BITP 3123 Distributed Application Development

Solution for Programming with Java Server Pages (JSP)

Author: Emaliana Kasmuri, FTMK, UTeM

Table of Contents

Solution for Programming with Java Server Pages

Purpose of Document

This document provides solution for the stated exercises in Lab 11 JSP.pdf.

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.

Tools

1. Eclipse Eclipse IDE for Enterprise Java Version: 2020-12
2. Tomcat 8.5 and above
3. Web browser preferably Chrome

Supplementary Files

1. Codes for lab11.zip from ulearn.

Solution Files

The solutions are available at Github, https://github.com/emalianakasmuri/jspdemo

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.

Figure 1: Structure of jspdemo

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

Outcome of Exercise 1

The structure of the project should be similar as shown in Figure 2

Figure 2: Project structure after adding 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**.

Outcome of Exercise 2

The structure of the project should be similar as shown in Figure 3.

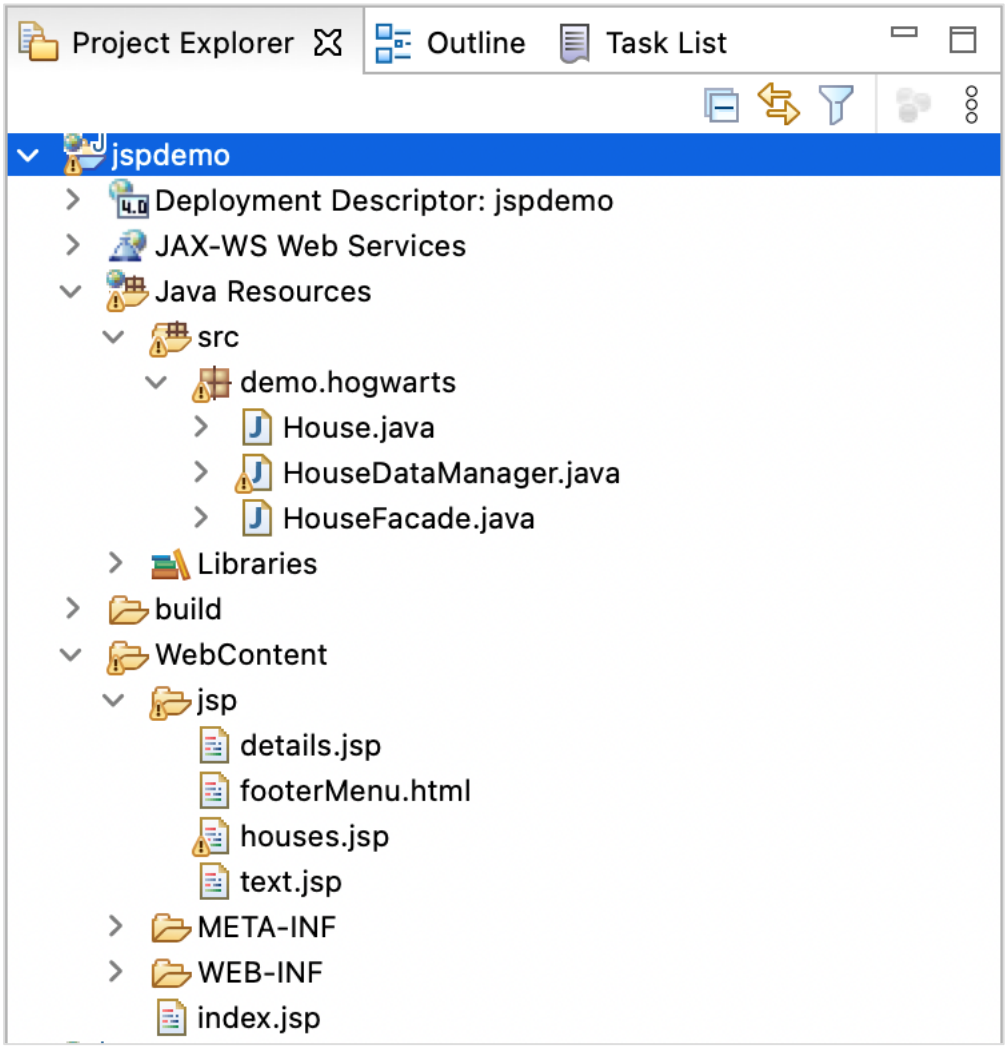

Figure 3: Structure of the project after importing files from Codes for Lab 11.zip

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 4.

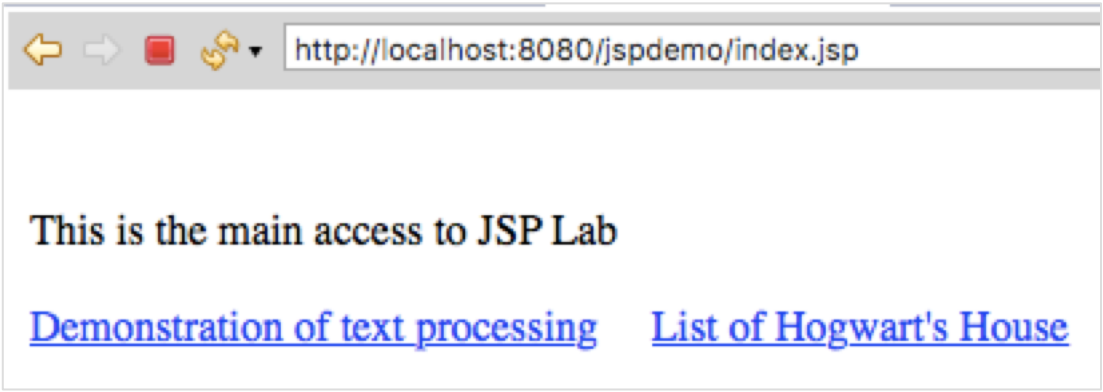

Figure 4: Main access to the application

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 it 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 5.

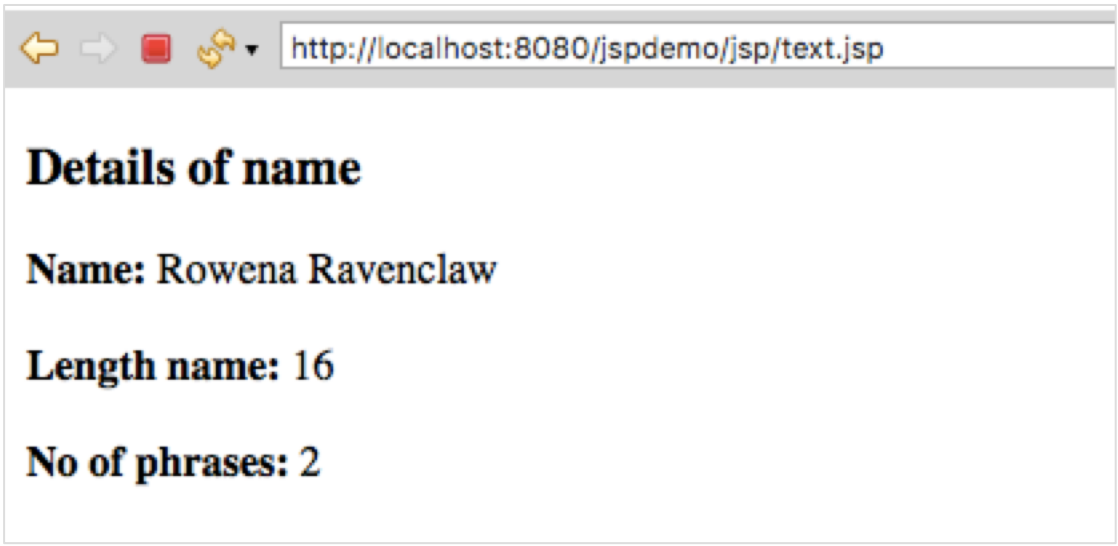

Figure 5: Expected output from text.jsp

Solution for Step 3

The solution for step 3 is highlighted in red as shown in Listing 1.

Listing 1: Amended text.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Demonstration of Text Processing</title>
</head>
<%

      // Define a string to represent a name
      String name = "Rowena Ravenclaw";

      // Get length of the name
      int length = name.length();

      // Get number of phrases in the name
      // insert your code here. Use split to count the phrases in the name
      String words[] = name.split(" ");

%>

<body>

<h3>Details of name</h3>

<b>Name:</b> <%= name %> <br><br>
<b>Length name:</b> <%= length %> <br><br>
<b>No of phrases:</b> <%= words.length %> <br><br>

</body>
</html>
```

Exercise 5: Including a Footer Menu

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

1. Open **text.jsp**.
2. Add the following code before the **</body>**.

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

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 below Figure 6.

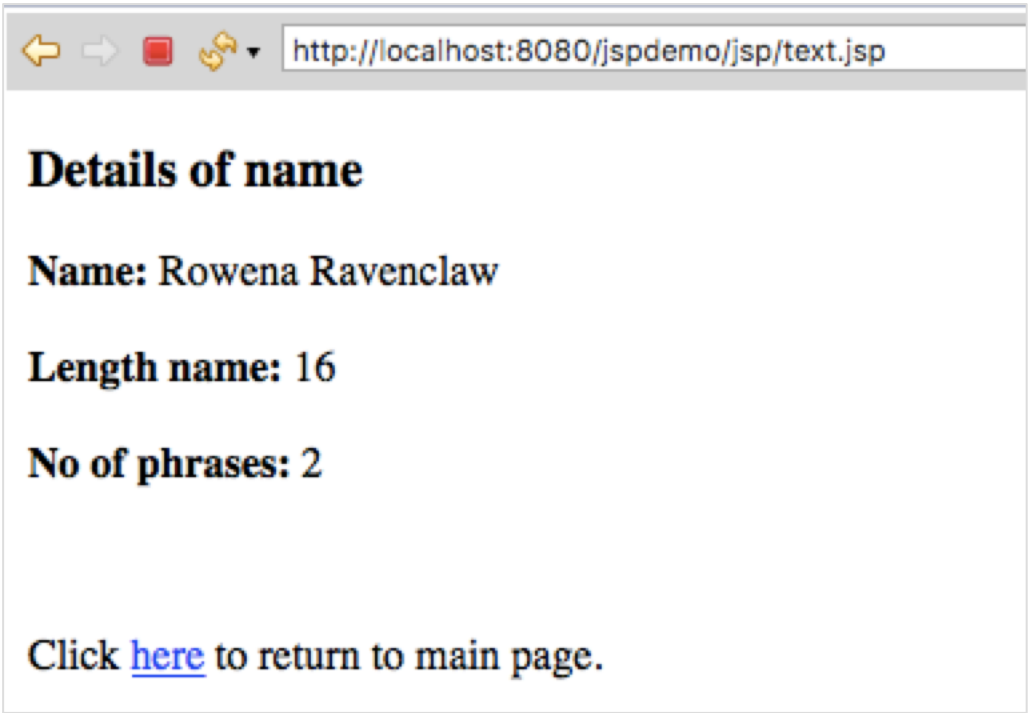

Figure 6: An expected output from `text.jsp` after including `footerMenu.html`

6. Click the link label as **here**. It shall take the browser to the main page as shown in Figure 4.

Solution for Step 2

The solution for step 2 is highlighted in red as shown in Listing 2.

Listing 2: text.jsp with footer

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Demonstration of Text Processing</title>
</head>
<%

      // Define a string to represent a name
      String name = "Rowena Ravenclaw";

      // Get length of the name
      int length = name.length();

      // Get number of phrases in the name
      // insert your code here. Use split to count the phrases in the name
      String words[] = name.split(" ");

%>

<body>

<h3>Details of name</h3>

<b>Name:</b> <%= name %><br><br>
<b>Length name:</b> <%= length %><br><br>
<b>No of phrases:</b> <%= words.length %><br><br>

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

</body>
</html>
```

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 unfinished code. Complete the code to display a list of houses in Hogwarts.
5. You are allowed 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 7.

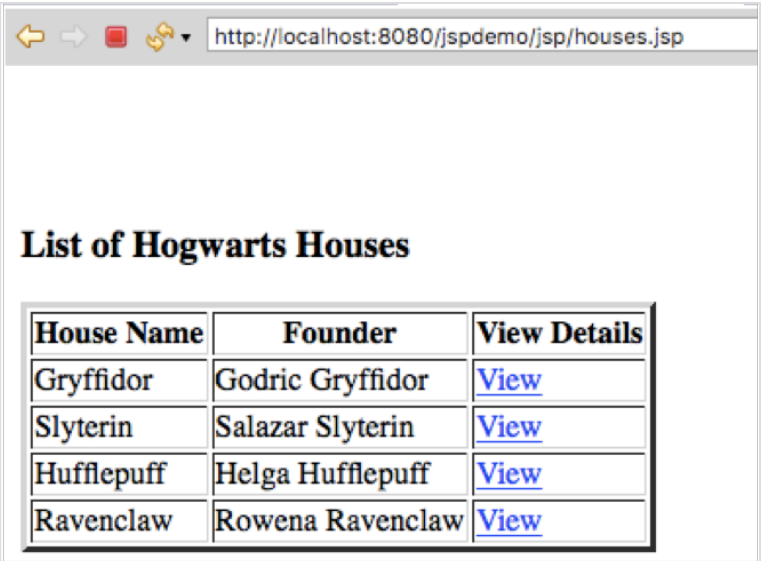

Figure 7: Expected output from houses.jsp

Solution for Step 4

The solution for this step is highlighted in red as shown in Listing 3.

Listing 3: houses.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>


<%@ page import="java.util.List" %>

<%@ page import="demo.hogwarts.House" %>
<%@ page import="demo.hogwarts.HouseFacade" %>


<%--This page shall display a list of houses in Hogwarts --%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>List of Hogwarts Houses</title>
</head>
```

```
<%

	// Get a list of house for HouseFacade, store in List<House>
	HouseFacade houseFacade = new HouseFacade();
	List<House> houses = houseFacade.getHouses();

%>

<body>

<br><br><br>
<h3>List of Hogwarts Houses</h3>

<table border="3">
	<tr>
		<th>House Name</th>
		<th>Founder</th>
		<th>View Details</th>
	</tr>
<%

	// Display the houses using for loop
	for (House house:houses) {
%>
	<tr>
		<td><%= house.getName() %></td>
		<td><%= house.getFounder() %></td>
		<td><a href="details.jsp?id=<%=house.getHouseId() %>">View</a></td>
	</tr>
<%

	}
%>
</table>

</body>
</html>
```

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 6 should appear at the end of the page.
4. Click at the link labelled as **here**. The browser should return to the main page.

Solution for Step 1

The solution for this step is highlighted in red as shown in Listing 4.

Listing 4: houses.jsp with footer.html

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>


<%@ page import="java.util.List" %>

<%@ page import="demo.hogwarts.House" %>
<%@ page import="demo.hogwarts.HouseFacade" %>


<%--This page shall display a list of houses in Hogwarts --%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>List of Hogwarts Houses</title>
</head>

<%

      // Get a list of house for HouseFacade, store in List<House>
      HouseFacade houseFacade = new HouseFacade();
      List<House> houses = houseFacade.getHouses();

%>

<body>

<br><br><br>
<h3>List of Hogwarts Houses</h3>

<table border="3">
      <tr>
            <th>House Name</th>
            <th>Founder</th>
            <th>View Details</th>
      </tr>
<%

      // Display the houses using for loop
      for (House house:houses) {
%>
```

```
		<tr>
			<td><%= house.getName() %></td>
			<td><%= house.getFounder() %></td>
			<td><a href="details.jsp?id=<%=house.getHouseId() %>">View</a></td>
		</tr>
<%

		}
%>
</table>

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


</body>
</html>
```

Display Total Number of Records

1. Add a total number of records 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.

Solution for Step 1

The solution for this step is highlighted in red as shown in Listing 5

Listing 5: houses.jsp with number of records

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>


<%@ page import="java.util.List" %>

<%@ page import="demo.hogwarts.House" %>
<%@ page import="demo.hogwarts.HouseFacade" %>


<%--This page shall display a list of houses in Hogwarts --%>

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>List of Hogwarts Houses</title>
</head>
```

```
<%

	// Get a list of house for HouseFacade, store in List<House>
	HouseFacade houseFacade = new HouseFacade();
	List<House> houses = houseFacade.getHouses();

%>

<body>

<br><br><br>
<h3>List of Hogwarts Houses</h3>

<table border="3">
	<tr>
		<th>House Name</th>
		<th>Founder</th>
		<th>View Details</th>
	</tr>
<%

	// Display the houses using for loop
	for (House house:houses) {
%>
	<tr>
		<td><%= house.getName() %></td>
		<td><%= house.getFounder() %></td>
		<td><a href="details.jsp?id=<%=
house.getHouseId() %>">View</a></td>
	</tr>
<%

	}
%>
</table>

Total records: <%= houses.size() %>

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

</body>
</html>
```

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 8.

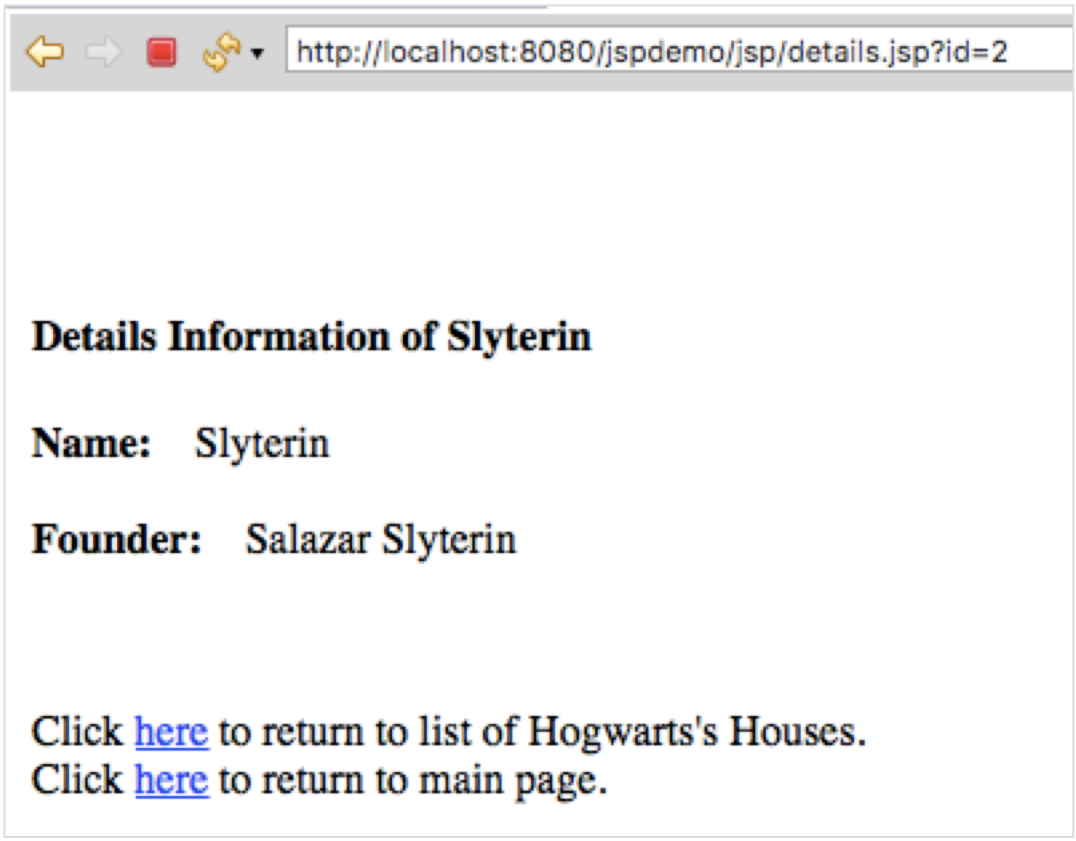

Figure 8: Expected output from `details.jsp`

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 8.
8. Return to the list of Hogwarts's Houses. Click other **View** from other row to test the application.

Solution for Step 3

The solution for step 3 is highlighted in red as shown in Listing 6.

Listing 6: details.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<%-- Import all the necessary classes --%>
<%@ page import="demo.hogwarts.House" %>
<%@ page import="demo.hogwarts.HouseFacade" %>

<%

      // Get parameter value from link
      // Use request.getParameter( ).
      //It is similar practice from the Servlet.
      int houseId = Integer.parseInt(request.getParameter("id"));

      // Wrap into House
      House house = new House();
      house.setHouseId(houseId);

      // Get house. Use house Facade
      HouseFacade houseFacade = new HouseFacade();
      house = houseFacade.getHouse(house);

%>


<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<!-- The title bar should display the house name -->
<title>Details Information of </title>
</head>
<body>

<br><br><br>
<!-- The heading should display the house name -->
<h4>Details Information of </h4>

<b>Name:</b> <%= house.getName() %> <br><br>
<b>Founder:</b> <%= house.getFounder() %> <br><br>

Click <a href="houses.jsp">here</a> to return to list of Hogwart's Houses
<!-- Include a footer menu -->
<jsp:include page="footerMenu.html" />

</body>
</html>
```

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 class.

Solution for Step 1 and 2

The solution for step 1 and 2 is highlighted in red as shown in Listing 7.

Listing 7: houses.java

```
package demo.hogwarts;

/**
 * This class represent a house in Hogwarts
 *
 * @author emalianakasmuri
 *
 */
public class House {

	private int houseId;
	private String name;
	private String founder;
	private String color;
	private String mascot;

	public int getHouseId() {
		return houseId;
	}
	public void setHouseId(int houseId) {
		this.houseId = houseId;
	}
	public String getName() {
		return name;
	}
	public void setName(String name) {
		this.name = name;
	}
	public String getFounder() {
		return founder;
	}
	public void setFounder(String founder) {
		this.founder = founder;
	}
	public String getColor() {
		return color;
	}
	public void setColor(String color) {
		this.color = color;
	}
	public String getMascot() {
		return mascot;
	}
	public void setMascot(String mascot) {
		this.mascot = mascot;
	}

}
```

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.

Solution for Step 3 and 4

The solution for step 3 and 4 is highlighted in red as shown in Listing 8.

Listing 8: `HouseDataManager.java`

```
package demo.hogwarts;

import java.awt.color.ColorSpace;
import java.util.ArrayList;
import java.util.List;

/**
 * This class manages data of Hogwarts house
 *
 * @author emalianakasmuri
 *
 */
public class HouseDataManager {

	private List<House> houses;

	/**
	 * Constructor
	 */
	public HouseDataManager() {

		houses = this.loadHouses();
	}

	/**
	 * This method gets a Hogwarts house
	 * @param house
	 * @return a Hogwarts house
	 */
```

```
	protected House getHouse (House house) {

		// Iterate the list find the house
		for (House currentHouse:houses) {

			if (house.getHouseId() == currentHouse.getHouseId()) {
				// If ids matched, return currentHouse

				return currentHouse;
			}

		}

		// Set id to 0 because no matched id
		house.setHouseId(0);

		return house;

	}

	/**
	 * This method get a list of Hogwarts houses
	 * @return
	 */
	protected List<House> getHouses() {

		return houses;
	}

	/**
	 * This method loads data into list
	 * @return
	 */
	private List<House> loadHouses() {

		// Declaration of sample data
		String houseNames[] = {"Gryffidor", "Slyterin", "Hufflepuff",
"Ravenclaw"};
		String firstNames[] = {"Godric", "Salazar", "Helga", "Rowena"};
		String colors[] = {"scarlet-gold", "green-silver", "yellow-black",
"blue-bronze"};
		String mascots[] = {"lion", "serpent", "badger", "eagle"};

		// Load data into list
		houses = new ArrayList<House>();
		for (int index=0; index < houseNames.length; index++) {

			// Wrap data into object
			House house = new House();
			house.setHouseId(index+1);
			house.setName(houseNames[index]);
			house.setFounder(firstNames[index] + " " +
houseNames[index]);
			house.setColor(colors[index]);
			house.setMascot(mascots[index]);
```

```
                    // Add into list
                    houses.add(house);
              }

              return houses;

        }


}
```

Amend The Presentation Layer

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

Solution for Step 2

The solution for step 3 and 4 is highlighted in red as shown in Listing 9

Listing 9:`detail.jsp` with additional data

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>

<%-- Import all the necessary classes --%>
<%@ page import="demo.hogwarts.House" %>
<%@ page import="demo.hogwarts.HouseFacade" %>


<%

      // Get parameter value from link
      // Use request.getParameter( ).
      //It is similar practice from the Servlet.
      int houseId = Integer.parseInt(request.getParameter("id"));

      // Wrap into House
      House house = new House();
      house.setHouseId(houseId);

      // Get house. Use house Facade
      HouseFacade houseFacade = new HouseFacade();
      house = houseFacade.getHouse(house);

%>


<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<!-- The title bar should display the house name -->
<title>Details Information of </title>
</head>
<body>

<br><br><br>
<!-- The heading should display the house name -->
<h4>Details Information of </h4>
```

```
<b>Name:</b> <%= house.getName() %><br><br>
<b>Founder:</b> <%= house.getFounder() %><br><br>
<b>Color:</b> <%= house.getColor() %><br><br>
<b>Mascot:</b> <%= house.getMascot() %><br><br>

Click <a href="houses.jsp">here</a> to return to list of Hogwart's Houses
<!-- Include a footer menu -->
<jsp:include page="footerMenu.html" />

</body>
</html>
```

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 9.

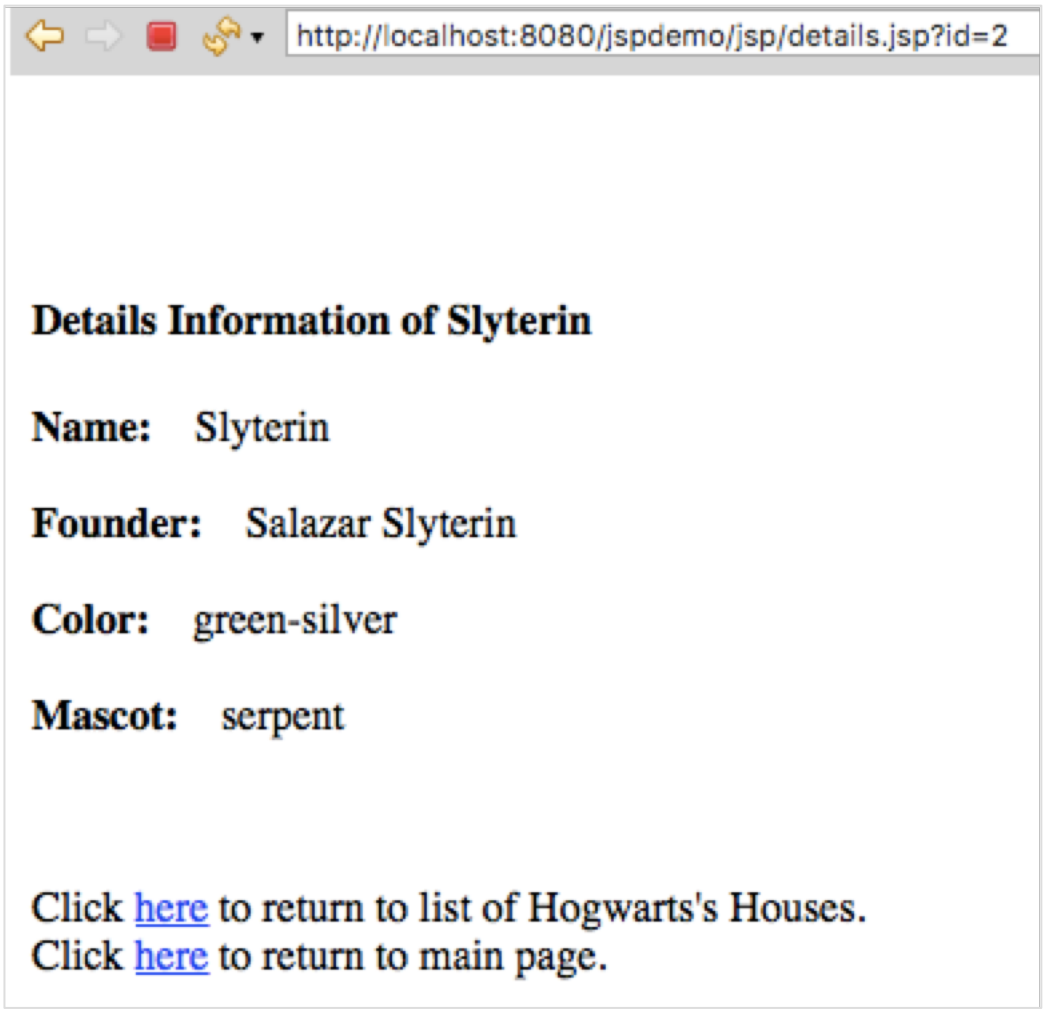

Figure 9: Expected output after adding applying changes to the houses data

Outcome for Step 2

The student should get the output as shown in Figure 9