**Springboot**

# 定场

马瘦毛长蹄子肥 ，儿子偷爹不算贼 ，瞎大爷娶了瞎大奶奶 ，老两口过了多半辈， 谁也没看见谁。开头呢，给大家念这么一段定场诗，希望您各位洗耳恭听。

天也不早了，人也不少了，我们闲言少叙，书归正传。

从今天开始呢，给大家说一段长篇"评书"《Springboot》。说一说发生在Springboot身上那些有趣的事，您各位捧场。

# 第一回 Springboot入门

## 场景1

作为这段书的开始呢，首先我们得先弄清楚几个问题。

1. Springboot是什么。
2. Springboot有什么或者说Springboot可以做什么。
3. Springboot让大家青睐的原因是什么。

首先Springboot是什么。从它的名字，我相信大家肯定也猜到了，它一定和大家非常熟悉的Spring有关系。恭喜你，的确如此。我们知道，Spring框架这个大家族中，产生了很多衍生框架，如我们熟知的Spring、SpringMVC等等，Spring的核心内容在于控制反转和依赖注入，这两个特点并非是一种技术，而是一种设计思想。在操作中，Spring依靠配置文件去创建Bean以及实现复杂的对象关系。因此，我们在使用Spring框架的时候，需要书写大量的配置文件，像我们熟悉的数据库连接、日志输出、Mapper等等。哪怕我们的项目仅仅是实现一个很简单的发送邮件的功能。相信做过Spring的各位Coder，一定不会陌生。

这个时候，Springboot应运而生，它是一种全新的编程规范，它的产生大大简化了框架的使用，我们不再需要大量而繁琐的配置文件，它的设计目的就是为了让我们的项目尽快跑起来，让我们更加专注于我们项目本身的功能，让配置不再成为我们项目上的绊脚石。

其次Springboot可以做什么。了解了Springboot是什么以后，它的用途相信大家也很清楚了，就是来打通我们的业务逻辑。这涉及到Springboot几个重要的概念：service、controller、model、mapper、dao。

dao和mapper其实一个东西，是与数据库打交道的，对数据库进行数据持久化操作，他的方法语句是直接针对数据库操作的。

service服务是一个相对独立的功能模块，主要负责业务逻辑应用设计。首先也要设计接口，然后再设计其实现该接口的类。这样我们就可以在应用中调用service接口进行业务处理。service层业务实现，具体调用到已经定义的DAO的接口，封装service层的业务逻辑有利于通用的业务逻辑的独立性和重复利用性。

controller控制器负责具体业务模块流程的控制，会调用Service层的接口来控制业务逻辑。

model存放我们的实体类。

TODO 待补充

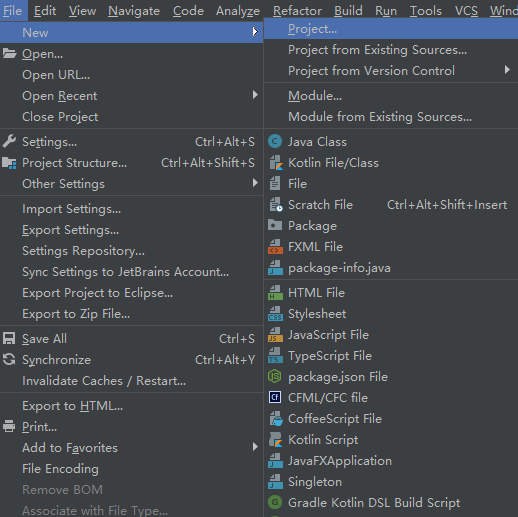
## 场景2

上一场景中，TODO

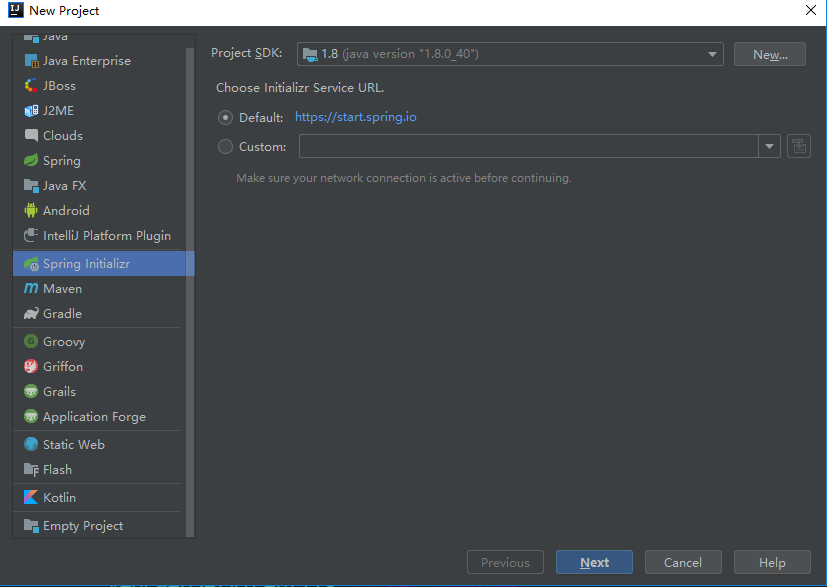
俗话说：口说无凭。因此，在这一场景中，我们新建一个基于Springboot的web项目，向大家展示一下Springboot的这些特点，让大家看看使用Springboot到底有多么简单。

新建一个Springboot web项目的步骤如下(我这里使用IDEA进行展示)：

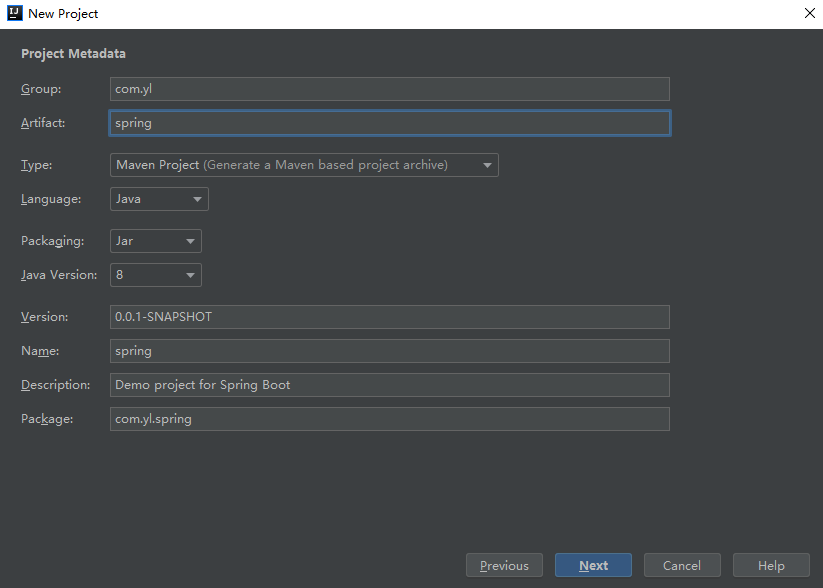
1. 选择菜单File 🡪 New 🡪 Project，如图：



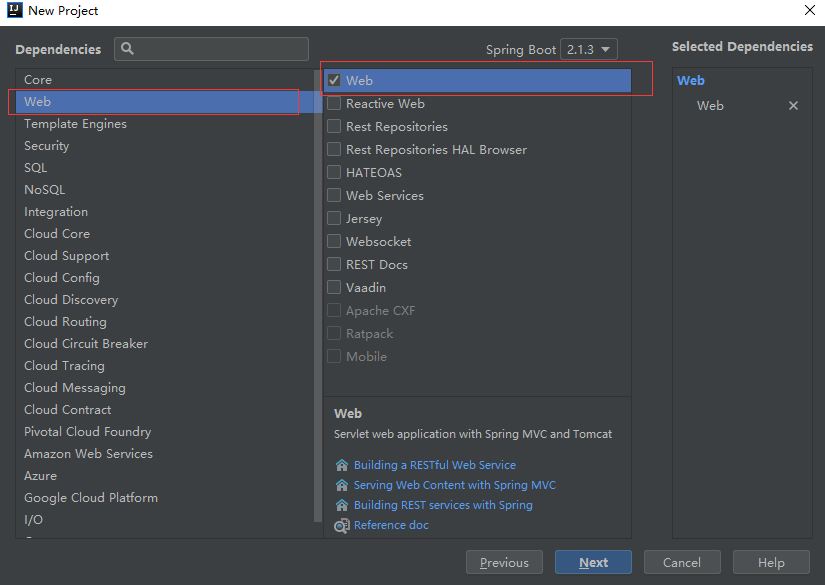
1. 在弹出的窗口中选择Spring Initializr，同时这里推荐使用JDK1.8，其余的选择和下图中一样即可。



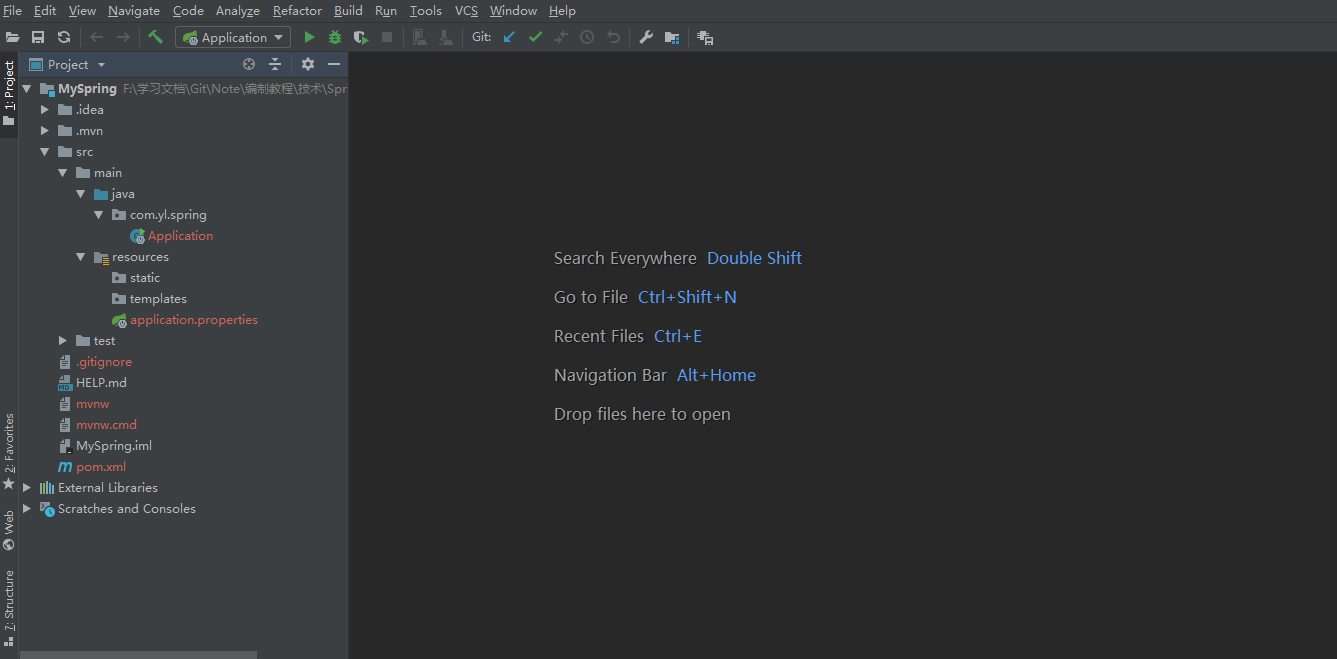
1. 配置一下我们项目的包名等信息，按照自己的需要配置即可。我的配置如下图所示。



1. 这里选择Web，同时如果需要数据库连接、使用MyBatis或者展示网页等，别忘了把对应的选项勾选上。作为Springboot web项目的开始，我们暂且不加这些特性(在后续的章节中，会一一对这些特性进行讲解)，这里先只建立一个最简单的项目展示它的开始。

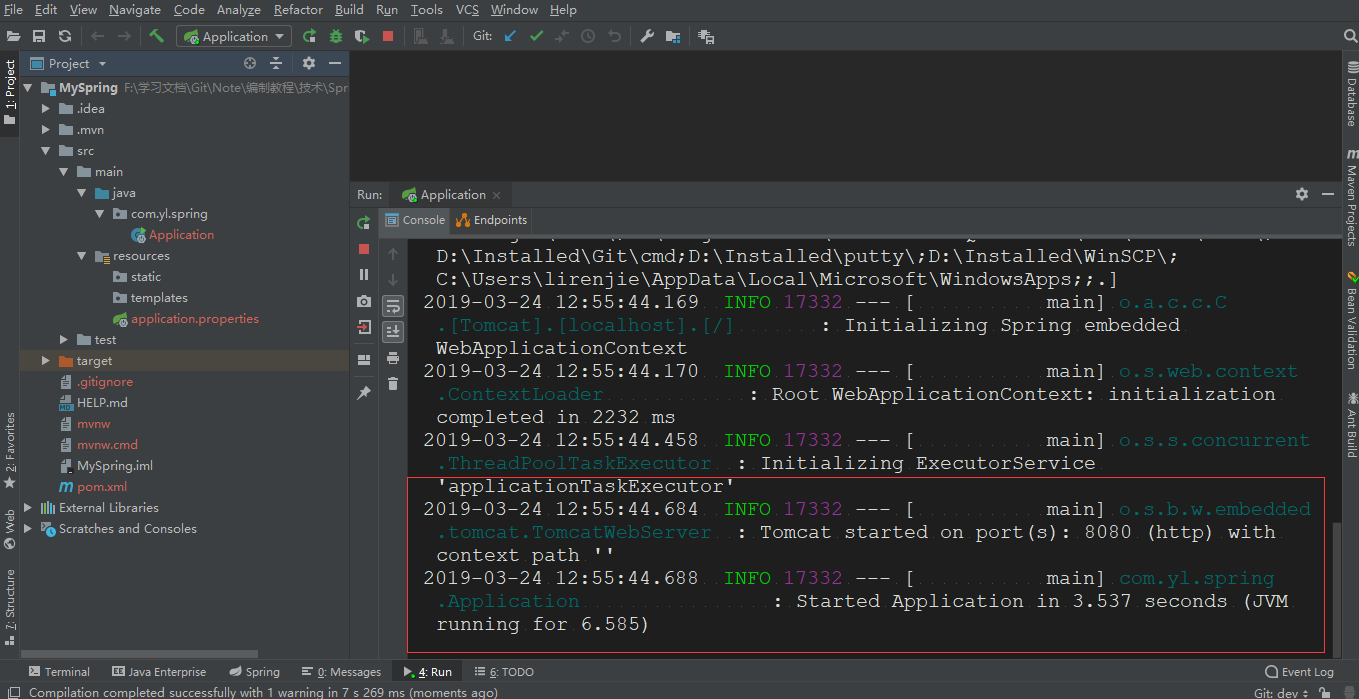


1. 经过上述几个步骤后，一个基本的Springboot web项目就完成了，直接选择Finish即可。新建后的项目结构如下图：

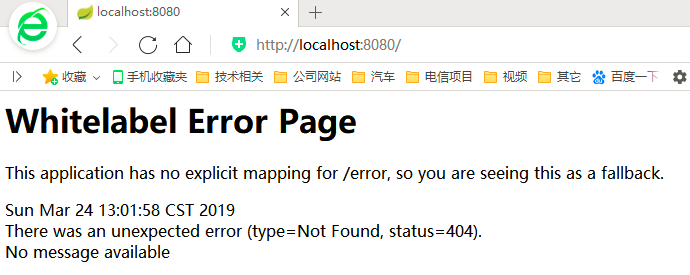


## 场景3

前面的章节中，我们说过，我们完全可以像运行普通Java一样去运行Springboot web项目，那么我们直接运行上一场景中建立的项目，会发生什么呢？

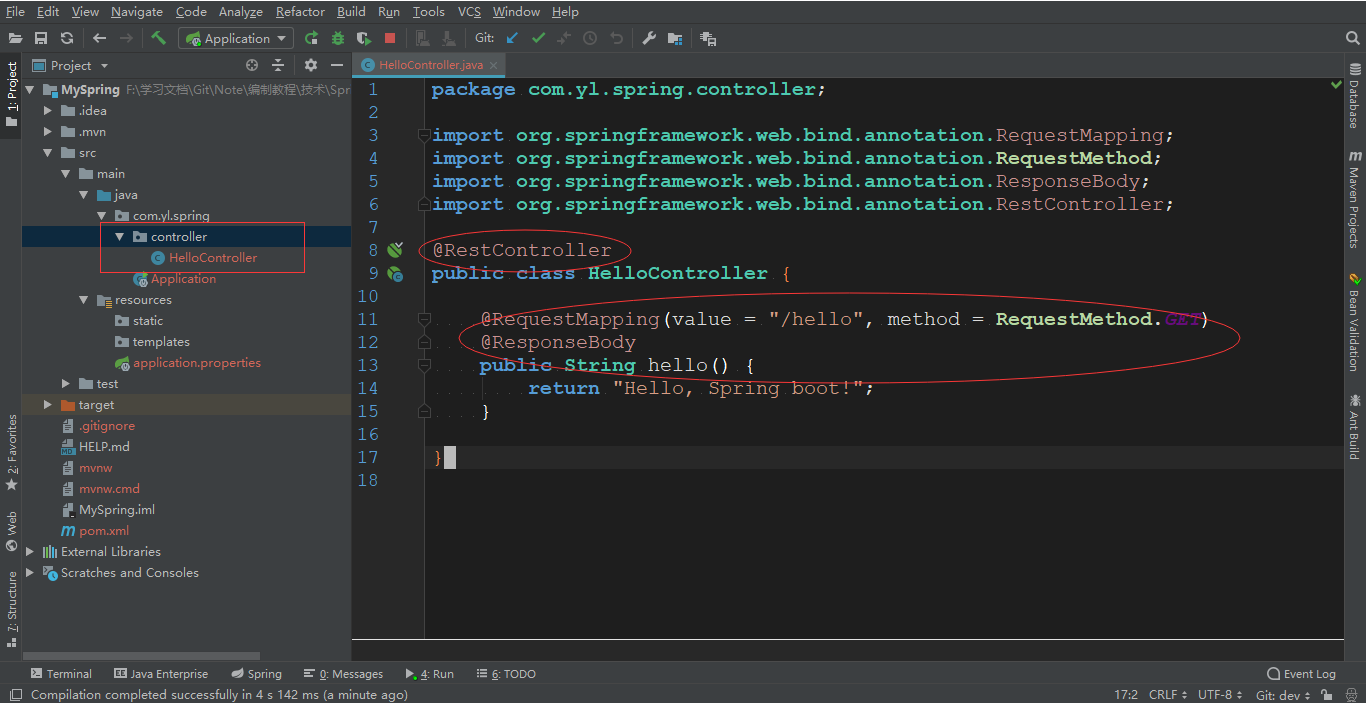


点击运行按钮之后，从日志可以看出，Tomcat服务器和应用都已经成功启动。做过web项目的各位都知道，我们在使用Spring完成项目的配置以及服务器的启动以后，在浏览器中可以访问URL：<http://localhost:8080>，这会显示Tomcat的默认页面。那么，我们的Springboot可以吗？答案令人沮丧。如下图：

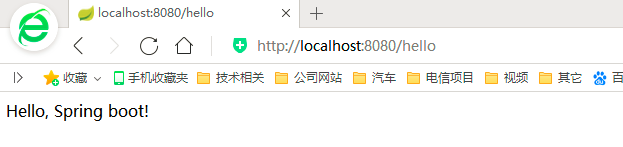


我们知道，Springboot整合了Tomcat服务器，但是为什么无法打开Tomcat的默认页面呢？TODO

因此，我们需要告诉程序，在我们访问具体的URL的时候，需要展示什么。这个时候，Springboot的Controller登场了。如下图：



我们新建了一个名为controller的包，并添加了一个Java类HelloController，详细的代码如图所示，需要特别注意的是类上添加的几个注解。完成以后，我们重新启动一下我们的项目，然后在浏览器里访问URL：<http://localhost:8080/hello>，结果令我们惊喜。



我们可以看到，从项目建立到启动运行，我们没有添加任何的配置文件，整个过程，我们就像新建一个普通Java项目一样，就完成了一个web项目的成功运行，不得不说It’s a miracle！看来，Springboot的简化绝非浪得虚名。

## 下回预告

在下回，我将向大家介绍，Springboot，如何展示一个HTML页面。咱们下回见。

# 第二回 HTML页面

## 前情回顾

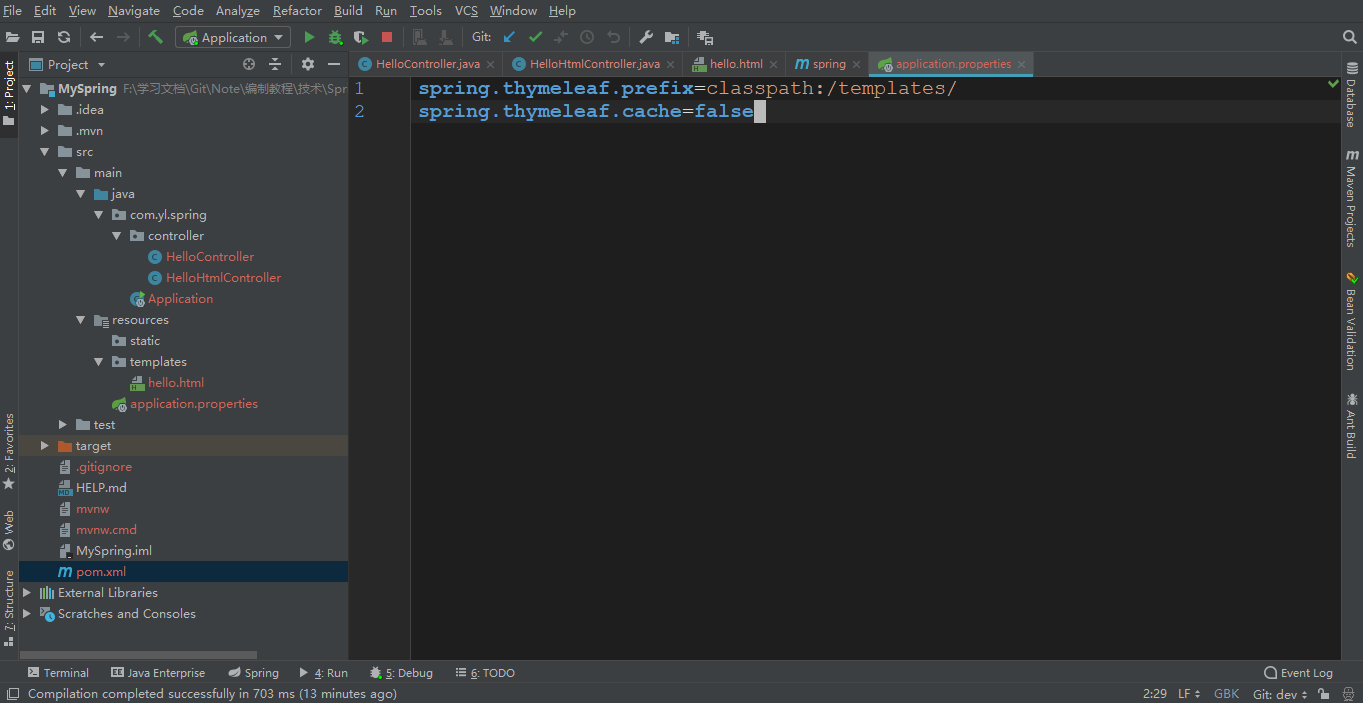
在上一回中，我们演示了如何运行建立并运行基本的Springboot项目，并成功向浏览器页面返回了一个字符串。在这一回中，我将向大家展示在访问URL的时候返回一个我们写好的HTML页面，而不再是一个单调的字符串。

## 场景1

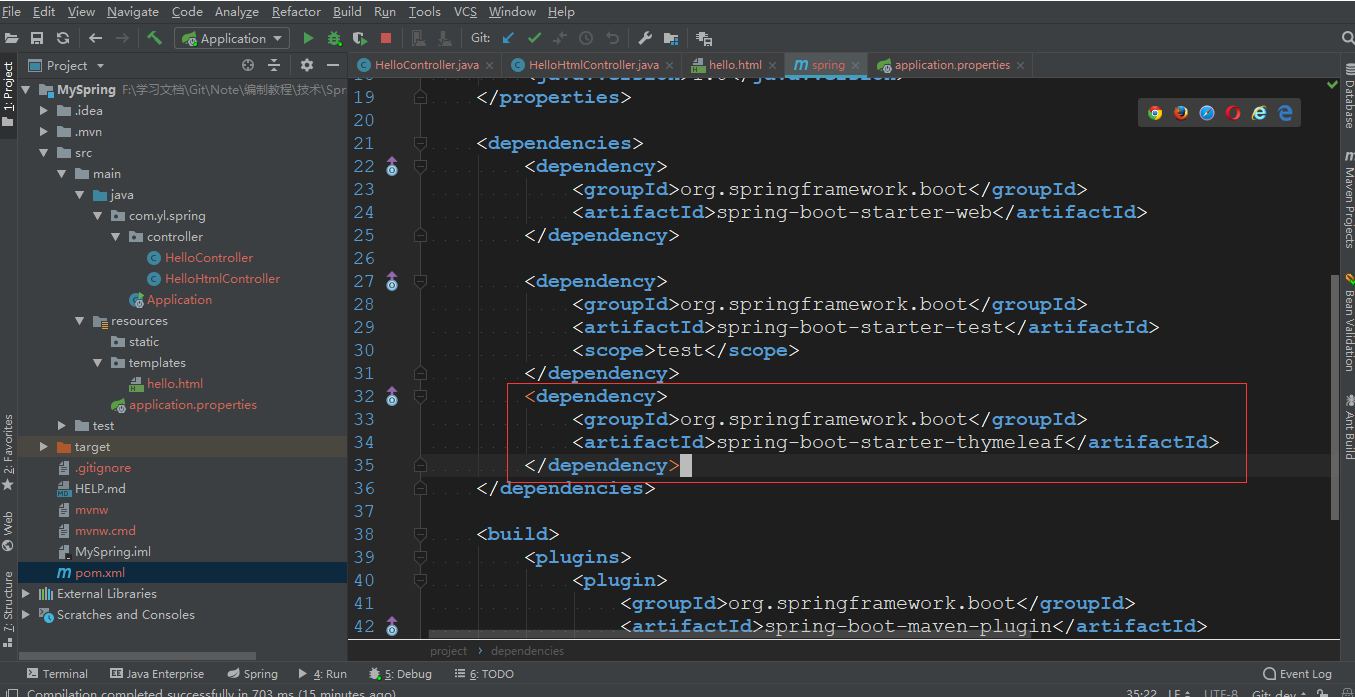
要返回一个HTML页面，需要用到一个模板引擎开发库Thymeleaf。可以处理和生成HTML、XML、JavaScript、CSS和文本，在Web和非Web环境下都可以正常工作。Thymeleaf可以跟Spring boot很好的集成。

在我们上一回建立的项目中，也很容易添加Thymeleaf的支持。

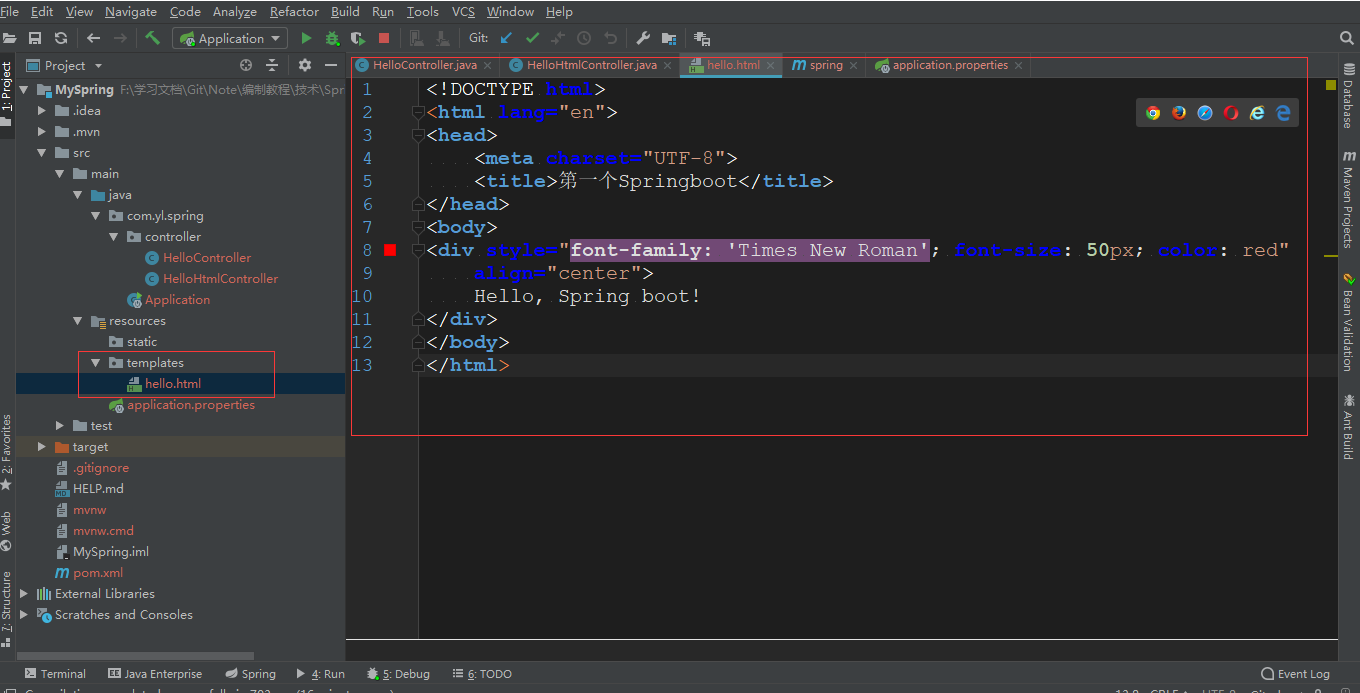
1. 在配置文件application.properties中，添加如下两项配置(其中spring.thymeleaf.cache=false这一项可以不用配置):



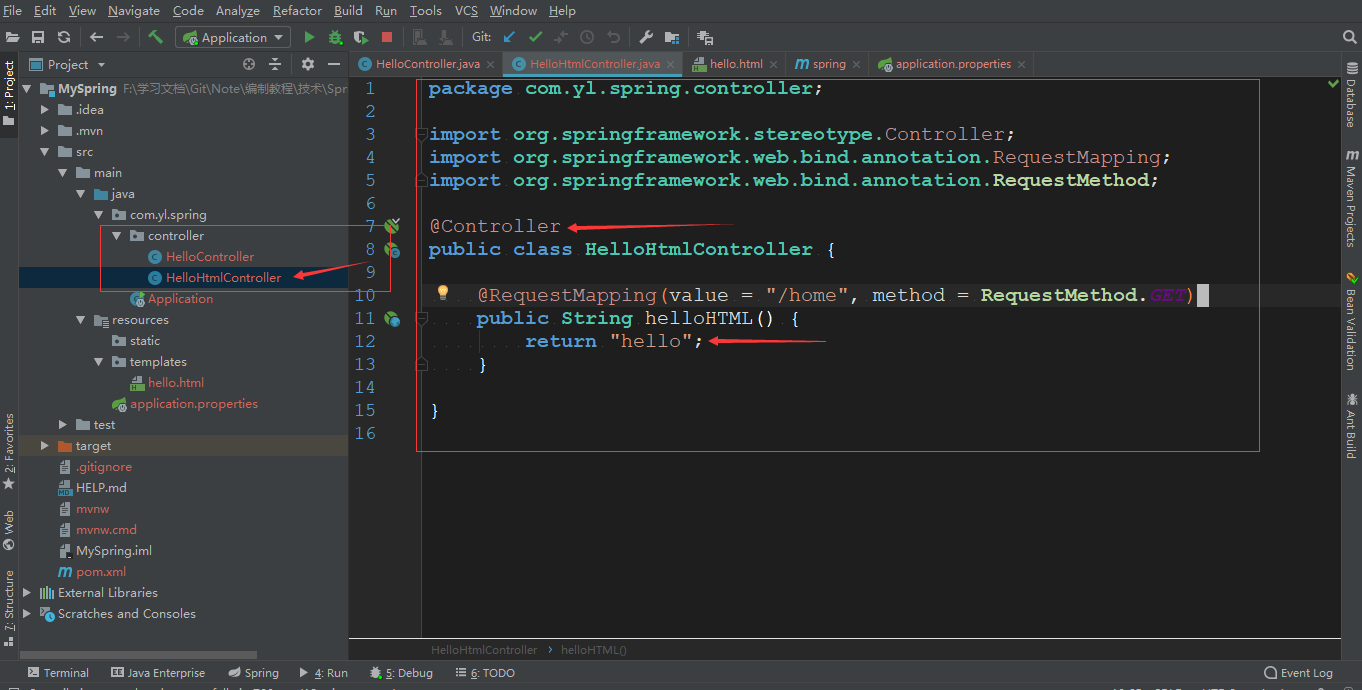
1. 在pom.xml中引入thymeleaf的dependence：



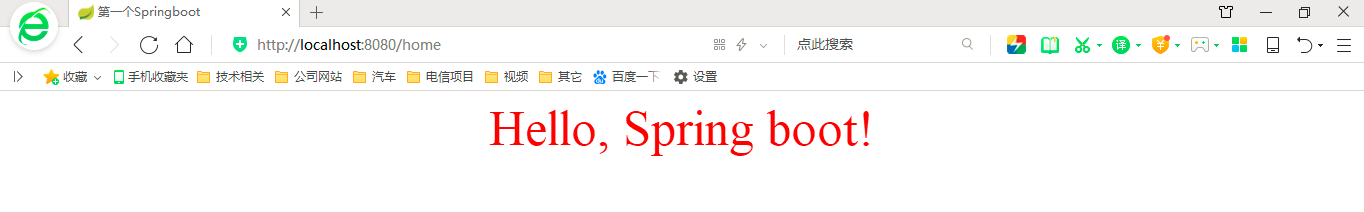
1. 配置完上面两项后，我们的项目就支持返回HTML页面了。我们添加一个简单的HTML页面：



1. 写好页面以后，还需要我们的控制器在我们访问URL的时候帮我们跳转到对应的页面上，这里需要特别注意与前面我们添加的控制器的区别，这里使用@Controller注解，而不是@RestController，如果使用后者，我们返回的hello将会被解析成字符串，而不是HTML页面。



1. 启动一下我们的项目，然后在浏览器输入<http://localhost:8080/home>



## 下回预告

在下回，我将向大家介绍，Springboot与数据库的交互(使用MySQL)。

# 第三回 整合MySQL访问

## 前情回顾

在上一回，给大家展示了Springboot如何展示一个HTML页面。在本回中，将向大家展示Springboot如何与MySQL交互。

## 场景1