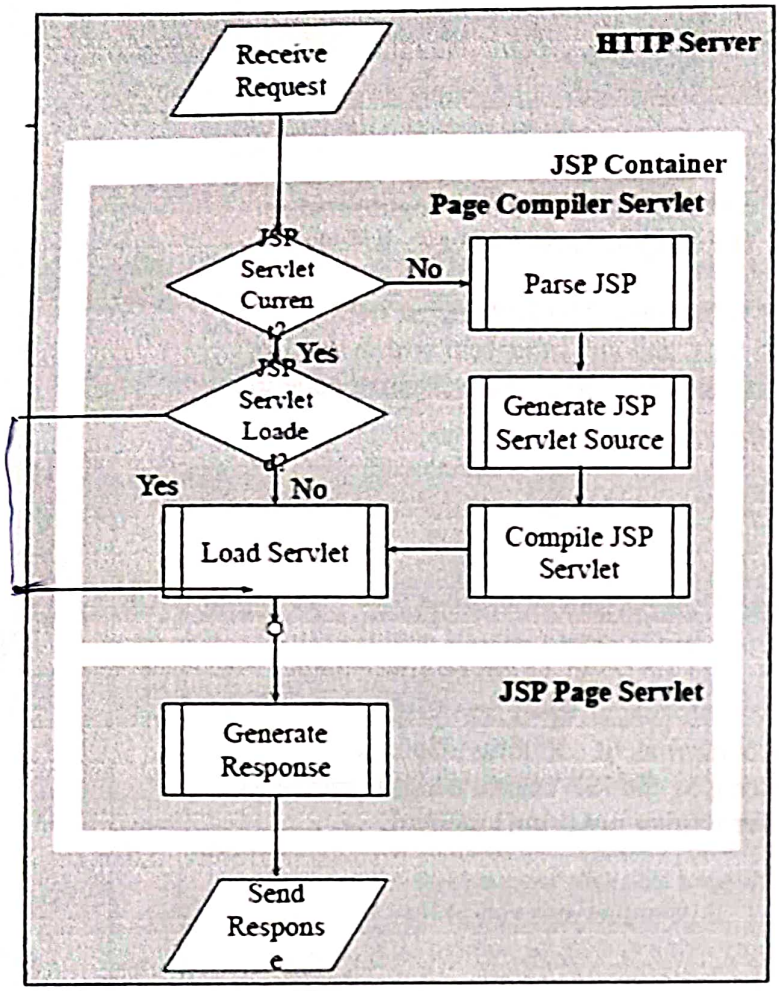
☆ JSP Architecture

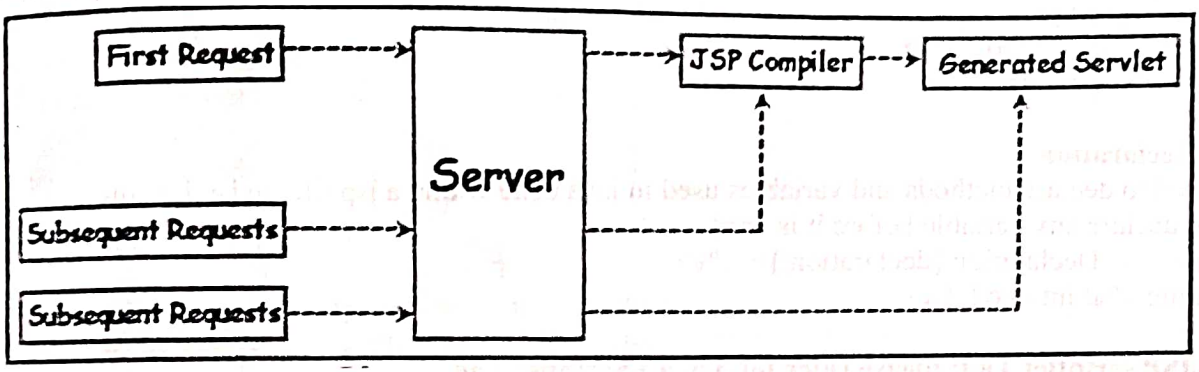


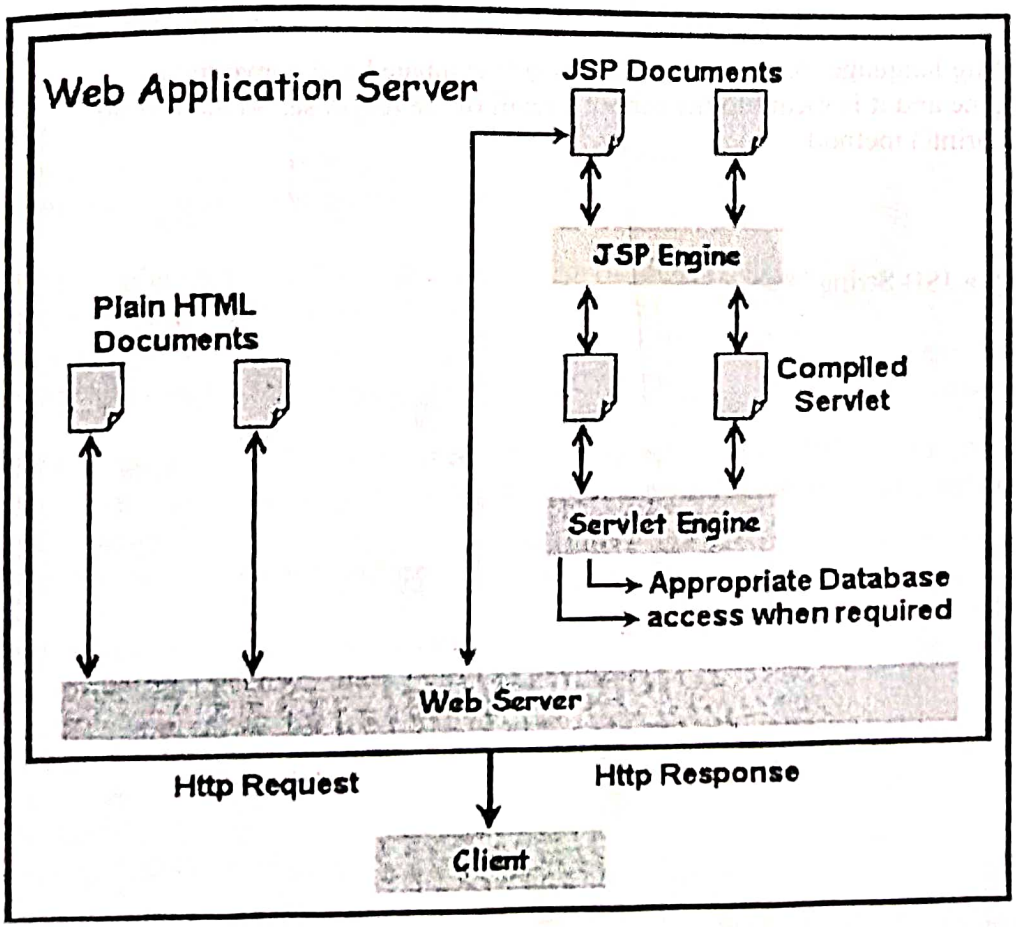
- ☐ JSPs run in two phases
  - ■ Translation Phase
  - ■ Execution Phase
- ☐ In translation phase
  - JSP page is compiled into a servlet
  - ■ called JSP Page Implementation class
- ☐ In execution phase
  - The compliled JSP is processed

☆ JSP Lifecycle

The JSP pages follow these phases:
1) Translation of JSP page.
2) Compilation of JSP page
3) Classloading
4) Instantiation
5) Initialization
6) Request Processing(web container invokes jspService() method)
7) Destroy (web container invokes jspDestroy() method)

Step 1: With the help of JSP translator the JSP page is first translated into Servlets. (Translation phase)
Step 2: The JSP translator is a part of web server which handles the translation of JSP page into Servlets.
Step 3: After that the Servlet page is compiled by the compiler and .class file is created which is binary file.
Step 4: All processes that take place in Servlets are performed same in JSL later that is initialization and sending response to the browser and then at the end invoking destroy () method. (Execution phase)





☆ Scripting Tags

3.6 Scripting Tags

There are five different Scriptlet elements in JSP are:-

1) Comments
2) Directives
3) Declaration
4) JSP scriptlet Tag
5) Expressions

3.6.1 Comments

Comments are used to write some text or statements that are ignored by the JSP compiler. It is very useful when someone wants to remember some logic or information in future.

Syntax: <%-- A JSP COMMENT--%>

3.6.2 Directives

It is used to give some specific instructions to web container when the jsp page is translated. It has three subcategories:

- Page:<%@ page...>
- Include:<%@ include...%>
- Taglib:<%@ taglib....%>

3.6.3 Declaration

It is used to declare methods and variables used in java code within a jsp file. In jsp it is the rule to declare any variable before it is used.

Syntax: <%! Declaration;[declaration;]+…%>

Example: <%! int a=62; %>

3.6.4 JSP scriptlet Tag: please refer to(3.5. a) Scriptlet Tag

3.6.5 Expressions

Its contains the scripting language expressions which is gets evaluated and converted to String by the JSP engine and it is meant to the output stream of the response. So there is no need to write the out.print() method.

Example:

```
<html>
        <body>
                <%= " a JSB String" %>
        </body>
</html>
```

☆ Implicit objects

3.7 implicit objects

There are 9 implicit objects in JSP. These objects are created by the web container and it is available to all the JSP pages.

3.7.1 out:- The out implicit object is an instance of a javax.servlet.jsp.JspWriter object. It is used to send the content in a response.

```
Example: out.println("Hello Java");
```

3.7.2 Request: - The request object is an instance of a javax.servlet.http.HttpServletRequest object. Each time a client requests a page the JSP engine creates a new object to represent that request. It is used to request information such as parameters, header information, server names, cookies, and HTTP methods.

```
Example: String name=request.getParameter("rname");
```

Some of the methods of request implicit objects are:

a) getAttributesNames()
b) getCookies()
c) getParameterNames()
d) getHeaderNames()
e) getSesssion
f) getSession(Boolean create)
g) getLocale()
h) getAttribute(String name)

Q9) Explain request and response implicit object

3.7.3 Response: - The response object is an instance of a javax.servlet.http.HttpServletResponse object. Just as the server creates the request object, it also creates an object to represent the response to the client.

```
Example: response.sendRedirect(http://www.google.com);
```

3.7.4 Config: - In JSP, config is an implicit object of type ServletConfig. This object can be used to get initialization parameter for a particular JSP page. The config object is created by the web container for each jsp page.

```
Example: String x=config.getParamter("rname");
```

Q4) Diff betw with example

3.7.5 Application: - In JSP, application is an implicit object of type ServletContext. The instance of ServletContext is created only once by the web container when application or project is deployed on the server.

```
Example: String x=application.getInitParameter("rname");
```

3.7.6 Session: - In JSP, session is an implicit object of type HttpSession. In java developer can use this object to set, get, and remove attribute or to get session information.

```
Example: session.setAttribute("user", x);
```

3.7.7 pageContext :- In JSP, pageContext is an implicit object of type PageContext class. The pageContext object can be used to set, get and remove attribute from one of the following scopes:

i) Page
ii) request
iii) session
iv) application

3.7.8 Page: - This object acts as an actual reference to the instance of the page.

Example: pageContext.removeAttribute("attrName", PAGE_SCOPE);

3.7.9: - In JSP, exception is an implicit object of type java.lang.Throwable class. This object can be used to print the exception but it should be used only in error pages.

Sr. No.	Name of the implicit object	Features	Scope	Instance of class
1.	request	Provides access to all information associated with the request.	Request	HttpServletRequest
2.	response	It is used for setting the headers, cookies & sending response to the client.	Page	HttpServletResponse
3.	out	It allows us to access the servlet output stream, write data to the buffer & send output to the client.	Page	javax.servlet.jsp.JspWriter(While working with servlet PrintWriter class)
4.	session	It is used to maintain the state and the user identity across multiple page request. It provides access to session data.	Session	HttpSession
5.	application	It represents the application to which jsp belongs & this object hold references to other objects that are commonly used for example – database connection pool	Application	ServletContext
6.	config	It contains initialization parameter, defined in deployment descriptor(web.xml) & ServletContext	Page	ServletConfig
7.	pageContext	It is used for transferring the control from current page to another page & for holding data that is shared between multiple components in the same page	Page	javax.servlet.jsp.PageContext
8.	page	It represents the jsp page itself. It is the synonym for "this" (which refers to current object in java)	Page	java.lang.Object
9.	exception	It refers to run time exception that results in an error page	Page	java.lang.Throwable

32) Explain jsp Declaration Tag with example

33) Explain jsp Scriptlet Tag and expression tag with example

There are three building blocks of JSP code:
a) Scriptlet tag
b) Expression tag
c) Declaration tag

3.5. a. Scriptlet Tag :- Java provide various scripting elements that helps the programmer to insert java code from your JSP code into the Servlets. The scriptlet elements have different components which help to write the code in jsp.
Scripting elements in JSP must be written within the <% %> tags. The JSP engine will process any code written within the pair of the <% %> tags, and any other text within the Jsp page will be treated as a html code. In short the sciptlet tag is used to execute the java code in JSP.
Syntax: -<% Java code %>
Example:

```
<html>
    <body>
        <% out.print("Welcome to JSP") %>
    </body>
</html>
```

3.5. b. Expressions Tag: - Expressions elements are comprises of scripting language expressions, which gets executed and converted to String by the JSP engine and it is meant as a response that is output stream. So there is no need for writing out.print() method.
Syntax: <%=statement %>
Example:

```
<html>
    <body>
        <%=" JSP based String" %.>
    </body>
</html>
```

3.5. c. Declaration Tag:- It is used to define or you can say declare methods and variables in JSP. The code written inside the jsp declaration tag is placed outside the service() method.
Syntax: <%! Declaration %>
Example:

```
<html>
  <body>
    <%! int data=50; %>
  </body>
</html>
```

88)

Explain scriptlet, declaration and expression tag in jsp.

(Same answer as above)

Q7) Explain directives of JSP

JSP - Directives

In this chapter, we will discuss Directives in JSP. These directives provide directions and instructions to the container, telling it how to handle certain aspects of the JSP processing.

A JSP directive affects the overall structure of the servlet class. It usually has the following form –

<%@ directive attribute = "value" %>

Directives can have a number of attributes which you can list down as key-value pairs and separated by commas.

The blanks between the @ symbol and the directive name, and between the last attribute and the closing %>, are optional.

There are three types of directive tag –

S.No.	Directive & Description
1	**<%@ page ... %>** Defines page-dependent attributes, such as scripting language, error page, and buffering requirements.
2	**<%@ include ... %>** Includes a file during the translation phase.
3	**<%@ taglib ... %>** Declares a tag library, containing custom actions, used in the page

JSP - The page Directive

The page directive is used to provide instructions to the container. These instructions pertain to the current JSP page. You may code page directives anywhere in your JSP page. By convention, page directives are coded at the top of the JSP page.

Following is the basic syntax of the page directive –

<%@ page attribute = "value" %>

Attributes

Following table lists out the attributes associated with the page directive –

S.No.	Attribute & Purpose
1	**buffer** Specifies a buffering model for the output stream.
2	**autoFlush** Controls the behavior of the servlet output buffer.
3	**contentType** Defines the character encoding scheme.
4	**errorPage** Defines the URL of another JSP that reports on Java unchecked runtime exceptions.
5	**isErrorPage** Indicates if this JSP page is a URL specified by another JSP page's errorPage attribute.
6	**extends** Specifies a superclass that the generated servlet must extend.
7	**import** Specifies a list of packages or classes for use in the JSP as the Java import statement does for Java classes.
8	**info** Defines a string that can be accessed with the servlet's **getServletInfo()** method.
9	**isThreadSafe**

10	**language** Defines the programming language used in the JSP page.
11	**session** Specifies whether or not the JSP page participates in HTTP sessions
12	**isELIgnored** Specifies whether or not the EL expression within the JSP page will be ignored.
13	**isScriptingEnabled** Determines if the scripting elements are allowed for use.

The include Directive

The **include** directive is used to include a file during the translation phase. This directive tells the container to merge the content of other external files with the current JSP during the translation phase. You may code the ***include*** directives anywhere in your JSP page.

The general usage form of this directive is as follows –

<%@ include file = "relative url" >

The filename in the include directive is actually a relative URL. If you just specify a filename with no associated path, the JSP compiler assumes that the file is in the same directory as your JSP. .

The taglib Directive

The JavaServer Pages API allow you to define custom JSP tags that look like HTML or XML tags and a tag library is a set of user-defined tags that implement custom behavior.

The **taglib** directive declares that your JSP page uses a set of custom tags, identifies the location of the library, and provides means for identifying the custom tags in your JSP page.

The taglib directive follows the syntax given below –

<%@ taglib uri="uri" prefix = "prefixOfTag" >

Here, the **uri** attribute value resolves to a location the container understands and the **prefix** attribute informs a container what bits of markup are custom actions.

Q10)

Differentiate between JSP Include Directive and JSP Include action

JSP Include Directive	JSP Include Action
i) Code Syntax: <%@ include file="Footer.jsp" %>	Code Syntax: <jsp:include page="Footer.jsp" />
ii) Output binding: Static binding (Loaded at compile time)	Output binding: Dynamic binding (Loaded at run time)
iii) Prefered when included page does not change often	Preferred when included page changes often
iv) Doesn't allow us to pass any parameters	Allows us to pass parameters using <jsp:params action
v) Doesn't pass request and response object or any configuration	Allows us to pass request and response object.
vi) Only one servlet will be generated for both source jsp file and included resources	Seperate servlet will be generated for source jsp file and included resource
vii) Allows	
viii)	
ix)	
x)	

Beans

4.1 Introduction

EJB is an essential part of a J2EE platform.J2EE application container contains the components that can be used by client for executing business logic. These components are called business logic and business data.EJB mainly comprises of business logic and business data. The EJB component always lies in some container which is called as EJB container. The EJB component is an EJB class which is written by the developer that implement business logic.

4.2 Introduction to Beans

JavaBeans are nothing it's a class that encapsulates many objects into a single object that is nothing but a bean. Java beans should follow some protocol such as:

1) They are serializable
2) Have a zero- argument constructor.
3) Allows access to properties using as getter and setter methods.

Example of JavaBeans class- students.java

```
package mypack;
public class students implements java.io.Serializable
{
private int RollNo;
private String name;
public students(){}
public void setRollNo(int id)
{
this.RollNo=RollNo;
}
public int getId()
{
return RollNo;
}
public void setName(String name)
{
this.name=name;
```

```
}
public String getName()
{
return name;
}
}
```

How to access the JavaBean class?

To access the JavaBean class, we should use getter and setter methods.

```
package mypack;
public class Test
{
public static void main(String args[])
{
students s=new students();   //object is created
s.setName("Rahul");          //setting value to the object
System.out.println(s.getName());
}
}
```

4.3 standard actions

Standard actions in JSP are used to control the behaviour of the Servlets engine. In JSP there are 11 standard actions tag. With the help of these tags we can dynamically insert a file, reuse the beans components, forward user etc.

Syntax: <jsp:action_name attribute="value" />

3 jsp:useBean
4 jsp:include
5 jsp:setProperty
6 jsp:forward
7 jsp:plugin
8 jsp:attribute
9 jsp:body
10 jsp:text
11 jsp:param
12 jsp:attribute
13 jsp:output

4.3.1 jsp:useBean:- This action tag is used when we want to use beans in the JSP page. Using this tag the beans can be easily invoked

Syntax: <jsp:useBean id="" class="" />

4.3.2 jsp:include:- This tag is used to insert a jsp file into another file , same as include directive . It is compute at the time request processing phase.

Syntax: <jsp:include page="page URL" flush="true/false">

4.3.3 jsp:setProperty:- This tag used to set the property of a bean. Before setting this property we need define a bean.

Syntax: <jsp:setProperty name="" property="">

4.3.4 jsp:getProperty:- To get the property of a bean we use this tag. It inserts the output in a string format which is converted into string.

Syntax: <jsp:getAttribute name="" property="">

4.3.5 jsp:forward:- It is basically used to forward the request to another jsp or any static page. Here the request can be forwarded with or with no parameters.

Syntax: <jsp:forward page="value">

4.3.6 jsp:plugin:- It is used for introducing Java components into JSP which is detects the browser and adds the <object> or <embed> JSP tags into the file

Syntax: <jsp:plugin type="applet/bean" code="objectcode" codebase="objectcodebase">

4.3.7 jsp:param:- It is the child object of the plugin object. It contains one or more actions to provide additional parameters.

Syntax: <jsp:params>

<jsp:param nname="val" value="val">

</jsp:param>

4.3.8 jsp:body:- This tag is used for defining the xml dynamically that is the elements can be generated during the request time than at the compilation time.

Syntax: <jsp:body></jsp:body>

4.3.9 jsp:attribute:-This tag is used for defining the xml dynamically that is the elements can be generated during the request time than at the compilation time.

Syntax: <jsp:attribute></jsp:attribute>

4.3.10 jsp:text:- To template text in JSP pages this tag is used. The body of this tag does not contains any elements .It only contains text and EL expressions.

Syntax: <jsp:text>template</jsp:text>

4.3.11jsp: output: - Its consists of XML template text which is placed within text action objects.In this output is declared as XML and DOCTYPE.

Syntax: <jsp:output doctype-root-element="" doctype –system="">

4. 4 session tracking types amd methods:

There are four techniques which can be used to identify a user session.

a) Cookies

b) Hidden Fields

c) URL Rewriting

d) Session Object

4.4.1 Cookie: -

A cookie is small information which is sent by the web server to a web client. It is save at the client side for the given domain and path. It is basically used to identify a client when sending a subsequent request. There are two types of cookies:

a) **Session cookies:** these are temporary cookies and its get deleted as soon as the user closes the browser and next time whenever the client visits the same websites, server will treat the request as a new client as cookies are already deleted.

b) **Persistent Cookie**: its remains on hard drive, until we delete them or they gets expire.

4.4.2 Hidden Filed:

Hidden field are similar to other input fields with the only difference is that these fields are not get displayed on the page but the values of these fields are sent to other input fields.

For example: **<input type="hidden" name="sessionId" value="unique value"/>**

4.4.3 URL Rewriting:

It is a process of appending or modifying the url structure when loading a page. The request made by the client is always treated as new request and the server cannot identify whether the request is new one or the previous same client .so due to this property of HTTP protocol and web servers are called stateless.

4.4.4 Session Object:

It is used for session management. When a user enters a website for the first time Httpsession is obtained via request. When session is created, server generates aunique ID and attaches that ID with every request of that user to server with which the server identifies the client.

How to access or get a session object: By calling getSession() method and it is an implicit object.

a) HttpSession x=request.getSession()

b) HttpSession y=new request.getSession(Boolean)

4.5 Custom Tags:

Custom tags, also known as JSP tag extensions (because they extend the set of built-in JSP tags), provide a way of encapsulating reusable functionality on JSP pages.

- One of the major drawbacks of scripting environments such as JSP is that it's easy to quickly put together an application without thinking about how it will be maintained and grown in the future.
- Use JavaBeans for representing and storing information and state. An example is building JavaBeans to represent the business objects in your application.
- Use custom tags to represent and implement actions that occur on those JavaBeans, as well as logic related to the presentation of information.
- A example from JSTL is iterating over a collection of objects or conditional logic.
- Custom tags have access to implicit objects like request, response, session, etc
- JavaBeans are java classes but all java class are not java beans.
- major one is Custom tag which can be use by the java beans to communicate with each other.
- JavaBeans are normal java classes and don't know anything about JSP.
- JavaBeans are normally used to maintain the data and custom tags for functionality or implementing logic on jsp page.

4.6 Reference for further reading.

https://docs.oracle.com/javase/7/docs/api/

4.7 Bilbliograpgy

https://www.wideskills.com/jsp/jsp-session-tracking-techniques

https://www.geeksforgeeks.org/url-rewriting-using-java-servlet/

https://www.javatpoint.com/java-bean

https://www.javatpoint.com/what-is-ejb

https://www.w3schools.in

https://www.java-samples.com/showtutorial.php?tutorialid=607