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

SR.NO	Title
1	JSP Life Cycle
2	Scripting in JSP
3	JSP Script let Tag
4	JSP Declaration Tag
5	JSP Expression Tag
6	JSP Comments
7	JSP Directives
8	JSP Taglib Directive
9	JSP Control Statements
10	JSP Implicit Objects
11	JSP out
12	JSP Request
13	JSP Response
14	JSP Config
15	JSP Application
16	JSP Session
17	JSP Page Context
18	JSP Page
19	JSP exception
20	JSP Action Tags
21	JSP forward and include tag
22	JSP useBean , getProperty and setProperty tags
23	JSP Cookie Handling
24	JSP Expression Language
25	JSP EL Implicit Objects
26	JSTL – Java Standard tag library
27	JSTL Core Tags
28	JSTL Formatting Tags
29	JSTL SQL Tags
30	JSTL SQL Tags
31	JSTL XML Tags
32	JSTL function Tags
33	JSP Custom Tag

JSP Tutorial

Java Server Pages (JSP) is a server-side technology used to create static and dynamic web applications. The static content is expressed by text-based format files such as HTML, XML, SVG whereas JSP elements are used to construct dynamic content.

JSP is a convenient way of writing servlets. It enables you to insert Java code into HTML pages through simple JSP tags. JSP source files are represented by .jsp extension.

Difference between Servlet and JSP

JSP	Servlet
JSP files are represented by .jsp extension.	Servlet files are represented by .java extension.
JSP code can be inserted into HTML page.	Servlet code cannot be inserted into HTML page whereas vice-versa is possible.
JSP contains in-built implicit objects.	In Servlet, the required objects are called explicitly.
JSP is an easy language as it contains less code and deals maximum with tags.	Servlet contains a heavy bug of java code.
While compilation, JSP code is initially converted into Servlet. So, the execution of JSP page takes much more time than Servlet.	The execution of Servlet code is faster than JSP.

JSP Features

Some key features of JSP technology are as follows:

- **Portable** - JSP is a platform independent technology. So, JSP web pages can run on any browser and web container independently.
- **Simple**- It provides an easy way to develop and deploy web applications.
- **Powerful** - JSP is a Java based technology. Thus, it is secured and robust.
- **Tag based approach** - Instead of heavy bug of java code, JSP uses simple pre-defined tags.
- **Customized tag** - JSP also allows users to define their own tag.

JSP Life Cycle

JSP life cycle

The life cycle of JSP page internally implemented as a Servlet. These are the following stages through which a JSP page has to pass:

- **Translation** – This is an initial phase of JSP lifecycle that exist when first request of the JSP page is made. In this phase, web container translates the JSP page into servlet class.
- **Compilation** – As soon as the JSP page is translated, web container compiles the servlet class. Both translation and compilation have been proceeded only when JSP page's servlet is older than JSP page.
- **Loading and Instantiating** – Once the translation and compilation have been done, JSP page servlet follows the Servlet lifecycle. Hence, class is loaded and instance is created.
- **Execution** – In this phase, the actual task is performed. Web container invokes jspInit() method to initialize servlet instance. To pass request and response objects, web container invokes jspService() method.
- **Destruction** – This is the final phase of lifecycle in which web container invokes jspDestroy() method to remove JSP page servlet, if it is no more further required.

Features Some key features of JSP technology are as follows:

- **Portable** - JSP is a platform independent technology. So, JSP web pages can run on any browser and web container independently.
- **Simple**- It provides an easy way to develop and deploy web applications.
- **Powerful** - JSP is a Java based technology. Thus, it is secured and robust.
- **Tag based approach** - Instead of heavy bug of java code, JSP uses simple pre-defined tags.
- **Customized tag** – JSP also allows users to define their own tag.

Scripting in JSP

JSP scripting provides an easy and efficient way to insert Java programming language in JSP pages. Here, Java is a default language.

JSP also allows you to use any other scripting language that is capable to call Java objects. In this case, you have to specify that language in the page directive. The following syntax is used to implement it.

<%@ page language = “Scripting-language” %>

Disabling Scripting

It is not mandatory to use scripting in JSP page. So, JSP allows you to disable scripting if it is not required. Although, by default scripting is always enable.

Scriptlet elements

In JSP scripting, scriptlet elements perform the actual task. Hence, Java statements are enclosed within these elements.

JSP scripting elements are capable to create and manipulate Java objects, declaring variables and methods, catching Java Exceptions etc. It enables the JSP page to connect with database and fetch queries.

JSP provides three types of scripting elements to embed Java code in JSP page. Each scripting element has its own purpose. The following scripting elements are:

- Scriptlet tag
- Declaration tag
- Expression tag

JSP Scriptlet Tag

JSP scriptlet tag allows you to insert programming logic inside JSP page. Hence, you can put valid Java code within scriptlet tags. Each Java statement must be followed by a semicolon.

Syntax: - *<% Java code %>*

Whatever the code written inside scriptlet tag <% %> is compiled as Java code. This code can be accessible anywhere inside the JSP page. Any number of Java statements and local variables can be inserted in scriptlet tag. Although, scriptlet doesn't allow method declaration within it.

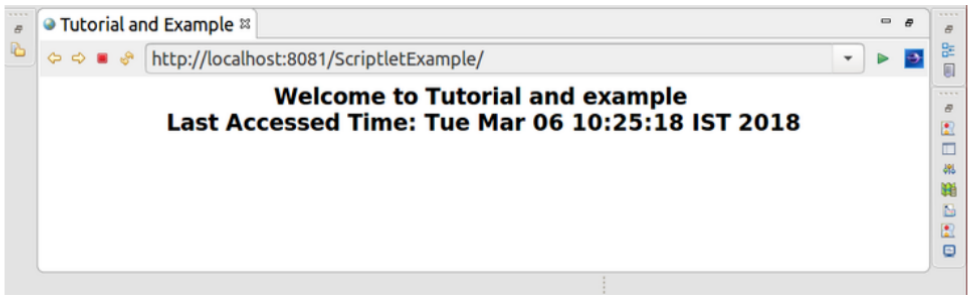
Example of JSP Scriptlet Tag

Here is a simple example of JSP scriptlet tag having a Java code within it.

index.jsp

```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<h2><center>Welcome to Tutorial and example
<%
java.util.Date date = new java.util.Date();
out.print("<br>Last Accessed Time: "+date);
%>
</center></h2>
</body>
</html>
```

Output:



Internal Procedure

Internally, JSP container inserts the content of scriptlet tag into the jspService() method.

Internal code:

```
public void jspService(HttpServletRequest request, HttpServletrResponse response) throws ServletException
{
PrintWriter out=response.getWriter();
response.setContentType("text/html");
java.util.Date date = new java.util.Date();
out.print("<br>Last Accessed Time: "+date);
}
```

JSP Declaration Tag

JSP declaration tag is used to declare variables and methods in JSP page. Unlike scriptlet tag, JSP container placed the content of declaration tag outside the jspService() method. So, variables and methods declared in declaration tag are static or instance.

The syntax of declaration tag is as follows:

<%! variable and method declaration %>

The variable and method inside declaration tag is up to the class level. We can put any number of variables and methods inside declarative tag.

Example of declaring variables in JSP Declaration Tag

Along with declaring variable, this example also shows that declaration tag doesn't get memory at each request.

index.jsp

```
<html>
<head>
<title>Insert title here</title>
</head>
<body>
<h1><center>
<%!
String name="Tutorial and Example";
int count=0;
%>
<%out.print("Website name:"+name);
out.print("<br>Number of time access:"+ ++count);
%>
</center></h1>
</body>
</html>
```

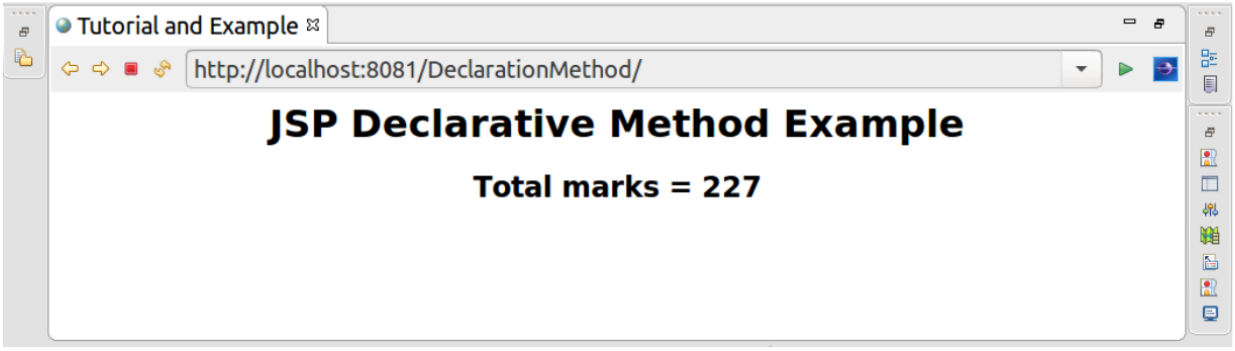
Output:

Example of declaring methods in JSP Declaration Tag

In this example, a method is declared inside declaration tag and has been accessing from scriptlet tag.

Index.jsp

```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<center>
<h1>JSP Declarative Method Example </h1>
<h2>
<%!
int marks(int english,int math,int science)
{
    return english+math+science;
}
%>
<%out.print("Total marks = "+marks(72,75,80));%>
</h2>
</center>
</body>
</html>
```



JSP Expression Tag

JSP expression tag allows you to place the result of Java expression in a convenient way. It can be seen as an alternative of out.print() method.

Thus, if we want to print any statement or result directly then we can use the below syntax:

```
<%=result%>
```

Note: Java statement in expression tag doesn't required a semicolon to terminate.

Example of JSP Expression Tag

In this example, we are just printing a string without using out.print() method.

```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<h1><center>
<%="Welcome to JSP tutorial" %>
</center></h1>
</body>
</html>
```

Output:

JSP Comments JSP provides separate tag for comment section. The content inside these tag is completely ignored by JSP engine. The syntax of comment tag is mentioned below. <%-- This is a comment section --%> In JSP, a tag inside another tag is not allowed. So, comment tag cannot be used inside the scriptlet elements. Instead of JSP comment, you can also use HTML comment.

JSP Comments

JSP provides separate tag for comment section. The content inside these tag is completely ignored by JSP engine.

The syntax of comment tag is mentioned below.

```
<%-- This is a comment section --%>
```

In JSP, a tag inside another tag is not allowed. So, comment tag cannot be used inside the scriptlet elements. Instead of JSP comment, you can also use HTML comment.

JSP Directives

In JSP, the role of directive is to provide directions to the JSP container regarding the compilation of JSP page. Directives convey information from JSP page to container that give special instruction at translation time.

The three types of directive provided by JSP is as follows: -

- Taglib Directive
- Include Directive
- Page Directive

Taglib Directive Taglib directive is used to define tag libraries in JSP page. It enables user to use custom tags within JSP file. A JSP page can contains more than one taglib directives. **Syntax of Taglib Directive** <@ taglib uri="uri" prefix="value"> Here, uri represents the path of tag library description and prefix represent the name of custom tag. **Example of Taglib directive** Here is a simple example of taglib directive. To run this code you need to add jstl jar file within lib directory of your project. It is better to run this code after learning JSTL. **index.jsp**

```
</html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<h2 align="center">
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:out value="Welcome to JSP Taglib Directive"/>
</h2>
</body>
</html>
```

Output: 

We will learn more about taglib directive in JSP custom tag.

JSP Taglib Directive

Taglib directive is used to define tag libraries in JSP page. It enables user to use custom tags within JSP file. A JSP page can contains more than one taglib directives.

Syntax of Taglib Directive

```
<@ taglib uri="uri" prefix="value">
```

Here, uri represents the path of tag library description and prefix represent the name of custom tag.

Example of Taglib directive

Here is a simple example of taglib directive. To run this code you need to add jstl jar file within lib directory of your project. It is better to run this code after learning JSTL.

index.jsp

```
</html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<h2 align="center">
<%@taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<c:out value="Welcome to JSP Taglib Directive"/>
</h2>
</body>
</html>
```

We will learn more about taglib directive in JSP custom tag.

JSP Include Directive

As the name implies, include directive is used to include the content of other files such as HTML, JSP, text into the current JSP page. These files are included during translation phase.

Syntax of include directive

<@ include file="file-name">

Include directive tag can be placed anywhere in JSP page.

Example of Include Directive

This example expresses the simple way to include more than one file of different types within JSP page through include directive.

header.html

```
<h1 align="center">
Tutorial and Example
</h1>
```

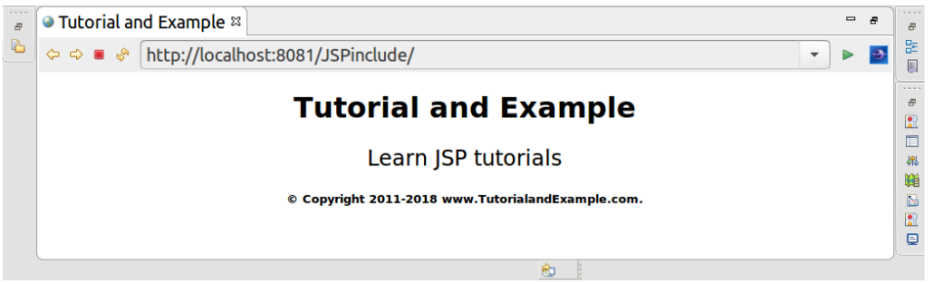
footer.jsp

```
<h5 align="center">
© Copyright 2011-2018 www.TutorialandExample.com.
</h5>
```

index.jsp

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Tutorial and Example</title>
</head>
<body>
<%@ include file="header.html"%>
<font size="5">
<p align="center">Learn JSP tutorials</p></font>
<%@ include file="footer.jsp"%>
</body>
</html>
```

Output:



JSP Page Directive

JSP page directive provides various attributes with unique properties. These attributes can be applied to an entire JSP page. The syntax of Page directive is as follows: -

```
<@page attribute="value">
```

Page directive tag can be placed anywhere inside the JSP page but placing it as the first statement is more preferable.

Attributes of Page Directive

These are the following attributes of page directive: -

language

This attribute defines the scripting language used in JSP page. By default, Java is used as a scripting language.

Syntax: -

```
<@page language="Scripting-language">
```

contentType

In servlets, we required to set the type of content on response attribute.

```
response.setContentType("text/html");
```

Like servlets, JSP use contentType attribute to define MIME type and character set of the response message. The default expression is: -

```
<@page contentType="text/html; charSet=UTF-8">
```

buffer

This attribute is use to buffer the response objects. Here, value represents the size of buffer. Instead of passing numeric value, it also allows to declare none. In this case, buffer uses its default size that is 8kb.

Syntax: -

```
<@page buffer="value">
```

info

In servlets, getServletInfo() method returns the information about the servlet. The similar role is played by info attribute in JSP. Hence, this attribute provides any type of description or information in a form of string.

Syntax: -

```
<@page info="value">
```

autoFlush

The purpose of this attribute is to flush the buffer automatically. For this, the value of autoFlush must be true.

Syntax: -

```
<@page autoFlush="true/false">
```

The default value of autoFlush is always true.

isThreadSafe

The role of isThreadSafe attribute is similar to SingleThreadModel interface in Servlet. Thus, it ensures that JSP handle only one type of request at a time.

Syntax: -

```
<@page isThreadSafe="true/false">
```

Here, the default value is true.

extends

The extends attribute inherits the superclass to serve its properties in current class. It is similar to extends keyword in Java.

```
<@page extends="package.class">
```

import

This attribute is used to import packages in JSP page. The package consists of similar type of classes and interfaces. It is similar to import keyword in Java.

```
<@page import="package-name">
```

session

Sessions are used to recognize the user. By default, the value of session is true in JSP. Hence, JSP always establish a session unless we make the value false.

```
<@page session="true/false">
```

errorPage

This attribute redirects the current page to JSP exception page if any exception occurs.

Syntax: -

```
<@page errorPage="value">
```

Here, URL of JSP exception page is passed.

isErrorPage

This attribute specifies that the current JSP page contain an error page. This error page can be utilized only when exception occur.

Syntax: -

```
<@page isErrorPage="true/false">
```

The default value of isErrorPage is always false.

isELIgnored

This attribute can be used to enable or disable the usage of Expression Language tags. By default, the value is true.

```
<@page isELIgnored="true/false">
```

Thus, if there is no requirement of Expression Language tag then it can be disabled from JSP page.

JSP Control Statements

There are various standard programming languages like C, C++, Java etc. that supports control statements. On the basis of that, JSP also follows the same methodology. Let's study the flow of control statements in Java Server Pages. Here, are some conditional statements about which we will discuss: -

- if else
- for loop
- while loop
- switch

IF...ELSE

JSP allows to use IF ELSE conditional statement as a programming logic of scriptlet tag. Apart from that, IF can also be used in many other ways like nested IF or standalone IF. Here is a simple example of IF ELSE statement.

index.jsp

```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<%! int num=8; %>
<% if(num%2==0)
{
    out.println("Number is even");
}
else
{
    out.println("Number is odd");
}
%>
</body>
</html>
```

For Loop

Here, is a simple example of for loop.

```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<%
for(int i=0;i<5;i++)
{
    for(int j=1;j<=i+1;j++)
    {
    out.print(j);
}
    out.print("<br>");
}
%>
</body>
</html>
```

While loop

Here, is a simple example of while loop.

```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<%
int i=0;
while(i<4)
{
    i++;
    out.print(i+"<br>");
}
%>
</body>
</html>
```

Switch Statement

Switch is used to maintain the flow of control statement. It contains various cases with one default case. The default case will execute only once, when none other case satisfies the condition. Here, is a simple example of switch statement.

index.jsp

```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<%! int weekday=4; %>
<%
switch(weekday)
{
case 1:
    out.print("Monday");
    break;
case 2:
    out.print("Tuesday");
    break;
case 3:
    out.print("Wednesday");
    break;
case 4:
    out.print("Thursday");
    break;
case 5:
    out.print("Friday");
    break;
case 6:
    out.print("Saturday");
    break;
default:
    out.print("Sunday");
    break;
}
%>
</body>
</html>
```

JSP Implicit Objects

In JSP, implicit objects are pre-defined objects, created by the web container. These objects are created during the translation phase from JSP to Servlet. JSP allows you to call these objects directly without initializing them.

Implicit objects are required to be declared in scriptlet tags only. Hence, in translation phase these objects are embedded within service() method of servlet.

Note: - Servlet provide these objects to enhance the ease of coding in JSP.

List of JSP implicit objects

JSP provides 9 implicit objects that can be used in JSP pages. Each object represents some unique identity. Here, is the list of all implicit objects with their classes.

Object	Classes
out	javax.servlet.jsp.JspWriter
request	javax.servlet.http.HttpServletRequest
response	javax.servlet.http.HttpServletResponse
config	javax.servlet.ServletConfig
application	javax.servlet.ServletContext
session	javax.servlet.http.HttpSession
exception	javax.servlet.http.HttpException
Page	java.lang.object
PageContext	javax.servlet.jsp.PageContext

We will learn more about each object with simple examples.

JSP out

In JSP, out is an implicit object that is associated with the response object. This object is used to write content to the output stream.

Methods of JSP out

Following are some important methods of JSP out: -

Method	Description
void print()	This is one of the most frequently used method. It prints the statement.
void clear()	This method is used to remove the content of buffer.
boolean isAutoFlush()	This method specifies whether the buffer is flushed or not.
int getBufferSize()	This method returns the size of buffer in bytes.
int getRemaining()	This method returns the unused size of buffer in bytes.

Example of JSP out

This example shows the functionality of all the above methods on JSP out object.

index.jsp

```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<%
String s="Tutorial and Example";
out.println("Learn JSP");
out.clear();
Boolean b=out.isAutoFlush() ;
out.println("<br><h2 align=center>IsAutoFlush:"+b);
out.println("<br>Website:"+s);
int i1=out.getBufferSize();
out.println("<br>Buffer Size:"+i1);
int i2=out.getRemaining();
out.println("<br>Remaining Size:"+i2+"</h2>");
%>
</body>
</html>
```

JSP Request

The request object is used to fetch the user's data from the browser and pass this data to the server. The working of JSP request object is similar to the servlet's HttpServletRequest interface object.

Example of JSP Request

In this example, request object fetches the form data.

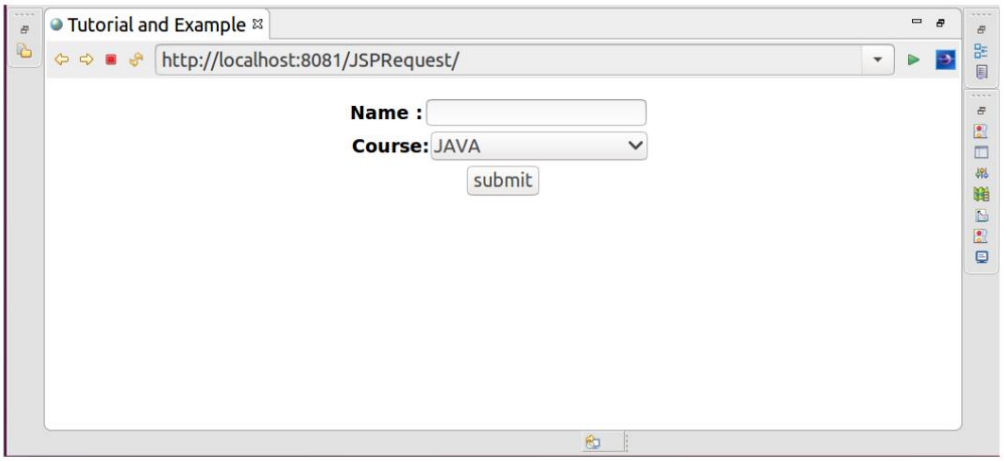
index.html

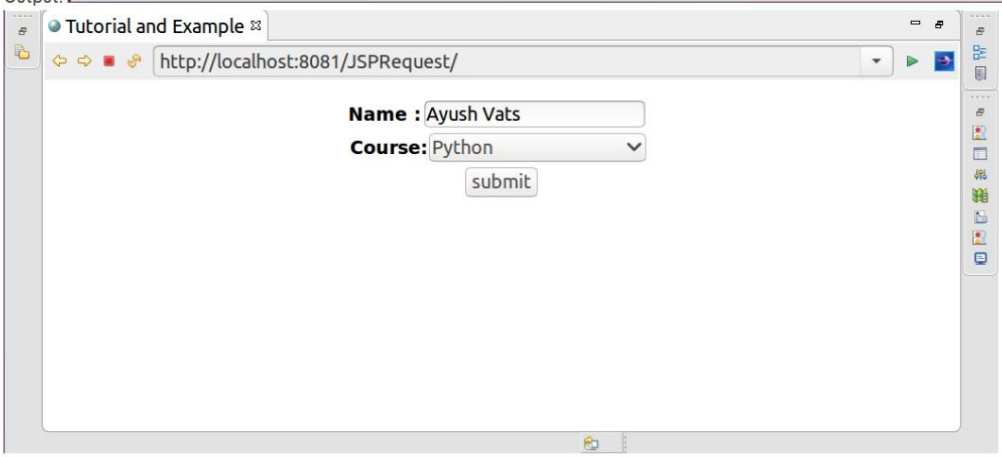
```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>Tutorial and Example</title>
</head>
<body>
<h3 align="center">
<form action="index.jsp">
Name :<input type="text" name="name"> <br>
Course:<select name="Course" style="width:230px">
<option value="JAVA">JAVA</option>
<option value="PHP">PHP</option>
<option value="Python">Python</option>
<option value="Other">Other</option>
</select> <br>
<input type="submit" value="submit">
</form>
</h3>
</body>
</html>
```

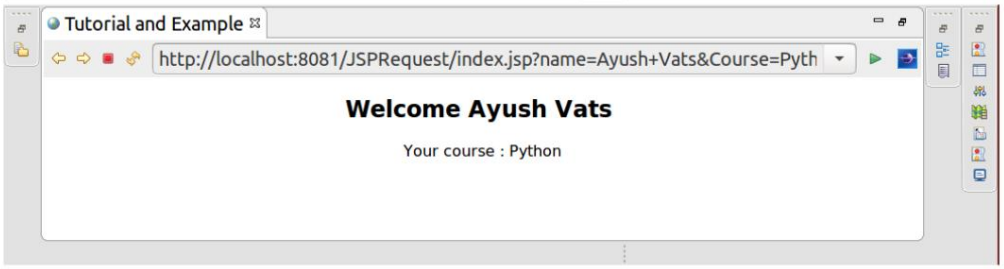
index.jsp

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Tutorial and Example</title>
</head>
<body>
<center>
<%
String str1=request.getParameter("name");
String str2=request.getParameter("Course");
%>
<h2>
<%="Welcome "+str1%> <br>
</h2>
<%="Your course : "+str2 %>
</center>
</body>
</html>
```

Output:


Tutorial and Example
http://localhost:8081/JSPRequest/
Name :
Course: JAVA
submit


Tutorial and Example
http://localhost:8081/JSPRequest/
Name : Ayush Vats
Course: Python
submit


Tutorial and Example
http://localhost:8081/JSPRequest/index.jsp?name=Ayush+Vats&Course=Pyth
Welcome Ayush Vats
Your course : Python

JSP response

Basically, JSP response object is an instance of servlet's HttpServletResponse interface. It is used to resolve client request.

Response object can be used to perform various functions such as encode URL, add cookies, add headers, send errors, set content type etc.

Example of JSP Response

In this example, user's password is set in index.jsp file. If the password entered in the form is same then valid.jsp file will execute. If entered password is unmatched then invalid.jsp file will execute.

index.html

```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<form action="index.jsp">
Name:<input type="text" name="name"> <br>
Password:<input type="password" name="pass"> <br>
<input type="submit" name="submit">
</form>
</body>
</html>
```

index.jsp

```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<%
String name=request.getParameter("name");
String pass=request.getParameter("pass");
if(pass.equals("abcd"))
{
    response.sendRedirect("valid.jsp");
}
else
{
    response.sendRedirect("invalid.jsp");
}
%>
</body>
</html>
```

Valid.jsp

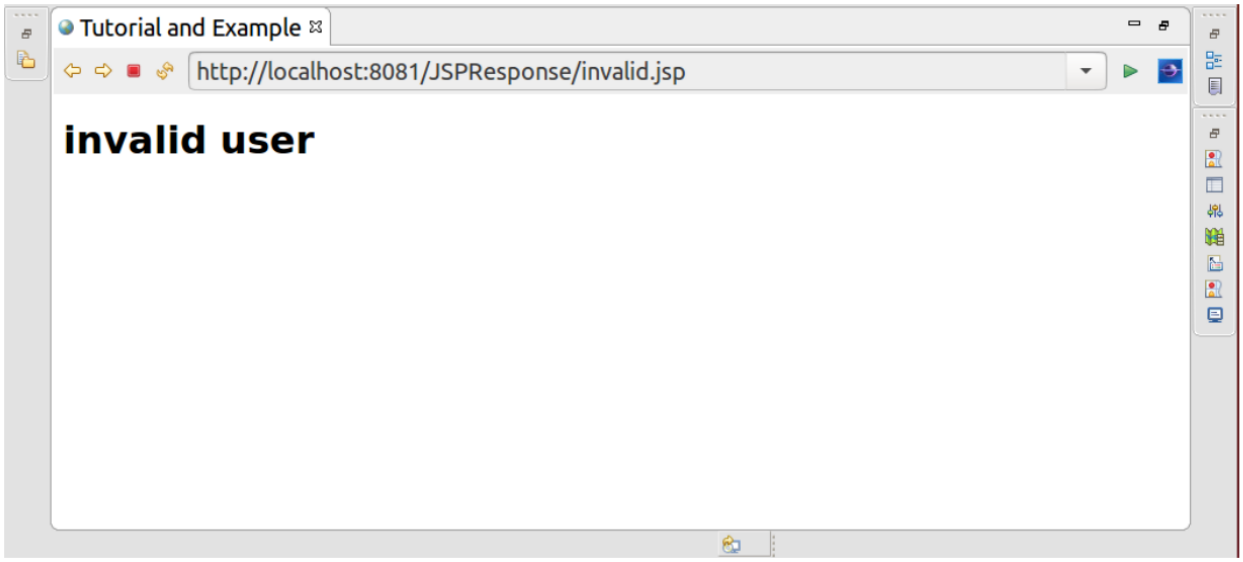
```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<h1>
<%="Valid user"%>
</h1>
</body>
</html>
```

invalid.jsp

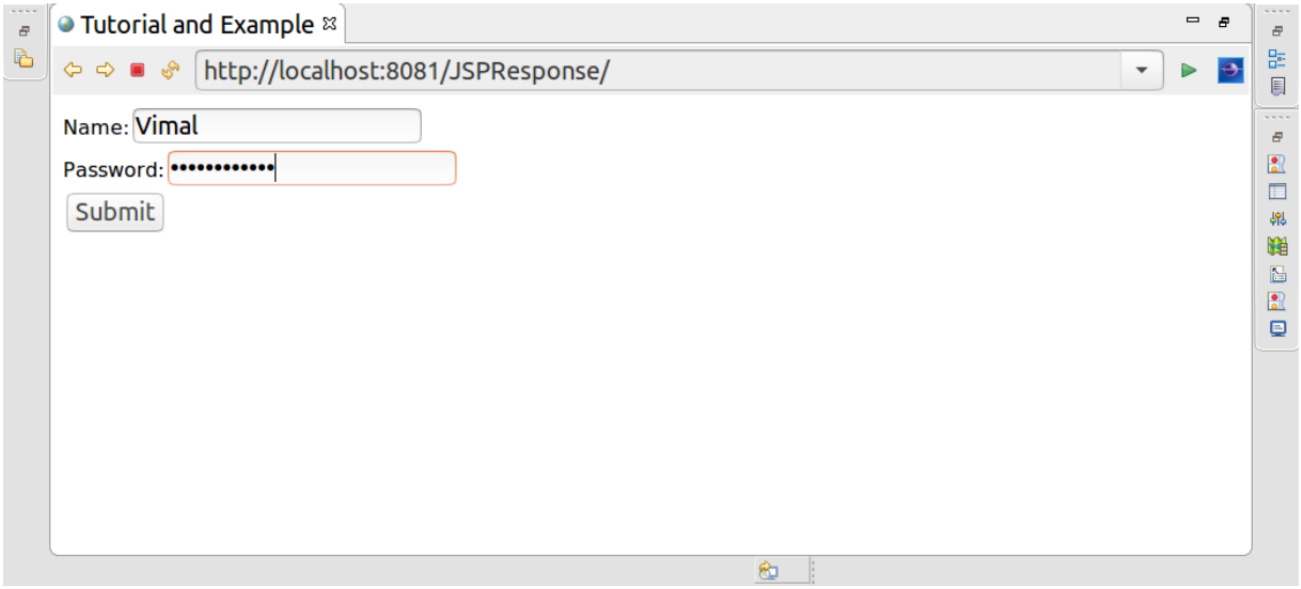
```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<h1>jsp response object method
<%="invalid user" %>
</h1>
</body>
</html>
```

Here, we already set "abcd" as password. So, if the entered password is correct then only the user is valid.

Output:

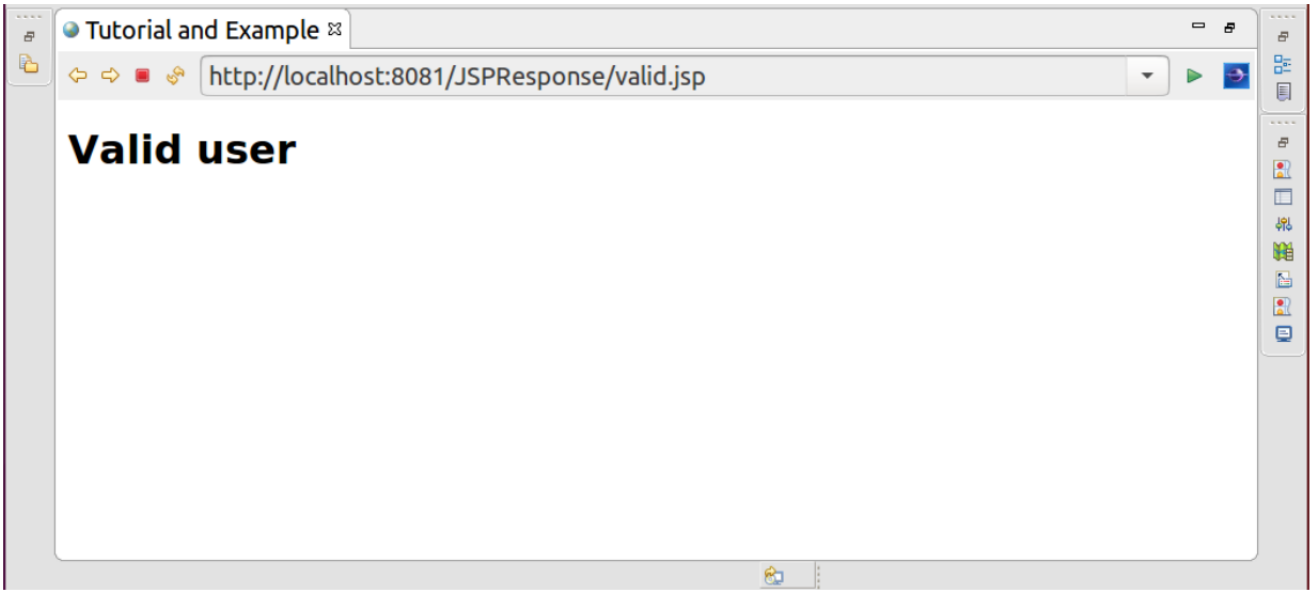


Output:



Here, w

already set "abcd" as password. So, if the entered password is correct then only the user is valid.



JSP config

JSP config object is used to send the configuration information to JSP page. Here, config is an instance of servlet's ServletConfig interface.

In this case, the information is send to JSP page through web.xml file. Config object is used to fetch this information. It is used widely for the initialization of parameters.

Note: - The scope of JSP config object is up to a single JSP page only.

Example of JSP Config

In this example, web.xml file contains the information. The config object in index.jsp file fetches that information.

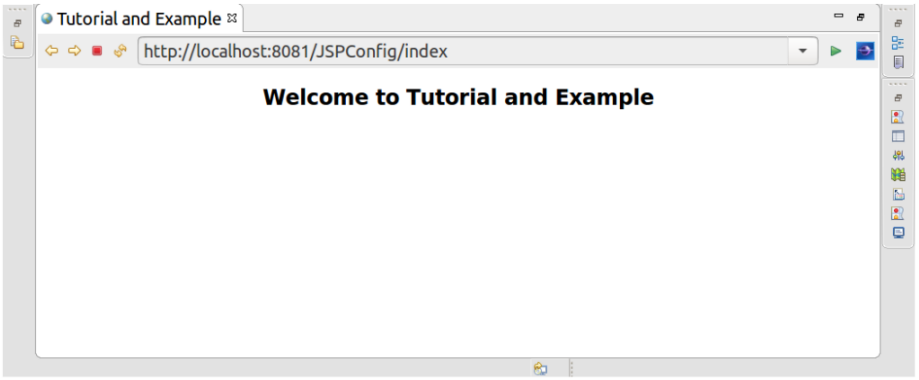
index.jsp

```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<%
String name=config.getServletName();
out.print("<h2 align=center>Welcome to "+name+"</h2>");
%>
</body>
</html>
```

web.xml

```
<web-app>
<servlet>
<servlet-name>Tutorial and Example</servlet-name>
<jsp-file>/index.jsp</jsp-file>
</servlet>
<servlet-mapping>
<servlet-name>Tutorial and Example</servlet-name>
<url-pattern>/index</url-pattern>
</servlet-mapping>
</web-app>
```

Output:



JSP application

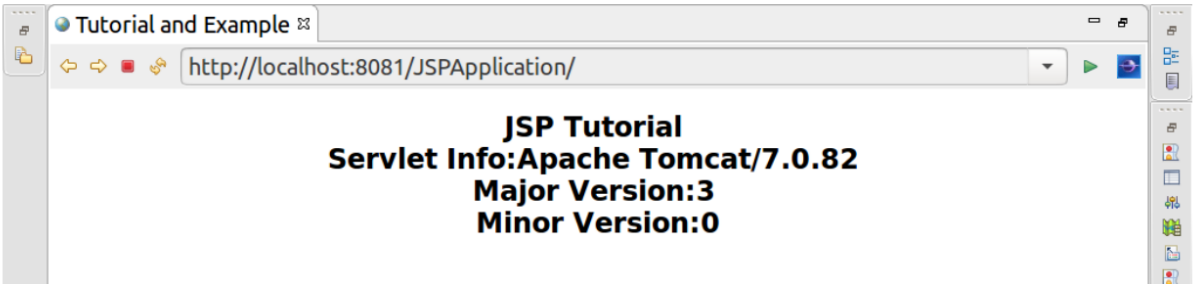
JSP application object is an instance of servlet's ServletContext interface. It is used initialize parameter of one or more JSP pages in an application. Thus, the scope of application object is wider than config object.

Example of JSP Application

This example shows the functionality of various methods of application object.

index.jsp

```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<%
String tut="JSP Tutorial";
application.setAttribute("var",tut);
String fetch=(String)application.getAttribute("var");
String info=application.getServerInfo();
int majversion=application.getMajorVersion();
int minversion=application.getMinorVersion();
%>
<h2 align="center">
<%=fetch %>
<br>
<%="Servlet Info:"+info %>
<br>
<%="Major Version:"+majversion %>
<br>
<%="Minor Version:"+minversion %>
</h2>
</body>
</html>
```



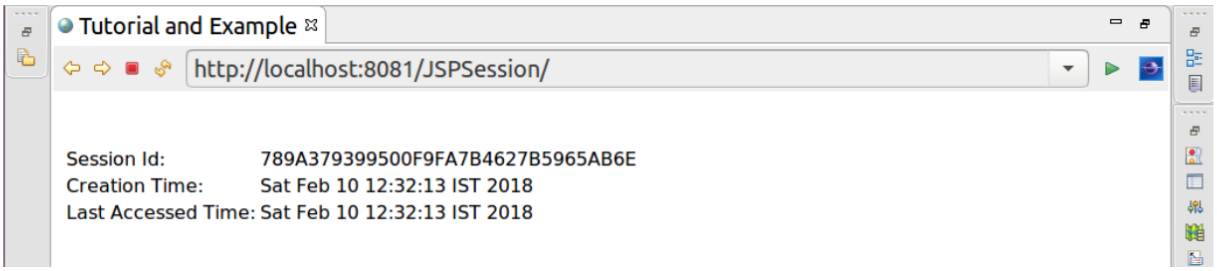
JSP session

In JSP, session object is used to establish a connection between a client and a server. It maintains the state between them so that server recognize the user easily.

The session object is an instance of servlet's HttpSession interface. This object contains the user's information and maintains the state till this session is active.

index.jsp

```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<%
String id=session.getId();
Date d1=new Date(session.getCreationTime());
Date d2=new Date(session.getLastAccessedTime());
%>
<table>
<%="<tr><td>Session Id:</td><td>"+id+"</td></tr>"%>
<br>
<%="<tr><td>Creation Time:</td><td>"+d1+"</td></tr>"%>
<br>
<%="<tr><td>Last Accessed Time:</td><td>"+d2+"</td></tr>"%>
</table>
</body>
</html>
```



JSP pageContext

The object pageContext is used to access all the information of JSP page. It is an instance of javax.servlet.jsp.PageContext.

The scope of pageContext can be defined explicitly. The object is valid up to the specified scope only. Several layers in which pageContext can be accessed are as follows: -

- SESSION_SCOPE
- REQUEST_SCOPE
- PAGE_SCOPE
- APPLICATION_SCOPE

Methods of pageContext

The various methods of pageContext are as follows: -

- void setAttribute(String attribute_name,Object attribute_value,int scope):- This method is used to set the attribute.
- Object getAttribute(String attribute_name,int scope):- This method return the attribute of object type.
- Object removeAttribute(String attribute_name,int scope):- This method is used to remove the attribute.
- Object findAttribute(String attribute_name):- This method search the attribute passed within its bracket.

Example of pageContext

This is a simple example that defines the way of set and get the various attributes.

index.jsp

```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<h1 align="center">Tutorials:</h1>
<h2 align="center">
<%
pageContext.setAttribute("Tutorial1","HTML",PageContext.SESSION_SCOPE);
pageContext.setAttribute("Tutorial2","Servlet",PageContext.REQUEST_SCOPE);
pageContext.setAttribute("Tutorial3","JSP",PageContext.PAGE_SCOPE);
pageContext.setAttribute("Tutorial4","JavaScript",PageContext.APPLICATION_SCOPE);
String name1=(String)pageContext.getAttribute("Tutorial1",PageConteOutput xt.SESSION_SCOPE);
String name2=(String)pageContext.getAttribute("Tutorial2",PageContext.REQUEST_SCOPE);
String name3=(String)pageContext.getAttribute("Tutorial3",PageContext.PAGE_SCOPE);
String name4=(String)pageContext.getAttribute("Tutorial4",PageContext.APPLICATION_SCOPE);
out.println(name1);
out.println("<br>"+name2);
out.println("<br>"+name3);
out.println("<br>"+name4);
%>
</h2>
</body>
</html>
```

JSP page

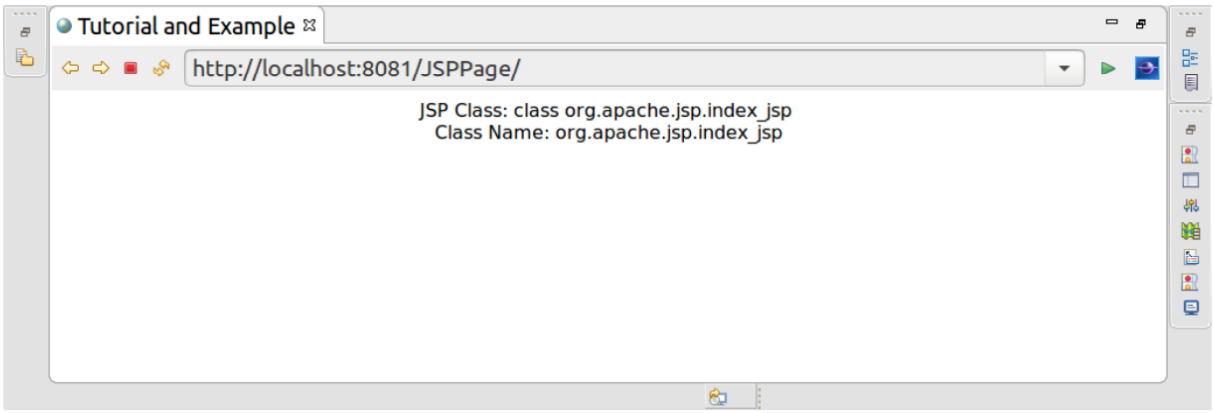
In JSP, the purpose of page object is to create the servlet instance. The role of page object is similar to this keyword in Java.

The page object represents the entire JSP page. It is an instance of java.lang.Object class. In JSP, the utilization of page object is very rare.

Example of JSP Page

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Tutorial and Example</title>
</head>
<body>
<center>
JSP Class: <%=page.getClass()%>
<br>
Class Name: <%=page.getClass().getName() %>
</center>
</body>
</html>
```

Output:



JSP exception

Like Java, exceptions can also be occurred in JSP. To handle these exceptions, JSP exception object is used.

Handling exceptions in JSP using exception object is much easier approach. This object is needed to be declared in a separate exception page. The exception page path is provided to the current page within page directive.

However, try catch block can also be used to handle these exceptions.

Example of exception

In this example, we use exception object to handle the exceptions. The result is generated on the basis of division of the provided numbers. If inappropriate number is given in the input then an exception occurs.

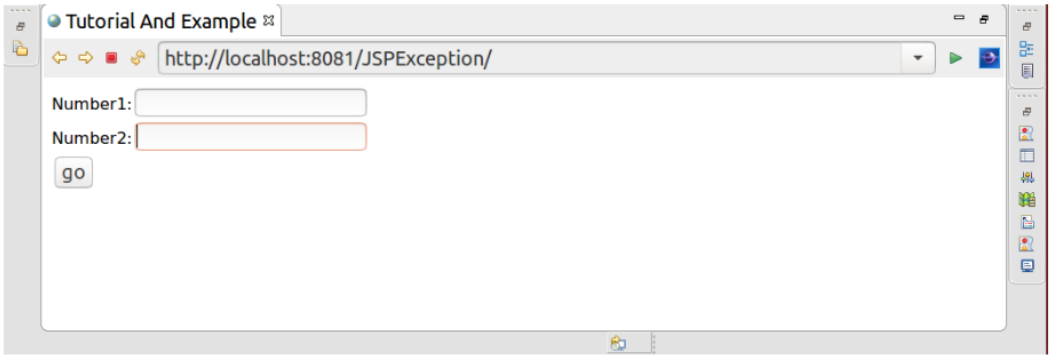
index.jsp

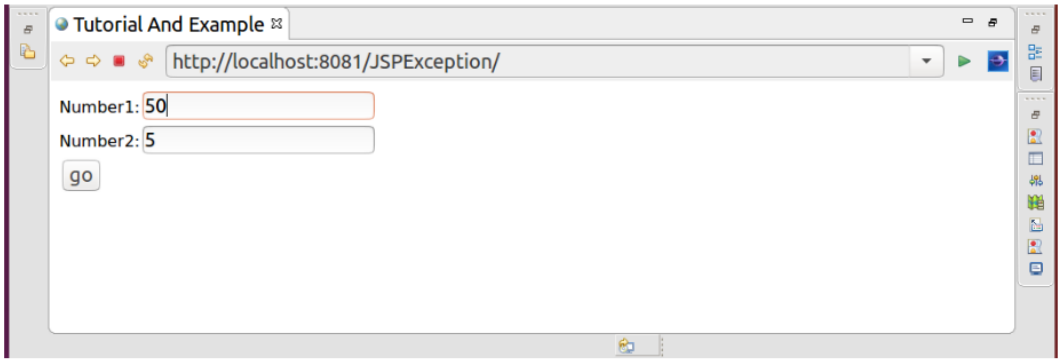
```
<form action="fetch.jsp">
Number1:<input type="text" name="num1"> <br>
Number2:<input type="value" name="num2"> <br>
<input type="submit" value="go">
</form>
```

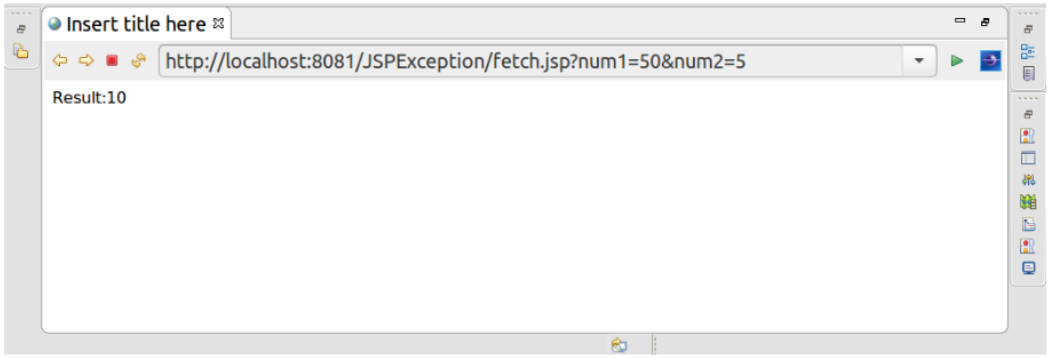
fetch.jsp

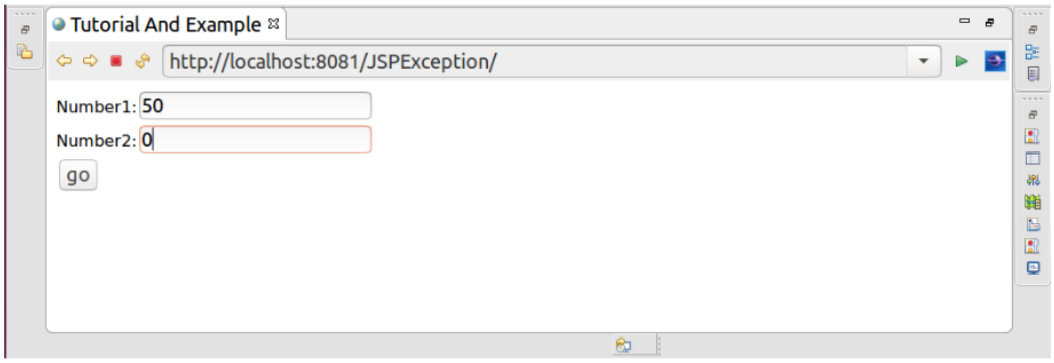
```
<%@ page errorPage="Errorgen.jsp"%>
<%
String s1=request.getParameter("num1");
String s2=request.getParameter("num2");
int i1=Integer.parseInt(s1);
int i2=Integer.parseInt(s2);
int sum=i1/i2;
out.println("Result:"+sum);
%>
Errorgen.jsp
<%@ page isErrorPage="true"%>
<html>
<body>
Exception Occurs: <%=exception%>
</body>
</html>
```

Output:


Tutorial And Example
http://localhost:8081/JSPException/
Number1:
Number2:
go


Tutorial And Example
http://localhost:8081/JSPException/
Number1: 50
Number2: 5
go


Insert title here
http://localhost:8081/JSPException/fetch.jsp?num1=50&num2=5
Result:10


Tutorial And Example
http://localhost:8081/JSPException/
Number1: 50
Number2: 0
go

JSP Action Tags

As the name implies, JSP action tags are used to perform various actions during the execution of JSP page. Here, each action performs some specific task. These tasks include:

- Forwarding the current page request to another page.
- Including external page to JSP page.
- Creating a JavaBean instance.

JSP provides additional functionality to JSP pages through these action tags. Unlike directive tags, action tag conveys the information to servlet container when request is being processed.

Syntax

```
<jsp:action_name attribute="value">
```

List of JSP Action Tags

These are the various action tags provided by JSP: -

- **jsp:forward** – This tag is used to forward the request of current JSP page to any another JSP, Servlet or HTML page.
- **jsp:include** – This tag includes the external resource to JSP page.
- **jsp:useBean** – This tag deals with the object of JavaBean.
- **jsp:setProperty** – This tag sets the property of JavaBean object.
- **jsp:getProperty** – This tag gets the property associated with JavaBean object.
- **jsp:text** – This tag is used to write template text in JSP page.
- **jsp:elements** – This tag is to define XML elements dynamically.
- **jsp:attribute** – This tag defines the attribute of dynamically generate XML elements.
- **jsp:body** – This tag defines the body of dynamically generate XML elements.
- **jsp:plugin** – This tag is used to integrate Java components with JSPpage.

JSP forward and include tag

In JSP, forward and include tags are the most frequently used action tags. Unlike Servlets, there is no need to create object of RequestDispatcher interface to use the functionality of forward and include. Thus, JSP provides direct tag for the same purpose.

JSP forward Tag

In JSP, forward tag terminates the execution of current JSP page and forward the control to new page which can be JSP, Servlet, HTML or any other. The response of new page is displayed to the user.

Syntax: -

```
<jsp:forward page="filename">
```

JSP include Tag

The role of include tag is to dispatch the external resource in the JSP page. The external resource can be JSP, Servlet HTML or any other page.

Syntax: -

```
<jsp:include page="filename">
```

Example of forward and include Tag

This example represents the functionality of both, forward and include tag.

index.jsp

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Tutorial and Example</title>
</head>
<body>
<font size="5">
<form action="CheckPage.jsp">
Name:<input type="text" name="name"> <br>
Password:<input type="password" name="pass"> <br>
<input type="submit" value="submit">
</form>
</font>
</body>
</html>
```

CheckPage.jsp

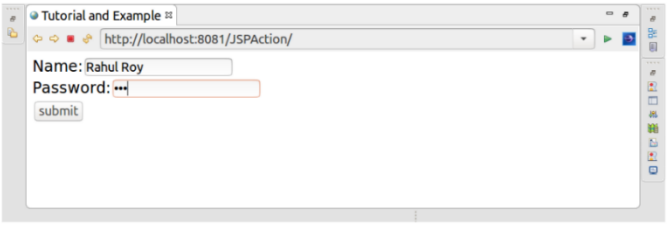
```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Tutorial and Example</title>
</head>
<body>
<font size="5">
<% String s=request.getParameter("pass");
if(s.length()>4)
{
%>
<jsp:forward page="WelcomePage.jsp"></jsp:forward>
<%
}
else
{
    %>
    <jsp:forward page="Error.jsp"></jsp:forward>
<%
}
%>
</font>
</body>
</html>
```

WelcomePage.jsp

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Tutorial and Example</title>
</head>
<body>
<font size="5">
<% String s=request.getParameter("name"); %>
<%="Welcome : "+s %>
</font>
</body>
</html>
```

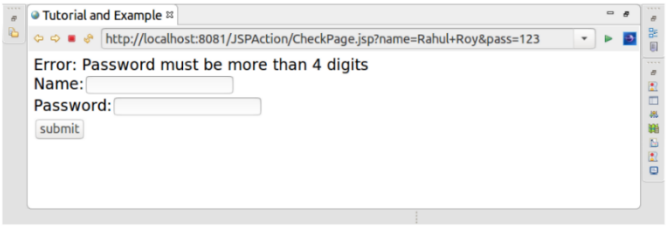
Error.jsp

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Tutorial and Example</title>
</head>
<body>
<font size="5">
Error: Password must be more than 4 digits <br>
<jsp:include page="index.jsp"></jsp:include>
</font>
</body>
</html>
```

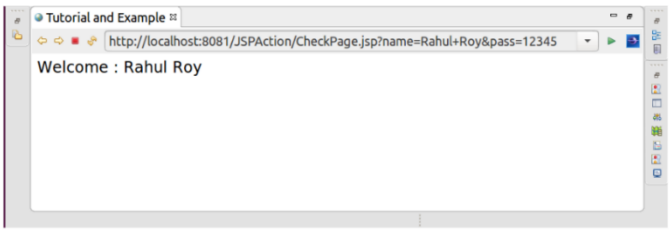


Output

The entered password must be more than 4 digits.



Enter a strong password (i.e. more than 4 digits) to see welcome page.



JSP useBean, getProperty and setProperty tags

Java bean is a special type of Java class that contains private variables and public setter and getter methods. Java bean class implements Serializable interface and provides default constructor.

JSP allows you to access the Java beans in JSP files. To invoke Java bean, useBean action tag is used and to access the properties of Java beans, JSP setter and getter method is used.

Example of useBean Tag

In this example, setproperty() and getproperty() method is used to set and get data of current Java class.

EmployeeBean.java

```
package com.tutorialandexample;
import java.io.Serializable;
public class EmployeeBean implements Serializable {
    private String name;
    private int id;
    private int age;
    public String getName() {
        return name;
    }
    public void setName(String name) {
        this.name = name;
    }
    public int getId() {
        return id;
    }
    public void setId(int id) {
        this.id = id;
    }
    public int getAge() {
        return age;
    }
    public void setAge(int age) {
        this.age = age;
    }
}
```

index.jsp

```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<jsp:useBean id="emp" class="com.tutorialandexample.EmployeeBean">
<jsp:setProperty property="name" name="emp" value="Sushant Singh"/>
<jsp:setProperty property="id" name="emp" value="101"/>
<jsp:setProperty property="age" name="emp" value="23"/>
</jsp:useBean>
Employee ID : <jsp:getProperty property="id" name="emp"/> <br>
Employee Name : <jsp:getProperty property="name" name="emp"/> <br>
Employee Age : <jsp:getProperty property="age" name="emp"/>
</body>
</html>
```

JSP Cookie Handling

Cookie is a small piece of information sent by server to recognize the user. This information is stored at client side in the textual form.

In JSP, cookie is an object of javax.servlet.http.Cookie class. This object is used widely for session management purposes.

Methods of JSP Cookie

Some of the important method of cookie are as follows: -

- void setMaxAge(int expiry): - This method is used to set the expiry time of cookie. Here, time is represent in the form of seconds.
- void getMaxAge(): - This method returns the declared time of cookie after which it will expire.
- String getName(): - This method returns the name of the cookie.
- void setValue(String value): - This method is used to set associate the value with cookie.
- String getValue(): - This method returns the value associated with the cookie.

Example of JSP Cookie assessment

Index.jsp

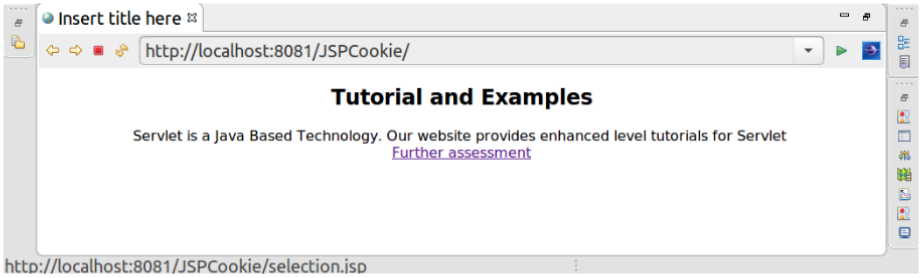
```
<html>
<body>
<%
String opt="Servlet";
Cookie[] ck=request.getCookies();
if(ck!=null)
{
    for(Cookie temp:ck)
    {
        if("technology".equals(temp.getName()))
        {
        opt=temp.getValue();
        break;
    }
}}
%>
<center>
<h2>Tutorial and Examples</h2>
<%=opt+" is a Java Based Technology." %>
<%="Our website provides enhanced level tutorials for "+opt %>
<br>
<a href="selection.jsp">Further assessment</a>
</center>
</body>
</html>
```

Selection.jsp

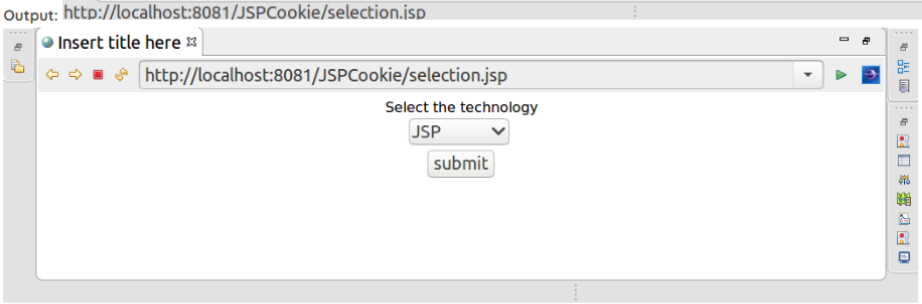
```
<html>
<body>
<center>
Select the technology
<form action="cookiepage.jsp">
<select name="tech">
<option>Servlet</option>
<option>JSP</option>
<option>JavaScript</option>
</select>
<br>
<input type="submit" value="submit">
</form>
</center>
</body>
</html>
```

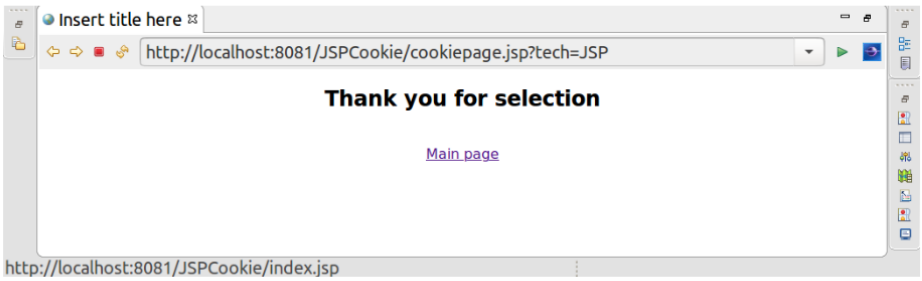
Cookiepage.jsp

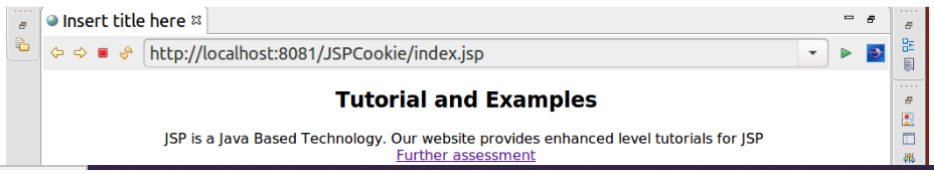
```
<html>
<body>
<center>
<%
String s=request.getParameter("tech");
Cookie c=new Cookie("technology",s);
c.setMaxAge(60);
response.addCookie(c);
%>
<h2>Thank you for selection</h2> <br>
<a href="index.jsp">Main page</a>
</center>
</body>
</html>
```



Output:







JSP Expression Language

JSP introduced the concept of expression language (EL) in JSP 2.0 version. The purpose of expression language is to access and manipulate data easily without using any type of scriptlet.

Why Expression language?

Before JSP 2.0, scriptlets are used to handle the business logic. But it is difficult task for web designers to understand and code the java part in scriptlets. Hence, now expression language is used to overcome this difficulty.

Syntax

This is the simple syntax of expression language.

```
${expression}
```

Expression Language Operator

JSP expression language provides an easy way to perform various arithmetic, relational, logical operations. These are some of the frequently used operators: -

- Arithmetic operators – It contains various mathematical operators such as addition (+), subtraction (-), division (/ or div), module (% or mod), multiplication (*).
- Logical operators – It contains operators like && (and), || (or).
- Relational operators – It contains operators like < (less than), > (greater than), <= (less than or equal to), >= (greater than or equal to).

Example of JSP Expression Language Operator

This is a simple example of JSP operators that perform various mathematical operations between two numbers (10 & 5) directly.

Index.jsp

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<title>Insert title here</title>
</head>
<body>
<h2>Arithmetic Operators : 10&5 </h2>
Addition :
${10+5}
<br>
Subtraction :
${10-5}
<br>
Multiplication :
${10*5}
<br>
Division :
${10/5}
<br>
<h2>Relational Operators : 10&5 </h2>
Equality Check (==) :
${10==5}
<br>
Non-Equality Check (!=) :
${10!=5}
<br>
Less number check (<) :
${10<5}
<br>
Greater number check (>) :
${10>5}
</body>
```

JSP EL Implicit Objects

Apart from operators, JSP also provide expression language implicit objects. The role of these implicit objects is to get the attribute associated with various object. Here, is the list of expression language implicit objects with their purposes.

Implicit Object	Description
requestScope	Fetches the attribute associated with request object.
sessionScope	Fetches the attribute associated with session object.
pageScope	It provides the attribute associate within pageScope.
applicationScope	Fetches the attribute associated with application object.
param	It retrieves the single value associated with request parameter.
paramValues	It retrieves the array of values associated with request parameter.
header	It retrieves the single value associated with header parameter.
headerValues	It retrieves the array of values associated with header parameter.
cookie	It is used to get the value of cookies.
initParam	It is used to map an initialize parameter.
pageContext	The role of this object is similar to JSP pageContext implicit object.

Example of Expression Language Implicit Object

This example shows the way of using various expression language implicit objects.

index.jsp

```
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Tutorial and Example</title>
</head>
<body>
<%
request.setAttribute("Tutorial", "Java");
application.setAttribute("Tutorial", "Python");
request.getSession().setAttribute("Tutorial", "PHP");
%>
<center>
Host name:
${header["host"]} <br>
Path name:
${pageContext.request.contextPath} <br>
Tutorials: <br>
${requestScope.Tutorial} <br>
${applicationScope.Tutorial} <br>
${sessionScope.Tutorial} <br>
</center>
</body>
</html>
```

JSTL - Java Standard Tag Library

Java Standard tag library (JSTL) is a collection of various type of JSP tags. Each tag control the behavior of JSP page and perform specific task such as database access, conditional execution, internationalization etc.

JSTL delivers the functionality of programming methodology without using Java code. Thus, it provides ease to develop and maintain web applications.

Types of JSTL tag

In JSP, JSTL tags are divided into following parts: -

- JSTL Core
- JSTL Formatting
- JSTL SQL
- JSTL XML
- JSTL function

To implement JSTL tag in JSP page, it is required to add JSTL jar within lib directory of your project.

JSTL Core Tags

In JSTL, core tags are most frequently used tags that provides variable support, manages URL and control the flow of JSP page. To use JSTL it is required to associate the below URL in your JSP page: -

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
```

It is not mandatory to use 'c' in the prefix of JSTL core tag. You can also use any other character or word. But as a convention, it is better to use it.

Core Tags

Here, is the list of JSTL core tags: -

- <c:out> tag: - This tag is used to print statements or results. It works as an alternative of <%=expression %> tag.
- <c:set> tag: - This tag is used to declare the variable with its value and scope.
- <c:remove> tag: - This tag is used to remove the attributes.
- <c:if> tag: - This tag works as a conditional statement and used for testing conditions.
- <c:choose> tag: - The role of this tag is similar to switch statement.
- <c:when> tag: - The role of this tag is similar to case in switch statement. Thus, it test the conditions.
- <c:otherwise> tag: - This tag behaves similar to default in switch statement. Thus, it executes when no condition for <c:when tag> matches.
- <c:catch> tag: - This tag is used to handle the exceptions.
- <c:import> tag: - This tag includes another files of any type such as JSP, HTML, XML etc. to current JSP file.
- <c:forEach> tag: - This is an iteration tag used to traverse the elements.
- <c:param> tag: - This tag is used to add the parameters.
- <c:url> tag: -

Example of JSTL core Tag

In this example, the functionality of certain JSTL core tags are shown in an easy way.

index.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
<head>
<title>Core Tag Example</title>
</head>
<body>
<c:set var="marks" scope="session" value="${35}"/>
<c:if test="${marks>30}">
<p>Congrats!! You <c:out value="${'passed'}"/> the exam </p>
</c:if>
<c:choose>
<c:when test="${marks>80}">
<c:out value="${'Good Performance'}"/>
</c:when>
<c:when test="${marks<80||marks>30}">
<c:out value="${'Average Performance'}"/>
</c:when>
<c:otherwise>
<c:out value="${'Bad Performance'}"/>
</c:otherwise>
</c:choose>
</body>
</html>
```

JSTL Formatting Tags

In JSTL, the role of formatting tag is to specify the type of format that represents date and time. The following syntax represent the URL of formatting tag: -

```
<%@ taglib uri = http://java.sun.com/jsp/jstl/fmt prefix = "fmt"  %>
```

Formatting tags

These are some commonly used formatting tags: -

- <fmt:parseNumber>: - This tag is used to parse the string on the basis of attribute associate with it.
- <fmt:parseDate>: - This tag is used to parse the string of a date and time.
- <fmt:formatDate>: - This tag handles the date in the various format.
- <fmt:message>: - This tag is used to display the internationalized message.
- <fmt:setTimeZone>: - This tag specifies the type of time zone.

Example of JSTL Format Tag

This is a simple example of JSTL format tag.

index.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
 <html>
  <head>
<title>Tutorial and Example</title>
 </head>
  <body>
<h2>Format Tag example</h2>
<c:set var="number" value="99.40" />
    <fmt:parseNumber var="n" integerOnly="true" type="number" value="${number}" />
    Parse Number is :: <c:out value="${n}" /> <br>
<c:set var="date" value="02-04-2018" />
<fmt:parseDate value="${date}" var="pd"  pattern="dd-MM-yyyy" />
Parse Date is :: <c:out value="${pd}" />
  <br>
  <c:set var="Date" value="<%=new java.util.Date()%>" />
Format Date is :: <fmt:formatDate type="time" value="${Date}"/>
  </body>
</html>
```

JSTL SQL Tags

In JSTL, SQL tags are used to access and manipulate various operations of database. It is responsible to interact the JSP page with any type of Database such as MySQL, Oracle, SQLite, etc. The below syntax is used to include SQL library within JSP: -

```
<%@ taglib prefix = "sql" uri = "http://java.sun.com/jsp/jstl/sql" %>
```

SQL Tags

These are some important JSTL SQL tags: -

- <sql:setDataSource>: - This tag creates a DataSource to communicate JSP with database server.
- <sql:query>: - This tag executes the SQL query that is specified with it.
- <sql:transaction>: - This tag performs various database action with single connection.
- <sql:param>: - This tag provides parameter to SQL statement.
- <sql:dateParam>: - This tag provides date in the form of parameter to SQL statement.
- <sql:update>: - This tag is used to update the value in database.

Example of JSTL SQL Tag

This is a simple example of JSTL SQL Tag that fetches data from MYSQL.

index.jsp

```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ page import="javax.servlet.http.*,javax.servlet.*" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>
<html>
<head>
<title>sql:query Tag</title>
</head>
<body>
<sql:setDataSource var="db" driver="com.mysql.jdbc.Driver"
     url="jdbc:mysql://localhost/jtp_gym"
     user="root"  password=""/>
<sql:query dataSource="${db}" var="rs">
SELECT * from admin;
</sql:query>
<table border="2" width="100%">
<tr>
<th>Student ID</th>
<th>Password</th>
<th>U Id</th>
</tr>
<c:forEach var="table" items="${rs.rows}">
<tr>
<td><c:out value="${table.id}"/></td>
<td><c:out value="${table.password}"/></td>
<td><c:out value="${table.user_id}"/></td>
</tr>
</c:forEach>
</table>
</body>
</html>
```

JSTL XML Tags

In JSTL, XML tags are used for creating and manipulating xml documents. The following syntax represent the URL of XML tag: -

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/xml" prefix="x" %>
```

To utilize the functionality of XML tags, it is required to download and add two jar files within the lib directory of your project. Following are the required jar files: -

- **XercesImpl.jar**
- **xalan.jar**

XML Tags

- <x:out>: - This role of this tag is similar to <%=expression%> tag but for XPath expression.
- <x:set>: - This tag associate the value with variable for XPath expression.
- <x:if>: - This tag treats as a conditional statement for XPath expression.
- <x:choose>: - This tag function as a switch statement for XPath expression.
- <x:when>: - This tag is used to test the conditions just same as Case in switch statement.
- <x:otherwise>: - This tag invokes if none of the <x:when> tag condition satisfies. Thus, it treats as default of switch statement.

Example of JSTL XML Tag

This example shows the functionality of various tags of JSTL XML.

index.jsp

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ taglib prefix="x" uri="http://java.sun.com/jsp/jstl/xml" %>
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
<h2>JSTL XML Tag Demo </h2>
<c:set var="website">
<website>
      <name>Tutorial and Example</name>
      <tutorial>Java Server Pages</tutorial>
    </website>
</c:set>
<x:parse xml="${website}" var="output"/>
<b>Website :: </b>
<x:out select="$output/website[1]/name" /><br>
<b>Tutorial ::</b>
<x:out select="$output/website[1]/tutorial" />
</body>
```

JSTL Function Tags

In JSTL, the purpose of function tags is similar to the various string methods in Java. Hence, these tag performs common string manipulation operations.

Functional tag is represented by following URL: -

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
```

Function Tags

These are some frequently used function tags: -

- fn:length(): - This tag returns the length of the string i.e. number of characters present in the string.
- fn:trim(): - This tag removes the spaces from both ends of the string.
- fn:replace(): - This tag is used to replace one string with another.
- fn:substring(): - This tag provides the substring (i.e. part of string) the string.
- fn:join(): - This tag is used to concatenates separate strings of an array
- fn:split(): - This tag breaks the string into array of substrings.
- fn:contains(): - This tag checks whether the input string contains the specified substring.
- fn:toUpperCase(): - This tag converts the lower case elements of a string into upper case.
- fn:toLowerCase(): - This tag converts the upper case elements of a string into lower case.

Example of JSTL Function Tag

This is an example of JSTL Function tag that shows various JSTL function tags functionality.

index.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/functions" prefix="fn" %>
<html>
<head>
<title>Function Tag</title>
</head>
<body>
<c:set var="web" value="Tutorial and Example"/>
Length of String :: ${fn:length(web)} <br>
Substring of String :: ${fn:substring(web,0,8)} <br>
Is it contains word "Tutorial" :: ${fn:contains(web,'Tut')}<br>
In UpperCase :: ${fn:toUpperCase(web)}
</body>
</html>
```

JSP Custom Tag

JSP custom tag provides flexibility to programmers, to declare their own tag. Thus, a user-defined tag can be created through custom tags to perform various operations.

These custom tags behave as operations on tag handler when JSP page is translated into servlet. These operations are invoked by web container when servlet of JSP page is executed.

How to create Custom Tag?

These are the steps that you need to follow to create custom tag: -

- Tag Handler class – Create a tag handler class with .java extension. This class is used to perform operations of custom tag.
- JSP file – Create a JSP file and include a tag library in it.
- TLD file – It is required to create tag library file (i.e. TLD file) in WEB-INF directory of your project. This file is represented by .tld extension.

It is required to add both external jars i.e. jsp-api.jar and servlet-api.jar in classpath of your project before deploying the application.

Example of JSP Custom Tag

In this example,

customTagHandler.java

```
package com.tutorialandexample;
import javax.servlet.jsp.tagext.*;
import javax.servlet.jsp.*;
import java.io.*;
public class CustomTagHandler extends SimpleTagSupport {
public void doTag() throws JspException,IOException
{
JspWriter jw=getJspContext().getOut();
jw.println("<h2>JSP Custom Tag</h2>");
}
}
```

Here, javax.servlet.jsp.tagext.* package contains all the classes that support custom tags and the role of SimpleTagSupport is to provide methods through which JSPWriter object can be accessed.

index.jsp

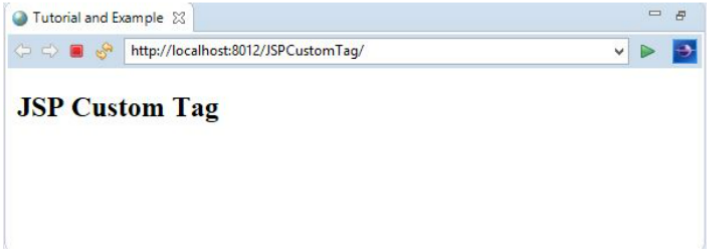
```
<html>
<head>
<title>Tutorial and Example</title>
</head>
<body>
   <%@ taglib uri="WEB-INF/xyz.tld" prefix="m" %>
<m:message/>
</body>
</html>
```

xyz.tld

```
<taglib>
<tlib-version>1.0</tlib-version>
<jsp-version>2.0</jsp-version>
<short-name>MyCustom</short-name>
<tag>
<name>message</name>
<tag-class>com.tutorialandexample.CustomTagHandler</tag-class>
<body-content>empty</body-content>
</tag>
</taglib>
```

Here, the name of tag (message) is given within the <name> tag and the full name of class is given within the <tag-class> tag. The value "empty" given within <body-content> means it doesn't contain any body.

Output



Note – We can also provide body and attribute within custom tags.

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

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

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

+91 8007592194 +91 9284926333

codewitharrays@gmail.com

https://codewitharrays.in/project