- springboot 常见面试题

Spring Boot 是一个用于简化 Spring 应用开发的框架。它提供了自动配置、依赖管理、运行时嵌入式容器等特性，帮助开发者快速构建和部署 Spring 应用。在面试中，你可能会遇到以下常见问题：

1. Spring Boot 的优势有哪些？

- 约定优于配置：提供默认配置，简化开发流程
- 快速开发：可以通过 Spring Initializer 快速生成项目模板
- 自动配置：根据项目中的依赖自动配置 Spring 应用
- 嵌入式容器：支持内置的 Tomcat、Jetty、Undertow 等 Servlet 容器
- 生产准备的特性：提供一系列生产环境所需的特性，如性能指标、健康检查等

2. Spring Boot 如何实现自动配置？

Spring Boot 使用 @EnableAutoConfiguration 注解激活自动配置，通过扫描项目中的依赖，加载相关的自动配置类。这些自动配置类通常位于 META-INF/spring.factories 文件中，并在启动时由 Spring Boot 自动加载。 3. Spring Boot 中如何自定义配置？

在 Spring Boot 中，可以通过 application.properties 或 application.yml 文件自定义配置。也可以通过使用 @ConfigurationProperties 注解将配置属性绑定到 POJO 类。 4. 什么是 Spring Boot Starter？

Spring Boot Starter 是一组便捷的依赖描述符，用于简化依赖管理。Starter 通常包含一组相关的依赖，例如，spring-boot-starter-web 会包含 Spring MVC、Tomcat 等用于开发 Web 应用的依赖。 5. Spring Boot 中如何使用 profiles？

Profiles 用于区分不同环境的配置，例如开发、测试、生产等。在 application.properties 或 application.yml 文件中，可以通过 spring.profiles.active 属性激活特定的 profile。同时，可以通过创建特定的配置文件，如 application-{profile}.properties 或 application-{profile}.yml 来为不同环境提供独立的配置。 6. Spring Boot 中如何实现 Actuator？

Actuator 是 Spring Boot 提供的生产级特性，用于监控和管理应用。要启用 Actuator，需要在项目中引入 spring-boot-starter-actuator 依赖。Actuator 提供了一系列预定义的端点，如 /health、/metrics 等，用于展示应用的运行状况和性能指标。 7. Spring Boot 中如何进行异常处理？

Spring Boot 默认提供了 ErrorController 接口，用于全局异常处理。可以实现该接口或使用 @ControllerAdvice 注解来自定义全局异常处理。另外，可以使用

@ExceptionHandler 注解在 Controller 类中处理特定的异常。 8. Spring Boot 中如何进行单元测试？

Spring Boot 提供了一套测试工具，如 SpringRunner 和@SpringBootTest

- 注解，以简化单元测试和集成测试的编写。以下是关于 Spring Boot 中测试相关的一些问题：

1. 什么是 SpringRunner？

SpringRunner 是 Spring Boot 提供的一个 JUnit 测试运行器，用于支持 Spring Boot 测试特性。在测试类上使用 @RunWith(SpringRunner.class) 注解，可以启用 SpringRunner。 2. 什么是 @SpringBootTest？

@SpringBootTest 是一个用于 Spring Boot 应用的集成测试注解，它会自动加载整个 Spring 应用上下文。使用该注解，可以方便地测试 Spring 应用的各个组件。 3. Spring Boot 中如何进行 Mock 测试？

在 Spring Boot 中，可以使用 Mockito 框架进行 Mock 测试。同时，Spring Boot 还提供了 @MockBean 和 @SpyBean 注解，用于方便地在测试中替换或包装 Spring Bean。 4. Spring Boot 中如何测试 Controller？

在 Spring Boot 中，可以使用 MockMvc 进行 Controller 测试。通过 @AutoConfigureMockMvc 注解自动配置 MockMvc 实例，并在测试方法中使用 perform() 方法模拟 HTTP 请求。 5. Spring Boot 中如何测试数据访问层（Repository）？

在 Spring Boot 中，可以使用 @DataJpaTest 注解进行数据访问层的测试。该注解会自动配置嵌入式数据库、JPA 实体管理器等组件，方便对 Repository 进行测试。 6. Spring Boot 中如何测试服务层（Service）？

在 Spring Boot 中，可以使用 @Test 注解进行服务层的单元测试。同时，可以结合 Mockito 框架对依赖的组件进行 Mock，以便对 Service 层的方法进行独立测试。 7. 如何在 Spring Boot 应用中使用缓存？

在 Spring Boot 应用中，可以通过添加 spring-boot-starter-cache 依赖并使用 @EnableCaching 注解启用缓存功能。然后可以在 Service 层的方法上使用 @Cacheable、@CachePut 和 @CacheEvict 等注解进行缓存操作。Spring Boot 支持多种缓存实现，如 EhCache、Redis、Caffeine 等。 8. 如何在 Spring Boot 应用中使用消息队列？

在 Spring Boot 应用中，可以使用 spring-boot-starter-amqp（RabbitMQ）、spring-boot-starter-kafka（Apache Kafka）等 Starter 依赖来简化消息队列的使用。Spring Boot 提供了自动配置和模板类（如 RabbitTemplate、KafkaTemplate），用于发送和接收消息。同时，可以使用 @RabbitListener 或 @KafkaListener 注解监听消息队列中的消息。