JSP

Think developing!

JSP

Java Server Pages (JSP) это технология для разработки веб-страниц, которые поддерживают динамическое содержание, которое помогает разработчикам Java вставить код в HTML страницы путем использования специальных тегов JSP, большинство из которых начинаются с **% <** и заканчиваться **%>**.

Компонент JavaServer Pages является одним из видов сервлетов Java, который предназначен для выполнения роли пользовательского интерфейса для

Java веб-приложений. Веб-разработчи пишут JSP, как текстовые файлы, которые сочетают HTML или XHTML код, XML элементы и встроенные JSP действия и команды.

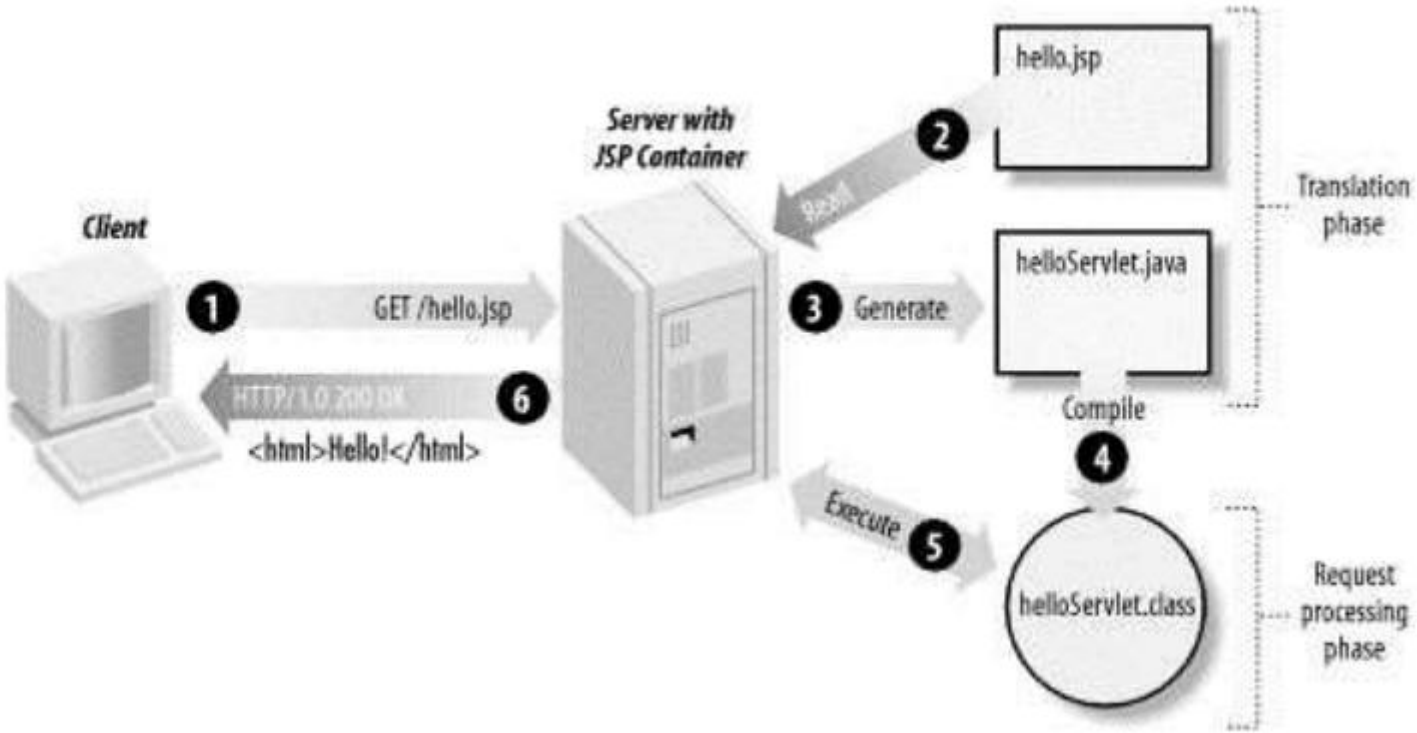
Используя JSP, вы можете получить входные данные от пользователей через формы веб-страницы, получать записи из базы данных или другого источника, а также создавать веб-страницы динамически.

Преимущества

JavaServer Pages часто служат той же цели, что и программы использующие Common Gateway Интерфейс (CGI). Но JSP обладают рядом преимуществ по сравнению с CGI:

- производительность значительно лучше, потому что позволяет встраивать динамические элементы в HTML страницы, вместо отдельных CGI файлов.
- JSP всегда компилируются, прежде чем обрабатывается сервером в отличие от CGI / Perl которая требует от сервера загрузить компилятор и целевой сценарий каждый раз, когда страница запрашивается.
- JavaServer Pages построены на основе сервлетов, JSP также имеет доступ ко всем Java APIs, включая JDBC, JNDI, EJB, JAXP т.д.
- JSP страницы могут использоваться в комбинации с сервлетами, которые занимаются бизнес-логикой
- Наконец, JSP является неотъемлемой частью J2EE, платформы для enterprise приложений.

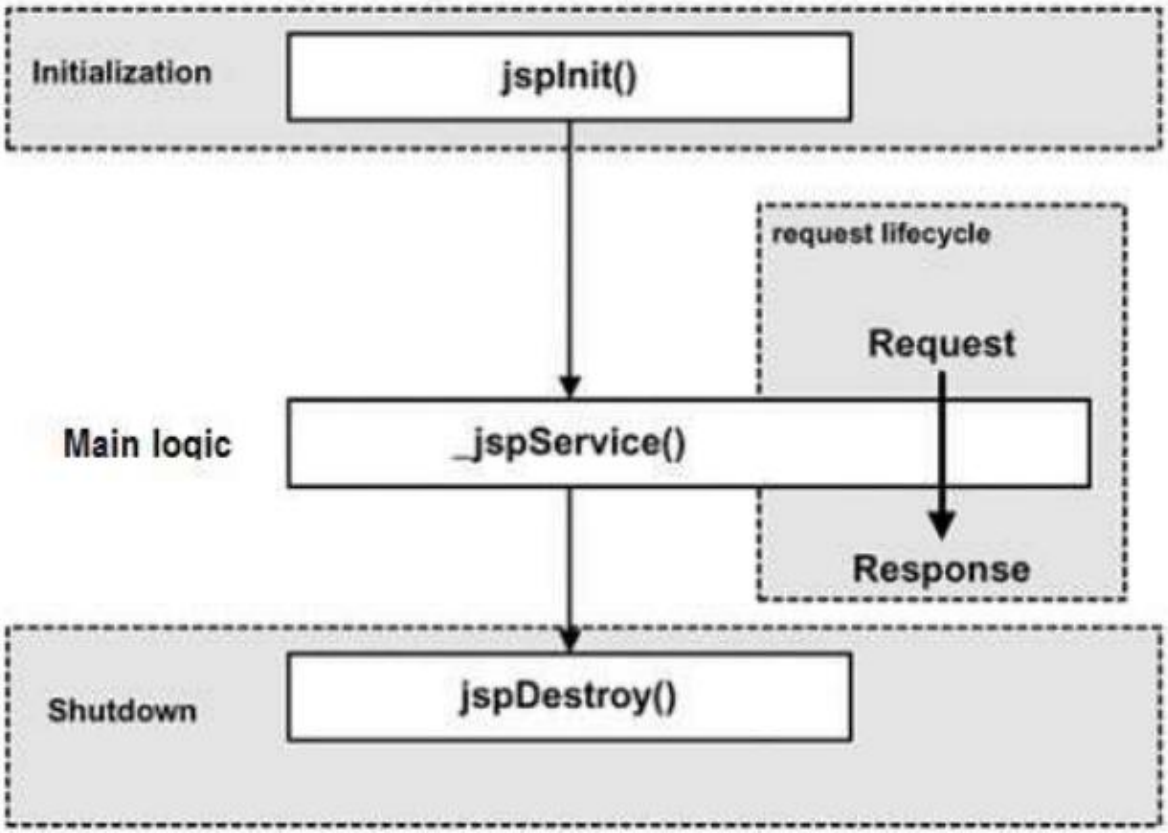
JSP Processing



JSP Processing

- Как и в обычной странице, ваш браузер посылает HTTP запрос к веб-серверу.
- веб-сервер распознает, что запрос HTTP для страницы JSP и направляет его в JSP engine. Это можно сделать с помощью URL или JSP страницы, которая заканчивается JSP. вместо HTML.
- JSP engine загружает страницы с диска и преобразует его в сервлет содержания. Эти преобразования очень простые, в которых все шаблоны текста преобразуется в println () и все JSP элементы преобразуются в Java код, который реализует соответствующее динамическое поведение страницы.
- JSP engine компилирует сервлет в исполняемый класс и передает исходный запрос в сервлет engine.
- Сервлета engine загружает класс сервлета и выполняет его. Во время выполнения сервлет формирует выходные данные в формате HTML, который сервлет передает веб-серверу внутри HTTP ответа.
- веб-сервер перенаправляет HTTP ответ на ваш браузер как статический HTML
- Наконец веб-браузер обрабатывает динамически генерируемые HTML страницы внутри ответа HTTP так, как будто это были статические страницы.

JSP – Life Cycle



JSP Compilation

Когда браузер запрашивает JSP, JSP engine сначала проверяет, есть ли необходимость в компиляции страницы. Если страница никогда не была скомпилирована, или, если JSP был изменен с момента последней компиляции, двигатель JSP компилирует страницы.

Процесс компиляции состоит из трех этапов:

- Parsing JSP.
- Преобразование JSP в сервлет.
- Компиляция сервлета.

JSP Initialization

Когда контейнер загружает JSP он вызывает jspInit () метод перед обслуживанием любых запросов.

Обычно инициализация выполняется только один раз как и метод инициализации сервлета

JSP Execution

Этот этап жизненного цикла JSP представляет все взаимодействия с запросами, пока не JSP удален.

Всякий раз, когда браузер запрашивает JSP страницу после загрузки и инициализации, JSP engine вызывает метод _jspService ():

```
void _jspService(HttpServletRequest request,
        HttpServletResponse response)
{
        // Service handling code...
}
```

JSP Cleanup

Эта фаза жизненного цикла JSP вызывается, когда JSP удаляется из контейнера.

JspDestroy () является JSP эквивалентом метода destroy сервлета. Переопределите jspDestroy когда вам нужно выполнить очистку, как освобождение соединения с базой данных или закрытие открытых файлов.

Scriptlet

Скриптлет может содержать любое количество JAVA выражений, переменных или деклараций метода, или выражений, которые являются действительными на языке сценариев страницы. Ниже приводится синтаксис скриптлета:

<% code fragment %>

Scriptlet

Любой текст, HTML теги или JSP элементы должны быть вне скриптлета:

```
<html>
<head><title>Hello World</title></head>
<body>
Hello World!<br/>
<%
out.println("Your IP address is " + request.getRemoteAddr());
%>
</body>
</html>
```

JSP Declarations

Декларация заявляет, одну или несколько переменных или методов, которые можно использовать в коде Java позже в JSP файле.

```
<%! int i = 0; %>
<%! int a, b, c; %>
<%! Circle a = new Circle(2.0); %>
```

JSP Expression

Элемент JSP выражение содержит сценарии языка выражения, которое вычисляется, преобразуется в строку, а результат вставляется там, где выражение появляется в файле JSP. Выражение JSP может содержать любое выражение языка Java, но вы не можете использовать точку с запятой в конце выражения.

```
<html>
<head><title>A Comment Test</title></head>
<body>
<p>
Today's date: <%= (new java.util.Date()).toLocaleString()%>
</p>
</body>
</html>
```

JSP Comments


```
<html>
<head><title>A Comment Test</title></head>
<body>
<h2>A Test of Comments</h2>
<%-- This comment will not be visible in the
   page source --%>
</body>
</html>
```

JSP Directives

Директивы JSP влияют на общую структуру класса сервлета, имеет следующий вид:

<%@ directive attribute="value" %>

Есть три типа тегов директивы:

<%@ page ... %> Аттрибуты страницы, как скриптовый язык, страница ошибки

<%@ include ... %> включает файл на этапе трансляции

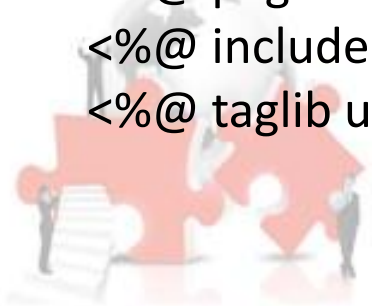
<%@ taglib ... %> Определяет библиотеку тегов с пользовательскими действиями

<%@ page errorPage="errorpage.jsp"%>

<%@ page contentType="text/html:charset=ISO-8859-1" %>

<%@ include file="header.jsp" %>

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



The page Directive

<%@ page attribute="value" %>

buffer Определяет буфер для выходного потока.

autoFlush поведение выходного буфера сервлета

contentType тип контент

errorPage URL страницы обработки ошибок

isErrorPage определяет страницу как страницу обработки ошибок

extends определяет суперкласс, от которого сгенерированный сервлет будет расширен

import список пакетов, кот будут использоваться

isThreadSafe потоковая модель для сгенерированного сервлета

session работет ли JSP страница с HTTP sessions

isScriptingEnabled определяет доступность скриптовых элементов на странице

JSP Actions

JSP действия используют конструкции в XML-синтаксис для управления поведением сервлета. Вы можете динамически вставлять файл, повторно использовать компоненты JavaBeans, перенаправить пользователя на другую страницу, либо сгенерировать HTML для Java плагина.

<jsp:action_name attribute="value" />

<jsp:include page="*urlSpec*" flush="true|false"/>

<jsp:forward page="*relativeURLspe*c" />

JSP Actions

jsp:include Includes a file at the time the page is requested

jsp:useBean Finds or instantiates a JavaBean

jsp:setProperty Sets the property of a JavaBean

jsp:getProperty Inserts the property of a JavaBean into the output

jsp:forward Forwards the requester to a new page

jsp:plugin Generates browser-specific code that makes an OBJECT or EMBED tag for the Java plugin

jsp:element Defines XML elements dynamically

jsp:attribute Defines dynamically defined XML element's attribute.

jsp:body Defines dynamically defined XML element's body.

jsp:text Use to write template text in JSP pages and documents.

JSP Actions


```
<%@page language="java" contentType="text/html"%>
<head><title>Generate XML Element</title></head>
<body>
    <jsp:element name="xmlElement">
        <jsp:attribute name="xmlElementAttr“>Value for the attribute</jsp:attribute>
        <jsp:body>Body for XML element</jsp:body>
    </jsp:element>
</body>
</html>
```

```
<html xmlns=http://www.w3c.org/1999/xhtml xmlns:jsp="http://java.sun.com/JSP/Page">
<head><title>Generate XML Element</title></head>
<body>
<xmlElement xmlElementAttr="Value for the attribute“>Body for XML element</xmlElement>
</body>
</html>
```

JSP Implicit Objects

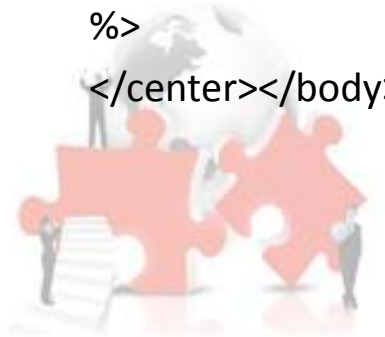
request	This is the HttpServletRequest object associated with the request.
response	This is the HttpServletResponse object associated with the response to the client.
out	This is the PrintWriter object used to send output to the client.
session	This is the HttpSession object associated with the request.
application	This is the ServletContext object associated with application context.
config	This is the ServletConfig object associated with the page.
pageContext	This encapsulates use of server-specific features like higher performance JspWriters.
page	This is simply a synonym for this, and is used to call the methods defined by the translated servlet class.
Exception	The Exception object allows the exception data to be accessed by designated JSP.

HTTP Header Request


```
<%@ page import="java.io.*,java.util.*" %>
<table width="100%" border="1" align="center“><tr bgcolor="#949494">
<th>Header Name</th><th>Header Value(s)</th>
</tr>
<%
    Enumeration headerNames = request.getHeaderNames();
    while(headerNames.hasMoreElements()) {
        String paramName = (String)headerNames.nextElement();
        out.print("<tr><td>" + paramName + "</td>\n");
        String paramValue = request.getHeader(paramName);
        out.println("<td> " + paramValue + "</td></tr>\n");
    }
%>
</table>
```

HTTP Header Response


```
<center>
<%
response.setIntHeader("Refresh", 5); // Set refresh, autoload time as 5 seconds
Calendar calendar = new GregorianCalendar(); // Get current time
String am_pm;
int hour = calendar.get(Calendar.HOUR);
int minute = calendar.get(Calendar.MINUTE);
int second = calendar.get(Calendar.SECOND);
if(calendar.get(Calendar.AM_PM) == 0)         am_pm = "AM";
else            am_pm = "PM";
String CT = hour+":"+ minute +":"+ second +" "+ am_pm;
out.println("Current Time is: " + CT + "\n");
%>
</center></body>
```



Form Processing


```
<html>
<body>
<center>
<h1>Using GET Method to Read Form Data</h1>
<ul>
<li><p><b>First Name:</b>
    <%= request.getParameter("first_name")%>
</p></li>
<li><p><b>Last Name:</b>
    <%= request.getParameter("last_name")%>
</p></li>
</ul>
</body>
</html>
```

Control-Flow Statements

```
<%! int day = 3; %>
<html>
<head><title>IF...ELSE Example</title></head>
<body>
<% if (day == 1 | day == 7) { %>
      <p> Today is weekend</p>
<% } else { %>
      <p> Today is not weekend</p>
<% } %>
</body>
</html>
```

Control-Flow Statements

```
<%! int day = 3; %>
<html>
<head><title>SWITCH...CASE Example</title></head>
<body>
<%
switch(day) {
case 0:
   out.println("It\'s Sunday.");
   break;
case 5:
   out.println("It\'s Friday.");
   break;
default:
   out.println("It's Saturday.");
}
%>
</body>
</html>
```

Control-Flow Statements

```
<%! int fontSize; %>
<html>
<head><title>FOR LOOP Example</title></head>
<body>
<%for ( fontSize = 1; fontSize <= 3; fontSize++){ %>
   <font color="green" size="<%= fontSize %>">
    JSP Tutorial
   </font><br />
<%}%>
</body>
</html>
```

Control-Flow Statements

```
<%! int fontSize; %>
<html>
<head><title>WHILE LOOP Example</title></head>
<body>
<%while ( fontSize <= 3){ %>
   <font color="green" size="<%= fontSize %>">
    JSP Tutorial
   </font><br />
<%fontSize++;%>
<%}%>
</body>
</html>
```

Servlet and JSP

- Filters
- Cookie
- Session
- File upload
- Send email

JSP - JSTL

JavaServer Pages Standard Tag Library (JSTL)

- Core Tags
- Formatting tags
- SQL tags
- XML tags
- JSTL Functions

Core Tags

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

<c:out > Like <%= ... >, but for expressions.

<c:set > Sets the result of an expression evaluation in a 'scope'

<c:remove > Removes a scoped variable (from a particular scope, if specified).

<c:catch> Catches any Throwable that occurs in its body and optionally exposes it.

<c:if> Simple conditional tag which evalutes its body if the supplied condition is true.

<c:choose> Simple conditional tag that establishes a context for mutually exclusive conditional

operations, marked by <when> and <otherwise>

<c:when> Subtag of <choose> that includes its body if its condition evalutes to 'true'.

<c:otherwise > Subtag of <choose> that follows <when> tags and runs only if all of the prior

conditions evaluated to 'false'.

<c:forEach > The basic iteration tag

<c:forTokens> Iterates over tokens, separated by the supplied delimeters.

<c:redirect > Redirects to a new URL.

Formatting tags

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

<fmt:formatNumber> To render numerical value with specific precision or format.

<fmt:parseNumber> Parses the string representation of a number, currency, or percentage.

<fmt:formatDate> Formats a date and/or time using the supplied styles and pattern

<fmt:parseDate> Parses the string representation of a date and/or time

<fmt:bundle> Loads a resource bundle to be used by its tag body.

<fmt:setLocale> Stores the given locale in the locale configuration variable.

<fmt:setBundle> Loads a resource bundle and stores it in the named scoped variable or the bundle

configuration variable.

<fmt:timeZone> Specifies the time zone for any time formatting or parsing actions nested in its body.

<fmt:setTimeZone> Stores the given time zone in the time zone configuration variable

<fmt:message> To display an internationalized message.

<fmt:requestEncoding> Sets the request character encoding

Formatting tags


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

<body>
<c:set var="balance" value="120000.2309" />
<p>Formatted Number (1): <fmt:formatNumber value="${balance}"
        type="currency"/></p> // £120,000.23
<p>Formatted Number (2): <fmt:formatNumber type="number"
        maxIntegerDigits="3" value="${balance}" /></p> // 000.231
<p>Formatted Number (8): <fmt:formatNumber type="number"
        pattern="###.###E0" value="${balance}" /></p> // 120E3
<p>Currency in USA :
<fmt:setLocale value="en_US"/>
<fmt:formatNumber value="${balance}" type="currency"/></p> // $120,000.23
</body>
</html>
```

SQL tags

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

<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.

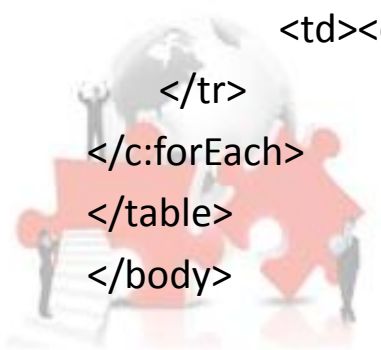
<sql:dateParam> Sets a parameter in an SQL statement to the specified java.util.Date value.

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

SQL tags


```
<%@ page import="java.io.*,java.util.*,java.sql.*"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/sql" prefix="sql"%>

<body>
<sql:setDataSource var="snapshot" driver="com.mysql.jdbc.Driver”
     url="jdbc:mysql://localhost/TEST” user="root" password="pass123"/>
<sql:query dataSource="${snapshot}" var="result“>SELECT * from Employees;</sql:query>
<table border="1" width="100%“><tr><th>Emp ID</th><th>Age</th></tr>
<c:forEach var="row" items="${result.rows}">
     <tr>
          <td><c:out value="${row.id}"/></td>
          <td><c:out value="${row.age}"/></td>
     </tr>
</c:forEach>
</table>
</body>
```



JSTL Functions

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

fn:contains() Tests if an input string contains the specified substring.

fn:endsWith() Tests if an input string ends with the specified suffix.

fn:indexOf() Returns the index withing a string of the first occurrence of a specified substring.

fn:join() Joins all elements of an array into a string.

fn:length() Returns number of characters in a string.

fn:replace() Returns a string resulting from replacing in an input string all occurrences with a

given string.

fn:split() Splits a string into an array of substrings.

fn:startsWith() Tests if an input string starts with the specified prefix.

fn:substring() Returns a subset of a string.

fn:toLowerCase() Converts all of the characters of a string to lower case.

fn:toUpperCase() Converts all of the characters of a string to upper case.

fn:trim() Removes white spaces from both ends of a string.

JSTL Functions


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

<body>

<c:set var="string1" value="This is first String."/>
<c:set var="string2" value="${fn:split(string1, ' ')}" />
<c:set var="string3" value="${fn:join(string2, '-')}" />

<p>String (3) : ${string3}</p>
</body>
```

String (3) : This-is-first-String.

JSP - Custom Tags

Custom Tags – это элемент языка JSP определенный пользователем. Когда страница JSP содержащих Custom Tag, тег преобразуется в операции в обработчике тега.Веб-контейнер затем вызывает эти операции, когда сервлет страницы JSP-выполняется.

```
<ex:Hello />

public class HelloTag extends SimpleTagSupport {

  public void doTag() throws JspException, IOException {
    JspWriter out = getJspContext().getOut();
    out.println("Hello Custom Tag!");
  }
}
```

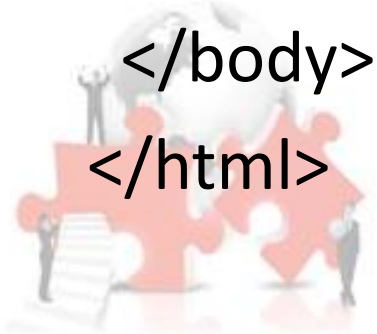
JSP - Custom Tags

webapps\ROOT\WEB-INF\custom.tld

```
<taglib>
  <tlib-version>1.0</tlib-version>
  <jsp-version>2.0</jsp-version>
  <short-name>Example TLD</short-name>
  <tag>
    <name>Hello</name>
    <tag-class>com.tutorialspoint.HelloTag</tag-class>
    <body-content>empty</body-content>
  </tag>
</taglib>
```

JSP - Custom Tags


```
<%@ taglib prefix="ex" uri="WEB-INF/custom.tld"%>
<html>
  <head>
    <title>A sample custom tag</title>
  </head>
  <body>
    <ex:Hello/>
  </body>
</html>
```



Custom Tags with body

```
<ex:Hello> This is message body </ex:Hello>

public class HelloTag extends SimpleTagSupport {
  StringWriter sw = new StringWriter();
  public void doTag() throws JspException, IOException  {
     getJspBody().invoke(sw);
     getJspContext().getOut().println(sw.toString());
   }
}
 <tag>
   <name>Hello</name>
   <tag-class>com.tutorialspoint.HelloTag</tag-class>
   <body-content>scriptless</body-content>
 </tag>
```

Custom Tag Attributes


```
public class HelloTag extends SimpleTagSupport {
  private String message;
  public void setMessage(String msg) {     this.message = msg;   }
  StringWriter sw = new StringWriter();

  public void doTag()     throws JspException, IOException    {
     if (message != null) {       /* Use message from attribute */
       JspWriter out = getJspContext().getOut();
       out.println( message );
     }  else {        /* use message from the body */
       getJspBody().invoke(sw);
       getJspContext().getOut().println(sw.toString());
     }
  }
}
```

Custom Tag Attributes

```
<taglib>
 <tag>   <name>Hello</name>
  <tag-class>com.tutorialspoint.HelloTag</tag-class>
  <body-content>scriptless</body-content>
  <attribute>
    <name>message</name>
  </attribute>
 </tag>
</taglib>

<%@ taglib prefix="ex" uri="WEB-INF/custom.tld"%>
<html>
<body>
  <ex:Hello message="This is custom tag" />
 </body>
</html>
```


SPILNA
SPRAVA
YOUR OUTSOURCING PARTNER
Q&A
Think developing!