Server:

Server is a software which manages all the resources along with which process the client request and serve the client request .

There are different types of Servers present namely

1.DataBase Server

2.Application Server

3.Web Server

1.DataBase Server:

DataBase Server is used to deal with the data.

Ex:Oracle,MySQL,MS-SQL,Derby,Mongo DB,Sybase,etc.

2.Application Server:

Application Server is used to execute a dynamic application or a realtime application.

Ex:JBOSS,IBMWebsphere,OracleWebLogic,etc

Dynamic Application/Realtime Application:An application which performs all the 3 different types of logics such as Presentation Logic ,Persistence Logic and Business Logic is known as Dynamic Application or a realtime application.

Ex:FaceBook,Whatsapp,etc..

3.Web Server:

Web Server is used to execute only web applications.

Ex:Apache-Tomcat Server and OracleGlassFish.

Deployment

*Making all the resources available to the Server is known as deployment.

There are 2 different types of deployment present namely

Apache-Tomcat Server comes with 2 different variants

1.zip

2.exe

Port Number

Port Number is the one which helps us to get connected to a particular Server.

Default Port Number for Apache-Tomcat Server is-----8080

Basic Steps to be followed in Installation of Apache-Tomcat Server

There are 3 different specifications for Apache-Tomcat Server

a)JAVA_HOME:

Open C-Drive,Program files and open Java and JDK folder and copy the entire path which includes all the folders related to jdk into the Environment Variables.

b)CATALINA_HOME:

Open Apache-Tomcat server folder from the respective directory and copy the entire path which includes all the folders related to Apache-Tomcat Server.

c)Path:

Open the Apache-Tomcat Server folder from the respective directory and copy the entire path which includes bin folder of Apache-Tomcat Server.

d)JRE_HOME(Optional):

Open C-Drive,Program Files and open Java and JRE Folder and copy the entire path which includes all the folders related to JRE.

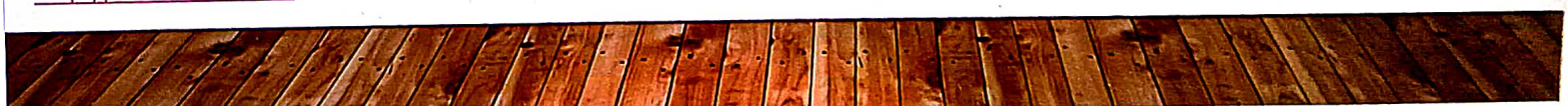
To start the Apache-Tomcat Server:

Go to the respective directory where the apache-Tomcat server folder is present ----open bin folder ----double click on startup(Windows Batch File).

To see Apache-Tomcat Server Page

Go to any of the browser and type below url

http://localhost:8080



How to change the Port Number:

Port Number for Apache-Tomcat Server is 8080
Which can be changed.

Goto the Apache-Tomcat Server folder ---open conf folder---right click on server.xml---open with Note pad—Hit Cntl+F(Find)---type (conn)— and hit on FindNext for exactly 8 times ----change the port number –8080 to 8050 ---save

Goto bin folder of Apache-Tomcat server—start the server---

Goto any of the browser –type –http://localhost:8050

Folders of Apache-Tomcat Server

There are different folders of Apache-Tomcat are present namely

1.bin

2.conf

3.lib

4.logs

5.webapps

6.work

1.bin:

It contains a set of Startup and Shutdown batch files which are used to start and stop the Apache-Tomcat Server.

2.conf:

It contains set of Configurations with respect to Apache-Tomcat Server.

3.lib:

It contains set of libraries in the form of jar which is used to perform some additional functionalities.

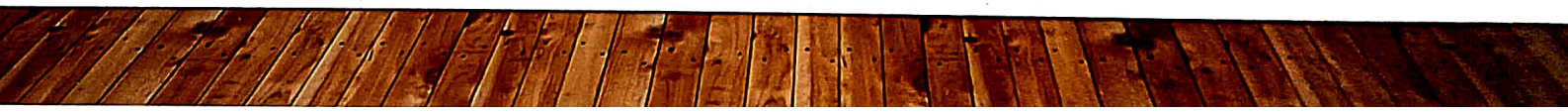
Ex:servlet-api.jar

4.logs:

It is used to store all the logs messages displayed on Server console. Since server console has limited memory.

5.webapps

It is used to deploy all the web -applications on to apache-tomcat server.



JEE Container

*Application Server is used to execute both web and enterprise applications(intern we call it as JEE Application).

*To run any JEE application on a server ,the server must contain a JEE container in it .

*JEE Container is an engine which is used to manage all the JEE components such as Servlet,JSP,EJB,etc.

*Server basically performs 2 important tasks namely

a>It manages all the resource.

b>Provides runtime environment

*Web Server is used to execute only web applications .

*There are 3 different types of logics associated with dynamic application which are as follows

1>Presentation Logic:

It is used to present the contents onto an application.

Technologies Involved:

Ex:HTML,CSS,JS,JSP,PHP,etc

2>Persistence Logic:

Persist means to store .Persistence Logic is used to persist data into the persistence systems(DataBase).

Technologies Involved:

Ex:JDBC,SQL,Hibernate,etc

3>Business Logic:

It performs the core functionality that is some set of calculation and validation operation on an application.

Technologies Involved:

Servlet,Spring,etc.

Steps to create a basic Web-Project(External Way of Execution)

*Click on Open perspective and select JEE as a Perspective.

*Open Navigator Mode.

*Right Click within the navigator and create a new Dynamic Web Project .

*Generate web.xml(Right click on Project and select Java EE Tools and click on Generate Deployment Descriptor Stub)-Inside WEB-INF folder.

There are 3 different criteria present for any resource in general .

*1>Create Resource:

Create a HTML Resource with any name which will be generated in WebContent Folder.

(Select Web Content Folder-→hit control+N-→type ht)

2>Configure the Resource:

All the resources must be configured in web.xml(Deployment Descriptor Stub)

3>Deploy the Resource:

All the resources must be deployed onto the webapps folder of Apache-Tomcat Server.

(Go To Apache-Tomcat Folder---→webapps-----create new Folder(web-application)---paste(WEB-INF(web.xml)+Home.html))

Note:

Goto Apache-Tomcat –start the server

Goto any browser –and type—

http://localhost:8080/basic----- (folder name which you have created inside webapps)

Steps to create a basic Web-Project(External Way of Execution)

*Click on Open perspective and select JEE as a Perspective.

*Open Navigator Mode.

*Right Click within the navigator and create a new Dynamic Web Project .

*Generate web.xml(Right click on Project and select Java EE Tools and click on Generate Deployment Descriptor Stub)-Inside WEB-INF folder.

There are 3 different criteria present for any resource in general .

*1>Create Resource:

Create a HTML Resource with any name which will be generated in WebContent Folder.

(Select Web Content Folder-→hit control+N-→type ht)

2>Configure the Resource:

All the resources must be configured in web.xml(Deployment Descriptor Stub)

3>Deploy the Resource:

All the resources must be deployed onto the webapps folder of Apache-Tomcat Server.

(Go To Apache-Tomcat Folder---→webapps-----create new Folder(web-application)---paste(WEB-INF(web.xml)+Home.html))

Note:

Goto Apache-Tomcat –start the server

Goto any browser –and type—

http://localhost:8080/basic----- (folder name which you have created inside webapps)

Configuration of Apache-Tomcat Server in Eclipse(For Internal way of execution)

*Before configuration of Apache-Tomcat ,Server must not be in use anywhere externally.

*Go to Window and Select Preferences.

*Open Server preference and click on Runtime Environments.

*Click on add and select the respective version of Apache-Tomcat Server and click on next.

*Browse for the appropriate version of Apache-Tomcat Server and open Apache-Tomcat Server folder and click on finish, Apply and close .

After this Just run the application Internally

A Servers folder will be generated within the navigator .

RightClick on Project ----→Run as--→ Run On Server

Welcome File or Landing Page

*A file or page which is automatically displayed whenever client uses an application is known as Welcome File or Landing Page.

*index is considered to the default welcome file or landing page which is automatically loaded by the JEE container whenever client uses an application .

*We can explicitly make any file as Welcome File or Landing Page by renaming it as index.

*If the configuration file is not available ,then the JEE container fails to load an application where it throws http 404 error(Client Error).

web.xml or Deployment Descriptor Stub

*It is a configuration file in xml format which is used to store the information with respect to the configurable resources of an application .

*Each and every application must mandatorily have only one web.xml present without which the JEE container fails to load an application where it throws http 404 error(Resources not available).

*The current version of xml used is 1.0.

*The root tag for a xml file is <web-app>

Servlet

Servlet is a server side java program which performs all the 3 different types of Logics such as Presentation Logic ,Persistence Logic and Business Logic along with which process Http Client request and get back some Http response.

Types of Servlet:

There are 2 different types of Servlet present namely

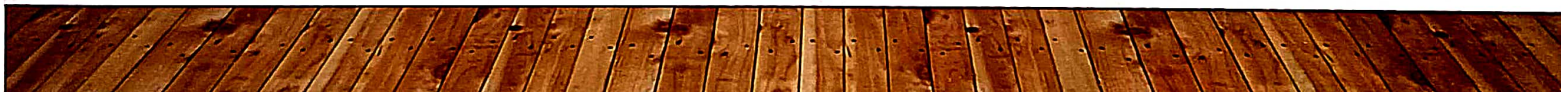
1.GenericServlet

2.HttpServlet.

GenericServlet

*Since it is not specific to any protocol or independent of protocol hence the name GenericServlet.

*GenericServlet Does not supports Session:



Session:Any activity which takes place between start time and stop time is known as Session.

*GenericServlet is an abstract class present in javax.servlet package.

*GenericServlet contains 3 methods in it out of which one is an abstract method and other 2 are concrete methods.

*The abstract method present in GenericServlet is named as service() which has to be overridden mandatorily for 2 important reasons namely.

1.since service() is an abstract method.

2.Since service() is the only method which takes ServletRequest and ServletResponse as a parameter which is responsible for processing the client request.

Method Signature:

+void service(ServletRequest req,ServletResponse resp)throws ServletException,IOExecption.

*The other two methods present in GenericServlet are named as init() and destroy() where overriding these 2 methods are optional since these 2 are concrete methods.

Writing a GenericServlet

```
public class OurServlet extends javax.servlet.GenericServlet
{
@override
public void service(ServletRequest req,ServletResponse resp)throws
ServletException,IOException
{
// Write Servlet Logic here//
}

}
}
```

HttpServlet

*Since it is specific to a particular type of protocol called Http protocol hence the name HttpServlet.

*HttpServlet Supports Session.

*HttpServlet is an abstract class present in javax.servlet.http package.

*HttpServlet contains only concrete methods without any abstract methods in it.

- In case of HttpSevlet we need to override a respective concrete methods called doXXX() method for a particular type of Http Request.

Method Signature:

protected void doXXX(HttpServletRequest req,HttpServletResponse resp)throws ServletException,IOException.

*There are 8 different types of Http Requests are present

1.POST

2.GET

3.PUT

4.TRACE

5.DELETE

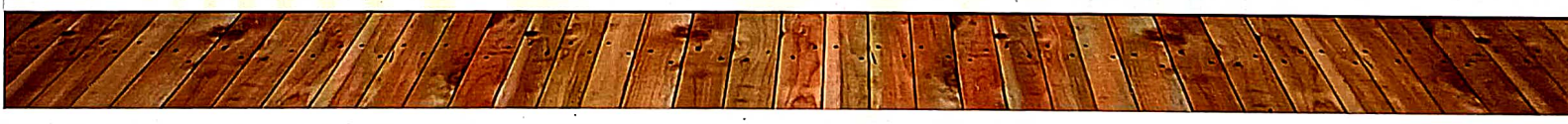
6.OPTION

7.HEAD

8.CONNECT

Writing a HttpServlet:

```
public class OurServlet extends javax.servlet.http.HttpServlet
{
@Override
protected void doXXX(HttpServletRequest req,HttpServletResponse resp) throws
ServletException,IOException
{
//Servlet Logic Here//
}
}
```



*Web Resources can be accessed based on unique url pattern.

*Since Servlet is also a web resource ,it has to be accessed based on Unique url pattern.

*Any resources can always be configured in two different ways namely

1.web.xml

2.Annotation

*To configure any resource using annotation ,it is mandatory to use JEE 3.x or above.(x=0,1,2,3….)

Configuration of Servlet in web.xml

There are 3 different properties to be configured for a Servlet in web.xml

1.Servlet-Name(Unique)

2.Url-Pattern(Unique)

3.Fully Qualified Class Name

UI/Form Data

Diagram

The data which is entered by the enduser(Client) on a form page and is submitted to the server in the form of key and value pair is known as UI/Form Data.

*All the keys associated with respect to form page are represented as Unique Identifier in the form of id/name.

*The UI/Form Data is always associated with a request and can be accessed only within the service() since service() is the only method which takes ServletRequest and ServletResponse as a parameter which is responsible for processing each and every client request.

*Once the request is made the data's are carried to the server as a part of request object in the form of key and value pair which is displayed in the url

*The UI/Form Data can be fetched by using getParameter().

Method Signature:

+ String getParameter(String key)

a)In this method the key is taken as an argument.

b)If the key is present then the method returns associated value.

c)If the key is not present then the method returns null but not any error or exception.

Steps to create a Dynamic Web-Application

1>Open Eclipse in JEE Perspective.

2>Open Navigator mode

3>Right click within the navigator and create a new dynamic web project and name it.

4>Generate web.xml(Deployment Descriptor Stub):(It will be generated inside WEB-INF folder)

5>Add servlet-api.jar file into the lib folder.

6.Create Resource:-→1>Form.html(Select webcontent folder---hit control n---type ht---and name the file).

2.FirstSevlet.java:→select the src folder→to create standard package structure(select src folder→hit control +n--→type pac(package)—select the application name and hit control+n to create a class.

7.Configure Resource:--→1.Using web.xml 2.Through annotation

8.Deploy Resource:-→webapps-→create a new folder→1.WEB-INF(lib folder+web.xml+classes folder).

2.Form.html

Code for UI/Form Data using web.xml

//Form.html

```
<!DOCTYPE html>
<html>
<meta charset="ISO-8859-1">
<body bgcolor="cyan">
<form action="btm">
Name:<input type="text" name="nm">
<br></br>
Place:<input type="text" name="pl">
<br></br>
<input type="submit" value="Request">
</form>
</body>
</html>
```

```
//FirstServlet
package org.jsp.dynamicApp;
import java.io.*;
import javax.servlet.*;
public class FirstServlet extends GenericServlet
{

    @Override
    public void service(ServletRequest req, ServletResponse resp)
throws ServletException, IOException
    {
        //Fetch Ui/Form Data
        String nm=req.getParameter("nm");
        String pl=req.getParameter("pl");
```

```
        //Display name and place as a response back to the client by
creating a new html page.
        PrintWriter out=resp.getWriter();
        out.println("<html><body bgcolor='yellow'>"
                + "<h1>My name is "+nm+" from "+pl+"</h1>"
                        + "</body></html>");
        out.close();

}

}
```

```
//Web.xml
<?xml version="1.0" encoding="UTF-8"?>
<display-name>DynamicProj</display-name>
  <welcome-file-list>
    <welcome-file>Form.html</welcome-file>
  </welcome-file-list>
  <servlet-mapping>
    <servlet-name>FirstServ</servlet-name>
    <url-pattern>/btm</url-pattern>
  </servlet-mapping>
  <servlet>
    <servlet-name>FirstServ</servlet-name>
    <servlet-class>org.jsp.dynamicApp.FirstServlet</servlet-class>
  </servlet>
</web-app>
```

Code for UI/Form Data using Annotation

*Delete web.xml/Don't generate web.xml

*Rename Form.html---→index.html

*Add annotation in FirstServlet.java

Http Post

*Post Request is used to post some contents or data from client to the server.

Ex:Posting Resume,Posting Profile in matrimonial site

*Post Request deals with unlimited data.

*Post Request is non-idempotent(Changes happens at the Server side).

*Post Request can not be bookmarked(Can not be saved).

In case of Post Request the data are carried to the server as a part of Http Request Body which is not displayed to the endUcer(Client).Hence the data are secured.

*Whenever we deal with n-number of data then the type of request is Post Request.

Http Get

*Get Request is used to get some contents or resources from the Server.

*Get Request deals with limited data. That is 1024 Characters.

*Get Request is Idempotent.(No Changes at the Server side).

*Get request is safe.

*Get Request can be bookmarked(Can be saved).

*Incase of get request the datas will be carried to the server in the form of key and value pair which is displayed even to the endUser .Hence the data are not secured.

*Whenever the type of request is not mentioned or not configured then by default the type of request is Get Request.

*Whenever we deal with a link then the type of request is get request.

Code to dynamically save the data into the database server in a secured manner using POST Request.

Or

Integration of JDBC and Servlet

Code to fetch the data from the DataBase Server using Get Request