Spring Boot 优点

- 使用JavaConfig有助于避免使用XML
- 避免大量的Maven导入和各种版本冲突
- 通过提供默认值快速开始开发
- 没有单独的Web服务器需要。这意味着你不再需要启动Tomcat，Glassfish或其他任何东西
- 更少的配置 因为没有web.xml文件。只需添加用@ Configuration注释的类，然后添加用@Bean注释的方法，Spring自动加载对象并像以前 一样对其进行管理。您甚至可以将@Autowired添加到bean方法中，以使Spring自动装入需要的依赖关系中

Spring Boot 的核心注解

启动类上面的注解是**@SpringBootApplication**，它也是 Spring Boot 的核心注解，主要组合包含了以下 3 个注解：

- @SpringBootConfiguration：组合了 @Configuration 注解，实现配置文件的功能。
- @EnableAutoConfiguration：打开自动配置的功能，也可以关闭某个自动配置的选项，如关闭数据源自动配置功能: @SpringBootApplication(exclude = { DataSourceAutoConfiguration.class })。
- @ComponentScan：Spring组件扫描

什么是 JavaConfig

Spring JavaConfig 提供了配置 Spring IoC 容器的纯Java 方法。因此它有助于避免使用 XML 配置

JavaConfig 优点

1. 面向对象的配置
    - 由于配置被定义为 JavaConfig 中的类，因此用户可以充分利用 Java 中的面向对象功能。一个配置类可以继承另一个，重写它的@Bean 方法等
2. 减少或消除 XML 配置
    - 基于依赖注入原则的外化配置的好处已被证明。但是，许多开发人员不希望在 XML 和 Java 之间来回切换。JavaConfig 为开发人员提供了一种纯 Java 方法来配置与 XML 配置概念相似的 Spring 容器。从技术角度来讲，只使用 JavaConfig 配置类来配置容器是可行的，但实际上很多人认为将JavaConfig 与 XML 混合匹配是理想的
3. 类型安全和重构友好
    - JavaConfig 提供了一种类型安全的方法来配置 Spring容器。由于 Java 5.0 对泛型的支持，现在可以按类型而不是按名称检索 bean，不需要任何强制转换或基于字符串的查找

YAML

- YAML是用来写配置文件的语言，是一种人类可读的数据序列化语言。
- 与属性文件相比，如果我们想要在配置文件中添加复杂的属性，YAML 文件就更加结构化，而且更少混淆。
- YAML 具有分层配置数据

优势

1. 配置有序，在一些特殊的场景下，配置有序很关键

2. 支持数组，数组中的元素可以是基本数据类型也可以是对象
3. 简洁

Spring Boot 的核心配置文件

- application (. yml 或者 . properties)： 由ApplicatonContext 加载，用于 spring boot 项目的自动化配置
- bootstrap (. yml 或者 . properties)：boostrap 由父 ApplicationContext 加载的，比 applicaton 优先加载，配置在应用程序上下文的引导阶段生效。一般来说我们在 Spring Cloud Config 或者 Nacos 中会用到它。且 boostrap 里面的属性不能被覆盖；

在自定义端口上运行 Spring Boot 应用程序

- 为了在自定义端口上运行 Spring Boot 应用程序，您可以在application.properties 中指定端口。server.port = 8090

Starter

- starter会把所有用到的依赖都给包含进来，避免了开发者自己去引入依赖所带来的麻烦。需要注意的是不同的starter是为了解决不同的依赖
- 一站式的获取你所需要的 Spring 和相关技术，而不需要依赖描述符的通过示例代码搜索和复制黏贴的负载
- 如果使用 Sping 和 JPA 访问数据库，只需要你的项目包含 spring-boot-starter-data-jpa 依赖项

spring-boot-starter-parent 主要作用

- 定义了 Java 编译版本为 1.8
- 使用 UTF-8 格式编码
- 继承自 spring-boot-dependencies，这个里边定义了依赖的版本，也正是因为继承了这个依赖，所以我们在写依赖时才不需要写版本号
- 执行打包操作的配置
- 自动化的资源过滤
- 自动化的插件配置
- 针对 application.properties 和 application.yml 的资源过滤，包括通过 profile 定义的不同环境的配置文件，例如 application-dev.properties 和 application-dev.yml

Spring Boot 打成的 jar 和普通的 jar 有什么区别

- Spring Boot 项目最终打包成的 jar 是可执行 jar ，这种 jar 可以直接通过 java -jar xxx.jar 命令来运行，这种 jar 不可以作为普通的 jar 被其他项目依赖，即使依赖了也无法使用其中的类。

- Spring Boot 的 jar 无法被其他项目依赖，主要还是他和普通 jar 的结构不同。普通的 jar 包，解压后直接就是包名，包里就是我们的代码，而 Spring Boot 打包成的可执行 jar 解压后，在 \BOOT-INF\classes 目录下才是我们的代码，因此无法被直接引用。如果非要引用，可以在 pom.xml 文件中增加配置，将 Spring Boot 项目打包成两个 jar ，一个可执行，一个可引用

运行 Spring Boot 有哪几种方式

1. 打包用命令或者放到容器中运行
2. 用 Maven/ Gradle 插件运行
3. 直接执行 main 方法运行