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.

ex:

home page

aboutus page

contactus page

services page

and etc.

2) Dynamic web pages

A web page with no fixed content is called dynamic web page.

ex:

live cricket score page

gmail inbox page

stock market share value page

and etc.

We have two types of web resource programs.

1) Static web resource program

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

ex:

Html program

CSS program

Bootstrap program

Angularjs program

Reactjs program

and etc.

2) Dynamic web resource program

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

ex:

servlet program

jsp program

and etc.

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

1) Client side web resource programs

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

ex:

html program

css program

bootstrap program

angularjs program

reactjs program

and etc.

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

2) Server side web resource programs

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

ex:

servlet program

jsp program

and etc.

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

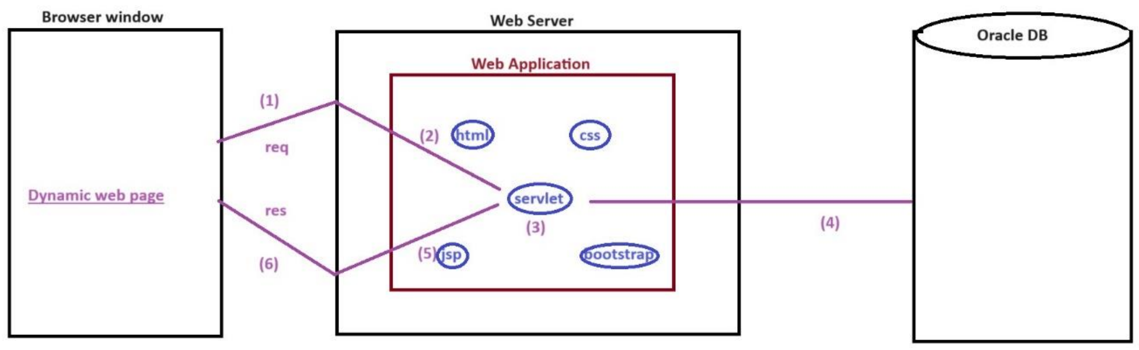
Web Application and Web Resource program Execution

===

Java application will execute manually.

Web application and web resource program wille execute at the time when we have requested.Hence there is no chance of executing them separately.

Diagram: servlet1.1



With respect to the diagram:

1) Enduser will give the request to web resource program.

2) Our server will trap that request and it passes that request to appropriate web resource program.

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

4) Web resource program will communicate with database software if neccessary.

5) Web resource program will give the output to web server.

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

Web Server

============

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

ex:

Tomcat , Resin and etc.

Responsibilities of a web server

----

1) It takes contineous request from client.

2) It passes that request to appropriate web resource program.

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

4) It provides middleware services only to deployed web application.

5) It provides environment to execute client side web resource programs at browser

window.

6) It takes the output from web resource program and sends to browser window

as dynamic response.

7) It automates whole process of web application and web resource program execution.

Web Container

=============

It is a software application or a program which 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 considers servlet container and jsp container are web containers.

Every server is designed to support servlet container and jsp container.

Note:

To execute Java we need JRE/JVM.

To execute serlvet program we need servlet container.

To execute jsp program we need jsp container.

Tomcat

========

Server	:	Web server
Version	:	7.x
Vendor	:	Apache foundation
website	:	www.tomcat.apache.org
Port No	:	8080
servlet container	:	catalina
Jsp container	:	Jasper
Download link	:	

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

Tomcat is a not a container.

It is a server containing servlet container and jsp container.

Before 6.x version tomcat is known as web server. But from 6.x version onwards tomcat is also known as application server.

Tomcat installation will ask following things.

1) Http portno

2) adminstrative username and password

3) JRE location (parallel to jdk)

4) Tomcat installation location.

Tomcat server installation

===========================

Double click to tomcat software --> yes --> next --> I agree -->

type of install : Full --> next -->

Http Connector port : 2525

adminstrator username : admin

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

How to set tomcat server to manual mode

==

start --> services (view local services) --> Select Apache tomcat --> click to stop link --> double click to Apache tomcat --> startup type : manual --> apply --> ok.

Servlet

========

It is a dynamic web resource program which is used to enhance the functionality of web server,proxy server or application server.

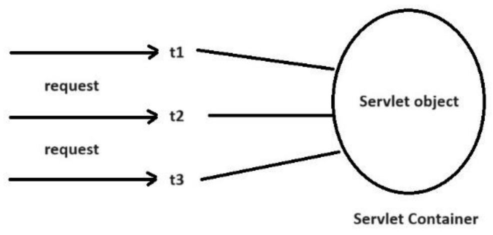
or

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

or

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

Diagram: servlet2.1



Important Terminologies

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

Servlet API's

==============

API is a collection of packages.

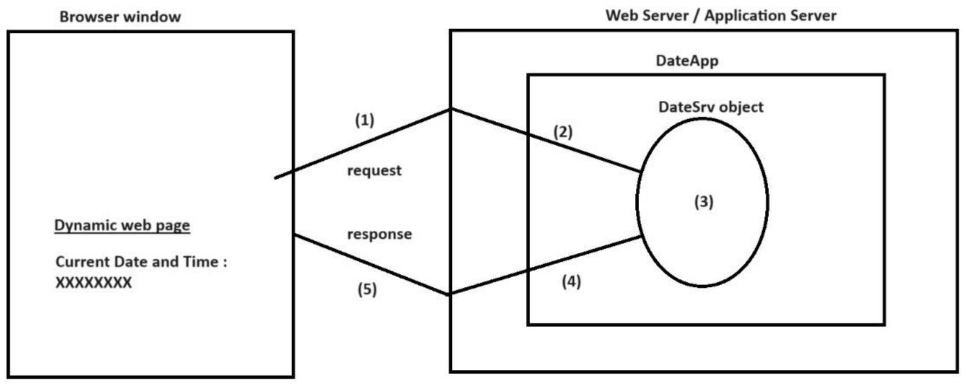
ex:

```
     javax.servlet
     javax.servlet.http
```

First web application development having Servlet program as web resource program

===

Diagram: servlet2.2



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 -->

generate web.xml file(click to checkbox) --> Finish.

step3:

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

ex:

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

libraries --> Add external jars --> select servlet-api.jar --> open -->ok.

step4:

Create "com.ihub.www" package inside "Java Resource/src" folder.

ex:

right click to src --> new --> package --> name : com.ihub.www --> finish.

step5:

Create a servlet program i.e "DateSrv" 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 program in web.xml file.

ex:

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 Tomcat 7.x server to eclipse IDE.

ex:

window --> preferences --> server --> runtime environment -->

click to add button --> select Apache Tomcat 7.0 --> Next -->

select tomcat installation directory(click to browse) --> ok --> finish-->ok.

step8:

Run dynamic project i.e DateApp.

ex:

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

Apache tomcat 7.0 server --> next --> finish.

step9:

Test the application by using below request below.

ex:

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

Types of URL Patterns

======================

Every servlet will recognize with the help of url pattern only.

Our client, web container, other web resource programs will recognize each servlet by using url pattern only.

URL pattern will hide technology name or class name from the 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 server is designed to support three types of url patterns.

1) Exact match url pattern

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

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.

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.

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/x/y/z.it     --invalid
```

MIME Types or setContentType

============================

MIME stands for Multipurpose Internet Mail Extension.

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

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 excel 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 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 Communiction

===================

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 use HTML to Servlet communication.

In html to servlet communication we can send the request to servlet by using html based hyperlinks and form pages.

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

But a 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/MIMEApp/html">  click Here </a>
```

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

ex:

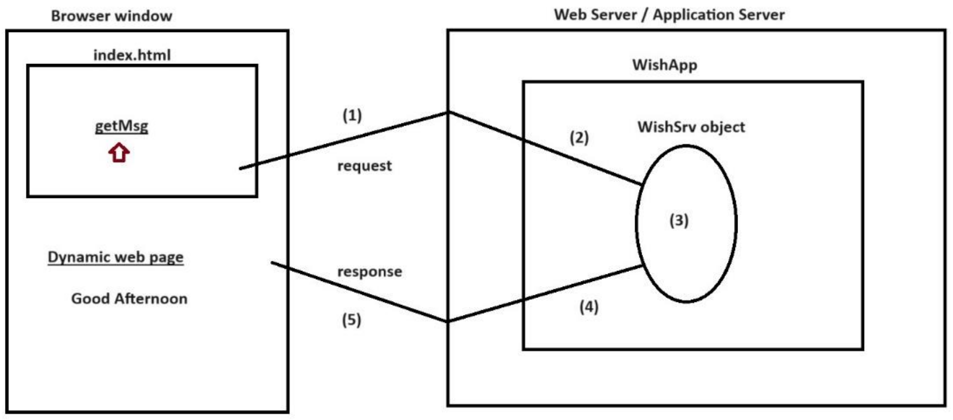
```
<form  action="http://localhost:2525/MIMEApp/html">
        -
        -
```

</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 not recommanded to extends a servlet class with GenericServlet class because it is not designed to give HTTP protocol features.

It is always recommanded to extends a servlet class with HttpServlet class because it is designed to 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>

  <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();
```

```
                //convert time to 24 hours
                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();

        }
}
```

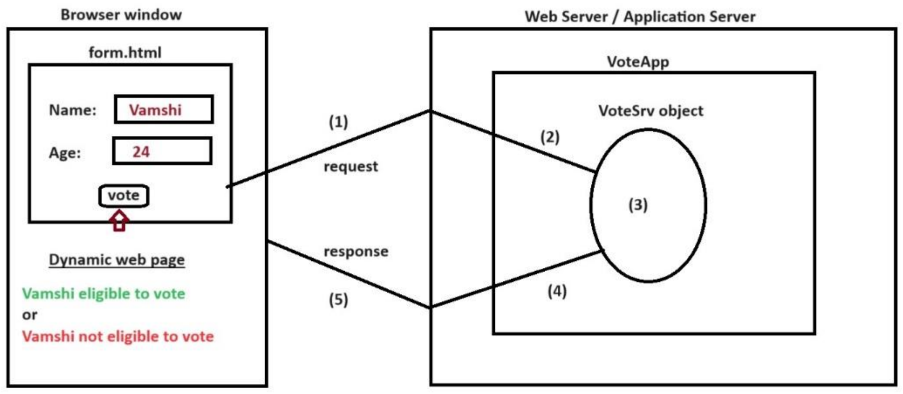
Request url

```
        http://localhost:2525/WishApp
```

Example application on HTML based form page to Servlet Communication

===
=

Diagram: servlet4.1



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 not recommanded to use service(-,-) method because it is not designed according to HTTP protocol.

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

We have seven doXxx(-,-) method as follow.

1)doGet(-,-)

2)doPost(-,-)

3)doPut(-,-)

4)doDelete(-,-)

5)doOption(-,-)

6)doTrace(-,-)

7)doHead(-,-)

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

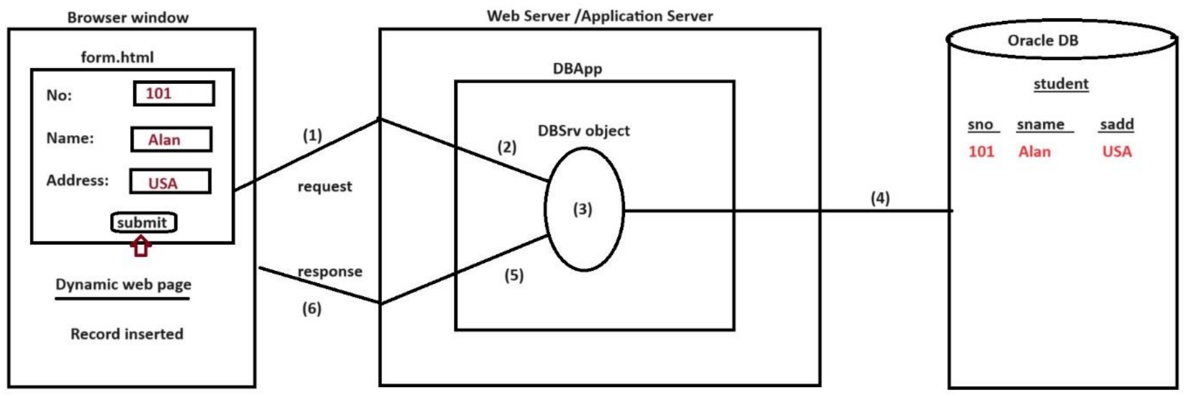
----

	http://localhost:2525/VoteApp/

Servlet to Database Communication

====================================

Diagram: servlet4.2



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.

Student table

============

drop table student;

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

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");

		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>Record Not
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/

Interview Question

=================

Q)Write a java program to multiply two arrays?

input:

```
        arr1 = {2,4,9};
        arr2 = {3,5};

output:
        8715 (249*35)

ex:
---

package com.ihub.www;

public class Example
{
        public static void main(String[] args)
        {
                int[] arr1 = {2,4,9};
                int[] arr2 = {3,5};

                String res1=arrayToString(arr1);
                String res2=arrayToString(arr2);

                //convert string to integer
                int a=Integer.parseInt(res1);
                int b=Integer.parseInt(res2);
                System.out.println(a*b);


        }
        //callie method
        public static String arrayToString(int[] arr)
        {
                StringBuffer sb=new StringBuffer();

                //for each loop
                for(int i:arr)
                {
                        sb.append(i);
                }

                return sb.toString();
        }
}
```

Form Validation

================

The process of checking format and pattern of form data is called form validation and 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,Php,.net,Python and etc.

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>

                <!--  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>

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

                                <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>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/

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 sends the request bit slow.
It will carry limited amount of data.	It will carry unlimited amount of data.
It is not suitable for secure data.	It is suitable for secure data.
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.

File Uploading

================

The process of capturing the 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 and etc.WE need to upload and download a file.

In servlet we don't have specific API to perform file uploading.

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

JAVAZOOM API comes with zip format 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 use file form 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
                        |
                        |----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 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>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");


                //file uploading logic
                try
                {
                        UploadBean ub=new UploadBean();
                        ub.setFolderstore("C:\\arun");
                        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

----

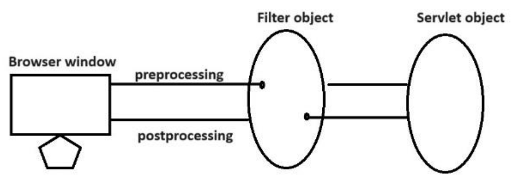
    http://localhost:2525/UploadApp/

Servlet Filters

===============

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

Diagram: servlet6.1



The main objective of Filter are

1) To count number of request coming to the application.

2) To peform form validations.

3) To perform encryption and decryption.

Like Servlet, Filter is also having having it's own Filter API.

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

1) Filter

2) FilterConfig

3) FilterChain

Deployment Directory structure

```
----------------------------
FilterApp
|
|---Java Resources
          |
          |-----src
                    |
                    |---com.ihub.www
                              |
                              |-----MyFilter.java
                              |-----MyServlet.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"> 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>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>
```

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>");
                        chain.doFilter(req,res);
                        pw.println("<center><h1> Filter Invoked After
</h1></center>");

        }



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

        }

}
```

MySERVLET

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>");

	}
}
```

Request url

----

	http://localhost:2525/FilterApp/

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 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.

Deployment Directroy structure

LifeCycleApp

|

```
|---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"> 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>
 </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.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>Servlet method called</h1></center>");

        }
        public void destroy()
        {

        }
}
```

Request url

```
        http://localhost:2525/LifeCycleApp/
```

Assignment

===========

ServletConfig object

====================

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

ServletConfig object will be created by the web container for every 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 four 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

```
ServletConfigApp
|
|---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"> 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>
        <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 name=e.nextElement();
                        pw.println(name+"<br>");
                }
                pw.println(config.getServletName());

                pw.close();

        }
}
```

Request url

----

        http://localhost:2525/ServletConfigApp/

ServletContext object

======================

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

ServletContext object created by the 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 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.

Deployment Directory structure

```
------------------------------
ServletContextApp
|
|---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">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>

 </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>



</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=e.nextElement();
                        pw.println(s+"<br>");
                }


                pw.close();


        }
}
```

Request url

----

        http://localhost:2525/ServletConfigApp/

Servlet to Servlet Communication

================================

Servlet to servlet communication is also know 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, the output of source servlet program will be discarded and output of destination servlet program goes to brower window as dynamic response.

To forward the request we need to use RequestDispatcher object.

We can create RequestDispatcher object as follow.

ex:

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

2) Including the response

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

To perform including the response we need to take the support of RequestDispatcher object.

We can create RequestDispatcher object as follow.

ex:

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

Deployment Directory structure

```
---------------------------
STSApp1
|
|---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" method="GET">

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

        Password: <input type="password" name="t2"/> <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>
 </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("<b style='color:red'> Sorry! Incorrect username or
password </b>");
                        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>Thank you for login 
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.

It always sends new request.

It uses browser window to send the request.

It will work inside as well as outside of the server.

We can perform send redirection as follow.

ex:

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

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"> Trains </a>  <br><br>

        </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 val=req.getParameter("t1");

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

                pw.close();
        }
}
```

Request url

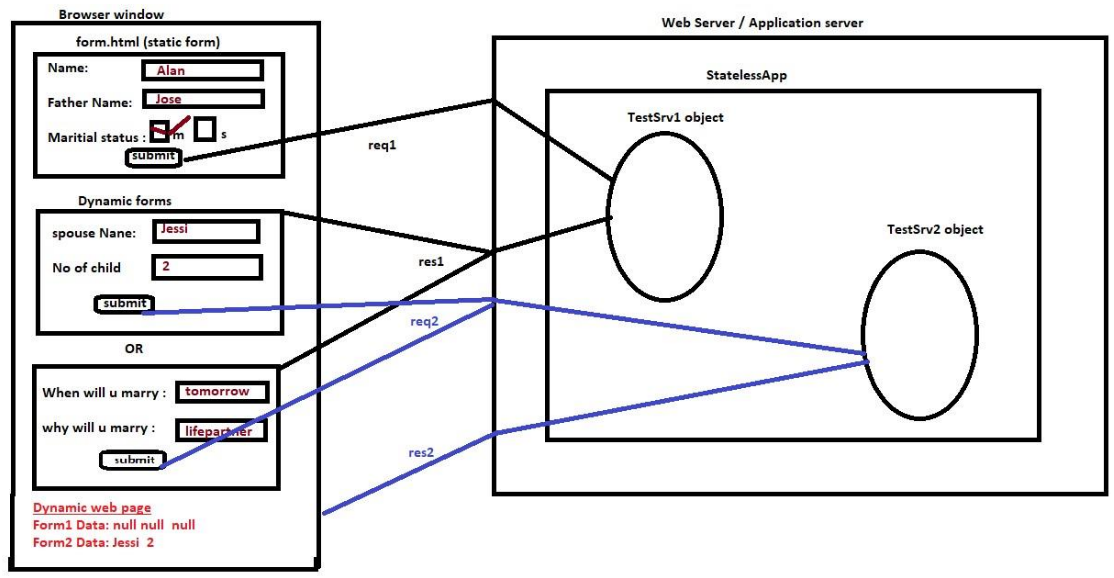
----

    http://localhost:2525/STSApp2/

Stateless Behaviour of web application

==

Diagram: servlet8.1



Above application demostrate stateless behaviour of web application.

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" method="GET">

          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>
  </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.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' method='GET'>");
			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='GET'>");
                        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/

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>
```

Once we enabled <load-on-startup> then our servlet container will create servlet object during the server startup or during the deployment of web application.

Session

=======

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

ex:

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

Login to gmail and logout from gmail is one session.

Session Tracking / Session Management

======================================

Session tracking makes our application as statefull web application even though 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 web 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

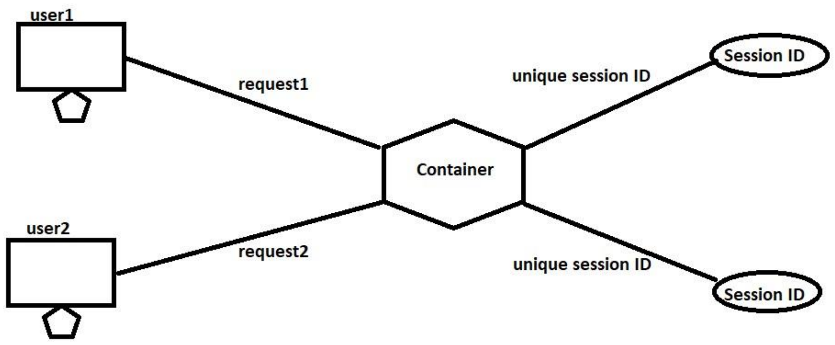
HttpSession

============

A HttpSession is an interface which is present in javax.servlet package.

HttpSession object is used to create a unique session ID for every request to identify that user is a existing user or new user.

Diagram: servlet9.1



We can create HttpSession object as follow.

ex:

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

The main objective of HttpSession are

1) To bind objects

2) To manipulate the data which is present in HttpSession.

Deployment Directory structure

SessionTrackingApp

|

```
|---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" method="GET">
```

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

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

        Maritital 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");

                //create HttpSession object
                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='GET'>");
                        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='GET'>");
                        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/SessionTrackingApp/