面试资料

# PHP

## oop的五大原则，三大特性

五大原则

1、单一职责原则 ：一个类只做一种事情，类应该是小而精，而不是大而全

2、接口隔离原则：把一个大的接口拆分为多个小接口，实现的时候可以通过 implements实现多个接口。

3、开放封闭原则：对修改是封闭的，对扩展是开放的

4、里氏替换原则：子类可以代替父类出现的地方，

5、依赖倒置原则：底层模块应该依赖高层模块，应该依赖于抽象，通过高层模块决定要生成的对象，底层模块依赖高层模块来生成对象。

三大特性：封装、继承、多态

## 接口和抽象类区别

接口就是比“抽象类"还“抽象”的“抽象类”, 可以更加规范的对子类进行约束。

接口不能实例化对象 可以用来声明引用变量的类型

接口中的方法修饰符号 只能是public

抽象类中的抽象方法可以有public protected default

接口中没有构造方法

抽象类可以有构造方法

## 如何修改session的生存时间

在php.ini 中设置 session.gc\_maxlifetime = 1440 //默认时间

代码实现

$lifeTime = 24 \* 3600; // 保存一天

session\_set\_cookie\_params($lifeTime);

session\_start();

## TP的特性有哪些

1.多表查询非常方便,在model中几句代码就可以完成对多表的关联操作

2.融合了smarty模板,使前后台分离

3.支持多种缓存技术,尤其对memcache技术支持非常好

4.命名规范,模型,视图,控制器严格遵循命名规则,通过命名一一对应

5.支持多种url模式

6.内置ajax返回方法,包括xml,json,html等

7.支持应用扩展,类库扩展,驱动扩展等

## 5、请介绍一下laravel框架

答: laravel框架的设计思想比较先进,非常适合应用各种开发模式,作为一个框架,它为你准备好了一切,composer是php的未来,没有composer,php肯定要走向没落

laravel框架最大的特点和优秀之处就是集合了php比较新的特点,以及各种各样的设计模式,Ioc模式,依赖注入等

## 6、请简述一下数据库的优化

答:数据库的优化可以从四个方面来优化:

1.从结构层: web服务器采用负载均衡服务器,mysql服务器采用主从复制,读写分离

2.从储存层: 采用合适的存储引擎,采用三范式

3.从设计层: 采用分区分表,索引,表的字段采用合适的字段属性,适当的采用逆范式,开启mysql缓存

4.sql语句层:结果一样的情况下,采用效率高,速度快节省资源的sql语句执行

## 7、怎么保证促销商品不会超卖

答:这个问题是我们当时开发时遇到的一个难点，超卖的原因主要是下的订单的数目和我们要促销的商品的数目不一致导致的，每次总是订单的数比我们的促销商品的数目要多，当时我们的小组讨论了好久，给出了好几个方案来实现：

使用redis的队列来实现。将要促销的商品数量以队列的方式存入redis中，每当用户抢到一件促销商品则从队列中删除一个数据，确保商品不会超卖。这个操作起来很方便，而且效率极+高，最终我们采取这种方式来实现

## 8、 商城秒杀的实现

答:抢购、秒杀是如今很常见的一个应用场景，主要需要解决的问题有两个：

1 高并发对数据库产生的压力

2 竞争状态下如何解决库存的正确减少（"超卖"问题）

对于第一个问题，已经很容易想到用缓存来处理抢购，避免直接操作数据库，例如使用Redis。第二个问题，我们可以使用redis队列来完成，把要秒杀的商品放入到队列中，因为pop操作是原子的，即使有很多用户同时到达，也是依次执行，文件锁和事务在高并发下性能下降很快，当然还要考虑其他方面的东西，比如抢购页面做成静态的，通过ajax调用接口，其中也可能会出现一个用户抢多次的情况，这时候需要再加上一个排队队列和抢购结果队列及库存队列。高并发情况下，将用户进入排队队列，用一个线程循环处理从排队队列取出一个用户，判断用户是否已在抢购结果队列，如果在，则已抢购，否则未抢购，库存减1，写数据库，将用户入结果队列。

## 9、composer自动加载机制

composer加载核心思想是通过composer的配置文件在引用入口文件(autoload.php)时,将类和路径的对应关系加载到内存中,最后将具体加载的实现注册到spl\_autoload\_register函数中.最后将需要的文件包含进来.

## 10、PHP常见的开发模式

单例模式：

1. $\_instance必须声明为静态的私有变量
2. 构造函数和析构函数必须声明为私有,防止外部程序new 类从而失去单例模式的意义
3. getInstance()方法必须设置为公有的,必须调用此方法 以返回实例的一个引用
4. ::操作符只能访问静态变量和静态函数
5. new对象都会消耗内存
6. 使用场景:最常用的地方是数据库连接。
7. 使用单例模式生成一个对象后， 该对象可以被其它众多对象所使用。
8. 私有的\_\_clone()方法防止克隆对象

单例模式，使某个类的对象仅允许创建一个。构造函数private修饰，

申明一个static getInstance方法，在该方法里创建该对象的实例。如果该实例已经存在，则不创建。比如只需要创建一个数据库连接。

工厂模式：工厂方法或者类生成对象，而不是在代码中直接new。

使用工厂模式，可以避免当改变某个类的名字或者方法之后，在调用这个类的所有的代码中都修改它的名字或者参数。

观察者模式：1：观察者模式(Observer)，当一个对象状态发生变化时，依赖它的对象全部会收到通知，并自动更新。

2：场景:一个事件发生后，要执行一连串更新操作。传统的编程方式，就是在事件的代码之后直接加入处理的逻辑。当更新的逻辑增多之后，代码会变得难以维护。这种方式是耦合的，侵入式的，增加新的逻辑需要修改事件的主体代码。

3：观察者模式实现了低耦合，非侵入式的通知与更新机制

## 11、api接口（签名，验签）

鉴权：验证用户是否拥有访问权限

签名的话可以用（sha1,md5）

接口是怎么做鉴权的（JWT）

接口里有数据加密（对称加密，非对称加密，签名，验签，防刷）

JWT过期解决

## 12、接口安全性的几种方法

非对称加密：rsa,公钥和私钥互换

开放接口的话可以用Oath2.0协议

https协议加密传输，但是速度慢

使用对ip访问实现接口限流，对短时间内同一ip，进行限制

使用token+图片验证码方法实现防止模拟请求

对一些特殊字符实现过滤，防止xss，sql注入

## 13、git

 git add 提交到暂存区

 git commit -m 提交到版本库

 git pull 拉取

 git push 推送

 git status 查看状态

 git merge 分支 合并分支

 git checkout 切换分支

## SQL注入

用户或者黑客通过构建特殊的输入作为参数传入我们的Web应用程序端，而这些输入大都是SQL语法里的一些组合，通过执行SQL语句进而执行攻击者所要的操作，其主要原因是程序员没有细致地过滤用户输入的数据，致使非法数据侵入系统而造成的。

解决：

1、占位符的方式，就是对sql语句进行预处理，然后执行sql语句

2、通过addslashes或者mysql\_real\_escape\_string这两个函数对用户输入的值进行转义处理，把一些特殊的字符转义掉。

## XSS与CSRF

XSS（跨站脚本攻击） 恶意攻击者往Web页面里插入恶意Script代码，当用户浏览该页之时，嵌入其中Web里面的Script代码会被执行，从而达到恶意攻击用户的目的。

CSRF（跨站请求伪造）攻击特点是伪造其他网站的操作，冒用身份比如利用cookie伪造登录凭证，不是窃取cookie，

区别：

(1)CSRF相对于XSS漏洞的危害程度更高一些。

(2)XSS有局限性，而CSRF则不。

(3)CSRF相当于是XSS的基础版，CSRF能做到的XSS都可以做到。

(4)XSS主要指向客户端，而CSRF针对服务端。

(5)XSS是利用合法用户获取其信息，而CSRF是伪造成合法用户发起请求。

## PHP垃圾回收机制



## mvc 开发模式， OOP 编程思想

  oop就是面向对象，是一种程序设计思想。面向对象专注于通过对象的一些方法去解决问题，不同功能用不同对象来扶着解决。

面向对象三大特征：封装，继承，多态

设计模式有三种：创建型模式（简单模式，工厂模式，单例模式），结构性模式，行为型模式

## linux常用命令、并使用linux进行开发、Lnmp环境的部署以及使用；

linux常用命令

  find / -name file1 从 '/' 开始进入根文件系统搜索文件和目录

  shutdown -h now 关闭系统

Mkdir 创建目录

Touch  创建文件不打开

Chmod  更改权限

Vim 文件名编辑文件/有文件则打开，没有文件则创建

Clear 清屏

Tail  文件名 查看文件末尾10行

Pwd  显示现在所在目录

cp 复制文件

Mv 移动文件

Kill 杀死进程

   lnmp是linux 、nginx、mysql、php

## ThinkPHP、Laravel、Yii，mvc面向对象思想；

Thinkphp：

易于上手，有中文文档，框架的兼容性较强，但是功能有限，因此更适合用于中小型项目开发，mvc架构

坏处：对ajax不友好，目录结构混乱，要花时间整理，上手容易，深入学习很难

Laravel：

laravel支持模板引擎，mvc架构，安全性高，支持数据库迁移，laravel框架可以无缝满足各种模式的开源应用程序的开发需求，可以创建高性能WEB应用程序

laravel依赖注入原理：

类的依赖通过构造器或在某些情况下通过传递参数实现调用 ，减少初始化框架的开销

Yii：

1.纯OOP

2.用于大规模Web应用

3.模型使用方便

4.开发速度快，运行速度也快。性能优异且功能丰富

5.使用命令行工具。

6.支持composer包管理工具

Laravel和TP区别：

渲染模板方式不同，

在laravel里考虑到了跨站请求伪造，tp里没有，

Laravel是重路由框架，所有功能都是由路由发起的，只要写路由就能访问，tp必须有控制器才能访问；

Laravel具有强大的社区化扩展；

中间件是laravel的特点，可以实现前后的处理，例如请求和返回

## 负载均衡配置，Nginx 的反向代理

 什么是负载均衡？：负载均衡建立在网络结构之上，提供了一种扩展网络设备和服务器的宽带、增加吞吐量、加强 网络数据的处理能力、提高网络的灵活性和可用性

负载均衡的算法：轮询法、随机法、最小连接法

 常见的软件负载均衡技术： 1、基于DNS的负载均衡

                2、反向代理负载均衡

                3、基于NAT的负载均衡技术（简称：LVS）

Nainx反向代理

## 注册登录流程

首先输入完手机号后，点击获取验证码，验证码的形式是（短信验证码），按钮开始60秒倒计时，获取到手机号发送短信验证码，把验证码存到（用户手机关联表），

验证码怎么存的：存到数据库，

获取到验证码以后，写上验证码，填写用户名和密码，把数据提交到后台，先查询（用户手机关联表）数据是否正确（判断验证码和code是否一致，判断是否过期，判断两个密码是否一致），

用户表跟验证码表怎么关联（通过手机号来关联）

密码怎么加密：（用password（HASH））

## 登录

可以用手机号登录，也可以用 用户名；也可以用第三方登录（通过手机号关联，（第三方登录怎么和手机号关联的，（我们在用户点击授权之后，弹出一个填写手机号的框，获取验证码，将用户基本信息跟手机号绑在一起）））；这就可以实现登录了

php八种数据类型有哪些？

字符串  整形 null  布尔型浮点型  数组 对象  资源

## 支付的流程

1. 拼接支付支付需要的参数

2、使用公钥进行加密

3、调用支付宝的接口，然后接受返回的接口

4、如果调用成功，页面会自动跳转到支付宝页 面

5、用户使用支付宝进行支付

6、支付宝把支付的结果通知我们的网站

## left join、right join、inner join 的区别

left join(左联接) 返回包括左表中的所有记录和右表中联结字段相等的记录

right join(右联接) 返回包括右表中的所有记录和左表中联结字段相等的记录

inner join(等值连接) 只返回两个表中联结字段相等的行

## Php有几种运行模式

运行模式有三种：cli运行模式；soapi运行模式；php-fpm；

PHP的通讯方式有两种：分别是php-fpm , socket

什么是 fast\_cgi,什么是 cgi.什么是 php-fpmcgi, php cgi,他们之间的区别

Cgi 看做是一个面向过程 php\_cgi（php 里的面向过程）

fast\_cgi 就是一个面向对象(mvc) php\_fpm (各种框架)

## PHP的超全局变量是什么

$GLOBALS 储存全局作用域中的变量

$\_SERVER 获取服务器相关信息

$\_REQUEST 获取 POST 和 GET 请求的参数

$\_POST 获取表单的 POST 请求参数

$\_GET 获取表单的 GET 请求参数

$\_FILES 获取上传文件的的变量

$\_ENV 获取服务器端环境变量的数组

$\_COOKIE 浏览器 cookie 的操作

## PHP处理时间的常用函数

（重点看函数的‘参数’和‘返回值’）

date\_default\_timezone\_get()返回默认时区。

date\_default\_timezone\_set()设置默认时区。

date()格式化本地时间／日期。

getdate()返回日期／时间信息。

gettimeofday()返回当前时间信息。

microtime() 返回当前时间的微秒数。

mktime()返回一个日期的 Unix 时间戳。

strtotime()将任何英文文本的日期或时间描述解析为 Unix 时间戳。

time()  返回当前时间的 Unix 时间戳。

## cookie和session的关系，区别

【【（1） cookie是存在客户端的浏览器上，而session存在服务器端的文件中【受phpini的影响】。

（2） 一个站点最多有20个cookie，每个cookie不能超过2kb，session是没有限制。

（3） session的安全性高于cookie，可以把重要的数据放在session中

（4） session是基于cookie的，sessionid是需要存在cookie当中的。

（5） cookie被禁用session还可以使用，可以通过url传参的方式使用。

## 禁用cookie后，session还能用吗

可以，在存储session的文件中，生成sessionID，通过get传参的方式将sessionID传到要实现session共享的页面，读取sessionID,从而从session中获取数据。

## 魔术方法

\_\_construct() 实例化对象时被调用，当\_\_construct和以类名为函数名的函数同时存在时，\_\_construct将被调用，另一个不被调用。

\_\_destruct() 当删除一个对象或对象操作终止时被调用。

\_\_call() 对象调用某个方法，若方法存在，则直接调用；若不存在，则会去调用\_\_call函数。

\_\_get() 读取一个对象的属性时，若属性存在，则直接返回属性值；若不存在，则会调用\_\_get函数。

\_\_set() 设置一个对象的属性时，若属性存在，则直接赋值；若不存在，则会调用\_\_set函数。

\_\_toString() 打印一个对象的时被调用。如echo $obj;或print $obj;

\_\_clone() 克隆对象时被调用。如：$t=new Test();$t1=clone $t;

\_\_sleep() serialize之前被调用。若对象比较大，想删减一点东东再序列化，可考虑一下此函数。

\_\_wakeup() unserialize时被调用，做些对象的初始化工作。

\_\_isset() 检测一个对象的属性是否存在时被调用。如：isset($c->name)。

\_\_unset() unset一个对象的属性时被调用。如：unset($c->name)。

\_\_set\_state() 调用var\_export时，被调用。用\_\_set\_state的返回值做为var\_export的返回值。

\_\_autoload() 实例化一个对象时，如果对应的类不存在，则该方法被调用。

## 大流量的网站解决访问量问题

(1)确认服务器硬件是否足够支持当前的流量

(2)优化数据库访问

(3)禁止外部的盗链

(4)控制大文件的下载

(5)使用不同主机分流主要流量

(6)使用流量分析统计软件

# Mysql篇

## mysql优化

四个方面:

1.设计:表结构,字段类型,存储引擎,范式满足.

2.功能:索引,查询缓存,分表,分区.

3.架构:多台MySQL服务器共同提供数据服务.读写分离,主从复制,负载均衡,高可用,冗余.

4.SQL业务逻辑:队列(缓冲),缓存(备份),重写SQL,避免性能底低下的SQL.

## mysql常见的存储引擎

Innodb和myisam

（innodb 写操作快、支持事务、数据行锁、 支持

外键 InnoDB 的主键范围更大，最大是 My 的 2 倍。

MyISAM 不支持事务和外键，读操作快、 支持数据表锁、 GIS 数据

在高并发的情况下，因为 innodb 支持行锁可以多行操作，而 myisam 只支持表锁只能一个一个操作所以慢）

1、innodb支持事务 【 start TRANSACTION ， commit , rollback 】

2、Myisam是比innodb要快的、移值性更强。

3、数据存储方式，myisam数据文件单独存储，索引文件也是单独存储的，而innodb的数据和索引是放在同一个文件里边的。

4、Myisam支持表锁，innodb支持行锁

5、Myisan支持全文索引，innodb不支持全文索引（5.6 innodb也支持全文索引）

6、Innodb支持外键，myisam不支持外键。

7、Innodb是聚簇索引，myisam非聚簇索引。

## 数据库索引

索引是对数据库表中一列或多列的值进行排序的一种结构，使用索引可快速访问数据库表中的特定信息

主键索引：数据记录里面不能有null,数据内容不能重复，在一张表里面不能有多个主键索引。 普通索引：使用字段关键字建立的索引，主要是提高查询速度

唯一索引：字段数据是唯一的，数据内容里面能否为null,在一张表里面，是可以添加多个唯一索引。

全文索引：在比较老的版本中，只有myisam引擎支持全文索引，在innodb5.6后引擎也支持全文索引，在mysql中全文索引不支持中文。我们一般使用sphinx集合coreseek来实现中文的全文索引。

### MySQL聚簇索引和非聚簇索引

聚簇索引（innodb）：

将数据存储与索引放到了一块，索引结构的叶子节点保存了行数据

（表数据文件本身就是按B+Tree组织的一个索引结构，聚簇索引就是按照每张表的主键构造一颗B+树，同时叶子节点中存放的就是整张表的行记录数据，也将聚集索引的叶子节点称为数据页）

非聚簇索引（myisam）：

将数据与索引分开存储，索引结构的叶子节点指向了数据对应的位置

### 聚簇索引和非聚簇索引的区别

聚簇索引的叶子节点存放的是主键值和数据行，支持覆盖索引；二级索引的叶子节点存放的是主键值或指向数据行的指针。

由于节子节点(数据页)只能按照一颗B+树排序，故一张表只能有一个聚簇索引。辅助索引的存在不影响聚簇索引中数据的组织，所以一张表可以有多个辅助索引

## 如何查看索引的使用情况

explain

## 数据库主从，主主复制，动静分离

主从复制：

原理

主从：

(1) 主库db的更新事件(update、insert、delete)被写到binlog

(2) 主库创建一个binlog dump thread，把binlog的内容发送到从库

(3) 从库启动并发起连接，连接到主库

(4) 从库启动之后，创建一个I/O线程，读取主库传过来的binlog内容并写入到relay log

(5) 从库启动之后，创建一个SQL线程，从relay log里面读取内容，从                                                              Exec\_Master\_Log\_Pos位置开始执行读取到的更新事件，将更新内容写入到slave的db

（

1.Master 数据库只要发生变化，立马记录到Binlog 日志文件中

2.Slave数据库启动一个I/O thread连接Master数据库，请求Master变化的二进制日志

3.Slave I/O获取到的二进制日志，保存到自己的Relaylog 日志文件中。

4.Slave 有一个 SQL thread定时检查Realylog是否变化，变化那么就更新数据

）

主从延时解决：

1、提高从库的配置

2、使用MYSQL中间件 mysql\_proxy 会自动识别延迟，如果存在延迟 读取主库的数据

3、sync\_binlog=1 每执行一次都做一次持久化

4、使用缓存，把数据存入缓存中，优先读取缓存的数据。

主从延时产生的原因：

1. 因为主库是多线程并发写入
2. 从库是单线程串行同步的

动静分离 ：分为静态页面和动态页面，静态页面的话就是我们把经常访问的页面，不需要经常读取数据库的页面，我们都给他做成静态页面，不用直接读取数据库，这样能减轻数据库压力，解决数据库缓慢

## char和varchar的区别

1、字符类型长度不同

char类型的长度是固定的，varchar的长度是可变的。

比如：使用char(10)，表示存储的字符将占10个字节（包括7个空字符）；使用varchar2(10),则表示只占3个字节，10是最大值，当存储的字符小于10时，按照实际的长度存储。

2、效率不同

char类型的效率比varchar的效率稍高。

3、空间占用不同

varchar2比char节省空间，但是在效率上比char稍差些。既要获得效率即必须牺牲一点空间，这就是设计上的"以空间换时间"。

## 事务

事务就是被绑定在一起作为一个逻辑工作单元的SQL语句分组，如果任何一个语句操作失败那么整个操作就被失败，以后操作就会回滚到操作前状态，或者是上有个节点。

## 锁

在所以的DBMS中，锁是实现事务的关键，锁可以保证事务的完整性和并发性。与现实生活中锁一样，它可以使某些数据的拥有者，在某段时间内不能使用某些数据或数据结构。当然锁还分级别的。

Mysql的锁实现了事物的隔离性，锁机制用于管理对共享资源的并发访问

## MySQL有几种锁？

使用方式：

共享锁（乐观锁）（读锁）version：其他事务可以读，但不能写。

排他锁（悲观锁）（写锁）forupdate ：其他事务不能读取，也不能写。

粒度：

表级锁、行级锁、页级锁

innodb行锁基于索引

myisam支持并发操作

## 一条SQL的生命周期是什么



# NoSql

## 什么是nosql

非关系型数据库

## Mongodb

MongoDB是一个介于关系数据库和非关系数据库之间的产品，是非关系数据库当中功能最丰富，最像关系数据库的。

它的特点是高性能、易部署、易使用，存储数据非常方便。主要功能特性有：

\*面向集合存储，易存储对象类型的数据。

\*支持动态查询。

\*支持完全索引，包含内部对象。

\*支持查询。

\*支持复制和故障恢复。

\*使用高效的二进制数据存储，包括大型对象（如视频等）。

\*自动处理碎片，以支持云计算层次的扩展性

\*文件存储格式为BSON（一种JSON的扩展）。

不适用的场景如下：

1）要求高度事务性的系统。

2）传统的商业智能应用。

3）复杂的跨文档（表）级联查询。

## Redis与memcache

 redis和memcache的区别

1:从数据存储 位置上来分，

 memcache的数据存在内存中，

redis既可以存储在内存中，也可以存储到磁盘中，达到持久化存储持久化的功能，

   memcache一旦断电，数据全部丢失，

   redis可以利用快照(ROB)和AOF把数据存到磁盘中，当恢复时又从磁盘中读取到内存中，当物理内存使用完毕后，可以把数据写入到磁盘中。

2：从存储类型上来分，

 memcache 和redis存储方式都是键值，

只不过redis值的类型比较丰富，有string(字符串),hash(列表),set(集合)，zset(有序集 合),

而 memcache主要存储的是字符串

3：从架构上来分，

 redis支持master-slave(主---从)模式应用，memcache支持分布式。

4：从存储数据的大小上来分：

 redis单个value的最大限制是1GB，memcache只能保存1MB的数据，

但是Memcache在存储100K以上的数据，性能稍微好一点。

 5：另外redis只支持单核，memcache可以支持多核

## redis进行持久化的两种方式

AOF(日志)，RDB(快照)

一种是RDB持久化（原理是将Reids在内存中的数据库记录定时dump到磁盘上的RDB持久化）

另外一种是AOF（append only file）持久化（原理是将Reids的操作日志以追加的方式写入文件）。

## redis死锁

死锁:一个客户端获取锁成功，但是在释放锁之前崩溃了，此时该客户端实际上已经失去了对公共资源的操作权，但却没有办法请求解锁（删除 Key-Value键值对），那么，它就会一直持有这个锁，而其它客户端永远无法获得锁。

解决：在执行setnx时，把value值利用起来，设置为当前的时间，在每次获取锁失败后都判断一次当前时间和redis的缓存值时间，是否超过设置的超时时间，超过则删除此key，重新获取锁。

## 对于大流量的网站,您采用什么样的方法来解决访问量问题?

(1)确认服务器硬件是否足够支持当前的流量

(2)优化数据库访问

(3)禁止外部的盗链

(4)控制大文件的下载

(5)使用不同主机分流主要流量

(6)使用流量分析统计软件

## 简述面向对象的优势

使程序局部化，易修改、好维护，安全、可靠、可重用。高效率高质量

## mysql事务及其特性

是指作为单个逻辑工作单元执行的一系列操作，要么完全地执行，要么完全地不执行。执行多个sql语句的时候，要么都成功要么都失败。四个特性：原子性： sql语句要么都成功，要么都不成功。把事务当作一个原子性的操作。一致性： 数据库在执行事务之前是一个一致性状态，执行事务之后还是一致性状态，不会因为执行事务导致不一致。持久性： 事务执行成功之后，对数据库的造成的影响是永久的，不可逆。隔离性： 多个事务互不影响。索引是什么 优点和缺点

索引:索引相当于是数据的目录（类似于字典的目录或者是书的目录），主要用来加快数据的查询速度。索引不是越多越好。优点：可以加快查询速度缺点:1、会降低插入速度。

索引也需要单独存储

## 简述单点登录的原理.(重复)

 单点登录的技术实现机制：当用户第一次访问应用系统1的时候，因为还没有登录，会被引导到认证系统中进行登录;根据用户提供的登录信息，认证系统进行身份 效验，如果通过效验，应该返回给用户一个认证的凭据--ticket;用户再访问别的应用的时候，就会将这个ticket带上，作为自己认证的凭据，应用 系统接受到请求之后会把ticket送到认证系统进行效验，检查ticket的合法性。如果通过效验，用户就可以在不用再次登录的情况下访问应用系统2和 应用系统3了

## Redis支持哪几种数据类型？

支持多种类型的数据结构1.string：最基本的数据类型，二进制安全的字符串，最大512M。2.list：按照添加顺序保持顺序的字符串列表。  3.set：无序的字符串集合，不存在重复的元素。4.sorted set：已排序的字符串集合。5.hash：key-value对的一种集合。

## Redis是单进程单线程的？

Redis是单进程单线程的，Redis利用队列技术将并发访问变为串行访问，消除了传统数据库串行控制的开销。Redis为什么是单线程的？多线程处理会涉及到锁，而且多线程处理会涉及到线程切换而消耗CPU。因为CPU不是Redis的瓶颈，Redis的瓶颈最有可能是机器内存或者网络带宽。单线程无法发挥多核CPU性能，不过可以通过在单机开多个Redis实例来解决。

## 使用Redis的优势？

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
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

## Redis如何做内存优化？

尽可能使用散列表（hashes），散列表（是说散列表里面存储的数少）使用的内存非常小，所以应该尽可能的将数据模型抽象到一个散列表里面。比如web系统中有一个用户对象，不要为这个用户的名称，姓氏，邮箱，密码设置单独的key,而是应该把这个用户的所有信息存储到一张散列表里面.

## Redis有哪些适合的场景？

1）Session共享(单点登录)2）页面缓存3）队列4）排行榜/计数器5）发布/订阅

## 什么是Redis？简述它的优缺点？

Redis本质上是一个Key-Value类型的内存数据库，很像memcached，整个数据库统统加载在内存当中进行操作，定期通过异步操作把数据库数据flush到硬盘上进行保存。因为是纯内存操作，Redis的性能非常出色，每秒可以处理超过 10万次读写操作，是已知性能最快的Key-Value DB。Redis的出色之处不仅仅是性能，Redis最大的魅力是支持保存多种数据结构，此外单个value的最大限制是1GB，不像 memcached只能保存1MB的数据，因此Redis可以用来实现很多有用的功能。比方说用他的List来做FIFO双向链表，实现一个轻量级的高性 能消息队列服务，用他的Set可以做高性能的tag系统等等。另外Redis也可以对存入的Key-Value设置expire时间，因此也可以被当作一 个功能加强版的memcached来用。 Redis的主要缺点是数据库容量受到物理内存的限制，不能用作海量数据的高性能读写，因此Redis适合的场景主要局限在较小数据量的高性能操作和运算上。

## 为什么Redis需要把所有数据放到内存中？

Redis为了达到最快的读写速度将数据都读到内存中，并通过异步的方式将数据写入磁盘。所以redis具有快速和数据持久化的特征。如果不将数据放在内存中，磁盘I/O速度为严重影响redis的性能。在内存越来越便宜的今天，redis将会越来越受欢迎。 如果设置了最大使用的内存，则数据已有记录数达到内存限值后不能继续插入新值。

## Redis为什么使用单线程

  官方解释如下：因为Redis是基于内存的操作，CPU不是Redis的瓶颈，Redis的瓶颈最有可能是机器内存的大小或者网络带宽。既然单线程容易实现，而且CPU不会成为瓶颈，那就顺理成章地采用单线程的方案了。RDB机制RDB其实就是把数据以快照的形式保存在磁盘上。什么是快照呢，你可以理解成把当前时刻的数据拍成一张照片保存下来。RDB持久化是指在指定的时间间隔内将内存中的数据集快照写入磁盘。也是默认的持久化方式，这种方式是就是将内存中数据以快照的方式写入到二进制文件中,默认的文件名为dump.rdb。

# VUE

## vue全家桶

vue：主要Vue

​ vue-router：关于路由方面的配置

​ vuex：数据共享和缓存用

axios：于后台交互用（之前名为为​ vue-resource 但是axios不是Vue里面的东西）

​ vue-cli：快速创建项目的脚手架

## vue的两个核心点

答：数据驱动、组件系统

数据驱动：ViewModel，保证数据和视图的一致性。

组件系统：应用类UI可以看作全部是由组件树构成的。

## 引进组件的步骤

答: 在template中引入组件；

在script的第一行用import引入路径；

用component中写上组件名称。

## 什么是 vue 生命周期？有什么作用？

答：每个 Vue 实例在被创建时都要经过一系列的初始化过程——例如，需要设置数据监听、编译模板、将实例挂载到 DOM 并在数据变化时更新 DOM 等。同时在这个过程中也会运行一些叫做 生命周期钩子 的函数，这给了用户在不同阶段添加自己的代码的机会。（ps：生命周期钩子就是生命周期函数）例如，如果要通过某些插件操作DOM节点，如想在页面渲染完后弹出广告窗， 那我们最早可在mounted 中进行。

# 小程序

## 微信小程序的优劣势

优势：

1、无需下载，通过搜索和扫一扫就可以打开。

2、良好的用户体验：打开速度快。

3、开发成本要比App要低。

4、安卓上可以添加到桌面，与原生App差不多。

5、为用户提供良好的安全保障。小程序的发布，微信拥有一套严格的审查流程， 不能通过审查的小程序是无法发布到线上的。

劣势：

1、限制较多。页面大小不能超过1M。不能打开超过5个层级的页面。

2、样式单一。小程序的部分组件已经是成型的了，样式不可以修改。例如：幻灯片、导航。

3、推广面窄，不能分享朋友圈，只能通过分享给朋友，附近小程序推广。其中附近小程序也受到微信的限制。

4、依托于微信，无法开发后台管理功能。

## 怎么封装微信小程序的数据请求的？

一、将所有的接口放在统一的js文件中并导出

二、在app.js中创建封装请求数据的方法

三、在子页面中调用封装的方法请求数据

## 提高微信小程序的应用速度？

一、提高页面加载速度

二、用户行为预测

三、减少默认data的大小

四、组件化方案

## 微信小程序与H5的区别？

第一条是运行环境的不同

传统的HTML5的运行环境是浏览器，包括webview，而微信小程序的运行环境并非完整的浏览器，是微信开发团队基于浏览器内核完全重构的一个内置解析器，针对小程序专门做了优化，配合自己定义的开发语言标准，提升了小程序的性能。

第二条是开发成本的不同

只在微信中运行，所以不用再去顾虑浏览器兼容性，不用担心生产环境中出现不可预料的奇妙BUG

第三条是获取系统级权限的不同

系统级权限都可以和微信小程序无缝衔接

第四条便是应用在生产环境的运行流畅度

长久以来，当HTML5应用面对复杂的业务逻辑或者丰富的页面交互时，它的体验总是不尽人意，需要不断的对项目优化来提升用户体验。但是由于微信小程序运行环境独立

## 小程序关联微信公众号如何确定用户的唯一性？

答案：使用wx.getUserInfo方法withCredentials为 true 时 可获取encryptedData，里面有 union\_id。后端需要进行对称解密

## 小程序调用后台接口遇到哪些问题？

一、数据的大小有限制，超过范围会直接导致整个小程序崩溃，除非重启小程序；

二、小程序不可以直接渲染文章内容页这类型的html文本内容，若需显示要借住插件，但插件渲染会导致页面加载变慢，所以最好在后台对文章内容的html进行过滤，后台直接处理批量替换p标签div标签为view标签，然后其它的标签让插件来做，减轻前端的时间。

# 微服务

微服务架构属于分布式系统

## 什么是微服务

微服务架构是一种架构模式或者说是一种架构风格，它提倡将单一应用程序划分成一组小的服务，每个服务运行在其独立的自己的进程中，服务之间互相协调、互相配合，为用户提供最终价值.

## 微服务的优点

微服务可以轻松适应其他框架或技术。单个进程的失败不会影响整个系统。为大企业和小型团队提供支持。可以在相对较短的时间内独立部署。

# 扩展

## oss

阿里云对象存储