OBJECT™
TECHNOLOGIES

Servlet I

What will be covered

- What is servlet
- Need of servlets
- Dynamic web applications
- Web servers
- First simple servlet
- Web application structure
- Deployment of web application
- What happens after hitting URL in browser
- Servlet life cycle
- Servlet API

What is servlet

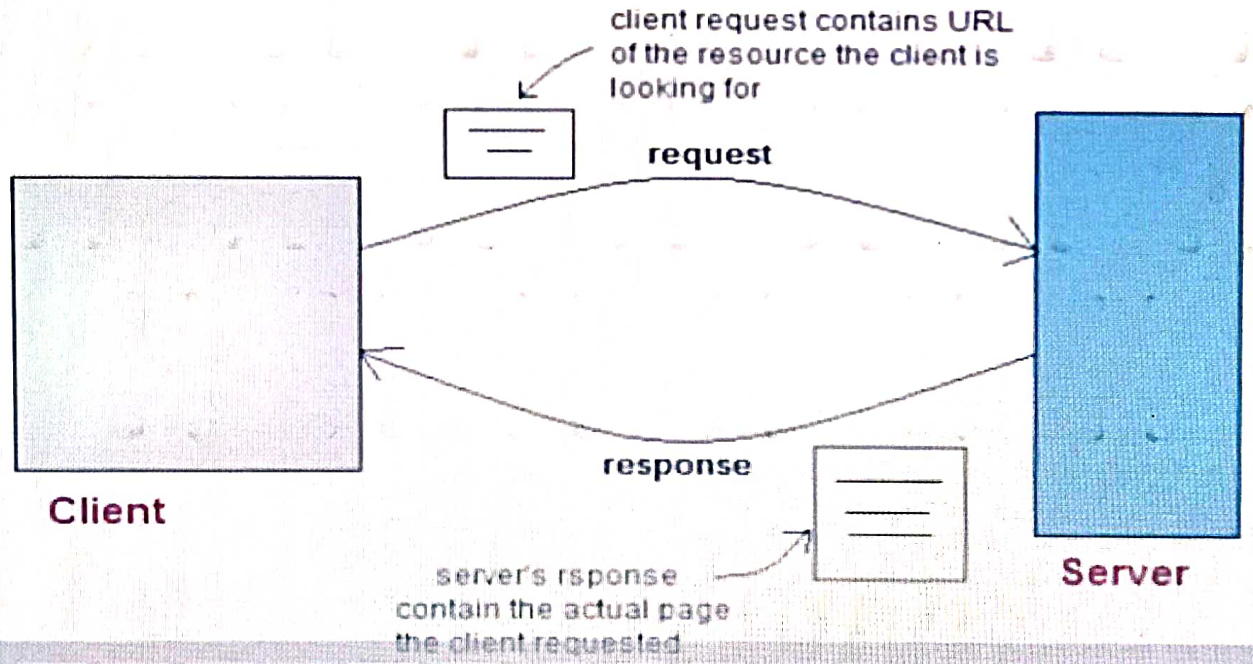
- Servlet is typically a java class
- The main functionality of servlet is to generate dynamic content as a response to client's requests. This content is mainly in the form of HTML.
- Servlet runs in hosting environment given by web server becuase servlets are deployed in web server
- Life cycle of servlets is controlled by web server.

Need of servlets

- Shortly after the Web began to be used for delivering services, service providers recognized the need for dynamic content.
- Applets, one of the earliest attempts toward this goal of delivering dynamic content, focused on using the client platform to deliver dynamic user experiences.
- Before Servlet, CGI (Common Gateway Interface) scripting language was common as a server-side programming language which was used to generate dynamic content on the server side. However, there were many disadvantages to this technology including platform dependence.
- To address these limitations, Java Servlet technology was created as a portable way to provide dynamic, user-oriented content.

Dynamic web applications

- A web application can be described as collection of web pages (e.g. a website) and when we call it dynamic, it simply means that the web pages are not same for all the users, web pages would be generated on server side based on the request made by client(user's browser).
- Dynamic web pages are generated on the server side based on the user's input and then sent back from server to client as in the form of response.



Web servers

- Web pages are a collection of data, including images, text files, hyperlinks, database files etc., all located on some computer which is called web server.
- The primary objective of any web server is to collect, process and provide web pages to the users as per requests.
- This intercommunication of a web server with a web browser is done with the help of a protocol named HTTP (Hypertext Transfer Protocol).
- Web servers can serve static as well as dynamic contents. Web Servers also assists in emailing services and storing files. Therefore it also uses SMTP (Simple Mail Transfer Protocol) and FTP (File Transfer Protocol) protocols to support the respective services.
- The hardware of the web servers are connected to the Internet that manages the data exchange facility within different connected devices. In contrast, the software of web server software is responsible for controlling how a user accesses delivered files.

Web servers

OBJECT TECHNOLOGIES

- Examples of web servers

Web server	Description
Apache Tomcat	It is an open source software and can be installed on almost all operating systems
Jetty	The Jetty web server is developed under the Eclipse Foundation.
WebLogic	This is primary application server offering from Oracle
WebSphere	IBM has also developed its own application server, called WebSphere
WildFly	Wildfly is an open-source Java application server, developed by Red Hat.
Glassfish	Glassfish is an open-source application server, also sponsored by Oracle.

First simple servlet

OBJECT TECHNOLOGIES

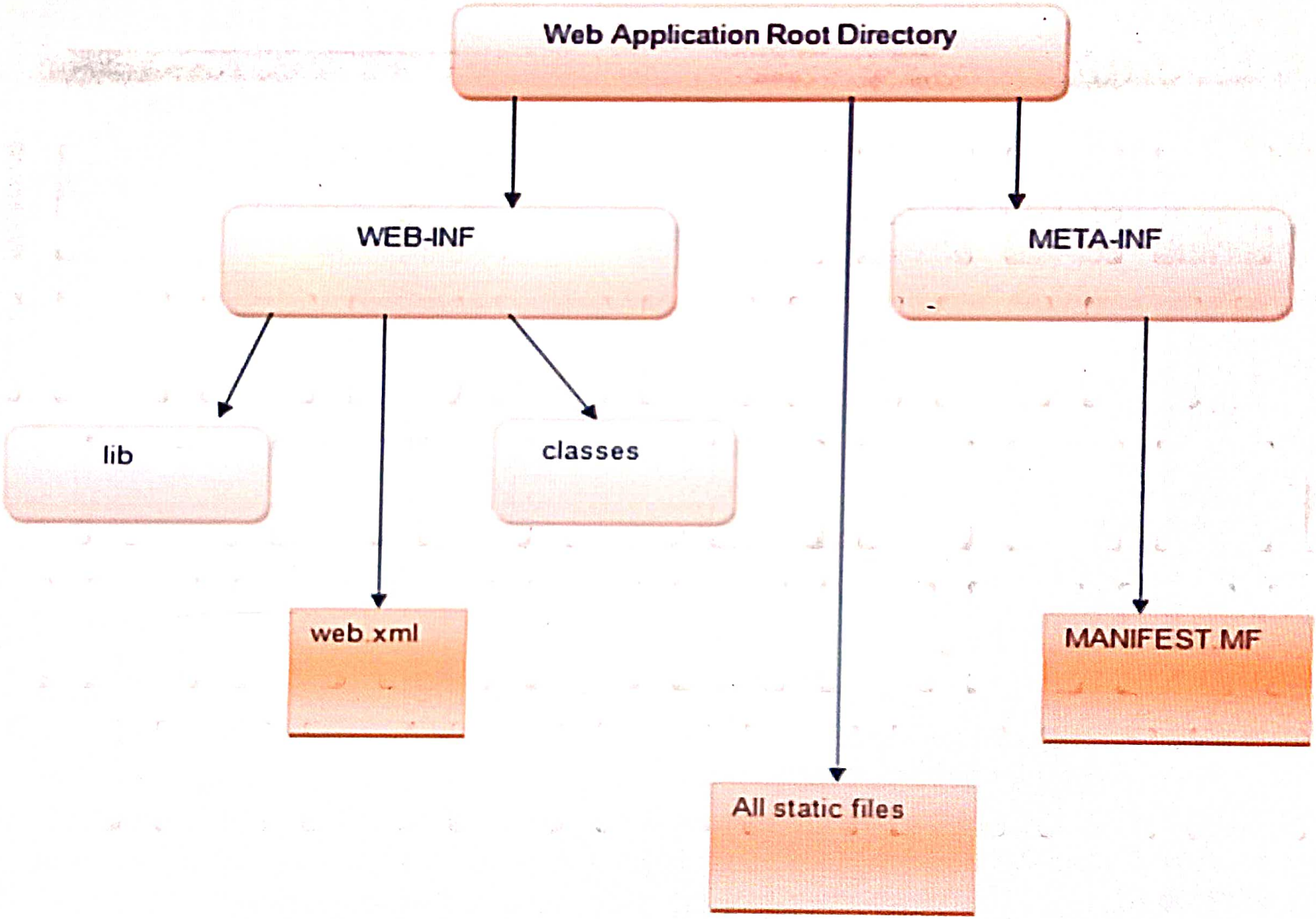
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.*;

@WebServlet("/hello")
public class HelloServlet extends HttpServlet {
    protected void doGet(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        PrintWriter out = response.getWriter();
        out.print("<h1> Welcome to servlets </h1>");
    }

    protected void doPost(HttpServletRequest request,
            HttpServletResponse response)
            throws ServletException, IOException {
        // TODO Auto-generated method stub
        doGet(request, response);
    }
}
```

First simple servlet

OBJECT TECHNOLOGIES

- Note :
- Each servlet class should directly or indirectly implement Servlet interface. Here servlet class is extending from HttpServlet which implments Servlet interface.
- Servlet related classes and interface imported from package javax.servlet and javax.servlet.http
- Servlet class provides service methods in the form of doGet() and doPost() which are overridden to generate dynamic HTML.
- These service methods provide request and response objects.
- Request object can be used for getting more information about request like request method, request paramters etc.
- Response object is used for opening the writing stream for stuffing HTML which finally gets displayed on the client's browser.

Web application structure



Web application structure

OBJECT

- **Web Application Root Directory** – This is the main or Root folder of web application. Usually name of this folder becomes your web application context. For example ,if our web application name is FirstWebApplication , then folder name will be FirstWebApplication and web application will be accessible via http://localhost:8080/FirstWebApplication
- **Static Files** – All static files like HTML, css ,javascript will be placed directly under web application root directory. These file can be further divided into subfolders as per need. If we want to make these files secure , we need to place these files under WEB-INF directory.
- **META-INF/MAINFEST.MF**- This is the manifest file. This is required when the application is deployed in the form of archieve(like war)

Web application structure

OBJECT

- **WEB-INF**- This is the special directory under web application root directory. This is special because this is secured folder and files available within this folder will not be accessible to client directly. Which means say if this directory has one file "index.html" then this file cannot be accessed directly via http://localhost:8080/FirstWebApplication/index.html
- **WEB-INF/lib**- All the required jar files or third party jar files will be placed inside this directory.
- **WEB-INF/classes**- All the java code including servlets for the web application will go inside classes folder. We can also put our own classes into a JAR file, and place it here, rather than putting those classes in the classes directory.
- **web.xml** – web.xml resides under WEB-INF folder and it is also known as deployment descriptor. All the configuration of web application like servlets configuration , filters configuration , welcome file list etc are configured in web.xml. We will discuss web.xml in detail

Deployment of web application

OBJECT

- Deploy a web application means to make it ready to be used by its clients
- This is achieved by structuring the files that constitute the web application in a certain standard way and by installing (usually, just moving) it in a certain location of the server.
- This location of the server is called deployment directory. This deployment directory will be differently named in different web servers.
- In apache tomcat this deplyment directory is called web-apps which hosts all the web application folders.
- Web application can even be hosted as in the form of web archieve(war) file.

Deployment of web application

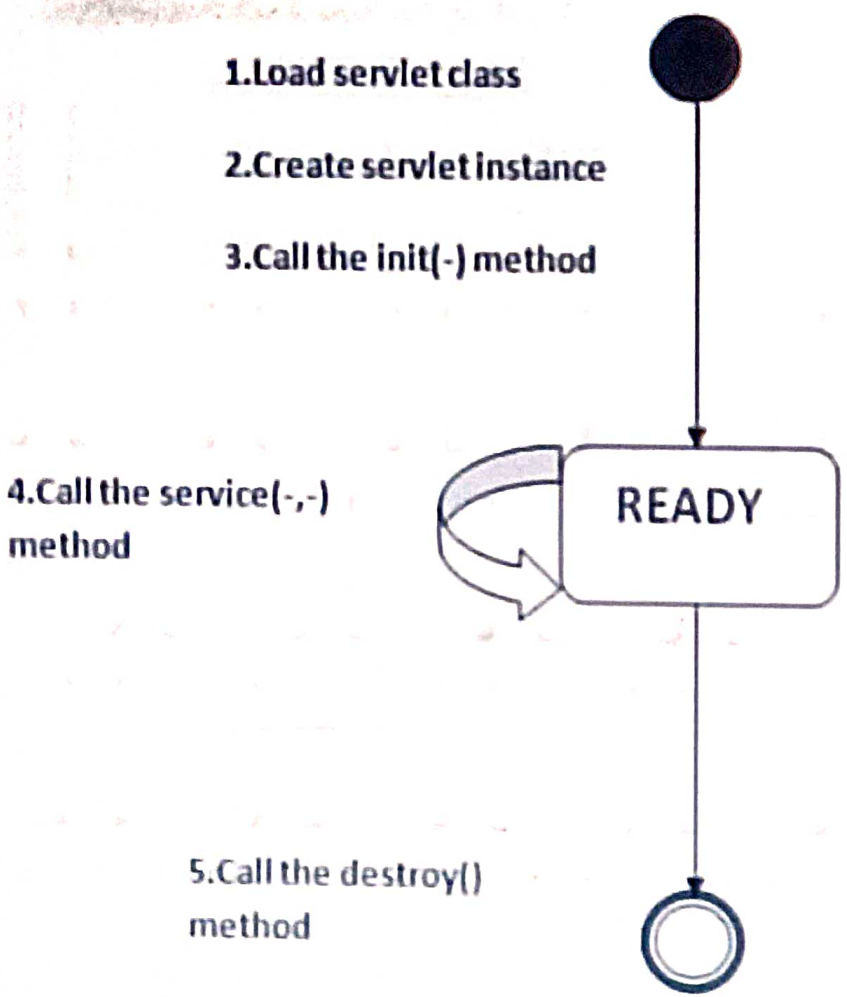
OBJECT TECHNOLOGIES

- Eclipse environment creates the required directory structure as needed by web application when the type of the project is selected as dynamic web project.
- For this project, runtime environment has to be specified for the web server under which the web application is getting deployed so that it makes use of web libraries.
- For deployment and undeployment of web application, menu is provided in the server's view.
- Add and remove option from the server allows to select the web application to be deployed or undeployed.

What happens after hitting URL in browser

- Client hits the URL in the browser
- Request is accepted by corrosponding web server.
- Web server checks the request URL and finds the appropriate resource for serving the response.
- In case of servlets, request url will be compared with the value in @WebServlet annotation written above the servlet class. This way web server understands which servlet should be used to generate response.
- If the servlet class is not loaded in the memory, it gets loaded.
- Servlet gets instantiated(object creation).
- Init method of servlet gets called for initialization of resources needed by the servlet.
- Appropriate service method gets called based on the request method (doGet() or doPost() etc) by sending new request and response objects for every request.
- For subsequent requests by client, only correct service method gets called with new request and response objects.

Servlet life cycle



- There are three states of a servlet: new, ready and end.
- The servlet is in new state if servlet instance is created.
- After invoking the init() method, Servlet comes in the ready state.
- In the ready state, servlet performs all the tasks.
- When the web container invokes the destroy() method, it shifts to the end state.

Servlet life cycle

OBJECT

- **Servlet class is loaded**
- The classloader is responsible to load the servlet class. The servlet class is loaded when the first request for the servlet is received by the web container.
- **Servlet instance is created**
- The web container creates the instance of a servlet after loading the servlet class. The servlet instance is created only once in the servlet life cycle.
- **init method is invoked**
- The web container calls the init method only once after creating the servlet instance. The init method is used to initialize the servlet. It is the life cycle method of the javax.servlet.Servlet interface. Syntax of the init method is given below:

```
public void init(ServletConfig config) throws ServletException
```

Servlet life cycle

OBJECT

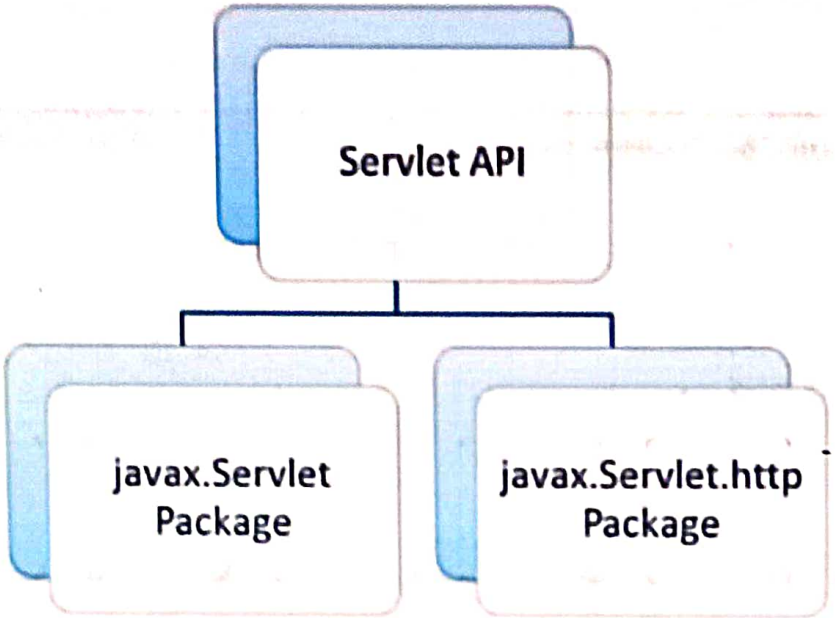
- **service method is invoked**
- The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice that servlet is initialized only once. The syntax of the service method of the Servlet interface is given below:

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

- **destroy method is invoked**
- The web container calls the destroy method before removing the servlet instance from the service. It gives the servlet an opportunity to clean up any resource for example memory, thread etc. The syntax of the destroy method of the Servlet interface is given below:

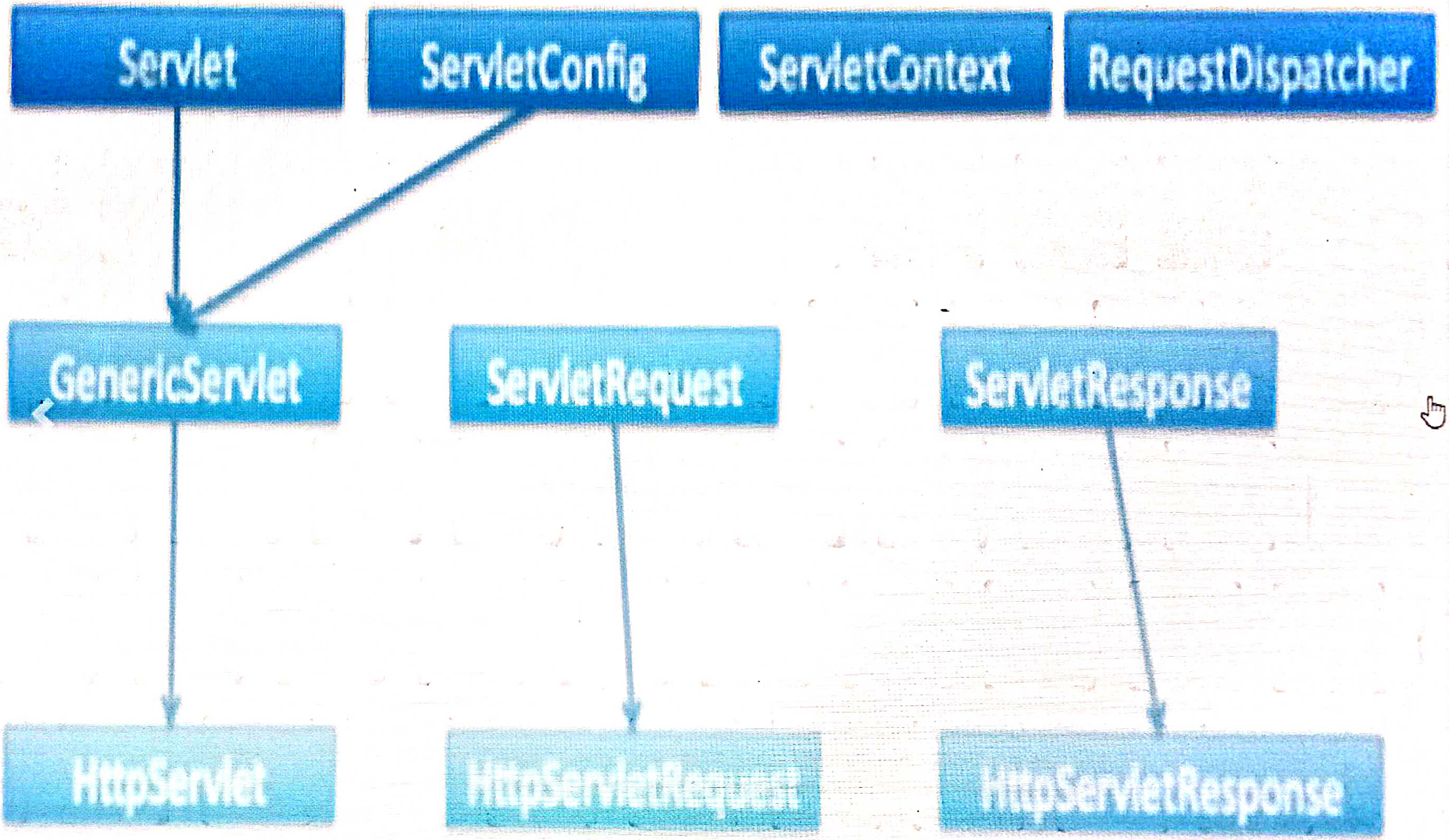
```
public void destroy()
```

Servlet API



- Servlet API is divided into 2 major packages javax.servlet and javax.servlet.http.
- javax.servlet package contains a number of classes and interfaces that define the agreement between the servlet class and the run-time environment provided an instance of the servlet class by a conforming servlet container.
- javax.servlet.http.* package contains a number of classes and interfaces that define the agreement between the servlet class under HTTP protocol

Servlet API


Servlet
ServletConfig
ServletContext
RequestDispatcher
GenericServlet
ServletRequest
ServletResponse
HttpServlet
HttpServletRequest
HttpServletResponse

Servlet API

OBJECT

- HttpServlet class is used as super class for creating our own servlets.
- This class provides service methods for each request method like doGet, doPost etc.
- These methods accept 2 arguments of types HttpServletRequest and HttpServletResponse. Creating these objects and passing it to call these methods is the responsibility of web server.

Servlet I

In this chapter we will be learning about basics of servlet and writing simple servlet and understanding it's deployment in the web application. We will be learning even about how to test the servlet by making request from client side.

What Is Servlet

A Servlet is a Java class that runs on a web server and processes HTTP requests from web clients, such as web browsers. Servlet needs to be hosted in the web application in a perticular directory structure and it's life cycle is under the control of web server.

Once the servlet receives the request, it can perform a variety of actions, such as accessing a database, processing form data, or generating dynamic web content. Once the servlet has completed its processing, it returns a response to the client in the form of an HTTP response.

They can be configured to handle different types of HTTP requests, such as GET, POST, and PUT requests. They can also be configured to handle different types of data, such as HTML, XML, and JSON.

Servlet Basics

What is servlet?

A Servlet is a Java class that runs on a web server and processes HTTP requests from web clients, such as web browsers. Servlet needs to be hosted in the web application in a perticular directory structure and it's life cycle is under the control of web server.

Once the servlet receives the request, it can perform a variety of actions, such as accessing a database, processing form data, or generating dynamic web content. Once the servlet has completed its processing, it returns a response to the client in the form of an HTTP response.

They can be configured to handle different types of HTTP requests, such as GET, POST, and PUT requests. They can also be configured to handle different types of data, such as HTML, XML, and JSON.

What is web server?

A web server is software and hardware that uses HTTP (Hypertext Transfer Protocol) and other protocols to respond to client requests made over the World Wide Web. The main job of a web server is to display website content through storing, processing and delivering webpages to users.

Web servers are used in web hosting, or the hosting of data for websites and web-based applications -- or web applications.

What is a Static Web Page?

What is a Static Web Page?

A static website contains simple HTML pages and supporting files (e.g., Cascading Style Sheets (CSS), JavaScript (JS)) hosted on a web server. When a site visitor requests a static page, say, by clicking a link, selecting a browser bookmark, or entering a URL; the web server sends the page directly to the web browser without modifying the final content of the page.

What is a Dynamic Web Page?

A dynamic page displays different content for different users while retaining the same layout and design. Such pages, usually written in CGI, AJAX, ASP or ASP.NET, servlet take more time to load than simple static pages. They're frequently implemented to show information that changes frequently, e.g., weather updates or stock prices. Dynamic pages usually contain application programs for different services and require server-side resources like databases. A database allows the page creator to separate the website's design from the content to be displayed to users. Once they upload content into the database, it is retrieved by the website in response to a user request.

Web Container

Web Container or Servlet Container is a part of Web Server that interacts with servlets. When a request comes in, it first goes to Container, it analyze the request and maps it to correct servlets.

The main responsibilities of Web Container are as follows -

- Servlets do not have main method, they are instantiated by Container.
- Creates threads to handle the request.
- Call doGet() or doPost() method based on HTTP method used
- Create HTTP Request and Response objects.
- Load and unload servlet.

Examples of open source web containers

1. Apache Tomcat from Apache Software Foundation.
2. GlassFish from Oracle.

   and many more ...

Creating Simple Servlet

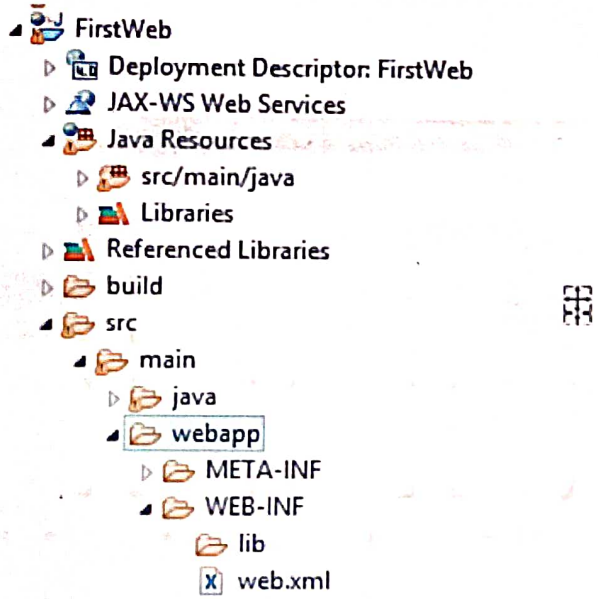
Step 1: Create a Project:

1. Lets create a Servlet application in Eclipse. Open Eclipse and then click File ? New ? Click Dynamic Web Project.
2. Give Project name and click Next.

Creating Simple Servlet

Step 1: Create a Project:

1. Lets create a Servlet application in Eclipse. Open Eclipse and then click File ? New ? Click Dynamic Web Project.
2. Give Project name and click Next.
3. Tick the checkbox that says Generate web.xml deployment descriptor
4. Initial Project structure:



Step 2: Create a Servlet class:

Use src folder. Use the option file->new->servlet. Specify the name of the servlet e.g. HelloServlet. It should extend from HttpServlet. Specify the url pattern /hello. Select the methods to override i.e. doGet() method.

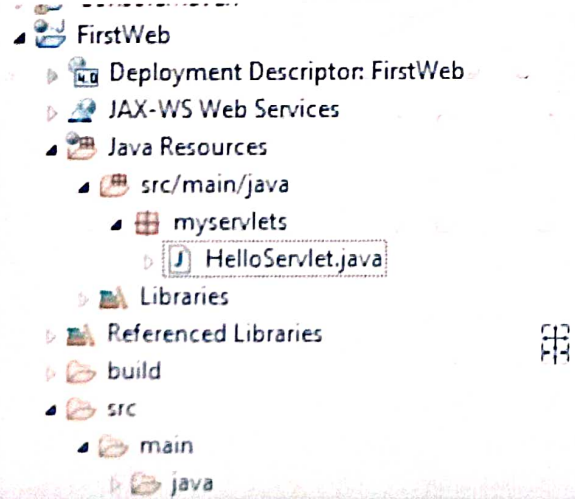
```
package myservlets;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/hello")
```

Step 2: Create a Servlet class:

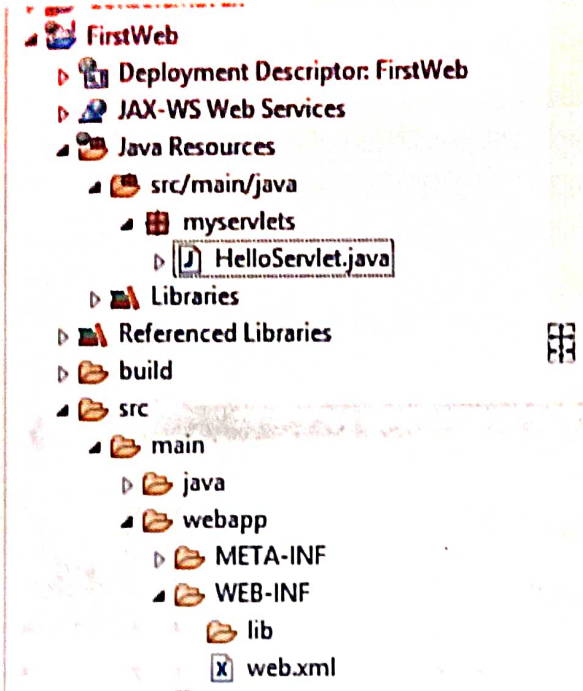
Use src folder. Use the option file->new->servlet. Specify the name of the servlet e.g. HelloServlet. It should extend from HttpServlet. Specify the url pattern /hello. Select the methods to override i.e. doGet() method.

```
package myservlets;
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/hello")
public class HelloServlet extends HttpServlet {
   private static final long serialVersionUID = 1L;
   protected void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException
{
      PrintWriter out = response.getWriter();
      out.print("<h1> Welcome to servlets </h1>");
   }
}
```

Final Project Structure :



Final Project Structure :



Note : The Servlet API 3.0 introduces a new package called javax.servlet.annotation which provides annotation types which can be used for annotating a servlet class. The annotations can replace equivalent XML configuration in the web deployment descriptor file (web.xml) such as servlet declaration and servlet mapping. Servlet containers will process the annotated classes at deployment time.

The annotation @WebServlet is used for declaring a servlet class (the class still must extend from the HttpServlet class) and configuring mapping for it.

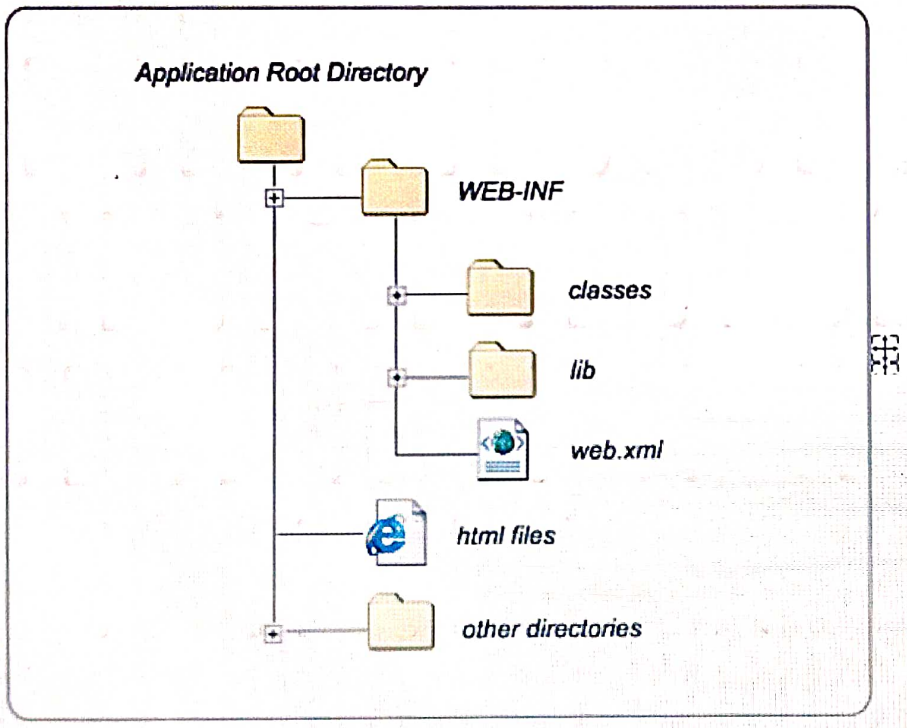
Equivalent configuration needed in web.xml if @WebServlet annotation is not written :

```
<servlet>
   <servlet-name>HelloServlet</servlet-name>
  <servlet-class>myservlets.HelloServlet</servlet-class>
</servlet>

<servlet-mapping>
   <servlet-name>HelloServlet</servlet-name>
   <url-pattern>/hello</url-pattern>
</servlet-mapping>
```

Equivalent configuration needed in web.xml if @WebServlet annotation is not written :

```
<servlet>
  <servlet-name>HelloServlet</servlet-name>
 <servlet-class>myservlets.HelloServlet</servlet-class>
</servlet>

<servlet-mapping>
  <servlet-name>HelloServlet</servlet-name>
  <url-pattern>/hello</url-pattern>
</servlet-mapping>
```

Web Application Strcuture

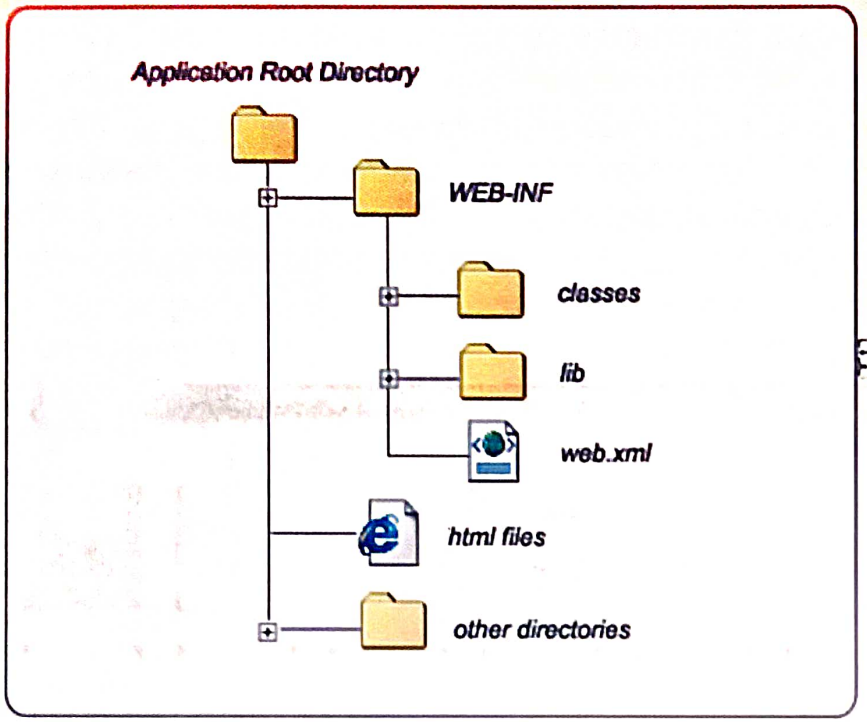


The Root Directory

The root directory of you web application can have any name. In the above example the root directory name is fwebapp .Under the root directory, all files can be put that should be accessible in your web application.

If your web application is mapped to the URL *http://localhost:8080/webapp/* then the index.html page will be accessible by

Web Application Strcuture



The Root Directory

The root directory of you web application can have any name. In the above example the root directory name is fwebapp .Under the root directory, all files can be put that should be accessible in your web application.

If your web application is mapped to the URL *http://localhost:8080/webapp/* then the index.html page will be accessible by the URL *http://localhost:8080/webapp/index.html*

If you create any subdirectories under the root directory, and place files in these subdirectories, these will be available by the subdirectory/file path, in the web application. For instance, if you create a subdirectory called pages, and put a file index.jsp inside it, then you could access that file from the outside, via this URL: *http://localhost:8080/webapp/pages/index.jsp*

The WEB-INF Directory :

The WEB-INF directory is located just below the web app root directory. This directory is a meta information directory. Files stored here are not supposed to be accessible from a browser (although your web app can access them internally, in your code). Inside the WEB-INF directory there are two important directories (classes and lib, and one important file (web.xml). These are described below.

web.xml :

The WEB-INF Directory :

The WEB-INF directory is located just below the web app root directory. This directory is a meta information directory. Files stored here are not supposed to be accessible from a browser (although your web app can access them internally, in your code). Inside the WEB-INF directory there are two important directories (classes and lib, and one important file (web.xml). These are described below.

web.xml :

The web.xml file contains information about the web application, which is used by the Java web server / servlet container in order to properly deploy and execute the web application. For instance, the web.xml contains information about which servlets a web application should deploy, and what URL's they should be mapped to.

classes directory:

The classes directory contains all compiled Java classes that are part of your web application. The classes should be located in a directory structure matching their package structure.

lib directory:

The lib directory contains all JAR files used by your web application. This directory most often contains any third party libraries that your application is using. You could, however, also put your own classes into a JAR file, and locate it here, rather than putting those classes in the classes directory.

Note : Instead of copying entire directory into server we can package them into war file and then do the deployment. A WAR (or "web archive") file is simply a packaged webapp directory or archive file (like zip,rar etc) .It will store jsp, servlets, classes, meta data information,images, and tag libararies etc. Its standard file extension is .war. WAR files are used to package Web modules. A WAR file is for a Web application deployed to a server.

Deployment and Execution

Step 1 : Download and unzip tomcat server

1. observe lib, bin and webapps folder in tomcat root directory

Step 2 : Linking apache tomcat to eclipse environment

- This setting is workspace specific
- It is applicable for all the projects in that workspace

1. Window -> preferences
2. server -> runtime environment

Deployment and Execution

Step 1 : Download and unzip tomcat server

1. observe lib, bin and webapps folder in tomcat root directory

Step 2 : Linking apache tomcat to eclipse environment

- This setting is workspace specific
- It is applicable for all the projects in that workspace

1. Window -> preferences
2. server -> runtime environment
3. Add - select apache tomcat v9/v10
4. Select the installation directory which cosists of lib, bin and webapps

Step 3 : Creating local instance of tomcat server in eclipse

- Open Java EE perspective
- Go to servers view

1. Create new server
2. Select runtime environment(auto popped up)
3. Double click on server to open it's configuration window
4. Select deploy path as webapps directory
5. Save the configuration
6. Right cleck on server and start the server
7. Check the message on the console about startup
8. Open the browser and type the url http://localhost:8080 to view the tomcat home page

Step 4: Testing the servlet

1. Deploy the web app using add and remove option of server
2. Start the server
3. Open the browser
4. Type the proper URL - http://localhost:8080/FirstWeb/hello

All above steps can be automated by using the option for servlet as run on server.

Step 5: Verifying deployed application in tomcat deployment directory

Step 4: Testing the servlet

1. Deploy the web app using add and remove option of server
2. Start the server
3. Open the browser
4. Type the proper URL - http://localhost:8080/FirstWeb/hello

All above steps can be automated by using the option for servlet as run on server.

Step 5: Verifying deployed application in tomcat deployment directory

1. Go to tomcat installation directory
2. Open webapps(deplyment directory)
3. Find the application folder
4. Verfiy the structure of the web application deplyed

Assignments

1. Write a servlet which displays 'Welcome to Java Programming' on the web page