学习编写标记文件(tag):

自定义标记的用途就是给程序员可以自定一些特定功能的标记，以及封装代码，达到分工，重用性等
多种好处.

JSP 规范 2.0 中新增了标记文件(tag file)的使用，免除以前自定义标记必须为一个 Java 类，以及必须
加上一个标记描述文件(tld)的编码难度.

以下我们来看标签文件的使用.

1. 存放:

一个标记文件以 tag 为後缀名，以同一类型的检签放到同一个文件夹中组成一个标记库，再放到
"/WEB-INF/tags/"中保存.

例如接下来我打算建立一组有关数学的标记库，我在"/WEB-INF/tags/"下建立一个"maths"文件夹，然
後就可以在"/WEB-INF/tags/maths/"下建立後缀名为 tag 的标记文件了.

现在我想先做一个数字相加的标记，起名为"add.tag"，放到 maths 标记库下.

2. 语法:

标记文件其实就是一个 jsp 文件，所以语法基本上跟 jsp 是一样的，在第一行加上以下这个元素来告诉
服务器它是一个标记文件:

<%@ tag %>

我们还要加上一些属性来告诉服务器这个标记文件的设定:

1) body-content - 设定这个标记的主体内容类型:

A. empty
这个是一个空标记.
B. scriptless
主体可以有内容，而 jsp 容器会去处理里面的 jsp 元素，换句话就是可以是
文本, EL 表达式, 标准动作甚至另一个自定义标记.
C. tagdependent
主体可以有内容，而 jsp 容器会把它们当作纯文件处理 .

2) pageEncoding - 设定这个标记的编码

我们的 add 标记是个空标记，而编码是"UTF-8"，所以加上这样的一句:

<%@ tag body-content="empty" pageEncoding="UTF-8" %>

标记中使用以下这个元素来声明属性:

<%@ attribute %>

attribute 元素也有几个属性:

1) name - 这个 attribute 的名称.

2) required - true/false，是否必须的.

3) rtexprvalue - true/false，这个 attribute 可否使用 EL 表达式，否则为纯文本.

4) type - 设定这个 attribute 的类型, jsp 容器会把结果自动转换成这个类.

我们的 add.tag 有两个属性，分别代表两个要相加的数字, x 跟 y，它们都是必须的，可以用 EL 表达式:

<%@ attribute name="x" required="true" rtexprvalue="true" %>
<%@ attribute name="y" required="true" rtexprvalue="true" %>

那这个标记都设定好了，那我们应该怎样读取这些传入的属性呢？其实很简单，标记文件就是一个
jsp 文件，这些输入的属性值都存放在作用域中，所以利用 EL 表达式就可以了:

${x + y}

这个标记文件就完成了，以下为完整代码，你也可以直接打开"/WEB-INF/tags/maths/add.tag":

<%--
maths 标记库 add 标记

功能: 把传入参数相加
参数:
x，数字 1，必须
y，数字 2，必须
--%>

```
<%@ tag body-content="empty" pageEncoding="UTF-8" %>

<%-- 声明属性 --%>
<%@ attribute name="x" required="true" rtexprvalue="true" %>
<%@ attribute name="y" required="true" rtexprvalue="true" %>

<%-- 内容 --%>
${x + y}
```

标记文件完成了，那我们要怎样在 jsp 文件中调用它呢？首先我们导入 maths 标记库，跟调用标准动作
一样使用 taglib 指令，但是把 url 属性换成 tagdir，指定我们自定义标记库的位置，并给它一个前缀
名，以下我们用"maths":

```
<%@ taglib tagdir="/WEB-INF/tags/maths/" prefix="maths" %>
```

然後我们就可以调用标准动作一样去使用它啦，当然必须的属性一定要有，不然会报错:

```
<maths:add x="10" y="10"/>
```

以下为调用 add 标记的 jsp 页的完整代码，同样的你可以直接打开"/add.jsp":

```
<%-- maths 标记库 add 标记的演试 --%>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%-- 调用自定义标记库 maths --%>
<%@ taglib tagdir="/WEB-INF/tags/maths" prefix="maths" %>

<html>
<head>
<title>Maths 标记库 add 标记演试</title>
</head>
<body>
结果:<br/>
<%-- 使用 add 标记 --%>
<maths:add x="10" y="10"/>
</body>
</html>
```

相信大家都了解基本工作原理了吧？你可以把标记文件看作一个 jsp 页面，你可以对它传入一些参数，
调用它的面页就像包含它一样，所以你可以标记文件打上一句"HelloWorld"，一段 html 代码，甚至包
含其他的动作，调用它的面页就会显示，大家可以打开"/helloworld.jsp"看一下，它调用了别

一个
标记库 others 里面的 helloworld 标记.

add 标记是一个空标记，我们学过 body-content="scriptless"，那我们应该怎样去处理标记主体的数
据呢？我们来新增一个新的标记库，叫 string，再增加一个标记文件"show.tag"，同样的加上 tab 指
令告诉服务器这个是一个标记文件，这一次 body-content 属性改为 scriptless:

```
<%@ tag body-content="scriptless" pageEncoding="UTF-8" %>
```

然後我们加入一个标准动作:

```
<jsp:doBody>
```

它的用途是读入标记的主体内容，当没有指定要保存到的变量时，它把内容直接输出到调用的页面上.

我们写一个"/show.jsp"来简单调用这个 show 标记:

```
<%-- string 标记库 show 标记的演试 --%>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%-- 调用自定义标记库 string --%>
<%@ taglib tagdir="/WEB-INF/tags/string" prefix="string" %>

<html>
<head>
<title>string 标记库 trim 标记演试</title>
</head>
<body>
<%-- 使用 show 标记 --%>
<string:show>
这个是标记的主体内容
</string:show>
</body>
</html>
```

<jsp:doBody>有三个属性:

1) var - 当指定这个变量，读取的主体内容就会以 String 保存这个变量内，不会直接输出页面.

2) varReader - 用途跟 var 一样，但是保存的不是一个 String，而是一个 java.io.Reader.

3) scope - 变量保存到的作用域，包括: page, request, session, application，默认为 page.

接下来我们扩充 string 标记库，加入一个新的标记"trim.tag", 同样的加入 tag 指令，设定 body-content 为 scriptless 代表它可以有主体内容，因为我们会使用到核心标记库，所以加上 taglib
指令导入核心标记库，并加入 jsp:doBody 指令读取主体内容并保存到变量 body 中:

```
<%@ tag body-content="scriptless" pageEncoding="UTF-8" %>

<%-- 导入核心标记库 --%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%-- 把主体内容保存到 body 变量中，没有指定作用域下，保存在 page --%>
<jsp:doBody var="body"/>
```

我们获得一个主体内容的字符串了，那可以编写一段代码来把字符串中的空白去掉，我们可以利用核
心标记库的一个标记 forTokens 来把字符串跟据空白来分割，再输出就可以达到这个效果了，以下给出
这个标记的完整代码，大家也可以直接打开"/WEB-INF/tags/string/trim.tag":

```
<%--
string 标记库 trim 标记

功能: 把标记主体的内文中的空白去掉
--%>

<%@ tag body-content="scriptless" pageEncoding="UTF-8" %>

<%-- 导入核心标记库 --%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>

<%-- 把主体内容保存到 body 变量中，没有指定作用域下，保存在 page --%>
<jsp:doBody var="body"/>

<%-- 实现把主体字符串的空白去掉 --%>
<c:forTokens var="char" items="${body}" delims=" ">${char}</c:forTokens>
```

这里有一个要注意的，最後一行必须要这样写，什么意思？很多人(包括我)可能会这样写:

```
<c:forTokens var="char" items="${body}" delims=" ">
${char}
</c:forTokens>
```

这个是好习惯，代码看上去整齐多了，但是 jsp 容器会把换行换成一个空白…哈哈，不懂的自己试一
下吧^^

这个演试可以看到标记文件不单没有以前标记处理类加上一个 tld 来得那么麻烦，而且在一个标记文
件中，我们可以再导入另一个标记库来扩充功能，实在是非常方便.

相信大家都对标记文件的基本应用都有一定的了解了，下面我们来看标记文件的其他功能.

如果我们想建立一些标记文件供网页人员使用，而且这个标记文件支持所有 html 元素的属性，那你可
能立刻就想到，我们必须为每一个可能出现的 html 标记属性建立一个属性标记，那是多费时的工作，
所以我们可以在 tab 指令中加上 dynamic-attributes 这个属性，表示这个标记可以容许任何的属性值
出现，jsp 容器会自动把所有未声明的属性保存到一个以你给出的值为命名的 Map 集合中，而这个集合
保存在 page 作用域中. 我们来新建一个标记文件到 others 标记库中，并加上以下代码行:

```
<%@ tag body-content="empty" pageEncoding="UTF-8"
dynamic-attributes="attributesList" %>
```

当调用这个标记时，所有未被声明的属性都会保存到"attributesList"这个 Map 集合中，我们可以用
核心标记库的 forEach 把它们列出，以下为完整代码，你也可以直接打开"/WEB-INF/tags/others/
showAttributes.tag":

```
 <%--
others 标记库 showAttributes 标记

功能: 读取 dynamic-attributes 所保存的属性集合, 输出到调用的页面上.
--%>
<%@ tag body-content="empty" pageEncoding="UTF-8"
dynamic-attributes="attributesList" %>
<%-- 导入核心标记库 --%>
<%@ taglib uri="http://java.sun.com/jsp/jstl/core"  prefix="c" %>

<%-- 输出所有属性 --%>
<p>本标记包含了以下的动态属性</p>
<%-- 使用 forEach 遍历"attributesList"集合并输出到调用页面 --%>
<c:forEach var="item" items="${attributesList}" varStatus="i">
```

```
<p>${i.index + 1}. ${item.key}: ${item.value}</p>
</c:forEach>
```

另外编写一个 jsp 页面调用 showAttributes 标记，以下为完整代码，你也可以直接打开 "/showAttributes.jsp":

```
 <%-- others 标记库 showAttributes 标记的演试 --%>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%-- 调用自定义标记库 others --%>
<%@ taglib tagdir="/WEB-INF/tags/others/" prefix="others" %>

<html>
<head>
<title>others 标记库 showAttributes 标记演试</title>
</head>
<body>
<others:showAttributes param1="hello" param2="on9" color="red"/>
</body>
</html>
```

接下来我们看一个属性的扩展功能 - 属性片段(attribute fragment)，先解译什么是片段，大家可以
把片段看作一段 jsp 代码, JSP 规范中所谓的 jsp fragment，例如我们之前使用了 jsp:doBody 读取的主
体内容就是一个片段. 我们可以将一段已命名的片段(named fragment)作为一个标记的属性使用，并
而在标记中多次或一次调用这个传入的片段. 要调用这个片段，我们使用另一个标淮动作:

```
<jsp:invoke/>
```

它是一个空标记，有以下的属性:

1) fragment - 要调用的片段名.

2) var - 给出变量名，把片段经过 jsp 容器计算过之後的结果作为字符串保存.

3) varReader - 同上，不过将结果作为一个 java.io.Reader 保存.

4) scope - 作用域= =，有 var 就有它，不多说了，默认 page.

当然，叫做属性片段，我们必须告诉 jsp 容器标记中那些标记属性的内容会是片段而不是静态文本. 我

们可以在属性元素上加上 fragment="true"这个属性:

```
<%@ attribute name="name" fragment="true" %>
```

接下来我们再编写一个标记来演试一下属性片段的应用，我们在 others 标记库中新增一个 "invoke"
标记，加上 tag 指令，建立两个属性 fragment1 跟 fragment2，用来接收两个属性片段，以下给出完整
代码，大家也可以直接打开"/WEB-INF/tags/others/invoke.tag":

```
<%--
others 标记库 invoke 标记

功能: 演试属性片段
--%>
<%@ tag body-content="scriptless" pageEncoding="UTF-8" %>
<%@ attribute name="fragment1" fragment="true" %>
<%@ attribute name="fragment2" fragment="true" %>

<%-- 调用标准动作 jsp:invoke 显示读取的属性片段 --%>
<p style="font-size:24px; font-weight:bold">这里输出属性片段一的计算结果:</p>
<jsp:invoke fragment="fragment1"/>
<p style="font-size:24px; font-weight:bold">这里输出属性片段二的计算结果:</p>
<jsp:invoke fragment="fragment2"/>
<%-- 调用标准动作 jsp:doBody 显示读取的主体内容 --%>
<p style="font-size:24px; font-weight:bold">这里输出主体内容:</p>
<jsp:doBody/>
```

上面可以看到我们调用了两次<jsp:invoke fragment="xxx"/>，大家可以把它看作一个占位符，它的
位置就是读取的属性片段的计算结果. 最後我们再一次调用<jsp:doBody/>来读取主体内容.

不懂? 没关系，下面我们先把调用页面"invoke.jsp"建立好，细心观察浏览器出来的效果，大家就可
以了解为什么我会说把<jsp:invoke/>看作一个占位符，其实<jsp:doBody/>也是同一个工作原理，下
面给出"invoke.jsp"的完整代码，同样大家可以直接打开"/invoke.jsp":

```
<%-- others 标记库 invoke 标记的演试 --%>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%-- 调用自定义标记库 others --%>
<%@ taglib tagdir="/WEB-INF/tags/others" prefix="others" %>
<%-- 调用自定义标记库 string --%>
```

```
<%@ taglib tagdir="/WEB-INF/tags/string" prefix="string" %>

<html>
<head>
<title>others 标记库 invoke 标记演试</title>
</head>
<body>
<%-- 使用 invoke 标记 --%>
<others:invoke>
<jsp:attribute name="fragment1"><%-- 第一个片段我们引用 helloworld 标记 --%>
<others:helloworld/>
</jsp:attribute>
<jsp:attribute name="fragment2"><%-- 第二个片段我们引用 trim 标记 --%>
<string:trim>
I am a boy!
</string:trim>
</jsp:attribute>
<jsp:body> <%-- 这里就是主体内容 --%>
这里就是<font color="red" size="25">主体内容</font>啦
</jsp:body>
</others:invoke>
</body>
</html>
```

(上面代码的排版有点乱，注释本来是自己开新行的，但是属性之间如果插入注释，服务器会报错= =)

这个 jsp 页面包含我们今天所建立的另外两个标记"helloworld"跟"trim"，它们分别就是两个属性片段
的内容，另外大家会看到<jsp:body>...</jsp:body>这个没介绍过的标准动作，因为一个标记的主体
如果已经包含了<jsp:attribute/>元素的话，主体的内容就必须放到<jsp:body/>之内，jsp 容器才可
正常的分别到哪些是属性，哪个是主体，所以以下的语法是不容许的:

```
<others:invoke>
<jsp:attribute name="fragment1">
属性一的内容
</jsp:attribute>
<jsp:attribute name="fragment2">
属性二的内容
</jsp:attribute>
<p>这里是主体内容...</p>
</others:invoke>
```

下来留一个小测试给大家，打开"/invoke.jsp"，在<jsp:body/>标记内加上这一段代码:

${"a" == "a"}

重载"/invoke.jsp"，大家会发现在网页的末尾，出现了"true"一字，然後我们来开"/WEB/INF/tags/others/invoke.tag"，把 tag 指令中的 body-content 属性改为"tagdependent"，重载"/invoke.jsp"看有什么不一样?

接下来这个会是个难点，这个部份希望大家可以返覆多看几遍，试著修改一下例题来了解一下参数不
同会产生什么不一样的结果.

我们一直的工作流程都是，一个 jsp 页面调用一个自定义的标记，给它一些属性，我们在标记里读取它
们，然後处理代码，返回给 jsp 容器，jsp 容器计算结果後给 jsp 页面输出静态的文本，那我们可不可以
在处理代码後，把结果以变量的形式返回呢？就好像我们写一个有返回值的函数一样？可能大家会想
想我们有<c:set/>这个声明/赋值的标准动作，但是想一下，标记文件跟 jsp 页面是两个不同的 page 作
用域，我们如果要传递变量的话就只有把变量放在 session/application 中了，那不是很麻烦?万一我
在jsp接收变量後没有把这个变量的内存空间释放掉，那服务器有10GB内存也不够用了，所以 JSP 规范
中提供了一个指令:

<%@ variable %>

它可以让我们在标记文件中的声明一个输出变量，让 jsp 页面读取，它有以下的属性:

1) name-given - 给这个变量取名.

2) name-from-attribute - 这个变量名由一个属性来决定(跟 name-given 只可选一).

3) alias - 给输出变量取别名(配合 name-from-attribute 使用).

4) variable-class - 给出输出变量类型，jsp 容器会自动转换，如果没指定为 String.

5) scope - 输出变量在 jsp 页面上的更新模式，可取值为: AT_BEGIN/AT_END/NESTED，默认为"NESTED".

相信第 1,4 属性大家都看懂，第 2,3 属性我们在下一个例子中再说明，我们来说明一下 scope 属性.

scope 的 3 个可取值代表什么意思呢?

1) AT_BEGIN - 输出变量在标记文件的<jsp:doBody/>/<jsp:invoke/>之前及标记文件执行结束後更新输出变量的值.

2) AT_END - 输出变量只有在标记文件执行结束後更新输出变量的值.

3) NESTED - 输出变量在标记文件的<jsp:doBody/>/<jsp:invoke/>之前更新输出变量的值, 在标记文件结束後, 将会把输出变量还原到在调用标记前的值

不懂? 我也搞了 4 个小时, 返覆的试验才清楚这三个取值有什么不一样. 现在不了解没有关系, 它们关
系到在标记文件中使用到<jsp:doBody/>/<jsp:invoke/>时, 其他的代码在这些动作之前之後的不同,
我们现在只需要记得一个不同就好了. 如果你需要把输出变量输出到 jsp 页面上(= =不然呢?), 就用
"AT_BEGIN", 如果你只需要在标记执行中在 jsp 页面读取到输出变量, 那就用"NESTED".

可能大家还是会搞混了, 我不就是为了输出一个变量到 jsp 页面吗? 你给我一个"NESTED"干吗? 其实输
出变量还有一个用途, 就是让标记文件可以把变量传到另一个标记文件上. 如果我的标记文体主体内
容是另一个标记, 我想把值传给它而已, 并没有需要传到 jsp 页面的作用域上时, "NESTED"就有用了,
因为还原的意思是, 如果调用的 jsp 页面本来就没有声明这个输出变量, 那就代表输出变量只有在标记
主体执行时出现过, 结束後它就卸载了.

下面我们就来看一个例子吧(还是不懂? 就只用"AT_BEGIN"吧, 管它还不还原), 以下建立了一个新的
标记文件"variable.tag"在 others 标记库中, 同样的加入 tag 指令, 导入核心标记库(?), 以下给出完
整代码, 大家也可以直接打开"/WEB-INF/tags/others/variable.tag":

```
<%--
others 标记库 variable 标记

功能: 演试 variable 指令输出变量的功能
--%>
<%@ tag body-content="empty" pageEncoding="UTF-8" %>
<%-- 导入核心标记库 --%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
```

```
<%-- 声明一个输出变量，取名为"v" --%>
<%@ variable name-given="v" scope="AT_BEGIN" %>
<%-- 对输出变量赋值 --%>
<c:set var="v" value="这个就是标记文件的输出的内容"/>
```

为什么要使用要<c:set/>动作呢？因为 variable 指令只是声明了一个变量，我们还需要给它赋值，下
面同样我们再建立一个"/variable.jsp"，看看 jsp 页面是如何读取到这个变量，以下给出完整代码，
大家也可以直接打开"/variable.jsp":

```
<%-- others 标记库 variable 标记的演试 --%>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%-- 导入 others 标记库 --%>
<%@ taglib tagdir="/WEB-INF/tags/others" prefix="others" %>

<html>
<head>
<title>others 标记库 variable 标记演试</title>
</head>
<body>
<%-- 调用 variable 标记并显示 --%>
<others:variable/>
输出变量 v: ${v}
</body>
</html>
```

有了 variable 指令，我们就可以把一个标记文件处理後的结果，放到输出变量中，再给它指定一个类
型，作为另一个标记文件的属性传递，是不是很方便呢？

但是问题来了，这个输出变量的取名是标记文件定义的，如果调用它的页面也有这个变量那怎办？所
以我们下面来了解一下 variable 指令上"name-from-attribute"跟"alias"这两个属性，怎样可以达成
"取别名"的效果.

我们来新建一个标记文件"alias"在"others"标记库中，同样加入 tag 指令，导入核心标记库，然後我
声明一个属性，它的作用就是用来存放输出变量的取名，让调用页面可以传入一个字符值来决定输出
变量的取名，然後我们使用 variable 指令:

```
<%@ variable name-from-attribute="output" alias="v" scope="AT_END" %>
```

"name-from-attribute"属性告诉 jsp 容器这个输出变量的取名为那个属性的值，而"alias"属性就为
这个输出变量取一个别名，以便在标记里使用这个变量，以下给出完整代码，大家也可以直接打开
"/WEB-INF/tags/others/alias.tag":

```
<%--
others 标记库 alias 标记

功能：演试 variable 指令的取别名功能
--%>
<%@ tag body-content="empty" pageEncoding="UTF-8" %>
<%-- 导入核心标记库 --%>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>

<%-- 存放输出变量的名称，记得这个属性必须为静态文本及必须的 --%>
<%@ attribute name="output" required="true" rtexprvalue="false" %>
<%-- 声明输出变量，在标记文件中使用别名为"v"，而在调用页面上的名称则
由属性 output 决定 --%>
<%@ variable name-from-attribute="output" alias="v" scope="AT_END" %>

<%-- 为输出变量赋值 --%>
<c:set var="v" value="<p>这个就是标记文件的输出的内容</p>"/>
```

同样的，我们建立一个"/alias.jsp"来演试一下，以下给出完整代码，大家也可以直接打开
"/variable.jsp":

```
<%-- others 标记库 alias 标记的演试 --%>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
<%-- 导入 others 标记库 --%>
<%@ taglib tagdir="/WEB-INF/tags/others" prefix="others" %>

<html>
<head>
<title>others 标记库 alias 标记的演试</title>
</head>
<body>
<%-- 调用 alias 标记，对 output 属性赋值就可以自定义输出变量名 --%>
<others:alias output="super"/>
${super}
<%-- 试一下用别的名称 --%>
<others:alias output="haha"/>
${haha}
```

```
</body>
</html>
```

使用了"name-from-attribute"就可以避免命名冲突了，记得存放输出变量取名的属性必须把"rtexprvalue"设为假，不然服务器也会报错的.

以上就是 JSP 规范 2.0 中新增的标记文件的语法介绍了^^

3. 打包

今天计一下我们都创建了 8 个自定义动作，如果我们想要跟人家分享我的作品，或者要去第三方部署的
话，那就麻烦了，所以以下我们学习怎样去把我们的标记库打包.

第一步，我们先改变一下文件结构，在网站根目录下，创建一个"META-INF"文件夹，然後把"WEB-INF"
里面的"tags"文件夹复制过去，形成以下文件结构:

```
网站根目录/
META-INF/
tags/
maths/
...tag 文件...
others/
...tag 文件...
string/
...tag 文件...
WEB-INF/
...tag 文件夹...
...jsp 文件...
```

我们先打包 string 标记库，所以把"maths"跟"others"文件夹删除，然後我们需要创建一个 tld 标签描
述文件(对，又是它，XML 又来了)在"META-INF"下，复制下面的内容到文件开始:

```
<?xml version="1.0" encoding="UTF-8" ?>
<taglib xmlns="http://java.sun.com/xml/ns/j2ee"
xmlns:xsi="http://www.w3.org/2001/XMLScheme-instance"
xsi:schemeLocation="http://java.sun.com/xml/ns/j2ee
http://java.sun.com/sml/ns/j2ee/web-jsptaglibrary_2_0.xsd"
version="2.0">

<!-- 标记库版本 -->
<tlib-version>1.0</tlib-version>
```

```
<!-- 命名空间前缀 -->
<short-name>string</short-name>
<!-- 文件夹名 -->
<uri>string</uri>

<!-- 包含的标记 -->

<tag-file>
<!-- 标记名 -->
<name>show</name>
<!-- 路径 -->
<path>/META-INF/tags/string/show.tag</path>
</tag-file>
<tag-file>
<name>trim</name>
<path>/META-INF/tags/string/trim.tag</path>
</tag-file>
</taglib>
```

taglib 元素的一大串属性不用管它，这个不是我们的讨论范围，以後要写 tld 文件就复制过去就对了，
版本跟命名空间前缀也可以不管，那个是给设计工具使用的，没什么影响，uri 元素必须为标记库的
文件夹名，之後的 tag-file 元素声明了这个标签库里面有什么标记，语法为:

```
...
<tag-file>
<name>标记名</name>
<path>相对路径</path>
</tag-file>
...
```

完成後打开"命令提示符"，转到网站的根目录，输入下面的命令:

```
jar cvf string.jar META-INF
```

完成後大家就会看到根目录下新增了一个 jar 文件"string.jar"，把它放到"/WEB-INF/lib"之下就 OK
了．那我们再新建一个 jsp 页面来调用这个已经打包好的标记库吧，以下为"jar.jsp"的完整代码，
同样大家可以直接打开"/jar.jsp":

```
<%-- 使用 jar 文件导入标记库的演试 --%>
<%@ page contentType="text/html; charset=UTF-8" language="java" %>
```

```
<%-- 导入 string 标记库包 --%>
<%@ taglib uri="string" prefix="string" %>

<html>
<head>
<title>使用 jar 文件导入标记库的演试</title>
</head>
<body>
<%-- 调用 trim 标记 --%>
<string:trim>
Please trim me!
</string:trim>

</body>
</html>
```

看到我们这一次 taglib 指令不再使用 tagdir 属性了, 直接用 uri 属性.

以上就是打包的全部讲解^^

P.S. 写在最後

因为开始要编写毕业的项目, 我发现班上有很多同学都没有能真正的掌握自定义标记的用法, 所以才
试著编写一篇讲解标记文件的文章, 顺便也可以放到 BLOG 上, 没想到一写就写了六百多行@"@. 但是
过程是非常愉快的, 写到一些知识点上发生自己也没搞清楚, 就跑去翻翻资料, 看看文献, 对自己本
身的能力也加强了好多, 毕竟我也是一个初学者.

希望这篇文章可以帮忙大家在标记文件的学习, 另外有很多的专业名词的翻译可能会跟大家认识有出
入, 希望大家见谅^^, 如果有其他问题可以在我的 BLOG 上留言, 我会尽可能的帮忙的, 谢谢.