1. spring mvc的原理？

Spring mvcde 的原理也就是在问工作原理。

1、客户端发出一个http请求给web服务器，web服务器对http请求进行解析，如果匹配DispatcherServlet的请求映射路径（在web.xml中指定），web容器将请求转交给DispatcherServlet.

2、DipatcherServlet接收到这个请求之后将根据请求的信息（包括URL、Http方法、请求报文头和请求参数Cookie等）以及HandlerMapping的配置找到处理请求的处理器（Handler）。

3-4、DispatcherServlet根据HandlerMapping找到对应的Handler,将处理权交给Handler（Handler将具体的处理进行封装），再由具体的HandlerAdapter对Handler进行具体的调用。

5、Handler对数据处理完成以后将返回一个ModelAndView()对象给DispatcherServlet。

6、Handler返回的ModelAndView()只是一个逻辑视图并不是一个正式的视图，DispatcherSevlet通过ViewResolver将逻辑视图转化为真正的视图View。

7、Dispatcher通过model解析出ModelAndView()中的参数进行解析最终展现出完整的view并返回给客户端。

2.既然你们使用了ActiveMQ，是如何进行处理分布式事件的？

使用消息队列来避免分布式事务。

比如在北京很有名的姚记炒肝点了炒肝并付了钱后，他们并不会直接把你点的炒肝给你，而是给你一张小票，然后让你拿着小票到出货区排队去取。为什么他们要将付钱和取货两个动作分开呢？原因很多，其中一个很重要的原因是为了使他们接待能力增强（并发量更高）

还是回到我们的问题，只要这张小票在，你最终是能拿到炒肝的。同理转账服务也是如此，当支付宝账户扣除1万后，我们只要生成一个凭证（消息）即可，这个凭证（消息）上写着“让余额宝账户增加 1万”，只要这个凭证（消息）能可靠保存，我们最终是可以拿着这个凭证（消息）让余额宝账户增加1万的，即我们能依靠这个凭证（消息）完成最终一致性。

一、结合MQ消息中间件实现的可靠消息最终一致性  
二、TCC补偿性事务解决方案  
三、最大努力通知型方案  
  
第一种方案：可靠消息最终一致性，需要业务系统结合MQ消息中间件实现，在实现过程中需要保证消息的成功发送及成功消费。即需要通过业务系统控制MQ的消息状态  
第二种方案：TCC补偿性，分为三个阶段TRYING-CONFIRMING-CANCELING。每个阶段做不同的处理。  
TRYING阶段主要是对业务系统进行检测及资源预留  
CONFIRMING阶段是做业务提交，通过TRYING阶段执行成功后，再执行该阶段。默认如果TRYING阶段执行成功，CONFIRMING就一定能成功。  
CANCELING阶段是回对业务做回滚，在TRYING阶段中，如果存在分支事务TRYING失败，则需要调用CANCELING将已预留的资源进行释放。  
第三种方案：最大努力通知xing型，这种方案主要用在与第三方系统通讯时，比如：调用微信或支付宝支付后的支付结果通知。这种方案也是结合MQ进行实现，例如：通过MQ发送http请求，设置最大通知次数。达到通知次数后即不再通知。  
具体的案例你也可以参考下这篇博客，它上面的这个案例就是结合电商支付做的系统分布式事务实现案例：http://www.roncoo.com/article/detail/124243

基于事务消息的MQ方案是目前公认的较为理想的分布式事务解决方案，各大电商都在应用这一方案。种方式适合的业务场景广泛，而且比较可靠。不过这种方式技术实现的难度比较大。目前主流的开源MQ（ActiveMQ、RabbitMQ、Kafka）均未实现对事务消息的支持，所以需二次开发或者新造轮子。

1. 什么是中间件，还了解哪些中间件？

中间件是一种独立的系统软件或服务程序，分布式应用软件借助这种软件在不同的技术之间共享资源。

小项目，或者是个人开发tomcat 大项目或者商业项目一般采用：weblgoic/webshere

其他的还有jboss、glasshfish等

一些示例项目或者小项目常采用jetty

4，知道高级消息队列吗？

高级队列Advanced Queuing(AQ)在oracle多个版本都可得到。

他是oracle原生消息软件并且在每一个版本都在加强。

这篇文章提供了一个AQ的高级概览。尤其是我们将看到如何启动一个队列并进行入列--出列操作，还有通过通知创建异步出列。

注意AQ支持数据库意外的消息监听（例如JMS消息队列）。本文仅涉及数据库内部消息通信。

5，服务之间是如何解藕的？

对于分布式事务最好的方式还是事务补偿或者BASE基于消息的最终一致性。

6,active挂了怎么办（这个项目用activeMQ传递消息的，要是activeMQ宕机了，那这消息就没法传递了）？

首先我们公司是用的mq集群，它提供了一种叫做失效转移也叫故障转移，FailOver的策略。失效转移提供了在传输层重新连接到其他任何传输器的功能。

客户端发现uri1地址失效了，它会立即转向uri地址列表中其他可以连接的消息服务器进行重连，以保证继续正常工作。

7，dubbo怎么处理容错的？

|  |  |
| --- | --- |
| Dubbo缺省会在启动时检查依赖的服务是否可用，不可用时会抛出异常，阻止Spring初始化完成，以便上线时，能及早发现问题，默认check=true。 | |
|  | 如果你的Spring容器是懒加载的，或者通过API编程延迟引用服务，请关闭check，否则服务临时不可用时，会抛出异常，拿到null引用，如果check=false，总是会返回引用，当服务恢复时，能自动连上。 |

可以通过check="false"关闭检查，比如，测试时，有些服务不关心，或者出现了循环依赖，必须有一方先启动。

关闭某个服务的启动时检查：(没有提供者时报错)

|  |
| --- |
| <dubbo:reference interface="com.foo.BarService" check="false" /> |

关闭所有服务的启动时检查：(没有提供者时报错)

|  |
| --- |
| <dubbo:consumer check="false" /> |

关闭注册中心启动时检查：(注册订阅失败时报错)

|  |
| --- |
| <dubbo:registry check="false" /> |

也可以用dubbo.properties配置：

**dubbo.properties**

|  |
| --- |
| dubbo.reference.com.foo.BarService.check=false  dubbo.reference.check=false  dubbo.consumer.check=false  dubbo.registry.check=false |

8，dubbo生产者消息发不出去怎么办？

如果你在确定你的所有配置是正确的，指定的Zookeeper注册中心IP地址是正确的，所有使用到的端口都已经通过防火墙的配置的话，遇到无法获取生产者提供的服务，或者注册失败等信息的话，不妨看一下是不是因为**内网IP**造成的。

因为，我的这个项目使用的是阿里云的云服务器，有两台，即是上文中的服务器A和服务器B，遇到这个问题之后发现阿里云默认的为我们将内网IP和主机名进行了关联。

9.jsonp的原理

就是利用<script>标签没有跨域限制的“漏洞”（历史遗迹啊）来达到与第三方通讯的目的。

10，一个系统怎么获取到另一个系统的cookie

cookie存入redis可以实现cookie跨域共享。

使用jsonp来通过ajax实现，而且这种方法还比较流行。

Redis用过redisnx吗，有哪几种数据结构？

反正我是不知道redisnx是什么，度娘也不清楚，如果面试中问道自己没有接触过或者没有听过的技术可以直接大胆的告诉他，没有接触过，或者没有听过。

Redis的数据结构有五种，分别是：

String——字符串

String 数据结构是简单的 key-value 类型，value 不仅可以是 String，也可以是数字（当数字类型用 Long 可以表示的时候encoding 就是整型，其他都存储在 sdshdr 当做字符串）。  
Hash——字典

在 Memcached 中，我们经常将一些结构化的信息打包成 hashmap，在客户端序列化后存储为一个字符串的值（一般是 JSON 格式），比如用户的昵称、年龄、性别、积分等。  
List——列表

List 说白了就是链表（redis 使用双端链表实现的 List），相信学过数据结构知识的人都应该能理解其结构。

Set——集合

Set 就是一个集合，集合的概念就是一堆不重复值的组合。利用 Redis 提供的 Set 数据结构，可以存储一些集合性的数据。

Sorted Set——有序集合

和Sets相比，Sorted Sets是将 Set 中的元素增加了一个权重参数 score，使得集合中的元素能够按 score 进行有序排列，

1.带有权重的元素，比如一个游戏的用户得分排行榜  
2.比较复杂的数据结构，一般用到的场景不算太多

线程池了解过吗？为什么要用到线程池？嗯，了解过。

诸如Web 服务器、数据库服务器、文件服务器或邮件服务器之类的许多服务器应用程序都面向处理来自某些远程来源的大量短小的任务。

Mysql存储引擎，使用innerdb的好处？一般情况下InnoDB表最好要有自增列做主键，因为**InnoDB引擎表是基于B+树的索引组织表(IOT)。**

B+ 树的特点：

a、所有关键字都出现在叶子结点的链表中(稠密索引)，且链表中的关键字恰好是有序的;

b、不可能在非叶子结点命中;

c、非叶子结点相当于是叶子结点的索引(稀疏索引)，叶子结点相当于是存储(关键字)数据的数据层。

Mysql读写分离你怎么做？思路？

搭设一台Master服务器（win8.1系统，Ip：192.168.0.104），搭设两台Slave服务器（虚拟机——一台Ubuntu，一台 Windows Server 2003）

原理：主服务器（Master）负责网站NonQuery操作，从服务器负责Query操作，用户可以根据网站功能模特性块固定访问Slave服务器，或者自己写个池或队列，自由为请求分配从服务器连接。主从服务器利用MySQL的二进制日志文件，实现数据同步。二进制日志由主服务器产生，从服务器响应获取同步数据库。

Session共享怎么做的？ 1.服务器实现的session复制或session共享，这类型的共享session是和服务器紧密相关的，比如webSphere或JBOSS在搭建集群时候可以配置实现session复制或session共享，但是这种方式有一个致命的缺点，就是不好扩展和移植，比如我们更换服务器，那么就要修改服务器配置。

2.利用成熟的技术做session复制，比如12306使用的gemfire，比如常见的内存数据库如redis或memorycache，这类方案虽然比较普适，但是严重依赖于第三方，这样当第三方服务器出现问题的时候，那么将是应用的灾难。

3.将session维护在客户端，很容易想到就是利用cookie，但是客户端存在风险，数据不安全，而且可以存放的数据量比较小，所以将session维护在客户端还要对session中的信息加密。

Aop了解吗？怎么用的？项目用过吗？

主要的功能是：日志记录，性能统计，安全控制，事务处理，异常处理等等。

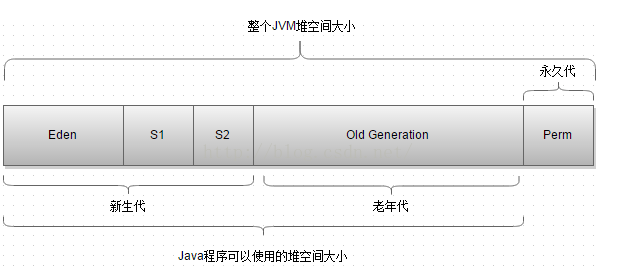
主要的意图是：将日志记录，性能统计，安全控制，事务处理，异常处理等代码从业务逻辑代码中划分出来，通过对这些行为的分离，我们希望可以将它们独立到非指导业务逻辑的方法中，进而改变这些行为的时候不影响业务逻辑的代码。

可以通过预编译方式和运行期动态代理实现在不修改源代码的情况下给程序动态统一添加功能的一种技术。AOP实际是GoF设计模式的延续，设计模式孜孜不倦追求的是调用者和被调用者之间的解耦,提高代码的灵活性和可扩展性，AOP可以说也是这种目标的一种实现。

在Spring中提供了面向切面编程的丰富支持，允许通过分离应用的业务逻辑与系统级服务（例如审计（auditing）和事务（transaction）管理）进行内聚性的开发。应用对象只实现它们应该做的——完成业务逻辑——仅此而已。它们并不负责（甚至是意识）其它的系统级关注点，例如日志或事务支持。

虚拟机内存结构？堆内存的具体结构？ JVM内存结构主要由类加载器子系统、运行时数据区（内存空间）、执行引擎以及与本地方法接口等组成。其中运行时数据区又由方法区、堆、Java栈、PC寄存器、本地方法栈组成。

堆内存具体结构



Zookeeper挂了怎么办？

Zookeeper 中的监视是轻量级的，因此容易设置、维护和分发。当客户端与 Zookeeper 服务器端失去联系时，客户端并不会收到监视事件的通知，只有当客户端重新连接后，若在必要的情况下，以前注册的监视会重新被注册并触发，对于开发人员来说 这通常是透明的。只有一种情况会导致监视事件的丢失，即：通过 exists() 设置了某个 znode 节点的监视，但是如果某个客户端在此 znode 节点被创建和删除的时间间隔内与 zookeeper 服务器失去了联系，该客户端即使稍后重新连接 zookeeper服务器后也得不到事件通知。

如果数据库商品表里面的字段太多，你怎么优化？

需要看具体sql，你查的字段多，当然返回的结果集就大，可考虑建立有效索引。

http的长连接和短连接

HTTP1.1规定了默认保持长连接（HTTP persistent connection ，也有翻译为持久连接），数据传输完成了保持TCP连接不断开（不发RST包、不四次握手），等待在同域名下继续用这个通道传输数据；相反的就是短连接。

**在HTTP/1.0中，默认使用的是短连接**。也就是说，浏览器和服务器每进行一次HTTP操作，就建立一次连接，任务结束就中断连接。

**从HTTP/1.1起，默认使用的是长连接，**用以保持连接特性。

虚拟机内存满了，怎么排查？

怎么排除不太清楚，如果因为内存满了报警可以用虚拟机可视化工具对虚拟机内存进行修改。

mysql查询慢具体步骤？zd具体怎么调用接口的？map原理！concurrentMap！用什么方式解决并发数据的安全！乐观锁悲观锁！mq的具体场景！jvm内存模型！堆内存具体结构！gc是用的算法！架构图！rpc的作用！怎么查看线程日志！怎么查看内存！某个服务器利用率过高怎么排查！订单，打折什么的怎么处理！zd的具体都有那些成员！aop原理！jdk8接口有什么特性！设计模式！代理模式！springmvc原理！bean生命周期！适配器模式！模板方法模式！分表为什么快！filter执行方式！spring方法a调用方法b事务怎么样！重写servlet的service方法会调用get和post方法么？

任丽芳 面试题：

我的问题，1.负载均横除了用nginx还能用别的吗？为什么用nginx,2.实际项目中什么地方用到了多线程，3.实际开发中用到那些设计模式，4.linux系统中如何查看日志文件