Servlet and JSP

Server :

Server is nothing but a computer and its have OS and it is used to stores, send and receives data.

Request Contains

- URL
- Method Type
- Form Data (Not Mandatory)

Response Contains

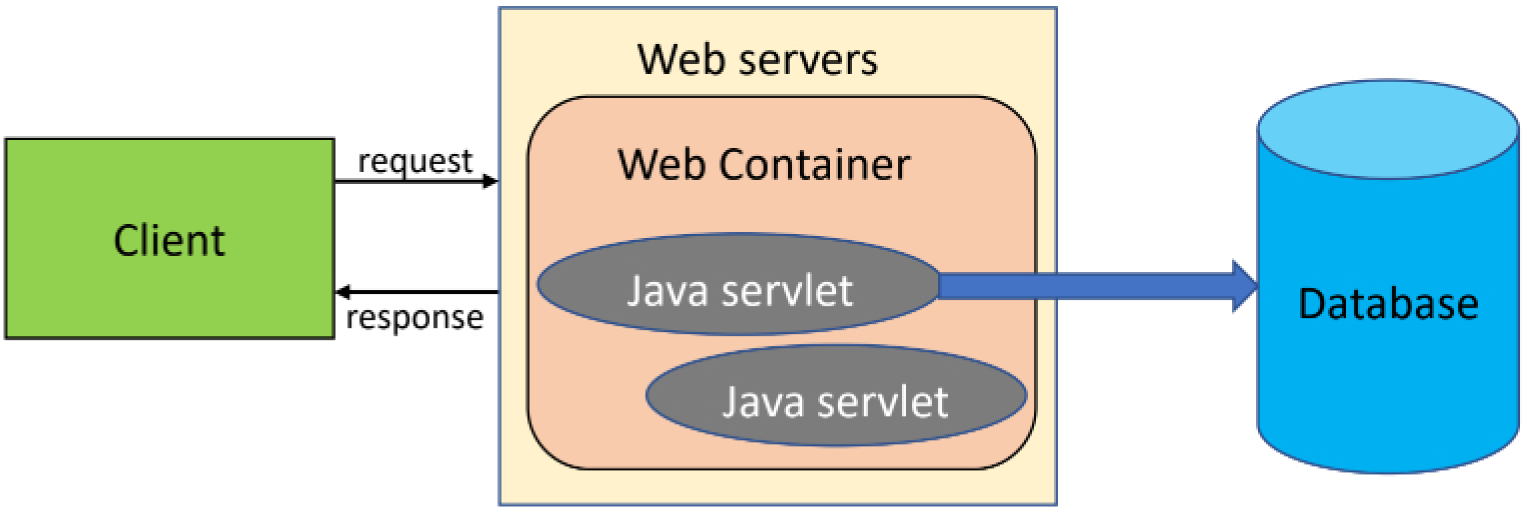
- Status Code (400,405,404,500 ...)
- Content Type
- Actual Content

Some Important types of servers :

- Web server
- Application server

Application Server	Web Server
Application server contains web container and EJB container	Web Server Contains only web Container.
Application server is relevant in case of dynamic content	Web server is good in case of static content
Multithreading is supported	Multithreading is not supported
Application server is supports HTTP and RMI protocol	Web Server is support only HTTP protocol
Ex: Weblogic,JBoss	Ex : Apache web server

Architecture of web servers and Web container

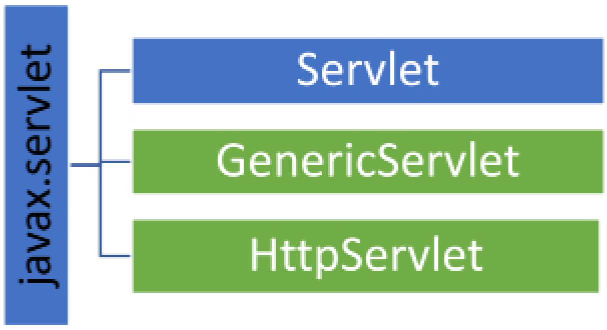


- The client sends a request to a web server.
- The web server which contains the servlet sends that request to the web container.
- The container passes the request to the respective servlet.
- The servlet methods are loaded.
- The servlet hands over the response to the container, which passes to the web server. Eventually response reaches the client.

Servlet

- Servlet is a one of the API
- Servlet is small piece of program and it is used to run the server.

Servlet Hierarchy



Servlet Interface:

- Servlet is an Interface
- Servlet is present inside a javax.servlet package

In a Servlet Interface having five Abstract method

init() : initialize the servlet and it is life cycle method of servlet and
invoked by the web container only once .
service( ) : provides the response for the incoming request and It is invoked at each request by the web container
destroy( ) : it is invoked only once and its indicate that servlet is being destroyed.
getServletConfig() : return the object of ServletConfig
getServletInfo( ) : returns information about servlet such as writer,copyright,version etc…

Deployment Descriptor:

Deployment Descriptor is the file used by the servlet container to define which servlet match up with which URL's.

It is nothing but a web.xml and it is present inside a WEB-INF folder

Ex :

```
<web-app>
        <servlet>
        <servlet-name>MyFirstServlet</servlet-name>
        <servlet-class>Fully Qualified class Name</servlet-class>
        </servlet>
        <servlet-mapping>
        <servlet-name>MyFirstServlet</servlet-name>
        <url -pattern>/MyFirstServlet</url -pattern>
        </servlet-mapping>
</web-app>
```

GenericServlet:

- GenericServlet is an Abstract Class and it is implements Servlet, Serializable interface
- GenericServlet is use to create a servlet.
- GenericServlet class can handle any type of request so it is protocol-independent.
- GenericServlet is present in javax.servlet.GenericServlet package.
- We creating Servlet by inheriting the GenericServlet class.

GenericServlet having One Abstract Method :

```
public void service(ServletRequest req,ServletResponse res) ;
```

You may create a generic servlet by inheriting the GenericServlet class and providing the implementation of the service method.

```
import java.io.*;
import javax.servlet.*;
public class ABC extends GenericServlet
{
        public void service(ServletRequest req, ServletResponse resp)
                throws ServletException, IOException
        {
                PrintWriter pw = resp.getWriter();
                pw.println("<html>");
                pw.println("<head><title>My first Servlet</title></head>");
                pw.println("<body>");
                pw.println("<h2>Welcome to Servlet</h2>");
                pw.println("</body>");
                pw.println("</html>");
        }
}
```

HttpServlet :

- HttpServlet is Abstract class and it is extends GenericServlet which is used to create a Servlet and it is present javax.servlet.HttpServlet package

or

- HttpServlet is special type of Servlet that handles an Http request and provides an Http response usually in the form of HTML Page.

Some Important Methods of HttpServlet class:

- **doGet(HttpServletRequest req, HttpServletResponse res) : void**
- **doPost(HttpServletRequest req, HttpServletResponse res) : void**
- **doPut(HttpServletRequest req, HttpServletResponse res) : void**
- **doDelete(HttpServletRequest req, HttpServletResponse res) : void**
- **doOptions(HttpServletRequest req, HttpServletResponse res) : void**
- **service(ServletRequest req,ServletResponse res) : void**
- **service(HttpServletRequest req, HttpServletResponse res) : void**

doGet( )	doPost( )
It is a default method for any Http request	It is not default request
It is not suitable for Encrypt operations	It is suitable for encrypt operation
It is not suitable for file uploading operation	It is suitable for file uploading operation
We can send maximum size of data 240 bytes	We can send unlimited data
Parameter is not encrypted	Parameter is Encrypted
doGet( ) trigger by itself	doPost( ) not trigger by itself

With the help of doGet( ) we pass video ,audio and image	With the help of doPost( ) we pass video ,audio and image

PrintWriter :

PrintWriter is a class is extends from a Writer class and it is used to print the formatted representation of objects to the text-output stream.

Ex :

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        PrintWriter printWriter = resp.getWriter();
        printWriter.write("<html><body><h1>Hi</h1></body></html>");
}
```

RequestDispatcher :

RequestDispatcher is an interface and it is providing the facility of dispatching the request to another resource.

Or

RequestDispatcher is an interface and it is provides the facility of dispatching the request from one servlet to another servlet or one servlet to web pages

methods of RequestDispatcher :

- forward(ServletRequest req, ServletResponse resp) : void
- include(ServletRequest req, ServletResponse resp) : void

Method	Description
forward(ServletRequest req, ServletResponse resp)	Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.

include(ServletRequest req, ServletResponse resp)	Includes the content of a resource (servlet, JSP page, or HTML file) in the response.

Why use Session Tracking?

To recognize the user It is used to recognize the particular user.

Session Tracking Techniques

There are four techniques used in Session tracking

- **URL Rewriting**
- **Hidden Form Field**
- **Cookies**
- **HttpSession**

Hidden Form Field

In case of Hidden Form Field, **a hidden (invisible) text field** is used for maintaining the state(information) of a user.
In such case, we store the information in the hidden field and get it from another servlet

Syntax :

```
<input type="hidden" name="uname" value="Jspider">
```

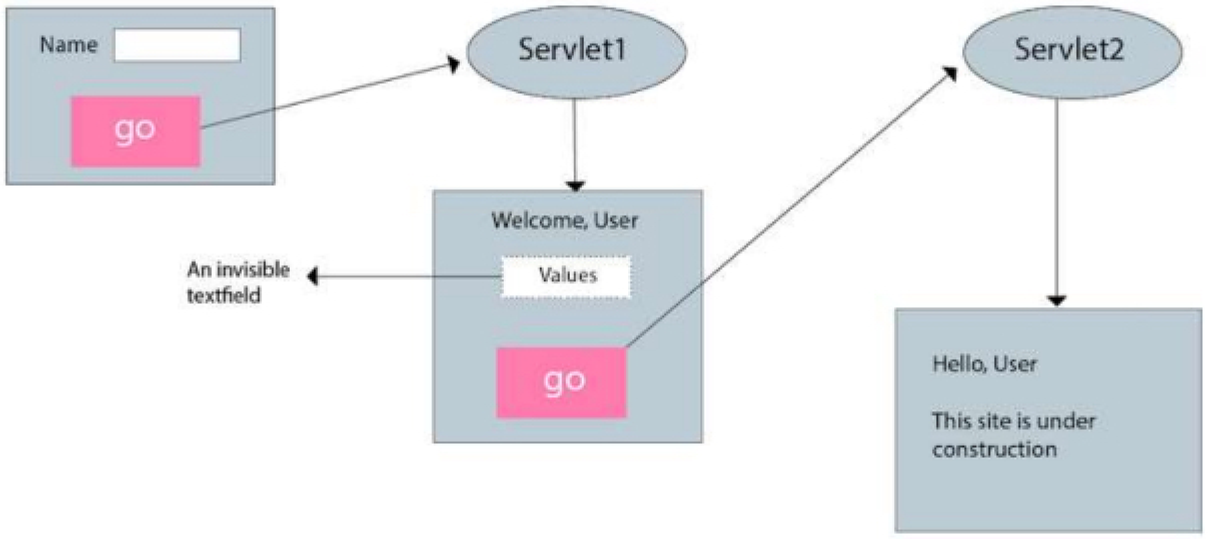
Here, uname is the hidden field name and Jspider is the hidden field value.

Advantage of Hidden Form Field

- it doesn't depend on the browser. Even If the cookies are disabled or not hidden

Disadvantage of Hidden Form Field:

- It is maintained at server side.
- Only textual information can be used.



URL Rewriting

In URL rewriting, we append a token or identifier to the URL of the next Servlet or the next resource.

We can send parameter name/value pairs using the following format:

url?name1=value1&name2=value2

- A name and a value are separated using an equal = sign
- a parameter name/value pair is separated from another parameter using the ampersand (&).
- When the user clicks the hyperlink, the parameter name/value pairs will be passed to the server. From a Servlet, we can use getParameter () method to obtain a parameter value.

Advantage of URL Rewriting

- It will always work whether cookie is disabled or not (browser independent).
- Extra form submission is not required on each page.

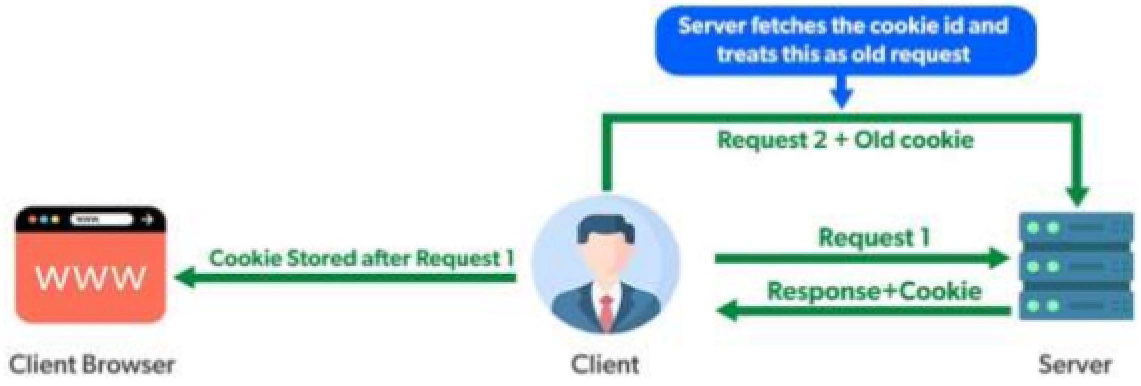
Disadvantage of URL Rewriting

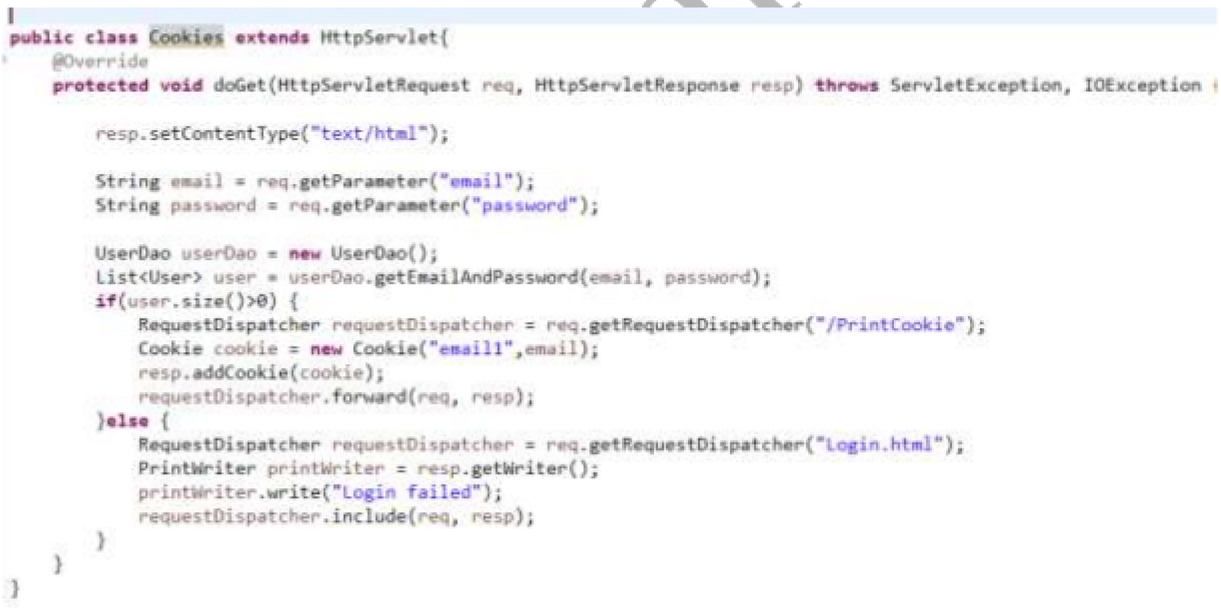
- It will work only with links.
- It can send Only textual information.

Cookies

- Cookies are the textual information that is stored in key-value pair format to the client's browser during multiple requests and it is present inside a **javax.servlet.http package**
- Basically, the server treats every client request as a new one so to avoid this situation cookies are used.

When the client generates a request, the server gives the response with cookies having an id which are then stored in the client's browser. Thus, if the client generates a second request, a cookie with the matched id is also sent to the server. The server will fetch the cookie id, if found it will treat it as an old request otherwise the request is considered new.


Server fetches the cookie id and treats this as old request
Request 2 + Old cookie
Cookie Stored after Request 1
Request 1
Response+Cookie
WWW
Client Browser
Client
Server

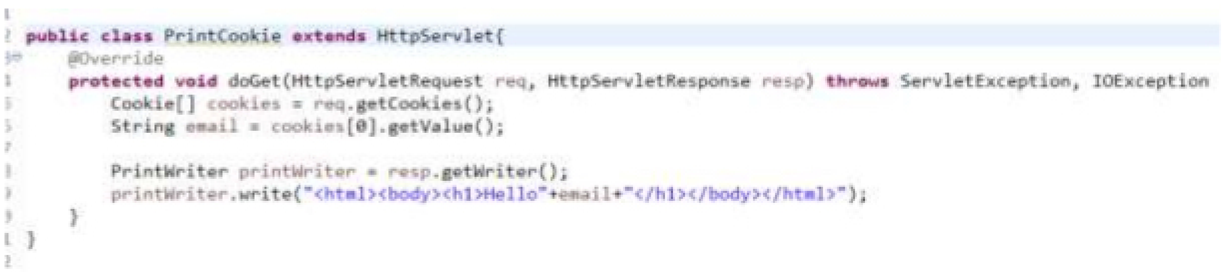


```
public class Cookies extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException

        resp.setContentType("text/html");

        String email = req.getParameter("email");
        String password = req.getParameter("password");

        UserDao userDao = new UserDao();
        List<User> user = userDao.getEmailAndPassword(email, password);
        if(user.size()>0) {
            RequestDispatcher requestDispatcher = req.getRequestDispatcher("/PrintCookie");
            Cookie cookie = new Cookie("email1",email);
            resp.addCookie(cookie);
            requestDispatcher.forward(req, resp);
        }else {
            RequestDispatcher requestDispatcher = req.getRequestDispatcher("Login.html");
            PrintWriter printWriter = resp.getWriter();
            printWriter.write("Login failed");
            requestDispatcher.include(req, resp);
        }
    }
}
```



```
public class PrintCookie extends HttpServlet{
    @Override
    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException
        Cookie[] cookies = req.getCookies();
        String email = cookies[0].getValue();

        PrintWriter printWriter = resp.getWriter();
        printWriter.write("<html><body><h1>Hello"+email+"</h1></body></html>");
    }
}
```

Task - **HttpSession**

JSP

- It stands for Java Server Pages.
- It is a server-side technology.
- It is used for creating web application.
- It is used to create dynamic web content.
- In this JSP tags are used to insert JAVA code into HTML pages.

JSP pages are more advantageous than Servlet:

- They are easy to maintain.
- recompilation or redeployment is required.
- JSP has access to entire API of JAVA.
- JSP are extended version of Servlet.

Features of JSP

- Coding in JSP is easy: - As it is just adding JAVA code to HTML/XML.
- Reduction in the length of Code: - In JSP we use action tags, custom tags etc.
- Connection to Database is easier: -It is easier to connect website to database and allows to read or write data easily to the database.
- Make Interactive websites: - In this we can create dynamic web pages which
- Extension to Servlet: - as it has all features of servlets, implicit objects and custom tags

JSP syntax for writing the java code

- Declaration Tag: -It is used to declare variables and methods.
    - Syntax: - <%!   %>

- Java Scriplets: - It allows us to add any number of JAVA code, local variables and logics.
    - Syntax: - <%   %>

- JSP Expression: - It is use to print the any data.
    - Syntax: - <%=   %>

JSP	Servlet
JSP is a HTML based code.	Servlet is a java code.
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 is faster than JSP.
JSP only accepts HTTP requests	Servlet can accept all protocol requests
In JSP, we cannot override its service() method	In Servlet, we can override the service() method.
In JSP session management is automatically enabled.	In Servlet by default session management is not enabled, user have to enable it explicitly.
In JSP there are inbuilt implicit objects	It does not have inbuilt implicit objects.
Packages can be imported into the JSP program(i.e. bottom , middle client -side or top)	Packages are to be imported on the top of the program.

- JAVA Comments: - It contains the text that is added for information which has to be ignored.
    - Syntax: - <%--   %>

Advantages of using JSP

- It does not require advanced knowledge of JAVA.
- It is capable of handling exceptions.
- Difficult to debug for errors.
- First time access leads to wastage of time
- Its output is HTML which lacks features.

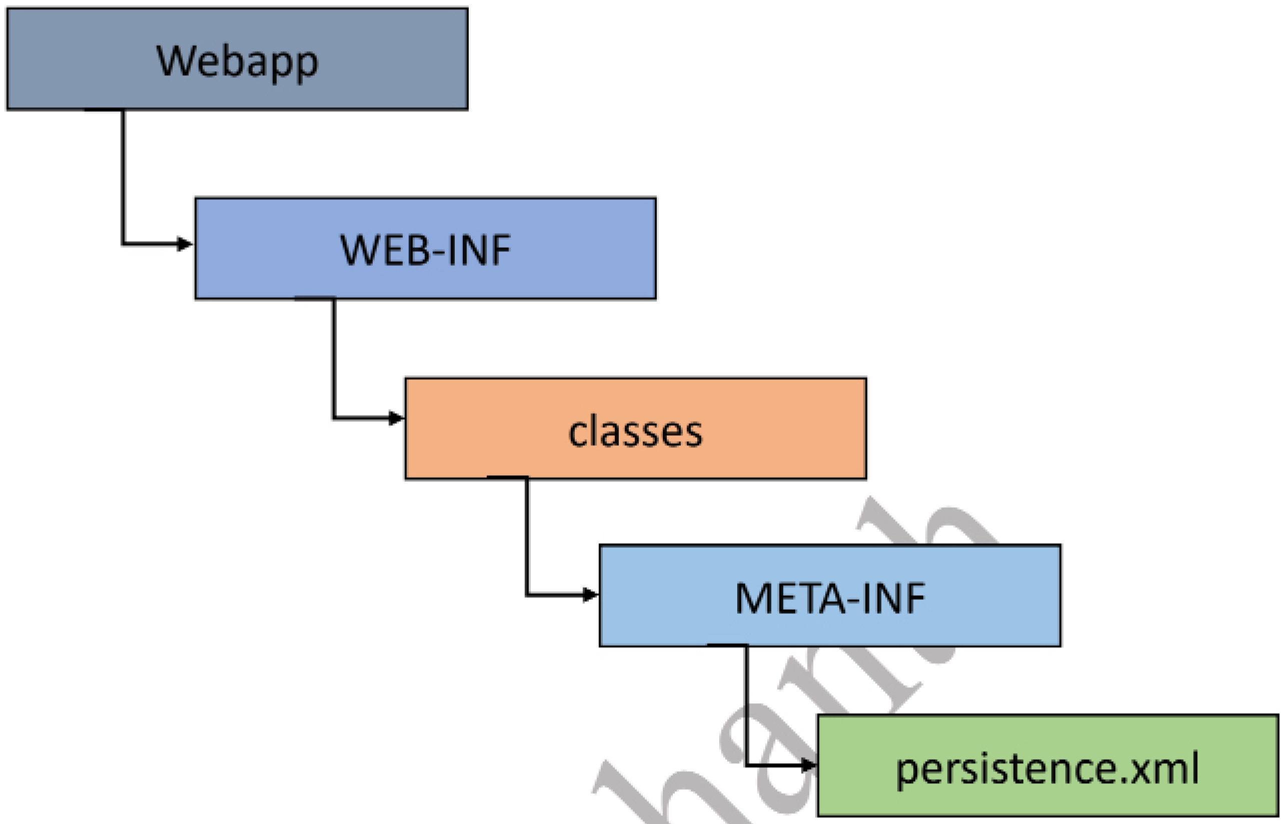
The Lifecycle of a JSP Page

- Parsing
- Compilation
- Servlet class is loaded .
- Servlet instance is created.
- init method is invoked.
- service method is invoked.
- destroy method is invoked.

In a Parsing, JSP page will be converted into Servlet page after this step that java file convert into class file this process is done in a Compilation , The class loader is responsible to load the servlet class into a web container and after the servlet class is loaded in a web container instance of servlet class is created after that we call the jpsInit( ) this method initialised the servlet and by calling jspService( ) , in service class this method to process a clients request and when we call the jspDestroy( ) in that time servlet is terminated.

Persistence File :

It is a file which is having the extension .xml and which is use to store a configurable data .



```
<html> <body>
<h1> Welcome to add two Number</h1>
<%! String user = "Pavan";
      int add(int a , int b){
       return a+b;
       }
%>
<h2>Sum is <%= add(10,20) %></h2>
<h2>Welcome my dear <%= user %></h2>
<%  int a = add(20,20);
         double avg = a/20;
%>
<h2>Average is <%= avg %></h2>
</body>
</html>
```