JSP Action Tags

Step 1: Creating a dynamic web project

- Open Eclipse
- Go the File menu. Choose New->Dynamic Web Project
- Enter the project name as JSPActionTags. Click on Next
- Enter nothing in the next screen and click on Next
- Check the checkbox Generate web.xml deployment descriptor and click on Finish
- This will create the project files in the Project Explorer

Step 2: Creating a JSP file index.jsp

- In the Project Explorer, expand the project JSPActionTags
- Expand WebContent. Right click on WebContent. Choose New->JSP File
- Enter the filename as index.jsp and click on Finish
- Enter the following code:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JSP Action Tags</title>
</head>
<body>
Usage of useBean tag<br>
<jsp:useBean id="productBean"
class="com.ecommerce.ProductBean" scope="session">
</jsp:useBean>
	<jsp:setProperty property="productId" name="productBean"
```

```
value="18791"/>
	<jsp:setProperty property="productName" name="productBean"
value="Optical Wireless Mouse"/>
	<jsp:setProperty property="price" name="productBean"
value="600.00"/>

<a href="showbean.jsp">Access bean properties from another
page</a><br>
<a href="forward.jsp">Use Forward action to go to Google</a><br>

<hr>

<jsp:text>
	<![CDATA[This is my content.<br/>This will show within a text
action tag exactly as it has been entered]]>
</jsp:text>

</body>
</html>
```

- Click on the Save icon

Step 3: Creating a JSP file forward.jsp

- In the Project Explorer, expand the project JSPActionTags
- Expand WebContent. Right click on WebContent. Choose New->JSP File
- Enter the filename as forward.jsp and click on Finish
- Enter the following code:

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

```
<jsp:forward page="forwarded.jsp"></jsp:forward>
```

- Click on the Save icon

Step 4: Creating a JSP file forwarded.jsp

- In the Project Explorer, expand the project JSPActionTags
- Expand WebContent. Right click on WebContent. Choose New->JSP File
- Enter the filename as forwarded.jsp and click on Finish
- Enter the following code:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Forward Test</title>
</head>
<body>
This page came from forward.jsp by using jsp:forward action tag

</body>
</html>
```

- Click on the Save icon

Step 5: Creating a JSP file showbean.jsp

- In the Project Explorer, expand the project JSPActionTags
- Expand WebContent. Right click on WebContent. Choose New->JSP File

- Enter the filename as showbean.jsp and click on Finish
- Enter the following code:

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
pageEncoding="UTF-8"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Display Bean properties</title>
</head>
<body>
<jsp:useBean id="productBean"
class="com.ecommerce.ProductBean" scope="session">
</jsp:useBean>
	Product Id:	<jsp:getProperty name="productBean"
property="productId" />  <br>
	Product Name:	<jsp:getProperty name="productBean"
property="productName" />  <br>
	Product Price:	<jsp:getProperty name="productBean"
property="price" />  <br>

</body>
</html>
```

- Click on the Save icon

Step 6: Creating a Bean class ProductBean.java

- In the Project Explorer, expand the project JSPActionTags
- Expand JavaResources. Right click on src. Choose New-> Class
- Enter Package as com.ecommerce

- Enter Name as ProductBean and click on Finish
- Enter the following code:

```
package com.ecommerce;
import java.io.Serializable;

public class ProductBean implements Serializable{
	private String productId;
	private String productName;
	private double price;

	public void setProductId(String value) {
		this.productId = value;
	}
	public void setProductName(String value) {
		this.productName = value;
	}
	public void setPrice(double value) {
		this.price = value;
	}

	public String getProductId() {
		return this.productId;
	}
	public String getProductName() {
		return this.productName;
	}
	public double getPrice() {
		return this.price;
	}


}
```

- Click on the Save icon

Step 7: Checking for servlet-api.jar

- Before building the project, we need to add servlet-api.jar to the project

- Servlet-api.jar file is already present in your practice lab. (Refer FSD: Lab Guide - Phase 2)

- To add it to the project, follow the below mentioned steps:

    - In the Project Explorer, right click on JSPActionTags and choose Properties
    - Select Java Build Path from the options on the left
    - Click on Libraries tab on the right
    - Under ClassPath, expand the node that says Apache Tomcat
    - If there is an existing entry for servlet-api.jar, then click on Cancel and exit the window
    - If it is not there, then click on Classpath entry and click on Add External JARs button on the right
    - From the file list, select servlet-api.jar file and click on Ok
    - Click on Apply and Close

Step 8: Building the project

- From the Project menu at the top, click on Build
- If any compile errors are shown, fix them as required

Step 9: Publishing and starting the project

- If you do not see the Servers tab near the bottom of the IDE, go to Window menu and click on Show View->Servers
- Right click on the Server entry and choose Add and Remove
- Click the Add button to move JSPActionTags from the Available list to the Configured list
- Click on Finish
- Right click on the Server entry and click on Publish
- Right click on the Server entry and click on Start
- This will start the server

Step 10: Running the project

- To run the project, open a web browser and type: http://localhost:8080/JSPActionTags

Step 11: Pushing the code to your GitHub repositories

- Open your command prompt and navigate to the folder where you have created your files.

```
cd <folder path>
```

- Initialize your repository using the following command:

  ```
  git init
  ```

- Add all the files to your git repository using the following command:

  ```
  git add .
  ```

- Commit the changes using the following command:

  ```
  git commit .  -m “Changes have been committed.”
  ```

- Push the files to the folder you initially created using the following command:

  ```
  git push -u origin master
  ```