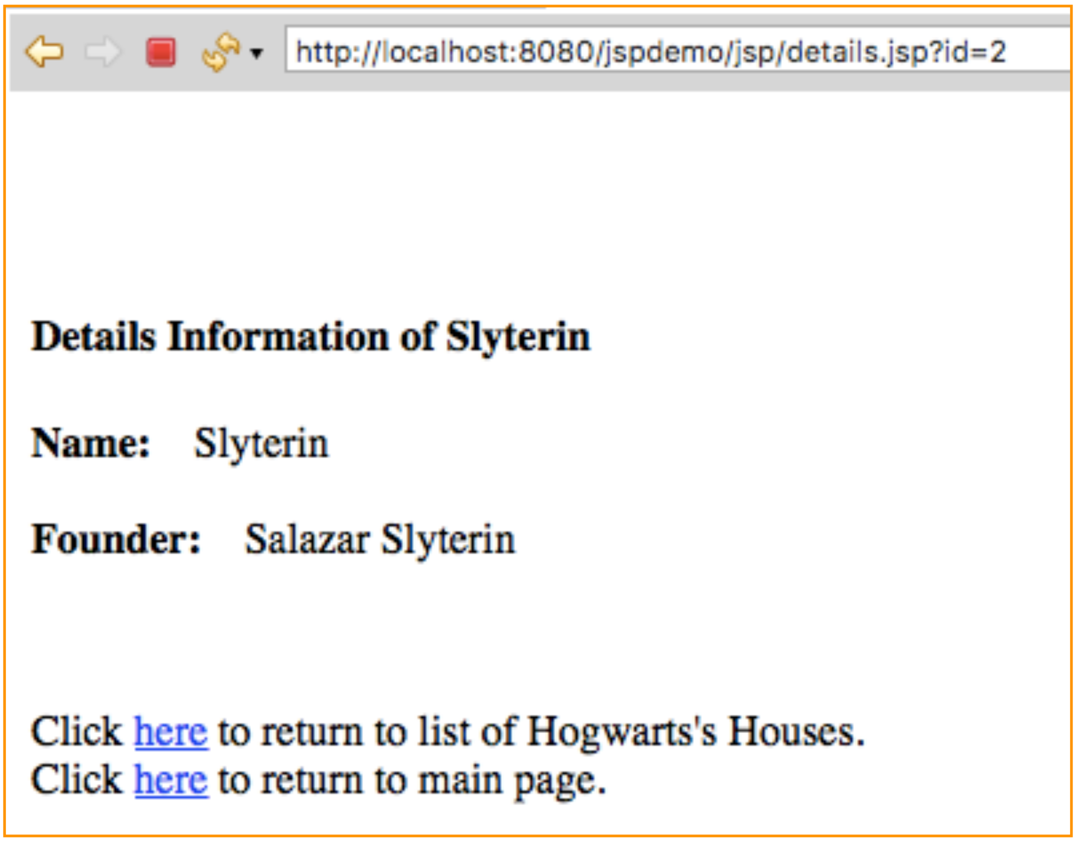
Purpose:To integrate Facade class with the new JSP to display the selected record.

The link labelled as View in the page that displays a list of Hogwarts's Houses shall display the selected record. The link will pass the house id to a page named details.jsp. This page shall display the complete information of the selected house.

1. Create a new JSP file in `jsp` folder. Name the file as `details.jsp`.
2. Implement the stated requirement for `details.jsp`.
3. Use the following pseudo-code to guide your implementation. You may refer to the previous code as well.

```
<%-- Import all the necessary classes --%>

<%

     // Get parameter value from link
     // Use request.getParameter( ).
     // It is similar practice from the Servlet.

     // Wrap into House

     // Get house. Use HouseFacade


%>
```

4. Display the details information in the HTML body as shown in Figure 6.


http://localhost:8080/jspdemo/jsp/details.jsp?id=2

Details Information of Slyterin

Name: Slyterin

Founder: Salazar Slyterin

Click here to return to list of Hogwarts's Houses.
Click here to return to main page.

Figure 6: Expected output from `details.jsp`

5. Save `details.jsp`.
6. Run the application.
7. Click the link labelled as **View** in the second row. It shall display the output as shown in Figure 6.
8. Return to the list of Hogwarts's Houses. Click other **View** from other row to test the application.

Exercise 8: Applying Changes to Hogwarts's Houses

Purpose: To apply changes to the model, business and presentation layer.

Amend The Model Layer

1. Add `color` and `mascot` in `House.java`. These two are `String` type of data.
2. Add `getters` and `setters` for the new attributes.
3. Save the classes.

Amend The Business Layer

1. Open `HouseDataManager.java`.
2. Add the following declaration in `loadHouses( )`.

```
String colors[] = {"scarlet-gold", "green-silver", "yellow-black",
"blue-bronze"};
String mascots[] = {"lion", "serpent", "badger", "eagle"};
```

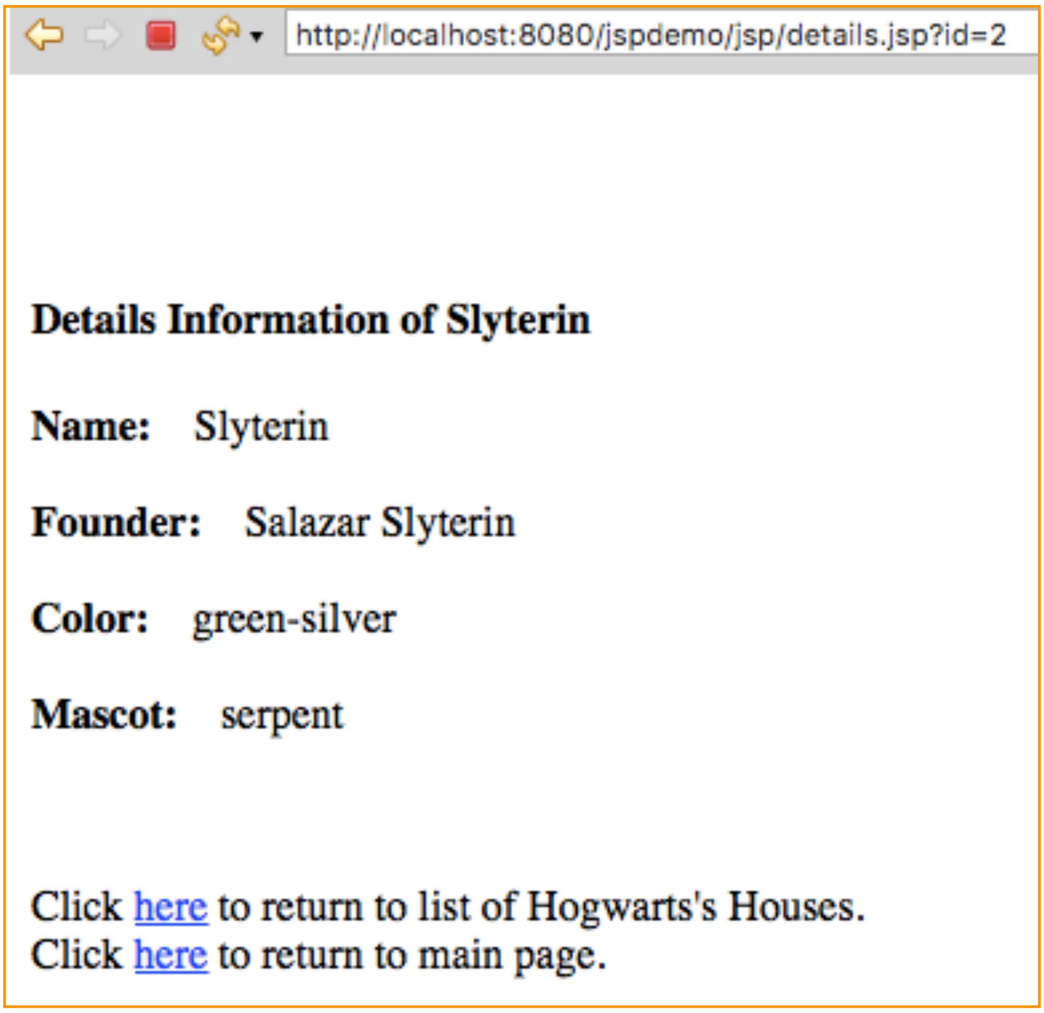
3. Amend the code in the loop. Assign the additional data to the new members of House.
4. Save the class.

Amend The Presentation Layer

1. Open `details.jsp`.
2. Add more codes to display color and mascot.
3. Save the file.

Test The Amendments

1. Run the application.
2. Click the link labelled as **View** in in the second row.
3. The browser shall display as output as shown in Figure 7 below.



http://localhost:8080/jspdemo/jsp/details.jsp?id=2

Details Information of Slyterin

Name: Slyterin

Founder: Salazar Slyterin

Color: green-silver

Mascot: serpent

Click here to return to list of Hogwarts's Houses.
Click here to return to main page.

Figure 7: Expected output after adding applying changes to the houses data

This page is intentionally left blank