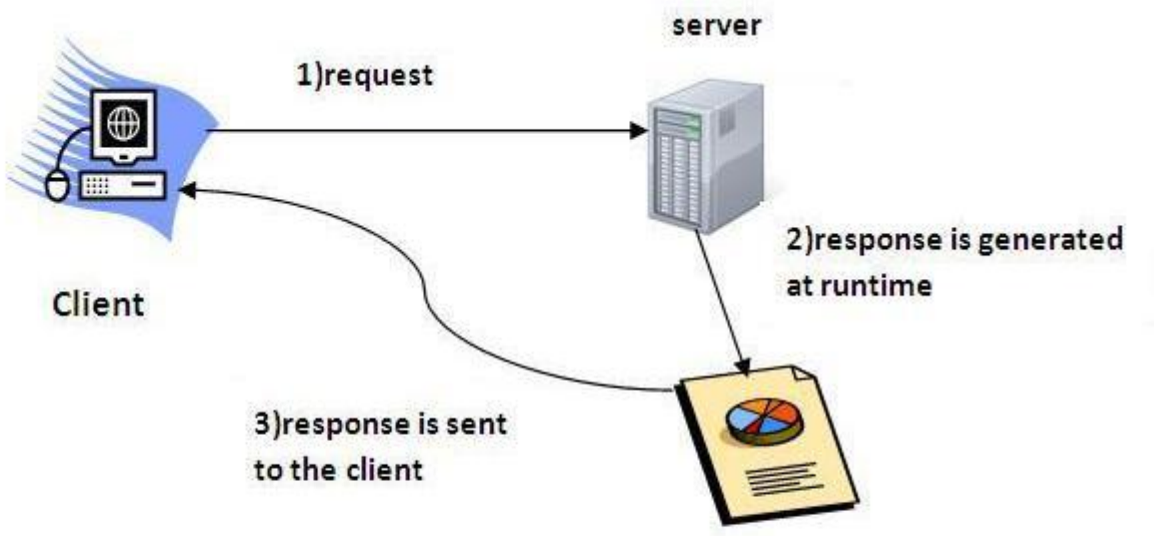
Servlet Basics

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

What is a 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 extend the capabilities of the servers and respond to the incoming request. It can respond to any type of requests.
- Servlet is a web component that is deployed on the server to create dynamic web page.


server
1)request
2)response is generated at runtime
Client
3)response is sent to the client

Difference between CGI and Servlet:

CGI (Common Gateway Interface)

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

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.

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.

Servlet API

- The **javax.servlet** and **javax.servlet.http** packages represent interfaces and classes for servlet API.
- The javax.servlet package contains many interfaces and classes that are used by the servlet or web container. These are not specific to any protocol.
- The javax.servlet.http package contains interfaces and classes that are responsible for http requests only.

Interfaces in javax.servlet package

There are many interfaces in javax.servlet package. Some of them are given below:

Servlet

ServletRequest

ServletResponse

RequestDispatcher

ServletConfig

ServletContext

Filter

FilterConfig

FilterChain

ServletRequestListener

ServletRequestAttributeListener

ServletContextListener

ServletContextAttributeListener

Classes in javax.servlet package

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

GenericServlet

ServletInputStream

ServletOutputStream

ServletRequestWrapper

ServletResponseWrapper

ServletRequestEvent

ServletContextEvent

ServletRequestAttributeEvent

ServletContextAttributeEvent

ServletException

UnavailableException

Interfaces in javax.servlet.http package

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

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

Classes in javax.servlet.http package

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

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

Servlet interface

Servlet interface provides common behaviour to all the servlets.

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

Methods of Servlet interface

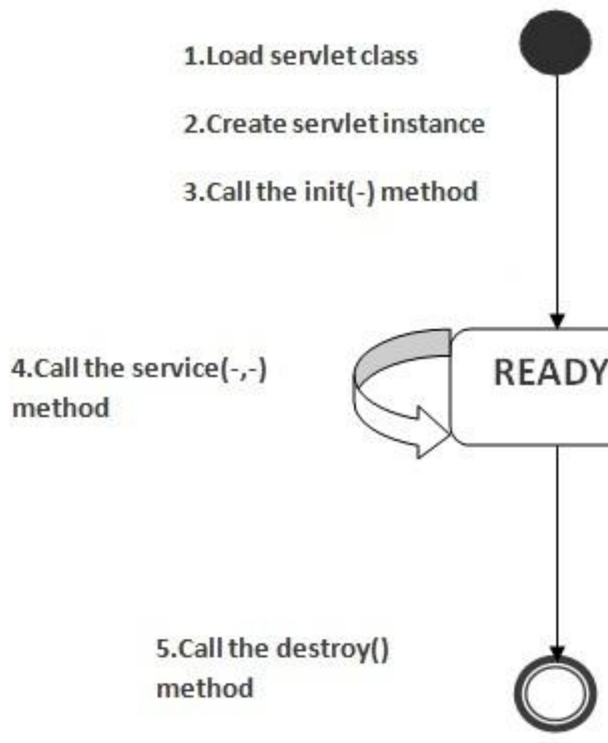
There are 5 methods in Servlet interface. The init, service and destroy are the life cycle methods of servlet. These are invoked by the web container.

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

Life Cycle of a Servlet (Servlet Life Cycle)

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

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


1.Load servlet class
2.Create servlet instance
3.Call the init(-) method
4.Call the service(-,-) method
READY
5.Call the destroy() method

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

1) Servlet class is loaded

The 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. Syntax of the init method is given below:

public void init(ServletConfig config) throws ServletException

4) service method is invoked

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

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

5) destroy method is invoked

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

```
public void destroy()
```

Steps to create a servlet example

There are given 6 steps to create a servlet example. These steps are required for all the servers.

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

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

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

1) Create a directory structures

The directory structure defines that where to put the different types of files so that web container may get the information and respond to the client.

The Sun Microsystem defines a unique standard to be followed by all the server vendors. Let's see the directory structure that must be followed to create the servlet.

```
└─ webapps
    └─ HelloServlet
        ├── WEB-INF
        │    ├── classes
        │    ├── src
        │    ├── lib
        │    └── web.xml
        ├── META-INF
        │    └── context.xml
        ├── index.html
        ├── *.html, *.jsp
        ├── css
        ├── images
        └── scripts
```

2) Create a Servlet

There are three ways to create the servlet.

1. By implementing the Servlet interface
2. By inheriting the GenericServlet class
3. By inheriting the HttpServlet class

Servlet Life Cycle Program:

MyServlet.java:

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

public class MyServlet 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.println(“Welcome to Servlet Program”);
        }
        public void destroy()
        {
                System.out.println("servlet is destroyed");
        }
        public ServletConfig getServletConfig()
        {
                return config;
        }
         public String getServletInfo()
        {
                return("verson 7");
        }
}
```

3) Compile the servlet

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

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

Two ways to load the jar file

1. set classpath
2. paste the jar file in JRE/lib/ext folder

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

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

The deployment descriptor is an xml file, from which Web Container gets the information about the servet to be invoked.

The web container uses the Parser to get the information from the web.xml file. There are many xml parsers such as SAX, DOM and Pull.

There are many elements in the web.xml file. Here is given some necessary elements to run the simple servlet program.

Description of the elements of web.xml file

There are too many elements in the web.xml file. Here is the illustration of some elements that is used in the above web.xml file. The elements are as follows:

<web-app> represents the whole application.

<servlet> is sub element of <web-app> and represents the servlet.

<servlet-name> is sub element of <servlet> represents the name of the servlet.

<servlet-class> is sub element of <servlet> represents the class of the servlet.

<servlet-mapping> is sub element of <web-app>. It is used to map the servlet.

<url-pattern> is sub element of <servlet-mapping>. This pattern is used at client side to invoke the servlet.

Example:

```
<web-app>
<servlet>
   <servlet-name>s1</servlet-name>
   <servlet-class>MyServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>s1</servlet-name>
    <url-pattern>/hello</url-pattern>
</servlet-mapping>
</web-app>
```

5) Start the Server and deploy the project

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

GenericServlet 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. Some of them are given below.

1. **public void init(ServletConfig config)** is used to initialize the servlet.
2. **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.
3. **public void destroy()** is invoked only once throughout the life cycle and indicates that servlet is being destroyed.

4. **public ServletConfig getServletConfig()** returns the object of ServletConfig.
5. **public String getServletInfo()** returns information about servlet such as writer, copyright, version etc.
6. **public void init()** it is a convenient method for the servlet programmers, now there is no need to call super.init(config)

Write a Simple GenericServlet Program to display simple message.

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

public class First extends GenericServlet
{
        public void service(ServletRequest req,ServletResponse res)
                throws IOException,ServletException
        {

                res.setContentType("text/html");

                PrintWriter out=res.getWriter();

                out.print("<b>hello generic servlet</b>");

        }
}
```

HttpServlet class

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

Methods of HttpServlet class

There are many methods in HttpServlet class. Some of them are given below:

1. **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.
2. **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.
3. **protected void doGet(HttpServletRequest req, HttpServletResponse res)** handles the GET request. It is invoked by the web container.
4. **protected void doPost(HttpServletRequest req, HttpServletResponse res)** handles the POST request. It is invoked by the web container.
5. **protected void doHead(HttpServletRequest req, HttpServletResponse res)** handles the HEAD request. It is invoked by the web container.

Write a Simple HttpServlet Program to display simple message.

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

public class Sample extends HttpServlet
{
        protected void service(HttpServletRequest req,HttpServletResponse res)
                throws IOException,ServletException
        {

                res.setContentType("text/html");

                PrintWriter out=res.getWriter();

                out.print("<b>hello HttpServlet</b>");

        }
}
```

Static Login Validation Program (username: admin & password: admin123) :

login.html:

```
<html>
  <body>
    <form action="sampleServlet">
    Enter your name:<input type="text" name="name"><br>
    Enter your password: <input type="password" name="pwd"><br>
     <input type="submit" value="Submit form "/>
    </form>
  </body>
</html>
```

SampleServlet.java:

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

public class SampleServlet extends HttpServlet{

   public void service(HttpServletRequest request, HttpServletResponse response)
   throws IOException
  {
     String username = request.getParameter("name");
```

```
      String password = request.getParameter("pwd");
      String uname="admin";
      String pass="admin123";
      PrintWriter out = response.getWriter();
      if(username.equals(uname) && password.equals(pass))
      {
          out.println("<html>");
          out.println("<body>");
          out.println("<h1>welcome to:"+username+"</h1>");
          out.println("</body>");
          out.println("</html>");
      }
      else
      {
       out.println("<h1>Invalid User</h1>");
      }
    }
}
```

Servlet Program to Register Student Data to Oracle Database through HTML form.

Register.html

```
<html>
  <head>
  </head>
  <body>
  <form name="myform" action="register">
  Enter your name: <input type="text" name="uname"><br>
  Enter your password: <input type="password" name="pwd"><br>
  Choose your Gender: <input type="radio" name="gender" value="Male">Male
                <input type="radio" name="gender" value="Female">Female<br>

  Enter your email: <input type="text" name="email"><br>
  <input type="submit"><input type="reset">
  </form>
  </body>
</html>
```

SRegister.java

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

public class SRegister extends HttpServlet
{
        protected void service(HttpServletRequest req,HttpServletResponse res) throws ServletException,
        IOException
        {
                PrintWriter pw=res.getWriter();
                res.setContentType("text/html");

                        String un = req.getParameter("uname");
                        String ps = req.getParameter("pwd");
                        String mail = req.getParameter("email");
                        String gen = req.getParameter("gender");

                try
                {
                Class.forName("oracle.jdbc.driver.OracleDriver");
                Connection con=DriverManager.getConnection
                                        ("jdbc:oracle:thin:@localhost:1521:XE","cse","cse");
               String qry = "insert into student values(?,?,?,?)";
                PreparedStatement pst=con.prepareStatement(qry);
                pst.setString(1,un);
               pst.setString(2,ps);
               pst.setString(3,mail);
               pst.setString(4,gen);
               int i = pst.executeUpdate();
               pw.println(i + "Row inserted successfully");
                pw.close();
                }
                catch (Exception e){
                        e.printStackTrace();
                }
        }
}
```

Servlet Program to Display Student Data from Oracle Database.

SDisplay.java

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

public class SDisplay extends HttpServlet
{
        protected void service(HttpServletRequest req,HttpServletResponse res) throws ServletException,
        IOException
        {
           res.setContentType("text/html");
           PrintWriter pw=res.getWriter();

          try
          {
                 Class.forName("oracle.jdbc.driver.OracleDriver");
                 Connection con=DriverManager.getConnection
                                ("jdbc:oracle:thin:@localhost:1521:XE","cse","cse");
                 Statement st=con.createStatement();
                 System.out.println("connection established successfully...!!");
                String qry = "select *from student";
                Statement stmt = con.createStatement();
                 ResultSet rs=stmt.executeQuery(qry);
                 pw.println("<table border=1>");
                pw.println("<tr>");
                pw.println("<th>Uname</th>");
                pw.println("<th>Pwd</th>");
                 pw.println("<th>Email</th>");
                pw.println("<th>Gender</th>");
                pw.println("</tr>");
                 while(rs.next())
                 {
                   pw.println("<tr>");
                         pw.println("<td>"+rs.getString(1)+"</td>");
                         pw.println("<td>"+rs.getString(2)+"</td>");
                         pw.println("<td>"+rs.getString(3)+"</td>");
                         pw.println("<td>"+rs.getString(4)+"</td>");
                         pw.println("</tr>");
                 }
                 pw.println ("</table>");
                 pw.close();
             }
            catch (Exception e)
                 {
                e.printStackTrace();
            }
        }
}
```

Dynamic Login Validation from oracle database registered users:

Login.html

```
<html>
  <body>
    <form action="ServletLogin" method="POST">
    Enter your name:<input type="text" name="name"><br>
    Enter your password: <input type="password" name="pwd"><br>
     <input type="submit" value="Submit form "/>
    </form>
  </body>
</html>
```

ServletLogin.java

```
import java.io.*;
import java.sql.*;
import javax.servlet.*;
import javax.servlet.http.*;
public class ServletLogin extends HttpServlet
{
        protected void doPost(HttpServletRequest req,HttpServletResponse res)throws ServletException,
        IOException
        {
                PrintWriter pw=res.getWriter();
                res.setContentType("text/html");
                String uname=req.getParameter("name");
               String pass=req.getParameter("pwd");
                int flag=0;
                try
                {
                        Class.forName("oracle.jdbc.driver.OracleDriver");
                        Connection con=DriverManager.getConnection
                                ("jdbc:oracle:thin:@localhost:1521:XE","cse","cse");
                        Statement st=con.createStatement();
                        ResultSet rs=st.executeQuery("Select * from student");
                        while(rs.next())
                        {
                                String un = rs.getString(1);
                                String pwd= rs.getString(2);
                             if(un.equals(uname) && pwd.equals(pass))
                             {
                                        flag=1;
                                        break;
                              }

                        }
                        if(flag==0)
                        {
                                    pw.println("Invalid user.. Pleae login again");
                        }
                        else
                        {
                                pw.println("Welcome to user:" + uname);
                        }
                        pw.close();
                }
                catch (Exception e){
                        e.printStackTrace();
                }
        }
}
```

Difference between GET and POST

Two commonly used methods for a request-response between a client and server are: GET and POST.

- **GET** - Requests data from a specified resource
- **POST** - Submits data to be processed to a specified resource

GET	POST
In case of Get request, only **limited amount of data** can be sent because data is sent in header/URL. (Maximum URL length is 2048 characters).	In case of post request, **large amount of data** can be sent because data is sent in body. No restrictions
Get request is **not secured** because data is exposed in URL bar.	Post request is **secured** because data is not exposed in URL bar.
Get request **can be bookmarked.**	Post request **cannot be bookmarked.**
Get request is **idempotent.** It means second request will be ignored until response of first request is delivered	Post request is **non-idempotent.**
Get request is **more efficient** and used more than Post.	Post request is **less efficient** and used less than get.
Only **ASCII characters** allowed	No restrictions. **Binary data** is also allowed
Parameters **remain in browser history**	Parameters are **not saved in browser history**

Example Program:

Dynamic Login Validation from oracle database registered users program (above program)

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.

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.

How to get the object of ServletConfig

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

Syntax of getServletConfig() method

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

Example of getServletConfig() method

```
ServletConfig config=getServletConfig();
```

Example Program:

web.xml:

```
<web-app>

<servlet>
        <servlet-name>myservlet</servlet-name>
        <servlet-class>ServletEx</servlet-class>
        <init-param>
              <param-name>username</param-name>
              <param-value>system</param-value>
        </init-param>
        <init-param>
              <param-name>password</param-name>
              <param-value>oracle</param-value>
        </init-param>
</servlet>

<servlet-mapping>
        <servlet-name>myservlet</servlet-name>
        <url-pattern>/servlet1</url-pattern>
</servlet-mapping>

</web-app>
```

ServletEx.java:

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

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

                ServletConfig config=getServletConfig();
```

```
                    String un = config.getInitParameter(“username”);
                    String ps = config.getInitParameter(“password”);
                    out.println(“the username is” + un);
                    out.println(“the password is” + ps);

   }

   out.close();
}

}
```

RequestDispatcher in Servlet

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.

Methods of RequestDispatcher interface

The RequestDispatcher interface provides two methods. They are:

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.

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.

How to get the object of RequestDispatcher

The getRequestDispatcher() method of ServletRequest interface returns the object of RequestDispatcher.

Syntax of getRequestDispatcher method

public RequestDispatcher getRequestDispatcher(String resource);

Example of using getRequestDispatcher method

```
RequestDispatcher rd=request.getRequestDispatcher("myservlet");

//myservlet is the url-pattern of the  servlet

rd.forward(request, response);//method may be include or forward
```

Example of RequestDispatcher interface

In this example, we are validating the password entered by the user. If password is servlet, it will forward the request to the WelcomeServlet, otherwise will show an error message: sorry username or password error! In this program, we are checking for hardcoded information. But you can check it to the database also that we will see in the development chapter. 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.

index.html

```
<form action="servlet1" method="post">

Name:<input type="text" name="uname"/><br/>

Password:<input type="password" name="pwd"/><br/>

<input type="submit" value="login"/>

</form>
```

Login.java

```
import java.io.*;

import javax.servlet.*;

import javax.servlet.http.*;

public class Login extends HttpServlet

{

        public void doPost(HttpServletRequest request, HttpServletResponse response)

                        throws ServletException, IOException

       {

                response.setContentType("text/html");

                PrintWriter out = response.getWriter();

                String n=request.getParameter("uname");

                String p=request.getParameter("pwd");

                if(n.equals(“admin”) && p.equals("admin"){

                RequestDispatcher rd=request.getRequestDispatcher("servlet2");

                rd.forward(request, response);

        }

        Else
```

```
          {
                    out.print("Sorry UserName or Password Error!");
                     RequestDispatcher rd=request.getRequestDispatcher("/index.html");
                    rd.include(request, response);


          }
   }


}
```

WelcomeServlet.java

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

web.xml

```
<web-app>
 <servlet>
   <servlet-name>Login</servlet-name>
   <servlet-class>Login</servlet-class>
  </servlet>
  <servlet>
```

```
   <servlet-name>WelcomeServlet</servlet-name>
   <servlet-class>WelcomeServlet</servlet-class>
  </servlet>
  <servlet-mapping>
   <servlet-name>Login</servlet-name>
   <url-pattern>/servlet1</url-pattern>
  </servlet-mapping>
  <servlet-mapping>
   <servlet-name>WelcomeServlet</servlet-name>
   <url-pattern>/servlet2</url-pattern>
  </servlet-mapping>
  <welcome-file-list>
   <welcome-file>index.html</welcome-file>
  </welcome-file-list>
</web-app>
```

HttpSession interface

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.

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.

How to get the HttpSession object ?

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.

Example:

index.html

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<h1>Login App using HttpSession</h1>
<a href="login.html">Login</a>
<a href="LogoutServlet">Logout</a>
<a href="ProfileServlet">Profile</a>
</body>
</html>
```

login.html

```
<form action="LoginServlet" method="get">

Name:<input type="text" name="name"><br>

Password:<input type="password" name="password"><br>

<input type="submit" value="submit">

</form>
```

link.html

```
<a href="login.html">Login</a>

<a href="LogoutServlet">Logout</a>

<a href="ProfileServlet">Profile</a>

<hr/>
```

LoginServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class LoginServlet extends HttpServlet
{
        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException
        {
                response.setContentType("text/html");
                PrintWriter out=response.getWriter();
```

```
		RequestDispatcher rd=request.getRequestDispatcher("link.html");
		rd.include(request, response);

		String name=request.getParameter("name");
		String password=request.getParameter("password");

		if(name.equals("admin") && password.equals("admin123"))
		{
			out.print("Welcome, "+name);
			HttpSession session=request.getSession();
			session.setAttribute("n",name);
		}
		else
		{
			out.print("Sorry, username or password error!");
			RequestDispatcher rd1=request.getRequestDispatcher("login.html");
			rd1.include(request, response);
		}
		out.close();
	}

}
```

ProfileServlet.java

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

public class ProfileServlet extends HttpServlet
{
	protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
	ServletException, IOException
	{
		response.setContentType("text/html");
		PrintWriter out=response.getWriter();
		RequestDispatcher rd=request.getRequestDispatcher("link.html");
		rd.include(request, response);

		HttpSession session=request.getSession(false);
		if(session!=null)
		{
			String name=(String)session.getAttribute("n");
			out.print("Hello, "+name+" Welcome to Profile");
		}
		else
		{
			    out.print("Please login first");
			RequestDispatcher rd1=request.getRequestDispatcher("login.html");
			rd1.include(request, response);
```

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

LogoutServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class LogoutServlet extends HttpServlet
{
        protected void doGet(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException
        {
                response.setContentType("text/html");
                PrintWriter out=response.getWriter();

                RequestDispatcher rd=request.getRequestDispatcher("link.html");
                rd.include(request, response);

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

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

                out.close();
        }

}
```

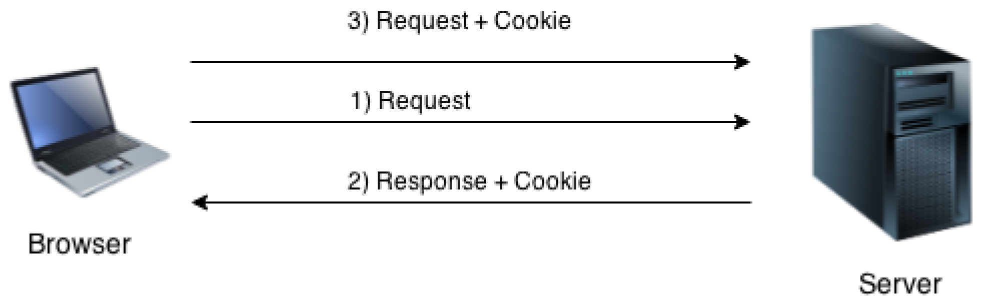
Cookies in Servlet

A **cookie** is a small piece of information that is persisted between the multiple client requests.

A cookie has a name, a single value, and optional attributes such as a comment, path and domain qualifiers, a maximum age, and a version number.

How Cookie works

By default, each request is considered as a new request. In cookies technique, we add cookie with response from the servlet. So cookie is stored in the cache of the browser. After that if request is sent by the user, cookie is added with request by default. Thus, we recognize the user as the old user.



Types of Cookie

There are 2 types of cookies in servlets.

1. Non-persistent cookie
2. Persistent cookie

Non-persistent cookie

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

Persistent cookie

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

Advantage of Cookies

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

Disadvantage of Cookies

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

Cookie class

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

Constructor of Cookie class

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

Useful Methods of Cookie class

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

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

Other methods required for using Cookies

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

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

How to create Cookie?

Let's see the simple code to create cookie.

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

How to delete Cookie?

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

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

How to get Cookies?

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

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

Example:

index.html

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
<h1>Login App using HttpSession</h1>
<a href="login.html">Login</a>
<a href="LogoutServlet">Logout</a>
```

```
<a href="ProfileServlet">Profile</a>
</body>
</html>
```

login.html

```
<form action="LoginServlet" method="POST">

Name:<input type="text" name="name"><br>

Password:<input type="password" name="password"><br>

<input type="submit" value="submit">

</form>
```

link.html

```
<a href="login.html">Login</a>

<a href="LogoutServlet">Logout</a>

<a href="ProfileServlet">Profile</a>

<hr/>
```

LoginServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class LoginServlet extends HttpServlet
{
        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException
        {
                response.setContentType("text/html");
                PrintWriter out=response.getWriter();
                RequestDispatcher rd=request.getRequestDispatcher("link.html");
                rd.include(request, response);

                String name=request.getParameter("name");
                String password=request.getParameter("password");

                if(name.equals("admin") && password.equals("admin123"))
                {
                        out.print("Welcome, "+name);
                        Cookie ck=new Cookie("n",name);
                        response.addCookie(ck);
                }
                else
                {
                        out.print("Sorry, username or password error!");
                        RequestDispatcher rd1=request.getRequestDispatcher("login.html");
                        rd1.include(request, response);
                }
                out.close();
        }

}
```

ProfileServlet.java

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

public class ProfileServlet extends HttpServlet
{
        protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException
        {
                response.setContentType("text/html");
                PrintWriter out=response.getWriter();
                RequestDispatcher rd=request.getRequestDispatcher("link.html");
                rd.include(request, response);

                 Cookie ck[]=request.getCookies();
                if(ck!=null)
                {
                        String name=ck[0].getValue();
                        if(!name.equals("")||name!=null)
                        {
                           out.print("<b>Welcome to Profile</b>");
                           out.print("<br>Welcome, "+name);
                        }
                }
                else
                {
                            out.print("Please login first");
                        RequestDispatcher rd1=request.getRequestDispatcher("login.html");
                        rd1.include(request, response);
                }
                out.close();
        }
}
```

LogoutServlet.java

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
public class LogoutServlet extends HttpServlet
{
	protected void doPost(HttpServletRequest request, HttpServletResponse response) throws
	ServletException, IOException
	{
		response.setContentType("text/html");
		PrintWriter out=response.getWriter();

		RequestDispatcher rd=request.getRequestDispatcher("link.html");
		rd.include(request, response);

		 Cookie ck=new Cookie("n","");
		ck.setMaxAge(0);
		response.addCookie(ck);

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

		out.close();
	}

}
```