

专题8：SpringBoot面试题（史上最全、定期更新）

本文版本说明：V2

此文的格式，由markdown 通过程序转成而来，由于很多表格，没有来的及调整，出现一个格式问题，尼恩在此给大家道歉啦。

由于社群很多小伙伴，在面试，不断的交流最新的面试难题，所以，《Java面试红宝书》，后面会不断升级，迭代。

本专题，作为 《Java面试红宝书》专题之一，《Java面试红宝书》一共**30个面试专题**，后续还会增加

《Java面试红宝书》升级的规划为：

后续基本上，**每一个月，都会发布一次**，最新版本，可以扫描扫架构师尼恩微信，发送“领取电子书”获取。

尼恩的微信二维码在哪里呢？请参见文末

面试问题交流说明：

如果遇到面试难题，或者职业发展问题，或者中年危机问题，都可以来 疯狂创客圈社群交流，加入交流群，加尼恩微信即可，

SpringBoot 概述

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。

Spring Boot、Spring MVC 和 Spring 有什么区别？

SpringFrame

SpringFramework 最重要的特征是依赖注入。所有 SpringModules 不是依赖注入就是 IOC 控制反转。

当我们恰当的使用 DI 或者是 IOC 的时候，我们可以开发松耦合应用。松耦合应用的单元测试可以很容易的进行。

SpringMVC

Spring MVC 提供了一种分离式的方法来开发 Web 应用。通过运用像 DispatcherServlet, MoudAndView 和 ViewResolver 等一些简单的概念，开发 Web 应用将会变的非常简单。

SpringBoot

Spring 和 SpringMVC 的问题在于：需要配置大量的参数。

```
<pre style="box-sizing: inherit; font-family: Menlo, Monaco, Consolas,
'Courier New', monospace; font-size: 13px; word-break: initial; word-
wrap: initial; white-space: pre; overflow: auto; margin: 16px 0px 14px; padding:
14px 15px 12px; border-radius: 3px; border: none; display: block; line-height:
1.6; background: rgb(246, 246, 246); color: rgb(61, 70, 77); font-style: normal;
font-variant-ligatures: normal; font-variant-caps: normal; font-weight: 400;
letter-spacing: normal; orphans: 2; text-align: start; text-indent: 0px; text-
transform: none; widows: 2; word-spacing: 0px; -webkit-text-stroke-width: 0px;
text-decoration-style: initial; text-decoration-color: initial;"><bean
  class="org.springframework.web.servlet.view.InternalResourceViewResolver">
<property name="prefix">
  <value>/WEB-INF/views/</value>
</property>
<property name="suffix">
  <value>.jsp</value>
</property>
</bean>
<mvc:resources mapping="/webjars/**" location="/webjars/"></pre>
```

Spring Boot 通过一个自动配置和启动的项来解决这个问题。为了更快的构建产品就绪应用程序，Spring Boot 提供了一些非功能性特征。

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、更广泛的入门体验。
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. 没有代码生成，也不需要XML配置。
5. 避免大量的 Maven 导入和各种版本冲突。

为什么要用SpringBoot？

- 为了解决java开发中的，繁多的配置、底下的开发效率，复杂的部署流程，和第三方技术集成难度大的问题，产生了spring boot。
- springboot 使用“习惯优于配置”的理念让项目快速运行起来，使用springboot很容易创建一个独立运行的jar，内嵌servlet容器
- springboot的核心功能一：独立运行spring项目，springboot可以以jar包的形式独立运行，运行一个springboot项目只需要 java -jar xxx.jar 来运行

- springboot的核心功能二：内嵌servlet容器，可以内嵌tomcat，接天jetty，或者undertow，这样我们就可以不用war包形式部署项目
- springboot的核心功能三，提供starter简化maven配置，spring提供了一系列starter pom 来简化maven的依赖加载，当使用了 spring-boot-starter-web时，会自动加载所需要的依赖包
- springboot的核心功能三：自动配置spring sprintboot 会根据在类路径的jar包，类，为jar包中的类自动配置bean，这样会极大的减少使用的配置，会根据启动类所在的目录，自动配置bean

什么是 Spring Boot Stater ?

启动器是一套方便的依赖没描述符，它可以放在自己的程序中。你可以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。

例如，如果你想使用 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，你就可以完美进行。

Spring Boot 还提供了其它的哪些 Starter 选项?

Spring Boot 也提供了其它的启动器项目包括，包括用于开发特定类型应用程序的典型依赖项。

spring-boot-starter-web-services - SOAP Web Services

spring-boot-starter-web - Web 和 RESTful 应用程序

spring-boot-starter-test - 单元测试和集成测试

spring-boot-starter-jdbc - 传统的 JDBC

spring-boot-starter-hateoas - 为服务添加 HATEOAS 功能

spring-boot-starter-security - 使用 SpringSecurity 进行身份验证和授权

spring-boot-starter-data-jpa - 带有 Hibeernate 的 Spring Data JPA

spring-boot-starter-data-rest - 使用 Spring Data REST 公布简单的 REST 服务

Spring 是如何快速创建产品就绪应用程序的?

Spring Boot 致力于快速产品就绪应用程序。为此，它提供了一些譬如高速缓存，日志记录，监控和嵌入式服务器等开箱即用的非功能性特征。

spring-boot-starter-actuator - 使用一些如监控和跟踪应用的高级功能

spring-boot-starter-undertow, spring-boot-starter-jetty, spring-boot-starter-tomcat - 选择您的特定嵌入式 Servlet 容器

spring-boot-starter-logging - 使用 logback 进行日志记录

spring-boot-starter-cache - 启用 Spring Framework 的缓存支持

Spring2 和 Spring5 所需要的最低 Java 版本是什么?

Spring Boot 2.0 需要 Java8 或者更新的版本。Java6 和 Java7 已经不再支持。

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。

自动配置

什么是 JavaConfig？

Spring JavaConfig 是 Spring 社区的产品，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：

(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 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。

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，

@EnableAutoConfiguration 给容器导入 META-INF/spring.factories 里定义的自动配置类。

筛选有效的自动配置类。

每一个自动配置类结合对应的 xxxProperties.java 读取配置文件进行自动配置功能

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.....

我们可以在 Spring Beans 里面直接使用这些配置文件中加载的值, 如:

- 1、使用 @Value 注解直接注入对应的值, 这能获取到 Spring 中 Environment 的值;
- 2、使用 @ConfigurationProperties 注解把对应的值绑定到一个对象;
- 3、直接获取注入 Environment 进行获取;

配置属性的方式很多, Spring boot使用了一种独有的 PropertySource 可以很方便的覆盖属性的值。

配置属性加载的顺序如下:

- 1、开发者工具 Devtools 全局配置参数;
- 2、单元测试上的 `@TestPropertySource` 注解指定的参数;
- 3、单元测试上的 `@SpringBootTest` 注解指定的参数;
- 4、命令行指定的参数, 如 `java -jar springboot.jar --name="Java技术栈"`;
- 5、命令行中的 `SPRING_APPLICATION_JSON` 指定参数, 如 `java -Dspring.application.json='{ "name": "Java技术栈" }' -jar springboot.jar`
- 6、`ServletConfig` 初始化参数;
- 7、`ServletContext` 初始化参数;
- 8、JNDI参数 (如 `java:comp/env/spring.application.json`) ;
- 9、Java系统参数 (来源: `System.getProperties()`) ;
- 10、操作系统环境变量参数;
- 11、`RandomValuePropertySource` 随机数, 仅匹配: `random.*`;
- 12、JAR包外面的配置文件参数 (`application-{profile}.properties (YAML)`)
- 13、JAR包里面的配置文件参数 (`application-{profile}.properties (YAML)`)
- 14、JAR包外面的配置文件参数 (`application.properties (YAML)`)
- 15、JAR包里面的配置文件参数 (`application.properties (YAML)`)
- 16、`@Configuration` 配置文件上 `@PropertySource` 注解加载的参数;
- 17、默认参数 (通过 `SpringApplication.setDefaultProperties` 指定) ;

数字小的优先级越高, 即数字小的会覆盖数字大的参数值, 我们来实践下, 验证以上配置参数的加载顺序。

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。

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？

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

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。

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。

eg:

```
@ImportResource({"classpath:some-context.xml","classpath:another-context.xml"})
```

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候。

spring boot 核心的两个配置文件：

- bootstrap (.yml 或者 .properties)：bootstrap 由父 ApplicationContext 加载的，比 application 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 bootstrap 里面的属性不能被覆盖；
- application (.yml 或者 .properties)：由 ApplicationContext 加载，用于 spring boot 项目的自动化配置。

什么是 Spring Profiles？

Spring Profiles 允许用户根据不同配置文件（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。

如何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
```

安全

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。

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：

1. 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
2. Spring Security 概念复杂，配置繁琐；Shiro 概念简单、配置简单
3. Spring Security 功能强大；Shiro 功能简单

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);
    }
}
```

项目中前后端分离部署，所以需要解决跨域的问题。

我们使用 cookie 存放用户登录的信息，在 spring 拦截器进行权限控制，当权限不符合时，直接返回给用户固定的 json 结果。

当用户登录以后，正常使用；当用户退出登录状态时或者 token 过期时，由于拦截器和跨域的顺序有问题，出现了跨域的现象。

我们知道一个http请求，先走filter，到达servlet后才进行拦截器的处理，如果我们把cors放在filter里，就可以优先于权限拦截器执行。

```
@Configuration
public class CorsConfig {

    @Bean
    public CorsFilter corsFilter() {
        CorsConfiguration corsConfiguration = new CorsConfiguration();
        corsConfiguration.addAllowedOrigin("*");
        corsConfiguration.addAllowedHeader("*");
        corsConfiguration.addAllowedMethod("*");
        corsConfiguration.setAllowCredentials(true);
        UrlBasedCorsConfigurationSource urlBasedCorsConfigurationSource = new
        UrlBasedCorsConfigurationSource();
        urlBasedCorsConfigurationSource.registerCorsConfiguration("/**",
        corsConfiguration);
        return new CorsFilter(urlBasedCorsConfigurationSource);
    }

}
```

什么是 CSRF 攻击？

CSRF 代表跨站请求伪造。这是一种攻击，迫使最终用户在当前通过身份验证的Web 应用程序上执行不需要的操作。CSRF 攻击专门针对状态改变请求，而不是数据窃取，因为攻击者无法查看对伪造请求的响应。

监视器

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。

如何在 Spring Boot 中禁用 Actuator 端点安全性？

默认情况下，所有敏感的 HTTP 端点都是安全的，只有具有 ACTUATOR 角色的用户才能访问它们。安全性是使用标准的 HttpServletRequest.isUserInRole 方法实施的。我们可以使用来禁用安全性。只有在执行机构端点在防火墙后访问时，才建议禁用安全性。

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。

前后端分离，如何维护Rest API接口文档？

前后端分离开发日益流行，大部分情况下，我们都是通过 Spring Boot 做前后端分离开发，前后端分离一定会有接口文档，不然会前后端会深深陷入到扯皮中。一个比较笨的方法就是使用 word 或者 md 来维护接口文档，但是效率太低，接口一变，所有人手上的文档都得变。在 Spring Boot 中，这个问题常见的解决方案是 Swagger，使用 Swagger 我们可以快速生成一个接口文档网站，接口一旦发生变化，文档就会自动更新，所有开发工程师访问这一个在线网站就可以获取到最新的接口文档，非常方便。

关于 Swagger 的用法，大家可以参考：SpringBoot整合Swagger2，再也不用维护接口文档了！

整合第三方项目

什么是 WebSockets？

WebSocket 是一种计算机通信协议，通过单个 TCP 连接提供全双工通信信道。

- 1、WebSocket 是双向的 -使用 WebSocket 客户端或服务器可以发起消息发送。
- 2、WebSocket 是全双工的 -客户端和服务端通信是相互独立的。
- 3、单个 TCP 连接 -初始连接使用 HTTP，然后将此连接升级到基于套接字的连接。然后这个单一连接用于所有未来的通信
- 4、Light -与 http 相比，WebSocket 消息数据交换要轻得多。

什么是 Spring Data ？

Spring Data 是 Spring 的一个子项目。用于简化数据库访问，支持NoSQL 和 关系数据存储。其主要目标是使数据库的访问变得方便快捷。Spring Data 具有如下特点：

SpringData 项目支持 NoSQL 存储：

1. MongoDB （文档数据库）
2. Neo4j （图形数据库）
3. Redis （键/值存储）
4. Hbase （列族数据库）

SpringData 项目所支持的关系数据存储技术：

1. JDBC
2. JPA

Spring Data Jpa 致力于减少数据访问层 (DAO) 的开发量. 开发者唯一要做的，就是声明持久层的接口，其他都交给 Spring Data JPA 来帮你完成！Spring Data JPA 通过规范方法的名字，根据符合规范的名字来确定方法需要实现什么样的逻辑。

什么是 Spring Batch？

Spring Boot Batch 提供可重用的函数，这些函数在处理大量记录时非常重要，包括日志/跟踪，事务管理，作业处理统计信息，作业重新启动，跳过和资源管理。它还提供了更先进的技术服务和功能，通过优化和分区技术，可以实现极高批量和高性能批处理作业。简单以及复杂的大批量批处理作业可以高度可扩展的方式利用框架处理重要大量的信息。

什么是 FreeMarker 模板？

FreeMarker 是一个基于 Java 的模板引擎，最初专注于使用 MVC 软件架构进行动态网页生成。使用 FreeMarker 的主要优点是表示层和业务层的完全分离。程序员可以处理应用程序代码，而设计人员可以处理 html 页面设计。最后使用 freemarker 可以将这些结合起来，给出最终的输出页面。

如何集成 Spring Boot 和 ActiveMQ?

对于集成 Spring Boot 和 ActiveMQ，我们使用依赖关系。它只需要很少的配置，并且不需要样板代码。

什么是 Apache Kafka?

Apache Kafka 是一个分布式发布 - 订阅消息系统。它是一个可扩展的，容错的发布 - 订阅消息系统，它使我们能够构建分布式应用程序。这是一个 Apache 顶级项目。Kafka 适合离线和在线消息消费。

什么是 Swagger? 你用 Spring Boot 实现了它吗?

Swagger 广泛用于可视化 API，使用 Swagger UI 为前端开发人员提供在线沙箱。Swagger 是用于生成 RESTful Web 服务的可视化表示的工具，规范和完整框架实现。它使文档能够以与服务器相同的速度更新。当通过 Swagger 正确定义时，消费者可以使用最少量的实现逻辑来理解远程服务并与其进行交互。因此，Swagger 消除了调用服务时的猜测。

前后端分离，如何维护接口文档？

前后端分离开发日益流行，大部分情况下，我们都是通过 Spring Boot 做前后端分离开发，前后端分离一定会有接口文档，不然会前后端会深深陷入到扯皮中。一个比较笨的方法就是使用 word 或者 md 来维护接口文档，但是效率太低，接口一变，所有人手上的文档都得变。在 Spring Boot 中，这个问题常见的解决方案是 Swagger，使用 Swagger 我们可以快速生成一个接口文档网站，接口一旦发生变化，文档就会自动更新，所有开发工程师访问这一个在线网站就可以获取到最新的接口文档，非常方便。

其他

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-activemq

spring-boot-starter-security

这有助于增加更少的依赖关系，并减少版本的冲突。

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

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。

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

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。

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

如何使用 Spring Boot 实现异常处理？

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 ControllerAdvice 类，来处理控制器类抛出的所有异常。

如何使用 Spring Boot 实现分页和排序？

使用 Spring Boot 实现分页非常简单。使用 Spring Data-JPA 可以实现将可分页的传递给存储库方法。

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。

Spring Boot 中如何实现定时任务？

定时任务也是一个常见的需求，Spring Boot 中对于定时任务的支持主要还是来自 Spring 框架。

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。

使用 Quartz，则按照 Quartz 的方式，定义 Job 和 Trigger 即可。

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。session 共享大家可以参考：Spring Boot 一个依赖搞定 session 共享，没有比这更简单的方案了！

打包部署

什么是嵌入式服务器？我们为什么要使用嵌入式服务器呢？

思考一下在你的虚拟机上部署应用程序需要些什么。

第一步：安装 Java

第二步：安装 Web 或者是应用程序的服务器（Tomcat/Wbesphere/Weblogic 等等）

第三步：部署应用程序 war 包

如果我们想简化这些步骤，应该如何做呢？

让我们来思考如何使服务器成为应用程序的一部分？

你只需要一个安装了 Java 的虚拟机，就可以直接在上面部署应用程序了，
是不是很爽？

这个想法是嵌入式服务器的起源。

当我们创建一个可以部署的应用程序的时候，我们将会把服务器（例如，tomcat）嵌入到可部署的服务器中。

例如，对于一个 Spring Boot 应用程序来说，你可以生成一个包含 Embedded Tomcat 的应用程序 jar。你就可以想运行正常 Java 应用程序一样来运行 web 应用程序了。

嵌入式服务器就是我们的可执行单元包含服务器的二进制文件（例如，tomcat.jar）。

Spring Boot 如何实现热部署？

Spring Boot 实现热部署其实很容易，引入 devtools 依赖即可，这样当编译文件发生变化时，Spring Boot 就会自动重启。在 Eclipse 中，用户按下保存按键，就会自动编译进而重启 Spring Boot，IDEA 中由于是自动保存的，自动保存时并未编译，所以需要开发者按下 Ctrl+F9 进行编译，编译完成后，项目就自动重启了。

如果仅仅是页面模板发生变化，Java 类并未发生变化，此时可以不用重启 Spring Boot，使用 LiveReload 插件就可以轻松实现热部署。

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>
```

为什么我们需要 spring-boot-maven-plugin?

spring-boot-maven-plugin 提供了一些像 jar 一样打包或者运行应用程序的命令。

- spring-boot:run 运行你的 SpringBooty 应用程序。
- spring-boot: repackage 重新打包你的 jar 包或者是 war 包使其可执行
- spring-boot: start 和 spring-boot: stop 管理 Spring Boot 应用程序的生命周期（也可以说是为了集成测试）。
- spring-boot:build-info 生成执行器可以使用的构造信息。

如何使用 Spring Boot 部署到不同的服务器？

你需要做下面两个步骤：

- 在一个项目中生成一个 war 文件。
- 将它部署到你最喜欢的服务器（websphere 或者 Weblogic 或者 Tomcat and so on）。

第二步：取决于你的服务器。

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。

硬核推荐：尼恩Java硬核架构班

又名疯狂创客圈社群 VIP

详情：

<https://www.cnblogs.com/crazymakercircle/p/9904544.html>



尼恩java 硬核架构班





定价19999 / 早鸟 3999
即将涨价 4999

已经发布

- 《高性能RPC的基础实操之：从0到1开始IM推一个IM》
- 《分布式高性能RPC的基础实操之：千万级用户分布式IM实操-含简历指导》
- 《亿级用户超高并发秒杀实操-含简历指导》
亮点：助力小伙伴搞定70W年薪，N个涨薪50%，2023春招面试涨薪神器
- 《横扫全网，工业级elasticsearch底层原理与高并发、高可用架构实操》
亮点：40岁老架构师细致解读，处处透着分布式、高性能中间件的原理和精髓
- 《第1部曲：超级底层：葵花宝典（高性能秘籍）__架构师视角解读OS操作系统》
亮点：大制作解读OS操作系统，并揭秘mmap、pagecache、zerocopy等底层的底层原理
2023春招面试涨薪大神器
- 《Rocketmq视频第2部曲：横扫全网工业级 rocketmq 高可用（HA）底层原理和实操》
亮点：起底式、较杀式解读 rocketmq如何保障消息的可靠性？
- 《Rocketmq视频第3部曲：超级内功篇、横扫全网 rocketmq 源码学习以及3高架构模式解读》
亮点：大制作解读 Rocketmq源码以及3高架构模式，助力大家内力猛增
- 《Rocketmq视频第4部曲：10Wqps消息推送中台架构、设计、编码、测试实操》
亮点：Netty实操、分库分表实操、Rocketmq工业级使用实操
- 《架构师内功篇：横扫全网 netty 高性能、高并发架构 底层原理、源码学习》
- 《架构师实操篇：redis cluster 工业级高可用实操》
- 《架构师实操篇：100W级别QPS日志平台实操》

规划中

- 《彻底穿透：skywalking 源码（代表链路跟踪）+ Java agent + bytebuddy 探针》
- 《架构师实操篇：基于netty 手写 rpc 框架-参考 dubbo、seata rpc框架》
- 《架构师实操篇：go语言学习，以及基于 go 手写 rpc 框架》
- 《架构师实操篇：千万级任务调度平台 架构与实操-基于尼恩17年的亿级搜索项目》
- 《架构师实操篇：工业级 亿级文档搜索 平台 架构与实操-基于尼恩17年的亿级搜索项目》

特色

会员制

提供技术方向指导，
职业生涯指导，少坑，少弯路

简历指导

这个很重要，
对于涨薪来说

实操性

以上项目，都是老架构师
在生产上实操过的项目

非水货

40岁老架构师，不是水货架构师
《Java高并发三部曲》为证

架构班（社群 VIP）的起源：

最初的视频，主要是给读者加餐。很多的读者，需要一些高质量的实操、理论视频，所以，我就围绕书，和底层，做了几个实操、理论视频，然后效果还不错，后面就做成迭代模式了。

架构班（社群 VIP）的功能：

提供高质量实操项目整刀真枪的架构指导、快速提升大家的：

- 开发水平
- 设计水平
- 架构水平

弥补业务中 CRUD 开发短板，帮助大家尽早脱离具备 3 高能力，掌握：

- 高性能
- 高并发
- 高可用

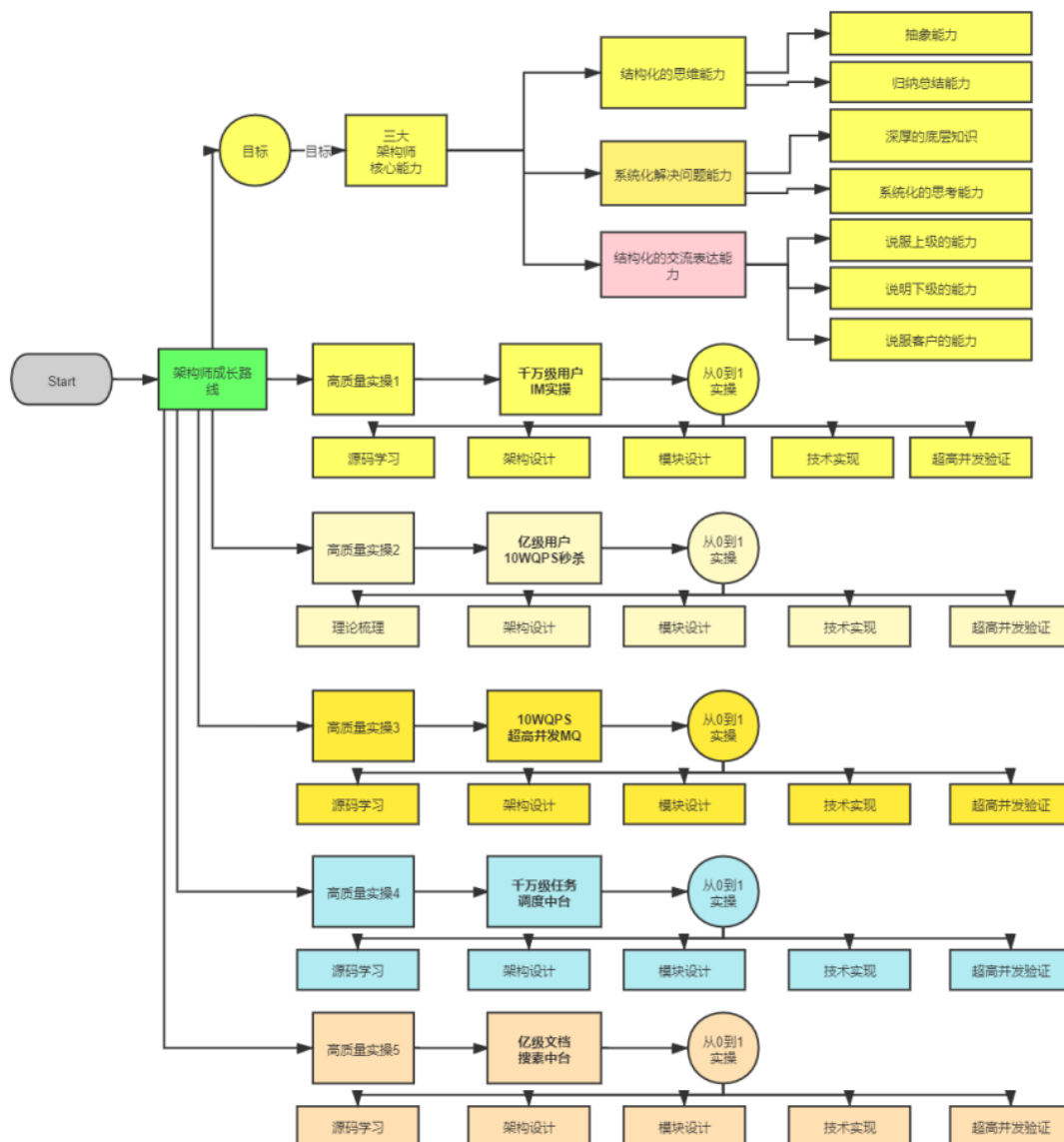
作为一个高质量的架构师成长、人脉社群，把所有的卷王聚焦起来，一起卷：

- 卷高并发实操
- 卷底层原理
- 卷架构理论、架构哲学
- 最终成为顶级架构师，实现人生理想，走向人生巅峰

架构班（社群 VIP）的目的：

- 高质量的实操，大大提升简历的含金量，吸引力，增强面试的召唤率
- 为大家提供九阳真经、葵花宝典，快速提升水平
- 进大厂、拿高薪
- 一路陪伴，提供助学视频和指导，辅导大家成为架构师
- 自学为主，和其他卷王一起，卷高并发实操，卷底层原理、卷大厂面试题，争取狠卷 3 月成高手，狠卷 3 年成为顶级架构师

N 个超高并发实操项目：简历压轴、个顶个精彩



【样章】第 17 章:横扫全网Rocketmq 视频第 2 部曲: 工业级 rocketmq 高可用(HA) 底层原理和实操

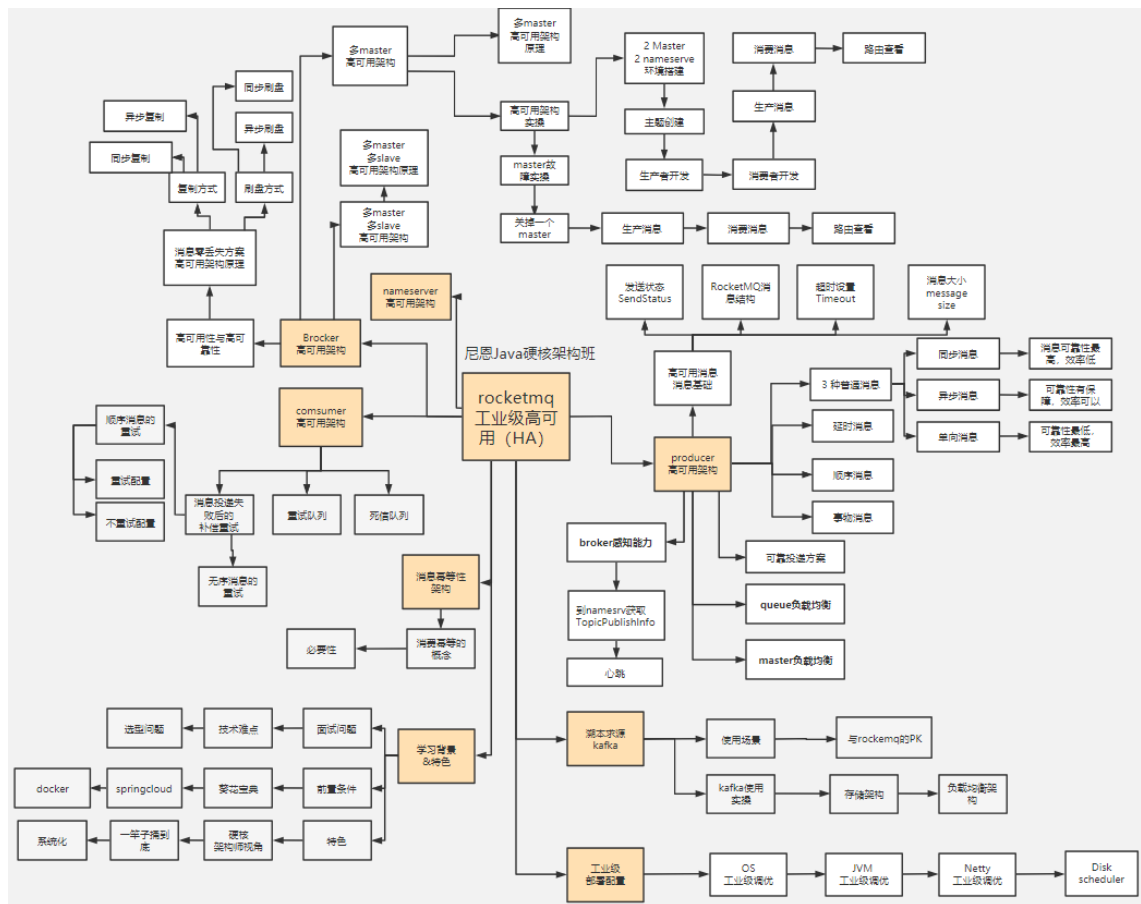
工业级 rocketmq 高可用底层原理, 包含: 消息消费、同步消息、异步消息、单向消息等不同消息的底层原理和源码实现; 消息队列非常底层的主从复制、高可用、同步刷盘、异步刷盘等底层原理。

工业级 rocketmq 高可用底层原理和搭建实操, 包含: 高可用集群的搭建。

解决以下难题:

- 1、技术难题: RocketMQ 如何最大限度的保证消息不丢失的呢? RocketMQ 消息如何做到高可靠投递?
- 2、技术难题: 基于消息的分布式事务, 核心原理不理解
- 3、选型难题: kafka or rocketmq , 该娶谁?

下图链接: <https://www.proceson.com/view/6178e8ae0e3e7416bde9da19>



成功案例：2 年翻 3 倍，35 岁卷王成功转型为架构师

详情：<http://topcoder.cloud/forum.php?mod=forumdisplay&fid=43&page=1>

最新	最后发表	热门	精华	最新	最后发表	热门	精华
 成功案例：[1057号卷王] 3年小伙拿到外企offer，薪酬涨了200%	 卷王1号	超级版主	前天 17:41	 成功案例：[693号卷王] 二线城市6年卷王喜提4大优质Offer，含央企offer，最高薪酬35W	 卷王1号	超级版主	2022-4-16
 成功案例：[645号卷王] 4年经验卷王逆袭，被毕业后，反涨24W	 卷王1号	超级版主	2022-9-21	 成功案例：[85号卷王] 双非2本小伙，春招大捷，喜提9个offer，最高薪酬近30万	 卷王1号	超级版主	2022-4-14
 成功案例：[878号卷王] 小伙8年经验，年薪60W	 卷王1号	超级版主	2022-8-13	 成功案例：[741号卷王] 卷王逆袭！6年小伙从很少面试机会到搞定35K*14薪Offer	 卷王1号	超级版主	2022-4-12
 年薪70W案例：通过尼恩的指导，小伙伴年薪从40W涨到70W	 卷王1号	超级版主	2022-2-11	 成功案例：[642号卷王] 热烈祝贺，6年卷王喜提优质国企offer	 卷王1号	超级版主	2022-4-7
 成功案例：[493号卷王] 5年小伙拿满意offer，就业寒冬逆势涨30%	 卷王1号	超级版主	前天 17:43	 成功案例：[796号卷王] 热烈祝贺，36岁卷王喜提52万优质offer	 卷王1号	超级版主	2022-3-25
 成功案例：[250号卷王] 就业极寒时代，收offer 涨25%	 卷王1号	超级版主	前天 17:38	 成功案例：[15号卷王] 小伙卷1年，涨薪9K+，喜收ebay等多个优质offer	 卷王1号	超级版主	2022-3-24
 成功案例：[612号卷王] 就业极寒时代，从外包到白研	 卷王1号	超级版主	前天 17:15	 成功案例：[821号卷王] 小伙狠卷3个月，喜提10多个offer	 卷王1号	超级版主	2022-3-21
 成功案例：[913号卷王] 热烈祝贺6年经验卷王，年薪40W	 卷王1号	超级版主	2022-9-21	 成功案例：[736号卷王] 3年半经验收22k offer，但是小伙志存高远，冲击25k+	 卷王1号	超级版主	2022-3-20
 成功案例：[959号卷王] 4年经验卷王，喜获百度、Boss直聘等N个优质offer，最高涨100%	 卷王1号	超级版主	2022-9-21	 成功案例：热烈祝贺一群小卷王offer拿到手软，甚至拒了阿里offer	 卷王1号	超级版主	2022-3-16
 成功案例：[529号卷王] 5年经验卷王喜收2大offer，最高涨5K	 卷王1号	超级版主	2022-9-21	 简历案例：简历一改，腾讯的邀请就来！热烈祝贺，小伙收到一大堆面试邀请	 卷王1号	超级版主	2022-3-10
 成功案例：[811号卷王] 热烈祝贺7年经验卷王，薪酬涨30%	 卷王1号	超级版主	2022-9-21	 成功案例：祝贺我圈两大超级卷王，一个过了阿里HR面，一个过了阿里2面	 卷王1号	超级版主	2022-3-10
 成功案例：[287号卷王] 不惧大寒潮，卷王逆市收4 offer，涨30%，可喜可贺	 卷王1号	超级版主	2022-5-30	 成功案例：小伙伴php转Java，卷1.5年Java，涨薪50%，喜收多个优质offer	 卷王1号	超级版主	2022-3-10
 成功案例：[1002号卷王] 5月份“被毕业”，改简历后，斩获顶级央企Offer，涨薪7000+	 卷王1号	超级版主	2022-7-5	 成功案例：4年小伙狠卷半年，拿到 移动、京东 两大顶级offer	 卷王1号	超级版主	2022-3-5
 成功案例：[7号卷王] 热烈祝贺小伙伴涨薪120%	 卷王1号	超级版主	2022-8-13	 成功案例：[267号卷王] 助力3年经验卷王，拿到蜂巢的17k x 14薪的offer	 卷王1号	超级版主	2022-2-27
 成功案例：[134号卷王] 大三小伙卷1年，斩获顶级央企Offer，成功逆袭	 卷王1号	超级版主	2022-7-6	 成功案例：[143号卷王] 二本院校00后卷神，毕业没到一年跳到字节，年薪45W	 卷王1号	超级版主	2022-2-27
 成功案例：[1008号卷王] 5年经验卷王收42W offer，月涨8000，可喜可贺	 卷王1号	超级版主	2022-5-30	 成功案例：[494号卷王] 尼恩分布式事务助力卷王拿到 中信银行offer	 卷王1号	超级版主	2022-2-27
 成功案例：[453号卷王] 非全日制 6年卷王喜提3 offer，年薪30W，可喜可贺	 卷王1号	超级版主	2022-5-21	 成功案例：[76号卷王] 2线城市卷王，狠卷1.5年，喜收22K offer	 卷王1号	超级版主	2022-2-27
 成功案例：[924号卷王] 6年卷王喜提4 offer，最高涨薪9000，可喜可贺	 卷王1号	超级版主	2022-5-21	 成功案例：[429号卷王] 小伙伴在社群卷5个月，涨8k+	 卷王1号	超级版主	2022-2-27
 成功案例：[15号卷王] 4年卷王入职 微软，涨薪50%，可喜可贺	 卷王1号	超级版主	2022-5-12	 成功案例：[154号卷王] 双非学校毕业卷王，连拿 京东到家&滴滴 两个大厂Offer	 卷王1号	超级版主	2022-2-27
 成功案例：[527号卷王] 4年卷王喜提2 offer，涨薪50%，可喜可贺	 卷王1号	超级版主	2022-5-13	 成功案例：[232号卷王] 涨薪10K，继续卷向食物链顶端	 卷王1号	超级版主	2022-2-27
 成功案例：[788号卷王] 3年卷王喜提优质Offer，涨薪60%	 卷王1号	超级版主	2022-5-11	 成功案例：狠卷1年技术，喜收 腾讯、阿里、微软 三大Offer，最高年薪56W	 卷王1号	超级版主	2022-2-27
 成功案例：热烈祝贺：非全日制卷王，喜提2个心仪offer，面3家过2家	 卷王1号	超级版主	2022-4-21	 成功案例：[449号卷王] 应届毕业卷王喜收 滴滴offer，年薪33W	 卷王1号	超级版主	2022-2-27
 成功案例：[732号卷王] 尼恩助力3年经验卷王收获 京东offer，年薪35W	 卷王1号	超级版主	2022-2-27	 成功案例：[551号卷王] 小伙伴学完后，成功进入大厂，并且推荐自己的朋友加VIP学习	 卷王1号	超级版主	2022-2-10
 成功案例：[558号卷王] 2年经验卷王，喜收 网易和阿里子公司两个优质offer	 卷王1号	超级版主	2022-2-27	 成功案例：[214号卷王] 助力2年经验卷王，成功拿到17K月薪	 卷王1号	超级版主	2022-2-10
 成功案例：[569号卷王] 双非应届卷王，喜收字节跳动实习offer	 卷王1号	超级版主	2022-2-25	 成功案例：[92号卷王] 课程实操助力社群小伙伴喜收 喜马拉雅Offer	 卷王1号	超级版主	2022-2-10
 成功案例：[420号卷王] 狠卷1年，卷王涨薪80%，涨薪12000元！	 卷王1号	超级版主	2022-2-25	 成功案例：社群卷王小伙伴成功过了滴滴三面 获滴滴Offer	 卷王1号	超级版主	2022-2-10
 成功案例：[76号卷王] 通过尼恩1年半的指导，专科学历小伙伴从0.8K涨到22K	 卷王1号	超级版主	2022-2-10	 [612号卷王]滴滴小伙伴，蹲点考察半年，觉得靠谱后加入 疯狂创客圈	 卷王1号	超级版主	2022-2-10

简历优化后的成功涨薪案例 (VIP 含免费简历优化)

简历优化，卷王逆袭部分成功案例

The following table summarizes the 20 successful salary increase cases shown in the grid:

Case Title	Timeline	Outcome
小伙8年经验 年薪60W	7月12日改简历, 8月10日接offer	涨薪: 改简历 + 新Offer
7年经验卷王 薪酬涨30%	7月11日改简历, 9月1日接offer	涨薪: 改简历 + 新Offer
4年经验卷王逆袭 被毕业后, 反涨24W	7月改简历, 8月30日接offer	涨薪: 改简历 + 新Offer
小伙5月份"被毕业", 改简历后 新获顶级央企Offer 涨薪7000+	5月29日改简历, 7月5日接offer	涨薪: 改简历 + 新Offer
5年卷王喜收2大Offer 最高涨5K	5月19日改简历, 9月13日接offer	涨薪: 改简历 + 新Offer
6年小伙伴 年薪40W	9月6日改简历, 9月21日接offer	涨薪: 改简历 + 新Offer
卷王逆袭成功案例 6年小伙从很少面试机会到 搞定35K*14薪	3月9日改简历, 4月11日接offer	涨薪: 改简历 + 新Offer
卷王逆袭成功案例 武汉6年喜收4个优质offer 最高的年薪35W	2月9日改简历, 4月15日接offer	涨薪: 改简历 + 新Offer
卷王逆袭成功案例 6年小伙喜提4个Offer 最高涨9k, 年薪35W	4月14日改简历, 5月17日接offer	涨薪: 改简历 + 新Offer
卷王逆袭成功案例 5年经验小伙收2个offer 最高涨薪8k, 年薪42W	5月9日改简历, 5月30日接offer	涨薪: 改简历 + 新Offer
小伙高中学历 薪酬涨120%	5月6日改简历, 7月22日接offer	涨薪: 改简历 + 新Offer
卷王逆袭成功案例 寒冬冻六之际卷王大逆袭 收3大offer, 涨30%	5月17日改简历, 5月27日接offer	涨薪: 改简历 + 新Offer
卷王逆袭成功案例 4年卷王入职微软, 涨50%	3月7日改简历, 5月12日接offer	涨薪: 改简历 + 新Offer
4年小伙喜收百度、Boss直聘 等N个顶级Offer 最高涨幅100%	6月27日改简历, 9月19日接offer	涨薪: 改简历 + 新Offer
卷王逆袭成功案例 4年卷王入收2个offer, 涨50%	3月23日改简历, 5月12日接offer	涨薪: 改简历 + 新Offer
卷王逆袭成功案例 非全日制卷王 面试3家 收2个offer 涨薪30%	4月13日改简历, 4月21日接offer	涨薪: 改简历 + 新Offer
卷王逆袭成功案例 非全日制 6年经验卷王 喜提3个Offer, 年包30W	5月9日改简历, 5月18日接offer	涨薪: 改简历 + 新Offer
卷王逆袭成功案例 双非二本小伙伴喜招大翻身 喜提9大offer	2月22日改简历, 4月13日接offer	涨薪: 改简历 + 新Offer
小伙大三暑期很焦虑 跟着尼恩卷一年 校招新获顶级央企Offer	去年5月19日加入VIP, 今年7月5日接offer	涨薪: 改简历 + 新Offer
卷王逆袭成功案例 3年经验卷王, 涨60%	4月16日改简历, 5月11日接offer	涨薪: 改简历 + 新Offer

修改简历找尼恩（资深简历优化专家）

- 如果面试表达不好，尼恩会提供 简历优化指导
- 如果项目没有亮点，尼恩会提供 项目亮点指导
- 如果面试表达不好，尼恩会提供 面试表达指导

作为 40 岁老架构师，尼恩长期承担技术面试官的角色：

- 从业以来，“阅历”无数，对简历有着点石成金、改头换面、脱胎换骨的指导能力。
- 尼恩指导过刚刚就业的小白，也指导过 P8 级的老专家，都指导他们上岸。

如何联系尼恩。尼恩微信，请参考下面的地址：

语雀：<https://www.yuque.com/crazymakercircle/gkkw8s/khigna>

码云：<https://gitee.com/crazymaker/SimpleCrayIM/blob/master/疯狂创客圈总目录.md>