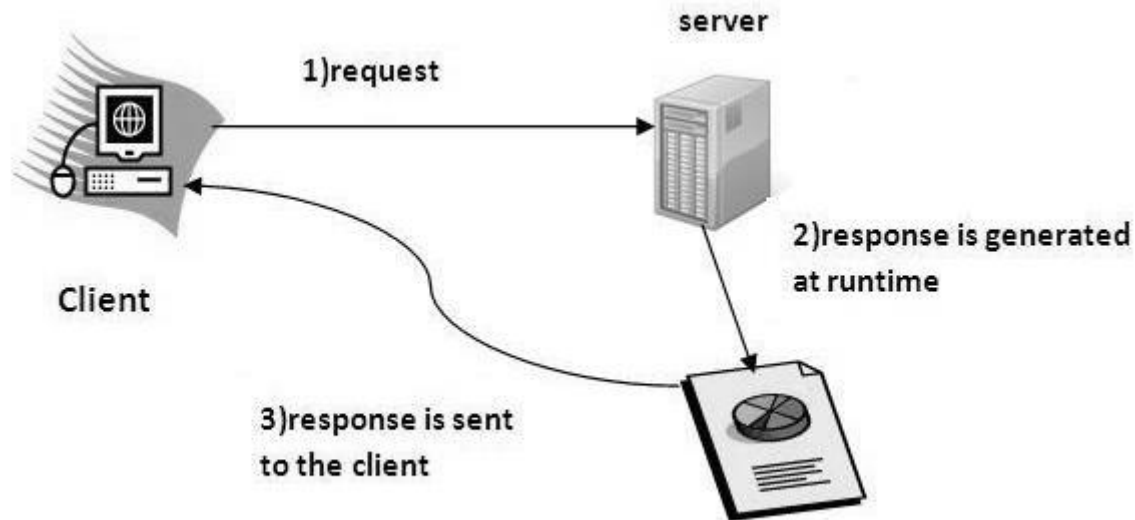


Unit -4 Servlets**4.1** The Life Cycle of a Servlet**4.2** The Java Servlet Development Kit**4.3** The Simple Servlet: Creating and compile servlet source code, start a web browser and request the servlet, example of echo servlet and deployment in tomcat server**4.4** The Servlet API, XML configuration in Tomcat**4.5** The javax.servlet Package: Reading database/table records and displaying using servlet

What is a Servlet?

Servlet can be described in many ways, depending on the context.

- Servlet is a technology which is used to create web application.
- Servlet is a web component that is deployed on the server to create dynamic web page.
- **Servlet** technology is used to create web application (resides at server side and generates dynamic web page).
- Servlet is an API that provides many interfaces and classes including documentations.
- Servlet is an interface that must be implemented for creating any servlet.
- Servlet is a class that extends the capabilities of the servers and responds to the incoming request. It can respond to any type of requests.



Before Servlet, CGI (Common Gateway Interface) scripting language was popular as a server-side programming language.

There are many interfaces and classes in the servlet API such as Servlet, GenericServlet, HttpServlet, ServletRequest, ServletResponse etc.

Web:

- Web consists of many client and server connected together using networks.
- Client makes request to web server. The web server receives the request, finds the resources and return response to the client. Server responds with some type of content to client.
- The client uses web browser to send request to server. The server responds to the browser with a set of instructions written in HTML.

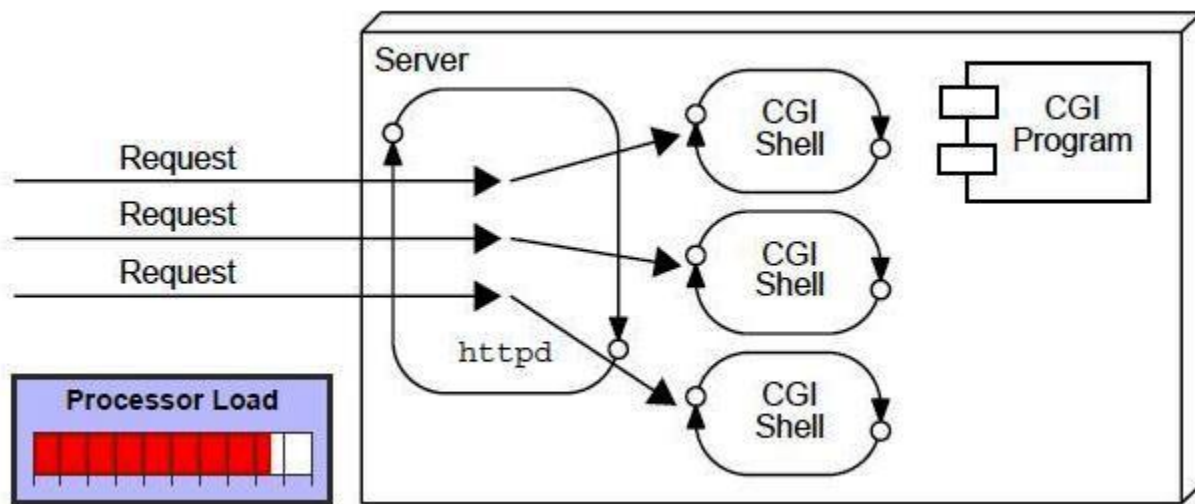
Web application

- A web application is an application accessible from the web.
- It is website with dynamic functionality on the server.

- A web application is composed of web components like Servlet, JSP, Filter etc. and other components such as HTML.
- The web components typically execute in Web Server and respond to HTTP request.
- Ex: google,yahoo

CGI (Common Gateway Interface)

- CGI technology enables the web server to call an external program and pass HTTP request information to the external program to process the request.
- In early days, Web server could dynamically construct a page by creating a separate process to handle each client request. The process would open connections to one or more databases in order to obtain required information. It communicated with the web server via interface called CGI. CGI allowed the separate process to read data from HTTP request and write data to the HTTP response.
- For each request, CGI starts a new process.

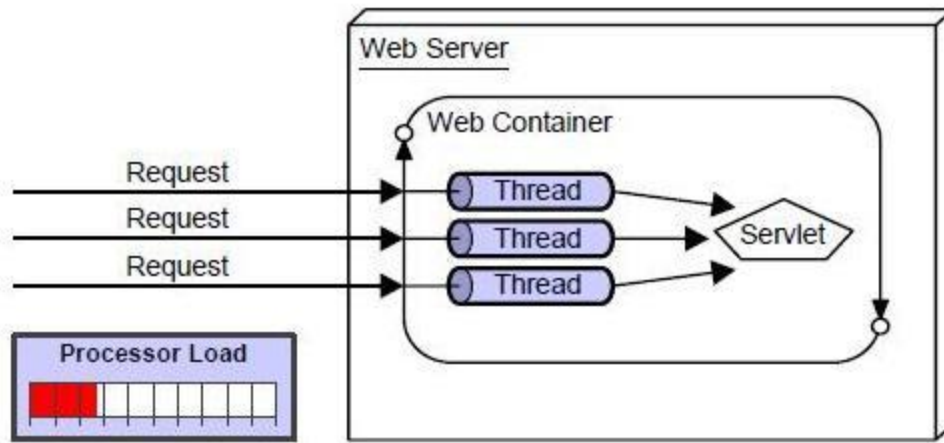


Disadvantages of CGI

There are many problems in CGI technology:

1. It was expensive in terms of processor and memory resources to create separate process for each client.
2. If number of client's increases, it takes more time for sending response.
3. For each request, it starts a process and Web server is limited to start processes.
4. CGI were platform dependent. Because it uses platform dependent language e.g. C, C++, perl.

Advantage of Servlet over CGI



There are many advantages of servlet over CGI.

1. The web container creates threads for handling the multiple requests to the servlet. Threads have a lot of benefits over the Processes such as they share a common memory area, lightweight, cost of communication between the threads are low. The basic benefits of servlet are as follows:
 1. **Better performance:** Servlets execute within address space of web server. It creates a thread for each request not process.
 2. **Portability:** Servlets are platform independent because they are written in java.
 3. **Robust:** Servlets are managed by JVM so no need to worry about memory leak, garbage collection etc.
 4. **Secure:** Java security manager on the server enforces a set of restrictions to protect the resource on a server machine.

Servlet Terminology

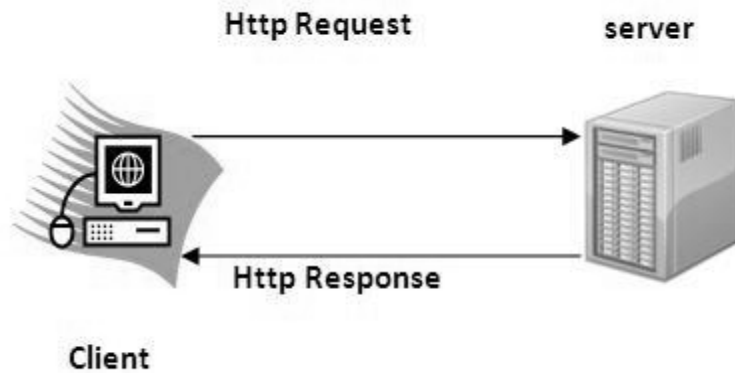
There are some key points that must be known by the servlet programmer like server, container, get request, post request etc. Let's first discuss these points before starting the servlet technology.

The basic **terminologies used in servlet** are given below:

1. HTTP
2. HTTP Request Types
3. Difference between Get and Post method
4. Container
5. Server and Difference between web server and application server
6. Content Type
7. Introduction of XML
8. Deployment

HTTP (Hyper Text Transfer Protocol)

- Http is the protocol that allows web servers and browsers to exchange data over the web.
- It is a request response protocol.
- Http uses reliable TCP connections by default on **TCP port 80**.
- It is stateless means each request is considered as the new request. In other words, server doesn't recognize the user by default.



Http Request Methods

- Every request has a header that tells the status of the client.
- There are many request methods.
- **Get** and **Post** requests are mostly used.

The http request methods are:

- GET
- POST
- HEAD
- PUT
- DELETE
- OPTIONS
- TRACE

No.	HTTP Request	Description
1	GET	Asks to get the resource at the requested URL.
2	POST	Asks the server to accept the body info attached. It is like GET request with extra info sent with the request.
3	HEAD	Asks for only the header part of whatever a GET would return. Just like GET but with no body.
4	TRACE	Asks for the loopback of the request message, for testing or troubleshooting.
5	PUT	Says to put the enclosed info (the body) at the requested URL.
6	DELETE	Says to delete the resource at the requested URL.
7	OPTIONS	Asks for a list of the HTTP methods to which the thing at the request URL can respond

What is the difference between Get and Post?

There are many differences between the Get and Post request. Let's see these differences:

No.	GET	POST
1	In case of Get request, only limited amount of data can be sent because data is sent in header.	In case of post request, large amount of data can be sent because data is sent in body.
2	Get request is not secured because data is exposed in URL bar.	Post request is secured because data is not exposed in URL bar.
3	Get request can be bookmarked	Post request cannot be bookmarked
4	Get request is idempotent . It means second request will be ignored until response of first request is delivered.	Post request is non-idempotent
5	Get request is more efficient and used more than Post	Post request is less efficient and used less than get.

Structure of Get Request

Data is sent in request header in case of get request.

It is the default request type.

Following informations are sent to the server.



Structure of Post Request

In post request original data is sent in message body.

Following information are passed to the server in case of post request.



Container

It provides runtime environment for JavaEE (j2ee) applications.

It performs many operations that are given below:

1. Life Cycle Management
2. Multithreaded support
3. Object Pooling
4. Security etc.

Server

It is a running program or software that provides services.

There are two types of servers:

1. Web Server
2. Application Server

Web Server

Web server contains only web or servlet container.

It can be used for servlet, jsp, struts, jsf etc. It can't be used for EJB.

Examples of Web Servers are: **Apache Tomcat** and **Resin**.

Application Server

Application server contains Web and EJB containers.

It can be used for servlet, jsp, struts, jsf, ejb etc.

Example of Application Servers are:

1. **JBoss** Open-source server from JBoss community.
2. **Glassfish** provided by Sun Microsystem. Now acquired by Oracle.
3. **Weblogic** provided by Oracle. It more secured.
4. **Websphere** provided by IBM.

Content Type

Content Type is also known as MIME (Multipurpose internet Mail Extension) Type. It is a **HTTP header** that provides the description about what are you sending to the browser.

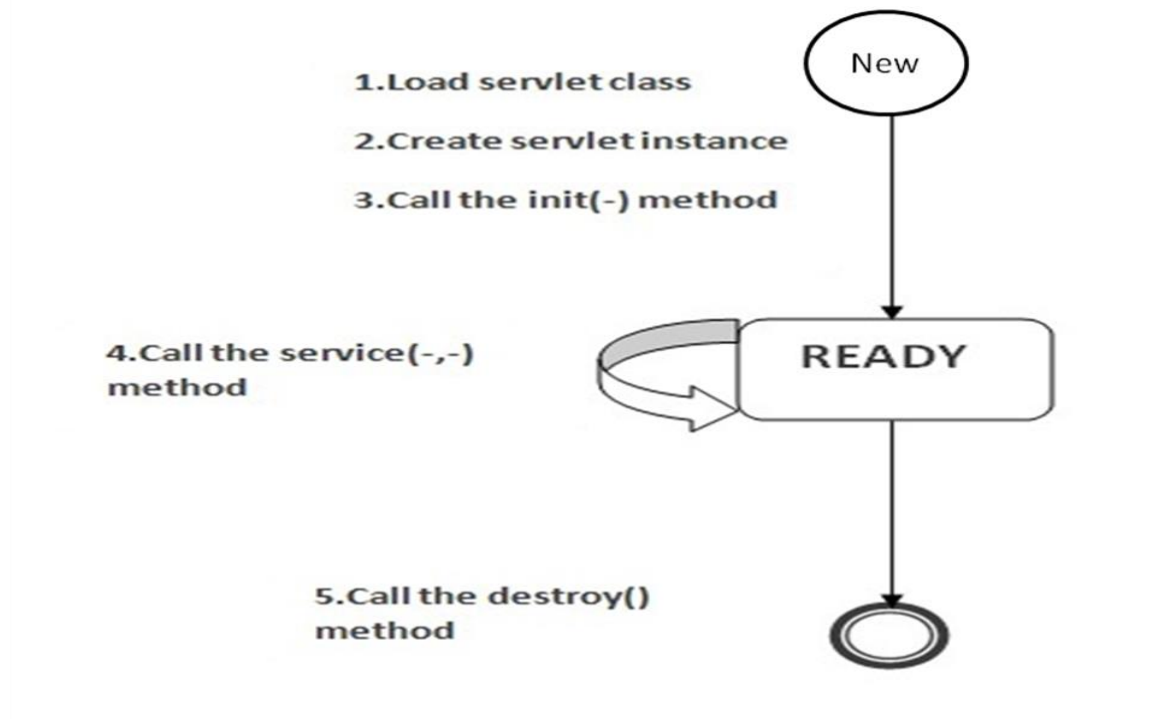
There are many content types:

- text/html
- text/plain
- application/msword
- application/vnd.ms-excel
- application/jar
- application/pdf
- application/octet-stream
- application/x-zip
- images/jpeg
- video/quicktime etc.

❖ Life Cycle of a Servlet (Servlet Life Cycle)

The web container maintains the life cycle of a servlet instance. Let's see the life cycle of the servlet:

1. Servlet class is loaded.
2. Servlet instance is created.
3. init method is invoked.
4. service method is invoked.
5. destroy method is invoked.



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.

1) Servlet class is loaded

- The class loader 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.

2) 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.

3) 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 :**

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

4) service() method is invoke

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 already 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
```

5) 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()
```

❖ Difference between Applet and servlet

No .	Applets	Servlets
1	It is Part of Core Java	It is Part of Advance Java
02	It is Client Side Program	It is Server Side Program
3	Applet executes on the web browser.	Servlet executes on the web server in response to requests from a web browser
4	It can be used for client side data validation.	It can be used for server side data validation.
5	It has rich GUI	Servlet has no GUI.
6	Require compatible browser	Can generate HTML, JavaScript, Applet code
7	It is uses the resources of Client	It is Processed at Server, No Client Resources required
8	Jar files can be accessed and downloaded by the Client	No Access
9	Require JRE or Web browser`s plug-in to run	Require Java Enabled Web Server
10	In applet JVM varies with Browser .	It uses Constant JVM
11	Use More Network bandwidth as runs loads and executes on Client machine	Less Network Bandwidth as runs on Server and only Results sent to Client
12	It May have Security issue	It runs Under Server Security so much less security issues.

❖ Java Servlet Development kit (JSDK)

- JSDK is a package containing all the classes and interface needed to develop servlets.
- It also contains web server and servlet engine to test your creation.
- It contains **servlet runner** program. **Servlet runner** is a program that runs on your machine and allows user to test servlet without running web server. you must have to start **servletrunner.exe** file before execution.
- It includes another **jsdk.jar** file which includes the class information necessary to compile servlets.
- Here we are using JSDK 2.0 version for developing and testing servlets.

Installation of JSDK 2.0

1. Download and install JSDK 2.0 [download jsdk2.0-winn32.exe].Default directory is **C:\JSDK2.0**
2. To compile servlet we have to set following path:
 - a. **classpath="C:\JSDK2.0\lib\jsdk.jar**
 - b. **JAVA_HOME="C:\Program files\java\jdk1.0.17**
3. To start servletrunner.
Open cmd and goto path and execute,
C:\JSDK2.0\bin\servletrunner.exe
4. Create new file having name HelloServlet.java in following path:
C:\JSDK2.0\examples\HelloServlet.java
Add code and Compile this file using **javac HelloServlet.java**
5. Open **servlet.properties** file from **C:\JSDK2.0\examples\servlet.properties**

and add following code in that file:

```
# Hello servlet
Servlet.HelloServlet.code=HelloServlet
```

6. Open browser and type and go

http:\\ localhost:8080\\servlet\\HelloServlet

directory structure for Servlet code using JSDK

```
C:\\ JSDK2.0
|---examples
|----HelloServlet.java
|---servlet.properties
```

Example-1

File :HelloServlet.java

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

public class First extends GenericServlet
{

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

        res.setContentType("text/html");

        PrintWriter out=res.getWriter();
        out.print("<html><body>");
        out.print("<b>hello world </b>");
        out.print("</body></html>");

    }
}
```

❖ Servlet API

- It provides the facility of implementing server side programming written in java.
- Servlet API allows to create and implement server side programs known as servlet.
- It is not part of JDK ,but it is supported by Apache Tomcat.
- It consists of two packages **javax.servlet** and **javax.servlet.http** a package which represents interfaces and classes for servlet API.
- The **javax.servlet** package contains many interfaces and classes that are used by the servlet or web container. These are not specific to any protocol.
- The **javax.servlet.http** package contains interfaces and classes that are responsible for http requests only.

1. Interfaces in javax.servlet package
2. Classes in javax.servlet package
3. Interfaces in javax.servlet.http package
4. Classes in javax.servlet.http package

✓ Interfaces in javax.servlet package

There are many interfaces in javax.servlet package. They are as follows:

1. **Servlet** : it is responsible for managing life cycle of servlet. It defines `init()`, `service()` and `destroy()`.
2. **ServletRequest**: to read data from client request .It defines an object to provide client request information to a servlet.
3. **ServletResponse**: to write data to a client response. It defines an object to assist a servlet in sending a response to the client.
4. **ServletConfig** :it allows servlets to get initialization parameter. Its object is used by servlet container to pass information to servlet during initialization.
5. **ServletContext**: defines set of methods that a servlet uses to communicate with its servlet container.
6. `RequestDispatcher`
7. `SingleThreadModel`
8. `Filter`
9. `FilterConfig`
10. `FilterChain`
11. `ServletRequestListener`
12. `ServletRequestAttributeListener`
13. `ServletContextListener`
14. `ServletContextAttributeListener`

✓ Servlet Interface

- **Servlet interface** provides common behaviour to all the servlets.
- Servlet interface needs to be implemented for creating any servlet (either directly or indirectly).
- It provides 3 life cycle methods that are used to initialize the servlet, to service the requests, and to destroy the servlet and 2 non-life cycle methods.

Methods of Servlet interface

There are 5 methods in Servlet interface. The init, service and destroy are the life cycle methods of servlet. These are invoked by the web container.

Method	Description
public void init (ServletConfig config)	Initializes the servlet. It is the life cycle method of servlet and invoked by the web container only once.
public void service (ServletRequest request, ServletResponse response)	provides response for the incoming request. It is invoked at each request by the web container.
public void destroy ()	is invoked only once and indicates that servlet is being destroyed.
public ServletConfig getServletConfig ()	returns the object of ServletConfig.
public String getServletInfo ()	returns information about servlet such as writer, copyright, version etc.

❖ Classes in javax.servlet package

There are many classes in javax.servlet package. They are as follows:

1. **GenericServlet** : it is used to define a generic, protocol independent servlet.
2. **ServletInputStream** : provides an input stream for reading binary data from a client request, including an efficient readLine method for reading data one at a time.
3. **ServletOutputStream**: provides an output stream for sending binary data to client.
4. **ServletException**: defines general exception a servlet can throw when it encounters difficulty indicates servlet error occurred.
5. **UnavailableException**
6. **ServletRequestWrapper**
7. **ServletResponseWrapper**
8. **ServletRequestEvent**
9. **ServletContextEvent**
10. **ServletRequestAttributeEvent**
11. **ServletContextAttributeEvent**
12. **UnavailableException**

✓ **GenericServlet class**

- It implements **Servlet**, **ServletConfig** and **Serializable** interfaces. It provides the implementation of all the methods of these interfaces except the service method.
- GenericServlet class can handle any type of request so it is protocol-independent.

You may create a generic servlet by inheriting the GenericServlet class and providing the implementation of the service method.

- **public void init(ServletConfig config)** is used to initialize the servlet.
- **public abstract void service(ServletRequest request, ServletResponse response)** provides service for the incoming request. It is invoked at each time when user requests for a servlet.
- **public void destroy()** is invoked only once throughout the life cycle and indicates that servlet is being destroyed.
- **public ServletConfig getServletConfig()** returns the object of ServletConfig.
- **public String getServletInfo()** returns information about servlet such as writer, copyright, version etc.
- **public ServletContext getServletContext()** returns the object of ServletContext.
- **public String getInitParameter(String name)** returns the parameter value for the given parameter name.
- **public Enumeration getInitParameterNames()** returns all the parameters defined in the web.xml file.
- **public String getServletName()** returns the name of the servlet object.
- **public void log(String msg)** writes the given message in the servlet log file.
- **public void log(String msg, Throwable t)** writes the explanatory message in the servlet log file and a stack trace.

✓ **Interfaces in javax.servlet.http package**

There are many interfaces in javax.servlet.http package. They are as follows:

- **HttpServletRequest**: enables servlets to read data to an HTTP request.
- **HttpServletResponse**: enables servlets to write data from an HTTP request
- **HttpSession**: allows session data to be read and written.
- **HttpSessionBindingListener**: informs an object, it is bound or unbound from a session.
- **HttpSessionListener**:
- **HttpSessionAttributeListener**
- **HttpSessionActivationListener**
- **HttpSessionContext** (deprecated now)

✓ **Classes in javax.servlet.http package**

There are many classes in javax.servlet.http package. They are as follows:

- **HttpServlet**: provides methods to handle HTTP request and responses.
- **Cookie** : allows state information to be stored on a client machine.
- **HttpSessionEvent**: encapsulates a session changed event.
- **HttpSessionBindingEvent**: indicates when a listener is bound to or unbound from a session value or that a session attribute changed.
- **HttpServletRequestWrapper**
- **HttpServletResponseWrapper**

HttpServlet class

- If you want to develop http specific servlet than HttpServlet class is used.
- The HttpServlet class extends the GenericServlet class and implements Serializable interface.
- **Methods of HttpServlet class**
- `public void service(ServletRequest req,ServletResponse res)` dispatches the request to the protected service method by converting the request and response object into http type.
- `protected void service(HttpServletRequest req, HttpServletResponse res)` receives the request from the service method, and dispatches the request to the `doXXX()` method depending on the incoming http request type.

Methods :

- `doGet(req, res)` handles the GET request.
- `doPost(req, res)` handles the POST request
- `doHead(req, res)` handles the HEAD request.
- `doOptions(req, res)` handles the OPTIONS request.
- `doPut(req, res)` handles the PUT request.
- `doTrace(req, res)` handles the TRACE request. It is invoked by the web container.
- `doDelete(req, res)` handles the DELETE request. It is invoked by the web container.
- `protected long getLastModified(req)` returns the time when HttpServletRequest was last modified since midnight January 1, 1970 GMT.

❖ XML configuration in Tomcat

- The web.xml file is known as deployment descriptor. It is a file from which Web container gets the information about the servlet to be invoked.
- Web.xml file is required for every application deployed on Tomcat. It must be located in WEB-INF directory of your root application.
- Web.xml file having many elements which are used to:
 - Define servlet specific information such as servlet name ,servlet class, description ,URL pattern, parameter name ,parameter value etc.
 - Defining error page
 - Security notes
 - Declaring roles, tag libraries
 - Define session configuration
- The web.xml file needs at least an opening and closing **<web-app>** tag.

Elements of web.xml file

No.	Element	Purpose
1	<web-app>	<ul style="list-style-type: none"> • It is the root element • All elements of web.xml file are contained inside this element. • Web.xml file needs at least opening and closing <web-app> tag,
2	<servlet>	<ul style="list-style-type: none"> • It is used to specify servlet specific information. • It is container tag for <servlet-name>, <servlet-class>,<description>,<init-param> elements. • <servlet-name>, <servlet-class> must be defined in each file.
3	<servlet-name>	<ul style="list-style-type: none"> • It specifies servlet name. • It can be any name but it must be same for the <servlet> tag which defines the servlet class and <servlet-mapping> which defines URL pattern.
4	<servlet-class>	<ul style="list-style-type: none"> • It is used to specify name of the servlet class file which is generated by compiling servlet source code. • It is sub element of <servlet>
5	<description>	<ul style="list-style-type: none"> • It defines description for the servlet. • It is optional
6	<servlet-mapping>	<ul style="list-style-type: none"> • Used to specify a URL mapping for servlet that has been defined using <servlet> element. • It contains two sub elements <servlet-name> and <url-pattern>. • <servlet-mapping> must be defined after <servlet> in web.xml.
7	<url-pattern>	<ul style="list-style-type: none"> • It is used to specify the url for the servlet.

8	<init-param>	<ul style="list-style-type: none"> • Used to specify parameter for servlet. • For every parameter separate <init-param> should be defined. • It contains <param-name> and <param-value> elements. • It is written inside <servlet>.
9	<param-name>	<ul style="list-style-type: none"> • Specifies name of servlet parameter.
10	<param-value>	<ul style="list-style-type: none"> • Specifies value of the servlet parameter
11	<load-on-startup>	<ul style="list-style-type: none"> • Used to specify that the servlet should be loaded automatically when web application is started. • You can specify loading priority for servlet. • Smaller value have high priority .

Simple web.xml file

```
<web-app xmlns="http://java.sun.com/xml/ns/javaee" version="2.5">
    <servlet>
        <servlet-name>test</servlet-name>
        <servlet-class>MyServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>test</servlet-name>
        <url-pattern>/dhruvi</url-pattern>
    </servlet-mapping>
</web-app>
```

❖ Apache server

- Apache is a *very* popular server
 - 66% of the web sites on the Internet use Apache
- Apache is:
 - Full-featured and extensible
 - Efficient
 - Robust
 - Secure (at least, more secure than other servers)
 - Up to date with current standards
 - Open source
 - Free

❖ Tomcat

- Tomcat is the Servlet Engine than handles servlet requests for Apache
 - Tomcat is a “helper application” for Apache
 - It’s best to think of Tomcat as a “servlet container”
- Apache can handle many types of web services
 - Apache can be installed without Tomcat
 - Tomcat can be installed without Apache
- It’s easier to install Tomcat standalone than as part of Apache
 - By itself, Tomcat can handle web pages, servlets, and JSP
- Apache and Tomcat are open source (and therefore free)

Follow this steps to develop and run servlet

1. Create **demo** folder in **webapps** directory of apache tomcat folder.
2. Add **WEB-INF** folder and **index.html** file in **demo** directory
3. Add **classes** folder and **web.xml** file in **WEB-INF** folder.
4. Create **First.java** file in **classes** folder, add code and compile that file to create class file
5. Edit **web.xml** file and add necessary elements.
6. Start tomcat server.
7. Open browser and type
http://localhost:8000/demo/First

❖ Servlet Example by implementing Servlet interface

Example-2

Write a servlet program to display hello message.

File: First.java

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

public class First implements Servlet
{
    ServletConfig config=null;

    public void init(ServletConfig config)
    {
        this.config=config;
        System.out.println("servlet is initialized");
    }

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

        res.setContentType("text/html");

        PrintWriter out=res.getWriter();
        out.print("<html><body>");
        out.print("<b>hello simple servlet</b>");
        out.print("</body></html>");

    }
    public void destroy()
    {
        System.out.println("servlet is destroyed");
    }
}
```

```
    }  
    public ServletConfig getServletConfig()  
    {  
        return config;  
    }  
    public String getServletInfo()  
    {  
        return "copyright 2007-1010";  
    }  
  
}
```

❖ Servlet Example by inheriting the GenericServlet class

Example-3 Write a servlet program to print Hello world.

File: First.java

```
import java.io.*;  
import javax.servlet.*;  
  
public class First extends GenericServlet  
{  
    public void service(ServletRequest req,ServletResponse res)  
    throws IOException,ServletException  
    {  
  
        res.setContentType("text/html");  
  
        PrintWriter out=res.getWriter();  
        out.print("<html><body>");  
        out.print("<b>hello generic servlet</b>");  
        out.print("</body></html>");  
    }  
}
```

File: web.xml

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

Steps to create the servlet using Tomcat server

1. Create a directory structure
2. Create a Servlet
3. Compile the Servlet
4. Create a deployment descriptor
5. Start the server and deploy the application

The servlet example can be created by three ways:

1. By implementing **Servlet** interface,
2. By inheriting **GenericServlet** class, (or)
3. By inheriting **HttpServlet** class

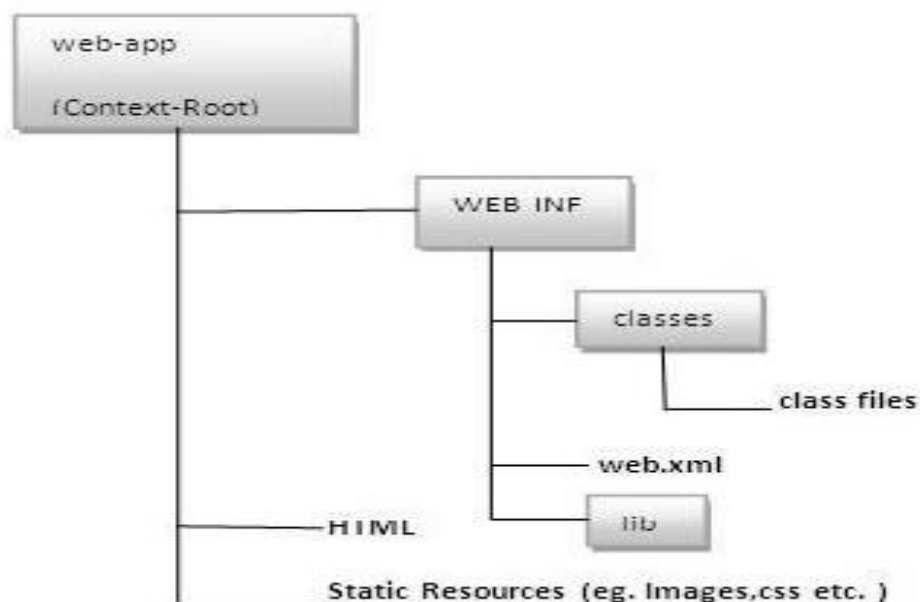
The mostly used approach is by extending HttpServlet because it provides http request specific method such as doGet(), doPost(), doHead() etc.

Here, we are going to use **apache tomcat server** in this example. The steps are as follows:

1. Create a directory structure
2. Create a Servlet
3. Compile the Servlet
4. Create a deployment descriptor
5. Start the server and deploy the project
6. Access the servlet

1) Create a directory structures

- The **directory structures** define that where to put the different types of files so that web container may get the information and respond to the client.
- The Sun Microsystems defines a unique standard to be followed by all the server vendors. Let's see the directory structure that must be followed to create the servlet.



As you can see that the servlet class file must be in the classes folder. The web.xml file must be under the WEB-INF folder.

2) Create a Servlet

There are three ways to create the servlet.

1. By implementing the Servlet interface
 2. By inheriting the GenericServlet class
 3. By inheriting the HttpServlet class
- The HttpServlet class is widely used to create the servlet because it provides methods to handle http requests such as doGet(), doPost(), doHead() etc.

DemoServlet.java

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

public class DemoServlet extends HttpServlet
{
    public void doGet(HttpServletRequest req,HttpServletResponse res)
        throws ServletException,IOException
    {
        res.setContentType("text/html");//setting the content type
        PrintWriter pw=res.getWriter();//get the stream to write the data

        //writing html in the stream
        pw.println("<html><body>");
        pw.println("Welcome to servlet");
        pw.println("</body></html>");

        pw.close();//closing the stream
    }
}
```

3) Compile the servlet

For compiling the Servlet, jar file is required to be loaded. Different Servers provide different jar files:

Jar file	Server
1) servlet-api.jar	Apache Tomcat
2) weblogic.jar	Weblogic
3) javaee.jar	Glassfish
4) javaee.jar	JBoss

Two ways to load the jar file

1. set classpath=" C:\apache-tomcat-7.0.29\lib\servlet-api.jar"
2. paste the jar file in JRE/lib/ext folder

Put the java file in any folder. After compiling the java file, paste the class file of servlet in **WEB-INF/classes** directory.

4) Create the deployment descriptor (web.xml file)

- The **deployment descriptor** is an xml file, from which Web Container gets the information about the servlet to be invoked.
- The web container uses the Parser to get the information from the web.xml file. There are many xml parsers such as SAX, DOM and Pull.
- There are many elements in the web.xml file.

web.xml file

```
<web-app>

<servlet>
<servlet-name>Demo</servlet-name>
<servlet-class>DemoServlet</servlet-class>
</servlet>

<servlet-mapping>
<servlet-name>Demo</servlet-name>
<url-pattern>/welcome</url-pattern>
</servlet-mapping>

</web-app>
```

5) Start the Server and deploy the project

To start Apache Tomcat server, double click on the startup.bat file under apache-tomcat/bin directory.

Set JAVA_HOME in environment variable?

To start Apache Tomcat server JAVA_HOME and JRE_HOME must be set in Environment variables.

Go to My Computer properties -> Click on advanced tab then environment variables -> Click on the new tab of user variable -> Write JAVA_HOME in variable name and paste the path of jdk folder in variable value -> ok -> ok -> ok.

There must not be semicolon (;) at the end of the path.

After setting the JAVA_HOME double click on the startup.bat file in apache tomcat/bin.

Note: There are two types of tomcat available:

1. Apache tomcat that needs to extract only (no need to install)
2. Apache tomcat that needs to install

It is the example of apache tomcat that needs to extract only.

Stratup.bat file

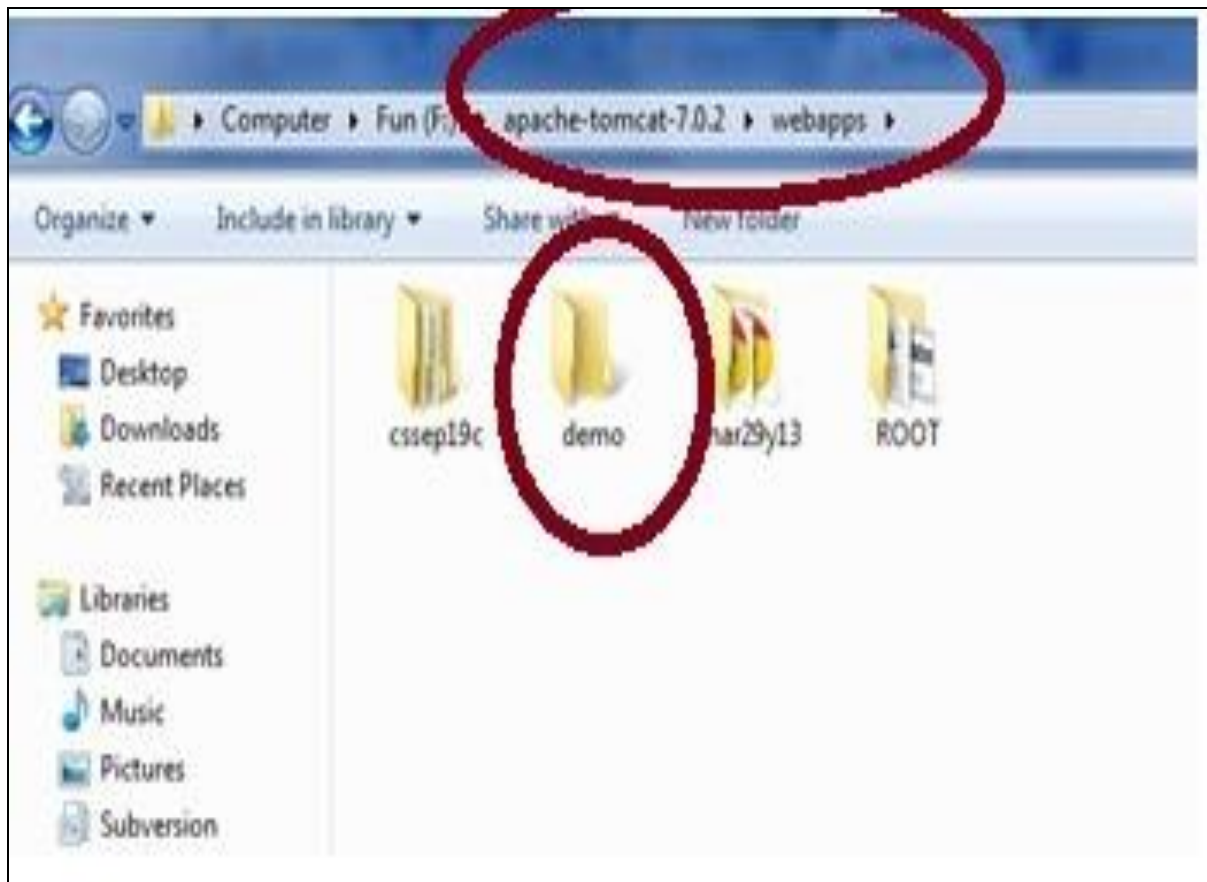
Now server is started successfully.

To change port number of apache tomcat

- Changing the port number is required if there is another server running on the same system with same port number. Suppose you have installed oracle, you need to change the port number of apache tomcat because both have the default port number 8080.
- Open server.xml file in notepad. It is located inside the **apache-tomcat/conf** directory. Change the Connector port = 8080 and replace 8080 by any four digit number instead of 8080. Let us replace it by 9999 and save this file.

5) How to deploy the servlet project

Copy the project and paste it in the webapps folder under apache tomcat.



But there are several ways to deploy the project. They are as follows:

- By copying the context(project) folder into the webapps directory
- By copying the war folder into the webapps directory
- By selecting the folder path from the server
- By selecting the war file from the server

Here, we are using the first approach.

You can also create war file, and paste it inside the webapps directory. To do so, you need to use jar tool to create the war file. Go inside the project directory (before the WEB-INF), then write:

```
projectfolder> jar cvf myproject.war *
```

Creating war file has an advantage that moving the project from one location to another takes less time.

6) How to access the servlet

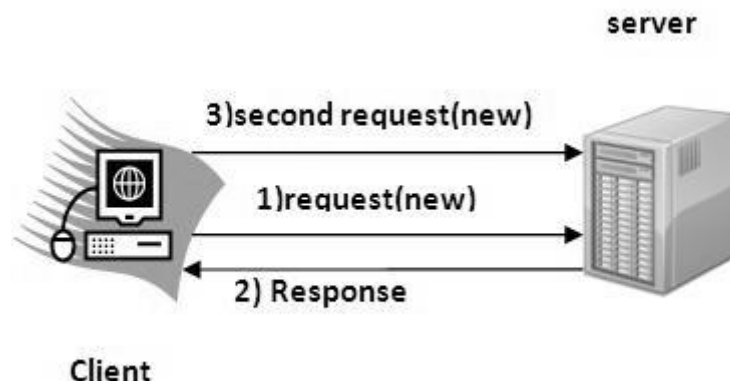
Open browser and write `http://hostname:portno/contextroot/urlpatternofservlet`. For example:

```
http://localhost:9999/demo/welcome
```

❖ Session Tracking in Servlets

1. Session Tracking
2. Session Tracking Techniques

- **Session** means a particular interval of time.
- **Session Tracking** is a way to maintain state (data) of an user. It is also known as **session management** in servlet.
- Http protocol is a stateless so we need to maintain state using session tracking techniques. Each time user requests to the server, server treats the request as the new request. So we need to maintain the state of an user to recognize to particular user.
- HTTP is stateless that means each request is considered as the new request. It is shown in the figure given below:



Why use Session Tracking?

To recognize the user It is used to recognize the particular user.

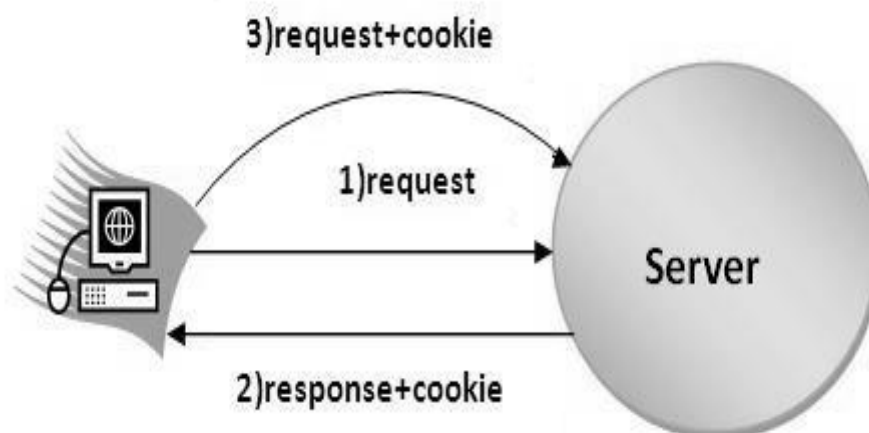
Session Tracking Techniques

There are four techniques used in Session tracking:

1. **Cookies**
2. **Hidden Form Field**
3. **URL Rewriting**
4. **HttpSession**

✓ Cookies in Servlet

- A **cookie** is a small piece of information that is persisted between the multiple client requests.
- A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.
- By default, each request is considered as a new request.
- In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.



- **Types of Cookie**

There are 2 types of cookies in servlets.

1. Non-persistent cookie
2. Persistent cookie

Non-persistent cookie

- It is **valid for single session** only. It is removed each time when user closes the browser.

Persistent cookie

- It is **valid for multiple sessions**. It is not removed each time when user closes the browser. It is removed only if user logout or sign out.

Advantage of Cookies

1. Simplest technique of maintaining the state.
2. Cookies are maintained at client side.

Disadvantage of Cookies

1. It will not work if cookie is disabled from the browser.
2. Only textual information can be set in Cookie object.

✓ Cookie class

javax.servlet.http.Cookie class provides the functionality of using cookies. It provides a lot of useful methods for cookies.

Constructor of Cookie class

Constructor	Description
Cookie()	constructs a cookie.
Cookie(String name, String value)	constructs a cookie with a specified name and value.

Useful Methods of Cookie class

There are given some commonly used methods of the Cookie class.

Method	Description
public void setMaxAge(int expiry)	Sets the maximum age of the cookie in seconds.
public String getName()	Returns the name of the cookie. The name cannot be changed after creation.
public String getValue()	Returns the value of the cookie.
public void setName(String name)	changes the name of the cookie.
public void setValue(String value)	changes the value of the cookie.

Other methods required for using Cookies

For adding cookie or getting the value from the cookie, we need some methods provided by other interfaces. They are:

1. **public void addCookie(Cookie ck):** method of HttpServletResponse interface is used to add cookie in response object.
2. **public Cookie[] getCookies():** method of HttpServletRequest interface is used to return all the cookies from the browser.

How to create Cookie?

```
Cookie ck=new Cookie("user","obj");//creating cookie object  
response.addCookie(ck);//adding cookie in the response
```

How to delete Cookie?

Let's see the simple code to delete cookie. It is mainly used to logout or signout the user.

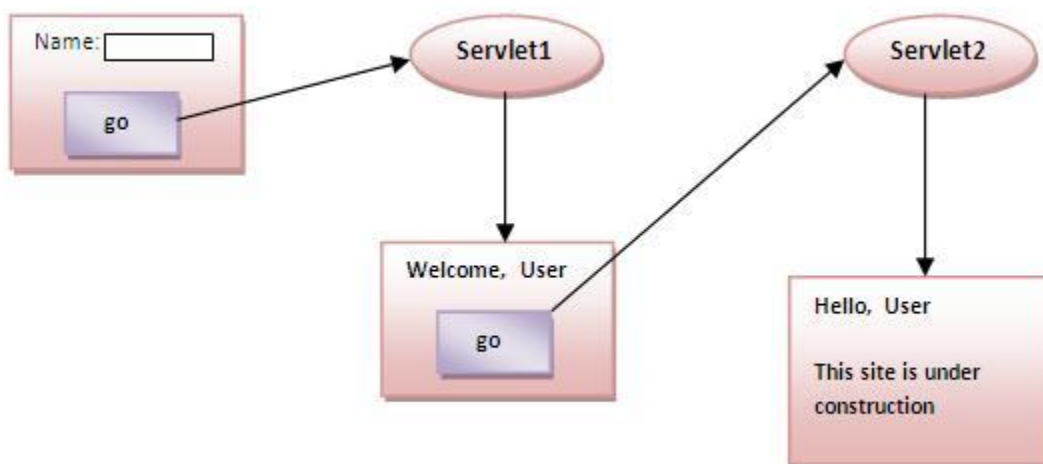
```
Cookie ck=new Cookie("user","");//deleting value of cookie  
ck.setMaxAge(0);//changing the maximum age to 0 seconds  
response.addCookie(ck);//adding cookie in the response
```

How to get Cookies?

Let's see the simple code to get all the cookies.

```
Cookie ck[]=request.getCookies();  
for(int i=0;i<ck.length;i++)  
{  
    out.print("<br>" +ck[i].getName()+" "+ck[i].getValue());//printing name and value of cookie  
}
```

Simple example of Servlet Cookies



In this example, we are storing the name of the user in the cookie object and accessing it in another servlet. As we know well that session corresponds to the

particular user. So if you access it from too many browsers with different values, you will get the different value.

Example-4 Write a program which use cookies and display welcome message for user in servlet.

Directory structure

Web-apps

```
|
  cookie
    |----WEB-INF-----classes--|-----FirstServlet.java
    |                               |-----SecondServlet.java
    |                               |
    |---index.html      web.xml
```

index.html

```
<form action="servlet1" method="post">
  Name:<input type="text" name="userName"/>    <br/>
  <input type="submit" value="go"/>
</form>
```

FirstServlet.java

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

public class FirstServlet extends HttpServlet
{

    public void doPost(HttpServletRequest request, HttpServletResponse response)
    {
        try
        {
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();

            String n=request.getParameter("userName");
            out.print("Welcome "+n);
        }
    }
}
```

```
Cookie ck=new Cookie("uname",n);           //creating cookie object
response.addCookie(ck);                     //adding cookie in the response
```

```
//creating submit button
```

```
out.print("<form action='servlet2'>");
out.print("<input type='submit' value='go'>");
out.print("</form>");
out.close();
}
catch(Exception e)
{
    System.out.println(e);
}
}
}
```

SecondServlet.java

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

public class SecondServlet extends HttpServlet
{
    public void doGet(HttpServletRequest request, HttpServletResponse response)
    {
        try
        {
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();
            Cookie ck[]=request.getCookies();
            out.print("Hello "+ck[0].getValue());
            out.close();
        }catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

web.xml

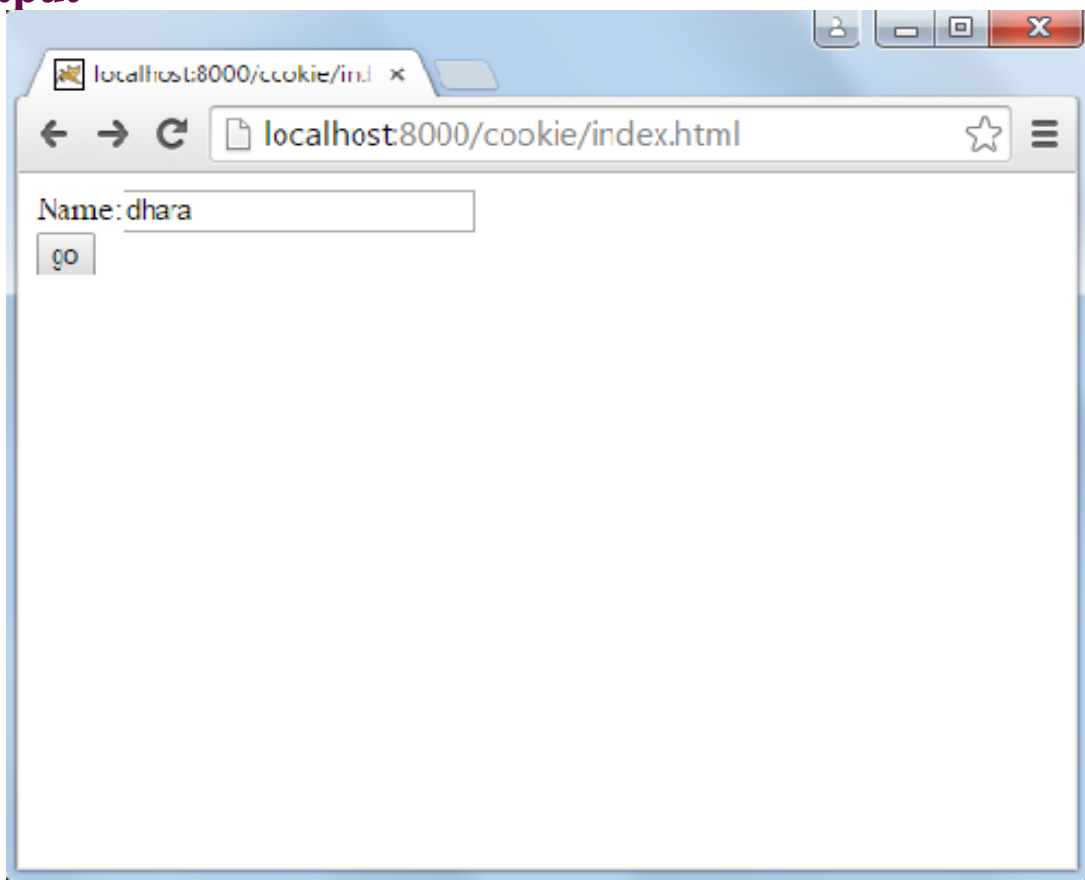
<web-app>

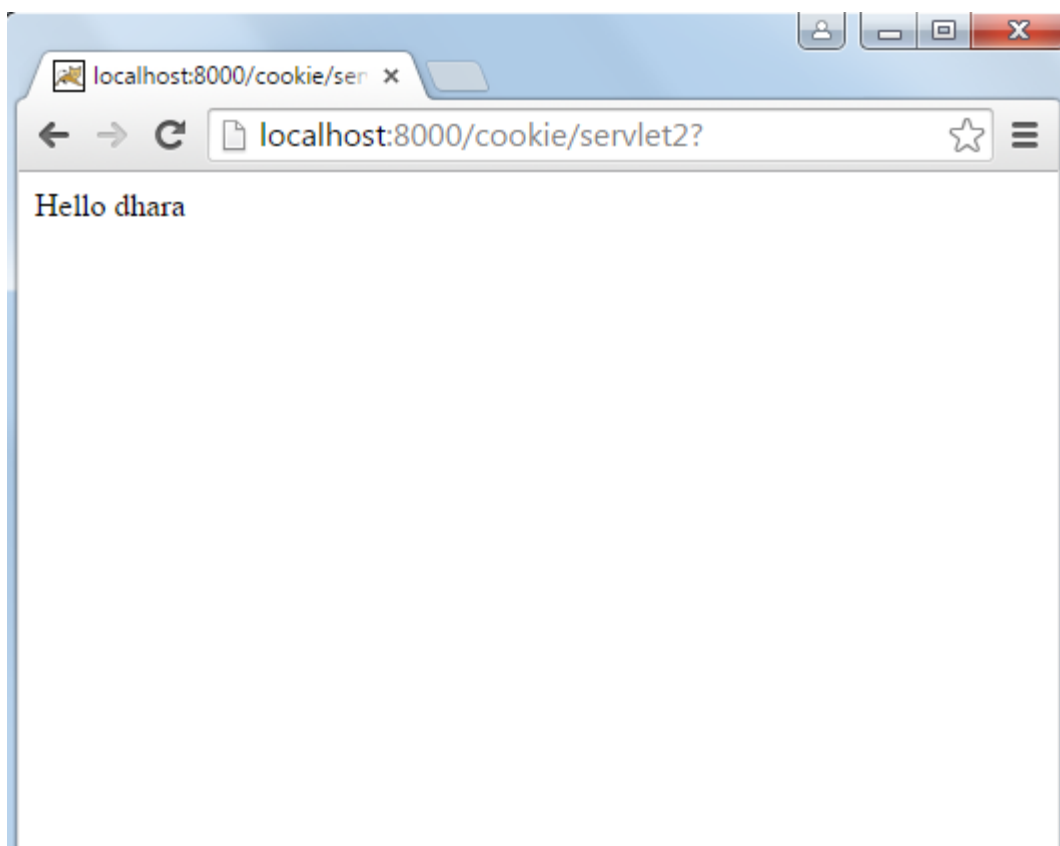
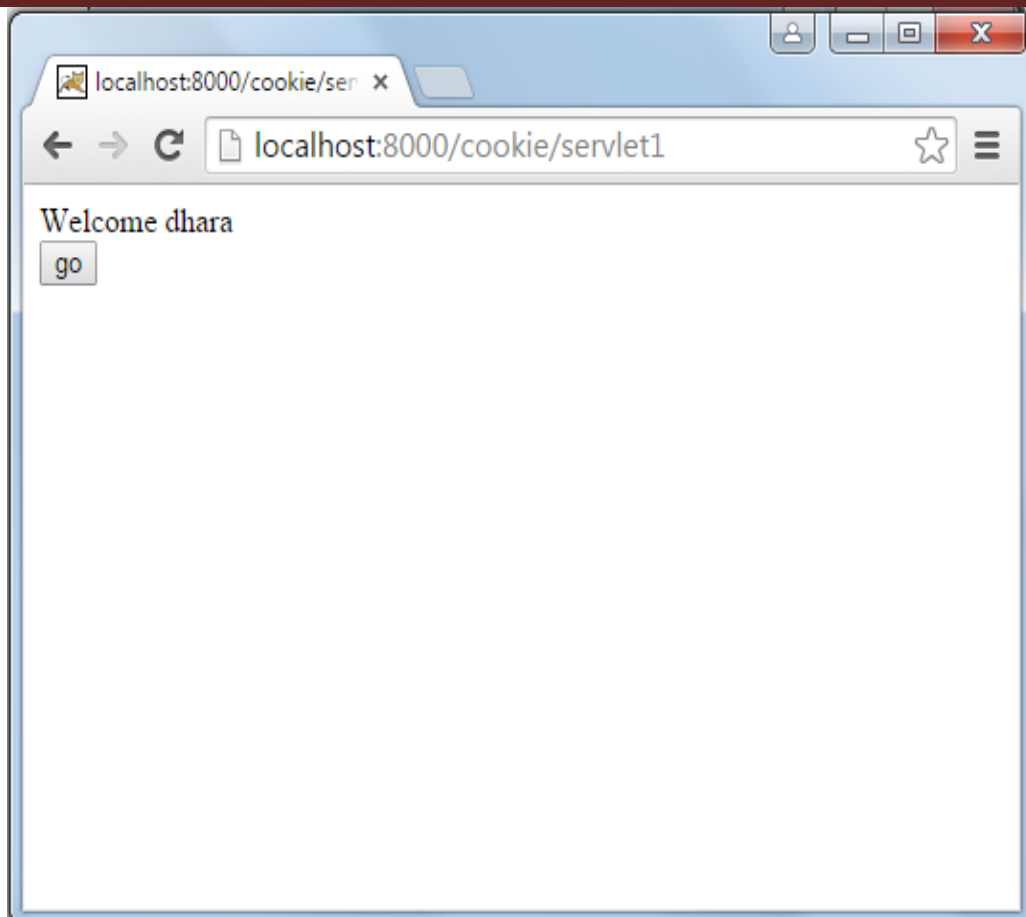
```
<servlet>
    <servlet-name>s1</servlet-name>
    <servlet-class>FirstServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>s1</servlet-name>
    <url-pattern>/servlet1</url-pattern>
</servlet-mapping>

<servlet>
    <servlet-name>s2</servlet-name>
    <servlet-class>SecondServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>s2</servlet-name>
    <url-pattern>/servlet2</url-pattern>
</servlet-mapping>
```

</web-app>

Output





Example-5 Write a program which uses session in Servlet.**index.html**

```
<html>
<head> <title> Login</title> </head>
<body>
    <form method="get" action="servlet1">
        Login page
        <table>
            <tr>
                <td> Login Id : </td>
                <td> <input type="text" name="t1" value=""/> </td>
            </tr>

            <tr>
                <td>Password : </td>
                <td> <input type="password" name="t2" value=""/> </td>
            </tr>
            <tr>
                <td><input type="submit" name="submit" value="Login"/> </td>
            </tr>
        </table>
    </form>
</body>
</html>
```

Session.java

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

public class Session extends HttpServlet
{
    public void doGet(HttpServletRequest request,HttpServletResponse response)
    throws IOException,ServletException
    {
        PrintWriter out = response.getWriter();
        HttpSession session = request.getSession();
        String n = request.getParameter("t1");
        session.setAttribute("name",n);

        out.print("<html>");
        out.print("<head> <title> Login</title> </head><body>");
    }
}
```

```
        out.print("<form method='get' action='servlet2'>");
        out.print("Your Name is " + n);
        out.print("<input type='submit' name='submit' value='Login'/>");
        out.print("</form>");
        out.print("</body>");
        out.print("</html>");
    }
}
```

Session1.java

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

public class Session1 extends HttpServlet
{

    public void doGet(HttpServletRequest request,HttpServletResponse response)
    throws IOException,ServletException
    {
        HttpSession session = request.getSession();
        PrintWriter out = response.getWriter();
        out.print("Value in Session[name] ::" +(String)session.getAttribute("name"));

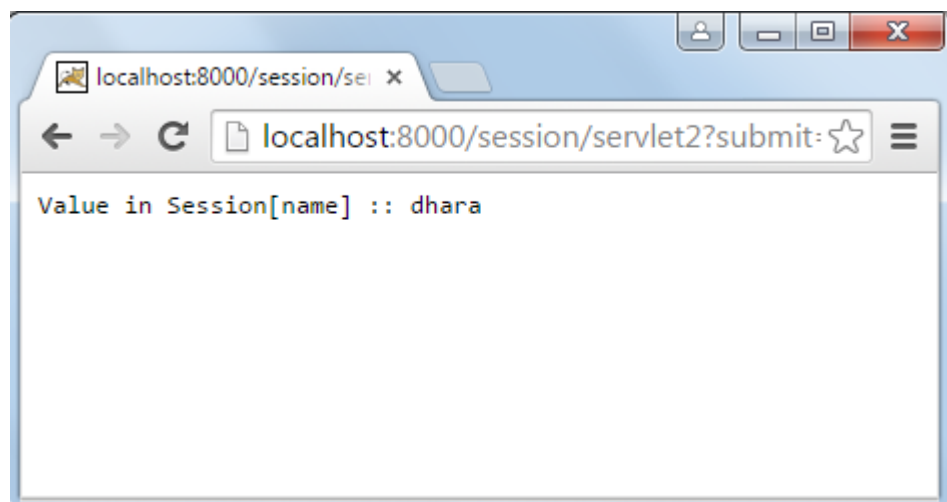
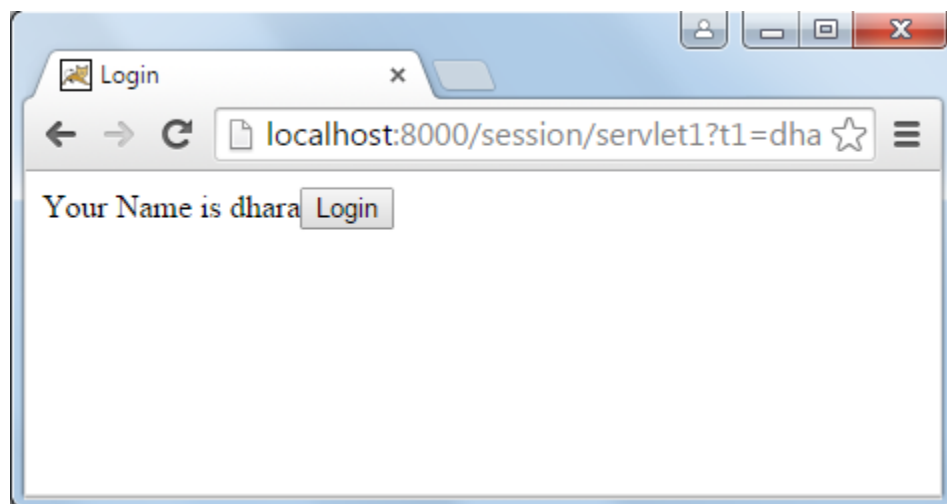
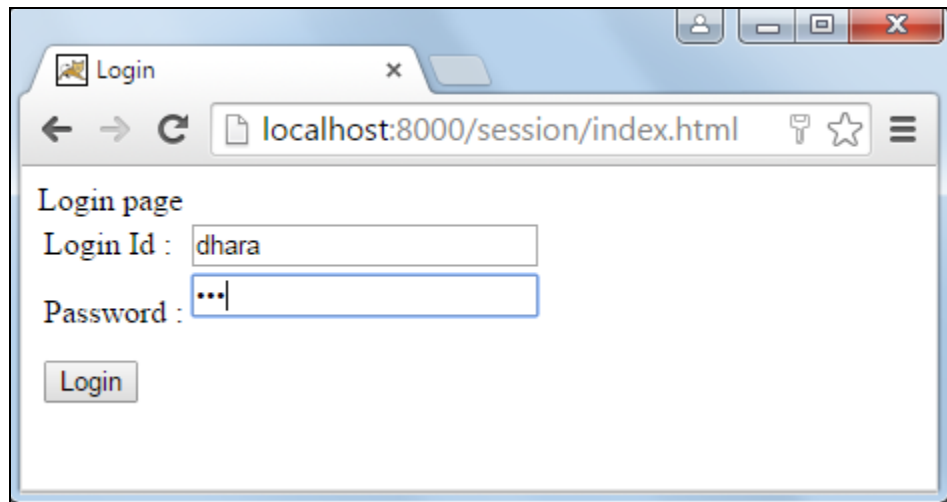
    }
}
```

web.xml

```
<web-app>
    <servlet>
        <servlet-name>sessiondemo</servlet-name>
        <servlet-class>Session</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>sessiondemo</servlet-name>
        <url-pattern>/servlet1</url-pattern>
    </servlet-mapping>

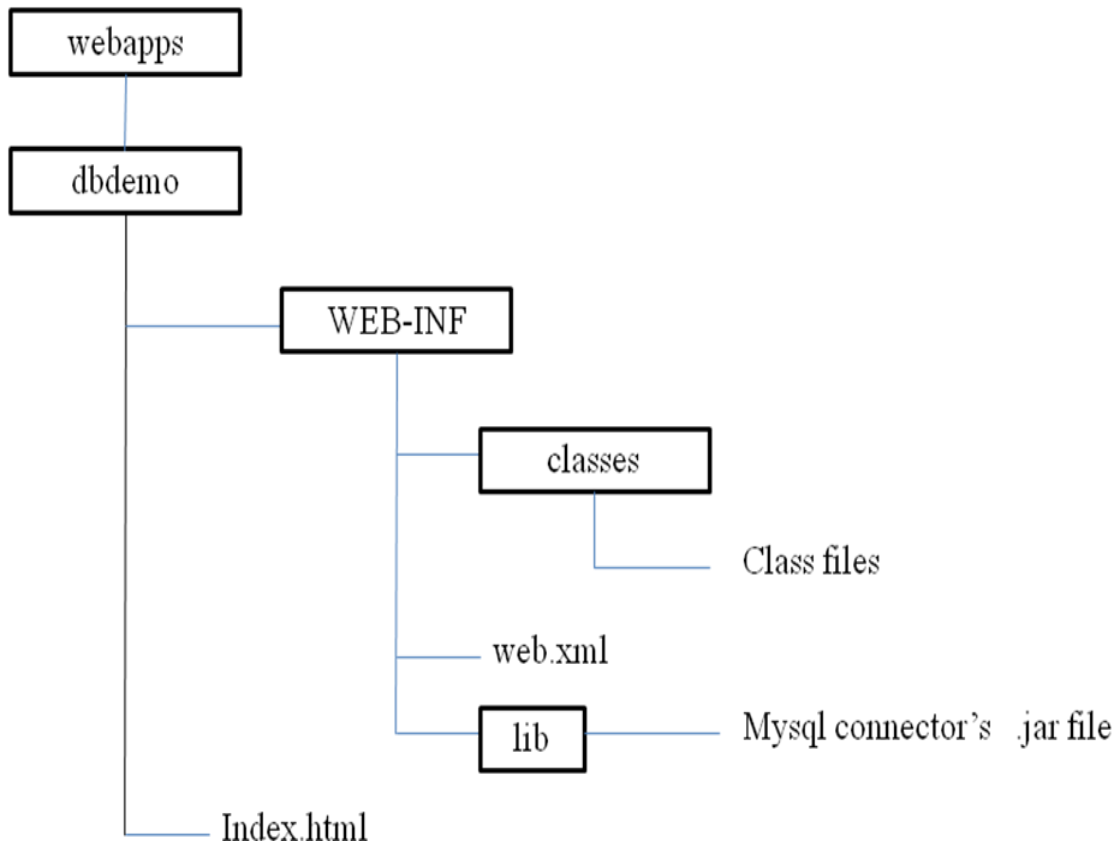
    <servlet>
        <servlet-name>sessiondemo1</servlet-name>
        <servlet-class>Session1</servlet-class>
    </servlet>
```

```
<servlet-mapping>  
    <servlet-name>sessiondemo1</servlet-name>  
    <url-pattern>/servlet2</url-pattern>  
</servlet-mapping>  
</web-app>
```



❖ The javax.servlet Package: Reading database/table records and displaying using servlet

- We can create database, access database tables, reading, writing or updating table using servlet programming.
- When we want to do servlet program using database, we must have to establish connection with database (for example Mysql) .
- To establish connection, copy **mysql-connector-java-5.0.7-bin.jar** driver file into **lib** folder of your **WEB-INF** directory where you have created your servlet.



Note: To perform database connection in this Servlet program you have to create table in your database named **stud** having column **id** and **name**.and **insert records into table**

Example-6 Write Servlet code which displays table data in output. **DisplayData.java**

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

```
public class DisplayData extends GenericServlet
{
    public void service(ServletRequest request, ServletResponse response)    throws
    ServletException, IOException
    {
        Connection cn;
        Statement stmt;
        PrintWriter out = response.getWriter();
        response.setContentType("text/html");

        try
        {
            Class.forName("com.mysql.jdbc.Driver");
            cn=DriverManager.getConnection("jdbc:mysql://localhost:3306/mydb","root", "");

            stmt=cn.createStatement();
            out.println("<b>"+ "-----Database connected-----"+ "</b>");
            // to create table
            //String sql="create table student1 (id int,name varchar(20))";
            //stmt.execute(sql);

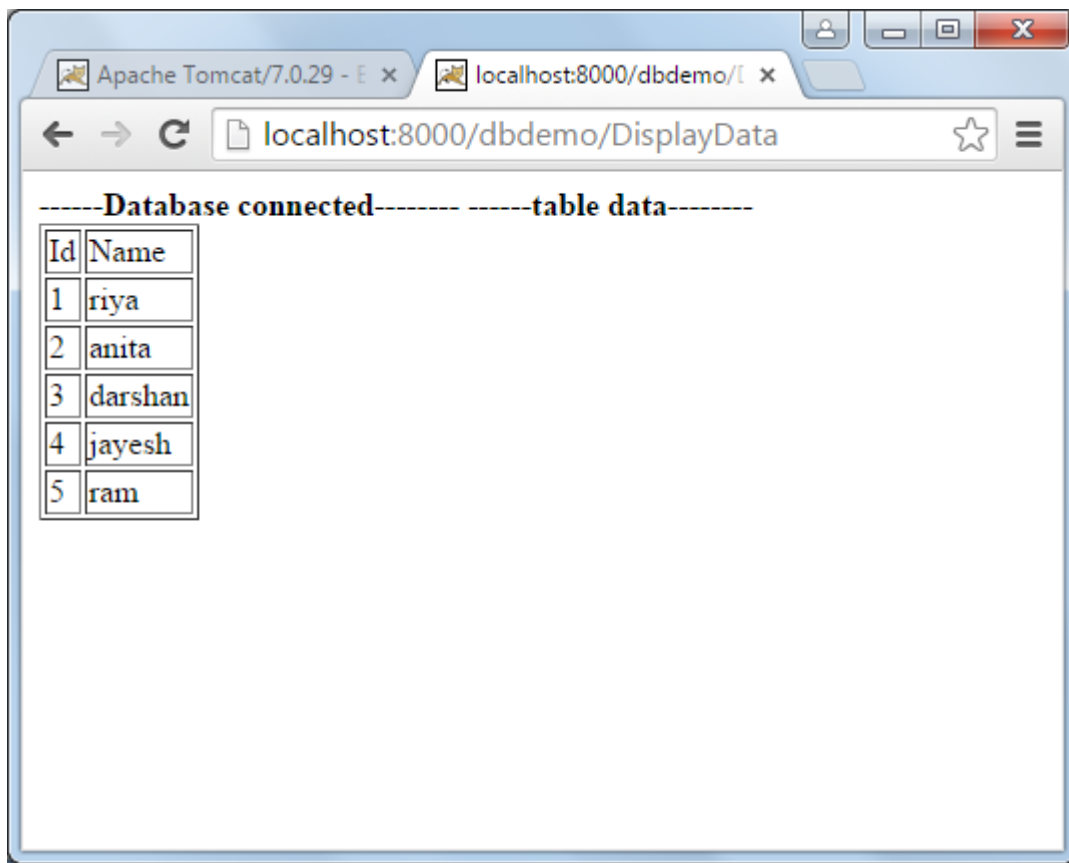
            String sql="select * from stud";
            ResultSet rs=stmt.executeQuery(sql);
            out.println("<b>"+ "-----table data-----"+ "</b>");
            out.println("<html><body><table border='1'>");
            out.println("<tr><td> Id </td>");
            out.println("<td>    Name </td></tr>");
            while(rs.next())
            {
                out.println("<tr>");
                out.print("<td>"+rs.getInt("id")+"</td>");
                out.print("<td>"+rs.getString("name")+"</td>");
                out.println("</tr>");
            }
            out.println("</table></body></html>");

            cn.close();
        }
        catch(Exception e)
        {
            System.out.println(e);
        }
    }
}
```

```
}
```

Web.xml file

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



Example-7 Write a servlet which displays today's date.

Dt.java

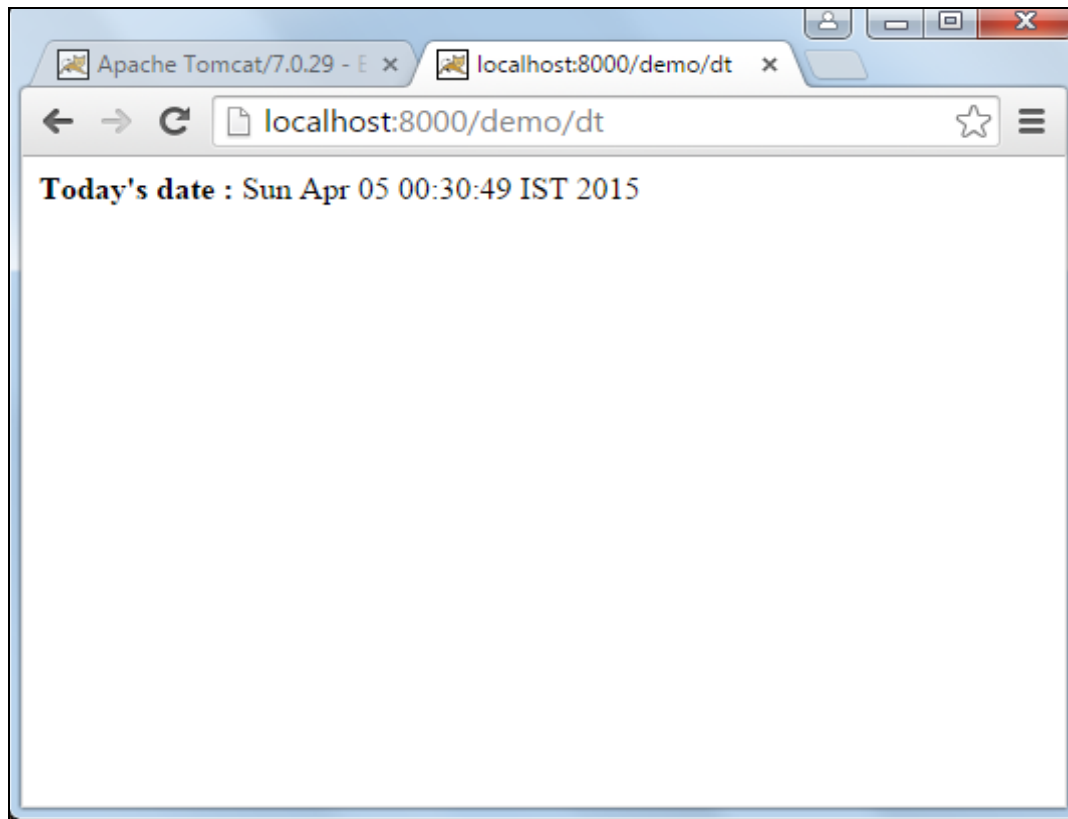
```
import java.util.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
```

```
public class dt extends GenericServlet
```

```
{  
    public void service(ServletRequest request, ServletResponse response)  
throws ServletException, IOException  
    {  
        PrintWriter out = response.getWriter();  
        response.setContentType("text/html");  
  
        Date today=new Date();  
        out.print("<b> Today's date : ");  
        out.print(today);  
    }  
}
```

Web.xml

```
<web-app>  
    <servlet>  
        <servlet-name>datedisp</servlet-name>  
        <servlet-class>dt</servlet-class>  
    </servlet>  
    <servlet-mapping>  
        <servlet-name>datedisp</servlet-name>  
        <url-pattern>/dt</url-pattern>  
    </servlet-mapping>  
</web-app>
```



Example-8 Write Servlet code which displays number of time user visited particular page.**PageCounter.java**

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

public class PageCounter extends HttpServlet
{
    String sCount;
    int count;

    public void init()
    {
        ServletConfig config=getServletConfig();
        sCount=config.getInitParameter("counter");
        count = Integer.parseInt(sCount);
    }
    public void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException,IOException
    {
        PrintWriter out = response.getWriter();
        count++;
        out.println("NO of Visitors = "+count);
    }
}
```

web.xml

```
<web-app>
    <servlet>
        <servlet-name>PageCounter</servlet-name>
        <servlet-class>PageCounter</servlet-class>
        <init-param>
            <param-name>counter</param-name>
            <param-value>0</param-value>
        </init-param>
    </servlet>

    <servlet-mapping>
        <servlet-name>PageCounter</servlet-name>
        <url-pattern>/PageCounter</url-pattern>
    </servlet-mapping>
</web-app>
```