JSP Implicit Objects

Step 1: Creating a dynamic web project

- Open Eclipse
- Go the File menu. Choose New->Dynamic Web Project
- Enter the project name as JSPImplicitObjects. 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 JSPImplicitObjects
- 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"%>
<%@ page errorPage = "handle-error.jsp" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>JSP Implicit Objects</title>
</head>
<body>

<%
```

```
        String responseCheck = request.getParameter("office");
        if (responseCheck != null ) {

response.setStatus(response.SC_MOVED_TEMPORARILY);
                response.setHeader("Location", "response-redirect.jsp?
office=" + responseCheck);
        }

        String errorCheck = request.getParameter("error");
        if (errorCheck != null ) {
                int x = 0;
                if (x == 0)
                        throw new RuntimeException("Exception has been
raised");
        }
%>
<%

  int serverPort =  request.getServerPort() ;
  out.println("The Server is running on port " +
String.valueOf(serverPort) + "<br>");
  out.println("Servlet Name is " + config.getServletName() + "<br>");
  out.println("Server Info:" + application.getServerInfo() + "<br>");

  String pageName = page.toString();
  out.println("The name of the page is " + pageName + "<br>");

  pageContext.setAttribute("userid", "John Doe");
  out.println("userId attribute from pageContext: " +
pageContext.getAttribute("userid") + "<br>");
%>
<a href="index.jsp?office=head_office">Show usage of response
object</a><br>
<a href="index.jsp?error=1">Show usage of error  object</a><br>


<%
  if (response.containsHeader("Office"))
```

```
            out.println("Current location is " +
response.getHeader("Office"));
 %>




</body>
</html>
```

- Click on the Save icon

Step 3: Creating a JSP file response-redirect.jsp

- In the Project Explorer, expand the project JSPImplicitObjects
- Expand WebContent. Right click on WebContent. Choose New->JSP File
- Enter the filename as response-redirect.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>Get Header Example</title>
</head>
<body>

<%
        String office = request.getParameter("office");
        if (office != null)
                out.println("value of Office obtained :" + office + "<br>");
```

```
        else
                out.println("No value of Office found<br>");
%>

</body>
</html>
```

- Click on the Save icon

Step 4: Creating a JSP file handle-error.jsp

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

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

<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Error Handling page</title>
</head>
<body>


<% exception.printStackTrace(response.getWriter()); %>
<hr>
An exception was generated. Details are above:<br>
</body>
</html>
```

- Click on the Save icon

Step 5: 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 JSPImplicitObjects 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 6: Building the project

- From the Project menu at the top, click on Build

- If any compile errors are shown, fix them as required

Step 7: 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 JSPImplicitObjects 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 8: Running the project

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

Step 9: 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