**构建微服务：Spring boot 入门篇**

**什么是**spring boot

Spring Boot是由Pivotal团队提供的全新框架，其设计目的是用来简化新Spring应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。用我的话来理解，就是spring boot其实不是什么新的框架，它默认配置了很多框架的使用方式，就像maven整合了所有的jar包，spring boot整合了所有的框架（不知道这样比喻是否合适）。

**使用**spring boot**有什么好处**

其实就是简单、快速、方便！平时如果我们需要搭建一个spring web项目的时候需要怎么做呢？

1）配置web.xml，加载spring和spring mvc

2）配置数据库连接、配置spring事务

3）配置加载配置文件的读取，开启注解

4）配置日志文件

...

配置完成之后部署tomcat 调试

...

现在非常流行微服务，如果我这个项目仅仅只是需要发送一个邮件，如果我的项目仅仅是生产一个积分；我都需要这样折腾一遍!

但是如果使用spring boot呢？

很简单，我仅仅只需要非常少的几个配置就可以迅速方便的搭建起来一套web项目或者是构建一个微服务！

使用sping boot到底有多爽，用下面这幅图来表达

[](http://images2015.cnblogs.com/blog/331425/201607/331425-20160712105132873-2014012760.jpg)

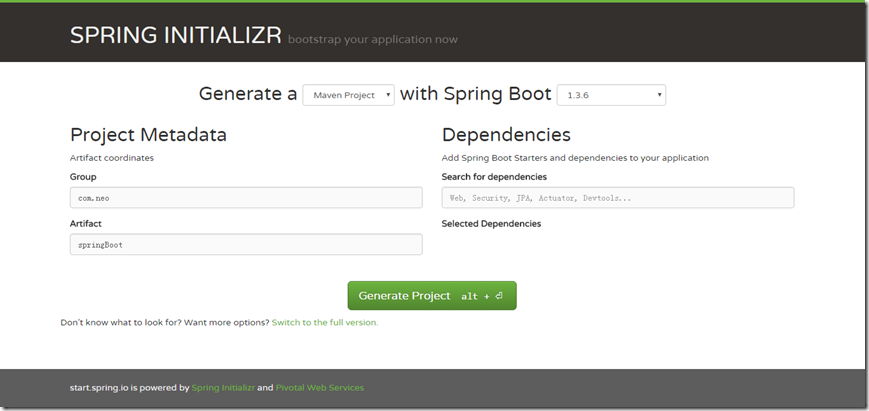
**快速入门**

说了那么多，手痒痒的很，马上来一发试试!

**maven构建项目**

1、访问http://start.spring.io/

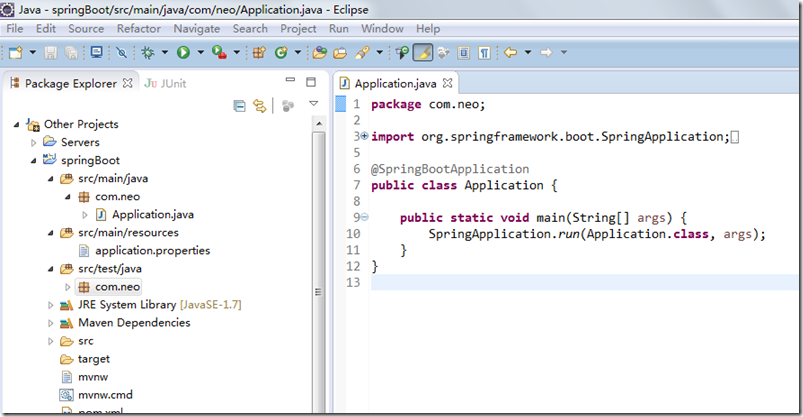
2、选择构建工具Maven Project、Spring Boot版本1.3.6以及一些工程基本信息，点击“[Switch to the full version.](http://start.spring.io/)”java版本选择1.7，可参考下图所示：

[](http://images2015.cnblogs.com/blog/331425/201607/331425-20160712105134607-1615281203.png)

3、点击Generate Project下载项目压缩包

4、解压后，使用eclipse，Import -> Existing Maven Projects -> Next ->选择解压后的文件夹-> Finsh，OK done!

**项目结构介绍**

[](http://images2015.cnblogs.com/blog/331425/201607/331425-20160712105137014-669859839.png)

如上图所示，Spring Boot的基础结构共三个文件:

src/main/java  程序开发以及主程序入口

src/main/resources 配置文件

src/test/java  测试程序

另外，spingboot建议的目录结果如下：

root package结构：com.example.myproject

com

+- example

+- myproject

+- Application.java

|

+- domain

| +- Customer.java

| +- CustomerRepository.java

|

+- service

| +- CustomerService.java

|

+- controller

| +- CustomerController.java

|

1、Application.java 建议放到跟目录下面,主要用于做一些框架配置

2、domain目录主要用于实体（Entity）与数据访问层（Repository）

3、service 层主要是业务类代码

4、controller 负责页面访问控制

采用默认配置可以省去很多配置，当然也可以根据自己的喜欢来进行更改

最后，启动Application main方法，至此一个java项目搭建好了！

**引入web模块**

1、pom.xml中添加支持web的模块：

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-starter-web</artifactId>

</dependency>

pom.xml文件中默认有两个模块：

spring-boot-starter：核心模块，包括自动配置支持、日志和YAML；

spring-boot-starter-test：测试模块，包括JUnit、Hamcrest、Mockito。

2、编写controller内容

**package** com.example.demo.contrller;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RestController;

@RestController

**public** **class** HelloWorldController {

@RequestMapping("/hello")

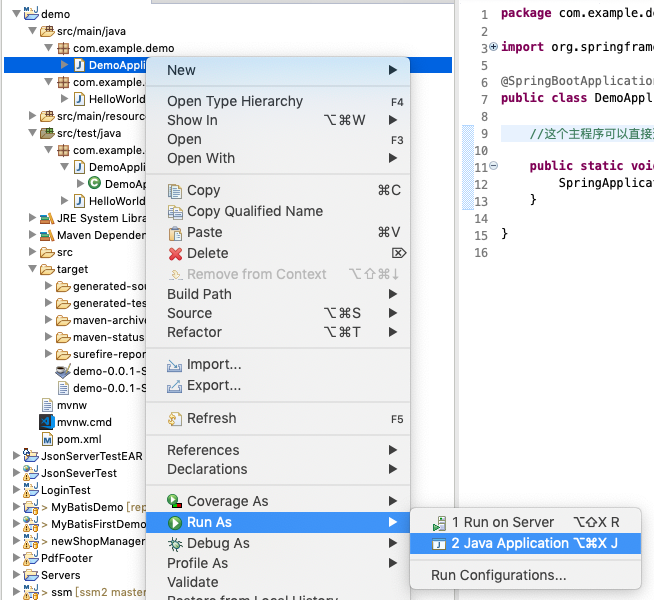
**public** String index() {

**return** "Hello World";

}

}

@RestController的意思就是controller里面的方法都以json格式输出，不用再写什么jackjson配置的了！



3、启动主程序，打开浏览器访问http://localhost:8080/hello，就可以看到效果了，有木有很简单！

启动主程序与运行java程序是一样的。

**如何做单元测试**

打开的src/test/下的测试入口，编写简单的http请求来测试；使用mockmvc进行，利用MockMvcResultHandlers.print()打印出执行结果。

**package** com.example.demo;

**import** org.junit.Before;

**import** org.junit.Test;

**import** org.junit.runner.RunWith;

**import** org.springframework.boot.test.context.SpringBootTest;

**import** org.springframework.http.MediaType;

**import** org.springframework.test.context.junit4.SpringRunner;

**import** org.springframework.test.web.servlet.MockMvc;

**import** org.springframework.test.web.servlet.request.MockMvcRequestBuilders;

**import** org.springframework.test.web.servlet.result.MockMvcResultHandlers;

**import** org.springframework.test.web.servlet.result.MockMvcResultMatchers;

**import** org.springframework.test.web.servlet.setup.MockMvcBuilders;

**import** com.example.demo.contrller.HelloWorldController;

//用来测试HelloWorldController

@RunWith(SpringRunner.**class**)

@SpringBootTest

**public** **class** HelloWorldControlerTests {

**private** MockMvc mvc;

@Before

**public** **void** setUp() **throws** Exception {

mvc = MockMvcBuilders.*standaloneSetup*(**new** HelloWorldController()).build();

}

@Test

**public** **void** getHello() **throws** Exception {

mvc.perform(MockMvcRequestBuilders.*get*("/hello").accept(MediaType.***APPLICATION\_JSON***))

.andExpect(MockMvcResultMatchers.*status*().isOk())

.andDo(MockMvcResultHandlers.*print*())

.andReturn();

}

}

**开发环境的调试**

热启动在正常开发项目中已经很常见了吧，虽然平时开发web项目过程中，改动项目启重启总是报错；但springBoot对调试支持很好，修改之后可以实时生效，需要添加以下的配置：

<dependencies>

<dependency>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-devtools</artifactId>

<optional>true</optional>

</dependency>

</dependencies>

<build>

<plugins>

<plugin>

<groupId>org.springframework.boot</groupId>

<artifactId>spring-boot-maven-plugin</artifactId>

<configuration>

<fork>true</fork>

</configuration>

</plugin>

</plugins>

</build>

该模块在完整的打包环境下运行的时候会被禁用。如果你使用java -jar启动应用或者用一个特定的classloader启动，它会认为这是一个“生产环境”。

**总结**

使用spring boot可以非常方便、快速搭建项目，使我们不用关心框架之间的兼容性，适用版本等各种问题，我们想使用任何东西，仅仅添加一个配置就可以，所以使用sping boot非常适合构建微服务。

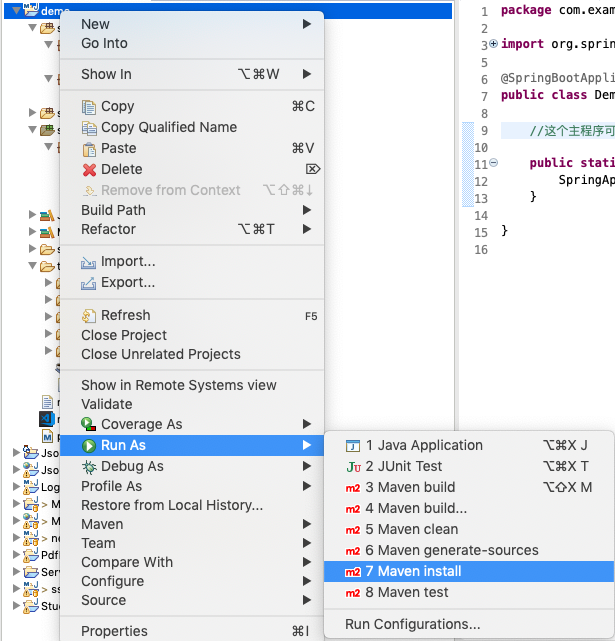
[**示例代码-github**](https://github.com/ityouknow/spring-boot-examples)

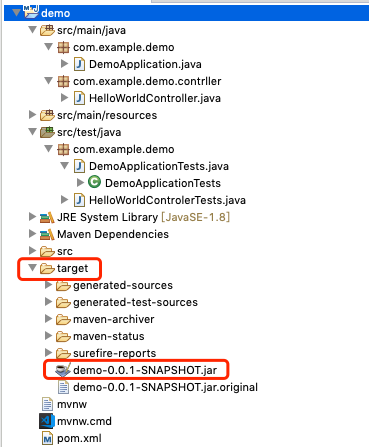
[**示例代码-码云**](https://gitee.com/ityouknow/spring-boot-examples)

[**Spring Boot 中文索引**](https://github.com/ityouknow/awesome-spring-boot)

原文参考：<https://www.cnblogs.com/ityouknow/p/5662753.html>

下面是Maven直接打包：

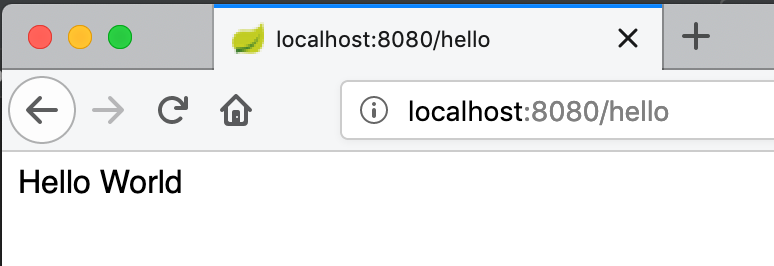




使用Maven打包后的jar包直接运行，这个里面是集成有Tomcat的，所以可以直接运行。



用浏览器打开就可以看到结果



但是奇怪的这个并不是json格式数据

