

JSP

By
Sudha Agarwal

INDEX

❖ JSP Introduction



WHAT IS JSP?

- ❖ **JSP (JavaServer Pages)** is server side technology to create dynamic java web application. JSP can be thought as an extension to servlet technology because it provides features to easily create user views.
- ❖ JSP Page consists of HTML code and provide option to include java code for dynamic content. Since web applications contain a lot of user screens, JSPs are used a lot in web applications.
- ❖ 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.
- ❖ Using JSP, you can collect input from users through Webpage forms, present records from a database or another source, and create Webpages dynamically.

ADVANTAGES OF JSP OVER SERVLETS

- ❖ We can generate HTML response from servlets also but the process is cumbersome and error prone, when it comes to writing a complex HTML response, writing in a servlet will be a nightmare. JSP helps in this situation and provide us flexibility to write normal HTML page and include our java code only where it's required.
- ❖ JSP provides additional features such as tag libraries, expression language, custom tags that helps in faster development of user views.
- ❖ JSP pages are easy to deploy, we just need to replace the modified page in the server and container takes care of the deployment. For servlets, we need to recompile and deploy whole project again.

ADVANTAGES OF JSP OVER SERVLETS

- ❖ JSP does not require additional files like, java class files, web.xml etc
- ❖ Actually Servlet and JSPs compliment each other.
- ❖ We should use Servlet as server side controller and to communicate with model classes whereas JSPs should be used for presentation layer.

LIFE CYCLE OF JSP PAGE

❖ JSP life cycle is also managed by container. Usually every web container that contains servlet container also contains JSP container for managing JSP pages.

❖ JSP pages life cycle phases are:

1. **Translation** – JSP pages doesn't look like normal java classes, actually JSP container parse the JSP pages and translate them to generate corresponding servlet source code. If JSP file name is home.jsp, usually its named as home_jsp.java.
2. **Compilation** – If the translation is successful, then container compiles the generated servlet source file to generate class file.

LIFE CYCLE OF JSP PAGE

3. **Class Loading** – Once JSP is compiled as servlet class, its lifecycle is similar to servlet and it gets loaded into memory.
4. **Instance Creation** – After JSP class is loaded into memory, its object is instantiated by the container.
5. **Initialization** – The JSP class is then initialized and it transforms from a normal class to servlet. After initialization, ServletConfig and ServletContext objects become accessible to JSP class.
6. **Request Processing** – For every client request, a new thread is spawned with ServletRequest and ServletResponse to process and generate the HTML response.
7. **Destroy** – Last phase of JSP life cycle where it's unloaded into memory.

LIFE CYCLE METHODS OF JSP

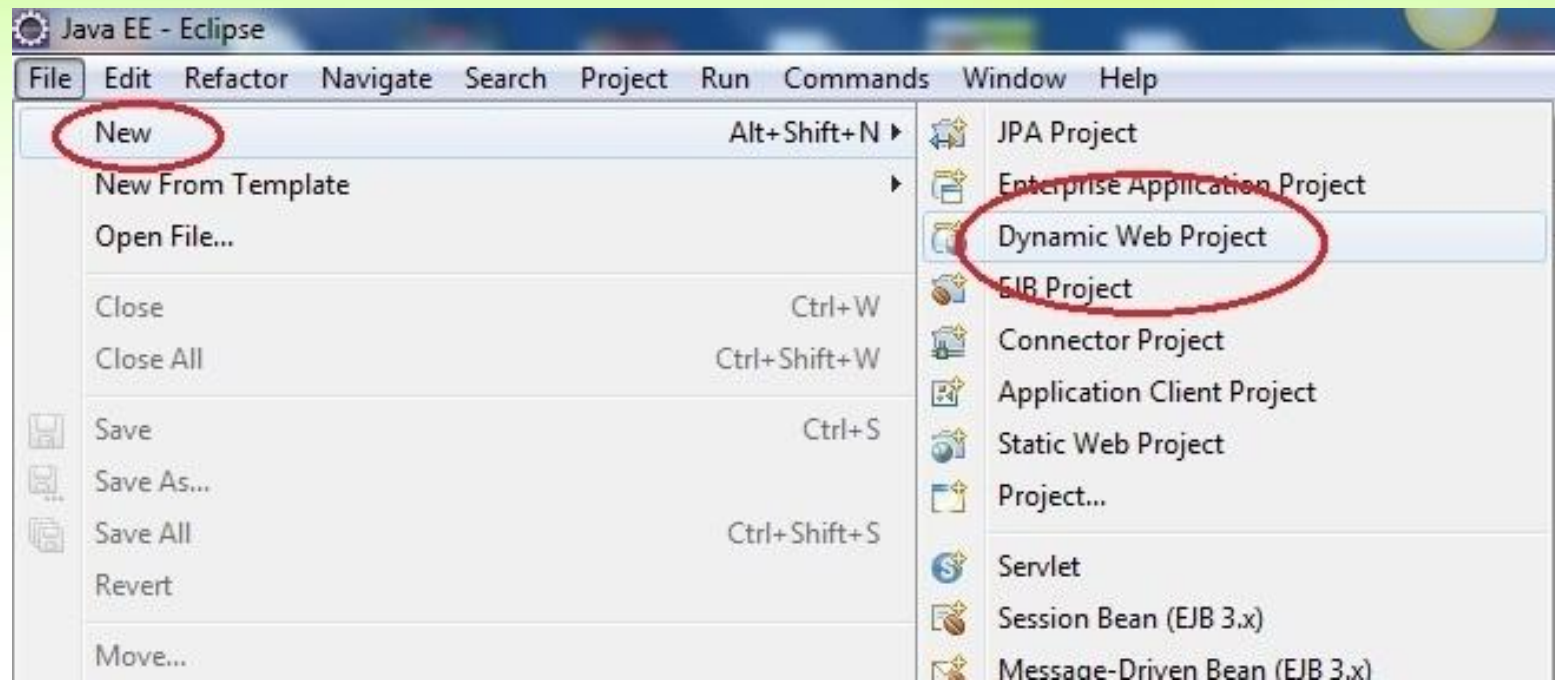
❖ JSP lifecycle methods are:

1. **jspInit()** declared in JspPage interface. This method is called only once in JSP lifecycle to initialize config params.
2. **_jspService(HttpServletRequest request, HttpServletResponse response)** declared in HttpJspPage interface and response for handling client requests.
3. **jspDestroy()** declared in JspPage interface to unload the JSP from memory.

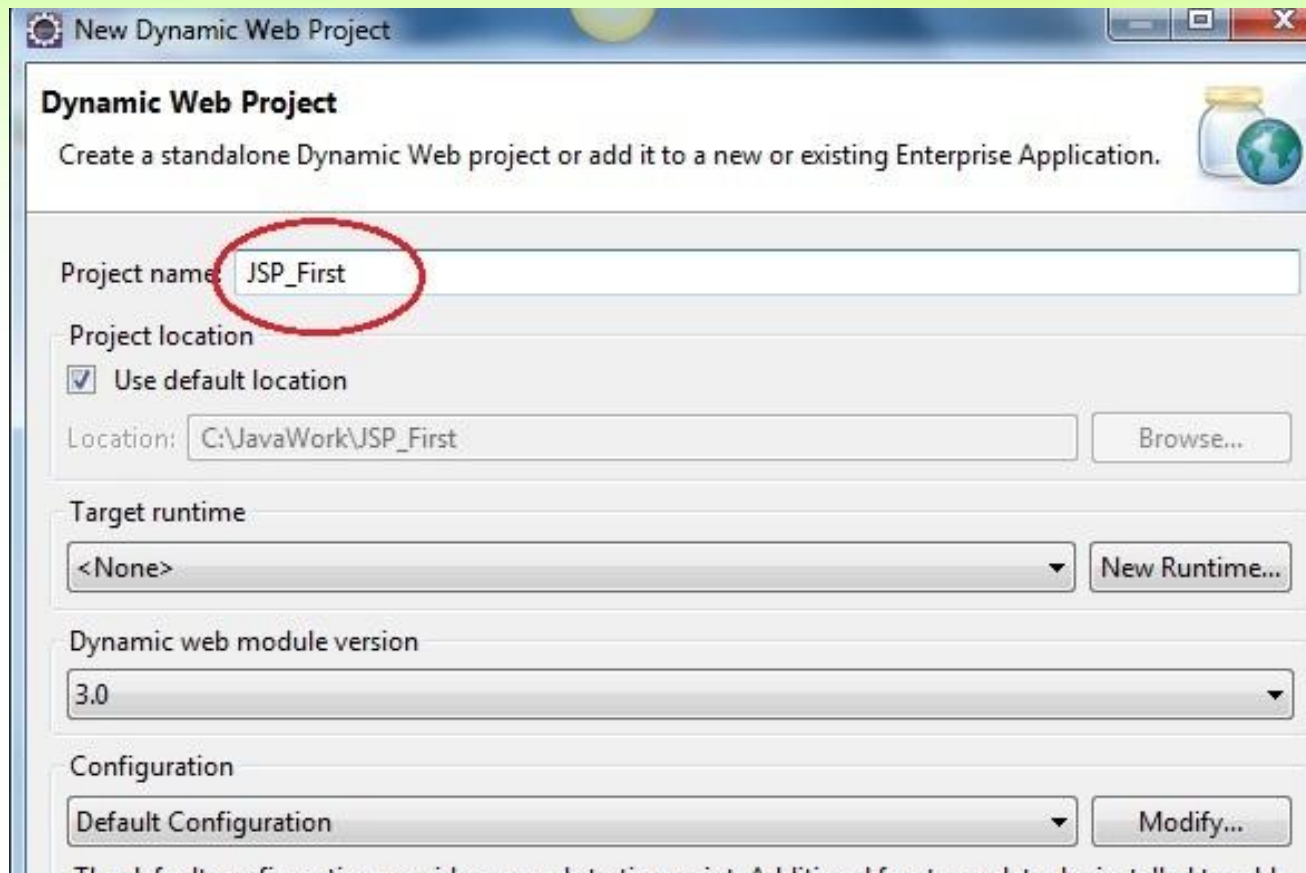
CREATING A JSP PAGE

- ❖ 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 `%>`.

CREATING A JSP PAGE IN ECLIPSE



CREATING A JSP PAGE IN ECLIPSE



New Dynamic Web Project

Create a standalone Dynamic Web project or add it to a new or existing Enterprise Application.

Dynamic Web Project

Project name: JSP_First

Project location

☒ Use default location

Location: C:\JavaWork\JSP_First Browse...

Target runtime

<None> New Runtime...

Dynamic web module version

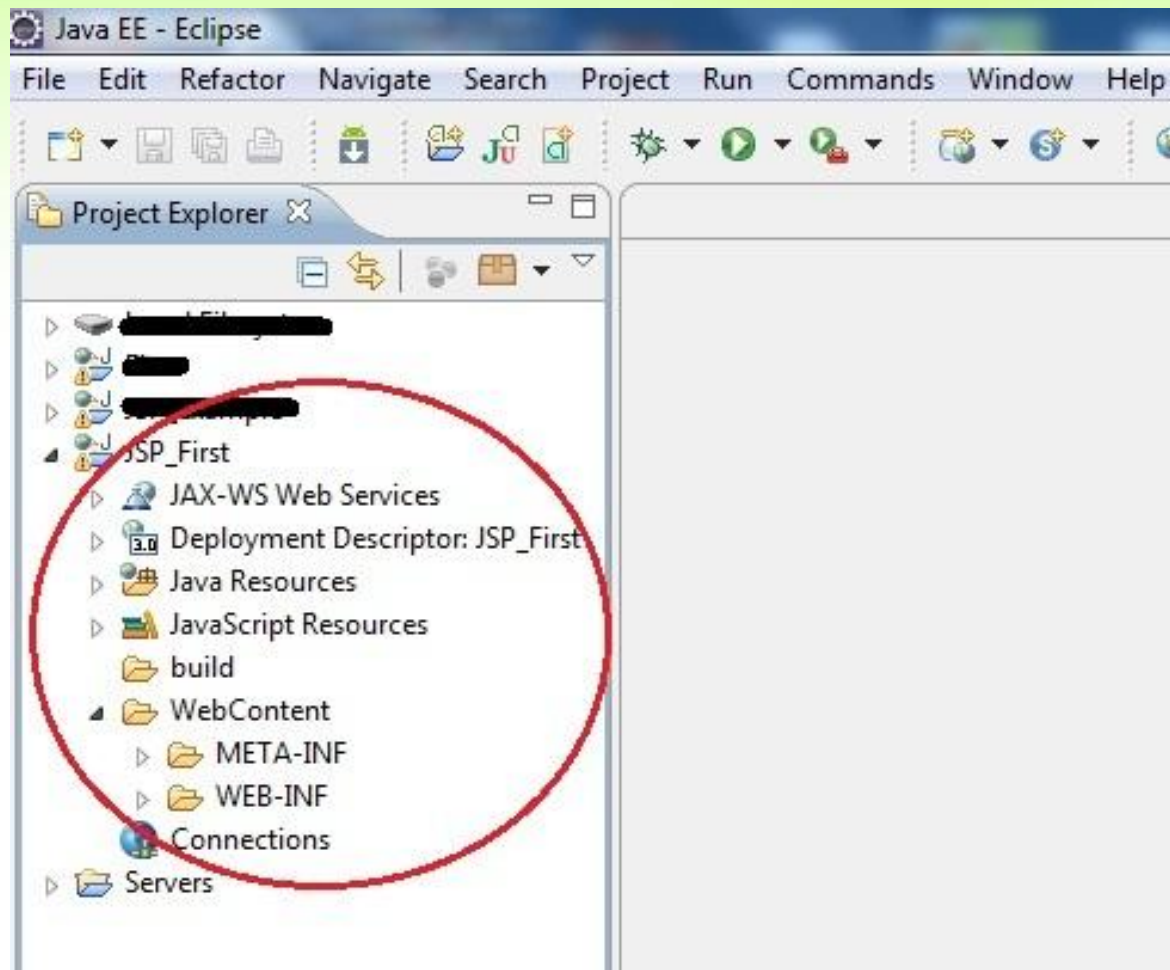
3.0

Configuration

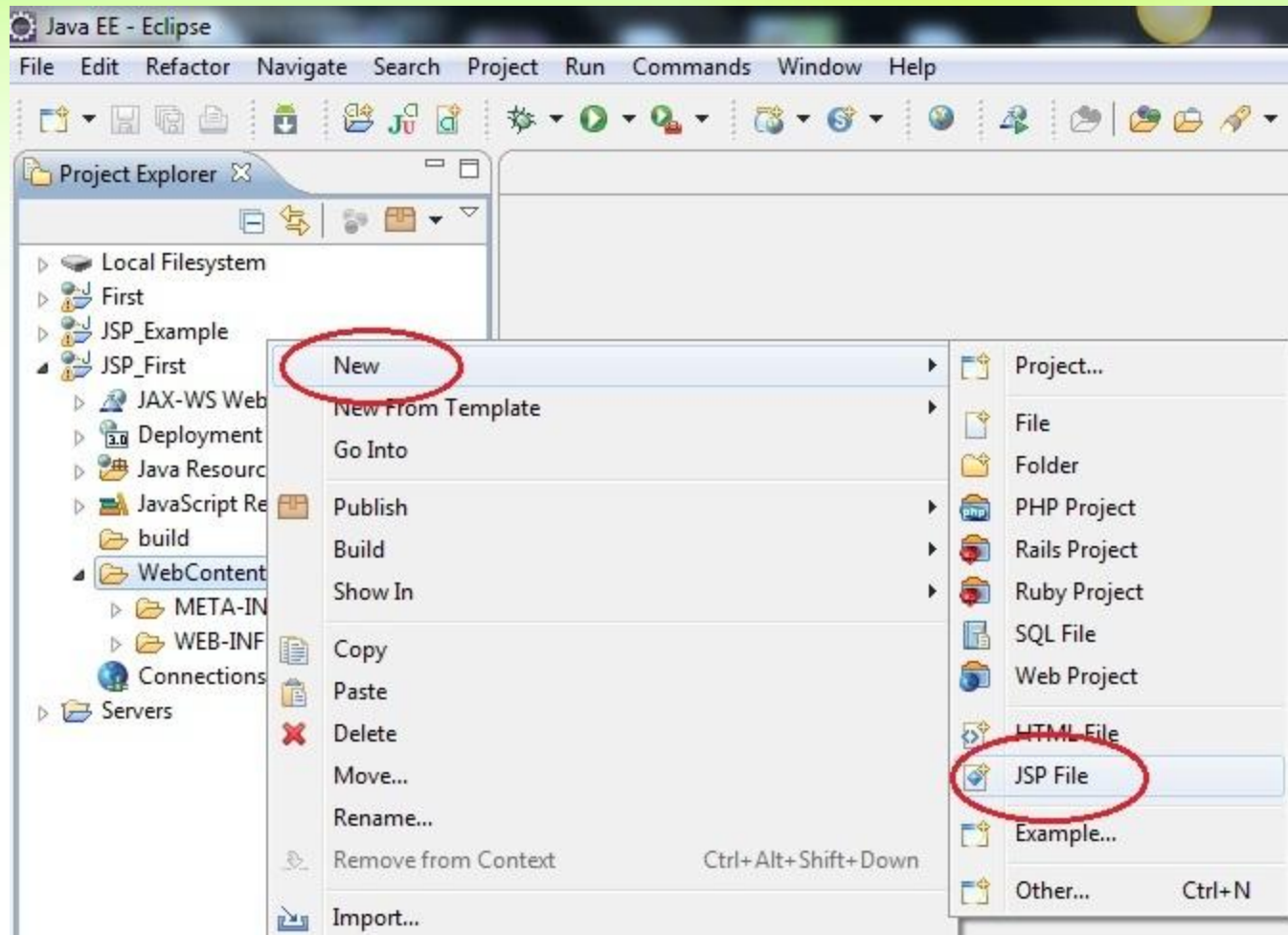
Default Configuration Modify...

The default configuration provides a good starting point. Additional frameworks can later be installed to add

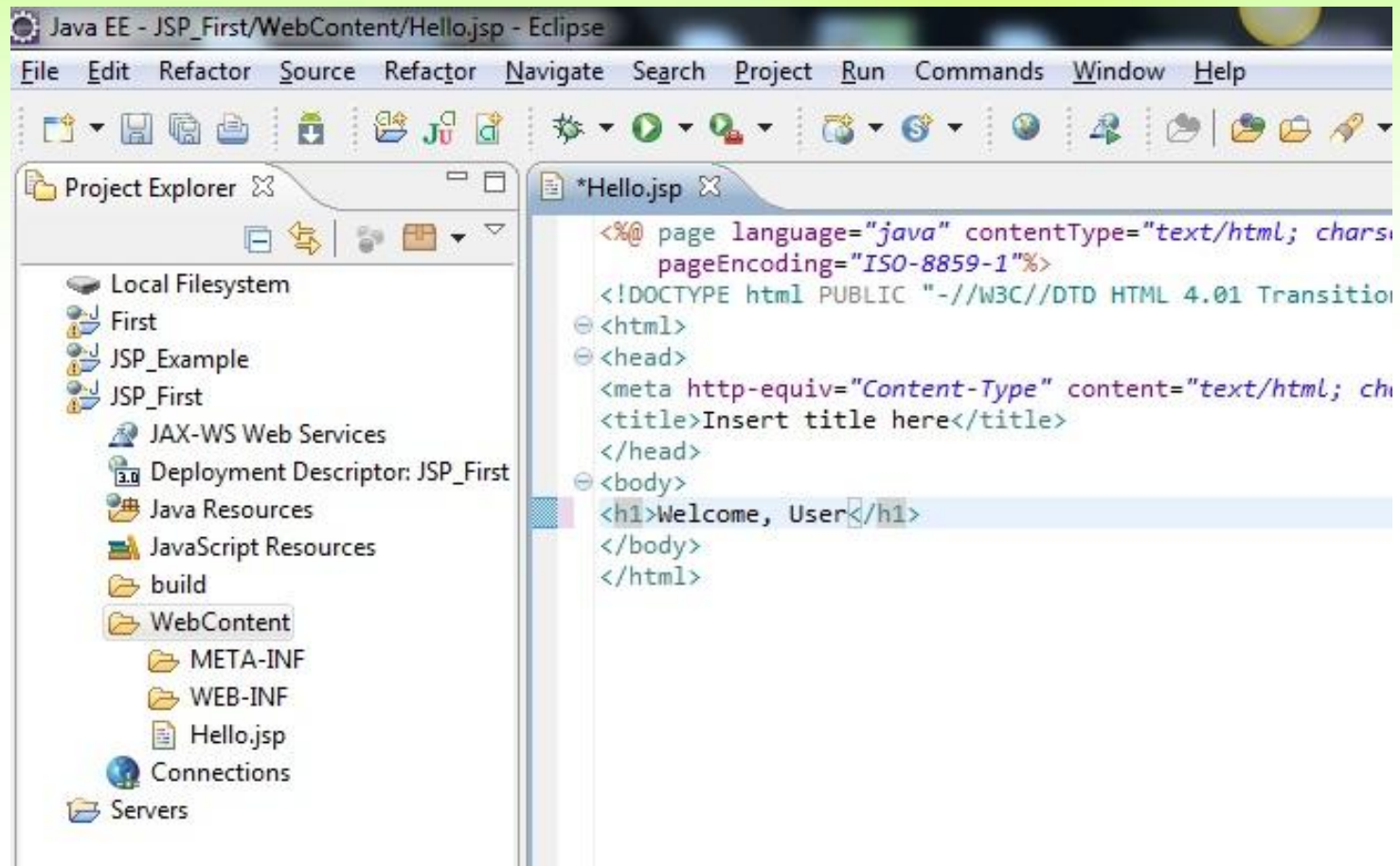
CREATING A JSP PAGE IN ECLIPSE



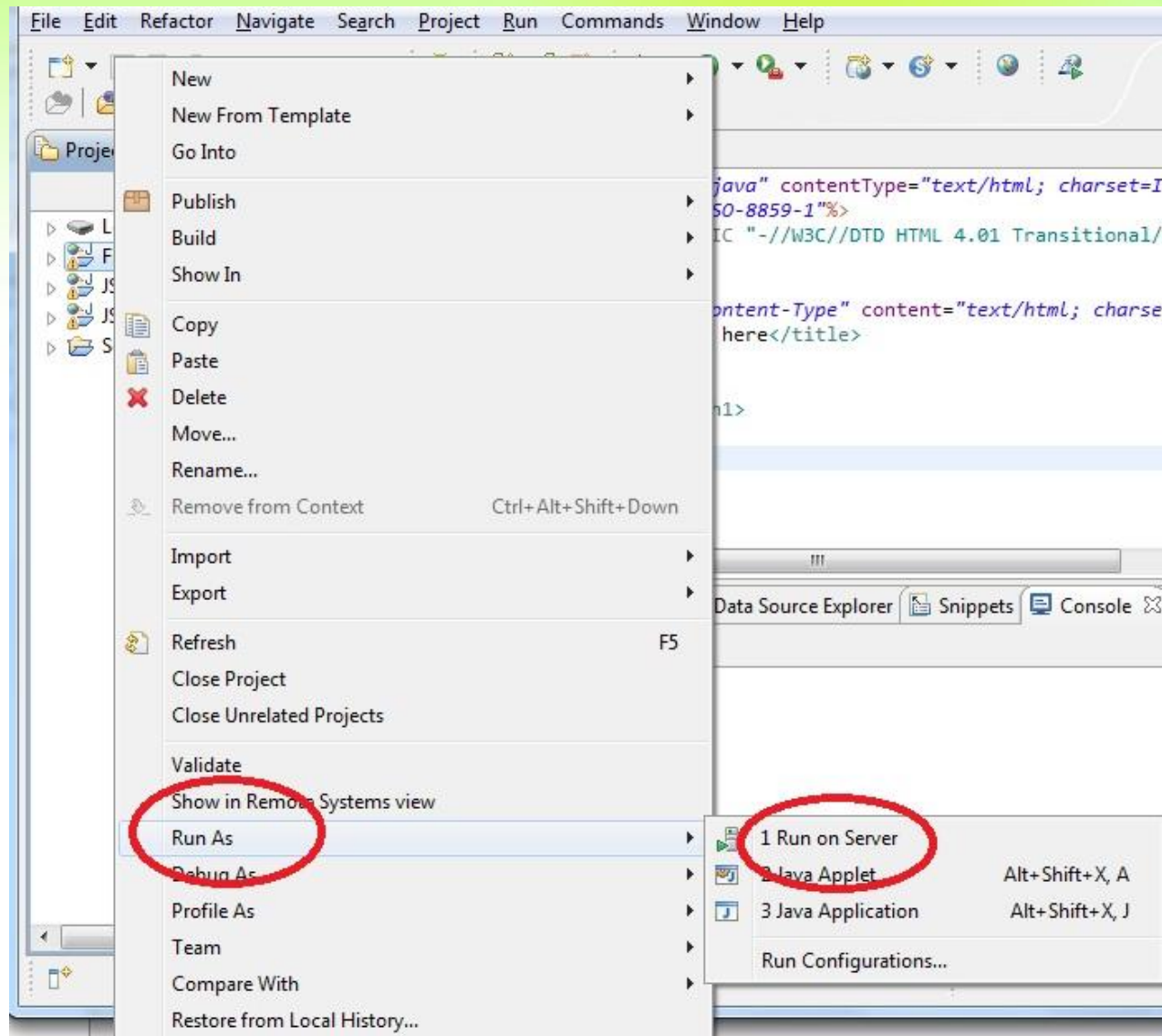
CREATING A JSP PAGE IN ECLIPSE



CREATING A JSP PAGE IN ECLIPSE



CREATING A JSP PAGE IN ECLIPSE



CREATING A JSP PAGE IN ECLIPSE



JSP SCRIPTING ELEMENT

- ❖ JSP Scripting elements are written inside `<% %>` tags.
- ❖ These codes inside `<% %>` tags are processed by the JSP engine during translation of the JSP page.
- ❖ Any other text in the JSP page is considered as HTML code or plain text.

```
<html>
  <head>
    <title>My First JSP Page</title>
  </head>
  <%
    int count = 0;
  %>
  <body>
    Page Count is <% out.println(++count); %>
  </body>
```

DIFFERENT TYPES OF SCRIPTING ELEMENTS

Scripting Element	Example
Comment	<code><%-- comment --%></code>
Directive	<code><%@ directive %></code>
Declaration	<code><%! declarations %></code>
Scriptlet	<code><% scriptlets %></code>
Expression	<code><%= expression %></code>

JSP COMMENT

❖ JSP Comment is used when you are creating a JSP page and want to put in comments about what you are doing. JSP comments are only seen in the JSP page. These comments are not included in servlet source code during translation phase, nor they appear in the HTTP response. Syntax of JSP comment is as follows :

```
<%-- JSP comment --%>
```

SCRIPTLET TAG

❖ Scriptlet Tag allows you to write java code inside JSP page. Scriptlet tag implements the `_jspService` method functionality by writing script/java code. Syntax of Scriptlet Tag is as follows :

```
<% java code %>
```

```
<html>
  <head>
    <title>My First JSP Page</title>
  </head>
  <%
    int count = 0;
  %>
  <body>
    Page Count is <% out.println(++count); %>
  </body>
</html>
```

DECLARATION TAG

- ❖ We know that at the end a JSP page is translated into Servlet class.
- ❖ So when we declare a variable or method in JSP inside Declaration Tag, it means the declaration is made inside the Servlet class but outside the service(or any other) method.
- ❖ You can declare static member, instance variable and methods inside Declaration Tag.

```
<html>
  <head>
    <title>My First JSP Page</title>
  </head>
  <%!
    int count = 0;
  %>
  <body>
    Page Count is:
    <% out.println(++count); %>
  </body>
</html>
```

DECLARATION TAG

❖ In the previous code, we have used the declaration tag to declare variable count. The previous JSP page becomes this Servlet :

```
public class hello_jsp extends HttpServlet
{
    int count=0;
    public void _jspService(HttpServletRequest request, HttpServletResponse response)
        throws IOException,ServletException
    {
        PrintWriter out = response.getWriter();
        response.setContentType("text/html");
        out.write("<html><body>");

        out.write("Page count is:");
        out.print(++count);
        out.write("</body></html>");
    }
}
```


DECLARATION TAG

If you want to include any method in your JSP file, then you must use the declaration tag, because during translation phase of JSP, methods and variables inside the declaration tag, becomes instance methods and instance variables and are also assigned default values.

```
<html>
  <head>
    <title>My First JSP Page</title>
  </head>
  <%!
    int count = 0;
    int getCount() {
      System.out.println( "In getCount() method" );
      return count;
    }
  %>
  <body>
    Page Count is:
    <% out.println(getCount()); %>
  </body>
</html>
```

DECLARATION TAG

```
public class hello_jsp extends HttpServlet
{
    int count = 0;
    int getCount() {
        System.out.println( "In getCount() method" );
        return count;
    }
    public void _jspService(HttpServletRequest request, HttpServletResponse response)
        throws IOException, ServletException
    {
        PrintWriter out = response.getWriter();
        response.setContentType("text/html");
        out.write("<html><body>");

        out.write("Page count is:");
        out.print(getCount());
        out.write("</body></html>");
    }
}
```

DIRECTIVE TAG

Directive Tag gives special instruction to Web Container at the time of page translation. Directive tags are of three types: page, include and taglib.

<%@ page ... %>	defines page dependent properties such as language, session, errorPage etc.
<%@ include ... %>	defines file to be included.
<%@ taglib ... %>	declares tag library used in the page

PAGE DIRECTIVE TAG

The Page directive defines a number of page dependent properties which communicates with the Web Container at the time of translation.

Basic syntax of using the page directive is

<%@ page attribute="value" %> where attributes can be one of the following :

import attribute

extends attribute

isThreadSafe attribute

errorPage attribute

autoFlush attribute

language attribute

session attribute

isErrorPage attribute

contentType attribute

buffer attribute

PAGE DIRECTIVE TAG

❖ The import attribute defines the set of classes and packages that must be imported in servlet class definition. For example

```
<%@ page import="java.util.Date" %>  
or  
<%@ page import="java.util.Date,java.net.*" %>
```

❖ language attribute defines scripting language to be used in the page.

❖ extends attribute defines the class name of the superclass of the servlet class that is generated from the JSP page.

❖ session attribute defines whether the JSP page is participating in an HTTP session. The value is either true or false.

PAGE DIRECTIVE TAG

- ❖ `isThreadSafe` attribute declares whether the JSP is thread-safe. The value is either true or false.
- ❖ `isErrorPage` attribute declares whether the current JSP Page represents another JSP's error page.
- ❖ `errorPage` attribute indicates another JSP page that will handle all the run time exceptions thrown by current JSP page. It specifies the URL path of another page to which a request is to be dispatched to handle run time exceptions thrown by current JSP page.
- ❖ `contentType` attribute defines the MIME type for the JSP response.

PAGE DIRECTIVE TAG

- ❖ autoFlush attribute defines whether the buffered output is flushed automatically. The default value is "true".
- ❖ buffer attribute defines how buffering is handled by the implicit out object.

INCLUDE DIRECTIVE TAG

- ❖ JSP "include directive is used to include one file to the another file
- ❖ This included file can be HTML, JSP, text files, etc.
- ❖ It is also useful in creating templates with the user views and break the pages into header & footer and sidebar actions.
- ❖ It includes file during translation phase. Syntax of include directive is:

```
<%@ include file="filename.jsp" %>
```

```
<html>
<body>
<%@ include file="header.jsp" %>
<br>
Contact Us at: we@studytonight.com
<br/>
<%@ include file="footer.jsp" %>
</body>
</html>
```

This says insert the complete content of **header.jsp** into this JSP page

This says insert the complete content of **footer.jsp** into this JSP page

INCLUDE DIRECTIVE TAG

```
<html>
  <head>
    <title>Welcome Page</title>
  </head>

  <body>
    <%@ include file="header.jsp" %>
    Welcome, User
  </body>
</html>
```

```
<html>
  <body>
    
  </body>
</html>
```

EXPRESSION TAG

❖ Expression Tag is used to print out java language expression that is put between the tags. An expression tag can hold any java language expression that can be used as an argument to the out.print() method.

Syntax of Expression Tag

```
<%= JavaExpression %>
```

❖ When the Container sees this

```
<%= (2*5) %>
```

❖ It turns it into this:

```
out.print((2*5));
```