Tommorrow Session(25/01/2023) => From 8.00PM to 11.00PM

Different types of application built using java
==========
 a. Standalone applications(JSE)
            1. CUI applications
            2. GUI applications
 b. Enterprise applications(JEE and frameworks)
            1. Web applications
            2. Distrubuted applications

Difference b/w Web applications vs Distrubuted applications?

webapplication
==========
webapplication is a server side application, it will be designed without distrubuting application logic over multiple jvm.
To build webapplication we need to use technologies like CGI,Servlet,JSP and so on...
The main purpose of webapplication is to generate dynamic response from server machine.
webapplication provides services to webclients only(using the browser we need to send the request)
            eg: BMS(webapp) -------> to send the request to this application u need browser + internet
                                                mobile based apps(internet)
                  Webapplication => client to server model
In case of webapplications to execute the program we need "webservers".

Distrubuted application
==========
Distrubutedapplication is a server side application, it will be designed by distrubuting application logic over multiple jvm.
To build distrubuted application we need to use technolgies like RMI(remote method invocation), EJB's, WebServices,.....
The main purpose of distrubuted application is to establish the communication b/w local machine and remote machine
to access the remote services.
Distrubuted applications will provide service to any type of clients.
                  Distrubutedapplication => buisness to buisness model.
In case of distrubutedapplications to execute the program we need "application servers".

webapplication
==========
      The application which is developed only using web based technologies like html,css,javascript,servlet,jsp etc is
      called "webapplication"

application ===> collection of many programs

webserver
==========
if we want webapplications to execute, we need one special software that special software only we call it as "webserver".
webserve provides environment to run webapplications
            eg: tomcat,resin,jetty,glassfish,jboss,oracleweblogic server,.....

Deployment and UnDeployment

==========
The process of placing the webapplication inside webserver is called "Deployment".

The process of removing webapplication from the webserver is called "UnDeployment".

webclient
========
To send the request from the user we need to have special software installed in the client machine.
The special software is only called as "browser".
eg: mozilla,chrome,safari,.....

web programming for static response
==========
static response
The response which won't be changed from person to person and time to time, such type of response is called as
"static response".
eg: login page of gmail
home page of icici bank

dynamic response
The response which is varied from person to person and time to time, such type of response is called as
"dynamic response".
eg: inbox page of gmail
balance information of icici bank.

Flow diagram of static response
==========
1. client send the request for static files to the server
2. Server searches wheter the requested resource(html file) is available or not.
3. If the request resource(html file) is available then server will provide that file as response.
4. If the requested resource(html file)is not available then we will get 404 status code saying reqeusted resource not available.

Note: To server static files, no processing is required at the server side, hence webserver always loves to serve static files.

web programming for dynamic response
==========
1. client sends the request for webserver
2. webserver will check whether the request is for static resource or dynamic resource(based on url)
3. if it is a static resource, then webserver only will search for static resource,if it is available server will provide the static file
contents(copy and paste) as the response the client.
if it is not available, then 404 status code will be sent as the response to the client saying the requested resource is not
available.

4. if the requested resource is for dynamic information, then webserver will forward the request to webcontainer.
5. webcontainer will search for the helper application, which needs to be executed.
6. if it is not available, then 404 status code will be sent as the response to the client saying the requested resource is not

available.
6. if it is available, then the requested helper application will be executed and it will be sent as the response to the webserver
and webserver inturns will send as the response to the end user.
7. During the execution if any problem occurs then it would reuslt in exception and status code 500 would be sent as a
response to the end user by the server.

Note: To generate the dynamic response at the server side we need some helper applications. To build these applications
which are capable of generating dynamic response we need to learn technologies like

a. CGI
b. Servlet
c. JSP

To design a webapplication, we already have CGI then what is the need to go for Servlet?

CGI => it stands for Common Gateway Interface.
CGI is a server side webtechnology which is built on top of 'c' language, c language is a process based language, which inturn
make CGI as "process based technology".
if we deploy any CGI application,then container will create a seperate process for every request.
Process is heavy weight component, to handle single process system has to consume a lot of memory and execution time.
Due to the above reason, more the request comes server would be getting a burden of creating a process which inturn reduce
the system performance and increase the response time for the client.
To reduce the burden on server and to increase the performance we need to use server side technology called "Servlet".

Servlet is a server side technology which is built on top on "Java language".
Java language is Thread based technology.
if we deploy a Servlet application at the server side then for every request servlet container will generate a seperate thread
on the respecitve Servlet Object.
In the above context, if we increase the no of requests container will create a separte thread instead of process.
When compared to process,threads are light weight,since it is light weight, server would not be burdened.
server would provide quick response for client requests which increase the peformance of the application.

To design a webapplication, we already have Servlet then what is the need to go for JSP?

Servlet
1. To build web applications using Servlet, we need to have knowledge of Java properly.
2. Servlet is mainly meant for Processing logic(pick the request and process the request).
3. Any modification done in the Servlet, then we need to perform compilation and reloading on the server explicitly.
4. If we build webapps using MVC design pattern, then Servlet will placed inside Controller logic.
5. In case of Servlet, we are unable to seperate both presentation logic and buisness logic.

JSP

1. To build web applications using JSP, it is not required to have any java knowledge only presentation skills are enough.
2. JSP is mainly meant for providing dynamic response to the client with good look and feel (only presentation).
3. Any modification done in the JSP, then it is not required to do compilation and reloading becoz jsp pages are "Autocompiled".
4. If we build webapps using MVC design pattern, then JSP will placed inside View logic.
5. In case of JSP, there will be a clear cut seperation b/w presentation logic and buisness logic becoz presentation logic deals with html tags and buisness logic deals with "JSP tags".

Architecture of webserver(tomcat)
==========
Tomcat(https://tomcat.apache.org/download-90.cgi)
1. It is webserver provided by apache foundation.
2. Every webserver will have webcontainer
   a. webcontainer is responsible to manage and execute servlet and jsps.
3. Internally webcontainer consists of 2 components
   a. catalina container(servlet container)
   b. jasper container(jsp container)
4. ServletContainer
   It is also known as ServletEngine.
   It is responsible for managing and executing servlet components.
   Tomcat servlet container name is "CATALINA".
5. JSP container
   It is also known as JSP Engine
   It is responsible for managing and exeucting jsp componenets
   Tomcat jsp container name is "JASPER".

Note:
start the tomcat server by opening bin folder and select tomcat9.exe and double click on it.
now send the request by opening browser of ur choice and hit the request as shown below
   http://localhost:9999/

Servlet
======
   It is an API which helps the programmer to build webapplications.
   ServletAPI provides 2 packages
      a. javax.servlet.*
      b. javax.servlet.http.*

javax.servlet.*
==========
   1. Servlet(I)
   2. GenericServlet(AC)
   3. ServletConfig(I)
   4. ServletContext(I)
   5. RequestDispatcher(I)
   6. ServletRequest(I)
   7. ServletResponse(I)

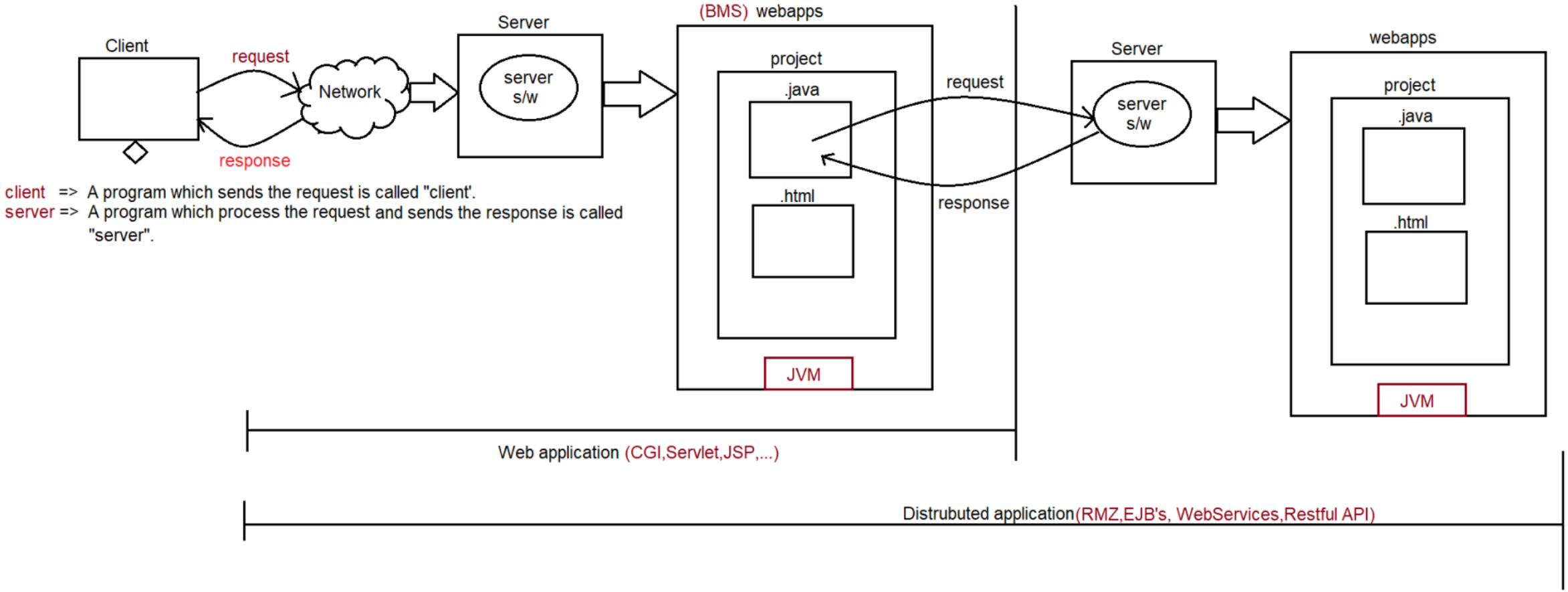
javax.servlet.http.*
==========

1. HttpServletReqeust(I)
2. HttpServletResponse(I)
3. HttpSession
4. HttpServlet(AC)

Note:

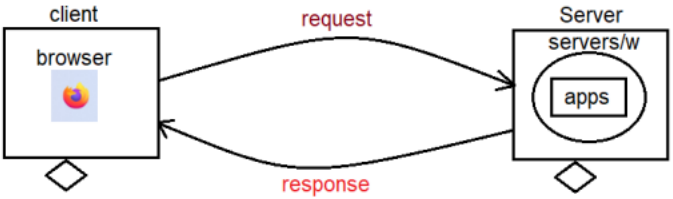
```
Compiled from "Servlet.java"
public interface javax.servlet.Servlet {
  public abstract void init(javax.servlet.ServletConfig) throws
javax.servlet.ServletException;
  public abstract javax.servlet.ServletConfig getServletConfig();
  public abstract void service(javax.servlet.ServletRequest,
javax.servlet.ServletResponse) throws javax.servlet.ServletException,
java.io.IOException;
  public abstract java.lang.String getServletInfo();
  public abstract void destroy();
}

C:\Users\nitin>javap javax.servlet.GenericServlet
Compiled from "GenericServlet.java"
public abstract class javax.servlet.GenericServlet implements
javax.servlet.Servlet,javax.servlet.ServletConfig,java.io.Serializable {
  public javax.servlet.GenericServlet();
  public void destroy();
  public java.lang.String getInitParameter(java.lang.String);
  public java.util.Enumeration<java.lang.String> getInitParameterNames();
  public javax.servlet.ServletConfig getServletConfig();
  public javax.servlet.ServletContext getServletContext();
  public java.lang.String getServletInfo();
  public void init(javax.servlet.ServletConfig) throws
javax.servlet.ServletException;
  public void init() throws javax.servlet.ServletException;
  public void log(java.lang.String);
  public void log(java.lang.String, java.lang.Throwable);
  public abstract void service(javax.servlet.ServletRequest,
javax.servlet.ServletResponse) throws javax.servlet.ServletException,
java.io.IOException;
  public java.lang.String getServletName();
}

C:\Users\nitin>javap javax.servlet.http.HttpServlet
Compiled from "HttpServlet.java"
public abstract class javax.servlet.http.HttpServlet extends
javax.servlet.GenericServlet {
  public javax.servlet.http.HttpServlet();
  protected void doGet(javax.servlet.http.HttpServletRequest,
javax.servlet.http.HttpServletResponse) throws javax.servlet.ServletException,
java.io.IOException;
  protected long getLastModified(javax.servlet.http.HttpServletRequest);
  protected void doHead(javax.servlet.http.HttpServletRequest,
javax.servlet.http.HttpServletResponse) throws javax.servlet.ServletException,
java.io.IOException;
  protected void doPost(javax.servlet.http.HttpServletRequest,
javax.servlet.http.HttpServletResponse) throws javax.servlet.ServletException,
java.io.IOException;
  protected void doPut(javax.servlet.http.HttpServletRequest,
```

```
javax.servlet.http.HttpServletResponse) throws javax.servlet.ServletException,
java.io.IOException;
  protected void doDelete(javax.servlet.http.HttpServletRequest,
javax.servlet.http.HttpServletResponse) throws javax.servlet.ServletException,
java.io.IOException;
  protected void doOptions(javax.servlet.http.HttpServletRequest,
javax.servlet.http.HttpServletResponse) throws javax.servlet.ServletException,
java.io.IOException;
  protected void doTrace(javax.servlet.http.HttpServletRequest,
javax.servlet.http.HttpServletResponse) throws javax.servlet.ServletException,
java.io.IOException;
  protected void service(javax.servlet.http.HttpServletRequest,
javax.servlet.http.HttpServletResponse) throws javax.servlet.ServletException,
java.io.IOException;
  public void service(javax.servlet.ServletRequest, javax.servlet.ServletResponse)
throws javax.servlet.ServletException, java.io.IOException;
  static {};
}
```

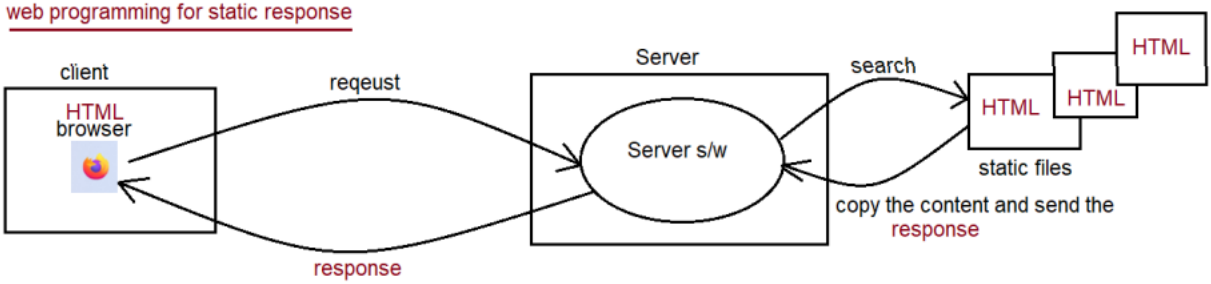


client => A program which sends the request is called "client'.
server => A program which process the request and sends the response is called "server".

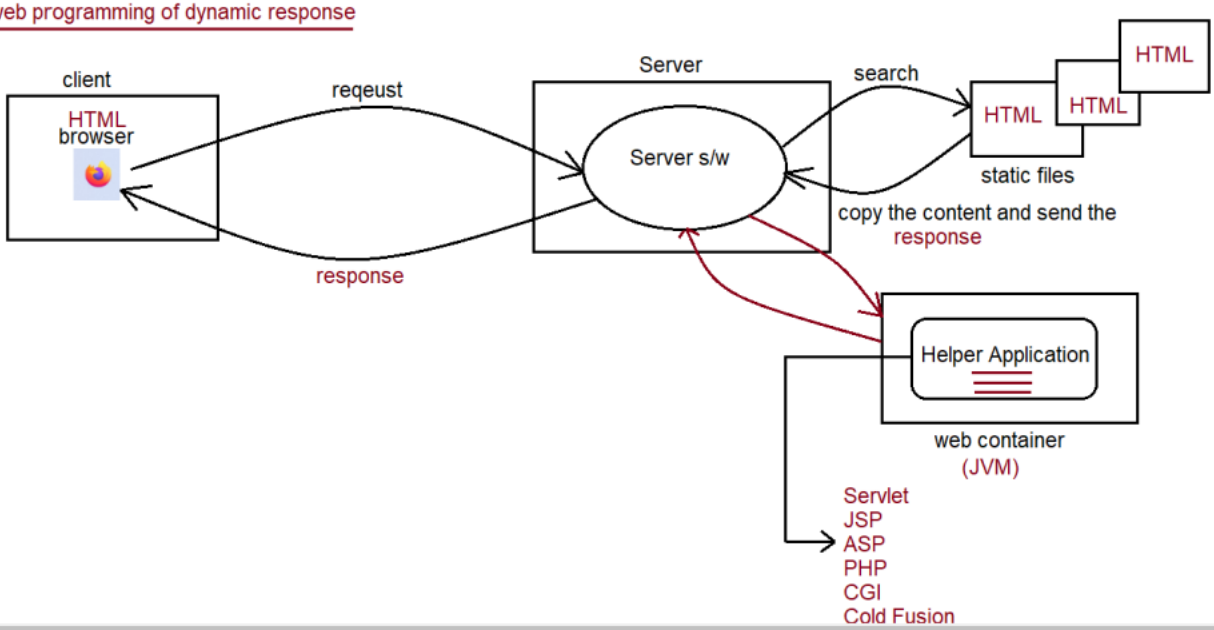
Client-Server Archietcture



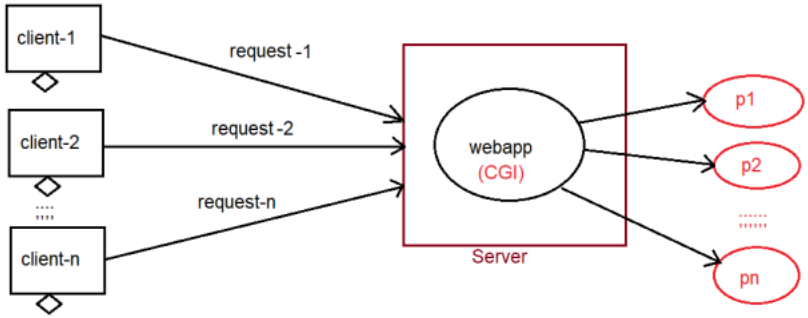
web programming for static response



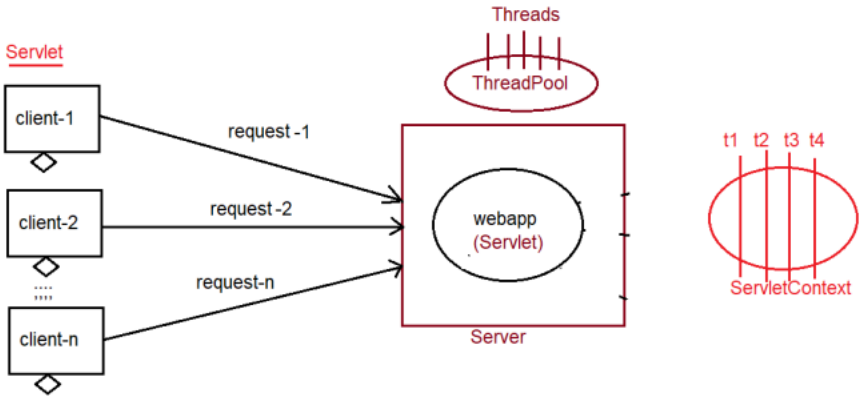
web programming of dynamic response



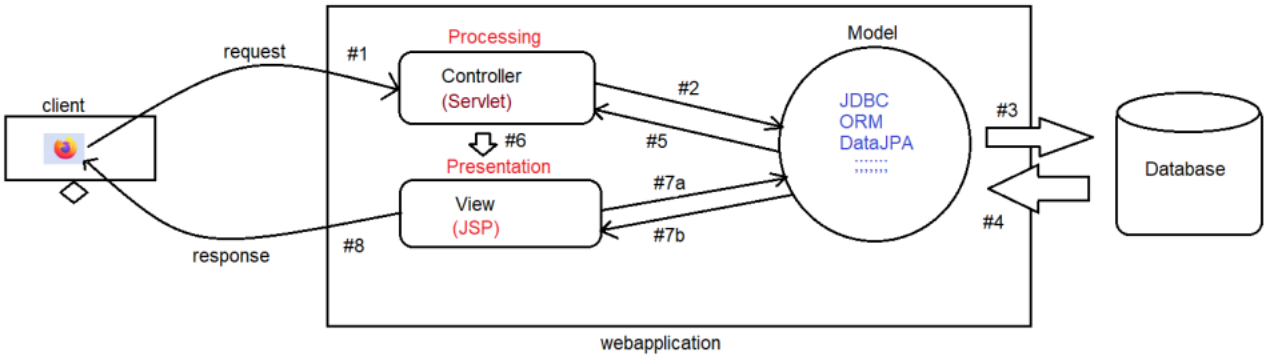
CGI



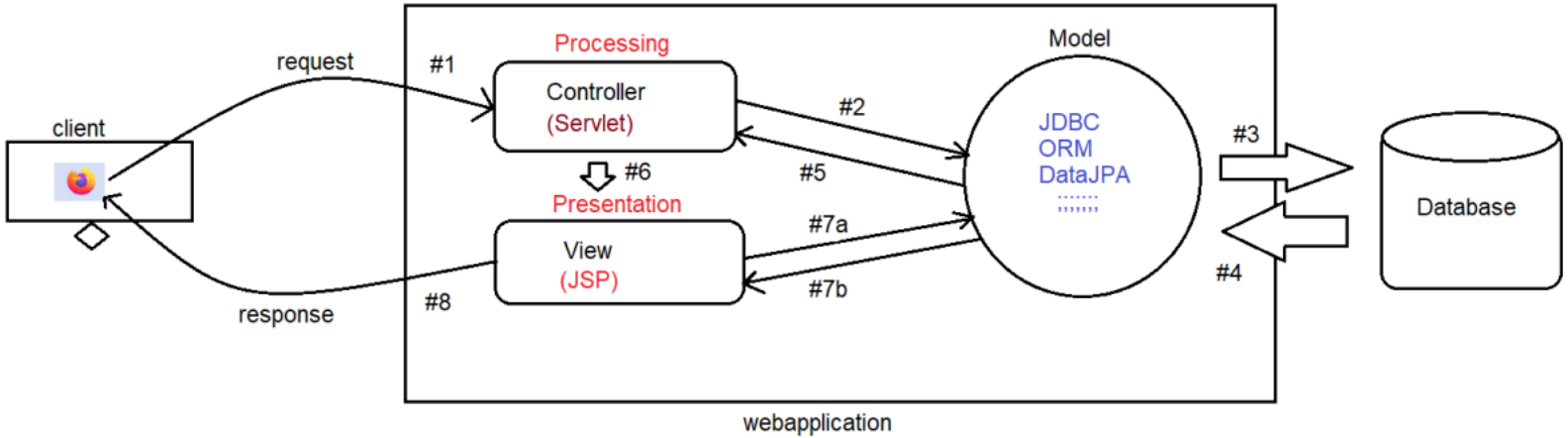
Servlet



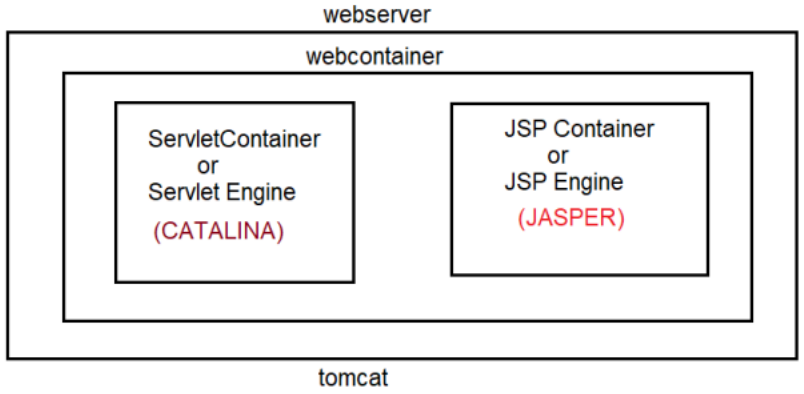
MVC

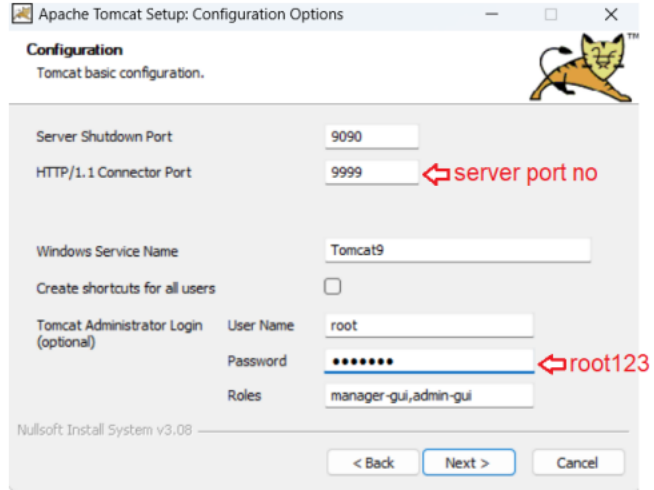


MVC



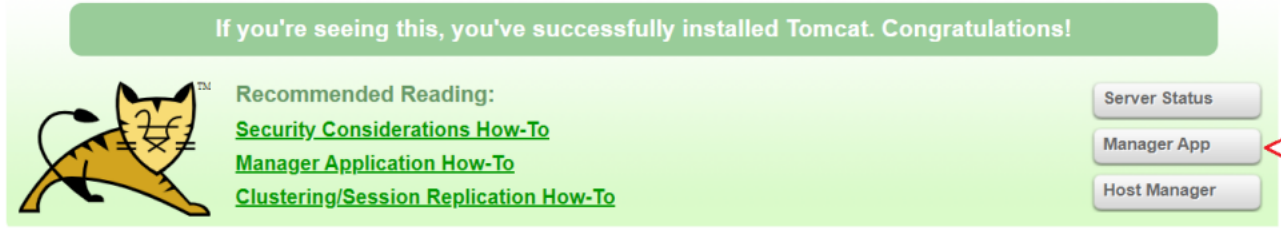
Tomcat Server

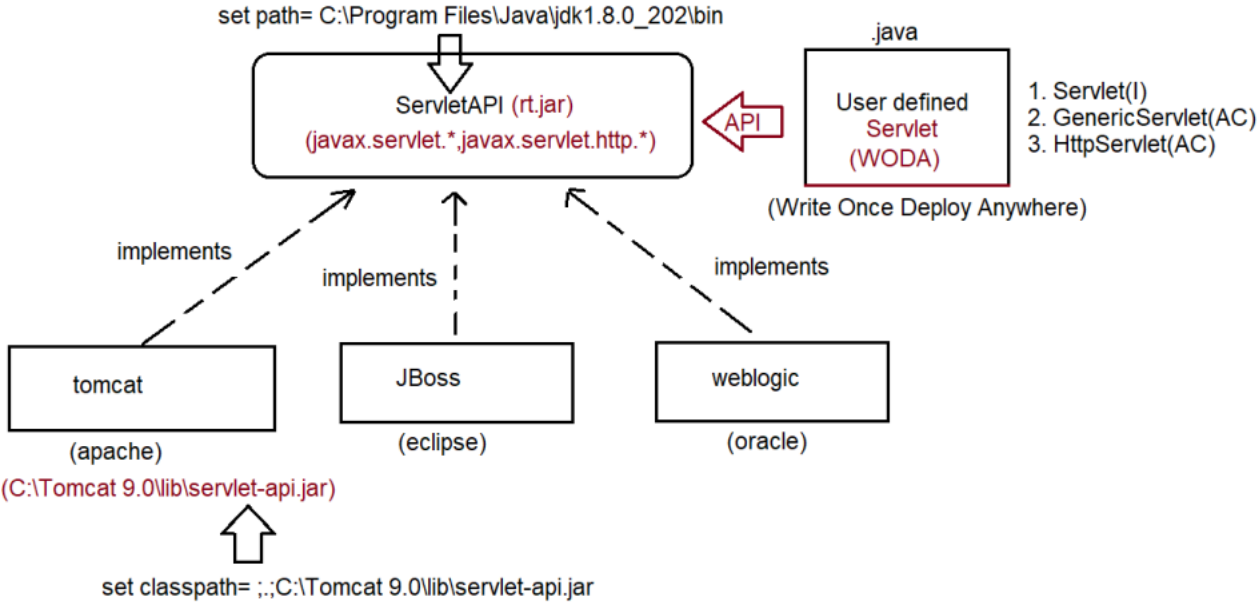




folder structure of tomcat

- bin ⇦ all executable files(.exe)
- conf ⇦ configuration files
- lib ⇦ .jars(related to technology)
- logs ⇦ (request,response details on the user)
- temp ⇦ (temporary files)
- webapps ⇦ Deployment folder(all our projects should be saved here)
- work ⇦ (JSP-SERVLET execution folder)


If you're seeing this, you've successfully installed Tomcat. Congratulations!
Recommended Reading:
Security Considerations How-To
Manager Application How-To
Clustering/Session Replication How-To
Server Status
Manager App
Host Manager
Enter username
Enter password


set path= C:\Program Files\Java\jdk1.8.0_202\bin
ServletAPI (rt.jar)
(javax.servlet.*,javax.servlet.http.*)
API
.java
User defined
Servlet
(WODA)
1. Servlet(I)
2. GenericServlet(AC)
3. HttpServlet(AC)
(Write Once Deploy Anywhere)
implements
implements
implements
tomcat
JBoss
weblogic
(apache)
(eclipse)
(oracle)
(C:\Tomcat 9.0\lib\servlet-api.jar)
set classpath= ;.;C:\Tomcat 9.0\lib\servlet-api.jar

BIG-7 methods
============
  What is the difference b/w GET and POST request type?

GET
      if we want to get some information from the server then we need to for GET request.
            eg:fetching the train ticket details by supllying source and destination information.
      usually get request are read only request and at the server side update operation won't be performed.
      In case of GET request,end user provided information will be attached to the url in the form of "QueryString"
                  eg: http://localhost:9999/App-01/test ?source=bengaluru&destination=shivamoghaa
      In case of GET request,end user data is available inside url as query string, it is less secured.
      In case of GET request,since the data is attached in the url as querystring, only small volume of data can be sent.
      Bookmarking and caching is supported in case of GET request.

POST
      we use POST request to send huge amount of data to the server.
            eg: uploading resume
      usally post request are write only request and at the server side update/insert operation would be performed.
       In case of POST request,end user provided information will be not be attached to the url in the form of "QueryString"
                  eg: http://localhost:9999/App-01/test
      In case of POST request,end user data is not available inside url as query string, so it is more secured.
      In case of POST request,since the data is not attached in the url as querystring, large volume of data can be sent.4
      Bookmarking and caching is not supported in case of POST request.

Note: What is idempotent request?
            By repeating the request multiple times, if there is no change in the response then such type of request we call
            as "idempotent" request.
                        eg: GET request is idempotent request,but POST is not Idempotent.

       What is safe request?
            By repeating the request multiple times,if there is no side effect at the server side then such type of request we call
            as "safe" request.
                        eg:GET request is safe request,but POST is not safe request.

How to send GET request?
      1. Type the address in the url bar and hit enter key(request is sent)
      2. clicking the hyperlink.(CLICK HERE)
      3. submit the form with method attribute as "GET".
            <form method ="GET">
                        <!--- -->
            </form>
      4. submit the form without method attribute(default is GET only)

```
            <form>
                  <!--- -->
            </form>
```

How to send POST request?

1. submit the form with method attribute as "POST".

```
                  eg: <form method ="POST">
                                    <!--- -->
                        </form>
```

Note:

When we send the request,automatically the HTTPProtocol create the HTTPRequest Object.

The relevant information will be assigned in the respective sections of HTTPRequest object.

This HTTPRequest object will be taken by HTTP Protocol and it will send it to the respective Server.

Upon sending the response, automatically the HTTPProtocol will create HTTPResponse Object.

The relevant information will be assigned in the respective sections of HTTPResponse Object.

This HTTPResponse object will be taken by HTTPProtocol to the browser and browser uses this information for

displaying the output.

To build webaplication we need to follow standard directory structure given by Server vendor

==========

```
      ProjectName
            |=> WEB-INF
                        |=> web.xml(deployment descriptor)
                        |=> classes
                                    |=> .class
                        |=> lib
                                 |=> *.jar

             |=> src/main/java
                        |=> .java
             |=> pages
                        |=> .jsp
```

How to create Servlet in Java?

To create Servlet in java there are 3 approaches

a. Servlet(I)
b. GenericServlet(AC)
c. HttpServlet(AC)

Servlet(I)

```
public interface Servlet {
      public abstract void init(ServletConfig config) throws ServletException;
      public abstract ServletConfig getServletConfig();
      public abstract void service(ServletRequest request,ServletResponse response)
throws ServletException, IOException;
      public abstract String getServletInfo();
      public abstract void destroy();
}
```

Whenever we write Servlet,Automatically for the written servlet container will perform the following actions

1. Depending on the url pattern supplied by the user for a dynamic resource
    a. Servlet Loading will happen ===========>static block will be executed.
    b. Servlet Instantiation will happen=======>constructor will be called and object will be created.
    c. Servlet Initialization will happen =======>void init(ServletConfig config) throws ServletException
    d. Serlvet Request Processing will happen==>void service(ServletRequest request,ServletResponse response)

        throws ServletException,IOException
    e. Servlet De-Instantiation will happen ====>void destroy();

eg#1.

```
import java.io.*;
import javax.servlet.*;
public class FirstServlet implements Servlet
{
	static
	{
		System.out.println("FirstServlet.class file is loading...");
	}
	public FirstServlet()
	{
		System.out.println("FirstServlet Object is instantiated...");
	}
	//For Servlet Initialization container calls this method
	public  void init(ServletConfig config) throws ServletException
	{
		System.out.println("Servlet initialziation...");
	}

	public ServletConfig getServletConfig()
	{
		return null;
	}

	//Request Processing logic
	public  void service(ServletRequest request,ServletResponse response)
		throws ServletException, IOException
	{
		System.out.println("Servlet Request Processing ...");
	}

	public  String getServletInfo()
	{
		return null;
	}

	//Servlet DeInstantion logic
	public  void destroy()
	{
		System.out.println("Servlet De-Instantiation...");
	}
}
```

2.

After creating a Servlet, for every servlet url-pattern matching should be provided and it should be informed to the
container via XMl,Annoation.

XML
====

```
<web-app>
      <servlet>
            <servlet-name>DemoServlet</servlet-name>
            <servlet-class>FirstServlet</servlet-class>
      </servlet>
      <servlet-mapping>
            <servlet-name>DemoServlet</servlet-name>
            <url-pattern>/test</url-pattern>
      </servlet-mapping>
</web-app>
```

3. Before compilation set path and classpath environmental variable.

```
            set path= C:\programfiles\jdk1.8\bin
            set classpath=;.;C:\Tomcat 9.0\lib\servlet-api.jar
```

After compilation of FirstServlet.java copy the .class file to classes folder present under WEB-INF/classes.

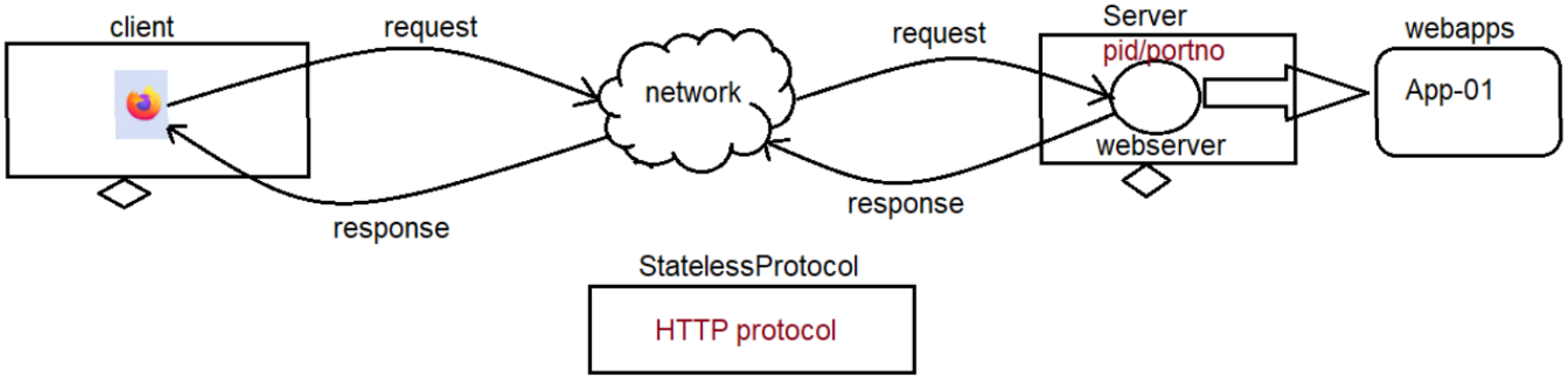
4. Now start the server by going to tomcat9.0/bin/tomcat9.exe file
5. Now send the request by typing the url as shown below

```
            http://localhost:9999/FirstApp/test
```

Protocol

Set of rules and guidenliness follwed by 2 parties for data exchange.

domainname
eg: google
facebook
instagram



Client
a. It will send the request

To send the request, it should follow the protocol(service protocol)

syntax::

service_protocolname :// ipaddressof receiver :portNo_of_the_program/resource_name

- http
- identification_no_of server
- port_no/process_id
- Application_name or Project_name

eg: http://localhost:9999/ App-01

protocol (name of the application)

If a client wants to send the data along with the request how to send?

To send the request along with the data we need to use "QueryString"

QueryString

a. It refers to name-value pair data which will be sent by the user along with request.

For App-01/test

eg: http://localhost:9999/App-01/test?username = sachin&password=tendulkar

QueryString

send name,age and email as the QueryString to App-01/test

eg: http://localhost:9999/App-01/test ? name=sachin&age=49&email=sachin@gmail.com

QueryString is starting

QueryString

From the client side to send the request there are BIG-7 Methods

1. GET
2. POST

"used by developers for sending the request to webapplications"

3. HEAD
4. PUT
5. DELETE
6. OPTIONS
7. TRACE

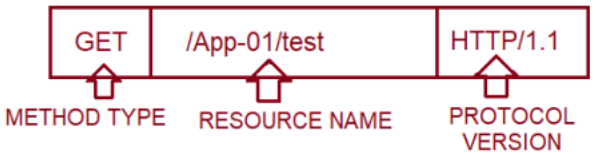
HTTP REQUEST

Request Line
Request Headers
Request Body

HTTP RESPONSE

Status Line
Response header
Response Body

Request Line



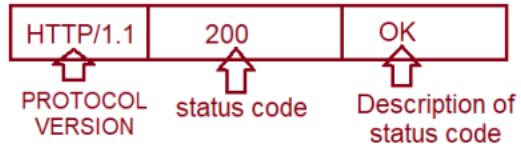
Request Headers

It will carry the information about the browser, its supporting encoding and decoding type,.....

Request Body

In the request body the user supplied input will be sent(QueryString)

Status Line



100 ---- 199 =====> INFORMATION
200 ---- 299 =====> SUCCESS
300 -----399 =====> REDIRECTIONAL
400 -----499 =====> CLIENT ERROR
500 -----599 =====> SERVER ERROR

Response Header

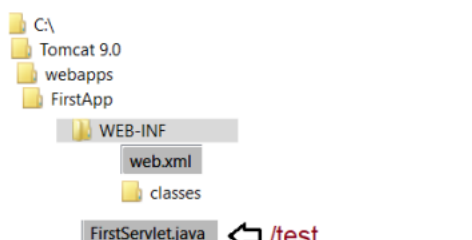
It provides configuration information about the server, content-type,MIME etc,
Browser will use these details to display the response.

Response Body

It holds the acutal response given by the server upon executing a java pgm or static response of html page.

Standard Directory Structure

```
ProjectName
    |--- WEB-INF
            |=> web.xml(deployment descriptor)
            |=> classes
                    |=> .class
            |=> lib
                    |=> *.jar
    |--- src/main/java
                    |=> .java
```



http://localhost:9999/FirstApp/test

FirstServlet

Standard folder structure of webapps to deploy in tomcat server
==========

```
webapps(deployment folder)
      |
SecondApp
     |=> WEB-INF
                 |=> web.xml(deployment descriptor)
                 |=> classes
                              |=> *.class
     |=>  src/main/java
                 |=> .java
```

Any Servlet will be executed by the container with its life cycle actions

a. Loading
b. Instantiation
c. Initialization
d. RequestProcessing phase
e. De-Instantiation

What is the meaning of webapps(deployment folder)?
Once we start the server,server would automatically go to webapp's folder and scans all the project present inside
that folder and deploys the project in the execution area.(meaning ready for providing the service)
Since it scans for all the projects, we say webapp's folder as "Deployment folder".
Once the Tomcat engine loads all the project into execution area,It will create sepearte object for every project
called "ServletContext" object.
Tomcat engine also scans web.xml file given by the user w.r.t every project.
It read the url-pattern for all the dynamic resource for future usage.

How to send the request to any application?
using the url pattern
eg:
http://localhost:9999/[NameOfTheApplication/ContextRoot]/[url_pattern of the resource]

Assume the request is sent for a particular url_pattern, then what actions will be taken care by tomcat engine?
eg: http://localhost:9999/SecondApp/demo

Step1: Browser will send the request with the follwing url pattern
http://localhost:9999/SecondApp/demo

Step2: HttpProtocol will create a HttpRequest object depending upon the Request_TYPE to carry the request
data from client to server.

Step3: Once the Protocol hands over the HttpRequest Object to the tomcat server, server will pick up the
RequestLine from the HttpRequestObject and take only ContextName/urlpattern for furthe processing.

Step4: Depending upon the dynamic resource identified by urlpattern,tomcat engine will hand over the control
to container for execution.
Container will scan for web.xml file and identifies for a particular url_pattern which servlet should be

executed,based on that the servlet will be executed by the container.

Step5: As per the container life cycle actions, the requested urlpattern servlet will be executed.

1. Servlet Loading(.class file loading)
   Class c = Class.forName("SecondServlet");
2. Servlet Instantiation(for loaded class create an Object)
   SecondServlet obj=(SecondServlet)c.newInstance();
3. Servlet Initalization(for the created object inject the required values)
   obj.init(ServletConfig config);
4. Request Processing phase(for client request this method will be called)
   obj.service(ServletRequest request,ServletResponse response);

Note:
To get the inputs supplied by the user to the Servlet, we need to use "ServletRequest" Object.
To write the output from the application to the browser, we need to use "ServletResponse" Object.
Inside the ServletResponse object we have empty "PrintWriter" object,using which we need to write the
Output to browser window.
If we use System.out.println() in the request processing logic, then output will be available on the console of
the tomcat engine.

5. Once the server is stopped(undeployment)/for the same resource if the request wont' come for some
   period of time automatically container will execute "De-Instantiation" event.
   obj.destroy()

Before sending the request, we need to keep the compiled code in WEB-INF/classes folder

```
C:\Tomcat 9.0\webapps\SecondApp>set path=C:\Program Files\Java\jdk1.8.0_202\bin
C:\Tomcat 9.0\webapps\SecondApp>set classpath=C:\Tomcat 9.0\lib\servlet-api.jar
C:\Tomcat 9.0\webapps\SecondApp>javac SecondServlet.java
```

Once the compiled code available then , we need to start the server

a. c:\tomcat9.0\bin\tomcat9.exe

Output For First Request

```
==================
SecondServlet .class file is loading...
SecondServlet Object is instantiated...
Servlet initialziation...
Servlet Request Processing ...
```

Output for Second Request

```
======================
Servlet Request Processing ...
```

As we noticed above, the processing time for second request is less when compared to first request becoz only
requestprocessing logic is executed.

How to maintain the uniformity of response time for all the request?
Ans.To uniform response time for all the request we need to configure the tomcat engine.
Tomcat engine can be configured in 2 ways
a. XML(<load-on-startup>any+ve number<load-on-startup>)
b. Annotation

To maintain the uniform response time we need to use <load-on-startup> tag.

```
<web-app>
      <servlet>
            <servlet-name>DemoServlet</servlet-name>
            <servlet-class>SecondServlet</servlet-class>
            <load-on-startup>10</load-on-startup>
      </servlet>
      <servlet-mapping>
            <servlet-name>DemoServlet</servlet-name>
            <url-pattern>/demo</url-pattern>
      </servlet-mapping>
</web-app>
```

At the time of Server startup
==========
SecondServlet .class file is loading...
SecondServlet Object is instantiated...
Servlet initialziation...

Output for Firstrequest
==========
Servlet Request Processing ...

Output for Second Request
==========
Servlet Request Processing ...

Servlet Code w.r.t Annoatation
==========
Use the following annotations on the top of Servlet class as shown below.

```
@WebServlet(urlPatterns="/test",loadOnStartup = 10)
```

Limitation of implementing Servlet interface
==========
If we create a servlet using Servlet(I), then it is mandatory for us to give the implementation for all the methods
of the interface whether it is required or not.
Becoz of this length of the code would increase and it decreases the readability.
To Overcome this problem we need to use "GenericServlet".

GenericServlet has already implemented Servlet Interface and it gives body for all the methods of Servlet interface
except service().
if we use GenericServlet to create a Servlet, then we need to give body only for service() as a result of which the
lines of code would be less, which increase the readablity of the application.

Compiled from "GenericServlet.java"

```
public abstract class GenericServlet implements Servlet,ServletConfig,Serializable
{
  public GenericServlet();
  public void init(ServletConfig config) throws ServletException;
  public void destroy();
  public ServletConfig getServletConfig();
  public ServletContext getServletContext();
  public String getServletInfo();

  public void init() throws javax.servlet.ServletException;
  public abstract void service(ServletRequest req, ServletResponse resp) throws
ServletException,IOException;
  public java.lang.String getServletName();
}
```

Note:
GenericSevlet is an best example for "Adapter class design pattern".
init() is overloaded in GenericServlet.

Note:
By default response type/content type is "text/html".

Code to demonstrate the creation of Servlet using GenericServlet
==========

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.*;

@WebServlet(urlPatterns="/disp")
public class FourthServlet  extends GenericServlet
{
      public  void service(ServletRequest request, ServletResponse response) throws

            ServletException,IOException
      {
                  PrintWriter out = response.getWriter();
                  out.println("<h1 style='color:blue;'>Writing Servlet using
Generic Servlet is easy....</h1>");
                  out.close();
      }
}
```

Behind the scenes
==========

In the above code 2 .class files will be used
a. FourthServlet.class
b. GenericServlet.class

=> Loading :: Container will load FourthServlet.class file for the url pattern ("/disp")
=> Instantiation :: Container will create an Object for FourthServlet.class
=> Initialization :: Container will call init(),First it will check in FourthServlet.class if not, it would check in GenericServlet.
init() is available inside GenericServlet but it has 2 methods with the name
init(SC config)
init()
container will call init(SC config) which internally makes a call to init().

Can we override the init logic?
We can override, but it is a good practise to override only init(),but not init(SC config) becoz config is local variable
in init(SC config),and the config variable memory would be gone once the control comes out of the init(SC config)
so better override init() but not init(SC config).

Code in GenericServlet
==========
```
public abstract class GenericServlet implements Servlet,ServletConfig,Serializable
{
        private transient SC config;
        public void init(SC config)throws SE
        {
                this.config=config;
                init();
        }
        public void init() throws SE
        {
                ;;;;;;
        }
        ;;;;;;;;
}
```

=> RequestProcessing phase :: Container will call service(req,resp) to provide response to the client.
First it will check in FourthServlet.class if not, it would check in GenericServlet.
service(req,resp) is available inside GenericServlet as abstract and we need to give the body
of this method inside FourthServlet as shown in the above program.

=> ServletDeInstantion=> Contianer will call destroy() to perform De-Instantion action.
First it will check in FourthServlet.class if not, it would check in GenericServlet.
destroy() is not avaialble in FourtServlet.class,so it would take from GenericServlet.class and it will execute.

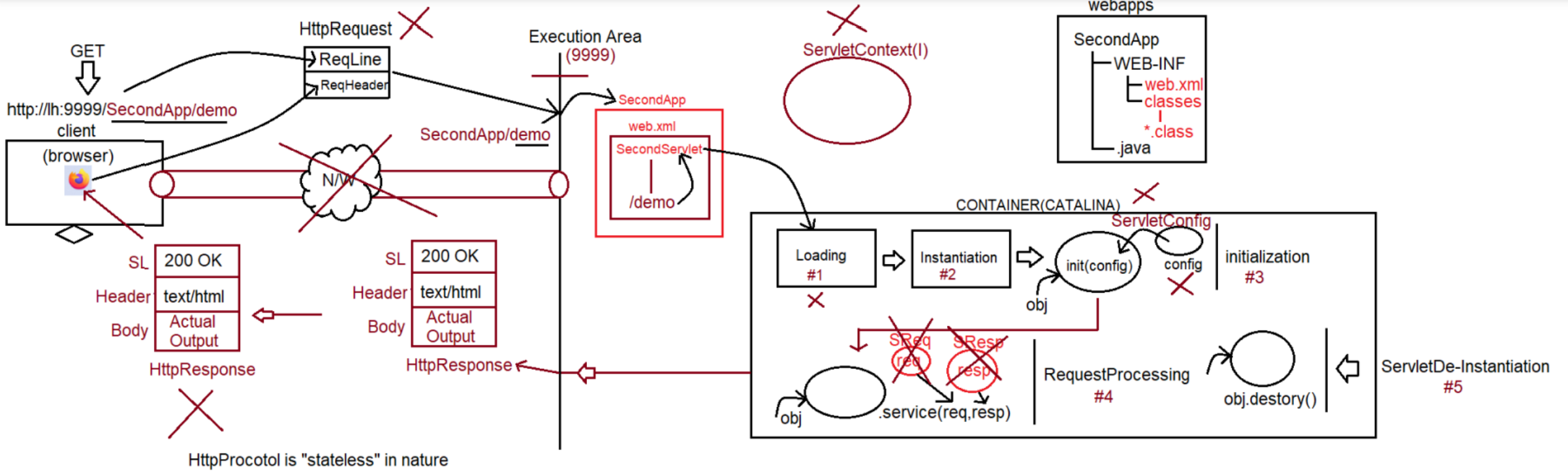
Note:
1. If our servlet class does not contains init() method
    a. GS: init(SC)
    b. GS: init()
    c. US: service(req,resp)

2. If our servlet class contains init(SC) method
    a. US: init(SC)
    b. US: service(req,resp)

3. If our servlet class contains init() method
    a. GS: init(SC)
    b. US: init(SC)
    c. US: service(req,resp)

why 2 init(),init(SC config) in GenericServlet?

```
            init(SC config) -> container
            init() -> developer

Which init method is best suited for developer?
            init() => best suited for writing initialziation logic.
```


GET
http://lh:9999/SecondApp/demo
client
(browser)
HttpRequest
ReqLine
ReqHeader
N/W
Execution Area
(9999)
SecondApp/demo
SecondApp
web.xml
SecondServlet
/demo
ServletContext(I)
webapps
SecondApp
WEB-INF
web.xml
classes
*.class
.java
CONTAINER(CATALINA)
Loading
#1
Instantiation
#2
init(config)
ServletConfig
config
initialization
#3
obj
SReq
req
SResp
resp
obj
.service(req,resp)
RequestProcessing
#4
obj.destory()
ServletDe-Instantiation
#5
SL
200 OK
Header
text/html
Body
Actual Output
HttpResponse
SL
200 OK
Header
text/html
Body
Actual Output
HttpResponse
HttpProcotol is "stateless" in nature

Different ways of Creating a Servlet
==========
1. Servlet(I) ==========> 5 abstract methods
2. GenericServlet(AC) ==> one abstract method (pvs(ServletRequest request, ServletResponse response)

Note:
When we build webapplications, internally httpprotocol is used and while sending the request , the request type can be
a. POST
b. GET

We can build Servlet in a easier way with the help of GenericServlet, then y need HttpServlet(AC)?
Ans. In case of GenericServlet(AC), to process the request we have only one method(pvs(req,resp)) which is generic for
any type of request like GET,POST,....
Becoz there is only one method available which is generic for any type of request,Debugging the application becomes
difficult.
henceforth to deal with only Httpprotocol, we have a special approach to create a servlet called "HttpServlet".

Note:
C:\Users\nitin>javap javax.servlet.http.HttpServlet
Compiled from "HttpServlet.java"
public abstract class HttpServlet extends GenericServlet {

protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException,IOException;
protected void doPost(HttpServletRequest request,HttpServletResponse response) throws ServletException,IOException;

protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException,IOException;
public void service(ServletRequest request, ServletResponse response) throws ServletException,IOException;

}

Note:
request method type is GET ==========> doGet(request,response)
request method type is POST ==========> doPost(request,response)

Note:
For a webapplication how to send GET request?
a. type url in the address bar of browser and hit the request.
b. by clicking the hyperlink in a webpage.
c. using <form method="GET">
d. default <form> method attribute is GET only.

For a webapplication how to send POST request?
a. using <form method="POST">

Life Cycle of HttpServlet
==========
1. when we submit the form browser prepares HttpRequest and sends to server.

2. WebServer checks the request is for static/dynamic information.
3. If the request is for Static information then webserver provides required information(copy and paste)
     if available otherwise 404 Status Code(Saying the requested resource is not available).
4. If the request is for dynamic information, then webserver hands over the control to "catalina" container.
5. wecontainer identifies the request based on "web.xml" or through "annotation". (/test)
6. webcontainer will check whether the ServletObject(TestServlet) is available or not (/test====> TestServlet.class)
7. If the Servletobject(TestServlet.class) is not available, then it will perform the following action
            a. Loading ===> static block
            b. instantiation ===> constructor
            c. initialization ===> init()[same as GenericServlet lifecyle]
8.RequestProcessing phase
      a. webcontainer will create ServletRequest,ServletResponse object by invoking

            public void service(ServletRequest request, ServletResponse response) throws ServletException,IOException;
       container will check in our class service(ServletRequest,ServletResponse) is available or not, if it is avaialble it will
       execute our service() only and provides the response.
        if our servlet class does not contain service(), then container will execute HttpServlet
            public void service(ServletRequest request,ServletResponse response)

HttpServlet
=========
public void service(ServletRequest request,ServletResponse response)
{
             HttpServletRequest hreq = (HttpServletRequest)request;
             HttpServletResponse hresp = (HttpServletResponse)response;
            service(hreq,hresp);//protected void service(hreq,hresp)
}
webcontainer will call protected service(HttpServletRequest hreq,HttpServletResponse hresp) throws SE,IOE
if our class contains protected void service(HttpServletRequest hreq,HttpServletResponse hresp) throws SE,IOE
then container will call our class service() only.
if our class doesnot contains protected void service(HttpServletRequest hreq,HttpServletResponse hresp) throws SE,IOE
then container will HttpServlet class service().

HttpServlet
=========
protected void service(HttpServletRequest request, HttpServletResponse response) throws ServletException,IOException
{
      String requestType = request.getMethod();
      if(requestType.equals("GET")){
                  doGet(request,response);//protected/public void doGet(request,response);
      }
      else if (requestType.equals("POST")){
                  doPost(request,response);//protected/public doPost(request,response);
      }

```
      ;;;;;
      ;;;;
      else
            return 501 status code saying Http method not implemented
}
```
webcontainer will check whether our servlet class contains doXXXX(req,resp).
if it contains doXXXX(req,resp) it will be executed and it provides the response.
if our servlet class does not contain doXXXX(req,resp) then HttpServlet class doXXXX(req,resp) will be called.

HttpServlet
=========
```
protected void doXXXX(HttpServlet request,HttpServletResponse response ) throws SE,IOE
{
      return 405|400 status code saying HttpMethod GET is not supported by this URL.
}
```

Note:
Hierachy of calling the methods

a. public service(SRreq,SResp)
b. protected service(HSReq,HSResp)
c. public void doXXXX(HSReq,HSResp)

case1:
   If our servlet class contains public void service(SReq,SResp) then for every type of request(POST,GET)
   same method will be executed.

case2:
   If our servlet class contains public void service(SReq,SResp) and protected void servcie(HSReq,HSResp)
   then for every type of request(POST,GET) public void service(SReq,SResp) same method will be executed.

case3:
   If our servlet class contains protected void servcie(HSReq,HSResp) and doGet() then for every type of
   request(POST,GET) protected void service(HSReq,HSResp) same method will be executed.

case4:
       we are sending GET request,but our servlet doesnot contain doGet(),it contains doPost() then which method would be
        called?
            it calls HttpServlet doGet() which would send 405 status code.

case5:
       we are sending POSt request,but our servlet doesnot contain doPost(),it contains doGet() then which method would be
        called?
            it calls HttpServlet doPost() which would send 405 status code.

case6:
      Assume we need to give common response for both the request type, then how to code?

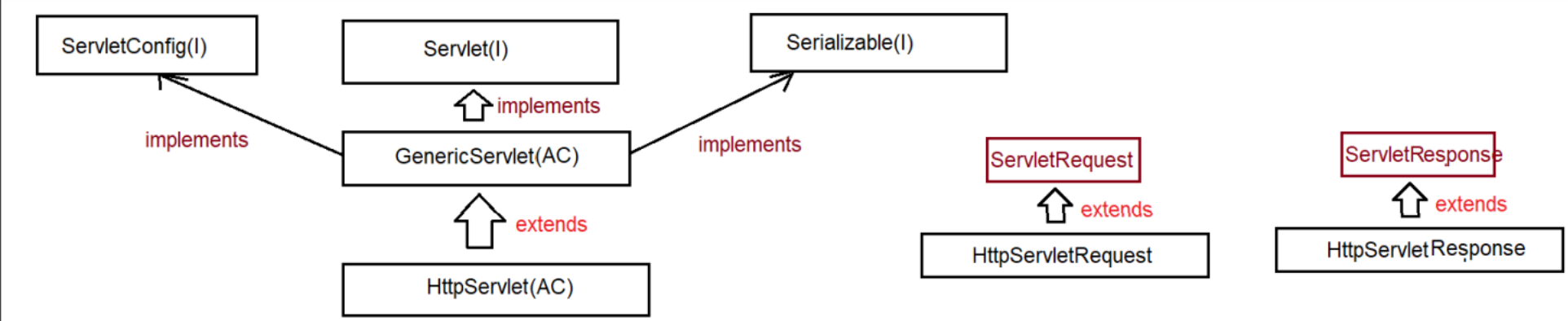
eg:

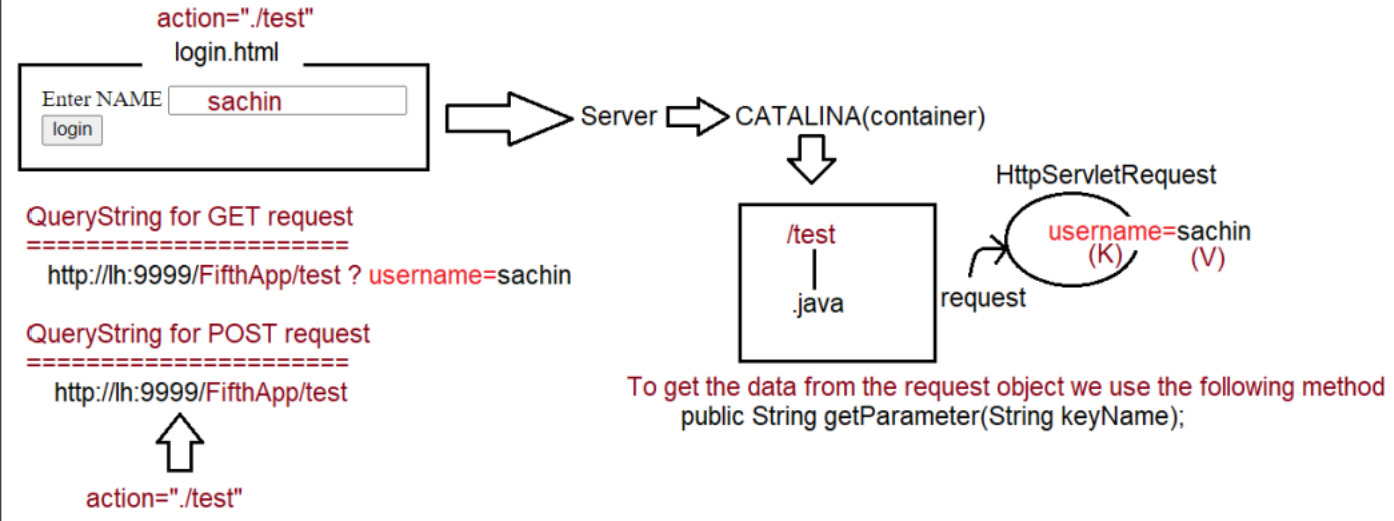
```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
import java.io.*;

@WebServlet(urlPatterns="/test")
public class TestServlet extends HttpServlet
{
      @Override
      public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException,IOException
      {
            doProcess(request,response);
      }
      @Override
      public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException,IOException
      {
            doProcess(request,response);
      }
      public void doProcess(HttpServletRequest request,HttpServletResponse
response)throws ServletException,IOException
      {
            System.out.println("Request method is of type ::
"+request.getMethod());
            String userName = request.getParameter("username");
            System.out.println("username is :: "+userName);
      }
}
```

```
Playing with request Object
=======================
To retrieve only one value from request object
   public abstract java.lang.String getParameter(java.lang.String);

To retrive multiple values from request object
      public abstract java.lang.String[] getParameterValues(java.lang.String);

To know the type of request from request object
        public abstract java.lang.String getMethod();
```

```
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.*;
import java.io.*;

@WebServlet(urlPatterns="/reg")
public class TestServlet extends HttpServlet
{

      @Override
      public void doPost(HttpServletRequest request, HttpServletResponse response)
throws ServletException,IOException
      {

            response.setContentType("text/html");

            System.out.println("Request type is :: "+request.getMethod());
```

```
            String username  = request.getParameter("username");
            String useremail = request.getParameter("useremail");
            String useraddr  = request.getParameter("useraddr");
            String[] courses = request.getParameterValues("course");

            PrintWriter out = response.getWriter();
            out.println("<html><head><title>OUTPUT</title></head>");
            out.println("<body>");
            out.println("<center>");
            out.println("<h1>Student Registration details</h1>");
            out.println("<table border='1'>");
            out.println("<tr><th>NAME</th><td>"+username+"</td></tr>");
            out.println("<tr><th>EMAIL</th><td>"+useremail+"</td></tr>");
            out.println("<tr><th>ADDR</th><td>"+useraddr+"</td></tr>");
            out.println("<tr><th>COURSE</th>");
            String data = "";
            for(String course: courses)
                  data +=course+" ";
            out.println("<td>"+data+"</td>");
            out.println("</tr>");
            out.println("</table>");
            out.println("</center>");
            out.println("</body>");
            out.println("</html>");

            out.close();
      }
}
```


ServletConfig(I)
Servlet(I)
Serializable(I)
implements
implements
implements
GenericServlet(AC)
extends
HttpServlet(AC)
ServletRequest
extends
HttpServletRequest
ServletResponse
extends
HttpServletResponse


action="./test"
login.html
Enter NAME
sachin
login
Server
CATALINA(container)
/test
.java
HttpServletRequest
username=sachin
(K)
(V)
request
QueryString for GET request
http://lh:9999/FifthApp/test ? username=sachin
QueryString for POST request
http://lh:9999/FifthApp/test
action="./test"
To get the data from the request object we use the following method
public String getParameter(String keyName);

Different ways of Creating a Servlet
==========
1.Servlet(I)
2.GenericServlet(AC)
3.HttpServlet(AC)

Dynamic response will be generated by Servlet.
Mapping a resource to particular urlpattern for the webcontainer can be configured in 2 ways
a. XML(legacy approach)
b. Annotation(Available from Servlet3.0V)

Note:
User input will be sent in the form of QueryString from the browser to protocol, and container will store in request object.
ServletRequest(I)
| extends
HttpServletRequest(I)

HttpRequest Structure
a.RequestLine(request type,resourcename,protocolversion)
b.RequestHeader(information about the client)
c.RequestBody(actual data(QueryString))

Methods
=======
public abstract Enumeration<String> getHeaderNames();
public abstract String getHeader(String keyName);

refer: RequestHeaderApp

Working with ServletResponse/HttpServletResponse
==========
1. public abstract PrintWriter getWriter() throws IOException;
To send character type of Response

2. To send binary information like videos,audio,images etc response will be through "Stream".
public abstract ServletOutputStream getOutputStream() throws IOException;

refer: ResponseInfoApp

Note:
PrintWriter => only character type of data.
ServletOutputStream => we can send both character type and binary type of data.

Q> In single application can we use both PrintWriter object and ServletOutputStream to send the response?
Ans. Not possible, it would result in "java.lang.IllegalStateException: getWriter() has already been called for this response".

MultiThreading technology used in servlet
==========
refer: diagram....

ServletContext(I) vs ServletConfig(I)
==========

ServletConfig(I)
==========
   1. Loading   ======> static block
   2. Instantiation ===> Object obj = Class.forName(String urlPattern).newInstance();
                                  Object will be created for the loaded class using Zero parameterized constructor.
   3. Initialization ====> For the created Servlet object, we can initialize the value using init(ServletConfig config)

ServletConfig
==========
      This object will be used to initialize the values for the loaded servlet.

How to initialize the value of ServletConfig object?
  To initalize the value of ServletConfig object, we need to configure the container.
  Container can be configured in 2 ways
      a. XML

web.xml

```
<servlet>
       <servlet-name>InitializationParamterApp</servlet-name>
        <servlet-class>in.ineuron.controller.InitializationParamterApp</servlet-class>
      <init-param>
            <param-name>name</param-name>
            <param-value>sachin</param-value>
      </init-param>
      <init-param>
            <param-name>age</param-name>
            <param-value>49</param-value>
      </init-param>
      <init-param>
            <param-name>address</param-name>
            <param-value>MI</param-value>
      </init-param>
</servlet>
```

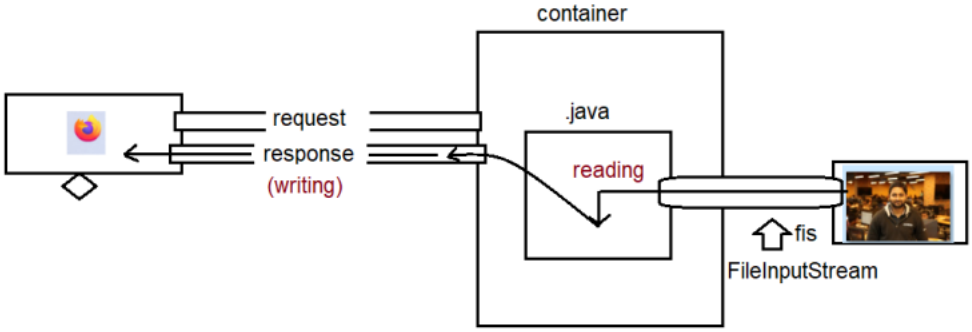
methods
==========
```
public  Enumeration<String> getInitParameterNames();
public  String getInitParameter(String key);
```
                refer: ServletConfigApp

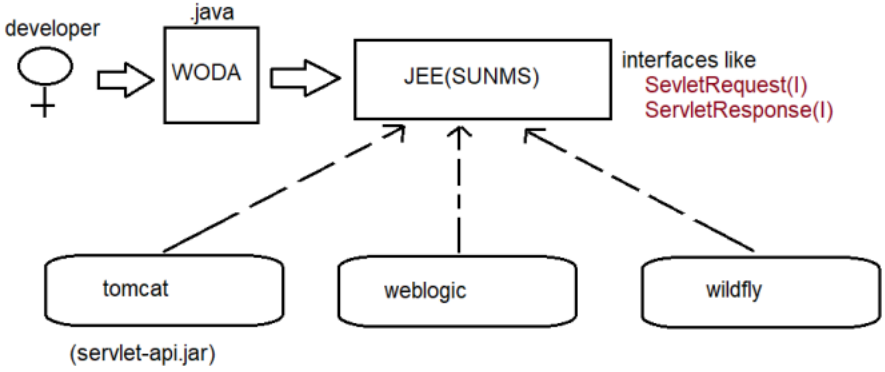
Note:
      => ServletConfig object is unique w.r.t every Servlet.
      => ServletConfig object stores the data in the form of Key,Value pair.
      => Assume we want the dbconfiguration details for a servlet like
                url => jdbc:mysql:///octbatch
                 username=> root
                    password => root123

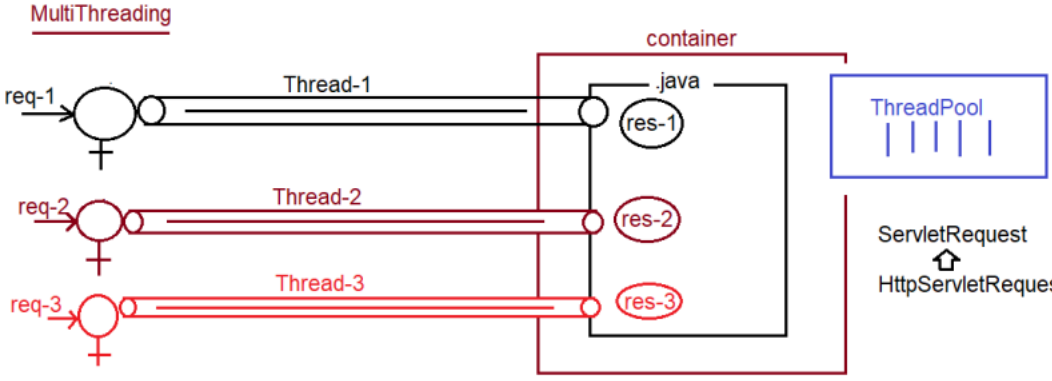
After adding the jar, jars will available to jdks/w configure for eclipse, not for tomcat server.
To make those jars available for tomcat server(catalina container) we need to put in deployment assembly(/WEB-INF/lib)

refer: ServletDBCommunication

Assignment
==========
Create a html form with fields like name,age,address for student
a. perform insertion operation(use db credentials in config object)


container
request
response
(writing)
.java
reading
fis
FileInputStream


developer
.java
WODA
JEE(SUNMS)
interfaces like
SevletRequest(I)
ServletResponse(I)
tomcat
weblogic
wildfly
(servlet-api.jar)

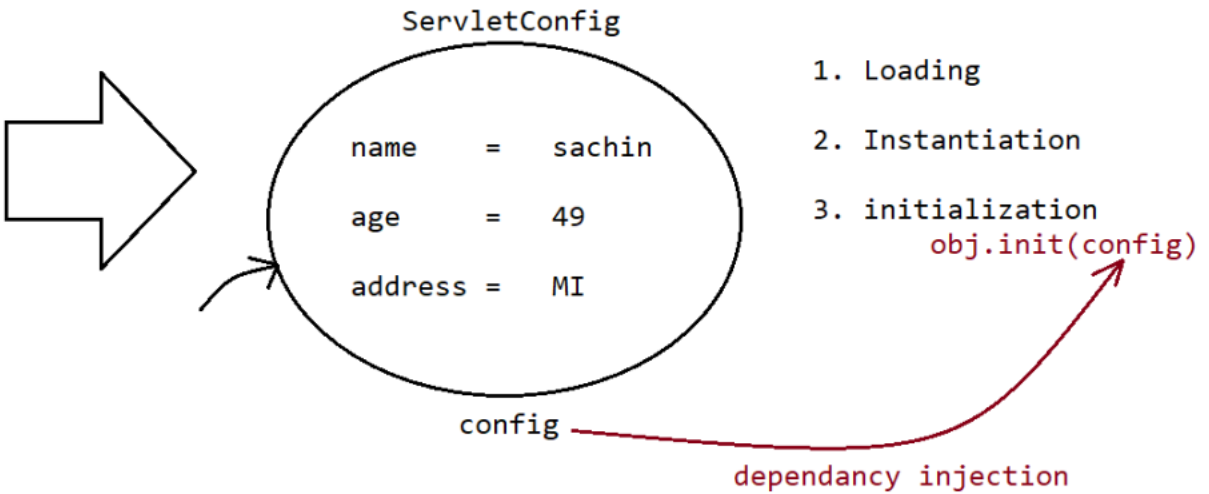

MultiThreading
container
req-1
Thread-1
.java
res-1
ThreadPool
req-2
Thread-2
res-2
ServletRequest
HttpServletRequest
ServletResponse
HttpServletResponse
req-3
Thread-3
res-3

Servlet Request object address is :: 1604484785
Servlet Response object address is :: 1837946685
Current Servlet object address is :: 2117018993
Current Request-Thread object address is :: 2111171375

Servlet Request object address is :: 516152835
Servlet Response object address is :: 268710827
Current Servlet object address is :: 2117018993
Current Request-Thread object address is :: 339457395

Servlet Request object address is :: 614214443
Servlet Response object address is :: 710656456
Current Servlet object address is :: 2117018993
Current Request-Thread object address is :: 778322011

```
<init-param>
    <param-name>name</param-name>
    <param-value>sachin</param-value>
</init-param>

<init-param>
    <param-name>age</param-name>
    <param-value>49</param-value>
</init-param>

<init-param>
    <param-name>address</param-name>
    <param-value>MI</param-value>
</init-param>
```



1.why there is no main method in Servlets??
Answer: JVM has be designed to make a call to main method with the following prototype to start the execution

```
public static void main(String[] args)
```

If the application is standalone application then jvm will come into picture so we need main method to start the execution.

if the application is webapplication, then container will execute our java code based on its life cycle actions like

a. Loading
b. Instantiation
c. Initalization
d. RequestProcessing
e. De-Instantiation

Since jvm does not play vital role to start the execution we don't need "main method" in Servlets.

2.Eclipse Shortcuts

ctrl+2, L => to genreate the return type of method,constructor,....
Alt+ctrl+arrowup -> To Duplicate the lines.
Alt+arrowup -> To move the line upwards.
Alt+arrowdonw -> To move the line downwards.
Alt+shift+m -> To move the selected lines to be a part of the method.

Alt+shift+R -> To replace the varaible name in every place of java code.
Alt+shift+s,r => To generate setters and getters
Alt+shift+s,S => To generate toString()
Alt+shift+s,v => To do override for any methods of the class.
cntrl+shift+o => To organize the import statements/import particular class
ctrl+a => To select everything
ctrl+shift+I=> To perform indentation.
ctrl+shift+f => To perform formatting.
ctrl+shift+/ => To perform commenting.
ctrl+shift+\ => To perform uncommenting.
ctrl+shfit+t => To open particular class from the jar.
ctrl+ o => To list all the methods of the class.
ctrl+spacebar => To give assistance for our code.
ctrl+d => To delete the line
ctrl+c => To copy
ctr+v => To paste
ctrl+shift+L => To list all the shortcuts of the eclipse
ctrl+2,f => To generate a instance variable
ctrl+shift+x => to make upper case
ctrl+shift+y => to change to lower case

3.ServletConfig InitParameters Using

1.Xml
2.Annotations

a.
```
@WebServlet(urlPatterns = {}, loadOnStartup= 10,
      initParams = {
                  @WebInitParam(name = "url", value = "jdbc:mysql:///octbatch"),
                  @WebInitParam(name = "user", value = "root"),
                  @WebInitParam(name = "password", value = "root123")
            }
```

Note:
	Servlet initialization parameters are key-value pair where both key and value are of type String.
	From the Servlet we can access these parameters but we can't modify.
	since we can't modify we just have only getXXXX() but not setXXXX().
	So we say Servlet Initialization parameters as "Deploy time constants".
	For every Servlet we will have only one ServletConfig object to hold its configuration information.

ServletContext(I)
==========
	For every webapplication web container will create only one ServletContext object to hold the configuration details.
	By using context object we can get configuration information like context paramters, requestdispatcher etc...

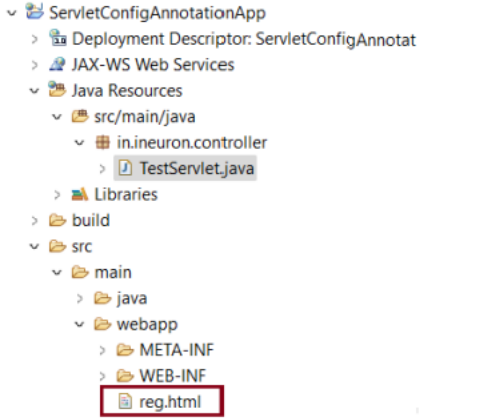
Assume there are 3 servlets and for all the servlets if the configuration details is common can we keep in config object?
	Ans. we can keep, but it is not a good practise because if we keep the data inside Context object it will be available to
		all the servlets of the application.

How to keep the data in ServletContext object?
	Ans. We can keep in only 1 way that is through XML.
		Annotation support not available becoz when container gets started only ServletContext object is created and
		no java code is coming into pitcure to give the information through Annotation.

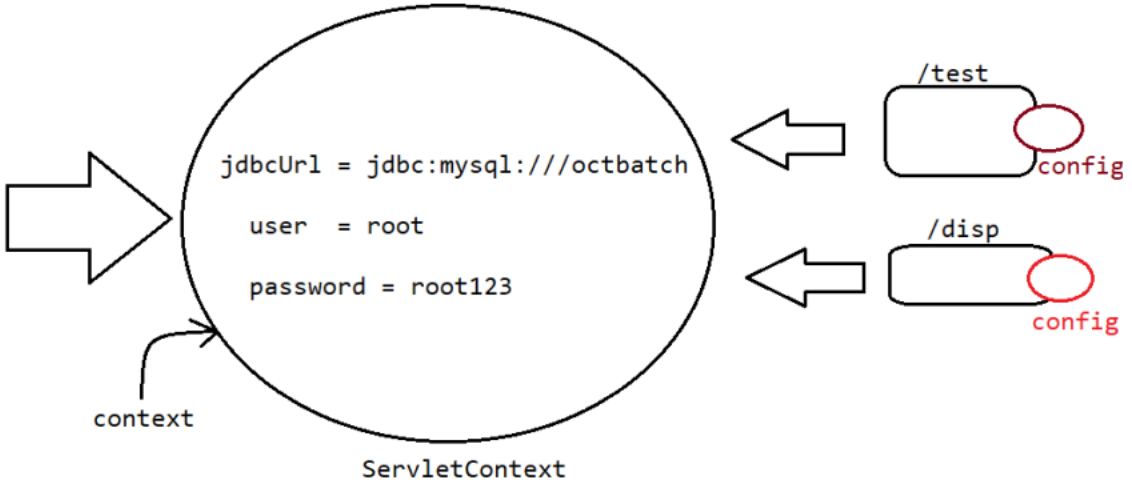
web.xml
=======
```
<web-app>
	<context-param>
		<param-name>jdbcUrl</param-name>
		<param-value>jdbc:mysql:///octbatch</param-value>
	</context-param>
	<context-param>
		<param-name>user</param-name>
		<param-value>root</param-value>
	</context-param>
	<context-param>
		<param-name>password</param-name>
		<param-value>root123</param-value>
	</context-param>
	<servlet>
		;;;
		;;;
	</servlet>
</web-app>
```

Inside servlet we can get the ServletContext data in 2 ways
	a. ServletConfig config = getServletConfig();
	 ServletContext context= config.getServletContext();
		methods
			public String getInitParameter(String name)
			public Enumeration getInitParamterNames()

```
      b. ServletContext context = getServletContext();
                  methods
                        public String getInitParameter(String name)
                        public Enumeration getInitParamterNames()


Note:
  when 2 servlets have different load-on-startup then
      a. lower load-on-startup will get chance first for execution

 when 2 servlets have same load-on-startup then
      a. it depends on container(not in the hands of the programmer)

 if we give negative value for load-on-startup then the container will not load any
of the servlet.
 we should give only positive value (zero can also can be given)

Difference b/w ServletContext vs ServletConfig object
=============================================
ServletContext
      a. For every webapplication, container will create only one ServletContext
object to hold the data at application level.
      b. It will be created at the time of application deployment and destroyed at
the time of application undeployment.
      c. <context-param>
                  <param-name></param-name>
                  <param-value></param-value>
         </context-param>
        d. 2 ways to get the Context Object
             ServletContext context = getServletContext();
             ServletConfig config = getServletConfig();
             ServletContext context = config.getServletContext();
      e. Configuration can be done only in one way through XML

ServletConfig
      a. For every Servlet, container will create only seperate ServletConfig
object to hold the data at servlet level.
      b. It will be created at the time of Servlet object creation and destroyed at
the time of Servlet Object Destruction.
      c. <init-param>
                  <param-name></param-name>
                  <param-value></param-value>
         </init-param>
      d. Approach to get ServletConfig object
                  ServletConfig config = getServletConfig()
      e. Configuration can be done in 2 ways
            a. XML
            b. Annotation
```




```xml
<context-param>
   <param-name>jdbcUrl</param-name>
   <param-value>jdbc:mysql:///octbatch</param-value>
</context-param>

<context-param>
   <param-name>user</param-name>
   <param-value>root</param-value>
</context-param>

<context-param>
   <param-name>password</param-name>
   <param-value>root123</param-value>
</context-param>
```



commonly used packages in servlet coding
javax.servlet.*
javax.servlet.http.*
==========
using these interfaces abstraction is promoted through which WODA is achieved
ServletRequest(I)
ServletResponse(I)
HttpServletRequest(I)
HttpServletResponse(I)
ServletContext(I)
ServletConfig(I)

```
void init(ServletConfig config) throws SE,IOE
{

}
void doXXXX(HttpServletRequest request, HttpServletResponse response)throws SE,IOE
{
      //request => QueryString data in the form of key-value pair
      //response=> PrintWriter object will be there with empty response
}
```

w.r.t tomcat server the implementation classnames are as shown below
==========
Implementation class of config is :: org.apache.catalina.core.StandardWrapperFacade
Implementation class of context is ::
org.apache.catalina.core.ApplicationContextFacade
Implementation class of request is :: org.apache.catalina.connector.RequestFacade
Implementation class of response is :: org.apache.catalina.connector.ResponseFacade

Different types of scope and attributes in Servlet
==========
scope refers to the accessability of a variable.
      a. local scope => restricted inside method
      b. global scope=> avaialble in all the methods of a particular task.

There are 3 types of parameters(k,v) possible in servlet
      a. Form parameters(QueryString[k,v])
      b. ServletInitializationParameters(ServletConfig[k,v])
      c. ContextInitializationParamters(ServletContext[k,v])

The above 3 parameters type are read-only.from the servlet we can perform only read operation, we cannot modify remove
values based on our requirement.so we say parameter type of data is not best suited for sharing the data between
component of the webapplication.
parameter data => both key and value should be String.

To resolve this problem we should go for "attributes" type of data.based on our requirement we can add the data,
remove the data and we can share the data between the components of the application.
attribute data => key should be String, value can be any Object.

Based on our requirement we need to store the attributes in particular scope.
In Servlet we have 3 types of scope
      1. request
      2. session(HttpSessionTracking)
      3. application/context .

1.request
	=> This scope is maintained by ServletRequest /HttpServletRequest object.
	=> This scope will start at the time of request object creation(before calling service())
	=> This scope will destroy at the time of request object destruction(after calling service())
	=> The data stored in the request object will be available for all components which are procesing that request.

3. application/context
	=> This scope is maintained by ServletContext object.
	=> This scope will start at the time of context objet creation(during deployment)
	=> This scope will destroy at the time of context object destruction(during undeployment)
	=> The data stored in the context object will be available for all the components of the application,irrespetive of
		request and the user.

write a program to display hit count(number of requests) of a webapplication?
	firstreq = > hitcount = 1

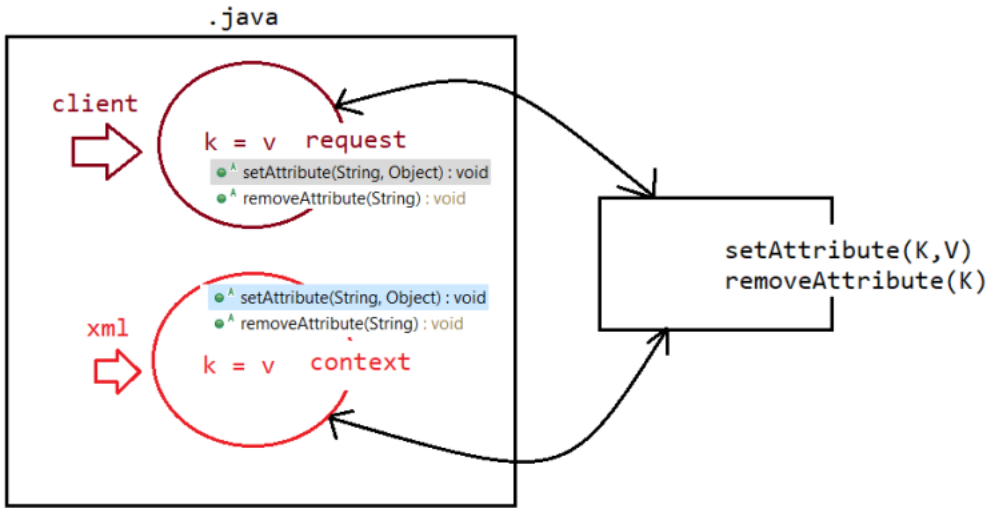
	refer: ScopeApp

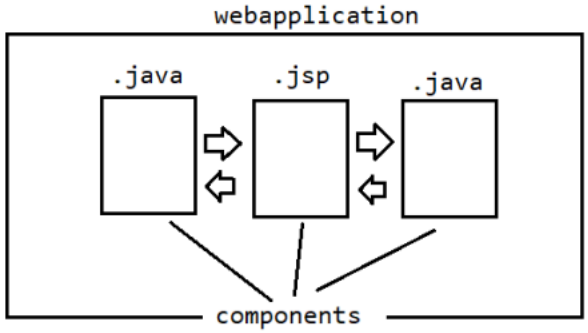
Write a program to display all the attribute information present in application scope?
	Note: In ApplicationScope container will add some attributes for internal purpose.

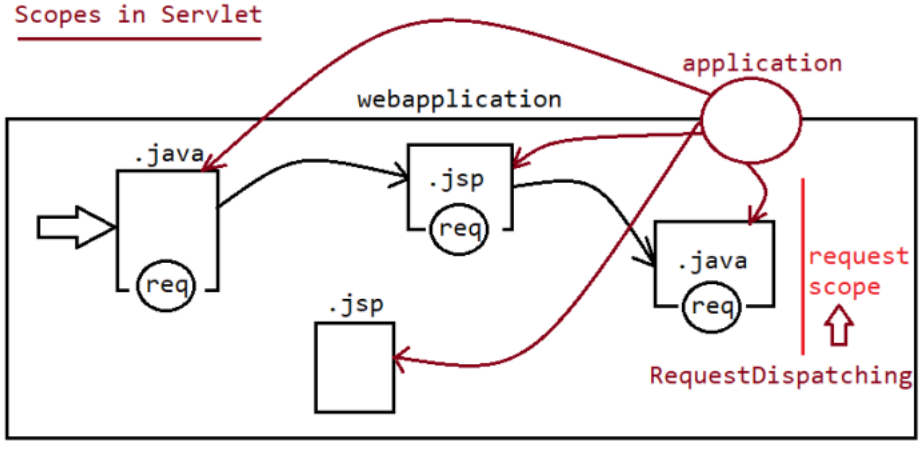
Getting information from the url
==========
	1. getRequestURI()
	2. getQueryString()
	3. getServletPath()
	4. getPathInfo()
	5. getContextPath()

```
package in.ineuron.controller;

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

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


@WebServlet("/test/iNeuron/*")
public class TestServlet extends HttpServlet {
	private static final long serialVersionUID = 1L;

	@Override
	public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
		PrintWriter out = response.getWriter();
```

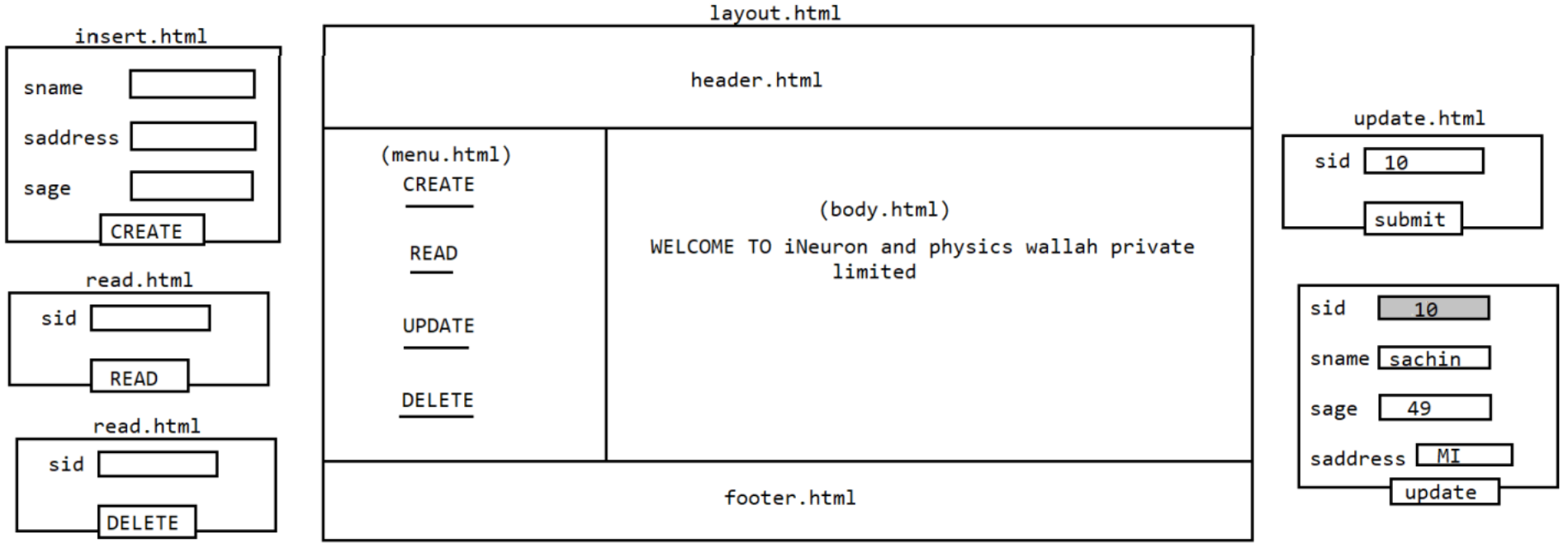
```
            out.println("<h1>Request URI  :: "+request.getRequestURI()+"</h1>");
            out.println("<h1>Context Path ::"+request.getContextPath()+"</h1>");
            out.println("<h1>Servlet Path ::"+request.getServletPath()+"</h1>");
            out.println("<h1>Path Info    ::"+request.getPathInfo()+"</h1>");
            out.println("<h1>Query String ::"+request.getQueryString()+"</h1>");
            out.close();
      }
}

request
      http://localhost:9999/RequestAppInfo/test/iNeuron/hyder/java?
name=sachin&password=tendulkar
response
      Request URI :: /RequestAppInfo/test/iNeuron/hyder/java
      Context Path ::/RequestAppInfo
      Servlet Path ::/test/iNeuron
      Path Info ::/hyder/java
      Query String ::name=sachin&password=tendulkar

Deployment
==========
      Harddeployment
            Creating an webapplication inside webapps folder of tomcat and starting
the server manually is called
            "hard-deployment".

      Smoothdeployment
            Creating an application outside the webapps folder of tomcat and
starting the server through some additional
            set up is called "smooth-deployment".

Note:
webapps(deployment folder)
      In case of eclipse integration with tomcat, internally eclipse uses "smooth"
deployment through which it clones
      our project and perfoms deployment as shown in the following path
                  *\.metadata\.plugins\org.eclipse.wst.server.core\tmp0\wtpwebapps
```


.java
client
k = v request
setAttribute(String, Object) : void
removeAttribute(String) : void
xml
setAttribute(String, Object) : void
removeAttribute(String) : void
k = v context
setAttribute(K,V)
removeAttribute(K)


webapplication
.java
.jsp
.java
components


Scopes in Servlet
application
webapplication
.java
req
.jsp
req
.java
req
.jsp
request
scope
RequestDispatching

Webapplication to perform CRUD operation on student table



Session Tracking mechanism
==========
As a part of webapplication,it is essential to manage the client previous request data at the time of processing later request.

Solution-1
	usage of request object
		request object gets created, when we send the request to the application and it gets destroyed at the time of
		sending the response, so we can't keep track of the client previous request data using "request object".

Solution-2
	usage of context object
		If we use context object, then the data will be shared to all components of the application and to all the users of
		the application where we can't differentiate the data between the user, so it is not a good approach.

In webapplication to manage the client previous request data at the time of processing later request , we need to have a
clear cut seperation b/w the clients data.
	To manage this, we need to go for Server side mechanism called "SessionTracking".

Session
======
	It refers to the amount of time the client spend with the server.
	State of the Session
		The data which is transferred b/w client and server through multiple no of requests during a particular session then
		that data is called "State of the Session".
	In webapplication as soon as we start the server ,ServletContext object is created, when we send the request to the
	application HttpServletRequest object is created, similarly w.r.t we need to write a code to create "HttpSession" object.
	If multiple users uses the application, then we need to write a code to create seperate "HttpSession" object and we need
	to manage the object, to manage these session objects we need to learn "SessionTracking mechanism".

There are 4 SessionTracking mechanism
==========
1. HttpSessionTracking mechanism.
2. Cookie Session Tracking mechanism.
3. URL-ReWriting Session Tracking mechanism.
4. Hidden Form Field Session Tracking mechanism.(developer specification)
As per SUN specification for ServletApi, we have only 3 mechanism for SessionTracking.

HttpSession Tracking mechanism
==========
   What is the difference b/w getSession() and getSession(false)?
Answer: Both the method will return HttpSession object only.
	getSession() => container will check whether session object existed w.r.t the particular user or not.
			if HttpSession object existed then it would return the same object, otherwise
			it will create a new HttpSession object w.r.t the

user and that object will be returned.

getSession(false) => container will check whether session object existed w.r.t the particular user or not.
if HttpSession object existed then it would return the same object, otherwise then container
will return null.

note: getSession() and getSession(true) both are same.

Q> If we allow mulitple users to access webapplication then to container we need to instruct the creation of HttpSession
objects,In this case how container will identify user specific HttpSession object in order to put the user specific attributes
and get the attributes?
Answer: HttpSession objects are created manually, and for these objects container will maintain unique ID in the form of
HexaDecimal value called SessionID.
Container will prepare Session id value in the form of key called "JSessionID".
Container will create a cookie and attach the session id to send it as a response to the browser every time when
the interaction happens b/w client and server.
when we use request.getSession(), container will get the sessionid and checks whether that user specific object is
available or not, if avaiable it will identify that object to process the data.

Note: we can destroy the HttpSession object manually using the method called
public void invalidate()

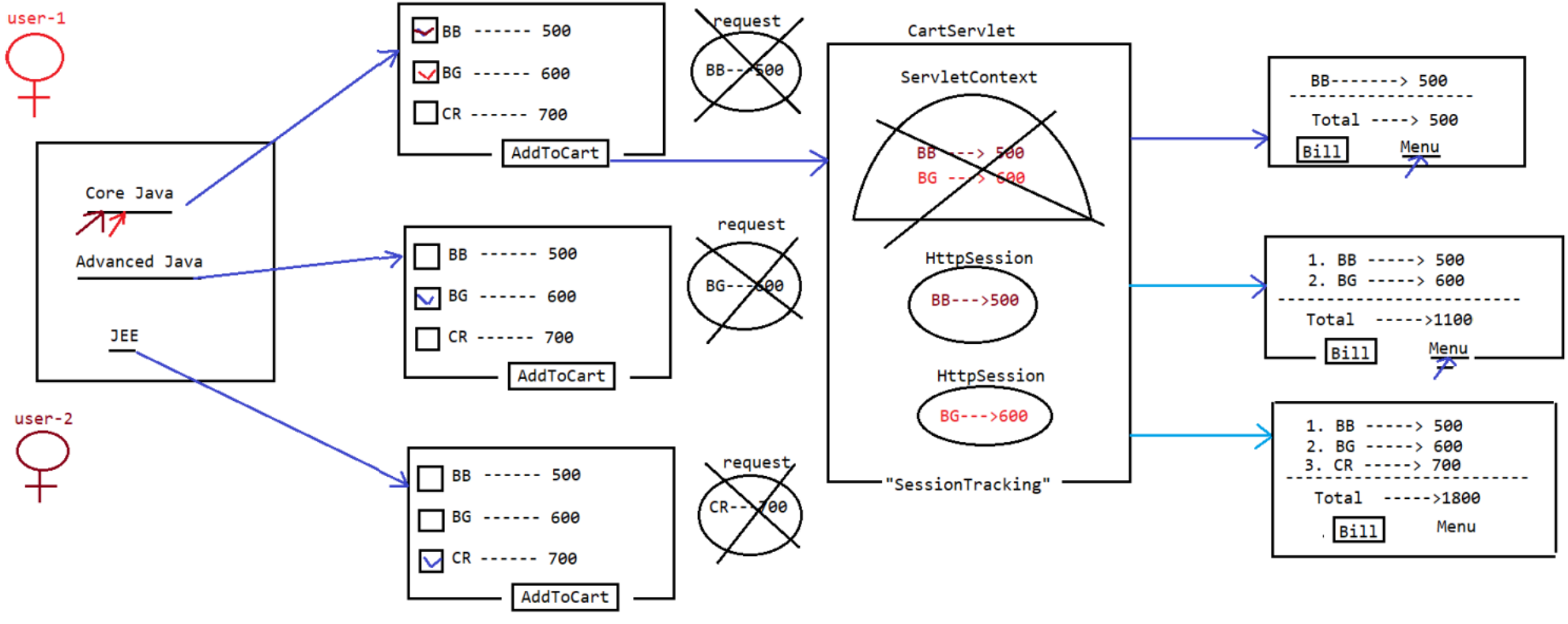
Limitations of HttpSession Tracking mechanism
==========
1. Creating HttpSession object w.r.t client at the server side is too costly which would decrease the performance of
the application(managing the session object at the server side).
2. HttpSession object is exchanged b/w the client and server through cookie file,if client disables cookie at the client side
then this mechanism won't work effectively.
To resolve this issue, we have a new mechanism called "Cookie Session Tracking".
refer: HttpSessionTrackingApp

Cookie Session Tracking
==========
In case of Cookie Session Tracking mechanism, we need to create a cookie for every request data w.r.t every user.
After creating a cookie, we need to send this cookie along with response object.
These cookies will be exchanged b/w client and server in the form "request-response" object.
To Create a cookie we use
public Cookie(String key,String value)
To retreive the cookies we use request object
public Cookies[] getCookies()
To add the cookies to response object we use
public void addCookie(Cookie c)
refer: CookieSessionTrackingApp

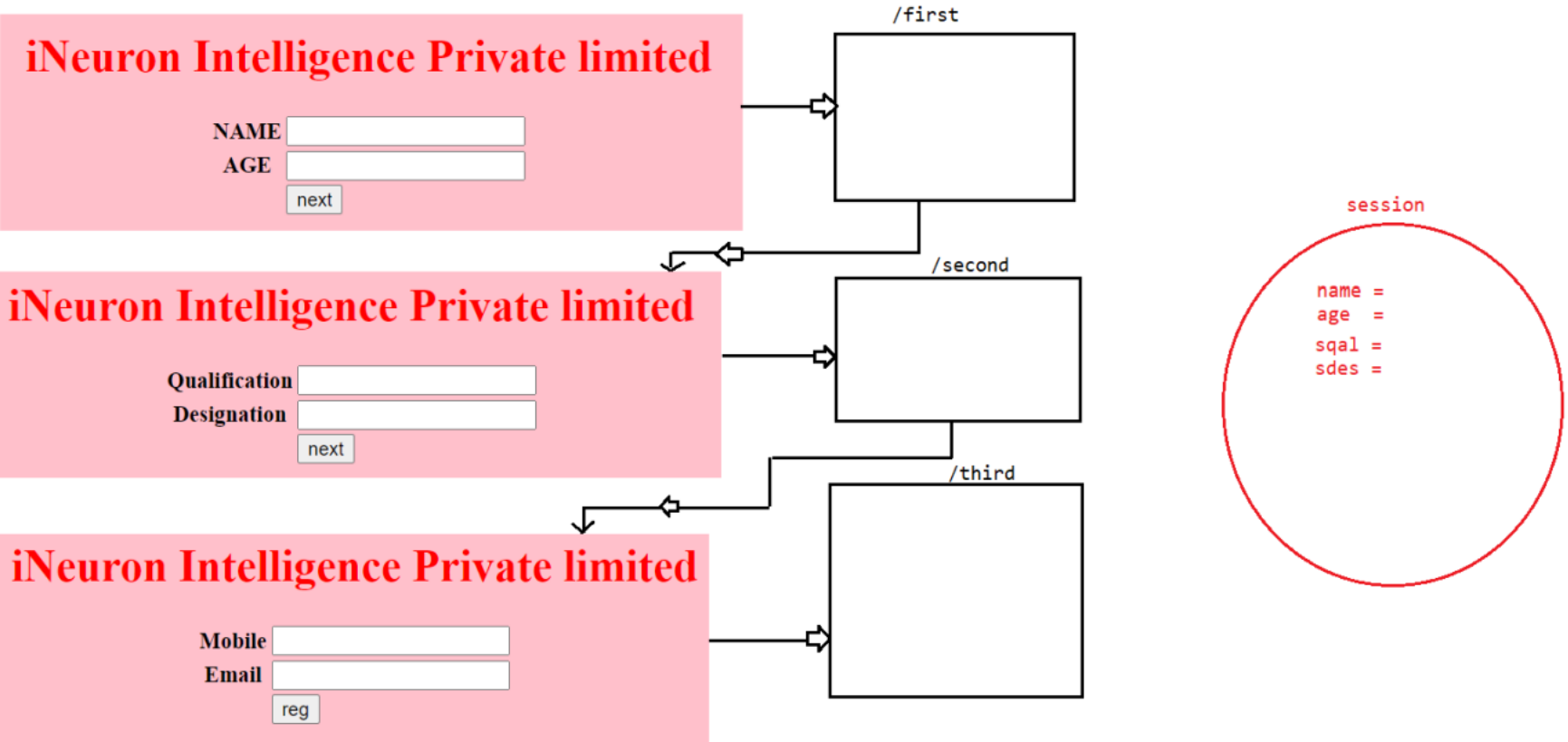
Drawback

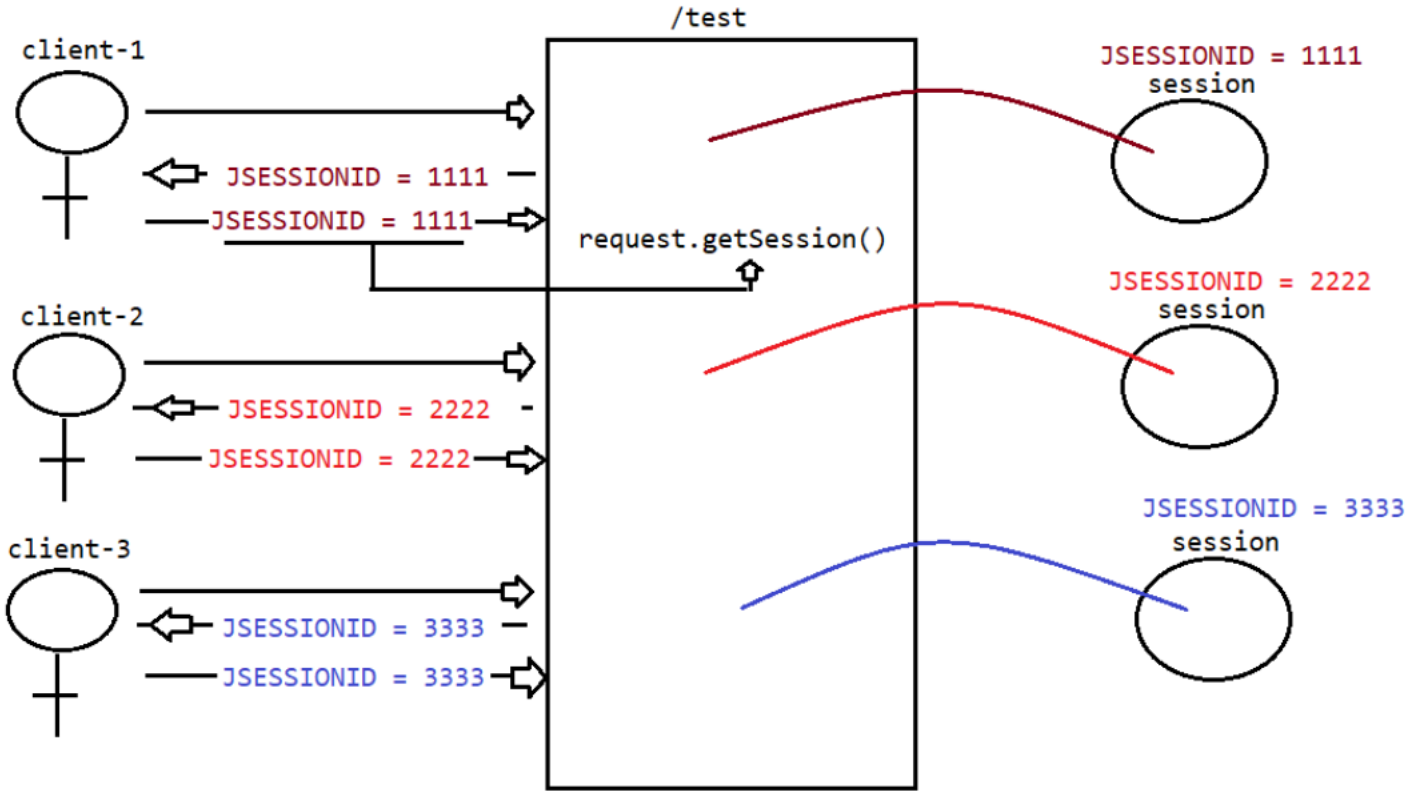
========
1. In case of Cookie Session tracking,cookies are maintained at the client side. if browser disables the cookie then this
    mechanism wont' work effectively.
2. Since cookies are stored in the client machine(browser), there is every possibility that client can misuse the data sent
     from the server.(security breach can happen)

      To overcome this limiation we need to use "URLReWriting Mechanism".


user-1
Core Java
Advanced Java
JEE
user-2
BB ------ 500
BG ------ 600
CR ------ 700
AddToCart
request
BB---500
BG---600
CR---700
CartServlet
ServletContext
BB ---> 500
BG ---> 600
HttpSession
BB--->500
HttpSession
BG--->600
"SessionTracking"
BB-------> 500
Total ----> 500
Bill
Menu
1. BB -----> 500
2. BG -----> 600
Total -----> 1100
Bill
Menu
1. BB -----> 500
2. BG -----> 600
3. CR -----> 700
Total -----> 1800
Bill
Menu

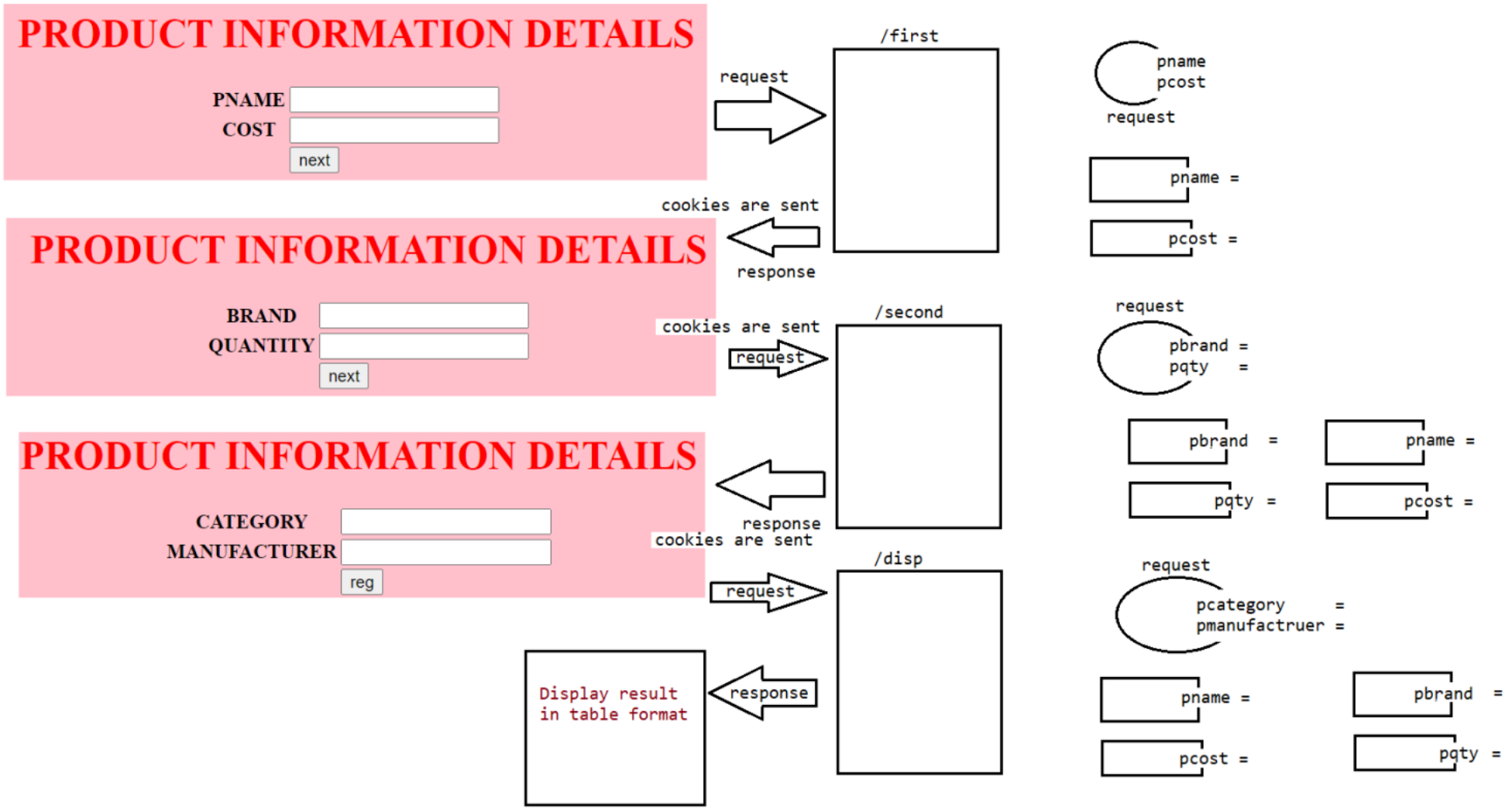
SessionTracking using HttpSession Object





To destory the session object manually
public void invalidate()

Cookie Tracking Mechanism



Servlet-api
=========
1. HttpSessionTracking
2. CookieSessionTracking
3. URLReWriting Tracking

HttpSessionTracking
================
1. Session id will stored in the form of cookie and in client side if browser disables the cookie HttpSessionTracking mechanism
     won't work.
2. Maintainence of Session object is at the server side which would be burden to the server.
            eg: jSESSIONID = ...DBEFH

CookieSessionTracking
===================
1. Cookies will be stored in the client side and if browser disables the cookies then we can't keep track of client data.
2. Since cookies are used to store the client data,it might result in "Security Breach".

URLRe-Writing Mechanism
======================
=> This mechanism is same as "HttpSessionTracking" mechanism,but sessionid won't be stored inside cookie rather the
      sessionId will be appended to the url everytime when the request-response happens b/w client and server.
                  eg: <form method ="get" action='"+response.encodeURL('./second') +"'>
                        </form>
                    |

                  ./second?JSESSIONID = ...

Note: In Realtime project we don't use technologies directly,we use framework to improve the productivity,so by default
          Framework support SessionTracking Mechanism through "URL-ReWriting" only.

Invented by Developer
===================
1. HiddenFormField
            Not used in realtime as it increases the lines of code.

URLPatternTypes
==============
 As per Servlet specification, we have 4 different ways of mapping the URLPattern
      a. Exact match URL Pattern => eg: /test
      b. Longest Path URL Pattern => eg: /controller/servlet/*
      c. URL Pattern by extension => eg: *.do
      d. Default URL Pattern => eg: /

Which of the following are valid url patterns?
 1. /test (valid)
 2. /test/*/test(invalid)
 3. /test/test/*(valid)
 4. *.test(valid)
 5. /(valid)

6. /test/test/*.do (valid)

Patterns for Servlet
==========
/ => FourthServlet
/test => FirstServlet
/test/test/* => SecondServlet
*.do => ThirdServlet

http://localhost:9999/URLPatternTypesApp/test/test/navindReddy.do => SS
http://localhost:9999/URLPatternTypesApp/test/test/navinReddy => SS
http://localhost:9999/URLPatternTypesApp/
=> DS
http://localhost:9999/URLPatternTypesApp/navinReddy.do => TS
http://localhost:9999/URLPatternTypesApp/test/hyder =>
DS

Note: When none of the Servlet is getting mapped, then default Servlet will get a chance once again.
Webcontainer will always gives preference in the following order
a. Exact match UrlPattern
b. Longest Path prefix UrlPattern
c. UrlPattern by Extension
d. Default UrlPattern

In realtime projects which type of url pattern is preferred?
URLPattern by extension is prefered(*.do).
In SpringMVC for inbuilt servlet called "DispatcherServlet(FC)", we configure it through "URL-pattern by extension".

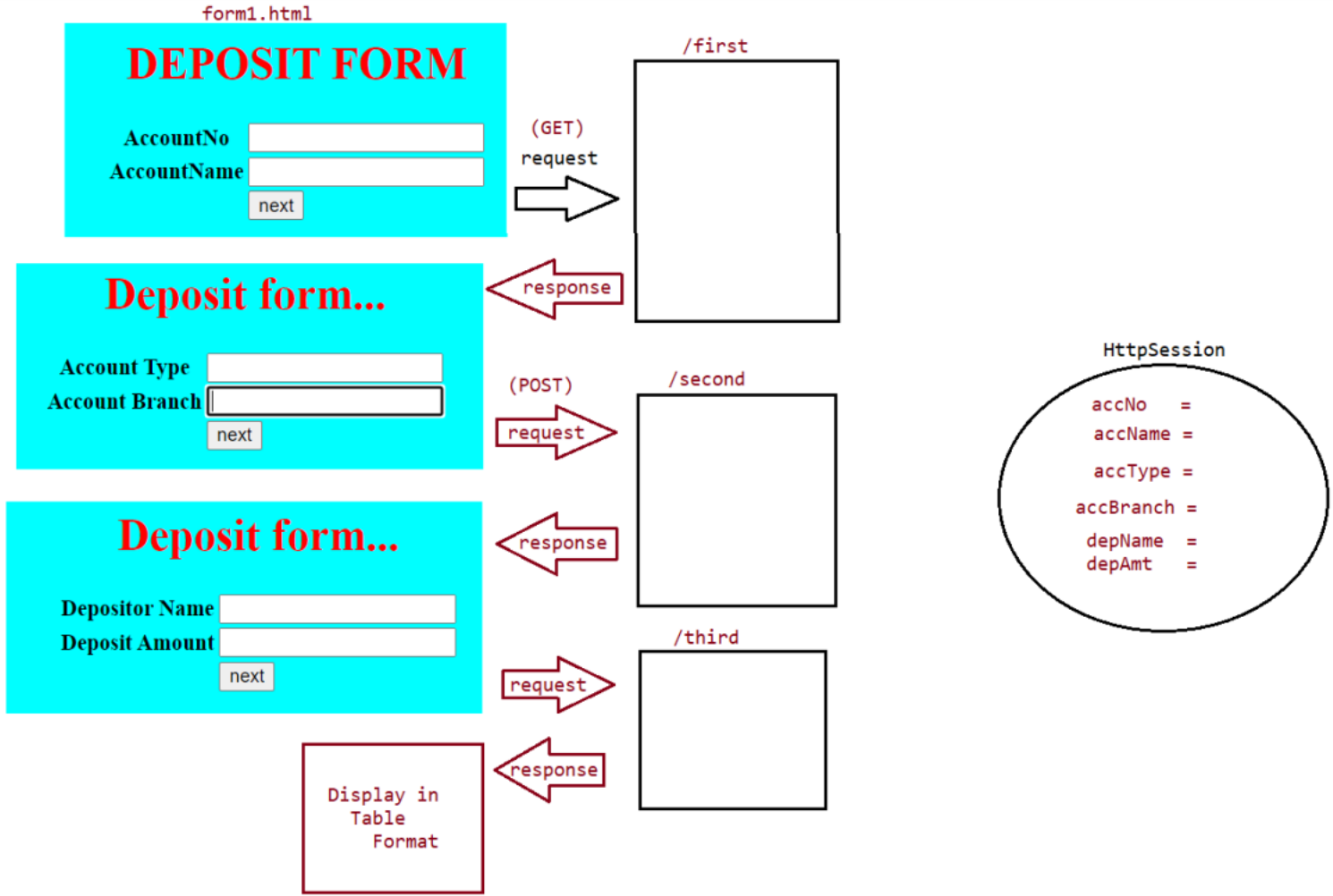
Configuring welcome pages
==========
It is highly recomended to configure the welcome-page/landingpage for our webapplication.
Advantage
1. It increses the easyness of the use of the webapplication for the end user.

Configuration can be done in XML only for html files, jsp files
==========
```
<web-app>
      <welcome-file-list>
            <welcome-file>home.jsp</welcome-file>
            <welcome-file>welcome.jsp</welcome-file>
            <welcome-file>index.jsp</welcome-file>
      </welcome-file-list>
</web-app>
```

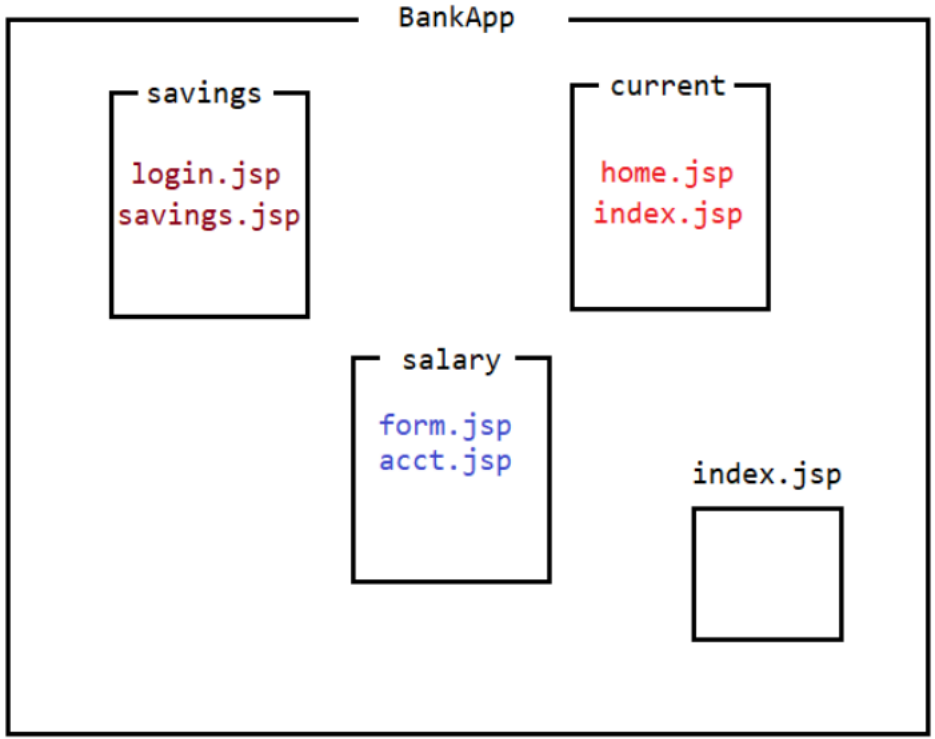
Note:
1. The order of searching a landingpage would be from top to bottom
2. In any webapplication index.html acts as a default welcome file,if index.html is not avaialable then index.jsp acts a default
welcome file.
eg:
http://localhost:9999/FirstApp/ ==========> search for index.html/index.jsp and load as the response to the client.

As per JEE specification, when we are configuring jsp pages inside <welcome-file>, we need to just specify the file name not

```
with "/".
            eg: <welcome-file>index.jsp</welcome-file>(valid)
                  <welcome-file>/index.jsp</welcome-file>(error)
```


form1.html
DEPOSIT FORM
AccountNo
AccountName
next
(GET)
request
/first
response
Deposit form...
Account Type
Account Branch
next
(POST)
request
/second
response
Deposit form...
Depositor Name
Deposit Amount
next
request
/third
response
Display in Table Format
HttpSession
accNo =
accName =
accType =
accBranch =
depName =
depAmt =

Configuring welcome pages in webapplication




```
<welcome-file-list>
   <welcome-file>home.jsp</welcome-file>
   <welcome-file>login.jsp</welcome-file>
   <welcome-file>index.jsp</welcome-file>
</welcome-file-list>
```

eg#1.
http://lh:9999/BankApp =====> index.jsp

eg#2.
http://lh:9999/BankApp/savings ====> login.jsp

eg#3.
http://lh:9999/BankApp/current ====> home.jsp

eg#4.
http://lh:9999/BankApp/salary ====> 404 status code

eg#5
http://lh:9999/BankApp/salary/home.jsp ===> 404status code

Public Area and Private Area of WebApplication

public area of the webapplication

public area of the application

(From the browser client can directly access the resource)

From the browser we can't access, container will access through urlpatterns

private area of the application

a. web.xml
b. annotation

Servlet
|
URLPattern

a. Exact match
b. Long path
c. Extension
d. Default

Filters/Interceptors
==========
   It can be used for PreProcessing and PostProcessing of the request before they reach the target resource in the
   web application.

Areas of Application
==========
      Logging, Security,Altering Request information, Compressing response,encryption of response,authenication etc...

Filters Concpets introduced in Servlet 2.3V

FilterApi contains 3 interfaces
1. Filter(I)
2. FilterConfig(I)
3. FilterChain(I)

To create a filter we need to implement Filter(I)
==========
```
public interface Filter {
  public void init(FilterConfig config) throws ServletException;
  public abstract void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain) throws IOException,
                                                                    ServletException;
  public void destroy();
}
```
refer: FilterApp

Behind the Scenes
==========
1. Whenever we are sending the request to TargetServlet, web container will check if any filter is configured for this servlet
     or not.
2. If any Filter is configured, web container forwards the request to Filter instead of Servlet.
3. After completing the Filter logic, Filter forwards the request to TargetServlet.
4. After processing the TargertServlet, the response will be forwarded to Filter instead of browser.
5. After executing the Filtering logic, filter forwards the total response to the browser.

eg: http://localhost:9999/Filterapp/test

output
```
This line is added by DemoFilter before processing the request...
This is Target Servlet...
This line is added by DemoFilter after processing the request...
```

FilterMapping to particular url-pattern
==========
```
<filter-mapping>
    <filter-name>DemoFilter</filter-name>
    <url-pattern>/test</url-pattern>
</filter-mapping>
```

FilterMapping to Total WebApplication
==========
```
<filter-mapping>
 <filter-mapping>
    <filter-name>DemoFilter</filter-name>
```

```
    <url-pattern>*</url-pattern>
</filter-mapping>
```

FilterChaining
==========
Dividing the Request Pre-Processing tasks into mulitple filters

Webcontainers rule for ordering the filter in FilterChain
==========
It is depended on the container can't be predicted by the user.

What is the difference b/w Filter doFilter() and FilterChain doFilter()?
Filter===> doFilter()
doFilter(ServletRequest, ServletResponse, FilterChain)throws SE,IOE
Total Filtering logic(pre+post) processing logic.
It is a call back method as it is called by the container automatically.

FilterChain===>doFilter()
doFilter(ServletRequest,ServletResponse) throws IOException, ServletException;
Forwarding the request to another filter/servlet.
It is not a callback method,because we have to call explicitly.

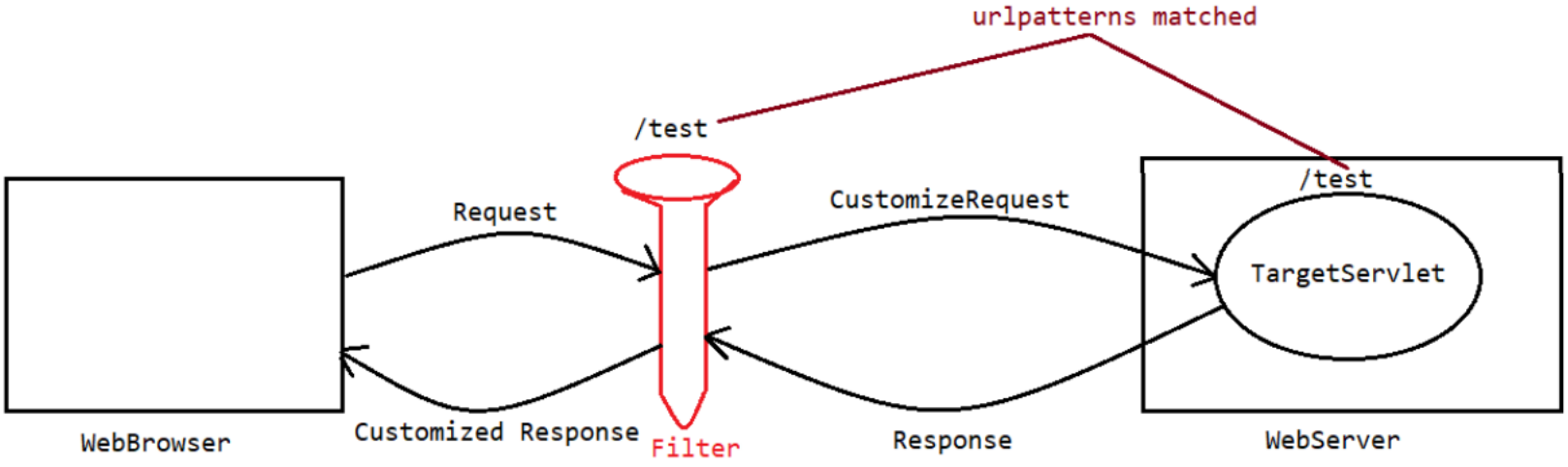
Wrappers and Listeners
==========
Sometimes we need to alter the request and response object inside filters, to do this we need to use "Wrappers" class.

eg1:
within the filter, we have to convert resume information from wordformat to pdfformat.(upload)

eg2:
within the filer, we have to compress the response and that compressed response we can send to browser so that
download time can be reduced.

There are 2 types of Wrappers
a. Request Wrappers
b. Response Wrappers

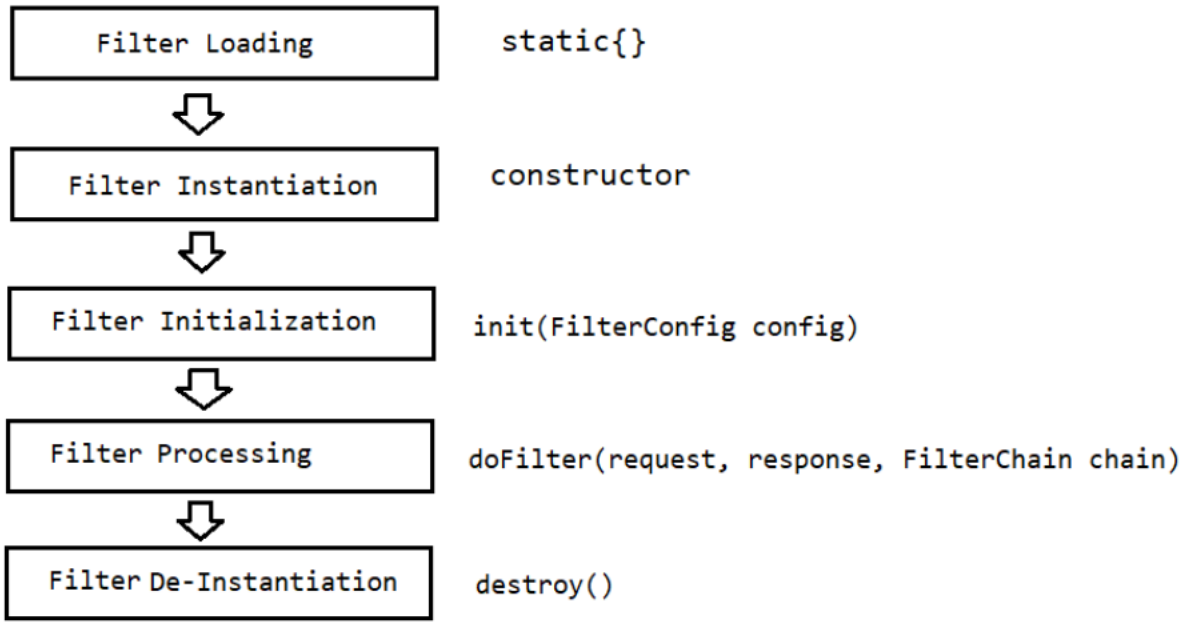
note:
1. No configuration is required for wrappers.
2. Annotation used to represent filter is :@WebFilter(urlPatterns={})



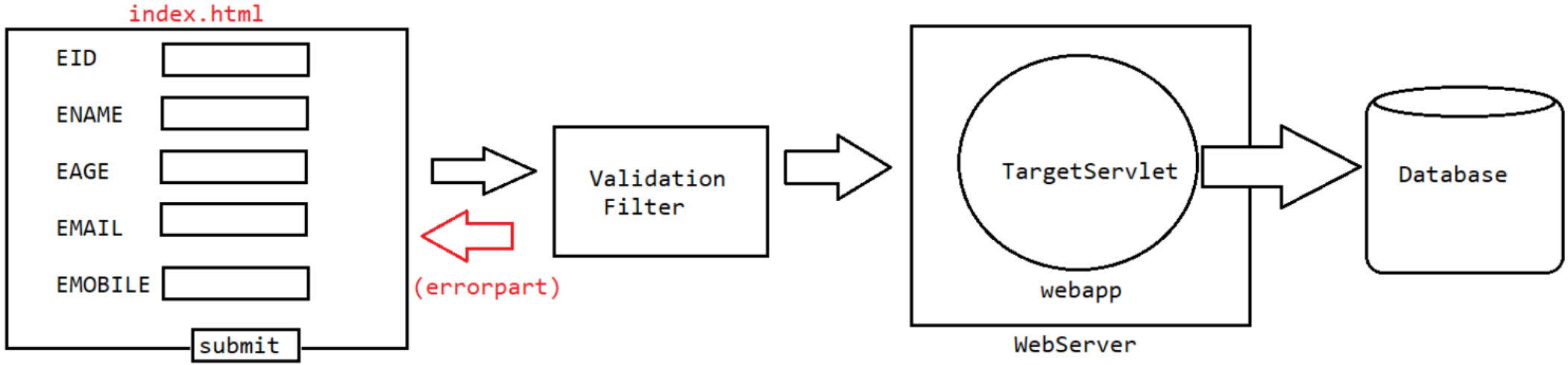
PreProcessing Activity

Authentication
Security
Encoding and Decoding

Filter Life Cycle Actions



For filters we don't need to give <load-on-startup> value, It is autoloaded, instantiated and initalized...




```
eid is empty     => EmployeeId is required
ename is empty   => EmployeeName is required
eage  is empty   => EmployeeAge is required
email is empty   => EmployeeEmail is requried
mobile is empty  => EmployeeMobile is required
```

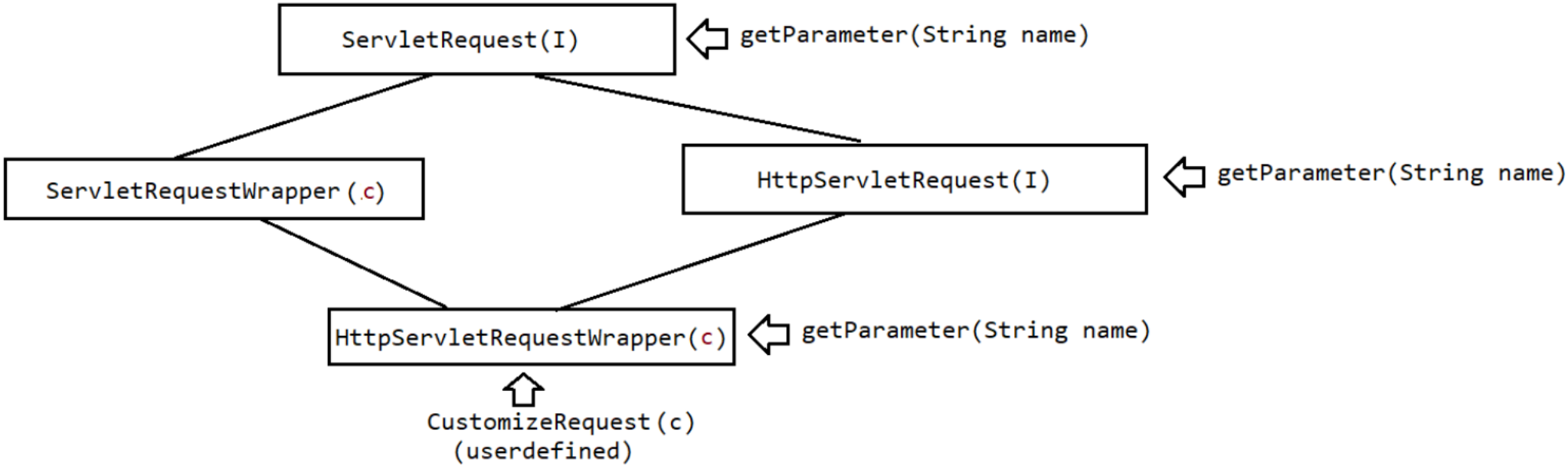
First level checking

```
id     => starts with iNeuron-
eage   => with in 20 to 30
email  => end with @ineuron.ai
mobile => 91-XXXXXXXX
```

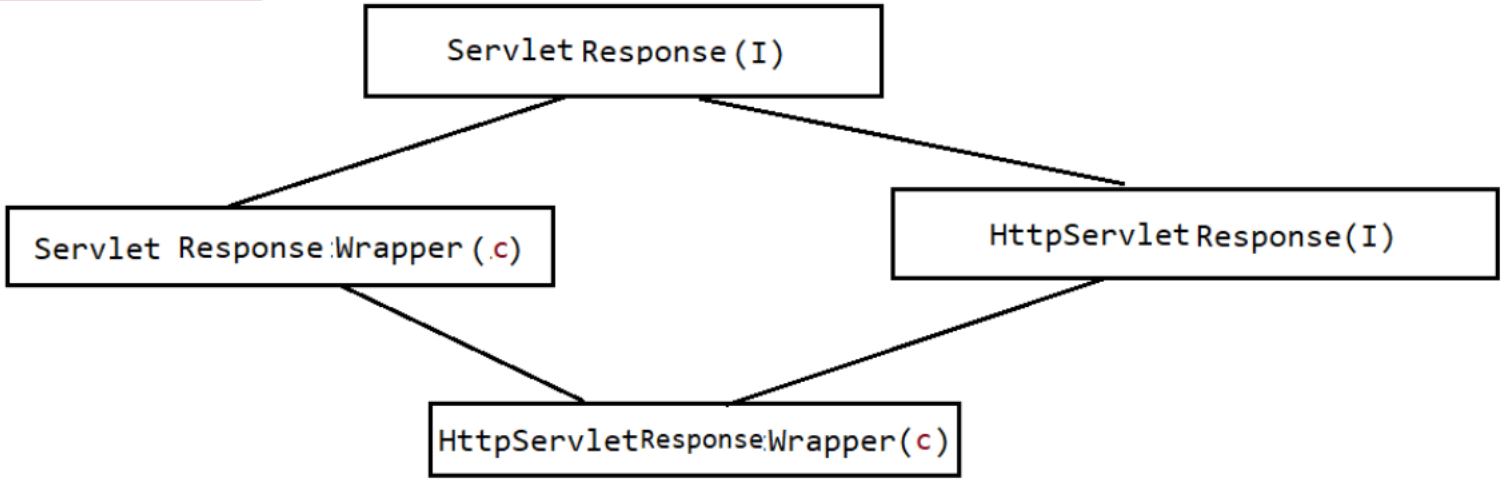
Second level checking

In which order filter gets executed in FilterChaining?

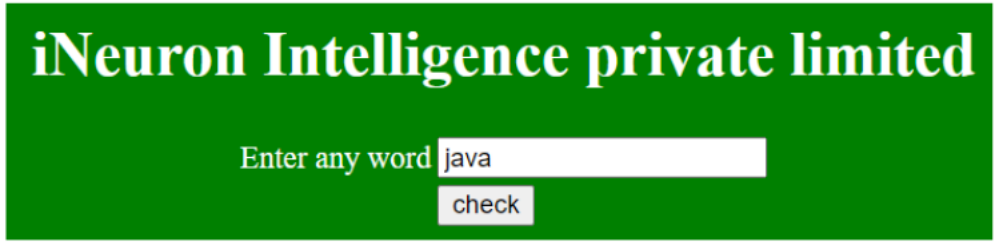
RequestWrapper

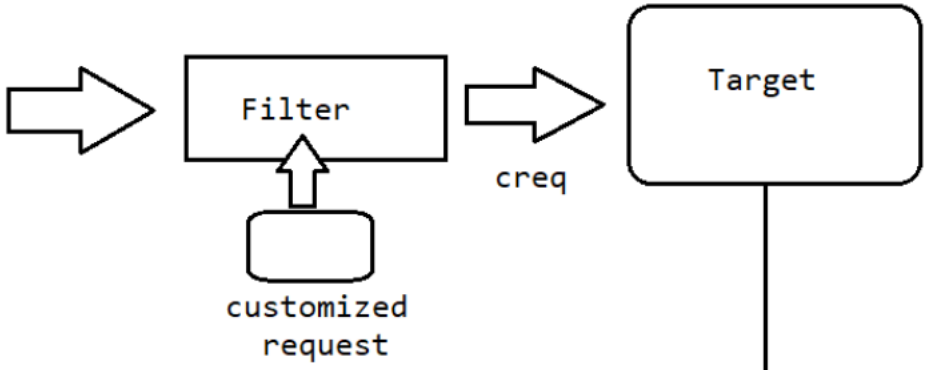


ResponseWrapper



HttpServletRequestWrapper

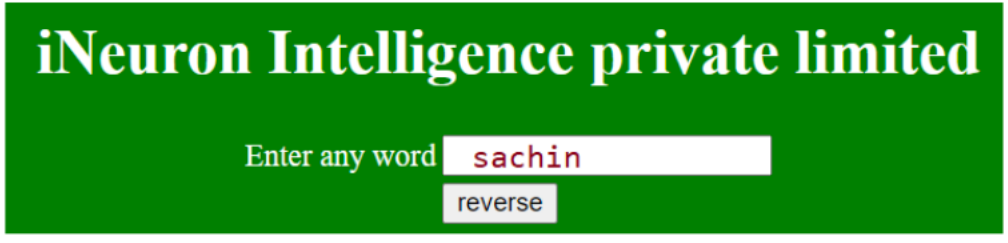




The enterd word is :: SLEEP

response

HttpServletResponseWrapper



output

The reverse word is nihcas