Servlets

Servlet is a simple java program that runs on server and capable of handling request and generating dynamic response.

Servlet used to create a web application. There are many interfaces and classes in the Servlet API such as **Servlet**, **GenericServlet**, **HttpServlet**, **ServletRequest**, **ServletResponse**, etc.

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

Client

Chrome, Firefox

DB

Request

Server

Response

* Servlet is a technology which is used to create a web application.
* Servlet is an API that provides many interfaces and classes including documentation.
* 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 requests.
* Servlet is a web component that is deployed on the server to create a dynamic web page.

# Create Servlet

There are 3 ways we can create servlet

1. Using Servlet Interface
2. Using GenericServlet Class
3. Using HttpServlet Class

# Creating Servlet using Servlet interface

**Life Cycle Method**:

1 – public void init(ServletConfig config);

2 – public void service(ServletRequest request,

ServletResponse response);

3 – public void destroy();

**Non Life Cycle Method**:

4 – public ServletConfig getServletConfig();

5 – public String getServletInfo();

1 – javax.servlet.Servlet

2 – must be override all the methods

3 – 5 methods in Servlet interface

**Deploy in server**

web.xml – Deployment Descriptor

url mapping -

# FirstProgram.java

import javax.servlet.Servlet

public class FirstProgram implements Servlet

{

ServletConfig con;  
public void destroy()

{

System.out.println(“destroy method call”);

}

Public void init(ServletConfig config) throws ServletException

{

System.out.println(“init method call”);

Con=config;

}

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

{

System.out.println(“service method call”);

res.setContentType(“text/html”);

PrinterWriter out = res.getWriter();

out.println(“Welcome to Servlet”);

out.println(“Today : “ + new Date().toString());

System.out.println(getServletConfig());

System.out.println(getServletInfo());

}

public ServletConfig getServletConfig()

{  
return con;

}

Public String getServletInfo()

{

Return “Hey Sreedhar this is My First Program”;

}

}

# Index.html

<html>

<body>

<h1> Home Page </h1>

<a href=”app”>Click Here to go to Servlet </a>

</body>

</html>

# Web.xml

<servlet>

<servlet-name>first </servlet-name>

<servlet-class>com.sreedhar.FirstProgram</servlet-class>

</servlet>

<servlet-mapping>

<servlet-name> first </servlet-name>

<url-pattern>/app</url-pattern>

</servlet-mapping>

# 2. Using GenericServlet Class

# SecondProgram.java

package com.sreedhar

public class SecondProgram extends GenericServlet

# 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** 🡪 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

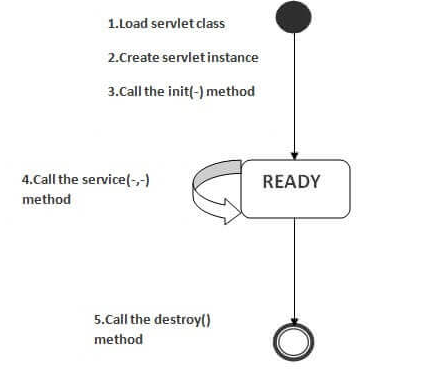
1. **service method is invoked** 🡪 The web container calls the service method each time when request for the servlet is received. If servlet is not initialized, it follows the first three steps as described above then calls the service method. If servlet is initialized, it calls the service method. Notice 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

1. **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()



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.

# Attributes in 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.

### **Attribute specific methods of ServletRequest, HttpSession and ServletContext interface**

There are following 4 attribute specific methods. They are as follows:

1. **public void setAttribute (String name, Object object)**

sets the given object in the application scope.

1. **public Object getAttribute (String name)**

Returns the attribute for the specified name.

1. **public Enumeration getInitParameterNames ()**

Returns the names of the context's initialization parameters as an Enumeration of String objects.

1. **public void removeAttribute (String name)**

Removes the attribute with the given name from the servlet context.

# Interaction between Client & Servlet

# Servlet demo Application development with Sessions

**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. It is shown in the figure given below:

### **Session Tracking Techniques**

There are four techniques used in Session tracking:

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

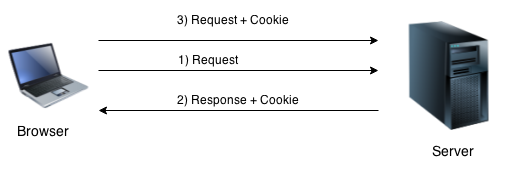
# Cookies in Servlet

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

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

### **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.



There are 2 types of cookies in servlets.

### **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.

Note: Gmail uses cookie technique for login. If you disable the cookie, gmail won't work.

### **Constructor of Cookie class**

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

### **Methods of Cookie class**

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

|  |  |
| --- | --- |
| **Method** | **Des-cription** |
| 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.

1. Cookie ck=**new** Cookie("username","nagalakshmi");//creating cookie object
2. 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.

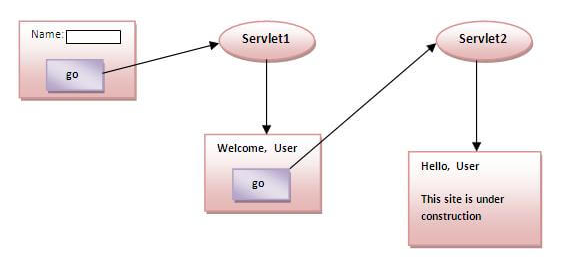
1. Cookie ck=**new** Cookie("user","");//deleting value of cookie
2. ck.setMaxAge(0);//changing the maximum age to 0 seconds
3. response.addCookie(ck);//adding cookie in the response

### **How to get Cookies?**

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

1. Cookie ck[]=request.getCookies();
2. **for**(**int** i=0;i<ck.length;i++){
3. out.print("<br>"+ck[i].getName()+" "+ck[i].getValue());//printing name and value of cookie
4. }

### **Simple example of Servlet Cookies**



### **index.html**

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

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

<input type="submit" value="go"/>

</form>

### **FirstServlet.java**

**import** java.io.\*;

**import** javax.servlet.\*;

**import** javax.servlet.http.\*;

**public** **class** FirstServlet **extends** HttpServlet {

**public** **void** doPost(HttpServletRequest request, HttpServletResponse response){

**try**{

    response.setContentType("text/html");

    PrintWriter out = response.getWriter();

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

    out.print("Welcome "+n);

    Cookie ck=**new** Cookie("uname",n);//creating cookie object

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

    //creating submit button

    out.print("<form action='servlet2'>");

    out.print("<input type='submit' value='go'>");

    out.print("</form>");

    out.close();

        }**catch**(Exception e){System.out.println(e);}

  }

}

### **SecondServlet.java**

**import** java.io.\*;

**import** javax.servlet.\*;

**import** javax.servlet.http.\*;

**public** **class** SecondServlet **extends** HttpServlet {

**public** **void** 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);}

    }

}

### **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>

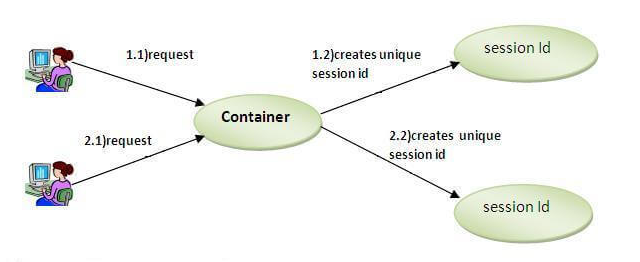
# HttpSession interface

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.



### **How to get the HttpSession object ?**

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

1. **public HttpSession getSession():**Returns the current session associated with this request, or if the request does not have a session, creates one.
2. **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**

* 1. **public String getId()**
  2. **public long getCreationTime()**
  3. **public long getLastAccessedTime()**
  4. **public void invalidate()**

### **Example of using HttpSession**

### **index.html**

<form action="servlet1">

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

<input type="submit" value="go"/>

</form>

### **FirstServlet.java**

**import** java.io.\*;

**import** javax.servlet.\*;

**import** javax.servlet.http.\*;

**public** **class** FirstServlet **extends** HttpServlet {

**public** **void** 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("<a href='servlet2'>visit</a>");

        out.close();

                }**catch**(Exception e){System.out.println(e);}

    }

}

### **SecondServlet.java**

**import** java.io.\*;

**import** javax.servlet.\*;

**import** javax.servlet.http.\*;

**public** **class** SecondServlet **extends** HttpServlet {

**public** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**try**{

        response.setContentType("text/html");

        PrintWriter out = response.getWriter();

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

        String n=(String)session.getAttribute("uname");

        out.print("Hello "+n);

        out.close();

                }**catch**(Exception e){System.out.println(e);}

    }

}

### **web.xml**

<web-app>

<servlet>

<servlet-name>s1</servlet-name>

<servlet-**class**>FirstServlet</servlet-**class**>

</servlet>

<servlet-mapping>

<servlet-name>s1</servlet-name>

<url-pattern>/servlet1</url-pattern>

</servlet-mapping>

<servlet>

<servlet-name>s2</servlet-name>

<servlet-**class**>SecondServlet</servlet-**class**>

</servlet>

<servlet-mapping>

<servlet-name>s2</servlet-name>

<url-pattern>/servlet2</url-pattern>

</servlet-mapping>

</web-app>

RequestDispatcher in Servlet

1. [RequestDispatcher Interface](https://www.javatpoint.com/requestdispatcher-in-servlet)
2. [Methods of RequestDispatcher interface](https://www.javatpoint.com/requestdispatcher-in-servlet#rdmethod)
   1. [forward method](https://www.javatpoint.com/requestdispatcher-in-servlet#rdforward)
   2. [include method](https://www.javatpoint.com/requestdispatcher-in-servlet#rdinclude)
3. [How to get the object of RequestDispatcher](https://www.javatpoint.com/requestdispatcher-in-servlet#rdhow)
4. [Example of RequestDispatcher interface](https://www.javatpoint.com/requestdispatcher-in-servlet#rdex)

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:

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.



68.2K

Best productivity apps to make life easier

As you see in the above figure, response of second servlet is sent to the client. Response of the first servlet is not displayed to the user.



|  |
| --- |
| As you can see in the above figure, response of second servlet is included in the response of the first servlet that is being sent to the client. |

How to get the object of RequestDispatcher

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

Syntax of getRequestDispatcher method

1. **public** RequestDispatcher getRequestDispatcher(String resource);

Example of using getRequestDispatcher method

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

//servlet2 is the url-pattern of the second 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 cheking for hardcoded information. 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="userName"/><br/>

Password:<input type="password" name="userPass"/><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("userName");

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

**if**(p.equals("servlet"){

        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("userName");

    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>

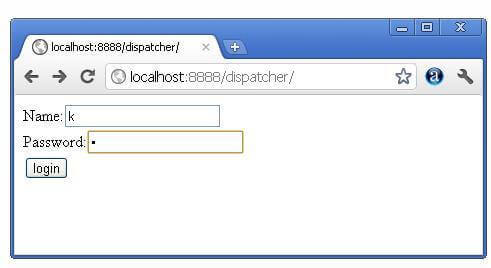
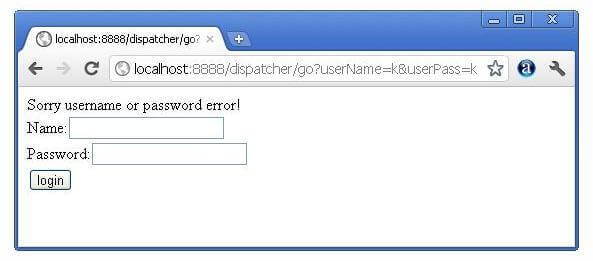
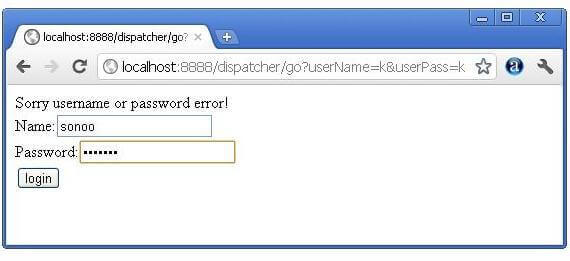
  </servlet-mapping>

  <welcome-file-list>

   <welcome-file>index.html</welcome-file>

  </welcome-file-list>

</web-app>

Servlet HttpSession Login and Logout Example

We can bind the objects on HttpSession instance and get the objects by using setAttribute and getAttribute methods.

In the previous page, we have learnt about what is HttpSession, How to store and get data from session object etc.

Here, we are going to create a real world login and logout application without using database code. We are assuming that password is admin123.

3M

Xbox Series X Debuting in November, 'Halo Infinite' Delayed to 2021

In this example, we are creating 3 links: login, logout and profile. User can't go to profile page until he/she is logged in. If user is logged out, he need to login again to visit profile.

In this application, we have created following files.

1. index.html
2. link.html
3. login.html
4. LoginServlet.java
5. LogoutServlet.java
6. ProfileServlet.java
7. web.xml

*File: index.html*

<!DOCTYPE html**>**

**<html>**

**<head>**

**<meta** charset="ISO-8859-1"**>**

**<title>**Servlet Login Example**</title>**

**</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>**

*File: link.html*

**<a** href="login.html"**>**Login**</a>** |

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

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

**<hr>**

*File: 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="login"**>**

**</form>**

*File: LoginServlet.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;

**import** javax.servlet.http.HttpSession;

**public** **class** LoginServlet **extends** HttpServlet {

**protected** **void** doPost(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

        response.setContentType("text/html");

        PrintWriter out=response.getWriter();

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

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

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

**if**(password.equals("admin123")){

        out.print("Welcome, "+name);

        HttpSession session=request.getSession();

        session.setAttribute("name",name);

        }

**else**{

            out.print("Sorry, username or password error!");

            request.getRequestDispatcher("login.html").include(request, response);

        }

        out.close();

    }

}

*File: LogoutServlet.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;

**import** javax.servlet.http.HttpSession;

**public** **class** LogoutServlet **extends** HttpServlet {

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

            response.setContentType("text/html");

            PrintWriter out=response.getWriter();

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

            HttpSession session=request.getSession();

            session.invalidate();

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

            out.close();

    }

}

*File: ProfileServlet.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;

**import** javax.servlet.http.HttpSession;

**public** **class** ProfileServlet **extends** HttpServlet {

**protected** **void** doGet(HttpServletRequest request, HttpServletResponse response)

**throws** ServletException, IOException {

        response.setContentType("text/html");

        PrintWriter out=response.getWriter();

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

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

**if**(session!=**null**){

        String name=(String)session.getAttribute("name");

        out.print("Hello, "+name+" Welcome to Profile");

        }

**else**{

            out.print("Please login first");

out.print(“ You can’t visit profile page directly”);

            request.getRequestDispatcher("login.html").include(request, response);

        }

        out.close();

    }

}

*File: web.xml*

**<?xml** version="1.0" encoding="UTF-8"**?>**

**<web-app** xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xmlns="http://java.sun.com/xml/ns/javaee" xsi:schemaLocation="http://java.sun.com/xml/ns/javaee

http://java.sun.com/xml/ns/javaee/web-app\_2\_5.xsd" id="WebApp\_ID" version="2.5"**>**

**<servlet>**

**<description></description>**

**<display-name>**LoginServlet**</display-name>**

**<servlet-name>**LoginServlet**</servlet-name>**

**<servlet-class>**LoginServlet**</servlet-class>**

**</servlet>**

**<servlet-mapping>**

**<servlet-name>**LoginServlet**</servlet-name>**

**<url-pattern>**/LoginServlet**</url-pattern>**

**</servlet-mapping>**

**<servlet>**

**<description></description>**

**<display-name>**ProfileServlet**</display-name>**

**<servlet-name>**ProfileServlet**</servlet-name>**

**<servlet-class>**ProfileServlet**</servlet-class>**

**</servlet>**

**<servlet-mapping>**

**<servlet-name>**ProfileServlet**</servlet-name>**

**<url-pattern>**/ProfileServlet**</url-pattern>**

**</servlet-mapping>**

**<servlet>**

**<description></description>**

**<display-name>**LogoutServlet**</display-name>**

**<servlet-name>**LogoutServlet**</servlet-name>**

**<servlet-class>**LogoutServlet**</servlet-class>**

**</servlet>**

**<servlet-mapping>**

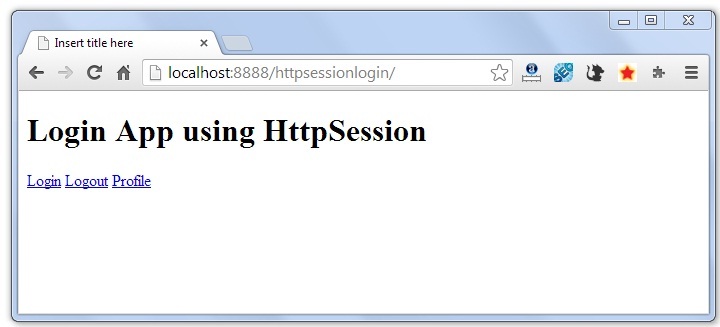
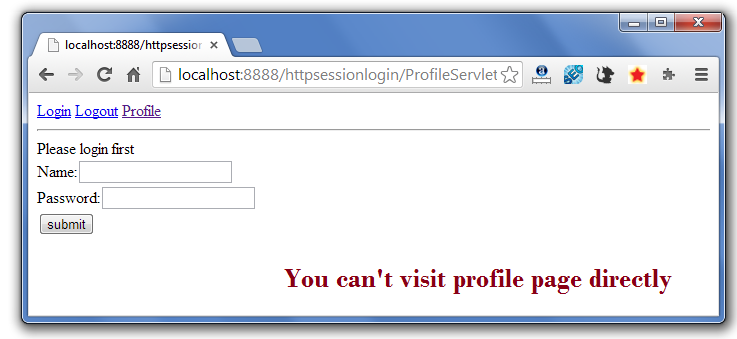
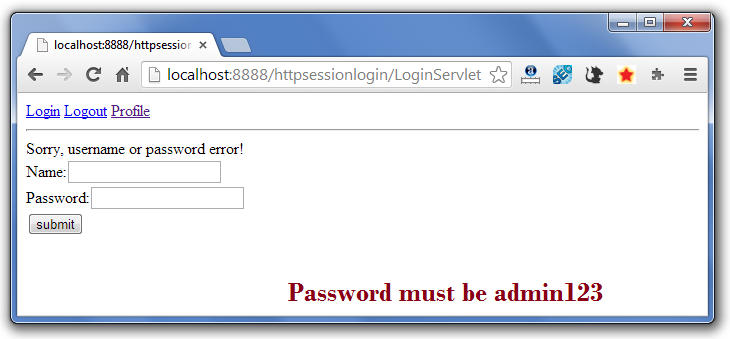
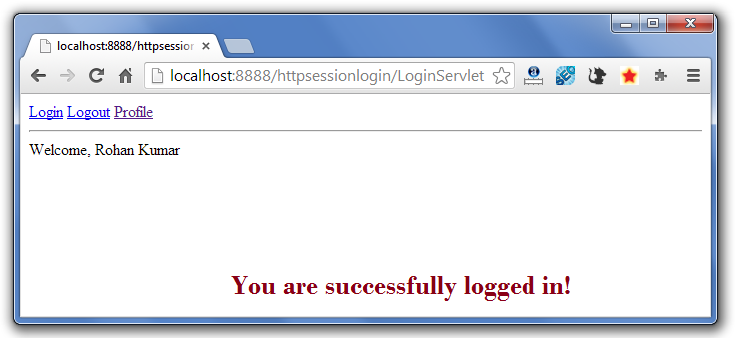
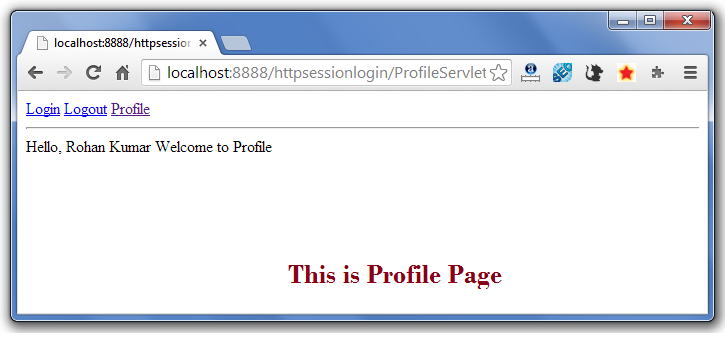
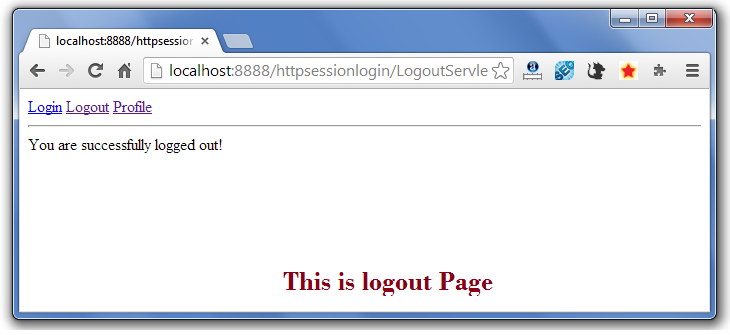
**<servlet-name>**LogoutServlet**</servlet-name>**

**<url-pattern>**/LogoutServlet**</url-pattern>**

**</servlet-mapping>**

**</web-app>**

Output

If again you click on the profile link, you need to login first.