SERVELETS

Web Application

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 page / Passive web page

A web page which shows the same content for every request.

ex:

home page

services page

portfolio page

aboutus page

contactus page

and etc.

2)Dynamic web page / Active web page

A web page which shows different content for every request.

ex:

gmail inbox page

stock market share value page

live cricket score page

and etc.

We have two types of web resource programs.

1)Static web resource program

A web resource program which is used to create static web pages is called static web resource program.

ex:

HTML program

CSS program

Bootstrap program

angular program

and etc.

2)Dynamic web resource program

A web resource program which is used to create dynamic web pages is called dynamic web resource program.

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 (browser window) is called client side web resource program.

ex:

HTML program

CSS program

Boostrap program

angular program

and etc.

Static web resource programs are also known as client side web resource programs.

2)Server side web resource program

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

ex:

servlet program

jsp program

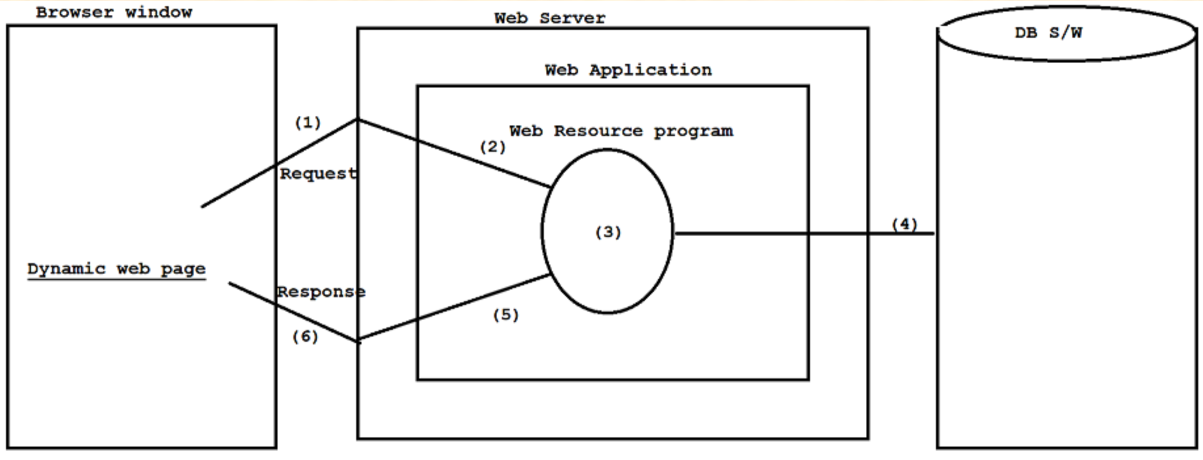
and etc.

Dynamic web reosurce programs are also known a server side web resource programs.

Understanding the setup Web application and Web Resource program execution

- Java program will execute manually.
- Web application and web resource program will execute when we have requested.
- There is no possibility to execute web application and web resource program manually.

Diagram: servlet1.1



With respect to the diagram

- Enduser will give the request web resource program.
- Server will trap that request and passes that request to appropriate web resource program.
- A web resource program will execute the logic to process the request.
- A web resource program will communicate with database if necessary.
- A web resource program will give the result to web server.
- Web server will send the output to browser window as dynamic response.

Web Server

A web server is a piece of software which is used to automate whole process of web application and web resource program execution.

ex:

Tomcat, Rasin and etc.

Responsibility of a web server

- It takes continues request from client.
- It traps the request and passes the request to appropriate web resource program.
- It provides environment to deploy or undeploy the web applications.
- It will add middleware services only to deployed web application.
- It provides environment to execute client side web resource program at browser window.
- It sends the output to the client as dynamic response.
- It is used to automate whole process of web application and web resource program execution.

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 deadth.
- Every server is designed to support web containers.
- Servlet container manage whole life cycle of servlet program.
- JSP container manage whole life cycle of jsp program.
- Some part of industry considers servlet container and jsp container are web containers.
- Tomcat is not a container , it contains servlet container and jsp container so we don't need to arrange seperately.
- Before 6.x version, Tomcat is known as web server.
- From 6.x version onwards , Tomcat is also known as application server.

Tomcat install will ask you following things.

- Http Connector port
- Adminstrator user and password
- JRE location(parallel to JDK)
- Tomcat Installation location

Tomcat

Version	:	7.x
Vendor	:	Apache Foundataion
website	:	https://tomcat.apache.org
Creator	:	James Duncan Davidson
Default Port	:	8080
Servlet container	:	Cataline
Jsp container	:	Jasper
Download link	:	

https://drive.google.com/file/d/0B9rC21sL6v0tZFdVcmxZUDA0Tms/view?usp=drive_link&resourcekey=0-VXlB_IpeWqDWwdbr1baCyA

Installation of Tomcat Server

Double click to tomcat software --> yes --> next --> I Agree --> select (full) -->

next --> HTTP connector port : 2525

adminstrator username : admin

password : admin --> next --> next --> Install.

Converting Tomcat server startup type from automatic mode to manual mode

Goto services(view local services) --> click to Apache tomcat 7.x --> click to stop link --> double click to apache tomcat --> startup type : manual --> apply -->ok.

Servlet API's

API is a collection of packages.

ex:

javax.servlet package

javax.servlet.http package

Important terminologies

javax.servlet.Servlet(I)

|

|

javax.servlet.GenericServlet(AC)

|

|

javax.servlet.http.HttpServlet(C)

What is Servlet?

Servlet is a server side web resource program which is used to enhance the funcationality of web server or application server.

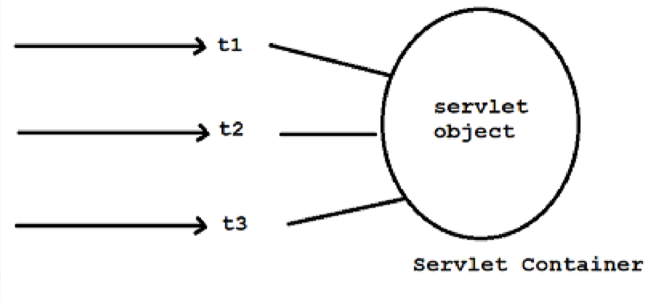
or

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

or

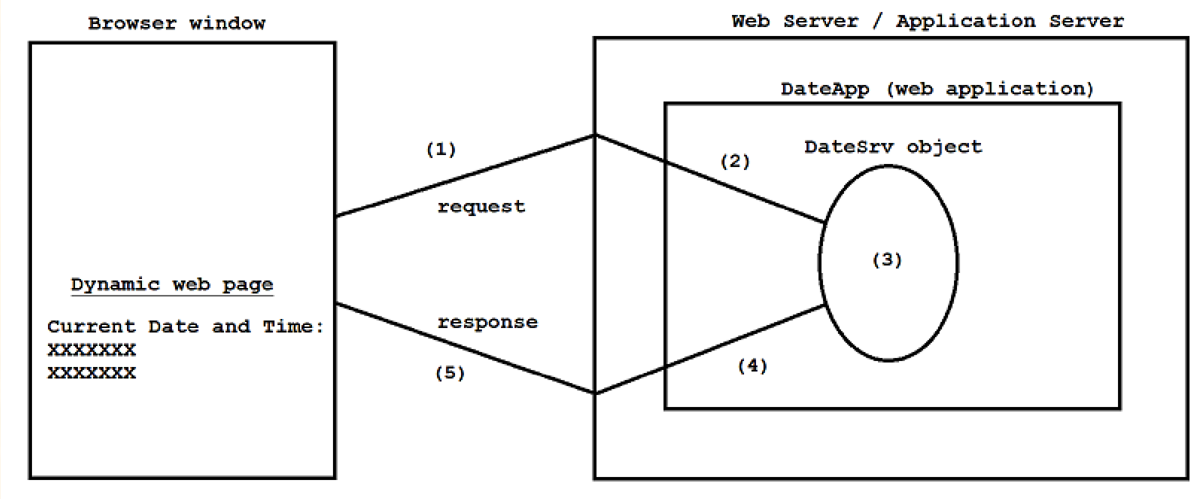
Servlet is a single instance multi-thread java based web resource program having the capability to develop web applications.

Diagram: servlet1.2



First web application development having servlet program as web resource program

Diagram: servlet2.1



Deployment Directory structure

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

dynamic web module version : 3.0 --> next --> next -->

select (generate web.xml) --> finish.

step3:

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

ex:

right click to DateApp --> build path --> configuration of build path

--> libraries --> Add external jars --> select servlet-api.jar file

---> open --> ok .

step4:

Create "com.ihub.www" package inside "java resources/src" folder.

ex:

right click to src folder --> new --> package -->

name : com.ihub.www --> finish.

step5:

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

ex:

right click to com.ihub.www --> 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();

	}
}
```

step6:

Configure each servlet 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>
```

step7:

Add Tomcat7.x server to eclipse IDE.

ex:

window --> preferences --> servers --> runtime environments -->

click to add button --> select Apache 7.0 --> next ---->

select Tomcat home folder (click to browse) --> finish --> ok.

Note:

Tomcat server we need to add only for one time in eclipse IDE.

step8:

Run the dynamic web project.

ex:

right click to DateApp ---> run as --> run on server -->

select Apache 7.x --> next --> finish.

step9:

Test the application by using below request url.

ex:

```
http://localhost:2525/DateApp/test
          |       |     |       |
hostname     portno  web app   url pattern
```

Note:

If we do any mistake in web.xml file then we will get 404 error.

If we do any mistake in servlet file then we will get 500 error.

Types of URL patterns

- Every servlet will recognize with the help of url pattern.
- Our container , server, client and other web resource programs will recognize each servlet by using url patterns.
- URL pattern will hide technology name or class name from outsider for security reason.

We have three types of url patterns.

1) Exact match url pattern

2) Directory match url pattern

3) Extension match url pattern

Every web server is designed to support these url variables.

1) Exact match url pattern

It always starts with forward slash '/' and it will not ends with '*' symbol.

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/a/test(invalid)
```

2) Directory match url pattern

It always starts with forward slash '/' 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 will start with '*' symbol having some extension.

ex:

web.xml

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

request url

```
http://localhost:2525/DateApp/x/y/z        (invalid)
http://localhost:2525/DateApp/x/y/z.do     (valid)
```

http://localhost:2525/DateApp/test.do (valid)

MIME Types / setContentType

MIME stands for Multipurpose Internet Mail Extension.

There are four formats we can display the output in servlet.

1) text/html

It will display the output in html format.

2) text/xml

It will display the output in xml format.

3) application/ms-word

It will display the output in word format.

4) application/vnd.ms-excel

It will display the output in xml format.

Deployment Directory structure

```
MIMEApp
|
|----Java Resources
|       |
        |------src
                |
                |---com.ihub.www
                        |
                        |----TestSrv1.java
                        |----TestSrv2.java
                        |----TestSrv3.java
                        |----TestSrv4.java
|
|----Web Content
          |
          |------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>SNO</th><th>SNAME</th><th>SADD</th></tr>");
                pw.println("<tr><td>101</td><td>raja</td><td>hyd</td></tr>");
                pw.println("<tr><td>102</td><td>ravi</td><td>delhi</td></tr>");
                pw.println("<tr><td>103</td><td>ramana</td><td>vizag</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>SNO</th><th>SNAME</th><th>SADD</th></tr>");
                pw.println("<tr><td>101</td><td>raja</td><td>hyd</td></tr>");
                pw.println("<tr><td>102</td><td>ravi</td><td>delhi</td></tr>");
                pw.println("<tr><td>103</td><td>ramana</td><td>vizag</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>SNO</th><th>SNAME</th><th>SADD</th></tr>");
		pw.println("<tr><td>101</td><td>raja</td><td>hyd</td></tr>");
		pw.println("<tr><td>102</td><td>ravi</td><td>delhi</td></tr>");
		pw.println("<tr><td>103</td><td>ramana</td><td>vizag</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>SNO</th><th>SNAME</th><th>SADD</th></tr>");
                pw.println("<tr><td>101</td><td>raja</td><td>hyd</td></tr>");
                pw.println("<tr><td>102</td><td>ravi</td><td>delhi</td></tr>");
                pw.println("<tr><td>103</td><td>ramana</td><td>vizag</td></tr>");
                pw.println("</table>");

                pw.close();

        }
}
```

Note:

If a web application contains multiple servlets then each servlet we need to configure in web.xml file using <servlet> and <servlet-mappin> 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

```
        localhost:2525/MIMEApp/html
        localhost:2525/MIMEApp/xml
        localhost:2525/MIMEApp/word
        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) Servelt 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 is quit complex.
- To overcome this limitation we need to use HTML to Servlet communication.
- In html to servlet commucation we can give request to the servlet program by using html based hyperlinks or form pages.
- The request which is generated by using hyperlink does not carry the data.
- The request which is generated by using form page will carry the data.
- In html based hyperlink to servlet communication we need to type our request url as href url.

ex:

```
<a href="http://localhost:2525/DateApp/test"> clickMe </a>
```

In html based form page to servlet communication we need to type our request url as

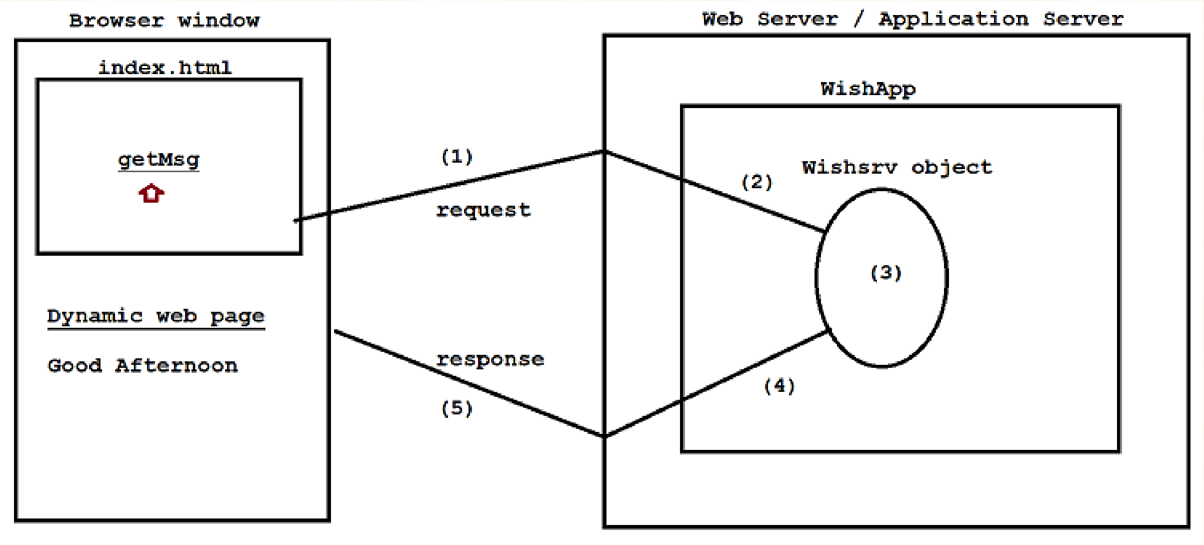
action url.

ex:

```
<form action="http://localhost:2525/DateApp/test">
        -
        -
</form>
```

Example application on HTML based hyperlink to servlet communication

Diagram: servlet3.1



Deployment Directory structure

```
WishApp
|
|----Java Resources
        |
        |-----src
                |
                |----com.ihub.www
                        |
                        |-----WishSrv.java
|
|----Web Content
        |
        |-----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 our class with GenericServlet class because it will not give HTTP protocol features.
- It is always recommanded to extends a class with HttpServlet class because it will give HTTP protocol features.

index.html

```
<center>


        <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>



</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);
```

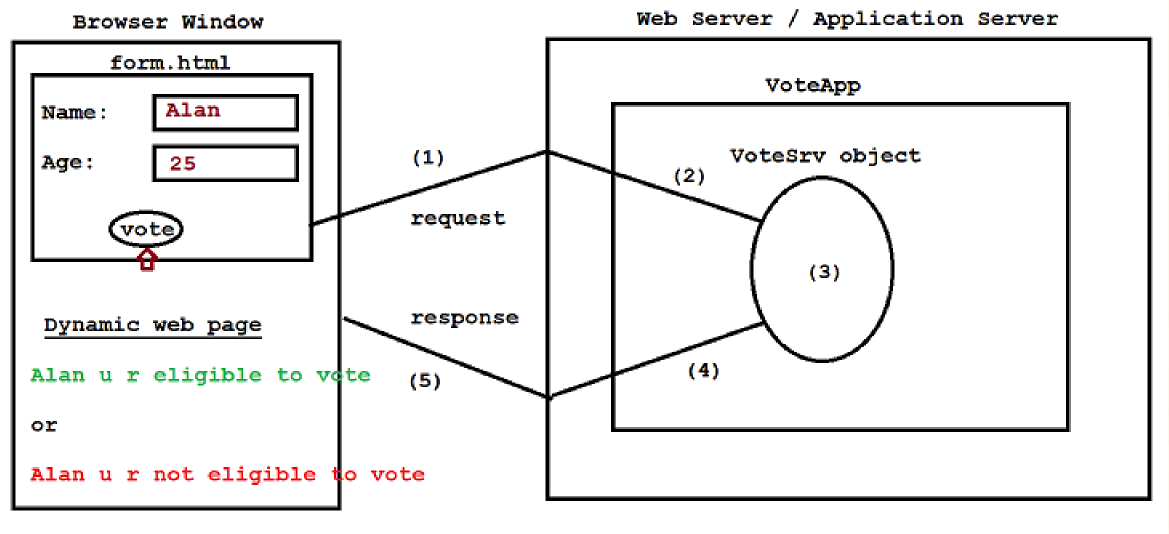
```
                if(h<12)
                        pw.println("<center><h1>Good Morning</h1></center>");
                else if(h<16)
                        pw.println("<center><h1>Good Afternoon</h1></center>");
                else if(h<20)
                        pw.println("<center><h1>Good Evening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

                pw.close();
        }
}
```

Request url

http://localhost:2525/WishApp

Example application on HTML based form page to servlet communication

Diagram: servlet3.2



Deployment Directory structure

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

Note:

- In above application we need to add "servlet-api.jar" file in project build path.
- We can send the request 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 with HttpServlet class , it is never recommanded to use service(-,-) method because it is designed according to HTTP protocol.
- It is always recommanded to use doXxx(-,-) method because they have designed according to HTTP protocol.

We have seven doXxx(-,-) methods.

1) doGet(-,-)

2) doPost(-,-)

3) doPut(-,-)

4) doDelete(-,-)

5) doHead(-,-)

6) doOption(-,-)

7) doTrace(-,-)

prototype of doXxx(-,-)

```
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

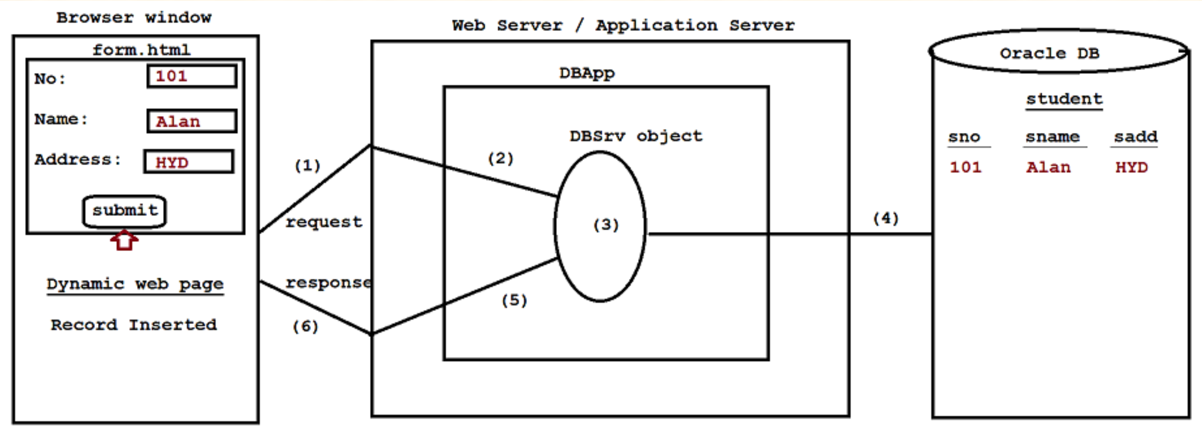
```
package com.ihub.www;


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");


        //convert 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/

HTML to Servlet to Database Communication

Diagram : servlet4.1



Deployment Directory structure

```
DBApp
|
|---Java Resources
|       |
        |------src
                |
                |----com.ihub.www
                        |
                        |----DBSrv.java
|
|---Web content
        |
        |----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 in "WEB-INF/lib" folder seperately.

form.html

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

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

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

	Address: <input type="text" name="t3"/> <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>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");

                //storing form data to 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)
                {
                        pw.println(e);
                }
                pw.close();
        }

}
```

Request url

http://localhost:2525/DBApp/

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

GET	POST
It is a default methodology.	It is not a default methodology.
It will carry limited amount of data.	It will carry unlimited amount of data.
It sends the request fastly.	It sends the request bit slow.
It not suitable for secure data.	It is suitable for secure data.
It shows the query string.	It does not show the query string.
It is not suitable to perform file uploading or encryption.	It is suitable to perform file uploading or encryption.
To process get methodology we will use doGet(-,-) method.	To process post methodology we will use doPost(-,-) method.

Form validation

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

Form validation can be performed in two ways.

1)Client side form validation

A validation which is performed at client side is called client side form validation.

To perform client side form validation we need to use javascript.

2)Server side form validation

A validation which is performed at server side is called server side form validation.

To perform server side form validation we need to use java/servelts.

Deployment Directory structure

```
ValidationApp
|
|-----Java Resources
        |
        |-----src
                |
                |---com.ihub.www
                        |
                        |---TestSrv.java
|
|-----Web Content
        |
        |-----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>
                <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>


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


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


        </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>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>form.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.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 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><b>Name is
mandatory</b></center>");
        return;
    }
    if(sage=="" || sage==null || sage.length()==0)
    {
        pw.println("<center><b>Age is
mandatory</b></center>");
        return;
    }
    else
    {
        try
        {
            age=Integer.parseInt(sage);
        }
        catch(NumberFormatException nfe)
        {
```

```
                    pw.println("<center><b>Age must be
numeric</b></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/

File Uploading

- The process of capturing a file from client machine file system and storing in server machine file system is called file uploading and reverse is called file downloading.
- While dealing with matrimonial application, job portal application , profile management application and etc. We need to upload or download a file.

- There is no specific API in servlet API to perform file uploading.
- To overcome this limitation we need to use third party API called JAVAZOOM API.
- JAVAZOOM API comes under zip file and once if we extracted then we will get three jar files.

ex:

```
        uploadbean.jar   --> Main jar file


        struts.jar          --> dependent jar file


        cos.jar              --> dependent jar file
```

JAVAZOOM API Download link

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

We can place file component in a form page as follow.

ex:

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

Deployment Directory structure

```
UploadApp
|
|-----Java Resources
        |
        |-----src
                |
                |---com.ihub.www
                        |
                        |---TestSrv.java
|
|-----Web Content
        |
        |-----form.html
        |
        |-----WEB-INF
                |
                |----web.xml
                |
                |------lib
                        |
                        |--uploadbean.jar
                        |--struts.jar
                        |--cos.jar
```

Note:

- In above application we need to add "servlet-api.jar" and "uploadbean.jar" file in project build path.
- Copy and paste "javazoom api" jar file 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>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>form.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.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 TestSrv extends HttpServlet
{
    protected void doPost(HttpServletRequest req,HttpServletResponse res)throws ServletException,IOException
    {
        PrintWriter pw=res.getWriter();
        res.setContentType("text/html");

        //upload the files
        try
        {
            UploadBean ub=new UploadBean();
            ub.setFolderstore("C:\\anil");
            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/

Servlet Life Cycle methods

- ✓ There are three life cycle methods in servlets.

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 used for request arrival event.
- This method will execute when request goes to servlet program.

3) public void destroy()

- It is used for destruction event.
- This method will execute just before servlet object destruction.

ServletConfig object

- A ServletConfig is an interface which is present in javax.servlet package.
- ServletConfig object will be created by the web container for every servlet.
- ServletConfig interface is used to read configuration information from web.xml file.

We can create ServletConfig object as follow.

ex:

```
ServletConfig config=getServletConfig();
```

Deployment Directory structure

```
ConfigApp
|
|-----Java Resources
        |
        |-----src
                |
                |---com.ihub.www
                        |
                        |---TestSrv.java
|
|-----Web Content
        |
        |-----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"> clickMe </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>


        <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>

  <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.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()+"<br>");
        }}
```

request url

http://localhost:2525/ConfigApp

ServletContext object

- ServletContext is an interface which is present in javax.servlet package.
- ServletContext object is created by the web container for every web application i.e it is one per web application.
- ServletContext object is used to read configuration information from web.xml file and it is for all servlets.
- We can create ServletContext object by using getServletContext() method.

ex:

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

ServletContext 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

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

```
|
|--------Web Content
|             |
              |----index.html
              |
              |-----WEB-INF
                      |
                      |---web.xml
```

Note:

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

index.html

```
<center>
        <h1>
                <a href="test">click</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("username", "system");
		context.setAttribute("password", "admin");
```

```
            pw.println(context.getAttribute("username")+"<br>");

            pw.println(context.getAttribute("password")+"<br>");


            context.removeAttribute("username");

            context.removeAttribute("password");


            pw.println(context.getAttribute("username")+"<br>");

            pw.println(context.getAttribute("password")+"<br>");



            pw.close();

        }


}
```

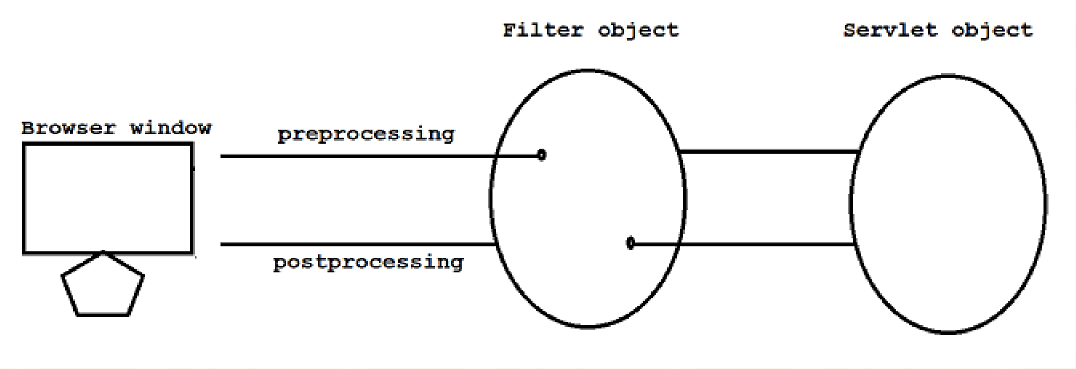
request url

http://localhost:2525/ServletContextApp/

Servlet Filters

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

Diagram:



The main advantages of using filter is to perform filter task such as

1) Counting number of request

2) To perform validation

3) Encyrption and Decryption

and etc.

- ✓ **Like Servlet, Filter is having it's own Filter API.**
- ✓ **The javax.servlet package contains thre 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
                        |
                        |----MyServlet.java
                        |----MyFilter.java
|
|----Web Content
        |
        |------index.html
        |
        |------WEB-INF
                |
                |---web.xml
```

Note:

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

index.html

```
<center>

        <h1>

                <a href="test"> clickMe </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>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>

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

</web-app>
```

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 is Invoked </h1></center><br>");
    }
}
```

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
{

    @Override
    public void init(FilterConfig config) throws ServletException {
        // TODO Auto-generated method stub

    }

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

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

                pw.println("<center><h1>Filter Invoked Before</h1></center><br>");

                chain.doFilter(req, res);

                pw.println("<center><h1>Filter Invoked After</h1></center><br>");


        }

        @Override
        public void destroy() {
                // TODO Auto-generated method stub

        }


}
```

Request url

http://localhost:2525/FilterApp/

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 servlet program will be discarded and output
- of destination servlet program goes to browser window as response.
- To forward the request we need to use RequestDispatcher object.
- We can create RequestDispatcher object as follow.

ex:

```
RequestDispatcher rd=req.getRequestDispatcher(url-pattern);

rd.forward(req,res);
```

2)Including the response

- In including the response, output of source servlet program and output of destination
- servlet program goes to browser window as response.
- In source servlet program output will be added but not code.
- To perform including the response we need to use RequestDispatcher object.
- We can create RequestDispatcher object as follow.

ex:

```
RequestDispatcher rd=req.getRequestDispatcher(url-pattern);

rd.include(req,res);
```

Deployment Directory structure

```
STSApp1
|
|----Java Resources
|       |
        |------src
|               |
                |---com.ihub.www
                        |
                        |---TestSrv1.java
                        |---TestSrv2.java
|
|----Web Content
        |
        |------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>
        <form action="test1">

                Username : <input type="text" name="t1"/> <br>
                Password : <input type="password" name="t2"/> <br>

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

        </form>
</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>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>index.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 uname=req.getParameter("t1");
        String pwd=req.getParameter("t2");

        if(pwd.equals("admin"))
        {
            RequestDispatcher rd=req.getRequestDispatcher("test2");
            rd.forward(req,res);
        }
        else
        {
            pw.println("<center><b><font color='red'>Sorry! incorrect
username or password</font></b></center>");
            RequestDispatcher rd=req.getRequestDispatcher("index.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 will forward the request to the application which is present in same server or different server.
- To forward the request we need to use sendRedirect() method of HttpServletResponse object.

ex:
```
res.sendRedirect(url);
```

- It always sends a new request.
- It works within the server or output side the server.
- It uses browser window to forward a new request.

Deployment Directory structure

```
STSApp2
|
|----Java Resources
|       |
        |------src
|               |
                |---com.ihub.www
                        |
                        |---TestSrv.java
|
|----Web Content
        |
        |------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>
                <br><br>
                <a href="test?t1=hotels"> Hotels </a>

                <br><br>


                <a href="test?t1=railways"> Railways </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.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");


		String value=req.getParameter("t1");


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


		pw.close();


	}

}
```

Request url

http://localhost:2525/STSApp2/

Q)How to enable <load-on-startup> and what happends 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>


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

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

Q)How many scopes are there in Servlet?

We have three types of scopes in servlet.

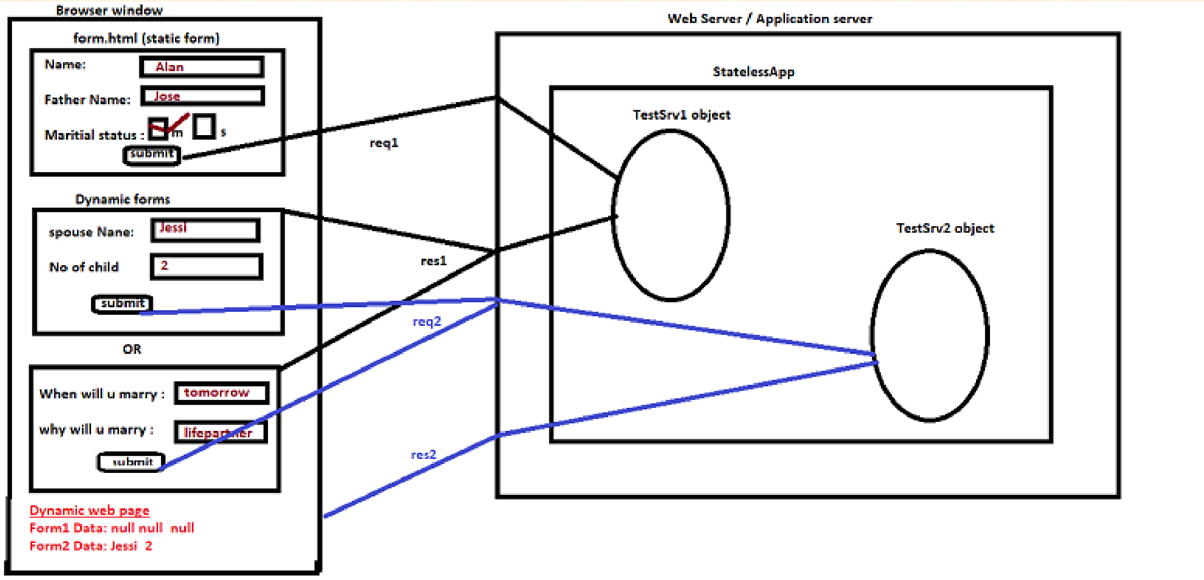
1) Request scope

2) Session scope

3) Application scope

Stateless Behaviour of web application

Diagram: servlet7.1



- ➢ Above diagram demostrate stateless behaviour of web application.
- ➢ By default every web application is a stateless web appplication.
- ➢ In stateless behaviour of web application , no web resource programs 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
|
|---Web Content
```

```
        |
        |----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">

        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>


 <session-config>
```

```
        <session-timeout>30</session-timeout>
 </session-config>
</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 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 fname=req.getParameter("t2");
                String ms=req.getParameter("t3");

                if(ms.equals("married"))
        {
            pw.println("<form action='test2'>");
            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'>");
            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 doGet(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("Form1 Data :"+name+" "+fname+" "+ms+"<br>");
		pw.println("Form2 Data :"+val1+" "+val2+"<br>");

		pw.close();
```

```
        }
}
```

Request url:

----

http://localhost:2525/StatelessApp/

Session

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

ex:

Login in to gmail and logout from gmail is one session.

Starting of java class and ending of java class is one session.

- In stateless web application no web resource programs can access previous request data while processing the current request during a session.
- To overcome this limitation we need to use session tracking or session management.

Session Tracking or Session Management

- Session tracking is used to make our web application as statefull web application even though our HTTP protocol is stateless.
- In stateless web application , no web resource programs can access previous request data while processing the current request during a session.
- In statefull web application , all web resource programs can access previous request data while processing the current request during a session.
- There are four technique to perform session tracking or session management.

1) Using hidden box fields

2) HttpCookies

3) HttpSession with Cookies

4) URL Rewriting

3) HttpSession with Cookies

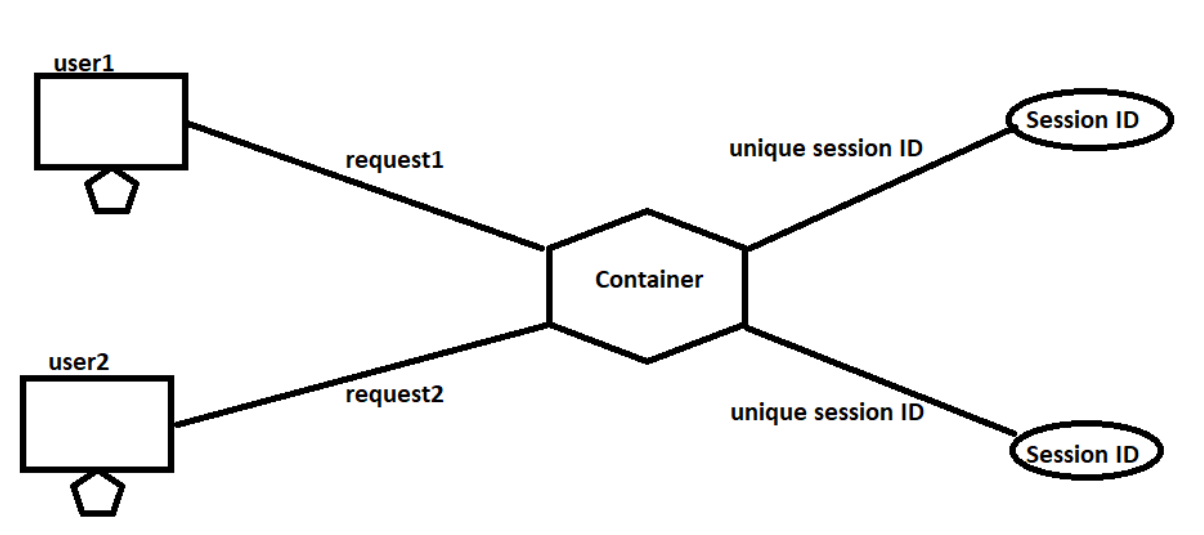
- In HttpSession with cookies, for every request it will create a unique session id.
- Our web container uses that session id to identify a user is a new user or existing user.

HttpSession object we can use to perform following task.

Bind the objects

To manipuate the data in HttpSession object.

Diagram:servlet7.2



Deployment Directory structure

```
SessionApp
|
|---Java Resources
|        |
         |------src
                  |
                  |---com.ihub.www
```

```
                    |
                    |---TestSrv1.java
                    |---TestSrv2.java
|
|---Web Content
      |
      |----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">
      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>

 <session-config>
        <session-timeout>30</session-timeout>
 </session-config>
</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 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 fname=req.getParameter("t2");
                String ms=req.getParameter("t3");

                //storing 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'>");
                        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'>");
                        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 doGet(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("Form1 Data :"+name+" "+fname+" "+ms+"<br>");
                pw.println("Form2 Data :"+val1+" "+val2+"<br>");

                pw.close();

        }
}
```

Request url

http://localhost:2525/SessionApp/