**一、基础篇**

**1.1、Java基础**

* 面向对象的特征：继承、封装和多态
* final, finally, finalize 的区别
* Exception、Error、运行时异常与一般异常有何异同
* 请写出5种常见到的runtime exception
* int 和 Integer 有什么区别，Integer的值缓存范围
* 包装类，装箱和拆箱
* String、StringBuilder、StringBuffer
* 重载和重写的区别
* 抽象类和接口有什么区别
* 说说反射的用途及实现
* 说说自定义注解的场景及实现
* HTTP请求的GET与POST方式的区别
* Session与Cookie区别
* 列出自己常用的JDK包
* MVC设计思想
* equals与==的区别
* hashCode和equals方法的区别与联系
* 什么是Java序列化和反序列化，如何实现Java序列化？或者请解释Serializable 接口的作用
* Object类中常见的方法，为什么wait  notify会放在Object里边？
* Java的平台无关性如何体现出来的
* JDK和JRE的区别
* Java 8有哪些新特性

**1.2、Java常见集合**

* List 和 Set 区别
* Set和hashCode以及equals方法的联系
* List 和 Map 区别
* Arraylist 与 LinkedList 区别
* ArrayList 与 Vector 区别
* HashMap 和 Hashtable 的区别
* HashSet 和 HashMap 区别
* HashMap 和 ConcurrentHashMap 的区别
* HashMap 的工作原理及代码实现，什么时候用到红黑树
* 多线程情况下HashMap死循环的问题
* HashMap出现Hash DOS攻击的问题
* ConcurrentHashMap 的工作原理及代码实现，如何统计所有的元素个数
* 手写简单的HashMap
* 看过那些Java集合类的源码

**1.3、进程和线程**

* 线程和进程的概念、并行和并发的概念
* 创建线程的方式及实现
* 进程间通信的方式
* 说说 CountDownLatch、CyclicBarrier 原理和区别
* 说说 Semaphore 原理
* 说说 Exchanger 原理
* ThreadLocal 原理分析，ThreadLocal为什么会出现OOM，出现的深层次原理

每次使用完ThreadLocal，都调用它的remove()方法，清除数据。

* 讲讲线程池的实现原理
* 线程池的几种实现方式
* 线程的生命周期，状态是如何转移的
* 可参考：《[Java多线程编程核心技术](http://mp.weixin.qq.com/s?__biz=MzI1NDQ3MjQxNA==&mid=2247484881&idx=2&sn=b0ecf85cd7c9e543c84e7a9859c20a26&chksm=e9c5fc60deb27576a6a9c453dabc585f43d9f29fd8a8f37ed0e7cc2f012c86b23fbd21763a39&scene=21" \l "wechat_redirect" \t "_blank)》

**1.4、锁机制**

* 说说线程安全问题，什么是线程安全，如何保证线程安全
* 重入锁的概念，重入锁为什么可以防止死锁
* 产生死锁的四个条件（互斥、请求与保持、不剥夺、循环等待）
* 如何检查死锁（通过jConsole检查死锁）
* volatile 实现原理（禁止指令重排、刷新内存）
* synchronized 实现原理（对象监视器）
* synchronized 与 lock 的区别
* AQS同步队列
* CAS无锁的概念、乐观锁和悲观锁
* 常见的原子操作类
* 什么是ABA问题，出现ABA问题JDK是如何解决的
* 乐观锁的业务场景及实现方式
* Java 8并法包下常见的并发类
* 偏向锁、轻量级锁、重量级锁、自旋锁的概念
* 可参考：《[Java多线程编程核心技术](http://mp.weixin.qq.com/s?__biz=MzI1NDQ3MjQxNA==&mid=2247484881&idx=2&sn=b0ecf85cd7c9e543c84e7a9859c20a26&chksm=e9c5fc60deb27576a6a9c453dabc585f43d9f29fd8a8f37ed0e7cc2f012c86b23fbd21763a39&scene=21" \l "wechat_redirect" \t "_blank)》

**1.5、JVM**

* JVM运行时内存区域划分
* 内存溢出OOM和堆栈溢出SOE的示例及原因、如何排查与解决
* 如何判断对象是否可以回收或存活
* 常见的GC回收算法及其含义
* 常见的JVM性能监控和故障处理工具类：jps、jstat、jmap、jinfo、jconsole等
* JVM如何设置参数
* JVM性能调优
* 类加载器、双亲委派模型、一个类的生命周期、类是如何加载到JVM中的
* 类加载的过程：加载、验证、准备、解析、初始化
* 强引用、软引用、弱引用、虚引用
* Java内存模型JMM

**1.6、设计模式**

* 常见的设计模式
* 设计模式的的六大原则及其含义
* 常见的单例模式以及各种实现方式的优缺点，哪一种最好，手写常见的单利模式
* 设计模式在实际场景中的应用
* Spring中用到了哪些设计模式
* MyBatis中用到了哪些设计模式
* 你项目中有使用哪些设计模式
* 说说常用开源框架中设计模式使用分析
* 动态代理很重要！！！

**1.7、数据结构**

* 树（二叉查找树、平衡二叉树、红黑树、B树、B+树）
* 深度有限算法、广度优先算法
* 克鲁斯卡尔算法、普林母算法、迪克拉斯算法
* 什么是一致性Hash及其原理、Hash环问题
* 常见的排序算法和查找算法：快排、折半查找、堆排序等

**1.8、网络/IO基础**

* BIO、NIO、AIO的概念
* 什么是长连接和短连接
* Http1.0和2.0相比有什么区别，可参考《[Http 2.0](http://mp.weixin.qq.com/s?__biz=MzI1NDQ3MjQxNA==&mid=2247484611&idx=1&sn=66c875392eedff8150633ddcd5d83e7a&chksm=e9c5fd72deb274648a607b9bc39bac34adadd768577b77354f6dc85422691605e210b69eeb7b&scene=21" \l "wechat_redirect" \t "_blank)》
* Https的基本概念
* 三次握手和四次挥手、为什么挥手需要四次
* 从游览器中输入URL到页面加载的发生了什么？可参考《[从输入URL到页面加载发生了什么](http://mp.weixin.qq.com/s?__biz=MzI1NDQ3MjQxNA==&mid=2247483724&idx=1&sn=e58dd30d124971c795584e8673d6cc71&chksm=e9c5f8fddeb271ebebbb6c350ed1abc252f1f26b4f35c4ce36e10bde9659a37520feabed2290&scene=21" \l "wechat_redirect" \t "_blank)》

**二、数据存储和消息队列**

**2.1、数据库**

* MySQL 索引使用的注意事项
* DDL、DML、DCL分别指什么
* explain命令
* left join，right join，inner join
* 数据库事物ACID（原子性、一致性、隔离性、持久性）
* 事物的隔离级别（读未提交、读以提交、可重复读、可序列化读）
* 脏读、幻读、不可重复读
* 数据库的几大范式
* 数据库常见的命令
* 说说分库与分表设计
* 分库与分表带来的分布式困境与应对之策（如何解决分布式下的分库分表，全局表？）
* 说说 SQL 优化之道
* MySQL遇到的死锁问题、如何排查与解决
* 存储引擎的 InnoDB与MyISAM区别，优缺点，使用场景
* 索引类别（B+树索引、全文索引、哈希索引）、索引的原理
* 什么是自适应哈希索引（AHI）
* 为什么要用 B+tree作为MySQL索引的数据结构
* 聚集索引与非聚集索引的区别
* 遇到过索引失效的情况没，什么时候可能会出现，如何解决
* limit 20000 加载很慢怎么解决
* 如何选择合适的分布式主键方案
* 选择合适的数据存储方案
* 常见的几种分布式ID的设计方案
* 常见的数据库优化方案，在你的项目中数据库如何进行优化的

**2.2、Redis**

* Redis 有哪些数据类型，可参考《[Redis常见的5种不同的数据类型详解](http://mp.weixin.qq.com/s?__biz=MzI1NDQ3MjQxNA==&mid=2247483987&idx=1&sn=5c5e4cd5bc73a7e6f84e5d6adfab0935&chksm=e9c5fbe2deb272f4b5b75bd2ac92bb27950452623ec83c0e1add7e30c773160421fab1571680&scene=21" \l "wechat_redirect" \t "_blank)》
* Redis 内部结构
* Redis 使用场景
* Redis 持久化机制，可参考《[使用快照和AOF将Redis数据持久化到硬盘中](http://mp.weixin.qq.com/s?__biz=MzI1NDQ3MjQxNA==&mid=2247483992&idx=1&sn=8f554bc490c4db1a78a30144f873e911&chksm=e9c5fbe9deb272fff47483c241e6d2a7aae99dc8f6fe9fee31f2dd214d0cf81b33d51f7a7dbe&scene=21" \l "wechat_redirect" \t "_blank)》
* Redis 集群方案与实现
* Redis 为什么是单线程的？
* 缓存雪崩、缓存穿透、缓存预热、缓存更新、缓存降级
* 使用缓存的合理性问题
* Redis常见的回收策略

**2.3、消息队列**

* 消息队列的使用场景
* 消息的重发补偿解决思路
* 消息的幂等性解决思路
* 消息的堆积解决思路
* 自己如何实现消息队列
* 如何保证消息的有序性

**三、开源框架和容器**

**3.1、SSM/Servlet**

* Servlet的生命周期
* 转发与重定向的区别
* BeanFactory 和 ApplicationContext 有什么区别
* Spring Bean 的生命周期
* Spring IOC 如何实现
* Spring中Bean的作用域，默认的是哪一个
* 说说 Spring AOP、Spring AOP 实现原理
* 动态代理（CGLib 与 JDK）、优缺点、性能对比、如何选择
* Spring 事务实现方式、事务的传播机制、默认的事务类别
* Spring 事务底层原理
* Spring事务失效（事务嵌套），JDK动态代理给Spring事务埋下的坑，可参考《[JDK动态代理给Spring事务埋下的坑！](http://mp.weixin.qq.com/s?__biz=MzI1NDQ3MjQxNA==&mid=2247484940&idx=1&sn=0a0a7198e96f57d610d3421b19573002&chksm=e9c5ffbddeb276ab64ff3b3efde003193902c69acda797fdc04124f6c2a786255d58817b5a5c&scene=21" \l "wechat_redirect" \t "_blank)》
* 如何自定义注解实现功能
* Spring MVC 运行流程
* Spring MVC 启动流程
* Spring 的单例实现原理
* Spring 框架中用到了哪些设计模式
* Spring 其他产品（Srping Boot、Spring Cloud、Spring Secuirity、Spring Data、Spring AMQP 等）
* 有没有用到Spring Boot，Spring Boot的认识、原理
* MyBatis的原理
* 可参考《[为什么会有Spring](http://mp.weixin.qq.com/s?__biz=MzI1NDQ3MjQxNA==&mid=2247484822&idx=1&sn=6fbee2a12b31b6102a18d3725671d41b&chksm=e9c5fc27deb275319641c3f30d168b85c7c196fd276d47efa35046b5dc54f5b77174c5bf8808&scene=21#wechat_redirect)》
* 可参考《[为什么会有Spring AOP](http://mp.weixin.qq.com/s?__biz=MzI1NDQ3MjQxNA==&mid=2247484827&idx=1&sn=b9d82f3fced6a875f8dfc22e5849b28e&chksm=e9c5fc2adeb2753c516ef8fc959c0c9dd84ccacaa40473b64bc58b5137c30562a0b45803ba8e&scene=21#wechat_redirect)》

**3.2、Netty**

* 为什么选择 Netty
* 说说业务中，Netty 的使用场景
* 原生的 NIO 在 JDK 1.7 版本存在 epoll bug
* 什么是TCP 粘包/拆包
* TCP粘包/拆包的解决办法
* Netty 线程模型
* 说说 Netty 的零拷贝
* Netty 内部执行流程
* Netty 重连实现

**3.3、Tomcat**

* Tomcat的基础架构（Server、Service、Connector、Container）
* Tomcat如何加载Servlet的
* Pipeline-Valve机制
* 可参考：《[四张图带你了解Tomcat系统架构](http://mp.weixin.qq.com/s?__biz=MzI1NDQ3MjQxNA==&mid=2247484905&idx=1&sn=6c8acd89476fadbc4cb9ccfda9c9c2e4&chksm=e9c5fc58deb2754e7519511bb0ed8dcbfa3fe29179663b53f3626643f8b9c82068d9b0464ee6&scene=21#wechat_redirect)！》

**四、分布式**

**4.1、Nginx**

* 请解释什么是C10K问题或者知道什么是C10K问题吗？
* Nginx简介，可参考《[Nginx简介](http://mp.weixin.qq.com/s?__biz=MzI1NDQ3MjQxNA==&mid=2247483994&idx=1&sn=b6591f62c7ea6b4adc5a5bf1bf4eac40&chksm=e9c5fbebdeb272fdd865a9c61a380f6b909fc988f99d00ce0aa8c3efca501644db46c40bd4f2&scene=21" \l "wechat_redirect" \t "_blank)》
* 正向代理和反向代理.
* Nginx几种常见的负载均衡策略
* Nginx服务器上的Master和Worker进程分别是什么
* 使用“反向代理服务器”的优点是什么?

**4.2、分布式其他**

* 谈谈业务中使用分布式的场景
* Session 分布式方案
* Session 分布式处理
* 分布式锁的应用场景、分布式锁的产生原因、基本概念
* 分布是锁的常见解决方案
* 分布式事务的常见解决方案
* 集群与负载均衡的算法与实现
* 说说分库与分表设计，可参考《[数据库分库分表策略的具体实现方案](http://mp.weixin.qq.com/s?__biz=MzI1NDQ3MjQxNA==&mid=2247483931&idx=1&sn=6eda41aa81c1243422a603205d2fad22&chksm=e9c5fbaadeb272bc92537803c14a6f55e1170b1a3b8f60160f66417800c0ace960dfe192717a&scene=21#wechat_redirect)》
* 分库与分表带来的分布式困境与应对之策

**4.3、Dubbo**

* 什么是Dubbo，可参考《[Dubbo入门](http://mp.weixin.qq.com/s?__biz=MzI1NDQ3MjQxNA==&mid=2247483791&idx=1&sn=49345f1a022734e81e9257f2b8d38a52&chksm=e9c5f83edeb2712805a77c1e1589e8f1d04bd17e55eeb2a45cabddb46d03615636908f058628&scene=21" \l "wechat_redirect" \t "_blank)》
* 什么是RPC、如何实现RPC、RPC 的实现原理，可参考《[基于HTTP的RPC实现](http://mp.weixin.qq.com/s?__biz=MzI1NDQ3MjQxNA==&mid=2247483900&idx=1&sn=c5ca198a66a701f81c2ab118fe7a734a&chksm=e9c5f84ddeb2715bc574e467cd6537ef81f223453e0989ffd136976b48dcc2d961a75be596de&scene=21" \l "wechat_redirect" \t "_blank)》
* Dubbo中的SPI是什么概念
* Dubbo的基本原理、执行流程

**五、微服务**

**5.1、微服务**

* 前后端分离是如何做的？
* 微服务哪些框架
* Spring Could的常见组件有哪些？可参考《[Spring Cloud概述](http://mp.weixin.qq.com/s?__biz=MzI1NDQ3MjQxNA==&mid=2247484125&idx=1&sn=ddba9fba6ae900f5ef71a68f70afebe5&chksm=e9c5fb6cdeb2727a1e8d16a05adb7d9df0170e392ae579d1e40075fb488d680f0c061aa45327&scene=21#wechat_redirect)》
* 领域驱动有了解吗？什么是领域驱动模型？充血模型、贫血模型
* JWT有了解吗，什么是JWT，可参考《[前后端分离利器之JWT](http://mp.weixin.qq.com/s?__biz=MzI1NDQ3MjQxNA==&mid=2247485183&idx=1&sn=05dac824dbb534710dd99d6c895fbaf5&chksm=e9c5ff4edeb27658173c8b06ad6d1241d3b7822c734ddf6ac064d40e63cb0cb0a0c90804b9c7&scene=21#wechat_redirect)》
* 你怎么理解 RESTful
* 说说如何设计一个良好的 API
* 如何理解 RESTful API 的幂等性
* 如何保证接口的幂等性
* 说说 CAP 定理、BASE 理论
* 怎么考虑数据一致性问题
* 说说最终一致性的实现方案
* 微服务的优缺点，可参考《[微服务批判](http://mp.weixin.qq.com/s?__biz=MzI1NDQ3MjQxNA==&mid=2247485005&idx=1&sn=78a1d286c6a15a81ea5dcf6634a70b54&chksm=e9c5fffcdeb276ea3c766a6e5954685db0e89bea8ff0f47c9a3ff2c0c02991f791a5160287c4&scene=21#wechat_redirect)》
* 微服务与 SOA 的区别
* 如何拆分服务、水平分割、垂直分割
* 如何应对微服务的链式调用异常
* 如何快速追踪与定位问题
* 如何保证微服务的安全、认证

**5.2、安全问题**

* 如何防范常见的Web攻击、如何方式SQL注入
* 服务端通信安全攻防
* HTTPS原理剖析、降级攻击、HTTP与HTTPS的对比

**5.3、性能优化**

* 性能指标有哪些
* 如何发现性能瓶颈
* 性能调优的常见手段
* 说说你在项目中如何进行性能调优

**六、其他**

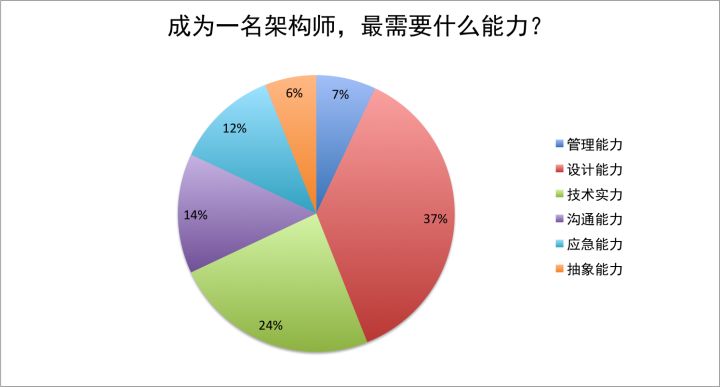
**6.1、设计能力**

* 说说你在项目中使用过的UML图
* 你如何考虑组件化、服务化、系统拆分
* 秒杀场景如何设计
* 可参考：《[秒杀系统的技术挑战、应对策略以及架构设计总结一二！](http://mp.weixin.qq.com/s?__biz=MzI1NDQ3MjQxNA==&mid=2247485294&idx=1&sn=681b3fc8833bc978344f66c8dd33ff32&chksm=e9c5fedfdeb277c96e03e8943bff709ad5354cbbe0c0d894cdb9d1e8cc8da92bf51f1e832e37&scene=21#wechat_redirect)》

**6.2、业务工程**

* 说说你的开发流程、如何进行自动化部署的
* 你和团队是如何沟通的
* 你如何进行代码评审
* 说说你对技术与业务的理解
* 说说你在项目中遇到感觉最难Bug，是如何解决的
* 介绍一下工作中的一个你认为最有价值的项目，以及在这个过程中的角色、解决的问题、你觉得你们项目还有哪些不足的地方

**6.3、软实力**

* 说说你的优缺点、亮点
* 说说你最近在看什么书、什么博客、在研究什么新技术、再看那些开源项目的源代码
* 说说你觉得最有意义的技术书籍
* 工作之余做什么事情、平时是如何学习的，怎样提升自己的能力
* 说说个人发展方向方面的思考
* 说说你认为的服务端开发工程师应该具备哪些能力
* 说说你认为的架构师是什么样的，架构师主要做什么
* 作者：只是因为情非得已  
  链接：https://www.zhihu.com/question/19558112/answer/244624134  
  来源：知乎  
  著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。
* **1 确认需求**
* 架构师要懂得用户需求，理解用户真正想要什么，这使得架构师必须要和分析人员不断沟通，反复确认需求规格说明书，以此来保证他精准清楚用户需求。
* 项目经理刘先生在受访时说：「架构师会与很多人沟通，例如开发人员，例如我们项目经理，有时甚至是用户本身。架构设计的目的很明确，目的是什么呢？挖掘用户需求。」
* **2 系统分解**
* 在架构师认可需求规格说明书后，架构师已明确用户需求是是什么，这时候便看架构师的分解能力了。
* 通过100offer入职的全栈技术架构师周先生从「纵向分解」和「横向分解」和我们说明了系统分解是什么——
* 「一般分为纵向分解和横向分解，纵向分解是将整个系统分层，从而将整体系统分解成下一级的子系统与组件。横向分解是在系统分解成不同的逻辑层或服务后，对逻辑层进行分块，确定层与层之间的关系。」
* **3 技术选型**
* 在系统分解后，架构师会最终形成软件整体架构，接下来，架构师的职责是技术选型。
* 「前端到底用瘦客户端还是富客户端呢？数据库是用MySQL还是MSSQL又或是Oracle呢？」架构师张先生在接受采访时说，「在了解用户需求后，分解完系统后，技术选型是非常重要的环节，提出各个方向，我再进行评估。不过，很多人都以为架构师是有决定权的，其实不是，架构师没有拍版的权力，决定由项目经理来做。 」
* 架构师在技术选型阶段会提供参考信息给项目经理，项目经理再从预算、进度、人力、资源等各方面情况来权衡，最终确认。
* **4 制定技术规格说明**
* 如前文调查显示，架构师在项目开发过程中是「灵魂人物」，并且要具备协调组织能力和懂得人员分工。
* 在制定技术规格说明阶段，架构师要协调起所有的开发人员，架构师通常会用技术规格说明书与开发人员保持沟通，让开发人员能从各个视角去观测、理解他们负责的模块或者子系统，确保开发人员能够按照架构意图实现各项功能。
* **在了解架构师的职责后，再来看看架构师该具备什么能力才能成为一家公司中的「灵魂人物」**。我们先来看一下调查数据——
* 
* 37%的受访人认为架构师的设计能力最重要，技术实力重要度排在第二占了24%，沟通能力则排在第三，占比14%，管理能力在大多数架构师眼中并不是最重要的，仅占了7%。此次，我们详细分析排在前三的能力。
* **1 设计能力－擅长整合分析**
* 架构是过程，并非结果。
* 架构是架构师洞察内在结构、原则、规律与逻辑的过程，架构师要做到清晰理解系统，以及简洁描述，这是分析整合的能力。
* 一个架构师必须具备极强的分析能力，要做到根据产品宗旨和目标，分析清楚产品定位以及产品业务，再整合利用现有的技术领域，找出最佳方案，实现产品概念。
* **2 技术实力－实现产品规划**
* 架构师首先要将代码写的清晰易懂，要能够实现功能，做到没有Bug，这要求架构师必须具备至少熟练掌握一门语言。
* 这是最重要的，每一名出色的架构师，必定是一位优秀程序员。架构师并不是纯粹的管理岗位，对那些爱写各式文档、画流程图、脱离代码、只说不做、高高在上的架构师，程序员们通常会称他们为——
* PPT 架构师。
* 不懂编程的架构师的职业生涯必定是短暂的，无论如何都不可本末倒置，要想实现自己的职业规划，不能荒废自己本身的技能，技术是架构师赖以生存的最基本能力。
* **所以，不推荐不热爱编程的人去做架构师，对于团队工作和个人发展来说，都会带来糟糕的后果。**
* **3 沟通能力－能够横向沟通**
* 架构师必须参与项目开发全过程，包括确认需求、系统分解、架构设计、技术选型、制定技术规格说明、系统实现、集成测试和部署各阶段，在这一系列过程中，架构师会与各部门沟通交流。
* 一个产品会有多部门合作，架构师在其中的沟通极为重要，直接影响产品进度与质量。架构师不仅要与开发人员沟通，也要和项目经理、分析人员甚至用户沟通，来实现产品的各种可能性。
* 所以，对于架构师来讲，不仅有技术方面的要求，还有能够横向沟通的要求
* 如何看待加班的问题