基于Springboot 的Thymeleaf模板引擎

第一章：Thymeleaf 模板引擎的简介

Thymeleaf是面向Web和独立环境的服务器Java模板引擎，能够处理HTML、XML、JavaScript、CSS甚至纯文本数据。

Thymeleaf的主要目标在于提供一种可被浏览器正确显示的、格式良好的模板创建方式，因此也可以用作静态建模。

Thymeleaf的可扩展性也非常棒。你可以使用它定义自己的模板属性集合，这样就可以计算自定义表达式并使用自定义逻辑。这意味着Thymeleaf还可以作为模板引擎框架。

1.1、Thymeleaf可以处理什么模板

Thymeleaf允许我们处理六种模板：

1. **HTML**

   该HTML模板模式将允许任何类型的HTML的输入，包括HTML5，HTML4和XHTML。

2. **XML**

   该XML模板模式将允许XML输入。在这种情况下，代码应该是格式良好的 - 没有未封闭的标签，没有未加引号的属性等等，如果发现格式错误，解析器将会抛出异常。

3. **TEXT**

该TEXT模板模式将允许非标记性质的模板使用特殊的语法。这种模板的例子可能是文本电子邮件或模板文档。

4. **JAVASCRIPT**

该JAVASCRIPT模板模式将允许在Thymeleaf应用程序的JavaScript文件的处理。

5. **CSS**

该CSS模板模式将允许参与Thymeleaf应用CSS文件的处理。

6. **RAW**

   该RAW模板模式将根本不处理模板。

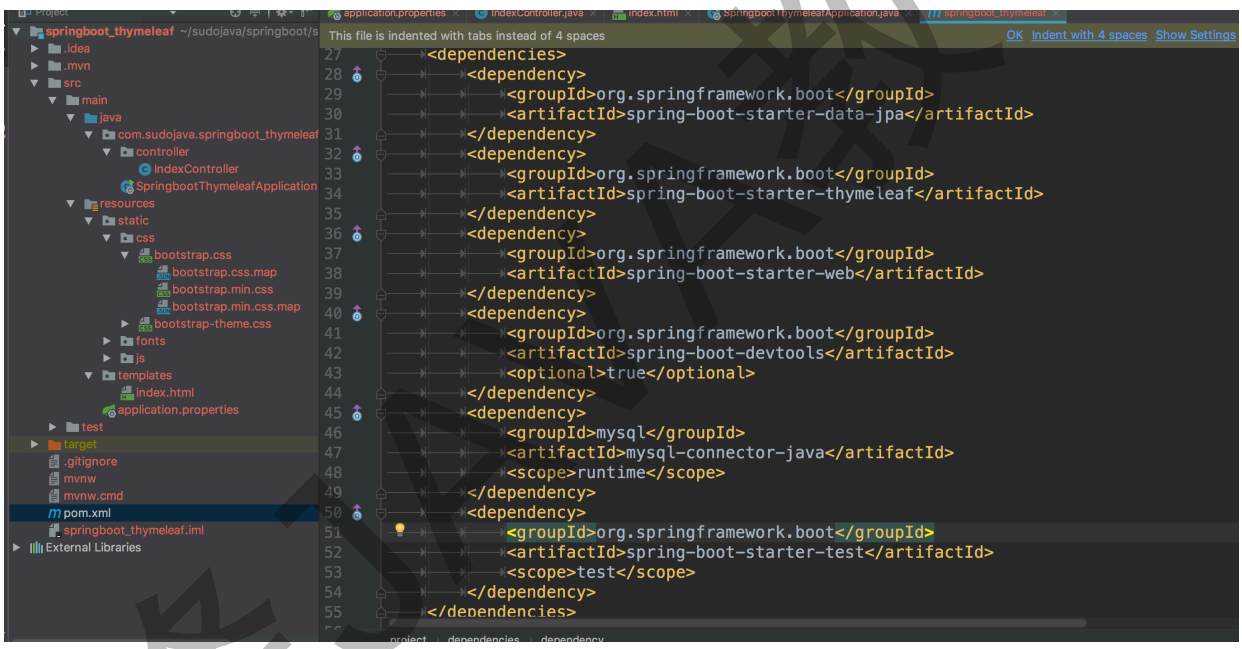
1.3、标准的Thymeleaf模板

```
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">
  <head>
    <title>Good Thymes Virtual Grocery</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link rel="stylesheet" type="text/css" media="all"
          href="../../css/gtvg.css" th:href="@{/css/gtvg.css}" />
  </head>
  <body>
    <p th:text="#{home.welcome}">Welcome to our grocery store!</p>
  </body>
</html>
```

第二章：搭建基于SpringBoot的Thymeleaf环境

工程环境图



pom.xml文件配置

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
   xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
   <modelVersion>4.0.0</modelVersion>
   <groupId>com.sudojava.springboot_thymeleaf</groupId>
   <artifactId>springboot_thymeleaf</artifactId>
   <version>0.0.1-SNAPSHOT</version>
   <packaging>jar</packaging>
   <name>springboot_thymeleaf</name>
   <description>Demo project for Spring Boot</description>
   <parent>
```

```
        <groupId>org.springframework.boot</groupId>
        <artifactId>spring-boot-starter-parent</artifactId>
        <version>1.5.8.RELEASE</version>
        <relativePath/> <!-- lookup parent from repository -->
    </parent>
    <properties>
        <project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>
        <project.reporting.outputEncoding>UTF-8</project.reporting.outputEncoding>
        <java.version>1.8</java.version>
    </properties>
    <dependencies>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-data-jpa</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-thymeleaf</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-web</artifactId>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <optional>true</optional>
        </dependency>
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <scope>runtime</scope>
        </dependency>
        <dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-starter-test</artifactId>
            <scope>test</scope>
        </dependency>
    </dependencies>
    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
    </build>
</project>
```

配置application.properties文件

```
spring.datasource.driver-class-name=com.mysql.jdbc.Driver
spring.datasource.username=root
spring.datasource.password=root
spring.datasource.url=jdbc:mysql://localhost:3306/mydb?
useUnicode=true&characterEncoding=UTF-8
#配置Hibernate的基本属性
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.MySQL5Dialect
```

第三章：标准表达式语法

Thymeleaf Standard Expression是Thymeleaf中语法中最重要的部分之一，以下是标准表达式功能的总结：

注意：Thymeleaf模板语法，都是以th属性开头：

1、简单表达式：

- 变量表达式：${...}
- 选择变量表达式：*{...}
- 消息表达式：#{...}
- 链接URL表达式：@{...}
- 片段表达式：~{...}

2、字面

- 文本字面：'one text'，'Another one!'，...
- 号码文字：12，34，6，7，8，.....
- 布尔文字：true、false
- 空文字：null
- 文字标记：one、sometext、main, ...

3、文字操作

- 字符串连接：+
- 文字替换：|The name is ${name}|

4、算术运算：

- 二元运算符：+, - , * , / , %
- 减号(一元运算符)

5、布尔运算符：

- 二元运算符：and，or
- 布尔否定：！，not

6、逻辑运算符

- 比较运算符：> , < , >= ,<= (gt,lt,ge,le)
- 布尔运算符：==，!=（eq，ne）

7、IF ELSE判断：

- IF-THEN： (if) ? (then)
- IF-THEN-ELSE： (if) ? (then) : (else)

3.1.1、变量表达式

${...} 表达式实际上是包含了上下文中的变量映射执行的OGNL(对象图导航语言)表达式。

测试案例：

HTML页面代码

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title></title>

    <meta charset="utf-8"/>
    <link th:href="@{css/bootstrap.min.css}" rel="stylesheet"/>

</head>
<body>
<p>Today is: <span th:text="${day}"></span>.</p>
<p>FirstName is <span th:text="${user.firstName}"></span> </p>
<p>lastName is <span th:text="${user.lastName}"></span> </p>
</body>
</html>
```

Java代码

```
package com.sudojava.springboot_thymeleaf.controller;

import com.sudojava.springboot_thymeleaf.domain.User;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;

import javax.servlet.http.HttpServletRequest;


@Controller
public class IndexController {

    @RequestMapping(value = "/",method = RequestMethod.GET)
```

```
    public String index(Model model){
        model.addAttribute("day","2007-12-12");
        User user = new User();
        user.setFirstName("zhang");
        user.setLastName("san");
        model.addAttribute("user",user);
        return "/index";
    }
}
```

3.1.2、选择表达式(星号语法)

可以简单理解为内层是对外层对象的引用。

HTML代码

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <title></title>

    <meta charset="utf-8"/>
    <link th:href="@{css/bootstrap.min.css}" rel="stylesheet"/>

</head>
<body>
<div th:object="${user}">
    <p>Name: <span th:text="*{firstName}">Sebastian</span>.</p>
    <p>Surname: <span th:text="*{lastName}">Pepper</span>.</p>
</div>
</body>
</html>
```

以上代码等同于:

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

Java代码

```
package com.sudojava.springboot_thymeleaf.controller;

import com.sudojava.springboot_thymeleaf.domain.User;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
```

```
import javax.servlet.http.HttpServletRequest;


@Controller
public class IndexController {

    @RequestMapping(value = "/",method = RequestMethod.GET)
    public String index(Model model){
        model.addAttribute("day","2007-12-12");
        User user = new User();
        user.setFirstName("zhang");
        user.setLastName("san");
        model.addAttribute("user",user);
        return "/index";
    }

}
```

我们来看看Thymeleaf官网给提供的例子：

```
/*
 * Access to properties using the point (.). Equivalent to calling property
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.
 */
${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.
 */
${countriesByCode.ES}
${personsByName['Stephen Zucchini'].age}

/*
 * Indexed access to arrays or collections is also performed with brackets,
 * writing the index without quotes.
 */
${personsArray[0].name}

/*
 * Methods can be called, even with arguments.
 */
```

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

3.1.3、表达式基本对象

在上下文变量上评估OGNL表达式时，一些对象可用于表达式以获得更高的灵活性。这些对象将被引用（按照OGNL标准），从#符号开始：

- `#ctx`：上下文对象。
- `#vars:` 上下文变量。
- `#locale`：上下文语言环境。
- `#request`:(只在Web上下文中）`HttpServletRequest`对象。
- `#response`:(只在Web上下文中）`HttpServletResponse`对象。
- `#session`:(只在Web上下文中）`HttpSession`对象。
- `#servletContext`:(只在Web上下文中）`ServletContext`对象。

3.1.4、表达式工具对象

除了这些基本的对象之外，Thymeleaf还会为我们提供一套实用对象，帮助我们在表达式中执行常见任务。

- `#execInfo`：关于正在处理的模板的信息。
- `#messages`：在变量表达式中获得外部化消息的方法，与使用#{...}语法获得的方式相同。
- `#uris`：用于转义URL / URI部分的方法
- `#conversions`：执行配置的*转换服务的方法*（如果有的话）。
- `#dates`：`java.util.Date`对象的方法：格式化，组件提取等
- `#calendars`：类似于`#dates`，但是对于`java.util.Calendar`物体。
- `#numbers`：格式化数字对象的方法。
- `#strings`：`String`对象的方法：contains，startsWith，prepending / appending等
- `#objects`：一般对象的方法。
- `#bools`：布尔评估的方法。
- `#arrays`：数组的方法。
- `#lists`：列表的方法。
- `#sets`：套的方法。
- `#maps`：地图的方法。
- `#aggregates`：在数组或集合上创建聚合的方法。
- `#ids`：处理可能重复的id属性的方法（例如，作为迭代的结果）。

3.1.5、URL链接地址

URL主要是实现了超链接点击的作用，包括地址栏传递参数等功能。

URL语法如下：

- 绝对网址：`http://www.thymeleaf.org`
- 相对URL，可以是：

1. 页面相对：`user/login.html`
2. 与上下文相关:(`/itemdetails?id=3`服务器中的上下文名称将自动添加)
3. 服务器相对:(`~/billing/processInvoice`允许在同一服务器中的另一个上下文（=应用程序）中调用URL。
4. 与协议相关的网址：`//code.jquery.com/jquery-2.0.3.min.js`

官网给我们的展示案例：

```
<!-- 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>
```

注意事项

- `th:href`是一个修饰符属性：一旦处理完毕，它将计算要使用的链接URL并将该值设置为该标记的`href`属性`<a>`。

链接的属性可以省略不写。

- 我们被允许使用URL参数的表达式（如您所见`orderId=${o.id}`）。所需的URL参数编码操作也将自动执行。
- 如果需要几个参数，则用逗号分隔：`@{/order/process(execId=${execId},execType='FAST')}`
- URL路径中也允许使用变量模板：`@{/order/{orderId}/details(orderId=${orderId})}`
- 以`/`（例如：）开头的相对URL`/order/details`将自动以应用程序上下文名称作为前缀。
- 如果cookie未启用或尚未知道，则`";jsessionid=..."`可能会向相对URL添加后缀，以便会话被保留。这就是所谓的*URL重写*，Thymeleaf允许你通过使用`response.encodeURL(...)`Servlet API 的机制为每个URL 插入自己的重写过滤器。
- 该`th:href`属性允许我们（可选地）`href`在我们的模板中有一个工作的静态属性，以便我们的模板链接在浏览器直接打开以供原型设计时保持可浏览的状态。

3.1.6、布尔表达式

布尔文字是`true`和`false`。例如：

```
<div th:if="${user.isAdmin()} == false"> ...
```

在这个例子中，这`== false`是写在括号外，所以是Thymeleaf照顾它。如果它是写在花括号内，那么OGNL / SpringEL引擎将负责这个操作：

```
<div th:if="${user.isAdmin() == false}"> ...
```

3.1.7、算术运算符号

一些算术运算也可用：`+`，`-`，`*`，`/`和`%`。

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

请注意，这些运算符也可以在OGNL变量表达式中使用（在这种情况下，将由OGNL而不是Thymeleaf标准表达式引擎执行）：

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

请注意，其中一些运算符存在文本别名：`div`（`/`），`mod`（`%`）。

第四章：Iteration迭代输出

Iteration主要是用于迭代输出集合的数据，集合的类型有List、Set、Map、Collection等集合。

这个`java.util.List`类不是唯一可用于Thymeleaf迭代的值。有一组相当完整的对象被认为可以被一个属性*迭代*`th:each`：

- 任何对象的实现`java.util.Iterable`
- 任何对象的实现`java.util.Enumeration`。
- 任何实现的对象`java.util.Iterator`，其值将被迭代器返回使用，而不需要缓存内存中的所有值。
- 任何对象的实现`java.util.Map`。当迭代映射时，iter变量将是类的`java.util.Map.Entry`。
- 任何数组。
- 任何其他对象将被视为包含对象本身的单值列表。

4.1、使用th:each迭代

我们可以使用th:each来迭代输出模板的产品列表

```
<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

  <head>
    <title>Good Thymes Virtual Grocery</title>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8" />
    <link rel="stylesheet" type="text/css" media="all"
          href="../../../css/gtvg.css" th:href="@{/css/gtvg.css}" />
  </head>
```

```
  <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>

</html>
```

4.2、th:each的状态变量

使用时`th:each`，Thymeleaf提供了一个有用的机制来跟踪你的迭代*状态*：*状态变量*。

状态变量在一个`th:each`属性中定义并包含以下数据：

- 当前*迭代索引*，从0开始。这是`index`属性。
- 当前*迭代索引*，从1开始。这是`count`属性。
- 迭代变量中的元素总数。这是`size`财产。
- 每个迭代的*iter变量*。这是`current`财产。
- 目前的迭代是偶数还是奇数。这些是`even/odd`布尔属性。
- 目前的迭代是否是第一个。这是`first`布尔属性。
- 目前的迭代是否是最后一个。这是`last`布尔属性。

```
<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>
```

4.3、通过懒惰的数据检索进行优化

有时我们可能想要优化对数据集合的检索（例如从数据库），以便只有在真正要使用这些集合时才能检索这些集合。

这个变量可以在不知道其*懒惰的*情况下使用，代码如下：

```
<ul>
  <li th:each="u : ${users}" th:text="${u.name}">user name</li>
</ul>
```

```
<ul th:if="${condition}">
  <li th:each="u : ${users}" th:text="${u.name}">user name</li>
</ul>
```

第五章：附录A：表达基本对象

Some objects and variable maps are always available to be invoked. Let's see them:

Base objects

- **#ctx** : the context object. An implementation of `org.thymeleaf.context.IContext` or `org.thymeleaf.context.IWebContext` depending on our environment (standalone or web).

Note `#vars` and `#root` are synomyns for the same object, but using `#ctx` is recommended.

```
/*
 * ======================================================================
 * See javadoc API for class org.thymeleaf.context.IContext
 * ======================================================================
 */

${#ctx.locale}
${#ctx.variableNames}
```

```
/*
 * ======================================================================
 * See javadoc API for class org.thymeleaf.context.IWebContext
 * ======================================================================
 */

${#ctx.request}
${#ctx.response}
${#ctx.session}
${#ctx.servletContext}
```

- **#locale** : direct access to the `java.util.Locale` associated with current request.

```
${#locale}
```

Web context namespaces for request/session attributes, etc.

When using Thymeleaf in a web environment, we can use a series of shortcuts for accessing request parameters, session attributes and application attributes:

> Note these are not *context objects*, but maps added to the context as variables, so we access them without `#`. In some way, they act as *namespaces*.

- **param** : for retrieving request parameters. `${param.foo}` is a `String[]` with the values of the `foo` request parameter, so `${param.foo[0]}` will normally be used for getting the first value.

```
/*
 * ======================================================================
 * See javadoc API for class org.thymeleaf.context.WebRequestParamsVariablesMap
 * ======================================================================
 */

${param.foo}          // Retrieves a String[] with the values of request
parameter 'foo'
${param.size()}
${param.isEmpty()}
${param.containsKey('foo')}
...
```

- **session** : for retrieving session attributes.

```
/*
 * ======================================================================
 * See javadoc API for class org.thymeleaf.context.WebSessionVariablesMap
 * ======================================================================
 */

${session.foo}                // Retrieves the session atttribute 'foo'
${session.size()}
${session.isEmpty()}
${session.containsKey('foo')}
...
```

- **application** : for retrieving application/servlet context attributes.

```
/*
 * ========================================================================
 * See javadoc API for class org.thymeleaf.context.WebServletContextVariablesMap
 * ========================================================================
 */

${application.foo}             // Retrieves the ServletContext atttribute 'foo'
${application.size()}
${application.isEmpty()}
${application.containsKey('foo')}
...
```

Note there is **no need to specify a namespace for accessing request attributes** (as opposed to *request parameters*) because all request attributes are automatically added to the context as variables in the context root:

```
${myRequestAttribute}
```

Web context objects

Inside a web environment there is also direct access to the following objects (note these are objects, not maps/namespaces):

- **#request** : direct access to the `javax.servlet.http.HttpServletRequest` object associated with the current request.

```
${#request.getAttribute('foo')}
${#request.getParameter('foo')}
${#request.getContextPath()}
${#request.getRequestName()}
...
```

- **#session** : direct access to the `javax.servlet.http.HttpSession` object associated with

the current request.

```
${#session.getAttribute('foo')}
${#session.id}
${#session.lastAccessedTime}
...
```

- **#servletContext** : direct access to the `javax.servlet.ServletContext` object associated with the current request.

```
${#servletContext.getAttribute('foo')}
${#servletContext.contextPath}
...
```

第六章：附录B：表达式实用程序对象

Execution Info

- **#execInfo** : expression object providing useful information about the template being processed inside Thymeleaf Standard Expressions.

```
/*
 * ======================================================================
 * See javadoc API for class org.thymeleaf.expression.ExecutionInfo
 * ======================================================================
 */

/*
 * Return the name and mode of the 'leaf' template. This means the template
 * from where the events being processed were parsed. So if this piece of
 * code is not in the root template "A" but on a fragment being inserted
 * into "A" from another template called "B", this will return "B" as a
 * name, and B's mode as template mode.
 */
${#execInfo.templateName}
${#execInfo.templateMode}

/*
 * Return the name and mode of the 'root' template. This means the template
 * that the template engine was originally asked to process. So if this
 * piece of code is not in the root template "A" but on a fragment being
 * inserted into "A" from another template called "B", this will still
 * return "A" and A's template mode.
 */
${#execInfo.processedTemplateName}
${#execInfo.processedTemplateMode}

/*
```

```
 * Return the stacks (actually, List<String> or List<TemplateMode>) of
 * templates being processed. The first element will be the
 * 'processedTemplate' (the root one), the last one will be the 'leaf'
 * template, and in the middle all the fragments inserted in nested
 * manner to reach the leaf from the root will appear.
 */
${#execInfo.templateNames}
${#execInfo.templateModes}

/*
 * Return the stack of templates being processed similarly (and in the
 * same order) to 'templateNames' and 'templateModes', but returning
 * a List<TemplateData> with the full template metadata.
 */
${#execInfo.templateStack}
```

Messages

- **#messages** : utility methods for obtaining externalized messages inside variables expressions, in the same way as they would be obtained using `#{...}` syntax.

```
/*
 * =========================================================================
 * See javadoc API for class org.thymeleaf.expression.Messages
 * =========================================================================
 */

/*
 * Obtain externalized messages. Can receive a single key, a key plus arguments,
 * or an array/list/set of keys (in which case it will return an array/list/set of
 * externalized messages).
 * If a message is not found, a default message (like '??msgKey??') is returned.
 */
${#messages.msg('msgKey')}
${#messages.msg('msgKey', param1)}
${#messages.msg('msgKey', param1, param2)}
${#messages.msg('msgKey', param1, param2, param3)}
${#messages.msgWithParams('msgKey', new Object[] {param1, param2, param3,
param4})}
${#messages.arrayMsg(messageKeyArray)}
${#messages.listMsg(messageKeyList)}
${#messages.setMsg(messageKeySet)}

/*
 * Obtain externalized messages or null. Null is returned instead of a default
 * message if a message for the specified key is not found.
 */
${#messages.msgOrNull('msgKey')}
${#messages.msgOrNull('msgKey', param1)}
```

```
${#messages.msgOrNull('msgKey', param1, param2)}
${#messages.msgOrNull('msgKey', param1, param2, param3)}
${#messages.msgOrNullWithParams('msgKey', new Object[] {param1, param2, param3,
param4})}
${#messages.arrayMsgOrNull(messageKeyArray)}
${#messages.listMsgOrNull(messageKeyList)}
${#messages.setMsgOrNull(messageKeySet)}
```

URIs/URLs

- **#uris** : utility object for performing URI/URL operations (esp. escaping/unescaping) inside Thymeleaf Standard Expressions.

```
/*
 * ============================================================================
 * See javadoc API for class org.thymeleaf.expression.Uris
 * ============================================================================
 */

/*
 * Escape/Unescape as a URI/URL path
 */
${#uris.escapePath(uri)}
${#uris.escapePath(uri, encoding)}
${#uris.unescapePath(uri)}
${#uris.unescapePath(uri, encoding)}

/*
 * Escape/Unescape as a URI/URL path segment (between '/' symbols)
 */
${#uris.escapePathSegment(uri)}
${#uris.escapePathSegment(uri, encoding)}
${#uris.unescapePathSegment(uri)}
${#uris.unescapePathSegment(uri, encoding)}

/*
 * Escape/Unescape as a Fragment Identifier (#frag)
 */
${#uris.escapeFragmentId(uri)}
${#uris.escapeFragmentId(uri, encoding)}
${#uris.unescapeFragmentId(uri)}
${#uris.unescapeFragmentId(uri, encoding)}

/*
 * Escape/Unescape as a Query Parameter (?var=value)
 */
${#uris.escapeQueryParam(uri)}
${#uris.escapeQueryParam(uri, encoding)}
${#uris.unescapeQueryParam(uri)}
```

```
${#uris.unescapeQueryParam(uri, encoding)}
```

Conversions

- **#conversions** : utility object that allows the execution of the *Conversion Service* at any point of a template:

```
/*
 * =====================================================================
 * See javadoc API for class org.thymeleaf.expression.Conversions
 * =====================================================================
 */

/*
 * Execute the desired conversion of the 'object' value into the
 * specified class.
 */
${#conversions.convert(object, 'java.util.TimeZone')}
${#conversions.convert(object, targetClass)}
```

Dates

- **#dates** : utility methods for `java.util.Date` objects:

```
/*
 * =====================================================================
 * 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,millisecond)}

/*
 * Create a date (java.util.Date) object for the current date and time
 */
${#dates.createNow()}

${#dates.createNowForTimeZone()}
```

```
/*
 * Create a date (java.util.Date) object for the current date (time set to 00:00)
 */
${#dates.createToday()}

${#dates.createTodayForTimeZone()}
```

Calendars

- **#calendars** : analogous to `#dates`, but for `java.util.Calendar` objects:

```
/*
 * ======================================================================
 * 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.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)}

${#calendars.createForTimeZone(year,month,day,timeZone)}
${#calendars.createForTimeZone(year,month,day,hour,minute,timeZone)}
${#calendars.createForTimeZone(year,month,day,hour,minute,second,timeZone)}
${#calendars.createForTimeZone(year,month,day,hour,minute,second,millisecond,timeZo
ne)}

/*
 * Create a calendar (java.util.Calendar) object for the current date and time
 */
${#calendars.createNow()}

${#calendars.createNowForTimeZone()}

/*
 * Create a calendar (java.util.Calendar) object for the current date (time set to
00:00)
 */
${#calendars.createToday()}

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

Numbers

- **#numbers** : utility methods for number objects:

```
/*
 * ======================================================================
 * 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')}


/*
 * ====================
 * Formatting currencies
 * ====================
 */

${#numbers.formatCurrency(num)}
${#numbers.arrayFormatCurrency(numArray)}
${#numbers.listFormatCurrency(numList)}
${#numbers.setFormatCurrency(numSet)}


/*
 * ======================
 * Formatting percentages
 * ======================
 */

${#numbers.formatPercent(num)}
${#numbers.arrayFormatPercent(numArray)}
${#numbers.listFormatPercent(numList)}
${#numbers.setFormatPercent(numSet)}

/*
 * Set minimum integer digits and (exact) decimal digits.
```

```
 */
${#numbers.formatPercent(num, 3, 2)}
${#numbers.arrayFormatPercent(numArray, 3, 2)}
${#numbers.listFormatPercent(numList, 3, 2)}
${#numbers.setFormatPercent(numSet, 3, 2)}


/*
 * ===============
 * Utility methods
 * ===============
 */

/*
 * Create a sequence (array) of integer numbers going
 * from x to y
 */
${#numbers.sequence(from,to)}
${#numbers.sequence(from,to,step)}
```

Strings

- **#strings** : utility methods for `String` objects:

```
/*
 * =========================================================================
 * 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
 */
${#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

- **#objects** : utility methods for objects in general

```
/*
 * ========================================================================
 * See javadoc API for class org.thymeleaf.expression.Objects
 * ========================================================================
 */

/*
 * Return obj if it is not null, and default otherwise
 * Also works with arrays, lists or sets
 */
${#objects.nullSafe(obj,default)}
${#objects.arrayNullSafe(objArray,default)}
${#objects.listNullSafe(objList,default)}
${#objects.setNullSafe(objSet,default)}
```

Booleans

- **#bools** : utility methods for boolean evaluation

```
/*
 * ========================================================================
 * See javadoc API for class org.thymeleaf.expression.Bools
 * ========================================================================
 */

/*
 * Evaluate a condition in the same way that it would be evaluated in a th:if tag
 * (see conditional evaluation chapter afterwards).
 * Also works with arrays, lists or sets
 */
${#bools.isTrue(obj)}
${#bools.arrayIsTrue(objArray)}
${#bools.listIsTrue(objList)}
${#bools.setIsTrue(objSet)}

/*
```

```
 * Evaluate with negation
 * Also works with arrays, lists or sets
 */
${#bools.isFalse(cond)}
${#bools.arrayIsFalse(condArray)}
${#bools.listIsFalse(condList)}
${#bools.setIsFalse(condSet)}

/*
 * Evaluate and apply AND operator
 * Receive an array, a list or a set as parameter
 */
${#bools.arrayAnd(condArray)}
${#bools.listAnd(condList)}
${#bools.setAnd(condSet)}

/*
 * Evaluate and apply OR operator
 * Receive an array, a list or a set as parameter
 */
${#bools.arrayOr(condArray)}
${#bools.listOr(condList)}
${#bools.setOr(condSet)}
```

Arrays

- **#arrays** : utility methods for arrays

```
/*
 * ======================================================================
 * See javadoc API for class org.thymeleaf.expression.Arrays
 * ======================================================================
 */

/*
 * Converts to array, trying to infer array component class.
 * Note that if resulting array is empty, or if the elements
 * of the target object are not all of the same class,
 * this method will return Object[].
 */
${#arrays.toArray(object)}

/*
 * Convert to arrays of the specified component class.
 */
${#arrays.toStringArray(object)}
${#arrays.toIntegerArray(object)}
${#arrays.toLongArray(object)}
${#arrays.toDoubleArray(object)}
```

```
${#arrays.toFloatArray(object)}
${#arrays.toBooleanArray(object)}

/*
 * Compute length
 */
${#arrays.length(array)}

/*
 * Check whether array is empty
 */
${#arrays.isEmpty(array)}

/*
 * Check if element or elements are contained in array
 */
${#arrays.contains(array, element)}
${#arrays.containsAll(array, elements)}
```

Lists

- **#lists** : utility methods for lists

```
/*
 * ========================================================================
 * See javadoc API for class org.thymeleaf.expression.Lists
 * ========================================================================
 */

/*
 * Converts to list
 */
${#lists.toList(object)}

/*
 * Compute size
 */
${#lists.size(list)}

/*
 * Check whether list is empty
 */
${#lists.isEmpty(list)}

/*
 * Check if element or elements are contained in list
 */
${#lists.contains(list, element)}
${#lists.containsAll(list, elements)}
```

```
/*
 * Sort a copy of the given list. The members of the list must implement
 * comparable or you must define a comparator.
 */
${#lists.sort(list)}
${#lists.sort(list, comparator)}
```

Sets

- **#sets** : utility methods for sets

```
/*
 * =========================================================================
 * See javadoc API for class org.thymeleaf.expression.Sets
 * =========================================================================
 */

/*
 * Converts to set
 */
${#sets.toSet(object)}

/*
 * Compute size
 */
${#sets.size(set)}

/*
 * Check whether set is empty
 */
${#sets.isEmpty(set)}

/*
 * Check if element or elements are contained in set
 */
${#sets.contains(set, element)}
${#sets.containsAll(set, elements)}
```

Maps

- **#maps** : utility methods for maps

```
/*
 * =========================================================================
 * See javadoc API for class org.thymeleaf.expression.Maps
 * =========================================================================
 */
```

```
/*
 * Compute size
 */
${#maps.size(map)}

/*
 * Check whether map is empty
 */
${#maps.isEmpty(map)}

/*
 * Check if key/s or value/s are contained in maps
 */
${#maps.containsKey(map, key)}
${#maps.containsAllKeys(map, keys)}
${#maps.containsValue(map, value)}
${#maps.containsAllValues(map, value)}
```

Aggregates

- **#aggregates** : utility methods for creating aggregates on arrays or collections

```
/*
 * ========================================================================
 * See javadoc API for class org.thymeleaf.expression.Aggregates
 * ========================================================================
 */

/*
 * Compute sum. Returns null if array or collection is empty
 */
${#aggregates.sum(array)}
${#aggregates.sum(collection)}

/*
 * Compute average. Returns null if array or collection is empty
 */
${#aggregates.avg(array)}
${#aggregates.avg(collection)}
```

IDs

- **#ids** : utility methods for dealing with `id` attributes that might be repeated (for example, as a result of an iteration).

```
/*
 * ========================================================================
 * See javadoc API for class org.thymeleaf.expression.Ids
 * ========================================================================
```

```
 */

/*
 * Normally used in th:id attributes, for appending a counter to the id attribute
value
 * so that it remains unique even when involved in an iteration process.
 */
${#ids.seq('someId')}

/*
 * Normally used in th:for attributes in <label> tags, so that these labels can
refer to Ids
 * generated by means if the #ids.seq(...) function.
 *
 * Depending on whether the <label> goes before or after the element with the
#ids.seq(...)
 * function, the "next" (label goes before "seq") or the "prev" function (label
goes after
 * "seq") function should be called.
 */
${#ids.next('someId')}
${#ids.prev('someId')}
```

第七章：附录C：标记选择语法

Thymeleaf's Markup Selectors are directly borrowed from Thymeleaf's parsing library: AttoParser.

The syntax for this selectors has large similarities with that of selectors in XPath, CSS and jQuery, which makes them easy to use for most users. You can have a look at the complete syntax reference at the AttoParser documentation.

For example, the following selector will select every `<div>` with the class `content`, in every position inside the markup (note this is not as concise as it could be, read on to know why):

```
<div th:insert="mytemplate :: //div[@class='content']">...</div>
```

The basic syntax includes:

- `/x` means direct children of the current node with name x.
- `//x` means children of the current node with name x, at any depth.
- `x[@z="v"]` means elements with name x and an attribute called z with value "v".
- `x[@z1="v1" and @z2="v2"]` means elements with name x and attributes z1 and z2 with values "v1" and "v2", respectively.
- `x[i]` means element with name x positioned in number i among its siblings.
- `x[@z="v"][i]` means elements with name x, attribute z with value "v" and positioned in number i among its siblings that also match this condition.

But more concise syntax can also be used:

- `x` is exactly equivalent to `//x` (search an element with name or reference `x` at any depth level, a *reference*being a `th:ref` or a `th:fragment` attribute).
- Selectors are also allowed without element name/reference, as long as they include a specification of arguments. So `[@class='oneclass']` is a valid selector that looks for any elements (tags) with a class attribute with value `"oneclass"`.

Advanced attribute selection features:

- Besides `=` (equal), other comparison operators are also valid: `!=` (not equal), `^=` (starts with) and `$=` (ends with). For example: `x[@class^='section']` means elements with name `x` and a value for attribute `class` that starts with `section`.
- Attributes can be specified both starting with `@` (XPath-style) and without (jQuery-style). So `x[z='v']` is equivalent to `x[@z='v']`.
- Multiple-attribute modifiers can be joined both with `and` (XPath-style) and also by chaining multiple modifiers (jQuery-style). So `x[@z1='v1' and @z2='v2']` is actually equivalent to `x[@z1='v1'][@z2='v2']` (and also to `x[z1='v1'][z2='v2']`).

Direct *jQuery-like* selectors:

- `x.oneclass` is equivalent to `x[class='oneclass']`.
- `.oneclass` is equivalent to `[class='oneclass']`.
- `x#oneid` is equivalent to `x[id='oneid']`.
- `#oneid` is equivalent to `[id='oneid']`.
- `x%oneref` means `<x>` tags that have a `th:ref="oneref"` or `th:fragment="oneref"` attribute.
- `%oneref` means any tags that have a `th:ref="oneref"` or `th:fragment="oneref"` attribute. Note this is actually equivalent to simply `oneref` because references can be used instead of element names.
- Direct selectors and attribute selectors can be mixed: `a.external[@href^='https']`.

So the above Markup Selector expression:

```
<div th:insert="mytemplate :: //div[@class='content']">...</div>
```

Could be written as:

```
<div th:insert="mytemplate :: div.content">...</div>
```

Examining a different example, this:

```
<div th:replace="mytemplate :: myfrag">...</div>
```

Will look for a `th:fragment="myfrag"` fragment signature (or `th:ref` references). But would also look for tags with name `myfrag` if they existed (which they don't, in HTML). Note the difference with:

```
<div th:replace="mytemplate :: .myfrag">...</div>
```

...which will actually look for any elements with `class="myfrag"`, without caring about `th:fragment` signatures (or `th:ref` references).

Multivalued class matching

Markup Selectors understand the class attribute to be **multivalued**, and therefore allow the application of selectors on this attribute even if the element has several class values.

For example, `div.two` will match `<div class="one two three" />`

第八章：基于SpringBoot+Thymeleaf的热部署

具体配置如下：

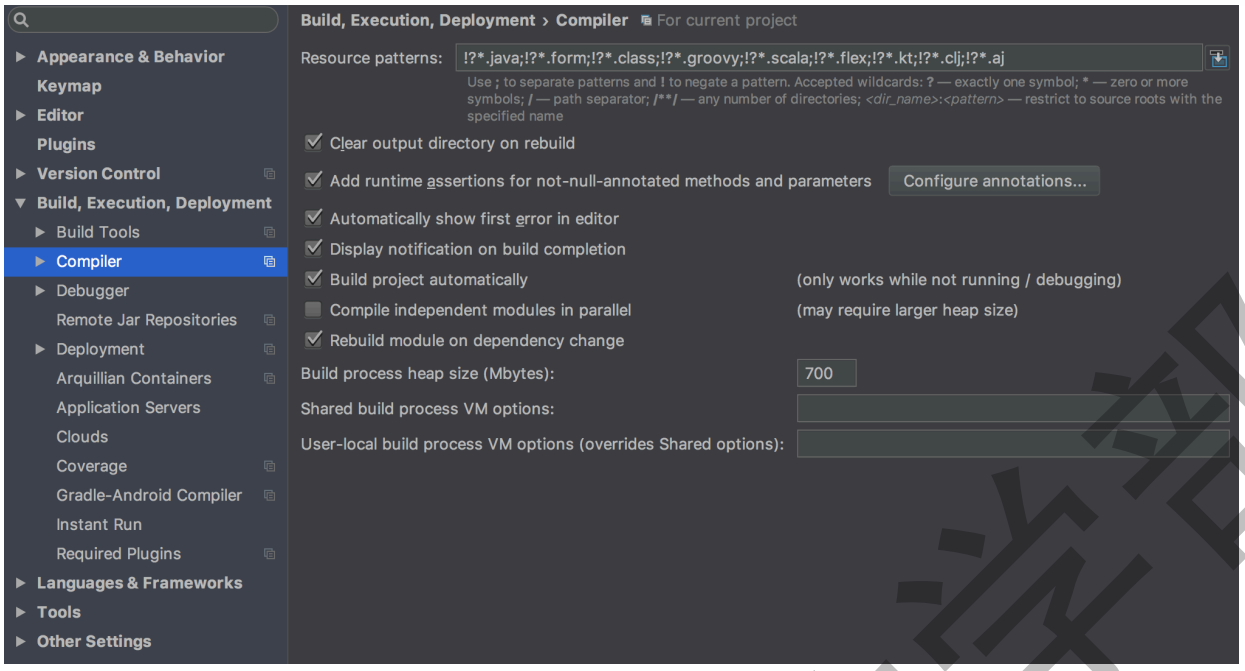
pom.xml 添加如下

```
<dependency>
            <groupId>org.springframework.boot</groupId>
            <artifactId>spring-boot-devtools</artifactId>
            <optional>true</optional>
</dependency>
```

```
<build>
   <plugins>
      <plugin>
         <groupId>org.springframework.boot</groupId>
         <artifactId>spring-boot-maven-plugin</artifactId>
         <configuration>
            <fork>true</fork>
         </configuration>
      </plugin>
   </plugins>
</build>
```

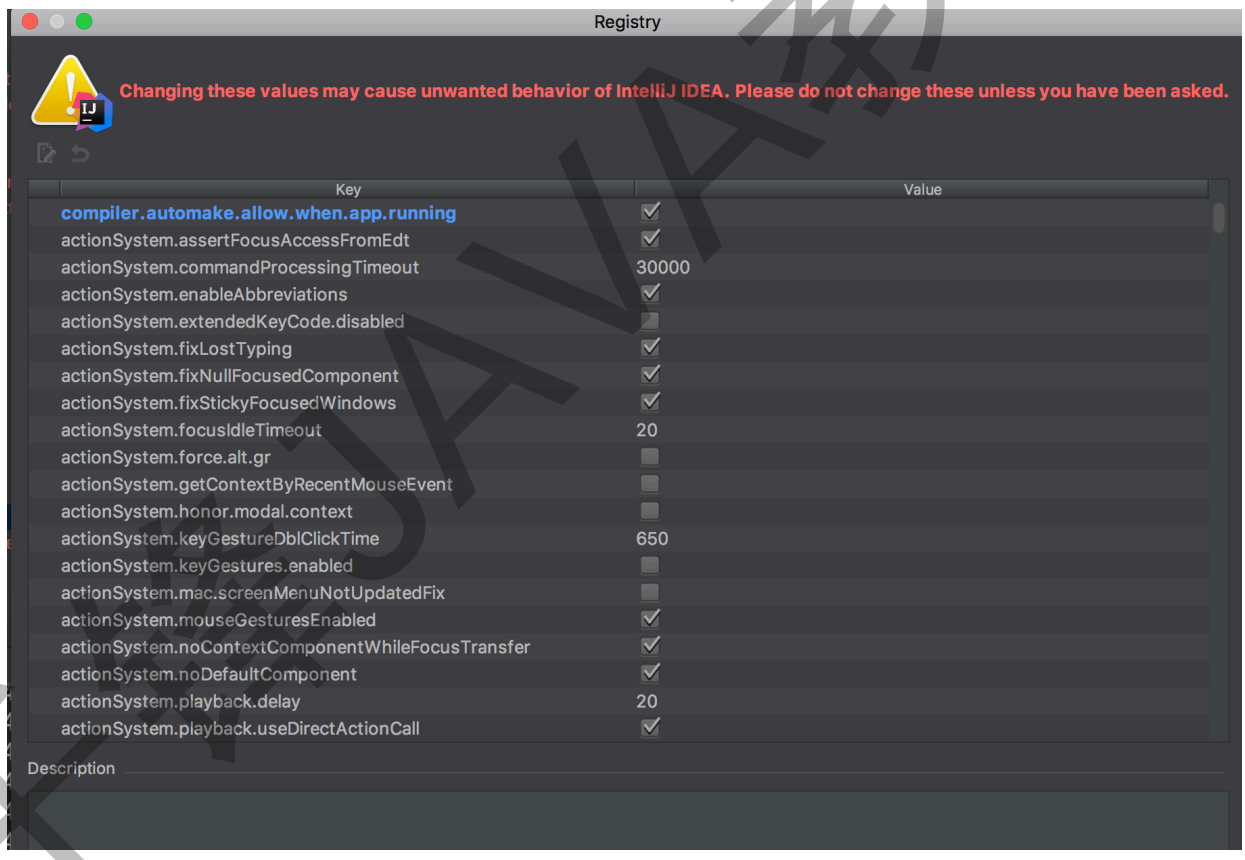
修改IntelliJ IDE的配置

选中自动编译功能



修改自动编译功能

按住：option+control+shift+/



修改Thymeleaf缓存属性

```
spring.thymeleaf.cache=false
```