# 前言

Spring一直是很火的一个开源框架，在过去的一段时间里，Spring Boot在社区中热度一直很高，所以决定花时间来了解和学习，为自己做技术储备。

# 正文

首先声明，Spring Boot不是一门新技术，所以不用紧张。从本质上来说，Spring Boot就是Spring,它做了那些没有它你也会去做的Spring Bean配置。它使用“习惯优于配置”（项目中存在大量的配置，此外还内置了一个习惯性的配置，让你无需手动进行配置）的理念让你的项目快速运行起来。使用Spring Boot很容易创建一个独立运行（运行jar,内嵌Servlet容器）、准生产级别的基于Spring框架的项目，使用Spring Boot你可以不用或者只需要很少的Spring配置。

## Spring Boot精要

Spring将很多魔法带入了Spring应用程序的开发之中，其中最重要的是以下四个核心。

* 自动配置：针对很多Spring应用程序常见的应用功能，Spring Boot能自动提供相关配置
* 起步依赖：告诉Spring Boot需要什么功能，它就能引入需要的库。
* 命令行界面：这是Spring Boot的可选特性，借此你只需写代码就能完成完整的应用程序，无需传统项目构建。
* Actuator：让你能够深入运行中的Spring Boot应用程序，一探究竟。

详细的我们就不展开，等你爱上后自然会去深入的了解，后续章节我们会一一展开介绍。接下来让我们开搞吧。我已经迫不及待的要尝尝Spring Boot的味道了。

## 系统要求

目前Spring Boot正式版为1.5.1.RELEASE默认情况下，Spring Boot 1.5.1.RELEASE需要Java 7和Spring Framework 4.3.6.RELEASE或更高版本,你也可以使用Spring Boot with Java 6和一些额外的配置（不建议）,使用Maven（3.2+）或Gradle 2（2.9或更高版本）和3来构建。

虽然你可以使用Java 6或7的Spring Boot，但我们通常推荐Java 8。

所以本博客系列统一使用Java 1.8，Spring Boot 1.5.1.RELEASE以及Maven3.3.9版本。开发工具使用IDEA（强烈推荐，可以看我另外一篇介绍IDEA入门文章，内有官方中文教程:[Java人员正确使用 IntelliJ IDEA的方式](http://tengj.top/2017/02/22/idea1-1/)）

## 光速入门-开发一个web服务

没有比较就没有伤害，让我们先看看传统Spring MVC开发一个简单的Hello World Web应用程序，你应该做什么，我能想到一些基本的需求。

* 一个项目结构，其中有一个包含必要依赖的Maven或者Gradle构建文件，最起码要有Spring MVC和Servlet API这些依赖。
* 一个web.xml文件（或者一个WebApplicationInitializer实现），其中声明了Spring的DispatcherServlet。
* 一个启动了Spring MVC的Spring配置
* 一控制器类，以“hello World”相应HTTP请求。
* 一个用于部署应用程序的Web应用服务器，比如Tomcat。

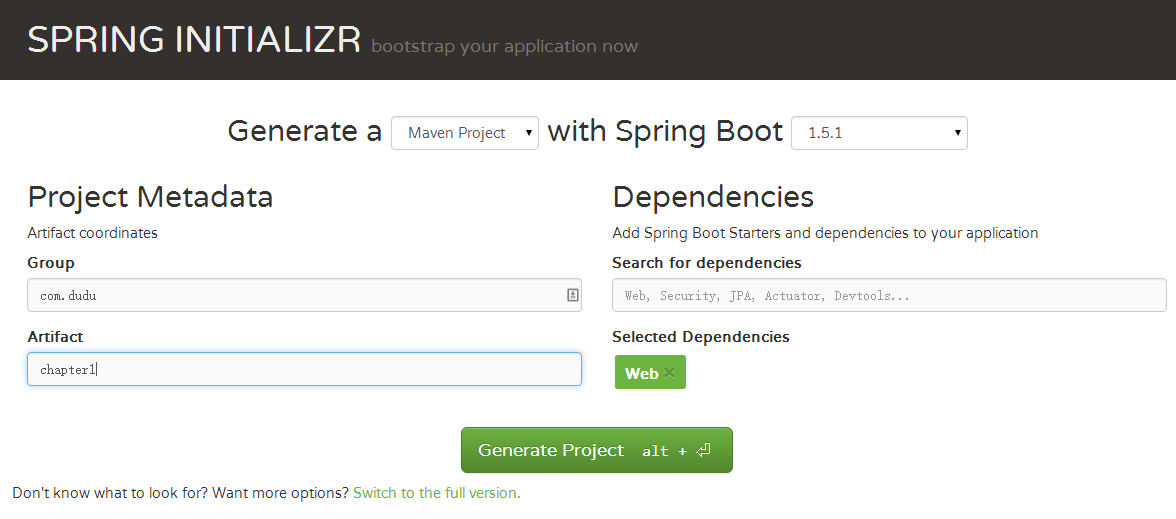
最让人难以接受的是，这份清单里面只有一个东西是和Hello World功能相关的，即控制器，剩下的都是Spring开发的Web应用程序必需的通用模板。

接下来看看Spring Boot如何搞定？  
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

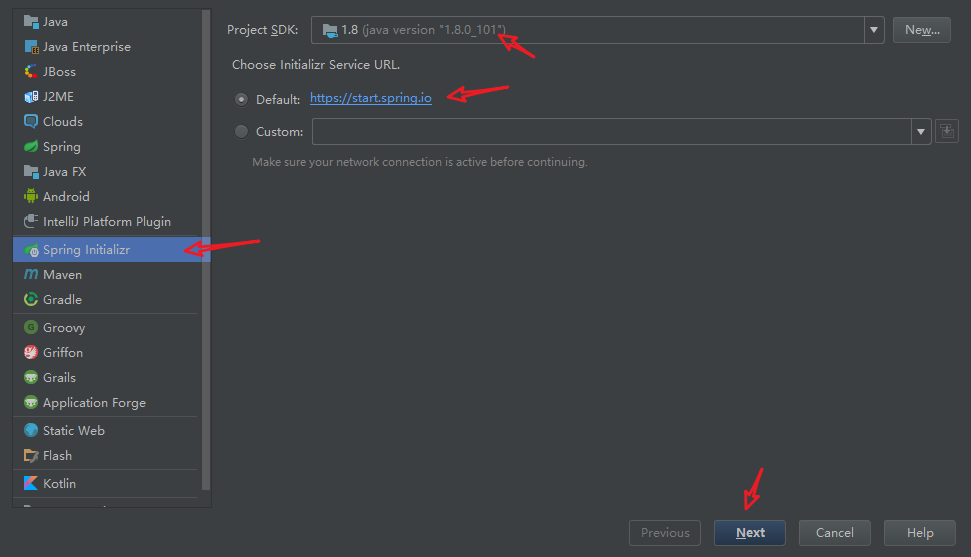
## 初始化Spring Boot项目

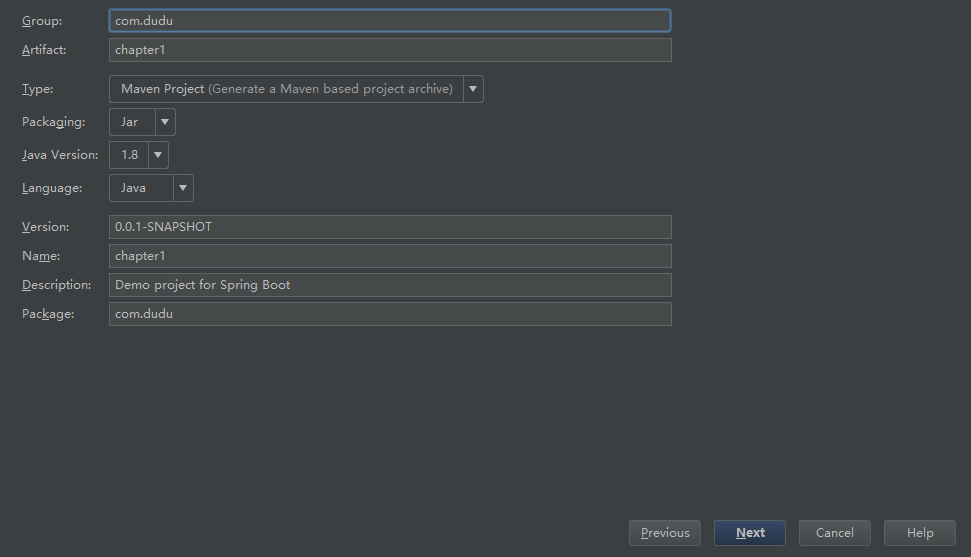
构建一个Sping Boot的Maven项目，强烈推荐Spring Initializr,它从本质上来说就是一个Web应用程序，它能为你生成Spring Boot项目结构。  
Spring Initializr有几种用法：

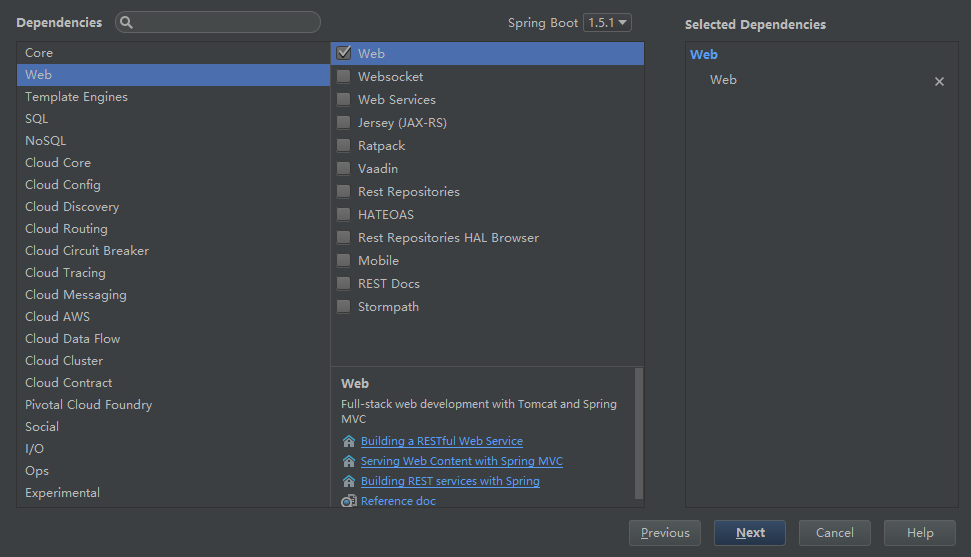
* 通过Web界面使用

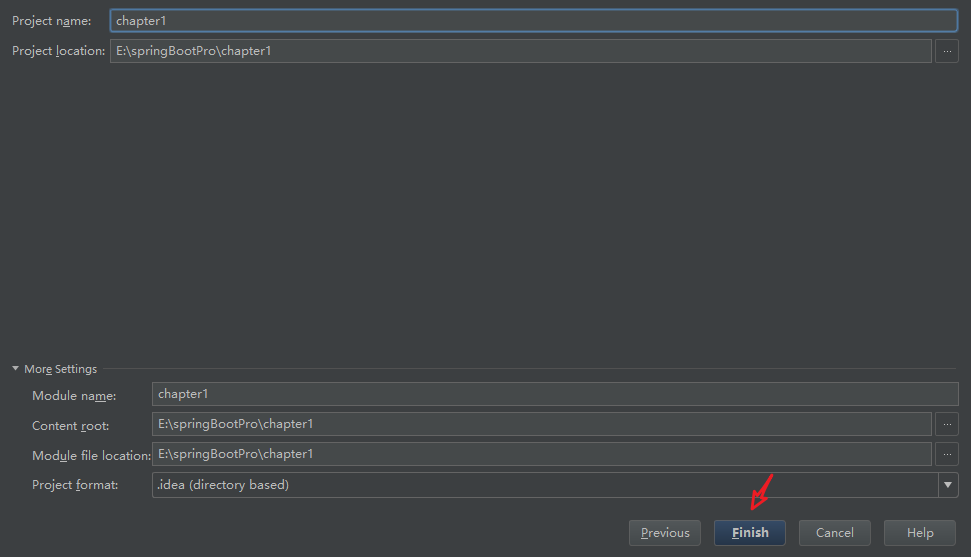
1. 访问：http://start.spring.io/
2. 选择构建工具Maven Project、Spring Boot版本1.5.1以及一些工程基本信息，可参考下图所示  
   [](http://7xqch5.com1.z0.glb.clouddn.com/springboot1-1.png)
3. 点击Generate Project下载项目压缩包
4. 导入到你的工程，如果是IDEA，则需要：  
   a.菜单中选择File–>New–>Project from Existing Sources...  
   b.选择解压后的项目文件夹，点击OK  
   c.点击Import project from external model并选择Maven，点击Next到底为止。  
   d.若你的环境有多个版本的JDK，注意到选择Java SDK的时候请选择Java 7以上的版本

* 通过IntelliJ IDEA使用(个人推荐)  
  IntelliJ IDEA是非常流行的IDE，IntelliJ IDEA 14.1已经支持Spring Boot了。  
  创建Spring Boot操作步骤如下：  
  1.在File菜单里面选择 New > Project,然后选择Spring Initializr，接着如下图一步步操作即可。

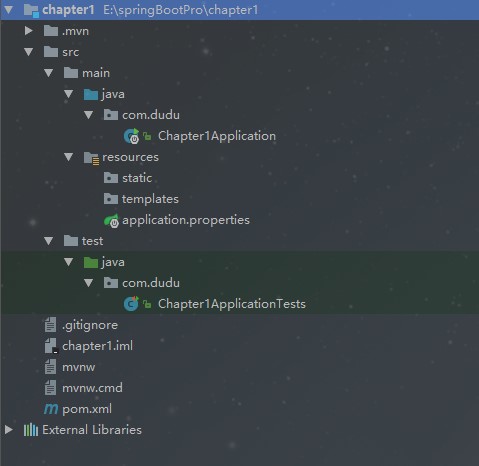
[](http://7xqch5.com1.z0.glb.clouddn.com/springboot1-2.png)

[](http://7xqch5.com1.z0.glb.clouddn.com/springboot1-3.png)

[](http://7xqch5.com1.z0.glb.clouddn.com/springboot1-4.png)

[](http://7xqch5.com1.z0.glb.clouddn.com/springboot1-5.png)

## 项目结构

根据上面的操作已经初始化了一个Spring Boot的框架了，项目结构如下：  
[](http://7xqch5.com1.z0.glb.clouddn.com/springboot1-6.png)  
如你所见，项目里面基本没有代码，除了几个空目录外，还包含如下几样东西。

* pom.xml：Maven构建说明文件。
* Chapter1Application.java：一个带有main()方法的类，用于启动应用程序（关键）。
* Chapter1ApplicationTests.java：一个空的Junit测试类，它加载了一个使用Spring Boot字典配置功能的Spring应用程序上下文。
* application.properties：一个空的properties文件，你可以根据需要添加配置属性。

## 解析pom.xml

大家跟我一起移步pom.xml,看看Spring Boot的跟普通Spring MVC工程的Maven配置有啥不一样

|  |
| --- |
| <?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.dudu</groupId>  <artifactId>chapter1</artifactId>  <version>0.0.1-SNAPSHOT</version>  <packaging>jar</packaging>   <name>chapter1</name>  <description>Demo project for Spring Boot</description>   <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.5.1.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> |

可以看出，有几个配置是不大一样的,我们挑几个重要的说说，要是客观不想过早的了解这些，自行跳过这一知识点即可。

### Spring Boot父级依赖

|  |
| --- |
| <parent>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-starter-parent</artifactId>  <version>1.5.1.RELEASE</version>  <relativePath/> <!-- lookup parent from repository --> </parent> |

这块配置就是Spring Boot父级依赖，有了这个，当前的项目就是Spring Boot项目了，spring-boot-starter-parent是一个特殊的starter,它用来提供相关的Maven默认依赖，使用它之后，常用的包依赖可以省去version标签。关于Spring Boot提供了哪些jar包的依赖，可查看C:\Users\用户.m2\repository\org\springframework\boot\spring-boot-dependencies\1.5.1.RELEASE\spring-boot-dependencies-1.5.1.RELEASE.pom  
这里我就贴一点点意思意思，如下：

|  |
| --- |
| <properties>  <!-- Dependency versions -->  <activemq.version>5.14.3</activemq.version>  <antlr2.version>2.7.7</antlr2.version>  <appengine-sdk.version>1.9.48</appengine-sdk.version>  <artemis.version>1.5.2</artemis.version>  <aspectj.version>1.8.9</aspectj.version>  <assertj.version>2.6.0</assertj.version>  <atomikos.version>3.9.3</atomikos.version>  <bitronix.version>2.1.4</bitronix.version>  <caffeine.version>2.3.5</caffeine.version>  <cassandra-driver.version>3.1.3</cassandra-driver.version>  <classmate.version>1.3.3</classmate.version>  <commons-beanutils.version>1.9.3</commons-beanutils.version>  <commons-collections.version>3.2.2</commons-collections.version>  <spring-data-releasetrain.version>Ingalls-RELEASE</spring-data-releasetrain.version>  ...... </properties> |

如果你不想使用某个依赖默认的版本，您还可以通过覆盖自己的项目中的属性来覆盖各个依赖项，例如，要升级到另一个Spring Data版本系列，您可以将以下内容添加到pom.xml中。

|  |
| --- |
| <properties>  <spring-data-releasetrain.version>Fowler-SR2</spring-data-releasetrain.version> </properties> |

原本默认版本是Ingalls-RELEASE的（看上面最后一行有说明Ingalls-RELEASE），现在就使用Fowler-SR2版本了，简单吧。

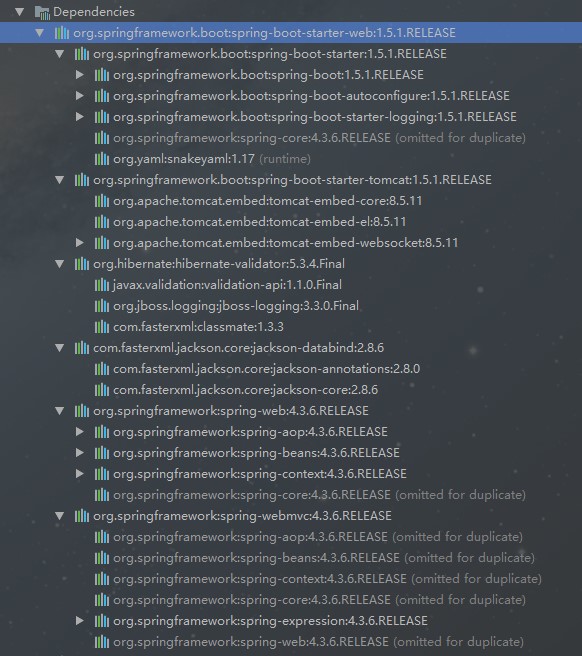
并不是每个人都喜欢继承自spring-boot-starter-parent POM。您可能有您需要使用的自己的公司标准parent，或者您可能更喜欢显式声明所有的Maven配置。  
如果你不想使用spring-boot-starter-parent，您仍然可以通过使用scope = import依赖关系来保持依赖关系管理：

|  |
| --- |
| <dependencyManagement>  <dependencies>  <dependency>  <!-- Import dependency management from Spring Boot -->  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-dependencies</artifactId>  <version>1.5.1.RELEASE</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies> </dependencyManagement> |

该设置不允许您使用如上所述的属性(properties)覆盖各个依赖项，要实现相同的结果，您需要在spring-boot-dependencies项之前的项目的dependencyManagement中添加一个配置，例如，要升级到另一个Spring Data版本系列，您可以将以下内容添加到pom.xml中。

|  |
| --- |
| <dependencyManagement>  <dependencies>  <!-- Override Spring Data release train provided by Spring Boot -->  <dependency>  <groupId>org.springframework.data</groupId>  <artifactId>spring-data-releasetrain</artifactId>  <version>Fowler-SR2</version>  <scope>import</scope>  <type>pom</type>  </dependency>  <dependency>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-dependencies</artifactId>  <version>1.5.1.RELEASE</version>  <type>pom</type>  <scope>import</scope>  </dependency>  </dependencies> </dependencyManagement> |

### 起步依赖 spring-boot-starter-xx

Spring Boot提供了很多”开箱即用“的依赖模块，都是以spring-boot-starter-xx作为命名的。举个例子来说明一下这个起步依赖的好处，比如组装台式机和品牌机，自己组装的话需要自己去选择不同的零件，最后还要组装起来，期间有可能会遇到零件不匹配的问题。耗时又消力，而品牌机就好一点，买来就能直接用的，后续想换零件也是可以的。相比较之下，后者带来的效果更好点（这里就不讨论价格问题哈），起步依赖就像这里的品牌机，自动给你封装好了你想要实现的功能的依赖。就比如我们之前要实现web功能，引入了spring-boot-starter-web这个起步依赖。我们来看看spring-boot-starter-web到底依赖了哪些,如下图：  
[](http://7xqch5.com1.z0.glb.clouddn.com/springboot1-7.png)

嘿嘿嘿，看来依赖了好多呢，如果让我自己弄估计要调半天，所以Spring Boot通过提供众多起步依赖降低项目依赖的复杂度。起步依赖本质上是一个Maven项目对象模型（Project Object Model，POM），定义了对其他库的传递依赖，这些东西加在一起即支持某项功能。很多起步依赖的命名都暗示了它们提供的某种或者某类功能。

### Spring Boot Maven插件

|  |
| --- |
| <build>  <plugins>  <plugin>  <groupId>org.springframework.boot</groupId>  <artifactId>spring-boot-maven-plugin</artifactId>  </plugin>  </plugins> </build> |

上面的配置就是Spring Boot Maven插件，Spring Boot Maven插件提供了许多方便的功能：

* 把项目打包成一个可执行的超级JAR（uber-JAR）,包括把应用程序的所有依赖打入JAR文件内，并为JAR添加一个描述文件，其中的内容能让你用java -jar来运行应用程序。
* 搜索public static void main()方法来标记为可运行类。

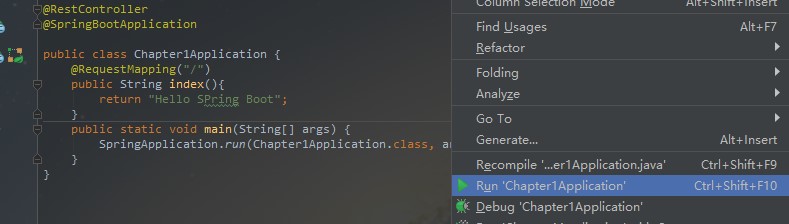
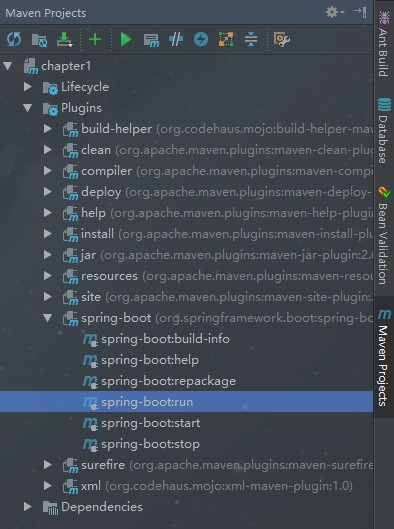
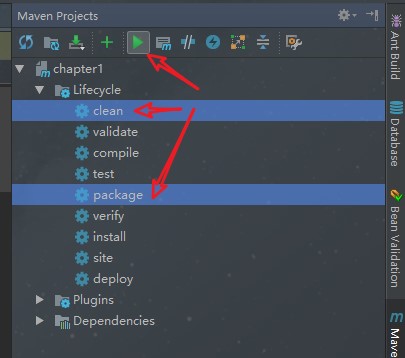
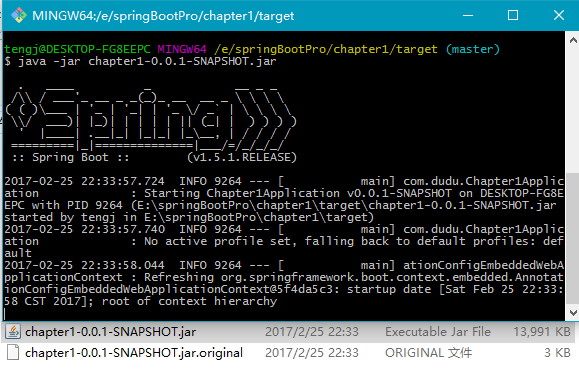
## 应用入口类

Chapter1Application是一个很关键的启动类，程序的入口就是这里,为了演示简单，我们不再新建控制类，而是直接在这个入口类中编写，添加@RestController以及index方法，如下：

|  |
| --- |
| package com.dudu;  import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RestController;  @RestController @SpringBootApplication public class Chapter1Application {   @RequestMapping("/")  public String index(){  return "Hello Spring Boot";  }  public static void main(String[] args) {  SpringApplication.run(Chapter1Application.class, args);  } } |

1. @SpringBootApplication是Sprnig Boot项目的核心注解，主要目的是开启自动配置。后续讲解原理的时候再深入介绍。
2. main方法这是一个标准的Java应用的main的方法，主要作用是作为项目启动的入口。
3. @RestController注解等价于@Controller+@ResponseBody的结合，使用这个注解的类里面的方法都以json格式输出。

最后，启动项目有三种方式：

* Chapter1Application的main方法  
  [](http://7xqch5.com1.z0.glb.clouddn.com/springboot1-8.png)
* 使用命令 mvn spring-boot:run”在命令行启动该应用，IDEA中该命令在如下位置：  
  [](http://7xqch5.com1.z0.glb.clouddn.com/springboot1-9.png)
* 运行“mvn package”进行打包时，会打包成一个可以直接运行的 JAR 文件，使用“java -jar”命令就可以直接运行。  
  [](http://7xqch5.com1.z0.glb.clouddn.com/springboot1-10.png)  
  [](http://7xqch5.com1.z0.glb.clouddn.com/springboot1-11.png)

打开浏览器访问[http://localhost:8080](http://localhost:8080/), 你就能看到页面显示Hello Spring Boot效果了，一个简单的Web的项目就是如此简单。

# 结束

虽然我上面讲解了那么多，但是实际开发步骤就是那么简单暴力，初始化一个Spring Boot，添加一个控制类，启动就能看到效果了。  
本章作为入门Spring Boot的入门介绍，相关的需要理解的概念就只有这几点：

* Spring Boot父级依赖的概念
* 起步依赖 spring-boot-starter-xx的概念
* 应用入口类的作用