

Topics:-

- **What is Servlet?**
- **What is HTTP?**
- **HTTP Request / Response:-**
- **What is XML?**
- **Request dispatcher**
- **Send redirect Method**
- **Session Management :-**
- **WebServlet Annotations :-**
- **Techniques to handle Session Management:-**
- **Setting up for Coding:-**
- **JSP (Java Server Pages):-**
- **Advantages JSP over Servlets:-**
- **Features of JSP:-**
- **JSP life cycle: -**
- **HTTP (Hyper Text Transfer Protocol) :-**
- **HTTP Requests:-**
- **Get HTTPsrequest:-**
- **Post HTTPsrequest:-**
- **Get and Post**
- **Request Dispatcher:-**
- **SendRedirect:-**
- **Url Writing:-**
- **HttpSession:-**
- **Cookie:-**
- **WebServlet Annotations :-**



Servlet
JAVA

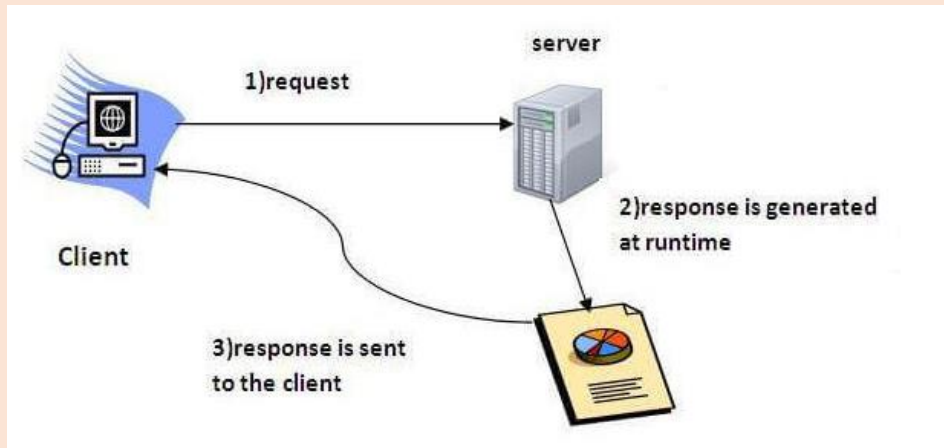
Advanced Java

Servlet

• What is Servlet?

Servlet technology is used to create a web application (resides at server side and generates a dynamic web page).

- Servlet is a technology which is used to create a web application.
- Servlet is an API that provides many interfaces and classes including documentation.
- Servlet is an interface that must be implemented for creating any Servlet.
- Servlet is a class that extends the capabilities of the servers and responds to the incoming requests. It can respond to any requests.
- Servlet is a web component that is deployed on the server to create a dynamic web page.

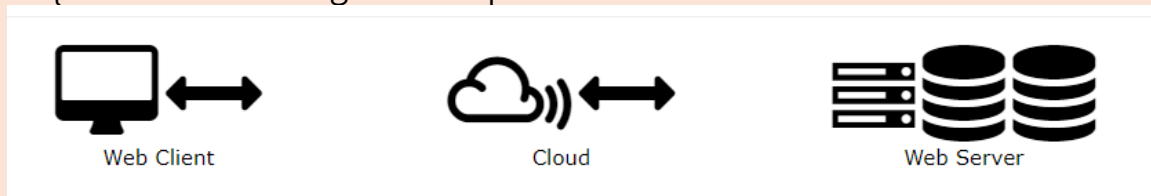


• What is HTTP?

HTTP stands for **H**yper **T**ext **T**ransfer **P**rotocol

WWW is about communication between web clients and servers

Communication between client computers and web servers is done by sending HTTP Requests and receiving HTTP Responses.



• HTTP Request / Response:-

Communication between clients and servers is done by requests and responses:

1. A client (a browser) sends an HTTP request to the web
2. A web server receives the request
3. The server runs an application to process the request
4. The server returns an HTTP response (output) to the browser
5. The client (the browser) receives the response

• What is XML?

- XML stands for Extensible Markup Language.
- XML plays an important role in many different IT systems.
- XML is often used for distributing data over the Internet .
- It is important for all web developers to have a good understanding of XML .

• Inside Servlets we have 2 methods:

Get – Requests data from a specified resource.

Post – Submits data to be processed to a specified resource.

“Servlet Collaboration” - Calling another Servlet from servlet:

- Request dispatcher

- Send redirect

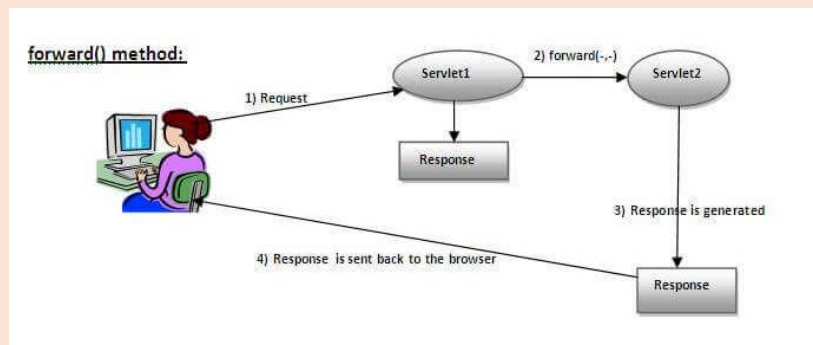
1. Request Dispatcher:-

- The RequestDispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp. This interface can also be used to include the content of another resource also. It is one of the way of servlet collaboration.
- There are two methods defined in the RequestDispatcher interface.

• Methods of RequestDispatcher interface:-

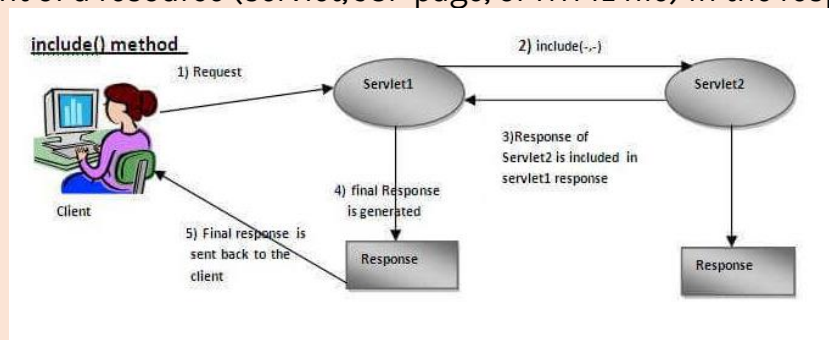
a) **public void forward(ServletRequest request, ServletResponse response)throws ServletException,java.io.IOException**

Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.



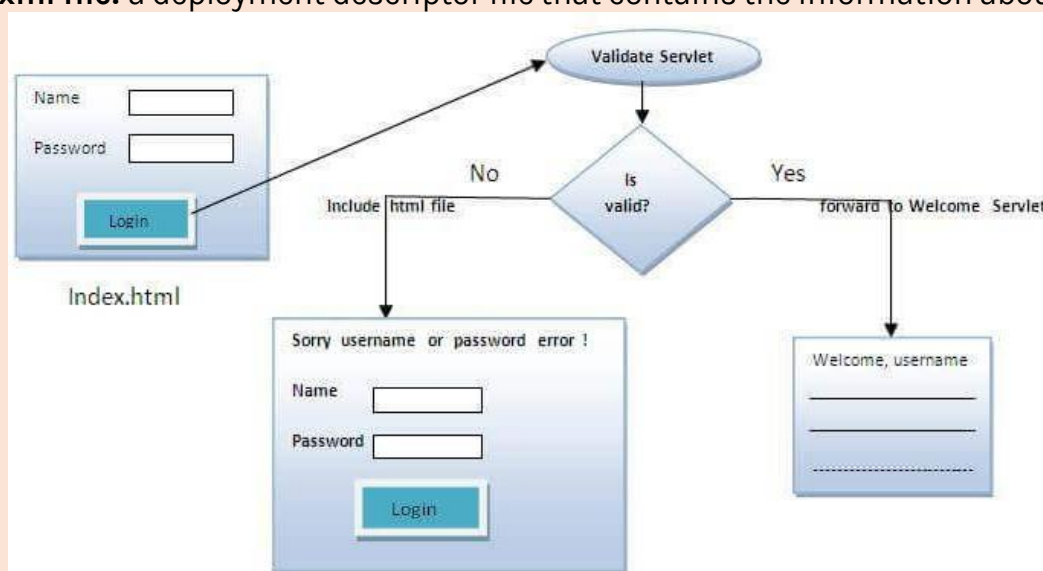
b) **public void include(ServletRequest request, ServletResponse response)throws ServletException,java.io.IOException**

Includes the content of a resource (servlet, JSP page, or HTML file) in the response.



Example -

- **index.html file:** for getting input from the user.
- **Login.java file:** a servlet class for processing the response. If password is servet, it will forward the request to the welcome servlet.
- **WelcomeServlet.java file:** a servlet class for displaying the welcome message.
- **web.xml file:** a deployment descriptor file that contains the information about the servlet.

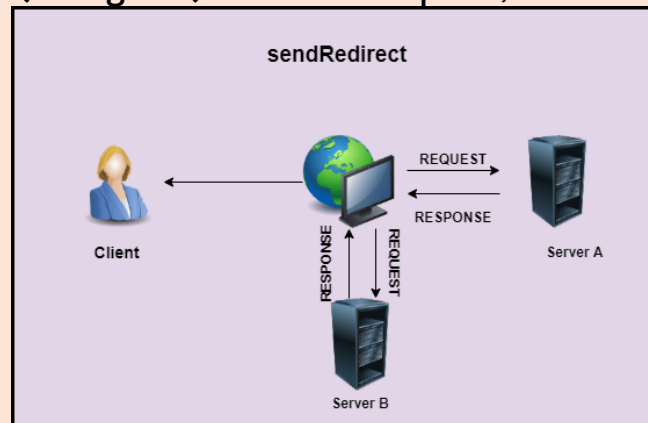


- **Send Redirect:-**

- The **sendRedirect()** method of **HttpServletResponse** interface can be used to redirect response to another resource, it may be **servlet, jsp** or **html file**.
- It accepts **relative** as well as **absolute URL**.
- It works at client side because it uses the url bar of the browser to make another request. So, it can work inside and outside the server.
- Send Redirect is method which send the response to the next servlet that its connect to other servlet .

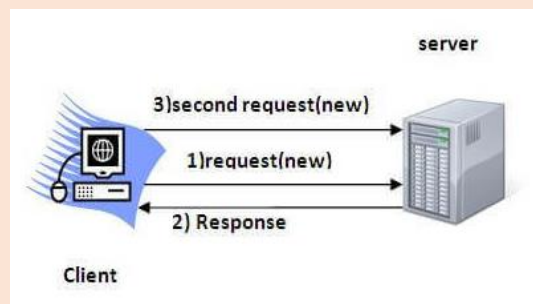
- **Syntax of SendRedirect():-**

public void sendRedirect(String URL)throws IOException;



- **Session Management :-**

- In simpler terms, a session is a state consisting of several requests and response between the client and the server.
- It is a known fact that HTTP and Web Servers are both stateless. Hence, the only way to maintain the state of the user is by making use of technologies that implement session tracking.

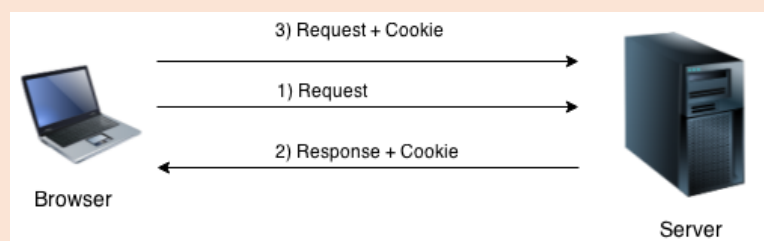


- **Techniques to handle Session Management:-**

- Cookies
- HttpSession

- **Cookies:-**

A cookie is a small piece of information that is persisted between the multiple client requests. By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.

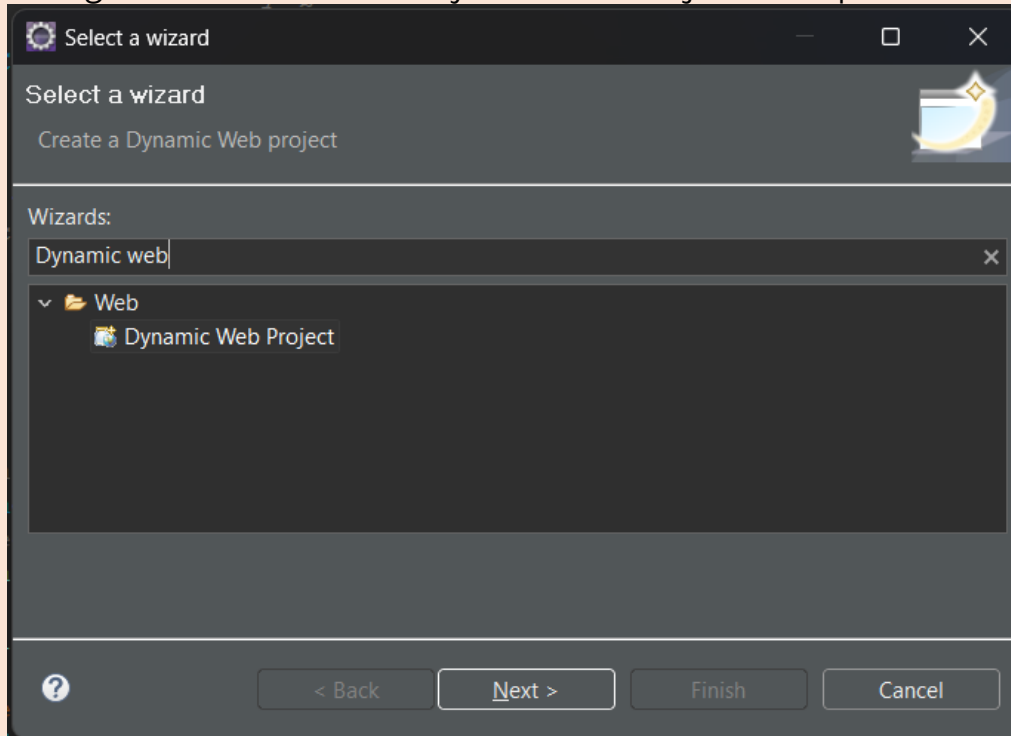


- **Setting up for Coding:-**

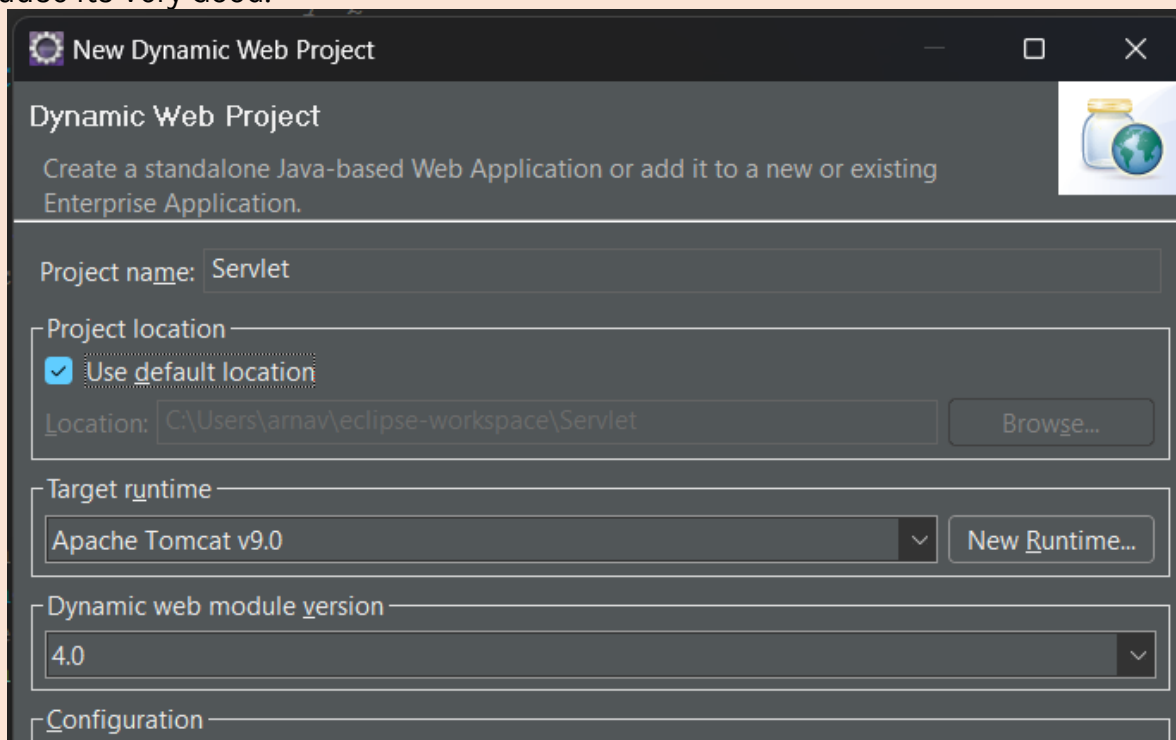
- First we Install the Apache Tomcat Server in our System/PC/Laptops.
- I am using apache-tomcat-9.0.73 version.



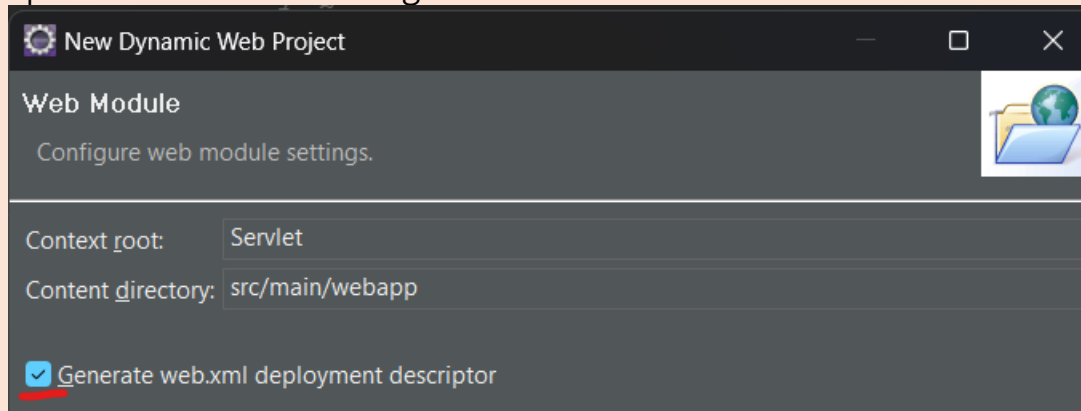
- For Start Coding We have to Select the Dynamic Web Project in Eclipse IDE.



- Enter the name of the Dynamic Web Project and Select the Tomcat-version, I prefer v9.0 because its Very Good.



- Make Sure you Always Select the Generate Web.xml Deployment Descriptor Otherwise It may Create a problem in the Servlet Programs



• **JSP (Java Server Pages):-**

- 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.
- JSP tags are used to insert Java code into HTML pages.
- It is an advanced version of Servlets.
- In this, java code can be inserted in HTML/XML pages or both.
- JSP is first converted into servlet by JSP container before processing the client's request.

• **Advantages JSP over Servlets:-**

- Easy to maintain.
- No recompilation or redeployment is required.
- JSP has accessed 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 helps user to interact in real time environment.
- **Portable, Powerful, flexible and easy to maintain:** - as these are browser and server independent.
- **No Redeployment and No Re-Compilation:** - It is dynamic, secure and platform independent so no need to re-compilation.
- **Extension to Servlet:** - as it has all features of servlets, implicit objects and custom tags

• **Advantages of using JSP: -**

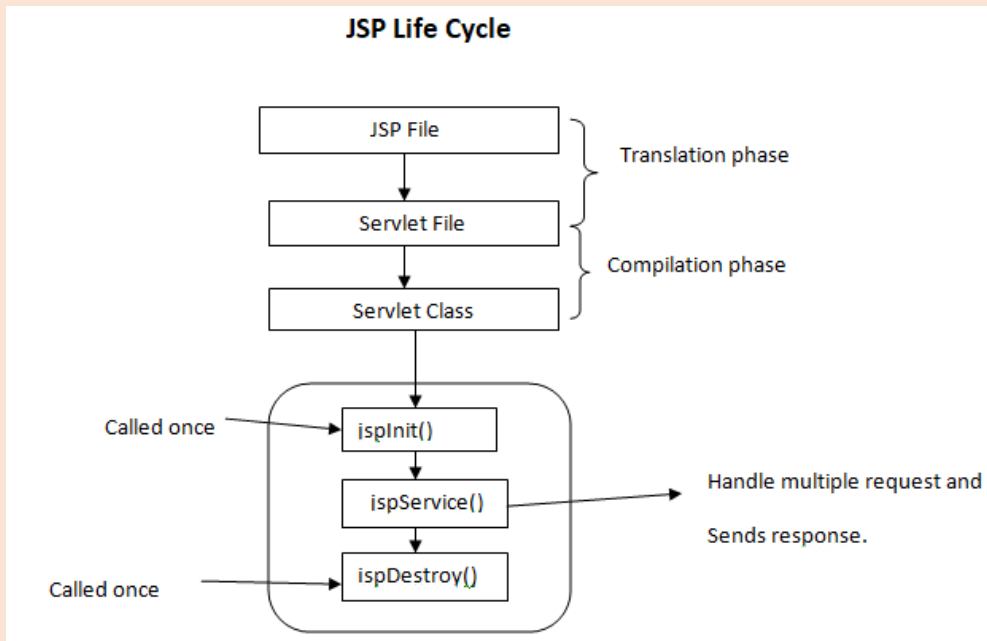
- It does not require advanced knowledge of JAVA.
- It is capable of handling exceptions.
- Easy to use and learn.
- It can tags which are easy to use and understand.
- Implicit objects are there which reduces the length of code.
- It is suitable for both JAVA and non-JAVA programmer

• **Disadvantages of using JSP: -**

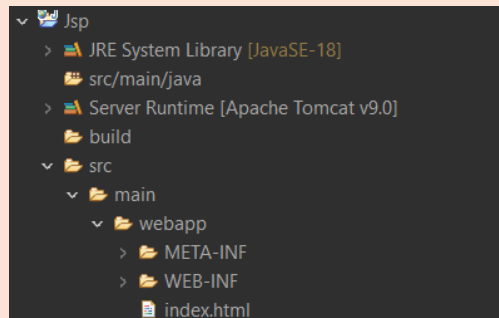
- Difficult to debug for errors.
- First time access leads to wastage of time
- Its output is HTML which lacks features.

• JSP life cycle: -

A Java Server Page life cycle is defined as the process that started with its creation which later translated to a servlet and afterward servlet lifecycle comes into play. This is how the process goes on until its destruction.



- After you Setup The whole package is look like this and the **Web.xml** file Present in the **WEB-INF**.



for this we have to create the dynamic web project (while creating).

- We do **Html coding stuff** in the **webapp** Folder of Our Dynamic Web Project.

```

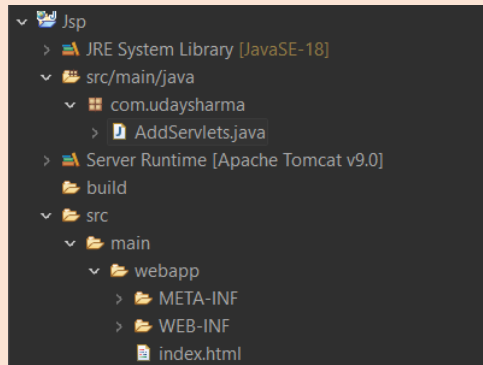
index.html x
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Into of JSP and Servlets</title>
6 </head>
7 <body>
8 <h1>My name is uday Sharma am doing Btect CSE and Now Learning JSP and Servlets</h1>
9 <form action="ADD">
10 Enter the 1st number <input type="Text" name="num1"><br>
11 Enter the 2nd number <input type="Text" name="num2"><br>
12 <input type="submit">
13 </form>
14 </body>
15 </html>
  
```

- After writing the code of html we execute the code which look like in the default web Brower.
My name is uday Sharma am doing Btect CSE and Now Learning JSP and Servlets

Enter the 1st number

Enter the 2nd number

- Now we make another class in this package we create as the reverse of the domain name of our domain like I take **"com.udaysharma"** and then create a class name as **'AddServlets'**.



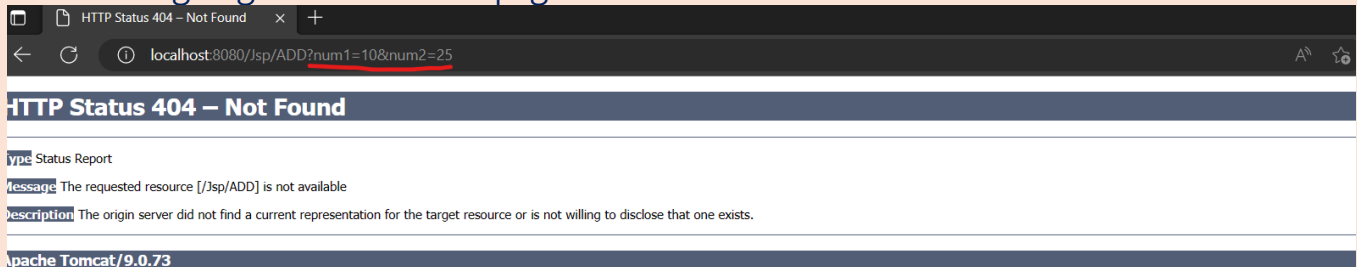
- We need to extend our class to “**HttpServlet**”.
- The code going to look like this. And its take **two objects** of **request** and **response**.

```

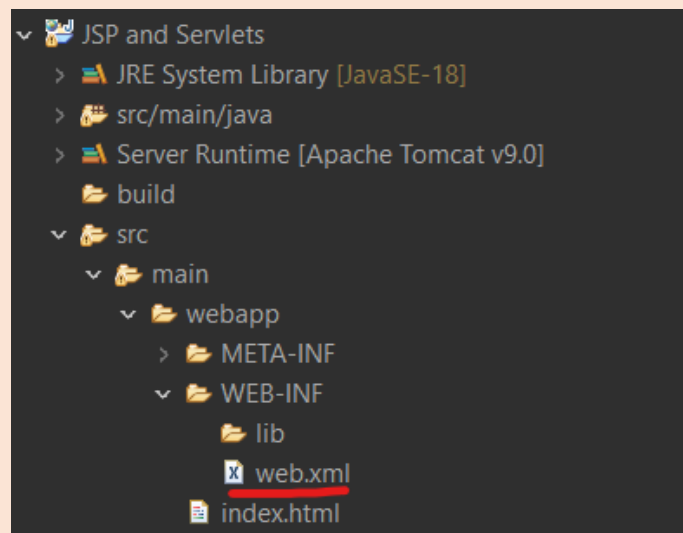
1 package com.udaysharma;
2 // this import is used to help extends the class in HttpServlet
3 import javax.servlet.http.HttpServlet;
4
5
6
7 public class AddServlets extends HttpServlet {
8     // we have to make two objects 1) Request 2) Response objects
9     // using request we can fetch data from the user
10    // and using response object we can give the response
11    public void service(HttpServletRequest req, HttpServletResponse resp) {
12        // we need to convert the String into integer
13        int i=Integer.parseInt(req.getParameter("num1"));
14        int j=Integer.parseInt(req.getParameter("num2"));
15        // performing the add operation
16        int k=i+j;
17        System.out.println("the result is :"+k);
18    }
19    public static void main(String[] args) {
20        // TODO Auto-generated method stub
21    }
22 }

```

- Now we going to run our Html page.



- The value we entered is passed we can see the under the **red line**.
- we need to **remove** this **error page** we have to **modified** the **xml page** present in the **WEB-INF**.



- we need to enter the **full address** of the class.


```

1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jcp.org/xml/ns/javaee"
3 <servlet>
4 <servlet-name>abc</servlet-name>
5 <servlet-class>com.udaysharma.AddServlets</servlet-class>
6 </servlet>
7 <servlet-mapping>
8 <servlet-name>abc</servlet-name>
9 <url-pattern>/add</url-pattern>
10 </servlet-mapping>
11 </web-app>

```

- After compiling it:- and entering the values 22 and 8 in num1 and num2:-

My name is uday Sharma am doing Btech CSE and Now Learning JSP and Servlets

Enter the 1st number
 Enter the 2nd number

- we land on an empty page like this: -

localhost:8080/JSP_and_Servlets/add?num1=22&num2=8

- Because we **get** the **output** at the **console** of the **Eclipse IDE Console**: -

```

Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe (Mar 21, 2023, 4:28:11 PM) [pid: 23028]
INFO: Starting Servlet engine: [Apache Tomcat/9.0.73]
Mar 21, 2023 4:28:13 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-nio-8080"]
Mar 21, 2023 4:28:13 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in [751] milliseconds
the result isn :30

```

- So we have to **modified** it that it **can show** on the **Client machine**, for this we using by the importing the ***java.io.PrintWriter***;
- We **providing** the **response** with the help of ***"response_object".getwriter()***.

Code:-

```

1 package com.udaysharma;
2 //this import is used to help extends the class in HttpServlet
3 import javax.servlet.http.HttpServlet;
4 import javax.servlet.http.HttpServletRequest;
5 import javax.servlet.http.HttpServletResponse;
6 import java.io.IOException;
7 // for printing the output on the client machine we using
8 import java.io.PrintWriter;
9 public class AddServlets extends HttpServlet {
10     // we have to make two objects 1) Request 2) Response objects
11     // using request we can fetch data from the user
12     //and using response object we can give the response
13     public void service(HttpServletRequest req, HttpServletResponse resp) throws IOException {
14         // we need to convert the String into integer
15         int i=Integer.parseInt(req.getParameter("num1"));
16         int j=Integer.parseInt(req.getParameter("num2"));
17         // performing the add operation
18         int k=i+j;
19         // we creating an object of PrintWriter
20         PrintWriter out=resp.getWriter();
21         out.println("the result isn :"+k);
22     }
23 }

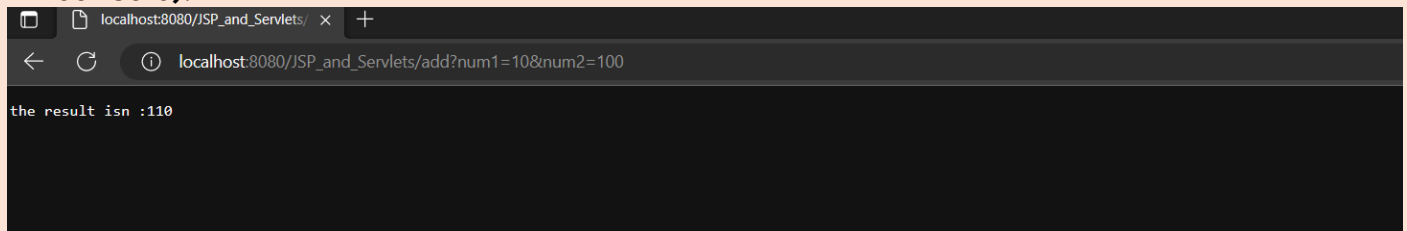
```

- We get the exception in the **printWriter** which is an **IO Exception** . So we just have to throw the exception -by **throwing** the exception of the class **AddServlets** or we can use the **try-catch method**.
- In this am using the **throw exception**.

My name is uday Sharma am doing Btect CSE and Now Learning JSP and Servlets

Enter the 1st number
 Enter the 2nd number

- As we can the **value entered** in the **client server/frontend** that **num1=10** and **num2=100** , and **this time we get the output on the client side not on the console(no output on console)**.

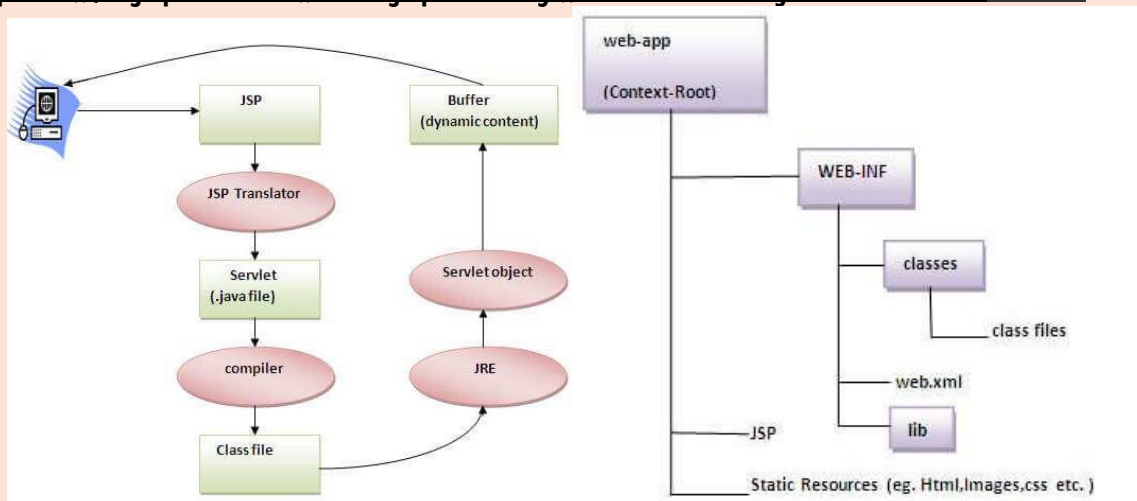


- As we can **see the num1 and num2 value** in the **url** Because is **Get HTTPRequest** by default:-

• ***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).

Note: jspInit(), _jspService() and jspDestroy() are the life cycle methods of JSP.

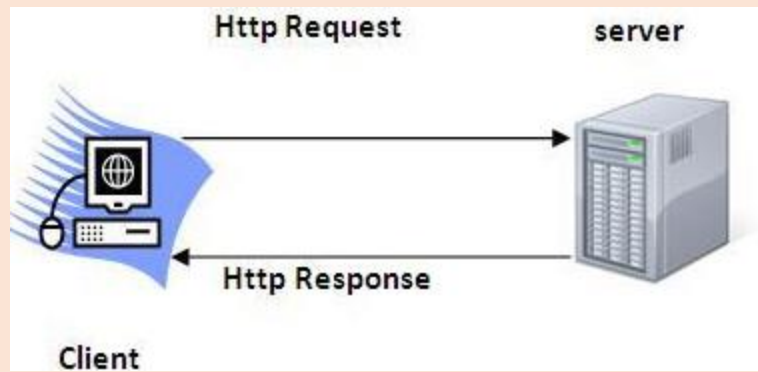


As depicted in the above diagram, JSP page is translated into Servlet by the help of JSP translator. The JSP translator is a part of the web server which is responsible for translating the JSP page into Servlet. After that, Servlet page is compiled by the compiler and gets converted into the class file.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

• ***HTTP (Hyper Text Transfer Protocol) :-***

The Hypertext Transfer Protocol (HTTP) is application-level protocol for collaborative, distributed, hypermedia information systems. It is the data communication protocol used to establish communication between client and server.

HTTP is TCP/IP based communication protocol, which is used to deliver the data like image files, query results, HTML files etc on the World Wide Web (WWW) with the default port is TCP 80. It provides the standardized way for computers to communicate with each other.



⇒ **The Basic Characteristics of HTTP (Hyper Text Transfer Protocol):-**

- It is the protocol that allows web servers and browsers to exchange data over the web.
- It is a request response protocol.
- It uses the reliable TCP connections by default on TCP port 80.
- It is stateless means each request is considered as the new request. In other words, server doesn't recognize the user by default.

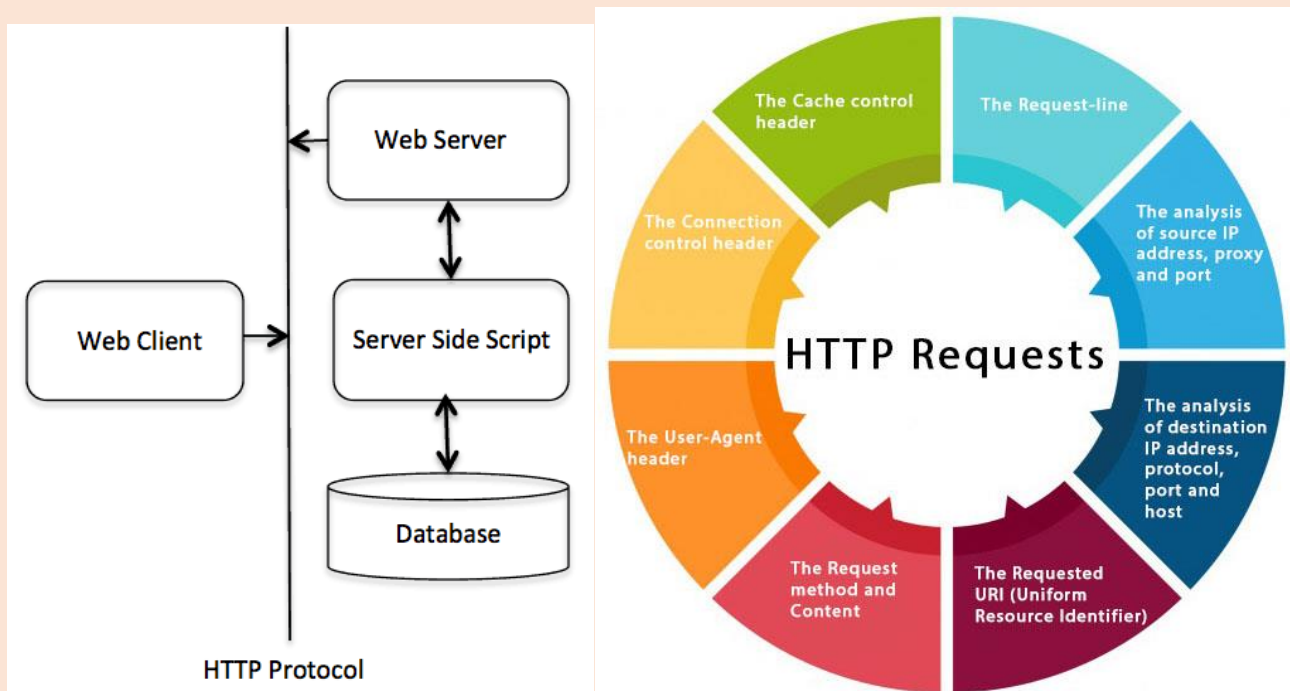
⇒ **The Basic Features of HTTP (Hyper Text Transfer Protocol):-**

There are three fundamental features that make the HTTP a simple and powerful protocol used for communication: -

- **HTTP is media independent:** - It specifies that any type of media content can be sent by HTTP as long as both the server and the client can handle the data content.
- **HTTP is connectionless:** -It is a connectionless approach in which HTTP client i.e., a browser initiates the HTTP request and after the request is sent the client disconnects from server and waits for the response.
- **HTTP is stateless:** --The client and server are aware of each other during a current request only. Afterwards, both of them forget each other. Due to the stateless nature of protocol, neither the client nor the server can retain the information about different request across the web pages.

• **The Basic Architecture of HTTP (Hyper Text Transfer Protocol):-**

The below diagram represents the basic architecture of web application and depicts where HTTP stands:



HTTP is request/response protocol which is based on client/server based architecture. In this protocol, web browser, search engines, etc. behave as HTTP clients and the Web server like Servlet behaves as a server

• **HTTP Requests:-**

The **request sent by the computer** to a **web server**, contains all sorts of potentially interesting information; it is known as **HTTP requests**.

The **HTTP client sends** the request to the server in the form of request message which includes following information:

- The Request-line.
- The analysis of source IP address, proxy and port..
- The analysis of destination IP address, protocol, port and host.
- The Requested URI (Uniform Resource Identifier).
- The Request method and Content.
- The User-Agent header.
- The Connection control header.
- The Cache control header.
- The **HTTP request method indicates** the method to be performed on the resource identified by the **Requested URI (Uniform Resource Identifier)**. This method **is case-sensitive** and should be used in **uppercase**.

The HTTP request methods are:

HTTP Request	Description
GET	Asks to get the resource at the requested URL.
POST	Asks the server to accept the body info attached. It is like GET request with extra info sent with the request.
HEAD	Asks for only the header part of whatever a GET would return. Just like GET but with no body.
TRACE	Asks for the loopback of the request message, for testing or troubleshooting.
PUT	Says to put the enclosed info (the body) at the requested URL.
DELETE	Says to delete the resource at the requested URL.
OPTIONS	Asks for a list of the HTTP methods to which the thing at the request URL can respond

- **Note:-**By default we use the **GetRequest** because our values goes to address bar.
- **GetRequest** is used to get the data from the server.

• ***Post HTTPsrequest:-***

```
index.html x web.xml AddServlets.java
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Into of JSP and Servlets</title>
6 </head>
7 <body>
8 <h1>My name is uday Sharma am doing Btect CSE and Now Learning JSP and Servlets</h1>
9 <form action="add" method='post'>
10 Enter the 1st number <input type="Text" name="num1"><br>
11 Enter the 2nd number <input type="Text" name="num2"><br>
12 <input type="submit">
13 </form>
14 </body>
15 </html>
```

- When we use the **postRequest** we see that the our **url** is clean because post request is used to **sent a request** or **submit data** to the **server** to accept its data.

Like this:-

localhost:8080/JSP_and_Servlets/index.html

My name is uday Sharma am doing Btect CSE and Now Learning JSP and Servlets

Enter the 1st number
Enter the 2nd number

• See under the red line that url is pretty clean now:-

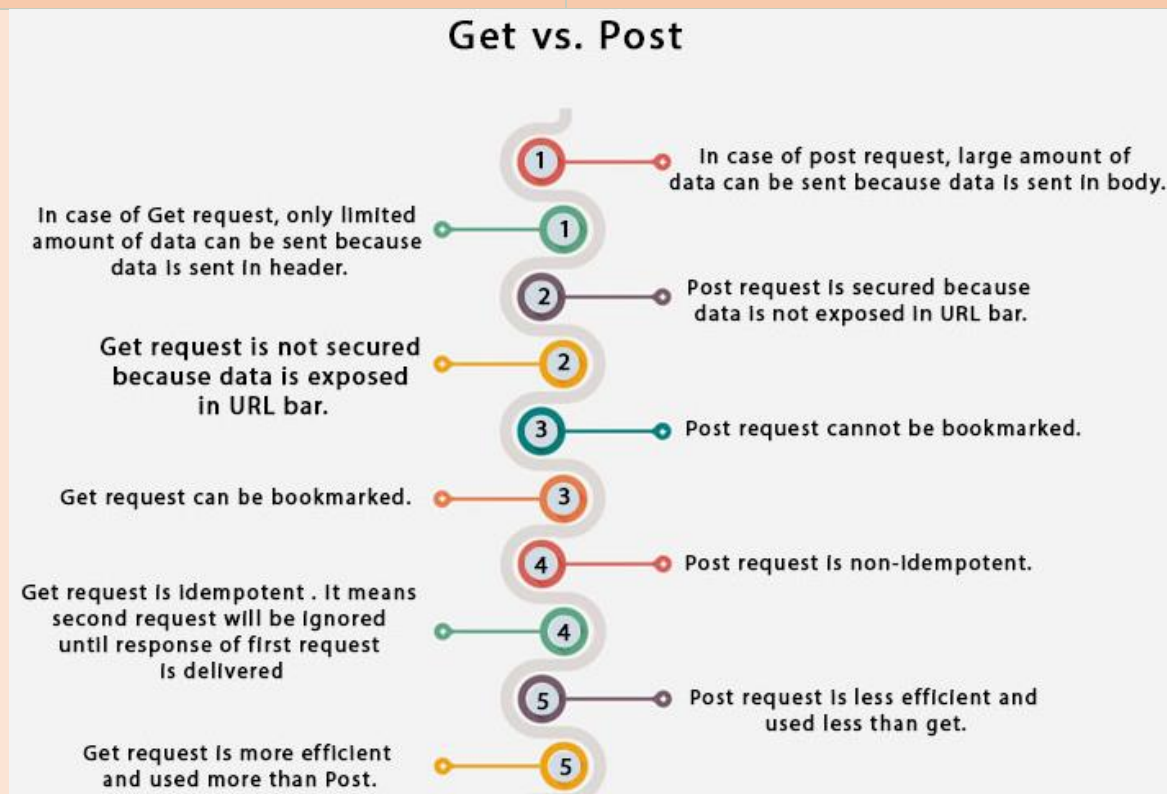
localhost:8080/JSP_and_Servlets/add

the result isn :18

• Get vs. Post

There are many differences between the Get and Post request. Let's see these differences:

GET	POST
1) In case of Get request, only limited amount of data can be sent because data is sent in header.	In case of post request, large amount of data can be sent because data is sent in body.
2) Get request is not secured because data is exposed in URL bar.	Post request is secured because data is not exposed in URL bar.
3) Get request can be bookmarked .	Post request cannot be bookmarked .
4) Get request is idempotent . It means second request will be ignored until response of first request is delivered	Post request is non-idempotent .
5) Get request is more efficient and used more than Post.	Post request is less efficient and used less than get.



• GET and POST:-

Two common methods for the **request-response** between a server and client are:

- **GET**- It requests the data from a specified resource
- **POST**- It submits the processed data to a specified resource
- For this **HTTPServlet** give us two extra method i.e 1) **dopost** 2) **doget** Requests.

Dopost Request :-

- In this programmer design the webapp to get the **post request** . If you send the post request which is **accepted** by the server otherwise you have a **get request** it will give **error** on the client machine or frontend side:-

Code:- it will only take a post request;-

```
index.html  web.xml  AddServlets.java x
1 package com.udaysharma;
2 //this import is used to help extends the class in HttpServlet
3 import javax.servlet.http.HttpServlet;
4
9 public class AddServlets extends HttpServlet {
10     // we have to make two objects 1) Request 2) Response objects
11     // using request we can fetch data from the user
12     //and using response object we can give the response
13     public void dopost(HttpServletRequest req, HttpServletResponse resp) throws IOException {
14         // we need to convert the String into integer
15         int i=Integer.parseInt(req.getParameter("num1"));
16         int j=Integer.parseInt(req.getParameter("num2"));
17         // performing the add operation
18         int k=i+j;
19         // we creating an object of PrintWriter
20         PrintWriter out=resp.getWriter();
21         out.println("the result isn :"+k);
22     }
23 }
```

- Our **AddServlet.java** method is **Set on Dopost method** so its only accept the **Dopost Request**

```
index.html x  web.xml  AddServlets.java
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Intro of JSP and Servlets</title>
6 </head>
7 <body>
8 <h1>My name is uday Sharma am doing Btect CSE and Now Learning JSP and Servlets</h1>
9 <form action="add" method='post'>
10 Enter the 1st number <input type="Text" name="num1"><br>
11 Enter the 2nd number <input type="Text" name="num2"><br>
12 <input type="submit">
13 </form>
14 </body>
15 </html>
```

- Its give not error but if we use **method='get'** instead of **post** its give error because we set the page to accept only **postrequest** .
- Now we use **getrequest**:-

```
index.html x  web.xml  AddServlets.java
1 <!DOCTYPE html>
2 <html>
3 <head>
4 <meta charset="UTF-8">
5 <title>Intro of JSP and Servlets</title>
6 </head>
7 <body>
8 <h1>My name is uday Sharma am doing Btect CSE and Now Learning JSP and Servlets</h1>
9 <form action="add" method='get'>
10 Enter the 1st number <input type="Text" name="num1"><br>
11 Enter the 2nd number <input type="Text" name="num2"><br>
12 <input type="submit">
13 </form>
14 </body>
15 </html>
```

- Its give error like this :-

HTTP Status 405 – Method Not Allowed

Type Status Report

Message HTTP method GET is not supported by this URL

Description The method received in the request-line is known by the origin server but not supported by the target resource.

Apache Tomcat/9.0.73

Note:- we can use both the method in our webapp.

Now making a another class name as Square Servlet which gives the square of the number of the k which is i+j;

- We can call the another servlets using the two methods:-
 - ***Request Dispatcher:-***
- (request).Dispatcher rd=req.getRequestDispatcher("Servlet_name");

```

1 package com.udaysharma;
2 import javax.servlet.RequestDispatcher;
3
4 public class AddServlets extends HttpServlet {
5     // we have to make two objects 1) Request 2) Response objects
6     // using request we can fetch data from the user
7     // and using response object we can give the response
8     public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {
9         // we need to convert the String into integer
10        int i=Integer.parseInt(req.getParameter("num1"));
11        int j=Integer.parseInt(req.getParameter("num2"));
12        // performing the add operation
13        int k=i+j;
14        k=k*k;
15        // we can call the servlet we have to method
16        // 1) Request Dispatcher
17        // 2) redirecting
18        // using the request dispatcher
19        req.setAttribute("k", k);
20        RequestDispatcher rd=req.getRequestDispatcher("sq");
21        rd.forward(req, resp);
22    }
23 }
24
25 package com.udaysharma;
26
27 import java.io.IOException;
28
29 public class Square_servlet extends HttpServlet {
30     public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException{
31         PrintWriter out=resp.getWriter();
32         int k=(int)req.getAttribute("k");
33         k=k*k;
34         out.print("hello to square "+k);
35     }
36 }

```


- So we Use the Request Dispatcher it send the data to the “sq” which is ***SquareServlet.java***.

```

web.xml x *AddServlets.java Square_servlet.java index.html
http://xmlns.jcp.org/xml/ns/javaee/web-app_4_0.xsd (xsi:schemaLocation with catalog)
1 <?xml version="1.0" encoding="UTF-8"?>
2 <web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://xmlns.jc
3 <servlet>
4 <servlet-name>abc</servlet-name>
5 <servlet-class>com.udaysharma.AddServlets</servlet-class>
6 </servlet>
7 <servlet-mapping>
8 <servlet-name>abc</servlet-name>
9 <url-pattern>/add</url-pattern>
10 </servlet-mapping>
11
12 <servlet>
13 <servlet-name>pqr</servlet-name>
14 <servlet-class>com.udaysharma.Square_servlet</servlet-class>
15 </servlet>
16 <servlet-mapping>
17 <servlet-name>pqr</servlet-name>
18 <url-pattern>/sq</url-pattern>
19 </servlet-mapping>
20 </web-app>

```

• **SendRedirect:-**

- We going to different servlet like we do above but this time by using the **SendRedirect** method.

```

*AddServlets.java x index.html web.xml Square_servlet.java
1 package com.udaysharma;
2 import java.io.IOException;
3 import javax.servlet.RequestDispatcher;
4 import javax.servlet.ServletException;
5 import javax.servlet.http.HttpServlet;
6 import javax.servlet.http.HttpServletRequest;
7 import javax.servlet.http.HttpServletResponse;
8 public class AddServlets extends HttpServlet {
9     // we have to make two objects 1) Request 2) Response objects
10    // using request we can fetch data from the user
11    //and using response object we can give the response
12    public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {
13        // we need to convert the String into integer
14        int i=Integer.parseInt(req.getParameter("num1"));
15        int j=Integer.parseInt(req.getParameter("num2"));
16        // performing the add operation
17        int k = i+j+1;
18        // PrintWriter out=resp.getWriter();
19        // out.println("<html><body bgcolor='color'>");
20        // out.println("Output : "+k);
21        // out.println("</body></html>");
22        // k=k*k;
23        // we can call the servlet we have to method
24        // 1) Request Dispatcher
25        // 2) redirecting
26        /* using the request dispatcher */
27        // req.setAttribute("k", k);
28        // RequestDispatcher rd=req.getRequestDispatcher("sq");
29        // rd.forward(req, resp);
30        /* Response redirect method*/
31        resp.sendRedirect("sq");
32    }
33 }

```

- After Restarting the Server we go to Our Webpage and its look like This :-

localhost:8080/JSP_and_Servlets/index.html

My name is uday Sharma am doing Btect CSE and Now Learning JSP and Servlets

Enter the 1st number

Enter the 2nd number

- After **Entering the value** of the num1 and num2 and **submit** it We get the **null point Exception** because our “AddServlet.java” file takes the value **sendRedriecting** to the “sq” Servlet which is “Square_servlet.java” file.

← ↻ ⓘ localhost:8080/JSP_and_Servlets/sq

HTTP Status 500 ? Internal Server Error

Type Exception Report

Message Cannot invoke "java.lang.Integer.intValue()" because the return value of "javax.servlet.http.HttpServletRequest.getAttribute(String)" is null

Description The server encountered an unexpected condition that prevented it from fulfilling the request.

Exception

```
java.lang.NullPointerException: Cannot invoke "java.lang.Integer.intValue()" because the return value of "javax.servlet.http.HttpServletRequest.getAttribute(String)" is null
    com.udaysharma.Square_servlet.doGet(Square_servlet.java:13)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:502)
    javax.servlet.http.HttpServlet.service(HttpServlet.java:596)
    org.apache.tomcat.websocket.server.WsFilter.doFilter(WsFilter.java:53)
```

Note The full stack trace of the root cause is available in the server logs.

Apache Tomcat/9.0.73

- After the modifying the “Square_servlet.java” code look like this:-

```
AddServlets.java  index.html  web.xml  Square_servlet.java ×
1 package com.udaysharma;
2
3 import java.io.IOException;
4
5
6
7
8
9
10 public class Square_servlet extends HttpServlet {
11     public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {
12         // PrintWriter out=resp.getWriter();
13         //int k=(int) req.getAttribute("k");
14         //out.print("hello to submation of number "+k);
15         //k=k*k;
16         // out.print("hello to squire "+k);
17         System.out.println("this is Sq colled");
18     }
19 }
```

- Again after compiling and restart the serve and entering the values and submit.

← ↻ ⓘ localhost:8080/JSP_and_Servlets/index.html

My name is uday Sharma am doing Btect CSE and Now Learning JSP and Servlets

Enter the 1st number

Enter the 2nd number

- We land on the blank page we no output but we can see in the url we are on the “sq” which is “Square_servlet.java” file

localhost:8080/JSP_and_Servlets/ ×

← ↻ ⓘ localhost:8080/JSP_and_Servlets/sq

- As we didn't use the **printwriter** we **didn't get the output on our page** instead we use the **System.out.println()** and we get the output in the console of eclipse IDE.

```
Console ×
Tomcat v9.0 Server at localhost [Apache Tomcat] C:\Program Files\Java\jdk-18.0.2\bin\javaw.exe (Mar 31, 2023, 7:43:04 PM) [pid: 26640]
INFO: Initializing ProtocolHandler ["http-nio-8080"]
Mar 31, 2023 7:43:05 PM org.apache.catalina.startup.Catalina load
INFO: Server initialization in [804] milliseconds
Mar 31, 2023 7:43:05 PM org.apache.catalina.core.StandardService startInternal
INFO: Starting service [Catalina]
Mar 31, 2023 7:43:05 PM org.apache.catalina.core.StandardEngine startInternal
INFO: Starting Servlet engine: [Apache Tomcat/9.0.73]
Mar 31, 2023 7:43:06 PM org.apache.coyote.AbstractProtocol start
INFO: Starting ProtocolHandler ["http-nio-8080"]
Mar 31, 2023 7:43:06 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in [722] milliseconds
this is Sq colled
```

- So we write on the url..
- So ,again after modifying the “sq” means “Square_servlet.java” using the **PrintWriter** instead of **System.out.println()** and save and restart the server.

Code:-

- **AddServlet.java:-**

```

AddServlets.java x index.html web.xml Square_servlet.java
10 // using request we can fetch data from the user
11 //and using response object we can give the response
12 public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {
13     // we need to convert the String into integer
14     int i=Integer.parseInt(req.getParameter("num1"));
15     int j=Integer.parseInt(req.getParameter("num2"));
16     // performing the add operation
17     int k = i+j+1;
18     // PrintWriter out=resp.getWriter();
19     // out.println("<html><body bgcolor='color'>");
20     // out.println("Output : "+k);
21     // out.println("</body></html>");
22     // k=k*k;
23     // we can call the servlet we have to method
24     // 1) Request Dispatcher
25     // 2) redirecting
26     /* using the request dispatcher */
27     // req.setAttribute("k", k);
28     // RequestDispatcher rd=req.getRequestDispatcher("sq");
29     // rd.forward(req, resp);
30     /* Response redirect method*/
31     resp.sendRedirect("sq");
32 }
33 }
34

```

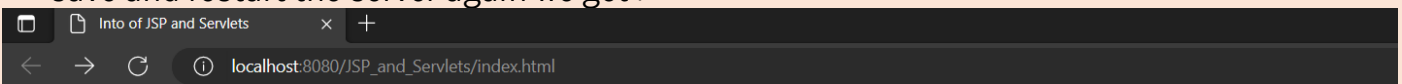
- **Square_servlet.java:-**

```

AddServlets.java index.html web.xml Square_servlet.java x
1 package com.udaysharma;
2
3 import java.io.IOException;
4
5
6
7
8
9
10 public class Square_servlet extends HttpServlet {
11     public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException{
12         /* this is used for the RequestDispatcher */
13         PrintWriter out=resp.getWriter();
14         //int k=(int)req.getAttribute("k");
15         //out.print("hello to submation of number "+k);
16         //k=k*k;
17         // out.print("hello to squre "+k);
18         // System.out.println("this is Sq colled");
19
20         /* this is used for the SendRedirect */
21         int k=Integer.parseInt(req.getParameter("k"));
22         out.print("hello to submation of number "+k);
23         k=k*k;
24         out.print("hello to squre "+k);
25         System.out.println("this is Sq colled");
26     }
27 }

```

- save and restart the server again we get :-

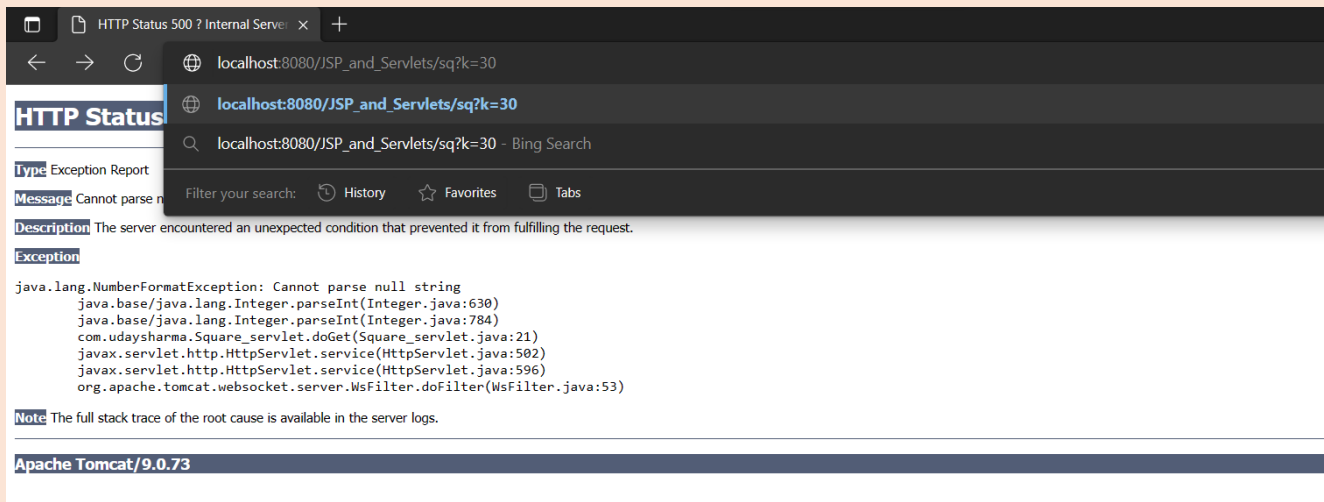


My name is uday Sharma am doing Btect CSE and Now Learning JSP and Servlets

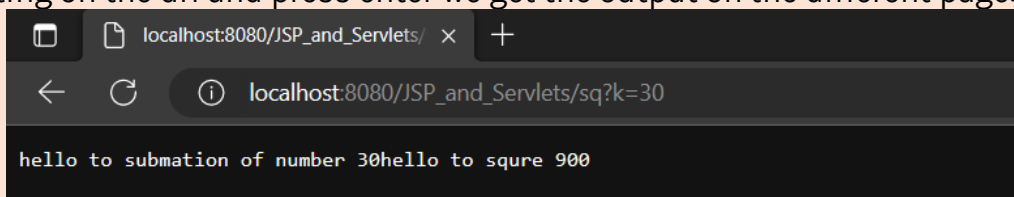
Enter the 1st number

Enter the 2nd number

- After **Entering the value** of the num1 and num2 and
- We write on the url This called the **url writing** we write **"sq?k=30"** on the url to tell the value to Sq servlet or relocate the **Square_servlet.java** file.



- After Writing on the url and press enter we get the output on the different page:-



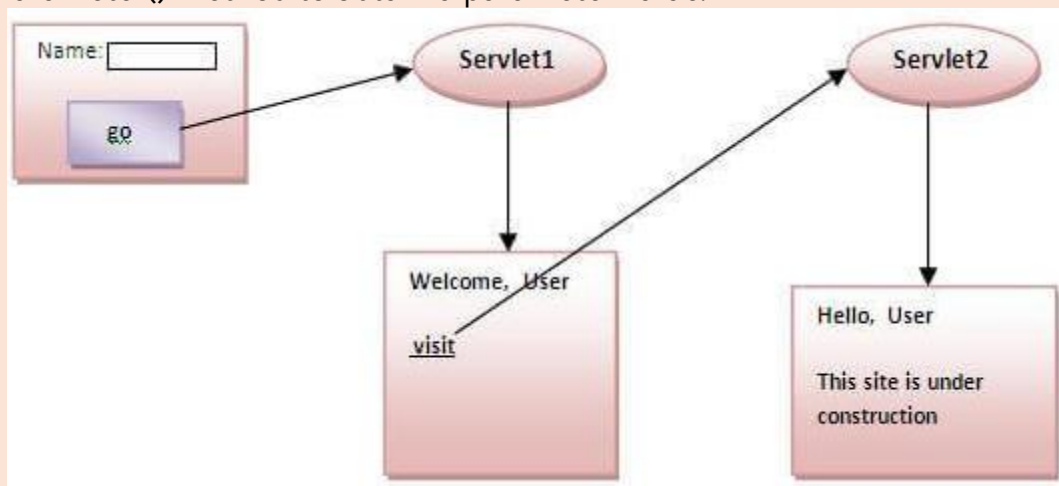
- In this how we can the data using the **sendredirect** method?

• Url Writing:-

- The other method **without** url writing to send the request to the **next servlet**.
- 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 is 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:-

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

• Disadvantage of URL Rewriting:-

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

Example of the Url Writing is :-

In this by using the **url Writing** we **transfer** the **data** from **one servlet to another** servlet by **writing** the **value** in the **url** of the **servlet** in which data transferred:-

- "AddServlet.java"

```

AddServlets.java x index.html x web.xml x Square_servlet.java
11 //and using response object we can give the response
12 public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {
13 // we need to convert the String into integer
14 int i=Integer.parseInt(req.getParameter("num1"));
15 int j=Integer.parseInt(req.getParameter("num2"));
16 // performing the add operation
17 int k = i+j+1;
18 // PrintWriter out=resp.getWriter();
19 // out.println("<html><body bgcolor='color'>");
20 // out.println("Output : "+k);
21 // out.println("</body></html>");
22 // k=k*k;
23 // we can call the servlet we have to method
24 // 1) Request Dispatcher
25 // 2) redirecting
26 /* using the request dispatcher */
27 // req.setAttribute("k", k);
28 // RequestDispatcher rd=req.getRequestDispatcher("sq");
29 // rd.forward(req, resp);
30 /* Response redirect method*/
31 resp.sendRedirect("sq?k="+k);
32 }
33 }

```

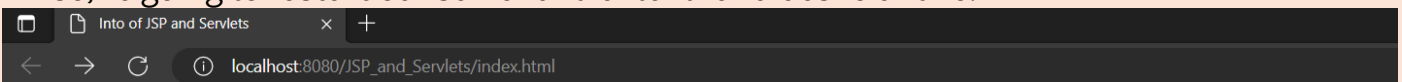
- **"Square_servlet.java"**

```

AddServlets.java x index.html x web.xml x Square_servlet.java x
1 package com.udaysharma;
2
3 import java.io.IOException;
4
5
6
7
8
9
10 public class Square_servlet extends HttpServlet {
11 public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException{
12 /* this is used for the RequestDispatcher */
13 PrintWriter out=resp.getWriter();
14 //int k=(int)req.getAttribute("k");
15 //out.print("hello to submation of number "+k);
16 //k=k*k;
17 // out.print("hello to squire "+k);
18 // System.out.println("this is Sq colled");
19
20 /* this is used for the SendRedirect */
21 int k=Integer.parseInt(req.getParameter("k"));
22 out.print("hello to submation of number "+k);
23 k=k*k;
24 out.print("hello to squire "+k);
25 System.out.println("this is Sq colled");
26 }
27 }

```

- So, we going to **restart our server** and enter the values 10 and 10.

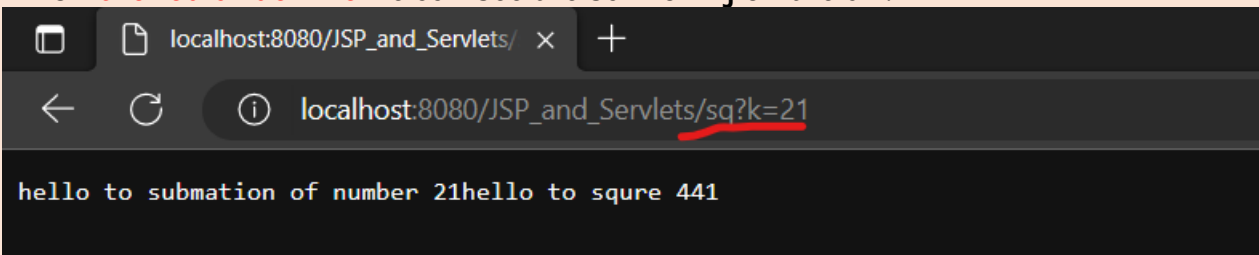


My name is uday Sharma am doing Btect CSE and Now Learning JSP and Servlets

Enter the 1st number

Enter the 2nd number

- On **the red underline** we can see the sum of **i+j** on the url.

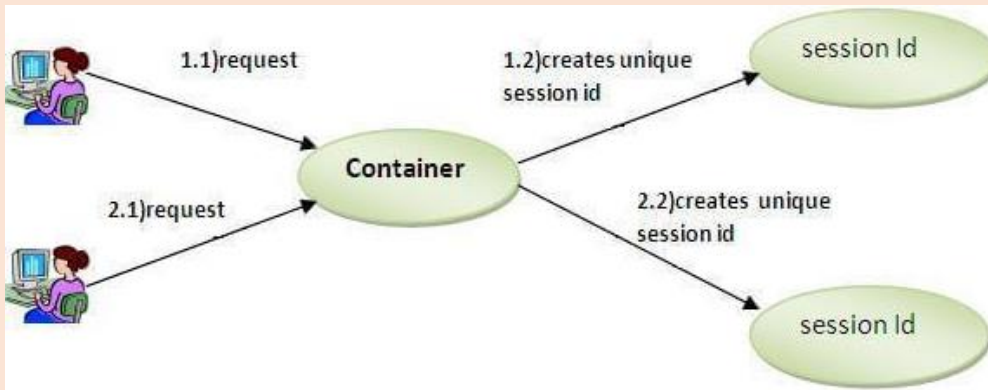


- **HttpSession:-**

In such case, container creates a session id for each user. The container uses this id to identify the particular user. An object of HttpSession can be used to perform two tasks:

1. bind objects

- view and manipulate information about a session, such as the session identifier, creation time, and last accessed time.



- How to get the HttpSession object ?**

The HttpServletRequest interface provides two methods to get the object of HttpSession:

- public HttpSession getSession():** Returns the current session associated with this request, or if the request does not have a session, creates one.
- public HttpSession getSession(boolean create):** Returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session.

- Commonly used methods of HttpSession interface:-**

- public String getId():** Returns a string containing the unique identifier value.
- public long getCreationTime():** Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
- public long getLastAccessedTime():** Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT.
- public void invalidate():** Invalidates this session then unbinds any objects bound to it.

Example of the HTTP Session:-

```
AddServlets.java x index.html web.xml Square_servlet.java
18 int k = 1+j+1;
19 // PrintWriter out=resp.getWriter();
20 // out.println("<html><body bgcolor='color'>");
21 // out.println("Output : "+k);
22 // out.println("</body></html>");
23 // k=k*k;
24 // we can call the servlet we have to method
25 // 1) Request Dispatcher
26 // 2) redirecting
27
28 /* using the request dispatcher */
29 // req.setAttribute("k", k);
30 // RequestDispatcher rd=req.getRequestDispatcher("sq");
31 // rd.forward(req, resp);
32
33 /* Response redirect method*/
34 // resp.sendRedirect("sq?k="+k); // url Rewriting
35
36 /* HttpSession */
37 HttpSession session=req.getSession();
38 session.setAttribute("k", k);
39 resp.sendRedirect("sq");
40 }
41 }
```

- This in Square_servlet.java file which is a another servlet file


```

27  /* HttpSession */
28      HttpSession session=req.getSession();
29      int k=(int)session.getAttribute("k"); // typecasting
30      out.print("hello to submation of number "+k);
31      k=k*k;
32      out.print(" |hello to squre "+k);
33      System.out.println(" this is Sq colled");
34  }
35 }

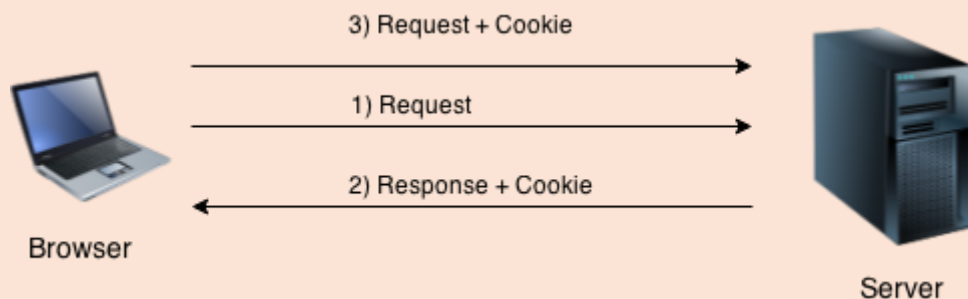
```

• Cookie:-

A **cookie** is a small piece of information that is persisted between the multiple client requests. A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

• How Cookie works

By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.



• Types of Cookie:-

There are 2 types of cookies in servlets.

1. Non-persistent cookie
2. Persistent cookie

• Non-persistent cookie:-

It is **valid for single session** only. It is removed each time when user closes the browser.

• Persistent cookie:-

It is **valid for multiple session**. It is not removed each time when user closes the browser. It is removed only if user logout or sign-out.

• Advantage of Cookies:-

1. Simplest technique of maintaining the state.
2. Cookies are maintained at client side.

• Disadvantage of Cookies:-

1. It will not work if cookie is disabled from the browser.
2. Only textual information can be set in Cookie object.

Note: Gmail uses cookie technique for login. If you disable the cookie, gmail won't work.

• Cookie class:-

`javax.servlet.http.Cookie` class provides the functionality of using cookies. It provides a lot of useful methods for cookies.

• Constructor of Cookie class:-

Constructor	Description
<code>Cookie()</code>	constructs a cookie.
<code>Cookie(String name, String value)</code>	constructs a cookie with a specified name and value.

- **Useful Methods of Cookie class:-**

There are given some commonly used methods of the Cookie class.

Method	Description
public void setMaxAge(int expiry)	Sets the maximum age of the cookie in seconds.
public String getName()	Returns the name of the cookie. The name cannot be changed after creation.
public String getValue()	Returns the value of the cookie.
public void setName(String name)	changes the name of the cookie.
public void setValue(String value)	changes the value of the cookie.

- **Other methods required for using Cookies:-**

For adding cookie or getting the value from the cookie, we need some methods provided by other interfaces. They are:

1. **public void addCookie(Cookie ck)**:method of **HttpServletResponse** interface is used to add cookie in response object.
2. **public Cookie[] getCookies()**:method of **HttpServletRequest** interface is used to return all the cookies from the browser.

- **import** javax.servlet.http.Cookie; // we have to import this for using the cookie

```

35 // resp.sendRedirect( "sq?k="+k); // url rewriting
36
37 /* HttpSession */
38 // HttpSession session=req.getSession();
39 // session.setAttribute("k", k);
40 // resp.sendRedirect("sq");
41 /* Cookie */
42 Cookie cookie = new Cookie("k",k+"");
43 resp.addCookie(cookie);
44 resp.sendRedirect("sq");
45 }
46 }
47

```

- We declare a **cookie** and **make object store** with **value of k** and **add** with our **response** and using the **send redirect** method to **send the request/data** to the **other servlet** name **“sq”/“Square_servlet.java”** file

```

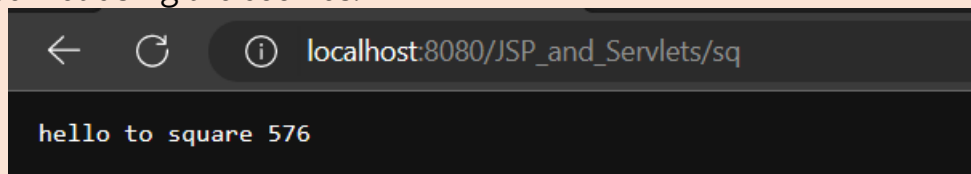
36 /* Cookie */
37 int k=0;
38 Cookie cookies[]=req.getCookies();
39 for(Cookie c:cookies) {
40     if(c.getName().equals("k")) {
41         k=Integer.parseInt(c.getValue());
42     }
43     k=k*k;
44     out.print(" hello to square "+k);
45     System.out.println(" this is Sq colled");
46 }
47 }

```

- Now we save our program and restart the server and enter the num1 and num2 value and submit



- And we can see that get the **output** on the **screen** as we can see in the **url** we are in **sq** mean The **data values** transfer to the “sq” /**Square_servlet.java** file . so this how we can send the request to other servlet using the cookies.



• WebServlet Annotations :-

- The **@WebServlet** annotation is used to declare a **servlet**. The annotated class must extend the **javax.servlet.http.HttpServlet** class.
- So we have to import the **javax.srvler.annotation.WebServlet**;

```

1 package com.udaysharma;
2 import java.io.IOException;
3 import javax.servlet.RequestDispatcher;
4 import javax.servlet.ServletException;
5 import javax.servlet.annotation.WebServlet;
6 import javax.servlet.http.Cookie;
7 import javax.servlet.http.HttpServlet;
8 import javax.servlet.http.HttpServletRequest;
9 import javax.servlet.http.HttpServletResponse;
10 import javax.servlet.http.HttpSession;
11 @WebServlet("/add")
12 public class AddServlets extends HttpServlet {
13     // we have to make two objects 1) Request 2) Response objects
14     // using request we can fetch data from the user
15     //and using response object we can give the response
16     public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException, ServletException {
17         // we need to convert the String into integer
18         int i=Integer.parseInt(req.getParameter("num1"));
19         int j=Integer.parseInt(req.getParameter("num2"));
20         // performing the add operation
21         int k = i+j;

```

• Full AddServlet.java file:-

```

package com.udaysharma;
import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
@WebServlet("/add")
public class AddServlets extends HttpServlet {
    // we have to make two objects 1) Request 2) Response objects
    // using request we can fetch data from the user
    //and using response object we can give the response
    public void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException,
    ServletException {
        // we need to convert the String into integer

```

```



int i=Integer.parseInt(req.getParameter("num1"));
int j=Integer.parseInt(req.getParameter("num2"));
// performing the add operation
int k = i+j;
// PrintWriter out=resp.getWriter();
// out.println("<html><body bgcolor='color'>");
// out.println("Output : "+k);
// out.println("</body></html>");
// k=k*k;
// we can call the servlet we have to method
// 1) Request Dispatcher
// 2) redirecting
/* using the request dispatcher */
// req.setAttribute("k", k);
// RequestDispatcher rd=req.getRequestDispatcher("sq");
// rd.forward(req, resp);
/* Response redirect method*/
// resp.sendRedirect("sq?k="+k); // url Rewriting
/* HttpSession */
// HttpSession session=req.getSession();
// session.setAttribute("k",k);
// resp.sendRedirect("sq");
/* Cookie */
Cookie cookie = new Cookie("k",k+"");
resp.addCookie(cookie);
resp.sendRedirect("sq");
}}

```

In upcoming Notes, Whole JSP is Covered
So, Stay Tuned.



➤ Go check out my [LinkedIn profile](#) for more notes and other resources content

@Uday Sharma
mrudaysharma4600@gmail.com
<https://www.linkedin.com/in/uday-sharma-602b33267>