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，使开发者能快速上手。

2. 为什么要用SpringBoot

- 快速开发，快速整合，配置简化、内嵌服务容器

3. SpringBoot与SpringCloud 区别

- SpringBoot是快速开发的Spring框架，SpringCloud是完整的微服务框架，SpringCloud依赖于SpringBoot。

4. Spring Boot 有哪些优点?

- Spring Boot 主要有如下优点：
    1. 容易上手，提升开发效率，为 Spring 开发提供一个更快、更简单的开发框架。
    2. 开箱即用，远离繁琐的配置。
    3. 提供了一系列大型项目通用的非业务性功能，例如：内嵌服务器、安全管理、运行数据监控、运行状况检查和外部化配置等。
    4. SpringBoot总结就是使编码变简单、配置变简单、部署变简单、监控变简单等等

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， 例如：`java 如关闭数据源自动配置功能： @SpringBootApplication(exclude = { DataSourceAutoConfiguration.class })。`
    - @ComponentScan：Spring组件扫描。

6. 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, Lockback 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，但是不管是那种日志框架他都支持将配置文件输出到控制台或者文件中。

7. SpringBoot Starter的工作原理

- `我个人理解SpringBoot就是由各种Starter组合起来的，我们自己也可以开发Starter`
- 在sprinBoot启动时由@SpringBootApplication注解会自动去maven中读取每个starter中的spring.factories文件,该文件里配置了所有需要被创建spring容器中的bean，并且进行自动配置把bean注入SpringContext中 //（SpringContext是Spring的配置文件）

8. 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

9. SpringBoot支持什么前端模板，

- thymeleaf，freemarker，jsp，官方不推荐JSP会有限制

10. SpringBoot的缺点

- 我觉得是为难人，SpringBoot在目前我觉得没有什么缺点，非要找一个出来我觉得就是
  - 由于不用自己做的配置，报错时很难定位。

11. 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

12. 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。

13. 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

14. SpringBoot 实现热部署有哪几种方式?

- 热部署就是可以不用重新运行SpringBoot项目可以实现操作后台代码自动更新到以运行的项目中
- 主要有两种方式：
    - Spring Loaded
        - Spring-boot-devtools

15. SpringBoot事物的使用

- SpringBoot的事物很简单，首先使用注解EnableTransactionManagement开启事物之后，然后在Service方法上添加注解Transactional便可。

16. Async异步调用方法

- 在SpringBoot中使用异步调用是很简单的，只需要在方法上使用@Async注解即可实现方法的异步调用。 注意：需要在启动类加入@EnableAsync使异步调用@Async注解生效。

17. 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

18. Spring Boot 有哪几种读取配置的方式?

- Spring Boot 可以通过 @PropertySource,@Value,@Environment, @ConfigurationPropertie注解来绑定变量

19. 什么是 JavaConfig?

- Spring JavaConfig 是 Spring 社区的产品，Spring 3.0引入了他，它提供了配置 Spring IOC 容器的纯Java 方法。因此它有助于避免使用 XML 配置。使用 JavaConfig 的优点在于：
  - 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。
  - 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。
  - 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。
- 常用的Java config：
  - @Configuration：在类上打上写下此注解，表示这个类是配置类
  - @ComponentScan：在配置类上添加 @ComponentScan 注解。该注解默认会扫描该类所在的包下所有的配置类，相当于之前的 <context:component-scan >。
  - @Bean：bean的注入：相当于以前的< bean id="objectMapper" class="org.codehaus.jackson.map.ObjectMapper" />
    - @EnableWebMvc：相当于xml的<mvc:annotation-driven >
    - @ImportResource： 相当于xml的 < import resource="applicationContext-cache.xml">

20. SpringBoot的自动配置原理是什么

- 主要是Spring Boot的启动类上的核心注解SpringBootApplication注解主配置类，有了这个主配置类启动时就会为SpringBoot开启一个@EnableAutoConfiguration注解自动配置功能。
- 有了这个EnableAutoConfiguration的话就会：
  1. 从配置文件META_INF/Spring.factories加载可能用到的自动配置类
  2. 去重，并将exclude和excludeName属性携带的类排除
  3. 过滤，将满足条件（@Conditional）的自动配置类返回

21. 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；
    - 2.YAML文件；
    - 3.系统环境变量；
    - 4.命令行参数；
    - 等等……

22. 什么是 YAML?

- YAML 是一种人类可读的数据序列化语言。它通常用于配置文件。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。可以看出 YAML 具有分层配置数据。

23. YAML 配置的优势在哪里？

- YAML 现在可以算是非常流行的一种配置文件格式了，无论是前端还是后端，都可以见到 YAML 配置。那么 YAML 配置和传统的 properties 配置相比到底有哪些优势呢?
    - 配置有序，在一些特殊的场景下，配置有序很关键
    - 简洁明了，他还支持数组，数组中的元素可以是基本数据类型也可以是对象
    - 相比 properties 配置文件，YAML 还有一个缺点，就是不支持 @PropertySource 注解导入自定义的 YAML 配置。

24. Spring Boot 是否可以使用 XML 配置？

- Spring Boot 推荐使用 Java 配置而非 XML 配置，但是 Spring Boot 中也可以使用 XML 配置，通过 @ImportResource 注解可以引入一个 XML 配置。

25. spring boot 核心配置文件是什么？bootstrap.properties 和 application.properties 有何区别？

- 单纯做 Spring Boot 开发，可能不太容易遇到 bootstrap.properties 配置文件，但是在结合 Spring Cloud 时，这个配置就会经常遇到了，特别是在需要加载一些远程配置文件的时侯。
- spring boot 核心的两个配置文件：
    - 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 配置就会使用这个文件。且 boostrap 里面的属性不能被覆盖；
    - application (. yml 或者 . properties)： 由ApplicatonContext 加载，用于 spring boot 项目的自动化配置。

26. 什么是 Spring Profiles?

- 在项目的开发中，有些配置文件在开发、测试或者生产等不同环境中可能是不同的，例如数据库连接、redis的配置等等。那我们如何在不同环境中自动实现配置的切换呢？Spring给我们提供了profiles机制给我们提供的就是来回切换配置文件的功能
- Spring Profiles 允许用户根据配置文件（dev，test，prod 等）来注册 bean。因此，当应用程序在开发中运行时，只有某些 bean 可以加载，而在 PRODUCTION中，某些其他 bean 可以加载。假设我们的要求是 Swagger 文档仅适用于 QA 环境，并且禁用所有其他文档。这可以使用配置文件来完成。Spring Boot 使得使用配置文件非常简单。

27. SpringBoot多数据源拆分的思路

- 先在properties配置文件中配置两个数据源，创建分包mapper，使用@ConfigurationProperties读取properties中的配置，使用@MapperScan注册到对应的mapper包中

28. SpringBoot多数据源事务如何管理

- 第一种方式是在service层的@TransactionManager中使用transactionManager指定DataSourceConfig中配置的事务
- 第二种是使用jta-atomikos实现分布式事务管理

29. 保护 Spring Boot 应用有哪些方法?

- 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

30. 如何实现 Spring Boot 应用程序的安全性?

- 为了实现 Spring Boot 的安全性，我们使用 spring-boot-starter-security 依赖项，并且必须添加安全配置。它只需要很少的代码。配置类将必须扩展WebSecurityConfigurerAdapter 并覆盖其方法。

31. 比较一下 Spring Security 和 Shiro 各自的优缺点？

- 由于 Spring Boot 官方提供了大量的非常方便的开箱即用的 Starter ，包括 Spring Security 的 Starter ，使得在 Spring Boot 中使用 Spring Security 变得更加容易，甚至只需要添加一个依赖就可以保护所有的接口，所以，如果是 Spring Boot 项目，一般选择 Spring Security 。当然这只是一个建议的组合，单纯从技术上来说，无论怎么组合，都是没有问题的。Shiro 和 Spring Security 相比，主要有如下一些特点：
    - Spring Security 是一个重量级的安全管理框架；Shiro 则是一个轻量级的安全管理框架
    - Spring Security 概念复杂，配置繁琐；Shiro 概念简单、配置简单
    - Spring Security 功能强大；Shiro 功能简单

32. Spring Boot 中如何解决跨域问题？

- 跨域可以在前端通过 JSONP 来解决，但是 JSONP 只可以发送 GET 请求，无法发送其他类型的请求，在 RESTful 风格的应用中，就显得非常鸡肋，因此我们推荐在后端通过 （CORS，Cross-origin resource sharing） 来解决跨域问题。这种解决方案并非 Spring Boot 特有的，在传统的 SSM 框架中，就可以通过 CORS 来解决跨域问题，只不过之前我们是在 XML 文件中配置 CORS，现在可以通过实现WebMvcConfigurer接口然后重写addCorsMappings方法解决跨域问题。

```
@Configuration
public class CorsConfig implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping("/**")
                .allowedOrigins("*")
                .allowCredentials(true)
                .allowedMethods("GET", "POST", "PUT", "DELETE", "OPTIONS")
                .maxAge(3600);
    }

}
```

33. Spring Boot 中的监视器是什么?

- Spring boot actuator 是 spring 启动框架中的重要功能之一。Spring boot 监视器可帮助您访问生产环境中正在运行的应用程序的当前状态。有几个指标必须在生产环境中进行检查和监控。即使一些外部应用程序可能正在使用这些服务来向相关人员触发警报消息。监视器模块公开了一组可直接作为 HTTP URL 访问的REST 端点来检查状态。

34. 如何使用 Spring Boot 实现全局异常处理?

- Spring 提供了一种使用 ControllerAdvice 处理异常的非常有用的方法。 我们通过实现一个 ControlerAdvice 类，来处理控制器类抛出的所有异常。

35. 我们如何监视所有 Spring Boot 微服务?

- Spring Boot 提供监视器端点以监控各个微服务的度量。这些端点对于获取有关应用程序的信息（如它们是否已启动）以及它们的组件（如数据库等）是否正常运行很有帮助。但是，使用监视器的一个主要缺点或困难是，我们必须单独打开应用程序的知识点以了解其状态或健康状况。想象一下涉及 50 个应用程序的微服务，管理员将不得不击中所有 50 个应用程序的执行终端。为了帮助我们处理这种情况，我们将使用位于的开源项目。 它建立在 Spring Boot Actuator 之上，它提供了一个 Web UI，使我们能够可视化多个应用程序的度量。

36. SpringBoot性能如何优化

- 如果项目比较大，类比较多，不使用@SpringBootApplication，采用@Compoment指定扫包范围

- 在项目启动时设置JVM初始内存和最大内存相同
- 将springboot内置服务器由tomcat设置为undertow

37. 如何重新加载 Spring Boot 上的更改，而无需重新启动服务器? Spring Boot项目如何热部署?

- 这可以使用 DEV 工具来实现。通过这种依赖关系，您可以节省任何更改，嵌入式tomcat 将重新启动。Spring Boot 有一个开发工具（DevTools）模块，它有助于提高开发人员的生产力。Java 开发人员面临的一个主要挑战是将文件更改自动部署到服务器并自动重启服务器。开发人员可以重新加载 Spring Boot 上的更改，而无需重新启动服务器。这将消除每次手动部署更改的需要。Spring Boot 在发布它的第一个版本时没有这个功能。这是开发人员最需要的功能。DevTools 模块完全满足开发人员的需求。该模块将在生产环境中被禁用。它还提供 H2 数据库控制台以更好地测试应用程序。

```
<dependency>
      <groupId>org.springframework.boot</groupId>
      <artifactId>spring-boot-devtools</artifactId>
</dependency>
```

38. SpringBoot微服务中如何实现 session 共享？

- 在微服务中，一个完整的项目被拆分成多个不相同的独立的服务，各个服务独立部署在不同的服务器上，各自的 session 被从物理空间上隔离开了，但是经常，我们需要在不同微服务之间共享 session ，常见的方案就是 Spring Session + Redis 来实现 session 共享。将所有微服务的 session 统一保存在 Redis 上，当各个微服务对 session 有相关的读写操作时，都去操作 Redis 上的 session 。这样就实现了 session 共享，Spring Session 基于 Spring 中的代理过滤器实现，使得 session 的同步操作对开发人员而言是透明的，非常简便。

39. 您使用了哪些 starter maven 依赖项?

- 使用了下面的一些依赖项
  - spring-boot-starter-web 嵌入tomcat和web开发需要servlet与jsp支持
  - spring-boot-starter-data-jpa 数据库支持
  - spring-boot-starter-data-redis redis数据库支持
  - spring-boot-starter-data-solr solr支持
  - mybatis-spring-boot-starter 第三方的mybatis集成starter
    - 自定义的starter(如果自己开发过就可以说出来)

40. Spring Boot 中的 starter 到底是什么？

- 首先，这个 Starter 并非什么新的技术点，基本上还是基于 Spring 已有功能来实现的。首先它提供了一个自动化配置类，一般命名为 `XXXAutoConfiguration` ，在这个配置类中通过条件注解来决定一个配置是否生效（条件注解就是 Spring 中原本就有的），然后它还会提供一系列的默认配

置，也允许开发者根据实际情况自定义相关配置，然后通过类型安全的属性(spring.factories)注入将这些配置属性注入进来，新注入的属性会代替掉默认属性。正因为如此，很多第三方框架，我们只需要引入依赖就可以直接使用了。当然，开发者也可以自定义 Starter

41. Spring Boot 中如何实现定时任务？

- 在 Spring Boot 中使用定时任务主要有两种不同的方式，一个就是使用 Spring 中的 @Scheduled 注解，另一-个则是使用第三方框架 Quartz。
- 使用 Spring 中的 @Scheduled 的方式主要通过 @Scheduled 注解来实现。

42. spring-boot-starter-parent 有什么用？

- 我们都知道，新创建一个 Spring Boot 项目，默认都是有 parent 的，这个 parent 就是 spring-boot-starter-parent ，spring-boot-starter-parent 主要有如下作用：
    1. 定义了 Java 编译版本为 1.8 。
    2. 使用 UTF-8 格式编码。
    3. 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。
    4. 执行打包操作的配置。
    5. 自动化的资源过滤。
    6. 自动化的插件配置。
    7. 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。
- 总结就是打包用的

43. SpringBoot如何实现打包

- 进入项目目录在控制台输入mvn clean package，clean是清空已存在的项目包，package进行打包
- 或者点击左边选项栏中的Mavne，先点击clean在点击package

44.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。