CS –29 MAJOR - 11
Advance Java Programming(J2EE)

Presented by : Dhruvita Savaliya

UNIT 3- JSP

Topics of JSP

- Introduction to JSP and JSP Basics
- JSP vs. Servlet
- JSP Architecture
- Life cycle of JSP
- JSP Elements:
  - Directives Elements (page, include, taglib)
  - Scripting Elements (Declaration, scriptlet, expression)
  - Action Elements (jsp:param, jsp:include, jsp:Forward, jsp:plugin, jsp:useBean, jsp:setAttribute, jsp:getAttribute)
- JSP Implicit Objects (request, response, out, session, application, pagecontext)
- JSP Scope
- Including and Forwarding from JSP Pages
  - include Action
  - forward Action
- Working with Session & Cookie in JSP
- Error Handling and Exception Handling with JSP
- JSP EL (Expression Language), JSP Standard Tag Libraries (JSTL)

Introduction :

- JSP is Java Server Pages (JSP) technology enables you to mix regular, static HTML with dynamically generated content.
- JSP are run in a server-side component known as JSP container which translates them into equivalent Java Servlet.
- Simply write the regular HTML in the normal manner, using familiar Web-page-building tools.
- Then enclose the code for the dynamic parts in special tags, most of which start with <% and end with %>.

➢ **JSP typically comprised of:**

- Static HTML/XML components
- Special JSP tags.
- Code written in the java language called" script less".

- **How is JSP More Advantage than Servlets?**
- JSP simplifies web development by combining the strengths of Java with the flexibility of HTML. Some advantages of JSP over Servlets are listed below:
- JSP code is easier to manage than Servlets as it separates UI and business logic.
- JSP minimizes the amount of code required for web applications.
- Generate content dynamically in response to user interactions.
- It provides access to the complete range of Java APIs for robust application development.
- JSP is suitable for applications with growing user bases.

- **Key Features of JSP**
- It is platform-independent; we can write once, run anywhere.
- It simplifies database interactions for dynamic content.
- It contains predefined objects like request, response, session and application, reducing development time.
- It has built-in mechanisms for exception and error management.
- It supports custom tags and tag libraries.

Comparison of JSP :

1. **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.
2. **JSP 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.
3. **JSP 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.
4. **JSP 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.
5. **JSP vs. Static HTML :** Regular HTML, of course, cannot contain dynamic information.

- **Advantages of JSP :**
- **Nobody can borrow the code :**
- The JSP code written and runs and remains on the server. So, issue of copy source code does not arise at all. All of JSP's functionality is handled before the page is sent to browser.
- **Faster loading of pages :**
- With JSP, decision can be made about what user want to see at web server prior to pages being dispatched. So only the content that the user is interested will be dispatched to the user. There is no extra code and extra content.
- **No browser compatibility Issues :**
- JSP pages can run same way in browser. The developer ends up sending standard HTML to a user browser. This largely eliminates scripting issues and cross browser compatibility.
- **JSP support :**
- JSP is supported by number of web server like Apache, Microsoft IIS and PWS, Netscape's Fast Tracks and Enterprise web server and others. Built in support for JSP is available Java Server from Sun Micro system.

- **Compilation :**
- JSP compiled before the web server processes it. This allows the server to handle JSP pages much faster, because in the older technologies such as CGI require the server to load an interpreter and the target script each time the page is requested.
- **JSP elements in HTML/XML pages :**
- JSP page look like HTML /XML page, it holds text marked with a collection of tags. While a regular JSP page is not a valid XML page, there is a variant JSP tag syntax that lets the developer use JSP tags within XML documents.

➢ Disadvantages of JSP :

- **Attractive Java Code**
- Putting java code within web page is really bad design, but JSP makes it tempting to do just that. Avoid this as far as possible. It is done using template using.
- **Java Code required**
- To relatively simple things in JSP can actually demand putting java code in a page.
- **Simple task are hard to code**
- Even including page headers and footers is a bit difficult with JSP.
- **Difficult looping in JSP**
- In regular JSP pages looping is difficult. In advance JSP we can use some custom tags for looping.
- **Occupies a lot of space.**
- JSP consumes extra hard drive and memory space.

Difference Between Servlet & JSP :

- **Servlet** technology is used to create a web application. A **servlet** is a Java class that is used to extend the capabilities of servers that host applications accessed by means of a request-response model. Servlets are mainly used to extend the applications hosted by web services.

- **JSP** is used to create web applications just like **Servlet** technology. A **JSP** is a text document that contains two types of text: static data and dynamic data. The static data can be expressed in any text-based format (like HTML, XML, SVG, and WML), and the dynamic content can be expressed by JSP elements.

- **When to Use Servlet or JSP**
- **Servlet:** When you need to handle business logic, form processing, and request management.
- **JSP:** When you need to build dynamic web pages with UI elements and embed minimal Java logic.

Servlet	JSP
Servlet is a Java code.	JSP is an HTML-based compilation code.
Writing code for servlet is harder than JSP as it is HTML in java.	JSP is easy to code as it is java in HTML.
Servlet plays a controller role in the ,MVC approach.	JSP is the view in the MVC approach for showing output.
Servlet is faster than JSP.	JSP is slower than Servlet because the first step in the JSP lifecycle is the translation of JSP to java code and then compile.
Servlet can accept all protocol requests.	JSP only accepts HTTP requests.
In Servlet, we can override the service() method.	In JSP, we cannot override its service() method.
In Servlet by default session management is not enabled, user have to enable it explicitly.	In JSP session management is automatically enabled.
In Servlet we have to implement everything like business logic and presentation logic in just one servlet file.	In JSP business logic is separated from presentation logic by using JavaBeansclient-side.

Servlet	JSP
Modification in Servlet is a time-consuming compiling task because it includes reloading, recompiling, JavaBeans and restarting the server.	JSP modification is fast, just need to click the refresh button.
There is no method for running JavaScript on the client side in Servlet.	While running the JavaScript at the client side in JSP, client-side validation is used.
Packages are to be imported on the top of the program.	Packages can be imported into the JSP program (i.e, bottom , middleclient-side, or top)
It can handle extensive data processing.	It cannot handle extensive data processing very efficiently.
The facility of writing custom tags is not present.	The facility of writing custom tags is present.
Servlets are hosted and executed on Web Servers.	Before the execution, JSP is compiled in Java Servlets and then it has a similar lifecycle as Servlets.
It does not have inbuilt implicit objects.	In JSP there are inbuilt implicit objects.

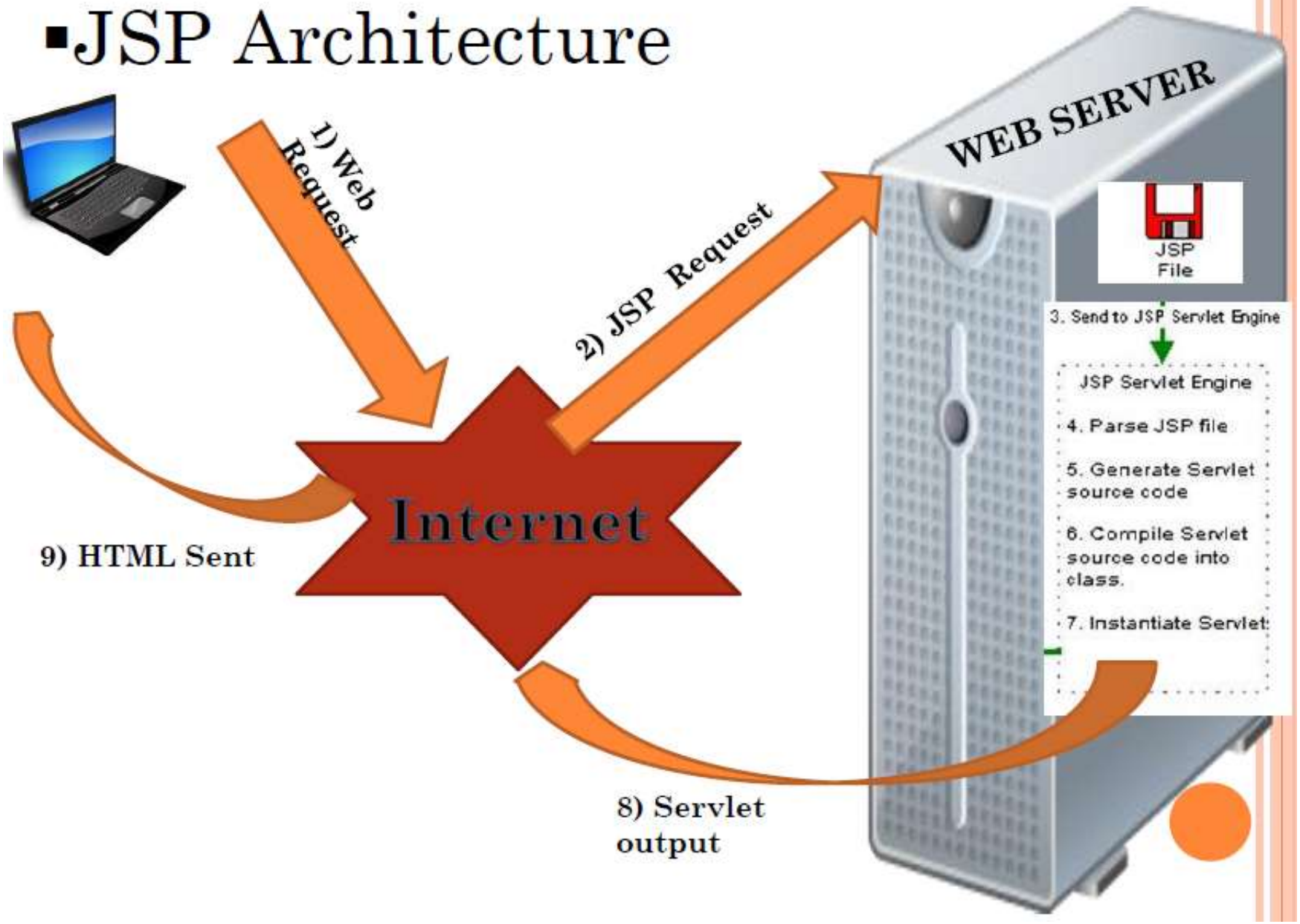
JSP Architecture :

- JSP (JavaServer Pages) architecture is a framework for building web applications in Java.
- It has three main parts: a web container, a JSP engine, and a servlet container.
- The JSP engine processes requests from the user and generates dynamic content using Java code and JSP tags.
- The servlet container manages the lifecycle of servlets and JSP pages.
- JSP are built by SUN Micro systems Servlet technology.
- JSP tag contains java code and its file extension is JSP.

- **Steps for JSP Request:**

1. User requesting a JSP page through internet via web browser.
2. The JSP request is sent to the WebServer.
3. Webserver accepts the requested. Jsp file and passes the JSP file to the JSP Servlet Engine.
4. If the JSP file has been called the first time then the JSP file is parsed other wise servlet is instantiated.
5. The next step is to generate a servlet from the JSP file. In that servlet file, all the HTML code is converted in println statements.
6. Now, The servlet source code file is compiled into a class file (byte code file)
7. The servlet is instantiated by calling the init and service methods of the servlet's life cycle.
8. Now, the generated servlet output is sent via the Internet form webserver to user's web browser.
9. Now in last step, HTML results are displayed on the user's web browser.

JSP Architecture

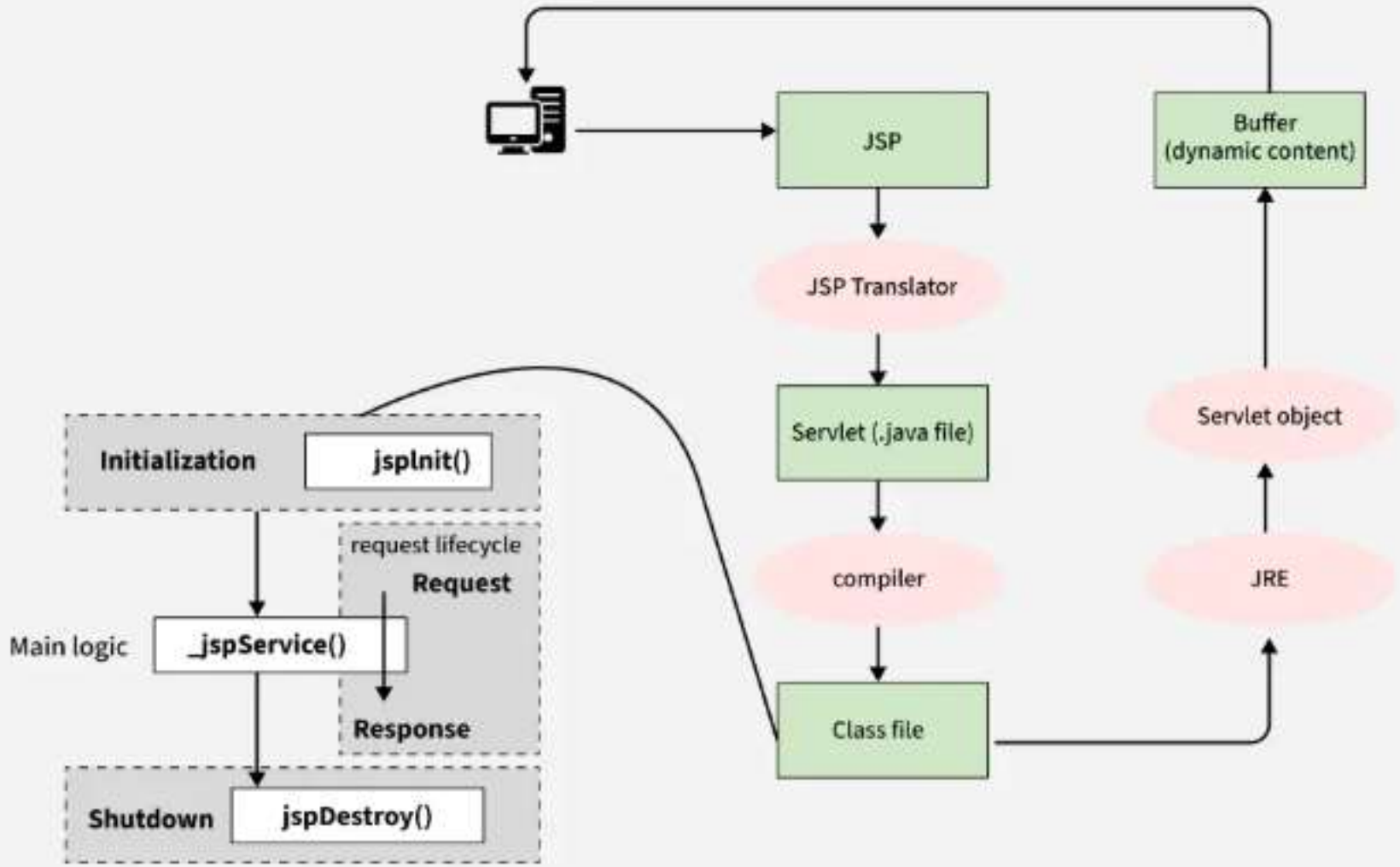


JSP Life Cycle :

- When a browser asks for a JSP, the JSP engine first checks to see whether it needs to compile the page. If the page has never been compiled, or if the JSP has been modified since it was last compiled, the JSP engine compiles the page.
- The compilation process involves three steps:

1. Parsing the JSP.
2. Turning the JSP into a servlet.
3. Compiling the servlet.

Life Cycle of JSP



- **Steps of JSP Life Cycle**

1. Translation of JSP page to Servlet
2. Compilation of JSP page(Compilation of JSP into test.java)
3. Classloading (test.java to test.class)
4. Instantiation(Object of the generated Servlet is created)
5. Initialization(jspInit() method is invoked by the container)
6. Request processing(_jspService()is invoked by the container)
7. JSP Cleanup (jspDestroy() method is invoked by the container)

- We **can** override jspInit(), jspDestroy() but we **can't** override _jspService() method.

- **1. Translation of JSP to Servlet**

  The JSP file is parsed and converted into a Java servlet source file (test.java).

  This step checks for syntax correctness.

  ```
  // The JSP is converted to a servlet class.
  public class TestServlet extends HttpServlet {
          // The generated servlet code
  }
  ```

- **2. Compilation of JSP page**

  The generated test.java file is compiled into a class file (test.class).

  This step converts the servlet code into bytecode.

  // JSP is automatically converted into a servlet, such as **public class TestServlet extends** HttpServlet { *// Generated servlet code here*

  ```
  }
  ```

- **3. Classloading**

  The container dynamically loads the compiled class.

- **4. Instantiation :**

The container creates an instance of the generated servlet class.

This instance handles multiple requests unless explicitly configured otherwise.

```
// The container creates an instance automatically.
TestServlet servlet = new TestServlet();
```

5. Initialization (jspInit()) :

This method is called only once when the JSP is first loaded.

It is used for initializing resources like database connections or configurations.

```
public void jspInit() {       // Initialization code, like setting up resources.
        System.out.println("JSP Initialized.");
}
```

- **6. Request Processing (_jspService()) :**

This method is called for every request.

It cannot be overridden because it is auto-generated and declared as final.

It receives HttpServletRequest and HttpServletResponse objects to handle the request.

```
public void _jspService(HttpServletRequest request, HttpServletResponse response) { // Code that handles the request, like generating HTML output.
        response.getWriter().write("<html><body>Hello,
                                        World!</body></html>");
}
```

7. JSP Cleanup (jspDestroy())

- This method is called once before removing the JSP from service.
- It is used for releasing resources, such as closing database connections or cleaning up open files.

```
public void jspDestroy() {
// Clean up resources like closing database connections.
System.out.println("JSP Destroyed.");
}
```

- This Lifecycle ensures that JSP pages are efficiently compiled, managed and cleaned up by the server container.

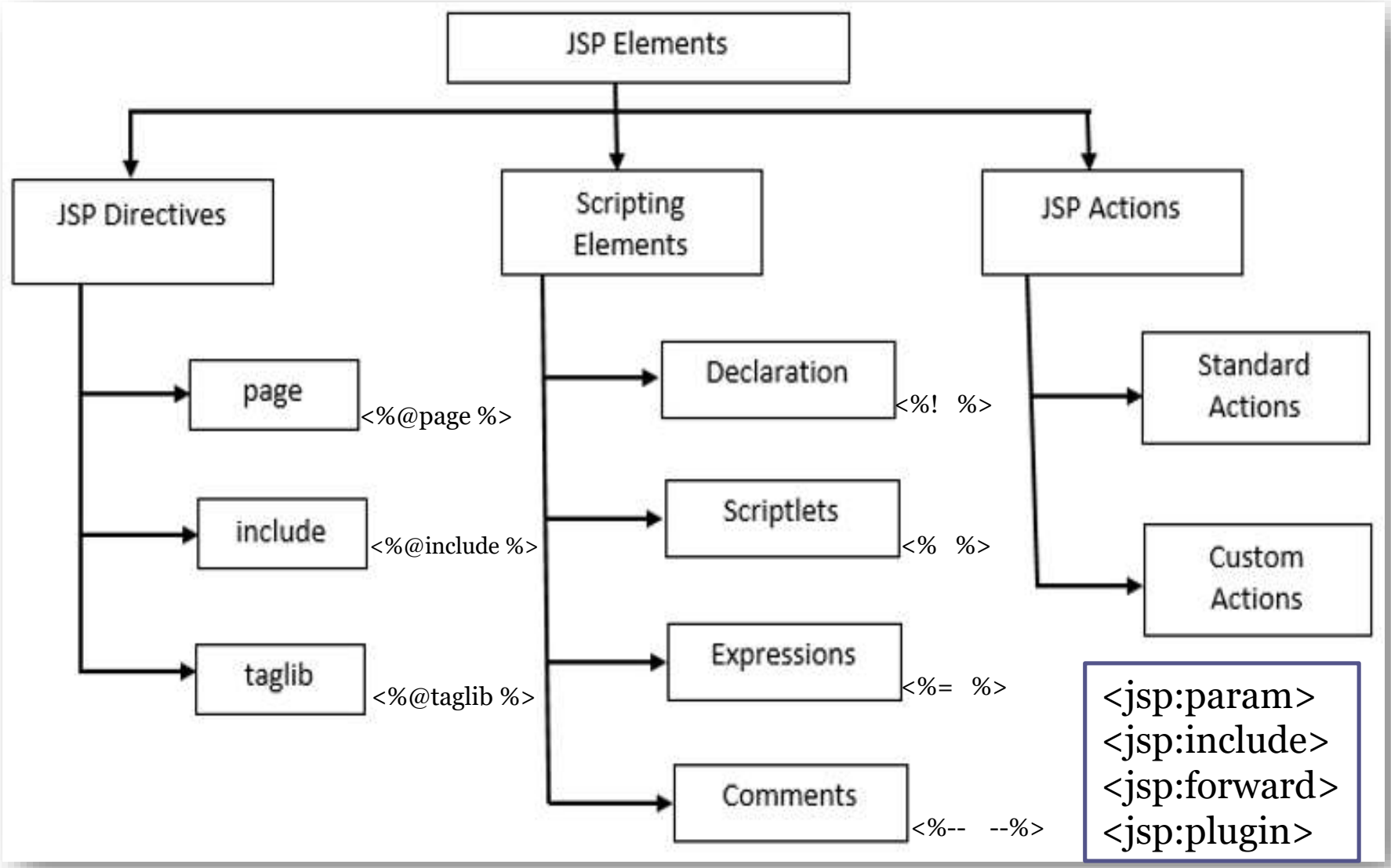
JSP Elements :

- JSP Element have a special identity to JSP compiler because it starts and ends with special kind of tags.
- Template (HTML) code is not compiled by the JSP compiler and also not recognized by the JSP container.
- It is known as **Component of JSPpages.**
- There are 3 basic JSP elements.

    1.Directive Element

    2.Scripting Element

    3.Action Element


JSP Elements
JSP Directives
Scripting Elements
JSP Actions
page
<%@page %>
include
<%@include %>
taglib
<%@taglib %>
Declaration
<%! %>
Scriptlets
<% %>
Expressions
<%= %>
Comments
<%-- --%>
Standard Actions
Custom Actions
<jsp:param>
<jsp:include>
<jsp:forward>
<jsp:plugin>

- 1.**Directive Element:**
- The role of the directives is to pass information to the JSP container.
- Following is the general syntax of Directive element.
- Syntax :-<%@ page attibute= “value” %>
- There are three types of directives

  1.1.page directive

  1.2.include directive

  1.3.taglib directive

- **1.1. Page Directive:**
  Page directive is used to specify attributes for the whole JSP page.
- **Syntax:<%@ page attibute= “value” %>**
  Attributes of JSP page directive are as under :

1. extends=“class_name”
2. import=“import_list_package”
3. contentType=“content_info”
4. session=“true|false”
5. buffer=“auto flush”
6. autoFlush=“true|false”
7. IsThreadSafe=“true|false”
8. info=“Information”
9. errorPage=“error_file”
10. language=“scripting language”
11. IsErrorPage“true|false”
12. pageEncoding

- **Import Page Directive :**
- The import attribute is used to import class, interface or all the members of a package.
- It is similar to import keyword in java class or interface.
- Example of import attribute:

```
<html>
   <body>
        <%@ page import="java.util.Date" %>
                Today is:
        <%= new Date() %>
   </body>
</html>
```

- **contentType Page Directive :**
- The contentTypebattribute defines the MIME(Multipurpose Internet Mail Extension) type of the HTTP response.
- The default value is "text/html;charset=ISO-8859-1".
- **Example:**

```
<BODY>
    <H2> The contentType Attribute </H2>
    <%@ page contentType="text/plain" %>
    This should be rendered in plain text,
    <B> not </B> as HTML.
</BODY>
```

- **errorPage Page Directive :**
  The errorPage attribute is used to define the error page, if exception occurs in the current page, it will be redirected to the error page.
  100/100=1 then 100/0=display error page
- **Example:**

```
<html>
    <body>
        <%@ page errorPage="myerrorpage.jsp" %>
        <%= 100/0 %>
    </body>
</html>
```

- **1.2.Include Directive:**
- The include directive is used to include the contents of any resource it may be jsp file, html file or text file.
- **Advantageof Include directive Code Reusability.**
- **Syntax :-<%@ include file="relative path" %>**
- **Example:**
- **File1 header.html**

```
<body>
    <h1>header file </h1>
</body>
```

- **File 2 JSP file :**

```
<html>
    <body>
        <%@ include file="header.html" %>
        <h1>JSP FILE CODE </h1>
    </body>
</html>
```

- **1.3 taglib Directive :**
- The JSP taglib directive is used to define a tag library that defines many tags.
- We use the TLD (Tag Library Descriptor) file to define the tags.
- In the custom tag section we will use this tag so it will be better to learn it in custom tag.
- **Syntax :**

  <%@ taglib uri="URI" prefix="pr" %>

  <pr:h1>

- **2.1.scriptlet tag:**
- A scriptlet tag is used to execute java source code in JSP.
- You can Declare a variable, use looping, statements use if-else, use switch-case etc in scriptlet tag.
- **Syntax:**

<%java source code%>

```
<html>
   <body>
      <%
        int a=8;
    out.print(“java code here”+a);
    out.print "Welcome"+request.getParameter(“txt_nm"));
      %>
   </body>
</html>
```

- **2.2.Expression tag:**
- The code placed within **JSP expression tag is *written to the output stream of the response.***
- So you need not write out.print() to write data. It is mainly used to print the values of variable or method.
- **Syntax:**
  <%=statement%>

Example :

```
<html>
   <body>
      <%= “Expression here ” %>
       <%= "Welcome "+request.getParameter(“txt_name")
   %>
   </body>
</html>
```

- **2.3.Declaration Tag:**
- The **JSP declaration tag is used *to declare fields and methods*.**
- The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet.
- So it doesn't get memory at each request.
- **Syntax:**
  <%! field or method declaration %>
- **Example :**

```
<html>
<body>
    <%!
      int method()
      {
              return 10+20;
      }
    %>
    <%= method() %>
</body>
</html>
```

- **Difference between JSP Scriptlet tag and Declaration tag.**

Jsp Scriplet Tag	Jsp Declaration Tag
The jsp scriptlet tag can only declare variables not methods.	The jsp declaration tag can declare variables as well as methods.
The declaration of scriptlet tag is placed inside the _jspService() method.	The declaration of jsp declaration tag is placed outside the _jspService() method.

Action Elements :

- Action element are high level jsp element which are used to create ,modify and use other objects.
- Syntax of action element's tags in jsp tags is just like XML syntax:
- There are basic 4 types of action Element:

1. <jsp:param>
2. <jsp:include>
3. <jsp:forward>
4. <jsp:plugin>
5. <jsp:useBean>
6. <jsp:setAttribute>
7. <jsp:getAttribute>

- **3.1.<jsp:param>**
- The <jsp:param> action is used with other tags to provide additional information in the form of name value pairs.
- This action is used in conjunction with jsp:include, jsp:forwardand jsp:pluginactions.
- **Syntax :**

  <jsp:paramname=”parameter_name” value=”parameter_value” />
- **Example:**

  <jsp:paramname=”font_size” value=”20” />

- **3.2.<jsp:include> :**
- The jsp: include action tag is used to include the content of another resource it may be jsp, html or servlet.
- The jsp include action tag includes the resource at **request time** (runtime) so it is better for dynamic pages because there might be changes in future.

- Advantage of include action tag:
- **Code reusability:** We can use a page many times such as including header and footer pages in all pages. So it saves a lot of time.
- **Syntax:**
- <jsp:include page="relativeURL|<%=expression%>"/>

- **Example :**
- **first_file.jsp**

```
<jsp:include page="second_file.jsp"></jsp:include>
<body>
    <h1>hello from first</h1>
</body>
```

- **second_file.jsp**

```
<body>
    <h1>second</h1>
</body>
```

JSP include directive	JSP include action
includes resource at translation time.	includes resource at request time.
better for static pages.	better for dynamic pages.
includes the original content in the generated servlet.	calls the include method.

- **3.3.<jsp:forward>:**
- With the forward action, the current page stops processing the request and forwards the request to another page.
- Execution never returns to the calling (current) page.
- **Syntax :**
- <jsp:forwardpage= “file_name” />
- **First_page.jsp**

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

second_page.jsp

```
<%= "Hello second page" %>
```

- **Forward one page to other page with parameter First.jsp**

```
<jsp:forward page="second_page.jsp">
        <jsp:param name=“par_1" value="<%= 10+20 %>">
        </jsp:param>
 </jsp:forward>
```

second_page.jsp

```
<%= request.getParameter(" par_1 ") %>
```

- **3.4.<jsp:plugin>:**

The jsp:plugin action tag is used to embed applet in the jsp file. The jsp:plugin action tag downloads plugin at client side to execute an applet or bean.

- **Syntax:**

<jsp:plugin type="applet|bean“ code="nameOfClassFile“ codebase="directoryNameOfClassFile“> </jsp:plugin>

```
<jsp:plugin type="applet" code="NewApplet.class" codebase=".">
        <jsp:fallback>
           <h2>sorry unable to open</h2>
        </jsp:fallback>
    </jsp:plugin>
```

Right click on source package→new→other→java→applet

NewApplet.java

```
import java.awt.*;
import javax.swing.JApplet;
public class NewApplet extends JApplet {
public void paint(Graphics g)
    {
        setBackground(Color.red);
        setForeground(Color.black);
        g.drawString("tybca", 100, 100);
    }
}
```

- <body>
- hello
- </body>
- →right click on WEB-INF → new → other → web → standard devlopment descriptor(we.xml)
- Web.xml
- <error-page>
- <error-code>404</error-code>
- <location>/error_page.jsp</location>
- </error-page>

- SVG : Scalable Vector Graphics
- SSI : Server-Side Includes
- ASP : Active Server Page
- WML : Wireless Markup Language