## 简单介绍一下以前做过的项目

这个问题应该所有的面试都会问，做了哪些项目，项目系统主要功能，有那些部分组成，项目的整体技术架构。参与过哪些模块的开发。某个模块的业务及设计。

这个可能需要自己提前准备一下

（下面的这些问题答案是按我自己的理解和以前项目的使用情况写的，可能有不完善的地方，可以自己再去找资料补充一下）

## 2.定时器用过哪些

我使用过的4种类型的定时器：@Scheduled注解、quartz、new Timer().schedule、使用线程控制。

最近做SpringBoot项目时 ，使用定时器@Scheduled注解是最简单的方式，只需要在方法上添加注解即可。

前几年做的项目用的quartz石英钟，需要创建专门配置文件spring-quartz.xml来配置，要在配置中定义工作任务的job，定义触发器Trigger并与job绑定等一系列配置。

还有就是使用new Timer().schedule（xxx，xxx，xxx），这个用的比较少。

最后使用线程控制自己写定时器这种以前也接触过，对线程理解要求高一些，不过这样子的定时器比较灵活。

## 3.微服务注册中心有哪些

1.Dubbo中的注册中心，Zookeeper和Redis。（一般项目都是用Zookeeper）。

2. SpringCloud中的注册中心，Eureka最常用

## 4.Redis数据类型

Redis是一个非关系型的内存数据库 数据储存在内存 也可以持久化到磁盘 是一个K V类型的储存系统 常用来做缓存。Redis支持五种数据类型：String（字符串），Hash（哈希），List（列表），Set（集合）及Zset(sorted set：有序集合)。可以说一下以前项目有用zset来做过排行榜的功能

## 5.外部接口怎么管控

这里的外部接口指的是暴露给外界调用的接口APi么？

那我之前的项目中对于这类的接口有用过自定义注解标注在接口上，在注解中定义访问时间，该时间内允许访问的最高次数，然后通过相应的拦截器来处理解析这个注解，控制该注解标注的接口的访问频率。防止恶意刷接口。

还有设置黑名单，对于黑名单用户，限制其操作。

再就很常见的接口签名验证，通过指定的规则和加密算法，对接口参数进行签名，通过签名检验的请求才进行下一步操作。

还有想腾讯啊，阿里他们对外的api接口采用的token方式的验证，接口调用需要先去他们那边拿到token值，然后token还有过期时间啊，接口每天的调用次数等限制。

再或者有些情况可以让前端加上验证码什么的，ip限制，ip白名单什么的。

整体系统还有接口设计时数据安全的把控也要做好，

## 6.怎么限流

在开发高并发系统时常常会考虑这些问题：缓存、降级和限流。

我理解的限流就是限制系统的输入和输出流量，设置一个需要限制的阈值，一旦系统的吞吐量到达这个阈值，就需要限制流量并采取一些措施以完成限制流量的目的。比如：延迟处理，拒绝处理，或者部分拒绝处理等等

我使用过Java 并发库 的Semaphore完成信号量控制进行并发数量限制。

还有像我上面提到的接口管控中用到的，自定义注解+拦截器+Redis实现限流。这种就是适用于分布式系统的全局限流。原理类似于计数器算法。

还有一些限流控制速率的方案，像是使用限流算法：漏桶算法和令牌桶算法。

还了解过nginx 限流，通过在nginx里配置并发数和连接数来完成限流。

## 7.代码优化

代码优化这块的内容就很大了，涉及到很多个方面的优化。我理解的是代码优化每天都要进行，而不是一两个月做一次大优化，那时做就已经晚了。另外由于优化是每天做的，所以你不需要一次的就过度优化，保持小步快跑即可。

优化方面有重复代码优化，避免大量重复代码。适当的抽出可复用的函数。

还有性能方面的优化，写代码时尽量考虑到这段逻辑会不会有性能问题，是否能更优性能的处理。

还有就是一些代码规范，像单例的使用啊，静态变量的使用啊，java对象的创建和回收啊，IO操作流的关闭等等很多小的规范和细节需要平时经验的积累。

## 8.内存泄露怎么处理

java内存的释放，也即清理那些不可达的对象，是由GC决定和执行的，GC会监控每一个对象的状态，包括申请、引用、被引用和赋值等。防止内存泄漏的措施可以在平时的编码中多做赋值为null的操作。

还有各种提供了close()方法的对象使用时，记得最后close。比如数据库连接（dataSourse.getConnection()），网络连接(socket)和io连接。

定位和分析内存泄漏可以使用

linux的top命令观察cpu使用率。

JDK自带的工具jstat和jmap去dump出JVM内存映像，然后使用内存泄漏工具分析这个dump文件，内存泄漏工具有很多，在百度上一搜就能搜到很多，最后从工具分析生成的分析结果可以很容易找到占用内存最多的对象，结合程序分析这个对象在程序中的使用，一般很容易就定位出内存泄漏的原因。

## 9.慢sql怎么处理

在项目自验项目转测试之前，在启动mysql数据库时开启慢查询，并且把执行慢的语句写到日志中，在运行一定时间后。通过查看日志找到慢查询语句。

慢查询定位及改进：

1. Show命令、慢查询日志、explain分析查询、profiling分析

② 进行相应的优化

## 10.消息队列

消息队列是用来解决 解耦、异步、削峰问题。

目前使用较多的消息队列有ActiveMQ，RabbitMQ，ZeroMQ，Kafka，MetaMQ，RocketMQ，

还有Redis也可作为队列使用对list 和 set 操作。

之前项目中有使用过RocketMQ，支持发布/订阅（Pub/Sub）和点对点（P2P）消息模型。Topic区分消息业务类型。

处理过消息重复消费的问题，当时是在消费者消费消息时，把对应的唯一值放入HashSet 或者Redis来避免同一条消息消费多次

## 11.熟悉哪些设计模式

单例模式：（很常见，略）

工厂模式（普通、多方法和静态工厂方法、抽象工厂）：该模式主要功能是统一提供实例对象的引用，很多框架里都有使用这种设计模式。Spring IOC就是使用的工厂模式.

策略模式：这个模式是将行为的抽象，即当有几个类有相似的方法，将其中通用的部分都提取出来，从而使扩展更容易。

门面模式：这个模式个人感觉像是Service层的一个翻版。比如Dao我们定义了很多持久化方法，我们通过Service层将Dao的原子方法组成业务逻辑，再通过方法向上层提供服务。门面模式道理其实是一样的。

代理模式：其实每个模式名称就表明了该模式的作用，代理模式就是多一个代理类出来，替原对象进行一些操作，Spring AOP就是使用的动态代理。

## 12.spring、springMvc、springBoot和springCloud的联系与区别

1. Spring 框架就像一个家族，有众多衍生产品例如 boot、security、jpa、mvc等等。但他们的基础都是Spring的ioc、aop等，然后在此两者的基础上实现了其他延伸产品的高级功能；

2. springMvc是在Spring之下基于Servlet 的一个MVC框架，主要解决WEB开发的问题，可以和spring无缝集成，使用起来需要一系列xml配置；

3. 为了简化开发者的使用，从而创造性地推出了springBoot框架，默认优于配置，简化了springMvc的配置流程；

但区别于springMvc的是，springBoot专注于微服务方面的接口开发，和前端解耦，虽然springBoot也可以做成springMvc前后台一起开发，但是这就有点不符合springBoot框架的初衷了；

4. 对于springCloud框架来说，它和springBoot一样，注重的是微服务的开发，但是springCloud更关注的是全局微服务的整合和管理，相当于管理多个springBoot框架的单体微服务；

## 13.库存超卖情况怎么处理

这种情况在什么秒杀啊，高并发场景经常会遇到，我之前有遇到过类似的情况，因为并发很高，如果直接操作数据库，就算利用数据量的各种锁机制控制或者版本号控制，这个读写量数据库性能压力还是很大的。所有我一般这种情况会考虑将存库从MySQL前移到Redis中，所有的写操作放到内存中，因为Redis的写性能和读性能都远高于MySQL，然后通过队列等异步手段，然后将所有写DB操作在单队列中排队，完全串行处理。当达到库存阀值的时候就不在消费队列，并关闭购买功能。这样来解决性能和超卖的问题。

还有就是以前项目中有用过redis分布式锁。使用这个分布式锁也能处理简单的秒杀场景，秒杀商品时候先加锁，如果没有获取到锁就释放请求。加锁后先进行库存判断如果不足释放请求。进行秒杀下单流程，如果成功库存做减一操作，最后释放分布式锁。

## 14.kafka、RabbitMQ使用情况

以前项目中有用到这两种队列。

之前用到的RabbitMQ高可用使用方案集群模式 + 镜像。

处理过RabbitMQ消息丢失问题，因为RabbitMQ在内存缓存消息，当MQ宕机时，会发生消息丢失现象。这一点需要RabbitMQ将消息持久化到硬盘上。减小消息丢失的可能。

还有RabbitMQ默认是autoAck=true，这样就使得消费者在接受到消息时，立马告知MQ我消费了数据，但还没有来得及处理。所以之前的项目中设置的是autoAck=false，当消息处理完成之后手动提交。

Kafka也有类似的操作，在消费者端关闭自动确认消息，自己手动确认 保证消息不会丢失。

## 15.怎么提高查询效率等情况

合理建立索引、优化索引相关的查询SQL

1>. 当结果集只有一行数据时使用LIMIT 1

2>. 避免SELECT \*，始终指定你需要的列

3>. 使用连接（JOIN）来代替子查询(Sub-Queries)

4>. 使用合理的字段属性长度

5>. 尽可能的使用NOT NULL

6>. 固定长度的表会更快

7>. 拆分大的DELETE 或INSERT 语句

8>. 查询的列越小越快

有些where条件会导致索引无效：

Ø where子句的查询条件里有！=，MySQL将无法使用索引。

Ø where子句使用了Mysql函数的时候，索引将无效

Ø 使用LIKE进行搜索匹配的时候，模糊匹配不能放在前面，否则失效。如’%xxxx’

③配置优化

MySQL服务器全局参数优化

④分表（横、纵）

减少单表的数据量及将常用列和不常用列、大列和小列分到不同的表。