Marwadi education foundation

Unit – 3
Servlets API

Prepared By
Prof. Ravikumar RN
Assistant Professor, CE Dept.

KNOWLEDGE IS THE CURRENCY
FOR THE 21st CENTURY

Contents

- Servlet Introduction,
- Servlet Life Cycle(SLC),
- Types of Servlet,
- Servlet Configuration with Deployment Descriptor,
- Working with ServletContext and ServletConfig Object,
- Attributes in Servlet,
- Response and Redirection using Request Dispacher and using sendRedirect Method,
- Filter API,
- Manipulating Responses using Filter API,
- Session Tracking: using Cookies,
- HTTPSession,
- Hidden Form Fields and URL Rewriting,
- Types of Servlet Event: ContextLevel and SessionLevel

Introduction to Servlets

Servlet technology is used to create web application (resides at server side and generates dynamic web page).

Servlet technology is robust and scalable because of java language. Before Servlet, CGI (Common Gateway Interface) scripting language was popular as a server-side programming language. But there was many disadvantages of this technology.

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

Scripting Language

Server-Side Scripting Language

PHP
ASP.NET
(C# OR Visual Basic)
C++
Java and JSP
Python
Ruby on Rails etc.

Server-side scripting is often used to provide a customized interface for the user.

Client-Side Scripting Language

JavaScript
VBScript
HTML (Structure)
CSS (Designing)
AJAX
jQuery etc.

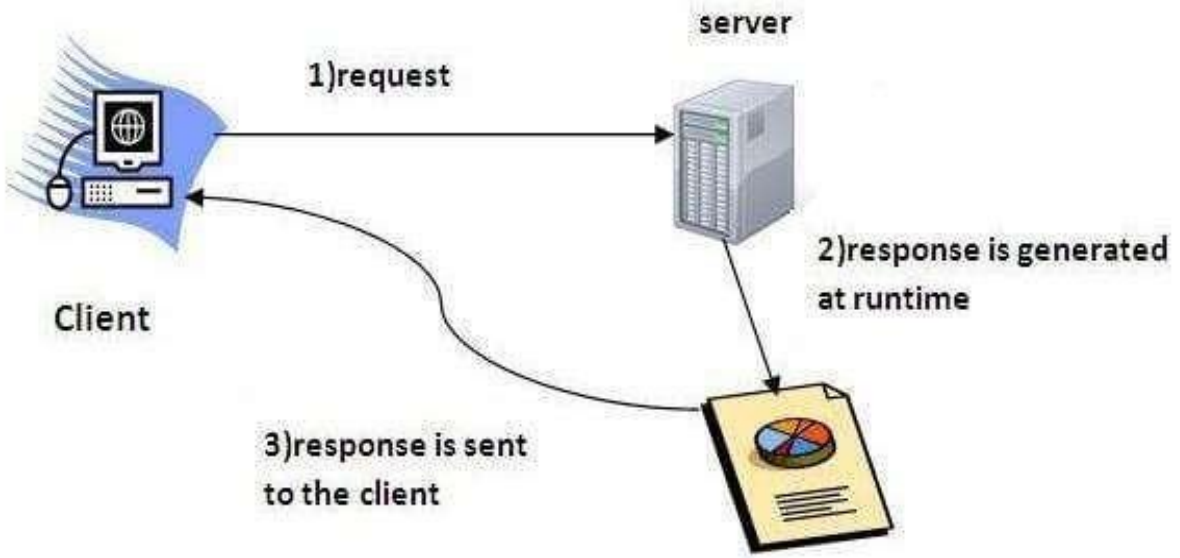
Client-side scripting is an important part of the Dynamic HTML. Usually run on client's browser.

What is Servlet?

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

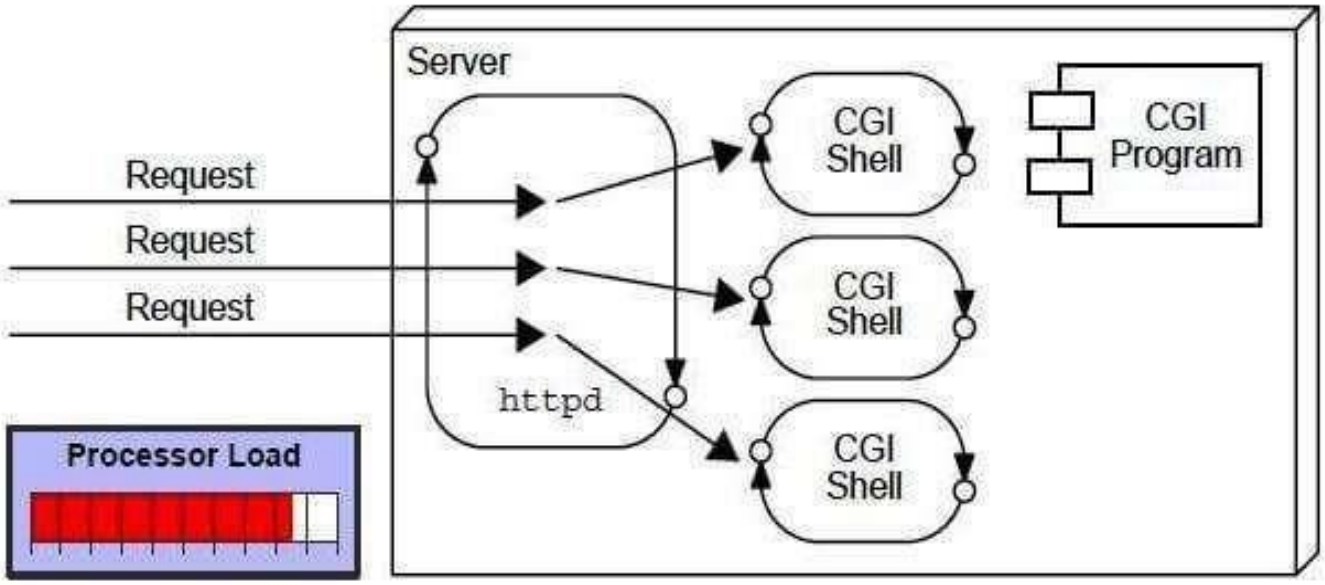
- Servlet is a technology i.e. used to create web application.
- Servlet is an API that provides many interfaces and classes including documentations.
- Servlet is an **interface** that must be implemented for **creating** any **servlet**.
- Servlet is a **class** that extends the capabilities of the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

Request and response programming model



Common Gateway Interface (CGI)

CGI technology enables the web server to call an external program and pass HTTP request information to the external program to process the request. **For each request, it starts a new process.**



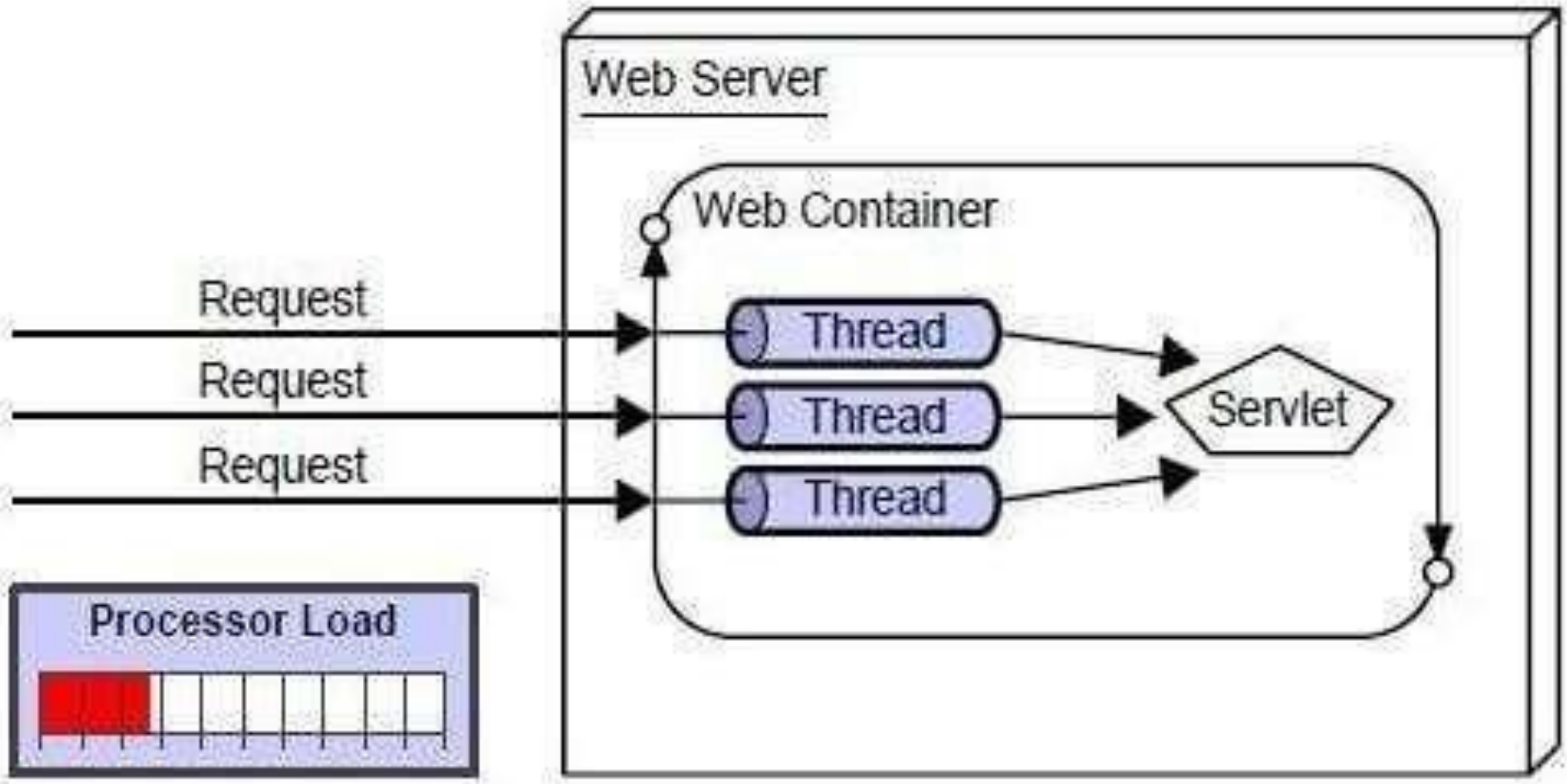
Common Gateway Interface (CGI)

Disadvantages of CGI

There are many problems in CGI technology:

- **If number of clients increases, it takes more time for sending response.**
- For each request, it starts a process and Web server is limited to start processes.
- It uses platform dependent language e.g. C, C++, perl.

Servlet


Web Server
Web Container
Request
Request
Request
Thread
Thread
Thread
Servlet
Processor Load

Advantage of Servlet

There are many advantages of servlet over CGI. 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:

Better performance: **because it creates a thread for each request not process.**
Portability: because it uses java language.
Robust: Servlets are managed by JVM so we don't need to worry about memory leak, garbage collection etc.
Secure: because it uses java language

javax.servlet package

CLASSES:

1.GenericServlet
2.ServletInputStream
3.ServletOutputStream
4.ServletRequestWrapper
5.ServletResponseWrapper
6.ServletRequestEvent
7.ServletContextEvent
8.ServletRequestAttributeEvent
9.ServletContextAttributeEvent
10.ServletException
11.UnavailableException

INTERFACES:

1.Servlet
2.ServletRequest
3.ServletResponse
4.RequestDispatcher
5.ServletConfig
6.ServletContext
7.SingleThreadModel
8.Filter
9.FilterConfig
10.FilterChain
11.ServletRequestListener
12.ServletRequestAttributeListener
13.ServletContextListener
14.ServletContextAttributeListener

javax.servlet.http package

CLASSES:

1. HttpServlet
2. Cookie
3. HttpServletRequestWrapper
4. HttpServletResponseWrapper
5. HttpSessionEvent
6. HttpSessionBindingEvent
7. HttpUtils (deprecated now)

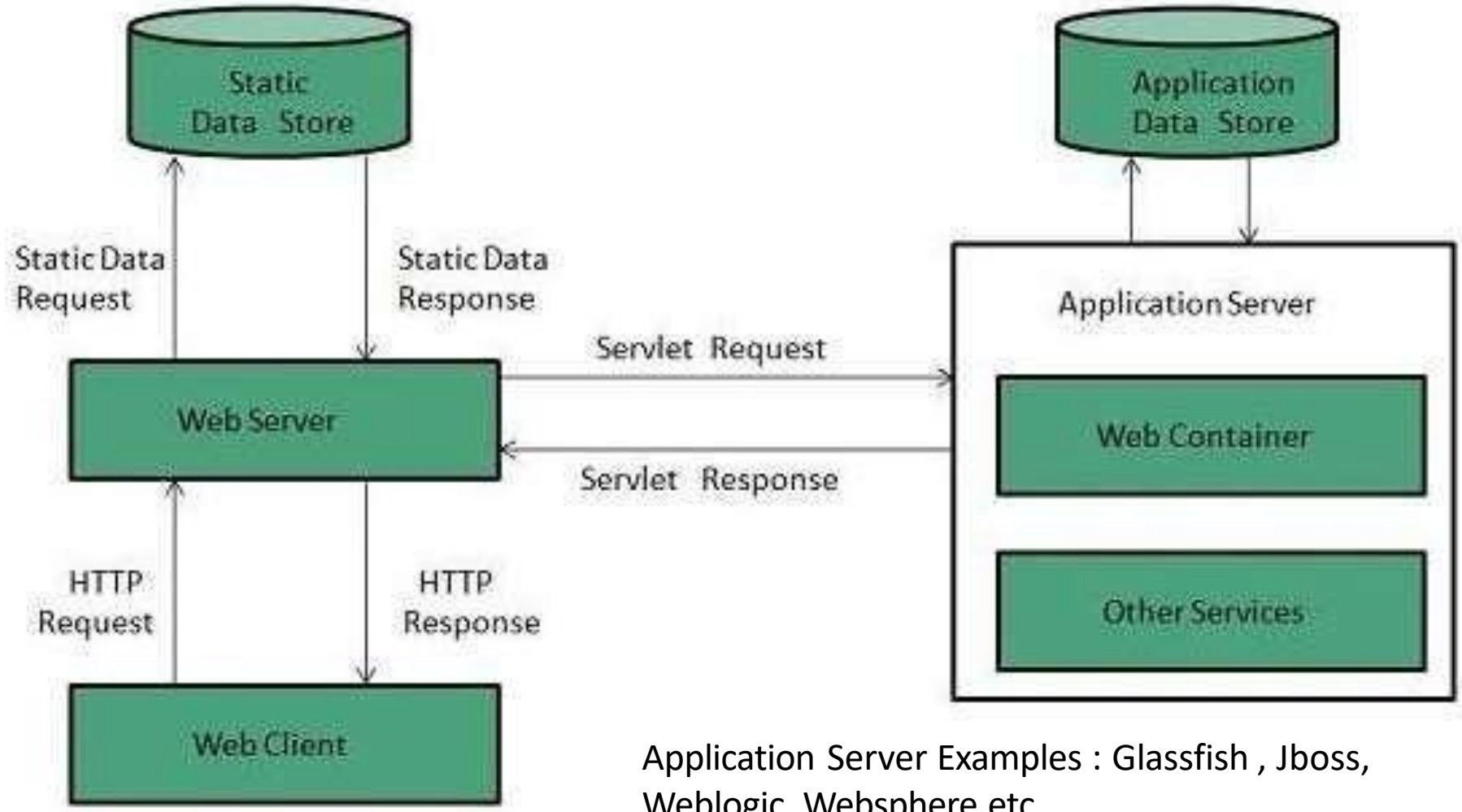
INTERFACES:

1. HttpServletRequest
2. HttpServletResponse
3. HttpSession
4. HttpSessionListener
5. HttpSessionAttributeListener
6. HttpSessionBindingListener
7. HttpSessionActivationListener
8. HttpSessionContext (deprecated now)

How does Servlet works ?

- When a client make a request for some servlet, he/she actually uses the **Web browser** in which request is written as a URL.
- The web browser then sends this request to Web server. The web server first finds the requested servlet.
- The obtained servlet gathers the relevant information in order to satisfy the client's request and builds a web page accordingly.
- This web page is then displayed to the client. Thus request made by client gets satisfied by the servlet.

How does Servlet works ?

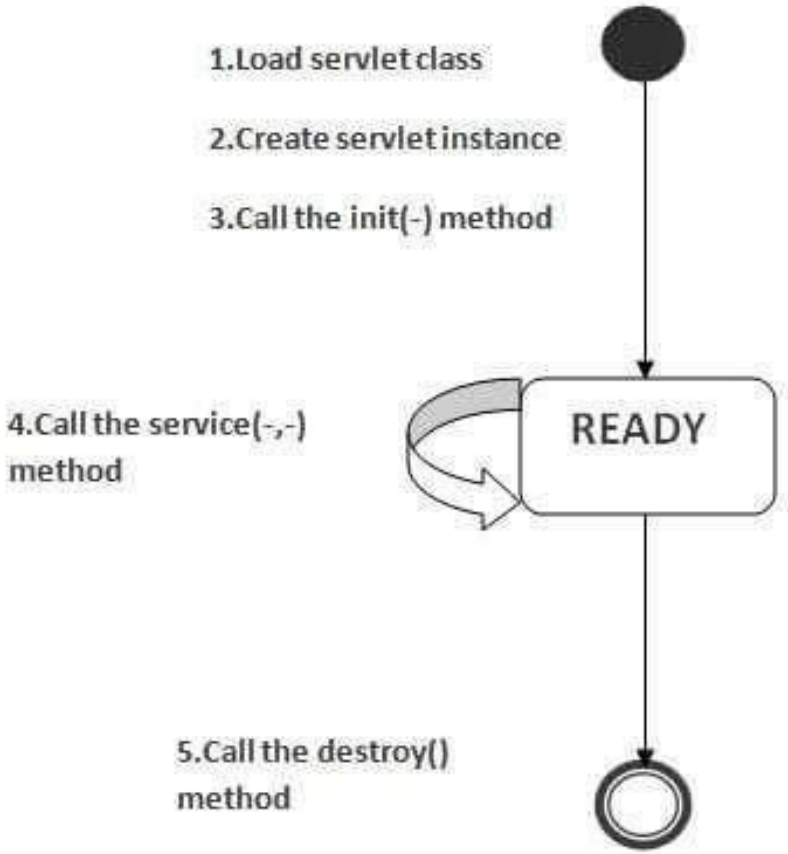


Application Server Examples : Glassfish , Jboss, Weblogic, Websphere etc

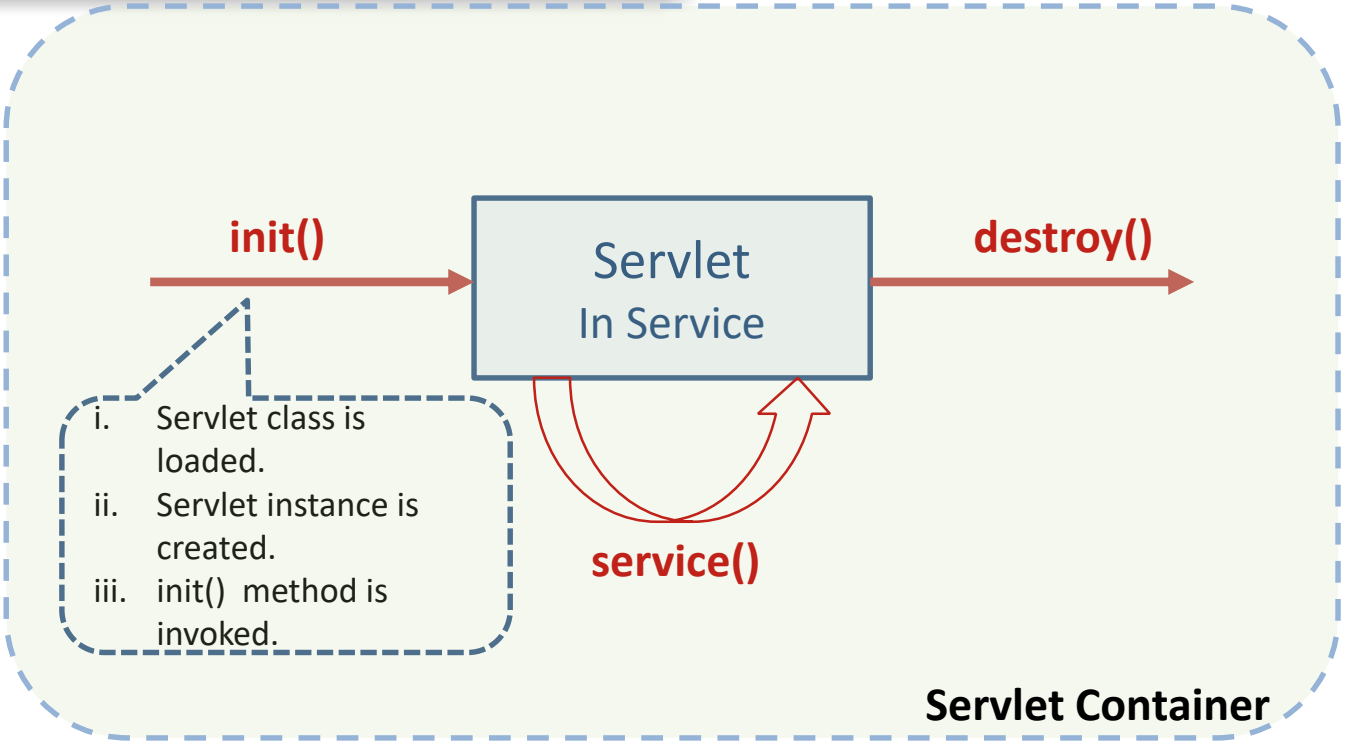
Servlet Life Cycle

The web container maintains the life cycle of a servlet instance.

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.



Servlet Life Cycle



Servlet Life Cycle

1. **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.

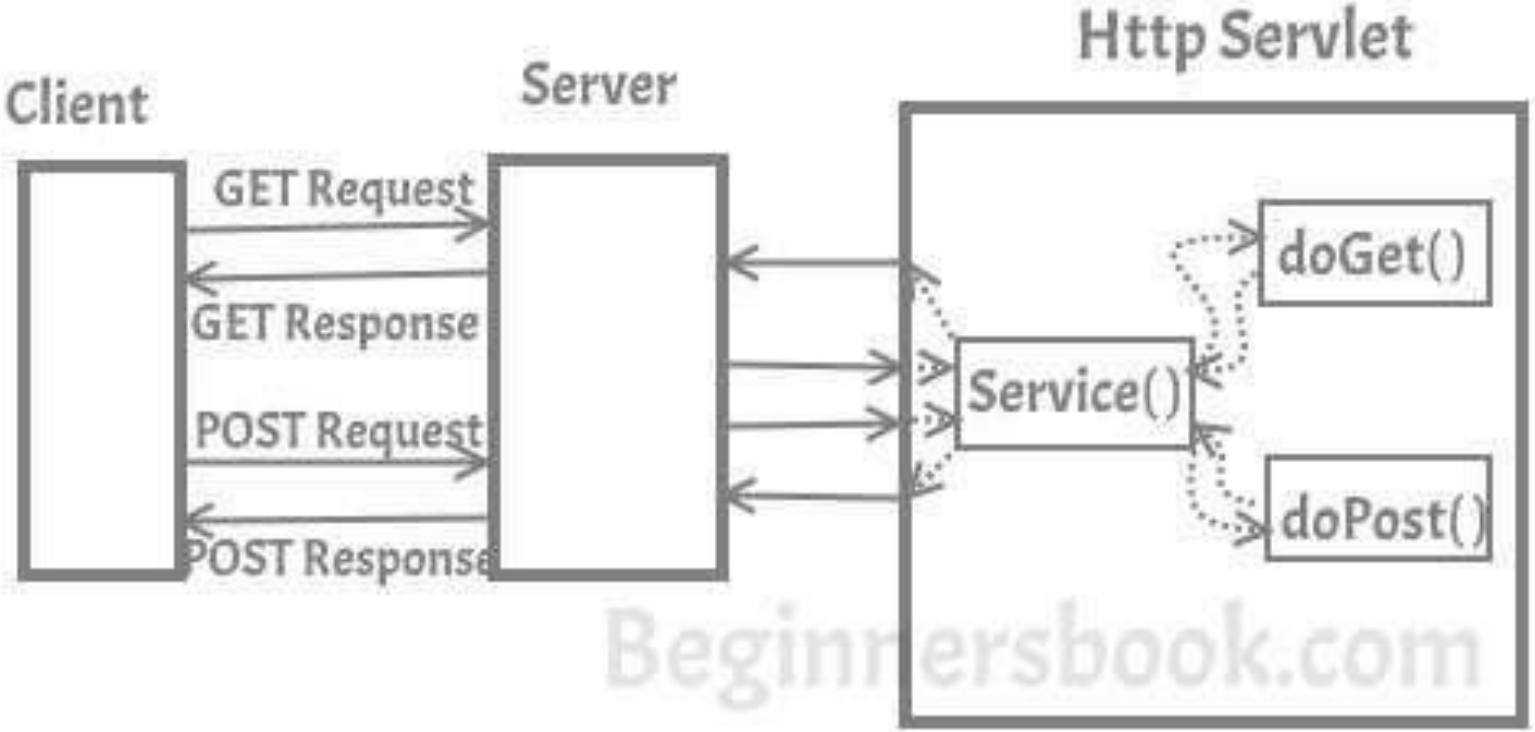
2. **Servlet instance is created:** The **web container creates the instance of a servlet** after loading the servlet class. The servlet instance is created only once in the servlet life cycle.

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.

4. **service method is invoked:** The web container calls the service method each time when request for the servlet is received.

5. **destroy method is invoked:** The web container calls the destroy method before removing the servlet instance from the service.

Servlet Life Cycle Example



Deployment Descriptor

Copying the **.class file** of the Servlet from the **current directory** to the **classes** folder of **Tomcat** (or any Web server) is known as **deployment**. When deployed, **Tomcat** is ready to load and execute the Servlet, at anytime, at the client request.

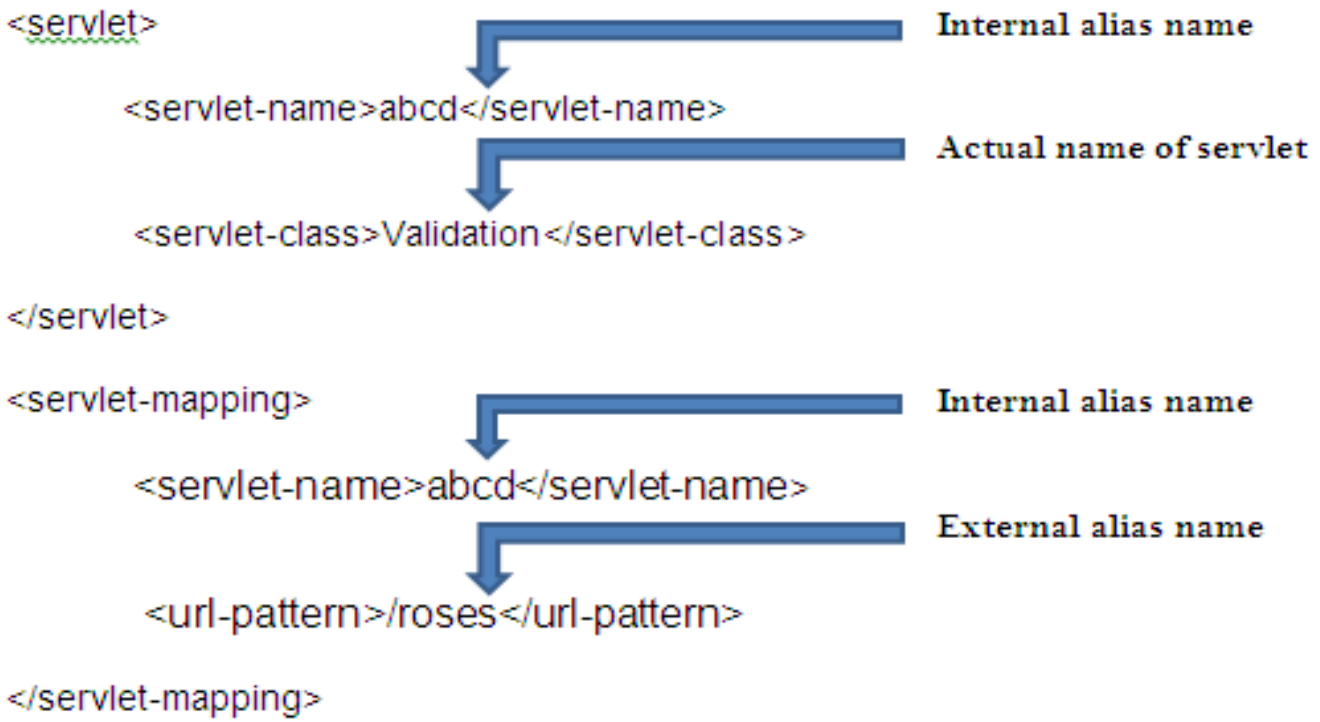
As the name indicates, the **deployment descriptor** describes the deployment information (or Web Information) of a Servlet. The deployment descriptor is an XML file known as **web.xml**. XML is the **easiest way to give the information to a server**, just writing in between the tags, instead of writing in a text file or RDBMS file. The name and tags of **web.xml** are Servlet API specifications.

Deployment Descriptor

The following activities can be done by the programmer in **web.xml** file.

a) Mapping alias name with the actual Servlet name

First and foremost is the alias name to the Servlet. Never a client is given the actual name of the Servlet. Always an alias name is given just for security (avoid hacking). The alias name is given in the following XML tags.



Deployment Descriptor

Marwadi education foundation

b) To write Initialization Parameters

Intialization parameteres are read by the Servlet from web.xml file. Programmer can write code to be used for initialization. An example code is given below

```
<init-param>
 <param-name>instructor</param-name>
 <param-value> Ravi</param-value>
</init-param>
```

Deployment Descriptor

c) To write tag libraries (this is mostly used in frameworks like Struts etc)
An example code is given used in Struts. This we will cover later.

```
<taglib>
  <taglib-uri>/tags/struts-bean</taglib-uri>
  <taglib-location>/WEB-INF/struts-bean.tld</taglib-location>
</taglib>
```

Following are a few important XML elements in web.xml.
<context-param>, <filter-mapping>, <taglib> and <mime-mapping> etc.

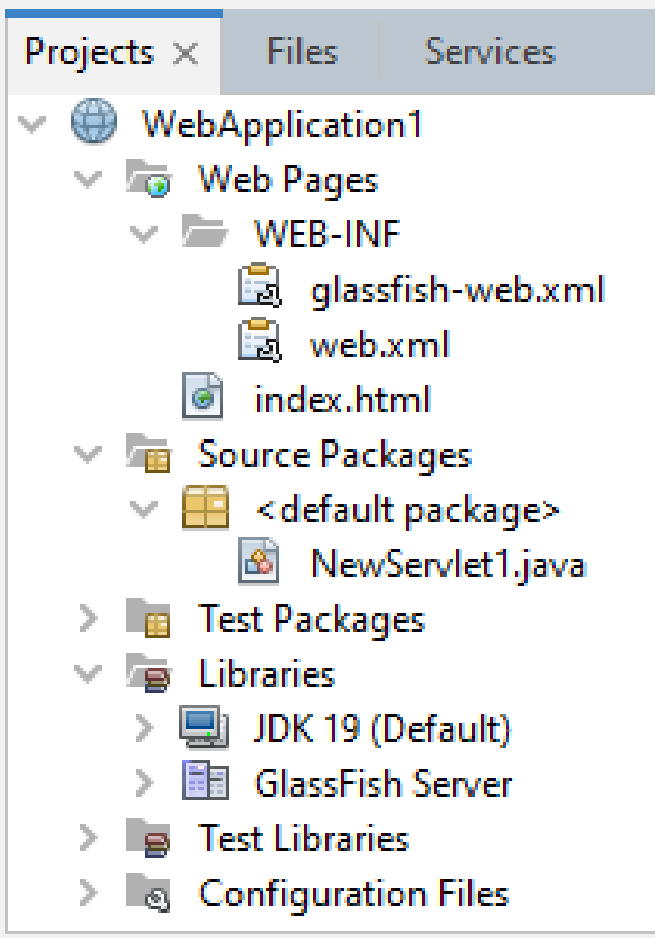
Note 1: When a Servlet code is modified and copied to classes folder, each time it is required to restart the Tomcat server.
Note 2: For every Servlet, there must be an entry in web.xml file with an alias name.

Deployment Descriptor

A **deployment descriptor (DD)** refers to a configuration file for an artifact that is deployed to some container/engine.

In the Java Platform, Enterprise Edition, a deployment descriptor describes how a component, module or application (such as a web application or enterprise application) should be deployed.

It directs a deployment tool to deploy a module or application with specific container options, security settings and describes specific configuration requirements.



Deployment Descriptor

XML is used for the syntax of these deployment descriptor files.

For **web applications**, the deployment descriptor must be called **web.xml** and must reside in the **WEB-INF** directory in the web application root.

For **Java EE applications**, the deployment descriptor must be named **application.xml** and must be placed directly in the **META-INF** directory at the top level of the application .ear file

In Java EE, there are **two types** of deployment descriptors:

1. Java EE deployment descriptors
2. runtime deployment descriptors

Deployment Descriptor

The Java EE deployment descriptors are defined by the language specification, whereas the runtime descriptors are defined by the vendor of each container implementation.

For example, **the web.xml** file is a **standard Java EE** deployment descriptor, specified in the Java Servlet specification, but the **sun-web.xml** file contains configuration data specific to the **Sun GlassFish Enterprise Server** implementation.

The **web.xml** file is located in the **WEB-INF** directory of your Web application. The first entry, under the root servlet element in web.xml, defines a **name** for the **servlet** and specifies the **compiled class** that executes the **servlet**.

Deployment Descriptor: web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE web-app
PUBLIC "-//Sun Microsystems, Inc.//DTD Web Application 2.3//EN"
"http://java.sun.com/dtd/web-app_2_3.dtd">

<web-app>
  <servlet>

  <servlet-name>MyServlet</servlet-name>
    <servlet-class>MyServlet</servlet-class>

    <init-param>
        <param-name>name</param-name>
        <param-value>cncy</param-value>
    </init-param>
    </servlet>

<servlet-mapping>
  <servlet-name>MyServlet</servlet-name>
  <url-pattern>/MyServlet</url-pattern>
</servlet-mapping>

</web-app>
```

xml header

Document Type Definition

Configures a web application.

Name used to access Java Servlet

Name of servlet .java class

Used to pass parameters to a servlet from the web.xml file.

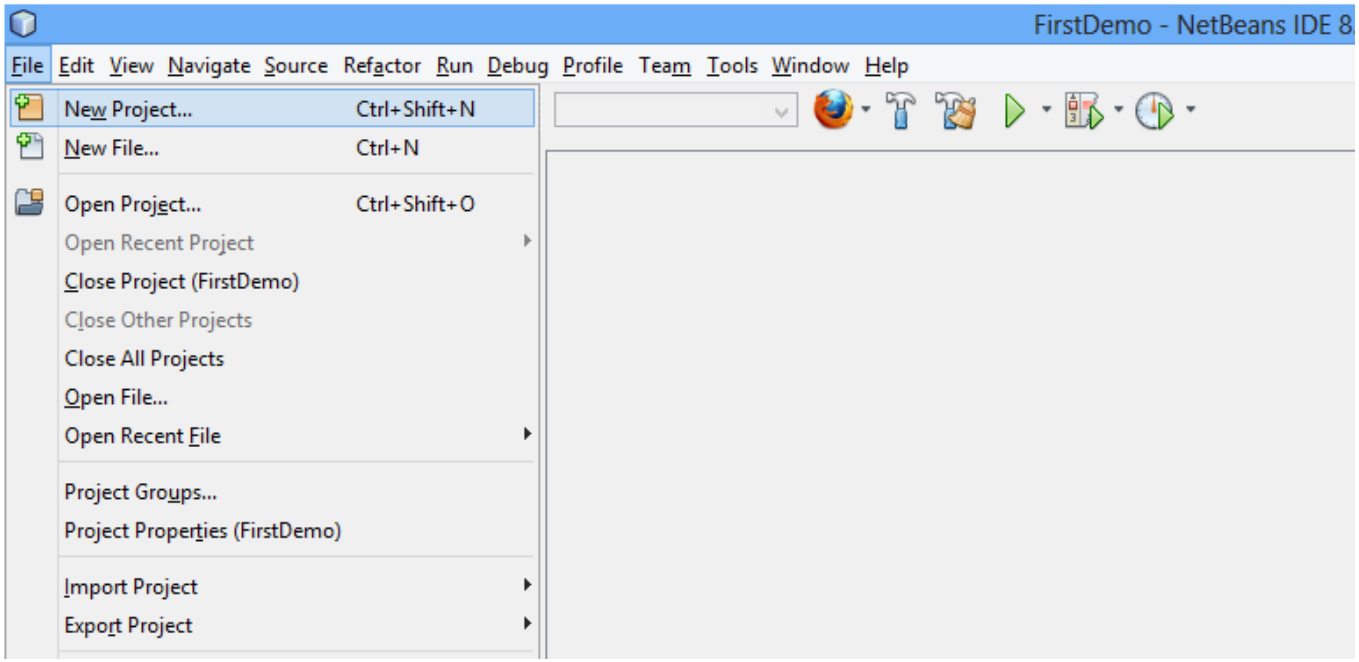
map the servlet to a URL or URL pattern

Controls behavior of Servlet

Steps to run Servlet Program in Netbeans

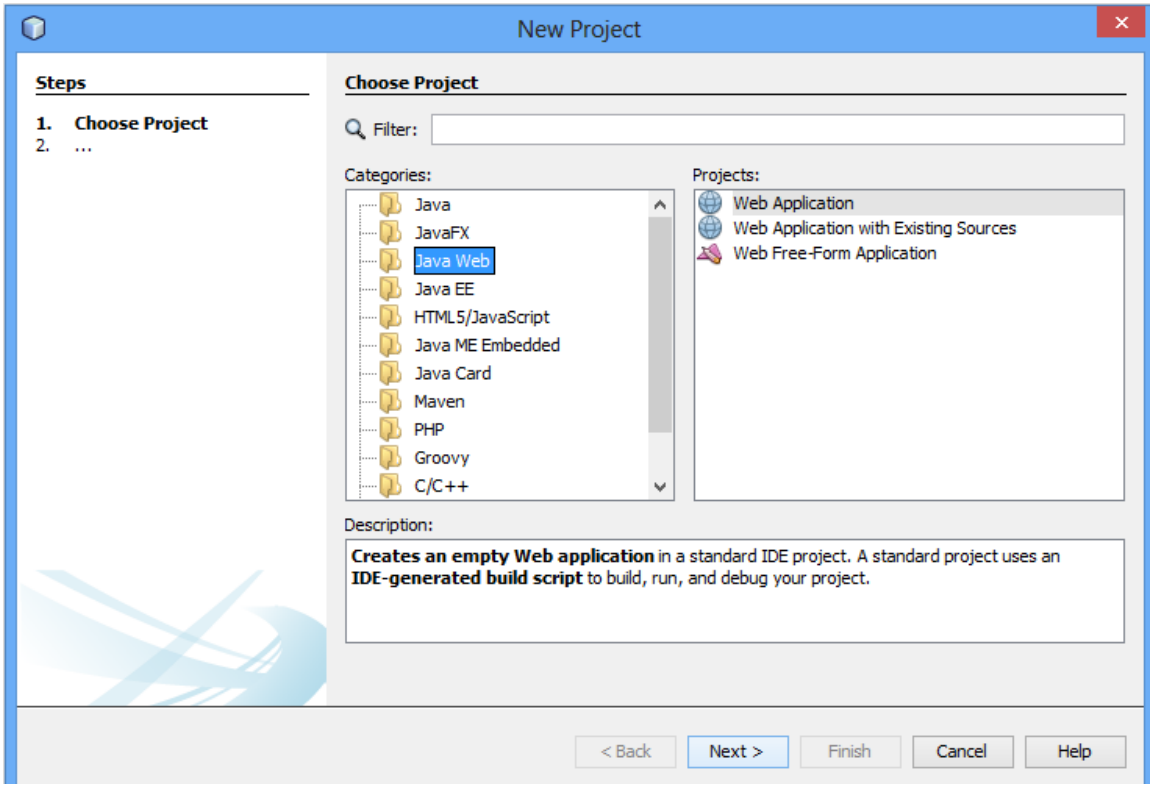
Marwadi education foundation

Step 1: Open Netbeans IDE, Select **File** -> **New Project**



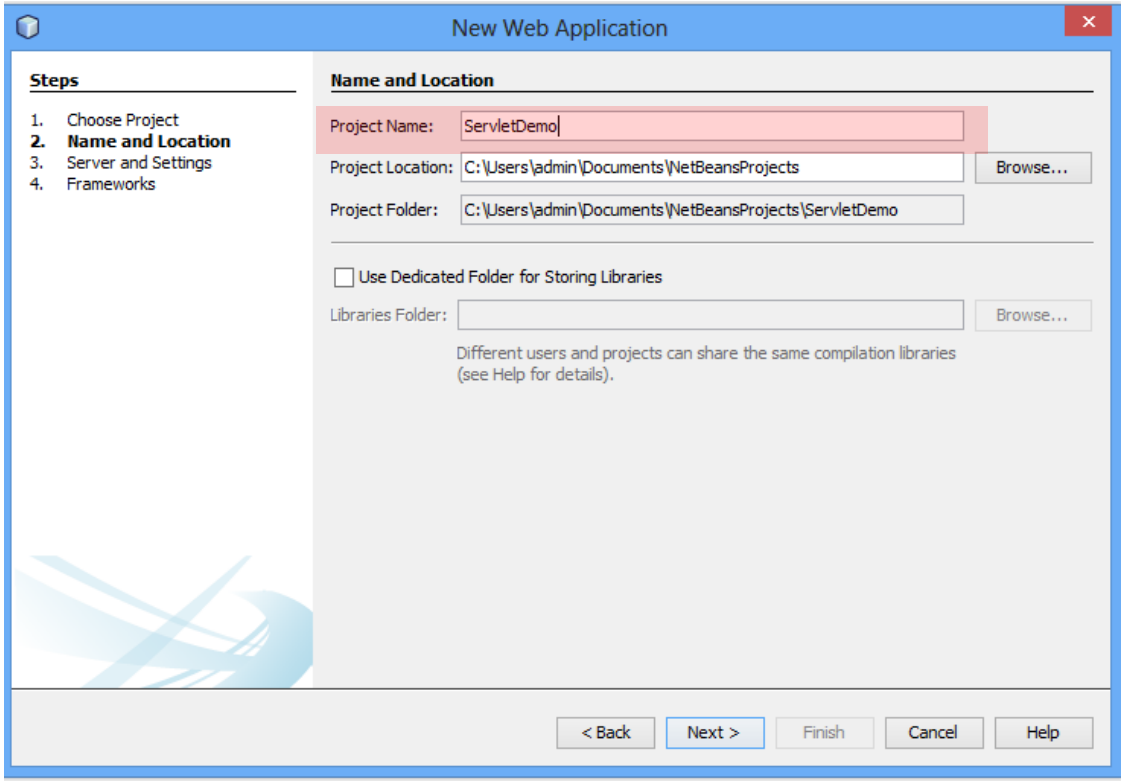
Steps to run Servlet Program in Netbeans

Step 2: Select **Java Web** -> **Web Application**, then click on Next



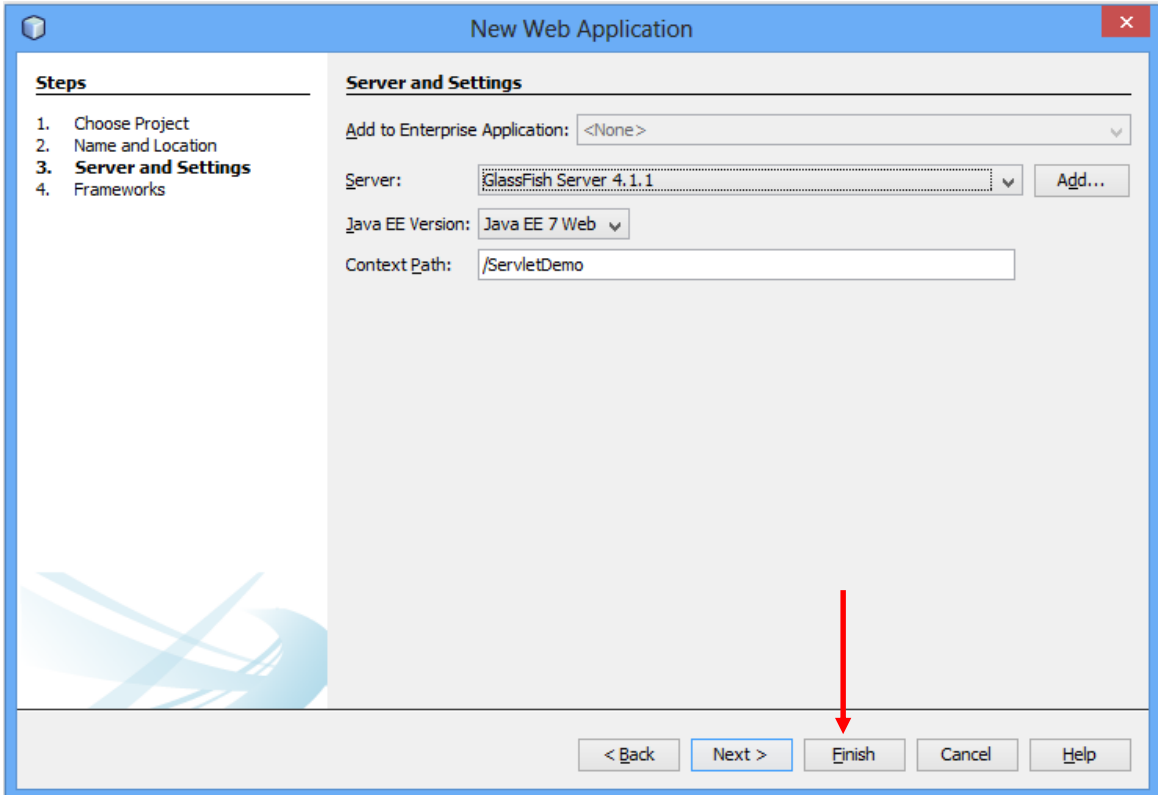
Steps to run Servlet Program in Netbeans

Step 3: Give a name to your project and click on Next,



Steps to run Servlet Program in Netbeans

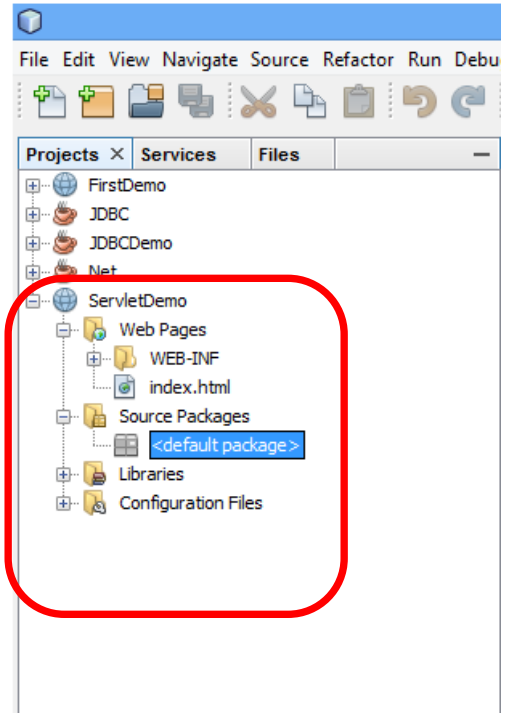
Step 4: and then, Click **Finish**



Steps to run Servlet Program in Netbeans

Marwadi education foundation

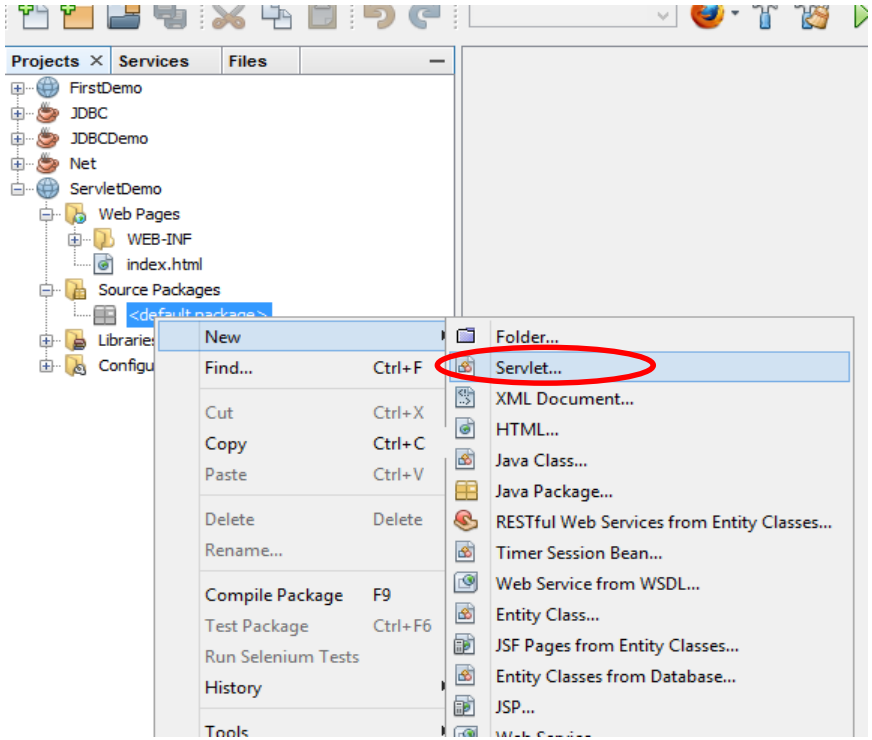
Step 5: The complete directory structure required for the Servlet Application will be created automatically by the IDE.



Steps to run Servlet Program in Netbeans

Marwadi education foundation

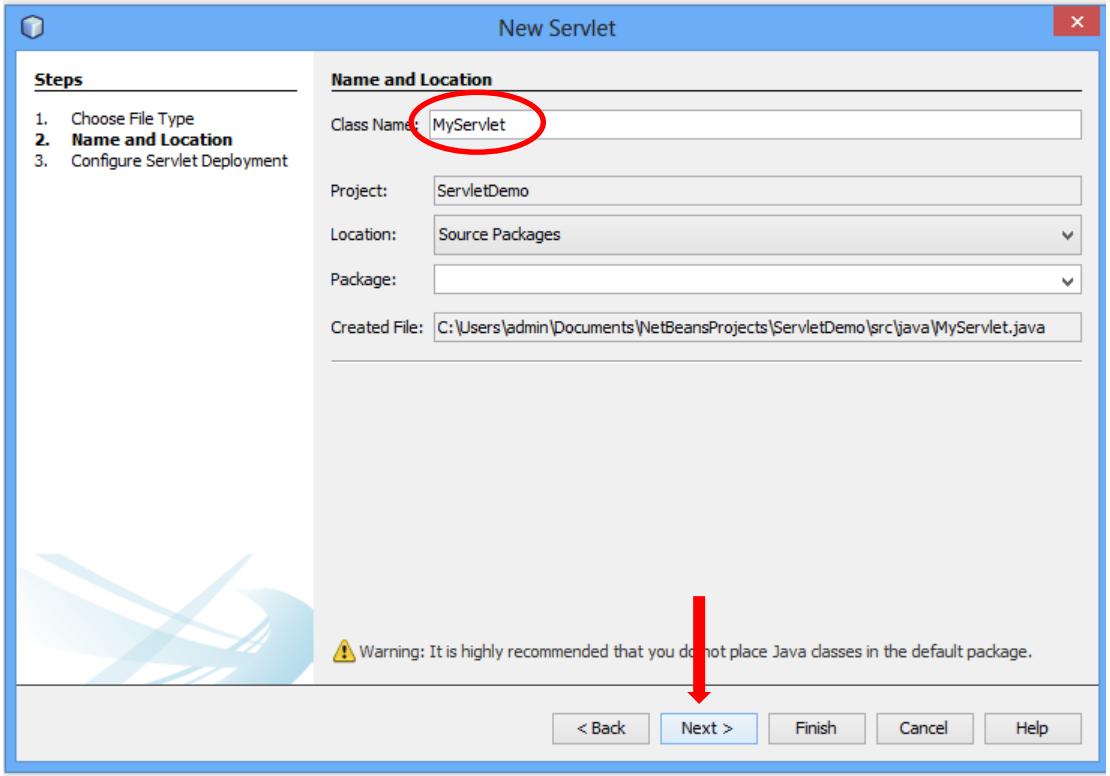
Step 6: To create a Servlet, open Source Package, right click on default packages -> New -> Servlet.



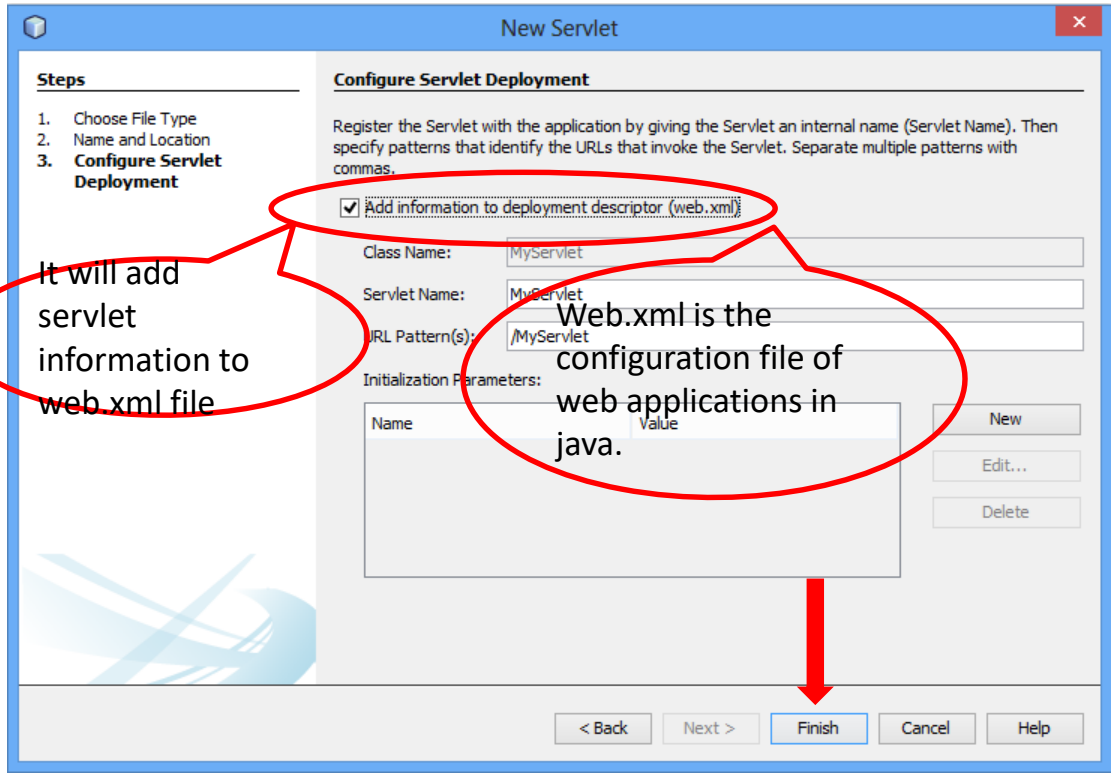
Steps to run Servlet Program in Netbeans

Marwadi education foundation

Step 7: Give a Name to your Servlet class file



Steps to run Servlet Program in Netbeans



Steps to run Servlet Program in Netbeans

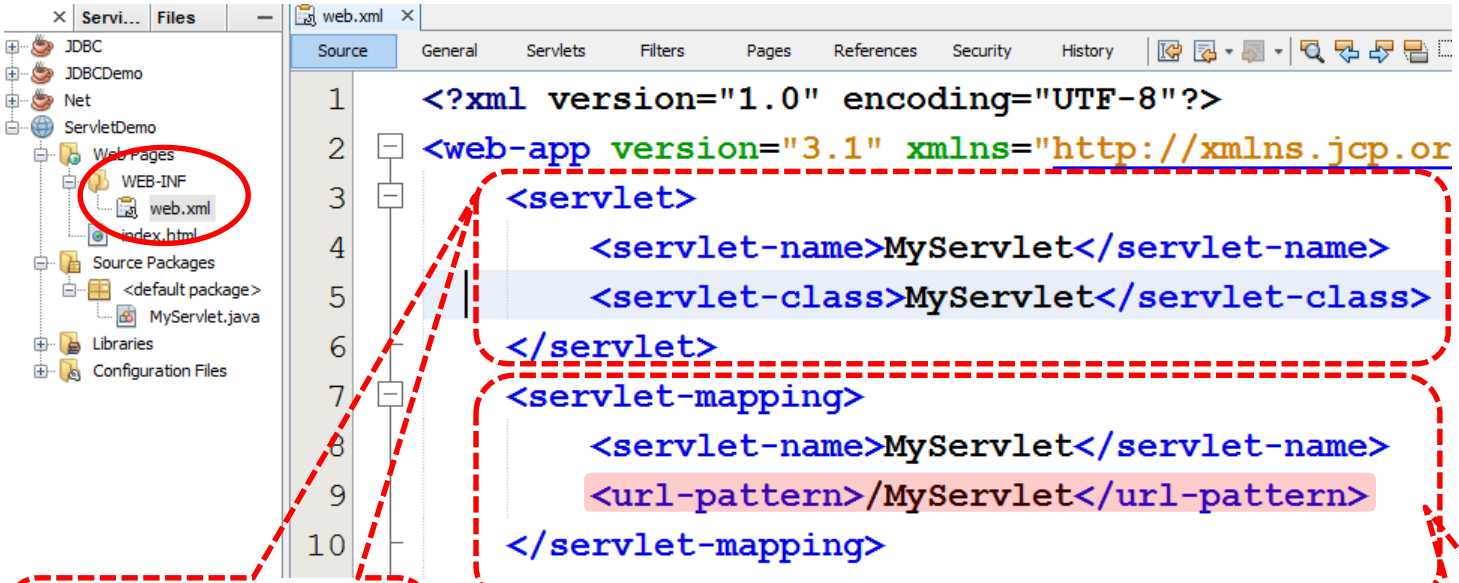
MyServet1.java

```
1  import java.io.*;
2  import javax.servlet.*;
3  import javax.servlet.http.*;
4  public class MyServlet1 extends HttpServlet
5  {     String msg="";
6      PrintWriter out;
7      public void init() throws ServletException
8      {          msg="hello world: my first servlet program";     }
9      public void doGet(HttpServletRequest request, HttpServletResponse response)throws ServletException, IOException
10     {
11         response.setContentType("text/html");
12         out=response.getWriter();
13         out.println(msg);
14     }
15     public void destroy()
16     {         out.close();
17     }
18 }
19
20
21
22
23
24
```

Steps to run Servlet Program in Netbeans

Marwadi education foundation

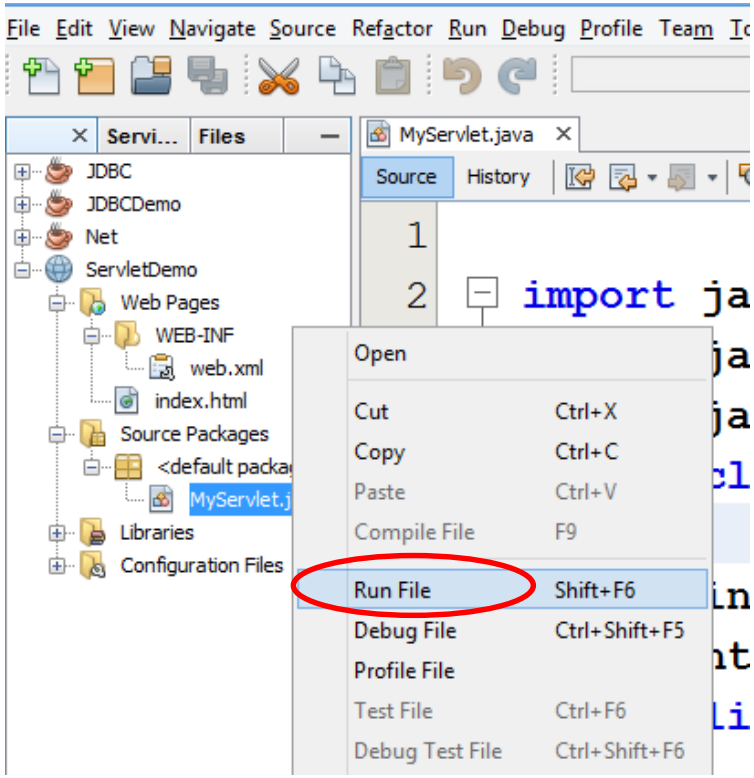
Step 9: open web.xml

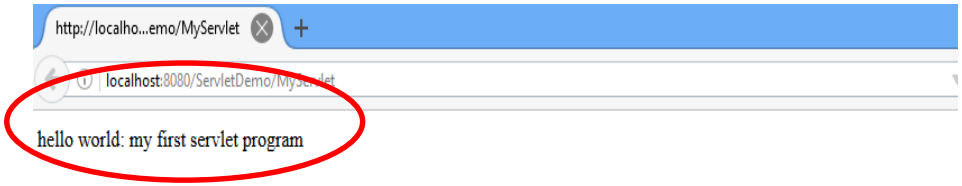


Steps to run Servlet Program in Netbeans

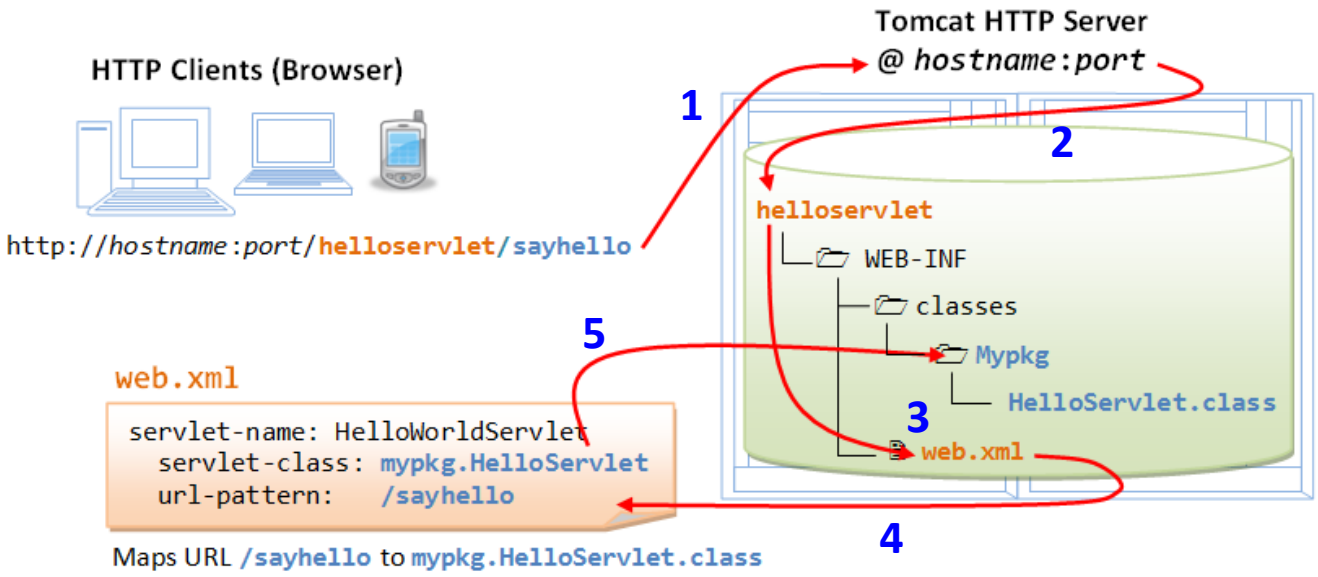
Marwadi education foundation

Step 11: Run your application, right click on your Project and select **Run**





Java Servlet



Ref: https://www.ntu.edu.sg/home/ehchua/programming/java/JavaServlets.html

Examples

PROG1 Servlet Life Cycle (Servlet Interface)

PROG2 Length of Entered String (HTTP Servlet using doGet( ))

PROG3 Area of Circle (HTTP Servlet using doPost( ))

PROG4 Add two numbers (HTTPServlet)

Types of Servlets

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.**

Servlets Interface

Servlet interface provides common behaviorur to all the servlets. Servlet interface defines methods that all servlets must implement.

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.**

Servlets Interface

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.

Check Program : ServletExample

Generic Servlets Class

GenericServlet class 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.**

Methods of GenericServlet Class

There are many methods in GenericServlet class. They are as follows:

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 void init() it is a convenient method for the servlet programmers, now there is no need to call super.init(config)

Methods of GenericServlet Class

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.

Check Program : genericexample

HTTPServlet Class

The HttpServlet class extends the GenericServlet class and implements Serializable interface. It provides http specific methods such as doGet, doPost, doHead, doTrace etc.

Methods of HttpServlet class

There are many methods in HttpServlet class. They are as follows:

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.

protected void doGet(HttpServletRequest req, HttpServletResponse res) handles the GET request. It is invoked by the web container.

protected void doPost(HttpServletRequest req, HttpServletResponse res) handles the POST request. It is invoked by the web container.

HTTPServlet Class

protected void doHead(HttpServletRequest req, HttpServletResponse res) handles the HEAD request. It is invoked by the web container.

protected void doOptions(HttpServletRequest req, HttpServletResponse res) handles the OPTIONS request. It is invoked by the web container.

protected void doPut(HttpServletRequest req, HttpServletResponse res) handles the PUT request. It is invoked by the web container.

protected void doTrace(HttpServletRequest req, HttpServletResponse res) handles the TRACE request. It is invoked by the web container.

protected void doDelete(HttpServletRequest req, HttpServletResponse res) handles the DELETE request. It is invoked by the web container.

protected long getLastModified(HttpServletRequest req) returns the time when HttpServletRequest was last modified since midnight January 1, 1970 GMT.

doGet() v/s doPost()

doGet()

A GET request results from request for a URL or from an HTML form, should be handled by doGet() method.

Syntax

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

doPost()

A POST request results from an HTML form that specifically lists POST as the METHOD and it should be handled by doPost() method.

Syntax

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

Get Vs Post

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.
6) The length restriction of GET method is limited (2,048 characters). GET requests can be viewed in browser history.	There is no restriction in sending the length of data.

How to create the Servlet?

1. Servlet (Interface)
    a) Init () : void
    b) Service(ServletRequest, ServletResponse) : void
    c) Destroy() : void
    d) getServletInfo() : String
    e) getServletConfig() : String

2. GenericServlet (Abstract Class)
    a) Service(ServletRequest, ServletResponse) : void

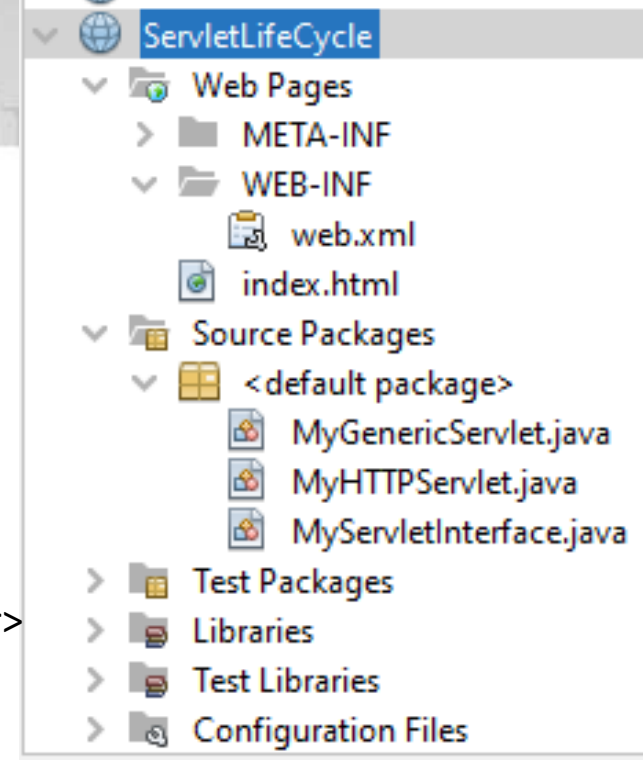
3. HttpServlet (Abstract Class)
    a) doGet(HttpServletRequest,HttpServletResponse)
    b) doPost (HttpServletRequest,HttpServletResponse)

Servlets Example

web.xml

```
<web-app>
 <servlet>
    <servlet-name>MyServletInterface</servlet-name>
    <servlet-class>MyServletInterface</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>MyHTTPServlet</servlet-name>
    <servlet-class>MyHTTPServlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>MyGenericServlet</servlet-name>
    <servlet-class>MyGenericServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>MyServletInterface</servlet-name>
    <url-pattern>/MyServletInterface</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>MyHTTPServlet</servlet-name>
    <url-pattern>/MyHTTPServlet</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>MyGenericServlet</servlet-name>
    <url-pattern>/MyGenericServlet</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
```

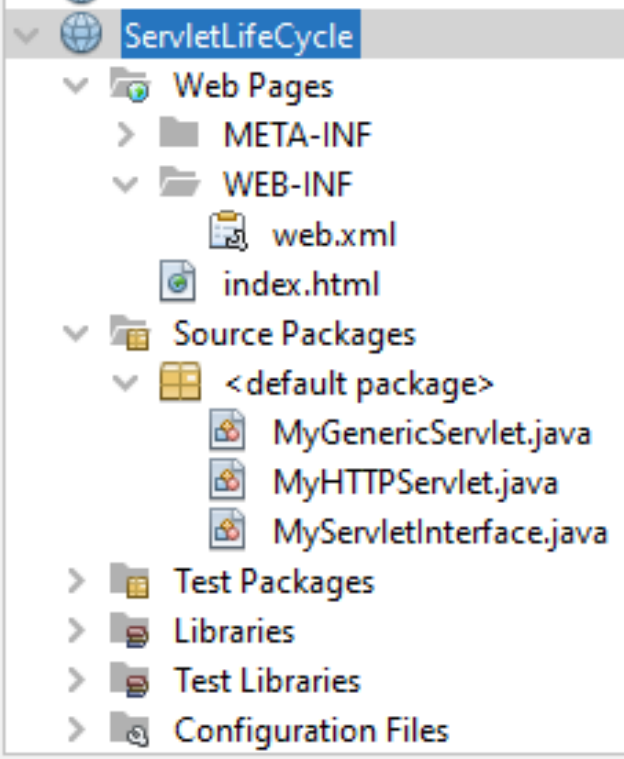
Servlets Example



Index.html

```
<body>
      <a href="MyServletInterface">Take me to Servlet Interface </a><br>
      <a href="MyGenericServlet">Take me to Generic Servlet</a><br>
       <a href="MyHTTPServlet">Take me to HTTP Servlet</a>

   </body>
```

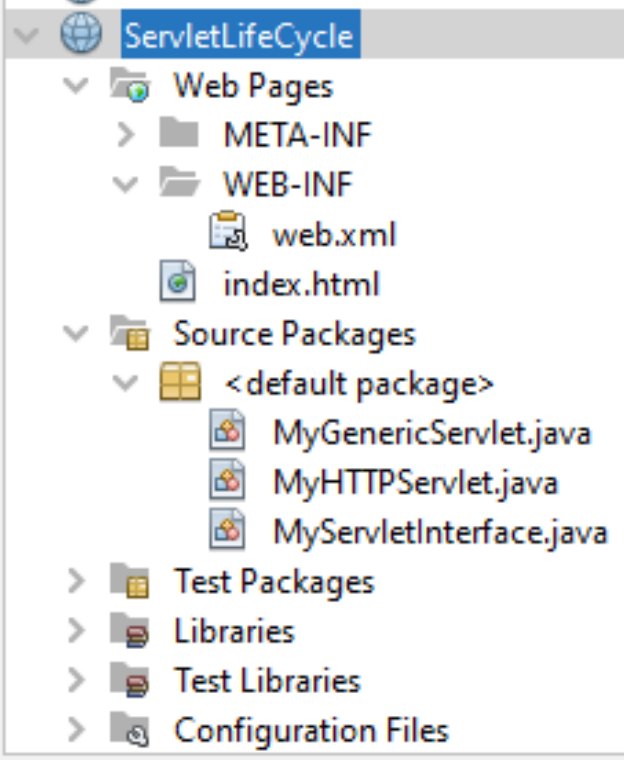
Servlets Example



MyHTTPServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class MyHTTPServlet extends HttpServlet {
  @Override
   protected void doGet(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      try (PrintWriter out = response.getWriter()) {
         out.println("<!DOCTYPE html>");
         out.println("<html>");
         out.println("<head>");
         out.println("<title>Servlet MyHTTPServlet</title>");
         out.println("</head>");
         out.println("<body>");
         out.println("<h1>THis is HTTP Servlet</h1>");
         out.println("</body>");          out.println("</html>");        }    } }
```

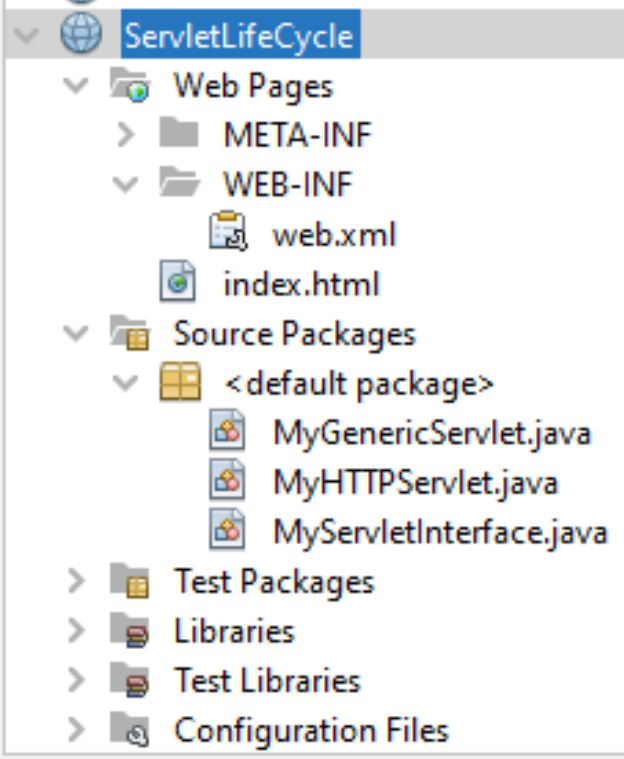
Servlets Example



MyServletInterface.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.Servlet;
import javax.servlet.ServletConfig;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;

public class MyServletInterface 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>This is service method of implemented from Servlet Interface</b>");
out.print("</body></html>");   }
```

Servlets Example

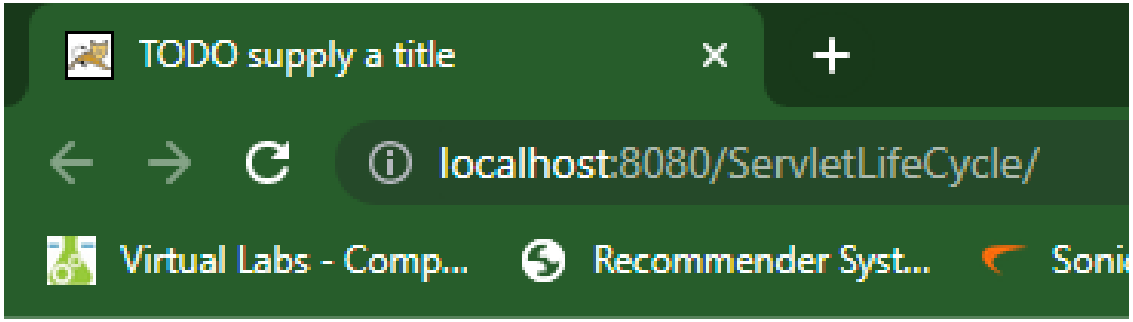


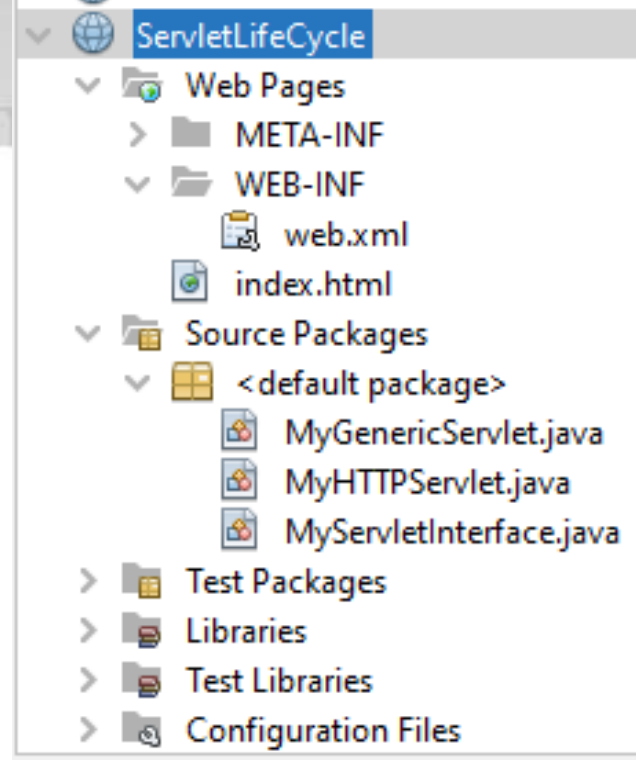
MyServletInterface.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.GenericServlet;
import javax.servlet.ServletException;
import javax.servlet.ServletRequest;
import javax.servlet.ServletResponse;
public class MyGenericServlet extends GenericServlet {

@Override
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 from service method of generic servlet</b>");
out.print("</body></html>");

}
}
```

Servlets Example

Output





ServletConfig Interface

An object of ServletConfig is created by the web container for each servlet. This object can be used **to get configuration information** from **web.xml** file.

If the configuration information is modified from the web.xml file, we don't need to change the servlet. So it is easier to manage the web application if any specific content is modified from time to time.

Advantage of ServletConfig

The core advantage of ServletConfig is that you don't need to edit the servlet file if information is modified from the web.xml file.

ServletConfig Interface

Methods of ServletConfig interface

public String getInitParameter(String name): Returns the parameter value for the specified parameter name.

public Enumeration getInitParameterNames(): Returns an enumeration of all the initialization parameter names.

public String getServletName(): Returns the name of the servlet.

public ServletContext getServletContext(): Returns an object of ServletContext.

ServletConfig Interface

How to get the object of ServletConfig

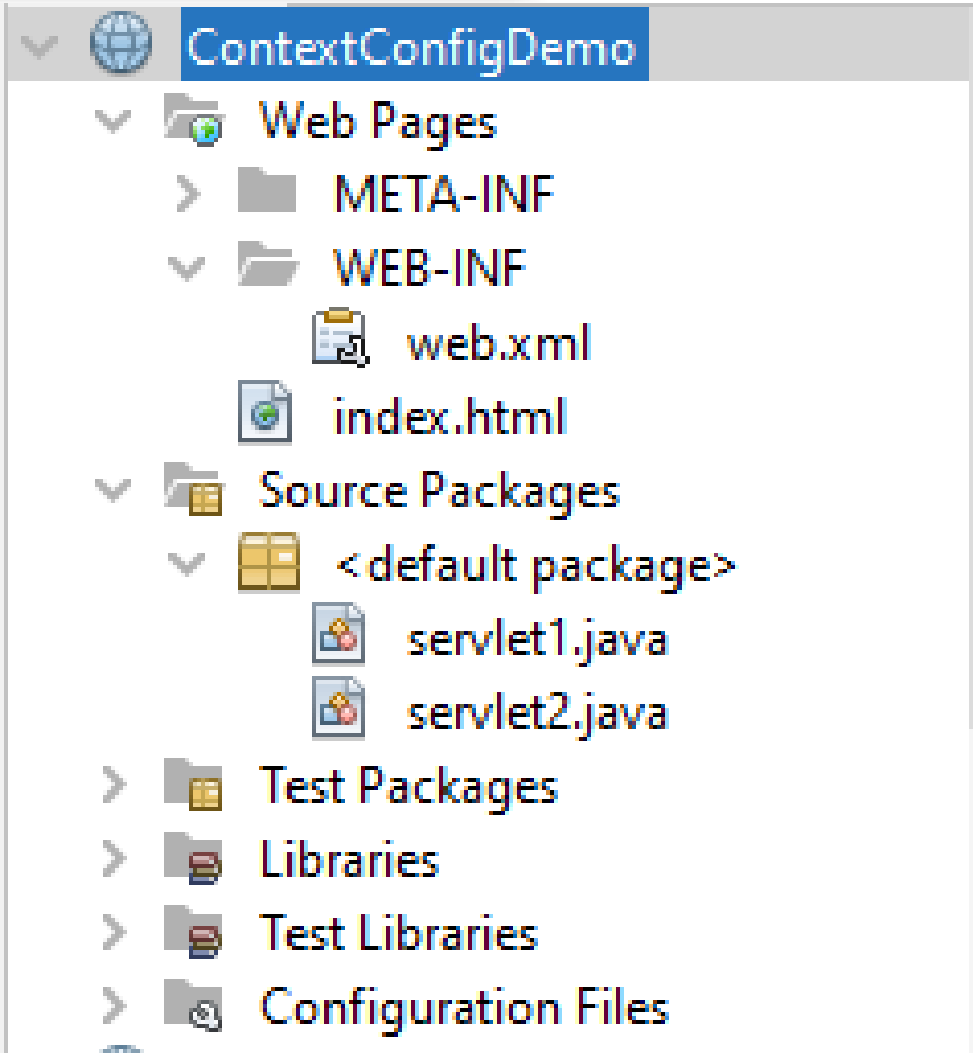
getServletConfig() method of Servlet interface returns the object of ServletConfig.

Syntax of getServletConfig() method

```
public ServletConfig getServletConfig();
```

Check Examples → config4,config5

ServletConfig Example



ServletConfig & ServletContext Example

Marwadi education foundation

```
<servlet>
    <servlet-name>servlet1</servlet-name>
    <servlet-class>servlet1</servlet-class>
<init-param>
  <param-name>programmer</param-name>
  <param-value>Asif</param-value>
</init-param>
 </servlet>


 <servlet>
    <servlet-name>servlet2</servlet-name>
    <servlet-class>servlet2</servlet-class>
    <init-param>
      <param-name>programmer</param-name>
      <param-value>Ravi</param-value>
    </init-param>
 </servlet>
```

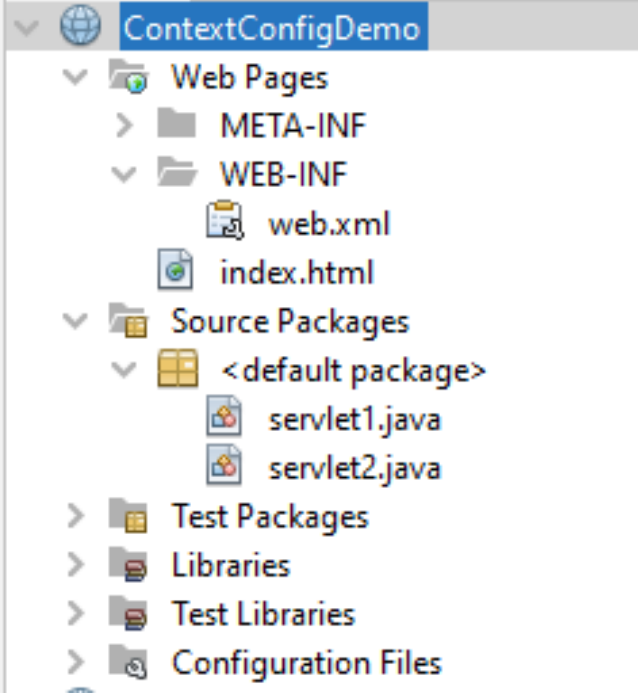
```
<servlet-mapping>
  <servlet-name>servlet1</servlet-name>
  <url-pattern>/servlet1</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>servlet2</servlet-name>
  <url-pattern>/servlet2</url-pattern>
</servlet-mapping>

<session-config>
  <session-timeout>
    30
  </session-timeout>
</session-config>
<context-param>
  <param-name>organisation</param-name>
  <param-value>Marwadi Info Tech</param-value>
</context-param>
```

ServletConfig & ServletContext Example

Index.html

```
<html>
   <head>
      <title>TODO supply a title</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width, initial-scale=1.0">
   </head>
   <body>
      <a href="servlet1"> Click for Servlet1 </a><br>
      <a href="servlet2"> Click for Servlet2 </a>
   </body>
</html>
```

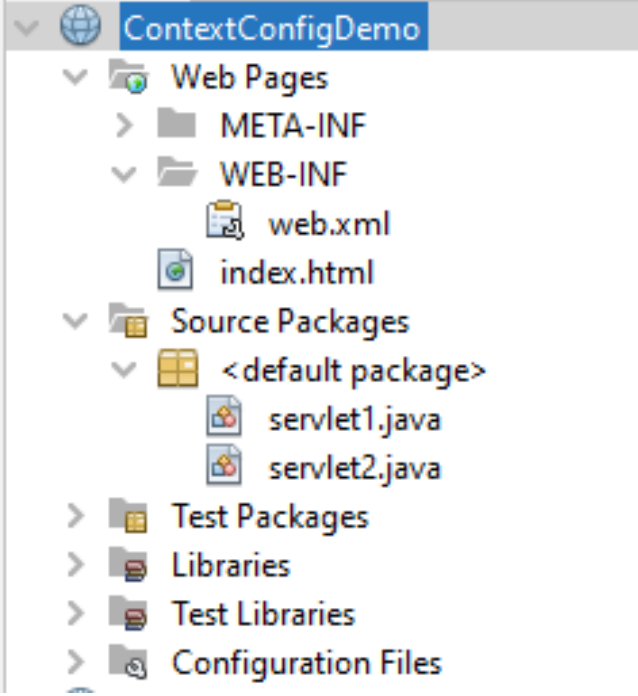
ServletConfig & ServletContext Example



Servlet1.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class servlet1 extends HttpServlet {
protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {

        ServletConfig config = getServletConfig();
        String developer = config.getInitParameter("programmer");
        out.println("This page is developed by " + developer);

        ServletContext context = getServletContext();
        String developed_At = context.getInitParameter("organisation");
        out.println("This page is developed " + developed_At);

} }
```

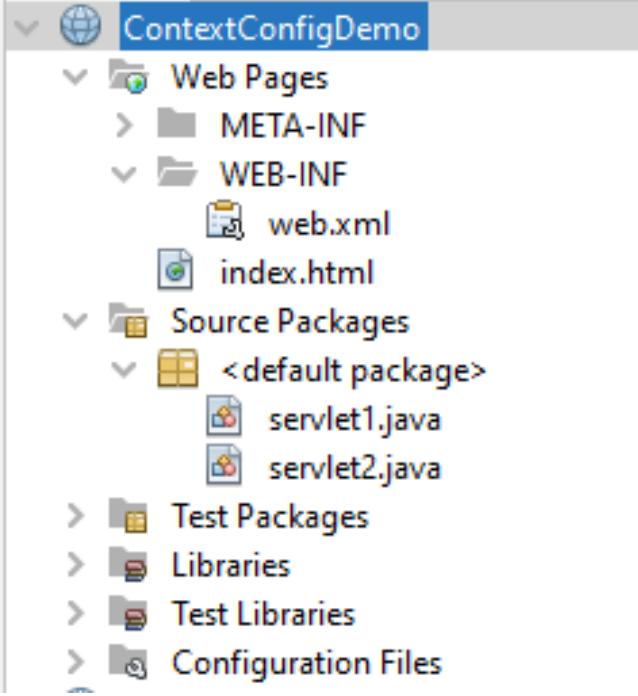
ServletConfig & ServletContext Example



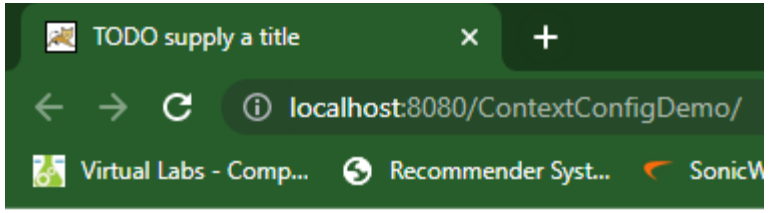
Servlet2.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletConfig;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class servlet2 extends HttpServlet {

protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {

        ServletConfig config = getServletConfig();
        String developer = (String)config.getInitParameter("programmer");
        out.println("This page is developed by " + developer);


        ServletContext context = getServletContext();
        String developed_At = (String)context.getInitParameter("organisation");
        out.println("This page is developed at " + developed_At);
}   }
```

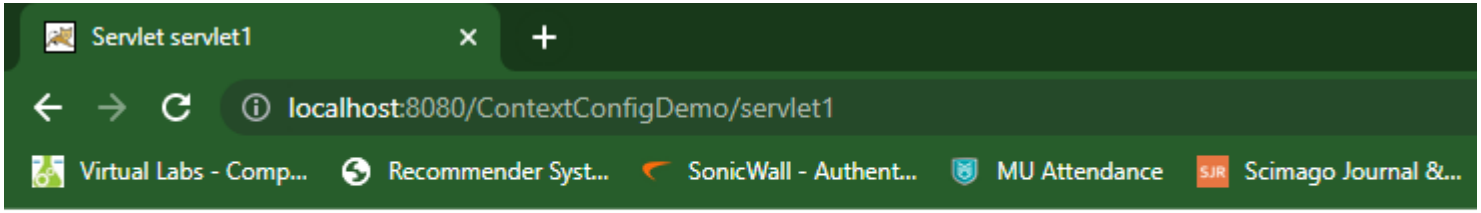
ServletConfig & ServletContext Example



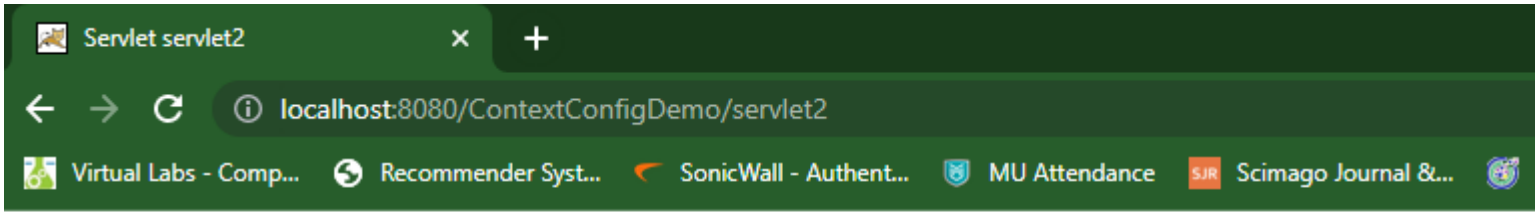
Output



Click for Servlet1
Click for Servlet2



This page is developed by Asif This page is developed Marwadi Info Tech



This page is developed by Ravi This page is developed at Marwadi Info Tech

Servlet Context Interface

An **object** of ServletContext is **created by the web container** at time of deploying the project. This object can be used to get configuration information **from web.xml** file. There is **only one** ServletContext **object** per web application.

If any information is shared to many servlet, it is better to provide it from the web.xml file using the **<context-param>** element.

Advantage of ServletContext

Easy to maintain if any information is shared to all the servlet, it is better to make it available for all the servlet. We provide this information from the web.xml file, so if the information is changed, we don't need to modify the servlet. Thus it removes maintenance problem.

Usage of ServletContext Interface

There can be a lot of usage of ServletContext object. Some of them are as follows:

1. The object of ServletContext provides an interface between the container and servlet.
2. The ServletContext object can be used to get configuration information from the web.xml file.
3. The ServletContext object can be used to set, get or remove attribute from the web.xml file.
4. The ServletContext object can be used to provide inter-application communication.

Commonly used methods of ServletContext interface

There is given some commonly used methods of ServletContext interface.

public String getInitParameter(String name): Returns the parameter value for the specified parameter name.

public void setAttribute(String name,Object object):sets the given object in the application scope.

public Object getAttribute(String name): Returns the attribute for the specified name.

public Enumeration getInitParameterNames(): Returns the names of the context's initialization parameters as an Enumeration of String objects.

public void removeAttribute(String name): Removes the attribute with the given name from the servlet context.

How to get the object of ServletContext interface

1. **getServletContext() method** of ServletConfig interface returns the object of ServletContext.

2. **getServletContext() method** of GenericServlet class returns the object of ServletContext.

Syntax of getServletContext() method:-

public ServletContext getServletContext()

Difference

ServletConfig	ServletContext
ServletConfig object is one per servlet class.	**ServletContext object is global to the entire web application.**
Object of ServletConfig will be created during the initialization process of the servlet.	Object of ServletContext will be created at the time of web application deployment
We have to give the request explicitly in order to create the ServletConfig object for the first time	ServletContext object can be available even before giving the first request
Scope: As long as a servlet is executing, the ServletConfig object will be available, it will be destroyed once the servlet execution is completed	Scope: As long as a web application is executing, the ServletContext object will be available, and it will be destroyed once the application is removed from the server
ServletConfig object is used while only one servlet requires information shared by it.	ServletContext object is used while application requires information shared by it
getServletConfig() method is used to obtain Servletconfig object	**getServletContext() method is used to obtain ServletContext object**
In web.xml — <init-param> tag will be appear under <servlet-class> tag.	**In web.xml — <context-param> tag will be appear under <web-app> tag.**

Key Point

The servletconfig object refers to the single servlet whereas servletcontext object refers to the whole web application.

Attributes in Servlet

Attribute is like a map object that can be used to set or get information from many servlets.

An **attribute in servlet** is an object that can be set, get or removed from one of the following scopes:

1. Request scope
2. Session scope
3. Application scope

The servlet programmer can pass information from one servlet to another using attributes. It is just like passing object from one class to another so that we can reuse the same object again and again.

Attributes Specific Methods in Servlets

There are 4 Attribute specific methods of ServletRequest, HttpSession and ServletContext interface

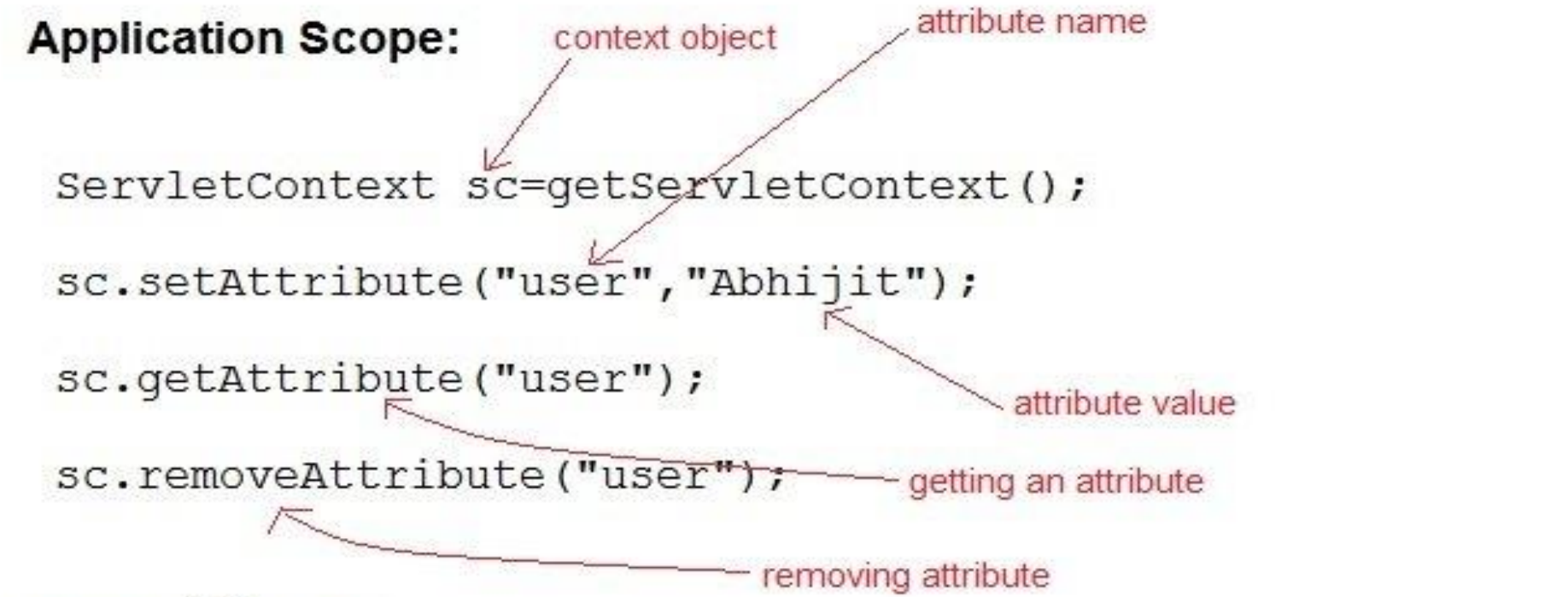
1.public void setAttribute(String name,Object object):sets the given object in the application scope.

2.public Object getAttribute(String name): Returns the attribute for the specified name.

3.public Enumeration getInitParameterNames(): Returns the names of the context's initialization parameters as an Enumeration of String objects.

4.public void removeAttribute(String name): Removes the attribute with the given name from the servlet context.

Application Scope:



request Scope:

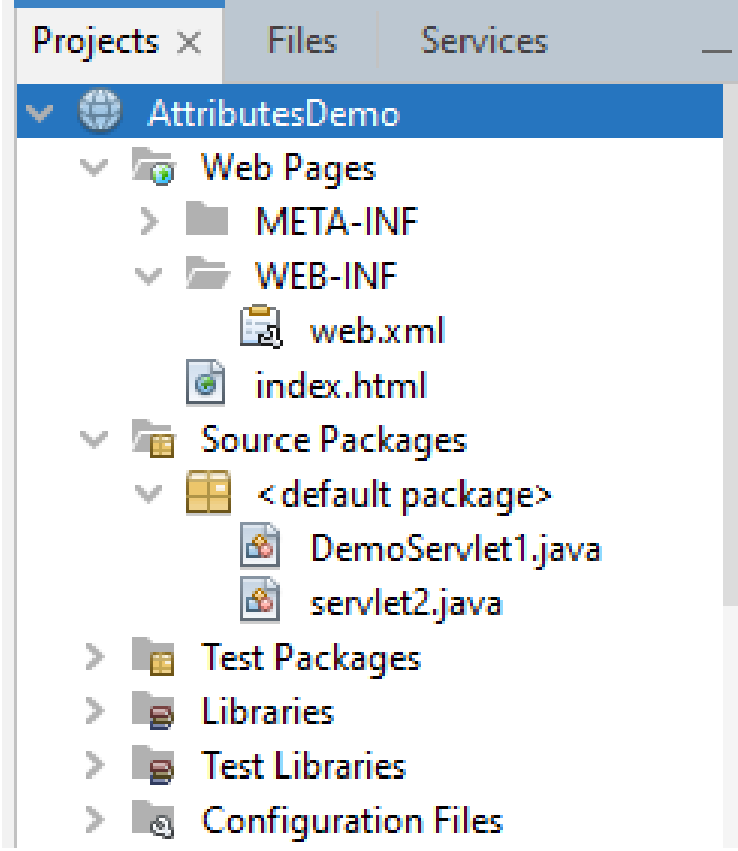
Example: Attributes in Servlet

Check Example → DemoServlet1, DemoServlet2

Example: Attributes in Servlet

web.xml

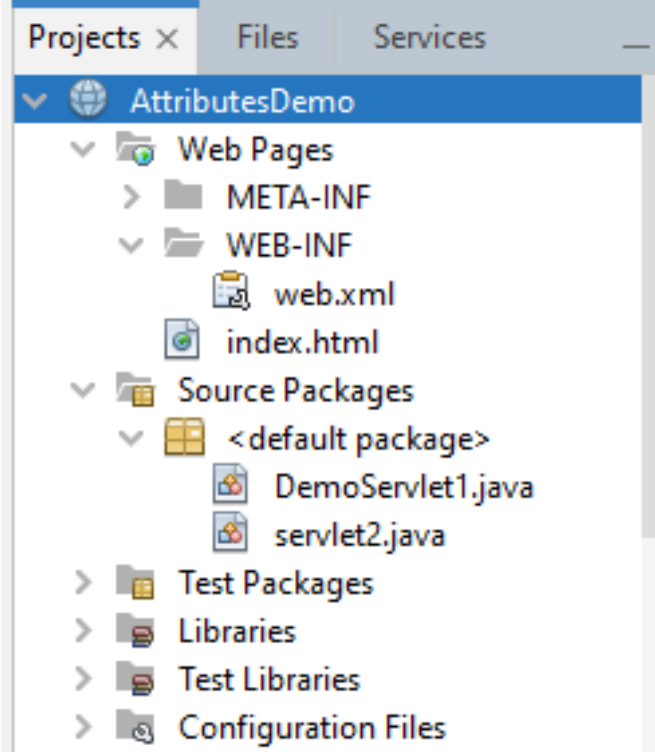
```
<servlet>
    <servlet-name>DemoServlet1</servlet-name>
    <servlet-class>DemoServlet1</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>servlet2</servlet-name>
    <servlet-class>servlet2</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>DemoServlet1</servlet-name>
    <url-pattern>/DemoServlet1</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>servlet2</servlet-name>
    <url-pattern>/servlet2</url-pattern>
  </servlet-mapping>
  <context-param>
    <param-name>org </param-name>
    <param-value>MEFGI</param-value>
  </context-param>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
```



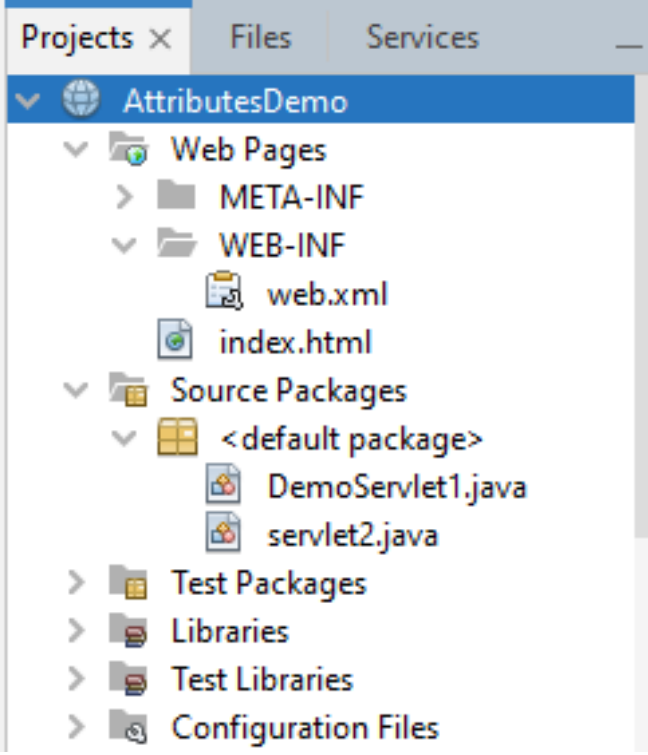
Example: Attributes in Servlet

index.html

```
<html>
   <head>
      <title>TODO supply a title</title>
      <meta charset="UTF-8">
      <meta name="viewport" content="width=device-width,
initial-scale=1.0">
   </head>
   <body>
      <a href="DemoServlet1">Click Here</a>
   </body>
</html>
```



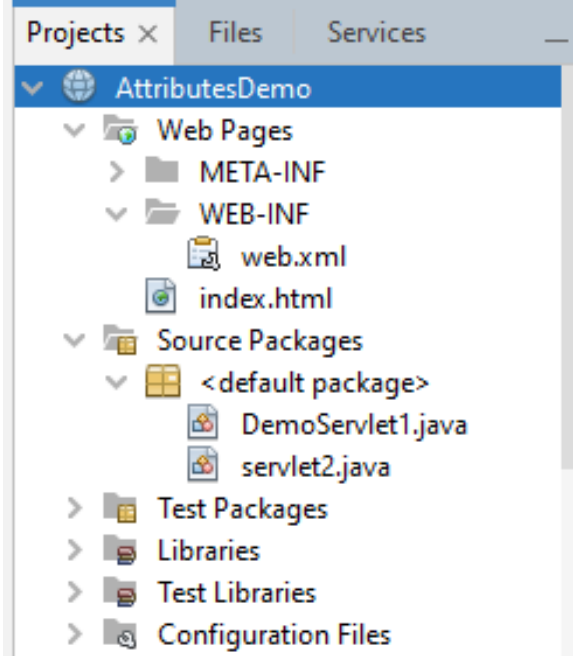
Example: Attributes in Servlet



DemoServlet1.html

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class DemoServlet1 extends HttpServlet {
 protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {
      ServletContext context=getServletContext();
      String o = (String)context.getInitParameter("org");
      out.println("Welcome to our organisation " + o);

      context.setAttribute("company","Amazon");
      out.println("<a href='servlet2'>visit</a>");
      out.close();
```

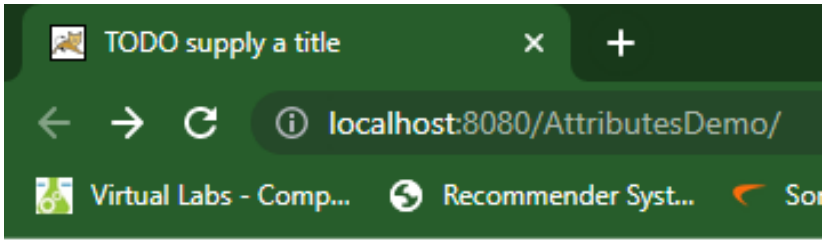
Example: Attributes in Servlet

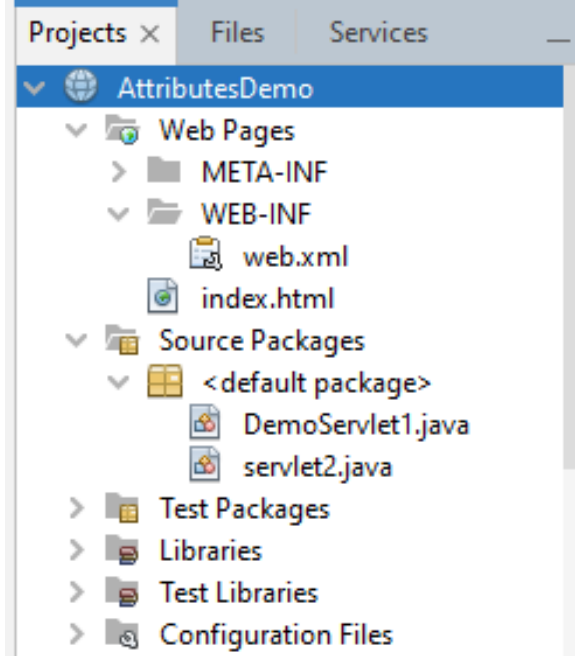


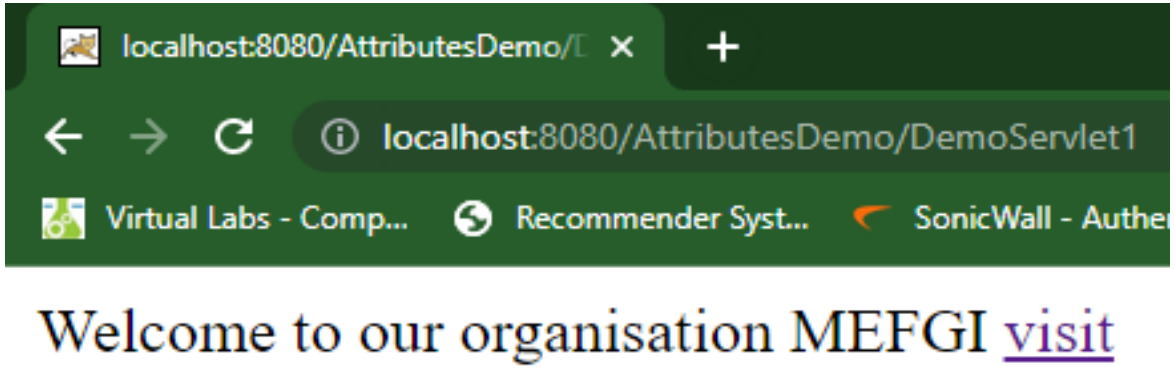
Servlet2.html

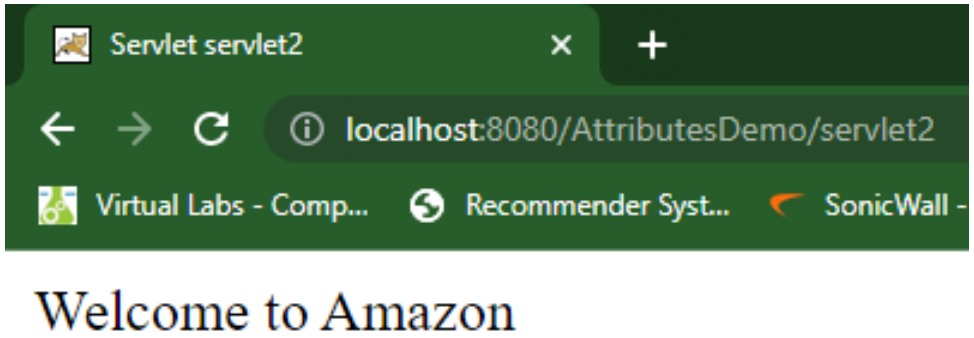
```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class servlet2 extends HttpServlet {
protected void processRequest(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        try (PrintWriter out = response.getWriter()) {

            ServletContext context=getServletContext();
            String n=(String)context.getAttribute("company");

            out.println("Welcome to "+n);

            out.close();
  } }   }
```

Example: Attributes in Servlet









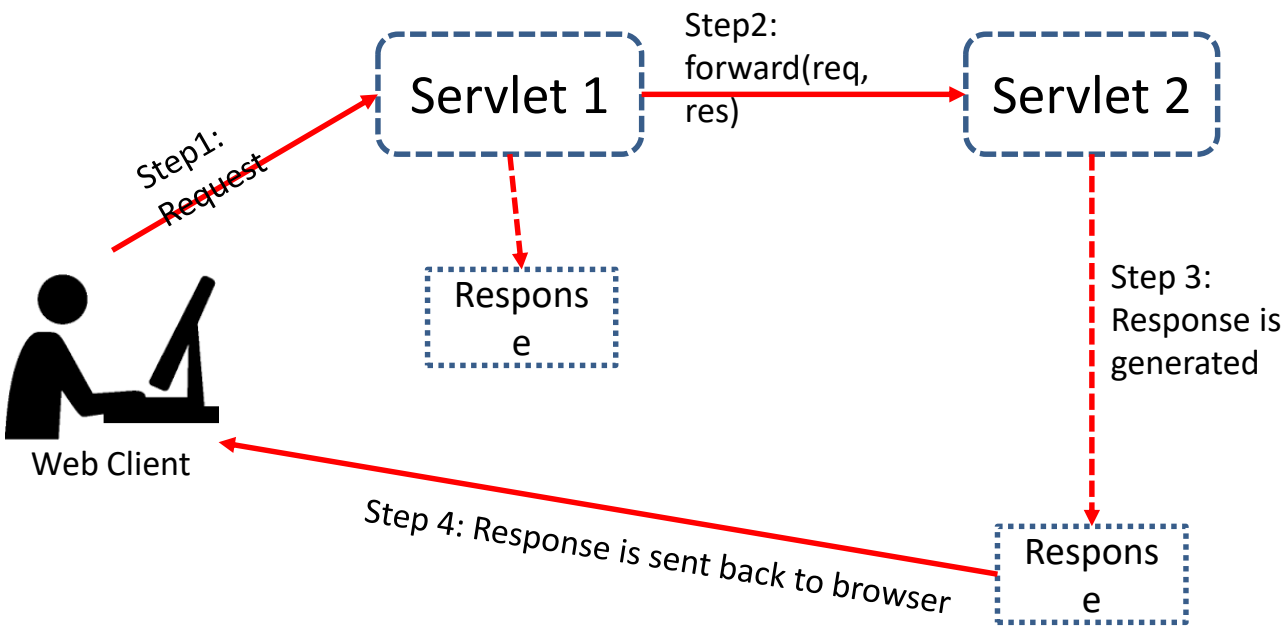
Request Dispatcher

The RequestDispatcher interface provides the facility of dispatching the request to another resource it may be html, servlet or jsp. This interface can also be used to include the content of another resource also. It is one of the way of servlet collaboration.

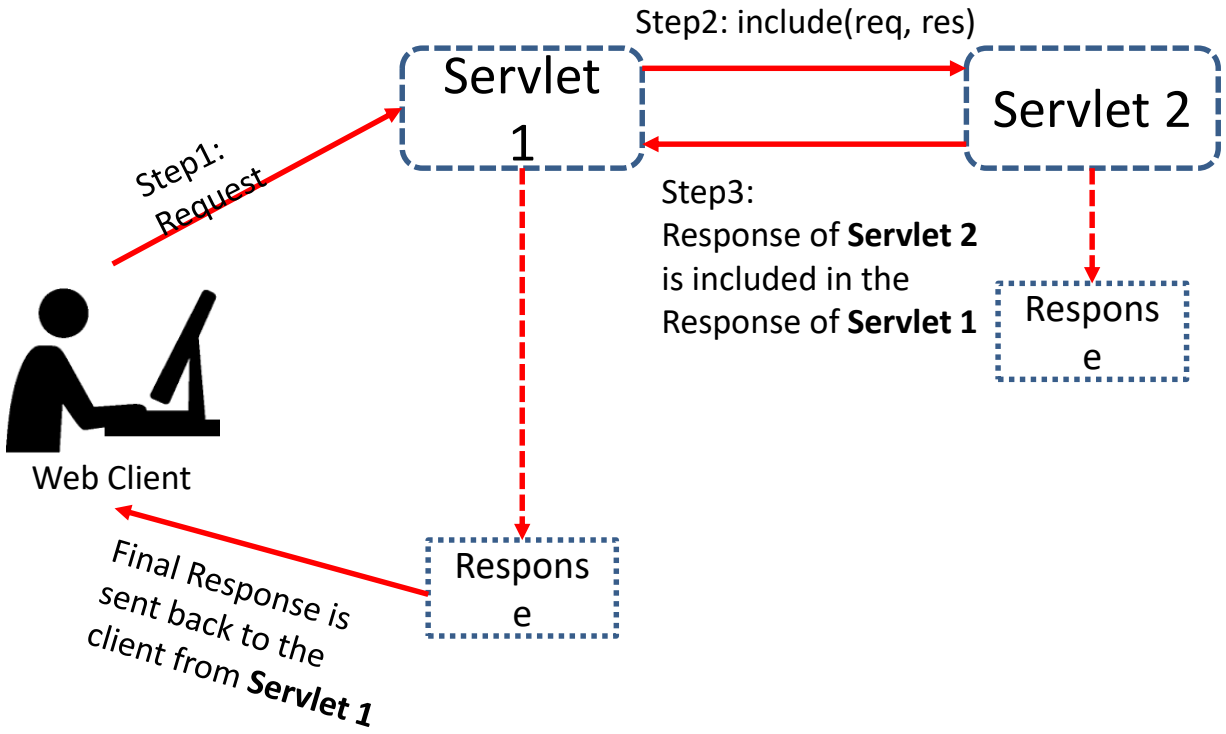
There are two methods defined in the RequestDispatcher interface.

1. **public void forward(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Forwards a request from a servlet to another resource (servlet, JSP file, or HTML file) on the server.
2. **public void include(ServletRequest request,ServletResponse response)throws ServletException,java.io.IOException:**Includes the content of a resource (servlet, JSP page, or HTML file) in the response.

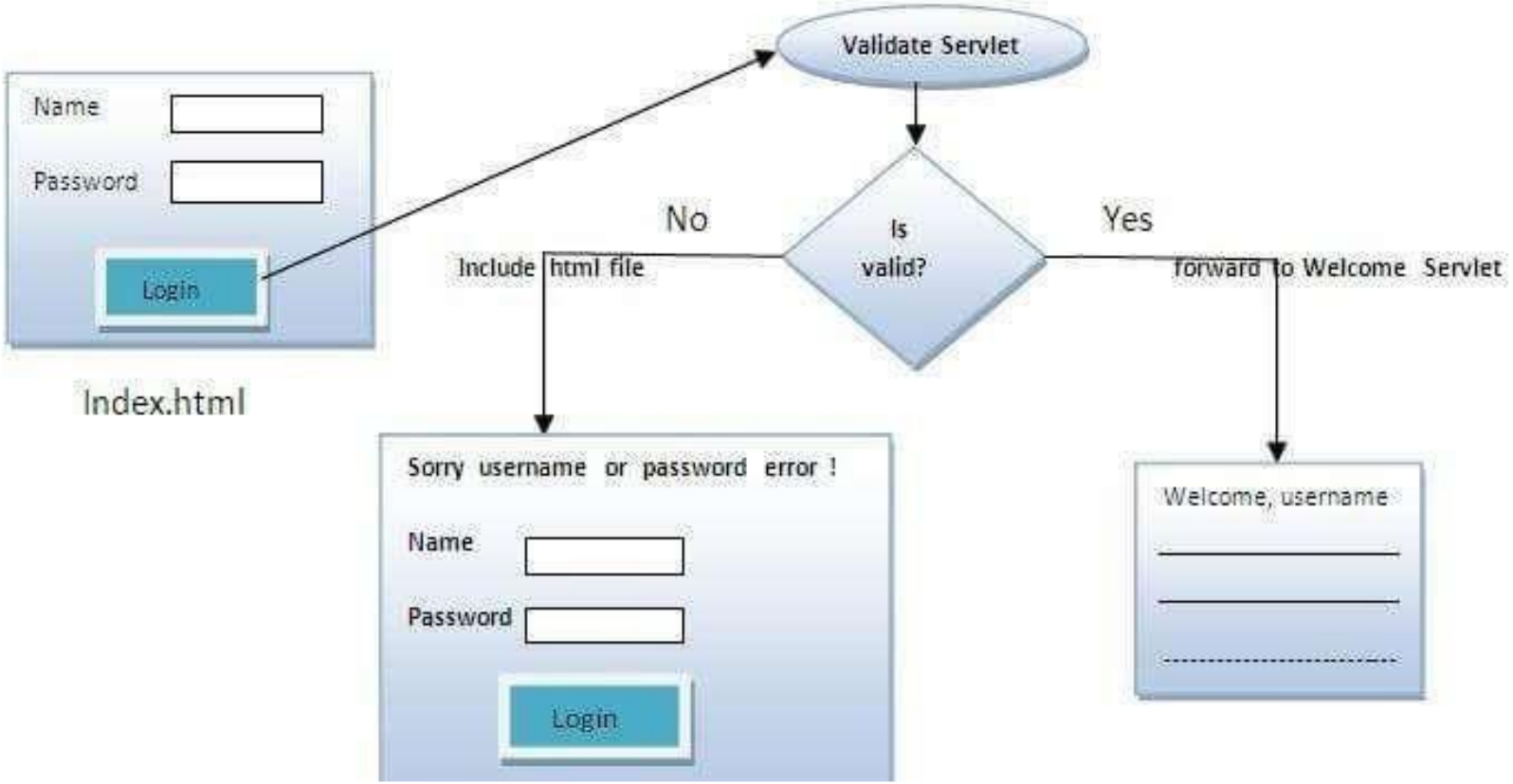
RequestDispatcher: forward()



RequestDispatcher: include()



Example : Request Dispatcher



Example : Request Dispatcher

In this example, we have created following files:

index.html file: for getting input from the user.

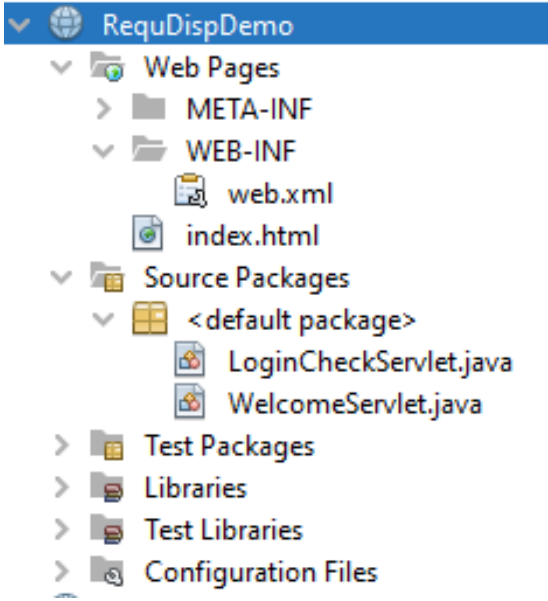
Login.java file: a servlet class for processing the response. If password is servet, it will forward the request to the welcome servlet.

WelcomeServlet.java file: a servlet class for displaying the welcome message.

web.xml file: a deployment descriptor file that contains the information about the servlet.

Check Program → requestdispatcher

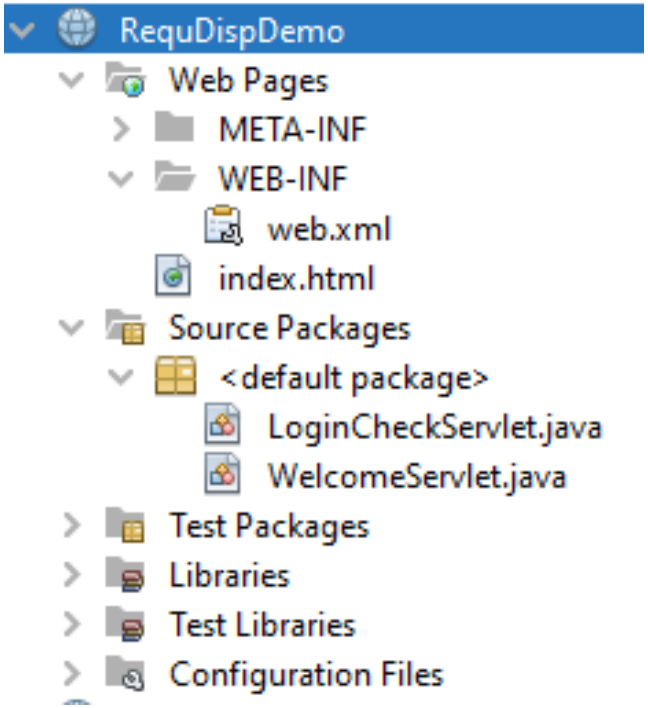
Example : Request Dispatcher



Web.xml

```
<servlet>
    <servlet-name>LoginCheckServlet</servlet-name>
    <servlet-class>LoginCheckServlet</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>WelcomeServlet</servlet-name>
    <servlet-class>WelcomeServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>LoginCheckServlet</servlet-name>
    <url-pattern>/LoginCheckServlet</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>WelcomeServlet</servlet-name>
    <url-pattern>/WelcomeServlet</url-pattern>
  </servlet-mapping>
  <session-config>
    <session-timeout>
      30
    </session-timeout>
  </session-config>
```

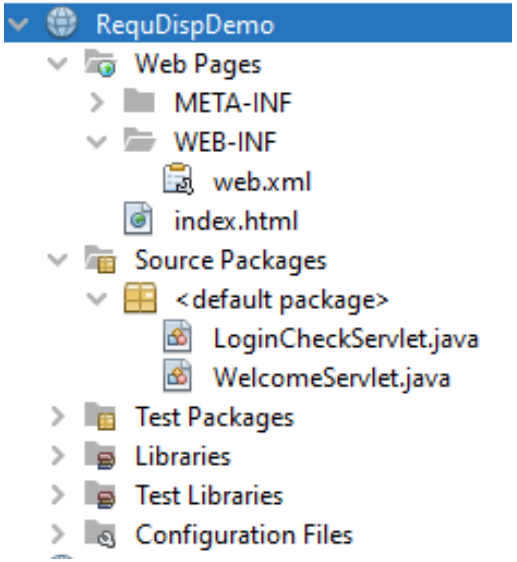
Example : Request Dispatcher



index.html

```
<form action="LoginCheckServlet" method="post">
        Enter Name: <input type="text" name="txtuname"/><br>
        Enter Password: <input type="password" name="txtpwd"/><br>
        <input type="submit" value="Login"/>
    </form>
```

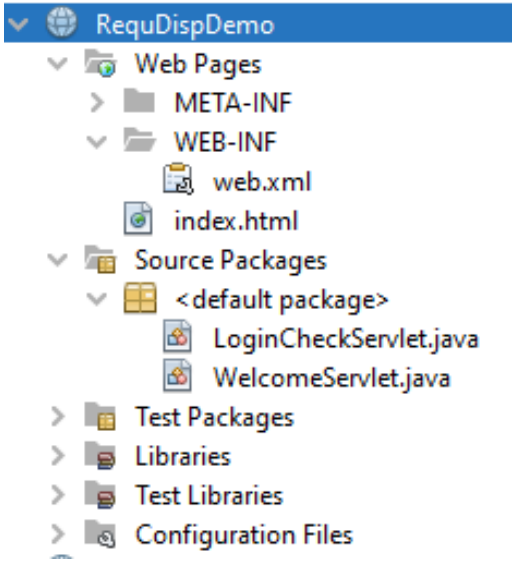
Example : Request Dispatcher



LoginCheckServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class LoginCheckServlet extends HttpServlet {
  @Override
   protected void doPost(HttpServletRequest request, HttpServletResponse response)
         throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      try (PrintWriter out = response.getWriter()) {
        String unm = request.getParameter("txtuname");
         String pass = request.getParameter("txtpwd");

         if(unm.equals("abc") && pass.equals("123"))
         {
            out.println("You're valid user");
              RequestDispatcher rd = request.getRequestDispatcher("WelcomeServlet");
              rd.forward(request,response)    }
         else          {
            out.println("User name or password is wrong.... Re-enter");
               RequestDispatcher rd = request.getRequestDispatcher("index.html");
              rd.include(request,response);          }      }   }  }
```

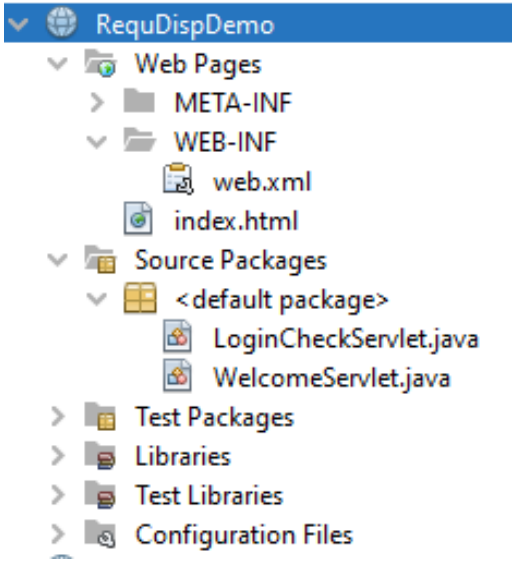
Example : Request Dispatcher

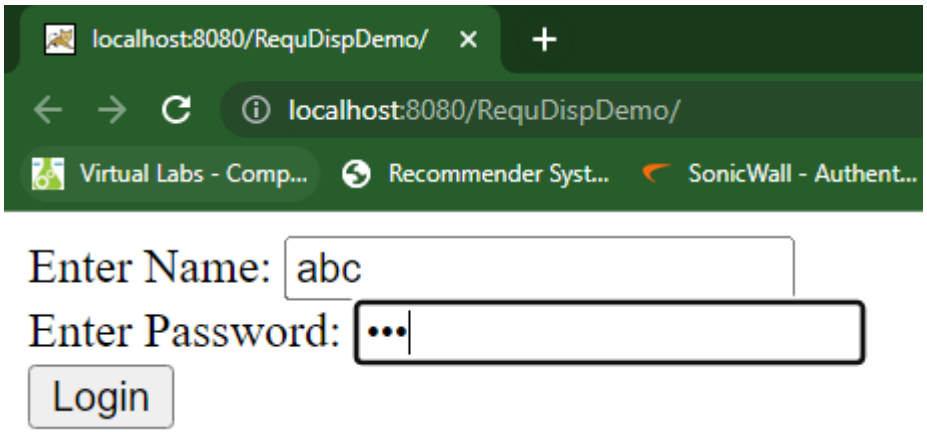


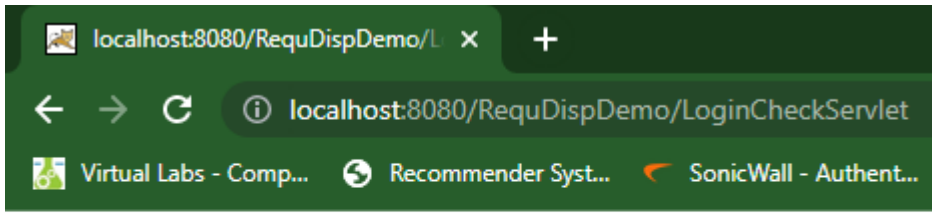
WelcomeServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

public class WelcomeServlet extends HttpServlet {

protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    response.setContentType("text/html;charset=UTF-8");
    try (PrintWriter out = response.getWriter()) {

        out.println("<h1>Welcome " + request.getParameter("txtuname") + "</h1>");

    }
  }
```

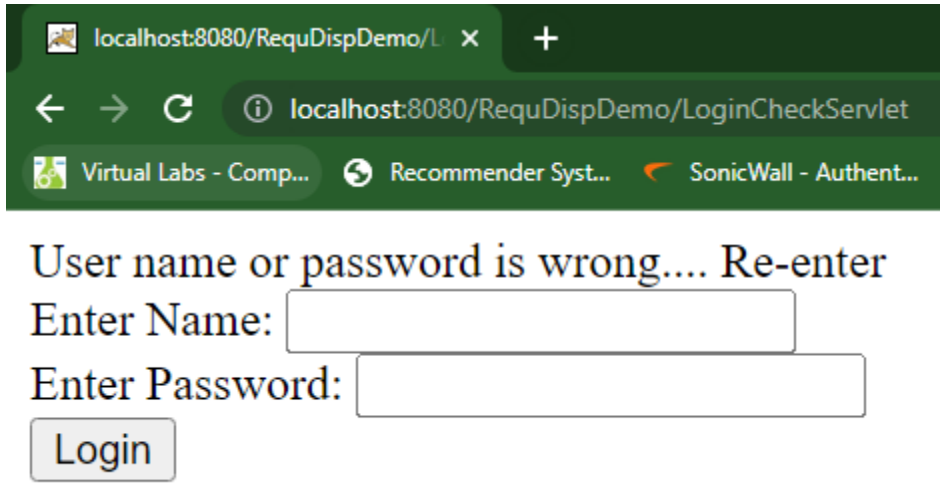
Example : Request Dispatcher







For wrong user/password



SendRedirect Method in servlet

The **sendRedirect()** method of **HttpServletResponse** interface can be used to redirect response to another resource, it may be servlet, jsp or html file.

It accepts relative as well as absolute URL.

It works at client side because it uses the url bar of the browser to make another request. So, it can work inside and outside the server.

Syntax of sendRedirect() method

public void sendRedirect(String URL)throws IOException;

Example

response.sendRedirect("http://www.youtube.com");

Check Example → GoogleSearch

SendRedirect Method in servlet

Web.xml

```
<servlet>
 <servlet-name>MySearcher</servlet-name>
<servlet-class>MySearcher</servlet-class>
</servlet>
<servlet-mapping>
<servlet-name>MySearcher</servlet-name>
 <url-pattern>/MySearcher</url-pattern>
</servlet-mapping>
```

Index.html

```
<form action="Servlet1">
<input type="text" name="name">
<input type="submit" value="YouTube
Search">
 </form>
```

SendRedirect Method in servlet


```
import java.io.IOException;
  import javax.servlet.ServletException;
  import javax.servlet.http.HttpServlet;
  import javax.servlet.http.HttpServletRequest;
  import javax.servlet.http.HttpServletResponse;

  public class Servlet1 extends HttpServlet {
     @Override
     protected void doGet(HttpServletRequest req, HttpServletResponse resp)
           throws ServletException, IOException {

        String name=req.getParameter("name");
       resp.sendRedirect("https://www.google.co.in/search?q="+name);

       //resp.sendRedirect("https://www.youtube.com/results?search_query=" + name);
     }
  }
```

Difference

forward() method	sendRedirect() method
The forward() method works at server side.	The sendRedirect() method works at client side.
It sends the same request and response objects to another servlet.	It always sends a new request.
It can work within the server only.	It can be used within and outside the server.
Example: request.**getRequestDispacher**("servlet2").forward(request,response);	Example: response.sendRedirect("servlet2");

Filter

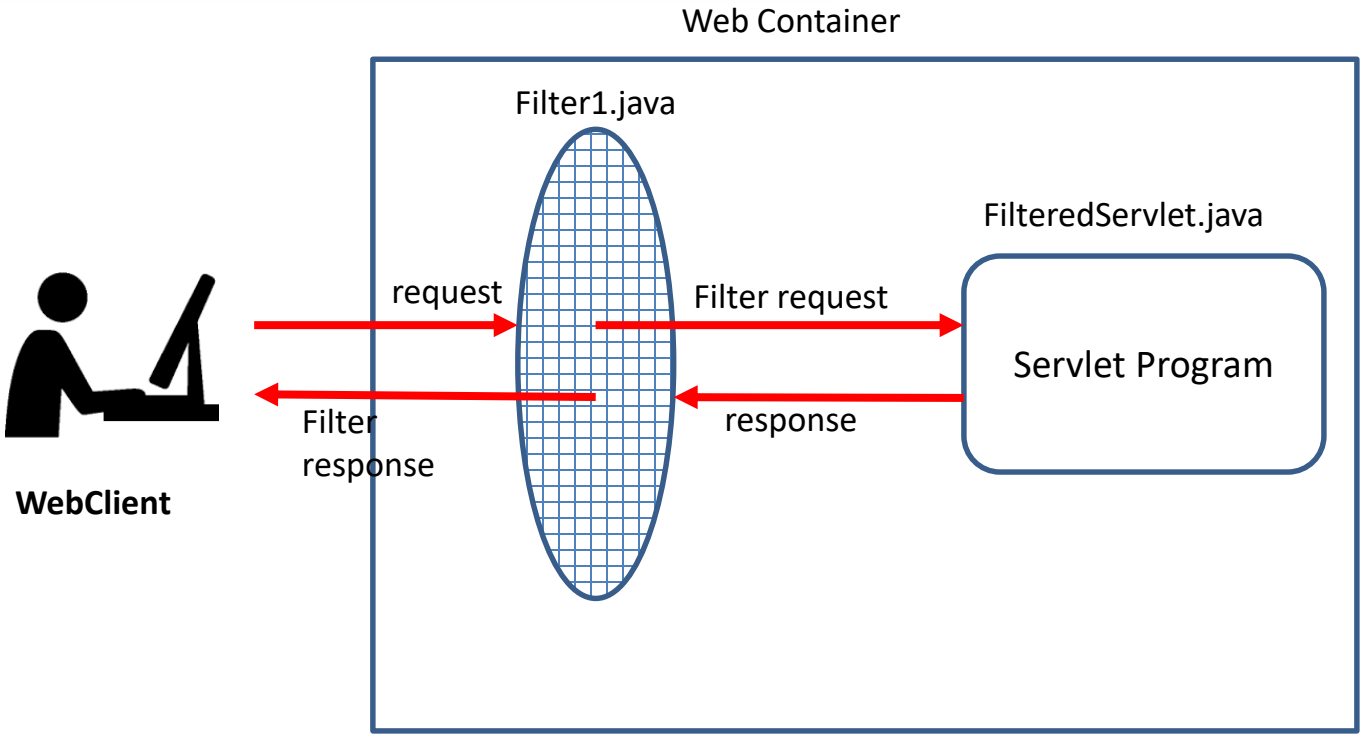
A **filter** is an object that is invoked at the preprocessing and postprocessing of a request.

It is mainly used to perform filtering tasks such as conversion, logging, compression, encryption and decryption, input validation etc.

The **servlet filter is pluggable**, i.e. its entry is defined in **the web.xml file**, if we remove the entry of filter from the web.xml file, filter will be removed automatically and we don't need to change the servlet.

So maintenance cost will be less.

Filter Example



Filter

Usage of Filter

recording all incoming requests

logs the IP addresses of the computers from which the requests originate

conversion

data compression encryption and

decryption input validation etc.

Advantage of Filter

Filter is pluggable.

One filter don't have dependency onto another resource.

Less Maintenance

Filter API

Like servlet filter have its own API. The javax.servlet package contains the three interfaces of Filter API.

1.Filter Interface
2.FilterChain Interface
3.FilterConfig

1. Filter Interface

For creating any filter, you must implement the Filter interface.

Filter interface provides the life cycle methods for a filter.

1. **public void init(FilterConfig config)**
   init() method is invoked only once. It is used to initialize the filter.

2. **public void doFilter(HttpServletRequest request,HttpServletResponse response, FilterChain chain)**
   doFilter() method is invoked every time when user request to any resource, to which the filter is mapped.It is used to perform filtering tasks.

3. **public void destroy()**
   This is invoked only once when filter is taken out of the service.

2. FilterChain interface

The object of FilterChain is responsible to invoke the next filter or resource in the chain. This object is passed in the doFilter method of Filter interface.

The FilterChain interface contains **only one method**:

public void doFilter(HttpServletRequest request, HttpServletResponse response): it passes the control to the next filter or resource.

3. FilterConfig

An object of FilterConfig is created by the web container. This object can be used to get the configuration information from the web.xml file.

Methods of FilterConfig interface

There are following 4 methods in the FilterConfig interface.

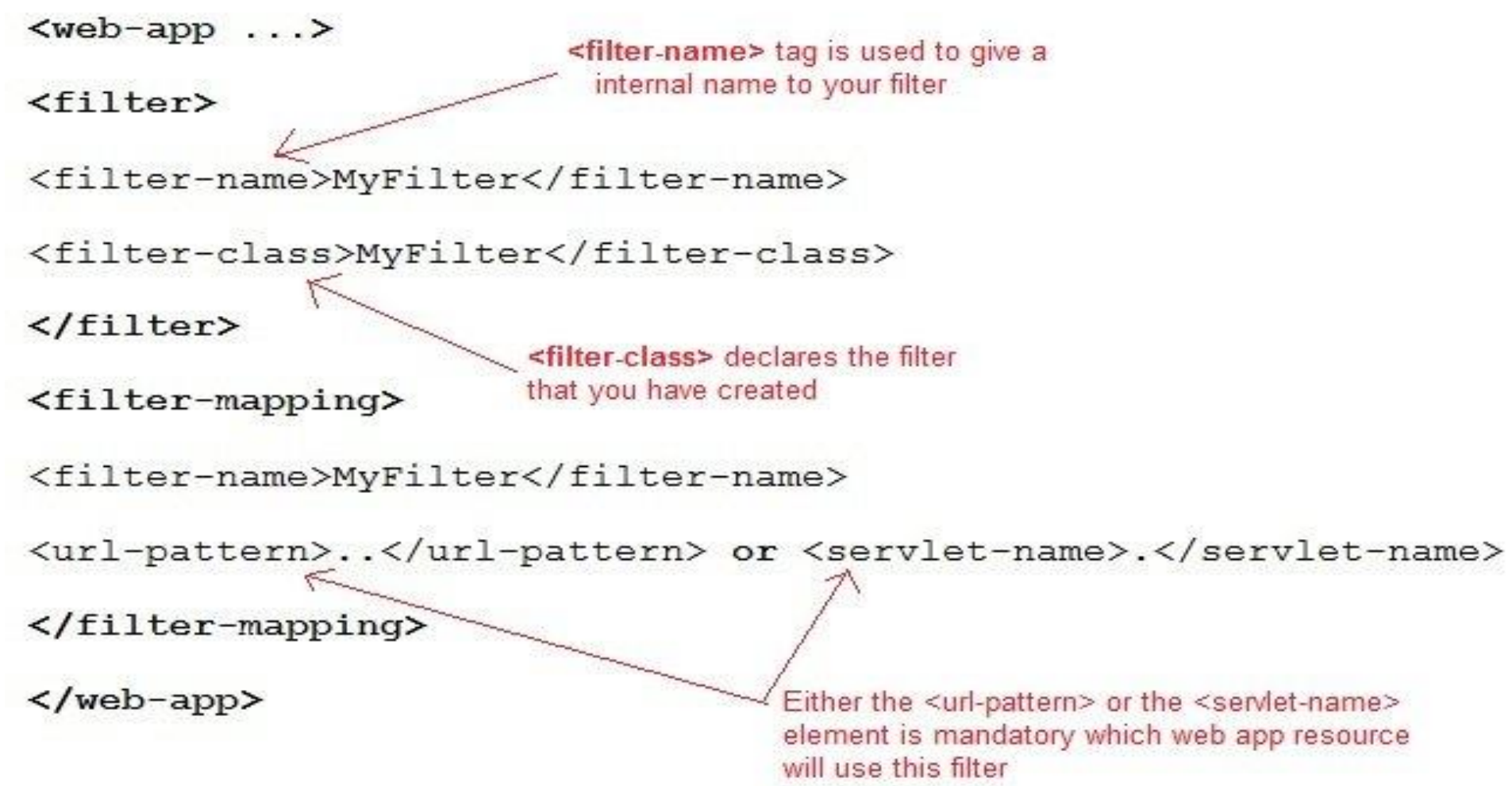
public void init(FilterConfig config): init() method is invoked only once it is used to initialize the filter.

public String getInitParameter(String parameterName): Returns the parameter value for the specified parameter name.

public java.util.Enumeration getInitParameterNames(): Returns an enumeration containing all the parameter names.

public ServletContext getServletContext(): Returns the ServletContext object.

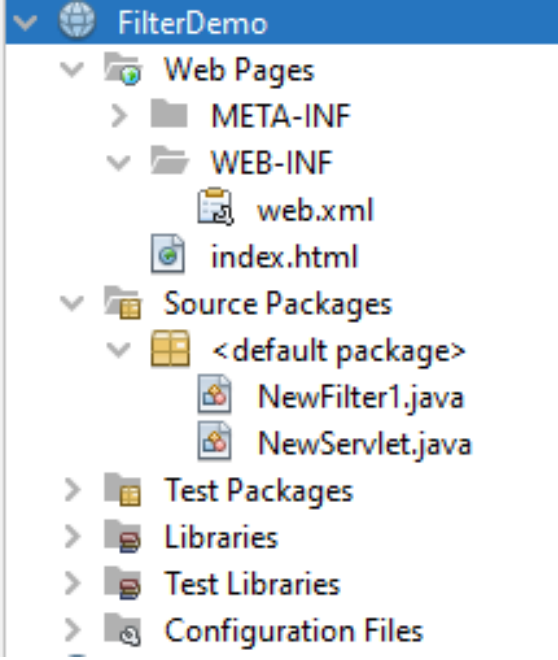
Declaring a Servlet Filter inside Deployment Descriptor



Example Filter

Check Program → filter1, filterconfig

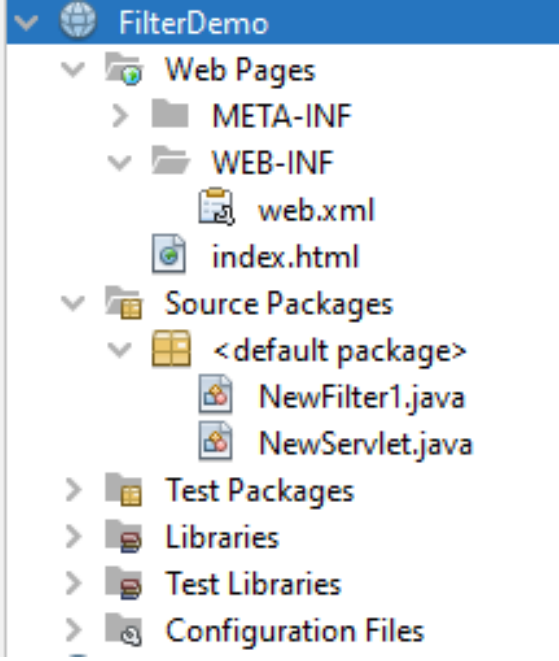
Example Filter



Web.xml

```
<filter>
        <filter-name>NewFilter1</filter-name>
        <filter-class>NewFilter1</filter-class>
    </filter>
    <filter-mapping>
        <filter-name>NewFilter1</filter-name>
        <url-pattern>/s11</url-pattern>
    </filter-mapping>
    <servlet>
        <servlet-name>NewServlet</servlet-name>
        <servlet-class>NewServlet</servlet-class>
    </servlet>
    <servlet-mapping>
        <servlet-name>NewServlet</servlet-name>
        <url-pattern>/s11</url-pattern>
    </servlet-mapping>
```

Example Filter



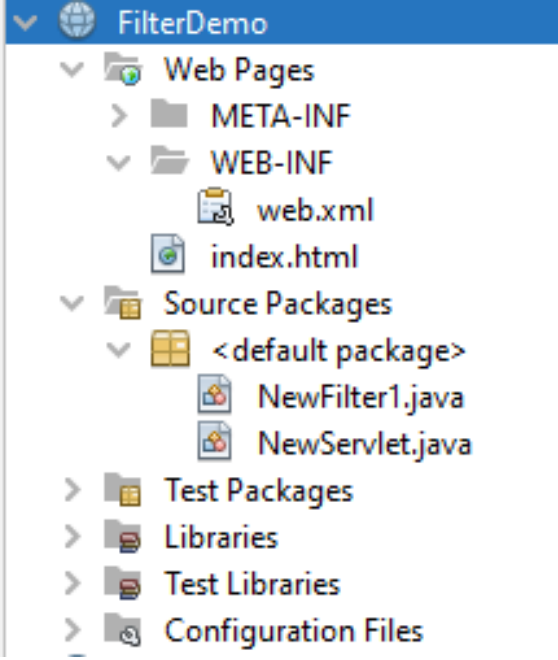
Web.xml

```
<filter>
    <filter-name>NewFilter1</filter-name>
    <filter-class>NewFilter1</filter-class>
  </filter>
  <filter-mapping>
    <filter-name>NewFilter1</filter-name>
    <url-pattern>/s11</url-pattern>
  </filter-mapping>
  <servlet>
    <servlet-name>NewServlet</servlet-name>
    <servlet-class>NewServlet</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>NewServlet</servlet-name>
    <url-pattern>/s11</url-pattern>
  </servlet-mapping>
```

Index.html

```
<form action="s1">
    Name:<input type="text" name="txtName"/><br>
    Password:<input type="password" name="txtPassword"/><br>
```

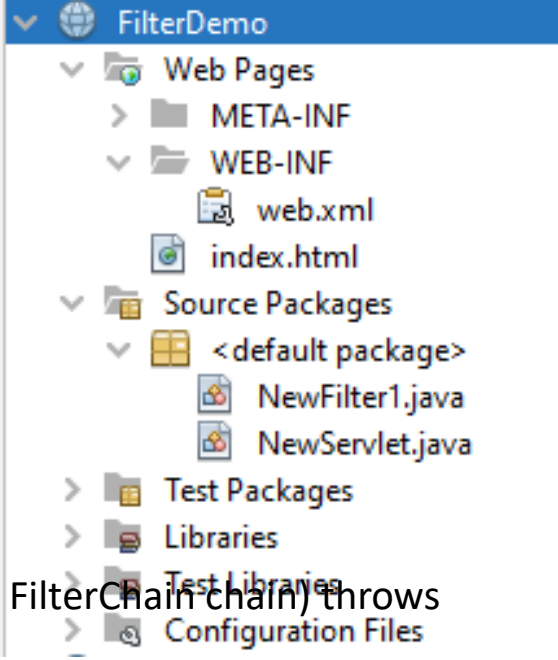
Example Filter



NewServlet.java

```
import java.io.IOException;
import java.io.PrintWriter;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
public class NewServlet extends HttpServlet {


    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
            response.setContentType("text/html");
            PrintWriter out = response.getWriter();

            String name = request.getParameter("txtName");
            out.print("Welcome to Filter program: " +name);
            out.close();
    }
}
```

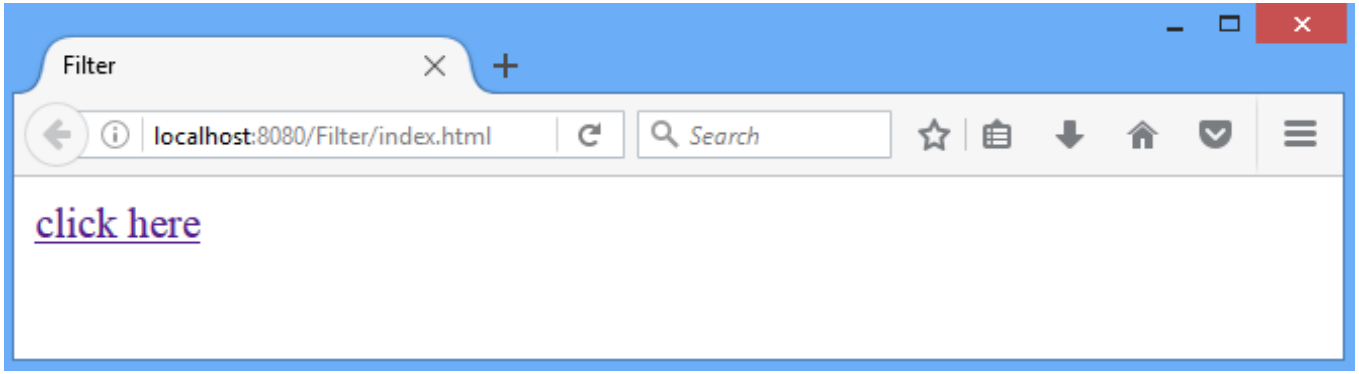
Example Filter

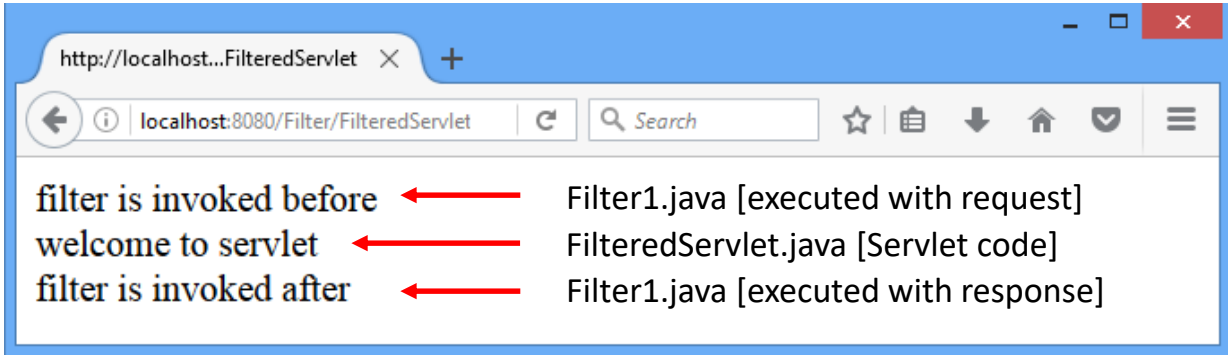


NewServlet.java

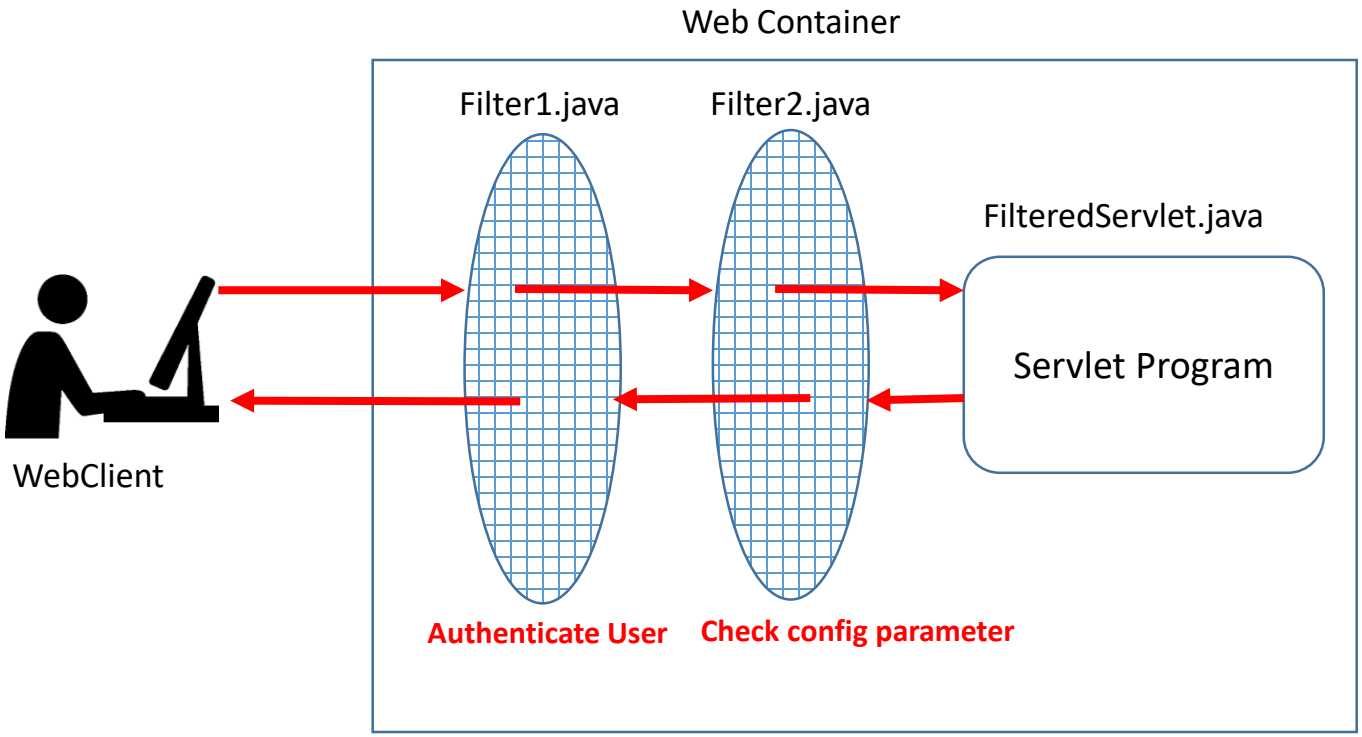
```
import java.io.*;
import javax.servlet.*;
public class NewFilter1 implements Filter {

    @Override
    public void init(FilterConfig filterConfig) throws ServletException { }
    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain) throws
IOException, ServletException {
        PrintWriter out = response.getWriter();

     String password=request.getParameter("txtPassword");
       if(password.equals("admin")){
       chain.doFilter(request, response);//sends request to next resource
       }
       else{
       out.print("username or password error!");
       RequestDispatcher rd=request.getRequestDispatcher("index.html");
       rd.include(request, response);
       }
    }
public void destroy() { }
}
```

Example Filter





Filter Example-2



Activity

Marwadi education foundation

Mentimeter

Session Tracking in Servlets

Marwadi education foundation

Session simply 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.

Why use Session Tracking?

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

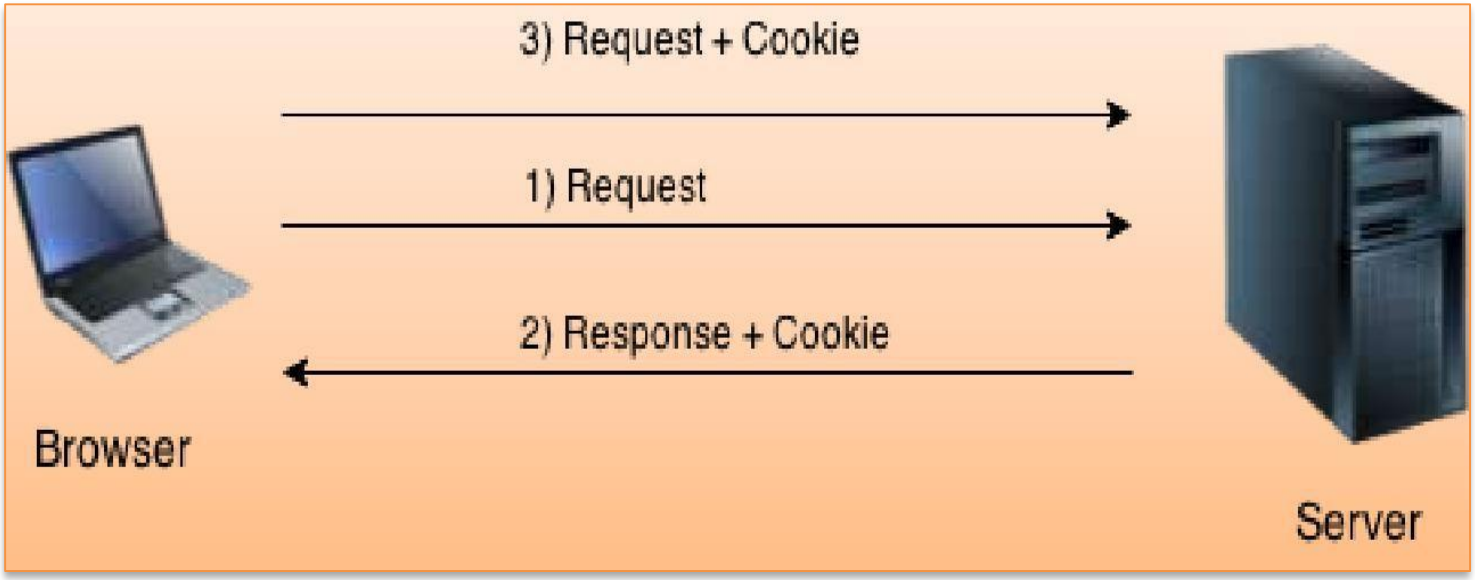
Session Tracking Techniques

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.



Cookies in Servlet

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 (because HTTP is stateless). 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**.

Cookies

Types of Cookie There are 2 types of cookies in servlets.

1. **Non-persistent cookie**

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

2. **Persistent cookie**

It is **valid for multiple session** . It is not removed each time when user closes the browser. It is removed only if user logout or signout.

Advantage of Cookies

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

Disadvantage of Cookies

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

Cookies

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.

Cookies

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.

Cookies

Other methods required for using Cookies:-

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","aarav"); //creating cookie object

response.addCookie(ck);  //adding cookie in the response
```

Cookies

How 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
```

Cookies can be removed by setting its expiration time to 0 or -1. If expiration time set to 0 than cookie will be removed immediately. If expiration time set to -1 than cookie will be removed when browser closed.

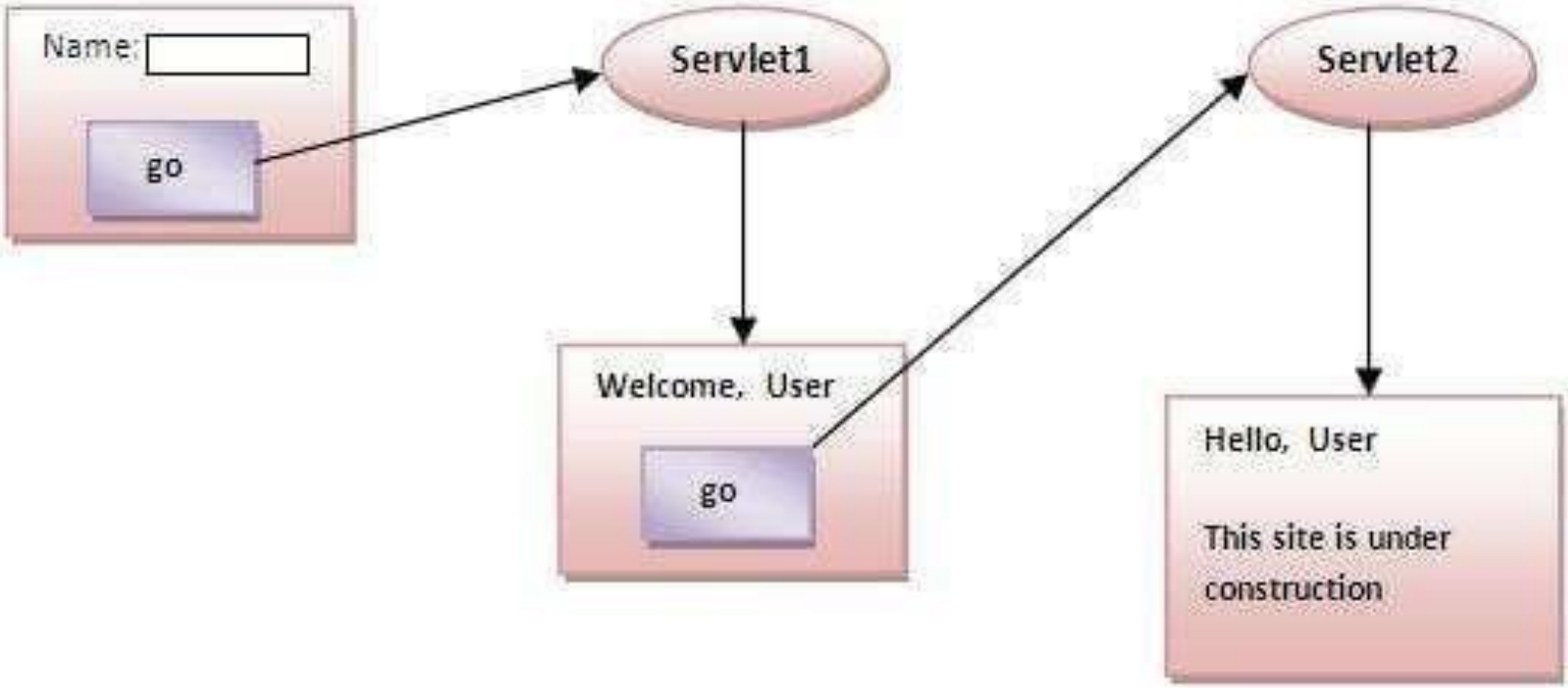
Cookies

How to get Cookies?

sample code to get all the cookies names and values.

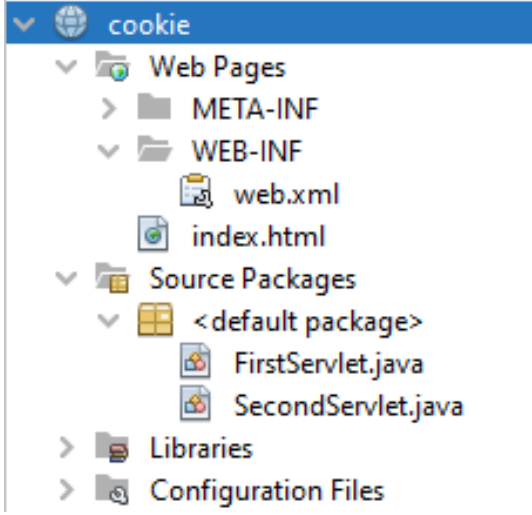
```
Cookie ck[]=request.getCookies();

for(int i=0;i<ck.length;i++){

 out.print("<br>"+ck[i].getName()+" "+ck[i].getValue());

}
```

Session Tracking Techniques



Check Program → cookie

Cookie Example



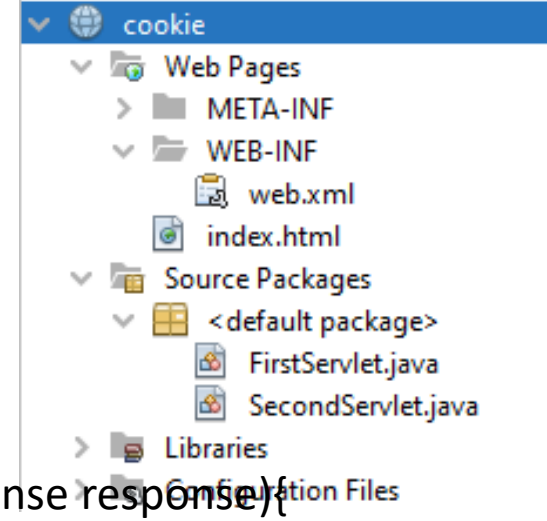
Web.xml

```
<servlet>
  <description>abcd</description>
  <display-name>abcd</display-name>
  <servlet-name>Servlet1</servlet-name>
  <servlet-class>FirstServlet</servlet-class>
 </servlet>
 <servlet>
  <description>xyz</description>
  <display-name>xyz</display-name>
  <servlet-name>SecondServlet</servlet-name>
  <servlet-class>SecondServlet</servlet-class>
 </servlet>
 <servlet-mapping>
  <servlet-name>Servlet1</servlet-name>
  <url-pattern>/servlet1</url-pattern>
 </servlet-mapping>
 <servlet-mapping>
  <servlet-name>SecondServlet</servlet-name>
  <url-pattern>/servlet2</url-pattern>
 </servlet-mapping>
 <welcome-file-list>
  <welcome-file>index.html</welcome-file>
 </welcome-file-list>
```

Index.html

```
<form action="Servlet1" method="post">
Name:<input type="text"
name="userName"/><br/>
<input type="submit" value="go"/>
</form>
```

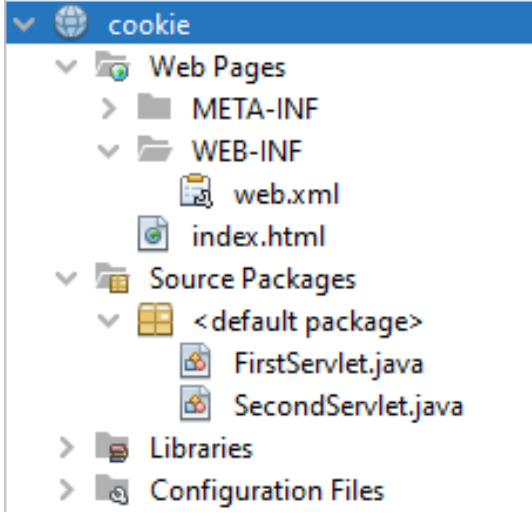
Cookie Example



FirstServlet.java

```
import java.io.*;
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

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

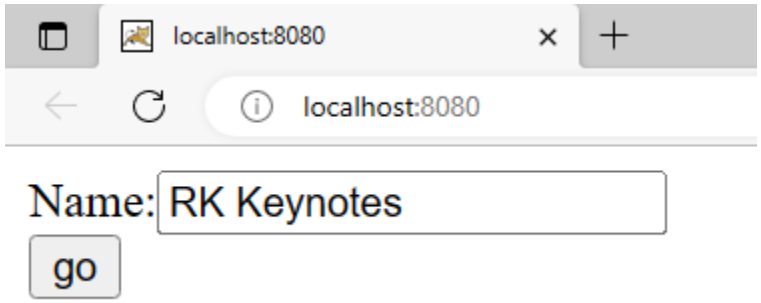
Cookie Example

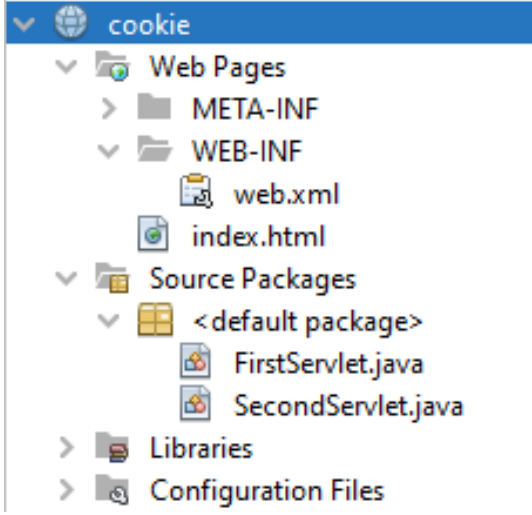


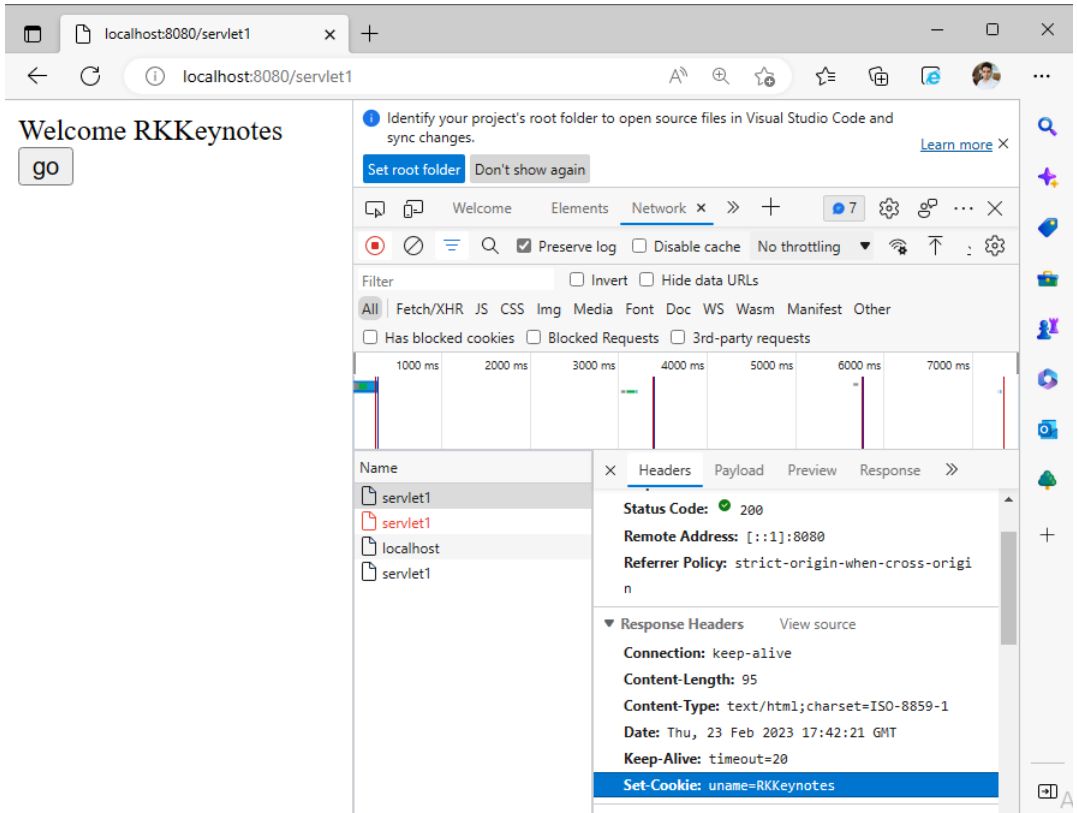
SecondServlet.java

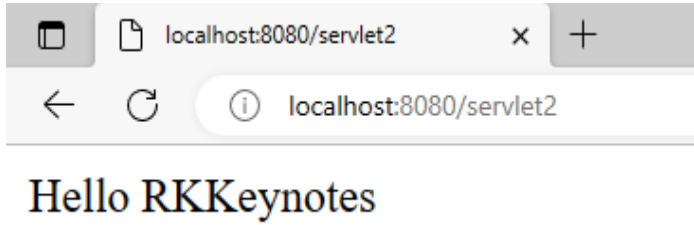
```
import java.io.*;
import javax.servlet.http.*;

public class SecondServlet extends HttpServlet {

@Override
public void doPost(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);}
        }  }
```

Cookie Example









Hidden Form Field

In case of Hidden Form Field **a hidden (invisible) textfield is used for maintaining the state of an user.**

In such case, we store the information in the hidden field and get it from another servlet. This approach is better if we have to submit form in all the pages and we don't want to depend on the browser.

<input type="hidden" name="uname" value="Ravi">

Here, **uname** is the hidden field name and **Ravi** is the hidden field value.

Real application of hidden form field

It is widely used in comment form of a website. In such case, we store page id or page name in the hidden field so that each page can be uniquely identified.

Hidden Form Field

The **welcome-file-list** element of **web-app**, is used to define a list of welcome files. Its sub element is **welcome-file** that is used to define the welcome file.

A welcome file is the file that is invoked automatically by the server, if you don't specify any file name.

By default server looks for the welcome file in following order:
1.welcome-file-list in web.xml
2.index.html
3.index.htm
4.index.jsp
If none of these files are found, server renders 404 error.

Hidden Form Field

Marwadi education foundation

Hidden Form Fields Real-world application examples:

- To remember user identity during sessions of email-operations.
- While developing shopping cart applications in online shopping websites where items selected by end-user generating multiple requests will be remembered.
- **To remember user identity during an e-commerce session.**
- To remember credit card/debit card details during a session while performing the web-based online transactions.
- To remember player identity while playing online games.
- To remember customer identity while performing online stock brokerage.
- To remember end-user choices and interests towards the look and appearance of web pages if the website allows customizing the look and appearance.
- To render direct advertisements on websites.

Hidden Form Field

Advantage of Hidden Form Field:

- It will always work whether cookie is disabled or not.

Disadvantage of Hidden Form Field:

- It is maintained at server side.
- Extra form submission is required on each pages.
- Only textual information can be used.
- The hidden box values of the form page can be viewed using the source code of the web page. That means there is no security (data secrecy is not there).

URL Rewriting

In URL rewriting, we **append** a token or identifier to the URL of the next Servlet or the next resource.

We can send parameter name/value pairs using the following format: **url?name1=value1&name2=value2&??**

A name and a value is separated using an equal = sign, a parameter name/value pair is separated from another parameter using the ampersand(&).

When the user clicks the hyperlink, the parameter name/value pairs will be passed to the server. From a Servlet, we can use getParameter() method to obtain a parameter value.

URL Rewriting

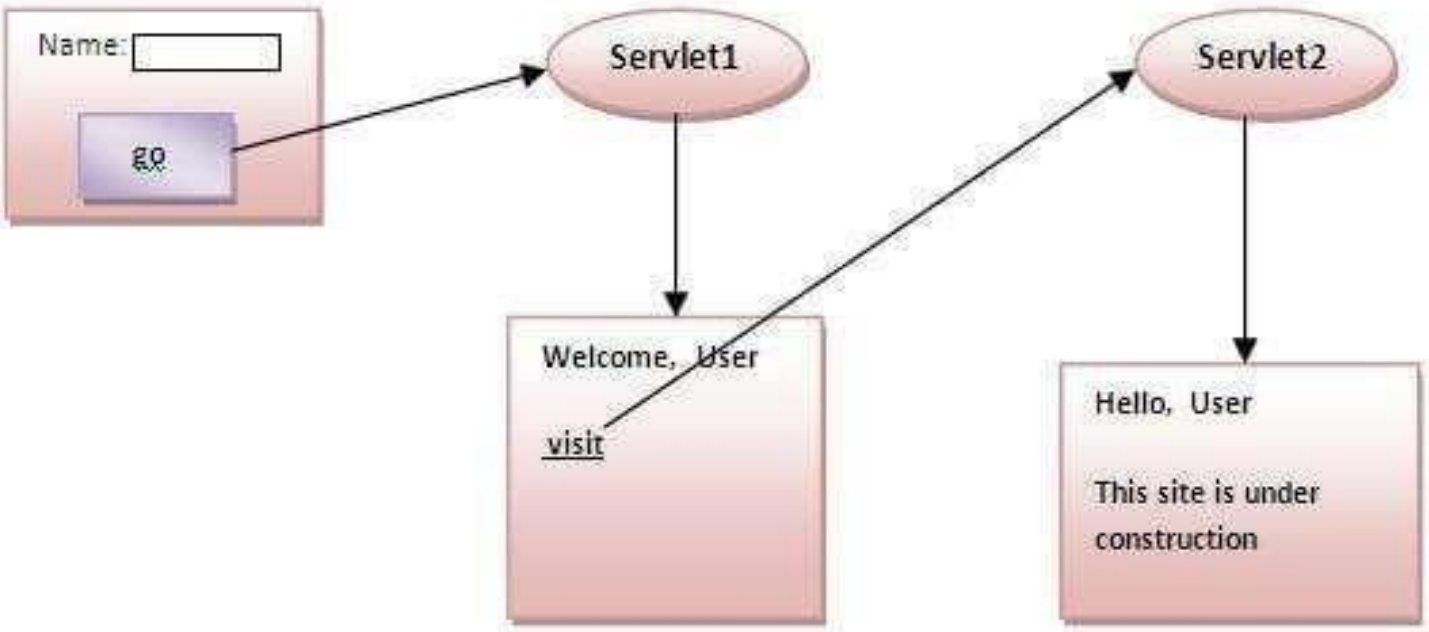
Advantage of URL Rewriting

- It will always work whether cookie is disabled or not (browser independent).
- Extra form submission is not required on each pages.

Disadvantage of URL Rewriting

- It will work only with links.
- It can send Only textual information.

URL Rewriting

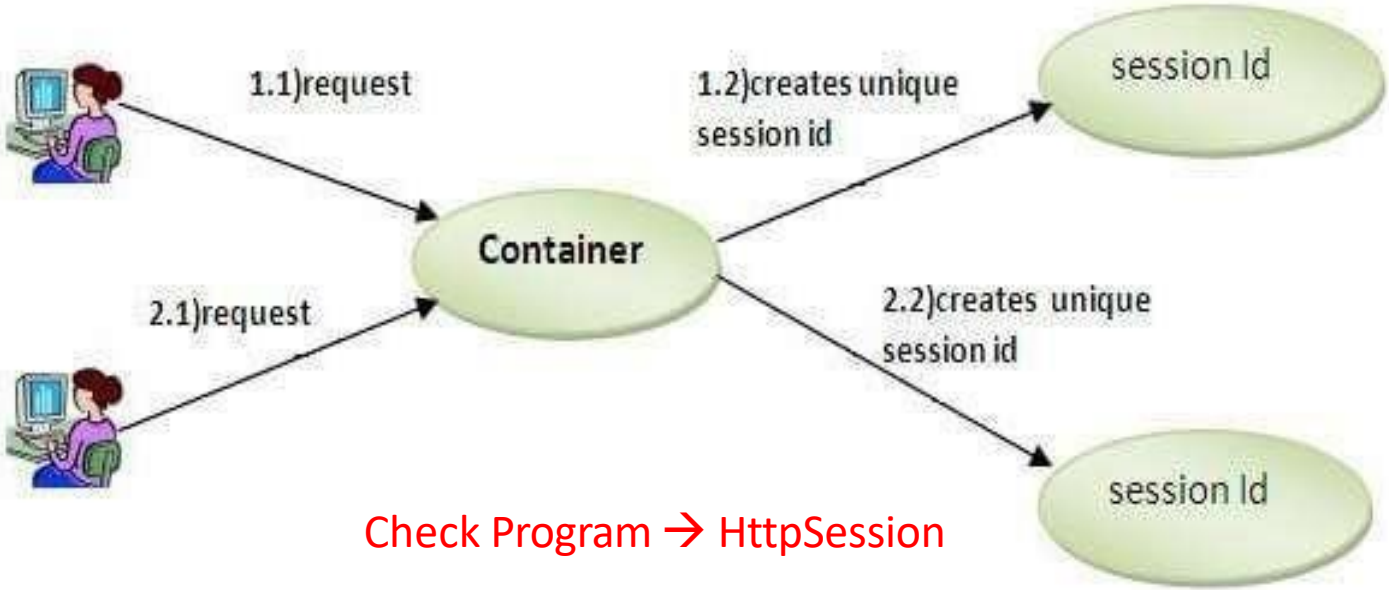


Check Program → urlrewritting

HttpSession interface

In such case, container creates a session id for each user. The container uses this id to identify the particular user. An object of HttpSession can be used to perform two tasks:

1. bind objects
2. view and manipulate information about a session, such as the session identifier, creation time, and last accessed time.



Check Program → HttpSession

HttpSession interface

1. On client's first request, the Web Container generates a unique session ID and gives it back to the client with response. This is a temporary session created by web container.
2. The client sends back the session ID with each request. Making it easier for the web container to identify where the request is coming from.
3. The Web Container uses this ID, finds the matching session with the ID and associates the session with the request.

Creating a new session

```
HttpSession session = request.getSession();
```

getSession() method returns a session. If the session already exist, it return the esisting session else create a new sesion

```
HttpSession session = request.getSession(true);
```

getSession(true) always return a new session

Getting a pre-existing session

```
HttpSession session = request.getSession(false);
```

return a pre-existing session

Destroying a session

```
session.invalidate();
```

destroy a session

HttpSession interface

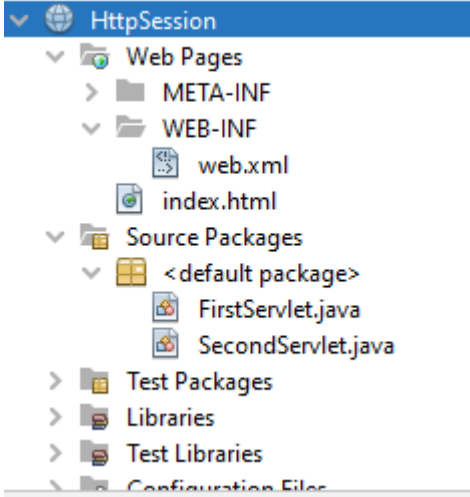
The HttpServletRequest interface provides two methods to get the object of HttpSession:

public HttpSession getSession(): Returns the current session associated with this request, or if the request does not have a session, creates one.
public HttpSession getSession(boolean create): Returns the current HttpSession associated with this request or, if there is no current session and create is true, returns a new session.

Commonly used methods of HttpSession interface

public String getId(): Returns a string containing the unique identifier value.
public long getCreationTime(): Returns the time when this session was created, measured in milliseconds since midnight January 1, 1970 GMT.
public long getLastAccessedTime(): Returns the last time the client sent a request associated with this session, as the number of milliseconds since midnight January 1, 1970 GMT.
public void invalidate(): Invalidates this session then unbinds any objects bound to it.

HttpSession Example



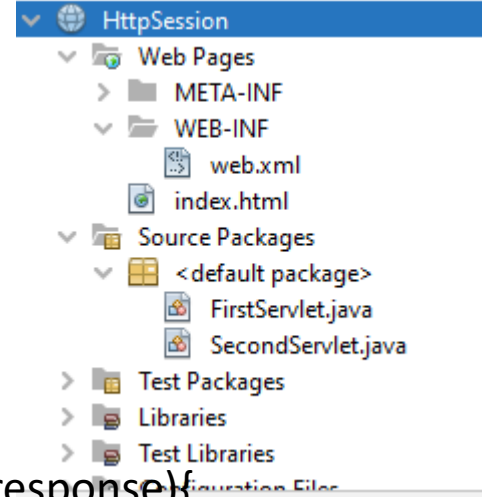
Web.xml

```
<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>
```

Index.html

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

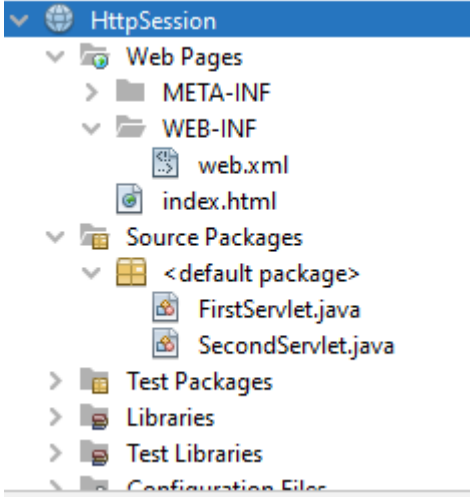
HttpSession Example



FirstServlet.java

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

public class FirstServlet extends HttpServlet {
public void doGet(HttpServletRequest request, HttpServletResponse response){
                    try{
                    response.setContentType("text/html");
                    PrintWriter out = response.getWriter();

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

                    HttpSession session=request.getSession();
                    session.setAttribute("uname",n);

                    out.print("<br><a href='servlet2'>visit</a>");
                    out.close();
          }catch(Exception e){System.out.println(e);}
           }
}
```

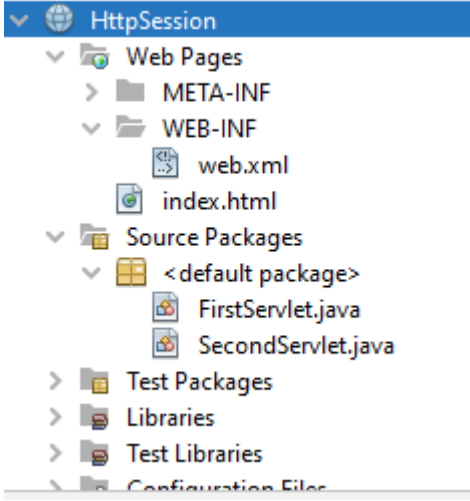
HttpSession Example

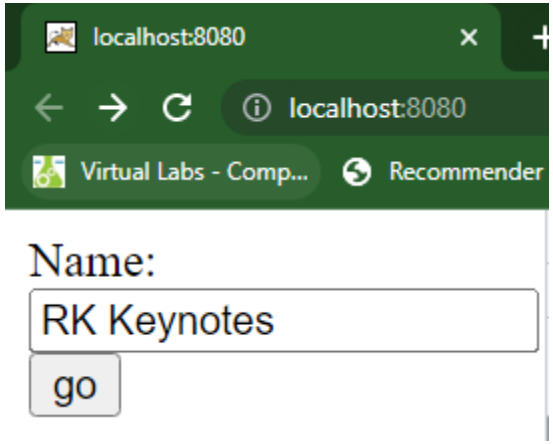


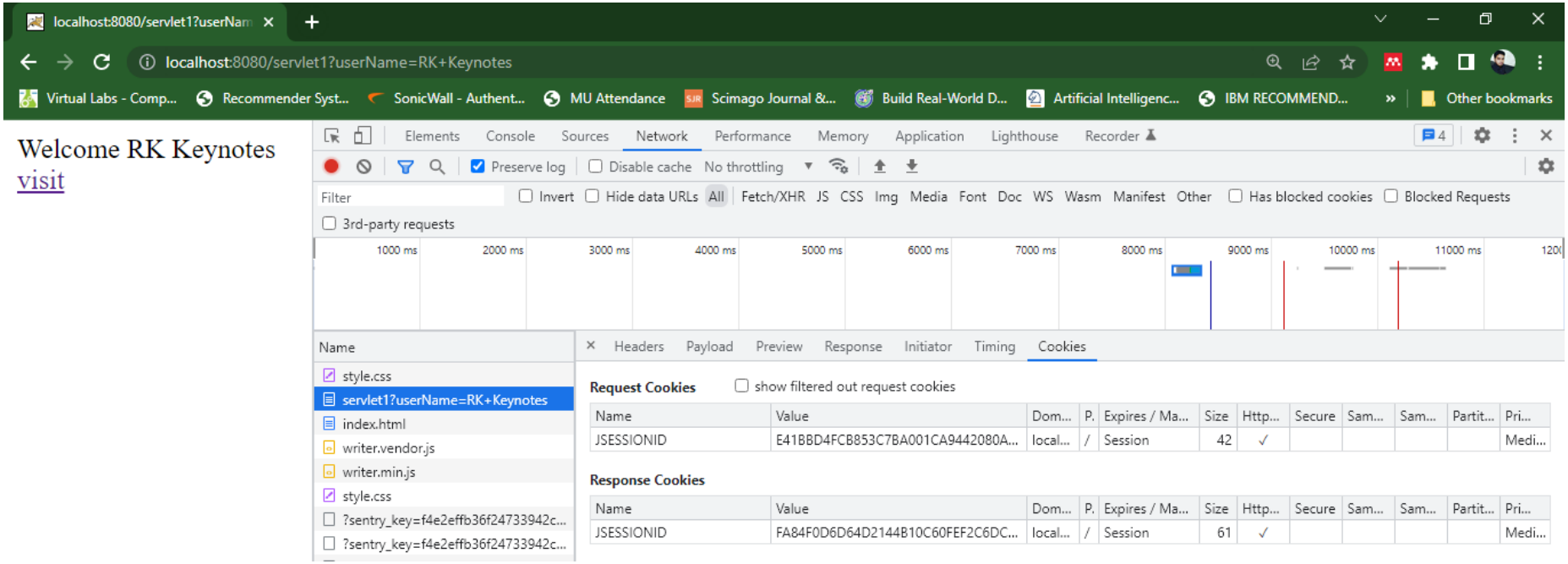
SecondServlet.java

```
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();

                    HttpSession session=request.getSession(false);
                    String n=(String)session.getAttribute("uname");
                    out.println("Hello "+n);

                    String n1=(String)session.getAttribute("uname");
                    out.println("Hi "+n1);
                    session.invalidate();
                    out.close();
          }catch(Exception e){System.out.println(e);}
           } }
```

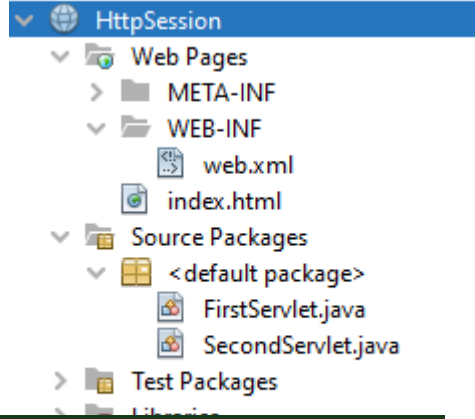
HttpSession Example

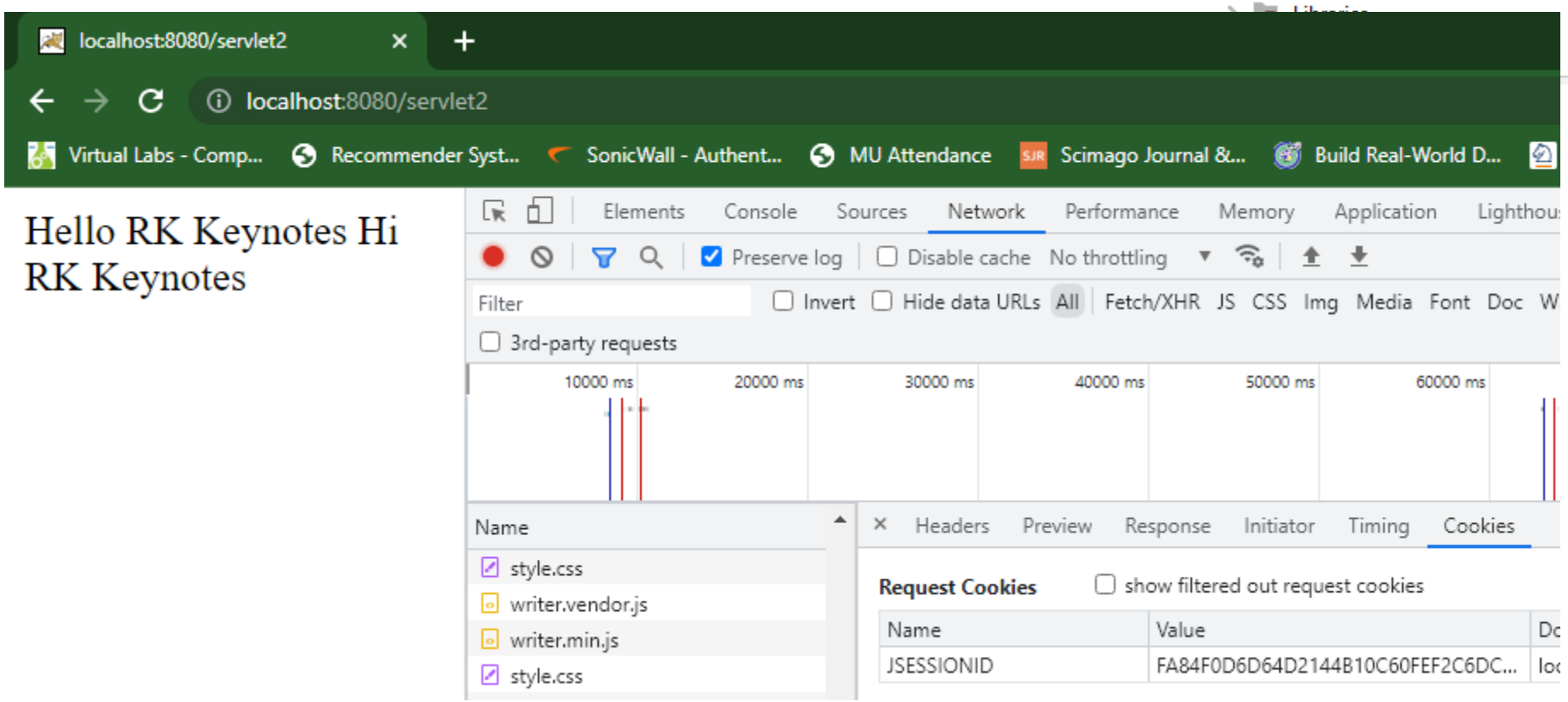






HttpSession Example





Refresh the page, the current session will get invalidated (means the current info removed from session)

Servlet Event

Events are basically occurrence of something. Changing the state of an object is known as an event.

We can perform some important tasks at the occurrence of these exceptions, such as counting total and current logged-in users, creating tables of the database at time of deploying the project, creating database connection object etc.

There are many Event classes and Listener interfaces in the **javax.servlet** and **javax.servlet.http** packages.

Types of Servlet Event: ContextLevel and SessionLevel.

Servlet Event

Servlet context-level (application-level) event:

It involves resources or state held at the extent of the application servlet context object.

Session-level event:

It involves resources or state related to the series of requests from one user session; that's, related to the HTTP session object.

Event classes & Event Interfaces

The event classes are as follows:

1. ServletRequestEvent
2. ServletContextEvent
3. ServletRequestAttributeEvent
4. ServletContextAttributeEvent
5. HttpSessionEvent
6. HttpSessionBindingEvent

The event interfaces are as follows:

1. ServletRequestListener
2. ServletRequestAttributeListener
3. ServletContextListener
4. ServletContextAttributeListener
5. HttpSessionListener
6. HttpSessionAttributeListener
7. HttpSessionBindingListener
8. HttpSessionActivationListener

HttpSessionEvent and HttpSessionListener

The HttpSessionEvent is notified when session object is changed. The corresponding Listener interface for this event is HttpSessionListener.

We can perform some operations at this event such as counting total and current logged-in users, maintaing a log of user details such as login time, logout time etc.

Methods of HttpSessionListener interface

1.public void sessionCreated(HttpSessionEvent e): is invoked when session object is created.

2.public void sessionDestroyed(HttpSessionEvent e): is invoked when session is invalidated.

ServletContextEvent

The ServletContextEvent is notified when web application is deployed on the server.

If you want to perform some action at the time of deploying the web application such as creating database connection, creating all the tables of the project etc, you need to implement ServletContextListener interface and provide the implementation of its methods.

Constructor of ServletContextEvent class
ServletContextEvent(ServletContext e)

Method of ServletContextEvent class
public ServletContext getServletContext(): returns the instance of ServletContext.

ServletContextEvent

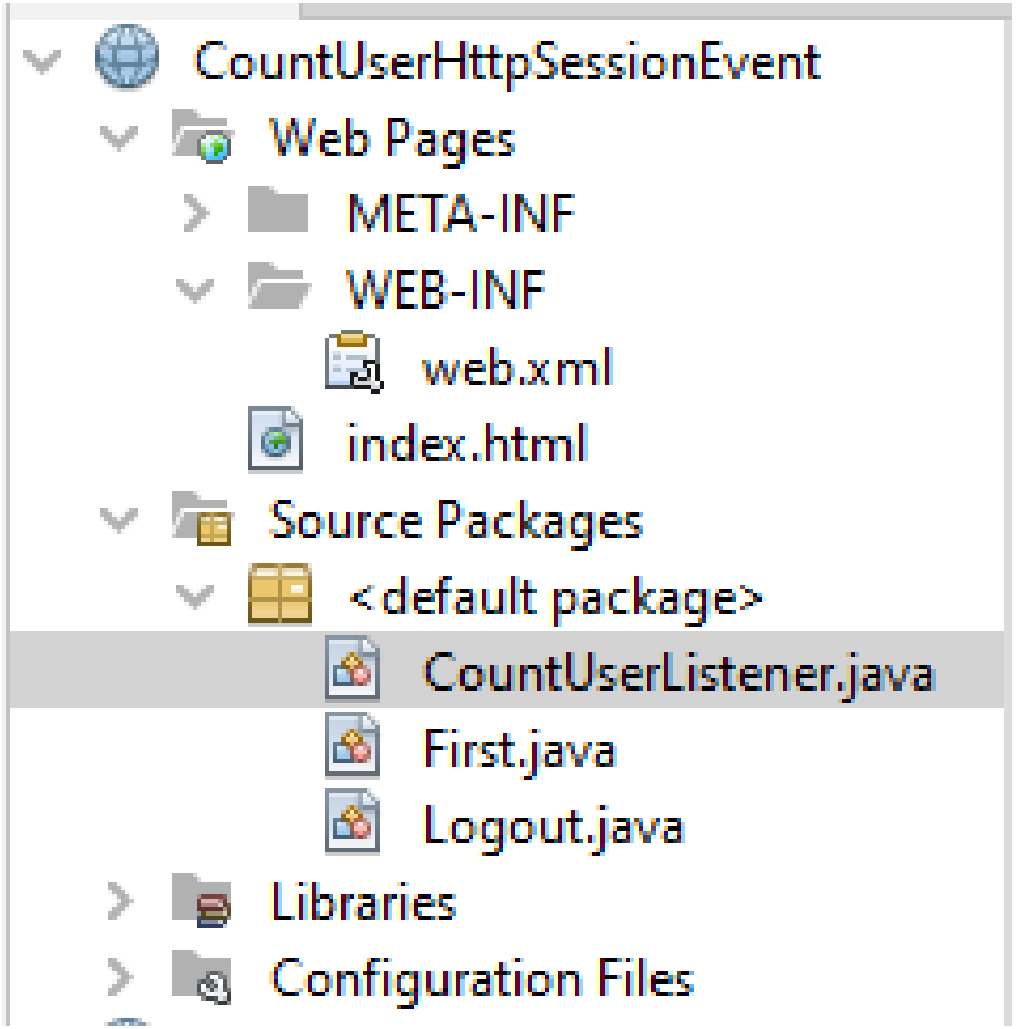
Methods of ServletContextListener interface

There are two methods declared in the ServletContextListener interface which must be implemented by the servlet programmer to perform some action such as creating database connection etc.

public void contextInitialized(ServletContextEvent e): is invoked when application is deployed on the server.

public void contextDestroyed(ServletContextEvent e): is invoked when application is undeployed from the server.

HttpSessionEvent & Listener Demo



HttpSessionEvent & Listener Demo

Marwadi education foundation

```
<servlet>
    <servlet-name>First</servlet-name>
    <servlet-class>First</servlet-class>
  </servlet>
  <servlet>
    <servlet-name>Logout</servlet-name>
    <servlet-class>Logout</servlet-class>
  </servlet>
  <servlet-mapping>
    <servlet-name>First</servlet-name>
    <url-pattern>/First</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
    <servlet-name>Logout</servlet-name>
    <url-pattern>/Logout</url-pattern>
  </servlet-mapping>
    <listener>
   <listener-class>CountUserListener</listener-class>
  </listener>
```

web.xml

HttpSessionEvent & Listener Demo

Marwadi education foundation

CountUserListener.java

```
import javax.servlet.ServletContext;
import javax.servlet.http.HttpSessionEvent;
import javax.servlet.http.HttpSessionListener;
public class CountUserListener implements HttpSessionListener{
   ServletContext ctx=null;
   static int total=0,current=0;

   public void sessionCreated(HttpSessionEvent e) {
   total++;
   current++;
   ctx=e.getSession().getServletContext();
   ctx.setAttribute("totalusers", total);
   ctx.setAttribute("currentusers", current);     }

   public void sessionDestroyed(HttpSessionEvent e) {
      current--;
      ctx.setAttribute("currentusers",current);
```

HttpSessionEvent & Listener Demo

Marwadi education foundation

First.java

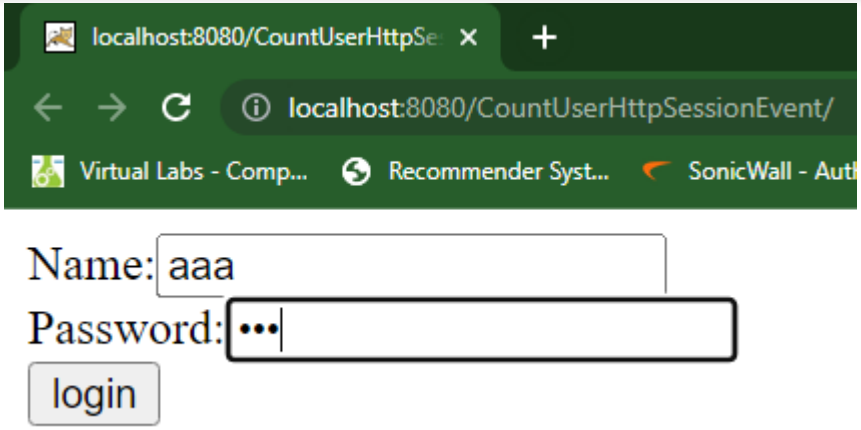
```
import java.io.IOException;  import java.io.PrintWriter;
import javax.servlet.*;  import javax.servlet.http.*;
public class First extends HttpServlet {
public void doGet(HttpServletRequest request,  HttpServletResponse response)
   throws ServletException, IOException {
      response.setContentType("text/html");
      PrintWriter out = response.getWriter();
      String n=request.getParameter("username");
      out.print("Welcome "+n);

      HttpSession session=request.getSession();
      session.setAttribute("uname",n);
      ServletContext ctx=getServletContext(); //retrieving data from ServletContext object
      int t=(Integer)ctx.getAttribute("totalusers");
      int c=(Integer)ctx.getAttribute("currentusers");
      out.print("<br>total users= "+t);
      out.print("<br>current users= "+c);
      out.print("<br><a href='Logout'>logout</a>");
      out.close();     } }
```

HttpSessionEvent & Listener Demo

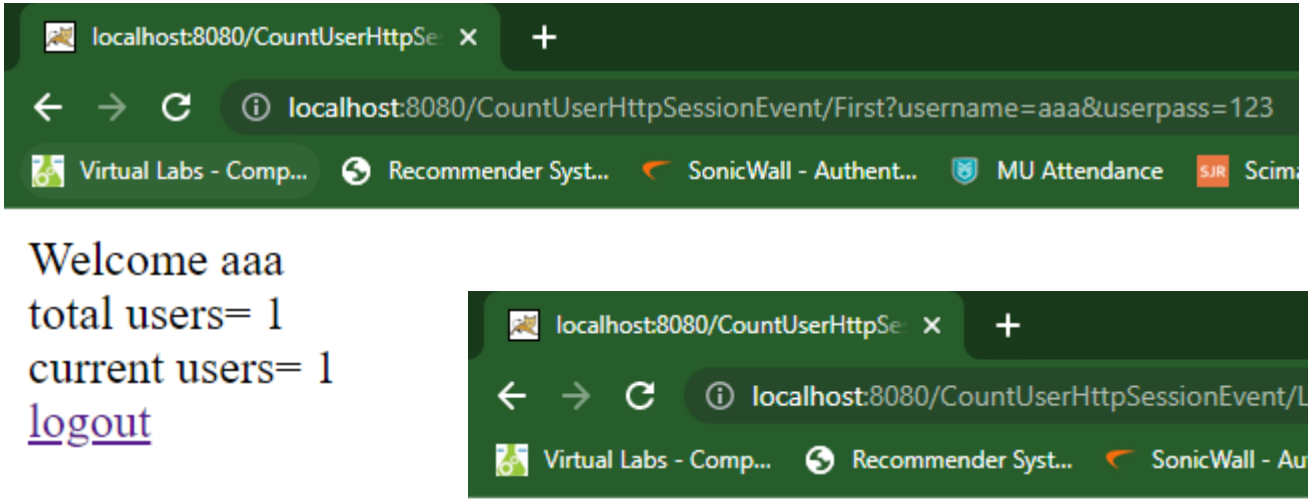
Logout.java

```
import java.io.IOException;  import java.io.PrintWriter;
import javax.servlet.*;  import javax.servlet.http.*;
public class Logout extends HttpServlet {
public void doGet(HttpServletRequest request,   HttpServletResponse response)
      throws ServletException, IOException {

      response.setContentType("text/html");
      PrintWriter out = response.getWriter();

      HttpSession session=request.getSession(false);
      session.invalidate();//invalidating session

      out.print("You are successfully logged out");

      out.close();
   }

}
```

HttpSessionEvent & Listener Demo



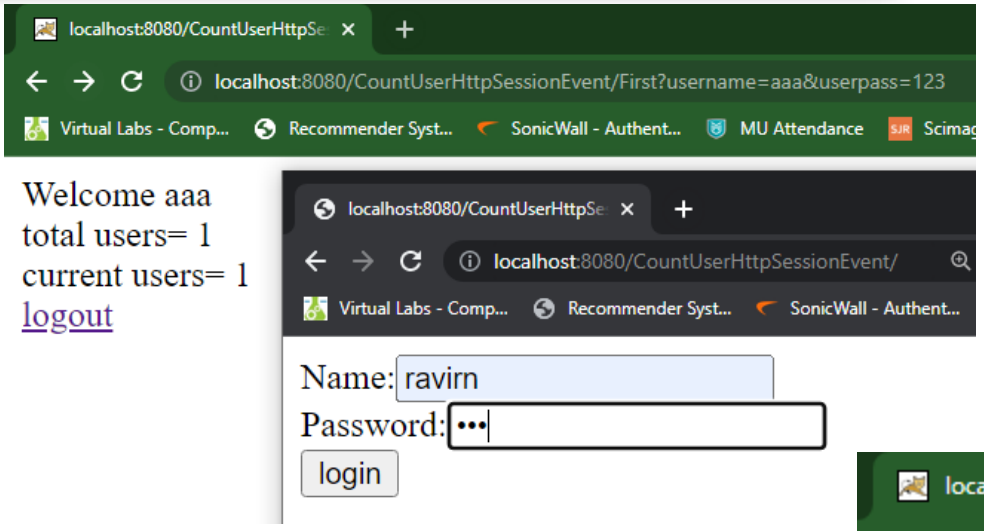
Output

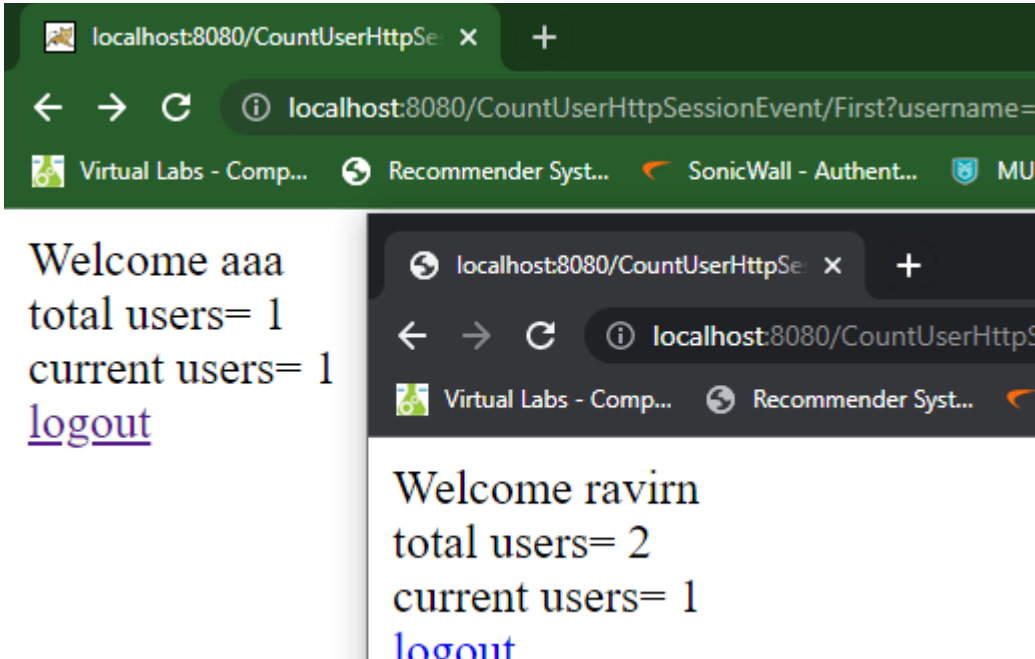


HttpSessionEvent & Listener Demo

Marwadi education foundation

Output





Activity

Mind Mapping

Mini Project

- Chat Application Demo.

- Create Login and Registration Form using HTTPServlet with validation and database connection.

Summary

- Servlet Introduction,
- Servlet Life Cycle(SLC),
- Types of Servlet,
- Servlet Configuration with Deployment Descriptor,
- Working with ServletContext and ServletConfig Object,
- Attributes in Servlet,
- Response and Redirection using Request Dispacher and using sendRedirect Method,
- Filter API,
- Manipulating Responses using Filter API,
- Session Tracking: using Cookies,
- HTTPSession,
- Hidden Form Fields and URL Rewriting,
- Types of Servlet Event: ContextLevel and SessionLevel

Up Next?

- Introduction to JSP
- Comparison with Servlet
- JSP Architecture
- JSP: Life Cycle
- Scripting Elements
- Directives
- Action Tags
- Implicit Objects
- Expression Language(EL)
- JSP Standard Tag Libraries(JSTL)
- Custom Tag
- Session Management
- Exception Handling
- CRUD Application

END OF UNIT - 3