

面试了一些人，简历上都说自己熟悉 Spring Boot，或者说正在学习 Spring Boot，一问他们时，都只停留在简单的使用阶段，很多东西都不清楚，也让我对面试者大失所望。

下面，我给大家总结下有哪些 Spring Boot 的面试题，这是我经常拿来问面试者的，希望对你有帮助。

1、什么是 Spring Boot?

Spring Boot 是 Spring 开源组织下的子项目，是 Spring 组件一站式解决方案，主要是简化了使用 Spring 的难度，简省了繁重的配置，提供了各种启动器，开发者能快速上手。

更多 Spring Boot 详细介绍请看这篇文章《[什么是 Spring Boot?](#)》。

2、为什么要用 Spring Boot?

Spring Boot 优点非常多，如：

- 独立运行
- 简化配置
- 自动配置
- 无代码生成和 XML 配置
- 应用监控
- 上手容易



更多关注 Java 大后端公众号

- ...

Spring Boot 集这么多优点于一身，还有理由不使用它呢？

3、Spring Boot 的核心配置文件有哪几个？它们的区别是什么？

Spring Boot 的核心配置文件是 application 和 bootstrap 配置文件。

application 配置文件这个容易理解，主要用于 Spring Boot 项目的自动化配置。

bootstrap 配置文件有以下几个应用场景。

- 使用 Spring Cloud Config 配置中心时，这时需要在 bootstrap 配置文件中添加连接到配置中心的配置属性来加载外部配置中心的配置信息；
- 一些固定的不能被覆盖的属性；
- 一些加密/解密场景；

具体请看这篇文章《[Spring Boot 核心配置文件详解](#)》。

4、Spring Boot 的配置文件有哪几种格式？它们有什么区别？

.properties 和 .yaml，它们的区别主要是书写格式不同。

1).properties

```
app.user.name = javastack
```



更多关注 Java 大后端公众号

2).yml

```
app:
  user:
    name: javastack
```

另外，.yml 格式不支持 @PropertySource 注解导入配置。

5、Spring Boot 的核心注解是哪个？它主要由哪几个注解组成的？

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。

6、开启 Spring Boot 特性有哪几种方式？

1) 继承 spring-boot-starter-parent 项目

2) 导入 spring-boot-dependencies 项目依赖

具体请参考这篇文章《[Spring Boot 开启的 2 种方式](#)》。



更多关注 Java 大后端公众号

7、Spring Boot 需要独立的容器运行吗？

可以不需要，内置了 Tomcat/ Jetty 等容器。

8、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

9、Spring Boot 自动配置原理是什么？

注解 `@EnableAutoConfiguration`, `@Configuration`, `@ConditionalOnClass` 就是自动配置的核心，首先它得是一个配置文件，其次根据类路径下是否有这个类去自动配置。

具体看这篇文章《[Spring Boot 自动配置原理、实战](#)》。

10、Spring Boot 的目录结构是怎样的？

```
cn
+- javastack
  +- MyApplication.java
  |
  +- customer
    +- Customer.java
    +- CustomerController.java
    +- CustomerService.java
```



更多关注 Java 大后端公众号

```
|   +- CustomerRepository.java
|
+- order
    +- Order.java
    +- OrderController.java
    +- OrderService.java
    +- OrderRepository.java
```

这个目录结构是主流及推荐的做法,而在主入口类上加上 `@SpringBootApplication` 注解来开启 Spring Boot 的各项能力,如自动配置、组件扫描等。具体看这篇文章[《Spring Boot 主类及目录结构介绍》](#)。

11、你如何理解 Spring Boot 中的 Starters?

Starters 可以理解为启动器,它包含了一系列可以集成到应用里面的依赖包,你可以一站式集成 Spring 及其他技术,而不需要到处找示例代码和依赖包。如你想使用 Spring JPA 访问数据库,只要加入 `spring-boot-starter-data-jpa` 启动器依赖就能使用了。

Starters 包含了许多项目中需要用到的依赖,它们能快速持续的运行,都是一系列得到支持的管理传递性依赖。具体请看这篇文章[《Spring Boot Starters 启动器》](#)。

12、如何在 Spring Boot 启动的时候运行一些特定的代码?

可以实现接口 `ApplicationRunner` 或者 `CommandLineRunner`,这两个接口实现方式一样,它们都只提供了一个 `run` 方法,具体请看这篇文章[《Spring Boot Runner 启动器》](#)。

13、Spring Boot 有哪几种读取配置的方式?



更多关注 Java 大后端公众号

Spring Boot 可以通过 `@PropertySource`, `@Value`, `@Environment`, `@ConfigurationProperties` 来绑定变量, 具体请看这篇文章《[Spring Boot 读取配置的几种方式](#)》。

14、Spring Boot 支持哪些日志框架? 推荐和默认的日志框架是哪个?

Spring Boot 支持 Java Util Logging, Log4j2, Logback 作为日志框架, 如果你使用 Starters 启动器, Spring Boot 将使用 Logback 作为默认日志框架, 具体请看这篇文章《[Spring Boot 日志集成](#)》。

15、SpringBoot 实现热部署有哪几种方式?

主要有两种方式:

- Spring Loaded
- Spring-boot-devtools

Spring-boot-devtools 使用方式可以参考这篇文章《[Spring Boot 实现热部署](#)》。

16、你如何理解 Spring Boot 配置加载顺序?

在 Spring Boot 里面, 可以使用以下几种方式来加载配置。

1) properties 文件;



更多关注 Java 大后端公众号

2) YAML 文件;

3) 系统环境变量;

4) 命令行参数;

等等.....

具体请看这篇文章《[Spring Boot 配置加载顺序详解](#)》。

17、Spring Boot 如何定义多套不同环境配置?

提供多套配置文件, 如:

```
application.properties  
  
application-dev.properties  
  
application-test.properties  
  
application-prod.properties
```

运行时指定具体的配置文件, 具体请看这篇文章《[Spring Boot Profile 不同环境配置](#)》。

18、Spring Boot 可以兼容老 Spring 项目吗, 如何做?

可以兼容, 使用 `@ImportResource` 注解导入老 Spring 项目配置文件。

19、保护 Spring Boot 应用有哪些方法?



更多关注 Java 大后端公众号

- 在生产中使用 HTTPS
- 使用 Snyk 检查你的依赖关系
- 升级到最新版本
- 启用 CSRF 保护
- 使用内容安全策略防止 XSS 攻击
- ...

更多请看这篇文章《[10 种保护 Spring Boot 应用的绝佳方法](#)》。

20、Spring Boot 2.X 有什么新特性？与 1.X 有什么区别？

- 配置变更
- JDK 版本升级
- 第三方类库升级
- 响应式 Spring 编程支持
- HTTP/2 支持
- 配置属性绑定
- 更多改进与加强...

具体请看这篇文章《[Spring Boot 2.x 新特性总结及迁移指南](#)》。



更多关注 Java 大后端公众号

更多请关注微信公众号：Java 技术栈，回复：资料



更多关注 Java 大后端公众号