Springboot简单理解：

可以将springboot看做搭建程序的脚手架，最主要的作用就是帮我们快速的构建庞大的spring项目，并且尽可能减少一切xml配置，做到开箱即用，快速上手，让我们关注业务而非配置。

为什么要学习springboot

Java一直被人诟病的一点就是臃肿、麻烦。当我们还在辛苦的搭建项目时，可能python程序员已经把功能写好了，究其原因主要是以下两点：

1. 复杂的配置：项目的各种配置其实是开发时的损耗，因为在思考spring特性配置和解决业务问题之间需要进行思维切换，所以写配置挤占了写业务逻辑的时间。
2. 混乱的依赖管理：项目的依赖管理是件吃力不讨好的事，决定项目中要使用哪些库已经让人头疼，你还要确保这些库不会和其他库有冲突，依赖管理也是一种损耗，添加依赖不是写程序代码，一旦选错了以来的版本，随之而来的不兼容问题毫无疑问会是生产力杀手。

Springboot让这一切成为过去，只需要run就能创建一个独立的，生产级别的spring应用。多数springboot应用只需要很少的spring配置，并使用java -jar启动它，就能得到一个生产级别的web工程。

Springboot的特点

1. 为所有的spring开发者提供一个非常快速的、广泛接受的入门体验
2. 开箱即用，启动器starter其实就是springboot提供的一个jar包
3. 提供了一些大型项目中常见的非功能性特性，如内嵌服务器、安全、指标、健康检测、外部化配置等、帮我们监控应用状态
4. 绝对没有代码生成，也无需xml配置

使用java配置

Java配置主要靠java类和一些注解，比较常用的注解有：

1. @Configuration:声明一个类作为配置类，代替xml文件
2. @Bean：声明在方法上，将方法的返回值加入Bean容器，代替<bean>标签
3. @Value：属性注入
4. @PropertySource：指定外部属性文件

@SpringBootApplication注解干了三件事

1. 声明配置类
2. 开启自动配置（通过引入的依赖jar包做相应的自动配置）
3. 指定要扫描的包，注入bean到ioc容器

Springboot自动配置理解：

org.springframework.boot:spring-boot-autoconfiguration.jar包帮我们完成了自动配置功能，该jar包下的META-INF下的spring.factories文件指明了很多会自动配置的配置类，这些配置类中就配置了要加入ioc容器中的bean，例如有连接池相关的配置类、tomcat相关的配置类等。

开发中只需要引入相关的启动器，springboot就会帮我们默认去配置该启动器中包含的相关jar包中的bean到ioc容器中，如springmvc相关的bean、连接池相关的bean，而不需要我们像以前一样需要到xml文件中手动配置，也不需要我们去管理这些依赖及版本。Springboot为大多数的开源框架都写了启动器，我们只要直接拿来用就可以了，也可以编写自己的启动器。

Spring.factories文件中指明的配置类提供了默认配置，这些默认配置可以通过application.properties文件来进行覆盖，这样虽然还是走默认配置，但是配置中的值改成了我们自定义的（不走默认配置的话，可以手动编写配置类覆盖默认的配置类，如手动写数据库连接池的配置类）