URL ReWritting

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;

/\*\*

\* Servlet implementation class AcceptServlet

\*/

public class AcceptServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public AcceptServlet() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#service(HttpServletRequest request, HttpServletResponse response)

\*/

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

String userName = request.getParameter("user");

String redirectUrl = null;

if(userName!=null)

redirectUrl="displayuser?userName="+userName;

else

redirectUrl="displayuser?userName=guest";

RequestDispatcher dispatcher = request.getRequestDispatcher(redirectUrl);

dispatcher.include(request, response);

}

}

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;

/\*\*

\* Servlet implementation class DisplayUserServlet

\*/

public class DisplayUserServlet extends HttpServlet {

private static final long serialVersionUID = 1L;

/\*\*

\* @see HttpServlet#HttpServlet()

\*/

public DisplayUserServlet() {

super();

// TODO Auto-generated constructor stub

}

/\*\*

\* @see HttpServlet#service(HttpServletRequest request, HttpServletResponse response)

\*/

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

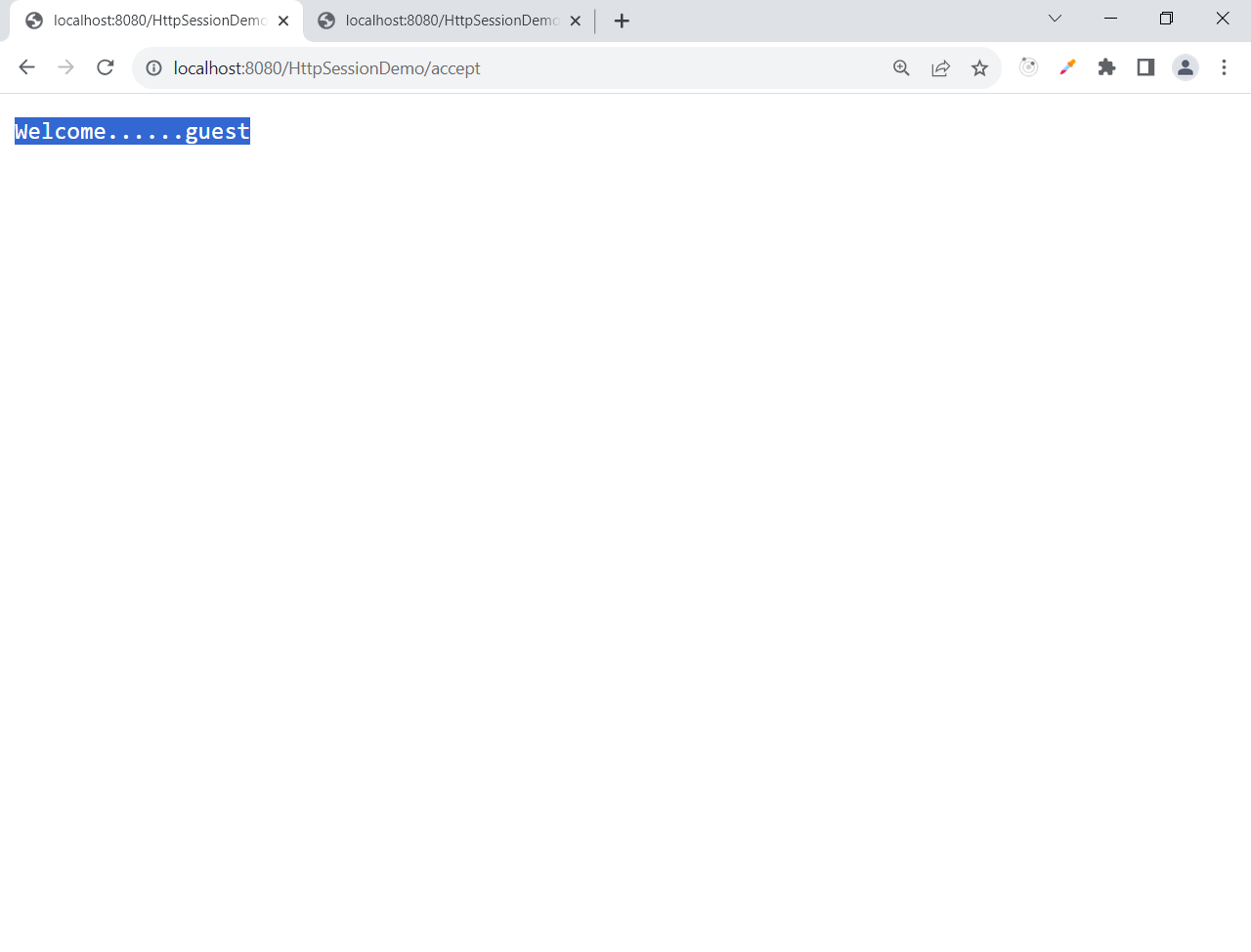
String userName = request.getParameter("userName");

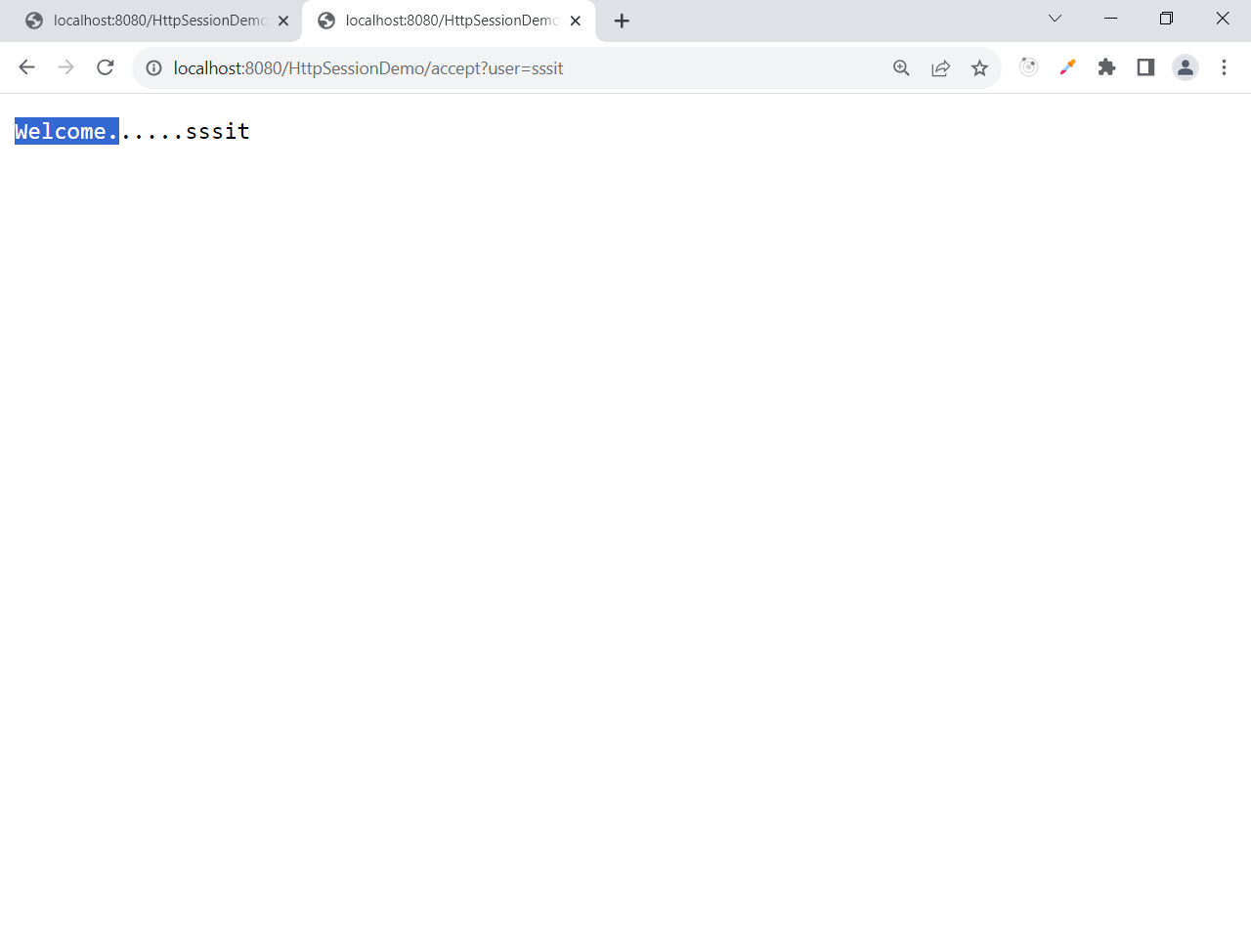
PrintWriter pw = response.getWriter();

pw.println("Welcome......" + userName);

}

}





Hidden Form fields

**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;

/\*\*

\* Servlet implementation class LoginCredentials

\*/

**public** **class** LoginCredentials **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

/\*\*

\* **@see** HttpServlet#HttpServlet()

\*/

**public** LoginCredentials() {

**super**();

// **TODO** Auto-generated constructor stub

}

/\*\*

\* **@see** HttpServlet#service(HttpServletRequest request, HttpServletResponse response)

\*/

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

PrintWriter pw = response.getWriter();

pw.println("<html>");

pw.println("<body>");

pw.println("<form action=validate>");

pw.println("Enter User Name:");

pw.println("<input type=text name=user /><br>");

pw.println("Enter Password:");

pw.println("<input type=password name=pwd /><br>");

pw.println("<input type=hidden name=isRequestFromLogin value=true />");

pw.println("<input type=submit value=login />");

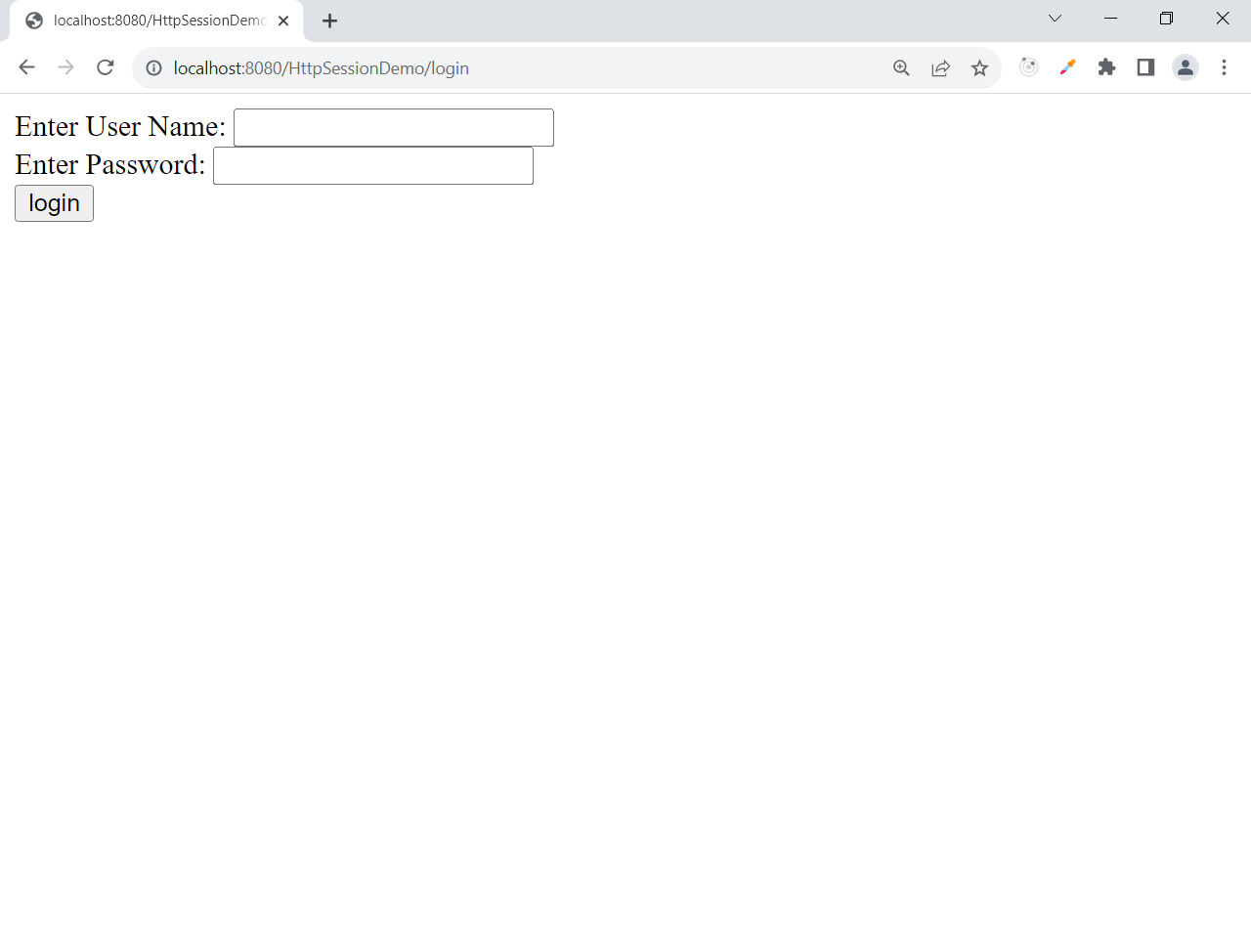
pw.println("</form>");

pw.println("</body>");

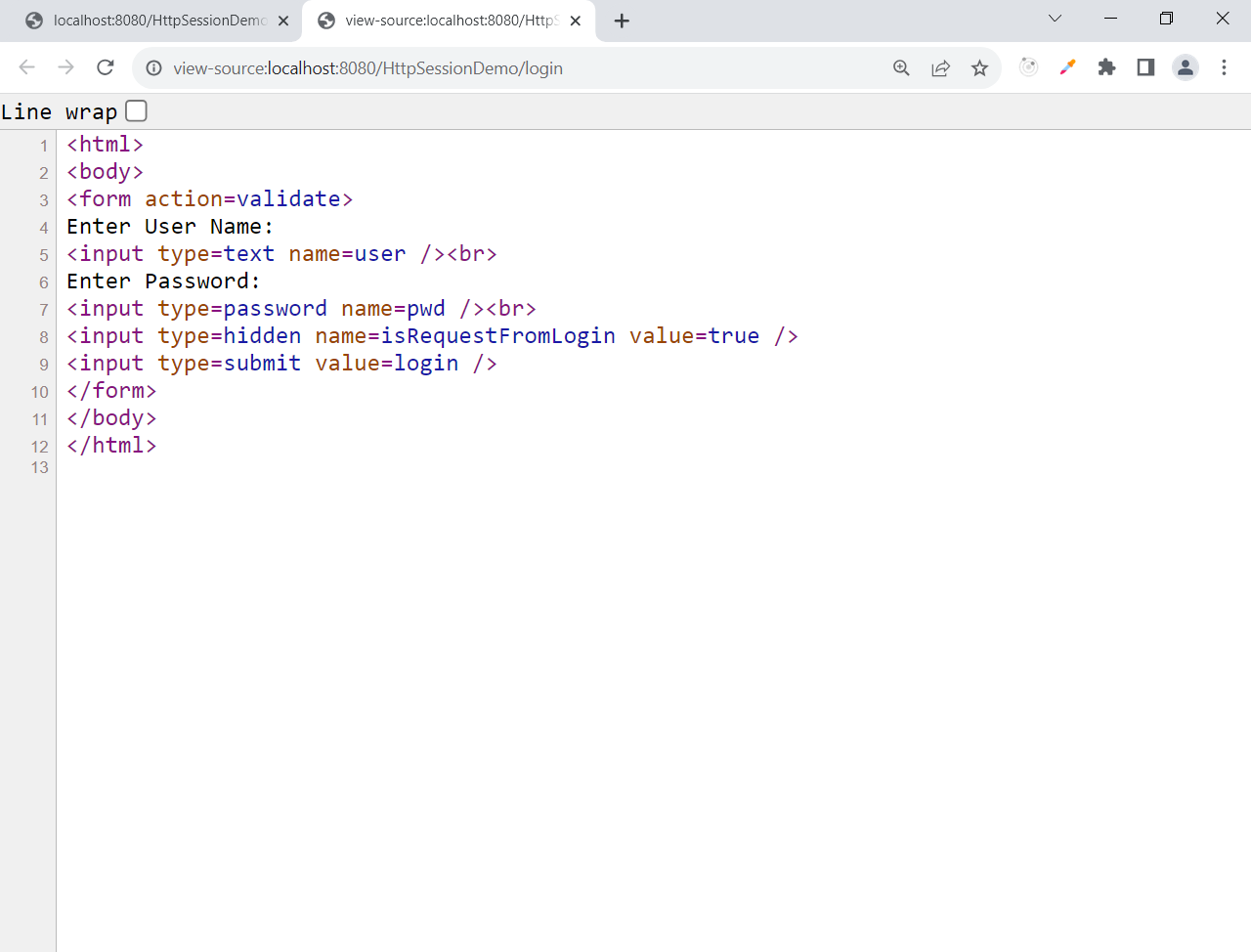
pw.println("</html>");

}

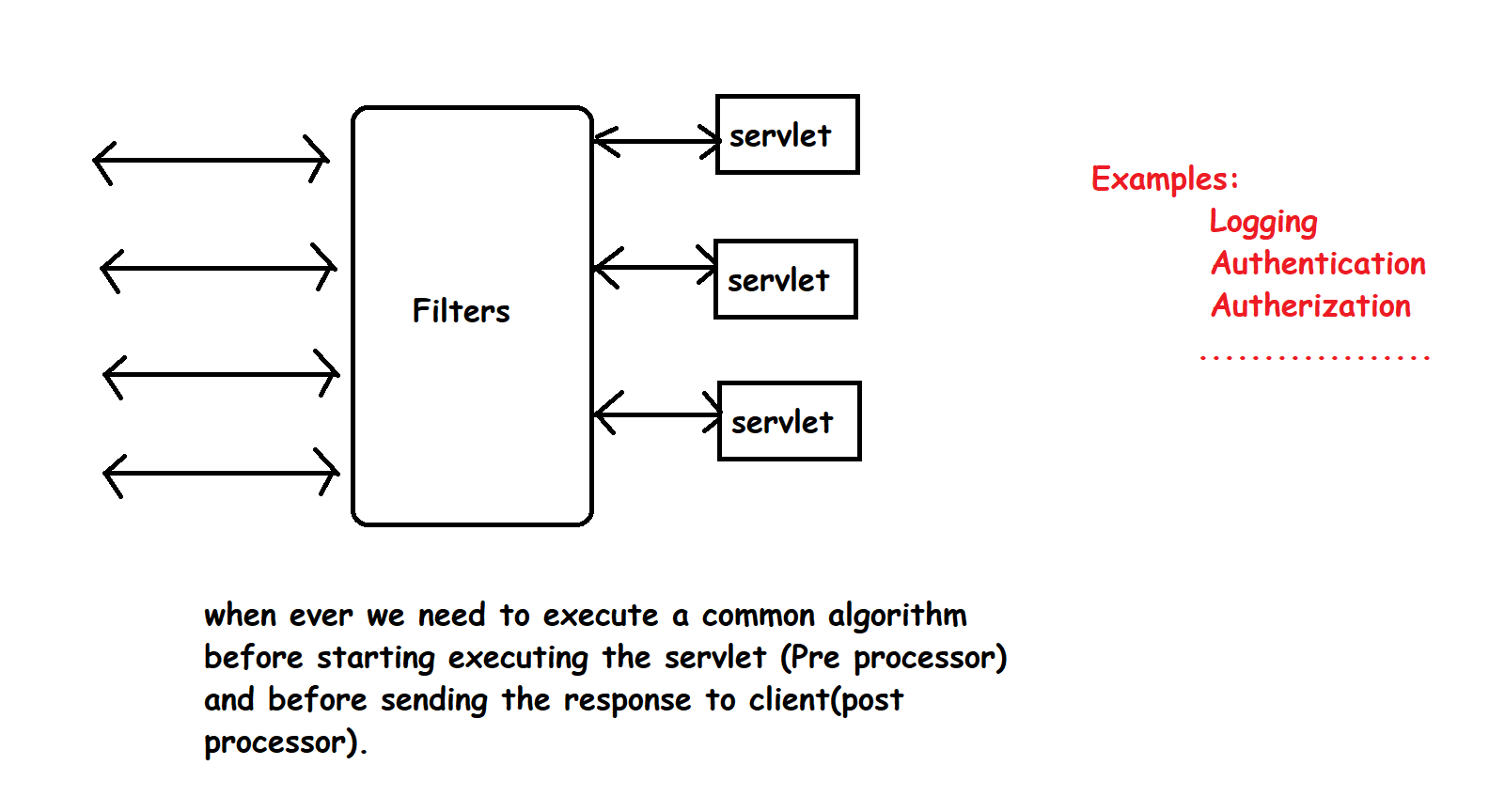
}



Right click 🡪 view page source



Filters

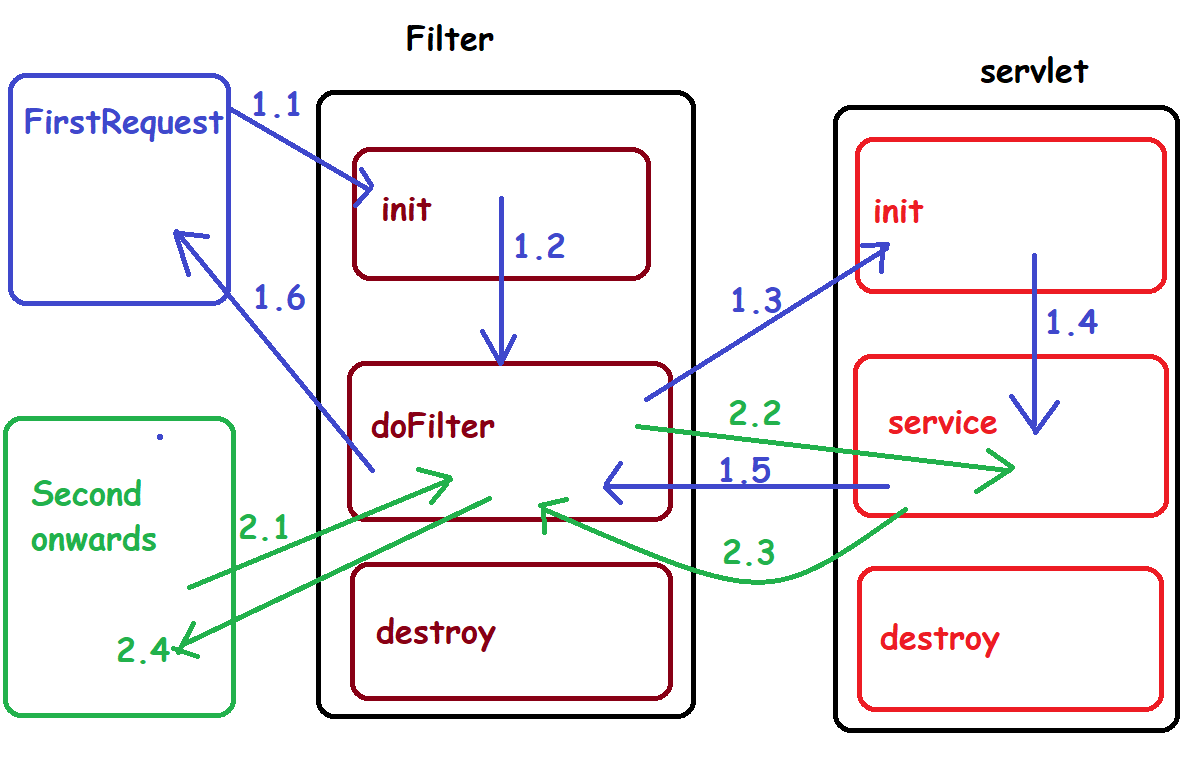


How to define a filter?

Two Steps:

1. Define the filter by implementing Filter interface
2. Configure the filter information in web.xml

Filter Life Cycle



Defination of LoggFilter

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;

/\*\*

\* Servlet Filter implementation class LogingDemo

\*/

public class LoggFilter implements Filter {

/\*\*

\* Default constructor.

\*/

public LoggFilter() {

// TODO Auto-generated constructor stub

}

/\*\*

\* @see Filter#destroy()

\*/

public void destroy() {

System.out.println("Begin Destroy execution......");

}

/\*\*

\* @see Filter#doFilter(ServletRequest, ServletResponse, FilterChain)

\*/

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

System.out.println("Begin DoFilter execution......");

chain.doFilter(request, response);

System.out.println("End of DoFilter execution......");

}

/\*\*

\* @see Filter#init(FilterConfig)

\*/

public void init(FilterConfig fConfig) throws ServletException {

System.out.println("Begin Init execution......");

}

}

Definition of DemoServlet

**import** java.io.IOException;

**import** java.io.PrintWriter;

**import** javax.servlet.ServletConfig;

**import** javax.servlet.ServletException;

**import** javax.servlet.http.HttpServlet;

**import** javax.servlet.http.HttpServletRequest;

**import** javax.servlet.http.HttpServletResponse;

/\*\*

\* Servlet implementation class DemoServlet

\*/

**public** **class** DemoServlet **extends** HttpServlet {

**private** **static** **final** **long** ***serialVersionUID*** = 1L;

/\*\*

\* **@see** HttpServlet#HttpServlet()

\*/

**public** DemoServlet() {

**super**();

// **TODO** Auto-generated constructor stub

}

/\*\*

\* **@see** Servlet#init(ServletConfig)

\*/

**public** **void** init(ServletConfig config) **throws** ServletException {

System.***out***.println("Begin Init execution in Servlet......");

}

/\*\*

\* **@see** Servlet#destroy()

\*/

**public** **void** destroy() {

System.***out***.println("Begin Destroy execution in Servlet......");

}

/\*\*

\* **@see** HttpServlet#service(HttpServletRequest request, HttpServletResponse response)

\*/

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

System.***out***.println("Begin Service execution in Servlet......");

PrintWriter pw = response.getWriter();

pw.println("<h1> Hi... </h1>");

}

}

Configure the servlet and Filter info. In web.xml

Servlet Info configuration

<servlet>

<description></description>

<display-name>DemoServlet</display-name>

<servlet-name>DemoServlet</servlet-name>

<servlet-class>DemoServlet</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name>DemoServlet</servlet-name>

<url-pattern>/demo</url-pattern>

</servlet-mapping>

Filter Info Configuration

<filter>

<display-name> LoggFilter </display-name>

<filter-name>LoggFilter</filter-name>

<filter-class>LoggFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>LoggFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

We are allowed to define multiple filters in a web application

Whenever we defined multiple filters, then the execution order is the order we followed to configure the filters in web.xml

<filter>

<display-name>AuthenticationFilter</display-name>

<filter-name>AuthenticationFilter</filter-name>

<filter-class>AuthenticationFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>AuthenticationFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<filter>

<display-name>LoggFilter</display-name>

<filter-name>LoggFilter</filter-name>

<filter-class>LoggFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>LoggFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

In the above case

1. Authentication Filter
2. LoggFilter

Use case – 2

<filter>

<display-name>LoggFilter</display-name>

<filter-name>LoggFilter</filter-name>

<filter-class>LoggFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>LoggFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

<filter>

<display-name>AuthenticationFilter</display-name>

<filter-name>AuthenticationFilter</filter-name>

<filter-class>AuthenticationFilter</filter-class>

</filter>

<filter-mapping>

<filter-name>AuthenticationFilter</filter-name>

<url-pattern>/\*</url-pattern>

</filter-mapping>

In the above case

1. LoggFilter
2. Authentication Filter

JSP 🡪 Java Server Pages

Servlet means writing a HTML Program inside the java

JSP means writing the Java Program inside HTML

How to develop a simple JSP Page?

Right Click On application 🡪 New 🡪 JSP (to create a JSP)

Right Click On application 🡪 New 🡪 Servlet (to create a servlet)

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

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html>

<html>

<head>

<meta charset=*"ISO-8859-1"*>

<title>Insert title here</title>

</head>

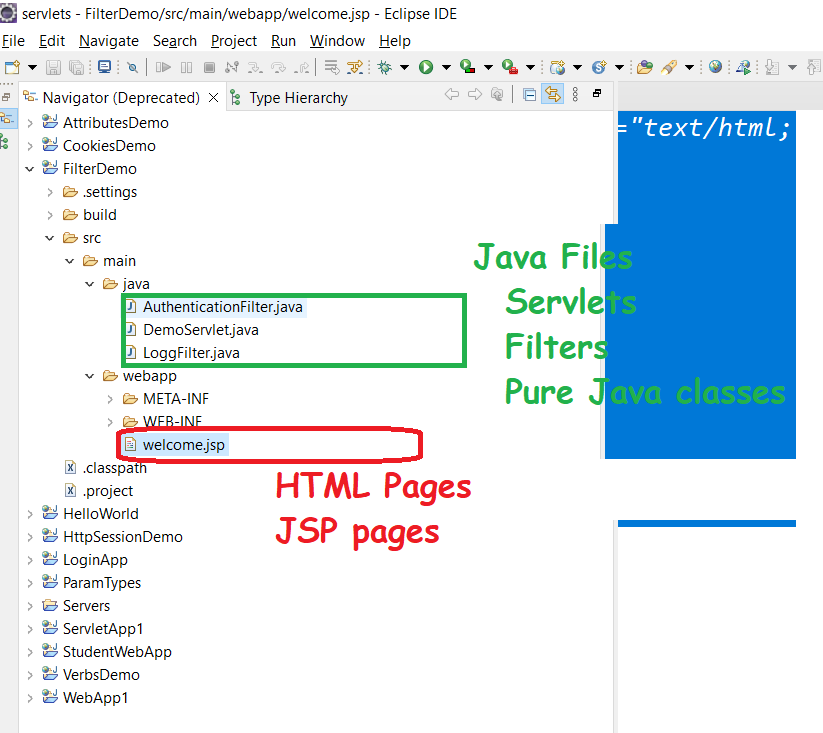
<body>

<h1> Hello JSP</h1>

</body>

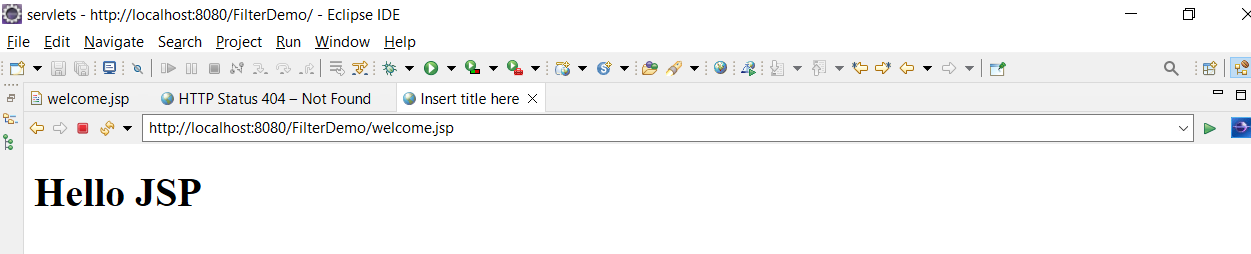
</html>

Location of Servlets and JSPs

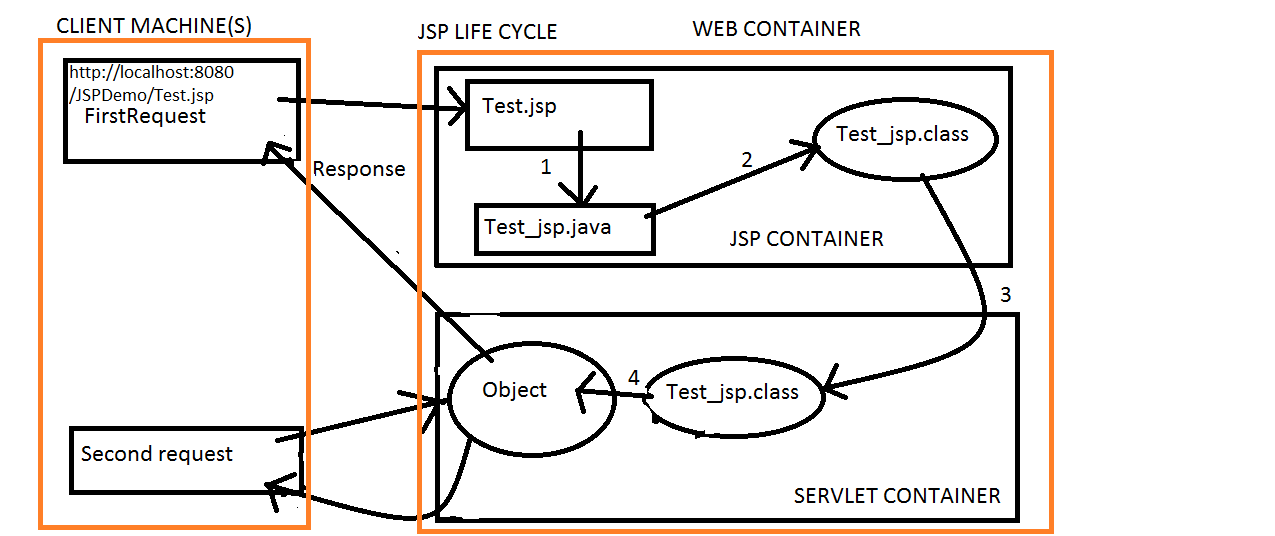


How to run the JSP?

1. Right click on Application 🡪 Run As 🡪 Run on server (once the server is up and running) then
2. In the browser, provide the JSP file name with extension



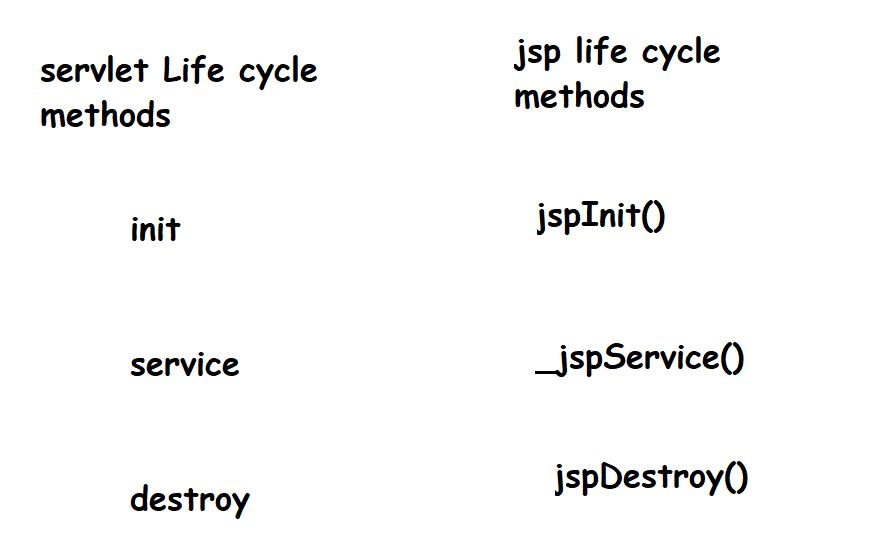
JSP Life cycle

****

The JSP pages follows these phases:

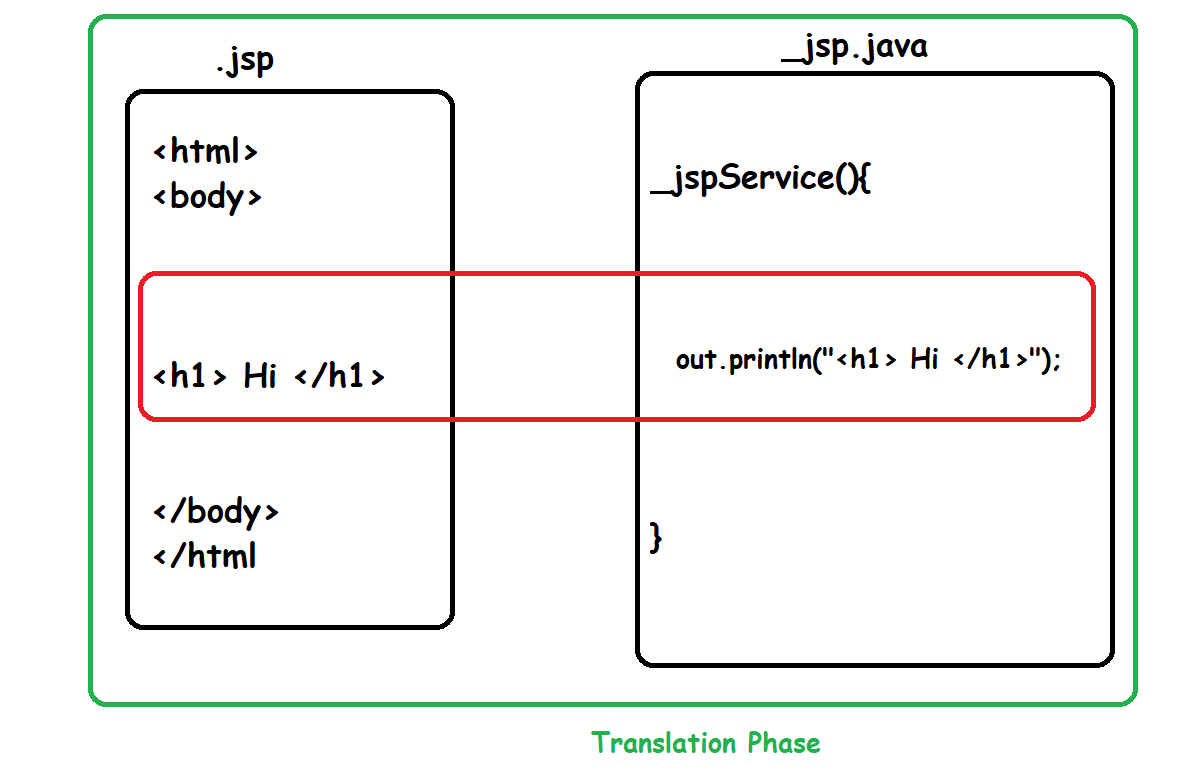
1. Translation of JSP Page(Converting .jsp to .java)
2. Compilation of JSP Page(.java to .class)
3. Loading Phase (class file is loaded by the classloader)
4. Instantiation (Object of the Generated Servlet is created).
5. Initialization ( jspInit() method is invoked by the container).
6. Reqeust processing and Response Generated Phase ( \_jspService() method is invoked by the container).
7. Destroy ( jspDestroy() method is invoked by the container).

Difference between Servlet and JSP life cycle methods



Note: we can’t override \_jspInit() and \_jspDestroy() methods

What ever we have written inside the body tag of jsp page, our JSP container copies that complete content to the \_jspService() method using Java statements



Vocabulary or elements available in JSP

Every thing in JSP is called as Tag

Tag is nothing but a keyword enclosed between < and >

All JSP tags are divided into Three categories depending on their usage:

1. JSP Scripting Tags
2. JSP Directives
3. JSP action tags

