Class 1

Servlets

Web Application

A web application is a collection of web resource programs having the capability to generate web pages.

We have two types of web pages.

1) Static web pages

A web page with fixed content is called static web page.

A static web page is also known as passive web page.

Ex: Home page
AboutUs page
Contact page
Facebook login page
and etc.

2) Dynamic web pages

A web page with dynamic content is called dynamic web page.

A dynamic web page is also known as active web page.

Ex: Live cricket score page
Stock Market share value page
news page
and etc.

We have two types of web resource programs.

1) Static web resource programs

Static web resource programs are used to generate static web pages.

Ex: HTML program
CSS program
Bootstrap program
Angularjs program and etc.

2) Dynamic web resource programs

Dynamic web resource programs are used to generate dynamic web pages.

Ex: Servlet program
JSP program
and etc.

Based on the position and execution these web resource programs are divided into two types.

1) Client side web resource program

A web resource program which executes at client side is called client side web resource program.

All static web resource programs are called client side web resource programs.

Ex: HTML program
CSS program
Bootstrap program
angularjs program
and etc.

2) Server side web resource program

A web resource program which executes at server side is called server side web resource program.

All dynamic web resource programs are called servers side web resource programs.

Ex: servlet program
jsp program
and etc

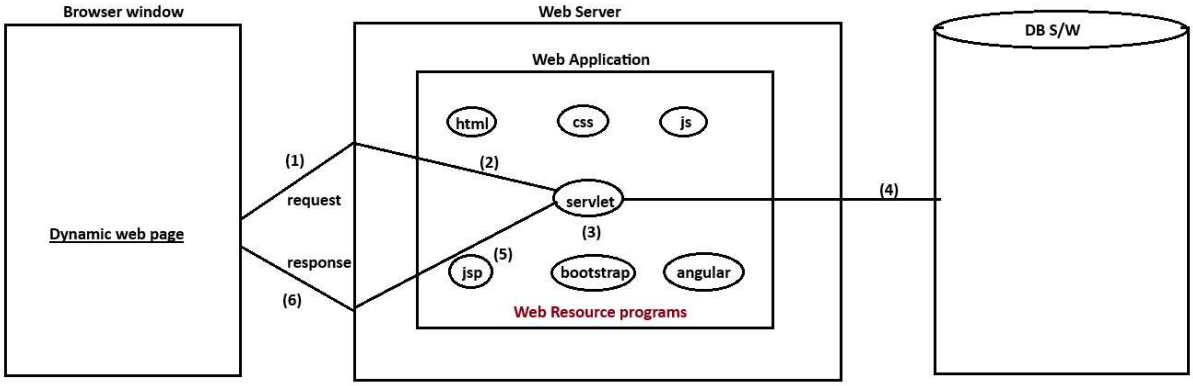
Class2

Web application and Web Resource program execution

Java program will execute manually.

Web application and web resource program will execute when we give the request.

Diagram: servlet2.1



With respect to the diagram

1) Enduser gives the request to web resource program.

2) Webserver traps that request and passes that request to appropriate web resource program.

3) Web resource program will execute the logic to process the request.

4) Web resource program communicates with database software if neccessary.

5) Web resource program gives output to web server.

6) Web server send the output to browser window as dynamic response.

Note:

The process of keeping the application in a server is called deployment and reverse is called undeployment.

Web Server

It is a installable software which is used to automate whole process of web application and web resource program execution.
Ex: Tomcat, Resin and etc.
Responsibilities of web server

1) It takes contineous request from client.

2) It pass the request to appropriate web resource program.

3) It provides environment to deploy and undeploy the web application.

4) It allows client side web resource programs to execute in a browser window.

5) It will add middleware services only to deployed web applications.

6) It gather the result and send to browser window as dynamic response.

7) It is used to automate whole process of web application and web resource program execution.

To execute java program we required JRE/JVM.

To execute servlet program we required servlet container.

To execute jsp program we required jsp container.

Web Container

A web container is a software application or a program which is used to manage whole life cycle of web resource program i.e from birth to death.

Servlet container manage whole life cycle of servlet program.

JSP container manage whole life cycle of jsp program.

Some part of industry consider servlet container and jsp container are web containers.

Every server is designed to support servlet container and jsp container. So we don't need to arrange it seperately.

Tomcat

Version	:	9.x
Creator	:	James Duncan Davidson
Vendor	:	Apache Software Foundation
Website	:	https://tomcat.apache.org
Port No	:	8080
Servlet container	:	Catalina
Jsp container	:	Jasper

Download link :

https://drive.google.com/file/d/1u547booDvVY630rN4drEQ8c8lU0In7T6/view?usp=drive_link

Tomcat is not a container. It is a server containing servlet container and jsp container.

Tomcat installation describes four things.

1) Http Connector port

2) Adminstrator username and password

3) JDK location

4) Tomcat installation location

Tomcat installation

Double click to Tomcat software --> yes --> Next --> I Agree --> select Full --> Next --->
Http Connector port : 8080

adminstrator username : admin
password : admin --> Next --> Next --> Install.

How to setup tomcat server to manual mode

goto services --> click to apache tomcat --> click to stop link --> double click to apache tomcat ---> startup type : manual --> apply --> ok.

Servlet

Servlet is a dynamic web resource program which enhanced the funcationality of web server or application server.

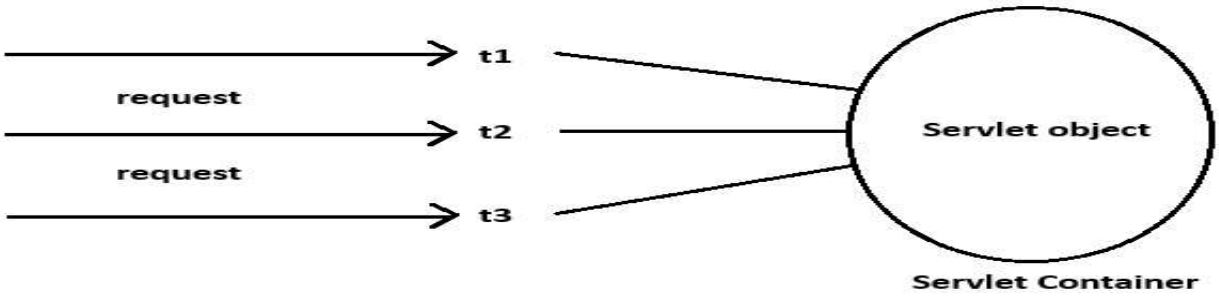
or

Servlet is a java based dynamic web resource program which is used to generate dynamic web pages.

or

Servlet is a single instance multithread java based web resource program which is used to develop web applications.

Diagram: servlet2.2



Servlet API

API is a collection of packages.

ex:

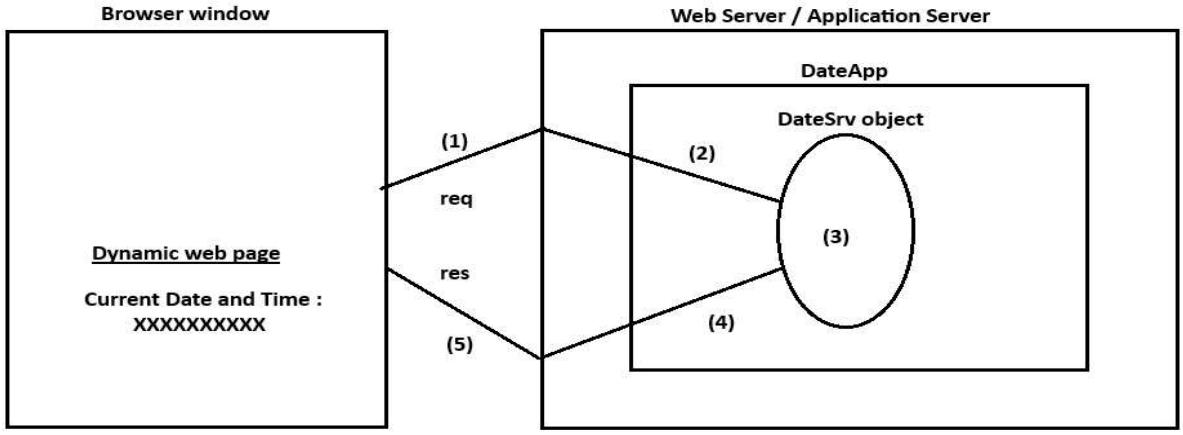
javax.servlet
javax.servlet.http

Servlet important terminologies

javax.servlet.Servlet(I)
|
|
javax.servelt.GenericServlet(AC)
|
|
javax.servlet.http.HttpServlet(C)

First web application development having servlet program as web resource program

Diagram: servlet2.3



Deployment Directory Structure

```
DateApp
|
|---Java Resources
        |
        |------src
                |
                |---com.ihub.www
                        |
                        |---DateSrv.java

|---WebContent
        |
        |------WEB-INF
                |
                |---web.xml
```

Note:
In above application we need to add "servlet-api.jar" file in project build path.

step1:

Launch eclipse IDE by choosing workspace location.

step2:

Create a dynamic web project i.e DateApp.

ex:

File --> New --> Dynamic web project --> Project Name : DateApp
---> Next --> Next --> select generate web.xml file --> Finish.

Step3:

Add "servlet-api.jar" file in project build path.

ex:

right click to DateApp --> Build path --> configure build path -->
libraries --> click to classpath --> Add external jars -->
select servlet-api.jar file --> open --> Apply & close.

step4:

Add Tomcat9 server to eclipse IDE.

ex:

window --> preferences --> server --> Runtime Environments -->
click to add button --> select Apache Tomcat 9 --> Next -->
select destination folder --> Finish --> apply and close.

step5:

Create a "com.ihub.www" package inside "JavaResources/src" folder.

ex:

right click to src folder --> new --> package --> Name : com.ihub.www --> Finish.

step6:

Create a DateSrv.java file inside "com.ihub.www" package.

ex:

right click to com.ihub.www package --> new --> class --> Name : DateSrv --> Finish.

DateSrv.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;

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

public class DateSrv extends GenericServlet
{
	public void service(ServletRequest req,ServletResponse res)throws
ServletException,IOException
	{
		PrintWriter pw=res.getWriter();
		res.setContentType("text/html");

		Date d=new Date();

		pw.println("<center><h1>Current Date and Time <br> "+d+" </h1></center>");

		pw.close();
	}
}
```

step7:

Configure each servlet program in web.xml file.

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <servlet>
                <servlet-name>DateSrv</servlet-name>
                <servlet-class>com.ihub.www.DateSrv</servlet-class>
  </servlet>
  <servlet-mapping>
                <servlet-name>DateSrv</servlet-name>
                <url-pattern>/test</url-pattern>
  </servlet-mapping>

</web-app>
```

step8:

Run dynamic web project.

ex:

right click to DateApp --> run as --> run on server --> select Apache Tomcat 9 --> next --> Finish.

step9:

Test the application by using below request url.

ex:

http://localhost:2525/DateApp/test

Note:

If we do any mistake in web.xml file then we will get 404 Error.
If we do any mistak in servlet program the nwe will get 500 Error.

Class3

Types of url patterns

Url pattern will hide servlet name, technology name from the outsider for security reason.

Our client, other web resource programs and web server will recognize each servlet program by using url pattern.

We have three types of url patterns.

1) **Exact match** url pattern

2) **Directory match** url pattern

3) **Extension match** url pattern

Every server is designed to support above three url patterns.

1) Exact match url pattern

It starts with '/' symbol followed by a name.

ex:

web.xml :- <url-pattern>/test</url-pattern>

Request url

http://localhost:2525/DateApp/test // valid
http://localhost:2525/DateApp/best // invalid
http://localhost:2525/DateApp/x/test // invalid

2) Directory match url pattern

It starts with '/' symbol and ends with '*' symbol.

ex:

web.xml

<url-pattern>/x/y/*</url-pattern>

Request url

http://localhost:2525/DateApp/x/y/z // valid
http://localhost:2525/DateApp/x/y/z/test // valid
http://localhost:2525/DateApp/y/x/z // invalid

3) Extension match url pattern

It starts with '*' symbol having some extension.

Ex: **web.xml**

<url-pattern>*.do</url-pattern>

Request url

http://localhost:2525/DateApp/temp // invalid
http://localhost:2525/DateApp/temp.do // valid
http://localhost:2525/DateApp/x/y/z.do // valid

MIME Types

MIME stands for **Multipurpose Internet Mail Extension.**

MIME **describes in how many formats we can display the output in servlets.**

There are four ways to display the outputs in servlets.

1) text/html

It is used to display the output in html format.

2) text/xml

It is used to display the output in xml format.

3) application/ms-word

It is used to display the output in word format.

4) application/vnd.ms-excel

It is used to display the output in excel format.

Deployment Directory Structure

```
MIMEApp
|
|---Java Resources
        |
        |------src
                |
                |---com.ihub.www
                        |
                        |---TestSrv1.java
                        |---TestSrv2.java
                        |---TestSrv3.java
                        |---TestSrv4.java

|---WebContent
        |
        |------WEB-INF
                |
                |---web.xml
```

Note:
In above application we need to add "servlet-api.jar" file in project build path.

TestSrv1.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class TestSrv1 extends GenericServlet
{
	public void service(ServletRequest req,ServletResponse res)throws
ServletException,IOException
	{
		PrintWriter pw=res.getWriter();
		res.setContentType("text/html");

		pw.println("<table border='1'>");
		pw.println("<tr><th>No</th><th>Name</th><th>Address</th></tr>");
		pw.println("<tr><td>101</td><td>Alan</td><td>Florida</td></tr>");
		pw.println("<tr><td>102</td><td>Jose</td><td>Texas</td></tr>");
		pw.println("<tr><td>103</td><td>Mark</td><td>Vegas</td></tr>");
		pw.println("</table>");

		pw.close();
	}
}
```

TestSrv2.java

```
package com.ihub.www;

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

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

```
public class TestSrv2 extends GenericServlet
{
	public void service(ServletRequest req,ServletResponse res)throws
ServletException,IOException
	{
		PrintWriter pw=res.getWriter();
		res.setContentType("text/xml");

		pw.println("<table border='1'>");
		pw.println("<tr><th>No</th><th>Name</th><th>Address</th></tr>");
		pw.println("<tr><td>101</td><td>Alan</td><td>Florida</td></tr>");
		pw.println("<tr><td>102</td><td>Jose</td><td>Texas</td></tr>");
		pw.println("<tr><td>103</td><td>Mark</td><td>Vegas</td></tr>");
		pw.println("</table>");

		pw.close();
	}
}
```

TestSrv3.java

```
package com.ihub.www;

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

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

public class TestSrv3 extends GenericServlet
{
	public void service(ServletRequest req,ServletResponse res)throws
ServletException,IOException
	{
		PrintWriter pw=res.getWriter();
		res.setContentType("application/ms-word");

		pw.println("<table border='1'>");
		pw.println("<tr><th>No</th><th>Name</th><th>Address</th></tr>");
		pw.println("<tr><td>101</td><td>Alan</td><td>Florida</td></tr>");
		pw.println("<tr><td>102</td><td>Jose</td><td>Texas</td></tr>");
```

```
                pw.println("<tr><td>103</td><td>Mark</td><td>Vegas</td></tr>");
                pw.println("</table>");

                pw.close();
        } }
```

TestSrv4.java

```
package com.ihub.www;

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

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

public class TestSrv4 extends GenericServlet
{
        public void service(ServletRequest req,ServletResponse res)throws
ServletException,IOException
        {
                PrintWriter pw=res.getWriter();
                res.setContentType("application/vnd.ms-excel");

                pw.println("<table border='1'>");
                pw.println("<tr><th>No</th><th>Name</th><th>Address</th></tr>");
                pw.println("<tr><td>101</td><td>Alan</td><td>Florida</td></tr>");
                pw.println("<tr><td>102</td><td>Jose</td><td>Texas</td></tr>");
                pw.println("<tr><td>103</td><td>Mark</td><td>Vegas</td></tr>");
                pw.println("</table>");
                pw.close();
        }
}
```

Note: If a web application contains four servlet programs then each servlet program we need to configure in web.xml file using multiple <servlet> and <servlet-mapping> tags.

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
```

```
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <servlet>
                <servlet-name>TestSrv1</servlet-name>
                <servlet-class>com.ihub.www.TestSrv1</servlet-class>
  </servlet>
  <servlet-mapping>
                <servlet-name>TestSrv1</servlet-name>
                <url-pattern>/html</url-pattern>
  </servlet-mapping>

  <servlet>
                <servlet-name>TestSrv2</servlet-name>
                <servlet-class>com.ihub.www.TestSrv2</servlet-class>
  </servlet>
  <servlet-mapping>
                <servlet-name>TestSrv2</servlet-name>
                <url-pattern>/xml</url-pattern>
  </servlet-mapping>

  <servlet>
                <servlet-name>TestSrv3</servlet-name>
                <servlet-class>com.ihub.www.TestSrv3</servlet-class>
  </servlet>
  <servlet-mapping>
                <servlet-name>TestSrv3</servlet-name>
                <url-pattern>/word</url-pattern>
  </servlet-mapping>
  <servlet>
                <servlet-name>TestSrv4</servlet-name>
                <servlet-class>com.ihub.www.TestSrv4</servlet-class>
  </servlet>
  <servlet-mapping>
                <servlet-name>TestSrv4</servlet-name>
                <url-pattern>/excel</url-pattern>
  </servlet-mapping>

</web-app>
```

Request url

http://localhost:2525/MIMEApp/html
http://localhost:2525/MIMEApp/xml
http://localhost:2525/MIMEApp/word

http://localhost:2525/MIMEApp/excel

Types of communication

We can communicate to servlet program in three ways.

1) Browser to servlet communication

2) HTML to servlet communication

3) Servlet to servlet communication

In browser to servlet communication **we need to type our request url in browser address bar.** But typing request url in browser address bar is quit complex.

To overcome this limitation **we need to HTML to servlet communication.**

In HTML to servlet communication **we can give the request to servlet program via HTML based hyperlinks and form pages.**

In HTML based hyperlink to servlet communication we need to type our request url as href url.

Ex:

```
<a href="http://localhost:2525/MIMEApp/html"> click here </a>
```

In HTML based form page to servlet communication we need to type our request url as action url.

ex:

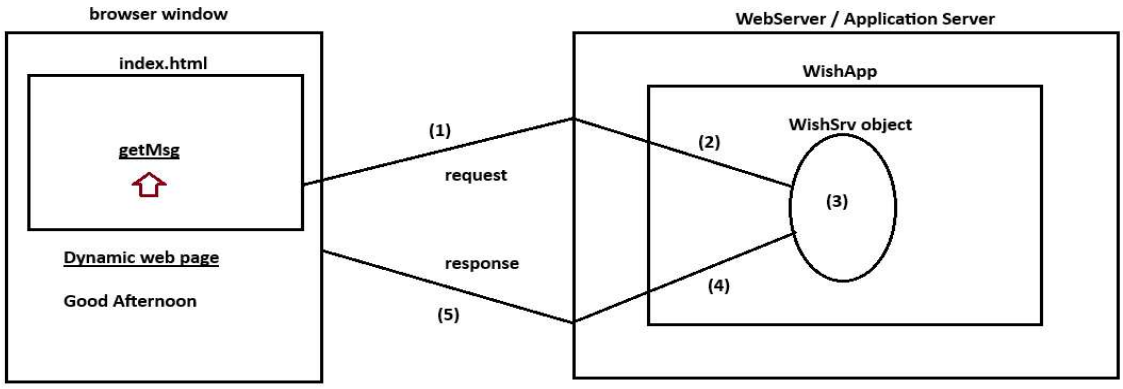
```
<form action="http://localhost:2525/MIMEApp/html">
        -
        -
        -
</form>
```

A request which is generated by using hyperlink does not carry the data.

A request which is generated by using form page will carry the data.

Example application on HTML based hyperlink to servlet communication

Diagram: servlet3.1



Deployment Directory Structure

WishApp
|
|---Java Resources
|
|------src
|
|---com.ihub.www
|
|---WishSrv.java
|---WebContent
|
|------index.html
|
|------WEB-INF
|
|---web.xml

Note:

In above application we need to add "servlet-api.jar" file in project build path.

It is never recommanded to extends a class with GenericServlet class because it won't give HTTP protocol features.

It is always recommanded to extends a class with HttpServlet class because it gives HTTP protocol features.

index.html

```
        <h1>
                <a href="test"> getMsg </a>
        </h1>
</center>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <servlet>
        <servlet-name>WishSrv</servlet-name>
        <servlet-class>com.ihub.www.WishSrv</servlet-class>
  </servlet>
  <servlet-mapping>
        <servlet-name>WishSrv</servlet-name>
        <url-pattern>/test</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
        <welcome-file>index.html</welcome-file>
  </welcome-file-list>

</web-app>
```

WishSrv.java

```
package com.ihub.www;
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Calendar;

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

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

```
	{
		PrintWriter pw=res.getWriter();
		res.setContentType("text/html");

		Calendar c=Calendar.getInstance();
		int h=c.get(Calendar.HOUR_OF_DAY);

		if(h<12)
			pw.println("<center><h1>Good Morning</h1></center>");
		else if(h<16)
			pw.println("<center><h1>Good Afternoon</h1></center>");
		else if(h<20)
			pw.println("<center><h1>Good Evening</h1></center>");
		else
			pw.println("<center><h1>Good Night</h1></center>");

		pw.close();
	}
}
```

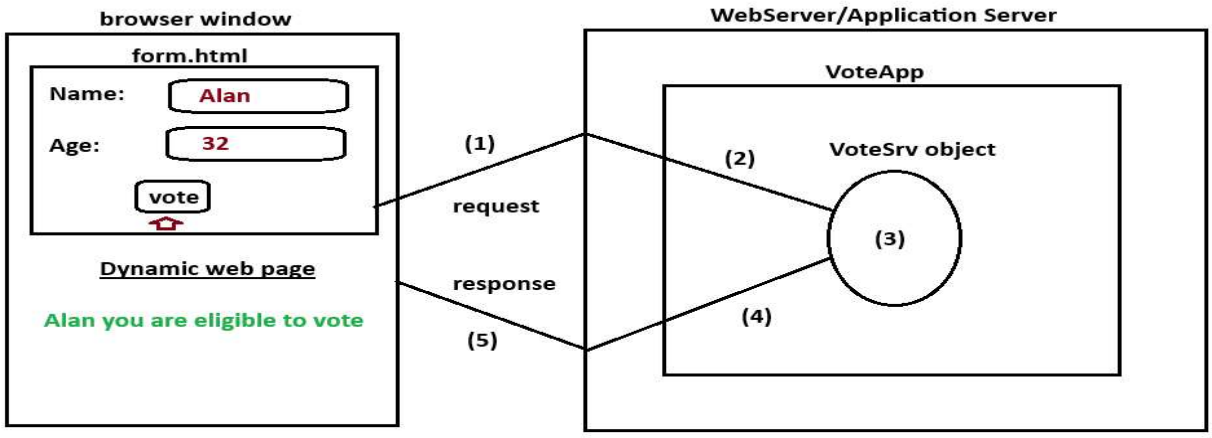
Request url

http://localhost:2525/WishApp/

class4

Example application on HTML based form page to servlet communication

Diagram: servlet4.1



Deployment Directory structure

VoteApp

```
|
|---Java Resources
        |
        |-----src
                |
                |---com.ihub.www
                        |
                        |---VoteSrv.java
|
|---WebContent
        |
        |---form.html
        |
        |---WEB-INF
                |
                |---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

We can communicate to servlet program in two methodologies.

1) GET methodology

It will carry limited amount of data.

2) POST methodology

It will carry unlimited amount of data.

While working HttpServlet class , it is never recommanded to use service(-,-) method because it is not designed according to HTTP protocol features.

It is always recommanded to use doXxx(-,-) methods because they designed according to HTTP protocol features.

We have seven doXxx(-,-) methods as given below.
ex:

doGet(-,-)
doPost(-,-)
doPut(-,-)
doDelete(-,-)
doOption(-,-)

doTrace(-,-)
doHead(-,-)

Prototype of doXxx(-,-) method

```
protected void doGet(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
{

}
```

form.html

```
<form action="test" method="GET">

        Name: <input type="text" name="t1"/> <br>

        Age: <input type="text" name="t2"/> <br>

        <input type="submit" value="vote"/>

</form>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

        <servlet>
                <servlet-name>VoteSrv</servlet-name>
                <servlet-class>com.ihub.www.VoteSrv</servlet-class>
        </servlet>
        <servlet-mapping>
                <servlet-name>VoteSrv</servlet-name>
                <url-pattern>/test</url-pattern>
        </servlet-mapping>

        <welcome-file-list>
                <welcome-file>form.html</welcome-file>
        </welcome-file-list>
```

```
</web-app>
```

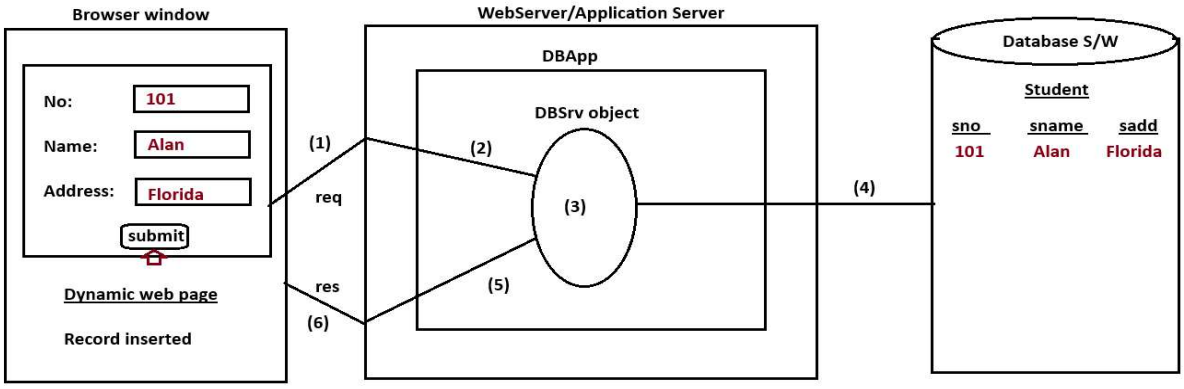
VoteSrv.java

```
package com.ihub.www;

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 VoteSrv extends HttpServlet
{
	protected void doGet(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
	{
		PrintWriter pw=res.getWriter();
		res.setContentType("text/html");

		//reading form data
		String name = req.getParameter("t1");
		String sage = req.getParameter("t2");

		//converting string age to int age
		int age = Integer.parseInt(sage);

		if(age<18)
			pw.println("<center><h1 style='color:red;'>"+name+" U r not eligible to
vote</h1></center>");
		else
			pw.println("<center><h1 style='color:green;'>"+name+" U r eligible to vote
</h1></center>");

		pw.close();
	}
}
```

Request url :- http://localhost:2525/VoteApp/

Servlet to Database Communication

Diagram: servlet4.2



Deployment Directory Structure

```
DBApp
|
|---Java Resources
        |
        |-------src
                |
                |---com.ihub.www
                        |
                        |---DBSrv.java
|
|---WebContent
        |
        |---form.html
        |
        |---WEB-INF
                |
                |-----web.xml
                |-----lib
                        |
                        |--ojdbc14.jar
```

Note:

In above application we need to add "servlet-api.jar" and "ojdbc14.jar" file in project build path.

Copy and paste "ojdbc14.jar" file inside "WEB-INF/lib" folder separetly.

student table

```
drop table student;

create table student(sno number(3),sname varchar2(10), sadd varchar2(12));
```

form.html

```
<form action="test" method="GET">
```

```
	<table align="center">

		<tr>
			<td>No:</td>
			<td><input type="text" name="t1"/></td>
		</tr>

		<tr>
			<td>Name:</td>
			<td><input type="text" name="t2"/></td>
		</tr>

		<tr>
			<td>Address:</td>
			<td><input type="text" name="t3"/></td>
		</tr>

		<tr>
			<td><input type="reset" value="reset"/></td>
			<td><input type="submit" value="submit"/></td>
		</tr>

	</table>

</form>
```

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <servlet>
	<servlet-name>DBSrv</servlet-name>
	<servlet-class>com.ihub.www.DBSrv</servlet-class>
  </servlet>

  <servlet-mapping>
	<servlet-name>DBSrv</servlet-name>
	<url-pattern>/test</url-pattern>
  </servlet-mapping>
```

```
  <welcome-file-list>
        <welcome-file>form.html</welcome-file>
  </welcome-file-list>

</web-app>
```

DBSrv.java

```
package com.ihub.www;

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 DBSrv extends HttpServlet
{
        protected void doGet(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
        {
                PrintWriter pw=res.getWriter();
                res.setContentType("text/html");

                //reading form data
                String sno=req.getParameter("t1");
                int no=Integer.parseInt(sno);
                String name = req.getParameter("t2");
                String add = req.getParameter("t3");

                //insert data into database
                Connection con=null;
                PreparedStatement ps=null;
                String qry=null;
                int result=0;
                try
                {
                        Class.forName("oracle.jdbc.driver.OracleDriver");
```

```
con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:XE","system","admin");
                        qry="insert into student values(?,?,?)";
                        ps=con.prepareStatement(qry);
                        //set the values
                        ps.setInt(1, no);
                        ps.setString(2, name);
                        ps.setString(3, add);
                        //execute
                        result=ps.executeUpdate();
                        if(result==0)
                                pw.println("<center><h1>No Record Inserted</h1></center>");
                        else
                                pw.println("<center><h1>Record Inserted</h1></center>");

                        ps.close();
                        con.close();
                }
                catch(Exception e)
                {
                        e.printStackTrace();
                }

                pw.close();
        }
}
```

Request url :- http://localhost:2525/DBApp/

CLASS 5

Servlet life cycle methods

We have three life cycle methods in servlet.

1) public void init(ServletConfig config)throws ServletException

> It is used for instantiation event.
> This method will execute just before servlet object creation.

2) public void service(ServletRequest req,ServletResponse res)throws

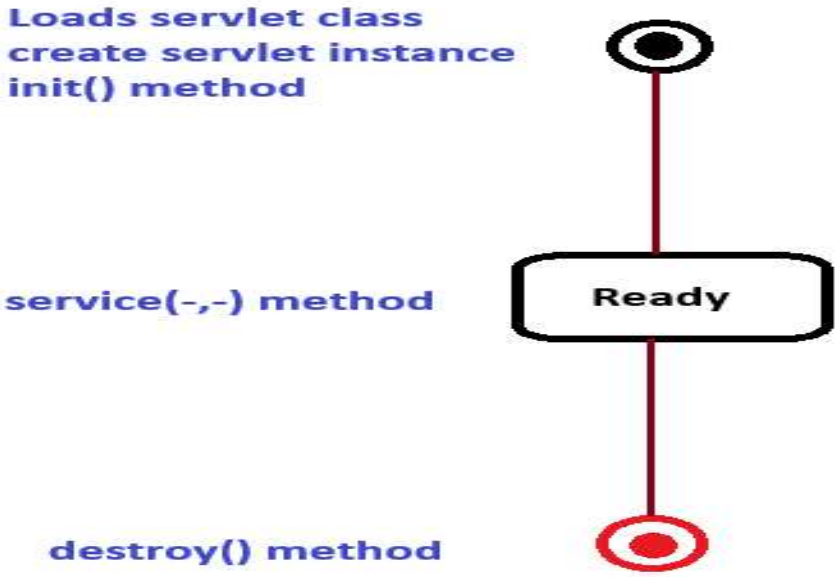
ServletException,IOException

> It is a request arrival event.
> This method will execute when request goes to servlet.

3) public void destroy()

> it is used for destruction event.
> This method will execute just before servlet object destruction.

Diagram: servlet5.1



Deployment Directory Structure

```
LifeCycleApp
|
|---Java Resources
|       |
        |------src
                |
                |---com.ihub.www
                        |
                        |---TestSrv.java
|---WebContent
|       |
        |------index.html
        |
        |------WEB-INF
                |
                |---web.xml
```

Note: In above application we need to add "servlet-api.jar" file in project build path.

index.html

```
<center>
	<h1>
		<a href="test"> Click Here </a>
	</h1>
</center>
```

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <servlet>
	<servlet-name>TestSrv</servlet-name>
	<servlet-class>com.ihub.www.TestSrv</servlet-class>
  </servlet>
  <servlet-mapping>
	<servlet-name>TestSrv</servlet-name>
	<url-pattern>/test</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
	<welcome-file>index.html</welcome-file>
  </welcome-file-list>

</web-app>
```

TestSrv.java

```
package com.ihub.www;

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

import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServlet;

public class TestSrv extends HttpServlet
```

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

	}
	public void service(ServletRequest req,ServletResponse res)throws
ServletException,IOException
	{
		PrintWriter pw=res.getWriter();
		res.setContentType("text/html");

		pw.println("<center><h1>This is service method</h1></center>");

		pw.close();
	}
	public void destroy()
	{

	}
}
```

Request url : http://localhost:2525/LifeCycleApp/

Form validation

The process of checking format and pattern of form data is called form validation and such logic is called form validation logic.

We can write form validation at client side as well as server side but performs server side form validation when client side form validation is not done.

Deployment Directory Structure

```
ValidationApp
|
|---Java Resources
|	|
	|------src
		|
		|---com.ihub.www
			|
			|---FormSrv.java
|---WebContent
|	|
	|------form.html
```

```
|------validation.js
|
|------WEB-INF
        |
        |---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

Form.html

```
<!DOCTYPE html>
<html>
    <head>
        <title>MyPage!</title>

        <!--  add external javascript  -->
        <script type="text/javascript"  src="validation.js"></script>

    </head>
    <body>

        <form name="myForm" action="test" method="GET" onsubmit="return
validate()">

            Name: <input type="text" name="t1"/> <br>
            Age : <input type="text" name="t2"/> <br>

            <!--  hiddex box field -->
            <input type="hidden" value="no" name="vflag"/>

            <input type="submit" value="submit"/>

        </form>

    </body>
</html>
```

validation.js

```
function validate()
{
    var name=document.myForm.t1.value;
    var age=document.myForm.t2.value;
    document.myForm.vflag.value="yes";
```

```
	if(name=="")
	{
		alert("Name is mandatory");
		document.myForm.t1.focus();
		return false;
	}
	if(age=="")
	{
		alert("Age is mandatory");
		document.myForm.t2.focus();
		return false;
	}
	else
	{
		if(isNaN(age))
		{
			alert("Age must be numeric");
			document.myForm.t2.value="";
			document.myForm.t2.focus();
			return false;
		}
	}



	return true;
}
```

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <servlet>
	<servlet-name>FormSrv</servlet-name>
	<servlet-class>com.ihub.www.FormSrv</servlet-class>
  </servlet>

  <servlet-mapping>
```

```
        <servlet-name>FormSrv</servlet-name>
        <url-pattern>/test</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
        <welcome-file>form.html</welcome-file>
  </welcome-file-list>

</web-app>
```

FormSrv.java

```
package com.ihub.www;

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 FormSrv extends HttpServlet
{
        protected void doGet(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
        {
                PrintWriter pw=res.getWriter();
                res.setContentType("text/html");

                //reading form data
                String name=req.getParameter("t1");
                String sage=req.getParameter("t2");
                String status=req.getParameter("vflag");
                int age=0;

                if(status.equals("no"))
                {
                        if(name=="" || name==null|| name.length()==0)
                        {
                                pw.println("<center>Name is mandatory</center>");
                                return;
                        }
                        if(sage=="" || sage==null|| sage.length()==0)
```

```
                {
                        pw.println("<center>Age is mandatory</center>");
                        return;
                }
                else
                {
                        try
                        {
                                age=Integer.parseInt(sage);
                        }
                        catch(NumberFormatException nfe)
                        {
                                pw.println("<center>Age must be numeric</center>");
                                return;
                        }
                }
        }

        if(status.equals("yes"))
        {
                age=Integer.parseInt(sage);
        }

        if(age<18)
                pw.println("<center><h1>U r not eligible to vote</h1></center>");
        else
                pw.println("<center><h1>U r eligible to vote</h1></center>");

        pw.close();
    }
}
```

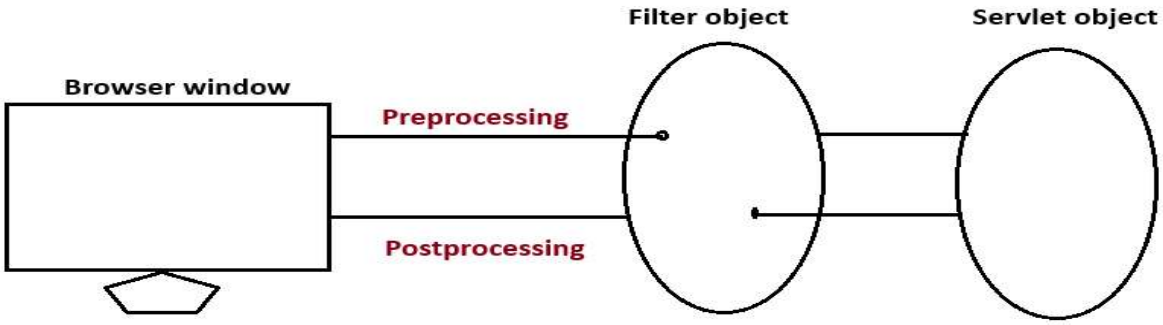
Request url :- http://localhost:2525/ValidationApp/

Servlet Filters

==========

Filter is an object which is executed at the time of preprocessing and postprocessing of the request.

Diagram: servlet5.2



The main objective of Filter is to perform filtering task.

1) To count number of request coming to the application

2) To perform form validation

3) To perform encryption and decryption

and etc.

Like Servlet, Filter also having it own API called Filter API.

A javax.servlet package having three interfaces of Filter API.

1) Filter

2) FilterChain

3) FilterConfig

1)Filter Interface

==========

For creating any filter, we must and should implements the Filter interface.

Filter interface provides the following 3 life cycle methods for filter.

i)public void init(FilterConfig config)

IT is used to initialize the filter.
It invokes only once .

ii)public void doFilter(HttpServletRequest req,HttpServletResponse res,FilterChain chain)

This method is invoked every time when user request to any resources to which the filter is mappend.

IT is used to perform filtering task.

iii)public void destroy()

This method is invoked only once when filter is taken out of the service.

2) FilterChain
==========
It is responsible to invoke the next filter or resource in the chain.

FilterChain contains only one method.

i) public void doFilter(HttpServletRequest req,HttpServletResponse res)

It passes the control to the next filter or resource.

3)FilterConfig
==========
For every filter our servlet container creates FilterConfig object.
It is one per filter.

Deployment Directory Structure

```
FilterApp
|
|---Java Resources
|       |
        |------src
                |
                |---com.ihub.www
                        |
                        |---MyFilter.java
                        |---MyServlet.java
|---WebContent
|       |
        |------index.html
        |------WEB-INF
                |
                |---web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

Index.html

```
<center>
	<h1>
		<a href="test"> Click Here </a>
	</h1>
</center>
```

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <welcome-file-list>
	<welcome-file>index.html</welcome-file>
  </welcome-file-list>

  <servlet>
	<servlet-name>MyServlet</servlet-name>
	<servlet-class>com.ihub.www.MyServlet</servlet-class>
  </servlet>

  <servlet-mapping>
	<servlet-name>MyServlet</servlet-name>
	<url-pattern>/test</url-pattern>
  </servlet-mapping>

  <filter>
	<filter-name>MyFilter</filter-name>
	<filter-class>com.ihub.www.MyFilter</filter-class>
  </filter>
  <filter-mapping>
	<filter-name>MyFilter</filter-name>
	<url-pattern>/test</url-pattern>
  </filter-mapping>

</web-app>
```

MyFilter.java

```
package com.ihub.www;

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

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 config)throws ServletException
	{

	}

	@Override
	public void doFilter(ServletRequest req, ServletResponse res, FilterChain chain)
			throws IOException, ServletException {

		PrintWriter pw=res.getWriter();
		res.setContentType("text/html");

		pw.println("<center>Filter Invoke Before</center><br>");
		chain.doFilter(req, res);
		pw.println("<center>Filter Invoke After</center><br>");

	}

	public void destroy()
	{

	}

}
```

MyServlet.java

```
package com.ihub.www;
```

```
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 MyServlet extends HttpServlet
{
	protected void doGet(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
	{
		PrintWriter pw=res.getWriter();
		res.setContentType("text/html");

		pw.println("<center><h1>Servlet invoked</h1></center>");
	}
}
```

Request url :- http://localhost:2525/FilterApp

CLASS 6

Servlet to Servlet Communication
==========
Servlet to servlet communication is also known as servlet chaining.

Servlet to servlet communication is possible in three ways.

1) Forwarding the request

2) Including the response

3) Send Redirection

1) Forwarding the request

In forwarding the request , output of source program will be discarded and output of destination program goes to browser window as dynamic response.
To forward the request we need to use RequestDispatcher object.

RequestDispatcher is an interface which is present in javax.servlet package.

We can create RequestDispatcher object by using getRequestDispatcher() method of HttpServletRequest.
ex:

```
RequestDispatcher rd=req.getRequestDispatcher();
rd.forward(req,res);
```

2) Including the response

In including the response , the out of source servlet program and destination servlet program combinely goes to browser window as dynamic response.

To include the response we need to use RequestDispatcher object.

RequestDispatcher is an interface which is present in javax.servlet package.

We can create RequestDispatcher object by using getRequestDispatcher() method of HttpServletRequest.
ex:

```
RequestDispatcher rd=req.getRequestDispatcher();
rd.include(req,res);
```

Deployment Directory Structure

```
STSApp1
|
|---Java Resources
|       |
        |-------src
                |
                |---com.ihub.www
                        |
                        |---TestSrv1.java
                        |---TestSrv2.java
|---WebContent
        |
        |----form.html
        |
        |----WEB-INF
                |
                |---web.xml
```

Note:
In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="test1" method="GET">

	<table align="center">

			<tr>
				<td>UserName:</td>
				<td><input type="text" name="t1" required/></td>
			</tr>

			<tr>
				<td>Password:</td>
				<td><input type="password" name="t2" required/></td>
			</tr>

			<tr>
				<td><input type="reset" value="reset"/></td>
				<td><input type="submit" value="submit"/></td>
			</tr>

	</table>

</form>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <servlet>
	<servlet-name>TestSrv1</servlet-name>
	<servlet-class>com.ihub.www.TestSrv1</servlet-class>
  </servlet>
  <servlet-mapping>
	<servlet-name>TestSrv1</servlet-name>
	<url-pattern>/test1</url-pattern>
  </servlet-mapping>

  <servlet>
```

```
        <servlet-name>TestSrv2</servlet-name>
        <servlet-class>com.ihub.www.TestSrv2</servlet-class>
  </servlet>
  <servlet-mapping>
        <servlet-name>TestSrv2</servlet-name>
        <url-pattern>/test2</url-pattern>
  </servlet-mapping>

  <welcome-file-list>
        <welcome-file>form.html</welcome-file>
  </welcome-file-list>

</web-app>
```

TestSrv1.java

```
package com.ihub.www;

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 TestSrv1 extends HttpServlet
{
        protected void doGet(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
        {
                PrintWriter pw=res.getWriter();
                res.setContentType("text/html");

                //reading form data
                String name = req.getParameter("t1");
                String pass = req.getParameter("t2");

                if(pass.equals("admin"))
                {
                        RequestDispatcher rd=req.getRequestDispatcher("test2");
                        rd.forward(req, res);
                }
```

```
		else
		{
			pw.println("<center><b style='color:red'>Sorry! Incorrect username or
password</b></center>");
			RequestDispatcher rd=req.getRequestDispatcher("form.html");
			rd.include(req, res);
		}

		pw.close();
	}
}
```

TestSrv2.java

```
package com.ihub.www;

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 TestSrv2 extends HttpServlet
{
	protected void doGet(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
	{
		PrintWriter pw=res.getWriter();
		res.setContentType("text/html");

		pw.println("<center><h1>Login Done Successfully!</h1></center>");

		pw.close();
	}
}
```

Request url :- http://localhost:2525/STSApp1/

3) Send Redirection

==========

It is used to forward the request to the application which is present in same server or different server.

To perform send redirection we need to use sendRedirection() method of HttpServletResponse object.
ex:

```
res.sendRedirect("url");
```

It always sends a new request.

It uses browser window to send a request.

It works inside as well as side of the server.

Deployment Directory Structure

```
STSApp2
|
|---Java Resources
|       |
        |-------src
                |
                |---com.ihub.www
                        |
                        |---TestSrv.java

|---WebContent
|       |
        |---index.html
        |
        |---WEB-INF
                |
                |---web.xml
```

Note: In above application we need to add "servlet-api.jar" file in project build path.

index.html

```
<center>

        <h1>
                <a href="test?t1=flights"> Flights </a>
        </h1>
```

```
	<h1>
		<a href="test?t1=hotels"> Hotels </a>
	</h1>

	<h1>
		<a href="test?t1=railways"> Train </a>
	</h1>

</center>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

	<welcome-file-list>
		<welcome-file>index.html</welcome-file>
	</welcome-file-list>

	<servlet>
		<servlet-name>TestSrv</servlet-name>
		<servlet-class>com.ihub.www.TestSrv</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>TestSrv</servlet-name>
		<url-pattern>/test</url-pattern>
	</servlet-mapping>

</web-app>
```

TestSrv.java

```
package com.ihub.www;

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 TestSrv extends HttpServlet
{
	protected void doGet(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
	{
		PrintWriter pw=res.getWriter();
		res.setContentType("text/html");

		//reading the data
		String value=req.getParameter("t1");

		res.sendRedirect("http://www.makemytrip.com/"+value);

		pw.close();

	}
}
```

Request url :- http://localhost:2525/STSApp2/

ServletConfig
==========
ServletConfig is an interface which is present in javax.servlet package.

ServletConfig object created by the web container one per servlet.

ServletConfig object is used to read configuration information from web.xml file.

We can create ServletConfig object as follow.

Ex: ServletConfig config=getServletConfig();

ServletConfig interface contains following methods.

1)public String getInitParameter(String name);

It will return parameter value based on specified parameter name.

2)public Enumeration getInitParameterNames();

It will return enumeration of all initialized parameter names.

3)public ServletContext getServletContext();

It will return ServletContext object.

4)public String getServletName();

It will return Servlet name.

Deployment Directory Structure

```
ConfigApp
|
|---Java Resources
|       |
        |-------src
                |
                |---com.ihub.www
                        |
                        |---TestSrv.java

|---WebContent
|       |
        |---index.html
        |
        |---WEB-INF
                |
                |---web.xml
```

Note:
In above application we need to add "servlet-api.jar" file in project build path.

index.html

```
<center>
        <h1>
                <a href="test"> Click Here </a>
        </h1>
</center>
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">
```

```
  <welcome-file-list>
        <welcome-file>index.html</welcome-file>
  </welcome-file-list>

    <servlet>
        <servlet-name>TestSrv</servlet-name>
        <servlet-class>com.ihub.www.TestSrv</servlet-class>
        <init-param>
                <param-name>driver</param-name>
                <param-value>oracle.jdbc.driver.OracleDriver</param-value>
        </init-param>
        <init-param>
                <param-name>url</param-name>
                <param-value>jdbc:oracle:thin:@localhost:1521:XE</param-value>
        </init-param>
  </servlet>

  <servlet-mapping>
        <servlet-name>TestSrv</servlet-name>
        <url-pattern>/test</url-pattern>
  </servlet-mapping>

</web-app>
```

TestSrv.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Enumeration;

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

public class TestSrv extends HttpServlet
{
        protected void doGet(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
        {
                PrintWriter pw=res.getWriter();
```

```
                res.setContentType("text/html");

                ServletConfig config=getServletConfig();

                pw.println(config.getInitParameter("driver")+"<br>");
                pw.println(config.getInitParameter("url")+"<br>");

                Enumeration<String> e=config.getInitParameterNames();
                while(e.hasMoreElements())
                {
                        String s = e.nextElement();
                        pw.println(s+"<br>");
                }

                pw.println(config.getServletName());

                pw.close();
        }
}
```

Request url :- http://localhost:2525/ConfigApp/

ServletContext object

==========

ServletContext is an interface which is present in javax.servlet package.

ServletContext object created by web container for every web application.

ServletContext object is used to read configuration information from web.xml file which is global.

We can create ServletContext object as follow.

ex:

```
ServletContext context = getServletContext();

or

ServletConfig config=getServletConfig();
ServletContext context=config.getServletContext();
```

ServletContext object contains following methods.

1)public String getInitParameter(String name);

It will return parameter value based on specified parameter name.

2)public Enumeration getInitParameterNames();

It will return enumeration of all initialized parameter names.

3)public void setAttribute(String name,Object obj);

It will set the attribute.

4)public Object getAttribute(String name);

It will return the attribute value.

5)public void removeAttribute(String name);

It will remove the attribute.

Deployment Directory Structure

```
ContextApp
|
|---Java Resources
        |
        |------src
                |
                |----com.ihub.www
                        |
                        |----TestSrv.java
|---WebContent
        |
        |-----index.html
        |-----WEB-INF
                |
                |---web.xml
```

Note : - In above application we need to add "servlet-api.jar" file in project build path.

Index.html

```
<center>
        <h1>
                <a href="test"> click Here </a>
        </h1>
</center>
```

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
```

```
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

  <servlet>
        <servlet-name>TestSrv</servlet-name>
        <servlet-class>com.ihub.www.TestSrv</servlet-class>
  </servlet>

  <servlet-mapping>
        <servlet-name>TestSrv</servlet-name>
        <url-pattern>/test</url-pattern>
  </servlet-mapping>

  <context-param>
                <param-name>driver</param-name>
                <param-value>oracle.jdbc.driver.OracleDriver</param-value>
        </context-param>
        <context-param>
                <param-name>url</param-name>
                <param-value>jdbc:oracle:thin:@localhost:1521:XE</param-value>
        </context-param>

  <welcome-file-list>
        <welcome-file>index.html</welcome-file>
  </welcome-file-list>
</web-app>
```

TestSrv.java

```
package com.ihub.www;

import java.io.IOException;
import java.io.PrintWriter;
import java.util.Enumeration;

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

public class TestSrv extends HttpServlet
{
```

```
	protected void doGet(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
	{
		PrintWriter pw=res.getWriter();
		res.setContentType("text/html");

		ServletContext context=getServletContext();

		pw.println(context.getInitParameter("driver")+"<br>");
		pw.println(context.getInitParameter("url")+"<br>");

		Enumeration<String> e=context.getInitParameterNames();
		while(e.hasMoreElements())
		{
			String s=(String)e.nextElement();
			pw.println(s+"<br>");
		}

		context.setAttribute("name","Alan");

		pw.println(context.getAttribute("name"));

		context.removeAttribute("name");

		pw.close();

	}
}
```

Request url :- http://localhost:2525/ConfigApp/

CLASS 7

File Uploading

The process of capturing a file from client machine file system and storing in a server machine file system is called file uploading and reverse is called file downloading.

While dealing with matrimonial applications, job portal applications, profile management applications we need to upload or download the files.

There is no specific API in servlet to perform file uploading.

Here we need to take the support of third party API called JAVAZOOM API.

JAVAZOOM API comes in zip file and once if we extracted we will get three jar files.

ex:

uploadbean.jar (main jar file)
struts.jar (dependent jar file)
cos.jar (dependent jar file)

We can take file component in a form page as follow.
ex:

```
<input type="file" name="f1"/>
```

JAVAZOOM API

Download link :
https://drive.google.com/file/d/1LB0WSJvSCCVOgz7xNwyuYtmy_0_TfJzq/view?usp=drive_link

Deployment Directory Structure

```
UploadApp
|
|---Java Resources
|       |
        |------src
                |
                |---com.ihub.www
                        |
                        |----UploadSrv.java

|---WebContent
        |
        |----form.html
        |
        |----WEB-INF
                |
                |------web.xml
                |------lib
                        |
                        |---uploadbean.jar
                        |---struts.jar
                        |---cos.jar
```

Note:
In above application we need to add "servlet-api.jar" file and "uploadbean.jar" file in project build path.

Copy and paste javazoom jar files inside "WEB-INF/lib" folder seperately.

Form.html

```
<form action="test" method="POST" enctype="multipart/form-data">

	File1 : <input type="file" name="f1"/> <br>

	File2 : <input type="file" name="f2"/> <br>

	<input type="submit" value="upload"/>

</form>
```

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

	<servlet>
		<servlet-name>UploadSrv</servlet-name>
		<servlet-class>com.ihub.www.UploadSrv</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>UploadSrv</servlet-name>
		<url-pattern>/test</url-pattern>
	</servlet-mapping>

	<welcome-file-list>
		<welcome-file>form.html</welcome-file>
	</welcome-file-list>

</web-app>
```

UploadSrv.java

```
package com.ihub.www;

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;

import javazoom.upload.MultipartFormDataRequest;
import javazoom.upload.UploadBean;

public class UploadSrv extends HttpServlet
{
	protected void doPost(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
	{
		PrintWriter pw=res.getWriter();
		res.setContentType("text/html");

		//file uploading logic
		try
		{
			UploadBean ub=new UploadBean();
			ub.setFolderstore("D:\\arvind");
			ub.setOverwrite(false);

			MultipartFormDataRequest nreq=new MultipartFormDataRequest(req);
			ub.store(nreq);

			pw.println("<center><h1> Files are uploaded
successfully</h1></center>");
		}
		catch(Exception e)
		{
			pw.println(e);
		}

		pw.close();
	}
}
```

Request url : - http://localhost:2525/UploadApp/

Q) What is the difference between GET and POST methodology?

GET	**POST**
----------	**----------**
It is a default methodology.	It is not a default methodology.
It sends the request fastly.	It is bit slow.
It carries limited amount of data.	It carries unlimited amount of data.
It is not suitable for secure the data.	It is suitable for secure.
It is not suitable for encryption and file uploading.	It is suitable for encryption and file uploading.
To process GET methodology we will use doGet(-,-) method.	To process POST methodology we will use doPost(-,-) method.

Q) How to enable <load-on-startup> and what happens if we enable <load-on-startup>?

We can enable <load-on-startup> inside web.xml file.

web.xml

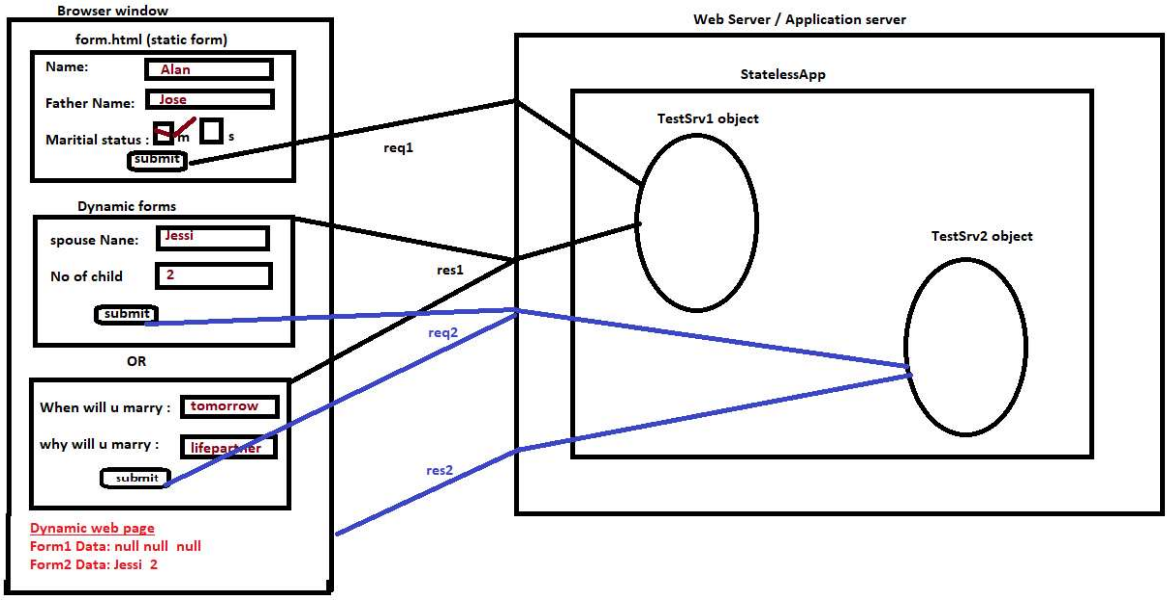
```
<web-app>
	<servlet>
		<servlet-name>TestSrv</servlet-name>
		<servlet-class>com.ihub.www.TestSrv</servlet-class>
		<load-on-startup>1</load-on-startup>
	</servlet>
	<servlet-mapping>
		<servlet-name>TestSrv</servlet-name>
		<url-pattern>/test</url-pattern>
	</servlet-mapping>
</web-app>
```

If we enable <load-on-startup> then our servlet container will create servlet project during the server startup or during the deployment of web application.

In general, our servlet container creates servlet object before we give the request.

Stateless Behaviour of Web Application

Diagram: servlet7.1



In above diagram demostrate stateless behaviour of web application.

In stateless web application, no web resource program can access previous request data while processing the current request.

To overcome this limitation we need to use session tracking.

Deployment Directory Structure

```
StatelessApp
|
|---Java Resources
|       |
        |------src
                |
                |---com.ihub.www
                        |
                        |----TestSrv1.java
                        |----TestSrv2.java
|---WebContent
        |
        |----form.html
        |
        |----WEB-INF
```

```
                |
                |------web.xml
```

Note:

In above application we need to add "servlet-api.jar" file in project build path.

form.html

```
<form action="test1" method="POST">

        Name: <input type="text" name="t1"/> <br>

        Father Name: <input type="text" name="t2"/> <br>

        Maritial Status :
        <input type="checkbox" name="t3" value="married"/> MARRIED
        <input type="checkbox" name="t3" value="single"/> SINGLE

        <br>

        <input type="submit" value="submit"/>

</form>
```

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

        <servlet>
                <servlet-name>TestSrv1</servlet-name>
                <servlet-class>com.ihub.www.TestSrv1</servlet-class>
                <load-on-startup>1</load-on-startup>
        </servlet>
        <servlet-mapping>
                <servlet-name>TestSrv1</servlet-name>
                <url-pattern>/test1</url-pattern>
        </servlet-mapping>

        <servlet>
                <servlet-name>TestSrv2</servlet-name>
```

```
		<servlet-class>com.ihub.www.TestSrv2</servlet-class>
		<load-on-startup>2</load-on-startup>
	</servlet>
	<servlet-mapping>
		<servlet-name>TestSrv2</servlet-name>
		<url-pattern>/test2</url-pattern>
	</servlet-mapping>

	<welcome-file-list>
		<welcome-file>form.html</welcome-file>
	</welcome-file-list>

</web-app>
```

TestSrv1.java

```
package com.ihub.www;

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 TestSrv1 extends HttpServlet
{
	protected void doPost(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
	{
		PrintWriter pw=res.getWriter();
		res.setContentType("text/html");

		//reading form data
		String name = req.getParameter("t1");
		String fname = req.getParameter("t2");
		String ms = req.getParameter("t3");

		if(ms.equals("married"))
		{
			pw.println("<form action='test2' method='POST'>");
			pw.println("Spouse Name : <input type='text' name='f2t1'/> <br>");
			pw.println("No of Child : <input type='text' name='f2t2'/> <br>");
```

```
                pw.println("<input type='submit' value='submit'/>");
                pw.println("</form>");
            }
            else
            {
                pw.println("<form action='test2' method='POST'>");
                pw.println("When will u marry : <input type='text' name='f2t1'/> <br>");
                pw.println("Why will u marry : <input type='text' name='f2t2'/> <br>");
                pw.println("<input type='submit' value='submit'/>");
                pw.println("</form>");
            }

            pw.close();
        }
}
```

TestSrv2.java

```
package com.ihub.www;

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 TestSrv2 extends HttpServlet
{
    protected void doPost(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
    {
        PrintWriter pw=res.getWriter();
        res.setContentType("text/html");

        //reading form1  data
        String name = req.getParameter("t1");
        String fname = req.getParameter("t2");
        String ms = req.getParameter("t3");

        //reading form2 data
        String val1=req.getParameter("f2t1");
        String val2=req.getParameter("f2t2");
```

```
                pw.println("Form 1 Data :"+name+" "+fname+" "+ms+"<br>");
                pw.println("Form 2 Data :"+val1+" "+val2+"<br>");

                pw.close();
        }
}
```

Request url:- http://localhost:2525/StatelessApp/

Session

The process of continue and related operations performed on web application with multiple request and response is called session.
ex:

login to gmail and logout from gmail is one session.
starting of java class and ending of java class is one session.

Our HTTP protocol stateless, which makes our web application also stateless.

In stateless web application no web resource program can access previous request data while processing the current request during a session.

To overcome this limitation we need to use session tracking.

Session Tracking or Session Management

Session tracking is used to make our web application as statefull web application even thought our HTTP protocol is stateless.

In stateless web application no web resource program can access previous request data while processing the current request during a session.

In statefull web application all we resource programs can access previous request data while processing the current request during a session.

There are four techniques to perform session tracking.
1) Using hidden box fields

2) HttpCookies

3) HttpSession with Cookies

4) URL Rewriting

HttpSession with Cookies

In HttpSession with cookies, for every request it will create a unique session id.

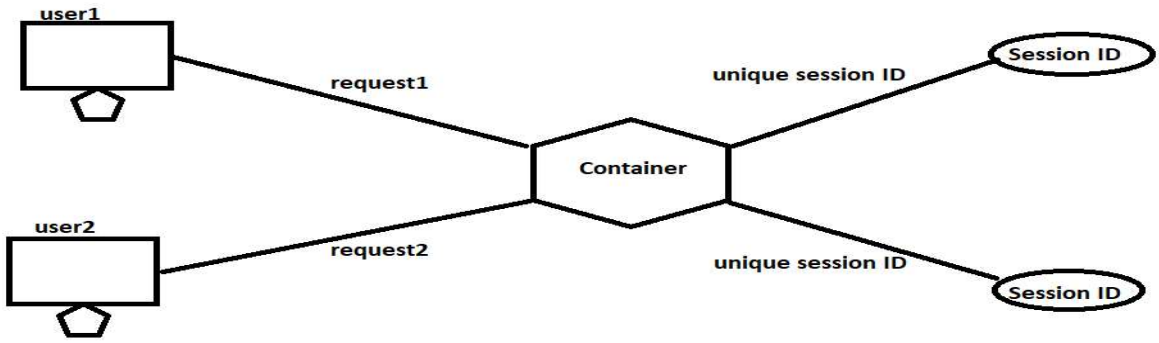
Our web container uses that session id to identify a user is a new user or existing user.

HttpSession object we can use to perform following task.

1)Bind the objects

2)To manipuate the data in HttpSession object.

Diagram:servlet7.2



form.html

```
<form action="test1" method="POST">

        Name: <input type="text" name="t1"/> <br>

        Father Name: <input type="text" name="t2"/> <br>

        Maritial Status :
        <input type="checkbox" name="t3" value="married"/> MARRIED
        <input type="checkbox" name="t3" value="single"/> SINGLE

        <br>

        <input type="submit" value="submit"/>

</form>
```

Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd" id="WebApp_ID" version="3.0">

	<servlet>
		<servlet-name>TestSrv1</servlet-name>
		<servlet-class>com.ihub.www.TestSrv1</servlet-class>
		<load-on-startup>1</load-on-startup>
	</servlet>
	<servlet-mapping>
		<servlet-name>TestSrv1</servlet-name>
		<url-pattern>/test1</url-pattern>
	</servlet-mapping>

	<servlet>
		<servlet-name>TestSrv2</servlet-name>
		<servlet-class>com.ihub.www.TestSrv2</servlet-class>
		<load-on-startup>2</load-on-startup>
	</servlet>
	<servlet-mapping>
		<servlet-name>TestSrv2</servlet-name>
		<url-pattern>/test2</url-pattern>
	</servlet-mapping>

	<welcome-file-list>
		<welcome-file>form.html</welcome-file>
	</welcome-file-list>

</web-app>
```

TestSrv1.java

```
package com.ihub.www;

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;
```

```
import javax.servlet.http.HttpSession;

public class TestSrv1 extends HttpServlet
{
	protected void doPost(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
	{
		PrintWriter pw=res.getWriter();
		res.setContentType("text/html");

		//reading form data
		String name = req.getParameter("t1");
		String fname = req.getParameter("t2");
		String ms = req.getParameter("t3");

		//store the data in HttpSession
		HttpSession session=req.getSession(true);
		session.setAttribute("pname", name);
		session.setAttribute("pfname", fname);
		session.setAttribute("pms", ms);

		if(ms.equals("married"))
		{
			pw.println("<form action='test2' method='POST'>");
			pw.println("Spouse Name : <input type='text' name='f2t1'/> <br>");
			pw.println("No of Child : <input type='text' name='f2t2'/> <br>");
			pw.println("<input type='submit' value='submit'/>");
			pw.println("</form>");
		}
		else
		{
			pw.println("<form action='test2' method='POST'>");
			pw.println("When will u marry : <input type='text' name='f2t1'/> <br>");
			pw.println("Why will u marry : <input type='text' name='f2t2'/> <br>");
			pw.println("<input type='submit' value='submit'/>");
			pw.println("</form>");
		}

		pw.close();
	}
}
```

TestSrv2.java

```
package com.ihub.www;

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;
import javax.servlet.http.HttpSession;

public class TestSrv2 extends HttpServlet
{
	protected void doPost(HttpServletRequest req,HttpServletResponse res)throws
ServletException,IOException
	{
		PrintWriter pw=res.getWriter();
		res.setContentType("text/html");

		//reading form1  data
		HttpSession session=req.getSession(false);
		String name = (String)session.getAttribute("pname");
		String fname = (String)session.getAttribute("pfname");
		String ms = (String)session.getAttribute("pms");

		//reading form2 data
		String val1=req.getParameter("f2t1");
		String val2=req.getParameter("f2t2");

		pw.println("Form 1 Data :"+name+" "+fname+" "+ms+"<br>");
		pw.println("Form 2 Data :"+val1+" "+val2+"<br>");

		pw.close();
	}
}
```

Request url :- http://localhost:2525/StatelessApp/