## 1.1快速开发一个Spring boot程序步骤如下：

### 创建一个Spring boot项目；

### 2）目录结构：

├── mvnw

├── mvnw.cmd

├── pom.xml Maven构建说明文件

└── src

├── main

│ ├── java

│ │ └── com

│ │ └── example

│ │ └── DemoApplication.java 带有main方法

│ └── resources

│ ├── application.properties 配置项目相关属性

│ ├── static 存放相关静态文件

│ └── templates 存放模板渲染文件

└── test

└── java

└── com

└── example

└── demo

└── DemoApplicationTests.java Junit

### 3）加入Spring boot的父级和起步依赖；

1. 父级依赖

<**parent**>

<**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-parent</**artifactId**>  
 <**version**>2.1.3.RELEASE</**version**>  
 <**relativePath**/> *<!-- lookup parent from repository -->*</**parent**>

mvn dependency:tree命令显示项目依赖关系的树形表示。

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。

1. 起步依赖

<**dependency**>

<**groupId**>org.springframework.boot</**groupId**>  
 <**artifactId**>spring-boot-starter-web</**artifactId**>  
</**dependency**>

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。

1. maven插件

<！-打包为可执行jar->

<build>

<plugins>

<plugin>

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

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

</ plugin >

</ plugins>

</ build>

mvn package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。

运行mvn dependency:tree，会发现现在有许多其他依赖项，包括Tomcat Web服务器和Spring Boot本身。

### 4）创建Spring boot的入口main方法；

@SpringBootApplication

**public class** SpringbootWebApplication {  
  
 **public static void** main(String[] args) {  
 *//启动了springboot程序，启动spring容器，启动内嵌的tomcat* SpringApplication.*run*(SpringbootWebApplication.**class**, args);  
 }  
}

@SpringBootApplication是Spring Boot的核心注解，主要目的是开启自动配置。

### 5）创建一个Spring mvc的Controller；

默认情况下，maven从src/main/java下编译源代码，需要创建目录结构。

@RestController

**public class** HelloController {  
  
 @RequestMapping(**"/boot/hello"**)  
 **public** @ResponseBody String hello() {  
 **return "Hello Spring Boot."**;  
 }  
}

@RestController注解被称为构造型注解。在这种情况下，我们的类是一个Web@Controller，因此Spring在处理传入的Web请求时会考虑使用它。

@RequestMapping注解提供“路由”信息。它告诉Spring，任何带有/路径的HTTP请求都映射到该方法。@RestController注解告诉Spring使得到的字符串直接返回给调用者。

### 6）运行Spring boot的入口main方法。

主方法，这是遵循Java约定的应用程序入口点的标准方法。主要通过调用Spring Boot的SpringApplication类的run方法。SpringApplication会引导应用程序，并启动Spring，Spring反过来又会启动自动配置的Tomcat Web服务器。我们将类作为参数传递给run方法，以告诉SpringApplication哪个是主要的Spring组件。

### 7）创建一个可执行的Jar

创建可以在生产环境中运行的完全独立的可执行jar文件。可执行jar是包含已编译类以及代码需要运行的所有jar依赖项的归档文件。（mvn package）

## 1.2程序解析

1. Spring Boot提供了哪些默认jar包的依赖，可查看该父级依赖的pom文件；
2. 如果不想使用某个默认的依赖版本，可以通过pom.xml文件的配置属性覆盖各个依赖项，比如覆盖Spring版本：

<**properties**>  
 <**java.version**>1.8</**java.version**>  
 <**spring.version**>5.0.0.RELEASE</**spring.version**>  
</**properties**>

1. @Controller及@ResponseBody依然是我们之前的Spring mvc，因为Spring boot的里面依然是使用我们的Spring mvc+Spring+Mybatis等框架。