面试题整理

笔记本:	My Notebook		
创建时间:	2023/9/7 16:37	**更新时间:**	2023/9/9 14:35
作者:	pe8jmft6		
URL:	https://c.binjie.fun/#/chat/1691719279859		

先介绍项目，介绍完会问spring boot面试题

1.spring boot 注解有哪些

@Component，@Controller，@RestController，@Service，@Autowired，@Resource，@Bean，@RequestMapping，@PostMapping，@RequestParam，@RequestBody（把前端传递过来的form表单转换成对象），@Responsebody（将User对象转换为JSON格式的响应体，并发送给客户端）

2.@Autowired和@Resource有什么区别?

@Autowired默认按照类型进行自动装配，可以与@Qualifier联合使用，@Resource默认按照名称进行自动装配，@Autowired可以用于字段、构造方法、Setter和普通方法，@Resource只能用于字段和Setter方法

3.Spring Boot自动装配原理

Spring Boot自动装配就是自动去把第三方组件的Bean装载到IOC容器里面，不需要开发人员再去写Bean相关的配置，在Spring Boot应用里面，只需要在启动类上加上@SpringBootApplication注解，就可以去实现自动装配。

4.@SpringBootApplication包含什么?

@SpringBootApplication是一个复合注解，真正去实现自动装配的注解是@EnableAutoConfiguration这个注解，引入Starter，启动依赖组件的时候，这个组件里面必须要包含一个@Configuration配置类，而在这个配置类里面，我们需要通过@Bean这个注解声明需要装配到IOC容器里面的Bean对象。

5.@Controller和@RestController的区别

@Controller：@Controller注解用于声明一个类是控制器（Controller），通常用于接收和处理用户的请求，并返回相应的视图。方法上的处理请求的注解可以使用@RequestMapping、@GetMapping、@PostMapping等。

@RestController：@RestController是Spring4之后新加入的注解，它是@Controller和@ResponseBody的结合体，方法上的处理请求的注解同样使用@RequestMapping、@GetMapping、@PostMapping等

6.Spring Boot的约定优于配置

约定优于配置是一种软件设计的范式，之前要使用大量的xml文件的配置，配置比较复杂，配置的操作跟我们本身的业务开发没有太大关系，后来出现了Spring Boot中的IOC容器，在服务启动时就可以帮我们管理对象，并且我们可以将一些配置文件写在yml里面，通过解析yml文件来读取配置文件，总的来说，约定优于配置，是一个比较常见的软件设计思想，它的核心本质都是为了去更加高效，以及更加便捷的去实现，软件系统的开发和维护

7.Spring Boot如何解决跨域问题

跨域是指浏览器在发送请求的时候，由于浏览器同源策略的限制，只能访问同源的资源，而不能访问其他源的资源。解决方式有两种：第一种，通过@CrossOrigin（origins=“http：//localhost:8080”）注解指定允许哪些origins允许跨域。第二种，使用WebMvcConfigurer接口来重写addCorssMappings方法来配置允许跨域的请求源

8.spring里面的事务和分布式事务的使用如何区分?

首先在spring里面并没有提供事务，它只是提供了对数据库事务管理的一个封装，我们可以通过配置 `@Transactional`注解在方法或类上标注事务的属性，使得开发人员可以只关心业务的开发，不需要再去关心连接的获取、连接的关闭、事务的提交、事务的回滚这样一些操作，我们可以更加

聚焦在业务的开发层面，所以spring里面的事务，本质上是数据库层面的一个事务，而这种事务管理，主要是针对于单个数据库里面的多个数据表的操作，去满足一个事务的ACID（原子性，一致性，隔离性，持久性）特性。而分布式事务是解决多个数据库事务操作的一个数据一致性问题，传统的关系数据库，不支持跨库的事务操作，所以需要引入分布式事务的解决方案，而spring里面并没有提供分布式事务的场景支持，所以spring里面的事务和分布式事务，在使用上并没有直接的关联关系，但我们可以使用一些主流的分布式事务解决框架，比如seata，集成到spring生态里面，去解决分布式事务的一个问题

9.spring中有两个id相同的bean会报错吗，如果会，会在哪个阶段报错?

会报错，因为spring里边id是唯一的，启动时会去验证id的唯一性，一旦重复就会报错，spring3.x版本中提供了一个@Configuration主键去声明一个配置类，然后使用@Bean这个注解，可以实现spring加载第一个Bean对象时不报错，后面有重复名字的Bean的实例就不会再注册了

10.spring中Bean的作用域有哪些?

Bean的生命周期一般有两种，singleton（单例），prototype（原型），Bean生命周期的功能，主要有三个选项，1.request，它是针对每一次http请求都会创建一个新的Bean。
2.session，以session会话为纬度，同一个session共享同一个Bean实例，不同的session，产生不同的Bean实例 3.globalSession，它是针对于全局session的一个纬度，共享同一个Bean的实例

11. Spring 事务的传播行为

1. REQUIRED：表示如果当前存在一个事务，则加入该事务，否则将新建一个事务；
2. REQUIRES_NEW：表示不管是否存在事务，都创建一个新的事务，原来的挂起，新的执行完毕，继续执行老的事务；
3. SUPPORTS：表示如果当前存在事务，就加入该事务；如果当前没有事务，那就不使用事务；
4. NOT_SUPPORTED： 表示不使用事务；如果当前存在事务，就把当前事务暂停，以非事务方式执行；
5. MANDATORY：表示必须在一个已有的事务中执行，如果当前没有事务，则抛出异常；
6. NEVER：表示以非事务方式执行，如果当前存在事务，则抛出异常；
7. NESTED：这个是嵌套事务； 如果当前存在事务，则在嵌套事务内执行； 如果当前不存在事务，则创建一个新的事务； 嵌套事务使用数据库中的保存点来实现，即嵌套事务回滚不影响外部事务，但外部事务回滚将导致嵌套事务回滚

12.Spring Bean的注入方式

1.使用xml方式进行注入，但这种方式现在已经很少用到
2.一般用注解的方式比较多，一般使用@ComponentScan注解来扫描声明@Controller、@Service、@Repository、@Component注解的类

13.过滤器和拦截器的区别?

拦截器和过滤器都是 AOP 编程思想的体现，都能实现权限检查、日志记录等； 他们的区别主要有： - 拦截器是基于反射实现，更准确的说是通过 jdk 的动态代理实现； 过滤器是基于函数回调
- 拦截器不依赖于 Servlet 容器； 过滤器依赖于 Servlet 容器，它属于 Servlet 规范规定的；
- 拦截器只能对 Controller 请求起作用； 过滤器则可以对几乎所有的请求起作用；
- 拦截器可以访问 controller 上下文的对象(如 service 对象、数据源等)；过滤器则不可以访问；
- 拦截器可以深入的方法前后、异常抛出前后等，并且可以重复调用； 过滤器只在 Servlet 前后起作用，并且只在初始化时被调用一次

14.SpringMVC 执行流程

(1) 用户发送请求到前端控制 DispatcherServlet，但是前端控制器并不处理请求，而是将请求转发给处理器映射器；
(2) 处理器映射器使用请求的 url(/user.do)去查找(xml、注解)处理器，查找到处理器之后，将 HandlerExecutionChain 返回 DispatcherServlet； (3) DispatcherServlet 请求处理器适配器执行处理器；
(4) 处理器适配器执行处理器；
(5) 处理器执行完成之后，将逻辑视图 ModelAndView 返回给处理器适配器；
(6) 处理器适配器将 ModelAndView 返回给 DispatcherServlet；
(7) DispatcherServlet 请求视图解析器解析逻辑视图 ModelAndView，解析完成之后，返回 View 对象给 DispatcherServlet；

(8) DispatcherServlet 根据 View 和模型数据进行视图渲染，响应用户请求

15.说说你对spring mvc的理解

1.把传统MVC框架里面的Controller控制器做了拆分，分成了前端控制器Dispatcherservlet和后端控制器Controller
2.把Model模型拆分成业务层Service和数据访问层Repository
3.在视图层，可以支持不同的视图

16.为什么使用spring框架

Spring框架的优势
轻量级框架，它有两大核心，IOC和AOP（对IOC和AOP的理解）
IOC/DI
向切面的编程（AOP）（AOP注解有哪些）见鑫哥712文档
MVC框架
事务管理（清谈一下对spring两种事务形式的理解）

17.为什么有些公司禁止使用@Transaction注解，会引申出嵌套式事务怎么解决，编程式事务和声明式事务的问题，还可能通过声明式事务引出AOP

18.SpringCloud中Redis和Mysql如何保持数据一致性

Redis是一个缓存性数据库，我们一般会把一些常用的，不怎么更新的数据放在Redis里面，通过Redis减少与数据库的交互，redis与数据库数据保持一致的方式有两种，1.先更新数据库，再更新缓存 2.先删除缓存，再更新数据库。如果先更新数据库再更新缓存，如果缓存更新失败，就会导致数据库跟Redis中的数据不一致，如果先删除缓存再更新数据库，我们查询缓存里边的数据不存在，就会将数据库的数据同步到Redis里边，如果在极端情况下，可以通过消息中间件RabbitMQ队列的方式来保证数据的一致性

19.说一说对SpringCloud的理解

总体来说，微服务是一种架构风格，它提倡将单一应用程序划分成一组小的服务，服务之间互相协调、互相配合； 对于一个大型复杂的业务系统： 它的业务功能可以拆分为多个独立的服务；各个服务之间是松耦合的； 通过远程协议进行通信（异步、同步）； 各个微服务均可被独立部署、扩容、升降级。

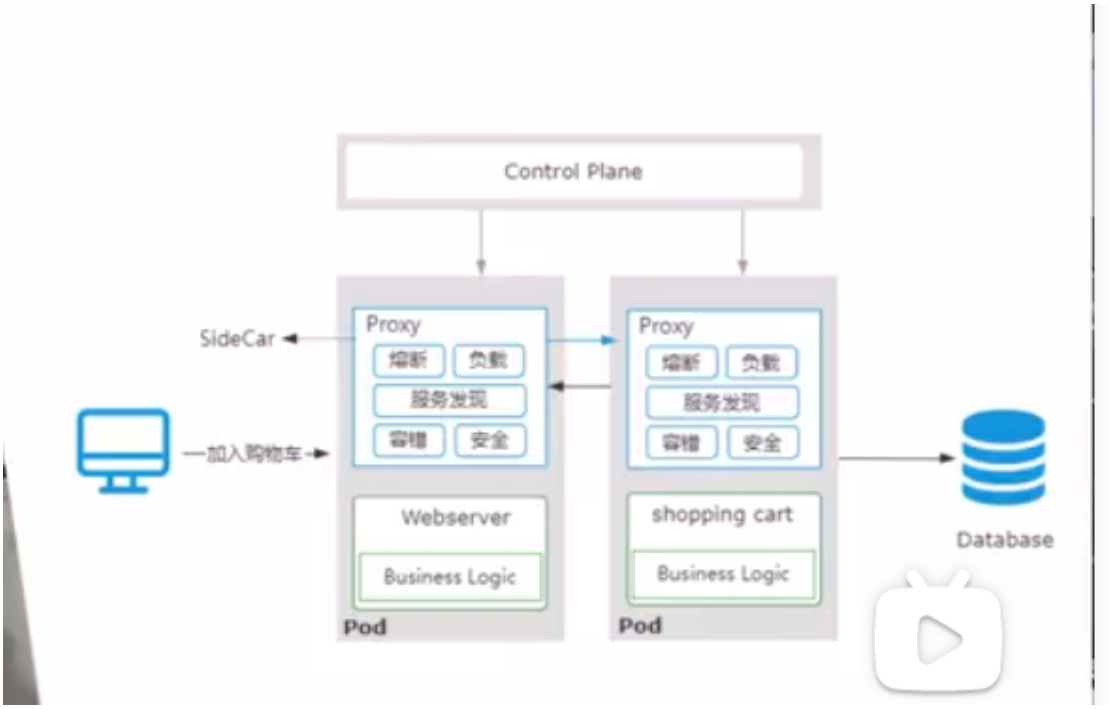
20.RPC协议和HTTP协议有什么区别?

首先，RPC协议和HTTP协议都是通过调用远程服务器进行数据获取或通讯的，PRC的中文是远程过程调用协议，它的核心 目标是为了让开发人员进行远程方法调用的时候就像调用本地方法一样去调用。
HTTP协议主要是浏览器和web服务器之间的通信，去设计的一个通信协议，底层依然采用TCP协议传输，HTTP协议就是实现跟web服务器之间的传输
区别：HTTP是通信协议，RPC是远程调用协议，RPC底层是基于TCP和HTTP协议进行通信的

21.说一下你对SPI机制的理解

22.什么是服务网格

服务网格也就是Service Mesh，它是专门用来处理服务端通信的一个基础设施层，主要是处理服务之间的一个通信，并且负责实现服务之间可靠性的调用，Service Mesh称为第三代微服务架构，会将不同的应用拆分到不同的服务上，然后部署在Docker容器和Kubernetes集群上
我们现在比较流行的SpringCloud及SpringCloudAlibaba，SpringCloud提供了很多组件提高服务之间的通信及容错，但对于开发人员来说，这种配置比较复杂，之所以称为第三代服务架构为服务网格，是因为它的通信、容错、服务治理是由sidcar代理的，多个服务之间的通信联系起来就是一个服务网格
第三代微服务架构



23.说一说你对一致性Hash算法的理解

24.谈一下你对分布式服务和微服务的理解

分布式服务是将任务分发到不同的服务节点上并进行处理，处理完毕后返回到前端，用户是感知不到的，微服务本身就是一个分布式服务，它是将不同的业务模块分配到对应的开发人员进行维护、更新、迭代，提高对业务进行的一个处理，并且实现服务之间的解耦

25.Nacos配置更新的工作流程

Nacos采用的是长轮询的方式。也就是说，由Nacos Client向Nacos Server端去发起配置更新查询的请 求。所谓长轮询，就是客户端发起一次轮询请求到服务器端，当服务器端的配置没有任何变更的时候，这个连接会一直打开，直到服务端有配置变更或者连接超时之后才返回。

26.你理解的服务降级是怎么做的

Sentinel实现服务限流和服务降级

27.服务注册中心是AP还是CP

28.什么是 Redis？它主要用来什么的?

Redis是一个基于Key-Value存储结构的Nosql开源内存数据库。
它提供了5种常用的数据类型，String、Set、ZSet、Hash、List。
针对不同的结构，可以解决不同场景的问题。
因此它可以覆盖应用开发中大部分的业务场景，比如top10问题、好友关注列表、热点话题等。
其次，由于Redis是基于内存存储，并且在数据结构上做了大量的优化所以IO性能比较好，在实际开
发中，会把它作为应用与数据库之间的一个分布式缓存组件。
并且它又是一个非关系型数据的存储，不存在表之间的关联查询问题，所以它可以很好的提升应用程
序的数据IO效率。
最后，作为企业级开发来说，它又提供了主从复制+哨兵、以及集群方式实现高可用在Redis集群里
面，通过hash槽的方式实现了数据分片，进一步提升了性能。

29.Redis 为什么这么快?

1.基于内存存储实现
2.高效的数据结构
3.合理的数据编码

4.合理的线程模型

30.什么是Redis缓存击穿、Redis缓存穿透、Redis缓存雪崩?

缓存穿透通俗点说，读请求访问时，缓存和数据库都没有某个值，这样就会导致每次对这个值进行查询请求都会穿透到数据库

缓存雪崩：指缓存中数据大批量到过期时间，而查询数据量巨大，请求都直接访问数据库，引起数据库压力过大甚至 down 机

缓存击穿：指热点key 在某个时间点过期的时候，而恰好在这个时间点对这个 Key 有大量的并发请求过来，从而大量的请求打到 db。

31.Redis 过期策略和内存淘汰策略

2.6. Redis 过期策略和内存淘汰策略

32.说说 Redis 的常用应用场景

- 缓存
- 排行榜
- 计数器应用
- 共享Session
- 分布式锁
- 社交网络
- 消息队列
- 位操作

33.Redis持久化有哪几种方式，怎么选?

首先，Redis本身是一个基于Key-Value结构的内存数据库，为了避免Redis故障导致数据丢失的问题，

所以提供了RDB和AOF两种持久化机制。

RDB是通过快照的方式来实现持久化的，也就是说会根据快照的触发条件，把内存里面的数据快照写

入到磁盘，以二进制的压缩文件进行存储。

RDB快照的触发方式有很多，比如执行bgsave命令触发异步快照，执行save命令触发同步快照，同

步快照会阻塞客户端的执行指令。

根据redis.conf文件里面的配置，自动触发bgsave主从复制的时候触发AOF持久化，它是一种近乎实时的方式，把Redis Server执行的事务命令进行追加存储。

简单来说，就是客户端执行一个数据变更的操作，Redis Server就会把这个命令追加到aof缓冲区的

末尾，然后再把缓冲区的数据写入到磁盘的AOF文件里面，至于最终什么时候真正持久化到磁盘，
是根据刷盘的策略来决定的。
另外，因为AOF这种指令追加的方式，会造成AOF文件过大，带来明显的IO性能问题，所以Redis针
对这种情况提供了AOF重写机制，也就是说当AOF文件的大小达到某个阈值的时候，就会把这个文
件里面相同的指令进行压缩。
因此，基于对RDB和AOF的工作原理的理解，我认为RDB和AOF的优缺点有两个。
RDB是每隔一段时间触发持久化，因此数据安全性低，AOF可以做到实时持久化，数据安全性较高
RDB文件默认采用压缩的方式持久化，AOF存储的是执行指令，所以RDB在数据恢复的时候性能比AOF要好

34.什么是消息中间件?

消息中间件就是MQ，我们用的比较多的是RabbitMQ，首先我们用的话建立和RabbitMQ服务器连接，声明消息队列，MQ由生产者、消费者、代理，三部分组成，通过生产者将消息发送到队列里，消费者使用@Rabbit MQListener对特定队列进行监听，把获取到的消息根据格式进行解析，将解析的数据存放到数据库或者推送到前端

35.MQ有哪些应用场景?

1.应用解耦
2.流量削峰
3.异步处理
4.消息通讯
5.远程调用
应用解耦：没有MQ之前，服务之间直接互相调用，例如订单系统调用库存系统，如果库存系统出现问题，下单成功，库存扣减失败，订单和库存存在耦合，使用MQ之后，我们可以在两个系统之间建立一个通道，把消息放到通道中，等待系统消费，如果当前系统发生故障，消息不会丢失，系统恢复以后，消息会继续消费，这样两个系统之间就解耦了。
流量削峰：一般用于秒杀，假设系统每秒最多可以处理20个请求，但是每秒有50个请求过来，我们将剩下的30个放入消息队列，慢慢处理。如果消息队列超过最大数量，可以直接抛弃用户请求或跳转到错误页面
异步处理：例如之前注册成功后，想给客户发短信或者邮件，需要串行执行，比较耗时，现在有了消息队列，我们可以把请求的消息放到队列中，通过监听队列可以同时实现短信和邮件的业务处理
消息通讯：MQ有一套完整的消息传输机制，有生产者、消费者，队列等，能够保证消息不丢失

36.消息中间件如何保证消息不丢失?

首先消息中间件由生产者、存储端、消费者组成，主要保证这三部分消息不丢失，消息就不会丢失，生产者可以采用同步方式发送，send 消 息方法返回成功状态，就表示消息正常到达了存储端Broker。如果 send 消息异常或者返回非成功状态，可以重试。可以使用事务消息，RocketMQ -事务消息机制就为了保证零丢失来设计。存储端确保消息持久化到磁盘，可以用刷盘机制，刷盘机制可以分同步刷盘和异步刷盘。消息阶段执行完业务逻辑，再反馈回Broker说消费成功，这样才可以保证消费阶段消息不丢失

37.消息队列如何保证消息的有序

38.消息队列有可能发生重复消费，如何避免，如何做到幂等

首先出现重复消费的原因有以下两点：1.一次性拉取多条消息，导致处理时间过长，就会发生重复消费，解决这个问题的方法就是避免一次性拉取多条消息，提高消息的处理时间。2.消费者去消费这条消息时，没有及时给代理标记状态，导致代理认为这条消息没有消费，所以也发生了重复消费。如果想解决这个问题，就要把offter更新。

39. Mybatis是如何进行分页的?

有三种方式来实现分页：
第一种，直接在Select语句上增加数据库提供的分页关键字，然后在应用程序里面传递当前页，以及每页展示条数即可。
第二种，使用Mybatis提供的RowBounds对象，实现内存级别分页
第三种，基于Mybatis里面的Interceptor拦截器，在select语句执行之前动态拼接分页关键字。

40.MyBatis中#{}和${}的区别?

- #{}内部使用预编译 PreparedStatement 机制执行 SQL，支持?占位符，可以防止 SQL 注入;
- ${}内部采用 Statement 机制将参数和 SQL 拼装在一起发送执行，不支持?占位符，参数只能拼接；
- 只要能用占位符的地方，都建议使用#{}
- 如果是表名或字段名或者 like ‘’ 位置，建议使用${}

41.MyBatis里面的缓存机制

一级缓存:
- 基于 PerpetualCache 的 HashMap 本地缓存，其存储作用域为 SqlSession，当 SqlSession flush 或 close 之后，该 Session 中的所有 Cache 就将清空； - 另外不同的 SqlSession 之间的缓存数据是互相不影响的；

二级缓存：
- 二级缓存开启方式是在 Mapper 的配置文件配置<cache />；
- 与一级缓存其机制相同，不同在于其存储作用域为 Mapper，并且可自定义存储源，如 Ehcache；
- 二级缓存是跨 SqlSession 的，多个 SqlSession 可以共用二级缓存；
- 进行了“添加、更新和删除”操作后，会刷新对应的缓存的数据。

42. MyBatis何时使用一级缓存，何时使用二级缓存?

对于一级缓存，可以在需要时灵活使用，在同一个 SqlSession 中进行多个查询操作，提高查询效率。但是需要注意的是，一级缓存只能在当前 SqlSession 中生效，不能在多个 SqlSession 之间共享缓存数据。所以，一级缓存适用于短暂的数据使用场景，不能用于多个请求之间的数据共享。

对于读多写少的场景，建议开启二级缓存。但是，应该注意二级缓存的配置，避免缓存数据过多导致内存溢出等问题。对于数据比较频繁更新、插入和删除的场景，应该谨慎使用二级缓存，避免数据的不一致性