JE®

JavaEra.com

CoreJava | JSP | Servlets | JDBC | Struts | Spring | Hibernate
Projects | FAQs | Sample Programs | eBooks | Certification Stuff
Question Banks | Communities | Tutorials | Softwares | Sample Resumes
Interview Tips | Forums | Discussions | Online Test Engines | Jobs

www.JavaEra.com

A Perfect Place For All Java Resources

www.JavaEra.com

Nataraz (Sathya)

Servlets Notes

JavaEra.com

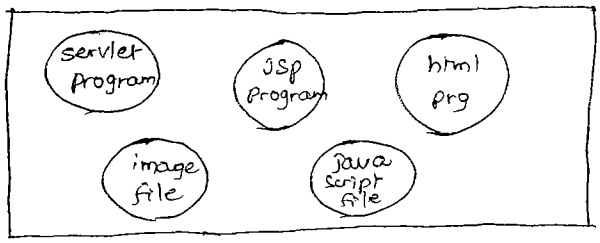
Servlets

* The logics and data of stand alone, desktop applications ~~is~~ are specific to that computer where these applications are running.
* The logics and data of client-server applications, two-tier applications like jdbc applications are specific to one network where they are running. In the above said two tier, client-server applications the server allows only known clients (recognized clients).
* To provide global visibility and accessability to the logics and data of the application use the internet environment based websites or web applications. These applications allow both known and unknown clients having 24x7 access to the resources and logic data.
* when website is under development / testing in a software company then it is called as web application. once web application is hosted on to the internet network by purchasing domain name (www.citibank.com) or host name and space in the internet network is called as website.
* A web application is a collection of web resource programs which can generate web pages. There are two types of web resource programs based on the web page they generate.

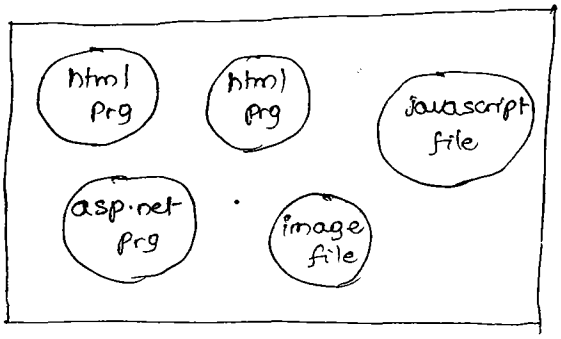
  1) static web resource programs → generates static webpages (Ex: HTML)
  2) dynamic web resource programs → generates dynamic webpages
  Ex: (servlet program, JSP programs, ASP programs, PHP programs)
  ↓
  Personal Home page for Hypertext processing

* static web page contains fixed content forever. The content of dynamic web page changes based on the input values of the request or time of the request generation.

Java based web application



.net Based web Application



* The web resource programs like image file, java script file can not generate web pages directly but they support other web resource programs to generate web pages.
* Stand alone applications, desktop applications and client-server applications will be executed by programmer manually. But the web resource programs of web application must be executed dynamically when they are requested by clients (Browser windows) so we can't think about manual execution for web resource programs. we can use a special software called web server software to execute the web resource programs dynamically when they are requested by clients.
* A web server software can also manage, execute multiple web applications simulteneously or parallelly.

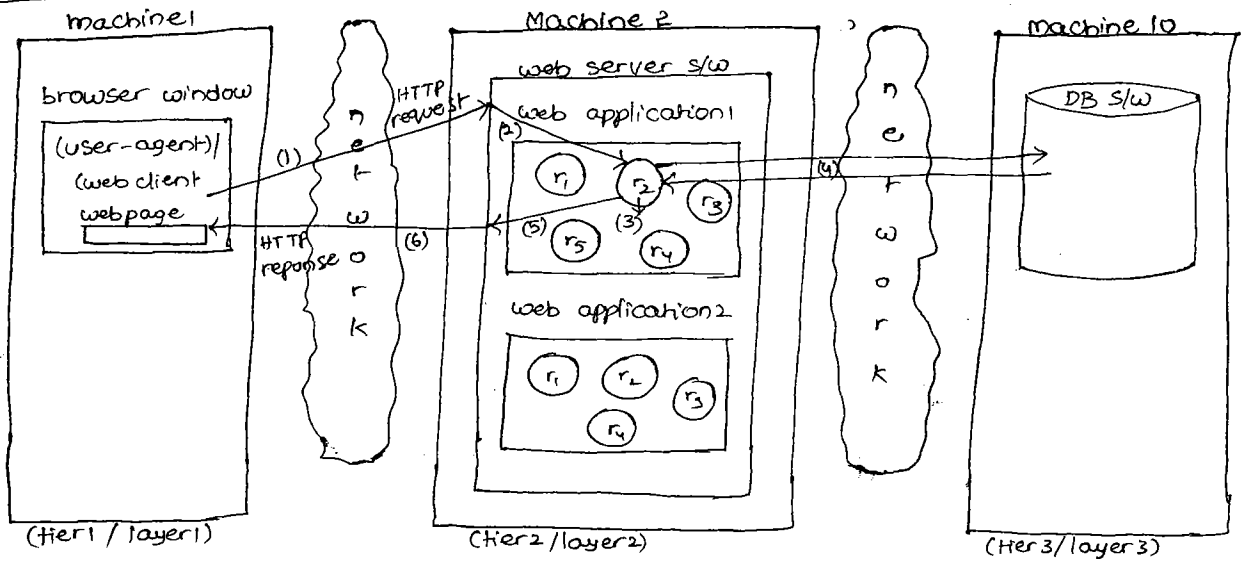
Examples of web server / Http server Softwares

Tomcat
Resin
JWS (Java web server)
PWS (personal web server)
IIS (Internet Information server)
and etc....

* To develop and execute jdbc application we need the following setup.
jdk s/w, Jdbc driver, DB s/w.
* Similarly to develop and execute web application we need the following setup.
browser s/w, web server s/w, db s/w, web technologies → To develop web resource programs

* web server s/w used to manage and execute web application.
* browser s/w used to request to webapplication.
* java based web servers internally uses jdk s/w.

<u>setup required to develope and execute web application</u>



* Each browser window acts as one client to web application.
* A web application can be developed as two tier application without database software or three tier application with database software.
* with respect to diagram

(1) Browser window generates http request to web application.

(2) The web server traps and takes the request and passes the request to appropriate web resource program.

(3) The web resource program process the request.

(4) web resource program interacts with data base software if necessary.

(5) The web resource program generated result goes to web server.

(6) web server sends these results to browser window as HTTP response in the form of web page.

* In the above diagram $r_1, r_2 \ldots$ are web resource programs.
* Based on the place of executing where web resource programs execute there are two types of web resource programs.

(1) server side web resource programs (they execute in the web server).

Ex: servlet prg, JSP prg and etc.

2) client side web resource program → They come to browser ~~to~~ window from the web application placed in web server for execution.

Ex: HTML program, Javascript program, Ajax program and etc...

* Decide whether web resource program is client side or server side based on the place where it executes not based on the place where it resides.
* A web application looks like thin client-fat server application.
* The process of keeping web application in web server is technically called as deployment and reverse process is called as undeployement.

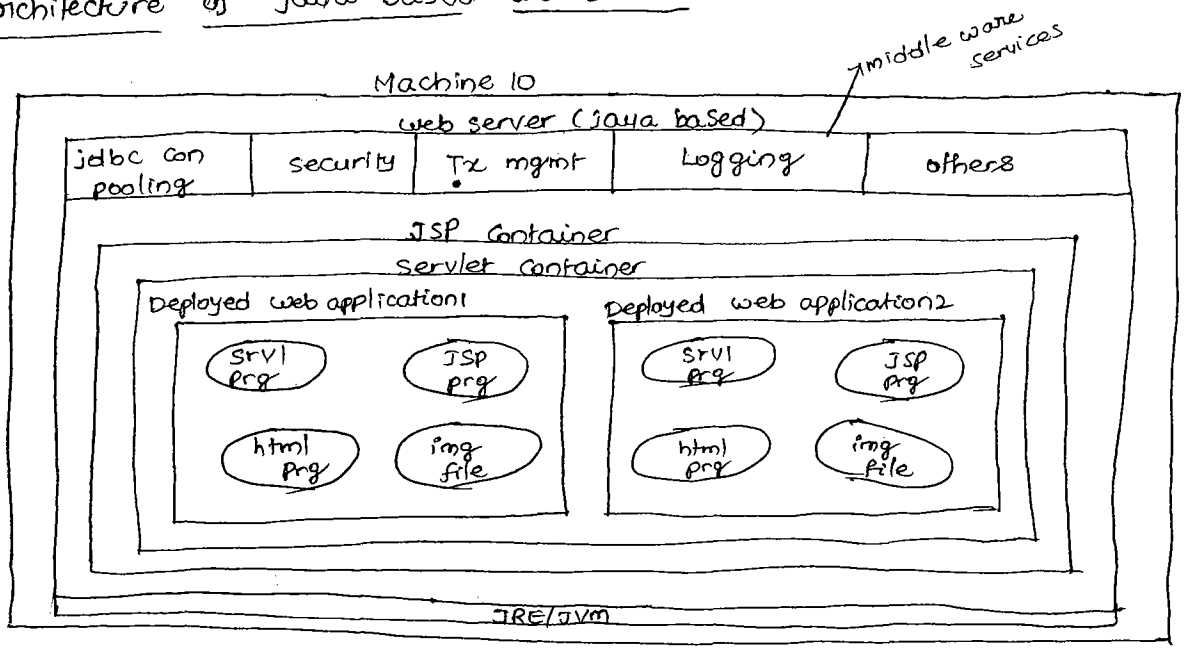
Responsibilities of web server

1) Listens to client request continuously (HTTP request)
2) Traps and takes client generated HTTP request.
3) passes the HTTP request to an appropriate web resource program of web application. (deployed web application)
4) provides container software to execute server side programs (web resource programs)
5) Gathers output generated by web resource programs.
6) passes output of web resource programs to browser window as http response in the form of web pages.
7) provides environment to deploy manage and to ~~undepoly~~ undeploy the web application.
8) Gives middle ware services and etc...

* A container is a software or software application that can manage the whole life cycle (object birth to death) of given resource. (like java class).
* A file or program is called as resource of the application.
* servlet container takes care of servlet program life cycle.
* JSP container takes care of JSP program lifecycle.
* Applet container (Applet viewer) takes care of applet program lifecycle.
* servlet container, JSP containers are part of web servers (java based).
* A container is like an acquarium who can take care of the whole life cycle of given resources called fishes.

* programmer is not responsible to develop containers and webservers but he is responsible to use them to execute web applications.

Architecture of java based web server



* once web server is started one daemon process will be started to listen to client's continuously and to trap and take the client generated HTTP request.
* The process that runs continuously is called as daemon process.
* Every java application contains two default threads.
  (1) main thread
  (2) Garbage collector thread (daemon thread).
* JSP container is the enhancement of servlet container and both containers use JRE/JVM supplied by web server.
* middle ware services are configurable additional services on the applications to make applications executing perfectly in all the situations.
* security middle ware service protects the application from unauthorised and unauthenticated users.
* Transaction management service executes the logics by applying do everything or nothing principle.
* Jdbc connection pool supplies set of readly available jdbc connection objects.

* Logging service keeps track of the application execution process through ~~confirm~~ conformation statements / messages / log messages.
* Middle ware services are not minimum logics of application development. They are additional or optional services to apply on applications.
* A web application can be there with or without middle ware services.
* Web server automatically activates servlet Container and JSP Container to execute servlet, JSP Programs when they are requested by clients.
* The client side web resource programs of web application goes to browser window for execution whereas the server side programs will be executed by using the containers of web server.
* In web application executing environment browser software is called as client software and ~~ser~~ web server software is called as server software.

Examples of Java based web servers

Tomcat ⟶ from Apache Foundation
Resin ⟶ from Resin soft
JWS ⟶ from Sun microsystem

* HTTP is a application level protocol define in set of rules to get communication between browser window and web server. The text that allows sequential reading is called as normal text.
  Ex: notepad text.
* The text that allows non-sequential reading through hyper links is called as hyper text.
  Ex: any web page content.
* Protocol HTTP is useful to transfer hyper text between browser window and web server, web server and browser window.

Different browser softwares

IE — MicroSoft
Net-scape Navigator — from Netscape
Fire Fox — from mozilla
~~Hot~~ Java — from RedHat

chrome —— from Google
opera —— from Opera soft

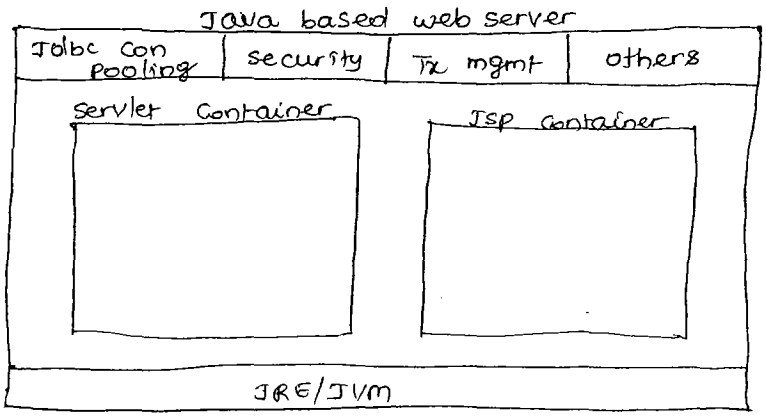
Different client side Technologies (for developing client side web resource programs)

html ⟶ from W3C
java script ⟶ from netscape
VB Script ⟶ from MicroSoft
AJAX ⟶ from Adaptive path

NOTE: Java script is no way related with the programming language java.

* The code that can not be executed independently and which must be embedded in other technology based program for execution is called as script code.
* Java script code can not be executed independently and it must be executed by embedding with html code so java script is called as scripting language.

Another diagram that shows Java based web server architecture



Jsp container is developed based servlet container

* when java web application is deployed in java web server then the servlet programs ~~will be~~ of that web application will be executed by servlet container and Jsp programs will be executed by Jsp Container.
* Directly or indirectly every java application (or) java resource (or) java Component uses JVM during execution.

Ex: appletviewer executes our applet program but this appletviewer internally uses (or) take support from JVM.

servlet Container executes servlet program but this servlet Container internally uses JVM support

Different server side Technologies (to develop server side web resource programs)

servlet → from sun microsystems → java based (2)

jsp → from sun microsystems → java based (3)

asp → from ~~miser~~ microsoft → (non-java based)

asp.net → from microsoft → (non-java based) (4)

PHP → from Apache → (non-java based) (5)

COLD Fusion → from Adobe → (non-java based)

SSJS → from Netscape → (non-java based)
(server side
java script)

Different web server softwares

Tomcat → from Apache → java based (1)
Resin → from Resin soft → java based (2)
JWS → from Sun microsystems → java based
Jetty → from Adobe → java based
} for servlet, JSP Programs

IIS → from microsoft → Non-java based (3)
PWS → from microsoft → (Non-java based)
} for asp, asp.net Programs

AWS → from Apache → Non-java based (4) (for php programs)
(pache web server)

Application server = web server + EJB Container + more middleware services

NOTE: EJB container is required to execute EJB Components.

A reusable java class object is called as component.

Application server software is enhancement of web server software.

Every application server software can act as web server.

Different application server softwares

web logic → from BEA systems (oracle corporation) (1)

web sphere → from IBM (3)

Jboss ⟶ from Apache / Red Hat (4)

GlassFish ⟶ from Sun micro systems (2)

OlogAS ⟶ from Oracle Corporation
(oracle log
Application server)

Jrun ⟶ from macromedia (Adobe)

*All application server softwares are java based softwares.

Different database softwares

oracle ⟶ oracle corporation (1)

mysql ⟶ from DevX (3)

SQLServer ⟶ from microsoft (2)

DB 2 ⟶ from IBM

postgresql ⟶ from university of California (4)

* We can use client side technologies like html, java script along with any vendor supplied server side technologies in web application development.

* To develop & execute java based web application

a) choose any browser software

b) choose one or more client side technologies like HTM, Java script

c) choose servlet, JSPs as server side technologies.

d) choose any java based web server software or application server software

e) choose any data base software.

* To develop and execute .net based web application.

a), b), e) are same as above

c) choose asp.net as server side technology.

d) choose microsoft supplied IIS, PWS server.

* To develop PHP based web applications

a), b), e) are same as above

c) choose PHP as server side technology.

d) use AWS as web server.

* To develop medium scale and large scale web applications use Java environment.

* To develop small and medium scale web applications use .net environment.
* To develop small scale web applications use PHP environment.

Tomcat

Type : Java based web server

version : Tomcat 6.0 (Compatable with Jdk 1.6)

Tomcat (7.0) (Compatable with Jdk 1.6/1.7)

Tomcat 5.0 (Compatable with Jdk 1.5)

Vendor : Apache foundation

open source software (the source code will be exposed)

default port no: 8080 (changable)

To download software : www.apache.org

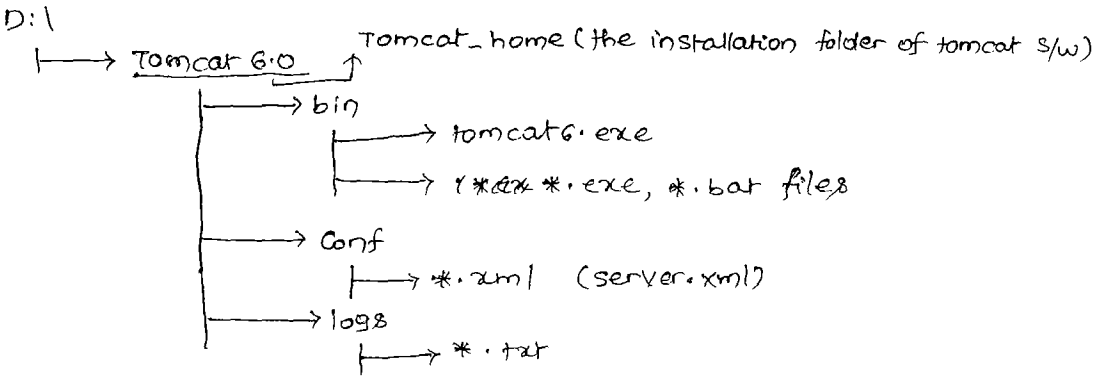
for docs : www.apache.org

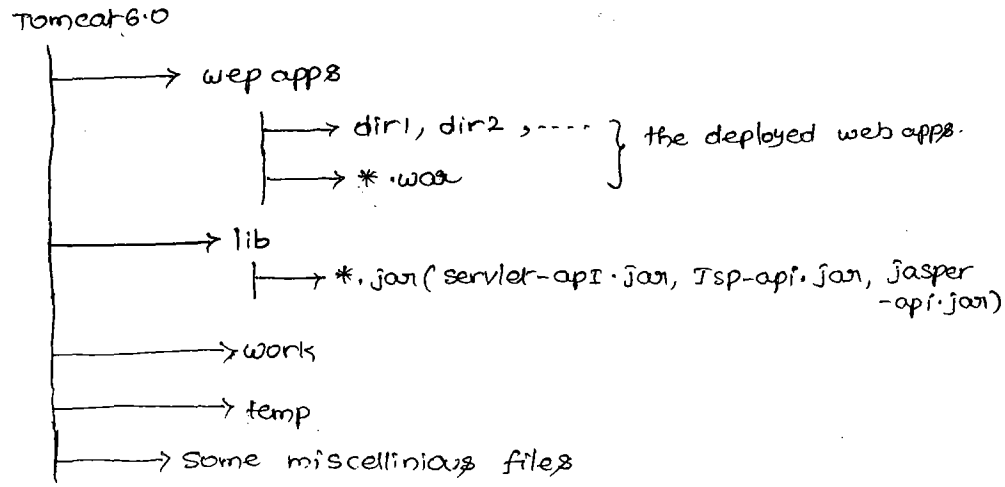
for faqs : www.forum.apache.org

: To install Tomcat 6.0 software use Apache → Tomcat – 6.0.26 set up file and you can change the default port no, default username and password details during installation. If not changed default port no is "8080", default username is "admin", default password is "admin".

(Every software installed in a computer will reside in a logical position called software port and every software port will be identified with its port no. (Communication end point)

After installing tomcat you will get the following folders and files in the installing folder





* The installation folder of tomcat is called as <tomcat_home>.
* To start tomcat server use <tomcat_home>/bin/tomcat6.exe file
* To launch home page of tomcat the procedure is

open browser window → Type http://localhost:8080/ in address bar

↑ port no. of tomcat server

<u>Procedure to change port no. of tomcat server after installation</u>:

Goto <tomcat_home>/conf/server.xml file and modify port attribute value of first <connector> tag → restart the server

* The browser window keeps every web page in the buffer before displaying it for end user.
* A buffer is a temperory memory which can hold the data for temperory period.
* while giving new port no. to any software service use 1025 to 65535 range number because 1 to 1024 range numbers are busy for with OS services.
* Programmer can supply Java web application to tomcat server either in the form of directory or war file.
* Programmer uses <tomcat_home>/webapps folder to deploy web applications.

* A specification is a document or reference that contains set of rules and guidelines to develop the softwares.
* There are two types of specifications.

(1) open specification (Any one can develop softwares based on this specification)

Ex : JDBC specification, servlet specification, JSP specification and etc.

(2) proprietary specification (only specific company which has given specification is allowed to develop the software)

Ex : .net specification.

servlet, jsp, ejb, jms, java mail and etc are the sub specifications of JEE specification.

: servlet specification contains rules and guidelines to develop servlet container.

* JSP specification contains rules and guidelines to develop JSP container.
* EJB specification contains rules and guidelines to develop EJB container by using various sub specifications of JEE the web server and application server softwares will be developed and these server softwares contains various containers like servlet container, JSP container and etc...
* JEE and its concepts like servlets, JSP, EJB and etc are not installable softwares because they are just given as specification.
* working with web server and its containers or application server and its containers is nothing but indirectly working with JEE and its concepts. (like servlet, JSP and etc..)

: Every software specification contains an API representing rules and guidelines. These rules are nothing but methods declared in the interfaces and abstract methods declared in the abstract classes. These guidelines are nothing but concrete methods defined in abstract classes and concrete classes.

while developing software based on software specification rules will be implemented (must) and guidelines will be utilized (optional).

* Servlet specification supplied api is the classes and interfaces of javax.servlet package and javax.servlet.http package.
* The methods of interfaces and the abstract methods of abstract classes available in above two packages represents rules of servlet specification whereas the concrete methods defined in classes and abstract classes of above two packages represent guidelines of servlet specification

What is servlets

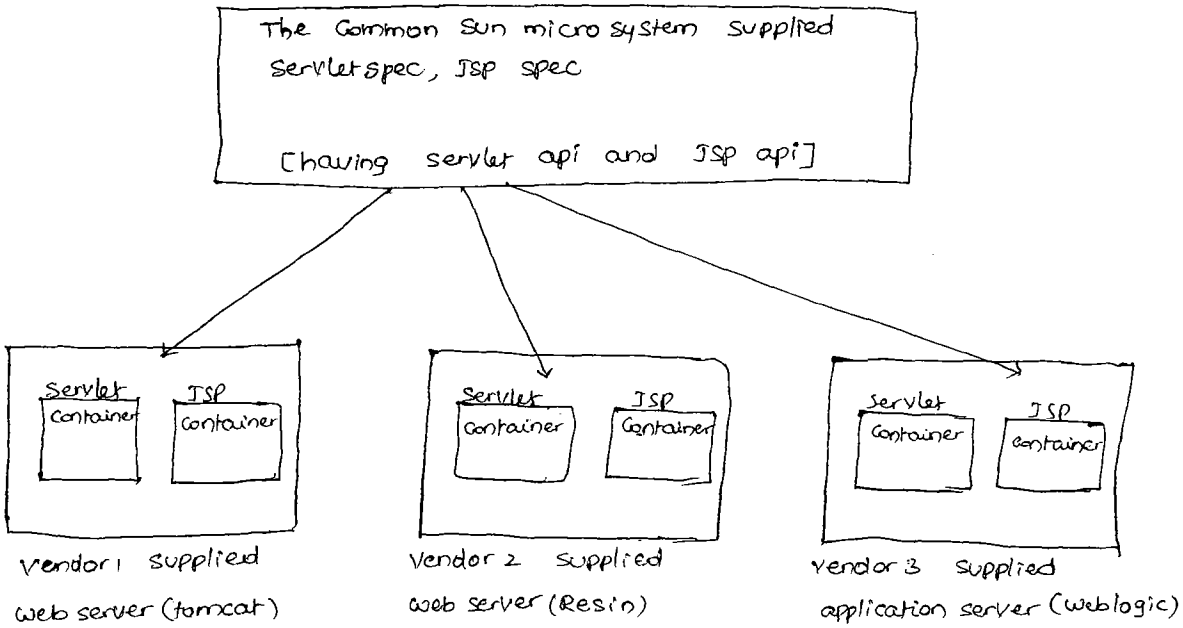
* Servlets is an open/open source api specification that contains set of rules and guidelines to develop servlet container software.
* Based on this servlet specification given by Sun micro systems different vendors develops different servlet container softwares and supplies them to programmers along with web server and application server software installation.
* In earlier days servlet container used to call as servlet engine.
* The tomcat serve the main name of servlet container is Catalina
* Since JEE is not a installable software the tomcat server supplies sun micro system's servlet api in the form of <tomcat_home>/lib/servlet-api.jar and it gives its servlet container software in the form of <tomcat_home>\lib\catalina.jar.
* Every software specification supplied api will be used in two angles.
  (1) vernods companies use specification api to develop the softwares.
  (2) Programmers use specification api to develop programs or applications.
* The servlet api of servlet specification will be used by vendor companies to develop servlet container software. whereas programmers use the same servlet api to develop servlet programs as web resource programs of web application. Due to this servlet container is always capable of executing servlet programs.

What is JSP?

* It is sun micro system supplied an open specification having set of rules and guidelines to develop JSP container.

* JSP specification is enhancement of servlet specification so JSP container is also the enhancement of servlet container.
* In tomcat server JSP container is called as Jasper.
* The tomcat server supplies sun microsystems JSP api in the form of <tomcat_home>\lib\ jsp-api.jar and supplies its own JSP container in the form of jasper.jar.
* The sun microsystems JSP api contains following packages

javax.servlet.jsp
javax.servlet.jsp.el
javax.servlet.jsp.tagext



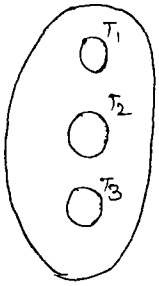
: Since all webserver softwares and their container softwares given by different vendors are developed based on common Sun micro system supplied servlet specification, JSP specification so the way we work with all these web server softwares and their container softwares in java based web applications development is going to be much similar.

Server side Technologies
(required to develope server side ~~web~~ web resource prgs of web apps)

Process Based
CGI - perl

Thread Based
servlets
JSP
asp
asp.net
PHP

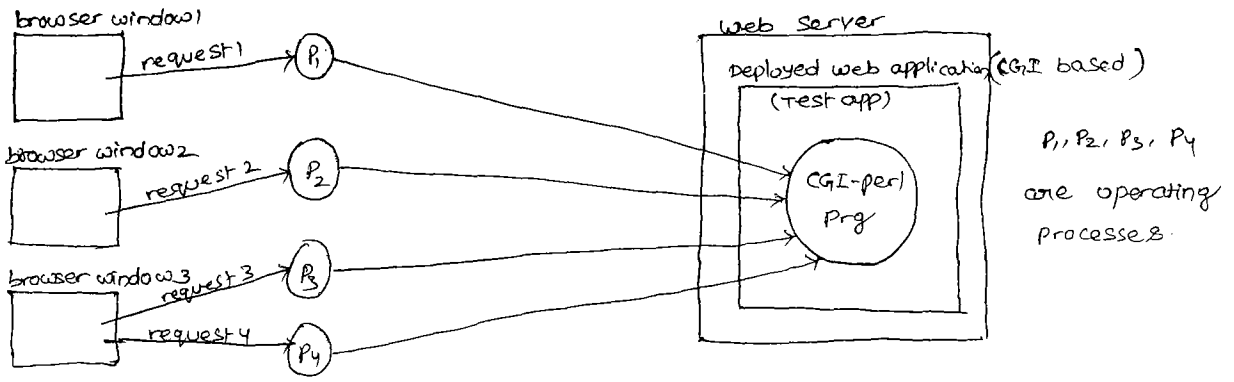
OS managed process



T_1, T_2, T_3 are threads.

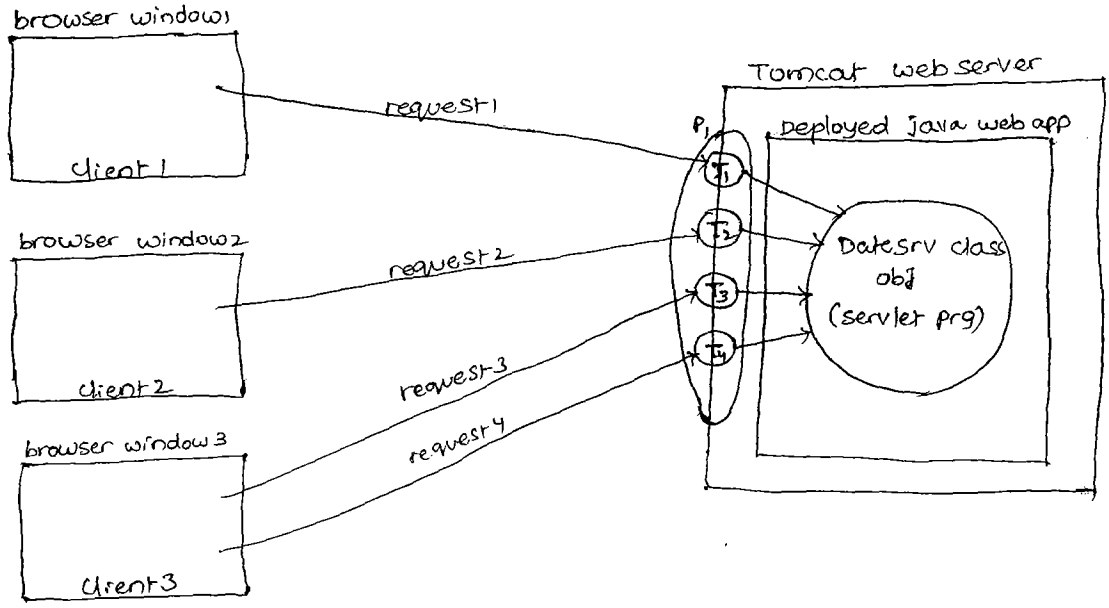
* A thread is a light weight process in OS managed process.
* Every Java application contains two default threads.
  (1) main thread
  (2) Garbage collector thread
* CGI is a specification to develop server side web resource program so perl programming language will be used to develop the webresource programs based on CGI specification.
* perl → Practical extraction reporting language

Understanding Process based serverside technologies. (CGI)



* The procedure of transfering control from one process to another process or from one thread to another thread is called as the scheduling based control jumping or context switching.
* since OS processes are heavy weight processes so they take lot of time for control jumping or context switching.
* Due to this when more requests are given CGI based web application more OS processes will be created on one per request basis and performance of website will be degraded. (That means web application becomes non scalable web application).
* If application is giving same performance irrespective of increase or decrease in number of clients / no. of requests is called as scalable application.
* Every servlet program is a java class using servlet api.

<u>understanding thread based server side technologies like servlet</u>



P1 → OS managed process representing the webserver startup

T_1, T_2, T_3, T_4 are threads started on servlet program related object representing request given by clients.

* The control jumping between threads of a process takes very much less time when compare to control jumping between two seperate OS processes. So we can say thread based server side technologies give better

performance when compared to process based server side technologies

* The web application that is developed based on thread based se r side technologies is scalable application.
* servlet program is a single instance multiple threads based J c component of java web application as shown above. That means 100 requests are given to single servlet program then the servlet C iner creates only one object for that servlet program class but stor 100 threads on that object representing 100 requests as shown a e.

Definitions of servlet

(1) Servlet is server side technology (basically specification) that allow the programmers to develop dynamic web resource programs or server s web resource programs in java based web application.

(2) Servlet is a software specification given by Sun micro systems t provides set of rules and guidelines for vendor companies to d lop the softwares called servlet containers.

(3) servlet is a single instance multiple threads ~~component~~ based J component in java web application to generate dynamic web pages

(4) servlet is a server side technology (basically specification) that c enhance functionalities of web server software or application server so e.

* servlet program is server independent because one servlet program n be executed in any java based web server or application server softwc without modifications.
* securing web application is nothing but applying authentication a authorization on the web application.
* checking the identity of a user is called as authentication.
* checking the access permissions of the user on particular res s of the project is called as authorization.
* For basic information on servlets, servlet container refer the nos 36 to 43.

jee 5 specification

Servlets 2.5 spec
JSP 2.0 spec
EJB 3.0 spec
⋮

jee 6 specification

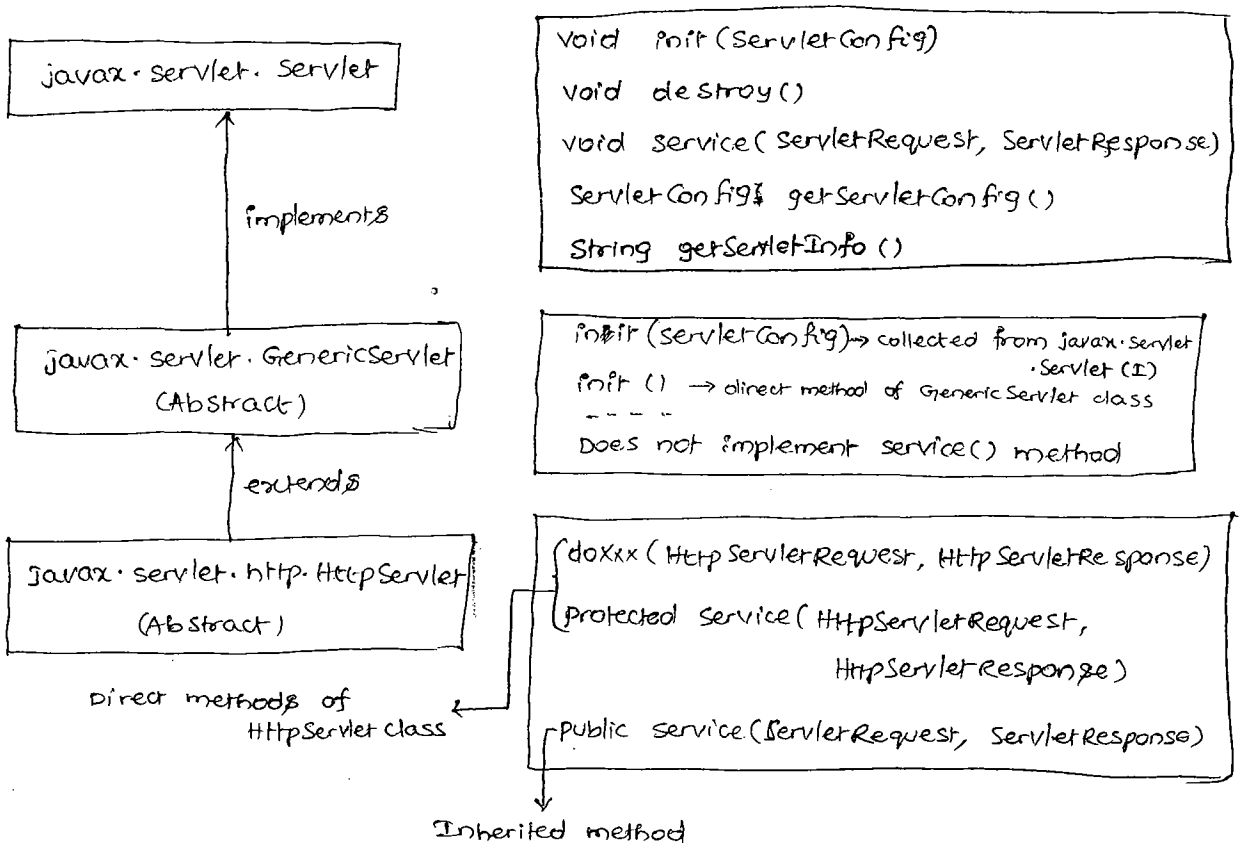
Servlets 3.0 spec
JSP 2.1 spec
EJB 3.1 spec
⋮

* In Jee module of java all are specifications.
* In Jse module of java awt, swing are technologies, JDBC, JNDI are specifications.
* The latest version of the servlet api specification is 3.0 but indu y is using servlet 2.5 api specification.
* Tomcat 6.0 supplies servlet container software based on servlet specification whereas tomcat 7 use servlet container software sed on servlets 3.0 specification.

The 3 important resources of servlet api

javax.servlet.Servlet (I)
javax.servlet.GenericServlet (Abstract class)
javax.servlet.http.HttpServlet (Abstract class)

* In java an abstract class can contain only abstract methods only concrete methods or mix of both.
* The above give predefined HttpServlet class is an abstract class containing no abstract methods.
* If you want to make ~~methods~~ methods/logics of one java cla accessable only through subclasses then make that class as at act class even though that class contains only concrete methods.



* Every servlet program is a java class that is developed based on servlet api. There are three ways to develop servlet program.

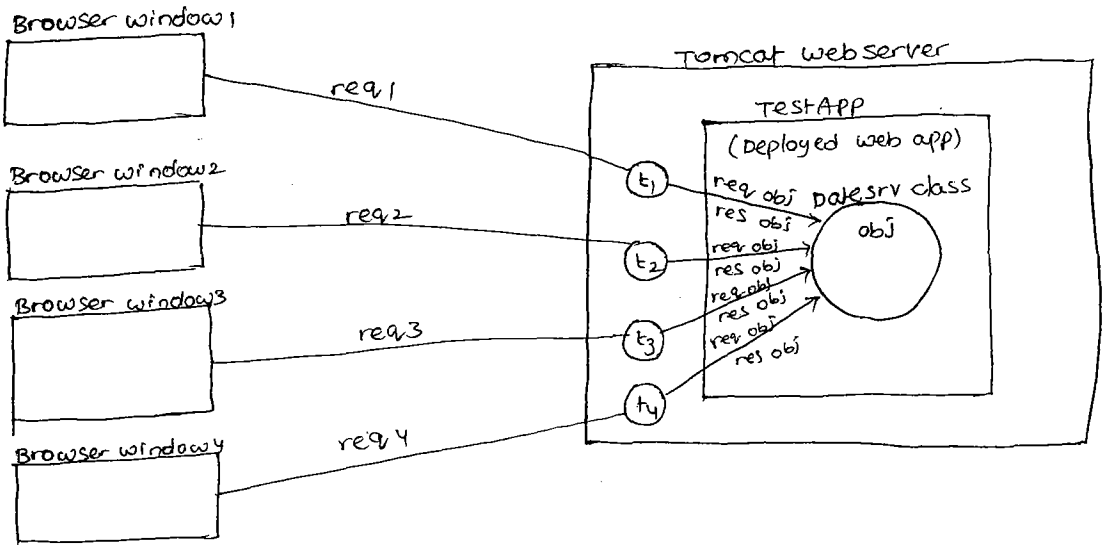
(1) Take a java class implementing javax.servlet.Servlet Interface and provide implementation for all the 5 methods of that interface.

(2) ~~Provide~~ Take java class extending from javax.servlet.GenericServlet class and provide implementation for service() method.

(3) Take java class extending from javax.servlet.http.HttpServlet class and override one of the 7 doXxx() methods or one of the two service(-) methods.

* In the above said three approaches the overridden/implemented service(-,-) / doXxx(-) programmer places request processing logic that generates web page by processing the request. But programmer never calls these methods manually but servlet container calls these methods automatically for every request given by client to servlet program.

* Programmer just supplies his servlet program related java class to servlet container then onwards servlet container is responsible to manage the whole life cycle of servlet program. (from object birth to death every operation will be taken care by container).
* For all the request given to servlet program the servlet container creates one object but for every request given to servlet program, the servlet container creates one seperate request, response objects and calls service (- ,-) method by keeping these requests, response objects as argument values.



* servlet program uses request object to read details from the request like form data.
* servlet program uses response object to send response content to browser window through web server.

when 10 requests are given to a servlet program from single or different browser windows (clients)

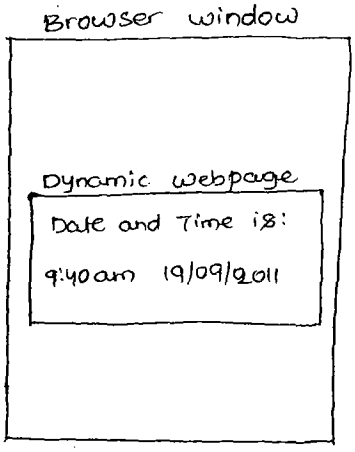
→ Servlet Container creates only one object for servlet program related java class.

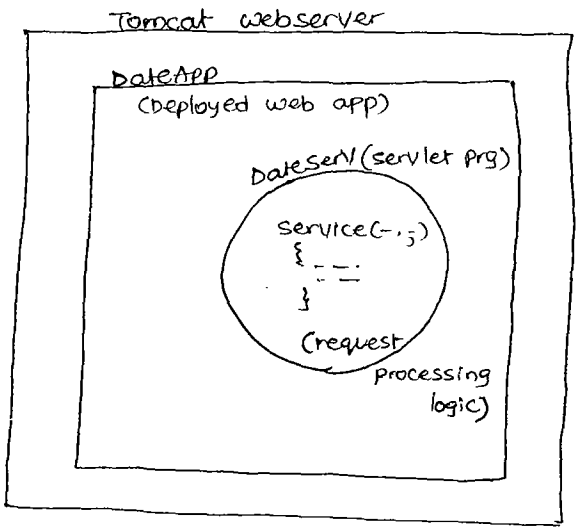
→ servlet Container creates 10 threads on servlet program object representing 10 requests.

→ Servlet Container creates 10 sets of request, response objects and calls service(-,-) or methods for 10 times having request, response objects as argument values.

* To make our servlet program java class visible to servlet container the java class must be taken as public class.

<u>Procedure to develop first java web application by taken servlet program as webresource program</u>

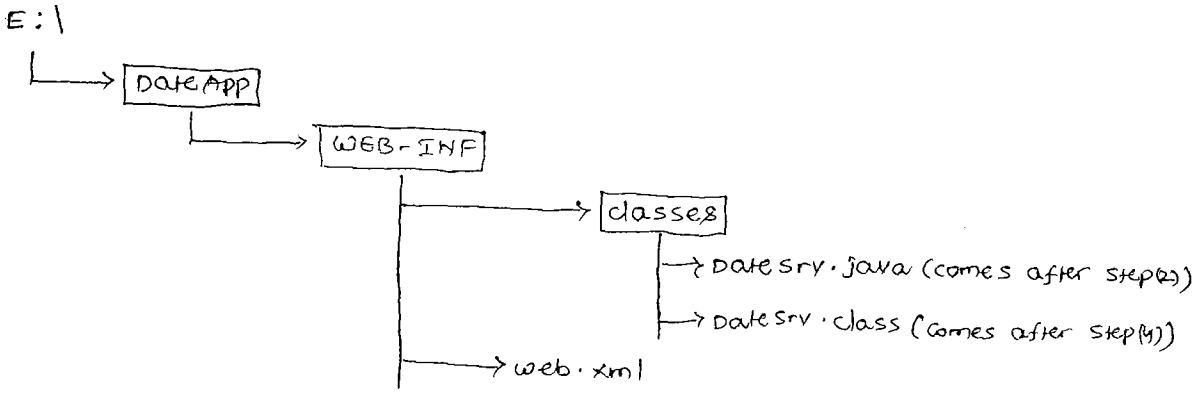




setup required	jdk 1.6 Tomcat 6, IE any version

<u>Step(1):</u> create the following deployment directory structure or packing directory structure to combine multiple webresource programs into single unit.

<u>NOTE:</u> A web application is a directory or war file containing multiple webresource programs. (one or more).



* Here DateApp folder is a webroot folder.
* After deployment this DateApp becomes name of the web application.
* The web application packing or deployment directory structure is common for all the servers and can be created in any location of computer.

Step(2): Develop servlet program (DateSrv.java) save that one in WEB-INF \classes folder of web application.

```
//DateSrv.java
import javax.servlet.*;
import java.io.*;
public class DateSrv extends GenericServlet
{
    //implement service(-,-)
    public void service(ServletRequest req, ServletResponse res) throws
                                    ServletException, IOException
    {
      //Set response Content type
        res.setContentType("text/html");
      //get stream obj
        PrintWriter pw = res.getWriter();
      //write req processing logic
        java.util.Date d1 = new java.util.Date();
      //write response Content to browser window through webserver.
        pw.println("Date and time is:"+d1.toString());
      //close stream object
        pw.close();
    }// service(-,-)
}//class
```

req(1): Represents the servlet Container supplied request object

req(2): Represents the servlet Container supplied response object.

text/html: MIME (multipurpose Internet mail Extention/Content type).

res.setContentType("text/html");

* Represents that passes instructions to browser window through web-server that this servlet program generate html Code base response text.

other mime types

* text/html
* application/ms word
* application/ and etc

printwriter pw = res.getWriter();

* Here pw is a stream object pointing to response object getWriter() called on res object returns the printwriter stream object.

NOTE: A stream object can represent either file or object as destination.

pw.println("---");

* This method makes the stream object pw writing data to the destination object response (res). This response object passes that data (argument value of println("...") method) to webserver and this webserver writes data to browser window as http response in the form of a web page.

pw.close();

* closes the stream connection with response object.

* Add the tomcat server supplied servlet-api.jar (contains servlet api) to classpath.

my computer → properties → advanced variables → environment variables →

variable name: CLASSPATH

value : D:\Tomcat 6.0\lib\servlet-api.jar; <other existing jar files>;.

→ ok → ok → ok

NOTE: Since servlet-api is not part of jdk software and since our servlet program is using servlet api to make java compiler (javac) recognising api the above jar file is added to the classpath.

Step(3): Compile our servlet program source code.

E:\DateApp\WEB-INF\classes > javac DateSrv.java

step (4): Develop web.xml file having the servlet programs configuration.

NOTE :

(1) A program or file of application is called as resource of the application.

(2) servlet program is called as the web resource program of the web application. All servlet programs must be configured in web.xml file.

(3) specifying the details of certain resource program and making underlaying software like container, server and etc recognizing the resource programs is called as resource configuration. By configuring servlet programs in web.xml file we make servlet container recognising servlet programs.

web.xml

```
<web-app>
    <servlet>
      <servlet-name> abc </servlet-name>
      <servlet-class> DateSrv </servlet-class>
    </servlet>
    <servlet-mapping>
      <servlet-name> abc </servlet-name>
      <servlet url-pattern> /test </url-pattern>
    </servlet-mapping>
</web-app>
```

→ logical name/ object name for our servlet class when servlet container creates the object.

→ java class that is acting as servlet program

→ must match ~~abc~~ with above logical name

→ url pattern of servlet program

* servlet program will be identified in the web.xml file through its logical name (abc). more ever servlet container uses this logical name as the object name when it creates object for our servlet program java class.

* The web server servlet container and the web resource programs of web application and clients identifies the servlet program through its url pattern (/test).

* web.xml file is called as deployment descriptor (dd) file because servlet container reads and verifies the entries of web.xml file the moment web application is deployed.

* web.xml file is called as configuration file because it contains servlet program configurations.

* physical presence of servlet program in WEB-INF/classes folder is not sufficient to make servlet container to recognize servlet program. It must be configured in web.xml file.

NOTE: step(1) to step(4) represents web application development.

Step(5): Start the tomcat server.

Tomcat_home\bin\tomcat6

Step(6): Deploy the above DateApp web application

copy e:\DateApp folder to Tomcat_home\webapps folder.

NOTE: Step(5) and Step(6) performs the deployement of web application.

Step(7): Test the web application

open browser window → type http://localhost:2020/DateApp/test ↵

Protocol → http
host name & port no. of tomcat server → localhost:2020
web application name/context path/context route → DateApp
URL pattern of DateSrv servlet program → test

NOTE:

* URL pattern given to servlet is useful, to hide the servlet related program class name from end users.
* Java web applications are WODA (write once Deploy Anywhere) applications because
  (1) The deployment directory structure is common for all webservers.
  (2) The web resource programs like servlet, JSPs are server independent programs.

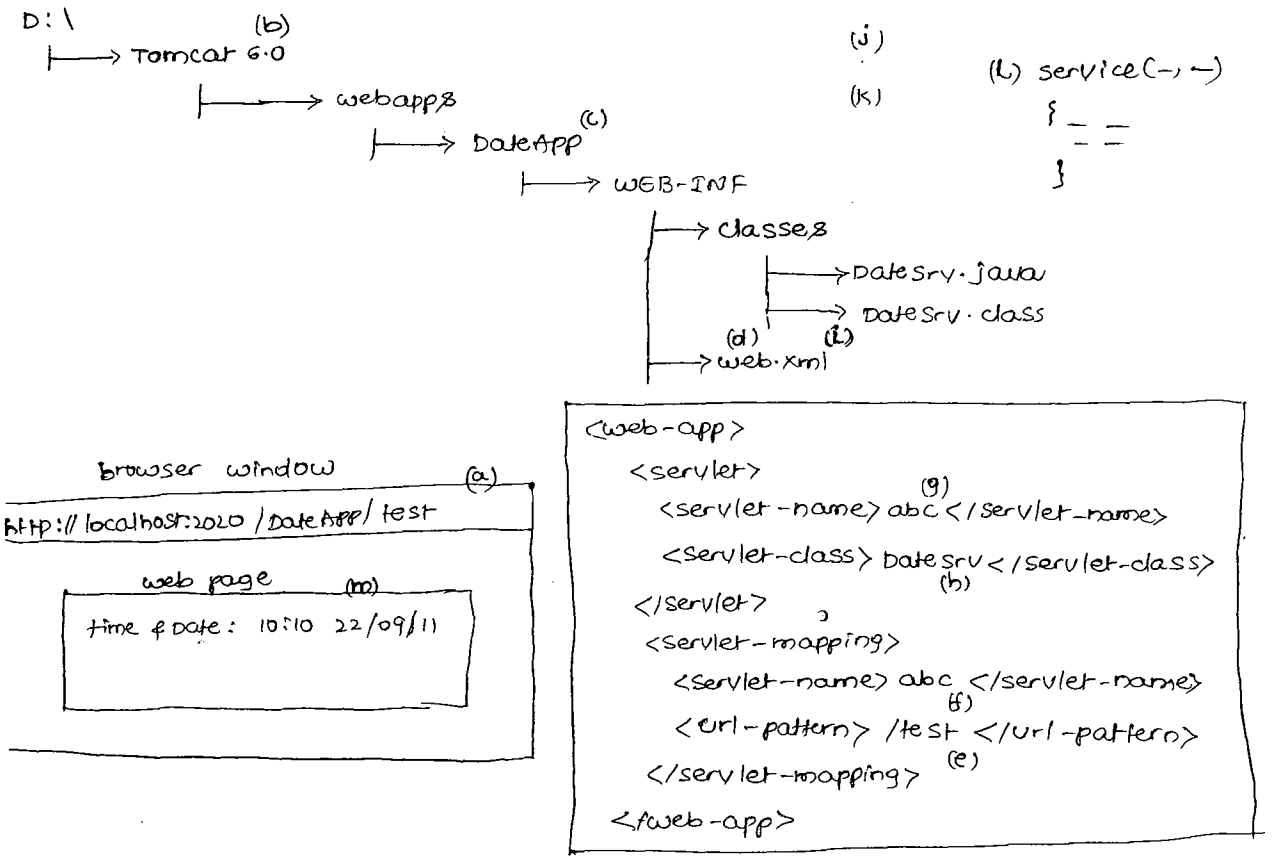
Can you explain flow of execution from request to response generation with the aid of an example application?

With respect to the given code

(a) End user types request URL in the address bar of browser window to generate request to servlet program.

(b) Based on localhost:2020 content of request url the request goes to the tomcat webserver of local machine.

(c) Based on DateApp word of request url the request goes to the



deployed DateApp web application of web server.

(d) The servlet Container uses web.xml file file entries to locate the servlet program that is requested.

(e), (f), (g), (h) Based on the url pattern /test the logical name abc the servlet Container gathers the class name of servlet program (DateSrv).

(i), (j) servlet Container loads DateSrv class from WEB-INF\classes folder of web application.

(j) servlet Container creates and locates DateSrv class object. If created the servlet Container finishes initialization process on our servlet class object.

(k) servlet Container creates one set of request, response object for the current request and calls service (-,-) method by keeping them as argument values.

(l) The service (-,-) method processes the request and generates the output.

(m) This output goes to browser window through webserver as http response in the form of dynamic web page.

PrintWriter pw= res.getWriter();

* In the above statement we are getting access to PrintWriter stream object that is available as built-in stream object of res object. This stream object points to res object as outputStream object so we can use this stream object to write some output content to response object from servlet program.
* Java I/O stream objects can perform read and write operations on the files/objects.
* The System.out.print statements generated output from servlets will appear on server console whereas the pw.println statements generated output will come on webpage of browser window.
* modifications done in web.xml file of deployed web application will be recognized by server automatically whereas the modifications done in servlet program source code will be reflected only after recompilation of servlet program and reloading of the web application.
* To reload web application in Tomcat web server the procedure is

open Tomcat home page (http://localhost:2020/ → Tomcat manager →

user name : admin
password : admin } choosen during installation

→ DateApp application → Reload.

* Servlet Container uses 0-arg Constructor to create object of our servlet class so make sure that our servlet class is having 0-arg Constructor directly or indirectly. Directly means programmer should place 0-arg constructor explicitly. Indirectly means servlet program deals with the javac generated default 0-arg constructor.
* The service method placed in servlet program will be called by Servlet Container for every request that is given to servlet program from

any browser window (client).

javax.servlet.ServletRequest (I)

↑ extends

javax.servlet.http.HttpServletRequest (I)

javax.servlet.ServletResponse (I)

↑ extends

javax.servlet.http.HttpServletResponse (I)

* The req object of service(-,-) method is not the object of javax.servlet.ServletRequest interface, it is the object of servlet Container supplied java class implementing javax.servlet.ServletRequest interface directly or indirectly. But programmer never specifies this class name in servlet program because its ~~chame~~ name changes based on the webserver/servlet container we use. In tomcat server req object class name is "org.apache.catalina.Connector.RequestFacade".
* The res object of service(-,-) method is not the object of javax.servlet.ServletRespose interface, it is the object of servlet Container supplied java class implementing javax.servlet.ServletResponse interface directly or indirectly. In Tomcat server the res object class name is "org.apache.catalina.connector.ResponseFacade".
* To know req, res objects class names in any server call getClass() methods on those objects from service(-,-) method of servlet program as shown below.

```
pw.println("<br> req obj class name is:"+req.getClass());
pw.println("<br> res obj class name is:"+res.getClass());
```

* All the request coming to our servlet program will use only one object of our servlet class but every request gets seperate request, response objects and also starts one seperate thread on servlet class object. ~~(refer~~
* To prove this practically write following code in the service(-,-) method of our servlet class.

```
pw.println ("<br> req obj class name is:" + req.getclass());
pw.println ("<br> res obj class name is:" + res.getClass());
```

gives seperate values for each request

```
pw.println ("<br><br> hash Code of req obj : " + req.hashCode());
pw.println ("<br><br> hash Code of res obj : " + res.hashCode());
```

gives seperate thread name for each request

```
pw.println ("<br><br> current req thread name is:" + Thread.currentThread
                                                    ().getName());
```

gives same value for all the requests

```
pw.println ("<br><br> hashCode of our servlet class obj : " + this.hashCode());
try
{
Thread.sleep (40000);
}
catch (Exception e)
{
    e.printStackTrace();
}
```

NOTE: once server is down all objects (including our servlet class objects) will be destroyed.

How many types of servlets are there?

* There is a possibility of developing n- types of servlets like http servlet, FTP servlet, smtp servlet and etc...
* Since the entire internet and all web server softwares are given based on the protocol http the programmers prefer developing their servlet programs as http servlet programs.
* Javax.servlet.GenericServlet is not a seperate type of servlet. ~~programs~~ It is given as common super class for all protocol specific servlet classes like http servlet and etc...
* As of now servlet api is giving only one sub class for GenericServlet class that is HttpServlet class because all servers are there supporting only http protocol.

How a servlet program developed based on GenericServlet class can work with HttpRequest and HttpResponse?

* Since GenericServlet class is common super class for protocol specific servlet classes. It can take HttpRequest but it can not gather all the

details from that HttpRequest. whereas the servlet program that is developed based on HttpServlet class can gather all the details of HttpRequest.

* Even though there are lot of details in servlet program but the servlet program will be identified with its URL pattern.
* we can provide multiple URL patterns for a single servlet program in web.xml file as shown below.

```
<web-app>
   <servlet>
      <servlet-name> abc </servlet-name>
      <servlet-class> DateSrv </servlet-class>
   </servlet>
   <servlet-mapping>
      <servlet-name> abc </servlet-name>
      <url-pattern>/test </url-pattern>
   </servlet-mapping>
   <servlet-mapping>
      <servlet-name> abc </servlet-name>
      <url-pattern>/test1 </url-pattern>
</web-app>
```

* If web application is having multiple servlet programs then all the servlet programs must be configured in web.xml file having URL patterns.
- By specifying different MIME Types as response content types we can make servlet programs sending web pages to browser window in different formats.
- Every software installed in your computer will provide MIME type or content type for its files, to know them we can use the windows registery editor (regedit) tool
- Browser window can display web pages in different formats based on the response content type of the output that is generated by web resource programs. To know MIME type or content type of any windows operating system managed file, the process is

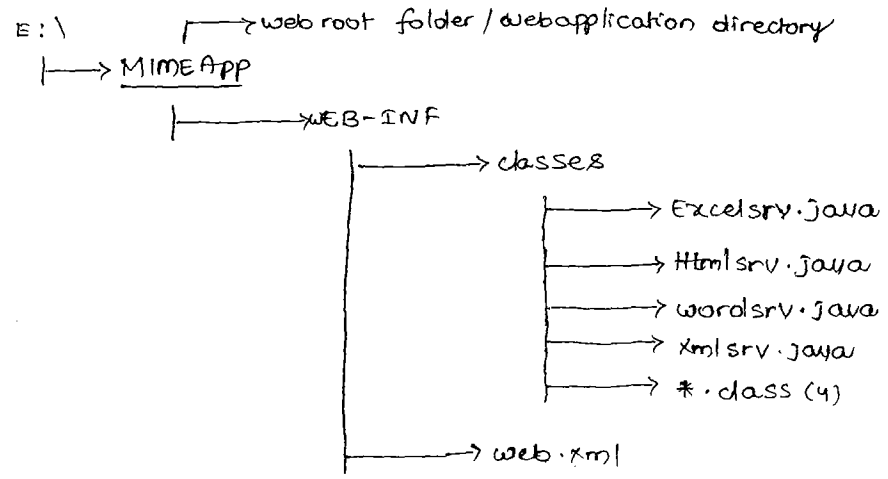
  start → Run → regedit → HKEY_CLASSES_ROOT → choose file extension → get content type from right side panel.

- .doc (ms word) - - - - - - - application/msword
- .xls (msexcel) ⟶ application/vnd.ms-excel
- .html ⟶ text/html
- .xml ⟶ text/xml
- .bmp ⟶ image/bmp
- .avi ⟶ video/avi

and etc...

Develop a java web application having multiple servlet programs and each servlet should generate a different format based web page.
(servlet programs or HtmlSrv, ExcelSrv, WordSrv, XmlSrv)

Step(1): Prepare the deployment directory structure of web application.



Step(2): Develop the source code of above servlet programs.

* For the source code of HtmlSrv.java, WordSrv.java, XmlSrv.java, ExcelSrv.java refer page no.s 58 and 59 of the booklet.

NOTE: The approach no. (1), (2) based servlet programs work with ServletRequest, ServletResponse objects where as the approach no. (3) based servlet program works with HttpServletRequest object and HttpServletResponse object.

NOTE: The simple ServletRequest, ServletResponse objects can't work with all the facilities of the protocol Http where as HttpServletRequest, HttpServletResponse objects can work with all the facilities of protocol HTTP.

Step(3): Add servlet-api.jar file to classpath.

Step(4): Compile the source files of all the servlets programs.

E:\MIMEApp\WEB-INF\classes > javac *.java

Step(5): Configure all the four servlet programs in web.xml file having four different URL patterns.

Refer the web.xml file of page no.s 59 and 60

Step(6): Start the server (Tomcat)

Step(7): Deploy the web application.

copy E:\MIMEApp folder to Tomcat_home\webapps folder

Step(8): Test the web application.

open browser window ⟶ type these URLs

http://localhost:2020/MIMEApp/hturl

http://localhost:2020/MIMEApp/wdurl

http://localhost:2020/MIMEApp/xlsurl

http://localhost:2020/MIMEApp/xmlurl

- A servlet program can generate both static and dynamic web pages. In the above application all the servlet programs are designed to generate static web pages.
- The WEB-INF folder of java webapplication deployment directory structure is called as private directory because it is not visible outside the web-server and it is visible only to that web server where our java web-application is deployed. more ever the web resource programs that are placed in WEB-INF folder or in its sub folders (like classes folder) will be recognised by servlet Container only when they are configured in web.xml file.

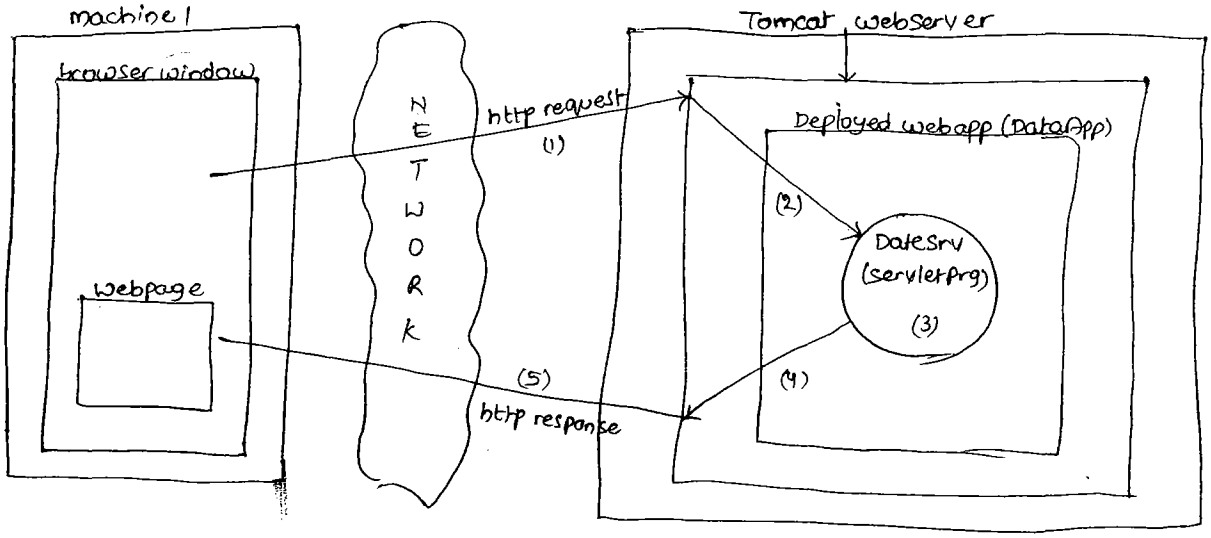
* Since we place our servlet programs in WEB-INF\classes folder so they must be configured in web.xml file.
* If multiple servlet programs of a web application are configured with same URL pattern the last servlet in that list (web.xml order) will be executed for that url pattern based request generation.

Understanding HTTP

* Protocol defines set of rules that are required to get communication between two parties.
* HTTP is a application level protocol that runs over network level protocol called TCP/IP having set of rules to get communication between browser window and web server.
* Application level protocols defines set of rules that are required to get communication between two softwares or software applications.

  Ex: HTTP, JDBC:odbc, Jdbc:oracle, SMTP and etc...
* Network level protocol defines set of rules to get communication between two physical computers of network.

  Ex: TCP/IP



Request url:

http://machine 10 : 2020/DateApp/test ? sno=101 & sname=raja

(query string having req params)

* The query string appended to the request URL sends data to destination web resource program (like servlet) along with the request.
* when request URL is typed in the address bar of browser window the browser window generates one http request having lots of details and we can remember these details as $H^{r}p^{r}$ details.

H ----> http request method (GET/POST/DELETE/.....)

H ——> http request headers (like user-agent, accept, accept-language....)

P ——> request path (http://machine IO:2020/DateApp/test)

P ——> request parameters/query string (sno=101 & sname=raja)

sno → request parameter name

101 → request parameter value

* Browser window is responsible to generate http request having multiple details based on the request URL that is generated.

The structure of http request

http request

misc info like http req method protocol and its version request path of web resource program
req headers user-agent: IE accept: text/html, text/xml.... accept-language: en-US --- ---
<<<< blank line <<<<
request body sno=101 & sname=raja

request body is also called as request payload

An example http request with content

Http request method — path to the resource on web server — Protocol & its version

POST / DateApp / test ? HTTP/1.1 } - misc info

Host: www.nataraj soft.com

User-agent: mozilla/IE 5.0

Accept: text/html, text/xml, image/jpeg, application/msword, */*

Accept-language: en-US

Accept-Encoding: gzip, deflate

Accept-charset : ISO-6431
Keep-Alive : 300
Connection : Keep-Alive
} request headers

<<<< blank line <<<<<<

sno = 101 & sname = raja } request body/ payload having request parameters.

* "GET" method based request can send limited amount of data along with request (max of 256 kilo bytes).
* "POST" method based request can send unlimited amount of data to server along with the request.

<u>What is the difference between request parameters and request headers of http request ?</u>

* Both are there to send input values to target web resource program like servlet program along with the request. The differences are.

Request parameters	Request headers
* Represents end user supplied input values to web resource program.	* Represents the browser window supplied input values to web resource program.
* End user supplies these values either through form page or by appending query string in the URL.	* Browser window internally adds request headers and their values to http request automatically.
* Request parameter names and values are not fixed.	* Request header names are fixed, predefined and their values will be generated by browser window through browser settings.
* multiple request parameters can have same names. Ex: sno = 101, sname = raja and etc.	* Request header names are unique names. Ex: User-agent : IE, accept-language : en-US

* For related information about all http request headers refer page no. 48.

* To know various details about http request page nos ~~pages~~ 46 to 49.
* servlet program can use all the details of http request as input values while processing the request.
* A Simple ServletRequest object can gather only misc info, req param values from http request given by client (browser window).
* A HttpServletRequest object of servlet program can gather all the details from http request (including headers) given by client.

Request URL

http://localhost:2020/DateApp/test?sno=101&sname=raja&sadd=hyd&sadd=vizag

req params

Different approaches of gathering req parameters values/ request body values from Http request being from servlet program

Approach1: (by using req.getParameters(-))

Code in service(-,-) of servlet program

```
String s1 = req.getParameter("sno");  ---> gives "101"
String s2 = req.getParameter("sname"); ---> gives "raja"
String s3 = req.getParameter("sadd");  ---> gives "hyd"
```

NOTE: In this approach we must know req parameter name to get its value, if request parameter is having multiple values than it gives only first value.

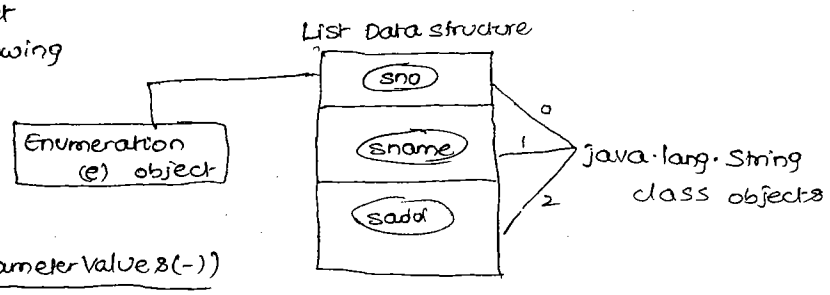
Approach(2) (by using req.getParameterNames())

Code in service(-,-) of servlet prg

```
Enumeration e = req.getParameterNames();  // given enumeration obj pointing to
                                             list data structure having req
                                             param names
while (e.hasMoreElements())
{
    String pname = (String)e.nextElement();  // gives each req param name
    String pvalue = req.getParameter(pname);  // gives each req param value
    pw.Println(pname + " ---> " + pvalue);
}//while
```

gives sno => 101
sname => raja
sadd => hyd

* In this approach we can get req param values without knowing their names.
* If one req param is having multiple values this gives only first value.



Approach(3) (By using req.getParameterValues(-))

```
String s[] = req.getParameter("sadd");
for(int i=0; i<s.length; i++) { pw.println(s[i]+"....."); } //gives hyd---vizag values
String s1 = req.getParameterValues("sadd")[0]; //gives hyd
String s2 = req.getParameterValues("sadd")[1]; //gives vizag
```

* This approach is useful to gather multiple values of single req parameter.
* To read a req param values from http request we can use either ServletRequest obj or HttpServletRequest obj.
* while opening tomcat manager through browser window if username and password related problems are raised, then add following two lines of code in Tomcat_home/user.xml

```
<role rolename="manager"/>
<user username="admin" password="admin" roles="manager"/>
```

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

* According to the above statement setContentType is the method that is declared in javax.servlet.ServletResponse Interface and that method is implemented in Container supplied java class implements this ServletResponse interface.
* In case of Tomcat the implementation class name is responseFacade.
* String s1=req.getParameter("sno"); → gives 101
* In the above statement getParameter is the method that is declared in predefined ServletRequest interface and implemented in servlet Container supplied java class that implements ServletRequest interface (In case of tomcat this class name is RequestFacade).

Ex:

```
interface xyz
{
 public void x();
}
class Test implements xyz
{
   public void x()
   {
    --
   }
   public void y()
   {
    --
    --
   }
}
xyz obj1=new Test(); // Here obj1 is reference variable of interface but not object.
   obj1.x();         // obj1 is object of implementation class Test
```

* when interface reference variable points to its implementation class object then it becomes object of implementation class and it is not the object of interface.

: So when you call method on this reference variable then interface method will not be executed but the method defined in implementation class will be executed.

: Different approaches of gathering request header values from Http request being from servlet program

Approach(1): (by using req.getHeader(-))

Code in service(-,-) of servlet Program

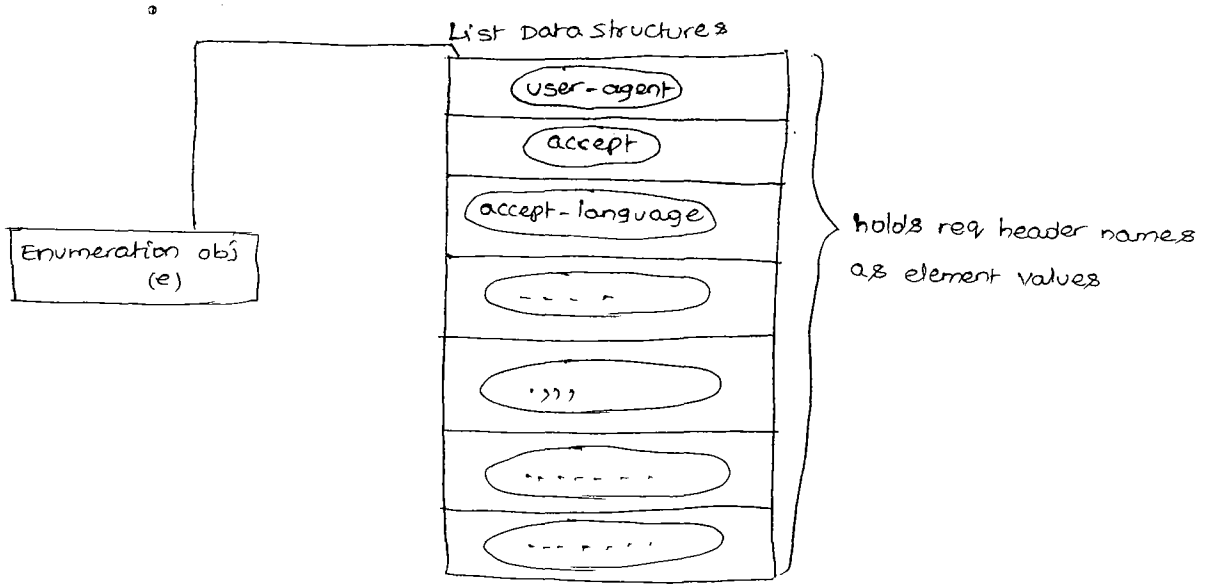
```
String s1 = req.getHeader("user-agent"); //gives browser s/w name
String s2 = req.getHeader("accept"); //gives the mime types supported by browser
                                        window like text/html, text/xml, etc...
```

*In this approach we must know request header name to get its header value.

Approach(2): (by using req.getHeaderNames())

Code in service(-,-) of servlet program

```
Enumeration e = req.HeaderNames(); ①
while (e.hasMoreElements())
{
  String hname = (String) e.nextElement(); ②
  String hvalue = req.getHeader(hname); ③
  pw.println(hname + " ==> " + hvalue);
}
```



1) Gives enumeration object pointing to list data structure having request header names
2) Gives each request header name
3) Gives each request value.

* This code gives all request header names and values.
* The above code must be executed with the support of HttpServletRequest object.
* Through browser language setting we can set or modify accept-language request header value.

  Internet Explorer → tools menu → Internet options → languages → add → choose language.

* In Nestcape Navigator Edit → preferences → Navigator → languages → add → choose language.

How to gather current browser window software name being from servlet program?

(A) use user-agent request header that holds browser software name as shown below.

In service (-) method

```
pw.println("<br><br> the current browser window" + req.getHeader("user-agent"));
```

↓ request header name

* The URL pattern of servlet program is technically called as servlet path.
* we can gather the misc info from HttpRequest by calling various getXxx methods on request object as shown below.

```
//example request url: http://localhost:2020/DateApp/test?sno=101
pw.println("<br> misc info of http request");

//methods available in javax.servlet.ServletRequest (I)

pw.println("<br> req Content length:" + req.getContentLength());
// gives request data length in bytes like 345 bytes (if not known gives -1)
pw.println("<br> req Content type:" + req.getContentType());
//gives req Content type like text/html (if not known gives null)
pw.println("<br>req protocol:" + req.getProtocol());
//gives http/1.1
pw.println("<br> req scheme:" + req.getScheme());
// gives HTTP
pw.println("<br> req browser window machine IP address:"
                                    + req.getRemoteAddr());

//gives 127.0.0.1
```

```
pw.println("<br>browser window machine host name:"+req.getRemoteHost());
// gives current computer name other wise IP address
pw.println("<br>browser window port no:" + req.getRemotePort());
// gives 2922 for Nestcape , 2926 for IE
pw.println("<br>server name"+req.getServerName());
//gives the domain name typed in the URL like local host
pw.println("<br>server port"+req.getServerPort());
// gives 2020

//methods available in javax.servlet.HttpServletRequest (I)
pw.println("<br>Context path:"+req.getContextPath());
//gives DateApp
pw.println("<br>req method is:"+req.getmethod());
// GET
pw.println("<br>req path info is:"+req.getPathInfo());
//gives additional Information kept in req url (otherwise null)
pw.println("<br>query String is:"+req.getQueryString());
//gives sno=101
pw.println("<br>request uri:"+req.getRequestURI());
// gives /DateApp/test
pw.println("<br>request url:"+req.getRequestURL());
//givest http://localhost:2020/DateApp/test
pw.println("<br>servlet path:"+req.getServletPath());
// givest /test
```

* servlet program can avoid browser dependent tags while generating html tags based response ~~if~~ by knowing browser software name through the request header user-agent.

* When web resource program (like servlet program) sends response to browser window the generated HTTP Response contain multiple details including response content. Those details can be remembered as SCH details.

S ⟶ Response status code (100 - 599)

100 - 199 ⟶ Info
200 - 299 ⟶ Success
300 - 399 ⟶ Redirection
400 - 499 ⟶ Incomplete web resource
500 - 599 ⟶ server Error

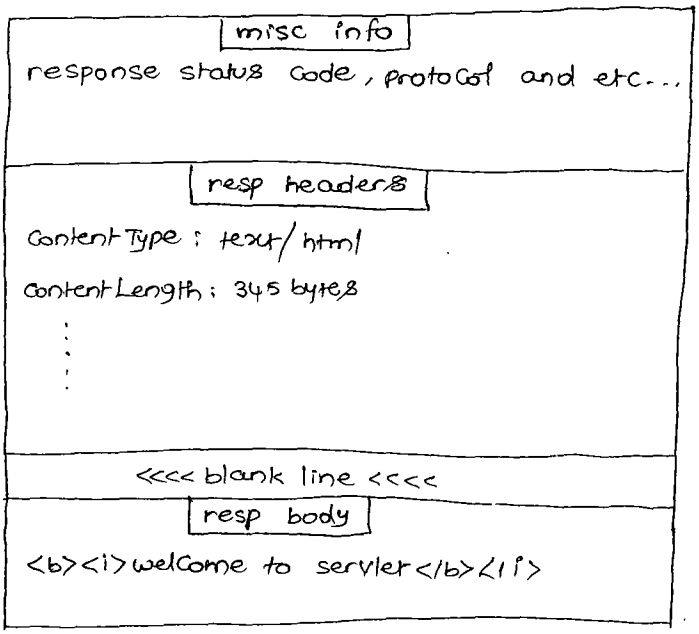
C ⟶ Content type (webpage Content)

H ⟶ response headers

like ContentType, ContentLength, refresh and etc.

* Response status code indicates the status of generated response to display on the browser window.
* Every generated Http response contains one http status code, default status code is 200.
* If web resource program generates warnings based web page then the status code is 100-199.
* If web resource program generates successful web page then the status code is 200-299.
* If request given to one web site is forwarded to another web site then the status code will be 300-399.
* If our web resource program is incomplete or invalid to process the request then the status code will be 400-499.
* If server fails to execute our web application the status code will be 500-599.
* 400-599 are error status codes, using them the programmer can debug the problems related to web application execution and server.
* 100-399 indicates success response status codes that means they display web pages on browser window having output content so these status codes will not appear on the webpages.
* For related information on http response status codes refer page no 49 and 50.
* Response headers provide instructions to browser window through web server towards displaying web pages on the browser windows.
* The content that we have placed in pw.println statements becomes response body http response.

Http Response Structure



resp body also called as resp content/resp payload.

ttp response with example content

Http status code

Http/1.1 200 OK → misc info

Set-Cookie : JSESSIONID = 148937ABEA179EA
Content-Type : text/html
Content-length : 1354
Date : Fri Feb 2008 8:40 34 GMT
Server : Apache-Tomcat /5.0
Connection : close

} response Headers

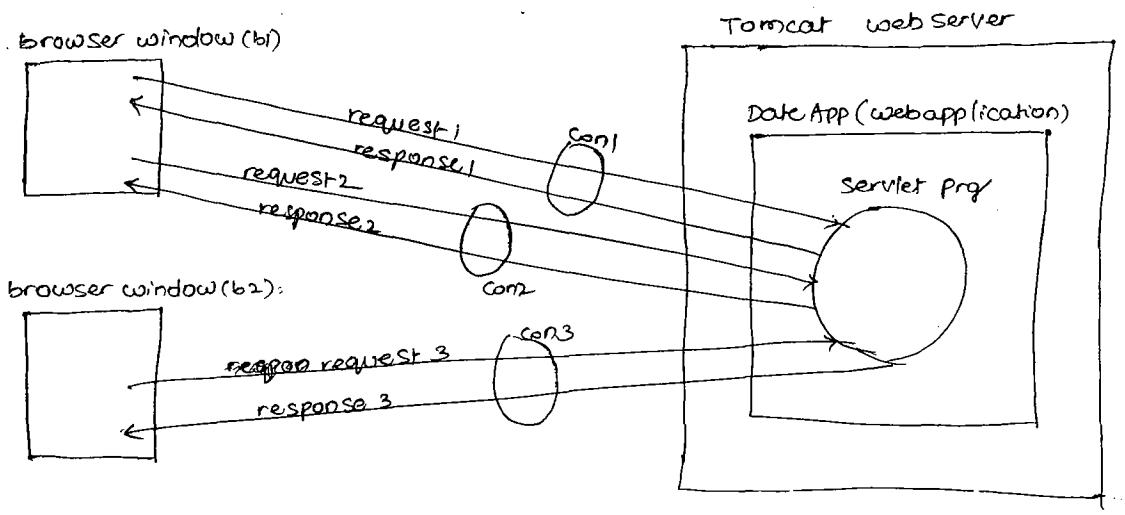
<<<< blank line <<<<

<html> <body> welcome servlet </body> </html> → response body

↓

response content / response body

* For every request generated by browser window one new connection will be created between browser window and web server and this connection will be closed automatically once request related response comes back to browser window.



* For related information on response headers refer page nos 50 and 51.

The list of all http request headers

accept, accept-encoding, authenization, connection, cookie, host, if-modified-since, referrer, user-agent, keep-alive, accept-charset and etc...

List of http response headers

Location, refresh, set-cookie, cache-control / Pragma, content-encoding, content length, ContentType, last-modified, date, server, connection and etc...

How to make browser window refreshing its web page automatically at regular intervals?

* Give instructions to browser window from servlet through server by using response header called refresh as shown below.

```
res.setHeader("refresh", "10");
                          ↓
                     time in seconds
```

By default every web resource program generated output / response content will be stored in the buffer before it is getting displayed on browser window as web page content. Due to this browser window may show old output collected from the buffer even though the web resource program of web application is capable of generating new and updated output.

* To solve the above problem instruct browser window for not storing the output in the buffer from web resource program through web server, by using response headers as shown below.

res.setHeader("cache-control", "no-cache"); for Http 1.1 based servers

res.setHeader("pragma", "no-cache"); for Http 1.0 based servers.

* Buffer is a temporary memory which stores the data for temporary period. Buffer is also called as cache.
* For related information on Http request details, Http response details refer page nos 46 - 51
* Every software and non-software objects life cycle (considering all the operations that are taken place from object birth to object death).
* Our servlet class object life cycle will be managed by Servlet Container.
* Event is an actionPerformed / raised on the object.
* servlet Container raises the following life cycle events in the life cycle of our servlet program. They are

1) Instantiation Event (raises when servlet Container creates our servlet class object)

2) RequestArrival Event (raises when ~~browser window~~ Servlet Container takes the browser window generated request)

3) Destruction Event (raises when servlet Container is about to destroy our servlet class object)

: when Container raises these events on our servlet class object it looks to call certain methods automatically so these methods are called as life cycle methods or Container callback methods. They are

init(ServletConfig cg)

service(ServletRequest req, ServletResponse res)

destroy()

Prototype of servlet life cycle methods

public void service (ServletRequest req, ServletResponse res) throws ServletException, IOException

public void init (ServletConfig cg) throws ServletException

public void destroy ( )

* For one life cycle event only one life cycle method will be called by ServletContainer so init ( ), protected service (HttpServletRequest, HttpServletResponse), doXxx ( ) methods are not life cycle methods.
* Servlet Container calls init (-) life cycle method for instantiation event.
* Servlet Container calls service (-,-) life cycle method for RequestArrival Event.
* ServletContainer calls destroy ( ) lifecycle method for Destruction Event.
* The method that is called by underlaying Container automatically based on the event that is raised on the object is called Container callback methods.
* Programmer never calls life cycle methods manually, they will be called by underlaying Container automatically.
* Servlet API gives supplied life cycle methods
  (i) to allow programmer to place his choice logics in the execution of servlet programs.
  (ii) to supply Servlet Container created objects to servlet program (also for programmer as parameters of life cycle methods. (like request object, response object, ServletConfig object and etc...)
* while overriding super class method in sub class, the method can have same modifier or access modifier.
* while overriding protected service ( ) method of predefined HttpServlet class in its sub class we can take either protected or public modifiers for that method.

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

Here protected method is overridden.

service() method - 1

public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException

service() method - 2

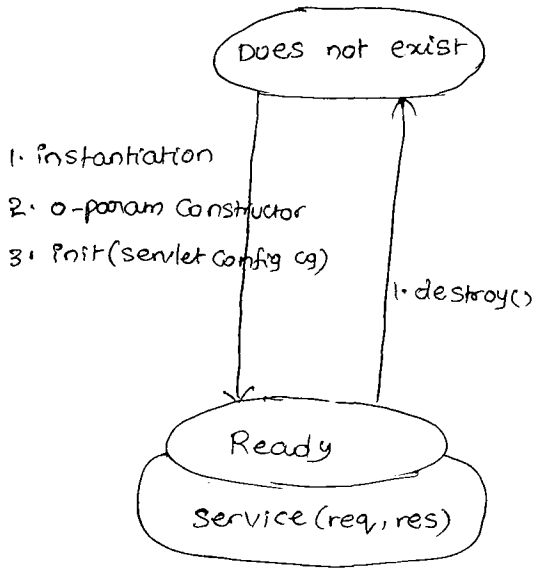
Protected void service(HttpServletRequest req, HttpServletResponse res) throws ServletException, IOException.

* servlet container is responsible to raise various life cycle events on our servlet class objects so logics placed in life cycle methods are not responsible to raise these events but they are responsible to process these events by executing programmers supplied logic.

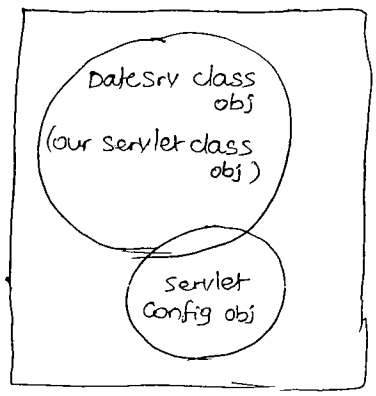
Ex: Code which is written in init() life cycle method never creates our servlet class object but programmer places this code has initialization logic to execute for instantiation event raised by servlet container by creating our servlet class object.

* Life cycle method and its logic are not capable of raising life cycle events on the object but when underlaying container raises life cycle events on the objects it automatically calls a relavent life cycle methods.

servlet life cycle Diagram :



(1)



(2)

(1) → servlet Config object is right hand object to our servlet class object it is one per servlet class object.

(2) → programmer uses this object to pass additional information to our servlet program and to gather information from our servlet program.

<u>what happens when our servlet program gets 1st requeqest from browser window?</u>

(1) Servlet Container loads our servlet class from WEB-INF\classes folder of deployed web application.

(2) servlet Container instantiates (object creation) our servlet class object as Class.forName("DateSrV").newInstance();

* class.forName("DateSrV") → loads our servlet class
* newInstance() → creates object for the loaded class DateSrV.

(3) During instantiation process the 0-param constructor of our servlet class executes.

(4) Servlet Container calls init(-) life cycle method having ServletConfig object as argument value on our servlet class object.

(4) servlet container creates one servlet Config object for our servlet class object.

* (1) to (5) steps completes instantiation and initialization on our servlet class object.

(6) servlet Container calls next life cycle method service(-,-) on our servlet class object. This will process the request and generated response goes to browser window as web page through web server.

<u>what happens when our servlet program gets other than first request?</u>

* Servlet Container checks the availability of our servlet class object.

(1) if available, servlet container calls service(-,-) (Public) on existing object of our servlet class to process the request and this response goes to browser window.

(2) if not available, servlet Container performs all operations of 1st request.

Example servlet program to understand the life cycle of servlet

Adding this servlet program to MIMEAPP web application.

Step(1): Add LCTestSrv.java in WEB-INF/classes folder of deployed MIMEAPP web application.

LCTestSrv.java

```
package P;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class LCTestSrv extends HttpServlet
{
  public LCTestSrv()
  {
    System.out.println("LCTestSrv: 0-param Constructor");
  }
  public void init(ServletConfig cg)
  {
    System.out.println("LCTestSrv: init(-)");
  }
  public void service(ServletRequest req, ServletResponse res) throws
                                        ServletException, IOException
  {
    System.out.println("LCTestSrv: service(-,-)");
    PrintWriter pw = res.getWriter();
    res.setContentType("text/html");
    // write business logic
    pw.println("Date and time is"+ new java.util.Date().toString());
    // close stream objects
    pw.close();
  } // service(-,-)
  public void destroy()
  {
    SOP("LCTestSrv: destroy()");
  }
} // class
```

* Compile the servlet program source code.

D:\Tomcat 6.0\webapps\MIMEApp\WEB-INF\classes> javac -d . LCTestSrv.java

Step(2): Configure the above servlet program in web.xml having url pattern.

In web.xml (already some code is there it is added to that one)

```
<servlet>
   <servlet-name> abcy </servlet-name>
   <servlet-class> p.LCTestSrv </servlet-class>
</servlet>
   <servlet-mapping>
     <servlet-name> abcy </servlet-name>
     <url-pattern> /lck/url-pattern>
   </servlet-mapping>
```

Request URL to test APP is

* The servlet program is there in package that must be specified in web.xml while configuring that servlet program.

Step(3): Start the server

* once webserver is down all objects created and managed by that server will be destroyed automatically (including our servlet class objects).
* init(-) is a one time execution block of servlet program life cycle.
* This method executes when servlet container raises instantiation event on our servlet class so the programmer generally keeps initialization logic like creating JDBC connection in this method.
* service(-,-) method is repeatedly executing block of servlet program life cycle. servlet Container calls this method for every request given to servlet program. So programmer generally keeps request processing and response generation logics in this method.
* destroy( ) is the one time execution block of our servlet program life cycle. Servlet Container calls this method when it is about to destroy our servlet class object. So programmer generally keeps uninitialization logic like closing JDBC Connection in this method.

When does servlet container creates our servlet class object?

When <load-on-startup> is not enabled on servlet program

(a) When servlet program gets first request from client.

(b) When servlet program gets first request after restarting web application.

(c) When servlet program gets first request after reloading the web application.

(d) When servlet program gets first request after restarting the server.

(e) When servlet program gets first request after redeploying the web application.

When <load-on-startup> is enabled

(a) The servlet container creates our servlet class object either during server startup or during the deployment of the web application.

Example code in web.xml

```
<servlet>
    <servlet-name> abc4 </servlet-name>
    <servlet-class> P.LCTestSrv </servlet-class>
    <load-on-startup> 5 </load-on-startup>
</servlet>
```

└→ load-on-startup > Priority value

```
<servlet-mapping>
    <servlet-name> abc4 </servlet-name>
    <url-pattern> /lc </urlpattern>
</servlet-mapping>
```

When servlet container destroys our servlet class object through garbage collector?

(a) When server is stopped or restarted.

(b) When ~~web~~ webapplication is stopped or reloaded.

(c) When webapplication is undeployed.

(d) When our servlet object sits idle continuously for long time.

NOTE: No special priority towards destroying our servlet class objects even though load-on-startup is enabled on the servlets.

Why should we enable load-on-startup on servlet program?

* The first request given to servlet goes to service(--,--) method only after the completion of instantiation and initialization process where as other

than first request coming to servlet program go to service(--,--) method directly. Due to this the request processing and response generation time of first request will be little bit high when compare to other than first request.

To minimize first request request processing, response generation time and equalise that time with other than first request make servlet container to complete instantiation and initialization operations on servlet program before first request either during server start up or during deployment of web application by enabling <load-on-startup> on servlet program.

NOTE: Don't enable <load-on-startup> on every servlet program. Enable it only on those servlet programs of web application which will be gaurenteely requested by clients immediately after deploying the web applications like the servlet programs which generate home pages or main menus and etc..

* when multiple servlet programs of web application are enabled with <load-on-startup> in which order the servlet container creates those servlet program objects will be decided based on their <load-on-startup> priority values.
* High value indicates low priority. Low value indicates high priority

NOTE: <load-on-startup> with negative value is equal to not enabling <load-on-startup> on servlet.

Servlet prgs of WA1 web application

	<l-o-s>
SrV1	1 (I)
SrV2	3 (III)
SrV3	2 (II)
SrV4	10 (IV)

Servlet prgs of WA2 webapplication

	<l-o-s>
SrV1	11 (III)
SrV2	0 (I)
SrV3	2 (II)
SrV4	-1 (Ignores <l-o-s>)

Servlet prgs of WA3 web application

	<los>
SrV1	4
SrV2	4
SrV3	4

Server decides the order

* In Tomcat 5, 5.5 versions "0" is having least priority towards <load-on-startup> priority values. where as in tomcat 6.0 "0" is having first priority or higher priority as shown above.

* If you keep <load-on-startup> tag with out any priority value then that servlet program gets first / higher priority compare to other servlet programs whose <load-on-startup> tags are there with values.

Ex: <load-on-startup></load-on-startup>

↓

This tag <load-on-startup> priority value is zero.

* when no value is passed explicitly as priority value, then the <load-on-startup> tag takes "0" as the priority value.

HTML to servlet Communication :

so far we have sent request to servlet program from browser window by typing request URL in the browser window. In this process to send data along with request we need to add query string to the request URL explicitly. But this work can be done only by technical people. The non technical end users like civil engineers, chemical engineers, kids can't do this work. so they need a graphical user interface to generate request with or with out data. For that purpose we can use either HTML form page or hyperlink to generate request with or with out data.

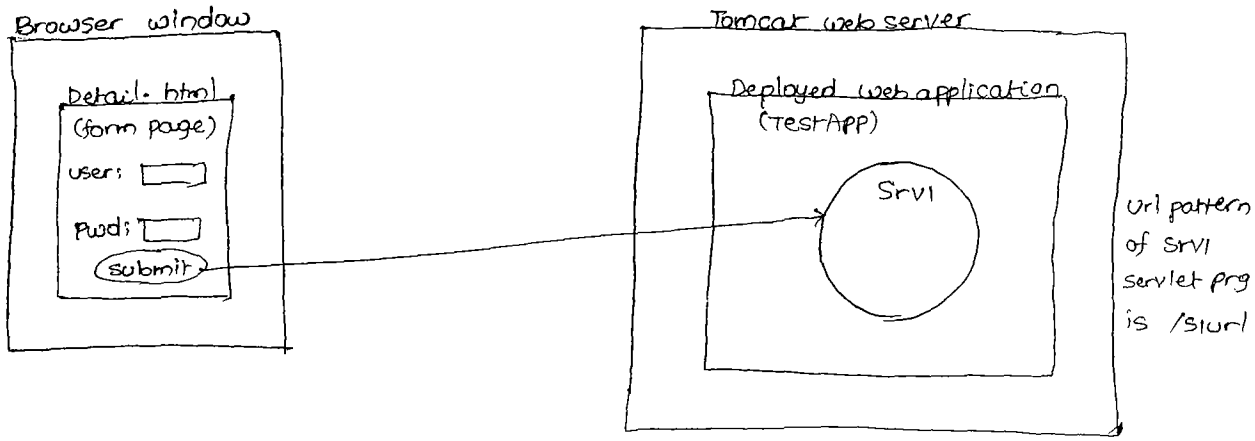
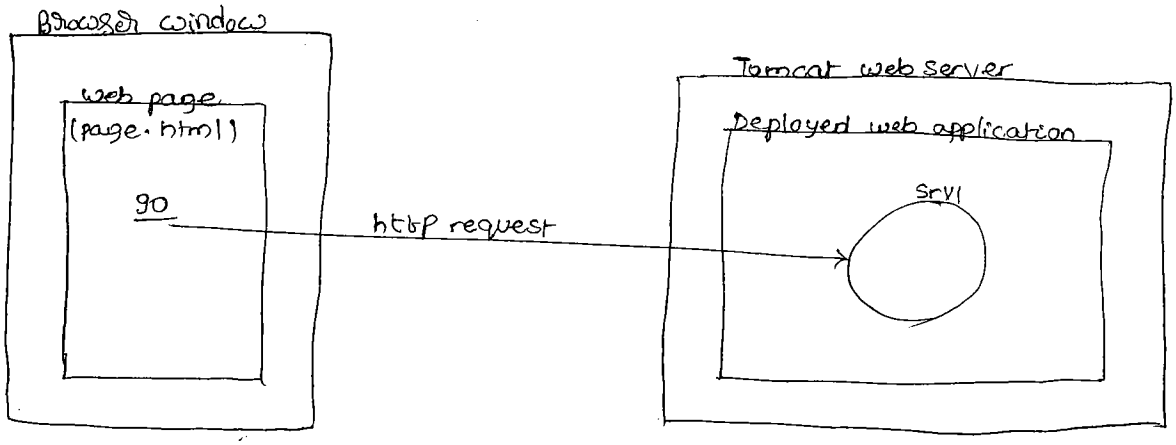


fig: Form page to servlet Communication

* The form page generated request carries form data as request parameters along with request.

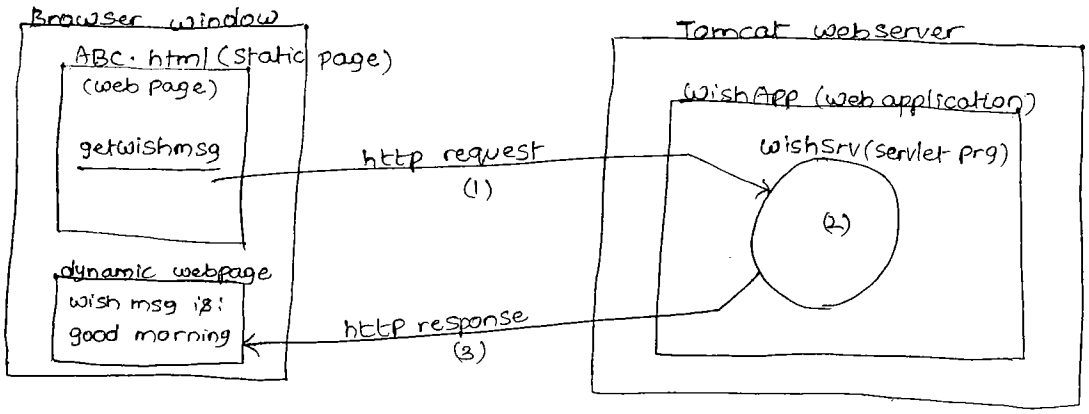
Hyperlink based webpage to servlet communication



Url pattern of srvl servlet Prg is \stUrl

* Generally the hyperlink generated request is blank request that means it can not carry any data along with the request.
* .html files of web application must be placed parallel to WEB-INF folder in deployment directory structure of web application. There is no need of configuring them in web.xml file.

Example application (hyperlink based web page servlet program communication)

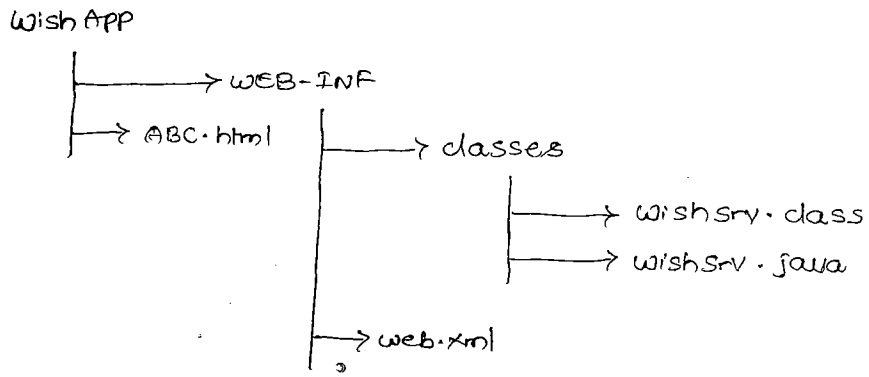


Url pattern of wishSrv servlet Prg is /wurl

* WishSrv servlet program generates the wish message based on the server machine's current time.

* .html based web pages are static pages always, whereas servlet, JSP programs based web pages can be static pages or dynamic pages.

Deployment Directory structure



* place servlet program request URL with URL pattern as the value of h-ref attribute.

↓

in <a> tag to make hyper link generate request communicating with servlet program.

ABC.html

```
<!--web page having hyper links-->
<a href="http://localhost:2020/wishApp/wurl">get wishmsg</a>
```

↓

url pattern of wishSrv servlet program

wishSrv.java

```
//wishSrv.java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*; import java.util.*;

public class wishSrv extends HttpServlet
{
    public void service(HttpServletRequest req, HttpServletResponse res)
                                    throws ServletException, IOException.
    {
        // set response content type
        res.setContentType("text/html");
```

```
//get printwriter obj
PrintWriter pw = res.getWriter();
//write request processing logic to generate wish message
Calender cl = Calender.getInstance(); //gives current date
                                        and time
//get current hour of the day
int h = cl.get(Calender.HOUR_OF_DAY); //in 24hrs format
//generate wish message
if (h<12)
   pw.println("Good morning");
else if (h<16)
   pw.println("Good afternoon");
else if (h<20)
   pw.println("Good evening");
else
   pw.println("Good night");
//close stream obj
pw.close();
}// service(-,-)
}// class
```

web.xml

```
<web-app>
    <servlet>
        <servlet-name> abc </servlet-name>
        <servlet-class> wishSrv </servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name> abc </servlet-name>
        <url-pattern> /wurl </url-pattern>
    </servlet-mapping>
</web-app>
```

NOTE: Do observe that html program is not configured in web.xml

Request URL to test the application

http://localhost:2020/wishApp/ABC.html

* In absolute request URL all details will be specified including URL pattern or identification name of the target web resource program.

Ex: In ABC.html

http://localhost:2020/wishApp/ABC.html

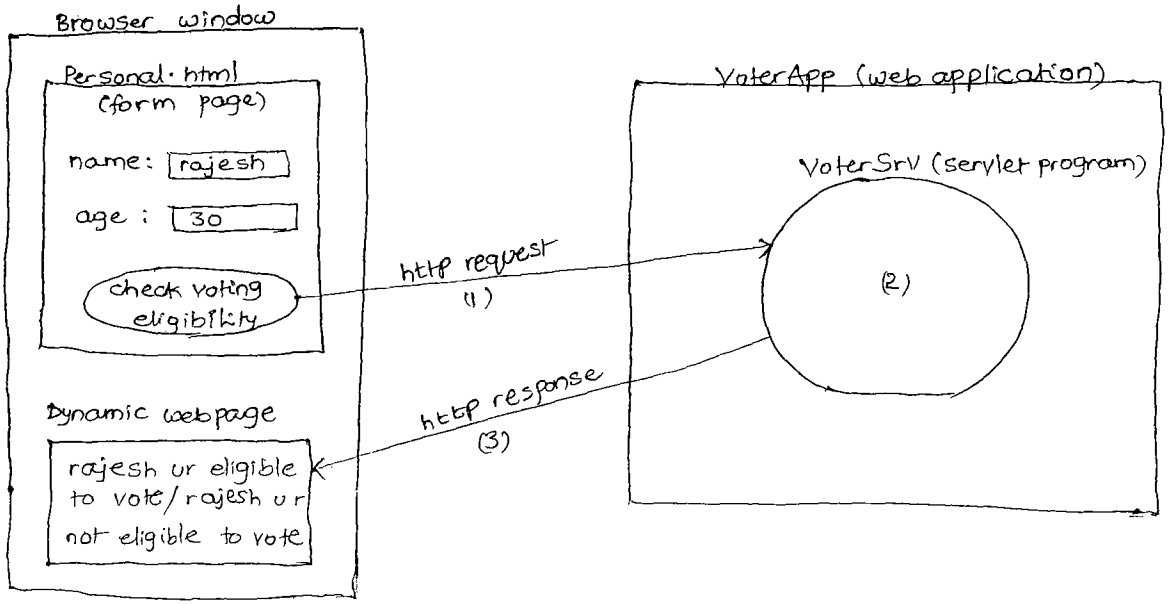
* In relative request URL either URL pattern or identification name of target web resource program will be specified.

Ex: In ABC.html

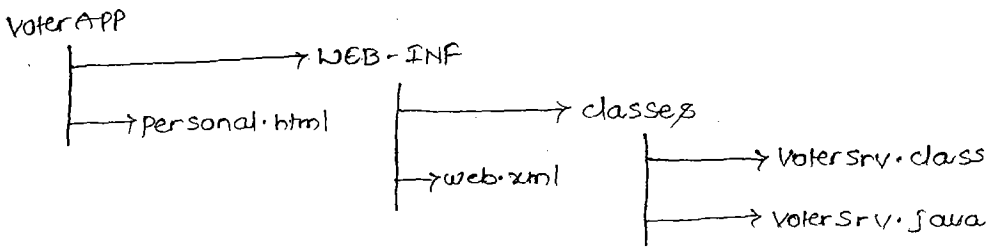
<a href="wurl">getwishmsg</a>

* If source and destination web resource programs are there in the same web application then it is recommanded to relative URL. otherwise it is recommanded to use absolute URL.
* To make web applications as "WODA" applications it is recommanded to use relative URLs.

Example application on form page to servlet communication



Deployment directory structure.



Personal.html

```
<!-- form page -->
<form action="vturl" method="get">
```

↓ URL pattern of VoterSrv servlet program (relative URL)

```
    Name : <input type="text" name="pname"><br>
    Age : <input type="text" name="page"><br>
    <input type="submit" value="check voting Eligibility">
</form>
```

→ name of the component

* When form page is submitted the form component names will go to server as request parameter names (pname, page) and the form component values will go to server as request parameter values.
* The "get" method based request generated by form page is recommended to process by using doGet(-,-) method in servlet program. Similarly use doPost(-,-) method to process "post" method based request generated by form page.
* Since doXxx() methods of servlet api are designed based on HTTP standards so it is recommended to override doXxx() methods in our servlet programs to process the request.

VoterSrv.java

```
// VoterSrv.java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
```

```
public class VoterSrv extends HttpServlet
{
  public void doGet(HttpServletRequest req, HttpServletResponser res)
                    throws ServletException, IOException
  {
    //get printWriter obj
    PrintWriter pw = res.getWriter();
    //set response Content type
    res.setContentType("text/html");
    //read form data from form page as request parameter values
    String s1 = req.getParameter("pname");
    String s2 = req.getParameter("page");
    //convert age value to numeric value
    int age = Integer.parseInt(s2.trim());
    //write logic to generate dynamic web page
    if (age >=18)
      pw.println("<font color='green' size='4'>"+s1+"u r eligible to
                                    vote </font>");
    else
      pw.println("<font color='green' size='4'>"+s1+" ur not eligible
                                    to vote </font>");
    //add hyperlink to dynamic web page
    pw.println("<br><br><a href='personal.html'> home </a>");
  }//doGet(--)
}//class
```

Form Components acting as request parameter names

relative URL

web.xml

configure VoterSrv servlet program with "/vturl" URL pattern

Request URL to test the application

http://localhost:2020/VoterApp/personal.html

* To make our servlet program as flexible servlet program working for both "get", "post" http request methods we can use one of the following three approaches.

Approach (1) : By keeping service(-,-) method in our servlet program

```
public class VoterSrv extends HttpServlet
 {
    public void service (HttpServletRequest req, HttpServletResponse res)
                                    throws ServletException, IOException
    {
        ========  //request processing logic
    }
 }
```

NOTE: The service(-,-) method of our servlet program can process both "get", "post" methods based request. But keeping request processing logic in service(-,-) method is not industry standard so try to keep request processing logic in doXxx methods.

Approach (2): By overriding both doGet(-,-), doPost(-,-) methods but keep request processing logic in one method and call that method from another method

```
public class VoterSrv extends HttpServlet
 {
    public void doGet (HttpServletRequest req, HttpServletResponse res)
                                    throws ServletException, IOException
    {
        ========  //request processing logic
    }
    public void doPost (HttpServletRequest req, HttpServletResponse res)
                                    throws ServletException, IOException
    {
        doGet (req, res);
    }
 }
```

NOTE: Here servlet program can process both Get, Post method based request. Here request processing logic is not duplicated.

Approach (3): keep request processing logic in a user-defined method and call that method from both doGet(-,-), doPost(-,-) methods

```
Public class VoterSrv extends HttpServlet
{
    Public void xyz(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException
    {
        ======  // request processing logic
    }
    Public void doGet(HttpServletRequest req, HttpServletResponse)
                        throws ServletException, IOException
    {
        xyz(req, res);
    }
    Public void doPost(HttpServletRequest req, HttpServletResponse res)
                        throws ServletException, IOException
    {
        xyz(req, res);
    }
}
```

* Here the key benifits are same as approach (2).
* Programmers generally prefer using approach (2) or approach (3) to make their servlet programs as flexible servlet programs against the http request methods "get", "post".

What happens if programmer calls destroy() method explicitly from the service(-,-) method of servlet program?

(A) servlet container will not destroy our servlet class object but logic of destroy() method executes along with service(-,-) method.

* when life cycle event is raised the servlet Container calls life cycle method. Since the programmer has called life cycle method automatically

the servlet container never raises the life cycle event.

What happens if init(-) method is called explicitly from the service(-,-) method of servlet program?

(A) servlet container never creates new object of our servlet class for the above call but logic of init(-) method executes along with service(-,-) method.

What is the difference between HttpRequest method GET and POST?

GET	POST
* Design to gather data from server by generating request.	* Design to send data to the server along with the request.
* GET send limited amount of data along with the request. (max of 256 kb)	* It can send unlimited amount of data along with the request.
* The form page generated query string will appear in browser's address bar so no data secracy.	* The form page generated query string will not appear in browser's address bar. so data secracy is available
* Not suitable for file uploading operations.	* suitable.
* Can not send data in encrypted format.	* can send.
* use doGet() method or service(-,-) method to process the request	* use doPost() method or service(-,-) method to process the request.
* Get is ~~not~~ idempotent	* Post is not idempotent.
* Default request method of HttpRequest.	* is not default and should be applied explicitly.

* It is recommanded to design form pages by having Post method.
* Hyperlink generated requests are get method based requests where as the form page can generate either get or post method based request.
* Before completing the request processing of given request if client is

allowed to generate next request and if the web application is processing both the request then it is called as double posting problem or idempotent problem.

* when form page submit button is clicked for multiple times this problem may raise. To prevent this problem take request method as Post and process that request in doPost(...) method with additional logics. This indicates Post is not idempotent because it can prevent double posting by canceling all the requests.

* The first web page of web application that comes automatically when request is given to web application is called as welcome page or home page of web application.

* In java based web applications the html or jsp programs can be configured as welcome pages.

Ex: In web.xml of VoterApp web application

```
<welcome-file-list>   → must be taken as subtag of <web-app> tag
    <welcome-file> Personal.html </welcome-file>
    <welcome-file> Personal.htm </welcome-file>
    <welcome-file> ABC.htm </welcome-file>
    <welcome-file> ABC.jsp </welcome-file>
</welcome-file-list>
```

only one file will become as welcome file here.

* when no welcome file is explicitly configured the java web application looks to take either index.jsp or index.html as default welcome file.

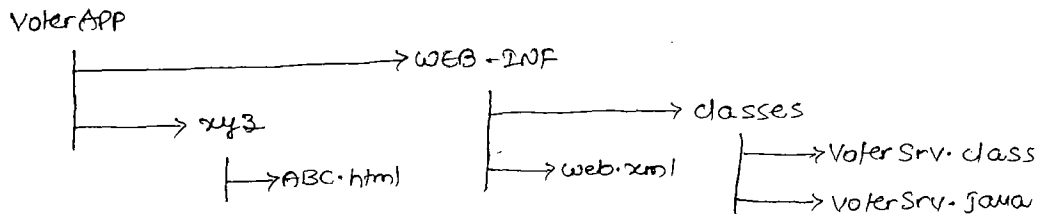
* If both are available the index.html will be taken as default welcome file.

* If multiple welcome files are configured then the web application picks up one welcome file based on the availability and configuration order.

* If all explicitly configured welcome files are not available and if index.html or index.jsp files are available then also web application runs without welcome file.

NOTE: we can not configure a servlet program as welcome file (in Tomcat 5.0, 6.0)

* The html files of web application can be placed in the sub directories of web root folder. while configuring these html files as welcome files we must specify that folder name.



In web.xml

```
<welcome-file-list>
        <welcome-file> xyz\ABC.html </welcome-file>
</welcome-file-list>
```

* Tomcat 6.0, 5.5 server allows to configure servlet program as welcome file by specifying its URL pattern.

In web.xml

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

test → URL pattern of servlet program

* To make our servlet program related java class visible and accessable to servlet container, the java class must be taken as public class.

What happens if I place main(-) method in our servlet program?

(A) servlet program execution will be taken care by servlet container through life cycle methods. Since main(-) is not the life cycle method of servlet program so it will not called by servlet container. only in stand alone applications we need main(-) method to begin the application execution. The container software managed programs or components will be executed through life cycle methods so main(-) method is not required in that components or programs.

How servlet program is executing without main (-) method?

1) The stand alone application that will be executed by JVM directly needs to have main (-) method to begin the execution. Since servlet program is not stand alone application and it is a web resource program which will be executed by servlet container through life cycle methods so there is no need of main (-) method in servlet program.

* If already running java application wants to create certain other java class object and wants to call methods of that class then that other class need not have main (-) method.

* Servlet Container is a continuously running java application/software which creates our servlet class object and calls life cycle methods on that object for executing servlet program. So our servlet program need not to have main (-) method.

—•—

NOTE: Since we never give our servlet program to JVM directly for execution so there is no need of placing main (-) method in our servlet program.

* Verifying the pattern and format of the form data before it is getting used in business logic as input values is called as form validation, and logic written for this is called as form validation logic.

Ex: checking whether required components are filled up with values or not, checking whether age value is given as numeric value or not, checking whether email id is having @, . symbols or not.

What is the difference between form validation logic and business logic?

A) Business logic uses the form data as input values and generates the results where as form validation logic verifies the pattern and format of the form data.

Ex(1): checking whether a, b values are typed or not and checking whether they are typed as numeric values or not comes under form validation

* Performing c = a+b operation comes under business logic.

Ex(2): * checking whether user name and password values are typed or not comes under form validation.

* checking given username and password values against db s/w, username and password details comes under business logic.

Form validations in web applications

Client side	Server side
Use VBScript code or Java script code in form page	write java code in servlet/ jsp programs before b.logic.

* Since script code that is embed with form page executes by coming to browser window we can say client side form validations are good compare to server side form validations because the client side form validation reduces network round trips between browser window and web server.

* Since there is a chance of disabling script code execution through browser settings it is recommanded to write both client side and server side form validations so that there is a gaurentee of performing server side form validations even though client side form validations are not done.

* To disable script code execution in IE

  Tools → internet options → security tab → custom level → scripting → active scripting → disable

* In netscap navigator

  Edit → Preferences → Advanced → Scripts & plugins → Enable java script for Navigator (deselet)

* return statement without value in any java method definition removes the control from current method definition and further statements in that method definition will not be executed.

* For example application that performs both client and server side form validations refer supplementary hand out ~~of~~ given on 13-10-2011

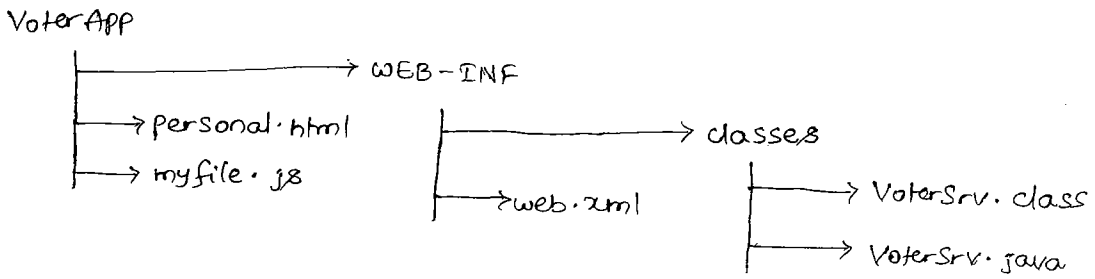
<form action = "vturl" method = "post" onsubmit = "return validate (this)">

* In the above statement the return key word takes the return value of validate function (true or false). ~~If it is true the browser~~ and sends to browser window.

* If that value is true then browser window sends the request to the requested web resource program of web application. If the value is false then the browser window blocks / stops request going to server.
  - true → No validation errors
  - false → validation errors are there.

NOTE: on safe side the software industry writes both client side and server side form validations as shown in the hand out example.

* To hide java script code from end users through view source option. ~~Then~~ and to make javascript code as reusable code for multiple html programs of web application then keep javascript code in .js file and link with html programs as shown below.



Personal.html

```
<!-- form page -->
<html>
    <head>
        <script language = "JavaScript" src = "myfile.js">
        </script>
    </head>
    <form action = "vturl" method = "post" onsubmit = "return Validate (this)">
        - - -
        - - -
    </form>
```

my file.js

```
function validate (frm)
{
    ____
    ____        //refer hand out code
    ____
}//validate (-)
```

* The code that can not be executed independently and it must be embedded along with other technology based program is called as script code.
* Javascript code can not be executed independently and it must be embedded with HTML program for execution. so the Javascript code is called as script code.

what is the difference between java and Javascript?

NOTE: Javascript is no way related with java

Java	Javascript
* It is a programming language to develop all kinds of applications.	* It is a scripting language given for form validations and for other operations.
* Java application code can be executed independently.	* JavaScript code must be embedded with html code for execution.
* Needs JRE/JVm for execution.	* Needs JavaScript engine for execution.
* It is object oriented language.	* It is object based language.

```
Public class Test
{
    Public void x() (b)
    {
        y(); (c)
    }
    public void y()
    {
        ___
    }
}
```

```
public class Demo extends Test
  {
      public void y()
        { (d)
          ____
          ____
        }
  }

    Demo d= new Demo();
    d.x(); (a)
```

<u>understanding</u> <u>init() vs init(ServletConfig cg) methods</u>

* init(ServletConfig cg) is lifecycle method of servlet program and init() is not lifecycle method and it is convienence method given to programmer to place initialization logic of servlet program.

<u>Scenario (1) :</u>

<u>GenericServlet.java</u> (Pre-defined class)

```
public abstract class GenericServlet implements Servlet
  {
      ServletConfig cg;
      public void init(ServletConfig cg)
        { (4)
          this.cg = cg;  //initialization logic of ServletConfig obj
          init();
        }
      public void init()  //empty method
        {
        }
        public ServletConfig getServletConfig() { return cg; }
        ____________  //other methods
  }
```

<u>NOTE</u>: In predefined HttpServlet class there are no init() methods.

(1) (2) (3)

<u>TestSrv.java</u> (our servlet program)

```
public class TestSrv extends GenericServlet/HttpServlet
  {
      public void init()
        { (5)
          ____ Programmer choice initialization logic like creating jdbc
          ____ Connection obj
        }
```

```
public void service( SReq req, SRes res) throws SE, IOE
  {  (6)
     ____
     ____   request processing logic
     ____
  }
}
```

* When servlet Container raises instantiation event by creating our servlet class object then Container calls init (-) method but not init () method. So init (-) method is called as life cycle method and init () method is not life cycle method.

* With respect to the above Code

(1) End user gives first request to our servlet program.

(2) ServletContainer creates our servlet class object (TestSrv) and also creates one servletConfig object.

(3) ServletContainer calls init (-) method as life cycle method by keeping servletConfig object as argument value, on our servlet class object.

(4) Since init (-) is not available in our servlet program the super class (predefined GenericServlet class) init (-) method executes.

(5) The super class init(-) method initializes the received ServletConfig object in GenericServlet class and calls init() method. Since init() method is available in our servlet program that will be executed. Here programmer places his servlet program related initialization logic.

(6) Servlet Container calls next lifecycle method called public void service (-,-) on our servlet class object.

NOTE: In scenario (1) programmer need not to initialise the servlet-config object explicitly in his servlet program. more ever we can call getServletConfig () method to get access to servletConfig object from any lifecycle method of his servlet program.

Scenario (2)

GenericServlet.java

```
  ___
  ___   // same as above code in scenario (1)
  ___
```

```
public abstract class GenericServlet implements Servlet
{
        ServletConfig cg;
        public void init(ServletConfig cg)
          {
            this.cg=cg;
            init();
          }
        public void init()
          {
          }
        public ServletConfig getServletConfig()
          {
            return cg;
          }
        ___ //other methods.
}
```

(1) (2) (3)

TestSrv.java (our servlet prg)

```
public class TestSrv extends GenericServlet/HttpServlet
{
        ServletConfig cg;
        public void init(ServletConfig cg)
          { (4)
            this.cg=cg;
            ___ Programmer choice initialization logic like creating
            ___ Jdbc Connection obj
          }
        public void service(SReq req, SRes res) throws SE, IOE
          { (5)
            ___ request processing logic
          }
}
```

* Here (1), (2), (3) are same as scenario (1)

(4) → init(-) method of servlet program executes having initialization logic

(5) → same as step (6) of scenario (1).

NOTE (1): In scenario (2) the programmer must explicitly initialize ServletConfig object from init(-) method to give access to ServletConfig

object from other lifecycle methods (refer above). If programmer forgets to do this ServletConfig object initialization then the service() destroy() lifecycle methods can not get access to ServletConfig object.

NOTE(2): In scenario (2) control did not pass on to init(-) method of super class (pre defined GenericServlet class) so our servlet program can not call getServletConfig() method to get ServletConfig object.

Scenario (3)

GenericServlet.java (Predefined servlet)

```
Public class GenericServlet implements Servlet
{
   ServletConfig cg;
   Public void init(ServletConfig cg)
   {
       this.cg=cg; (6)
       init();
   }
   public void init()
   { (7)
   }
   Public ServletConfig getServletConfig()
   {
       return cg;
   }
}        ======  //other methods
```

TestSrv.java (our servlet program)

```
(1) (2) (3)
Public class TestSrv extends HttpServlet/GenericServlet
{
   public void init(ServletConfig cg)
   { (4)
     ========  // Programmer choice initialization logic like create jdbc
                                                          connection
     super.init(cg); (5)
   }
   Public void service(ServletRequest req, ServletResponse res) throws SE, IOE
   { (8)
     ========  //request processing logic
   }
}
```

(1), (2), (3) steps of Scenario (3) are same as (1), (2), (3) steps of scenario(1)

(4) init(-) method of our servlet program executes.

(5) Because of super.init(cg) method call the super class (Generic Servlet) init(-) method executes.

(6) init(-) method of Generic servlet class initializes the ServletConfig object contains servletConfig initialization logic

(7) Since init() method is not available in our servlet init() method of GenericServlet class will execute.

(8) same as step (6) of Scenario (1)

NOTE: In scenario (3)

(i) Programmer should call super.init(cg) method in the definition of init(-) method of our servlet program and he can use getServletConfig() method call to get servletConfig object.

(ii) If Programmer forgets to call super.init(cg) method the service(-,-), destroy() lifecycle methods can not get access to servletConfig object even after calling getServletConfig() method.

* The best scenario to keep init() method in our servlet program is scenario (1) (overriding init() method).

* Always give chance to execute init(-) method of predefined GenericServlet class during our servlet class instantiation and initialization process. Because this method contains servletConfig initialization logic and makes programmer not performing that work explicitly.

why servlet API has given init() method when it is not lifecycle method ?

A) Refer the 4 paragraphs of init() V/s init(ServletConfig cg) discussion of page no. 55 of booklet.

* A good servlet programmer always overrides init() method in his servlet program instead of init(-) method (follow scenario (1)).

* self class methods and super class public/protected methods in sub class can be called without object.

In any life cycle method of servlet program

ServletConfig cg = getServletConfig(); //gives access to ServletConfig obj

↓

It is public method of GenericServlet class so it can be called without object in our servlet programs

understanding two service(--,--) methods and seven doXxx(-,-) methods of predefined HttpServlet class

* we can use one of the two service(-,-) methods or one of the seven doXxx(-,-) methods to place request processing logic in our servlet program but only public service(-,-) method is life cycle method of servlet program.

HttpServlet.java (Predefined class)

```
public abstract class HttpServlet extends GenericServlet
{
  public void service(ServletRequest req, ServletResponse) throws SE, IOE
  { (4)
    // type casting
     HttpServletRequest hreq = (HttpServletRequest)req;
     HttpServletResponse hres = (HttpServletResponse)res;
    // calls protected service(-,-) method
      service(hreq, hres);
  }
  // protected service(-,-) / service(-,-) method2
  protected void service(HttpServletRequest hreq, HttpServletResponse hres)
                                                    throws SE, IOE
  { (5)
  //reads the request method of given request
    String method = req.getMethod();
   //calls one of 7 doXxx(-,-) methods based req method
    if (method.equals("GET"))
          doGet(hreq, hres);
    else if (method.equals("Post"))
          doPost(hreq, hres);
    else if (method.equals("head"))
          doHead(hreq, hres);
```

```
        }
protected void doGet(HttpServletRequest hreq, HttpServletRequest hres)
                                        , throws SE, IOE
    {
        ------
        ------   logic send 405 error response to browser window
        ------
    }
protected void doPost(HttpServletRequest hreq, HttpServletResponse hres)
                                        throws SE, IOE
    {
        ------
        ------   logic send 405 error response to browser window
        ------
    }
    ------
    ------   // implementation of other doXxx(-,-) methods sending 405
    ------      error response to browser window.

}// class
```

TestSrv.java (our Servlet program)

```
(1) (2) (3)
public class TestSrv extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res)
    {  (6)
        ------
        ------   our request processing logic
        ------
    }
```

with respect to above code

(1) client gives request to our servlet program having "get" method.

(2) servlet container creates (or) locates our servlet class object. If created servlet container also completes the initialization process.

(3) The servlet container creates servletRequest, servletResponse objects for current request and calls the life cycle method public service(-,-) on our servlet

class object. keeping ServletRequest, ServletResponse objects as argument values.

(4) Since public service(--) method is not available in our servlet program, the super class public service(--) method executes (Predefined HttpServlet class). This method converts ServletRequest object to HttpServletRequest object, ServletResponse object to HttpServletResponse object and calls protected service(--) method having those objects as arguments.

(5) Since protected service(--) method is not available in our servlet program, the protected service(--) method of super class (Predefined HttpServlet class) executes. This protected service(--) method calls doGet() method based on "get" method of given request.

(6) Since doGet(--) method is available in our servlet program that will be executed and generated response goes to browser window.

NOTE: Don't let doXxx(--) methods of predefined HttpServlet getting executed directly (or) indirectly because they send "405" error response to browser window indicating that our servlet is incomplete servlet program towards processing request.

* When request arrival event is raised (means request comes to container) servlet container calls public service(--) method on our servlet class object and it never calls protected service(--) method, doXxx(--) methods for that event. So only public service(--) method is called as servlet life cycle method, protected service(--) and doXxx(--) methods are called convenience methods given to programmer to keep request processing logic in our servlet programs.

* For understanding flow of execution in our servlet programs towards the request processing refer the six scenarios given in page no. 56 of booklet.

* Eventhough public service(--) method is lifecycle method but the good programmer always prefers overriding doXxx(--) methods in his servlet

program to place request processing logics. Because these methods are given based on the http protocols standard. Moreover the super class (Pre-defined HttpServlet class) version of this methods can generate appropriate error messages like 405 when those methods are really not overridden in our servlet program.

* most of the programmers in real world overrides both doGet(-,-) and doPost(-,-) methods in servlet program, but they place request processing logic only in one method and calls that method from another method.
* In java Abstract class can have only abstract methods (or) only concrete methods (or) mix of both.

.. In predefined HttpServlet class when all methods are given as Concrete methods. why that class itself is given as abstract class?

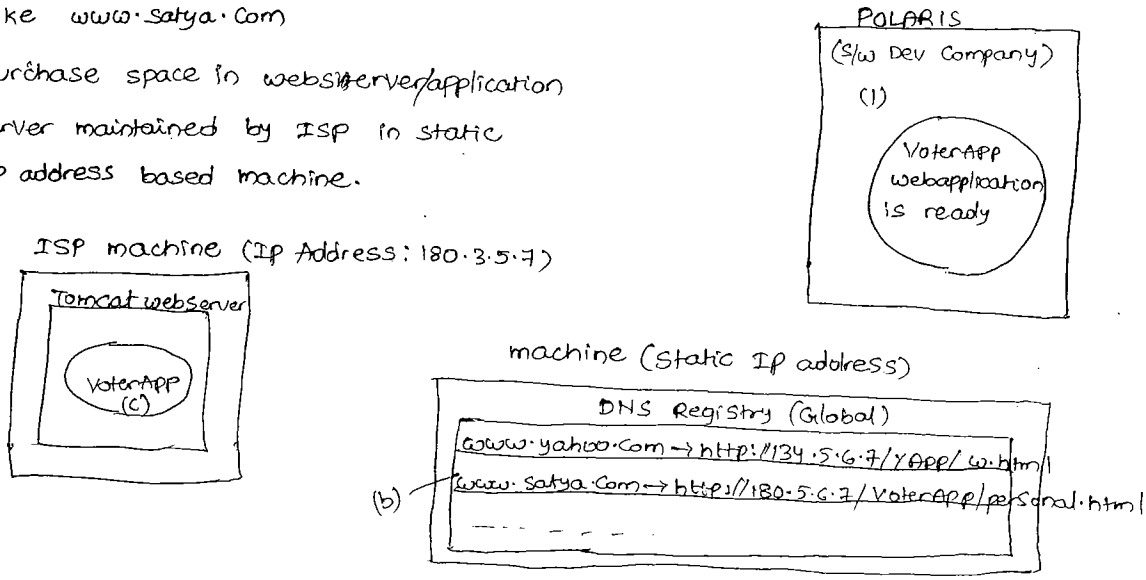
A) Refer the last 3 paragraphs of page no. 56 of booklet.

* Every website that is hosted in the internet will be having one domain name like www.yahoo.com, www.gmail.com and etc...
* These domain names along with home page request URLs will be registered in DNS ~~register~~ registering.

Procedure to host web application on to the internet having domain name

(2) Purchase domain name from ISP like www.satya.com

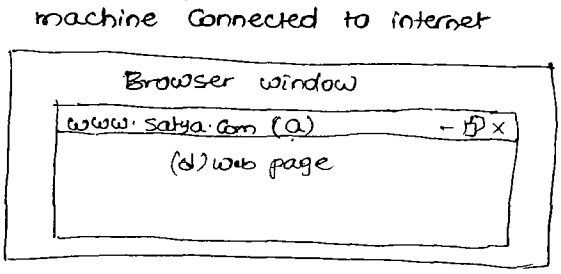
(3) purchase space in webserver/application server maintained by ISP in static IP address based machine.



(4) move VoterApp app the server of ISP machine

(5) register VoterApp app in DNS registry.

(6) Access the web site →



* Registering web application in DNS registry is nothing but keeping domain name of web application along with home page URL.
* The ISP machine where ever web application hosted will be having static/ fixed IP address.
* once we purchase space from the server of Isp machine they supply FTP application along with user name and password to us and we can use that FTP application to interact with our space of ISP machine from any place so using this we host or move web applications to the server of ISP machine.

Flow of accessing

(a) End user types domain name as URL in browser windows address bar.

(b) Through internet network the request goes to DNS registry, gathers home page request URL of given domain name (like satya.com) from DNS registry.

(c) Based on home page request URL the web resource program of hosted web application in tomcat server of ISP machine will be executed.

(d) The web resource program generates one web page to browser window.

* we can add or remove web resource programs in the hosted website of ISP server from our local machine through FTP application. (This is nothing but updating web pages content after hosting).

http request methods

GET
POST
HEAD
DELETE
PUT
TRACE
OPTIONS

* The regularly used two request methods of real world programming are get, post.

GET:

* Given to gather/to get more data from the server.
* The response of this method based request contains both headers and body. ~~Head~~

Head:

* Same as get but this method based request generated response contains only response headers.
* This method is useful to check the availability of web resource programs.

NOTE: Get() method is useful to gather data from server by sending limiter amount of data (upto 256 kb) from the request.

Post:

* Design to send unlimited amount of data along with the request.

Put:

* useful to add new web resource program in web application from client.

delete

* useful to remove web resource program of web application from client.

NOTE: Put(), delete are useful in the development of FTP applications.

Options

* server determines using which request methods the current web resource program can be requested.

NOTE: If our servlet program contains doGet() method overriding then the options request method based request given to that servlet program

returns get, Head, Trace, options.

Trace

* A Trace request method based request sends the flow of execution details of certain web resource program like servlet. So these details can be used for debugging operations.

List of form Components in form page

(1) TextBox
(2) Password Box
(3) select box/ComboBox
(4) List box
(5) TextArea
(6) Radio Buttons
(7) check Box
(8) Button (Standard button, submit button, reset button)
(9) File uploading Component

TextBox

In form page

```
Name: <input type="text" name="pname">
```

In servlet program (to read comp value)

```
String s1 = req.getParameter("pname");
```

Password box

In form page

```
Age: <input type="password" name="page">
```

In servlet program

```
String s1 = req.getParameter("page");
```

TextArea

In form page

```
Address: <textarea name="taddress" rows="4" cols="5">
         enter ur address
         </textarea>
```

In servlet program (to read component value)

```
String s1= req.getParameter("taddress");
```

Select Box / Combo Box (allow us to ~~enter~~ select one item a time)

Qualicatification :

```
< select name = "qlfy" >
    <option value = "engg" >B.E/B.Tech </option>
    <option value = "medico"> MBBS </option>
    <option value = "arts"> B.A </option>
</select>
```

in servlet program

```
String s1 = req.getParameter("qlfy");
```

* while working with Select Box the selected item will not come to server as request parameter value, the value available in the value attribute of option tag for selected item will come to server as request parameter value.

Ex: From the above select Box if BE/B.Tech is choosen then the value engg will go to server as request parameter value.

List box (allows us to select multiple items at a time)

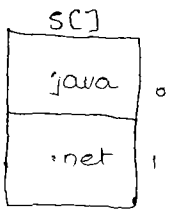
In form page

```
Courses: <select name = "crs" multiple>
    <option value = "java" > JAVA PKG </option>
    <option value = ".net" > .NET PKG </option>
    <option value = "Orade"> Orade PKG </option>
</select>
```

In servlet program

```
String S[] = req.getParameterValues("crs");
```

* If JAVA PKG, .NET PKG items are choosen then S[] holdes java, .net as elements values.



Radio buttons

* By giving same name for multiple radio buttons we can group them into single unit. So only one radio button can be selected at a time.

In form page

```
Gender : <input type="radio" name="g" value="m">Male   
         <input type="radio" name="g" value="F"> Female
```

In servlet program

```
String s1 = req.getParameter("g") ; // gives m when male radio is selected
                                    // gives F when Female radio is selected.
```

checkBoxes

* When multiple checkBoxes are grouped into single unit by giving same name then we can select multiple checkBoxes at a time.

In form page

```
Hobbies : <input type="checkbox" name="ch1" value="read> Reading
          <input type="checkbox name="ch1" value="sleep") sleeping
          <input type="checkbox name="ch1" value="roaming">roaming
```

In servlet program

```
String s[] = req.getParameterValues ("ch1");
```

* If reading, roaming check boxes are selected then the s[] holdes read, roam values.

NetBeans

Type: IDE for java environment

Version: 6.7.1 (compatable with jdk1.6)

vendor: Sun micro system

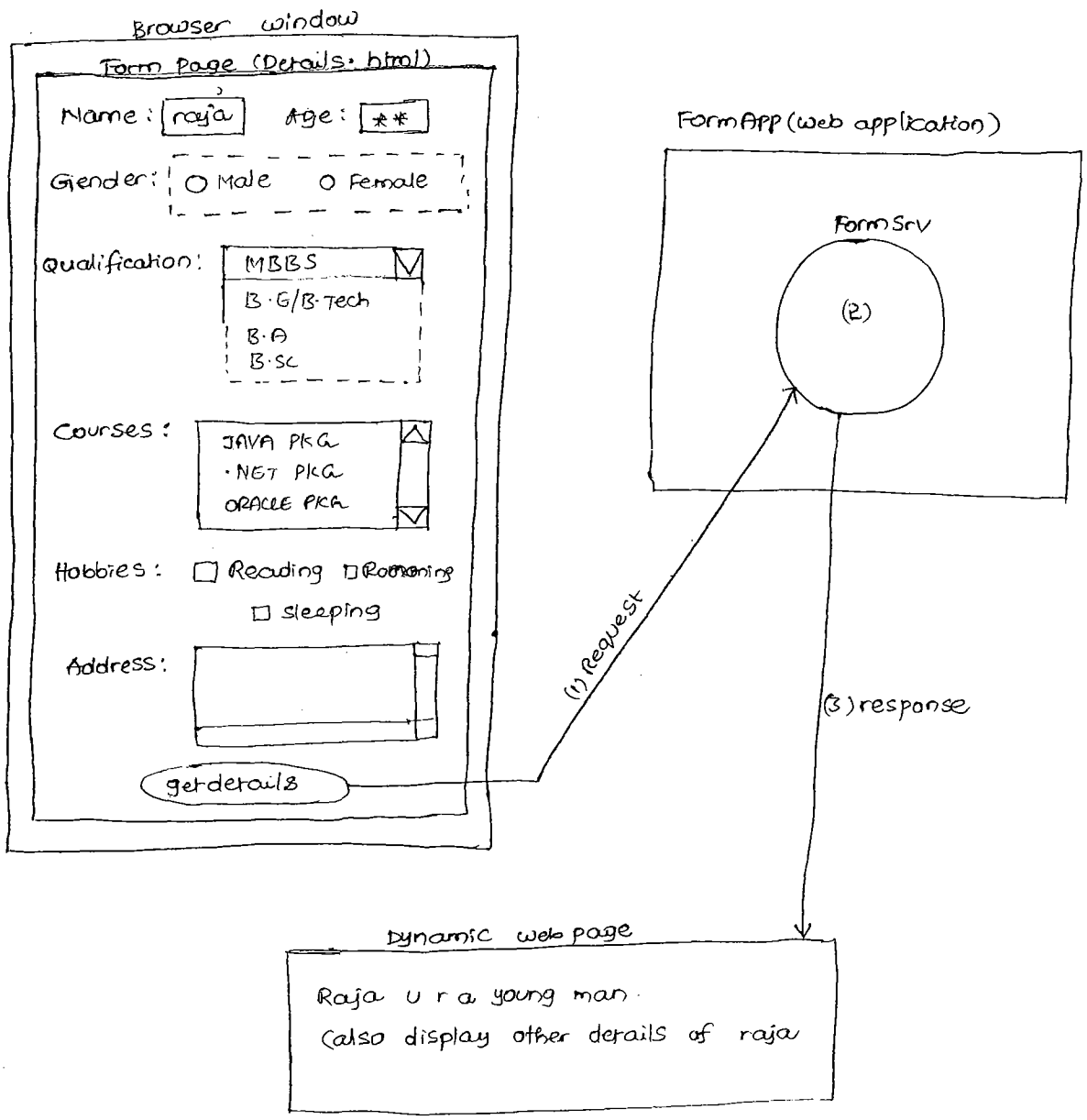
gives GlassFish 2.x as Built-in Server

open source s/w

to download s/w: www.netbeans.org

for docs: www.netbeans.org

* It is always recommanded to design the form components of form page as HTML table rows and column values. This allows us to align the form page easily.



Procedure to develop and execute above application by using NetBeans IDE

Step(1): create java web project having name FormApp.

File menu → new project → java web → web application → next →

Project Name : FormApp

→ next → server : GlassFish 2.1 → next → Finish

Step(2): Add form page to the project (Details.html)

Right click on webpages folder → new → HTML → HTML File Name : details

→ Finish

```
<form action="furl" method="get">
            ↓
        url pattern of FormSrv servlet program
  <table border="1">
      <tr>
        <td> Name: </td>
        <td> <input type="text" name="pname" ></td>
      </tr>
      <tr>
        <td> Age: </td>
        <td> <input type="text" name="page" > </td>
      </tr>
      <tr>
        <td> Gender: </td>
        <td>
            <input type="radio" name="g" value="m" check
            <input type="radio" name="g" value="F"
        </td>
      </tr>
      <tr>
        <td> Qualification </td>
        <td>
            <select name="qlfy">
                <option value="java"> JAVA PKG </option>
                <option value="Net"> MBBS </option>
                <option value="arts"> B.A </option>
                <option value="Science"> B.sc </option>
            </select>
        </td>
      </tr>
      <tr>
        <td> Courses </td>
        <td>
            <select name="crs" multiple>
                <option value="Java"> JAVA PKG </option>
                <option value="Net"> .NET PKG </option>
                <option value="oracle"> oracle PKG </option>
        </td>
      </tr>
```

```
<tr>
    <td> Hobbies </td>
    <td>
        <input type = " checkbox"  name = "ch1"  value = "read"  checked />
                                                    reading    
        <input type = "checkbox"  name = "ch1"  value = "roam" />  Roaming
                                                       
        <input type = "checkbox"  name = "ch1"  value = "sleep" />  sleeping
    </td>
</tr>
<tr>
    <td> Address </td>
    <td>
        <text area  name = address"  rows = "4"  cols = "20>
        enter address
        </textarea
    </td>
</tr>
<tr>
    <td  colspan = "2" > < input type = " submit"  value = "getdetails" /> </td>
</tr>
```

merges two cells into single cell

```
</table>
</form>
```

Step (3): Add FormSrv servlet to source packages folder of project.

Right click on Source packages folder → new → servlet →

class name : [FormSrv] → next →

url pattern : [furl] → Finish

NOTE: The above IDE generated servlet program gives processRequest(--,--) method to programmer for keeping request processing logic and this method will be called internally from doGet(--,--), doPost(--,--) methods like approach (3) of developing our servlet program as flexible servlet program. (Refer approach (3))

Step (4): keep the following request processing logic in processRequest(--) method of above servlet program.

```
{
 //general settings
 PrintWriter pw = res.getWriter();
 res.setContentType("text/html");
 //read from form page
 String name = req.getParameter("pname");
 int age = Integer.parseInt(req.getParameter("page"));
 String gen = req.getParameter("g");
 String qlfy = req.getParameter("qlfy");
 String crs[] = req.getParameterValues("crs");
 String hb[] = req.getParameterValues("ch");
 String address = req.getParameter("taddress");
 if (gen.equals("F"))
   {
     if (age <= 5)
         pw.println(name + "ur baby girl");
     else if (age<=12)
         pw.println(name + "ur child girl");
     else if (age<=19)
         pw.println(name + "ur teenage girl");
     else if (age<=30)
         pw.println(name + "ur a young woman");
     else if (age<=45)
         pw.println(name + "ur middle age woman");
     else
         pw.println(name + "ur old lady");
   }
 else if (gen.equals("M"))
   {
     if (age <= 5)
         pw.println(name + "ur baby boy");
     ==========
   }
```

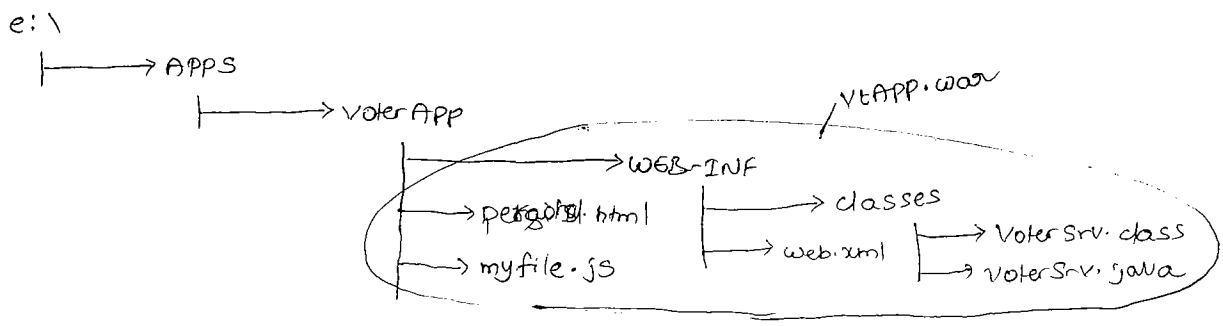
```
//close stream obj
pw.close();
} // print form data
pw.println("<br>name = "+name);
pw.println("<br>age = "+age);
pw.println("<br>Address = "+address);
pw.println("<br>Gender = "+gen);
pw.println("<br>Qualification = "+qlfy);
pw.println("<br>courses = ");
  for (int i=0; i< crs.length ; ++i)
    {
      pw.println (crs[i] + ".....");
    }
pw.println("<br>hobbies = ");
  for (int i=0; i< hb.length; ++i)
    {
      pw.println(hb[i] + "....");
    }
//close stream obj
  pw.close();
}
```

Step(4) : Run the project

Right click on FormApp project → Run

* we can prepare WAR file on deployment directory structure of web application. Each war file represents one web application.

Procedure to prepare war file



To prepare war file

e:\ Apps\ VoterApp> jar cf VtApp.war .

* Gives war file by combining everything of VoterApp folder.

* There are 3 ways to deploy web applications in servers.

(1) Hard deployment (Copy .war or directory of web application to a fixed installation folder of server software like Tomcat_home\webapps folder).

(2) Console Deployment (use admin console window for deployment)

(3) Tool based deployment (Deployment using tools like Ant, maven, IDE)

* (3) is recommanded process.

<u>Procedure to deploy webapplication in Tomcat server through console deployment process</u>

<u>Step(1)</u> : Prepare war file (VtApp.war) as shown above

<u>Step(2)</u>: Launch Tomcat web application manager window

start tomact server → launch home page → Tomcat manager → submit user name password

<u>Step(3)</u> : Deploy the war file

Tomcat web application manager window → Select WAR file to upload → browse → select the above VtApp.war file → Deploy

* Test the application by using the following request URL.

http://localhost:2020/<u>VtApp</u>/Personal.html

↓

war file name as context path

<u>NOTE</u>: In most of the servers when war file is deployed, its file name automatically becomes context path of the web application.

* To perform hard deployment of web application in tomcat server.

copy VoterApp folder (or) VtApp.war file to tomcat_home\webapps folder.

Weblogic

type : Application Server s/w

Vendor: BEA Systems (Oracle Corp)

Version: 10.3 (Comparable with JDK 6)

Commercial s/w

default port no : 7001

jar file that represents whole J2ee parts apis : weblogic.jar

to download software : www.oracle.com or www.commerce.bea.com

for docs : www.edocs.bea.com or www.oracle.com

allows to create domains. each domain act as one application server.

K If multiple projects of a company are using then weblogic s/w will be installed only once in a common machine but for multiple projects multiple domains will be created in that s/w.

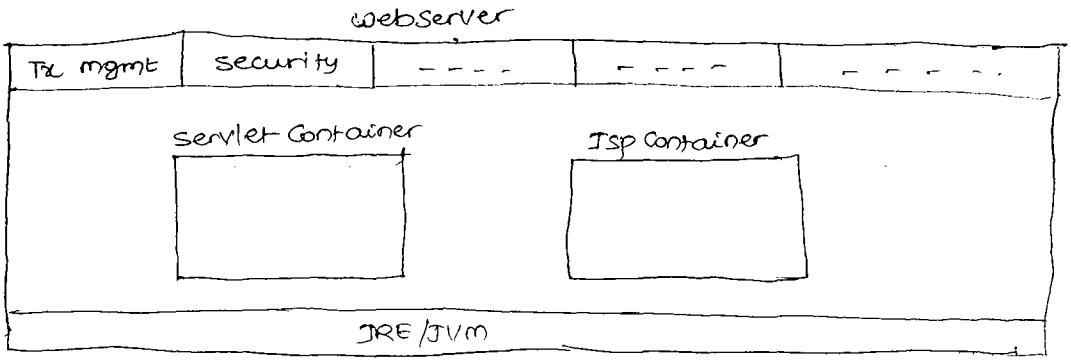
Procedure to create user define domain server in weblogic 10.3

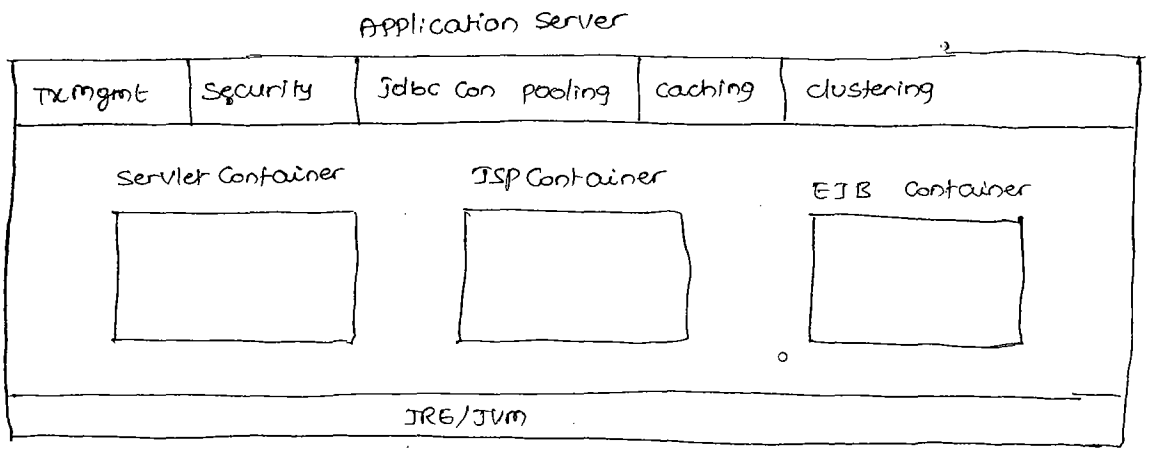
start → Programs → oracle weblogic → Quick start → Get started with weblogic server → create new weblogic domain → next → Gereate a domain . . . . → next →

Domain name : Adv.JavaBatchDomain

→ next → User name : [javaboss] password : [javaboss1] Conform User password : [javaboss1] → next → select Adminstration Server → next → Listen port : [7070] → next → create.

Application Server = web server + EJB Container + middle ware services





* EJB Container is required to manage and execute EJB component.
* war file ⟶ web application archive (represents web application)
* jar file ⟶ Java archive (represents java api / zip file)
* ear file ⟶ entriprise application archive (jar file + war file + jar file + war file + ...)
* rar file ⟶ resource adaptor archive (represents a J2ee application interacting with siebel / SAP s/w)

<u>What is the difference between web server and application server ?</u>

web server	application server
(1) Allows to deploy and execute web applications.	(1) Allows to deploy and execute web applications, EJB Components, Enterprise applications and resource adapter apps.
(2) Developed based on servlet, JSP api specifications.	(2) developed based on all J2EE api specifications. (servlets, JSP, EJB, JMS etc)
(3) Gives servlet Container, JSP Container.	(3) Gives both servlet Container, JSP Container and EJB Container.
(4) Doesn't allow to create domain	(4) Allows to create domains.
(5) Allows only Http protocol request	(5) Allows both Http and non Http (IIOP, T3 and etc) protocol based request.
(6) Gives minimum no. of middleware services.	(6) Gives more no. of middleware services.

(7) suitable for small scale and medium scale web applications.	(7) suitable for large scale web applications and for JEE applications.
(8) Recognizes .war file as application	(8) Recognizes .war, .ear, .jar, .rar files as applications.
9) Ex: JWS, Tomcat, Resin and etc.	(9) Ex: weblogic, web sphere, JBoss, Glass_Fish and etc.

Procedure to perform the console deployment of web application ~~Adv.bat~~ Adv.javaBatchDomain

Step(1): Prepare war file representing your web application like VtApp.war
↓
refer previous class

Step(2): Start Adv.javaBatchDomain server of web logic.

start → Programs → oracle weblogic → User projects → Adv.javaBatchDomain → start admin server for web logic.

Step(3): open adminstration Console of the above domain server.

open browser window → type below URL.

http://localhost:7070/Console

username: javaboss

Password: javaboss1

Step(4): Deploy the web application through admin Console.

Admin Console → environment → ~~Ser~~ Deployments → install → upload your files → Deployment archive → Browse and select VtApp.war file → next → next → next → next → finish → save

Step(5): Test the above deployed web application

Admin Console → deployments → launch VtApp.war file → select personal.html file related URL

k.10 undeploy the web application in web logic

Admin Console → deployments → ☑ VtApp → delete

* To perform hard deployment in Adv.JavaBatchDomain of weblogic server copy webapplications directory (VoterApp) or its war file (VtApp) to oracle weblogic_home\user_projects\domains\autodeploy folder.

* When war file is used for hard deployment (VtApp.war) use the following request URL to test the application.

http://localhost:7070/VtApp/Personal.html

* The web application that is deployed in weblogic server through hard deployment process can not be undeployed from admin console but can be undeployment through hard deployment process itself.

* When web application is hard deployed in web logic server through directory based hard deployment (copying VoterApp folder to autodeploy folder) then use following request URL to test the application.

http://localhost:7070/VoterApp/Personal.html

↓

Directory name as Context path

* In web logic server when you modify the source code of Servlet programs placed in deployed web application, just recompilation is enough to get the effect of modifications and there is no need of performing the reloading of web application.

* The web application and its web resource programs that are having relative URLs is called as WODA applications.

* Each server supplied seperate implementation classes implementing various interfaces of servlet, JSP api's but we never specify these implementation class names in our servlet programs to make our servlet programs as WODA programs.

Glass Fish

* type : Application Server S/W

Vendor: Sun microsystems

Version: 2.x (Compatable with jdk 1.6/1.5)

open source s/w

Allows to create domains. Default domain is domain1

jar file that represents all jee apis : javaee.jar

default port no: 8001 (for accessing web applications)

4848 (for accessing admin console)

* The GlassFish 2.x software that comes with netBeans 6.7.1 IDE can be used with or without IDE.

<u>Procedure to change the default http protocol service related portno in domain1 server of Glass Fish</u>

Go to GlassFish_home\AppServer\domains\domain1\config\domain.xml file and change port attribute value of first <http-listener> tag.

<u>Procedure to perform console deployment of web application in domain1 server of Glass Fish</u>

<u>Step(1):</u> Preparing war file representing web application like VtApp.war

<u>Step(2):</u> start domain1 server of Glass Fish.

Start → Programs → Sun Micro Systems → Application Server → Start default server

<u>Step(3):</u> open admin console of domain1 server

open browser window → type http://localhost:4848

username : admin

Password : admin admin

<u>Step(4):</u> Deploy the web application.

Admin Console → applications → web applications → deploy →

Type : web Application (.war)

location : Browse and select VtApp.war → ok

<u>Step(5):</u> Test the web application

open the browser window → type below URL

http://localhost:5151/VtApp/Personal.html

war file name as context path

* The Glassfish server supports only war file based hard deployment and it does not support directory based hard deployment.

Procedure to perform hard deployment of web application in GlassFish 2.x server.

Step(1): create war file representing the web application (like VtApp.war)

Step(2): Start domain1 server of GlassFish.

Start → programs → Sun microsystems → Application server → start default server.

Step(3): Deploy the web application

copy the above VtApp.war file to <GlassFish_home>\AppServer\Domains\domain1\autodeploy folder.

Step(4): Test the application.

open browser window → type below URL.

http://localhost:5151/VtApp/personal.html

war file name as context path.

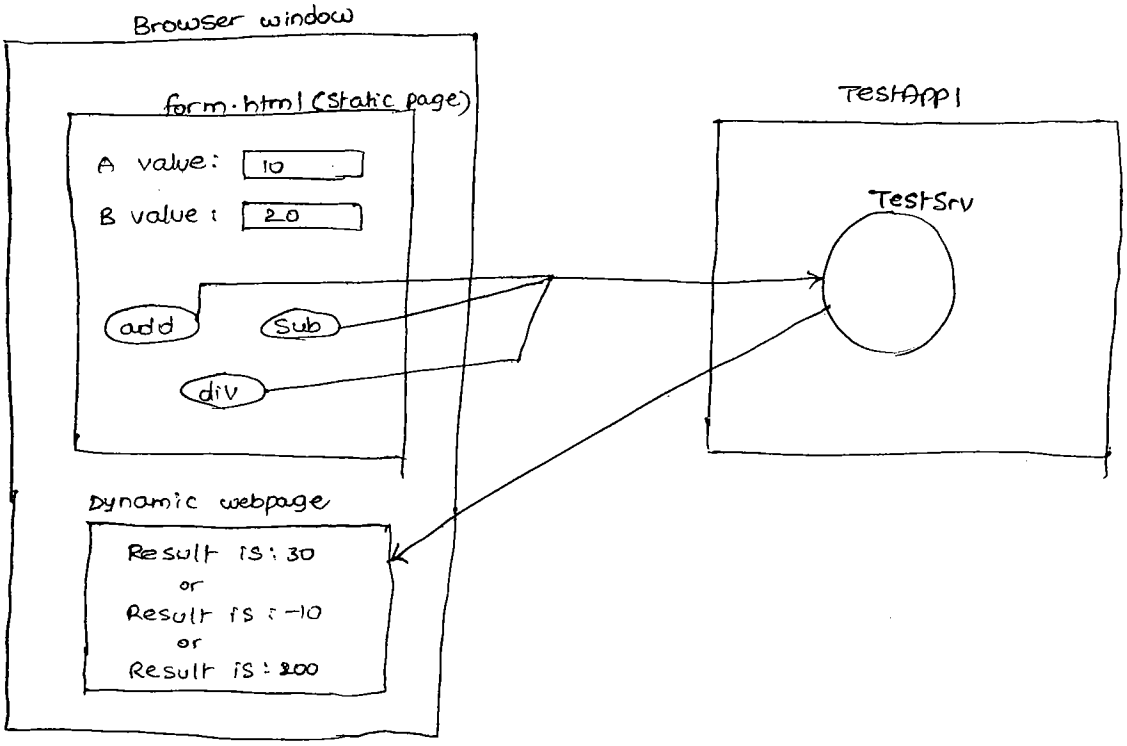
* When form page submit button is taken with logical name then the caption of the submit button will go to server as request parameter value. otherwise submit button caption never goes to server as request parameter value.

```
<form action="Vturl" method="get>
    ____ ____ ____
    ____ ____ ____
  <Input type="submit" name="S1" value="check">
<\form>
```

Here S1=check will go to server as request parameter name and value.

```
<form action="Vturl" method="get">
    ____ ____ ____
    ____ ____ ____
  <input type="submit" value="check">
<\form>
```

NOTE: Here related to submit button no value will go to server as request parameter value.



To handle the above form page related request processing give same name to all the three submit buttons with different captions and use that caption as criteria value in servlet program to differentiate request processing logic for each submit button.

form.html

```
<form action="url" method="get">
        A value: <input type="text" name="t1"><br>
        B value : <input type="text" name="t2"><br>
        <input type="submit" name="S1" value="add">
        <input type="submit" name="S2" value="sub">
        <input type="submit" name="S1" value="div">
</form>
```

TestSrv.java

```
public class TestSrv extends HttpServlet
{
    protected void service(HttpServletRequest request, HttpServletResponse response)
                                    throws ServletException, IOException
    {
```

```
response.setContentType("text/html");
printWriter pw = response.getWriter();
//read form data
 int a = Integer.parseInt(request.getParameter("t1"));
 int b = Integer.parseInt(request.getParameter("t2"));
 //read caption of submit button
 String cap = request.getparameter("s1");
 if(cap.equals("add"))
    {
       pw.println("sum is on result is:" +(a+b));
       pw.Println("add btn is clicked");
    }
  else if(cap.equals("sub"))
    {
       pw.println("result is : +(a-b));
       pw.Println("sub btn is clicked");
    }
  else
    {
      pw.println("result is : +(a/b));
      pw.Println("div btn is clicked");
    }
  //close stream object
   pw.close();
 }
}
```

web.xml

configure TestSrv servlet program with "/turl" pattern

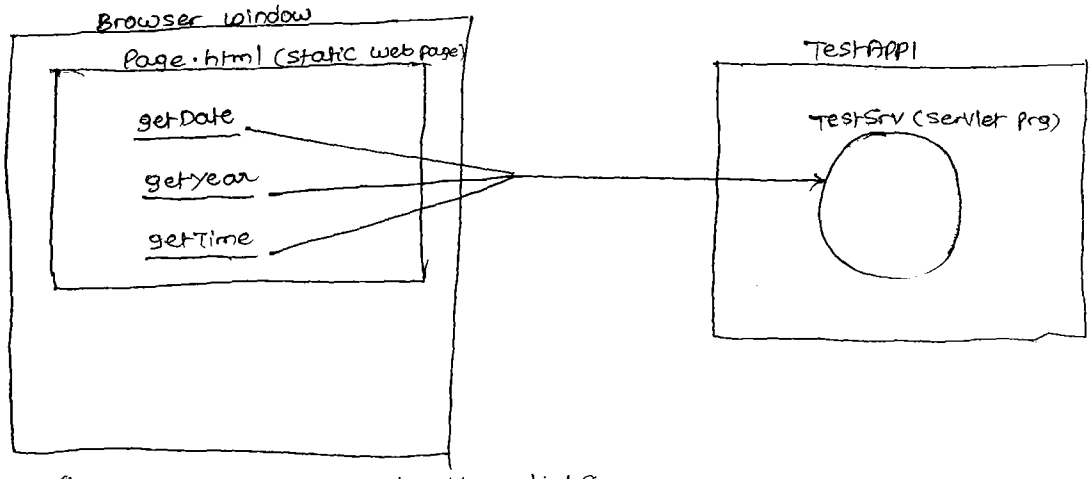


fig: Handling with multiple Hyper links

Page.html

```
<form action=//"
<a href = "turl? p1=link1" >getDate </a>
<br><br>
<a href="turl ? p1 = link2" > getyear </a>
<br><br>
<a href = "turl ? p1= link3" > getTime </a>
```

query string

NOTE: The query string that is appended to the URL of href attribute sends request parameter to server along with the request.

TestSrv.java

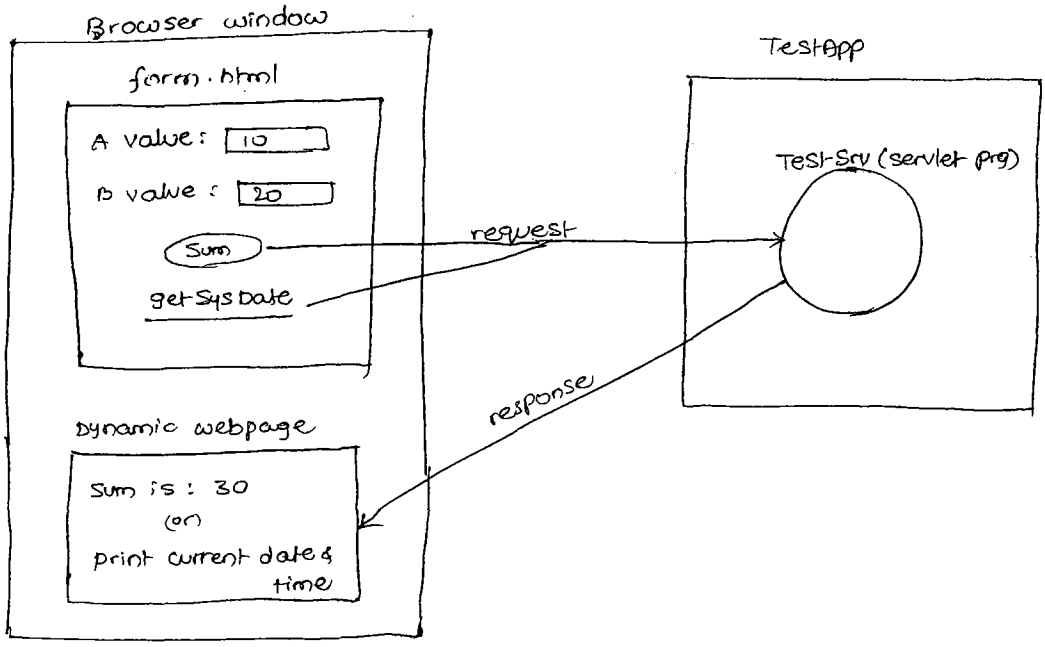
```
public class TestSrv extends HttpServlet
{
  protected void service(HttpServletRequest request, HttpServletResponse response)
                                    throws ServletException, IOException
  {
      //general settings.
      response.setContentType("text/html");
      printWriter pw = response.getwrite();
      //read p1 request parameter value
      string pval = request.getParameter("p1");
      //write rear processing logic based on hyperlink that is clicked
      calendar cl = new Calendar.getInstance(); //give sys date
      if (pval.equals("link1"))
        {
          pw.println("link1 is clicked");
          pw.println("current Day is:"+ cl.get(Calendar.DATE));
        }
      else if (pval.equals("link2"))
        {
          pw.println("link2 is clicked");
          pw.println("current year is:"+cl.get(calendar.YEAR));
        }
      else
        {
          pw.println("link3 is clicked");
          pw.println("current time is:" +cl.get(calendar.HOUR) +":"
                                    +cl.get(calendar.MINUTE));
        }
```

```
        //close stream object
        pw.close();
    }
}
```

web.xml

configure TestSrv servlet program with "/tur" url pattern.

Scenario (3) (Handling both Hyperlink, submit buttons based form page generated request)



* Give same name for submit button and hyperlink related additional request parameter having two different values. Use that value as criteria value in servlet program to differentiate the logics for submit button, hyperlinks.

* Calendar cl = Calendar.getInstance();

This getInstance() is static method of Calendar class returning one sub class object of Calendar class as return value. Calendar class is an abstract class.

* In the above statement cl is not the object of Calendar class it is one subclass object of Calendar class.

form.html

```
<form action="turl" method="get">
        ↳ URL pattern of TestSrv servlet program
    A value <input type="text" name="t1"> <br>
    B value <input type="text" name="t2"> <br>
    <input type="submit" name="s1" value="add">
</form>
<br> <br>
<a href="turl?s1=link1">getSysDate</a>
```

TestSrv.java

```
public class TestSrv extends HttpServlet
{
  Protected void service(HttpServletRequest request, HttpServletResponse response)
                                    throws ServletException, IOException.
  {
      //general settings.
      response.setContentType("text/html");
      PrintWriter pw = response.getWriter();
      //read s1 req parameter value.
      String pval = request.getParameter("s1");
      //write req. processing logic based on hyperlink or submit
      //button on that is clicked
      if (pval.equals("add")) //if add btn is clicked
        {
          //read form data
          int a = Integer.parseInt(request.getParameter("t1"));
          int b = Integer.parseInt(request.getParameter("t2"));
          pw.println("sum is :" +(a+b));
        }
      else if (pval.equals("link1")) //if hyperlink is clicked
        {
          Date d = new Date();
          pw.println("Date is:" + d);
        }
        //close stream object
        pw.close();
  }
}
```

web.xml

* same as previous application.
* A servlet program is identified through its URL pattern according to servlet specification given by sun micro systems there is a possibility of giving three types of URL patterns.

(1) Exact match
(2) Directory match
(3) Extension Math.

Exactmatch

* URL pattern begins with "/" symbol and should ~~not~~ contain * symbol.
* multiple words can be there in the URL pattern seperated with "/" symbol.

Ex: In web.xml

<url-pattern> /test1 </url-pattern>

request URLs from browser window to request the above servlet program

http://localhost:2020/DateApp/test1 (valid)

http://localhost:2020/DateApp/xy3/test1 (invalid)

/DateApp/test1/abc (invalid)

/DateApp/abc2 (invalid)

other examples

ex(1) <url-pattern> /test1/abc </url-pattern>

ex(2) <url-pattern> /abc/test1/xy3 </url-pattern>

ex(3) <url-pattern> /test1.cpp </url-pattern>

* The URL pattern provided for servlet programs can hide the servlet technology and the class names of servlet programs from the end users of the web sites. This gives little bit prevention for website from hackers and jackers.

Directory Match

* The URL pattern must begin with "/" symbol. and must end with * symbol. and can also have multiple words seperated with "/" symbol.

Ex(1) : /test1/xy31/*

request URLs from browser window to request the above servlet program

http://localhost:2020/DateApp/test1/xy31/abc (valid)

/DateApp/xy31/test1/abc (invalid)

/DateApp/test1/xy31/abc.c (valid)

/DateApp/x/y/test.do (invalid)

/DateApp/test1/xy31 (valid)

/DateApp/test1 (invalid)

other example directory match URL pattern

Ex(1): <url-pattern> /x/y/* </url-pattern>

Ex(2): <url-pattern> /abc/xy3/* </url-pattern

Ex(3): <url-pattern> /abc/* </url-pattern

Extension Match URL pattern

* URL pattern must begin with * symbol and must end with extension word/letter

Ex: <Url-pattern> *.do </url-pattern>

↓

extension word.

request urls from browser window to request the above servlet program

http://localhost:2020/DateApp/abc.do (valid)

/DateApp/abc/xy3/abc1.do (valid)

/DateApp/.do (valid)

/DateApp/abc.do/xy3 (in valid)

/DateApp/abc.do/xy3.do (valid)

/DateApp/abc.xy3 (invalid)

other example extension match URL patterns

Ex (1): <URL-pattern> *.abc </url-pattern>

Ex (2): <url-pattern> *.a </url-pattern>

* we can not prepare URL pattern of a servlet program by mixing up multiple styles because all servers are designed to just recognize only the above 3 styles of URL patterns.

<url-pattern> /xyz/abc/ *.abc </url-pattern>

↓

Invalid URL pattern formation

* If two servlets are configured with two different styles of URL patterns like exact match and extension match if matching same request URL given by browser window then server gives ~~important~~ priority to exactmatch url pattern based servlet.

Example scenario

In web.xml

DateSrv ⟶ with /test.do url pattern (exactmatch)

DateSrv1 ⟶ with /*.do url pattern (extensionmatch)

request from browser window

http://localhost:2020/DateApp/test.do

↓

This test.do matches both DateSrv, DateSrv1 servlet program but priority will be given to DateSrv program with URL pattern exactmatch.

MyEclipse = Eclipse + Built-plugins to work with advanced technologies

* A plugin is patch software or software application that can ~~ex~~ enhance the functionalities of existing s/w and s/w applications.

* In Java environment plugins come as jar files. In IDE softwares plug ins provides wizards to develop s/w technologies based applications

What is the difference between Eclipse IDE and MyEclipse IDE?

Eclipse	MyEclipse
1) open source	(1) Commercial
2) Provides environment to develop JSE module applications.	(2) Provides environment to develop JSE, JEE and frame software based applications.
3) Does not provide built-in plugin to work with advanced technologies we must ~~them~~ supply them manually.	(3) It provides built-in plugin to work with advanced technologies and also allows to add more plugins explicitly.
4) suitable for small scale companies.	(4) suitable for medium scale and large scale companies.

Basic information of myEclipse:

Type : IDE software for java environment.

Version : 8.x (Compatable with Jdk 1.6)

Vendor : Eclipse organization.

Commercial software

Gives tomcat as built-in server. But also allows the programmer to configure other external servers.

To download software : www.myeclipseide.com

for documents : www.myeclipseide.com

MyEclipse 8.x cheat code:

Subscriber : gocto.cn

Subscription code : ELR8ZC-855444-666653573987614Z

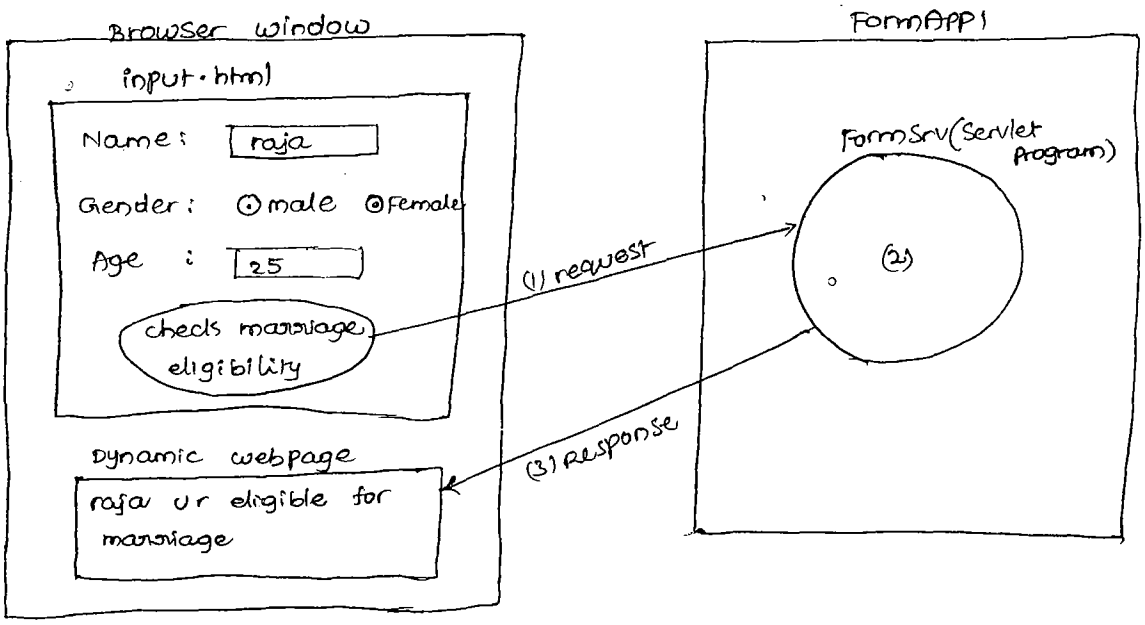
* Eclipse Geleelio, EG Eclipse are alternate IDEs for myEclipse which are given based on Eclipse IDE.

Procedure to develop below application by using myEclipse IDE

Step(1): Lanch myEclipse IDE by choosing one work space folder to save the projects.

work space : E:\myEclipse 8.x

Example application



Step (2): Submit the cheat code

MyEclipse menu → Subscription information →

Subscriber : _____

Subscription code : _____ → Finish

Step (3): create web project in myEclipse IDE.

File menu → New → web project →

project name : FormApp1

Context root URL : /FormApp1 → Finish

Step (4): Add form.html to web root folder of the project.

Expand project → Right click on web root folder → new → html → Filename.

form.html

```
<form method = "get" action = "furl" name = "f1">
Name : <input type = "text" name = "pname" /> <br />
Gender : <input type = "radio" name = "g1" value = "M" checked/ > male
         <input type = "radio" name = "g1" value = "F" /> female <br>
```

Age : input type="text" name="page"/>

: input type="submit" value="checked marriage eligibility"/>

</form

* Add servlet ogram to the scr folder of the project.

Right click on scr folder → new → servlet

Nam : formSrv → select doGet, doPost methods → next →

servl /JSP mapping URL : / → Finish

step(5) : In the generated FormSrv servlet keep the following logic and call that met d.

In FormSrv.ja

```
public id doGet (-,-) throws SE, IOE
{
  //ge al settings
  r. sponse. setContentType ("text/html");
  f otwriter pw= response. getwriter();
  //r d form data
  ring name=request. getParameter("pname");
  age = Integer. parseInt (request. getParameter("page"));
  ing gen=request. getParameter("gl");
  i (gen. equals ("m"))

    if (age >=21)
     {
      pw. println("mr"+name+" ur eligible to marriage");
     }
    else
     {
      pw. println ("mr"+name+" ur not eligible to marriage");
     }
  //if
  e if(gen. equals ("F"))

   if ( age >=18)
     {
      pw. println ("miss"+ name + " u r eligible to marriage");
     }
```

```
            else
              {
                pw.println("miss" + name + "u r not eligible to marriage");
              }
        } // else if.
    } // doGet(-,-)
    public void doPost(-,-) throws SE, IOE
    {
        doGet(-,-);
    }
```

Step(6): Configure Tomcat 6.x server with myEclipse IDE

window menu → Preferences → myEclipse → Servers → Tomcat →

Tomcat 6.x → Tomcat → Tomcat_home directory : D:\Tomcat 6.0

⊙Enable
o Disable

→ apply → ok.

Step(7): Start the tomcat Server from myEclipse IDE.

Go to servers icon in the tool bar → Tomcat 6.x → Start.

Step(8): Deploy the above project in Tomcat server.

Go to deploy icon of the tool bar → project FormApp1 →

add server Tomcat 6.x [V] → [Finish] → [ok]

Step(9): Test the application.

open browser window → type this URL.

http://localhost:2020/FormApp1/form.html ↵

Procedure to Configure Adv.JavaBatchDomain Server of weblogic 10.3 with MyEclipse IDE:

window menu → Preferences → myEclipse → Servers → web logic →

weblogic 10.x → ⊙ enable →

BEA Home directory : c:\oracle\middleware

Adminstration username: Javaboss

Adminstration password: Javaboss1

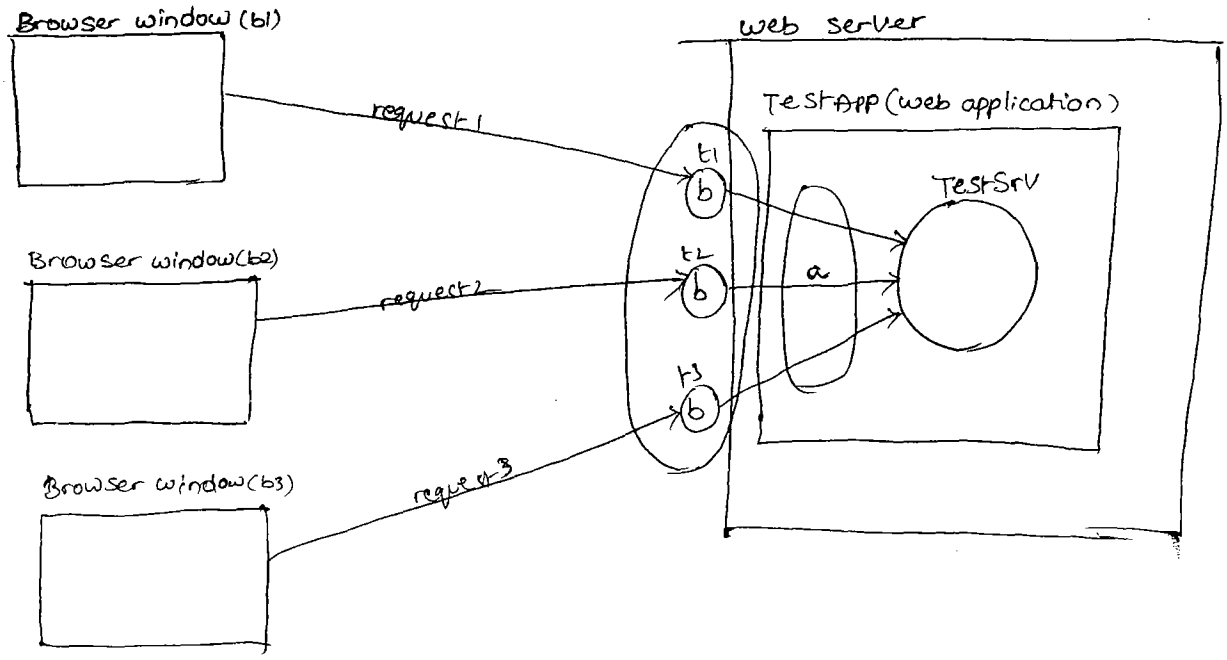
Execution domain root : e:\middleware\user_Projects\domains\
Adv.javaBatch Domain.

→ Apply → ok.

Procedure to configure the domain/ server of "GlassFish 2.x" with myEclipse IDE:

Window menu → preferences → myEclipse → servers → GlassFish → GlassFish 2.x → ⊙ Enable → Home_directory : D:\Sun\AppServer → Apply → ok.

* If multiple threads are acting on single variable (or) object simulteniously (or) concurrently then we say object/ variable is not thread safe. Because, there is a chance of data corruption in that situation.
* To make the above said variable/object as thread safe use synchronization concepts.
* Instance variables of our servlet program are not thread safe by default whereas the local variables declared in the service(-) method of servlet program are thread safe by default.



```
public class TestSrv extends HS
{
    int a;
    public void service(-,-) throws SE,IOE
    {int b;
      - -
      - -
    }
}
```

t1, t2, t3 are threads representing requests

* In the above diagram the multiple threads representing multiple requests of servlet program will act on single copy of instance variable "a" and every thread gets its own copy of local variable "b" so, we can say variable "a" is not thread safe and variable "b" is thread safe.
* By default our servlet class object and its instance variables are not thread safe. To make them as thread safe, use synchronization concept.
* To store results, inputs of servlets program in a perminant place like data base software (or) to gather input values of servlet program from perminant storage unit we need to make servlet program interacting with database software where by placing JDBC code in servlets program.
* Every JDBC code contains three important operations.
  (1) create JDBC connection object
  (2) use JDBC connection object to create other JDBC objects and to develop persistance logic
  (3) close JDBC connection object.
* There are three approaches to place JDBC code in our servlet program.

<u>Approach (1)</u>

(a) create jdbc connection objects in init()
(b) use jdbc connection object and other object in service(-,-) /doXxx(-,-) methods.

(c) close jdbc Connection object and other objects in destroy() method.

* In approach(1) the JDBC Connection object must be declared as instance variable so it is not thread safe (it is disadvantage). Programmer must use synchronization concepts to make the connection object as thread safe.

* In approach (1) all the requests given to servlet program will use single JDBC connection to interact with database software. This is improve the performance. (It is advantage).

Approach(2):

(a) Create JDBC connection object in service(-,-) / doXxx(-,-) methods

(b) Use JDBC connection object and other objects in service(-,-) / doXxx(-,-) methods.

(c) close JDBC connection object and other objects in service(-,-)/ destroy() method.

* Here JDBC connection object is local variable to service(-,-) method so it is thread safe object (it is advantage).

* Each request given to servlet program will establish one new connection with database software. This kills the performance (It is disadvantage).

Approach (3):

(a) Get JDBC Connection object from server managed JDBC connection pool being from service(-,-) method / doXxx(-,-) method.

(b) Use JDBC connection object and other JDBC objects in service(-,-)/ doXxx(-,-) methods.

(c) Return JDBC connection object back to JDBC connection pool from service(-,-) / doXxx(-,-) methods.

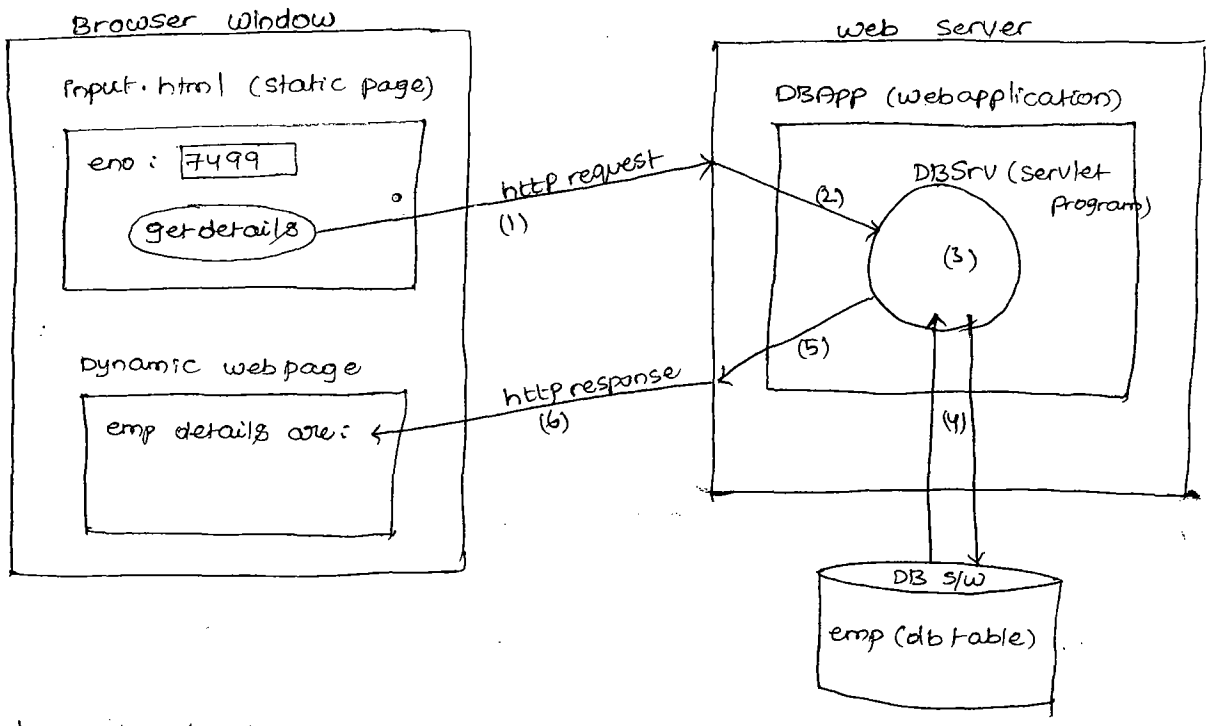
* Here Connection object is local variable of service(-,-) method. so it acts as thread safe object. (It is advantage).

* By using minimum no. of JDBC connection objects of connection pool we can make more servlet programs and all requests coming to

servlet programs interacting with database software. This gives better performance. (It is also advantage).

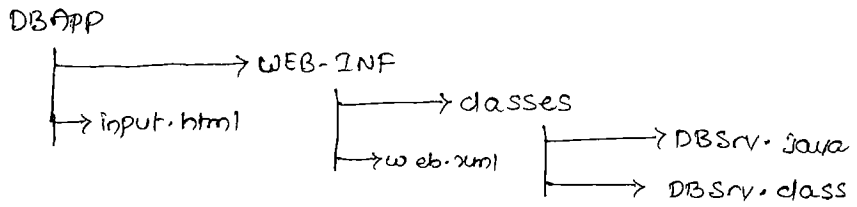
* Approach (3) is the best for servlet to database software communication.

Example application on servlet to database software communication

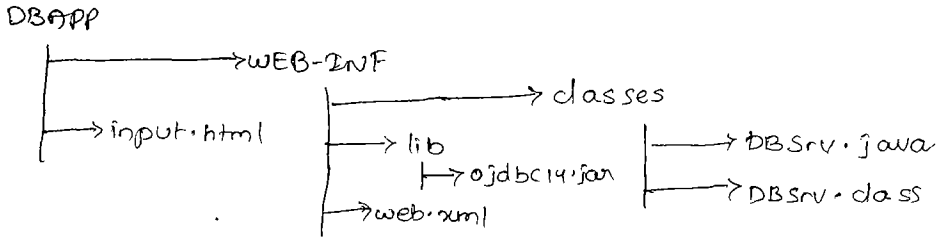


Deployment structure

when DBSrv program uses type 1 JDBC driver:



when DBSrv program uses type 4 (oracle thin) jdbc driver



* when stand-alone java application uses 3rd party API (other than JDK APIs) then the third party API related jar files must be placed in class path to make java compiler and JRE to recognize and use third party API.
* If java web application uses third party API in its web resource programs (like servlet, JSP programs) then third party API related jar file should be added to class path and should also be added to WEB-INF\lib folder of web application. Here jar files added to class path will be used by javac to recognize third party API during the compilation of servlet program. Similarly jar files added to WEB-INF\lib folder will be used by servlet container to recognize and use third party API during the execution of servlet program.

Ex: If servlet program uses oracle thin driver then keep ojdbc14.jar file in class path and WEB-INF\lib folder of web application as shown above.

* The jar files added to class-path are not visible in IDE softwares and in container softwares of servers.
* Stand alone java application will be compiled and executed from command prompt. whereas servlet program compilation takes place in servlet container of servlet server/Application server.
* If above web application is created in IDE software then the third party API related jar file must be placed in the libraries folder of the project.

Source code of above diagram based application

input.html

```
<form action="dburl" method="get">
    Employee no: <input type="text" name="tno"> <br>
                <input type="submit" value="getEmpDetails">
</form>
```

DBSrv.java (to interact with DB s/w using type 4 driver and approach (1))

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;
public class DBSrv extends HttpServlet
{
    Connection con=null;
    PreparedStatement ps=null;
    public void init()
    {
      try
      {
        // create jdbc connection object
         Class.forName ("oracle.jdbc.driver.OracleDriver");
         con = DriverManager.getConnection ("jdbc:oracle:thin:@localhost:1521:
                                                    satya", "scott", "tiger");
         ps = con.prepareStatement ("select ename, job, sal from Emp
                                                    where empno = ?");
      }// try
      catch (Exception e)
       {
        e.printStackTrace();
       }
    }//init()
    public void doGet (HttpServletRequest req, HttpServletResponse res) throws
                                                    ServletException, IOException
    {
     try
      {
       // read form data
        int no = Integer.parseInt (req.getParameter ("tno"));
       //write business logic
       //set parameter values to SQL query
        ps.setInt (1, no);
       // execute the query
        ResultSet rs = ps.executeQuery ();
       //process the resultset object
        String name = null, desg = null, bsal=null;
```

```
if (rs.next())
 {
   name = rs.getString(1);
   desg = rs.getString(2);
   bsal = rs.getString(3);
 }
//display emp details as web page content
 PrintWriter pw = res.getWriter();
  res.setContentType("text/html");
  pw.println("Emp details: <br>");
  pw.println("<br> Emp Name: "+name);
  pw.println("<br> Emp salary: "+bsal);
  pw.println("<br> Emp desg : "+desg);
 //close stream objects
  pw.close();
  rs.close();
}// try
catch(Exception e)
 {
   e.printStackTrace();
 }
}// doGet(-,-)
public void doPost(HttpServletRequest req, HttpServletResponse res) throws
                                        ServletException, IOException.
 {
  doGet(req, res);
 } //doPost(-,-)
public void destroy()
 {
  try {
       if (ps != null)
           ps.close
      }
  catch(Exception e)
      {
       e.printStackTrace();
      }
  try {  if (con != null)
           con.close();
      }
  catch(Exception e)
      {  e.printStackTrace();
      }
 }
}// class
```

web.xml

Configure DBSrv servlet program with "/dburl" as url-pattern.

Request URL to test application

http://localhost:2020/DBApp/input.html

* If multiple web applications of a server are utilizing same third party API (like oracle thin driver) then instead of keeping the 3rd party API related jar file (like ojdbc14.jar) in WEB-INF\lib folder of every application it is recommended to place only once in common library folder of server software instantiation.

Common library folder in Tomcat 6.0

<Tomcat_home>\lib\ folder

Common library folder in Tomcat 5.0/5.5

<Tomcat_home>\common\lib folder

<Tomcat_home>\server\lib folder

Common library folder in weblogic (Any version)

<BEA_home>\wlserver_<version>\common\lib folder

Common library folder in GlassFish 2.x (any version)

<GlassFish_home>\AppServer\lib folder

* when web resource program of web application uses the third party API class or interface then the servlet container looks for third party API class or interface in the following places and in the following order.

(i) In WEB-INF\classes folder itself (if not available then (ii))

(ii) In the jar files added to WEB-INF\lib folder (if not available then (iii))

(iii) In the jar files added to common library folder of underlaying server. (like Tomcat_home\lib folder) (if here also not comming then class not found exception is occured).

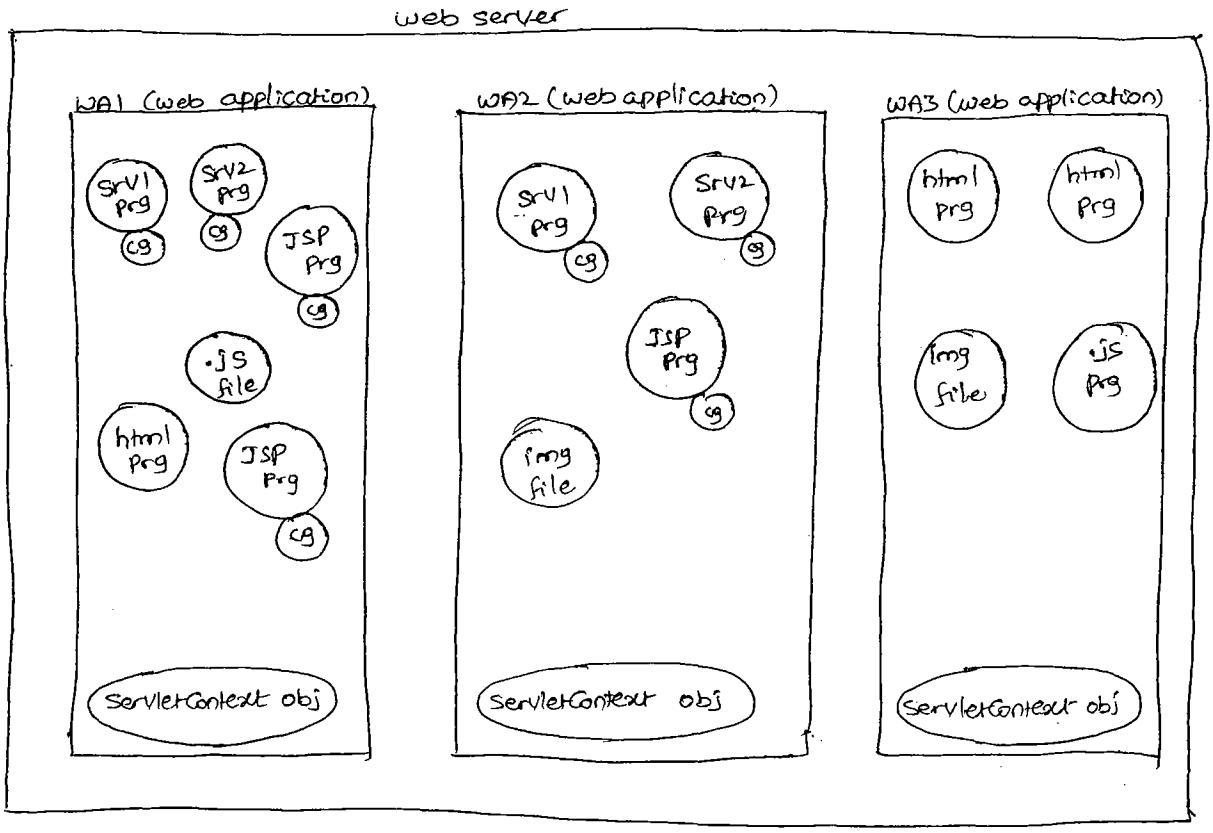
* Every server internally uses one Jdk software so then servlet program uses Type-1 jdbc driver to interact with database software then there is no need of adding any jar files in WEB-INF\lib folder and classpath.

What is the difference between ServletContext object and ServletConfig object?

(A) ServletConfig object (cg):

* It is one for our servlet program / JSP program.

* servlet Config object means it is the object of servlet Container supplied java class (implementing) implementing javax.servlet.ServletConfig interface.
* This object is useful to pass additional data to servlet and to read additional details from servlet.
* This object is useful to read servlet init parameter values from web.xml file of the web application.
* servlet Container creates the servletConfig object in the instantiation and initialization process of our servlet class object.
* servlet Container destroys our servlet class object in the destruction process of our servlet object.



cg - servlet Config object

img file — image file

js File - javaScript Code

Srv prg — servlet program

ServletContext object

* It is one per web application. So it is called as the global memory of web application.
* servlet Context object means it is the object of a java class (Container supplier) implementing javax.servlet.ServletContext interface.
* servlet Container creates this object either during deployment of the web application or during server start up.
* Servlet Container destroys this object automatically when web application is undeployed or reloaded or stopped or when server is stopped/re-started.
* using this object we can read global init parameters or Context parameters from the web.xml file of the web application.
* using this object we can know the details of underlaying server like server name, version and the servlet-api version supported by the server.
* using this object we can get Context path of the current web application and absolute path of the any web resource program in web application.
* The data kept is in servletContext object is visible and accessible in all servlet, JSP programs of web application.

In one web server 10 web applications are deployed. In that six web applications are there in running mode and four web applications are there in stopped mode. Can you tell me how many ServletContext objects are currently available in that web application server?

(A) Six.

There is a web application with 10 servlet programs. In that three servlet programs are already requested by clients and other 3 servlet programs are enabled with load-on-startup. can you tell me how many servletConfig objects are currently available in that web application?

(A) 3+3 = 6

NOTE: ServletContainer creates ServletConfig object for Servlet program only when the class of servlet program is instanciated (object creation).

* our servlet class object, request object, response object, ServletConfig object, ServletContext object can not be created by programmer manually. The servlet container creates all these objects but programmer can get access to these objects in servlet program.

. To get access to our servlet class object use "this" keyword.

* To get access to request, response objects use parameters of service(--) doXxx(--)

* To get access to ServletConfig obj

approach(1):

```
public class TestSrv extends HS/GS
{
   ServletConfig cg;
   public void ~~service~~ init (ServletConfig cg)
    {
       this.cg=cg;
       //use cg here
    }
   public void service(--,--) /doXxx(--,--)
    {
       use cg here
       ——  —
       —   —
    }
}
```

approach(2):

```
public class TestSrv extends HS/GS
{
   public void init()
   {
      ServletConfig cg= getServletConfig();
      use cg here
   }
   public void service(--,--) doXxx(--,--) method
    {
      ServletConfig cg = getServletConfig();
      — —
    }
}
```

↳ public method of predefined generic servlet class

NOTE: approach(2) is good.

NOTE: self class methods and super class public, protected methods can be called in the sub class with out object.

* To get access to servletContext object

approach(1):

```
public class TestSrv extends HS/GS
{
    public void init()
    {
        ServletConfig cg = getServletConfig();
        ServletContext sc = cg.getServletContext();
        // use sc here
    }
    public void service(--,--) / doXxx(--,--)
    {
        ServletConfig cg = getServletConfig();
        ServletContext sc = cg.getServletContext();
        //use sc here
        --  --  --
    }
}
```

→ method belongs to ServletConfig (I)

approach(2):

```
public class TestSrv extends HS/GS
{
    public void init()
    {
        ServletContext sc = getServletContext();
        // use sc here
    }
    public void service(--,--) /doXxx(--,--)
    {
        ServletContext sc = getServletContext();
        // use sc here
        --  --
    }
}
```

↳ public method of predefined Generic Servlet class

* Directly or indirectly we need servletConfig object to get access to ServletContext object.

NOTE: Approach(2) is good.

* getServletContext() method definition of predefined GenericServlet class internally uses servletConfig object to give ServletContext object.

* To make JDBC code of servlet program as flexible code to modify it is recommanded to gather the following four details of JDBC code from outside the servlet program. They are

(i) JDBC driver class name
(ii) JDBC URL
(iii) DB username
(iv) DB password.

* There are two ways to pass input values to servlet program from out side the servlet program.

(1) request parameters / form data.
→ end user / visitor of web application sends these values through browser window (from client side).
→ Since end user is non-technical person this data will be non-technical data like name, age, address of end user.
→ servlet program uses request object to read request parameter values.

(2) Servlet init parameters
→ Programmer passes this data to servlet program from web.xml (server side).
→ Since programmer is technical person, this data can be technical data like jdbc driver, URL and etc details.
→ servlet program uses servletConfig object to read init parameter values of web.xml.

* To make the JDBC code of servlet program as flexible code to modify it is recommanded to gather the above said JDBC details (driver class, URL, username, password) from web.xml file as servlet init parameter values.

* init parameters are specific to each servlet program and thus having programmer choice name and value.

* For revised DBApp application that uses servlet init parameter to make jdbc code of DBSrv servlet as flexible code to modify refer application (4) of the page nos 63-65.

* Send technical input values to servlet program from web.xml file as init parameter values. Send non-technical input values to servlet program as request parameter values / form data.

Different ways of reading servlet init parameter values

```
ServletConfig cg = getServletConfig();
```

approach(1)

```
String s1 = cg.getInitParameter("driver");
```

here we must know init parameter name to get the value

approach(2)

```
Enumeration e = cg.getInitParameterNames();
while (e.hasMoreElements())
  {
    String name = (String)e.nextElement();
    String val = cg.getInitParameter(name);
  }
```

//gives all init parameter names and values.

* when you configure multiple init parameters in web.xml file having same logical name then the last value will be effected as that init parameter value.

* using ServletConfig object we can gather the container created object name of our servlet class.

```
S.O.P("current servlet logical name/ instance name" + cg.getServletName());
```

gives <servlet-name> tag value from web.xml

* If multiple servlet / JSP programs of web application are looking to use same init parameters then instead of placing them in every servlet configuration it is recommanded to write them only once in web.xml as context parameters (or) global init parameters and use ServletContext

object in all servlet/JSP programs to read these context parameter values.

* If multiple servlet/JSP programs of a web application are talking with DB software by using same JDBC driver to make their JDBC code as flexible code instead of specifying JDBC driver details as init parameters in every servlet program configuration it is recommended to place JDBC driver details only once in web.xml as context parameters as shown below.

In web.xml of DBApp web application

```
<web-app>
    <context-param>
        <param-name> driver </param-name>
        <param-value> oracle.jdbc.driver.OracleDriver </param-value>
    </context-param>
    <c-p>
        <p-n> dburl </p-n>
        <p-v> jdbc:oracle:thin:@localhost:1521:ord </p-v>
    </c-p>
    <c-p>
        <p-n> dbuser </p-n>
        <p-v> scott </p-v>
    </c-p>
    <c-p>
        <p-n> dbpwd </p-n>
        <p-v> tiger </p-v>
    </c-p>
    <servlet>
        <servlet-name> db </servlet-name>
        <servlet-class> DBSrv </servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name> db </servlet-name>
        <url-pattern> /dburl </url-pattern>
    </servlet-mapping>
```

Context parameters of web application

```
<servlet>
    <servlet-name> t </servlet-name>
    <servlet-class> TestSrv </servlet-class>
</servlet>
<s-m>
    <s-n> t </s-n>
    <u-p> /testurl </u-p>
</s-m>
</web-app>
```

In init() method of DBSrv servlet program belonging to DBApp webapplication

```
public void init()
{
    try
    {
        //read init param values from web.xml
        ServletContext sc = getServletContext();
        String s1 = sc.getInitParameter("driver");
                                        → Context Parameter name
        String s2 = sc.getInitParameter("dburl");
        String s3 = sc.getInitParameter("dbuser");
        String s4 = sc.getInitParameter("dbpwd");
        ==
        ==
    }           //same as previous application code
    ==
}
```

NOTE: init parameters are specific to that servlet/JSP program for which they are configured. Context parameters are visible in all web resource programs (Java based) of web application.

Different ways of reading context parameter values

```
ServletContext sc = getServletContext();
```

approach (1) :

```
String s1 = sc.getInitParameter("driver");
```

here we must know context parameter name to get the value.

approach (2)

```
Enumeration e = sc.getInitParameterNames();
while (e.hasMoreElements())
  {
    String name = (String)e.nextElement();
    String val = sc.getInitParameter(name);
  }
```

//gives all context parameter names and values.

* we can use same name for both init parameter of servlet program and for context parameter. we can access both these parameter values in our servlet program using servletConfig, servletContext objects respectively.

* To gather misc info about underlaying server use servlet context object as shown below.

```
// gathering misc info using servletContext obj
pw.println("server info is:" + sc.getServerInfo()); // gives Apache Tomcat 6.0
pw.println("servlet api spec version impl by underlaying server:" +
                    sc.getMajorVersion() + "." + sc.getMinorVersion());
                                  gives 2.5
pw.println("Context path of current web application:" + sc.getContextPath());
                                                  gives /DBApp
pw.println("absolute path of input.html on server:" + sc.getRealPath(
                                                  "/input.html"));
                                                  gives Tomcat installation
                                                  path and input.html
```

* Tomcat 5.x → given based on servlet api 2.4 spec
* Tomcat 6.x → given based on servlet api 2.5 spec
* Tomcat 7.x → given based on servlet api 3.0 spec.

what is the difference between code placed in the constructor and code placed in the init() method of our servlet program?

→ since servlet container creates servletConfig object after ~~ex~~ constructor execution and before ~~executi~~ init() method execution in the instantiation

and initialization process of servlet program so ServletConfig object is not visible in constructor but visible and accessable in the init() method.

Since ServletContext object can not be accessed without ServletConfig object so the ServletContext object is not accessable in the constructor but ServletContext object is accessable in the init() method.

The code placed in the constructor of servlet program can not work with init parameters and context parameters where as the code placed in init() method can utilize those parameters so it is always recommanded to place the initialization logic only in init() method.

Jdbc Con object

1. Direct Jdbc Con obj

2. Pooled Jdbc Con obj

To create direct Jdbc Con obj

```
class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection(url, uname, pwd);
```

* The Jdbc connection object that is created by programmer manually is called as direct Jdbc Connection object.
* The Jdbc connection object that is collected from Jdbc Connection pool is called as pooled Jdbc connection object.

3 types of Jdbc con pools.

(1) Driver managed Jdbc con pool

(2) Third party s/w managed Jdbc Connection pool (like Apache DBCP, C3PO, proxool and etc).

(3) Server managed Jdbc connection pool (In web server, application server environment).

* Oracle thin driver gives one built-in Jdbc Connection pool in its ojdbc14.jar file. That Connection pool is called as DriverManagedConnection

pool. use: (1), (2) type Jdbc Connection pools in stand alone applications. use (3) type connection pool in those applications that are deployable in the server like web applications, EJB Components and etc..

* In Drivermanaged, third party s/w managed jdbc Connection pool environment we need to use type1/ type2/ type3 jdbc drivers.
* In server managed Jdbc connection pool environment we need to use type3 with type1/ type2/ type4 jdbc drivers.
* Each Jdbc Data source object represents one Jdbc Connection pool to get each connection object from connection pool we need to depend upon this data source object.
* Jdbc DataSource object means it is the object of a java class that implements javax.sql.DataSource interface.
* All Jdbc connection objects of connection pool represents connectivity with same database software. Jdbc Connection pool for oracle means all jdbc Connection objects in that Connection pool represents Connectivity with oracle database software.
* The advantage of Jdbc Connection pool is by using minimum no. of Jdbc Connection objects we can make more clients talking with DB software.

jndi registry s/w

nickname (or) alias name (or) jndi name

sathya	Student obj ref
apple	Customer class obj
india	Date class obj

* The registry s/w can maintain objects / object references having nick names or alias name to provide global visibility to them.
* To provide global visibility to jdbc DataSource object we place that object reference in registry software..
* Java applications use jndi api to interact with registry s/ws.

* The process of getting object/object reference from registry based on its nickname or alias name is called as <u>look-up</u> operation

* jndi api means working with javax.naming and its sub packages.

* jndi registry s/w are

  Rmi registry

  cos registry

  weblogic registry

  Jnp registry and etc...

* Every application server software comes with one built-in registry s/w.

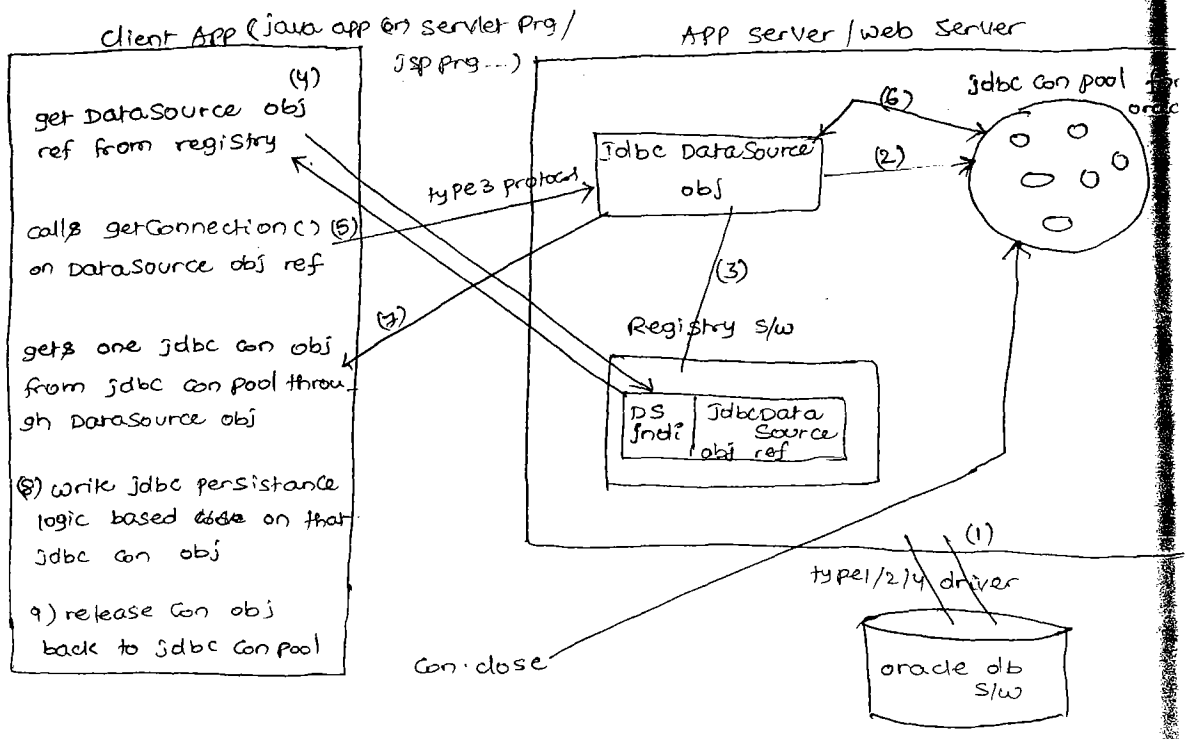
  Jboss ——> Jnp registry

  weblogic ——> weblogic registry

  websphere ——> cos registry

* The connection object of jdbc application represents connectivity between java application and database software. Similarly the initial context object of jndi programming represents connectivity between java application and registry software.

<u>Understanding the server managed jdbc connection pool environment</u>



* In server managed Jdbc Connection pool environment type3 with type4 jdbc drivers will be utilized. Here type4 will be utilized as a driver to interact with database software and to create Connection objects in the Connection pool where as type3 will be used as a protocol by client to get one jdbc Connection object from jdbc Connection pool.

with respect to diagram

(1) T.L or P.L makes application server to interact with DB s/w using type1 or type2 or type4 drivers and creates jdbc connection pool having jdbc Connection objects.

(2), (3) → T.L (or) P.L creates datasource object representing the above jdbc Connection pool and also keeps that datasource object reference in registry software having nickname for global visibility.

(4) → client application gets Datasource object reference from registry software through look-up operation.

(5), (6), (7) → client application calls getConnection() on DataSource object reference this call uses type3 protocol to get one jdbc Connection object from Connection pool.

(8) → refer diagram

(9) → refer diagram.

NOTE: The released Connection object in jdbc connection pool becomes ready to give service to next request or next client.

Procedure to create jdbc Connection pool for mysql/oracle and jdbc DataSource in GlassFish 2.x server

step(1): place ojdbc14.jar file in (sun-home) \ appserver \ domains \ domain1 \ lib \ ext \ folder.

step(1): place mysql related connector j jdbc driver jar file (mysql-connector-java-3.0.8-stable-bin.jar file) in GlassFish-home \ Appserver \ domains \ domain1 \ lib \ ext folder.

Step(2): start GlassFish server and open its admin console (domain1 server)

Step(3): create jdbc connection pool for mysql.

Admin console → resources → jdbc → Connection pools → new →

Name : [mypool1]

Resource type : [javax.sql.DataSource]

Database vendor : [mysql]

→ next → initial and minimum pool size : [2] (initial capacity)

max pool size : [20]

Pool resize quantity : [2]

idle time out : [300]

☑ Database name : [db1]

☑ Password : [root]

☑ Port : [3306]

☑ Port number : [3306]

☑ Servername : [localhost]

☑ URL : jdbc:mysql:// localhost:3306/db1

☑ url : jdbc:mysql://localhost:3306/db1

☑ user : root

→ Finish → Launch mypool1 (click on mypool1) → Ping

Step(4): create jdbc Datasource representing the above jdbc connection pool

Admin console → resources → jdbc → jdbc resources → new →

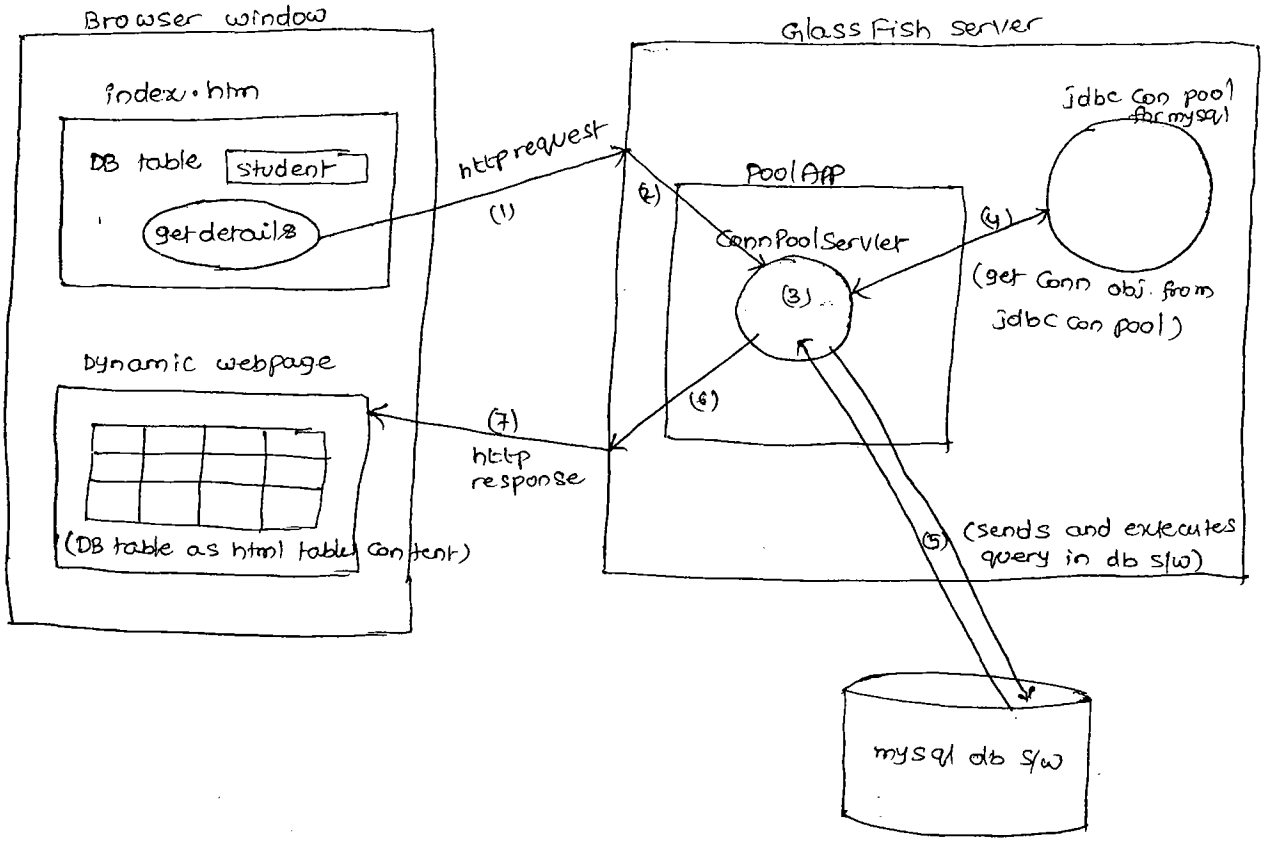
JNDI name : [Ds Jndi] (any name) (it becomes nickname or alias name once Datasource object is registered with registry s/w)

Pool name : [mypool1]

→ ok

NOTE: once ok button is clicked the above created DataSource object will be registered with registry software of GlassFish server automatically with JNDI name "DSJndi"

Example web application where servlet program uses the connection object of jdbc connection pool to interact with DB s/w



E:\Apps
- → PoolApp
  - → WEB-INF
    - → classes
      - → ConnPoolServlet.java
      - → ConnpoolServlet.class
    - → web.xml
  - → index.htm

* Preparing war file

  E:\Apps\PoolApp > jar cf PoolApp.war .

* Deploy the above war file in GlassFish 2.x server

  (copy PoolApp.war file to <GlassFish_home>\AppServer\domains\domain1\autodeploy folder)

Request URL to test the application

(http:// localhost: 5151/PoolApp/ index.html)

* For above diagram based example application refer application 5 of the page nos 65-67.

* Always develop web applications by keeping non-technical end-users in mind for that when exception is raised in web resource programs instead of displaying the exception related technical messages on browser window it is recommanded to display those messages as non-technical guiding messages for end user. For this we need to write some presentation logic in the catch blocks of servlet programs.

* Jndi api is part of java JSE module so we need not to add any jar files to classpath while working with Jndi api.

<u>Procedure to create JDBC Connection pool for oracle Data source in oracle-weblogic 10.3</u>

<u>Step(1)</u>: Start Adv.JavaBatchDomain Server of weblogic and open its admin Console.

start → programs → oracle weblogic → User_Projects → Adv.JavaBatch Domain → start Admin server for weblogic server domain.

<u>step(2)</u>: open browser window the following URL.

http://localhost:7001/Console ↵

user name: javaboss

password: javaboss

<u>Step(3)</u>: create JDBC Data source pointing to the JDBC Connection pool for oracle.

Admin console → services → JDBC → Datasources → New →

Name: mydsl (logical name)

JNDI Name: DSJndi (Required for client while performing lookup operation)

Database type: Oracle

Database driver: "Oracle's Driver (thin) for service connections"

→ next → next

Database name: Satya

Host name: localhost

port no: 1521

Database username: scott

Database password: tiger

Conform password: tiger

→ Test Configuration → next → select admin server → finish

NOTE: when finish button is clicked the data source object that represents JDBC connection pool for oracle will be registered with registry software with Jndi name "DSJndi".

Step(4): specify the JDBC connection pool additional properties.

Admin Console → services → JDBC → Data sources → mydsl → Connection pool tab

→ initial capacity [1]

maximum capacity [30]

capacity increment [2] // specifies the no. of new JDBC connection objects that should be created at a time if there a need of creating new JDBC connection objects in the connection pool.

→ save → Advanced → shrink frequency: [1000] seconds → [save]

* For utilizing the above server managed connection pool we can run the application ⑤ of the booklet. For that

Step(i): make sure that the Adv. JavaBatchDomain server is in running mode.

Step(ii): make sure that DSJndi is specified as the argument value of ic.lookup(-) method in line no. 612.

Step(iii): Prepare the deployment directory structure or war file representing the web application.

```
PoolApp
 |---> WEB-INF
 |      |---> classes
 |      |      |---> ConnPoolServlet.java
 |      |      |---> ConnPoolServlet.class
 |      |---> web.xml
 |---> index.html
```

Step(iv): Deploy the above PoolApp application in Adv. JavaBatchDomain server of weblogic.

copy the PoolApp folder to weblogic_home\middleware\User_Projects\Domains\Adv. JavaBatchDomain\Auto Deploy folder.

Step(V): Test the application

http://localhost:7001/poolApp/index.html

Jboss server

Type : Application server software

version : 5.x (Compatable with jdk 1.6)

vendor : Apache /RedHat

open source software

Default port no: 8080

Allows to create domains and also gives 5 built-in domains. They are

(1) default
(2) web
(3) standard
(4) all
(5) minimal

To download software: download s/w as zip file from www.apache.org website.

Installation process

* Extract Jboss-5.1.0.GA-Jdk6.zip file to a folder.
* Jar file that represents all JEE APIs : jboss-javaee.jar (<jboss_home>\client)

Procedure to change the port no. of "default" domain server of Jboss 5.x

Go to Jboss_Home\server\default\deploy\jbossweb.jar\server.xml file and modify the post attribute of first <Connector> tag.

To start jboss server ("default" domain)

use jboss_home\bin\run.batch file.

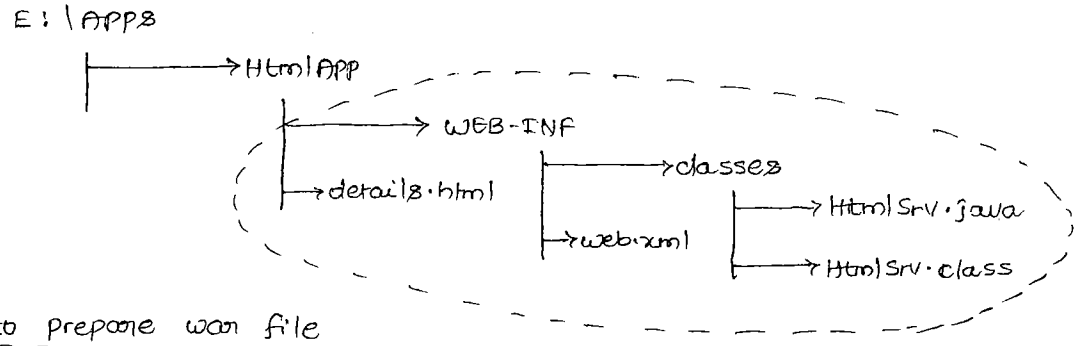
* In Jboss server there is no console deployment. But hard deployment is available (only war file based hard deployment is supported).

NOTE:

Jboss, GlassFish server directory based hard deployment. They only support .war file based hard deployment.

Procedure to deploy Java web application in "default" domain of Jboss 5.x server

Step(1): create .war file on the deployment directory structure of web application.



to prepare war file

E:\Apps\HtmlApp > jar cf HtmlApp.war .

Step(2): start "default" domain server of Jboss.

Step(3): Deploy the web application in Jboss server

copy the above "HtmlApp.war" file to Jboss-home\server\default\deploy folder.

Step(4): Test the web application

open the browser window and type this url

http://localhost:5151/htmlApp/details.html

Common library folder: The common library folder of "default" domain of Jboss is: Jboss-home\server\default\lib folder

Procedure to configure Jboss with myEclipse IDE:

window menu → preferences → myEclipse → servers →

Jboss 5.x → Enable Jboss_home directory: G:\Jboss 5.x soft\Jboss-5.1.0.GA

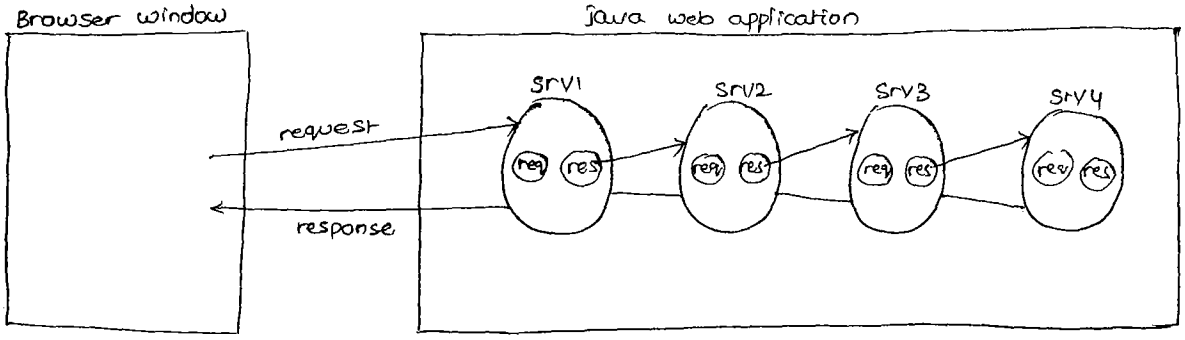
server name: default (domain name)

→ apply → ok.

31/10/11

Servlet chaining

* Taking request from a browser window and processing that request by using multiple servlets as a chain is called as servlet chaining. In servlet chaining we perform communication between servlet programs to process the request given by a client.

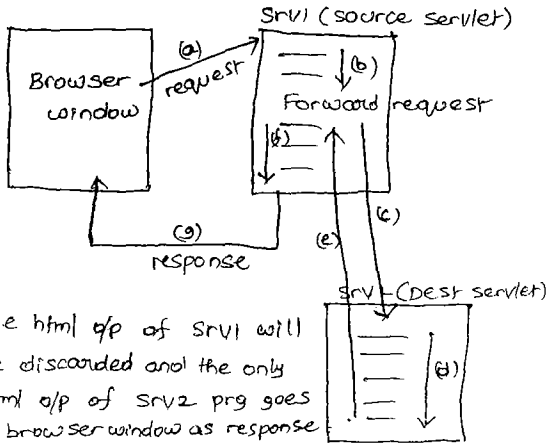


Servlet chaining based webapplication

* All servlet programs that participate in a servlet chaining will use same request and response objects because they process the same request that is given by client
* To perform servlet chaining we need RequestDispatcher object. Request_Dispatcher object means it is the object of a container supplied java class implementing javax.servlet.RequestDispatcher interface.

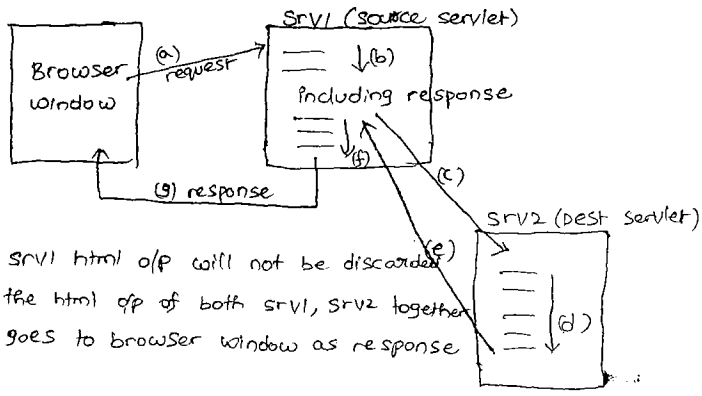
Servlet chaining

1. Forwarding Request mode of servlet chaining



The html o/p of Srv1 will be discarded and the only html o/p of Srv2 prg goes to browser window as response

2. including Response mode of servlet chaining.



Srv1 html o/p will not be discarded the html o/p of both Srv1, Srv2 together goes to browser window as response

* In any mode of servlet chaining all the servlet programs / web resource programs will use same request and response objects. If srv1, srv2, srv3, srv4 servlet programs are there in forwarding request mode of servlet chaining then the html o/p of srv1, srv2, srv3 programs will be discarded and only the o/p of srv4 servlet program goes to browser window.
* If srv1, srv2, srv3, srv4 servlet programs are there in including response mode of servlet chaining then the html o/p of all servlet programs together goes to browser window as response.
* The source servlet program uses RequestDispatcher object to perform servlet chaining with destination web resource program (like servlet program, JSP program html program and etc...)

There are 3 approaches to create RequestDispatcher object in source servlet program pointing to destination webresource program.

<u>Approach(1)</u> (by using req obj)

<u>Ex</u>: In srv1 source code

```
RequestDispatcher rd = req.getRequestDispatcher("/s2url");
                                                  ↳ optional
                                                  ↳ URL pattern of destination servlet program (srv2)
rd.forward(req, res);
      (or)
rd.include(req, res);
```

<u>Ex(2)</u>: In source srv1 program

```
RequestDispatcher rd = req.getRequestDispatcher("/abc.html");
                    (or)
RequestDispatcher rd = req.getRequestDispatcher("/abc.JSP");

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

<u>Approach (2)</u> <u>(by using servletContext obj)</u>

<u>Ex(1)</u>: In source srv1 program

```
ServletContext sc = getServletContext();
RequestDispatcher rd = sc.getRequestDispatcher("/s2url");
                                                  ↳ mandatory
                                                  ↳ URL pattern of dest srv program (srv2)
rd.forward(req, res)
      (or)
```

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

Ex(2): In source srv1 Program

```
ServletContext sc = getServletContext();
RequestDispatcher rd = sc.getRequestDispatcher("/abc.html");
                                                  ↳ mandatory
                    (or)
RequestDispatcher rd = sc.getRequestDispatcher("/abc.jsp");
rd.forward(req,res);
        (or)
rd.include(req,res);
```

Approach (3) (by using ServletContext obj)

Ex(1): In Src srv1 Program

```
ServletContext sc = getServletContext();
RequestDispatcher rd = sc.getNamedDispatcher("s2");
rd.forward(req,res);
        (or)
rd.include(req,res);
```

↳ logical name of destination servlet program (like srv2) given in web.xml file

Ex(2): In Src srv1 Program

```
RequestDispatcher rd = sc.getNamedDispatcher("j2");
rd.forward(req,res);
        (or)
rd.include(req,res);
```

↳ logical name of destination JSP Program like abc.jsp

NOTE(1): If source servlet program calls rd.forward(req,res) method then it performs forwarding request mode of servlet chaining with destination web resource program.

NOTE(2): In source servlet program calls rd.include(req,res) method then it performs including response mode of servlet chaining with destination web resource program.

* servlet Configuration in web.xml is mandatory so it contains logical name and URL pattern.
* JSP program configuration in web.xml file is optional so it may or may not contain logical name and URL pattern. we can not configure html

program in web.xml file.

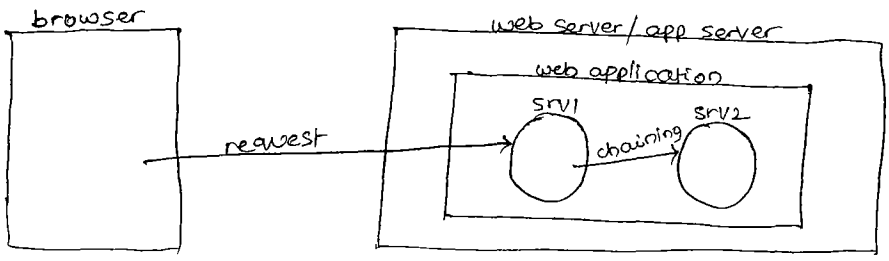
What is the difference between getRequestDispatcher( ) and getNamedDispatcher( ) methods ?

getRequestDispatcher( )	getNamedDispatcher( )
(1) Invokable on both request, ServletContext objects.	(1) Invokable only on ServletContext object.
(2) Expects URL pattern of destination servlet program or file names of destination JSP or html programs as argument value	(2) Expects logical name of the destination servlet/JSP program as argument value.
(3) This method generated RequestDispatcher object can point only the destination servlet, JSP program and html program.	(3) This method generated RequestDispatcher object can point only the destination servlet, JSP programs.

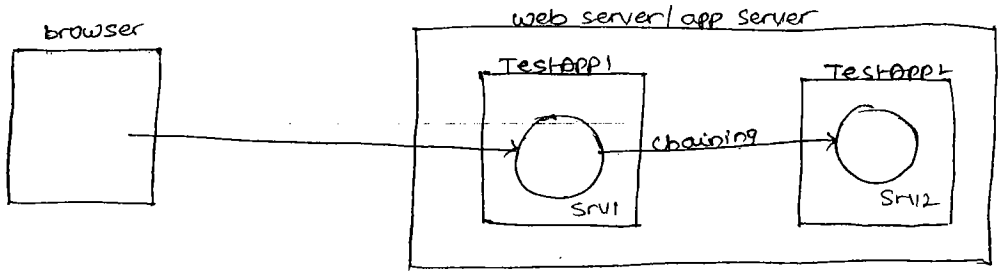
What is the difference between the RequestDispatcher object that is created based on RequestObject and ServletContext object ?

1) The Request object based RequestDispatcher object expects that source servlet program and destination web resource program in a same web application.

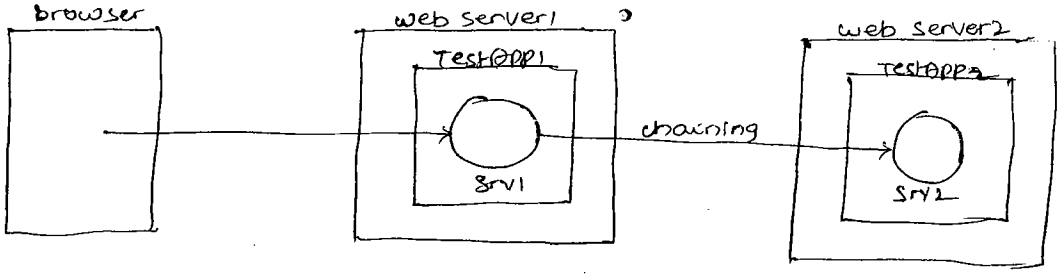
* The ServletContext object based RequestDispatcher object allows to keep the source servlet program and destination web resource program either in the same web application or in two different web applications of same server but they can not be there in two different web applications of two different servers.



Here srv1 can use req obj or ServletContext obj based RequestDispatcher obj.



Here Srv1 should use ServletContext obj based RequestDispatcher obj



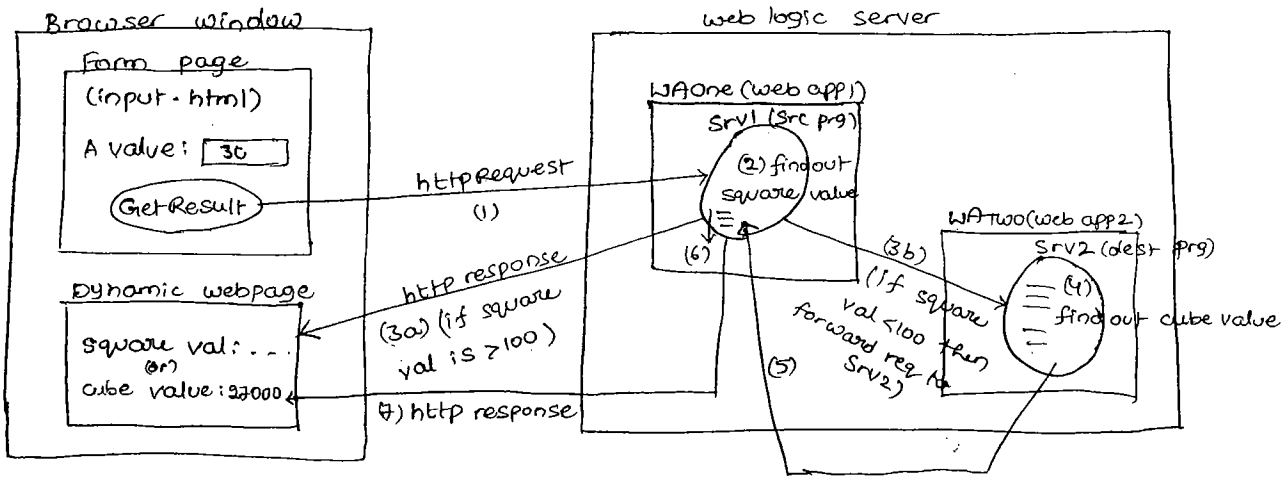
This kind of servlet chaining is not possible with RequestDispatcher obj

use send redirection Concept

3/11/11

* Servlet chaining is all about performing servlet-servlet Communication.

<u>performing servlet chaining between two servlet programs of two different web applications which reside in the same server</u>



<u>NOTE:</u> In the above diagram Srv1 program forwards the request to Srv2 only when the generated square value is less than 100 other wise the Srv1 directly sends response to browser window displaying that square value.

deployment directory structure

```
WAone
|---> WEB-INF
|---> input.html
|        |---> classes
|        |        |---> Srv1.java
|        |        |---> Srv1.class
|        |---> web.xml
```

```
WATWO
|---> WEB-INF
|        |---> classes
|        |        |---> Srv2.java
|        |        |---> Srv2.class
|        |---> web.xml
```

Deploy both these webapplications in web logic server (Adv. javaBatchDomain).

Copy WAone, WATWO web applications to <oracle weblogic_home>\user_projects\domains\Adv. javaBatchDomain\autodeploy folder.

* In Srv1 servlet program of the above WAOne webapplication we must create RequestDispatcher object based on ServletContext object.

Source Code (WAOne)

input.html

```
<form action="S1url" method="get">
        └→ url pattern of Srv1 servlet program
    A value: <input type="text" name="t1"><br>
    <input type="submit" value="getResult">
<\form>
```

Srv1.java

```
//Srv1.java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class Srv1 extends HttpServlet
{
    public void service(HttpServletRequest, HttpServletResponse res) throws
                                        ServletException, IOException
    {
        //general settings
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
```

```
//read form data
int no = Integer.parseInt (req.getParameter("t1"));
//find out square value
int resl= no * no;
if (resl>=100)    //display square value on browser window
 {
   pw.println("srv1: square val is :"+resl);
 }
else
 {
    //forward the request to srv2 program of WATwo web application
    //get Access to servlet Context obj of WAone web application
     ServletContext sc1= getServletContext();
    //get Access to servlet Context obj of WATwo web application
     ServletContext sc2 = sc1.getContext("WATwo");
    //Create RequestDispatcher obj pointing to srv2 program of
                                        WATwo web application.
     RequestDispatcher rd= sc2.getRequestDispatcher("/s2url");
     // forward the req to srv2 program from srv1 program
     rd.forward (req, res);
   }//else
 }// service(--,--)
}//class
```

web.xml

Configure srv1 program with /s1url as URL pattern.

Source code (WATwo)

srv2.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class srv2 extends HttpServlet
 {
   public void service (HttpServletRequest req, HttpServletResponse res) throws
                                        ServletException, IOException.
   {
     //general settings
     PrintWriter pw = res.getWriter();
```

```
        res.setContentType("text/html");
        //read form data
        int no= Integer.parseInt(req.getParameter("H"));
        //find out cube value
        int res2 = no * no * no;
        //display cube value
        pw.println("srv2: cube value is:"+res2);
    }//service(--)
}//class
```

web.xml

Configure srv2 program with the URL pattern /s2url

URL to test the application

http://localhost:7070/wfone/input.html

* In most of the servers the getContext() method of javax.servlet.ServletContext interface is not implemented so when that method is called in those servers we will get java.lang.NullPointerException. Due to this servlet chaining between two different servlet programs of two different web applications is not possible in most of the servers. (like weblogic 10.3, Tomcat all versions, JBOSS versions, Glass Fish all version) but the above said chaining can be achieved in weblogic 8.x and 9.x servers.

* Since getContext() method of servlet Context interface is not implemented properly in all servers so we can say the RequestDispatcher object based servlet chaining is good only when both source servlet program and destination web resource program resides in same web application. In remaining situations it is recommanded to use send redirection concept.

What is the difference between rd.include(), rd.forward() methods?

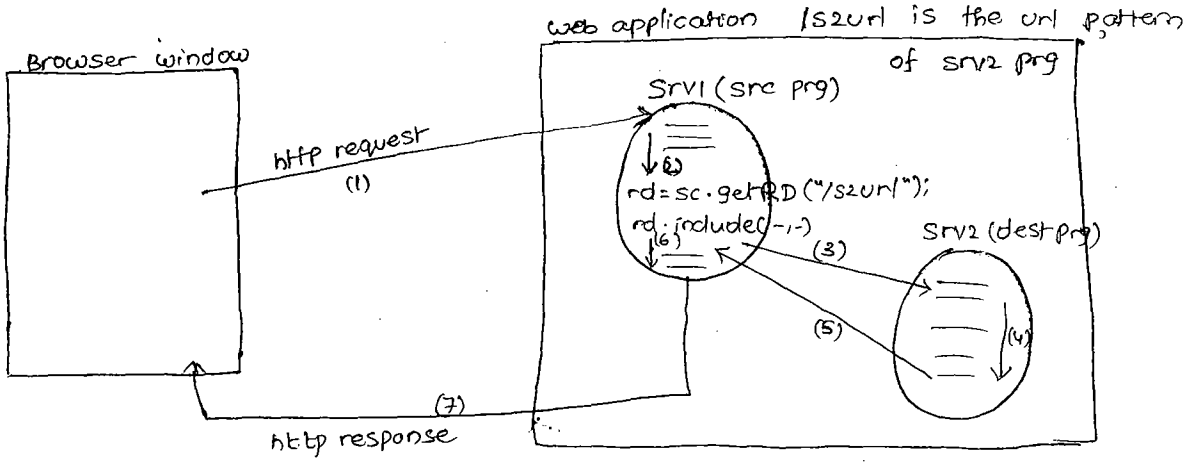
(A) Refer respective diagrams and key points of previous concepts.

* To know the current java version >java -version

* when multiple Jdk softwares are installed in your computer having different versions then that Jdk version will be activated whose <jdk_home>\bin directory is placed as first value of path environment variables.

2/11/11

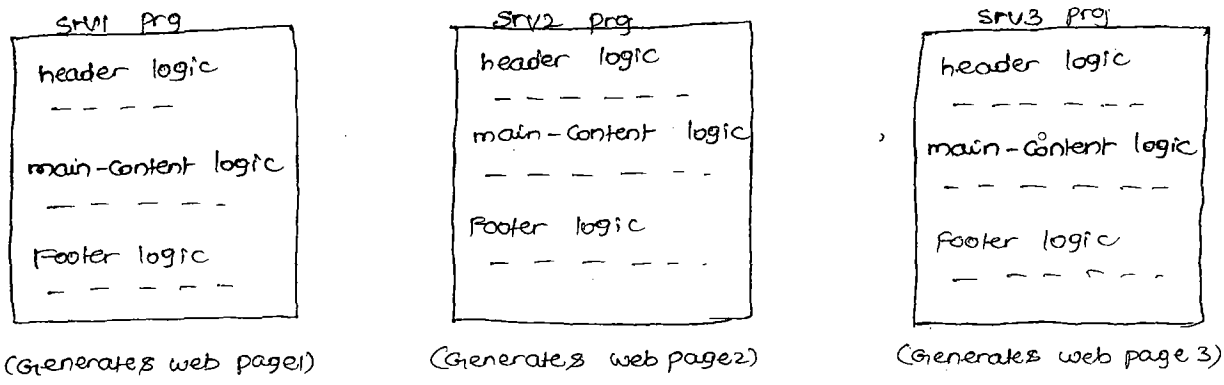
Understanding rd.include(-,-)



Here srv1 program (src prg), html o/p will not be discarded. moreover srv1, srv2 servlet programs html outputs together goes to browser window as response.

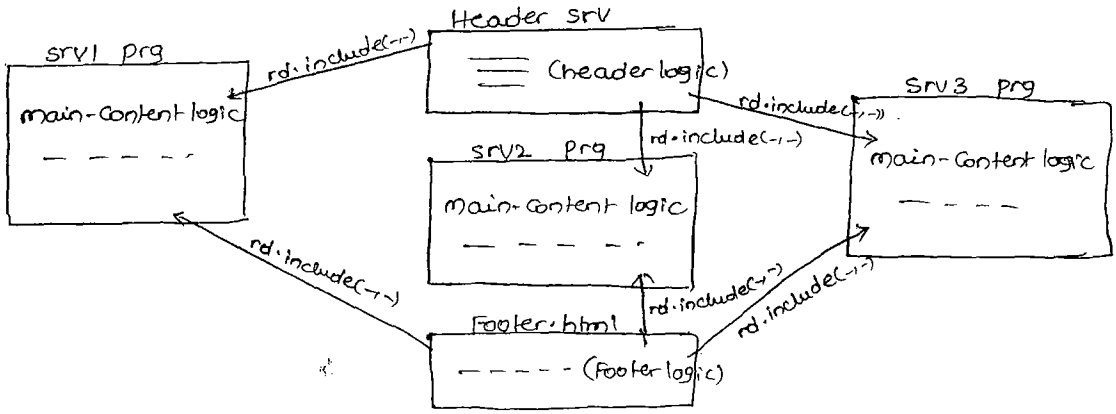
* rd.include(-,-) method performs including response mode of servlet chaining.
* Both srv1 and srv2 programs will use same request and response object so the request data coming to srv1 is visible and accessable in srv2. To pass additional data between srv1 and srv2 use request attributes.
* Browser window gets response from srv1 program by having the html o/p of both srv1 and srv2 programs
* srv2 html o/p can be included any where in the html o/p of srv1 by calling rd.include(-,-) method in the required place.
* srv1 and srv2 can be there in the same web application (or) can be there in two different web applications of same server.
* Srv2 can be a servlet program, JSP program (or) html program.
* In real time rd.include(-,-) is useful to include the o/p of reusable logics like header and footer logics in the output of (html output) main web resource programs.

Problem :



* All web pages of web site generally contains same header and footer contents but the main content will be changed in every web page.
* In the above diagram header and footer logics are not reusable because they are placed in all the three servlet programs eventhough they are same for all the three servlet programs.

solution :



* In solution diagram header and footer logics are seperated from main servlet programs of web application and they are placed in two seperate web resource programs but their html output is included in main servlet programs html output. This indicates header and footer logics are made as reusable logics
(using rd.include(-,-))

sample code based on solution diagram

Headersrv.java (having header logic)

```
public class HeaderSrv extends HS
{
    public void service(--) throws ServletException, IOException.
    {
        ======  PrintWriter pw = res.getWriter();
                res.SetContentType("text/html");
                pw.println("<font color=red size=6> S A T H Y A </font>");
                pw.println("<br><br><br><hr>");  //Don't place pw.close() in
    }                                            HeaderSrv.
} /headurl is the Headersrv servlet program
```

Footer.html (having footer logic)

```
<hr>
<br><br><br>
<b><i> &copy; All rights reserved 2101-11 </b></i>
```

MainServlet prgs

```
Public class TestSrv 1/2/3...n extends HS
{
    Public void service(--) throws SE, IOE
    {
        RequestDispatcher rd1=null, rd2=null;
        try
        {
            //include header content from HeaderSrv
            rd1 = req.getRequestDispatcher("/headerurl");
            rd1.include(req,res);
            ======  //logic to generate main content of web page
            //include footer content from Footer.html
            rd2=req.getRequestDispatcher("/footerurl");
            rd2.include(req.res);
        }
        catch(Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

* once the rd.forward(-,-) method executed in source servlet program then the effect of rd.include(-,-) method call will not be there in that servlet program that means rd.forward(-,-) method call discards both directed and included html outputs of source servlet program.

* For example application on both rd.forward(-,-), rd.include(-,-) method calls refer app (6) of the page nos 67-70.

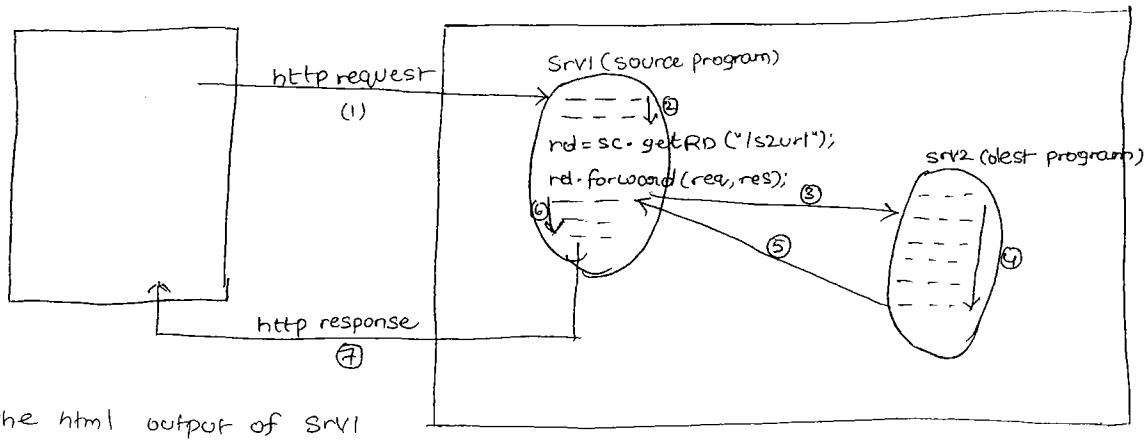
* ~~Depto~~ For example application on rd.forward() and rd.include() also refer app (8) of material.

* while working with rd.include(-,-) method don't commit the response in destination web source program by calling pw.close() method because it won't allow us to add further output generated by source servlet program and other destination programs. But we can do this work while working with rd.forward(-,-) method.

* Don't commit the response in source servlet program by calling pw.close() method before rd.forward(-,-) method call because this process throws java.lang.IllegalStateException (can't forward after response has been committed)

NOTE: pw.close() commit the response of web resource program that means it does not allow to add further content to response.

Understanding rd.forward(-,-) method



The html output of Srv1 will be discarded and only the html output of Srv2 will go to browser window as response.

Key points:

* rd.forward(--) method is there to perform forwarding request mode of servlet chaining.
* Both srv1 and srv2 servlet programs will use same request and response objects so the request data coming to srv1 (lik request parameters, headers) are visible and accessable in srv2 program.
* To pass additional data between srv1 and srv2 programs use "request" attributes.
* All the statements placed in srv1 program before and after rd.forward method will be executed but the entire html output of srv1 program will be discarded.
* Both srv1 and srv2 can be there either in the same webapplication (or) two different webapplications of same server.
* srv2 can be a servlet program (or) JSP program (or) html program.
* rd.forward(--) is very useful to configure Error Servlet for other main servlet programs of the web application.

What is Error servlet?

(A) The servlet program that executes only when exceptions are raised in other servlet program is called as Error servlet. This servlet is useful to display exception related messages as non-technical guiding messages on browser window when exceptions are raised in other servlet programs of web application.

Sample code to configure Error Servlet

ErrSrv.java

```
public class ErrSrv extends HS/GS
{
  public void service(--) throws SE, IOE
  {
    res.setContentType("text/html");
    PrintWriter pw = res.getWriter();
    Pw.println("<font color=red>Internal Problem </font>");
```

```
        pw.println("<br> Go to <a  href="input.html"> Home </a>");

    }// service(-,-)

  }// class.
// "/eurl" is the url pattern of ErrSrv program (in web.xml).
```

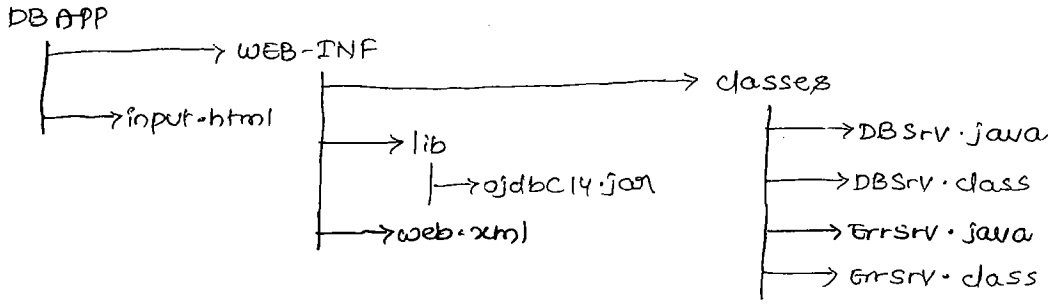
<u>DBSrv.java</u> (main servlet program of web application)

```
public class DBSrv extends HS
{
    public void init()
    {
      ---  -  -
      -  -  -
      -  - - -      Same as previous application
      - -  - -      (JDBC code)
      -  - - -
      - - - -
    }
    public void doGet(-,-) throws SE, IOE
    {
      try
      {
        ---  -  -
        -  - - -      Same as previous application
        -  -  -       (JDBC code)
        - -  - -
      }
      catch (Exception e)
      {
        try
        {
          RequestDispatcher rd=req.getRequestDispatcher("/eurl");
          rd.forward(req,res);
        }
        catch(Exception e1)
        {
          e1.printStackTrace();
        }
      }
    }
```

```
public void doPost (--) throws SE, IOE
  {
    doGet (req, res);
  }
public void destroy()
  {
    - - -
    - - -
  }
}
```

* In the above sample code when exception is raised in the doGet(--) method of DBSrv servlet Program the Error servlet (ErSrv) program will be executed automatically because of the rd.forward(--) method call.
* If servlet program contains rd.forward(--) method call then that servlet program html output will be discarded but their s.op("--") statements output will not be discarded.
* Always keep rd.forward(--) method call in source servlet program with condition statement show that the html output of that source servlet program will be discarded only when that condition is satisfied otherwise the html output will not be discarded.
* In real time projects rd.forward(--) useful to we configured Error servlet and to make web applications as more end user friendly web applications.

<u>Deployment directory structure of the above sample code based webapplication</u>

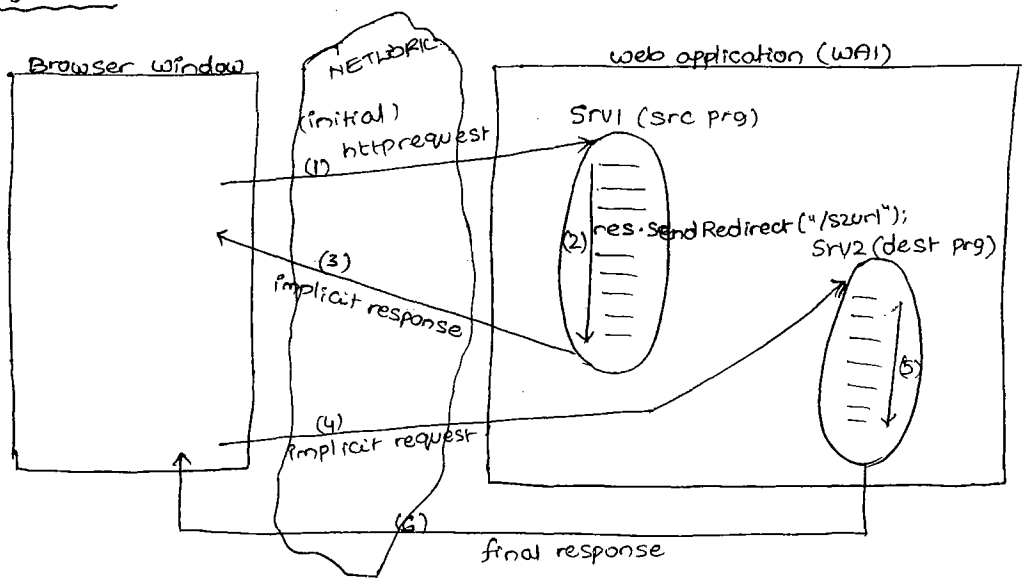


* Develop the above web application resources based on above sample code and application ④ of the booklet.

* The limitation with RequestDispatcher object based servlet chaining is it can not be used when source servlet program and destination web resource program are placed in two different web applications of same server (very few servers are supporting this) (or) in two different web applications of two different servers. To overcome this problem use send redirection concept of response.sendRedirect() method.

Understanding sendRedirection

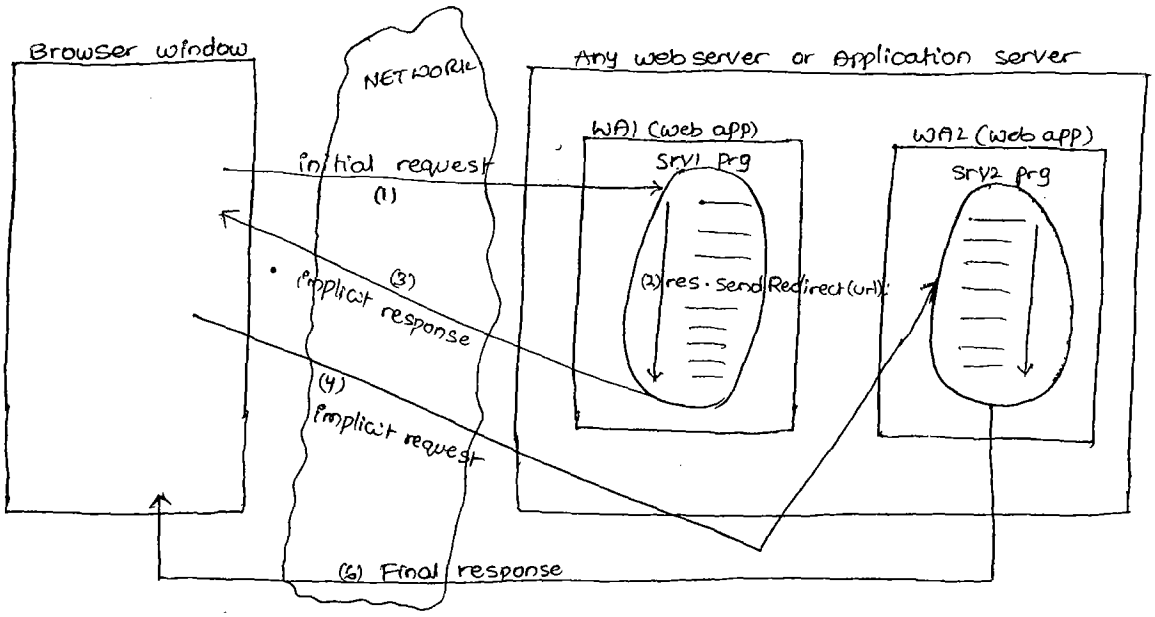
Diagram (1)



The html output of Srv1 (src) servlet program will discarded and only the html output of Srv2 (dest) servlet program will be displayed on browser window as response.

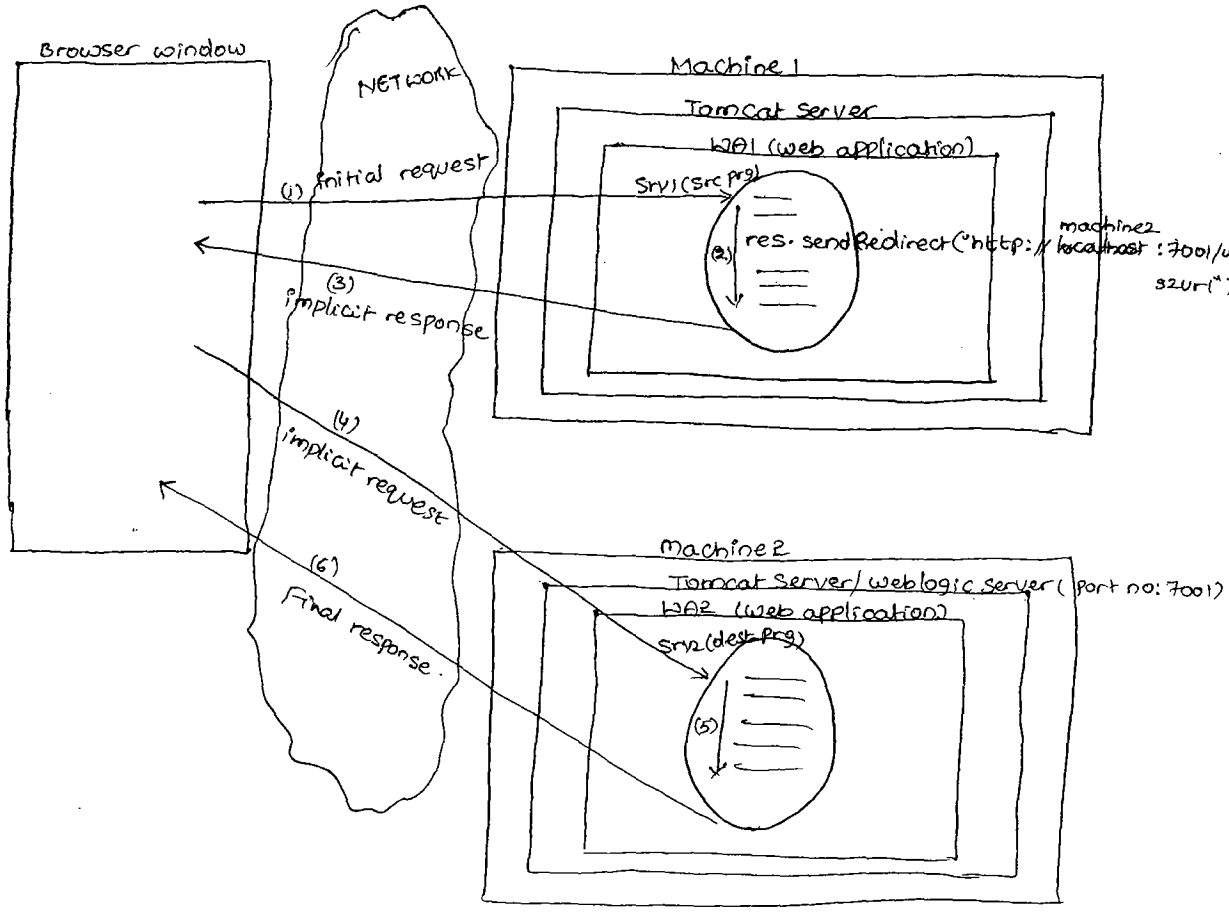
/s2url is the URL pattern of Srv2 servlet program.

Diagram(2)



url : http://localhost:2020/WA2/s2url (Request url of Srv2 program).

Diagram(3)



* Using send redirection concept we can not perform response inclusion but we can forward request from ~~one web~~ servlet program to destination web resource program.

* With respect to the diagrams

(1) Browser window gives initial request to srv1 servlet program.

(2) All the statements of srv1 servlet program executes including res.sendRedirect(-) method.

(3) srv1 program generates implicit response to browser window having the URL placed in sendRedirect(-) method as argument value. The response status code of this implicit response is 300-399 (indicates redirection).

(4) Browser window uses the URL coming from implicit response (because of 300-399 status code) and generates implicit request to srv2 program.

(5) All the statements of srv2 program executes.

(6) The output of srv1 program will be discarded and only the html output of srv2 program goes to browser window as final response.

Key points

* srv1 and srv2 will not use same request and response objects. They use different sets of req, res objects. So the request data coming to srv1 program is not visible and accessible in srv2 program.

* To send additional data from srv1 program srv2 program append query string to the URL of res.sendRedirect(-) method in srv1 program and read those values in srv2 program as request parameter values.

In srv1 prg

```
res.sendRedirect("/s2url?p1=val1&p2=val2");
        (or)
res.sendRedirect("http://localhost:2020/WA2/s2url?p1=val1&p2=val2");
        (or)
res.sendRedirect("http://machine2:7001/WA2/s2url?p1=val1&p2=val2");
```

In srv2 prg

```
String s1= req.getParameter("p1");
String s2= req.getParameter("p2");
```

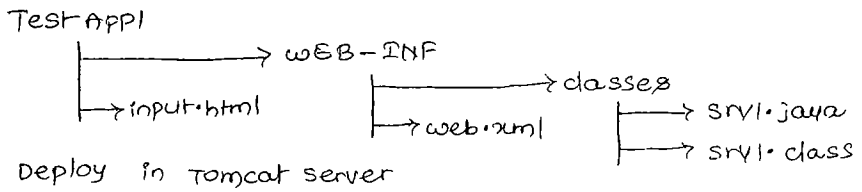
* Srv1 and Srv2 can be there in the same web application or can be there in two different web applications of same server or two different servers (these two servers can be there in the same machine or in two different machines).
* If Srv1 and Srv2 resides in the same web application we can pass relative path in sendRedirect (-) method otherwise we must pass absolute URL.
* Srv2 can be a servlet program or JSP program or html program or ASP program or ASP.net program or php program etc...
* The movement res.sendRedirect(-) method executes in Srv1 program the entire html output of Srv1 program will be discarded.
* In real time to pass the request of one website to another website without worrying about their technology environment we can use send Redirection.

Ex: IBM has acquired rational.com so the request given to rational.com will be redirected to certain web resource program of ibm.com.
Oracle Corporation has acquired sun microsystems so the request given to sun.com will be redirected to a webresource program of oracle.com.

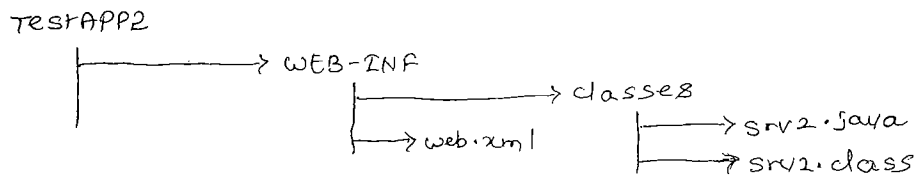
Example application on send Redirection

NOTE: In the below application Srv1 program is sending three values to Srv2 program by appending query string to the url of response.send Redirection() method. Those values are form data, sum result.

Directory structure



Deploy in Tomcat server

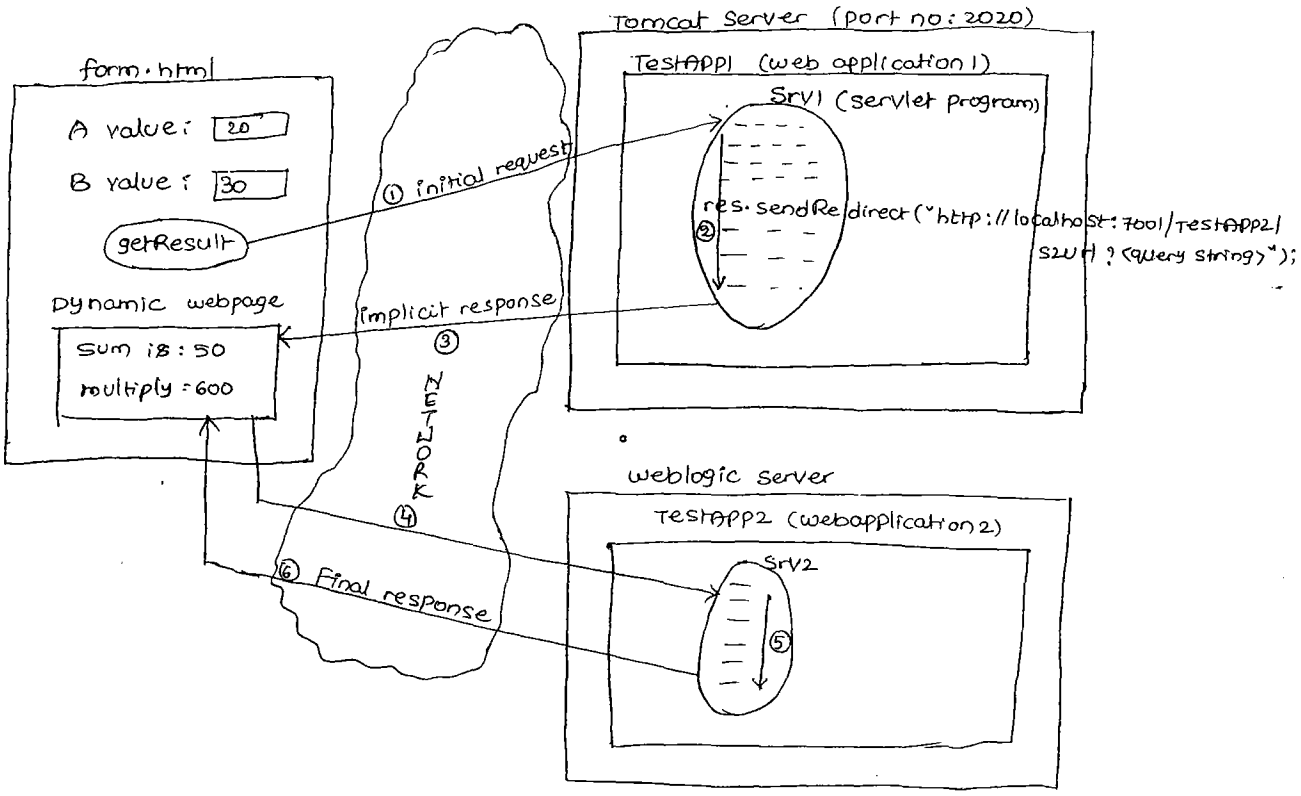


Deploy in Adv.JavaBatchDomain of weblogic server.

TestAPP1 (web application)

input.html

```
<form action="s1url">
    A value : <input type="text" name="t1"> <br>
    B value : <input type="text" name="t2"> <br>
              <input type="submit" value="getResult">
</form>
```



Srv1.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class Srv1 extends HttpServlet
{
    Public void service (-,-) throws SE, IOE
    {
        // general settings
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        //read form data
        int val1 = Integer.parseInt(req.getParameter("t1"));
```

```
int val2 = Integer.parseInt(req.getParameter("t2"));
//find sum value
    int sum = val1+val2;
pw.println("<b> srv1 : sum is </b>" + sum);
//redirect the request to srv2 program of TestApp2
s.o.p("srv1 : before res.sendRedirect(-)");
res.sendRedirect("http://localhost:7001/TestApp2/s2url ? p1= "+val1+" & p2= "+val2+" & p3= "+sum);
s.o.p("srv2 : after res.sendRedirect(-)");
//close stream obj
pw.close();
}// service(-,-)
}//class
```

query string having data to send data to srv2 program.

web.xml

Configure Srv1 servlet program with "/s1url" url-pattern.

TestApp2 source code (web application2 of different server)

Srv2.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class srv2 extends HttpServlet
{
  public void service(-,-) throws SE, IOE
  {
    //general services
    PrintWriter pw = res.getWriter();
    res.setContentType("text/html");
    //read request param values send srv1 program
    int val1 = Integer.parseInt(req.getParameter("p1"));
    int val2 = Integer.parseInt(req.getParameter("p2"));
    int sum = Integer.parseInt(req.getParameter("p3"));
    //find out multiply value
    int mul = val1 * val2;
    //display results
    pw.println("srv2 : sum is :" + sum);
    pw.Println("srv2 : multiply is :" + mul);
    s.o.p("form srv2 program");
    pw.close();
  }
}
```

web.xml

Configure Srv2 program with "/s2url" url-pattern.

Request URL to test the application

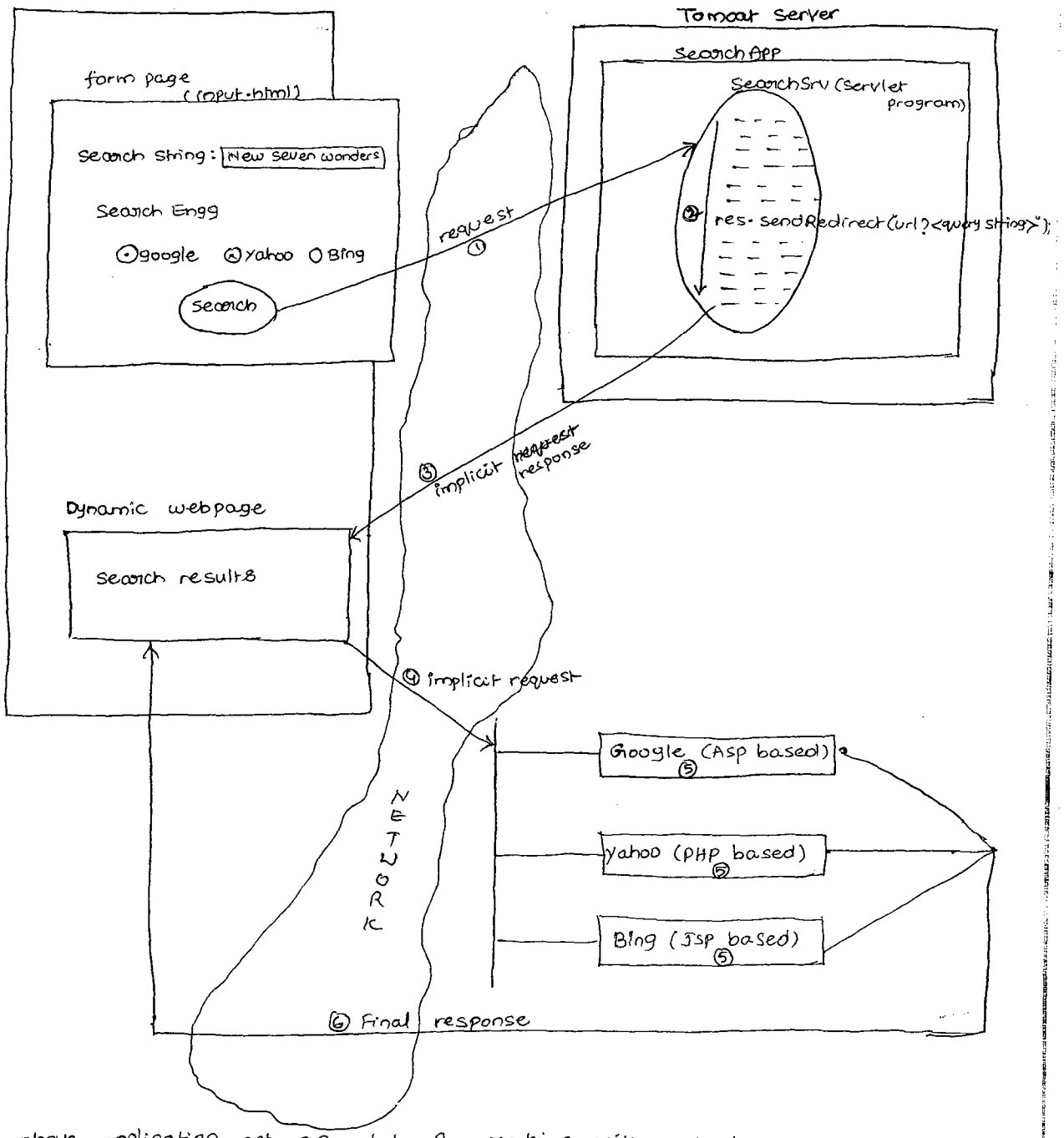
http://localhost:2020/TestApp1/form.html

what is the difference between rd.forward() and res.sendRedirect() ?

rd.forward(-,-)	res.sendRedirect(-,-)
* Perform forward mode of servlet chaining.	* Performs sendRedirection mode of communication.
* The source servlet program communicates with destination webresource program directly.	* The source servlet program communicates with destination servlet program by having network round trip with browser window.
* The source servlet program and destination web resource program uses same request and response objects so request data coming to source servlet program is visible and accessable in destination webresource program.	* The source servlet program and destination web resource program will not use same request and response object so the request data coming to source servlet program is not visible and accessable in destination program.
* Source servlet program can use request attributes to send additional data to destination program.	* Append query string to the url of response.sendRedirect() method to send additional data from source servlet program to destination program.
* Source servlet and destination programs can be there in same web application (any server) (or) can be there in two different web applications of same server (only in few servers).	* The source servlet program and destination program can be there in same web application (or) in two different web applications of same server (or) different servers.
* Destination program can be a servlet (or) JSP (or) html program.	* Destination program can be a servlet (or) JSP (or) html (or) ASP (or) ASP.net (or) PHP programs etc...
* During forwarding request operation url in the browser address bar will not be changed.	* During sendRedirection operation the url in the browser address bar will be changed.
* Suitable when source servlet program and destination servlet program resides in the same web application.	* Suitable when source servlet program and destination program resides in the two different web applications.

Example on send Redirection :

servlet program redirects the request to non java web resource programs of Internet websites.



* The above application act as hub for working with multiple search engines.
* The above application the servlet SearchSrv redirects the request to search engine based on the search engine that is choosen in the form page.

* To work with above application gather the web resource program names and request parameter names of different search engines as shown below.

For google

http://www.google.co.in/search?q=new+seven+wonders

request parameter value

For yahoo

http://search.yahoo.com/search?p=new+seven+wonders

request parameter name

For bing

http://www.bing.com/search?q=new+seven+wonders

web resource program name

* For ~~exam~~ source code of above diagram refer application (9) of the page numbers 73-75.

How many ways are there to pass data between the source servlet program and destination web resource program?

If source servlet program and destination web resource program resides in same web application then use

(1) request attributes

(2) session attributes

(3) servletContext attributes.

If source servlet program and destination web resource program resides in two different web applications of same/different server(s).

(1) append query string to URL of res.sendRedirect(-) method

```
res.sendRedirect("url?<query string>");
```

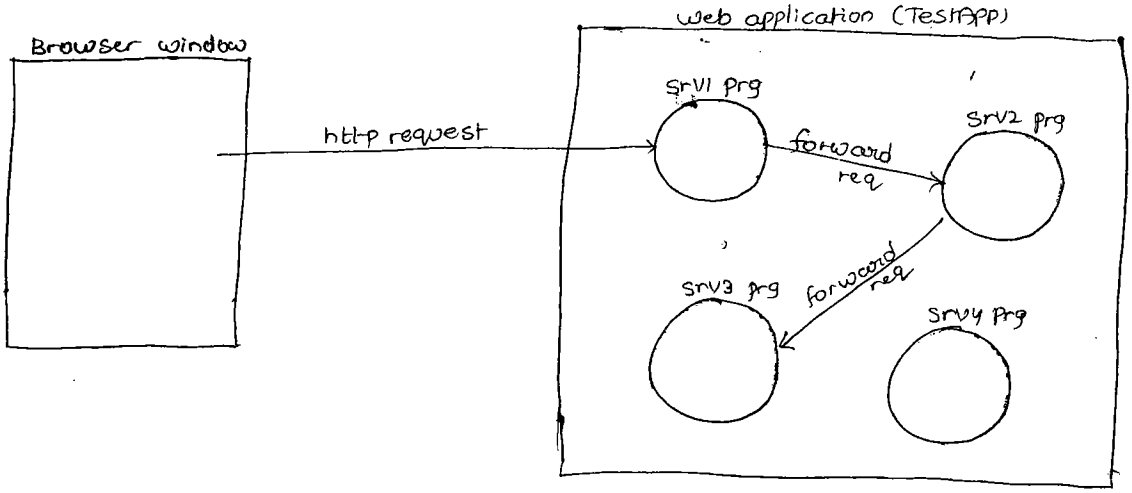
Ex:

```
res.sendRedirect("http://localhost:7001/TestApp2/S2Url?p1=10&p2=20");
```

* Attribute is a special logical name that can hold java object as an value.
* Attributes provide more visibility and accessability to the data of the application compare to the variables declared in servlet program.
* Attributes allows us to store only objects. If we give simple values to attributes then they will be converted into wrapper class objects and will be stored as attribute values.

Request attributes

* These attributes allocates memory in request object and they are visible through out request cycle.



* Request attributes that are created in one servlet program of servlet chaining are visible in other servlet programs of same servlet chaining because all the servlet programs of a servlet chaining will use same request and response objects.
* In the above diagram the request attribute created in Srv1 program is visible and accessable in Srv2, Srv3 program but not visible in Srv4 program. because Srv1, Srv2, Srv3 programs are using same request and response objects by participating in servlet chaining and Srv4 program is not participating in that servlet chaining.

To create request attribute

```
req.setAttribute("name", "raja");
req.setAttribute("age", new Integer(50));
req.setAttribute("total", 600);
```

attribute name → attribute value

attribute value will be converted into wrapper class object Integer through autoboxing.

To modify request attributes values

```
req.setAttribute("name", "rajal");
req.setAttribute("age", new Integer(45));
req.setAttribute("total", 550);
```

NOTE: The method setAttribute(--,--) can create new request attribute or can modify existing request attribute value.

To read values from request attributes

```
String S1= (String) req.getAttribute("name");

Integer S2 = (Integer) req.getAttribute("age");

int S3 = req.getAttribute("total");
```

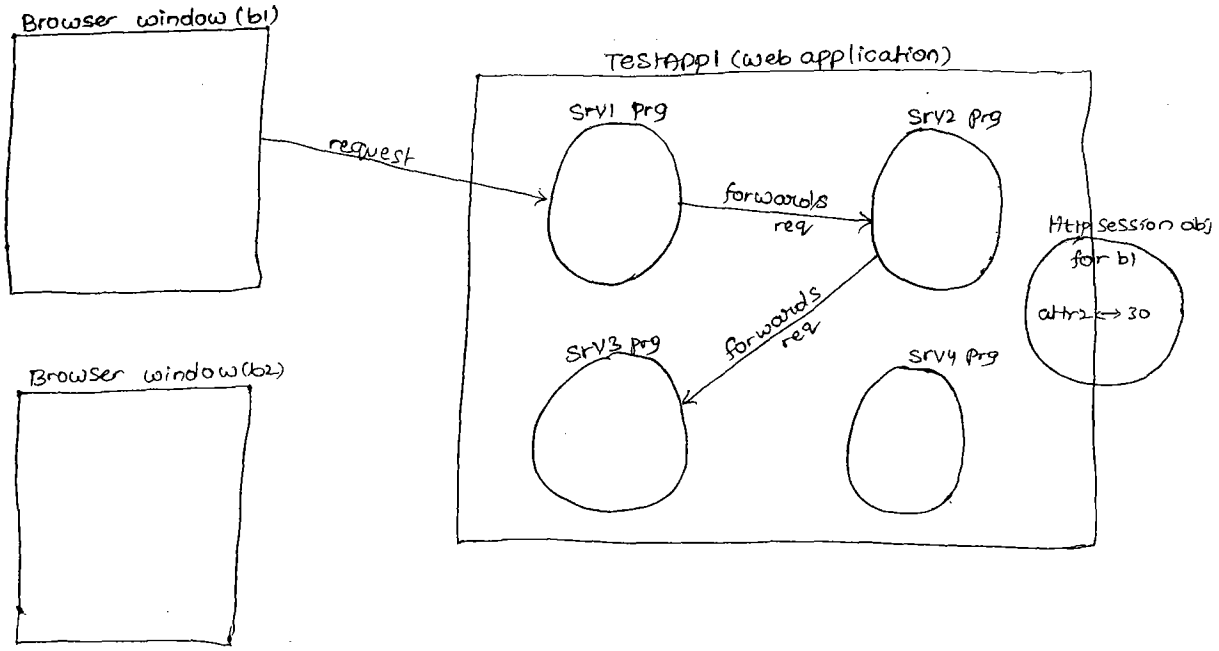
Converts wrapper class object Integer to simple int through autounboxing concept.

To delete/remove request attribute

```
req.removeAttribute("name");
req.removeAttribute("age");
req.removeAttribute("total");
```

Session Attribute

* Session attributes allocates memory in http session object. The http session object allocates memory on the server and it is one per browser window so the session attributes of http session object are visible and accessable in all web resource programs of web application irrespective of their request and response objects but they must get request from that browser window for which http session object and its attributes are created.



* The session attribute created in srv1 servlet program by getting request from browser window b1 is visible and accessable in all other servlet programs of web application but they must get request from same browser window b1.
* session attributes are global attributes in the web application but they are specific to a browser window (client).

To create / access Http session object for browser window on servlet

```
HttpSession ses = req.getSession();
```

↓ creates or locates HttpSession object

NOTE: This req.getSession() method checks the availability of session object on server for browser window, if available it provides access to that session object otherwise (if not available) it creates new session object (HttpSession object) for browser window on the servlet.

Session attributes

To create ses attribute or to modify ses attribute value

```
ses.setAttribute("attr2", new Integer(30));
```

To read ses attribute value

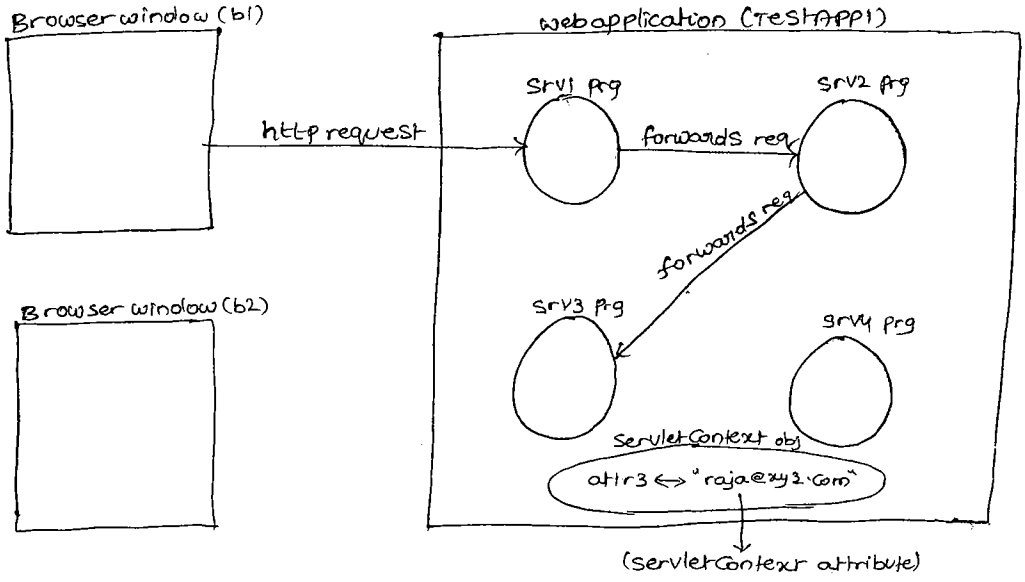
```
~~String~~ Integer s1 = (~~String~~ Integer) ses.getAttribute("attr2");
```

To remove ses attribute

```
ses.removeAttribute("attr2");
```

ServletContext attributes

* ServletContext attributes allocate memory in servletContext object. Since servlet context object is one per web application and global memory of web application so the servletContext attributes are visible and accessable in all web-resource programs of web application irrespective of request and response objects they are using and irrespective of the browser window from which they are getting request.
* In the below diagram the servletContext attribute created in srv1 program is visible and accessable in all the remaining servlet programs of web application irrespective of the request and response objects they are using and irrespective of browser window from which they are getting request.



To create or modify servletContext attribute

```
ServletContext sc = getServletContext();  //gives access to servletContext object
sc.setAttribute("attr3", "raja@xyz.com");
```

To read servletContext attribute value

```
String s1 = (String) sc.getAttribute("attr3");
```

To remove servletContext attribute

```
sc.removeAttribute("attr3");
```

What is the difference between request parameters and request attributes?

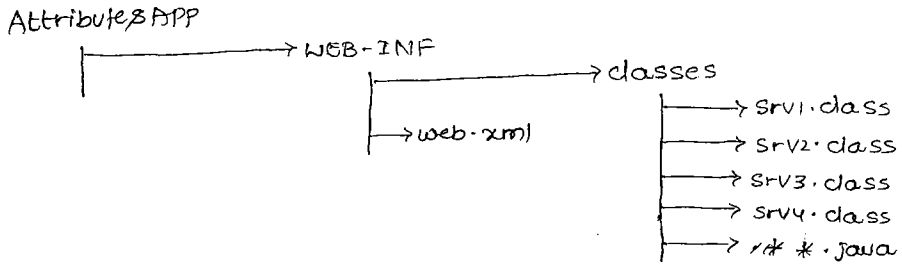
1) Request parameters are client supplied input values to web resource program (form data). Request attributes are programmer created additional data to pass from one web resource program to another web resource program when they are participating in chaining (like servlet chaining).

Conclusion

* If source servlet program and destination program resides in the use same request and response objects then use request attributes to send the data.
* If source servlet program and destination program are getting request from same browser window then we use session attributes for sending data.

* If source servlet program and destination webresource program can not satisfy above two conditions then use ServletContext attributes for sending data.

Example application on to understand the scope of various attributes



Srv1.java

```
Public class Srv1 extends HttpServlet
{
  public void service(HttpServletRequest req, HttpServletResponse res) throws SE, IOE
  {
    //general settings
     res.setContentType("text/html");
     PrintWriter pw = res.getWriter();
     //creating request attribute
      req.setAttribute("attr1", "raja");
      //creating session attribute
      HttpSession ses = req.getSession(); //creates session obj
      ses.setAttribute("attr2", new Integer(30)); // creates session attribute
      //creating servletContext attribute
       ServletContext sc = getServletContext(); //creates session obj
       sc.setAttribute("attr3", "raja@xyz.com"); // creates servletContext attribute
       // forward req to srv2 prg
       RequestDispatcher rd = req.getRequestDispatcher("/srv2");
        rd.forward(req, res);
  } //service(-,-)
} // class
```

srv2.java

```
public class srv2 extends HttpServlet
{
    public void service(-,-) throws SE,IOE
    {
        //general settings
         res.setContentType("text/html");
         PrintWriter pw = res.getWriter();
        //reading req attribute value
        pw.println("srv2: attr1(request) attribute value:" + req.getAttribute("attr1"));
        //reading session attribute value
         HttpSession ses=req.getSession(); //gives access to session obj
         pw.println("srv2: attr2(session)attribute value:"+ ses.getAttribute("attr2"));
        //reading servletContext attribute value
         ServletContext sc = getServletContext();
         pw.println("srv2: attr3 (servletContext) attribute value:"+ sc.getAttribute("attr3"));
        // forward req to srv3 prg
         RequestDispatcher rd= req.getRequestDispatcher("/srv3");
         rd.forward(req, res);
    } // service(-,-)
} //class
```

srv3.java

same as srv2.java and don't forward request to any other servlet.

srv4.java

same as srv3.java

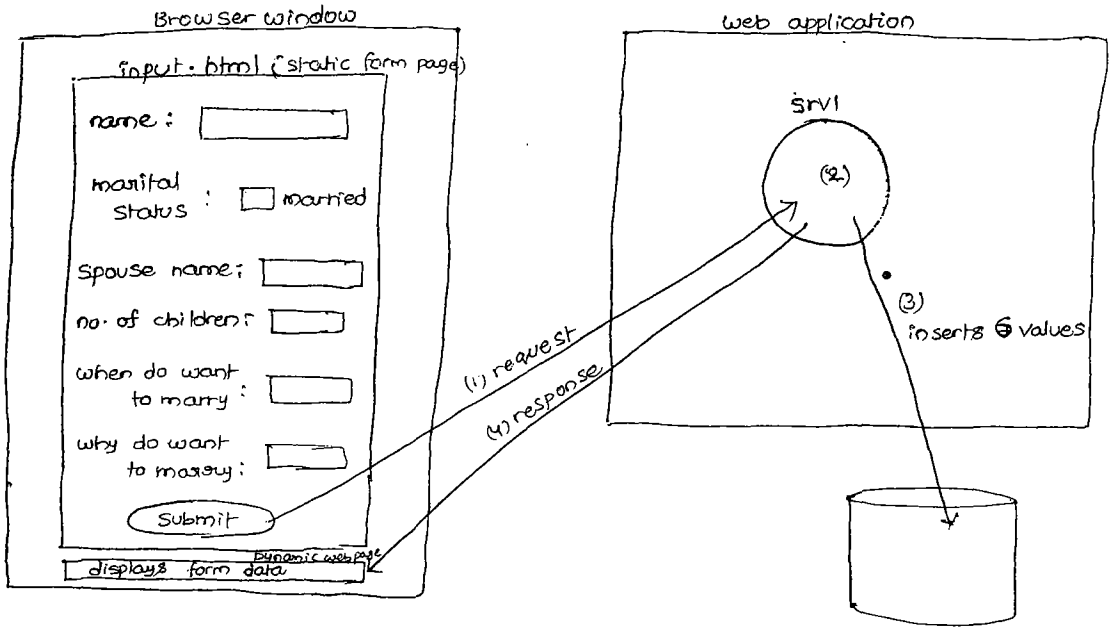
To test the above application give first request to srv1 program and give other than first request from to other than srv1 program from same or different browser windows and check the visibility of attributes.

http://localhost:2020/AttributesApp/srv1
http://localhost:2020/AttributesApp/srv2
http://localhost:2020/AttributesApp/srv3
http://localhost:2020/AttributesApp/srv4

web.xml

Configure srv1 servlet program with /srv1 program url pattern.
" srv2 " /srv2 " "
" srv3 " /srv3 " "
" srv4 " /srv4 " "

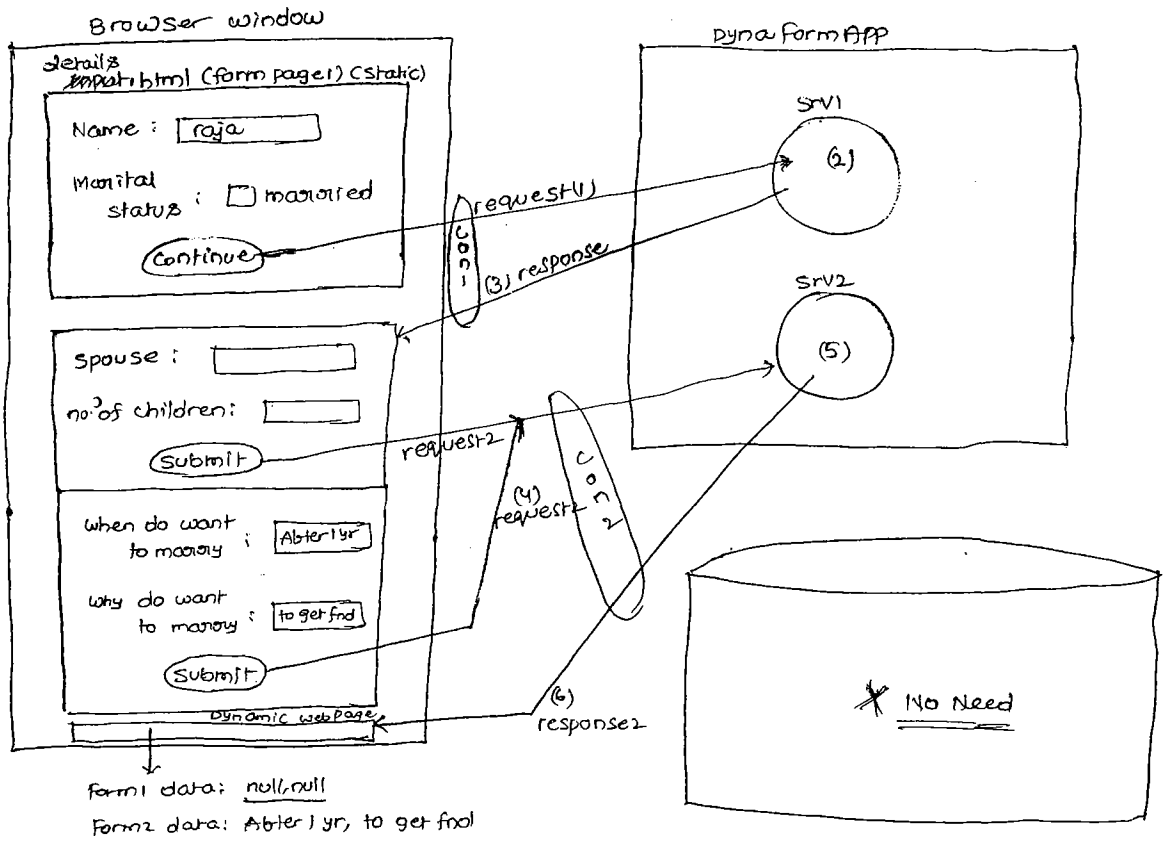
* The form page that is prepared through .html file is called as static form page (fixed content).
* The form page that comes as response generated by servlet or JSP program is called as dynamic form page (dynamic content).
* Instead of asking all the details of end user in a single form page it is recommanded to ask those details in multiple form pages having the support of dynamic form pages.



Bad designing of form page

* In the above static form page the end user needs to read and answer some unnecessary questions when he selects or deselects the marital status check box. To overcome that problem work with dynamic form page shown below.
* In the below diagram the content of form page 2 is dynamic and dynamically generated by srv1 program based on the marital status value of request1 data.
* All web applications are stateless web applications by default that means they can not remember client data across the multiple request during a session.
* A session is a set of continuous and related operations (request) performed on the web application by a client (browser window /end user).

Ex: sign-in to sign-out in gmail.com, start of the class to end of a class on a particular day.



Good form Designing but web application is stateless here. that means while processing request2 in srv2 program we can't use request1 data (form1 data).

* The stateless behaviour of web application is nothing but while processing current request in any web resource program we can't use previous request data that means while processing request2 we can't use request1 data. similarly while processing request3 we can't use request1 and request2 data as shown in the above diagram.

* web applications are stateless because the protocol http is given as stateless protocol. According to this one new connection will be created between browser window and web server for every request and this connection will be closed automatically once request related response goes to browser window.

Ex: If browser window gives 10 requests to a web application then 10 connections will be created between browser window and webserver, due to this one connection related client's request data can not be used in other connection related client's request processing (that means we can't use previous request data in web resource program while processing current request).

* For the above diagram based application refer application(10) of page nos 74 - 76
* By placing <form> tag, <input> tag in pw.println( ) statements of servlet program we can generate dynamic form page from that servlet program. (refer Srv1.java of page no. 75).
* In Naukari.com registration process multiple forms will be there. In that first form page is static form page and other form pages are dynamic form pages. In that based on the data given in form page1 the questions in form page2 will be rendered. similarly base on data in form page2 the question the questions in form page 3 will be rendered.
* web applications are stateless because they are using a stateless protocol called Http.

<u>Why HTTP is designed as stateless protocol. what is the problem if http comes as statefull protocol ?</u>

(A) If HTTP is statefull protocol for multiple requests given by client to web application single connection will be used between browser window and web server across the multiple requests. This may make clients to engage connections with web server for long time even though the connections are idle. Due to this the web server may reach to maximum connections even though most of its connections are idle.

To overcome that problem HTTP is given as stateless so no client can engage connection with web server for long time. more ever the connection will be closed automatically at the end of each request related response generation. In internet environment since there is a chance of having huge amount of clients for each web site it is recommanded to have stateless behaviour for http.

* If web application is capable of remembering a client data during a session across the multiple requests then that web application is called as stateful web application.
* In state ful web applications the web resource programs can use the data of previous request while processing current request that means while processing

request2 it can use request1 data and etc...

* Even though http is stateless protocol we need to make our web applications as stateful web applications. For this we need to work with the following session tracking or session management techniques.

(1) Hidden Form Fields
(2) Http Cookies
(3) Http Session with Cookies
(4) Http session with URL rewriting (recommanded)

* Session tracking / session management is all about making web application as stateful web application by remembering client data across the multiple requests during a session.

<u>Real Time implementations of this session tracking / session management</u>

(1) Remembering email id and password during gmail.com email operations.
(2) Remembering username and password during online shopping operations in websites.
(3) Remembering previous forms data until last form data arrives in online applications / registrations
(4) while customizing webpage look in website as required for enduser.
(5) while rendering direct advertisements in websites.

and etc...

* If advertisements are rendered based on the kind of content surfing / browsing done by end user then that advertisement is called as direct advertisement.

<u>Ex</u>: In google.com when you search for the results of seven wonders we will get advertisement related to tours and travels.

<u>working with Hidden Form fields based session tracking technique</u>

* An invisible text box of the form page is called as hidden box.
* when form page is submitted ~~the form data goes to~~ the hidden box value goes to web resource program along with the request as request parameter value.

<u>In form page</u>

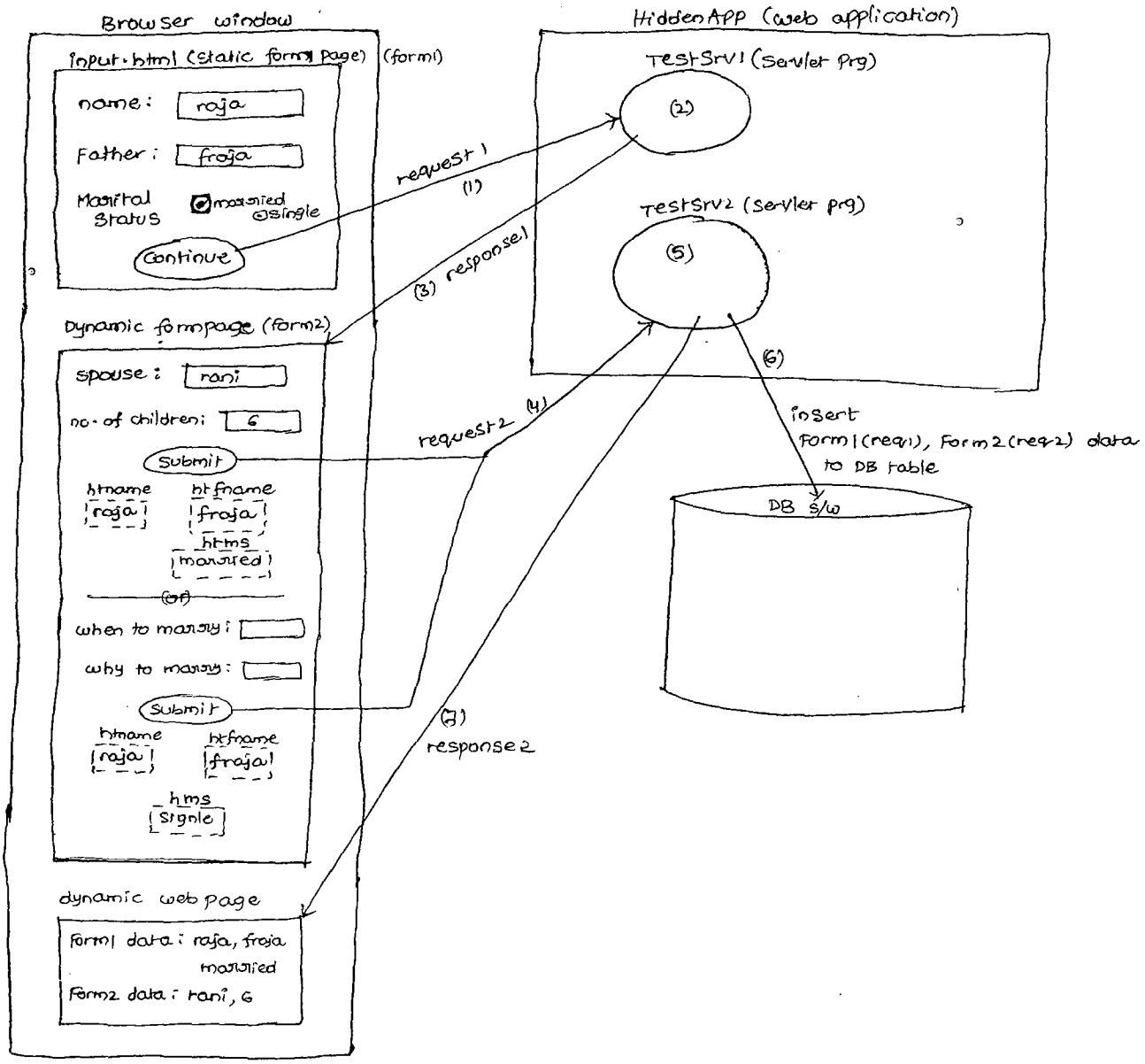
```
<input type="hidden" name="H" value="hello"/>
```

In servlet program

string s1= request.getParameter("t1");

gives "hello" to s1 variable

Making web application as stateful with the support of hidden boxes (hidden form fields)



* In the above diagram TestSrv1 program receiving form1/request1 data and generating form2 dynamically. In this form2, form1 data is added as hidden box values so when TestSrv2 program gets request from form2 it is able to read form1/request1 data along with form2/request2 data. This is nothing but session tracking based on hidden form fields.

* The above diagram based web application is stateful because TestSrv2 is able to use request1 data while processing request2.
* For above diagram based application refer application (11) of the page nos 76–78.

Advantages of Hidden form fields based session tracking technique

* Basic knowledge of HTML is enough to work with this technique.
* Hidden boxes resides in web pages of browser window so ~~they~~ they do not provide burden to the server.
* This technique can be used along with all ~~ser~~ kinds of server side technologies and all kinds of web servers and application servers.

Disadvantages

* Hidden form fields can not store java objects as values. They can just store text values.
* Hidden boxes doesn't provide data secracy because their data can be viewed through view source option
* Hidden boxes data travels over the network along with request and response. so we can say network traffic will be increased.
* while creating each dynamic form page we need to add the previous form pages data as hidden box values. This increases burden on the programmer.

Http Cookies

* Cookies are the small textual informations which allocate memory at client side by remembering client data across the multiple request during a session.
* The web resource programs of web application create cookies at server side but these cookies comes to client side along with the response and allocates memory at client side.
* Cookies of client side goes back to their web application along with the request given by clients (browser windows to web resource programs).
* there are two types of cookies

  (1) In memory cookies / per session cookies

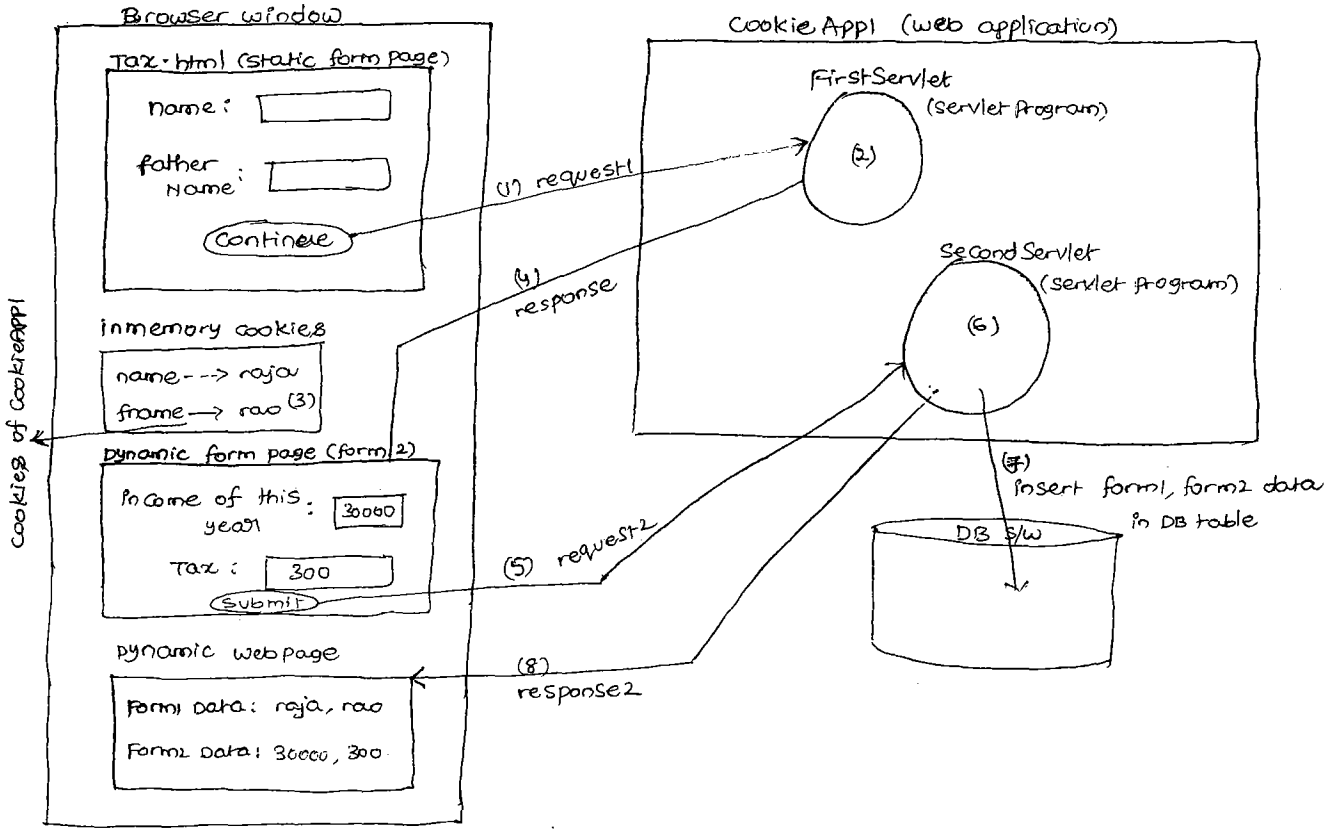
  (2) persistent cookies

(1) → allocates memory in browser window, having "-1" has the expiry time (no expiry time). These cookies will be destroyed automatically once Browser window is closed.

(2) → These cookies allocates memory in the files of client machine hard disk having positive number as expiry time. These cookies will not be destroyed when browser window is closed but they will be destroyed when their expiry time is completed / reached.

* The cookie that is created without expiry time is called as In memory cookie. The cookie that is created with the expiry time is called as persistent cookie.

* To work with cookies we must deal with pre defined http servlet class based servlet programs.



* Every cookie contains name and value as string information.

* At client side or browser window we can see multiple cookies belonging to different domains or web applications. Every cookie remembers the web application name or domain name for which it is created.

* Cookies of one web application that are there at client side will go to their web application along with the request then browser window gives request back to that web application.

with respect to the diagram

(1) The form page1 Tax.html gives request1 to FirstServlet program of CookieApp1 web application.

(2) FirstServlet program reads form1 data, creates two in memory cookies having that form1 data.

(3), (4) → FirstServlet program generates response to browser window adding the cookies & having dynamic form page. These cookies allocate memory on browser window as in memory cookies.

(5) The dynamic form page (form2) generates request to second servlet program of CookieApp1 application so the two cookies of CookieApp1 will go to SecondServlet program along with the request.

(6) SecondServlet program reads form2 data normally but reads form1/request1 data from the cookies of request2. This indicates SecondServlet is able to use form1/request1 data while processing form2 request/request2. (which is nothing but session tracking).

(7) SecondServlet writes both form1, form2 data to db table.

(8) SecondServlet program generates dynamic web page having form1 and form2 data.

* The web resource programs of web application send cookies to client machine along with the response as 'set-cookie' response header values.

* the browser window sends the cookies ~~of~~ back to web application along with the request given to web resource programs as ~~resque~~ cookies request header value.

<u>Cookie API</u> : (working with javax.servlet.http.Cookie class)

<u>to create cookies and to add them to response</u>

```
Cookie ck1 = new Cookie ("ap", "hyd");  //creating new Cookie
```

"ap" → Cookie name, "hyd" → Cookie value

(must be string)

```
resp.addCookie(ck1);  //adding Cookie to response.
```

↓ Here ck1 Cookie is in memory cookie (because no expiry time is set)

```
Cookie ck2 = new Cookie("mh", "mumbai");
ck2.setMaxAge(1800);  → setting expiry time for cookie
resp.addCookie(ck2);  → adding cookie to response.
```

Here ck2 cookie is persistent cookie having 1800 seconds as expiry time.

to know max Age (Expiry time) of Cookie

```
int time = ck1.getMaxAge();  → gives -1 (for in memory cookie)
int time = ck2.getMaxAge();  → give 1800 seconds
```

to modify cookie value

```
ck1.setValue("new hyd");
ck2.setValue("navi mumbai");
```

To know domain name/web application name of the cookie

```
String d1 = ck1.getDomain();
String d2 = ck2.getDomain();
```

To set Comment to Cookie

```
ck1.setComment("holds ap's capital city");
ck2.setComment("holds mh's capital city");
```

to get Comment of Cookie

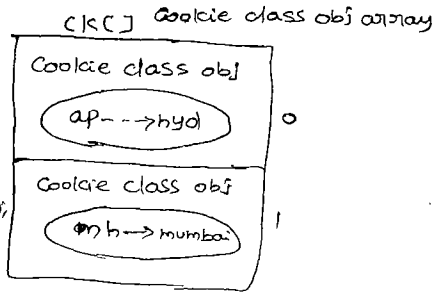
```
String c1 = ck1.getComment();
String c2 = ck2.getComment();
```

to read cookie values

```
String ck[] = req.getCookies();
```

↓ reads all the cookies from the current http request.

```
if (ck != null)
  {
    for (int i=0; i<ck.length; i++)
      {
        pw.println("Cookie name"+ck[i].getName());
        pw.println("Cookie value"+ck[i].getValue());
      }
  }
```



To delete Cookies

* Cookies can not be deleted ~~manually and~~ programatically.
* In memory Cookies will be destroyed automatically once their browser window is closed. where as persistent cookies will be destroyed if their expiry time is reached / completed.
* For example application to know the basics and behaviour of Cookies refer application (12) of the page nos (78) and (79).
* In windows XP while working with internet explorer the persistant cookies of web application will be stored in the files created in c:\documents and settings\administrator\cookies folder and the notation of the file is :

<windows user> @ <web application name> [<Cookies count>] . txt

Ex : administrator @ CookieApp [2] . txt

Writing HTML code in servlet program with attribute values of html tags

```
pw.println("<table border=\"1" "align=\"center\" >");
                    (or)
pw.println("<table border=1 align=center>");
                    (or)
pw.println("<table border='1' align='center'>");
```

: If we try to modify the content of the files where persistant cookies are stored then persistant cookies including that file will be destroyed.

* Internet explorer allocates the memory for in memory cookies on browser window where as the memory for persistant cookies will be allocated as files on the client machine hard disk.
: In netscape navigator both in memory and persistant cookies allocate memory on browser window but persistant cookies will be there with expiry time.

NOTE:

: Netscape navigator also internally uses client machine hard disk files for storing persistant cookies.

* For source code of example application based on the above diagram refer application (13) (13) of the booklet page numbers (79) to (81)

(79) to (81)

* Every cookie remembers the domain name or web application name to which it belongs to.
* Cookies belonging to yahoo.com web site will go to yahoo.com web site only when browser window or client gives request back to that yahoo.com web site.
* In yahoomail.com, gmail.com web site related login page ☑ remember me on this computer option stores the given login details (username, password) as persistent cookies on client machine hard disk. That means by using persistent cookies we can remember client data during session and after session.

Advantages of Http Cookies :

(1) Cookies allocates memory at client side so they do not give any burden to server.

(2) All server side technologies and all web servers, application servers support cookies.

(3) Persistent cookies can remember client data during session and after session with expiry time.

Disadvantages

(1) Cookies can not store java objects as values. They can store only strings.

(2) cookies data can be viewed ~~the to~~ through browser setting or through the files of file system. so they do not provide data secracy.

(3) Cookies can be deleted explicitly through browser settings it may fail session tracking

(4) There is a restriction on no. of cookies that can be there in browser window. per ~~b~~ browser window and per web application max of 20 cookies. All together max of 300 cookies per browser window.

(5) cookies can be blocked / restricted coming to browser window. This fails session tracking.

IE → to delete cookies

Tools → Internet options → delete cookies

Nestcape to delete

Tools → Cookie manager → remove all cookies

IE → to block cookies

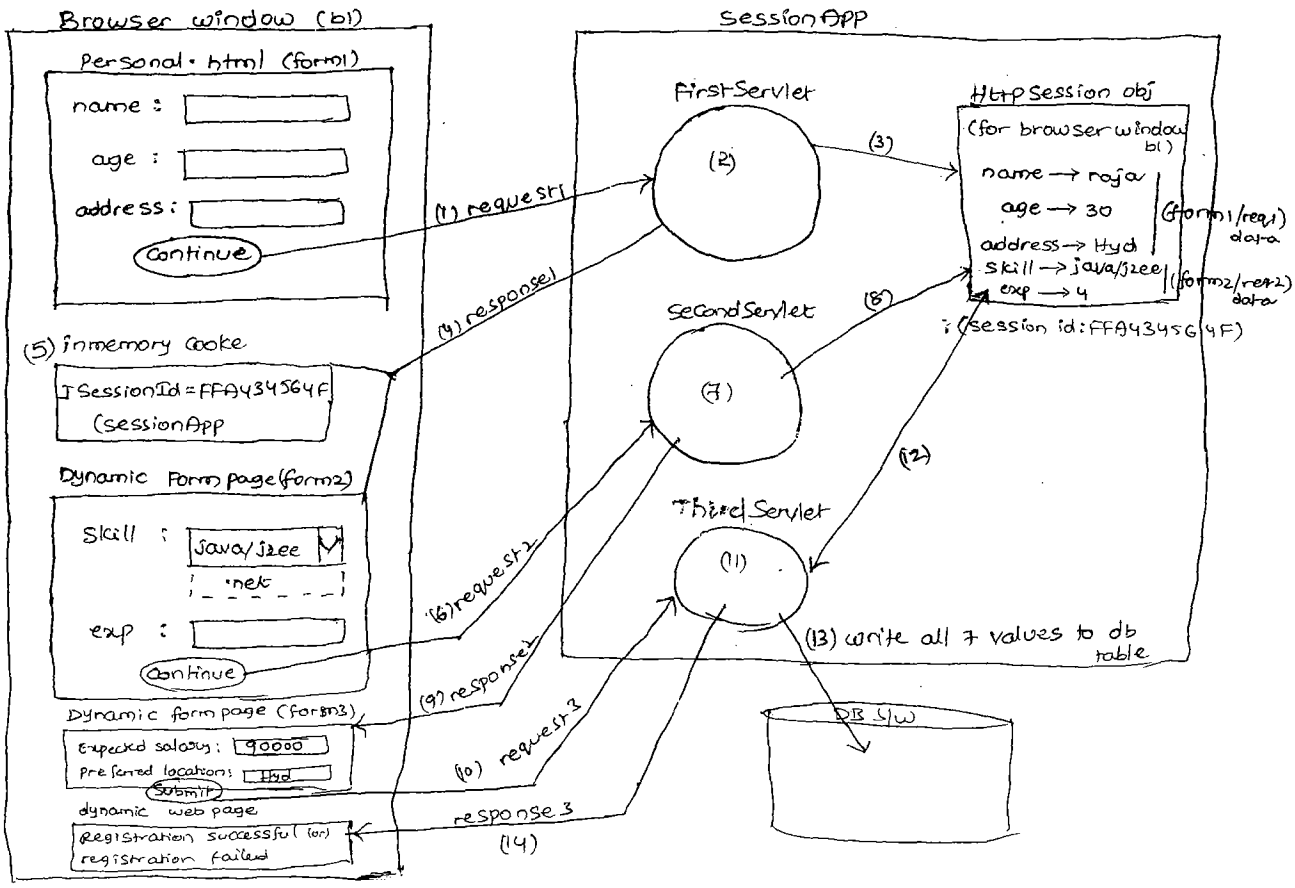
Tools → Internet options → Privacy → use slider

Netscape → to block cookies

Tools → Cookie manager → Block Cookies from this site

Http Session with cookies :

* Http session object allocates memory on the server and remembers client data across the multiple requests in the form of session attribute values.
* HttpSession object is one per browser window (client) so each Http Session object can be used to remember client data during a session.
* HttpSession object means it is the object of a servlet container supplied java class implementing javax.servlet.http.HttpSession interface.
* Every HttpSession object contains session id and this session id goes to browser window from web application and comes back to web application from browser window in the form of in memory cookie value so this technique is called HttpSession with cookies technique.



with respect to the diagram

(1) Browser window b1 launches personal.html (form1) and sends the request to FirstServlet of SessionApp application.

(2) FirstServlet reads Form1 data / request1 data.

(3) FirstServlet creates HttpSession object on the server for browser window b1 and writes Form1/ request1 data to that session object as session attribute values.

(4) FirstServlet generates form2 as dynamic form page and sends the session id of HttpSession object to browser window as in memory cookie value.

(5) The in memory cookie having session id of HttpSession object (b1) allocates memory on the browser window. That cookie also remembers its web application name SessionApp.

(6) End user fills up the form page (form2) and sends the request2 to Second servlet of SessionApp. Along with the request the session id will go as cookie value.

(7) Second servlet reads form2 data.

(8) Second Servlet uses the session id read from the cookie to get access to the HttpSession object of browser window b1 and writes form2 /request2 data to that HttpSession object as session attribute values.

(9) Second Servlet generates form3 as dynamic form page.

(10) form3 generates request3 to third servlet of SessionApp application. Along with this request session id goes to third servlet as cookie value.

(11) Third Servlet reads form3 data and session id from the cookie.

(12) Third Servlet uses session id collected from the cookie to get access to HttpSession object of browser window b1 and reads form1 / request1, form2/ request2 data from the session attributes of that HttpSession attributes. This indicates ThirdServlet is able to use request1 / form1 data and request2/ form2 data while processing form3 request or request3. This is nothing but session tracking.

(13) Third servlet writes form1, form2, form3 values to the db table as a record.

(14) Third Servlet generates dynamic web page having conformation message.

* when HttpSession object is created the session id of that session object automatically goes to browser window as in memory cookie value.
* In HttpSession with Cookies session tracking technique based web application the browser window (client) will be identified across the multiple request during a session based on the session id sent by the browser window along with the request.

Session API (working with javax.servlet.http.HttpSession interface)

To create/locate HttpSession object

a) HttpSession ses = req.getSession();

* This method creates new HttpSesion object on server for browser window if HttpSession object is not already available for browser window otherwise this method provides access to existing HttpSession object of that browser window.
* This method can create the new session between browserwindow and web application if session is not already between them otherwise this method makes current request to join in the existing session.
* If this method is called in FirstServlet program of above diagram then it makes request1 of browser window b1 begining new Session between browser window b1 and web application session App. If same method is called SecondServlet, ThirdServlet then it makes request2, requests of browser window b1 participating in existing session.

b) HttpSession ses = req.getSession(false);

* This method ~~creates~~ gives access to existing HttpSession object of browser window, if not available this method returns null (indicating new HttpSession object can not be created).
* when this method is called the request can always join in existing session but can not create new session between browser window and web application.
* In the above diagram it is recommanded to call req.getSession() in FirstServlet ~~and~~ to start the session and it is recommanded to call req.getsession(false) method in SecondServlet, ThirdServlet to make them participating in existing session.

c) HttpSession ses = req.getSession(true);
"same as (a)"

Conclusion:

to create new session / to locate existing session → use (a), (c) options
only to locate existing session → use (b) option.

to know Session ID

```
String id = ses.getID();
```

TO know session object creation time / session started time

```
long ms = ses.getCreationTime();
Date d1 = new Date(ms);
```

* In the above code ms represents milli seconds that are elapsed from 1970 Jan 1st 00:00 hours to the date and time of HttpSession object creation. Epoch standard

* TO know the last accessed time of HttpSession object

```
long ms = ses.getLastAccessedTime();
Date d2 = new Date(ms);
```

* The above code gives the last accessed date and time of HttpSession object.

TO get access to ServletContext object

```
ServletContext sc = ses.getServletContext();
```

TO know whether session is new or not

```
boolean b = ses.isNew();
```

* This method returns true when ses object is new object and just created object and its session id still yet to be delivered (sent) to client (browser window) otherwise this method returns false. (when session object is old object and its session id is already there with client).

* If this method is called in FirstServlet program of diagram (which receives only first request of the browser window) then this method returns true. If this method is called in other Servlet programs of web application then this method returns false.

To Invalidate the session

* Invalidating the session is nothing but closing the session between browser window and web application. In this process the entire data from session object will be removed and makes session object inactive object, ready for garbage collection.

a) when ses.invalidate() method is called

b) when browser window is closed.

NOTE: since session id will be stored as in memory cookie value of browser window and that in memory cookie will be destroyed once browser window is closed so the session will be invalidated once browser window is closed.

c) when MaxInactiveInterval / SessionIdletimeout period is completed/reached.

* If session object is continuously idle for certain amount of time then it will be invalidated automatically. The default session idle timeout period is 30 minutes (in most of the servers). But this can be changed explicitly either by using programatic approach or declarative approach.

(I) Programatic approach (Java code)

In servlet prg/ JSP prg

```
ses.setMaxInactiveInterval(1500);
```

seconds

(II) Declarative approach (xml code)

In web.xml

```
<web-app>
    <session-config>
        <session-timeout>20</session-timeout>
    </session-config>
</web-app>
```

minutes

* once session idle timeout period or maxInactiveInterval period is completed the underlaying web server automatically expires the session (invalidates the session).

To know current maxInactiveInterval period / SessionIdletimeout period

```
int t1 = ses.getMaxInactiveInterval();
```

NOTE: The above method returns 30 if no value explicitly set as SessionIdle timeout period.

If you set sessionIdleTimeout period in both programatic and declarative approaches with two different values, can you tell me which value will be effected finally ?

(A) Since the code of servlet program executes after web.xml code so the time specified through programatic approach will be effected by overriding the time specified through declarative approach.

* In HttpSession object the client data will be preserved in the form of session attribute values.
* HttpSession object and its session attributes are visible and accessable in all web resource programs of web application but they must get request from that particular browser window (client) for which this session attribute, and session objects are created.

TO create / modify session attribute

```
ses.setSetAttribute("age", new Integer(30));
```

ses.putValue(-,-) is deprecated method of setAttribute(-,-)

TO read session attribute value

```
Integer si= (Integer)ses.getAttribute("age");
```

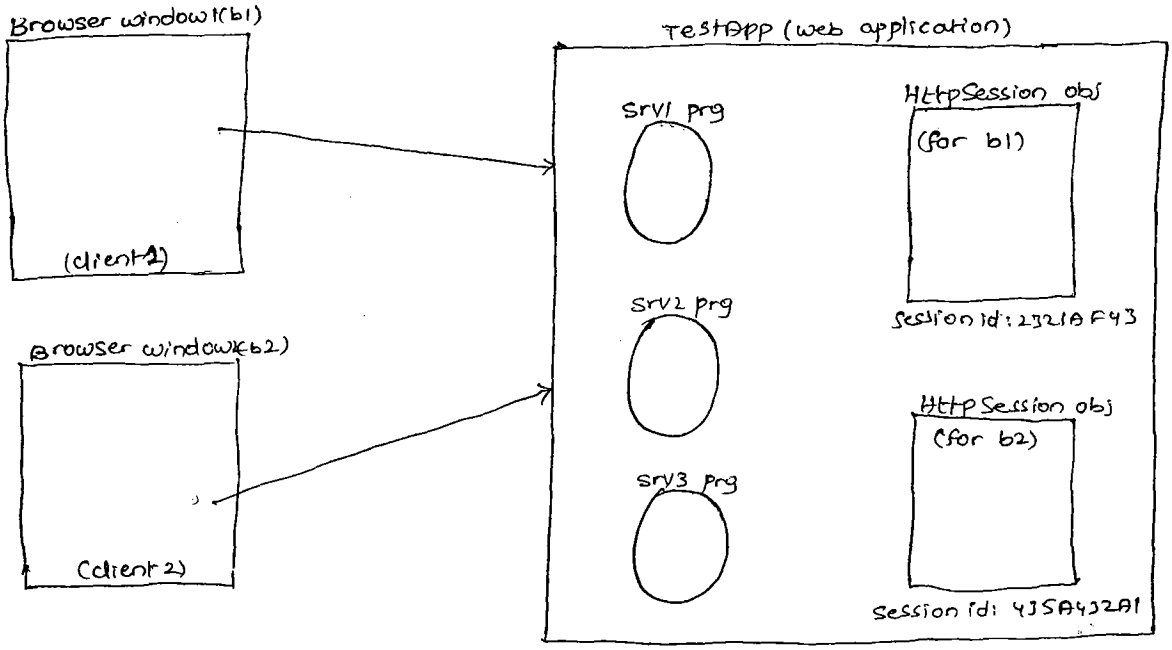
ses.getValue(-,-) is deprecated method of getAttribute(-,-)

TO remove session attribute

```
ses.removeAttribute("age");
```

ses.removeValue(-,-) is deprecated method of removeAttribute(-,-)

* For example application on HttpSession with cookies based on previous diagram refer page nos ~~80~~ - ~~88~~ 81 - 85
* If multiple clients (browser windows) are giving request to a webapplication in which HttpSession object based session tracking is enabled then for every browser window one HttpSession object will be created on the server having unique session id.



* In HttpSession with Cookies session tracking enabled web application the movement HttpSession object is created on server for browser window the servlet Container automatically creates one in memory cookie having sessionid and adds that Cookies to the response to send to the browser window (no need of performing manual work).

* while working HttpSession object based web application if browser window is closed in the middle of the session then the existing session will be invalidated and that session will not be continued with new browser window.

: while working with Http Session object based application what happens if underlaying server is restarted in the middle of the session ?

1) The session will be continued.

Sever collects the data of Http session objects and writes to files through serialization process having session ids when programmer shutdown the server. when server is restarted HttpSession object will be created having old data and old session ids by reading data from the above said file through de-Serialization process. Due to this the session will be continued even though the server is restarted in the middle of session.

Advantages of Http session with Cookies session tracking technique.

(1) Http session objects allocates memory on the server holding client data

during session as session attribute values. So there will be data secracy for client's session data.

(2) client data during session will not travel along with request and response over the network so this reduces network traffic between client and web server.

(3) The session attributes of Http session objects can take java objects as values.

(4) All java based web servers and application servers support this technique.

(5) This technique allows the programmer to specify session idle timeout period to invalidate inactive session objects.

Dis advantages:

(1) Http session objects allocates memory on the server. This increases burden on the server.

(2) If cookies are restricted coming to browser window this technique fails to perform session tracking.

Http session with url rewriting

* The limitation with third technique is if cookies are restricted to coming to browser window then third technique fails to perform session tracking because it uses in memory cookie to send session id to browser from web application along with response and to bring session id back to web application from browser window along with request.

To overcome that problem work with Http session with URL rewriting which doesnot use cookies to send and receive session id. More ever this technique appends session id to a url that goes to browser window from web application along with the response and that comes back to web application from browser window along with the request. Generally we append session id to the action url (the action attribute of form tag) of dynamic form page generation code.

NOTE:

res.encodeURL("s2url"); → gives s2url;jsessionid: 424354 1A3F723

This method use URL appended with current object's session id as shown above.

* By taking above kind of URL as action URL of dynamic form page we can perform http session with URL rewriting based session tracking.

In Servlet program

```
pw.println("<form action=" + res.encodeURL("surl") + " method=get>");
pw.println("- - - - - - -");
pw.println("- - - - - - -");
pw.println("</form>");
```

*** keeps the url appended with sessionid as the action attribute value of <form> tag

* For example application on Httpsession with URL rewriting refer application (15) of the page nos 85-88.

Advantages and disadvantages of Http Session with URL rewriting

same as Http Session with Cookies technique but this technique also works even though cookies are restricted coming to browser window.

Conclusion on Session tracking technique

* while developing large scale and commercial web site which contains huge customer base take the support of Http Cookies technique.

Ex: www.gmail.com, www.yahoomail.com

* while developing medium scale and certain organization based website which contains limited amount of customers use http session with URL rewriting technique.

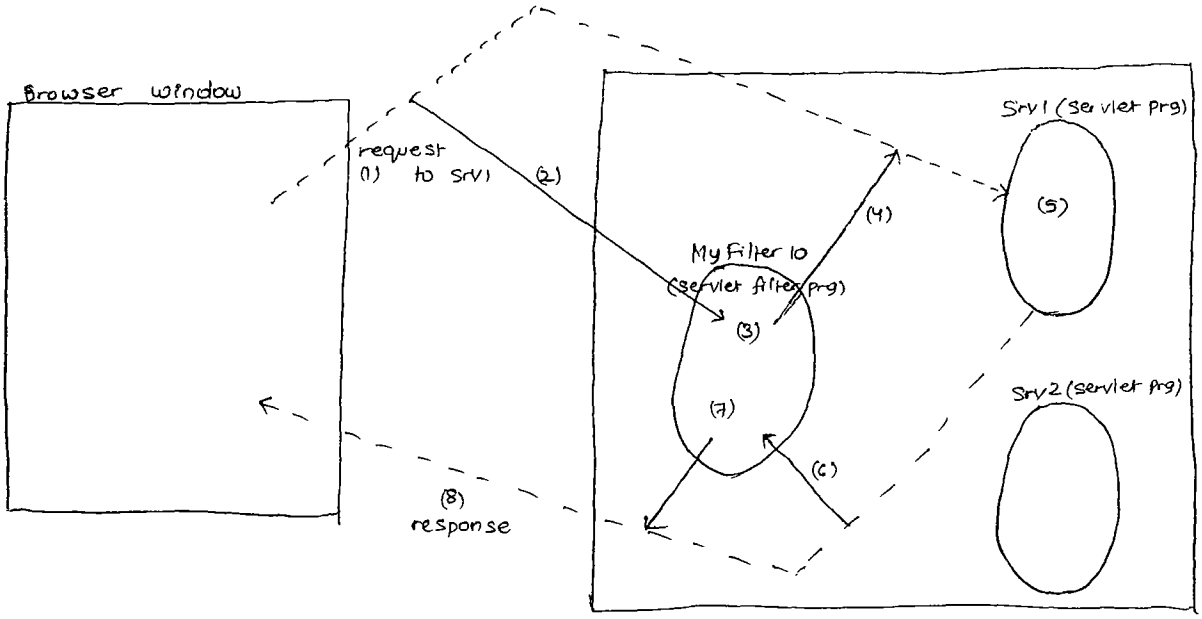
Ex: www.citibank.com, www.satyatechnology.it.com

* we can use multiple session tracking techniques in a single web application development. For example the online shopping websites like www.yebhi.com uses both http Cookies, http session with URL rewriting techniques.

Cookies → for holding shopping cart information

URL rewriting → for holding login details (username, password)

* A servlet filter is a special web resource program of java web application that is capable of trapping and taking request and response of other web resource programs of that web applications.
* In servlet filter program we can keep the common and global pre request processing logic and post response generation logic
* To add new functionalities to web application without changing the source code of existing web resource programs we can take the support of servlet filter programs.



With respect to the above diagram

(1) Browser window generates request to Srv1 servlet program.

(2) The servlet filter program traps and takes that request.

(3) Servlet filter program executes the common and global pre requesting process logic (like authentication logic)

(4) Servlet filter program forwards the request to actual Srv1 servlet program.

(5) The main request processing logic of Srv1 program executes and generate response.

(6) Servlet filter program traps and takes the response.

(7) Filter program executes the common and global post response generation logic (like adding more content at the end of response).

(8) Servlet filter program sends response to browser window.

Servlet filter basics

* servlet filter program is a java class that implements javax.servlet.Filter interface.
* Every servlet filter program must be configured in web.xml file using filter, filter mapping tags.
* To link servlet filter program with servlet program of web application the url pattern of servlet program must be taken as the url pattern of servlet filter program.
* servlet container creates our servlet filter class object either during server start up (or) during the deployment of the web application.
* we can map servlet filter program with one servlet program, multiple servlet programs (or) all servlet programs of web application.
* There are 3 lifecycle methods in servlet filter program

(1) public void init (FilterConfig fg)

(2) public void doFilter (servletRequest req, servletResponse res, Filterchain fc)

This object points to the target mapped servlet programs of filter program.

(3) public void destroy()

(1) → executes during instantiation and initialization process of our servlet filter program.

(2) → Executes for every request trapping and response trapping.

(3) → Executes when ServletContainer is about to destroy our servlet filter class object

There are 3 types of filter programs.

(1) request Filter (Contains only pre request processing logics)

(2) response Filter (Contains only post response generation logics)

(3) request - response Filter (Contains both pre-request processing and post response generation logics).

Examples of request Filters

Authentication Filter

Authorization Filter

Examples of response Filters

Signature Filter

Tags Filtering Filter (removes html tags from response of the servlet programs which are not supported by browser window).

Compression Filter (reduces the size of the response content)

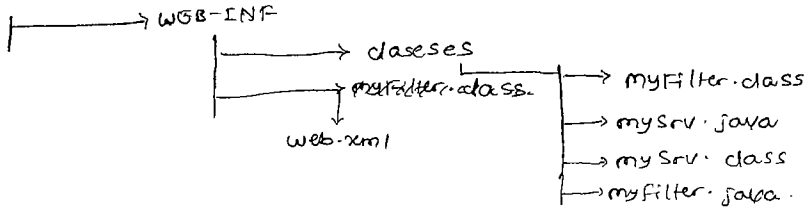
Examples of request-response Filter

PerformanceTestFilter (This filter holds request traping time and response traping time of certain servlet program and calculates difference between both timings to decide the performance of the servlet program).

* FilterConfig object is right hand object of our servlet Filter class object and it can be used to read init parameter values of filter program in web.xml file.
* init(-) method of servlet Filter program contains initialization logic.
* doFilter(-,-,-) of servlet Filter program contains pre-request processing logic and post response generation logics.
* destroy( ) of servletFilter program contains uninitialization logic.

understanding flow of execution for servlet filter program when it is mapped with a servlet program.

① TestAP



MyFilter.java

② public class myFilter implements javax.servlet.Filter
↳ contains all three lifecycle methods of servlet ~~program~~ Filter.

```
{
    public void init (FilterConfig cg)
    {
        ③
        // initialization logic
    }
    public void doFilter (ServletRequest req, ServletResponse res, FilterChain fc)
    {
        ④ } Pre-request processing logic
        fc.doFilter (req, res); ⑩
    }
        ⑯ } post-response generation logic
```

```
        public void destroy()
        {
            ____ //uninitialization logic
        }
    }
```

MySrv.java (servlet program)

```
public class MySrv extends HttpServlet
 {
   public void service(-,-) throws SE, IOE
    {
      ____ (15)
      ____
      ____
    }
 }
```

web.xml

```
<servlet>
    <servlet-name> abc (13) </servlet-name>
    <servlet-class> MySrv (14) </servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name> abc (12) </servlet-name>
    <url-pattern> /test1 (11) </url-pattern>
</servlet-mapping>
<filter>
    <filter-name> xyz (7) </filter-name>
    <filter-class> MyFilter (8) </filter-class>
</filter>
<filter-mapping>
    <filter-name> xyz (6) </filter-name>
    <url-pattern> /test1 (5) </url-pattern>
</filter-mapping>
</web-app>
```

↓ matching with servlet program (mysrv) url pattern

Browser window

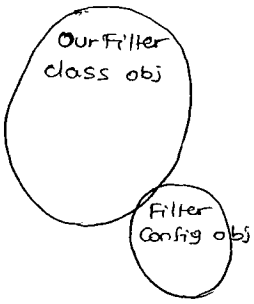
http://localhost:2020/TestApp/test1 (4)

(17) response will be displayed on browser window.

Compression Filter (reduces the size of the response content)

Examples of request-response Filter

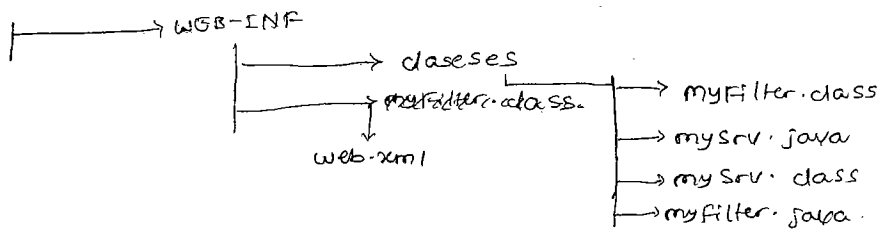
performanceTestFilter (This filter holds request traping time and response traping time of certain servlet program and calculates difference between both timings to decide the performance of the servlet program).



* FilterConfig object is right hand object of our servlet Filter class object and it can be used to read init parameter values of filter program in web.xml file.
* init (-) method of servlet Filter program contains initialization logic.
* doFilter(-,-,-) of servlet Filter program contains pre-request processing logic and post response generation logics.
* destroy( ) of servletFilter program contains uninitialization logic.

understanding flow of execution for servlet filter program when it is mapped with a servlet program.

(1)



MyFilter.java

(2)

```
public class myFilter implements javax.servlet.Filter
{
    public void init (FilterConfig cg)
    {
        (3) ___ //initialization logic
    }
    public void doFilter (ServletRequest req, ServletResponse res, FilterChain fc)
    {
        (9) ___ } pre-request processing logic
        fc.doFilter (req, res); (10)
        ___ (16) } post-response generation logic
    }
```

↳ contains all three lifecycle methods of servlet ~~Program~~ Filter.

```
        public void destroy()
        {
            ____ //uninitialization logic
        }
    }
```

MySrv.java (servlet program)

```
public class MySrv extends HttpServlet
{
    public void service(-,-) throws SE, IOE
    {
        ____ (15)
        ____
        ____
    }
}
```

web.xml

```
<servlet>
    <servlet-name> abc </servlet-name>   (13)
    <servlet-class> MySrv </servlet-class>   (14)
</servlet>
<servlet-mapping>
    <servlet-name> abc </servlet-name>   (12)
    <url-pattern> /test1 </url-pattern>   (11)
</servlet-mapping>
<filter>
    <filter-name> xyz </filter-name>   (7)
    <filter-class> MyFilter </filter-class>   (8)
</filter>
<filter-mapping>
    <filter-name> xyz </filter-name>   (6)
    <url-pattern> /test1 </url-pattern>   (5)
</filter-mapping>
</web-app>.
```

↓ matching with servlet program (mysrv) url pattern

Browser window

http://localhost:2020/TestApp/test1 (4)

(17) response will be displayed on browser window.

with respect to

(1) Programmer deploys TestApp web application in web server/application server.

(2) Based on web.xml entries servlet Container creates our filter class object (myFilter) either during server startup or during the deployment of the web application.

(3) servlet container completes initialization process of filter by executing the logic of init(-) method.

(4) End user gives request to servlet program mySrv from browser window (having /test1 in the request url).

(5), (6), (7), (8) → since MyFilter program is configured with MySrv program the servlet Filter traps and takes the above request that is given to mySrv program. In this process servlet Container locates MyFilter class object.

(9) servlet container calls life cycle method doFilter(-,-,-) and executes prerequ processing logic.

(10) The fc.doFilter(-,-) method call forwards the request to the target web resource program (mySrv).

(11), (12), (13), (14) → servlet Container completes instantiation and initialization of MySrv program based on its configuration done in web.xml file.

(15) servlet Container calls service method of MySrv program to process the request and to generate the response.

(16) The filter program (MyFilter) traps the response of mySrv program (servle program) and executes post response generation logic that is placed after fc.doFilter(-,-) method.

(17) The filter program MyFilter sends the response to browser window.

<u>Sample web application</u>

Srv1, Srv2, Srv3 are servlet programs.

F1 is servlet Filter program.

<u>to configure filter with all the servlet programs of web application</u>

Configure F1 filter having url pattern /*

<u>to configure F1 filter with Srv1, Srv2 programs of web application</u>

Srv1 program url pattern : /x/y/s1url

Srv2 program url pattern : /x/y/s2url

F1 filter program url pattern : /x/y/*

<u>to configure F1 filter with Srv1 program of web application</u>

Srv1 program url pattern : /s1 url

F1 filter program url pattern : /s1url

What is the difference between doFilter(-,-,-) of javax.servlet.Filter interface and doFilter(-,-) method of javax.servlet.FilterChain interface?

(A) doFilter(-,-,-) method of Filter interface is life cycle method of servlet Filter program so programmer uses this method to place prerequest processing and post response generation logics.

doFilter(-,-) method of FilterChain is not life cycle method so programmer uses this method to invoke next filter in the chain or to invoke the mapped servlet program or JSP program of current servlet filter program.

Sample web application

Srv1, Srv2, Srv3 are servlet programs

F1, F2 are servlet filter programs

To configure F1, F2 filter programs with Srv1 program

url pattern of Srv1 program : /s1url

url pattern of F1 filter program : /s1url

url pattern of F2 filter program : /s1url

To configure F1, F2 filter programs with Srv1, Srv2 programs

url pattern of Srv1 program : /x/y/s1url

url pattern of Srv2 program : /x/y/s2url

url pattern of F1 filter program : /x/y/*

url pattern of F2 filter program : /x/y/*

To configure F1, F2 filter programs with all servlet programs of webapplication

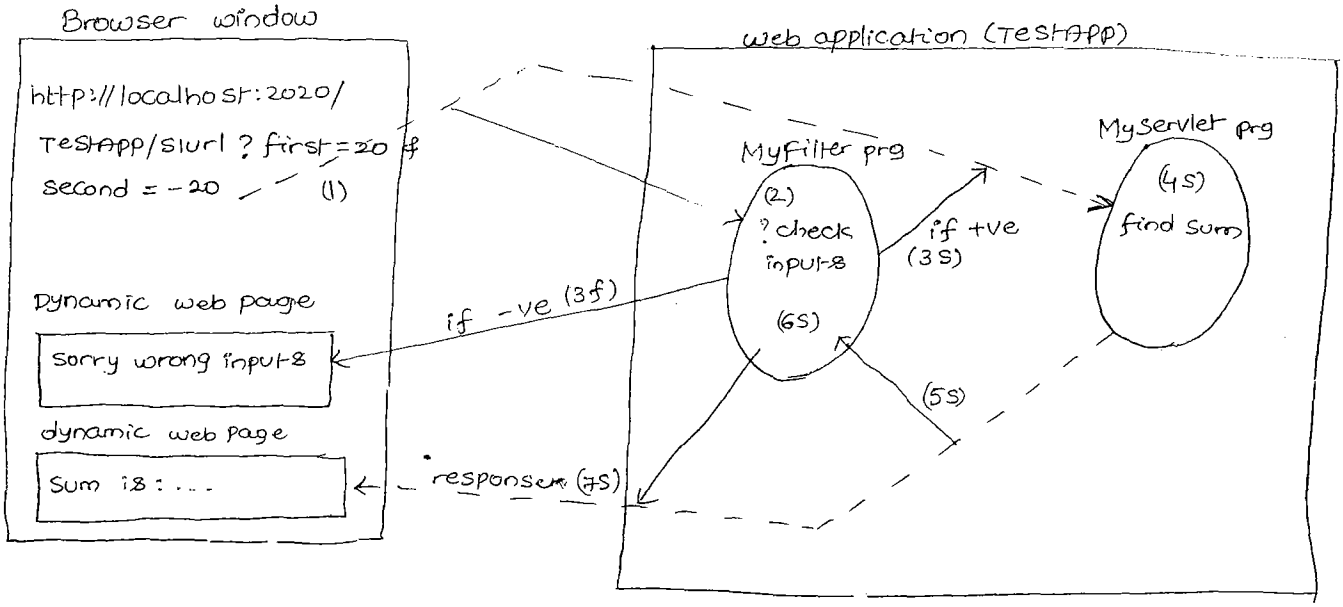
url pattern of ~~Srv~~ F1 filter program : /*

url pattern of F2 filter program : /*

* when filter is configured with servlet program then filter can trap the request and response of servlet program.

Example application

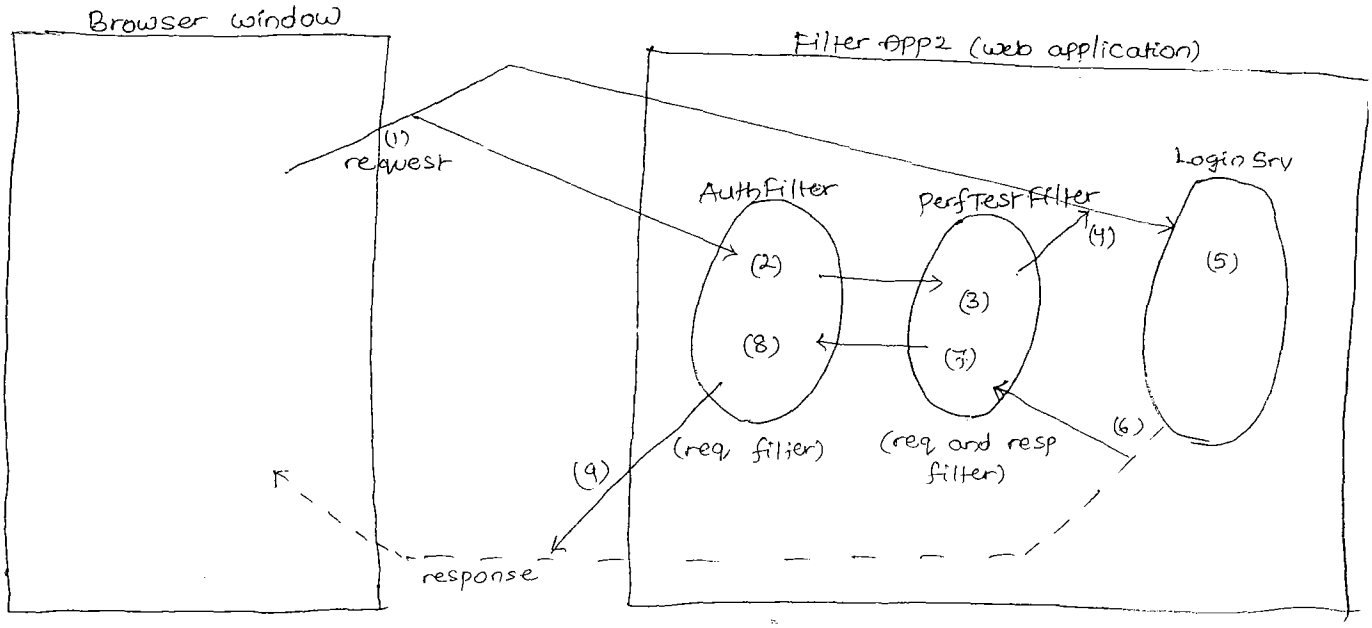
* In the below diagram MyFilter program traps the request going to myServlet program and checks the input values (request parameter values). If those values are positive then it forwards the request to MyServlet program (3s) otherwise it blocks the request in Filter program and sends error response to browser window (3f).



/slurl is the url pattern of myServle Program and MyFilter program.

* In the above diagram MyFilter also traps the response of MyServlet prog but does not contain post response generation logic so the above filter is called as request filter.
* For above diagram based application refer application (16) of page nos (89)

<u>Example application on filter</u>



* After taking request from client the amount of time taken by servlet progr to process the request is technically called as response time of servlet.

* The tomcat server maintains log files in Tomcat_home \ logs folder on one per day basis having date in the file name.
* Instead of using S.O.P (-) statements which writes log messages to server console it is recommanded to use log (-) method of ServletContext object which writes the log messages to current day's log file of <tomcat_home>\ log folder.

```
ServletContext sc = getServletContext();
sc.log("hello");
```

* JDBC code to perform username and password based authentication

```
PreparedStatement PS = Con.prepareStatement("select * from userlist where
                                    username = ? and password = ?");

ps.setString(1, "raja");
ps.setString(2,"hyd");
ResultSet rs = PS.executeQuery();
if (rs.next())
  {
     ===   //Authentication successful
  }
else
  {
     ===   //Authentication failed
  }
```

Userlist (db table)

username	password
raja	hyd
ravi	Vizag
ramesh	Delhi

* For above diagram based web application's source code refer application (17) of the page nos 90-93.
* The process of checking the identity of a user through his username and password is called as authentication.
* The process of verifying the access permissions of a user on particular resource is called as authorization.

To count no. of requests (hits) coming to web application when webappli cation is in active mode we can develop filter program as shown below.

Step(1): Develop the servlet filter program as shown below in WEB-INF classes folder of web application (like URLAPP) having the logic of coun request.

```
//CountFilter.java
import javax.servlet.*;
import javax.servlet.http.*;
public class CountFilter implements Filter
  {  FilterConfig fg;
     public void init(FilterConfig fg)
       { this.fg=fg;
       } int cnt=0;
     public void doFilter(ServletRequest req, ServletResponse res, FilterChain
                                                  throws ServletException, IOExceptio
       {
          cnt++;
          ServletContext sc=fg.getServletContext();
          sc.setAttribute("hitcnt",cnt);
          fc.doFilter(req,res);
       }
     public void destroy()
       {
       }
  }//class
```

Step(2): Configure above filter in web.xml file with url pattern "/*"

Step(3): Read and display servletContext attribute value in every web resour program of web application.

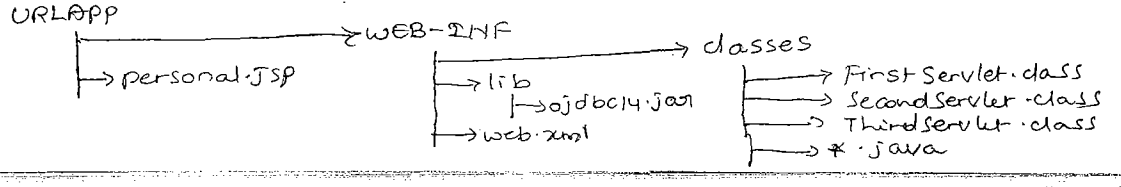
In personal.jsp

```
<% out.println("Request Count:" +application.getAttribute("hitcnt")); %>
```

In FirstServlet.java, SecondServlet.java, ThirdServlet.java

```
ServletContext sc= getServletContext();
pw.println("<br>Request Count" +sc.getAttribute("hitcnt"));
```

Deployment directory structure



Request url

http://localhost:2020/URLAPP/personal.jsp

* Use application (IS) as base application to develop this application.

Applet to servlet communication

* In web applications we need form pages to submit the request to web resource programs by having enduser supplied data. These form pages provide Graphical user interface to end user to submit the request to web resource programs having data. There are two ways to develop form pages.
  1) As html form page (using <form>, <input> and etc... tags)
  2) As java applet based form page.
* there are two types of applets
  (a) Trusted applet
  (b) untrusted applet
* An applet is a Compiled java class that can be sent over the network as web page.
* when untrusted applet is launched in the browser window by gathering it from network it can't interact with the files of file system so untrusted applets can not write virus to the Computer (that means provides security)
* Trusted applets that are launched in the browser window can interact with file system and can write virus to file system (that means no security).
* while designing form page if performance is important (should be launched very fastly) then use html form pages. If security is important that take untrusted applet as form page.

What are the differences between Applet and servlet?

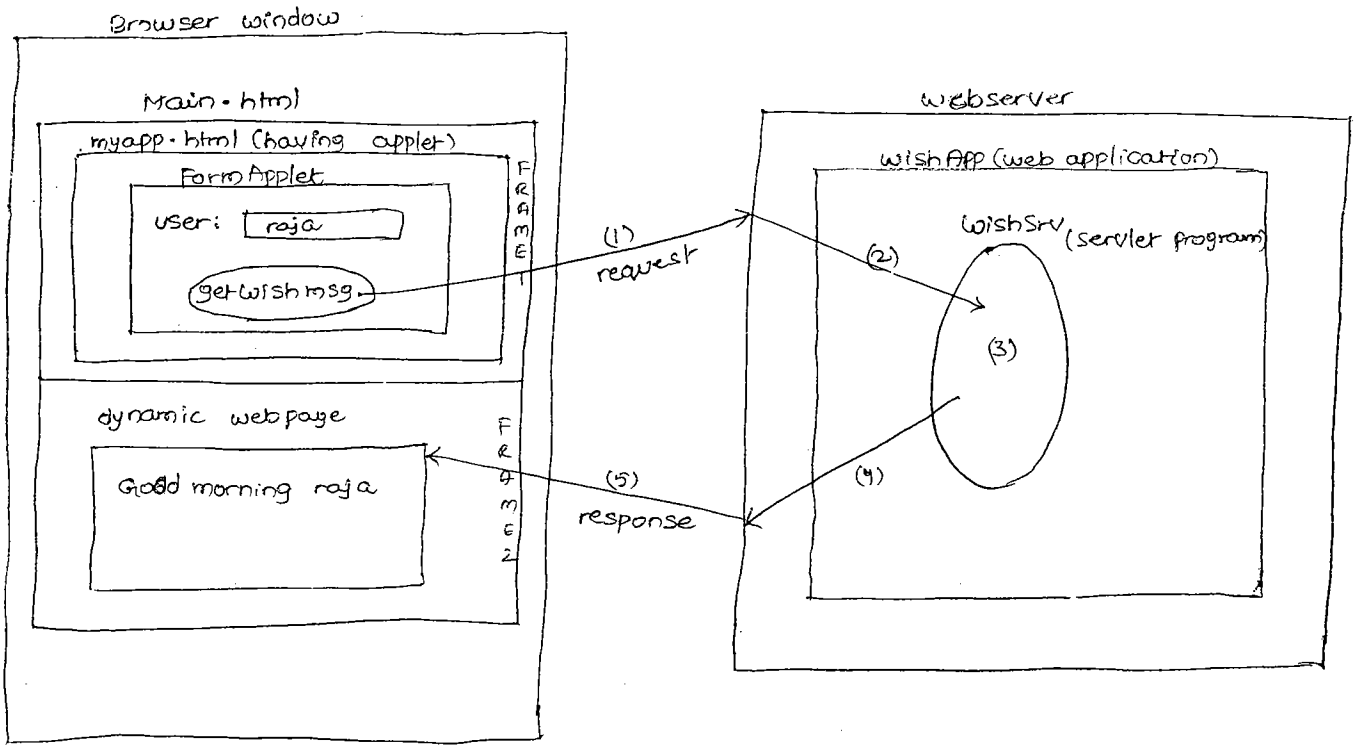
Applet	Servlet
* client side technology to develop client side web resource programs.	* server side technology to develop server side web resource programs
* Comes to browser window for execution.	* uses servletContainer of webserver for execution.
* Generate static form pages.	* Generate dynamic web pages & form pages
* Need not be configured in web.xml file.	* must be configured in web.xml file.

* Uses life cycle methods for execution. They are init(), start(), stop(), destroy(), paint()

* Executes through life cycle methods. They are init(), service(), destroy().

* Applet to servlet communication means making the applet based form page sending request to servlet program having data.

* In html form page to servlet communication the html tags automatically prepares request URL by having form data as query string values where in applet to servlet communication all these operations should be performed by the programmer manually



wishApp
- → WEB-INF
  - → classes
    - → WishSrv.java
    - → WishSrv.class
  - → web.xml
- → main.html
- → myapp.html
- → FormApplet.java
- → FormApplet.class

jar files in classpath

servlet-api.jar

wishsrv.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
public class wishsrv extends HttpServlet
{
    public void service(HttpServletRequest req, HttpServletResponse res) throws
                                                ServletException, IOException.
    {
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        //read form data (applet data)
        String uname = req.getParameter("user");
        //write b.logic
        Calendar cl = Calendar.getInstance();  //gives current date and time.
        int h = cl.get(Calendar.HOUR_OF_DAY); //gives current ~~day~~ hour of day
        if (h<12)
            pw.println(" God morning" +uname);
        else if (h<=16)
            pw.println(" Good afternoon" +uname);
        else if (h<=20)
            pw.println("Good evening" +uname);
        else
            pw.println("Good night" +uname);
        //close the stream object
        pw.close();
    }
}
```

main.html

```
<frameset rows="40%,*">
    <frame name="f1" src="myapp.html"/>
    <frame name="f2"/>
</frameset>
```

* frame with name is called as named frame.

myapp.html

```
<applet code="FormApplet.class" width="200" height="200">
</applet>
```

FormApplet.java

```
import java.applet.*;
import java.awt.*;
import java.awt.event.*;
import java.net.*;
public class FormApplet extends Applet implements ActionListener
  {
      Label l1;
      TextField tf1;
      Button b1;
      public void init()
       {
          l1 = new Label("User:");
          add(l1);
          tf1 = new TextField(10);
          add(tf1);
          b1 = new Button("Get wish");
          b1.addActionListener(this);
          add(b1);
       }
      public void actionPerformed(ActionEvent ae)
       {
        try{
            //example req url : http://localhost:2020/wishApp/wurl?user=raja++
            String qrystr = "?user="+tf1.getText().replace(' ','+'); //gives ?user raja+++
            String requrl = "http://localhost:2020/wisApp/wurl"+qrystr;
            //prepare req url as URL class obj.
            URL url = new URL(requrl);
            //get Access to AppletContext obj
            AppletContext apc = getAppletContext(); //public method of Applet class
            //send request to servlet program
            apc.showDocument(url,"f2"); // sends the req to wishSrv prg and
                                        //displays the generated webpage in
                                        //a frame window whose name is "f2
            }
        catch(Exception e)
          {
           e.printStackTrace();
          }
       }
  }
```

web.xml

```
<web-app>
    <servlet>
        <s-n> xyz </s-n>
        <s-c> wishSrv </s-c>
    </servlet>
```

```
<s-m>
    <s-n> xyz </s-n>
    <u-p> /wurl </u-p>
</s-m>
</web-app>
```

* use following request url to test the application

  http://localhost:2020/wisApp/main.html

* AppletContext object represents the current executing environment of applet programming so it can be used to make current applet talking with another applet or another web resource program of web application. To get this AppletContext object we can use getAppletContext() predefined java.applet.Applet class

File downloading :

* selecting file from the client machine file system and sending that file to server machine file system through network is called as file uploading and reverse is called as file downloading.
* All the files of a computer managed by operating system together is called as file system of computer.
* servlet api gives built-in support for file uploading. since it is complex we generally prefer working with third party api called java zoom api perform this file uploading.
* java zoom api comes in the form of jar files and we can download them from www.javazoom.net website. The jar files are

  uploadbean.jar (main jar file)

  struts.jar, cos.jar } dependent jar files to uploadbean.jar

* The classes and interfaces of uploadbean.jar files uses the classes and interfaces of struts.jar file, cos.jar file.
* If java web application uses third party api (other than jdk apis, jee apis) then add 3rd party api related main jar files to classpath and add the third party api related main and dependent jar files to WEB-INF\lib folder of web application.

Ex: If servlet, JSP programs of java web application uses third party api called java zoom api then add the java zoom api related main jar file uploadbean.jar file to classpath. similarly the java zoom api related main and dependent jar files uploadbean.jar, struts.jar, cos.jar to WEB INF\lib folder of web application.

* To select a file from client machine file system the form page of client machine web application should have file uploading component. For that we take <input> tag as shown below.

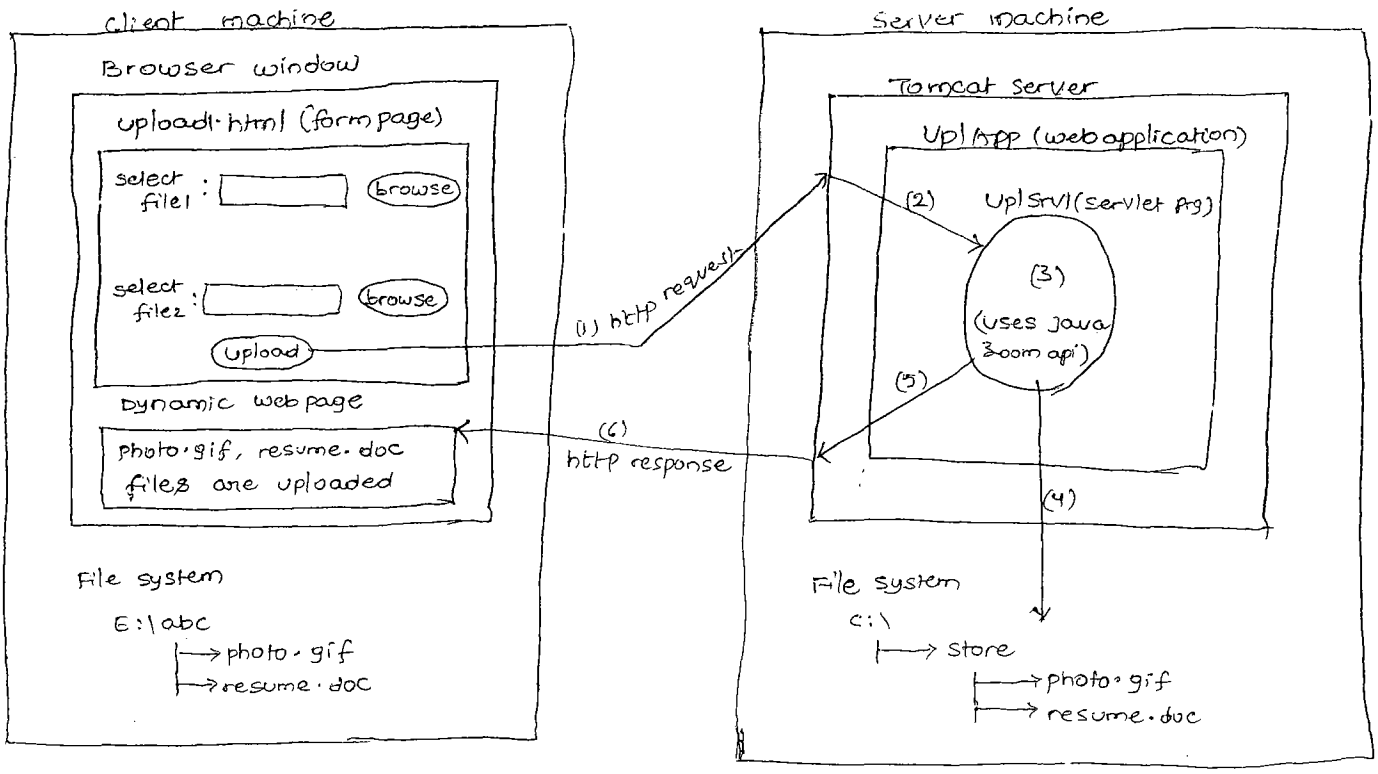
In form page

select file: <input type="file" name="f1"/>

select file: [ ] Browse

file uploading component

* The form page that contains file uploading component is recommanded to generate "post" method based http request.



* For the source code of above application refer application (18) that is given in page nos 93 and 94 of booklet.

* we can see file uploading and downloading applications in email account applications (to send + receive attachments), job portal websites like naukari.com (Job seeker uploads the resume, HR dept. of company downloads the resume). and matrymoni website
* we can perform file downloading operations in two modes.
  (1) making the response of web resource program as downloadable file
  (2) making downloading resources of the web application (img file, audio file, video file and etc...) as downloadable file through hyperlinks (or) buttons.
* write the following two lines of content in web resource program like servlet, JSP to make that response of that web resource program as downloadable file.

```
res.addHeader("Content-Disposition", "attachment; filename=abc.html");
res.setContentType("text/html");
```

response header

response content type.

name of the downloadable file that holds response of the web resource program

* An action performed on java object or Component is called as an Event.
* Executing some logics when event is raised is called as Event handling.
* To perform event handling we use event Listeners.
* The reusable java object is called as Component. we use AWT, Swing concepts of JSE module to perform event handling.
* To perform event handling we need four important details.
  (1) Source object on which the event will be raised (like Button Component)
  (2) Event class name (like java.awt.event.ActionEvent)
  (3) Event Listener (like java.awt.event.ActionListener)
  (4) Event handling method (like public void actionPerformed(-) {---})
* From Servlet api 2.4 onwards we can perform event handling on request, session and ServletContext object through servlet listeners.
* Event handling on request object allows us to keep track of the creation and destruction of request object and we can use these timings to calculate request processing time of each request. This event handling also allows us to keep track of when each request attribute is created, modified and removed.

Details that are required to perform event handling on request object

source object : request object
Event class : javax.servlet.ServletRequestEvent
Event listener : java.servlet.ServletRequestListener
Event handling methods : (1) requestInitialized(-)
(2) requestDestroyed(-)

For Event handling on request object

source object : request object
Event class : javax.servlet.servletRequestAttributeEvent
Event Listener : java.servlet.ServletRequestAttributeListener
Event handling methods : (1) attributeAdded(-)
(2) attributeRemoved(-)
(3) attributeReplaced(-)

For event handling o request attributes.

* Event handling on ServletContext object helps us to keep track of when ServletContext object is created and destroyed. Based on this we can keep track of timings of web application deployment and undeployment or reloadi or stop operations.
* This Event handling can also be used to keep track of creation, modificatio destruction of ServletContext attributes.

Details that are required to perform eventhandling on ServletContext object

source object : ServletContext object
Event class : javax.servlet.ServletContextEvent
Listeners : javax.servlet.ServletContextListener (I)
Event handling methods : (1) ContextInitialized(-)
(2) ContextDestroyed(-)

For event handling on ServletContext object.

source object : ServletContext object
Event class : java.servlet.ServletContextAttributeEvent
Listener : javax.servlet.ServletContextAttributeListener (I)
methods : (1) attributeAdded(-)
(2) attributeRemoved(-)
(3) attributeReplaced(-)

For event handling ServletContext attribu

* Event handling on session object can keep track of when HttpSession object is created and destroyed. Using this we can know how much time ~~there~~ the

user is there in the session. we can also use this event listener to keep track of when each session attribute is created or modified or destroyed.

Details that are required to perform event handling on HttpSession object

source object : HttpSession object
Event class : javax.servlet.http.HttpSessionEvent
Listener : javax.servlet.http.HttpSessionListener
event handling methods : (1) sessionCreated(-)
(2) sessionDestroyed(-)

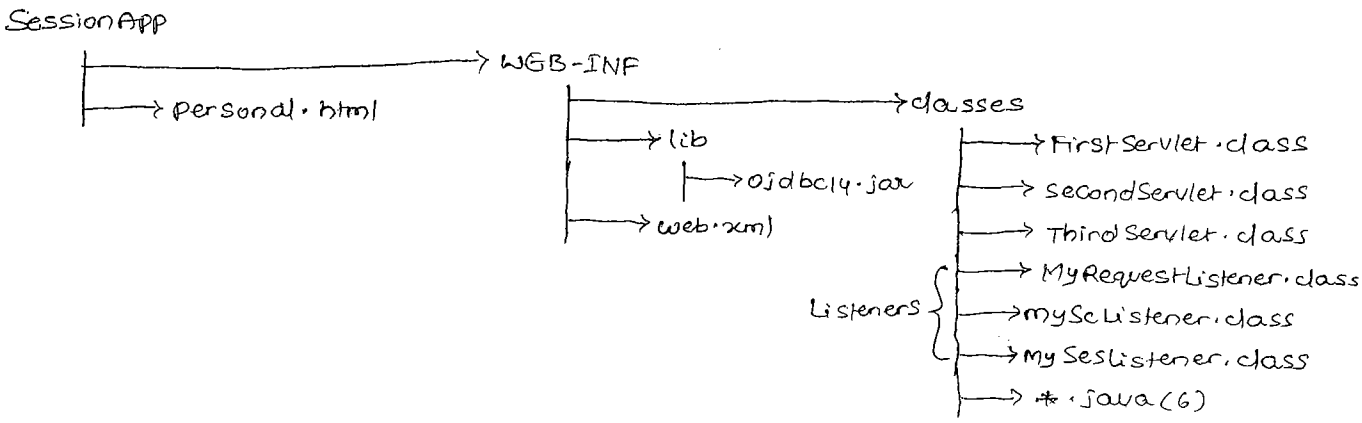
} For event handling on HttpSession object

source object : HttpSession object
Event class : javax.servlet.http.HttpSessionBindingEvent
Listener : javax.servlet.http.HttpSessionAttributeListener
event handling methods : attributeAdded(-)
attributeRemoved(-)
attributeReplaced(-)

} For event handling on HttpSession attribute.

* Through servlet listeners we can perform event handling on request, servlet context, session objects being from outside the servlet, JSP programs.

Example application (with respect to SessionApp application)



Every ServletListener class must be configured in web.xml file using <listener>, <listener-class> tags.

Step(1): Add listener classes to WEB-INF/classes folder of web application as shown above.

```
// MyReqlistener.java
  javax.servlet.*;
import javax.servlet.http.*;
```

```
public class MyReqListener implements ServletRequestListener
{ Long sttime, endtime;
  public void requestInitialized(ServletRequestEvent sre) //executes when
                                                          //req obj is created
    {
      sttime = System.currentTimeMillis();
    }
    //executes when req obj is destroyed.
    public void requestDestroyed(ServletRequestEvent sre)
    {
      endtime = System.currentTimeMillis();
      ServletContext sc = sre.getServletContext();
      sc.log("Request is processed for " + (endtime-sttime) + "ms");
      System.out.println("request is processed for" + (endtime-sttime) + "ms")
    }
}
```

MyScListener.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class MyScListener implements ServletContextListener
{
  Long sttime=0;
  long endtime=0;
  public void contextInitialized(ServletContextEvent sce)
  {
    sttime=System.currentTimeMillis(); ServletContext sc=sce.getServletContext
    sc.log("web application is deployed at" + new Date());
  }
  public void contextDestroyed(ServletContextEvent sce)
  {
    endtime = System.currentTimeMillis();
    sc.log("web application is undeployed/reloaded/stopped at" + new Date());
    sc.log("web application is there in running continuosly for" + (endtime-sta
    S.O.P("web application is there in running continuously for" + (endtime-sttim
  }
}
```

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
import java.io.*;
public class MySesListener implements HttpSessionListener
{ long sttime, endtime;
  public void sessionCreated(HttpSessionEvent se)
  {
      sttime = System.currentTimeMillis();
  }
  //execute when HttpSession obj is destroyed
  public void sessionDestroyed (HttpSessionEvent se)
  {
      endtime = System.currentTimeMillis();
      HttpSession ses = se.getServletContext(); getSession();
      //get access to servletContext obj
      ServletContext sc = ses.getServletContext();
      sc.log (ses.getAttribute("name") + " is there in the session for" +
                                        (endtime-sttime) + "ms");
      s.o.p (ses.getAttribute("name") + " is there in the session for" +
                                        (endtime-sttime) + "ms");
  }
}
```

step (2): compile all the listener classes.

```
>javac *.java
```

step (3): Configure all the three listener classes in web.xml file.

In web.xml

```
<listener>
    <listener-class> MyReqListener </listener-class>
</listener>
<listener>
    <listener-class> MySesListener </listener-class>
</listener>
<listener>
    <listener-class> mysclistener </listener-class>
</listener>
```

Step (4): Test the application and also observe the log file (D:\Tomcat logs\ localhost.2011-12-04 file)
Current Date

* In our java web applications no servlet listener acts as default listener. That means every listener configuration is mandatory in web.file.
* servlet listeners are useful to keep track of various operations related to request, session, ServletContext objects without disturbing the existing code of web application.
* To make the java class as servlet listener class the class must implement XxxListener interface.

Procedure to Configure Jboss 5.x servers with NetBeans IDE

Step (1):

Tools → Servers → add server → Jboss application server → next →
Server location: E:\Jboss5.x Soft\Jboss-5.1.0.GA → next →
Domain: default → finish → close

* In servlet programming we can use servletOutputStream class object pointing to response object to write the o/p of the servlet program to browser window as web page through webserver and it is alternate for printwriter stream object.

Ex:
```
res.GetContentType("text/html");
ServletOutputStream sos = res.getOutputStream();
sos.println("welcome to servlets");
sos.println("hello");
```

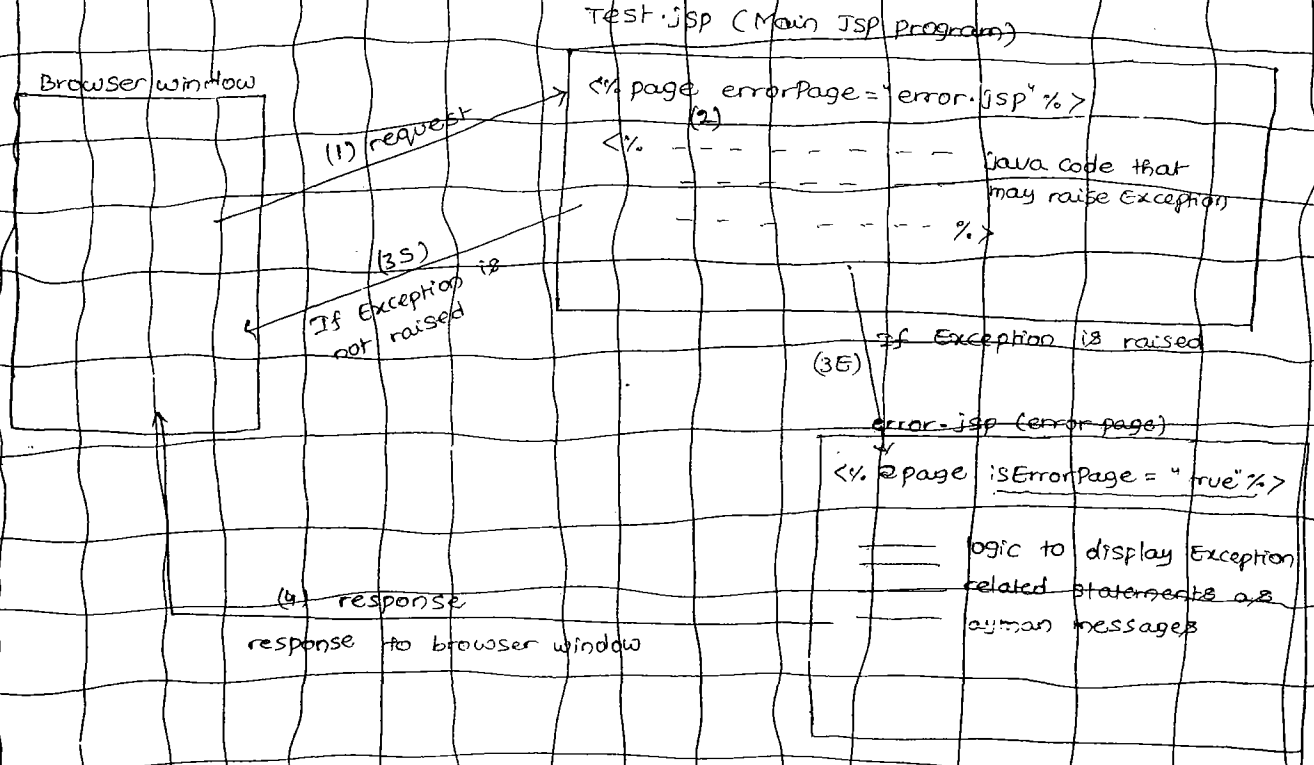
* In one servlet program we can not use both stream objects (ServletOutputStream, PrintWriter) at a time pointing to res object.
* PrintWriter is a character stream available in java.io package and this stream is good when servlet program generates more character data and less binary data to write to browser window.
* ServletOutputStream is byte stream supplied by servlet api and this stream is good when servlet program generates more binary data an

less text data to write to browser window.

* For better performance of servlet and to reduce one method call use pw.print() method in servlet program instead of pw.println() method because this pw.println() method internally calls pw.print() method to complete the requirement.
* We can also apply this recommendation while working with ServletOutputStream.

Local Error page configuration

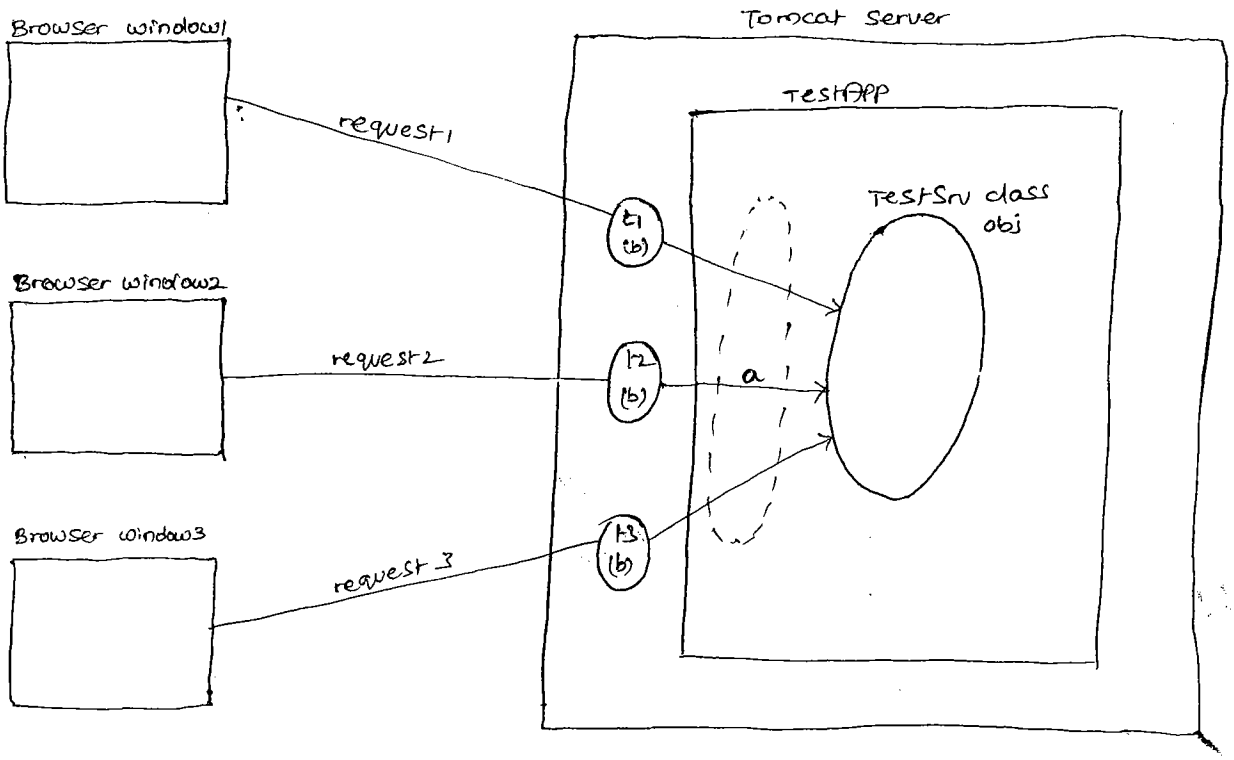
JSP Concept Not related to servlets


Test.jsp (Main JSP program)
Browser window
(1) request
<% page errorPage="error.jsp" %>
(2)
<% - - -

Thread safety

* If multiple threads are running on single object or variable simultenco or concurrently then data of the object may corrupt. This says our object is not thread safe object.
* To make object/ variable as thread safe apply lock on object or variable through synchronization and allow only one thread at a time to manipulate object/ variable data. The instance variables of our servlet class are not thread safe by default where as the local variables declared in service(-,-) or doXxx(-,-) methods of our servlet class are thread safe by default.

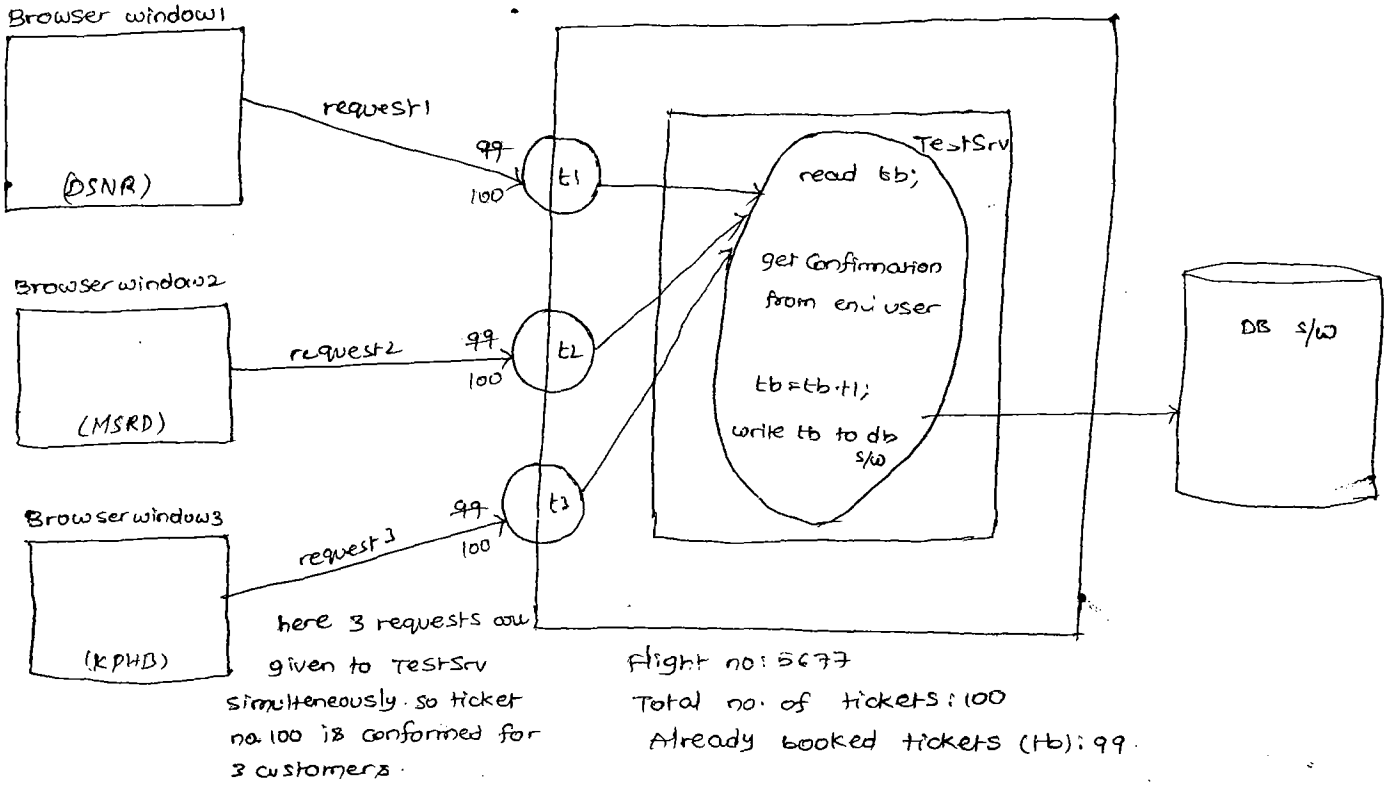
```
public class TestSrv extends HS/GS
{
    int a;
    public void service(-,-)/doXxx(-,-) throws SE,IOE
    {
        int b;
        ---
    }
}
```



* In the above diagram all the three threads started on TestSrv class ob will act on single copy of instance variable 'a' but every thread gets its own copy of local variable 'b'.

* By default every servlet program is a single instance multiple threads component so by default no servlet program is thread safe.
* Making servlet program as thread safe program is nothing but making the instance variables of servlet class as thread safe through synchronizations and other concepts.

<u>understanding the problems related to working with non thread safe servlets (regular servlets)</u>



* Simultaneous request or concurrent request in servlet execution is nothing but servlet program is allowing another request from clients to execute the logic before completition of existing request processing. In such situations servlet program may generate wrong results like booking same ticket number for multiple pessangers as show in the above diagram.

* To make our servlet program as thread safe servlet program we can use one of the following four techniques.

(1) work with only local variables of service(--)/doXxx(--) methods

(2) work with synchronized service(--)/doXxx(--) methods

(3) work with synchronized blocks in service(--)/doXxx(--) methods

(4) make your servlet class implementing javax.servlet.SingleThreadModel (I).

(1) Working with only local variables

```
public class TestSrv extends HS/GS
{
    ——— no instance variables

    public void service(-,-)/doXxx(-,-) throws SE, IOE
    {
        int a,b,c;
        Connection con;
        ———————— //write entire using local variables
    }
}
```

* Local variables of service(-,-)/doXxx(-,-) methods are thread safe variables by default. So the above servlet becomes thread safe servlet. Because it does not have instance variables and working with just local variables.

NOTE: This technique is not recommended to use because it is practically impossible to develop servlet classes without instance variables.

(2) Working with synchronized service(-,-)/doXxx(-,-)

```
public class TestSrv extends HS/GS
{
    Connection con;
    int a,b,c;          } Instance variables

    public synchronized void service(-,-)/doXxx(-,-)
    {
        ———————
        ——— //write logic using instance variables
        ———————
    }
}
```

* All requests that are given to servlet starts threads on our servlet class object and executes service(-,-)/doXxx(-,-).
* In the above code multiple threads started on our servlet class object will act on single copy of instance variables but only one thread is allowed at a time to act on those instance variables because the service(-,-) method is taken as synchronized method.
* Instead of making whole service(-,-) as synchronized method it is recommended to take the support of synchronized blocks on special objects on which you are looking for thread safety.

(3) working with synchronised blocks in service(-,-)/doXxx(-,-)

```
public class TestSrv extends GS/HS
 {
    Connection con;
    int a,b,c;
    public void service(-,-)/doXxx(-,-)
     {
        ____
        ____
        ____
        synchronized(con obj)
         {
            ____
            ____  logic related to con obj
            ____
         }
        synchronized(Session obj)
         {
            ____  logic related to HttpSession obj
            ____  (creating, modifying, reading and writing, moving attribute values)
            ____
         }
        synchronized(ServletContext obj)
         {
            ____  logic related to ServletContext obj
            ____  (creating, modifying, reading and moving attribute values)
            ____
         }
     }//
 }
```

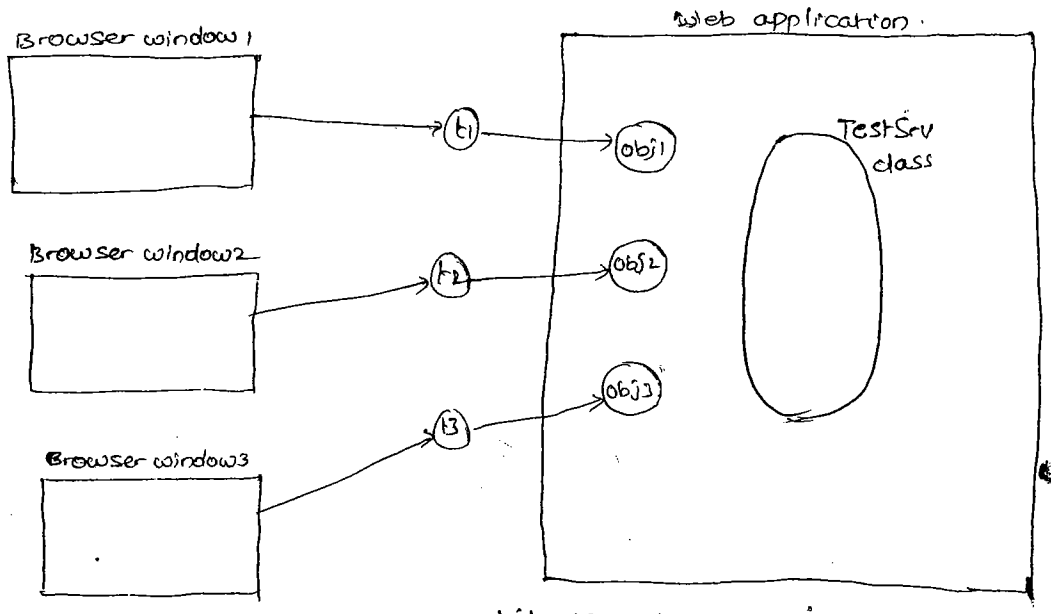
* working with these synchronized blocks is the most popular technique of real world to make servlet programs as thread safe.
* Since HttpSession object and ServletContext objects are sharable objects in multiple web resource programs of web application it is recommended to use them in synchronized blocks as shown above.

(4) Making our servlet class implementing javax.servlet.SingleThreadModel (I)

```
public class TestSrv extends HttpServlet implements SingleThreadModel
 {
    int a,b;
    Connection con;
    public void service(-,-)/doXxx(-,-)
     {
        ____
        ____
        ____
     }
 }
```

* By seeing the implementation of SingleThreadModel interface the underlayin Server automatically makes our servlet as thread safe. But the technique of thread safety that is used by server will vary server to server.

* This Interface has been depricated from servlet api 2.4 because differ servers are using different mechanisms while implementing this interfac and to make the servlet as thread safe some servers are even creatin multiple objects for servlet class for multiple requests on one per request basis. This is violation of servlet specification that says a servlet is a single instance multiple threads component.



obj1, obj2, obj3 are the objects of TestSrv class

* Every JSP program is not thread safe by default because the JSP equalent servlet is not automatically implements single Thread model interface and not automatically generates the necessary synchronised blocks in _jspService(--,--) method.

* To make JSP program as thread safe program use <% @ page isThreadSa "false" %> tag in JSP program. The default value of isThreadSafe attribute is "true".

* isThreadSafe="true" makes JSP equalent servlet as non thread safe servlet. where as isThreadSafe="false" makes JSP and its equalent servlet as thread safe.

File downloading

1. Making the output of web resource program as downloadable file (approach1)

2. Making the resource (file) of server machine file system as downloadable file (approach 2)

* In approach1 web resource program will not be downloaded. The output of the web resource program as downloadable file through browser window. place the following ~~the~~ two lines of code in servlet/Jsp program to make its response as downloadable file according to approach (1).

```
response.addHeader("Content-Disposition", "attachment; filename=abc.html");
response.setContentType("text/html")
```

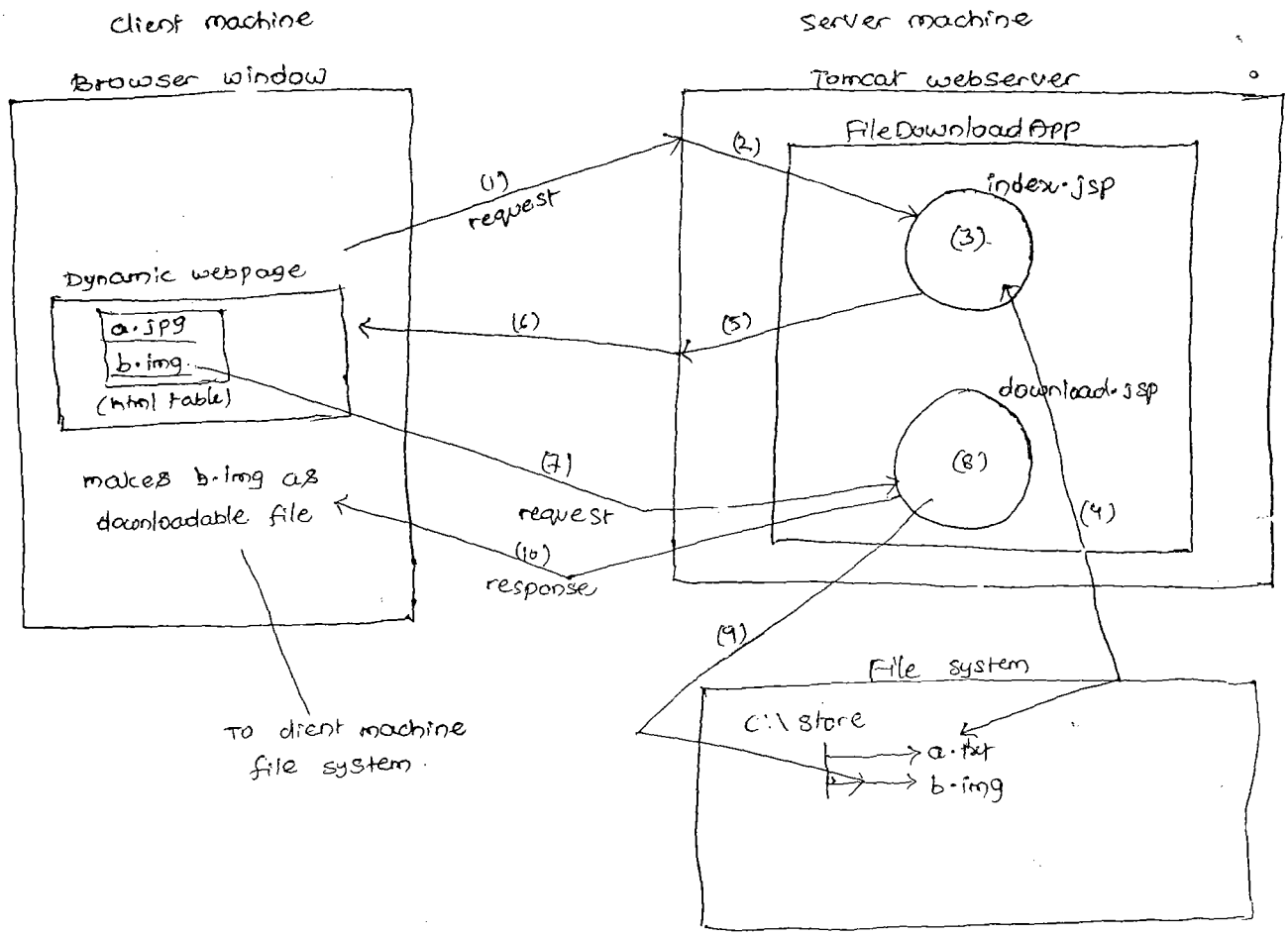
*1

MIME type

filename that holds response of current web resource program.

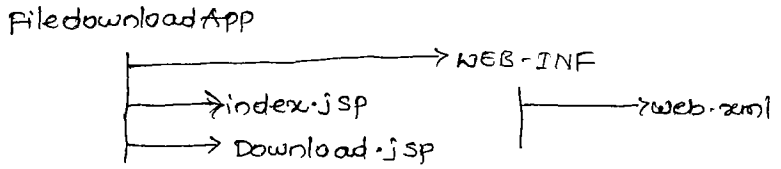
*1 → special response header to guide webserver towards sending the response of the web resource program ~~as~~ to client.

approach (2) based file downloading application



* The above application makes all the files of c:/store folder as downloadable files. Here index.jsp responsibility is to render all the files of c:/store folder as hyperlinks to client and download.jsp responsibility is to download specific file of c:/store folder to client machine file system.
* java.io.File class object can represent a file or directory of file system.
* The listFiles() method of java.io.File class gives all the files and sub directories of certain folder in the form of java.io.File class object array. [File[]]

Deployement directory structure



Request URL to test the application

http://localhost:2020/FileDownloadApp/index.jsp

Source code

index.jsp

```
<%@ Page import="java.io.File, java.util.*" %>
<H1> List of All files under c:\store </H1>
<%
    //locate folder on server machine file system
     File dir = new File ("c:\\store");
     File[] lf = dir.listFiles();  //gives all files and subdirectories of c:\st
     //add all the file names of c:\store to al obj (ArrayList obj)
     ArrayList al = new ArrayList();
     for (int i=0; i<lf.length; i++)
      {
         if (lf[i].isFile())
          {
             al.add (lf[i].getName());
          }//if
      }//for
```

```
//logic to display the file names of c:\store folder as downloadable
                                                    files (hyper links based)
if (al.size()!=0)
{
 %>
    <table border=1> <tr><th><b> Filename</b></th> </tr>

 <%
    String fname=null;
    for (int i=0; i<al.size(); i++)
     {
       fname= (String)al.get(i);
      %>
      <tr>
        <td> <a href="Download.jsp?filename=<%= fname %>">
                        <%= fname %> </a> </td> </tr>
    <%} //for
  }//if
 %>
</table>
```

web.xml

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

download.jsp

```
<%@page import="java.io.*" %>
 <%
   //reading req param value (file name)
   String filename = request.getParameter("filename");
   //locate the file to be downloaded
   File f= new File("c:\\store\\" + filename);
   //get servletOutputStream object
   ServletOutputStream op = response.getOutputStream();
   //get mime type of the file to be downloaded dynamically
   String mimetype = application.getMimeType("c:\\store\\"+ fileName);
   //set the mime type as response content type
   response.setContentType((mimetype != null)?mimetype:" application/octet-
                                                          stream");
```

```
//set response content length
response.setContentLength((int)f.length());
//work with special header to make file as downloadable file.
response.setHeader("Content-Disposition", "attachement; filename="+
                                                    fileName);
//Stream and buffer based logic to download the file
byte[] buf = new byte[1024];
int length=0;
DataInputStream dis=new DataInputStream(new FileInputStream(f))
while((dis!=null) && (length=dis.read(buf))!=-1)
  {
    op.write(buf,0,length);
  }//while
  dis.close();
  op.flush();
  op.close();
%>
```

Security in servlets :

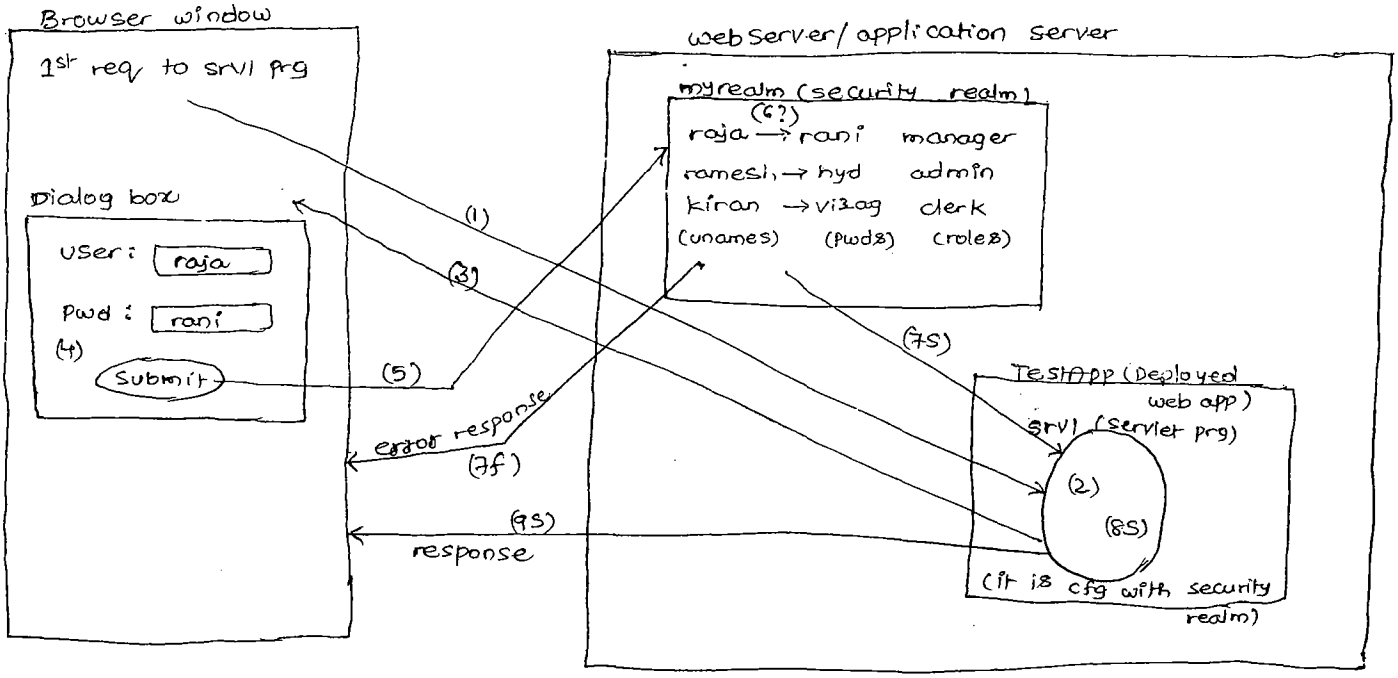
* security = Authentication + Authorization.
* security means authentication + authorization.
* checking the identity of a user is called as authentication.
* checking the access permissions of a user to use certain resources of the application is called as authorization.

Ex: Every employee must be authenticated to use banking project. But an cashier is authorized to use cash module. Adminstrator is authorized to use every module.

* Programmers are not responsible to perform network security. These operations will be taken care by network adminstrator through firewalls.
* Programmers are responsible for application managed security and server/Container managed security in web applications.
* Application managed security means it is the security that is developed by programmer manually inside the application.
* server managed security means it is the authentication and authorization performed by Container on web resource programs by using security middleware

service.

* In server managed security the logical context where users, passwords and roles are defined is called as security realm and every server comes with one default security realm called myrealm.
* In tomcat server this default realm related configurations can be done in <tomcat_home>\conf\tomcat-users.xml file.



srvl servlet program url pattern is /slurl

* A role defines access permissions (set of access permissions) instead of specifying access permission for each user seperately define a role specifying access permissions and assign that role to users.

with respect to diagram

(1) End user gives direct request to srvl program.

(2),(3) Since no user name and password is coming along with the request and since srvl is configured with security realm the srvl sends 401 response status code based response having one dialog box.

(4),(5) End user submits dialog box having username and password and that request will be traped by security realm.

(6?) security realm verifies whether given username, password correct or not against the username and password of realm.

(7s) If correct, request goes to srvl program.

(7f) If not correct ~~error~~ renders same dialog box for two more times then shows 403 status code based error response.

(8S)(9S) SRV1 program executes and generated response goes to browser winc

* we can configure server managed security authentication on our webresourc program in four modes.

(1) BASIC
(2) ~~DESA~~ DIGEST
(3) FORM
(4) CLIENT-CERT

BASIC → uses base 64 algorithm internally.

* does not encrypt given username and password.
* Generates built-in dialog box for end user to submit username and passwor

DIGEST :

* uses MD5 hashing algorithm internally.
* It is stronger than base-64 algorithm.
* Remaining all are same as BASIC, but it encrypts given username and pass

FORM :

* same as BASIC but allows programmer to design his own form page, error page instead of ready made dialog box, ready made errors message.

CLIENT-CERT :

* works with digital certificates using some algorithms like RSA
* use https and SSL concepts to implement this.
* To configure security realm and authentication models in the web resource programs of java web application use various configurations in web.xml file.
* For example code on, BASIC, DIGEST, FORM based authentication refer supplyment hand out given on 25-12-2011.

procedure to create users and roles in default security realm (myrealm) of Tomcat server

Go to <Tomcat-home>\conf\tomcat-users.xml → add these entries.

~~role rolename~~
<user username = "raja1" password = "raja1" roles = "manager"/>
<user username = "raja2" password = "raja2" roles = "admin"/>
<user username = "raja3" password = "raja3" roles = "manager, admin, tomcat">

* Uncomment existing roles and users
* This server managed security can be used as outer layer security for web application even though web application contains application level security. This is very useful in milatory based, NASA based, ISRO based webapplications.
* The client-cert authentication model sends digital certificates to browser window as the response of initial request and that digital certificate will be installed at client side with users permission. Now onwards browser window and server communicates with each other having encrypted data ~~based~~ that is generated the algorithm that is used to generate digital certificate. In this browser window (client) and server should communicate with each other by using HTTPS protocol (HTTP over SSL).
* In CLIENT-CERT authentication model server allows only those clients which send request having digital certificate.
* To make server sending digital certificate to browser as the response of initial request, the programmers must develop and configure digital certificate with server
* In java on built-in tool "keytool" is given to generate digital certificates using different algorithms.

<u>Procedure to create digital certificate, to concate digital certifaicate with server and to work with CLIENT-CERT authentication model using HTTPS Protocol</u>

<u>step(1)</u>: Generate digital certificate using "keytool" by specifying RSA algorithm

```
>keytool -genkey -alias tomcat1 -keyalg RSA
```

↓ any name (logical name of digital certificate

```
Enter key store password: sathya1
Re-enter new password : sathya1
:
:
:
correct?
yes
Enter key password for <tomcat1> : (press Enter key)
```

<u>step(2)</u>: The above steps based digital certificate will be generated in C:\Documents and setting\<u>welcome</u> folder as .key store file.

↓ currently logged in windows username

Step(3): Configure the above generated digital certificate with Tomcat Server by enabl. HTTPS protocol.

Go to <Tomcat-home>\conf\server.xml file → un comment that <Connector> to that points to SSL HTTP/1.1 protocol (line no. 66) → make sure that the <Connector> tag at line 66 (3rd <Connector> tag) is having following content.

```
<Connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
    maxThreads="150" scheme="https" secure="true" clientAuth="false"
    sslProtocol="TLS" keystoreFile="c:\Documents and Settings\welcome E\.keys
    keystorePass="satyal"/>
```

The above generated digital certificate fi

The above choosen password.

Step(4): Restart the server.

Step(5): Give request to tomcat server from browser window.

https://localhost:8443

https://localhost:8443/waOne/input.html

Protocol.

Proto port no. of Http with SSL environment

*For related information on SSL refer the SSL chapter of tomcat documentation.

NOTE: Industry prefers working with either form based or CLIENT-CERT models for securing web applications.

When CLIENT-CERT is enabled we can even also enable Basic or DIGEST or FORM authentication models.