


pdw2009的专栏

≡ 目录视图


≡ 摘要视图

RSS 订阅

个人资料



pdw2009



访问：85827次
积分：2114
等级：ELITE 5
排名：第9554名

原创：114篇 转载：70篇
译文：0篇 评论：7条

文章搜索

文章分类

blackberry (3)

c&pro*c (7)

CSS/JS (2)

delphi (11)

j2ee (14)

linux unix 应用 (58)

oracle (5)

php (1)

tuxedo (4)

c&pro*c (6)

webservice (6)

TB编程 (4)

C++ (1)

AutoIT (1)

hbase-hadoop (8)

mysql c (2)

C# (2)

mysql (2)

glib (3)

android (6)

thymeleaf (6)

spring (5)

netty (0)

Ionic (16)

AngularJS (9)

zk-dubbo (6)

【博乐】把C币，装进你的口袋 专家亲授Hadoop技术，你报名了吗？ 软考套餐限时7折优惠 CSDN移动客户端，下载送50C币

Thymeleaf模板的使用

分类：thymeleaf 2015-03-28 14:08 1278人阅读 评论(0) 收藏 举报

Thymeleaf模板

使用模板的要点：
页面主体结构固定，具体参数可变，尽可能让参数动态化，才能提高模板的复用性

=====

Thymeleaf's core is a DOM processing engine

Processor: An Object which applies some logic to a DOM node

Standard Dialect: a set of processor,provided by Thymeleaf core library

Template Engine : can be configured several dialects at a time,called process chain

=====

Template Resolver

Template Resolvers are objects that implement an interface from the Thymeleaf API called org.thymeleaf.templateresolver.ITemplateResolver

public TemplateResolution resolveTemplate(final TemplateProcessingParameters
templateProcessingParameters);

All implementaion class :
-ClassLoaderTemplateResolver
-FileTemplateResolver
-ServletContextTemplateResolver
-UrlTemplateResolver

Initial Template Engine use ServletContextTemplateResolver

private static void initializeTemplateEngine() {

 ServletContextTemplateResolver templateResolver = new ServletContextTemplateResolver();

 // XHTML is the default mode, but we will set it anyway for better understanding of code
 templateResolver.setTemplateMode("XHTML");

 // This will convert "home" to "/WEB-INF/templates/home.html"
 // 设置模板的前置路径
 templateResolver.setPrefix("/WEB-INF/templates/");
 //设置模板统一的后缀名
 templateResolver.setSuffix(".html");

文章存档

2015年08月 (25)
 2015年07月 (15)
 2015年06月 (2)
 2015年05月 (2)
 2015年04月 (4)
 2015年03月 (8)
 2015年02月 (5)
 2015年01月 (6)
 2014年12月 (5)
 2014年09月 (2)
 2014年07月 (5)
 2014年06月 (4)
 2014年05月 (3)
 2014年04月 (2)
 2014年03月 (16)
 2014年02月 (2)
 2014年01月 (2)
 2013年11月 (7)
 2013年10月 (2)
 2013年07月 (4)
 2013年06月 (1)
 2013年04月 (6)
 2013年03月 (1)
 2013年02月 (2)
 2013年01月 (2)
 2012年12月 (5)
 2012年10月 (1)
 2012年08月 (2)
 2012年07月 (5)
 2012年06月 (5)
 2012年05月 (1)
 2012年02月 (1)
 2011年08月 (1)
 2011年06月 (2)
 2011年04月 (1)
 2011年03月 (1)
 2010年12月 (1)
 2010年08月 (9)
 2010年07月 (6)
 2010年06月 (1)
 2010年05月 (2)
 2010年04月 (7)

阅读排行

判断ftp上文件是否存在 (3573)
 HBase安装配置之伪分布 (3283)
 在linux编译proc出错解决 (3056)
 aix shell 获取昨天的日期 (2394)
 java httpclient 无证书访问 (2351)
 delphi 实现两个exe文件 (2306)
 aix shell 判断文件中是否 (2130)
 hbase 启动顺序 (2128)
 shell 读取oracle数据库表 (1842)
 Hadoop 常用命令 (1758)

评论排行

delphi 实现两个exe文件 (3)

```
// Set template cache TTL to 1 hour. If not set, entries would live in cache until expelled by LRU
templateResolver.setCacheTTL(Long.valueOf(3600000L));
```

```
// Cache is set to true by default. Set to false if you want templates to
// be automatically updated when modified.
templateResolver.setCacheable(true);
```

```
templateEngine = new TemplateEngine();
templateEngine.setTemplateResolver(templateResolver);
```

```
}
```

- 1.new one TemplateResolver instance
- 2.config the resolver
- 3.new Template engine
- 4.set resolver to this engine
- 5.invoke engine's process method to work

```
=====
```

```
<!DOCTYPE html SYSTEM "http://www.thymeleaf.org/dtd/xhtml1-strict-thymeleaf-3.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org">
```

或者使用下面的文档声明也可以，这样就没有Thymeleaf对文档的校验功能了，因为没有引入对应的DTD，而且IDE可能会提示错误，但是不影响对模板的解析。

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

```
=====
```

```
#{}

```

specify that a text should correspond to a specific message

引用外部文件(message)中的内容，进行替换

首先，需要指定外部文件的位置，如果没有指定，则使用默认的Standard Message Resolver会到/WEB-INF/templates/下寻找properties文件

home_en.properties文件是如何被定为到的呢？

通过WebContext ctx = new WebContext(request, response, servletContext, request.getLocale());

其中，request.getLocale()就说明了当前系统的所在地，就能确定读取en/zh_CN了。

当然，可以配置外部文件的位置！

```
th:text="#{home.welcome}"
```

The th:text attribute, which evaluates its value expression and sets the result of this evaluation as the body of the tag it is in

th:text 计算表达式的结果，并使用这个结果来替换当前标签中的内容

```
=====
```

```
public void process(
```

```
    final HttpServletRequest request, final HttpServletResponse response,
    final ServletContext servletContext, final TemplateEngine templateEngine)
    throws Exception {
```

```
    WebContext ctx = new WebContext(request, response, servletContext, request.getLocale());
    ctx.setVariable("today", Calendar.getInstance());
```

```
    templateEngine.process("home", ctx, response.getWriter());
```

记录拆花上shell(2)

thymeleaf中文乱码问题(1)

aix shell 获取昨天的日期(1)

JAVA客户端调用memca(0)

centos安装 nginx(0)

nginx 相关好文章收藏(0)

glib单向列表(0)

GLib双向链表(0)

glib入门(0)

推荐文章

最新评论

thymeleaf中文乱码问题
yeyu710: 按楼主的办法解决了，谢谢

delphi 实现两个exe文件共享内存地狱圣者: A程序里面有个grid,可以新增保存数据，B程序跨进程把外部数据存到A程序中去。这个怎么写啊

delphi 实现两个exe文件共享内存地狱圣者: A程序里面有个grid,可以新增保存数据，B程序跨进程把外部数据存到A程序中去。这个怎么写啊

aix shell 获取昨天的日期
zhuzhuxiazst: 赞..

delphi 实现两个exe文件共享内存renying123: 楼主，我用你这代码运行了下，怎么获取不到共享的内容呀？？求解~~

记录拆花上shell
pdw2009: 字符串的抬头去尾代码:\$x=aabbaarealwvww\$echo "\${x%w*w}"aabb...

记录拆花上shell
pdw2009: 字符串拆花截取：read DATEmonth=\$(echo \$DATE | awk '{ print...

}

Thymeleaf中提供了2个实现IContext的实现类

org.thymeleaf.context.Context implements IContext

org.thymeleaf.context.WebContext implements IWebContext

WebContext 提供了更多的方法可用

=====

Unescaped Text

照原样对文本进行输出，不对> < 进行转义

th:utext="Big Character" 将输出为: Big Character

=====

Using and displaying variables

1.设置变量

SimpleDateFormat dateFormat = new SimpleDateFormat("dd MMMM yyyy");

Calendar cal = Calendar.getInstance();

WebContext ctx = new WebContext(request, servletContext, request.getLocale());

ctx.setVariable("today", dateFormat.format(cal.getTime()));

templateEngine.process("home", ctx, response.getWriter());

2.在模板中使用变量

th:text="{today}"

3.使用Thymeleaf提供的内置变量，不用提前设置，直接在模板中使用

th:text="{#calendars.format(today,'dd MMMM yyyy')}"

=====

Thymeleaf Standard Dialect: the Thymeleaf Standard Expression syntax

Thymeleaf 支持的运算符和表达式的应用

所有的操作符都可以嵌套使用，非常强大！

1.Text literals: '...'

2.Number literals: 0,1.0,12.3,etc

Simple expression: 表达式语法

1.Message expression : #{}

2.Variable expression : \${}

3.Link URL expression: @{}

4.Selection Variable expression: *{}

结合th:object使用，在某个范围内进行变量的查找，而不是在context中查找，缩小了查询的范围，效率如何呢？

如何没有与th:object结合使用，*{}与\${}效果一样，因为其范围自动扩展到context。

Binary operations: 运算符

1.String concatenation: +

2.Arithetic operations : +, -, *, /, %

3.Comparators: >, <, >=, <=

4.Boolean operators: and, or

5.Equality operators: ==, !=

Unary operations:

1.Minus sign(): - 负号

2.Boolean negation: !, not 否定符

Conditional operators: 条件表达式

- 1.If-then-else: (if)?(then):else 三元运算符
- 2.If-then: (if) ? (then) ,省略了else部分, 如果条件不成立, 返回null
- 3.Default: (value)?:(defaultValue) , use second value only first is null

All this operations can be combined and nested:

'User is of type ' + (\${user.isAdmin()}) ? 'Administrator' : (\${user.type}) ? 'Unknown'

=====

Message 在模板中获取消息

动态指定message的内容/甚至可以动态指定访问哪个message

th:utext="#{home.welcome(\${session.user.name})}", 其中\${session.user.name}作为参数, 替换home.welcome的映射文本中的{0}

th:utext="#{\${welcomeMsgKey}(\${session.user.name})}", 其中\${welcomeMsgKey}动态指定message文件中的key

=====

Variables 在模板中获取数据

\${...} expressions are in fact OGNL (Object-Graph Navigation Language) expressions executed

on the map of variables contained in the context.

模板中获取变量值的方式, 使用\${},针对不同类型数据, 用法如下:

```
/*
 * Access to properties using the point (.). Equivalent to calling property getters.
 * 通过.进行导航, 相当于调用getters()
 */
```

```
${person.father.name}
```

```
/*
 * Access to properties can also be made by using brackets ([]) and writing
 * the name of the property as a variable or between single quotes.
 * 使用[]等效于使用., 但是[]在某些场合能完成.不能完成的任务
 */
```

```
${person['father']['name']}
```

```
/*
 * If the object is a map, both dot and bracket syntax will be equivalent to
 * executing a call on its get(...) method.
 * 访问Map集合
 */
```

```
${countriesByCode.ES}
${personsByName['Stephen Zucchini'].age}
```

```
/*
 * Indexed access to arrays or collections is also performed with brackets,
 * writing the index without quotes.
 * 访问数组
 */
```

```
${personsArray[0].name}
```

```
/*
 * Methods can be called, even with arguments.
 * 调用对象的方法
 */
```

```
${person.createCompleteName()}
```

```
${person.createCompleteNameWithSeparator('-')}
```

=====
Expression utility objects 在模板中使用内置对象

内置对象，提供了很多方便的功能，日期格式化，字符串处理，数字格式化等

#dates, formatting, component extraction, etc

#calendars

#numbers, formatting numeric objects.

#strings, contains, startsWith, prepending/appending, etc

#booleans

#arrays

#lists

#sets

#maps

#aggregates, creating aggregates on arrays or collections

#messages, equal to using #{}

#ids, deal with id attributes, eg: as a result of an iteration

#ctx等等，还有很多！

=====
Link URLs 在模板中使用URL链接

@{ }

Several types of URLs:

1. Absolute URL, like http://localhost:8080/thymeleaf

2. Relative URL, which can be:

Page-relative, like: user/login.html 页面相对定位

Context-relative, like: /itemdetails?id=1 (context name in server will be added automatically) 项目根路径定位, 模板解析时会自动加上应用程序的名称作为前缀

Server-relative, like: ~/billing/processInvoice (allow calling URLs in another context in the same server) 相同服务器根目录下的定位

如果要使用相对定位，必须指定一个实现了IWebcontext接口的对象，因为需要从其中获取httprequest对象，从而得到应用程序的根路径，才能处理相对路径

相对路径，并且在URL上使用OGNL表达式取参数，而且thymeleaf会自动对URL进行编码：

```
<a href="details.html" th:href="@{/order/details(orderId=${o.id})}">view</a>
```

```
<a th:href="@{/details/' + ${user.login}(orderId=${o.id})}">view</a>
```

=====
Literals 模板中使用简单文本

字符串/数字

a. 原样输出

```
<p>The year is <span th:text="2011">1492</span>.</p>
```

b. 字符串拼接

```
th:text="'The name of the user is ' + ${user.name}"
```

=====
Arithmetic operations 模板中对变量进行算数运算

+ - * / %

有两种计算方式，都可以。

1. 使用Thymeleaf进行运算--->先OGNL表达式取到变量值，然后thymeleaf再进行运算

```
th:with="isEven=(${prodStat.count} % 2 == 0)"
```

2. 使用OGNL进行运算--->直接在OGNL表达式中进行运算

```
th:with="isEven=${prodStat.count % 2 == 0}"
```

Comparators and Equality 模板中使用比较符和等号

模板中不能直接使用 > < >= <=

需要进行转义:

```
> gt;
< lt;
>= ge; gte;
<= le; lte;
== eq;
!= ne; neq;
```

```
th:text="Execution mode is ' + ( ${execMode} == 'dev')? 'Development' : 'Production' )"
```

Conditional expressions 三元运算，第一个表达式的结果为boolean类型

```
if ? then : else ---> A ? B : C
```

```
<tr th:class="${row.even}? 'even' : 'odd'"> 设置tr的class属性，用来控制行的显示效果很方便
```

嵌套使用条件表达式:

```
<tr th:class="${row.even}? (${row.first}? 'first' : 'even') : 'odd'"> 灵活控制首行，偶数行，奇数行的class属性值，便于进行css样式定义
```

还可以省略else部分，当表达式结果为false，返回null，否则返回'alt'

```
<tr th:class="${row.even}? 'alt'">
...
</tr>
```

Default Expression 具有默认值的表达式，第一个表达式的结果只要不为null，就取第一个表达式的结果

being the second one evaluated only in the case of the first one returning null .

```
A ? : B ---> A不为null，则取A，否则取B
```

如果第一个表达式的计算结果为null，则取第二个表达式的结果

```
<div th:object="${session.user}">
...
<p>Age: <span th:text="*{age}?: '(no age specified)'">27</span>.</p>
</div>
```

等效于:

```
<p>Age: <span th:text="*{age != null}? *{age} : '(no age specified)'">27</span>.</p>
```

条件表达式嵌套:

```
<p>Name: <span th:text="*{firstName} ? : (*{admin} ? 'Admin' : #
{default.username})">Sebastian</span>.</p>
```

静态方法的调用

```
<p th:text="${@myapp.translator.Translator@translateToFrench('textVar')}">Some text here...
</p>
```

=====

Setting the value of any attribute 在模板中对目标设置任何属性, action, class, value, etc

非常强大, 不仅处理HTML模板方便, 处理FO-XSL(PDF模板)一样通用。

设置action属性

```
<form action="subscribe.html" th:attr="action=@{/subscribe}">
```

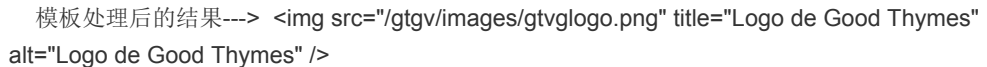
设置value属性

```
<input type="submit" value="Subscribe me!" th:attr="value=#{subscribe.submit}"/>
```

一次设置多个属性

```

```

模板处理后的结果---> 

设置属性更优雅的方式(仅在HTML模板中有效, 处理PDF模板, 只能用th:attr=""来实现, 因为PDF模板中的属性在Thymeleaf中好像没有定义)

```
<input type="submit" value="Subscribe me!" th:value=#{subscribe.submit}"/>
```

```
<form action="subscribe.html" th:action=@{/subscribe}>
```

```
<li><a href="product/list.html" th:href=@{/product/list}>Product List</a></li>
```

Thymeleaf支持HTML中几乎所有的属性定义, 使用时具体参考Thymeleaf的手册

th:bgcolor, th:border, th:cellpadding, th:cellspacing, th:colspan, th:align, th:src, th:width, th:size 等等

控制样式:

第一种方式直接在tr上定义css样式, 如bgcolor,border等

第二种方式, 在tr上定义class属性, 通过外部css样式进行控制。(复杂样式, 采用第二种方案)

=====

Appending and prepending 在已有属性上追加属性

th:attrappend 追加

th:attrprepend 放到前面

```
<input type="button" value="Do it!" class="btn" th:attrappend="class=${ ' ' + cssStyle}" />
```

```
---> <input type="button" value="Do it!" class="btn warning" />
```

遍历prods集合, 每次生成一行

```
<tr th:each="prod : ${prods}" class="row" th:classappend="${prodStat.odd}? 'odd'">
```

```
<td>${prod.name}</td>
```

```
</tr>
```

=====

Fixed-value boolean attributes 设置某些具有固定值的属性

XHTML/HTML5中, 有一些特殊的属性, 要么没有值, 要么为某个固定的值

checked

selected

disabled

multiple

readonly

如果计算结果为true, 则使用它的固定值, 否则不处理。

```
<input type="checkbox" name="active" th:checked="${user.active}" />
```

还有:

th:autofocus

th:default

th:hidden

...

=====

Iteration 循环遍历

th:each="{obj : \${objList}}" 循环所在标签的片段，每次片段中用到的都是当前遍历到的对象

后台准备数据

public void process(
 HttpServletRequest request, HttpServletResponse response,
 ServletContext servletContext, TemplateEngine templateEngine) {
 ProductService productService = new ProductService();
 List<Product> allProducts = productService.findAll();
 WebContext ctx = new WebContext(request, servletContext, request.getLocale());
 ctx.setVariable("prods", allProducts);
 templateEngine.process("product/list", ctx, response.getWriter());
}

模板中进行遍历，每次遍历生成一个tr，td列取到的对象为当前遍历到的对象

<tr th:each="prod : \${prods}">
 <td th:text="\${prod.name}">Onions</td>
 <td th:text="\${prod.price}">2.41</td>
 <td th:text="\${prod.inStock}? #{true} : #{false}">yes</td>
</tr>

Keeping iteration status 跟踪迭代过程中的状态变化

遍历过程中，提供了如下属性：

index starting
count starting
size total amount of items in the iterated variables
current current
even/odd boolean 偶数个/奇数个
first/last boolean 第一个/最后一个

在th:each中,在iter variable变量后面，定义一个status variable,通过该变量来获取遍历过程中的状态

<tr th:each="prod,iterStat : \${prods}" th:class="\${iterStat.odd}? 'odd'">

prefix 前缀

suffix 后缀

If you don't explicitly set an iteration variable, Thymeleaf will always create one for you by suffixing 'Stat' to the name of the iter variable.

如果没有定义status variable,thymeleaf会自动为我们提供一个来使用，通过在iter variable后面添加后缀Stat来使用。

<tr th:each="prod : \${prods}" th:class="\${prodStat.odd}? 'odd'">

=====

Conditional evaluation 条件语句

Simple conditionals: "if" and "unless"

在某种条件成立时，对某个fragment进行处理：

当th:if条件成立时，该标签中其它th:*标签才会发挥作用，如果条件不成立，则th:*不会执行

```
<a href="comments.html"  
th:href="@{/product/comments(prodId=${prod.id})}"
```



```
th:if="{not #lists.isEmpty(prod.comments)}">view</a>
```

th:if 判断表达式结果是否为真的规则:

If value is not null:

value is a boolean and is true;

value is a number and is non-zero;

value is a character and is non-zero;

value is a String and is not "false", "off" or "no";

value is not a boolean, a number, a character or a String;

If value is null, th:if will evaluate to false;

另外, 还可以用th:unless, 进行条件控制。如果条件为真, 则不执行。

```
<a href="comments.html"
```

```
th:href="@{/comments(prodId=${prod.id})}"
```

```
th:unless="{#lists.isEmpty(prod.comments)}">view</a>
```

Switch statement Switch选择语句

Note that as soon as one th:case attribute is evaluated as true, every other th:case attribute in the same switch context is evaluated as false.

The default option is specified as th:case="*" :

```
<div th:switch="{user.role}">
  <p th:case="admin">User is an administrator</p>
  <p th:case="{roles.manager}">User is a manager</p>
  <p th:case="*">User is some other thing</p>
</div>
```

Template Layout 模板布局/模板重用

Including template fragments

th:fragment

th:include

用法:

在某个标签中使用th:fragment属性定义一个变量名, 然后在其他模板中通过th:include, 即可引入。

a.html

```
<div th:fragment="hello"> 定义一个fragment, 并取名为"hello"
  include me.!
</div>
```

b.html

```
<div th:include="a :: hello">what?</div> 从名称为a.html的模板中引入名称为"hello"的fragment
```

***还可以不使用th:fragment属性进行引入, 通过DOMSelector

"templatename::[domselector]" like XPath expressions.

***将模板整个引入

"templatename"

th:include中同样可以嵌套使用表达式

```
<div th:include="footer :: (${user.isAdmin}? #{footer.admin} : #{footer.normaluser})"></div>
```

th:include

引入fragment中定义的内容

th:substituteby

引入fragment所在标签和内容，并替换当前的标签

如：

footer.html

```
<footer th:fragment="copy">
    &copy; 2011 The Good Thymes Virtual Grocery
</footer>
```

main.html

-----> th:include只引入内容

```
<div th:include="footer :: copy"></div>
result:
<div>
    &copy; 2011 The Good Thymes Virtual Grocery
</div>
```

-----> 整个引入，并替换掉host tag

```
<div th:substituteby="footer :: copy"></div>
result:
<footer>
    &copy; 2011 The Good Thymes Virtual Grocery
</footer>
```

Removing template fragments

th:remove 为了静态显示时提供充分的数据，但是在模板被解析后，又不需要这些模拟的数据，需要将其删除

可选属性：

th:remove="all", 删除所有

th:remove="all-but-first", 删除所有，但保留第一个

th:remove="body", 删除内容，但保留标签

th:remove="tag", 删除标签，但保留内容

Local variables 模板中自定义变量

```
th:with="firstPerson=${persons[0]}"
```

使用自定义的变量：

```
<span th:text="${firstPer.name}">
```

```
<div th:with="firstPer=${persons[0]},secondPer=${persons[1]}">
```

```
<p>The name of the first person is <span th:text="${firstPer.name}">Julius Caesar</span>.
```

```
</p>
```

```
<p>But the name of the second person is <span th:text="${secondPer.name}">Marcus
```

```
Antonius</span>.</p>
```

```
</div>
```

原理：定义的变量将被添加到context的map中，这样就能像其它变量一样被使用到了。

自定义变量的有效范围：仅在被定义的标签内有效，比如，上面的firstPerson,仅在当前div中有效

另一个用途：

定义变量存放message中的配置

然后在表达式中用

如, 从message中获取date.format的配置, 然后复制给变量df, 然后在其它地方(自定义变量所在标签内)进行使用

```
<p th:with="df=#{date.format}">
  Today is: <span th:text="${#calendars.format(today,df)}">13 february 2011</span>
</p>
```

也可以直接:

```
<p>
  Today is: <span th:with="df=#{date.format}" th:text="${#calendars.format(today,df)}">13
february 2011</span>
</p>
```

优先级的问题:

th:with 优先级高于 th:text

th:each 优先级高于 th:*

Attribute precedence 属性的优先级问题

Thymeleaf attributes have a numeric precedence:

- 1 fragment inclusion th:include
- 2 fragment iteration th:each
- 3 condition evaluation th:if/th:unless/th:switch/th:case
- 4 Local variable definition th:object, th:with
- 5 General attribute modification th:attr, th:attrprepend, th:attrappend
- 6 Specific attribute modification th:value, th:href, th:src, etc
- 7 Text th:text, th:utext
- 8 Fragment specification th:fragment
- 9 Fragment removal th:remove

```
<ul th:each="item : ${items}">
  <li th:text="${item.description}">Item description here...</li>
</ul>
```

由于th:each的优先级高于其它th:*, 所以可以简写为:

```
<ul>
  <li th:each="item : ${items}" th:text="${item.description}">Item description here...</li>
</ul>
```

还可以这样写, 将循环放到后面, 只是阅读性不好, 但不影响结果:

```
<ul>
  <li th:text="${item.description}" th:each="item : ${items}">Item description here...</li>
</ul>
```

Inlining

Text inlining

好处: 简化书写

弊端: 以prototype呈现(静态地打开网页),将原样显示

用法, 在标签上定义th:inline="text"

该标签中任意地方都可以使用内联样式获取数据

```
<p>Hello, <span th:text="${session.user.name}">Sebastian</span>!</p>
```

简化为:

```
<p>Hello, [[${session.user.name}]]!</p>
```

parent tag中定义

```
<body th:inline="text">
...
<p>hello: [[${session.user.name}]]</p>
...
</body>
```

JavaScript inlining

```
<script th:inline="javascript">
/*<![CDATA[*/
...
var username = [[${session.user}]];
...
/*>*/
</script>
```

thymeleaf将自动对user对象进行转换，而且转换为javascript中的user对象

```
<script th:inline="javascript">
/*<![CDATA[*/
...
var user = {age:null,'firstName':'John','lastName':'Apricot','name':'John
Apricot','nationality':'Antarctica'};
...
/*>*/
</script>
```

Validation and Doctypes

Validating templates

使用Thymeleaf的DTD进行校验和声明Thymeleaf的命名空间

```
<!DOCTYPE html SYSTEM "http://www.thymeleaf.org/dtd/xhtml1-strict-thymeleaf-3.dtd">
```

```
<html xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org">
```

Doctype translation Thymeleaf对模板处理完成后，会自动将DOCTYPE转换为正确的类型，这样浏览器端就能正常解析了！

```
<!DOCTYPE html SYSTEM "http://www.thymeleaf.org/dtd/xhtml1-strict-thymeleaf-3.dtd">
```

After Thymeleaf process the template, will automatically transform the DOCTYPE to:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
```

Template Resolver

```
org.thymeleaf.templateresolver.ClassLoaderTemplateResolver
return Thread.currentThread().getContextClassLoader().getResourceAsStream(templateName);
```

```
org.thymeleaf.templateresolver.FileTemplateResolver
    return new FileInputStream(new File(templateName));
```

```
org.thymeleaf.templateresolver.UrlTemplateResolver
    return (new URL(templateName)).openStream();
```

---->

Prefix and suffix:

```
templateResolver.setPrefix("/WEB-INF/templates/");
templateResolver.setSuffix(".html");
```

Encoding to be applied when reading templates:

```
templateResolver.setEncoding("UTF-8");
```

Default template mode, and patterns for defining other modes for specific templates:

```
// Default is TemplateMode.XHTML
templateResolver.setTemplateMode("HTML5");
templateResolver.getXhtmlTemplateModePatternSpec().addPattern("*.xhtml");
```

Default mode for template cache, and patterns for defining whether specific templates are cacheable or not:

```
// Default is true
templateResolver.setCacheable(false);
templateResolver.getCacheablePatternSpec().addPattern("/users/*");
```

TTL in milliseconds for parsed template cache entries originated in this template resolver. If not set, the only way to remove an entry from the cache will be LRU (cache max size exceeded and the entry is the oldest).

```
// Default is no TTL (only LRU would remove entries)
templateResolver.setCacheTTLMs(60000L);
```

Also, a Template Engine can be set several template resolvers, in which case an order can be established between them for

template resolution so that, if the first one is not able to resolve the template, the second one is asked, and so on:

When several template resolvers are applied, it is recommended to specify patterns for each template resolver so that

Thymeleaf can quickly discard those template resolvers that are not meant to resolve the template, enhancing performance.

Doing this is not a requirement, but an optimization:

```
ClassLoaderTemplateResolver classLoaderTemplateResolver = new
ClassLoaderTemplateResolver();
classLoaderTemplateResolver.setOrder(Integer.valueOf(1));
// This classloader will not be even asked for any templates not matching these patterns
classLoaderTemplateResolver.getResolvablePatternSpec().addPattern("/layout/*.html");
classLoaderTemplateResolver.getResolvablePatternSpec().addPattern("/menu/*.html");
```

```
ServletContextTemplateResolver servletContextTemplateResolver = new
ServletContextTemplateResolver();
servletContextTemplateResolver.setOrder(Integer.valueOf(2));
```

=====

Message Resolver

The implementation being used was an

org.thymeleaf.messageresolver.StandardMessageResolver object as default in web application.

you can create your own by just implementing the org.thymeleaf.messageresolver.IMessageResolver interface.

And why would you want to have more than one message resolver? for the same reason as template resolvers:

message resolvers are ordered and if the first one cannot resolve a specific message, the second one will be asked, then the third, etc.

```
// For setting only one
templateEngine.setMessageResolver(messageResolver);
```

Template Cache 模板缓存管理

```
// Default is 50
StandardCacheManager cacheManager = new StandardCacheManager();
cacheManager.setTemplateCacheMaxSize(100);
...
templateEngine.setCacheManager(cacheManager);

// Clear the cache completely
templateEngine.clearTemplateCache();
// Clear a specific template from the cache
templateEngine.clearTemplateCacheFor("/users/userList");
```

- 上一篇 Thymeleaf基本知识
- 下一篇 spring security 3学习

猜你在找

- Java之路
- 零基础学Java系列从入门到精通
- C语言及程序设计初步
- Python编程基础视频教程(第六季)
- 8小时学会HTML网页开发

准备好了么？跳吧！

更多职位尽在 CSDN JOB

| | | | |
|-----------------|----------|--------------|----------|
| web前端工程师（资深） | 我要跳槽 | PHP工程师（高级） | 我要跳槽 |
| 上海持创信息技术有限公司 | 10-15K/月 | 上海持创信息技术有限公司 | 7-10K/月 |
| 高级android工程师 | 我要跳槽 | IOS工程师（资深） | 我要跳槽 |
| 深圳市前海蓝极科技发展有限公司 | 18-18K/月 | 上海持创信息技术有限公司 | 10-15K/月 |



查看评论

暂无评论

您还没有登录,请[\[登录\]](#)或[\[注册\]](#)

* 以上用户言论只代表其个人观点，不代表CSDN网站的观点或立场

核心技术类目

- 全部主题
- Hadoop
- AWS
- 移动游戏
- Java
- Android
- iOS
- Swift
- 智能硬件
- Docker
- OpenStack
- VPN
- Spark
- ERP
- IE10
- Eclipse
- CRM
- JavaScript
- 数据库
- Ubuntu
- NFC
- WAP
- jQuery
- BI
- HTML5
- Spring
- Apache
- .NET
- API
- HTML
- SDK
- IIS
- Fedora
- XML
- LBS
- Unity
- Splashtop
- UML
- components
- Windows Mobile
- Rails
- QEMU
- KDE
- Cassandra
- CloudStack
- FTC
- coremail
- OPhone
- CouchBase
- 云计算
- iOS6
- Rackspace
- Web App
- SpringSide
- Maemo
- Compuware
- 大数据
- aptech
- Perl
- Tornado
- Ruby
- Hibernate
- ThinkPHP
- HBase
- Pure
- Solr
- Angular
- Cloud Foundry
- Redis
- Scala
- Django
- Bootstrap

公司简介 | 招贤纳士 | 广告服务 | 银行汇款帐号 | 联系方式 | 版权声明 | 法律顾问 | 问题报告 | 合作伙伴 | 论坛反馈

网站客服 杂志客服 微博客服 webmaster@csdn.net 400-600-2320 | 北京创新乐知信息技术有限公司 版权所有 | 江苏乐知网络技术有限公司 提供商务支持
京 ICP 证 070598 号 | Copyright © 1999-2014, CSDN.NET, All Rights Reserved 