# 2020/10/22 中软华腾

## SpringCloud的组件有什么,分别有什么作用

1. Zuul:服务注册,服务发现,服务续约,服务下线
2. Ribben:客户端负载均衡,负载服务之间的调用
3. Config:配置中心,微服务架构中的服务实例非常的多，服务的配置文件分散在每个服务中，每次修改服务的配置文件和重新服务实例都是一个很麻烦的工作，Spring Cloud Config作为分布式配置管理中心就是用来统一的管理服务的配置文件。
4. Bus:消息总线,消息总线是在微服务中给各个微服务广播消息的一个组件，我们使用消息总线构建一个消息中心，其他微服务来接入到消息中心，当消息总线发起消息，接入的微服务都可以收到消息从而进行消费。
5. Feign:封装了Ribben使参数传递更简单,需要在主服务打上@EnableFeignClients [并且需要在Feign接口上打@FeignClient(Value=”服务名”根据注册中心服务名查找,降级方法=”降级类.class”)](mailto:并且需要在Feign接口上打@FeignClient(Value=\”服务名\”,降级方法=\”降级类.class\”))
6. Hytrix:熔断器,防止雪崩效应(某一个服务发生故障会导致调用它的服务跟着异常，然后导致整个调用链调用的异常，甚至导致整个微服务瘫痪 ),将出现故障的服务通过熔断、降级等手段隔离开来,他有
7. 资源隔离:

资源隔离包括线程池隔离和信号量隔离，作用是限制调用分布式服务的资源使用，某一个调用的服务出现问题不会影响其他服务调用 ，这里可以简单的理解为资源隔离就是限制请求的数量



1. 服务熔断:

熔断机制是对服务链路的保护机制，如果链路上的某个服务不可访问，调用超时，发生异常等，服务会触发降级返回托底数据，然后熔断服务的调用(失败率达到某个阀值服务标记为短路状态)，当检查到该节点能正常使用时服务会快速恢复。

1. 降级机制:

超时降级、资源不足时(线程或信号量)降级，降级后可以配合降级接口返回托底数据。

1. 缓存:

提供了请求缓存、请求合并实现 ， 在高并发的场景之下，Hystrix请求缓存可以方便地开启和使用请求缓存来优化系统，达到减轻高并发时请求线程的消耗、降低请求响应时间的效果。

## Feign是如何实现服务调用的,如何使用

Feign是一个声明式的http客户端，使用Feign可以实现声明式REST调用，它的目的就是让Web Service调用更加简单。Feign整合了Ribbon和SpringMvc注解，这让Feign的客户端接口看起来就像一个Controller。

使用:主配置类增加@EnableFeignClients标签 , 其value属性可以指定Feign的客户端接口的包,当然也可以省略value属性,Feign的客户端接口是用来调用微服务的,需要加上注解[并且需要在Feign接口上打@FeignClient(Value=”服务名”根据注册中心服务名查找,降级方法=”降级类.class”)](mailto:并且需要在Feign接口上打@FeignClient(Value=\”服务名\”,降级方法=\”降级类.class\”)),Feign的接口会被动态代理

## Hytrix是什么,他有什么作用,如何开启

Hytrix是熔断器,是有服务熔断,降级机制,缓存,资源隔离的作用,一般结合Fegin整合了Hytrix一般在开启了Feign代理接口的服务的yml配置中开启feign:hystrix:enabled: true

## 一个多线程应用,其中一个方法只能让两个方法进入,如何实现

　使用Semaphore类,Semaphore也是一个线程同步的辅助类，可以维护当前访问自身的线程个数，并提供了同步机制。使用Semaphore可以控制同时访问资源的线程个数

## 数据库的ABCD组合索引,当Sql条件为Where C = ? And D = ?时,会调用索引吗

不会调用索引,因为数据库索引有最左匹配原所有的参数条件都要包含A=?才会调用

## SpringBoot的自动装配原理

**启动类注解SpringbootApplication是一个复合注解,其中包含了@SpringBootConfiguration/@ComPonentSan/@EnableAutoConfiguration,其中@SpringBootConfiguration注解相当于@Configuration声明是一个配置之类,@ComPonentSan扫描组件,SpringBoot主启动类的同级路径及子路径,@EnableAutoConfiguration是主要做自动配置的注解,然而这个注解也是一个复合注解,其中包含@Import({AutoConfigurationImportSelector.class}),这里导入了这个类,这个类实现了DeferredImportSelector接口,然后复写了selectImports方法,方法中通过getAutoConfigurationEntry返回我们所需要的配置信息(Entry对象)还包含去重和检查排除类的操作,而这个方法中又调用了getCandidateConfigurations方法,返回了配置类全包名的list集合,然后这个方法通过loadFactoryName方法加载了META-INF/spring.factories文件,这个文件包含了我们所需要的配置类的全包名,然后将全路径包名返回,最终返回了一个字符串数组,spring就根据全包名自动装配上这些配置文件下定义的bean对象(@SpringBootConfiguration ?)，从而达到了自动装配的目的.**

## 线程池参数有什么

1.corePoolSize：核心线程数

核心线程会一直存活，及时没有任务需要执行。

当线程数小于核心线程数时，即使有线程空闲，线程池也会优先创建新线程处理。

设置allowCoreThreadTimeout=true（默认false）时，核心线程会超时关闭。

2.queueCapacity：任务队列容量（阻塞队列）

当核心线程数达到最大时，新任务会放在队列中排队等待执行。

3.maxPoolSize：最大线程数

当线程数>=corePoolSize，且任务队列已满时。线程池会创建新线程来处理任务。

当线程数=maxPoolSize，且任务队列已满时，线程池会拒绝处理任务而抛出异常。

4.keepAliveTime：线程空闲时间

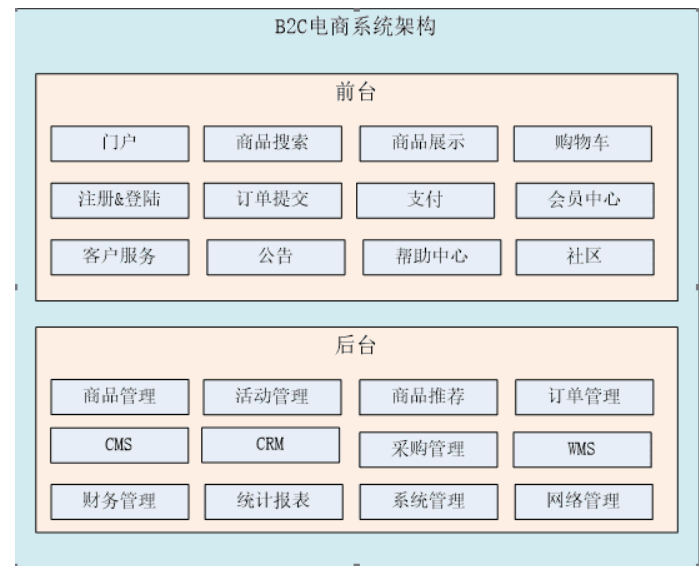
当线程空闲时间达到keepAliveTime时，线程会退出，直到线程数量=corePoolSize。

如果allowCoreThreadTimeout=true，则会直到线程数量=0。

5.allowCoreThreadTimeout：允许核心线程超时

6.rejectedExecutionHandler：任务拒绝处理器

## 8.电商项目架构



后台管理系统：管理商品、订单、类目、商品规格属性、用户管理以及内容发布等功能。  
前台系统：用户可以在前台系统中进行注册、登录、浏览商品、首页、下单等操作。  
会员系统：用户可以在该系统中查询已下的订单、收藏的商品、我的优惠券、团购等信息。  
订单系统：提供下单、查询订单、修改订单状态、定时处理订单。  
搜索系统：提供商品的搜索功能。  
单点登录系统：为多个系统之间提供用户登录凭证以及查询登录用户的信息

# 2020/10/23 中软国际 华为外包 一面

## 说一下解释执行和编译执行

解释执行:将编译好的字节码,一行一行的翻译为机器码,不用等待

编译执行:以方法为单位,一次性将字节码翻译为机器码后执行,需要等待,但是效率高

目前主流的JVM都是混合模式,解释执行和编译执行配合运行

对于重要的,重复调用的方法会使用编译执行,将其翻译为机器码,这样可以调高执行效率,对于不重要的代码使用解释执行,为了不干扰程序的正常运行，JIT编译(及时编译)时放在额外的线程中执行的，HotSpot根据实际CPU的资源，以 1:2的比例分配给C1和C2线程数。在计算机资源充足的情况，字节码的解释运行和编译运行时可以同时进行，编译执行完后的机器码会在下次调用该方法时启动，已替换原本的解释执行

细节看资料:https://www.cnblogs.com/lingz/p/9394238.html

## 软强虚弱四种引用

强引用:开发中用到最多的一种,当内存空间不足时gc也不会回收强引用,会报OutOfMemery错误,让程序终止也不会回收,如果想中断或者回收强引用,可以显示的赋值为NULL这样就会被回收

软引用:如果gc root可达且内存足够的时候gc不会回收软引用,当内存不足时会回收

弱引用:只要发生了GC都会被回收,ThreadLocal中的静态内部类ThreadLocalMap里的Entry是一个软引用的继承类

虚引用:虚引用跟之前的三种不同,他不影响生命周期.如果一个对象跟虚引用关联,跟没有关联一样,任何时候都会被垃圾器回收

## 怎看看sql是否用了索引

Explain查看执行计划,如果其中key字符为Null则没有索引,key一般为主键名或索引名

## sql执行器的工作原理

1. 连接器:管理连接,权限验证(账号密码是否正确)
2. 查询缓存:Mysql会将之前执行过的语句以key-value的形式存储起来,并将SQL语句作为Key值,第二次查询的时候可以直接查询结果
3. 分析器:识别你Sql语句是否正确,如果错误直接报错
4. 优化器:在表中有多个索引的时候,决定用哪个索引,或在连表查询的时候,决定连接顺序
5. 执行器:首先判断有没有权限操作这张表,如果有权限就会根据引擎去调用这个引擎的接口,可以在数据库慢查询日志中看到一个Rows\_examined的字段,其中记录了这条sql扫描了多少行.这个值在每次执行的时候会累加

## int的范围,什么时候Interger可以用==比较,原理是什么

Int范围:-2147483648到+-2147483648。 -2^31 -- 2^31-1

值范围在-128-127之间可以直接用==比较,因为这个时候缓存在常量池,共用一个地址所以可以==比较

## 说一下你对线程的认识,说一下线程同步

线程是一个程序执行的最小进程,线程的创建有三种方式

1. 继承Thread类 重写run方法,
2. 实现Runnable接口 没有返回值
3. 实现Callable接口(是Runnable接口的补充) 有返回值

线程的状态有5中:新建(New),就绪状态(Runnable),运行状态(Running),阻塞状态(Blocked),死亡状态(Dead)

线程同步有四种方法:用synchronized关键词修饰方法(同步方法),用synchronized修饰代码块(同步代码块),上锁Lock.lock,使用Voilate修饰要被操作的变量

## java.util.concurrent包你有多少了解

详细了解看:

https://blog.csdn.net/windsunmoon/article/details/36903901?ops\_request\_misc=%257B%2522request%255Fid%2522%253A%2522160343727419724835852760%2522%252C%2522scm%2522%253A%252220140713.130102334.pc%255Fall.%2522%257D&request\_id=160343727419724835852760&biz\_id=0&utm\_medium=distribute.pc\_search\_result.none-task-blog-2~all~first\_rank\_v2~rank\_v25-1-36903901.pc\_search\_result\_cache&utm\_term=java.util.concurrent%E5%8C%85%E8%A7%A3%E6%9E%90&spm=1018.2118.3001.4187

## 线程池如何创建

常见的线程池有四种:

定长线程池(FixdThreadPool):只有核心线程数的,线程数量固定,执行完立刻回收,任务队列为链表结构的有界队列(队列有最大上限),使用场景:控制线程最大并发数

定时线程池(ScheduleThreadPool):核心线程数固定,非核心线程数无限,执行完限制10ms回收,是延时阻塞队列(会将短时间的任务排在前面),使用场景:执行定时任务或周期性任务

可缓存线程池(CachedThreadPool):无核心线程,无上限非核心线程,执行结束60S后回收,队列为不存储元素的阻塞队列.使用场景(执行大量耗时少的任务).

单线程化线程池(SingleThreadPool0):核心线程只有一个,无非核心线程,执行完立刻回收,任务队列为链表结构的有界队列,使用场景:不适合并发但会引起IO阻塞性以及对影响UI线程的操作,如数据库操作,文件操作

其中定时线程池和可缓存线程池会因为线程数过大而造成内存溢出,定长线程池和单线程线程池会因为任务队列过大而内存溢出

现在都不推荐使用以上几种线程池,推荐使用ThreadPoolExcuter自己写线程池

(此公司135加班到8.30,24不加班,周末如果加班双倍工资,月加班50小时,有班车接送,但是收费高)