一、PHP基础

1、用PHP打印出前一天的时间格式是2006-5-10 22:21:21
2、php权限控制修饰符有哪些?
3、SESSION 与COOKIE 的区别是什么，请从协议，产生的原因与作用说明?
4、session依赖于cookie，cookie存储着sessionid。禁用cookie，session是否可以使用。
5、说明 PHP 中传值和传引用的区别?
6、说出php 常用的字符串函数。
7、php中魔术常量和魔术方法有哪些，做简要说明
魔术方法:
魔术常量:
8、isset、empty、is_null的区别
9、什么是面向对象/OOP思想？主要特征是什么?
10、说出php 常用的数组函数。
11、写出下列几个预定义全局变量的作用?
12、include 和 require 都能把另外一个文件包含到当前文件中，他们有什么区别?
13、表单中get与post提交方法的区别?
14、php异常级别，如何处理异常?
15、Echo ,print(),print_r()的区别是什么?
16、用 PHP 写出显示客户端 IP 与服务器 IP 的代码
17、如何实现多个线程安全的写入一个文件数据
18、请谈谈您对 MVC 的理解?
19、如何快速下载一个远程 http 服务器上的图片文件到本地?
20、谈谈你对设计模式的看法，详细阐述工厂模式和单例模式。
设计模式的分类
单利模式:
工厂模式:
21、PHP 字符串中单引号与双引号的区别?
22、++i 和 i++哪一个效率高，为什么?
23、防盗链的原理和实现方式?
24、简单解释CGI,fastCGI,PHP-FPM
25、fastcgi通过端口监听和通过文件监听的区别?
26、PHP的垃圾收集机制是怎样的
27、了解 XSS 攻击吗? 如何防止 ?
28、PHP7新特性有哪些
29、接口和抽象类的区别是什么?

二、HTTP/HTTPS协议

1、请问http状态码以1、2、3、4、5开头的分别代表什么意思?
详细:
2、用户输入一个网址（www.baidu.com）,到用户看到对应的网页加载完毕，此过程都发生了什么?
3、解释TCP的三次握手。
4、描述一下 HTTP 与 HTTPS 的区别?
5、什么是Http协议无状态协议?怎么解决Http协议无状态协议?
6、常用的HTTP方法有哪些?
7、URI和URL的区别
8、HTTP请求报文与响应报文格式

三、ThinkPHP5\Laravel5

1、ThinkPHP5框架的优点
2、ThinkPHP5的生命周期是如何执行的
3、ThinkPHP5的URL访问格式是怎样的?
4、ThinkPHP5路由模式有几种?
一、普通模式

12、active-class 是哪个组件的属性?嵌套路由怎么定义?
13、怎么定义 vue-router 的动态路由?怎么获取传过来的动态参数?
14、vue-router 有哪几种导航钩子?
15、scss 是什么？在 vue.cli 中的安装使用步骤是？有哪几大特性?
16、mint-ui 是什么？怎么使用？说出至少三个组件使用方法?
17、v-model 是什么?怎么使用? vue 中标签怎么绑定事件?
18、axios 是什么?怎么使用?描述使用它实现登录功能的流程?
19、axios+tp5 进阶中，调用 axios.post('api/user')是进行的什么操作? axios.put('api/user/8')呢?
20、vuex 是什么？怎么使用？哪种功能场景使用它?
21、mvvm 框架是什么？它和其它框架（jquery）的区别是什么？哪些场景适合?
22、自定义指令(v-check、v-focus)的方法有哪些?它有哪些钩子函数?还有哪些钩子 函数参数?
23、vue-router 是什么?它有哪些组件?
24、Vue 的双向数据绑定原理是什么?
25、你是怎么认识 vuex 的?
26、请说下封装 vue 组件的过程?
27、vue-loader 是什么?使用它的用途有哪些?
28、请说出 vue.cli 项目中 src 目录每个文件夹和文件的用法?

八、微信公众号/小程序

1、简单描述下微信小程序的相关文件类型?
2、你使用过哪些方法，来提高微信小程序的应用速度?

九、Linux

1、绝对路径用什么符号表示？当前目录、上层目录用什么表示？主目录用什么表示? 切换目录用什么命令?
2、怎么查看当前进程？怎么执行退出？怎么查看当前路径?
3、怎么清屏？怎么退出当前命令？怎么执行睡眠？怎么查看当前用户 id？查看指定帮助用什么命令?
4、Ls 命令执行什么功能？ 可以带哪些参数，有什么区别?
5、建立软链接(快捷方式)，以及硬链接的命令。
6、目录创建用什么命令？创建文件用什么命令？复制文件用什么命令?
7. 文件权限修改用什么命令？格式是怎么样的?
8、查看文件内容有哪些命令可以使用?
9、随意写文件命令？怎么向屏幕输出带空格的字符串，比如"hello world"?
10、终端是哪个文件夹下的哪个文件？黑洞文件是哪个文件夹下的哪个命令?
11、移动文件用哪个命令？改名用哪个命令?
12、复制文件用哪个命令？如果需要连同文件夹一块复制呢？如果需要有提示功能呢?
13、删除文件用哪个命令？如果需要连目录及目录下文件一块删除呢？删除空文件夹用什么命令?
14、Linux 下命令有哪几种可使用的通配符？分别代表什么含义?
15、用什么命令对一个文件的内容进行统计？(行号、单词数、字节数)
16、Grep 命令有什么用？ 如何忽略大小写？ 如何查找不含该串的行?
17、Linux 中进程有哪几种状态？在 ps 显示出来的信息中，分别用什么符号表示的?
18、怎么使一个命令在后台运行?
19、利用 ps 怎么显示所有的进程? 怎么利用 ps 查看指定进程的信息?
20、哪个命令专门用来查看后台任务?
21、把后台任务调到前台执行使用什么命令?把停下的后台任务在后台执行起来用什么命令?
22、终止进程用什么命令? 带什么参数?
23、怎么查看系统支持的所有信号?
24、搜索文件用什么命令? 格式是怎么样的?
25、查看当前谁在使用该主机用什么命令? 查找自己所在的终端信息用什么命令?
26、使用什么命令查看用过的命令列表?
27、使用什么命令查看磁盘使用空间？ 空闲空间呢?
28、使用什么命令查看网络是否连通?
29、使用什么命令查看 ip 地址及接口信息?
30、查看各类环境变量用什么命令?
31、通过什么命令指定命令提示符?
32、查找命令的可执行文件是去哪查找的? 怎么对其进行设置及添加?

33、通过什么命令查找执行命令?
34、怎么对命令进行取别名?
35、du 和 df 的定义，以及区别?
36、awk 详解。
37、当你需要给命令绑定一个宏或者按键的时候，应该怎么做呢?
38、如果一个linux新手想要知道当前系统支持的所有命令的列表，他需要怎么做?
39、如果你的助手想要打印出当前的目录栈，你会建议他怎么做?
40、你的系统目前有许多正在运行的任务，在不重启机器的条件下，有什么方法可以把所有正在运行的进程移除呢?
41、bash shell 中的hash 命令有什么作用?
42、哪一个bash内置命令能够进行数学运算。
43、怎样一页一页地查看一个大文件的内容呢?
44、数据字典属于哪一个用户的?
45、怎样查看一个linux命令的概要与用法？假设你在/bin目录中偶然看到一个你从没见过的的命令，怎样才能知道它的作用和用法呢?
46、使用哪一个命令可以查看自己文件系统的磁盘空间配额呢?

十、MYSQL

2、数据库事务的四个特性及含义
3、drop,delete与truncate的区别
4、索引的工作原理
5、索引的优点
6、索引的缺点
7、什么样场合下不建议创建索引?
8、数据库范式
9、MySQL中myisam与innodb的区别
10、MySQL中varchar与char的区别以及varchar(50)中的50代表的涵义
11、表中有大字段X(例如：text类型)，且字段X不会经常更新，以读为为主，请问您是选择拆成子表，还是继续放一起?写出您这样选择的理由
12、数据库优化的思路
13、什么情况下设置了索引但无法使用
14、数据库中的事务是什么?
15、22.SQL注入漏洞产生的原因？如何防止?
防止SQL注入的方式：
16、 说说对SQL语句优化有哪些方法？（选择几条）
17、 char和varchar的区别?
18、MySQL数据库作发布系统的存储，一天五万条以上的增量，怎么优化?
19. 对于大流量的网站,您采用什么样的方法来解决各页面访问量统计问题?
20、什么是队列？排它锁，Myisam 死锁如何解决?

十一、NOSQL数据库

1、Redis、Memcache与MongoDB的区别
2.mongodb持久化原理
3、什么是NoSQL数据库？NoSQL和RDBMS有什么区别？在哪些情况下使用和不使用NoSQL数据库?
4、MongoDB的特点是什么?
5、什么是缓存穿透?
6、如何避免缓存穿透?
7、什么是缓存雪崩?
8、如何避免缓存雪崩?
9、缓存数据的淘汰
10、memcache 缓存什么数据
11、Redis 如何防止高并发
12、redis 消息队列先进先出需要注意什么

十二、排序算法

1. 冒泡排序

一、PHP基础

1、用PHP打印出前一天的时间格式是2006-5-10 22:21:21

方式一：echo date('Y-m-d H:i:s',strtotime('-1 day',time()));

方式二：echo date("Y-m-d H:i:s",time()-24*3600);

2、php权限控制修饰符有哪些?

public: 对外公开，可以在任何地方访问;

protected:当前包下及子类可以访问;

private：只有当前类可以访问

3、SESSION 与COOKIE 的区别是什么，请从协议，产生的原因与作用说明?

HTTP协议是无状态的协议。一旦数据交换完毕，客户端与服务器端的连接就会关闭，再次交换数据需要建立新的连接。这就意味着服务器无法从连接上跟踪会话。于是需要引入一种机制，COOKIE于是就顺应而生。

Session是另一种记录客户状态的机制，不同的是Cookie保存在客户端浏览器中，而Session保存在服务器上。客户端浏览器访问服务器的时候，服务器把客户端信息以某种形式记录在服务器上。这就是Session。

区别：（位置，大小，安全）

1、cookie 是存放在浏览器端，不同的浏览器存储的cookie数量和数据的大小都不一致。大多数情况下单个域名限制最多保存20个cookie，每个cookie保存的数据不能超过4K。

2、session存储在服务端，默认是以文件的形式存储，也可以存储在数据库和redis、memcache等缓存内存中。

3、session是占用的服务器内存，所以内存越大，能存的值就越大，原则上讲无上限，一般用于存储对安全要求较高的重要数据；

4、session依赖于cookie，cookie存储着sessionid。禁用cookie，session是否可以使用。

可以通过其他方式传递sessionid。具体操作如下：

- 设置php.ini中的session.use_trans_sid = 1或者编译时打开打开了–enable-trans-sid选项，
- 让PHP自动跨页传递session id。
- 手动通过URL传值、隐藏表单传递session id。
- 用文件、数据库等形式保存session_id,在跨页过程中手动调用

5、说明 PHP 中传值和传引用的区别?

传值：

是把实参的值赋值给形参，那么对形参的修改，不会影响实参的值

传引用：

真正的以地址的方式传递参数

传递以后，形参和实参都是同一个对象，只是他们名字不同而已

对形参的修改将影响实参的值

6、说出php 常用的字符串函数。

- stripos — 查找字符串首次出现的位置（不区分大小写）
- stripslashes — 反引用一个引用字符串
- stristr — strstr 函数的忽略大小写版本
- strlen — 获取字符串长度
- strnatcasecmp — 使用“自然顺序”算法比较字符串（不区分大小写）
- strnatcmp — 使用自然排序算法比较字符串
- strncasecmp — 二进制安全比较字符串开头的若干个字符（不区分大小写）
- strncmp — 二进制安全比较字符串开头的若干个字符

7、php中魔术常量和魔术方法有哪些，做简要说明

魔术方法：

1. __construct() 类的默认构造方法，如果construct()和与类同名的方法共同出现时，默认调用construct()而不是同类名方法。一般情况下用户自定义构造方法也会使用construct()。
2. __destruct() 类的析构函数，当该对象的所有引用都被删除，或者对象被显式销毁时执行。
3. **get(**$name$**)**可以简单归纳为：用**object->a的方式读取对象的属性时，如果属性a存在且是public型，那么直接返回该属性的值；如果属性a不存在或者是protected/private这样的不可直接访问的类型，就会调用**get($name)方法并以返回值为准。一般可以使用该方法使外部可限制性地访问内部属性，或者完成类似java中的反射操作。
4. **set(**$name$**,value) 与**get($name$)类似，用object->a = 17的方式给属性赋值时，如果属性a存在且是public型，那么直接给属性a赋值皆可；如果属性a不存在或者是protected/private型，就会调用**set(**$name$**,value)方法。**
5. __call($name$,arguments) / callStatic($name$,arguments) 当调用不存在或者不可访问的方法时，会调用call($name$,arguments)方法。而当在静态方法中调用不存在或者不可访问的方法时，会调用callStatic($name$, arguments)方法。
6. **toString() 当打印对象时会被直接调用。如echo $object;**
7. __clone() 当对象被拷贝时直接调用。如$a = newAction()$;a = $object;
8. __isset($name$)/$unset$(name) 对不存在或者不可访问的属性使用isset()或者empty()时，isset()会被调用；当unset一个不存在或者不可访问的属性时，unset()会被调用，否则直接unset该属性皆可。

魔术常量:

1. LINE 返回文件中的当前行号。 **2. FILE** 返回所在文件的完整路径。包含文件名 **3. FUNCTION** 返回所在函数名称。 **4. CLASS** 返回所在类的名称。 **5. METHOD** 返回所在类方法的名称。需要注意**METHOD**返回的是"class::function"的形式，而**FUNCTION**则返回"function"的形式。

6.DIR

返回文件所在的目录。如果用在被包括文件中，则返回被包括的文件所在的目录。它等价于 *dirname(**FILE**)*。除非是根目录，否则目录中名不包括末尾的斜杠。不包含文件名。（PHP 5.3.0中新增） =

8、isset、empty、is_null的区别

- empty（）

bool empty (mixed var)

如果 var 是非空或非零的值，则 empty() 返回 FALSE。换句话说，""、0、"0"、NULL、FALSE、array()、var $var; 以及没有任何属性的对象都将被认为是空的，如果 var 为空，则返回 TRUE

- isset()

如果 var 存在则返回 TRUE，否则返回 FALSE。

如果已经使用 unset() 释放了一个变量之后，它将不再是 isset()。若使用 isset() 测试一个被设置成 NULL 的变量，将返回 FALSE。同时要注意的是一个 NULL 字节（"0"）并不等同于 PHP 的 NULL 常数。

- is_null()：

参数满足下面三种情况时，is_null()将返回TRUE，其它的情况就是FALSE

1、它被赋值为NULL

2、它还没有赋值

3、它未定义，相当于unset(),将一个变量unset()后，不就是没有定义吗

9、什么是面向对象/OOP思想？主要特征是什么？

面向对象是程序的一种设计方式，它利于提高程序的重用性，使程序结构更加清晰。

主要特征：封装、继承、多态

封装：将方法，属性等封装到一个类中，通过声明访问权限来控制访问。

继承：通过继承父类，子类可以访问到父类的公共的和受保护的方法和属性。PHP只支持单继承，一个子类只支持继承一个父类。但是可以通过链式继承来实现多继承的效果

多态：子类继承了来自父级类中的属性和方法，并对其中部分方法进行重写。于是多个子类中虽然都具有同一个方法，但是这些子类实例化的对象调用这些相同的方法后却可以获得完全不同的结果

10、说出php 常用的数组函数。

array_keys()	**返回数组中所有的键名。**
array_map()	把数组中的每个值发送到用户自定义函数，返回新的值。
array_merge()	把一个或多个数组合并为一个数组。
array_pop()	删除数组的最后一个元素（出栈）。
array_push()	将一个或多个元素插入数组的末尾（入栈）。
array_rand()	返回数组中一个或多个随机的键。
in_array()	检查数组中是否存在指定的值。
key()	从关联数组中取得键名。
krsort()	对数组按照键名逆向排序。
ksort()	对数组按照键名排序。

11、写出下列几个预定义全局变量的作用?

$_SERVER['DOCUMENT_ROOT'] //当前运行脚本所在的文档根目录 $_SERVER['HTTP_HOST '//当前请求的 Host: 头部的内容 $_SERVER['REMOTE_ADDR']//正在浏览当前页面用户的 IP 地址 $_SERVER['HTTP_REFERER'] //链接到当前页面的前一页面的 URL 地址

$_SERVER['SERVER_NAME'] //当前运行脚本所在服务器主机的名称 $_FILES//包含有所有上传的文件信息 S_FILES['userfile']['name']//客户端机器文件的原名称 $_FILES['userfile']['type']//文件 MIME 类型，如果浏览器提供此信息的话，如“image/gif”。

12、include 和 require 都能把另外一个文件包含到当前文件中，他们有什么区别?

- 1、require 的使用方法如 require("MyRequireFile.php"); 。这个函数通常放在 PHP 程序的最前面，PHP 程序在执行前，就会先读入 require 所指定引入的文件，使它变成 PHP 程序网页的一部份。
- 2、include 使用方法如 include("MyIncludeFile.php"); 。这个函数一般是放在流程控制的处理部分中。PHP 程序网页在读到 include 的文件时，才将它读进来。这种方式，可以把程序执行时的流程简单化。
- 3、require一个文件存在错误的话，那么程序就会中断执行了，并显示致命错误
- 4、include一个文件存在错误的话，那么程序不会中端，而是继续执行，并显示一个警告错误。
- 5、include有返回值，而require没有
- include_once() 和required_once 区别于include和require 在于会先检查要导入的档案是不是已经在该程序中的其它地方被导入过了，如果有的话就不会再次重复导入

13、表单中get与post提交方法的区别?

- 1.GET请求的数据会附在URL之后（就是把数据放置在HTTP协议头中）。POST把提交的数据则放置在是HTTP包的包体中。
- 2、GET方式提交的数据受到特定的浏览器及服务器对它的限制，传输的数据量会比较小（2KB）；
- POST相对会较大，主要取决于服务器的处理程序的能力(IIS4中最大量为80KB，IIS5中为100KB)。
- 3、POST的安全性要比GET的安全性高

14、php异常级别，如何处理异常?

- Deprecated 最低级别的错误(不推荐，不建议)

1. 使用一些过期函数的时候会出现，程序继续执行

- Notice 通知级别的错误

1. 使用一些未定义变量、常量或者数组key没有加引号的时候会出现，程序继续执行

- Waning 警告级别的错误

1. 程序出问题了，需要修改代码！！！ 程序继续执行

- Fatal Error 错误级别的错误

1. 程序直接报错，需要修改代码！！！ 中断程序执行，可使用register_shutdown_function()函数在程序终止前触发一个函数

- Parse error 语法解析错误

1. 语法检查阶段报错，需要修改代码！！！ 中断程序执行，除了修改ini文件，将错误信息写到日志中，什么也做不了

- E_USER_相关的错误
- 用户定义的错误，用户手动抛出错误，进行自定义错误处理

PHP异常与错误的抛出

1. 异常抛出：throw new Exception('Some Error Message');
2. 错误抛出：trigger_error()
3. trigger_error() 触发的错误不会被 try-catch 异常捕获语句捕获

15、Echo ,print(),print_r()的区别是什么?

- print（）　只能打印出简单类型变量的值(如int,string)
- print_r（）可以打印出复杂类型变量的值(如数组,对象)
- echo　输出一个或者多个字符串

16、用 PHP 写出显示客户端 IP 与服务器 IP 的代码

客户端 IP：$_SERVER["REMOTE_ADDR"]服务器端IP：_SERVER["SERVER_ADDR

17、如何实现多个线程安全的写入一个文件数据

1、采用锁机制，当一个用户在对此文件进行读写入操作时，将此文件锁定，操作完毕后解除锁定， 在该用户进行读写入操作过程中，其他用户不能操作此文件，需要排队等待。

2、PHP本身不支持多线程，但是可以通过扩展pthreads来完成

18、请谈谈您对 MVC 的理解?

由模型(model),视图(view),控制器(controller) 完成的应用程序 由模型发出要实现的功能到控制器,控制器接收组织功能传递给视图; MVC 是一个设计模式，它强制性的使应用程序的输入、处理和输出分开。 使用 MVC 应用程序被分成三个核心部件：模型、视图、控制器。它们各自处理自己的任务。 视图是用户看到并与之交互的界面，模型表示企业数据和业务规则， 控制器接受用户的输入并调用模型和视图去完成用户的需求。

MVC 的优点： 低耦合性、高重用性和可适用性、较低的生命周期成本、 快速的部署、可维护性、可扩展性，有利于软件工程化管理 MVC 的缺点： 没有明确的定义，完全理解 MVC 并不容易。不适合小型规模的应用程序。

19、如何快速下载一个远程 http 服务器上的图片文件到本地?

- curl
- fopen()
- file_get_content()

  curl多用于互联网网页之间的抓取，fopen多用于读取文件，而file_get_contents多用于获取静态页面的内容。

  1. fopen /file_get_contents 每次请求都会重新做DNS查询，并不对DNS信息进行缓存。但是CURL会自动对DNS信息进行缓存。对同一域名下的网页或者图片的请求只需要一次DNS查询。这大大减少了DNS查询的次数。所以CURL的性能比fopen /file_get_contents 好很多。
  2. fopen /file_get_contents在请求HTTP时，使用的是http_fopen_wrapper，不会keeplive。而curl却可以。这样在多次请求多个链接时，curl效率会好一些。
  3. curl可以模拟多种请求，例如：POST数据，表单提交等，用户可以按照自己的需求来定制请求。而fopen / file_get_contents只能使用get方式获取数据。

20、谈谈你对设计模式的看法，详细阐述工厂模式和单例模式。

设计模式的分类

总体来说设计模式分为三大类：

创建型模式，共五种：工厂方法模式、抽象工厂模式、单例模式、建造者模式、原型模式。

结构型模式，共七种：适配器模式、装饰器模式、代理模式、外观模式、桥接模式、组合模式、享元模式。

行为型模式，共十一种：策略模式、模板方法模式、观察者模式、迭代子模式、责任链模式、命令模式、备忘录模式、状态模式、访问者模式、中介者模式、解释器模式

单利模式：

实例化出来的对象是唯一的。

所有的单例模式至少拥有以下三种公共元素：

\1. 它们必须拥有一个构造函数，并且必须被标记为private

\2. 它们拥有一个保存类的实例的静态成员变量

\3. 它们拥有一个访问这个实例的公共的静态方法

单例类不能再其它类中直接实例化，只能被其自身实例化。它不会创建实例副本，而是会向单例类内部存储的实例返回一个引用。（比如，单例模式只让一个对象去访问数据库，从而防止打开多个数据库连接。）

```
class Single {
    /* 单例模式只允许创建类的对象 */
    private function __construct()
    {
    }
    static $instance = null;
    static function GetObject($className){
        if (!isset(self::$instance)) /* 如果生产的实例未空 */
        {
            $obj = new $className();
            self::$instance = $obj;
            return self::$instance;
        }else{
            return self::$instance;
        }
    }
}
 $singleA = Single::GetObject("A");
 $singleB = Single::GetObject("A");
 var_dump($singleA);
 var_dump($singleB);
```

工厂模式：

工厂模式就是一种类，具有为您创建对象的某些方法，这样就可以使用工厂类创建对象，而不直接使用new。这样如果想更改创建的对象类型，只需更改该工厂即可。

```
class A{};
class B{};
class WorkStation {
    /* 工厂类 */
    static function GetObject($className){
        $obj = new $className();
        return $obj;
    }
}
$o1 = WorkStation::GetObject("A");
$o2 = WorkStation::GetObject("B");
var_dump($o1);
var_dump($o2);
```

21、PHP 字符串中单引号与双引号的区别?

单引号不能解释变量，而双引号可以解释变量。 单引号不能转义字符，在双引号中可以转义字符。

22、++i 和 i++哪一个效率高，为什么?

++i 效率比 i++的效率更高，因为++i 少了一个返回 i 的 过程

23、防盗链的原理和实现方式?

原理：通过判断request请求头的refer是否来源于本站。

实现方式：

1）Apache 防盗链的第一种实现方法，可以用 Rewrite 实现。首先要确认 Apache 的 rewrite module 可用：能够控制 Apache httpd.conf 文件的，打开 httpd.conf，确保有这么一行配置：

LoadModule rewrite_module modules/mod_rewrite.so

然后在相应虚拟主机配置的地方，加入下列代码：

ServerName www.itcast.com

防盗链配置 参数

RewriteEngine On

RewriteCond %{HTTP_REFERER} !^http://itcast.com/.*$ [NC]

RewriteCond %{HTTP_REFERER} !^http://itcast.com$ [NC]

RewriteCond %{HTTP_REFERER} !^http://www.itcast.com/.*$ [NC]

RewriteCond %{HTTP_REFERER} !^http://www.itcast.com$ [NC]

RewriteRule .*.(gif|jpg|swf)$ http:///www.itcast.com/img/nolink.gif[R,NC]

itcast.com/www.itcast.com 表示自己的信任站点。gif|jpg|swf 表示要保护文件的扩展名(以|分开)。nolink.gif 盗链后的重定向页面/图片。用以输出警示信息，这张图片应该尽可能的小。

24、简单解释CGI,fastCGI,PHP-FPM

- CGI：CGI的英文是（COMMON GATEWAY INTERFACE）公共网关接口，它的作用就是帮助服务器与语言通信。是 Web Server 与 Web Application 之间数据交换的一种协议。
- FastCGI：同 CGI，是一种通信协议，但比 CGI 在效率上做了一些优化，fast-cgi则是一个进程可以处理多个请求。
- PHP-CGI：是 PHP （Web Application）对 Web Server 提供的 CGI协议的接口程序。
- PHP-FPM：是 PHP（Web Application）对 Web Server 提供的 FastCGI 协议的接口程序，额外还提供了相对智能一些任务管理。

25、fastcgi通过端口监听和通过文件监听的区别?

监听方式	形式	nginx链接fastcgi方式
端口监听	fastcgi_pass 127.0.0.1:9000	TCP链接
文件监听	fastcgi_pass /tmp/php_cgi.sock	Unix domain Socket

26、PHP的垃圾收集机制是怎样的

在PHP5.3版本之前，PHP只有简单的基于引用计数的垃圾回收，当一个变量的 引用计数refcount变为0时， PHP将在内存中销毁这个变量，只是这里的垃圾并不能称之为垃圾。 并且PHP在一个生命周期结束后就会释放此进程/线程所占的内容，这种方式决定了PHP在前期不需要过多考虑内存的泄露问题。 但是随着PHP的发展，PHP开发者的增加以及其所承载的业务范围的扩大，在PHP5.3中引入了更加完善的垃圾回收机制。 新的垃圾回收机制解决了无法处理循环的引用内存泄漏问题。

27、了解 XSS 攻击吗? 如何防止 ?

XSS 是跨站脚本攻击，首先是利用跨站脚本漏洞以一个特权模式去执行攻击者构造的脚本，然后利用不安全的 Activex 控件执行恶意的行为。 使用 htmlspecialchars()函数对提交的内容进行过滤，使字符串里面的特殊符号实体化。

附：Thinkphp框架提供了扩展函数包remove_xss

28、PHP7新特性有哪些

1.运算符（NULL 合并运算符）

2、函数返回值类型声明

3、标量类型声明

4.use 批量声明

29、接口和抽象类的区别是什么?

答:抽象类是一种不能被实例化的类，只能作为其他类的父类来使用。抽象类是通过关键字 abstract 来声明的。抽象类与普通类相似，都包含成员变量和成员方法，两者的区别在于，抽象类中至少要包含一个抽象方法，抽象方法没有方法体，该方法天生就是要被子类重写的。抽象方法的格式为：abstract function abstractMethod();

接口是通过 interface 关键字来声明的，接口中的成员常量和方法都是 public 的，方法可以不写关键字 public，接口中的方法也是没有方法体。接口中的方法也天生就是要被子类实现的。抽象类和接口实现的功能十分相似，最大的不同是接口能实现多继承。在应用中选择抽象类还是接口要看具体实现。子类继承抽象类使用 extends，子类实现接口使用 implements。

二、HTTP/HTTPS协议

1、请问http状态码以1、2、3、4、5开头的分别代表什么意思?

1XX临时/信息响应

2XX成功

3XX重定向

4XX客户端/请求错误

5XX服务器错误

详细:

200——交易成功
301——删除请求数据
302——在其他地址发现了请求数据
400——错误请求，如语法错误
401——请求授权失败
403——请求不允许
404——没有发现文件、查询或URl
500——服务器产生内部错误
501——服务器不支持请求的函数
502——服务器暂时不可用，有时是为了防止发生系统过载

2、用户输入一个网址（www.baidu.com）,到用户看到对应的网页加载完毕，此过程都发生了什么?

1.域名解析（DNS解析）

2.发起TCP的3次握手

3.建立TCP连接后发起http请求

4.服务器端响应http请求，浏览器得到html代码

5.浏览器解析html代码，并请求html代码中的资源

6.浏览器对页面进行渲染呈现给用户

3、解释TCP的三次握手。

1） Client首先发送一个连接试探，ACK=0 表示确认号无效，SYN = 1 表示这是一个连接请求或连接接受报文，同时表示这个数据报不能携带数据，seq = x 表示Client自己的初始序号（seq = 0 就代表这是第0号包），这时候Client进入syn_sent状态，表示客户端等待服务器的回复

2） Server监听到连接请求报文后，如同意建立连接，则向Client发送确认。TCP报文首部中的SYN 和 ACK都置1，ack = x + 1表示期望收到对方下一个报文段的第一个数据字节序号是x+1，同时表明x为止的所有数据都已正确收到（ack=1其实是ack=0+1,也就是期望客户端的第1个包），seq = y 表示Server 自己的初始序号（seq=0就代表这是服务器这边发出的第0号包）。这时服务器进入syn_rcvd，表示服务器已经收到Client的连接请求，等待client的确认。

3） Client收到确认后还需再次发送确认，同时携带要发送给Server的数据。ACK 置1 表示确认号ack= y + 1 有效（代表期望收到服务器的第1个包），Client自己的序号seq= x + 1（表示这就是我的第1个包，相对于第0个包来说的），一旦收到Client的确认之后，这个TCP连接就进入Established状态，就可以发起http请求了。

1536136824697

4、描述一下 HTTP 与 HTTPS 的区别?

https 协议需要到 ca 申请证书，一般免费证书很少，需要交费。 http 是超文本传输协议，信息是明文传输，https 则是具有安全性的 ssl 加密传输协议 http 和 https 使用的是完全不同的连接方式用的端口也不一样,前者是 80,后者是 443。 http 的连接很简单,是无状态的 HTTPS 协议是由 SSL+HTTP 协议构建的可进行加密传输、身份认证的网络协议 要比 http 协议安全

5、什么是Http协议无状态协议?怎么解决Http协议无状态协议?

- 无状态协议对于事务处理没有记忆能力缺少状态意味着如果后续处理需要前面的信息
  - 也就是说，当客户端一次HTTP请求完成以后，客户端再发送一次HTTP请求，HTTP并不知道当前客户端是一个”老用户“。
- 可以使用Cookie来解决无状态的问题，Cookie就相当于一个通行证，第一次访问的时候给客户端发送一个Cookie，当客户端再次来的时候，拿着Cookie(通行证)，那么服务器就知道这个是”老用户“。

6、常用的HTTP方法有哪些?

- GET： 用于请求访问已经被URI（统一资源标识符）识别的资源，可以通过URL传参给服务器
- POST：用于传输信息给服务器，主要功能与GET方法类似，但一般推荐使用POST方式。
- PUT： 传输文件，报文主体中包含文件内容，保存到对应URI位置。
- HEAD： 获得报文首部，与GET方法类似，只是不返回报文主体，一般用于验证URI是否有效。
- DELETE：删除文件，与PUT方法相反，删除对应URI位置的文件。
- OPTIONS：查询相应URI支持的HTTP方法。

7、URI和URL的区别

URI，是uniform resource identifier，统一资源标识符，用来唯一的标识一个资源。

- Web上可用的每种资源如HTML文档、图像、视频片段、程序等都是一个来URI来定位的
- URI一般由三部组成：
- ①访问资源的命名机制
- ②存放资源的主机名
- ③资源自身的名称，由路径表示，着重强调于资源。

URL是uniform resource locator，统一资源定位器，它是一种具体的URI，即URL可以用来标识一个资源，而且还指明了如何locate这个资源。

- URL是Internet上用来描述信息资源的字符串，主要用在各种WWW客户程序和服务器程序上，特别是著名的Mosaic。
- 采用URL可以用一种统一的格式来描述各种信息资源，包括文件、服务器的地址和目录等。URL一般由三部组成：
- ①协议(或称为服务方式)
- ②存有该资源的主机IP地址(有时也包括端口号)
- ③主机资源的具体地址。如目录和文件名等

8、HTTP请求报文与响应报文格式

请求报文包含四部分：

- a、请求行：包含请求方法、URI、HTTP版本信息
- b、请求首部字段
- c、请求内容实体
- d、空行

响应报文包含四部分：

- a、状态行：包含HTTP版本、状态码、状态码的原因短语
- b、响应首部字段
- c、响应内容实体
- d、空行

三、ThinkPHP5\Laravel5

1、ThinkPHP5框架的优点

ThinkPHP5.0版本引入了更多的PHP新特性，优化了核心，减少了依赖，实现了真正的惰性加载，支持composer，并针对API开发做了大量的优化，包括路由、日志、异常、模型、数据库、模板引擎和验证等模块都已经重构。是WEB开发和API开发的首选框架。

主要特性：

- 规范：遵循PSR-2、PSR-4规范，Composer及单元测试支持；
- 严谨：异常严谨的错误检测和安全机制，详细的日志信息，为你的开发保驾护航；
- 灵活：减少核心依赖，扩展更灵活、方便，支持命令行指令扩展；
- API友好：出色的性能和REST支持、远程调试，更好的支持API开发；
- 高效：惰性加载，及路由、配置和自动加载的缓存机制；
- ORM：重构的数据库、模型及关联，MongoDb支持；

2、ThinkPHP5的生命周期是如何执行的

1、入口文件

> 般入口文件以定义一些常量为主，支持的常量请参考后续的内容或者附录部分。

2、引导文件

```
加载系统常量定义;

加载环境变量定义文件;
注册自动加载机制;
注册错误和异常处理机制;
加载惯例配置文件;
执行应用;
```

3、注册自动加载

```
系统会调用 `Loader::register()`方法注册自动加载，在这一步完成后，所有符合规范的类库（包括`Composer`依赖加载的第三方类库）都将自动加载。
```

4、注册错误和异常机制

5、应用初始化

6、URL访问检测

7、路由检测

8、分发请求

9、响应输出

10、应用结束

3、ThinkPHP5的URL访问格式是怎样的?

ThinkPHP5.0在没有启用路由的情况下典型的URL访问规则是：

> http://serverName/index.php/模块/控制器/操作/[参数名/参数值...]
>
> URL采用PATH_INFO访问地址，其中PATH_INFO的分隔符是可以设置的。
>
> 注意：5.0取消了URL模式的概念，并且普通模式的URL访问不再支持。

4、ThinkPHP5路由模式有几种?

一、普通模式

关闭路由，完全使用默认的`PATH_INFO`方式URL：

```
'url_route_on'  =>  false,
```

路由关闭后，不会解析任何路由规则，采用默认的`PATH_INFO`模式访问URL：

```
http://serverName/index.php/module/controller/action/param/value/...
```

> 但仍然可以通过操作方法的参数绑定、空控制器和空操作等特性实现URL地址的简化。

可以设置`url_param_type`配置参数来改变pathinfo模式下面的参数获取方式，默认是按名称成对解析，支持按照顺序解析变量，只需要更改为：

```
// 按照顺序解析变量
'url_param_type'    =>  1,
```

二、混合模式

开启路由，并使用路由定义+默认`PATH_INFO`方式的混合：

```
'url_route_on'  =>  true,
'url_route_must'=>  false,
```

该方式下面，只需要对需要定义路由规则的访问地址定义路由规则，其它的仍然按照第一种普通模式的`PATH_INFO`模式访问URL。

三、强制模式

开启路由，并设置必须定义路由才能访问：

```
'url_route_on'          =>  true,
'url_route_must'        =>  true,
```

这种方式下面必须严格给每一个访问地址定义路由规则（包括首页），否则将抛出异常。

首页的路由规则采用`/`定义即可，例如下面把网站首页路由输出`Hello,world!`

```
Route::get('/',function(){
    return 'Hello,world!';
});
```

5、TP5在安全方面做了哪些操作?

输入安全：

- 设置`public`目录为唯一对外访问目录，不要把资源文件放入应用目录；
- 开启表单令牌验证避免数据的重复提交，能起到`CSRF`防御作用；
- 使用框架提供的请求变量获取方法（Request类`param`方法及`input`助手函数）而不是原生系统变量获取用户输入数据；
- 对不同的应用需求设置`default_filter`过滤规则（**默认没有任何过滤规则**），常见的安全过滤函数包括`stripslashes`、`htmlentities`、`htmlspecialchars`和`strip_tags`等，请根据业务场景选择最合适的过滤方法；
- 使用验证类或者验证方法对业务数据设置必要的验证规则；
- 如果可能开启强制路由或者设置MISS路由规则，严格规范每个URL请求；

数据库安全：

- 尽量少使用数组查询条件而应该使用查询表达式替代；
- 尽量少使用字符串查询条件，如果不得已的情况下 使用手动参数绑定功能；
- 不要让用户输入决定要查询或者写入的字段；
- 对于敏感数据在输出的时候使用`hidden`方法进行隐藏；
- 对于数据的写入操作应当做好权限检查工作；
- 写入数据严格使用`field`方法限制写入字段；
- 对于需要输出到页面的数据做好必要的`XSS`过滤；

6、Laravel5.6框架的生命周期是如何实行的?

Laravel 应用的所有请求入口都是`public/index.php`文件，所有请求都会被 web 服务器（Apache/Nginx）导向这个文件。

`index.php`文件载入 Composer 生成的自动加载设置，然后从`bootstrap/app.php`脚本获取 Laravel 应用实例，Laravel 的第一个动作就是创建服务容器实例

HTTP/Console 内核

接下来，请求被发送到 HTTP 内核或 Console 内核（分别用于处理 Web 请求和 Artisan 命令），这取决于进入应用的请求类型。这两个内核是所有请求都要经过的中央处理器，现在，就让我们聚焦在位于`app/Http/Kernel.php`的 HTTP 内核。

HTTP 内核继承自`Illuminate\Foundation\Http\Kernel`类，该类定义了一个`bootstrappers`数组，这个数组中的类在请求被执行前运行，这些`bootstrappers`配置了错误处理、日志、检测应用环境以及其它在请求被处理前需要执行的任务。

HTTP 内核还定义了一系列所有请求在处理前需要经过的 HTTP 中间件，这些中间件处理 HTTP 会话的读写、判断应用是否处于维护模式、验证 CSRF 令牌等等。

HTTP 内核的`handle`方法签名相当简单：获取一个`Request`，返回一个`Response`，可以把该内核想象作一个代表整个应用的大黑盒子，输入 HTTP 请求，返回 HTTP 响应。

服务提供者

内核启动过程中最重要的动作之一就是为应用载入服务提供者，应用的所有服务提供者都被配置在`config/app.php`配置文件的`providers`数组中。首先，所有提供者的`register`方法被调用，然后，所有提供者被注册之后，`boot`方法被调用。

服务提供者负责启动框架的所有各种各样的组件，比如数据库、队列、验证器，以及路由组件等，正是因为他们启动并配置了框架提供的所有特性，所以服务提供者是整个 Laravel 启动过程中最重要的部分。

分发请求

一旦应用被启动并且所有的服务提供者被注册，`Request`将会被交给路由器进行分发，路由器将会分发请求到路由或控制器，同时运行所有路由指定的中间件。

7、Laravel框架的优点

回答案例一、

1、代码比较明白易懂，跟英语句子差不多，关键词就是函数，

2、文档非常丰富，社区也是非常活跃，现在全球范围内占有率最高，基本上所有的问题都可以找到答案； 3、大量的第三方开源库（composer收录的超过5500个包），可以快速方便的实现模块功能，第三方优秀的包官方都有详细使用手册。例如：laravel/collective 4、安全机制非常齐全，提交表单的数据验证（验证有差不多80种，能想到的基本都有），提交数据时产生随机_token验证，避免非法提交,能避免跨域攻击； 5、中间件和路由，对访问进行过滤及控制，调用函数类和方法前进行判断请求的合法性，避免非法请求；

6、错误处理机制简单好用，如果出错直接调用$error->all()，即可输出全部错误，对表单验证尤其好用；

回答案例二、

1.强大的 rest router:用简单的回调函数就可以调用,快速绑定 controller 和 router 2.artisan:命令行工具,很多手动的工作都自动化 3.可继承的模板,简化 view 的开发和管理 4.blade 模板:渲染速度更快 5.ORM 操作数据库 6.migration:管理数据库和版本控制 7.测试功能也很强大 8.composer 也是亮点。

laravel 框架引入了门面,依赖注入,Ioc 模式,以及各种各样的设计模式等

8、laravel 的路由有哪几种

①匿名函数路由，路由对应的处理程序可以直接写在 routes.php 中的匿名函数中 ②控制器路由，将路由映射到控制器中的方法上 ③资源路由 ④控制器路由，将请求对应到控制器类的方法里，方法的名字规范是：请求动词+方法名，比如 getIndex(), postSave() 还可以对路由使用 group()进行分组，分组可以按照路由前缀，命名空间，应用的中间件，域名进行分组， 路由可以传递参数，绑定模型等

9、laravel 中的服务提供器的作用是什么

服务容器可以在项目启动期间扩展应用的功能，具体表现为可以在项目中扩展依赖注入容器，事件监听，中间件，路由，宏，视图 Composer，发布静态文件，发布配置文件。

10、请概述中间件的作用

中间件是在浏览器和控制器间过滤请求的，比如过滤未登录的用户的请求和过滤掉跨站请求伪造的请求。在 中间件中定义一个 terminate 方法，该方法内的代码可在响应发送到浏览器之后执行代码，比如将 session 保存到数据库，计算并记录项目运行的时间，将响应值缓存到 memcache 等。

11、THINKPHP5如何实现关联模型?

数据库的关联模型主要有一对一，一对多，多对多。

1. 一对一
    1. hasOne，一个用户都有一个个个人资料
    2. belongsTo ，相对关联，一份档案对应（属于）一个用户
2. 一对多
    1. hasMany一篇文章可以有多个评论
    2. belongsTo 相对关联，一条评论对应一个评论
3. 多对多
    1. belongsToMany 一个用户对应多个角色，一个角色对应多个用户
    2. belongsToMany 相对关联

四、接口

1、什么是RESTful API?

符合REST设计标准的API，即RESTful API。

RESTful 既是 Representational State Transfer，（资源）表现层的状态转化。

表现层：资源表现层。"资源"是一种信息实体，

比如，文本可以用txt格式表现，也可以用HTML格式、XML格式、JSON格式表现，甚至可以采用二进制格式；图片可以用JPG格式表现，也可以用PNG格式表现。

URI只代表资源的实体，不代表它的形式。严格地说，有些网址最后的".html"后缀名是不必要的，因为这个后缀名表示格式，属于"表现层"范畴，而URI应该只代表"资源"的位置。它的具体表现形式，应该在HTTP请求的头信息中用Accept和Content-Type字段指定，这两个字段才是对"表现层"的描述。

状态转化：

访问一个网站，就代表了客户端和服务器的一个互动过程。在这个过程中，势必涉及到数据和状态的变化。

互联网通信协议HTTP协议，是一个无状态协议。这意味着，所有的状态都保存在服务器端。因此，**如果客户端想要操作服务器，必须通过某种手段，让服务器端发生"状态转化"（State Transfer）。而这种转化是建立在表现层之上的，所以就是"表现层状态转化"。**

客户端用到的手段，只能是HTTP协议。具体来说，就是HTTP协议里面，四个表示操作方式的动词：GET、POST、PUT、DELETE。它们分别对应四种基本操作：**GET用来获取资源，POST用来新建资源（也可以用于更新资源），PUT用来更新资源，DELETE用来删除资源。**

http://www.ruanyifeng.com/blog/2014/05/restful_api.html

2、微信API开发中ACCESS TOKEN还没过期，但提示“失效”。请说出解决方案

微信access_token是公众号的全局唯一票据，公众号调用各接口时都需使用access_token。正常情况下access_token有效期为7200秒，重复获取将导致上次获取的access_token失效。

一般情况下，我们会将 access_token 本地缓存化，在缓存时间内进行调取缓存使用，这样可以保证在一定时间内不在请求微信接口，提交功能的访问效率等等。但是遇到高并发的情况，则有可能重复去获取access_token,而我们存储的缓存未必是最后一次请求的结果，这样将直接导致access_token失效。

解决方案：

1.调用一个没有调用次数限制的接口看看access_token是否过期

2.做一个定时任务，每分钟去监测，看是否过期

3、接口安全方面如何设计?

我们当时是这么做的，使用 HTTP 的 POST 方式,对固定参数+附加参数进行数字签名,使用的是 md5 加密,比如:我想通过标题获取一个信息,在客户端使用 信息标题+日期+双方约定好的一个 key 通过 md5 加密生成一个签名(sign),然后作为参数传递到服务器端,服务器端使用同样的方法进行校验,如何接受过来的 sign 和我们通过算法算的值相同，证明是一个正常的接口请求，我们才会返回相应的接口数据。

4、写过接口吗，怎么定义接口的

答：写过。接口分为两种：一种是数据型接口，一种是应用型接口。

数据型接口：是比抽象类更抽象的某种“结构”——它其实不是类，但是跟类一样的某种语法结构，是一种结构规范，规范我们类要以什么格式进行定义，一般用于团队比较大，分支比较多的情况下使用。

应用型接口： API（application interface） 数据对外访问的一个入口我主要是参与的 APP 开发中接口的编写，客户端需要什么样的数据，我们就给他们提供相应的数据，数据以 json/xml 的格式返回，并且配以相应的接口文档。

五、Javascrip/Jquery/Ajax

1、事件冒泡和事件委托

事件冒泡:当一个元素上的事件被触发的时候,比如说鼠标点击了一个按钮,同样的事件将会在那个元素的所有祖先元素中被触发. 这一过程被称为事件冒泡;这个事件从原始元素开始一直冒泡到 DOM 树的最上层

阻止事件冒泡和捕获: 标准浏览器 e. stopPropagation(); IE9 之前 event.canceBubble=true;

阻止默认事件:为了不让 a 点击之后跳转,我们就要给他的点击事件进行阻止

```
return false; e.preventDefault();
```

事件委托:让利用事件冒泡的原理,让自己的所触发的事件,让他的父元素代替执行

2、添加删除替换插入到某个节点的方法

obj.appendChild() //追加子节点 obj.insertBefore() //插入子节点,参数 2 为 null 时等同于 appendChild obj.replaceChild() //替换子节点,参数:新节点、旧节点(应该是没用过)

obj.removeChild() //删除子节点 此外,jQuery 还提供了 append、perpend、after、before,appendTo、prependTo、insertAfter、insertBefore

3、Javascript 作用域链?

理解变量和函数的访问范围和生命周期, 全局作用域与局部作用域的区别, JavaScript 中没有块作用域,函数的嵌套形成不同层次的作用域,嵌套的层次形成链式形式,通过作用域链查找属性的规则需要深入理解. 当一个函数内部访问的成员不存在时,则向其外部的函数进行访问,如果再访问不到,继续向外查找,直到全局作用域(script), 没有则报错

sacript 内作为 0 级作用域(全局变量和函数),0 级作用域的函数内(全局函数内)为 1 级作用域,其内的第二层函数为 2 级作用域.....访问变量时,从最后内层的作用域开始向 0 级查找

1536309834831

4、谈谈 this 对象的理解.

1.普通函数中 this 是 window

2.构造函数中 this 是当前的实例对象

3.对象中的 this 是指向其本身

4.原型对象的方法中 this 是当前的实例对象

5.计时器中 this 是 window

6.事件处理函数,this 是触发该事件的

5、什么是闭包(closure),为什么要用它?

简单的理解是函数的嵌套形成闭包,闭包包括函数本身以及它的外部作用域,函数 a 内嵌套 b,且返回 b,当调用函数 a 时,用变量接收函数 b,就形成了闭包 (这里还可以扩展一下作用域和作用域链的问题,js 只有全局作用域、函数作用域,没有块级作用域{},作用域链的查找方式, 由内向外查找,函数作用域内的变量,全局不能直接访问,要想访问,就要用闭包) 优点:使用闭包可以形成独立的空间,缓存数据,延长变量的生命周期

缺点:延长了作用域链,需要释放的变量不能及时释放,可能引发内存泄漏(这里又能扩展,内存泄漏也称作"存储渗漏",用动态存储分配函数动态开辟的空间,在使用完毕后未释放,结果导致一直占据该内存单元.直到程序结束.(其实说白了就是该内存空间使用完毕之后未回收)即所谓内存泄漏.内存泄漏形象的比喻是"操作系统可提供给所有进程的存储空间正在被某个进程榨干",最终 结果是程序运行时间越长,占用存储空间越来越多,最终用尽全部存储空间,整个系统崩溃.)

6、对 JSON 的了解?

JSON 指的是 JavaScript 对象表示法(JavaScript Object Notation) 1.轻量级数据交互格式 2.可以形成复杂的嵌套格式 3.解析非常方便 4.易于读写,占用带宽小 JSON 的表现形式 1.var arrJson = [{"name":"xiaoming","age": 18}, {"name":"xiaohong","age": 19}]; 2.var strJson = {"name":"Lily","age": 12};

7、DOM 操作——怎样添加、移除、移动、复制、创建和查找节点?

1.添加节点 createElement()、createTextNode()、craeteDocumentFragment() 2.添加、移除、替换、插入 appendChild()、removeChild()、replaceChild()、insertBefore() 3.查找节点 getElementById() getElementsByTagName() getElementsByClassName() -->IE9+ querySelect() -->IE8+ querySelectAll() -->IE8+

8、JavaScript 原型,原型链? 有什么特点?

1. 原型对象也是普通的对象,是对象一个自带隐式的 **proto** 属性,原型也有可能有自己的原型,如果一个原型对象的原型不为 null 的话,我们就称之为原型链
2. 原型链是由一些用来继承和共享属性的对象组成的(有限的)对象链(基本思想是利用原型让一个引用类型继承另一个

引用类型的属性和方法)
理解:每个构造函数都有一个原型对象,原型对象都包含一个指向构造函数的指针,而实例都包含一个指向原型对象
的内部指针.那么,假如我们让原型对象等于另一个类型的实例,结果会怎么样呢?显然,此时的原型对象将包含一个指向另
一个原型的指针,相应地,另一个原型中也包含着一个指向另一个构造函数的指针.假如另一个原型又是另一个类型的实例, 那么上述关系依然成立,如此层层递进,就构成了实 例与原型的链条.这就是所谓原型链的基本概念

3. 当我们需要一个属性的时,JavaScript 引擎会先看当前对象中是否有这个属性,如果没有的话,就会查找他的 Prototype 对象是否有这个属性

9、你用过 require.js 吗?它有什么特性?

1.实现 js 文件的异步加载,避免网页失去响应; 2.管理模块之间的依赖性,便于代码的编写和维护

10、前端开发的优化问题

1. 减少 http 请求次数:CSS Sprites, JS、CSS 源码压缩、图片大小控制合适;网页 Gzip,CDN 托管,data 缓存 ,图片服
务器.
2. 前端模板 JS+数据,减少由于 HTML 标签导致的带宽浪费,前端用变量保存 AJAX 请求结果,每次操作本地变量,不用请求,减少请求次数
3. 用 innerHTML 代替 DOM 操作,减少 DOM 操作次数,优化 javascript 性能.
当需要设置的样式很多时设置 className 而不是直接操作 style.
4. 少用全局变量、缓存 DOM 节点查找的结果.减少 IO 读取操作.
5. 避免使用 CSS Expression(css 表达式)又称 Dynamic properties(动态属性).
6. 图片预加载,将样式表放在顶部,将脚本放在底部 加上时间戳.
7. 避免在页面的主体布局中使用 table,table 要等其中的内容完全下载之后才会显示出来,显示比 div+css 布局

11、Ajax 是什么?它最大的特点是?优缺点?

Ajax (asynchronous javascript and xml),即异步 JavaScript 和 xml,主要用来实现客户端与服务器端的异步数据交互,实现页面的局部刷新.早期浏览器并不能原生支持 ajax,可以使用隐藏帧(iframe)方式变相实现异步效果,后来的浏览器提供了对 ajax 的原生支持 .

ajax 的最大的特点+优点: Ajax 可以实现异步通信效果,实现页面局部刷新,避免用户不断刷新或者跳转页面,带来更好的用户体验;按需获取数据,节约带宽资源;

ajax 的缺点 1、ajax 不支持浏览器 back 按钮.

2、安全问题, AJAX 暴露了与服务器交互的细节,诸如跨站点脚步攻击、SQL 注入攻击.

3、对搜索引擎的支持比较弱.

4、AJAX 不能很好支持移动设备.

12、如何创建一个 Ajax?简述 ajax 的过程.ajax 的交互模型?

1.创建 XMLHttpRequest 对象,也就是创建一个异步调用对象 2.创建一个新的 HTTP 请求,并指定该 HTTP 请求的方法、URL 及验证信息(即请求报文行) xhr.open('get','5-register-get.php?name='+name);

3.设置请求报文头;(Content-Type:请求资源的 MIME 类型) xhr.setRequestHeader('Content-Type','application/x-www-form-urlencoded');//get 方式不需要 4.设置响应 HTTP 请求状态变化的函数

5. 发送 HTTP 请求
   xhr.send(null); 6.获取异步调用返回的数据
   7.使用 JavaScript 和 DOM 实现局部刷新

13、ajax 请求时,如何解析 json 数据

使用 eval() 或者 JSON.parse() 鉴于安全性考虑,推荐使用 JSON.parse()更靠谱,对数据的安全性更好.
【JSON.parse(): 将 JSON 字符串转换成对象.】

14、阐述一下异步加载

1. 异步加载的方案: 动态插入 script 标签
2. 通过 ajax 去获取 js 代码,然后通过 eval 执行
3. script 标签上添加 defer 或者 async 属性
4. 创建并插入 iframe,让它异步执行 js

15、请解释一下 JavaScript 的同源策略.

同源策略是客户端脚本(尤其是 Javascript)的重要的安全度量标准.它最早出自 Netscape Navigator2.0,其目的是防止某个文档或脚本从多个不同源装载.所谓同源指的是:协议,域名,端口相同,同源策略是一种安全协议,指一段脚本只能读取来自同一来源的窗口和文档的属性.

16、如何解决跨域问题?

理解跨域的概念:协议、域名、端口都相同才同域,否则都是跨域

出于安全考虑,服务器不允许 ajax 跨域获取数据,但是可以跨域获取文件内容.基于这一点,可以动态创建 script 标签,使用标签的 src 属性访问 js 文件的形式,获取 js 脚本,并且这个 js 脚本中的内容是函数调用,该函数调用的参数是服务器返回的数据.为了获取这里的参数数据,需要事先在页面中定义回调函数,在回调函数中处理服务器返回的数据,这就是解决跨域问题的主流解决方案 .

当然也可以利用PHP做代理来实现。

17、解释 jsonp 的原理,以及为什么不是真正的 ajax

Jsonp 并不是一种数据格式,而 json 是一种数据格式,jsonp 是用来解决跨域获取数据的一种解决方案,具体是通过动态创建script 标签,然后通过标签的 src 属性获取 js 文件中的 js 脚本,该脚本的内容是一个函数调用,参数就是服务器返回的数据,为了处理这些返回的数据,需要事先在页面定义好回调函数,本质上使用的并不是 ajax 技术

18、页面编码和被请求的资源编码如果不一致如何处理?

对于 ajax 请求传递的参数,如果是 get 请求方式,如果传递中文参数,由于不同的浏览器对参数编码的处理方式不同,在有些浏览器会乱码,所以对于 get 请求的参数需要使用 encodeURIComponent 函数对参数进行编码处理,后台开发语言都有相应的解码 api.对于 post 请求不需要进行编

六、NodeJS

1、对 Node 的优点和缺点提出了自己的看法

优点:

1) 因为 Node 是基于事件驱动和无阻塞的,所以非常适合处理并发请求,因此构建在 Node 上的代理服务器相比其他技术实现(如 Ruby)的服务器表现要好得多.

2) 与 Node 代理服务器交互的客户端代码是由 javascript 语言编写的,因此客户端和服务器端都用同一种语言编写,这是非常美妙的事情

缺点:

1) Node 是一个相对新的开源项目,所以不太稳定,它总是一直在变.

2)容易写出糟糕的代码,callback 的执行流程有时并不是很符合直觉,需要定期 review 和重构来加以避免.

3) 单线程,要考虑的东西比较多,一旦这个线程死了,就全站瘫痪

2、Node.js 的适用场景?

1)实时应用:如 在线聊天,通知推送等 2)分布式应用:通过高效的并行 IO 使用已有的数据 3)工具类应用:海量的,小到前端压缩部署(如 grunt),大到桌面图形应用程序 4)游戏类应用:领域对实时和并发有很高的要求的项目 5)利用稳定接口提升 web 渲染能力 6)前后端编程语言环境统一

3、解释一下 Backbone 的 MVC 实现方式?

Backbone 将数据呈现为模型, 你可以创建模型、对模型进行验证和销毁,甚至将它保存到服务器. 当 UI 的变化引起模型属性改变时,模型会触发"change"事件; 所有显示模型数据的视图会接收到该事件的通知,继而视图重新渲染. 你无需查找 DOM来搜索指定 id 的元素去手动更新 HTML

七、VueJS

1、Vue.js 是什么

Vue.js (读音 /vjuː/，类似于 **view**) 是一套构建用户界面的**渐进式框架**。与其他重量级框架不同的是，Vue 采用自底向上增量开发的设计。Vue 的核心库只关注视图层，它不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与单文件组件和 Vue 生态系统支持的库结合使用时，Vue 也完全能够为复杂的单页应用程序提供驱动。

2、Vue.jS的特点有哪些?

简洁：页面由 HTML 模板+J瀆澮濬 数据+V灈澽 实例组成 数据驱动：自动计算属性和追踪依赖的模板表达式 组件化：用可复用、解耦的组件来构造页面 轻量：代码量小，不依赖其他库 快速：精确有效批量 DOM 更新 模板友好：可通过npm,bower等多种方式安装，很容易融入

3、vue 中的 MVVM 模式?

MVVM拆分之后，由三部分组成，分别是

- M

    - 主要用来负责数据的存储
- V
    - 主要用来页面的展示
- VM
    - 负责项目中逻辑的展示，M 与 V 的连接枢纽

一个应用项目的开发，都会有数据来进行支撑，一般由M进行表示。 将数据（M）封装好之后，需要呈现在浏览器或者其他设备中，并展示到视图（V）中呈现给我们的用户。 那么数据（M）的改变如何能传递给视图页面（V），视图页面（V）的改变又如何进一步让数据察觉并更新呢？这时候，VM的作用就显示了出来，VM就是负责将数据请求过来之后，更新视图页面，同时将视图页面的改变反应到数据中，可以说VM 就是沟通 V 与 M之间的桥梁，起到枢纽连接的作用。

4、说出至少 4种 vue当中的指令和它的用法?

答:v-if:判断是否隐藏;

v-for:数据循环出来;

v-bind:class:绑定一个属性;

v- model:实现双向绑定

5、请详细说下你对 vue生命周期的理解?

答:总共分为 8 个阶段创建前/后，载入前/后，更新前/后，销毁前/后。 创建前/后:在 beforeCreated 阶段，vue 实例的挂载元素$el 和数据对象 data 都为 undefined，还未初始化。 在 created 阶段，vue 实例的数据对象 data 有了，$el 还没 有。 载入前/后:在 beforeMount 阶段，vue 实例的$el 和 data 都初始化了，但还是挂载之前 为虚拟的 dom 节点， data.message 还未替换。在 mounted 阶段，vue 实例挂载完成， data.message 成功渲染。 更新前/后:当 data 变化时，会触发 beforeUpdate 和 updated 方法。 销毁前/后:在执行 destroy 方法后，对 data 的改变不会再触发周期函数，说明此时 vue 实例已经解除了事件监听以及和 dom 的绑定，但是 dom 结构依然存在

6、v-show 指令，v-if 的区别

条件渲染指令，与 v-if 不同的是，无论 v-show 的值为 true 或 false，元素都会存在于 HTML 代码中；而只有当 v-if 的值 为 true，元素才会存在于 HTML 代码中。 v-show 指令只是设置了元素 CSS 的 style 值

7、如何让 css 只在当前组件中起作用

在每一个 vue 组件中都可以定义各自的 css，js，如果希望组件内写的 css 只对当前组件起作用，只需要在 style 中写入scoped，即

8、指令 keep-alive

在 vue-router 写着 keep-alive，keep-alive 的含义：

如果把切换出去的组件保留在内存中，可以保留它的状态或避免重新渲染。为此可以添加一个 keep-alive 指令

9、路由嵌套

路由嵌套会将其他组件渲染到该组件内，而不是进行整个页面跳转 router-view 本身就是将组件渲染到该位置，想要进行页面跳转，就要将页面渲染到根组件，在起始配置路由时候写到：

var App = Vue.extend({ root });

router.start(App,'#app');

这里首先将根组件注册进来，用于将路由中配置好的各个页面渲染出来，然后将根组件挂载到与#app 匹配的元素上.

10.vuejs 中使用事件名

在 vuejs 中，我们经常要绑定一些事件，有时候给 DOM 元素绑定，有时候给组件绑定。绑定事件在 HTML 中用 v-on:click-"event",这时 evet 的名字不要出现大写，因为在 1.x 中不区分大小写，所以如果我们在 HTML 写 v-on:click="myEvent"而在 js 中写 myEvent 就出错误，所以在 vuejs 的 1.x 绑定事件时候，要尽量避免使用大写字母。在 2.0 中没有该限制！

11、嵌套路由怎么定义?

在实际项目中我们会碰到多层嵌套的组件组合而成，但是我们如何实现嵌套路由呢？因此我们需要在 VueRouter 的参数中使用 children 配置，这样就可以很好的实现路由嵌套。

index.html，只有一个路由出口

12、active-class 是哪个组件的属性?嵌套路由怎么定义?

答:vue-router 模块的 router-link 组件。

13、怎么定义 vue-router 的动态路由?怎么获取传过来的动态参数?

答:在 router 目录下的 index.js 文件中，对 path 属性加上/:id。 的 params.id

使用 router 对象

14、vue-router 有哪几种导航钩子?

三种，

第一种：是全局导航钩子：router.beforeEach(to,from,next)，作用：跳转前进行判断拦截。

第二种：组件内的钩子

第三种：单独路由独享组件

15、scss 是什么？在 vue.cli 中的安装使用步骤是？有哪几大特性?

css 的预编译。

使用步骤：

第一步：用 npm 下三个 loader（sass-loader、css-loader、node-sass）

第二步：在 build 目录找到 webpack.base.config.js，在那个 extends 属性中加一个拓展.scss

第三步：还是在同一个文件，配置一个 module 属性

第四步：然后在组件的 style 标签加上 lang 属性 ，例如：lang=”scss”

有哪几大特性:

1、可以用变量，例如（$变量名称=值）；

2、可以用混合器，例如（）

3、可以嵌套

16、mint-ui 是什么？怎么使用？说出至少三个组件使用方法?

基于 vue 的前端组件库。npm 安装，然后 import 样式和 js，vue.use（mintUi）全局引入。在单个组件局部引入：import {Toast} from ‘mint-ui’。

组件一：Toast(‘登录成功’);

组件二：mint-header;

组件三：mint-swiper

17、v-model 是什么?怎么使用? vue 中标签怎么绑定事件?

答:可以实现双向绑定，指令(v-class、v-for、v-if、v-show、v-on)。vue 的model 层的 data 属性。

绑定事件:

18、axios 是什么?怎么使用?描述使用它实现登录功能的流程?

答:请求后台资源的模块。npm install axios -S 装好，然后发送的是跨域，需在配置文 件中 config/index.js

进行设置。后台如果是 Tp5 则定义一个资源路由。js 中使用 import 进来，然后.get 或.post。返回在.then 函

数中如果成功，失败则是在.catch 函 数中

19、axios+tp5 进阶中，调用 axios.post(‘api/user’)是进行的什么操作? axios.put(‘api/user/8')呢?

答:跨域，添加用户操作，更新操作。

20、vuex 是什么？怎么使用？哪种功能场景使用它?

vue 框架中状态管理。在 main.js 引入 store，注入。新建了一个目录 store，….. export 。场景有：单页应用中，组件之间的状态。音乐播放、登录状态、加入购物车

21、mvvm 框架是什么？它和其它框架（jquery）的区别是什么？哪些场景适合?

一个 model+view+viewModel 框架，数据模型 model，viewModel 连接两个

区别：vue 数据驱动，通过数据来显示视图层而不是节点操作。

场景：数据操作比较多的场景，更加便捷

22、自定义指令(v-check、v-focus)的方法有哪些?它有哪些钩子函数?还有哪些钩子 函数参数?

答:全局定义指令:在 vue 对象的 directive 方法里面有两个参数，一个是指令名称，另外 一个是函数。组件内定义

指令:directives

钩子函数:bind(绑定事件触发)、inserted(节点插入的时候触发)、update(组件内相 关更新)

钩子函数参数:el、binding

23、vue-router 是什么?它有哪些组件?

答:vue 用来写路由一个插件。router-link、router-view

24、Vue 的双向数据绑定原理是什么?

答:vue.js 是采用数据劫持结合发布者-订阅者模式的方式，通过 Object.defineProperty()来劫持各个属性的 setter，getter，在数据变动时发布消息 给订阅者，触发相应的监听回调。

具体步骤:第一步:需要 observe 的数据对象进行递归遍历，包括子属性对象的属性，都加上 setter

和 getter这样的话，给这个对象的某个值赋值，就会触发 setter，那么就能监听到了数据变化

第二步:compile 解析模板指令，将模板中的变量替换成数据，然后初始化渲染页面视图，并 将每个指令对应的节点绑定更新函数，添加监听数据的订阅者，一旦数据有变动，收到通知， 更新视图

第三步:Watcher 订阅者是 Observer 和 Compile 之间通信的桥梁，主要做的事情是:

1、在自身实例化时往属性订阅器(dep)里面添加自己

2、自身必须有一个 update()方法

3、待属性变动 dep.notice()通知时，能调用自身的 update()方法，并触发 Compile 中 绑定的回调，则功成身退。

第四步:MVVM 作为数据绑定的入口，整合 Observer、Compile 和 Watcher 三者，通过 Observer 来监听自己的 model 数据变化，通过 Compile 来解析编译模板指令，最终利用 Watcher 搭起 Observer 和 Compile 之间的通信桥梁，达到数据变化 -> 视图更新;视图 交互变化(input) -> 数据 model 变更的双向绑定效果。

25、你是怎么认识 vuex 的?

vuex 可以理解为一种开发模式或框架。比如 PHP 有 thinkphp，java 有 spring 等。

通过状态（数据源）集中管理驱动组件的变化（好比 spring 的 IOC 容器对 bean 进行集中管理）。

应用级的状态集中放在 store 中； 改变状态的方式是提交 mutations，这是个同步的事物； 异步逻辑应该封装在 action 中。

26、请说下封装 vue 组件的过程?

答:首先，组件可以提升整个项目的开发效率。能够把页面抽象成多个相对独立的模块，解决 了我们传统项目开发:**效率低、难维护、复用性**等问题。 然后，使用 Vue.extend 方法创建一个组件，然后使用 Vue.component 方法注册组件。子 组件需要数据，可以在 props 中接受定义。而子组件修改好数据后，想把数据传递给父组 件。可以采用 emit方法。

27、vue-loader 是什么?使用它的用途有哪些?

答:解析.vue 文件的一个加载器，跟 template/js/style 转换成 js 模块。用途:js 可以写 es6、style 样式可以 scss 或 less、template 可以加 jade 等

28、请说出 vue.cli 项目中 src 目录每个文件夹和文件的用法?

答 :assets 文件夹是放静态资源;components 是放组件;router 是定义路由相关的配置;view 视图;app.vue 是一个应用主组件;main.js 是入口文件

八、微信公众号/小程序

1、简单描述下微信小程序的相关文件类型?

答:微信小程序项目结构主要有四个文件类型,如下

1. `.json` 后缀的 `JSON` 配置文件
2. `.wxml` 后缀的 `WXML` 模板文件
3. `.wxss` 后缀的 `WXSS` 样式文件
4. `.js` 后缀的 `JS` 脚本逻辑文件

2、你使用过哪些方法，来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

九、Linux

1、绝对路径用什么符号表示？当前目录、上层目录用什么表示？主目录用什么表示? 切换目录用什么命令?

答案： 绝对路径： 如/etc/init.d 当前目录和上层目录： ./ ../ 主目录： ~/ 切换目录： cd

2、怎么查看当前进程？怎么执行退出？怎么查看当前路径?

答案： 查看当前进程： ps 执行退出： exit 查看当前路径： pwd

3、怎么清屏？怎么退出当前命令？怎么执行睡眠？怎么查看当前用户 id？查看指定帮助用什么命令?

答案： 清屏： clear 退出当前命令： ctrl+c 彻底退出 执行睡眠： ctrl+z 挂起当前进程fg 恢复后台 查看当前用户 id： ”id“：查看显示目前登陆账户的 uid 和 gid 及所属分组及用户名 查看指定帮助： 如 man adduser 这个很全 而且有例子； adduser --help 这个告诉你一些常用参数； info adduesr；

4、Ls 命令执行什么功能？ 可以带哪些参数，有什么区别?

答案： ls 执行的功能： 列出指定目录中的目录，以及文件 哪些参数以及区别： a 所有文件l 详细信息，包括大小字节数，可读可写可执行的权限等

5、建立软链接(快捷方式)，以及硬链接的命令。

答案： 软链接： ln -s slink source 硬链接： ln link source

6、目录创建用什么命令？创建文件用什么命令？复制文件用什么命令?

答案： 创建目录： mkdir 创建文件：典型的如 touch，vi 也可以创建文件，其实只要向一个不存在的文件输出，都会创建文件 复制文件： cp

7. 文件权限修改用什么命令？格式是怎么样的?

文件权限修改： chmod 格式如下：

> $ chmod u+x file 给 file 的属主增加执行权限 $ chmod 751 file 给 file 的属主分配读、写、执行(7)的权限，给 file 的所在组分配读、执行(5)的权限，给其他用户分配执行(1)的权限 $ chmod u=rwx,g=rx,o=x file 上例的另一种形式 $ chmod =r file 为所有用户分配读权限 $ chmod 444 file 同上例 $ chmod a-wx,a+r file同上例 $ chmod -R u+r directory 递归地给 directory 目录下所有文件和子目录的属主分配读的权限

8、查看文件内容有哪些命令可以使用?

答案： vi 文件名 #编辑方式查看，可修改 cat 文件名 #显示全部文件内容 more 文件名 #分页显示文件内容 less 文件名 #与 more 相似，更好的是可以往前翻页 tail 文件名 #仅查看尾部，还可以指定行数 head 文件名 #仅查看头部,还可以指定行数

9、随意写文件命令？怎么向屏幕输出带空格的字符串，比如”hello world”?

答案:

写文件命令：vi

向屏幕输出带空格的字符串:echo hello world

10、终端是哪个文件夹下的哪个文件？黑洞文件是哪个文件夹下的哪个命令?

答案： 终端 /dev/tty

黑洞文件 /dev/null

11、移动文件用哪个命令？改名用哪个命令?

答案： mv mv

12、复制文件用哪个命令？如果需要连同文件夹一块复制呢？如果需要有提示功能呢?

答案： cp cp -r ？？？？

13、删除文件用哪个命令？如果需要连目录及目录下文件一块删除呢？删除空文件夹用什么命令?

答案： rm rm -r rmdir

14、Linux 下命令有哪几种可使用的通配符？分别代表什么含义?

答案： “？”可替代单个字符。

“*”可替代任意多个字符。

方括号“[charset]”可替代 charset 集中的任何单个字符，如[a-z]，[abABC]

15、用什么命令对一个文件的内容进行统计？(行号、单词数、字节数)

答案:

wc 命令 - c 统计字节数 - l 统计行数 - w 统计字数。

16、Grep 命令有什么用？ 如何忽略大小写？ 如何查找不含该串的行?

答案： 是一种强大的文本搜索工具，它能使用正则表达式搜索文本，并把匹 配的行打印出来。 grep [stringSTRING] filename grep [^string] filename

17、Linux 中进程有哪几种状态？在 ps 显示出来的信息中，分别用什么符号表示的？

答案： （1）、不可中断状态：进程处于睡眠状态，但是此刻进程是不可中断的。不可中断， 指进程不响应异步信号。 （2）、暂停状态/跟踪状态：向进程发送一个 SIGSTOP 信号，它就会因响应该信号 而进入 TASK_STOPPED 状态;当进程正在被跟踪时，它处于 TASK_TRACED 这个特殊的状态。 “正在被跟踪”指的是进程暂停下来，等待跟踪它的进程对它进行操作。

（3）、就绪状态：在 run_queue 队列里的状态

（4）、运行状态：在 run_queue 队列里的状态 （5）、可中断睡眠状态：处于这个状态的进程因为等待某某事件的发生（比如等待 socket 连接、等待信号量），而被挂起 （6）、zombie 状态（僵尸）：父亲没有通过 wait 系列的系统调用会顺便将子进程的尸体（task_struct）也释放掉 （7）、退出状态

> D 不可中断 Uninterruptible（usually IO） R 正在运行，或在队列中的进程 S 处于休眠状态 T 停止或被追踪 Z 僵尸进程 W 进入内存交换（从内核 2.6 开始无效） X 死掉的进程

18、怎么使一个命令在后台运行?

答案： 一般都是使用 & 在命令结尾来让程序自动运行。(命令后可以不追加空格)

19、利用 ps 怎么显示所有的进程? 怎么利用 ps 查看指定进程的信息?

答案： ps -ef (system v 输出)

ps -aux bsd 格式输出

ps -ef | grep pid

20、哪个命令专门用来查看后台任务?

答案：

job -l

21、把后台任务调到前台执行使用什么命令?把停下的后台任务在后台执行起来用什么命令?

答案： 把后台任务调到前台执行 fg

把停下的后台任务在后台执行起来 bg

22、终止进程用什么命令? 带什么参数?

答案:

kill [-s <信息名称或编号>][程序] 或 kill [-l <信息编号>]

kill-9 pid

23、怎么查看系统支持的所有信号?

答案:

kill -l

24、搜索文件用什么命令? 格式是怎么样的?

答案:

find <指定目录> <指定条件> <指定动作>

whereis 加参数与文件名

locate 只加文件名

find 直接搜索磁盘，较慢。

find / -name "string*"

25、查看当前谁在使用该主机用什么命令? 查找自己所在的终端信息用什么命令?

答案: 查找自己所在的终端信息：who am i

查看当前谁在使用该主机：who

26、使用什么命令查看用过的命令列表?

答案:

history

27、使用什么命令查看磁盘使用空间? 空闲空间呢?

答案:

df -hl 文件系统 容量 已用 可用 已用% 挂载点 Filesystem Size Used Avail Use% Mounted on /dev/hda2 45G 19G 24G 44% / /dev/hda1 494M 19M 450M 4% /boot

28、使用什么命令查看网络是否连通?

答案： netstat

29、使用什么命令查看 ip 地址及接口信息?

答案:

ifconfig

30、查看各类环境变量用什么命令?

答案:

查看所有 env 查看某个，如 home： env $HOME

31、通过什么命令指定命令提示符?

答案:

> \u：显示当前用户账号
>
> \h：显示当前主机名
>
> \W：只显示当前路径最后一个目录
>
> \w：显示当前绝对路径（当前用户目录会以~代替）
>
> $PWD：显示当前全路径
>
> $：显示命令行'$'或者'#'符号
>
> #：下达的第几个命令
>
> \d：代表日期，格式为week day month date，例如："MonAug1"
>
> \t：显示时间为24小时格式，如：HH：MM：SS
>
> \T：显示时间为12小时格式
>
> \A：显示时间为24小时格式：HH：MM
>
> \v：BASH的版本信息 如export PS1='[\u@\h\w#]$'

32、查找命令的可执行文件是去哪查找的? 怎么对其进行设置及添加?

答案:

whereis [-bfmsu][-B <目录>...][-M <目录>...][-S <目录>...][文件...]

补充说明：whereis 指令会在特定目录中查找符合条件的文件。这些文件的烈性应属于原始代码，二进制文件，或是帮助文件。

> -b 只查找二进制文件。
>
> -B<目录> 只在设置的目录下查找二进制文件。 -f 不显示文件名前的路径名称。 -m 只查找说明文件。 -M<目录> 只在设置的目录下查找说明文件。 -s 只查找原始代码文件。 -S<目录> 只在设置的目录下查找原始代码文件。 -u 查找不包含指定类型的文件。 which 指令会在 PATH 变量指定的路径中，搜索某个系统命令的位置，并且返回第一个搜索结果。 -n 指定文件名长度，指定的长度必须大于或等于所有文件中最长的文件名。 -p 与-n 参数相同，但此处的包括了文件的路径。 -w 指定输出时栏位的宽度。 -V 显示版本信息

33、通过什么命令查找执行命令?

答案： which 只能查可执行文件

whereis 只能查二进制文件、说明文档，源文件等

34、怎么对命令进行取别名?

答案： alias la='ls -a'

35、du 和 df 的定义，以及区别?

答案：

du 显示目录或文件的大小

df 显示每个<文件>所在的文件系统的信息，默认是显示所有文件系统。 （文件系统分配其中的一些磁盘块用来记录它自身的一些数据，如 i 节点，磁盘分布图，间接块，超级块等。这些数据对大多数用户级的程序来说是不可见的，通常称为 Meta Data。） du 命令是用户级的程序，它不考虑 Meta Data，而 df 命令则查看文件系统的磁盘分配图并考虑 Meta Data。 df 命令获得真正的文件系统数据，而 du 命令只查看文件系统的部分情况。

36、awk 详解。

答案：

> awk '{pattern + action}' {filenames} #cat /etc/passwd |awk -F ':' '{print 1 ” \t ”7}' //-F 的意思是以':'分隔 root /bin/bash daemon /bin/sh 搜索/etc/passwd 有 root 关键字的所有行

> #awk -F: '/root/' /etc/passwd root ✕ 0:0:root:/root:/bin/bash

37、当你需要给命令绑定一个宏或者按键的时候，应该怎么做呢?

答案：

可以使用bind命令，bind可以很方便地在shell中实现宏或按键的绑定。

在进行按键绑定的时候，我们需要先获取到绑定按键对应的字符序列。

比如获取F12的字符序列获取方法如下：先按下Ctrl+V,然后按下F12 .我们就可以得到F12的字符序列 ^[[24~。

接着使用bind进行绑定。

[root@localhost ~]# bind '"\e[24~":"date"'

注意：相同的按键在不同的终端或终端模拟器下可能会产生不同的字符序列。

【附】也可以使用showkey -a命令查看按键对应的字符序列。

38、如果一个linux新手想要知道当前系统支持的所有命令的列表，他需要怎么做?

答案:

使用命令compgen -c，可以打印出所有支持的命令列表。

> [root@localhost ~]$ compgen -c
>
> l.
>
> ll
>
> ls
>
> which
>
> if
>
> then
>
> else
>
> elif
>
> fi
>
> case
>
> esac
>
> for
>
> select
>
> while
>
> until
>
> do
>
> done
>
> ...

39、如果你的助手想要打印出当前的目录栈，你会建议他怎么做?

答案:

使用Linux 命令dirs可以将当前的目录栈打印出来。

> [root@localhost ~]# dirs
>
> /usr/share/X11

【附】：目录栈通过pushd popd 来操作。

40、你的系统目前有许多正在运行的任务，在不重启机器的条件下，有什么方法可以把所有正在运行的进程移除呢?

答案:

使用linux命令 'disown -r '可以将所有正在运行的进程移除。

41、bash shell 中的hash 命令有什么作用?

答案:

linux命令'hash'管理着一个内置的哈希表，记录了已执行过的命令的完整路径, 用该命令可以打印出你所使用过的命令以及执行的次数。

> [root@localhost ~]# hash
>
> hits command
>
> 2 /bin/ls
>
> 2 /bin/su

42、哪一个bash内置命令能够进行数学运算。

答案:

bash shell 的内置命令let 可以进行整型数的数学运算。

> #! /bin/bash ... ... let c=a+b ... ...

43、怎样一页一页地查看一个大文件的内容呢?

答案:

通过管道将命令”cat file_name.txt” 和 ’more’ 连接在一起可以实现这个需要.

[root@localhost ~]# cat file_name.txt | more

44、数据字典属于哪一个用户的?

答案:

数据字典是属于’SYS’用户的，用户‘SYS’ 和 ’SYSEM’是由系统默认自动创建的

45、怎样查看一个linux命令的概要与用法？假设你在/bin目录中偶然看到一个你从没见过的的命令，怎样才能知道它的作用和用法呢?

答案:

使用命令whatis 可以先出显示出这个命令的用法简要，比如，你可以使用whatis zcat 去查看‘zcat’的介绍以及使用简要。

> [root@localhost ~]# whatis zcat
>
> zcat [gzip] (1) – compress or expand files

46、使用哪一个命令可以查看自己文件系统的磁盘空间配额呢?

答案:

使用命令repquota 能够显示出一个文件系统的配额信息

【附】只有root用户才能够查看其它用户的配额。

十、MYSQL

1、理解主键和外键?

主 键:

数据库表中对储存数据对象予以唯一和完整标识的数据列或属性的组合。**一个数据列只能有一个主键**，且主键的取值不能缺失，即不能为空值（Null）。

外 键:

在一个表中存在的**另一个表的主键**称此表的外键。

2、数据库事务的四个特性及含义

数据库事务transanction正确执行的四个基本要素。ACID,原子性(Atomicity)、一致性(Correspondence)、隔离性(Isolation)、持久性(Durability)。 **原子性**:整个事务中的所有操作，要么全部完成，要么全部不完成，不可能停滞在中间某个环节。事务在执行过程中发生错误，会被回滚（Rollback）到事务开始前的状态，就像这个事务从来没有执行过一样。 **一致性**:在事务开始之前和事务结束以后，数据库的完整性约束没有被破坏。 **隔离性**:隔离状态执行事务，使它们好像是系统在给定时间内执行的唯一操作。如果有两个事务，运行在相同的时间内，执行 相同的功能，事务的隔离性将确保每一事务在系统中认为只有该事务在使用系统。这种属性有时称为串行化，为了防止事务操作间的混淆，必须串行化或序列化请 求，使得在同一时间仅有一个请求用于同一数据。 **持久性**:在事务完成以后，该事务所对数据库所作的更改便持久的保存在数据库之中，并不会被回滚。

3、drop,delete与truncate的区别

drop直接删掉表。truncate删除表中数据，再插入时自增长id又从1开始 。delete删除表中数据，可以加where字句。

4、索引的工作原理

数据库索引，是数据库管理系统中一个排序的数据结构，以协助快速查询、更新数据库表中数据。**索引的实现通常使用B树及其变种B+树。**

在数据之外，数据库系统还维护着满足特定查找算法的数据结构，这些数据结构以某种方式引用（指向）数据，这样就可以在这些数据结构上实现高级查找算法。这种数据结构，就是索引。

为表设置索引要付出代价的：一是增加了数据库的存储空间，二是在插入和修改数据时要花费较多的时间(因为索引也要随之变动)。

5、索引的优点

第一，通过创建唯一性索引，可以保证数据库表中每一行数据的唯一性。

第二，可以大大加快数据的检索速度，这也是创建索引的最主要的原因。

第三，可以加速表和表之间的连接，特别是在实现数据的参考完整性方面特别有意义。

第四，在使用分组和排序子句进行数据检索时，同样可以显著减少查询中分组和排序的时间。

第五，通过使用索引，可以在查询的过程中，使用优化隐藏器，提高系统的性能。

6、索引的缺点

第一，创建索引和维护索引要耗费时间，这种时间随着数据量的增加而增加。

第二，索引需要占物理空间，除了数据表占数据空间之外，每一个索引还要占一定的物理空间，如果要建立聚簇索引，那么需要的空间就会更大。

第三，当对表中的数据进行增加、删除和修改的时候，索引也要动态的维护，这样就降低了数据的维护速度。

7、什么样场合下不建议创建索引?

第一，对于那些在查询中很少使用或者参考的列不应该创建索引。这是因为，既然这些列很少使用到，因此有索引或者无索引，并不能提高查询速度。相反，由于增加了索引，反而降低了系统的维护速度和增大了空间需求。

第二，对于那些只有很少数据值的列也不应该增加索引。这是因为，由于这些列的取值很少，例如人事表的性别列，在查询的结果中，结果集的数据行占了表中数据行的很大比例，即需要在表中搜索的数据行的比例很大。增加索引，并不能明显加快检索速度。

第三，对于那些定义为text, image和bit数据类型的列不应该增加索引。这是因为，这些列的数据量要么相当大，要么取值很少。

第四，当修改性能远远大于检索性能时，不应该创建索引。这是因为，**修改性能和检索性能是互相矛盾的**。当增加索引时，会提高检索性能，但是会降低修改性能。当减少索引时，会提高修改性能，降低检索性能。因此，当修改性能远远大于检索性能时，不应该创建索引。

8、数据库范式

1 第一范式（1NF） 在任何一个关系数据库中，第一范式（1NF）是对关系模式的基本要求，不满足第一范式（1NF）的数据库就不是关系数据库。 所谓第一范式（1NF）是指数据库表的每一列都是不可分割的基本数据项，同一列中不能有多个值，即实体中的某个属性不能有多个值或者不能有重复的属性。如果出现重复的属性，就可能需要定义一个新的实体，新的实体由重复的属性构成，新实体与原实体之间为一对多关系。在第一范式（1NF）中表的每一行只包含一个实例的信息。简而言之，**第一范式就是无重复的列。**

2 第二范式（2NF） 第二范式（2NF）是在第一范式（1NF）的基础上建立起来的，即满足第二范式（2NF）必须先满足第一范式（1NF）。第二范式（2NF）要求数据库表中的每个实例或行必须可以被惟一地区分。为实现区分通常需要为表加上一个列，以存储各个实例的惟一标识。这个惟一属性列被称为主关键字或主键、主码。 第二范式（2NF）要求实体的属性完全依赖于主关键字。所谓完全依赖是指不能存在仅依赖主关键字一部分的属性，如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。为实现区分通常需要为表加上一个列，以存储各个实例的惟一标识。简而言之，**第二范式就是非主属性非部分依赖于主关键字。**

3 第三范式（3NF） 满足第三范式（3NF）必须先满足第二范式（2NF）。简而言之，第三范式（3NF）要求一个数据库表中不包含已在其它表中已包含的非主关键字信息。例如，存在一个部门信息表，其中每个部门有部门编号（dept_id）、部门名称、部门简介等信息。那么在员工信息表中列出部门编号后就不能再将部门名称、部门简介等与部门有关的信息再加入员工信息表中。如果不存在部门信息表，则根据第三范式（3NF）也应该构建它，否则就会有大量的数据冗余。简而言之，**第三范式就是属性不依赖于其它非主属性。（我的理解是消除冗余）**

9、MySQL中myisam与innodb的区别

1.InnoDB支持事物，而MyISAM不支持事物 2.InnoDB支持行级锁，而MyISAM支持表级锁 3.InnoDB支持MVCC，而MyISAM不支持 4.InnoDB支持外键，而MyISAM不支持 5.InnoDB不支持全文索引，而MyISAM支持。 6.InnoDB不能通过直接拷贝表文件的方法拷贝表到另外一台机器， myisam 支持 7.InnoDB表支持多种行格式， myisam 不支持 8.InnoDB是索引组织表， myisam 是堆表

10、MySQL中varchar与char的区别以及varchar(50)中的50代表的涵义

(1)、varchar与char的区别

在单字节字符集下， char (N) 在内部存储的时候总是定长， 而且没有变长字段长度列表中。 在多字节字符集下面， char(N)如果存储的字节数超过 N，那么 char (N) 将和 varchar (N) 没有区别。在多字节字符集下面，如果存
储的字节数少于 N，那么存储 N 个字节，后面补空格，补到 N 字节长度。 都存储变长的数据和变长字段长度列表。
varchar(N)无论是什么字节字符集，都是变长的，即都存储变长数据和变长字段长度列表。

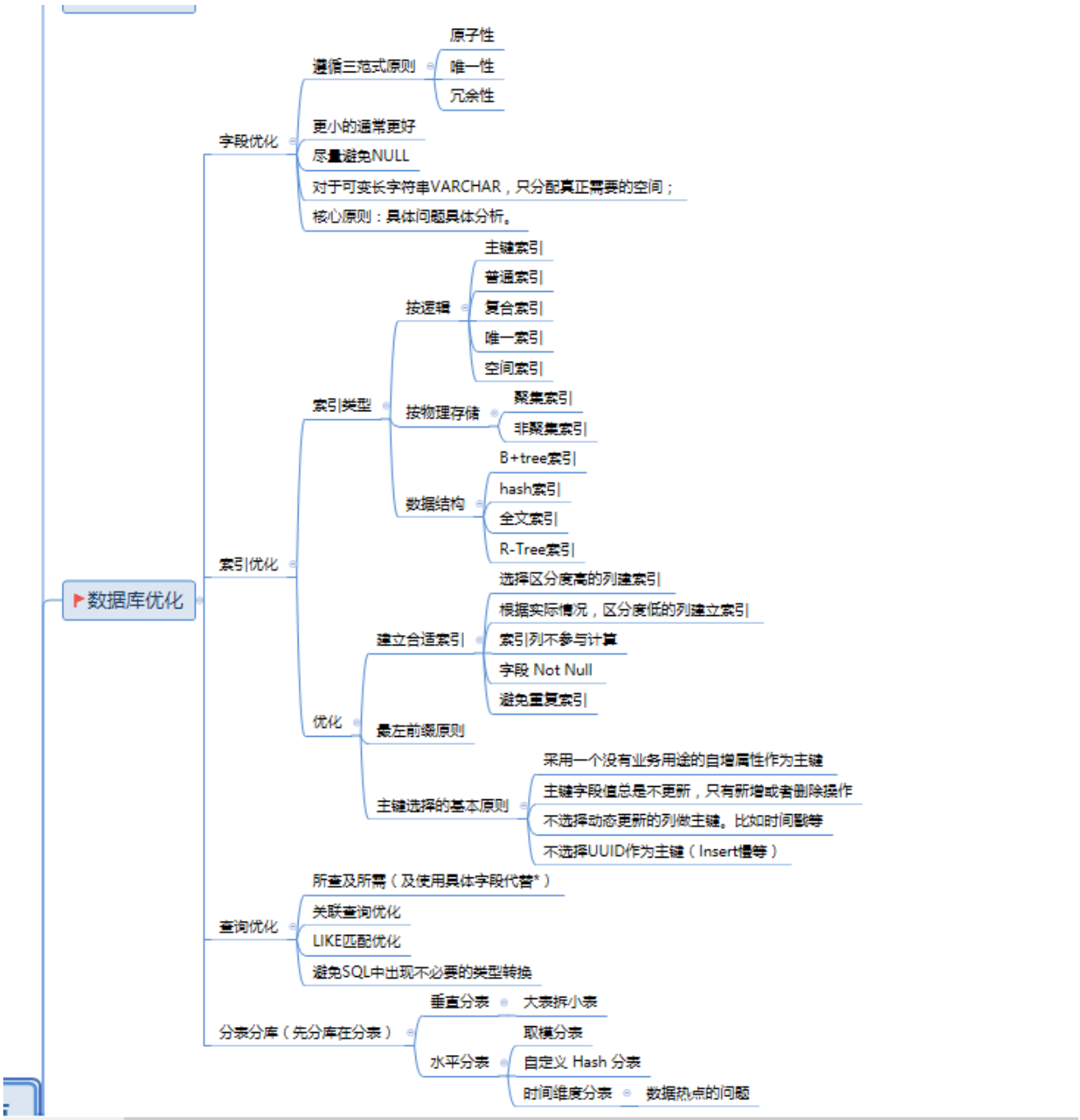
(2)、varchar(50)中50的涵义

最多存放50个字符，varchar(50)和(200)存储hello所占空间一样，但后者在排序时会消耗更多内存，因为order by col采用fixed_length计算col长度(memory引擎也一样)。在早期 MySQL 版本中， 50 代表字节数，现在代表字符数。

11、表中有大字段X(例如：text类型)，且字段X不会经常更新，以读为为主，请问您是选择拆成子表，还是继续放一起?写出您这样选择的理由

拆带来的问题：连接消耗 + 存储拆分空间；
不拆可能带来的问题：查询性能；
如果能容忍拆分带来的空间问题,拆的话最好和经常要查询的表的主键在物理结构上放置在一起(分区) 顺序IO,减少连接消耗,最后这是一个文本列再加上一个全文索引来尽量抵消连接消耗
如果能容忍不拆分带来的查询性能损失的话:上面的方案在某个极致条件下肯定会出现问题,那么不拆就是最好的选择

12、数据库优化的思路



13、什么情况下设置了索引但无法使用

① 以“%”开头的LIKE语句，模糊匹配

② OR语句前后没有同时使用索引

③ 数据类型出现隐式转化（如varchar不加单引号的话可能会自动转换为int型）

14、数据库中的事务是什么?

事务（transaction）是作为一个单元的一组有序的数据库操作。如果组中的所有操作都成功，则认为事务成功，即使只有一个操作失败，事务也不成功。如果所有操作完成，事务则提交，其修改将作用于所有其他数据库进程。如果一个操作失败，则事务将回滚，该事务所有操作的影响都将取消。ACID 四大特性,原子性、隔离性、一致性、持久性。

（1）原子性：即不可分割性，事务要么全部被执行，要么就全部不被执行。

（2）一致性或可串性。事务的执行使得数据库从一种正确状态转换成另一种正确状态

（3）隔离性。在事务正确提交之前，不允许把该事务对数据的任何改变提供给任何其他事务，

（4） 持久性。事务正确提交后，其结果将永久保存在数据库中，即使在事务提交后有了其他故障，事务的处理结果也会得到保存。

15、22.SQL注入漏洞产生的原因？如何防止?

SQL注入产生的原因：程序开发过程中不注意规范书写sql语句和对特殊字符进行过滤，导致客户端可以通过全局变量POST和GET提交一些sql语句正常执行。

防止SQL注入的方式：

开启配置文件中的magic_quotes_gpc 和 magic_quotes_runtime设置

执行sql语句时使用addslashes进行sql语句转换

Sql语句书写尽量不要省略双引号和单引号。

过滤掉sql语句中的一些关键词：update、insert、delete、select、 * 。

提高数据库表和字段的命名技巧，对一些重要的字段根据程序的特点命名，取不易被猜到的。

Php配置文件中设置register_globals为off,关闭全局变量注册

控制错误信息，不要在浏览器上输出错误信息，将错误信息写到日志文件中。

16、 说说对SQL语句优化有哪些方法？（选择几条）

（1）Where子句中：where表之间的连接必须写在其他Where条件之前，那些可以过滤掉最大数量记录的条件必须写在Where子句的末尾.HAVING最后。

（2）用EXISTS替代IN、用NOT EXISTS替代NOT IN。

（3） 避免在索引列上使用计算

（4）避免在索引列上使用IS NULL和IS NOT NULL

（5）对查询进行优化，应尽量避免全表扫描，首先应考虑在 where 及 order by 涉及的列上建立索引。

（6）应尽量避免在 where 子句中对字段进行 null 值判断，否则将导致引擎放弃使用索引而进行全表扫描

（7）应尽量避免在 where 子句中对字段进行表达式操作，这将导致引擎放弃使用索引而进行全表扫描

17、 char和varchar的区别?

是一种固定长度的类型，varchar则是一种可变长度的类型，它们的区别是：

char(M)类型的数据列里，每个值都占用M个字节，如果某个长度小于M，MySQL就会在它的右边用空格字符补足．（在检索操作中那些填补出来的空格字符将被去掉）在varchar(M)类型的数据列里，每个值只占用刚好够用的字节再加上一个用来记录其长度的字节（即总长度为L+1字节）．

varchar得适用场景:

字符串列得最大长度比平均长度大很多 2.字符串很少被更新，容易产生存储碎片 3.使用多字节字符集存储字符串

Char得场景:

> 存储具有近似得长度（md5值,身份证，手机号）,长度比较短小得字符串（因为varchar需要额外空间记录字符串长度），更适合经常更新得字符串，更新时不会出现页分裂得情况，避免出现存储碎片，获得更好的io性能

18、MySQL数据库作发布系统的存储，一天五万条以上的增量，怎么优化?

a. 设计良好的数据库结构，允许部分数据冗余，尽量避免join查询，提高效率。

b.业务允许的情况下，也就是说不需要事物机制，建议用myisam引擎，相比较而言，myisam比innodb的批量插入要快很多

c. mysql库主从读写分离。

d. 找规律分表，减少单表中的数据量提高查询速度。

e。添加缓存机制，比如memcached，apc等。

f. 不经常改动的页面，生成静态页面。

g. 书写高效率的SQL。比如 SELECT * FROM TABEL 改为 SELECT field_1, field_2, field_3 FROM TABLE.

19. 对于大流量的网站,您采用什么样的方法来解决各页面访问量统计问题?

答：a. 确认服务器是否能支撑当前访问量。

b. 优化数据库访问。

c. 禁止外部访问链接（盗链）, 比如图片盗链。

d. 控制文件下载。

e. 使用不同主机分流。

f. 使用浏览统计软件，了解访问量，有针对性的进行优化。

20、什么是队列？排它锁，Myisam 死锁如何解决?

答:在默认情况下 MYisam 是表级锁，所以同时操作单张表的多个动作只能以队列的方式进行；排它锁又名写锁，在 SQL 执行过程中为排除其它请求而写锁，在执行完毕后会自动释放；

死锁解决：先找到死锁的线程号，然后杀掉线程 ID。

十一、NOSQL数据库

1、Redis、Memcache与MongoDB的区别

从以下几个维度，对redis、memcache、mongoDB 做了对比，

1、性能

都比较高，性能对我们来说应该都不是瓶颈

总体来讲，TPS方面redis和memcache差不多，要大于mongodb

2、操作的便利性

memcache数据结构单一(key-value)

redis丰富一些(string、hash、zset、set、list)，数据操作方面，redis更好一些，较少的网络IO次数

mongodb支持丰富的数据表达，索引，最类似关系型数据库，支持的查询语言非常丰富

3、内存空间的大小和数据量的大小

redis在2.0版本后增加了自己的VM特性，突破物理内存的限制；可以对key value设置过期时间（类似memcache）

memcache可以修改最大可用内存,采用LRU算法

mongoDB适合大数据量的存储，依赖操作系统VM做内存管理，吃内存也比较厉害，服务不要和别的服务在一起

4、可用性（单点问题）

对于单点问题，

redis，依赖客户端来实现分布式读写；主从复制时，每次从节点重新连接主节点都要依赖整个快照,无增量复制，因性能和效率问题，

所以单点问题比较复杂；不支持自动sharding,需要依赖程序设定一致hash 机制。

一种替代方案是，不用redis本身的复制机制，采用自己做主动复制（多份存储），或者改成增量复制的方式（需要自己实现），一致性问题和性能的权衡

Memcache本身没有数据冗余机制，也没必要；对于故障预防，采用依赖成熟的hash或者环状的算法，解决单点故障引起的抖动问题。

mongoDB支持master-slave,replicaset（内部采用paxos选举算法，自动故障恢复）,auto sharding机制，对客户端屏蔽了故障转移和切分机制。

5、可靠性（持久化）

对于数据持久化和数据恢复，

redis支持（快照、AOF）：依赖快照进行持久化，aof增强了可靠性的同时，对性能有所影响

memcache不支持，通常用在做缓存,提升性能；

MongoDB从1.8版本开始采用binlog方式支持持久化的可靠性

6、数据一致性（事务支持）

Memcache 在并发场景下，用cas保证一致性

redis事务支持比较弱，只能保证事务中的每个操作连续执行

mongoDB不支持事务

7、数据分析

mongoDB内置了数据分析的功能(mapreduce),其他不支持

8、应用场景

redis：数据量较小的更性能操作和运算上

memcache：用于在动态系统中减少数据库负载，提升性能;做缓存，提高性能（适合读多写少，对于数据量比较大，可以采用sharding）

MongoDB:主要解决海量数据的访问效率问题

2.mongodb持久化原理

mongodb与mysql不同，mysql的每一次更新操作都会直接写入硬盘，但是mongo不会，做为内存型数据库，数据操作会先写入内存，然后再会持久化到硬盘中去，那么mongo是如何持久化的呢 mongodb在启动时，专门初始化一个线程不断循环（除非应用crash掉），用于在一定时间周期内来从defer队列中获取要持久化的数据并写入到磁盘的journal(日志)和mongofile(数据)处，当然因为它不是在用户添加记录时就写到磁盘上，所以按mongodb开发者说，它不会造成性能上的损耗，因为看过代码发现，当进行CUD操作时，记录(Record类型)都被放入到defer队列中以供延时批量（groupcommit）提交写入，但相信其中时间周期参数是个要认真考量的参数，系统为90毫秒，如果该值更低的话，可能会造成频繁磁盘操作，过高又会造成系统宕机时数据丢失过。

3、什么是NoSQL数据库？NoSQL和RDBMS有什么区别？在哪些情况下使用和不使用NoSQL数据库?

NoSQL是非关系型数据库，NoSQL = Not Only SQL。 关系型数据库采用的结构化的数据，NoSQL采用的是键值对的方式存储数据。 在处理非结构化/半结构化的大数据时；在水平方向上进行扩展时；随时应对动态增加的数据项时可以优先考虑使用NoSQL数据库。 在考虑数据库的成熟度；支持；分析和商业智能；管理及专业性等问题时，应优先考虑关系型数据库。

4、MongoDB的特点是什么?

（1）面向文档（2）高性能（3）高可用（4）易扩展（5）丰富的查询语言

5、什么是缓存穿透?

一般的缓存系统，都是按照key去缓存查询，如果不存在对应的value，就应该去后端系统查找（比如DB）。如果key对应的value是一定不存在的，并且对该key并发请求量很大，就会对后端系统造成很大的压力。这就叫做缓存穿透。

6、如何避免缓存穿透?

1：对查询结果为空的情况也进行缓存，缓存时间设置短一点，或者该key对应的数据insert了之后清理缓存。

2：对一定不存在的key进行过滤。可以把所有的可能存在的key放到一个大的Bitmap中，查询时通过该bitmap过滤。

7、什么是缓存雪崩?

当缓存服务器重启或者大量缓存集中在某一个时间段失效，这样在失效的时候，也会给后端系统(比如DB)带来很大压力。

8、如何避免缓存雪崩?

1：在缓存失效后，通过加锁或者队列来控制读数据库写缓存的线程数量。比如对某个key只允许一个线程查询数据和写缓存，其他线程等待。

2：不同的key，设置不同的过期时间，让缓存失效的时间点尽量均匀。

3：做二级缓存，A1为原始缓存，A2为拷贝缓存，A1失效时，可以访问A2，A1缓存失效时间设置为短期，A2设置为长期

9、缓存数据的淘汰

缓存淘汰的策略有两种：(1) 定时去清理过期的缓存。 （2）当有用户请求过来时，再判断这个请求所用到的缓存是否过期，过期的话就去底层系统得到新数据并更新缓存。

两者各有优劣，第一种的缺点是维护大量缓存的key是比较麻烦的，第二种的缺点就是每次用户请求过来都要判断缓存失效，逻辑相对比较复杂，具体用哪种方案，大家可以根据自己的应用场景来权衡。

10、memcache 缓存什么数据

一、经常被读取并且实时性要求不强可以等到自动过期的数据。例如网站首页最新文章列表、某某排行等数据。

二、经常被读取并且实时性要求强的数据。比如用户的好友列表，用户文章列表，用户阅读记录等。

三、统计类缓存，比如文章浏览数、网站 PV 等。

四、活跃用户的基本信息或者某篇热门文章。

五、session 数据

11、Redis 如何防止高并发

其实 redis 是不会存在并发问题的，因为他是单进程的，再多的 command 都是 one by one 执行的。我们使用的时候，可能会出现并发问题，比如 get 和 set 这一对。redis 为什么会有高并发问题redis 的出身决定 Redis 是一种单线程机制的 nosql 数据库，基于 key-value，数据可持久化落盘。由于单线程所以 redis 本身并没有锁的概念，多个客户端连接并不存在竞争关系，但是利用 jedis 等客户端对 redis 进行并发访问时会出现问题。发生连接超时、数据转换错误、阻塞、客户端关闭连接等问题，这些问题均是由于客户端连接混乱造成。 同时，单线程的天性决定，高并发对同一个键的操作会排队处理，如果并发量很大，可能造成后来的请求超时。 在远程访问 redis 的时

候，因为网络等原因造成高并发访问延迟返回的问题。解决办法 在客户端将连接进行池化，同时对客户端读写 Redis 操作采用内部锁 synchronized。 服务器角度，利用 setnx 变向实现锁机制。

12、redis 消息队列先进先出需要注意什么

答:通常使用一个 list 来实现队列操作，这样有一个小限制，所以的任务统一都是先进先出，如果想优先处理某个任务就不太好处理了，这就需要让队列有优先级的概念，我们就可以优先处理高级别的任务，实现方式有以下几种方式：

1）单一列表实现：队列正常的操作是 左进右出（lpush,rpop）为了先处理高优先级任务，在遇到高级别任务时，可以直接插队，直接放入队列头部（rpush），这样，从队列头部（右侧）获取任务时，取到的就是高优先级的任务（rpop）

2）使用两个队列，一个普通队列，一个高级队列，针对任务的级别放入不同的队列，获取任务时也很简单，redis 的 BRPOP 命令可以按顺序从多个队列中取值，BRPOP 会按照给出的 key 顺序查看，并在找到的第一个非空 list 的尾部弹出一个元素，redis> BRPOP list1 list2 0list1 做为高优先级任务队列list2 做为普通任务队列这样就实现了先处理高优先级任务，当没有高优先级任务时，就去获取普通任务方式 1 最简单，但实际应用比较局限，方式 3 可以实现复杂优先级，但实现比较复杂，不利于维护方式 2 是推荐用法，实际应用最为合适

十二、排序算法

1. 冒泡排序

思路分析：在要排序的一组数中，对当前还未排好的序列，从前往后对相邻的两个数依次进行比较和调整，让较大的数往下沉，较小的往上冒。即，每当两相邻的数比较后发现它们的排序与排序要求相反时，就将它们互换。 **代码实现：**

```
$arr=array(1,43,54,62,21,66,32,78,36,76,39);
function bubbleSort($arr)
{
  $len=count($arr);
  //该层循环控制 需要冒泡的轮数
  for($i=1;$i<$len;$i++)
  { //该层循环用来控制每轮 冒出一个数 需要比较的次数
    for($k=0;$k<$len-$i;$k++)
    {
       if($arr[$k]>$arr[$k+1])
        {
            $tmp=$arr[$k+1];
            $arr[$k+1]=$arr[$k];
            $arr[$k]=$tmp;
        }
    }
  }
  return $arr;
}
```

2. 选择排序

思路分析：在要排序的一组数中，选出最小的一个数与第一个位置的数交换。然后在剩下的数当中再找最小的与第二个位置的数交换，如此循环到倒数第二个数和最后一个数比较为止。 **代码实现：**

```
function selectSort($arr) {
//双重循环完成，外层控制轮数，内层控制比较次数
 $len=count($arr);
    for($i=0; $i<$len-1; $i++) {
        //先假设最小的值的位置
        $p = $i;
        for($j=$i+1; $j<$len; $j++) {
            //$arr[$p] 是当前已知的最小值
            if($arr[$p] > $arr[$j]) {
            //比较，发现更小的,记录下最小值的位置；并且在下次比较时采用已知的最小值进行比较。
                $p = $j;
            }
        }
        //已经确定了当前的最小值的位置，保存到$p中。如果发现最小值的位置与当前假设的位置$i不同，则位置互换即可。
        if($p != $i) {
            $tmp = $arr[$p];
            $arr[$p] = $arr[$i];
            $arr[$i] = $tmp;
        }
    }
    //返回最终结果
    return $arr;
}
```

3.插入排序

思路分析：在要排序的一组数中，假设前面的数已经是排好顺序的，现在要把第n个数插到前面的有序数中，使得这n个数也是排好顺序的。如此反复循环，直到全部排好顺序。 **代码实现：**

```
function insertSort($arr) {
    $len=count($arr);
    for($i=1, $i<$len; $i++) {
        $tmp = $arr[$i];
        //内层循环控制，比较并插入
        for($j=$i-1;$j>=0;$j--) {
            if($tmp < $arr[$j]) {
                //发现插入的元素要小，交换位置，将后边的元素与前面的元素互换
                $arr[$j+1] = $arr[$j];
                $arr[$j] = $tmp;
            } else {
                //如果碰到不需要移动的元素，由于是已经排序好是数组，则前面的就不需要再次比较了。
```

```
                break;
            }
        }
    }
    return $arr;
}
```

4.快速排序

思路分析：选择一个基准元素，通常选择第一个元素或者最后一个元素。通过一趟扫描，将待排序列分成两部分，一部分比基准元素小，一部分大于等于基准元素。此时基准元素在其排好序后的正确位置，然后再用同样的方法递归地排序划分的两部分。 **代码实现**：

```
function quickSort($arr) {
    //先判断是否需要继续进行
    $length = count($arr);
    if($length <= 1) {
        return $arr;
    }
    //选择第一个元素作为基准
    $base_num = $arr[0];
    //遍历除了标尺外的所有元素，按照大小关系放入两个数组内
    //初始化两个数组
    $left_array = array();  //小于基准的
    $right_array = array();  //大于基准的
    for($i=1; $i<$length; $i++) {
        if($base_num > $arr[$i]) {
            //放入左边数组
            $left_array[] = $arr[$i];
        } else {
            //放入右边
            $right_array[] = $arr[$i];
        }
    }
    //再分别对左边和右边的数组进行相同的排序处理方式递归调用这个函数
    $left_array = quick_sort($left_array);
    $right_array = quick_sort($right_array);
    //合并
    return array_merge($left_array, array($base_num), $right_array);
}
```

十三、开发实战问题

1、做秒杀时锁表的情况如何解决?

当时我们做秒杀时考虑了好几种方案，其中有一种就是使用事务加上排他锁来实现。

2、架构类的东西接触过吗?

有接触过，曾经自己在自己的服务器上配置过。我以前做过以下几个架构方面的配置和测试; 1、数据库的读写分离、主从复制及集群。 2、Nginx 负载均衡 3、redis 集群及主从

3、如何处理负载、高并发?

从低成本、高性能和高扩张性的角度来说有如下处理方案:

1、HTML 静态化其实大家都知道，效率最高、消耗最小的就是纯静态化的 html 页面，所以我们尽可能使我们的网站上的页面采用静态页面来实现，这个最简单的方法其实也是最有效的方法。

2、图片服务器分离把图片单独存储，尽量减少图片等大流量的开销，可以放在一些相关的平台上，如骑牛等

3、数据库集群和库表散列及缓存数据库的并发连接为 100，一台数据库远远不够，可以从读写分离、主从复制，数据库集群方面来着手。另外尽量减少数据库的访问，可以使用缓存数据库如 memcache、redis。

4、镜像：尽量减少下载，可以把不同的请求分发到多个镜像端。

5、数据库优化

6、负载均衡：Apache 的最大并发连接为 1500，只能增加服务器，可以从硬件上着手，如 F5 服务器。当然硬件的成本比较高，我们往往从软件方面着手。负载均衡 （Load Balancing） 建立在现有网络结构之上，它提供了一种廉价有效透明的方法扩展网络设备和服务器的带宽、增加吞吐量、加强网络数据处理能力，同时能够提高网络的灵活性和可用性。

4、怎么实现第三方登录?

答:第三方登陆主要是基于 author 协议来实现，下面简单说下实现流程: 1、首先我们需要以开发者的身份向第三方登陆平台申请接入应用，申请成功后，我们会获得一个 appID 和一个 secrectID. 2、当我们的网站需接入第三方登陆时，会引导用户跳转到第三方的登陆授权页面，此时把之前申请的 appID 和 secrectID 带给登陆授权页面。 3、用户登陆成功后即得到授权，第三方会返回一个临时的 code 给我们的网站。 4、我们的网站接受到 code 后，再次向我们的第三方发起请求，并携带接收的 code,从第三方获取 access_token. 5、第三方处理请求后，会返回一个 access_token 给我们的网站，我们的网站获取到 access_token 后就可以调用第三方提供的接口了，比如获取用户信息等。最后把该用户信息存入到我们站点的数据库，并把信息保存到 session 中，实现用户的第三方登陆。

5、用户免登陆

答:当用户第一次访问应用系统的时候，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，认证系统进行身份校验，如果通过校验，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 送到认证系统进行校验，检查 ticket 的合法性。如果通过校验，用户就可以在不用再次登录的情况下访问应用系统

6、缓存在什么时候应该失效

缓存的时长超过了缓存的失效时间就会自动失效，或者是数据库中的数据被更新后，缓存应该被手动清楚， 以防止读取到缓存中的脏数据。或者是缓存满了之后，会被缓存的回收机制自动删除

7、怎么保证促销商品不会超卖

答: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。 第一种方案是：①在每次下订单前我们判断促销商品的数量够不够，不够不允许下订单，更改库存量时加上一个条件，只更改商品库存大于 0 的商品的库存，当时我们使用 ab 进行压力测试，当并发超过 500，访问量超过 2000 时，还是会出现超卖现象。所以被我们否定了。 第二种方案是：②使用 mysql 的事务加排他锁来解决，首先我们选择数据库的存储引擎为 innoDB，使用的是排他锁实现的，刚开始的时候我们测试了下共享锁，发现还是会出现超卖的现象。有个问题是，当我们进行高并发测试时，对数据库的性能影响很大，导致数据库的压力很大，最终也被我们否定了。 第三种方案是：③使用文件锁实现。当用户抢到一件促销商品后先触发文件锁，防止其他用户进入，该用户抢到促销品后再解开文件锁，放其他用户进行操作。这样可以解决超卖的问题，但是会导致文件得 I/O 开销很大。 **最后我们使用了 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、购物车的原理

答:服务器通过追踪每个用户的行动，以保证在结账时每件商品都物有其主。 主要涉及以下几点: 1、把商品添加到购物车，即订购 2、删除购物车中已定购的商品 3、修改购物车中某一本图书的订购数量 4、清空购物车 5、显示购物车中商品清单及数量、价格 实现购物车的关键在于服务器识别每一个用户并维持与他们的联系。但是 HTTP 协议是一种"无状态(Stateless)"的协议，因而服务器不能记住是谁在购买商品，当把商品加入购物车时，服务器也不知道购物车里原先有些什么，使得用户在不同页面间跳转时购物车无法"随身携带"，这都给购物车的实现造成了一定的困难。 目前购物车的实现主要是通过 cookie、session 或结合数据库的方式。

虽然 cookie 可用来实现购物车，但必须获得浏览器的支持，再加上它是存储在客户端的信息，极易被获取，所以这也限制了它存储更多，更重要的信息。所以一般 cookie 只用来维持与服务器的会话，例如国内最大的当当网络书店就是用 cookie 保持与客户的联系，但是这种方式最大的缺点是如果客户端不支持 cookie 就会使购物车失效。Session 能很好地与交易双方保持会话，可以忽视客户端的设置。在购物车技术中得到了广泛的应用。但 session 的文件属性使其仍然留有安全隐患。结合数据库的方式虽然在一定程度上解决了上述的问题，但从上面的例子可以看出：在这种购物流程中涉及到对数据库表的频繁操作，尤其是用户每选购一次商品，都要与数据库进行连接，当用户很多的时候就加大了服务器与数据库的负荷。

10、订单、库存两个表 如何保证数据的一致性?

答:在一个电子商务系统中，正常的应该是订单生成成功后，相应的库存进行减少。必须要保证两者的一致性，但有时候因为某些原因，比如程序逻辑问题，并发等问题，导致下单成功而库存没有减少的情况。这种情况我们是不允许发生的，MySQL 中的事务刚好可以解决这一问题，首先得选择数据库的存储引擎为 innoDB,事务规定了只有下订单完成了，并且相应的库存减少了才允许提交事务，否则就事务回滚，确保数据一致性。

11、支付宝流程怎么实现的

首先要有一个支付宝账号，接下来向支付宝申请在线支付业务，签署协议。协议生效后有支付宝一方会给网站方一个合作伙伴 ID,和安全校验码，有了这两样东西就可以按照支付宝接口文档开发支付宝接口了，中间主要涉及到一个安全问题。整个流程是这样的：我们的网站通过 post 传递相应的参数（如订单总金额，订单号）到支付页面，支付页面把一系列的参数经过处理，以 post 的方式提交给支付宝服务器，支付宝服务器进行验证，并对接收的数据进行处理，把处理后的结果返回给我们网站设置的异步和同步回调地址，通过相应的返回参数，来处理相应的业务逻辑，比如返回的参数代表支付成功，更改订单状态。

12、支付宝的支付流程，notify_url和return_url的区别

买家付款成功后，如果接口中有制定的return_url 买家付完款后会跳到 return_url所在的页面,这个页面可以展示给客户看,这个页面只有付款成功才会跳转.

notify_url:服务器后台通知,这个页面是程序后台运行的(买家和卖家都看不到),买家付完款后,支付宝会调用notify_url这个页面所在的页面并把相应的参数传递到这个页面,这个页面根据支付宝传递过来的参数修改网站订单的状态,更新完订单后需要在页面上打印出一个success给支付宝,如果反馈给支付宝的不是success,支付宝会继续调用这个页面. 流程:买家付完款(trade_status=WAIT_SELLER_SEND_GOODS)--->支付宝通知notify_url---& gt;如果反馈给支付宝的是success(表示成功,这个状态下不再反馈,如果不是继续通知,一般第一次发送和第二次发送的时间间隔是3分钟) 剩下的过程,卖家发货,买家确认收货,交易成功都是这个流程

13、微信的支付流程实现

1. 设置支付目录：确保实际支付时的请求目录与后台配置的目录一致
2. 设置授权域名：只有被设置过的域名才能够获取用户OPENID，否则将获取失败。
3. 用户通过微信浏览器打开商户的h5，商户后台系统生成商户订单，并调用微信统一下单接口，生成预付单，用户点击发起支付，微信系统检查参数合法性何授权域权限，用户确认支付输入密码，提交支付授权并验证，异步通知商户支付结果，告知微信通知处理结果，返回支付结果并发消息提示给用户。查询商户后台支付结果，调用查询API查询支付结果并返回。

14、微信支付模式有哪些?

1. 刷卡支付--MICROPAY
2. 扫码支付--NATIVE
3. 公众号支付--JSAPI
4. APP支付--APP

15、微信支付调用API需要遵循哪些规则?

1. 为了安全性，数据传输方式采用HTTPS
2. 数据提交方式采用的POST提交
3. 数据格式提交何返回都为XML格式
4. 同意采用UTF-8字符编码
5. 采用MD5签名算法

16、登陆界面登陆密码，防止明文传输。一般在开发中会怎么处理?

可以使用RSA非对称加密算法。 首先服务器端会产生一对公钥和私钥，然后客户端在需要加密的时候会异步请求服务器端，获取公钥，并利用公钥进行加密操作（加密表单密码元素），服务器端收到加了密的用户密码，再使用私钥进行解密，从而获取加密前的明文密码。由于仅仅有了公钥并不能解密，必须使用服务器的私钥解码，所以在web端的项目中使用时能提高密码的安全性。