

freelance_Project available to buy contact on 8007592194

SR.NO	Project NAME	Technology
1	Online E-Learning Platform Hub	React+Springboot+MySql
2	PG Mates / RoomSharing / Flat Mates	React+Springboot+MySql
3	Tour and Travel management System	React+Springboot+MySql
4	Election commition of India (online Voting System)	React+Springboot+MySql
5	HomeRental Booking System	React+Springboot+MySql
6	Event Management System	React+Springboot+MySql
7	Hotel Management System	React+Springboot+MySql
8	Agriculture web Project	React+Springboot+MySql
9	AirLine Reservation System / Flight booking System	React+Springboot+MySql
10	E-commerce web Project	React+Springboot+MySql
11	Hospital Management System	React+Springboot+MySql
12	E-RTO Driving licence portal	React+Springboot+MySql
13	Transpotation Services portal	React+Springboot+MySql
14	Courier Services Portal / Courier Management System	React+Springboot+MySql
15	Online Food Delivery Portal	React+Springboot+MySql
16	Muncipal Corporation Management	React+Springboot+MySql
17	Gym Management System	React+Springboot+MySql
18	Bike/Car ental System Portal	React+Springboot+MySql
19	CharityDonation web project	React+Springboot+MySql
20	Movie Booking System	React+Springboot+MySql

freelance_Project available to buy contact on 8007592194

21	Job Portal web project	React+Springboot+MySql
22	LIC Insurance Portal	React+Springboot+MySql
23	Employee Management System	React+Springboot+MySql
24	Payroll Management System	React+Springboot+MySql
25	RealEstate Property Project	React+Springboot+MySql
26	Marriage Hall Booking Project	React+Springboot+MySql
27	Online Student Management portal	React+Springboot+MySql
28	Resturant management System	React+Springboot+MySql
29	Solar Management Project	React+Springboot+MySql
30	OneStepService LinkLabourContractor	React+Springboot+MySql
31	Vehical Service Center Portal	React+Springboot+MySql
32	E-wallet Banking Project	React+Springboot+MySql
33	Blogg Application Project	React+Springboot+MySql
34	Car Parking booking Project	React+Springboot+MySql
35	OLA Cab Booking Portal	React+Springboot+MySql
36	Society management Portal	React+Springboot+MySql
37	E-College Portal	React+Springboot+MySql
38	FoodWaste Management Donate System	React+Springboot+MySql
39	Sports Ground Booking	React+Springboot+MySql
40	BloodBank mangement System	React+Springboot+MySql
41	Bus Tickit Booking Project	React+Springboot+MySql
42	Fruite Delivery Project	React+Springboot+MySql
43	Woodworks Bed Shop	React+Springboot+MySql
44	Online Dairy Product sell Project	React+Springboot+MySql
45	Online E-Pharma medicine sell Project	React+Springboot+MySql
46	FarmerMarketplace Web Project	React+Springboot+MySql
47	Online Cloth Store Project	React+Springboot+MySql
48		React+Springboot+MySql
49		React+Springboot+MySql
50		React+Springboot+MySql

Servlet Tutorial

Introduction

Servlet is a server side Java based programming language widely used to create dynamic web applications. Servlet runs inside a servlet container. It is a portable language so, it will not rely only on one type of web servers. Servlets can be seen as an improved variant of Common Gateway Interface (CGI).

What is servlet container?

Servlet container is a part of web server used to handle various server side technologies request. Servlet container is used to manage the lifecycle and provide runtime environment.

Common Gateway Interface (CGI)

CGI is an initial server side scripting language used to generate webpages dynamically. Unlike servlets, it is not based on Java. CGI programs can be written in many other programming languages like C, C++ and Perl etc. It has some drawbacks which are overcome by Java servlets.

Advantages of servlet over CGI

Servlet	CGI
Servlets are platform independent.	CGI is platform dependent.
In servlets, each request is handled by a thread which provides better performance.	In CGI, each request is handled by a process that degrades the performance.
As servlets are based on Java, they provide more security.	CGI provides less security as its programs are written in programming languages like C, C++ etc.
Servlets can easily handle cookies for session tracking.	CGI is not capable of handling cookie.

Features

Servlet provides various features. Some of them are listed below.

- **Server side:** Servlet is a server side technology. So, it is capable to handle server requests and responses.
- **Dynamic:** Servlet creates dynamic webpages. So the content of these pages can be updated dynamically with time and requirements.
- **Portable:** Java is a platform independent language. So being based on Java, servlets created on one operating system having any web server can easily run on another operating system having any other web server.
- **Secured and Robust:** Since Servlet is based on Java programming language so, it are secure and robust like Java.
- **Better performance:** Servlet uses thread instead of a process. So, we can perform multiple tasks more efficiently.

Life Cycle of Servlet

Servlet container is responsible to manage the lifecycle of a servlet. Servlet provides three methods as a part of its lifecycle. Let's study the lifecycle of servlets with the following steps.

- **Loading class:** This is an initial stage of servlet in which a servlet class is loaded whenever a request is made.
- **Creating instance:** As soon as the class is loaded, servlet container created the instance of that class.
- **init():** In this step, the servlet container invokes an `init()` method to initialize the servlet instance. An object of `ServletConfig` interface is passed within this method. Note that **init() method** is invoked **once in a lifetime** of a servlet.
- **service():** After initialization, servlet container invokes `service()` method for every request. Each request is handled by a separate thread. This method is used to perform various operations.

- **destroy():** This is the final stage of the servlet. In this phase, servlet container invokes **destroy()** method for closing the servlet. As soon as **destroy()** method is invoked the memory allocated to servlet and its object is collected by the automatic garbage collector. Note that **destroy()** method is invoked **once in a lifetime** of a servlet.

Servlet Packages

There are two packages in Java Servlet that provide various features to servlet. These two packages are `javax.servlet` and `javax.servlet.http`.

1. **javax.servlet package:** This package contains various servlet interfaces and classes which are capable of handling any type of protocol.
2. **javax.servlet.http package:** This package contains various interfaces and classes which are capable of handling a specific http type of protocol.

Overview of some important interfaces and classes

1. javax.servlet package interface

Some of the important interfaces are listed below.

Interface	Overview
Servlet	This interface is used to create a servlet class. Each servlet class must require to implement this interface either directly or indirectly.
ServletRequest	The object of this interface is used to retrieve the information from the user.
ServletResponse	The object of this interface is used to provide response to the user.
ServletConfig	ServletConfig object is used to provide the information to the servlet class explicitly.
ServletContext	The object of ServletContext is used to provide the information to the web application explicitly.

2. javax.servlet package classes

Some of the important classes are listed below.

Classes	Overview
GenericServlet	This is used to create servlet class. Internally, it implements the Servlet interface.
ServletInputStream	This class is used to read the binary data from user requests.
ServletOutputStream	This class is used to send binary data to the user side.
ServletException	This class is used to handle the exceptions occur in servlets.
ServletContextEvent	If any changes are made in servlet context of web application, this class notifies.

1. javax.servlet.http package interface

Some of the important interface of this package are listed below:

Interface	Overview
HttpServletRequest	The object of this interface is used to get the information from the user under http protocol.
HttpServletResponse	The object of this interface is used to provide the response of the request under http protocol.
HttpSession	This interface is used to track the information of users.
HttpSessionAttributeListener	This interface notifies if any change occurs in HttpSession attribute.
HttpSessionListener	This interface notifies if any changes occur in HttpSession lifecycle.

2. javax.servlet.http package classes

Some of the important interface of this package are listed below.

Class	Overview
HttpServlet	This class is used to create servlet class.
Cookie	This class is used to maintain the session of the state.
HttpSessionEvent	This class notifies if any changes occur in the session of web application.
HttpSessionBindingEvent	This class notifies when any attribute is bound, unbound or replaced in a session.

web.xml file

Creating a servlet class is not enough. We have to deploy that class also. So, to deploy the class we use web.xml file. In this file we have to map our class configurations.

```
<?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_3_0.xsd"
id="WebApp_ID" version="3.0">
  <display-name>Filename</display-name>
  <servlet>
<servlet-name>ServletName</servlet-name>
<servlet-class>ClassName</servlet-class>
</servlet>
<servlet-mapping>
```



```

<servlet-name>ServletName</servlet-name>
<url-pattern>/PathName</url-pattern>
</servlet-mapping>
<welcome-file-list>
  <welcome-file>index.html</welcome-file>
  <welcome-file>index.htm</welcome-file>
  <welcome-file>index.jsp</welcome-file>
  <welcome-file>default.html</welcome-file>
  <welcome-file>default.htm</welcome-file>
  <welcome-file>default.jsp</welcome-file>
</welcome-file-list>
</web-app>

```

We just required to edit and add some tag in our web.xml file through a text editor. Each tag contains some unique specification.

<servlet-name>: We can provide any specific name that represents a servlet class. The name can be same as of class name.

<servlet-class>: A name of servlet class is given with this tag.

<welcome-file-list>: If we want to attach other files such as html, htm, jsp with our servlet class then we have to deploy that file in this tag. Here each file is mapped separately in <welcome-file> tag.

Servlet Interface

Java provides a servlet interface in javax.servlet package that is implemented by servlet program either directly or indirectly. This interface provides five methods and these methods must be declared within the servlet program if we are implementing a servlet interface. So implementing the servlet interface is a direct approach.

We also discuss about the indirect approach later

Methods of Servlet interface

Methods provided by servlet interface are mentioned below.

Method	Description
public void <code>init(ServletConfig con)</code>	This method is invoked by servlet container once only. The purpose of it is to indicate that the servlet is ready to use.
public void <code>service(ServletRequest req, ServletResponse res) throws ServletException</code>	This method is invoked every time whenever a response is given to a request. So this method is used to perform actual tasks.
public <code>ServletConfig</code> <code>getServletConfig()</code>	This method return an object of <code>ServletConfig</code> that contains initialization parameters of the servlet.
public String <code>getServletInfo()</code>	This method is used to provide the information about the servlet.
public void <code>destroy()</code>	This method is invoked by servlet container once only. It is used to indicate that now the server is terminated and no more tasks will execute.

First servlet program using Servlet interface This is the simple example of servlet interface with its methods. **FirstServlet.java**

```
import javax.servlet.*;

public class FirstServlet implements Servlet
{
    public void init(ServletConfig con)
    {
```

```

        System.out.println("init method is invoked once only");
    }
    public void service(ServletRequest req,ServletResponse res) throws
ServletException
    {
        System.out.println("service method is invoked");
    }
    public ServletConfig getServletConfig()
    {
        return null;
    }
    public String getServletInfo()
    {
        return "info";
    }
    public void destroy()
    {
        System.out.println("destroy method is invoked once only");
    }
}

```

Web.xml Now we have to deploy are servlet class in web.xml file. So, open your web.xml file and edit the required tag.

```

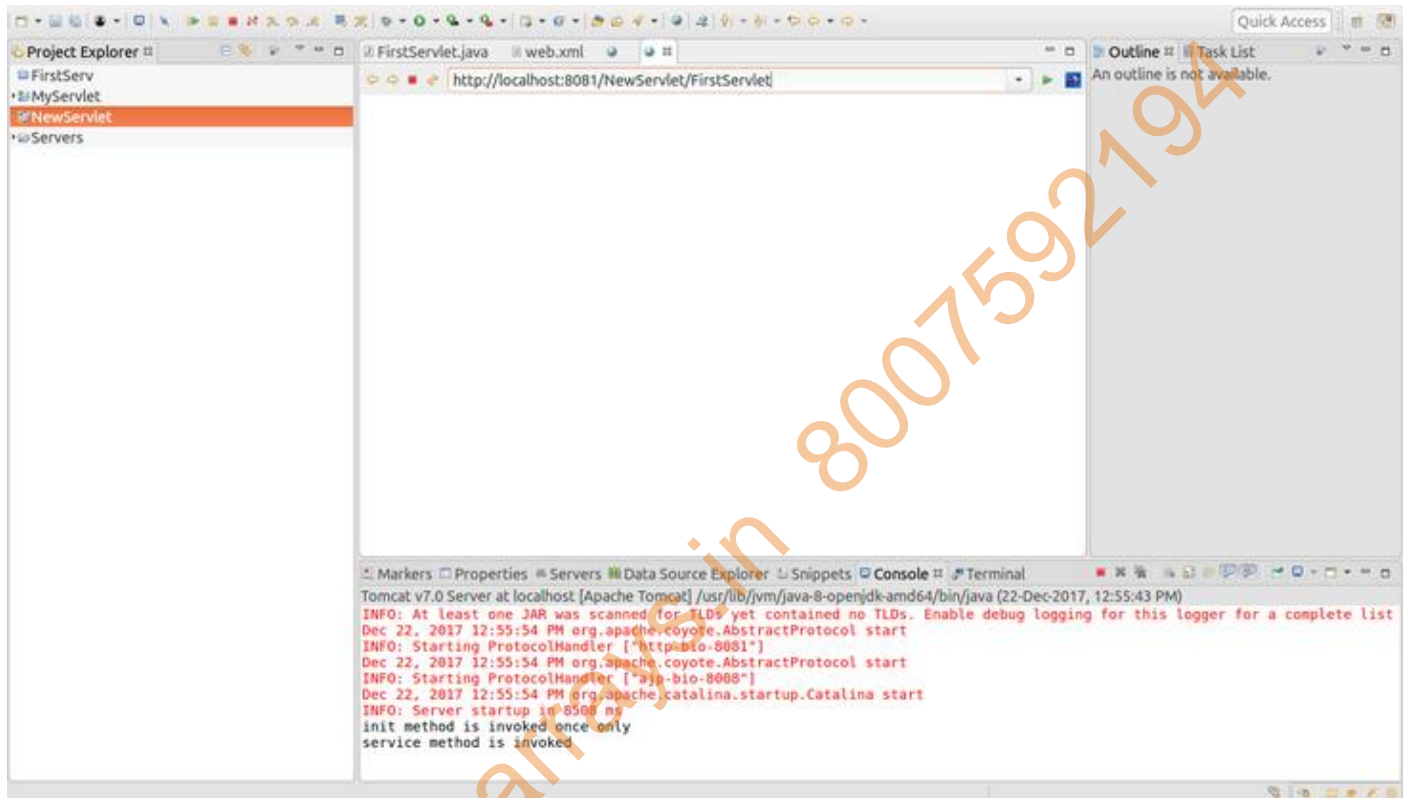
<web-app>
<servlet>
<servlet-name>FirstServlet</servlet-name>
<servlet-class>FirstServlet</servlet-class>
</servlet>

```

```

<servlet-mapping>
<servlet-name>FirstServlet</servlet-name>
<url-pattern>/FirstServlet</url-pattern>
</servlet-mapping>
</web-app>

```



See here init() method and service() method is invoked but destroy() method is still not invoked. This is because server is still in running mode. **Output:**

```

Dec 22, 2017 1:48:26 PM org.apache.catalina.startup.Catalina start
INFO: Server startup in 4582 ms
init method is invoked once only
service method is invoked
service method is invoked
service method is invoked
Dec 22, 2017 1:49:34 PM org.apache.catalina.core.StandardServer await
INFO: A valid shutdown command was received via the shutdown port. Stopping the Server instance.
Dec 22, 2017 1:49:34 PM org.apache.coyote.AbstractProtocol pause
INFO: Pausing ProtocolHandler ["http-bio-8081"]
Dec 22, 2017 1:49:34 PM org.apache.coyote.AbstractProtocol pause
INFO: Pausing ProtocolHandler ["ajp-bio-8008"]
Dec 22, 2017 1:49:34 PM org.apache.catalina.core.StandardService stopInternal
INFO: Stopping service Catalina
destroy method is invoked once only
Dec 22, 2017 1:49:34 PM org.apache.coyote.AbstractProtocol stop

```

Everytime when we refresh our page service() method is invoked. As soon as we stop the server destroy() method is also invoked which indicates that now no more task will execute.

Servlet Request and Response

Servlet handles various client requests and provides their responses. It provides two interfaces i.e ServletRequest and ServletResponse for this purpose. Let's discuss about these interfaces in detail.

1. ServletRequest interface

ServletRequest is an interface whose object is used to provide the information of each request to servlet. This information may be any name, type, value or other attribute. This interface is present in javax.servlet package. Servlet container is responsible to create ServletRequest object which is given with service() method argument.

Methods of ServletRequest

These are some important methods provided by ServletRequest interface.

Method	Description
String getParameter(String name)	This method returns the value given in request as a String.
Object getAttribute(String name)	This method provides the attribute of request as an Object.
String getServerName()	This method provides the server name to which request is sent.
int getServerPort()	This method returns the port number to which request is sent.
boolean isSecure()	This method indicates whether the server is secure or not.

2. ServletResponse interface

The object of `ServletResponse` interface is used to send the responses to the clients. The information send in responses can be a binary or character data. `ServletResponse` interface is present in `javax.servlet` package and passes as an argument of `service()` method.

Methods of ServletResponse

Method	Description
<code>PrintWriter getWriter()</code>	This method is used to send character data in response.
<code>int getBufferSize()</code>	This method returns the capacity of buffer in sent response.
<code>ServletOutputStream getOutputStream()</code>	This method is used to send binary data in response.
<code>boolean isCommitted()</code>	This method indicates the completion of response.
<code>void reset()</code>	This method is used to remove the data present in buffer.
<code>void setContentType(String type)</code>	This method is used to set the type of content.

Servlet Request and Response example with HTML

In this example, the name is given as a request in HTML form. The object of `HttpServletRequest` interface is used to handle the request and `HttpServletResponse` interface is used to provide the response.

index.html

```
<form action="serv" method="get">
<h3>Enter your Name:</h3>
<input type="text" name="username">
```

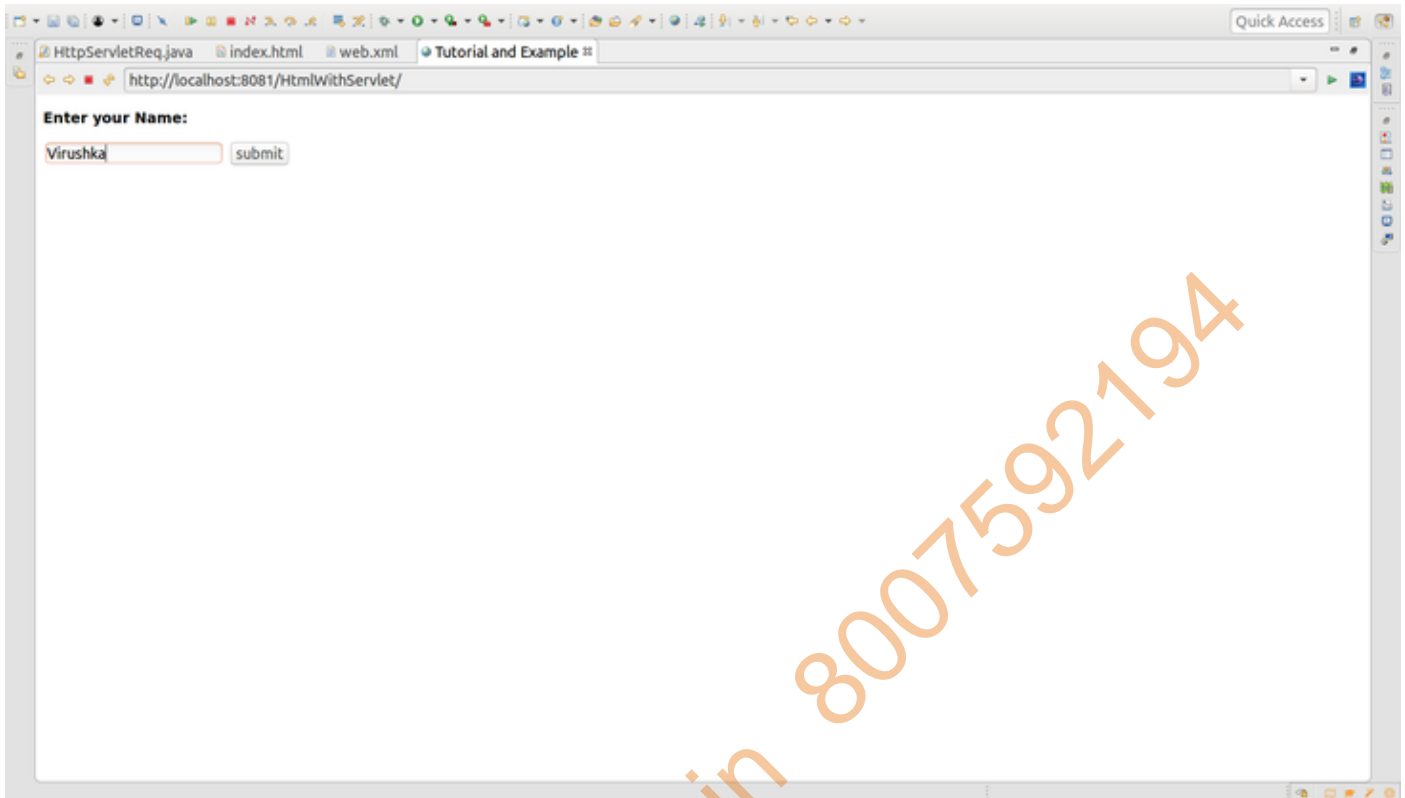
```
<input type="submit" value="submit">
</form>
```

HttpServletReq.java

```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;
@WebServlet("/serv")
public class HttpServletReq extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
ServletException, IOException
    {
        res.setContentType("text/html");
        PrintWriter pw=res.getWriter();
        String s1=req.getParameter("username");
        pw.println("<h1>Welcome:"+s1+"</h1>");
        String s2=req.getServerName();
        pw.println("<h1>Server name:"+s2+"</h1>");
        int i=req.getServerPort();
        pw.println("<h1>Server Port:"+i+"</h1>");
        boolean b=req.isSecure();
        pw.println("<h1>Is the server secure:"+b+"</h1>");
    }
}
```

```
}
```

Output:



ServletConfig

Servlet container uses the object of ServletConfig interface to provide the information to servlet during initialization. ServletConfig object fetch this information from web.xml file. So if we have to provide some dynamic information to servlet then we can pass it from web.xml file without making any change in our servlet class.

From servlet 3.0 we can also declare this information in servlet class itself. This can be done via annotations. Now we just required to pass the name with its value within @WebInitParam annotation. This annotation is present in javax.servlet.annotation.WebInitParam package.

Methods of ServletConfig

Method	Description
String <code>getInitParameter</code> (String name)	This method returns the value of specific parameter name.
Enumeration <code>getInitParameterNames</code> ()	This method provides the enumeration of names.
ServletContext <code>getServletContext</code> ()	This method returns the object of ServletContext.
String <code>getServletName</code> ()	This method returns the name of servlet instance.

Example of ServletConfig interface

In this example, we are using annotations that contains subject as name and marks as it's specific value.

DemoServletConfig.java

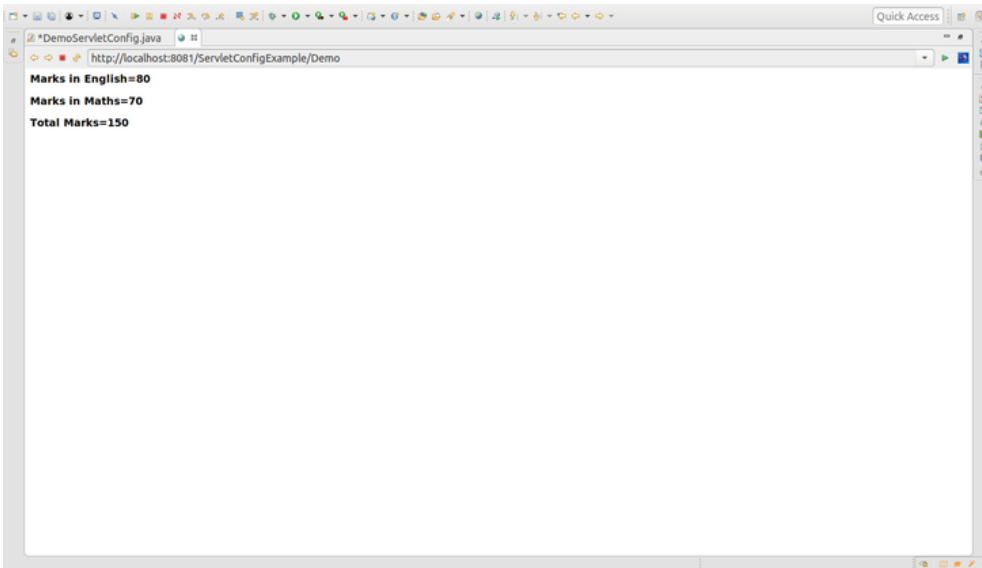
```
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;
import javax.servlet.annotation.WebInitParam;

@WebServlet(
    urlPatterns = {"/Demo"}, initParams = {
        @WebInitParam(name= "English", value="80"),
        @WebInitParam(name= "Maths", value= "70"),
    }
)

public class DemoServletConfig extends HttpServlet
```

```
{  
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws  
        ServletException, IOException  
    {  
        res.setContentType("text/html");  
        PrintWriter pw=res.getWriter();  
        ServletConfig con=getServletConfig();  
        String nm1=con.getInitParameter("English");  
        String nm2=con.getInitParameter("Maths");  
        Integer int1=new Integer(nm1);  
        Integer int2=new Integer(nm2);  
        int sum=int1+int2;  
        pw.println("<html><body>");  
        pw.println("<h3>Marks in English="+int1+"</h3>");  
        pw.println("<h3>Marks in Maths="+int2+"</h3>");  
        pw.println("<h3>Total Marks="+sum+"</h3>");  
        pw.println("</body></html>");  
    }  
}
```

Output:



ServletContext

ServletContext interface is used to provide the configuration information per web application explicitly. Thus unlike ServletConfig, ServletContext interface can be used to provide information to more than one servlet of web application from web.xml file.

So if we want to pass the same information to all servlets of a web application then passing it from web.xml file at once is much efficient way than providing the same information in each servlet separately.

For sharing information from web.xml we need to pass the name in <param-name> and value in <param-value> tags. <context-param> contains these tags.

Methods of ServletContext interface

Methods	Description
Object <code>getAttribute(String name)</code>	This method returns the attribute of specific name.
Enumeration <code>getAttribute()</code>	This method returns the enumeration of attributes names.
String <code>getInitParameter(String name)</code>	This method returns the parameter of name in the form of string.

String <code>getServletInfo()</code>	This method returns the information about servlet container such as its name and version.
void <code>removeAttribute(String name)</code>	We can remove the attributes of name using this method

Example of ServletContext interface

In this example, two java servlet classes are taken. Both classes contains different information of students of same college. So instead of providing college name every time we are sharing this information from web.xml file.

DemoServletContext.java

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

public class DemoServletContext extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
    ServletException, IOException
    {
        res.setContentType("text/html");
        PrintWriter pw=res.getWriter();
        ServletContext con=getServletContext();
        String st=con.getInitParameter("College");
        pw.println("<html><body>");
        pw.println("<h1>Name:Anuj</h1>");
        pw.println("<h1>Roll number:101</h1>");
        pw.println("<h1>College:"+st+"</h1>");
        pw.println("</body></html>");
    }
}
```

```
}
}
```

DemoServletContext1.java

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

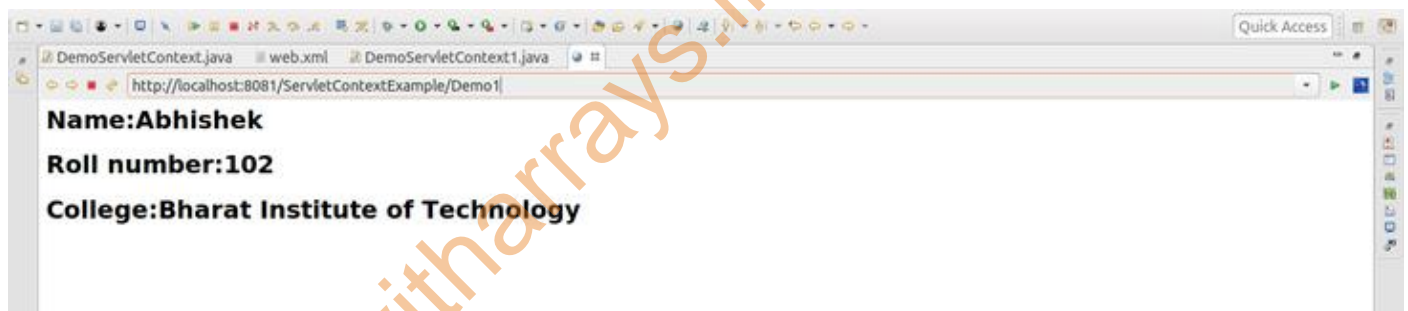
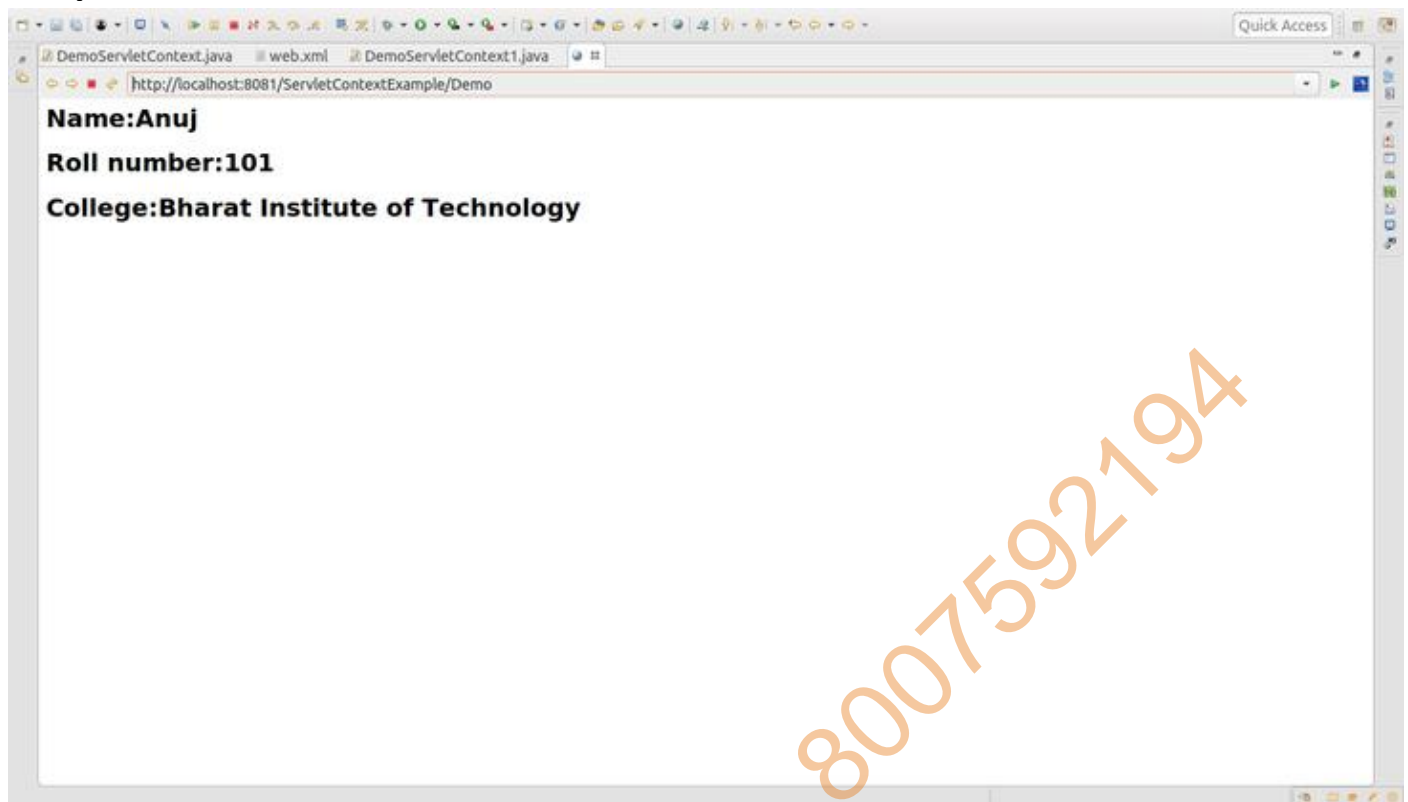
public class DemoServletContext1 extends HttpServlet
{
    public void doGet(HttpServletRequest req, HttpServletResponse res) throws
    ServletException, IOException
    {
        res.setContentType("text/html");
        PrintWriter pw=res.getWriter();
        ServletContext con=getServletContext();
        String st=con.getInitParameter("College");
        pw.println("<html><body>");
        pw.println("<h1>Name:Abhishek</h1>");
        pw.println("<h1>Roll number:102</h1>");
        pw.println("<h1>College:"+st+"</h1>");
        pw.println("</body></html>");
    }
}
```

Web.xml

```
<servlet>
    <servlet-name>DemoServletContext</servlet-name>
    <servlet-class>DemoServletContext</servlet-class>
</servlet>
```

```
<servlet>
  <servlet-name>DemoServletContext1</servlet-name>
  <servlet-class>DemoServletContext1</servlet-class>
</servlet>
<context-param>
  <param-name>College</param-name>
  <param-value>Bharat Institute of Technology</param-value>
</context-param>
<servlet-mapping>
  <servlet-name>DemoServletContext1</servlet-name>
  <url-pattern>/Demo1</url-pattern>
</servlet-mapping>
<servlet-mapping>
  <servlet-name>DemoServletContext</servlet-name>
  <url-pattern>/Demo</url-pattern>
</servlet-mapping>
```

Output:



RequestDispatcher

RequestDispatcher is an interface used to receive requests from the users and bind it with other files such as HTML file, Servlet file, JSP file etc. Servlet container is responsible to create RequestDispatcher object.

RequestDispatcher provides `forward()` and `include()` methods. These methods are used to call RequestDispatcher.

Methods of RequestDispatcher

Method	Description
void forward(ServletRequest req, ServletResponse res)	If this method is invoked then the request of current file is send forward to the another file of any type such as HTML, Servlet, JSP etc. and the response of that file is provided by the server.
void include(ServletRequest req, ServletResponse res)	If this method is invoked then the content of current file such as HTML, Servlet, JSP is included with the response

Example of RequestDispatcher interface

In this example we create a form in html file. ServletChecker.java file checks the password field of that form. If the entered password length is less than 8 then include() method is invoked and generates error else forward() method is invoked and forward that request to another java file.

index.html

```
<form action="serv" method="post">
<table>
<h1>
<tr><td><h3>Name:</h3><td><input type="text"
name="name"></h3></td></tr> <br>
<tr><td><h3>Password:</h3><td><input type="password" name="pass"
placeholder="Must be of 8 characters"></h3></td></tr> <br>
<tr><td><h1><input type="submit" value="sign up"></h1></td></tr>
</h1>
</table>
</form>
```

ServletChecker.java

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

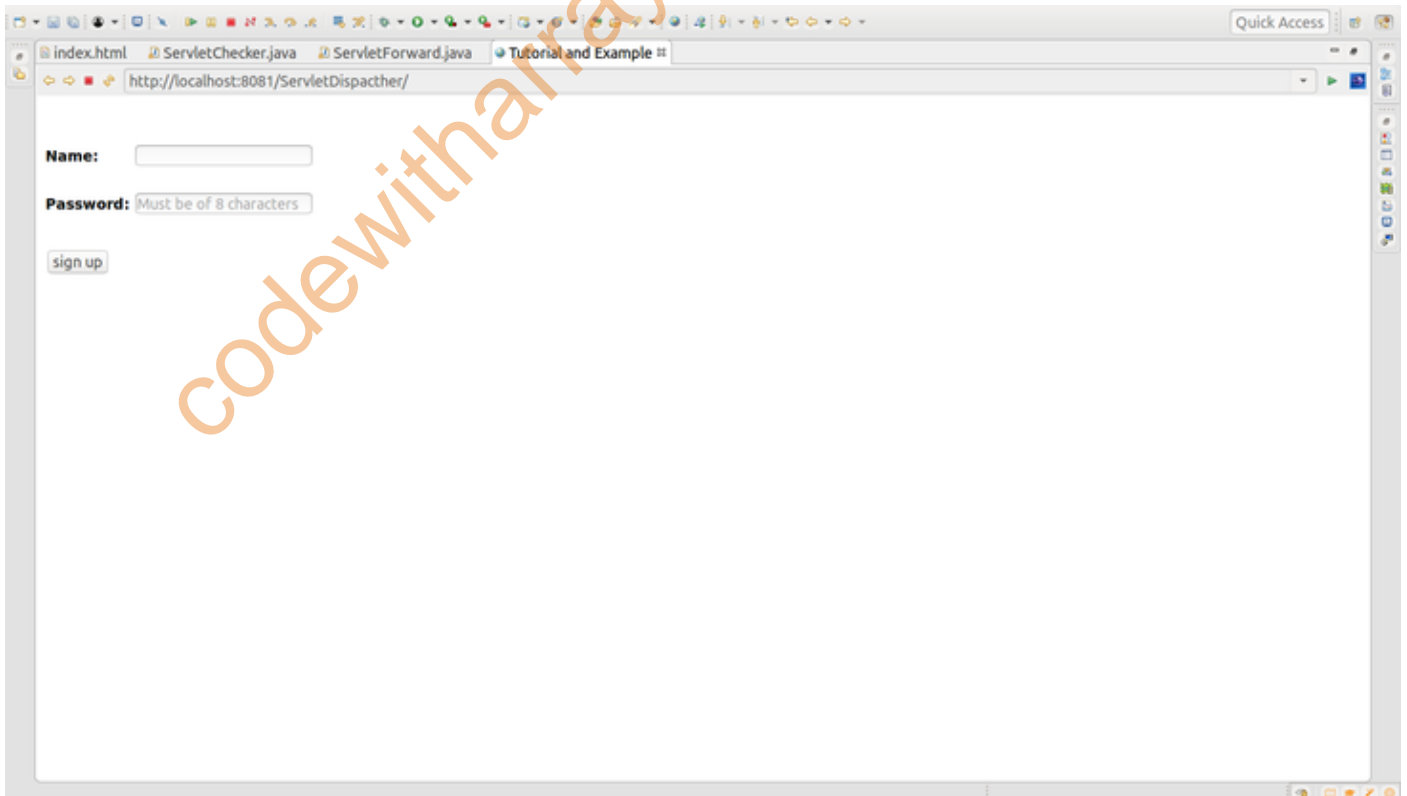
```
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;
@WebServlet("/serv")
public class ServletChecker extends HttpServlet
{
    public void doPost(HttpServletRequest req,HttpServletResponse res) throws
ServletException,IOException
    {
        res.setContentType("text/html");
        PrintWriter pw=res.getWriter();
        String st1=req.getParameter("pass");
        int i1=st1.length();
        if(i1<8)
        {
            pw.println("<h1>Error:Password must be of 8 character</h1>");
            RequestDispatcher rd=req.getRequestDispatcher("/index.html");
            rd.include(req,res);
        }
        else
        {
            RequestDispatcher rd1=req.getRequestDispatcher("ServletForward");
            rd1.forward(req,res);
        }
    }
}
```

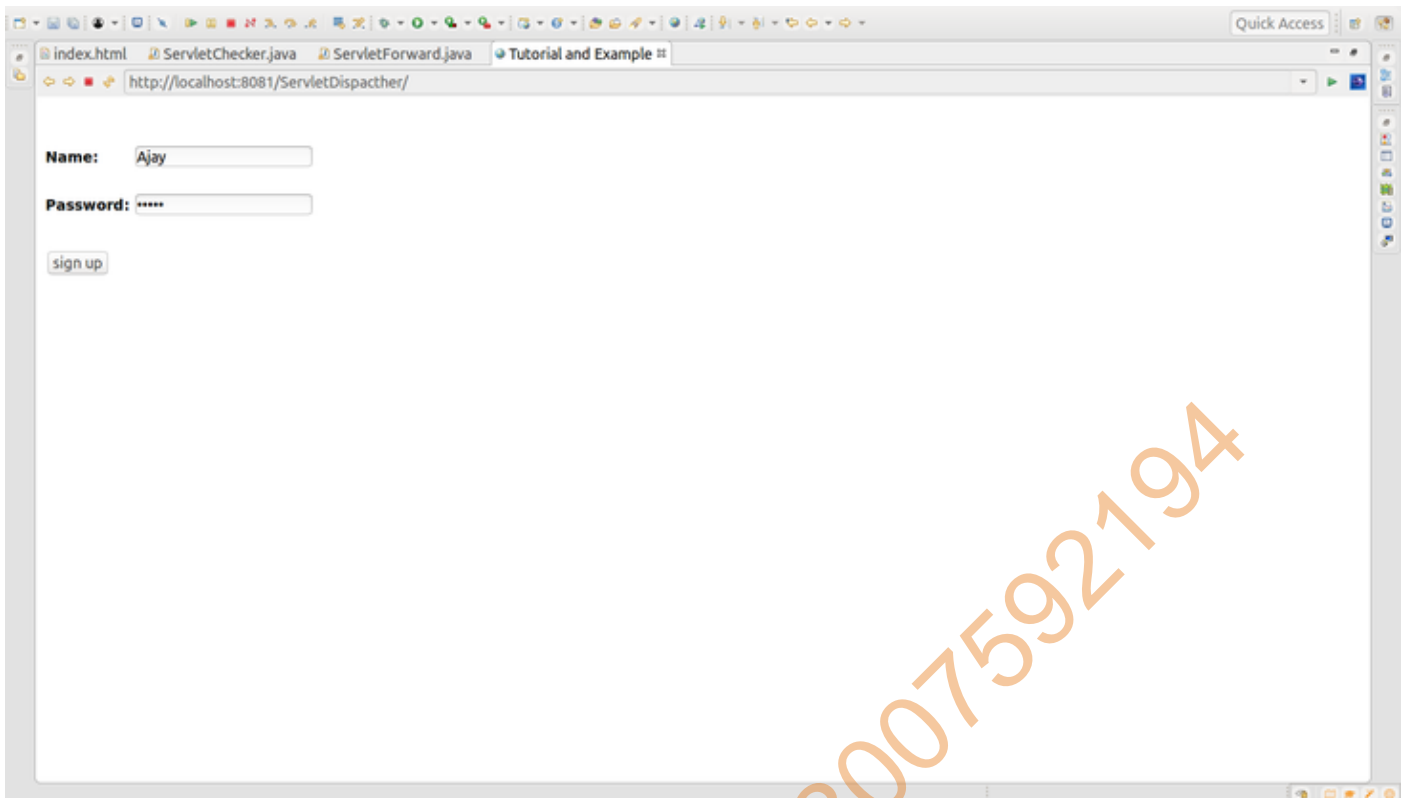
ServletForward.java

```
import java.io.*;
```

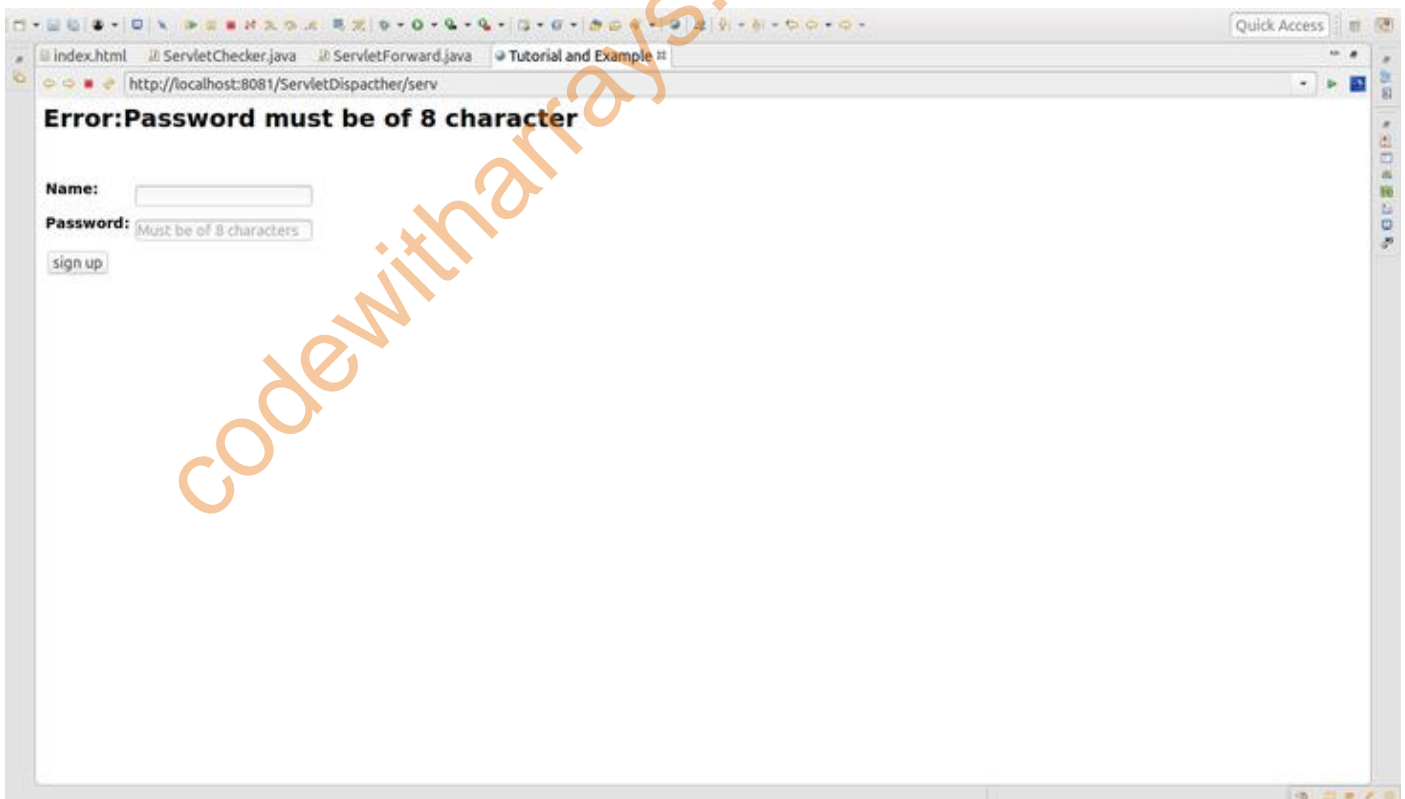
```
import javax.servlet.*;  
import javax.servlet.http.*;  
import javax.servlet.annotation.WebServlet;  
@WebServlet("/ServletForward")  
public class ServletForward extends HttpServlet  
{  
    public void doPost(HttpServletRequest req,HttpServletResponse res) throws  
ServletException,IOException  
    {  
        res.setContentType("text/html");  
        PrintWriter pw=res.getWriter();  
        String st1=req.getParameter("name");  
        pw.println("<h1>Welcome "+ st1+"</h2>");  
        pw.println("<h3>You registered successfully</h3>");  
    }  
}
```

Output:

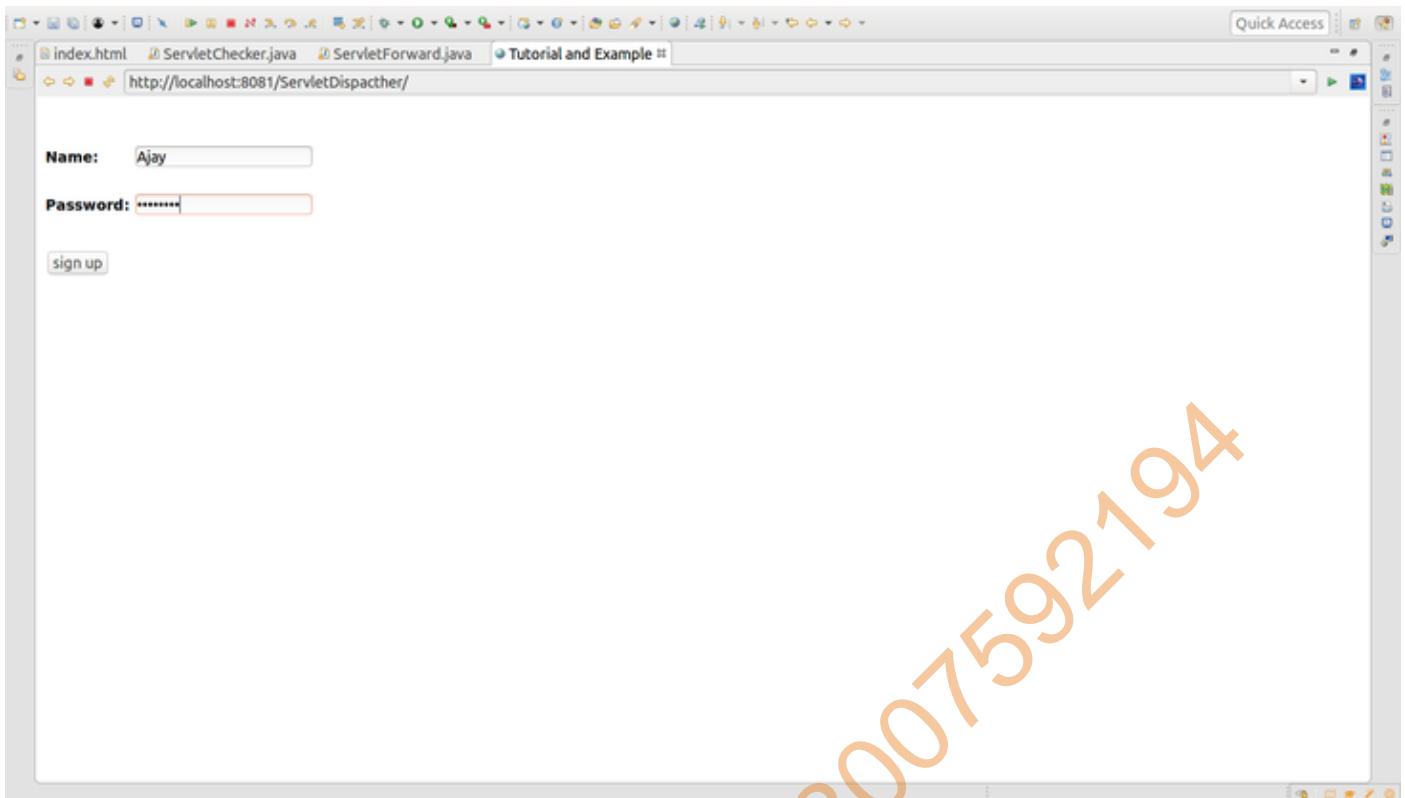




The inserted length of the password is less than 8 character. So it will generate error message and include the current form within it.



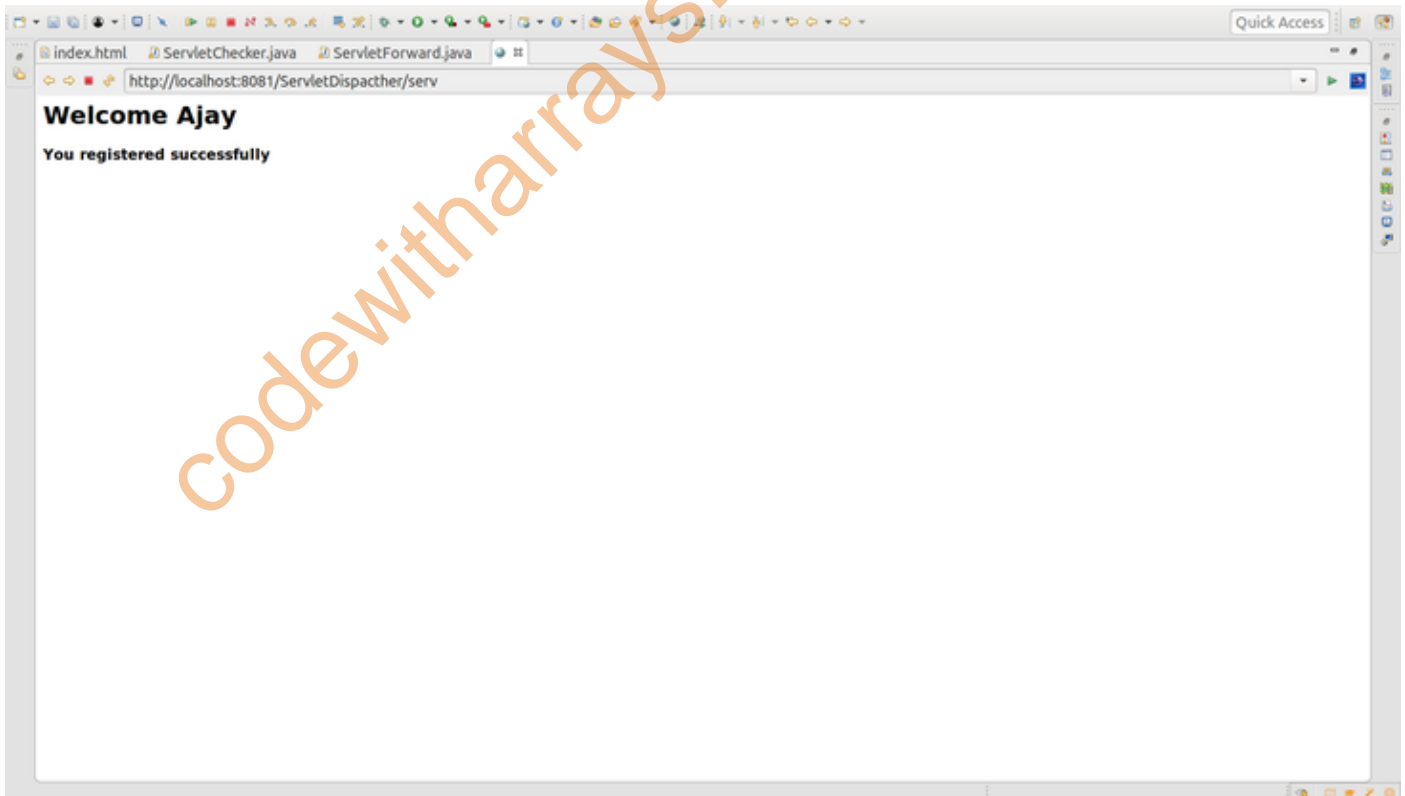
As expected error message generated.



A screenshot of a web browser window. The address bar shows `http://localhost:8081/ServletDispatcher/`. The browser tabs include `index.html`, `ServletChecker.java`, `ServletForward.java`, and `Tutorial and Example`. The page content shows a registration form with the following elements:

- Name:** A text input field containing the text "Ajay".
- Password:** A password input field with masked characters "*****".
- sign up**: A button located below the password field.

Now inserted length of the form is greater than 8 character. So `forward()` method is invoked and sent as a request to another page.



The following response generated.

Servlet Session Management

What is session?

A session is a link between a client and the server. This link represents some identity that makes client unique so that it is known by the server whenever a request is made.

Need to maintain this session?

Whenever a client made a request, the server treats it as a new request. In some applications we require that a series of requests made by a client will associate with one another. So we need to maintain the state of a user.

How to maintain a session?

To maintain a session, servlet container uses various methods to make a particular client unique so that server can easily identify it.

Different ways to maintain session

There are several ways in servlet to maintain session:

- HttpSession
- Cookies
- Hidden Form Field
- URL Rewriting

1. Servlet HttpSession

- HttpSession is an interface used by servlet container to create a session between Http client and Http server. So using this interface we can maintain the state of a user.
- The object of HttpSession stores various information about the user. This interface is present in javax.servlet.http package.
- **Methods of HttpSession**
- These are some important methods provided by HttpSession interface.

Methods	Description
String getId()	This method returns a unique id of client used to identify it.

long <code>getLastAccessedTime()</code>	This method provides the time when last request is made. It returns the time in milliseconds.
Long <code>getCreationTime()</code>	This method provides the time when first request is made. It returns the time in milliseconds.
void <code>setMaxInactiveInterval()</code>	This method sets time in seconds after which the session becomes invalid.
Object <code>getAttribute(String name)</code>	This method provides the value of attribute bind with the name passed within it.

- **Example of HttpSession**

- In this example, we will maintain the state of a particular client. When a client makes first request then its unique id is generated. With each request, number of visit of a user is incremented by one unless request will be made within a specific period of time given in `setMaxInactiveInterval()` method unless the session becomes expire.

- **HttpSessionExample.java**

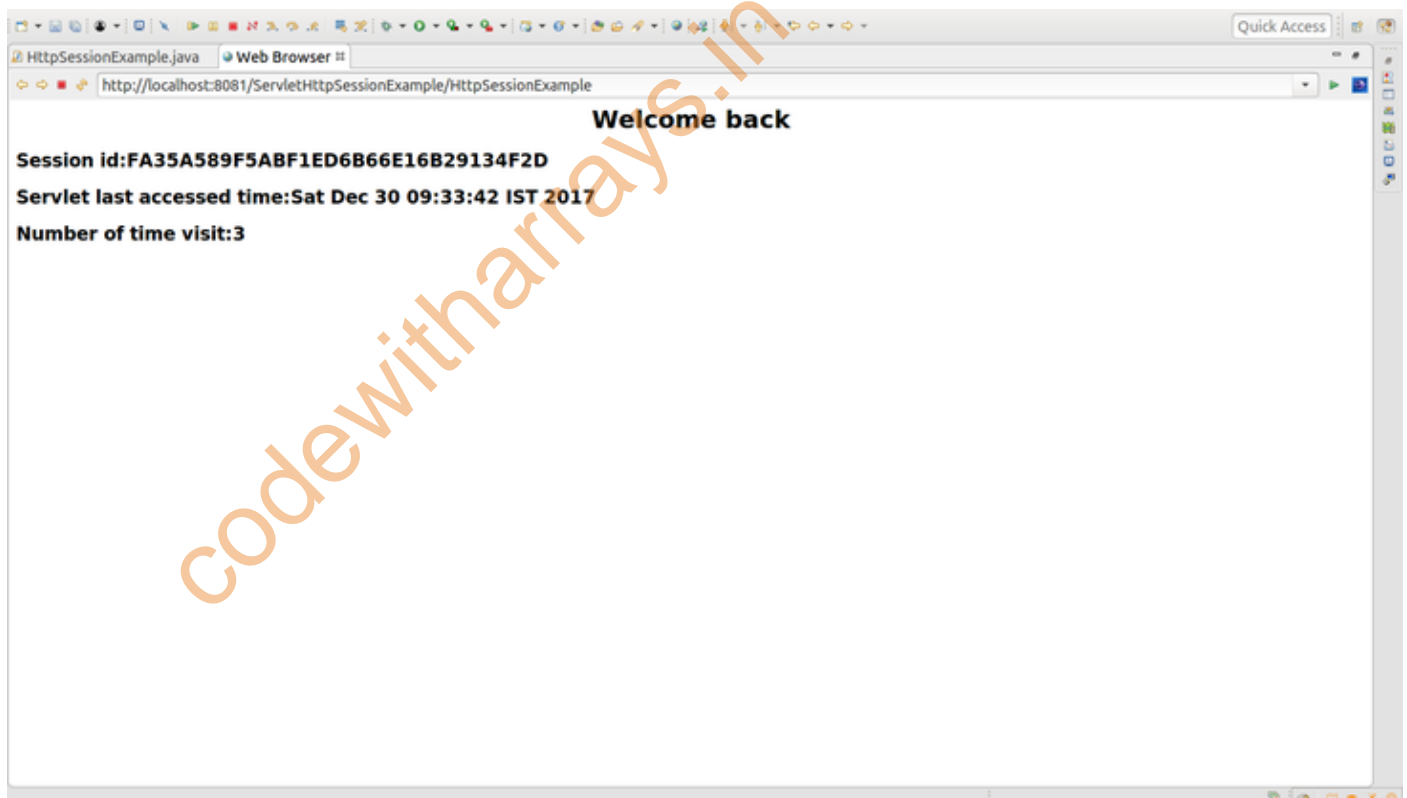
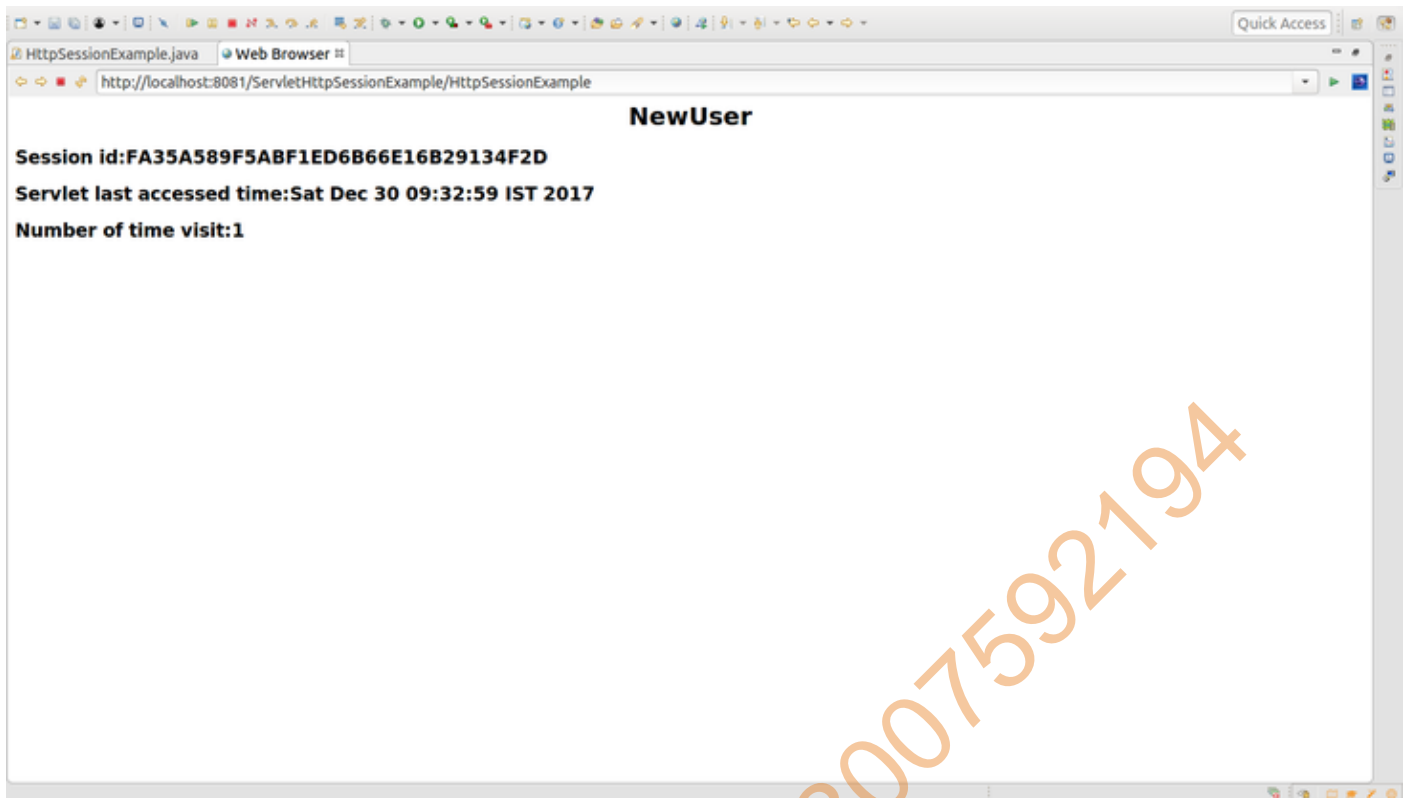
```

import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import java.util.*;
import javax.servlet.annotation.WebServlet;
@WebServlet("/HttpSessionExample")
public class HttpSessionExample extends HttpServlet {
    public void doGet(HttpServletRequest req, HttpServletResponse res)
    throws ServletException, IOException {
        res.setContentType("text/html");
        HttpSession session = req.getSession();
        res.setContentType("text/html");
        PrintWriter pw=res.getWriter();
        Integer attribute=(Integer)session.getAttribute("attribute");
        if (attribute == null)
        {
            attribute = new Integer(1);
            pw.println("<h1><center>NewUser</center></h1>");

```


- } else {
- pw.println("<h1><center>Welcome back</center></h1>");
- attribute = new Integer(attribute.intValue() + 1);
- }
- session.setAttribute("attribute",attribute);
- session.setMaxInactiveInterval(60);
- pw.println("<h2>Session id:"+session.getId()+"</h2>");
- pw.println("<h2>Servlet last accessed time:"+new
Date(session.getLastAccessedTime())+"</h2>");
- pw.println("<h2>Number of time visit:"+attribute+"</h2>");
- }}

codewitharrays.in 8007592194



2.Servlet Cookie

Cookie is a class that is used for session management. It contains a small amount of information which persists between the server and client. This information includes name, value and other attributes.

Servlet send this information to web browser and browser saved it so that whenever browser makes a request then it also sends this information along with it. Thus through cookies a server can easily identify a client.

Methods of Cookie

Method	Description
String getName()	This method returns the name of cookie.
String getValue()	This method returns the current value of cookie.
void setValue(String name)	We can set the value of cookie using this method.
void setMaxAge(int i)	This method is used to set the maximum age of cookie after which the session becomes expire.
int getMaxAge()	This method returns the maximum age of a cookie.

3.Servlet URL Rewriting

This is another approach to maintain the session of a user. In this mechanism, when a client make a request then some extra information is appended in the URL of that request.

This extra information uniquely identifies a user and it may be a path info, parameter attribute etc. Unlike cookies, URL Rewriting will also work when cookies are disabled in our browser.

Example of URL Rewriting

Let's see an example of URL Rewriting.

Welcome.jsp

```
<html>
```

```

<body>
<center>
<h1>Welcome to Tutorial and Example</h1>
</center>
</body>
</html>

```

DemoURLRewriting.java

```

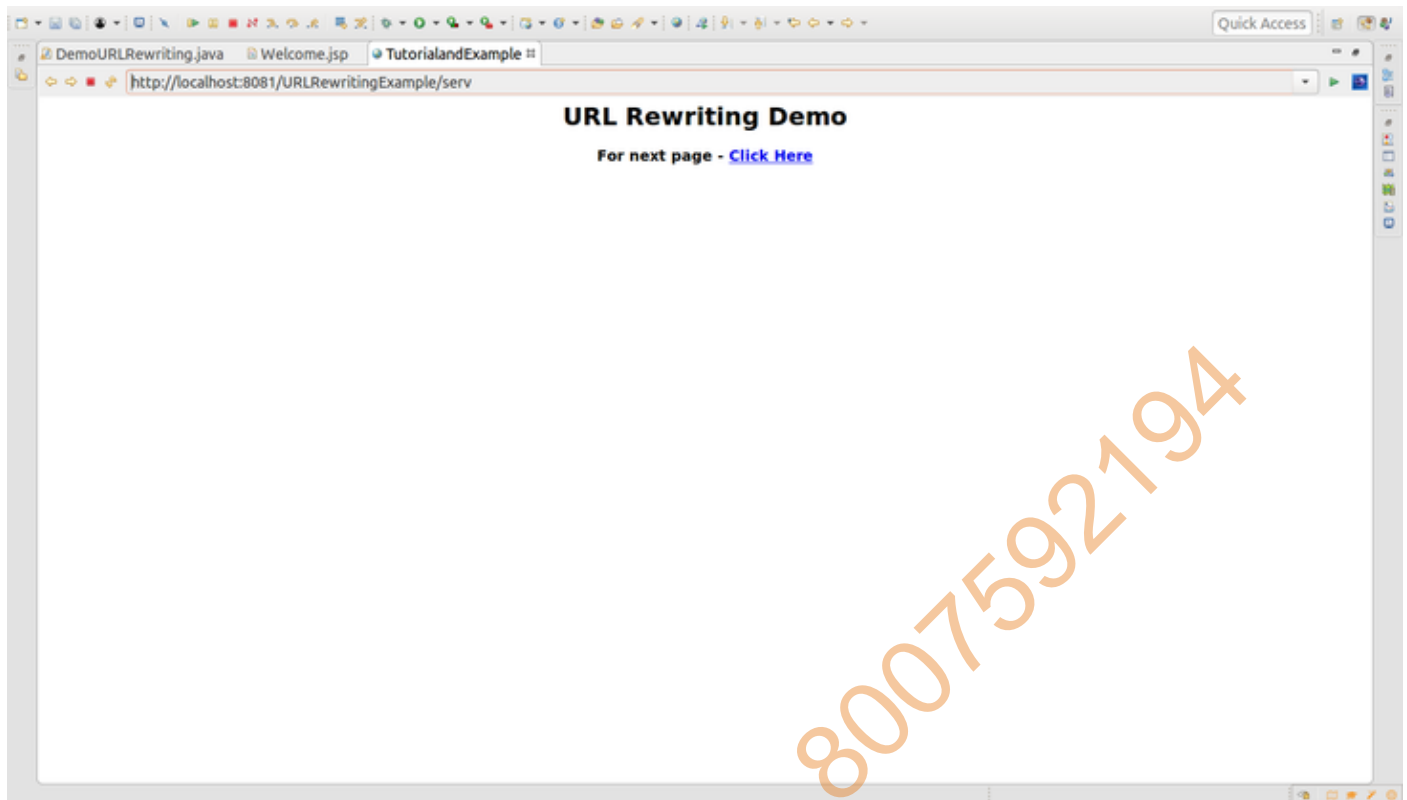
import java.io.*;
import javax.servlet.*;
import javax.servlet.http.*;
import javax.servlet.annotation.WebServlet;
@WebServlet("/serv")
public class DemoURLRewriting extends HttpServlet {
protected void doGet(HttpServletRequest req,
HttpServletResponse res) throws ServletException, IOException {
res.setContentType("text/html");
PrintWriter pw=res.getWriter();
String path = req.getContextPath();
String addURL = res.encodeURL(path + "/Welcome.jsp");
pw.println("<html>");
pw.println("<head>");
pw.println("<title>TutorialandExample</title>");
pw.println("</head>");
pw.println("<body><center>");
pw.println("<h1>URL Rewriting Demo</h1>");
pw.println("<h3>For next page - <a href=\"\" + addURL

```

```
+ "\"> Click Here</a></h3>");  
pw.println("</center></body>");  
pw.println("</html>");  
}  
}
```

Output:





4. Servlet Hidden Form Field

Hidden form field is an invisible field that saves the information regarding the state in client browser. As this information is invisible so client can't see it but it is visible to servlet.

These fields are added to HTML form and when client submit form then these fields are also sent to the server with form.

Example of Hidden Form Field

index.html

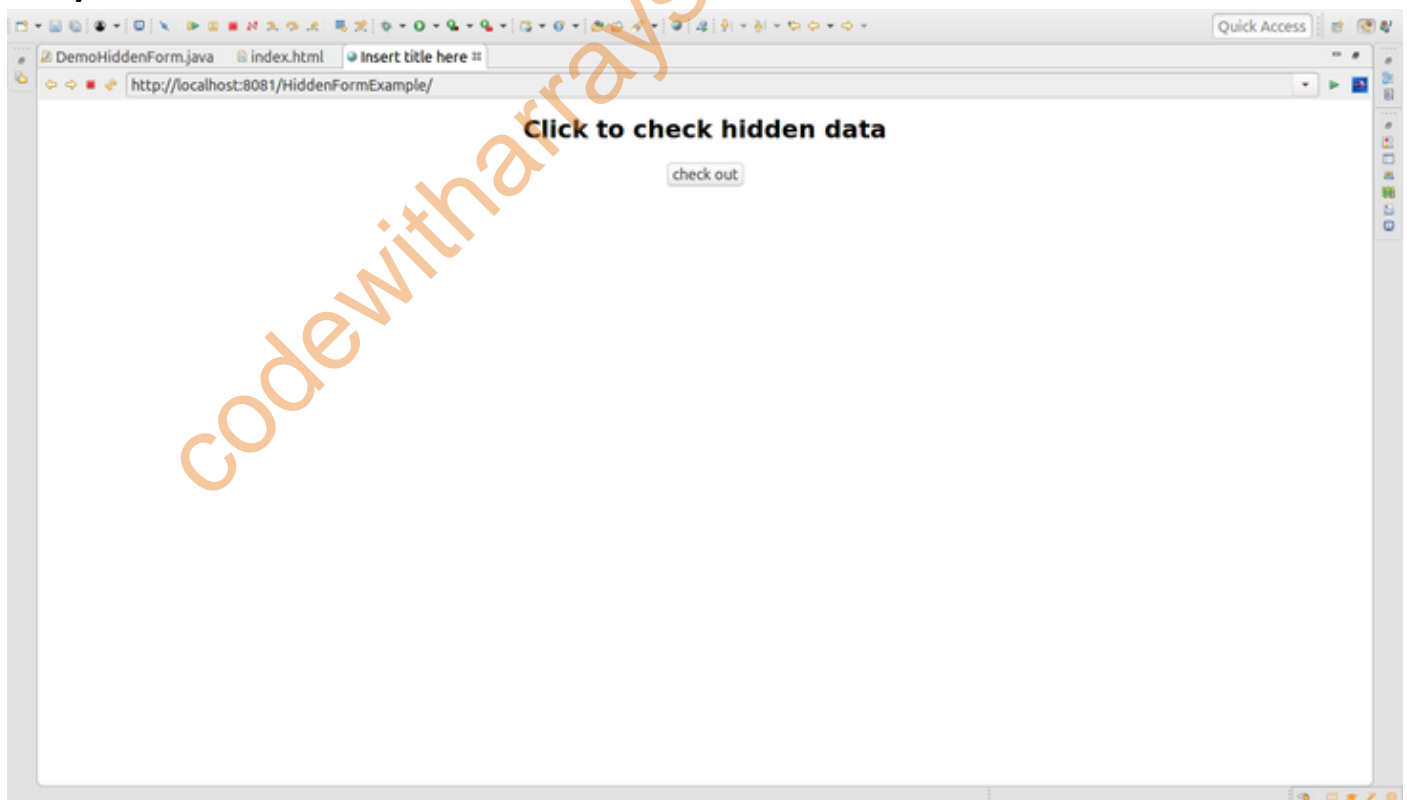
```
<center>
<h1>Click to check hidden data</h1>
<form action="serv" method="post">
<input type="hidden" name="website" value="TutorialandExample">
<input type="submit" value="check out">
</form>
</center>
```

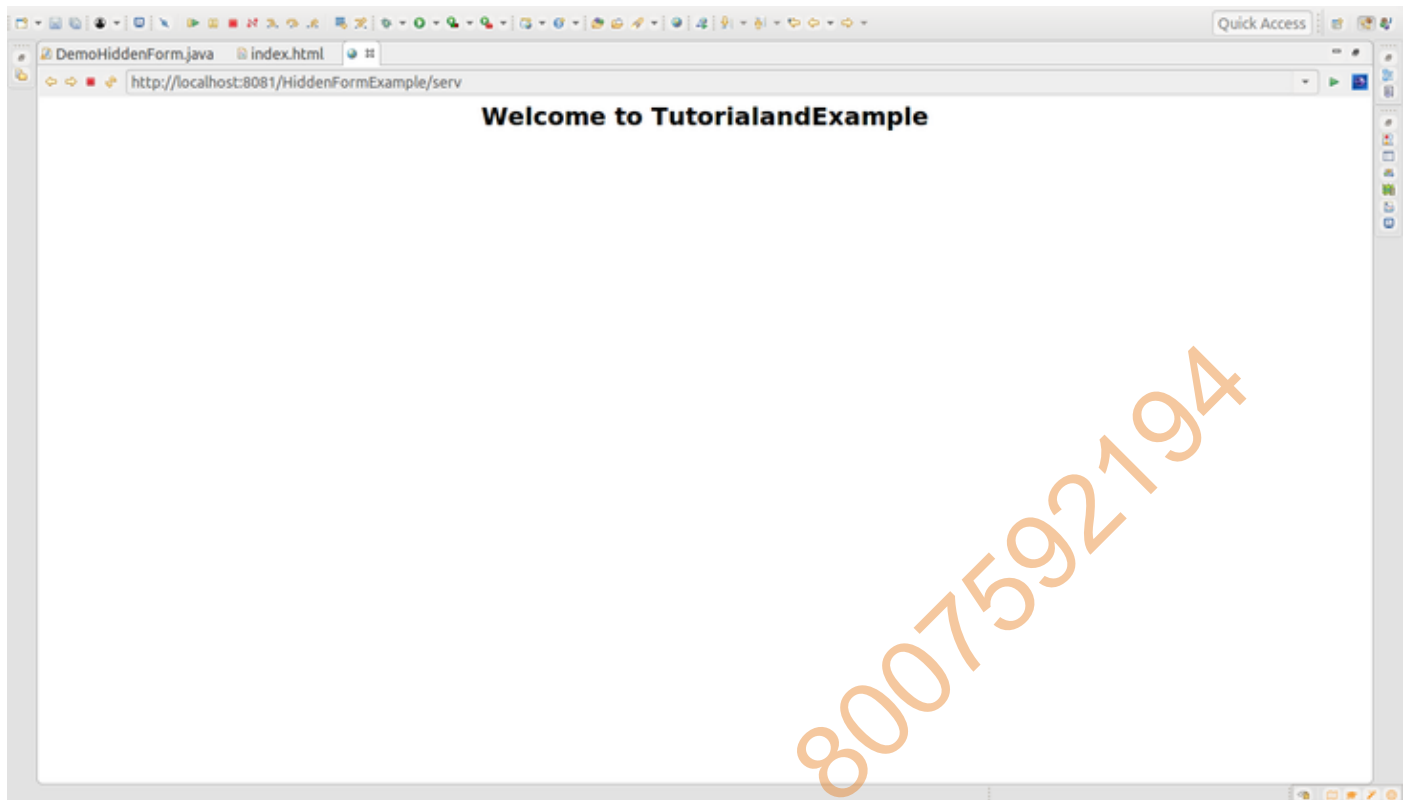
DemoHiddenForm.java

```
import java.io.*;
```

```
import javax.servlet.*;  
import javax.servlet.http.*;  
import javax.servlet.annotation.WebServlet;  
@WebServlet("/serv")  
public class DemoHiddenForm extends HttpServlet {  
    protected void doPost(HttpServletRequest req,  
        HttpServletResponse res) throws ServletException, IOException {  
        String web = req.getParameter("website");  
        res.setContentType("text/html");  
        PrintWriter out = res.getWriter();  
        out.println("<h1><center>Welcome to " + web+"</center></h1>");  
    }  
}
```

Output:





Asynchronous Servlet

In servlets, each request is handled by a thread. Thus the large amount of threads are required for heavy applications. Sometimes a situation arises when a thread waits for a resource or wait for an event. In this case thread may become blocked and degrades the performance of the servlet.

To overcome these situations servlet provides asynchronous support. This ensures that no threads associated with requests become idle.

Let's discuss about asynchronous interfaces and classes.

AsyncContext

AsyncContext is an interface present in javax.servlet package. The object of AsyncContext is used to perform various asynchronous operations and for that ServletRequest.startAsync() method is invoked.

Syntax used to enable asyncSupport:

`@WebServlet(urlPatterns="/pathname", asyncSupported=true)`

Methods of AsyncContext

Method	Description
--------	-------------

ServletRequest getRequest()	This method provides the request associated with AsyncContext when startAsync() method is invoked.
ServletResponse getResponse()	This method provides the response associated with AsyncContext when startAsync() method is invoked.
void dispatch()	Through this method we can dispatch the request and response of AsyncContext object to servlet container.
void setTimeout()	We can set a bound of time to AsyncContext by using this method.
long getTimeout()	It provides the time value if declared through setTimeout() method otherwise provide a default value of set by container.
void complete()	The purpose of this method it to complete asynchronous operation.



<https://www.youtube.com/@codewitharrays>



<https://www.instagram.com/codewitharrays/>



<https://t.me/codewitharrays> Group Link: <https://t.me/cceesept2023>



[+91 8007592194](tel:+918007592194) [+91 9284926333](tel:+919284926333)



codewitharrays@gmail.com



<https://codewitharrays.in/project>