

# JSP STANDARD TAG LIBRARY



**Библиотека JSTL – спецификация  
библиотечных компонент, которые  
включают повторно используемые действия  
для JSP-ориентированных приложений**

**JSTL 1.2.x**

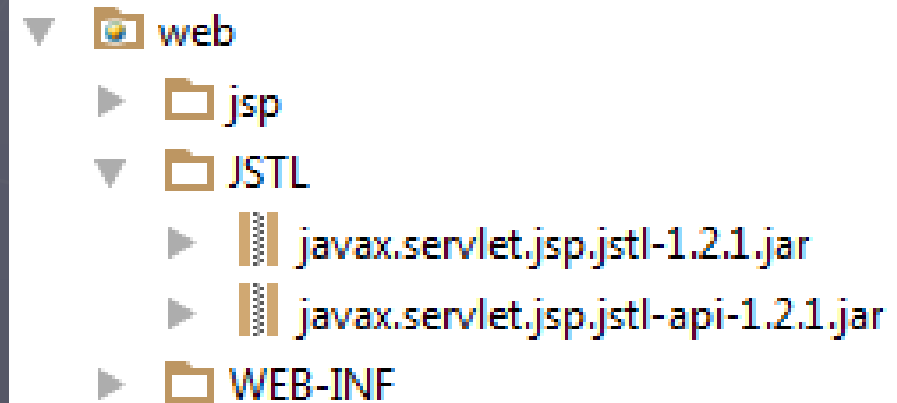
**Состав JSTL - группы тегов:**

- 1) основные теги — core,**
- 2) теги форматирования — formatting,**
- 3) теги для работы с SQL — sql,**
- 4) теги для обработки XML — xml,**
- 5) функции-теги для обработки строк —  
functions**

Библиотека	Число тегов	Описание
<b>core</b>	14	<i>Основные:</i> <b>if/then</b> выражения и конструкции множественного выбора; вывод; создание и удаление контекстных переменных; управление свойствами JavaBeans компонентов; перехват исключений; <b>forEach</b> для итерирования коллекций; создание URL и импортирование их содержимого.
<b>formatting</b>	12	<i>Интернационализация и форматирование:</i> установка локализации; локализация текста и структуры сообщений; форматирование и анализ чисел, процентов, денежных единиц и дат.
<b>sql</b>	6	<i>Доступ к БД:</i> описание источника данных; выполнение запросов, обновление данных и транзакций; обработка результатов запроса.
<b>xml</b>	10	<i>XML-анализ и преобразование:</i> преобразование XML; доступ и преобразование XML через XPath и XSLT.
<b>fn</b>	16	<i>Строки:</i> обработка объектов типа String и определение размера коллекций.

## ► Подключение

- javax.servlet.jsp.jstl-1.2.x.jar и javax.servlet.jsp.jstl-api-1.2.x.jar



```
<%@ page contentType="text/html; charset=UTF-8" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
  <body>
    <c:out value="JSTL подключена"/>
  </body>
</html>
```

# JSTL core

## Теги общего назначения:

`<c:out />` — вычисляет и выводит значение выражения;

`<c:set />` — создает и устанавливает переменную в указанную область видимости;

`<c:remove />` — удаляет переменную из указанной области видимости;

`<c:catch />` — перехватывает обработку исключения

```
<%@ page contentType="text/html; charset=UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
  <body>
    <c:set var="user" value="guest" scope="page"/>
    <c:if test="${ not empty user and user eq 'guest' }">
      User is Guest
    </c:if>
    <br/>
  </body>
</html>
```

выполняет

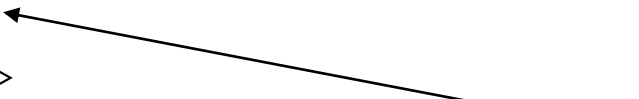
вычисляет значение атрибута test

Создает ссылку и позволяет извлекать значение,

```
<c:if test="${ not empty user and user eq 'guest' }"
      var = "testOperation" scope = "page">
  TRUE
</c:if>
```

Сохранить результаты в другом атрибуте

```
<c:set var="user" scope="page">  
  guest  
</c:set>
```



```
<c:out value="{user}"/>
```

Без value

```
<c:remove var="user"/>
```



атрибут user будет удален из области  
видимости и при попытке его поиска  
будет возвращено значение null

## Теги условного перехода:

`<c:if />` — тело тега выполняется, если значение выражения true;

`<c:choose />` (`<c:when />`, `<c:otherwise />`)  
— то же, что и `<c:if />` с поддержкой нескольких условий и действия, производимого по умолчанию



```
<c:set var="number" value="50"/>
<c:choose>
  <c:when test="{ number > 10 }" >
    <c:out value="число { number } больше 10"/>
  </c:when>
  <c:when test="{ number > 40 }" >
    <c:out value="число { number } больше 40"/>
  </c:when>
  <c:when test="{ number > 60 }" >
    <c:out value="число { number } больше 60"/>
  </c:when>
  <c:otherwise>
    <c:out value="число { number } меньше 10"/>
  </c:otherwise>
</c:choose>
```

число 50 больше 10

```
<c:set var="number" value="7.1" scope="session"/>
<c:if test="${ number < 9 }">
    <c:out value ="Number ${ number }"/> is smaller than (9
</c:if>
```

число 7.1 не может быть приведено к  
ожидаемому типу

## HTTP Status 500 - Internal Server Error

**type** Exception report

**message** Internal Server Error

**description** The server encountered an internal error that prevented it from fulfilling this request.

**exception**

org.apache.jasper.JasperException: /jstlexample.jsp(16,6) PWC6236: According to TLD or attribute directive in tag file, attribute test

**note** [The full stack traces of the exception and its root causes are available in the GlassFish Server Open Source Edition 4.1.1 logs.](#)

GlassFish Server Open Source Edition 4.1.1

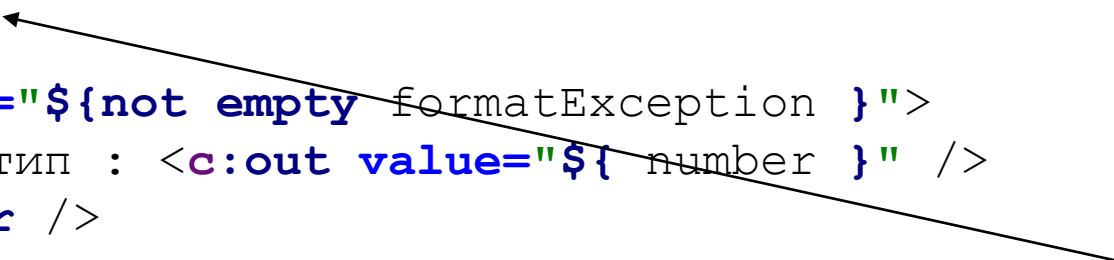
```

<c:set var="number" value="7.1" scope="session" />
<c:catch var="formatException">

    <c:if test="\${number < 9}">

        <c:out value="Number \${ number }" /> меньше чем (9)
    </c:if>
</c:catch>
<br/>
<c:if test="\${not empty formatException }">
    Неверный тип : <c:out value="\${ number }" />
    <br /><hr />
    Сгенерировано исключение
    <hr />
    \${ formatException }
</c:if>

```



обработка одного или нескольких исключений, генерация которых не должна приводить к переходу на заданные страницы обработки ошибок

## Итераторы:

`<c:forEach />` — выполняет тело тега для каждого элемента коллекции, массива, итерируемого объекта;

`<c:forEachTokens />` — выполняет тело тега для каждой лексемы в строке.

## Теги обработки URL:

`<c:redirect/>` — перенаправляет запрос на указанный адрес URL;

`<c:import/>` — добавляет на JSP содержимое указанного веб-ресурса;

`<c:url/>` — формирует адрес с учетом контекста приложения `request.getContextPath()`;

`<c:param/>` — добавляет параметр к запросу, сформированному при помощи `<c:url/>`.

## а) Добавим список

```
public class MessageList extends ArrayList<Message>{

    this.add(new Message(72, "Hello"));
    this.add(new Message(31, "Bye"));
    this.add(new Message(11, "Good Bye"));
    this.add(new Message(37, "Wow"));
    this.add(new Message(92, "Hey"));
    this.add(new Message(77, "Hi"));
    this.add(new Message(71, "Ok"));
}

public MessageList() { }

@Override
public String toString() {
    String str = "";
    for(Message msg: this) {
        str += msg;
    }
    return str;
}
}
```

## Б) к экземпляру запроса в методе doGet() сервлета добавим атрибут lst

```
@WebServlet(name = "ServletEL", urlPatterns = "/ServletEL")
public class ServletEL extends HttpServlet {

    protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {

        MessageList list = new MessageList();
        request.setAttribute("lst", list);
        request.getRequestDispatcher("jstl_foreach.jsp").forward(request, response);
    }
}
```

## B) JSP – ВЫВОД СПИСКА

```
<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<html>
<head><title>Core: forEach</title></head>
<body>
<table>
  <c:forEach var="elem" items="${lst}" varStatus="status">
    <tr>
      <td><c:out value="{ elem }" /></td>
      <td><c:out value="{ elem.id }" /></td>
      <td><c:out value="{ elem.text }" /></td>
      <td><c:out value="{ status.count }" /></td>
    </tr>
  </c:forEach>
</table>
</body></html>
```

ДОСТУП

Размещение коллекции

Число элементов

Message [id=72, text=Hello]	72 Hello	1
Message [id=31, text=Bye]	31 Bye	2
Message [id=11, text=Good Bye]	11 Good Bye	3
Message [id=37, text=Wow]	37 Wow	4
Message [id=92, text=Hey]	92 Hey	5
Message [id=77, text=Hi]	77 Hi	6
Message [id=71, text=Ok]	71 Ok	7

```
<c:forEach var="elem" items="${lst}" varStatus="status"
begin="2" end="4">
```

## Диапазон

```
<c:forEach var="elem" items="${ lst }" varStatus="status"
step="2">
```

## Изменение размера шага движения

относительная нумерацию в порядке  
доступа к элементам в цикле



```
<  
<c:set var="str" value="1, 2 - 3 : 4 . 5" />  
<c:forTokens var="token" items="${ str }" delims=".,-:)">  
  <c:out value="${ token }" /><br/>  
</c:forTokens>
```

СПИСОК СИМВОЛОВ-  
разделителей, отбрасываемых  
при выделении подстрок



1  
2  
3  
4  
5

## ► Включение ресурсов

```
<c:import url="\WEB-INF\jspf\footer.jspf"
          charEncoding="utf-8" />
<c:import url="\jspf\header.jspf" charEncoding="utf-8" >
  <param name="title" value="import header info"/>
</c:import>
<c:import url="ftp://somestore.com/project/sample.war"/>
<c:import url="\jsp\calendar.jsp" />
```

## ► Динамические адреса и перенаправление

```
<c:redirect url="jsp/destination.jsp">
  <c:param name="firstname" value="N"/>
  <c:param name="lastname" value="P"/>
</c:redirect>
```

передаются со строкой запроса

destination.jsp

```
<c:forEach var="ps" items="${param}">
  <c:out value="${ps.key} : ${ps.value}"/><br/>
</c:forEach>
```

Выдача имен переданных параметров и их значений

# JSTL formatting

теги форматирования и  
интернационализации

для страницы JSP

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

для JSP-документа

```
<jsp:root version="1.2" xmlns:fmt="http://java.sun.com/jsp/jstl/fmt">
```

## Теги интернационализации:

`<fmt:setLocale/>` — устанавливает региональные установки для страницы на основе объекта класса `Locale`;

`<fmt:setBundle/>`, `<fmt:bundle/>` — устанавливают объект `ResourceBundle`

`<fmt:message/>` — извлекает локализованное сообщение из ресурса, определенного тегами `<fmt:setBundle/>` или `<fmt:bundle/>`;

`<fmt:requestEncoding />` — с атрибутом `value="utf-8"` устанавливает кодировку входящего запроса.

## Теги форматирования дат и чисел:

`<fmt:timeZone/>`, `<fmt:setTimeZone/>` — устанавливает часовой пояс, используемый для форматирования;

`<fmt:formatNumber/>`, `<fmt:formatDate/>` — форматирует числа/даты с учетом установленной локали (региональных установок) либо указанного шаблона;

`<fmt:parseNumber/>`, `<fmt:parseDate/>` — переводит строковое представление числа/даты в объекты подклассов `Number` / `Date`.

```

<%@ page language="java" contentType="text/html; charset=UTF-8" p
<%@ taglib prefix="fmt" uri="http://java.sun.com/jsp/jstl/fmt" %>
<html>
  <head>
    <title>Формат даты</title>
  </head>
  <body>
    <jsp:useBean id="now" class="java.util.Date"
    <fmt:setLocale value="en-EN" />
      Вывод даты в формате English<br/>
    Сегодня: <fmt:formatDate value="${now}" /><br/>
      <fmt:setLocale value="ru-RU" />
    Вывод даты в формате Russian<br/>
    Сегодня:      <fmt:formatDate value="${now}" /><br/>
    Стил ь времени:
    (short): <fmt:formatDate value="${now}"
              type="time" timeStyle="short" /><br/>
    (medium):<fmt:formatDate value="${now}"
              type="time" timeStyle="medium" /><br/>
    (long): <fmt:formatDate value="${now}"
             type="time" timeStyle="long" /><br/>
  </body>
</html>

```

---

Вывод даты в формате English  
 Сегодня: May 12, 2016  
 Вывод даты в формате Russian  
 Сегодня: 12.05.2016  
 Стил ь времени: (short): 10:54  
 (medium):10:54:29  
 (long): 10:54:29 EEST

```
<c:set var="currentNumber" value="118000"/>
<c:out value="Вывод формата числа : ${currentNumber}"/> <br/>
    Формат (по умолчанию) :
        <fmt:formatNumber value="${currentNumber}" /><br/>
    Процентный формат :
        <fmt:formatNumber value="${currentNumber}"
            type="percent"/><br/>
        <fmt:setLocale value="be-BY"/>
    Белорусские рубли :
        <fmt:formatNumber value="${currentNumber}"
            type="currency"/><br/>
    Французская валюта :
        <fmt:setLocale value="fr-FR"/>
        <fmt:formatNumber value= "${currentNumber}«
            type="currency"/><br/>
```

Вывод формата числа : 118000  
Формат (по умолчанию) : 118 000  
Процентный формат : 11 800 000%  
Белорусские рубли : Руб118 000  
Французская валюта : 118 000,00 €

```
<%@ page language="java" contentType="text/html; charset=utf-8" %>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt" prefix="fmt"%>
<fmt:setLocale value="en_US" scope="session" />
<fmt:setBundle basename="resource.pagecontent" var="rb" />
<html>
  <head>
    <title>
      <fmt:message key="label.title" bundle="${ rb }" />
    </title>
  </head>
  <body>
    <fmt:message key="label.welcome" bundle="${ rb }" />
  <hr/>
    <fmt:message key="footer.copyright" bundle="${ rb }" />
  </body>
</html>
```

```
# Locale en_US
label.title=Example
label.welcome=Welcome!
footer.copyright=Copyright 2018
```

```
# Locale ru_RU
label.title=Пример
label.welcome=Добро пожаловать!
footer.copyright=Все права защищены 2018
```



# JSTL sql

- ▶ выполнение запросов SQL непосредственно из JSP и обработки результатов запроса в JSP

```
<%@
```

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

— для обычной страницы JSP

```
<jsp:root version="1.2" xmlns:sql="http://java.sun.com/jsp/jstl/sql">
```

— для JSP-документа

## Теги:

`<sql:dateParam>` — определяет параметры

`<sql:param>` — определяет параметры `<sql:query>` либо `<sql:update>`;

`<sql:query>` — выполняет запрос к БД;

`<sql:setDataSource>` — устанавливает data source (пула соединений) для `<sql:query>`, `<sql:update>`, и `<sql:transaction>` тегов;

`<sql:transaction>` — объединяет внутренние теги `<sql:query>` и `<sql:update>` в одну транзакцию;

`<sql:update>` — выполняет запрос на вставку/удаление/изменение БД.

# JSTL xml

для импорта, парсинга и обработки  
данных из XML-документов в документе  
JSPX

```
<jsp:root version="1.2" xmlns:x= "http://java.sun.com/jsp/jstl/xml"
```

## Список тегов:

`<x:set>` — XML-версия тега `<c:set>`;

`<x:out>` — XML-версия тега `<c:out>`;

`<x:forEach>` — XML-версия тега `<c:forEach>`;

`<x:if>` — XML-версия тега `<c:if>`;

`<x:choose>` — XML-версия тега `<c:choose>`;

`<x:when>` — XML-версия тега `<c:when>`;

`<x:otherwise>` — XML-версия тега `<c:otherwise>`;

`<x:parse>` — разбор XML-документа;

`<x:transform>` — трансформация XML-документа с применением XSLT-преобразования;

`<x:param>` — XML-версия тега `<c:param>`, определяющая параметры для другого тега `<x:transform>`.

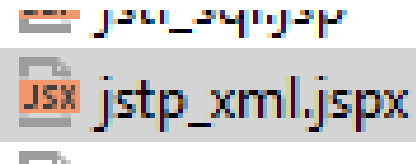
```

<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page"
          xmlns:x="http://java.sun.com/jsp/jstl/xml"
          version="2.0">
  <jsp:directive.page contentType="text/html; charset=UTF-8"/>

  <html>
  <head><title>XML inside</title></head>
  <body>
    <!--using tag x:parse-->
    <x:parse var="doc">
      </x:parse>
    <!--using tags x:forEach and x:out-->
    <x:forEach select="$doc/students/student" var="stud">
      Name:
      <x:out select="$stud/name"/><br/>
      Login:
      <x:out select="$stud/@login" /><br/>
      Faculty:
      <x:out select="$stud/@faculty" /><br/>
      <hr/><br/>
    </x:forEach>
  </body></html>
</jsp:root>

```

выполняет разбор  
документа, помещает  
построенный объект в  
переменную doc



```
<x:parse var="doc">
  <c:import url="../xml/students.xml" />
</x:parse>
<!--using tag x:set-->
Student name:
<x:set var="studentName"
      select="$doc/students/student[1]/name" />
<x:out select="$studentName/" />
```

```
<c:import url="../xml/students.xml" var="studXML"/>
<x:parse var="doc" doc="{studXML}" />
```

# JSTL functions

копируют известные методы класса String

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

## Список тегов:

`${fn:length(аргумент)}` — подсчитывает число элементов в коллекции или длину строки;

`${fn:toUpperCase(String str)}` и `${fn:toLowerCase(String str)}` — изменяет регистр строки;

`${fn:substring(String str, int from, int to)}` — извлекает подстроку;

`${fn:substringAfter(String str, String after)}` и `${fn:substringBefore(String str, String before)}` — извлекает подстроку до или после указанной во втором аргументе;



`{fn:trim(String str)}` — обрезает все пробелы по краям строки;

`{fn:replace(String str, String str1, String str2)}` — заменяет все вхождения строки `str1` на строку `str2`;

`{fn:split(String str, String delim)}` — разбивает строку на подстроки, используя `delim`, как разделитель;

`{fn:join(массив, String delim)}` — соединяет массив в строку, вставляя между элементами подстроку `delim`;

`{fn:escapeXml(String xmlString)}` — сохраняет в обрабатываемой строке теги;

`{fn:indexOf(String str, String searchString)}` — возвращает индекс первого вхождения строки `searchString`;

`<c:if test="\${fn:startsWith(String str, String part)}">` — возвращает истину, если строка начинается с подстроки `part`;

`<c:if test="\${fn:endsWith(String str, String part)}">` — возвращает истину, если строка заканчивается на подстроку `part`;

`<c:if test="\${fn:contains(String name, String searchString)}">` — возвращает истину, если строка содержит подстроку `searchString`;

`<c:if test="\${fn:containsIgnoreCase(String name, String searchString)}">` — возвращает истину, если строка содержит подстроку `searchString`, без учета регистра.

# ПОЛЬЗОВАТЕЛЬСКИЕ ТЕГИ

- ▶ механизм расширения API JSP
- ▶ Перенос java-кода из страницы JSP в обычный java-класс (страница JSP должна содержать как можно меньше логики)

# Создание пользовательских тегов

1) класс обработчика тега (определяет поведение)

2) дескрипторный файл библиотеки тегов  
Tag Library Descriptor (файл .tld) (описание тегов и установка соответствия)

## ► 1. Создание класса

Требования:

- реализация интерфейса  
`javax.servlet.jsp.tagext.Tag`

`TagSupport`



`BodyTagSupport`

- `import` классов `javax.servlet.jsp`  
`java.io` и др.

# Пример создания тега

- ▶ без атрибутов, тела, информации из страницы

doStartTag()

возврат

SKIP\_BODY

игнорировать любое содержимое между начальными и конечными элементами создаваемого тега

возврат

EVAL\_BODY\_INCLUDE

включить тело тега в поток вывода

`int doAfterBody()` — вызывается после обработки тела

возврат

возврат

`EVAL_BODY_AGAIN`  
метод будет вызван еще раз после  
вывода в поток тела тега

`SKIP_BODY`  
вызов метода `doEndTag()`



`int doEndTag()` - один раз, когда отработаны все остальные методы

возврат

возврат

`EVAL_PAGE`

разрешает дальнейшую обработку страницы

`SKIP_PAGE`

прекратить дальнейшую обработку страницы



TagSupport : поле id  
поле pageContext:PageContext

Доступ:

ServletRequest getRequest()

ServletResponse getResponse()

ServletContext getServletContext()

ServletConfig getServletConfig()

HttpSession getSession()

JspWriter getOut() — **ПОТОК ВЫВОДА**

ErrorData getErrorData() —  
**информация об ошибках**

+

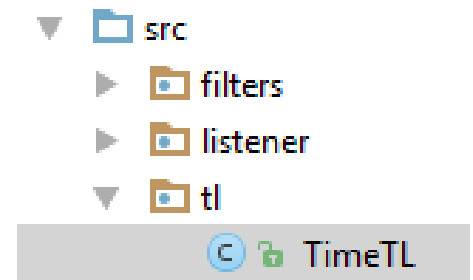
forward(String relativeUrlPath);

include(String relativeUrlPath) - **текущие  
ресурсы ServletRequest или ServletResponse,  
определяемые относительным адресом**

```
package tl;

import javax.servlet.jsp.JspException;
import javax.servlet.jsp.JspWriter;
import javax.servlet.jsp.tagext.TagSupport;
import java.io.IOException;
import java.util.GregorianCalendar;
import java.util.Locale;
```

```
@SuppressWarnings("serial")
public class TimeTL extends TagSupport {
    @Override
    public int doStartTag() throws JspException {
        GregorianCalendar gc = new GregorianCalendar();
        String time = "<hr/>Time : <b> " + gc.getTime() + " </b><hr/>";
        try {
            JspWriter out = pageContext.getOut();
            out.write(time);
        } catch (IOException e) {
            throw new JspException(e.getMessage());
        }
        return SKIP_BODY;
    }
    @Override
    public int doEndTag() throws JspException {
        return EVAL_PAGE;
    }
}
```



## 2. Идентификация класса для сервера и связывание его с именем XML-тега

Дескрипторный файла библиотеки тегов \*.tld

- 1) `<taglib>` - корневой элемент
- 2) `< tlib-version >` — версия
- 3) `< short-name >` — краткое имя
- 4) `<uri>` — имя для доступа из JSP в директиве taglib
- 5) `< info >` — область применения библиотеки
- 6) `<tag>` - описание тегов в элементах
  - 1) `name` — имя базового тега (+ `short-name` )
  - 2) `tag-class` — полное имя класса-обработчика тега;
  - 3) `info` — краткое описание тега;
  - 4) `body-content` — тип тела тега: `empty`; `scriptless`; `tagdependent`

# Пример дескрипторного файла (TLD )

```
<?xml version="1.0" encoding="ISO-8859-1" ?>
```

```
<taglib xmlns="http://java.sun.com/xml/ns/javaee"  
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"  
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee  
http://java.sun.com/xml/ns/javaee/web-jsptaglibrary_2_1.xsd"  
        version="2.1">
```

```
<tlib-version>1.0</tlib-version>
```

```
<short-name>pnvtdg</short-name>
```

```
<uri>customtags</uri>
```

```
<tag>
```

```
<name>info-time</name>
```

```
<tag-class>tl.TimeTL</tag-class>
```

```
<body-content>empty</body-content>
```

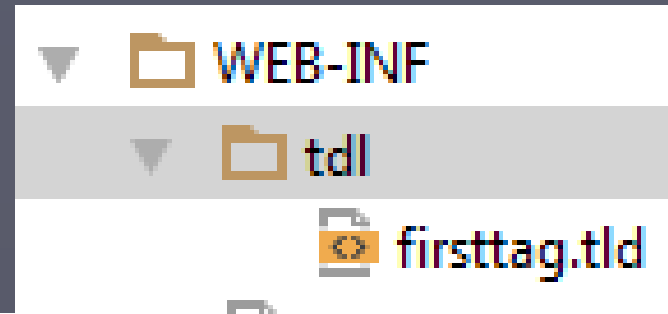
```
</tag>
```

```
</taglib>
```



### 3. Зарегистрировать библиотеку пользовательских тегов

a) web.xml - <jsp-config>



```
<jsp-config>
  <taglib>
    <taglib-uri>customtags</taglib-uri>
    <taglib-location>/WEB-INF/tdl/firsttag.tld</taglib-location>
  </taglib>
</jsp-config>
```

### Подключение в JSP

```
<%@ taglib prefix="pnvtdl" uri="customtags" %>
```

б) не регистрируем в web.xml

Доступ - прямой путь к tld-файлу

```
<%@ taglib prefix="pnvtld" uri="../../../WEB-INF/tld/firsttag.tld" %>
```

(доступ удаленный)

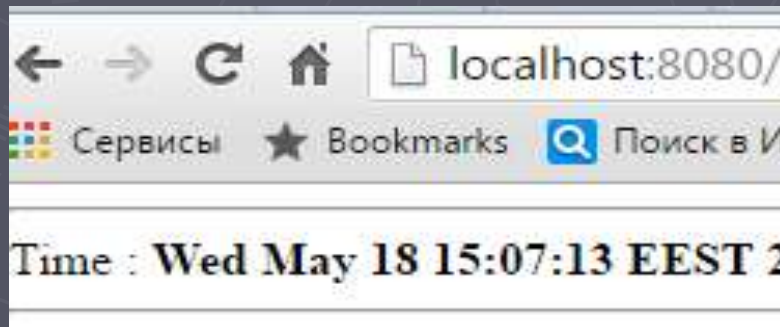
в) упаковка в jar и доступ как к библиотеке

```
<%@ taglib prefix="pnvtld" uri="../../../WEB-INF/lib/firsttag.jar" %>
```

# jsp

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<html>
<head>
  <title> Time form tdl </title>
</head>
<body>
  <%@ taglib prefix="pnvtdl" uri="customtags" %>
  <%--<%@ taglib prefix="pnvtdl" uri="../WEB-INF/tld/firsttag.tld" %>--%>
  <%--<%@ taglib prefix="pnvtdl" uri="../WEB-INF/lib/firsttag.jar" %>--%>

  <pnvtdl:info-time/>
</body>
</html>
```



# Атрибуты тегов

<attribute>    </attribute>:

- ▶ name — (o) имя атрибута;
- ▶ required — (o) true -false;
- ▶ rtexprvalue — (н)
  - ▶ true                    атрибут -    \${expression}
  - ▶ false (default) атрибут - строка данных
- ▶ type — (н) тип атрибута (def - java.lang.String)



# 1. Определение класса

```
@SuppressWarnings("serial")
public class AttrTL extends TagSupport {
    private String usertype;
    public void setUserType(String usertype) {
        this.usertype = usertype;
    }
    @Override
    public int doStartTag() throws JspException {
        try {
            String to = null;
            if ("admin".equalsIgnoreCase(usertype)) {
                to = "Welcome, you have " + usertype + " permission";
            } else {
                to = "No permission for this page " + usertype;
            }
            pageContext.getOut().write("<hr/>" + to + "<hr/>");
        } catch (IOException e) {
            throw new JspException(e.getMessage());
        }
        return SKIP_BODY;
    }
}
```

Поле атрибута

Проверка значения

# 2.usertype.tld

```
<?xml version="1.0" encoding="ISO-8859-1"?>
```

```
<taglib xmlns="http://java.sun.com/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xsi:schemaLocation="http://java.sun.com/xml/ns/javaee http://jav
        version="2.1">
```

```
<tlib-version>1.0</tlib-version>
```

```
<short-name>pnvtag</short-name>
```

```
<uri>custom2</uri>
```

Имя базового  
тега

```
<tag>
```

```
<name>attr</name>
```

Имя класса

```
<tag-class>tl.AttrTL</tag-class>
```

Имя атрибута

```
<body-content>empty</body-content>
```

```
<attribute>
```

```
<name>usertype</name>
```

```
<required>true</required>
```

```
<rteprvalue>true</rteprvalue>
```

```
</attribute>
```

```
</tag>
```

```
</taglib>
```


### 3. web.xml

```
<jsp-config>
  <taglib>
    <taglib-uri>custom2</taglib-uri>
    <taglib-location>/WEB-INF/tdl/usertype.tld</taglib-location>
  </taglib>
</jsp-config>
```

# tdl\_usertype.jsp

```
<%@ page language="java" contentType="text/html;  
    charset=UTF-8" pageEncoding="UTF-8"%>  
<%@ taglib prefix="pnvtld" uri="custom2"%>  
  
<html>  
    <head>  
        <title>tag: User Type</title>  
    </head>  
    <body>  
  
        ${ pageContext.request.setAttribute("user", "admin") }  
  
        <pnvtld:attr usertype="${ user}"/>  
    </body>  
</html>
```

УСТАНОВИМ  
ЗНАЧЕНИЕ В  
request



---

Welcome, you have admin permission

---

# Тело тега

В файле .tld

`<tag>`

`<body-content>scriptless</body-content>`

```
<pnvtdl:table rows="{rw.size}" head="Таблица 1"
    ${rw.revenue}
</pnvtdl:table>
<br/>
<pnvtdl:table>Содержимое таблицы</pnvtdl:table>
```

```
request.setAttribute("rw", map);
```

# класса BodyTagSupport:

doStartTag() → return EVAL\_BODY\_INCLUDE  
return EVAL\_BODY\_BUFFERED

↓  
doInitBody()

↓  
первая обработка тела

тело тега → экземпляр класса BodyContent

```
@SuppressWarnings("serial")
public class BobbyTag extends BodyTagSupport {

    @Override
    public int doAfterBody() throws JspException {
        BodyContent content = this.getBodyContent();
        String body = content.getString();
        String res = null;
        if ( body.contains("F")) {
            res = body.replaceAll("\\.", "?");
            res = res.replaceFirst("A", "a");
        } else {
            res = body;
        }
        JspWriter out = content.getEnclosingWriter();
        try {
            out.write(res);
        } catch (IOException e) {
            throw new JspTagException(e.getMessage());
        }
        return SKIP_BODY;
    }
}
```

# Функции–теги

Назначение:

- ▶ Валидация
- ▶ Форматирование
- ▶ Преобразование

класс

```
public class FuncTL {  
    public static String CheckForNull(Object ob) {  
        //...  
        return res;  
    }  
}
```

Возврат значения



```
public class FuncTL {  
    public static String checkForNull(Object ob) {  
        String res = null;  
        if (ob == null || ob.toString().isEmpty()) {  
            res = "Атрибут нулевой";  
        } else {  
            res = ob.toString();  
        }  
        return res;  
    }  
}
```

## Функция проверки

# .tld

```
<tlib-version>1.0</tlib-version>
```

```
<short-name>pnvtg</short-name>
```

```
<uri>custom3</uri>
```

путь к классу-описанию



```
<function>
```

```
  <name>check</name>
```

```
  <function-class>tl.FuncTL</function-class>
```

```
  <function-signature>
```

```
    java.lang.String checkForNull(java.lang.Object)
```

```
  </function-signature>
```

```
</function>
```

сигнатуру функции с параметрами



# .jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib prefix="pnvtg" uri="custom3"%>
<html>
<head>
    <title>
        tag: function
    </title>
</head>
<body>
    Не пустой параметр: ${pnvtg:check("Проверка строки")}
    <br/>
    Строка нулевой длины: ${pnvtg:check("")}

</body>
</html>
```

Не пустой параметр: Проверка строки  
Строка нулевой длины: Атрибут нулевой

Атрибуты и параметры с request

# Элементы action для тегов

## Простой строковый атрибут

```
<pnvtdl:attr usertype="admin" />
```

## Сложное содержимое

```
<pnvtdl:attr usertype="VIP" />
```

```
<pnvtdl:attr>  
  <jsp:attribute name="usertype">  
    "VIP"  
  </jsp:attribute>  
</pnvtdl:attr>
```

# Атрибут и тело

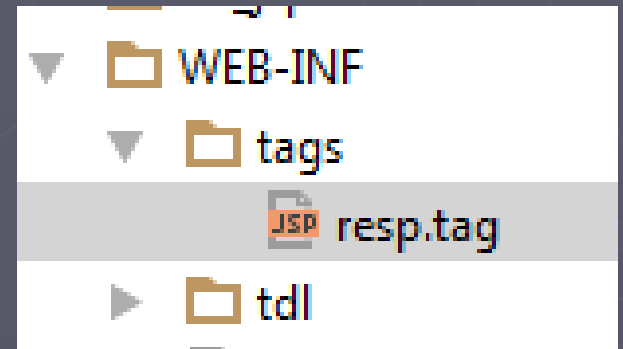
```
<pnvtdl:table>  
    <jsp:attribute name="rows">  
        ${requestScope.rw.size}  
    </jsp:attribute>  
    <jsp:body>  
        ${requestScope.rw.revenue}"  
    </jsp:body>  
</pnvtdl:table>
```

TDL

<http://docs.oracle.com/javaee/5/tutorial/doc/bnalj.html>

# Создание тегов на основе файлов .tag

.tag -> tag header -> компиляция



```
<%@ attribute name="greeting" required="true" %>
<%@ attribute name="name" required="true" %>
<h2><font color="black">${greeting}, ${name}!</font></h2>
```

# .jsp

```
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%@ taglib prefix="h" tagdir="/WEB-INF/tags" %>
<%@ 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>
      Hello
    </title>
  </head>

  <body bgcolor="white">

    <c:set var="greeting" value="Привет" />
    <h2>${greeting}. Введите ваше имя</h2>
    <form method="get">
      <input type="text" name="username" size="25">
      <p></p>
      <input type="submit" value="Submit">
      <input type="reset" value="Reset">
    </form>

    <c:if test="${fn:length(param.username) > 0}" >
      <h:resp greeting="${greeting}"
              name="${param.username}" /></c:if>
    </body>
  </html>
```

Привет. Введите ваше имя

Submit Reset

Привет, Петя!

# Директивы .tag

taglib	Аналогично директиве для JSP
include	Аналогично директиве для JSP
tag	Аналогично директиве для JSP
attribute	
variable	EL



