15.04.2020
Custom taglib

Намиот Д.Е.
dnamiot@gmail.com

JSP taglib

- <%@ taglib uri=“ссылка на описание" prefix= "как выделять в тексте" %>

- Расширение для JSP тегов

- <jsp:include page=“next.jsp”/>

- jsp: стандартный префикс
  include – тег
  page - атрибут

- в директиве taglib указаны:
  prefix
  ссылка на описание (XML файл), где перечислены теги и атрибуты

JSP taglib

• Это теги. Выполняются уже в процессе работы сервлета. Соответственно, JSP компилятор должен сформировать какой-то код для вставки в тело сервлета

• Задачи:
Определить, что это пользовательский тег
Проверить правильность написания
Сформировать Java-код, который реализует его семантику

• тег задается в описании. jsp – стандартный тег, пользовательский префикс компилятор берет из директивы taglib

• директивы - статические элементы, известны при компиляции

• <myprefix:my_tag /> - определяется компилятором

JSP taglib

• Что проверяет (может проверять компилятор):

• теги
<jsp:include vs <jsp:include

• тело тега
<jsp:include page=“a.jsp”/>
<jsp:include page=“a.jsp> тело </jsp:include> наличие
<jsp:include page=“a.jsp></jsp:include> пустое
<jsp:include page=“a.jsp> может ли быть Java-код в теле </jsp:include>
наличие скриплетов

JSP taglib

• атрибуты
<mypref:myTag a1”=“v1” a2=“v2”/>

Перечень атрибутов
Обязательный/необязательный
Можно ли использовать выражения:

<mypref:myTag a1”=“${expr}” a2=“<%=expr%>”/>

• вся информация о теге сосредотачивается в XML файле
TLD – tag library descriptor

• Библиотека: описание тегов и Java-код.
все может быть в одном ‚jar файле

• Есть стандартная библиотека. Ее описание известно компилятору

TLD

```
<taglib>
<tlib-version>1.0</tlib-version>
<short-name>bar-baz</short-name>
<tag-file>
  <name>d</name>
  <path>/WEB-INF/tags/bar/baz/d.tag</path> </tag-file>
</taglib>
```

TLD

```
<tag>
  <name>present</name>
  <tag-class>com.mycompanyTag</tag-class>
  <body-content>scriptless</body-content>
   …
   <attribute>
      <name>test</name>
      <required>true</required>
      <rtexprvalue>true</rtexprvalue>
   </attribute>
     ...
</tag>
```

TLD

Deployment:

.jar file - как обычно WEB-INF/lib

.tld file – технически где угодно, путь к нему указывается в директиве taglib
Чаще всего: /WEB-INF   или /WEB-INF/tags

Расширение - любое

TLD

Deployment:

.jar file - как обычно WEB-INF/lib

.tld file – технически где угодно, путь к нему указывается в директиве taglib
Чаще всего: /WEB-INF   или /WEB-INF/tags

Расширение - любое

TLD

description (optional) описание (комментарий).

display-name (optional) имя для IDE

icon (optional) иконка для IDE.

name имя тега.

tag-class полное имя класса, который реализует логику.

tei-class (optional) имя класса типа javax.servlet.jsp.tagext.TagExtraInfo. Для описания переменных.

TLD

body-content - тип тела тега

variable (optional) описание переменной, которая будет создана в теге

attribute Declares an attribute of the custom tag. See Declaring Tag Attributes for Tag Handlers.

dynamic-attributes true или false Поддержка динамических имен. По умолчанию - ложь. Если истина, то тег должен реализовывать специальный интерфейс javax.servlet.jsp.tagext.DynamicAttributes.

example (optional) пример использования. Для IDE.

tag-extension (optional) дополнительная информация для IDE.

body-content

body-content

- Если не указано – нет ограничений. Возможные ограничения:
- **tagdependen**t: тело рассматривается как текст и интерпретируется самим тегом.
- **empty**: нет тела (пусто).
- **scriptless**: текст, EL и другие custom tags.

attribute

description (optional) описание.

name уникальное (внутри тега) имя атрибута.

required (optional) true | false Обязательный или нет. По умолчанию false.

rtexprvalue (optional) true | false Можно ли использовать выражение в качестве значения

type (optional) тип атрибута. По умолчанию java.lang.String.

fragment (optional) true | false По умолчанию false. Если true то значение не вычисляется контейнером. Это будет делать тег. В этом случае не указывается rtexprvalue и тип. Тип будет javax.servlet.jsp.tagext.JspFragment.

attribute

```
<tag>
    <name>present</name>
    <tag-class>com.msu.MyTag</tag-class>
    <body-content>scriptless</body-content>
    ...
    <attribute>
        <name>test</name>
        <required>true</required>
        <rtexprvalue>true</rtexprvalue>
    </attribute>
    ...
</tag>
```

variable

```
<tlt:iterator var="departmentName" type="java.lang.String"
        group="${myorg.departmentNames}">

    <tr>
        <td><a
  href="list.jsp?deptName=${departmentName}">
            ${departmentName}</a></td>
    </tr>

</tlt:iterator>
```

variable

Что задается для переменной:

Имя

Тип

Новый или существующий объект

Область видимости

Variable: видимость

NESTED – внутри тега

AT_BEGIN – от начала до конца объемлющего тега (или до конца страницы)

AT_END – от начала до конца объемлющего тега (или до конца страницы)

Variable: описание

description (optional) описание переменной.

name-given - имя

variable-class – тип переменной

declare (optional) true | false декларирован или нет данный объект. По умолчанию true

scope – область видимости AT_BEGIN, AT_END, или NESTED По умолчанию NESTED.

Variable: описание

```
<tag>
    <variable>
        <name-given>var</name-given>
        <variable-class>java.lang.String</variable-
    class>
        <declare>true</declare>
        <scope>NESTED</scope>
    </variable>
</tag>
```

Custom tags

<pref:MyTag a1="v1" />

||

V

```
MyTag mt = new com.mycompany.MyTag();
mt.setA1("v1");
mt.doTag();
```

Custom tags

<pref:MyTag
a1="v1" a2="v2"> тело тега </pref:MyTag>

||

V

```
MyTag mt = new com.mycompany.MyTag();
mt.setA1("v1");
mt.setA2("v2");
mt.setJspBody(new JspFragment(…));
mt.doTag();
```

Custom tags

```
<pref:MyTag  a1="v1"/>
                ||
                V
public MyTag extends SimpleTagSupport {
    private String a1;
    public void setA1(String a1) { this.a1=a1; }
    public String getA1() { return this.a1; }

    public void doTag() throws JspException, IOException {
        // бизнес - процесс
        getJspContext().getOut().write("Hello, world.");
    }
}
```

Custom tags: body

Для работы с телом тега

getJspBody() – получить тело тега

invoke() - выполнить

Параметр:

JspWriter объект или null

В последнем случае используется JspWriter из JSPContext (пишет на страницу)

Custom tags: body

<pref:IfSimple cond="true"> это тело тега </pref:ifSimple>

```
public class IfSimpleTag extends SimpleTagSupport {
    private boolean cond = false;
    public void setCond(boolean cond) {
        this.cond = cond;
    }
    public void doTag() throws JspException, IOException {
        if(cond){
            getJspBody().invoke(null);
        }
    }
}
```

Custom tags: body

<pref:toUpper cond="true"> это тело тега </pref:toUpper>

```
public class toUpperWriter extends SimpleTagSupport {
    public void doTag() throws JspException, IOException {
        StringWriter sw = new StringWriter();
        getJspBody.invoke(sw);
        getJspContext().
            getOut().println(sw.toString().toUpperCase());
    }
}
```

Custom tags: body

```
private String var;
private Collection group;
private Iterator iterator;

public void setVar(String var) {
    this.var = var;
}
public void setGroup(Collection group) {
    this.group = group;
    if(group.size() > 0)
        iterator = group.iterator();
}
```

Custom tags: body

```
public void doTag() throws JspException, IOException {
    if (iterator == null) return;
    while (iterator.hasNext()) {
        // устанавливаем значение объекта
        getJspContext().setAttribute(var, iterator.next());
        // выполняем тело тега
        getJspBody().invoke(null);
    }
}
```

Тело тега выполняется несколько раз

setAttribute(var, iterator.next()); - PAGE_SCOPE

Но можно ведь и использовать SESSION_SCOPE, APPLICATION_SCOPE

Поддержка переменных

```
public class IteratorTEI extends TagExtraInfo {
    public VariableInfo[] getVariableInfo(TagData data) {
        String type = data.getAttributeString("type");
        if (type == null) type = "java.lang.Object";
        return new VariableInfo[] {
            new VariableInfo(data.getAttributeString("var"),
            type,
            true,
            VariableInfo.NESTED)
        };
    }
}
```

Поддержка переменных

<tei-class>

iterator.IteratorTEI

</tei-class>

Занесение объекта в соответствующий scope – уже в коде

JSTL

Area	Subfunction	Prefix
Core	Variable support	`c`
	Flow control	
	URL management	
	Miscellaneous	
XML	Core	`x`
	Flow control	
	Transformation	
I18N	Locale	`fmt`
	Message formatting	
	Number and date formatting	
Database	SQL	`sql`
Functions	Collection length	`fn`
	String manipulation	

JSTL: Core

Area	Function	Tags	Prefix
Core	Variable support	`remove` `set`	`c`

	Flow control	`choose`
		`when`
		`otherwise`
		`forEach`
		`forTokens`
		`if`	

	URL management	`import`
		`param`
		`redirect`
		`param`
		`url`
		`param`	

	Miscellaneous	`catch` `out`	

JSTL: Core

```
<c:set var="foo" scope="session" value="..."/>

<c:set var="foo"> ... </c:set>

<c:remove var="cart" scope="session"/>

<c:if test="${!empty param.Add}">
   ...
</c:if>

<c:forEach var="item" items="${sessionScope.cart.items}">
    ...
    <tr>
        <td align="right" bgcolor="#ffffff">
        ${item.quantity}
        </td>
     </tr>
    ...
</c:forEach>
```

JSTL: Core

```
<c:import var="data" url="http://www.company.com"/>
<c:out value="${data}"/>
```

Аналог jsp:include но ссылка может быть внешняя

```
<c:out value="value" [escapeXml="{true|false}"]
     [default="defaultValue"] />
```

```
<c:catch var ="catchTheException">
   <% int x = 1/0;%>
</c:catch>
```

```
<c:if test = "${catchTheException != null}">
   <p>The type of exception is : ${catchTheException} <br />
   There is an exception: ${catchTheException.message}</p>
</c:if>
```

JSTL: i18n

Area	Function	Tags	Prefix
I18N	Setting Locale	`setLocale` `requestEncoding`	`fmt`

	Messaging	`bundle`
		`message`
		`param`
		`setBundle`	

	Number and Date Formatting	`formatNumber`
		`formatDate`
		`parseDate`
		`parseNumber`
		`setTimeZone`
		`timeZone`	

JSTL: i18n

text_en.properties:

login.label.username = Username
login.label.password = Password
login.button.submit = Sign in

text_ru.properties:

login.label.username = Имя пользователя
login.label.password = Пароль
login.button.submit = Войти

Все это в пакете. E.g.

com.my.i18n

JSTL: i18n

```
<fmt:setLocale value="${language}" />
<fmt:setBundle basename="com.my.i18n.text" />

<form method="post">
   <fmt:message key="login.label.username" />:
   <input type="text" id="username" name="username">
   <br>
   <fmt:message key="login.label.password" />:
   <input type="password" id="password" name="password">
   <br>
   <fmt:message key="login.button.submit" var="buttonValue" />
   <input type="submit" name="submit" value="${buttonValue}">
 </form>
```

JSTL: SQL

Area	Function	Tags	Prefix
Database	Setting the data source	`setDataSource`	`sql`

	SQL	`query`
		`  dateParam`
		`  param`
		`transaction`
		`update`
		`  dateParam`
		`  param`	

JSTL: SQL

```
<sql:setDataSource dataSource="jdbc/MyDB" />

<sql:query var=“goods" > select * from Goods where id = 101</sql:query>

<table>
<c:forEach var=“goodsRow" begin="0"
          items="${goods.rowsByIndex}">
    <tr>
      <td> ${goodsRow[0]} </td>
      <td> ${goodsRow[1]} </td>
    </tr>
</c:forEach>
</table>
```

sql:query возвращает RowSet

JSTL: SQL

```
<sql:update dataSource="${myDS}" var="newCitizen">
        UPDATE users SET address=?, telephone=? where inn=?
        <sql:param value="${param.address}" />
        <sql:param value="${param.tel}" />
        <sql:param value="${param.inn}" />
</sql:update>
```

JSTL: functions

Area	Function	Tags	Prefix
Functions	Collection length	`length`	`fn`

	String manipulation	`toUpperCase`, `toLowerCase`
		`substring`, `substringAfter`, `substringBefore`
		`trim`
		`replace`
		`indexOf`, `startsWith`, `endsWith`, `contains`, `containsIgnoreCase`
		`split`, `join`
		`escapeXml`	

JSTL: functions

```
<%@ 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>
<form method=“post”>
<input type="text" name="username" size="25">
<input type="submit" value="Submit">
<input type="reset" value="Reset">
</form>

<c:if test="${fn:length(param.username) > 0}" >
  <%@include file="response.jsp" %>
</c:if>
```