Hi Friends,

Just go through the following small story.. (taken from **"*You Can Win* "** - by Shiv Khera)

Once upon a time a very strong woodcutter asks for a job in a timber merchant, and he got it. The paid was really good and so were the work conditions. For that reason, the woodcutter was determined to do his best. His boss gave him an axe and showed him the area where he was supposed to work. The first day, the woodcutter brought 18 trees "Congratulations," the boss said. "Go on that way!" Very motivated for the boss' words, the woodcutter try harder the next day, but he only could bring 15 trees. The third day he try even harder, but he only could bring 10 trees. Day after day he was bringing less and less trees. The woodcutter thought that "I must be losing my strength". He went to the boss and apologized, saying that he could not understand what was going on.

The boss asked, "When was the last time you sharpened your axe?"

"Sharpen? I had no time to sharpen my axe. I have been very busy trying to cut trees.

If we are just busy in applying for jobs & work, when we will sharpen our skills to chase the job selection process?

My aim is to provide good and quality content to readers to understand easily. All contents are written and practically tested by me before publishing. If you have any query or questions regarding any article feel free to leave a comment or you can get in touch with me on venus.kumaar@gmail.com.

This is just a start, I have achieved till now is just 0.000001n% .

With Warm Regards

Venu Kumaar.S
venus.kumaar@gmail.com

Servlet

JEE: Java Enterprise Edition

→ It is not installable s/w, it is given as specification.
JEE specification contains rules and guidelines to develop web-server and application server software like weblogic, tomcat etc.

→ **Note:** Working with JEE is nothing but working with one or other web-server or application server to develop the app.

→ these webserver, application server s/ws are installable softwares.

→ for JEE module JSE module is base module.

→ Using JEE module the following applications can be developed.
1) web applications(web sites)
2) Distributed appls
3) Enterprise appls
4) n-tier appls

→ The applications which can provide 24X7 access to their resources using internet environment are called as websites.

→ The c/s applications which provide location transparency are called as distributed applications. If client application is able to recognize the server application location change dynamically are called as location transparency.

→ A web app/web site can be developed as distributed or non-distributed application.

→ All credit card, debit card processing applications will be developed generally as distributed applications.

→ A large scale application that deals with complex and heavy weight business logic by having middleware service support is called as enterprise application. Ex: Banking Application.

→ The additional services that are configurable on the applications to make applications running perfectly in all the situations are called as middleware services.
Ex: Security service, Tx management Service, Pooling service etc.

→ **Note:** Middleware services are not minimum logics of the application these are additional and optional logics to develop.

→ The application that contains multiple layers/tiers (more than two) is called n-tier applications.
Ex: online shopping website environment.

→ Generally enterprise applications will be developed as n-tier applications.

→ Jee module concepts are Servlets, JSP, EJB, JTA(Java Tx API), JMS(Java Messaging Service), Java mail, JAAS(Java Authentication and Authorization service) , JMX(Java mgt console), JCA(Java Connector API) and etc.

→ Struts, spring and hibernate technologies are not part of sun micro systems JEE module. They are given by third party organizations as alternate and enhancement to JEE concepts.

→ While working with all JEE concepts we must take support of Web Serve or application server s/w.

→ Majority projects of java environment will be developed based on JEE module.

→ Every project belongs to one domain. This JEE module is suitable for developing banking financial service, Insurance, health care domain projects.

→ JEE module is suitable to develop banking financial service, Insurance, Health care domain projects.

→ JSE module is suitable to develop gaming, automation and retail domain projects.

→ JME module is suitable for developing mobile gaming, telecommunication domain projects.

→ The latest version of JEE module is 6.

API(Application progaramming Interface):

→ It is the base for the programmer to develop certain software technology based software applications.

→ Every s/w technology can develop user defined APIs and software applications.

→ In C language API is set of functions which come in the form of header files.

→ In C++ API is set of functions, classes which come in the form of header files.

→ In Java API s set of classes, Interfaces, enums, annotations which come in the form of packages.

Example of Java APIs:

1) **AWT API** → working with java.awt, java.awt.event packages
2) **Reflection API** → Working with java.lang, reflect pkg.
3) **JDBC API** → Working with java.sql, javax.sql pkg.
4) **Utility API** → working with java.util pkg.

→ **We can see three types of APIs in java:**

1) **Built-in APIs** → given by sun micro system along with jdk s/w. AWT, JDBC etc.
2) **user defined APIs** → given by programmers.
3) **3rd party APIS** → given by 3rd party organizations. ex: java zoom API.

→ the java related APIs come as in the form of jar files.(java zip file)

→ jdk\jre\lib\rt.jar file represents all the built-in APIs of jdk s/w.

Based on the place where web resource programs execute there are two types of web resource programs.
1) **Server side web resource programs** (they execute in the web server) Ex: Servlet prog, JSP prog and etc.

2)**Client side web resource program** → they come to browser window from the web application placed in web server for execution.
Ex: HTML prog, JS prog, Ajax prog and etc.

Decide whether web resource program is client side or server side based on the place where it executes not based on the place where it resides.

→ A web application looks like thin client-fat server application.

→ The process of keeping web application in web server is technically called as deployment and reverse process is called as undeployment.

Web server Responsibilities:

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

→ A **container** is a software or software application that can manage the whole life cycle(object birth to death) of given resource (like java class).

→ A file or program is called as resource of the application.

→ Servlet container takes care of servlet program lifecycle.

→ JSP container takes care of JSP program lifecycle.

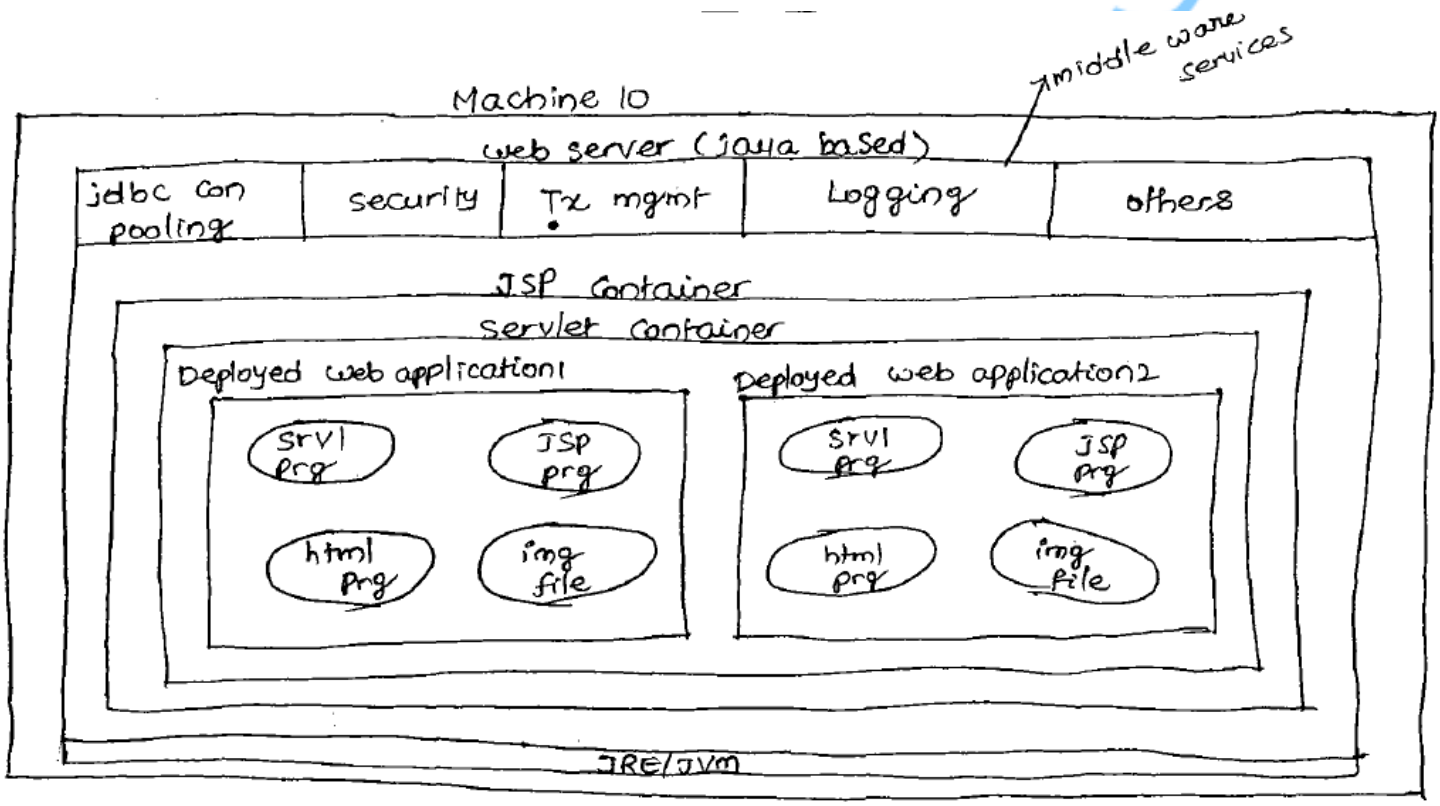
→ Applet container(Applet Viewer) takes care of applet program lifecycle.

→ Servlet container, JSP containers are part of web servers (java based).

→ A container is like an aquarium who can take care of the whole life cycle of given resources called fishes.

→ Programmer is not responsible to develop containers and web servers but he is responsible to use them to execute web applications.

Architecture of java based Web server:



→ Once web server is started one daemon process will be started to listen to client's continuously and to trap and take the client generated HTTP request.

→ The process that runs continuously is called as daemon process.

→ Every Java application contains two default threads. 1) Main Thread 2) Garbage collector Thread(Daemon Thread).

→ JSP container is the enhancement of servlet container and both containers uses JRE/JVM supplied by web server.

→ middleware services are configurable additional services on the applications to make applications executing perfectly in all the situations.

→ Security middleware service protects the application from unauthorized and unauthenticated users.

→ Transaction management service executes the logics by applying do everything or nothing principle.

→ JDBC connection pool supplies set of readily available JDBC connection objects.

→ Logging service keeps track of the application execution process through conformation statements/messages/log messages.

→ Middleware services are not minimum logics of application development. They are additional or optional services to apply on applications.

→ A web application can be there with or without middleware services.

→ Web server automatically activates servlet container and JSP container to execute servlet, JSP

programs when they are requested by clients. The client side web resource programs of web application goes to browser window for execution whereas the server side programs will be executed by using the containers of web server.

→ In web application executing environment browser software is called as client software and web server software is called as server software.

Ex: Java based Web Servers

Tomcat → from Apache Foundation

Resin → from Resin soft

JWS → from sun microsystem

→ HTTP is an application level protocol define in set of rules to get communication between browser window and web server. The text that allows sequential reading is called as normal text.

Ex: notepad.

→ The text that allows non-sequential reading through hyperlinks is called as hypertext.

Ex: Any web page content.

→ Protocol HTTP is useful to transfer hypertext between browser window and web server, web server and browser window.

Firefox → from Mozilla

Netscape Navigator → from Netscape

Internet Explorer → from Microsoft

Hot Java → from RedHat

Chrome → from google

Opera → from operasoft

Client side technologies for developing client side web resource programs :

HTML → from W3C

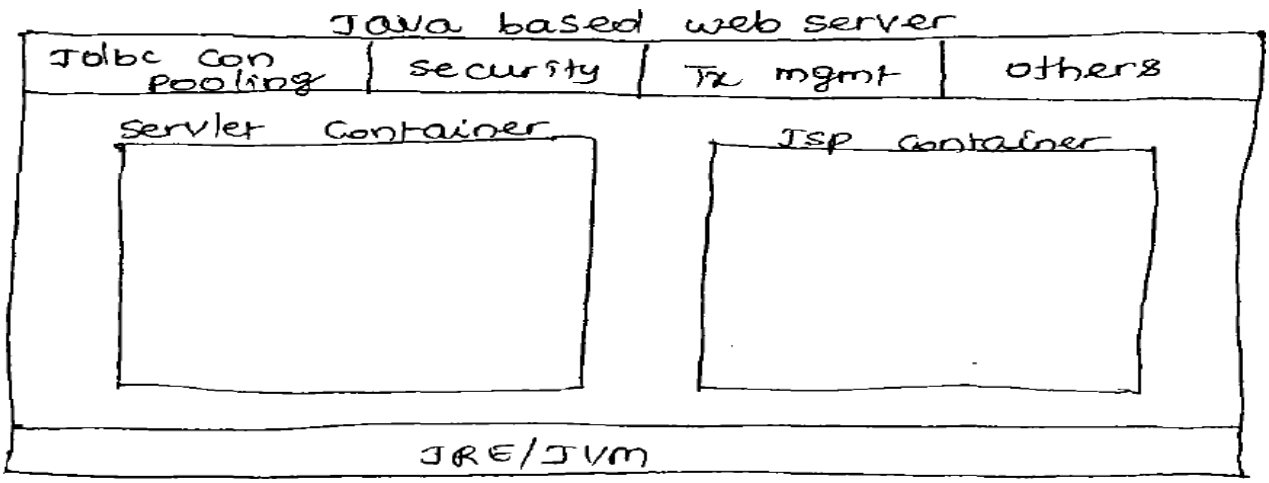
JavaScript → from Netscape

VB Script → from Microsoft

AJAX → from Adaptive path

→ The code that can not be executed independently and which must be embedded in other technology based program for execution is called as Script code.

→ Java script code can not be executed independently and it must be executed by embedding with html code so JS is called as script language.



JSP Container is developed based servlet Container

→ When java web application is deployed in java web server then the servlet programs of that web application will be executed by servlet container and JSP programs will be executed by JSP container.

→ Directly or indirectly every java application or java resource or java component uses JVM during execution.
Ex: Appletviewer executes an applet program but this applet viewer internally uses or take support from JVM.
Servlet container executes servlet program but this servlet container internally uses JVM support.

To develop server side web resource probgrams --

Different server side Technologies:

Servlet→ from Sun microSystems → java based (2)
JSP→ from Sun microSystems → java based (3)
ASP → from Microsoft → non-java based
ASP.net→ from Microsoft → non-java based (4)
PHP→ from Apache → non-java based (5)
ColdFussion→from Adobe → non-java based
SSJS (Server side Java Script) → from Netscape → non-java based

Different Web Server Softwares :

Tomcat →from Apache → java based (1)

Resin → from Resin soft → java based (2)
JWS → from Sunmicrosystems → java based
Jetty → from Adobe → java based
IIS → from Microsoft → non-java based(3)
PWS → from Microsoft → non-java based
AWS (Apache Web Server) → from Apache → non-java based(4)(for PHP programs)

Application Server = Web Server + EJB Container + more middleware services.

→ **Note:** EJB container is required to execute EJB components.

A reusable java class object is called as component.
Application server software is enhancement of web server software.
Every applciation server software can act as web server

Different Application Server softwares:

Weblogic → from BEA systems(Oracle corporation)(1)
Websphere → from IBM(3)
Jboss → from Apache/Redhat(4)
GlassFish → from sunmicrosystems(2)
OlogAS (Oracle log Applcation Server) → from Oracle Corporation
Jrun → from Macromedia(Adobe)

→ All application server softwares are java based softwares.

Different Database softwares

Oracle → Oracle corporation(1)
MySql→ from Devx(3)
SQLServer → from microsoft(2)
DB2 → from IBM
PostgreSql → from University of California (4)

→We can use client side technologies like HTML, JS along with any vendor supplied server side technologies in web application development

→ **To develop & execute java based web application**

a. Choose any browser software
b. Choose one or more client side technologies like HTML, JS.
c. Choose Servlet, JSPs as server side technologies.

d. Choose any java based web server software or application server software.
e. Choose any database software

→ **To develop and execute .net based web application.**
a. Choose any browser software
b. Choose one or more client side technologies like HTML, JS.
c. Choose asp.net as server side technology
d. Choose microsoft supplied IIS, PWS server
e. Choose any database software

→ **To develop PHP based web application.**
a. Choose any browser software
b. Choose one or more client side technologies like HTML, JS.
c. Choose PHP as server side technology
d. Choose AWS as web server
e. Choose any database software

→ To develop medium scale and large scale web applications use Java environment

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

→ To develop small scale web applications use PHP environment.

→ The servlet-api of servlet specification will be used by vendor companies to develop servlet container software. Where as programmers use the same servlet api to develop servlet programs as web resource programs of web application. Due to this servlet container is always capable of executing servlet programs.

JSP:

→ It is sun microsystem supplied an open specification having set of rules and guidelines to develop JSP container.

→JSP specification is enhancement of servlet specification so JSP container is also the enhancement of servlet container.

→ In Tomcat server JSP container is called as Jasper.

→ The Tomcat server supplies sun microsystems Jsp-api in the form of <tomcat_home>/lib/jsp-api.jar and supplies its own JSP container in the form of jasper.jar.

The Sun micro Systems JSP api contains following packages:

javax.servlet.jsp;
javax.servlet.jsp.el;
javax.servlet.jsp.target;

Server-side technologies:

Process based → CGI-Perl

Thread based → Servlet, JSP, ASP, Asp.net, PHP

Servlet:

1. Servlet is server side technology that allows the programmers to develop dynamic web resource programs or sever web resource programs in java based web application.
2. Servlet is a software specification given by sun micro systems provides set of rules and guidelines for vendor companies to develop the softwares called servlet containers.
3. servlet is a single instance multiple threads based java component in java web application to generate dynamic web pages.
4. Servlet is a server side technology that can enhance functionalities of web server software or application server software.

→ Servlet program is server independent because one servlet program executed in any java based web server or application server software without modifications.

→ Securing web application is nothing but applying authentication and authorization on the web application.

→ Checking the identity of a user is called as authentication.

→ Checking the access permissions of the user on particular resources of the project is called as authorization.

java 5 → servlet 2.5
jsp 2.0
Ejb 3.0

java 6 → servlet 3.0
jsp 2.1
Ejb 3.0

→ In JEE module of java all are specifications.

→ In JSE module of java awt, swing, are technologies, JDBC, JNDI are specifications.

3 important resources of servlet-api:

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

→The predefined HttpServlet class is an abstract class containing no abstract methods.

→ If we want to make method of one java class accessible only through subclasses then make that class as abstract class even though that class contains only concrete methods.

javax.servlet.Servlet

void int(ServletConfig)
void service(ServletRequest, ServletResponse)
void destroy()
ServletConfig getServletConfig()
String getServletInfo()
^
| implements
|

javax.servlet.GenericServlet(Abstarct)
init(ServletConfig) --> collected from javax.servlet.Servlet
init() --> direct method of GenericServlet class
Does not implement service() method

^
| extends
|

javax.servlet.http.HttpServlet(Abstract)
doXxx(HttpServletRequest, HttpServletResponse)
protected service(HttpServletRequest, HttpServletResponse)
→ above two methods are direct methods of HttpServlet class
public service(ServletREquest, ServletResponse) --> inherited method

→ Every server program is a java class that is developed based on servlet-api. Thre are three ways to develop servlet program.

1) take a java class implementing javax.servlet.Servlet interface and provide implementation for all the 5 methods of that interface.
2) Take java class extending from javax.servlet.GenericServlet class and provide implementation for service() method.
3) Take java class extending from javax. servlet.http.HttpServlet class and override one of the 7 doXxx() methods or one of the two service() methods.

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

→ Programmer just supplies his servlet program related java class to servlet container then onwards servlet container is responsible to manage the whole life cycle of servlet program.

→ For all the request given to servlet program the servlet container creates one object but for every request given to servlet program the servlet container creates one separate request, response objects and

calls service(-,-) method by keeping these requests, response objects as argument values.

fig-22

→ servlet program uses request object to read details from the request like form data.

→ Servlet program uses response object to send response content to browser window through web server.

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

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

→ Servlet container creates 10 threads on servlet program object representing 10 requests.

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

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

program

Develop servlet program:

```
import jvavx.servlet.*;
improt java.io.*;
public class DateSrv extends GenericServlet
{
//implement service(-,-)
public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException
{
//set response content type
res.setContentType("text/html");

//get stream obj
PrintWriter pw=res.getWriter();

//Write req processing logic
java.util.Date d1=new java.util.Date();

//Write response content to browser window through webserver
pw.println("Date and Time = " + d1.toString());

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

req → represents the servlet container supplied request object

res → represents the servlet container supplied response object
text/html : MIME(Multipurpose Internet Mail Extension/Content type)
res.setContentType("text/html");
Represents that passes instructions to browser window through web server that this servlet program generate html code base response text.

other MIME types
application/msword
application/

printWriter pw=res.getWriter();
here pw is a stream object pointing to response object getWriter() called on res object returns the PrintWriter stream object.

→ **Note:** A stream object can represent either file or object as destination

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

pw.close→ closes the stream connection with response object.

→ Add the tomcat server supplied "servlet-api.jar" to classpath.
variable name:CLASSPATH
value:;d:\tomcat6.0\lib\servlet-api.jar;

Develop web.xml file having the servlet programs configuration

→ **Note:**
1) a program or file of application is called resource of the application.
2) Servlet program is called as the web resource program of the web application. All servlet programs must be configured in web.xml file.
3) specifying the details of certain resource program and making under-laying software like container, server and etc recognizing the resource programs is called as resource configuration. By configuring servlet programs in web.xml file we make servlet container recognizing servlet programs.

web.xml

```
<web-app>
<servlet>
        <servlet-name>abc</servlet-name> //logical name
        <servlet-class>DateSrv</servlet-class> //java class that is acting as servlet program
</servlet>
<servlet-mapping>
        <servlet-name>abc</servlet-name> // must match with the above logical name
        <url-pattern>/test</url-pattern> //url pattern of servlet program
</servlet-mapping>
</web-app>
```

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

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

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

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

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

Step1: start the tomcat server.
tomcat-home\bin\tomcat6
Step2: Deploy the above DateSrv web application
copy e:\DateSrv folder to Tomcat-home, e\webapps folder
above two steps performs the deployment of web application.

step3: Test the web application protocol

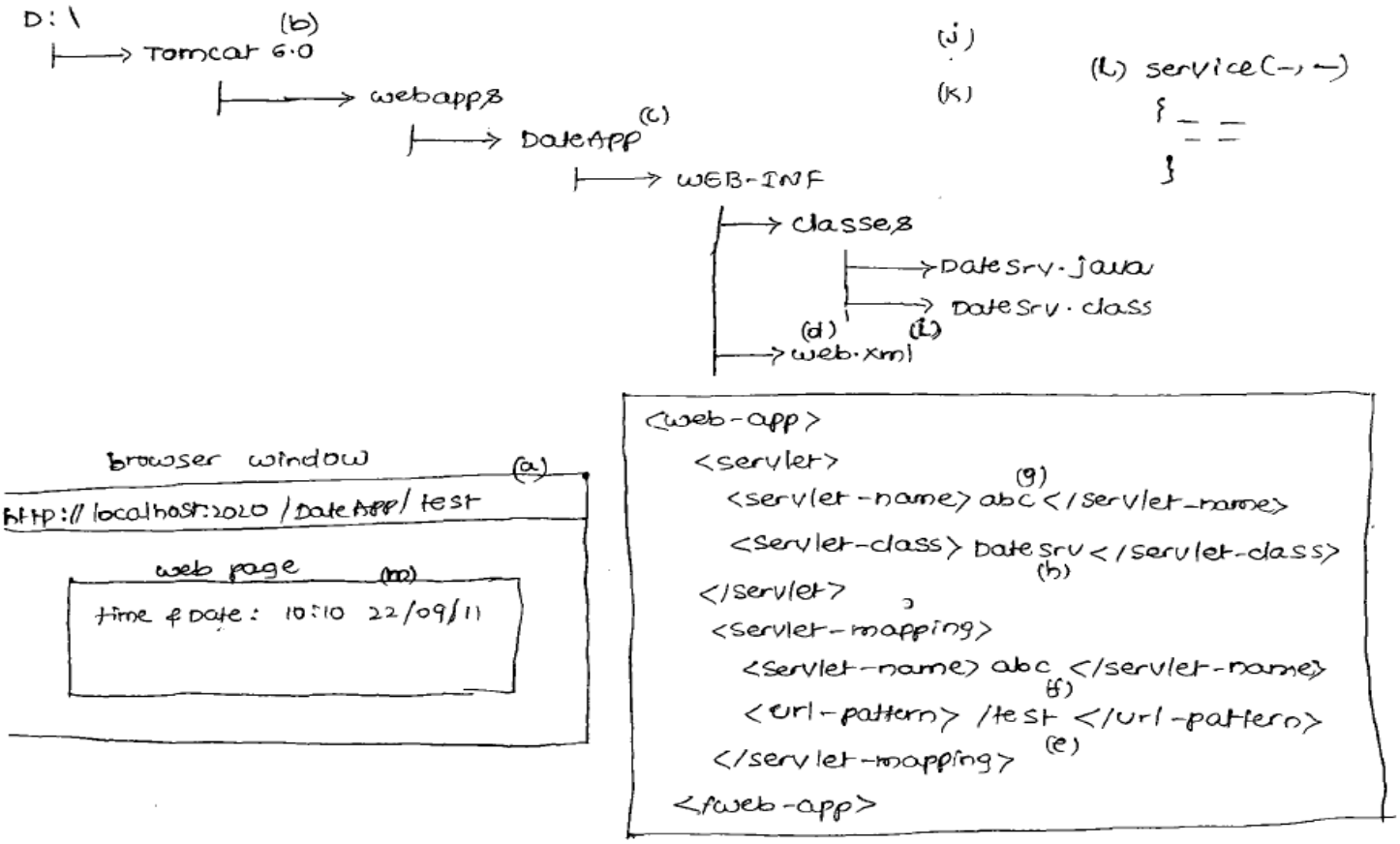
http://localhost:8080/DateApp/test

→Url pattern given to servlet is useful to hide the servlet related program class name from end users.

→ Java web applications are WODA(Write Once Deploy Anywhere) applications because:
1) The deployment directory structure is common for all web services.
2) The web resource programs like servlet, JSPs are server independent programs.

Flow of execution from request to response generation:

With respect to the above code:



a) End user types request URL in the address bar of browser window to generate request to servlet program.
b) Based on localhost:8080 content of request URL the request goes to the tomcat web server of local machine.
c) Based on DateSrv word of request URL the request goes to the deployed DateSrv web application of web server.
d) The servlet container uses web.xml file entries to locate the servlet program that is requested.
e),f),g),h) based on the URL pattern /test the logical name abc the servlet container gathers the classname of servlet program(DateSrv)
i) Servlet container loads DateSrv class from WEB-INF\classes folder of web application.
j) Servlet container creates and locates DateSrv class object, If created the servlet container finishes initialization process on our servlet class object.
k) Servlet container creates one set of request, response object for the current request and calls service(-,-) method by keeping them as argument values.
l) The **service(-,-)** method processes the request and generates the output.
m) This output goes to browser window through web server as the response in the form of dynamic web page.

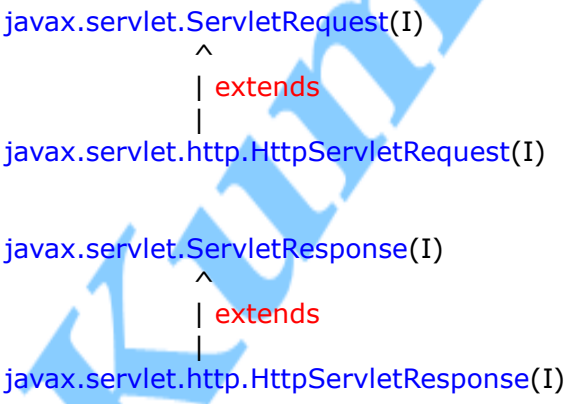
→ The **System.out.print()** statements generated output from servlets will appear on server console

whereas the printWriter.println() statements generated output will come on web page of browser window.

→ modifications done in web.xml file of deployed web application will be recognized by server automatically whereas the modifications done in servlet program source code will be reflected only after recompilation of servlet program and reloading of the web application.

→ servlet container uses 0-arg constructor to create object of our servlet class so make sure that our servlet class is having 0-arg constructor directly or indirectly. Directly means programmer should place 0-arg constructor explicitly. Indirectly means servlet program deals with the javac generated default 0-arg constructor.

→ The service() method placed in servlet program will be called by servlet container for every request that is given to servlet program from any browser window.



→ The req object of **service(-,-)** method is not the object of javax.servlet.ServletRequest interface, it is the object of servlet container supplied java class implementing javax.servlet.servletRequest interface. In tomcat sever req object class name is "org.apache.catalina.connector.RequestFacade".

→ The res object of **service(-,-)** method is not the object of javax.servlet.ServletResponse interface, it is the object of servlet container supplied java class implementing javax.servlet.servletResponse interface. In tomcat sever req object class name is "org.apache.catalina.connector.ResponseFacade".

→ To know req, res objects class-names in any server call getClass() method on those objects from service(-,-) method of servlet program.

```
pw.println("<br> req object class name = "+req.getclass());
pw.println("<br> res object class name = "+res.getclass());
```

→ All the request coming to our servlet program will use only one object of our servlet class but every request separate request, response objects and also starts one separate thread on servlet class object.

Code related to this is as follows:

In service(-,-) method you should include the following:

```
pw.println("<br> req object class name = "+req.getclass());
pw.println("<br> res object class name = "+res.getclass());

// Gives seperate values for each req
pw.println("<br> Hash code of req object = "+req.hashCode());
pw.println("<br> Hash code of res object = "+res.hashCode());

//Gives Seperate Thread name for each request
pw.println("<br> Current req Thread name = "+Thread.currentThread().getName());

//Gives same value for all the requests
pw.println("<br> Hash code of our Servlet class object = "+this.hashCode());

try { Thread.sleep(40000); }
catch(Exception e) { e.printStackTrace(); }
```

→ We can provide multiple URL patterns for a single servlet program in web.xml file.

```
web.xml
<web-app>
<servlet>
        <servlet-name>abc</servlet-name> //logical name
        <servlet-class>DateSrv</servlet-class> //java class that is acting as servlet program
</servlet>
<servlet-mapping>
        <servlet-name>abc</servlet-name> // must match wtih the above logical name
        <url-pattern>/test</url-pattern> //url pattern of servlet program
</servlet-mapping>

<servlet-mapping>
        <servlet-name>abc</servlet-name> // must match wtih the above logical name
        <url-pattern>/test1</url-pattern> //url pattern of servlet program
</servlet-mapping>
</web-app>
```

→ If web application is having multiple servlet programs then all the servlet programs must be configured in web.xml file having URL patterns.

.doc → application/msword
.xls → application/vnd.ms-excel
.html → text/html
.xml → text/xml
.bmp → image/bmp

.avi → video/avi

Different approaches of gathering req parameters values req body values from HTTP reques being from servlet program:

Approach1: by using req.getParameters(-);

code in service(-,-) of servlet program
String s1=req.getParameter("sno");
String s2=req.getParameter("name");
String s3=req.getParameter("sadd");

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

Approach2:by using req.getParameterNames();

code in service(-,-) of servlet program

Enumeration e=req.getParameterNames(); //given enumeration obj pointing to list ds having req param names

while(e.hasMoreElements())
{
String pname=(String)e.nextElement();//Gives each req param name
String pvalue=req.getParameter(pname); //Gives each req param value
pw.println(pname+"-->"+pvalue);
}

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

Approach3: by using req.getParameterValues(-);
String s[]=req.getParameter("sadd");
for(int i=0;i<s.length;i++){pw.println(s[i]+"...");} //gives all address values
String s1=req.getParameterVelues(s[0]); //gives first value.
String s1=req.getParameterVelues(s[1]); //gives second value.

→ **Note:**

→ This approach is useful to gather multiple values of single req parameter.

→ To read a req param values from http request we can use either ServletRequest object or HttpServletRequest object.

→ **Note:**
While opening Tomcat management through browser window if username and password related problems are raised, then add following two lines of code in Tomcat_home/user.xml

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

res.setContentType("text/html"); --> According to the above statement setContentType is the method that is declared in javax.servlet.ServletResponse interface and that method is implemented in container supplied java class implements this ServletResponse interface.

→ **Note:** When interface reference variable points to its implementation class object then it becomes object of implementation class and it is not the object of interface.

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

approaches to gather request header values form Http request being from servlet program:

Approach1: by using req.getHeader(-);

code in service(-,-) of servlet program
String s1=req.getHeader("user-agent"); //gives browser software name
String s2=req.getHeader("accept"); //gives the MIME types supported by browser window like text/html, text/xml, etc..
In this approach we must know request header name to get its header value.

Approach2: by using req.getHeaderNames();

code in service(-,-) of servlet program

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

this code gives all request header names and values.
The above code must be executed with the support of httpServletRequest Object.

To gather Current browser window software name being from servletprogram:

user user-agent request header that holds browser software name

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

→ The URL pattern of servlet program is technically called as servlet path.
We can gather the misc information from HttpRequest by calling various get methods on request object.

Ex: http://localhost:8080/DateApp/test?sno=1

```
//Methods available in javax.servlet.servletRequest(I)

//gives request data length in bytes, if not known gives -1
pw.println("<br> req content length: "+req.getContentLength());

//gives request content type like text/html. if not known gives null)
pw.println("<br> req content type:"+req.getContentType());

//Gives http/1.1
pw.println("<br> req Protocol: "+req.getProtocol());

//Gives Http
pw.println("<br> req SCheme:"+req.getScheme());

//gives 192.168.32.42
pw.println("<br> req browser window machine IP address: "+req.getREmoteAddr());

//gives current computer name otherwise IP address
pw.println("<br> req browser window machine host name:"+req.getRemoteHost());

//gives 2922 for Netscape, 2926 for IE
pw.println("<br> req browser window port no:"+req.getRemotePort());

//gives 8080
pw.println("<br> Server Port:"+req.getServerPort());


//Methods available in javax.servlet.HttpServletRequest(I)

//Gives DateApp
pw.println("<br> Context Path:"+req.getContextPath());

//GET
pw.println("<br> req method:"+req.getMethod());

//Gives additional information kept in req url otherwise null
pw.println("<br> req Path info is: "+req.getPathInfo());

//Gives sno=1
pw.println("<br> Query String :"+req.getQueryString());

//Gives /DateApp/test
pw.println("<br> req URI:"+req.getRequestURI());

//Gives http://localhost:8080/DateApp/test
pw.println("<br> req URL:"+req.getRequestURL());
```

```
//gives /test
pw.println("<br> Servlet path: "+req.getServletPath());
```

→ When web resource program (like servlet program) sends response to browser window the generated HttpResponse contain multiple details including response content. Those details can be remembered as SCH(Status Content Header) details

s→ Response status code(100-599)
100-199 --> Info (warning based)
200-299 --> Success
300-399 --> Redirection
400-499 --> Incomplete web resource
500-599 --> Server Error
default status code is 200

c→ content type (Web page content)

H→ Response headers like contentType, ContentLength, refresh and etc..,

List of all Http request headers:
accept, accept-encoding, authorization, connection, cookie, host, if-modified-since, referrer, user-agent, keep-alive, accept-charset and etc...

List of all Httpresponse headers:
Location, refresh, set-cookie, cache-control/pragma, content-encoding, content length, content-type, last-modified, date, server, connection and etc...

To make browser window refreshing its web page automatically at regular intervals:
Give instructions to browser window from servlet through server by using response header called refresh.
res.setHeader("refresh","10"); time in seconds

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

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

```
res.setHeader("cache-control",'no-cache"); for http1.1 based servers
res.setHeader("pragma",'no-cache"); for http1.0 based servers
```

→ Servlet life cycle methods:
init(Servletconfig cg) → instantiation event

service(ServletRequest req, ServletResponse res) → req arrival event

destroy() → destruction event

public void init(Servletconfig cg) throws ServletException
public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException
destroy()

When our servlet program gets first request from browser window:

1) Servlet container loads our servlet class from WEB-Inf\classes folder of deployed web application.
2) Servlet container instantiates(Object creation) our servlet class object as
Class.forName("DateSrv").newInstance();
Class.forName("DateSrv") --> loads our servlet class
newInstnace() --> creates object for the loaded class DateSrv
3) During instantiation process the 0-param constructor of our servlet class executes.
4) Servlet container creates one ServletConfig object for our servlet class object.
5) Servlet container calls init(-) life cycle method having ServletConfig object as arg value on our servlet class object.

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

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

when our servlet program gets other than first request:

Servlet container checks the availability of our servlet class object
1) if available, servlet container calls service(-,-) (public) on existing object of our servlet class to process the request and this response goes to browser window.
2) if not available, servlet container preforms all operations of first request

→ **The difference between Httprequest method GET and POST:**

GET

→ Design to gather data from server by generating req.
→ GET send limited amount of data along with the req (max of 256kb)
→ The form page generated qry string will appear in browser's address bar so not data secrecy
→ Not suitable for file uploading operations
→ Can not send data in encrypted format.
→ Use doGet() method or service(-,-) method to process the req.
→ Get is idempotent
→ Default req method of HttpRequest.

POST

→ Design to send data to the server along with the req.

→ It can be send unlimited amount of data along with the req.

→ The form page generated qry string will not appear in browser's address bar. so data secrecy is available.

→ suitable for file uploading.

→ can send data in encrypted format.

→ use doPost() or service(-,-) to process the req.

→ post is not idempotent

→ it is not default and should be applied explicitly

→ It is recommended to design form pages by having post method.

→ hyperlink generated requests are get method based requests where as the form page can generate either get or post method based req.

→ before completing the req processing of given req if client is allowed to generate next req and if the web app is processing both the req then it is called as double posting problem or idempotent problem.

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

→ The first web page of web app that comes automatically when req is given to web appl is called as welcome page or home page of web app.

→ in java based web apps the html or jsp programs can be configured as welcome pages:

web.xml
<welcome-file-list>
<welcome-file>Personal.html</welcome-file>
<welcome-file>Personal.htm</welcome-file>
<welcome-file>Abc.htm</welcome-file>
<welcome-file>Abc.jsp</welcome-file>
</welcome-file-list>
from the above only one file will become as welcome fir here.

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

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

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

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

→ Tomcat 6.0 server allows to configure servlet program as welcome file by specifying its URL pattern.
<welcome-file-list>
<welcome-file> test</welcome-file> test is url pattern of servlet program

</welcome-file-list>

→ Verifying the pattern and format of the form data before it is getting used in business logic as input values is called as form validation and logic written for this is called as form validation logic.
Ex: checking whether required components are filled up with values or not, checking whether age value is given as numeric value or not, checking whether email id is having @, . symbols or not.

→ **Difference between form validation logic and BL:**
BL uses the form data as input values and generates the results where as form validation logic verifies the pattern and format of the form data.

→ **Form validations in web appl:**
1. client side --> use VB script or js code in form page
2. Server side --> write java code in servlet/jsp programs before BL.

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

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

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

→ For **example** appl that performs both client and server side form validations<form action="Vturl" method="POST" onSubmit="return validate(this)">

→ In the above statement the return key word takes the return value of validate function (true or false) and sends to browser window.

→ If that value is "true" (no validation errors) then browser window sends the request to the requested web resource program of web appl. If the value is "false" (validation errors are there) then the browser window blocks/stops req going to server.

→ To hide js code from end users through view source option and to make js code as reusable code for multiple html programs web appl then keep js code in .js file and link with html programs.

personal.html
```
<html>
<head>
<script language="javaScript" src="myfilr.js">
</script>
</head>
<form action="vturl" method="post" onsubmit="return validate(this)">
```

```
</form>

myfile.js

function validate(frm)
{
-
-
-
}
```

Http request methods:
GET(), POST(), HEAD(), DELETE(), PUT(), TRACE() OPTIONS()
GET: given to gather/ to get more data from the server.
→ the response of this method based request contains both headers and body.

HEAD: → same as get but this method based req generated res contains only res headers.
→ this method useful to check the availability of web resource programs.
POST: → design to send unlimited amount of data along with the req
PUT: useful to add new web resource program in web appl from client.
DELETE: useful to remove web resource program of web appl from client.
Note: put(), delete() are useful in the development of FTP applications.

OPTIONS: server determines using which req methods the current web resource program can be req.
NOTE: if our servlet program contains doGet() method overriding then the options req method based req given to that servlet program returns get.Head.Trace, options.
TRACE: A trace req method based req sends the flow of execution details of certain web resource program like servlet. so these details can be used for debugging operations.

Components:
Textbox:
In form page: <input type="text" name="pname">
inservlet prog to read component value : String s1=req.getParameter("pname");

passwordbox:
In form page: <input type="password" name="pwd">
inservlet prog to read component value : String s1=req.getParameter("pwd");

TextArea:
In form page: <input type="textArea" name="address" rows="4" cols="5">
enter address
</textarea>
in servlet prog to read component value : String s1=req.getParameter("address");

SelectBox/ComboBox allows us to select one item at a time
<select name="qlfy">
<option value="be">BE/BTech</option>

```
  <option value="mtech">MTech</option>
  <option value="mca">MCA</option>
  <option value="msc">M.Sc</option>
</select>
```

in servlet prog to read component value : String s1=req.getParameter("qlfy");

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

```
<select name="course" multiple>
  <option value="c">C</option>
  <option value="cpp">C++</option>
  <option value="java">JAVA</option>
  <option value="oracle">Oracle</option>
</select>
```

in servlet prog to read component value : String s[]=req.getParameter("course");

RadioButtons:

In form page: <input type="radio" name="sex" value="male">Male
<input type="radio" name="sex" value="female">Female
in servlet prog to read component value : String s1=req.getParameter("sex");

CheckBoxes:

In form page:<input type="checkbox" name="hb1" value="read">Reading
<input type="checkbox" name="hb1" value="sleep">Sleeping
in servlet prog to read component value : String s[]=req.getParameter("hb1");

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

			<tr>
				<td style="COLOR: #400080; FONT-WEIGHT: bold;">Name:</td>
				<td><input type="text" name="pname"></td>
			</tr>
```

```
		<tr>
			<td style="COLOR: #ff0080; FONT-WEIGHT: bold;">Age:</td>
			<td><input type="password" name="pass"></td>
		</tr>

		<tr>
			<td style="COLOR: #800040; FONT-WEIGHT: bold;">Gender:</td>
			<td style=""><input type="radio" name="g" value="M" checked>Male
				<input type="radio" name="g" value="f">Female</td>
		</tr>

		<tr style=" height : 24px;">
			<td style="COLOR: #004000; FONT-WEIGHT: bold;">Qualification:</td>
			<td><select name="qlfy">
					<option value="engg">BE/BTech</option>
					<option value="medico">MBBS</option>
					<option value="arts">B.A</option>
					<option value="science">B.Sc</option>
			</select></td>
		</tr>

		<tr>
			<td style="COLOR: #8080ff; FONT-WEIGHT: bold;">Courses:</td>
			<td><select name="crs">
					<option value="java">JAVA</option>
					<option value="net">.net</option>
					<option value="oracle">Oracle</option>
					<option value="science">B.Sc</option>
			</select>
		</tr>

		<tr>
			<td style="COLOR: #004080; FONT-WEIGHT: bold;">Hobbies:</td>
			<td><input type="checkbox" name="ch1" value="read" checked />Reading
				   <input type="checkbox" name="ch1" value="roam"
checked />Roaming

				   <input type="checkbox" name="ch1" value="sleep"
				checked />Sleeping   </td>
		</tr>

		<tr>
			<td style="COLOR: #800000; FONT-WEIGHT: bold;">Address:</td>
			<td><textarea name="address" rows="4" cols="20"> Enter
Address:</textarea>
			</td>
		</tr>
		<tr>
			<td colspan="2"><input type="submit" value="getdetails" /></td>
		</tr>
```

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

</body>
</html>

public void doGet(Htt)
{
PrintWriter out=res.getWriter();
res.setContentType("text/html");

//read from data
int no=null


String name=null, gender=null, qlfy=null, courses=null, hobbies=null, address=null;

String name=req.getParameter("pname");
age=Integer.parseInt(req.getParameter("page"));
String gender=req.getParameter("g");
String qlfy=req.getParameter("age");
String courses[]=req.getParameter("crs");
String hobbies[]=req.getParameter("ch1");
String address=req.getParameter("taddress");
}
//display the details as web page content
out.print("<br>Name = " + name);
out.print("<br>Age = " + age);
out.print("<br>Gender = " + gender);
out.print("<br>Address = " + address);
out.print("<br>Qualification = " + qlfy);
out.print("<br>Courses = " + courses);
for(int i=0;i<crs.length;i++)
{
out.print(crs[i]);
}
out.print("<br>Hobbies = " + hobbies);
for(int i=0;i<ch1.length;i++)
{
out.print(ch1[i]);
}
out.close();

}
```

```
connection con=null;
PreparedStatement ps=null;
public void init()
{
try
{
//create jdbc connection object
Class.forName("com.mysql.jdbc.Driver");
con=DriverManager.getConnection("jdbc:mysql://localhost/sun","root","root");
ps=con.prepareStatement("select * from personal where age=?");
}
catach(Exception e)
{
e.printStackTrace();
}
}//init()

public void doGet(Htt)
{
try{
//read from data
int no=Integer.parseInt(req.getParameter("pass");
//write business logic
ps.setInt(1,no);
//execute the qry
ResultSet rs=ps.executeQuery();
//process the resultSet object
String name=null, gender=null, qlfy=null, courses=null, hobbies=null, address=null;
int age;
if(rs.next())
{
name=rs.getString(1);
age=rs.getInt(2);
gender=rs.getString(3);
qlfy=rs.getString(4);
courses=rs.getString(5);
hobbies=rs.getString(6);
address=rs.getString(7);
}
//display the details as web page content
out.print("<br>Name = " + name);
out.print("<br>Age = " + age);
out.print("<br>Gender = " + gender);
out.print("<br>Address = " + address);
out.print("<br>Qualification = " + qlfy);
out.print("<br>Courses = " + courses);
out.print("<br>Hobbies = " + hobbies);
out.close();
rs.close();
```

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

}//doGet()
public void doPost()
{
doGet(req,res);
}//doPost()

public void destroy()
{
try{
if(ps!=null){ps.close()}
}
catch(Exception e)
{
e.printStackTrace();
}
try{
if(con!=null){con.close()}
}
catch(Exception e)
{
e.printStackTrace();
}
}//destroy()
}//class


out.print("<table width=50% border=1>");
out.print("<caption>Result:</caption>");

//Printing col names
ResultSetMetaData rsmd=rs.getMetaData();
int total=rsmd.getColumnCount();
out.print("<tr>");
for(int i=1;i<=total;i++)
{
out.ptintln("<th>"+rsmd.getColumnName(i)+"</th>");
}
out.println("</tr>");

//printing result

while(rs.next())
{
```

```
out.println("<tr><td>"+rs.getString(1)+"</tr></td>"+"<tr><td>"+rs.getInt(2)+"</tr></td>"+"<tr><td>"+rs.getString(3)+"</tr></td>"+"<tr><td>"+rs.getString(4)+"</tr></td>"+"<tr><td>"+rs.getString(5)+"</tr></td>"+"<tr><td>"+rs.getString(6)+"</tr></td>"+"<tr><td>"+rs.getString(7)+"</tr></td>"+);
}
out.print("</table>");
```

index.html
```
<a href="servlt/url">create</a>
```

Differences between Web Server & Application Server

Web Server

→ allows to deploy and execute web appl
→ Developed based on servlet, jsp appl specifications
→ Gives servlet container, JSP container
→ doesn't allow to create domain
→ allows only Http protocol req.
→ Gives min no. of middleware services.
→ suitable for small scale and medium scale web appl.
→ recognizes .war files as appl.
→ Ex: JWS, Tomcat, Resin and etc.

Application Server

→ allows to deploy and execute web applications, EJB components, enterprise applications, resource adapter apps.
→ Developed based on all JEE api specifications(Servlets, Jsp, Ejb, JMS etc)
→ Gives both servlet container, Jsp container
→allows to create domains
→ allows both Http and non-Http(IIOP, T3 and etc) protocol based req.
→ Gives more no. of middleware services.
→ suitable for large scale web appl and for JEE appls.
→ Recognizes .war, .ear,.jar, .rar files as applications.
→ Ex: weblogic, jboss, web sphere, Glass Fish and etc.

Difference between ServletConfig object and ServletContext object:

→ ServletConfig object(cg):

→ it is one for our servlet program/jsp Program
→ServletConfig object means it is the object of servlet container supplied java class implementing

javax.servlet.servletConfig interface.

→ The object is useful to pass additional data to servlet and to read additional details from servlet.

→ This object is useful to read servlet init parameter values from web.xml file of the web appl.

→ ServletContainer creates the servletConfig obj in the instantiation and initialization process of our servlet class object.

→ servlet container destroys our servlet class object in the destruction process of our servlet object.

→ ServletContext object:

→ It is one per web application. so it is called as the global memory of web appl.

→ ServletContext object means it is the object of a java class(container supplier) implementing javax.servlet.ServletContext interface.

→ servlet container creates this object either during deployment of the web appl or during server strat up.

→ Servlet container destroys this object automatically when web appl is un-deploys or reloaded or stopped or when server is stopped /restarted.

→ using this object we can read global init parameters or context parameters from the web.xml file of the web appl.

→ using this object we can know the details of under-laying server like server name, version and the servlet-api version supported by the server.

→ using this object we can get context path of the current web appl and absolute path of the any web resource program in web appl.

→ The data kept in servletContext object is visible and accessible in all servlet, jsp programs of web appl.

→Note: ServletContainer creates ServletConfig object for servlet program only when the class of servlet program is instantiated(object creation).

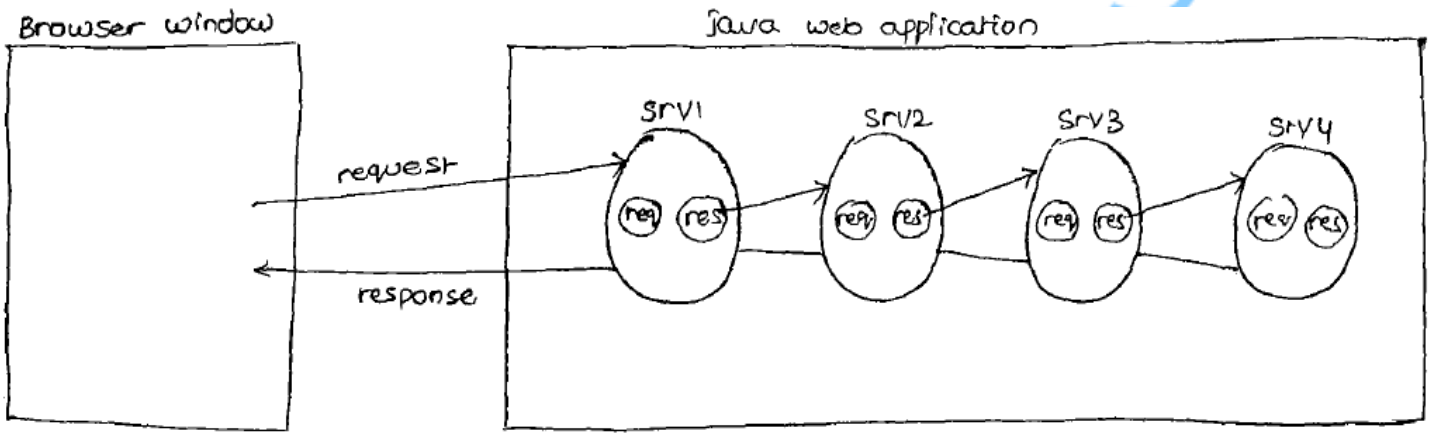
Servlet Chaining:

→ taking req from a browser window and processing that req by using multiple servlets as a chain is called servlt Chaining.

→ in this we perform communication between servlet programs to process the request given by a client.

→ All servlet programs that participate in a servlet chaining will use same request and response objects because they process the same req that is given by client.

→ To perform servlet chaining we need RequestDispatcher object. RequestDispatcher object means it is the object of a container supplied java class implementing javax.servlet.REquestDispatcher interface.

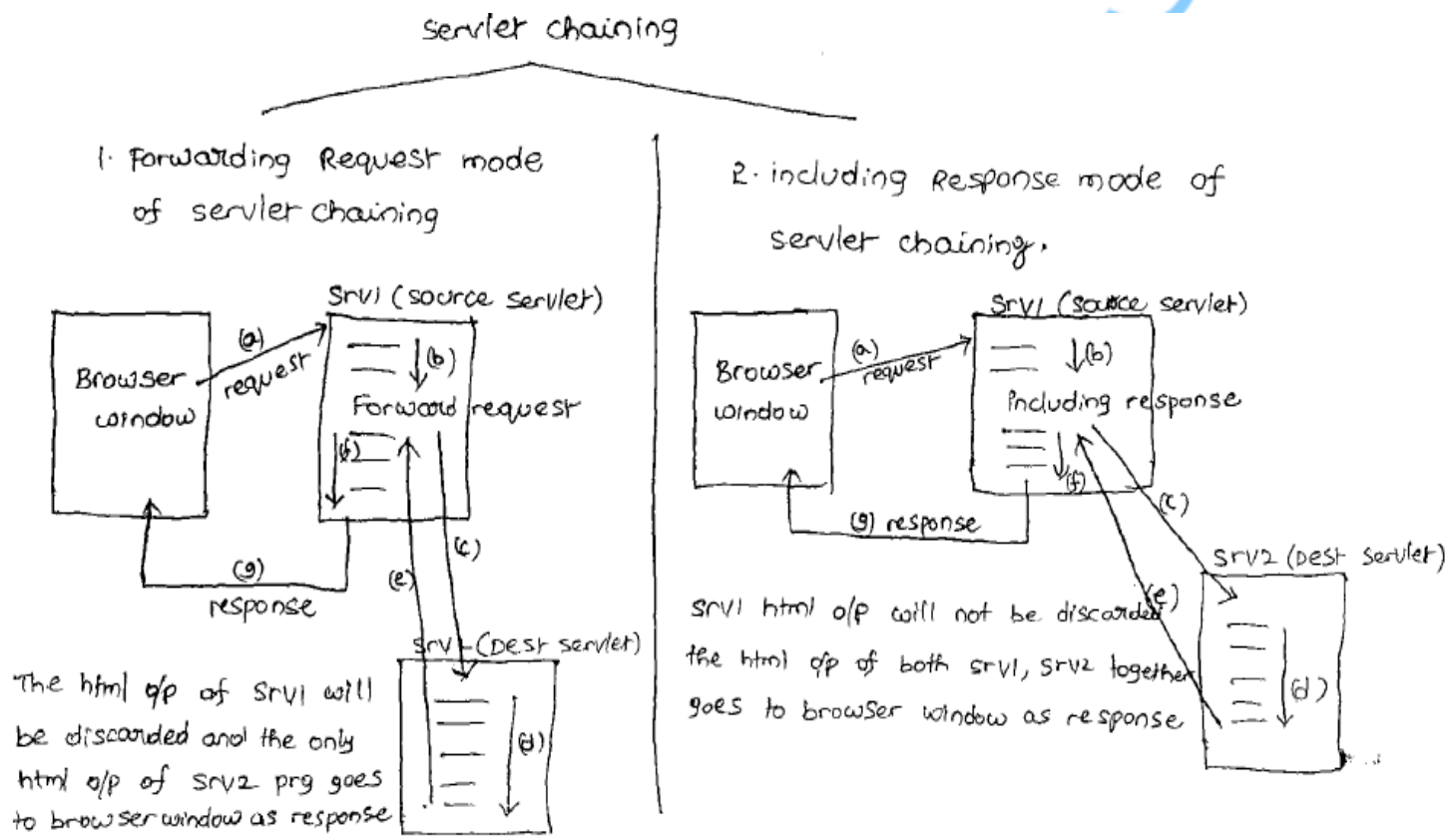


servlet chaining → 1. forwarding Request mode of servlet chaining

the html o/p of srv1 will be discarded and the only html o/p srv2 prog goes to browser window as response

2. including response mode of servlet chaining:

srv1 html o/p will not be discarded the html o/p of both srv1 and srv2 together goes to browser window as response.



The difference between rd.forward() and res.sendRedirect()

→ **rd.forward()**

→ perform forward mode of servlet chaining

→ the source servlet program communicates with destination web resource program directly.

→ the source servlet program and destination web resource program uses same request and response objects so req data coming to source servlet is visible and accessible in destination web resource program.

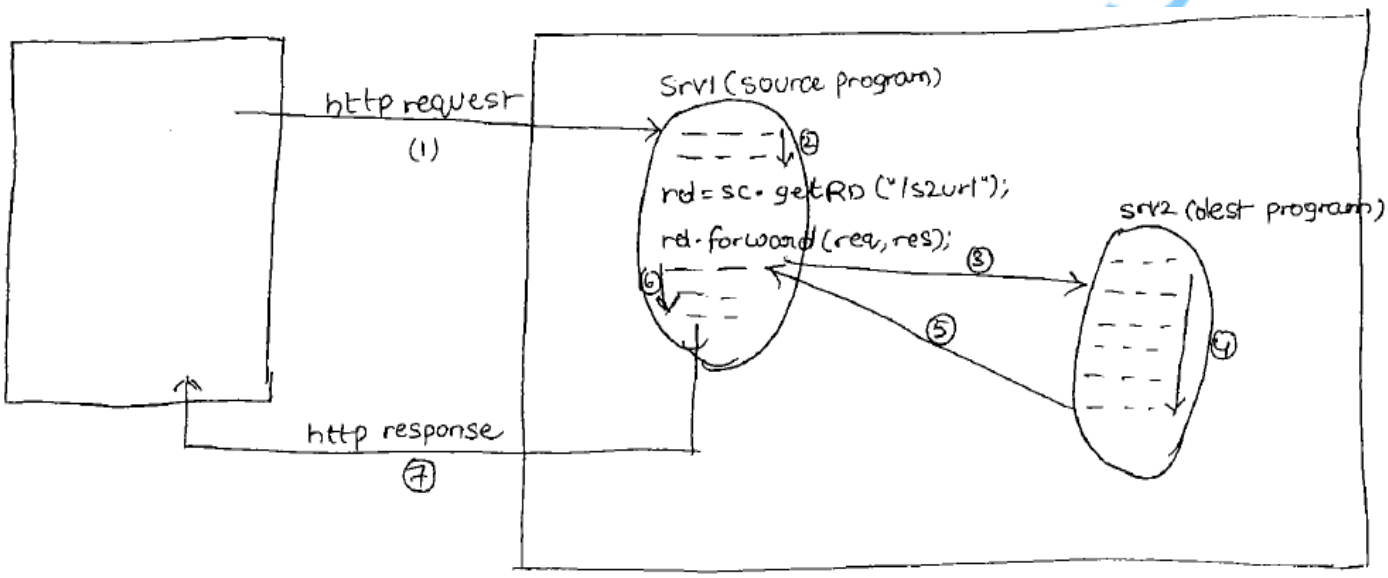
→ source servlet program can use req attributes to send additional data to destination program.

→ source servlet and destination programs can be there in same web appl (any server) or can be there in two different web appls of same server (only in few servers).

→ Destination prog can be a servlet or jsp or html prog.

→ During forwarding req operation url in the browser address bar will not be changed.

→ suitable when source servlet prog and destination servlet prog resides in the same web appl.



→ **res.sendRedirect()**

→ Performs sendRedirection() mode of communication.

→ The source servlet program communicates with destination servlet program by having network round trip with browser window.

→ The source servlet program and destination web resource program will not use same request and response objects so req data coming to source servlet is not visible and accessible in destination webresource program.

→ Append qry string to the url of response.senRedirect() method to send additional data from source servlet prog to destination prog.

→ The source servlet program and destination prog can be there in same web appl or in two different web appls of same server or diff servers.

→ Destination prog can be a servlet or jsp or html or asp or asp.net or php prog etc.

→ During sendRedirection opration the url in the browser address bar will be changed.

→ Suitable when source servlet program and destination program resides in the two different web appl.

→ **Session Attribute:**

→ session attributes allocates memory in http session object. The http session object allocates memory on the server and it is one per browser window so the session attributes of http session object are visible and accessible in all web resource programs of web application irrespective of their req and res objects but they must get req from that browser window for which http session object and its attributes are created.

→ session attributes are global attributes in the web application but they are specific to a browser window(client).

→ To create/access HttpSession obj for browser window on servlet :

→ HttpSession ses=req.getSession();

This method checks the availability of session object on server for browser window, if available it provides access to that session object otherwise (if not available) it creates new session object **(HttpSession object) for browser window on the server.**

```
ses.setAttribute("");
ses.getAttribute();
ses.removeAttribute("");
```

→ **ServletContext obj:**

creates obj in serverlet context

```
ServletContext sc=getServletContext();
sc.setAttribute("attr","new attr");
String s=(String)sc.getAttribute("attr");
sc.removeAttribute("attr");
```

Difference between req parameters and req attributes:

→ req parameters are client supplied input values to web resource program(form data).

→ req attributes are programmer created additional data to pass from one web resource program to another web resource program when they are participating in chaining(like servlet chaining)

→ if source servlet prog and destination program use same request and response objects then use req attributes to send the data.

→ if source servlet program and destination program are getting req from same browser window then we use session attributes for sending data.

→ if source servlet program and destination web-resource program can not satisfy above two conditions then use ServletContext attributes for sending data.

SessionTracking:

Why http is designed as stateless protocol? what is the problem if http comes as statefull protocol?

→If HTTP is state-full protocol for multiple requests given by client to web appl single connection will be used browser window and web server across the multiple requests. This may make clients to engage connections with web server for long time even though the connections are idle. Due to this the web server may reach to maximum connections even though most of its connections are idle.

→To overcome this problem HTTP is given as stateless so no client can engage connection with web server for long time. Moreover the connection will be closed automatically at the end of each request related response generation. In internet environment since there is a chance of having huge amount of clients for each web site it is recommended to have stateless behaviour for http.

→ If web appl is capable of remembering a client data during a session across the multiple requests then that web appl is called as state-full web appl.

→ In state-full web appls the web resource programs can use the data of previous request while processing current req that means while processing request2 it can use request1 data and etc..

→ Even though http is stateless protocol we need to make our web appls as state-full web appls. For this we need the following session tracking or session management techniques.

Session Tracking:
1) Hidden Form Fields
2) Http Cookies
3) Http Session with Cookies
4) Http Session with URL rewriting(recommended)

→ session tracking/session management is all about making web appl as state full web appl by remembering client data across the multiple requests during a session.

→ **Real time implentations of session tracking:**

→ 1) Remembering email id and password during email operations.
--> 2) Remembering username and password during online shopping operations in websites.
→ 3) Remembering previous forms data until last form data arrives in online applications/registrations.
→ 4) while customizing webpage look in website as required for end-user.
→ 5) While rendering direct advertisements in websites and etc..

→ If advertisements are rendered based on the kind of content surfing/browsing done by end user then that advertisement is called as direct advertisement.

Ex: in google.com when we search for the results of seven wonders we will get advertisement related to tours and travels.

1) Hidden Form Fields:

→ An invisible text box of the form page is called as hidden box.

→ When form page is submitted the hidden box value goes to web resource program along with the request as request parameter value.

```
In form page: <input type="hidden" name="t1" value="Hello"/>
In servlet: String s = request.getparameter("t1"); --> gives "Hello" to "s" variable.
```

Advantages:

→ basic knowledge of HTML is enough to work with this technique.

→ Hidden boxes resides in web pages of browser window so they do not provide burden to the server.

→ This technique can be used along with all kinds of server side technologies and all kinds of web servers and application servers.

Disadvantages:

→ Hidden form fields can not store java objects as values. They can just store text values.

→ Hidden boxes doesn't provide data secrecy because their data can be viewed through view source option.

→ Hidden boxes data travels over the network along with request and response so we can say network traffic will be increased.

→ while creating each dynamic form page we need to add the previous form pages data as hidden box values. This increases burden on the programmer.

2) **Http Cookies:**

→ these are the small textual informations which allocate memory at client side by remembering client data across the multiple request during a session.

→ the web resource programs of web applications creates cookies at server side but these cookies comes to client side along with response and allocates memory at client side.

→ cookies of client side goes back to their web application along with the request by clients.

→ **There are two types of cookies:**

1) In memory cookies/per session cookies: allocates memory in browser window having "-1" has the expiry time(no expiry time). These cookies will be destroyed automatically once browser window is closed.

2) Persistent cookies : These cookies allocates memory in the files of client machine hard disk having positive number as expiry time. These cookies will not be destroyed when browser window is closed but they will be destroyed when their expiry time is completed/reached.

Ex: in gmail.com → login page → remember me on this computer option.

→ the cookie that is created without expiry time is called as "in memory cookie".

→the cookie that is created with the expiry time called as "persistent cookie".

→ to work with cookies we must deal with predefined httpServlet class based servlet programs.

→ every cookie contains name and value as String information.

→ At client side or browser window we can see multiple cookies belonging to different domains or web appls. Every cookie remembers the web appl name or domain name for which it is created.

→ The web resource programs of web application send cookies to client machine along with the response as "setCookie" response header values.

→ The browser window sends the cookies back to web application along with the request given to web resource programs as cookies request header value.

Cookie API: working with javax.servlet.http.cookie class

To create cookies and to add them to response:

```
Cookie ck1 = new Cookie("tn","hyd");// creating new cookie
resp.addCookie(ck1); //adding cookie to response
```

ck1 → here ck1 cookie is in memory Cookie because no expiry time is set

Cookie ck2 = new Cookie("mh", "mumbai");

ck2.setMaxAge(1800); → setting expiry time for cookie

resp.addCookie(ck2); → adding cookie to response

→ck2 is persistent cookie having 1800 seconds as expiry time.

To know max age(expiry time) of Cookie

→ int time = ck1.getMaxAge(); → gives -1 for "in mem cookie"

→ int time = ck2.getMaxAge(); → give 1800 seconds

To Modify cookie value:

→ ck1.setValue("Sec-bad");

→ ck2.serValue("Navi Mumbai");

To know domain name/web appl name of the cookie:

→ String d1 = ck1.getDomain();

→ String d2 = ck2.getDomain();

To set comment to Cookie:

→ ck1.setComment("Holds Telangana Capital City");

→ ck2.setComment("Holds Maharastra capital City");

To get Comment:

→ String s1 = ck1.getComment();

→ String s2 = ck2.getComment();

To read Cookie values:

```
String ck[] = req.getCookies(); → reads all the cookies from the current http req.
if(ck!=null)
{
        for(int i=0;i<ck.length;i++)
        {
        pw.println("Cookie Name:"+ck[i].getName());
        pw.print("Cookie Value:"+ck[i].getValue());
        }
}
```

→ cookies can't be deleted programatically.

→ in windows Xp while working with internet explorer the persistant cookies of web application will be stored in the files created in "c:\documents and Settings\administrator\Cookies" folder and the notaion of the file is

<windowsUser>@<Web Application name>[<Cookies count>].txt

Ex: venukumaar@cookieApp[1].txt

Advantages:

→ Cookies allocates memory at client side so they do not give any burden to server.

→ All server side technologies and all web servers, application servers support cookies.

→ persistent cookies can remember client data during and after session wit expiry time.

DisAdvantages:

→ Cookies can not store java objects as values. They can store only Strings.

→ Cookies data can be viewed through browser setting or through the files of file system. so they do not provide data secrecy.

→ Cookies can be deleted explicitly through browser settings it may fail session tracking

→ There is a restriction on number of cookies that can be there in browser window. per browser window and per web application maximum of 20 cookies. All together maximum of 300 cookies per browser window.

→ Cookies can be blocked/restricted coming to browser window. This fails session tracking.

→ To delete cookies

IE : tools → internet options → delete cookies

Netscape: Tools → cookie manager → remove all cookies

→ To block Cookies

IE : internet options → privacy → user slider

Netscape: Tools → Cookie manager → Block cookies from this site

3) Http Session with Cookies:

→ HttpSession object allocates memory on the server and remembers client data across the multiple requests in the form of "session" attribute values.

→ HttpSession object is one per browser window(client) so each HttpSession object can be user to remeber client data during a session.

→ HttpSession object means it is the object of a servlet container supplied java class implementing javax.servlet.http.HttpSession interface.

→Every HttpSession object contains session id and this session id goes to browser window from web appl and comes back to web appl from browser window in the form of "in memory cookie value". So this technique is called "HttpSession with cookies" technique.

→ When HttpSession object is created the session id of that session object automatically goes to browser window as "in memory cookie value".

→ In HttpSession with Cookies session tracking technique based web application the browser window(client) will be identified across the multiple req during a session based on the session id sent by the browser window along with the req.

Session API: working with javax.servlet.http.HttpSession (interface)

→ **To create/locate HttpSession object**

a) HttpSession ses = req.getSession();

→This method creates new HttpSession object in server for browser window if HttpSession object is not already available for browser window otherwise this method provides access to existing HttpSession object of that browser window.

→ This method can create the new session between browser window and web application if session is not already between them otherwise this method makes current request to join in the existing session.

→ If this method is called in FirstServlet program , then it makes request1 of browser window b1 begining new Session between browser window b1 and web application session App. If same method is called SecondSevlet, ThirdSevlet then it makes request2, request3 of browser window b1 participating in existing session.

b) HttpSession ses = req.getSession(false);

→ This method gives access to existing HttpSession object of browser window, if not available this method returns null(indicating new HttpSession Object can not be created)

→ When this method is called the request can always join in existing session but can not create new session between browser window and web appl.

c) HttpSession ses = req.getSession(true);

same as (a)

Conclusion: To create new session/to locate existing session ==> use (a),(c) options
Only to Locate existing session ==> user (b) option.

→**To know session Id:** String id=ses.getId();

→ **To know session object creation time/session started time:** long ms = ses.getCreationTime();

Date d1 = new Date(ms);

In the above code "ms" represents milliseconds that are elapsed from 1970 jan 1st 00:00 hrs to the date and time of HttpSession object creation.

→ **To know the last accessed time of HttpSession object:**

long ms=ses.getLastAccessedTime();
Date d2=new Date(ms);

→ **To get access to servletContext object:** ServletContext sc = ses.getServletContext();

→ **To know whether session is new or not:** boolean b=ses.isNew();

This method returns true when ses object is just created obj and its session id still yet to be delivered(sent) to client (browser window) otherwise this method returns false(when session object is old obj and its session is is already there with client).

→ **To invalidate the session:**

→ invalidating the session is nothing but closing the session between browser window and

web appl. In this process the entire data from session object will be removed and makes session object inactive object, ready for garbage collection.

→ a) when ses.invalidate() is called.

→ b) when browser window is closed.

→ **Note:** Since session id will be stored as in memory cookie value of browser window and that in memory cookie will be destroyed once browser window is closed so the session will be invalidated once browser window is closed.

→ c) when MaxInactiveInterval/SessionIdle → timeout period is completed/reached.

If session object is continuously idle for certain amount of time then it will be invalidated automatically. The default session idle timeout period is 30 minutes(in most of the servers). But this can be changed explicitly either by using programmatic approach or declarative approach.

i) **programatic approach(java code):**

In servlet/JSP prog → ses.setMaxInactiveInterval(1500); seconds

ii) **Declarative approach(xml code):**

```
<web-app>
    <session-config>
        <session-timeout>20</session-timeout> 20 in min
    </session-config>
<web-app>
```

→ once session idel timeout period or maxInactiveInterval period is completed the underlaying webserver automatically expires the session (invalidates the session).

→ **To know current maxInactiveInterval period/SEssionIdletimeout period:**

```
int t1=ses.getMaxInactiveInterval();
```

→ **Note:**The above method returns 30 if no value explicitly set as SessionIdle timeout period.

*** If you set sessionIDletimeout period in both programmatic and declarative approaches with two different values, which value will be effected finally?

→ since the code of servlet program executes after web.xml code so the time specified through programmatic approach will be effected by overriding the time specified through declarative approach.

4) Http Session with URL rewriting(recommended) : The limitation with third technique is if cookies are restricted to coming to browser window then third technique fails to perform session tracking because it uses in memory cookie to send sessionid to browser from web application along with response and to bring session id back to web application from browser window along with request.
==> to overcome that problem work with Httpsession with URL rewriting which does not use cookies to send and receive session id. moreover this technique appends session id to a url that goes to browser window from web application along with the response and that comes back to web application from browser window along with the request. Generally we append session id to the action url (the action attribute of form tag) at dynamic form page generation code.

→ **Note:** res.encodeURL("s2url"); → gives s2url;sessionId:4243541A3F723
this method use URL appended with current object's session id as shown above.

→ by taking above kind of URL as action url of dynamic form page we can perform httpsession with URL rewriting based session tracking.

In servlet Program:

```
pw.println("<form action ="+res.encodeURL("surl")+"method=get>");
pw.println("....");
pw.println("....");
pw.println("</form>");
```

→ res.encodeURL("surl")→ keeps the url appended with sessionid as the action attribute value of <form> tag.

→ **Ex:** application on Httpsession with URL rewriting
Advantages and disadvantages: same as Httpsession with cookies technique but this technique also works even though cookies are restricted coming to browser window.

session tracking conclusion :

→ while developing large scale and commercial web site which contains huge customer base take the support of Http Cookies technique.
Ex: www.gmail.com, yahoo.co.in.

→ while developing medium scale and certain organization based website which contains limited amount of customers use httpsession with url rewriting.
Ex: citibank.com, icici.com

→ we can use multiple session tracking techniques in a single web application development. for ex: the online shopping web sites like www.yebhi.com uses both http cookies(for holding shopping cart information) and httpsession with url rewriting techniques(for finding logging details username, password)

Servlet Filter:

Basics:

→servlet filter program is a java class that implements javax.servlet.Filter interface.

→ every Servlet filter program must be configured in web.xml file using filter, filter mapping tags.

→ To link servlet filter program with servlet program of web application the url pattern of servlet program must be taken as the url pattern of servlet filter program.

→ servlet container creates our servlet filter class object either during server start-up or during the deployment of the web application.

→ we can map servlet filter program with one servlet program, multiple servlet programs or all servlet programs of web application.

→ there are 3 life cycle methods in servlet Filter program.

1) **public void init(FilterConfig fg)** → executes during instantiation and initialization process of our servlet filter program.

2) **public void doFilter(servletRequest req, servletResponse res, Filterchain fc)** → fc points to the target mapped sevlet programs of filter program.
Executes for every request trapping and response trapping.

3) **public void destroy()** → executes when ServletContainer is about to destroy our servlet filter class object.

→ **there are 3 types of filter programs.**

1) request Filter (contains only pre-request processing logics)
Ex: Authentication filter
Authorization filter

2) response filter (contains only post response generation logics)
Ex: signature Filter, Tags Filtering Filter(removes html tags from response of the servlet programs which are not supported by browser window)
Compression Filter(reduces the size of the response content)

3) request-response filter(contains both pre-request processing and post-response generation logics).
Ex: performanceTest Filter (This filter holds request trapping time and response trapping time of certain servlet program and calculates difference between both timings to decide the performance of the servlet program)

→ FilterConfig object is right hand object of our servlet Filter class object and it can be used to read init parameter values of Filter program in web.xml file.

→ init() method of servletFilter program contains initialization logic.

→ doFilter(-,-,-) of ServletFilter program contains pre-request processing logic and post response generation logics.

→ destroy() of servletFilter program contains un-initialization logic.

→ It is a special web resource program of java web application that is capable of trapping and taking request and response of other web resource programs of that web applications.

→ In this we can keep the common and global prerequest processing logic and post response generation logic.

→ To add new functionalities to web application without changing the source code of existing web resource programs we can take the support of servlet filter programs.

with respect to the above diagram:
1) browser window generates request to srv1 servlet program.
2) the servlet filter program traps and takes that req.
3) servlet filter program executes the common and global pre-requesting process logic (like authentication logic).
4) Servlet filter program forwards the req to actual srv1 servlet program.
5) the main req processing logic of srv1 program executes and generate response.
6) Servlet filter program traps and takes the response.
7) filter program executes the common and global post response generation logic (like adding more content at the end of response).
8) servlet filter program sends response to browser window.

Differnce between doFilter(-,-,-) and doFilter(-,-):

→ **doFilter(-,-,-)** is Filter interface(javax.servlet.Filter), it is lifecycle method of ServletFilter program so programmer uses this metod to place prerequest processing and post response generation logics.

→ **doFilter(-,-)** is FilterChain interface(javax.servlet.FilterChain), it is not lifecycle method of ServletFilter program so programmer uses this metod to invoke next filter in the chain or to invoke the mapped servlet program or jsp program of current servlet filter program.

→ A action performed on java object or component is called as an **Event**.

→ Executing some logics when event is raised is called as **EventHandling**.

→ To perform event handling we use "event Listerns".

→ The reusable java object is called as Component. We use AWT, Swing concepts of JSE module to perform event-handling.

→ **To perform event handling we need the following four important details:**

1) source object on which the event will be raised (like Button component)
2) Event class name (like java.awt.event.ActionEvent)
3) Event Listener (like java.awt.event.ActionListener)
4) Event handling method (like public void actionPerformed(-){-----})

→ From **servlet api 2.4** onwards we can perform event handling on request, session and servletContext object through servlet listeners.

→ **Details that are required to perform event handling on request object:**

→ Event-handling on request object allows us to keep track of the creation and destruction of request object and we can use these timings to calculate request processing time of each request. This event handling also allows us to keep track of when each request attribute is created, modified and removed.

→ **for event handling on request object**

source object: request object
Event class: javax.servlet.ServletRequestEvent
Event Listener: java.servlet.SerletRequestListener
Event Handling methods: 1) requestInitialized(-)
2) requestDestroyed(-)

→ **for event handling on request Attributes**

source object: request object
Event class: javax.servlet.ServletRequestAttributeEvent
Event Listener: java.servlet.SerletRequestAttributeListener
Event Handling methods: 1) attributeAdded(-)
2) attributeRemoved(-)
3) attributeReplaced(-)

→ Details that are required to perform event handling on ServletContext object:

→ Event handling on ServletContext object helps us to keep track of when ServletContext object is created and destroyed. Based on this we can keep track of timings of web application deployment and un-deployment or reload or stop operations.

→ This Event handling can also be used to keep track of creation, modification, destruction of ServletContext attributes.

→ for event handling on servletcontext object

Source Object: servletcontext object
Event Class: javax.servlet.ServletContextEvent
Event Listener: java.servlet.SerletContextListener (I)
Event Handling methods: 1) contextInitialized(-)
2) contextDestroyed(-)

→ for event handling on servletcontext Attributes

source object: servletcontext object
Event class: javax.servlet.ServletContextAttributeEvent
Event Listener: java.servlet.ServletContextAttributeListener(I)
Event Handling methods: 1) attributeAdded(-)
2) attributeRemoved(-)
3) attributeReplaced(-)

→ Details that are required to perform event handling on HttpSession object:

→ Event handling on session object can keep track of when httpSession object is created and destroyed. using this we can know how much time the user us there in the session. we can also use this event handling to keep track of when each session attribute is created or modified or destroyed.

→ for event handling on HttpSession object
source object: HttpSession object
Event class: javax.servlet.http.HttpSessionEvent
Event Listener: java.servlet.http.HttpSessionListener (I)
Event Handling methods: 1) sessionCreated(-)
2) sessionDestroyed(-)

→ for event handling on HttpSession Attributes
source object: HttpSession object
Event class: javax.servlet.http.HttpSessionBindingEvent
Event Listener: java.servlet.http.HttpSessionAttributeListener(I)
Event Handling methods: 1) attributeAdded(-)
2) attributeRemoved(-)
3) attributeReplaced(-)

→ Through servlet listeners we can perform event handling on request, servletContext, session objects being from outside the servlet, jsp programs.

→ Thread safety:

→ If multiple threads are running on single object or variable simultaneously or concurrently then data of the object may corrupt. This says out object is not thread safe object.

→ To make object/variable as Thread safe apply lock on object or variable through synchronization and allow only one thread at a time to manipulate object/variable data. The instance variables of our servlet class are not thread safe by default where as the local variables declared in service(-,-) or doXxx(-,-) methods of our servlet class are thread safe by default.

→ by default every servlet program is a single instance multiple threads components so by default no servlet program is thread sage.

→ making servlet program as thread safe program is nothing but making the instance variables of servlet class as thread safe through "synchronization" and other concepts.

→ to make out servlet program as thread safe servlet program we can user one of the following four techniques.

1) work with only local variables of service(-,-)/doXxx(-,-) methods
2) work with synchronized service(-,-)/doXxx(-,-) methods
3) work with synchronized blocks in service(-,-)/doXxx(-,-) methods
4) make our servlet class implementing javax.servlet.SingleThreadModel(I)

1) **work with only local variables :**

```
public class TestSrv extends HttpServlet/GenericServlet
{
	---
	--- no instance variables
	public void service(-,-)/doXxx(-,-) throuws ServletException, IOException
	{
		int a,b,c;
		Connection con;
		---
		write entire logic using local variables.
	}
}
```

→ Local variables of service(-,-)/doXxx(-,) methods are thread safe variables by default. So the above servlet becaomes thread safe servlet. Because if does not have instance variables and working with just local variables.

→ **Note:** This technique is not recommended to use because it is practically impossible to develop servlet classes withoutt instance variables.

2) **work with synchronized service(-,-)/doXxx(-,-) methods**

```
public class TestSrv extends HttpServlet/GenericServlet
{
	// instance variables
	int a,b,c;
	Connection con;
	public synchronized void service(-,-)/doXxx(-,-) throuws ServletException, IOException
	{
```

```
        //write logic using instance variables.
        ---

    }
}
```

→ all requests that are given to servlet starts threads on our servlet class object and executes servic(-,-)/doXxx(-,-).

→ In the above code multiple threads started on our servlet class object will act on single copy of instance variables but only one thread is allowed at a time to act on those instance variables because the service(-,) method is taken as synchronized method.

→. Instead of making whole service(-,) as synchronized method, it is recommended to take the support of synchronized blocks on special objects on which you are looking for thread safety.

3) Working with synchronized blocks in service(-,-)/doXxx(-,-)

```
public class TestSrv extends HttpServlet/GenericServlet
{
    // instance variables
    int a,b,c;
    Connection con;
    public void service(-,-)/doXxx(-,-) throuws ServletException, IOException
    {
        ---
        ---
        synchronized(con obj)
        {
        //logic related to connection object
        }

        synchronized(Session obj)
        {
        //logic related to HttpSession object
        (Creating, modifying, reading and writing, moving attribute values)
        }

        synchronized(ServletContext obj)
        {
        //logic related to HttpSession object
        (Creating, modifying, reading and writing, moving attribute values)
        }

    }
}
```

→ working with these synchronized blocks is the most popular technique of real world to make servlet programs as thread safe.

→ since httpsession object and ServletContext objects are sharable objects in multiple web resource programs of web application it is recommended to use them in synchronized blocks as shown

above.

4) making our servlet class implementing javax.servlet.SingleThreadModel(I):

```
public class TestSrv extends HttpServlet implements SingleThreadModel
{
        // instance variables
        int a,b,c;
        Connection con;
        public  void service(-,-)/doXxx(-,-) throuws ServletException, IOException
        {
                //write logic using instance variables.
                ---
                ---
        }
}
```

→ by seeing the implementation of SingleThreadModel interface the under-laying server automatically makes our servlet as thread safe. But the technique of thread safety that is used by server will vary server to server.

→ This interface has been deprecated from servlet api2.4, because different servers are using different mechanisms while implementing this interface and to make the servlet as thread safe some servers are event creating multiple objects for servlet class for multiple requests on one per request basis. This is violation of servlet specification that says a servlet is a single instance multiple threads component.

→ Every JSP program is not thread safe by default, because the jsp equivalent servlet is not automatically implements SingleThreadModel interface and not automatically generates the necessary synchronized blocks in _jspService(-,-) method.

→ To make Jsp program as thread safe program use <%@page isThreadSafe="false"%> tag in jsp program. the default value of isThreadSafe attribute is "true".

→ **isThreadSafe = "true":** makes Jsp equivalent servlet as non thread safe servlet, where as isThreadSafe = "false" makes jsp and its equivalent servelt as thread safe.

File Down Loading:

→ 1. making the output of web resource program as down-loadable file (**approach1**)

→ 2. making the resource(file) of server machine file system as down-loadable file (**approach2**)

→ in **Approach1** web resource program will not be downloaded. The output of the web resource program as downloadable file through browser window.place the following two lines of code in servlet/jsp program to make its response as downloadable file according to approach1.

response.addHeader("content-disposition","attachment:filename=abc.html"); → filename that holds response of current web response of current web resource program.

response.setContentType("text/html"); → MIMe type

→ special response header to guide webserver towards sending the response of the web resource program

to client.

→ **approach(2) based file downloading application:**

→ **Security in servlets:**

→ security = authentication + authorization

→ checking the identity of a user is called as authentication.

→ checking the access permissions of a user to use certain resources of the application is called as authorization.

Ex: every emp must be authenticated to use banking project, but cashier is authorized to use cash module. Administrator is authorized to use every module.

→ programmers are not responsible to perform network security. these operations will be taken care by network administrator through firewalls.

→ programmers are responsible for application managed security and server/container managed security in web applications.

→ application managed security means it is the security that is developed by programmer manually inside the application.

→ server managed security means it is the authentication and authorization performed by container on web resource programs by using security middleware service.

→ In server managed security the logical context where users, passwords and roles are defined is called as security realm and every server comes with one default security realm called myrealm.

→ In tomcat server this default realm related configurations can be done in <tomcat_home>\conf\tomcat-users.xml file.

→ A role defines access permissions (set of access permissions) instead of specifying access permission for each user separately define a role specifying access permissions and assign that role to users.

→ we can configure server managed security authentication on our web resource program in the following four modes:

1) BASIC
2) DIGEST
3) FORM
4) CLIETN-CERT

BASIC: users base 64 algorithm internally

→ does not encrypt given username and password

→ generates built-in dialog box for end user to submit username and password.

DIGEST:

→ uses MD5 hashing algorithm internally

→ It is stronger than base-64 algorithm

→ remaining all are same as BASIC, but it encrypts given username and password.

FORM:

→ same as BASIC but allows programmer to design his own form page, error page instead of ready made dialog box, ready made errors message.

CLIENT-CERT:

→ Works with digital certificates using same algorithms like RSA.

→ use https and SSL concepts to implement this.

→ To configure security realm and authentication models in the web resource programs of java web application use various configurations in web.xml file.

→ **Procedure to create users and roles in default security realm(myrealm) of tomcat server:**

goto <tomcat_home>\conf\tomcat-users.xml → add these entries:
<user username = "guru" password = "guru" roles = "admin"/>
<user username = "guru1" password = "guru1" roles = "manager"/>
<user username = "guru2" password = "guru2" roles = "manager, admin"/>

→ uncomment existing roles and users

→ this server managed security can be used as outer layer security for web application even though web application contains application level security. This is very useful in military based, NASA based, ISRO based web applications.

→ The client-cert authentication model sends digital certificates to browser window as the response of initial request and that digital certificate will be installed at client side with users permission. Now onwards browswe window and server communicates with each other having encrypted data that is generated the algorithm that is used to generate digital certificate. In this browser window(client) and server should communicate with each other by using HTTPS protocol (HTTP over SSL).

→ In CLIENT-CERT authentication model server allows only those, clients which sends request having digital certificate.

→ To make server sending digital certificate to browser as the response of initial request, the programmers must develop and configure digital certificate with server.

→ In java on built-in tool "key tool" is given to generate digital certificates using different algorithms.

→ procedure to create digital certificate using "key tool" by specifying RSA algorithm

> keytool -genkey -alias tomcat1 -keyalg RSA → tomcat1 any name(logical name of digital certificate)

Enter key store password: guru1
Re-enter new password : guru1
.
.
.
correct?
yes
Enter key password for<tomcat1>:(press Enter key)

Step2: The above steps based digital certificate will be generated inc:\documents and setting\welcome folder as .key store file.

→ currently logged in windows username

Step3: configure the above generated digital certificate with tomcat server by enable HTTPS protocol.

Goto <tomcate_home> \conf\server.xml file → un comment that <connector> to that points to SSL HTTP/1.1 protocol ==. make sure that the <connector> tag having following content

```
<connector port="8443" protocol="HTTP/1.1" SSLEnabled="true"
        maxThreads="150" scheme="https" secure="true" clientAuth = "false"
sslProtocol="TLS" keystoreFile="c:\Documents and Settrings\welcome E:\.keys"
        keystorePass="guru1"/>
```

step4: Restart the server
step5: Give request to tomcat server from browser window
https://localhost:8443 ==> port no. of http with ssl environment
https://localhost:8443/WAOne/input.html

→ for related information on SSL refer the ssl chaper of tomcat documentation

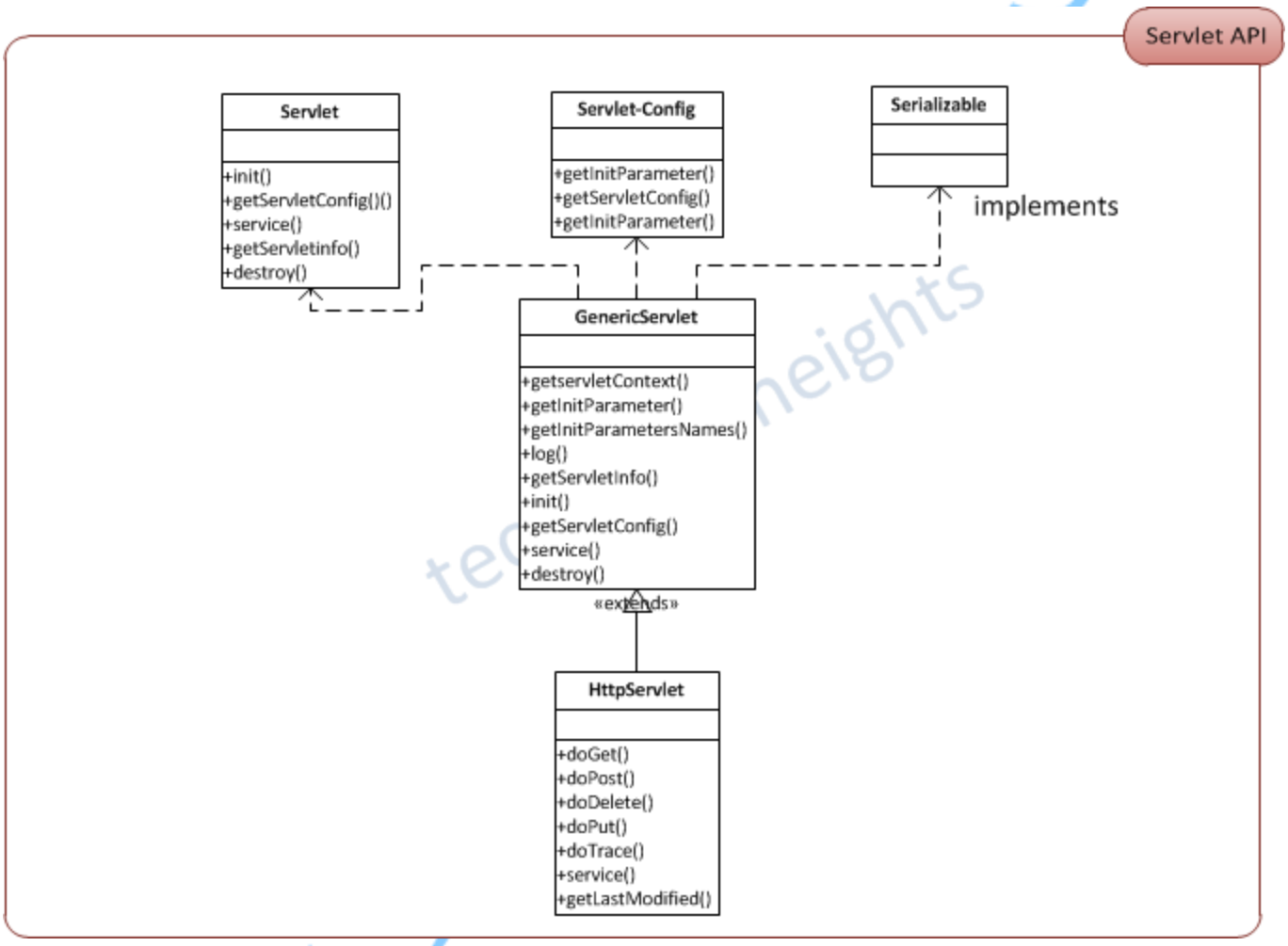
→ **NoTE:** Industry prefers working with either form based or CLIENT-CERT models for securing web applications.

When CLIENT-CERT is enabled we can even also enable basic or DIGEST or FORM authentication models.

Servlet:

Advantages of Servlet:-

- Each request is run in a separate thread, so servlet request processing is significantly faster than traditional CGI processing.
- Servlet is scalable.
- Servlets are robust and Object Oriented.
- Servlets are platform independent.
- The Web-container provides additional services to the servlets, such as error handling & security.



It defines Life Cycle methods that all Servlets must implement.

Method	Description
`void init(ServletConfig config) throws ServletException`	Called the when Servlet is initialized or created. Web Container calls the init method exactly once after instantiating the Servlet. An UnavailableException is thrown if the servlet cannot be initialized.
`public ServletConfig getServletConfig()`	Returns **ServletConfig** object.
`public void service(ServletRequest req, ServletResponse res) throws ServletException, IOException`	Called by the Web Container to allow the servlet to respond to a request. Parameters: Object of ServletRequest(req) that contains the client's request and Object of ServletResponse(res)that contains the servlet's response. An IOException is thrown if IO problem occurs.
`String getServletInfo()`	Returns a String containing information about Servlet.

Method	Description
`public void destroy()`	Called when the servlet is unloaded (being taken out of service method). Happens during web-container shut down.

ServletConfig:-

The ServletConfig interface is implemented by the server. It allows a servlet to get information about Configuration during it's initialization.

Method	Description
`String getInitParameter(String name)`	Returns a String which contains the value of the initialization parameter and name which defines names of initialization parameter.
`Enumeration getInitParameterNames()`	Returns Enumeration of String objects containing the names of all initialization parameters.
`public ServletContext getServletContext()`	Returns the reference of ServletContext.

Generic Servlet:-

It is abstract class.

GenericServlet implements the Servlet, ServletConfig, Serializable interfaces.

It implements some lifecycle methods.

GenericServlet may be directly extended by a servlet, Even though it's more common to extend a protocol-specific subclass such as HttpServlet.

Method	Description
`void log(String s)`	It writes the specified message to a servlet log file.
`void log(String s, Throwable e)`	It writes an explanatory message and a stack trace for a given Throwable exception to the servlet log file.

Http Servlet:-

It is an abstract class and extends the Generic Servlet. To create your own Servlet, you need to subclass HttpServlet.
A subclass of HttpServlet must override at least one methods listed below:

Method	Description
`protected void doGet( HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException`	Called by the server through service() to allow a servlet to handle a GET request. Parameters: Object HttpServletRequest(req) that contains the request the client has made of the servlet. Object of HttpServletResponse(resp) that contains the response the servlet sends to the client
`protected void doPost( HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException`	Called by the server through service() to allow a servlet to handle a POST request. Parameters: Object of HttpServletRequest(req) that contains the request the client has made of the servlet. Object of HttpServletResponse that contains the response the servlet sends to the client.
`protected long getLastModified( HttpServletRequest req)`	Returns last modified time in miliseconds.
`protected void doHead( HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException`	
`protected void doPut( HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException.`	Called by the server to allow a servlet to handle a PUT requests. Parametrs: Object of HttpServletRequest(req)that contains the request the client made of the servlet. Object of HttpServletResponse(resp) that contains the response the servlet returns to the client.
`protected void doTrace( HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException`	Called by the server to allow a servlet to handle a TRACE request.
`protected void service( HttpServletRequest req, HttpServletResponse resp)`	Receives standard HTTP and responds by calling doXXX() method defined in the servlet.

Method	Description
`protected void doGet( HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException` `throws ServletException, IOException`	Called by the server through service() to allow a servlet to handle a GET request. Parameters: Object HttpServletRequest(req) that contains the request the client has made of the servlet. Object of HttpServletResponse(resp) that contains the response the servlet sends to the client
`doOptions protected void doOptions( HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException`	Called by the server to allow a servlet to handle a OPTIONS request.

HttpServlet	GenericServlet
HttpServlet defines a HTTP protocol specific servlet.	GenericServlet defines a generic, protocol-independent servlet.
It inherit generic servlet class.	It implements servlet interface.
It use doGet and doPost method instead of service.	It use service method.

HttpServletRequest is Interface use to reterive information from the client for HTTP servlet.

HttpServletRequest methods

Method	Description
`String a=getParameter("userid");`	Return value of a request Parameter as a String ,or null if parameter does not exist.
`String hobbies[]= request.getParameterValues("hobby");`	Returns an array of String Objects Containing all of values the given request has,or null if parameter does not exist
`Enumeration e= request.getParameterNames();`	Returns an Enumeration of string Objects Containing the names of parameters contained in this request. If the request has no parameter,the methods returns an empty Enumeration.
`String a=request.getProtocol();`	Returns the names and version of protocolthe request uses in form of protocol/major Version,minor Version,for e.g HTTP//1.1
`String header= request.getHeader(String header`	Returns the value of specified request header as a string.if the request did not include header of specified,this method returns

Method	Description
name);	null.the Header is case-insensitive.
Enumeration e=request.getHeaderNames();	Returns all the values of specified request header as an enumeration of String Objects.
String a=request.getMethod();	returns the name of HTTP method with which this request was made,e.g GET, POST etc.
HttpSession session =request.getSession();	Returns the current session associated with this request,or if the request does not have session,creates one.
request.setAttribute(String name,Object o);	Stores an attribute in this request.Attribute are reset between requests.this method is most often used in conjuction with RequestDispatcher.
Request.getAttribute(String name);	Returns the value of named attribute as an Object,or null if no attribute of given name exists.

HttpServletResponse

HttpServletResponse Interface used to send response to the client.

HttpServletResponse methods

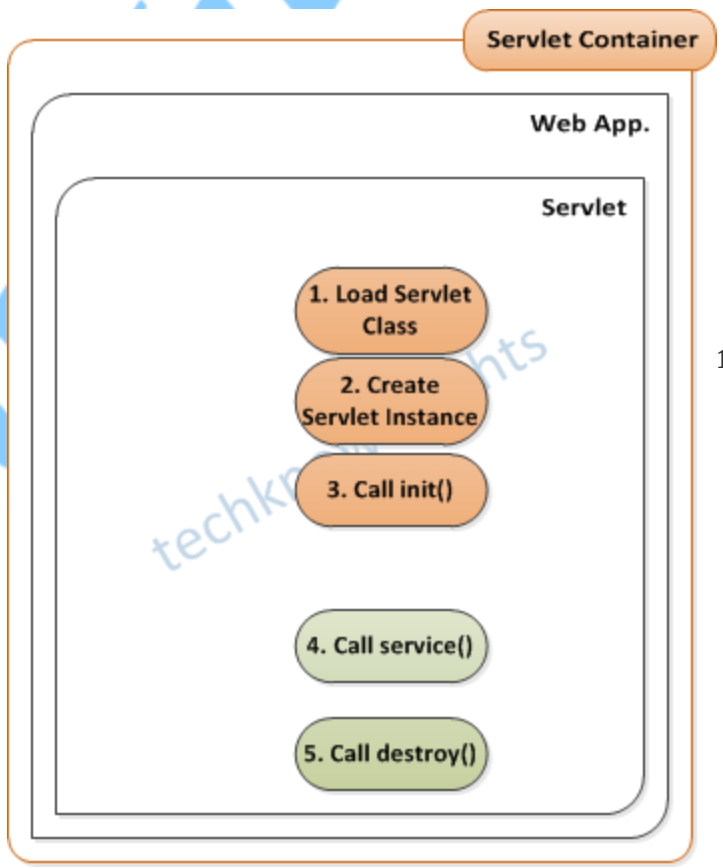
Method	Description
PrintWriter pw=response.getWriter();	Returns a PrintWriter object that can send character text to client.
response.setContentType("text/html");	Sets the content type of response being sent to client.
response.sendRedirect("www.google.com");	Sends a temporary redirect response to client using the specified redirect location url.
response.setHeader(String Header Name,Header Value); response.setHeader("refresh", "1");	Set the response Header with given name and value.If Header had already been set,the new value overwrite the previous one.
response.encodeURL(String url);	Encodes the specified URL by including the session ID in it,or,if encoding is not needed,returns the URL unchanged.

Method	Description
	The implementation of the method includes the logic to determine whether the session ID needs to be encoded in the URL. e.g-if browser supports cookie,or session tracking is turned off,URL encoding is unnecessary.

Servlet Life Cycle

The **life cycle** contains the following steps:

- Load Servlet Class.
- Create Instance of Servlet.
- Call the servlets init() method.
- Call the servlets service() method.
- Call the servlets destroy() method.



1. init() method:-

- The servlet is initialized by calling the init () method.
- The init method is called only one time,when servlet is first created,never call again for each user request.
- The init() method simply create or load some data that will be used overall life of the servlet.

```
public void init() throws ServletException
{
//initialize code....
}
```

2. service()method:-

- The servlet container calls the service() method to handle requests coming from the client or browsers and to write the formatted response back to the client.
- The service() method checks the HTTP request type such as GET ,POST etc and calls doGet, doPost etc.

```
public void service(ServletRequest request,
ServletResponse response) throws ServletException, IOException
{
}
```

doGet() Method:-

- GET is HTTP method. it works to tell server to get a resource and send it server.
- it should be handled by doGet() method.

```
public void doGet(HttpServletRequest request, HttpServletResponse response)
throws ServletException, IOException
{
// Servlet code
}
```

The doPost() Method:-

- POST ,you can request something and at same time send form data to server.
- it should be handled by doPost() method.

```
public void doPost(HttpServletRequest request,HttpServletResponse response)
```

```
throws ServletException, IOException
{
// Servlet code

}
```

3. destroy() method :-

- The destroy() method is called only one time at the end of the life cycle of a servlet.
- It gives opportunity to clean up any resource such as Memory,thread etc.

```
public void destroy()
{
// Finalization code...
}
```

sendRedirect() method sends the request with the new URL, which connects to this URL.

- Every time it creates a new request or response object.

```
response.sendRedirect(www.jspts.com);
```

Inter-Servlet Communication

- Servlets that are present in the same web-server can communicate with each other and can also share resources, such as variables amongst each other.
- Communication between servlets can be implemented by using the RequestDisptacher interface.

RequestDispatcher:

RequestDispatcher object can forward a client's request to a resource or include the resource itself in the response back to the client. A resource can be another servlet, or an HTML file, or a JSP file, etc. There are two ways to delegate a request:-

1. **Forward**
2. **Include**

Forward	**Include**
Used to forward a request to one Servlet to another Servlet.This method must be used when output is generated by second Servlet If PrintWriter object is accessed by the first servlet already,then exception is thrown by this method	This method is used to invoke one servlet from the another servlet like the forward method. However,you can also include the output of first servlet with current servlet. The first servlet can make use of the PrintWriter Object even after calling the include method.
ServletContext sc= this.getServletConfig.getServletContext(); RequestDispatcher rd=sc.getRequestDispatcher("/SecondServlet"); rd.forward(response,request);	ServletContext sc= getServletContext(); RequestDispatcher rd=sc.getRequestDispatcher("/SecondServlet"); rd.include(response,request);

ServletConfig vs ServletContext

ServletConfig	**ServletContext**
One Servlet Config Object per Servlet	One Servlet Context per Web-Application.
Use it to pass deploy the Information to Servlet e.g-(A DataBase name or EJB look up name).	Use it to access Web-application parameter (it has setAttribute/getAttribute method).ServletContext also configure in Deployment Descriptor.
Use it access the servletContext	Use it as a kind of application bulletin-board,where you can put up messages or use it to get the server information including the name version of container
ServletConfig sc=this.getServletConfig();	ServletContext sc=this.getServletConfig. getServletContext();
String a=sc.getInitParameter("email");	String a=sc.getInitParameter("Companyemail");

ServletConfig	ServletContext
Entry in web.xml:- <servlet> <servlet-name>LoginServlet</servlet-name> <sevlet-class>LoginServlet</sevlet-class> <init-param> <param-name>**email**</param-name> <param-value>**abc@yahoo.com**</param-value> </init-param> </servlet>	**Entry in web.xml:-** <web-apps> <context-param> <param-name>**companyemail**</param-name> <param-value>**efg@rkgit.com**</param-value> </context-param> </web-apps>
No setAttribute and getAttribute in ServletConfig.	ServletContext has setAttribute(String name,Object o)to set value during coding and getAttribute to reterive the set value.

Session Management

Session Introduction

- HTTP is a stateless protocol ,where each request and response both is independent of other web interactions.
- it is necessary to save state information so that information can be collected from several interactions between a browser and a server. Sessions provide such a mechanism.
- A new session is created if one does not already exist.
- It is required to implement session management capabilities that link both the authentication and access control modules commonly available in web applications.

Approaches to Session-Tracking:-

1. Session API
2. URL -rewriting
3. Cookies
4. Hidden Form Field

Session API(HttpSession interface)

Java Servlet API provides an interface called HttpSession that can be used to keep track of sessions in the Current servlet context.

Method	Description
HttpSession s=request.getSession()	This method is use to reterive the current the HttpSession that is associated with user. if session does not exist,then a session can be created by using getSession(true)
boolean b=s.IsNew();	returns the Value true If the new Session ID has been created and has not been sent the client.
s.invalidate()	Returns nothing and it is used to destroy the existing session.
long l=s.getCreationTime();	This function returns the time when the session was created in miliseconds.
long l=s.getLastAccessedTime();	This function returns the previous time a request was made with same sessionId.
s.setAttribute("userid",a)	Used to set session attribute in session Object.
Object o=s.getAtribute("userid");	Used t retrieve the set Attribute value.

Setting Session timeout:

You don't have to use them to get rid of stale (inactive) sessions.
The container can do it for you.

Three ways a session can die:

1. It times out.
2. You call invalidate() on the session object.
3. The application goes down (Crashes or is undeployed).

Configure Session Time out in DD:

```
<session-config>
<session-timeout>15</session-timeout>//15 min
</session-config>
or
session.setMaxInactiveInterval(20*60); //20 minute
```

URL-rewritting

- If Client Cookie is disable so Session API fails .
- If client won't take cookies, you can use URL rewriting as a back up.
- URL rewriting is a better way to maintain sessions when the browsers don't support cookie So URL rewriting is a better way to maintain sessions
- Now put the session ID with URL.
- It likes `http://abc.com/email.do;jsessionid="0AAB678C99D1E415",`

response.encodeURL("/WelcomeServlet"); response.encodeRedirectURL("/WelcomeServlet");

Cookies

What is Cookie?

- Cookies are small text files that are used by a Web server to keep track of users.
- A cookie has value in the form of key-value pairs.
- They are created by the server and sent to the client with the HTTP response headers.
- javax.servlet.http.Cookie class is used to represent a cookie.
- **To create cookie object:**

  Cookie ck=new Cookie("key","value");
- **To add cookie to browser:**

  Response.addCookie(ck); // addCookie method takes cookie objects as argument
- **To get cookie:**

  Cookie ck[]=request.getCookies();
  - A server can send one or more cookies to the client.
  - A web-browser, which is the client software, is expected to support 20 cookies per host and the size of each cookie can be a maximum of 4 bytes each.

Cookie Method	Description
Cookie c=new Cookie("userpref","red");	Creating a cookie Object.
c.setMaxAge(int);	This method is used to specify the maximum amount of time for which the client browser retain the cookie value.
HttpServletResponse.addCookie(c);	To send cookie client.
Cookie c[]=HttpServletResponse.getCookies.	Retrieve all cookies.
c.getName()	To retrieve cookie name.

Using Hidden Form Fields

- Hidden form fields is simplest session tracking techniques.
- It can be used to keep track of users by placing hidden fields in a form.
- The values that have been entered in these fields are sent to the server when the user submits the form.
- **For Example:**

```
<input type="hidden" name="text1" value=20>
```

Attributes Types in Servlet

Request Attributes:-

- Request Attribute is Object which is associated with request.
- Attributes mainly use to communicate by servlet to another servlet.
- Methods of the ServletRequest Interface are accessed by Attributes which are-

1. setAttribute
2. getAttribute
3. getAttributeNames
4. removeAttribute

setAttribute:-It stores attribute in current(this)request. Between Requests,Attribute are reseted.

getAttribute:-Returns Object which consist the value of the named attribute.

getAttributeNames:-Return Enumeration which contain the names of the attributes available to this request.

RemoveAttribute:- Removes Attribute from current(this) request.

Session Attributes

- When you are creating and maintaining session for a client So it is important to use.

- If you want to identify the user, to set and get attributes into the session.
- Session Attribute implement by HttpSession interface which give some methods like:-

setAttribute:-

- It is use to create or binds the object with session with specified name.
- If name of an object is same name so object already bound to the session then object is replaced.

getAttribute:-Returns that object which is bound with specified name to this session.

getAttributeNames:-Returns names of all the objects bound to this session.

removeAttribute:-Remove that object which is bound with specified name to this session.

Context Attributes:-A servlet binds an object attribute into the context by name which is available to another servlet and part of same Web Application.

1. setAttribute
2. getAttribute
3. getAttributeNames
4. removeAttribute

   setAttribute:-Create and bind an object to a given attribute name in this servlet context.

   getAttribute:-Returns the attribute of Servlet Container with the given name.

   getAttributeNames:-Returns the attribute names available within this servlet context.

   removeAttribute:-Removes the attribute from the servlet context with the given name .

Filters

- Servlet filters were first introduced in the Servlet 2.3 Specification.

- It used for some additional processing and transformation of data present in Request and Response Objects.
- Filters allow you to add for performing some business processing before or after the servlet in Java classes.
- It is easily done without changing any Servlet Code by only small modification of deployment descriptor element placed in Web.xml file

The package **javax.servlet** provides three Interfaces:-

1. Filter.
2. FilterConfig.
3. FilterChain.

Filter

If You want to create filter so you write Java Class that implement **javax.servlet.Filter**.

Methods	Description
`doFilter(ServletRequest req, ServletResponse res, FilterChain chain)`	Calls by Container when Client request a servlet that has this particular filter placed in web.xml. If to pass these objects to the next filter in the chain, use third parameter.
`init(FilterConfig filtConfig)`	Call by Container When filter is loaded.
`destroy()`	Calls when the filter is unloaded.

FilterConfig:-

FilterConfig provides an access to the initialization parameters of the servlet and to the object ServletContext.

Method	Description
`String getInitParameter(String name)`	Returns a String that has value of the named initialization parameter.**name**id is specify a parameter name.
`Enumeration getInitParameterNames()`	Returns the names of the filter's initialization parameters
`String getFilterName()`	Returns the filter-name of this filter,defined in Web.xml.

Method	Description
`ServlteContext getServletContext()`	Returns Object of Servlet Context used to interact with its Container.

FilterChain:-

FilterChain use to invoke the next filter upto last filter in chain and this method sends the output back to the user's Web browser.

Method	Description
`public void doFilter( ServletRequest req, ServletResponse resp) throws IOException, ServletException`	Object of ServletRequest(req) to pass along the chain. Object of ServletResponse(resp)response to pass along the chain.

Listeners

`

- In Servlet 2.3,It is part of the Java Servlet as defined but they have their own specific functionalities.
- Listeners are defined as Java interfaces.
- Listeners achieve some tasks during Life Cycle of Application for Developers.
- Listener gives us a great control over application without disturbing architecture of the application.

Servlet Listener Interfaces

Package:-javax.Servlet
Interface Name:-ServletContextListener.
contextInitialized():- Creates Servlet Context.
contextDestroyed():- Destroy Servlet Context.

Package:-javax.Servlet
Interface Name:-ServletContextAttributeListener
attributeAdded:-Add the Attribute of Servlet Context Object.
attributeReplaced:-Replacement the Attribute of Servlet Context Object.
attributeRemoved:-Removal the Attribute of Servlet Context Object.

Package:-javax.Servlet.http
Interface Name:-HttpSessionListener
sessionCreated():-Create and invalidate the Session.
sessionDestroyed():-Timeout the session

Package: javax.Servlet.http
Interface:-HttpSessionAttributeListener.
attributeAdded():-Adds the Session Attribute.
attributeReplaced():-Replacement of Session Attribute.
attributeRemoved() Remove the Session Attribute.

Servlet Listener Event Classes

Class	Methods
ServletContextListener	getServletContext()
ServletContextAttributeEvent extends ServletContextEvent	getName() getValue()
HttpSessionEvent	getSession()
HttpSessionBindingEvent extends HttpSessionEvent	getName() getValue()

ConfigDemo.java

```
package com.ducat.servlet;

import java.io.IOException;

import java.io.PrintWriter;

import javax.servlet.ServletConfig;

import javax.servlet.ServletContext;

import javax.servlet.ServletException;

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

```
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 * Servlet implementation class ConfigDemo
 */
public class ConfigDemo extends HttpServlet
{
	public void init(ServletConfig config) throws ServletException
	{
		super.init(config);
	}
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
	{
		ServletConfig sc = this.getServletConfig();
		String jdbcDriverName =sc.getInitParameter("jdbc");
		String dsnName = sc.getInitParameter("dsnName");
		PrintWriter pw = response.getWriter();
		pw.println("Inside ConfigDemo Servlet ");
		pw.println("JDBC Driver Name "+jdbcDriverName);
		pw.println("DSN Name "+dsnName);
		ServletContext ss = sc.getServletContext();
		pw.println("Email is"+ss.getInitParameter("email"));
		pw.close();
	}
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
```

```
	{}
}
```

ConfigServlet2.java

```
package com.ducat.servlet;

import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
/**
 * Servlet implementation class ConfigServlet2
 */
public class ConfigServlet2 extends HttpServlet
{
	public void init(ServletConfig config) throws ServletException
	{
		super.init(config);
	}
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
	{
		ServletConfig sc = this.getServletConfig();
		String jdbcDriverName =sc.getInitParameter("jdbc");
```

```
		String dsnName = sc.getInitParameter("dsnName");
		PrintWriter pw = response.getWriter();
		pw.println("Inside Config Servlet2");
		pw.println("JDBC Driver Name:-"+jdbcDriverName);
		pw.println("DSN Name :-"+dsnName);
		pw.println("Email is:-
"+this.getServletConfig().getServletContext().getInitParameter("email"));
		pw.close();
	}
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
ServletException, IOException
	{
		// TODO Auto-generated method stub
	}
}
```

web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app id="WebApp_ID" version="2.4" xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee http://java.sun.com/xml/ns/j2ee/web-
app_2_4.xsd">
<context-param>
<param-name>email</param-name>
<param-value>shivank@yahoo.com</param-value>
</context-param>
	<servlet>
```

```
        <servlet-name>ConfigDemo</servlet-name>
        <servlet-class>com.ducat.servlet.ConfigDemo</servlet-class>
        <init-param>
        <param-name>jdbc</param-name>
        <param-value>sun.jdbc.odbc.JdbcOdbcDriver</param-value>
        </init-param>
        <init-param>
        <param-name>dsnName</param-name>
        <param-value>FirstDSN</param-value>
        </init-param>
    </servlet>
    <servlet>
    <servlet-name>ConfigServlet2</servlet-name>
    <servlet-class>com.ducat.servlet.ConfigServlet2</servlet-class>
    </servlet>
    <servlet-mapping>
    <servlet-name>ConfigDemo</servlet-name>
    <url-pattern>/ConfigDemo</url-pattern>
    </servlet-mapping>
    <servlet-mapping>
    <servlet-name>ConfigServlet2</servlet-name>
    <url-pattern>/ConfigServlet2</url-pattern>
    </servlet-mapping>
</web-app>
```

output:- Servlet Confing and context