**一、Maven有哪些优点和缺点**

优点如下：

简化了项目依赖管理：

易于上手，对于新手可能一个"mvn clean package"命令就可能满足他的工作

便于与持续集成工具（jenkins）整合

便于项目升级，无论是项目本身升级还是项目使用的依赖升级。

有助于多模块项目的开发，一个模块开发好后，发布到仓库，依赖该模块时可以直接从仓库更新，而不用自己去编译。

maven有很多插件，便于功能扩展，比如生产站点，自动发布版本等

缺点如下：

maven是一个庞大的构建系统，学习难度大

maven采用约定优于配置的策略（convention over configuration），虽然上手容易，但是一旦出了问题，难于调试。

当依赖很多时，m2eclipse 老是搞得Eclipse很卡。

中国的网络环境差，很多repository无法访问，比如google code， jboss 仓库无法访问等。

**5.3.2 Maven坐标**

在Maven仓库中唯一定位到一个Maven工程（jar包）。

groupid：公司或组织域名+项目名称

artifactid：模块名称

version：版本

5.4.5 依赖的范围

 compile

scope标签的默认值

如果我们需要导入Spring、MyBatis、Log4j等等这样在主体程序中正式使用的jar包，需要使用compile范围。

空间角度：对main目录、test目录下的代码都有效

时间角度：对开发阶段、在服务器运行阶段都有效

 test

用于测试的jar包，使用test范围。

空间角度：对main目录无效，对test目录有效

时间角度：对开发阶段有效，不参与服务器部署

 provided

用于在开发时使用servlet或jsp相关的API的jar包，使用provided范围。

空间角度：对main目录、test目录下的代码都有效

时间角度：对开发阶段有效，不参与服务器部署

**什么是Redis？**

Redis 是一个使用 C 语言写成的，开源的 key-value 数据库。。和Memcached类似，它支持存储的value类型相对更多，包括string(字符串)、list(链表)、set(集合)、zset(sorted set –有序集合)和hash（哈希类型）。

这些数据类型都支持push/pop、add/remove及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，redis支持各种不同方式的排序。与memcached一样，为了保证效率，数据都是缓存在内存中。区别的是redis会周期性的把更新的数据写入磁盘或者把修改操作写入追加的记录文件，并且在此基础上实现了master-slave(主从)同步。目前，Vmware在资助着redis项目的开发和维护。

**1. 使用Redis有哪些好处？**

(1) 速度快，因为数据存在内存中，类似于HashMap，HashMap的优势就是查找和操作的时间复杂度都是O(1)

(2) 支持丰富数据类型，支持string，list，set，sorted set，hash

(3) 支持事务，操作都是原子性，所谓的原子性就是对数据的更改要么全部执行，要么全部不执行

(4) 丰富的特性：可用于缓存，消息，按key设置过期时间，过期后将会自动删除

**3.悲观锁和乐观锁**

在使用WATCH命令监控一个KEY后，当前队列中的命令会由于外部命令的执行而放弃，这是乐观锁的体现。

悲观锁

认为当前环境非常容易发生碰撞，所以执行操作前需要把数据锁定，操作完成后释放锁，其他操作才可以继续操作。

乐观锁

认为当前环境不容易发生碰撞，所以执行操作前不锁定数据，万一碰撞真的发生了，那么放弃自己的操作。

**1.读写分离的好处：**

性能优化：主服务器专注于写操作，可以用更适合写入数据的模式工作；同样，从服务器专注于读操作，可以用更适合读取数据的模式工作。

强化数据安全，避免单点故障：由于数据同步机制的存在，各个服务器之间数据保持一致，所以其中某个服务器宕机不会导致数据丢失或无法访问。从这个角度说参与主从复制的Redis服务器构成了一个集群。

**1.负载均衡**

当用户访问量非常大时，我们需要使用多台Tomcat来构成集群，**由Nginx负责**将负载分摊到每一台Tomcat服务器上。

### 一. 什么是springboot

1.用来简化spring应用的初始搭建以及开发过程 使用特定的方式来进行配置（properties或yml文件）  
2.创建独立的spring引用程序 main方法运行  
3.嵌入的Tomcat 无需部署war文件  
4.简化maven配置  
5.自动配置spring添加对应功能starter自动化配置

**Spring Boot 的配置文件有哪几种格式？它们有什么区别？**

    .properties 和 .yml，它们的区别主要是书写格式不同。

1).properties

app.user.name = javastack  
    2).yml

    app:  
      user:  
        name: javastack

    另外，.yml 格式不支持 @PropertySource 注解导入配置。

### 三. springboot自动配置的原理

1.@EnableAutoConfiguration这个注释告诉SpringBoot“猜”你将如何想配置Spring,基于你已经添加jar依赖项。如果spring-boot-starter-web已经添加Tomcat和Spring MVC,这个注释自动将假设您正在开发一个web应用程序并添加相应的spring设置。会自动去maven中读取每个starter中的spring.factories文件 该文件里配置了所有需要被创建spring容器中的bean

2.使用@SpringbootApplication注解 可以解决根类或者配置类（我自己的说法，就是main所在类）头上注解过多的问题，一个@SpringbootApplication相当于@Configuration,@EnableAutoConfiguration和 @ComponentScan 并具有他们的默认属性值

### 六.微服务

以前的模式是 所有的代码在同一个工程中 部署在同一个服务器中 同一个项目的不同模块不同功能互相抢占资源

微服务 将工程根据不同的业务规则拆分成微服务 微服务部署在不同的机器上 服务之间进行相互调用

Java微服务的框架有 dubbo（只能用来做微服务），spring cloud（提供了服务的发现，断路器等）

**微服务的特点：**

按业务划分为一个独立运行的程序，即服务单元

服务之间通过HTTP协议相互通信

自动化部署

可以用不同的编程语言

可以用不同的存储技术

服务集中化管理

微服务是一个分布式系统

**微服务的优势**：

1、将一个复杂的业务拆分为若干小的业务，将复杂的业务简单化，新人只需要了解他所接管的服务的代码，减少了新人的学习成本。

2、由于微服务是分布式服务，服务于服务之间没有任何耦合。微服务系统的微服务单元具有很强的横向拓展能力。

3、服务于服务之间采用HTTP网络通信协议来通信，单个服务内部高度耦合，服务与服务之间完全独立，无耦合。这使得微服务可以采用任何的开发语言和技术来实现，提高开发效率、降低开发成本。

4、微服务是按照业务进行拆分的，并有坚实的服务边界，若要重写某一业务代码，不需了解所以业务，重写简单。

5、微服务的每个服务单元是独立部署的，即独立运行在某个进程中，微服务的修改和部署对其他服务没有影响。

6、微服务在CAP理论中采用的AP架构，具有高可用分区容错特点。高可用主要体现在系统7x24不间断服务，他要求系统有大量的服务器集群，从而提高系统的负载能力。分区容错也使得系统更加健壮。

---------------------

### 七. springcloud如何实现服务的注册和发现

服务在发布时 指定对应的服务名（服务名包括了IP地址和端口） 将服务注册到注册中心（eureka或者zookeeper）

这一过程是springcloud自动实现 只需要在main方法添加@EnableDisscoveryClient 同一个服务修改端口就可以启动多个实例

调用方法：传递服务名称通过注册中心获取所有的可用实例 通过负载均衡策略调用（ribbon和feign）对应的服务

### 八.ribbon和feign区别

Ribbon添加maven依赖 spring-starter-ribbon 使用@RibbonClient(value="服务名称") 使用RestTemplate调用远程服务对应的方法  
feign添加maven依赖 spring-starter-feign 服务提供方提供对外接口 调用方使用 在接口上使用@FeignClient("指定服务名")

Ribbon和Feign的区别：  
Ribbon和Feign都是用于调用其他服务的，不过方式不同。  
1.启动类使用的注解不同，Ribbon用的是@RibbonClient，Feign用的是@EnableFeignClients。  
2.服务的指定位置不同，Ribbon是在@RibbonClient注解上声明，Feign则是在定义抽象方法的接口中使用@FeignClient声明。  
3.调用方式不同，Ribbon需要自己构建http请求，模拟http请求然后使用RestTemplate发送给其他服务，步骤相当繁琐。  
Feign则是在Ribbon的基础上进行了一次改进，采用接口的方式，将需要调用的其他服务的方法定义成抽象方法即可，  
不需要自己构建http请求。不过要注意的是抽象方法的注解、方法签名要和提供服务的方法完全一致。

### 九.springcloud断路器的作用

当一个服务调用另一个服务由于网络原因或者自身原因出现问题时 调用者就会等待被调用者的响应 当更多的服务请求到这些资源时，导致更多的请求等待， 这样就会发生连锁效应（雪崩效应） 断路器就是解决这一问题断路器  
有完全打开状态  
一定时间内 达到一定的次数无法调用 并且多次检测没有恢复的迹象 断路器完全打开，那么下次请求就不会请求到该服务半开,短时间内 有恢复迹象 断路器会将部分请求发给该服务 当能正常调用时 断路器关闭,当服务一直处于正常状态 能正常调用 断路器关闭

什么是服务熔断？什么是服务降级？

熔断机制是应对雪崩效应的一种微服务链路保护机制。当某个微服务不可用或者响应时间太长时，会进行服务降级，进而熔断该节点微服务的调用，快速返回“错误”的响应信息。当检测到该节点微服务调用响应正常后恢复调用链路。在SpringCloud框架里熔断机制通过Hystrix实现，Hystrix会监控微服务间调用的状况，当失败的调用到一定阈值，缺省是5秒内调用20次，如果失败，就会启动熔断机制。

服务降级，一般是从整体负荷考虑。就是当某个服务熔断之后，服务器将不再被调用，此时客户端可以自己准备一个本地的fallback回调，返回一个缺省值。这样做，虽然水平下降，但好歹可用，比直接挂掉强。

Hystrix相关注解

@EnableHystrix：开启熔断

@HystrixCommand(fallbackMethod=”XXX”)：声明一个失败回滚处理函数XXX，当被注解的方法执行超时（默认是1000毫秒），就会执行fallback函数，返回错误提示。

**Eureka和zookeeper都可以提供服务注册与发现的功能，请说说两个的区别？**

Zookeeper保证了CP（C：一致性，P：分区容错性），Eureka保证了AP（A：高可用）

1.当向注册中心查询服务列表时，我们可以容忍注册中心返回的是几分钟以前的信息，但不能容忍直接down掉不可用。也就是说，服务注册功能对高可用性要求比较高，但zk会出现这样一种情况，当master节点因为网络故障与其他节点失去联系时，剩余节点会重新选leader。问题在于，选取leader时间过长，30 ~ 120s，且选取期间zk集群都不可用，这样就会导致选取期间注册服务瘫痪。在云部署的环境下，因网络问题使得zk集群失去master节点是较大概率会发生的事，虽然服务能够恢复，但是漫长的选取时间导致的注册长期不可用是不能容忍的。

2.Eureka保证了可用性，Eureka各个节点是平等的，几个节点挂掉不会影响正常节点的工作，剩余的节点仍然可以提供注册和查询服务。而Eureka的客户端向某个Eureka注册或发现时发生连接失败，则会自动切换到其他节点，只要有一台Eureka还在，就能保证注册服务可用，只是查到的信息可能不是最新的。除此之外，Eureka还有自我保护机制，如果在15分钟内超过85%的节点没有正常的心跳，那么Eureka就认为客户端与注册中心发生了网络故障，此时会出现以下几种情况：

①、Eureka不在从注册列表中移除因为长时间没有收到心跳而应该过期的服务。

②、Eureka仍然能够接受新服务的注册和查询请求，但是不会被同步到其他节点上（即保证当前节点仍然可用）

③、当网络稳定时，当前实例新的注册信息会被同步到其他节点。

因此，Eureka可以很好的应对因网络故障导致部分节点失去联系的情况，而不会像Zookeeper那样使整个微服务瘫痪。