# Java Server Pages(JSP)

# Web Applications

- A web application is such a website where the pages are generated dynamically, based on the user's choices and actions.
- There are 3 main components of any web application. There is:
  - The Web Browser
  - The Web Server
  - The Database
- when a user sends a request to the webserver, the server then submits a query to the database SQL. The results of the SQL query will be collected by the web server and then combining them with HTML send the page back to the user at the browser.

# JSP(Java Server Pages)

- **JSP** technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than servlet such as expression language, JSTL (JavaServer Pages Tag Library (JSTL) is a set of tags that can be used for **implementing some common operations such as looping, conditional formatting, and others**.), etc.

- A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development. It provides some additional features such as Expression Language, Custom Tags, etc.

# Advantages of JSP over Servlet

There are many advantages of JSP over the Servlet. They are as follows:

1) Extension to Servlet

2) Easy to maintain

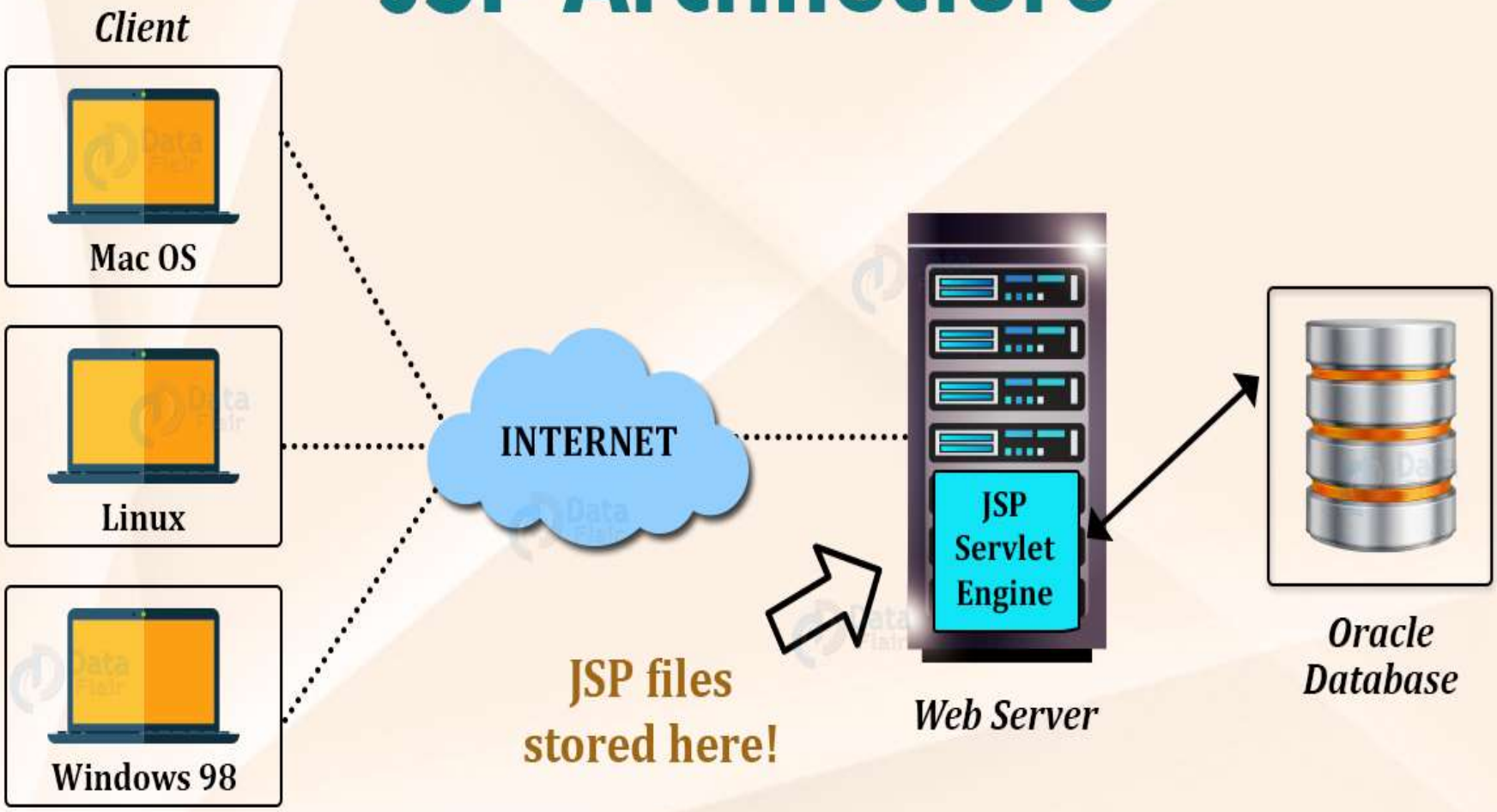
3) Fast Development: No need to recompile and redeploy

4) Less code than Servlet

# Disadvantages of JSP

- It is hard to trace JSP pages error because JSP pages are translated to servlet.
- As JSP output is HTML, it is not rich in features.
- It is very hard to debug or trace errors because JSP pages are first translated into servlets before the compilation process.
- Database connectivity is not easy.
- JSP pages require more disk space and time to hold JSP pages as they are compiled on the server.

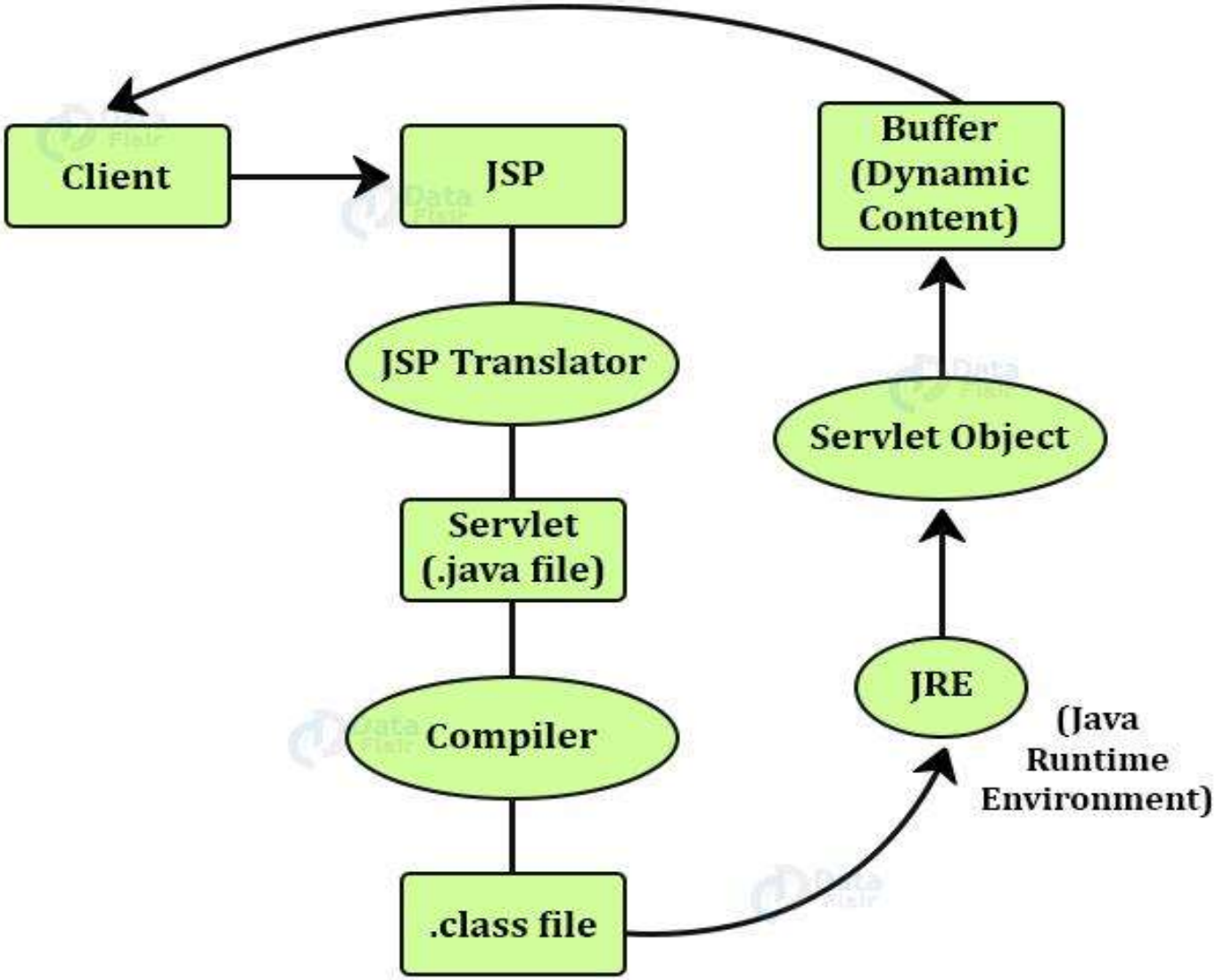
# JSP Architecture



# Difference between a Web Server, a Web Container, and an Application Server

- **Web server:** It receives HTTP request, intercepts it. It also processes the HTTP response to the web browser of the client. E.g., Apache web server.
- **Web container:** A web container is J2EE compliant implementation. Its main job is to run the runtime environment to JSP and servlets. Request receiver at web browser is forwarded here, and the result generated is sent back to the web server. E.g., Tomcat.
- **Application Server:** It is regarded as the complete server as it provides the facility of both a web server as well as a web container. It is a combination of both formers. E.g., Oracle Application Server, Bea WebLogic.

# Phases of JSP Life Cycle

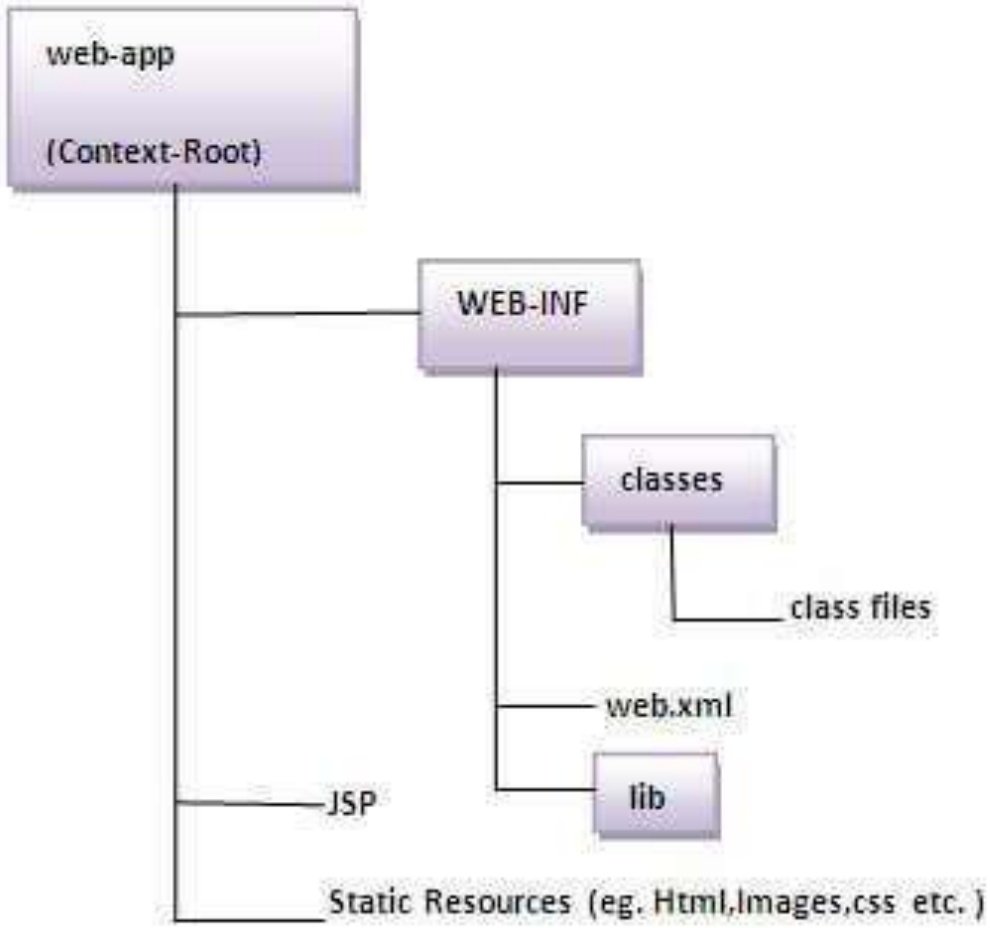


# Stages of Java Server Page

- JSP passes through the following phases:
  1. Translation of JSP Page
  2. Compilation of JSP Page
  3. Classloading (the classloader loads class file)
  4. Instantiation (Object of the Generated Servlet is created).
  5. Initialization ( the container invokes jspInit() method).
  6. Request processing ( the container invokes _jspService() method).
  7. Destroy ( the container invokes jspDestroy() method).

- JSP page is translated into Servlet by the help of JSP translator. The JSP translator is a part of the web server which is responsible for translating the JSP page into Servlet.
- After that, Servlet page is compiled by the compiler and gets converted into the class file. Moreover, all the processes that happen in Servlet are performed on JSP later like initialization, committing response to the browser and destroy.

# The Directory structure of JSP



# JSP Scripting elements

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:

**<% java source code %>**

Example:

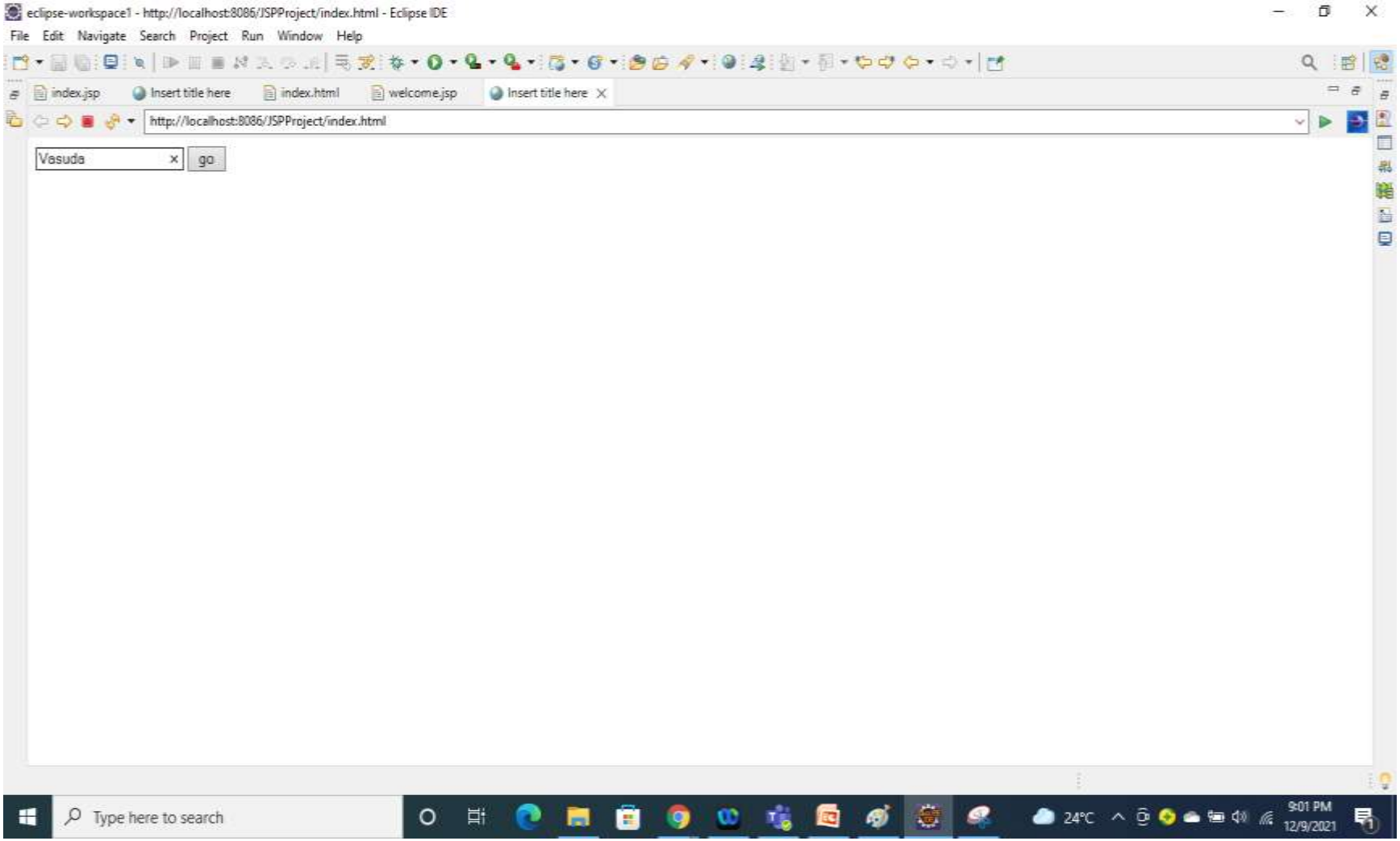
```
<html>
<body>
<% out.print("welcome to jsp"); %>
</body>
</html>
```

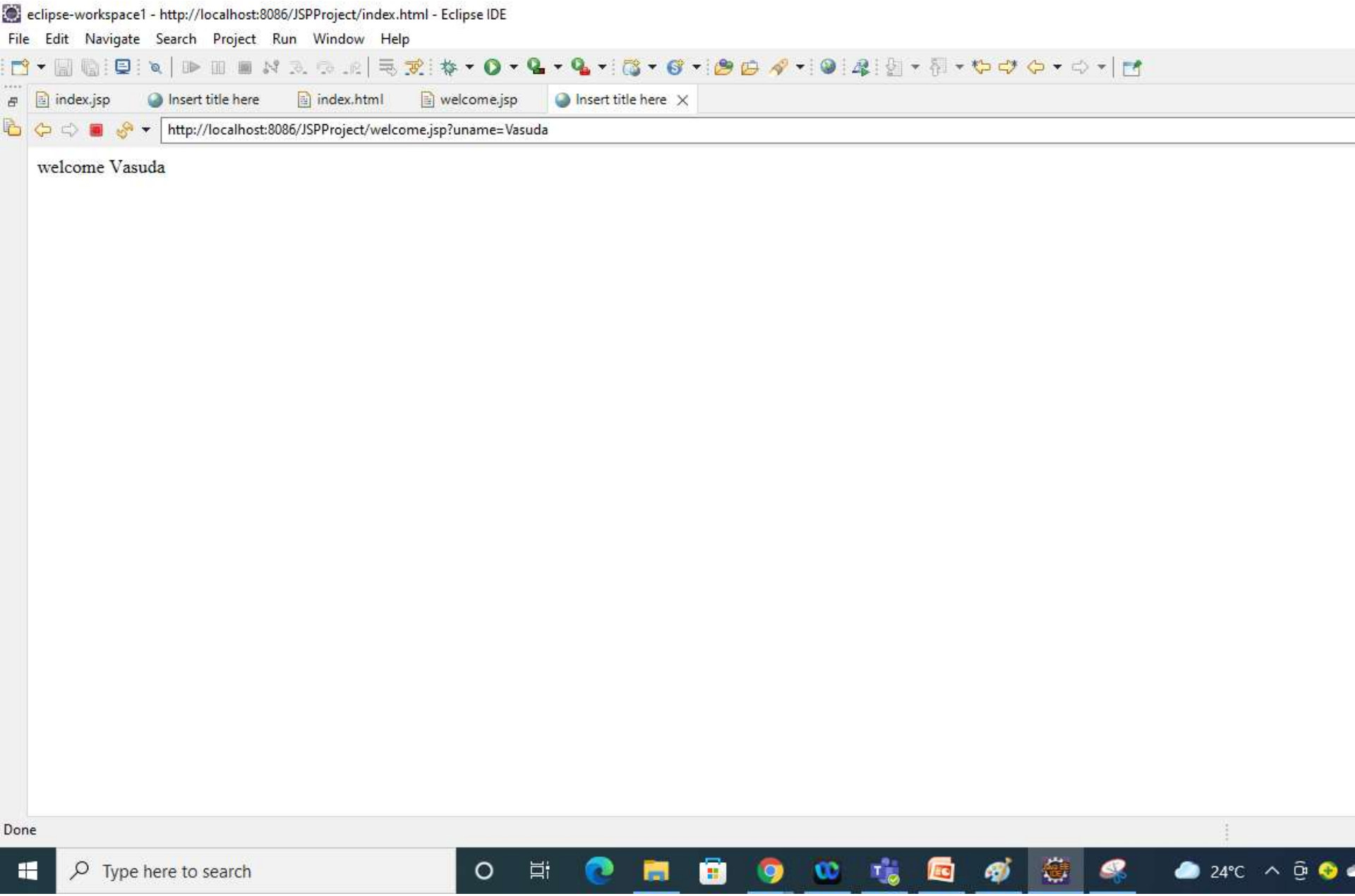
**index.html:**

```
<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);
%>
</form>
</body>
</html>
```


eclipse-workspace1 - http://localhost:8086/JSPProject/index.html - Eclipse IDE
File Edit Navigate Search Project Run Window Help
index.jsp
Insert title here
index.html
welcome.jsp
Insert title here
http://localhost:8086/JSPProject/index.html
Vasuda
go
Type here to search
24°C
9:01 PM
12/9/2021


eclipse-workspace1 - http://localhost:8086/JSPProject/index.html - Eclipse IDE
File Edit Navigate Search Project Run Window Help
index.jsp
Insert title here
index.html
welcome.jsp
Insert title here
http://localhost:8086/JSPProject/welcome.jsp?uname=Vasuda
welcome Vasuda
Done
Type here to search
24°C

# JSP 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 of JSP expression tag:

**<%= statement %>**

- Example:

```
<html>
<body>
<%= "welcome to jsp" %>
</body>
</html>
```

***Note: Do not end your statement with semicolon in case of expression tag.***

# Ex2: JSP expression tag that prints current time

```
<html>
<body>
Current Time: <%= java.util.Calendar.getInstance().getTime() %>
</body>
</html>
```

# JSP expression tag that prints the user name

## Index.jsp:

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

## Welcome.jsp:

```
<html>
<body>
<%= "Welcome "+request.getP
arameter("uname") %>
</body>
</html>
```

# Difference between Scriptlet and Expression Tag

| Scriptlet tag | Expression Tag |
| --- | --- |
| Scriptlet tag doesn’t evaluates a Java expression | Expression tag evaluates a Java expression |
| It will not display any result itself until the user displays it. | It directly writes the result to the client side. |
| We will have to write out.print() statements. | There is no need to write out.println for printing because these are converted into out.print() statement |

# JSP 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 of JSP declaration tag:

## <%! field or method declaration %>

**Index.jsp:**

```
<html>
<body>
<%! int data=50; %>
<%= "Value of the variable is:"+data %>
</body>
</html>
```

**Index.jsp:**

```
<html>
<body>
<%!
int cube(int n){
return n*n*n*;
}
%>
<%= "Cube of 3 is:"+cube(3) %>
</body>
</html>
```

# Difference between JSP Scriptlet tag and Declaration tag

| Jsp Scriptlet 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. |

# JSP Implicit Objects

- Web containers create these implicit objects when they translate JSP pages to servlet. JSP Implicit Objects are also called **pre-defined variables**.
- There are **9 jsp implicit objects**.

| Object | Type |
|---|---|
| out | JspWriter |
| request | HttpServletRequest |
| response | HttpServletResponse |
| config | ServletConfig |
| application | ServletContext |
| session | HttpSession |
| pageContext | PageContext |
| page | Object |
| exception | Throwable |

# out implicit object

For writing any data to the buffer, JSP provides an implicit object named out. It is the object of JspWriter. In case of servlet you need to write:

PrintWriter out=response.getWriter();

But in JSP, you don't need to write this code.

**Ex:**

```
<html>
<body>
<% out.print("Today is:"+java.util.Calendar.getInstance().getTime()); %
    >
 </body>
</html>
```

# JSP 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.jsp:

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

## Welcome.jsp:

```
<%= "Welcome "+request.getParameter("uname") %>
```

# JSP response implicit object

- In JSP, response is an implicit object of type HttpServletResponse. The instance of HttpServletResponse is created by the web container for each jsp request.
- It can be used to add or manipulate response such as redirect response to another resource, send error etc.

**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");
%>
```

# JSP config implicit object

- 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.
- Generally, it is used to get initialization parameter from the web.xml file.

**index.html**

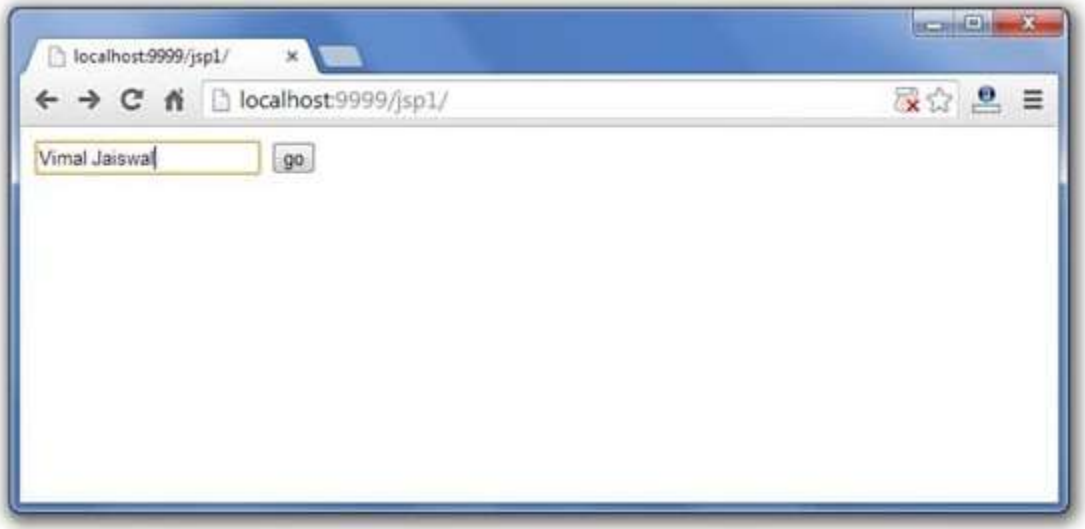
```
<form action="welcome">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
```

**web.xml file**

```
<web-app>

<servlet>
<servlet-name>sonoojaiswal</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>sonoojaiswal</servlet-name>
<url-pattern>/welcome</url-pattern>
```

**welcome.jsp**

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

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


localhost:9999/jsp1/
localhost:9999/jsp1/
Vimal Jaiswal
go


localhost:9999/jsp1/welc
localhost:9999/jsp1/welcome?uname=Vimal+Jaiswal
Welcome Vimal Jaiswal
driver name is=sun.jdbc.odbc.JdbcOdbcDriver

# JSP application implicit object

- 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.
- This object can be used to get initialization parameter from configuaration file (web.xml). It can also be used to get, set or remove attribute from the application scope.
- This initialization parameter can be used by all jsp pages.

**index.html**

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

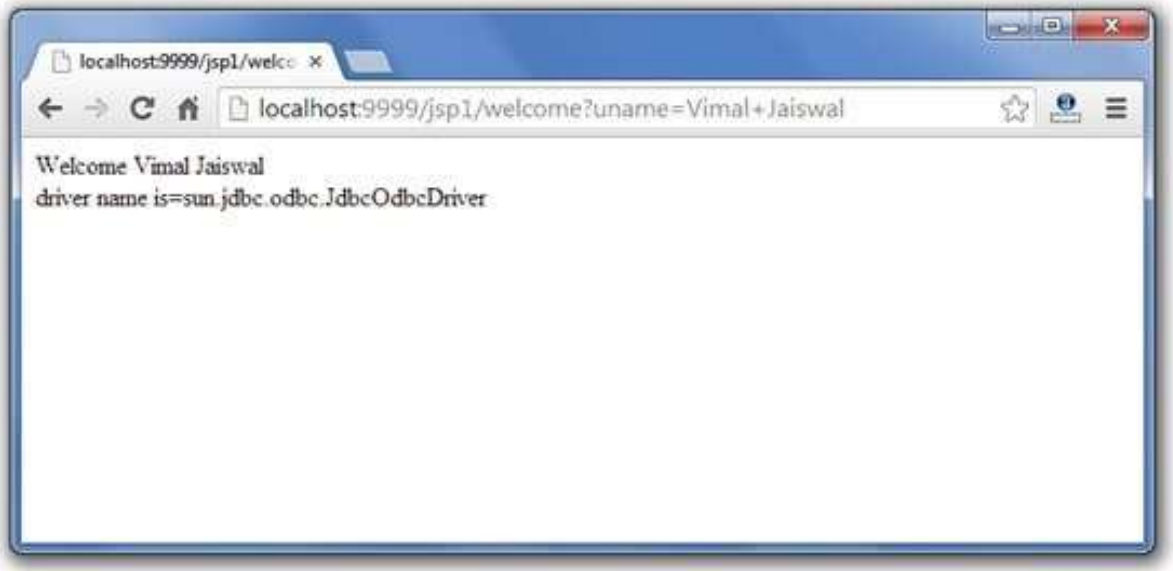
**web.xml file**

```
<web-app>

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

<servlet-mapping>
<servlet-name>sonoojaiswal</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>
```

**welcome.jsp**

```
<%

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

String driver=application.getInitParameter("dname");
out.print("driver name is="+driver);

%>
```


localhost:9999/jsp1/
localhost:9999/jsp1/
Vimal Jaiswal
go


localhost:9999/jsp1/welc
localhost:9999/jsp1/welcome?uname=Vimal+Jaiswal
Welcome Vimal Jaiswal
driver name is=sun.jdbc.odbc.JdbcOdbcDriver

# 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**

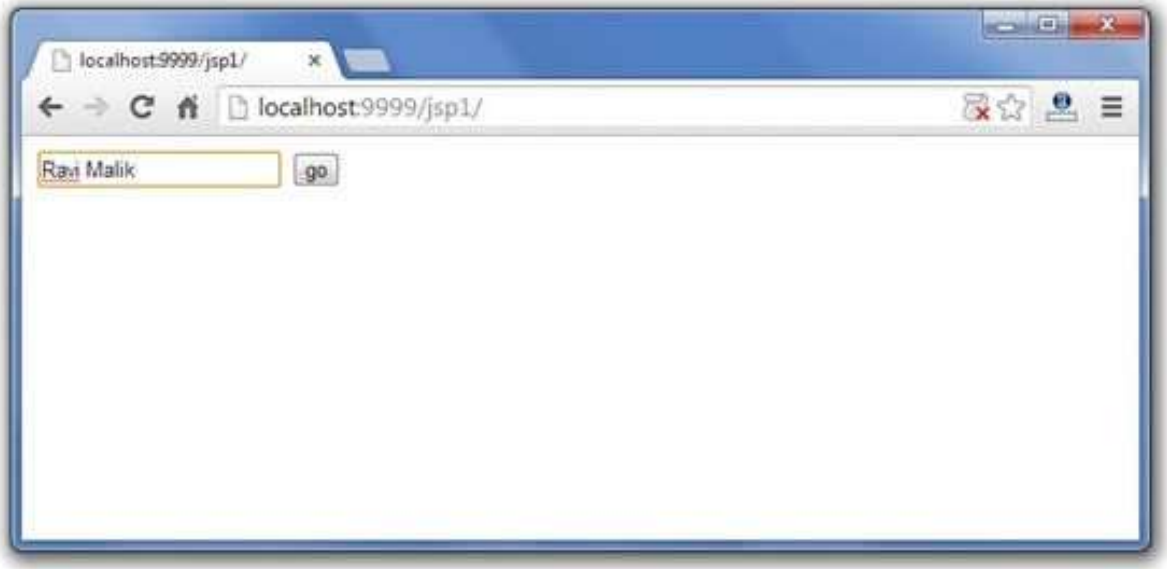
```
<html>
<body>
<form action="welcome.jsp">
<input type="text" name="uname">
<input type="submit" value="go"><br/>
</form>
</body>
</html>
```

## welcome.jsp

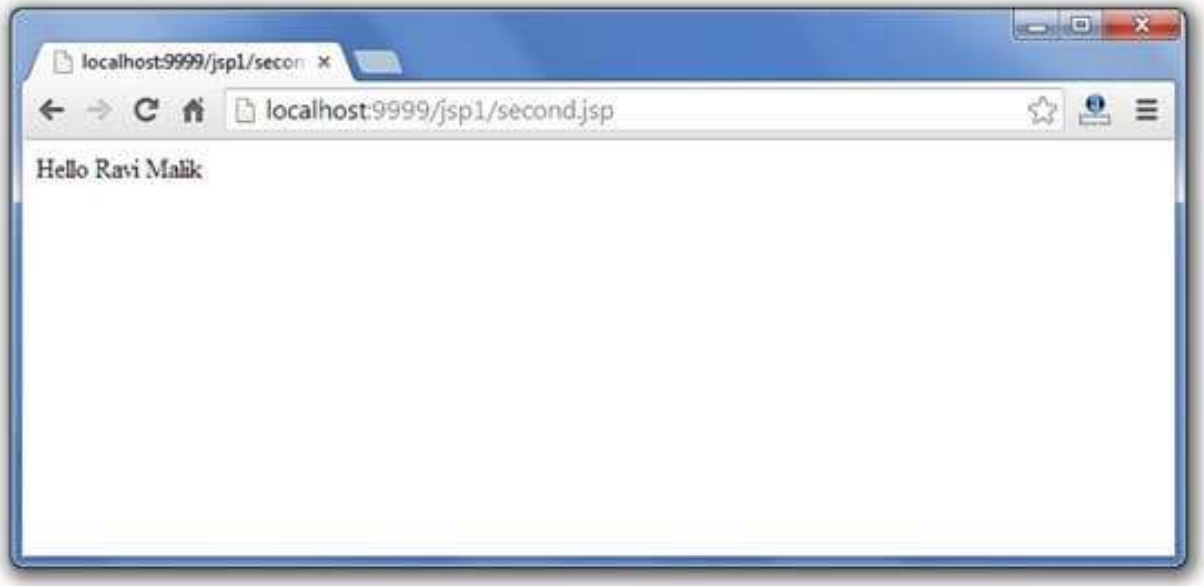
```
<html>
<body>
<%

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>
```


localhost:9999/jsp1/
localhost:9999/jsp1/
Ravi Malik
go


localhost:9999/jsp1/welc
localhost:9999/jsp1/welcome.jsp?uname=Ravi+Malik
Welcome Ravi Malik
second jsp page


localhost:9999/jsp1/secon
localhost:9999/jsp1/second.jsp
Hello Ravi Malik

# 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
out.print("Hello "+name);

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


localhost:9999/jsp1/
localhost:9999/jsp1/
Ravi Malik
go


localhost:9999/jsp1/welc
localhost:9999/jsp1/welcome.jsp?uname=Ravi+Malik
Welcome Ravi Malik
second jsp page


localhost:9999/jsp1/secon
localhost:9999/jsp1/second.jsp
Hello Ravi Malik

# 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. 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.

**error.jsp**

```
<%@ page isErrorPage="true" %>
<html>
<body>

Sorry following exception occured:<%= exception %>

</body>
</html>
```