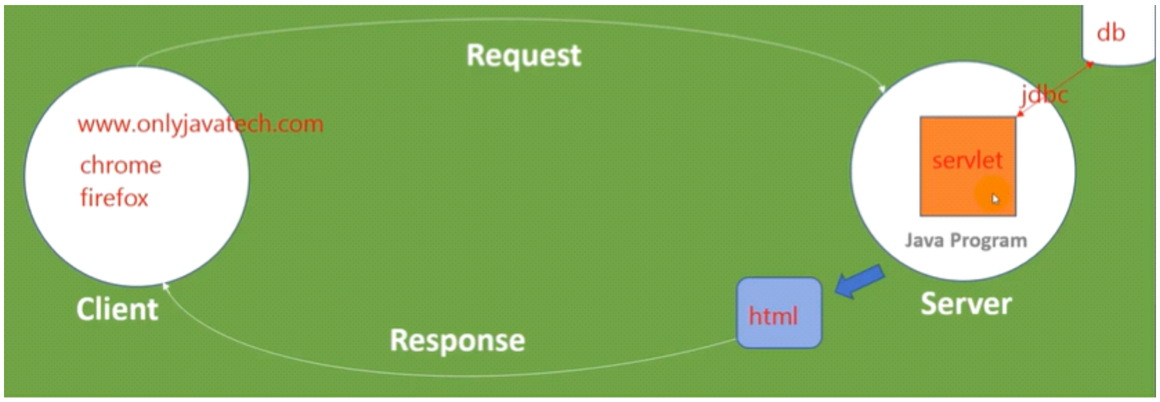
NAME : ROHIT KUMAR PANDEY

SUBJECT : ADVANCE JAVA

Servlet

Servlet is a simple java program that runs on server and capable of handling request and generating dynamic response.



How to create servlet using Servlet Interface

Package : javax.servlet

Interface : Servlet

Servlet Interface methods:

- public abstract void init (javax.servlet.ServletConfig) (Life Cycle)

- public ServletConfig getServletConfig()
- public void service(javax.servlet.ServletRequest,javax.servlet.ServletResponse) (Life Cycle)
- public abstract java.lang.String getServiceInfo()
- public abstract void destroy()   (Life Cycle)

For creating a servlet ,we need to override all the methods of servlet interface.

Also we need to do mapping of the servlet created using *web.xml* file called ***deployment descriptor***.

Deployment Descriptor

(web.xml)

It is a file that contains configuration of your java web application.

It resides in WEB-INF folder

<web-app>

- Servlet declaration
- Servlet mapping
- Initialization parameter
- Welcome-file config
- Filter
- Listener

- Session config

  Etc..

</web-app>

HTML file:

```
<html>

<body>

<h2>Hello World!</h2>

</body>

</html>
```

Java File(Servlet):

```
package com.servlet;

import java.io.IOException;

import java.io.PrintWriter;


import javax.servlet.*;

public class FirstServlet implements Servlet

{

        ServletConfig conf;

        //Life cycle method

        public void init(ServletConfig conf)

        {
```

```
		this.conf=conf;
		System.out.println("Creating Object");
	}
	public void service(ServletRequest req,ServletResponse res) throws
IOException
	{
		System.out.println("Servicing.....");
		PrintWriter out=res.getWriter();
		out.println("Servicing----");


	}
	public void destroy()
	{
		System.out.println("Going to destroy .....");
	}


	//Non Life Cycle methods


	public ServletConfig getServletConfig()
	{
		return conf;
	}
	public String getServletInfo()
	{
		return("Servlet created by Rohit pandey ...");
```

```
      }
}
```

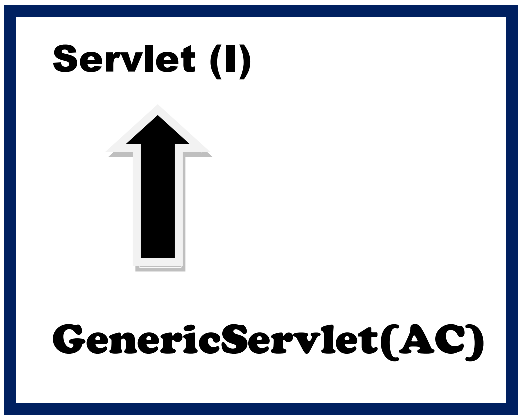
Deployment Descriptor(Web.xml):

```
<!DOCTYPE web-app PUBLIC
 "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
 "http://java.sun.com/dtd/web-app_2_3.dtd" >
<web-app>
  <display-name>Archetype Created Web Application</display-name>
  <servlet>
           <servlet-name>First</servlet-name>
           <servlet-class>com.servlet.FirstServlet</servlet-class>
      </servlet>
      <servlet-mapping>
           <servlet-name>First</servlet-name>
           <url-pattern>/web</url-pattern>
      </servlet-mapping>

</web-app>
```

How to create servlet using GenericServlet Class

Generic servlet provides defination of all the abstract method of servlet interface except service method.

So we neeed to provide implementaton of service method only if we create servlet by extending GenericServlet class.



Java File(Servlet):

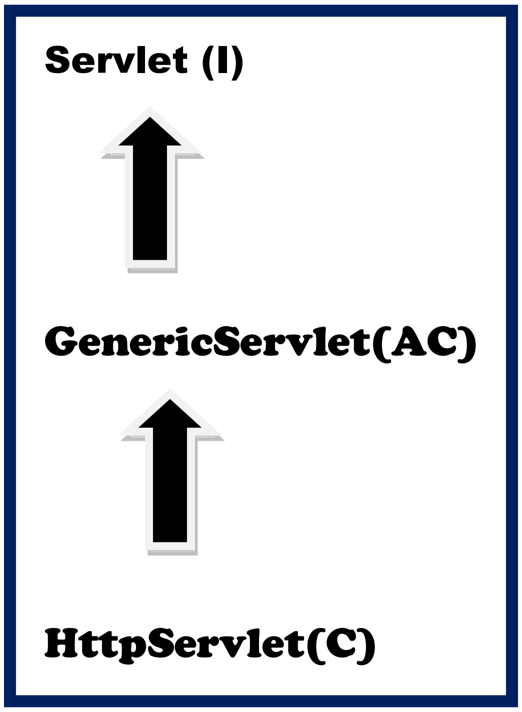
```
package com.servlet;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.GenericServlet;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class SecondServlet extends GenericServlet
{
	public void service(ServletRequest req,ServletResponse res)
throws IOException
	{
		System.out.println("Servicing using Generic Servlet");
		PrintWriter out=res.getWriter();
		out.println("Servicing using Generic Servlet----");
	}

}
```

How to create servlet using HttpServlet Class

Package: javax.servlet.http

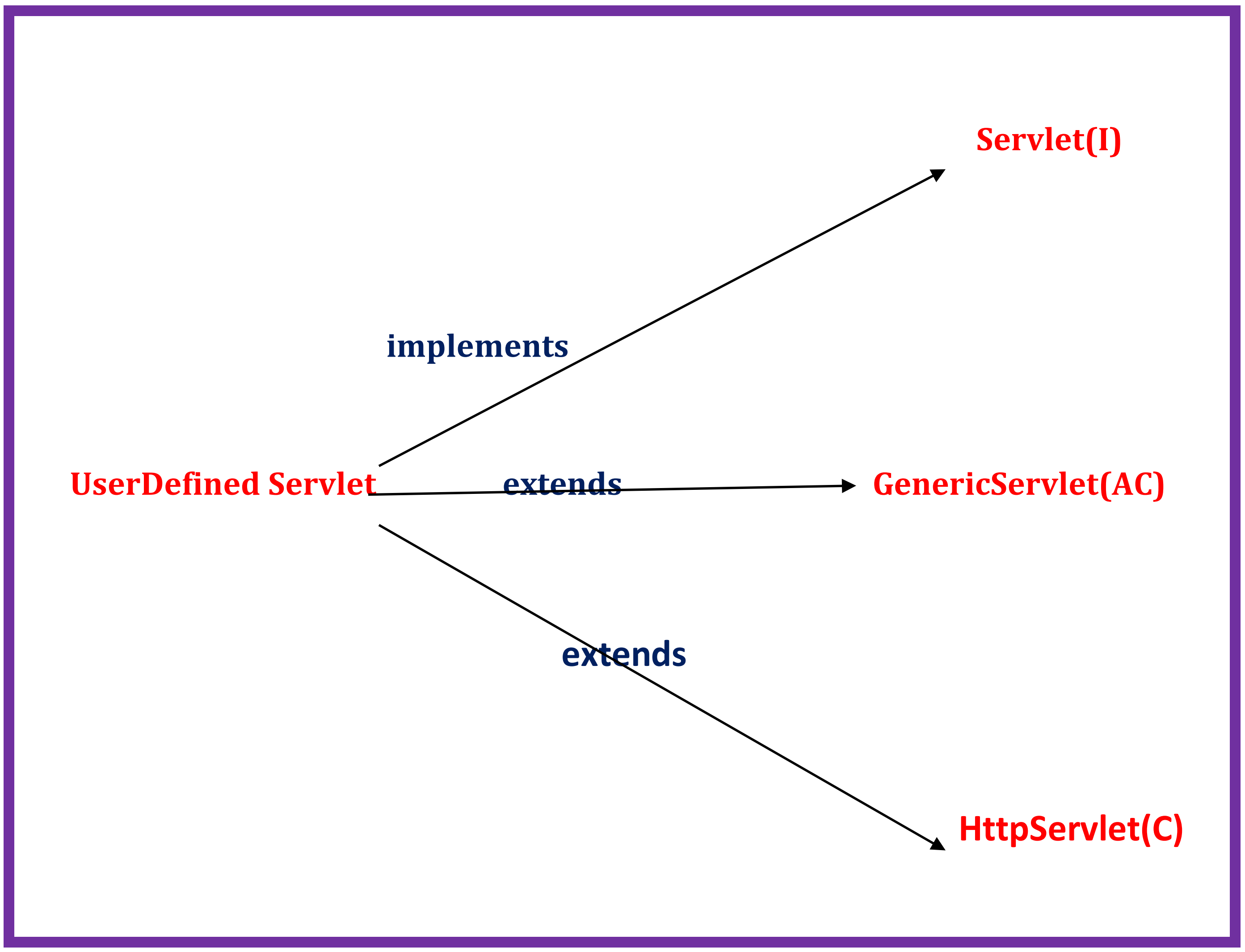
Class:HttpServlet

HttpServlet is used to create protocol specific servlet.



HttpServlet provides the defination of service() method and ,from inside that service method ,http protocol specific methods like doGet,doPost,doHead etc. are called.

HttpServlet is the most common way of creating a servlet.



Java File(Servlet):

```
package com.servlet;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class ThirdServlet extends HttpServlet
{
	public void doGet(HttpServletRequest request,HttpServletResponse 
response) throws IOException
	{
		PrintWriter out=response.getWriter();
		out.println("Servlet creation using HTTP SERVLET");
	}
}
```

Servlet Life Cycle

Load and Instantiate

↓

Calls init() method for servlet initialization

↓

Service() method get called for processing the request.

The server will respond to n number of request from the client

↓

Destroy () method called for releasing the resources.

Submitting form data using servlet

HTML File:

```
<html>
	<head>
		<title>Form Submit</title>
		<style>
			.container{
				width:60%;
				border:1px solid black;
				margin: auto;
				padding:20px;
```

```
                        }
                </style>
        </head>
        <body>
                <div class="container">
                <h1>My Form</h1>
                        <form action="registerServlet" method="post">
                                <table>
                                        <tr>
                                                <td>Enter Your Name : </td>
                                                <td><input type="text" name="uname"
placeholder="Rohit Pandey"></td>
                                        </tr>
                                        <tr>
                                                <td>Enter password : </td>
                                                <td><input type="password" name="pass" ></td>
                                        </tr>
                                        <tr>
                                                <td>Enter email : </td>
                                                <td><input type="email" name="uemail"
placeholder="rohit@gmail.com"></td>
                                        </tr>
                                        <tr>
                                                <td>Select Gender : </td>
                                                <td><input type="radio" name="gender"
value="Male">Male &nbsp; &nbsp;<input type="radio" name="gender"
value="Female">Female</td>
                                        </tr>
                                        <tr>
                                                <td>Select Your Cource : </td>
                                                <td>
                                                        <select name="cource">
                                                                <option>Java</option>
                                                                <option>Python</option>
                                                                <option>Database</option>
                                                                <option>Android</option>
                                                                <option>Networking</option>
                                                                <option>OS</option>
                                                        </select>
                                                </td>

                                        </tr>
                                        <tr>
                                                <td style="text-align:center;"><input
type="checkbox" value="checked" name="condition"></td>
                                                <td><a
href="https://www.facebook.com/terms.php">Agree terms and condition</a></td>
                                        </tr>
                                                <td></td>
                                                <td><button
type="submit">Register</button></td>
                                                <td><button type="reset">Reset</button>
                                        <tr>
                                </table>
                        </form>
                </div>

            </body>
        </html>
```

Java File(Servlet):

```
package com.servlet;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class RegisterServlet extends HttpServlet
{
	public void doPost(HttpServletRequest request,HttpServletResponse
response) throws IOException
	{
		String Name=request.getParameter("uname");
		String Password=request.getParameter("pass");
		String Email=request.getParameter("uemail");
		String Gender=request.getParameter("gender");
		String Cource=request.getParameter("cource");
		String terms=request.getParameter("condition");

		response.setContentType("text/html");
		PrintWriter out=response.getWriter();
		out.println("<h1>Welcome to register servlet<h1>");
		if(terms!=null)
		{
			if(terms.equals("checked"))
			{
				out.println("<h2>Name : "+Name+"</h2>");
				out.println("<h2>Password : "+Password+"</h2>");
				out.println("<h2>Email : "+Email+"</h2>");
				out.println("<h2>Gender : "+Gender+"</h2>");
				out.println("<h2>Cource : "+Cource+"</h2>");
			}
			else
			{
```

```
                              out.println("<h2>You have not acceped terms and 
condition </h2>");

                         }
               }
               else
               {
                         out.println("<h2>You have not acceped terms and 
condition </h2>");

               }


     }
}
```

Differences Between GET and POST method

There are numbers of Htpps methods are there:

Get,Post,Head → Introduced in Http 1.0 V

Options,Put,Delete,Trace → Introduced in Http 1.1 V

GET: If you want to get information from the server then we should go for get method.

- Get request are read only request,and no updations will be performed in the server side.
- End user provided information will be appended to the URL as a part of query string .
  Eg: www.xyz/download?mname=kabali

- By using get request we can only send character data or text data to the server and binary data such as image video etc cannot be send.
- We can send only limited amount of information only(Upto 2KB).
- Security is very less therefore, we cannot send sensitive information using get request.
- Bookmarking of get request is always possible.
- Caching of get request is always possible.

POST : If you want to submit some information to the server then we should go for post.

- Post request usually perform update or write information to the server side.

There are three parts in HTTP Request:

Request Line	***Request Header***	***Request Body***

- In post method end user provided information will be encapsulated to the request body and will be send to the server.
- No restrictions on length of body,so huge amount of information can be send.
- Both binary and text data can be send to the server.

- Security is more ,so we can send sensitive information also.
- Bookmarking of post request is not possible since all information are not there in url only.
- Caching of post request is not possible.

Welcome File and Welcome File list

www.facebook.com **index.jsp,index.html**

$$\begin{tikzcd}\textbf{Client} \arrow[r, "\textbf{request}"] & \textbf{Server}\end{tikzcd}$$

If required page is not mentioned specifically, and client send the request to the server,then server respond this type of request with index.html or idex.jsp type of tage.

index.html or index.jsp page is called HOMEPAGE .

If we want some other our created file (like home.jsp) to act as homepage then we can configure it in deployement descriptor file.

To make this configure we can use wecome file and welcome file list tag in web.xml file.

```
<webapp>

        <welcome-file-list>
```

<welcome-file>home.html</welcome-file>

</welcome-file-list>

</webapp>

Ex:

Lets create a new JSP file home.jsp :

```
<html>
<head>

</head>
<body>
    <h1>This is Home.jsp file</h1>
    <a href="index.jsp">Register Here</a>
</body>
</html>
```

Now in the deployment descriptor file add the below tag:

```
    <welcome-file-list>
    <welcome-file>home.jsp</welcome-file>
  </welcome-file-list>
```

Request Dispatcher

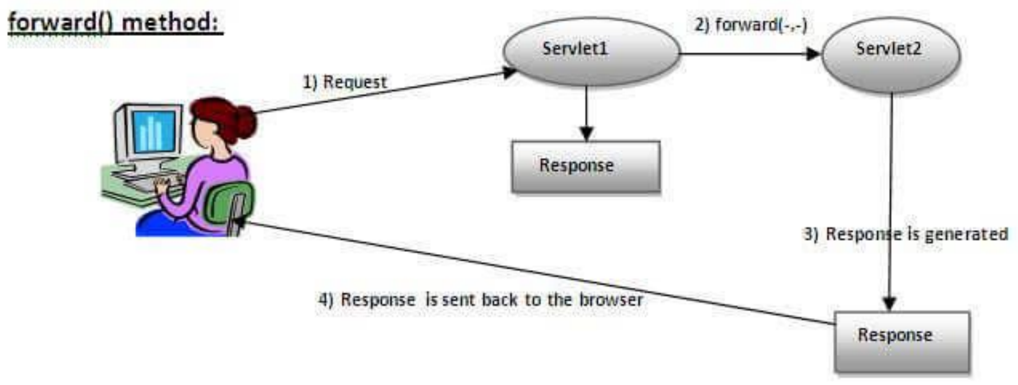
It is responsible for dispatching the request to another resource it may be html,servlet or jsp.

Methods of RequestDispatcher interface

The RequestDispatcher interface provides two methods. They are:

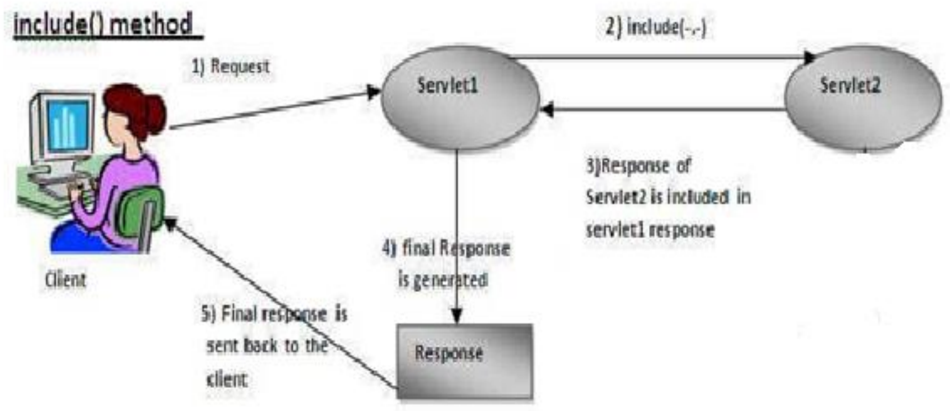
1. **public void forward(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Forwards a

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



Response of 1st servelt is lost and response of last servlet only will reach to the cliet.

2. **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.


include() method
1) Request
2) include(-,-)
Servlet1
Servlet2
3)Response of Servlet2 is included in servlet1 response
4) final Response is generated
Response
5) Final response is sent back to the client
Client

Response of 1st servlet is included with the result of 2nd servlet and then it will reach to the client.

Ex: Lets create another servelt file "SuccessServlet"

```
package com.servlet;


import java.io.IOException;

import java.io.PrintWriter;


import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;


public class SuccessServlet extends HttpServlet

{

     public void doPost(HttpServletRequest request,HttpServletResponse
response) throws IOException, ServletException
```

```
    {
        response.setContentType("text/html");
        PrintWriter out=response.getWriter();
        out.println("<h2>This is success Servelet<h2>");
        out.println("<h2>Successfully Registered<h2>");
    }

}
```

When everything is fine, then the request is to be forwarded to success servlet from register servlet.

If the terms and condition is not agreed,then the message to be displayed ,and with that,the form will also have to be displayed.Therefore the index.jsp file has to be included in the response.

```
package com.servlet;
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
public class RegisterServlet extends HttpServlet

{

     public void doPost(HttpServletRequest request,HttpServletResponse
response) throws IOException, ServletException

     {

          String Name=request.getParameter("uname");

          String Password=request.getParameter("pass");

          String Email=request.getParameter("uemail");

          String Gender=request.getParameter("gender");

          String Cource=request.getParameter("cource");

          String terms=request.getParameter("condition");


          response.setContentType("text/html");

          PrintWriter out=response.getWriter();

          out.println("<h1>Welcome to register servlet<h1>");

          if(terms!=null)

          {

               if(terms.equals("checked"))

               {

                    out.println("<h2>Name : "+Name+"</h2>");

                    out.println("<h2>Password : "+Password+"</h2>");

                    out.println("<h2>Email : "+Email+"</h2>");

                    out.println("<h2>Gender : "+Gender+"</h2>");

                    out.println("<h2>Cource : "+Cource+"</h2>");
                    // Assume data saved to DB

                    RequestDispatcher
rd=request.getRequestDispatcher("/success");

                    rd.forward(request, response);
                }
            }
            else
            {
                out.println("<h2>You have not acceped terms and
condition </h2>");


                //We have to include index.html

                RequestDispatcher
rd=request.getRequestDispatcher("index.jsp");
                rd.include(request, response);
            }


        }
    }
```

Q)We have to create a project that includes:

An html file in which we have to take two numbers as input .

Then we have to call a Servlet “AddServlet” which find the sum of the numbers.

If the sum is -ve the request to be forwared to another servlet “PositiveServlet” which will make the negative value as +ve.

Then we have to forward the request to another servlet “SquareServlet” which will find the square of the number.

After printing the result,the initial form has to displayed in both cases.

Parameters and Attributes in servlet

Parameters: These are those values which are provided by user to any servlet to process the request during the request operation.

It can came from user form or web.xml file.

Servlet only read that value for request processing.t cannot modify that value.

Parameters are mostly data send using form ,initialization parameters etc.

How to get initialization parameter:

- String name=request.getParameter(“name of your parameter”)

Attributes: These are the objects that are attached by one servlet and other servlet can fetch that object to process to logic.

Servlet can easily modify,add and remove the content of the attribute when required.

How to perform operation with attribute:

- setAttribute(String name,Object value)
- object value: getAttribute(String name)
- removeAttribute(String name)

EX:

We have to create a form which takes two input integer and then send it to a servet "s1" then the sum of two numbers is calculated and it is set to the request as setAttribute then it is forwarded to servlet "s2".Servlet s2 fetches all the data that is n1,n2,sum and then find the product of two numbers and print sum and product on screen.

HTML File:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

<html>
<head>

</head>
<body>
	<h1>This is Home.jsp file</h1>
	<a href="index.jsp">Register Here</a>
	<br><br><br><br>
	<form action="s1" method="post">
		N1 : <input type="number" name="n1"><br>
		N2 : <input type="number" name="n2"><br>

		<button type="submit">OK</button>
	</form>
</body>
</html>
```

Servlet s1:

```java
package com.attr;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class s1 extends HttpServlet
{

	protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {

		int n1=Integer.parseInt(request.getParameter("n1"));
		int n2=Integer.parseInt(request.getParameter("n2"));

		int s=n1+n2;

		request.setAttribute("sum", s);

		RequestDispatcher rd=request.getRequestDispatcher("s2");
		rd.forward(request, response);
	}

}
```

Servlet s2:

```java
package com.attr;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class s2 extends HttpServlet
{


	protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
		int n1=Integer.parseInt(request.getParameter("n1"));
		int n2=Integer.parseInt(request.getParameter("n2"));
		int p=n1*n2;
		int sum=(Integer)request.getAttribute("sum");
		PrintWriter out=response.getWriter();
		out.println("Sum is : "+sum);
		out.print("Product is : "+p);

	}

}
```

Session Tracking in Servlet

Session tracking is a way to maintain state (data) of an user.

It is also known as State management.

HTTP is a stateless protocol. Content of older request is not saved,and server treat each request as a new request.

Example:

HTML file:

```html
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
```

```
	<head>
		<title>Welcome</title>
	</head>
	<body>
		<form action="servlet1" method="post">
			<input type="text" name="name" placeholder="Rohit
Pandey" style="font-size:35px">
			<button type="submit" style="font-size:35px">Go to
Servlet 1</button>
		</form>
	</body>
</html>
```

Servlet1:

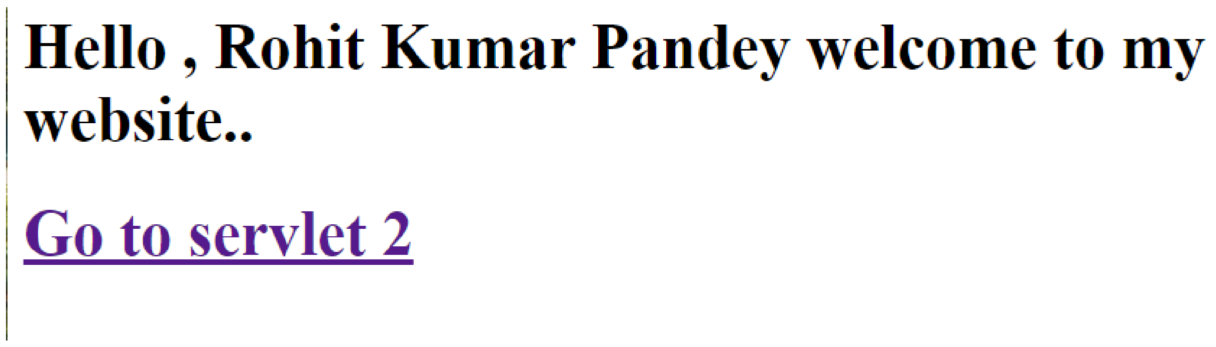
```
package SessionTracking;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;


public class servlet1 extends HttpServlet {


	protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {

	String name=request.getParameter("name");

	response.setContentType("text/html");

	PrintWriter out=response.getWriter();

	out.println("<h1>Hello , "+name+" welcome to my website..</h1>");

	out.println("<h1><a href='servlet2'>Go to servlet 2</a></h1>");
```

```
      }


}
```

Servlet2:

```
package SessionTracking;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


public class servlet2 extends HttpServlet {

      protected void doGet(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {
            String name=request.getParameter("name");
            response.setContentType("text/html");
            PrintWriter out=response.getWriter();
            out.println("<h1>Hello , "+name+" welcome back to my
website..</h1>");
            out.println("<h2>Thank You</h2>");
      }

}
```

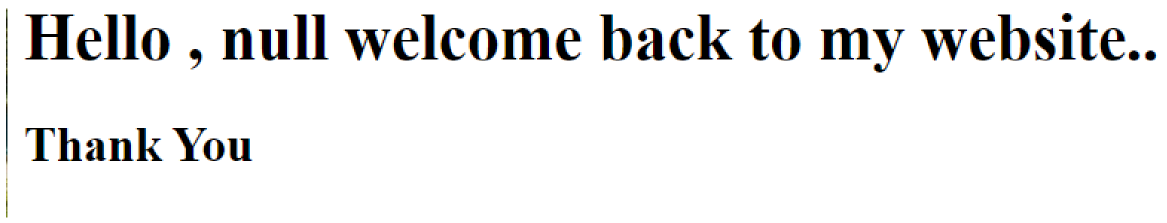
Output Screen 1:


Hello , Rohit Kumar Pandey welcome to my website..

Go to servlet 2

After clicking on – **Go to servlet2**

Output Screen 2:


Hello , null welcome back to my website..

Thank You

We can see that server cannot remember the name of the user in second request to the server.

Therefore session tracking has to be done ,which saves the details of user to the server for sometime.

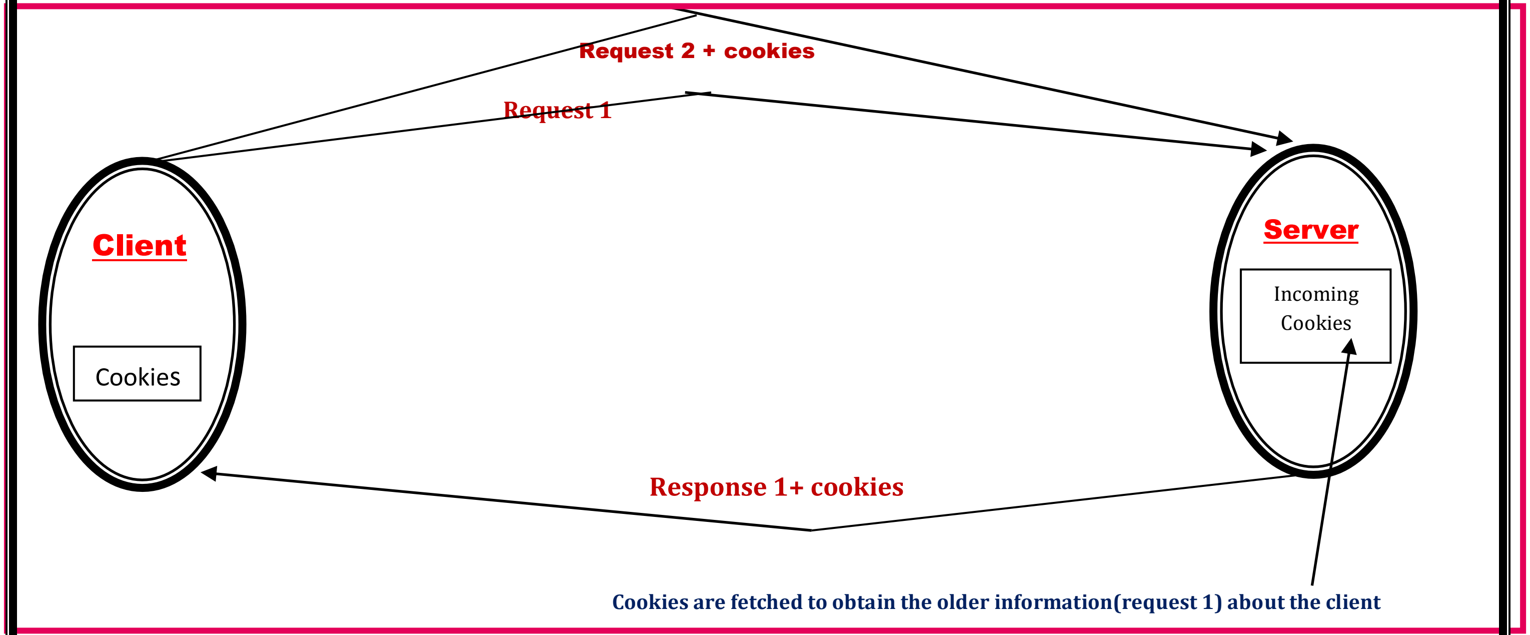
Session Tracking Techniques:

There are four techniques used in Session tracking

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

Cookies in Servlet

Cookies are the textual information which are stored in key value pair format to the client's browser during multiple request.



When client send request 1,then while giving response ,server sends some extra text information called cookies to the client.This cookies are saved to the client browser.

When client send another request to the server then ,the cookiesa are also sent with it.This cookies are fetched at the server to obtain the older information about the client,i.e the information about request 1.

Cookies are mainy the user id,login id of the client etc

How to use Cookies:

- In order to use cookies in java, there is a class ***"Cookie"*** present in ***javax.servlet.http*** package.
- To make cookie, just create a object of cookie class and pass name and its value.

- To add cookie in response just use addCookie(Cookie) method of response interface.

In the above example we can use Cookies to get the desired result as follow :

Servlet 1:

```
package SessionTracking;


import java.io.IOException;

import java.io.PrintWriter;


import javax.servlet.ServletException;

import javax.servlet.http.Cookie;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;


public class servlet1 extends HttpServlet {


	protected void doPost(HttpServletRequest request, 
HttpServletResponse response) throws ServletException, IOException {

	String name=request.getParameter("name");

	Cookie cookie=new Cookie("User_Name" , name);

	response.addCookie(cookie);


	response.setContentType("text/html");

	PrintWriter out=response.getWriter();
	out.println("<h1>Hello , "+name+" welcome to my website..</h1>");
	out.println("<h1><a href='servlet2'>Go to servlet 2</a></h1>");


	}


}
```

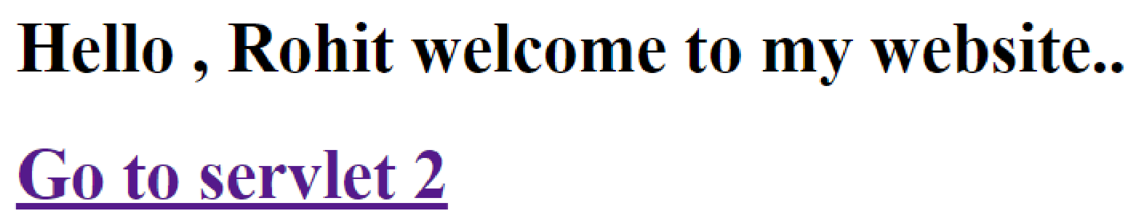
Servlet2:

```
package SessionTracking;


import java.io.IOException;
import java.io.PrintWriter;


import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;



public class servlet2 extends HttpServlet {


	protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
		//String name=request.getParameter("name");
		PrintWriter out=response.getWriter();
```

```
            Cookie cookies[]= request.getCookies();

            boolean f=false;

            String name="";

            if(cookies==null)

            {

                  out.println("You are new user..Go to homepage..");

            }

            else

            {

                  for(Cookie c : cookies)

                  {

                        String tname=c.getName();

                        if(tname.equals("User_Name"))

                        {

                              f=true;

                              name=c.getValue();

                        }

                  }

            }

            response.setContentType("text/html");


            if(f)

            {

                  out.println("<h1>Hello , "+name+" welcome back to my
website..</h1>");

                  out.println("<h2>Thank You</h2>");

            }
```

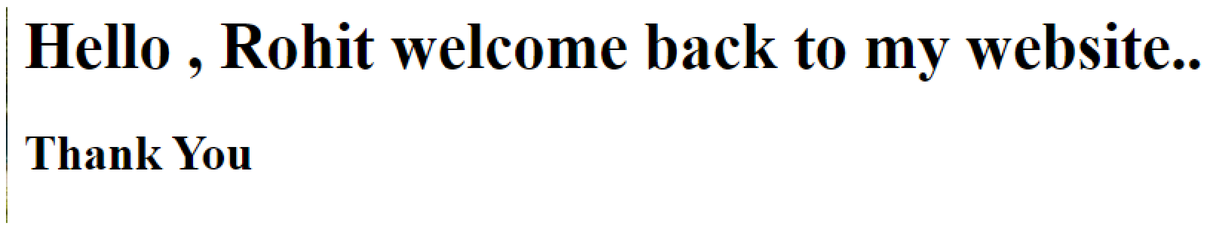
```
                else
                {
                        out.println("You are new user..Go to homepage..");
                }
        }

}
```

Output Screen 1:



```
Hello , Rohit welcome to my website..
Go to servlet 2
```

Output Screen 2:

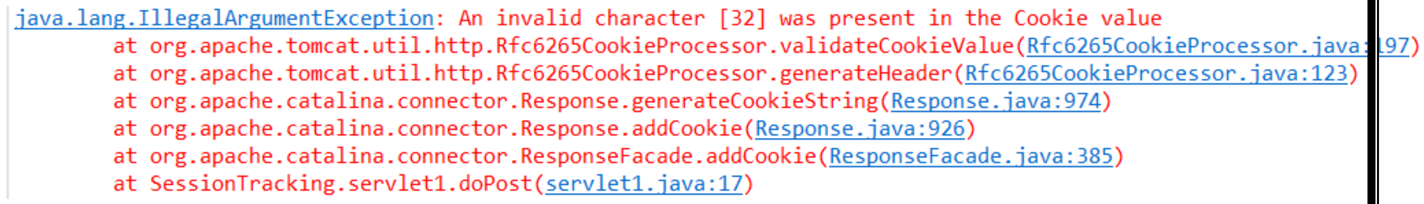


```
Hello , Rohit welcome back to my website..
Thank You
```

Disadvantages of Cookie:

Special symbols such as semicolon (;), comma (,), equal sign (=), and space cannot be used in the cookie value, otherwise an exception will occur.

If we give name as Rohit Kumar Pandey in the above example then we will get the following exception.



```
java.lang.IllegalArgumentException: An invalid character [32] was present in the Cookie value
	at org.apache.tomcat.util.http.Rfc6265CookieProcessor.validateCookieValue(Rfc6265CookieProcessor.java:197)
	at org.apache.tomcat.util.http.Rfc6265CookieProcessor.generateHeader(Rfc6265CookieProcessor.java:123)
	at org.apache.catalina.connector.Response.generateCookieString(Response.java:974)
	at org.apache.catalina.connector.Response.addCookie(Response.java:926)
	at org.apache.catalina.connector.ResponseFacade.addCookie(ResponseFacade.java:385)
	at SessionTracking.servlet1.doPost(servlet1.java:17)
```

Q)Create Complete Ajax Based Registration Module

SignUP.jsp:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
      pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<title>Welcome</title>
<!-- Compiled and minified CSS -->
<link rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css/materialize
.min.css">

<!-- Compiled and minified JavaScript -->
<script
      src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/materialize.m
in.js"></script>

</head>
<body
      style="background: url(bg.jpg); background-size: cover; background-attachment:
fix;">
      <div class="container">
            <div class="row">
                  <div class="col m6 offset-m3">
                        <div class="card">
                              <div class="card-content">
                                    <h3 style="margin-top: 10px;" class="center-
align">Register
                                          Here</h3>
                                    <h5 id="msg"></h5>
                                    <div class="form center-align">
                                          <form id="myform" action="register"
method="post">
                                                <input type="text" name="name"
placeholder="Enter Name">
                                                <input type="password"
name="password"
```

```
                placeholder="Enter Password"> <input type="email"
                name="email" placeholder="Enter Email">
            <button type="submit" class="btn red">SUBMIT</button>
</form>
<div class="loader center-align" style="margin-top:10px; ">
            <div class="preloader-wrapper big active">
                <div class="spinner-layer spinner-blue">
                    <div class="circle-clipper left">
                        <div class="circle"></div>
                    </div>
                    <div class="gap-patch">
                        <div class="circle"></div>
                    </div>
                    <div class="circle-clipper right">
                        <div class="circle"></div>
                    </div>
                </div>

                <div class="spinner-layer spinner-red">
                    <div class="circle-clipper left">
                        <div class="circle"></div>
                    </div>
                    <div class="gap-patch">
                        <div class="circle"></div>
                    </div>
                    <div class="circle-clipper right">
                        <div class="circle"></div>
                    </div>
                </div>

                <div class="spinner-layer spinner-yellow">
                    <div class="circle-clipper left">
                        <div class="circle"></div>
```

```
                                                                  </div>
                                                                  <div class="gap-
patch">
                                                                         <div
class="circle"></div>
                                                                  </div>
                                                                  <div class="circle-
clipper right">
                                                                         <div
class="circle"></div>
                                                                  </div>
                                                            </div>

                                                            <div class="spinner-layer
spinner-green">
                                                                  <div class="circle-
clipper left">
                                                                         <div
class="circle"></div>
                                                                  </div>
                                                                  <div class="gap-
patch">
                                                                         <div
class="circle"></div>
                                                                  </div>
                                                                  <div class="circle-
clipper right">
                                                                         <div
class="circle"></div>
                                                                  </div>
                                                            </div>
                                                     </div>
                                                     <h5>Please Wait...</h5>
                                              </div>

                                        </div>
                                 </div>
                           </div>
                    </div>
             </div>
       </div>

<script
  src="https://code.jquery.com/jquery-3.5.1.min.js"
  integrity="sha256-9/aliU8dGd2tb6OSsuzixeV4y/faTqgFtohetphbbj0="
  crossorigin="anonymous">
</script>
  <script>
      $(document).ready(function(){
                    console.log("Page is ready ...")
                    $(".loader").hide();
                    $(".form").show();
                    $("#myform").on('submit',function(event){
                           event.preventDefault();
```

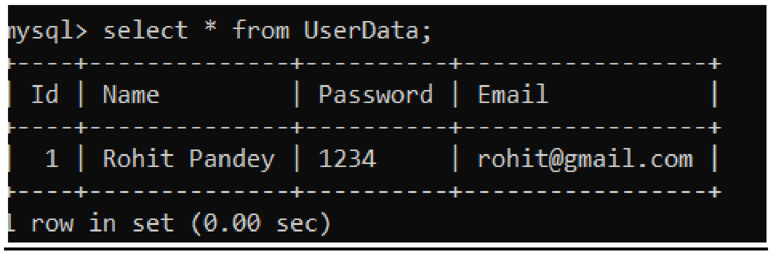
```
                                var f=$(this).serialize();
                                console.log(f);
                                $(".loader").show();
                                $(".form").hide();

                                $.ajax({

                                      url:"register",
                                      data:f,
                                      type:'POST',
                                            success:function(data,textStatus,jqXHR)
                                            {
                                                  console.log(data);
                                                  console.log("Success......")
                                                  $(".loader").hide();
                                                  $(".form").show();
                                                  },

      error:function(jqXHR,textStatus,errorThrown){
                                                        console.log(data);
                                                        console.log("Error......")
                                                        $(".loader").hide();
                                                        $(".form").show();
                                                        }

                                      })
                                })
                    })
    </script>
  </body>
  </html>
```

RegisterServlet:

```
package RegModule;



import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;
import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;


public class RegisterServlet extends HttpServlet {

	private static final long serialVersionUID = 1L;


	protected void doPost(HttpServletRequest request,
HttpServletResponse response) throws ServletException, IOException {

		String Name=request.getParameter("name");

		String Password=request.getParameter("password");

		String Email=request.getParameter("email");


		PrintWriter out=response.getWriter();

		out.println(Name);

		out.println(Password);

		response.setContentType("text/html");

		//connection

		try

		{

			Thread.sleep(3000);

			Class.forName("com.mysql.cj.jdbc.Driver");

			Connection
con=DriverManager.getConnection("jdbc:mysql://localhost:3306/test","ro
ot","76448");
```

```
                String s="insert into UserData (name,password,email)
values(?,?,?);";

                PreparedStatement st=con.prepareStatement(s);

                st.setString(1, Name);

                st.setString(2, Password);

                st.setString(3, Email);


                st.executeUpdate();

                out.println("<h1>Done...</h1>");

            }

            catch (Exception e)

            {

                e.printStackTrace();

                out.print("<h1>Error...</h1>");

            }



        }


}
```

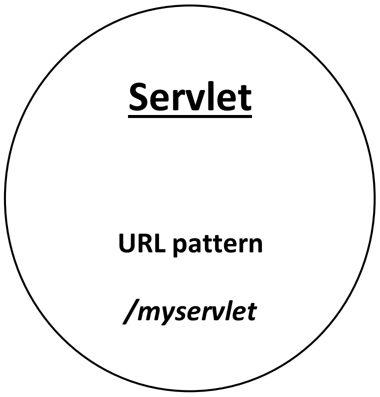
<u>Database State:</u>



```
ysql> select * from UserData;
+----+--------------+----------+-----------------+
| Id | Name         | Password | Email           |
+----+--------------+----------+-----------------+
|  1 | Rohit Pandey | 1234     | rohit@gmail.com |
+----+--------------+----------+-----------------+
 row in set (0.00 sec)
```

URL Rewriting using Java Servlet

URL rewriting is a process of appending or modifying any url structure while loading a page.



While calling this servlet,we can append the data to be sent with the URL for calling the servlet.

EX:

www.javatech.com/myservlet? name=John &roll=101

Now at the server end ,this data is easily accessible using request.getParameter() method.

String Name=request.getParameter("name");   //Name=John

int roll=request.getParameter("roll");   //roll=101

Example: We have a form which takes name as input.We submit that form in servlet 1.Then from servlet 1 we fetch the data ,and then send that data to servlet2 using url rewriting.

HTML File:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
	<head>
		<title>Welcome..</title>
	</head>
	<body>
		<form action="urlServlet1" method="post">
			<input type="text" name="name" placeholder="Rohit Pandey">
			<button type="submit">OK</button>
		</form>

	</body>
</html>
```

Servlet1:

```
package URLRewritting;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class urlServlet1 extends HttpServlet {

	protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
		PrintWriter out= response.getWriter();
		response.setContentType("text/html");
		String name=request.getParameter("name");
		out.println("<h1>Your name : "+ name+"</h1>");
		out.println("<a href='urlServlet2?name="+name+"'>Go to second
Servlet</a>");
	}

}
```

Servlet2:

```
package URLRewritting;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class urlServlet2 extends HttpServlet {

	protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException {
		PrintWriter out= response.getWriter();
		response.setContentType("text/html");
		String name=request.getParameter("name");
		out.println("<h1>This is servlet 2 </h1>");
		out.println("<h1>Your name : "+ name+"</h1>");
	}

}
```

Hidden Form Field

In hidden form field a hidden (invisible) textfield is used for maintaining the state of an user.

<input type=”hidden” name=”user_name” value=”Rohit Pandey”>

In the above example instead of using anchor tag in servlet1, to go to servlet 2,we can use a form with hidden input field as shown below.

Servlet1:

```
package URLRewritting;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class urlServlet1 extends HttpServlet {
	protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
		PrintWriter out= response.getWriter();
		response.setContentType("text/html");
		String name=request.getParameter("name");
		out.println("<h1>Your name : "+ name+"</h1>");
				out.println("<form action='urlServlet2'><input
type='hidden' name='name' value="+name+"><button type='submit'>Go to Servlet
2</button></form>");
	}

}
```

Session Tracking using HttpSession

Session simply means small interval of time
Used for state management.
Whenever a client sends a request to the server then,server creates an object of HttpSession for that client. Each object has a unique session ID Inside that object,we can keep whatever data we want.When client again sends some request to server then server will check,if there is an object of HttpSession for that client.If object is present then it will not create a new object,and from the older object we can fetch whatever data required to us about the older request of that client.

The HttpSession Object is destroyed only if-

- Browser is closed
- Invalidate the HttpSession object explicitely.
- The time is expired

Using HttpSession in Java:

We have an interface "HttpSession" in "javax.servlet.http" package.

For using HttpSession we need to get the object of HttpSession.We can get the object using HttServletRequest parameter-

HttpSession session=request.getSession();

HttpSession has a number of useful methods-

- **setAttribute(String key,Object value)** : Used for setting some value to the session object.
- **Object obj = getAttribute(String key)**: used for getting some value from session object.
- **getId():** Used to get the id of the session object.
- **removeAttribute(String k)** : Used for removing an attribute from the session object.
- **invalidate()** : Used to completely invalidate the session object.

JSP

Jsp (Java server pages) is an extension of servlet technology,but JSP provides more functionality than servlet.

Jsp pages have extention as ".jsp". Ex: index.jsp.

JSP pages are similar to HTML pages but , JSP is a backend technology.

In JSP we can write :

- HTML
- CSS
- Javscript
- Java

At runtime this jsp page is converted in servlet .

Disadvantages of Servlet

- While responding to client , the Servlet generate static content
  ```
  out.println("<h1>Hello Rohit</h1>")
  ```
  Designing in servlet is very difficult.Using JSP we can write dynamic content inside static content.
- For every request in servlet you have to write service method which is very tiresome process.
- Whenever modification is made in static content(Presentation logic) then servlet needs to be recompiled and redeployed.

Important Tags of JSP

Declarative 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.

```
<%!
        Variables;
        Methods;
%>
```

Scriptlet tag:

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

```
<%
        Java code
%>
```

Expression Tag:

```
<%= statement %>
```

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.

Example:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
      <head>
            <title>Welcome</title>
      </head>                                                     Declarative
      <body>
            <h1>JSP Page</h1>
            <%!
                  int a=50,b=10;
                  String name="techsoft india";
                  public int doSum()
                  {
                        return a+b;
                  }
                  public String reverse()
                  {
                        StringBuffer br=new StringBuffer(name);
                        return br.reverse().toString();
                  }
            %>
            <%                                              Scriptlet
                  out.println(doSum());
                  out.println(reverse());
            %>

            <h1> Sum is : <%= doSum() %></h1>

      </body>                                         Expression
</html>
```

Directive Tag(JSP Directives):

The **jsp directives** are messages that tells the web container how to translate a JSP page into the corresponding servlet.

There are three types of directives:

- page directive
- include directive
- taglib directive

Syntax of JSP Directive

<%@ directive attribute="value" %>

JSP page directive

The page directive defines attributes that apply to an entire JSP page.

<%@ page attribute="value" %>

Attributes of JSP page directive

- import
- isErrorPage
- extends
- contentType
- session
- isThreadSafe

```
<html>
<body>

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

</body>
</html>
```

Example:

<%@page import=”java.util.ArrayList”%>

<%@page isErrorPage=”true” %>

<%@page session =”false” %>

<%@page extends=”Student” %>

Jsp Include Directive:

The include directive is used to include the contents of any resource it may be jsp file, html file or text file.

<%@ include file="resourceName" %>

```
<html>
         <body>

               <%@ include file="header.html" %>

               Today is: <%= java.util.Calendar.getInstance().getTime() %>

         </body>
</html>
```

JSP Taglib directive:

This is used,when we want to use other tag library in our JSP page,such as JSTL(JSP Standard Tag Library) or custom library created by user.

```
<%@taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"  %>
<!DOCTYPE html>
<html>
      <head>
            <title>Welcome</title>
      </head>
      <body>
            <h1>JSP Page</h1>
            <h2>JSP Tag Lib Directive</h2>>
            <c:set var="name" value="techsoft india"></c:set>
            <c:out value="${name }"></c:out>
            <c:if test="${3>2 }">
                  <h2>This is true block</h2>
            </c:if>
      </body>
</html>
```

Error Handling in JSP

Main JSP File:

```
<%@page errorPage="error.jsp" %>
<!DOCTYPE html>
<html>
      <head>
            <title>Welcome</title>
      </head>
      <body>
            <h1>JSP Page</h1>
            <%!
                  int n1=20;
                  int n2=0;
            %>
            <%
                  int div=n1/n2;
            %>
            <h1>Division=<%=div %></h1>
      </body>
</html>
```

We have declared “error.jsp” file as our error page. So whenever some error occurs in the main jsp page,the content of error.jsp page will be displayed on the screen.

ErrorPage:

```
 <%@page isErrorPage="true" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Error!!!</title>
</head>
<body>
	<h1>Sorry !!! Something went wrong</h1>
	<h2><%=exception %></h2>
</body>
</html>
```

Creating Custom Tags in JSP

Custom tags are user-defined tags.

Advantages of Custom Tags

The key advantages of Custom tags are as follows:

1. **Eliminates the need of scriptlet tag** The custom tags eliminates the need of scriptlet tag which is considered bad programming approach in JSP.
2. **Separation of business logic from JSP** The custom tags separate the the business logic from the JSP page so that it may be easy to maintain.
3. **Re-usability** The custom tags makes the possibility to reuse the same business logic again and again.

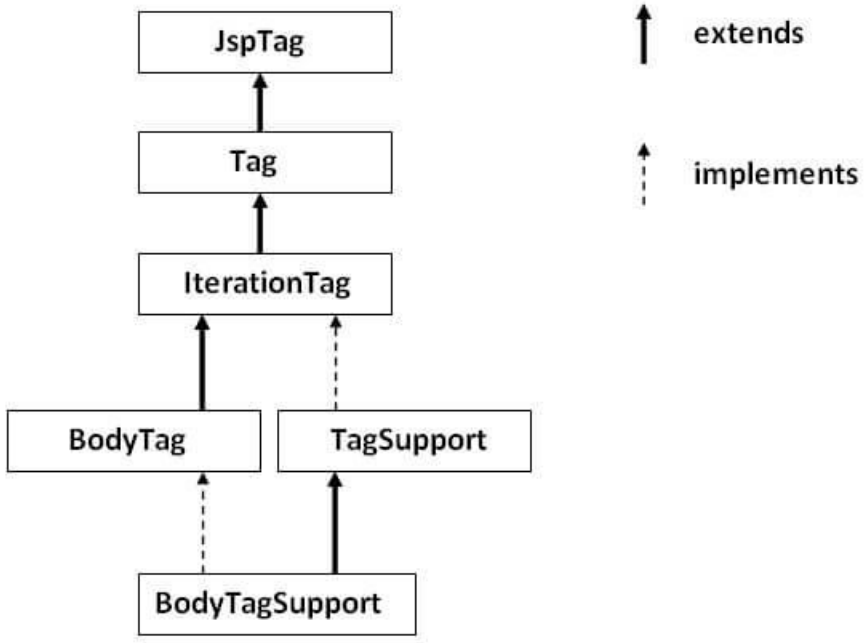
Steps to create custom tag:

For creating any custom tag, we need to follow following steps:

1. **Create the Tag handler class** and perform action at the start or at the end of the tag.
2. **Create the Tag Library Descriptor (TLD) file** and define tags
3. **Create the JSP file that uses the Custom tag defined in the TLD file**

1)Create tag handler class

The javax.servlet.jsp.tagext package contains classes and interfaces for JSP custom tag API. The JspTag is the root interface in the Custom Tag hierarchy.



To create the Tag Handler, we are inheriting the **TagSupport class** and overriding its method **doStartTag()**.

```
package tags;

import javax.servlet.jsp.JspException;

import javax.servlet.jsp.JspWriter;

import javax.servlet.jsp.tagext.TagSupport;
```

```
public class MyTagHandler extends TagSupport

{

      @Override

      public int doStartTag() throws JspException

      {

            try

            {

                  //taks...

                  JspWriter out=pageContext.getOut();

                  out.println("<h1>This is custom tag...</h1>");

                  out.println("<p>Custom para </p>");

            }

            catch(Exception e)

            {

                  e.printStackTrace();

            }

            return SKIP_BODY;

      }

}
```

public int doStartTag()throws JspException	it is invoked by the JSP page implementation object. The JSP programmer should override this method and define the business logic to be performed at the start of the tag.

2) Create the TLD file

Tag Library Descriptor (TLD) file contains information of tag and Tag Hander classes. It must be contained inside the **WEB-INF** directory.

```
<taglib>
  <tlib-version>2.0</tlib-version>
  <jsp-version>2.0</jsp-version>
  <short-name>quote</short-name>
<uri>tag lib version id</uri>
  <description>
     This tag library contains several tag extensions
     useful for formatting content for HTML.
  </description>

  <tag>
    <name>mytag</name>
    <tag-class>tags.MyTagHandler</tag-class>
    <body-content>tagdependent</body-content>
  </tag>
</taglib>
```

3) Create the JSP file

Let's use the tag in our jsp file. Here, we are specifying the path of tld file directly.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
    <%@taglib uri="/WEB-INF/mylib.tld" prefix="t" %>
<!DOCTYPE html>
<html>
      <head>
            <title>Welcome</title>
      </head>
      <body>
      <h1>Hello</h1>
      <t:mytag></t:mytag>
      </body>
</html>
```

Custom tag with attribute

Tag Handler

```
package tags;

import java.io.IOException;

import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.TagSupport;

public class tagHandler2 extends TagSupport
{
	public int num;
	public String color;
	public void setColor(String color) {
		this.color = color;
	}

	public void setNum(int num)
	{
		this.num = num;
	}

	@Override
	public int doStartTag() throws JspException
	{
		JspWriter out=pageContext.getOut();

		try
		{
			out.println("<div style='color:"+color+"'>");
			for(int i=1;i<=10;i++)
			{
				out.println(i*num+"<br>" );

			}
			out.println("</div>");
		}
		catch (IOException e)
		{

			e.printStackTrace();
		}

		return SKIP_BODY;
	}
}
```

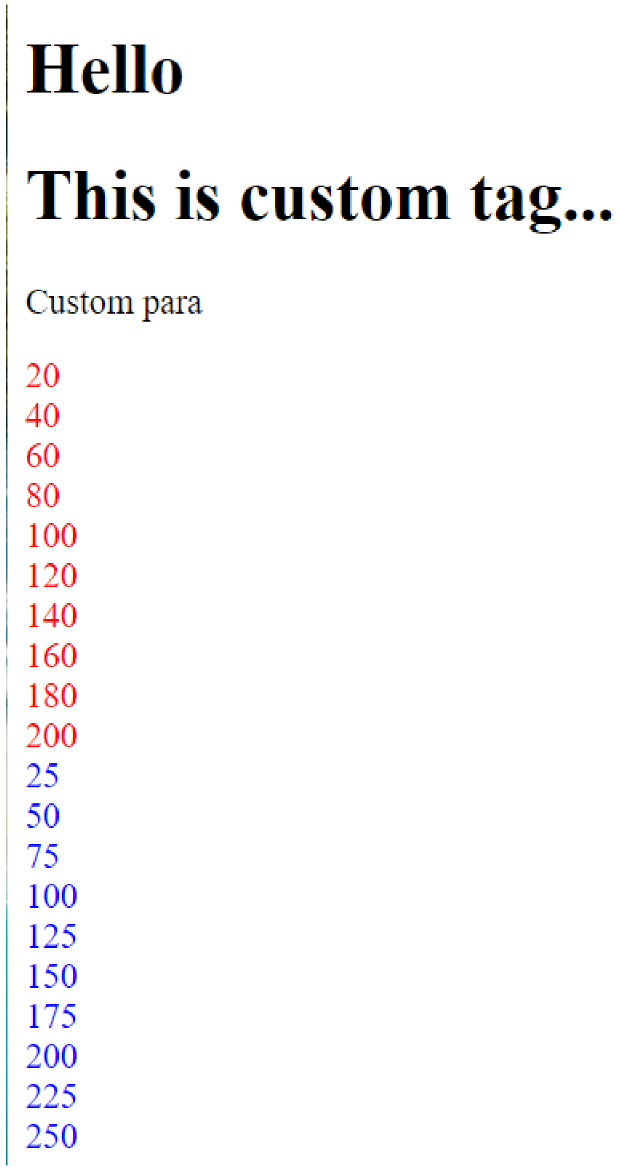
TLD File

```
<taglib>
  <tlib-version>2.0</tlib-version>
  <jsp-version>2.0</jsp-version>
  <short-name>quote</short-name>
<uri>tag lib version id</uri>
  <description>
     This tag library contains several tag extensions
     useful for formatting content for HTML.
  </description>

  <tag>
    <name>mytag</name>
    <tag-class>tags.MyTagHandler</tag-class>
  </tag>

    <tag>
    <name>printTable</name>
    <tag-class>tags.tagHandler2</tag-class>
    <body-content>tagdependent</body-content>
    <attribute>
      <name>num</name>
      <required>true</required>
    </attribute>

    <attribute>
      <name>color</name>
      <required>true</required>
    </attribute>
  </tag>
</taglib>
```

JSP File

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
    <%@taglib uri="/WEB-INF/mylib.tld" prefix="t" %>
<!DOCTYPE html>
<html>
      <head>
            <title>Welcome</title>
      </head>
      <body>
      <h1>Hello</h1>
      <t:mytag></t:mytag>
      <t:printTable num="20" color="red"></t:printTable>
      <t:printTable num="25" color="blue"></t:printTable>
      </body>
</html>
```

Output:



```
Hello

This is custom tag...

Custom para

20
40
60
80
100
120
140
160
180
200
25
50
75
100
125
150
175
200
225
250
```

JSP Implicit Object

Implicit objects are created durin translation phase of jsp to servlet. These objects can be directly used in scriptlets that goes in the service method.They are created by the web container automatically and they can be accessed using objects.

There are **9 jsp implicit objects.** A list of the 9 implicit objects is given below:

Object	Type	Info
out	JspWriter	Used for writing any data to the buffer. It is the object of JspWriter.
request	HttpServletRequest	It is object of type HttpServletRequest. It can be used to get request information such as parameter, server name, server port, content type etc.
response	HttpServletResponse	I is an object of type HttpServletResponse.It can be used to add or manipulate response such as redirect response to another resource, send error etc.
config	ServletConfig	It is an implicit object of type ServletConfig.It is used to get initialization parameter from the web.xml file.
application	ServletContext	It is an implicit object of type ServletContext.It is 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.
session	HttpSession	It is an implicit object of type HttpSession.Used to set,get or remove attribute or to get session information.
pageContext	PageContext	It is an implicit object of type PageContext class.Used to set,get or remove attribute from page,request,session,application
page	Object	In JSP, page is an implicit object of type Object class.
exception	Throwable	It is an implicit object of type java.lang.Throwable class. Used to print the exception. But it can only be used in error pages.

Example:

```
            <%
                  out.println("This is my implicit Object");
                  request.getParameter("name");
                  response.setContentType("text/html");
                  config.getInitParameter("name");
                  application.getServerInfo();
                  session.setAttribute("name", "Rohit");
            %>
```

How to redirect from one JSP page to another JSP page

We have a method sendRedirect("Page Url") present in response object which can be used to redirect a page to another page or resource.

Page 1

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Page1</title>
</head>
<body>
      <h1>This is page 1</h1>
      <a href="Page2.jsp">Go to Page 2</a>
</body>
</html>
```

Page 2:

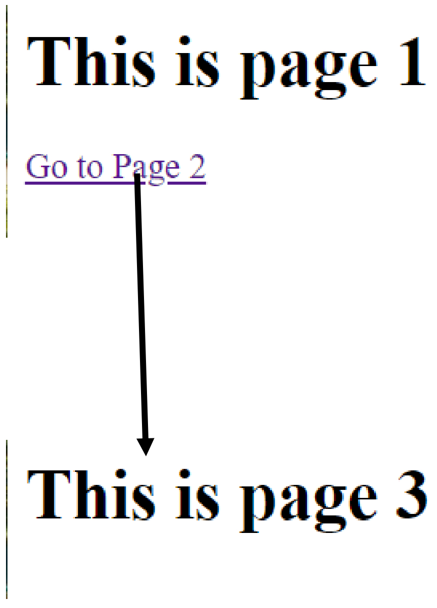
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title> Page2</title>
</head>
<body>
```

```
      <h1>This is page 2</h1>
      <%
            response.sendRedirect("Page3.jsp");
      %>
</body>
</html>
```

Page 3:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title> Page3</title>
</head>
<body>
      <h1>This is page 3</h1>
</body>
</html>
```

Output

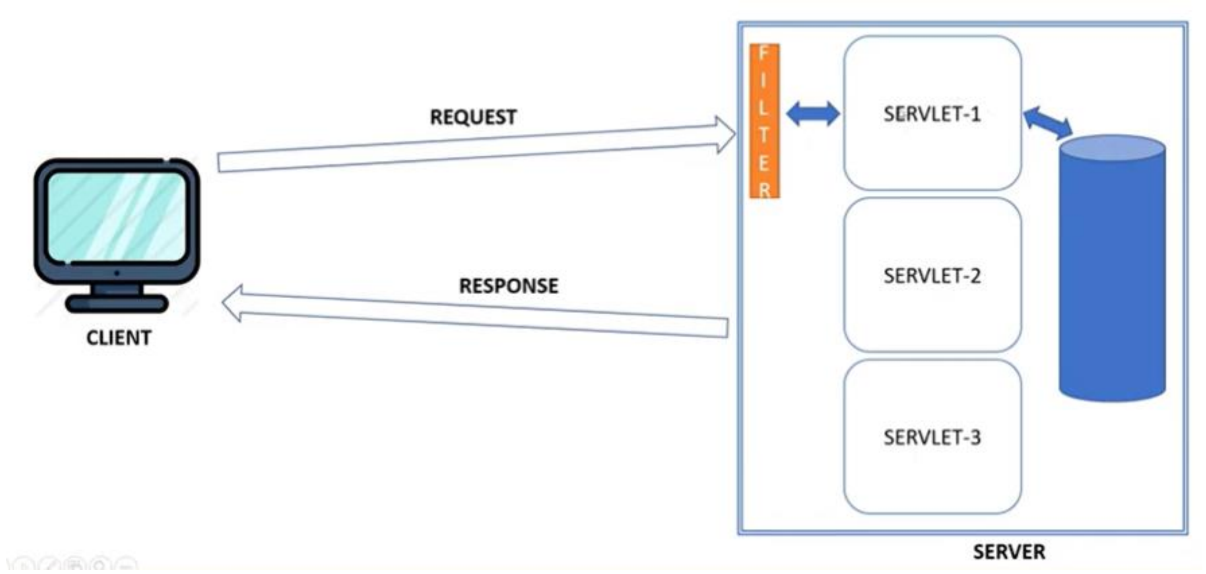


```
This is page 1

Go to Page 2
```

```
This is page 3
```

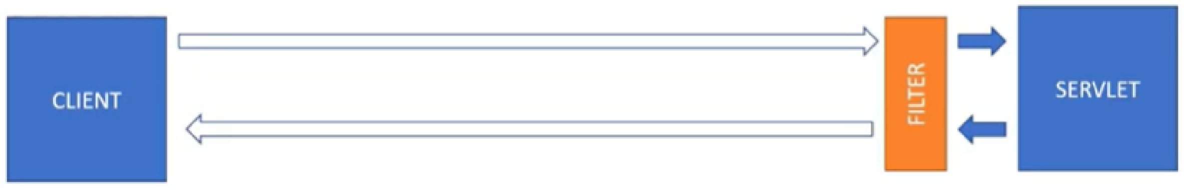
Filters

A filter is an object that is invoked at the preprocessing and postprocessing of a request.



The **servlet filter is pluggable**, i.e. its entry is defined in the web.xml file, if we remove the entry of filter from the web.xml file, filter will be removed automatically and we don't need to change the servlet.

Usage of Filter



- Authentication and authorization of request and resources.
- Formatting of request body or header before sending it to servlet.
- Compressing the response data to the client.
- Alter response by adding cookies,header information etc.
- Input validations etc.

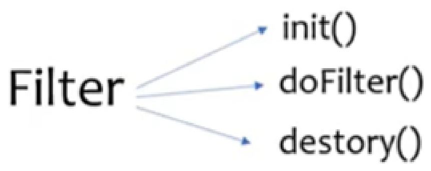
Filter API

Like servlet filter have its own API. The javax.servlet package contains the three interfaces of Filter API.

1. Filter
2. FilterChain
3. FilterConfig

Filter interface

For creating any filter, you must implement the Filter interface. Filter interface provides the life cycle methods for a filter.



public void init(FilterConfig config)	init() method is invoked only once. It is used to initialize the filter.
public void doFilter(HttpServletRequest request,HttpServletResponse response, FilterChain chain)	doFilter() method is invoked every time when user request to any resource, to which the filter is mapped.It is used to perform filtering tasks.
public void destroy()	This is invoked only once when filter is taken out of the service.

Practicle : Creating Filter

We have a JSP Page

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
```

```
<html>
	<head>
		<title>Filter</title>
	</head>
	<body>
		<h1><a href="ProfileServlet">Go to Profile Servlet</a></h1>
		<h1><a href="OrderServlet">Go to Order Servlet</a></h1>
	</body>
</html>
```

We have two servlets Profile Servlet and Order Servlet.

Profile Servlet

```
package Filter;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;


public class ProfileServlet extends HttpServlet {
	protected void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
	{
		System.out.println("Profile servlet executed...");
		PrintWriter out=response.getWriter();
		response.setContentType("text/html");
		out.println("<h1>Welcome to profile page</h1>");
		out.println("<h1>This is profile servlet</h1>");
	}

}
```

Order Servlet :

```
package Filter;

import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class OrderServlet extends HttpServlet {

	protected void doGet(HttpServletRequest request, HttpServletResponse 
response) throws ServletException, IOException
	{
		PrintWriter out=response.getWriter();
		response.setContentType("text/html");
		out.println("<h1>Welcome to order page</h1>");
		out.println("<h1>This is order servlet</h1>");
		out.println("<h1>This is order servlet "
+request.getContextPath() +"</h1>");
	}
}
```

We create a filter which will be executed before calling the servlet and after executing the servlet.

MyFilter

```
package Filter;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class MyFilter implements Filter
{

	public void init(FilterConfig filterConfig) throws ServletException
	{


	}

	public void doFilter(ServletRequest request, ServletResponse response, 
FilterChain chain)
			throws IOException, ServletException
	{

		//PreProcessing task....
```

```
            System.out.println("Before Filter.....");
            //.....
            //......
            //End of preprocessing task


            chain.doFilter(request, response);  //forward to servlet...

            //Post Processing task....
            System.out.println("After Servlet");
            //....
            //....
            //End of postprocessing task

      }

      public void destroy()
      {


      }

}
```

In the Web.xml file we have to do the configuration as follow

```
<web-app>
      <!-- Filter -->
      <filter>

            <filter-name>Filter1</filter-name>
            <filter-class>Filter.MyFilter</filter-class>
      </filter>

      <filter-mapping>
            <filter-name>Filter1</filter-name>
            <url-pattern>/ProfileServlet</url-pattern>
            <url-pattern>/OrderServlet</url-pattern>
      </filter-mapping>
</web-app>
```

Using Filter in projects

We can use filter for user authentication.

Let we have a project in which we have to display a page only if user is logged in. In such cases we can use filter.

At first,after login we store the details of user in session.Then while getting a request for sensitive page,we can use a preprocessing filter to check the content of the session.If the session attribute is not null then it means that user is logged in,so we will do Filter the request to the required page.But if the session attribute is null,means that user is not logged in,in this case we will show error message to the user.

```
package Filter;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpSession;

public class MyFilter implements Filter
{

	public void init(FilterConfig filterConfig) throws ServletException
	{


	}

	public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain)
			throws IOException, ServletException
	{

		HttpServletRequest req=(HttpServletRequest)request;
		HttpSession s=req.getSession();
		User user=(User)s.getAttribute("user");
		if(user!=null)
```

```
				{
					chain.doFilter(request, response);  //forward to servlet...
				}

				else
				{
					System.out.println("Not logged in..");
				}



		}

		public void destroy()
		{


		}

	}
```

JSTL

(JSP STANDARD TAG LIBRARY)

The JSP standard tag library(JSTL) is a collection of predefined tags to simplify the jsp development.

Advantage of JSTL

- **Fast Development** JSTL provides many tags that simplify the JSP.
- **Code Reusability** We can use the JSTL tags on various pages.
- **No need to use scriptlet tag** It avoids the use of scriptlet tag.

JSTL Tags

There JSTL mainly provides five types of tags:

Tag Name	Description

Core tags	The JSTL core tag provide variable support, URL management, flow control, etc. The URL for the core tag is **http://java.sun.com/jsp/jstl/core**. The prefix of core tag is **c**.
Function tags	The functions tags provide support for string manipulation and string length. The URL for the functions tags is **http://java.sun.com/jsp/jstl/functions** and prefix is **fn**.
Formatting tags	The Formatting tags provide support for message formatting, number and date formatting, etc. The URL for the Formatting tags is **http://java.sun.com/jsp/jstl/fmt** and prefix is **fmt**.
XML tags	The XML tags provide flow control, transformation, etc. The URL for the XML tags is **http://java.sun.com/jsp/jstl/xml** and prefix is **x**.
SQL tags	The JSTL SQL tags provide SQL support. The URL for the SQL tags is **http://java.sun.com/jsp/jstl/sql** and prefix is **sql**.

JSTL Core Tags List

Tags	Description
c:out	It display the result of an expression, similar to the way <%=...%> tag work.
c:import	It Retrives relative or an absolute URL and display the contents to either a String in 'var',a Reader in 'varReader' or the page.
c:set	It sets the result of an expression under evaluation in a 'scope' variable.
c:remove	It is used for removing the specified scoped variable from a particular scope.
c:catch	It is used for Catches any Throwable exceptions that occurs in the body.
c:if	It is conditional tag used for testing the condition and display the body content only if

	the expression evaluates is true.
c:choose, c:when, c:otherwise	It is the simple conditional tag that includes its body content if the evaluated condition is true.
c:forEach	It is the basic iteration tag. It repeats the nested body content for fixed number of times or over collection.
c:forTokens	It iterates over tokens which is separated by the supplied delimeters.
c:param	It adds a parameter in a containing 'import' tag's URL.
c:redirect	It redirects the browser to a new URL and supports the context-relative URLs.
c:url	It creates a URL with optional query parameters.

Example:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>

  <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
  <%@ page isELIgnored="false" %>
<!DOCTYPE html>
<html>
      <head>
            <title>TagExample</title>
      </head>
      <body>
                  <!-- 1. out tag   -->
                  <!-- 2. set tag -->
                  <c:set var="i" value="23" scope="application"></c:set>
                  <c:out value="${i }"></c:out>


                  <!-- 3. remove tag -->
                  <c:remove var="i"/>
                  <c:out value="${i }">this is default value</c:out>
```

```
                    <hr>

                    <!-- 4. if tag -->
                    <c:set var="i" value="0" scope="application"></c:set>
                    <c:if test="${i==23 }">
                          <h1>Condition is true...</h1>
                    </c:if>

                    <!-- 5. switch,when,otherwise : java switch -->
                    <c:choose>
                          <c:when test="${i>0 }">
                                <h1>this is case 1 : num is positive</h1>
                          </c:when>

                          <c:when test="${i<0 }">
                                <h1>this is case 2 : num is negative </h1>
                          </c:when>

                          <c:otherwise>
                                <h1>this is default case : num is zero</h1>
                          </c:otherwise>
                    </c:choose>

                    <!-- 6. forEach tag :  Iterative statement  -->
                    <c:forEach var="j" begin="1" end="10">
                          <h1>Value of j is  : <c:out value="${j }"></c:out>
</h1>
                    </c:forEach>

                    <!-- 7. url , redirect -->
                    <c:url var="myurl" value="https://www.google.com/search">
                          <c:param name="q" value="servlet tutorial"></c:param>
                    </c:url>
                    <c:out value="${myurl }"></c:out>
                    <c:redirect url="${myurl }"></c:redirect>

      </body>
</html>
```

JSTL Function Tags

The JSTL function provides a number of standard functions, most of these functions are common string manipulation functions.

JSTL Functions	Description

fn:contains()	It is used to test if an input string containing the specified substring in a program.
fn:containsIgnoreCase()	It is used to test if an input string contains the specified substring as a case insensitive way.
fn:endsWith()	It is used to test if an input string ends with the specified suffix.
fn:escapeXml()	It escapes the characters that would be interpreted as XML markup.
fn:indexOf()	It returns an index within a string of first occurrence of a specified substring.
fn:trim()	It removes the blank spaces from both the ends of a string.
fn:startsWith()	It is used for checking whether the given string is started with a particular string value.
fn:split()	It splits the string into an array of substrings.
fn:toLowerCase()	It converts all the characters of a string to lower case.
fn:toUpperCase()	It converts all the characters of a string to upper case.
fn:substring()	It returns the subset of a string according to the given start and end position.
fn:substringAfter()	It returns the subset of string after a specific substring.
fn:substringBefore()	It returns the subset of string before a specific substring.
fn:length()	It returns the number of characters inside a string, or the number of items in a collection.
fn:replace()	It replaces all the occurrence of a string with another string sequence.

Example:

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
    <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
  <%@ page isELIgnored="false" %>
  <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<!DOCTYPE html>
<html>
      <head>
            <title>Fun</title>
      </head>
      <body>
            <h1>Fun page</h1>
            <c:set var="name" value="RohitINDIA"></c:set>
            <h1><c:out value="${name }"></c:out></h1>
            <h1>Length of name is : <c:out value="${fn:length(name)
}"></c:out> </h1>
            <h1><c:out value="${fn:toLowerCase(name) }"></c:out> </h1>
            <h1><c:out value="${fn:toUpperCase(name) }"></c:out> </h1>
            <h1><c:out value="${fn:contains(name,'INDIA') }"></c:out> </h1>
      </body>

</html>
```

JSTL SQL Tags

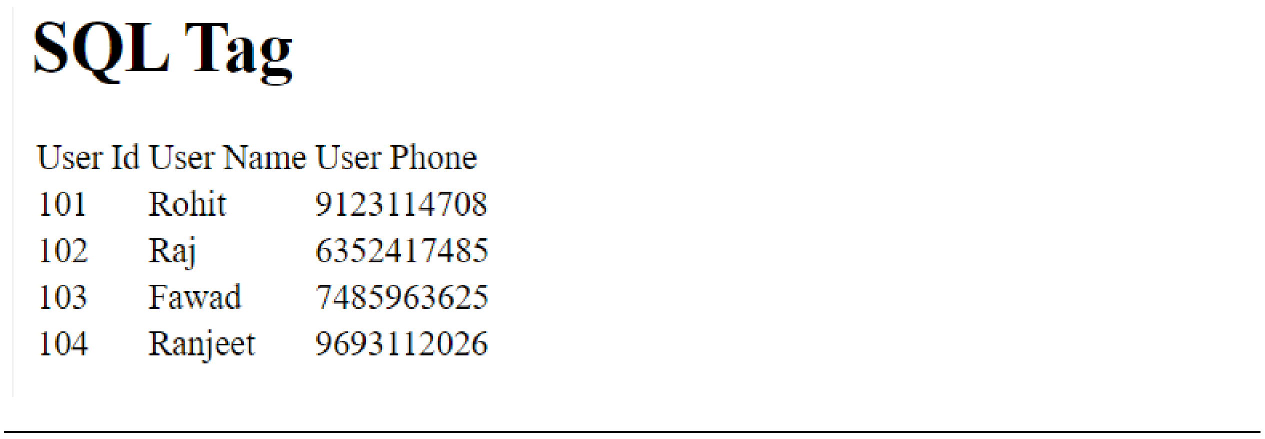
The JSTL sql tags provide SQL support. The SQL tag library allows the tag to interact with RDBMSs (Relational Databases) such as Microsoft SQL Server, mySQL, or Oracle.

SQL Tags	Descriptions
sql:setDataSource	It is used for creating a simple data source suitable only for prototyping.
sql:query	It is used for executing the SQL query defined in its sql attribute or the body.
sql:update	It is used for executing the SQL update defined in its sql attribute or in the tag body.
sql:param	It is used for sets the parameter in an SQL statement to the specified value.

sql:dateParam	It is used for sets the parameter in an SQL statement to a specified java.util.Date value.
sql:transaction	It is used to provide the nested database action with a common connection.

Example :

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
     <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
  <%@ page isELIgnored="false" %>
   <%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql" %>
   <%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<!DOCTYPE html>
<html>
      <head>
            <title>Sql tags</title>
      </head>
      <body>
            <h1>SQL Tag</h1>
            <sql:setDataSource driver="com.mysql.cj.jdbc.Driver"
url="jdbc:mysql://localhost:3306/test" user="root" password="76448"
var="ds"></sql:setDataSource>
            <sql:query var="rs" dataSource="${ds }">select * from
user;</sql:query>

            <table>
                  <tr>
                        <td>User Id</td>
                        <td>User Name</td>
                        <td>User Phone</td>
                  </tr>
                  <c:forEach items="${rs.rows }" var="row">
                        <tr>
                              <td><c:out value="${row.UserID }"></c:out>
</td>
                              <td><c:out value="${row.UserName }"></c:out>
</td>
                              <td><c:out value="${row.Phone }"></c:out> </td>
                        </tr>
                  </c:forEach>
            </table>
      </body>
</html>
```



SQL Tag

```
User Id User Name User Phone
101     Rohit     9123114708
102     Raj       6352417485
103     Fawad     7485963625
104     Ranjeet   9693112026
```