

JSP

By
Sudha Agarwal

INDEX

❖ JSP Introduction



IMPLICIT OBJECTS IN JSP

- ❖ JSP provide access to some implicit object which represent some commonly used objects for servlets that JSP page developers might need to use.
- ❖ For example you can retrieve HTML form parameter data by using request variable, which represent the `HttpServletRequest` object.

```
<%  
    String user = request.getParameter("user");  
%>  
  
Hello, <% out.println(user); %>
```

The "request" object is implicit here,
associated with `HttpServletRequest` object

The "out" object is implicit in JSP, associated with
the `JspWriter` object.

IMPLICIT OBJECTS IN JSP

- ❖ **request** : The HttpServletRequest object associated with the request.
- ❖ **response** : The HttpServletResponse object associated with the response that is sent back to the browser.
- ❖ **out** : The JspWriter object associated with the output stream of the response.
- ❖ **session** : The HttpSession object associated with the session for the given user of request.
- ❖ **application** : The ServletContext object for the web application.

IMPLICIT OBJECTS IN JSP

❖ **config** : The ServletConfig object associated with the servlet for current JSP page.

❖ **pageContext** : The PageContext object that encapsulates the environment of a single request for this current JSP page.

❖ **page** : The page variable is equivalent to this variable of Java programming language.

❖ **exception** : The exception object represents the Throwable object that was thrown by some other JSP page.

EXCEPTION HANDLING

❖ Exception Handling is a process of handling exceptional condition that might occur in your application. Exception Handling in JSP is much easier than Java Technology exception handling. Although JSP Technology also uses the same exception class objects.

WAYS TO PERFORM EXCEPTION HANDLING IN JSP

❖ JSP provide 3 different ways to perform exception handling:

1. Using **isErrorPage** and **errorPage** attribute of page directive.
2. Using **<error-page>** tag in Deployment Descriptor.
3. Using simple **try...catch block**.

ISERRORPAGE AND ERRORPAGE ATTRIBUTE

❖ isErrorPage attribute in page directive officially appoints a JSP page as an error page.

❖ Error.jsp

```
<%@ page isErrorPage="true" %>

<html>
<body>

<strong>You are here because we are not able to
find the page you have asked for.</strong>



</body>
</html>
```

This attribute officially designate this page as an error page.

ISERRORPAGE AND ERRORPAGE ATTRIBUTE

ATTRIBUTE

❖ **errorPage** attribute in a page directive informs the Web Container that if an exception occurs in the current page, forward the request to the specified error page.

❖ sum.jsp

```
<%@ page errorPage="error.jsp" %>
```

```
<html>
```

```
<body>
```

```
<% int x=10/0; %>
```

```
The sum is <%= x %>
```

```
</body>
```

```
</html>
```

Tells the Web Container that if some exception occurs here, forward the request to **error.jsp**

ERROR PAGE IN DD

❖ You can also declare error pages in the DD for the entire Web Application. Using `<error-page>` tag in the Deployment Descriptor.

```
<error-page>  
<exception-type>java.lang.Throwable</exception-type>  
<location>/error.jsp</location>  
</error-page>
```

```
<error-page>  
<error-code>404</error-code>  
<location>/error.jsp</location>  
</error-page>
```

USING THE TRY...CATCH BLOCK

```
<html>
<head>
  <title>Try...Catch Example</title>
</head>
<body>
  <%
    try{
      int i = 100;
      i = i / 0;
      out.println("The answer is " + i);
    }
    catch (Exception e){
      out.println("An exception occurred: " + e.getMessage());
    }
  %>
</body>
</html>
```

STANDARD TAG(ACTION ELEMENT)

- ❖SP specification provides Standard(Action) tags for use within your JSP pages.
- ❖These tags are used to remove or eliminate scriptlet code from your JSP page because scriptlet code are technically not recommended nowadays. It's considered to be bad practice to put java code directly inside your JSP page.
- ❖Standard tags begin with the **jsp:** prefix. There are many JSP Standard Action tag which are used to perform some specific task.

STANDARD TAG(ACTION ELEMENT)

❖ **jsp:useBean** : instantiates a JavaBean

❖ **Usage** : `<jsp:useBean id="beanId" />`

❖ **jsp:getProperty** : retrieves a property from a JavaBean instance.

```
<jsp:useBean id="beanId" ... />  
...  
<jsp:getProperty name="beanId" property="someProperty" .../>
```

❖ Where, beanName is the name of pre-defined bean whose property we want to access.

❖ **jsp:setProperty** : store data in property of any JavaBeans instance.

```
<jsp:useBean id="beanId" ... />  
...  
<jsp:setProperty name="beanId" property="someProperty" value="some value"/>
```

JSP:USEBEAN

❖ We provide the identifier for the bean and class is the full path of the bean class.

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/T
R/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Action JSP1</title>
</head>
<body>
<jsp:useBean id="name" class="demotest.DemoClass">
</body>
</html>
```

JSP:USEBEAN

❖TestBean.java

```
package demotest;

import java.io.Serializable;

public class TestBean implements Serializable{

    private String msg = "null";

    public String getMsg() {
        return msg;
    }

    public void setMsg(String msg) {
        this.msg = msg;
    }

}
```


JSP:USEBEAN

❖TestBean.java

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Guru Action 3</title>
</head>
<body>
<jsp:useBean id="GuruTest" class="demotest.TestBean" />
<jsp:setProperty name="GuruTest" property="msg" value="GuruTutorial" />
<jsp:getProperty name="GuruTest" property="msg" />
</body>
</html>
```

JSP:INCLUDE

❖ includes the runtime response of a JSP page into the current page.

```
<jsp:include page="page URL" flush="true/false">
```

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/T
R/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Date Guru JSP</title>
</head>
<body>
<jsp:include page="date.jsp" flush="true" />
</body>
</html>
```

JSP:FORWARD

- ❖ It is used to forward the request to another jsp or any static page.
- ❖ Here the request can be forwarded with no parameters or with parameters. Usage : **<jsp:forward page="Relative URL" />**

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">
<title>Guru Action JSP1</title>
</head>
<body>
<jsp:forward page="jsp_action_42.jsp" />
</body>
</html>
```

JSP:PLUGIN

- ❖ It is used to introduce Java components into jsp, i.e., the java components can be either an applet or bean.
- ❖ It detects the browser and adds <object> or <embed> tags into the file

```
<jsp:plugin type="applet/bean" code="objectcode" codebase="objectcodebase">
```

- ❖ Here the type specifies either an object or a bean
- ❖ Code specifies class name of applet or bean
- ❖ Code base contains the base URL that contains files of classes

JSP:PLUGIN

- ❖ It is used to introduce Java components into jsp, i.e., the java components can be either an applet or bean.
- ❖ It detects the browser and adds <object> or <embed> tags into the file

```
<jsp:plugin type="applet/bean" code="objectcode" codebase="objectcodebase">
```

- ❖ Here the type specifies either an object or a bean
- ❖ Code specifies class name of applet or bean
- ❖ Code base contains the base URL that contains files of classes

JAVABEAN COMPONENTS

- ❖ A JavaBeans component is a Java class with the following features:
- ❖ A no-argument constructor.
- ❖ Properties defined with accessors and mutators (getter and setter method).
- ❖ Class must not define any public instance variables.
- ❖ The class must implement the `java.io.Serializable` interface.

JAVABEAN COMPONENTS

```
import java.io.Serializable;

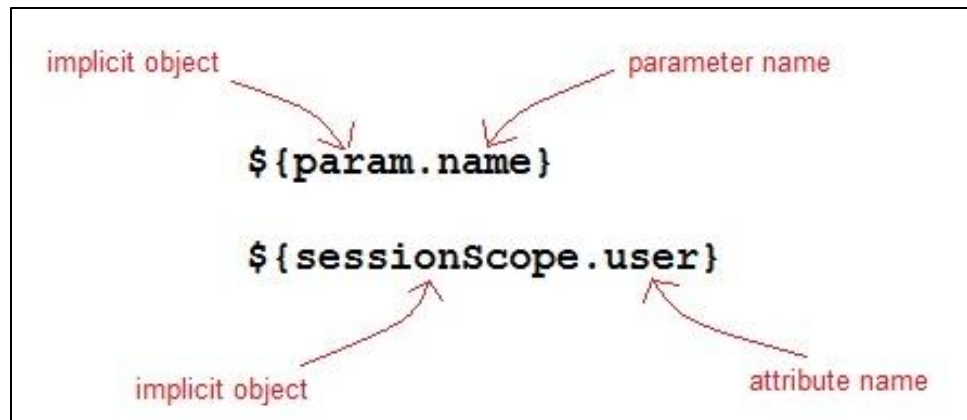
public class StudentBean implements Serializable
{
    private String name;
    private int roll;
    public StudentBean()
    {
        this.name="";
        this.roll="";
    }
    public void setName(String name)
    {
        this.name = name;
    }
    public String getName()
    {
        return name;
    }
    public int getRoll()
    {
        return roll;
    }
    public void setRoll(int roll)
    {
        this.roll = roll;
    }
}
```


EXPRESSION LANGUAGE

❖ Expression Language(EL) was added to JSP 2.0 specification. The purpose of EL is to produce scriptless JSP pages. The syntax of EL in a JSP is as follows:

```
${expr}
```

❖ Here expr is a valid EL expression. An expression can be mixed with static text/values and can also be combined with other expressions to form larger expression.



HOW EL EXPRESSION IS USED?

❖ EL expression can be used in two ways in a JSP page

1. As attribute values in standard and custom tags. Example:

```
<jsp:include page="${location}">
```

❖ Where location variable is separately defines in the jsp page.

❖ Expressions can also be used in jsp:setProperty to set a properties value, using other bean properties like : If we have a bean named Square with properties length, breadth and area.

```
<jsp:setProperty name="square" property="area" value="${square.length*square.breadth}" />
```

HOW EL EXPRESSION IS USED?

2. To output in HTML tag :

```
<h1>Welcome ${name}</h1>
```

❖ To deactivate the evaluation of EL expressions, we specify the `isELIgnored` attribute of the `page` directive as below:

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

EL IMPLICIT OBJECTS

❖ **pageContext** : It represents the PageContext object.

❖ **pageScope** : It is used to access the value of any variable which is set in the Page scope

❖ **requestScope** : It is used to access the value of any variable which is set in the Request scope.

❖ **sessionScope** : It is used to access the value of any variable which is set in the Session scope

❖ **applicationScope** : It is used to access the value of any variable which is set in the Application scope

EL IMPLICIT OBJECTS

- ❖ **Param** : Map a request parameter name to a single value
- ❖ **paramValues** : Map a request parameter name to corresponding array of string values.
- ❖ **Header** : Map containing header names and single string values.
- ❖ **headerValues** : Map containing header names to corresponding array of string values.
- ❖ **Cookie** : Map containing cookie names and single string values.

EL IMPLICIT OBJECTS

❖ Index.jsp

```
<form method="POST" action="welcome.jsp">  
  Name <input type="text" name="user" >  
  <input type="submit" value="Submit">  
</form>
```

❖ Welcome.jsp

```
<html>  
  <head>  
    <title>Welcome Page</title>  
  </head>  
  
  <body>  
    <h1>Welcome ${param.name}</h1>  
  </body>  
</html>
```

ARITHMETIC OPERATIONS IN EL

Arithmetic Operation	Operator
Addition	+
Substraction	-
Multiplication	*
Division	/ and div
Remainder	% and mod

LOGICAL AND RELATIONAL OPERATORS

Logical and Relational Operator	Operator
Equals	<code>==</code> and <code>eq</code>
Not equals	<code>!=</code> and <code>ne</code>
Less Than	<code><</code> and <code>lt</code>
Greater Than	<code>></code> and <code>gt</code>
Greater Than or Equal	<code>>=</code> and <code>ge</code>
Less Than or Equal	<code><=</code> and <code>le</code>
and	<code>&&</code> and <code>and</code>
or	<code> </code> and <code>or</code>
not	<code>!</code> and <code>not</code>

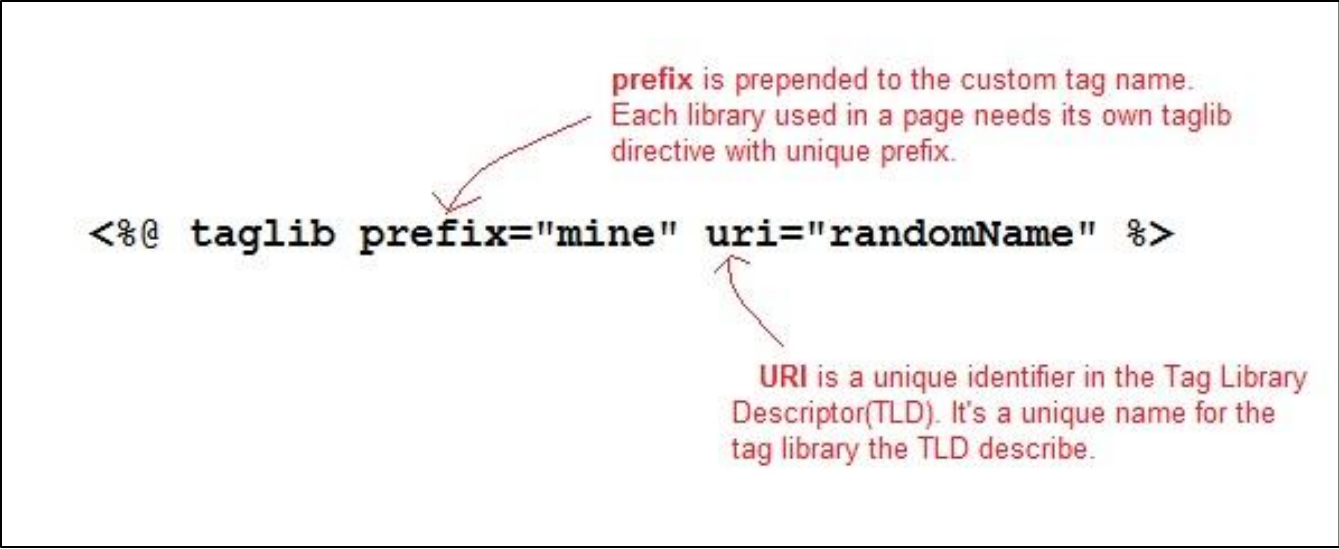
TAGLIB DIRECTIVE

- ❖ The taglib directive is used to define tag library that the current JSP page uses.
- ❖ A JSP page might include several tag library. JavaServer Pages Standard Tag Library (JSTL), is a collection of useful JSP tags, which provides many commonly used core functionalities.
- ❖ It has support for many general, structural tasks such as iteration and conditionals, readymade tags for manipulating XML documents, internationalization tags, and for performing SQL operations.
- ❖ Syntax of taglib directive is:

```
<%@ taglib prefix="prefixOfTag" uri="uriOfTagLibrary" %>
```

TAGLIB DIRECTIVE

- ❖ The prefix is used to distinguish the custom tag from other library custom tag. Prefix is prepended to the custom tag name. Every custom tag must have a prefix.
- ❖ The URI is the unique name for Tag Library.



```
<%@ taglib prefix="mine" uri="randomName" %>
```

- ❖ You can name the prefix anything, but it should be unique.

USING TAGLIB DIRECTIVE

❖ To use the JSTL in your application you must have the jstl.jar in your webapps /WEB-INF/lib directory. Download the jar file from page <https://tomcat.apache.org/taglibs/index.html>

❖ There are many readymade JST Libraries available which you use to make your life easier. Following is a broad division on different groups of JST libraries :

1. Core Tags - URI → <http://java.sun.com/jsp/jstl/core>
2. Formatting Tags - URI → <http://java.sun.com/jsp/jstl/fmt>
3. SQL Tags - URI → <http://java.sun.com/jsp/jstl/sql>
4. XML Tags - URI → <http://java.sun.com/jsp/jstl/xml>
5. JSTL Functions - URI → <http://java.sun.com/jsp/jstl/functions>



JSTL

- ❖ JSP Standard Tag Library(JSTL) is a standard library of readymade tags and it is a collection of custom JSP tag libraries that provide common web development functionality.
- ❖ The JSTL contains several tags that can remove scriplet code from a JSP page by providing some ready to use, already implemented common functionalities.

ADVANTAGES OF JSTL

❖ **Standard Tag:** It provides a rich layer of the portable functionality of JSP pages. It's easy for a developer to understand the code.

❖ **Code Neat and Clean:** As scriptlets confuse developer, the usage of JSTL makes the code neat and clean.

❖ **Automatic JavabeansInterospection Support:** It has an advantage of JSTL over JSP scriptlets. JSTL Expression language handles JavaBean code very easily. We don't need to downcast the objects, which has been retrieved as scoped attributes. Using JSP scriptlets code will be complicated, and JSTL has simplified that purpose.

❖ **Easier for humans to read:** JSTL is based on XML, which is very similar to HTML. Hence, it is easy for the developers to understand.

ADVANTAGES OF JSTL

- ❖ **Easier for computers to understand:** Tools such as Dreamweaver and front page are generating more and more HTML code.
- ❖ HTML tools do a great job of formatting HTML code. The HTML code is mixed with the scriptlet code.
- ❖ As JSTL is expressed as XML compliant tags, it is easy for HTML generation to parse the JSTL code within the document.

JSTL

❖ **JSTL is divided into 5 groups**

❖ **JSTL Core:** JSTL Core provides several core tags such as **if**, **forEach**, **import**, **out** etc to support some basic scripting task. Url to include JSTL Core Tag inside JSP page is →

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

❖ **JSTL Formatting:** JSTL Formatting library provides tags to format text, date, number for Internationalised web sites. Url to include JSTL Formatting Tags inside JSP page is →

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

JSTL

❖ **JSTL sql:** JSTL SQL library provides support for Relational Database Connection and tags to perform operations like insert, delete, update, select etc on SQL databases. Url to include JSTL SQL Tag inside JSP page is →

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

❖ **JSTL XML:** JSTL XML library provides support for XML processing. It provides flow control, transformation features etc. Url to include JSTL XML Tag inside JSP page is →

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

JSTL

❖ **JSTL functions:** JSTL functions library provides support for string manipulation. Url to include JSTL Function Tag inside JSP page is →

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

JSTL CORE LIBRARY

❖ The JSTL core library contains several tags that can be used to eliminate the basic scripting overhead such as for loop, if...else conditions etc from a JSP Page. Let's study some important tags of JSTL Core library.

❖ **JSTL if tag:** The if tag is a conditional tag used to evaluate conditional expressions. When a body is supplied with if tag, the body is evaluated only when the expression is true. For Example :

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
  <head>
    <title>Tag Example</title>
  </head>
  <body>
    <c:if test="${param.name == 'studytonight'}">
      <p>Welcome to ${param.name} </p>
    </c:if>
  </body>
</html>
```

JSTL CORE LIBRARY

❖ **JSTL out tag:** The out tag is used to evaluate an expression and write the result to JspWriter. For Example :

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
  <head>
    <title>Tag Example</title>
  </head>
  <body>
    <c:out value="${param.name}" default="StudyTonight" />
  </body>
</html>
```

❖ The value attribute specifies the expression to be written to the JspWriter. The default attribute specifies the value to be written if the expression evaluates null.

JSTL CORE LIBRARY

❖ **JSTL forEach tag:** This tag provides a mechanism for iteration within a JSP page. JSTL forEach tag works similarly to enhanced for loop of Java Technology. You can use this tag to iterate over an existing collection of items. For Example :

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
  <head>
    <title>Tag Example</title>
  </head>
  <body>
    <c:forEach var="message" items="${errorMsgs}" >
      <li>${message}</li>
    </c:forEach>
  </body>
</html>
```

❖ Here the attribute items has its value as an EL expression which is a collection of error messages. Each item in the iteration will be stored in a variable called message which will be available in the body of the forEach tag.

JSTL CORE LIBRARY

❖ **JSTL import tag:** `< c:import>` tag is used to dynamically add the contents from the provided URL to the current page, at request time. The URL resource used in the `< c:import>` url attribute can be from outside the web Container. For Example :

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
  <head>
    <title>Tag Example</title>
  </head>
  <body>
    <c:import url="http://www.example.com/hello.html">>
    <c:param name="showproducts" value="true"/>
    </c:import>
  </body>
</html>
```


JSTL CORE LIBRARY

❖ **JSTL url tag:** The JSTL url tag is used to store a url in a variable and also perform url rewriting when necessary. For Example

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<html>
  <head>
    <title>Tag Example</title>
  </head>
  <body>
    <a href='<c:url value="/home.jsp"/>' > Go Home </a>
  </body>
</html>
```

JSTL SQL TAGS

❖ The JSTL SQL tag library provides tags for interacting with relational databases (RDBMSs) such as Oracle, MySQL, or Microsoft SQL Server.

JSTL SQL TAGS

<sql:setDataSource>

Creates a simple DataSource suitable only for prototyping

<sql:query>

Executes the SQL query defined in its body or through the sql attribute.

<sql:update>

Executes the SQL update defined in its body or through the sql attribute.

<sql:param>

Sets a parameter in an SQL statement to the specified value.

<sql:dateParam>

Sets a parameter in an SQL statement to the specified java.util.Date value.

<sql:transaction >

Provides nested database action elements with a shared Connection, set up to execute all statements as one transaction.

LAB

- ❖ Q1. Do the following with only JSPs (and no servlets) and using JSP scripting elements
- ❖ Create a method that take two int numbers and return their sum as int.
- ❖ Invoke the method from within template text using JSP elements (example o/p - Sum of 3 and 4 is 7.)
- ❖ Print above line to console as well using JSP elements.