

JAVA FOR STUDENTS

ФИЛИАЛ КАФЕДРЫ ПОИТ БГУИР В ЕРАМ
SYSTEMS

КУРС: ВЕБ-ТЕХНОЛОГИИ (JAVA)
SERVLETS FUNDAMENTALS

Olga Smolyakova , PhD

Oracle Certified Java 6 Programmer

Olga_Smolyakova@epam.com

Содержание

1. `<jsp:usebean>`
2. Конструкции JSP
3. Неявные объекты на jsp-странице
4. Статическое и динамическое содержимое
5. JSTL, обзор
6. Expression Language
7. JSTL: core tags
8. JSTL: fmt tags
9. JSTL: sql, xml tags, functions

JSP:USEBEAN

index.jsp

```
...  
<body>  
    <form action="ServletForJspElement" method="post">  
        <input type="hidden" name="command"  
            value="naming" /> Введите имя:<br />  
        <input type="text" name="name" value="" />  
        <br /> Введите фамилию:<br />  
        <input type="text" name="surname" value="" /><br />  
        <input type="submit" value="Отправить" /><br />  
    </form>  
</body>  
...
```

SimpleBean.java

```
package _java._ee._02._servlet;

import java.util.Date;

public class SimpleBean {
    private String name;
    private String surname;
    private Date date;

    public String getName() { return name; }
    public void setName(String name) { this.name = name; }
    public String getSurname() { return surname; }
    public void setSurname(String surname) { this.surname = surname; }
    public Date getDate() { return date; }
    public void setDate(Date date) { this.date = date; }
}
```

ServletForJSPElement.java

```
package _java._ee._02._servlet;
import ...
public class ServletForJspElement extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(
        HttpServletRequest request,
        HttpServletResponse response)
        throws ServletException, IOException {
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html");
        if ("naming".equals(request.getParameter("command"))) request.
            getRequestDispatcher("jspusebean/usebean.jsp").
            forward(request, response);
    }
}
```

usebean.jsp

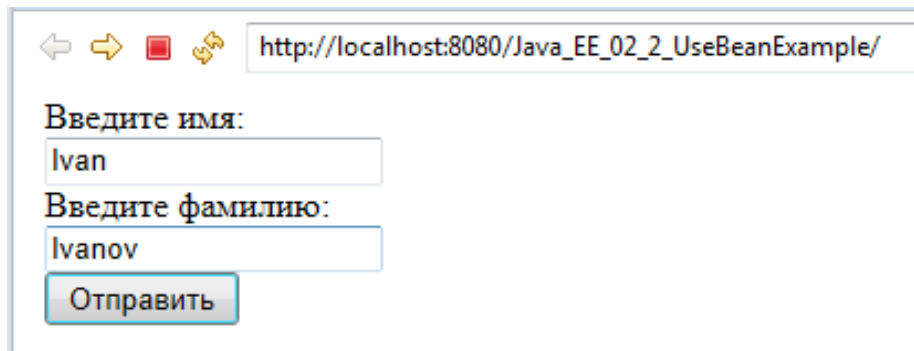
```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
...
<body>
<jsp:useBean id="naming" class="_java._ee._02._servlet.SimpleBean" />

    <jsp:setProperty property="*" name="naming"/>
    <jsp:getProperty property="name" name="naming"/>
    <jsp:getProperty property="surname" name="naming"/>
    <jsp:getProperty property="date" name="naming"/>

    <jsp:useBean id="pageDate" class="java.util.Date" />
    <jsp:setProperty name="naming" property="date" value="${pageDate}" />
    <jsp:getProperty property="date" name="naming"/>

    <c:out value="${pageScope.naming.name}" />
</body>
...
```

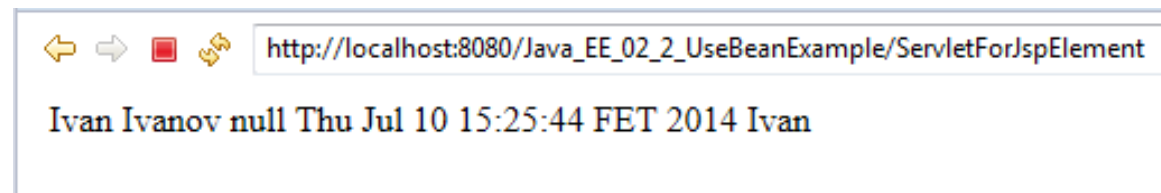
Результат:



← → ■ 💰 http://localhost:8080/Java_EE_02_2_UseBeanExample/

Введите имя:

Введите фамилию:



← → ■ 💰 http://localhost:8080/Java_EE_02_2_UseBeanExample/ServletForJspElement

Ivan Ivanov null Thu Jul 10 15:25:44 FET 2014 Ivan

ServletForJSPElement.java

```
package _java._ee._02._servlet;

import ...

public class ServletForJspElement extends HttpServlet {
    private static final long serialVersionUID = 1L;

    protected void doPost(HttpServletRequest request,
                           HttpServletResponse response)
        throws ServletException, IOException {
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html");
        SimpleBean s = new SimpleBean();
        s.setName(request.getParameter("name"));
        s.setSurname(request.getParameter("surname"));
        Object a = s;
        request.setAttribute("mybean", a);
        if ("naming".equals(request.getParameter("command"))) request.
            getRequestDispatcher("jspusebean/usebean.jsp").
            forward(request, response);
    }
}
```

usebean.jsp

```
<%@ page language="java" contentType="text/html; charset=ISO-8859-1"
    pageEncoding="ISO-8859-1"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
...
<body>
<jsp:useBean id="mybean" class="_java._ee._02._servlet.SimpleBean"
type="java.lang.Object" scope="request"/>

    <jsp:getProperty property="name" name="mybean"/>
    <jsp:getProperty property="surname" name="mybean"/>
    <jsp:getProperty property="date" name="mybean"/>

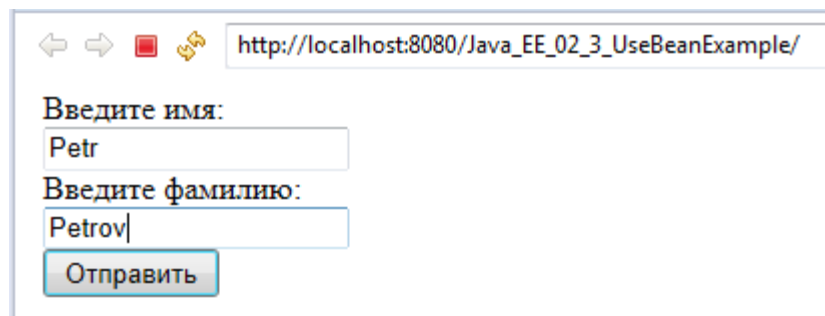
    <jsp:useBean id="pageDate" class="java.util.Date" />

    <c:set target="${mybean}" property="date" value="${pageDate}"/>
    <jsp:getProperty property="date" name="mybean"/>

    <c:out value="${requestScope.mybean.name}" />

</body>
...
```

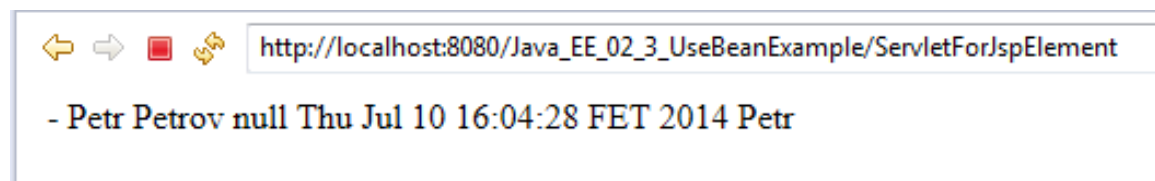
Результат:



← → 🛑 💰 http://localhost:8080/Java_EE_02_3_UseBeanExample/

Введите имя:

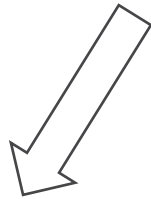
Введите фамилию:



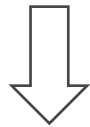
⏮ ⏭ 🛑 💰 http://localhost:8080/Java_EE_02_3_UseBeanExample/ServletForJspElement

- Petr Petrov null Thu Jul 10 16:04:28 FET 2014 Petr

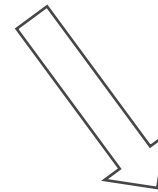
```
<jsp:useBean id="mybean" class="_java._ee._02._servlet.SimpleBean"
scope="request"/>
```



объект *mybean* есть
в контексте запроса



Идентификатор
id="mybean"
будет ассоциирован
с существующим
объектом.



объекта *mybean* нет в
контексте запроса

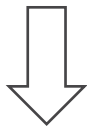


Создастся новый объект типа
"_java._ee._02._servlet.SimpleBean"
и идентификатор *id="mybean"*
будет ассоциирован с новым
объектом.

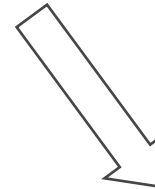
```
<jsp:useBean id="mybean" type="_java._ee._02._servlet.SimpleBean"
scope="request"/>
```



объект *mybean* есть в контексте
запроса и ссылка типа
`_java._ee._02._servlet.SimpleBean`
может на него ссылаться



Идентификатор
`id="mybean"` будет
ассоциирован с
существующим объектом.



объекта *mybean* нет в
контексте запроса

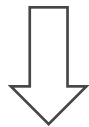


Идентификатор `id="mybean"` не
будет ассоциирован ни с каким
объектом.

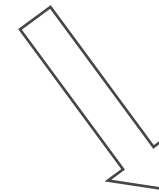
```
<jsp:useBean id="mybean"  
    beanName="_java._ee._02._servlet.SimpleBean"  
    type="_java._ee._02._servlet.SimpleBean" scope="request"  
>
```



объект *mybean* есть в контексте
запроса и ссылка типа
`_java._ee._02._servlet.SimpleBean`
может на него ссылаться



Идентификатор
`id="mybean"` будет
ассоциирован с
существующим объектом.



объекта *mybean* нет в
контексте запроса



Создастся новый объект с типом,
указанным в `beanName` и
идентификатор *mybean* будет
ассоциирован с ним..


НЕЯВНЫЕ ОБЪЕКТЫ НА JSP-СТРАНИЦЕ

Неявные объекты (implicit objects) - это объекты, автоматически доступные как часть стандарта JSP без их специального объявления или импорта. Эти объекты можно использовать в коде JSP.

- **request (запрос)** - javax.servlet.HttpServletRequest
- **response (ответ)** - javax.servlet.HttpServletResponse
- **out (вывод)** - javax.servlet.jsp.JspWriter
- **pageContext (содержание страницы)** - javax.servlet.jsp.pageContext
- **session (сеанс)** - javax.servlet.HttpSession
- **application (приложение)** - javax.servlet.ServletContext
- **config (конфигурация)** - javax.servlet.ServletConfig
- **page (страница)** - java.lang.Object
- **exception (исключение)** - java.lang.Throwable

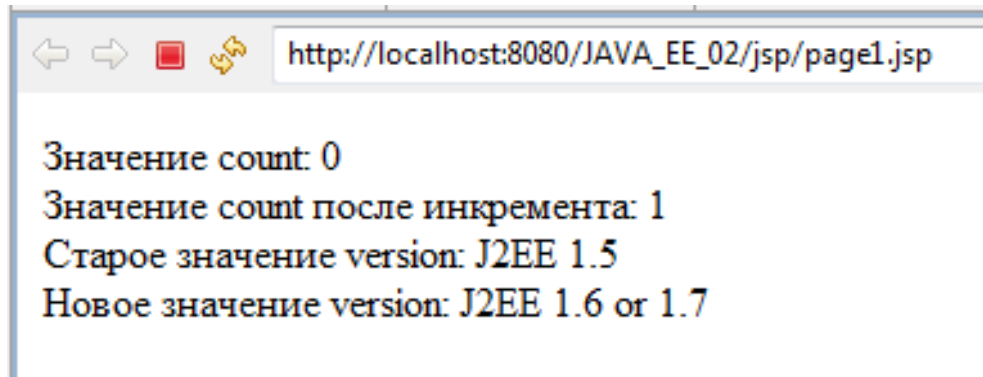
jsp-синтаксис

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<html>
<head>
<title>JSP-страница</title>
</head>
<%!
    private int count = 0;
    String version = new String("J2EE 1.5");
    private String getName() {
        return "J2EE 1.6 or 1.7";
    }%>
<%
    out.println("Значение count: ");
    %=count++%>
<br />
<%
    out.println("Значение count после инкремента: " + count);
    %>
<br />
```



```
<%  
    out.println("Старое значение version: ");  
%>  
<%=version%>  
<br />  
<%  
    version = getName();  
    out.println("Новое значение version: " + version);  
%>  
</html>
```

Результат:



xml-синтаксис

```
<jsp:root xmlns:jsp="http://java.sun.com/JSP/Page" version="2.0">
  <jsp:directive.page contentType="text/html; charset=UTF-8"
/>
<html>
<body>
  <jsp:declaration>private int count = 0;
  String version = new String("J2EE 1.5");
  private String getName() {
    return "J2EE 1.6";
  }</jsp:declaration>
  <jsp:scriptlet>out.println("Значение count: ");
</jsp:scriptlet>
  <jsp:expression>count++</jsp:expression>
  <br />
  <jsp:scriptlet>out.println("Значение count после
инкремента:" + count);
</jsp:scriptlet>
```



```
<br />
<jsp:scriptlet>out.println("Старое значение version: ");
</jsp:scriptlet>
<jsp:expression>version</jsp:expression>
<br />
<jsp:scriptlet>version = getName();
                out.println("Новое значение version: " + version);
</jsp:scriptlet>
</body>
</html>
</jsp:root>
```

ЭЛЕМЕНТЫ JSP, ОБЗОР

JSP elements

Directives

page

include

taglib

Scripting Elements

Declarations

Expressions

Scriptlets

Comments

JSP Actions

<jsp:useBean>

<jsp:setProperty>

<jsp:getProperty>

<jsp:include>

<jsp:forward>

<jsp:plugin>

<jsp:param>

JSP SCRIPTING ELEMENTS

Объявляет переменную или метод действительными для скриптового языка, используемого на JSP-странице.

JSP Syntax

```
<%! declaration; [ declaration; ]+ ... %>
```

XML Syntax

```
<jsp:declaration>  
Code fragment [ declaration; ] + ...  
</jsp:declaration>
```

Examples

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

Scripting
Elements

Declarations

Expressions

Scriptlets

Comments

Содержит правильное выражение скриптового языка, используемого на JSP-странице.

JSP Syntax

```
<%= expression %>
```

XML Syntax

```
<jsp:expression> expression </jsp:expression>
```

Examples

The map file has `<%= map.size() %>` entries.
Good guess, but nope. Try `<%= numguess.getHint() %>`

Выражение – правильное выражение языка Java, которое может корректно приводится к типу String. Результат выражения включается в формируемый ответ

Scripting
Elements

Declarations

Expressions

Scriptlets

Comments

Содержат правильные участки кода, написанные на языке Java.

Скриплеты позволяют включать несколько java-операторов внутрь метода `_jspService()`

JSP Syntax

```
<% code fragment %>
```

XML Syntax

```
<jsp:scriptlet> code fragment </jsp:scriptlet>
```

Examples

```
<%      String name = null;
        if (request.getParameter("name") == null) {
%>
<%@ include file="error.html" %>
<%
        } else {
        foo.setName(request.getParameter("name"));
        if (foo.getName().equalsIgnoreCase("integra"))
        name = "acura";
        if (name.equalsIgnoreCase("acura")) {
%>
```

Scripting
Elements

Declarations

Expressions

Scriptlets

Comments

Комментарии, которые отправляются клиенту

JSP Syntax

```
<!-- comment [ <%= expression %> ] -->
```

Example 1

```
<!-- This file displays the user login screen -->
```

Результат:

```
<!-- This file displays the user login screen -->
```

Example 2:

```
<!-- This page was loaded on <%= (new  
java.util.Date()).toLocaleString() %> -->
```

Результат:

```
<!-- This page was loaded on January 1, 2000 -->
```

Scripting
Elements

Declarations

Expressions

Scriptlets

Comments

Комментарии jsp-страницы - игнорируются при трансляции JSP

JSP Syntax

```
<%-- comment --%>
```

Examples

```
<%@ page language="java" %>
<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>
```

Scripting
Elements

Declarations

Expressions

Scriptlets

Comments

JSP DIRECTIVES

Directives

Директивы – это инструкции jsp-компилятору. Эти инструкции указывают, какое действия должно быть предпринято компилятором.

Общий синтаксис директивы следующий

<%@ Directive-name attribute-value pairs %>

В jsp определены три основные директивы

page

include

taglib

Page Directive - Определяет атрибуты, которые относятся ко всей jsp-странице

JSP Syntax

```
<%@ page
[ language="java" ]
[ extends="package.class" ]
[ import="{package.class | package.*}, ..." ]
[ session="true | false" ]
[ buffer="none | 8kb | sizekb" ]
[ autoFlush="true | false" ]
[ isThreadSafe="true | false" ]
[ info="text" ]
[ errorPage="relativeURL" ]
[ contentType="mimeType [ ;charset=characterSet ]" ]
"text/html ; charset=ISO-8859-1" ]
[ isErrorPage="true | false" ]
%>
```

Directives

page

include

taglib

XML Syntax

```
<jsp:directive.page pageDirectiveAttrList />
```

`pageDirectiveAttrList` – аналогичен используемому при jsp-синтаксисе

Examples

```
<%@ page import="java.util.*, java.lang.*" %>

<%@ page buffer="5kb" autoFlush="false" %>

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

<jsp:directive.page errorPage="error.jsp" />
```

Директива `<%@ page %>` относится ко всей jsp-странице и ко всем статическим включаемым в нее файлам (все вместе называется единицей трансляции)

Directives

page

include

taglib

Директива **include** позволяет вставлять текст или код в процессе трансляции страницы JSP в сервлет. Синтаксис директивы **include** имеет следующий вид:

`<%@ include file="Относительный URI включаемой страницы" %>`

JSP Syntax

```
<%@ include file="relativeURL" %>
```

XHTML Syntax

```
<jsp:directive.include file="relativeURL" />
```

Directives

page

include

taglib

Examples

include.jsp:

```
<html>
<head>
<title>An Include Test</title>
</head>
<body bgcolor="white">
<font color="blue">
The current date and time are
<%@ include file="date.jsp"%>
</font>
</body>
</html>
```

date.jsp:

```
<%@ page
import="java.util.*" %>
<%= (new java.util.Date()
).toLocaleString() %>
```

Directives

page

include

taglib

Директива **include** рассматривает ресурс, например, страницу JSP, как статический объект.

Поскольку директива **include** подключает файлы в ходе трансляции страницы, то после внесения изменений в панель навигации требуется повторная трансляция всех использующих ее JSP страниц.

Directives

page

include

taglib

Директива **taglib** объявляет, что данная страница JSP использует библиотеку тегов, уникальным образом идентифицируя ее с помощью URI, и ставит в соответствие префикс тега, с помощью которого возможны действия в библиотеке. Если контейнер не может найти библиотеку тегов, возникает фатальная ошибка трансляции.

Директива **taglib** имеет следующий синтаксис:

```
<%@ taglib uri="URIToTagLibrary"  
           prefix="tagPrefix" %>
```

Directives

page

include

taglib

Example

```
<%@ taglib uri=http://www.jspcentral.com/tags  
                                prefix="public" %>  
  
<public:loop>  
.  
.  
</public:loop>
```

Префикс "имяПрефикса" используется при обращении к библиотеке.

Пример использования библиотеки тегов **mytags**.

```
<%@ taglib uri=http://www.taglib/mytags  
                                prefix="customs" %>  
  
.  
.  
.  
<customs:myTag>
```

JSP ACTIONS

JSP Actions

`<jsp:useBean>`

объявление объекта JavaBean, который будет использоваться на странице JSP

`<jsp:setProperty>`

установка значения свойства объекта JavaBean

`<jsp:getProperty>`

чтение значения свойства объекта JavaBean

`<jsp:include>`

включение в страницу JSP дополнительных статических и динамических ресурсов

`<jsp:forward>`

перенаправление обработки на другой статический ресурс, например сервлет

`<jsp:plugin>`

подключение дополнительных программных модулей (компонент JavaBean или апплет)

`<jsp:param>`

определение значения параметра

JSP Actions - действия JSP могут воздействовать на стандартный поток вывода, использовать, модифицировать и создавать объекты.

Существует набор стандартных действий, которые должны быть в обязательном порядке реализованы любым контейнером JSP, удовлетворяющим спецификации.

Кроме этого, возможно создание новых действий с помощью директивы библиотеки тегов *taglib*.

JSP Actions

<jsp:useBean>

<jsp:setProperty>

<jsp:getProperty>

<jsp:include>

<jsp:forward>

<jsp:plugin>

<jsp:param>

jsp:useBean

Тег **jsp:useBean** позволяет ассоциировать экземпляр Java-класса, определенный в данной области видимости **scope**, с заданным внутренним идентификатором этого класса **id** в данной странице JSP.

- **id** - идентификатор экземпляра класса внутри страницы JSP;
- **scope** - область видимости (page, request, session, application).

JSP Actions

<jsp:useBean>

<jsp:setProperty>

<jsp:getProperty>

<jsp:include>

<jsp:forward>

<jsp:plugin>

<jsp:param>

Синтаксис действия **jsp:useBean**:

```
<jsp:useBean id="идентификатор"
    scope = "page | request | session |
application"

    class = "имяКласса" type = "имяТипа" |
    type = "имяТипа" | class = "имяКласса"
|
    beanName = "имяКомпонентаJavaBean" |
type = "имяТипа" |
    type = "имяТипа" | beanName =
"имяКомпонентаJavaBean" |
    type = "имяТипа">
    <!-- Тело -->
</jsp:useBean>
```

JSP Actions

<jsp:useBean>

<jsp:setProperty>

<jsp:getProperty>

<jsp:include>

<jsp:forward>

<jsp:plugin>

<jsp:param>

jsp:setProperty

Тег **jsp:setProperty** позволяет присваивать значения свойствам компонента JavaBean.

Действие jsp:setProperty имеет следующий синтаксис:

```
<jsp:setProperty name="идентификатор"  
    property = "*" |  
    property = "имяСвойства" |  
    property = "имяСвойства" | param = "имяПараметра" |  
    property = "имяСвойства" | value = "значение" |  
    property = "имяСвойства" | value = <%= выражение %>  
>
```

JSP Actions

<jsp:useBean>

<jsp:setProperty>

<jsp:getProperty>

<jsp:include>

<jsp:forward>

<jsp:plugin>

<jsp:param>

Значения одного или нескольких свойств компонента JavaBean могут быть установлены несколькими способами:

- с помощью параметров объектов типа request (запрос);
- с использованием строковой константы;
- с помощью выражения, вычисляемого во время запроса.

Пример использования тега jsp:setProperty

```
<jsp:useBean id="user" class="hall.users" />
<jsp:setProperty name="user"
                 property="name" value="alex" />
<jsp:setProperty name="user"
                 property="name" value="serg" />
```

JSP Actions

<jsp:useBean>

<jsp:setProperty>

<jsp:getProperty>

<jsp:include>

<jsp:forward>

<jsp:plugin>

<jsp:param>

jsp:getProperty

Данный тег включает значение типа String или объект типа, преобразованный к типу String, в выходной поток.

```
<jsp:getProperty name="идентификатор"  
                property = "имяСвойства" />
```

Пример использования тега **jsp:getProperty**.

```
<jsp:useBean id="itemBean" ... />  
...  
<ul>  
    <li>Количество предметов :  
        <jsp:getProperty name="itemBean"  
                        property="numItems" /></li>  
    <li>Цена за штуку :  
        <jsp:getProperty name="itemBean"  
                        property="unitCost" /></li>  
</ul>
```

JSP Actions

<jsp:useBean>

<jsp:setProperty>

<jsp:getProperty>

<jsp:include>

<jsp:forward>

<jsp:plugin>

<jsp:param>

jsp:include

Действие jsp:include позволяет подключать статические и динамические ресурсы в контекст текущей страницы JSP.

В отличие от директивы include, которая вставляет файл на этапе трансляции страницы JSP, действие jsp:include вставляет файл при запросе страницы.

JSP Actions

<jsp:useBean>

<jsp:setProperty>

<jsp:getProperty>

<jsp:include>

<jsp:forward>

<jsp:plugin>

<jsp:param>

Синтаксис действия `jsp:include`

```
<!-- Первый вариант записи тега jsp:include -->
<jsp:include page="URLАдрес" [flush="true" | "false"] />

<!-- Второй вариант записи тега jsp:include -->
<jsp:include page="URLАдрес" [flush="true"]>
    <jsp:param .../>
    [<jsp:param .../>
    [...] ]
</jsp:include>
```

Примером может быть включение страницы-приветствия:

```
<jsp:include page="/general/welcome.html" />
```

JSP Actions

`<jsp:useBean>`

`<jsp:setProperty>`

`<jsp:getProperty>`

`<jsp:include>`

`<jsp:forward>`

`<jsp:plugin>`

`<jsp:param>`

jsp:forward

Действие jsp:forward позволяет во время выполнения страницы JSP перенаправлять текущий запрос на другую страницу JSP, некоторый статический ресурс или класс Java-сервлета, находящийся в том же контексте, что и текущая страница JSP.

```
<!-- Первый вариант записи тега jsp:forward -->
<jsp:forward page="URLАдрес" [flush="true" | "false"]
/>

<!-- Второй вариант записи тега jsp:forward -->
<jsp:forward page="URLАдрес" [flush="true"]>
    <jsp:param .../>
    [<jsp:param .../>
    [...] ]
</jsp:forward>
```

JSP Actions

<jsp:useBean>

<jsp:setProperty>

<jsp:getProperty>

<jsp:include>

<jsp:forward>

<jsp:plugin>

<jsp:param>

jsp:plugin

Тег `jsp:plugin` замещается тегом `OBJECT` или `EMBED`. Пользователь получает дополнительные модули в составе ответа.

Элементы `jsp:param` определяют параметры апплета или компонента `JavaBean`, элементы `jsp:fallback` - некоторое содержание, используемое браузером клиента в том случае, если дополнительный модуль по какой-либо причине не может быть вызван, если `OBJECT` / `EMBED` не поддерживаются браузером клиента или при возникновении других проблем.

Действие `jsp:fallback`, как и `jsp:params`, является "дочерним" действием тега `jsp:plugin` и вне тега не применяется.

JSP Actions

`<jsp:useBean>`

`<jsp:setProperty>`

`<jsp:getProperty>`

`<jsp:include>`

`<jsp:forward>`

`<jsp:plugin>`

`<jsp:param>`

Синтаксис действия `jsp:plugin`

```
<jsp:plugin type="bean | applet" />
  code="кодОбъекта"
  codebase="размещениеОбъекта"
  {align="выравнивание"}
  {archive="списокАрхивов"}
  {height="высота"}
  {hspace="горизонтальныйОтступ"}
  {jreversion="номерВерсии"}
  {name="наименованиеКомпонента"}
  {vspace="вертикальныйОтступ"}
  {width="ширина"}
  {nspluginurl="urlАдресДляNetscapeNavigator"}
  {iepluginurl="urlАдресДляInternetExplorer"}>
  {<jsp:params
    <jsp:param name="наименованиеПараметра"
               value="значениеПараметра" />
  </jsp:params>
  {<jsp:fallback>произвольный текст
    </jsp:fallback>}
</jsp:plugin>
```

JSP Actions

`<jsp:useBean>`

`<jsp:setProperty>`

`<jsp:getProperty>`

`<jsp:include>`

`<jsp:forward>`

`<jsp:plugin>`

`<jsp:param>`

jsp:param

Действие jsp:param позволяет задавать значения параметрам в следующих конструкциях:

- jsp:include
- jsp:forward
- jsp:plugin
- jsp:params

Синтаксис действия **jsp:param**

```
<jsp:param  
  name="наименованиеПараметра"  
  value="значениеПараметра" />
```

JSP Actions

<jsp:useBean>

<jsp:setProperty>

<jsp:getProperty>

<jsp:include>

<jsp:forward>

<jsp:plugin>

<jsp:param>

JSTL, ОБЗОР

JSTL (JavaServer Pages Standard Tag Library) – это библиотека тэгов и функций, которые могут понадобиться разработчику.

JSTL	URI (JSTL 1.1)	Prefix
Core tag library	http://java.sun.com/jsp/jstl/core	c
I18N-capable formatting	http://java.sun.com/jsp/jstl/fmt	fmt
SQL tag library	http://java.sun.com/jsp/jstl/sql	sql
XML tag library	http://java.sun.com/jsp/jstl/xml	xml
Functions tag library	http://java.sun.com/jsp/jstl/functions	fn

Библиотека тегов JSTL состоит из четырёх групп тегов: основные теги – **core**, теги форматирования – **formatting**, теги для работы с SQL – **sql**, теги для обработки XML – **xml**.

Library	Actions	Description
core	14	Основные: if/then выражения и switch конструкции; вывод; создание и удаление контекстных переменных; управление свойствами JavaBeans компонентов; перехват исключений; итерирование коллекций; создание URL и импортирование их содержимого.
formatting	12	Интернационализация и форматирование: установка локализации; локализация текста и структуры сообщений; форматирование и анализ чисел, процентов, денег и дат.
sql	6	Доступ к БД: описание источника данных; выполнение запросов, обновление данных и транзакций; обработка результатов запроса.
xml	10	XML-анализ и преобразование: преобразование XML; доступ и преобразование XML через XPath и XSLT

Подключение библиотеки **core**.

- `<%@taglib uri="http://java.sun.com/jstl/core" prefix="c" %>` – для обычной JSP.
- `<jsp:root version="1.2" xmlns:c="http://java.sun.com/jstl/core"> ...</jsp:root>` – для XML формата JSP.

Подключение библиотеки **fmt**.

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

Подключение библиотеки **sql**.

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

Подключение библиотеки **xml**.

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

Подключение библиотеки **functions**.

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

EXPRESSION LANGUAGE

В JSTL вводится понятие **Expression Language (EL)**. EL используется для упрощения доступа к данным, хранящимся в различных областях видимости (page, request, application) и вычисления простых выражений.



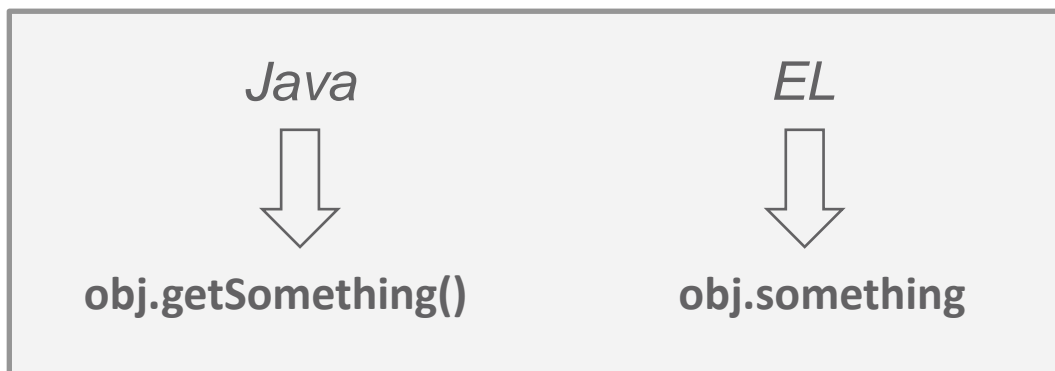
Язык выражений нужен для того, чтобы манипулировать данными при их отображении в JSP.

EL вызывается при помощи конструкции “\${имя}”

Начиная с версии спецификации **JSP 2.0 / JSTL 1.1**, **EL** является частью JSP и поддерживается без всяких сторонних библиотек.

С версии web-app 2.4 **атрибут isELIgnored** по умолчанию имеет значение **true**. В более ранних версиях необходимо указывать его в директиве **page** со значением **true**.

EL лучше всего подходит для работы с объектами, придерживающихся нотации JavaBean..



EL-идентификатор ссылается на переменную, возвращаемую вызовом `pageContext.findAttribute(имя)`.

В общем случае переменная может быть сохранена в любой области видимости:

page	PageContext
request	HttpServletRequest
session	HttpSession
application	ServletContext

В случае если переменная не найдена, возвращается null.

```
<c:set var="salary" scope="request" value="${40}" />
<c:out value="${requestScope.salary}" />
```

```
<c:set var="salary" scope="session" value="${30}" />
<c:out value="${sessionScope.salary}" />
```

Для EL доступны следующие неявные объекты:

pageContext - объект PageContext, предоставляет доступ к следующим объектам.

context -контекст сервлета страницы JSP.

session - объект-сессия клиента.

request - объект-запрос, выполняемый JSP-страницей.

pageScope - java.util.Map

requestScope - java.util.Map

sessionScope - java.util.Map

applicationScope - java.util.Map

param - java.util.Map (ServletRequest.getParameter(String name))

paramValues - java.util.Map (ServletRequest.getParameterValues(String name))

```
<c:out value="${param.name}" />
<c:out value="${paramValues.name[0]}" />
```



Для EL доступны следующие неявные объекты:

header - java.util.Map (HttpServletRequest.getHeader(String name))

headerValues - java.util.Map

cookie - java.util.Map

initParam - java.util.Map (ServletContext.getInitParameter(String name))

```
<c:out value="${header[\\"host\\"]}" />
```

Оператор [] используется как для доступа к элементам массива, так и для списков (List) и отображений (Map). Например:

```
${obj.arrayValue[0]+obj.arrayValue[1]}  
${obj.mapValue["price"+obj.mapValue["tax"]]}
```



```
obj.getArrayValue(0)+obj.getArrayValue(1)  
obj.getMapValue().get("price")+obj.getMapValue().get("tax")
```

Со списками, массивами чаще работают через `<c:forEach>`.

Типы операторов

Стандартные операторы отношения	== (или eq), != (или neq), < (или lt), > (или gt), <= (или le), >= (или ge)
Арифметические операторы	+, -, *, / (или div), % (или mod)
Логические операторы	&& (или and), (или or), ! (или not)
Оператор empty	используется для проверки переменной на null, или “пустое значение”. Термин “пустое значение” зависит от типа проверяемого объекта. Например, нулевая длина для строки или нулевой размер для коллекции.

```
<c:if test="${not empty user and user.name neq 'guest'}">  
    User is Customer.  
</c:if>
```

Операторы сравнения для EL.

Оператор EL (для JSP)	Аналог в Java	Операторы $>$, $<$, $+$, $-$, $*$, $/$ тоже допустимы.
a eq b	a.equals(b) (значения null корректно учитываются)	
a lt b	a < b либо a.compareTo(b)<0 в зависимости от типа объектов	
a gt b	a > b либо a.compareTo(b)>0 в зависимости от типа объектов	
le, ge	Нестрогие неравенства (\leq , \geq) или конструкции с compareTo в зависимости от типа.	
and, or, not	&&, , !	
empty a	Проверка a на null, на пустую строку, пустой массив, пустую коллекцию.	

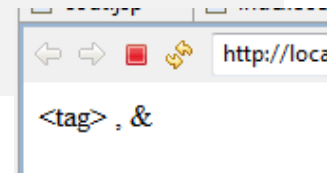
`$\${obj.something.justAnotherSomething eq 'hello'}$`

JSTL: CORE TAGS

<c:out> - вычисляет и выводит значение выражения.

Attribute	Description	Required	Default
value	Выводимая информация	Yes	None
default	значение, выводимое по умолчанию	No	body
escapeXml	True – если тэг должен опускать специальные символы xml	No	true

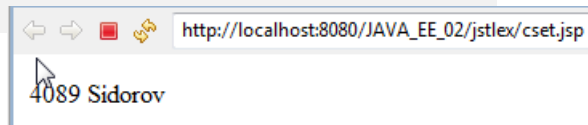
```
<%@ page language="java" contentType="text/html; charset=utf-8"
    pageEncoding="utf-8"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
...
<body>
    <c:out value="${'<tag> , &'}" default="deftest" />
</body>
</html>
```



<c:set> - устанавливает переменную в указанную область видимости.

Attribute	Description	Required	Default
value	Значение для установки	No	body
target	Имя переменной, свойство которой будет модифицировано	No	None
property	Модифицируемое свойство	No	None
var	Имя переменной, чтобы хранить информацию	No	None
scope	Область видимости	No	Page

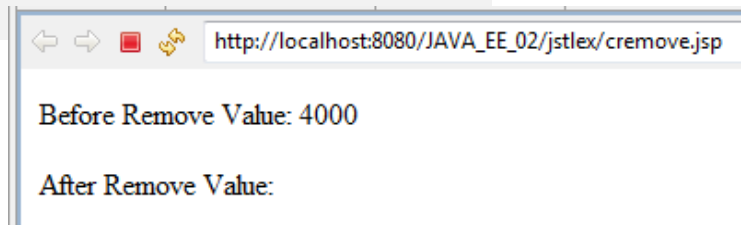
```
<c:set var="salary" scope="session" value="\${2000*2}" />
<c:set var="salary" scope="session" value="\${salary+89}" />
<c:out value="\${salary}" />
<jsp:useBean id="name"
    class="_java._ee._02._servlet.usebean.SimpleBean" />
<c:set target="\${name}" property="surname" value="Sidorov" />
<c:out value="\${name.surname}" />
```



<c:remove > - удаляет переменную из указанной области видимости.

Attribute	Description	Required	Default
var	Имя удаляемой переменной	Yes	None
scope	Область видимости	No	All scopes

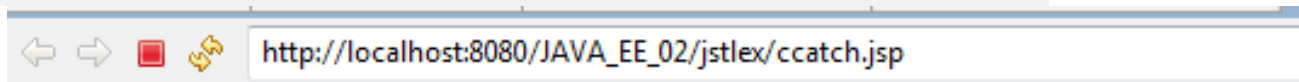
```
<c:set var="salary" scope="session" value="\${2000*2}" />
<p>
    Before Remove Value:
    <c:out value="\${salary}" />
</p>
<c:remove var="salary" />
<p>
    After Remove Value:
    <c:out value="\${salary}" />
</p>
```



<c:catch> - перехватывает обработку исключения.

Attribute	Description	Required	Default
var	Имя переменной, которая перехватит java.lang.Throwable, если оно будет выброшено каким-нибудь элементом в теле	No	None

```
<c:catch var = "catchException">
  <jsp:setProperty property="name" name="myBean"/>
</c:catch>
<c:if test = "${catchException != null}">
  <p>The exception is : ${catchException} <br />
  There is an exception: ${catchException.message}</p>
</c:if>
```

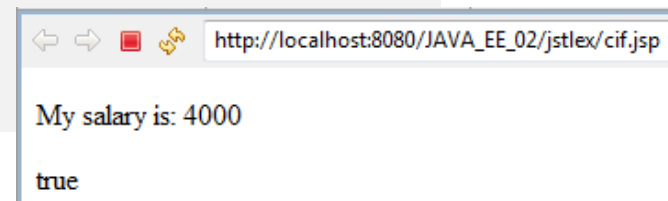


The exception is : org.apache.jasper.JasperException: java.lang.NullPointerException
There is an exception: java.lang.NullPointerException

<c:if> - тело тега вычисляется только в том случае, если значение выражения true.

Attribute	Description	Required	Default
test	Условие	Yes	None
var	Переменная для хранения результатов сравнения	No	None
scope	Область видимости для переменной, которая хранит результат сравнения	No	page

```
<c:set var="salary" scope="session" value="${2000*2}" />
<c:if test="${salary > 2000}" var="testcif">
    <p>
        My salary is:
        <c:out value="${salary}" />
    <p>
</c:if>
<c:out value="${testcif}" />
```

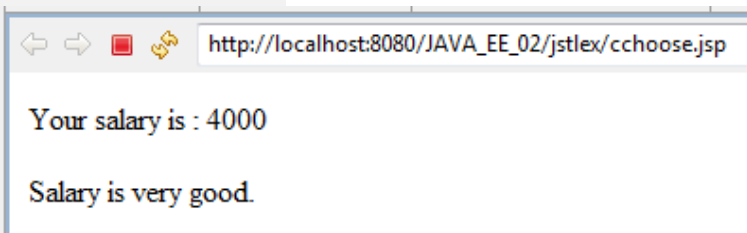


<c:choose><c:otherwise > - то же что и **<c:if />** с поддержкой нескольких условий и действия, производимого по умолчанию

<c:when>

Attribute	Description	Required	Default
test	условие	Yes	None

```
<c:set var="salary" scope="session"
value="\${2000*2}" />
<p>Your salary is : <c:out
value="\${salary}" /></p>
<c:choose>
  <c:when test="\${salary <= 0}">
    Salary is very low to survive.
  </c:when>
  <c:when test="\${salary > 1000}">
    Salary is very good.
  </c:when>
  <c:otherwise>
    No comment sir...
  </c:otherwise>
</c:choose>
```



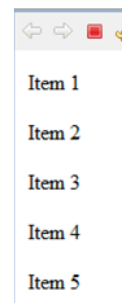
<c:import> - добавляет на JSP содержимое указанного WEB-ресурса

Attribute	Description	Required	Default
url	URL для импортирования	Yes	None
context	/ указание расположения web-приложения	No	Current application
charEncoding	Кодировка импортируемых данных	No	ISO-8859-1
var	Переменная для хранения импортированного текста	No	Print to page
scope	Область видимости переменной, что хранит импортируемый текст	No	Page
varReader	Имя альтернативной переменной для java.io.Reader	No	None

<c:forEach> - выполняет тело тега для каждого элемента коллекции

Attribute	Description	Required	Default
items	источник(коллекция) для организации цикла	No	None
begin	Начальное значение счетчика цикла	No	0
end	Конечное значение счетчика цикла	No	Last element
step	Шаг цикла	No	1
var	Переменная для хранения текущего значения счетчика цикла	No	None
varStatus	Имя переменной, хранящей статус (количество пройденных циклов) счетчика цикла	No	None

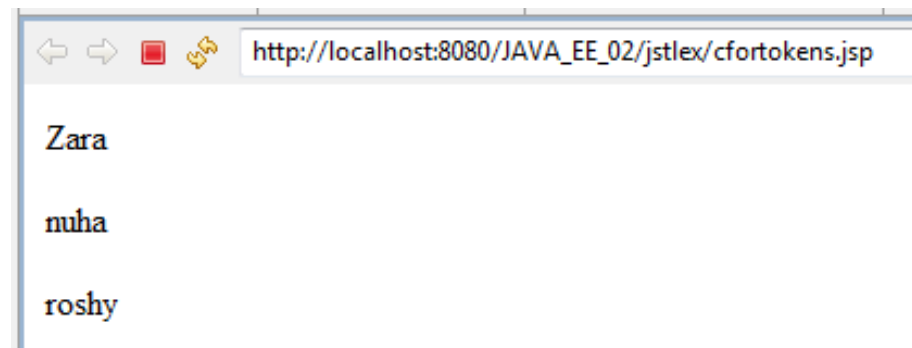
```
<c:forEach var="i" begin="1" end="5">
  Item <c:out value="$ {i} "/><p>
</c:forEach>
```



<c:forTokens> - выполняет тело тега для каждой лексемы в строке

Attribute	Description	Required	Default
delims	Символы-разделители	Yes	None

```
<c:forTokens items="Zara,nuha,roshy" delims="," var="name">
  <c:out value="{name}" /><p>
</c:forTokens>
```



<c:url> - формирует адрес с учётом контекста приложения `request.getContextPath()`

Attribute	Description	Required	Default
value	Основной URL	Yes	None
context	/ расположение web-приложения	No	Current application
var	Имя переменной, для сохранения URL	No	Print to page
scope	Область видимости сохраненного URL	No	Page

<c:param> - добавляет параметр к запросу, сформированному при помощи <c:url/>

Attribute	Description	Required	Default
name	Имя параметра запросы для установления в URL	Yes	None
value	Значение параметра запроса для установления в URL	No	Body

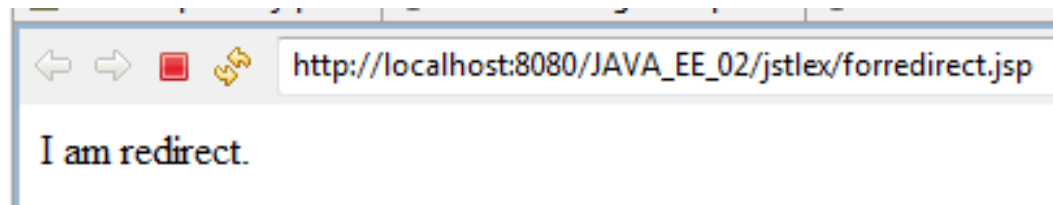
<c:redirect > - перенаправляет запрос на указанный URL

Attribute	Description	Required	Default
url	URL для перенаправления с помощью пользовательского браузера	Yes	None
context	/ расположение web-приложения	No	Current application

```
<body>
I am direct.
<c:redirect url="forredirect.jsp"/>
</body>
```

forredirect.jsp

```
<body>I am redirect.
</body>
```



JSTL: FMT TAGS

Tag	Описание
<fmt:formatNumber>	Формирует целочисленное значение с определенной точностью или форматом
<fmt:parseNumber>	Разбирает строковое представление числа, валюты или процентов
<fmt:formatDate>	Форматирует дату и/или время, используя определенные стили и шаблоны
<fmt:parseDate>	Разбирает строковое представление даты/времени
<fmt:bundle>	Загружает ресурс-bundle, который будет использовать телом тега

Tag	Описание
<fmt:setLocale>	Сохраняет указанную локаль в конфигурационной переменной
<fmt:setBundle>	Загружает ресурс-bundle и хранит его в именованной переменной в контексте или в конфигурационной переменной
<fmt:timeZone>	Определяет часовой пояс для любого времени, форматируя и разбирая код в своем теле.
<fmt:setTimeZone>	Сохраняет часовой пояс в конфигурационной переменной
<fmt:message>	Отображает интернационализованное сообщение
<fmt:requestEncoding>	Устанавливает кодировку запроса.

[-] localization

- local_en.properties
- local_ru.properties
- local.properties

locale.properties

```
local.message = Hello  
local.locbutton.name.en = EN  
local.locbutton.name.ru = RU
```

locale_en.properties

```
local.message = Hello  
local.locbutton.name.en = EN  
local.locbutton.name.ru = RU
```

locale_ru.properties

```
local.message = \u041F\u0440\u0438\u0432\u0435\u0442\u0430.  
local.locbutton.name.en = \u0430\u043D\u0433\u043B\u0438\u043A\u043E  
local.locbutton.name.ru = \u0440\u0443\u0441\u0441\u043A\u0438\u0439
```



```
public class Controller extends HttpServlet {  
    private static final long serialVersionUID = 1L;  
  
    protected void doPost(HttpServletRequest request,  
        HttpServletResponse response)  
        throws ServletException, IOException {  
        request.getSession(true).setAttribute("local",  
            request.getParameter("local"));  
        request.getRequestDispatcher("index.jsp").  
            forward(request, response);  
    }  
}
```

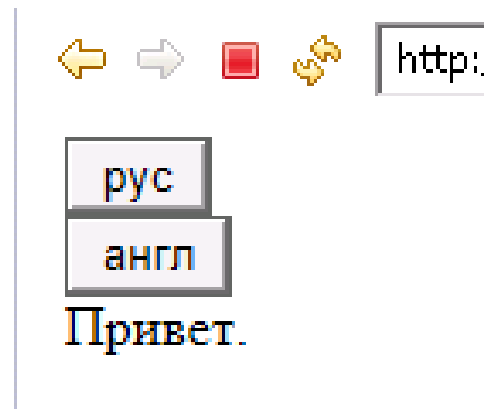


index.jsp

```
...
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"
prefix="c"%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/fmt"
prefix="fmt"%>
...
<fmt:setLocale value="${sessionScope.local}" />
<fmt:setBundle basename="localization.local" var="loc" />
<fmt:message bundle="${loc}" key="local.message"
var="message" />
<fmt:message bundle="${loc}" key="local.locbutton.name.ru"
var="ru_button" />
<fmt:message bundle="${loc}" key="local.locbutton.name.en"
var="en_button" />
...
```

```
...  
<body>  
  <form action="Controller" method="post">  
    <input type="hidden" name="local" value="ru" />  
    <input type="submit" value="${ru_button}" /><br />  
  </form>  
  
  <form action="Controller" method="post">  
    <input type="hidden" name="local" value="en" />  
    <input type="submit" value="${en_button}" /><br />  
  </form>  
  
  <c:out value="${message}" />  
</body>  
...
```

Результат:



JSTL: SQL, XML TAGS, FUNCTIONS

SQL tags:

- `<sql:setDataSource>`
- `<sql:query>`
- `<sql:update>`
- `<sql:param>`
- `<sql:dateParam>`

XML tags

- `<x:out>`
- `<x:parse>`
- `<x:set>`
- `<x:if >`
- `<x:forEach>`
- `<x:choose>`
- `<x:when>`
- `<x:otherwise>`
- `<x:transform>`
- `<x:param>`

JSTL Functions:

- `fn:contains()`
- `fn:containsIgnoreCase()`
- `fn:endsWith()`
- `fn:escapeXml()`
- `fn:indexOf()`
- `fn:join()`
- `fn:length()`
- `fn:replace()`
- `fn:split()`
- `fn:startsWith()`
- `fn:substring()`
- `fn:substringAfter()`
- `fn:substringBefore()`
- `fn:toLowerCase()`
- `fn:toUpperCase()`
- `fn:trim()`



СПАСИБО ЗА ВНИМАНИЕ!

ВОПРОСЫ?

Java for students

Филиал кафедры ПОИТ БГУИР в Eram Systems
курс: Веб-технологии (JAVA)

Author: Olga Smolyakova , PhD
Oracle Certified Java 6 Programmer
[Olga Smolyakova@epam.com](mailto:Olga_Smolyakova@epam.com)