ThymeLeaf 简介 - 与 Spring Boot 集成

韩博孙

2020 年 11 月 5 日 麻省理工学院

使用 Thymeleaf 与 Spring Boot Web 应用程序集成的基础知识，以及 Thymeleaf 标记的四种基本用法

本教程讨论如何设置一个包含 ThymeLeaf 库的 Spring Boot 应用程序，如何创建可重用的部分并在实际页面上使用它们。本教程还讨论了可以解决最常见设计问题的三个基本属性。

下载源 - 398.4 KB

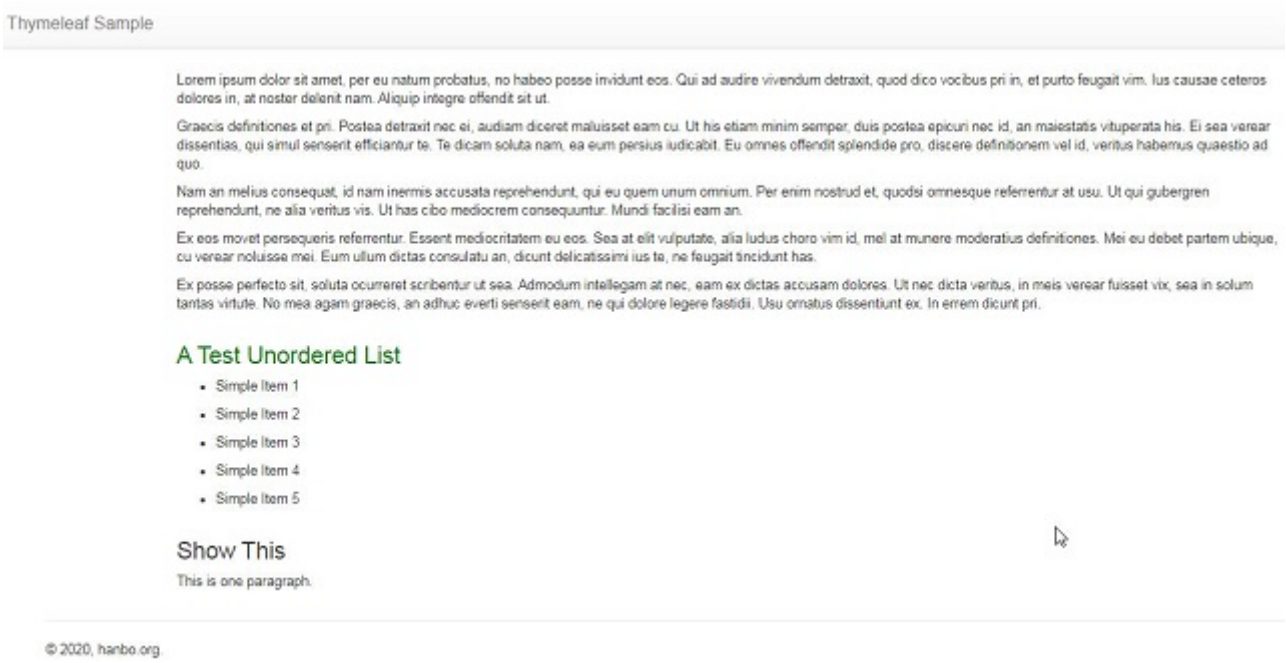
介绍

之前自己做的一个大项目，我用的是Spring Boot，把应用打包成war文件，这样就可以使用`TagLib`. 该项目是成功的。有一个未解决的问题。在`TagLib`Java的9或以上下运行时jar文件抛出异常。没有解决这个问题的替代罐子。我喜欢`TagLib`HTML 片段的模板和模块化。如果这个罐子在路上给我带来麻烦，我需要有一个解决方案。

解决方案是 ThymeLeaf。它出现在 201x 的某个时候。我不记得什么时候了。我不在乎学习，因为我可以使用`TagLib`我需要做的所有设计。对于任何框架，我都可以使用其 10% 的能力来解决手头所有问题的 95%。这是一个方便的技能。现在这`TagLib`将是一个问题，是时候切换到 ThymeLeaf 了。这是进化的一部分。

在本教程中，我想讨论如何设置一个包含 ThymeLeaf 库的 Spring Boot 应用程序，如何创建可重用的部分并在实际页面上使用它们。本教程还将讨论可以解决最常见设计问题的三个基本属性。

这是页面正确呈现时的屏幕截图：



您所看到的所有标题栏、四个段落、无序列表、名为“ **Show This** ”的部分和页脚（以及页面末尾的 JavaScript 文件）都是来自另一个文件的组件。如何重用组件是本教程最大的部分。在我们继续之前，让我们看一下 Maven POM 文件。

Maven POM 文件

示例项目的 Maven POM 文件是标准的 Spring Boot 项目，使用 Spring Boot starter 父项作为其父项。

要编译和打包应用程序，Maven POM 文件应具有以下两个依赖项：

XML 复制代码

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

第一个依赖项是用于 Web 应用程序的 Spring Boot starter。它负责构建简单 Web 应用程序的大部分引导程序。

第二个依赖项是 ThymeLeaf 与 Spring Boot 的集成。它负责配置应用程序中的 ThymeLeaf 页面渲染。

使用这两个 jar，设置 Spring 配置以便应用程序可以运行的困难部分全部解决了。它为程序员节省了大量时间。

项目文件夹结构

对于这个项目，我使用了与以前相同的项目文件结构。Java 文件位于文件夹*src/main/java/*下的子文件夹中。HTML 页面模板文件位于*src/main/resources/templates 中*。ThymeLeaf 模板使用的片段位于*src/main/resources/templates/parts 中*。并且静态文件如 CSS 和 JavaScript 文件位于*src/main/resources/static/ 的*子文件夹中。

这些文件夹位置基于默认配置。我确信我可以在*application.properties*文件中更改这些位置。这些可以通过网上搜索找到。

Java代码

为了运行这个示例应用程序，只需要两个 Java 文件。一个是主条目*App.java*。另一个文件是名为*IndexController.java*的控制器类文件。

主入口文件如下所示：

爪哇 复制代码

```
package org.hanbo.spring.sample;

import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;

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

这是使用 Spring Boot 启动 Spring Web 应用程序的典型方式。为了让应用程序的功能更加丰富，我可以添加更多的注释，但是对于一个简单的示例应用程序，这足以让应用程序运行起来。

另一个文件是控制器类。里面，有两个方法，每个方法都用来处理HTTP请求：

爪哇 缩小▲ 复制代码

```
package org.hanbo.spring.sample.controllers;

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

import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;

@Controller
public class IndexController
{
   @RequestMapping(value="/", method=RequestMethod.GET)
   public String home()
   {
      final String retVal = "redirect:/mixedup";

      return retVal;
   }

   @RequestMapping(value="/mixedup", method=RequestMethod.GET)
   public ModelAndView mixedup()
   {
      List<String> items = new ArrayList<String>();
      items.add("Simple Item 1");
      items.add("Simple Item 2");
      items.add("Simple Item 3");
      items.add("Simple Item 4");
      items.add("Simple Item 5");

      ModelAndView retVal = new ModelAndView();
      retVal.setViewName("mixedup");
      retVal.addObject("siteName", "Thymeleaf Sample");
      retVal.addObject("listTitle",
         "<span style=\"color: green;\">A Test Unordered List</span>");
      retVal.addObject("items", items);
      retVal.addObject("conditionVal", 5);

      return retVal;
```

```
    }
}
```

第二个请求处理对GETURL的 HTTP请求：http://localhost:8080/mixedup。

第二种方法返回 的对象ModelAndView。该对象具有视图的名称，即模板页面文件（包含所有占位符）。该对象还具有数据模型，即可以设置为页面模板上占位符的值的对象的哈希图。

该请求将触发使用 ThymeLeaf 标记创建的页面模板。正如介绍中所讨论的，我创建了这个示例应用程序来练习一些简单的方面，这应该可以解决我将面临的 95% 的设计问题。这些简单的方面包括：

- 如何使用 Java 方法返回的模型数据为页面上的元素设置 HTML 文本
- 如何遍历一组项目并将它们显示在页面上
- 如何有条件地显示元素
- 如何将另一个文件中的 HTML 组件添加到页面中，并使用 Java 代码中的数据模型来设置元素文本。最后一部分是对页面的一部分进行切割和切片，将它们定义为组件，以便它们可以在不同的页面中使用。

接下来将讨论模板页面如何处理数据模型中的值。

ThymeLeaf 页面模板

对于这个示例应用程序，我只有一页。它被称为*mixup.html*，位于*src/main/ resouces /templates.html*。此页面由几个不同的部分构成。并且这些部件或组件具有由控制器类中的方法呈现的值的占位符。

该页面的完整源代码如下所示：

HTML 缩小▲ 复制代码

```
<!DOCTYPE HTML>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">

    <title>HanBo-ORG Mixedup Page</title>
    <link rel="stylesheet" th:href="@{/bootstrap/css/bootstrap.min.css}"/>
    <link rel="stylesheet" th:href="@{/bootstrap/css/bootstrap-theme.min.css}"/>
    <link rel="stylesheet" th:href="@{/css/index.css}"/>
</head>
<body>
   <div th:replace="components/parts::header">
   </div>

   <div class="container">
      <div th:insert="components/parts::info">
      </div>
      <div th:replace="components/parts::looptest">
      </div>
      <div th:if="${conditionVal == 5}">
         <div th:replace="components/parts::showThis"></div>
      </div>
      <div th:unless="${conditionVal == 5}">
         <div th:replace="components/parts::showThat"></div>
      </div>
   </div>

   <div th:replace="components/parts::footer"></div>

   <div th:replace="components/parts::stdjs"></div>
 </body>
</html>
```

让我们从顶部开始。对于任何渲染引擎，都需要导入渲染引擎的命名空间，以便在渲染处理过程中找到特殊标签。对于 ThymeLeaf，它是这样完成的：

HTML 复制代码

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

然后，在该`head`部分，它是第一个使用 ThymeLeaf 标记的地方：

HTML 复制代码

```
<link rel="stylesheet" th:href="@{/bootstrap/css/bootstrap.min.css}"/>
<link rel="stylesheet" th:href="@{/bootstrap/css/bootstrap-theme.min.css}"/>
<link rel="stylesheet" th:href="@{/css/index.css}"/>
```

在这种情况下，我使用了属性`th:href`。我将`string`常量作为 URL传递给这三行。

接下来，我使用在另一个文件中定义的部件插入或放置在此页面文件中定义的位置，如下所示：

HTML 复制代码

```
<div th:replace="components/parts::header">
</div>
```

还有一些这样的用法：

HTML 复制代码

```
      <div th:replace="components/parts::looptest">
      </div>
      <div th:if="${conditionVal == 5}">
         <div th:replace="components/parts::showThis"></div>
      </div>
      <div th:unless="${conditionVal == 5}">
         <div th:replace="components/parts::showThat"></div>
      </div>
...
   <div th:replace="components/parts::footer"></div>

   <div th:replace="components/parts::stdjs"></div>
```

从这些行中取出一个，该属性称为`th:replace`。也可以`th:insert`。还有 被`th:include`弃用的，很快就将不可用。

这些属性的值指定部件文件的位置以及此文件中的哪个代码片段将放置在此处。例如，值“`components/parts::looptest`”表示包含可重用部件的文件位于模板文件夹中的子文件夹“*components*”，文件名为“*parts.html*”。该`::`分离器在指定的*部分*文件，其中片段可以找到。在本例中，零件文件中的片段称为“`looptest`”。

该文件的最后一个重要部分是值的有条件放置，类似于`if ... else ...` 块。这是用 ThymeLeaf 完成的：

HTML 复制代码

```
<div th:if="${conditionVal == 5}">
   ...
</div>
<div th:unless="${conditionVal == 5}">
   ...
</div>
```

请注意，`else`块（属性`th:unless`）必须具有与 的条件检查相同的条件检查`if`。如果您使用不同的条件检查，显示将非常奇怪。

是时候看看碎片是什么样子了。这些片段在一个名为*parts.html 的*文件中定义。我们接下来会讨论这个。

百里香叶碎片

为了构建具有可重用部件的页面，部件必须放置在某处，以便在呈现页面时，可以从该公共位置提取这些部件并将其添加到页面中。ThymeLeaf 的做法不同。可重复使用的部分可以打包成一个文件。每个部分都有一个 ID。然后在构建的页面中，可以通过目标文件夹、部件文件和片段ID来引用这些片段。

请记住在上一节中，以传统方式替换、插入或包含的部件。它可能看起来像这样：

HTML 复制代码

```
<div th:replace="components/parts::header">
</div>
```

属性`th:replace`“ `components/parts::header`”中的值。值“ *components/parts* ”是定义片段的文件。如果您假设该文件名为“ *components/parts.html* ”，那么您是对的。查找文件的基本路径是“*资源/模板*”。所以相对完整路径是“ *resources/templates/components/parts.html* ”。该文件如下所示：

HTML 缩小▲ 复制代码

```
<!DOCTYPE HTML>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
   <head>
      <meta charset="UTF-8"/>
      <title>Spring Boot Thymeleaf Application - Fragments</title>
   </head>
   <body>
      <div th:fragment="header">
    <nav class="navbar navbar-default navbar-fixed-top">
      <div class="container-fluid">
        <div class="navbar-header">
          <button type="button" class="navbar-toggle collapsed" data-toggle="collapse"
          data-target="#navbar" aria-expanded="false" aria-controls="navbar">
            <span class="sr-only">Toggle navigation</span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
            <span class="icon-bar"></span>
          </button>
          <a class="navbar-brand" th:href="@{/}" th:text="${siteName}"></a>
        </div>
        <div id="navbar" class="navbar-collapse collapse">
        </div>
      </div>
    </nav>
      </div>

      <div class="row page-start" th:fragment="info">
         <div class="col-xs-12">
            <p>
            Lorem ipsum dolor sit amet, per eu natum probatus, no habeo posse invidunt eos.
            Qui ad audire vivendum detraxit, quod dico vocibus pri in, et purto feugait vim.
            Ius causae ceteros dolores in, at noster delenit nam.
            Aliquip integre offendit sit ut.
            </p>
            <p>
            Graecis definitiones et pri.
            Postea detraxit nec ei, audiam diceret maluisset eam cu.
            Ut his etiam minim semper, duis postea epicuri nec id,
            an maiestatis vituperata his.
            Ei sea verear dissentias, qui simul senserit efficiantur te.
            Te dicam soluta nam,
            ea eum persius iudicabit. Eu omnes offendit splendide pro,
            discere definitionem vel id,
            veritus habemus quaestio ad quo.
            </p>
            <p>
            Nam an melius consequat, id nam inermis accusata reprehendunt,
            qui eu quem unum omnium.
            Per enim nostrud et, quodsi omnesque referrentur at usu.
            Ut qui gubergren reprehendunt,
            ne alia veritus vis. Ut has cibo mediocrem consequuntur. Mundi facilisi eam an.
            </p>
            <p>
            Ex eos movet persequeris referrentur. Essent mediocritatem eu eos.
            Sea at elit vulputate, alia ludus choro vim id,
            mel at munere moderatius definitiones. Mei eu debet partem ubique,
```

```
            cu verear noluisse mei. Eum ullum dictas consulatu an,
            dicunt delicatissimi ius te, ne feugait tincidunt has.
            </p>
            <p>
            Ex posse perfecto sit, soluta ocurreret scribentur ut sea.
            Admodum intellegam at nec, eam ex dictas accusam dolores.
            Ut nec dicta veritus, in meis verear fuisset vix,
            sea in solum tantas virtute. No mea agam graecis, an adhuc everti senserit eam,
            ne qui dolore legere fastidii. Usu ornatus dissentiunt ex. In errem dicunt pri.
            </p>
         </div>
      </div>

      <div class="row" th:fragment="looptest">
         <div class="col-xs-12">
            <h3 th:utext="${listTitle}"></h3>
            <ul th:each="item: ${items}">
               <li th:text="${item}"></li>
            </ul>
         </div>
      </div>

      <div class="row" th:fragment="showThis">
         <div class="col-xs-12">
            <h3>Show This</h3>
            <p>This is one paragraph.</p>
         </div>
      </div>

      <div class="row" th:fragment="showThat">
         <div class="col-xs-12">
            <h3>Show This</h3>
            <p>This is another paragraph.</p>
         </div>
      </div>

      <div class="container-fluid" th:fragment="footer">
         <div class="row footer">
            <div class="col-xs-12">
               <hr/>
               &copy 2020, hanbo.org.
            </div>
         </div>
      </div>

      <div th:fragment="stdjs">
   <script type="text/javascript" th:src="@{/jquery/js/jquery.min.sj}"></script>
   <script type="text/javascript" th:src="@{/bootstrap/js/bootstrap.min.sj}"></script>
      </div>
   </body>
</html>
```

在此文件中，片段定义为：

HTML 复制代码

```
<div class="row" th:fragment="showThat">
...
</div>
```

该属性`th:fragment`设置片段的 ID。此 ID 是在范围分隔符“`::`”之后引用的 ID 。当这个片段被渲染到实际页面时，整个`div`元素及其所有子元素都被渲染到页面。然后整个页面的占位符将被设置为来自`ModelAndView`控制器方法返回的对象的实际值。

下一个问题是，我们如何为占位符设置值？在这个例子中，有四种不同的方式：

- 为 HTML 元素放置文本
- 为 HTML 元素放置未转义的文本
- 遍历元素集合并将它们呈现为文本

- 和if-else条件块，正如我们在上一节中看到的那样

另一个值得学习的概念是switch-case块，它很容易，所以我不会在这里介绍。

首先，如何从ModelAndView呈现的页面显示基于文本的值。文本string定义如下：

爪哇 复制代码

```
...

ModelAndView retVal = new ModelAndView();
...
retVal.addObject("siteName", "Thymeleaf Sample");
```

将此文本呈现为 HTML 元素是这样完成的：

HTML 复制代码

```
...
<a class="navbar-brand" ... th:text="${siteName}"></a>
...
```

除了显示纯文本外，显示 HTML 文本并呈现它也很有用。这很容易完成，如下所示。假设 html 的 Java 代码string定义如下：

爪哇 复制代码

```
...

ModelAndView retVal = new ModelAndView();
...
retVal.addObject("listTitle",
   "<span style=\"color: green;\">A Test Unordered List</span>");
...
```

要呈现 HTML string，方法如下：

HTML 复制代码

```
...
<h3 th:utext="${listTitle}"></h3>
...
```

在这两种情况下，我都使用“ ${variableName}”来引用存储在ModelAndView对象内部的对象/变量。这些是" hashmap" inModelAndView对象中实际值的键。

呈现string列表或其他可迭代集合中的值列表。以下是这是如何完成的。假设定义列表对象的 Java 代码如下：

爪哇 复制代码

```
List<String> items = new ArrayList<String>();
items.add("Simple Item 1");
items.add("Simple Item 2");
items.add("Simple Item 3");
items.add("Simple Item 4");
items.add("Simple Item 5");

ModelAndView retVal = new ModelAndView();
...
retVal.addObject("items", items);
...
```

然后列表中这些项目的渲染是这样完成的：

复制代码

```
...
<div class="col-xs-12">
   ...
   <ul th:each="item: ${items}">
      <li th:text="${item}"></li>
   </ul>
</div>
...
```

这与单值文本的显示略有不同。首先，您必须获得集合的迭代器。这是通过属性完成的`th:each`，然后像在 for 循环中一样使用迭代器变量，通过“`${item}`”获取实际值。

我们已经看到`if-else`语句的使用。让我们再讲一遍。假设变量和值的 Java 代码设置如下：

爪哇 复制代码

```
ModelAndView retVal = new ModelAndView();
...
retVal.addObject("conditionVal", 5);
```

方式`if-else`工作是这样的：

HTML 复制代码

```
<div th:if="${conditionVal == 5}">
   ...
</div>
<div th:unless="${conditionVal == 5}">
   ...
</div>
```

第一部分很容易理解。如果变量“`conditionVal`”等于`5`. 第二个有点难以理解。它使用`th:unless`. 它基本上说，除非变量“`conditionVal`”等于`5`，否则会显示，如果等于`5`，则不会显示。正如我之前所说，在这两种情况下，要检查的值必须相同。或者渲染会显示一些非常奇怪的东西。

如何运行此示例应用程序

要构建此示例应用程序，请使用命令提示符并转到*POM.xml*文件所在的文件夹。然后运行以下命令：

复制代码

```
mvn clean install
```

一旦构建成功（它会成功，因为它是一个简单的应用程序），运行以下命令。它将启动 Web 应用程序：

复制代码

```
java -jar target\hanbo-thymeleaf-sample-1.0.1.jar
```

Web 应用程序将成功启动。之后，使用网络浏览器访问以下 URL：

复制代码

```
http://localhost:8080/
```

然后您将看到本教程开头的屏幕截图。这是本教程的结论。

概括

在本教程中，我讨论了如何将 ThymeLeaf 页面渲染引擎与 Spring Boot 支持的 Web 应用程序集成。我的主要目标是找到我正在使用的现有渲染引擎的替代方案。这也是学习新事物的机会。我对这个新渲染引擎的印象是：

- 将 ThymeLeaf 与基于 Spring Boot 的 Web 应用程序集成非常容易。
- 自定义配置很容易，因此您无需依赖默认配置。
- 渲染规则很容易掌握。
- 由于它的用户友好性，我使用 ThymeLeaf 做任何我以前做过的事情都没有问题。

我不确定的是与 Spring Security 的集成。Spring Security 使用自己的命名空间作为标签，而 ThymeLeaf 使用自己的命名空间。两个导入命名空间的方式不同。我有点不确定这种差异以及它是否会产生任何问题。我相信不应该有任何，因为许多项目都很好地使用了两者。稍后我将为此创建另一个教程。

我匆忙创建了本教程。这不是我最好的作品。完成了。我希望你喜欢它。

历史

- 10/11/2020 - 初稿

执照

分享

关于作者

韩博孙

组长 The Judge Group

美国

手表该会员

没有提供传记

评论和讨论

添加评论或问题 | 电子邮件提醒 | Search Comments

-- 本论坛暂无消息 --