Servlets:

From java 1.2 onwards after introducing Servlets, Jsp concepts into java,
we have three flavour in java.

1. Java2SE/ J2SE
2. Java2EE/ J2EE
3. Java2ME/ J2ME

2 represents version number.

From java 1.5 onwards, sun micro system delete the 2 for representation and start calling directly as

1. JavaSE / JSE
2. JavaEE / JEE
3. JavaME / JME

JSE:

---

1. Java Stanadard Edition.
2. It is used to develop Standalone applications.
3. It is installable software.
4. Standalone applications may be CUI, GUI.
5. CUI we can develop through commandprompt.
6. GUT we can develop through Browsers (windows/Frame).
7. Standalone resource are sharable by the only one person at a time,
   that means data is not sharable between multiple client or enduser at a time.
8. we required command prompt to compile and execute the programs by using javac, java commands.
9. We required JDK software intallation.
10. JDK contains both compiler and JVM.
11. We are always using command prompt for giving the data to program and getting response from program and print the data (output) on command prompt by using System.out.println() statements.
12. To work with JSE programs we required JDK API.
13. We save our .java files and .class files any folder and in any directory.

14. we can compile and execute the program from any where in our machine.

JEE:
-----

1. Java Enterprise Edition.(Java2 Platfrom Enterprise Ediition)
2. It is not installable software.
3. We can develop web, distributed, enterprise application.
4. These application resources can be sharable between multiple user at a time through web in throughout world.
5. To develop applications in JEE, we required JDK software installation and JEE API.
6. To compile the program we required commandprompt and javac command. This command given by JDK software.
7. To execute the program we required Browsers(google chrome, Mozilla firefox, Internet Explorer, Safari, Opera Mini) and Servers(Apache tomcat, Glass Fish, web logic..)-
8. We are always view the data or content on browsers only either static or dynamic.
9. To print the data on browser, we use out.println() statements.
10. We are always save our .class file in specific folder that is "webapps".
11. To provide response to all the enduser, that means to execute program and giving response all the endusers, with in the same time we required servers. (Apache Tomcat, GlassFish, Weblogic,JBoss).

Software Technology:
----

1. Technology will provides some rules and guide lines.
2. Rules are comming in the form interface.
3. Guide line are comming in the form of classes.
4. The combination of interface and classes simply we can called as API.
5. These technology are usefull for developing technology based software.
6. Here JDBC, SERVLETS, JSP are technologies.
7. Apache Tomcat, GlassFish, JBoss, Weblogic all are technology based softwares.

Need of web application:

By using JSE, we can able to develop standablone applications, means the applications are running under only one desktop and those applicastion services will be access by only one end user at a time.(limited people can access).

ex: all corejava programs

JDBC programs, Socket promming etc....

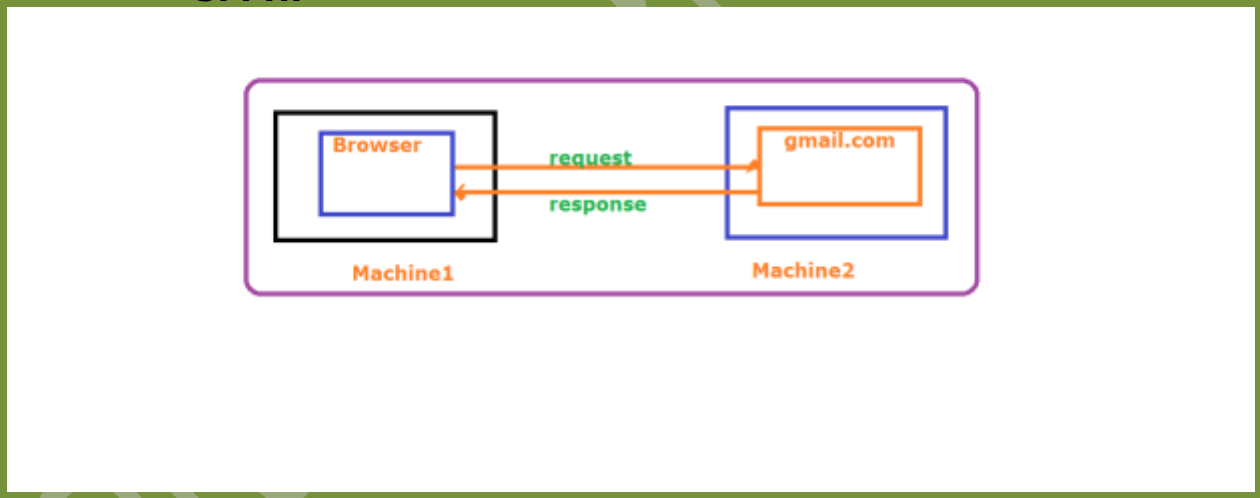
If we want provides all the services or resources of any application throughout the world we should choose webapplications.

WebApplication:

It is one type of applications running through web or running under the servers for provides resources to as.l the end-user through out world at a time is called web application.

we can develop web applications we have three technologies.

1. **Java**
2. **Asp.Net**
3. **PHP**



Web application always uses one http protocol for making communication between client to server.

php technology we are using to develop websites,the number of pages of website are less than 10 pages and number requests comming to that website is low.

note: php suitable for small-scale applicatons.

Asp.net technology is used for developing small business applications of around pages lessthan 20 and numbe requests comming to applications are medium.
note: Asp.net is used for medium-scale applications.

Java technology is used for developing ecommerce applications, banking application etc... These are having multiple pages like morethan 30 and number of request also more.
note: Java is used for large-scale applications.

After arival of jquery and angular-js , JAVA TECHNOLOGY is suitable for all type of applications.

Webapplication means request and response will be happend through http protocol.

these web applications we can also called thin client- thick server. thin client means only browser is sufficient no need of any extra software configurations. Thick server means we required extra software installation process.

Types of web applications:

Mainly web applications are two types.

1. **static web applications**
2. **dynamic web applications**

static web applications means the content or response of the web application is common for all the enduser is called static web application.

To develop any static web applications we required static web components.
A language or technology is used to develop for web applications is called web component.

a. **HTML**
b. **java script**
c. **css**
d. **ajax**

ex: www.nareshit.com

www.java4s.com

Dynamic web applications means the content or response of the web application is changing from one enduser to another enduser is called dynamic web applications.

To develop dynamic web applications we required the following web components

- **a. Servlets**
- **b. Jsp**
- **c. Php**
- **d. Asp.Net**

ex: www.gmail.com
www. facebook.com

If we want to develop the standalone applications we required JSE API, to compile the program we required compiler, we will get that component by using "javac" tool. To executing the program we required jvm, we will get that component by using "java" tool.

In the same manner to develop the web application we requied JEE API, to compile the web application we required compiler, we will get the component by using "javac" too by installation of JDK software.
But to execute the program we required server softwares.

These server softwares are developing by the third party vendors. These server softwares are implementing on top of specifications given by JEE API through java software.

Java software provides rules and guidelines through technology wise like Servlet, Jsp.
These rules are followed by the third party vendors and developing technology based software like Tomcat, Apache web server, Glass Fish,
weblogic etc.....

loosely coupling and runtime polymorphisam:
----

interface Sim{

```
	public abstract void call();
	public abstract void sms();
}

class Vodafone implements Sim{
	public void call(){
		System.out.println("call connection through vodafone");
	}
	public void sms(){
		System.out.println("sms delivered your charges are: 1/-
");
	}
}

class Idea implements Sim{
	public void call(){
		System.out.println("waiting from network signal");
	}
	public void sms(){
		System.out.println("There is no sms offer");
	}
}

class Mobile{
	public static void main(String[] s1){
		Sim s = new Vodafone();
		s.call();
		s.sms();
		//loosely coupling
		s = new Idea();
		s.call();
		s.sms();
	}
}
```

Q)what is the difference between web application and web site?
site means memory/location which is holds applications.
In general, every body feels like both are same but there is a small difference between web site and web application.

Website is the collection of web applications. It will provide different services to
different endusers. Every application having one unique service.

Webapplication is the collection of web components to provides services to end user.

ex: www. nareshit. com

nareshit web site having multiple or collection of webapplications like course details, faculty details, course timings, new batches, special batches etc……

Before deploy program related to web in internet we can called as web application , after deply web program in the internet we can called as website.

JEE:

- **It is one type of flavour/edition in java from 1.2**
- **It is a technology.**
- **It will provide specifications(rule and guide lines).**
- **Rules comming in the form of interface**
- **Guide lines comming in the form classes.**
- **It is the combination following specifications.**
  1. **Protocols specifications.**
  2. **Servlet specifications.**
  3. **jsp specifications.**
  4. **EJB specifications.**
  5. **Middleware services specifications.**

For above spefications are implemented/provide services by the same organizations/company (oracle) or other third party vendor companies.

ServerName	Third Party Vendor Name
Apache Tomcat	Apache Foundations
Web Sphere	IBM
GlassFish	Oracle
Web logic	BEA (Oracle)

Server functionalities:

----

1. getting client request.
2. loading/deploy the web components (loading the bytecode)
3. creating instances(object).
4. processing the request
5. if required communicating with files, database, any middileware services
6. generating response
7. handover response to client
8. destroying the instances.
9. undeploy the web components.

Q) What is JEE Technology based servers?
A) The Server, which is implemented on top JEE specifications or rules is called
JEE technology based servers.

Types of Servers:

Mainly Servers are two types.
1. Web Server.
2. Application Server.

WebServer:

A server which will provide services to/implementation for/ logic for servlet specifications and jsp specifications and protocol specifications.
ex: Apache Tomcat.
It will support web protocols like http, https, smtp ....

Application Server:

A server which will provide services to/implementation for/ logic for
all specifications given by JEE Technology is called Applications Server.

What is WebComponent:

It is reusable object for providing services to end user.(servlet, jsp, ejb..)

What are the mandatory setps for developing web project?

1. Browsers:
a) Mozilla Firefox.
b) Opera Mini.
c) Safari

d) Google Chrome
e) Internet explorer etc...

2. Web Components:
   a) Servlets
   b) Jsp
   c) html
   d) css
   e) java script
   f) images
   g) .property files

3. Servers
   a) Tomcat
   b) weblogic
   c) websphere
   d) glassfish

Typs of websites:

---

there two types websites.
1. Informative websites
2. service oriented websites.

The websites will provides only static information for all the endusers is called informative websites.

The websites will provides different dynamic web content to different endusers based on input or request.

JEE technology based WebServer:

A server which is using jee specification and providing implementations or services to enduser is called jee based webser.

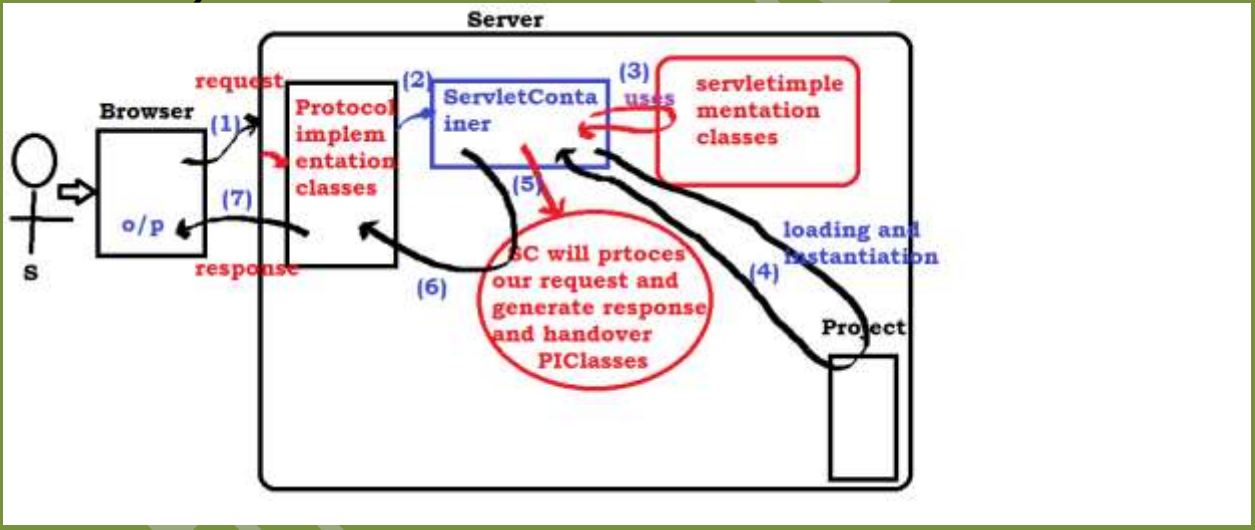
Internals of webserver:

WebServer mainly contains the following implementatios.
1. Http protocol implementation classes.
2. Servlet implementations classes
3. jsp implementation classes
4. Servlet container
5. Jsp container

Whenever enduser make a reuest to project(servlet component) for services the following steps will happen.

1. Our request will be handover to http protocol, it will carry that request and handover to httpprotocol implementation classes.
2. Httpimplementation classes will handover that request to Servlet container.
3. Servlet container will uses the servlet implementation classes.
4. By using the functionalities of servlet implementation classes Servlet container will processing(executing) our request.
5 & 6. After process the request we will get appropriate response that will be handover to again httpprotocol implementation classes by servlet container.

1. **Finally that response will be display on the browser (handover to enduser).**



whenever enduser making a request to jsp page the following steps will happen.

a. Our request will be handover to http protocol, it will carry that request and handover to httpprotocol implementation classes.
b. Httpimplementation classes will handover that request to Servlet container.
c. Servlet container will calls jsp container.
d. Jsp container will uses the jsp implementation classes.
e. By using jsp implementation classes container will process the jsp page
g. That response will be handover to http implementation classes.
h. Finally the response will hadover to enduser (display content on browser).

Jee technology based Application Server:

A server which will provides services or implementation for all the specifications gen JEE is application server.

```
class Servlet{
	String servletName = "welcomeservlet";
	private Servlet(){}
	//static Servlet obj = new Servlet();
	static Servlet obj = null;
	static Servlet getObjectForServlet(){
		//return new Servlet();
		if(obj == null){
			obj = new Servlet();
			return obj;
		}
		else
		return obj;
	}
}
class Container{
	public static void main(String[] s){
		/*Servlet s1 = new Servlet();
		System.out.println(s1.servletName);
		Servlet s2 = new Servlet();
		System.out.println(s2.servletName);
		*/

		Servlet s1 = Servlet.getObjectForServlet();
		System.out.println(s1.servletName);
		Servlet s2 = Servlet.getObjectForServlet();
		System.out.println(s2.servletName);
System.out.println(s1.hashCode()+"...."+s2.hashCode());
	}
}
```

If we want to generate a dynamic content or dynamic response to enduser we have following technologies like

CGI

Servlet

Common Gateway Interface:

CGI will provides a process/address for each and every request to generate appropriate dynamic response.

Here we are facing the one main problem like context switching, that means to providing the services for different client simultaneously, CGI will shift the controle frome one process to another process for that CGI will take more time.(time consuming process).

More resources are utilize.

Platform dependent.

Poor in performance

To avoiding above problems java provides one special web component that is SERVLET.

Servlet:

----

servelt is one technology will provides dynamic response to enduser.

For providing response to enduser servlet container will uses one separate thread.

Servlet will work on multithread mechanisam.

That provides response with in the less time, that means performance of an application is more.

servlet concept developed on top of java technology.

So it is the platform independency concept.

Thread is a light weight process, uses less memory.

Servlet will reponse to all the enduser equally.

Definition:

Servlet is one JEE web technology, is used to generate dynamic web content.

Servlet is one JEE Web technology, is used to create single instance-multiple response process.

Servlet is one JEE Web technology, will increases services of web and application servers.

Servlet is one JEE Web technology, will provices rules and guide lines to develop our own web components.

Servlet is one class (userdefine)it is used to extends new features.

If we work with Servlets we required one predefine .jar file that servlet-api.jar file, it is comes with web or application server.

How to download apache tomact server:

Goto google, type apache tomcat, click on search button.

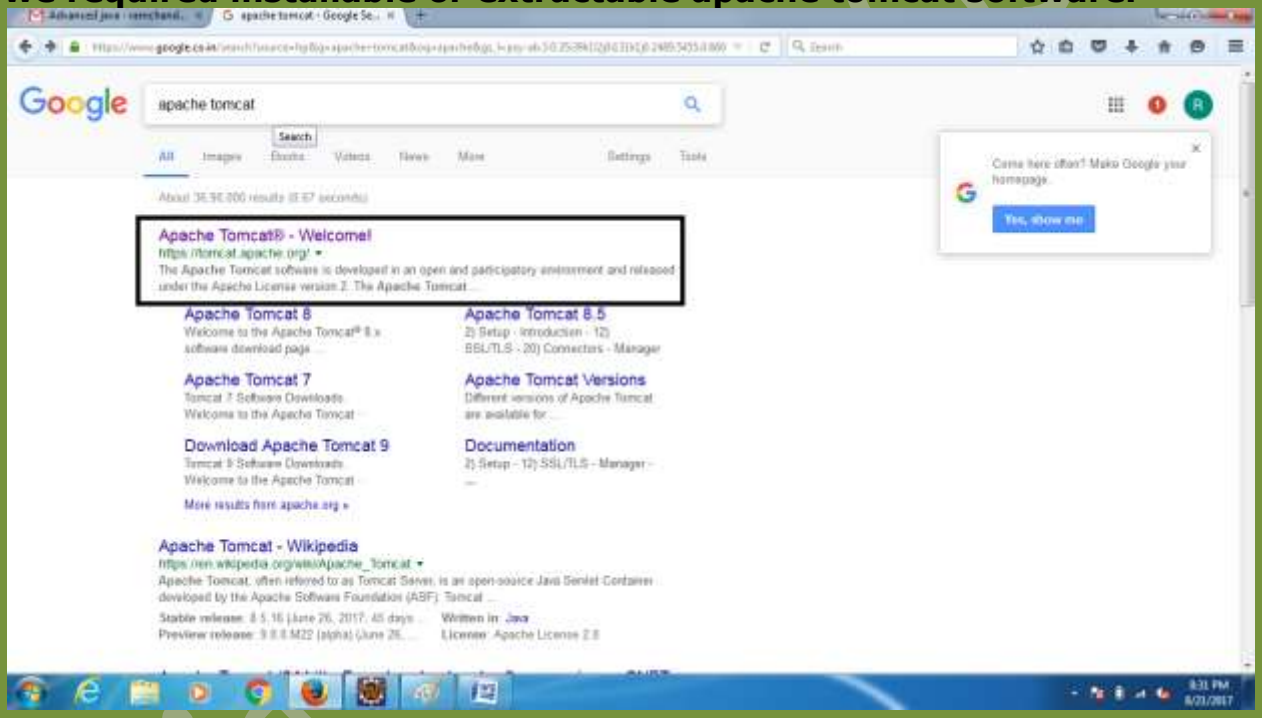
click first link which is available in our paint image.

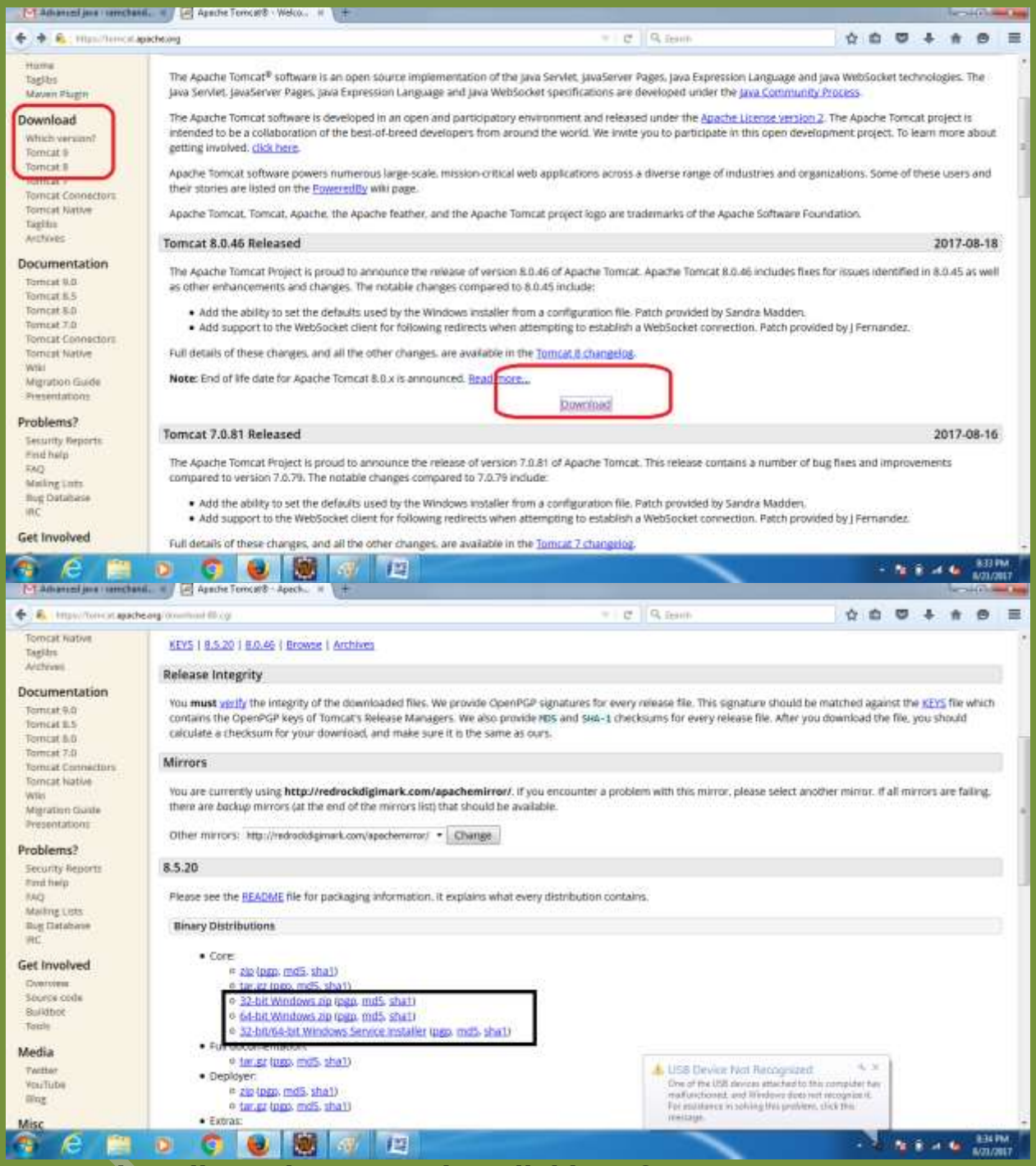
In new tab apache tomact home page will come. Left side of that page we have download link and different versions.

Click on any one type of version.

In the body of the browser we have binary distributions, under that we can download the either installable apache tomcat and extractable apche tomcat.

Whenever we work with eclipse we required installable apache software, whenever we work with "webapps" folder directly either we required installable or extractable apache tomcat software.





How to install apache tomcat installable software:

After downloading the apache tomcat software, double click on software.

click on yes

click on next

click on IAgree

click on next

change port nummbers

- **server shutdown port number : 7892**
- **http connector port number : 7890**
- **AJp connector port number : 7891**
- **username: ram**
- **password: ram**

click on next
select our JRE folder location
click on next
select destination location for installation
like C:\Program Files\Apache Tomcat Foundation\Tomcat 8.0
click install
UNSELECT

- **run apache tomcat**
- **show read me**

click Finish.

Download any latest eclipse software and placed into one folder.
right click on zip file and click on extract here.
----
open eclipse folder, click on .exe (application) (which is in round shape), select workspace(to save our programs in harddisk) location, click on ok.

How to add server to eclipse and developing first dynamic web project:

File-->new-->other-->web-->dynamic web project -->next-->
give the name for project: FirstProgramSer
click new Runtime: select apachetomcat version 8.0
select web module : 3.0
next
browse tomcat installation location
ok
select installation jre like jdk1.8.0_20
click on finish.

Firstprogram in servlet with Eclipse:

---

1. Install Eclipse in our system.
2. Develop one Dynamic web project.
   Goto File-->select new-->click on others
   --->goto Web folder-->select Dynamic web project
   --->click on next
   --->enter project name : FirstProgramSer
   --->select Dynamic web module version: 2.5
   --->click on Finish.
3. Install Apache Tomcat Server.
4. place servlet-api.jar file in buildpath/classpath.
   right click on our project-->select buildpath
-->click on configure build path-->click on add external jars
-->goto apache tomcat(tomacat1.7)/lib-->select servlet-api.jar file
-->click on open--->click ok.

5. Create ourown servlet class.

Extract our project---> extract java resources-->right click on src
-->new --->click on class-->Type class name -->click on finish.

type bellow code:

WelcomeServlet.java:

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class WelcomeServlet extends GenericServlet{
    @Override
    public void service(ServletRequest request,
                        ServletResponse response)
    throws ServletException,IOException{
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
        out.println("<html>");
        out.println("<body>");
out.println("<marquee>Hello welcome first program</marquee>");
        out.println("</body>");
```

```
            out.println("</html>");
        }
}
```

6. Develop web.xml file:

extract our project-->extract WebContent-->extract WEB-INF and check web.xml file, if available use it. If not available create web.xml in the following manner.

right click on WEB-INF-->select new -->click on File--> given name as web.xml--> click on finish button.

Type bellow code in that web.xml file:

```
<web-app>
        <servlet>
                <servlet-name>hello</servlet-name>
                <servlet-class>WelcomeServlet</servlet-class>
        </servlet>
        <servlet-mapping>
                <servlet-name>hello</servlet-name>
                <url-pattern>/hs</url-pattern>
        </servlet-mapping>

</web-app>
```

7. Run the project.

right click on the project-->run as ---> Run on Server ---->choose an existing server -->click finish.

8. check server console in the eclipse for our tomcat is properlly loading or not and check port number.

If any problem in loading those problem are shown in the eclipse console

9. Open browser and send request to server by using the following url.

http://localhost:portnumber/projectname/urlmappingname

http://localhost:1111/FirstProgramSer/hs

10. Check browser body for output, if any problem error will be display on browser.

Q) Is it mandatory to declare Servlet class as public?

A: Yes. For compiling Servlet program, there is no problem, but executing Servlet Program we required “public”. The reason is

Servlet Container will create object for our class, once we write Servlet class as public then only that call will visible to Servlet Container and creating object for that class.

Q) Is it mandatory to declare service (-,-) as public?
A: Yes. That is the principal of method overriding concept; sub class method access modifier must be same as super class method access modifier or increasable.

Q) Can we change service () return type?
A: No. Sub class method return type must be same super class method return type.

Q) Is it mandatory to write throws ServletException, IOException?
A: No. But our logic required or demands definitely we need to write.

Q) What is meaning of "out" variable and can we change or not?
A: It is just reference variable, if we want we can change the variable name also.

Q) Can we change service () name to other method?
A: No. We need to use same method name, Servlet container only recognize the method name like service (). Sub Class Override method must be same as super class method.

Q) What is functionality of out.println ()?
A: It will add response/information into ServletResponse object and carry that response/information from web server to browser and print that information/response on browser body.

Q) What is use of resp.setContentType ("text/html")?
A) This method is used to setting content-type HTTP response header to provide, information to the browser about the type of content we are sending.
This is also called MIME. (Multipurpose Internet Mail Extension).
text/html
text/plain

text/xml
application/msword
application/pdf
application/vnd.ms-excel
application/jar
application/javascript
application/json (javascript object notation)
applicaiton/zip
application/gzip
application/octet-stream
multiport/form-data
video/quicktime
video/avi
video/mp4
audio/mp4
image/gif
image/jpeg
image/png

<u>web.xml:</u>

It is descriptive file.

It provides or talks about Servlet mapping details and registration details.

By using these details only Servlet Container will select one appropriate Servlet class (user define class).

In web.xml we have one root tag that is <web-app>
<web-app> contains two main tags those are

1. **<servlet>**
2. **<servlet-mapping>**

<servlet> will provide details about Servlet_Class.

<servlet-mapping> will provide details about servlet URL names nothing but Servlet mapping details.

<servlet-name> will talks about Servlet registration name.

<servlet-class> will talks about actual Servlet class name.

<url-pattern> will talks about public URL name.
Servlet registration name and public URL name can be anything.

Whenever end user send request, from request container will read public URL name and comparing with <url-pattern> content/element.
If both are matched, container will read name/element from <servlet-name> of <servlet-mapping> later container will match this name/element with <servlet-name> element of <servlet>.

If both are matched, container will read Servlet Class Name from <servlet-class> later container will create object for that select class.

Internal flow of First Servlet program:

1. **Whenever request is sending by the end user, that request will be read by the web server**
2. **and handover to http implementation classes.**
3. **That request will be handover to Servlet Container.**
4. **Servlet container internally uses Servlet implementation classes.**
5. **Servlet container creates ServletContext object**
6. **Servlet container creates Thread Object**
7. **Servlet Container creates ServletRequest object with input data.**
8. **Servlet Container creates ServletResponse object created with empty.**
9. **ServletContainer creates PrintWriter Object**
10. **ServletContainer creates ServletConfig object**

11. ServletContainer will loads bytecode of our class and executes static loading and initailization phases.

12. Servlet Container creates Servlet (WelcomeServlet) object or instance and non-static loading phase and initilization phase and executes constructor.

13. By using Servlet instance, ServletContainer will inject ServletConfig into init(-) and executes.

14. By using servlet instance, Servlet container calls service method by passing ServletRequest object and ServletResponse

15. service (-,-) of our own Servlet class (WelcomeServlet) will be executing.

16. With the support of PrintWriter Output/response will be placed into ServletResponse object..

17. Generates response in the form html page.
18. Handover html form (response) to http implementation classes.

19. Finally output will be printed on the browser.

20. Destory the Thread object

21. Destroy ServletRequest object.

22. Destroy ServletResponse object.

Note: Servlet object (WelcomeServlet) is created only one time. That object will be useful for providing response to all end users. Servlet architecture depends one design pattern that is Singleton Design Pattern. That Servlet is a singleton object.

As many requests are coming to our application, those many the same servlet object will used by Servlet Container to provides services to end users.

As many requests are coming to our application, those many times new ServletRequest object and ServletResponse object will be creating by Servlet container and after providing the response to end user both the object will be destroy by the Servlet Container.

Generating Dynamic Response Ro Client:

DateServletDemo.java:

```
package com.kit.aj;

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

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

public class DateServletDemo extends GenericServlet{
	@Override
	public void service(ServletRequest request,
			ServletResponse response)throws ServletException,
			IOException{
		response.setContentType("text/html");
		PrintWriter out = response.getWriter();
		Date d = new Date();
SimpleDateFormat sdf = new SimpleDateFormat("MMM-dd-YYYY
hh:mm:ss");
		String date = sdf.format(d);
		out.println("<html>");
		out.println("<body>");
		out.println("The Current Time Is: "+date);
		out.println("<body>");
		out.println("<html>");
	}

}
```

Web.xml:

```
<web-app>
	<servlet>
		<servlet-name>date</servlet-name>
	<servlet-class>com.kit.aj.DateServletDemo</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>date</servlet-name>
		<url-pattern>/date</url-pattern>
	</servlet-mapping>
</web-app>
```

Servlet-Singleton:

Servlet Container always create single object for our servlet class.

For multiple requests Servlet Container will uses only one servlet object.

As many requests coming our application those many times ServletRequest and ServletResponse object will be creating by the servlet container.

SingletonServletDemo:

```
package com.aj.ram;

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

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

public class SingletonServletDemo extends
	GenericServlet{
	@Override
	public void service(ServletRequest request,
				ServletResponse response)throws
ServletException,IOException{
```

```
            response.setContentType("text/html");
            PrintWriter pw = response.getWriter();
            System.out.println("SingletoServletDemo Object:
"+this+"..."+this.hashCode());
            System.out.println("Current     Thread Object:
"+Thread.currentThread()+"..."+

      Thread.currentThread().hashCode()+"..."+Thread.currentThrea
d().getName());
            System.out.println("ServletRequest     Object:
"+request+"..."+request.hashCode());
            System.out.println("ServletResponse    Object:
"+response+"..."+response.hashCode());

            try {
                  Thread.sleep(60000);
            } catch (InterruptedException e) {
                  e.printStackTrace();
            }
            pw.print("<html>");
            pw.print("<body>");
            pw.println("SingletoServletDemo Object:
"+this+"..."+this.hashCode());
            pw.print("<br/>");
            pw.println("Current    Thread Object:
"+Thread.currentThread()+"..."+

      Thread.currentThread().hashCode()+"..."+Thread.currentThrea
d().getName());
            pw.print("<br/>");
            pw.println("ServletRequest     Object:
"+request+"..."+request.hashCode());
            pw.print("<br/>");
            pw.println("ServletResponse    Object:
"+response+"...."+response.hashCode());

            pw.print("</body>");
            pw.print("</html>");
      }

}
```

Web.xml:

```
<web-app>
	<servlet>
		<servlet-name>single</servlet-name>
		<servlet-
class>com.aj.ram.SingletonServletDemo</servlet-class>
	</servlet>

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

Servlet Execution Models:

Servlet execution depends two types of models.

1. **Single Instance – Multi Thread Model.**
   In the above model servlet container will create only one instance for our servlet class. The same instance/object will provides services to all requests or multiple requests.
   All requests are executing simultaneously by the single instance.

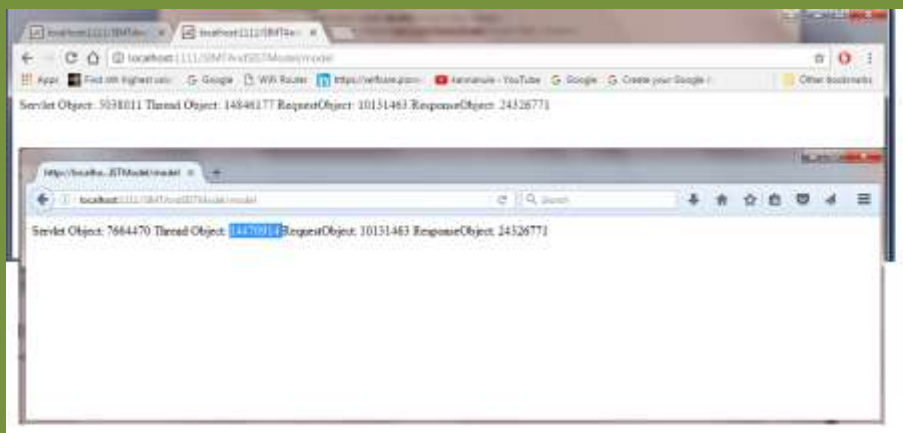
2. **Single Instance – Single Thread Model.**
   In above model servlet container will create multiple servlet objects for multiple requests.
   To obtain this feature we implements javax.servlet.SingleThreadModel interface. It is marker or tagling interface. It provides the permissions to develop SIST model.
   It doesn't have any method.
   For each and every request one new object will be creating, that object used to processing the only one client request.

   If keep on requests are increases, the number of objects will be increases that will effects to performance of an application.
   No two threads working on same service () concurrently.
   Whenever we implements SingleThreadModel interface internally servlet container uses synchronized concept to allow only one thread at a time to execute service ().

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

```
import java.io.PrintWriter;

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


public class ServletModelDemo extends GenericServlet implements
SingleThreadModel{
          public void service(ServletRequest request, ServletResponse
response)
          throws ServletException,IOException{
               response.setContentType("text/html");
               PrintWriter out = response.getWriter();
               out.println("<html>");
               out.println("<body>-");
               try{
                    Thread.sleep(30000);
               }
               catch(InterruptedException e){
                    System.out.println("problem comes here");
                    e.printStackTrace();
               }
               out.println("Servlet Object: "+this.hashCode());
               out.println("Thread  Object: "+
                         Thread.currentThread().hashCode());
               out.println("RequestObject: "+request.hashCode());
               out.println("ResponseObject: "+response.hashCode());
               out.println("</body>");
               out.println("</html>");

          }

}

<web-app>
     <servlet>
          <servlet-name>model</servlet-name>
          <servlet-class>ServletModelDemo</servlet-class>
     </servlet>
     <servlet-mapping>
          <servlet-name>model</servlet-name>
          <url-pattern>/model</url-pattern>
     </servlet-mapping>
```

</web-app>



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

import javax.servlet.GenericServlet;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class WelcomeServlet extends GenericServlet{
    public void service(ServletRequest
request,ServletResponse response)
                throws ServletException,IOException{
                        response.setContentType("text/html");
                        PrintWriter out = response.getWriter();
                        out.println("<body bgcolor='yellow'>");
                        out.println("<h1>THIS IS SERVLET
RESPONSE</H1>");
                        out.println("</body>");
                        try{
                                System.out.println("try block");
                                out.println("current
object:"+this.hashCode());
```

```
                        out.println("thread object
:"+Thread.currentThread().hashCode());

   out.println("ServletRequest:"+request);
                        out.println("ServletResponse:
"+response);
                        out.println("PrintWriter: "+out);
                        Thread.sleep(10000);
                    }
                    catch(InterruptedException ie){
                        System.out.println("catch block");
                        ie.printStackTrace();
                    }
                }
}
```

Execution: making request from two browsers:

Output:

```
current object:1558299375
thread object :1450446715
ServletRequest:org.apache.catalina.connector.RequestFacad
e@58b3ebc5
 ServletResponse:
org.apache.catalina.connector.ResponseFacade@5c8c640b
 PrintWriter:
org.apache.catalina.connector.CoyoteWriter@198ee1fe

current object:1558299375
 thread object :728287330
ServletRequest:org.apache.catalina.connector.RequestFacad
e@24dbc169
ServletResponse:
org.apache.catalina.connector.ResponseFacade@7a976a0a
PrintWriter:
org.apache.catalina.connector.CoyoteWriter@28b3290d
```

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;
import javax.servlet.GenericServlet;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
```

```
import javax.servlet.ServletResponse;
public class WelcomeServlet extends GenericServlet{
     public void service(ServletRequest request,ServletResponse
response)
                throws ServletException,IOException{
                     response.setContentType("text/html");
                     PrintWriter out = response.getWriter();
                     out.println("<body bgcolor='yellow'>");
                     out.println("<h1>THIS IS SERVLET RESPONSE</H1>");
                     out.println("</body>");
                     try{
                          System.out.println("try block");
                          out.println("current
object:"+this.hashCode()+"<br/>");
                          out.println("thread object
:"+Thread.currentThread().hashCode()+"<br/>");

     out.println("ServletRequest:"+request.hashCode()+"<br/>");
                          out.println("ServletResponse:
"+response.hashCode()+"<br/>");
                          out.println("PrintWriter:
"+out.hashCode()+"<br/>");
                          Thread.sleep(10000);
                          Date d = new Date();
                          out.println("Response time is: "+d);

                     }
                     catch(InterruptedException ie){
                          System.out.println("catch block");
                          ie.printStackTrace();
                     }
                }
        }
```

Eclipse Browser:

THIS IS SERVLET RESPONSE
current object:1436977784
thread object :1548666826
ServletRequest:770424710

ServletResponse: 231101396
PrintWriter: 1817374046
Response time is: Mon Nov 13 17:22:10 IST 2017

Outside of eclipse browser:

THIS IS SERVLET RESPONSE
current object:1436977784
thread object :2086127640
ServletRequest:2020102986
ServletResponse: 1030406691
PrintWriter: 1321757788
Response time is: Mon Nov 13 17:22:13 IST 2017

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Date;
import javax.servlet.GenericServlet;
import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class WelcomeServlet extends GenericServlet{

    ServletConfig sconfig = null;
    ServletContext scontext = null;
    @Override
    public void init(ServletConfig sc)throws ServletException{
          sconfig = sc;
          scontext = sconfig.getServletContext();
    }

    public void service(ServletRequest request,ServletResponse
response)
                throws ServletException,IOException{
                      response.setContentType("text/html");
                      PrintWriter out = response.getWriter();
                      out.println("<body bgcolor='yellow'>");
                      out.println("<h1>THIS IS SERVLET
RESPONSE</H1>");
                      out.println("</body>");
                      try{
```

```
                                System.out.println("try block");
                                out.println("current
        object:"+this.hashCode()+"<br/>");
                                out.println("thread object
        :"+Thread.currentThread().hashCode()+"<br/>");

            out.println("ServletRequest:"+request.hashCode()+"<br/>");
                                out.println("ServletResponse:
        "+response.hashCode()+"<br/>");
                                out.println("PrintWriter:
        "+out.hashCode()+"<br/>");
                                out.println("ServletConfig:
        "+sconfig+"<br/>");
                                out.println("servletContext:
        "+scontext+"<br/>");
                                Thread.sleep(10000);
                                Date d = new Date();
                            }
                            catch(InterruptedException ie){
                                System.out.println("catch block");
                                ie.printStackTrace();
                            }
                        }
        }
```

Servlet Life Cycle:

The activities which are doing by servlet container from loading to destroy on servlet is called servlet life cycle.
There are 5 Life cycle phases available in Servlet.

1. **Loading phase.**
2. **Instantiation phase.**
3. **Initialization phase.**
4. **Servicing phase.**
5. **Destroy phase.**

Loading Phase:

Whenever end user sending request to servlet, first servlet container will loads servlet class bytecode from secondary memory to primary memory. In the meanwhile static variable and static block static method are executing.

Instantiation phase:

In this phase servlet container will creates object for our own servlet class. Meanwhile of object creation for our servlet class non-static/instance variables, instance methods and instance blocks will be executing and construcotr will be executing.

Initialization phase:

In above phase we don't fill with initialization values in to servlet object. That problem will be solved in this step. Servlet container itself reading initialization values from web.xml file and creates ServletConfig object and filled those initialization values and calling the init (ServletConfig sc) by passing ServletConfig object.

Servicing phase:

Servlet container will creates ServletRequest object and ServletResponse object and calling service (-,-) by passing request and response objects by using servlet object witch is created in instantiation phase.

Destroy phase:

In phase servlet object will be destroy by calling destroy(). Whenever our server shutdown or restart or project undeploy from the server this phase be executing by the container.

We have 3 life cycle methods.

1. **init (-)**
2. **service(-,-)**
3. **destroy()**

The above methods automatically call by the container.
The methods which are calls by the container automatically are called life cycle methods.

How many ways can we develop our own servlet class?
There are three ways.

1. **By implementing javax.servlet.Servlet interface**
2. **By extends javax.servlet.GenericServlet class**
3. **By extends javax.servlet.HttpServlet class**

Developing our own servlet By implementing javax.servlet.Servlet interface:

```
import java.io.IOException;

import javax.servlet.Servlet;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class LifeCycleServletDemo implements Servlet{
	static int a = m1();
	static int m1(){
		System.out.println("static m1 method");
		System.out.println("a: "+0);
		return 111;
	}
	static{
		System.out.println("static blocks");
	}
	int b = m2();
	int m2(){
		System.out.println("non-static m2 method");
		System.out.println("b: "+0);
		return 222;
	}
	{
		System.out.println("non-static blocks");
	}
	@Override
	public void init(ServletConfig arg0) throws ServletException {
		System.out.println("init method");

	}
	@Override
	public ServletConfig getServletConfig() {
		System.out.println("getServletConfig method");
		return null;
	}

	@Override
	public String getServletInfo() {
		System.out.println("getServletInfo method");
		return null;
```

```
	}
	@Override
	public void service(ServletRequest arg0, ServletResponse arg1)
			throws ServletException, IOException {
		System.out.println("service method");
	}
	@Override
	public void destroy() {
		System.out.println("destroy method");

	}
}
```

Drawback: The drawback of program is we are always overrides the all the methods of javax.servlet.Servlet interface.
But when we work with Generic and HttpServlet we are not facing this type problem.

web.xml:

```
<web-app>
	<servlet>
		<servlet-name>life</servlet-name>
		<servlet-class>LifeCycleServletDemo</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>life</servlet-name>
		<url-pattern>/life</url-pattern>
	</servlet-mapping>
</web-app>
```

<load-on-startup>:

Whenever end user make a request to servlet first time servlet container will process the loading, instantiation, initialization later service phases will executing.

But whenever any end user make a request to servlet in second time, servlet container not executing loading, instantiation and initialization phase, control directly goes to service () and given response to end user that means second user will get response within the less time comparing to first end user.

We can resolve above problem with support of <load-on-startup>.

This tag we can use within <servlet> and after <servlet-class>.

This tag always requires integer number, may be the number is 0 or negative or positive integer.

If we are writing other than integer number like float and String values we will get SAXParseException and server is not starting.

If there is any problem in xml we are always getting SAXParseException.

Whenever we use <load-on-startup>, servlet container will do the loading phase, instantiation and initialization phase before request coming from end user.

The smallest integer value will give information to servlet container to load that particular class first.

Negative values are not giving information to servlet container to load that particular servlet class before request coming from end user.

Two servlet classes <load-on-startup> tag values are same then servlet container will take the decision to load any one of the servlet class.

Example on <load-on-startup>:

```
import java.io.IOException;

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


public class FirstServlet extends GenericServlet{
     static int a = m1();
     static int m1(){
          System.out.println("static m1 method");
          System.out.println("a: "+0);
```

```
		return 111;
	}
	static{
		System.out.println("static blocks");
	}
	public FirstServlet() {
		System.out.println("FirstServlet constructor");
	}
	int b = m2();
	int m2(){
		System.out.println("non-static m2 method");
		System.out.println("b: "+0);
		return 222;
	}
	{
		System.out.println("non-static blocks");
	}
	@Override
	public void init(ServletConfig arg0) throws ServletException {
		System.out.println("init method");
		System.out.println("----------------------");

	}
	@Override
	public void service(ServletRequest arg0, ServletResponse arg1)
			throws ServletException, IOException {
		System.out.println("service method");

	}
}
```

```
import java.io.IOException;

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


public class SecondServlet extends GenericServlet{

	static int a = m1();
	static int m1(){
		System.out.println("static m1 method");
		System.out.println("a: "+0);
```

```
		return 111;
	}
	static{
		System.out.println("static blocks");
	}
	public SecondServlet() {
		System.out.println("SercondServlet constructor");
	}
	int b = m2();
	int m2(){
		System.out.println("non-static m2 method");
		System.out.println("b: "+0);
		return 222;
	}
	{
		System.out.println("non-static blocks");
	}
	@Override
	public void init(ServletConfig arg0) throws ServletException {
		System.out.println("init method");
		System.out.println("----------------------");

	}
	@Override
	public void service(ServletRequest arg0, ServletResponse arg1)
			throws ServletException, IOException {
		System.out.println("service method");

	}
}
```

```
import java.io.IOException;

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


public class SecondServlet extends GenericServlet{

	static int a = m1();
	static int m1(){
		System.out.println("static m1 method");
		System.out.println("a: "+0);
```

```
		return 111;
	}
	static{
		System.out.println("static blocks");
	}
	public SecondServlet() {
		System.out.println("SercondServlet constructor");
	}
	int b = m2();
	int m2(){
		System.out.println("non-static m2 method");
		System.out.println("b: "+0);
		return 222;
	}
	{
		System.out.println("non-static blocks");
	}
	@Override
	public void init(ServletConfig arg0) throws ServletException {
		System.out.println("init method");
		System.out.println("----------------------");

	}
	@Override
	public void service(ServletRequest arg0, ServletResponse arg1)
			throws ServletException, IOException {
		System.out.println("service method");

	}
}
```

```
import java.io.IOException;

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


public class SecondServlet extends GenericServlet{

	static int a = m1();
	static int m1(){
		System.out.println("static m1 method");
		System.out.println("a: "+0);
```

```
		return 111;
	}
	static{
		System.out.println("static blocks");
	}
	public SecondServlet() {
		System.out.println("SercondServlet constructor");
	}
	int b = m2();
	int m2(){
		System.out.println("non-static m2 method");
		System.out.println("b: "+0);
		return 222;
	}
	{
		System.out.println("non-static blocks");
	}
	@Override
	public void init(ServletConfig arg0) throws ServletException {
		System.out.println("init method");
		System.out.println("----------------------");

	}
	@Override
	public void service(ServletRequest arg0, ServletResponse arg1)
			throws ServletException, IOException {
		System.out.println("service method");

	}
}
```

```
import java.io.IOException;

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


public class FifthServlet extends GenericServlet {
	static int a = m1();
	static int m1(){
		System.out.println("static m1 method");
		System.out.println("a: "+0);
```

```
		return 111;
	}
	static{
		System.out.println("static blocks");
	}
	public FifthServlet() {
		System.out.println("fifthServlet constructor");
	}
	int b = m2();
	int m2(){
		System.out.println("non-static m2 method");
		System.out.println("b: "+0);
		return 222;
	}
	{
		System.out.println("non-static blocks");
	}
	@Override
	public void init(ServletConfig arg0) throws ServletException {
		System.out.println("init method");
		System.out.println("----------------------");

	}
	@Override
	public void service(ServletRequest arg0, ServletResponse arg1)
			throws ServletException, IOException {
		System.out.println("service method");

	}
}
```

Web.xml:

```
<web-app>
	<servlet>
		<servlet-name>fs</servlet-name>
		<servlet-class>FirstServlet</servlet-class>
		<load-on-startup>2</load-on-startup>
	</servlet>
	<servlet>
		<servlet-name>ss</servlet-name>
		<servlet-class>SecondServlet</servlet-class>
		<load-on-startup>2</load-on-startup>
	</servlet>
	<servlet>
		<servlet-name>ts</servlet-name>
```

```
        <servlet-class>ThirdServlet</servlet-class>
        <load-on-startup>2</load-on-startup>
    </servlet>
    <servlet>
        <servlet-name>fos</servlet-name>
        <servlet-class>FourthServlet</servlet-class>
        <load-on-startup>-2</load-on-startup>
    </servlet>
    <servlet>
        <servlet-name>fifth</servlet-name>
        <servlet-class>FifthServlet</servlet-class>

    </servlet>

    <servlet-mapping>
        <servlet-name>fs</servlet-name>
        <url-pattern>/fs</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ss</servlet-name>
        <url-pattern>/ss</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>ts</servlet-name>
        <url-pattern>/ts</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>fos</servlet-name>
        <url-pattern>/fos</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
        <servlet-name>fifth</servlet-name>
        <url-pattern>/fifth</url-pattern>
    </servlet-mapping>
</web-app>
```

```
<web-app>
    <servlet>
        <servlet-name>fs</servlet-name>
        <servlet-class>FirstServlet</servlet-class>

    </servlet>

    <servlet-mapping>
```

```
        <servlet-name>fs</servlet-name>
        <url-pattern>/fs/ss/hai12</url-pattern>
    </servlet-mapping>
    <servlet>
        <servlet-name>fis</servlet-name>
        <servlet-class>FifthServlet</servlet-class>

    </servlet>

    <servlet-mapping>
        <servlet-name>fis</servlet-name>
        <url-pattern>/</url-pattern>
    </servlet-mapping>
    <!-- <welcome-file-list>
        <welcome-file>/fs/ss/hai12</welcome-file>
    </welcome-file-list> -->
</web-app>
```

URL pattern and its types:

With support URL we can able to communicating with our required servlet for getting proper response.

URL is provides security to our web component and also hides the technology, which we were used in developing web component.

If the end user not understands the technology which I was used, he is unable send virus or hacking programs.

That means simply we can say URL provides security from unauthorized end users.

These three types:

1. **Exact match url pattern.**
2. **Directory match url pattern.**
3. **Extension match url pattern.**

Exact match url pattern.

Starts with '/' followed by words or numeric's or special symbols.

In web.xml

<url-pattern>/life</url-pattern>

From browser: /life (valid)
/life1(invalid)
In web.xml:

<url-pattern>/life123</url-pattern>

From browser: /life123(valid)
/life (invalid)

In web.xml:

<url-pattern>/life/mylife/123</url-pattern>

From browser: /life/mylife/123 (valid)
/life1/mylife(invalid)

In web.xml:

<url-pattern>/678ram</url-pattern>
Fromm browser: /678ram

<u>Directory match url patten:</u>
Start with / and ends with *
In web.xml:

<url-patten>/life/sb/*</url-pattern>

From browser: /life/sb/ram
/life/sb/sam
/life/sb/A.java
/life/sb/A.c
/sb/life/B.cpp(invalid)

<u>Extension match url pattern:</u>

Starts with * and ends with .extension

In web.xml:

<url-patten>*.java</url-pattern>
From browser: /ram.java

/sam.java
/ram/sam.java
/ram/sa/123.java

'*' can be replaced with single word or multiple words but extension must be match with ".java".

How to develop different types of response:

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class MSWordServletDemo extends GenericServlet {
     @Override
     public void service(ServletRequest req,
                 ServletResponse res)throws
ServletException,IOException{
          res.setContentType("application/msword");
          PrintWriter out = res.getWriter();
          out.println("<html>");
          out.println("<body>");
          out.println("<table border='2'>");

     out.println("<tr><th>FacultyName</th><th>Subjects</th></tr>");
          out.println("<tr><td>Ram</td><td>AdvancedJava</td></tr>");
          out.println("<tr><td>Ravi</td><td>Oracle</td></tr>");
          out.println("<tr><td>SathishB</td><td>Spring</td></tr>");

     out.println("<tr><td>Natraz</td><td>AllJavaCourses</td></tr>");
          out.println("</table>");
          out.println("</body>");
          out.println("</html>");
     }
}
```

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

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

```
public class MSExcelServletDemo extends GenericServlet {
	@Override
	public void service(ServletRequest req,
			ServletResponse res)throws
ServletException,IOException{
		res.setContentType("application/vnd.ms-excel");
		PrintWriter out = res.getWriter();
		out.println("<html>");
		out.println("<body>");
		out.println("<table border='2'>");

	out.println("<tr><th>FacultyName</th><th>Subjects</th></tr>");
		out.println("<tr><td>Ram</td><td>AdvancedJava</td></tr>");
		out.println("<tr><td>Ravi</td><td>Oracle</td></tr>");
		out.println("<tr><td>SathishB</td><td>Spring</td></tr>");

	out.println("<tr><td>Natraz</td><td>AllJavaCourses</td></tr>");
		out.println("</table>");

		out.println("</body>");
		out.println("</html>");
	}

}
```

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

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


public class XMLServletDemo extends GenericServlet{
	@Override
	public void service(ServletRequest req,
			ServletResponse res)throws
ServletException,IOException{
		res.setContentType("text/xml");
		PrintWriter out = res.getWriter();
		out.println("<html>");
		out.println("<body>");
		out.println("<table border='2'>");
```

```
		out.println("<tr><th>FacultyName</th><th>Subjects</th></tr>");
			out.println("<tr><td>Ram</td><td>AdvancedJava</td></tr>");
			out.println("<tr><td>Ravi</td><td>Oracle</td></tr>");
			out.println("<tr><td>SathishB</td><td>Spring</td></tr>");

		out.println("<tr><td>Natraz</td><td>AllJavaCourses</td></tr>");
			out.println("</table>");

			out.println("</body>");
			out.println("</html>");
		}

}
```

Web.xml:

```
<web-app>
	<servlet>
		<servlet-name>ms</servlet-name>
		<servlet-class>MSWordServletDemo</servlet-class>
	</servlet>

	<servlet-mapping>
		<servlet-name>ms</servlet-name>
		<url-pattern>/ms</url-pattern>
	</servlet-mapping>
	<servlet>
		<servlet-name>xs</servlet-name>
		<servlet-class>XMLServletDemo</servlet-class>
	</servlet>

	<servlet-mapping>
		<servlet-name>xs</servlet-name>
		<url-pattern>/xs</url-pattern>
	</servlet-mapping>
	<servlet>
		<servlet-name>es</servlet-name>
		<servlet-class>MSExcelServletDemo</servlet-class>
	</servlet>

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

Execution: urls:

http://localhost:1111/ResponseTypes/ms
http://localhost:1111/ResponseTypes/xe
http://localhost:1111/ResponseTypes/es

How to develop welcome files in servlet:

Welcome files or Home pages are used to navigate the web pages of website.

These are provides more flexibility to interact with website.

We can develop these files with the support of bellow tag.

```
<welcome-file-list>

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

If we are not writing <welcome-file-list> in web.xml by default web server will check whether index.hmtl is there or not, if exists that index.html will treat as welcome that will be given to end user, if not exists server will check index.jsp if exists server will treat that file as welcome file and handover that file to end user as a welcome file else normal browser will handover to end user.

We can place .html, .jsp and servlet as welcome pages.
Whatever the order we placed the files in <welcome-file> in the same order welcomes files selected.

We can place more one file in <welcome-file> as a welcome page.
But at a time only one page will treated as welcome page.

If we are not placing <welcome-file-list> by default control
Will check about index.html if not exist index.jsp.

In web.xml file servlet url pattern and .html file name is then first preference gives to Servlet only.
In web.xml file servlet url pattern and .jsp file name is then first preference gives to Servlet only.

```
<web-app>
	<servlet>
		<servlet-name>ws</servlet-name>
		<servlet-class>WelcomeFileServlet</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>ws</servlet-name>
		<url-pattern>/index.jsp</url-pattern>
	</servlet-mapping>
	<welcome-file-list>
		<welcome-file>index.jsp</welcome-file>
	</welcome-file-list>
</web-app>
```

In above web.xml servlet url name welcome file name same In that time first preference will gives to servlet only not for .jsp file. Between servlet and .html if same problem coming container will give the preference to servlet only.

What is the difference between welcome servlet and default servlet??

Welcome servlet will display as a home page to end user. Whenever end user sending wrong url, container will display 404 Error on the browser, instead of printing 404 error, if we want to display any standard servlet as a response to end user, we should go for default servlet.

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

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


public class WelcomeFileServlet extends GenericServlet{
	@Override
```

```
	public void service(ServletRequest req, ServletResponse
res)throws
	ServletException,IOException{
		System.out.println("service method");
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		out.println("<html>");
		out.println("<body>");
		out.println("welcome file servlet");
		out.println("</body>");
		out.println("</html>");
	}

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

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


public class HelloServletDemo extends GenericServlet{
	@Override
	public void service(ServletRequest req, ServletResponse
res)throws
	ServletException,IOException{
		System.out.println("service method");
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		out.println("<html>");
		out.println("<body>");
		out.println("hellow servlet");
		out.println("</body>");
		out.println("</html>");
	}
}


<web-app>
	<servlet>
		<servlet-name>ws</servlet-name>
		<servlet-class>WelcomeFileServlet</servlet-class>
	</servlet>
```

```
        <servlet-mapping>
                <servlet-name>ws</servlet-name>
                <url-pattern>/ws</url-pattern>
        </servlet-mapping>
        <!-- <servlet>
                <servlet-name>hs</servlet-name>
                <servlet-class>HelloServletDemo</servlet-class>
        </servlet>
        <servlet-mapping>
                <servlet-name>hs</servlet-name>
                <url-pattern>/</url-pattern>
        </servlet-mapping>  -->
        <welcome-file-list>
                <welcome-file>ws</welcome-file>
        </welcome-file-list>
</web-app>
```

If we want to work with default servlet delete <welcome-file-list> and uncomment above code.

Important object of servlet:

ServletContext
ServletRequest
ServletResponse
PrintWriter
ServletConfig
Servlet object
HttpSession

The order of above objects creation will be depends on <load-on-startup>.

With <load-on-startup>
1. ServletContext

2. ServletConfig
3. Servlet object

4.ServletRequest
5.ServletResponse
6.PrintWriter

7.HttpSession

Without <load-on-startup>
1. ServletContext

2.ServletRequest
3.ServletResponse
4.PrintWriter

5. ServletConfig
6. Servlet object

7.HttpSession

Scope Objects:

These are used for carrying the information from one request to another or one servlet to another servlet or one client to another client are called Scope objects.

We have three scope objects.

1. **Request**
2. **Session**
3. **Context /application(in jsp)**

Example on Request Scope Objects:

If the response is handover to end user or browser automatically the request scope object will destroyed.

To place the data into request scope object we required the following method.

request.setAttribute(-,-)

To read the data from request scope object we required the following method.

request.getAttribute(-)

```
import java.io.IOException;
import javax.servlet.GenericServlet;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class AtmServlet extends GenericServlet {
```

```
	@Override
	public void service(ServletRequest request,
			ServletResponse response)
	throws ServletException,IOException{
		request.setAttribute("accNo", 1234567);
		RequestDispatcher rd =
				request.getRequestDispatcher("/bs");
		rd.forward(request, response);
	}
}

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class BankServlet extends GenericServlet {
	@Override
	public void service(ServletRequest request,
			ServletResponse response)
	throws ServletException,IOException{
		response.setContentType("text/html");
		PrintWriter out = response.getWriter();
		Integer accNo =
				(Integer) request.getAttribute("accNo");
		out.println("<html>");
		out.println("<body>");
		out.println("Account Number is: "+accNo);
		out.println("</body>");
		out.println("</html>");
	}
}

<web-app>
	<servlet>
		<servlet-name>fs</servlet-name>
		<servlet-class>AtmServlet</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>fs</servlet-name>
		<url-pattern>/fs</url-pattern>
	</servlet-mapping>
	<servlet>
```

```
        <servlet-name>bs</servlet-name>
        <servlet-class>BankServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>bs</servlet-name>
        <url-pattern>/bs</url-pattern>
    </servlet-mapping>
</web-app>
```

Session Object:

Whenever end user making request to Server, servlet container will create request, response, session objects.
We can able to interface with request and response objects directly but not session object.

To interact with session object we request the below syntax or code

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

For own browser, servlet container will create one session object. The same session object content will be used by different servlets in our application.

Once the browser is closed, automatically session object will be destroyed.

```
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 AtmServlet extends HttpServlet {
    @Override
    public void service(HttpServletRequest request,
                HttpServletResponse response)
    throws ServletException,IOException{
        HttpSession session = request.getSession();
        session.setAttribute("accNo", 777777);
        RequestDispatcher rd =
                    request.getRequestDispatcher("/bs");
        rd.forward(request, response);
```

```
	}
}

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

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.http.HttpSession;
public class BankServlet extends HttpServlet {
	@Override
	public void service(HttpServletRequest request,
			HttpServletResponse response)
	throws ServletException,IOException{
		response.setContentType("text/html");
		PrintWriter out = response.getWriter();
		HttpSession session = request.getSession();
		Integer accNo =
				(Integer) session.getAttribute("accNo");
		out.println("<html>");
		out.println("<body>");
		out.println("Account Number is: "+accNo);
		out.println("</body>");
		out.println("</html>");
	}
}

Web.xml:
<web-app>
	<servlet>
		<servlet-name>fs</servlet-name>
		<servlet-class>AtmServlet</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>fs</servlet-name>
		<url-pattern>/fs</url-pattern>
	</servlet-mapping>
	<servlet>
		<servlet-name>bs</servlet-name>
		<servlet-class>BankServlet</servlet-class>
	</servlet>
	<servlet-mapping>
```

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

Context Object:

```
import java.io.IOException;

import javax.servlet.GenericServlet;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class AtmServlet extends GenericServlet {
    @Override
    public void service(ServletRequest request,
            ServletResponse response)
    throws ServletException,IOException{

        ServletContext context = getServletContext();
        context.setAttribute("accNo", 123456);
        RequestDispatcher rd =
                request.getRequestDispatcher("/bs");
        rd.forward(request, response);
    }
}
import java.io.IOException;
import java.io.PrintWriter;

import javax.servlet.GenericServlet;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class BankServlet extends GenericServlet {
    @Override
    public void service(ServletRequest request,
            ServletResponse response)
    throws ServletException,IOException{
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();

        ServletContext context = getServletContext();
```

```
            Integer accNo =
                        (Integer) context.getAttribute("accNo");
            out.println("<html>");
            out.println("<body>");
            out.println("Account Number is: "+accNo);
            out.println("</body>");
            out.println("</html>");
      }
}
```

```
<web-app>
      <servlet>
            <servlet-name>fs</servlet-name>
            <servlet-class>AtmServlet</servlet-class>
      </servlet>
      <servlet-mapping>
            <servlet-name>fs</servlet-name>
            <url-pattern>/fs</url-pattern>
      </servlet-mapping>
      <servlet>
            <servlet-name>bs</servlet-name>
            <servlet-class>BankServlet</servlet-class>
      </servlet>
      <servlet-mapping>
            <servlet-name>bs</servlet-name>
            <url-pattern>/bs</url-pattern>
      </servlet-mapping>
</web-app>
```

Html and Servlet communication:

1. Sending data directly from browser URL:

```
2. import java.io.IOException;
3. import java.io.PrintWriter;
4.
5. import javax.servlet.GenericServlet;
6. import javax.servlet.ServletException;
7. import javax.servlet.ServletRequest;
8. import javax.servlet.ServletResponse;
9.
10.
11.     public class AddingServletDemo extends GenericServlet {
12.           @Override
13.           public void service(ServletRequest request,
14.                       ServletResponse response)
```

```
15.          throws ServletException,IOException{
16.               System.out.println("service method");
17.               String firstNumber = request.getParameter("fno");
18.               //case sensitive means url fno this fno must be
     match
19.               String secondNumber =
     request.getParameter("sno");
20.
21.               int fno = Integer.parseInt(firstNumber);
22.               int sno = Integer.parseInt(secondNumber);
23.
24.               int result = fno+sno;
25.
26.               response.setContentType("text/html");
27.               PrintWriter out = response.getWriter();
28.               out.println("Result Is: "+result);
29.          }
30.     }
```

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

From browser Side:

We need to enter data in the following format:

http://localhost:portnumber/projectname/url?firstparamname=paramvalues&secondparamname=paramvalue&.....

http://localhost:1111/HtmlServletCommunication/add?fno=200&sno=150

Output in the browser is:

Result Is: 350

“add” → is URL name it must be match with <url-pattern> in web.xml.

“?” → is used for separate the URL from input parameter name.

“=” → is used for separate parameter name and value.

“&” → is used for separate one parameter to another parameter.

Note: don’t use any other special character and spaces in the

URL, if we type wrongly we get errors.

In the above approach end user unable type URL in the correct format.

To resolve this problem we should for html pages.

How to create .html pages in eclipse:

Right click on project WebContent folder → click on new → click on file/click on others/ search by typing file → select File option →click on next → type file name like add.html → click on finish.

add.html:

```
<html>
    <body>
        <form action="./add">
    Enter First Number : <input type='text' name='fno'/><br/>
    Enter Second Number: <input type='text' name='sno'/><br/>
                                <input type='submit' value='ADD'/>
        </form>
    </body>
    </html>
```

AddingServletDemo:

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

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


public class AddingServletDemo extends GenericServlet {
    @Override
    public void service(ServletRequest request,
            ServletResponse response)
    throws ServletException,IOException{
        System.out.println("service method");
        String firstNumber = request.getParameter("fno");
        //case sensitive means url fno this fno must be match
        String secondNumber = request.getParameter("sno");
        int fno = Integer.parseInt(firstNumber);
        int sno = Integer.parseInt(secondNumber);
        int result = fno+sno;
        response.setContentType("text/html");
```

```
            PrintWriter out = response.getWriter();
            out.println("Result Is: "+result);
      }
}


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

From browser send request like bellow:

http://localhost:1111/HtmlServletCommunication/add.html
We will get add.html page
Later enter first number value is 500 and second number value is 200 and click ADD button
After click on ADD button browser engine will convert above URL into following manner
http://localhost:1111/HtmlServletCommunication/add?fno=500&sno=200
on browser we will get output like

      Result Is: 700

In the above approach also there is one small drawback that is Every time end user click on back arrow button or end user will make new request from brower url to add.html for entering new values for adding.
To avoiding above problem we should use hyperlink.

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

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


      public class AddingServletDemo extends GenericServlet {
            @Override
            public void service(ServletRequest request,
```

```
                    ServletResponse response)
          throws ServletException,IOException{
               System.out.println("service method");
               String firstNumber = request.getParameter("fno");
               //case sensitive means url fno this fno must be match
               String secondNumber = request.getParameter("sno");
               int fno = Integer.parseInt(firstNumber);
               int sno = Integer.parseInt(secondNumber);
               int result = fno+sno;
               response.setContentType("text/html");
               PrintWriter out = response.getWriter();
               out.println("<a href='./add.html'>click me</a>");
               out.println("Result Is: "+result);
          }
     }
```

In above approach end user every time click on “click me” hyperlink.

Sometimes may be end user not showing interest to click on hyperlink.

To resolve this problem we should add bellow code in the place of hyperlink code.

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

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


public class AddingServletDemo extends GenericServlet {
     @Override
     public void service(ServletRequest request,
                    ServletResponse response)
          throws ServletException,IOException{
               System.out.println("service method");
               String firstNumber = request.getParameter("fno");
               //case sensitive means url fno this fno must be match
               String secondNumber = request.getParameter("sno");
               int fno = Integer.parseInt(firstNumber);
               int sno = Integer.parseInt(secondNumber);
```

```
            int result = fno+sno;
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            //out.println("<a href='./add.html'>click me</a>");
            RequestDispatcher rd =
request.getRequestDispatcher("./add.html");
            rd.include(request, response);
            out.println("Result Is: "+result);
      }
   }
```

In the above approach old values are not display in the text fields.

Whatever values we entered in request, if we want to display those values in the text field we need to write the following code in the place of RequestDispatcher in the previous program.

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

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


   public class AddingServletDemo extends GenericServlet {
        @Override
        public void service(ServletRequest request,
                    ServletResponse response)
        throws ServletException,IOException{
            System.out.println("service method");
            String firstNumber = request.getParameter("fno");
            //case sensitive means url fno this fno must be match
            String secondNumber = request.getParameter("sno");
            int fno = Integer.parseInt(firstNumber);
            int sno = Integer.parseInt(secondNumber);
            int result = fno+sno;
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();

            out.println("<html>");
            out.println("<body>");
            out.println("Enter First  Number: <input type='text'
name='fno' value='"+fno+"'/><br/>");
```

```
            out.println("Enter Second Number: <input type='text'
name='sno' value='"+sno+"'/><br/>");
            out.println("<input type='submit' value='add'>");
            out.println("Result Is: "+result);
            out.println("</body>");
            out.println("</html>");
      }
   }
```

In above program if the end user sending request to AddingServletDemo directly we face one exception java.lang.NumberFormatException

To overcome this problem we need to write the following code.

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

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


public class AddingServletDemo extends GenericServlet {
     @Override
     public void service(ServletRequest request,
               ServletResponse response)
     throws ServletException,IOException{
          System.out.println("service method");

          response.setContentType("text/html");
          PrintWriter out = response.getWriter();

          String firstNumber = request.getParameter("fno");
          String secondNumber = request.getParameter("sno");

          System.out.println("firstnumber: "+firstNumber);
          System.out.println("secondnumber: "+secondNumber);
          int result=0;
          if( (firstNumber == null || secondNumber ==null) ){
               firstNumber="";
               secondNumber="";
          }
          else{
               int fno = Integer.parseInt(firstNumber);//""
```

```
                    int sno = Integer.parseInt(secondNumber);//"200"
                    result = fno+sno;//300
                }
                out.println("<form action='./add'>");
                out.println("Enter First Number: <input type='text'
   name='fno'  /></br>");
                out.println("Enter second number: <input type='text'
   name='sno' /></br>");
                out.println("<input type='submit' value='add'/>");
                out.println("</form>");
                if(!(firstNumber.isEmpty())){
                    out.println("result: "+result);
                }
            }
      }
```

If the end user not inserting any data into text fields and click on Add button will get again java.lang.NumberFormatException to resolve this problem, we can use HTML5 attribute like "required" in text field declaration in AddingServletDemo.java file we can re write the code as bellow.

```
            out.println("<form action='./add'>");
            out.println("Enter First Number: <input type='text'
                        required  name='fno'  /></br>");
            out.println("Enter second number: <input type='text'
                        required   name='sno' /></br>");
            out.println("<input type='submit' value='add'/>");
            out.println("</form>");
```

Calculation Application:

cal.html:

```
<form>
    <a href='./add.html'>ADDITION</a><br/>
    <a href='./sub.html'>SUBTRACTION</a><br/>
    <a href='./mul.html'>MULTIPLICATION</a><br/>
    <a href='./div.html'>DIVISION</a><br/>
</form>
```

add.html:

```
<form action='./add'>
```

```
	Enter First Number: <input type='text' name='fno'/><br/>
	Enter Second Number: <input type='text' name='sno'/><br/>
	<input type='submit' value='add'/>
</form>
```
sub.html:
```
<form action='./sub'>
	Enter First Number: <input type='text' name='fno'/><br/>
	Enter Second Number: <input type='text' name='sno'/><br/>
	<input type='submit' value='sub'/>
</form>
```
mul.html:
```
<form action='./mul'>
	Enter First Number: <input type='text' name='fno'/><br/>
	Enter Second Number: <input type='text' name='sno'/><br/>
	<input type='submit' value='mul'/>
</form>
```
div.html:
```
<form action='./div'>
	Enter First Number: <input type='text' name='fno'/><br/>
	Enter Second Number: <input type='text' name='sno'/><br/>
	<input type='submit' value='div'/>
</form>

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

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


public class AddServlet extends GenericServlet{
	@Override
	public void service(ServletRequest req, ServletResponse res)
			throws ServletException, IOException {
		System.out.println("service method");
		int i = Integer.parseInt(req.getParameter("fno"));
		int j = Integer.parseInt(req.getParameter("sno"));
		int k = i+j;
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		out.println("result is: "+k);
	}
}
```

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

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


public class SubServlet  extends GenericServlet{
     @Override
     public void service(ServletRequest req, ServletResponse res)
               throws ServletException, IOException {
          System.out.println("service method");
          int i = Integer.parseInt(req.getParameter("fno"));
          int j = Integer.parseInt(req.getParameter("sno"));
          int k = i-j;
          res.setContentType("text/html");
          PrintWriter out = res.getWriter();
          out.println("result is: "+k);
     }

}

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

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


public class MulServlet  extends GenericServlet{
   @Override
     public void service(ServletRequest req, ServletResponse res)
               throws ServletException, IOException {
          System.out.println("service method");
          int i = Integer.parseInt(req.getParameter("fno"));
          int j = Integer.parseInt(req.getParameter("sno"));
          int k = i*j;
          res.setContentType("text/html");
          PrintWriter out = res.getWriter();
          out.println("result is: "+k);
     }
```

```
}

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

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


public class DivServlet  extends GenericServlet{
     @Override
     public void service(ServletRequest req, ServletResponse res)
               throws ServletException, IOException {
          System.out.println("service method");
          int i = Integer.parseInt(req.getParameter("fno"));
          int j = Integer.parseInt(req.getParameter("sno"));
          int k = i/j;
          res.setContentType("text/html");
          PrintWriter out = res.getWriter();
          out.println("result is: "+k);
     }

}


<web-app>
     <servlet>
          <servlet-name>add</servlet-name>
          <servlet-class>AddServlet</servlet-class>
     </servlet>
     <servlet-mapping>
          <servlet-name>add</servlet-name>
          <url-pattern>/add</url-pattern>
     </servlet-mapping>
<servlet>
          <servlet-name>sub</servlet-name>
          <servlet-class>SubServlet</servlet-class>
     </servlet>
     <servlet-mapping>
          <servlet-name>sub</servlet-name>
          <url-pattern>/sub</url-pattern>
     </servlet-mapping><servlet>
          <servlet-name>mul</servlet-name>
          <servlet-class>MulServlet</servlet-class>
```

```
    </servlet>
    <servlet-mapping>
        <servlet-name>mul</servlet-name>
        <url-pattern>/mul</url-pattern>
    </servlet-mapping><servlet>
        <servlet-name>div</servlet-name>
        <servlet-class>DivServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>div</servlet-name>
        <url-pattern>/div</url-pattern>
    </servlet-mapping>
</web-app>
```

In the above we are unnecessarily using 4 servlet classes.
Then Servlet container will create objects that means,
Internally memory unnecessarily wasted to avoid this problem
we should go for the following approach.

Servlet API hierarchy:

Upto now we observed, servlet API is using for carrying the data from browser to server and server to browser.

For carrying input and output values servlet technology supplying different objects like bellow.

- ➔ servlet initialization values.
- ➔ request input values
- ➔ response output values
- ➔ attributes values

1. ServletContext and ServletConfig object are meant for carrying the servlet initialization values.
2. ServletRequest and HttpServletRequest objects are meant for carrying request input values.
3. ServletResponse and HttpServletResponse meant for carrying response output values.
4. ServletRequest, HttpSession, ServletContext objects are meant for carry attribute values.
5. RequestDispatcher objects are meant for making a communication between one servlet to another servlet.
6. HttpSession and Cookie meant for carrying a specific client session values.

ServletConfig: It will use to read the initialization values from web.xml file. ServletConfig will read data in the form of key and value pair combination.

To provide the initialization values from web.xml file
We need to use one new tag like <init-param>.
In this tag we have two tags.

1. **<param-name>**
2. **<param-value>**

<init-param> is the sub tag of <servlet>.
We can write multiple <init-param> tags in <servlet>

In the following program we are reading database details from web.xml file by using ServletConfig object.
After that we are making communication with db and inserting one record.

ServletConfig object is not only reading the data related to database (driver,url,system,manager) but also we can able read any type of data from web.xml file.

Servlet1.java

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

import javax.servlet.GenericServlet;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class Servlet1 extends GenericServlet{
     int a;
     int b;
```

```
	@Override
	public void init(ServletConfig sc)throws ServletException{
		System.out.println("servlet1 intit method: ");
		System.out.println("config: "+sc);
		a = Integer.parseInt(sc.getInitParameter("p1"));
		b = Integer.parseInt(sc.getInitParameter("p2"));
	}
	@Override
	public void service(ServletRequest request, ServletResponse
response)
			throws IOException,ServletException{
		response.setContentType("text/html");
		PrintWriter out = response.getWriter();
		out.println("<body bgcolor='yellow'>");
		out.println("The Result is: "+(a+b));
	}
}
```

Web.xml:

```
<web-app>
	<servlet>
		<servlet-name>s1</servlet-name>
		<servlet-class>Servlet1</servlet-class>
		<init-param>
			<param-name>p1</param-name>
			<param-value>100</param-value>
		</init-param>
		<init-param>
			<param-name>p2</param-name>
			<param-value>200</param-value>
		</init-param>
	</servlet>
	<servlet-mapping>
		<servlet-name>s1</servlet-name>
		<url-pattern>/s1</url-pattern>
	</servlet-mapping>
</web-app>
```

ServletConfifDemo.java

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.Statement;

import javax.servlet.GenericServlet;
```

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


public class ServletConfigDemo extends GenericServlet {
	Connection con=null;
	@Override
	public void init(ServletConfig sc)throws ServletException{
		System.out.println("init method");
		String driver = sc.getInitParameter("driver");
		String url = sc.getInitParameter("url");
		String username = sc.getInitParameter("username");
		String password = sc.getInitParameter("password");
		try{
			Class.forName(driver);
	con = DriverManager.getConnection(url,username,password);
			Statement st = con.createStatement();
	int count = st.executeUpdate("insert into empram
			values(444,'nidhi',5757,'nw')");
			System.out.println("count: "+count);
		}
		catch(Exception e){
			e.printStackTrace();
		}
	}

	@Override
	public void service(ServletRequest req,ServletResponse res)
throws ServletException,IOException{
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		if(con!=null){
			System.out.println("con: "+con);
			out.println("connection created");
		}
		else{
			out.println("connection not established");
		}
	}

}
```

Web.xml:

```
<web-app>
	<servlet>
		<servlet-name>sc</servlet-name>
		<servlet-class>ServletConfigDemo</servlet-class>
		<init-param>
			<param-name>driver</param-name>
			<param-value>oracle.jdbc.driver.OracleDriver</param-
value>
		</init-param>
		<init-param>
			<param-name>url</param-name>
			<param-
value>jdbc:oracle:thin:@localhost:1521:xe</param-value>
		</init-param>
		<init-param>
			<param-name>username</param-name>
			<param-value>system</param-value>
		</init-param>
		<init-param>
			<param-name>password</param-name>
			<param-value>manager</param-value>
		</init-param>
	</servlet>
	<servlet-mapping>
		<servlet-name>sc</servlet-name>
		<url-pattern>/sc</url-pattern>
	</servlet-mapping>

</web-app>
```

In the above we are always placing same type of record
If we want place different types of records, we should
Go for following program.

```
<body bgcolor="pink">
	<form action="./sc">
		Enter Employee Number:
<input type="text" name="eno"/><br/>
		Enter Employee Name  :
<input type="text" name="ename"/><br/>
		Enter Employee Salary:
 <input type="text" name="esal"/><br/>
		Enter Employee dept  :
```

```
<input type="text" name="edept"/><br/>
     <input type="submit" value="Enter"/>
     </form>
</body>
```

ServletConfigDemo.java

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

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


public class ServletConfigDemo extends GenericServlet {
     Connection con=null;
     Statement st = null;
     @Override
     public void init(ServletConfig sc)throws ServletException{
          System.out.println("init method");
          String driver = sc.getInitParameter("driver");
          String url = sc.getInitParameter("url");
          String username = sc.getInitParameter("username");
          String password = sc.getInitParameter("password");
          try{
               Class.forName(driver);
               con =
DriverManager.getConnection(url,username,password);

          }
          catch(Exception e){
               e.printStackTrace();
          }
     }

     @Override
     public void service(ServletRequest req,ServletResponse res)throws
     ServletException,IOException{
          res.setContentType("text/html");
```

```
            PrintWriter out = res.getWriter();
            int eno = Integer.parseInt(req.getParameter("eno"));
            String ename = req.getParameter("ename");
            int esal = Integer.parseInt(req.getParameter("esal"));
            String edept = req.getParameter("edept");
            if(con!=null){
                  System.out.println("con: "+con);
                  out.println("connection created");
                  try {
                        st = con.createStatement();
                        int count = st.executeUpdate("insert into empram
values("+eno+",'"+ename+"',"+esal+",'"+edept+"')");
                        System.out.println("count: "+count);
                  } catch (SQLException e) {
                        e.printStackTrace();
                  }

            }
            else{
                  out.println("connection not established");
            }
      }
}
Web.xml:
<web-app>
      <servlet>
            <servlet-name>sc</servlet-name>
            <servlet-class>ServletConfigDemo</servlet-class>
            <init-param>
                  <param-name>driver</param-name>
                  <param-value>oracle.jdbc.driver.OracleDriver</param-
value>
            </init-param>
            <init-param>
                  <param-name>url</param-name>
                  <param-
value>jdbc:oracle:thin:@localhost:1521:xe</param-value>
            </init-param>
            <init-param>
                  <param-name>username</param-name>
                  <param-value>system</param-value>
            </init-param>
            <init-param>
                  <param-name>password</param-name>
                  <param-value>manager</param-value>
            </init-param>
```

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

In above program we are passing initialization values to single servlet individually at a time by using <init-param> tag.

Whatever the data we are giving ServletConfigDemo class that data will be not shared by the remaining servlet classes.

If we want to provide a common data for all servlets At a time, we need write those details in <context-param>.

Whatever data we have in <context-param>, that data will placed into ServletContext object, for one project container will create only one Servletcontext object, this object will be equally shared by the all servlet class with in the project.

Program on ServletContext Object:

```
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
import java.sql.Statement;

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


public class FirstServlet extends GenericServlet {
```

```
	Connection con=null;
	Statement st = null;
	@Override
	public void init(ServletConfig sc)throws ServletException{
		System.out.println("init method");
		ServletContext sc1 = sc.getServletContext();
		String driver = sc1.getInitParameter("driver");
		String url = sc1.getInitParameter("url");
		String username = sc1.getInitParameter("username");
		String password = sc1.getInitParameter("password");
		try{
			Class.forName(driver);//NPE
			con =
DriverManager.getConnection(url,username,password);

		}
		catch(Exception e){
			e.printStackTrace();
		}
	}

	@Override
	public void service(ServletRequest req,ServletResponse
res)throws
	ServletException,IOException{
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		int eno = Integer.parseInt(req.getParameter("eno"));
		String ename = req.getParameter("ename");
		int esal = Integer.parseInt(req.getParameter("esal"));
		String edept = req.getParameter("edept");
		if(con!=null){
			System.out.println("con: "+con);
			out.println("connection created");
			try {
				st = con.createStatement();
				int count = st.executeUpdate("insert into
empram values("+eno+",'"+ename+"',"+esal+",'"+edept+"')");
				System.out.println("count: "+count);
			} catch (SQLException e) {
				e.printStackTrace();
			}

		}
```

```
            else{
                  out.println("connection not established");
            }
      }
}

import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;

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


public class SecondServlet extends GenericServlet {
      Connection con=null;
      Statement st = null;
      @Override
      public void init(ServletConfig sc)throws ServletException{
            System.out.println("init method");
            ServletContext sc1 = sc.getServletContext();
            String driver = sc1.getInitParameter("driver");
            String url = sc1.getInitParameter("url");
            String username = sc1.getInitParameter("username");
            String password = sc1.getInitParameter("password");
            try{
                  Class.forName(driver);
                  con =
DriverManager.getConnection(url,username,password);

            }
            catch(Exception e){
                  e.printStackTrace();
            }
      }
```

```
	@Override
	public void service(ServletRequest req,ServletResponse
res)throws
	ServletException,IOException{
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		if(con!=null){
			System.out.println("con: "+con);
			out.println("connection created");
			try {
				st = con.createStatement();
				ResultSet rs = st.executeQuery("select * from
empram");
				while(rs.next()){
					System.out.println(rs.getInt(1));
					System.out.println(rs.getString(2));
					System.out.println(rs.getInt(3));
					System.out.println(rs.getString(4));
					System.out.println("-----------------");
				}

			} catch (SQLException e) {
				e.printStackTrace();
			}

		}
		else{
			out.println("connection not established");
		}
	}
}
<body bgcolor="pink">
	<form action="./fs">
		Enter Employee Number: <input type="text"
name="eno"/><br/>
		Enter Employee Name  : <input type="text"
name="ename"/><br/>
		Enter Employee Salary: <input type="text"
name="esal"/><br/>
		Enter Employee dept  : <input type="text"
name="edept"/><br/>
							<input type="submit"
value="Enter"/>
	</form>
```

```
</body>
<web-app>
	<context-param>
		<param-name>driver</param-name>
		<param-value>oracle.jdbc.driver.OracleDriver</param-
value>
	</context-param>
	<context-param>
		<param-name>url</param-name>
		<param-
value>jdbc:oracle:thin:@localhost:1521:xe</param-value>
	</context-param>
	<context-param>
		<param-name>username</param-name>
		<param-value>system</param-value>
	</context-param>
	<context-param>
		<param-name>password</param-name>
		<param-value>manager</param-value>
	</context-param>

	<servlet>
		<servlet-name>fs</servlet-name>
		<servlet-class>FirstServlet</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>fs</servlet-name>
		<url-pattern>/fs</url-pattern>
	</servlet-mapping>
	<servlet>
		<servlet-name>ss</servlet-name>
		<servlet-class>SecondServlet</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>ss</servlet-name>
		<url-pattern>/ss</url-pattern>
	</servlet-mapping>
</web-app>
```

Drawbacks of GenericServlet:

When ever client sending the request data, that data will be attached to browser URL and visible to everyone so security problem.

We cannot get http protocol specification like

--> No security.

--> Client state persistence.

--> RequestHeader information.

To overcome that problems we should go for HttpServlet interface.

Http Protocol:

Http is a communication channel between web client (browser) and web server (tomcat) for data exchanging.

Http is a stateless protocol.

The meaning of stateless protocol is once response is committed, it doesn't remember old client (request).

If we are sending one more request to server, http will treat the same client as new client.

If we are using http, every request will be treated as new client.

Http is a specification.

That means it will provide rules and guidelines to web client and web server.

Set of rules for web client development is called Httpclientspecification.

Set of rules for httpserver development is called httpserverspecification.

httpclientspecification describes the following responsibilities to web client.

1. preparing http request with user input data.

2. sending the request to server.

3. reading output from httpresonse object and handover to enduser.

httpserverspecification describes the follwing responsibilities to httpserver

1. reading the data from webclient and converting into httpservletrequest object.

2. finding the request resource.

3. and executing service(-,-) method by sending httpservletrequest and response object.

4. preparing httpresonse object

5. finally handover to webclient.

Sample program on HttpServlet:

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class HttpServletDemo extends HttpServlet {
	@Override
	public void doPut(HttpServletRequest req,
			HttpServletResponse res)
		throws ServletException,IOException{
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		out.println("This http servlet program");
	}
}
```

```
<web-app>
	<servlet>
		<servlet-name>hs</servlet-name>
		<servlet-class>HttpServletDemo</servlet-class>

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

```
<body bgcolor='red'>
<form action="./hs" method='get'>
	<input type='submit' value='click'/>

</form>
	</body>
```

Servlet Container always calls first public service(-,-).
If not avialable it will calls protected service(-,-).
If not available it will doGet(-,-).
doPut(- ,-),doHead(- ,-),doDelete(- ,-),doOptions(- ,-),doTrace(- ,-) methods will calls doGet(-,-).

How many way can we communicating with doPost()?
Only one way.
That is <form> method attribute.
Ex:
<form action='./hs' method="POST">
Whenever we writing like
<form action='./hs' method='get/post'>
control goes to doGet(- ,-).
The main difference between doGet() and doPost()
is data will be add browser url bar.
there may be a change loss of security, to overcome
this problem we will go for doPost().

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

import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class HttpServletDemo extends HttpServlet {
	/*@Override
	public void service (ServletRequest req,
			ServletResponse res)
		throws ServletException,IOException{
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		out.println("This is HttpServlet-public service");
	}
	@Override
```

```
        protected void service (HttpServletRequest req,
                        HttpServletResponse res)
                throws ServletException,IOException{
                res.setContentType("text/html");
                PrintWriter out = res.getWriter();
                out.println("This is HttpServlet-protected service");
        }*/
        @Override
        protected void doGet (HttpServletRequest req,
                        HttpServletResponse res)
                throws ServletException,IOException{
                res.setContentType("text/html");
                PrintWriter out = res.getWriter();
                out.println("This is HttpServlet-doGet()");
        }
        @Override
        protected void doPost (HttpServletRequest req,
                        HttpServletResponse res)
                throws ServletException,IOException{
                res.setContentType("text/html");
                PrintWriter out = res.getWriter();
                out.println("This is HttpServlet-doPost()");
        }
}
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;

//@WebServlet("/as")
public class AddServlet extends HttpServlet {
        /*@Override
        public void doPost (HttpServletRequest req,
                        HttpServletResponse res)
                throws ServletException,IOException{
                res.setContentType("text/html");
                PrintWriter out = res.getWriter();
                int fno =
Integer.parseInt(req.getParameter("fno"));
```

```
                int sno =
   Integer.parseInt(req.getParameter("sno"));
                int result = fno+sno;
                out.println("This is HttpServlet-doPost <br/>");
                out.println("Result: "+result);*/
          @Override
          public void doGet (HttpServletRequest req,
                     HttpServletResponse res)
                throws ServletException,IOException{
                res.setContentType("text/html");
                PrintWriter out = res.getWriter();
                int fno =
   Integer.parseInt(req.getParameter("fno"));
                int sno =
   Integer.parseInt(req.getParameter("sno"));
                int result = fno+sno;
                out.println("This is HttpServlet-doGet() <br/>");
                out.println("Result: "+result);
                Enumeration headerNames =
   req.getHeaderNames();
                while(headerNames.hasMoreElements()){
                     String headerName = (String)
   headerNames.nextElement();
                     out.println(headerName+"<br/>");
                     String headerValue =
   req.getHeader(headerName);
                     out.println(headerValue+"<br/>");

                }
          }
}<!-- <body bgcolor='red'>
<form action="./hs" method='get'>
     Enter First No: <input type='text' name="fno"/>
     Enter Second No: <input type='text' name="sno"/>
     <input type='submit' value='add'/>
</form>
</body> -->



<body bgcolor='red'>
<form action="./hs" method='get'>
     fno: <input type='text' name="fno"/>
     sno: <input type='text' name="sno"/>
```

```
     <input type='submit' value='add'/>
</form>
</body>



<!-- <body bgcolor='red'>
<form action="./hs" method='put/get' >
     <input type='submit' value='click'/>
</form> -->
<!-- <body bgcolor='red'>
<form action="./hs" method='get/put' >
     <input type='submit' value='click'/>
</form> -->
</body><!-- <body bgcolor='pink'>
<form action="./hs" method='put' >
     <input type='submit' value='click'/>
</form>
</body> -->
<!-- <body bgcolor='pink'>
<form action="./hs" method='post' >
     <input type='submit' value='click'/>
</form>
</body> -->
<!-- <body bgcolor='red'>
<form action="./hs" method='get'>
     fno: <input type='text' name="fno"/>
     sno: <input type='text' name="sno"/>

     <input type='submit' value='click'/>
</form>
</body> -->

<!-- <body bgcolor='pink'>
<form action="./hs" method='post'>
     fno: <input type='text' name="fno"/>
     sno: <input type='text' name="sno"/>

     <input type='submit' value='click'/>
</form>
     </body> -->

<web-app>
     <servlet>
          <servlet-name>hs</servlet-name>
```

```
            <servlet-class>AddServlet</servlet-class>

      </servlet>
      <servlet-mapping>
            <servlet-name>hs</servlet-name>
            <url-pattern>/hs</url-pattern>
      </servlet-mapping>
</web-app>

<!-- <web-app>
      <servlet>
            <servlet-name>hs</servlet-name>
            <servlet-class>HttpServletDemo</servlet-class>

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

Differences between doGet() and doPost()?

doGet():

1. **It is design for getting the data from server**
2. **Data will visible in brower url.**
3. **We can not send secure data.**
4. **We can send only limited data.**
5. **We can send only 1024 characters data**
6. **We can't upload files the reason is url supports only characters but not binary format data**
7. **We have four approaches to work with doGet().**
    a. **By default browser**
    b. **By method='get'**
    c. **Writing data directly in URL.**
    d. **By using HyperLink.**

doPost():

1. **It is design for uploading the data into server**
2. **Data will not visible in brower url.**
3. **We can send secure data.**
4. **We can send unlimited data.**
5. **There is no limitation for data sending.**
6. **We can upload files**
7. **We have only one approaches to work with doPost().**

a. **By method='get'**

Write a program to communicating doGet() by using hyperlink:
Test.html:

```
<html>
	<body bgcolor='yellow'>
		<form>
			<a href='./rd'>CLICK_HERE</a>
		</form>

	</body>
</html>
```

RequestDemo.java

```
import java.io.IOException;
import java.io.PrintWriter;
import java.util.Arrays;
import java.util.Enumeration;
import java.util.Iterator;
import java.util.Map;
import java.util.Set;

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class RequestDemo extends HttpServlet{
	@Override
	public void doPost(HttpServletRequest
request,HttpServletResponse
			response)throws ServletException,IOException{
		response.setContentType("text/html");
		PrintWriter out = response.getWriter();
		out.println("<body bgcolor='yellow'>");
		out.println("<h2>THIS IS REQUESTDEMO DOGET
METHOD</H2>");
		out.println("<body>");


	}
}
```

Working Servlet without Eclipse:

1. Download apache tomcat software.
2. Install the appache tomcat.

Develop project structure as bellow.

develop Add.html File:

```
<html>
	<body bgcolor="Red">
		<form action="./as" method="Get">
Enter First Number: <input type="text" name="fno"/>
Enter Second Number: <input type="text" name = "sno"/>
	<input type="submit" value="add"/>
		</form>
	</body>
</html>
```

and save into folder like ServeltDemo(ProjectFolder)

develop AddServlet.java file:

```
import java.io.PrintWriter;
import java.io.IOException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import javax.servlet.ServletException;
public class AddServlet extends HttpServlet{
	@Override
	public void doGet(HttpServletRequest req,

HttpServletResponse res)throws
ServletException,IOException{
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
	int fno = Integer.parseInt(req.getParameter("fno"));
```

```
        int sno = Integer.parseInt(req.getParameter("sno"));
        int result = fno+sno;
            out.println("Result is: "+result);
        }
}
```
and saved into ServletDemo/WEB-INF/classes folder.

develop web.xml file:
==========
```
        <web-app>
        <servlet>
            <servlet-name>as</servlet-name>
            <servlet-class>AddServlet</servlet-class>
        </servlet>
        <servlet-mapping>
            <servlet-name>as</servlet-name>
            <url-pattern>/as</url-pattern>
        </servlet-mapping>
</web-app>
```

and placed into ServletDemo/WEB-INF folder

open command prompt and set classpath for servlet-api.jar file like bellow:

set classpath=.;C:\Program Files\Apache Software Foundation

\Tomcat 8.0\lib\servlet-api.jar;

compile the AddServlet.java program.
for this goto classes folder like bellow:
C:\Users\lenovo\Desktop> cd ServletDemo (enter)
C:\Users\lenovo\Desktop\ServletDemo>cd WEB-INF (ender)
C:\Users\lenovo\Desktop\ServletDemo\WEB-INF\cd classes

(enter)
C:\Users\lenovo\Desktop\ServletDemo\WEB-INF\classes>
    javac AddServlet.java
we will get AddServlet.class file in the same classes folder.

Copy the entire our project (ServletDemo) and paste into C:\Program Files\Apache Tomcat Foundations\Tomcat 8.0\ "webapps" folder.

goto Tomcat 8.0\bin folder and double click on startup batchfile
and recongnize http port number in the server console like http-nio-4445 at end of ther server console

open browser and send a request to server like bellow.
http://localhost:4445/ServerDemo/Add.html (enter)
enter first and second values add click on ADD button.
we will get output like Result: <value> based on your

input values.
Note: If we are working with any jar files placed those jar files into WEB-INF/lib folder.
Servlet Collabaration or chainging:

making a communication between one servlet to another servlet is called servlet chaining.

1. creating one servlet object within the another servlet and call service(-,-)
2. use getServlet() of ServletContext interface.(depricated)
3. RequestDispatcher
--> forward(-,-)
--> include(-,-)

ServletChainingDemo1.java

```
import java.io.IOException;

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


public class ServletChainingDemo1 extends GenericServlet{

	@Override
	public void init(ServletConfig sc )throws ServletException{
		System.out.println("scd1 init method");
	}
	@Override
	public void service(ServletRequest req, ServletResponse res)
```

```
                throws ServletException, IOException {
            System.out.println("scd1 service method");
            ServletChainingDemo2 sd = new
ServletChainingDemo2();
            sd.service(req,res);

      }
}
```

ServletChainingDemo2.java

```
import java.io.IOException;

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


public class ServletChainingDemo2 extends GenericServlet{

      ServletConfig sc1 = null;
      @Override
      public void init(ServletConfig sc )
      throws ServletException{
            System.out.println("scd2 init method");
            sc1 = sc;
      }
      @Override
      public void service(ServletRequest req, ServletResponse res)
                  throws ServletException, IOException {
            System.out.println("scd2 service method");
            if(sc1 == null){
      System.out.println("config object not created by the container");
            }
            else
      System.out.println("config object created by the container");
      }

}
<web-app>
      <servlet>
            <servlet-name>sc1</servlet-name>
            <servlet-class>ServletChainingDemo1</servlet-class>
      </servlet>
      <servlet-mapping>
```

```
            <servlet-name>sc1</servlet-name>
            <url-pattern>/sc1</url-pattern>
      </servlet-mapping>
      <servlet>
            <servlet-name>sc2</servlet-name>
            <servlet-class>ServletChainingDemo2</servlet-class>
      </servlet>
      <servlet-mapping>
            <servlet-name>sc2</servlet-name>
            <url-pattern>/sc2</url-pattern>
      </servlet-mapping>
</web-app>
```

In the above approach container is not creating any implicit objects for ServletChainingDemo2 class there is no execution for static phases by default.

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

public class Servlet1 extends GenericServlet{
      static{
            System.out.println("static block-servlet1");
      }
      ServletContext sc1;
      ServletConfig sc2;
      @Override
      public void init(ServletConfig sc)throws ServletException{
            System.out.println("init method-servlet1");
            System.out.println("sc: "+sc);
            sc1 = sc.getServletContext();
            sc2=sc;
      }
      @Override
      public void service(ServletRequest request, ServletResponse
response){
            System.out.println("service method-servlet1");
            System.out.println("this: "+this.hashCode());
            System.out.println("thread:
"+Thread.currentThread().getName());
            System.out.println("request: "+request.hashCode());
```

```
            System.out.println("response: "+response.hashCode());
            System.out.println("scontext: "+sc1);
            System.out.println("sconfig: "+sc2);

            System.out.println("==================");
            Servlet2 obj = new Servlet2();
            obj.service(request, response);
      }

}
```

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

public class Servlet2 extends GenericServlet{
      static{
            System.out.println("static block-servlet2");
      }
      ServletContext sc1;
      ServletConfig sc2;
      @Override
      public void init(ServletConfig sc)throws ServletException{
            System.out.println("init method-servlet2");
            System.out.println("sc: "+sc);
            sc1 = sc.getServletContext();
            sc2=sc;
      }
      @Override
      public void service(ServletRequest request, ServletResponse
response){
            System.out.println("service method-servlet2");
            System.out.println("this: "+this.hashCode());
            System.out.println("thread:
"+Thread.currentThread().getName());
            System.out.println("request: "+request.hashCode());
            System.out.println("response: "+response.hashCode());
            System.out.println("scontext: "+sc1);
            System.out.println("sconfig: "+sc2);
      }

}
```

```
<web-app>
```

```
	<servlet>
		<servlet-name>s1</servlet-name>
		<servlet-class>Servlet1</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>s1</servlet-name>
		<url-pattern>/s1</url-pattern>
	</servlet-mapping>
	<servlet>
		<servlet-name>s2</servlet-name>
		<servlet-class>Servlet2</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>s2</servlet-name>
		<url-pattern>/s2</url-pattern>
	</servlet-mapping>
</web-app>
```

To avoiding this problem we should use one method that is getServlet() of ServletContext interface.

With the help of this method will get automatic static loading and initialization as well as non-static loading and initialization phases and finally calls service method.

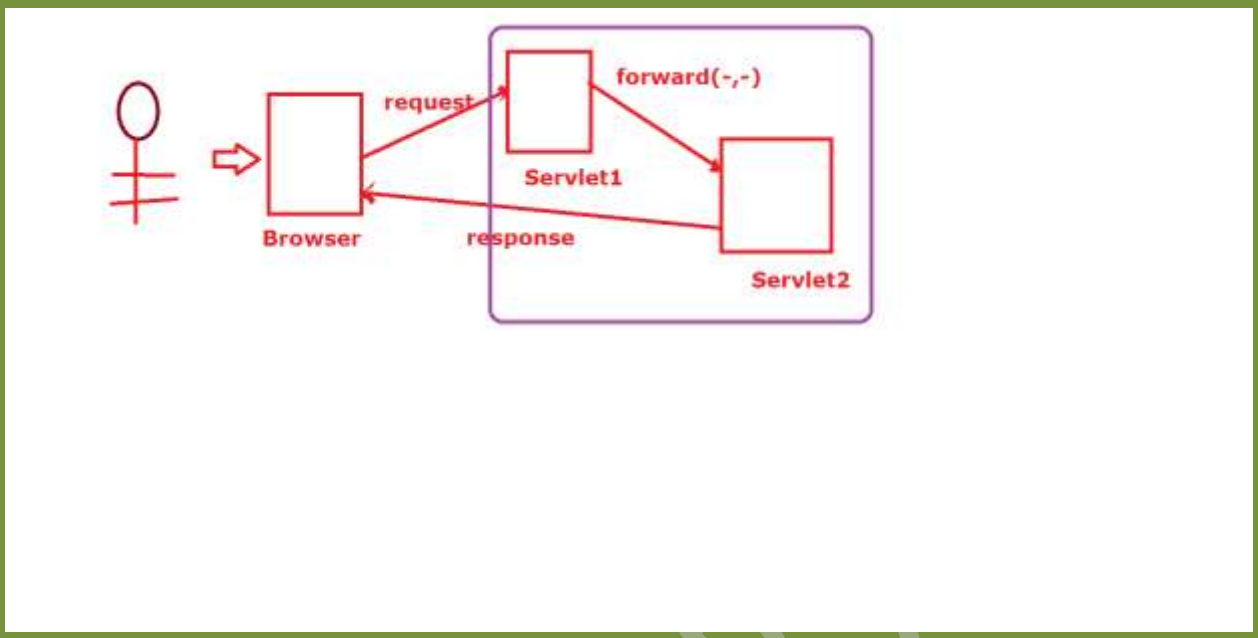
But it is deprecated in servlet version, if we writing program on top of this method we will exception

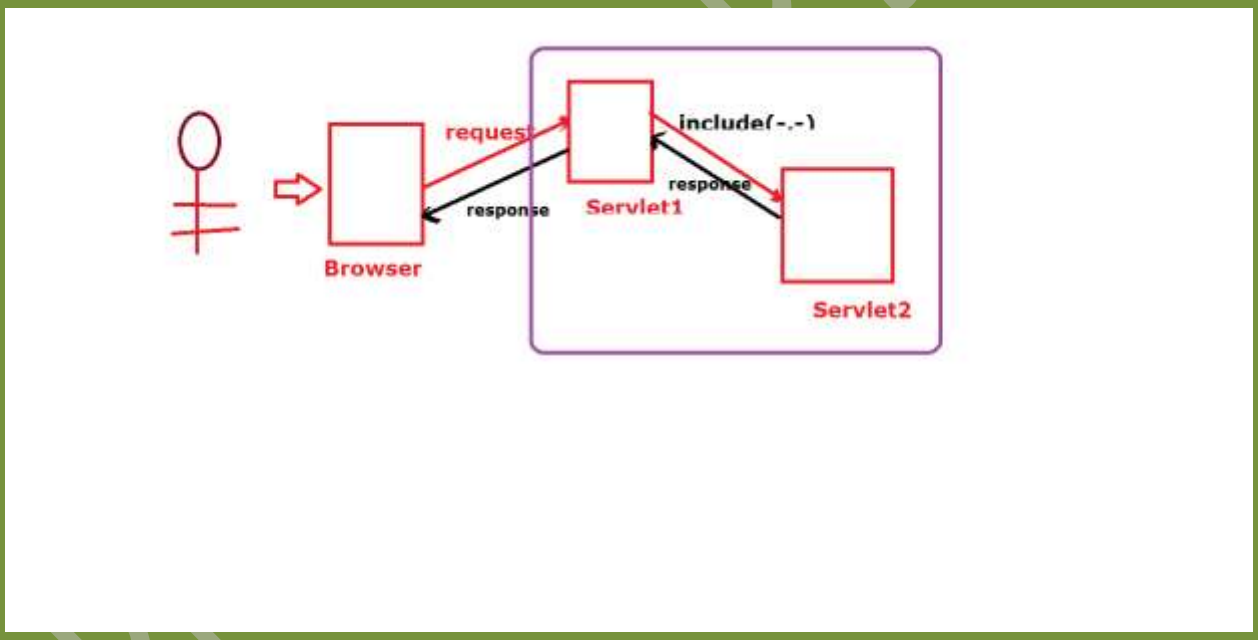
```
ServletContext sct=req.getServletContext();
System.out.println("sct: "+sct);
SecondServlet ss =
(SecondServlet)sct.getServlet("./sc2");
System.out.println("ss: "+ss);
```

The above code is not use full in new servlet versions. To avoiding above problems we have the best approach in Servlet concept that is RequestDispatcher.

RequestDispatcher is use full for making a communication between one servlet to another servlet. We can achieve this object by using one method that is getRequestDispathcer(-) of ServletRequest object finally.

We need to use two method like forward(-,-) and include(-,_ methods.





With help of RequestDispatcher we can able to communicating with the following resources.

1. **Servlet.**
2. **JSP Files.**
3. **Html files.**
4. **Normal Text Files.**

Program on both forward(-,-) and include(-,-)

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.GenericServlet;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class Servlet1 extends GenericServlet{
     @Override
     public void service(ServletRequest request, ServletResponse
response)
               throws IOException,ServletException{
          response.setContentType("text/html");
          PrintWriter out = response.getWriter();
          out.println("<body bgcolor='yellow'>");
          out.println("<h1>THIS IS SERVLET-1 RESPONSE</H1>");
          out.println("<h1>S1-Request:"+request.hashCode()+"</h1>");
          out.println("<h1>S1-Response:"+response.hashCode()+"</h1>");
          out.println("<body>");
          RequestDispatcher rd = request.getRequestDispatcher("./s2");
          //rd.forward(request, response);
          rd.include(request, response);
     }
}


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

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

public class Servlet2 extends GenericServlet{

     @Override
     public void service(ServletRequest request, ServletResponse
response)
     throws IOException,ServletException{
          response.setContentType("text/html");
          PrintWriter out = response.getWriter();
          out.println("<body bgcolor='Green'>");
          out.println("<h1>THIS IS SERVLET-2 RESPONSE</H1>");
```

```
            out.println("<h1>S2-Request:"+request.hashCode()+"</h1>");
            out.println("<h1>S2-Response:"+response.hashCode()+"</h1>");
            out.println("<body>");

      }

}


<web-app>
      <servlet>
            <servlet-name>s1</servlet-name>
            <servlet-class>Servlet1</servlet-class>
      </servlet>
      <servlet-mapping>
            <servlet-name>s1</servlet-name>
            <url-pattern>/s1</url-pattern>
      </servlet-mapping>
      <servlet>
            <servlet-name>s2</servlet-name>
            <servlet-class>Servlet2</servlet-class>
      </servlet>
      <servlet-mapping>
            <servlet-name>s2</servlet-name>
            <url-pattern>/s2</url-pattern>
      </servlet-mapping>
</web-app>

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

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


public class Servlet1 extends GenericServlet{

      @Override
      public void init(ServletConfig sc)throws ServletException{
            System.out.println("servlet-1 init method");
      }
      @Override
```

```
	public void service(ServletRequest req, ServletResponse res)
			throws ServletException, IOException {
		System.out.println("servlet1 service method");
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		out.println("<H1>THIS IS SERVLET-1 SERVICE");
		//RequestDispatcher rd = req.getRequestDispatcher("./s2");
		//rd.forward(req, res);
		//rd.include(req, res);

		//RequestDispatcher rd1 =
req.getRequestDispatcher("./hello.html");
		//RequestDispatcher rd1 =
req.getRequestDispatcher("./first.jsp");
		RequestDispatcher rd1 =
req.getRequestDispatcher("./second.txt");
		rd1.forward(req, res);
		out.println("<H1>THIS IS SERVLET-1 RESPONSE");

	}

}
```

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

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


public class Servlet2 extends GenericServlet{

	@Override
	public void init(ServletConfig sc)throws ServletException{
		System.out.println("servlet-2 init method");
	}
	@Override
	public void service(ServletRequest req, ServletResponse res)
			throws ServletException, IOException {
		System.out.println("servlet2 service method");
```

```
            res.setContentType("text/html");
            PrintWriter out = res.getWriter();
            out.println("<H1>THIS IS SERVLET-2 SERVICE");
            out.println("<H1>THIS IS SERVLET-2 RESPONSE");

      }

}


<web-app>
      <servlet>
            <servlet-name>s1</servlet-name>
            <servlet-class>Servlet1</servlet-class>
      </servlet>
      <servlet-mapping>
            <servlet-name>s1</servlet-name>
            <url-pattern>/s1</url-pattern>
      </servlet-mapping>
      <servlet>
            <servlet-name>s2</servlet-name>
            <servlet-class>Servlet2</servlet-class>
      </servlet>
      <servlet-mapping>
            <servlet-name>s2</servlet-name>
            <url-pattern>/s2</url-pattern>
      </servlet-mapping>
</web-app>
```

Hello.html:

```
<h1>THIS IS HTML FILE</h1>
```

First.jsp

```
<h1>THIS IS JSP FILE</h1>
```

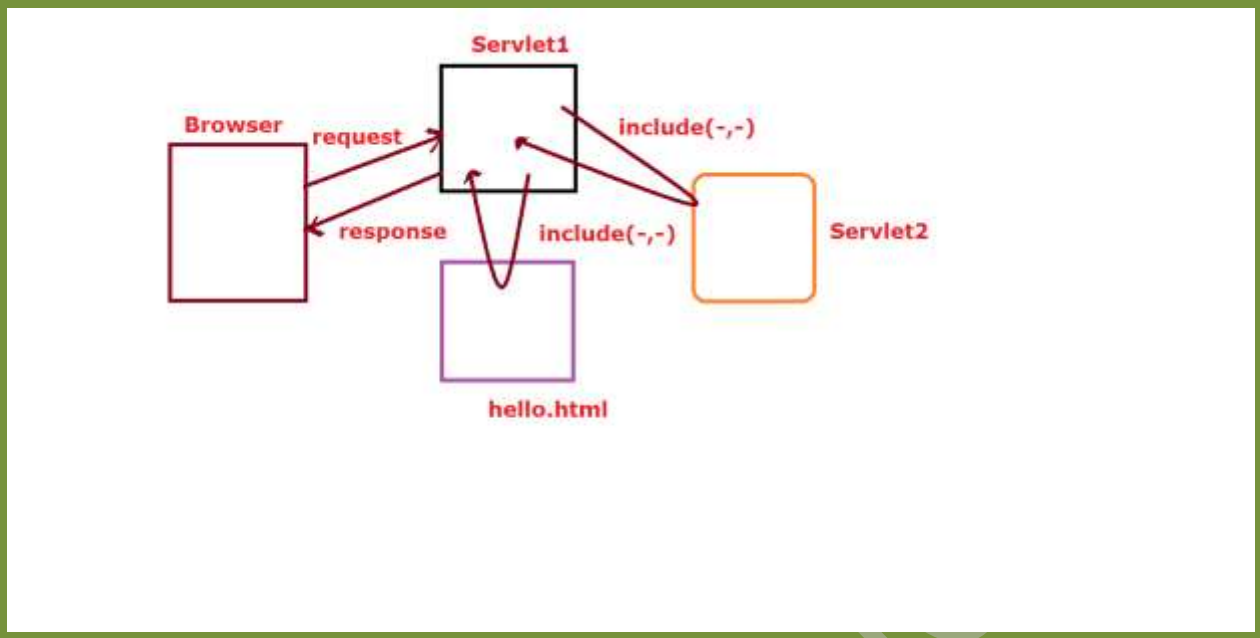
second.txt:

```
<h1>THIS IS NORMAL TEXT FILE</h1>
```

Rewrite the above Servlet1.java as bellow

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

import javax.servlet.GenericServlet;
```

```
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class Servlet1 extends GenericServlet{
     @Override
     public void init(ServletConfig sc)
               throws ServletException{
          System.out.println("servlet-1 init method");
     }
     @Override
     public void service(ServletRequest req,
               ServletResponse res)
               throws ServletException, IOException {
          System.out.println("servlet1 service method");
          res.setContentType("text/html");
          PrintWriter out = res.getWriter();
          out.println("<H1>THIS IS SERVLET-1 SERVICE");
RequestDispatcher rd = req.getRequestDispatcher("./s2");
          //rd.forward(req, res);
          rd.include(req, res);

RequestDispatcher rd1=req.getRequestDispatcher("./hello.html");
rd1.include(req, res);
/*RequestDispatcher rd2=req.getRequestDispatcher("./first.jsp");
rd2.forward(req, res);
 RequestDispatcher rd3=req.getRequestDispatcher("./second.txt");
          rd3.forward(req, res);*/
          out.println("<H1>THIS IS SERVLET-1 RESPONSE");
     }

}
```



After writing the forward(-,-) in servlet don't use again either forward(-,-) or include(-,-) in the same servlet if use also those are not executing.

But after writing include(-,-), we can able to write forward(-,-) or include(-,-) methods again in the same servlet.

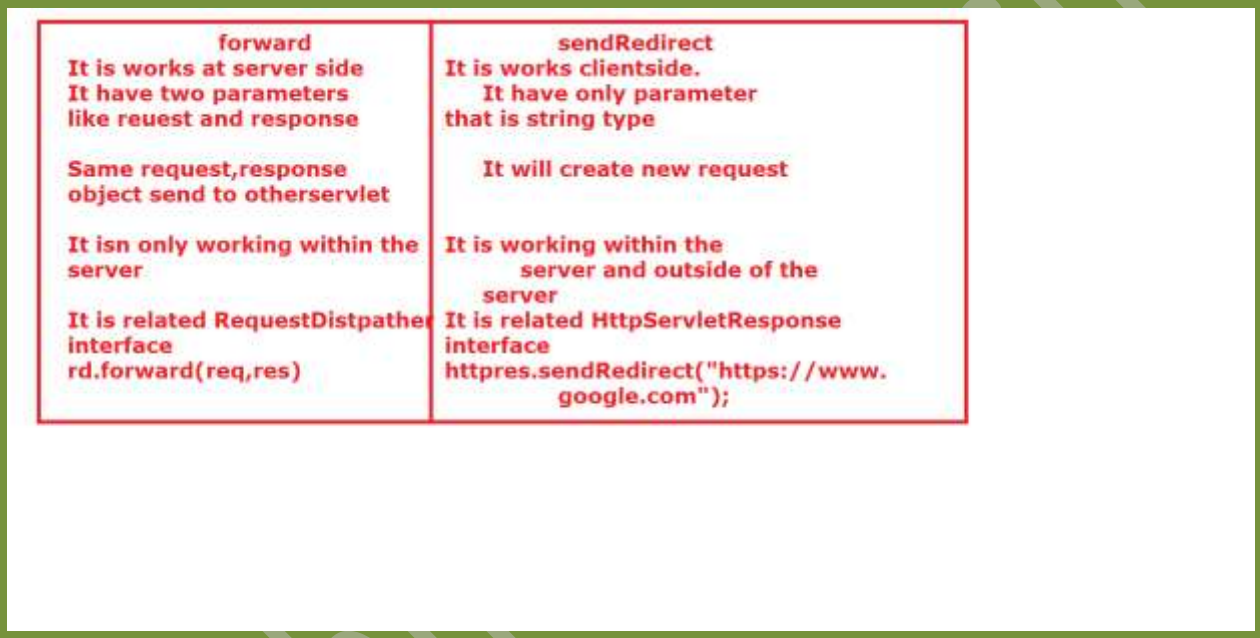
forward(-,-) not carrying the response information, whereas include(-,-) will carrying the response information from either servlet or html file or jsp file or normal text files.

Program sendRedirect:

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

import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class Servlet1 extends HttpServlet{
	@Override
	public void doGet(HttpServletRequest req,
			HttpServletResponse res)
			throws ServletException, IOException {
```

```
            System.out.println("servlet1 service method");
            res.setContentType("text/html");
            PrintWriter out = res.getWriter();
            out.println("<H1>THIS IS SERVLET-1 SERVICE");
            //res.sendRedirect("https://www.google.com");
            res.sendRedirect("./s2");


            out.println("<H1>THIS IS SERVLET-1 RESPONSE");
      }
}
import java.io.IOException;
import java.io.PrintWriter;

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

public class Servlet2 extends GenericServlet{
      @Override
      public void init(ServletConfig sc)throws ServletException{
            System.out.println("servlet-2 init method");
      }
      @Override
      public void service(ServletRequest req,
                  ServletResponse res)
                  throws ServletException, IOException {
            System.out.println("servlet2 service method");
            res.setContentType("text/html");
            PrintWriter out = res.getWriter();
            out.println("<H1>THIS IS SERVLET-2 SERVICE");
            out.println("<H1>THIS IS SERVLET-2 RESPONSE");
      }
}
<web-app>
      <servlet>
            <servlet-name>s1</servlet-name>
            <servlet-class>Servlet1</servlet-class>
      </servlet>
      <servlet-mapping>
            <servlet-name>s1</servlet-name>
            <url-pattern>/s1</url-pattern>
```

```
	</servlet-mapping>
	<servlet>
		<servlet-name>s2</servlet-name>
		<servlet-class>Servlet2</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>s2</servlet-name>
		<url-pattern>/s2</url-pattern>
	</servlet-mapping>
</web-app>
```



forward	sendRedirect
It is works at server side It have two parameters like reuest and response Same request,response object send to otherservlet It isn only working within the server It is related RequestDistpather interface rd.forward(req,res)	It is works clientside. It have only parameter that is string type It will create new request It is working within the server and outside of the server It is related HttpServletResponse interface httpres.sendRedirect("https://www.google.com");

Test1 project:

TestServlet1.java

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

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

public class TestServlet1 extends HttpServlet{
	static{
		System.out.println("testservlet1 loading");
	}
	@Override
```

```
	public void service(HttpServletRequest request,
			HttpServletResponse response)
			throws IOException,ServletException{
		response.setContentType("text/html");
		PrintWriter out = response.getWriter();
		out.println("<body bgcolor='yellow'>");
		//response.sendRedirect("./s3");
		//response.sendRedirect("http://localhost:3016/Test2/s2");
		response.sendRedirect("https://www.google.com");
response.sendRedirect(http://localhost:3016/Test2/s2?user=ram”);
		out.println("THIS IS TESTSERVLET1 RESPONSE");
	}
}
```

TestServlet3.java

```
import java.io.IOException;

import java.io.PrintWriter;


import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;


public class TestServlet3 extends HttpServlet{

	static{

		System.out.println("testservlet3 loading");

	}

	@Override

	public void service(HttpServletRequest request,

			HttpServletResponse response)

			throws IOException,ServletException{
```

```
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            out.println("<body bgcolor='yellow'>");

            out.println("THIS IS TESTSERVLET3 RESPONSE");
      }
}
```

Web.xml:

```
<web-app>
      <servlet>
            <servlet-name>s1</servlet-name>
            <servlet-class>TestServlet1</servlet-class>
            <load-on-startup>0</load-on-startup>
      </servlet>
      <servlet-mapping>
            <servlet-name>s1</servlet-name>
            <url-pattern>/s1</url-pattern>
      </servlet-mapping>
      <servlet>
            <servlet-name>s3</servlet-name>
            <servlet-class>TestServlet3</servlet-class>
            <load-on-startup>1</load-on-startup>
      </servlet>
      <servlet-mapping>
            <servlet-name>s3</servlet-name>
            <url-pattern>/s3</url-pattern>
      </servlet-mapping>
</web-app>
```

Test2 Project:

TestServlet2.java

```
import java.io.IOException;

import java.io.PrintWriter;
```

```
import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;


public class TestServlet2 extends HttpServlet{

     static{

          System.out.println("testservlet2 loading");

     }

     @Override

     public void service(HttpServletRequest request,

               HttpServletResponse response)

               throws IOException,ServletException{

          response.setContentType("text/html");

          PrintWriter out = response.getWriter();

          out.println("<body bgcolor='yellow'>");

          String username=request.getParameter("user");
          out.println("username: "+username);

          out.println("THIS IS TESTSERVLET2 RESPONSE");

     }

}


Web.xml:

<web-app>
     <servlet>
          <servlet-name>s2</servlet-name>
          <servlet-class>TestServlet2</servlet-class>
```

```
            <load-on-startup>0</load-on-startup>
      </servlet>
      <servlet-mapping>
            <servlet-name>s2</servlet-name>
            <url-pattern>/s2</url-pattern>
      </servlet-mapping>
</web-app>
```

Program on HttpServletRequest about parameters:

If we have morethan one text field on top of same name and trying to read all the values from all text field we should go for one method like getParameterValues(-).

If we have only one text field on top of one name and trying to read the data from that table we should prefer method like getParameter(-).

First.html

```
<html>
<body bgcolor='pink'>
      <form action="./s1">

            Enter Name: <input type='text' name='name'/><br/>

            <table>

                  <tr>
                        <td>Courses</td>
                  </tr>
                  <tr>
                        <td>Course1</td>
                        <td><input type='text' name="course" value=''/>
                  </tr>
                  <tr>
                        <td>Course2</td>
                        <td><input type='text' name="course" value=''/>
                  </tr>
                  <tr>
                        <td>Course3</td>
                        <td><input type='text' name="course" value=''/>
```

```
                </tr>
                <tr>
                     <td>Course4</td>
                     <td><input type='text' name="course" value=''/>
                </tr>
           </table>
           <input type='submit' value='CLICK'/>
     </form>
</body>
</html>

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

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

public class Servlet1 extends HttpServlet{
     @Override
     public void doGet(HttpServletRequest req,
                HttpServletResponse res)
                throws ServletException, IOException {
           System.out.println("servlet1 service method");
           res.setContentType("text/html");
           PrintWriter out = res.getWriter();
           out.println("<H1>THIS IS SERVLET-1 SERVICE</H1><br/>");
           String name = req.getParameter("name");
           out.println("Name: "+name);
           String[] s = req.getParameterValues("course");
           for(String s1: s){
                out.println(s1+"<br/>");
           }
           out.println("<H1>THIS IS SERVLET-1 RESPONSE");
     }
}


<web-app>
     <servlet>
           <servlet-name>s1</servlet-name>
           <servlet-class>Servlet1</servlet-class>
     </servlet>
     <servlet-mapping>
           <servlet-name>s1</servlet-name>
```

```
            <url-pattern>/s1</url-pattern>
      </servlet-mapping>
</web-app>
```

```
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 Servlet1 extends HttpServlet{
      @Override
      public void doGet(HttpServletRequest req,
                  HttpServletResponse res)
                  throws ServletException, IOException {
            System.out.println("servlet1 service method");
            res.setContentType("text/html");
            PrintWriter out = res.getWriter();
            out.println("<H1>THIS IS SERVLET-1 SERVICE</H1><br/>");
            Enumeration e= req.getParameterNames();
            while(e.hasMoreElements()){
                  String tfn = (String) e.nextElement();
                  String tfv = req.getParameter(tfn);
                  out.println(tfn+"...."+tfv);
            }
            out.println("<H1>THIS IS SERVLET-1 RESPONSE");
      }
}


<html>
<body bgcolor='pink'>
      <form action="./s1">
            Enter Name  : <input type='text' name='name'/><br/>
            Enter Age   : <input type='text' name='age'/><br/>
            Enter course: <input type='text' name='course'/><br/>
            Enter fee   : <input type='text' name='fee'/><br/>

            <input type='submit' value='CLICK'/>
      </form>
</body>
```

```
</html>

<web-app>
	<servlet>
		<servlet-name>s1</servlet-name>
		<servlet-class>Servlet1</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>s1</servlet-name>
		<url-pattern>/s1</url-pattern>
	</servlet-mapping>
</web-app>
```

Program on getPrameterNames() and header information:

Without having any idea on text field names which are avialble in .html file .jsp file, if we want to read text field names and its related content better choose method like getParameterNames().

```
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 Servlet1 extends HttpServlet{

	@Override

	public void doGet(HttpServletRequest req,

			HttpServletResponse res)
```

```
            throws ServletException, IOException {
        System.out.println("servlet1 service method");
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("<H1>THIS IS SERVLET-1
SERVICE</H1><br/>");
        Enumeration e= req.getParameterNames();
        while(e.hasMoreElements()){
            String tfn = (String) e.nextElement();
            String tfv = req.getParameter(tfn);
            out.println(tfn+"...."+tfv);
        }
        Enumeration<String> e1 = req.getHeaderNames();
        while(e1.hasMoreElements()){
            String headerName = e1.nextElement();
            String headerValue = req.getHeader(headerName);
            out.println(headerName+":::"+headerValue);
            System.out.println(headerName+"::"+headerValue);
        }
        out.println("<H1>THIS IS SERVLET-1 RESPONSE");
    }
}
```

```
<web-app>
    <servlet>
        <servlet-name>s1</servlet-name>
        <servlet-class>Servlet1</servlet-class>
```

```
	</servlet>
	<servlet-mapping>
		<servlet-name>s1</servlet-name>
		<url-pattern>/s1</url-pattern>
	</servlet-mapping>
</web-app>
```

```
<html>
<body bgcolor='pink'>
	<form action="./s1">
		Enter Name  : <input type='text' name='name'/><br/>
		Enter Age   : <input type='text' name='age'/><br/>
		Enter course: <input type='text' name='course'/><br/>
		Enter fee   : <input type='text' name='fee'/><br/>

		<input type='submit' value='CLICK'/>
	</form>
</body>
</html>
```

Program On getParameterMap():

This method will read the data like text filed name and its values in the format of Map object from ServletRequest Object.

```
import java.io.IOException;

import java.io.PrintWriter;

import java.util.Arrays;

import java.util.Iterator;

import java.util.Map;

import java.util.Set;


import javax.servlet.ServletException;

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

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class RequestDemo extends HttpServlet{
	@Override
	public void doPost(HttpServletRequest
request,HttpServletResponse
			response)throws ServletException,IOException{
		response.setContentType("text/html");
		PrintWriter out = response.getWriter();
		out.println("<body bgcolor='yellow'>");
		Map<String,String[]>map =request.getParameterMap();
		Set<Map.Entry<String,String[]>> set = map.entrySet();
		Iterator <Map.Entry<String,String[]>> it = set.iterator();
		while(it.hasNext()){
			System.out.println("---------------");
			Map.Entry<String,String[]> ent= it.next();

	System.out.println(ent.getKey()+"..."+Arrays.toString(ent.get
Value()));
		}
	out.println("<body>");
	}
}
```

Program on Attributes of HttpServletRequest Object:

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

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 Servlet1 extends HttpServlet{
	@Override
	public void doGet(HttpServletRequest req,
			HttpServletResponse res)
			throws ServletException, IOException {
		System.out.println("servlet1 service method");
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		out.println("<H1>THIS IS SERVLET-1
SERVICE</H1><br/>");
		String name = req.getParameter("name");
		req.setAttribute("username",name);
		RequestDispatcher rd =
req.getRequestDispatcher("./s2");
		rd.forward(req, res);
```

```
            out.println("<H1>THIS IS SERVLET-1 RESPONSE");

      }

}
```

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

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


public class Servlet2 extends HttpServlet{
      @Override
      public void doGet(HttpServletRequest req,
                  HttpServletResponse res)
                  throws ServletException, IOException {
            System.out.println("servlet2 service method");
            res.setContentType("text/html");
            PrintWriter out = res.getWriter();
            String userName = (String) req.getAttribute("username");
            out.println("UserName Is: "+userName);

      }

}
```

```
<web-app>
      <servlet>
            <servlet-name>s1</servlet-name>
            <servlet-class>Servlet1</servlet-class>
      </servlet>
      <servlet-mapping>
            <servlet-name>s1</servlet-name>
            <url-pattern>/s1</url-pattern>
      </servlet-mapping>
      <servlet>
            <servlet-name>s2</servlet-name>
            <servlet-class>Servlet2</servlet-class>
      </servlet>
      <servlet-mapping>
            <servlet-name>s2</servlet-name>
            <url-pattern>/s2</url-pattern>
      </servlet-mapping>
```

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

```
<html>
<body bgcolor='Yellow'>
     <form action="./s1">
          Enter Name  : <input type='text' name='name'/><br/>
          <input type='submit' value='CLICK'/>
     </form>
</body>
</html>
```

HttpServlet is always working with HttpProtocol.

HttpProtocol is stateless protocol.

Every time this protocol will treat client as a unique client.

To recognize client as a unique client or same client we should

go for Session Tracking.

We can achieve session tracking by using the following four ways.

a. **Cookie**
b. **Hidden Form Fields**
c. **URL Rewriting**
d. **HttpSession**

Programm on cookie:

```
import java.io.IOException;

import java.io.PrintWriter;

import java.util.Enumeration;
```

```
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;

public class Servlet1 extends HttpServlet{
	@Override
	public void doGet(HttpServletRequest req,
			HttpServletResponse res)
			throws ServletException, IOException {
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		String name = req.getParameter("name");
		out.println("In servlet1: "+name);
		Cookie c = new Cookie("name", name);
		res.addCookie(c);
		out.println("<form action='./s2' method='get'>");
		out.println("<input type='submit' value='click' >");
		out.println("</form>");
```

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

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



public class Servlet2 extends HttpServlet{
	@Override
	public void doGet(HttpServletRequest req,
			HttpServletResponse res)
			throws ServletException, IOException {
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		/*Cookie[] c = req.getCookies();
		out.println("in servlet 2: "+c[0].getValue());
		*/
	}
```

```
}
```

```
<html>
<body bgcolor='Yellow'>
	<form action="./s1">
		Enter Name  : <input type='text' name='name'/><br/>
		<input type='submit' value='CLICK'/>
	</form>
</body>
</html>
```

```
<web-app>
	<servlet>

		<servlet-name>s1</servlet-name>
		<servlet-class>Servlet1</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>s1</servlet-name>
		<url-pattern>/s1</url-pattern>
	</servlet-mapping>
	<servlet>
		<servlet-name>s2</servlet-name>
		<servlet-class>Servlet2</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>s2</servlet-name>
		<url-pattern>/s2</url-pattern>
	</servlet-mapping>
</web-app>
```

Program on hidden form fileds:

```
import java.io.IOException;

import java.io.PrintWriter;

import java.util.Enumeration;


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;


public class Servlet1 extends HttpServlet{
	@Override
	public void doGet(HttpServletRequest req,
			HttpServletResponse res)
			throws ServletException, IOException {
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		String username = req.getParameter("name");
		out.println("In servlet1: "+username);
		out.println("<form action='./s2' method='get'>");
		out.println("<input type='hidden' name='hname'
value='"+username+"'/><br/>");
		out.println("<input
type='submit'value='submit'/><br/>");
		out.println("</form>");



	}
}
import java.io.IOException;
```

```
import java.io.PrintWriter;


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



public class Servlet2 extends HttpServlet{
	@Override
	public void doGet(HttpServletRequest req,
			HttpServletResponse res)
			throws ServletException, IOException {
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		String username = req.getParameter("hname");
		out.println("Servlet-Hidden: "+username);

	}

}
```

```
<html>
<body bgcolor='Yellow'>
	<form action="./s1">
Enter Name:<input type='text' name='name'/><br/>
```

```
            <input type='submit' value='CLICK'/>
      </form>
</body>
</html>
```

```
<web-app>
      <servlet>
            <servlet-name>s1</servlet-name>
            <servlet-class>Servlet1</servlet-class>
      </servlet>
      <servlet-mapping>
            <servlet-name>s1</servlet-name>
            <url-pattern>/s1</url-pattern>
      </servlet-mapping>
      <servlet>
            <servlet-name>s2</servlet-name>
            <servlet-class>Servlet2</servlet-class>
      </servlet>
      <servlet-mapping>
            <servlet-name>s2</servlet-name>
            <url-pattern>/s2</url-pattern>
      </servlet-mapping>
</web-app>
```

Program on url rewriting:

```
import java.io.IOException;

import java.io.PrintWriter;

import java.util.Enumeration;


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;

public class Servlet1 extends HttpServlet{
	@Override
	public void doGet(HttpServletRequest req,
			HttpServletResponse res)
			throws ServletException, IOException {
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		String username = req.getParameter("name");
		out.println("In servlet1: "+username);

		out.println("<a
href='./s2?name="+username+"'>visit</a>");

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

import javax.servlet.ServletException;
import javax.servlet.http.Cookie;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
```

```
import javax.servlet.http.HttpServletResponse;



public class Servlet2 extends HttpServlet{
	@Override
	public void doGet(HttpServletRequest req,
			HttpServletResponse res)
			throws ServletException, IOException {
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		String username = req.getParameter("name");
		out.println("Servlet-Hidden: "+username);


	}


}
```

```
<web-app>
	<servlet>
		<servlet-name>s1</servlet-name>
		<servlet-class>Servlet1</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>s1</servlet-name>
		<url-pattern>/s1</url-pattern>
	</servlet-mapping>
	<servlet>
		<servlet-name>s2</servlet-name>
		<servlet-class>Servlet2</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>s2</servlet-name>
```

```
            <url-pattern>/s2</url-pattern>
      </servlet-mapping>
</web-app>
```

```
<html>
<body bgcolor='Yellow'>
      <form action="./s1">
Enter Name:<input type='text' name='name'/><br/>
            <input type='submit' value='CLICK'/>
      </form>
</body>
</html>
```

```
import java.io.IOException;

import java.io.PrintWriter;


import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;


public class Servlet1 extends HttpServlet{

      @Override

      public void doGet(HttpServletRequest req,

                  HttpServletResponse res)

                  throws ServletException, IOException {

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

```
            PrintWriter out = res.getWriter();
            String username = req.getParameter("name");
            out.println("In servlet1: "+username);

            HttpSession session = req.getSession();
            session.setAttribute("enduser",username);

            RequestDispatcher rd =
                        req.getRequestDispatcher("./s2");
            rd.forward(req, res);
      }
}
import java.io.IOException;
import java.io.PrintWriter;

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


public class Servlet2 extends HttpServlet{
      @Override
```

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

		HttpSession session = req.getSession();
		String enduserName =
			(String) session.getAttribute("enduser");
		out.println("servlet2-session: "+enduserName);



	}

}
```

```
<web-app>
	<servlet>
		<servlet-name>s1</servlet-name>
		<servlet-class>Servlet1</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>s1</servlet-name>
		<url-pattern>/s1</url-pattern>
	</servlet-mapping>
	<servlet>
		<servlet-name>s2</servlet-name>
		<servlet-class>Servlet2</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>s2</servlet-name>
		<url-pattern>/s2</url-pattern>
	</servlet-mapping>
```

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

```
<html>
<body bgcolor='Yellow'>
     <form action="./s1">
Enter Name:<input type='text' name='name'/><br/>
          <input type='submit' value='CLICK'/>
     </form>
</body>
</html>
```

Filter:

--> Is one java object.

--> Is one web component is usesd for developing web application like servlet and jsp

--> It is one servlet container/server managed java object.

--> Like Servlet, Filter is also having life cycle methods.

like init(-)

destroy()

doFilter(-,-,-)

All the above methods are calling by ServletContainer.
Thats why we are calling filter is one servlet container manager object.

It is acting as mediator between brower and servlet.
Usage:

Checks Logging details valid or not (verification/validation)

Checks whether end user valid or not(Authentication)

Doing some common operations for all servlets.

Types of Filters:

If we want to execute or process any logic before servlet logic after servlet logic exectuion, we should use filter.
Based on above pointer Filters are classified into two types.

a. **Pre Filter:**
   Before logic of servlet execution, filter logic will be executes.
b. **Post Filter.**
   After logic of servlet execution, filter logic will be executes.

In filter concept we have Three main import interfaces.

a. **Filter**

b. **FilterConfig**
c. **FilterChain**

Both Filter and FilterChain interfaces having one common method name is doFilter().
But Filter and FilterChain interface doFilter() method functionalities are different from one to another.
<u>What is difference between doFilter(-,-,-) of Filter interface and doFilter(-,-) FilterChain interface:</u>

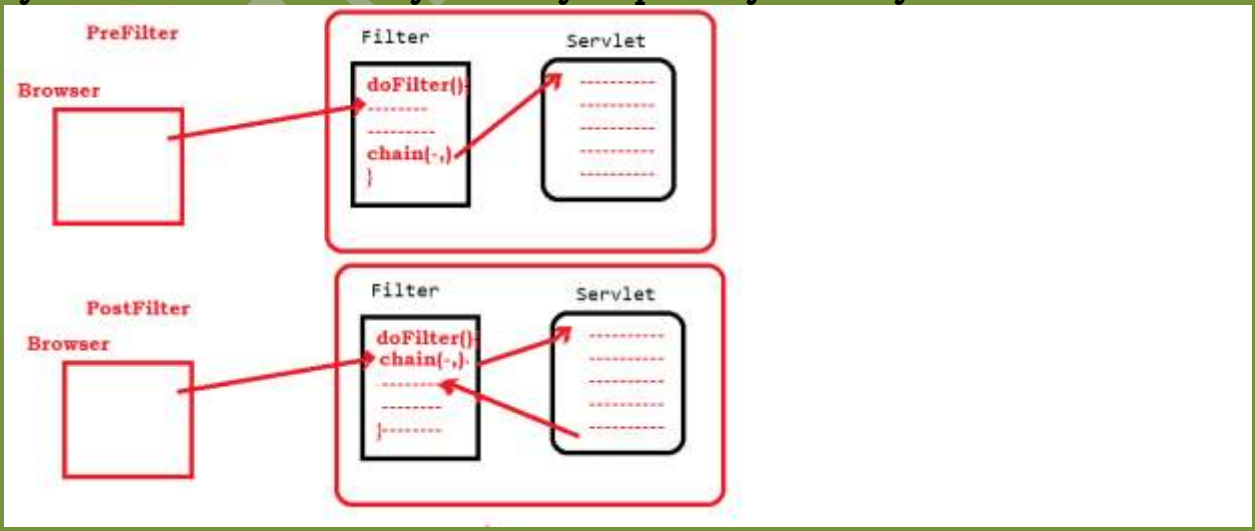
<u>doFilter(-,-,-) of Filter interface:</u>

1. **This is a life cycle method of Filter.**
2. **We should override this method in our filter class.**
3. **Calling and execution every thing taken care by servlet container by default.**
4. **It is having 3 parameters like ServletRequest, ServletResponse, FilterChain.**
5. **Acutal logic like authentication, logging details verification and writing common logic of all servlets we need write in this method.**

<u>doFilter(-,-) of FilterChain interface:</u>

1. **This is a not life cycle method of FilterChain.**
2. **No need override this method in our filter class.**
3. **Servelt Container no calling and execute this method by default.**
4. **It is having 2 parameters like ServletRequest, ServletResponse.**
5. **This method is useful making communication between filter to servlet.**

Filter object always creating by Servletcontainer meanwhile of loading the byte code from secondary memory to primary memory.



Proramming on filter:

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

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


public class MyFilter implements Filter {

	@Override
	public void destroy() {
		System.out.println("Filter destroy()");
	}

	@Override
	public void doFilter(ServletRequest req, ServletResponse res,
			FilterChain chain) throws IOException,
ServletException {
		System.out.println("Filter doFilter()");
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		out.println("file executing before servlet<br/>");
		chain.doFilter(req, res);
		out.println("file executing after servlet<br/>");
	}

	@Override
	public void init(FilterConfig arg0) throws ServletException {
		System.out.println("Filter init()");

	}

}


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

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


public class MyServlet extends GenericServlet {

	@Override
	public void service(ServletRequest req, ServletResponse res)
			throws ServletException, IOException {
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		out.println("MyServlet is processing the request<br/>");

	}

}
```

```xml
<web-app>
	<servlet>
		<servlet-name>ms</servlet-name>
		<servlet-class>MyServlet</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>ms</servlet-name>
		<url-pattern>/ms</url-pattern>
	</servlet-mapping>

	<filter>
		<filter-name>mf</filter-name>
		<filter-class>MyFilter</filter-class>
	</filter>
	<filter-mapping>
		<filter-name>mf</filter-name>
		<url-pattern>/ms</url-pattern>
	</filter-mapping>


</web-app>
```

```xml
<form action="./ms">
	<input type='submit' value='submit'/>
</form>
```

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class MyFilter implements Filter {
	@Override
	public void destroy() {
		System.out.println("Filter destroy()");
	}
	@Override
	public void doFilter(ServletRequest req, ServletResponse res,
			FilterChain chain) throws IOException,
ServletException {
		System.out.println("Filter doFilter()");
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		out.println("file executing before servlet<br/>");
		String username = req.getParameter("username");
		String password = req.getParameter("password");
		if(password.equals("ramchandra")){
			System.out.println("if");
			chain.doFilter(req, res);
		}
		else{
			System.out.println("else");
			RequestDispatcher rd =
req.getRequestDispatcher("./test.html");
			rd.include(req,res);
		}

		out.println("file executing after servlet<br/>");
	}
	@Override
	public void init(FilterConfig arg0) throws ServletException {
		System.out.println("Filter init()");
	}

}
```

import java.io.IOException;

import java.io.PrintWriter;

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



public class MyServlet extends GenericServlet {

	@Override
	public void service(ServletRequest req, ServletResponse res)
			throws ServletException, IOException {
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		out.println("MyServlet is processing the request<br/>");

	}

}<body bgcolor='yellow'>
<form action="./ms">
	Enter UserName: <input type='text' name='username'/><br/>
	Enter Password: <input type='password' name='password'/><br/>
	<input type='submit' value='submit'/>
</form>
</body>

<web-app>
	<servlet>
		<servlet-name>ms</servlet-name>
```

```
            <servlet-class>MyServlet</servlet-class>
      </servlet>
      <servlet-mapping>
            <servlet-name>ms</servlet-name>
            <url-pattern>/ms</url-pattern>
      </servlet-mapping>
      <filter>
            <filter-name>mf</filter-name>
            <filter-class>MyFilter</filter-class>
      </filter>
      <filter-mapping>
            <filter-name>mf</filter-name>
            <url-pattern>/ms</url-pattern>
      </filter-mapping>


</web-app>
```

```
<body bgcolor='yellow'>
<form action="./ms">
Enter UserName : <input type='text' name='username'/><br/>
Enter Password : <input type='password' name='password'/><br/>
Enter Employee Id: <input type='text' name='eid'/><br/>
Enter Employee Name:<input type ='text' name='ename'/><br/>
Enter Employee Sal: <input type='text' name='esal'/><br/>
Enter Employee Dept: <input type='text' name="edept"/>
      <input type='submit' value='submit'/>
</form>
</body>
```

```
import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.Filter;

import javax.servlet.FilterChain;

import javax.servlet.FilterConfig;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.ServletRequest;
```

```
import javax.servlet.ServletResponse;
public class MyFilter implements Filter {
	@Override
	public void destroy() {
		System.out.println("Filter destroy()");
	}
	@Override
public void doFilter(ServletRequest req, ServletResponse res,
	FilterChain chain) throws IOException, ServletException {
		System.out.println("Filter doFilter()");
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		out.println("file executing before servlet<br/>");
		String username = req.getParameter("username");
		String password = req.getParameter("password");
		if(username.equals("yaane") &&
password.equals("ramchandra")){
			System.out.println();
			chain.doFilter(req, res);
		}
		else{
			System.out.println("else");
RequestDispatcher rd = req.getRequestDispatcher("./test.html");
			rd.include(req,res);
		}
```

```
            out.println("file executing after servlet<br/>");
        }
        @Override
        public void init(FilterConfig arg0) throws ServletException {
            System.out.println("Filter init()");
        }
}
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.sql.Statement;
import javax.servlet.GenericServlet;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class MyServlet extends GenericServlet {
        Connection con;
        Statement st;
        @Override
        public void init(ServletConfig sc)throws ServletException{
```

```
            System.out.println("init method of myservlet");
            System.out.println("sc: "+sc);
            String driver = sc.getInitParameter("driver1");
            System.out.println("driver: "+driver);
            String url = sc.getInitParameter("url1");
            String username = sc.getInitParameter("username1");
            String password = sc.getInitParameter("password1");
            try {
                  Class.forName(driver);
                  con =
DriverManager.getConnection(url,username,password);
                  st = con.createStatement();
            }
            catch (ClassNotFoundException e) {
                  System.out.println("catch block-CNFE");
                  e.printStackTrace();
            }
            catch (SQLException e) {
                  System.out.println("catch block-SQL");
                  e.printStackTrace();
            }
      }
      @Override
      public void service(ServletRequest req, ServletResponse res)
                  throws ServletException, IOException {
```

```
            res.setContentType("text/html");
            PrintWriter out = res.getWriter();
            out.println("MyServlet is processing the request<br/>");
            int eid = Integer.parseInt(req.getParameter("eid"));
            String ename = req.getParameter("ename");
            int esal = Integer.parseInt(req.getParameter("esal"));
            String edept = req.getParameter("edept");
            try {

                int updatecount= st.executeUpdate("insert into
sbajemp values("+eid+",'"+ename+"',"+esal+",'"+edept+"')");
                out.println("Number Of records inserted:
"+updatecount);
            }
            catch (SQLException e) {
                System.out.println("service-catch");
            }
        }
}
Or
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
```

```
import java.sql.SQLException;


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



public class MyServlet extends GenericServlet {
	Connection con;
	//Statement st;
	@Override
	public void init(ServletConfig sc)throws ServletException{
		System.out.println("init method of myservlet");
		System.out.println("sc: "+sc);
		String driver = sc.getInitParameter("driver1");
		System.out.println("driver: "+driver);
		String url = sc.getInitParameter("url1");
		String username = sc.getInitParameter("username1");
		String password = sc.getInitParameter("password1");
		try {
			Class.forName(driver);
		con = DriverManager.
```

```
                getConnection(url,username,password);
        //st = con.createStatement();
        }
        catch (ClassNotFoundException e) {
                System.out.println("catch block-CNFE");
                e.printStackTrace();
        }
        catch (SQLException e) {
                System.out.println("catch block-SQL");
                e.printStackTrace();
        }
}
@Override
public void service(ServletRequest req, ServletResponse res)
                throws ServletException, IOException {
        res.setContentType("text/html");
        PrintWriter out = res.getWriter();
        out.println("MyServlet is processing the request<br/>");
        int eid = Integer.parseInt(req.getParameter("eid"));
        String ename = req.getParameter("ename");
        int esal = Integer.parseInt(req.getParameter("esal"));
        String edept = req.getParameter("edept");
        try {
                //int updatecount=
```

```
        //st.executeUpdate("insert into sbajemp values(
                    //"+eid+",'"+ename+"',"+esal+",'"+edept+"')");
        PreparedStatement ps =
                    con.prepareStatement(
                              "insert into sbajemp values(?,?,?,?)");
                    ps.setInt(1, eid);
                    ps.setString(2, ename);
                    ps.setInt(3, esal);
                    ps.setString(4, edept);
                    int updatecount = ps.executeUpdate();
        out.println("Number Of records inserted: "+updatecount);
              }
              catch (SQLException e) {
                    System.out.println("service-catch");
              }
        }
}
<web-app>
    <servlet>
        <servlet-name>ms</servlet-name>
        <servlet-class>MyServlet</servlet-class>

        <init-param>
            <param-name>driver1</param-name>
            <param-value>oracle.jdbc.driver.OracleDriver</param-
value>
        </init-param>
        <init-param>
            <param-name>url1</param-name>
```

```
                <param-
value>jdbc:oracle:thin:@localhost:1521:xe</param-value>
            </init-param>
            <init-param>
                <param-name>username1</param-name>
                <param-value>system</param-value>
            </init-param>
            <init-param>
                <param-name>password1</param-name>
                <param-value>manager</param-value>
            </init-param>
        </servlet>
        <servlet-mapping>
            <servlet-name>ms</servlet-name>
            <url-pattern>/ms</url-pattern>
        </servlet-mapping>
        <filter>
            <filter-name>mf</filter-name>
            <filter-class>MyFilter</filter-class>
        </filter>
        <filter-mapping>
            <filter-name>mf</filter-name>
            <url-pattern>/ms</url-pattern>
        </filter-mapping>
</web-app>
```

FilterConfig object is created by servlet container meanwhile of loading MyFilter.class file from secondary memory to primary memory.

Meanwhile creating FilterConfig object by container, it will <param-name> and <param-value> information of <init-param> of <filter> in web.xml file.

Those values will be inject into FilterConfig object. Later that object will be placed into init(FilterConfig fc) of MyFilter class.

We can read these values again form our application by using one method of FilterConfig object that is getInitParameter(-).

Listeners

Liserners are java objects.

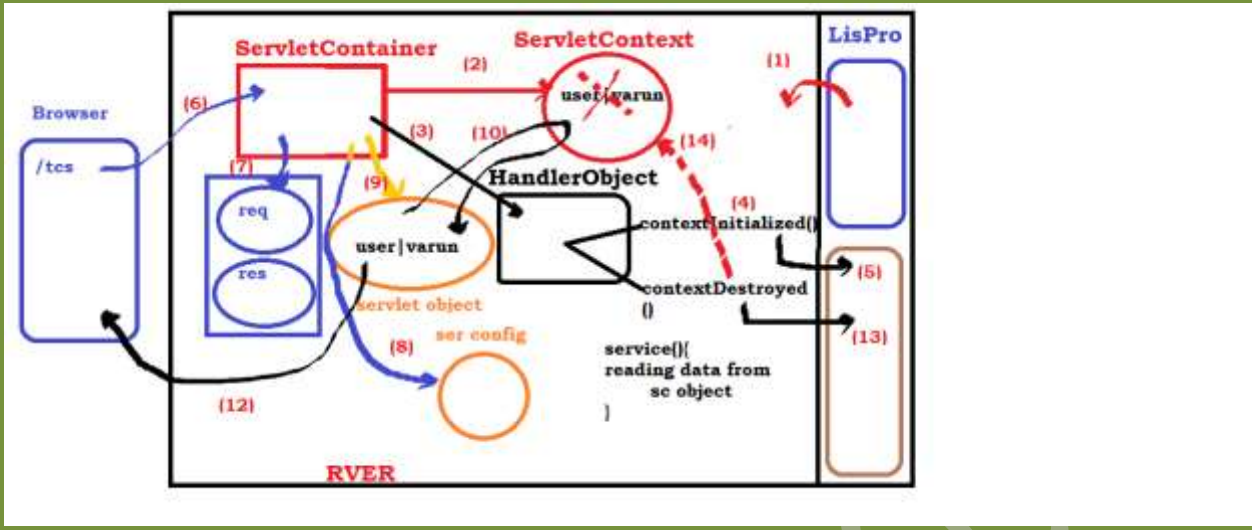
These are listening or getting some information from ServletContainer mean while of scope object creation by container for doing some special operations in background.

Here scope object means session,context,request.

Programon listener:

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

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

public class TestServlet extends GenericServlet{
    @Override
    public void service(ServletRequest req,
                ServletResponse res)
                throws ServletException, IOException {
            System.out.println("servlet1 servicece method");
            res.setContentType("text/html");
            PrintWriter out = res.getWriter();
            ServletContext sc = req.getServletContext();
            String name = (String) sc.getAttribute("name");
            System.out.println("name: "+name);
            out.println("<H1>THIS IS SERVLET-1 RESPONSE");
    }
```

```
}


import javax.servlet.ServletContext;
import javax.servlet.ServletContextEvent;
import javax.servlet.ServletContextListener;


public class ServletContextHandler implements ServletContextListener{

	@Override
	public void contextDestroyed(ServletContextEvent arg0) {
		System.out.println("servletcontext is destroyed");
		ServletContext sc = arg0.getServletContext();
		sc.removeAttribute("name");

	}

	@Override
	public void contextInitialized(ServletContextEvent arg0) {
		System.out.println("servletcontext is created");
		ServletContext sc = arg0.getServletContext();
		sc.setAttribute("name", "ram");


	}

}
<web-app>
<listener>
	<listener-class> ServletContextHandler</listener-class>

</listener>
	<servlet>
		<servlet-name>s1</servlet-name>
		<servlet-class>TestServlet</servlet-class>
	</servlet>
	<servlet-mapping>
		<servlet-name>s1</servlet-name>
		<url-pattern>/test</url-pattern>
	</servlet-mapping>

</web-app>
```



Program on FilterConfig interface:

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


import javax.servlet.Filter;
import javax.servlet.FilterChain;
import javax.servlet.FilterConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class MyFilter implements Filter {
	FilterConfig config;
	@Override
	public void destroy() {
		System.out.println("Filter destroy()");
	}
```

```
	@Override
public void doFilter(ServletRequest req, ServletResponse res,
	FilterChain chain) throws IOException, ServletException {
		System.out.println("Filter doFilter()");
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		String myname = config.getInitParameter("myname");
		out.println("myname: "+myname+"</br>");
		out.println("file executing before servlet<br/>");
		chain.doFilter(req, res);
		out.println("file executing after servlet<br/>");
	}
	@Override
	public void init(FilterConfig fc)
			throws ServletException {
		System.out.println("Filter init()");
		config = fc;
	}
}
import java.io.IOException;
import java.io.PrintWriter;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
```

```
import java.sql.SQLException;


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



public class MyServlet extends GenericServlet {

	@Override
	public void service(ServletRequest req, ServletResponse res)
			throws ServletException, IOException {
		System.out.println("servlet service method");
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		out.println("MyServlet is processing the request<br/>");

	}
}
```

```
<web-app>
	<servlet>
		<servlet-name>ms</servlet-name>
		<servlet-class>MyServlet</servlet-class>
	</servlet>
	<servlet-mapping>
```

```
            <servlet-name>ms</servlet-name>
            <url-pattern>/ms</url-pattern>
      </servlet-mapping>
      <filter>
            <filter-name>mf</filter-name>
            <filter-class>MyFilter</filter-class>
            <init-param>
                  <param-name>myname</param-name>
                  <param-value>ramchandra</param-value>
            </init-param>
      </filter>
      <filter-mapping>
            <filter-name>mf</filter-name>
            <url-pattern>/ms</url-pattern>
      </filter-mapping>
</web-app>
```

How to develop dynamic web project without web.xml file:

To develop DWProject(to communicate with servlet) we required mainly two components.

a. **Web.cml**
b. **Annotation**

To communicating with servlet through annotation we required one annotation like @WebServlet("/url_name");

In the above '/' is mandatory.

Before '/' don't keep dot(.) operator.

Example:

```
<form action="./ts">
      <input type="submit" value="go"/>
</form>
```

import java.io.IOException;

```
import java.io.PrintWriter;


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



@WebServlet("/ts")
public class TestServlet extends GenericServlet{

	@Override
	public void service(ServletRequest req, ServletResponse res)
			throws ServletException, IOException {
		res.setContentType("text/html");
		PrintWriter out = res.getWriter();
		out.println("<h1>This is Test Servlet</h1>");

	}


}
```

DBOperations

DBOperation.html:

```
<html>
	<body bgcolor='yellow'>
		<a href='./Select.html'>SELECT</a><br/>
		<a href='./Insert.html'>INSERT</a><br/>
		<a href='./Delete.html'>DELETE</a><br/>
		<a href='./Update.html'>UPDATE</a><br/>
	</body>
</html>
```

Select.html:

```
<html>
	<body bgcolor='yellow'>
		<form action='./dos' method='get'>
	Enter Employee Number: <input type='text' required name='eid'/>
		<input type='submit' value='SELECT'/>
		</form>
	</body>
</html>
```

Insert.html:

```
<html>
	<body bgcolor='yellow'>
		<form action='./dos' method='POST'>
		Enter Employee Number: <input type='text' required
name='eid'/><br/>
		Enter Employee Name  : <input type='text' required
name='ename'/><br/>
		Enter Employee Salary: <input type='text' required
name='esal'/><br/>
		Enter Employee age   : <input type='text' required
name='eage'/><br/>
		<input type='submit' name='btn' value='INSERT'/>
		</form>
	</body>
</html>
```

Delete.html:

```
<html>
	<body bgcolor='yellow'>
		<form action='./dos' method='post'>
```

```
Enter Employee Number: <input type='text' required name='eid'/>
          <input type='submit' name='btn' value='DELETE'/>
          </form>
     </body>
</html>
```

Update.html:

```
<html>
     <body bgcolor='yellow'>
          <form action='./dos' method='post'>
          Enter Employee Number: <input type='text' required
name='eid'/>
     Enter Employee Update Sal: <input type='text' required
name='esal'/>
          <input type='submit' name='btn' value='UPDATE'/>
          </form>
     </body>
</html>
```

Web.xml:

```
<web-app>
     <servlet>
          <servlet-name>dos</servlet-name>
          <servlet-class>DBOServletDemo</servlet-class>
          <init-param>
               <param-name>driver</param-name>
               <param-value>oracle.jdbc.driver.OracleDriver</param-
value>
          </init-param>
          <init-param>
               <param-name>url</param-name>
          <param-value>jdbc:oracle:thin:@localhost:1521:xe</param-
value>
          </init-param>
          <init-param>
               <param-name>user</param-name>
               <param-value>system</param-value>
          </init-param>
          <init-param>
               <param-name>password</param-name>
```

```
                <param-value>manager</param-value>
            </init-param>
            <load-on-startup>0</load-on-startup>
        </servlet>
        <servlet-mapping>
            <servlet-name>dos</servlet-name>
            <url-pattern>/dos</url-pattern>
        </servlet-mapping>
</web-app>
```

DBOServletDemo.java

```
import java.io.IOException;

import java.io.PrintWriter;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

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 DBOServletDemo extends HttpServlet{

    Connection con=null;

    @Override

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

```
        String driver = sc.getInitParameter("driver");
        String url = sc.getInitParameter("url");
        String user = sc.getInitParameter("user");
        String password = sc.getInitParameter("password");

        System.out.println("driver :"+driver);
        System.out.println("url          :"+url);
        System.out.println("user    :"+user);
        System.out.println("password:"+password);

        try{
             System.out.println("try of init");
             Class.forName(driver);
             con=DriverManager.getConnection(url,user,password);
             System.out.println("con: "+con);
        }catch(Exception e){
             System.out.println("catch block of init");
             e.printStackTrace();
        }
   }
   @Override
   public void doGet(HttpServletRequest request,HttpServletResponse
             response)throws ServletException,IOException{
        response.setContentType("text/html");
        PrintWriter out = response.getWriter();
```

```
            out.println("<body bgcolor='red' ><h1>");




            try{
                  System.out.println("try of doGet");
                  PreparedStatement ps = con.prepareStatement(
                        "select * from employee1 where eid=?");
                  int eid =
Integer.parseInt(request.getParameter("eid"));
                  ps.setInt(1, eid);
                  ResultSet rs = ps.executeQuery();
                  while(rs.next()){
out.println(rs.getInt(1)+"\t"+rs.getString(2)+
            "\t"+rs.getInt(3)+"\t"+rs.getInt(4));

                  }
            }catch(Exception e){
                  System.out.println("catch block of doget");
                  e.printStackTrace();
            }
      }

      @Override
```

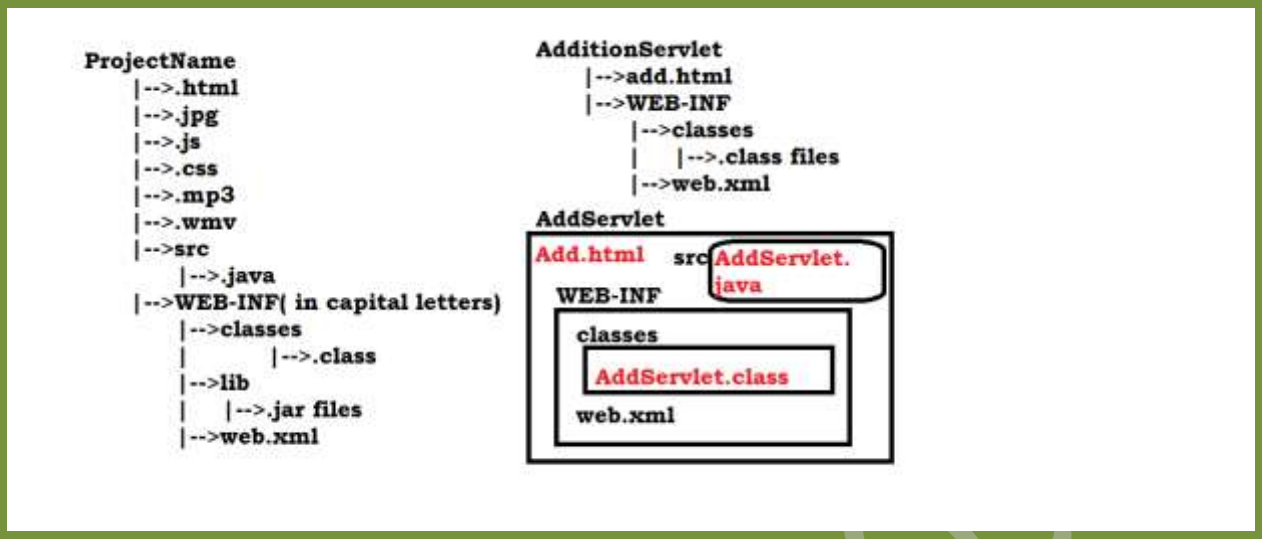
```
     public void doPost(HttpServletRequest request,
HttpServletResponse
                response)throws ServletException,IOException{
           response.setContentType("text/html");
           PrintWriter out = response.getWriter();
           String operation = request.getParameter("btn");
           if(operation.equalsIgnoreCase("insert")){
                int eid =
Integer.parseInt(request.getParameter("eid"));
                String ename = request.getParameter("ename");
                int esal =
Integer.parseInt(request.getParameter("esal"));
                int eage =
Integer.parseInt(request.getParameter("eage"));
                try {
                     System.out.println("try block of dopost-insert");
                     PreparedStatement ps =
con.prepareStatement("insert into employee1 values(?,?,?,?)");
                     ps.setInt(1, eid);
                     ps.setString(2, ename);
                     ps.setInt(3, esal);
                     ps.setInt(4, eage);
                     int count=ps.executeUpdate();
                     out.println(count+".records inserted");
                } catch (SQLException e) {
                     System.out.println("catch of dopost-insert");
```

```
                    e.printStackTrace();
               }



          }
          else if(operation.equalsIgnoreCase("delete")){
               int eid =
Integer.parseInt(request.getParameter("eid"));
               try {
                    System.out.println("try block of dopost-delete");
                    PreparedStatement ps =
con.prepareStatement("delete from employee1 where eid=?");
                    ps.setInt(1, eid);
                    int count=ps.executeUpdate();
                    System.out.println(count+".records deleted");
               } catch (SQLException e) {
                    System.out.println("catch block dopost-delete");
                    e.printStackTrace();
               }
          }
          else if(operation.equalsIgnoreCase("update")){
     int eid = Integer.parseInt(request.getParameter("eid"));
     int usal = Integer.parseInt(request.getParameter("esal"));
     try {
          System.out.println("try block of dopost-update");
```

```
            PreparedStatement ps = con.prepareStatement("update
employee1 set esal=esal+? where eid=?");

            ps.setInt(1, usal);

            ps.setInt(2, eid);

            int count=ps.executeUpdate();

            System.out.println(count+".records updated");

      } catch (SQLException e) {

            System.out.println("catch block of dopost-update");

            e.printStackTrace();

      }

            }


      }



                              }
```

How to work with servlet without eclipse:

FirstCreate project structure like bellow.



Add.html:

```
<html>
	<body bgcolor='yellow'>
		<form action='./as' method='GET'>
	Enter First Number: <input type='text' name='fno'/><br/>
	Enter Second Number: <input type='text' name='sno'/><br/>
		<input type='submit' value='ADD'/>
		</form>
	</body>
</html>
```

Web.xml:

```
<web-app>
	<servlet>
		<servlet-name>as</servlet-name>
		<servlet-class>AddServlet</servlet-class>
	</servlet>
	<servlet-mapping>
```

```
            <servlet-name>as</servlet-name>
            <url-pattern>/as</url-pattern>
      </servlet-mapping>

</web-app>
```

AddServlet.java:

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;

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


public class AddServlet  extends HttpServlet{
      @Override
      public void doGet(HttpServletRequest request,
                  HttpServletResponse response)
      throws ServletException,IOException{
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            out.println("<body bgcolor='yellow'>");
            int i = Integer.parseInt(request.getParameter("fno"));
```

```
            int j = Integer.parseInt(request.getParameter("sno"));
            int result=i+j;
            out.println("Result: "+result);
            out.println("</body>");
        }
}
```