JavaEra.com

A Perfect Place for All Java Resources

Core Java | Servlet | JSP | JDBC | Struts | Hibernate | Spring

Java Projects | FAQ's | Interview Questions | Sample Programs

Certification Stuff | eBooks | Interview Tips | Forums | Java Discussions

For More Java Stuff Visit

www.JavaEra.com

A Perfect Place for All Java Resources

Servlets

(Natraj Notes)

www.JavaEra.com

→ web resource programs generate this webpages

→ there are two types of webresource programs

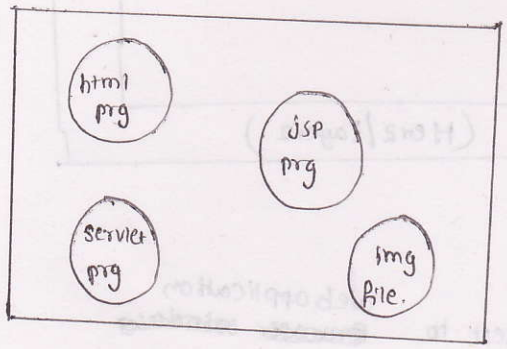
① Static webresource programs : Generates static web pages

Eg: html prgs.

② Dynamic webresource programs : Generates Dynamic webpages

Eg: servlet prg, JSP prg, asp-net prg and etc....

→ A web application is collection of webresource programs.



img file, java script file can't generate webpages directly but they act as helper programs to another web resource prgs.

→ webapplication is a collection of static, dynamic and helper (like img files) webresource programs.

→ To execute java apms we need JRE + JVM

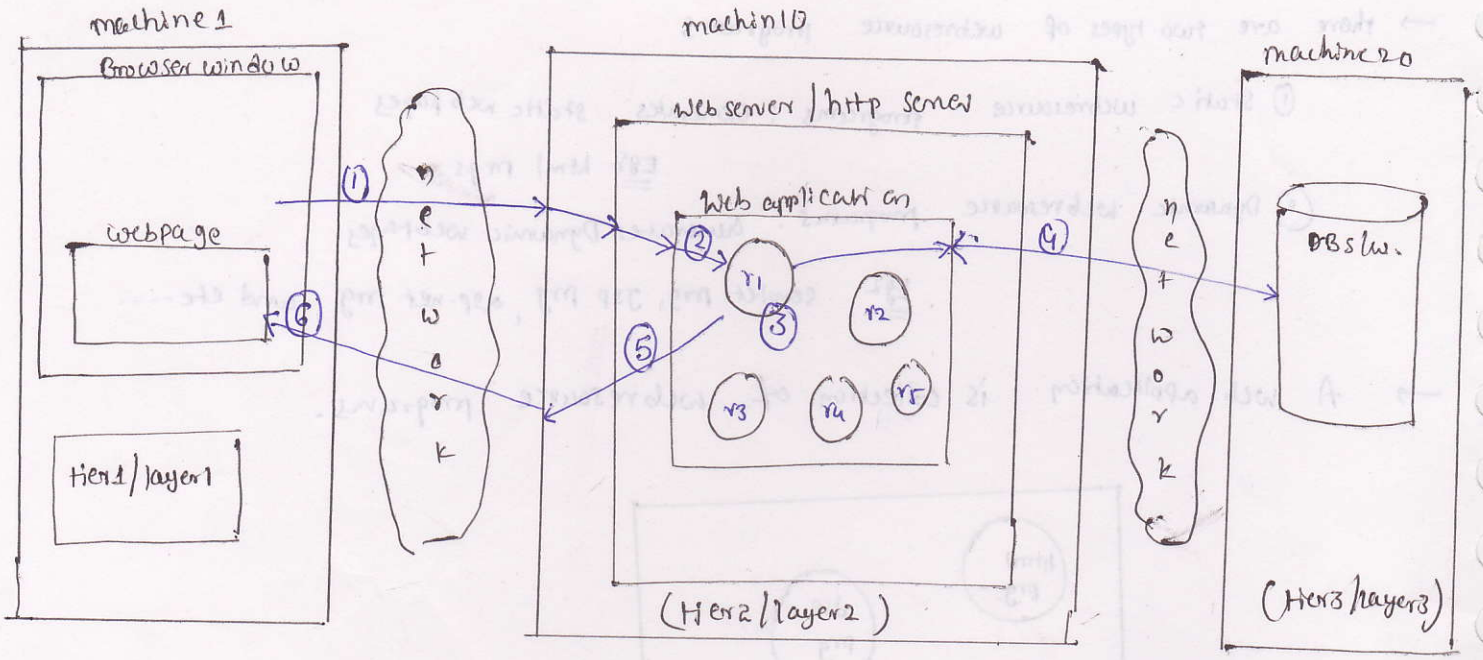
→ To execute Applet programs we need Applet viewer (or) Browser window.

→ To execute Servlet programs we need Servlet Container,

To execute JSP programs we need JSP Container.

→ webserver s/w (like Tomcat) supplies this Servlet, Jsp Containers.

The Development and execution of setup of webapplication.



W.r.to diagram

(1) Browser window generates request to ~~Browser window~~ web application

(2) webserver takes the request and ~~response~~ passes the request to appropriate webresource program of webapplication.

(3) & (4) This webresource program will process the request and generates the results. In this process the webresource program talks with DBS/W if necessary

(5) Web Resource program sends the result to webserver

(6) Web server sends the result to browser window as http ~~reque~~ response in the form of webpage.

NOTE:

Web application canbe developed as two-tier application (without DB) (or) as three tier application (with DB s/w

10/09/2012

I) list of client side Technologies :-

note :- These are given to develope client side web resources.

html, Java script, VB script, AJAX (asynchronous java script and xml)
(W3C) (netscape) (Microsoft) (Adaptive path)

II) list of server side technolgies :-

note :- These are given to develope server side web resource programs.

Servlets ⟶ (sun microsystem (oracle corp)) Javabased (II)
JSP ⟶ (from sun ms (oracle corp)) Javabased (III)
} (I)

php (personal home page for hypertext process) ⟶ (from Apacha foundation) non-java (V)

Cold Fusion ⟶ from open comunity Javabased

Asp (Active Server page) ⟶ from microsoft non-java

Asp.net ⟶ from microsoft non-java (IV)

SSJS (Serverside Java Script) ⟶ from Netscap non-java

III) list of webserver s/ws (http server s/ws) :

Tomcat ⟶ from Apache (java based) (I)

Resin ⟶ from Resinsoft (java) (II)

Jetty ⟶ from Adobe (java)

JWS (Java web server) ⟶ from Sun ms (java)

PWS (personal web server) ⟶ from microsoft (non-java)

AWS (Apache web server) ⟶ from Apache (non-java) (IV)

IIS (Internate Information server) ⟶ from microsoft (non-java) (III)

IV) list of Application Server s/w :- (enhancement of webserver)

Application Server s/w is enhancement of webserver s/w.

Weblogic ⟶ from BEA systems (oracle corp) (I)

Websphere ⟶ from IBM (Internation Business Machine) (III)

JBoss ⟶ from Apache (Redhat)

GlassFish ⟶ from Sun Ms (oracle corp) (II)

JRUN ⟶ from macromedia (Adobe)

Oracle 10gAS ⟶ from Oracle Corp

⋮

NOTE :- all Application Server s/w's are Java Based Server s/w's

(v) List of DB s/w's :-

Oracle — oracle corp ms-access — from microsoft

Sybase — open comunity DB 2 — IBM company

MySQL — Devex (sun ms) (oracle corp)

SQl server — from microsoft

*) To Develop & execute JavaBased web Application

(1) choose any Browser s/w.

(2) choose one (or) more client side Technologies to develop client-side webresource programe

(3) choose any database s/w.

(4) choose Java based web server / Application Server.

(5) choose Javabased server side Technologies. to develop Server Side webresource programs. (servlet, JSP programs)

*) Procedure to Develop microsoft technologies based webApplication :-

(1), (2), (3) steps Same as Java Based web application

(4) use micro soft Server (PWS, IIS)

(5) Use Asp (or) Asp.net to develop ServerSide web Resource programs.

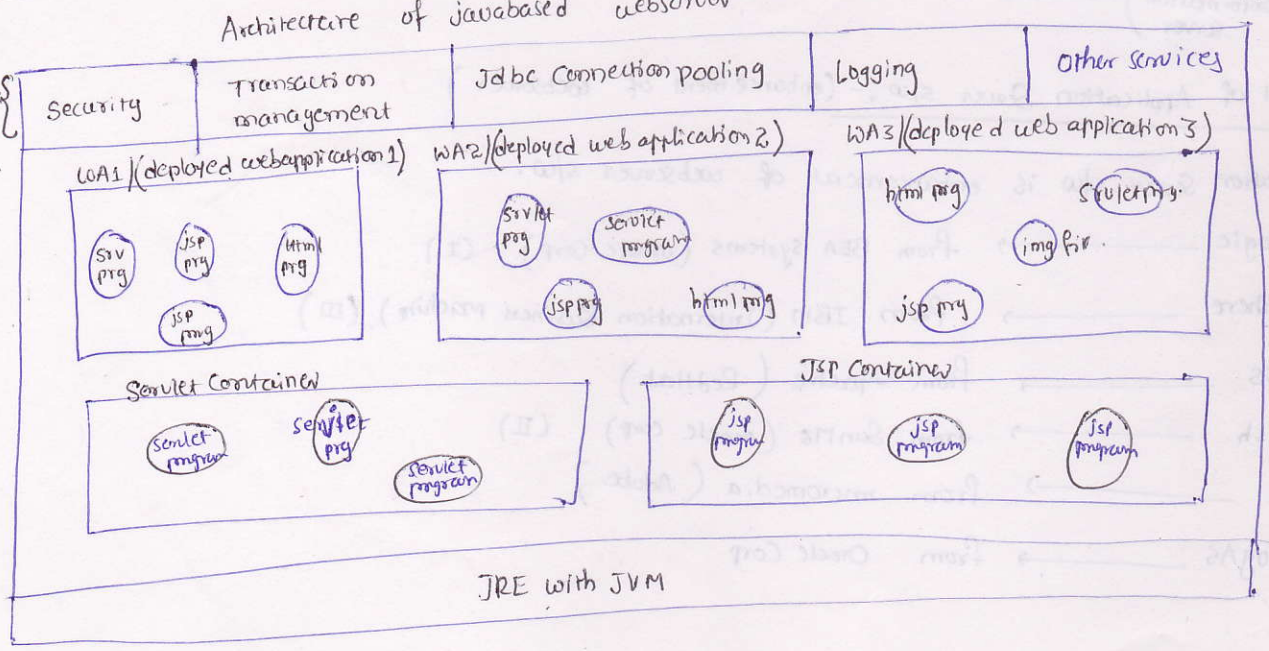
*) procedure to develop and execute PHP based webapplication :-

(1), (2), (3) steps are same as above

(4) Use apache webserver.

(5) Use PHP to develop Server-side webresource programs.

Architecture of webserver (Java based web server) :-



*) Middleware Services are the additional, optional and configural services on the deployed web applications these services makes our applications more perfect and accurate.

→ When Servlet programs are requested they goto Servlet Container for execution.

Similerly from JSP programs are requested they goto JSP Container for execution.

"Container" is a s/w (or) s/w application that can manage the whole lifecycle of (birth to death) given resource.

→ Servlet Container takes care of the whole life cycle of Servlet programing.

→ JSP Container is enhancement of Servlet Container and takes care of the whole lifecycle of JSP program.

The lifecycle operations are

1. Object creation (Instantiation)
2. memory allocation
3. calling lifecycle methods based on the events that are raised
4. execution of resources
5. Destruction of object. and etc....

→ Every Container is like an acquarium taking the responsibilities of fishes servival (given resources mangement and execution).

11/09/2012

Tomcat

- → Type : web server s/w
- → Version : 6.0 (compatable with jdk 1.6 +)
- → Vendor : Apache Foundation
- → Creator : Mr. David DuncanSon
- → Opensource s/w.
- → default port no : 8080 (changable)
- → To download the s/w : www.apache.org
- → for docs and help : www.apache.org
- → name of the Servlet container : CATALINA (catalina.jar)
- → name of the JSP container : JASPER (jasper.jar)

jar → java archile means compressed the code

While installing tomcat 6.0 we need to choose the following :-

① Installation folder

② Admin User name, password

③ Port no : 8080 (to change 3030)

D:\ Tomcat 6.0 (<Tomcat_home>

- conf
  - *.xml
- bin
  - *.exe (tomcat6.exe)
- lib
  - servlet-api.jar (servlet api)
  - Catalina.jar (servlet container)
  - jsp-api.jar (jsp api)
  - jasper.jar (jsp container)
  - *.jar (other jars)
- webapps
  - dir1, dir2, ... (deployed web applications)
  - *.war files (webapplication archive)
- logs
  - *.txt (log files keep in track of server operations)
- work
  - CATALINA
- Other miscellaneous files

To start tomcat web server :-

<Tomcat_home> \bin\ tomcat6.exe file

To see the home page of tomcat :-

open browser window ⟶ Type this url

http://localhost:3030/

host name (or) IP address of computer where tomcat is installed

NOTE :- When tomcat server is started one deamon process will be launch to listen to client request continuously.

Procedure to change port no of tomcat after its installation :-

goto <tomcat_home>\conf\server.xml file and modify the port attribute value of 1st <connection> ⟶ Restart the server.

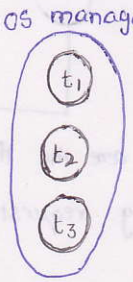
Server Side Web Technologies

1. Process Based
   → CGI

2. Thread Based
   → Servlets
   → jsp
   → asp.net

A light weight subprocess is called Thread.

OS controles processes but Threads can be controlled through Java programing using JRE support



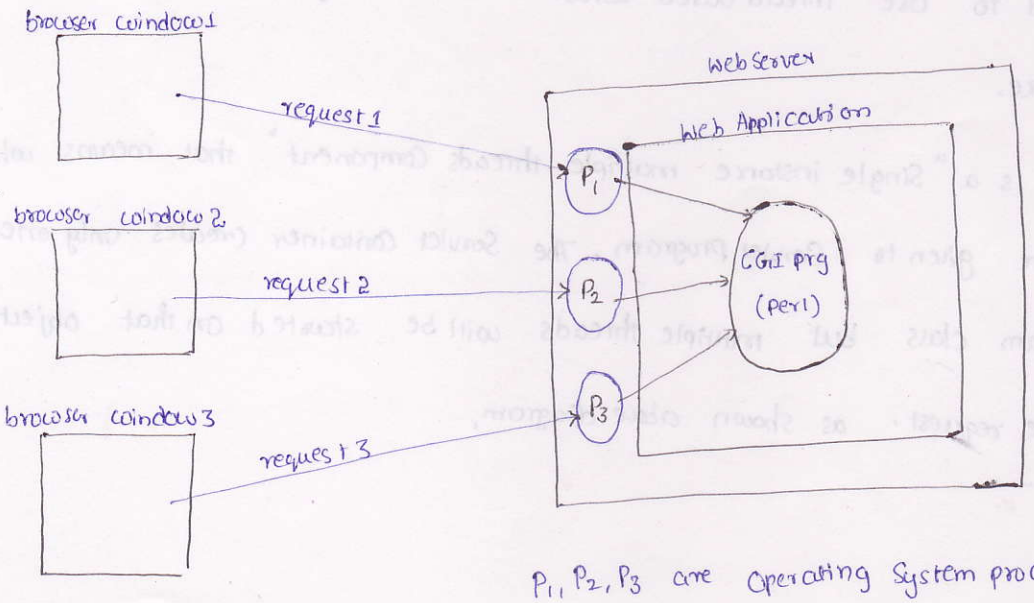
t_1, t_2, t_3 are Threads

→ Transfering control b/w two processes (or) b/w two Threads is called as "Context switch" (or) "control jumping" (or) "sheduling".

→ The sheduling on processes takes more time compare to the sheduling on Threads.

-: Understanding CGI Environment :-

Perl
practicle extraction
Reporting Language

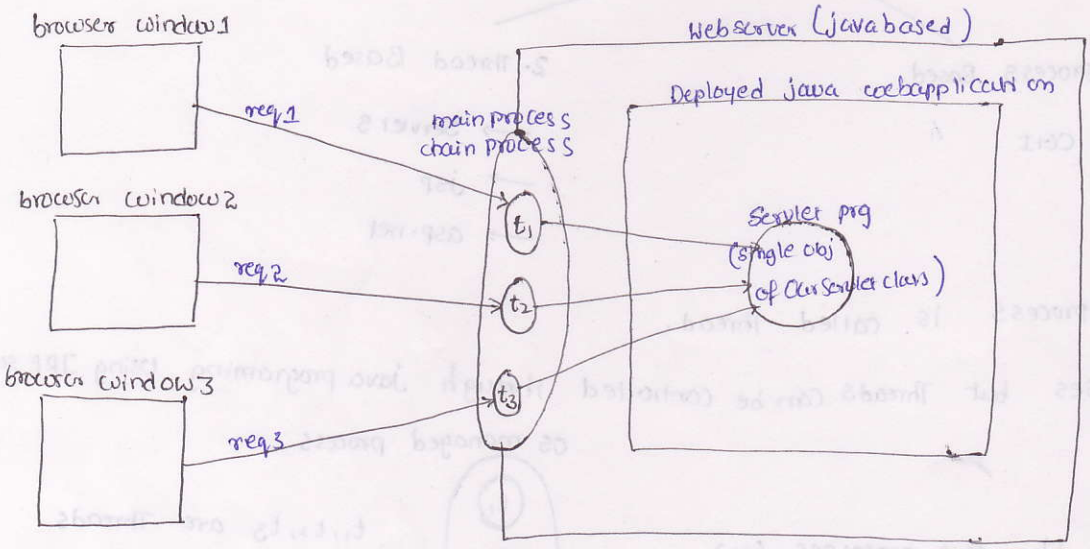


P_1, P_2, P_3 are Operating System processes

→ Since processes based scheduling takes more time the CGI web applications performance will be degraded when multiple requests are given simultaneously. This makes the CGI web applications as non-scalable web applications.

→ If application gives good performance irrespective of increase (or) decrease in request count then that application is called as "Scalable applications".

*) Understanding Servlet / JSP Environment



t_1, t_2, t_3 are the threads started in Main process presenting requests given to Servlet prog.

→ Since scheduling on threads takes less time the above diagram based web application gives good performance in all situations. This makes web applications as "Scalable applications".

→ It is the recomended to use Thread based Server side technologies in web application development for better performance.

→ Every Servlet program is a "Single instance multiple threads Component" that means when multiple requests are given to Servlet Program. The Servlet Container creates only one object for that Servlet program class but multiple threads will be started on that object representing multiple request. as shown above diagram.

12/09/2012

Q: What is servlet ?

Ans: → "Servlet is a Java based server side technology. to extend the functionalities of web server and application server."

→ "Servlets is a Java based Server side web technology. to develop dynamic web resource programs of Java web application having the capability to generate dynamic web pages."

→ "Servlet is an API specification. That will be used by vendor (creators of s/w's) companies as rules and guidelines to develop Servlet Container softwares. and Something will be used by programers as base to develop java based web resource programs."

→ "Servlets is a Single Instance and multiple threads principle base Server side technology to develop Server side Components as web resource programs of web applications."

NOTE:- A reusable java object is called as "component"

For Basics of Servlets refer page No's 36 to 39 and 41 ~~and~~ to 43

→ Java applications are WORA (write Once Run Any where).

Java web applications are WODA application (Write once Deploye Any where).

Servlet API latest version is 3.0 but the running version is 2.5

In Java environment API comes in the form of packages having classes, interfaces, Annotations, enums and etc....

Servlet 2.5 API means	javax.servlet package
	javax.servlet.http Package

→ Servlets, JSP, EJB and etc are called JEE module technologies. But JEE module is "not a installable software". All web server and Application Server s/w's provide are given based on JEE module. so, to work with JEE module and Technologies we need to install one or other web server (or) Application Server s/w's. like tomcat, weblogic & etc.. This softwares also supplied Jar files representing the API's of JEE technologies

src → winzip

In Tomcat Server

servlet-api.jar represents Servlet api

jsp-api.jar represents Jsp api

In weblogic server

weblogic.jar represents all jee apis

In GlassFish Server

javaee.jar represents all jee apis

→ 3 important resources of servlet api :-

javax.servlet.Servlet (I)

↑ implements

javax.servlet.GenericServlet (AC)

↑ extends

javax.servlet.http.HttpServlet (AC)

→ HttpServlet is an abstract class containing no Abstract methods

→ In Java abstract class can have only abstract method (or) only concreate methods (or) mix of both.

for related information on the above 3 resources of Servlet API refer the diagram given in page no : 54

→ Every servlet program is a Java class that uses ServletAPI having the logic to generate webpages. There are 3 approaches to develop Servlet programs

Approach 1 :- take a Java class implementing Javax.servlet.Servlet (Interface) and provide implementation to all the 5 methods of that interface.

Approach 2 :- take a Java class extending from Javax.servlet.GenericServlet (class) and provide implementation to

Public void service (—, —) method

Approach 3 :- Take a Java class extending from javax.servlet.http.HttpServlet (class) and Override one of the two service(-,-) methods, one of the seven doXxx(-,-) methods.

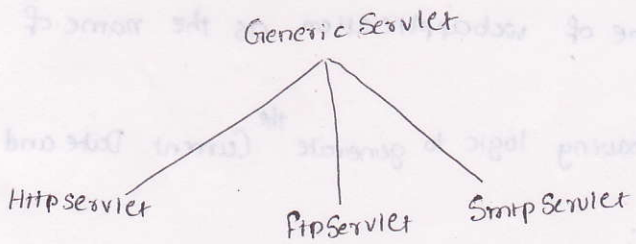
NOTE :- Since all web servers are designed based on the protocol http it is recommended to use Approach 3 to develop the protocol http based Servelet programs.

13/09/2012

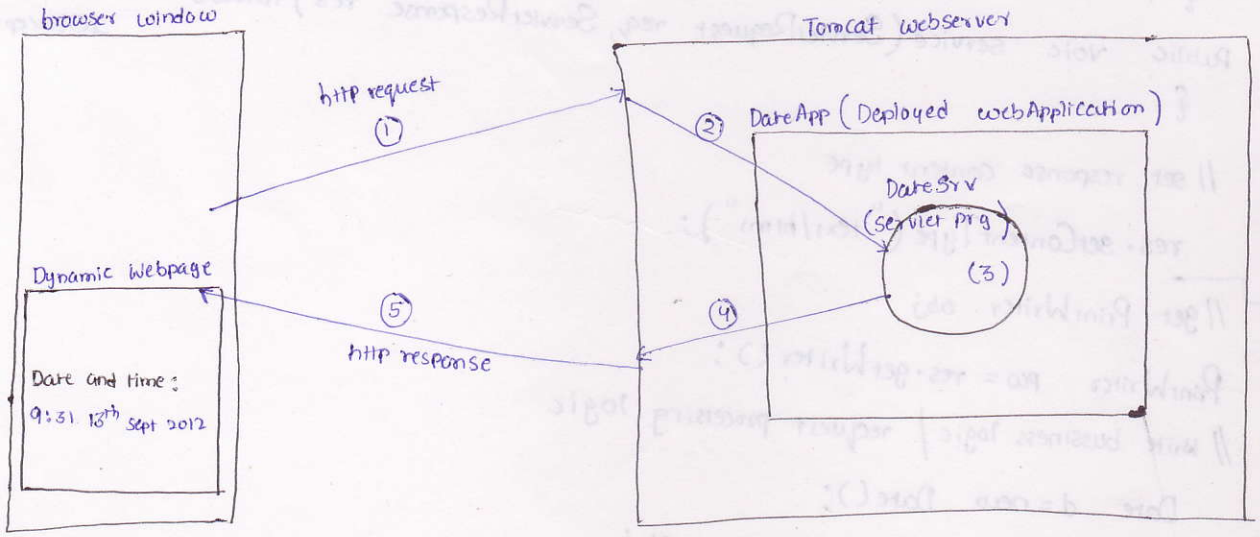
Q:- How many Types of Servlets are there?

Ans:- There is a possibility of developing 'n' types of servlets. like http Servlet, ftp servlet, Smtp Servlet and etc... for all these protocol specific servlet classes Generic Servlet is the Common Superclass containing Commonproperties and logics. so, Generic Servlet is not a Separate Type of Servlet.

As of now Servlet API is giving only one Subclass to Generic Servlet class i.e. HttpServlet class Because all webservers are designed based on the protocol http.

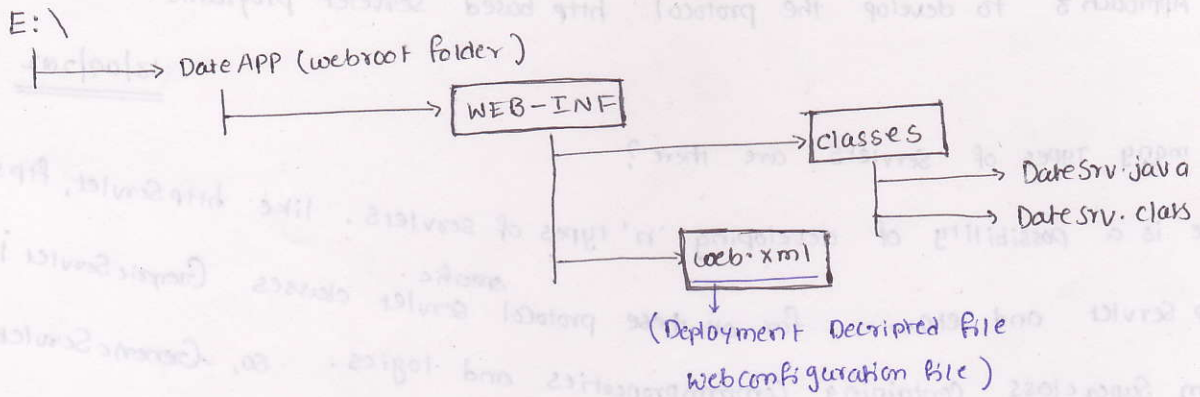


-: An Example on First Web Application Development :-



The procedure to develop and execute the above given java webapplication DateApp :-

Step-I :- Create Deployement directory structure / web Application Packing folder for the above Web application



NOTE: ① The above packing mechanism / deployement directory structure is common in all servers But this directory structure is designed by Sunmicro systems.

② WEB-INF, classes, web.xml names are fixed names

③ We generally take the name of webapplication as the name of webroot folder.

Step-II :- Develop the servlet program having logic to generate the Current Date and time as response.

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

Public class DateSrv extends GenericServlet
{
Public voic service(ServletRequest req, ServletResponse res) throws ServletException, IOException
{
// set response Content type
res.setContentType("text/html");
res.setStatus(404);
// get PrintWriter obj
PrintWriter pw = res.getWriter();
// Write bussiness logic / request processing logic
Date d = new Date();
// Write date and time to response obj
pw.println("<b><Center> Date and Time is " + d.toString() + "</center></b>");
// close the streams
res.setHeader("refresh", "10");
pw.close();
} //service(-,-)
} //class
```

NOTE:- The class of our Servlet program must be taken as public class inorder to making the visibility to Servlet Container.

→ Servlet Container creates object of Servlet class and calls Service(-,-) on that object for multiple times, for multiple requests on one per request basis.

→ for every request is given to servlet program the Servlet Container creates one set of (request, response) objects and calls Service(-,-) having request, response objects has arguments. so, programer receives these request, response object as parameters of the Service(-,-) and uses them while writing logics in Service(-,-)

14/09/2012

Step-III :- Add <tomcat_home>\lib\servlet-api.jar (represents servlet api) file to CLASSPATH environment Variable

My Computer → properties → advance tab → env variables → System/user

Variables → Variable name : CLASSPATH

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

→ ok → ok → ok.

NOTE:- The above jar file in CLASSPATH makes the Java Compiler (javac) to recognize Servlet API during the compilation of our servlet programs.

The APIS of Jdk can be used to directly without adding any values to CLASSPATH Servlet API is not part of Jdk API's. so, The Servlet API related Jar files must be added to CLASS PATH.

Step-IV :- Compile our servlet program. (DateSrv.java)

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

Step-V :- Develop web.xml file as shown below

Specifying the details about certain resource and making underlaying container or Server recognizing that resource these called as Resource Configuration.

Every Servlet program must be configured in web.xml specifying logical name, class name and identity name (URL pattern) to make the Servlet Container recognizing the Servlet program.

Contrl+B

```
<web-app>
    <servlet>
        <servlet-name> abc </servlet-name>
            logical name
        <servlet-class> DateSrv </servlet-class>
            Our Servlet prg class name
    </servlet>

    <servlet-mapping>
        <servlet-name> abc </servlet-name>
            logical name (must match with above logical name)
        <url-pattern> /test1 </url-pattern>
            url pattern
    </servlet-mapping>
</web-app
```

NOTE :- 1 The above given logical name becomes object name for our Servlet class when Servlet Container creates object for it.

→ Every Servlet program is identified with its url-pattern in Server Based executing environment.

→ Servlet Container reads web.xml file the moment deployed webapplication. So, it is called DD file (Deployement Descriptor)

→ Servlet Container, webserver, Clients other web resource programs and etc can't identified Servlet program through its logical name (or) class name. But they identify Servlet program through its URL pattern.

NOTE :- Step-I to Step-V indicates the ~~deployement~~ development of webapplication.

Step-VI :- Start the tomcat Server.

Use <tomcat_home>\bin\tomcat6.0.exe file (or)

(or)

<Tomcat_home>\bin> java -jar bootstrap.jar

Step-VII :- Deployee the web application into

Copy E:\DateApp folder to tomcat_home

Step-VIII :- Test the web application by giving request to Servlet program from client (Browser window)

Open browser window ⟶ Type this URL

http://localhost:2020/DateApp/test1 → url pattern of Servlet program. (DateSrv)

http ↓ protocol

localhost:2020 ↓ hostname and port number of tomcat Server

DateApp ↓ webapplication name (or) Context Path (or) context root

In the above Servlet program :-

res.setContentType("Text/html")

This statement makes web-server to pass instructions to browser window to display Text based webpage by receiving html code from server.

here text/html is called as MIME Type (Multipurpose Instruction Content type)

Other mime types are

application/msword (to display webpage as Msword document).

application/Ms-Excel (to display webpage as Ms-Excel sheet)

text/xml (to display webpage as Xml document)

-

-

-

by using regedit tool (Registry editor) we can gather the MIME TYPES of Various files.

Start ⟶ Run ⟶ regedit ⟶ HKEY_Classes_Root ⟶ choose the extension and get the MIME TYPE.

PrintWriter pw = res.getWriter();

Every response object contains one built-in Stream called PrintWriter. The above Statement use that PrintWriter Stream object to programer pointing to Response object.

```
Pw.Println("<b> hello </b>");
```
↳ given message

In this statement println method makes the PrintWriter stream passing given messages to response object. This response object passes the messages to webserver. This webserver passes the messages to browser window in the form of webpage.

NOTE:- To lanch tomcat manager window smoothly from <tomcat_home> page make sure that following line of code is there in tomcat-users.xml file under
↳ available in <tomcat_home>\conf folder.

```
<tomcat-users>
    <user name="admin" password="admin" roles="admin, manager"/>
```

The physical availability of servlet program in the deployed web application is not sufficient. it must be configured in web.xml file in order to make the underlaying servlet container recognizing then servlet program.

15-09-2012

→ The modifications done in the source code of the servlet program that is there in the deployed webapplication of tomcat server will be reflected only after the recompilition of servlet program and reloading of the webApplication

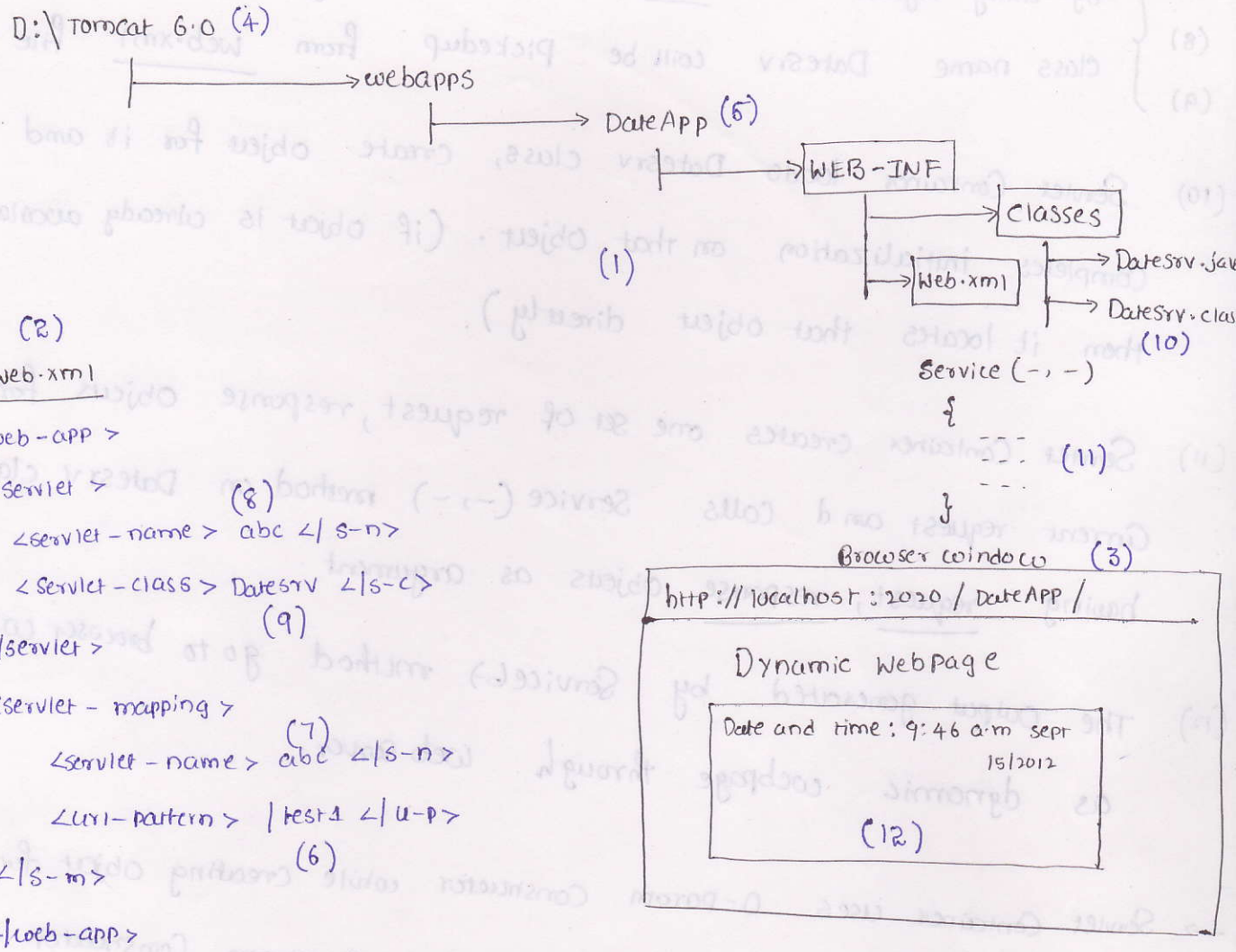
to recompile DateSrv program:

D:\Tomcat 6.0\webapps\DateApp\WEB-INF\classes> javac DateSrv.java

to reload DateApp webApplication:

Tomcat_homepage (http://localhost:2020) → Tomcat manager →
Username : admin
password : admin → OK → DateApp → Reload.

*) Can you Explain the flow of execution from re arrival to Response Generation with support of an example web application.



W.r.to above diagram :-

(1) programmer deployees DateApp application in Tomcat Server.

(2) The Servlet Container reads web.xml Entries and stores them in Map data Stru internally.

(3) End user gives request to Datesrv program by typing request URL in browser window.

(4) Based on local host :2020 of request url the tomcat Server will be located.

(5) Based on DateApp of request URL the deployeed DateApp web application in Tomcat Server will be located.

(6) Based on test1 of request URL the Servlet Container looks for Servlet configuration. whose url pattern is /test1

(7) (8) (9) By using logical name abc of Servlet mapping, servlet tags the class name DateSrv will be pickedup from web.xml file Entries.

(10) Servlet Container loads DateSrv class, create object for it and completes initialization on that object. (if object is already available then it locates that object directly).

(11) Servlet Container creates one set of request, response objects for current request and calls service(-,-) method on DateSrv class object having request, response objects as argument.

(12) The output generated by service(-,-) method go to browser window as dynamic webpage through web server.

→ Servlet Container uses 0-param constructor while creating object for our Servlet class. so, we must make sure that 0-param constructor is available in our Servlet-class directly (or) Indirectly
(placed manually) (the compiler generated default 0-param constructor)

Q:- can be placed only parameterized constructor in our Servlet class?

Ans:- not possible. Bcz Servlet container uses 0-param constructor of our Servlet class that create the object of our Servlet class. if not available then the Servlet Container throws

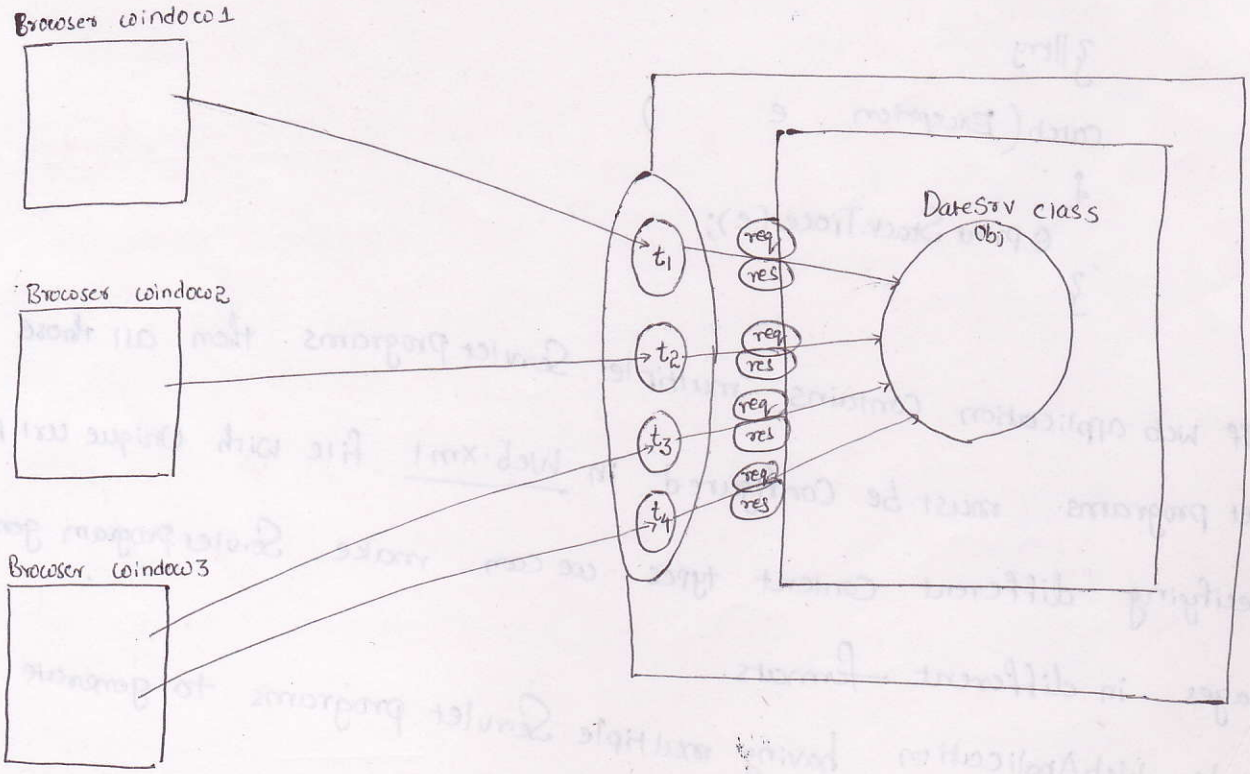
Java.lang.InstantiationException

NOTE:- if there are no constructors in java class then compiler generates 0-param constructor in that class as default constructor during compilation. Otherwise this default constructor will not be generated

→ pw.println() Generated messages of Servlet program goes to browser window as the content of webpage.

→ S.O.P() Generated messages of Servlet program goes to Server Console as log, conformation messages.

17/09/2012



→ when 4 requests are given to a Servlet program

a) The Servlet Container creates one object of that Servlet class.

b) Servlet Container starts 4 threads on that object representing 4 requests on per request basis.

c) Servlet Container creates 4 sets of ~~response~~ request, response Objects on one set per each request. (if any change is occur recompile, reload.)

→ Every Thread can be identified through its "thread name". Every Object can be identified with its Unique number called "HashCode". To get the HashCode of any Object call HashCode () of on that object.

→ To prove the above diagram related statements practically add the following code in the Service (-,-) of DateSrv.java

```
//write date and time to respone obj
pw.println("<b> <center> Date and time is "+d.toString()" </center> <b>");
try
{
    Thread.sleep(30000);
    pw.println("<br> Current request thread name is " + Thread.currentThread().getName());
    pw.println("<br> Current request related req obj hashcode is" + req.hasCode());
    pw.println("<br> Current request related res obj hashcode is" + res.hashCode());
    pw.println("<br> Current DateSrv (servlet class) obj hashcode is" + this.hashCode());
}//try
catch(Exception e)
{
    e.printStackTrace(e);
}
```

NOTE:- If web application contains multiple Servlet programs then all those Servlet programs must be configured in web.xml file with Unique url patterns.

→ By specifying different content types we can make Servlet program generating webpages in different formats.

*) Example WebApplication having multiple Servlet programs to generate the webpages in different formats.

Step-I :- Create the deployement directory structure.

E:\MIME App
- WEB-INF
  - classes
    - HtmlSrv.java
    - ExcelSrv.java
    - WordSrv.java
    - XmlSrv.java.
    - *.class (4)
  - web.xml

Step-II :- Develop source code of Servlet programs and web.xml file.

Refer Application (1) of page no: 58 to 60

In 60th page 154 to 163 lines strike of it.

NOTE :- *) Using ServletRequest, ServletResponse objects we can't work with all the features of protocol http. where as using HttpServletRequest, HttpServletResponse objects u can work with all the features of protocol Http.

*) When our Servlet class extends from Generic Servlet class then we need to use ServletRequest, ServletResponse objets. When our Servlet Class extends from HttpServlet class then we need to use HttpServlet Request, HttpServlet Response Objects.

Step-III :- Compile the .java files of above application.

E:\MIMEAPP\WEBINF\classes > Javac *.java

Step-IV :- Deployee the web application in tomcat server.

Copy E:\MIMEApp folder to

<Tomcat_home>\webapps folder.

Step-V :- Start the tomcat server and Test the web application by using the following Request urls.

http://localhost:2020/MIMEApp/htmlurl
http://localhost:2020/MIMEApp/wdurl
http://localhost:2020/MIMEApp/xlsurl
http://localhost:2020/MIMEApp/xmlurl

NOTE:- The Modifications done in Web.xml file will be recognized by Underlaying Server automatically. Where as the modifications done in the sourcecode of Servlet Program will be recognized by underlaying Server only after Re-compilation of Servlet program and the reloading of the webapplication.

Q:- How can we provide multiple-url patterns to a Servlet program.

Ans:- place the <url-pattern> for multiple times Under <Servlet-mapping>

Eg:-

```
<servlet>
    <servlet-name> abc </s-n>
    <servlet-class> DateSrv </s-c>
</servlet>
<servlet-mapping>
    <servlet-name> abc </s-n>
    <url-pattern> /test1 </url-pattern>
    <url-pattern> /test2 </url-pattern>
</ servlet-mapping>
</servlet>
```

*) <Url-Pattern> heights our Servlet classname and Technology name from our Siders (clients) who gives the request to Our Servlet program.

*) The WEB-INF folder of Java Webapplication is called as private directory. because it can be accessed and used only the underlaying webserver, where the webapplication is deployeed.

→ only underlaying Server can use WEB-INF folder that means clients, programs Can't use this WEB-INF folder.

Q:- When ServletRequest / HttpServletRequest, ServletResponse / HttpServletResponse are the interfaces. How can you say the parameters of ~~req~~ Service (req, res) are the objects! 18/09/2012

Ans:- req, res are not the objects of the above said interfaces.

→ req (request obj) is the object of underlaying Servlet Container Supplied java class that implements javax.servlet.ServletRequest (I) directly (or) indirectly.

→ res (response obj) is the Object of a Underlaying Container supplied java class. that implements javax.servlet.ServlerResponse (I) directly (or) indirectly.

→ In tomcat server request object classname : org.apache.catalina.Connector.RequestFacade

response object classname : org.apache.catalina.Connector.ResponseFacade.

NOTE:- The classnames of request, response objects will change based on the server / Servlet Container we use. So, programers never specify these classnames in Servlet programs to make ServletPrograms as the Server independent programs.

javax.servlet.ServletRequest (I)
↑ extends
javax.servlet.http.HttpServletRequest (I)

javax.servlet.ServletResponse (I)
↑ extends
javax.servlet.http.HttpServletResponse (I)

To know the classnames of request, response object :-

pw.println("
 req obj class name" + req.getClass()); o/p:- req obj class name : RequestFaced

pw.println("
 res Obj class name" + res.getClass()); o/p:- res obj class name : ResponseFaced

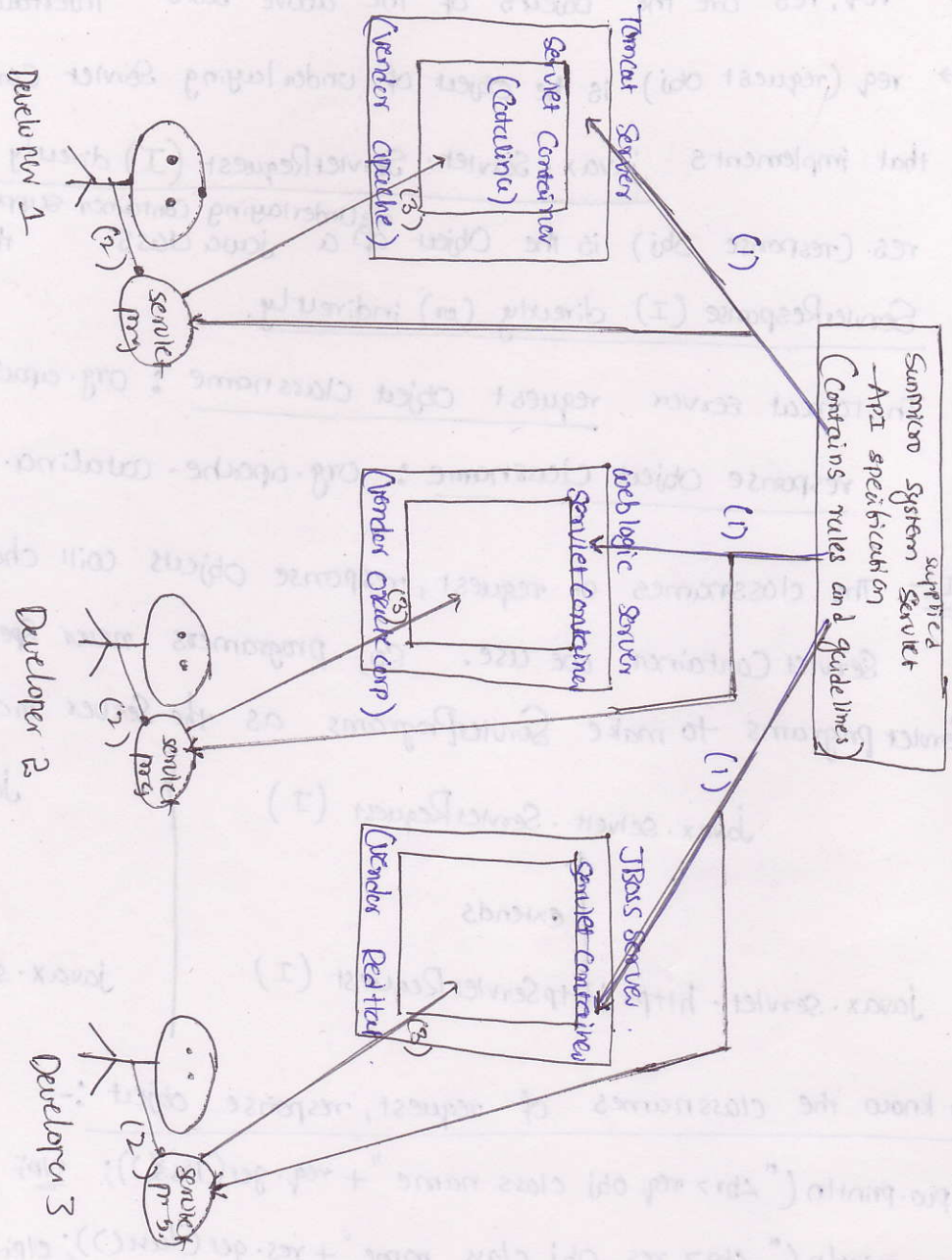
*) understanding the development process of ServletContainer and Servlet programs :-

→ Sunmicro System Supplied ServletSpecification Contains API having rules and guidelines to develops Servlet Container and facilities to develop Servlet programs

→ Vendor Companies use Servlet-API to develop Servlet-Container. Similarly programers use Servlet-API to develop Servlet programs.

→ Servlet Container is a Vendor developed Set of classes implementing Various interfaces of Servlet API packages. In all these classes logic will be there and to execute Servlet programs.

→ Since the Servlet Container and Servlet Programs are given based on Servlet-api, so, the Servlet Container can execute Servlet Programs easily.



→ w.r.to the diagram

(1) vendor Companies and develop Servlet Container for there ApplicationServer, WebServer s/w's.

(2) programer use Servlet-API to develop Servlet programs having logic to generate Webpages.

(3) Programer's keeps Servlet programs in Servlet Container for execution. and Servlet Container executes them automatically when they get requests from Client.

→ Since all servers and their containers are coming based on to the common supports supplied servlet, JSP specification so, the way we work with all these servers is going to be much similar.

→ While overriding superclass method in subclass the method can have same (or) access modifier

Eg:- The protected (-,-) method of superclass can be overridden in subclass with public or protected modifier.

public → access modifier
protected → same

→ While working with HttpServlet class we can work with two service(-,-) :

① service(-,-) method 1 | public service(-,-) :-

public void service(ServiceRequest req, ServiceResponse res) throws SE, IOException

② service(-,-) method 2 | protected service(-,-) :-

protected void service(HttpServletRequest req, HttpServletResponse res) throws SE, IOE

*) we can override the protected service(-,-) (service(-,-) method 2) of HttpServlet class in its subclass (our servlet program) either with protected same (or) public access modifier.

20/09/2012

→ U must specify our servlet class name with package name when that class is there in a package. (class name (or) interface name with package name is called as "Fully qualified name.")

First java webapplication (DateApp)

DateSrv.java

```
package P1;
public class DateSrv extends GenericServlet
{
    -----
    -----
    -----
}
```

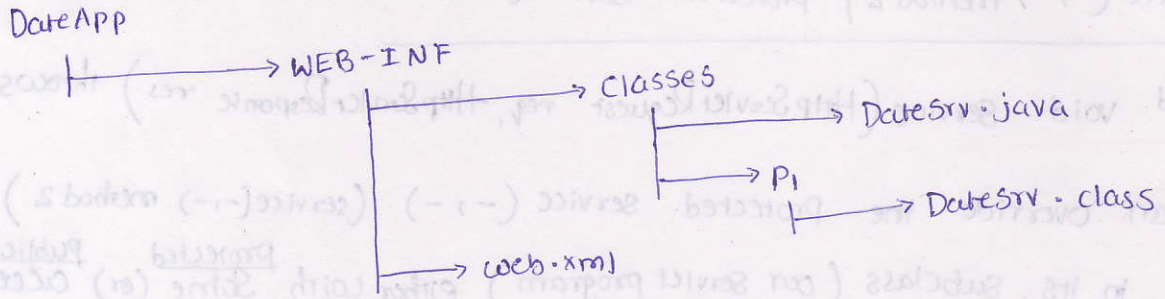
...\DateApp\WEB-INF\classes> javac -d . DateSrv.java

Web.xml

```
<web-app>
    <servlet>
        <servlet-name> abc </servlet-name>
        <servlet-class> P1.DateSrv </servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name> abc </servlet-name>
        <url-pattern> /test1 </url-pattern>
    </servlet-mapping>
</web-app>
```

Facebook PHP
twitter Java
gmail Asp

The deployement directory structure of webapplication :-



Request url :

http://localhost:2020/DateApp/test1

→ If Java webapplication is there ~~when~~ with in multiple modules then it is recomended to place the Servlet programs of each module in a separate module.

* URL pattern ~~heights~~ hide servlet class name and technology name from the visitors of the web application

* According to Servlet Specification we can prepare 3 types of Url-pattern and all servers are designed to recognize only these types of URL-patterns.

They are

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

① Exact match :-

→ url pattern must begin "/" character

→ url pattern must not have "*" character.

Eg:-

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

request urls

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

/DateApp/test1/abc (Invalied)

/DateApp/abc (Invalied)

/DateApp/xyz/test1 (Invalied)

Other examples url patterns (Exact match)

① <url-pattern> /test1/test2 </url-pattern>

② <url-pattern> /test1.do </url-pattern>

③ <url-pattern> /abc.do/test1 </url-pattern>

② Directory match :-

→ url pattern must begin with "/" character

→ url pattern must end with "*" character

→ url pattern can

Eg:-

<url-pattern> /x/y/* </url-pattern>

↳ in the place of '*' any thing write in req url

'*' → 0 (or) more characters can write

request urls

http://localhost:2020/DateApp/x/y/abc.do (Valied)

/DateApp/y/x/abc.abc (invalied)

/DateApp/x/y/x/y/abc (valied)

/DateApp/x/y (valied)

/DateApp/x/y/abc.cpp (valied)

/DateApp/m/n/123 (invalied)

Other example directory match ~~**~~ url patterns :-

a) <url-pattern> /test1/* </url-pattern>

b) <url-pattern> /x/y/* </url-pattern>

c) <url-pattern> /abc.do/xyz/* </url-pattern>

If expose the technology name of the website hackers & Jackers may Hack the website. Quickly TO overcome this problem use url-pattern to hide the technology name.

③ Extension match :-

→ url pattern must begin with "*" Symbol.

→ url pattern mush have extension word or letter

*. <extension word / letter>

Eg:- <url-pattern> *.do </url-pattern>

Example request urls :-

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

/DateApp/xyz/abc/k.do (valied)

/DateApp/.do (valied)

/DateApp/x.abc (Invalied)

/DateApp/x.do.abc (Invalied)

/DateApp/a.x.z.do (Valied)

Other example extension match url patterns :-

① <url-pattern> *.c </url-pattern>

② <url-pattern> *.xyz </url-pattern>

③ <url-pattern> *.do </url-pattern>

→ <url-pattern> /test1.do </url-Pattern>

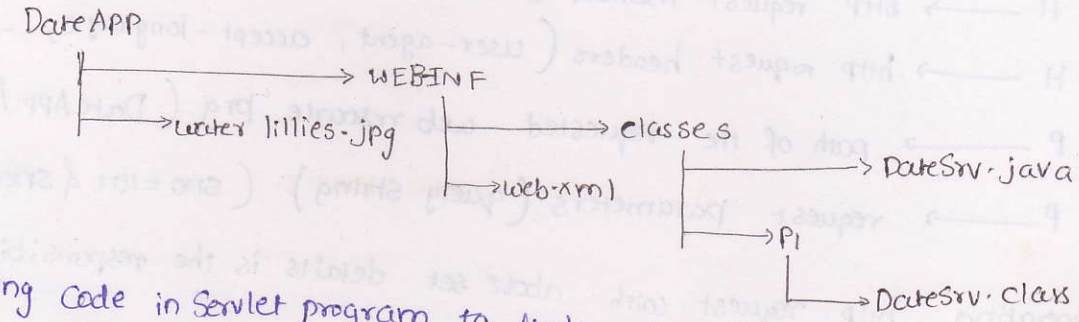
This url-pattern comes under Exact match url-pattern. we can't frame url-pattern by mixing up multiple ~~styles~~ styles.

<url-pattern> /x/y/*.do </url-pattern> (Invalied url-pattern formation. bcz here we mixed up the ~~Exact~~ Extension match, Directory match styles).

*) Procedure to display image on the webpage generated by Servlet programing.

Step-I :- Place Image file of under webroot folder of Deployee web Application

Eg:- In DateApp folder of <tomcat_home>/webApps folder.



Step-II :- write following code in Servlet program to display the image.

In Service(-,-) of DateSrv.java

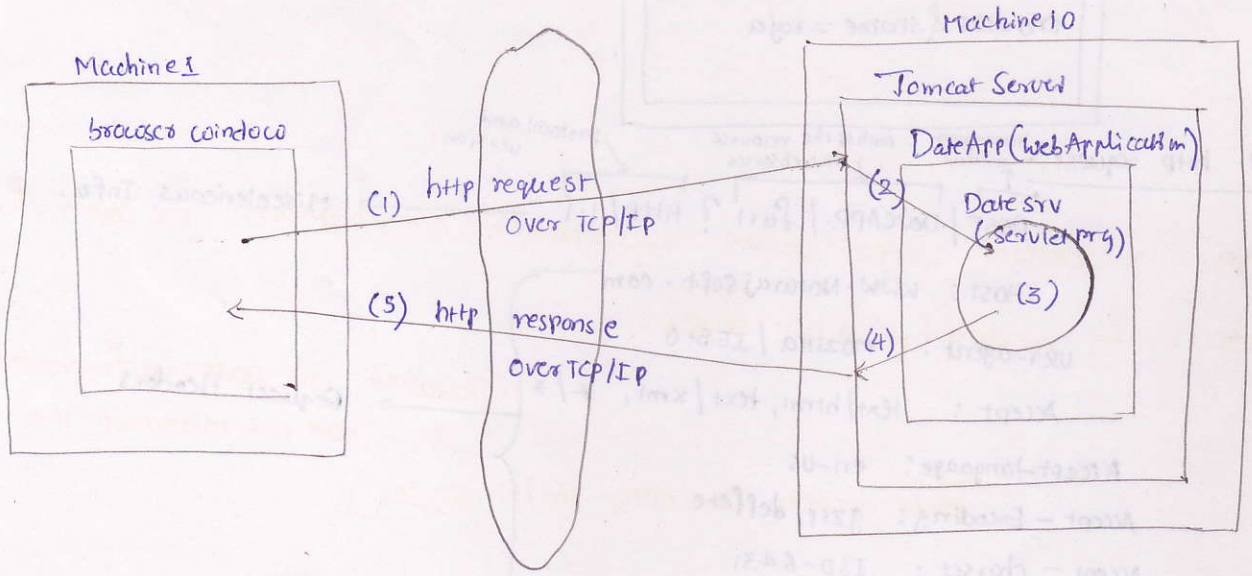
```
pw.println("<br><br><img src='water lillies.jpg' width='100' height='200'>");
pw.close();
```

21/09/2012

→ The text that allows non-sequential access through hyper link is called as "hypertext"

→ protocol http defines set of Rules that are required to pass hypertext b/w browser to Server and Server to browser

→ http is application level protocol that runs over network level protocol TCP/IP.



→ when browser window generates request to webapplication the request contains multiple details given by browser window. we can remember this details h^2p^2 details

Request url :-

http://localhost:2020/DateApp/test1 ? sno=101 & sname=raja

query String

h2P2 details:-

H ⟶ http request method (GET/POST/.....)

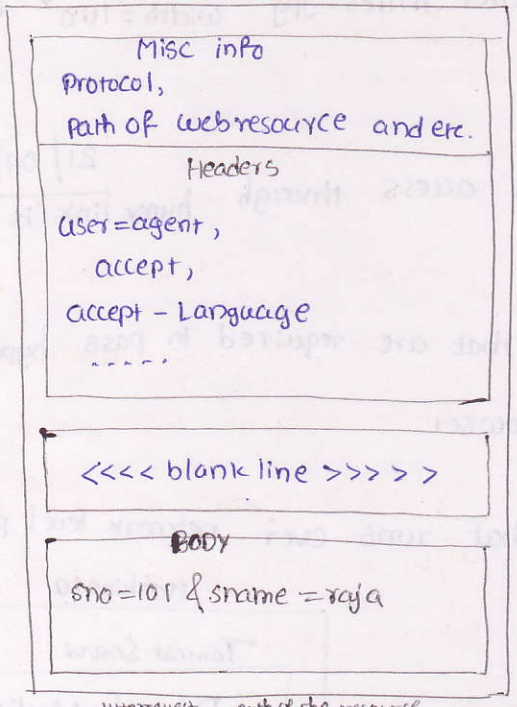
H ⟶ http request headers (user-agent, accept-language, ---)

P ⟶ path of the requested web resource prg (DateApp/test1)

P ⟶ request parameters (query string) (sno=101 & sname=raja)

Generating http request with above set details is the responsibility of browser window.

Block diagram of Http request :-



An example http request :-

HTTrequest method — path of the resource on webserver — protocol and version

POST / DateApp / fun1 ? HTTP/1.1 ⟶ Miscelenceous Info.

HOST : www.Nataraj soft.com

User-agent : Mozilla / IE5.0

Accept : text/html, text/xml, */* ⟶ Request Headers

Accept-language : en-US

Accept-Encoding : gzip, deffere

Accept-charset : ISO-6431

Keep-Alive : 300

Connection : keep-Alive.

<<<<< blank line <<<<<< ⟶ blank Line

sno=101 & sname=raja ⟶ Request Body / Pay load

Q:- What is the diff b/w HttpRequestHeaders and RequestParameters.

RequestHeaders	RequestParameters
→ holds the browser supplied input values.	→ holds the Visitor (end user) supplied input values (form data)
→ Header names are fixed but the values are browser specified	→ Parameter names and values are user-defined
→ Manditory in every request	→ optional
→ Header names are unique	→ duplicate names are allowed
Eg:- User-agent : Accep : and etc.	Eg:- sno = 101 & sname = raja

for related info on http request and its header values refer page NO: 46 to 49

→ list of HttpRequestHeaders :-

Accept - Encoding
Cookie
Connection
keep - alive
Host
refferer
User - agent
if - modified - since
Authorization
Accept - charset and etc.....

→ to gather various details from client generate request being from Servlet program we need to use either ServletRequest object (or) HttpServletRequest object. Using HttpServletRequest Obj we can gather all the details from request. Using ServletRequest Object we can gather only few details that mean we can't gather details like header values & some mescelaneous info.

→ Different approaches of gathering request parameter values being Servlet programs :-

NOTE:- Use either ServlerRequest obj / HttpServlerRequest obj

Example request url:

http://localhost:2020/DateApp/test1?sno=101&sname=raja&sname=king

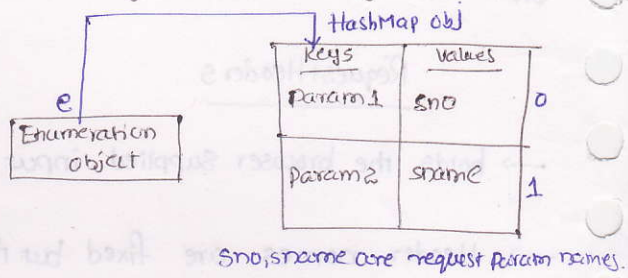
Approach 1:

```
String s1=req.getParameter("sno"); // gives 101
String s2=req.getParameter("sname"); // gives raja
```

NOTE:- ① we must know parameter name inorder to get its value.

② If parameter contains multiple values then it gives only one value. (1st value).

Approach 2:-




```
Enumeration e = req.getParameterNames();
while (e.hasMoreElement())
{
    String pname = (String) e.nextElement();
    String pval = req.getParameter(pname);
    pw.println(pname + "——→" + pval);
}
```

sname → raja
sno → 101

gives names and values of all the request parameters but gives only 1st value when request parameter contains multiple values

Approach 3:-

```
String s[] = req.getParameterValues("sname");
```
↳ holds raja, king as element values

```
→ String s1 = req.getParameterValues("sname")[0]; // gives raja
  String s2 = req.getParameterValues("sname")[1]; // gives king.
```

NOTE:- This is useful to gather the multiple values of each request parameter.

if u give any request code then u must write in service(-,-)

22/09/2012

Different approaches of reading req headers values being from Servlet program:-

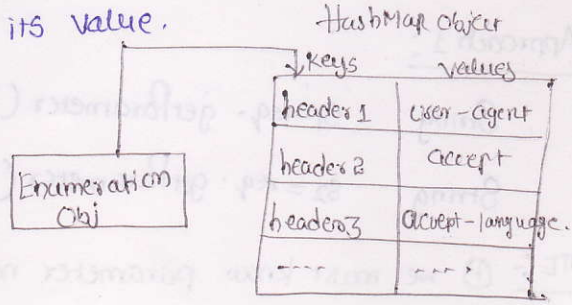
NOTE: we must use HttpServletRequest object for this

approach 1:

```
String s1 = req.getHeader("user-agent"); // gives browser s/w name
String s2 = req.getHeader("accept-language"); // gives en-US
```

note: here we must know header name to get its value.

approach 2:




```
Enumeration e = req.getHeaderNames();
while (e.hasMoreElements())
{
```

```
String hname = (String) e.nextElement();  // gives each header name
String hval = req.getHeader(hname);  // gives each header value
pw.println(hname + "  ---> " + hvalue);
}
```

gives all the request header names and values.

→ Using browser settings we can change certain header values.

→ header values will change based on the browser s/w we use.

***) we can change accept-Language value through browser setting.

In IE

tools menu ⟶ internet options ⟶ languages ⟶ add ...

In Netscape

Edit menu ⟶ preferences ⟶ Navigator ⟶ Languages ⟶ add ⟶ ...

Q:- How to know the browser s/w name being from Servlet program?

Ans:- The User-Agent request header as shown below.

```
String name = req.getHeader("user-agent");
```

Servlets is the part of JEE module. So, for ServletAPI documentation prefer the JEE API docs (get from Javadoccee6.zip file extract)

→ use getXxx() on ServletRequest, HttpServletRequest objects to gather miscellaneous info from client generated request.

// using the methods of javax.servlet.ServletRequest (I)

```
pw.println("<br> req content length " + req.getContentLength());  ⟶ 20 KB (or) -1
pw.println("<br> req Content type " + req.getContentType());  ⟶ text/html ..... (or) null
pw.println("<br> protocol " + req.getProtocol());  ⟶ HTTP/1.1
pw.println("<br> scheme" + req.getScheme());  http
pw.println("<br> Browser port no" + req.getRemotePort());  ⟶ 1678 (any number)
pw.println("<br> Client machine host name" + req.getRemoteHost());  ⟶ 127.0.0.1
pw.println("<br> client machine ip address" + req.getRemoteAddr());  ⟶ 127.0.0.1
```

(↓ varying number) (↓ when length is unknown)

(when contentType is unkn

// using the method of javax.servlet.http.HttpServletRequest (I)

pw.println("
 req method" +req.getMethod()); → GET

pw.println("
 query String" +req.getQueryString()); → sno=? & sname=?

pw.println("
 context Path" +req.getContextPath()); → DateApp

pw.println("
 request uri" +req.getRequestURI()); → /DateApp/test1

pw.println("
 request url" +req.getRequestURL()); → http://localhost:2020/DateApp/test1

pw.println("
 Servlet Path" + req.getServletPath()); → test1
↳ url pattern

Tomcat6.0/lib/catalina/winzip/ue venh
RequestFaced [GetProtocol()]

*) Under Standing the OOPS related to Various method calls that are used in ServletProgram

```
String s1 = req.getProtocol()
```

① getProtocol() is declared in javax.servlet.ServletRequest (I)

② This method is implemented in Container Supplied java class that implements javax.servlet.ServletRequest (I). incase of Tomcat server this implementation Class name is ~~javax.~~ org.apache.Catalina.Connector.RequestFacade

③ This method returns the protocol of the CurrentRequest.

```
PrintWriter pw = res.getWriter();
```

① getWriter() is declared in javax.servlet.ServletResponse (I)

② This method is implemented in Container Supplied java class that implements javax.servlet.ServletResponse (I). In Tomcat Server this implementation class name is org.apache.catalina.Connector.ResponseFacade.

③ This method contains logic to contain PrintWriter Stream object pointing to response object.

NOTE:- We Never specify Container Supplied classes that are implementing interfaces. like RequestFacade, ResponseFacade.

In our Servlet program bcz this classnames will change Server to Server (container to container)

23/09/2012

When webresource program (Servlet program) sends output to browser window as Http response that response contains multiple details. we generally remember these details as "SCH" details.

S ⟶ Http response status code (100-599)

C ⟶ response content

H ⟶ response Headers

Response status codes :- Tells the status of response generation

100-199 ⟶ info

200-299 ⟶ success

300-399 ⟶ Redirection

400-499 ⟶ Incomplete webresource prg

500-599 ⟶ Server Error

Response Content :- nothing but output displayed on the webpage

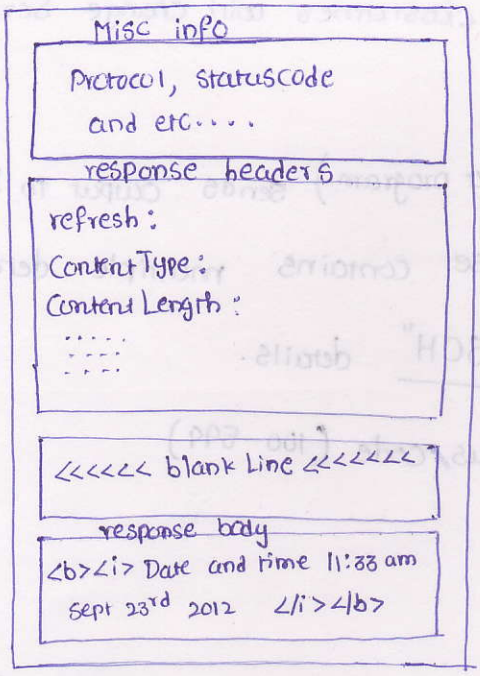
msgs placed in pw.println(-) become response content

Response Headers :- server | webresource prg passes instructions to browser window as response header value.

response headers are

refresh, ContentType, ContentLength, Cache-control and etc.

Block Diagram of Http response



An Example Http Response :-

Misceleneous info ← { Http/1.1 200 OK → http status code

Response Headers ←
- Set-cookie : JSESSIONID = 1498 3 A B E A 179 E A
- Conten-Type : text/html
- Content-length : 1354
- Date : Fri Feb 2008 8:40:34 GMT
- Server : Apache-Tomcat/5.0
- Connection : close

<<<<<<<< Blank Line <<<<<<

Response body (or) Content (or) Payload
- <html> <body> <h1> welcome to servlets </h1> </body> </html>
-   Date and time 11:33 am sept 23rd 2012

→ Server Automatically sets default values to certain Response headers. if you want to override those values (or) if you want to set values to other headers then use response.setHeader(-,-) method in Servlet program.

→ The default response header code is 200

Q:- How to make browser window refreshing the web page automatically at reguler interface?

Ans:- Use the response header refresh as shown below in servlet program.

```
res.setHeader("refresh", "10");
```

→ time in seconds.
→ header value
→ header name
→ makes browser window to refresh the webpage after every 10 seconds.

→ for related info on HttpResponse and its status codes, Headers refer pageno: 49-51

The following are the Http Response Headers :-

Location

Refresh

Set-cookies

Cache-control

Content-encoding

Content Length

Content Type

Last modified

Date

Server

Content-Desposition

⋮

and etc....

res.setContentType(-) :- gives instruction to browser window to display webpage in different formats. based on the specified Mime type

Q:- How to make webResource program to pass instruction to browser window for not storing output in the buffer of browser window?

Ans:- Buffer is a temporary memory holding data for temporary period. Browser window holds the output of servlet program in the buffer and uses it across the multiple same request. and reduces network round trips b/w browser window and server.

at certain times this buffering becomes disadvantage if the web resource program output changes dynamically for every request. To disable this buffer write following code in servlet program.

```
res.setHeader("pragma", "no-cache");      http 1.0 based server
res.setHeader("cache-control", "no-cache");   http 1.1 based server
```

servlet API
jee.9620docs / jee5docs / index.htm
javax.http, servlet.http,

→ we can make a servlet program sending response to browser window having programer choice status code by using res.setStatus(404) method

Eg:- res.setStatus(404)

{ Universal Resource Indicator (URI) (is the part of URL)
Universal Resource Locator (URL)

24/09/2012

→ In our servlet program we can take two types of streams pointing to response object.

① PrintWriter (Character Stream)
② ServletOutputStream (byte Stream)

NOTE:→ When there is more text data and less/no binary data (images) to write to webpage then Use Character Stream (PrintWriter)

→ When there is more Binary data and less text data to write then use byte Stream (ServletOutputStream).

*) Pointing to response Object you can create only one Stream in our servlet program

Eg:- Character Stream pointing to response Object

```
PrintWriter pw = res.getWriter();
pw.println("<b> hello </b>");
```

} more text data and less binary data / no binary data to write to webpage

byte Stream pointing to response obj

ServletOutputStream sos = res.getOutStream();

sos.println(" hello ");

jdk1.6 → src → winzip → open with EditPlus

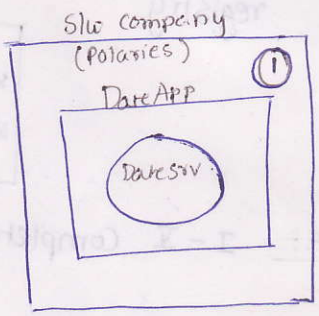
→ Instead of Calling Println() on Stream Objects (ServletOutputStream, PrintWriter) it is recomended to call Print(). because the Println() also internally calls print().

→ What is the diff b/w pw.print(), pw.println() methods?

Ans:- pw.print() writes the data to destination (like response obj) in non-synchronized environment. where as pw.println() performs the same activity in Synchronized environment.

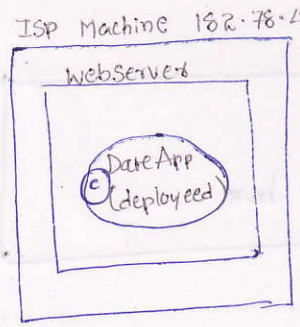
→ keeping the webapplication in the internet network by having domain name and by acquiring space is nothing but hosting of the website.

→ procedure to host java webapplication in the internet network



2) Client Org purchases the Domain name www.sathya.com

3) Client Org purchases space in the Isp machine webserver that contains static (fixed) IP address



(4) Admin uploads the webapplication (DateApp) to the Isp machine webserver using FTP Application

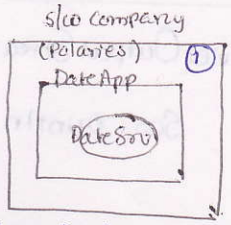
(5) register the domain name with DNS Registry (global)

b
www.sathya.com → http://182.78.4.3:2020/DateApp/Test1
www.yahoo.com → http://184.52.5.4:8080/YahooApp/Index.jsp

(6) access the website

www.sathya.com a

Step-I :- Developers Completes the development of webapplication in s/w company

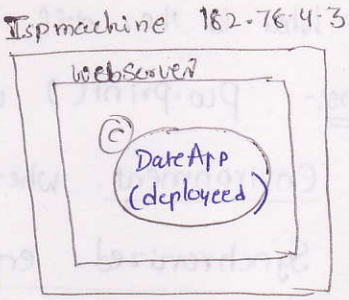


Step-II :- The Client organization purchases domain name from (ISP) (like www.godaddy.com)

ISP name.

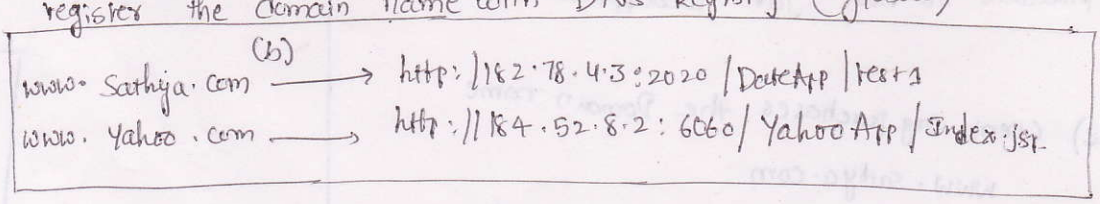
note :- Assume that we purchased Sathya.com as domain name.

Step-III :- Client Organization purchases Space in the webserver of ISP machine that Contains static (fixed) Ip address.



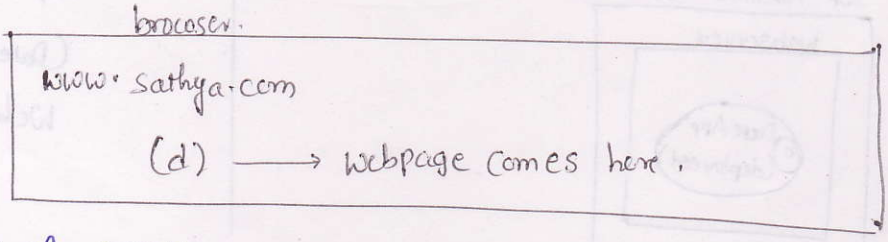
Step-IV :- The administrator of s/w company uses FTP application and Uploads the webapplication to the webserver of Isp machine.

Step-V :- Administrator (or) ISP registor the domains name with global DNS registry.



NOTE :- I - V Completes the hosting of the website.

Step-VI :- End user access the website by using domain name.



a,b,c,d are flow of execution.

NOTE :- Administrator / Client Organization Uses FTP application to Connect to ISP machine and to add (or) replace WebResource programs in deployed web application for the maintanance of the website.

→ ServletConfig object is Righthand Object to our Servlet class Object.

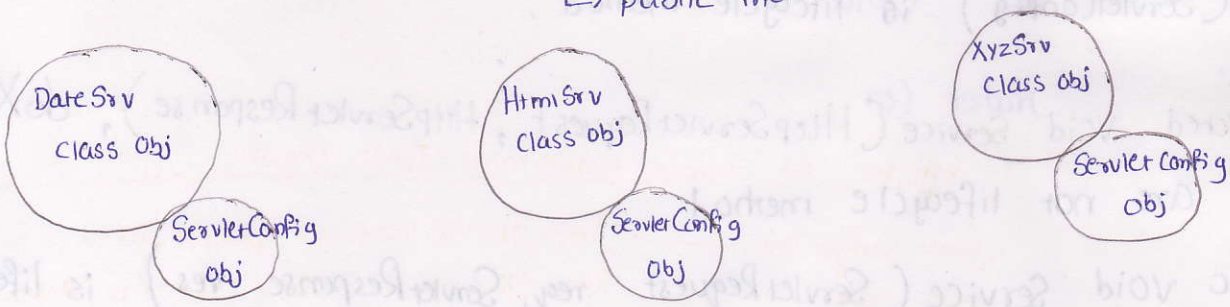
→ It is one per Servlet class object.

→ This object is useful to programer to know about Servlet program and to gather data to required for the servlet program from outside the Servlet program.

→ ServletConfig object means it is the object of ServletContainer supplied Java class that implements javax.servlet.ServletConfig (I)

→ we can access this object in our Servlet program by calling getServletConfig() method

ServletConfig cg = getServletConfig();
↳ public method of GenericServlet class



25/09/2012

Servletlifecycle keeping track of all the operations which takes place ~~between~~ from our ServletClass object creation to distruction.

→ In this servlet life cycle there are 3 importent lifecycle ~~methods~~ Events

(1) Instantiation Event :- raises when ServletContainer creates our servlet class.

(2) Request Arrival Event :- raises when ServletContainer receives Each request of our Servlet program.

(3) Destruction Event (raises when ServletContainer is about to the destroy our Servlet class Object).

→ The methods that will be called by underLaying Container automatically for life-cycle events are called as "lifecycle methods" (or) "Container Callback methods"

→ ServletContainer calls the life-cycle methods of ServletProgram are :-

① init(ServletConfig cg) NOTE:- ServletContainer calls this method for Instantiation Event

② Service(ServletRequest, ServletResponse) (public service(-, -))

NOTE:- ServletContainer calls this method for Request ArrivalEvent

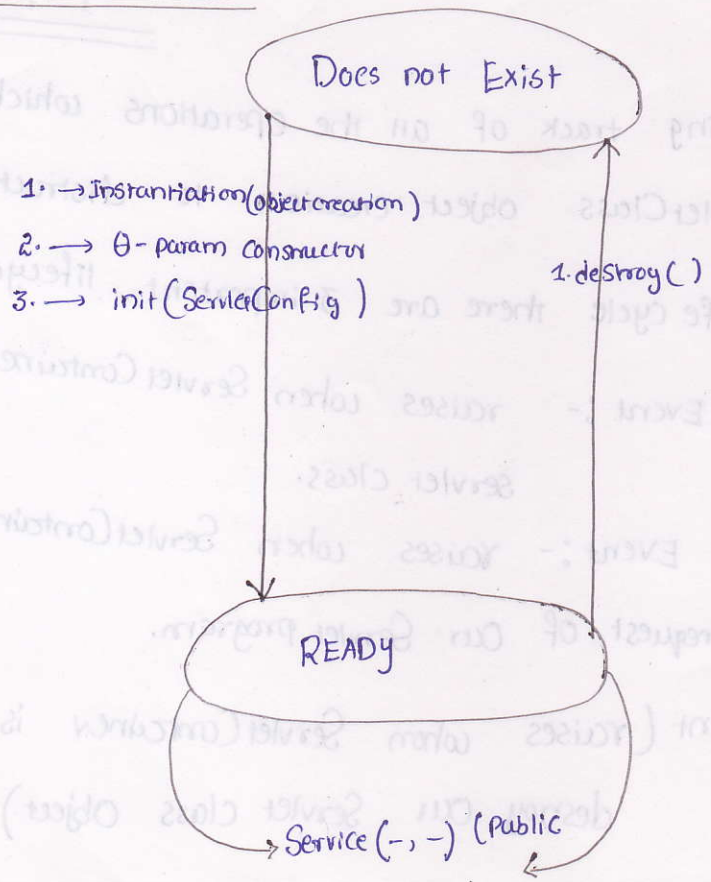
③ destroy()

NOTE:- ServletContainer calls this method for Destruction Event

init() → is not lifecycle method
init(ServletConfig) is lifecycle method

protected void Service(HttpServletRequest, HttpServletResponse), doXxx(-,-) are not lifecycle method
public void Service(ServletRequest req, ServletResponse res) is lifecycle method.

Servlet lifecycle diagram :-



25/09/2012

(1)

Servlet Life Cycle Methods

1. init(-) method

It is one time execution block of our servlet program. Servlet container calls this method automatically when it instantiate our servlet class. Programmer generally places initialization logic of servlet program in this init (-) like creating JDBC connection.

PROTOTYPE:

Prototype of this method is

public void init(ServletConfig cg)throws ServletException

- Servlet container throws servletException if problems are there in the normal execution of servlet program.

2. Service (-,-) Method:

- It is repeatedly executing lifecycle method of our servlet program. Container call this method every time, when it receives request from the client to servlet program.
- programmer generally places request processing and response generation logics in this method.. it is most important life cycle method of servlet program.

PROTOTYPE:

Prototype of this method is

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

- Servlet container throws IOException when it fails to give PrintWriter stream object from response object.

3. destroy () Method:

- This is one time execution block of servlet program . this method will be called when servlet container is about to destroy our servlet class object.
- Programmer generally places un-initialization logic in this method like closing JDBC connection object.

PROTOTYPE:

public void destroy ();

- When you don't place these life cycle methods in our servlet program. Then these methods available in super class like GenericServlet or HttpServlet will be executed automatically.

ServletConfig object:

- ServletConfig is an interface. We cannot create object to interface. So it is the object of servlet container supplied java class which implements javax.servlet.ServletConfig interface.
- ServletConfig object is right-hand object to our servlet class object.
- This object is created by the servlet container as one per our servlet class object basis.

- ◇ Programmer can use this object in our servlet program to gather information about current servlet program and to pass additional information to our servlet program.
- ◇ Servlet container exposes ServletConfig object to our servlet program as parameter of init(-) life cycle method.

Note: Servlet life cycle methods are useful for the programmers to place our choice logics in the servlet program and also useful to get access to servlet container created objects in our servlet program as the parameters of lifecycle methods.

Question: What happens when our servlet program gets first request from browser window?

Ans: (a) servlet containers loads our servlet class from WEB-INF\classes folder of deployed web application and instantiates that class using 0- parameter constructor.

Instantiation

object creation

Class.forName("DataSrv"). newInstance();

Note: In this process o-param constructor of our servlet class executes.

Initialization

(b) Servlet container ~~calls~~ creates our ServletConfig object as right hand object of our servlet class object.

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

request processing

(d) Servlet container creates two objects request, response objects representing current request. 1st request

(e) Servlet container calls another lifecycle method on our servlet class object having request, response objects as argument values. This service (-,-) method processes the current request and sends response/webpage to client (browser window) through webserver.

QUE: What happens when our servlet program gets other than first request from browser window?

Ans: Servlet container ~~checks~~ checks the availability of our servlet class object.

If not available:

Servlet container performs all the activities as it is as discussed above for first request.

If available:

request processing

Servlet container locates that object (our servlet class object) → creates request, response for current request (means other than first request)

—> calls service (-,-)method on that existing servlet class object having request, response objects as arguments

—> this service (-,-) (life cycle method) processes the current request and sends response to browser window as webpage through web server.

<load - on - startup>

- ◇ Generally servlet containers creates our servlet class object when it gets first request from client for that servlet program.
- ◇ When <load-on-startup> is enabled on servlet program then servlet container creates our servlet class object either during the deployment of the web application (server is started without deploying the web application) or during the server startup (if web application is deployed before starting the server)

To enable <load-on-startup> on servlet program write following code in web.xml during the configuration of our servlet program.

```
<web-app>
<servlet>
<servlet-class>lc</servlet-name>
<servlet-class>LcTestSrv </servlet-class>
<load-on-startup> 1 <load-on-startup>
</servlet>
<servlet-mapping>
<servlet-name>Lc </servlet-name>
<url-pattern> /lctest </url-pattern>
<servlet-mapping>
</web-app>
```

◇ servlet container creates object for our servlet class either during server startup (or) during deployment of web application when <load-on-startup> is enabled it is technically called as "pre instantiation of servlet program"

Question: What is the advantage of enabling <load-on-startup> on servlet program (or) what is the advantage of pre-instantiation of servlet program?

Ans: When <load-on-startup> is not enabled first request given to servlet program participates in instantiation, (our servlet class object creation) initialization(creating ServletConfig object and calling init(-)) before processing request by calling service (-,-) method. where as other than first request given to same servlet program directly participates in request processing by calling service (-,-). Due to this the response time (time taken by servlet to generate webpage) of first request is little bit higher when compared to other than first request.

◇ To minimize response time of first request and to equalize with other than first request make servlet container performing instantiation & initialization operations on our servlet class before first request either during server startup on during the deployment of web application by enabling <load-on-startup> on servlet program.

NOTE:- ◇ It is recommended to enable <load-on-startup> only on those servlets for which there is a guarantee that they will be requested immediately after deployment of web application like servlet program which generate home page, main menu page etc.,

Question: when Servlet container creates our servlet class object?

Ans: When <load-on-startup> is not enabled:

1. when our servlet program gets 1st request from client.
2. when servlet program gets 1st request after reloading/redeploying the web application.
3. when servlet program gets 1st request after restarting the server.

When <load-on-startup> is enabled:

1. Either during server startup or during the deployment of the web application.
2. When web application is reloaded/redeployed.
3. When server is restarted.

Question: When servlet container destroys our servlet class object?

1. When web application is reloaded/redeployed.
2. When server is crashed
3. When server is shutted down/restarted.

4. If our servlet class object is continuously idle for long time.

Note: No special importance will be there towards destroying object when <load-on-startup> enabled on our servlet program.

Question: What is the use of <load-on-startup> priority value?

Ans: When multiple servlet programs of java web application are enabled with <load-on-startup> then servlet container performs ~~per~~ Pre-instantiation (initialization) on that servlet programs based on this <load-on-startup> priority values (decides the order of creating objects) either during server startup or during deployment of the web application.

◇ from Tomcat 6.x server onwards.
- —> High value indicates low priority
- —> Low value indicates high priority
- —> Negative value ignored

Note: no special meaning for "0"

◇ Before tomcat 6.x server
- —> High value indicates low priority
- —> Low value indicates high priority
- —> negative value ignores the <load-on-startup> that is enabled
  o –indicates least/last priority.

Example Scenario –I (for Tomcat 6.x)

(Load-on-startup)

srv1	1 (1)	4 (II)	4	Server decides the order by using its own algorithms
srv2	10 (IV)	6 (III)	4	
srv3	5 (II)	0 (I)	4	
srv4	7 (III)	-4 ignores	4	

srv1, srv2, srv3, srv4 are servlet programs of web application.

Example scenario –II (for Tomcat 5.x)

	<l-o-s>	<l-o-s>	<l-o-s>	
~~Server~~ Srv 1	1 (1) (IV)	4 (II) (1)	4	Server decides the order
~~Server~~ Srv 2	10 (IV) (I)	6 (III) (2)	4	
~~Server~~ Srv 3	5 (II) (III)	0 (I) (3)	4	
~~Server~~ Srv 4	7 (III) (II)	-4 ignores	4	

◇ Enabling <l-o-s> on servlet <l-o-s> Tomcat 6.x

—>	LcTestSrv	10(IV)	program with negative priority value is equal to not enabling <l-o-s> on servlet program
—>	HtmlSrv	6 (III)	
—>	XmlSrv	-2 (ignores <l-o-s>	
—>	WordSrv	0 (I)	
—>	ExcelSrv	1 (II)	

◇ We can take <load-on-startup> without priority value but it is not recommended to take.

◇ In tomcat 5.x <load-on-startup> without priority value always gets 1st priority.

◇ From tomcat 6.x onwards <load-on-startup> without priority value gets second priority if zero based <load-on-startup> is there otherwise it gets first priority.

→ life Cycle methods are Useful in two angles for programers

① To place programer choice logics to generate webpages in Servlet programs

② To Access and use ServletContainer Supplied Objects like request, response Objects in ServletProgram by receiving them as Parameters of lifecycle methods

→ The prototype of lifecycle methods :-

```
public void init(ServletConfig cfg) throws ServletException
public void Service(ServletRequest req, ServletResponse res) throws ServletException,
                                                                   IOEXCeption
public void destroy()
```

Q:- What is the purpose of each lifecycle method of our Servlet Program?

Ans:- refer Page (1) of suplimentary handout given on

Q:- Can you explain what happens when our Servlet program gets 1st request & other than 1st request from browser window?

Ans:- refer both Question & Answers of page no: ② of Suplimentary handout given on 25/09/2012

→ Adding new Servlet program to the existing MIMEApp application (i.e. 2nd App) to understand the lifecycle of Servlet Program.

Step-I :- make Sure that MIMEApp aplication (App ② refer 17/09/2012) is deployeed in Tomcat Server

Step-II :- Develop the following Servlet program in WEB-INF/classes folder of deployeed MIMEApp application.

```
// LcTestSrv.java
Package P1;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
```

```
Public class LcTestSrv extends HttpServlet
{
    public LcTestSrv()
    {
        S.O.P("LcTestSrv: 0-Param Constructor");
    }
    public void init(ServletConfig cg)
    {
        S.O.P("LcTestSrv (-)");
    }
    Public void service(ServletRequest req, ServletResponse res) throws SE,IOE
    {
        // get PrintWriter Obj
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
        //Write response to brocoser window
        pw.println("<b> Date and Time </b>" + new java.util.Date());
        S.O.P("LcTestSrv: Service(-,-)");
        // close Stream
        pw.close();
    } //service(-,-)
    Public void destroy()
    {
        S.O.P("LcTestSrv: destroy()");
    }
} //class
//> javac -d . LcTestSrv.java
```

Step-III :- Configure the above servlet program in web.xml file

In web.xml

```
<servlet>
    <s-n> Lc </s-n>
    <s-c> P.LcTestSrv </s-c>
</servlet>
```

```
<servlet-mapping>
    <s-n> lc </s-n>
    <u-p> /lcurl </u-p>

</s-m>
```

Step-IV :- Start the Server and give request to Servlet Program

Request url: http://localhost:2020/MIMEAPP/Lcurl

26/09/2012

→ When <load-on-startup> is enabled on a servlet program during its configuration in web.xml then the Servlet Container creates and initializes object of that Servlet class either during Server start-up (or) during deployement of the web application. This process is called pre-instantiation and initialization of Servlet program.

To enable <load-on-startup> on Servlet program :-

```
<servlet>
    <servlet-name> lc </s-n>
    <servlet-class> P1.LcTestSrv </s-c>
    <load-on-startup> 1 </load-on-startup>
</servlet>
```

↳ priority value

```
<servlet-mapping>
    <servlet-name> lc </s-n>
    <url-pattern> /lcurl </u-p>
</servlet-mapping>
```

→ if webapplication is deployed in a Server i.e. already in Running mode then Servlet Container creates Objects on (pre-instantiation and Initialization) <load-on-startup> enabled Servlet programs during the deployement of the web application.

→ if Server is started after deployeing the webapplication then Servlet Container performs pre-instantiation and initialization on

<load-on-startup> enabled Servlets program during the Server Startup.

Q:- What is <load-on-startup> and what is the advantage of enabling <load-on-startup> on our Servlet program?

Ans:- Refer page No: 2 & 3 of Suplimentary handout i.e. 25/09/2012

Q:- When Servlet Container creates Our Servlet Class obj?

Ans:- Refer page No: 3 of 25/09/2012 Suplimentary handouts.

Q:- when Servlet Container destroyes our Servlet class obj?

Ans:- Refer page No: 3 & 4 of 25/09/2012 Suplimentary handout.

Q:- What is the Use of priority Value placed in <load-on-startup>?

Ans:- Refer page No: 4 of 25/09/2012 handout

27/09/2012

→ If there is no <load-on-startup> priority value the Container takes (0) "zero" as the default preority value.

→ Container does not raise lifecycle event by using the logics placed in lifecycle methods. Container maintains its own code to perform lifecycle events. but it also calls lifecycle methods of Servlet program to execute the programmer supplied logics.

Q:- what happents if destroy() is called explicitly from Service(-,-)?

Ans:- Servlet Container doesnot destroy our Servlet class object. but logics of destroy() executes along with Service(-,-)

NOTE:- Container Calls lifecycle methods when lifecycle events are raised when lifecycle methods are called manually the container doesnot raise life cycle events.

Q:- what happents if init(-) is called from Service(-,-) explicitly?

Ans:- Servlet Container doesnot create new object for our Servlet Class

but the logics of init(-) will be executed along with Service(-,-)

Q:- How Servlet program is executing without main()?

① In standalone applications main() is required to begin the application execution. Servlet program is not standalone application. so, the main() is not required in servlet program.

② Servlet Container executes our servlet program through lifecycle methods. Since main is not lifecycle method. so, it is not required in servlet programing.

③ If a Running Java application wants to create Object of another java class and wants to call methods on it. then there is no need of main() another Java class. Servlet Container is a Continuously Running java application to create Our ServletClass object and to call its methods. So, main() is not required in Our Servlet program.

*) In Tomcat Server the ServletContainer executes the Servlet program as shown below

programer clicks on tomcat6.exe file ⟶ this locates bootStrap.jar file ⟶ by reading manifest file the org.apache.catalina.startup.BootStrap class will be located ⟶ main(-) of this class executes ⟶ this main(-) starts server and Container ⟶ This Container becomes ready to execute our Servlet program by creating Object of Servlet class and by calling lifecycle methods on that Object.

Q:- What happenes when main(-) is placed in Our servlet program?

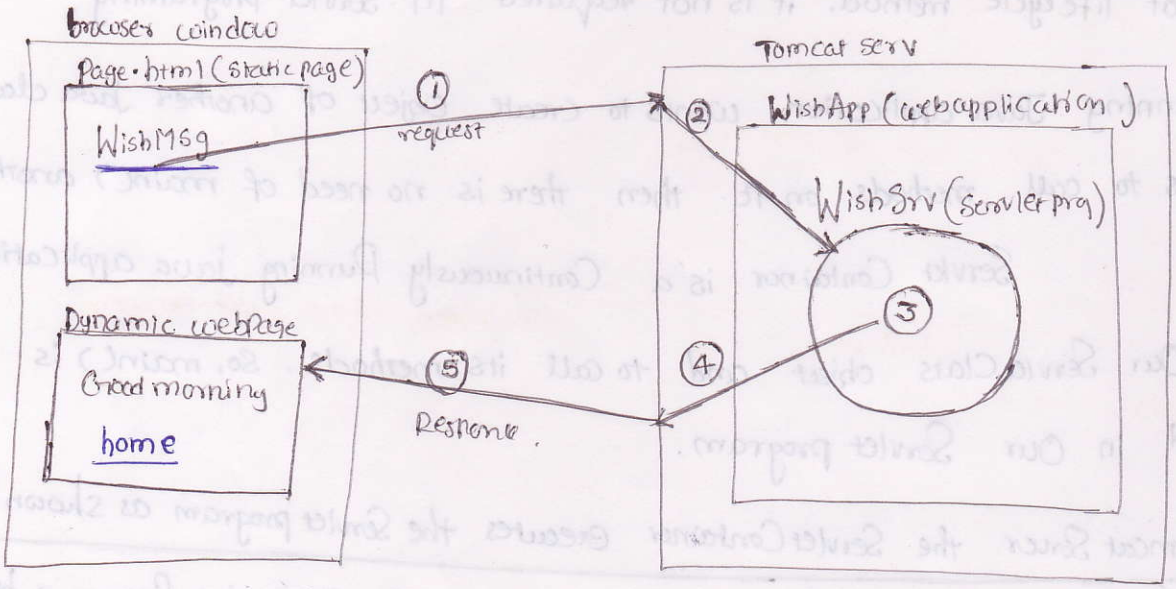
Ans:- Since main(-) is not the lifecycle method. so, the Servlet Container never calls that method during the execution of Servlet program.

If you want to supplies input values to servlet programs from browser window so far we have used query String. But working with this query String is very complex for non-technical endusers like civil engineers, Chemical engineers and etc.

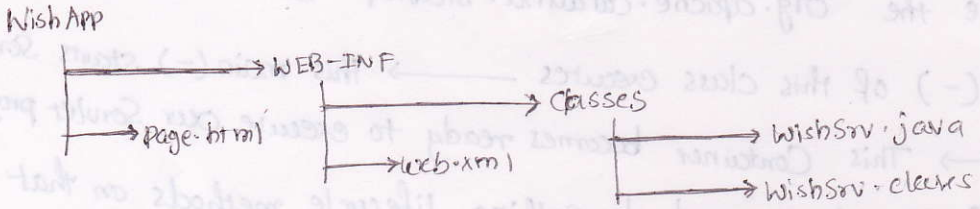
To overcome this problem use HTML programs based webpages having hyperlinks core forms to generate the request to Servlet program. For this we need to go for "HTML to Servlet Communication".

→ HyperLink generated request does not carry any data by default where as form generated request carries data (form data)

Example App on Html → Servlet Communication using hyperlinks.



Deployement Directory Structure:



→ place html, JSP programs under webroot folder of our webapplication

→ place the above "WishApp" folder in <Tomcat_home>\webapps folder

→ html prgms need not be cfg in web.xml i.e. they will identified directly through their file names.

28/09/2012

→ WEB-INF is private directory of webapplication. So, the resources placed inside WEB-INF and its subfolders will be recognized by Container as webResource programs only when these are Configured in web.xml file.

Eg:- Servlet programs.

→ The WebResource programs that are placed outside the WEB-INF folder will be recognized by Server automatically. So, Their Configuration in web.xml file is Optional.

Eg:- Html, Jsp programs.

*) Source code of above web application :-

```
// page.html
<a href="http://localhost:2020/WishApp/wurl">WishMsg</a>
```

↳ href url ↳ url pattern of the servlet program

```
// WishSrv.java
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
import java.io.*;
public class WishSrv extends HttpServlet
{
protected void service(HttpServletRequest req, HttpServletResponse res) throws SE,IOE
{
    // general Settings
    PrintWriter pw = res.getWriter();
    res.setContentType("text/html");

    // Write b.logic to generate response.
    Calender cl = Calender.getInstance(); // gives current date and time
    int h = cl.get(Calender.HOUR_OF_DAY); // gives current hour (24 hours format)
```

↳ constant

```
    String msg = null;
    if(h<12)
        msg = "Good morning";
    else if(h<18)
```

```
        msg = "Good after noon";
    else if (h < 22)
        msg = "Good Evening";
    else
        msg = "Good night";

// This msg is should be printed on the browser (display response)
    pw.println("<font color='red' size='5'> +msg+ "</font>");

// close Stream
    pw.close();
  } // service(-,-)
} // class
```

→ // web.xml

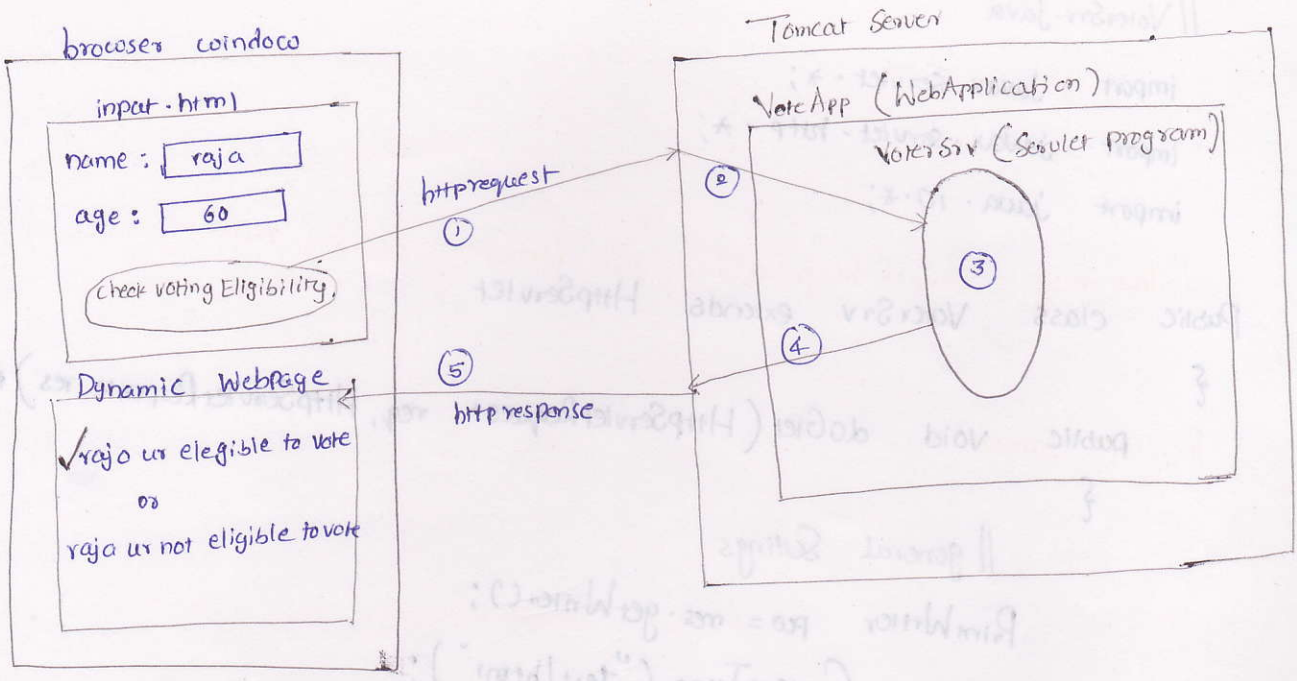
```
<web-app>
  <servlet>
    <servlet-name> abc </servlet-name>
    <servlet-class> WishSrv </servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name> abc </servlet-name>
    <url-pattern> /wurl ~~web~~ </url-pattern>
  </servlet-mapping>
</web-app>
```

→ deployee the above WishApp application in tomcat Server and Start the Server.

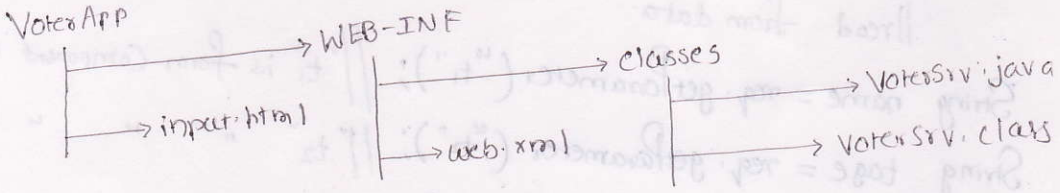
→ Request Url to test the above application

http://localhost:2020/WishApp/page.html

*) HTML form page to Servlet Communication :-



Directory structure:-




```
//input.html

<form action="http://localhost:2020/VoterApp/vturl" method="get">
```

↳ Action Url

↳ url pattern of VoterSrv program

form components:

```
Name : <input type="text" name="t1"/> <br>
Age : <input type="text" name="t2"/> <br>
<input type="submit" value="Check Voating Eligibility">

</form>
```

NOTE1 :- → When form is submitted the form components names and its values goto server as request parameter names and values

Eg:- t_1 = raja & t_2 = 30

↳ request param names

↳ request param values

→ form page can send request to target webResource program either in "get" mode or in "post" mode

↳ Can send limited amount of data (max of 256 kb)

↳ Can send unlimited amount of data

→ It is the recomended to process the "Get" mode request in Servlet program using doGet(--,--)

4) Similarly use doPost(-,-) to process Post mode request.

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

Public class VoterSrv extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws SE, IOE
    {
        // general Settings
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        //read from data
        String name = req.getParameter("t1"); // "t1" is form Component name.
        String tage = req.getParameter("t2"); // "t2" " " " "
        int age = Integer.parseInt(tage);

        //write b.logic (or) request processing logic
        if (age >= 18)
            pw.println("<b><center><font color=green size=6>" + "Hame" + " ur eligible to vote</font></center></b>");
        else
            pw.println("<b><center><font color=red size=6>" + name + " ur not eligible to vote</font></center></b>");
        pw.println("<a href="http://localhost:2020/VoterApp/input.html"> home</a>");
        // Close Stream
        pw.close();
    }// doGet(-,-)
}// class
```

recompilation

Compilation

```
// web.xml
<web-app>
    <servlet>
        <s-n> xyz </s-n>
        <s-c> VoterSrv </s-c>
    </servlet>
```

```
<servlet-mapping>
    <s-n> xyz </s-n>
    <u-p> /vturl </u-p>
<servlet-mapping>
</web-app>
```

Request URL to test the Application.

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

29/09/2012

→ Inorder to ~~[illegible]~~ make our webapplication deployable in any server irrespective of the modifications done server port number and Host name (or) IP address of server computer we need to work with relative URL's instead of absolute URL's.

→ We can make the Java webapplications WODA ( ) by utilizing relative uri's

Scenario (1):-

(1) <form action = "http://localhost:2020/VoterApp/vturl" method = "get">

- - - -

- - -

</form>

→ Hyper Links Generated request are "get" ~~method~~ mode request. where as the form page generated request are post mode request (or) "get" mode request.

Q:- How to make our servlet program as flexible servlet program handling with both "get" mode and "post" mode request?

Approach (1):- keep request processing logic in servlet prgm by overriding service(-,-) method.

```
public class VoterSrv extends HttpServlet
{
    public void service(-,-) throws SE,IOE
    {
        ---
        ---  //request processing logic.
    }
}
```

NOTE:- Service(-,-) method can handle both "GET", "POST" mode requests

NOTE:- keeping request processing logic in service(-,-) is not industry standard. Use doXxx(-,-) like doGet(-,-), doPost(-,-) for this operation.

Approach(2):- Override both doGet(-,-) and doPost(-,-) methods in our servlet program. and keep request processing logic in one method and call that method from another method.

```
public class VoterSrv extends HttpServlet
{
    public void doGet(-,-) throws SE,IOE
    {
        ---
        ---
        ---
    }
    public void doPost(-,-) throws SE,IOE
    {
        doGet(req, res);
    }
}
```

While working with service(-,-) method there is no possibility to write separate request processing logics for "Get", "post" method based requests. but this is possible while working with doGet(-,-), doPost(-,-) methods

we can write "Get" method based request in doGet(-,-) method and we can write "Post" method based request in doPost(-,-) method.

Approach(3):- keep request processing logic in userdefined method of servlet programs and call that method from both doGet(-,-), doPost(-,-) methods. This user defined method must have same signature of doXxx() methods.

```
public class VoterSrv extends HttpServlet
{
 public void xyz( HSR req, HSR res) throws SE,IOE.
 {
    ---
    ---   //request processing logic
    ---
    --
 }
 public void doGet(-,-) throws SE,IOE.
 {
    xyz(req, res);
 }
 public void doPost(-,-) throws SE,IOE
 {
    xyz(req, res);
 }
}
```

→ Slw industry prefers either approach (2) or approach (3) while developing servlet programs

Q:- what happents if you modify source code of the home page/webpage by using the view Source option of browser settings!

Ans: if the webpage is accessed from the webapplication deployed in webserver then modifications will not be reflected. even though you refresh the page.

If the webpage is accessed from the local .html which is not placed in the server then the modifications will be reflected.

NOTE:- Some servers violating the single instance multiple threads component principle of servlet programing by creating more than one object for our servlet class.

Eg:-

Tomcat server creates 2nd object for our servlet class when it requires 150+ requests simultaneously (or) concurrently (or) parallaly from clients

05/10/2012

Reading data from different kinds of form components being from Servlet program :-

① Text Boxes :-

In form page :-

Name : <input type="text" name="t1">
↳ logical name

In Servlet prg

String s1 = req.getParameter("t1");

② Password Boxes :-

In form page :-

Age : <input type="password" name="t2">
↳ logical name

In Servlet prg

String s2 = req.getParameter("t2");

③ TextAreas :-

In form page :-

Address : <textarea rows="10" cols="20" name="t3">
↳ logical name

enter address

</textarea>

In Servlet prg

String addrs = req.getParameter("t3");

④ RadioButtons :-

When multiple RadioButtons are grouped into Single unit they allow us to select only one RadioButton at a time. By giving Samename for multiple Radio buttons we can group them into Single unit.

In formpage :-

Gender : <input type="radio" name="gen" value="M" selected> Male
↳ value ↳ lable

<input type="radio" name="gen" value="F">FeMale
↳ value ↳ lable

In servlet program :-

String g1 = req.getParameter("gen");

"g1" holds value "M" when Male radio button is clicked.

"g_1" holds value "F" when Female radio btn is clicked.

NOTE:- When Radio button is clicked its label will not goto Server as request parameter value but the value placed in value attribute will goto Server as request parameter value.

⑤ Check Boxes :-

NOTE:- Eventhough multiple checkBoxes are grouped into Single unit there is a possibility to Select multiple check boxes at a time.

In form page :-

```
hobbies : <input type="checkbox" name="hb1" value="roaming" checked> Travelling
          <input type="checkbox" name="hb1" value="sleep"> Relaxation
          <input type="checkbox" name="hb1" value="watching"> Watching TV.
```

↳ values ↳ label

When check box is Selected its label will not goto Server as Request Parameter Value but its value kept in Value attribute goes to Server as Request parameter value.

In Servlet Prg :-

```
String p[] = req.getParameterValues("hb1");
```

→ if all the three check boxes are selected then p[] holds roaming, sleep, watching as the values.

p[]	
roaming	0
sleep	1
watching	2

⑥ Select Box / Combo Box :-

In form page :-

```
<select name="qlfy">
    <option value="engg">B.E/B.Tech </option>
    <option value="medico">MBBS/BDS </option>
    <option value="arts">BA </option>
    <option value="Commerce">B.com / M.com </option>
</select>
```

↳ value ↳ label

In Servlet Prg :- `String s1 = req.getParameter("qlfy");` // gives engg as Value when B.E/B.Tech item is selected.

→ when an item of select box is selected then its label will not goto Server as Request Parameter value. but the value placed in its attribute "value" will goto Server as Request Parameter value.

⑦ ListBox :-

Select Box allowing to select multiple items at a time

In form page :-

```
<select name="courses" multiple>
    <option value="java">JAVA PKG</option>
    <option value=".net">.NET PKG</option>
    <option value="oracle">Oracle PKG</option>
</select>
```

↳ standalone attribute.

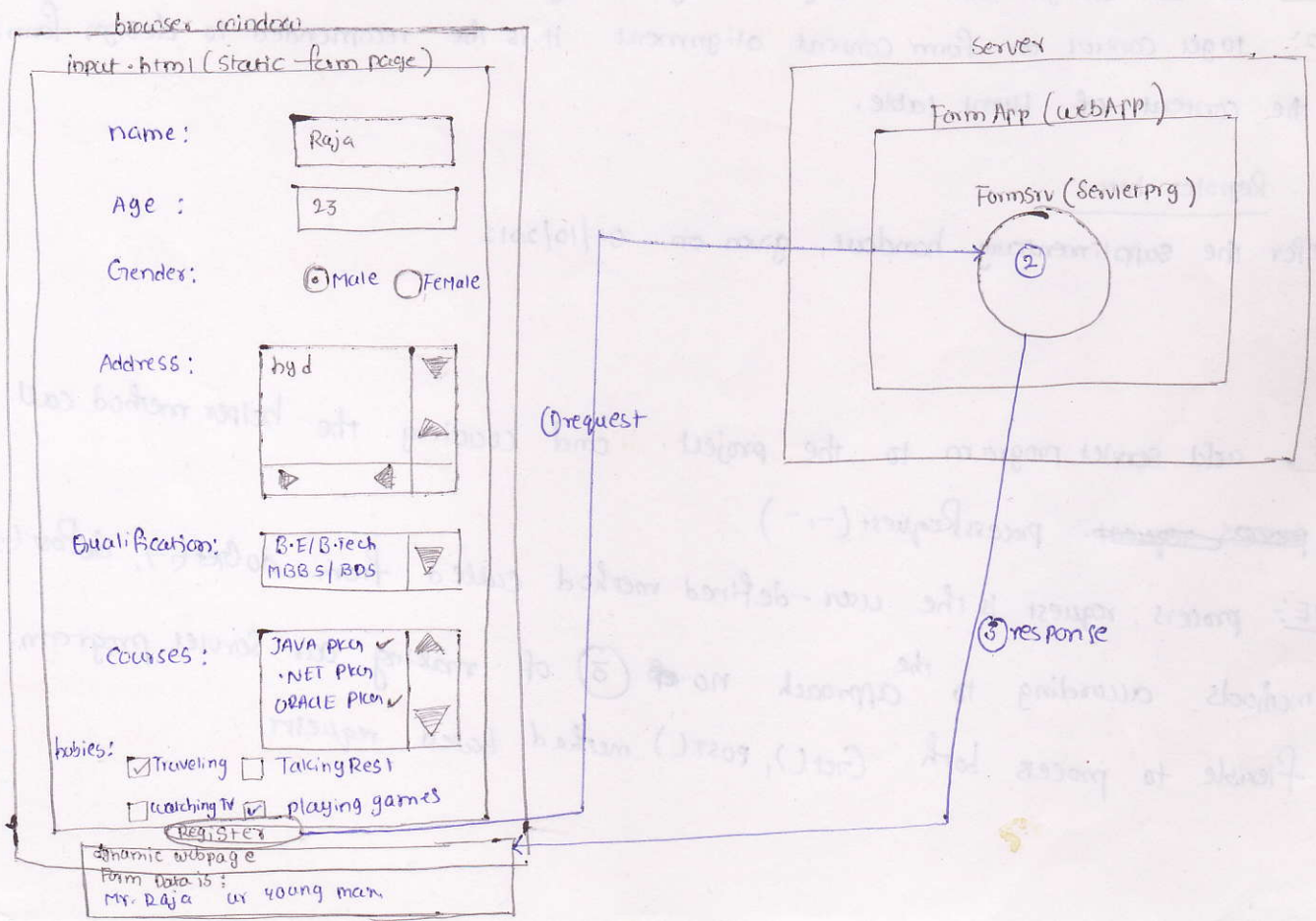
NOTE:- we can select multiple items by holding the Cntrl + P.

In Servlet prg :-

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

if java pkg, ~~.NET PKG~~ Oracle pkg items are choosen then crs[] holds java, Oracle has the elements values.

*) Example application on HTML to Servlet Communication.



NetBeans :-

Type : IDE slw for java

Version : 6.7.1 (Compatible with jdk 1.8 +)

Vendor : sun ms (oracle corp)

Opensource

gives GlassFish 2.1 as built-in Server

to download slw and to get help;

Procedure to develop above application by using NetBeans IDE.

Step-I :- create web project in NetBeans IDE

File Menu → New project → Java web → webapplication → next →

Project Name: FormApp → next → Server : GlassFish 2.1 → next → Finish

→ delete Index.jsp.

Step-II :- add Register.html to the webpages folder of the project.

Expand project → Right click on webpages → New → HTML → HTML File name: register

→ Finish

NOTE 1:- we can design this form page through drag and drop operation.

NOTE 2:- to get control on form content alignment it is the recomended to design form page as the content of Html table.

Register.html

refer the supplimentary handout given on 04/10/2012

Step-III:- add servlet program to the project. and coading the helper method call

~~process request~~. processRequest(--)

NOTE:- process request is the user-defined method called from doGet(--), doPost(--) methods according to the approach no. (3) of making our Servlet program flexible to process both Get(), POST() method based requests.

4/10/12

```
<<<<<<<<<<<<<<  Reading data from diff types of Form comps<<<<<<<<<<<<
-----------------register.html------------------------
<form action="furl" method="get">
 <table border="1">
      <tr>
         <td>Name:</td>
         <td><input type="text" name="tname" value="" /></td>
      </tr>
      <tr>
         <td>Age:</td>
         <td><input type="password" name="tage" value="" /></td>
      </tr>
      <tr>
         <td>Gender:</td>
         <td>
            <input type="radio" name="gen" value="M" checked /> Male
            <input type="radio" name="gen" value="F"/>FeMale
         </td>
      </tr>
      <tr>
         <td>Address:</td>
         <td>
            <textarea name="taddress" rows="4" cols="20">
               enter address
            </textarea>
         </td>
      </tr>
      <tr>
         <td>Qualification:</td>
         <td>
            <select name="qlfy">
               <option value="Engg">Engineer</option>
               <option value="medicine">Doctor</option>
               <option value="arts">Graduate</option>
               <option value="commerice">CA</option>
            </select>
         </td>
      </tr>
      <tr>
         <td>Courses:</td>
         <td>
            <select name="crs" size="4" multiple>
               <option value="java">JAVA PKG</option>
               <option value=".net">.NET pkg</option>
               <option value="oracle">Oracle PKG</option>
               <option value="SAS">SAS</option>
            </select>
         </td>
      </tr>
      <tr>
         <td>Hobies</td>
         <td>
            <input type="checkbox" name="hb1" value="roaming" checked />  Travelling
            <input type="checkbox" name="hb1" value="sleep" />  Relax
            <input type="checkbox" name="hb1" value="watching"/> Watching TV
            <input type="checkbox" name="hb1" value="playing"/> Games
         </td>
      </tr>
      <tr>
```

```
60          <td colspan="2"><input type="submit" value="register" /> </td>
61        </tr>
62    </table>
63  </form>
64  ------------------------------FormSrv.java-----------------
65  import java.io.IOException;
66  import java.io.PrintWriter;
67  import javax.servlet.ServletException;
68  import javax.servlet.http.HttpServlet;
69  import javax.servlet.http.HttpServletRequest;
70  import javax.servlet.http.HttpServletResponse;
71
72  public class FormSrv extends HttpServlet {
73
74     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
75     throws ServletException, IOException {
76        // general settings
77        PrintWriter pw=response.getWriter();
78        response.setContentType("text/html");
79        // read form data
80        String name=request.getParameter("tname");
81        int age=Integer.parseInt(request.getParameter("tage"));
82        String gen=request.getParameter("gen");
83        String addrs=request.getParameter("taddress");
84        String qlfy=request.getParameter("qlfy");
85        String crs[]=request.getParameterValues("crs");
86        String hb[]=request.getParameterValues("hb1");
87        //write b.logic /request processing logic
88        String msg=null;
89        if(gen.equalsIgnoreCase("M"))
90        {
91           if(age<=5)
92              msg="Master."+name+" u r baby boy";
93           else if(age<=12)
94              msg="Master."+name+" u r small boy";
95           else if(age<=19)
96              msg="Mr."+name+" u r teenage boy";
97           else if(age<=35)
98              msg="Mr."+name+" u r young man";
99           else if(age<=50)
100             msg="Mr."+name+" u  r Middle aged man";
101          else
102             msg="Mr."+name +" u r old man";
103       }
104       else
105       {
106          if(age<=5)
107             msg="Master."+name+" u r baby girl";
108          else if(age<=12)
109             msg="Master."+name+" u r small girl";
110          else if(age<=19)
111             msg="Miss."+name+" u r teenage girl";
112          else if(age<=35)
113             msg="Miss/Mrs."+name+" u r young Girl/woman";
114          else if(age<=50)
115             msg="Miss/Mrs."+name+" u  r Middle aged woman";
116          else
117             msg="Miss/Mrs."+name +" u r old woman";
118       }
```

```
119         // display the form data
120         pw.println("<br>Name="+name);
121         pw.println("<br>Age="+age);
122         pw.println("<br>Address="+addrs);
123         pw.println("<br>Gender="+gen);
124         pw.println("<br>Qualification="+qlfy);
125         pw.println("<br>the Courses are <br>");
126         if(crs!=null)
127         {
128            for(int i=0;i<crs.length;++i)
129            {
130            pw.println(crs[i]+"....");
131            }
132         }
133         pw.println("<br>the hobies are <br>");
134         if(hb!=null)
135         {
136            for(int i=0;i<hb.length;++i)
137            {
138            pw.println(hb[i]+"....");
139            }
140         }
141         pw.println("<br><br>");
142         pw.println("<b><i><u><center>"+msg+"</b></i></u></center>");
143        //close stream
144         pw.close();
145      }//doGet(-,-)
146
147
148      protected void doGet(HttpServletRequest request, HttpServletResponse response)
149      throws ServletException, IOException {
150         processRequest(request, response);
151      }
152
153     protected void doPost(HttpServletRequest request, HttpServletResponse response)
154      throws ServletException, IOException {
155         processRequest(request, response);
156      }
157
158   }
159   ------------------------------------web.xml---------------------------
160   <web-app>
161   <servlet>
162         <servlet-name>FormSrv</servlet-name>
163         <servlet-class>FormSrv</servlet-class>
164      </servlet>
165      <servlet-mapping>
166         <servlet-name>FormSrv</servlet-name>
167         <url-pattern>/furl</url-pattern>
168      </servlet-mapping>
169   </web-app>
170
171
```

Right click on project → New → Servlet → ClassName: [FormSrv] → next → Url patterns [/fun] → finish → write following code in processRequest(--)

Step-4 :- Run the project.

Step-5 :- Right Click on project → RUN

NOTE:- for the above steps based complete application refer the supplimentary handout (04/10/2017)

Except listBox, checkBox components the remaining components of form frams request parameters having ~~Emp~~ Empty strings as values even though they are not buildup (or) no item is selected. but this won't happen for checkboxes and listboxes

Form Validations :-

The process of verifying format and pattern of the data is called Form Validation. and logic used for it is called by Form Validation logic.

Eg:-

→ checking weather required fields are typed or not

→ checking weather age is given as numeric value or not

→ checking email-id is having '@', '_' symbols or not and etc.

→ It is recomended to use the validated form data as input values of B.logic otherwise B.logic may raise exceptions or wrong results.

Q:- What is the diff b/w Form Validation logic and B.logic?

Ans:- ·) Form Validation logic verifies the format and pattern of the data.

·) where as B.logic generates results by using form data.

Eg:- checking weather a/b values are type or not as numeric values is called Form validation logic.

using a/b values as inputs and finding their sum is called as B.logic.

FormValidation logics

1. Client-side logics

→ ~~[illegible]~~ comes to browser window for execution.

→ place these logics .html files using JavaScript and VBScript code

2. Server-side logics

→ resides and executes in servlet/jsp prgs

→ place these logics as java stmts in Servlets/JSP prgs.

NOTE :-

→ JavaScript is good to write Client-side form validation logic

→ prefer client side form validations logics because it executes in browser window and also reduces n/w round trips b/w browser-window and web server

→ There is a posibilities of Disabiling Script code execution through browser settings then writing only client-side FormValidation logics is not recomended.

To solve this problem there are two solutions

Solution 1:

→ write both client-side and server-side form validation logics. (This performs Server Side form Validations eventhough client side script code execution is disabled).

→ This approach may degrade the performance Bcz Same Form Validation logics executes at both client side and server side.

Solution 2:

→ Display error messages and stop sending the request to server until script code execution is enabled through browser settings.

→ when method return type is Void and if you place return statement with out return value then that stmt transfer the control from current method definition to its Calling ~~[illegible]~~ routine

05/10/2012

→ onsubmit, on click, on dbl click and etc are javascript events

→ when submit button is clicked onsubmit event is raised we generally call one userdefined javascript function against this onsubmit event to perform the client side form validations.

→ By adding <noscript> tag in form pages we can display error messages when script code execution is disabled through browser settings.

<noscript> <font color=red> enable java script </font> </noscript>

→ To disable script code execution IE:-

Tools → Internet options → security Tab → Custom level → scripting → Active scripting → ⊙ Disable → Ok → Ok.

→ To avoid the Javascript code visibility to enduser using View source option of browser window It is recommended to place javascript code in a separate file and link that file with .html file (formpage) by using the "src" attribute of <script> tag.

→ For example application on Client side and Server side FormValidation the supplimentary handout given on 05/10/2012.

Q:- How to disable server side FormValidation logic execution when client side form Validation logic are taken place:

Ans:- make form page sending a flag to server side indicating FormValidations are done through Java script support. based on this flag programmer can write logics to avoid execution of ServerSide form Validation logics.

refer line no 13, 44, 45, 81 lines code of handout given on 05/10/2012

*) Html to Servlet Communication :-

06/10/2012

The First page \ default page of the WebApplication is called as Home page (or) welcome page. This page Comes automatically the movement will give request to Website.

We can Configure html program (or) Jsp program (or) Servlet program as welcome file (or) Welcome page in web.xml file as shown below.

In web.xml :-

```
<|servlet-mapping>
<welcome-file-list>
        <welcome-file> Details.html <|welcome-file>
<|welcome-file-list>
<|web-app>
```

We can Specify multiple files as welcome files but only one file will be taken as welcome file at a time.

In web.xml :-

```
<welcome-file-list>
        <welcome-file> Details.htm <|welcome-file>
        <welcome-file> ABC.html <|welcome-file>
        <welcome-file> XYZ.jsp <|welcome-file>
        <welcome-file> Details.html <|welcome-file>
<|welcome-file-list>
```

NOTE:- When multiple welcome files are configured, Server picks up the welcome file based on the availability and configuration order of welcome files.

→ If no <welcome-file> files are configured in web.xml file then the servlet container Server looks to take index.html (or) index.jsp as default welcome file.

→ If both are available the priority will be given to index.html

Q:- if all explicitly configure welcome files are not available and index.html & index.jsp files are available then can you tell me what happents?

05/10/2012

L1-UU-1L

```
>>>>>>>>>>>>>VoterApp Application with client side and server side Validations>>>>>>>>>>>>>>
-----------------------------------Details.html-----------------------------------
<!-- Details.html -->
<html>
 <head>
   <noscript> <font color=red> enable java script </font> </noscript>
   <script language="JavaScript" src="myValidation.js">
   </script>
</head>
  <b><center> Person Details </center></b>
<br><br>
<form action="vturl" method="post" onsubmit="return validate(this)">
 Person  Name: <input type="text" name="pname"><br>
 Person age: <input type="password" name="page"><br>
 <input type="submit" value="Check Elgibility">
</form>
-----------------------------------myValidation.js-----------------------------------
  function validate(frm)
      {
        // read form data
        var  name=frm.pname.value;
        var age=frm.page.value;
        // form validation logic (client side)
         if(name=="")  //required rule on name
         {
           alert(" person  name is required");
           frm.pname.focus();
           return false;
         }
        if(age=="") // required rule on age
        {
          alert("person age is required");
          frm.page.focus();
          return false;
         }
         else
         {
            if(isNaN(age))
            {
              alert(" age must be a numeric value");
              frm.page.focus();
              frm.page.value=""; // empties the text box value
              return false;
              }//if
         }//else
         return true;
      }//function
-----------------------------------web.xml-----------------------------------
<web-app>
 <servlet>
   <servlet-name>abc1</servlet-name>
   <servlet-class>VoterSrv</servlet-class>
 </servlet>
 <servlet-mapping>
   <servlet-name>abc1</servlet-name>
   <url-pattern>/vturl</url-pattern>
 </servlet-mapping>
 <welcome-file-list>
   <welcome-file>Details.html</welcome-file>
</welcome-file-list>

 </web-app>
-----------------------------------VoterSrv.java-----------------------------------
//VoterSrv.java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class VoterSrv extends HttpServlet
{

   public void doGet(HttpServletRequest req,HttpServletResponse res)throws ServletException,IOException
   {
      System.out.println("VoterSrv:service(-,-)");
      //set response content type
      res.setContentType("text/html");
```

Client side form validation logic.

form page

File having Javascript Code

<input type="hidden" name="h1" value="disabled" />

File having Javascript code

frm.h1.value="enabled";

Deployment Descriptor file

Servlet program

(*) This statement is calling Validate( ) function having current form object as argument against onsubmit event. The return statement placed here catches true/false given by validate function and gives that value to browser window. Browser window blocks the control flow when the return value is false (when form validation errors are there). Browser window sends the control to server when this return value is true (No form validation errors)

```
76        // get PrintWriter obj
77        PrintWriter pw=res.getWriter();
78        //read form data from the form page generated request
79        String name=req.getParameter("pname");
80        String tage=req.getParameter("page");
81        String isstatus = req.getParameter("hi");
          int age=0;
82        //write form validation logic  -> if(isstatus.equals("disabled")){
83           //required rule on person name
84        if(name.length()==0 || name==null || name.equals(""))
85        {
86           pw.println("<font color=red size=2 > Person name is required </font>");
87           return ;
88        }
89
90        //required rule person age
91        int age=0;
92        if(tage.length()==0 || tage==null || tage.equals(""))
93        {
94           pw.println("<font color=red size=2> Person Age is required </font>");
95           return ;
96        }
97        else   // value must be numeric rule on age
98        {
99           try{
100          //convert given age value to numeric value
101          age=Integer.parseInt(tage.trim());
102          }
103          catch(NumberFormatException nfe)
104          {
105             pw.println("<font color=red size=2>Age must be numeric value</font>");
106             return;
107          }//catch
108       }//else
109       }//else --- if
          age=Integer.parseInt(tage);
110       // write request processing logic/ B.logic
111       pw.println("<img src='Water lilies.jpg' width=700 height=100>");
112
113       if(age<18)
114          pw.println("<br><br><font color=red size=4>"+name+" u r not elgible to vote");
115       else
116          pw.println("<br><br><font color=green size=4>"+name+" u r elgible to vote");
117
118
119       pw.println("<br><br> <a href='Details.html'>home</a>");
120       pw.println("<br><br> <a href='Details.html'><img src=Sunset.jpg width=100 height=200/></a>");
121
122       //close streams
123       pw.close();
124    }//doGet(-,-)
125
126    public void doPost(HttpServletRequest req,HttpServletResponse res)throws ServletException,IOException
127    {
128       System.out.println("VoterSrv:doPost(-,-)");
129       doGet(req,res);
130    }
131
132    public static void main(String args[])
133    { System.out.println("VoterSrv:main(-)"); }
134
135 }//class
136 //>javac VoterSrv.java
137
138
139
```

Serverside form validation logic.

graphical hyper link

VoterApp
- → WEB-INF
  - → classes
    - → ~~VoterApp.java~~
    - → VoterSrv.java
    - → VoterSrv.class
  - → web.xml
- → Details.html
- → myValidation.js
- → Sunset.jpg
- → water lilies.jpg

Request url:-

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

Ans:- WebApplication Runs without welcome-file. moreover it will not take index.~~jsp~~ html

(or) index.jsp as default welcome-file.

→ While configuring JSP, html files as welcome-file we need to specify their file names. Similarly while configuring Servlet program as welcome-file we need to specify its Url pattern.

In web.xml file :-

```
<servlet>
    <servlet-mapping>
        <servlet-name> abc </servlet-name>
        <url-pattern> /test1 </url-pattern>
    </servlet-mapping>

    <welcome-file-list>
        <welcome-file> test1 </welcome-file>
    </welcome-file-list>
</web-app>
```

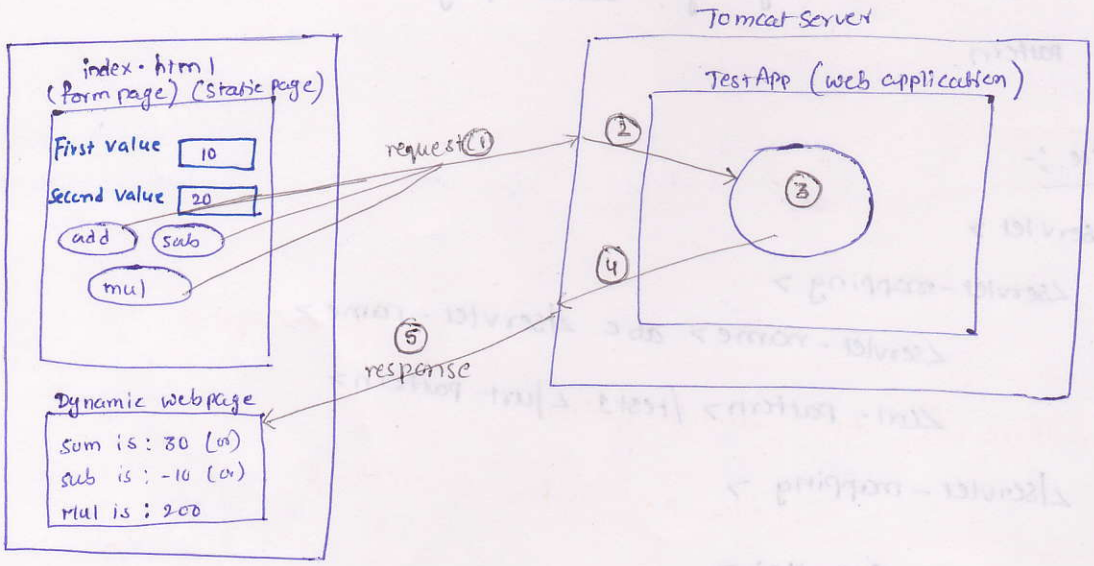
↳ Url pattern of Servlet program.

→ The submit button caption will not goto the Server as request parameter value by default. but it will go when you provide logical name to Submit button. Chaving Submit button name as RequestParam name and caption as RequestParam Value.

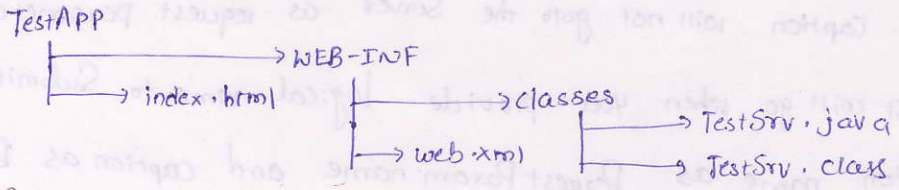
```
<form action="vturl" method="get">
    - - -
```

when form is submitted no request parameter related to submit button goes to server.

Q:- How to handle the situation when multiple submit buttons of a form page generating request to single servlet program?



→ Give same name for multiple submit buttons having different captions. Receive these captions as request param values in servlet program and use them to differentiate the logics of submit buttons.



Using NETBEANS IDE :-

index.html :-

```
<form action="turl" method="get">
    First value : <input type="text" name="t1"> <br>
    second value : <input type="text" name="t2"> <br>
    <input type="submit" name="s1" value="add"> <br>
    <input type="submit" name="s1" value="sub"> <br>
    <input type="submit" name="s1" value="mul"> <br>
</form>
```

TestSrv.java :-

```
public class TestSrv extends HttpServlet {

  protected void doGet (HttpServletRequest request, HttpServletResponse response) throws SE, IOE
  {
    // general settings
      response.setContentType("Text/html");
      PrintWriter pw = response.getWriter();
    // read form data
      int val1 = Integer.parseInt(request.getParameter("t1"));
      int val2 = Integer.parseInt(request.getParameter("t2"));
      String cap = request.getParameter("s1");   // gives caption
    // write b.logic for each submit button
    if (cap.equals("add"))   // when add button is clicked.
    {
      int result = val1+val2;
      pw.println("sum is:" +result);
    }
    else if (cap.equals("sub"))   // when sub button is clicked
    {
      int result = val1-val2;
      pw.println("sub is:" + result);
    }
    else   // when mul button is clicked
    {
      int result = val1 * val2;
      pw.println("mul is:" +result);
    }
    // close streams
      pw.close();
  } // doGet(-,-)
protected void doPost (-,-) throws SE, IOE
  {
    doGet(request, response);
  } // dopost(-,-)
} // class
```

Web.xml :-

cfg TestSrv servlet prg having /turl as the url pattern.

Request url:

http://localhost:2020/TestApp/index.html (tomcat)

07/10/2012

Generally, request Generated by hyperlink does not Carry any data by default.

But by appending queryString to the url of href attribute we can make that work happening.

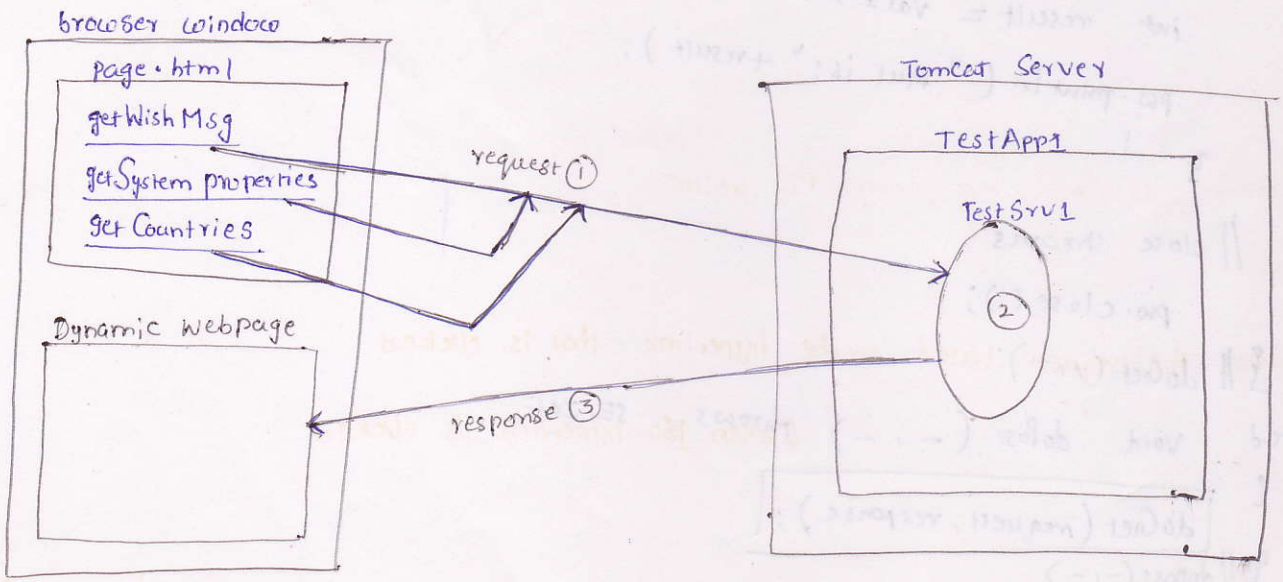
<a href="vturl"> go </a> (vturl is the url pattern of Servlet program)

Sends the request to Servlet program.

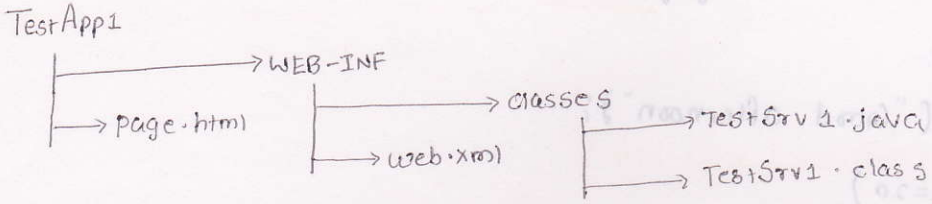
<a href="vturl?sno=101&sname=raja"> go </a>

Sends the request to Servlet program having sno=101, sname=raja as req Param names and values.

Q:- how to handle the situation when when multiple hyperLinks are pointing to Single servlet program?



Deployement Directory structure :-



request uri :-

http://localhost:2020/TestApp1/page.html

Prepare all the 3 hyper links carry request parameter values and use that value as creiteria value To differentiate RequestProcessing logic for each hyper link in Servlet program

Page.html :-

```
<a href="turl1?p1=link1"> getWishMsg </a> <br>
<a href="turl1?p1=link2"> get System Properties </a> <br>
<a href="turl1?p1=link3"> get Contries </a>
```

NOTE:- Form Page can generate either "get" mode (or) "Post" mode request. where as hyperLink generates only Get mode request.

TestSrv1.java :-

```
public class TestSrv1 extends HttpServlet {

    protected void doGet(-,-) throws SE, IOE
    {
    // general Settings
        response.setContentType("Text/html");
        PrintWriter pw = response.getWriter();
    // read additional req param (p1) value
        String val1 = request.getParameter("p1");

    // process the request based on the hynerlink that is clicked
        if (val.equals("link1")) // when 1st hynerlink is clicked
        {
            Calender cl = calender.getInstance();
            int h = cl.get(Calender.HOUR_OF_DAY); // gives current hour of the day
```

ctrl + shift + ;
gives package import statement.

javaee/st-II/javadocs/6docs/api/index.h
→ see the API
java.util pkg
locale class

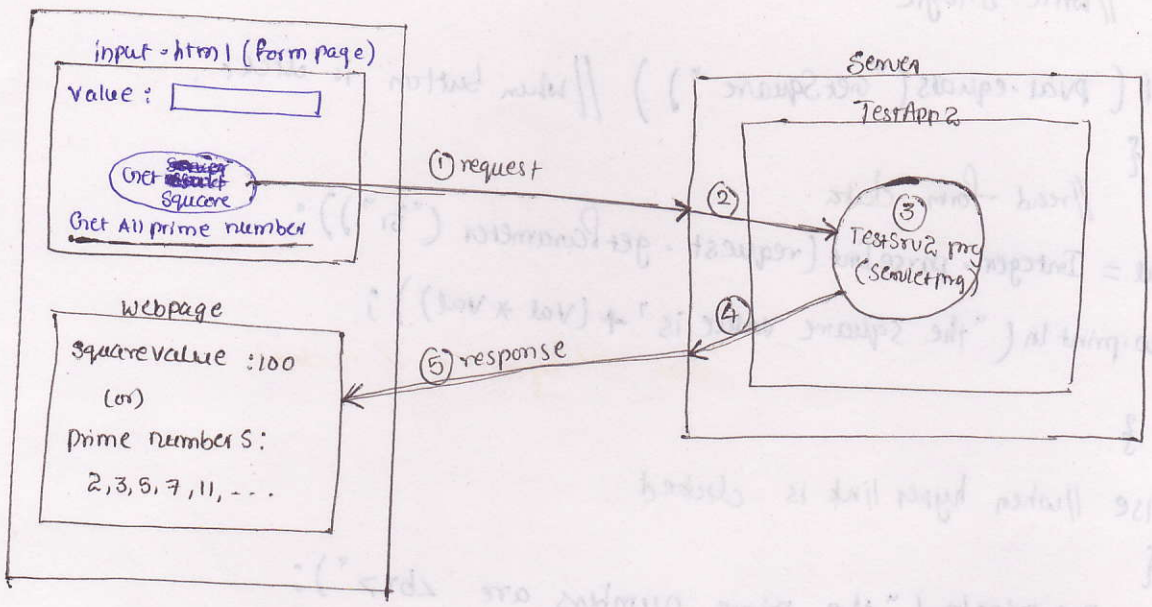
```
if (h<=12)
    pw.println("Good morning");
else if (h<=16)
    pw.println("Good afternoon");
else if (h<=20)
    pw.println("Good evening");
else
    pw.println("Good neight");
}//if
else if(val.equals("link2")) //when 2nd hyper link is clicked
{
    Properties p=System.getProperties();
    pw.println("the system properties are"+p);
}
else
{
pw.println("the available countries and languages are <br>");
    Locale l[] = Locale.getAvailableLocales();
    for(int i=0; i<l.length ; ++i)
    {
pw.println("<br>"+ [i].getDisplayCountry()+"——→"+[i].getDisplayLanguage());
    }
}//else
}// doGet(--)
protected void doPost(--,--) throws SE, IOE
{
    doGet(request, response);
}//doPost(--)
}//class.
```

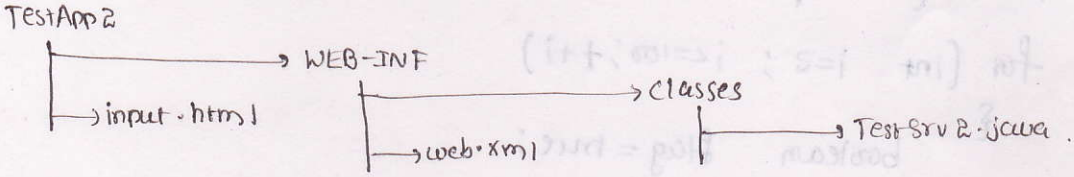
Note:- Locale means Country + Language (lik en-US, fr-CA)

web.xml :-

Configure Test1 prg with "/test1" url pattern.

Q:- How to Handle the situation in Servlet program when submit button and hyperlink of form page Generates request to single Servlet program?





request URL:

http://localhost:2020/TestApp2/input.html

input.html

```
<form  Action = "turl2"  method="get" >
    Value : <input  type = "text"  name="t1" >  <br>
    <input type = "submit"  value = "GetSquare"  name = "s1" >  <br>
</form>
<a  href = "turl2 ? s1=link1" > Get all prime numbers </a>
```

TestSrv2.java

```
public  class  TestSrv2  extends  HttpServlet
{
    protected  void  doGet (-,-)  throws  SE,IOE
    {
        //general settings
        PrintWriter  pw = response.getWriter();
        response.setContentType("text/html");
```

```
//read additional request param value (s1)
String pval = request.getParameter("s1");
    //write b.logic
if( pval.equals("GetSquare") ) //when button is clicke.
{
    //read form data
int val = Integer.parseInt(request.getParameter("t1"));
pw.println("the square value is" + (val * val));
}
else //when hyper link is clicked
{
    pw.println("the prime numbers are <br>");
        for (int i=2; i<=100; ++i)
        {
            boolean flag = true;
        for (int k=2; 1<i ; ++k)
        {
            if(i%k == 0)
            {
                flag = false;
                break;
            }
        }
            if (flag == true)
                pw.println("<br>" + i);
            }
    }
}//doGet(--,--)
protected void doPost (--,--) throws SE, IOE
{
    doGet( request, response);
}
}
```

web.xml:

Configure TestSrv2 program with /test2 url-pattern.

08/10/2012

Q:- What is the difference b/w the http request method get & post

Ans:- GET, POST methods are not java methods. These are different methodologies given by the protocol http to send requests to server from Client ~~Application~~ (browser windows)

GET	POST
→ It is default request method.	→ not default
→ It is designed to get data from Server by generating request.	→ It is designed to send data to Serve along with the request.
→ Can send limited amount of data to the Server (max of 256 KB)	→ can send Unlimited amount of data to the Server.
→ Shows the form pages generated query String in browser address bar	→ doesn't show query String
→ does not provide data Secrusy	→ provide data Secrusy.
→ not Suitable for applying encoding, encription, Security algorithms.	→ Suitable for applying encoading, encription, Security algorithms.
→ does not allow to deal with multipath form data. form Containing different types of conten	→ ~~form~~ allows ~~Containing~~ ~~different types of~~ ~~Content~~ allows
→ not Suitable for file uploading	→ suitable

→ Sends data to the Server quick fasly.	→ It slow compared to get.
→ Use service(-,-)/doGet(-,-) to process the request. In servlet Program	→ Use Service(-,-)/doPost(-,-) to process the request
→ It is Idempotent	→ not Idempotent.

NOTE:- Befor complition of given request processing if Same client is allowed to generate Same request to Server then it is called "double posting problem (or) Idempotent behaviour". This problem raises when Same Submit button is clicked for multiple times (or) if refresh button is clicked after generating request from form page.

→ In doPost(-,-) that processes ~~post~~ Post method based request we can write additional logic to privent double posting (or) Idempotent problems

08/10/2012

Understanding two init methods of servlet API

init () , init(ServletConfig cg)

1) public void **init(ServletConfig cg) throws ServletException**
   Life cycle method
2) public void init() throws ServletException
   Not a life cycle method. It is convience method given for programmers.

- The pre-defined GenericServlet class of servlet API contains both init(ServletConfig cg) method & init() method.
- In pre-defined HttpServlet class there are no init() method definitions.
- When instantiation event is raised servlet container calls init(-) method as life cycle method but it does not call init() method as life cycle method.

Understanding flow of execution related to both init method:

Scenario 1: → It is an abstract class

```
//GenericServlet.java (pre-defined class)
public abstract class GenericServlet implements Servlet
{ (5)
ServletConfig config;
public void init(ServletConfig cg) throws ServletException
{
config=cg; //initialization logic of ServletConfig object
init();
}
public void init() throws ServletExcetption
{
//null body method
}
public servletConfig getServletConfig()
{
return config;
}
...........
........... //other methods of GenericServlet class
...........

}
```

OurServlet class:

Testsrv.~~class~~ .java

```
public class Testsrv extends GenericServlet/HttpServlet ---> (1) (2) (3) (4)
{
public void init() ----------> (6)
{
```

```
.........//our servlet program related initialization logic
.........
}
public void service(-,-)throws ServletException,IOException
{ (7)
.......
.....
}
}
```

With respect to above scenario

1) End user gives first request to our servlet program (Testsrv.java) (let us assume no <load-on-startup> is enabled on this servlet program.
2) Servlet container creates our servlet class object using 0-argument constructor.
3) Servlet container creates ServletConfig object as right hand object as our servlet class object and also raises instantiation event.
4) To process instantiation event servlet container calls init(-) life cycle method on our servlet class object having ServletConfig object as argument value of that init() method. Since that method is not available in our servlet class the super class init(-) method will be executed.
5) this super class init(-) method logic is there to assign container supplied config object with the ServletConfig interface reference variable and also calls init() method. → ServletConfig object to initialization
6) Since init() method is available in our servlet class that method executes and the initialization process of our servlet program will be completed.
7) Servlet container raises request arrival event and it calls public service(-,-) method on our servlet class object. This method execution will process the request and sends the response to browser window.

NOTE:- Self class methods & super class public methods can be called without objects.

In scenario 1 our servlet program can get access to ServletConfig object by calling getServletConfig() method of predefined GenericServlet class in the life cycle methods or other methods of our servlet program.

ServletConfig cg= getServletConfig();

public methods of super class can be called in sub class methods without objects.

NOTE:- In Scenario 02 we can't call getServletConfig() in our Servlet programs. To access ServletConfig object, bcz the control doesnot go to init(-) method of predefined GenericServlet class where ServletConfig object initialization logic is available. Code is same as Code

Scenario 2:

GenericServlet.java (pre-defined class)

public abstract class GenericServlet implements Servlet

NOTE:- The method i.e called by ServletContainer directly when lifecycle event is raised is called as life-cycle method. (like init(-) symbol). If method is called by container indirectly for lifecycle event then it is not called life-cycle method (like init() method).

```
{
ServletConfig config;
public void init(ServletConfig cg)
{
config=cg;
init();
}
public void init()
{
//null body method
}
public ServletConfig getServletConfig()
{
return  config;
}
……//other methods of GenericServlet class
…….
}
```

Ourservlet.class

Testsrv.java

……..

……..

(1) (2) (3) (4)

```
public class Testsrv extends GenericServlet/HttpServlet
{
ServletConfig cg;
public void init(ServletConfig cg)
{
this.cg = cg; //(5) explicit initialization logic of ServletConfig object
…… //our servlet program related initiliaztin ~~lois~~ logic
}
public void service(ServletRequest req, ServletResponse res) throws ServletException,
IOException
{
……(6)
……
}
}
```

In scenario 2 OurServlet program contains init(-) method directly, so the control will not go to init(-) method of the GenericServlet class. So programmer must initialize ServletConfig object in ~~its~~ is init(-) method of servlet program to make it visible to other methods of these servlet program that means in scenario 2 programmer must not forget the explicit initialization of servlet-config

object in the init(-) method of his servlet program. We cannot call a method on a object which holds null value.

Scenario 3:

GenericServlet.java (pre-defined class)

```
public abstract class GenericServlet implements Servlet
{
ServletConfig config;
public void init(ServletConfig cg) throws ServletException
{
(6)
config cg; //initialization logic of ServletConfig object
init();
}
public void init() throws ServletException
{
(7) //null method
}
public ServletConfig getServletConfig()
{
return config;
}
...........
........... //other methods of GenericServlet class
...........
}//class
```

Ourservlet class:

Testsrv.class

```
.........
.........
    (1) (2) (3) (4)
```

servlet Container calls with init(-) having servlet config obj

```
public class TestSrv extends GenericServlet/HttpServlet
{
public void init(ServletConfig cg)
{ (5)
super.init(cg);
............... (8)
............//our servlet program realted initialization logic
}
public void service(ServletRequest, ServletResponse res) throws ServletException,IOException
{
(9)
ServletConfig cg;
```

.....
.....
}
}

In scenario 3 programmer can use getServletConfig() method call to get access to ServletConfig object to its servlet program.

- In scenario 1 use getServletConfig() method to get access to ServletConfig object.
- In scenario 2 initialize ServletConfig object explicitly in init(-) method to use that object.
- In scenario 3 call super.init(-) method in our init(-) method and also call getServletConfig() method to get access to ServletConfig object.
- Always give chance to init(-)method of pre-defined GenericServlet class to execute once in the instantiation and initialization process of our servlet program because it initializes ServletConfig object and makes programmer free from that process. Due to this scenario 1 is most recommended approach to place init() methods in our servlet program.

Understanding two service (-,-) methods and 7 doXxx() methods of pre-defined HttpServlet class:

- When we develop our servlet program by extending it from pre-defined HttpServlet class we can place request processing logic in our servlet program either by using one of the two service methods or by using doXxx(-,-) methods
  1) public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException
     public service(-,-) or service method
  2) protected void service(HttpServletRequest req, HttpServletResponse res)throws ServletException,IOException
     Protected service(-,-) or service method
  3) public void doXxx(HttpServletRequest req, HttpServletResponse res) throws ServletException,IOException
     doXxx- total 7 number of doxx(-,-) methods are available like doGet(-,-), doPost(-,-),doDelete(-,-),doPut(-,-), doHead(-,-),doTrace(-,-),doOptions(-,-)
- Even though 7 doXxx(-,-) methods are there the regularly used doxx(-,-) methods are doGet(-,-) & doPost(-,-).
- Only public service(-,-) is the life cycle method
- Servlet container calls this public service(-,-) method as life cycle method when the request arrival event is raised.
- Protected service(-,-) and doXxx(-,-) are not life cycle methods. they are convinience methods to a program. to use HttpServletRequest, HttpServletResponse objects directly without any typecasting
- by overriding multiple doXxx(-,-) methods in our Servlet program we can execute different logics for different request method based request generated by client (browser, window

Scenario 1 to understand flow of execution related to service (-,-) , doXxx(-,-) methods:

//HttpServlet.java(pre-defined class)

```
public abstract class HttpServlet extends GenericServlet
{
  public void service(ServletRequest req, ServletResponse res) throws SE, IOE
  {(4)
    //typecasting
    HttpServletRequest request = (HttpServletRequest) req;
    HttpServletResponse response = (HttpServletResponse)res;
    // calls protected service(-,-) method
    service(request, response);
  }
protected void service(HttpServletRequest request, HttpServletResponse response)
  {(5)
    //read request method of browser generated request
    String method = request.getMethod();
    //call appropriate doXxx(-,-) based on request method
    if (method.equals("GET"))
        doGET(request, response);
    if (method.equals("POST"))
        doPost(request, response);
  }
```

```
protected void doXxx(HSR req, HSR res) throws
{ sends 405 error response back to browser window
}
.....//other methods
}//class
```

Ourservlet program:

//Testsrv.java

```
public class Testsrv extends HtpServlet
{
 (1)(GET) (2) (3)
public void doGet(HttpServletRequerst req, HttpServletResponse res)throws
ServletException, IOException
{
..........
..........//some logic (6)
}
}
```

With respect to above scenario code

(1) End user gives request to our servlet program having request method "GET" from client(browser window).

(2) Servlet container creates or locates our servlet class object if created the servlet container completes the instantiation and initialization related life cycle operations.

(3) Servlet container raises request arrival event and calls public service(-,-) method on our servlet class object as life cycle method having request, response objects as arguments. Since public service(-,-) method is not available in our servlet program, the super class (pre-defined HttpServlet class) public service(-,-) method executes.

(4) ~~When~~ the public service(-,-) method of pre-defined HttpServlet class converts simple servlet Request object to HttpServlet Request object & servlet Response object to HttpServletResponse object & calls protected service(-,-) method having these two objects. Since protected service(-,-) method is not available in our servlet class the super class (pre-defined HttpServlet class) protected service(-,-) method will be executed.

(5) The protected service(-,-) method of pre-defined HttpServlet class reads ~~equest~~ request method of client generated HttpRequest and calls an appropriate doXxx(-,-) method i.e., doGet (-,-)method. Since doGet (-,-) method is available in our servlet program that method will be executed.

(6) The doGet(-,-) method of TestSrv program will process the request & sends generated response to browser window as web page.

To understand more scenarios regarding these service(-,-) &doXxx(-,-) method related flow of execution refer the 6 cases given in page 56.

The execution of our servlet program don't let the control going to doXxx(-,-) methods of pre-defined HttpServlet class because they always generate 405 error response page indicating our servlet is totally incomplete to process the request.

```
public class TestSrv extends HttpServlet
{
public void service(ServletRequest req, ServletResponse res)throws
ServletException,IOException
{
super.service(req,res);
}
Protected void service(HttpServletRequest req,HttpServletResponse res) throws
ServletException,IOException
{
.........
.........
}
}
```

Here protected method is executed because of super.service(req,res) control ~~goes~~ passes the to super class and public service(-,-) method super class internally calls the protected service(-,-) method so ~~this~~ that protected service(-,-) method is executed.

→ There are 2 init() in our Servlet-api. but only init(ServletConfig cg) is the lifecycle method init() is not the lifecycle method.

→ There are multiple approaches / scenarios to keep init( ) methods in our Servlet program.

→ To Understand this scenarios page No: 1 to 5 of the Supplementary handouts given on 08/10/2012.

09/10/2012

→ A good programmer never Override init(-) in its servlet prg.

Because if is overrides this method either he has to initialize Servlet Config implicitly (or) he has to call super.init explicitly as shown in Scenario 2 and Senario 3. To over come this problem the override init() as shown in approach (1)

Q:- When init(ServletConfig cg) method is lifecycle method? why another init() method is given?

Ans:- init() method is given as convinience method for programer to place his own initialization logic and to avoid the explicit initialization of Servlet Config object. for elobarated anser refer page no: (55)

*) Understanding two Service() and Seven doXxx( ) methods of Servlet-api

to understand the flow of execution related to two service() methods and Seven doXxx() methods refer given supimentary handout page nos: 5 to 7 (08/10/2012)

→ Refer the 6 scenario of page no: 56 to understand the flow of executions related to two Service(-,-) methods and Seven doXxx(-,-) method

Q:- when all the methods of pre-defined HttpServlet class (or) concreate methods why the class it self is taken as abstract class?

Ans:- Refer the last 4 paragraphs of page No: 56 of the booklet.

→ A Client can give 7 methods based request to Our Servlet program. They are

① GET ② POST ③ HEAD ④ PUT ⑤ DELETE ⑥ TRACE ⑦ OPTIONS

→ To process these method based request in our servlet program the Servlet-api as given 7 doXxx(-,-) methods.

doGet(-,-) doPost(-,-) doPut(-,-) doDelete(-,-) doHead(-,-) doTrace(-,-) doOptions(-,-)

→ In real time projects GET, POST request methods based requests will be generated and doGet(-,-), doPost(-,-) will be placed in Servlet prgms to process these requests.

*) GET:- Designed to get data from Server by sending request. This request can carry max of 256 kB data. This request related response contains the response body.

*) HEAD:- Same as GET. but this request related response does not contain response body. This method is useful to check the availability of web resource program.

*) POST:- Designed which send request to Server having the capability to carry unlimitted amount of data along with the request.

*) PUT :- FTP application Sends put( ) method based request to Server to place new web resource programs in web application.

*) DELETE :- FTP application DELETE method based request to Delete existing web resource program from Server.

*) TRACE :- Client uses this method to know the flow of ~~request~~ execution from request generation to response arrival. Useful for debbugging

*) Options :- This method based request returns the possible HttpRequest method which can be used to generate request to web resource program from clients

Eg :- If client gives options method requests to A Servlet Program that overrides doGet(-,-) method.

~~having opti~~ then the response looks like this

Allow GET, HEAD, TRACE, OPTIONS

11/10/2012

→ A good programmer places init( ) method, doGet(-,-), doPost(-,-) methods in his Servlet program to place the initialization logics.

Q :- When public service(-,-) is lifecycle method? why we are prefering to place doXxx(-,-) methods in our Servlet program to maintain the RequestProcessing logic?

Ans :- public service(-,-) does not give HttpServletRequest, HttpServletResponse objects to work with all the facilities of Http protocol. But doXxx(-,-) method provides these objects.

While working with Service(-,-) we need to write RequestProcessing logic of

different request methods based requests in a single method definition.
↳ like get, Post, ...

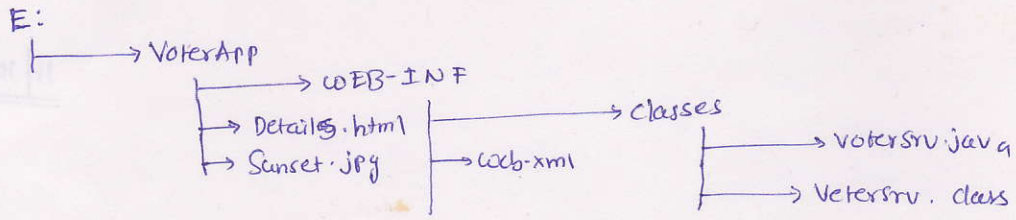
where as we as use different doXxx(-,-) methods to place the request processing logics of different request methods like GET method logic in doGet(-,-), POST method logic in doPost(-,-) and etc.....

→ we can deployee our java webapplications either in the form of directory or war file

→ War file ⟶ Webapplication archeive
(user Jar command to create war file)

Procedure to prepare war file representing the Java webapplication :-

Step-I :- Create the deployeement directory structure of web application



Step-II :- prepare the war file.

E:\VoterApp > jar cf VoterApp.war
↳ represents current directory.

c ⟶	create war file
f ⟶	specify the war file name

There are 3 approches to deployee Java web applications in Servers.

① Hard deployeement :- Copy war file or directory to a fixed folder of server s/w installation (like <tomcat_home>)

② Console deployeement :-
Deployee war file through admin console of server)

③ Tools based deployeement :- Deployeement using IDEs, maven tool, ant tool and etc.

Programmers generally prefer working with approach ③.

Procedure to perform console deployement in Tomcat Server :-

Step I :- prepare war file on your application. (like VoterApp.war file).

Step II :- open the tomcat manager window (http://localhost:2020/) (admin console)

http://localhost:2020
<tomcat-home> → Tomcat manage → submit Username : admin, Password : admin → go to WAR file to deplo
secti

→ Browse and select the WAR file (VoterApp.WAR) → Deployer.

Step III :- Test the web application.

Request URL :- http://localhost:2020/VoteApp/Details.html
↳ WAR file name as Context path.

TO perform Hand deployment of web application copy its directory (like VoterApp) (or) its WAR file (like VoterApp.war) to <tomcat-home>\webapps folder. (WAR file name / directory name acts as Context Path)

Weblogic :-

Type : Application Server s/w

Version : 10.3 (Compatible with jdk 1.6)

Commercial s/w

default port no : 7001

Vendor : BEA Systems (Oracle corporation)

to download s/w : www.commerce.bea.com
www.oracle.com

allows to create domains. Each domain acts as one application Server

jar file that represents all jee api's : weblogic.jar (<weblogic_home>\wlserver_10.3\server\lib) (oracle\middleware)

NOTE :- In Weblogic 10.x there are no default domains.

→ If multiple projects of a Company are using weblogic then weblogic will be installed only once in a common machine and multiple domains will be created in that weblogic for multiple projects on one per project basis

Procedure to create Usedefined domain in weblogic :-

Start → programs → Oracle weblogic → Quick Start → ⊙ Getting Started with weblogic server

→ ⊙ Create a new web logic domain → next → ⊙ Generate a domain configured.....

→ next

Domain Name : [AdvBatchDomain] → next

Username : [Javaboss]

User password : [Javaboss1]

Conform passcord : [Javaboss1] → next → next → ☑ Administration server

→ next → Listenport : [7070] → next → Create → done

Procedure to perform console deployement of webapplication in AdvBatchDomain server of weblogic :-

Step-I :- Prepare the war file (like VoterApp.war)

Step-II :- Start AdvBatchDomain server

Start → programs → Oracle weblogic → User projects → AdvBatchDomain →

Start admin server for weblogic.

Step-III :- Open the admin console of this domain Server.

Open Browser window → Type this URL (http://localhost:7070/Console)

Username : [Javaboss]

Passcord : [Javaboss1] → Login (gives admin console)

Step-IV :- Deployee the web application

Admin Console (Right) → Deployeements → install (Left) → Upload your files →

Deployement archive [ ] [V] → Browse and select VoterApp.war file → next

next → next → next → finish

Step-V :- Test the web application

Open Browser window → Type this URL (http://localhost:7070/VoterApp/Details.html)

Procedure to undeployee web application from admin console in weblogic :-

Admin Console ⟶ Deployements ⟶ [✓] VoterApp ⟶ Delete.

To perform hard deployement of web application in Adv Batch Domain Server of weblogic

Copy the directory | war file of the web application to <oracle weblogic-home>/middleware/user_projects/domains/AdvBatchDomain/autodeploy folder

directory ↓ VoterApp

war file ↓ VoterApp.war

NOTE :- ① In weblogic Servers the deployeed directory name | war filename acts as Context path of the webapplication.

② ⟶ In tomcat server when the source code of Servlet program is modified we need to recompile source file and we need to reload the webapplication. In case of weblogic Server just the recompilation is enough.

12/10/2012

Q:- What is the diff b/w Hot deployeement and cold deployeement?

Ans:- *) If web application is deployeed in the servers then the server is in Running mode. that is called Hot deployment.

*) If webapplication is deployeed in the server when the server is in stopped mode. then it is called Cold deployement.

*) Console deployement is always Hot deployeement. Hot deployeement can be either Hard deployement (or) Cold deployeement.

In weblogic server started any command prompt is opened the webapplications not deployed in the autodeployee folder. so, first close the Command prompts. then deployee the web applications in autodeployee folder. then deployeed is successfull.

GlassFish

type : Application Server s/w

Vendor : SunMS (Oracle corp)

default port no : 8080 (for Http requests)
4848 (for admin console)

Version : 2.x (Compatible with jdk 1.5+)

OpenSource

Built-in and Default server of NetBeans IDE.

NOTE:- The GlassFish Server that comes with NetBeans IDE can be used with or without IDE.

allows to create Domains. The default domain name is domain1.

jar files that represents all JEE APIS : javaee.jar (<GlassFish_home>\Appserver\lib folder)

default domain (domain1) credential details are

1) Username : admin
2) password : adminadmin

(*) procedure to create User defined domain server in GlassFish 2.x

D:\sun\AppServer\bin > asadmin create-domain --adminport 4343 --user testuser1 mydomain1

admin console port number

domain username

user domain name

please enter admin password > testuser1

please enter the admin password again > testuser1

please enter the master password ↵

⋮

Domain mydomain1 created

To change the Http port number of the above mydomain1 server

goto <GlassFish_home>\Appserver\domains\mydomain1\Config\domain.xml and modify "port" attribute value of first <http-listener> tag (at 50th line port no = 6565).

(*) procedure to deployee the Java webapplication in mydomain1 server of Glassfish. (console deployement)

Step-I :- Prepare war file representing web application (like VoterApp.war)

Step-II :- Start mydomain1 server of Glassfish

D:\sun\Appserver\bin> asadmin start-domain mydomain1
↳ domain name

Step-III :- Open the admin Console of Glass-fish

Open Browser window → Type this URL (http://localhost:4343) ~~console~~

UserName [testuser1] → Click on Login
Password [testuser1]

Step-IV :- Deployee the web application

adminConsole → ~~Enterprise~~ Applications → webapplications → Deployee →

Type: [WebApplication] → Browse and select war file (VoterApp.war)

→ OK

Step-V :- Test the webapplication

open Browser window → Type this URL (http://localhost:6565/VoterApp/Details.html)

If u undeployee the webapplication

select the webapplication ☑ VoterApp → click Undeployee option.

Hot deployment of application in GlassFish

In GlassFish Server there is no support for directly based hot deployement. but the war file based hot deployement is possible. for this Copy the war file to

<Glassfish_home>\Appserver\domains\mydomain1\autodeployee folder
↳ our domain name.

open browser window → Type URL: http://localhost:6565/VoterApp/Details.html

To stop the Glass Fish domain server

D:\sun\AppServer\bin> asadmin stop-domain mydomain1

Domain mydomain1 stopped.

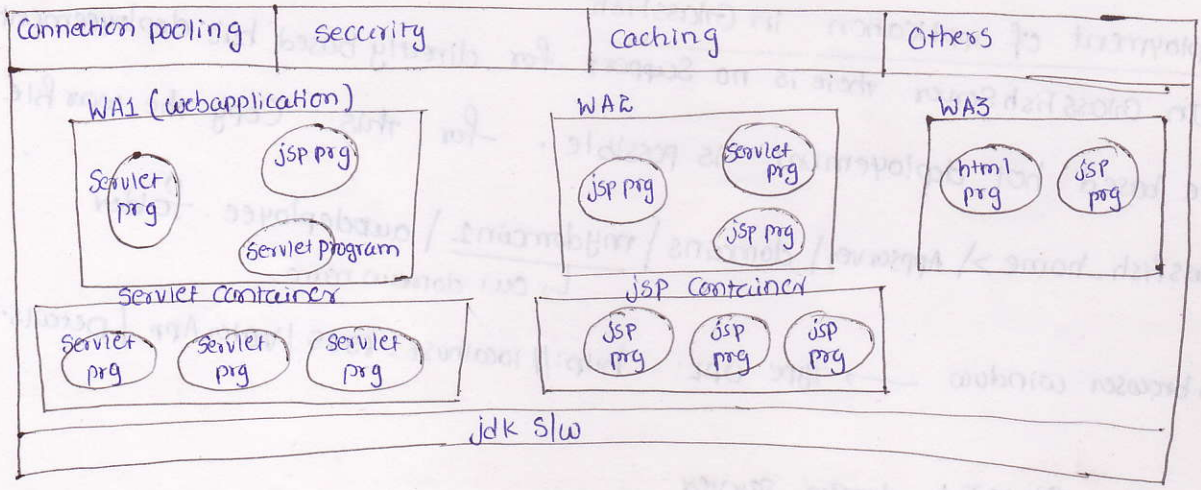
Q: what is the diff b/w webserver and Application server ?

Webserver	Application Server
→ given to manage and execute webapplication (WAR)	→ given to manage and execute webapplication (war), EJB, Enterprise application (ear) Resource Adaptor Application (rar) and etc.
→ use only Servlet, JSP container	→ use Servlet container, JSP Container and EJB container.
→ Developed based on Servlet, JSP specifications	→ Developed based on Servlet, JSP, EJB, JMS, etc... specifications.
→ Doesn't alloco to create domains	→ allocos ~~gives less middleware services~~.
→ gives less middleware services	→ more
→ allocos only HTTP request	→ allocos both http and non-http request
→ Suitable for small and medium scale application	→ Suitable for large scale webapplications other JEE applications
Eg: Tomcat, resin, Jetty ...etc..	→ Eg: weblogic, websphere, JBoss, GlassFish and etc.

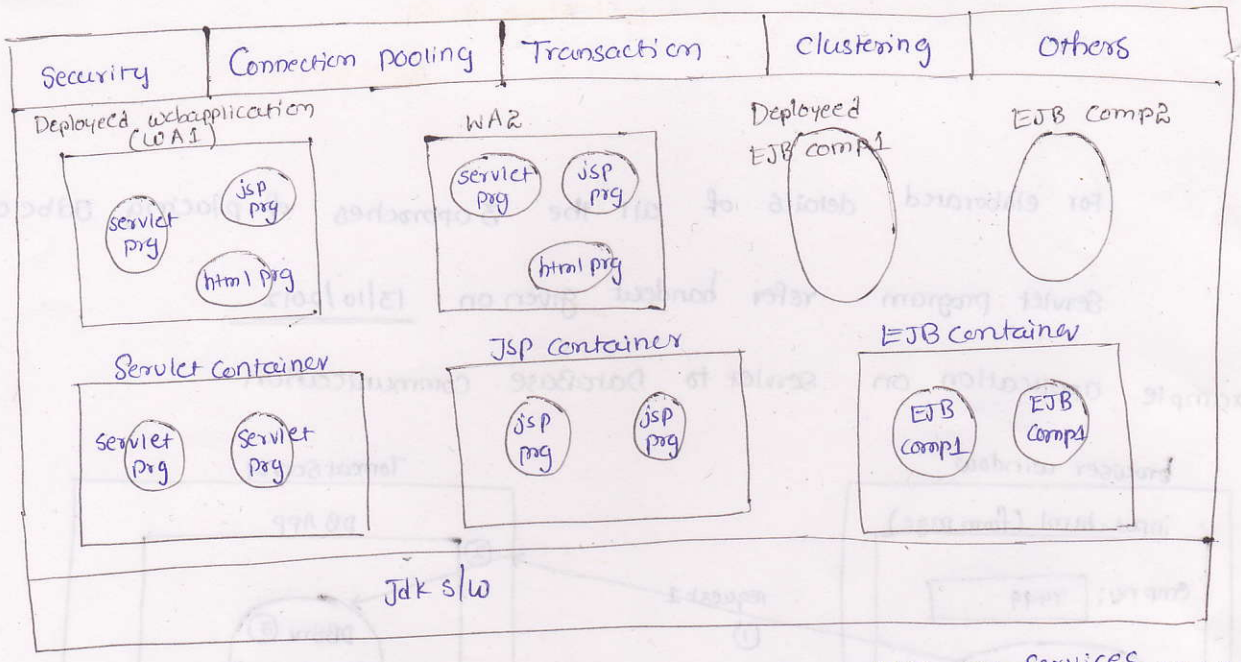
NOTE:- The large ~~scale~~ and complex application is called Enterprise app (ear file)

ear file = war file + jar file + ...

Eg: Banking Apps



the java/jee app that talks with ERP (like SAP) / CRM (like siebel) s/w s is Resource Adaptor App (rar)



Application Server = webserver + EJB container + more middleware services.....

13/10/2012

Servlet to DataBase s/w Communication :-

→ To save received form data in Database table, to save the generated results in Database table, to gather inputs from DataBase table we need to make webResource programs like Servlet programs to interact with DataBase s/w.

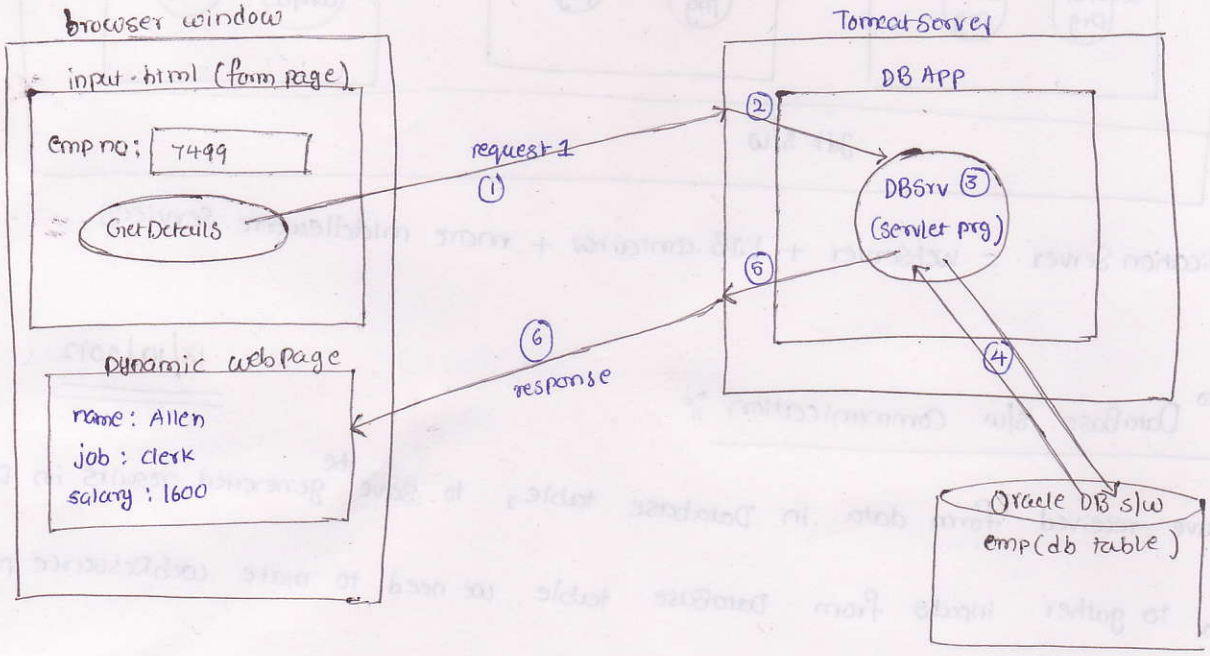
→ If multiple Threads are started on Single object | Variable Simulteneously (or) Concurrently then that object or Variable is not Threadsafe. To make it Threadsafe using Synchronization Consept.

→ Our Servlet program is Single Instance and multiple Thread Components. So, Instance Variables of servlet program are not Threadsafe by default. But local Variables of Service of doXxx(-,-) methods are Thread Safe by default. To make the Instance Variables of Servlet program as Thread Safe Variables use them in Service(-,-) by using the Synchronized blocks.

→ For Servlet to Data Base s/w communication place Jdbc code in Servlet programing. There are 3 approches for this.

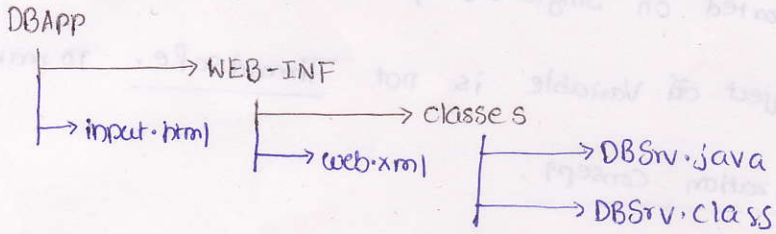
For elaborated details of all the 3 approches of placing Jdbc code in Servlet program refer handout given on 13/10/2012

*) Example application on servlet to DataBase communication

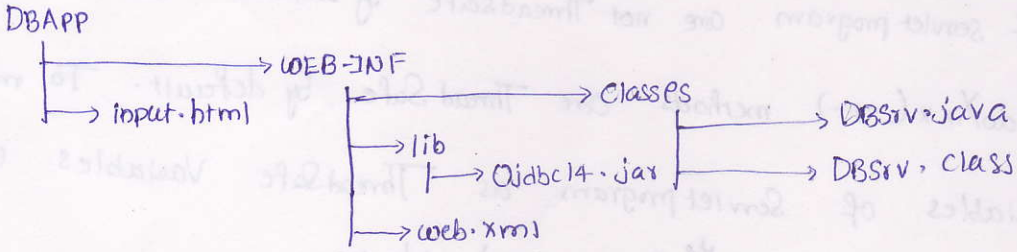


Deployement Directory Structure (when servlet program uses Type1 Jdbc Driver)

Type1 jdbc Driver is a built-in jdbc driver of jdk/jre.



Deployement Directory Structure (when servlet program uses Oracle thin Driver)



jarfiles in class path

Servlet-api, Ojdbc14.jar (optional)

13/10/2012

Servlet to DataBase s/w communication :-

Approaches to place JDBC code in the servlet program:

There are three approaches to place JDBC code in the servlet program are:

Approach 1:

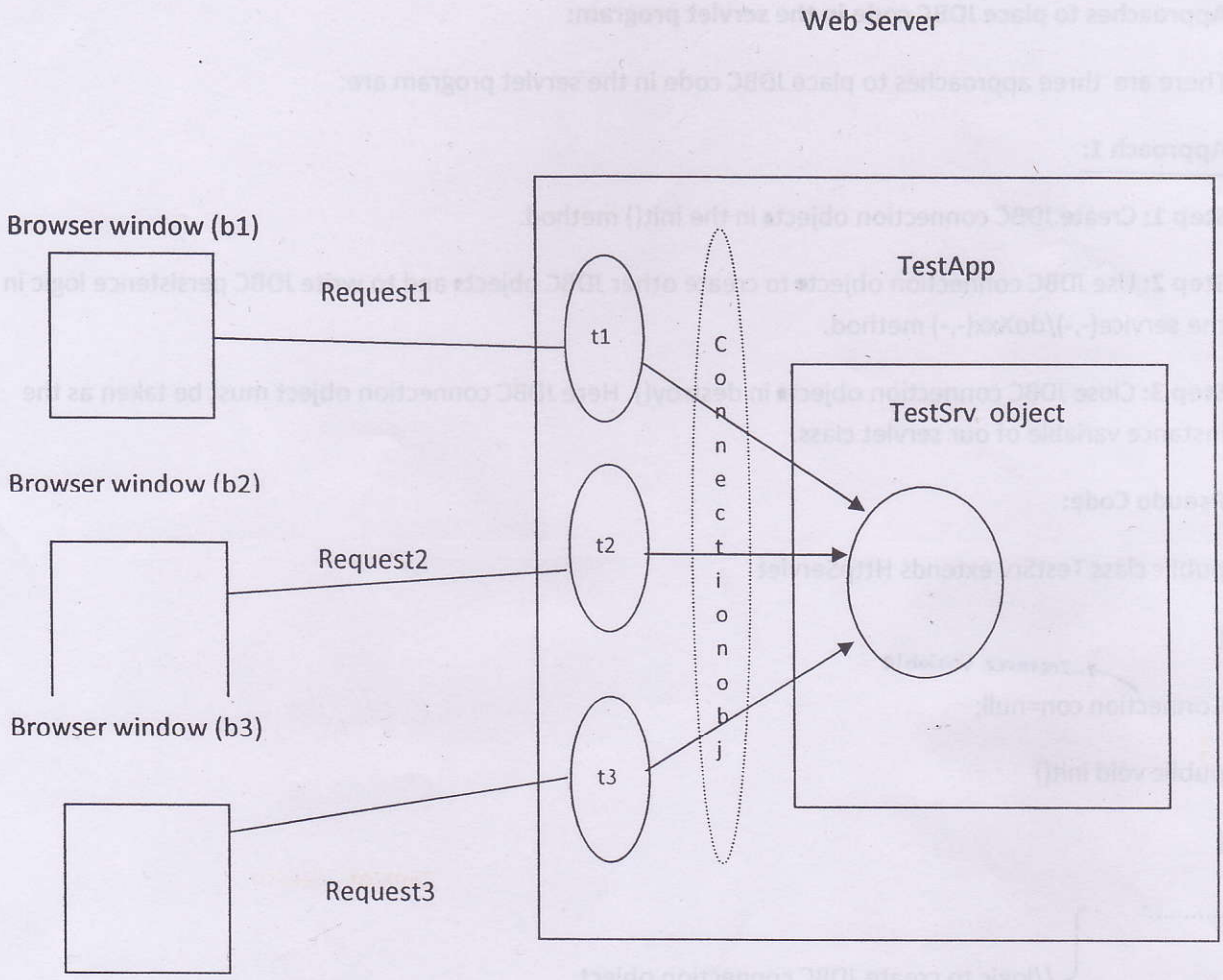
Step 1: Create JDBC connection objects in the init() method.

Step 2: Use JDBC connection objects to create other JDBC objects and to write JDBC persistence logic in the service(-,-)/doXxx(-,-) method.

Step 3: Close JDBC connection objects in destroy(). Here JDBC connection object must be taken as the instance variable of our servlet class.

Pseudo Code:

```
public class TestSrv extends HttpServlet

{

Connection con=null;   → Instance Variable

public void init()

{

..........
..........   } //logic to create JDBC connection object
..........

}

public void service(-,-)/doXxx(-,-) throws ServletException,IOException
{
..............
..............   } // logic to use JDBC connection object to create other JDBC object and to
                     write JDBC persistence logic
}
public void destroy()
{
..........
..........   } //logic to close JDBC connection object
}
}//class
```



All three threads are using single copy of JDBC connection object.
t_1, t_2, t_3

Advantages:

All the requests coming to servlet program will use single JDBC connection object to interact with database software. This improves the performance.

Disadvantages:

JDBC connection object is instance variable of servlet program. So it is not thread safe. To make it thread safe use synchronization concepts explicitly.

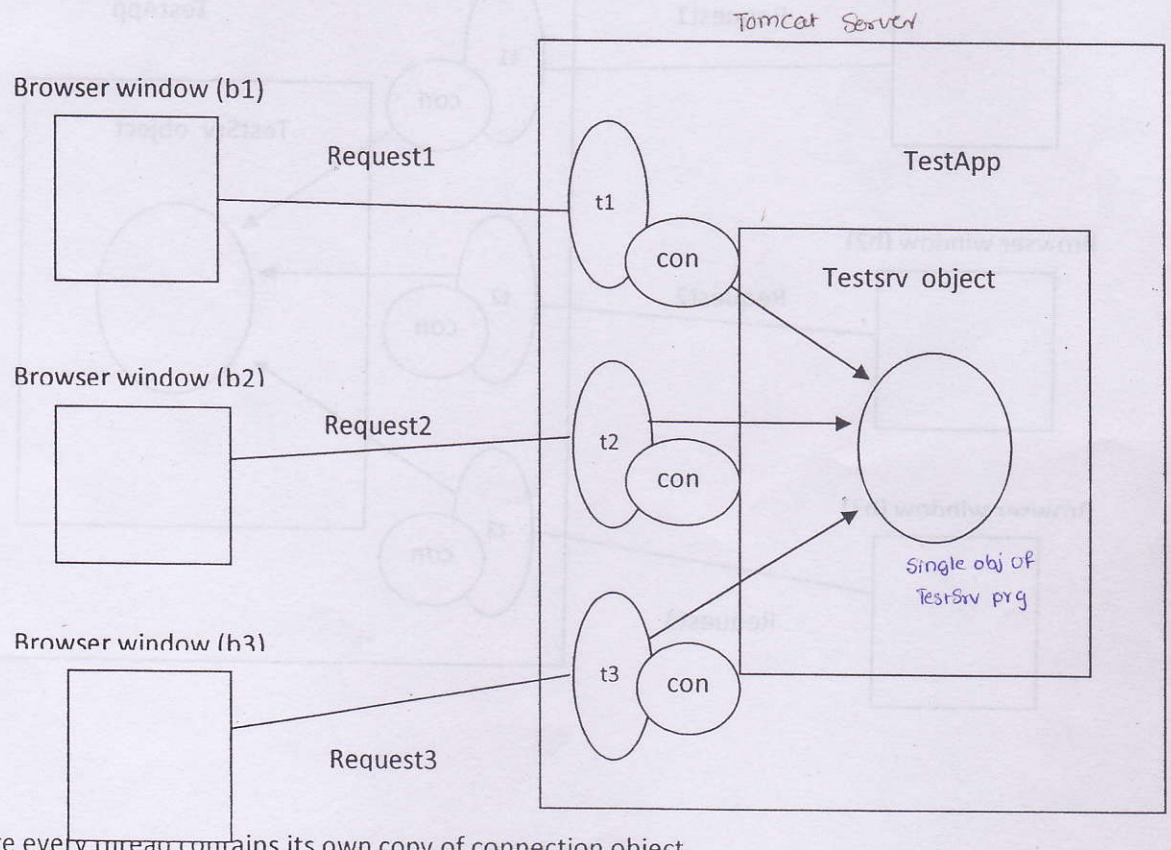
Approach 2:

① Create JDBC connection object in service(-,-)/doXxx(-,-) methods. ② Use JDBC connection object to create other JDBC objects and to write JDBC persistence logic in service(-,-)/doXxx(-,-) method. ③ Close

JDBC connection object service(-,-)/doXxx(-,-) method. Here JDBC connection object should be taken as local variable of service(-,-)/doXxx(-,-).

Pseudo Code:

```
public class TestSrv extends HttpServlet
{
        public void service(-,-)/doXxx(-,-) throws ServletException,IOException
        {
        //create JDBC con object
        ...........
        ...........
        //write JDBC persistence logic
        .............
        .............
        //close JDBC con object
        .............
        .............
        }
}
```



Here every thread contains its own copy of connection object.

Advantages:

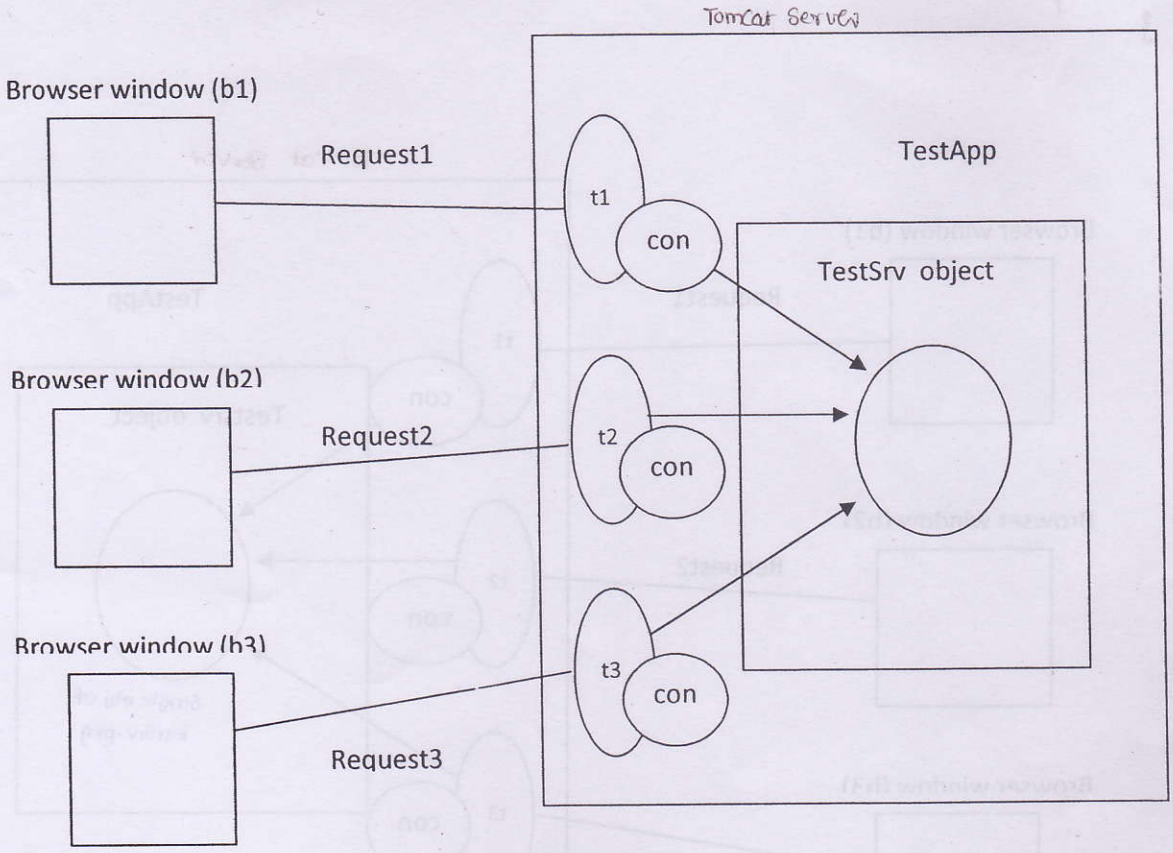
Here JDBC connection object is local variable of service(-,-)/doXxx(-,-) method. So it is thread safe by default.

Disadvantage:

Every request given to servlet program establishes a new connection between servlet program and database software. This kills the performance.If 1000 requests are given to server/servlet program 1000 connection objects will be created, due to this performance will be degraded

Approach 3:

① Being from service(-,-)/doXxx(-,-) method get JDBC connection object from JDBC connection pool managed by application server/web server. ② Use the JDBC connection object in service(-,-)/doXxx(-,-) method to create other JDBC objects to write JDBC persistence logic. ③ Release JDBC connection object back to JDBC connection pool being from service(-,-)/doXxx(-,-) methods. Here JDBC connection object is local variable in service(-,-)/doXxx(-,-) method.



Advantages:

- Servlet program is not responsible to create, manage and destroy JDBC objects. All these operations taken care by JDBC connection pool.
- Connection object is local variable to service(-,-)/doXxx(-,-). So it is thread safe by default.
- Multiple requests coming to servlet program communicate with database software by using minimum number of JDBC connection objects.
- Software Industry prefers using approach3 to add JDBC code in servlet program in large scale web application like banking projects.
- Similarly industry prefers using approach 1 in small scale projects (web applications).

Jar files in WEB-INF/lib folder:

- ojdbc14.jar

  When standalone application uses third- party API then the third-party API like ojdbc14.jarrelated jar files must be added to classpath. These jar files will be used by java compiler and runtime environment during the compilation and execution of the standalone application.

  Jar files added to classpath are always visible to command prompts but not visible to web servers/ application servers and its containers.

  Standalone applications execution takes place from command prompt. Servlet program compilation takes place from command prompt and execution takes place the servlet container of server.

Thread safety:

If multiple threads are running on single object/variable simultaneously or concurrently then that object/variable is not thread safe. To make that object/variable as thread safe allow only one thread at a time on the object/variable to manipulate the data through synchronization concepts. By default our servlet class object and its instance variable are not thread safe because our servlet program is single instance and multiple threads component. To make our servlet program and its instance variables as thread safe work with either synchronized service/doXxx (-,-) methods or synchronized blocks in service/doXxx(-,-) methods.

→ The stand alone java applications

Compilation and execution takes place from Command prompt. where as the Servlet program Compilation takes place from Command prompt and the execution takes place from Servlet Container.

In stand alone java application uses Third party API (other than Jdk APIs) then Third party related Jar files must be added to classpath.

If Java web application uses Third party API in its webresource programs (Servlet programs) Then the third-party API related Jar files must be added to classpath and must also be added to the WEB-INF/lib folder of webapplications deployement directory structure

Eg:- If the Servlet programs of Java web application uses Oracle Thin Driver then its ojdbc14.jar file should be added to CLASSPATH and also be added to WEB-INF/lib folder as shown above.

In the above scenerio the jar-files added to classpath will be used by Javac Tool to recognize the 3rd party API during the Compilation of Servlet programs. Similerly the jar files added to WEB-INF/lib folder will be used by Servlet Container to recognize and use 3rd party API during the execution of Servlet program.

Source code of the above application :-

```
// input.html
<form action="dbcal" method="post">
    <b> Enter Emp Number </b>
    <input type="text" name="tenO"> <br>
<input type="submit" value="getEmpDetails">
</form>
```

```
//DBSrv.java (shows Servlet to DB s/w communication using 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 con obj using type1 driver
        Class.forName("oracle.jdbc.driver.OracleDriver");
        con=DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl"
                                        "scott","tiger");
            //create preparedStatement object
~~PreparedStatement~~ ps=con.prepareStatement("select * from emp where empno=?");
        } //try
        catch(Exception e)
        { e.printStackTrace(); }
    }//init()

    public void doGet(HttpServletRequest req, HttpServletResponse res) throws SE,IOE
    {
        // general Settings
        PrintWriter pw=res.getWriter();
        res.setContentType("text/html");
        try
        {
            //read form data
            int no=Integer.parseInt(req.getParameter("tno"));
            //Write jdbc code for Database interaction
                //set value to query parameter
```

```
ps.setInt(1,no);
//execute the Query
ResultSet rs = ps.executeQuery();
//process the ResultSet
if (rs.next())
{
pw.println("Employee no"+rs.getInt(1));
pw.println("Employee name"+rs.getString(2));
pw.println("Employee Job"+rs.getString(3));
pw.println("Employee manager"+rs.getString(4));
pw.println("Employee HireDate"+rs.getDate(5));
pw.println("Employee Salary"+rs.getLong(6));
pw.println("Employee Comm"+rs.getInt(7));
pw.println("Employee Deptno"+rs.getInt(8));
}//if

//close ResultSet object
rs.close();
}//try  pw.Println("<br><br><a href="input.html">home</a>");
Catch(SQLException se)
{ se.printStackTrace(); }
}//doGet(..)
```

HyperLink, come to back

```
public void doPost(HttpServletRequest req, HttpServletResponse res) throw SE,IOE
{
doGet(req,res);
}

public Void destroy()
{
try
{
if (ps!=null)
ps.close();
}
Catch(Exception e)
{ e.printStackTrace(); }
```

to avoid Nullpointer exception.

```
try
{
```

```
        if(con!=null)
            con.close();
        }
        catch(Exception e)
        { e.printStackTrace(); }

    }//destroy();
}//class
```

Tomcat6.0\webapps\DBApp\WEB-INF\classes>

/> javac DBSrv.java

NOTE:- Make sure that Ojdbc14.jar file is in place in WEB-INF/lib folder

//web.xml

```
<web-app>
    <servlet>
        <servlet-name> abc </s-n>
        <servlet-class> DBSrv </s-c>
        <load-on-startup> 1 </load-on-startup>
    </servlet>
    <servlet-mapping>
        <servlet-name> abc </s-n>
        <url-pattern> dburl </u-p>
    </servlet-mapping>
    <welcome-file-list>
        <welcome-file> input.html </wecome-file>
    </welcome-file-list>
</web-app>
```

optional but recomended

Start Tomcat Server :-

Request url :- http://localhost:2020/DBApp/

15/10/2012

→ if multiple webapplications deployeed in a server are using same 3rd party API then instead of placing that 3rd party API related jar file is WEB-INF/lib folder of every webapplication it is recomended to place only once in the server library folder.

→ In tomcat server the server library folder is

<Tomcat-home>\lib folder.

→ In weblogic server the server library folder is

<weblogic-home>\middleware\user-projects\domains\<domain-name>
(specific to one domain)

<weblogic-home>\middleware\wlserver_10.3\server\lib
(Common for all domains)

→ In Glass Fish the server library folder is

<GlassFish-home>\Appserver\domains\<domain-name>\lib\ext folder
(specific to one domain)

<GlassFish-home>\Appserver\lib folder
(Common for all domains)

→ keeping JAR files in server library folder is not recomended process.

reasons:- ① while moving the web application from one server to another server the jar files should be moved seperately,

② programmers alloays think about current webapplication development and they never bother (worry) about other parallel web applications development.

③ when the webresource programs of Java webapplication uses 3rd party API classes and interfaces then the Servlet Container searches for them in the following places and following order to

recognize them.

① In WEB-INF/class folder of current web application.

② In the jar files placed in WEB-INF/lib folder of current web application.

③ In the server library folder of Underlaying Server.

MyEclipse :-

Type : IDE s/w for Java Environment

Version : 8.x (compatible with jdk 1.5+)

Vendor : Eclipse org

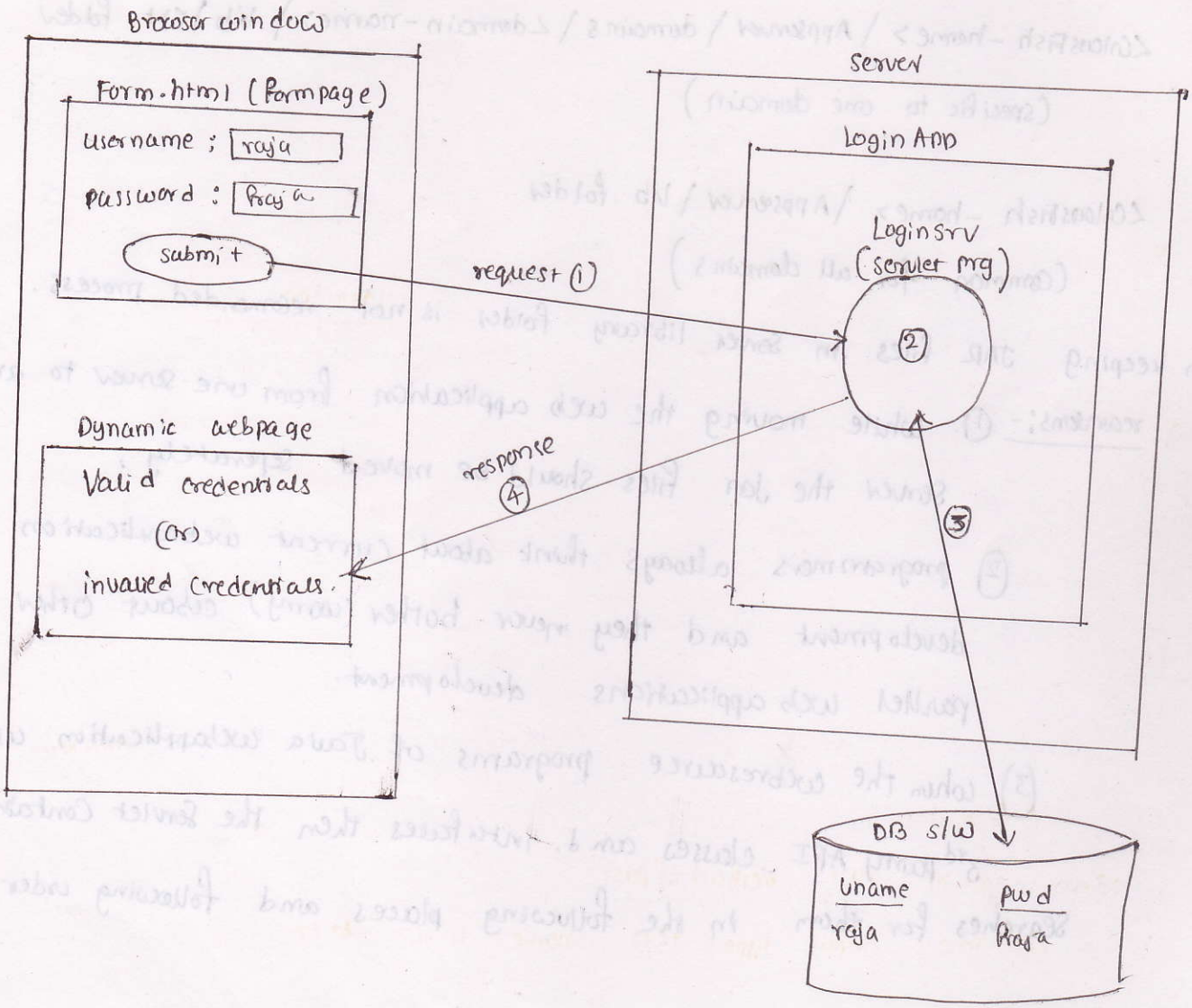
Commercial s/w

Gives tomcat as built-in server but allows to cfg all external servers

To download : www.Myeclipseide.com

for docs and help : " "

MyEclipse = Eclipse + Built-in plug-ins (to work with advanced technologies)



procedure to develop above webapplication by using MyEclipse 8.x

Step-I :- Create DataBase table in oracle.

SQL> create table userlist (uname varchar2 (20), pwd varchar2 (20));

Step-II :- Deside the SQL Query to be used

SQL> Select count (*) from userlist where uname = 'raja' and pwd = 'rani';

$\frac{\text{count}(*)}{1}$ (valied credentiency)

SQL> select count (*) from userlist where uname = 'raja' and pwd = 'rani1';

$\frac{\text{count}(*)}{0}$ (invalied credentiency)

Step-III :- launch MyEclipse -IDE by choosing its workspace folder
↳ (The place where projects will be stored).

MyEclipse menu ⟶ Subscription information ⟶ Submit key value to register.

Step-IV :- Create web project in MyEclipse IDE

File ⟶ new ⟶ webproject ⟶ project name [LoginApp] ⟶ finish

Step-V :- Add Ojdbc14.jar file to be libraries of the project.

Right click on project ⟶ Build path ⟶ Add external Archeives ⟶ browse and select Ojdbc14.jar

Step-VI :- Add form page to the webroot folder of the project

Right click on webroot ⟶ New ⟶ Html ⟶ Filename [form.html] ⟶ Finish

// form.html

```
<form action = "/uri" method = "post" >
```
↳ url pattern of loginsrv

```
Username : <input type = "text" name = "ti" /> <br />
```

Password: <input type="text" name="t2" />

<input type="submit" value="Submit" />

</form>

Step-VII :- Add the Servlet program to the project

Right click on Src folder → new → servlet → Name: LoginSrv → ☑ doGet() ☑ doPost()

→ next → Servlet/jsp/mappin URL: /lurl → finish

// LoginSrv.java (select to DB slw Interaction using Approach 2)

} Package imports (cntrl+shift+O get package cntrl+shift+i)

```
public class LoginSrv extends HttpServlet
{
    public void doGet(-,-) throws SE, IOE
    {
    try
    {
        // read form data
    String user = request.getParameter("t1");
    String pwd = request.getParameter("t2");

        // general settings
    PrintWriter pw = response.getWriter();
    response.setContentType("Text/html");

        // write Jdbc code
            // create Con object using oracle thin driver
    Class.forName("oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl",
                                             "scott","tiger");
            // Create jdbc prepared Statement object
PreparedStatement ps = con.prepareStatement("select count(*) from userlist
                                             where uname=? and pwd=?");
```

```
// set form data as the query param values
        ps.setString(1,user);
        ps.setString(2,pwd);
// execute the Query.
        ResultSet rs = ps.executeQuery();
// process the Result
        int result = 0;
        if(rs.next())
        result = rs.getInt(1);

        if(result == 0)
          pw.println("<font color=red size=6> invalied credentiency </font>");
        else
          pw.println("<font color=red size=6> Valied credentiency </font>");

// close jdbc objects
        rs.close();
        ps.close();
        con.close();
    }// try
    catch(Exception e)
    {
        e.printStackTrace();
    }
  }// doGet(-,-)
public void doPost(-,-) throws SE,IOE
  {
      doGet(request, response);
  }// doPost(-,-)
}// class
```

Step - VII :- Configure the external tomcat Server with MyEclipse IDE

Window → preferency → MyEclipse → servers → tomcat → Configure Tomcat 6.0

→ Enable → Tomcat home directory : D:\Tomcat 6.0 → Apply → ok.

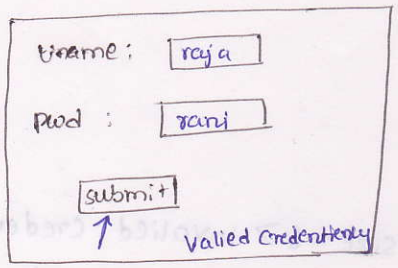
Step IX :- Run the project

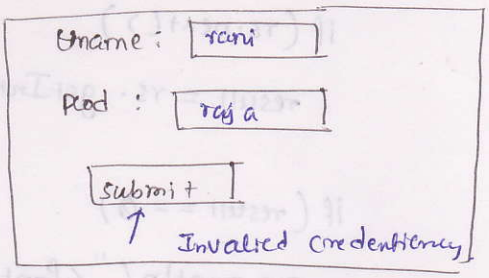
Rightclick on project → Runas → MyEclipse Server application → tomcat 6.x → ok.

Step X :- Test the project

Open browser window (Global Symbol in the toolbar)

→ Type url: http://localhost:2020/LoginApp/form.html





16/10/2012

Annotations :-

*) Data about Data is called as "Metadata".

Sofar we have used xml files to configure the resources and to perform metadata operations.

→ Annotations are the Java statements which are ultimate for xml files to perform metadata operations and to resources configuration.

Syntax of Annotation :-

@ <Annotation_name> (Param1 = Val1, Param2 = Val2 .....)

The two types of Annotations :-

① Documentation Annotations

Use full while working Documentation Comments (/** ..... */)

note :- use full to generate documentation of java code these are there in Java Since jdk 1.0.

② programming Annotations

→ given for programming

16/10/2012

Servlet 3.0 Annotations

By Shing Wai Chan on Dec 02, 2008

The JSR 315: Java Servlet 3.0 Specification expert group is in the process of making Public Review available. You can look at Rajiv's blog for more details. The reference implementation is available in GlassFish v3 nightly build. In Servlet 3.0, for ease of development, several new annotations are defined. These annotations are resided in the package **javax.servlet.annotation**. They are intended to provide meta data only. In other words, one still need to extend the corresponding class or implement the corresponding interface.

Now, one can have Servlet, Filter and ServletContextListener in a war file without `web.xml`. In this blog, I will discuss the following annotations:

- @WebServlet
- @ServletFilter
- @WebServletContextListener

Servlet Annotation (@WebServlet)

In JSR 315, one can specify the servlet meta data by using `@WebServlet`. For instance,

```
@WebServlet(name="mytest",
    urlPatterns={"/myurl"},
    initParams={ @InitParam(name="n1", value="v1"), @InitParam(name="n2",
value="v2")                                                              })
public class TestServlet extends javax.servlet.http.HttpServlet {
    ....
}
```

In this example, the class `TestServlet` is a servlet as it extends `HttpServlet`. The `@WebServlet` provides the following meta data:

- the name of the servlet, `mytest`, corresponds to `<servlet-name>` under `<servlet>` in `web.xml`
- the url pattern of the servlet, `/myurl`, corresponds to `<url-pattern>` under `<servlet-mapping>` in `web.xml`
- initialization parameters of the servlet, `n1=v1`, `n2=v2`, corresponds to `<init-param>` under `<servlet>` in `web.xml`

```
<init-param>
  <param-name>n1</param-name>
  <param-value>v1</param-value>
</init-param>
<init-param>
  <param-name>n2</param-name>
  <param-value>v2</param-value>
</init-param>
```

Note that in this case, `@InitParam` is used to specify the name/value pairs.

Servlet Filter Annotation (@ServletFilter)

One can specify the servlet filter meta data by using `@ServletFilter`. For instance,

```
@ServletFilter(urlPatterns={"/myurl"}.
    initParams={    @InitParam(name="mesg",    value="my    filter")    })
public    class    TestFilter    implements    javax.servlet.Filter    {
    ....
    public void init(FilterConfig filterConfig) throws ServletException {
        ....
    }
    public    void    doFilter(ServletRequest    req,    ServletResponse    res,
FilterChain    chain)    throws    IOException,    ServletException    {
        ....
    }
    public    void    destroy()    {
        ....
    }
}
```

In this example, the class `TestFilter` is a servlet filter as it implements `Filter`. The `@ServletFilter` provides the following meta data:

- the url pattern of the filter applied, `/myurl`
- initialization parameter of the filter, `mesg=my filter`, corresponds to `<init-param>` under `<filter>` in `web.xml`

  Note that in this case, `@InitParam` is used to specify the name/value pairs.

Servlet Context Listener Annotation (@WebServletContextListener)

One can specify the servlet content listener met data by using `@WebServletContextListener`. For instance,

```
@WebServletContextListener
public    class    TestServletContextListener    implements
javax.servlet.ServletContextListener    {
    ....
    public    void    contextInitialized(ServletContextEvent    sce)    {
        ....
    }
    public    void    contextDestroyed(ServletContextEvent    sce)    {
        ....
    }
}
```

In this example, the class `TestServletContextListener` is a servlet context listener as it implements `ServletContextListener`. The `@WebServletContextListener` provides the meta data that this is a servlet context listener in a given war file.

→ introduced from jdk1.5

→ all java technologies that are developed based on jdk1.5 gives support for annotations.

Example Documentation Annotations

@See @author @version @param and etc...

Example programing Annotations

@Override @FailSafe @Id @webServlet and etc....

→ xml files based resource configurations gives good flexibility of modification and Bad performance. (xml parsers are Heavy weight slws. so, they give bad performance).

→ Annotations give Bad flexibility of modification because they will be written directly in .Java file. But gives good performance.

→ All the Java technologies that are designed based on jdk1.5+ gives Support for programing Annotations.

Servlet 3.x, Struts 2.x, EJB 3.x, Spring 2.5+, Hibernate 3.2+ and etc...

→ Annotations are like xml tags. and the Annotation parameters are like xml tag attributes.

→ We can apply annotations at 3 levels.

① at resource level (on the top of class or interface)

② at method level (on the top of method definitions/declarations)

③ at Field level (on the top of instance variable declarations)

→ at the time of designing each annotation its level of utilization will be specified.

→ Every annotation is a special @interface. and the parameters of annotations are methods declared in that interface.

→ Every annotation interface will be implemented by the underlying JRE (or) Container (or) Server.

→ The implementation of Annotation interfaces is not the responsibility of programer.

Servlet 3.x supports Annotations based Servlet programing. so, web.xml is not required while working with Servlet 3.x.

The important annotations of Servlet3.x :-

@WebServlet (to configure servlet program)

@WebFilter (to configure ServletFilter program)

@WebListener (to configure ServletListener)

@WebInitParam (to configure ServletInitParameter)

and etc..... Somany other annotations are there.

Servers Supporting Servlet3.x programing :-

Tomcat 7.x

GlassFish 3.x

Weblogic 11.x

Jboss 6.x and etc...

Packages of Servlet 3.x api :-

javax.servlet pkg

javax.servlet.http pkg

javax.servlet.annotation pkg (contains annotations)

javax.servlet.descripter pkg

→ You can gather documentation about Servlet3.x annotations by using Javaee6 docs

NEJavaDVD/ST-I/JEEDocs\Javadocee6\Index.html.

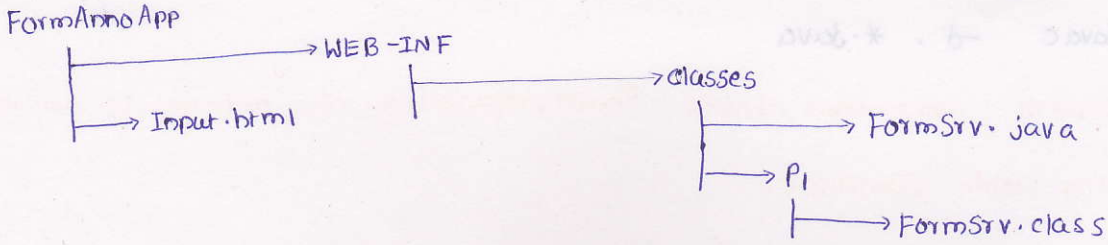
*) procedure develop, deployee and execute Servlet 3.x based web application

Step I:- Keep the Tomcat 7 server ready

to start Tomcat 7 server use <Tomcat7_home>\bin\Tomcat7.exe file

Step-II:- Add <Tomcat7_home>\lib\servlet-api.jar file to classpath.

Step-III:- prepare the deployement directory structure of webapplication having the webresource program.



NOTE:- do observe that there is no web.xml file

```
// Input.html
    <form action = "turi" >
        Name: <input type=text name=username> <br>
        <input type = Submit value=Send >
    </form>

// FormSrv.java
    package P1;
    import java.io.*;
    import javax.servlet.*;
    import javax.servlet.http.*;
    import javax.servlet.annotation.WebServlet;

@WebServlet(name = "abc", urlPatterns = {"/turi", "/test1"})
    logical name of Servlet
    Public class FormSrv extends HttpServlet
    {
    Public void doGet(HttpServletRequest req, HttpServletResponse res) throws SE, IOE
    {
        // general settings
        res.setContentType("text/html");
        PrintWriter pw = res.getWriter();

        //read form data
        String uname = req.getParameter("username");
    pw.print("Hello Mr:" +uname+ ": todays Date:" +new java.util.Date());

        // Close Stream
        pw.close();
    }
    public void doPost(HttpServletRequest req, HttpServletResponse res) throws SE, IOE
    {
        doGet(req, res);
    }
}
```

1> javac -d . *.java

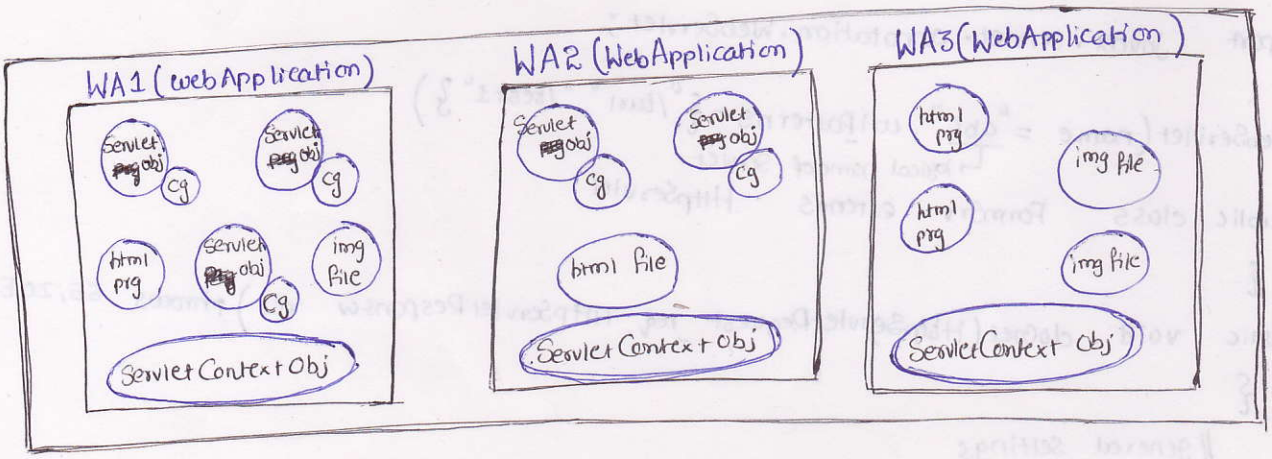
Step-IV :- deployee the above webapplication in <Tomcat7_home>\webapps folder

Step-V :- Test the webapplication

http://localhost:3030/FormAnnoApp/input.html

17/10/2012

Q:- What is the difference b/w ServletConfig object and ServletContext object?



cg ⟶ ServletConfig obj

ServletConfig obj :-

→ It is one per ~~Servlet program~~ our servlet class object / Jsp program.

→ It is the Object of a Underlaying ServletContainer supplied java class that implements javax.servlet.ServletConfig (I).

→ It is called righthand object to our Servlet / Jsp program.

→ Using this object we can gather information about CurrentServlet program and we can gather data from web.xml file

→ ServletContainer creates this object when instantiation event is raised on ↳ our Servlet class object creation ServletProgram.

→ Servlet Container destroyes this object when destruction event is raised in our Servlet program.
↳ when our Servlet class object is destroyed

→ This object is required to gather Servlet init parameter values from web.xml file.

ServletContext obj :-

→ It is one per web application. So, It is called the global memory of webapplication

→ It is the object of underlaying Servlet Container supplied java class that implements javax.servlet.ServletContext (I).

→ ServletContainer creates this object either when webapplication is deployeed (or) the Server is startedup.

→ ServletContainer destroyes this object when webapplication is stopped (or) reloaded (or) undeployeed.

→ Using this object we can gather global init parameters / context parameters & values from web.xml file.

→ Using this object we can gather the details about underlaying Server and the webresource programs of webapplication.

→ Using this object we can create the streams pointing to the webResource programs.

→ This object is Visible and accessible in all Servlet/Jsp programs of current webapplication.

Q:- In a Server 10 webapplications are deployeed in the Six webapplications are there in running mode and 4 webapplications are there in stopped mode. Can you tell me how many ServletContext objects are currently available in the Web~~application~~ Server?

Ans:- Six. (not ten bcz container destroyes ServletContext objects when their webapplications go to stopped mode).

Q:- In a webserver 10 webapplications are deployeed in that four webapplications are there in running mode, three webapplications are there in Stopped mode and other three webapplications are in Reloaded mode. Can you give me the count of Currently available ServletContext objects?

Ans:- 4 + 3 = 7

4 → for Running web applications
3 → for Reloaded web applications (means stopped and started.)

Q:- In a deployeed webapplication 10 servlet programs are there in that four ServletPrograms are enabled with <load-on-startup> and other three servlet programs are already requested by clients. Can you tell me how many ServletConfig objects are Currently available?

Ans:- 4 + 3 = 7

4 → for <load-on-startup> enabled servlets.
3 → for already requested servlets.

→ In servlet programing we never create the following objects

1. our servlet class obj
2. request obj
3. response obj
4. Servlet Config obj
5. ServletContext obj

All these objects will be created by ServletContainer and can be accessed by programmer in his servlet programs by using different techniques.

To access our servlet class obj

use "this" keyword

To access request, response objs

use the params of service(-,-) / doXxx(-,-) methods

To access ServletConfig obj

a) use the parameter of init(-) method

```
public class TestSrv extends HS|GS
{
```

```
        ServletConfig cg;
    public void init(ServletConfig cg)
    {
        this.cg = cg;
    }

    public void doXxx(-,-) throws SE, IOE
    {
        use cg here
        -----
        ----
    }
}
```

b) Call getServletConfig() method (belongs to GenericServlet (c))

```
public class TestSrv extends HS/GS
{
    public void init()
    {
        ServletConfig cg = getServletConfig();
        ------
        ------
    }
    public void doXxx(-,-) throws SE, IOE
    {
        ServletConfig cg = getServletConfig();
        ---
        ---
    }
}
```

↳ public method of GenericServlet class.

<u>TO Access ServletContext obj :-</u>

(i) use cg.getServletContext() method

```
public class TestSrv extends HS/GS
{
    public void init()
    {
        ServletConfig cg = getServletConfig(); // gives ServletConfig obj
        ServletContext sc = cg.getServletContext(); // gives ServletContext obj
        -----
    }
}
```

```
public void doXxx(-,-) throws SE,IOE
{
    ServletConfig cg = getServletConfig()
    ServletContext sc = cg.getServletContext();
    ----
}
}
```

b) use getServletContext() method (belongs GenericServlet(c))

```
public class TestSrv extends HS|GS
{
    public void init()
    {
ServletContext sc = getServletContext(); // gives access to ServletContext obj
    -----
    }
    public void doXxx(-,-) throws SE,IOE
    {
ServletContext sc = getServletContext(); // gives access to ServletContext
    -----
    }
}
```

↳ public method of GenericServlet class but it also internally uses ServletConfig object.

NOTE:- ① we can't access ServletContext object without accessing ServletConfig object directly or indirectly.

② Static methods, self class methods and Super class methods in subclass can be called without object.

Eg:- The above getServletConfig(), getServletContext() methods.

18/10/2012

We can pass data to our servlet program from outside the servlet program in two ways.

① As Form Data (request params) collected from End user (From client side).

② As Init/Context param values collected from web.xml & through program (From Server Side)

→ If the data to be passed is non-technical data (like name, address, age) then send it as a Formdata (Through programmer)

→ If the data to be passed is technical data (like jdbc driver class, url, db username, pwd and etc...) then send it as init/context param values (Through programer)

*) To make the Jdbc code of servlet program as flexible code to modify don't hard code Jdbc properties (Driver class name, Url, db User name, db password) directly in Servlet Program
↳ Technical data

*) It is recomended to gether these details from web.xml file as Servlet init parameter values by using the ServletConfig object.

→ Servlet init parameters are specific to each servlet program.

→ Every servlet init parameter contains one name and value (User defined)

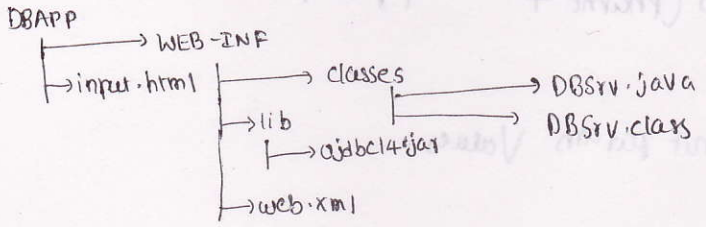
Servlet init parameters should be specified in web.xml file During the configuration of Servlet Program.

→ For improvised DB App application that gathers Jdbc properties from web.xml file to make the Jdbc code of servlet program as flexible code to modify referer application ④ of the page no's; 63-65.

line 363-380 :- init parameters of dbsrv program holding Jdbc properties.

to acheive the flexibility of modification

404-410 :- Reading Servlet-init parameter values from web.xml file.



Request url:
http://localhost:2020/DBAPP/input.html

Q:- What is the difference between request ~~paramters~~ and servlet init ~~paramters~~

request params	servlet init params
→ will be passed by End user from client side.	→ will be passed by programer from server side
→ Good for passing non-technical data to servlet program	→ Good for passing technical data to servlet program.
→ Servlet Program reads these values Using Request Object	→ Servlet Program reads these values Using ServletConfig object
→ Specific to each form page	→ Specific to each Servlet Program.
→ One request parameter can have multiple values and we can read, use those values	→ One init param can have multiple values but we can read only last value.

Different approaches of reading ServletInitParam values from web.xml file :-

① by using getInitParameter(-) method

```
ServletConfig cg = getServletConfig();  // access the ServletConfig obj
String s1 = cg.getInitParameter("dbuser");
String s2 = cg.getInitParameter("dbpwd");
```

Note:- In this Approach we must init param name to get its value.

ST-I | JEE Docs | Javadoc6 | index.html
Javax.servlet

② by using getInitParameterNames() method

```
Enumeration e = cg.getInitParameterNames();
While.hasMoreElements()
{
String pname = (String)e.nextElement();  // gives init param name.
String pval = cg.getInitParameter(Pname);  // gives init param value.
pw.println(Pname + " " + pval);
}
```

// gives all the init param values

→ To know the class name of ServletConfig object call getClass() on that object.

→ pw.println("
 cg object class name" + cg.getClass());

→ By using ServletConfig object we can know the object name of our current Servlet class (nothing but logical name of Servlet program given in web.xml file).

→ pw.println("
 current Servlet class object name/logical name" + cg.getServletName());

*) If multiple Servlet programs of a webapplication are looking to use same init parameter then instead of writing them in every ServletConfiguration it is recomended to write them only once in web.xml file as Context param values. so that, we can read and use them in every Servlet program of webapplication by using ServletContext object.

*) Context parameters are visible in all the webresource programs of webapplication

Eg:- (1) In web.xml file of "DBApp" application

```
<web-app>
    <context-param>
        <param-name> driver </p-n>
        <param-value> sun.jdbc.odbc.JdbcOdbcDriver </p-v>
    </c-p>
    <context-param>
        <p-n> url </p-n>
        <p-v> jdbc:odbc:oradsn </p-v>
    </c-p>
    <context-param>
        <p-n> dbuser </p-n>
        <p-v> scott </p-v>
    </c-p>
    <context-param>
        <p-n> dbpwd </p-n>
        <p-v> tiger </p-v>
    </c-p>
<servlet>
    <s-n> abc </s-n>
```

Context parameters and these are visible in all servlet programs of the web application

```
<s-c> DBSrv </servlet-class>
<load-on-startup> 1 </l-o-s>
</servlet>

<servlet-mapping>
    <s-n> abc </s-n>
    <u-p> /dburl </u-p>
</s-m>
<servlet>
    <s-n> xyz </s-n>
    <s-c> TestSrv </s-c>
</servlet>
    <s-m>
        <s-n> xyz </s-n>
        <u-p> /testurl </s-n>
</s-m>
    <welcome-file-list>
        <welcome-file> input.html </w-f>
    </w-f-l>
</web-app>
```

→ in Init() of DBSrv.java

```
public void init()
{
    try
    {
        // get access to ServletContext obj
        ServletContext sc = getServletContext();
        // read Context param values
        String s1 = sc.getInitParameter("driver");
        " s2 = sc.getInitParameter("url");
        " s3 = sc.getInitParameter("dbuser");
        " s4 = sc.getInitParameter("dbpwd");
        // create jdbc con obj
        Class.forName(s1);
        Con = DriverManager.getConnection(s2, s3, s4);
        // create prepared Statement object
```

```
ps = con.prepareStatement("select * from emp where empno = ?");
}//try
catch (Exception e)
{
    e.printStackTrace();
}
}//init()
```

→ If multiple Servlets of programs a web application are looking to use same jdbc properties Then it is recomended to take them as Context param Values in web.xml file for global visibility as shown above.

---

about components and business methods (Xml Entities)

③ → Service provider publishes the WSDL document in UDDI registry for global visibility.

④ → Service client queries WSDL document from UDDI registry

⑤ → Service client understands WSDL document and develops client application it is choice technology.

⑥ → The client application calls the IS methods of webservice component.

⑦, ⑧ web service client application calls the business method of webservice component and gathers the result by using SOAP over HTTP protocol.

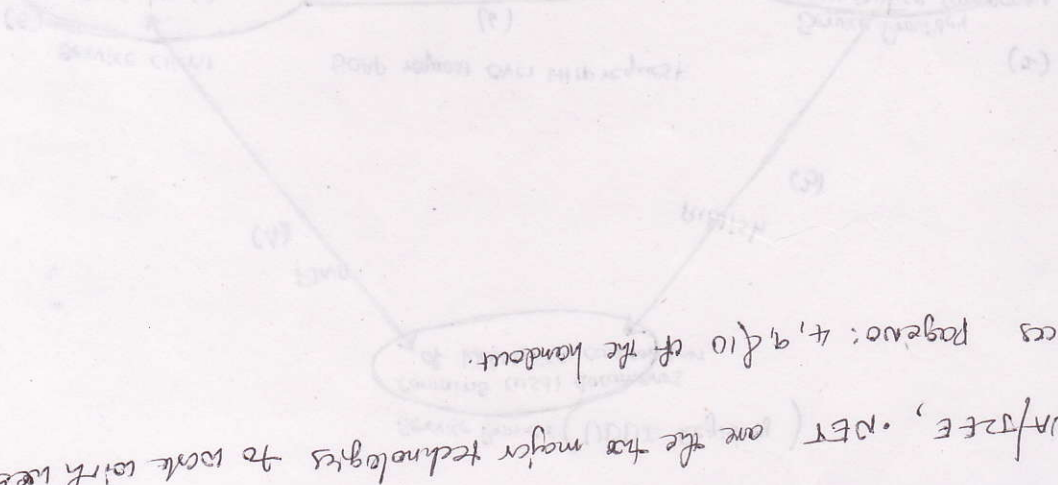
I → To develop all the resources of webservice enabled application in Java

II → Use JAXR API to make Service provider and Service client interacting with UDDI registry. II → Use JAXRPC/JAXWS to develop webservice components and clients.

III → Use JAXB to get the interaction b/w webservice client and component.

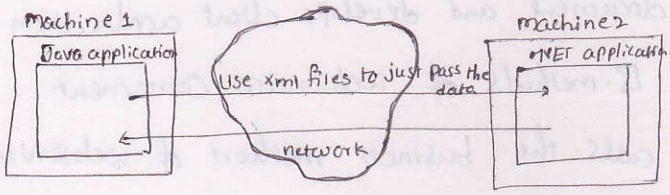
NOTE: As of Now JAVA/J2EE, .NET are the two major technologies to locate with webservices.

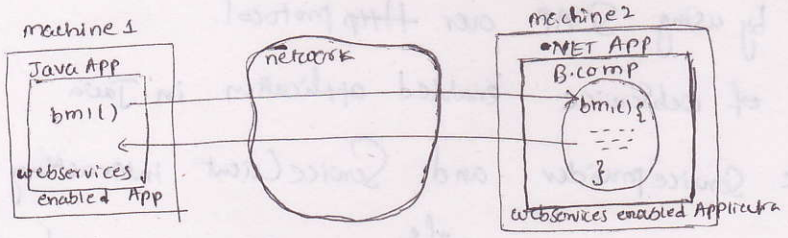
For advantages of webservices Refer no: 4, 9, 10 of the handout.



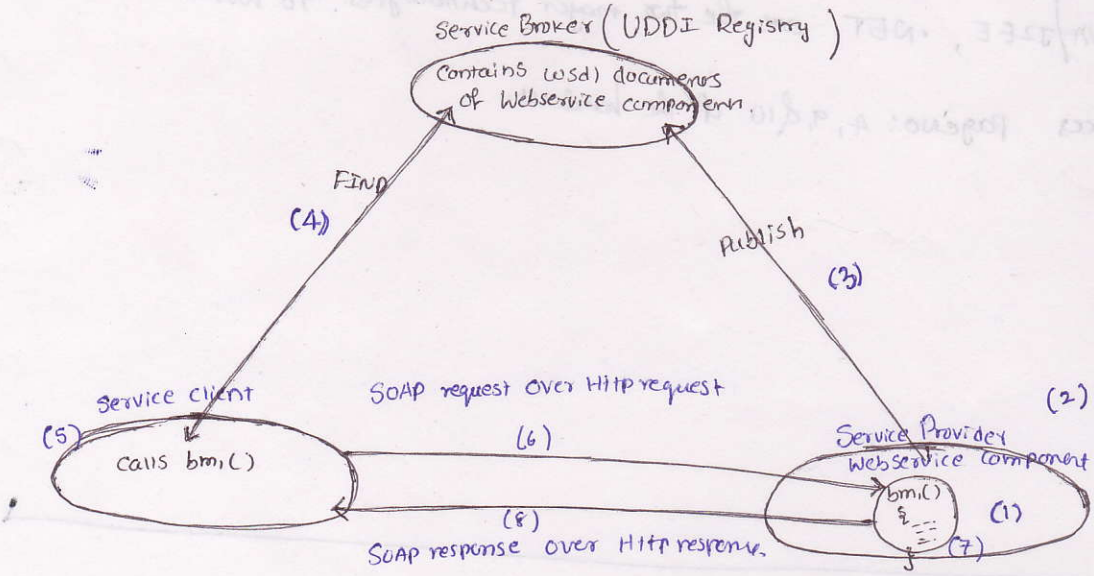
→ If u want to pass data b/w two incompatible applications then pass the data in the form of Xml files.

→ If u want to see the interaction b/w two incompatable applications then there must be developed as web-services enabled applications





Overview diagram of webservices :-



JAXR, JAX-RPC / JAX-WS, JAXB

Webservices is an Xml, SOAP, Http ~~Request~~ Based distributed technology that allows us to develop interoperatable Distributed applications by using the technology of programer choice.

→ All Hardware Components are interoperatable (works everywhere) Without worrying about Compatibility to develop interoperatable s/w Components Use web services. that means we can use .NET Components in Java and Vice versa.

W.r.to the diagram

(1) → Service provider develops web service component as web application it is choice technology (.NET, Java) having business methods.

(2) → Service provider Generates WSDL document for webservice component having the details

19/10/2012

Q:- can we give same name for servlet init parameter and context parameter placed in web.xml?

Ans:- Yes, possible, Bcz we use servlet-config object to read init parameter value and we use ServletContext object to read context parameter value.

Q:- What is the difference b/w servlet init parameters and context parameters?

init parameters	context parameters
→ specific to each servlet program	→ common for all the web resource program of web application
→ useful to pass technical data to servlet program.	→ useful to pass global technical data to multiple web resource programs.
→ should be read by using ServletConfig object in servlet program.	→ should be read by using ServletContext object in servlet program
→ allowed duplicate init parameters names	→ does not allow duplicate duplicate context parameter names.

Different approaches of reading context param values from web.xml file :-

Approach 1 :- (Using sc.getInitParameter(-) method)

```
ServletContext sc = getServletContext();   // gives ServletContext obj
String s1 = sc.getInitParameter("dbPwd");  // gives context param value
```

here we must know context param name to get its value

Approach 2 :- (Using sc.getInitParameterNames() method)

```
Enumeration e = sc.getInitParameterNames();
While(e.hasMoreElements())
{
    // get each context param name
String pname = (String) e.nextElement();
    // get each context parame value
String pval = sc.getInitPameter(pname);
pw.println(pname + "→" + pval);
}
```

here we get all context param names and values.

→ To make the jdbc code on servlet program as the flexible code to modify we can also think about taking properties file support but not recomended. Because it is better to use web.xml file for that purpose instead of taking separate properties file.

→ The latest servlet specification version is 3.0, Tomcat 7 is developed based on Servlet3.0 specification. Tomcat 6, Glass Fish 2.x, weblogic10.3 are developed based on Servlet Specification 2.5. we can also gather the following additional details by using ServletContext Object.

```
// get Access to ServletContext obj
ServletContext sc = getServletContext();
pw.println("<br> sc obj class name" + sc.getClass());
pw.println("<br> the underlying server info" + sc.getServletInfo());
pw.println("<br> The servlet specification version" + sc.getMajorVersion() +
                                                    "." + sc.getMinorVersion());
pw.println("<br> context path of current webapplication" + sc.getContextPath());
```

Q:- ~~What is the xdi~~

**** ServletContainer creates the ServletConfig object after the execution of our servlet class constructor and before the execution of init(-) lifecycle method during the instansiation and initialization of our servlet program. You can't access ServletContext object with out using ServletConfig object (directly or indirectly)

This indicates ServletConfig object is not visible and ServletContext object is not accessible in our ~~Ser~~ constructor of our servlet program.

But these two objects are visible and accessible in the init(-) of servlet program

**** Q:- What is the difference b/w code placed in the constructor and code placed in the init(-) of our servlet program?

Ans:- The code placed in the constructor can't read and use servlet init parameters, context parameters. but this is possible in the code placed in the init()

****

Q:- What is the difference b/w Generic Servlet class and HttpServletClass ?

Generic Servlet	HttpServlet
→ This class based Servlet program can handle different protocols based request, response	→ This class based Servlet program can handle only Http protocol based requests, responses
→ Doesn't support SessionTracking (fully)	→ Supports SessionTracking completely.
→ Abstract class with one abstract methods	→ Abstract class without abstract methods.
→ provides only Service(-,-) method (public) to process the request	→ Provides two Service(-,-) methods & 7 doXXX(-,-) methods to process the request.
→ gives ServletRequest, ServletResponse obj's	→ Gives HttpServletRequest, HttpServletResponse Objects
→ Does not allow to work with all the features of protocol Http	→ Allows.

Approach-3 :-

JDBC con pool (Which contains set of readyly available jdbc con objs)

① DriverManaged Connection pool
→ useful for standalone applications (which run outside the server)

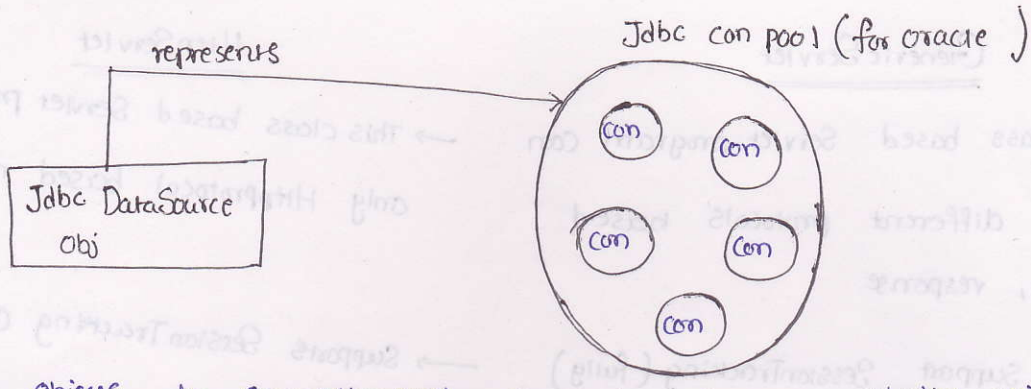
② ServerManaged ConnectionPool
→ available in webservers, application servers
→ useful in deployable applications (like webapplications, ejb comps)

JDBC Conection Objects

① Direct JDBC con obj
→ It is created by programer manually

```
Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");
Connection con = DriverManger.getConnection(-,-,-);
```

② Pooled Jdbc Con obj
→ It is the con obj that is collected from jdbc con pool.



→ all Connection objects in Connection pool represents the connectivity with Sing DataBase slw.

Eg:- Jdbc Connection pool for oracle means "all Connection objects in that pool represents Connectivity with Oracle DB slw".

→ JDBC DataSource object represents Jdbc connection pool. So, each connection object from pool must be accessed to DataSource object.

→ JDBC DataSource object means It is the object of a DriverSupplied Java class that implements javax.sql.DataSource (I).

→ To provide global visibility to objects and object references we place them in a registry slw having nicknames and alias names.

→ To provide global visibility to DataSource object its reference will be placed in registry slw.

The following are the Registry slws:-

GlassFish registry,
Weblogic registry,
Cos registry,
RMI registry,
and etc....

Every Server gives one built-in registry slw.

Weblogic registry is built-in registry of weblogic Server

GlassFish registry is built-in registry of GlassFish Server

jdbc api

Java app ⟶ DB s/w

(Java.sql, javax.sql pkgs)

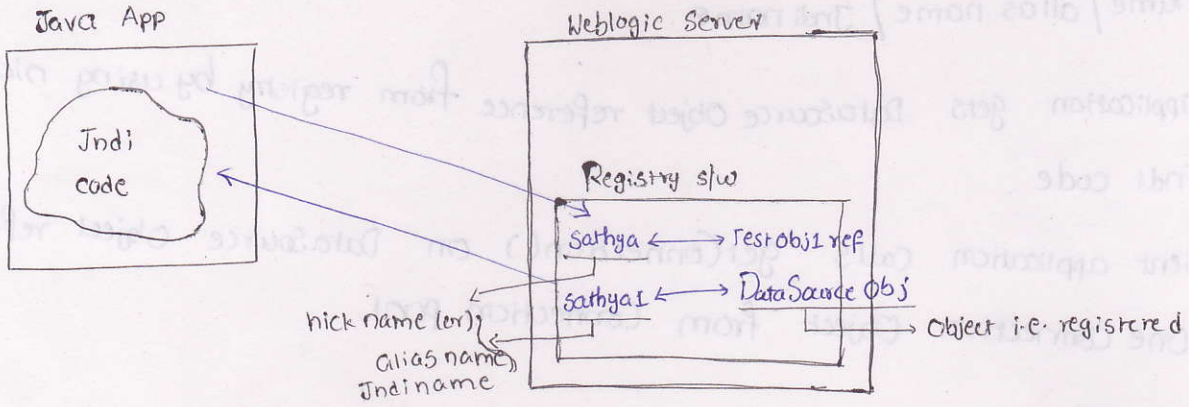
Jndi

(Java naming and directory Interface)

Java app —jndi api⟶ Registry s/w

(javax.naming)

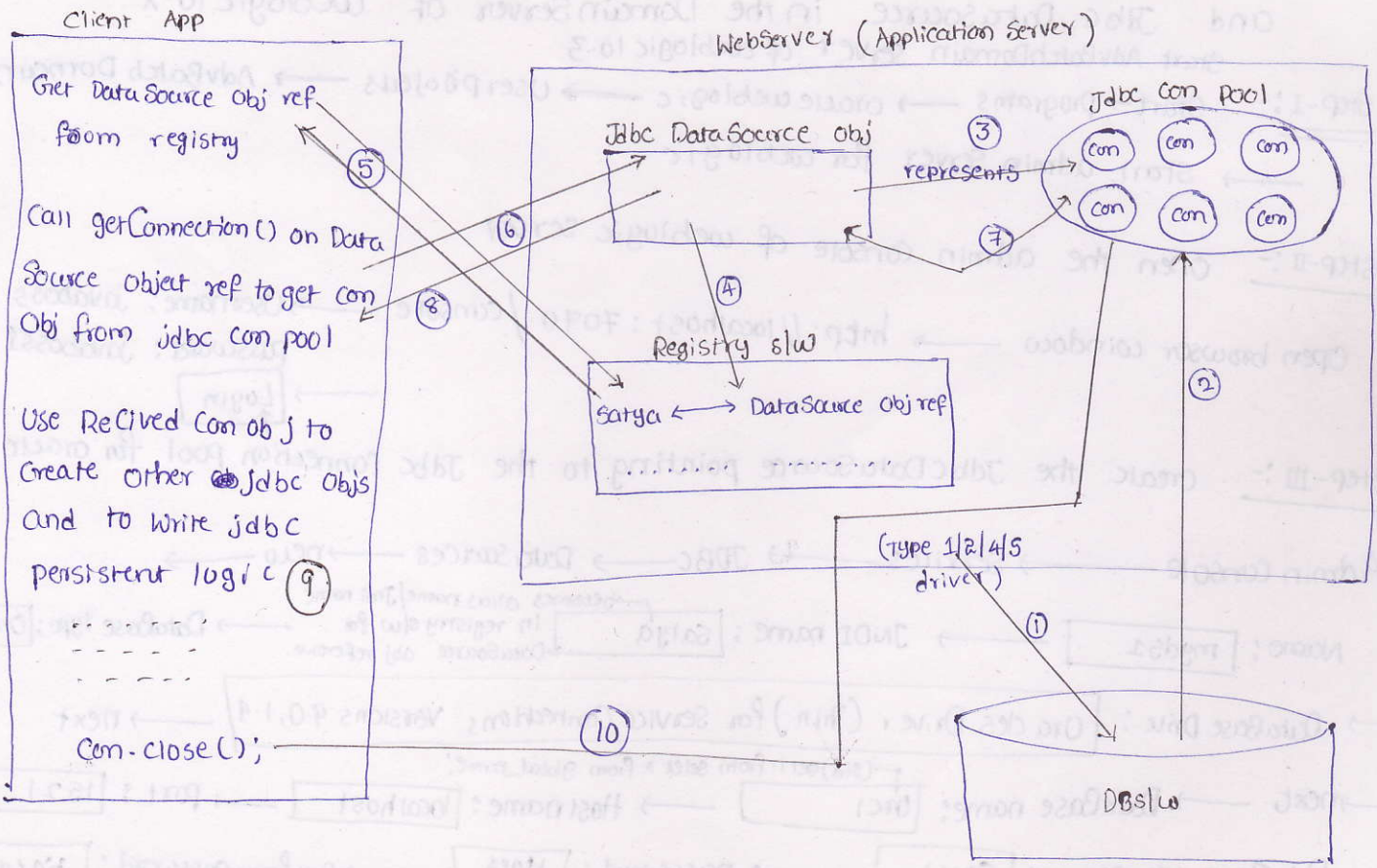
Jndi api is the part of JSE module



jdbc connection obj represents the connectivity b/w Java app and DB s/w

Jndi Initial/Context obj represents the connectivity b/w Java app and Registry s/w.

Example Scenario diagram to use Server managed Jdbc Connection pool in our Java application :-



W.r.to the diagram

①&② Programmer uses type 1/2/4/3 Jdbc driver to interact with database s/w and to create set of Jdbc Connection objects.
set of having readyly available Jdbc connection objects.

③&④ programer creates Jdbc dataSource objects representing Jdbc Connection pool and keeps that object reference in registry s/w for global visibility having nick name / alias name / Jndi name.

⑤ Client application gets DataSource Object reference from registry by using nickname and Jndi code

⑥,⑦,⑧ Client application calls getConnection() on DataSource object reference to get one connection Object from Connection pool.

⑨ refer diagram

⑩ Client application calls the con.close(). this method returns Jdbc Connection Object back to connection pool. So that Connection Object can be used by Other request and Other clients.

*) procedure to create Jdbc Connection pool for oracle and Jdbc DataSource in the Domain Server of weblogic 10.x

Start AdvBatchDomain server of weblogic 10.3

Step-I :- Start → programs → oracle weblogic → User projects → AdvBatch Domain → Start admin Server for weblogic

Step-II :- Open the admin console of weblogic server

Open browser window → http://localhost:7070/console → Username: javaboss
password: javaboss1
→ Login

Step-III :- create the JdbcDataSource pointing to the Jdbc connection pool for oracle.

Admin Console → services → JDBC → DataSources → new →

Name: mydS1 → JNDI name: Satya (becomes alias name/Jndi name in registry s/w for DataSource obj reference) → DataBase type: Oracle

→ DataBase Drive: Oracles Driver (Thin) for Service Connections Versions 9.0, 1.9 → next

→ next → DataBase name: orcl (sid) get it from select * from global_name; → Hostname: localhost → port: 1521

→ DataBase username: Scott → password: tiger → Conform password: tiger
→ next

→ TestConfiguration → next → [✓] AdminServer → Finish

NOTE:- In the above step creates DataSource objects representing JDBC connection pool for oracle and also keeps DataSource Obj reference in weblogic registry having nickname (or) alias name (satya)

→ Specify the param values for Jdbc connection pool

Admin Console → Services → DataSources → launch [myds1] (Right side) →

[Connection pool] → Initial Capacity : [4] → Maximum Capacity : [30] →

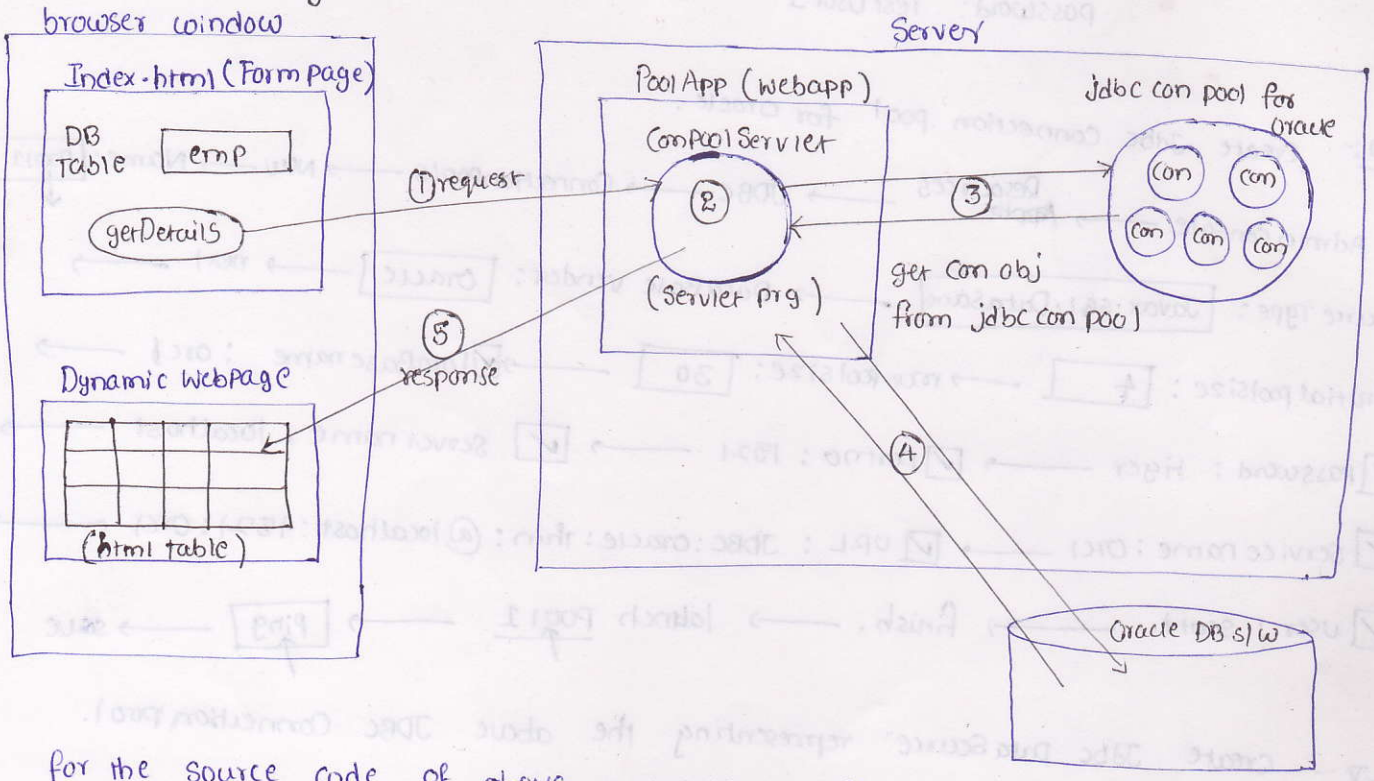
Capacity increment : [2] → Save → Advanced → Shrink Frequency : [500]

↳ no. of Connection Objects to be created when there is a need of creating new Jdbc Connection Obj's in the connection pool

→ Save

↳ The idle Jdbc connection Objects of connection pool at regular intervals (after every 500 seconds)

*) Example application on Approach 3 base Servlet to Database s/w Communication (Servlet prg using Server managed Jdbc Connection pool)



for the source code of above application refer Application ⑤ of the Page No: 65 to 67

colcount → 3

BFR	rs(Resultset obj)		
101	raja	hyd	1
345	ramesh	vizag	2
789	ravi	chennai	3
			ALR

→ always display application Error related messages as non-technical messages by keeping Non-technical Endusers in mind. for that this also recomended to Write Some presentation logic in the catch blocks of the Servlet program.

(*) procedure to create Jdbc datasource representing the Jdbc Connection pool for oracle in GlassFish 2.x server.

Step-I :- copy OJdbc14.jar file to <GlassFish_home>\Appserver\domains\mydomain1\lib\ext folder

Step-II :- Start the Domain Server of GlassFish (mydomain1)

D:\sun\Appserver\bin > asadmin start-domain mydomain1

Step-III :- Open the administration console of mydomain1 server. Openbrowser window → Type this url http://localhost:4343

Username: testuser1
password: testuser1

Step-IV :- Create Jdbc Connection pool for Oracle.

Admin Console → Resources → JDBC → ConnectionPools → New → Name: [pool1]

Resource Type: [Javax.sql.DataSource] → DataBase vendor: [Oracle] → next →

Initial poolsize: [4] → max poolsize: [30] → ☑ DataBase name : orcl →

☑ password : tiger → ☑ portno : 1521 → ☑ server name : localhost →

☑ Service name : orcl → ☑ URL : JDBC:oracle:thin:@localhost:1521:orcl →

☑ user : scott → finish. → launch pool1 → [Ping] → save

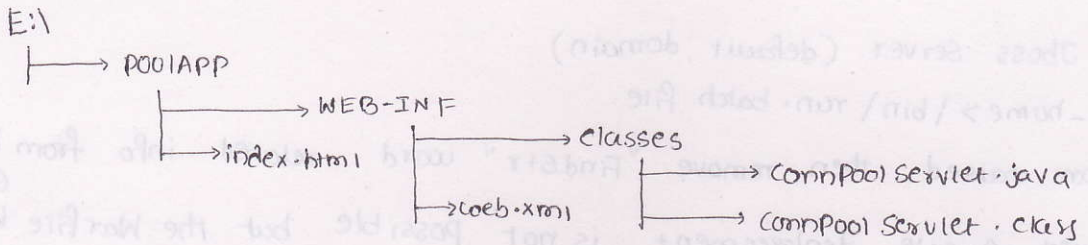
Step-V :- Create Jdbc DataSource representing the above JDBC Connection pool.

Admin Console → Resources → JDBC → JDBC resources → [New] →

JNDI Name: [Satya] (any name can take) → poolname: [pool1] → OK.
→ refer previous step.

procedure to deployee Application (5) of the Page no: 65-67 in mydomain1 Server of glass fish

Step-I :- prepare war file representing PoolApp web application



E:\PoolApp > jar cf PoolApp.war .

Step-II :- make sure that mydomain1 Server of glassFish in running mode

Step-III :- deployee the above war file in mydomain1 Server.

copy PoolApp.war file to Glassfishhome\Appserver\domains\mydomain1\autodeployee folder.

Step-IV :- Test the web application

Open Browser window → Type this URL http://localhost:6565/PoolApp/

Enter the table name: [emp] [submit]

Working with ServerManaged ConnectionPool is nothing but applying middleware service in our web application.

29/10/2012

JBOSS

Type : Appn Server slw

Version : 5.x (Compatible with JDK1.6)

Vendor : Apache foundation (Red Hat)

Creator : marc fluery

OpenSource (From Apache Foundation)

Commercial (from Red Hat)

Default port NO for web apps : 8080

Allows to create Domains but gives 5 default Domains. They are

1) all
2) default
3) minimal
4) Standard
5) web

To download s/w → www.Jboss.org (or) www.Apache.org

Download s/w as zip file & extract it for installation Jboss-5.1.0.GA-Jdk6.zip file

→ To start the Jboss server (default domain)

<Jboss_home>\bin\run.batch file.

→ If problems are raised then remove "FindStr" word related info from line No's 60 & 67.

→ In Jboss server console deployement is not possible but the war file based hard deployement is possible.

→ When Jboss server is started the domain whose name is "default" will be activated automatically.

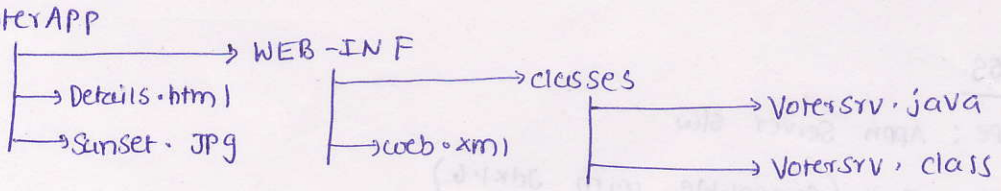
→ To change "http" port no. of Jboss server use server.xml file of any domain related folder structure.

To change the port number of default named domain server :-

Goto <Jboss_home>\server\default\deployee\Jboss web.sar\server.xml ---> change the port attribute value of first <connector> tag.

*) procedure to deployee the Java web appn in default domain server of Jboss :-

Step ① :- prepare war file representing the web appn



E:\VoterApp > Jar cf VoterApp.war

Step ② :- Start Jboss server use <Jboss_home>\bin\run.batch file

Step ③ :- Deployee the war file

copy the above VoterApp.war file to <Jboss_home>\server\default\deployee folder

Step ④ :- Test the appn

http://localhost:5555/VoterApp/Details.html

Summery Table

Server	Hard Deployement	Console Deployement
Weblogic	Directory (or) war file (autodeployee folder)	Supports
Jboss	War file (deployee folder)	does not supports
GlassFish	Directory (or) War file (auto deployee folder)	supports
Tomcat	Directory (or) war file (webapps folder)	supports
Websphere	Does not Support hard deployement	Supports

Configuring Different Servers in IDE'S

→ procedure to Configure AdvBatch domain server with MyEclipse IDE (in weblogic)

Window menu → preferences → [+] MyEclipse → [+] Servers → [+] Weblogic

→ [+] Weblogic 10.x → ⊙ enable → Username [Javaboss] →

password [Javaboss1]

Beahome directory : [D:\oracle\middleware]

Execution domain root : [D:\oracle\middleware\user-projects\domains\Advbatch Domain]

→ Apply → OK

→ procedure to Configure mydomain1 Server of GlassFish 2.x with MyEclipse IDE.

Window menu → preferences → [+] MyEclipse → [+] servers → [+] GlassFish

→ [+] GlassFish 2.x → ⊙ enable → Home Directory : [D:\sun\AppServer]

Domain name : MyDomain1

→ Apply → Ok

→ procedure to Configure Jboss 5.x Server with MyEclipse IDE

window → preferences → [+] MyEclipse → [+] Servers → [+] JBoss →

[+] JBoss 5.x → ⊙ enable → JBoss home directory : E:\Jboss 5.x soft\Jboss-5.1.0.GA

Server name : [default] → domain name is default

→ apply → Ok

→ procedure to Configure TomcatServer with NetBeans IDE

Step① :- Tools menu → Servers → add Server → Tomcat 6.0 →

Server Location : D:\Tomcat 6.0

Username : admin

Password : admin → Finish

→ Startup tab → ☑ use custom Catalina script → browser select D:\<Tomcat_home>\Tomcat 6.exe

Step② :- Link Tomcat Server with web project

Right click on web project → properties → ☑ Run → Server : Tomcat 6.0 → OK

Step③ :- Run the web project.

→ procedure to Configure weblogic Server with NetBeans IDE :-

Tools menu → Servers → add server → Oracle weblogic Server →

Server location : c:\oracle\middleware\wlserver-10.3 → next →

Local instances : Admin server [Localhost : 7070 ▽

Username : Javaboss

Password : Javaboss 1 → finish → close

→ procedure to Configure JBoss 5.x server with NetBeans IDE :-

Tools → Server → add server → JBoss Application server → (Browse)

Server Location : E:\Jboss 5.x soft\Jboss-5.1.0.GA → next →

Domain → default → finish → close

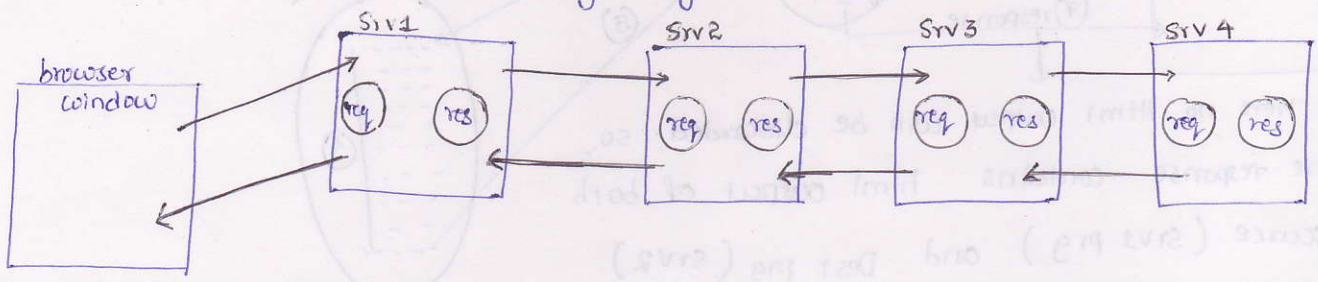
Servlet chaining :-

30/10/2012

→ The process of making multiple Servlet programs communicating with each other in a chain process the client generated request is called "Servlet chaining"

→ Servlet chaining is all about makeing Servlet programs interacting with other servlet programs

→ all the Servlet programs of Servlet chaining will use same request and response objects because ~~they~~ a process same requests given by client.

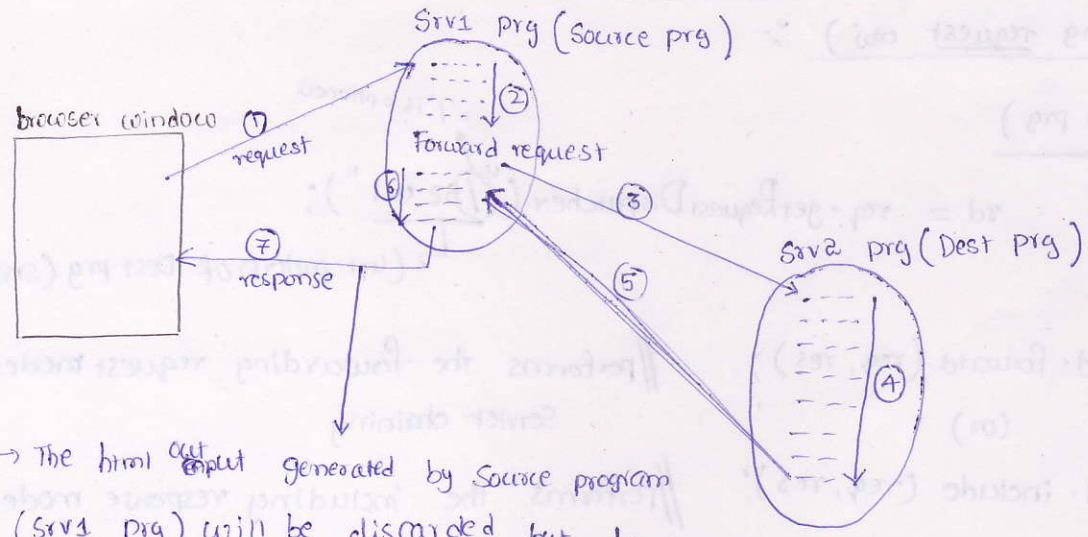


The two modes of Servlet chaining

- → Forwarding Request mode of Servlet chaining
- → including Response mode of Servlet chaining

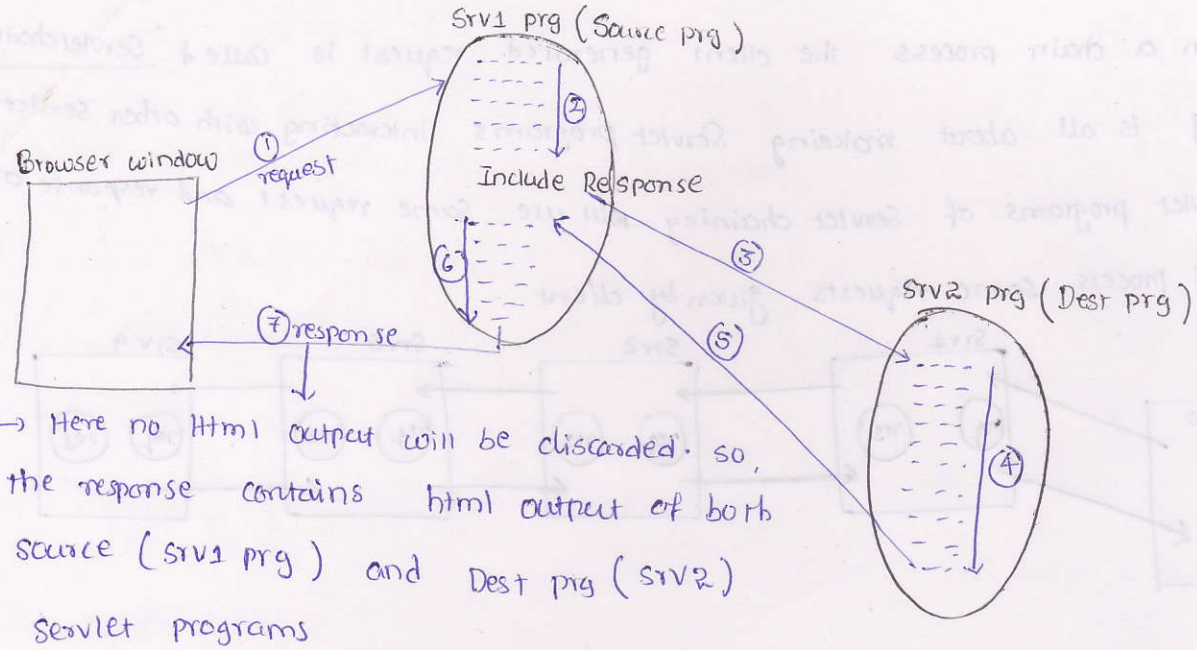
Any no of Servlet programs can be there in Servlet chaining (Generally this count is >=2)

Forwarding Request Mode of Servlet chaining



→ The html output generated by Source program (Srv1 prg) will be discarded but html output of Destination program (Srv2 prg) goes to browser window as response through Source program (Srv1)

Including Response mode of Servlet Chaining



→ Here no Html output will be discarded. so, the response contains html output of both source (srv1 prg) and Dest prg (srv2) servlet programs

→ The source servlet program should use RequestDispatcher Object to communicate with Destination program through servlet chaining.

→ RequestDispatcher Obj is the object of underlying ServletContainer supplied java class that implements javax.servlet.RequestDispatcher (I).

There are 3 approaches to create RequestDispatcher object

Approach 1 :- (by using request obj) :-

In Srv1 prg (Source prg)

'/' is optional

```
RequestDispatcher rd = req.getRequestDispatcher("/S2 url");
```

(url pattern of Dest prg (srv2 prg))

```
rd.forward(req, res);      // performs the forwarding request mode of servlet chaining
(or)
rd.include(req, res);      // performs the including response mode of servlet chaining
```

Approach 2 :- (by using ServletContext obj) :-

In source prg (srv1 prg)

Here mandatory

```
ServletContext sc = getServletContext();
RequestDispatcher rd = sc.getRequestDispatcher("/S2url");
```

(url pattern of Dest prg (srv2 prg))

rd.forward(req, res);

(or)

rd.include(req, res);

Approach 3 :- (by using ServletContext obj) :-

ServletContext sc = getServletContext();

RequestDispatcher rd = sc.getNamedDispatcher("abc");

sc → ServletContext obj

"abc" → logical name of Destination prg given in web.xml file

rd.forward(req, res);

(or)

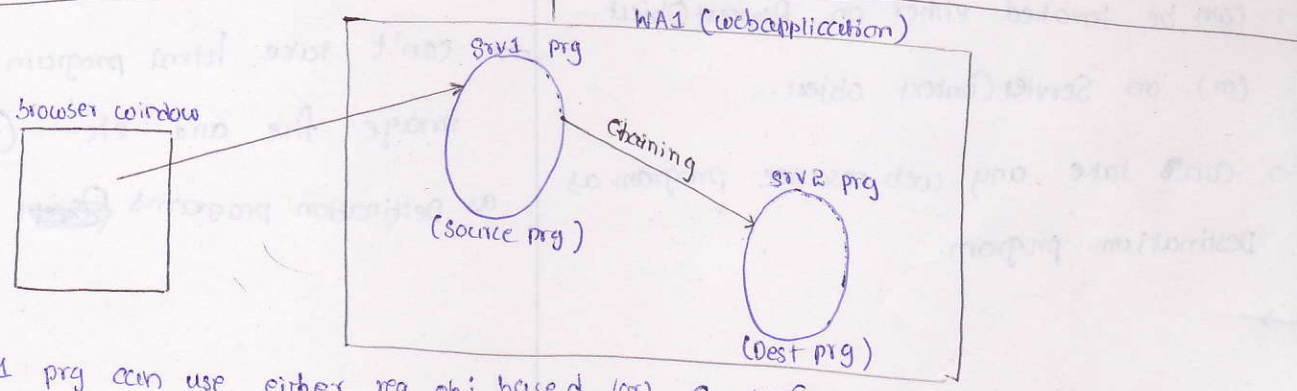
rd.include(req, res);

Q:- What is the difference b/w Creating RequestDispatcher either by using request obj (or) by using ServletContext object

Ans:-

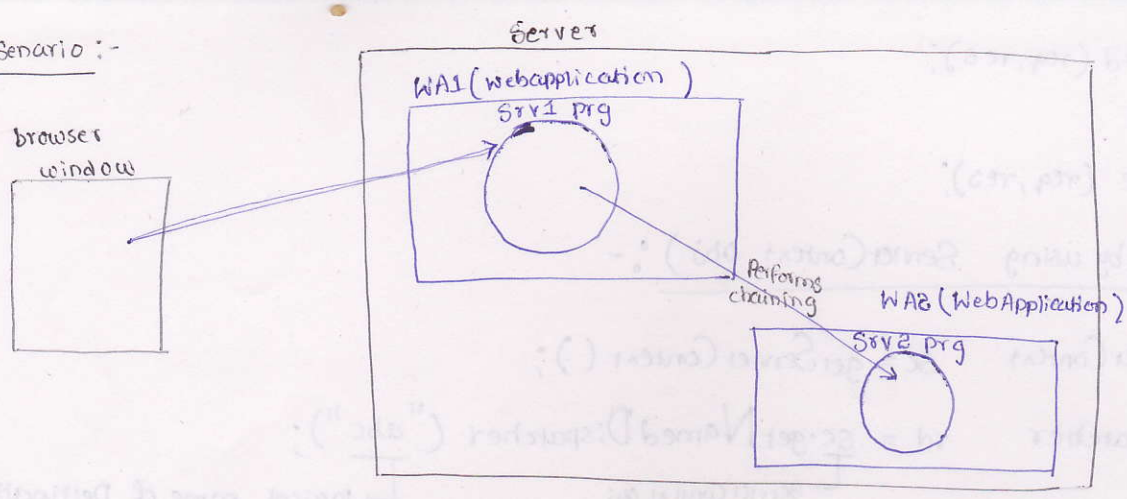
By using Request object	By using ServletContext obj
→ Source servlet prg and Destination prg must be there in ~~same~~ same web application.	→ Source servlet prg and Destination prg can be there in same webapplication (or) two different webapplication of same server.
→ use getRequestDispatcher() to get RequestDispatcher object	→ use either getRequestDispatcher() (or) getNamedDispatcher() to get RequestDispatcher obj.
→ we must know url pattern ~~(or) logical name~~ of Destination Servlet program (or) file name of Destination html, jsp program	→ we must know url pattern / logical name of Destination Servlet prg (or) file name of Destination html / jsp program

1st scenario :-



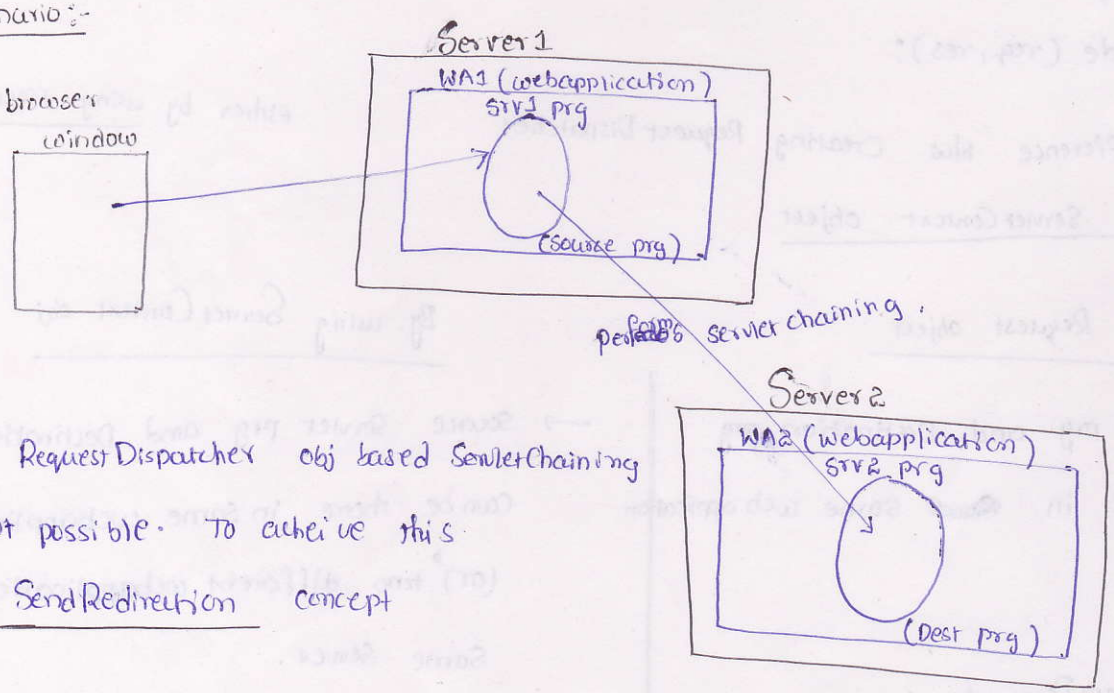
Here Srv1 prg can use either req obj based (or) ServletContext obj based RequestDispatcher object for ServletChaining with Srv2 prg.

IInd Senario :-



Here the Srv1 prg must use the ServletContext obj based RequestDispatcher obj for servlet chaining

IIIrd Senario :-



here RequestDispatcher obj based ServletChaining is not possible. To acheive this use SendRedirection concept

Q:- What is the diff b/w getRequestDispatcher (-) and getNamedDispatcher (-).

getRequestDispatcher (-)	getNamedDispatcher (-)
→ expects url pattern of Destination Servlet program / file name of Destination jsp / html program as argument value.	→ expects the logical name of Destination servlet / jsp program as argument value.
→ can be invoked either on RequestObject (or) on ServletContext object.	→ can be invoked only on ServletContext obj
→ can take any web resource program as Destination program.	→ can't take html program, text file, image file and etc. (exept servlet, jsp prg). as Destination programs
→	

RequestDispatcher obj pointing to Dest jsp prg :-

RequestDispatcher rd = req / sc.getRequestDispatcher("/ABC.jsp");

(or)

RequestDispatcher rd = req / sc.getRequestDispatcher("/testurl");
↳ (url pattern of the Dest jsp prg)

(or)

RequestDispatcher rd = sc.getNamedDispatcher("XYZ");
↳ (logical name of Dest jsp prog)

RequestDispatcher obj pointing to Dest html prg :-

RequestDispatcher rd = req / sc.getRequestDispatcher("/test.html");

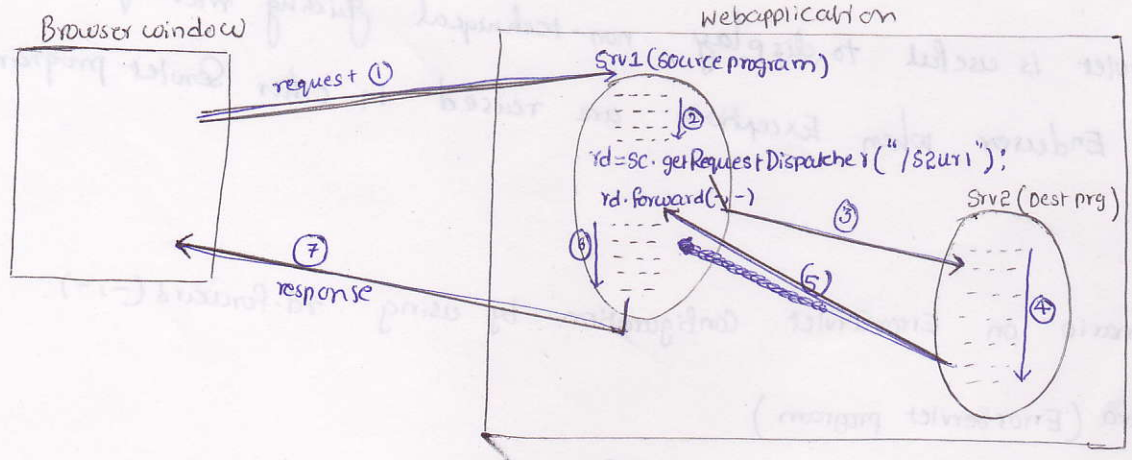
→ Configuration of Servlet program in web.xml is mandatory. So, In Servlet chaining we can identify Servlet prg either with url pattern or with logical name

→ Configuration of JSP program in web.xml is optional. So, In servlet chaining we can identify the jsp prg using its file name or logical name or url pattern.

→ we can not cfg html file in web.xml. So In Servlet chaining it will be identified with its file name.

Understanding rd.forward(-,-) :-

31/10/2012



The html output of srv1 prg will be discard and only the html output of srv2 prg goes to Browser window as response.

Key points :-

(i) → rd.forward(-,-) is there perform forwarding request mode of Servletchaining.

(ii) → The entire html output of Srv1 prg will be discarded but the html output of Srv2 prg goes to browser window as response.

(iii) → Statements placed before and after rd.forward(-,-) in Srv1 program will be executed but their html output will be discarded.

(iv) → Both Srv1 and Srv2 will use Same request and response Object. so, the request data Coming to Srv1 is visible and accessible in Srv2.

(v) → To pass additional data from Srv1 to Srv2 use the request attributes

(vi) → Srv1 and Srv2 Can be there in a Same webapplication or in two different web applications of Same Server.

(vii) → Srv2 Can be a Servlet program (or) Jsp program (or) Html program (or) Textfile and etc...

(viii) → rd.forward(-,-) is very useful to configure ErrorServlet in webapplication.

Q:- What is ErrorServlet?

Ans: Servlet program of webapplication that executes only when the exception is raised in other Servlet programs of webapplication is called as ErrorServlet.

This Servlet is useful to display non-technical guiding messages on Browser window for Enduser when Exceptions are raised in other Servlet programs of webapplication.

Example Senario on ErrorServlet Configuration by using rd.forward(-,-).

```
// ErrSrv.java (ErrorServlet program)
public class ErrSrv extends HttpServlet
{
    public void doGet(-,-) throws SE,IOE
    {
        // general settings
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");
```

```
// Display non-Technical Guiding msgs for end user

pw.println("<br> <font color = red > Internal problem </font >");

pw.println(" <br> <a href = "home.html"> home </a> ");

// close Stream

    pw.close();

}// doGet(-,-)


public void doPost(-,-) throws SE, IOE
  {
      doGet(req, res);
  }//doPost(-,-)

}//class
```

Configure "ErrSrv" prg in web.xml file having "/errurl" url pattern.

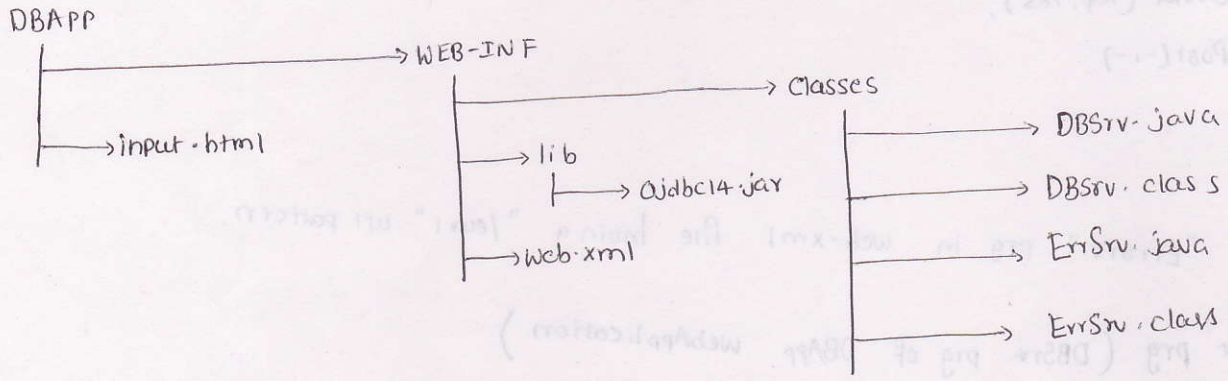
MainServlet prg (DBSrv prg of DBApp WebApplication)

DBSrv.java

```
public class DBSrv extends HttpServlet
  {
      public void init()
        {
          ----
          ----
          ---
        }

      Public void doGet(-,-) throws SE, IOE
        {
          try
          {
            ---
            --- Code that may raise exception
            --
          }//try
          catch(Exception e)
          {
            //forward the request to ErrSrv prg

RequestDispatcher rd = req.getRequestDispatcher("/errurl");

        rd.forward(req, res);
        }//catch
      }//doGet(-,-)
```

```
public void doPost(-,-) throws SE, IOE
  {
      doGet(-,-);
  }

public void destroy()
  {
      ----
      ----
  } //destroy
} //class
```

Deployement Directory Structure of DBAPP

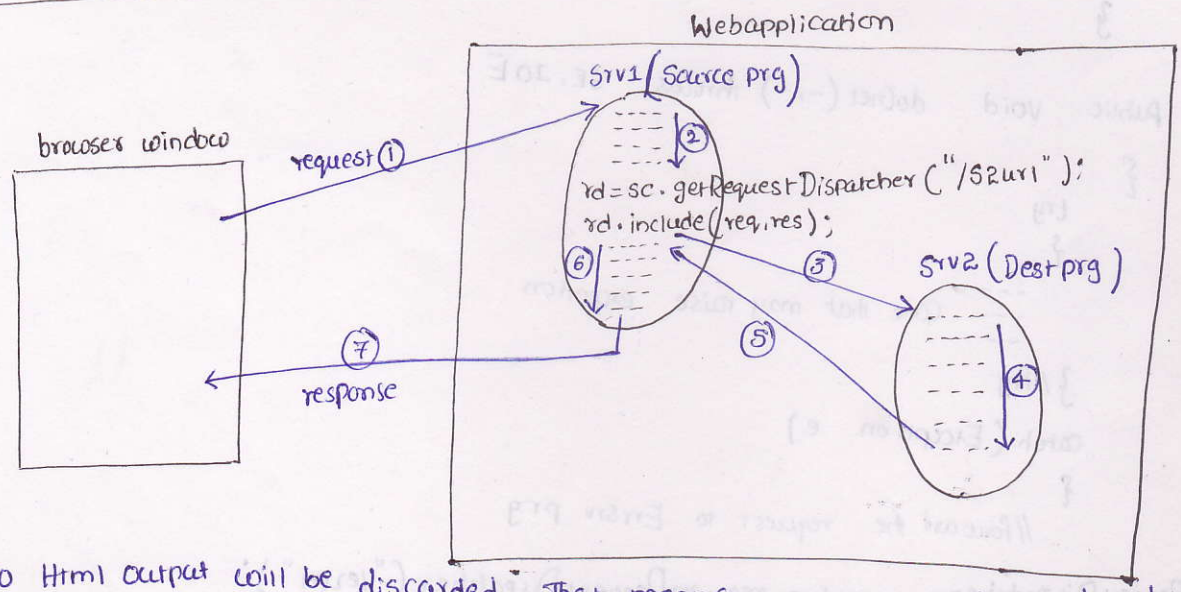


request uri :- http://localhost:2020/DBAPP/input.html

→ we always place rd.forward(-,-) method call in source servlet program as a conditional statement to execute.

01/11/2012

Understanding rd.include(-,-) :-



Here no Html output will be discarded. That means response contains the html output of both Srv1 program and Srv2 program.

(i) performs including response mode of Servlet chaining. That means includes the o/p of destination program to the output of Source program.

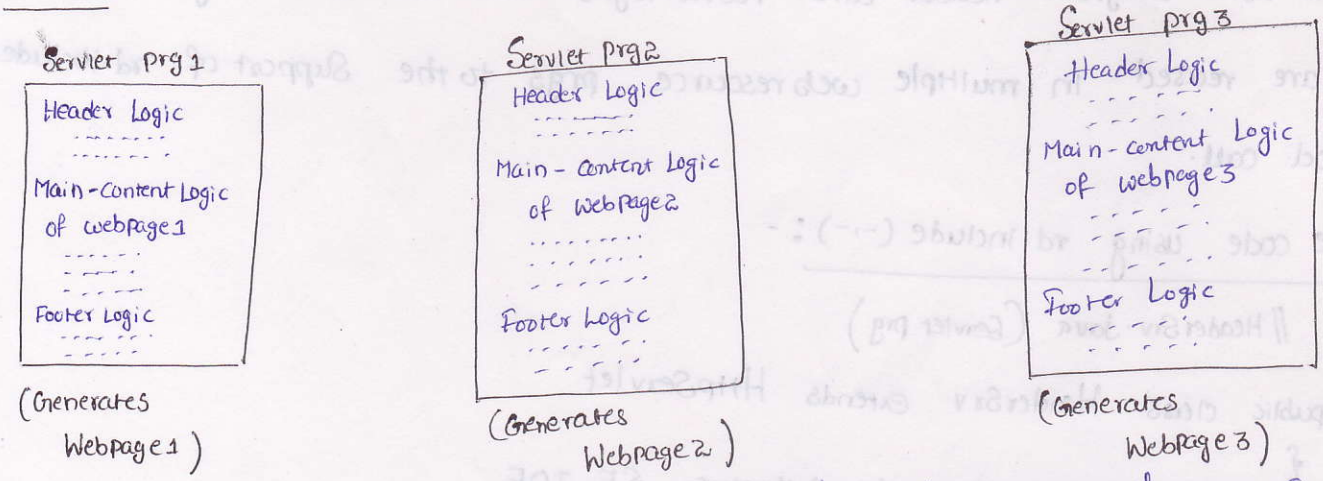
(ii) Same as rd.forward(-,-) $(iv)^{th}$ point

(iii) Response generated by Source program contains the Html output of both source, Destination programs.

(iv) Same as rd.forward(-,-) $(v)^{th}$ point, $(vi)^{th}$ point are same as rd.forward(-,-) $(vi)^{th}$ & $(vii)^{th}$ points

This rd.include(-,-) is very useful to make certain common logics like header, footer logics as reusable logics in multiple web resource programs of web application by including their output dynamically.

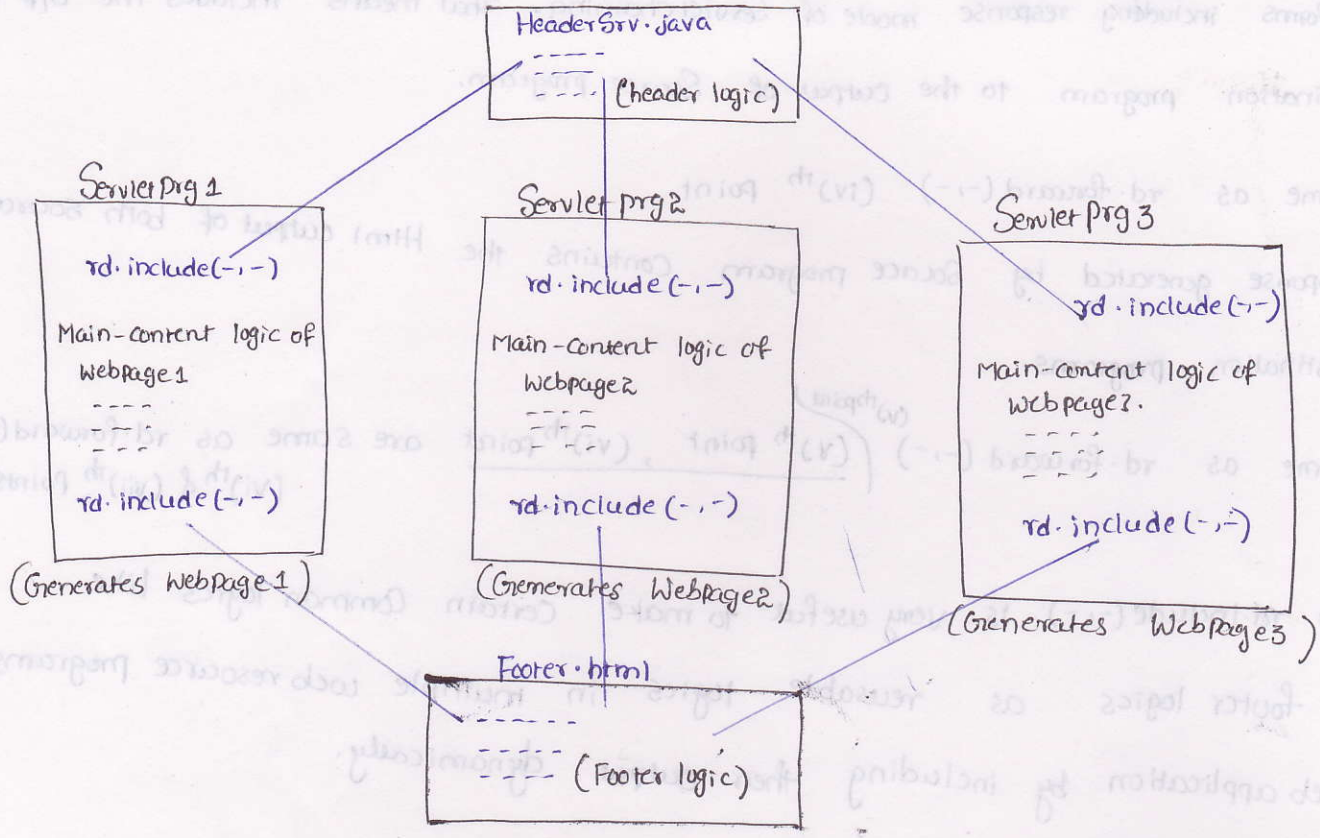
Problem:



Here header and footer logics are not reusable logics. because even they are same in all web resource prgs they are placed in every web resource prg.

Multiple webpages of webapplication contains same header and footer content but their main content differs.

Solution:-



To the above diagram header and Footer logics are developed only once. but they are reused in multiple web resource prgs to the support of rd.include(-,-) method call.

Sample code using rd.include(-,-):-

```
// HeaderSrv.java (Servlet prg)
public class HeaderSrv extends HttpServlet
{
    public void doGet(-,-) throws SE, IOE
    {
        // General Settings
        PrintWriter pw = res.getWriter();
        res.contentType("text/html");

        // prepare the header Content
        pw.println("<br><b><font color='red' size='6'><marquee>SATYA
                    </font></marquee></b>");

        pw.println("<br><br><br>");

        // donot close stream
        // pw.close();
    }
    public void doPost(-,-) throws SE,IOE
    {
        doGet(req,res);
    }//doPost(-,-)
}// class
```

Configure HeadSrv prg in web.xml file with "/headurl" url pattern

NOTE:- While working with rd.include(-,-) you must not commit the response by calling pw.close() in destination program. Because stops adding further response. But we can do this work while working with rd.forward(-,-) method.

Footer.html (html prg)

```
<br> <br> <br>
<hr>
<center> <b> &copy; all copy rights reserved for sathya students 2011-2012 </b>
</center>
```

Main Servlet prg

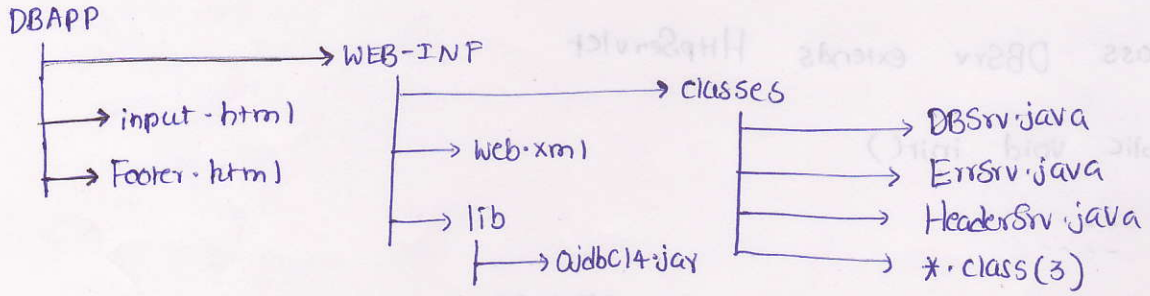
DBSrv.java (Servlet prg of DBApp)

```
public class DBSrv extends HttpServlet
{
    public void init()
    {
      ----
      ----
    }
    public void doGet(-,-) throws SE, IOE
    {
      try
      {
        // includes the header content
RequestDispatcher rd1 = req.getRequestDispatcher("/headurl");
  rd1.include(req, res);
        // logic to generate main-content
        ------
        ------
        ------
        ------
        ------
        // include the footer content
RequestDispatcher rd2 = req.getRequestDispatcher("/Footer.html");
  rd2.include(req, res);
      }//try
      catch(Exception e)
      {
RequestDispatcher rd = req.getRequestDispatcher("/errurl");
  rd.forward(req, res);
      }
    }//doGet(-,-)
```

```
public void doPost(-,-) throws SE,IOE
{
    doGet(-,-);
}// doPost(-,-)

public void destroy()
{
    ---
    ---
    ---
}//destroy
}// class
```

for the above Sample Based code complete example application refer App (6) of page no's 67 to 70

Deployement Directory Structure of DBAPP



input.html (formpage) footer.html (for footer logic) web.xml (DD file) DBSrv.java (mainServlet prg)

739 – 741 for including header content
760 – 764
771
825 Error Servlet

HeaderSrv.java (contains Header Logic)

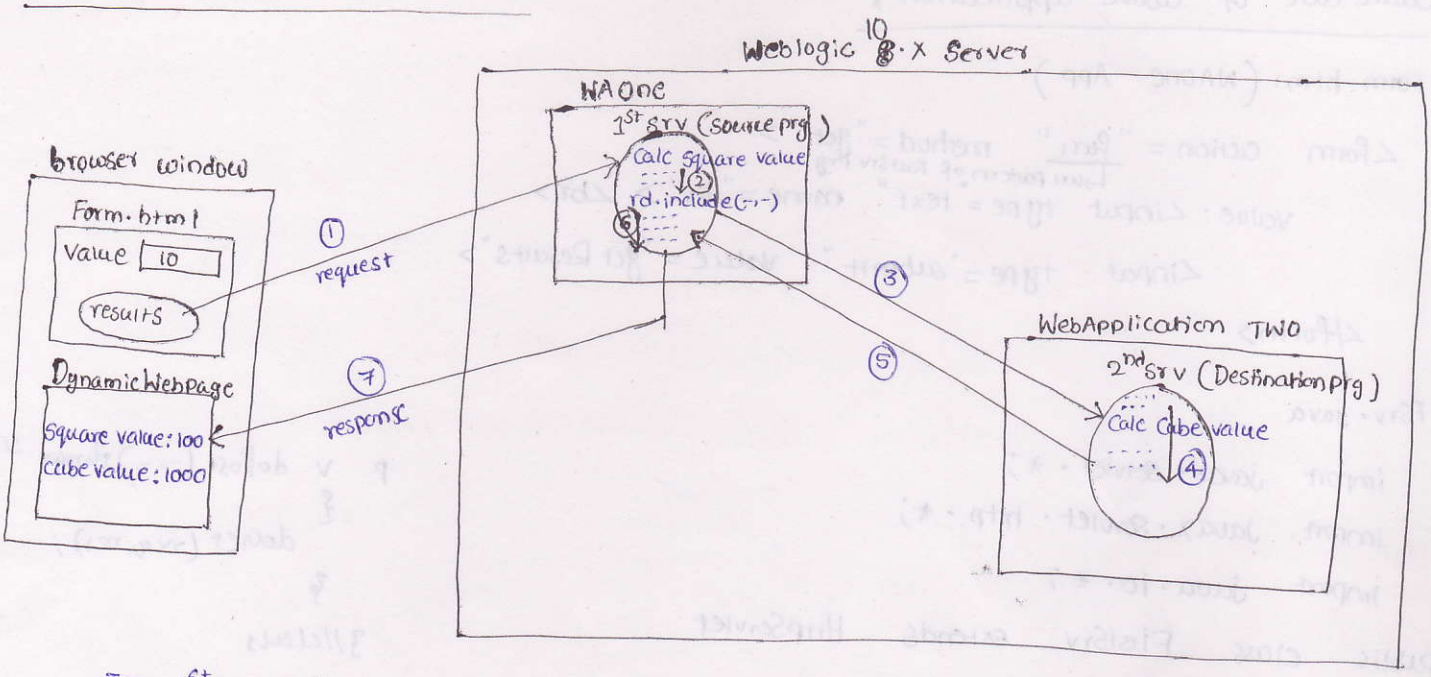
→ If rd.forward(-,-) is executed in Source Servlet program then the effect of rd.include(-,-) will not be there because rd.forward(-,-) not only discards the html output of Source Servlet program, It also discards the ~~html~~ included html output given by rd.include(-,-).

02/11/2012

→ while working with rd.include(-,-) don't commit the response by calling pw.close() method in Destination webresource program. Similarly don't Commit the response by calling pw.close() method in Source Servlet program before calling rd.forward(-,-) method (or) rd.include(-,-) method.

for 2nd Example application on Servlet chaining by using rd.forward(-,-) and rd.include(-,-) refer application (8) of the booklet.

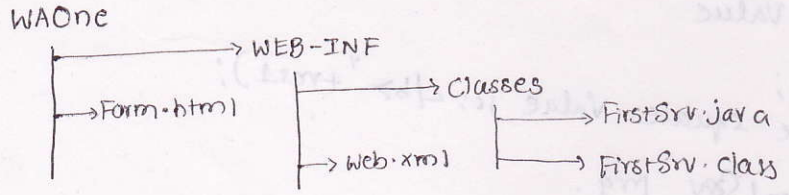
Performing ServletChaining b/w two servlet programs of two different webapplications residing in the same Server:-

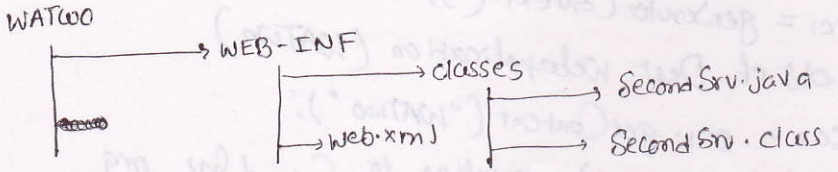


In 1st srv prg (source prg) RequestDispatcher obj must be created by using ServletContext object. Because the source and Destination programs are there two different webapplications of Same Server.

→ The above diagram based servletchaining is possible ~~only~~ in weblogic server (jdk 1.6) but not possible in other servers. Because getContext(-) of method is not implemented in these servers.

Deployement Directory Structure :-





~~Create TestDomain1 in weblogic 8.x and deployee both webapplications in that domain server~~

~~copy WAOne, WATwo folders to <weblogic8.x home>/user_projects/domains/TestDomain1/applications folder~~

~~Weblogic 8.x is compatible with Jdk1.4. So, activate Jdk1.4 (given by weblogic 8.x) before compiling the above Servlet programs.~~

~~Cmd prompt > set path = D:\BEA8.x\jdk1.4.1_05\bin~~
↳ weblogic8.x home

Source code of above application :-

```
// Form.html (WAOne App)
<form action = "furi" method = "get">
        ↳ uri pattern of FirstSrv prg
    Value : <input type = text" name = "t1" > <br>
        <input type = "submit" value = "get Results">
<</form>

// FSrv.java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class FirstSrv extends HttpServlet
{
    Public void doGet(HSR req, HSR res) throws SE,IOE
    {
        S.o.p(" doGet(-,-) of FirstSrv ");
        //general settings
        res.setContentType("text/html");
        PrintWriter pw = res.getWriter();
        //read form data
        int no = Integer.parseInt(req.getParameter("t1"));
        // find out the square value
        int res1 = no * no;
        pw.println("<b> The square value is: <b> " +res1);
    // Include the response of Second Srv prg.
        // get ServletContext obj current webapplication (WAOne)
        ServletContext sc1 = getServletContext();
        // get ServletContext obj of Dest webapplication (WATwo)
        ServletContext sc2 = sc1.getContext("WATwo");
        // create RequestDispatcher obj (rd) pointing to SecondSrv prg
    RequestDispatcher rd = sc2.getRequestDispatcher("/sum");
        rd.include(req,res);
    }// doGet(-,-);
    p v doPost(-,-) throws SE,IOE
    {
        doGet(req,res);
    }
}//class
```

// web.xml (WAOne App)

Configure FirstSrv program having "/furl" url pattern.

// SecondSrv.java (WATwo App)

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

public class SecondSrv extends HttpServlet
{
  public void doGet(HttpServletRequest req, HttpServletResponse res) throws SE, IOE
  {
      S.O.P("doGet(--) of SecondSrv");
        // general settings
        res.setContentType("text/html");
        PrintWriter pw = res.getWriter();

        // read form data
int no = Integer.parseInt(req.getParameter("t1"));

        // findout the cube value
        int res2 = no * no * no;
      pw.println("<br> <b> The cube value is " + res2);

        // dont close the stream
  }// doGet(--)

public void doPost(HSR req, HSR res) throws SE, IOE
  {
      doGet(req, res);
  }// doPost(--)
}// class
```

// web.xml (WATwo App)

Configure SecondSrv program having "/surl" url pattern.

→ Copy WAOne, WATwo folders to <Oracle Weblogic_home>\user_project\Domains\AdvBatchDomain\autodeploye folder

→ Start AdvBatchDomain Server.

→ Request url: http://localhost:7070/WAOne/Form.html

→ The above code will not run in Tomcat, GlassFish, Jboss servers. Because the method getContext(-) is not implemented in this servers.

03/11/2012

Limitations with RequestDispatcher obj :-

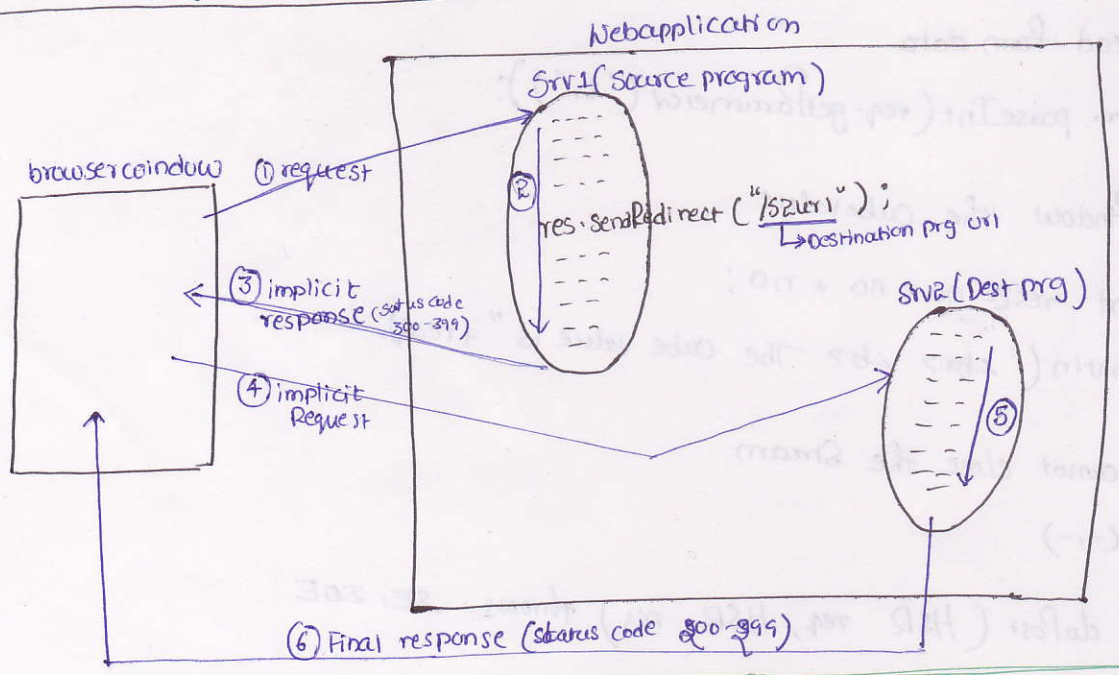
① → The Source Servlet program and Destination program of Servlet chaining can't be there in two different webapplications of two different servers

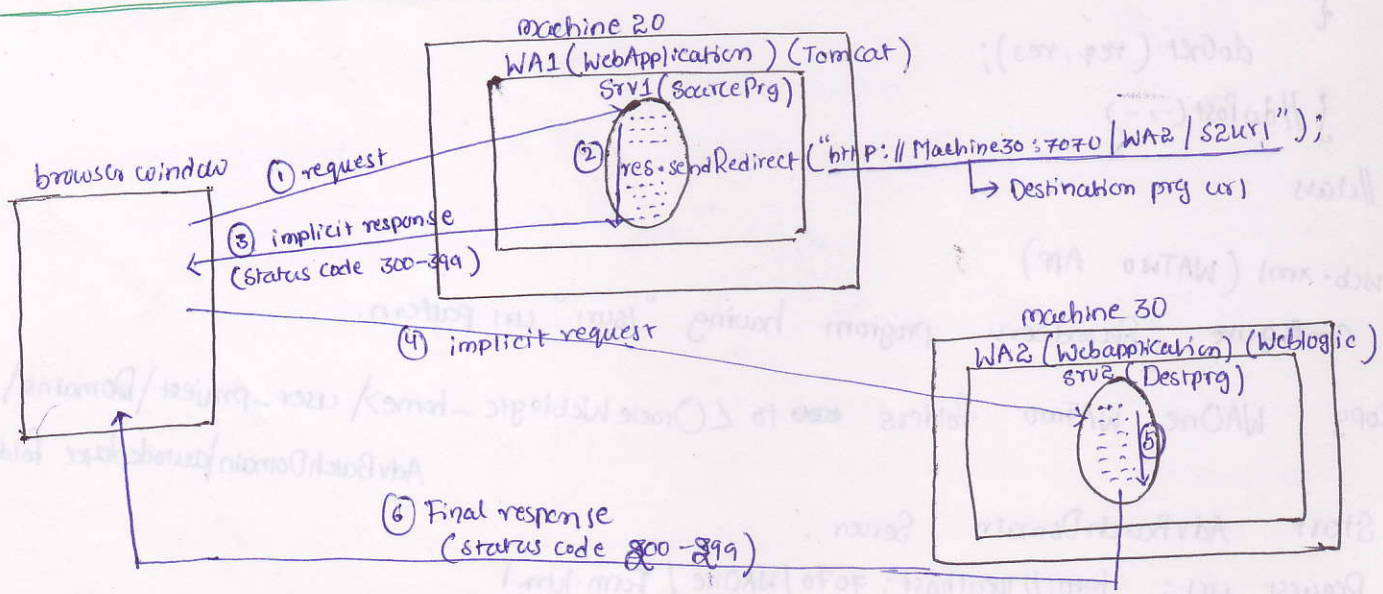
② → Source Servlet program and Destination program can be there in two different Webapplications of same server. But this communication works only in Weblogic server.

③ → The Destination program can't be non-java program like php prg (or) ASP.net prg and etc.....

→ Overcome the above problems and to perform Servletchaining without using RequestDispatcher object use SendRedirection concept support of res.sendRedirect(-)

Understanding the SendRedirection ~~method~~ Consept :-





W.r.to the Diagram

① → Client gives request to Source program Srv1

② → All the stmts in Srv1 program executes including res.sendRedirect(-) method.

Because of this the Html output generated by Srv1 program will be discarded

③ → Srv1 program sends implicit response to the browser window having destination program url collected from sendRedirect(-) method. The status code of this response is 300-399 asking browser window to perform Redirection.

④ → Browser window collects Destination program url from implicit response and generates implicit request to Destination Program (SrV2) (This is nothing but sendRedirection)

⑤ → All the stmts of Srv2 program executes.

⑥ → Srv2 program sends Final response to browser window. having only the html output of Srv2 program.
↳ status code is (200-299)

KeyPoints :-

ⓐ res.sendRedirect(-) method performs sendRedirection based servletchaining.

ⓑ both Srv1 and Srv2 prgs will not use same request & response objects. So, the Request data coming to Srv1 prgram are not Visible and accessible in Srv2 program.

ⓒ To pass additional data from Srv1 program to Srv2 program append query string to the url of res.sendRedirect(-)

Eg:- in srv1 prg

```
res.sendRedirect("/S2url?sno=101&sname=srikanth");
```

(or)

```
res.sendRedirect("http://localhost:7070/WA2/S2url?sno=101&sname=srikanth");
```

in srv2 prg

```
String s1 = req.getParameter("sno");
String s2 = req.getParameter("sname");
```

(d) Srv1 interacts with Srv2 by having one network round trip with browser window.

(e) Srv1 and Srv2 prgms can be there in same webapplication (or) can be there in two different webapplications of same (or) diff servers.

(f) When Srv1 and Srv2 resides in same webapplication then we can pass relative url (or) absolute url in res.sendRedirect(-) ~~otherwise we must pass~~

(g) Srv2 can be a ServletPrg or JspPrg (or) HtmlPrg (or) PHP prg (or) Asp.net prg and etc...

(h) In real time projects sendRedirection is useful when one company acquires (purchase) another company.

Eg scenario:-

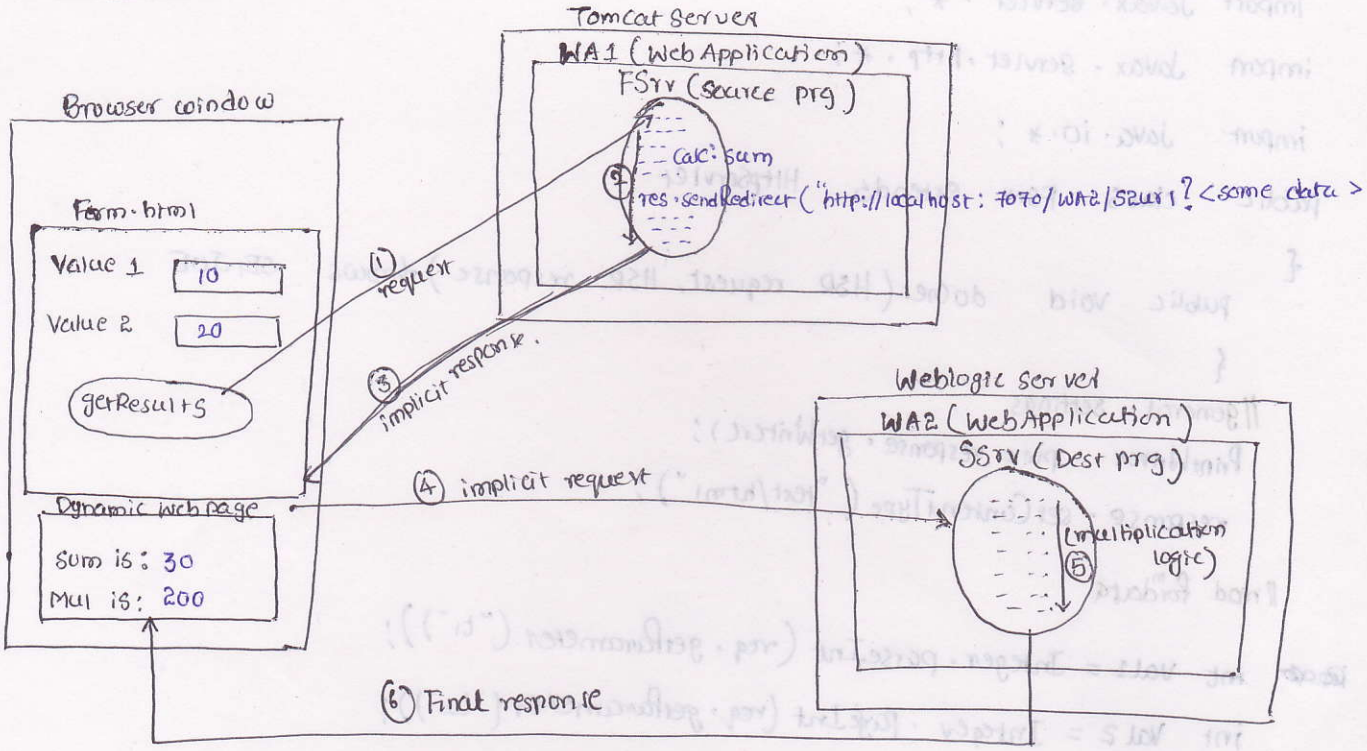
When Company B acquires Company A then a request coming from A.com should be redirected to B.com by using sendRedirection concept.

Eg:- (1) IBM company acquires Rational Company.
So, the request given to Rational.com will be Redirected to a program of IBM.com

(2) Oracle Corporation has acquired Sun Micro systems.
So, the request given to www.sun.com will be redirected to a WebResource program of www.oracle.com

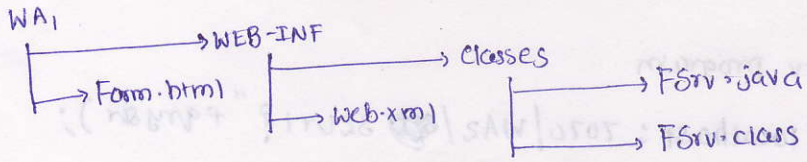
NOTE:- using sendRedirection source program can't include the response of Destination program. But source program can Redirect the request to Destination program.
↳ similar to forwarding request

Example application on sendRedirection :-

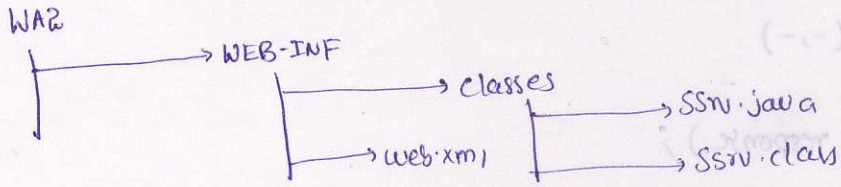


→ In the above diagram FSrv. sends Value1, Value2, Value3 sum data to SSrv program by appending query string to the url of res.sendRedirect(-)

Deployement Directory Structure :-



Deployee in Tomcat Server (copy WA1 folder to <Tomcat_home>\webapps\WA1



Deployee in AdvBatchDomain server of weblogic
(copy WA2 folder to <weblogic_home>\user_projects\domains\AdvBatchdomain\auto deploye folder

04/11/2012

Source code :-

```
//Form.html (WA1 webApp)
<form action="furl" method="get">
    Value1: <input type="text" name="t1"> <br>
    value2: <input type="text" name="t2"> <br>
    <input type="submit" value="GetResults">
</form
```

```
// FSrv.java (WA1 webapp)
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class FSrv extends HttpServlet
{
    public void doGet(HSR request, HSR response) throws SE, IOE
    {
    //general settings
    PrintWriter pw = response.getWriter();
    response.setContentType("text/html");
    //read formdata
    int val1 = Integer.parseInt(req.getParameter("t1"));
    int val2 = Integer.parseInt(req.getParameter("t2"));
    int result = val1+val2;
    pw.println("FSrv: sum is: "+result);
    // Frame query string having formdata and some result,
    String qrystr = "P1" = "+val1+" &P2="+val2+"&P3="+result;
    //redirect the request to SSrv program
    res.sendRedirect("http://localhost:7070/WA2/SSURL?"+qrystr);
    s.o.p("FSrv: in doGet(-,-)");
    } //doGet(-,-);
    public void doPost(-,-)
    {
        doGet(request, response);
    }
}
//web.xml (WA1 webapp)
Configure FSrv with "/FURL" url pattern.

//SSrv.java (WA2 webApp)
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class SSrv extends HttpServlet
{
```

```
public void doGet(--)
  // general settings
  PrintWriter pw = res.getWriter();
  response.setContentType("text/html");
  //read data given by Fsrv prg
int Val1 = Integer.parseInt(request.getParameter("P1"));
int Val2 = Integer.parseInt(request.getParameter("P2"));
int Sum = Integer.parseInt(request.getParameter("P3"));
  //cal multiplication value.
   int mul = Val1 * Val2;
  //display both results
  pw.println("<br>Ssrv : sum is " +sum);
  pw.println("<br>ssrv : mul is " +mul);
   S.o.p("ssrv : doGet(--) method");
}
  public void doPost(--)
  {
    doGet(request, response);
  }
}
```

//web.xml (WA2 App)

Configure SSrv with "/S2url" url pattern.

Start tomcat server, Advbatch domain server of weblogic

request url: http://localhost:2020/WA1/Form.html

→ In servlet chaining we can forward / redirect the request to local (or) remote destination program from source program but we can't include response of remote destination program. (we can include the response of local destination program)

Q:- What is the difference b/w rd.forward(-,-) and res.sendRedirect(-,-)

rd.forward(-,-)	res.sendRedirect(-,-)
① Performs forward request from of servlet chaining	① performs sendRedirection mode of servlet chaining
② Both source and destination programs use same request and response objects	② will not use
③ Source program talks with destination prgm directly	③ source prgm talks with destination prgm after having network round trip with browser window.
④ To pass additional data from source prgm to destination program use request attributes	④ To pass additional data from source program to destination prgm append query string to the url of res.sendRedirect(-,-) method.
⑤ Both source and destination programs can be their same webapplication (or) in two different webapplications of same server	⑤ Both source and destination programs can be their in same webapplication and can be there in two different webapplications same (or) different servers.
⑥ Destination prgm can be html prgm (or) servlet prgm (or) jsp program. does not allow asp.net, PHP programs as destination programs	⑥ Destination program can be any program including Asp.net, PHP programs.
⑦ During forwarding request operation url in the browser address bar will not be change	⑦ will be change
⑧ recommend to use when destination program be local to a source program	⑧ recomended to use when destination program is remote to source program.

What is the difference b/w rd.forward(-,-) and rd.include(-,-)

rd.forward(-,-)	rd.include(-,-)
① performs forwarding request mode of servlet chaining	① performs including response mode of servlet chaining
② Html output of ~~source~~ source prgrm will be discarded. so the only html output of destination program goes to browser window as response	② The html outputs of both source and destination programs together goto browser window.
③ useful for error servlet configuration	③ use ful to include the outputs from common logics. (like header, footer logics)

Q:- What happents if we place both rd.forward(-,-) and res.sendRedirect(-,-) in one servlet program

Ans:- java.lang.illegalStateException will be thrown

Q:- what happents if we place both rd.forward(-,-) and rd.include(-,-) in one servlet program.

Ans:- The effect of rd.include(-,-) will not be there only the effect of rd.forward(-,-) takes place.

Q:- what happents if we call multiple rd.forward(-,-) methods pointing to multiple destination programs in one source program.

Ans:- In tomcat server Exception will be thrown illegal state Exception.
In GlassFish server the rd.forward(-,-) method call effect takes place.

NOTE:-

① The effect of send redirection will be there if we place both rd.include(-,-) and res.sendRedirect(-) method in one servlet program.

② if we place multiple rd.include(-,-) methods in one servlet program all effects will be effected.

③ If you place multiple res.sendRedirect(-) method calls in one servlet program then java.lang.illegalStateException will be raised.

→ The following are the urls in different search engines to perform search operations.

Google :- http://www.google.co.in/search?q=seven+wonders+of+india

(web resource name) — request param name — request param value

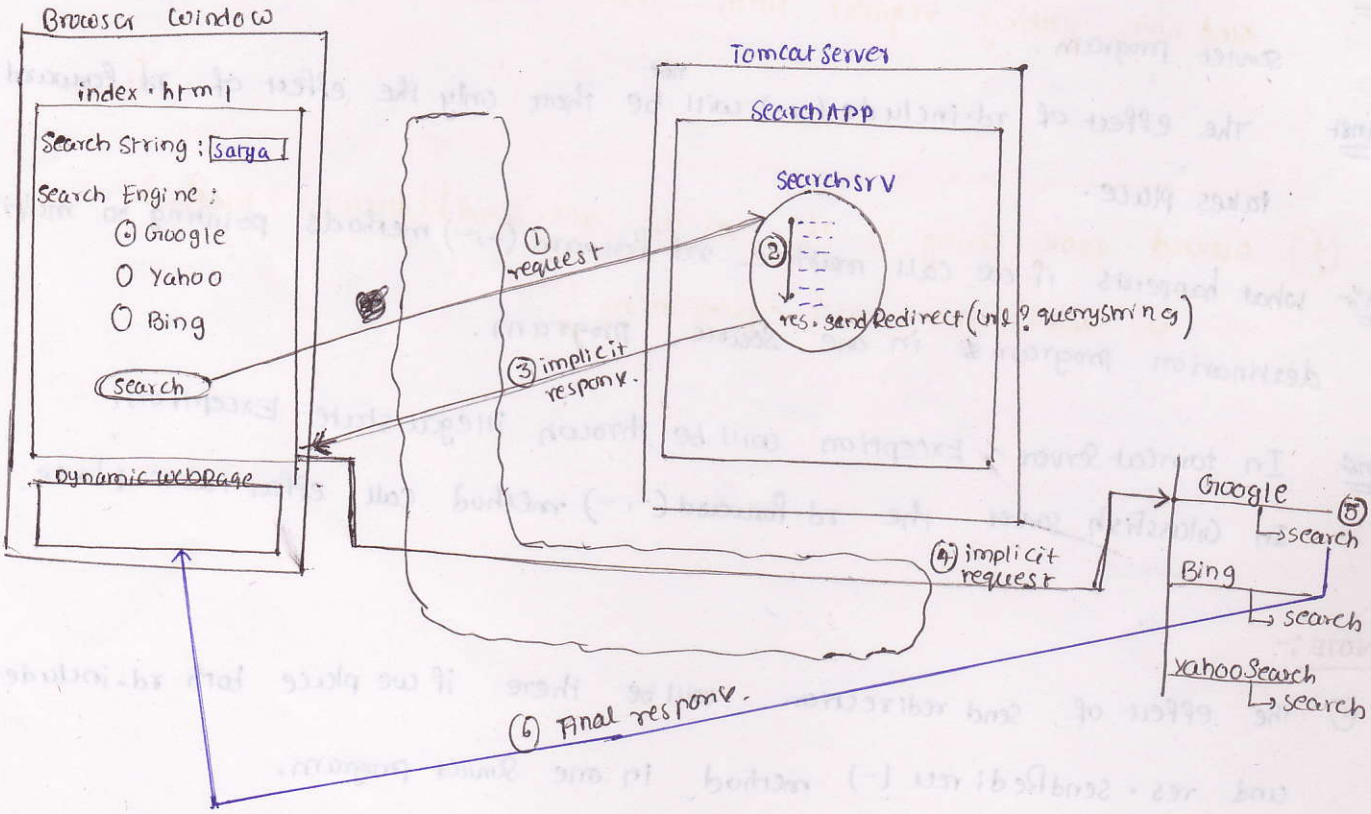
Bing :- http://www.bing.com/search?q=new+seven+wonders

Yahoo search :-

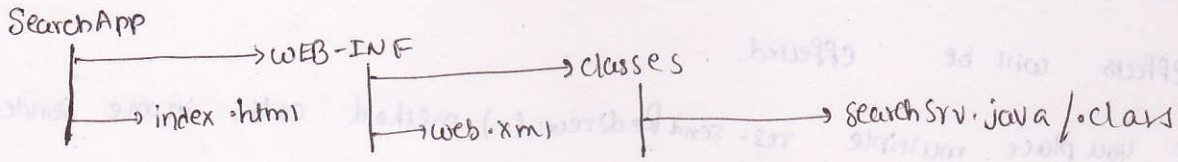
http://in.search.yahoo.com/search?p=new+seven+wonders.

(*) Example application on send Redirection to make our webapplication as hub to use multiple search engines :

NOTE :- In this application our local servlet program interacts with multiple remote unknown technologies based webresource programs of different search engines. for this ~~use~~ we must use send Redirection



Deployement Directory structure :-



request url :

http://localhost:2020/searchApp/index.html

for above diagram based sourced code refer application (9) of page no's (73) & (74)

Q:- How to pass data b/w webresource programs ?

Ans: when source program and dest program resides in the same webapplication

(1). request attributes :

(use when source prg and dest prg uses same req and response)

(2) session attributes :

(use when source prg and dest prg gets request from same browser window)

(3) Servlet Context / Application attributes :

(uses when source prg and dest prgs are not using same req, res objects and not getting request from same browser window).

When source prg and dest prg reside in two diff webapplication :

(4) append query string to the url of res.sendRedirect(-) method

res.sendRedirect(url? <queryString)

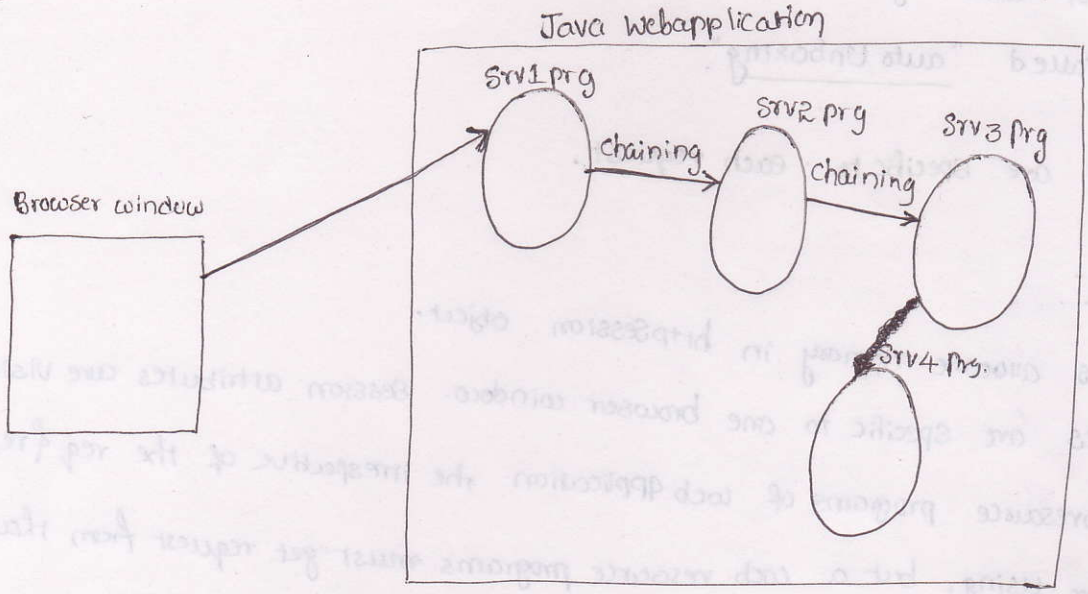
NOTE:- attribute is a name that can had a value. this attributes is no way related with Xml, Html tags attributes.

Request Attributes :-

→ allocates memory in request obj

→ visible throughout the request cycle of current request.

→ request attributes are visible in multiple ~~web~~resource prgs who participates in single servlet chaining



to create request attribute

```
req.setAttribute("name", "raja");
req.setAttribute("age", 30);
```

attribute name must be string and value must be an object.

→ here we can pass simple values as attribute values because they will be converted into objects automatically through "auto boxing".

to modify req attribute values :-

```
req.setAttribute("name", "ramesh");
req.setAttribute("age", 30);
```

↳ will be converted into integer wrapper class obj through autoboxing

NOTE:- setAttribute(-,-) creates the new attribute if the attribute is not already available. Otherwise modifies the existing attribute value.

to read req attribute values :-

```
String name = (String) req.getAttribute("name");
int age = (Integer) req.getAttribute("age");
```

→ Autounboxing Concept

here autounboxing feature is taken.

or

```
Integer o1 = (Integer) req.getAttribute("age");
```

int age = O1.intValue();

To remove request attributes :-

req.removeAttribute("name");
req.removeAttribute("age");

NOTE:- The process of converting simple values to wrapper class objects is called "auto Boxing" automatically.

and reverse is called "auto Unboxing".

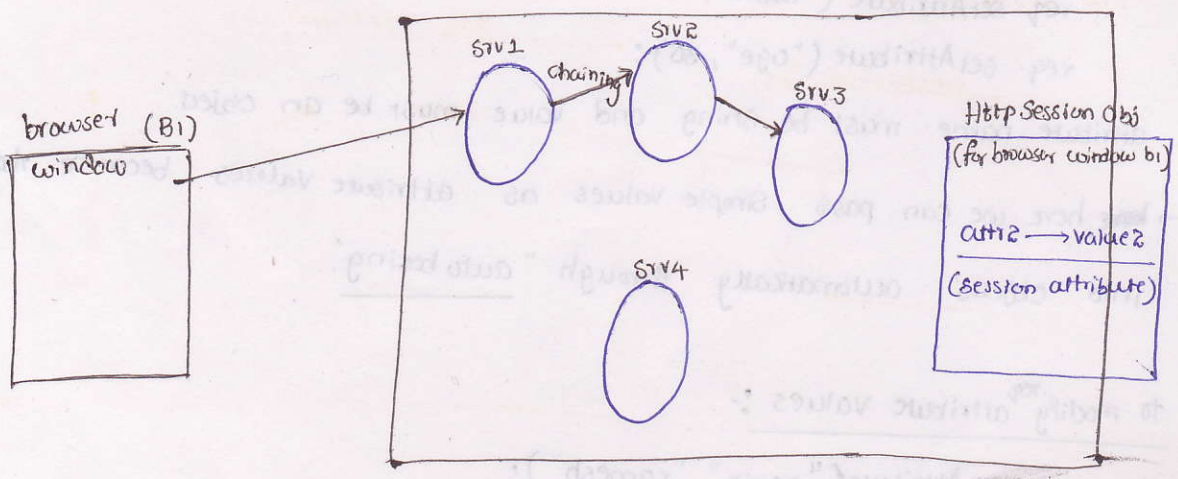
→ request attributes are specific to each request.

Session attributes :-

Session attributes allocate memory in HttpSession object.

→ Session attributes are specific to one browser window. Session attributes are visible in all the webresource programs of web application the irrespective of the req/res objects they are using. but a web resource programs must get request from that browser window for which session attributes are created.

→



Session attributes are global but they are specific to a browser window.

The session attribute created in Srv1 program by getting request from browser window b1 is visible and accessible in all the web resource programs of web application but they must get the request from the same browser window b1. Otherwise not visible.

to create session attributes :-

```
HttpSession ses = req.getSession();  // gives session obj
ses.setAttribute("attr2", "value2");
```

ses.setAttribute("age", 30);

NOTE:- Session object allocates memory in the Server but it specific to one client (browser window)

so, the session attributes that resides in Session obj are also specific to browser window

to modify session attribute values :-

ses.setAttribute("attr2", "val2");
ses.setAttribute("age", 35);

to read session attribute values :-

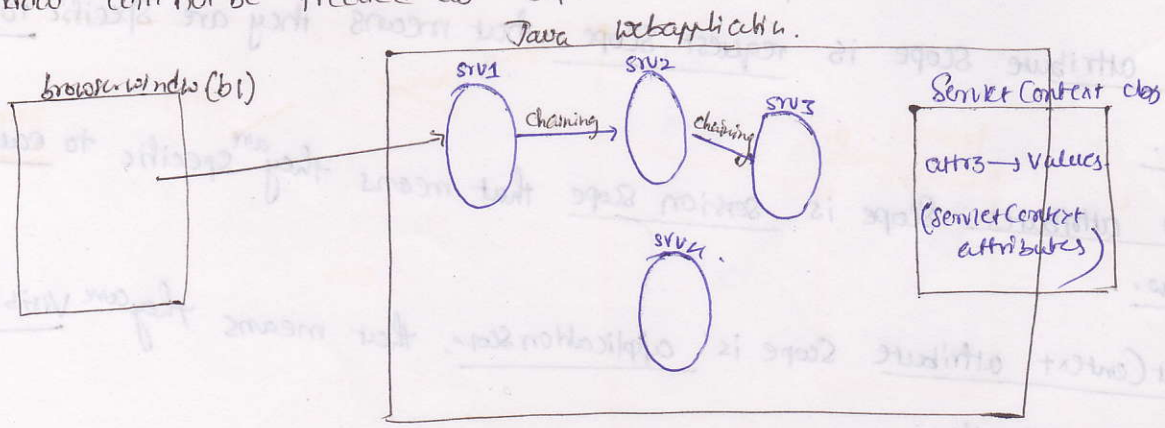
String val = (String) ses.getAttribute("attr2");
int age = (Integer) ses.getAttribute("age");

to remove session attribute values :-

ses.removeAttribute("attr2");
ses.removeAttribute("age");

→ tab window will not be treated as separate browser window as client.

→



ServletContext attributes :- this attributes allocate memory in ServletContext object

→ these attributes are global attributes of a Web application.
→ these attributes are visible in all the web resource programs of web application.
→ irrespective of the request, response obj they are using and irrespective of a browser window from which they are getting request.
→ these attributes are condition less global attributes of a webapplication.
→ ServletContext attribute created in Srv1 program is visible and accessible in all other programs irrespective of any condition.

to create ServletContext attribute :-

```
ServletContext sc = getServletContext();  // gives ServletContext obj
sc.setAttribute("attr3", "val3");
sc.setAttribute("age", 60);
```

to modify ServletContext attribute Values :-

```
sc.setAttribute("attr3", "val33");
sc.setAttribute("age", 45);
```

To read ServletContext attribute Values :-

```
String val = (String) sc.getAttribute("attr3");
int age = (Integer) sc.getAttribute("age");
```

To remove ServletContext attributes :-

```
sc.removeAttribute("attr3");
sc.removeAttribute("age");
```

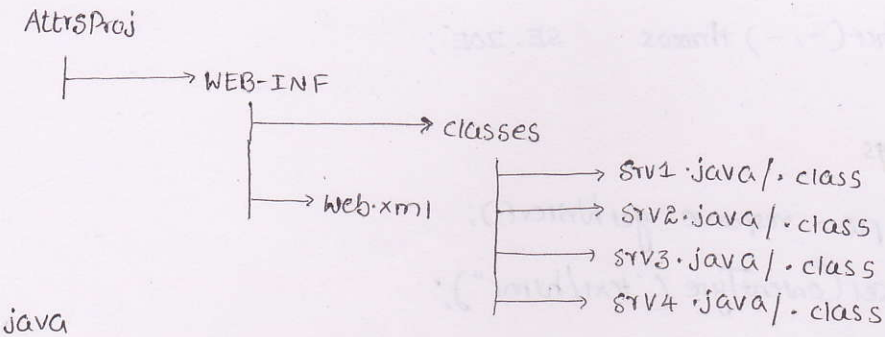
→ request attribute scope is request scope that means they are specific to each request.

→ session attributes scope is session scope that means they are specific to each browser window.

→ ServletContext attribute scope is application scope that means they are visible throughout the webapplication

~~Session Tracking~~ 07/11/2012

Example application to demonstrate the behaviour of different attributes



Using NetBeans IDE

```
// Srv1.java

public class Srv1 extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response) throws
                                                                                SE, IOE
    {
    // Create request attribute
      request.setAttribute("attr1", "value1");
    // Create Session attribute
      HttpSession ses = request.getSession();
      ses.setAttribute("attr2", "value2");
    // Create ServletContext attribute
      ServletContext sc = getServletContext();
      sc.setAttribute("attr3", "value3");

    // forward the request to Srv2 prg
    RequestDispatcher rd = request.getRequestDispatcher("/Srv2");
      rd.forward(request, response);
    }

    public void doPost(HttpServletRequest request, HttpServletResponse response) throws SE,IOE
    {
      doGet(request, response);
    }
}//class
```

```
// srv2.java

public class srv2 extends HttpServlet
{
  protected void doGet(-,-) throws SE, IOE;
  {
    // general settings
      PrintWriter pw = response.getWriter();
        response.setContentTyre("text/html");

    //read request attribute value
pw.println("srv2: attr1(req) attribute value" +request.getAttribute("attr1"));
    //read session attribute value
  HttpSession ses = request.getSession();
pw.println("<br> srv2: attr2(ses) attribute value:" +ses.getAttribute("attr2"));
    //read ServletContext attribute value
    ServletContext sc = getServletContext();
pw.println("<br> srv2: attr3(sc) attribute value:" +sc.getAttribute("attr3"));

    // forward the request to srv3 prg
RequestDispatcher rd = request.getRequestDispatcher("/srv3");
    rd.forward(request, response);
  }
protected void doPost(HSR Request, HSR response) throw SE, IOE
    {
      doGet(request, response);
    }
  }
```

//srv3.java

Same as srv2.java but don't forward the request to another Servlet program.

// srv4.java

Same as srv2.java but don't forward the request to any other Servlet prg.

// web.xml

Cfg srv1 prg with /srv1 url pattern
Cfg srv2 prg with /srv2 url pattern
Cfg srv3 prg with /srv3 url pattern
Cfg srv4 prg with /srv4 url pattern

request urls :-

http://localhost:2020/AttrsProj/srv1
http://localhost:2020/AttrsProj/srv2
http://localhost:2020/AttrsProj/srv3
http://localhost:2020/AttrsProj/srv4

Session Tracking :-

There are 2 Types of form pages.

① Static form page ② Dynamic form page

for related information on these static and Dynamic formpages refer page No's: 1 to of 07/11/2012 handout

Q:- Why web applications are stateless by default?

Ans:- refer page No: ④ & ⑤ of 07/11/2012

Q:- What is the diff b/w Session Management and state management?

Ans:- refer page No: 8 of 07/11/2012

→ To make web applications as stateful webapplication we need one of the following SessionTracking Techniques.

1) Hidden Form fields
2) Http cookies
3) Http Session with cookies
4) HttpSession with URLRewriting.

① Hidden Form Fields (Hidden Box) :-

In Form Page :-

```
<input type="hidden" name="t1" value="hello">
```

In Servlet Prg

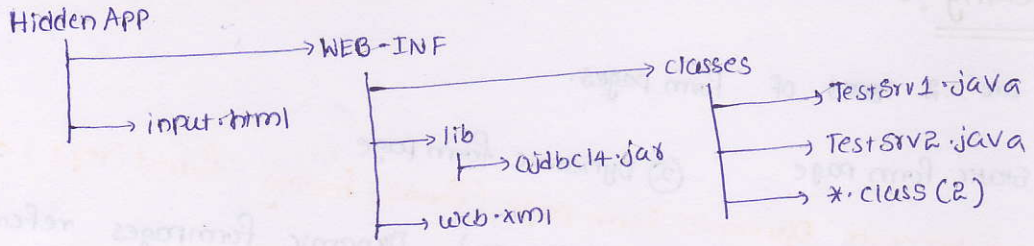
```
String s1 = req.getParameter("t1"); // gives hello
```

→ Hidden Box is an invisible TextBox of form page.

for Fundamentals of HiddenFormFields Technique refer page no's: ⑤ - ⑧ of the 07/11/2012 handout.

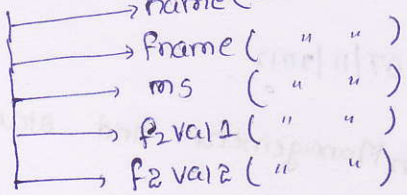
→ Source code of Example Application that uses HiddenForm fields sessionTracking technique according to the diagram of page No: ⑥.

Deployement Directory Structure :-



Request url: http://localhost:2020/HiddenApp/input.html

DB table in oracle : person_tab




```
SQL> Create table person_tab (name varchar2(20), fname varchar2(20), ms varchar2(10),
                              f2val1 varchar2(20), f2val2 varchar2(20));

Table created.
```

```
// input.html
<form action="testurl1" method="get">
    ↳ url pattern of TestSrv1 prg.
Name : <input type="text" name="tname"> <br>
Father's Name : <input type="text" name="tfname"> <br>
Marital Status : <input type="CheckBox" name="tms" value="Married">
                                                  married <br>
```

```
<input type="submit" value="continue">
</form>

// TestSrv1.java
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;

public class TestSrv1 extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws SE,IOE
    {
        //general settings
        PrintWriter pw = res.getWriter();
        res.setContentType("text/html");

        //read form1/req1 data
        String name = req.getParameter("tname");
        String fname = req.getParameter("tfname");
        String ms = req.getParmaters("tms");
        //Generate Dynamic Form page (form2) having form1(req1) data
        //as hidden box values
        if(ms==null)
            ms="single";

        if(ms.equals("Married"))
        {
            pw.println("<form action='testsrv2'>");
            pw.println("spouse name: <input type='text' name='st1'><br>");
            pw.println("No.of children: <input type='text' name='st2'><br>");
            //keep form 1 in form2 as hidden box values
            pw.println("<input type='hidden' name='Hname' value="+name+">");
            pw.println("<input type='hidden' name='Hfname' value="+fname+">");
            pw.println("<input type='hidden' name='Hms' value="+ms+">");
            pw.println("<input type='submit' value='submit'>");
            pw.println("</form>");
        }//if
```

(*1)

```
else
{
    pw.println("<form action='testsrv2'>");
    pw.println("When do u want to marry: <input type='text' name='st1'><br>");
    pw.println("why do u want to marry: <input type='text' name='st2'><br>");
    // keep form1 in form2 as hidden box values
Same as (*1) {
    - - - - -
    - - - - -
    - - - - -
    - - - - -
}//else
    pw.close();
}//doGet
```

```
public void doPost(HSR req, HSR res) throws SE,IOE
{
    doGet(req, res);
}
}//class
```

```
// TestSrv2.java

import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.sql.*;

public class TestSrv2 extends HttpServlet
{
public void doGet(HSR req, HSR res) throws SE,IOE
{
    // general Settings
    PrintWriter pw = res.getWriter();
    res.setContentType("text/html");

    // read form1/req1 and form2/req2 data
    String name = req.getParameter("Hname");
    String fname = req.getParameter("Hfname");
    String ms = req.getParameter("Hms");
    String f2v1 = req.getParameter("st1");
    String f2v2 = req.getParameter("st2");
try {
    // insert form1 and form2 data in DB table as record
Class.forName("Oracle.jdbc.driver.OracleDriver");
Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:orcl",
                                              "scott", "tiger");
PreparedStatement ps = con.prepareStatement("insert into Person_tab values
                                             (?,?,?,?,?)");
    ps.setString(1,name);
    ps.setString(2,fname);
    ps.setString(3,ms);
```

Hidden

Textbox names of forms

```
ps.setString(4, f2v1);
ps.setString(5, f2v2);
int result = ps.executeUpdate();
if (result == 0)
    pw.println("<br><b> Registrain Failed </b>");
else
    pw.println("<br><b> Registration Success </b>");

//Display dynamic webpage having Form1 and Form2 data

pw.println("<br>Form1 data is "+name+" "+fname+" "+ms);
pw.println("<br><b> Form2 data is "+f2v1+" "+f2v2);

//Close streams

pw.close();
ps.close();
con.close();
} catch (Exception e) { e.printStackTrace(); }
} //doGet(--)
public void doPost(HSR req, HSR res) throws SE, IOE
{
    doGet(req, res);
}
}
```

//web.xml

Configure TestSrv1 prg with /testurl1 and TestSrv2 prg with /testurl2 url pattern

Q:- can you explain some real world example scenarios where Session tracking is applied?

Ans:- Refer page NO (7) of November 7th 2012 handout.

09/11/2012

Http cookies :-

→ Cookies are small textual informations, which allocates memory at client side and remembers client data across the multiple request during a session.

→ Eventhough cookies are created at server side they come to client side for memory allocation along with the response and they go back to webapplication from client side along with the request.

→ There are two types of cookies.

① in memory cookies | per session cookies

② persistent cookies.

for Basics of Http cookies refer page no's ⑧ to ⑫.

→ for understanding the basics of cookies refer the application ⑫ of PageNO ⑦⑧ and ⑦⑨.

SetCookies.java // servlet program to create the cookie.

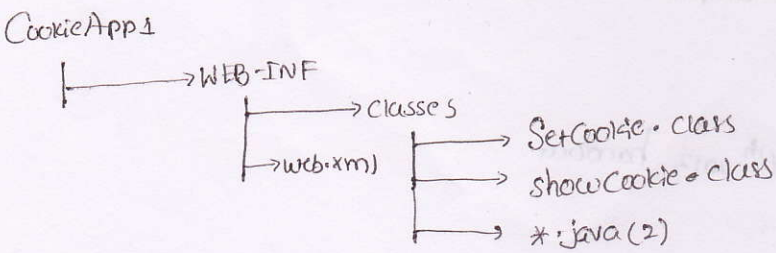
1407 - 1410 // two inmemory cookies are added to the response.

1415 - 1421 // two persistent cookies are added to the response.

1445 // reads all the cookies coming along with the request

1446 - 1451 // logic to display the received cookie name and value in the form of Html table row content.

Directory Structure :-



request urls:-

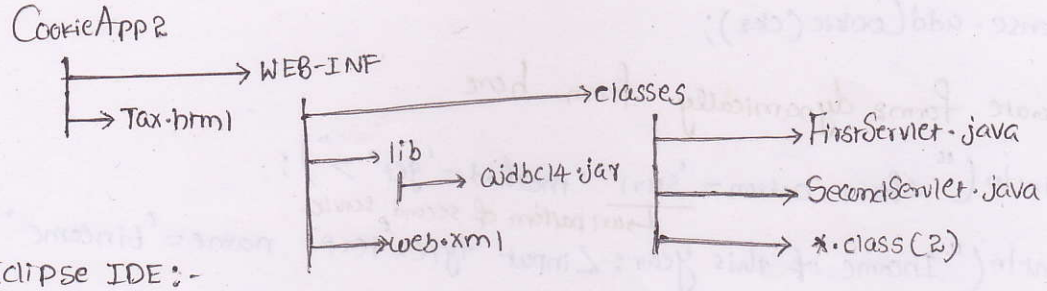
http://localhost:2020/CookieApp1/test1

/CookieApp2/test2

See to server

→ Cookies based Example application Demonstrating SessionTracking w.r.to Diagram of Page no: (9) of Nov 7th 2012 handout.

Deployement Directory structure :-



Using MyEclipse IDE :-

Step-I :- File → New → Project Name : [CookieApp2] → Finish

Step-II :- In the right side click on project CookieApp2 → webroot → new → html → [Tax.html]

Tax.html

```
<form action="/fcurl" method="get">
    Name : <input type="text" name="tname"/> <br/>
    Father Name: <input type="text" name="tfname"/> <br/>
    <input type="submit" value="Continue"/>
</form>
```

Step-III :- Expand Project → src → Servlet → Name : FirstServlet → Url pattern : fcurl

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

public class FirstServlet extends HttpServlet
{
public void doGet (HSR request, HSR response) throws SE, IOE
{
    //general settings
    PrintWriter pw = response.getWriter();
    response.setContentType("text/html");
    //read form1/request1 data
    String name = request.getParameter("tname");
    String fname = request.getParameter("tfname");
    //create in-memory cookies having form1/request1 data
```

```
Cookie ck1 = new Cookie("name", name);
Cookie ck2 = new Cookie("fname", fname);
    //add Cookies to response
    response.addCookie(ck1);
    response.addCookie(ck2);
    //Generate form2 dynamically from here
    pw.println("<form action='sur1' method='get'>");
                 ↳ url pattern of second servlet
    pw.println("Income of this year: <input type='text' name='tincome'>");
    pw.println("Tax: <input type='text' name='tax'>");
    pw.println("<input type='submit' value='submit'>");
    pw.println("</form>");
    //close streams
    pw.close();
}//doGet(--);
public void doPost(HSR request, HSR response) throws SE,IOE
{
    doGet(request, response);
}
}
```

Step-IV: ~~click~~ Expand cookieproject2 → src → servlet → name: secondServlet

→ url pattern: sur1

```
// SecondServlet.java
    import javax.servlet.*;
    import javax.servlet.http.*;
    import java.io.*;
    import java.sql.*;
public class SecondServlet extends HttpServlet {
p v doGet(HSR request, HSR response) throws SE,IOE {
        //general settings
PrintWriter pw = response.getWriter();
response.setContentType("text/html");
```

```
//read form2 / request2 data
int income = Integer.parseInt(request.getParameter("tincome"));
int tax = Integer.parseInt(request.getParameter("tax"));
//read form1/request1 data from Cookies
String name = null;
String fname = null;
Cookie ck[] = request.getCookies();
if(ck != null)
{
    name = ck[0].getValue();
    fname = ck[1].getValue();
}
```

→ Session tracking.

```
try {
//write Jdbc code insert form1 /form2 data in db table as recorded
    Class.forName("Oracle.jdbc.driver.OracleDriver");
    Connection con = DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521
                                                 :orcl", "scott", "tiger");
PreparedStatement ps = con.prepareStatement("insert into Tax_tab values(?,?,?,?)")
    ps.setString(1, name);
    ps.setString(2, fname);
    ps.setInt(3, income);
    ps.setInt(4, tax);
    int result = ps.executeUpdate();
    if(result == 0)
    {
        pw.println("Registration Failed");
    }
    else
    {
        pw.println("Registration Success");
    }
}//try
catch(Exception e)
{
    e.printStackTrace();
}
```

```
//Generate Dynamic ~~Settings~~ Webpage having Form1 and Form2 data.
    pw.println("<br> Form1 Data is : "+name+"....."+fname);
    pw.println("<br> Form2 Data is : "+income+"....."+tax);
//close stream
    pw.close();
}
public void doPost(HSR request, HSR response) throws SE,IOE
{
    doGet(request, response);
}
}
```

//web.xml

Configure FirstServlet program with /fur1 url pattern.
" SecondServlet program with /sur1 url pattern.

DB table in oracle

```
SQL> create table Tax_tab (name Varchar2(20), fname Varchar2(20), income number,
                           tax number);
Table created.
```

- Click on CookieApp2 ⟶ Run as ⟶ Tomcat 6.x
  - http://localhost:2020/cookieApp2/Tax.html

→ If we use Inmemory cookies then it is called SessionTracking

→ If we use Persistent cookies then it won't come under SessionTracking but it comes under State management. Because the persistent cookies remember client data even after session.

→ when browser window is closed then the session b/w browser window and client will be terminated automatically.

→ persistent cookies may bring virus to computer because they allocate memory on client machine filesystem.

③ HttpSession with cookies :-

→ This technique uses HttpSession Object and its attributes for session tracking

→ HttpSession Object and its attributes are globally visible in multiple webresources of web applications but they are specific to a browser window.

→ Every HttpSession object contains one SessionId, this sessionid goes to browser window and comes back to web application in the form of cookie. Due to this, this technique is called as "HttpSession with cookies technique."

for Basics of HttpSession with Cookies base Session Tracking refer page No's 13-20

④ HttpSession with URL rewriting :-

→ This technique is the extension of HttpSession with cookies. where cookies will not be used to send SessionId to browser window along with the response. and Cookies will not be used to bring the SessionId back to webapplication along with the request.

for detailed info about HttpSession with URL re-writing technique see page no 20 & 21 given handout 07/11/2012

Servlet Listeners (Event handling in web app'n's) 14/11/2012

*) Event is an action, performed on the Component or object. Executing some logic when Event is raised is called "Event handling." For this we need Event Listening.

To perform Event Handling we need the following details

1) Source object ⟶ Eg: AWT Button
2) Event class ⟶ Eg: ActionEvent
3) Event Listener (I) ⟶ Eg: ActionListener
4) Event Handling method ⟶ actionPerformed (ActionEvent ae)

→ From ServletAPI 2.3 onwords Servlet Listeners are introduced to perform Event Handling on request, ServletContext, HttpSession objects. By using these Listeners we can keep tracks of various activities (operations) related these objects creation, destruction and these objects related attributes creation, modification and Destruction.

→ By keeping track of when request object is created and destroyed we can find out the request processing time of each request.

→ By keeping track of when ServletContext object is created and destroyed we can findout the web application deployement, undeployement, start, stop timings.

→ By keeping track of when HttpSession object is created & destroyed we can findout that hoomuch time each user is there in the session.

Every ServletListener must be developed as separate java class implementing Listener interface and that class must be configured in web.xml file.

API details of Servlet Listeners :-

Source Obj	Event class	EventListenter (I)	Event Handling Method
request Obj	ServletRequestEvent (javax.servlet pkg)	ServletRequestListener (javax.servlet pkg)	requestDestroyed (-) requestInitialized (-)
request Obj	ServletRequestAttributeEvent (javax.servlet pkg)	ServletRequestAttributeListener (javax.servlet pkg)	attributeAdded (-) attributeRemoved (-) attributeReplaced (-)
Servlet Context obj	ServletContextEvent (javax.servlet pkg)	ServletContextListener (javax.servlet pkg)	contextDestroyed (-) contextInitialized (-)
Servlet Context obj	ServletContextAttributeEvent (javax.servlet pkg)	ServletContextAttributeListener (javax.servlet pkg)	attributeAdded (-) attributeRemoved (-) attributeReplaced (-)
HttpSession Obj	HttpSessionEvent (javax.servlet.http pkg)	HttpSessionListener (javax.servlet.http pkg)	sessionCreated (-) sessionDestroyed (-)
HttpSession Obj	HttpSessionBindingEvent	HttpSessionAttributeListener	attributeAdded (-) attributeRemoved (-) attributeReplaced (-)

→ Tomcat Server gives logfiles in <tomcat_home>/logs folder on one per day basis.

to write messages to this logfile use sc.log (-) method.

Eg:- ServletContext sc=getServletContext();
sc.log("from webApplication");

*) procedure to apply 3 servlet Listeners on Java webapplication (URLApp)
↳ Appli: 15 of Pagno: 8

Step1:- keep the URLApp application ready.

Step2:- Develop the following 3 Listener classes and place them in WEB-INF/classes folder.

```
import javax.servlet.*;
import javax.servlet.http.*;
//MyReqListener.java
public class MyReqListener implements ServletRequestListener
{
    long sttime, endtime;
    public void requestInitialized(ServletRequestEvent sre)
    {
        sttime = System.currentTimeMillis();
    }
    public void requestDestroyed(ServletRequestEvent sre)
    {
```

↳ executes when request object is created
↳ gives current time in milliseconds
↳ execute when request obj is destroyed

```
endtime = System.currentTimeMillis();
    //getServletContext obj
ServletContext sc = sre.getServletContext();
    //Write each request processing time to log file
sc.log(sre.getServletRequest().getServletPath() + "has taken" + (endtime - sttime) + "milliseconds
    ↳ gives request obj for request processing"  ↳ gives current webresource path);
}
}
    //get req obj
ServletRequest req = sre.getServletRequest();
HttpServletRequest hreq = (HttpServletRequest) req;
    //Write request processing time to log file
sc.log(hreq.getServletPath() + "has taken" + (endtime - sttime) + "ms for request processing");
}
}
```

```
//MyStListener.java
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
public class MyStListener implements ServletContextListener
{
  long sttime, endtime;
  public void contextInitialized(ServletContextEvent sce)
    ↳ executes when servletContext obj is created
  {
    sttime = System.currentTimeMillis();
    ServletContext sc = sce.getServletContext();
    sc.log("webapplication is deployed at " + new Date());
  }
  public void contextDestroyed(ServletContextEvent sce)
    ↳ executes when servletContext obj is destroyed
  {
    endtime = System.currentTimeMillis();
    //getServletContext obj
    ServletContext sc = sce.getServletContext();
sc.log("webApplication is undeployed / stopped / reloaded at " + new Date());
sc.log("webApplication is there in running mode continuously for " + (endtime - sttime) + "ms");
}
}
```

```
// MySesListener.java

import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;

public class MySesListener implements HttpSessionListener
{
  long sttime, endtime;

  public void sessionCreated(HttpSessionEvent hse)
  {
    sttime = System.currentTimeMillis();

    // get Session object
    HttpSession ses = hse.getSession();
    // get ServletContext object
    ServletContext sc = ses.getServletContext();

    sc.log("Session is started at "+new Date());
  } // method

  public void sessionDestroyed(HttpSessionEvent hse)
  {
    endtime = System.currentTimeMillis();

    // get ServletContext, session objs
    HttpSession ses = hse.getSession();
    ServletContext sc = ses.getServletContext();
    // calc Session duration
    long duration = endtime - sttime;
    // write msgs to log file
    sc.log("Session is completed at "+new Date());
    sc.log("ses.getAttribute("name")+" is there in Session for"+duration+"ms");
  }
}
```

Step-3:- Configure these Listener classes in web.xml file.

in web.xml

```
<web-apps>
  <servlet>
    <s-n> fsrv </s-n>
    <s-c>
  </servlet>
  <servlet>
```

In this example previous HttpSession program 3 servlet classes will be write in this

after the closing third servlet </servlet-mapping> write these Listeners program no: 15 page no: 86 xml

```
<listener>
    <listener-class> MyReqListener </listener-class>
</listener>

<listener>
    <listener-class> MyScListener </listener-class>
</listener>

<listener>
    <listener-class> MySesListener </listener-class>
</listener>
</web-app>
```

Step(4):- Execute the URLApp application in reguler manner and observe logfile of <Tomcat_home>\log folder

→ we can use Listeners to perform monitoring on deployed webapplications without touching the webresource programs ~~their code~~ code.

→ our servlet container creates request, response objects for the request given to html programs.

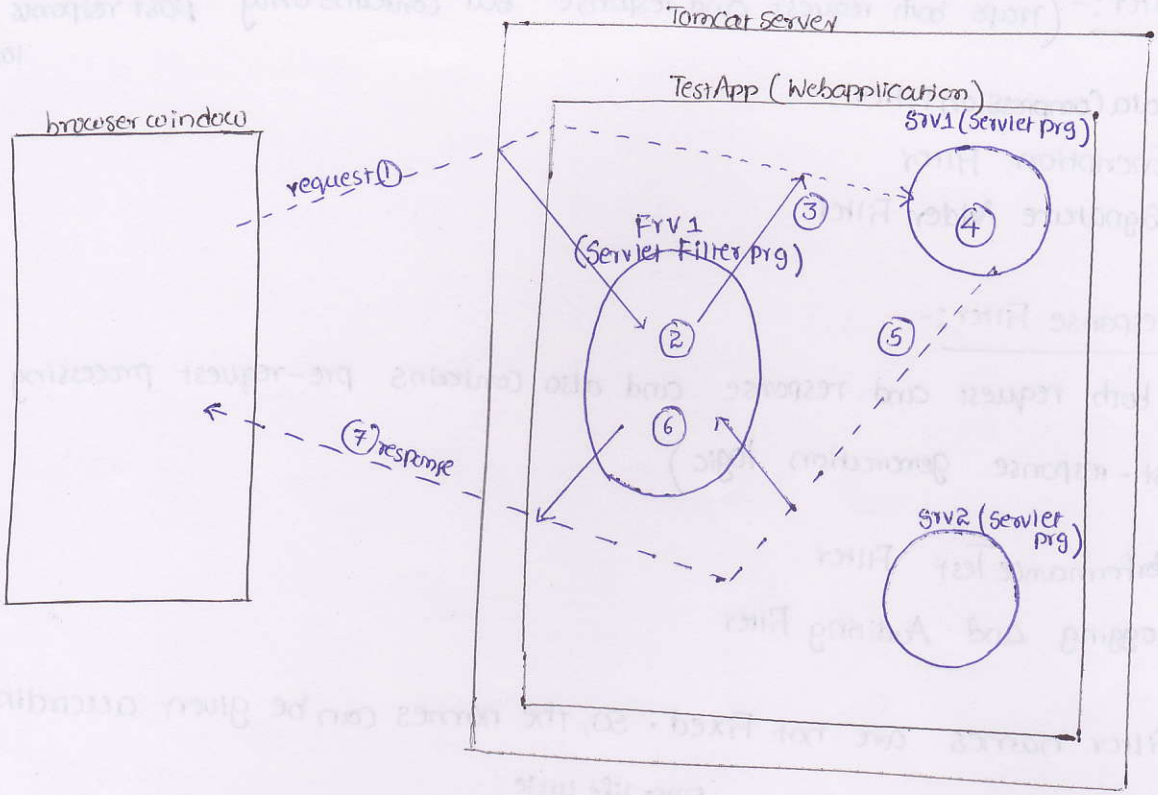
15/11/2012

Servlet Filters :-

→ Servlet Filter is a special web resource program of Java webapplication. That traps requests and response of other web resource programs of web application.

→ Servlet Filter and executes pre-request processing logic after trapping the request. Similerly it executes post response generation logics after trapping the response.

→ Servlet Filters are very useful to execute additional logics in webappn without disturbing the existing logics of webresource programs.



→ Every Servlet Filter is a Java class. implementing javax.servlet.Filter (I)

→ Servlet Filters are introduced from Servlet API 2.3

W.r.to the diagram

① Browser window gives request to Srv1 program.

② Filters traps this request and executes pre-request processing logic.

③ Filter passes the request to the actually requested Srv1 program.

④ Srv1 program executes the B.Logic and generates the results

⑤ Srv1 program generates the response to browser window

⑥ ServletFilter traps that response and executes post-response generation logic

⑦ ServletFilter sends Final response to browser window.

We can develop 3 types of Servlet filters :-

① Request Filter (traps both request and response but contains only pre-request processing logic)

Eg:- Authentication Filter
Authorization Filter
~~Logging and Auditing~~
RequestDumper Filter

② Response Filter :- (traps both request and response but contains only post response generated logic)

Eg:- Data Compression Filter
Encription Filter
Signature Adder Filter

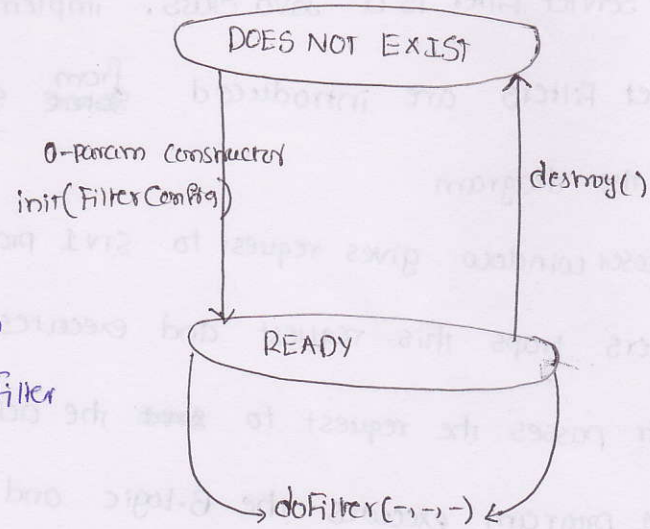
③ Request - Response Filter :-

(traps both request and response and also contains pre-request processing logic and post-response generation logic)

Eg:- Performance Test Filter
Logging and Auditing Filter

→ Servlet Filter names are not Fixed. So, the names can be given according to their functionalities.

Filter life cycle



→ There is no FilterContext obj but we can access ServletContext obj in our ServletFilter prg by using FilterConfig obj

→ ServletContainer creates and initializes our Filterclass obj either during Server startup or during the deployement of web application.

→ Servlet Container destroyes this object when webapplication is stoped / reloaded / undeployed.

→ init(FilterConfig), doFilter(-,-,-), destroy() are the lifecycle methods of ServletFilter Component / program.

→ There is no neccessity of enabiling <load-on-startup> on Filter program.

→ Every Filter program must be Configured in web.xml file using <filter>, <filter-mapping> tags. Every filter program is identified with its URL pattern.

16/11/2012

init(FilterConfig) :-

→ Useful to place initialization logics like creating jdbc con obj. using FilterConfig obj we can gather init param values of Filter prg.

doFilter(ServletRequest req, ServletResponse res, FilterChain fc) :-

→ Useful to place pre-request processing and processing response logics. FilterChain obj represents the webresource prg that is mapped without Filter prg.

```
public void doFilter(ServletRequest req, ServletResponse res, FilterChain fc)
{
    -------
    -------
    -------   } (pre-request processing logic)
    -------

    fc.doFilter(req, res);  ---> passes the control to webresource prg that is mapped with Filter.

    -------
    -------   } (post-response generation logic)
    -------
}
```

destroy() :-

→ Useful to place uninitialization logics like closing jdbc objs.

→ To map ServletFilter with Servlet prg / JSP prg, then the url pattern of Servlet/JSP prg must be taken as the url pattern of ServletFilter prg.

→ We can map (link) one or more ServletFilter prgs with one or more Servlet/JSP prgs.

Example :-

Srv1, Srv2, Srv3, Srv4 are Servlet programs

Frv1 is Filter program

To map Frv1 Filter with all Servlet prgs :-

url pattern of Frv1 prg is /*

To map Frv1 Filter prg with Srv1 prg

Srv1 url pattern is /S1url

Frv1 url pattern is /S1url

To Map Frv1 with srv1, srv2 prgs :

Srv1 prg url pattern /X/S1url

Srv2 prg url pattern /X/S2url

Frv1 prg url pattern /X/*

To Map Frv1 with srv1, srv2 prgs and Frv2 with srv3, srv4 prgs :

Srv1 prg url pattern /X/S1url

Srv2 prg url pattern /X/S2url

Frv1 prg url pattern /X/*

Srv3 prg url pattern /Y/S3url

Srv4 prg url pattern /Y/S4url

Frv2 prg url pattern /Y/*

To Map Frv1 with srv1, srv2, srv3 prgs and Frv2 with srv3, srv4 prgs :

Srv1 prg url pattern /X/S1url

Srv2 prg url pattern /X/S2url

Srv3 prg url pattern /X/S3url

Srv4 prg url pattern /Y/S3url

Frv1 prg url pattern /X/*

Frv2 prg url pattern /*/S3url

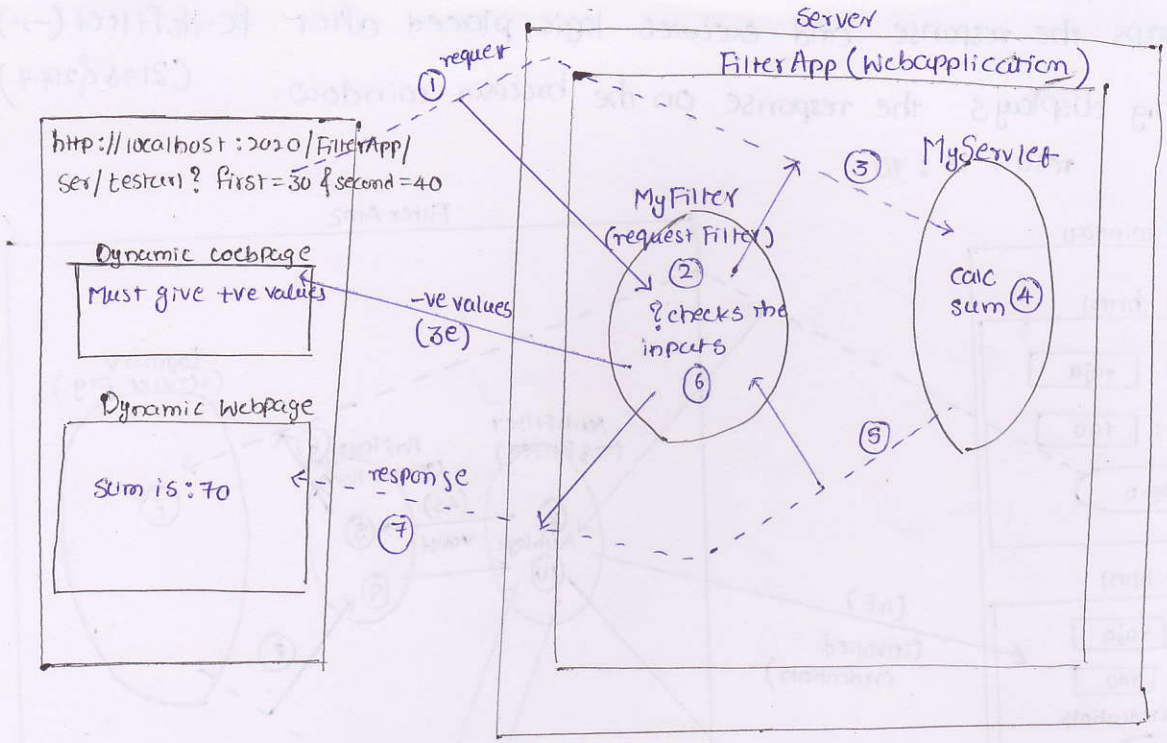
To Map Frv1, Frv2 with Srv1 prg :

Srv1 url pattern /S1url

Frv1 url pattern /S1url

Frv2 url pattern /S1url

NOTE :- when multiple ~~Fils~~ Filter programs are mapped with one Servlet prg then Filter prgs trap the request going to Servlet prg in the cfg order of web.xml and traps the response coming from Servlet prg in the reverse orger of cfg.



NOTE:- In this Application MyFilter block the request going to MyServlet prog when -ve inputs are given other wise it passes the request to MyServlet prog.

For the above diagram based web application refer the application (16) Page no: 89 & 90

→ When Filter program traps the request of web resource program then Filter and web resource program uses <u>same request</u> & <u>response objects</u>.

<u>Flow of execution related to application (16)</u>

A) programmer deploges the FilterApp webapplication

B) Servlet Container Creates MyFilter class object based on its configurations done in web.xml file (2156-2162)

C) Servlet Container maps the MyFilter with MyServlet based on their configurations done in web.xml file (2162 & 2171)

D) Enduser gives request to MyServlet program (http://localhost:2020/FilterApp/ser/testurl? first=30 & second=40)

E) MyFilter traps the request and calls doFilter(-,-,-) on MyFilter class object (2128)

F) Pre-request processing logic in doFilter(-,-,-) method execute (2131-2141)

G) fc.doFilter(-,-) call. forward the request to MyServlet program (2145)

H) Servlet Container Creates/locates MyServlet class obj based on its Configuration done in web.xml file. (2165-2172)

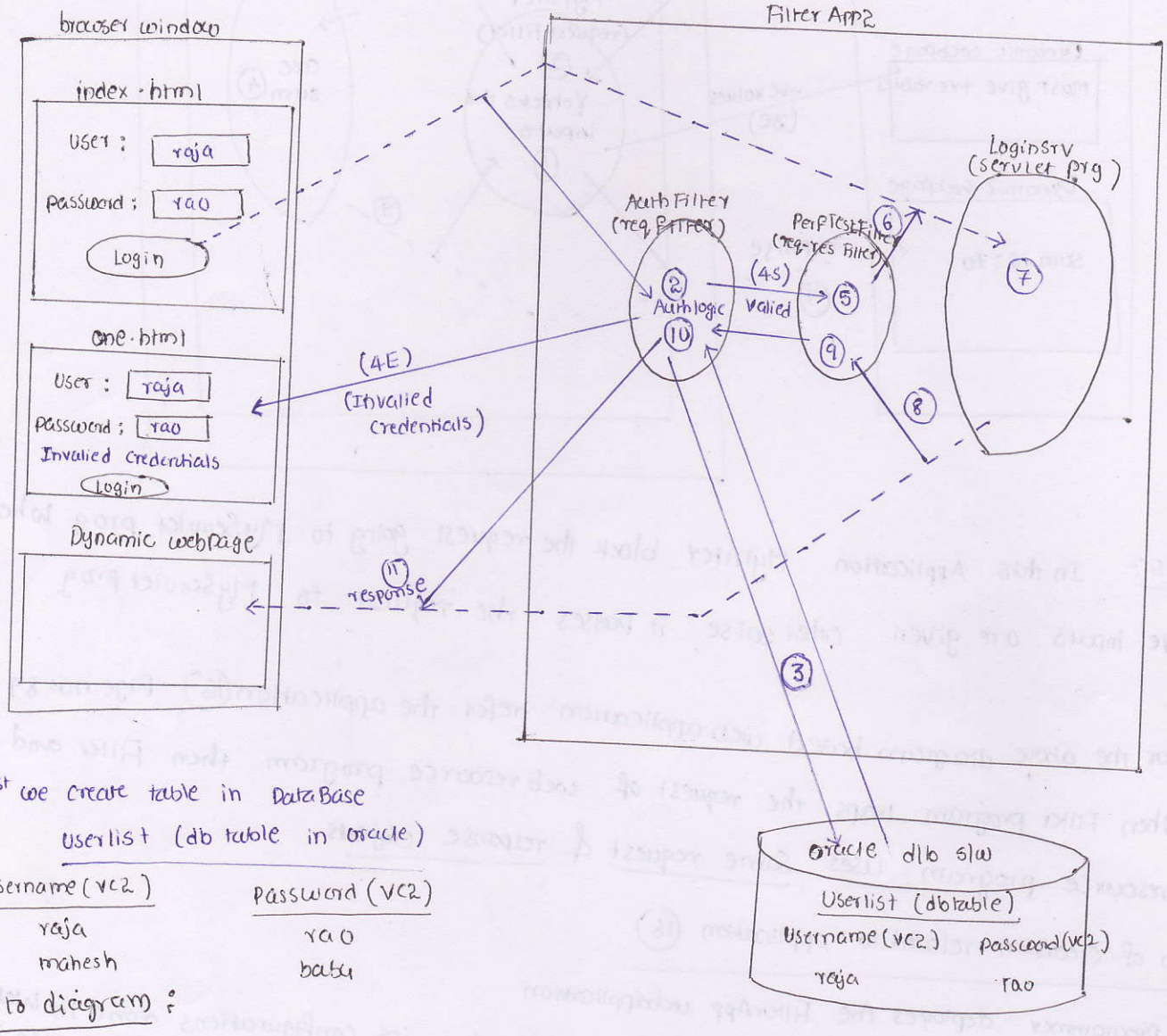
I) Servlet Container calls doGet(-,-) on MyServlet class object indirectly (2182-2194)

J) MyServlet program generates response

K) MyFilter traps the response and executes logic placed after fc.doFilter(--) method. (2146 & 2147)

L) MyFilter prg displays the response on the browser window.

result is : 70



1st we create table in DataBase

Userlist (db table in oracle)

Username (vc2)	Password (vc2)
raja	rao
mahesh	babu

W.r.to diagram :

(1) The Form page index.html generates request to LoginSrv prg

(2) & (3) Auth Filter traps & takes the request and executes Authentication logic to check the credentials (User, password)

(4E) If credentials are invalied the Form page will error msg will be displayed

(4S) & (5) If credentials are valied the next Filter PerfTestFilter traps the request and it notices request trapping time.

(6) & (7) perfTestFilter forwards the request to LoginSrv program to execute its B.Logic

(8) & (9) LoginSrv program generates the response and PerfTestFilter traps and takes the response. This filter notices response trapping time and writes the difference b/w Request trapping time and Response trapping time to Logfile.

(10) & (11) AuthFilter traps the response and sends the response to browser window.

(*) For above diagram based example application refer appn (17) of page NO's 90 to 93

2220 /loginurl } Here two filters are applied on single SOV program.
2228 /loginurl }

2236 /loginurl

2204 "loginurl" refer line no: 2236

2245 "implements Filter" → manditory

2252: To provide global visibility to FilterConfig obj (f = FilterConfig → it is the parameter init())

2263: } reading form data
2264: }

2266 – 2268: writing the details about current request to log file.

2280: if credentials (uname, & password) are valied

chain.doFilter(request, response);
↳ to forward the control to next filter.

2287: if credentials are invalied

2290: login_page → refer: 2262

2289–2291: - logic to display one.html when the given credinticals are invalied credentials.

2270 to 2295: logic for authentication (checks the username, pwd values).

one.html
↳ Same as Index.html but contains error msg.

2315 :- LoginSrv change to "loginurl"

2239: AuthFilter.java (requestFilter having authentication logic)

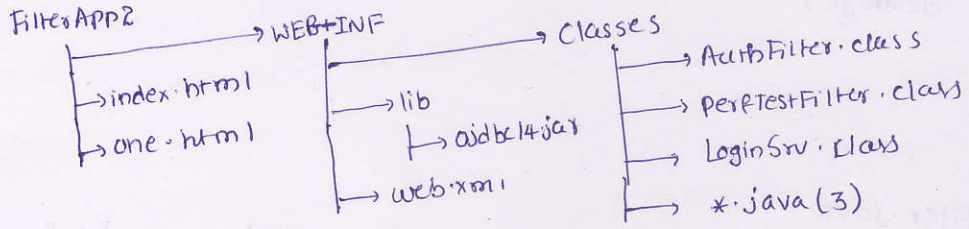
2322: PerfTestFilter.java (req-res Filter having logic to calculate request processing time).

2339: getting request trapping time.

2341: forward the request to loginSrv program

2343: getting response trapping time

Deployment directory structure



Request url:-

http://localhost:2020/FilterApp2/index.html

→ Developing Servlet filter that counts and displays no. of request that are given to webapplications.

Step-I :- keep ~~URL app~~ URLApp application ready.

Step-II :- Develop the filter program as shown below in WEB-INF/classes folder

```
// CountFilter.java
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class CountFilter implements Filter
{
    int cnt;
    ServletContext sc;
    public void init(FilterConfig fg)
    {
        //get ServletContext object
        sc = fg.getServletContext();
    } init(-)
    public void doFilter(ServletRequest req, ServletResponse res, FilterChain fc) throws SE,IOE
    {
        S.O.P("CountFilter: doFilter(-,-)");
        // Count requests
        cnt++;
        sc.setAttribute("reqcnt", cnt);
        fc.doFilter(req, res);
    } //doFiter(-,-,-)
    public void destroy()
    {
    } sc=null;
} //class
```

/> Javac CountFiter.java

Step-III :> configure the above servlet program in web.xml file having "/*" as the url pattern.

```
<listener>
    <listener-class> MyReqListener </l-c>
</listener>
<listener>
    <listener-class> MyScListener </l-c>
</listener>
    <listener>
    <listener-class> MySesListener </l-c>
</listener>
```

```
<filter>
    <filter-name> cnt </f-n>
    <filter-class> CountFilter </f-c>
</filter>

<filter-mapping>
    <filter-name> cnt </f-n>
    <filter-class> /* </f-c>
</filter-mapping>
```

↳ makes filter program to trap all the request and all the responses of web application

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

Step-IV :- Display RequestCount in every webresource program by reading the ServletContext Attribute value.

In FirstServlet.java & SecondServlet.java & ThirdServlet.java

```
ServletContext sc=getServletContext();
Pw.println("<br><br> request count is : " +sc.getAttribute("reqcnt"));
```

Step-V :- execute the application in Normal manner.

NOTE :- The above application is useful to findout the no.of visits that are given to web application.

Assignment :-

Write a ServletFilter program to allow the user joining in the existing Session if is already logged in. other wise show the login page to begin the new Session.

```
doFilter()
req.getSession() != null &&
~~req.getSession~~ session.getAttribute
!= null
fc.doFilter(req,res);
RequestDispatcher
else
{
Requ  RD = req.getRequestDispatcher
("login.html");
```

Servlet Thread Safety :-

18/11/2012

→ If multiple threads are acting on single object or variable simultaneously (or) concurrently then there is a possibility of Data Corruption.

To overcome this problem allow only one thread at a time on to that object on variable to make object (or) variable as Thread safe.

→ Our Servlet program is not thread safe by default. Because it is single instance multiple threads component. (when multiple requests are given multiple threads will be started concurrently on single object of our Servlet class).

→ To make our Servlet program as Thread safe use the following techniques.

1. place only local variables in service(-,-) / doXxx(-,-) methods.
2. use Synchronized blocks in service(-,-) / doXxx(-,-) methods
3. make our Servlet class implementing javax.servlet.SingleThreadModel (I)

NOTE:- 2nd technique is recomended.

→ Making Servlet program as Thread safe program is nothing but making its data (variables as Thread safe). refer 17/09/2012, 13/10/2012 pages

Technique 1 (place local variables in service(-,-) / doXxx(-,-) methods)

→ Every thread started on our Servlet class object for each request represents service(-,-) / doXxx(-,-)

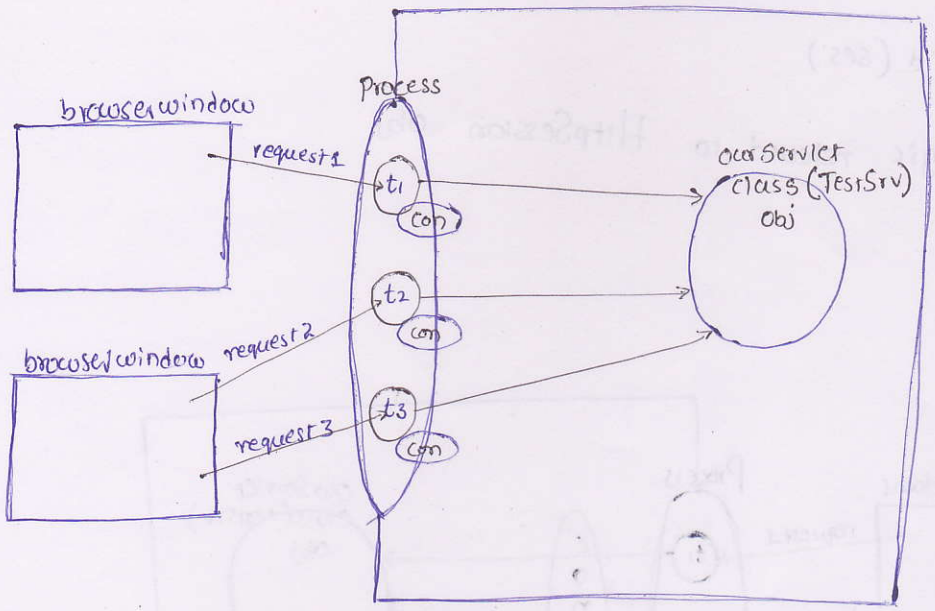
→ In this technique we should not take Instance Variable in Servlet prg. We should only local variables in service(-,-) / doXxx(-,-)

→ Local Variable of service(-,-) / doXxx(-,-) are thread safe by default bcoz every thread gets its own copy of local variable

Sample Code

```
public class TestSrv extends HttS/GenS
{
    public void service(-,-) / doXxx(-,-)
    {
```

```
Connection con=null; //local variable
------
------ //logic to create and use "con" obj
------
}
}
```



t1, t2, t3 are threads

NOTE:- if Technique 1 fails when we can't avoid working with Instance Variables.

Technique-2 (Working with Synchronized block)

→ This technique allows instance variables.

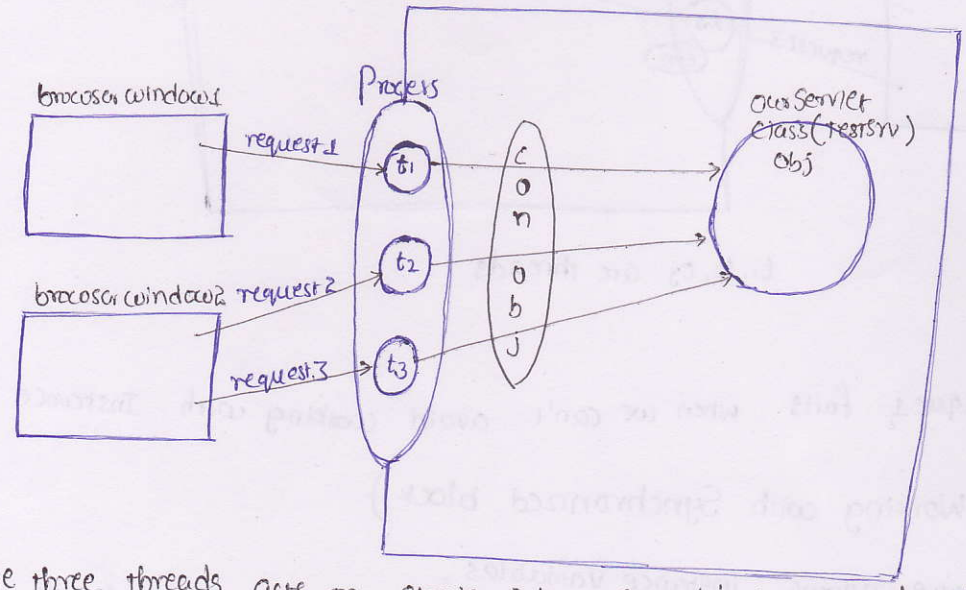
→ Place synchronized blocks in Service(-,-)/doXxx(-,-) methods on important objects like Connection object, ServletContext object, HttpSession object and ... etc.

→ This is most recomended technique to use

Sample code:-

```
public class TestSrv extends H.S/G.S
{
    Connection con=null; // instance Variable but thread safe
    public void init()
    {
        ------ //logic to create con obj
    }
    public void service(-,-)/doXxx(-,-)
    {
        Synchronized (con)
        {
            ------ //logic to use "con" obj
        }
```

```
Synchronized (sc)
{
    ----
    ----   // logic related to ServletContext obj
    ----
}
----
----
Synchronized (ses)
{
    ----   //logic related to HttpSession obj
    ----
}
} //method
} // class
```



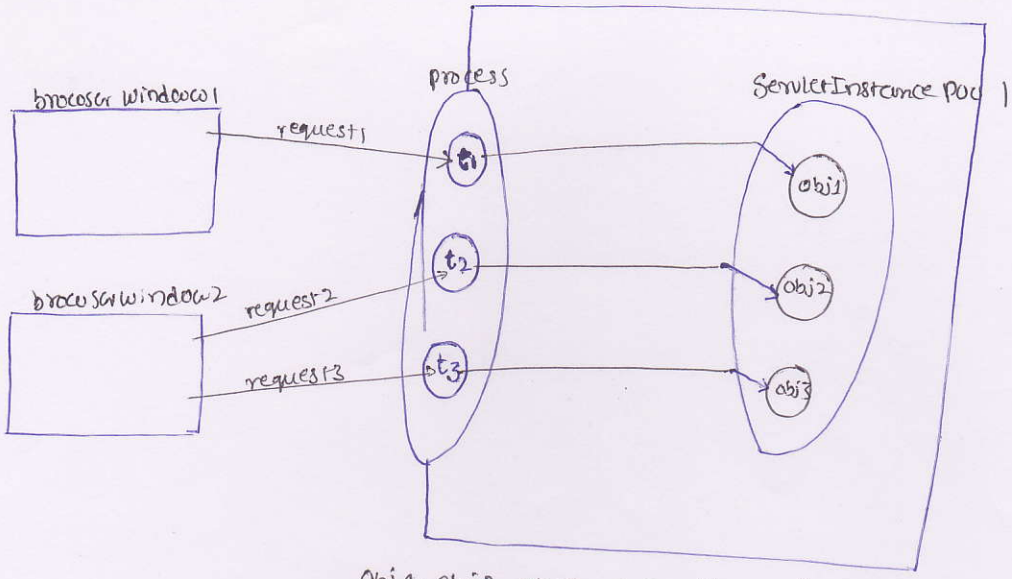
All the three threads acts on single jdbc con obj but one thread at a time.

<u>Technique (3)</u>:- (implementing javax.servlet.SingleThreadModel (I))

→ This marker interface makes Servlet Container to create <u>ServletInstance pool</u>. having set of readly available objects of our Servlet class. and uses them <u>for multiple request on one per request</u> basis. Due to this every thread acts on only one object of our servlet class

<u>Sample code</u>:-

```
public class TestSrv extends H.S/G.S implements SingleThreadModel
{
    ----   //any thing (local variables | Instance variable) can be placed.
    ----
} //class
```

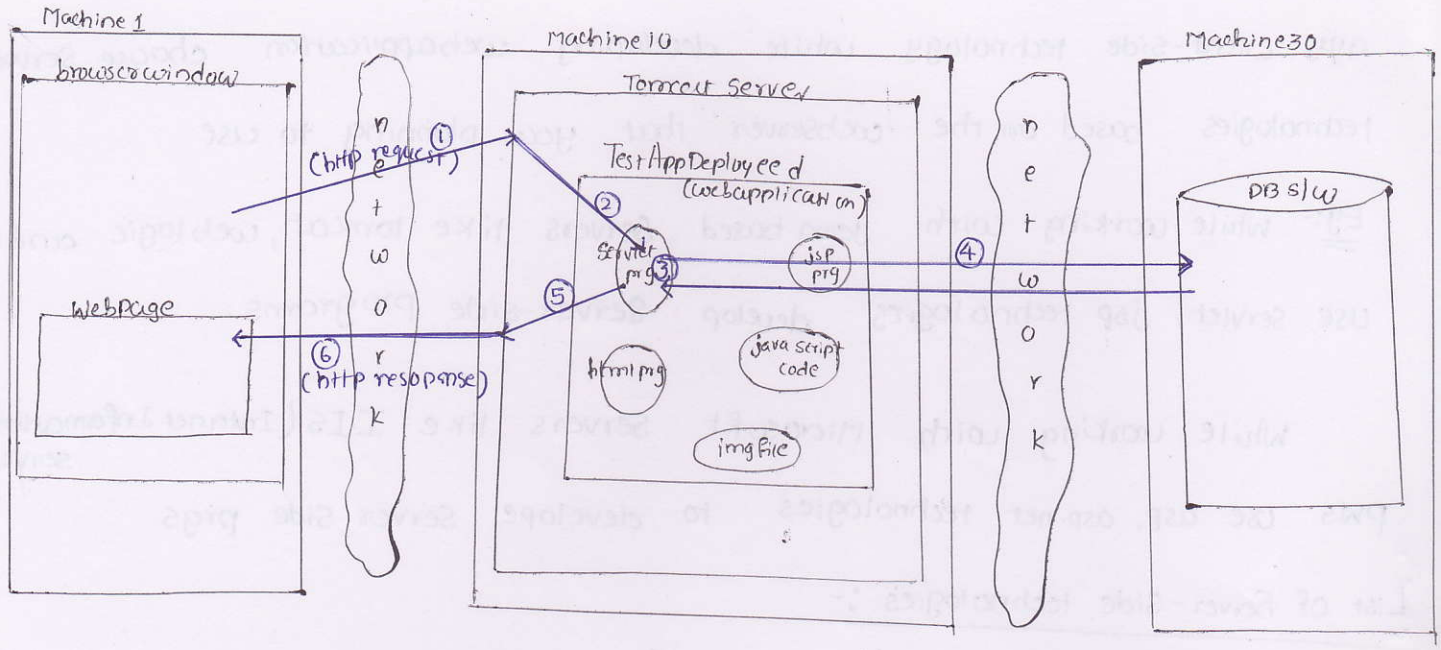


Obj1, Obj2, Obj3 are objects of our Servlet class (TestSrv)

→ creating multiple objects of our servlet class through ServletInstance pooling against Servlet Specification and also kills the performance. Due to this the javax.servlet.SingleThreadModel (I) has been depricated from ServletAPI 2.4.

JSP (Java Server Pages)

19/11/2012



→ Web application is collection of webresource programs generating webpages

→ The web application i.e. hosted on the internet n/w having Domain name (like www.xy.com) is called Website

→ To develop webresource programs we need web technologies. There are 2 types of web technologies.

① Serve-side webtechology :-

→ (required to develop server-side webresource programs)

Eg:- Servlets, JSP, ASP.net, ASP, PHP and etc....

NOTE:- Server-Side web resource programs executes in Server.

Eg (servlet prg | comp, jsp prg | comp, asp.net prg.....)

② Client-Side Webtechnologies :-

→ Required to develop client side web ~~application~~ resource prgs.

Eg:- html, javascript, vbscript, ajax and etc....

NOTE:- Client web resource prgs come to browser window (Client) from the webapplication of webserver (Server) for execution.

Eg:- html prg, java script code, vbscript code and etc.

NOTE:- Decide weather webresource program is Client-side (or) Server-side based on the place where it executes not based on the place where it resides.

I would try to update our site JavaEra.com everyday with various interesting facts, scenarios and interview questions. Keep visiting regularly.....

Thanks and I wish all the readers all the best in the interviews.

www.JavaEra.com

A Perfect Place for All Java Resources