|  |
| --- |
| 技术问题(至少50个) |
| 事务篇：  问题1：事务的  问题2：对Spring事务的理解  回答：  问题：对分布式事务了解吗？  回答：简单的说，就是一次大的操作由不同的小操作组成，这些小的操作分布在不同的服务器上，而且属于不同的应用，分布式事务需要保证这些小的操作要么全部成功，要么全部失败，分布式事务就是为了保证不同的数据库一致性。  问题：你是如何解决分布式事务的？  回答：  1.两阶段提交方案/XA方案： 举个例子：公司每隔一段时间都会搞一次团建，团建就相当于是一次事件，组织者也就是负责人，第一个阶段：他回询问各自的意见（比如果去哪，都有啥活动），如果都ok就去做这件事，如果其中一个人不ok就取消。第二个阶段：  前提是第二个都ok，大家就可以一起完成了。 这里的负责人就是个事务管理器，主要负责协调多个数据库的事务，保证了数据的一致性，首先去执行各地的本地事务，然后就可以正式提交了整个事务了，如果有一个数据库执行失败了，就需要回滚。可以使用spring+JTA来实现。  缺点：这种方案适用于单体应用，（一个应用连接多个数据源，会跨库操作，严重依赖于数据库操作事务，操作效率低，不适合用于高并发场景，现在都用微服务了，主张的是一个应用连接一个数据库，不允许跨库操作）   1. TCC方案（强一致性）补偿性事务:属于两阶段提交方式的一个变种。T（Try）：对各个服务的资源做确认，及对资源锁定和保留。C（Confirm）：各个服务实际的操作。C（Cancel）：只要有其中一个服务的业务方法出现错误，就需要补偿（即已经执行成功的操作进行回滚）   举个例子：跨行转账：T：先将两个银行账户的资金冻结；C：实际进行转账，A减钱，B加钱；C：如果有任何银行的操作失败了，就需要回滚补偿。比如：A已经扣款成功，B在增加资金的时候失败了，此时就需要把A的钱加回去。  缺点：回滚操作是严重依赖于开发人员的代码进行回滚或者补偿的，代价比较大，所以适应场景是有限的。一般是要求数据一致性要求特别高的时候才用到TCC方案，比如说银行关于钱的一些业务。  3.本地消息表：1）A系统在自己的本地事务操作的同时，会插入一条数据到消息表  2）接着A系统会把找个数据发送到MQ，A系统就是消息的生产者  3）B系统接收到消息（B系统是个消息的消费者），在自己的本地事务中，往自己的本地消息插入一个数据，同时执行自己的业务，如果消息被处理过了，那么这个事务就需要回滚。  4）B系统执行成功后，会更新自己的本地消息表及A系统的本地消息表的状态  5）如果B系统处理失败了，首先不会更新自己的本地消息表，A系统会自己定时扫描自己的消息表，如果发现数据没有改变成成功的状态，会往B继续发送消息。  6）这个方案保证的是消息的最终一致性，即使B失败无数次，A会一直发送消息，直到B成功。A和B并不是严格一致的，而是最终一致的，我们在这里牺牲了一致性，换来了性能的大幅度提升。当然，这种玩法也是有风险的，如果B一直执行不成功，那么一致性会被破坏。  缺点：这个方案严重依赖数据库的消息表来管理事务，不适合于高并发的场景，实际使用少。  4.可靠消息的最终一致性：如果没有本地消息表，直接使用MQ来完成整个事务的操作，RocketMQ  1）A系统先发送一条prepared消息到MQ，如果消息发送失败，就取消操作别执行  2）如果发送成功，MQ会给A发送回调消息，那么A系统就可以处理自己的本地事务，如果失败了，告诉MQ进行回滚。  3）A系统执行成功给MQ发送确认消息，此时B系统就会接收到这个消息然后执行本地事务  注：一般一个项目用到分布式事务也就两三个场景，全用分布式事务的话成本太大，会导致项目特别复杂。  问题：分布式事务如何保证事务的一致性：  回答：  多线程篇：  问题2：进程和线程的区别  回答：进程是正在运行的应用程序，进程是线程的集合。  线程就是一条执行路径，一个独立的执行单元。  多线程就是为了提高程序的效率，多线程之间每个线程互不影响。  问题3：对线程的应用场景：多线程下载，QQ，爬虫，前端开发ajax（异步上传），分布式job（需要同时一个执行多个任务调度）、使用多线程体现程序的效率。  问题4：线程的5种状态：   1. 创建（new Thread()）、 2. 就绪（调用线程start()方法）、 3. 运行（当线程获得CPU事件后，才进入运行状态，真正开始执行run()方法）、 4. 阻塞（由于各种原因进入阻塞状态：1>线程通过调用sleep()方法进入睡眠状态；   2>线程调用一个在I/O上被阻塞的操作，即该操作在输入输出操作完成之前不会返回到它的调用者；3>线程试图得到一个锁，而该锁正被其他线程持有；4>线程在等待某个触发条件。）   1. 死亡（两个原因会导致线程死亡：1）run方法正常退出而自然死亡；2）出现异常而未被捕获，从而终止了run方法而使线程猝死。   为了确定线程在当前是否存活（要么是可运行的，要么是被阻塞了），需要使用isAlive()方法。如果是可运行或被阻塞，这个方法返回true；如果线程仍旧是new状态且不是是可运行的，或者线程死亡了，则返回false；  问题4：  Mybiatis篇：  问题：什么是Mybatis一级缓存？  回答：是指在内存中开启一块区域，用来保存用户对数据库的操作信息和数据库返回的数据，如果下一次用户再执行相同的请求，那么直接从内存中读取数据而不是从数据库中读取。  问题：一级缓存中数据生命周期的影响因素（）  回答：1.对sqlsession执行commit操作，也就意味着用户执行了update，delete等相关操作，那么数据库中的数据势必会发生变化，如果用户请求数据仍使用之前的内存中的数据，那么将读到脏数据，所以在执行sqlsession操作后，会清除保存数据的hashMap，用户再发起查询请求时就会重新读取数据并放入一级缓存中了。  2.关闭sqlsession，一般在mybatis集成spring时，会把SqlSessionFactory设置为单例注入到IOC容器中，不把sqlsession也设置为单例的原因是sqlsession是线程不安全的，所有不能为单例。那也就意味着其实是有关闭sqlsession的过程的，其实，对应每一个service中的sqlsession是不同的，这是通过mybatis-spring中的org.mybatis.spring.mapper.MapperScannerConfigurer创建sqlsession自动注入到service中的。  而每个sqlsession单独使用一个缓存空间，不同的sqlsession是不能互相访问数据的，在sqlsession关闭后，其中数据自然被清空。  问题2：为什么要使用二级缓存？  回答：一级缓存中，Sqlsession被关闭后再执行同样的查询操作也会再访问一遍数据库，为了解决这个问题，需要使用二级缓存，一级缓存的作用域仅限于一个sqlsession，但是二级缓存的作用域是一个namespace。如果开启了二级缓存，那么在关闭sqlsession后，会把该sqlsession一级缓存中的数据添加到namespace的二级缓存中。  问题：如何开启二级缓存？  回答：1.打开总开关，只需要mybatis总配置文件中加入一行设置  <settings>  <!--开启二级缓存-->  <setting name=”cacheEnabled” value=”true”/>  </settings>  2.打开需要使用二级缓存的mapper的开关，在需要开启二级缓存的mapper.xml中加入cache标签<cache/>  3.pojo序列化：让需要使用二级缓存的pojo类实现Serializable接口。通过之前的三步操作就可以使用二级缓存了。注意：只有关闭了sqlsession之后，才会把其中一级缓存数据写入二级缓存。  Redis篇：  问题：Redis缓存穿透？  回答：指正查询一个数据库一定不存在的数据，首先会从redis缓存中进行查询，如果key不存在或者key已经过期，会进数据库中进行查询，查不到数据则不写入缓存，这将导致这个不存在的数据每次请求都要到数据库去查询，当在大流量流入时，频繁的访问存储层导致数据库直接宕机，造成缓存穿透。  问题：redis缓存穿透的解决方案？  回答：常见的两种，第一种采用布隆过滤器：提前对所有可能查询的参数以hash形式存储，在控制层先进行校验，不存在则直接被拦截，放弃下面的操作。从而避免了对底层存储系统的查询压力。第二种采用缓存空对象，如果从数据库查询不存在，那么会直接在redis存储一个null或者别的特别的字段，这样下次再访问会从redis中查询，防止穿透的发生。  第三种使用互斥锁排队：就是根据key获取value为空的时候，上锁，从数据库总load数据后再释放锁，若其他线程获取锁失败，则等待一段时间后重试，但是这种方法使得逻辑变得更复杂了，治标不治本。  问题：什么是redis缓存雪崩？如何解决？  回答：指在某一个时间，缓存集中过期失效，产生的原因之一是在对数据进行缓存的时候是指的过期时间相同导致的，解决的办法1.就是在缓存数据的时候尽可能的分散设置过期时间的范围，热数据设置缓存时间长一点；2.加锁排队；3.建立备份缓存；  问题：缓存的击穿：是指一个key非常热，在不停的抗着大并发，当这个key在失效的瞬间，持续的大并发就穿破缓存，直接请求数据库。解决办法：直接将数据生命周期设置成不过期。（*实际项目中除了中真正能碰见如此大并发的项目比较少，如果能碰见如此大的并发，那一定是积累了大量的客户才会这样，比如像淘宝天猫拼多多这种平台，一般的平台没有这么大的并发*）  问题：redis持久化问题？  回答：redis提供了RDB持久化和AOF持久化，RDB是指在制定的时间间隔内将内存中的数据集快照写入磁盘，这也是默认的持久化方式，这种方式就是将内存中数据以快照的方式写入到二进制文件中，默认文件名称为dump.rdb，我们可以配置在n秒内，如果超过m个key被修改就自动做快照，下面是默认的快照保存配置。  save 900 1 #900秒内如果超过1个key被修改，则发起快照保存  save 300 10 #300秒内容如超过10个key被修改，则发起快照保存  save 60 10000  问题：拦截器和过滤器有哪些区别？  回答：  1.拦截器是基于java反射机制的，过滤器是基于函数回调  2.拦截器只对action请求起作用，而过滤器可以对几乎所有请求起作用  3.拦截器不依赖于servlet容器，过滤器依赖于servlet容器  4.在action生命周期中，拦截器可以多次被调用，而过滤器只能在容器初始化的时候被调用一次。  问题：get和load区别？429  回答：get是立即加载，load是延迟加载。Get查询时会先访问一级缓存，在访问二级缓存，都没有，访问数据库，load查询时会先访问一级缓存，没有则直接创建一个代理对象，需要时代理对象才会去查询二级缓存和数据库，get如果没有查询到会返回null，load没找到会直接抛异常。当我们使用session.load()方法加载对象时，此时并不会发出SQL语句，当前得到的是代理对象，里面只保存了实体对象的id，只有当我们使用这个对象，得到其他属性，这时才会发出sql语句，从数据库查询我们的对象；而当我们session.get()方法来得到一个对象时，不管我们用不用这个对象，此时都会发出sql语句去数据库查询。  问题：session的清理和清空有什么区别？433  回答：session清理缓存是指按照缓存中对象的状态的变化来同步更新数据库，清空是直接关闭session。  ....... |
| **人资问题(至少30个)** |

|  |
| --- |
| 您好，我叫刘佳鑫，  辽宁铁岭人，今年28岁，  2014年6月份 毕业于沈阳工程学院，毕业后去外地工作，工资不高，听朋友介绍it行业前景好，工资高。  培训经历：2014.10--2015.10，it培训1年  第一家公司：2015.10---2016.11 畅捷通信息技术股份有限公司  第二家公司（外包）：2016.12---2019.04，北京华路时代信息技术股份有限公司 项目结项了  北京市海淀区上地三街9号B座4层B407室。（做完了结项了）  （大概3年半工作经历）  上一个项目做的是金融，  做的是p2p的一个信用借贷的项目，这个项目用到了项目管理：项目使用maven来管理前后台两个子系统系统项目 数据库：MySQL+Redis 持久化层：MyBatis 业务层：spring Web：SpringMVC 前端：jQUery+Bootstrap3  项目一共分为前台后台两个系统，前台用户系统，后台管理员系统，我参与的项目有前后台的个人信息绑定，用户实名制认证，前后台登陆注册，借款还款等等。  问题1：你的离职原因是啥  回答：项目结项了。  问题2：之前是做什么的？  回答：电厂集控值班员。  问题3：哪个大学毕业的？  回答：毕业于沈阳工程学院  问题4：学的什么专业？  回答：热能与动力工程  问题5：之前在哪儿工作？  回答：重庆綦江区  问题6：学历学信网可查吗？  回答：可查  问题7：说说你的经历  回答：2014年6月毕业，之后就到重庆工作了，干了三个月，那儿的生活不是我想要的，工资低，危险。正好有朋友搞it而且已经毕业了  问题8：你觉得自己有哪些缺点？  回答：不会拒绝，直男，觉得谁都是好朋友  问题9：优点  回答：乐观，努力，刻苦  问题10：下班都会做什么？  回答：  问题11  回答：  问题12  回答：  问题13  回答：  问题14  回答：  问题15  回答：  问题16  回答：  问题17  回答：  问题18  回答：  问题19  回答：  问题20  回答：  问题21  回答：  问题22  回答：  问题23  回答：  问题24  回答：  问题25  回答：  问题26  回答：  问题27  回答：  问题28  回答：  问题29  回答：  问题30  回答：  回答3：  ........ |