[**第一篇：构建第一个SpringBoot工程**](http://blog.csdn.net/forezp/article/details/70341651)

参考：<http://blog.csdn.net/forezp/article/details/70341651>

转载请标明出处：   
<http://blog.csdn.net/forezp/article/details/70341651>   
本文出自[方志朋的博客](http://blog.csdn.net/forezp)

## 简介

spring boot 它的设计目的就是为例简化开发，开启了各种自动装配，你不需要写各种配置文件，引入相关的依赖就能迅速搭建起一个web工程。它采用的是建立生产就绪的应用程序观点，优先于配置的惯例。

可能你有很多理由不放弃SSM,SSH，但是当你一旦使用了springboot ,你会觉得一切变得简单了，配置变的简单了、编码变的简单了，部署变的简单了，感觉自己健步如飞，开发速度大大提高了。就好比，当你用了IDEA，你会觉得再也回不到Eclipse时代一样。另，本系列教程全部用的IDEA作为开发工具。

## 建构工程

你需要：

* 15分钟
* jdk 1.8或以上
* maven 3.0+
* Idea

打开Idea-> new Project ->Spring Initializr ->填写group、artifact ->钩上web(开启web功能）->点下一步就行了。

## 工程目录

创建完工程，工程的目录结构如下：

- src

-main

-java

-package

-SpringbootApplication

-resouces

- statics

- templates

- application.yml

-test

- pom

* pom文件为基本的依赖管理文件
* resouces 资源文件   
  + statics 静态资源
  + templates 模板资源
  + application.yml 配置文件
* SpringbootApplication程序的入口。

pom.xml的依赖：

<?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.forezp</groupId>

<artifactId>springboot-first-application</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>springboot-first-application</name>

<description>Demo project for Spring Boot</description>

<parent>

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

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

<version>1.5.2.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-web</artifactId>

</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>

其中spring-boot-starter-web不仅包含spring-boot-starter,还自动开启了web功能。

## 功能演示

说了这么多，你可能还体会不到，举个栗子，比如你引入了Thymeleaf的依赖，spring boot 就会自动帮你引入SpringTemplateEngine，当你引入了自己的SpringTemplateEngine，spring boot就不会帮你引入。它让你专注于你的自己的业务开发，而不是各种配置。

再举个栗子,建个controller：

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

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

@RestController

public class HelloController {

@RequestMapping("/")

public String index() {

return "Greetings from Spring Boot!";

}

}

启动SpringbootFirstApplication的main方法，打开浏览器localhost:8080,浏览器显示：

Greetings from Spring Boot!

#### 神奇之处：

* 你没有做任何的web.xml配置。
* 你没有做任何的sping mvc的配置; springboot为你做了。
* 你没有配置tomcat ;springboot内嵌tomcat.

#### 启动springboot 方式

cd到项目主目录:

mvn clean

mvn package 编译项目的jar

* mvn spring-boot: run 启动
* cd 到target目录，java -jar 项目.jar

## 来看看springboot在启动的时候为我们注入了哪些bean

在程序入口加入：

@SpringBootApplication

public class SpringbootFirstApplication {

public static void main(String[] args) {

SpringApplication.run(SpringbootFirstApplication.class, args);

}

@Bean

public CommandLineRunner commandLineRunner(ApplicationContext ctx) {

return args -> {

System.out.println("Let's inspect the beans provided by Spring Boot:");

String[] beanNames = ctx.getBeanDefinitionNames();

Arrays.sort(beanNames);

for (String beanName : beanNames) {

System.out.println(beanName);

}

};

}

}

程序输出：

Let’s inspect the beans provided by Spring Boot:   
basicErrorController   
beanNameHandlerMapping   
beanNameViewResolver   
characterEncodingFilter   
commandLineRunner   
conventionErrorViewResolver   
defaultServletHandlerMapping   
defaultViewResolver   
dispatcherServlet   
dispatcherServletRegistration   
duplicateServerPropertiesDetector   
embeddedServletContainerCustomizerBeanPostProcessor   
error   
errorAttributes   
errorPageCustomizer   
errorPageRegistrarBeanPostProcessor

….   
….

在程序启动的时候，springboot自动诸如注入了40-50个bean.

## 单元测试

通过@RunWith() @SpringBootTest开启注解：

@RunWith(SpringRunner.class)

@SpringBootTest(webEnvironment = SpringBootTest.WebEnvironment.RANDOM\_PORT)

public class HelloControllerIT {

@LocalServerPort

private int port;

private URL base;

@Autowired

private TestRestTemplate template;

@Before

public void setUp() throws Exception {

this.base = new URL("http://localhost:" + port + "/");

}

@Test

public void getHello() throws Exception {

ResponseEntity<String> response = template.getForEntity(base.toString(),

String.class);

assertThat(response.getBody(), equalTo("Greetings from Spring Boot!"));

}

}

* 1
* 2
* 3
* 4
* 5
* 6
* 7
* 8
* 9
* 10
* 11
* 12
* 13
* 14
* 15
* 16
* 17
* 18
* 19
* 20
* 21
* 22
* 23
* 24
* 25

运行它会先开启sprigboot工程，然后再测试，测试通过 ^.^

源码下载：<https://github.com/forezp/SpringBootLearning>

## 结语

市面上有很多springboot的书，有很多springboot的博客，为什么我还要写这样一个系列？到目前为止，我没有看过一本springboot的书，因为还没来得及看，看的都是官方指南，当然也参考了很多的博客，他们都写的非常的棒！在看官方指南和博客的时候，发现他们有很多不同之处，所以我打算写一个来源于官方，通过自己理解加整合写一个系列，所以取名叫《springboot 非官方教程》。我相信我写的可能跟其他人的写的会不太一样。另外，最主要的原因还是提高自己，怀着一个乐于分享的心，将自己的理解分享给更多需要的人。

## 参考资料

[Building an Application with Spring Boot](https://spring.io/guides/gs/spring-boot/)

### 优秀文章推荐：

* 更多springboot 教程：[springBoot非官方教程 | 终章：文章汇总](http://blog.csdn.net/forezp/article/details/70341818)
* 更多springcoud 教程：[史上最简单的 SpringCloud 教程 | 终章: 文章汇总](http://blog.csdn.net/forezp/article/details/70148833)