1. 秒杀活动 ① 锁机制 ② redis队列

<?php

$fp = fopen("lock.txt", "w+");

if(flock($fp,LOCK\_EX | LOCK\_NB))

{

//..处理订单

flock($fp,LOCK\_UN);

}

else

{

echo "系统繁忙，请稍后再试";

}

fclose($fp);

?>

1. jQ和JS不太明白混淆，建议讲讲；

JS ：javascript脚本语言

jQ ：jquery的缩写，实际就是javascript封装的代码库

1. 数据库的优化（查询变慢时）；
2. Sql语句优化，索引优化，字段优化；

① 存储层(数据)：存储引擎、列类型、范式规范

② 设计层(单台服务器)： 索引、缓存、分区

③ 架构层(多台服务器)： 读写分离(主从复制)

④ sql语句层

1. Web服务器优化（并发量变高时）；
2. 如何优化大数据，大流量的网站（如何处理网站高并发问题）；
3. 关于服务器的安全问题及公司有几台服务器，分别存放什么；

① 安装杀毒软件（Windows Server、Linux）

② Linux设置防火墙（哪些服务可以通过，哪些服务不允许通过）

③ 设置SeLinux防护系统

5人左右小公司，一台服务器足够

10人-20人公司，二台服务器或三台服务器

20人以上的大公司，3-5台

1. Memcache与redis区别，及他们在项目中的应用；

① 存储方式不同

② 数据支持类型

Redis不仅仅支持简单的k/v类型的数据，同时还提供list，set，hash等数据结构的存储

③使用底层模型不同

新版本的redis直接自己构建了VM 机制 ，因为一般的系统调用系统函数的话，会浪费一定的时间去移动和请求

④ Redis支持数据的持久化，可以将内存中的数据保持在磁盘中，重启的时候可以再次加载进行使用

1. Mysql中varchar和char的区别，及哪种效率高，原因；

varchar与char都可以用于存储字符串类型，但是varchar是变长，而char是定长。

如果数据长度固定，采用char的效率比较高

如果数据长度不固定，建议采用varchar类型，更加节省空间

1. Session与cookie区别及session共享，及它们在项目中的应用

cookie是在服务器端设置，在客户端保存

session无论设置还是保存都是在服务器端完成的

session共享，多个网站如何共享session？

① 文件存储

② Session入DB数据库（两个网站同时连一个DB）

③ Session入Memcache（两个网站同时连一个Memcache服务器）

1. Ajax是什么？它的核心技术，原理，优缺点及在项目中的应用；

异步的Javascript与XML，核心技术XMLHttpRequest，可以在无刷新的前提下获取数据，体验度非常好。

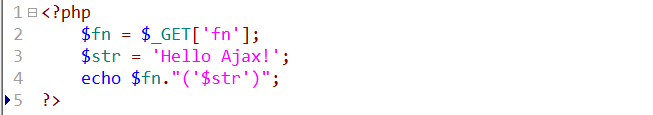
缺点：不利于SEO优化，无法被搜索引擎所抓取

1. Ajax跨域问题解决；

可以使用JSONP技术解决Ajax的跨域问题，代码如下：



deal.php



1. Smarty模板特点；

速度快、编译型、支持缓存技术、丰富的插件机制、语句自由if/else

1. 接口(interface)和抽象类(abstract)的区别；

① 接口也是是抽象类

② 接口可以多继承，抽象类不行

③ 接口定义方法，不能实现，而抽象类可以实现部分方法。

④ 接口中基本数据类型为static 而抽类象不是的。

1. 设计模式（单例模式，工厂模式实现的原理及应用）；

① 单例模式主要用于封装DB类，防止其重复生成对象

② 工厂模式，传递什么样的数据，就生成什么样的对象，如PDO类的封装

单例模式

<?php

require\_once("DB.php");

class DatabaseConnection

{

public static function get()

{

static $db = null;

if ( $db == null )

$db = new DatabaseConnection();

return $db;

}

private $\_handle = null;

private function \_\_construct()

{

$dsn = 'mysql://root:password@localhost/photos';

$this->\_handle =& DB::Connect( $dsn, array() );

}

public function handle()

{

return $this->\_handle;

}

}

print( "Handle = ".DatabaseConnection::get()->handle()."\n" );

print( "Handle = ".DatabaseConnection::get()->handle()."\n" );

?>

工厂模式

<?php

interface IUser

{

function getName();

}

class User implements IUser

{

public function \_\_construct( $id ) { }

public function getName()

{

return "Jack";

}

}

class UserFactory

{

public static function Create( $id )

{

return new User( $id );

}

}

$uo = UserFactory::Create( 1 );

echo( $uo->getName()."\n" );

?>

1. 算法问题（冒泡，选择，二分法）；

冒泡：

$arr=array(1,43,54,62,21,66,32,78,36,76,39);

function getpao($arr)

{

$len=count($arr);

//设置一个空数组 用来接收冒出来的泡

//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;

}

选择排序：

function select\_sort($arr) {

//实现思路 双重循环完成，外层控制轮数，当前的最小值。内层 控制的比较次数

//$i 当前最小值的位置， 需要参与比较的元素

for($i=0, $len=count($arr); $i<$len-1; $i++) {

//先假设最小的值的位置

$p = $i;

//$j 当前都需要和哪些元素比较，$i 后边的。

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，

// 应该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;

}

二分法：

<?php

function search($key, $array) {

$top = count($array) - 1;

$low = 0;

$mid = 0;

while ($low <= $top) {

$mid = floor(($top+$low)/2);

if($array[$mid] == $key){

return $mid;

}

if($array[$mid] > $key){

$top = $mid - 1;

}else {

$low = $mid + 1;

}

}

return false;

}

$arr = array(1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 888);

$position = search(70, $arr);

if($position !== false){

print('搜索元素的索引是:' . $position . "\n");

}else{

print("没有搜索到元素\n");

}

?>

1. 如何排查PHP网站运行速度慢的原因；

① 网站服务器速度或租用空间所在服务器速度

② 电信和联通互访瓶颈问题

③ 从网站本身找问题（网站程序设计、网站设计问题、网页内容）

④ 优化图片、优化CSS、减少http请求

⑤ 数据库表结构不合理

⑥ 用explain数据库查询语句不合理

⑦ 尽量使用静态页面

⑧ 过多引用了其他网站内容

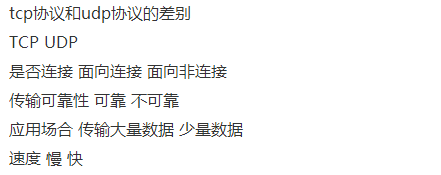
1. TCP协议，HTTP协议传输的原理；
2. TCP协议的三次握手；

① 第一次握手，客户端发送请求，但是服务器端还未确认

② 第二次握手，服务器已确认，准备返回数据到客户端

③ 第三次握手，客户端已接收到返回结果并确认

1. UDP和TCP的区别；



1. 网络的七层结构；

① 物理层 ② 数据链路层 ③ 网络层 ④ 传输层 ⑤ 会话层 ⑥ 表示层 ⑦ 应用层

1. 如何屏蔽PHP的错误信息；

① 错误抑制符@

② ini\_set(‘display\_errors’,’Off’);

1. InnoDB及MyIsam的区别；

① 结构不同(Myisam一共有三个文件，innoDB有两个文件)

② 事务处理方面（MyISAM类型的表强调的是性能，其执行数度比InnoDB类型更快，但是不提供事务支持）

③ 增删改查操作（如果执行大量的SELECT，MyISAM是更好的选择）

④ MyISAM：只支持表级锁，用户在操作myisam表时，select，update，delete，insert语句都会给表自动加锁，如果加锁以后的表满足insert并发的情况下，可以在表的尾部插入新的数据。

InnoDB：支持事务和行级锁，是innodb的最大特色。行锁大幅度提高了多用户并发操作的新能。但是InnoDB的行锁，只是在WHERE的主键是有效的，非主键的WHERE都会锁全表的。

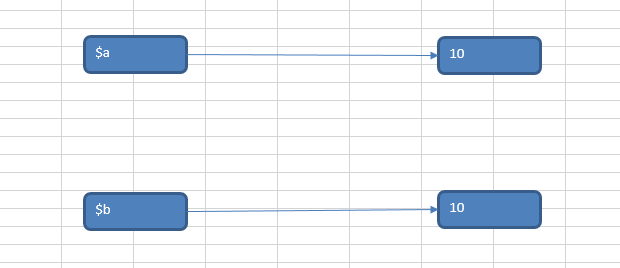
⑤ MyISAM：支持 FULLTEXT类型的全文索引

InnoDB：不支持FULLTEXT类型的全文索引，但是innodb可以使用sphinx插件支持全文索引，并且效果更好。

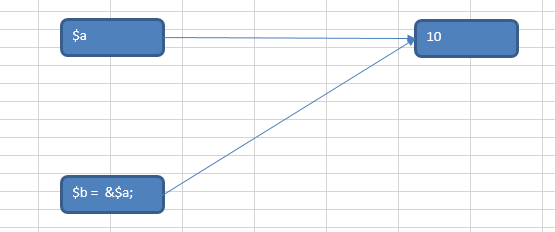
⑥ 外键（MyISAM：不支持，InnoDB：支持）

1. 面向对象的特征（封装、继承、多态）
2. Php中传值和引用传值的区别，什么时候用传值，什么时候用引用传值；

① 按值传递



② 按引用传递



主要看数据量的大小，如果数据量比较大，建议使用引用传递。如果数据量比较小可以使用按值传递。

1. 如何防止xss攻击和sql注入；

① addslashes

② mysql\_real\_escape\_string

③ htmlpurifier插件来解决xss攻击

1. 栈和队列；

栈：相当于水桶效果，先进后出

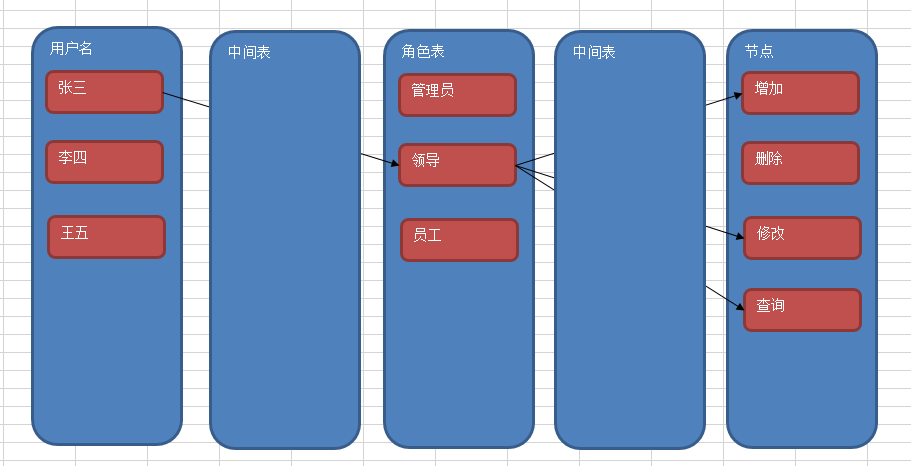
队列：相当于排队效果，先进现出

1. 项目中购物车的实现及其表的设计，使用了几张表及表中有哪些字段等；
2. 项目中登录，注册模块，订单，支付，物流模块的实现（学员不知道如何去说）；
3. 用什么方法检查PHP脚本的执行效率和数据库sql的效率，并定位和分析脚本执行和数据库查询的瓶颈所在？

① microtime

② slow\_query\_log

1. 现在有一个系统，要求每天登陆后进行打卡登记，用户每天第一次登陆要提醒用户登记。
2. 写出实现思想；

32、大部分未找到工作的学员不会表述项目，建议讲师串讲电商项目具体模块实现的业务逻辑及表的设计和使用的技术；  
33、rbac不太明白。资料   


34、秒杀活动，有没有写过接口、linux、memcache、redis

① 调用别人接口（支付宝、快递、短信）

get请求 ：file\_get\_contents来实现

post请求 ：curl库

② 给Android或IOS提供数据接口（中间数据格式，XML与JSON）

编写一个方法，读取数据库，把返回的数据生成为XML或JSON格式，一般为JSON格式。

json\_encode ：把一个数组或对象生成JSON格式  
35、实际项目的小问题、数据库优化、高并发、秒杀

① 优化你的MySQL查询缓存

② 用EXPLAIN使你的SELECT查询更加清晰

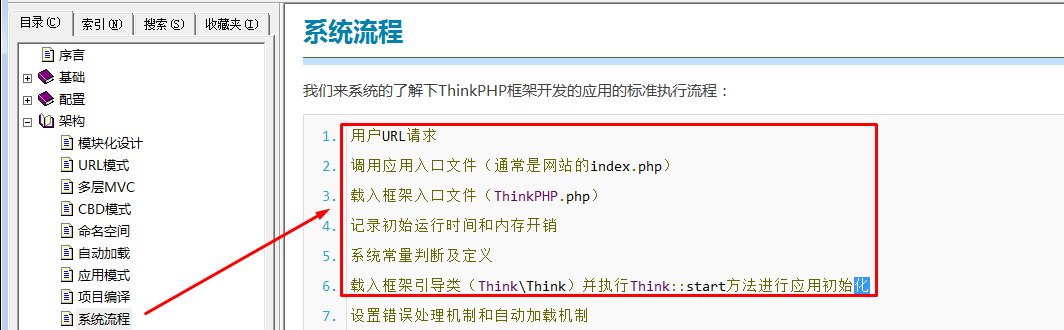
③ 利用LIMIT 1取得唯一行

④ 不要使用BY RAND()命令

⑤ 尽量避免SELECT \*命令

36、session运行原理，浏览器兼容、微信二次开发、写接口、session共享在多台服务器上怎么实现

<http://www.jb51.net/article/91391.htm> PHP编写接口给IOS  
37、怎么理解TP框架的底层代码，底层工作原理

  
38、缓存在项目中的实际应用、购物车是怎么实现的、支付的业务逻辑  
39、redis与memached的选择。  
40、mysql优化缓存、http协议、数据库事务

所谓的事务就是就是要求事务进行原子性的划分，在执行过程中要么多成功，要么都失败。

$pdo->beginTransaction(); 开启事务

$pdo->commit(); 提交事务（成功）

$pdo->rollback(); 回滚事务（失败）

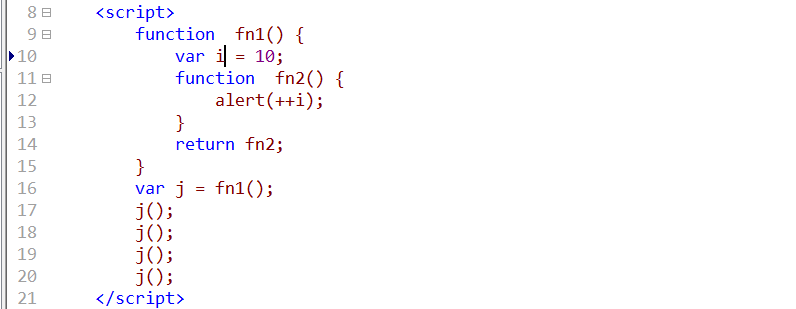
特别注意：事务处理对数据表引擎有要求，必须使用InnoDB引擎  
41、http和ajax请求的区别

http：同步模式，请求—处理—返回结果—请求—处理…

ajax：异步模式，请求—处理-返回一部分数据—请求…

42、项目中我们如何考虑安全问题、JS闭包、规避数据库锁死的办法、网站接口技术，项目中实现接口，如何调用

JS闭包：就是函数中的函数

① 可以访问局部变量

② 可以让局部变量一直驻留计算机的内存中（类似PHP中的静态变量）

① 尽量避免并发地执行涉及到修改数据的语句

② 要求每个事务一次就将所有要使用的数据全部加锁，否则就不予执行。

③ 将经常更新的数据库和查询数据库分开