## What is the problem with Servlet and how JSP provides solution for these problems?

### Disadvantages with Servlet

1. Servlet is a mixture of java skills and web related HTML skills, because you have to write the business logic in java and for presentation you should the HTML, so the role based development is missing in pure servlet. The developer who is writing servlet should know java and HTML both.

2. The servlet technology require more steps to develop, servlet require too long time for development.

3. If your application is built on using only servlet technology, it is very difficult for enhancement and bug fixing.

4. Servlets of web application requires strong knowledge of Java.

5. Servlets are very difficult for Non-Java Programmers to understand and carry out.

6. We know that, Servlet is a picture of both Presentation logic (HTML) and Business Logic (Java). At the time of development of Servlet by using both of the above (Presentation and Business Logic), it may become imbalance, because a Servlet developer can't be good in both Presentation logic and Business logic.

7. Servlet never provides separation between or clarity between Presentation Logic and Business Logic. So that servlets do not give Parallel Development.

8. If we do any changes in a servlet, then we need to do Re-deployment process i.e. Servlets modifications requires redeployment, which is one of the time-consuming process.

9. If we develop any web application with servlets, then it is mandatory for the web application developer to configure web-application configuration file (Deployment descriptor- web.xml).

10. Servlets do not offer any implicit object [Implicit objects generally provided by containers during the dynamic program execution].

11. Servlets do not contain a facility called Custom Tags Development.

12. Servlets don't provide Global/Implicit exception handling facility.

### Advantages of JSP over Servlet

**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.

**(2) Explain JSP life cycle with neat sketch.**

Following steps are involved in the JSP life cycle:

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

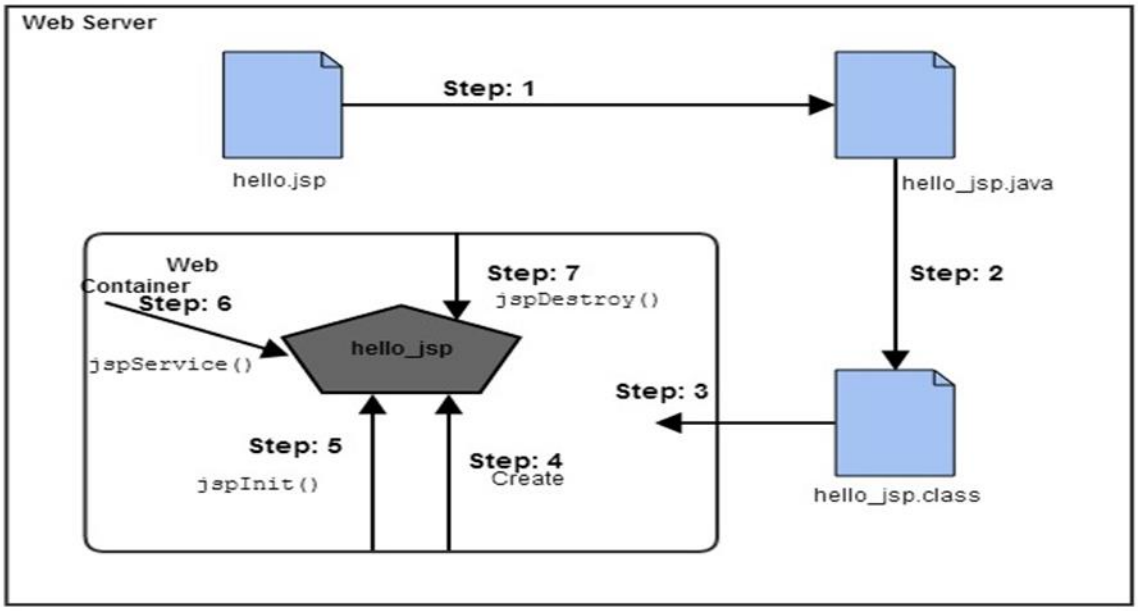
*We can override jspInit(), jspDestroy() but we can't override _jspService() method.*

**Translation of JSP page to Servlet:**

This is the first step of the JSP life cycle. This translation phase deals with the Syntactic correctness of JSP. Here test.jsp file is translated to test.java.

1. **Compilation of JSP page:** Here the generated java servlet file (test.java) is compiled to a class file (test.class).
2. **Classloading:** Servlet class which has been loaded from the JSP source is now loaded into the container.
3. **Instantiation:** Here an instance of the class is generated. The container manages one or more instances by providing responses to requests.
4. **Initialization:** jspInit() method is called only once during the life cycle immediately after the generation of Servlet instance from JSP.

5. **Request processing:** _jspService() method is used to serve the raised requests by JSP. It takes request and response objects as parameters. This method cannot be overridden.
6. **JSP Cleanup:** In order to remove the JSP from use by the container or to destroy the method for servlets jspDestroy()method is used. This method is called once, if you need to perform any cleanup task like closing open files, releasing database connections jspDestroy() can be overridden.



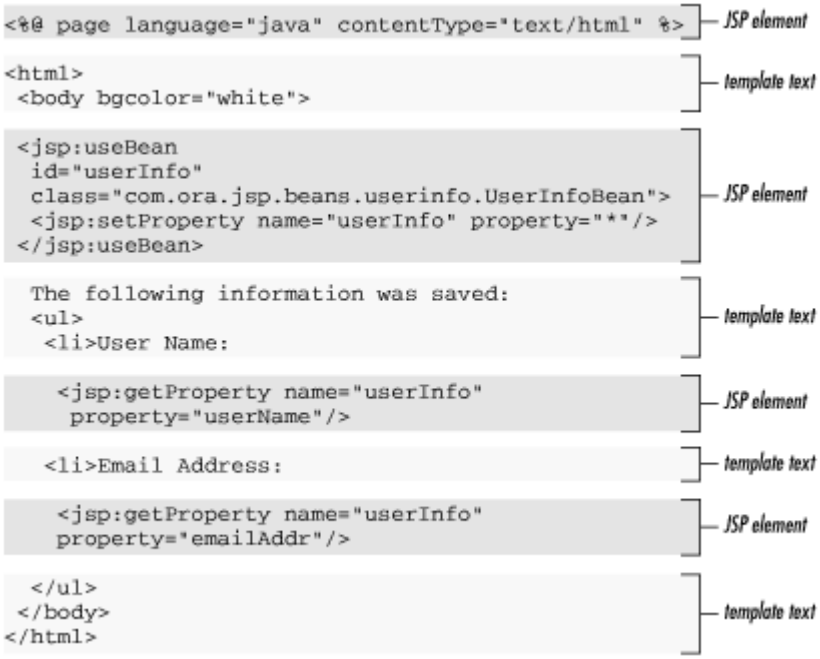

## 3) Explain the anatomy of JSP page.

### Anatomy of JSP Page

JSP page is a simple web page which contains the JSP elements and template text.

- The template text can be scripting code such as Html, Xml or a simple plain text.

- Various Jsp elements can be action tags, custom tags, JSTL library elements. These JSP elements are responsible for generating dynamic contents.




**Example:**

You can write any JSP program as an example for Anatomy of JSP page.

JSP program may contain elements, action tags, directives etc.,

## 5) Discuss the JSP page translation and processing phases.

JSP (JavaServer Pages) is a technology used for developing dynamic web pages in Java. The JSP page translation and processing phases are two important stages in the JSP life cycle. Let's discuss these phases in detail:

1. JSP Page Translation Phase: During this phase, the JSP engine takes the JSP file as input and performs the following tasks:

a. Parsing: The JSP engine parses the JSP file to identify the different elements of the page such as HTML, Java code, and JSP tags.

b. Compilation: After parsing, the JSP engine compiles the Java code present in the JSP file into servlet code. This servlet code is equivalent to the Java code generated by a servlet container for a servlet.

c. Class loading: Once the servlet code is generated, it is loaded into memory by the classloader.

2. JSP Page Processing Phase: During this phase, the JSP engine processes the request and generates a response. The following steps are involved in this phase:

a. Instantiation: When a request is made for a JSP page, the JSP engine creates an instance of the servlet class.

b. Initialization: After the servlet class instance is created, the JSP engine initializes it by calling its init() method.

c. Request Processing: After initialization, the JSP engine processes the request by executing the servlet's service() method. The service() method generates the HTML output for the JSP page.

d. Destruction: Once the request is processed, the JSP engine destroys the servlet instance by calling its destroy() method.

In conclusion, the JSP page translation and processing phases are crucial for the JSP life cycle, and understanding them is essential for developing efficient and robust web applications using JSP technology.

## (6) Write about the JSP processing

) Write about the JSP processing

JSP (JavaServer Pages) processing is a fundamental part of the JSP technology that enables developers to create dynamic web pages with server-side logic. When a user requests a JSP page, the JSP engine processes the page to generate an HTML output. Let's discuss the JSP processing in detail:

1. Request Processing: When a request is made for a JSP page, the JSP engine receives the request and creates an instance of the servlet class that corresponds to the JSP page. The JSP engine then initializes the servlet by calling its init() method.
2. Translation: After initialization, the JSP engine translates the JSP page into a servlet. This servlet is generated by the JSP engine and contains the equivalent Java code for the JSP page.
3. Compilation: Once the JSP engine translates the JSP page into a servlet, it compiles the servlet into bytecode. The bytecode is then loaded into the memory by the classloader.
4. Execution: After the servlet is compiled and loaded into the memory, the JSP engine executes the servlet's service() method. The service() method generates the HTML output for the JSP page. The JSP engine can execute the service() method multiple times to serve multiple requests.
5. Destruction: Once the request is processed, the JSP engine destroys the servlet instance by calling its destroy() method. The destroy() method can be used to release any resources used by the servlet.

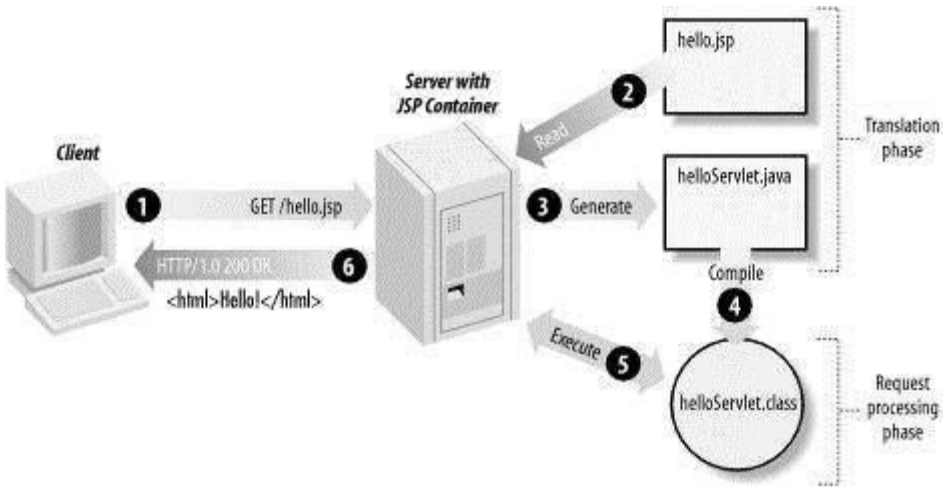
In conclusion, JSP processing is a crucial aspect of the JSP technology that enables developers to create dynamic web pages with server-side logic. By understanding the JSP processing, developers can build efficient and robust web applications using JSP technology.

## 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 servlet content. This conversion is very simple in which all template text is converted to println ( ) statements 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



## Write a JSP to generate the multiplication table of a given number.

```
<html>
<head>
<script language="javascript">
	function multi()
	{
		let i=1;
		var str=" ";
		var x=window.prompt("Enter any number :");
		var y=window.prompt("Enter limit :");
		while(i<=y)
		{
			str=str+x +' x '+i+' = '+x*i+"\n";
			i++;
		}
		alert(str);
	}

multi();
</script>
</head>
</html>
```

## Write about the page directive of JSP with its attributes

## JSP directives

- Directives control the processing of an entire JSP page. It gives directions to the server regarding processing of a page.
- Directive Tag gives special instruction to Web Container at the time of page translation.
- JSP directives are the messages to JSP container. They provide global information about an entire JSP page.

There are three types of directives:

1. Page directive
2. Include directive
3. Taglib directive

**Syntax:**

<%@ directive name [attribute name="value" attribute name="value" .....]%>

### Page directive

- It provides attributes that get applied to entire JSP page.
- It defines page dependent attributes, such as scripting language, error page, and buffering requirements.
- It is used to provide instructions to a container that pertains to current JSP page.

**Syntax:**

<%@ page attribute="value" %>

Attributes of JSP page directive

- import
- contentType
- extends
- info
- buffer
- language
- isELIgnored
- isThreadSafe
- autoFlush
- session
- pageEncoding
- errorPage
- isErrorPage

#### import

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.

## 4)info

This attribute simply sets the information of the JSP page which is retrieved later by using getServletInfo() method of Servlet interface.

## Example of info attribute

```
<html>
<body>

<%@ page info="composed by Sonoo Jaiswal" %>
Today is: <%= new java.util.Date() %>

</body>
</html>
```

The web container will create a method getServletInfo() in the resulting servlet.For example:

```
public String getServletInfo() {
  return "composed by Sonoo Jaiswal";
}
```

---

## 5)buffer

The buffer attribute sets the buffer size in kilobytes to handle output generated by the JSP page.The default size of the buffer is 8Kb.

## Example of buffer attribute

```
<html>
<body>

<%@ page buffer="16kb" %>
Today is: <%= new java.util.Date() %>

</body>
</html>
```

---

## Example of import attribute

```
1. <html>
2. <body>
3.
4. <%@ page import="java.util.Date" %>
5. Today is: <%= new Date() %>
6.
7. </body>
8. </html>
```

---

## 2)contentType

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



## Example of contentType attribute

```
1. <html>
2. <body>
3.
4. <%@ page contentType=application/msword %>
5. Today is: <%= new java.util.Date() %>
6.
7. </body>
8. </html>
```

---

## 3)extends

The extends attribute defines the parent class that will be inherited by the generated servlet.It is rarely used.

---

## 4)info

## 4)info

This attribute simply sets the information of the JSP page which is retrieved later by using getServletInfo() method of Servlet interface.

## Example of info attribute

```
<html>
<body>

<%@ page info="composed by Sonoo Jaiswal" %>
Today is: <%= new java.util.Date() %>

</body>
</html>
```

The web container will create a method getServletInfo() in the resulting servlet.For example:

```
public String getServletInfo() {
  return "composed by Sonoo Jaiswal";
}
```

---

## 5)buffer

The buffer attribute sets the buffer size in kilobytes to handle output generated by the JSP page.The default size of the buffer is 8Kb.

## Example of buffer attribute

```
<html>
<body>

<%@ page buffer="16kb" %>
Today is: <%= new java.util.Date() %>

</body>
</html>
```

---

## 6)language

The language attribute specifies the scripting language used in the JSP page. The default value is "java".

---

## 7)isELIgnored

We can ignore the Expression Language (EL) in jsp by the isELIgnored attribute. By default its value Language is enabled by default. We see Expression Language later.

```
1. <%@ page isELIgnored="true" %>//Now EL will be ignored
```

---

## 8)isThreadSafe

Servlet and JSP both are multithreaded.If you want to control this behaviour of JSP page, you c attribute of page directive.The value of isThreadSafe value is true.If you make it false, the web conta multiple requests, i.e. it will wait until the JSP finishes responding to a request before passing anoth make the value of isThreadSafe attribute like:

<%@ page isThreadSafe="false" %>

The web container in such a case, will generate the servlet as:

```
1. public class SimplePage_jsp extends HttpJspBase
2.   implements SingleThreadModel{
3. .....
4. }
```

---

## 9)errorPage

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.

## Example of errorPage attribute

```
1. //index.jsp
2. <html>
3. <body>
4.
```

```
5. <%@ page errorPage="myerrorpage.jsp" %>
6.
7.  <%= 100/0 %>
8.
9. </body>
10. </html>
```

---

## 10)isErrorPage

The isErrorPage attribute is used to declare that the current page is the error page.

**10) Explain about scriptlets of JSP using code snippets.**

The scripting elements provides the ability to insert java code inside the jsp. There are three types of scripting elements:

- scriptlet tag
- expression tag
- declaration tag

---

## JSP scriptlet tag

A scriptlet tag is used to execute java source code in JSP. Syntax is as follows:

```
1. <%  java source code %>
```

# Example of JSP scriptlet tag

In this example, we are displaying a welcome message.

```
1. <html>
2. <body>
3. <% out.print("welcome to jsp"); %>
4. </body>
5. </html>
```

---

## Example of JSP scriptlet tag that prints the user name

In this example, we have created two files index.html and welcome.jsp. The index.html file gets the username from the user and the welcome.jsp file prints the username with the welcome message.

*File: index.html*

```
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

**11) Explain various action tags used in JSP.**

# JSP Action Tags

There are many JSP action tags or elements. Each JSP action tag is used to perform some specific tasks.

The action tags are used to control the flow between pages and to use Java Bean. The Jsp action tags are given below.

| JSP Action Tags | Description |
|---|---|
| jsp:forward | forwards the request and response to another resource. |
| jsp:include | includes another resource. |
| jsp:useBean | creates or locates bean object. |
| jsp:setProperty | sets the value of property in bean object. |
| jsp:getProperty | prints the value of property of the bean. |

| | |
|---|---|
| jsp:plugin | embeds another components such as applet. |
| jsp:param | sets the parameter value. It is used in forward and include mostly. |
| jsp:fallback | can be used to print the message if plugin is working. It is used in jsp:plugin. |

## JSP ACTION TAGS

The following are the action elements used in JSP:

**1. <jsp:include> Action**

Like **include page directive** this action is also used to insert a JSP file in another file.

**<jsp:include page="page URL" flush="Boolean Value" />**

```
<html>
<head>
<title>Demo of JSP include Action Tag</title>
</head>
<body>
 <h3>JSP page: Demo Include</h3>
<jsp:include page="sample.jsp" flush="false" />
</body>
</html>
```

**2. <jsp:forward> Action**

It is used for redirecting the request. When this action is encountered on a JSP page the control gets transferred to the page mentioned in this action.

**<jsp:forward page="URL of the another static, JSP OR Servlet page" />**

```
<html>
 <head>
<title>Demo of JSP Forward Action Tag</title>
 </head>
<body>
<h3>JSP page: Demo forward</h3>
 <jsp:forward page="second.jsp" />
</body>
</html>
```

**3. <jsp:param> Action**

This action is useful for passing the parameters to Other JSP action tags such as JSP include & JSP forward tag.

**<jsp: param name="param_name" value="parameter_value"**

**4. <jsp:useBean> Action**

This action is useful when you want to use Beans in a JSP page, through this tag you can easily invoke a bean.

**<jsp: useBean id="unique_name_to_identify_bean"**
**class="package_name.class_name" />**

**5. <jsp:setProperty> Action**

This action tag is used to set the property of a Bean, while using this action tag, you may need to specify the Bean's unique name (it is nothing but the id value of useBean action tag).

**<jsp: useBean id="unique_name_to_identify_bean"**
**class="package_name.class_name" />**

**6. <jsp:getProperty> Action**

It is used to retrieve or fetch the value of Bean's property.

**<jsp: useBean id="unique_name_to_identify_bean"**
**class="package_name.class_name" />**

## 12) Explain various implicit objects used in JSP with example

### JSP Implicit Objects

JSP container makes some Java objects available to the JSP page. No specific declaration or initialization is required within the JSP page. These objects are called implicit objects. List of the JSP implicit objects is included below

| Object | Type |
|---|---|
| Out | JspWriter |
| Request | HttpServletRequest |
| Response | HttpServletResponse |
| Config | ServletConfig |
| Application | ServletContext |
| Session | HttpSession |
| pageContext | PageContext |
| Page | Object |
| Exception | Throwable |

### out implicit object

- Out is one of the implicit objects to write the data to the buffer and send output to the client in response
- Out object allows us to access the servlet's output stream
- Out is object of javax.servlet.jsp.jspWriter class
- While working with servlet, we need printwriter object

**Example:**

```
<html>
<head>
</head>
<body>
<% int num1=10;int num2=20;
out.println("num1 is " +num1);
out.println("num2 is "+num2);
%>
</body>
</html>
```

### request implicit object

The JSP request is an implicit object of type HttpServletRequest i.e. created for each jsp request by the web container.

It can be used to get request information such as parameter, header information, remote address, server name, server port, content type, character encoding etc.

It can also be used to set, get and remove attributes from the jsp request scope.

**index.html**

```
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
```

**welcome.jsp**

```
<%
String name=request.getParameter("uname");
out.print("welcome "+name);
%>
```

## response implicit object

- "Response" is an instance of class which implements HttpServletResponse interface
- Container generates this object and passes to _jspservice() method as parameter
- "Response object" will be created by the container for each request.
- It represents the response that can be given to the client
- The response implicit object is used to content type, add cookie and redirect to response page

**index.html**

```
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
```

**welcome.jsp**

```
<%
response.sendRedirect("http://www.google.com");
%>
```

## config implicit object

- "Config" is of the type java.servlet.servletConfig
- It is created by the container for each jsp page
- It is used to get the initialization parameter in web.xml

**index.html**

```
<form action="welcome">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
```

**web.xml file**

```
<web-app>

<servlet>
<servlet-name>s1</servlet-name>
<jsp-file>/welcome.jsp</jsp-file>

<init-param>
<param-name>dname</param-name>
<param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
</init-param>

</servlet>

<servlet-mapping>
<servlet-name>s1</servlet-name>
<url-pattern>/welcome</url-pattern>
</servlet-mapping>

</web-app>
```

**welcome.jsp**

```
<%
out.print("Welcome "+request.getParameter("uname"));

String driver=config.getInitParameter("dname");
out.print("driver name is="+driver);
%>
```

## application implicit object

- Application object is an instance of javax.servlet.ServletContext and it is used to get the context information and attributes in JSP.
- Application object is created by container one per application, when the application gets deployed.
- ServletContext object contains a set of methods which are used to interact with the servlet container. We can find information about the servlet container.

**index.html**

```
<form action="welcome">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
```

**web.xml file**

```
<web-app>
```

```
<servlet>
<servlet-name>s1</servlet-name>
<jsp-file>/welcome.jsp</jsp-file>
</servlet>

<servlet-mapping>
<servlet-name>s1</servlet-name>
<url-pattern>/welcome</url-pattern>
</servlet-mapping>

<context-param>
<param-name>dname</param-name>
<param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
</context-param>

</web-app>
```

**welcome.jsp**

```
<%
out.print("Welcome "+request.getParameter("uname"));
String driver=application.getInitParameter("dname");
out.print("driver name is="+driver);
%>
```

## session implicit object

In JSP, session is an implicit object of type HttpSession. The Java developer can use this object to set, get or remove attribute or to get session information.

**index.html**

```
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

**welcome.jsp**

```
<html>
<body>
<%
String name=request.getParameter("uname");
out.print("Welcome "+name);
session.setAttribute("user",name);
<a href="second.jsp">second jsp page</a>
%>
</body>
</html>
```

**second.jsp**

```
<html>
<body>
<%
String name=(String)session.getAttribute("user");
out.print("Hello "+name);
%>
</body>
</html>
```

## pageContext implicit object

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

- page
- request
- session
- application

In JSP, page scope is the default scope.

**index.html**

```
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

**welcome.jsp**

```
<html>
<body>
<%

String name=request.getParameter("uname");
out.print("Welcome "+name);

pageContext.setAttribute("user",name,PageContext.SESSION_SCOPE);

<a href="second.jsp">second jsp page</a>

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

**second.jsp**

```
<html>
<body>
<%
String name=(String)pageContext.getAttribute("user",PageContext.SESSION_SCOPE);
out.print("Hello "+name);
%>
</body>
</html>
```

### page implicit object

In JSP, page is an implicit object of type Object class. This object is assigned to the reference of auto generated servlet class.

This object is an actual reference to the instance of the page. It can be thought of as an object that represents the entire JSP page.

It is written as:

```
Object page=this;
```

For using this object it must be cast to Servlet type. For example:

```
<% (HttpServlet) page.log ("message"); %>
```

Since, it is of type Object it is less used because you can use this object directly in jsp. For example:

```
<% this.log("message"); %>
```

### exception implicit object

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

## 15) How do you handle errors and exceptions in JSP with example

## 16) Write about error-handling in JSP.