SERVLETS:

Types of Applications:

1. Standalone Applications
2. Client-Server Applications
3. Web Applications
4. Distributed Applications
5. Enterprise Applications

1. Standalone Applications:

Applications that can be accessed by single user at a time are called as Standalone Applications.

Ex: MS-Word, Music Player etc.

- Standalone Applications can be implemented using C, C++, and Java etc.

Problems:

a) Applications have to be installed across all the machines. This may give maintenance when you change application features.

b) Data sharing is not possible.

2. Client-Server Applications :

In the case of Client-Server Applications, Application can be divided into two parts. One Part will be installed on the Server Machine and other part will be installed on multiple client machines.

i.e. multiple clients can access the centralized server.

Ex: Yahoo Messenger Talk etc.

- Client-Server Applications can be implemented using C,C++,and Java etc.
- Data sharing is possible.

Problems:

Client software has to install across all the machines. This may give maintenance when you change Client software features.

3. Web Applications :

To solve the problem with client-server application Web-based application is introduced which is run in WWW.

In the case of Web Applications, Application software will be installed on the Web Server Machine only.

i.e. multiple clients can access the centralized web server using any web Browser.

Ex: jtcindia.org, gmail.com

- Web Applications can be implemented using Servlets, JSP, Struts and JSF etc.
- Data sharing is possible.

- No Maintenance problem because modifications will happen only at web server.

4. Distributed application:

Distributed application is current trend in company which allows the business partner to share the information among them

5. Enterprise Applications:

An enterprise application is the term used to describe applications -- or software -- that a business would use to assist the organization in solving enterprise problems. When the word "enterprise" is combined with "application," it usually refers to a software platform that is too large and too complex for individual or small business use.

Enterprise applications are typically designed to interface or integrate with other enterprise applications used within the organization, and to be deployed across a variety of networks (Internet, Intranet and corporate networks) while meeting strict requirements for security and administration management.

If you want to develop Web Application then it can be of two types:

- **Static Web Application**
- **Dynamic Web Application**

Static Web Application:

In this type of application view will be same for all the client and client cannot interact with the application by sending some data to server for processing.

Dynamic Web Application:

In this type of application client can send the data to server and that data will be processed and depending on the processing the response will be given to client so the view of application will vary from client to client.

Servlet can be used for developing dynamic Web Application.

Servlets is Server side component which receives the Request from Browser, Process the Request and sends the Response to Browser.

General Technical Terms:

1. Web Server
2. Web Client
3. Web Container
4. Http
5. TCP/IP
6. DNS
7. Web Application

8. Web Technologies
9. Web Frameworks
10. HTML

Web Server: Web Server is an Application which receives the Http Request from Web Client and processes that request with the help of Web Container and send the Response to Web Client.

Ex: Apache Server, PWS (ASP), IIS

Web Client: Web Client is an Application which sends the Http Request to Web Server and receives the Response from Web Server.

Ex: IE, OPera, Mozilla etc.

Web Container: Web Container is an Application is responsible managing the complete lifecycle of Servlet or JSP. Web Container offers the following free services.

A) Low Level Services

1. IO Streams
2. Threads
3. Networking

B) Middle Level Services

1. Resource Management
2. Lifecycle Management
3. Declarative Security Management
4. JSP Lifecycle support

*** As Developer, You have to write the code only for High Level Services (Business operations of an Application which you are developing.)***

Http (Hyper Text Transfer Protocol):

Http sits on the Top of TCP/IP for Transferring Web Client Info to Web Server and Web Server Info to Web Client.

TCP/IP:

Original Data Transfer will happen thro TCP/IP

IP (Internet Protocol) is responsible for carrying data from one place to another.

TCP (Transport Control Protocol) sits on the Top of IP and monitors the Data Transmission.

DNS: Domain Naming Service is registry where domain names will be binded with IP Address.

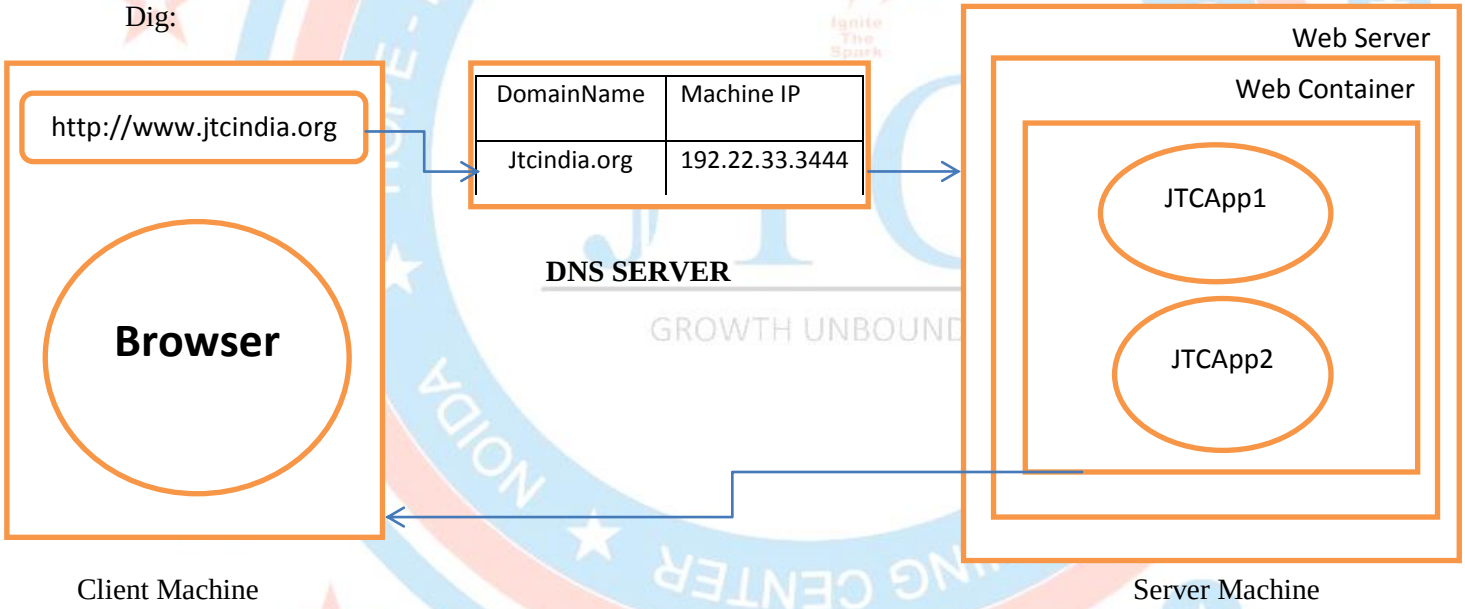
Web Technologies: Servlets, JSP

Web Frameworks: Struts, JSF, SpringMVC etc

Servlets:

- Servlets is Web Technology which is used to develop the Web Applications and provided by Sun MicroSystem.
- Web Application is a type of application in which whole application will be on server machine and client will access the application using browser.
- To develop Web Applications you need to develop Client Side Component as well as Server Side Component.
- Client Side component can be developed using HTML/JSP/XHTML etc.
- Server Side Component can be developed using Servlet/JSP/Filter etc.

Dig:



When you hit the browser with some URL 1st request will be given to DNS server. In DNS server domain will be resolved of IP address and request will be send to the server where application is running.

Once server receives the request it processes the request and send the response to the client.

Servers identify the client IP address from the Http request.

Servlets Version:

1. Servlet2 2.2 is released under J2EE1.2
2. Servlets 2.3 is released under J2EE1.3
3. Servlets 2.4 is released under J2EE1.4
4. Servlet 2.5 is released under JEE5
5. Servlet 3.0 is released under JEE6
6. Servlet 3.1 is released under JEE7

Setup Tomcat 6

1. Make sure that JDK 6 is installed.
2. Install Tomcat 6 with following steps:
   - Click on installer called "apache-tomcat-6.0.37" which is in student DVD under Tomcat 6.0 folder.
   - Click on next button.
   - Click on I agree button.
   - Select installation type as FULL.
   - Click on next button.
   - Provide installation location as E:\Tomcat 6.0 and Click on Nextbutton.
   - Provide
     - Port number :9999
     - Username: som
     - Password:som
   - Click on next button.
   - Make sure that JRE 6 is select and click on install button.
   - Uncheck the 2 check boxes and click on finish button.
3. Observe Tomcat 6 Directory Structure:

   Diagram……..
4. Start the server with the following steps:
   - Start->Programs->Apache Tomcat 6.0->Configure Tomcat.
   - Modify startup type as manual and click on apply.
   - Click on START button.
5. Check the Tomcat status
   - Open the browser

- Type the following URL

  http://localhost:9999/

6. Stop the server with the following steps:
   - Start->Programs->Apache Tomcat 6.0->Configure Tomcat.
   - Click on next button.

Steps to devlop and run first Servlets 2.x based Web App

1. Consider the login requirement.

   Diagram……
2. Create the Dynamic web Project in eclipse as follows:
   - Select file->New -> Dynamic web Project
   - Provide
     - Project Name: Jtc1
     - Click on New Runtime button(for the first time)
       - Select Apache Tomcat 6.0 and click on Next button.
       - Browse the Tomcat Home Directory i.e E:\Tomcat 6.0
       - Click on finish button..
   - Click on finish button.
3. Observe the following Directory structure of Dynamic Web Project inn eclipse.

   Diagram……
4. Create login html under WebContent folder.
5. Create a package called com.jtcindia.servlets.
6. Create LoginServlet.java file in package com.jtcindia.servlets.
7. Update the web.xml with LoginServlet configuration.
8. Deploy into Tomcat 6 as follows:
   a. Right click on Jtc1.

b. Seelct Run As->Run on Server.

9. Open the Browser.
10. Provide the following URL:

    http://localhost:9999/jtc1/login.html

Jtc 1: Files Required:

Jtc 1: Example using various Parameters

Required Files:

==========

1.Login.html

2.web.xml

3.JtcServlet.java

1. Login.html

```
<html><head>
<title>Insert title here</title>
</head><body>
<h1>JTC Test</h1>
<hr /><br>
<form action="test.jtc" method="post">
<table>
<tr>
<td>UserName</td>
<td><input type="text" name="uname" />
</td></tr>
<tr>
<td colspan="2" align="center"><input
type="submit"value="Submit" />
</td></tr></table>
</form></body>
</html>
```

2. web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app>
<welcome-file-list>
<welcome-file>login.html</welcome-file>
</welcome-file-list>

<context-param>
<param-name>State</param-name>
<param-value>UTTAR PRADESH</param-value>
</context-param>

<context-param>
<param-name>country</param-name>
<param-value>INDIA</param-value>
</context-param>

<servlet>
<servlet-name>test</servlet-name>
<servlet-class>com.jtcindia.Servlets.JtcServlet</servlet-class>

<init-param>
<param-name>Email</param-name>
<param-value>som@jtcinda.org</param-value>
</init-param>
<init-param>
<param-name>Phone</param-name>
<param-value>33333</param-value>
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>

<servlet-mapping>
<servlet-name>test</servlet-name>
<url-pattern>/test.jtc</url-pattern>
</servlet-mapping>
</web-app>
```

3. JtcServlet.java

```
==============
package com.jtcindia.Servlets;
import .IOException;
import .PrintWriter;
import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.*;
public class JtcServlet extends HttpServlet {
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call    :+91-9990399111
* */
String phone;
String email;
String state;
String country;
public void init(ServletConfig sc) {
System.out.println("TestServlet -init()");
// 1.config parameter

phone = sc.getInitParameter("Phone");
email = sc.getInitParameter("Email");
// 2.context parameter

ServletContext ctx = sc.getServletContext();
state = ctx.getInitParameter("State");
country = ctx.getInitParameter("country");
}
public void service(HttpServletRequest req,
HttpServletResponse res)
throws IOException, ServletException {
System.out.println(("TestServlet-service()"));
// 3.Request parameter
String un = req.getParameter("uname");
// 4.display parameters

PrintWriter out = res.getWriter();
out.println("<h1>Username: " + un + "</h1>");
out.println("<h1>Phone: " + phone + "</h1>");
out.println("<h1>Email: " + email + "</h1>");
out.println("<h1>state: " + state + "</h1>");
out.println("<h1>Country: " + country + "</h1>");
}

public void destroy() {
System.out.println("TestServlet-destroy()");
}
}
```

Steps Tomcat 7

1. Make sure that JDK 7 is installed.
2. Install Tomcate 7 with the following steps:
    - Click on installer called "apache-tomcate-7.0.42" which is in student DVD under Tomcat 7 folder.
    - Click on next button.
    - Click on I agree button.
    - Select installation type as FULL.
    - Click on next button.
    - Provide installation location as E:\Tomcat 7.0 and click on next button.
    - Provide
        - Port number:8888
        - Username:som
        - Password:som
    - Click on next button.
    - Make sure that JDK 7 is selected and click on install button.
    - Uncheck the 2 boxes and click on finish button.
3. Observe Tomcat 7 Directory structure:

   Diagram……
4. Start the server with the following steps:-
    - Start->Program->Apache Tomcat 7.0-> Configure Tomcat.
    - Modify startup type as manual and click on Apply.
    - Click on START button.
5. Check the Tomcat status
    - Open the browser

- Type the following URL

  http://localhost:8888

6. Stop the server with the following steps:
   - Start->Program->Apache Tomcat 7.0-> Configure Tomcat..
   - Click on stop button.

Steps to devlop and run first Servlets 3.x based web App

1. Consider the login requirement (same as jtc1).
2. Create the Dynamic Web project in eclipse as follows:
   - Select File ->New ->Dynamic Web Project.
   - Provide
     - Project Name:Jtc2
     - Click on New Runtime button(for the first time)
       - Select Appache Tomcate 7.0 and click on next button.
       - Browser the Tomcat Home Directory i.e E:\Tomcat 7.0
       - Click on finish button.
   - Click on finish button.
3. Observe the following Directory structure of Dynamic Web Project in eclipse.

   Diagram……..

4. Create login.html under Webcontent folder.
5. Create a package called com.jtcindia.servlets.
6. Create LoginServlet.java file in package com.jtcindia.servlets.
7. Deploy into Tomcat 7 as follows:
   - Right click on Jtc2.
   - Select Run As->Run on server.

8. Open the Browser
9. Provide the following URL:

http://localhost:8888/Jtc2/login.html

Jtc2: Files Required:

1. Login.html
2. LoginServlet.java

1. **Login.html**

   //same as Jtc1

2. **LoginServlet.java**

```
package com.jtcindia.servlets;

import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;

/*
 * @AUTHOR:SomPrakash Rai
 * @company:Java Training Center
 * @see: www.jtcindia.org
 */
@WebServlet(name="myLogin", urlPatterns={"/login.jtc"})
public class LoginServlet extends HttpServlet {
	//Same as Jtc1

}
```

Request Processing Flow for Jtc1 & Jtc2

- Click sends the request for a resource called login.html.
- Response of login.html will be displayed to client.
- Client entr the Username and Password and submits the forms.
- Browser collects the form data & form action and sends to server via HTTP.

- Server receives the request and delegates to Web Container.
- Web Container collects the foem action value i.e URL pattern.
- Web Container checks whether any servlet is configured with the matching URL pattern.
- Form the XML information (Servlet 2.x).
- From the Annotation information (Servlet 3.x).
- After indentifying the servlet class, Web Container performs the servlet life cycle process.
- Finally response of LoginServlet will be delivered to client.

Working With Welcome files

- You can make one or more web resource as main web pages for your web application by specifying welcome pages.

```
<welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>
```

- Once welcome files are specifyed, you can send the request using the following URLs.
    - http://localhost:9999/Jtc1/
    - http://localhost:8888/Jtc1/
- When you specify multiple resource in the welcome file list then those files will be verified in the order as defined in the web.xml file.
    - When login.html is found the that will be displayed.
    - When login.html is not found it goes to default.html.
    - When default.html is found the that will be displayed.

- When default.html is not found it goes to index.html.
- And so on.

- With servlet 3.0 you have two options to use welcome files.
  - Specify welcome files as above in web.xml.
  - Use index.xml- which is default welcome file.

Working With Various forms fields

Text fields	Radio buttons	Checkboxes
List box	Text area	

Jtc3: Files Required in Servlet 2.x:

1. Register.html
2. RegisterServlet.java
3. Web.xml

Register.html

Name som

Email som@jtcindia.com

Phone 9990399111

Gender Male Female

Module 1

Module2

Module3

Module4

Timings Select

Remarks Ok

Register Now

Name :som

Email :som@jtcindia.org

Phone :9990399111

Gender :Male

Module 1

Module2

RegisterServlet.java

```
String fn=req.getParameter("fname");
String ln=req.getParameter("lname");
String email=req.getParameter("email");
String phone=req.getParameter("phone");
String ge=req.getParameter("gender");
String ti=req.getParameter("timings");
String co=req.getParameter("course");
String re=req.getParameter("remarks");

//Process the data
If(couse!=null){
        for(String c:cous){
}
```

1. **Register.html**

```
<!DOCTYPE html>
<html>
<head>
<meta charset="ISO-8859-1">
<title>No.1 Online BookStore</title>

</head>
<body>

<h1>JTC BookStore</h1>
<h2>
<font color="green"><b>Customer Registeration
form</b></font>

</h2>
<form action="register.jtc">

<table bgcolor="light green">
<tr>
<td>First name</td>
<td><input type="text" name="fname">
</td>
</tr>
<tr>
<td>Last Name</td>
<td><input type="text" name="Lname">
</td>
</tr>
<tr>
<td>Email-ID</td>
<td><input type="text" name="email">
</td>
</tr>

<tr>
<td>phone no</td>
<td><input type="text" name="phone">
</td>
</tr>
<tr>
<td>Gender</td>
<td><input
type="radio"name="gender"value="male"/>Male<input
name="radio"name="gender"values="female"/>Female</td>
</tr>

<tr>
<td>Card Type</td>
<td><select name="ctype">
<option value="Visa">Visa</option><option
value="Master">Master</option><option
```

2. **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">
  <display-name>servlet3</display-name>
  <welcome-file-list>
    <welcome-file>regis.html</welcome-file>
  </welcome-file-list>
  <servlet>
  <servlet-name>register</servlet-name>
  <servlet-class>com.bookstore.servlet.RegisServlet</servlet-
class>
  </servlet>
  <servlet-mapping>
  <servlet-name>register</servlet-name>
  <url-pattern>/register.jtc</url-pattern>
  </servlet-mapping>
</web-app>
```

3. **RegisterServlet.java**

```
package com.bookstore.servlet;
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.http.*;
import javax.servlet.*;
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call    :+91-9990399111
* */

public class RegisServlet extends HttpServlet {
        public void init(ServletConfig sc){
                System.out.println("LoginServlet-init()");

        }
public void service(HttpServletRequest req,
HttpServletResponse res) throws IOException,
ServletException{
                System.out.println("TestServlet-service()");

                String fn=req.getParameter("fname");
                String ln=req.getParameter("lname");
                String ph=req.getParameter("phone");
                String em=req.getParameter("email");
```

```
value="Amex">Amex</option>
</select>
</td>
</tr>
<tr>
<td>Card Number</td>
<td>
<input
type="checkbox"name="color"value="red"/>Red<input
type="checkbox"name="color"value="Blue"/>
                                Blue<input
type="checkbox"name="color"value="Green/">Green</td>
</tr>
<tr>
<td colspan="2"align="center"><input
type="submit"value="Register Now"/></td>
</tr>
</table>
</form>
</body>
</html>
```

```
                System.out.println(fn);
                System.out.println(ln);
                System.out.println(ph);
                System.out.println(em);

                PrintWriter out=res.getWriter();

                out.println("<html>");
                out.println("<body>");
                out.println("<h1>JTC BookStore</h1>");
                out.println("<h2>Hi"+fn+" "+ln+"Ur Login
has been completed succesfull</h2>");
                out.println("</body>");
                out.println("</html>");
//              String un=req.get
        }
                public void destroy() {
                        System.out.println("LoginServlet-
destroy()");
                }

}
```

Jtc4: Files Required in Servlet 3.x:

1. **Register.html**(same as Jtc3)	2. **RegisterServlet.java**	3. **web.xml**

RegisterServlet.java

```
package com.bookstore.servlet;
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.http.*;
import javax.servlet.*;
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call    :+91-9990399111
* */
@Webservlet(name="regServlet",urlPatterns={"/register.jtc"})
public class RegisServlet extends HttpServlet {
        public void init(ServletConfig sc){
                System.out.println("LoginServlet-init()");
        }

        public void service(HttpServletRequest req,
HttpServletResponse res) throws IOException,
ServletException{
//Sam as Jtc3
```

Web.xml

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

When we use Servlet 3.0 then we can configure servlet using annation and XML.both

Which request will be processed in the following case ?.

Case 1:

Using Annotation

Servlet-name testServlet
url-pattern /test.jtc

IN HTML

<form action="test.jtc>…
<form action="hello.jtc">
<form action="admin.jtc">

- Test.jtc request will be processed .
- Hello.jtc & admin.jtc request will not be processed.

ServletName	ServletClass
testServlet	com.jtcindia.servlets.TestServlets
ServletName	Url-Patterns
testServlet	/test.jtc

Case 2:

Using Annotation

Servlet-name testServlet
url-pattern /test.jtc

IN HTML

<form action="test.jtc>…
<form action="hello.jtc">
<form action="admin.jtc">

- Test.jtc & hello.jtc request will be processed .
- Hello.jtc & admin.jtc request will not be processed.

ServletName	ServletClass
testServlet	com.jtcindia.servlets.TestServlets
ServletName	Url-Patterns
testServlet	/test.jtc
	/hello.jtc

Case 3:

Using Annatotaion

- Using Annotation

Servlet-name testServlet
url-pattern /admin.jtc

In Xml

Servlet-name testServlet
url-pattern /admin.jtc

In Html

<form action="test.jtc>…
<form action="hello.jtc">
<form action="admin.jtc">

- All the request will be

ServletName	ServletClass
testServlet	com.jtcindia.servlets.TestServlets
mytestServlet	com.jtcindia.servlets.TestServlets
ServletName	Url-Patterns
testServlet	/test.jtc
	/hello.jtc
myTestServlet	/admin.jtc

processed.

- Test.jtc, hello.jtc will be processed with one object.
- Admin .jtc another object will be created.

Case 4:
Using Annatotaion

- Using Annotation

Servlet-name testServlet
url-pattern /admin.jtc

In Xml

Servlet-name testServlet
url-pattern /admin.jtc

In Html

<form action="test.jtc>…
<form action="hello.jtc">
<form action="admin.jtc">

ServletName	ServletClass
testServlet	com.jtcindia.servlets.TestServlets
mytestServlet	com.jtcindia.servlets.TestServlets
ServletName	Url-Patterns
testServlet	/test.jtc
	/hello.jtc
myTestServlet	/admin.jtc

Annotation

web.xml

- Login.jtc & hello.jtc will not be processed.
- Admin.jtc will be processed.

Jtc 5:Files Required in Servlet 3.x:

1. index.html
2. TestServlet.java
3. Web.xml

1. Index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="test.jtc">
<input type="submit" value="test.jtc">
```

3. 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/x
ml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-
app_3_0.xsd"
        id="WebApp_ID" version="3.0">
        <display-name>Jtc1</display-name>
        <welcome-file-list>
```

```
</form><br/>
<form action="hello.jtc">
<input type="submit" value="hello.jtc">
</form><br/>
<form action="admin.jtc">
<input type="submit" value="admin.jtc">
</form>
</body>
</html>
```

2. TestServlet.java

```
package com.jtcindia.servlets;

import java.io.IOException;
import java.io.Writer;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import
javax.servlet.http.HttpServletRequest;
import
javax.servlet.http.HttpServletResponse;
/*
* @Author   : Som Prakash Rai
* @Join      : Java Training Center
* @visit      : www.jtcindia.org
*@Call   :+91-9990399111
* */

@WebServlet(name="testServlet",urlPatterns="
/test.jtc")
public class TestServlet extends HttpServlet
{

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

		//verifying the html action
		String uri=req.getRequestURI();
	Writer out=res.getWriter();
	res.setContentType("text/html");
	out.write("<h1>Request processed with
action:"+uri);
	}

}
```

```
		<welcome-file>index.html</welcome-
file>
	</welcome-file-list>
	<servlet>
		<servlet-
name>myTestServlet</servlet-name>
		<servlet-class></servlet-class>
	</servlet>

	<servlet-mapping>
		<servlet-
name>myTestServlet</servlet-name>
		<url-pattern>/admin.jtc</url-
pattern>

	</servlet-mapping>
</web-app>
```

- The URL Pattern in<servlet-mapping> must be same as action value of HTML <form>.
- The <url-pattern> should start with/
- <servlet-name> is a logical name and must be same in <service> & <servlet-mapping>.

Working Actiion attribute of <form>

Case 1: <form action="register.jtc">
http://localhost:9999/Jtc3/register.jtc Request will be submitted to current App (Jtc3)

Case 2: <form action="register.jtc">
http://localhost:9999/register.jtc Request will be submitted to Root of TOMCAT (Jtc3)

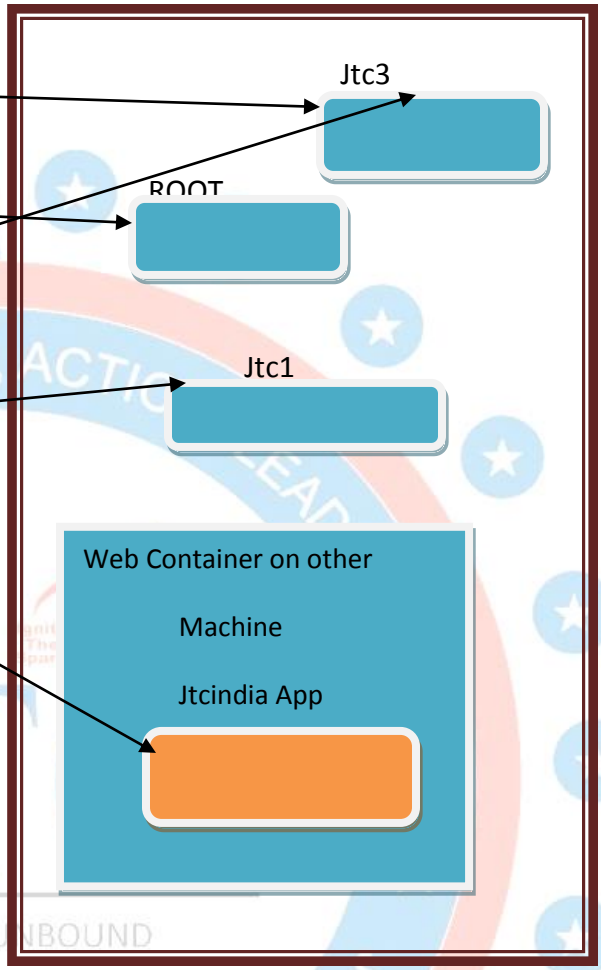
Case 3: <form action="/Jtc3/register.jtc">
http://localhost:9999/Jtc3/register.jtc Request will be submitted to (Jtc3)

Case 4: <form action="/Jtc1/register.jtc">
http://localhost:9999/register.jtc Request will be submitted to (Jtc1)

Case 5: <form action="http://www.jtcindia.org/register.jtc">
http://localhost:9999/register.jtc Request will be submitted to www.jtcindia.org

```
<form action="register.jtc">
        --------
        </form>
<form action="register.jtc">
        ---------
        </form>
<form action="/Jtc3/register.jtc">
        --------------
        </form>
<form action="/Jtc1/register.jtc">
        ------------------
        </form>
<form action="http://www.jtcindia.org/register.jtc">
        ---------
        </form>
```

```
<form action="register.jtc">
        --------
        </form>
<form action="register.jtc">
        ---------
        </form>
<form action="/Jtc3/register.jtc">
        --------------
        </form>
<form action="/Jtc1/register.jtc">
        ------------------
        </form>
<form action="http://www.jtcindia.org/register.jtc">
        ---------
        </form>
```



Servlet Instance Creation:

- By default the servlet instance will be created when first client will send the request to the servlet.
- Only one instance will be created for one servlet and will be used to peocess all the request by using multiple threads.
- If you want to create the instace while starting the server or container or container the you can use the follwing:
    - In web.xml

      <servlet>

      ....

      <load-on-startup>X</load-on-startup>

      </servlet>
    - In Annotations

      @WebServlet(....,loadOnStartup=x)

- Note: X will be int type value. It should not be-ve integer.
- It indicates in which order the servlet instance will be created.

Parameters:

- Parameters is name-value pair.
- Parameter name and value are of type string.
- Parameter is read-only i.e Web Container stires the parameter in the corresponding object and you can read & use that value.you can not modify the parameters.

There are 3 type of Parameters:

1. ServletRequest Parameters.
2. ServletConfig Parameters.
3. ServletContgext Parameters.

1. ServletRequest Parameters:

- Client Submitted data which is commimg from web client to web Server along with HttpRequest are called as request Parameters.
- Web Container collects client Submitted Data and Stores that in HttpServletRequest object as Request Parameters.
- As a Devloper, You can collect that data from request object as follows.

Case 1:

```
String fn=req.getParameter("fname");
        String ln=req.getParameter("lname");
```

Case 2:

```
Map<String,String[]> map=req.getParameterMap();
Set pname=map.keySet();
Iterator it=pname.iterator();
while(it.hasNext()){
        String pnm=(String)it.next();
        Object val=map.get(pnm);
        String[] values=(String[])val;
        System.out.println("\n pname"+pnm+"\n Values");
        for(int i=0;i<values.length;i++){
                System.out.println(values[i]);
        }
        }
```

Case 3:

To Access only request parameters

```
        Enumeration<String> ens=req.getParameterName();
List<String> list=Collections.list(ens);
for(String pn:list){
        String pv=req.getParameter(pn);
        System.out.println(pn+":"+pv);
}
```

Container is storing multiple value for sname key in map in the form of string arry.

```
        Map<String,String[]> map=…;
        String course[]=new String[2];
        Course[0]=”Module 1”;
        Course[1]=”Module 2”;
        Map.put{“course”,course);
```

2. ServletConfig Paramateres:

- ServletConfig is an interface available in Javax.servlet package and web Container vendor is responsible to provide the subclass for this interface.
- Every servlet will have its own ServletConfig object and can not be shared.
- When you want to use any data which is common for all users but specific to a particular servlet, that data can be specified as config Parameters or init Parameters.
- With Servlet 2.x: specify the config parameters in web.xml.

```
<servlet>
<servlet-name>helloServlet</servlet-name>
<servlet-class>com.jtcindia.servlets.HelloServlet</servlet-class>
<init-param>
```

```
<param-name>email</param-name>
<param-value></param-value>
</init-param>
            </servlet>
```

- With Servlet 3.0: Specify the config parameters in servlet class with annotaions.

```
@webServlet(name="helloServ",urlPatterns={"/hello.jtc},
            initParams={
                        @WebInitParam(name="email",value="hellosom@jtc"),
                        @WebInitParam(name="phone",value="9999"),
}
}
Public class HelloServlet extends HttpServlet{
}
```

- Web Container collects data from either web.xml or annotation and stores that in ServletConfig object as config Parameters.
- As a Devloper, you can collect that data from config object as follows.

```
String em=config.getinitParameters("email");
```

- You can use following inhertited method from HttpServlet

```
Public ServletConfig getServletConfig()
```

  - Here when HttpServlet class init() implementation will be invoked (from init() mwthod then config object will be returned otherwise null will be returned.

Assume that HttpServlet class is implemented as follows

```
Public abstract class HttpServlet{
Private ServletConfig config;
Public void init(ServletConfig config){
        This.config=config;
}
Public ServletConfig getServletConfig(){
        Return this.config;
        }
        ….
}
```

Case 1:

```
Class Abstract extends HttpServlet{
        Protected void service(…){
        ServletConfig cfg=getServletConfig();
                //return the config object.
                //Since init() method from HttpServlet will be called and config will be initialized
                }
                }
```

Case 2:

```
public void init(ServletConfig config) throws ServletException {
        super.init(config); //Invoking the HttpServlet init() method
}
public void service(ServletRequest arg0, ServletResponse arg1)
            throws ServletException, IOException {
        ServletConfig cfg=getServletConfig();
        //returns null value
        //Since init() method from your class will be called and you are not calling
HttpServlet init() so config won't be initialized
}
```

Case 3:

```
public void service(ServletRequest arg0, ServletResponse arg1)
            throws ServletException, IOException {
        ServletConfig cfg=getServletConfig();
        //returns null value
        //Since init() method from your class will be called and you are not calling
HttpServlet init() so config won't be initialized
}
```

3. **ServletContext Parameters:**

- **Ser**vletContext is an interface available in Javax.servlet package and container vendor is responsible to provide the subclass for this interface.
- One web Application will have only one ServletContext object i.e ServletContext object can be shared with all the servlets running in the Container.
- When you want to use any data which is common for all the users and common to all the Servlet then data can be specified as Context as context Parameters in the web.xml as follows.

```
<context-param>
<param-name>website</param-name>
<param-value>www.jtcindia.org</param-value>
</context-param>
```

- **Web container collects data from web.xml and stores that in ServletContext object as Context Parameters.**
- **As Devloper, You can collect that data from context object as follows:**

  String web=context.getInitParameter("websit");

- **You can use the following method with ServletConfig or ServletContext object to access the Corresponding parameter names.**

1. Index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>THIS IS INDEX HTML</h1>
<form action="hello.jtc">
<h2>Enter Name</h2>
<input type="text" name="fname">
<br/>
<input type="submit" value="Hello Test">
</form>
<form action="hai.jtc">
<h2>Enter Phone</h2>
<input type="text" name="phone">
<br/>
<input type="submit" value="Hai Test">
</form>
</body>
   </html>
```

2. Web.xml

```
  <display-name>Jtc2</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
  <context-param>
  <param-name>website</param-name>
  <param-value>www.jtcindia.org</param-value>
  </context-param>
  <servlet>
  <servlet-name>helloSevlet</servlet-name>
  <servlet-
class>com.jtcindia.servlet.HelloServlet</servlet-
class>
  <init-param>
  <param-name>email</param-name>
  <param-value>hellosom@jtc.com</param-value>
  </init-param>
  </servlet>
  <servlet-mapping>
  <servlet-name>helloservlet</servlet-name>
  <url-pattern>/hello.jtc</url-pattern>
  </servlet-mapping>
```

4. HelloServlet.java

```
package com.jtcindia.servlet;

import java.io.IOException;
import java.io.Writer;

import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import
javax.servlet.http.HttpServletRequest;
import
javax.servlet.http.HttpServletResponse;
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call   :+91-9990399111
* */

public class HelloServlet extends
HttpServlet {
      ServletConfig cfg=null;
      @Override
      public void init(ServletConfig
config) throws ServletException {
            this.cfg=cfg;
            System.out.println("init()
method of HelloServlet");

      }
      @Override
      protected void
service(HttpServletRequest req,
HttpServletResponse res)
                  throws
ServletException, IOException {
            System.out.println("servic()
method of HelloServlet");
            String
fname=req.getParameter("fname");
            String
phone=req.getParameter("phone");
            Writer out=res.getWriter();

      res.setContentType("text/html");
            out.write("<h1>Response from
HelloServlet");
            out.write("<hr/>Request
```

```
  <servlet>
  <servlet-name>haiservlet</servlet-name>
  <servlet-
class>com.jtcindia.servlet.HaiServlet</servlet-
class>
  <init-param>
  <param-name>email</param-name>
  <param-value>haisom@jtc.com</param-value>
  </init-param>
  </servlet>
  <servlet-mapping>
  <servlet-name>haiservlet</servlet-name>
  <url-pattern>/hai.jtc</url-pattern>
  </servlet-mapping>


   </web-app>
```

3. HaiServlet.java

```
package com.jtcindia.servlet;

import java.io.IOException;
import java.io.Writer;

import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/*
* @Author    : Som Prakash Rai
* @Join       : Java Training Center
* @visit        : www.jtcindia.org
*@Call    :+91-9990399111
* */

public class HaiServlet extends HttpServlet {
//          ServletConfig cfg=null;
//          @Override
//          public void init(ServletConfig config) throws
ServletException {
//                    this.cfg=cfg;
//                    System.out.println("init() method of
HelloServlet");
//
//          }
            @Override
            protected void service(HttpServletRequest req,
HttpServletResponse res)
                              throws ServletException,
Parameters");

      out.write("<br/>Fname:"+fname);

      out.write("<br/>Phone:"+phone);
            out.write("<hr/>Servlet Config
Parameters");
            String
eml=cfg.getInitParameter("email");
            out.write("<br/>"+cfg);
            out.write("<br/>Email:"+eml);
            out.write("<hr/>Servlet
Context Parameters");
            ServletContext
ctx=cfg.getServletContext();
            String
web=ctx.getInitParameter("website");
            out.write("<br/>"+ctx);
            out.write("<br/>Web:"+web);
      }

}
```

```
IOException {
                System.out.println("servic() method of HelloServlet");
                String fname=req.getParameter("fname");
                String phone=req.getParameter("phone");
                Writer out=res.getWriter();
                res.setContentType("text/html");
                out.write("<h1>Response from HelloServlet");
                out.write("<hr/>Request Parameters");
                out.write("<br/>Fname:"+fname);
                out.write("<br/>Phone:"+phone);
                out.write("<hr/>Servlet Config Parameters");
        ServletConfig cfg=getServletConfig();
        String eml=cfg.getInitParameter("email");
                out.write("<br/>"+cfg);
                out.write("<br/>Email:"+eml);
                out.write("<hr/>Servlet Context Parameters");
                ServletContext ctx=cfg.getServletContext();
                String web=ctx.getInitParameter("website");
                out.write("<br/>"+ctx);
                out.write("<br/>Web:"+web);
        }

}
```

Jtc7: Files Required in Servlet 3.x:

1. **index.html(same as Jtc6)**
2. **HelloServlet.java**
3. **HaiServlet.java**
4. **Web.xml**

HelloServlet.java	**HaiServlet.java**
package com.jtcindia.servlet; import java.io.IOException; import java.io.Writer;	package com.jtcindia.servlet; import java.io.IOException; import java.io.Writer;

```
import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebInitParam;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/*
* @Author    : Som Prakash Rai
* @Join        : Java Training Center
* @visit         : www.jtcindia.org
*@Call    :+91-9990399111
* */


@WebServlet(name="helloServlet",urlPattern
s={"/hello.jtc"},initParams={@
WebInitParam(name="email",
value="hellosom@jtc.com")})
public class HelloServlet extends HttpServlet {
	ServletConfig cfg=null;
	@Override
	public void init(ServletConfig config)
throws ServletException {
		this.cfg=cfg;
		System.out.println("init() method
of HelloServlet");

	}
	@Override
	protected void service(HttpServletRequest
req, HttpServletResponse res)
				throws ServletException,
IOException {
		System.out.println("servic()
method of HelloServlet");
		String
fname=req.getParameter("fname");
		String
phone=req.getParameter("phone");
		Writer out=res.getWriter();
		res.setContentType("text/html");
		out.write("<h1>Response from
HelloServlet");
		out.write("<hr/>Request
Parameters");
		out.write("<br/>Fname:"+fname);
		out.write("<br/>Phone:"+phone);
		out.write("<hr/>Servlet Config
Parameters");
		String
```

```
import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebInitParam;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/*
* @Author    : Som Prakash Rai
* @Join        : Java Training Center
* @visit         : www.jtcindia.org
*@Call    :+91-9990399111
* */

@WebServlet(name="haiServ", urlPatterns={"/hai.jtc"},
initParams={@WebInitParam(name="email",value="haisom@jtc.com")})
public class HaiServlet extends HttpServlet {

	@Override
	protected void service(HttpServletRequest req,
HttpServletResponse res)
				throws ServletException, IOException {
		System.out.println("servic() method of HelloServlet");
		String fname=req.getParameter("fname");
		String phone=req.getParameter("phone");
		Writer out=res.getWriter();
		res.setContentType("text/html");
		out.write("<h1>Response from HelloServlet");
		out.write("<hr/>Request Parameters");
		out.write("<br/>Fname:"+fname);
		out.write("<br/>Phone:"+phone);
		out.write("<hr/>Servlet Config Parameters");
	ServletConfig cfg=getServletConfig();
	String eml=cfg.getInitParameter("email");
		out.write("<br/>"+cfg);
		out.write("<br/>Email:"+eml);
		out.write("<hr/>Servlet Context Parameters");
		ServletContext ctx=cfg.getServletContext();
		String web=ctx.getInitParameter("website");
		out.write("<br/>"+ctx);
		out.write("<br/>Web:"+web);
	}

}

Web.xml
<web-app>
<context-param>
	<param-name>website</param-name>
	<param-value>www.jtcindia.org</param-value>
	</context-param>
```

eml=cfg.getInitParameter("email");
out.write("
"+cfg);
out.write("
Email:"+eml);
out.write(" Servlet Context Parameters");
ServletContext ctx=cfg.getServletContext();
String web=ctx.getInitParameter("website");
out.write("
"+ctx);
out.write("
Web:"+web);
}
 	
}
	</web-app>

Servlet Lifecycle:

We need to understand task happing at following:

- Task happing at container startup.
- Task happing at request processing time.
- Task happing at container shutdown.

Consider the following case:

- **I have devloped one web Application with the following:**
  - I have Developed one Web Application with the following requirement.
  - I have written 2 servlets with the name HelloServlet and HaiServlet.
  - I have written 1 listener with the name HelloListener
  - I have written 1 Filter with the name HelloFilter
  - I configured all the above components in the web.xml.
  - I also configured context parameters and config parameters for HelloServlet.
  - I specified <load-on-startup> for the HelloServlet

- **Following is the sample web.xml**

```
<web-app>
<context-param>
<param-name>state</param-name>
<param-value>UP</param-value>
</context-param>
<servlet>
<servlet-name>hello</servlet-name>
<servlet-class>com.jtcindia.HelloServlet</servlet-class>
```

```
<init-param>
<param-name>city</param-name>
<param-value>Noida</param-value>
</init-param>
<load-on-startup>1</load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>hello</servlet-name>
<url-pattern>/hello.jtc</url-pattern>
</servlet-mapping>
<servlet>
<servlet-name>hai</servlet-name>
<servlet-class>com.jtcindia.HaiServlet</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>hai</servlet-name>
<url-pattern>/hai.jtc</url-pattern>
</servlet-mapping>
</web-app>
```

- I did the following tasks. After Deploying the above said application on Server.
    1. Started the Server.
    2. Send the 1st request to HelloServlet.
    3. Send the 2nd request to HelloServlet.
    4. Send the 1st request to HaiServlet.
    5. Send the 2nd request to HaiServlet.
    6. Shutdown the Server.

Case-1) Started the Server.

1. Container reads information from web.xml and stores in the main memory using SAX parser. If any problem happened while reading the web.xml container will not be ready to receive the request from server.

2. Container creates ServletContextObject and initializes the context object with context parameters specified in the web.xml
3. Container creates the thread pool.
4. All the listener configured in the web.xml will be initialized (It is a Java class loading, creating instance, initialization and installation.)
5. All the filters configured in the web.xml will be initialized.
6. Container checks whether any servlet is configured with <load-on-startup> tag. If any one or more servlets found with <load-on-startup> tag then those servlets will be initialized by the container at container startup with the give priority by doing the following tasks:-
   A. Container loads the Servlet Class into main memory.
   B. Container creates the instance of servlet by invoking default constructor.
   C. Container creates servlet config object and initializes the config object with the config parameters specified in web.xml.
   D. Container associates the servlet context object with servlet config object.
   E. Container invokes the init() method by passing ServletConfig object as parameter to initialize the servlet instance with required resources.

Case-2) Send the 1st request to HelloServlet.

1. Container collects the url pattern of incoming request(/hello.jtc) and checks whether the corresponding servlet is initialized or not.
2. If the servlet is initialized then container picks one thread from the thread pool and handovers the remaining request processing.
3. Threads start request processing by doing the following tasks:
   A. ServletRequest object will be created and will be initialized with the data coming from the client along with HttpRequest.
   B. HttpServlet response object will be created and will be initialized with response stream.
   C. Service() method will be invoked by passing ServletRequest and ServletResponse object as parameters.
   D. Once service() method is executed then response stream will be flushed to client over the network and destroys the request and response object.
   E. Once the response is delivered to the client, thread will be returned to pool.

interface-> ServletRequest

interface -> HttpServletRequest

interface ->HttpServletRequestImpl

```
ServletRequest sreq= new HttpServletRequestImpl();
service(sreq);
```

Case 3) Send the 2nd request to HelloServlet.

Same as Case 2.

Case 4) Send the 1st request to HaiServlet.

1. Container collects the url-pattern of incoming and checks whether the corresponding servlet is initialized or not.
2. If the servlet is not initialized then container initializes that servlet by doing the following tasks:
   A. Container loads the Servlet Class into main memory.
   B. Container creates the instance of servlet by invoking default constructor.
   C. Container creates servlet config object and initializes the config object with the config parameters specified in web.xml.
   D. Container associates the servlet context object with servletconfig object.
   E. Container invokes the init() method by passing ServletConfig object as parameter to initialize the servlet instance with required resources.
3. If the servlet is already initialized then container picks one thread from the thread pool and handovers the remaining request processing.
4. Threads start request processing by doing the following tasks:
   A) ServletRequest object will be created and will be initialized with the data coming from the client along with HttpRequest.
   B) HttpServlet response object will be created and will be initialized with response stream.
   C) service() method will be invoked by passing ServletRequest and ServletResponse object as parameters.
   D) Once service() method is executed then response stream will be flushed to client over the network and destroys the request and response object.
   E) Once the response is delivered to the client, thread will be returned to pool.

Case-5) Send the 2nd request to HaiServlet.

Same as Case 2.

Case-6) Shutdown the Server.

At Container Shutdown Time, Following tasks will happen.

1. Thread pool will be destroyed.
2. All the servlets will be destroyed one by one. For destroying one servlet instance, container invokes the destroy() method on the servlet instance.
3. All the filters will be destroyed one by one. While destroying one filter instance, container invokes the destroy() method on the filter instance.
4. All the listeners will be destroyed.
5. If any session objects (private object for use) running in the container those also will be destroyed and finally container is down and is unable to process any user request.

RequestDispatcher

- It is an interface available in javax.servlet package.the subclass is implemented by container vendir.
- It has two method as follows:
  - Public void forward(ServletRequest req,ServletResponse res)
  - Public void include(ServletRequest req,ServletResponse res)

Forward() Method

- Forward() method is used to forward the request from:
  - Servlet to HTML
  - Servlet to jsp
  - Servlet to servlet
  - Jsp to html
  - Jsp to jsp
  - Jsp to servlet.
- You can place many forward() methods in one conditionally to forward the control to other resource but only one forward() will be executed.

Usage:

```
RequestDispatcher rd=null;
rd=req.getRequestDispatcher("test.html");
rd=req.getRequestDispatcher("test.jsp");
rd=req.getRequestDispatcher("testjtcl");
rd.forward(req,res);
```

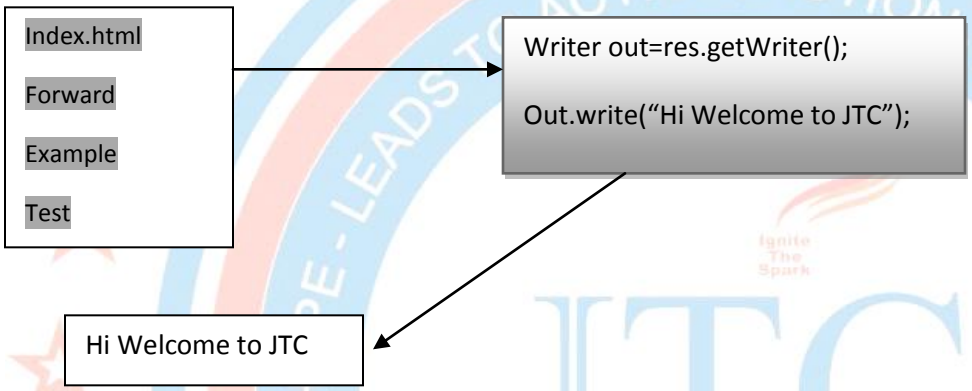
- When you try to invoke two forward method then java.lang.illegalStateException:

  Cannot forward after response has commited

- We can restrict the userfrom accessing the page directory by placing the page under WEB-INF drectory.
- When we have other statement after forward method then those statement will be executed but we will not be able to write any content in the response stream.

Jtc8: Files Required in Servlet 2.x:

1. Index.html
2. TestServlet.jav
3. Web.xml



Index.html

```
<body>
<h1>THIS IS INDEX HTML</h1>
<a href="test.jtc">Test</a>
<br/>
</body>
```

Web.xml

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

<servlet>
  <servlet-name>helloservlet</servlet-name>
  <servlet-
class>com.jtcindia.servlet.HelloServlet</servlet-
class>
<servlet-mapping>
  <servlet-name>helloservlet</servlet-name>
```

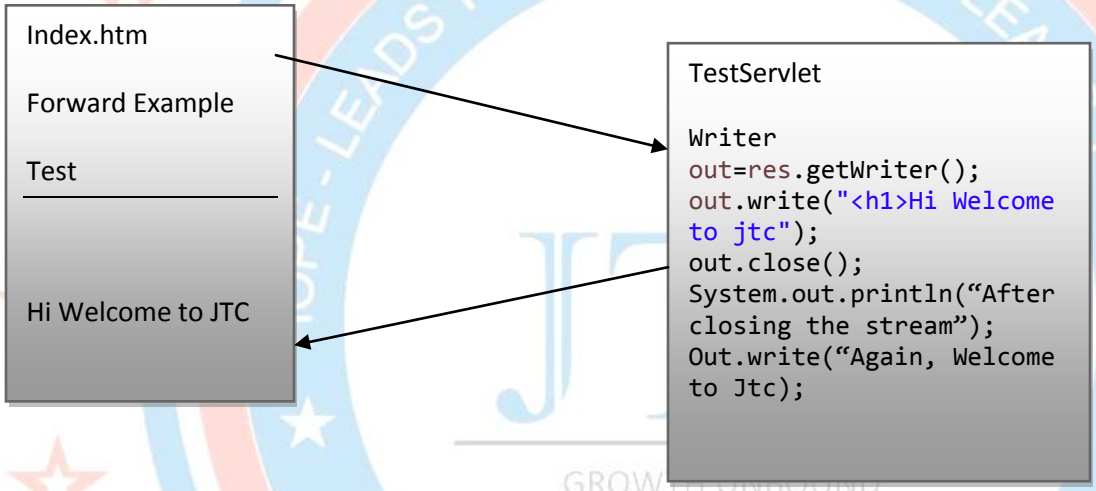
TestServlet.java

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

Writer out=res.getWriter();
out.write("<h1>Hi Welcome to jtc");
```

```
<url-pattern>/test.jtc</url-pattern>
</servlet-mapping>
```

Jtc9: Files Requred in Servlet 2.x:

1. Index.html
2. Web.xml
3. TestServlet.java



Index.html

```
<body>
<h1>THIS IS INDEX HTML</h1>
<a href="test.jtc">Test</a>
<br/>
</body>
```

Web.xml

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

TestServlet.java

```
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call    :+91-9990399111
* */

public class HelloServlet extends HttpServlet
{
        protected void
service(HttpServletRequest req,
HttpServletResponse res)
```

```
<servlet>
  <servlet-name>helloservlet</servlet-name>
  <servlet-
class>com.jtcindia.servlet.HelloServlet</servlet-
class>
<servlet-mapping>
  <servlet-name>helloservlet</servlet-name>
  <url-pattern>/test.jtc</url-pattern>
  </servlet-mapping>
```

```
                    throws ServletException,
IOException {

Writer out=res.getWriter();
out.write("<h1>Hi Welcome to jtc");
out.close();
System.out.println("After closing the
stream");
Out.write("Again, Welcome to Jtc);
System.out.println("*Service completed-Last
Statement");
```

Jtc10: files reqquired on Servlet 2.x:

1. Login.html
2. Home.html
3. LoginServlet.java
4. Required.html
5. Web.xml

1.Login.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="test">
<table align="center">
<tr><td colspan="2" align="center">
<h1>Account Login</h1></td></tr>
<tr><td><h2>Username</h2></td>
<td><input type="text" name="uname"/></td></tr>
<tr><td><h2>Password</h2></td>
<td><input type="text"
name="password"/></td></tr>
<tr><td colspan="2" align="center">
<input type="submit" value="Login"></td></tr>
</table>

</form>
```

2.home.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<table align="center">
<tr><td align="center">
<font color="green" size="5"
face="Cambria">Login failed!<br/>
Username orPassword is invalid
</font></td></tr>
</table>
</body>
</html>
```

4.LoginServlet.java

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

3.Required.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<table align="center">
<tr><td align="center">
<font color="green" size="5" face="Cambria">Login
failed!<br/>
Username orPassword is mandatory
</font></td></tr>
</table>
</body>
</html>
```

6.web.xml

```
<welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
  <servlet>
  <servlet-name>LoginServlet</servlet-name>
  <servlet-
class>com.jtcindia.servlet.LoginServlet</servlet-
class>
  </servlet>
  <servlet-mapping>
  <servlet-name>LoginServlet</servlet-name>
  <url-pattern>/test</url-pattern>

  </servlet-mapping>
</web-app>
```

```
package com.jtcindia.servlet;

import java.io.IOException;
import java.io.Writer;

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 LoginServlet extends HttpServlet{
@Override
protected void service(HttpServletRequest req,
HttpServletResponse res)throws ServletException,
IOException {
System.out.println("service of Testseletr");
String unm=req.getParameter("uname");
String pw=req.getParameter("password");
Writer out=res.getWriter();
out.write("<h1>Hi, Welcome to Jtc");
String page="";
boolean check=true;
if(unm==null ||unm.trim().isEmpty()){
page="required.html";
check=false;
RequestDispatcher
rd=req.getRequestDispatcher(page);
rd.forward(req, res);
}else if(pw==null || pw.trim().isEmpty()){
page="required.html";
check=false;
```

	RequestDispatcher
	rd=req.getRequestDispatcher(page);
	rd.forward(req, res);
	}
	if(check){
	if(unm.equals(pw)){
	page="home.html";
	RequestDispatcher
	rd=req.getRequestDispatcher(page);
	rd.forward(req, res);
	}else{
	page="home.html";
	RequestDispatcher
	rd=req.getRequestDispatcher(page);
	rd.forward(req, res);
	}
	}
	out.write("<h1>Again,Welcome to JTC");
	System.out.println("**service Completed-LAST-Statement");
	}
	}

Login.html

Account Login

Username som

Password ***

Login

Home.html

Login Success……

Username or Password is invalid

Error.html

Login Failed!

Username or Password is invalid

Required.html

Login Failed!

Username & Password are mndatory

LoginServlet

```
//Collection the data

String un=req.getParameter("uname");
String pw=req.getParameter("password");

// Valiadtion the data and decide the
page

String page=”….”;

……….

//forward the page contenet to client

RequestDispatcher rd=null;

rd=req.getRequestDispatcher(“page”);

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

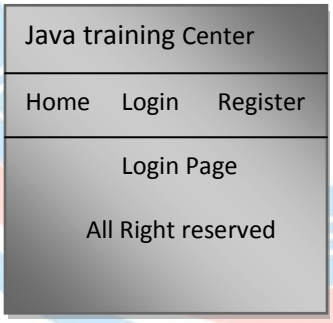
include() Method:

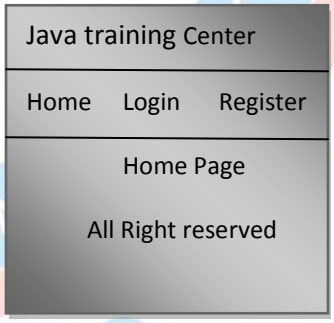
include() method is used to include the response of one servlet or jsp in another servlet or jsp and after evaluating the request control will return back.

We can have more than one include() statement in one servlet or jsp but we can have only one forward() method in one servlet or jsp. After using forward you cannot use request and response object processing statements.

Jtc11: Files Required in Servlet 2.X:

1) Index.html
2) Header.html
3) Footer.html
4) Menu.html
5) Home.html
6) Login.html
7) Register.html
8) ShowHomeServlet.java
9) ShowLoginServlet.java
10) ShowRegisterServlet.java
11) Web.xml





1.index.html	2.header.html
`<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01`	`<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01`

```
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<a href="showhome.jtc">GO TO HOME</a>
</body>
</html>
```

2.footer.html

```
</head>
<body>
<center>
<h2>Java Training Center</h2>
</center>
</body>
</html>
```

5.login.html

```
<body>
<center>
<h1>LOGIN PAGE</h1>

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

7.register.html

```
<body>
<center>
<h1>REGISTER PAGE</h1>

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

8.ShowHomeServlet

```
package com.jtcindia.servlet;

import java.io.IOException;

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

public class ShowHomeServlet extends
HttpServlet{
	@Override
	protected void
service(HttpServletRequest req,
```

```
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<center>
<h2>Java Training Center</h2>
</center>
</body>
</html>
```

4.home.html

```
</head>
<body>
<center>
<h1>HOME PAGE</h1>

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

6.menu.html

```
<body>
<center>
<font color="blue" size="3">
<a href="showhome.jtc">HOME</a>
</font>
<font color="blue" size="3">
<a href="showlogin.jtc">LOGIN</a>
</font>
<font color="blue" size="3">
<a href="showregister.jtc">REGISTER</a>
</font>
</center>
</body>
</html>
```

9.showLoginServlet.java

```
package com.jtcindia.servlet;

import java.io.IOException;

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

public class ShowLoginServlet extends HttpServlet{
	@Override
	protected void service(HttpServletRequest req,
```



```
HttpServletResponse res)
                    throws ServletException,
IOException {

       System.out.println("service()....Of
ShowHomeServlet started**");
              RequestDispatcher
rd1=req.getRequestDispatcher("header.html");
              rd1.forward(req, res);
              RequestDispatcher
rd2=req.getRequestDispatcher("menu.html");
                    rd2.forward(req, res);
              RequestDispatcher
rd3=req.getRequestDispatcher("home.html");
              rd3.forward(req, res);
              RequestDispatcher
rd4=req.getRequestDispatcher("footer.html");
              rd4.forward(req, res);
              System.out.println("service() of
ShowHomeServlet Completed");
       }

}
```

11.web.xml

```
  <display-name>Jtc11</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
  <servlet>
<servlet-name>ShowHomeServlet</servlet-name>
<servlet-
class>com.jtcindia.servlet.ShowHomeServlet
</servlet-class>
  </servlet>
  <servlet-mapping>
  <servlet-name>ShowHomeServlet</servlet-name>
  <url-pattern>/showhome.jtc</url-pattern>

  </servlet-mapping>
  <servlet>
  <servlet-name>ShowLoginServlet</servlet-
name>
  <servlet-
class>com.jtcindia.servlet.ShowLoginServlet
</servlet-class>
  </servlet>
  <servlet-mapping>
  <servlet-name>ShowLoginServlet</servlet-
name>
  <url-pattern>/showhome.jtc</url-pattern>
  </servlet-mapping>
  <servlet>
  <servlet-name>ShowRegisterServlet</servlet-
```

```
HttpServletResponse res)
                    throws ServletException,
IOException {
                    System.out.println("service()....Of
ShowLoginServlet started**");
                    RequestDispatcher
rd1=req.getRequestDispatcher("header.html");
                    rd1.forward(req, res);
                    RequestDispatcher
rd2=req.getRequestDispatcher("menu.html");
                    rd2.forward(req, res);
                    RequestDispatcher
rd3=req.getRequestDispatcher("home.html");
                    rd3.forward(req, res);
                    RequestDispatcher
rd4=req.getRequestDispatcher("footer.html");
                    rd4.forward(req, res);
                    System.out.println("service() of
ShowLoginServlet Completed");
          }

}
```

10..ShowRegisterServlet.java

```
package com.jtcindia.servlet;

import java.io.IOException;

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

public class ShowregisterServlet extends HttpServlet{
          @Override
          protected void service(HttpServletRequest req,
HttpServletResponse res)
                    throws ServletException,
IOException {
                    System.out.println("service()....Of
ShowRegisterServlet started**");
                    RequestDispatcher
rd1=req.getRequestDispatcher("header.html");
                    rd1.forward(req, res);
                    RequestDispatcher
rd2=req.getRequestDispatcher("menu.html");
                    rd2.forward(req, res);
                    RequestDispatcher
rd3=req.getRequestDispatcher("home.html");
                    rd3.forward(req, res);
```

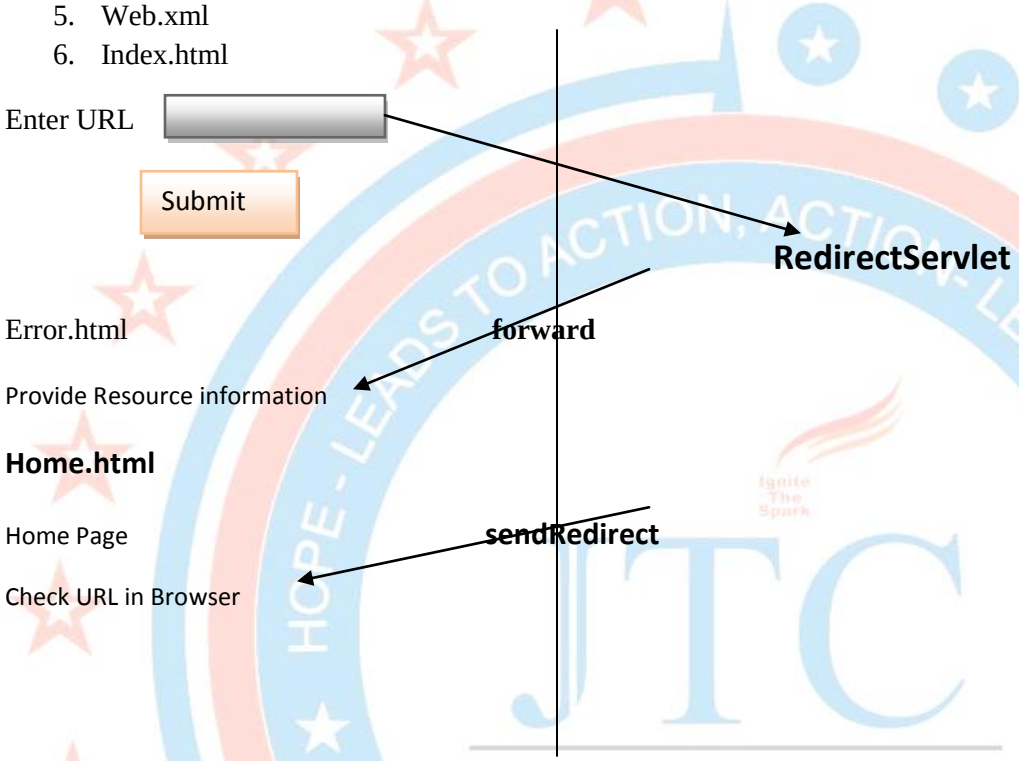
name>
  <servlet-class></servlet-class>
  </servlet>
  <servlet-mapping>
  <servlet-name>ShowRegisterServlet</servlet-name>
  <url-pattern>/showhome.jtc</url-pattern>
  </servlet-mapping>
</web-app>
	RequestDispatcher rd4=req.getRequestDispatcher("footer.html");
	rd4.forward(req, res);
	System.out.println("service() of ShowRegisterServlet Completed");
	}
	
	}

Send Redirect

Forward() of Request Dispatcher	SendRedirect() of ServletResponse
Using forward() method , you can send the control from one web componenet to another web componenet which are in same application.	Using senRequest() method, you can send the control from one web component of one the control from one web component of one application to another web component same or another web application.
While you are forwarding control using froward() method, you can send the data as an request attribute. In A: Req.setAttribut("AM","Som@jtc"); Rd.forward(req,res); In B: Object obj=req.getAttribute("EM")	While you are forwarding control using sendRedirect() method, you can send the data as an Query String. In A: url:"http://localhost:9999/Jtc1/login.jtc?Username=som&pasword=som; Res.sendRedirect(url); In B: String un=req.getParameter("uname");
Forwarding is happing completely at server side and client can not observe that.	Redirection is happing at server sid eand client side and client can observe the change in the url.

Jtc12: files Required in Servlet 2.x:

1. Index.html
2. Home.html
3. Error.html
4. RedirectServlet.java
5. Web.xml
6. Index.html



1.index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="redirect.jtc">
<h1>Enter URL</h1>
<br/>
<input type="text" name="page"/><br/><br/>
<input type="submit" value="Submit"/>

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

2.home.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h2>Redirect Example
<br/>Check the url in Browser</h2>
<hr/><center>
<h1>HOME PAGE</h1>
</center>
</body>
</html>
```


3.error.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h2>Redirect Example<br/>Check the URL in
Browser</h2>
<hr/>
<center>
<font color="red" size="5">
Provide the Url/Page name</font>
</center>
</body>
</html>
```

5.web.xml

```
  <display-name>Jtc12</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
  <servlet>
  <servlet-name>RedirectServlet</servlet-name>
  <servlet-
class>com.jtcindia.servlet.RedirectServlet</ser
vlet-class>
  </servlet>
  <servlet-mapping>
  <servlet-name>RedirectServlet</servlet-name>
  <url-pattern>/redirect.jtc</url-pattern>

  </servlet-mapping>
</web-app>
```

4.RedirectServlet.java

```
package com.jtcindia.servlet;

import java.io.IOException;

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

public class RedirectServlet extends
HttpServlet {
       @Override
protected void service(HttpServletRequest
req, HttpServletResponse res)
                     throws ServletException,
IOException {
              String
page=req.getParameter("page");
if(page==null || page.trim().length()==0){
              RequestDispatcher rd=null;
       rd=req.getRequestDispatcher("error.ht
ml");
                     rd.forward(req, res);
                     return;
              }
              if(page.startsWith("www")){
                     page="http://"+page;
       }
              res.sendRedirect(page);

       }

}
```

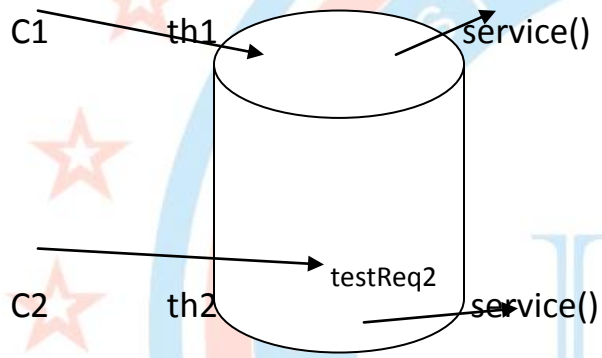
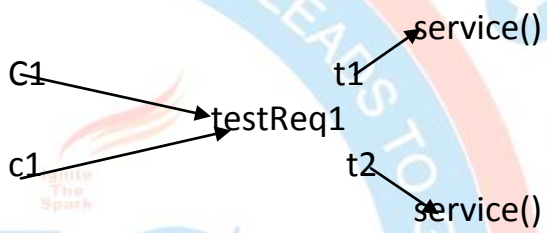
Servlet Thread Models

- Servlet implemented with 2 thread models:
  1. Single thread Model
  2. Multi Thread Model(*)

Single Thread Model	Multi Thread Model
With this model, container create new servlet instance for every incoming request:	With this model,container creates only one servlet instance per Servlet and the same will be used for

	w=every incoming request.

If you want to follwing single thread model for your servlet then your servlet class has to implemented javax.servlet.single ThreadModel marker interface.
 	
Class HelloServlet exgtends H.S imp singleThreadModel{
…….
}
	Every container uses Multi Thread Model as a default model.

It is thread safe	It is not thread safe
	

Note: Single ThreadModel interface is deprecated in J2EE 1.4

Jtc13: Files Required in Servlet 2.x:

1. Index.html
2. TestServlet.java
3. Web.xml

1.index.html	2.TestServlet
`<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01`	package com.jtcindia.servlet;

```
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<form action="test.jtc">
<h1>Enter Name</h1>
<br/><br/><input type="text" name="uname"/>
<br/><br/><input type="submit" value="Check">

</form>

</body>
</html>
```

2.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/java
ee http://java.sun.com/xml/ns/javaee/web-
app_3_0.xsd" id="WebApp_ID" version="3.0">
  <display-name>Jtc13</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
  <servlet>
  <servlet-name>TestServlet</servlet-name>
  <servlet-
class>com.jtcindia.servlet.TestServlet</servlet-
class>
  </servlet>
  <servlet-mapping>
  <servlet-name>TestServlet</servlet-name>
  <url-pattern>/test.jtc</url-pattern>
  </servlet-mapping>
</web-app>
```

```
import java.io.IOException;
import java.io.Writer;

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

public class TestServlet extends HttpServlet implements
SingleThreadModel {
        public TestServlet(){
System.out.println("**TestServlet Def Cons:"+this);
        }
@Override
protected void service(HttpServletRequest req,
HttpServletResponse res) throws ServletException,
IOException {
String unam=req.getParameter("uname");
Writer out=res.getWriter();
        out.flush();
Thread th=Thread.currentThread();
for(int i=0;i<10;i++){
System.out.println(th.getName()+"\t"+unam+"\t"+this);
        out.write("<br/>"+th.getName()+"\t"+unam+"\t"+thi
s);
                try{
                        Thread.sleep(1000);
                }catch(Exception e){
                        e.printStackTrace();
                }
out.write("<h1>Hi"+unam+"<br/>Response from Server");
                }
        }
}
```

Steps to create Servlet class in Eclipse

- Right click on SRC
- Select New-> Servlet
- Provide
    - Package
    - Class Name

- Change the super class name if required.
- Click on next button
- Provide the name (logincal name of Servlet).
- Add required init-parameters by clicking Add Button.
- Specify the url Mapping by clicking Add button
- Click on next button.
- Select the required method signature and click on finished.

Part=4

Exploring HttpServletRequest and HttpServletResponse

HttpServletrequest

- When you send the request using http protocol that request is called as HttpRequest.
- httpRequest containes two parts.
  - HttpServletHeaders.
  - httpRequest Body.
- When you send the request with Http method Get then first parameter data will be converted to Query String and that query String will be attched to the URL as follwos:

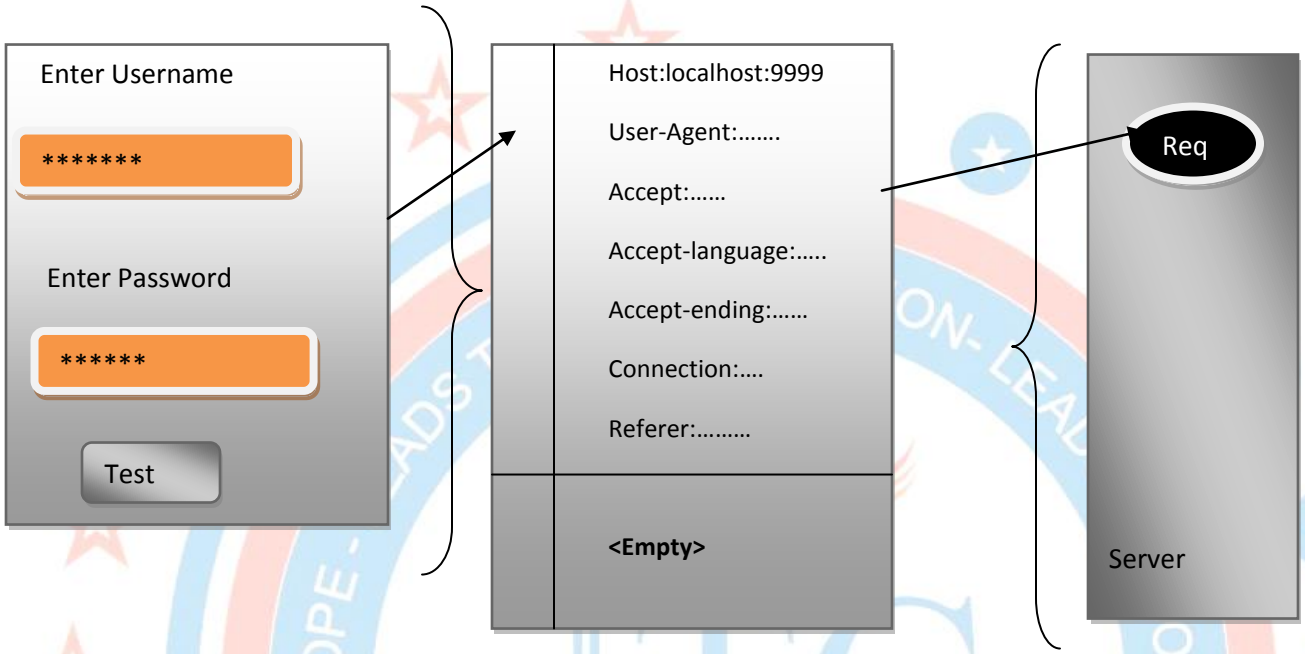
  http://loacalhost:9999/Jtc1/login.html

  http://loacalhost:9999/Jtc1/login.jtc?username=som&password=jtcindia

http://localhost:9999/Jtc1/login.jtc	URL
/Jtc1	CONTEXT PATH
http	PROTOCOL
Localhost	HOST
9999	PORT
/Jtc1/login.jtc	Url
Username=som&password=jtcindia	Query Strign

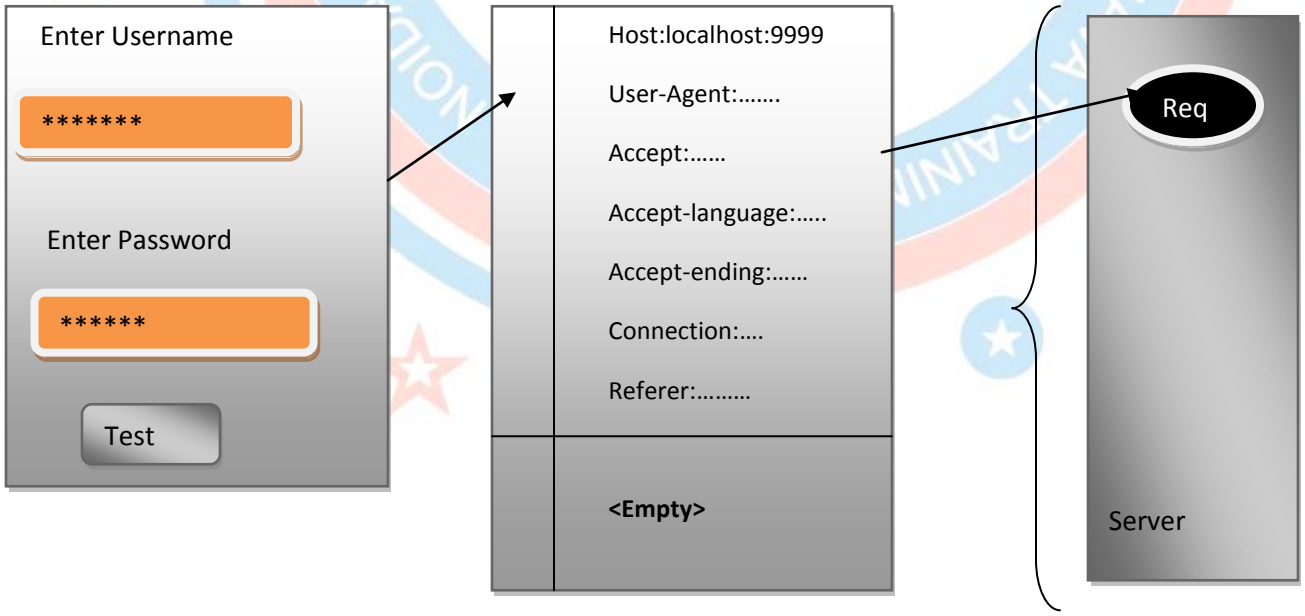
In the case Http method GET. This data will be placed in HttpRequest Headers and HttpRequest Body is empty.

http://loacalhost:9999/Jtc1/login.jtc?username=som&password=jtcindia



When you send the request with Http method POST then parameter data will be converted to query String and that Query String will be placed in Httprequest Body.Then URL is like.

http://loacalhost:9999/Jtc1/login.jtc?username=som&password=jtcindia



Get	Post
When you send the request with Http method GET. This data will be attached to URL as Query String.	When you send the request with Http method POST, this data will be placed in HttpRequest Body.
Using GET, you can send only limited amount of Data.	Using POST, you can unlimited amount of data.
GET is not secure becoz data will be visible in the URL	POST is more secure

- At Serverside, Container is reponsible for creating HttpServletRequest and HttpServletResponse objects and Initialized those objects with the data.
- HttpServletRequest containes the followings information
  - Request parameters
  - Request Headers
  - Request Cookies
  - Other Information.
- HttpServletResponse contains the Byte and Charecters.
  - Response Hgeaders
  - Response Stream interms of Byte and Chareter.

Jtc14: Files Required in Servlet 2.x:

1. **Index.html**
2. **TestServlet.java**
3. **Web.xm**

1.Index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<center>
<h1>Request & Response Example</h1>
</center>
<form action="test.jtc" method="post">
<h2>Enter Uname</h2>
<br/>
<input type="text" name="uname"/>
<h2>Enter Password</h2>
<input type="password"
```

2.TestServlet

```
package com.jtcindia.servlet;

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

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

public class TestServlet extends HttpServlet
{
	@Override
protected void service(HttpServletRequest
req, HttpServletResponse res)	throws
ServletException, IOException {
	//1. Request parameters
```

```
name="password"/><br/>
<input type="submit" value="Test"/>
</form>
</body>
</html>
```

3.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">
  <display-name>Jtc14</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
  <servlet>
  <servlet-name>TestServlet</servlet-name>
  <servlet-
class>com.jtcindia.servlet.TestServlet</ser
vlet-class>
  </servlet>
  <servlet-mapping>
  <servlet-name>TestServlet</servlet-name>
  <url-pattern>/test.jtc</url-pattern>

  </servlet-mapping>
</web-app>
```

```
        String un=req.getParameter("uname");
        String
pw=req.getParameter("password");
        //2. display request parameters
        PrintWriter out=res.getWriter();
        out.println("<h1>Username:"+un);
        out.println("<br/>Password:"+pw);
        out.println("<hr/>");
        out.println("Request Headers");
        //3.Request Headers
Enumration e=req.getHeaderNames();
while (e.hasMoreElements()) {
        String hn=e.nextElement().toString();
        String hv=req.getHeader(hn);
        out.println("<br/>"+hn+":"+hv);

              }
out.println("<hr/>");
out.println("Loacal info");
//4. Locals supported by Browser
out.println("<br/>req.getLocal():"+
req.getLocale());
out.println("<hr/>");
out.println("Other Info");
//5. Other information from
Requestout.println("<br/>Method:"+
req.getMethod());
out.println("<br/>RequestURI:"+
req.getRequestURI());
out.println("<br/>RequestURL:"+
req.getRequestURL());
out.println("<br/>Protocol:"+
req.getProtocol());
out.println("<br/>ContentLength:"+
req.getContentLength());
out.println("<br/>ContentType(:"+req.getConte
ntType());
out.println("<br/>RemoteAddr:"+req.getRemoteA
ddr());
out.println("<br/>RemotePort:"+req.getRemoteP
ort());
out.println("<br/>RemoteHost:"+req.getRemoteH
ost());
out.println("<br/>ServerPort:"+req.getServerP
ort());
out.println("<br/>ServerName:"+req.getServerN
ame());
out.println("<br/>QueryString:"+req.getQueryS
tring());
out.println("<br/>ServletPath:"+req.getServle
tPath());
out.println("<br/>ContextPath:"+req.getContex
tPath());
```



```
        }
}
```

Jtc15: Files Required in Servlet 2.x:

1. Index.html
2. TestServlet.java
3. Web.xml

1.index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>
<a href="test.jtc">SEND REQUEST</a>

</h1>
</body>
</html>
```

3.web.xml

```
<welcome-file-list>
 <welcome-file>index.html</welcome-file>
  </welcome-file-list>
  <servlet>
  <servlet-name>TestServlet</servlet-name>
  <servlet-
class>com.jtcindia.servlet.TestServlet
</servlet-class>
  </servlet>

  <servlet-mapping>
  <servlet-name>TestServlet</servlet-name>
  <url-pattern>/test.jtc</url-pattern>
  </servlet-mapping>
</web-app>
```

2.TestServlet.java

```
package com.jtcindia.servlet;

import java.io.IOException;
import java.io.Writer;
import java.util.Date;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class TestServlet extends HttpServlet{
static int count=0;
@Override
protected void service(HttpServletRequest req,
HttpServletResponse res)throws
ServletException, IOException {
        count++;
System.out.println("---service---:"+count);
Writer out=res.getWriter();
        Date dt=new Date();
        out.write("<h1>"+dt);
if(count<=10){
res.setHeader("Refresh","1");
        }else{
res.setHeader("Refresh",
"1:URL=http://jtcindia.com");
                }
        }

}
```

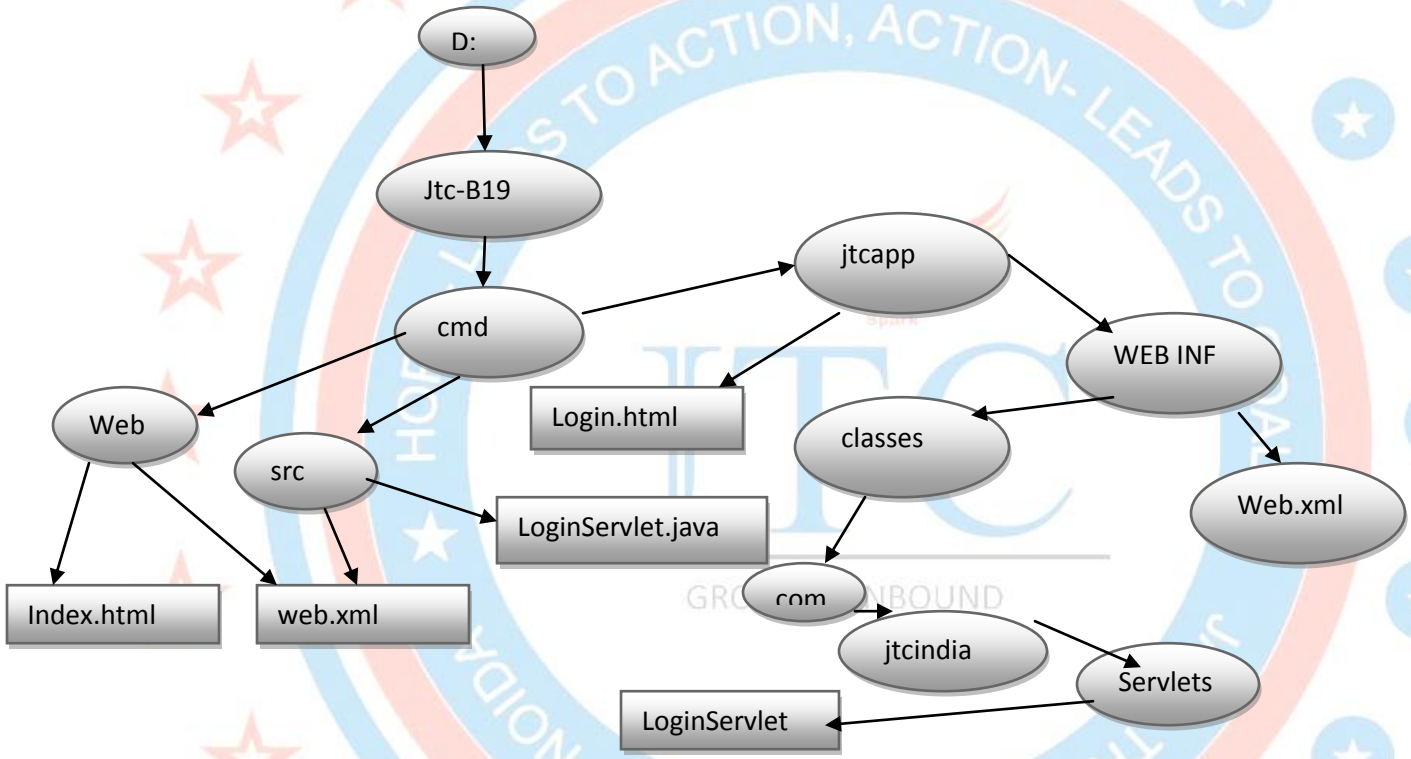
- If you want to use AutoRefres from Html. Only the you can use the following html code.

```
<html>
<head>
<meta http-equiv="Content-Type" content="5”;url=http://www.jtcindia.org">
<title>Insert title here</title>
```

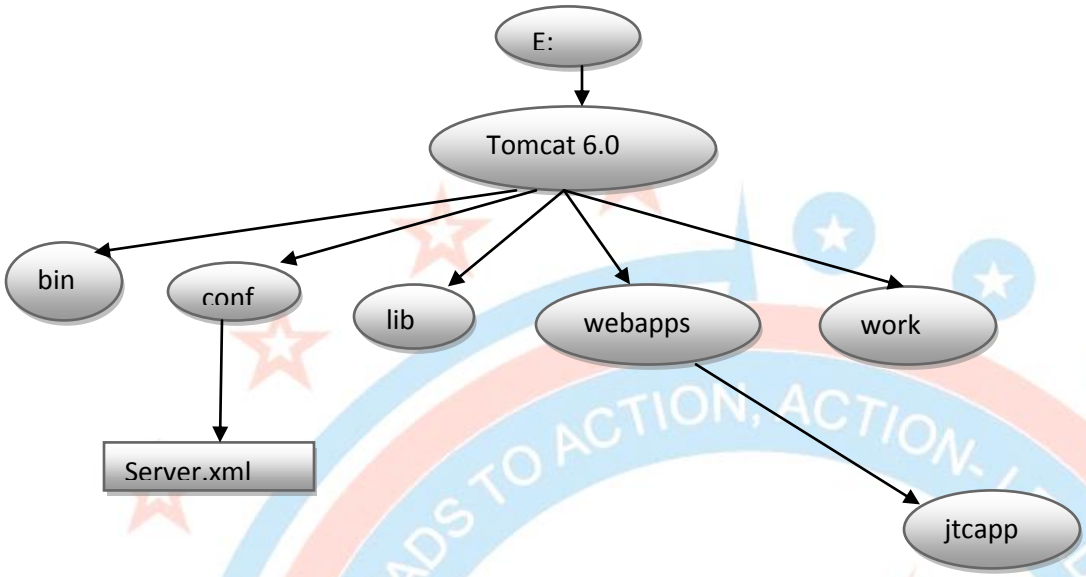
```
        </head>
        <body>
<h1>Redirecting to www.jtcindia.org</h1>
</body>
</html>
```

Working with Commannd Prompt:

- Create Devlopment Directory Structure



- Use any text editor to create the required file.
- You need to set the classpath for the servlet-api.jar to compile your servlet class
  - Servlet-api.jar file are is available in E:\Tomcat 6.0\lib
  - Set CLASSPATH=%CLASSPATH%;E:\Tomcate 6.0\lib\Servlet-api.jar;
- After compiling you will get the class files.
- Create Devlopment Directory Structure and copy the files in the required directory.



- Copy the application to E:Tomcat 6.0\\webapp directory
- Start the server from outside the eclips.
- Using web Browser test your application.
  http://localhost:9999/jtcapp/

introduction to JSP

- Jsp stands for java Server Page.
- Jsp is a combination of HTML tags and JAVA code.
- JSP will be processed by web Container by creating some Servlet class for it.
- Jsp will be used to design dynnamic response for the client.
- The extension for the file must be .jsp.
- You can access some predifined web componenet objects:
  - HttpServletRequeest request
  - HttpSession session
  - Writer out
  - ServletContext application etc.
- You need to use some scripting element to write the java code:
  1. To write the statement which are valid in any method
     ```
     <%
        //java Implementation valid inside a method
     %>
     ```
     Note: you must define;(semi colon) at last
  2. To write the data in the response stream
     ```
     <%=<varName>%>
     ```

Note: you should't define;(semi colon) at last

3. To import the package in jsp

<%@page import="java.util.*"%>
<%@page import="javau.io.*"%>

Jtc16: Files Required in Servlet 2.x:

1. Index.jsp

```
<%@page import="java.util.Date"%>
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-
8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>THIS IS INDEX JSP</h1>
<%
for(int i=0;i<5;i++){
      System.out.println(i);
      out.write("<br/>"+i);
}

%>
<br/>
Current Time
<%
Date dt=new Date();
out.write("<br/>"+dt);
String str="JTC";
%>
<br/>
<font color="red" size="5"><%=dt %></font>
<br/>
<font color="red" size="5"><%=str %></font>
</body>
</html>
```

Jtc 17: Files Required in Servlet 2.x:

1. Index.jsp
2. Show.jsp

Index.jsp

Store the Data in Ser
vletContext object

```
<%
application.setAttribute("MSG","
Java Training Center");
List<String> emails=new
ArrayList<String>();
emails.add("som@jtc.com");
emails.add("Rahul@jtc.com");
emails.add("Neha@jtc.com");
emails.add("Rahul@jtc.com");
application.setAttribute("EMAILS
",emails);
%>
```

Show.jsp
Get the data from ServletContext
object and display to user

```
<%=application.getAttribute("MSG")
%>
<br/>
<%
Object
obj=application.getAttribute("EMAIL
S");
List<String>
values=(List<String>)obj;
for(String eml:values){

%>
<font color="red"><br/><%=eml
%></font>
<%
}
%>
```

1.index.jsp

```
<%@page import="java.util.ArrayList"%>
<%@page import="java.util.List"%>
<%@ page language="java"
contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>THIS IS INDEX JSP</h1>
```

2.show.jsp

```
<%@page import="java.util.List"%>
<%@ page language="java"
contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>THIS IS SHOW JSP</h1>
<%=application.getAttribute("MSG") %>
<br/>
<%
```

```
<%
String str="jtc";
application.setAttribute("MSG",str);
application.setAttribute("MSG","Java Training Center");
List<String> emails=new ArrayList<String>();
emails.add("som@jtc.com");
emails.add("Rahul@jtc.com");
emails.add("Neha@jtc.com");
emails.add("Rahul@jtc.com");
application.setAttribute("EMAILS",emails);

%>
<a href="show.jsp">Show Data</a>
</body>
</html>
```

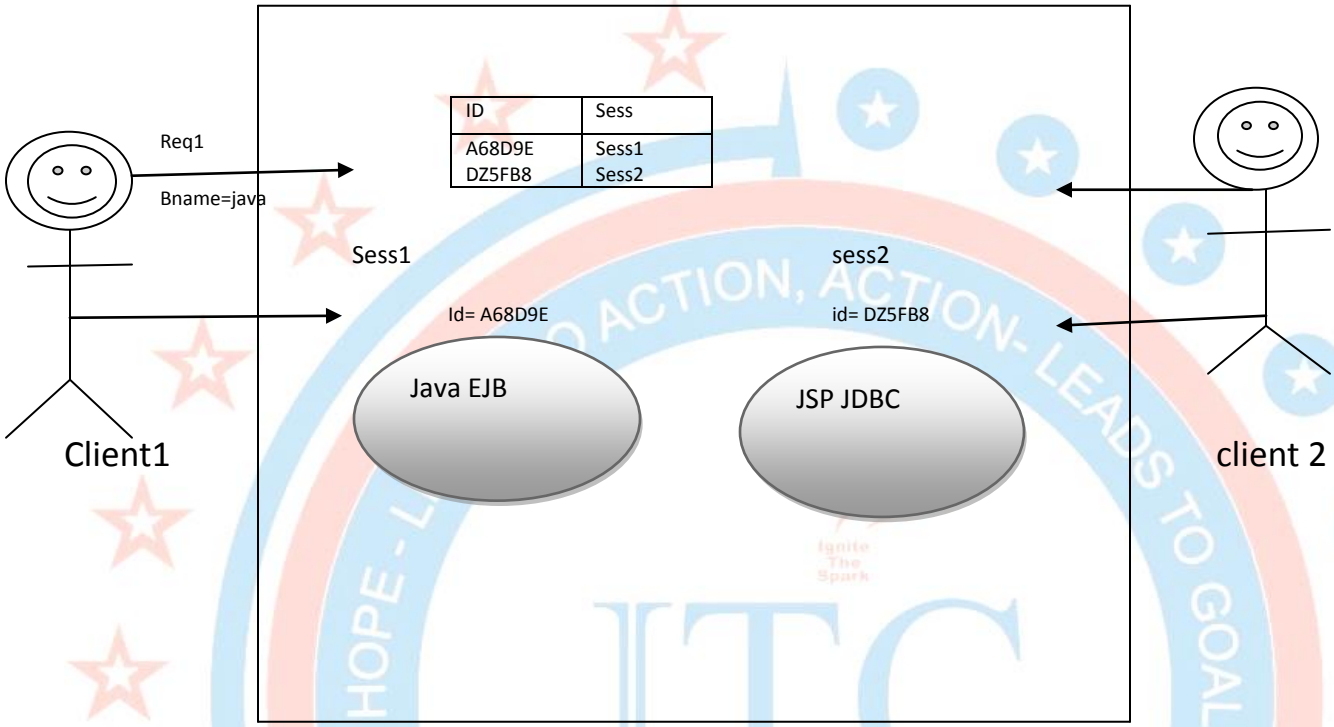
```
Object obj=application.getAttribute("EMAILS");
List<String> values=(List<String>)obj;
for(String eml:values){

%>
<font color="red"><br/><%=eml %></font>
<%
}
%>
</body>
</html>
```

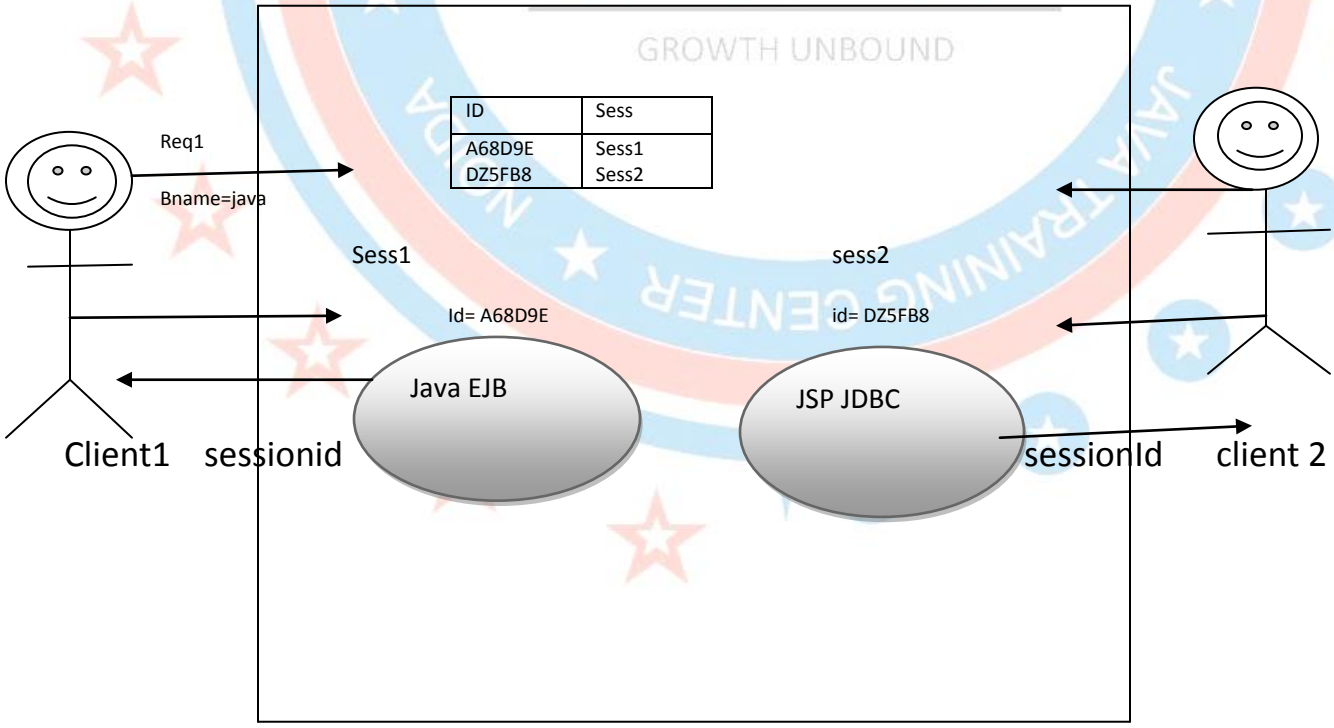
Part 5….Session Managmenet

Session Management

- HTTP protocol and web Server are stateless i.e for web server every request is a new request to process annd they can't identify whether it is cming from same client or new client.
- Somtime in web applications, we need to identify the client and process the request as per the client.
- Ex:
- In a shopping cart application we should know
  - Who is sending the request we should know
  - In which card the item has to addes
  - Who is sending place order request so that it can charge the amount to correct client.
- Session is a period of time where user sends multiple request and receive multiple response.
- In multiple request user may send data that should be stores annd managed in the server.
- Session will be used to stored client specific data. Session is a cinversional state between cclient and server.
- With in the User session you have to do 2 thingd
  - Identify the client.
  - Managed the conversational state (client specific DATA).
- To identify the client as new or ild,contaniner uses session ID.
- To store client Conversational state or data,you have to use HttpSession object.

Using session to manage client specific data.


ID
Sess
A68D9E
Sess1
DZ5FB8
Sess2
Req1
Bname=java
Sess1
sess2
Id= A68D9E
id= DZ5FB8
Java EJB
JSP JDBC
Client1
client 2


ID
Sess
A68D9E
Sess1
DZ5FB8
Sess2
Req1
Bname=java
Sess1
sess2
Id= A68D9E
id= DZ5FB8
Java EJB
JSP JDBC
Client1
sessionid
sessionId
client 2

You can use the following method with HttpServletRequest object to access the HttpSession object.

```
HttpSession sess=request.getSession();
HttpSession sess=request.getSession(boolean);
```

Class HttpServletRequestImpl Implements HttpServletRequest{

public HttpSession getSession(){

- Check whether the incomming request contains the cookies with same JESSIONID or not.
- If the incomming request contains the cookies with name JESSIONID then following task will be performed.
  - collects the value of cookie which is session Id.
  - Picks the session object related to this sessionId.
  - Returns this existing Session objects.
- If the incomming request does not contain the cookie with name JESSIONID yhen following task will be performed:
  - Create session object.
  - Generates unique session id
  - Stores the session id in session object.
  - Create the cookie with the name JESSIONID and with the value client session Id.
  - Adds the cookie to response object.
  - Returns this New Session object.
  - }
  - }

Q) What exactly getSession() method is doing?

Ans:

Check whether session object is available for this user or not.

If session object is available then returns that object.

If session object is not available then creates the new session object and returns that object.

Q) What exactly getSession(true) method is doing?

Ans:

Same as getSession()

Q) What exactly getSession(false) method is doing?

Ans:

Checks whether session object is available for this user or not.

If session object is available then returns that objects.

If session object is not available then returns null.

Sessio management Techniques:

- There are four session management techniques:
  - HttpSession
  - Cookies
  - Url-Rewriting
  -
  - Hidden Fields
- You can use the following to sore the client conversational data
  - HttpSession(***)
  - Cookies
- You can use following to carry the session ID
  - Cookies(***)
  - Url_rewriting(***)
  - Hidden Fields

Jtc 18:Files Requred in Servlet 2.X:

1. Index.jsp
2. Showcart.jsp
3. Showbooks.jsp
4. Web.xml
5. servletBookServlet.java
6. AddToCartServlet.java
7. SearchBookServlet.java
8. showCartServlet.java
9. RemoveFromCart.java

Index.jsp

Select Category

Java

SearchBook

showbooks.jsp

Java ADD TO CART

EJB ADD TO CART

JSP ADD TO CART

SHOW CART

Showcart.js

EJB REM Frem Cart

JSP REM Frem Cart

Place Order

Add To Cart

Your Order has placed

Successfully

searchBooksServlet.java

- Collect the category
- Search the book related to the category
- If not found the store error message to request
- If found store the list of books to session
- Foeward the request to showbooks.jsp

AddToCartServlet.java

- Get the session object for client
- Check session is existing session or no
- If existing session is not found the store error message to request
- If existing session found then collects the bookname
- Store the book name to session object
- Forward the request showbooks.jsp

- Validate the existing session
- If existing session is not found the store error message to request
- If existing session found then
  - Access the selected books name from session
  - Store the selected books name to request
  - Forward the request to showcart.jsp

- Validate the existing session
- If existing session is not found the store error message to request
- If existing session found then collect the bookname
- Remove the book name to session object
- Forward the request to showCartServlet.java

1.index.jsp

```
<%@ page language="java"
contentType="text/html; charset=ISO-
8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd"
>
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-
1">
<title>Insert title here</title>
</head>
<body>
<center>
<h1>JTC Bookstore</h1>
<h2>Book Search</h2>
<form action="searchbooks.jtc"
method="post">
<table>
<tr>
<td><h2>Select Category</h2></td></tr>
<tr><td><select name="category">
<option value="java">java</option>
<option
value="Testing">Testing</option>
<option value=".NET">.Net</option>
<option value="SAP">SAP</option>
</select> </td></tr>
<tr><td><input type="submit"
value="SearchBooks"/>
</td></tr>
</table>
</form>
</center>

</body>
</html>
```

3.showcart.jsp

```
<%@page import="java.util.*"%>
<%@ page language="java"
contentType="text/html; charset=ISO-
8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD
HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd"
>
<html>
<head>
```

2.showbooks.jsp

```
<%@page import="java.util.*"%>
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<center>
<h1>JTC Bookstore</h1>
<h2>Book Search</h2>
<font color="green" size="6" >${ADDED }</font>
</center><br/>
<%Object obj=request.getAttribute("MSG");
if(obj!=null){
%>

<br/><center>
<font color="red" size="6">
<%=obj %>
</font>
<br/><a href="index.jsp">GO TO SEARCH PAGE</a>
</center>
<%
}else{
       obj=session.getAttribute("BOOKS");

       ArrayList<String>
blist=(ArrayList<String>)obj;
       for(String bnm:blist){

%>
<form action="addtocart.jtc" method="post">
<input type="hidden" name="name" value="<%=bnm%>"/>
<font size="5"><%=bnm %>
<input type="submit" value="ADD TO CART"/>
</font>
</form>
<%
}
%>
<br/>
<form action="showcart.jtc">
<input type="submit" value="SHOW CART">
</form>
<%
```

```
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-
1">
<title>Insert title here</title>
</head>
<body>
<center>
<h1>JTC BookStore</h1>
<h2>Book Search</h2>
</center>
<%Object
object=request.getAttribute("MSG");
if(object!=null){
%>
<br/>
<center>
<font color="red"
size="6"><%=object%></font>
</center>
<%
}else{
	object=request.getAttribute("CAR
T");
	ArrayList<String>
blist=(ArrayList<String>)object;
	for(String bnm:blist){
%>
<form action="removeformcart.jtc"
method="post">
<input type="hidden" name="bname"
value="<%=bnm%>"/>
<font size="5"><%=bnm%><input
type="submit" value="Remove From
Cart"/></font>
</form>
<%
	}
%>
<br/>
<center>
<a href="Placeorder.jsp">PLACE
ORDER</a></center>
<%
}
%>
<center>
<br/><a href="showbooks.jsp">ADD TO
CART</a>
</center>
</body>
</html>
```

6.SearchBookServlet.java

```
package com.jtcindia.servlets;
}
%>

</body>
</html>
```

4.placeorder.jsp

```
<%@page import="java.util.*"%>
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<center>
<h1>Bookstore</h1>
<h2>Book Search</h2>
<h1>your Order has placed successfully</h1>
<%
session.invalidate();

%>
<br/>
<a href="index.jsp">GO TO SEARCH PAGE</a>
</center>
</body>
</html>
```

5.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">
  <display-name>Jtc18</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <servlet-name>SearhBooks</servlet-name>
    <servlet-
class>com.jtcindia.servlets.SearchBooksServlet</servl
et-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>SearhBooks</servlet-name>
    <url-pattern>/searchbooks.jtc</url-pattern>
  </servlet-mapping>
```

```
import java.io.IOException;
import java.util.ArrayList;

import javax.servlet.RequestDispatcher;
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 SearchBooksServlet extends HttpServlet
{

protected void service(HttpServletRequest req,
HttpServletResponse res)
throws ServletException, IOException {
String cat=req.getParameter("category");
if(cat!=null && cat.equals("java")){
ArrayList<String> blist=new ArrayList<String>();
        blist.add("java");
        blist.add("Servlets");
        blist.add("EJB");
        blist.add("JDBC");
        blist.add("JSP");
        blist.add("RMI");
        HttpSession sess=req.getSession();
        sess.setAttribute("BOOKS",blist);
        }else{
req.setAttribute("MSG","No books found with
category"+cat);
        }
RequestDispatcher
rd=req.getRequestDispatcher("showbooks.jsp");
rd.forward(req, res);        }
}
```

6.AddToCartServlet.java

```
package com.jtcindia.servlets;

import java.io.IOException;

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

public class AddToCartServlet extends HttpServlet {

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

```
  <servlet>
  <servlet-name>addToCartServlet</servlet-name>
  <servlet-
class>com.jtcindia.servlets.AddToCartServlet</servlet
-class>
  </servlet>

  <servlet-mapping>
  <servlet-name>addToCartServlet</servlet-name>
  <url-pattern>/addtocart.jtc</url-pattern>
  </servlet-mapping>
  <servlet>
  <servlet-name>showCartServlet</servlet-name>
  <servlet-
class>com.jtcindia.servlets.ShowCartServlet</servlet-
class>
  </servlet>
  <servlet-mapping>
  <servlet-name>showCartServlet</servlet-name>
  <url-pattern>/showcart.jtc</url-pattern>
  </servlet-mapping>

  <servlet>
  <servlet-name>removeFormCart</servlet-name>
  <servlet-
class>com.jtcindia.servlets.RemoveFromCartServlet</se
rvlet-class>
  </servlet>
  <servlet-mapping>
  <servlet-name>removeFormCart</servlet-name>
  <url-pattern>/removeformcart.jtc</url-pattern>
  </servlet-mapping>
</web-app>
```

7.ShowCartServlet.java

```
package com.jtcindia.servlets;

import java.io.IOException;
import java.util.Collections;
import java.util.Enumeration;
import java.util.List;

import javax.servlet.RequestDispatcher;
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 ShowCartServlet extends HttpServlet {
protected void service(HttpServletRequest req, HttpServletResponse
```

```
//Accessing the existing session object
HttpSession sess=req.getSession(false);
//validating session is available or not
        if(sess==null){
req.setAttribute("MSG","Session is destroyed");
        }else{
        String bnm=req.getParameter("bname");
//Adding the client selected book to session
sess.setAttribute(bnm, bnm);
req.setAttribute("ADDED",bnm+"is added to cart");
                }
RequestDispatcher
rd=req.getRequestDispatcher("showbooks.jsp");
rd.forward(req, res);
        }
}
```

8.RemoveFromCart.java

```
package com.jtcindia.servlets;

import java.io.IOException;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
public class RemoveFromCartServlet extends
HttpServlet {
protected void service(HttpServletRequest request,
HttpServletResponse response) throws
ServletException, IOException {
HttpSession httpSession=request.getSession(false);
if(httpSession==null){
request.setAttribute("MSG","Session is destroyed");
}else{
String bnm=request.getParameter("bname");
//Removing the client selected book from session
httpSession.removeAttribute(bnm);
        }
RequestDispatcher
rd=request.getRequestDispatcher("ShowCart.jsp");
rd.forward(request, response);
}
}
res)throws ServletException, IOException {
HttpSession httpSession=req.getSession(false);
if(httpSession==null){
req.setAttribute("MSG","Session is destroyed");
RequestDispatcher rd=req.getRequestDispatcher("showbooks.jsp");
        rd.forward(req, res);
        }else{
Enumeration<String> enms=httpSession.getAttributeNames();
        List<String> selectlist=Collections.list(enms);
        selectlist.remove("BOOKS");
if(selectlist.size()==0){
        req.setAttribute("MSG","No Books Selected");
        }else{
        req.setAttribute("CART",selectlist);
        }
RequestDispatcher rd=req.getRequestDispatcher("ShowCart.jsp");
        rd.forward(req, res);
        }
        }
}
```

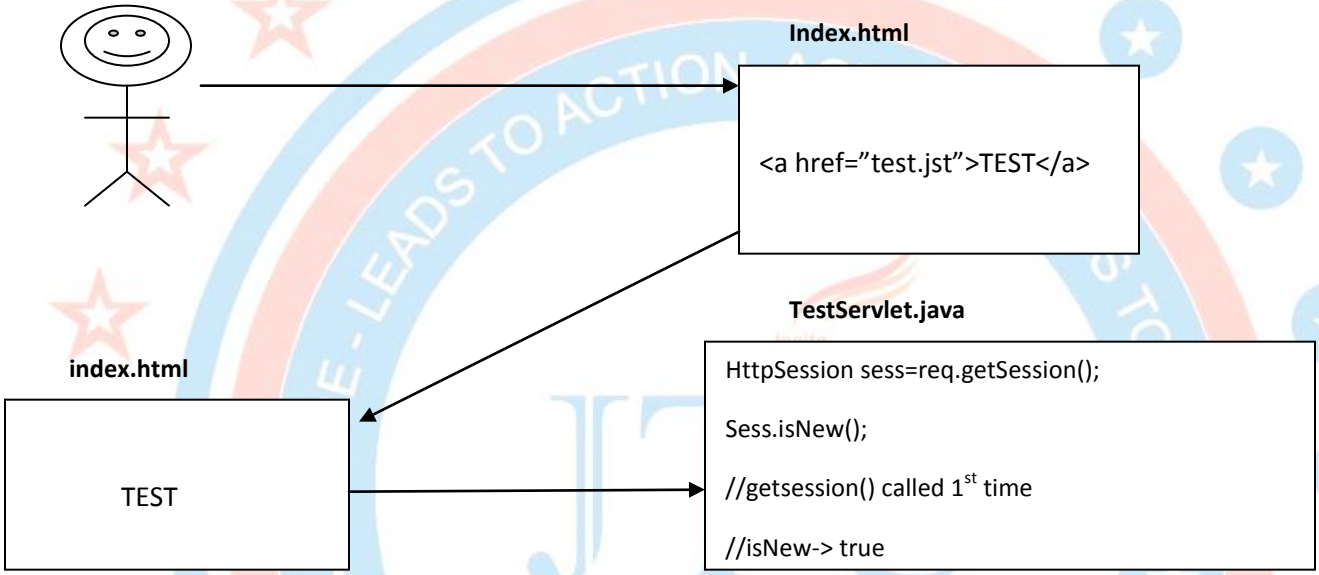
Exploring HttpSession:

- HttpSession is an interface available in javax.servlet.http package
- Subclass for HttpSession interface is implemented by container vendor.
- You can get the HttpSession object with the following methods of HttpServletRequest.
  - HttpSession getSession()
  - HttpSession getSession(boolean)
- You can store and access the client specific data in HttpSession object as an attribut with the following methods:
  - Void setAttribute(String aName,Object val)
  - Object getAttribute(String aName)
  - Void removeAttribute(String aName)
  - Enumeration getAttributeNames()
- You can also use the following methods to store the user data
  - Void putValue(String, Object)
  - Object getValue(String)
  - Void removeValue(String)
  - String[] getValueNames()
- Note: These 4 method are deprected.

Method	Description
Public String getId()	Return the session ID for the client
Public long getCreationTime()	Return the session creation time in milisecond
Public long getLastAccessedTime()	Return the time in milisecond when the session accessed last time
Public int getMaxInactiveInterval()	Return the time interval. If session will not be used within that interval then session will be destroyed.
Public ServletContext getServletContext()	Return ServletContext object.
Public void invalidate()	Destroyed the session immediately
Public boolean isNew()	Checks whether session object is newly created object or existing object
Public void setMaxInactiveInterval(int sec)	Set the session inactive interval.

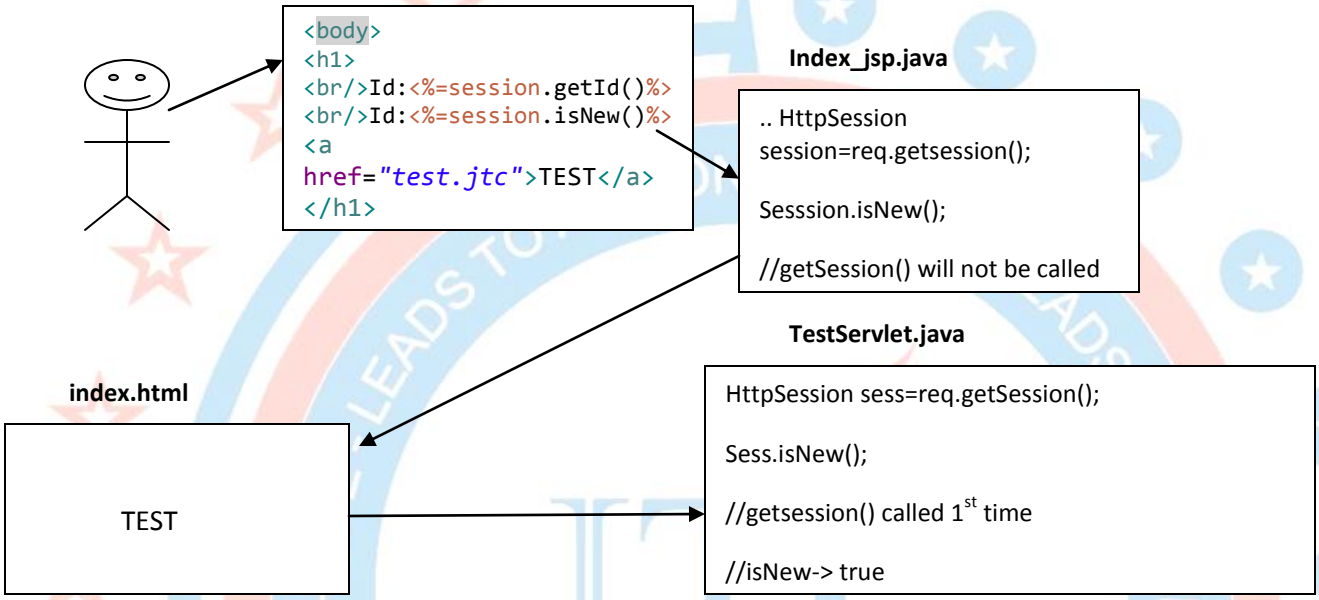
Jtc19: Files Required in Servlet 2.x:

1. Index.html
2. TestServlet.java
3. Web.xml



Jtc21:Files Required in Servlet 2:x:

1. Index.jsp
2. TestServlet.jsvs
3. Web.xml



1. Index.jsp

```
<%@page session="false" %>
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>
<br/>Id:<%=session.getId()%>
<br/>Id:<%=session.isNew()%>
<a href="test.jtc">TEST</a>
</h1>

</body>
</html>
```

2. Web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

3. **TestServlet.java**

```
package com.jtcindia.servlet;

import java.io.IOException;
import java.io.Writer;

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 TestServlet extends HttpServlet {
        @Override
        protected void
service(HttpServletRequest req,
HttpServletResponse res)
                        throws
ServletException, IOException {
                HttpSession
session=req.getSession();
                Writer out=res.getWriter();

        out.write("<h1><br/>ID:"+session.getId(
```

```
<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/java
ee http://java.sun.com/xml/ns/javaee/web-
app_3_0.xsd" id="WebApp_ID" version="3.0">
  <display-name>Jtc_20</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
  <servlet-name>TestServlet</servlet-name>
  <servlet-
class>com.jtcindia.servlet.TestServlet</servlet-
class>
  </servlet>
  <servlet-mapping>
  <servlet-name>TestServlet</servlet-name>
  <url-pattern>/test.jtc</url-pattern>
  </servlet-mapping>

</web-app>
```

```
));

        out.write("<br/>isNew:"+session.isNew(
));
        //true Session is created and defined
name identify or not
        }


}
```

- You can use following method with session object to destroyed the session object explicity
    - Sess.invalidate();
- You can use the following method with session object to destroyed the session object automatically when session object will not be used by client for the specified interval
    - Sess.setMaxInactiveInterval(int intervalInSecond)
- You can use the following in the web.xml filr to destroyed the session object automatically when session object will not be used by client for the specified interval

```
<session-config>
        <session-timeout>timeInMinut</session-timeout>
</session-config>
```

Servlet Scopes

- There are 3 scopes available in Servlet.
  - Request scope
  - Session scope
  - Context or Application scope
- You can store and manipulated data with the above scoped objects as an attribute with the following methods.
  - Void setAttribute(String, Objetc)
  - Void removeAttribute(String)
  - Object getAttribute(String)
  - Enumeration getAttributeNames()

Request Scope:

- When the data will be stored in the HttpServletRequest object then the scope will be request scope.
- The data from request scope can be access by that single user in that request only before sending the rquest to the client.

Session Scope:

- When the data will be stored in the HttpSession object then the scope will be session scope.
- The datat from session scope can be accessed by singl user accessed by single user across multiple requests (within the same session).

Context or Application Scope:

- When the datta will be stored in the servletContext object then the scope will be context scope.
- The dtaat from context or application scope can be accessed by multiple users accrossss multiple requests.

Exploring Cookies:

- Cookie is a class available in javax.servlet.http package.
- Cookie is a simple information with name and value.
- Name and value of the cookie will be of String type.
- Normally cookie's will be created at server machnic and will be persisted or stored at client machine.
- Cookies created at server machanic will come to client machanic along with HttpResPonse.
- Cookies persisted at client machanic will go to server machine along with HttpRequest.

Creating Cookie

Cookie ck=new Cookie("email",som@jtc.com);

Adding cookies to response

Response.addCookie(ck);

Accessing cookies from request

```
Cookie ck[]=request.getCookies();
For(cookie c:ck){
String cn=c.getName();
String cv=c.getValue();
System.out.println(cn+"
+cv);
}
```

Web Container Tasks:

By default web container will do the following regarding session management:

- Created one Special Cookie with
  - JSESSIONID as Cookie value.
  - Session as cookie value.
    - Cookie ck=new Cookie("JSESSIONID",ses.getId());
- Adds that cookie to response.
  - Response.addCookie(ck);
- Collects the special cookies and indentifies the session object based on session Id collected from special cookies.
  - Code…….

Assume that sessionMap is the Map object which containes key and value.

Key will be SESSIONID

Value will be SessionObject

- Use the following options to remove the Cookies stored in the client machine"
  - You need to specify the max age of cookies as 0.
  - You need to add the same cookie to the response.

```
Cookie c=……..;
c.setMaxAge(0);
res.addCookie(c);
code……..
String bnm=request.getParameter("bname");
Cookie ck[]=request.getCookies();
        for(Cookie c:cs){
                c.setMaxAge(0);
                res.addCookie(c);
        }
```

}

Jtc22: Files Required in Servlet 2.x:

1. Index.html
2. AddServlet.java
3. RemoveServlet.java

1.index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>Cookie Example</h1>
<form action="add.jtc" method="post">
<input type="text" name="bname"/><br/>
<input type="submit" value="ADD"/>
</form><hr/>

<form action="remove.jtc" method="post">
<h2>Enter Book Name</h2>
<input type="text" name="bname"/>
<br/><input type="submit" value="Remove"/>
</form>
</body>
</html>
```

3.RemoveServlet.java

```
package com.jtcindia.servlet;

import java.io.IOException;
import java.io.Writer;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class RemoveServlet extends HttpServlet {
@Override
protected void service(HttpServletRequest req, HttpServletResponse res) throws ServletException,
```

2.AddServlet.java

```
package com.jtcindia.servlet;

import java.io.IOException;
import java.io.Writer;

import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;

public class AddServlet extends HttpServlet {
protected void service(HttpServletRequest req, HttpServletResponse res)
        throws ServletException, IOException {
        Writer out=res.getWriter();
        String bnm=req.getParameter("bname");
        Cookie c1=new Cookie(bnm, null);
        res.addCookie(c1);
        out.write("<h1>"+bnm+" "+bnm);
        Cookie cs[]=req.getCookies();
        if(cs==null){
        out.write("<h2>You are new client");
        HttpSession sess=req.getSession();
        }else{
                boolean found=false;
                for(Cookie c:cs){
                String nm=c.getName();
                String val=c.getValue();
                out.write("<h2>"+nm+":"+val);
                if(nm.equals("JESSIONID")) found=true;
        }
        if(found){
        out.write("<h2>you are old client");
        HttpSession sess=req.getSession();
        }
        }
out.write("<hr/>");
RequestDispatcher rd=req.getRequestDispatcher("index.html");
rd.include(req, res);
        }
}
```

```
IOException {
        String bnm=req.getParameter("bname");
        Writer out=res.getWriter();
        Cookie cs[]=req.getCookies();
        if(cs==null){
        out.write("<h2>You are new client");
        HttpSession sess=req.getSession();
        }else{
        boolean found=false;
        for(Cookie c:cs){
        String nm=c.getName();
        String val=c.getValue();
        if(nm.equals("JESSIONID")){
        found=true;
        out.write("<h2>"+nm+":"+val);
        }else if(nm.equals(bnm)){
                c.setMaxAge(0);
                res.addCookie(c);
                }else{
                out.write("<h2>"+nm+":"+val);
                        }
                if(found){
                out.write("<h2>you are old
client");
                }else{
                out.write("<h2>you are New
client");
                HttpSession sess=req.getSession();
                }
                }
        out.write("<hr/>");
RequestDispatcher
rd=req.getRequestDispatcher("index.html");
rd.include(req, res);
        }
        }
}
```

4.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">
  <display-name>Jtc22</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
  <servlet>
  <servlet-name>AddServlet</servlet-name>
  <servlet-
class>com.jtcindia.servlet.AddServlet</servlet-class>

  </servlet>
  <servlet-mapping>
  <servlet-name>AddServlet</servlet-name>
  <url-pattern>/add.jtc</url-pattern>

  </servlet-mapping>
  <servlet>
  <servlet-name>RemoveServlet</servlet-name>
  <servlet-
class>com.jtcindia.servlet.RemoveServlet</servlet-
class>
  </servlet>

  <servlet-mapping>
  <servlet-name>RemoveServlet</servlet-name>
  <url-pattern>/remove.jtc</url-pattern>
  </servlet-mapping>
</web-app>
```

URL-Rewriting and Hidden Fields

- Container uses Session ID to identify the client as Old or New .
- Container sends the session ID to client machine as a cookie with name JSESSIONID.
- Sometimes, you may get the problems with cookies.
    - When your browser is not supporting the cookies.
    - When client deletes the cookies.
- When any proble is happends to cookies then request will not carry the cookie with name JSESSIONID.
- If request is coming without JSESSIONID cookie then container will treat that client as new and provides the new session object and new sessionid i.e client is loosing previous session object and conversational data available in that session object.

- As a Alternative, you can use URL's or hidden filed to carry the session ID from client to server and from server to client.

URL-Rwriting

- URL-Rewriting is the process of attaching the sessionid to the url. It is also called th encoding th URL.
- You can use the following method with response object to encode the URL:
    - String url=response.encoeURL("hello.jsp");

      url will be hello.jsp;jsessionid=A12jkd6d57 sdsdsd

- encodingURL() method takesthe URL as parameter and the encode url as follows:
- encodeURL() Deprecated

Hidden Fields:

- you caan store the session ID in the hidden fields as follows.
    - <input type="hidden" name="JSESSION" value="<%=session.getId()%>">

You can Devlop the web Application using the following:

- Servlet and jsp
- Struts1
- Struts 2
- Jsf
- Spring MVC

- When you Devlop the web Application using Servlets and Jsp then you are responsible for implemented URL Rewriting.
- When you Devlop the web Application using Web Frameworks then you ar not responsible for implementing URL Rewriting becase every web framework has the Built-in support for URL Rewriting.

Filters

- Filter is a web component like Servlet.
- Web Container is responsible for manging complete lifcycle or filter.
- Servlet is responsible for core Request processing.
- Before core request Processing by servlet i.e before calling the service() method, you may want to perform some tasks which is called as POST PROCCESSING tasks.
- Following are various PRE PROCCESSING tasks which youc an perform on incoming request before core request processing
    - Logging
    - Security check includs Authentication and Autnorization
    - Verification Session Validity etc.
- Following are various POST PROCESSING tasks whicch you can perform on out –going response after core request proccessing
    - Data compression
    - Data Encroption or Encoding
    - URL Rewritinh etd
- If you writ the code for PRE PROCESSING tasks and POST PROCESSING tasks accrosss all the servlets then vode gets duplicated and gives the maintance problem when you try to change that code.
- To avoid the code duplication problem and maintance problem, you need to write the PRE PROCESSING tasks and POST PROCESSING tasks code in acenterized place called filter.

Step to Devlop the Filter with Servlet 2.x:

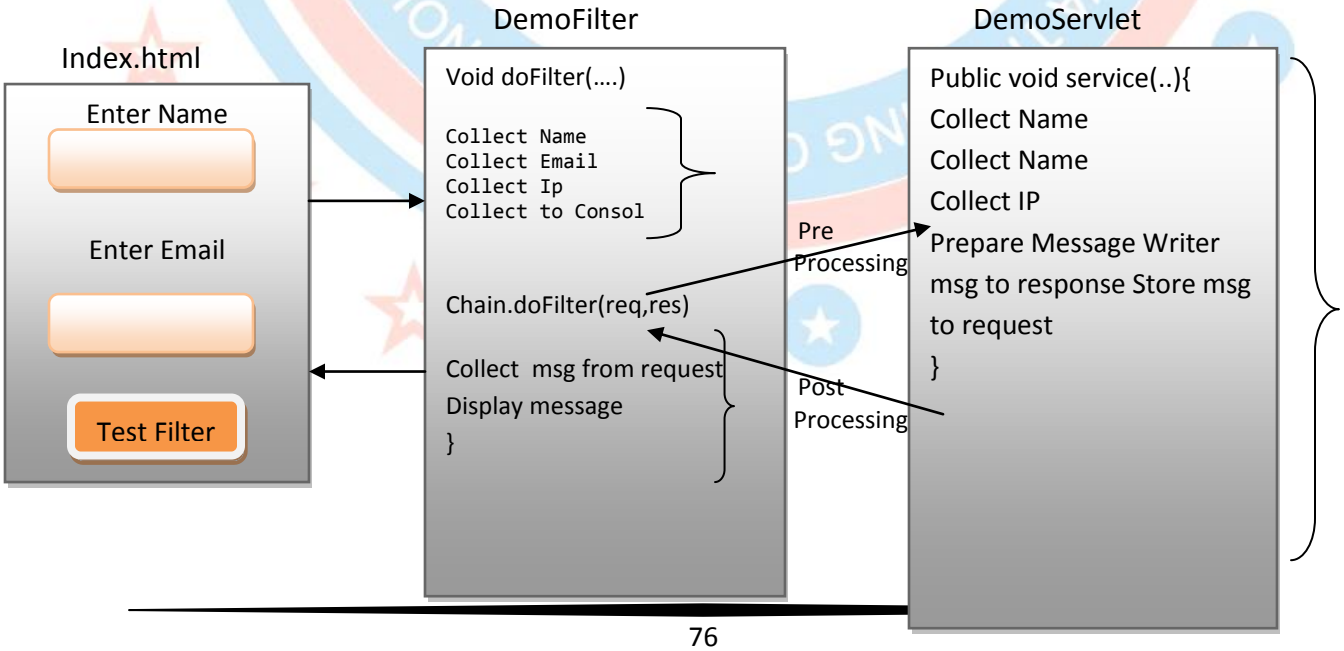
1. Write your own filter class by implementing javax.servvlet.Filter interface.
2. Your filter class has to override the following 3 lifecyle methods.
   a. Public void init(FilterConfig fc)
   b. Public void doFiletr(ServletRequest req,ServletResponse res,FilterChain fc)
   c. Public void destroy()
3. Configre the fileter the web.xml as follows

```
<filter>
<filter-name>demoFilter</filter-name>
<filter-class>com.jtcindia.servlet.DemoFilter</filter-class>
<init-param>
<param-name>city</param-name>
<param-value>Noida</param-value>
</init-param>
</filter>
<filter-mapping>
<filter-name>demoFilter</filter-name>
<url-pattern>/test</url-pattern>
</filter-mapping>
```

Step to Devlop the filter with Servlet 3.0

1. Write your own filter class by implemented javax.servlet.Filter interface.
2. Your filter class has to override the following 3 lifecyle methods.
   a. Public void init(FilterConfig fc)
   b. Public void doFiletr(ServletRequest req,ServletResponse res,FilterChain fc)
   c. Public void destroy()
3. Configure the filter information in the filter class only with the annotations as follows

```
@WebFilter(filterName="demoFilter",urlPatterns={"/demo.jtc"},
initParams={@WebInitParam(name="city",value="NOIDA")})

public class implements Filter{
//method implementation}
```



Jtc23:Files Required in Servlet 2.x:

1. index.html
2. DemoServlet.java
3. DemoFilter
4. Web.xml

1.index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>Java training Center</h1>
<h2>Filter Demo</h2>
<form action="test" method="post">
<table>
<tr>
<td>Name</td>
<td><input type="text" name="name"/></td>
</tr>
<tr>
<td>Email</td>
<td><input type="text" name="email"/></td>
</tr>
<tr>
<td colspan="2" align="center">
<input type="submit" value="Test Filter"/>
</td></tr>
</table>
</form>
</body>
</html>
```

4.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/jav
aee http://java.sun.com/xml/ns/javaee/web-
app_3_0.xsd" id="WebApp_ID" version="3.0">
  <display-name>Jtc23</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
```

2.DemoServlet.java

```
package com.jtcindia.servlet;

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

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

public class DemoServlet extends
HttpServlet {
	@Override
public void init(ServletConfig config)
throws ServletException {
	System.out.println("DemoServlet-
init()**");
String
ci=config.getInitParameter("city");
	System.out.println(ci);
	}
	@Override
protected void service(HttpServletRequest
req, HttpServletResponse res)
		throws ServletException,
IOException {
	System.out.println("DemoServlet
service()***");
	String
nm=req.getParameter("name");
	String
em=req.getParameter("email");
	String ip=req.getRemoteAddr();
	String msg="<h1>Hello!"+nm+"<br>";
 msg=msg+"you Email Id is"+em+"<br>";
msg=msg+"You sre sending the requesting
from IP Address:"+ip;
	PrintWriter out=res.getWriter();
	out.println(msg);
```

```
  </welcome-file-list>
  <servlet>
  <servlet-name>DemoServlet</servlet-name>
  <servlet-
class>com.jtcindia.servlet.DemoServlet</servlet-
class>
  <init-param>
  <param-name>city</param-name>
  <param-value>Delhi</param-value>
  </init-param>
  </servlet>
  <servlet-mapping>
  <servlet-name>DemoServlet</servlet-name>
  <url-pattern>/test</url-pattern>
  </servlet-mapping>
  <filter>
  <filter-name>demoFilter</filter-name>
  <filter-
class>com.jtcindia.servlet.DemoFilter</filter-
class>
  <init-param>
  <param-name>city</param-name>
  <param-value>Noida</param-value>
  </init-param>
  </filter>
  <filter-mapping>
  <filter-name>demoFilter</filter-name>
  <url-pattern>/test</url-pattern>

  </filter-mapping>
</web-app>
```

```
        }
public void destroy() {
System.out.println("***Destroyed()");
}
}
```

3.DemoFilter.java

```
package com.jtcindia.servlet;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class DemoFilter implements Filter
{
public void destroy() {
System.out.println("destroy()");
        }
public void doFilter(ServletRequest req,
ServletResponse res,FilterChain chain)
throws IOException, ServletException {
        String
nm=req.getParameter("name");
        String
em=req.getParameter("email");
        String ip=req.getRemoteAddr();
        System.out.println(nm);
        System.out.println(em);
        System.out.println(ip);
        chain.doFilter(req, res);
System.out.println("DemoFilter-
doFilter()-after");
        Object
obj=req.getAttribute("MSG");
        System.out.println(obj);
        String msg=obj.toString();
        System.out.println(msg);
        }
public void init(FilterConfig config)
throws ServletException {
System.out.println("**DemoFilter..Init()"
);
String
ci=config.getInitParameter("city");
System.out.println(ci);
        }
}
```


- All the Filter configred will be intialized by the container at container start-up.
- At the time of ontialized the filter at container Start-up, container will be the following tasks:
  - Filter class will be loaded.
  - Filter instance will be created by calling default consuctore.
  - Container will cast the instance into javax.servlet. Filter type to ensure it is filter.
  - Container creates the FilterConfig object and initializez FilterConfig object with the config parameters specified in web.xml or Annotations.
  - FilterConfig object will be initialzed with ServletContext object.
  - Container calls the init() method by passing FilterConfig object as parameter.
- At the time of destroying the filter at Container shutdown-up, container calls the destroy() method to release the resource initialized by init() method.
- If you want to invoke one filter before multiple servlets then you have to configure the filter with any one of the following ways:

```
<filter>
  <filter-name>filterB</filter-name>
  <filter-class>com.jtcindia.SessionValidationFilter</filter-class>
  </filter>
```

Option 1:

```
    <filter-mapping>
    <filter-name>filterC</filter-name>
    <url-pattern>/add.jtc</url-pattern>
                </filter-mapping>
    <filter-mapping>
          <filter-name>JtcSessionFilter</filter-name>
    <servlet-name>removeServlet</servlet-name>
</filter-mapping>
```

Option 2:

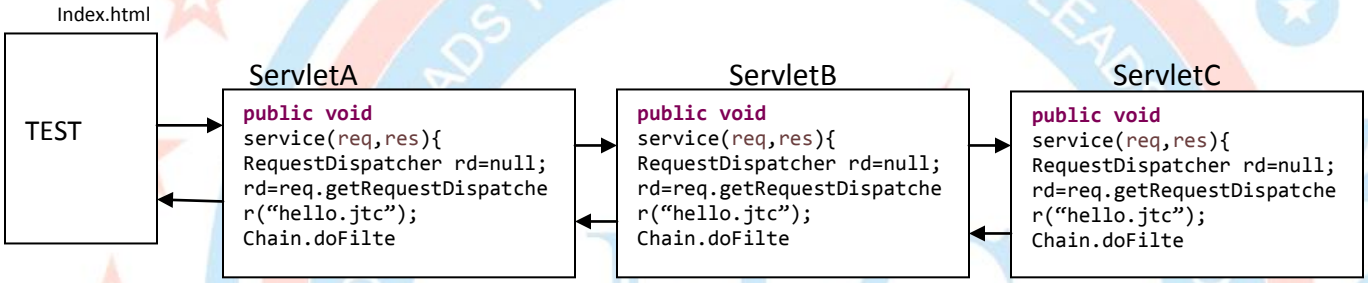
```
    <filter-mapping>
    <filter-name>filterC</filter-name>
    <url-pattern>/add.jtc</url-pattern>
          <url-pattern>/remove.jtc</url-pattern>
    </filter-mapping>
```

Option 3:

```
    <filter-mapping>
<filter-name>JtcSessionFilter</filter-name>
<url-pattern>*.jtc</url-pattern>
</filter-mapping>
```

Servlet Chaining:

- Invoking multiple Servlet one by one as chain is called as servlet Chaining.



Jtc25: Files Requred in Servlet 2.x:

1. Index.html
2. servletA. Java
3. ServletB.java
4. ServletC.java
5. Web.xml

1.index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>Servlet Chaining Example</h1>
<a href="test">TEST</a>
</body>
</html>
```

2.ServletA.java

```
package com.jtcindia.servlet;

import java.io.IOException;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import
javax.servlet.http.HttpServletRequest;
import
javax.servlet.http.HttpServletResponse;

public class ServletA extends
HttpServlet {
	@Override
	protected void
```

3.ServletB.java

```
package com.jtcindia.servlet;

import java.io.IOException;

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

public class ServletB extends HttpServlet {
	@Override
	protected void service(HttpServletRequest
req, HttpServletResponse res)
		throws ServletException, IOException {
System.out.println("ServletB service()
started....");
RequestDispatcher
rd=req.getRequestDispatcher("hello.jtc");
	rd.forward(req, res);
	System.out.println("ServletB service()
completed");
	}
}
```

5.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/java
ee http://java.sun.com/xml/ns/javaee/web-
app_3_0.xsd" id="WebApp_ID" version="3.0">
  <display-name>Jtc25</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
  </welcome-file-list>
  <servlet>
  <servlet-name>ServletA</servlet-name>
  <servlet-
class>com.jtcindia.servlet.ServletA</servlet-class>
  </servlet>
  <servlet-mapping>
  <servlet-name>ServletA</servlet-name>
  <url-pattern>/test</url-pattern>
 </servlet-mapping>
  <servlet>
  <servlet-name>ServletB</servlet-name>
  <servlet-
class>com.jtcindia.servlet.ServletB</servlet-class>
```

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

	System.out.println("ServletA
service() started....");
		RequestDispatcher
rd=req.getRequestDispatcher("hello.jtc")
;
		rd.forward(req, res);

	System.out.println("ServletA
service() completed");
	}

}
```

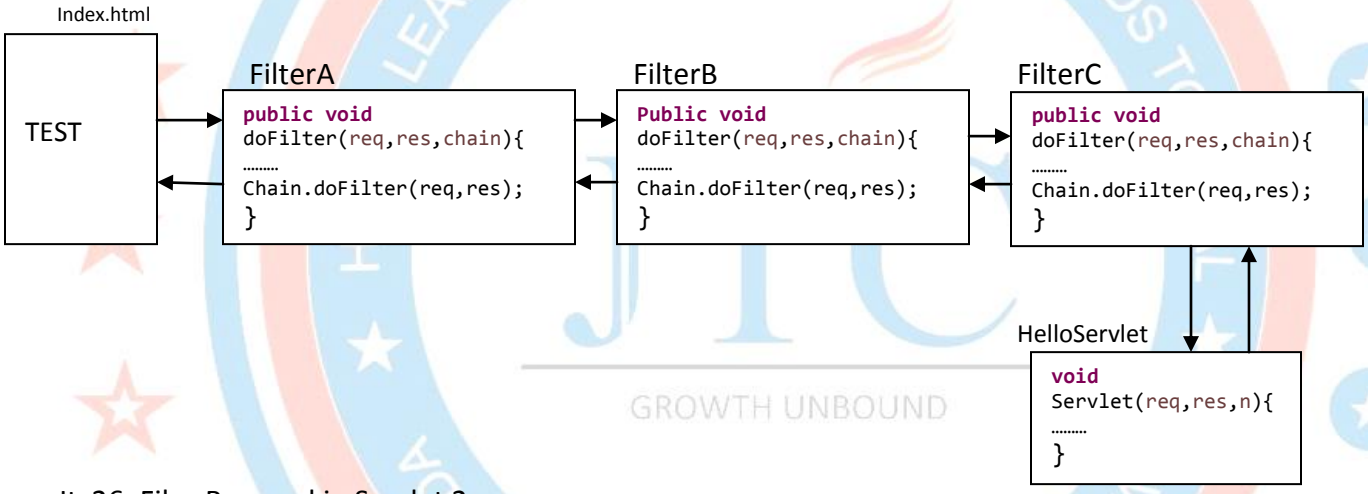
4.ServletC.java

```
package com.jtcindia.servlet;

import java.io.IOException;

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

public class ServletC extends
HttpServlet {
	@Override
	protected void
service(HttpServletRequest req,
HttpServletResponse res)
		throws
ServletException, IOException {
System.out.println("ServletC service()
started....");
		RequestDispatcher
rd=req.getRequestDispatcher("hello.jtc")
;
		rd.forward(req, res);
System.out.println("ServletC service()
completed");
	}
}
```

```
  </servlet>
  <servlet-mapping>
  <servlet-name>ServletB</servlet-name>
  <url-pattern>/test</url-pattern>
  </servlet-mapping>
  <servlet>
  <servlet-name>ServletC</servlet-name>
  <servlet-
class>com.jtcindia.servlet.ServletC</servlet-class>
  </servlet>
  <servlet-mapping>
  <servlet-name>ServletC</servlet-name>
  <url-pattern>/test</url-pattern>
  </servlet-mapping>
</web-app>
```

Filter Chaining

- Invoking multiple Filters one by one as chain is called as Filter Chaining..



Jtc26: Files Requred in Servlet 2.x:

1. Index.html
2. FilterA. Java
3. FilterB.java
4. FilterC.java
5. Web.xml

1.index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
```

2.FilterA.java

```
package com.jtcindia.servlet;

import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
```

```
<body>
<h1>Servlet Chaining Example</h1>
<a href="hello.jtc">TEST</a>
</body>
</html>
```

2.FilterB.java

```
package com.jtcindia.servlet;
import java.io.IOException;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
/*
* @Author   : Som Prakash Rai
* @Join     : Java Training Center
* @visit    : www.jtcindia.org
*@Call   :+91-9990399111
* */


public class FilterB implements Filter {
public void destroy() {
        System.out.println("destroy()");
        }
public void doFilter(ServletRequest req,
ServletResponse res,
        FilterChain chain) throws IOException,
ServletException {
System.out.println("FilterB
doFilter()..started....");
        chain.doFilter(req, res);
        System.out.println("FilterB
doFilter()..Completed");
        }
public void init(FilterConfig config) throws
ServletException {
        System.out.println("FilterB init()");
        }
}
```

5.HelloServlet.java

```
package com.jtcindia.servlet;

import java.io.IOException;
import java.io.Writer;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/*
```

```
import javax.servlet.ServletResponse;
/*
* @Author   : Som Prakash Rai
* @Join     : Java Training Center
* @visit    : www.jtcindia.org
*@Call   :+91-9990399111
* */

public class FilterA implements Filter {
public void destroy() {
        System.out.println("destroy()");
        }
public void doFilter(ServletRequest req,
ServletResponse res,
        FilterChain chain) throws IOException,
ServletException {
System.out.println("Filter a
doFilter()..started....");
        chain.doFilter(req, res);
        System.out.println("Filter a
doFilter()..Completed");
        }


public void init(FilterConfig config) throws
ServletException {
        System.out.println("Filter a init()");
        }
}
```

4.FilterC.java

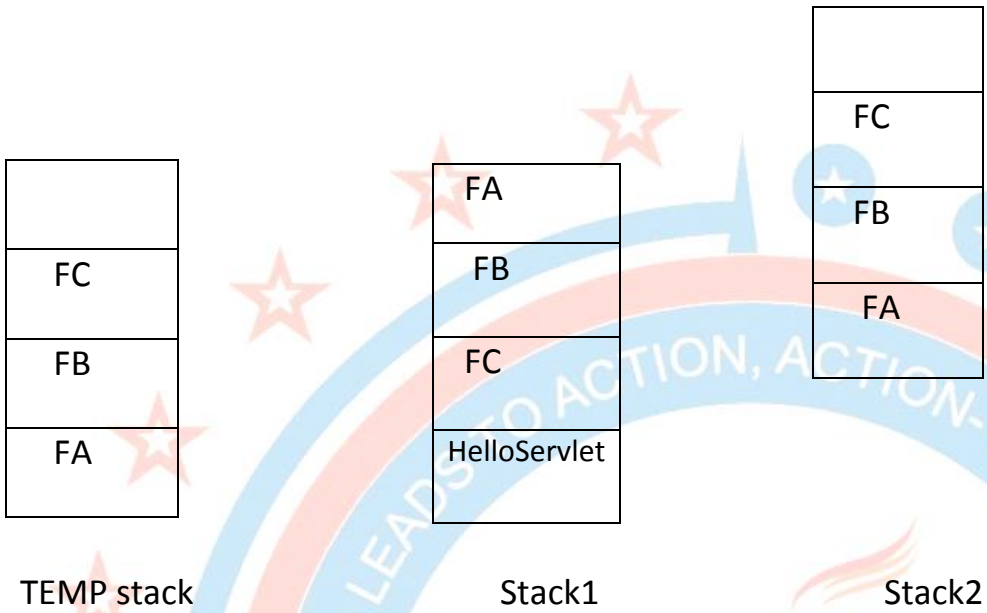
```
package com.jtcindia.servlet;
import java.io.IOException;

import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
/*
* @Author   : Som Prakash Rai
* @Join     : Java Training Center
* @visit    : www.jtcindia.org
*@Call   :+91-9990399111
* */

public class FilterC implements Filter {
public void destroy() {
        System.out.println("destroy()");
        }
public void doFilter(ServletRequest req,
ServletResponse res,
```



```
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call   :+91-9990399111
* */

public class HelloServlet extends HttpServlet
{
protected void service(HttpServletRequest req,
HttpServletResponse res)  throws
ServletException, IOException {
System.out.println("HelloServlet
service()...Started.....");
        Writer out=res.getWriter();
        out.write("<h1> Verify the server
console");
System.out.println("*HelloServlet class
service()..completed...");
        }
}
```

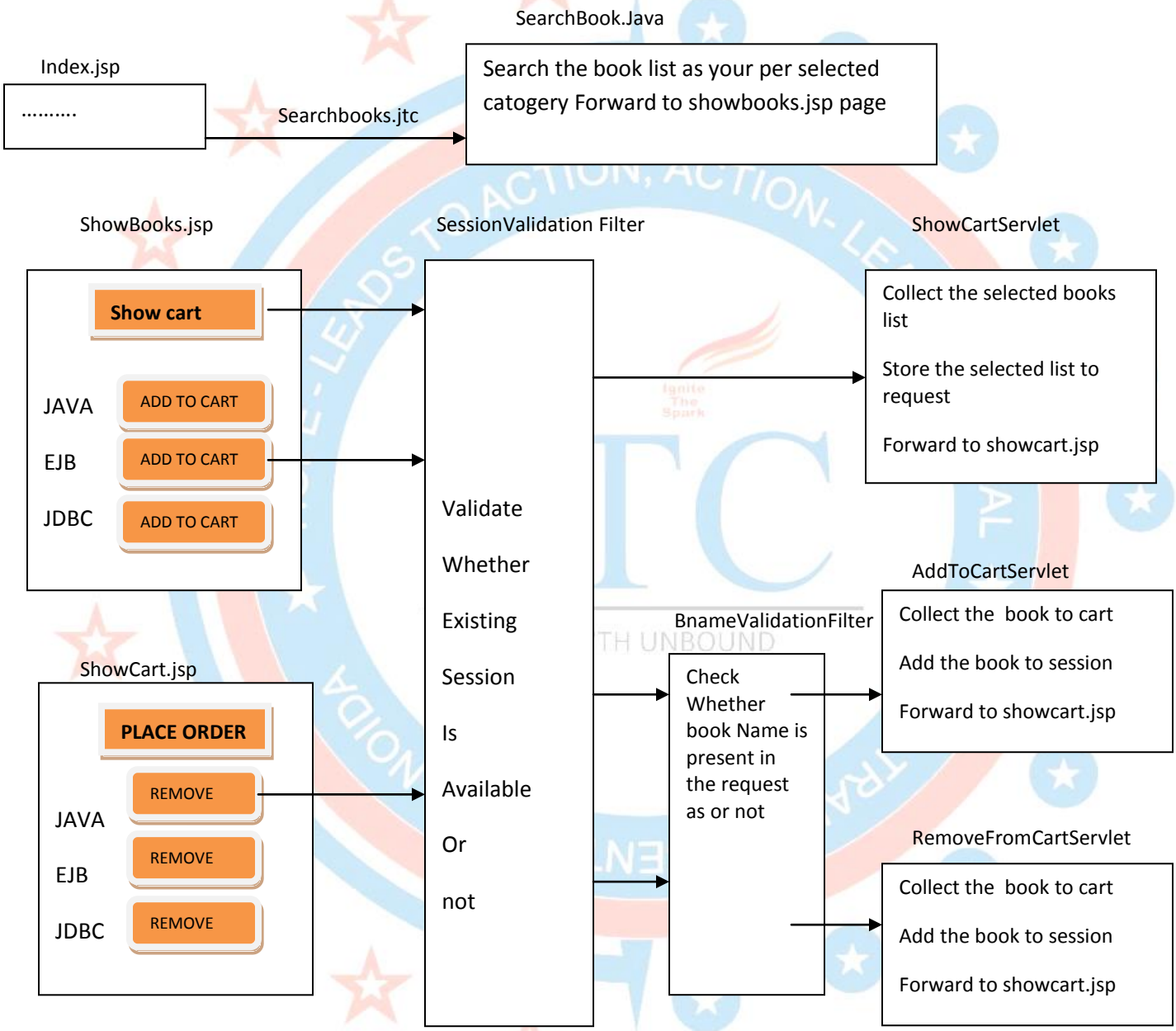
```
        FilterChain chain) throws IOException,
ServletException {
System.out.println("FilterC
doFilter()..started....");
        chain.doFilter(req, res);
        System.out.println("FilterC
doFilter()..Completed");
        }


public void init(FilterConfig config) throws
ServletException {
        System.out.println("FilterC init()");
        }
}
```

- When you send the request to any Servlet with some url-pattern then container will do the following tasks.
    - Creates the Stack1
    - Collects the incoming Request URL (hello.jtc)
    - Indentifies the Servlet which is matching with Incoming Request URL (HelloServlet).
    - Pushes the Servlet call into Stack1.
    - Creates the TEMP Stack.
    - Identifies one or more Filters which are matching with Incoming Request URL(FA,FB,FC).
    - Pushes the filters into TEMP stack.
    - Pops the filter from TEMP Stack and Pushes the Filter into Stack1.
    - Now starts the processing the current request by doing the following:
        1. Pops the web component from Stack1
        2. Check whether web componenet is Servlet or filter.
        3. If web component is filter then
            a. Pushes into Stack2.
            b. doFilter() method will be called.
            c. When chain.doFilter() is ecountered then repeated step 1.
        4. If web componenet is servlet then directroy service() method will be called.
        5. Once service() method of servlet is completed then Pops the web componenet from Stack 2 one by one and executes the remaing code of filters for POST PROCESSING.



Servlet Chaining	Filter Chaining
Invoking the servlet one by one a chain is called as Servlet Chaining.	Invoking will implemented filter Chain is called as Filter Chaining.
You can implemented servlet Chaining with the help of RequestDispatcher	Container will implemented Filter Chaining automatically based on the configuration of Filter in web.xml.
Oder of servlet in Servlet chain should be managed by you progamatically.	Order of filter in filter chain will be managed by container declarativlely based on the configuration of filters in web.xml(Servlet2.x) Alphabetic order of fully qualified names of the filters(Srvlet 3.x)
If you want to change the order of servlet in servlet chain,then you have to modify the code.	If you want to change the order of filters in Filter chain. Then you just change order of filter configration in web.xml

If you want to add or remove the Servlets to or from Servlet chain, then. You have to mpdify the code.	If you want to add or remove the filters to or from Filter chain then You just change order of filter configuration in web.xml(Servlets 2.x) Change the name of the Filters(Servlets 3.x)



Part-7

Listener:

- When your web application is runing in the web Container, following tasks will be done by the web container and devloper.
    1. Creating ServletContext object
    2. Destroying ServletContext object
    3. Creating HttpServletRequest object
    4. Destroying HttpServletRequest object
    5. Creating HttpSession object
    6. Destroying HttpSession object
    7. Adding attributes to ServletContext.
    8. Removing Attributes from ServletContext
    9. Replacing attributes from ServletContext
    10. Adding attributes to HttpServletRequest
    11. Removing Attributes from HttpServletRequest
    12. Replacing attributes from HttpServletRequest
    13. Adding attributes to HttpSession
    14. Removing Attributes from HttpSession
    15. Replacing attributes from HttpSession
    16. Passivating HttpSession object
    17. Activating HttpSession object
- Following are the List of Listerner interfaces available in Servlet in Servlet API.

Listerner Interface	Event Class	Task will be listened
ServletContextListener	ServletContextEvent	Task 1, Task 2
ServletRequestListener	ServletRequestEvent	Task 3,Task 4
HttpSessionListener	HttpSessionEvent	Task 5, Task 6
ServletContextAttributeListener	ServletContextAttributeEvent	Task 7, Task 8, Task 9
ServletRequestAttributeListener	ServletRequestAttributeEvent	Task 10, Task 11,Task 12
HttpSessionAttributeListener	HttpSessionBindingEvent	Task 13,Task 14,Task15
HttpSessionBindingListener	HttpSessionBIndingEvent	
HttpSessionActivationListener	HttpSessionEvent	Task 16,Task 17

Steps to write the Listener with Servlet 2.x

- Write your own listener class by implementing the required Listener interface.
- Override the Corressponding methods of the Listener interface which you are implementing.

```
Class MyListner implements ServletContextListener{
      Public void contextInitialized(sce){
            //your code here
            }
      Public void contextDestroyed(sce){
            //your code here
      Ctx=sce.getServletContext();
}
}
```

- Register the listener by writing the following in web.xml

```
<listener>
  <listener-class>com.jtc.MyListener</listener-class>
</listener>
```

Steps to write the Listerner With Servlet 3.0

- Write your own listener class by implementing the required Listner interface.
- Override the corresponding methods of the Listener interface which you are implementing.
- Register the listing by using @WebListener annonation.

@webListerner

```
      Class MyListner implements ServletContextListener{

      Public void contextInitialized(sce){

            //your code here

            }

      Public void contextDestroyed(sce){

            //your code here

      Ctx=sce.getServletContext();

}

}
```

Note: Alll the Listeners Configured will be intialized by container at container start-up.

Container Tasks at ontainer Start-up

- Creates the ServletContext object.
  - ServletContext sctx=new ServlletContextImpl();
- Gets the context parameters from web.xml and stores in ServletContext object.
- Creates ServletContextEvent
  - ServletContextEventListener sce=new ServletContextEvent(sctx);
- Create the ServletContextListener object
  - ServletContextListener listner= new ServletContextListenerImple();
- Invoke the contextinitialized() method with ServletContextListerner object
  - Listener.contextInitialized(sce).

Container Tasks at Container Shutdown time

- Invoke the contextDestroyed() method with ServletContextListener object
  - Listener.contextDestoryed(sce).
- Destroyes the SeervletContext object

Jtc27: files Required in Servlet 2.x:

1. Index.html
2. TestServlet.java
3. MyContextListener.java
4. MyContextAttributeListerner.java
5. MySessionListener.java
6. MyRequestListener.java
7. Web.xml

1. Index.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>Java Training Center</h1>
<h2>Listener Example</h2>
<form action="test.jtc" method="post">
```

3. TestServlet.java

```
package com.jtcindia.Servlet;

import java.io.IOException;
import java.io.Writer;

import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
```

```
Enter Email:<input type="text"
name="email">
<br/>
<input type="submit" value="submit">
</form>
</body>
</html>
```

2. MyContextListener.java

```
package com.jtcindia.Servlet;

import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call    :+91-9990399111
* */
public class MyContextListener implements
ServletContextListener{
public MyContextListener(){
System.out.println("** MyContextListener() def
consol");
        }
        @Override
public void contextDestroyed(ServletContextEvent
event) {
        ServletContext
ctx=event.getServletContext();
        System.out.println("ContextDestroyed:"+ctx
);
        }

        @Override
public void contextInitialized(ServletContextEvent
event) {
        ServletContext
ctx=event.getServletContext();
        System.out.println("ContextInitialized:"+ctx)
;
        }
}
```

4. MyContextAttributeListener.java

```
package com.jtcindia.Servlet;

import javax.servlet.ServletContextAttributeEvent;
import javax.servlet.ServletContextAttributeListener;
/*
```

```
*@Call    :+91-9990399111
* */
public class TestServlet extends HttpServlet {
        @Override
protected void service(HttpServletRequest req, HttpServletResponse
res)
                        throws ServletException, IOException {
System.out.println("TestServlet-> service()");
System.out.println("Acccessing Session object");
        HttpSession session=req.getSession();
        String eml=req.getParameter("email");
        ServletContext ctx=getServletContext();
        System.out.println("Storing attribute in Context");
        ctx.setAttribute("email",eml);
System.out.println("Replacing attributes in Context");
        ctx.setAttribute("email","som@jtc.com");
        System.out.println("Removing attribute in Context");
        ctx.removeAttribute("email");
System.out.println("validating Session Objects");
        session.invalidate();
        Writer out=res.getWriter();
        out.write("<h1>Verify the server console");
        }
}
```

5. MySessionListener.java

```
package com.jtcindia.Servlet;

import javax.servlet.http.HttpSessionEvent;
import javax.servlet.http.HttpSessionListener;
/*
* @Author    : Som Prakash Rai
```

```
* @Author    : Som Prakash Rai
* @Join       : Java Training Center
* @visit       : www.jtcindia.org
*@Call    :+91-9990399111
* */
public class MyContextAttributeListener implements
ServletContextAttributeListener {
public MyContextAttributeListener(){
System.out.println("** MyContextAttributeListener()
Def Console");
        }
        @Override
public void
attributeAdded(ServletContextAttributeEvent event)
{
        String nm=event.getName();
        String val=event.getValue().toString();
        System.out.println("**
AttributeAdded():"+nm+"\t"+val);
        }
        @Override
public void
attributeRemoved(ServletContextAttributeEvent
event) {
        String nm=event.getName();
        String val=event.getValue().toString();
        System.out.println("attributeRemove():"+n
m+"\t"+val)
        }

        @Override
public void
attributeReplaced(ServletContextAttributeEvent
event) {
        String nm=event.getName();
        String val=event.getValue().toString();

        System.out.println("attributeRemove():"+n
m+"\t"+val);
        }
    }
```

6. MyRequestListener.java

```
package com.jtcindia.Servlet;

import javax.servlet.ServletRequestEvent;
import javax.servlet.ServletRequestListener;
/*
* @Author    : Som Prakash Rai
* @Join       : Java Training Center
* @visit       : www.jtcindia.org
*@Call    :+91-9990399111
* */
```

```
* @Join       : Java Training Center
* @visit       : www.jtcindia.org
*@Call    :+91-9990399111
* */

public class MySessionListener implements
HttpSessionListener {
        public MySessionListener(){
                System.out.println("**
MySessionListener() def Cons");
        }

        @Override
        public void sessionCreated(HttpSessionEvent
arg0) {
                System.out.println("SessionCreated");

        }

        @Override
        public void sessionDestroyed(HttpSessionEvent
arg0) {

        System.out.println("**SessionDestroyed");

        }

}
```

7. 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/javae
e http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
id="WebApp_ID" version="3.0">
  <display-name>Jtc27</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
```



```
public class MyRequestListener implements
ServletRequestListener {
	public MyRequestListener(){

	System.out.println("MyRequestListener()
def console");
	}

	@Override
	public void
requestDestroyed(ServletRequestEvent arg0) {

	System.out.println("requestDestroyed");

	}

	@Override
	public void
requestInitialized(ServletRequestEvent arg0) {

	System.out.println("requestInitialized");

	}
}
```

```
  </welcome-file-list>
  <servlet>
  <servlet-name>testServlet</servlet-name>
  <servlet-
class>com.jtcindia.Servlet.TestServlet</servlet-
class>
  </servlet>
  <servlet-mapping>
  <servlet-name>testServlet</servlet-name>
  <url-pattern>/test.jtc</url-pattern>
  </servlet-mapping>
  <listener>
  <listener-
class>com.jtcindia.Servlet.MyContextListener</listen
er-class>
  </listener>
  <listener>
  <listener-
class>com.jtcindia.Servlet.MyContextAttributListener
</listener-class>
  </listener>
  <listener>
  <listener-
class>com.jtcindia.Servlet.MyRequestListener</listen
er-class>
  </listener>
  <listener>
  <listener-
class>com.jtcindia.Servlet.MySessionListener</listen
er-class>
  </listener>

</web-app>
```

Jtc28: Files Required in Servlet 3.x:

1. Index.html(same as Jtc27)
2. TestServlet.java (same as Jtc27)
3. MyContextListener.java(same as Jtc27)
4. MyContextAttributeListerner.java(same as Jtc27)
5. MySessionListener.java(same as Jtc27)
6. MyRequestListener.java(same as Jtc27)
7. Web.xml(same as Jtc27)

Jtc29: Files Required in Servlet 2.x:

1. Index.jsp
2. Logout.jsp

3. LogoutServlet.java
4. MyContextListener.java
5. MySessionListener.java
6. Web.xml

Index.jsp

```
<%@ page language="java"
contentType="text/html; charset=ISO-
8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML
4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type"
content="text/html; charset=ISO-8859-
1">
<title>Insert title here</title>
</head>
<body>
<h1>Java Training Center</h1>
<table align="right">
<tr>
<td><h1>Total
Visited:<%=application.getAttribute("TV
") %></h1>
</td>
</tr>
<tr>
<td><h1>Total
Active:<%=application.getAttribute("TA"
) %></h1>
</td>
</tr>
</table>
<br/>
<br/>
<h1>This is Index Page</h1><br/><br/>
<a href="test">LOGOUT</a>
</body>
</html>
```

LogoutServlet.java

```
package com.jtcindia.servlet;

import java.io.IOException;

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

Logout.jsp

```
<%@ page session="false" %>
<%@ page language="java" contentType="text/html;
charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01
Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html;
charset=ISO-8859-1">
<title>Insert title here</title>
</head>
<body>
<h1>Java Training Center</h1>
<table align="right">
<tr>
<td>
<h1>Total Visited:<%=application.getAttribute("TV")
%></h1>
</td></tr>
<tr>
<td><h1>Total
Active:<%=application.getAttribute("TA") %></h1>
</td></tr>
</table><br/><br/>
<h1>You Have logged out Successfully</h1>
<br/><br/>
<a href="index.jsp">GO TO INDEX PAGE</a>

</body>
</html>
```

MyContextListener.java

```
package com.jtcindia.servlet;

import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;
/*
* @Author    : Som Prakash Rai
* @Join      : Java Training Center
* @visit     : www.jtcindia.org
*@Call    :+91-9990399111
* */
```

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
/*
* @Author    : Som Prakash Rai
* @Join       : Java Training Center
* @visit        : www.jtcindia.org
*@Call    :+91-9990399111
* */


public class LogoutServlet extends HttpServlet {
	@Override
	protected void service(HttpServletRequest
req, HttpServletResponse res)
		throws ServletException,
IOException {
		HttpSession
session=req.getSession(false);
		if(session!=null){
			session.invalidate();
			RequestDispatcher
rd=req.getRequestDispatcher("logout.jsp");
			rd.forward(req, res);
	}
	}
}
```

MySessionListener.java

```
package com.jtcindia.servlet;

import javax.servlet.ServletContext;
import javax.servlet.http.HttpSession;
import javax.servlet.http.HttpSessionEvent;
import javax.servlet.http.HttpSessionListener;
/*
* @Author    : Som Prakash Rai
* @Join       : Java Training Center
* @visit        : www.jtcindia.org
*@Call    :+91-9990399111
* */

public class MySessionListener implements
HttpSessionListener {

	@Override
	public void
sessionCreated(HttpSessionEvent event) {
		HttpSession
session=event.getSession();
		ServletContext
ctx=session.getServletContext();


public class MyContextListener implements ServletContextListener {

	@Override
	public void contextDestroyed(ServletContextEvent event) {
	System.out.println("contextDestroyed");

	}

	@Override
	public void contextInitialized(ServletContextEvent event) {
		System.out.println("contextInitialized");
		ServletContext ctx=event.getServletContext();
		ctx.setAttribute("TV", 0);
		ctx.setAttribute("TA",0);

	}

}
```

Web.xml

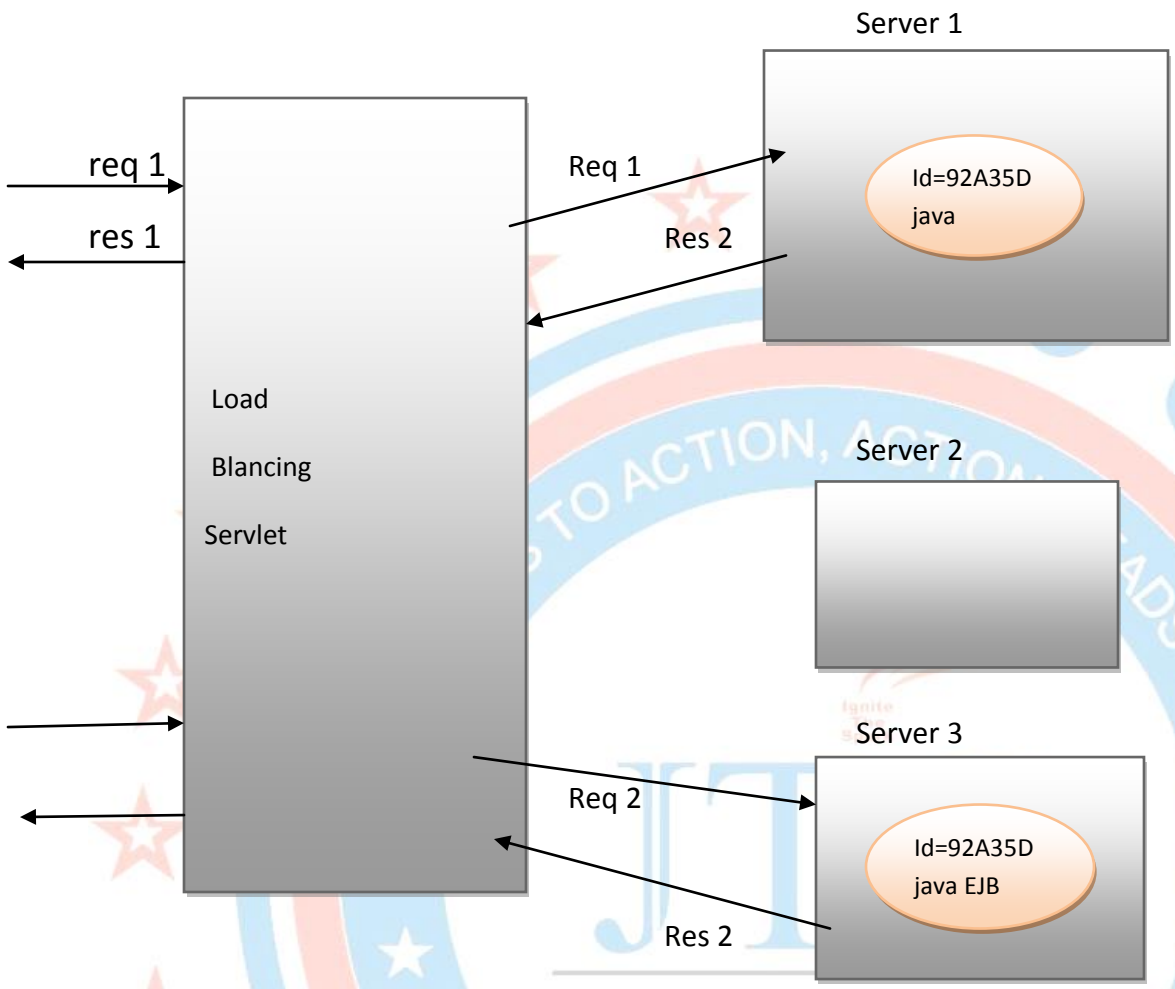
```
<?xml version="1.0" encoding="UTF-8"?>
<web-app
xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javae
e http://java.sun.com/xml/ns/javaee/web-app_3_0.xsd"
id="WebApp_ID" version="3.0">
  <display-name>Jtc29</display-name>
  <welcome-file-list>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
  <servlet-name>logoutServlet</servlet-name>
  <servlet-
class>com.jtcindia.servlet.LogoutServlet</servlet-
class>
  </servlet>
  <servlet-mapping>
  <servlet-name>logoutServlet</servlet-name>
  <url-pattern>/test</url-pattern>
  </servlet-mapping>

  <listener>
  <listener-
class>com.jtcindia.servlet.MyContextListener</listen
er-class>
  </listener>
  <listener>
  <listener-
class>com.jtcindia.servlet.MySessionListener</listen
er-class>
```

```
                int
tv=(Integer)ctx.getAttribute("TV");
                tv++;
                ctx.setAttribute("TV",tv);
                int
ta=(Integer)ctx.getAttribute("TA");
                ta++;
                ctx.setAttribute("TA",ta);
        }
        @Override
        public void
sessionDestroyed(HttpSessionEvent event) {
                HttpSession
session=event.getSession();
                ServletContext
ctx=session.getServletContext();
                int
ta=(Integer)ctx.getAttribute("TA");
                ta--;
                ctx.setAttribute("TA",ta);

        }
}
```

```
  </listener>

</web-app>
```

Session Migration:

- When you have large-scale enterprise application, then lakhs of users may visit the application at a time.
- To manage lakhs of users concurrently, you need to implemented Clusterd Environment with More Load balancing Servers and more application servers.
- When user sends the request, first LBS takes that request and forwards to free servlet at Clustered Unit.
- Some times One user request may be forwarded to different servers.

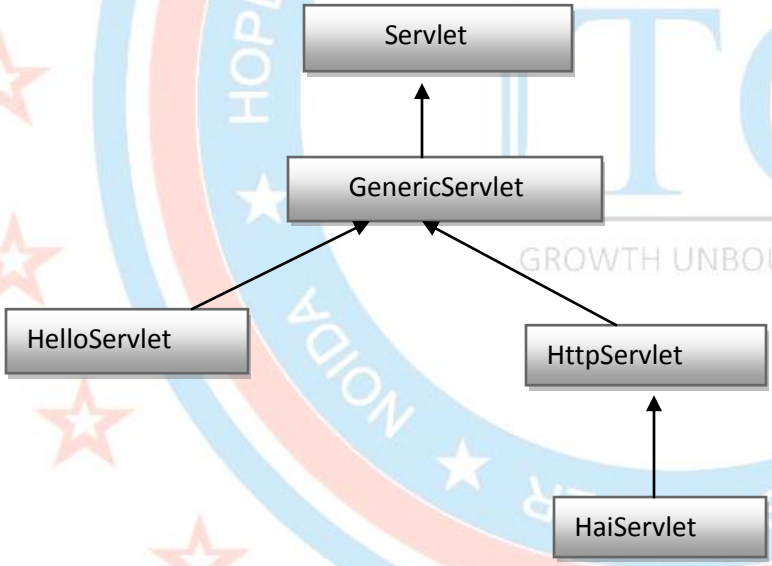


- User first request given to SERVER 1 where session object is created for thatuser.
- User second request given to SERVER 3.
- Now session object of that user is required at SERVER 3.
- In this case you have to migrate the session object from SERVER 1 to SERVER 3.
- Following events will happen at session Migration:
  - Session will be passivated at SERVER 1.
  - Session will be Activated at SERVER 3.
- To perform any task at these events, you can register HttpSessionActionvationListener with the following methods:

- Void session WillPassivate(event): will be called by the web container at SERVER 1.
- Void sessionDidActivate(event);will be called by the web container at SERVER 3.

Servlet API

- Javax.servlet.Servlet is an interface which is root for all the servlets you are devloping.
- Javax.servlet.GenericServlet class is the directly sub class of Servlet interface.
- GenericServlet is overriding all the methods of Servlet interface except service() method because of that GenericServlet is declared as abstract.



- When you devlope the servlet by extending GenericServlet then you have to override the service() method as follows.

```
Public void service(ServletRequest req, ServletRequest res)->GENERIC SERVICE

}
class HelloServlet extends GenericServlet{
      public void service(HttpServletRequest req, HttpServletResponse res){
                  //processing code
            }
```

```
}
```

- Javax.servlet.http.HttpServlet class is the direct sub class of GenericServlet.
- HttpServlet is overriding all the methods of GenericServlet but HttpServlet is declared as abstract.
- When you devlope the Servlet by extending HttpServlet then you can override any of the following 9 methods:
  - `void service(HttpServletRequest req, HttpServletResponse res)`
  - `void doGet(HttpServletRequest req, HttpServletResponse resp)`
  - `void doPost(HttpServletRequest req, HttpServletResponse resp)`
  - `void doPut(HttpServletRequest req, HttpServletResponse resp)`
  - `void doDelete(HttpServletRequest req, HttpServletResponse resp)`
  - `void doHead(HttpServletRequest req, HttpServletResponse res)`
  - `void doTrace(HttpServletRequest req, HttpServletResponse res)`
  - `void doOptions(HttpServletRequest req, HttpServletResponse res)`

- Whatever the Servlet class you are extending or Whatever the method you are overriding, container always calls the GENERIC SERVICE method.

```
Class cls=Class.forName("com.jtcindia.HelloServlet");
Object obj=cls.newInstance();
Servlet s=(Servlet)obj;
s.service(req,res);
```

Case 1: I have written HelloServlet by extending HttpServlet and overriden Generic service method.

```
interface Servlet{
public abstract void service(ServletRequest req,ServletResponse res) ;

}
abstract class GenericServlet implements Servlet{
public abstract void service(ServletRequest req,ServletResponse res) ;
}
abstract class HttpServlet extends GenericServlet{
public abstract void service(ServletRequest req,ServletResponse res) ;
//some code
}

 class HelloServlet extends HttpServlet {
```

```
public abstract void service(HttpServletRequest req, HttpServletResponse res){
            //some code

        }
```

Explanation:

```
Class cls=Class.forName("com.jtcindia.HelloServlet");
Object obj=cls.newInstance();
Servlet s=(Servlet)obj;
s.service(req,res);
```

- Container calls Generic service service method and Generic service method which is override in the HelloServlet will be called.

Case 2: I have written HelloServlet by extendinng Httpservlet and overriden HTTP service method.

```
Class HelloServlet extends GenericServlet {

    @Override
    public void service(ServletRequest req, ServletResponse res)
            {
        HttpServletRequest req1=(HttpServletRequest)S.req;
        HttpServletResponse req2=(HttpServletResponse)S.res;
        service(req1,req2);
    }

  protected void service(ServletRequest req, ServletResponse res){
    //some code
        }
    ….
    }
        Class HelloServlet extends HttpServlet(){
    protected void service(ServletRequest req, ServletResponse res){
        //some code
  }
```

Explanation:

- Container calls generic service method.
- Generic service method of HttpServlet which is inherited to HelloServlet will be called.
- Generic service method of HttpServlet invokes the HTTP service method.
- Generic service method of HelloServlet will be called.

Case 3: I have written HelloServlet by extending GenericServlet and Override Generic service method.

```
class HttpServlet extends GenericServlet {
@Override
protected void service(ServletRequest req, ServletResponse res)
                {
        //some code
        }
```

Explanation:

- Container calls generic service method.
- Generic service method which is overriden in HelloServlet will be called.

Case 4: I have written HelloServlet by extending GenericServlet and overriden HTTP service method.

```
Abstract class GenericServlet impl servlet {
        @Override
        protected void service(ServletRequest req, ServletResponse res)
                        throws ServletException, IOException {
                //some code
                }
class HttpServlet extends GenericServlet {
        @Override
        protected void service(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException {
                //some code
                }
```

Explanation:

You will get compile time error.

Case 5: I have writen HelloServlet by extending HttpServlet and overriden HTTP service method and Generic service method.

```
class HttpServlet extends GenericServlet {
        @Override
        protected void service(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException {
                //some code
```

```
            }
public void service(ServletRequest req, ServletResponse res)
                    throws ServletException, IOException {
            //some code

      }
```

Explanation:

- Container calls generic service method.
- Generic service method which is overriden in HelloServlet will be called.

Case 6: I have written HelloServlet by extending HttpServlet and overriden doGet() method.

```
 class HttpServlet extends GenericServlet {
      @Override
      public void service(ServletRequest req, ServletResponse res)
                    throws ServletException, IOException {
            //calls http service method
            }
      }

      protected void service(ServletRequest req, ServletResponse res)
                    throws ServletException, IOException {
            String m=req.getMethod();
            doGet(req,res);
            }else if(m.equals("POST")){
            doPost(req,res);
      }
protected void doGet(HttpServletRequest req, HttpServletResponse res)
                    throws ServletException, IOException {
            //send the error with Http status Code:405
      }
      @Override
      protected void doPost(HttpServletRequest req, HttpServletResponse res)
                    throws ServletException, IOException {
            //send the error with Http status Code:405
      }
 class HelloServlet extends HttpServlet {

      @Override
      public void doGet(HttpServletRequest req, HttpServletResponse res)
                    throws ServletException, IOException
            //send the error with Http status Code:405
      }
```

Explanation:

- Container calls generic service method.
- Generic service method of httpService which is inherited to HelloServlet will be called.
- Generic service method of HttpServlet invokes the HTTP service method.
- HTTP service method of HttpServlet will be called.
- HTTP service method of HttpServlet checks the incoming request http method and invokes the corresponding doXX() method.
    A. doGet() method will be called from HelloServlet.
    B. doPost() method will be called from HelloServlet.
- HTTP Status 405-HTTP method POST is not supported by this URL.

Case 7: I have written HelloServlet by extendingHttpServlet and overriden doGet(),doPost() and HTTP service methods.

A. Whst will happen when is send the request with GET?

ANS: HTTP service method of HelloService will be called.

B. What will happen I send the request with POST?
   Ans: HTTP service method of HelloServlet will be called.

Note:

- When you override GENERIC SERVICE method in your method, then that only will be called for any incoming request http method.
- When you override HTTP SERVICE method in your servlet, then that only will be called for any incoming request http method.
- When you override any doXX() method in your servlet, then that method will be called as per matching incoming request http method.