Experiment No.___ *Date___/___/2022*

TITLE OF EXPERIMENT: - A JAVA Program to write a simple JSP page to display a simple message.

DIVISION:__________________________ BRANCH: ____________________

BATCH:______________________________ ROLL NO.: __________________

PERFORMED ON DATE: ____

SIGNATURE OF TEACHING STAFF:

EXPERIMENT NO. 10

Aim: Write a simple JSP page to display a simple message.

Software:

1.	Command prompt
2.	JDK 16
3.	Internet

Theory:

JSP Technology

JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, JSTL, etc.

A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

Advantages of JSP over Servlet

There are many advantages of JSP over the Servlet. They are as follows:

1) Extension to Servlet

JSP technology is the extension to Servlet technology. We can use all the features of the Servlet in JSP. In addition to, we can use implicit objects, predefined tags, expression language and Custom tags in JSP, that makes JSP development easy.

2) Easy to maintain

JSP can be easily managed because we can easily separate our business logic with presentation logic. In Servlet technology, we mix our business logic with the presentation logic.

3) Fast Development: No need to recompile and redeploy

If JSP page is modified, we don't need to recompile and redeploy the project. The Servlet code needs to be updated and recompiled if we have to change the look and feel of the application.

4) Less code than Servlet

In JSP, we can use many tags such as action tags, JSTL, custom tags, etc. that reduces the code. Moreover, we can use EL, implicit objects, etc.

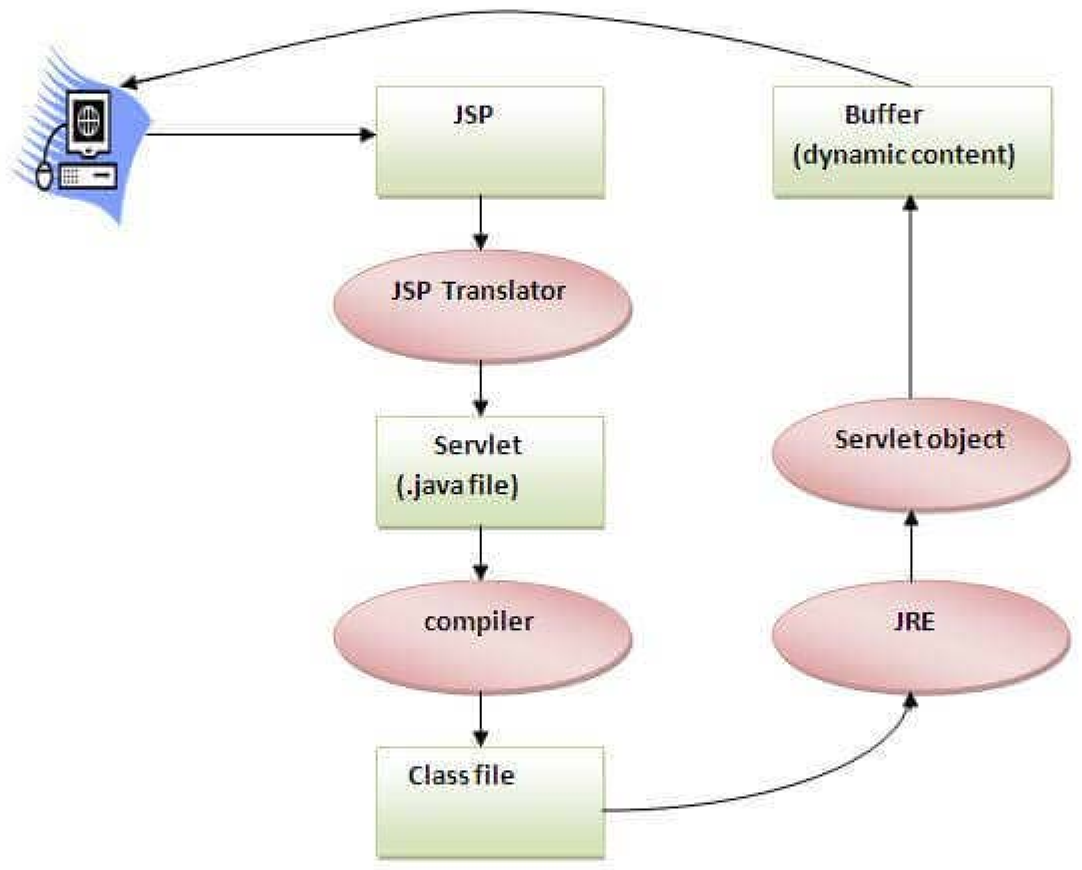
The Lifecycle of a JSP Page

The JSP pages follow these phases:

- Translation of JSP Page
- Compilation of JSP Page
- Classloading (the classloader loads class file)
- Instantiation (Object of the Generated Servlet is created).
- Initialization (the container invokes jspInit() method).
- Request processing (the container invokes _jspService() method).

- Destroy (the container invokes jspDestroy() method).

Note: jspInit(), _jspService() and jspDestroy() are the life cycle methods of JSP.



As depicted in the above diagram, JSP page is translated into Servlet by the help of JSP translator. The JSP translator is a part of the web server which is responsible for translating the JSP page into Servlet. After that, Servlet page is compiled by the compiler and gets converted into the class file. Moreover, all the processes that happen in Servlet are performed on JSP later like initialization, committing response to the browser and destroy.

Creating a simple JSP Page

To create the first JSP page, write some HTML code as given below, and save it by .jsp extension. We have saved this file as index.jsp. Put it in a folder and paste the folder in the web-apps directory in apache tomcat to run the JSP page.

index.jsp

Let's see the simple example of JSP where we are using the scriptlet tag to put Java code in the JSP page. We will learn scriptlet tag later.

```
<html>
<body>
<% out.print(2*5); %>
</body>
</html>
```

It will print **10** on the browser.

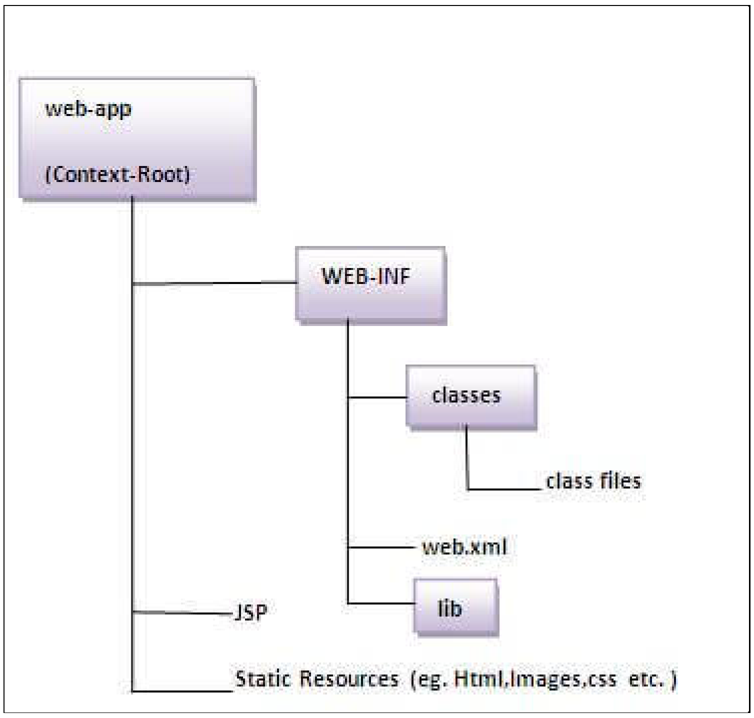
How to run a simple JSP Page?

Follow the following steps to execute this JSP page:

- Start the server
- Put the JSP file in a folder and deploy on the server
- Visit the browser by the URL http://localhost:portno/contextRoot/jspfile, for example, http://localhost:8888/myapplication/index.jsp

The Directory structure of JSP

The directory structure of JSP page is same as Servlet. We contain the JSP page outside the WEB-INF folder or in any directory.



Do I need to follow the directory structure to run a simple JSP?

No, there is no need of directory structure if you don't have class files or TLD files. For example, put JSP files in a folder directly and deploy that folder. It will be running fine. However, if you are using Bean class, Servlet or TLD file, the directory structure is required.

Program:

The Steps

1. **Open eclipse and choose your workspace:**
   - Open the folder where you unzipped eclipse
   - Double click "eclipse.exe"
   - Browse to your workspace and then click OK
   - If the Welcome page is opened, go to the workbench by clicking this icon
2. **Create a new Dynamic web project:**
   - In the menu bar File / New / Dynamic Web Project
   - Name your project - To be consistent with the tutorial, name it "JSPExample". Your project should appear in the "Project Explorer" view located at the left side

of the eclipse window. If the Project Explorer view is not shown, you can show it from Window / Show View / Project Explorer.

3. **Create a JSP file:**
   - In the "Project Explorer" view, expand your project by clicking the project's corresponding '+' sign
   - R-click WebContent / New / JSP. If JSP is not shown in the list, go to other / Web / JSP.
   - Name your JSP (To be consistent with the tutorial, name it "myFirstJSP")
4. **Write HTML code to display "Hello World!"** at the center of the page. To do that place the following code:

```
<center>

   <font color="gray" size="7"> Hello World! </font>

</center>
```

in the body of your HTML body (i.e between the body tags <body> ... </body>). The result HTML document should look as follows:

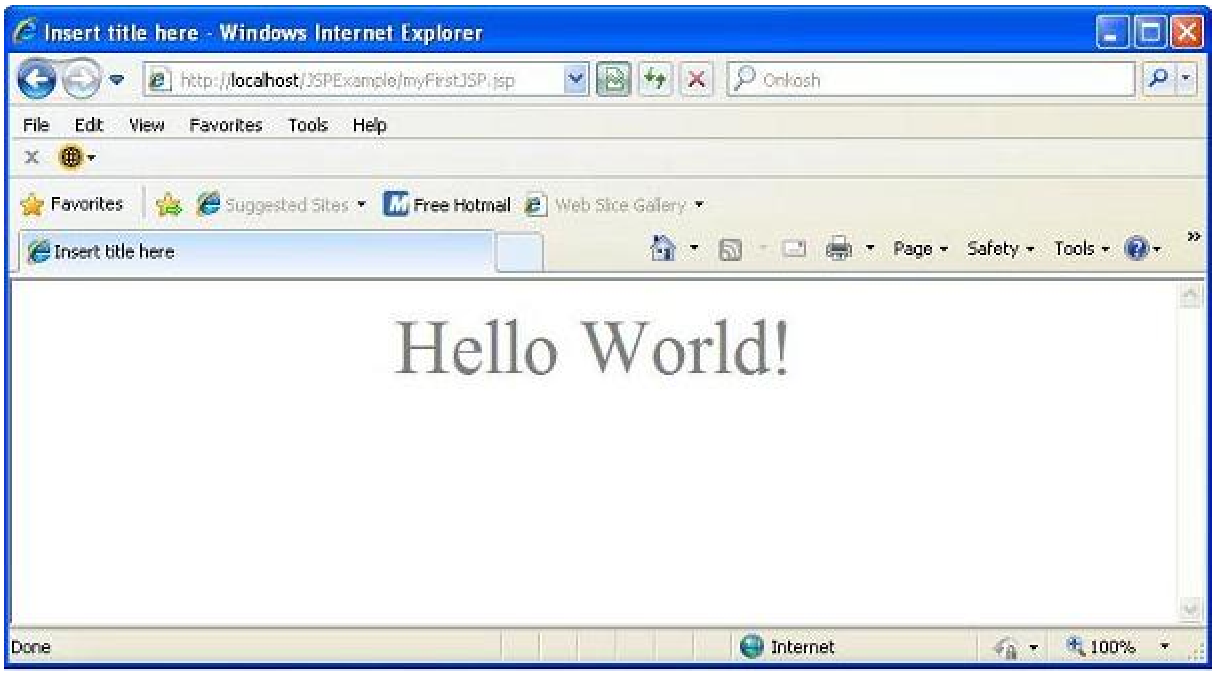
```
<%@ page language="java"

            contentType="text/html; charset=ISO-8859-1"

            pageEncoding="ISO-8859-1"

%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"

            "http://www.w3.org/TR/html4/loose.dtd">

<html>

  <head>

    <meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

    <title>Insert title here</title>

  </head>

  <body>

    <center>

      <font color="gray" size="7">  Hello World!  </font>

    </center>

  </body>

</html>
```

5. **Add your project to "Tomcat":**
   - Open the server view by clicking the "Servers" tab at the bottom of your page. If the "Servers" tab is not shown, you can show it from the menu bar, Window / Show View/ Servers. Or Window / Show View / Otherâ€¦ / Server / Servers.
   - In the "Servers" view, R-click Tomcat-server record / Add and Remove Projects. (If Tomcat-server record is not there, please check Steps 5-6 in the Configurations section by clicking next from here)
   - Add your project
6. **Start "Tomcat":** In the "Servers" view, R-click Tomcat-server record / Start
7. **Test your project:**
   - In your web browser (Internet Explorer or Firefox) :
     - If the port is set to 80: Type http://localhost/YourProjectName - In our example the URL is http://localhost/JSPExample
     - If the port is set to 8080: Type http://localhost:8080/YourProjectName - In our example the URL is http://localhost:8080/JSPExample

     Please note that Tomcat's default port is 8080, but the version refered to in the Downloads section (the pre-configured version), has its port set to 80. We will proceed in this tutorial assuming that the port is set to 80. (You can check your version's port from Tomcat-Installation-Directory / conf / server.xml)

   - Click your JSP file name shown in the directory listing - in our example it is myFirstJSP.jsp

   This should be the result:


Insert title here - Windows Internet Explorer
http://localhost/JSPExample/myFirstJSP.jsp
Onkosh
File Edit View Favorites Tools Help
Favorites
Suggested Sites
Free Hotmail
Web Slice Gallery
Insert title here
Page Safety Tools
Done
Internet
100%

Hello World!

8. **Set your project's welcome file** (let the server open "myFirstJSP.jsp" once you open the project -- to pass the second part of step 7 --).
   - Open "web.xml": WebContent / Web-INF / R-click web.xml / Open With / Text Editor
   - Set the "welcome-file-list" to your home page (In our example it is myFirstJSP.jsp). Place:

```
<welcome-file> myFirstJSP.jsp</welcome-file>
```

   in the "welcome-file-list" between:

```
<welcome-file-list>  and </welcome-file-list>
```

- web.xml should look as follows:

```
<?xml version="1.0" encoding="UTF-8"?>

<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

   xmlns="http://java.sun.com/xml/ns/javaee"

   xmlns:web="http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"

   xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

      http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd"

   id="WebApp_ID" version="2.5">

  <display-name>JSPExample</display-name>

  <welcome-file-list>

   <welcome-file>myFirstJSP.jsp</welcome-file>

  </welcome-file-list>

</web-app>
```

Restart the server and test your project again using http://localhost/yourProjectName - in our example, the URL will be http://localhost/JSPExample/ .

Please note that the server does not recognize changes in "web.xml" except if you restarted, or started, the server

If you have passed these steps successfully and saw the intended results, this means that you were able to use the JSP technology to display ordinary, static, HTML pages

Conclusion: