<http://blog.csdn.net/u013096666/article/details/53380600>

转载：http://blog.csdn.net/u014695188/article/details/52347318

整体步骤：  
（1）            在pom.xml中引入thymeleaf;  
（2）            如何关闭thymeleaf缓存  
（3）            编写模板文件.html

[**spring**](http://lib.csdn.net/base/javaee) Boot默认就是使用thymeleaf模板引擎的，所以只需要在pom.xml加入依赖即可：

**[html]** [view plain](http://blog.csdn.net/u014695188/article/details/52347318) [copy](http://blog.csdn.net/u014695188/article/details/52347318)

[在CODE上查看代码片](https://code.csdn.net/snippets/1855154)

1. **<dependency>**
2. **<groupId>**org.springframework.boot**</groupId>**
4. **<artifactId>**spring-boot-starter-thymeleaf**</artifactId>**
5. **</dependency>**

Thymeleaf缓存在开发过程中，肯定是不行的，那么就要在开发的时候把缓存关闭，只需要在application.properties进行配置即可：

**[html]** [view plain](http://blog.csdn.net/u014695188/article/details/52347318) [copy](http://blog.csdn.net/u014695188/article/details/52347318)

[在CODE上查看代码片](https://code.csdn.net/snippets/1855154)

1. ########################################################
2. ###THYMELEAF (ThymeleafAutoConfiguration)
3. ########################################################
4. #spring.thymeleaf.prefix=classpath:/templates/
5. #spring.thymeleaf.suffix=.html
6. #spring.thymeleaf.mode=HTML5
7. #spring.thymeleaf.encoding=UTF-8
8. # ;charset=**<encoding>** is added
9. #spring.thymeleaf.content-type=text/html
10. # set to false for hot refresh
12. spring.thymeleaf.cache=false

编写模板文件src/main/resouces/templates/helloHtml.html

**[html]** [view plain](http://blog.csdn.net/u014695188/article/details/52347318) [copy](http://blog.csdn.net/u014695188/article/details/52347318)

[在CODE上查看代码片](https://code.csdn.net/snippets/1855154)

1. <!DOCTYPE html**>**
2. **<html** xmlns="http://www.w3.org/1999/xhtml" xmlns:th="http://www.thymeleaf.org"
3. xmlns:sec="http://www.thymeleaf.org/thymeleaf-extras-springsecurity3"**>**
4. **<head>**
5. **<title>**Hello World!**</title>**
6. **</head>**
7. **<body>**
8. **<h1** th:inline="text"**>**Hello.v.2**</h1>**
9. **<p** th:text="${hello}"**></p>**
10. **</body>**
11. **</html>**

编写访问路径(com.kfit.test.web.TemplateController):

**[html]** [view plain](http://blog.csdn.net/u014695188/article/details/52347318) [copy](http://blog.csdn.net/u014695188/article/details/52347318)

[在CODE上查看代码片](https://code.csdn.net/snippets/1855154)

1. package com.kfit.test.web;
3. import java.util.Map;
4. import org.springframework.stereotype.Controller;
5. import org.springframework.web.bind.annotation.RequestMapping;


9. /\*\*
11. \* 模板测试.
13. \* @author Administrator
15. \*
17. \*/
19. @Controller
21. publicclass TemplateController {
22. /\*\*
24. \* 返回html模板.
26. \*/
28. @RequestMapping("/helloHtml")
29. public String helloHtml(Map**<String**,Object**>** map){
31. map.put("hello","from TemplateController.helloHtml");
32. return"/helloHtml";
33. }
34. }

启动应用，输入地址：http://127.0.0.1:8080/helloHtml 会输出：

Hello.v.2

from TemplateController.helloHtml

使用freemarker  
使用freemarker也很简单，  
在pom.xml加入freemarker的依赖：

**[html]** [view plain](http://blog.csdn.net/u014695188/article/details/52347318) [copy](http://blog.csdn.net/u014695188/article/details/52347318)

[在CODE上查看代码片](https://code.csdn.net/snippets/1855154)

1. **<dependency>**
2. **<groupId>**org.springframework.boot**</groupId>**
3. **<artifactId>**spring-boot-starter-freemarker**</artifactId>**
4. **</dependency>**

剩下的编码部分都是一样的，说下application.properties文件：

**[html]** [view plain](http://blog.csdn.net/u014695188/article/details/52347318) [copy](http://blog.csdn.net/u014695188/article/details/52347318)

[在CODE上查看代码片](https://code.csdn.net/snippets/1855154)

1. ########################################################
2. ###FREEMARKER (FreeMarkerAutoConfiguration)
3. ########################################################
4. spring.freemarker.allow-request-override=false
5. spring.freemarker.cache=true
6. spring.freemarker.check-template-location=true
7. spring.freemarker.charset=UTF-8
8. spring.freemarker.content-type=text/html
9. spring.freemarker.expose-request-attributes=false
10. spring.freemarker.expose-session-attributes=false
11. spring.freemarker.expose-spring-macro-helpers=false
12. #spring.freemarker.prefix=
13. #spring.freemarker.request-context-attribute=
14. #spring.freemarker.settings.\*=
15. #spring.freemarker.suffix=.ftl
16. #spring.freemarker.template-loader-path=classpath:/templates/#comma-separatedlist
17. #spring.freemarker.view-names= #whitelistofviewnamesthatcanberesolved

com.kfit.test.web.TemplateController：

**[html]** [view plain](http://blog.csdn.net/u014695188/article/details/52347318) [copy](http://blog.csdn.net/u014695188/article/details/52347318)

[在CODE上查看代码片](https://code.csdn.net/snippets/1855154)

1. /\*\*
2. \* 返回html模板.
3. \*/
5. @RequestMapping("/helloFtl")
6. public String helloFtl(Map**<String**,Object**>** map){
7. map.put("hello","from TemplateController.helloFtl");
8. return"/helloFtl";
9. }

访问地址：<http://127.0.0.1:8080/helloFtl>

Hello.v.2

from TemplateController.helloFtl

thymeleaf和freemarker是可以共存的。

------------------------------------------------------------------------------------------------------------------------------------------------

本文记录一下几点：   
一、资源文件的约定目录结构   
二、Maven配置   
三、开发时修改thymeleaf模板自动重新加载配置   
四、thymeleaf常用基础知识点

一、资源文件的约定目录结构

Maven的资源文件目录：/src/[**Java**](http://lib.csdn.net/base/javaee)/resources   
spring-boot项目静态文件目录：/src/java/resources/static   
spring-boot项目模板文件目录：/src/java/resources/templates   
spring-boot静态首页的支持，即index.html放在以下目录结构会直接映射到应用的根目录下：

**[html]** [view plain](http://blog.csdn.net/u014695188/article/details/52347318) [copy](http://blog.csdn.net/u014695188/article/details/52347318)

[在CODE上查看代码片](https://code.csdn.net/snippets/1855154)

1. classpath:/META-INF/resources/index.html
2. classpath:/resources/index.html
3. classpath:/static/index.html
4. calsspath:/public/index.html

由于使用thymeleaf的**[HTML5](http://lib.csdn.net/base/html5" \o "HTML5知识库" \t "_blank)**模板，所以我将index.html模板文件直接放到了/src/java/resources/templates目录下。然而这个目录并不是首页文件的默认目录，所以我们需要手动将应用根路径映射到/src/java/resources/templates/index.html下。这个在spring-mvc的Controller下映射一下就可以了。

**[html]** [view plain](http://blog.csdn.net/u014695188/article/details/52347318) [copy](http://blog.csdn.net/u014695188/article/details/52347318)

[在CODE上查看代码片](https://code.csdn.net/snippets/1855154)

1. @RequestMapping("/")
2. public String index(){
3. return "index";
4. }

在spring-boot下，默认约定了Controller试图跳转中thymeleaf模板文件的的前缀prefix是”classpath:/templates/”,后缀suffix是”.html”   
这个在application.properties配置文件中是可以修改的。   
如下配置可以修改试图跳转的前缀和后缀

**[html]** [view plain](http://blog.csdn.net/u014695188/article/details/52347318) [copy](http://blog.csdn.net/u014695188/article/details/52347318)

[在CODE上查看代码片](https://code.csdn.net/snippets/1855154)

1. spring.thymeleaf.prefix: /templates/
2. spring.thymeleaf.suffix: .html

更多有关thymeleaf中的默认这是可以查看org.springframework.boot.autoconfigure.thymeleaf.ThymeleafProperties这个类的属性 

二、Maven配置   
在pom.xml中加入如下依赖

**[html]** [view plain](http://blog.csdn.net/u014695188/article/details/52347318) [copy](http://blog.csdn.net/u014695188/article/details/52347318)

[在CODE上查看代码片](https://code.csdn.net/snippets/1855154)

1. **<dependency>**
2. **<groupId>**org.springframework.boot**</groupId>**
3. **<artifactId>**spring-boot-starter-thymeleaf**</artifactId>**
4. **</dependency>**

原来关于spring-boot-starter-web等的依赖就可以去掉了，因为spring-boot-starter-thymeleaf是包含这些依赖的。而关于jsp的依赖也可以去掉了，因为我们已经完全抛弃jsp了。   
  
三、开发时修改thymeleaf模板自动重新加载配置   
Spring-boot使用thymeleaf时默认是有缓存的，即你把一个页面代码改了不会刷新页面的效果，你必须重新运行spring-boot的main()方法才能看到页面更改的效果。我们可以把thymeleaf的缓存关掉，用于支持页面修改后重新发布到spring-boot内嵌的tomcat中去。在application.properties配置文件中加入以下配置。

**[html]** [view plain](http://blog.csdn.net/u014695188/article/details/52347318) [copy](http://blog.csdn.net/u014695188/article/details/52347318)

[在CODE上查看代码片](https://code.csdn.net/snippets/1855154)

1. # Allow Thymeleaf templates to be reloaded at dev time
2. spring.thymeleaf.cache: false
3. server.tomcat.access\_log\_enabled: true
4. server.tomcat.basedir: target/tomcat

四、thymeleaf常用基础知识点   
  
1、在html页面中引入thymeleaf命名空间，即<html xmlns:th=http://www.thymeleaf.org></html>，此时在html模板文件中动态的属性使用th:命名空间修饰   
  
2、引用静态资源文件，比如CSS和JS文件，语法格式为“@{}”，如@{/js/blog/blog.js}会引入/static目录下的/js/blog/blog.js文件   
  
3、访问spring-mvc中model的属性，语法格式为“${}”，如${user.id}可以获取model里的user对象的id属性   
  
4、循环，在html的标签中，加入th:each=“value:${list}”形式的属性，如<span th:each=”user:${users}”></span>可以迭代users的数据   
  
5、判断，在html标签中，加入th:if=”表达式”可以根据条件显示html元素   
<span th:if="${not #lists.isEmpty(blog.publishTime)}">   
<span id="publishtime" th:text="${#dates.format(blog.publishTime, 'yyyy-MM-dd HH:mm:ss')}"></span>   
</span>   
以上代码表示若blog.publishTime时间不为空，则显示时间   
  
6、时间的格式化，   
${#dates.format(blog.publishTime,'yyyy-MM-dd HH:mm:ss')}   
表示将时间格式化为”yyyy-MM-dd HH:mm:ss”格式化写法与Java格式化Date的写法是一致的。   
  
7、字符串拼接，有两种形式   
比如拼接这样一个URL:/blog/delete/{blogId}   
第一种：th:href="'/blog/delete/' + ${blog.id }"   
第二种：th:href="${'/blog/delete/' + blog.id }"