National University of Computer & Emerging Sciences, Karachi

Computer Science Department

Fall 2022, Lab Manual - 08

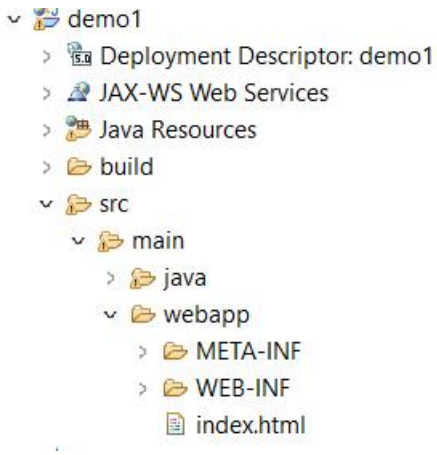
Course Code: SL3001	Course : Software construction and Development
Instructor :	Miss Nida Munawar

Lab # 08

1st step

Right click on project > new > html file >Index.html

It will automatically save inside src



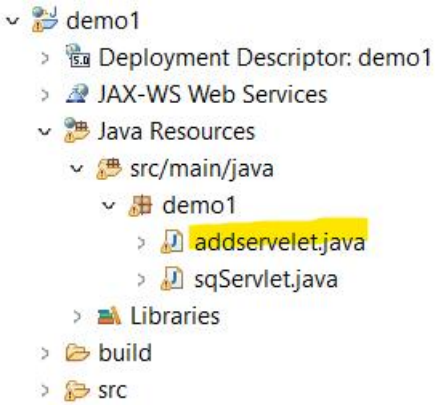

```
<!DOCTYPE html>
<html>
```

```
<body>
hello <br>
<form action = "add"  method = "post">
enter 1st number <input type = "text" name =
"num1"><br>
enter 2nd number <input type = "text" name =
"num2"><br>
<input type = "submit"> <br>
</form>
</body>
</html>
```

2nd step

Right click on project > new > Dynamic web pages >addservelet.java

It will automatically save inside java resources

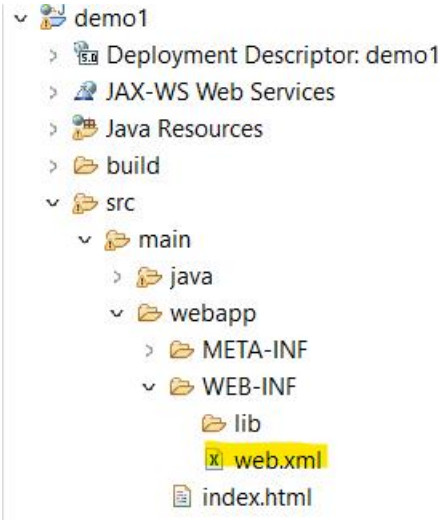


package demo1;

```
import java.io.IOException;

import java.io.PrintWriter;


import jakarta.servlet.RequestDispatcher;

import jakarta.servlet.ServletException;

import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;


public class addservelet extends HttpServlet {

	public void service(HttpServletRequest req , HttpServletResponse res) throws IOException, ServletException {

		int i = Integer.parseInt(req.getParameter("num1"));

		int j = Integer.parseInt(req.getParameter("num2"));

		int k = i+j;

		System.out.println(k); // it will show output on console not on browser

	}}
```

3rd step

Open web.xml and click on source tab



Add <servlet> and <servlet mapping> with the inner tags

In both <servlet> and <servlet mapping> the <servlet-name> must be same

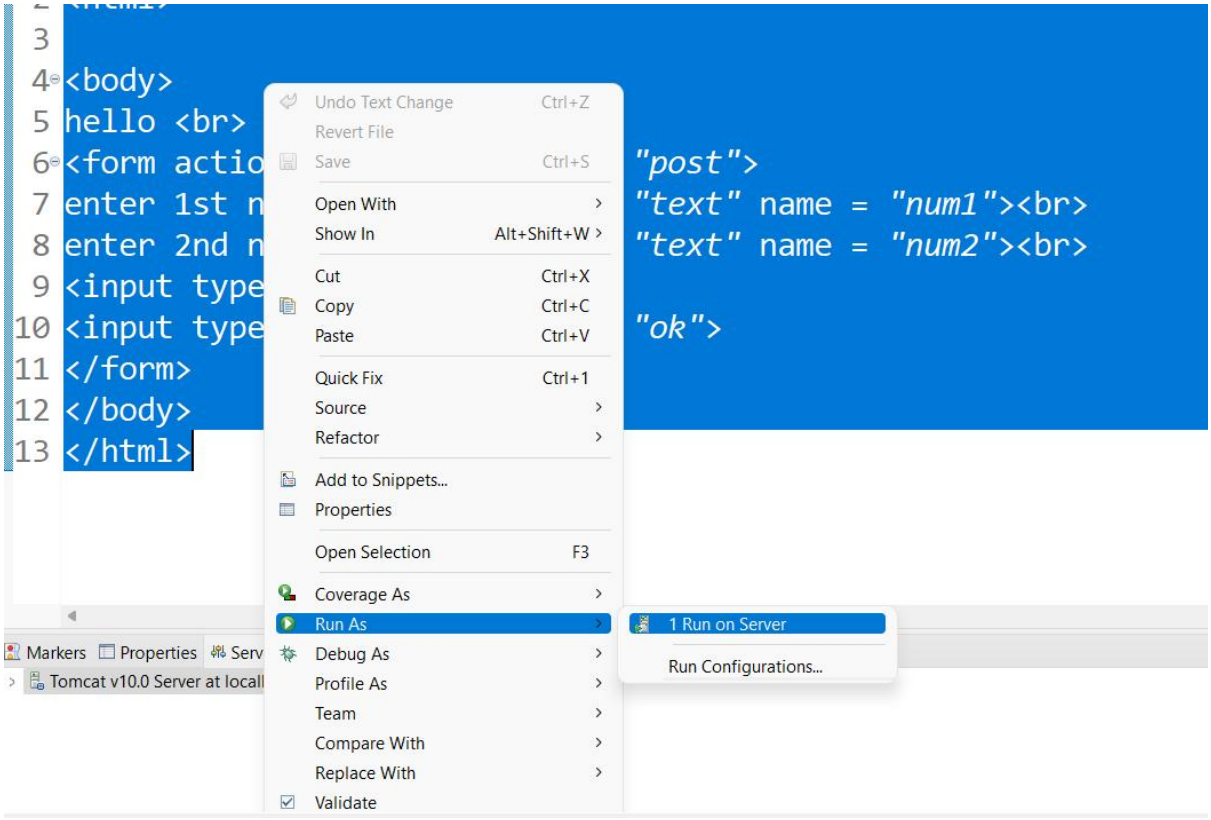
<servlet class> is your package-name.servlet-class

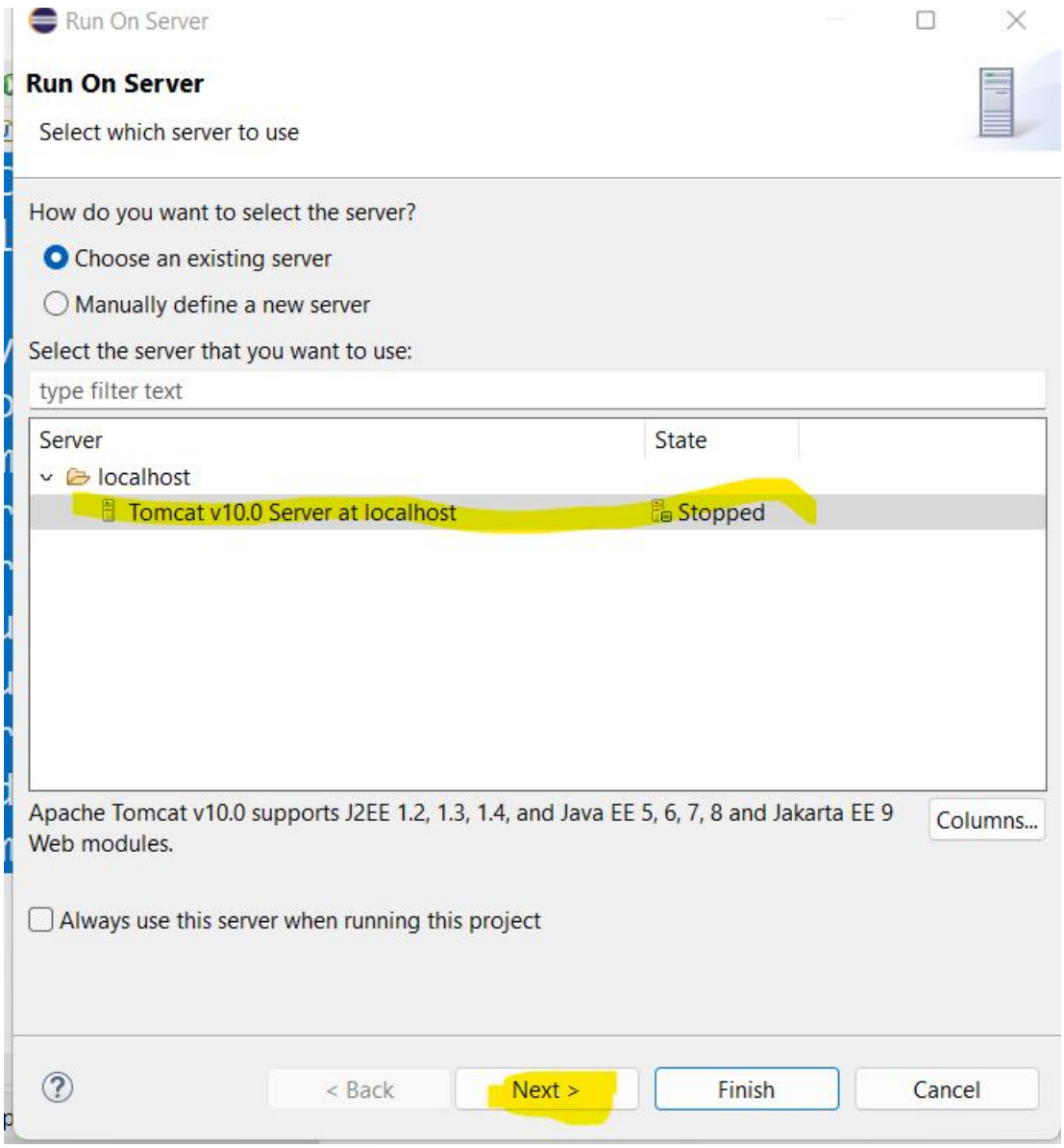
The name of the action you specify inside of your index.html file is <url-pattern>, and it always begins with /.

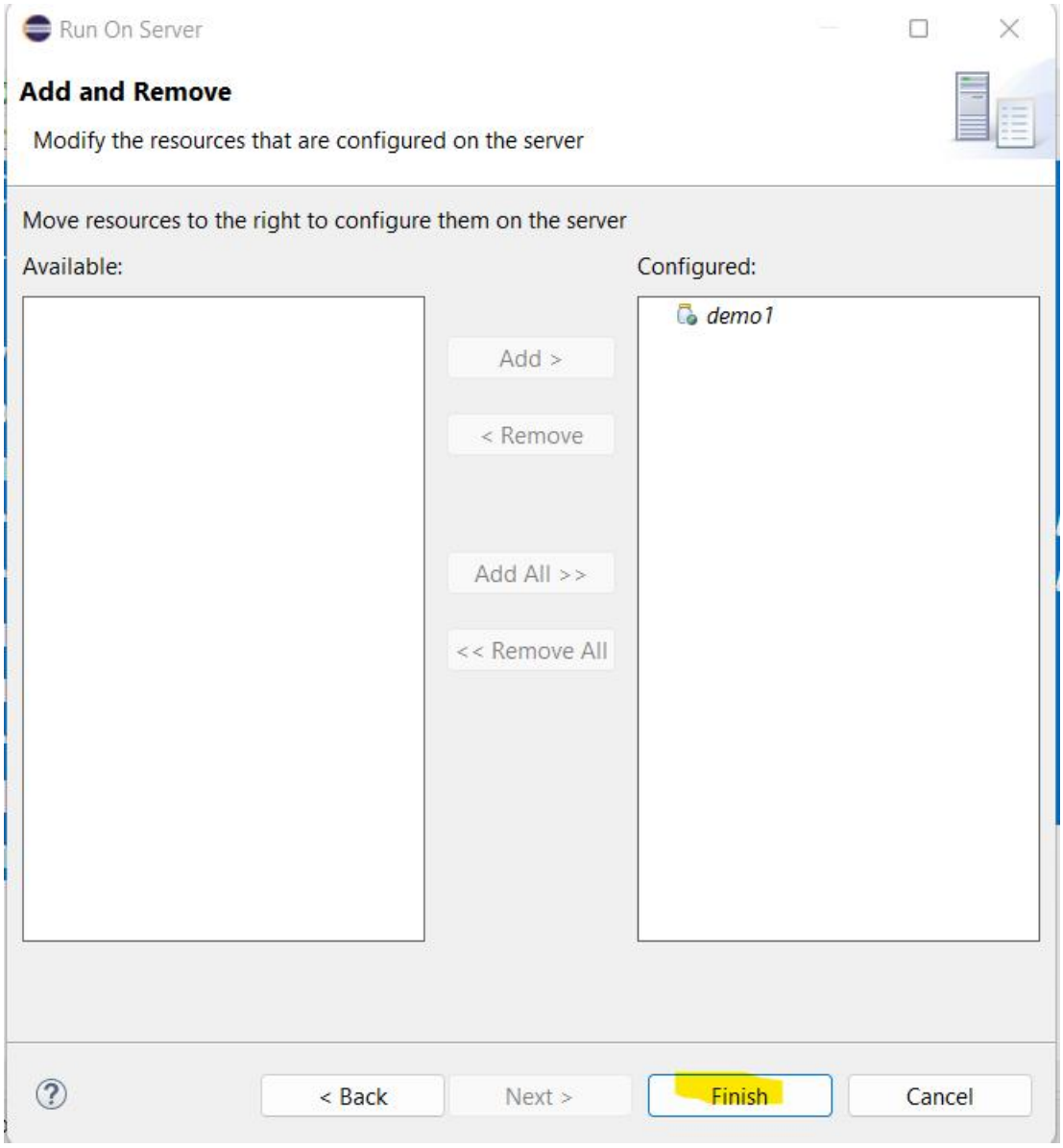
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xmlns="https://jakarta.ee/xml/ns/jakartaee"
xmlns:web="http://xmlns.jcp.org/xml/ns/javaee
"
xsi:schemaLocation="https://jakarta.ee/xml/ns
/jakartaee
https://jakarta.ee/xml/ns/jakartaee/web-
app_5_0.xsd" id="WebApp_ID" version="5.0">
  <servlet>
  <servlet-name> abc </servlet-name>
  <servlet-class>demo1.addservelet</servlet-
class>
```

```
    </servlet>
    <servlet-mapping>
    <servlet-name> abc</servlet-name>
    <url-pattern>/add</url-pattern>
    </servlet-mapping>
</web-app>
```

Right click on your index.html file >




Run On Server
Run On Server
Select which server to use
How do you want to select the server?
Choose an existing server
Manually define a new server
Select the server that you want to use:
type filter text
Server
State
localhost
Tomcat v10.0 Server at localhost
Stopped
Apache Tomcat v10.0 supports J2EE 1.2, 1.3, 1.4, and Java EE 5, 6, 7, 8 and Jakarta EE 9 Web modules.
Columns...
Always use this server when running this project
< Back
Next >
Finish
Cancel



Now it will redirect you to the browser

Printing on browser

Modify your servlet class

package demo1;

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

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

public class addservelet extends HttpServlet {
	public void service(HttpServletRequest req , HttpServletResponse res) throws IOException, ServletException {
		int i = Integer.parseInt(req.getParameter("num1"));
		int j = Integer.parseInt(req.getParameter("num2"));
		int k = i+j;
		PrintWriter out = res.getWriter();
		out.println("add is " + k);

	}

}
```

Also use doGet and doPost method instead of service()

RequestDispatcher in Servlet

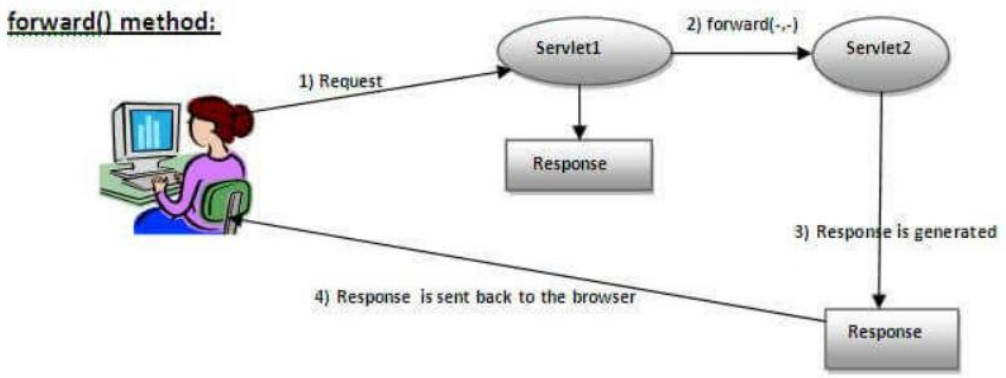
The RequestDispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp. This interface can also be used to include the content of another resource also. It is one of the way of servlet collaboration.

Servlet to Servlet calling using request dispatcher

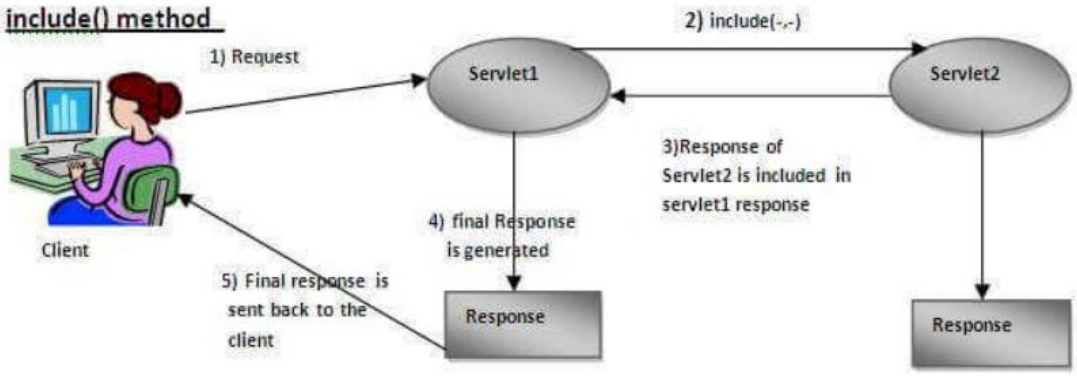
In this calling your client will not be notified that he/she redirects to another servlet RD works by using same req and res object

Two methods

1.



2.



Servlet to Servlet request

Add another java class to your project like you did it before

```
package demo1;


import java.io.IOException;

import java.io.PrintWriter;


import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;


public class sqServlet extends HttpServlet {


                // TODO Auto-generated method stub
```

```
        public void service(HttpServletRequest req , HttpServletResponse res)
throws IOException{


                PrintWriter out = res.getWriter();

                out.println("hello sqrt");

        }

    }
```

Modify your addservelet file

```
package demo1;

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

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

public class addservelet extends HttpServlet
{
    public void service(HttpServletRequest
req , HttpServletResponse res) throws
IOException, ServletException {
        int i =
Integer.parseInt(req.getParameter("num1"));
```

```
        int j =
Integer.parseInt(req.getParameter("num2"));
        int k = i+j;
        RequestDispatcher rd =
req.getRequestDispatcher("sq");
        rd.forward(req, res);
    }

}
```

Modify your web.xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance"
xmlns="https://jakarta.ee/xml/ns/jakartaee"
xmlns:web="http://xmlns.jcp.org/xml/ns/javaee
"
xsi:schemaLocation="https://jakarta.ee/xml/ns
/jakartaee
https://jakarta.ee/xml/ns/jakartaee/web-
app_5_0.xsd" id="WebApp_ID" version="5.0">
  <servlet>
  <servlet-name> abc </servlet-name>
  <servlet-class>demo1.addservelet</servlet-
class>
  </servlet>
  <servlet-mapping>
  <servlet-name> abc</servlet-name>
```

```
  <url-pattern>/add</url-pattern>
  </servlet-mapping>

  <servlet>
  <servlet-name> RD </servlet-name>
  <servlet-class>demo1.sqServlet</servlet-
class>
  </servlet>
  <servlet-mapping>
  <servlet-name> RD</servlet-name>
  <url-pattern>/sq</url-pattern>
  </servlet-mapping>
</web-app>
```

Passing a value from one servlet to another

Modify addservelet file

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

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

public class addservelet extends HttpServlet
{
      public void service(HttpServletRequest req , HttpServletResponse res)
throws IOException, ServletException {
            int i = Integer.parseInt(req.getParameter("num1"));
            int j = Integer.parseInt(req.getParameter("num2"));
            int k = i+j;
            req.setAttribute("k", k);
            RequestDispatcher rd = req.getRequestDispatcher("sq");
            rd.forward(req, res);

      }
```

```
}
```

Modify sqServlet file

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

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

public class sqServlet extends HttpServlet {
	public void service(HttpServletRequest req , HttpServletResponse res)
throws IOException{
		int k = (int) req.getAttribute("k");// you have to cast the obj to
int
		PrintWriter out = res.getWriter();
		out.println("value of k = " + k);
	}

}
```

session management

There are four techniques used in Session tracking:

1. **Cookies**
2. **Hidden Form Field**
3. **URL Rewriting**
4. **HttpSession**

URL Rewriting

In URL rewriting, we append a token or identifier to the URL of the next Servlet or the next resource. We can send parameter name/value pairs using the following format:

url?name1=value1&name2=value2&??

A name and a value 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.

Send Redirect

Servlet to Servlet calling using redirect

In this session management your client will be notified that he/she redirects to another servlet

Modify addservelet file

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

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

public class addservelet extends HttpServlet
{
	public void service(HttpServletRequest req , HttpServletResponse res)
throws IOException, ServletException {
		int i = Integer.parseInt(req.getParameter("num1"));
		int j = Integer.parseInt(req.getParameter("num2"));
		int k = i+j;
res.sendRedirect("sq");

	}


}
```

Modify sqServlet file

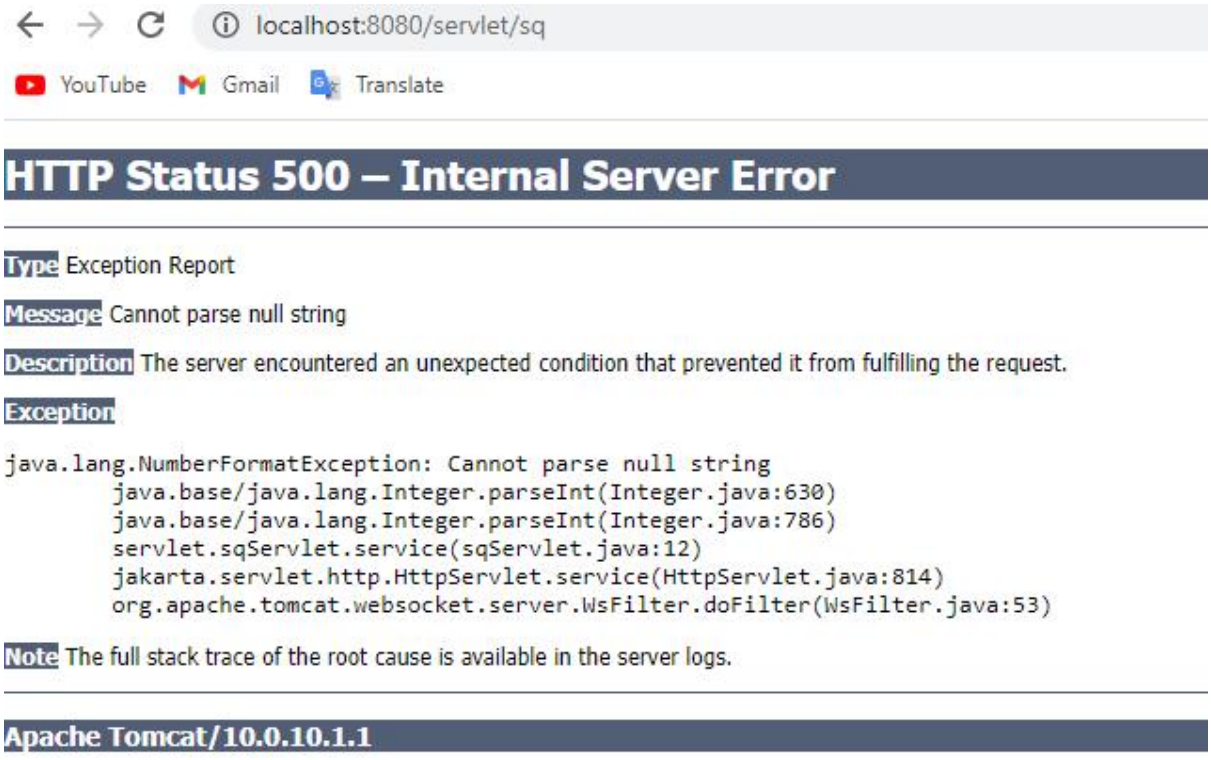
```
import java.io.IOException;
import java.io.PrintWriter;

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

public class sqServlet extends HttpServlet {
	public void service(HttpServletRequest req , HttpServletResponse res)
throws IOException{
```

```
            System.out.println("redirect");
      }

}
```

Passing a value from one servlet to another using URL rewriting

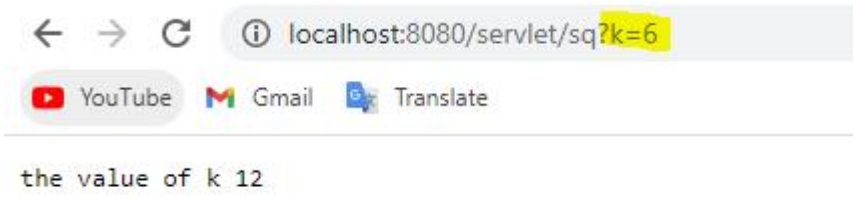
Modify sqServlet file

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

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

public class sqServlet extends HttpServlet {
      public void service(HttpServletRequest req , HttpServletResponse res)
throws IOException{
            int k = Integer.parseInt(req.getParameter("k"));
             k=k+k;
            PrintWriter out = res.getWriter();
            out.println("the value of k " + k);

      }

}
```



You will get null string error because it expects parameter you can pass the parameter in url

?k=6



Second way to pass parameter

Modify addservelet file

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

import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
```

```
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

public class addservelet extends HttpServlet
{
	public void service(HttpServletRequest req , HttpServletResponse res)
throws IOException, ServletException {
		int i = Integer.parseInt(req.getParameter("num1"));
		int j = Integer.parseInt(req.getParameter("num2"));
		int k = i+j;
res.sendRedirect("sq?k=" + k);

	}


}
```

Disadvantage of URL Rewriting

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

2 HttpSession interface

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

Modify addservelet file

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

import jakarta.servlet.RequestDispatcher;
import jakarta.servlet.ServletException;
```

```java
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.servlet.http.HttpSession;

public class addservelet extends HttpServlet
{
   public void service(HttpServletRequest req , HttpServletResponse res)
throws IOException, ServletException {
          int i = Integer.parseInt(req.getParameter("num1"));
          int j = Integer.parseInt(req.getParameter("num2"));
          int k = i+j;
        HttpSession session = req.getSession();
         session.setAttribute("k", k);
        res.sendRedirect("sq");

   }


}
```

Modify sqServlet file

```
import java.io.IOException;

import java.io.PrintWriter;


import jakarta.servlet.http.HttpServlet;

import jakarta.servlet.http.HttpServletRequest;

import jakarta.servlet.http.HttpServletResponse;

import jakarta.servlet.http.HttpSession;


public class sqServlet extends HttpServlet {

      public void service(HttpServletRequest req , HttpServletResponse res)
throws IOException{
```

```
HttpSession session = req.getSession();

int k = (int) session.getAttribute("k");

k=k+k;

PrintWriter out = res.getWriter();

out.println("the value of k " + k);

                }}
```

Cookies in Servlet

A **cookie** is a small piece of information that is persisted between the multiple client requests.

Modify addservelet file

```
import java.io.IOException;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.Cookie;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

public class addservelet extends HttpServlet
{
	public void service(HttpServletRequest req , HttpServletResponse res)
throws IOException, ServletException {
		int i = Integer.parseInt(req.getParameter("num1"));
		int j = Integer.parseInt(req.getParameter("num2"));
		int k = i+j;
		Cookie coo = new Cookie("k", k + "");
		res.addCookie(coo);

	  res.sendRedirect("sq");

	}


}
```

Modify sqServlet file

```
import java.io.IOException;
import java.io.PrintWriter;
import jakarta.servlet.http.Cookie;
```

```
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;


public class sqServlet extends HttpServlet {
	public void service(HttpServletRequest req , HttpServletResponse res)
throws IOException{
		int k = 0;
		Cookie cookies [] = req.getCookies();
		for(Cookie c : cookies ) {
			if(c.getName().equals("k")) {
			k = Integer.parseInt(c.getValue())	;
			}
		}
		k= k *k;
		PrintWriter out = res.getWriter();
		out.println("the value of k " + k);

	}

}
```

ServletContext Interface

An object of ServletContext is created by the web container at time of deploying the project. This object can be used to get configuration information from web.xml file. There is only one ServletContext object per web application.

If any information is shared to many servlet, it is better to provide it from the web.xml file using the **<context-param>** element.

Advantage of ServletContext

Easy to maintain if any information is shared to all the servlet, it is better to make it available for all the servlet. We provide this information from the web.xml file, so if the information is changed, we don't need to modify the servlet. Thus it removes maintenance problem.

Modify addservelet file

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

import jakarta.servlet.ServletContext;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

public class addservelet extends HttpServlet
{
	public void service(HttpServletRequest req , HttpServletResponse res)
throws IOException, ServletException {
		PrintWriter out = res.getWriter();
		out.print("hello " );
		ServletContext ctx  = getServletContext();
		String n = ctx.getInitParameter("name");
		out.println(n);

	}


}
```

These values will be same for all servlets

Modify web.xml file

```xml
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="https://jakarta.ee/xml/ns/jakartaee"
xmlns:web="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
https://jakarta.ee/xml/ns/jakartaee/web-app_5_0.xsd
http://xmlns.jcp.org/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd" id="WebApp_ID" version="5.0">
  <display-name>servlet</display-name>
  <servlet>
  <servlet-name> abc </servlet-name>
  <servlet-class>servlet.addservelet</servlet-class>
  </servlet>
  <servlet-mapping>
  <servlet-name> abc</servlet-name>
  <url-pattern>/add</url-pattern>
  </servlet-mapping>
```

```xml
<context-param>
<param-name> name </param-name>
```

```
<param-value> nida </param-value>

</context-param>


</web-app>
```

ServletConfig Interface

An object of ServletConfig is created by the web container for each servlet. This object can be used to get configuration information from web.xml file.

If the configuration information is modified from the web.xml file, we don't need to change the servlet. So it is easier to manage the web application if any specific content is modified from time to time.

ServletConfig and **ServletContext**, both are objects created at the time of servlet initialization and used to provide some initial parameters or configuration information to the servlet. But, the difference lies in the fact that information shared by ServletConfig is for a specific servlet, while information shared by ServletContext is available for all servlets in the web application.

Modify addservelet file

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

import jakarta.servlet.ServletConfig;
import jakarta.servlet.ServletContext;
import jakarta.servlet.ServletException;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

public class addservelet extends HttpServlet
{
	public void service(HttpServletRequest req , HttpServletResponse res)
throws IOException, ServletException {
		PrintWriter out = res.getWriter();
		out.print("hello " );
		ServletConfig cf  = getServletConfig();
		String n = cf.getInitParameter("name");
		out.println(n);

	}
```

```
}
```
Even we have sevlet context here but it will print the value that is specific to server

Modify web.xml file

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="https://jakarta.ee/xml/ns/jakartaee"
xmlns:web="http://xmlns.jcp.org/xml/ns/javaee"
xsi:schemaLocation="https://jakarta.ee/xml/ns/jakartaee
https://jakarta.ee/xml/ns/jakartaee/web-app_5_0.xsd
http://xmlns.jcp.org/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-
app_2_5.xsd" id="WebApp_ID" version="5.0">
  <display-name>servlet</display-name>
  <servlet>
  <servlet-name> abc </servlet-name>
  <servlet-class>servlet.addservelet</servlet-class>
   <init-param>
<param-name> name</param-name>
<param-value> Nida Munawar</param-value>
</init-param>
  </servlet>
  <servlet-mapping>
  <servlet-name> abc</servlet-name>
  <url-pattern>/add</url-pattern>
  </servlet-mapping>



<context-param>
<param-name> name </param-name>
<param-value> nida </param-value>

</context-param>
</web-app>
```

Servlets - Annotations

So far, you have learnt how Servlet uses the deployment descriptor (web.xml file) for deploying your application into a web server. Servlet API 3.0 has introduced a new package called javax.servlet.annotation. It provides annotation types which can be used for annotating a servlet class. If you use annotation, then the deployment descriptor (web.xml) is not required. But you should use tomcat7 or any later version of tomcat.

Annotations can replace equivalent XML configuration in the web deployment descriptor file (web.xml) such as servlet declaration and servlet mapping. Servlet containers will process the annotated classes at deployment time.

@WebServlet

To declare a servlet.

@WebInitParam

To specify an initialization parameter.

I commented out my all the servlet tags in web.xml

Modify addservelet file

```
import java.io.IOException;
import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.Cookie;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
@WebServlet("/add")
public class addservelet extends HttpServlet
{
	public void service(HttpServletRequest req , HttpServletResponse res)
throws IOException, ServletException {
		int i = Integer.parseInt(req.getParameter("num1"));
		int j = Integer.parseInt(req.getParameter("num2"));
		int k = i+j;
		Cookie coo = new Cookie("k", k + "");
		res.addCookie(coo);

res.sendRedirect("sq");


	}

}
```

Modify sqServlet file

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

import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.Cookie;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;

@WebServlet("/sq")
public class sqServlet extends HttpServlet {
	public void service(HttpServletRequest req , HttpServletResponse res)
throws IOException{

		int k = 0;
		Cookie cookies [] = req.getCookies();
		for(Cookie c : cookies ) {
			if(c.getName().equals("k")) {
			k = Integer.parseInt(c.getValue())	;
			}
		}
		k= k *k;
		PrintWriter out = res.getWriter();
		out.println("the value of k " + k);




	}

}
```

JSP

In servlet if we want to work with both designing and development This is difficult to do in servlet since we need to build separate print statements for each opening and closing tag, and the code would be messy.so instead of using html inside java, we use java inside html called JSP

Example without JSP

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

import jakarta.servlet.ServletException;
import jakarta.servlet.annotation.WebServlet;
import jakarta.servlet.http.HttpServlet;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
@WebServlet("/add")
public class addservelet extends HttpServlet
{
	public void service(HttpServletRequest req , HttpServletResponse res)
throws IOException, ServletException {
```

```
            int i = Integer.parseInt(req.getParameter("num1"));
            int j = Integer.parseInt(req.getParameter("num2"));
            int k = i+j;
            PrintWriter out = res.getWriter();
            out.print("<html><body bgcolor=red>");
            out.println(k);
            out.print("</html> </body>");


    }

}
```

JSP Tutorial

JSP technology is used to create web application just like Servlet technology. It can be thought of as an extension to Servlet because it provides more functionality than servlet

A JSP page consists of HTML tags and JSP tags. The JSP pages are easier to maintain than Servlet because we can separate designing and development.

Here we don't need annotations and Http request and response jsp gives implicit objects

Create a new file add.jsp

```
<%@ 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 bgcolor="pink">
<%
int i = Integer.parseInt(request.getParameter("num1"));
          int j =
Integer.parseInt(request.getParameter("num2"));
          int k = i+j;
          out.println("the value of k = " + k);
          %>
```

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

Modify file index.html

```
<!DOCTYPE html>
<html>

<body>
<form action = "add.jsp"  method = "post">
enter 1st number <input type = "text" name = "num1"><br>
enter 2nd number <input type = "text" name = "num2"><br>
<input type = "submit"> <br>

</body>
</html>
```

Why servlet?

Jsp is converted to Servlet automatically. your server understands servlet not jsp

JSP Scripting elements

The scripting elements provides the ability to insert java code inside the jsp. There are three types of scripting elements:

- scriptlet tag
- expression tag
- declaration tag
- directive tag

JSP scriptlet tag

In JSP, java code can be written inside the jsp page using the scriptlet tag.

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

The code written in scriptlet tag automatically pasted in the service() method inside servlet class

1. <% java source code %>

JSP Declaration Tag

The **JSP declaration tag** is used *to declare fields and methods*

The code written inside the jsp declaration tag is placed outside the service() method of auto generated servlet.

So it doesn't get memory at each request.

Syntax of JSP declaration tag

The syntax of the declaration tag is as follows:

1. <%! field or method declaration %>

Difference between JSP Scriptlet tag and Declaration tag

Jsp Scriptlet Tag	Jsp Declaration Tag
The jsp scriptlet tag can only declare variables not methods.	The jsp declaration tag can declare variables a
The declaration of scriptlet tag is placed inside the _jspService() method.	The declaration of jsp declaration tag is place

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.

Syntax of JSP page directive

```
<%@ page attribute="value" %>
```

Attributes of JSP page directive

- import
- contentType
- extends
- info
- buffer
- language
- isELIgnored
- isThreadSafe
- autoFlush
- session
- pageEncoding
- errorPage
- isErrorPage

1)import

The import attribute is used to import class,interface or all the members of a package.It is similar to import k

Example of import attribute

```
1. <html>
2. <body>
3.
4. <%@ page import="java.util.Date" %>
5. Today is: <%= new Date() %>
```

6.
7. </body>
8. </html>

extends

The extends attribute defines the parent class that will be inherited by the generated servlet.It is rarely used.

JSP expression tag

The code placed within **JSP expression tag** is *written to the output stream of the response*. So you need not write out.print() to write data. It is mainly used to print the values of variable or method.

Syntax of JSP expression tag

1. <%= statement %>

Example of JSP expression tag

In this example of jsp expression tag, we are simply displaying a welcome message.

1. **<html>**

2. <body>
3. <%= "welcome to jsp" %>
4. </body>
5. </html>

Example with all tags

```
<%@ page language="java"
contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html>
<!-- directive tag -->
<%@page import = "java.util.*"  %>
<html>
<head>
<meta charset="ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<!-- declarative tag -->
<%! int i = 4;%>>
<!-- Scriptlet tag -->
<% out.print(4+4);
Scanner s = new Scanner(System.in);
%>

<!-- Expression tag -->
the value of i is <%= i %>>
</body>
</html>
```

to find servlet(.java) files converted from jsp

```
<%=getClass().getResource(getClass().getSimpl
eName() + ".class")%>
```