

**Experiment No. 9**

**Aim of the Experiment:** Write a simple JSP page to display a simple message (It may be a simple html page).

**Objective:**

To display a message using JSP page.

**Resources:** Eclipse IDE 2018, JDK 1.8.0 is required, Apache tomcat 9.0 server

**Course Outcome Addressed:** CO6

**Theory:**

JavaServer Pages (JSP) is a technology for developing Webpages that supports dynamic content. This helps developers insert java code in HTML pages by making use of special JSP tags, most of which start with <% and end with %>. A JSP page looks similar to an HTML page, but a JSP page also has Java code in it. We can put any regular Java Code in a JSP file using a scriptlet tag which start with <% and ends with %>. JSP pages are used to develop dynamic responses.

A JavaServer Pages component is a type of Java servlet that is designed to fulfill the role of a user interface for a Java web application. Web developers write JSPs as text files that combine HTML or XHTML code, XML elements, and embedded JSP actions and commands.

Using JSP, you can collect input from users through Webpage forms, present records from a database or another source, and create Webpages dynamically.

JSP tags can be used for a variety of purposes, such as retrieving information from a database or registering user preferences, accessing JavaBeans components, passing control between pages, and sharing information between requests, pages etc.

JavaServer Pages often serve the same purpose as programs implemented using the Common Gateway Interface (CGI).

**Advantages of JSP in comparison with the CGI:**

- Performance is significantly better because JSP allows embedding Dynamic Elements in HTML Pages itself instead of having separate CGI files.
- JSP are always compiled before they are processed by the server unlike CGI/Perl which requires the server to load an interpreter and the target script each time the page is requested.

**Subject: Advanced Java Programming Lab (Elective - II) (304198)(C)**

Exp. No.9

- JavaServer Pages are built on top of the Java Servlets API, so like Servlets, JSP also has access to all the powerful Enterprise Java APIs, including JDBC, JNDI, EJB, JAXP, etc.
- JSP pages can be used in combination with servlets that handle the business logic, the model supported by Java servlet template engines.

Finally, JSP is an integral part of Java EE, a complete platform for enterprise class applications. This means that JSP can play a part in the simplest applications to the most complex and demanding.

**Advantages of JSP:**

vs. Active Server Pages (ASP)

The advantages of JSP are twofold. First, the dynamic part is written in Java, not Visual Basic or other MS specific language, so it is more powerful and easier to use. Second, it is portable to other operating systems and non-Microsoft Web servers.

vs. Pure Servlets

It is more convenient to write (and to modify!) regular HTML than to have plenty of `println` statements that generate the HTML.

vs. Server-Side Includes (SSI)

SSI is really only intended for simple inclusions, not for "real" programs that use form data, make database connections, and the like.

vs. JavaScript

JavaScript can generate HTML dynamically on the client but can hardly interact with the web server to perform complex tasks like database access and image processing etc.

vs. Static HTML

Regular HTML, of course, cannot contain dynamic information.

**JSP Processing:**

The following steps explain how the web server creates the Webpage using JSP –

- As with a normal page, your browser sends an HTTP request to the web server.
- The web server recognizes that the HTTP request is for a JSP page and forwards it to a JSP engine. This is done by using the URL or JSP page which ends with `.jsp` instead of `.html`.
- The JSP engine loads the JSP page from disk and converts it into a servlet content. This conversion is very simple in which all template text is converted to `println( )` statements

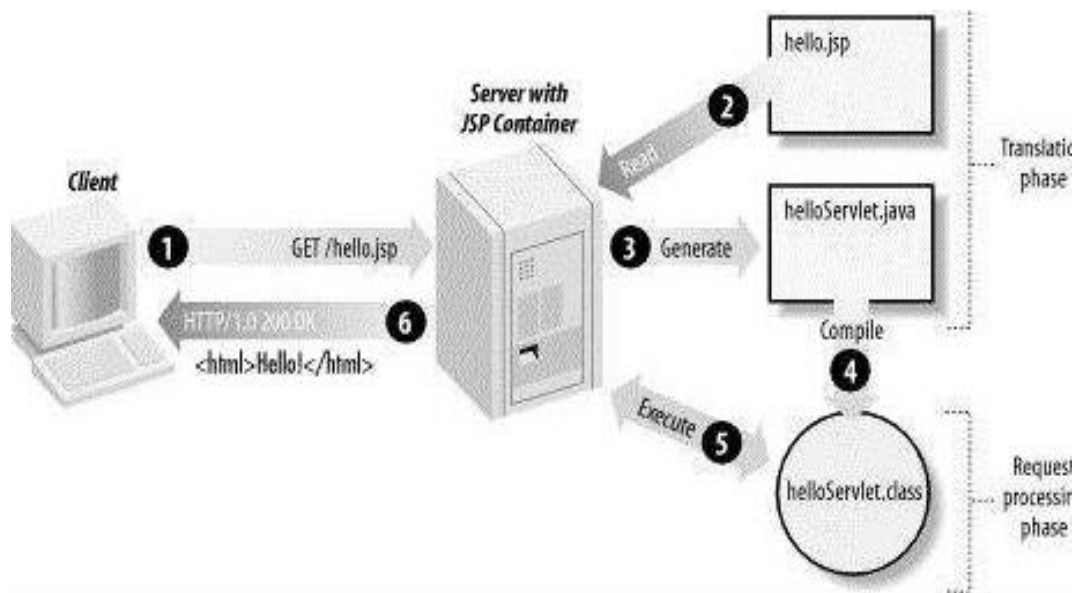
**Subject: Advanced Java Programming Lab (Elective - II) (304198)(C)**

Exp. No.9

and all JSP elements are converted to Java code. This code implements the corresponding dynamic behavior of the page.

- The JSP engine compiles the servlet into an executable class and forwards the original request to a servlet engine.
- A part of the web server called the servlet engine loads the Servlet class and executes it. During execution, the servlet produces an output in HTML format. The output is further passed on to the web server by the servlet engine inside an HTTP response.
- The web server forwards the HTTP response to your browser in terms of static HTML content.
- Finally, the web browser handles the dynamically-generated HTML page inside the HTTP response exactly as if it were a static page.

All the above mentioned steps can be seen in the following diagram



Typically, the JSP engine checks to see whether a servlet for a JSP file already exists and whether the modification date on the JSP is older than the servlet. If the JSP is older than its generated servlet, the JSP container assumes that the JSP hasn't changed and that the generated servlet still matches the JSP's contents. This makes the process more efficient than with the other scripting languages (such as PHP) and therefore faster.

So in a way, a JSP page is really just another way to write a servlet without having to be a Java programming wiz. Except for the translation phase, a JSP page is handled exactly like a regular servlet

**Steps for creating a JSP Page in Eclipse:**

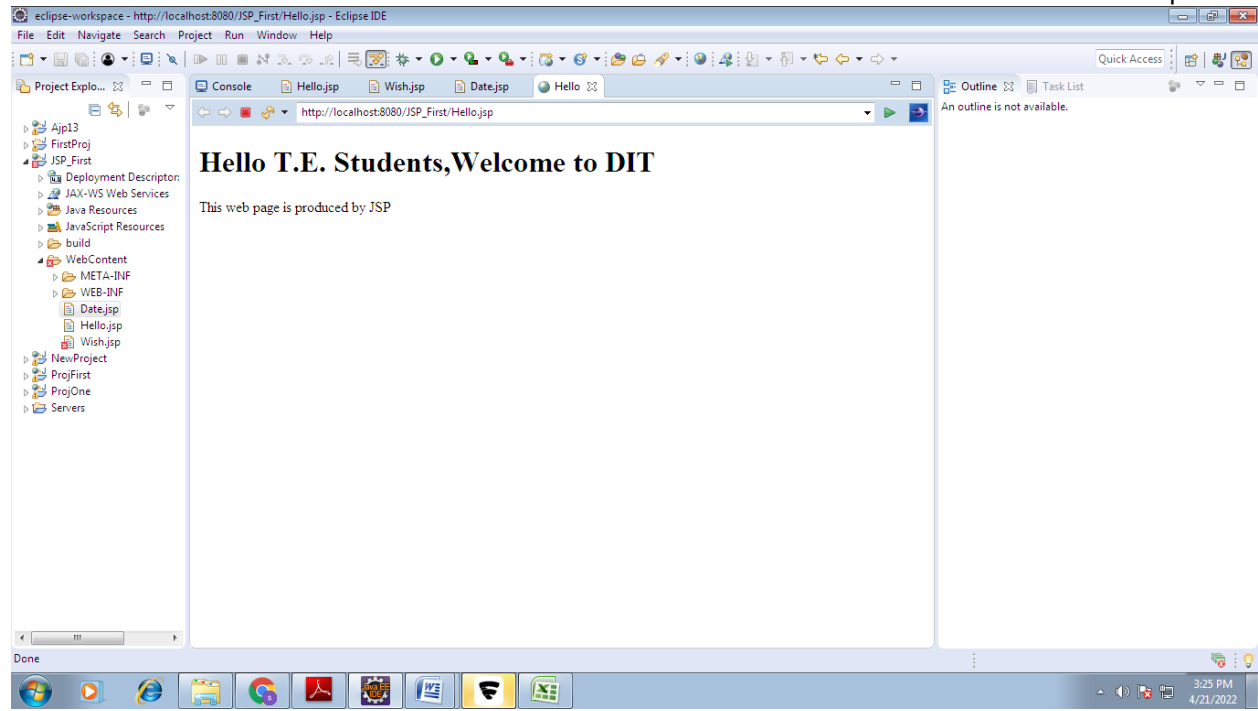
1. Open Eclipse, Click on **New** → **Dynamic Web Project**
2. Give a name to your project and click on OK
3. You will see a new project created in Project Explorer
4. To create a new JSP file right click on Web Content directory, **New** → **JSP file**
5. Give a name to your JSP file and click Finish.
6. Write something in your JSP file. The complete HTML and the JSP code, goes inside the `<body>` tag, just like HTML pages.
7. To run your project, right click on **Project**, select **Run As** → **Run on Server**
8. To start the server, Choose existing server name and click on finish.
9. See the Output in your browser.

**SOURCE CODE for Hello:**

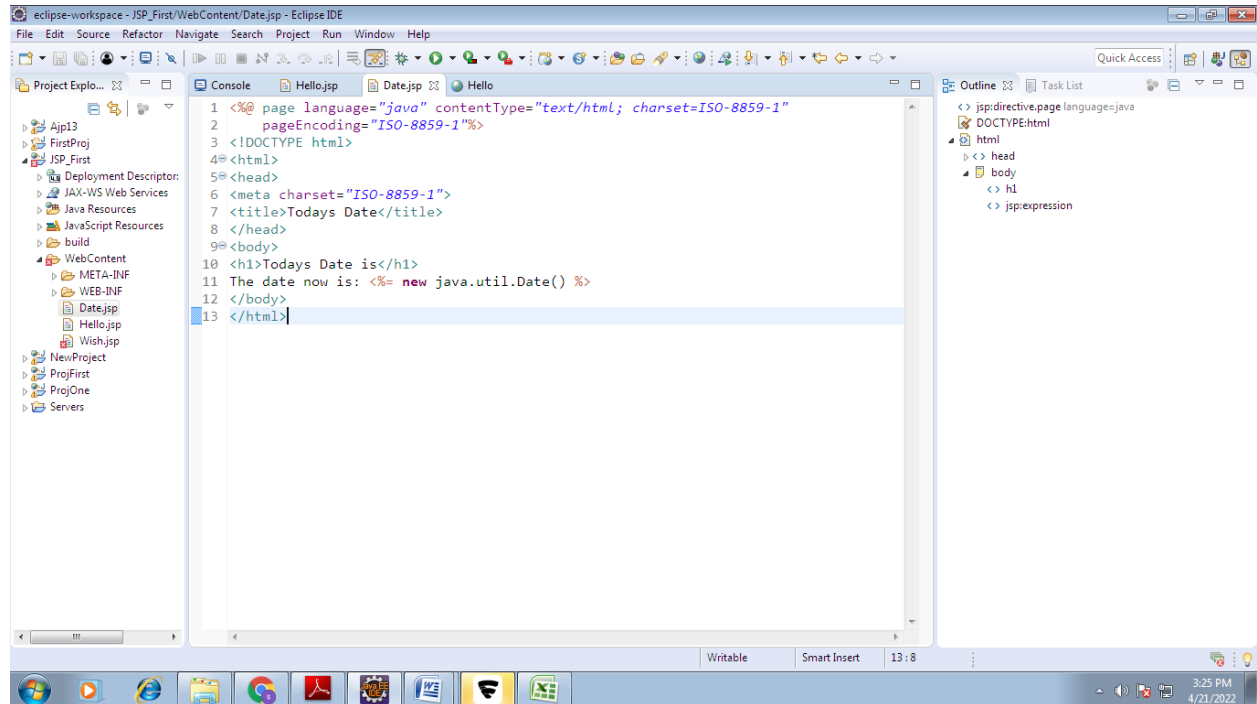
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Hello</title>
</head>
<body>
<h1>Hello T.E. Students, Welcome to DIT</h1>
<%
    out.println("<p>This web page is produced by JSP");
%>
</body>
</html>
```

**OUTPUT:**

```
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="ISO-8859-1">
7 <title>Todays Date</title>
8 </head>
9 <body>
10 <h1>Todays Date is</h1>
11 The date now is: <%= new java.util.Date() %>
12 </body>
13 </html>
```

**SOURCE CODE for Date:**

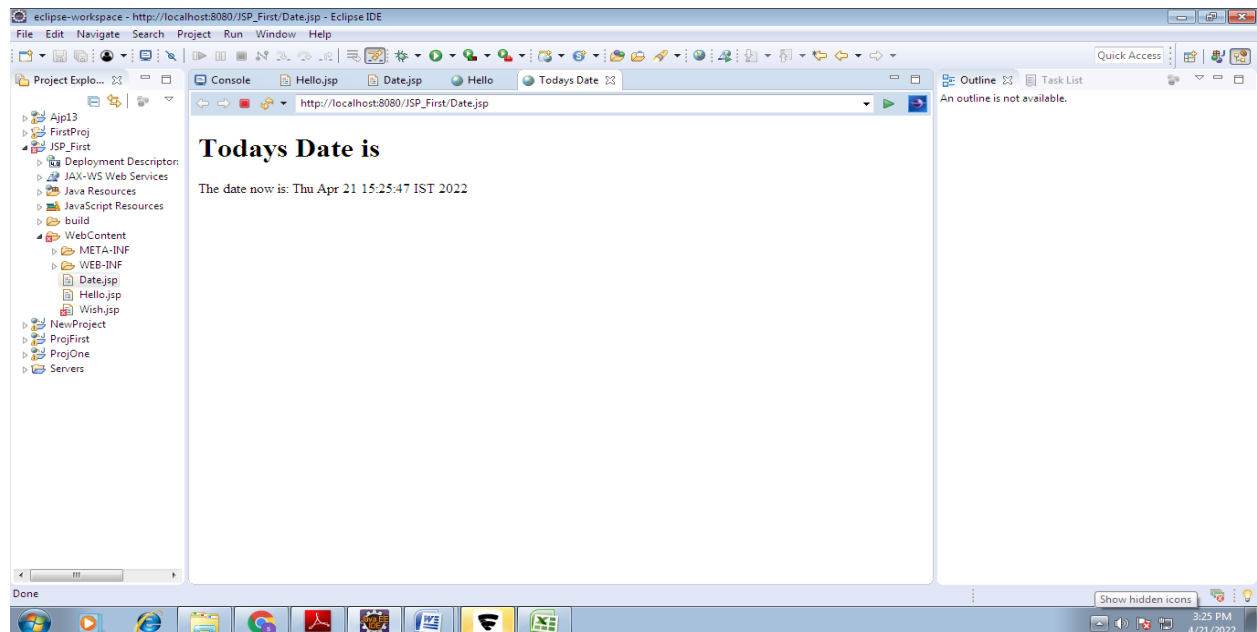
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Todays Date</title>
</head>
<body>
<h1>Todays Date is</h1>
The date now is: <%= new java.util.Date() %>
</body>
</html>
```

**OUTPUT:**

The screenshot shows the Eclipse IDE with the file `Date.jsp` open. The code is as follows:

```
1 <%@ page language="java" contentType="text/html; charset=ISO-8859-1"
2   pageEncoding="ISO-8859-1"%>
3 <!DOCTYPE html>
4 <html>
5 <head>
6 <meta charset="ISO-8859-1">
7 <title>Todays Date</title>
8 </head>
9 <body>
10 <h1>Todays Date is</h1>
11 The date now is: <%= new java.util.Date() %>
12 </body>
13 </html>
```

The Outline view on the right shows the document structure: `jsp:directive.page language=java`, `DOCTYPE:html`, `html`, `head`, `body`, `h1`, and `jsp:expression`.



**Conclusion:**

.

**References:**

Herbert Schildt, "Java : The Complete Reference" Tata McGraw-Hill (7<sup>th</sup> Edition).

**Questions:**

1. What is JSP?
2. How JSP work?
3. What are the life-cycle methods for a JSP?
4. List out some advantages of using JSP.
5. What are Servlets?