Thymeleaf 简介

Thymeleaf 是一个跟 Velocity、FreeMarker 类似的模板引擎，它可以完全替代 JSP 。相较与其他的模板引擎，它有如下三个极吸引人的特点

1. Thymeleaf 在有网络和无网络的环境下皆可运行，即它可以让美工在浏览器查看页面的静态效果，也可以让程序员在服务器查看带数据的动态页面效果。这是由于它支持 html 原型，然后在 html 标签里增加额外的属性来达到模板 + 数据的展示方式。浏览器解释 html 时会忽略未定义的标签属性，所以 thymeleaf 的模板可以静态地运行；当有数据返回到页面时，Thymeleaf 标签会动态地替换掉静态内容，使页面动态显示。
2. Thymeleaf 开箱即用的特性。它提供标准和 Spring 标准两种方言，可以直接套用模板实现 JSTL、 OGNL 表达式效果，避免每天套模板、改 JSTL、改标签的困扰。同时开发人员也可以扩展和创建自定义的方言。
3. Thymeleaf 提供 Spring 标准方言和一个与 SpringMVC 完美集成的可选模块，可以快速的实现表单绑定、属性编辑器、国际化等功能。

Spring Boot 与 Thymeleaf

如果希望以 Jar 形式发布模块则尽量不要使用 JSP 相关知识，这是因为 JSP 在内嵌的 Servlet 容器上运行有一些问题 (内嵌 Tomcat、 Jetty 不支持 Jar 形式运行 JSP，Undertow 不支持 JSP)。

Spring Boot 中推荐使用 Thymeleaf 作为模板引擎，因为 Thymeleaf 提供了完美的 Spring MVC 支持

Spring Boot 提供了大量模板引擎，包括：

- FreeMarker
- Groovy
- Mustache
- Thymeleaf
- Velocity
- Beetl

第一个 Thymeleaf 模板页

新建一个名为 spring-boot-thymeleaf 的 Spring Boot 项目，并引入 Thymeleaf 的 starter pom

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>

<dependency>
    <groupId>net.sourceforge.nekohtml</groupId>
    <artifactId>nekohtml</artifactId>
    <version>1.9.22</version>
</dependency>
```

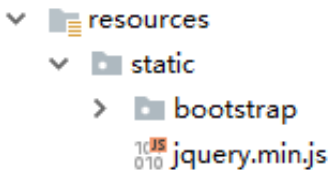
示例 JavaBean

此类用来在模板页面展示数据用，包含 name 和 age 属性

```
package com.lusifer.spring.boot.thymeleaf.bean;

public class PersonBean {
    private String name;
    private Integer age;

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public Integer getAge() {
        return age;
    }

    public void setAge(Integer age) {
        this.age = age;
    }
}
```

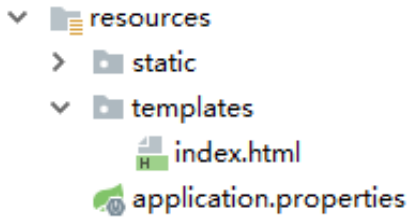
脚本样式静态文件

根据默认原则，脚本样式、图片等静态文件应该放置在 src/main/resources/static 下，这里引入 Bootstrap 和 jQuery



演示页面

根据默认原则，页面应该放置在 `src/main/resources/templates` 下，在该目录下新建 index.html




```
<!DOCTYPE html SYSTEM "http://www.thymeleaf.org/dtd/xhtml1-strict-
thymeleaf-spring4-4.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:th="http://www.thymeleaf.org">
<head>
    <meta content="text/html;charset=UTF-8" />
    <meta http-equiv="X-UA-Compatible" content="IE=edge" />
    <meta name="viewport" content="width-device-width, initial-scale=1" />
    <link th:src="@{bootstrap/css/bootstrap.min.css}" rel="stylesheet" />
    <link th:src="@{bootstrap/css/bootstrap-theme.min.css}"
rel="stylesheet" />
</head>
<body>
<div class="panel panel-primary">
    <div class="panel-heading">
        <h3 class="panel-title">访问 model</h3>
        <div class="panel-body">
            <span th:text="${singlePerson.name}" />
        </div>
    </div>
</div>

<div class="panel panel-primary">
    <div class="panel-heading">
        <h3 class="panel-title">列表</h3>
        <div class="panel-body">
            <ul class="list-group">
                <li class="list-group-item" th:each="person:${people}">
                    <span th:text="${person.name}"></span>
                    <span th:text="${person.age}"></span>
                    <button class="btn" th:onclick="'getName(\'' +
${person.name} + '\');'">获得名字</button>
                </li>
            </ul>
        </div>
    </div>
</div>
```

```
<script th:src="@{jquery.min.js}" type="text/javascript"></script>
<script th:src="@{bootstrap/js/bootstrap.min.js}" type="text/javascript">
</script>
<script th:inline="javascript">
    var single = [[${singlePerson}]];
    console.log(single.name + "/" + single.age);

    function getName(name) {
        console.log(name)
    }
</script>
</body>
</html>
```

数据准备

```
package com.lusifer.spring.boot.thymeleaf;

import com.lusifer.spring.boot.thymeleaf.bean.PersonBean;
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

import java.util.ArrayList;
import java.util.List;

@Controller
@SpringBootApplication
public class ThymeleafApplication {

    @RequestMapping(value = "/")
    public String index(Model model) {
        PersonBean person = new PersonBean();
        person.setName("张三");
        person.setAge(22);

        List<PersonBean> people = new ArrayList<>();
        PersonBean p1 = new PersonBean();
        p1.setName("李四");
        p1.setAge(23);
        people.add(p1);

        PersonBean p2 = new PersonBean();
        p2.setName("王五");
        p2.setAge(24);
```

```
        people.add(p2);

        PersonBean p3 = new PersonBean();
        p3.setName("赵六");
        p3.setAge(25);
        people.add(p3);

        model.addAttribute("singlePerson", person);
        model.addAttribute("people", people);

        return "index";
    }

    public static void main(String[] args) {
        SpringApplication.run(ThymeleafApplication.class, args);
    }
}
```

在 application.yml 中配置属性解析器

```
# Thymeleaf Start
spring:
  thymeleaf:
    cache: false # 开发时关闭缓存,不然没法看到实时页面
    mode: LEGACYHTML5 # 用非严格的 HTML
    encoding: UTF-8
    content-type: text/html
# Thymeleaf End
```

测试运行

Thymeleaf 常用语法

引入 Thymeleaf

修改 html 标签用于引入 thymeleaf 引擎，这样才可以在其他标签里使用 `th:*` 语法，这是下面语法的前提。

```
<!DOCTYPE html SYSTEM "http://www.thymeleaf.org/dtd/xhtml1-strict-
thymeleaf-spring4-4.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:th="http://www.thymeleaf.org">
```

获取变量值

```
<p th:text="'Hello! , ' + ${name} + '!'" >name</p>
```

可以看出获取变量值用 `$` 符号,对于javaBean的话使用 `变量名.属性名` 方式获取,这点和 `EL` 表达式一样.

另外 `$` 表达式只能写在th标签内部,不然不会生效,上面例子就是使用 `th:text` 标签的值替换 `p` 标签里面的值,至于 `p` 里面的原有的值只是为了给前端开发时做展示用的.这样的话很好的做到了前后端分离.

引入 URL

Thymeleaf 对于 URL 的处理是通过语法 `@{…}` 来处理的

```
<a th:href="@{http://www.baidu.com}">绝对路径</a>
<a th:href="@{/}">相对路径</a>
<a th:href="@{css/bootstrap.min.css}">Content路径,默认访问static下的css文件夹
</a>
```

类似的标签有:`th:href` 和 `th:src`

字符串替换

很多时候可能我们只需要对一大段文字中的某一处地方进行替换，可以通过字符串拼接操作完成：

```
<span th:text="'Welcome to our application, ' + ${user.name} + '!'">
```

一种更简洁的方式是：

```
<span th:text="|Welcome to our application, ${user.name}!|">
```

当然这种形式限制比较多，|...|中只能包含变量表达式${...}，不能包含其他常量、条件表达式等。

运算符

在表达式中可以使用各类算术运算符，例如+, -, *, /, %

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

逻辑运算符>, <, <=,>=，==,!=都可以使用，唯一需要注意的是使用<,>时需要用它的HTML转义符：

```
th:if="${prodStat.count} &gt; 1"
th:text="'Execution mode is ' + ( (${execMode} == 'dev')? 'Development' :
'Production')"
```

条件

if/unless

Thymeleaf 中使用 `th:if` 和 `th:unless` 属性进行条件判断，下面的例子中，标签只有在 `th:if` 中条件成立时才显示：

```
<a th:href="@{/login}" th:unless=${session.user != null}>Login</a>
```

`th:unless` 于 `th:if` 恰好相反，只有表达式中的条件不成立，才会显示其内容。

switch

Thymeleaf 同样支持多路选择 Switch 结构：

```
<div th:switch="${user.role}">
  <p th:case="'admin'">User is an administrator</p>
  <p th:case="#{roles.manager}">User is a manager</p>
</div>
```

默认属性 default 可以用 * 表示：

```
<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>
```

循环

渲染列表数据是一种非常常见的场景，例如现在有 n 条记录需要渲染成一个表格，该数据集合必须是可以遍历的，使用 `th:each` 标签：

```
<body>
  <h1>Product list</h1>

  <table>
    <tr>
      <th>NAME</th>
      <th>PRICE</th>
      <th>IN STOCK</th>
    </tr>
    <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>
  </table>

  <p>
    <a href="../home.html" th:href="@{/}">Return to home</a>
  </p>
</body>
```

可以看到，需要在被循环渲染的元素（这里是）中加入 `th:each` 标签，其中 `th:each="prod : ${prods}"` 意味着对集合变量 `prods` 进行遍历，循环变量是 `prod` 在循环体中可以通过表达式访问。

Thymeleaf 参考手册

声明

修改 html 标签用于引入 thymeleaf 引擎，这样才可以在其他标签里使用 `th:*` 语法，这是下面语法的前提。

```
<!DOCTYPE html SYSTEM "http://www.thymeleaf.org/dtd/xhtml1-strict-
thymeleaf-spring4-4.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:th="http://www.thymeleaf.org">
```

使用文本

语法	说明
{home.welcome}	使用国际化文本,国际化传参直接追加(value...)
${user.name}	使用会话属性
@{}	`<link rel="stylesheet" type="text/css" media="all"href="../../css/gtvg.css" th:href="@{/css/gtvg.css}" />`
–	–
${} 中预存对象（表达式中基本对象）	
param	获取请求参数，比如${param.name},http://localhost:8080?name=jeff
session	获取 session 的属性
application	获取 application 的属性
execInfo	有两个属性 templateName和 now(是 java 的 Calendar 对象)
ctx	
vars	
locale	
httpServletRequest	
httpSession	
–	–
th扩展标签	
th:text	普通字符串
th:utext	转义文本
th:href	
th:attr	`<img src="../../images/gtvglogo.png" th:attr="src=@{/images/gtvglogo.png},title=#{logo},alt=#{logo}" />`
th:with	定义常量
th:attrappend	
th:classappend	
th:styleappend	

其它 th 标签

th:abbr	th:accept	th:accept-charset
th:abbr	th:accept	th:accept-charset
th:accesskey	th:action	th:align

th:alt	th:archive	th:audio
th:autocomplete	th:axis	th:background
th:bgcolor	th:border	th:cellpadding
th:cellspacing	th:challenge	th:charset
th:cite	th:class	th:classid
th:codebase	th:codetype	th:cols
th:colspan	th:compact	th:content
th:contenteditable	th:contextmenu	th:data
th:datetime	th:dir	th:draggable
th:dropzone	th:enctype	th:for
th:form	th:formaction	th:formenctype
th:formmethod	th:formtarget	th:frame
th:frameborder	th:headers	th:height
th:high	th:href	th:hreflang
th:hspace	th:http-equiv	th:icon
th:id	th:keytype	th:kind
th:label	th:lang	th:list
th:longdesc	th:low	th:manifest
th:marginheight	th:marginwidth	th:max
th:maxlength	th:media	th:method
th:min	th:name	th:optimum
th:pattern	th:placeholder	th:poster
th:preload	th:radiogroup	th:rel
th:rev	th:rows	th:rowspan
th:rules	th:sandbox	th:scheme
th:scope	th:scrolling	th:size
th:sizes	th:span	th:spellcheck
th:src	th:srclang	th:standby

th:start	th:step	th:style
th:summary	th:tabindex	th:target
th:title	th:type	th:usemap
th:value	th:valuetype	th:vspace
th:width	th:wrap	th:xmlbase
th:xmllang	th:xmlspace	th:alt-title 或th:lang-xmllang（如果其中两个属性值相同）

对于 html5 元素名称的另一种友好写法

```
<table>
    <tr data-th-each="user : ${users}">
    <td data-th-text="${user.login}">...</td>
    <td data-th-text="${user.name}">...</td> </tr>
</table>
```

表达式语法

简单表达式语法

#{...} : Message 表达式

```
<p th:utext="#{home.welcome(${session.user.name})}"> Welcome to our grocery store, Sebastian Pepper!</p>
<p th:utext="#{${welcomeMsgKey}(${session.user.name})}"> Welcome to our grocery store, Sebastian Pepper!</p>
```

${} : 变量表达式

```
ongl 标准语法，方法也可以被调用
```

*{} : 选择变量表达式

```
<div th:object="${session.user}">
    <p>Name: <span th:text="*{firstName}">Sebastian</span>.</p>
    <p>Surname: <span th:text="*{lastName}">Pepper</span>.</p>
    <p>Nationality: <span th:text={nationality}">Saturn</span>.</p>
</div>
等价于
<div>
```

```
    <p>Name: <span th:text="${session.user.firstName}">Sebastian</span>.
</p>
    <p>Surname: <span th:text="${session.user.lastName}">Pepper</span>.</p>
    <p>Nationality: <span
th:text="${session.user.nationality}">Saturn</span>.</p>
</div>
```

当然了，这两者可以混合使用

还有一种方式

```
<div>
    <p>Name: <span th:text="*{session.user.name}">Sebastian</span>.</p>
    <p>Surname: <span th:text="*{session.user.surname}">Pepper</span>.</p>
    <p>Nationality: <span th:text="*
{session.user.nationality}">Saturn</span>.</p>
</div>
```

@{} : 链接 URL 表达式

```
<!-- Will produce 'http://localhost:8080/gtvg/order/details?orderId=3'
(plus rewriting) --> <a href="details.html"

th:href="@{http://localhost:8080/gtvg/order/details(orderId=${o.id})}">view
</a> <!-- Will produce '/gtvg/order/details?orderId=3' (plus rewriting) -->

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

<!-- Will produce '/gtvg/order/3/details' (plus rewriting) -->

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

变量

分类	示例
文本	‘one text’ , ‘Another one!’ ,…
数字	0 , 34 , 3.0 , 12.3 ,…
真假	true , false
文字符号	one , sometext , main ,…

字符连接

分类	示例
+	‘The name is ‘+${name}
|...|	|The name is ${name}|

算数运算

语法	示例
+, -, *, /, %	二元运算符
-	减号（一元运算符）

真假运算

分类	示例
and , or	二元运算符
! , not	否定（一元运算符）

比较运算

分类	示例
>, <, >=, <= (gt, lt, ge, le)	比较
== , != (eq , ne)	平等

条件运算

分类	示例
if-then	(if) ? (then)
if-then-else	(if) ? (then) : (else)
Default	(value) ?: (defaultvalue)

综合示例

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

内置对象

#dates

```
/*
* ============================================================================
* See javadoc API for class org.thymeleaf.expression.Dates
* ============================================================================
*/
/*
* Format date with the standard locale format
* Also works with arrays, lists or sets
*/
${#dates.format(date)}
${#dates.arrayFormat(datesArray)}
${#dates.listFormat(datesList)}
${#dates.setFormat(datesSet)}
/*
* Format date with the ISO8601 format
* Also works with arrays, lists or sets
*/
${#dates.formatISO(date)}
${#dates.arrayFormatISO(datesArray)}
${#dates.listFormatISO(datesList)}
${#dates.setFormatISO(datesSet)}
/*
* Format date with the specified pattern
* Also works with arrays, lists or sets
*/
${#dates.format(date, 'dd/MMM/yyyy HH:mm')}
${#dates.arrayFormat(datesArray, 'dd/MMM/yyyy HH:mm')}
${#dates.listFormat(datesList, 'dd/MMM/yyyy HH:mm')}
${#dates.setFormat(datesSet, 'dd/MMM/yyyy HH:mm')}
/*
* Obtain date properties
* Also works with arrays, lists or sets
*/
${#dates.day(date)} // also arrayDay(...), listDay(...), etc.
${#dates.month(date)} // also arrayMonth(...), listMonth(...), etc.
${#dates.monthName(date)} // also arrayMonthName(...), listMonthName(...),
etc.
${#dates.monthNameShort(date)} // also arrayMonthNameShort(...),
listMonthNameShort(...), etc.
${#dates.year(date)} // also arrayYear(...), listYear(...), etc.
${#dates.dayOfWeek(date)} // also arrayDayOfWeek(...), listDayOfWeek(...),
etc.
```

```
${#dates.dayOfWeekName(date)} // also arrayDayOfWeekName(...),
listDayOfWeekName(...), etc.
${#dates.dayOfWeekNameShort(date)} // also arrayDayOfWeekNameShort(...),
listDayOfWeekNameShort(...), etc.
${#dates.hour(date)} // also arrayHour(...), listHour(...), etc.
${#dates.minute(date)} // also arrayMinute(...), listMinute(...), etc.
${#dates.second(date)} // also arraySecond(...), listSecond(...), etc.
${#dates.millisecond(date)} // also arrayMillisecond(...),
listMillisecond(...), etc.
/*
* Create date (java.util.Date) objects from its components
*/
${#dates.create(year,month,day)}
${#dates.create(year,month,day,hour,minute)}
${#dates.create(year,month,day,hour,minute,second)}
${#dates.create(year,month,day,hour,minute,second)}
${#dates.create(year,month,day,hour,minute,second,millisecond)}
/*
* Create a date (java.util.Date) object for the current date and time
*/
${#dates.createNow()}
/*
* Create a date (java.util.Date) object for the current date (time set to
00:00)
*/
${#dates.createToday()}
```

#Calendars

```
/*
* =============================================================================
* See javadoc API for class org.thymeleaf.expression.Calendars
* =============================================================================
*/
/*
* Format calendar with the standard locale format
* Also works with arrays, lists or sets
*/
${#calendars.format(cal)}
${#calendars.arrayFormat(calArray)}
${#calendars.listFormat(calList)}
${#calendars.setFormat(calSet)}
/*
* Format calendar with the ISO8601 format
* Also works with arrays, lists or sets
*/
${#calendars.formatISO(cal)}
${#calendars.arrayFormatISO(calArray)}
```

```
${#calendars.listFormatISO(calList)}
${#calendars.setFormatISO(calSet)}
/*
* Format calendar with the specified pattern
* Also works with arrays, lists or sets
*/
${#calendars.format(cal, 'dd/MMM/yyyy HH:mm')}
${#calendars.arrayFormat(calArray, 'dd/MMM/yyyy HH:mm')}
${#calendars.listFormat(calList, 'dd/MMM/yyyy HH:mm')}
${#calendars.setFormat(calSet, 'dd/MMM/yyyy HH:mm')}
/*
* Obtain calendar properties
* Also works with arrays, lists or sets
*/
${#calendars.day(date)} // also arrayDay(...), listDay(...), etc.
${#calendars.month(date)} // also arrayMonth(...), listMonth(...), etc.
${#calendars.monthName(date)} // also arrayMonthName(...),
listMonthName(...), etc.
${#calendars.monthNameShort(date)} // also arrayMonthNameShort(...),
listMonthNameShort(...), etc.
${#calendars.year(date)} // also arrayYear(...), listYear(...), etc.
${#calendars.dayOfWeek(date)} // also arrayDayOfWeek(...),
listDayOfWeek(...), etc.
${#calendars.dayOfWeekName(date)} // also arrayDayOfWeekName(...),
listDayOfWeekName(...), etc.
${#calendars.dayOfWeekNameShort(date)} // also
arrayDayOfWeekNameShort(...), listDayOfWeekNameShort(...), etc.
${#calendars.hour(date)} // also arrayHour(...), listHour(...), etc.
${#calendars.hour(date)} // also arrayHour(...), listHour(...), etc.
${#calendars.minute(date)} // also arrayMinute(...), listMinute(...), etc.
${#calendars.second(date)} // also arraySecond(...), listSecond(...), etc.
${#calendars.millisecond(date)} // also arrayMillisecond(...),
listMillisecond(...), etc.
/*
* Create calendar (java.util.Calendar) objects from its components
*/
${#calendars.create(year,month,day)}
${#calendars.create(year,month,day,hour,minute)}
${#calendars.create(year,month,day,hour,minute,second)}
${#calendars.create(year,month,day,hour,minute,second,millisecond)}
/*
* Create a calendar (java.util.Calendar) object for the current date and
time
*/
${#calendars.createNow()}
/*
* Create a calendar (java.util.Calendar) object for the current date (time
set to 00:00)
*/
```

```
${#calendars.createToday()}
```

#numbers

```
/*
* ======================================================================
* See javadoc API for class org.thymeleaf.expression.Numbers
* ======================================================================
*/
/*
* ==========================
* Formatting integer numbers
* ==========================
*/
/*
* Set minimum integer digits.
* Also works with arrays, lists or sets
*/
${#numbers.formatInteger(num,3)}
${#numbers.arrayFormatInteger(numArray,3)}
${#numbers.listFormatInteger(numList,3)}
${#numbers.setFormatInteger(numSet,3)}
/*
* Set minimum integer digits and thousands separator:
* 'POINT', 'COMMA', 'WHITESPACE', 'NONE' or 'DEFAULT' (by locale).
* Also works with arrays, lists or sets
*/
${#numbers.formatInteger(num,3,'POINT')}
${#numbers.arrayFormatInteger(numArray,3,'POINT')}
${#numbers.listFormatInteger(numList,3,'POINT')}
${#numbers.setFormatInteger(numSet,3,'POINT')}
/*
* ==========================
* Formatting decimal numbers
* ==========================
*/
/*
* Set minimum integer digits and (exact) decimal digits.
* Also works with arrays, lists or sets
*/
${#numbers.formatDecimal(num,3,2)}
${#numbers.arrayFormatDecimal(numArray,3,2)}
${#numbers.listFormatDecimal(numList,3,2)}
${#numbers.setFormatDecimal(numSet,3,2)}
/*
* Set minimum integer digits and (exact) decimal digits, and also decimal
separator.
* Also works with arrays, lists or sets
```

```
*/
${#numbers.formatDecimal(num,3,2,'COMMA')}
${#numbers.arrayFormatDecimal(numArray,3,2,'COMMA')}
${#numbers.listFormatDecimal(numList,3,2,'COMMA')}
${#numbers.setFormatDecimal(numSet,3,2,'COMMA')}
/*
* Set minimum integer digits and (exact) decimal digits, and also thousands
and
* decimal separator.
* Also works with arrays, lists or sets
*/
${#numbers.formatDecimal(num,3,'POINT',2,'COMMA')}
${#numbers.arrayFormatDecimal(numArray,3,'POINT',2,'COMMA')}
${#numbers.listFormatDecimal(numList,3,'POINT',2,'COMMA')}
${#numbers.setFormatDecimal(numSet,3,'POINT',2,'COMMA')}
/*
* ========================
* Utility methods
* ========================
*/
/*
* Create a sequence (array) of integer numbers going
* from x to y
*/
${#numbers.sequence(from,to)}
${#numbers.sequence(from,to,step)}
```

#strings

```
/*
* ======================================================================
* See javadoc API for class org.thymeleaf.expression.Strings
* ======================================================================
*/
/*
* Null-safe toString()
*/
${#strings.toString(obj)} // also array*, list* and set*
/*
* Check whether a String is empty (or null). Performs a trim() operation
before check
* Also works with arrays, lists or sets
* * Also works with arrays, lists or sets
*/
${#strings.isEmpty(name)}
${#strings.arrayIsEmpty(nameArr)}
${#strings.listIsEmpty(nameList)}
${#strings.setIsEmpty(nameSet)}
```

```
/*
* Perform an 'isEmpty()' check on a string and return it if false,
defaulting to
* another specified string if true.
* Also works with arrays, lists or sets
*/
${#strings.defaultString(text,default)}
${#strings.arrayDefaultString(textArr,default)}
${#strings.listDefaultString(textList,default)}
${#strings.setDefaultString(textSet,default)}
/*
* Check whether a fragment is contained in a String
* Also works with arrays, lists or sets
*/
${#strings.contains(name,'ez')} // also array*, list* and set*
${#strings.containsIgnoreCase(name,'ez')} // also array*, list* and set*
/*
* Check whether a String starts or ends with a fragment
* Also works with arrays, lists or sets
*/
${#strings.startsWith(name,'Don')} // also array*, list* and set*
${#strings.endsWith(name,endingFragment)} // also array*, list* and set*
/*
* Substring-related operations
* Also works with arrays, lists or sets
*/
${#strings.indexOf(name,frag)} // also array*, list* and set*
${#strings.substring(name,3,5)} // also array*, list* and set*
${#strings.substringAfter(name,prefix)} // also array*, list* and set*
${#strings.substringBefore(name,suffix)} // also array*, list* and set*
${#strings.replace(name,'las','ler')} // also array*, list* and set*
/*
* Append and prepend
* Also works with arrays, lists or sets
*/
${#strings.prepend(str,prefix)} // also array*, list* and set*
${#strings.append(str,suffix)} // also array*, list* and set*
/*
* Change case
* Also works with arrays, lists or sets
*/
${#strings.toUpperCase(name)} // also array*, list* and set*
${#strings.toLowerCase(name)} // also array*, list* and set*
/*
* Split and join
*/
${#strings.arrayJoin(namesArray,',')}
${#strings.listJoin(namesList,',')}
${#strings.setJoin(namesSet,',')}
```

```
${#strings.arraySplit(namesStr,',')} // returns String[]
${#strings.listSplit(namesStr,',')} // returns List<String>
${#strings.setSplit(namesStr,',')} // returns Set<String>
/*
* Trim
* Also works with arrays, lists or sets
*/
${#strings.trim(str)} // also array*, list* and set*
/*
* Compute length
* Also works with arrays, lists or sets
*/
${#strings.length(str)} // also array*, list* and set*
/*
* Abbreviate text making it have a maximum size of n. If text is bigger, it
* will be clipped and finished in "..."
* Also works with arrays, lists or sets
*/
${#strings.abbreviate(str,10)} // also array*, list* and set*
/*
* Convert the first character to upper-case (and vice-versa)
*/
${#strings.capitalize(str)} // also array*, list* and set*
${#strings.unCapitalize(str)} // also array*, list* and set*
/*
* Convert the first character of every word to upper-case
*/
${#strings.capitalizeWords(str)} // also array*, list* and set*
${#strings.capitalizeWords(str,delimiters)} // also array*, list* and set*
/*
* Escape the string
*/
${#strings.escapeXml(str)} // also array*, list* and set*
${#strings.escapeJava(str)} // also array*, list* and set*
${#strings.escapeJavaScript(str)} // also array*, list* and set*
${#strings.unescapeJava(str)} // also array*, list* and set*
${#strings.unescapeJavaScript(str)} // also array*, list* and set*
/*
* Null-safe comparison and concatenation
*/
${#strings.equals(first, second)}
${#strings.equalsIgnoreCase(first, second)}
${#strings.concat(values...)}
${#strings.concatReplaceNulls(nullValue, values...)}
/*
* Random
*/
${#strings.randomAlphanumeric(count)}
```

#objects

```
/*
* =============================================================================
* See javado
```

循环

```
<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>

迭代器的状态
index：当前的索引，从0开始
count：当前的索引，从1开始
size：总数
current:
even/odd:
first
last
<table>
    <tr>
        <th>NAME</th>
        <th>PRICE</th>
        <th>IN STOCK</th>
    </tr>
    <tr th:each="prod,iterStat : ${prods}" th:class="${iterStat.odd}?
'odd'">
    <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>
</table>
```

判断

if

```
<a href="comments.html" th:href="@{/product/comments(prodId=${prod.id})}"
th:if="${not #lists.isEmpty(prod.comments)}">view</a>
```

unless

```
<a href="comments.html" th:href="@{/comments(prodId=${prod.id})}"
th:unless="${#lists.isEmpty(prod.comments)}">view</a>
```

switch

```
<div th:switch="${user.role}">
    <p th:case="'admin'">User is an administrator</p> <p th:case="#
{roles.manager}">User is a manager</p>
</div>

<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>
```

模板布局

```
th:fragment

示例

templates/footer.html

<!DOCTYPE html SYSTEM "http://www.thymeleaf.org/dtd/xhtml1-strict-thymeleaf-
4.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
    <body>
         <div th:fragment="copy">
            © 2011 The Good Thymes Virtual Grocery
         </div>
    </body>
</html>

templates/index.html中使用

    <body> ...
        <div th:include="footer :: copy"></div>
    </body>

或者
    ...
```

```
    <div id="copy-section">
        © 2011 The Good Thymes Virtual Grocery
    </div>
    ...
```

使用

```
    <body> ...
        <div th:include="footer :: #copy-section"></div>
    </body>
```

th:include 和 th:replace 区别

th:include 加入代码

th:replace 替换代码

模板传参：参数传递顺序不强制

定义

```
<div th:fragment="frag (onevar,twovar)">
    <p th:text="${onevar} + ' - ' + ${twovar}">...</p>
</div>
```

使用

```
<div th:include="::frag (${value1},${value2})">...</div>

<div th:include="::frag (onevar=${value1},twovar=${value2})">...</div>
等价于 <div th:include="::frag" th:with="onevar=${value1},twovar=${value2}">)
```

th:block

```
<table>
    <th:block th:each="user : ${users}">
    <tr>
        <td th:text="${user.login}">...</td> <td th:text="${user.name}">...
</td>
    </tr>
    <tr>
        <td colspan="2" th:text="${user.address}">...</td>
    </tr>
    </th:block>
</table>
```

推荐下面写法（编译前看不见）

```
<table>
    <tr>
        <td th:text="${user.login}">...</td>
        <td th:text="${user.name}">...</td> </tr>
        <tr>
        <td colspan="2" th:text="${user.address}">...</td>
    </tr>
    <!--/*/ </th:block> /*/-->
</table>
```

results matching ""

No results matching ""

th:inline

th:inline 可以等于 text , javascript(dart) , none

`text: [[…]]`

```
<p th:inline="text">Hello, [[#{test}]]</p>
```

`javascript: /*[[…]]*/`

```
<script th:inline="javascript">
    var username = /*[[
        #{test}
    ]]*/;
    var name = /*[[
        ${param.name[0]}+${execInfo.templateName}+'-
'+${#dates.createNow()}+'-'+${#locale}
    ]]*/;
</script>
<script th:inline="javascript">

/*<![CDATA[*/

    var username = [[#{test}]];
```

```
    var name = [[${param.name[0]}+${execInfo.templateName}+'-
'+${#dates.createNow()}+'-'+${#locale}]];

/*]]>*/

</script>
```

adding code: /* [+…+]*/

```
var x = 23;
/*[+
var msg = 'Hello, ' + [[${session.user.name}]]; +]*/
var f = function() {
...
```

removind code: /[- / and /* -]*/

```
var x = 23;
/*[- */
var msg = 'This is a non-working template'; /* -]*/
var f = function() {
...
```

Thymeleaf 自定义标签

我们知道在 JSP 中，可以使用 JSTL 标签简化开发，并且 JSTL 还具备自定义标签的功能，那 Thymeleaf 如何实现自定义标签呢

Maven

```
<dependency>
    <groupId>org.thymeleaf</groupId>
    <artifactId>thymeleaf-spring4</artifactId>
    <version>3.0.9.RELEASE</version>
</dependency>
<dependency>
    <groupId>nz.net.ultraq.thymeleaf</groupId>
    <artifactId>thymeleaf-layout-dialect</artifactId>
    <version>2.3.0</version>
</dependency>
```

创建方言类

方言类需要继承 Thymeleaf 的 `AbstractProcessorDialect`

```
package com.ooqiu.gaming.server.web.admin.config.thymeleaf.dialect;

import
com.ooqiu.gaming.server.web.admin.config.thymeleaf.tag.SysDictTagProcessor;
import org.thymeleaf.dialect.AbstractProcessorDialect;
import org.thymeleaf.processor.IProcessor;
import org.thymeleaf.standard.StandardDialect;
import org.thymeleaf.standard.processor.StandardXmlNsTagProcessor;
import org.thymeleaf.templatemode.TemplateMode;

import java.util.HashSet;
import java.util.Set;

/**
 * Thymeleaf 方言：系统用
 * <p>Title: SysDialect</p>
 * <p>Description: </p>
 *
 * @author Lusifer
 * @version 1.0.0
 * @date 2018/3/4 9:34
 */
public class SysDialect extends AbstractProcessorDialect {
    // 定义方言名称
    private static final String DIALECT_NAME = "Sys Dialect";

    public SysDialect() {
        // 设置自定义方言与“方言处理器”优先级相同
        super(DIALECT_NAME, "sys", StandardDialect.PROCESSOR_PRECEDENCE);
    }

    /**
     * 元素处理器
     * @param dialectPrefix 方言前缀
     * @return
     */
    @Override
    public Set<IProcessor> getProcessors(String dialectPrefix) {
        Set<IProcessor> processors = new HashSet<IProcessor>();

        // 添加自定义标签处理器
        processors.add(new SysDictTagProcessor(dialectPrefix));
        processors.add(new StandardXmlNsTagProcessor(TemplateMode.HTML,
dialectPrefix));
        return processors;
    }
}
```

创建标签处理器

标签处理器需要继承 Thymeleaf 的 `AbstractElementTagProcessor`

```
package com.ooqiu.gaming.server.web.admin.config.thymeleaf.tag;

import org.thymeleaf.context.ITemplateContext;
import org.thymeleaf.model.IModel;
import org.thymeleaf.model.IModelFactory;
import org.thymeleaf.model.IProcessableElementTag;
import org.thymeleaf.processor.element.AbstractElementTagProcessor;
import org.thymeleaf.processor.element.IElementTagStructureHandler;
import org.thymeleaf.templatemode.TemplateMode;

/**
 * 自定义标签
 * <p>Title: SysDictTagProcessor</p>
 * <p>Description: </p>
 *
 * @author Lusifer
 * @version 1.0.0
 * @date 2018/3/4 10:52
 */
public class SysDictTagProcessor extends AbstractElementTagProcessor {

    // 标签名
    private static final String TAG_NAME = "dict";

    // 优先级
    private static final int PRECEDENCE = 10000;

    public SysDictTagProcessor(String dialectPrefix) {
        super(
                // 此处理器将仅应用于HTML模式
                TemplateMode.HTML,

                // 要应用于名称的匹配前缀
                dialectPrefix,

                // 标签名称：匹配此名称的特定标签
                TAG_NAME,

                // 将标签前缀应用于标签名称
                true,

                // 无属性名称：将通过标签名称匹配
                null,
```

```
                // 没有要应用于属性名称的前缀
                false,

                // 优先(内部方言自己的优先)
                PRECEDENCE
        );
    }

    /**
     * 处理自定义标签 DOM 结构
     *
     * @param iTemplateContext             模板页上下文
     * @param iProcessableElementTag       待处理标签
     * @param iElementTagStructureHandler 元素标签结构处理器
     */
    @Override
    protected void doProcess(ITemplateContext iTemplateContext,
IProcessableElementTag iProcessableElementTag, IElementTagStructureHandler
iElementTagStructureHandler) {
        // 创建将替换自定义标签的 DOM 结构
        IModelFactory modelFactory = iTemplateContext.getModelFactory();
        IModel model = modelFactory.createModel();

        // 需要替换的页面元素
        model.add(modelFactory.createOpenElementTag("div"));
        model.add(modelFactory.createText("Hello Thymeleaf Dialect"));
        model.add(modelFactory.createCloseElementTag("div"));

        // 利用引擎替换整个标签
        iElementTagStructureHandler.replaceWith(model, false);
    }
}
```

注入方言

```
package com.ooqiu.gaming.server.web.admin.config.thymeleaf;

import
com.ooqiu.gaming.server.web.admin.config.thymeleaf.dialect.SysDialect;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;

/**
 * Thymeleaf 方言配置
 * <p>Title: ThymeleafDialectConfig</p>
 * <p>Description: </p>
 *
 * @author Lusifer
```

```
 * @version 1.0.0
 * @date 2018/3/4 10:57
 */
@Configuration
public class ThymeleafDialectConfig {

    /**
     * 系统方言
     *
     * @return
     */
    @Bean
    public SysDialect sysDialect() {
        return new SysDialect();
    }
}
```

前台 HTML 中声明使用

增加命名空间配置：`xmlns:sys=""`

```
<!DOCTYPE html SYSTEM "http://www.thymeleaf.org/dtd/xhtml1-strict-thymeleaf-spring4-4.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
xmlns:th="http://www.thymeleaf.org" xmlns:sys="">
```

使用标签

```
 <sys:dict />
```