**1**、酒店预订怎么实现？怎么设计表

你好，我大概的说下我们的业务流程，我们的业务流程是：用户在网站浏览酒店信息，可以根据地区检索出该地区的酒店信息。列表展示酒店的信息由：酒店的名称，酒店图片，酒店位置，评论人数，评论分数以及最低入住价格。用户选中要入住的酒店进入酒店详情页面，查看酒店的介绍以及酒店的房型列表，用户根据他要入住的时间和离店的时间，检索出这个时间段内的所有可选房型（房间数量-当天的订单-当天未离店订单=剩余房间数量）显示给用户。用户选择好房型后就可以进行下单，要求有订单的开始时间，结束时间，房间数量，住客姓名，抵店时间，联系方式，备注信息等等。

 那我的表是这么设计的，总共有6张表，分别是：

用户表user，里面有下面几个字段，(用户编号，用户名称，用户密码，用户联系方式)

酒店表hotel，里面有(酒店编号，酒店名称，酒店图片，评论人数，评论分数，最低入住价格，所在地区)

酒店图片表pic(图片编号，图片地址，图片排序，图片所属酒店)

评论表comment(评论编号，评论内容，评论时间，用户编号，酒店编号)

房型表house(房型编号，床型，早餐，宽带，人数上限，房价，房间数量，最长预定时间)

订单表order(订单编号，开始时间，结束时间，房间数量，住客姓名，最晚抵店时间，联系电话，使用优惠券，备注，订单状态)

以上就是我对这个酒店预订系统的设计

**2**、预定时间怎么写入数据库的

以预订当时的时间戳作为预订时间写入数据库。用户下订单时会选择一个抵店时间，将该抵店时间以时间戳方式存入数据库中。离店时间以当时的日期转为时间戳方式存入数据库中

**3**、怎么判断还有没有房间

我可以根据用户的入住时间和离店时间来检索这个有效时间段内房间的库存。房间数量扣除在这个时间段内入住的订单和在这个时间段内离店的订单。扣除后等到的数量才是这段时间内有效房间数量。

**4**、怎么记录每天的房间库存

我的思路是根据一个公式来推理实现的，每天房间的库存=房型下房间数量-（当天入住的订单+当天未离店的订单），这样我就可以得到每天还有多少房间是剩余的了。

**5**、怎么在数据库里对房间做唯一标识

上面所设计的房型表就是我们的房间表，每个房间是唯一的，我们是使用数字作为编号的，也即使用主键作为唯一标识。

**6**、最近出的新功能

   最近我们出了个会员机制，客户第一次预订酒店成功后，可以办理会员卡，凭借会员卡，下次来的时候可以打折，会员在一些比较特殊的日期预订酒店成功，可以享受不一样的优惠措施。

7、**怎么保证促销商品不会超卖**

   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：

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**、redis集群怎么做

1、[Redis](http://lib.csdn.net/base/redis)集群提供了以下两个好处  
1、将数据自动切分(split)到多个节点  
2、当集群中的某一个节点故障时，redis还可以继续处理客户端的请求。  
2、集群的方案：

   redis-cluster集群,采用无中心结构，每个节点保存数据和整个集群状态,每个节点都和其他所有节点连接,主要通过节点的配置，辅以redis的主从来完成集群。由于这块东西我使用得很少，所以只是平时抽时间去研究过，并没有真正的在线上实现过。

**9**、redis和memcacahe、mongoDB的区别

答：都是非关系型数据库，性能都非常高，但是mongoDB和memcache、redis是不同的两种类型。后两者主要用于数据的缓存，前者主要用在查询和储存大数据方面，是最接近数据库的文档型的非关系数据库。

 这里我主要谈谈memcache和redis的区别。

①从数据存储位置上来分，memcache的数据存在内存中，而redis既可以存储在内存中，也可以存储的到磁盘中，达到持久化存储的功能，memcache一旦断电，数据全部丢失，redis可以利用快照和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从存储数据的类型上来分，memcache和redis存储的方式都是键值对，只不过redis值的类型比较丰富，有string(字符串),hash(哈希)，list(列表),set(集合)zset(有序集合)，而memcache主要存储的是字符串。

 ③从架构层次来分，Redis支持master-slave(主—从)模式应用，memcache支持分布式。

 ④另外从存储数据的大小上来分，Redis单个value的最大限制是1GB，memcached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，当然关于redis取代memcache的说法，在一般情况下，两者性能都很高，在大多的业务场景选择上，redis的选择可能更加具有优势，但也不能说可以完全取代,最终还是取决于你的应用场景。

**10**、持久化redis有几种方式？

答：主要有两种方式：

① 快照持久化

在redis配置文件中已经自动开启了，

格式是：save N M

表示在N秒之内，redis至少发生M次修改则redis抓快照到磁盘。

当然我们也可以手动执行save或者bgsave（异步）命令来做快照

②append only file  AOF持久化

 总共有三种模式，如

appendfsync everysec默认的是每秒强制写入磁盘一次

appendfsync always 每次执行写操作的时候就强制写入磁盘

appendfsync no 完全取决于os，性能最好但是持久化没法保证

其中第三种模式最好。redis默认的也是采取第三种模式。

**11**、mysql存储引擎

答：常用的主要分为两种，一种是innodb,一种是myisam,两者的主要区别是

     ①myisam不支持事务处理，而innoDB支持事务处理

②myisam 不支持外键，innoDB支持外键

③myisam支持全文检索，而innoDB在MySQL5.6版本之后才支持全文检索

④数据的存储形式不一样，mysiam表存放在三个文件：结构、索引、数据，innoDB存储把结构存储为一个文件，索引和数据存储为一个文件

⑤myisam在查询和增加数据性能更优于innoDB，innoDB在批量删除方面性能较高。

⑥myisam支持表锁，而innoDB支持行锁

**12、 sql注入是什么及如何预防sql注入？**

答：SQL注入攻击指的是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。因此我们在做开发过程中一定要预防sql注入，主要从两方面着手：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。

**13、有用过预处理么？**

答：用过，PDO类中，有个prepare方法可以实现预处理，PDOStament类中 的excute方法可以执行预处理，预处理的参数分为两种，一种是:字符串占位符，另一种是?占位符，:字符串占位符在执行预处理传递参数时传入的是关联数组，而?占位符传递的是索引数组。两者不能混合使用，但一般推荐使用:字符串占位符。

**14、用框架还用自己的处理吗**

答：一般成熟的开源框架中都考虑到了数据安全这方面的东西，但有时候我们可能会使用一些原生的SQL语句时，我们就需要考虑自己对sql语句进行预处理。当然有时候框架中的过滤方法我们不希望采用，比如使用文本编辑器时，我们可以使用自己的过滤方式。

**15、mysql优化怎么做的？**

答：mysql优化主要从以下几个方面来实现：

  ①设计角度：存储引擎的选择，字段类型选择，范式

  ②功能角度：可以利用mysql自身的特性，如索引，查询缓存，碎片整理，分区、分表等

  ③sql语句的优化方面:尽量简化查询语句，能查询字段少就尽量少查询字段，优化分页语句、分组语句等。

  ④部署大负载架构体系：数据库服务器单独出来，负载大时可以采用主从复制，读写分离机制进行设计

  ⑤从硬件上升级数据库服务器。

**16、订单表用是什么存储引擎**

答：因为订单表存在着事务的处理，比如下了订单，商品的库存就要减少，这里就涉及到了事务，所以就用到innodb。

**19**、sql语句的优化

答：首先我们得确定哪些sql语句需要优化，一般在一个系统中，查询语句最多，所以我们主要是针对查询语句进行优化。主要采用两种方式来确定要优化的sql语句：

   ①使用慢查询日志，设置需要优化的sql语句的执行时间，记录下超过该设置时间的语句，即为需要优化的语句。

   ②使用profiling机制，记录下每条sql语句的执行时间，找出执行较慢的语句，即为需要优化的语句。

   我们主要通过给表字段添加索引的方式进行优化，加上索引后，sql语句的执行时间显著提高了，但并不是加上索引了这条sql语句就会用到索引，所以首先看执行慢的语句后面是否有加索引，我们可以使用explain或者desc加在要执行的sql语句前，查看是否使用到索引。有几个地方需要注意的是：

  ①为了避免建议索引而造成索引文件过大，有时候我们会使用复合索引，这时候要遵循最左原则。

  ②like查询，前%不会用到索引

  ③如果条件中有or，则要求or的索引字段都必须有索引，否则不能用到索引。

  ④如果列类型是字符串，一定要在条件中将数据使用引号引用起来，否则不使用索引。

  ⑤优化group by 语句

  ⑥尽量避免模糊匹配，这样会导致全盘扫描

**21**、 索引有几种

答：索引主要有：

      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。

**23**、左前索引原则

答：左前索引主要指的是在复合索引中，给两个或多个字段建立了复合索引后，在sql语句后的条件中，只有复合索引前面的字段在条件的前面时，该索引才起作用，比如创建了个复合索引index （a，b），在使用where或者orderby条件时，如果只有条件b的，该索引不会生效，必须有条件a且必须要在条件b的前面该索引才会生效。

**24**、分布式数据库

答：我所知道的分布式数据库有memcache，主要是分布式的非关系型数据库，用于缓存处理。

    分布式是指将不同的业务分布在不同的地方。 而集群指的是将几台服务器集中在一起，实现同一业务。  
分布式中的每一个节点，都可以做集群。 而集群并不一定就是分布式的。

举例：就比如新浪网，访问的人多了，他可以做一个群集，前面放一个响应服务器，后面几台服务器完成同一业务，如果有业务访问的时候，响应服务器看哪台服务器的负载不是很重，就将给哪一台去完成。  
而分布式，从窄意上理解，也跟集群差不多， 但是它的组织比较松散，不像集群，有一个组织性，一台服务器垮了，其它的服务器可以顶上来。

  memcache的应用场景

1、适用memcached的业务场景？

1）如果网站包含了访问量很大的动态网页，因而数据库的负载将会很高。由于大部分数据库请求都是读操作，那么memcached可以显著地减小数据库负载。

2）利用memcached可以缓存session数据、临时数据以减少对他们的数据库写操作。

4）缓存一些很小但是被频繁访问的文件。

5）访问比较频繁，安全性不高，丢失无所谓，修改比较频繁的数据，比如一些用户的在线状态

2 、不适用memcached的业务场景？

1）缓存对象的大小大于1MB

memcache本身就不是为了处理庞大的多媒体（large media）和巨大的二进制块（streaming huge blobs）而设计的。

2）key的长度大于250字符

3）应用运行在不安全的环境中

4）业务本身需要的是持久化数据或者说需要的应该是database

**25**、nginx日志，怎么统计每个ip的访问量

（参考阿铭哥手册）

   stub\_status模块主要用于查看Nginx的一些状态信息，例如统计nginx的访问量,首先我们得查看该模块有没有安装，如果没有安装，得先安装，安装好后，修改nginx的配置文件，开启该模块，然后就可以使用以下命令来进行统计，如：

 1.根据访问IP统计UV

awk '{print $1}'  access.log|sort | uniq -c |wc -l

2.统计访问URL统计PV

awk '{print $7}' access.log|wc -l

3.查询访问最频繁的URL

awk '{print $7}' access.log|sort | uniq -c |sort -n -k 1 -r|more

4.查询访问最频繁的IP

awk '{print $1}' access.log|sort | uniq -c |sort -n -k 1 -r|more

统计nginx日志中访问最多的100个ip及访问次数

awk ‘{print $1}’ access.log|sort | uniq -c |sort -n -k 1 -r| head -n 100

**26**、http协议

HTTP是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统

HTTP协议的主要特点可概括如下：  
1.支持客户/服务器模式。  
2.简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用 的有GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。由于HTTP协议简单，使得HTTP服务器的程序规模小，因而通信速度很快。  
3.灵活：HTTP允许传输任意类型的数据对象。正在传输的类型由Content-Type加 以标记。  
4.无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请 求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。  
5.无状态：HTTP协议是无状态协议。无状态是指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，则它必须重传，这样可能导致每次 连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。

**27**、cookie与session的区别

1、cookie数据存放在客户的浏览器上，session数据放在服务器上。

2、cookie不是很安全，别人可以分析存放在本地的COOKIE并进行COOKIE欺骗  
        考虑到安全应当使用session。

3、session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能  
        考虑到减轻服务器性能方面，应当使用COOKIE。

4、单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie。

5、所以个人建议：  
        将登陆信息等重要信息存放为SESSION  
       其他信息如果需要保留，可以放在COOKIE中

**28**、php在储存session以什么形式存在

PHP为session的存储提供了三种方式: 文件/ 内存/ 自定义存储,默认是使用文件存储.在访问量大的网站上采用这种方式就不大合 适,因为这样会导致大量的输入输出的冗余.我们可以在php.ini更改配置文件或者php脚本中通过相应的函数来设置session文件的存储类型来改变session文件的存储形式

**29**、xss攻击怎么防止

XSS又称CSS，全称Cross SiteScript(跨站脚本攻击)， XSS攻击类似于SQL注入攻击，是Web程序中常见的漏洞，XSS属于被动式且用于客户端的攻击方式，所以容易被忽略其危害性。其原理是攻击者向有XSS漏洞的网站中输入(传入)恶意的HTML代码，当用户浏览该网站时，这段HTML代码会自动执行，从而达到攻击的目的。如，盗取用户Cookie信息、破坏页面结

常见的恶意字符XSS输入：

1. XSS 输入通常包含 JavaScript 脚本，如弹出恶意警告框：<script>alert("XSS");</script>

2. XSS 输入也可能是 HTML 代码段，譬如：

(1) 网页不停地刷新 <meta http-equiv="refresh" content="0;">

(2) 嵌入其它网站的链接 <iframe src=http://xxxx width=250 height=250></iframe>

构、重定向到其它网站等。

方法：利用php htmlentities()函数

php防止XSS跨站脚本攻击的方法：是针对非法的HTML代码包括单双引号等，使用htmlspecialchars()函数。

在使用htmlspecialchars()函数的时候注意第二个参数, 直接用htmlspecialchars($string)的话，第二个参数默认是ENT\_COMPAT，函数默认只是转化双引号(")，不对单引号(')做转义。

所以，htmlspecialchars()函数更多的时候要加上第二个参数，应该这样用: htmlspecialchars($string,ENT\_QUOTES)。当然，如果需要不转化如何的引号，用htmlspecialchars($string,ENT\_NOQUOTES)。

另外，尽量少用htmlentities(), 在全部英文的时候htmlentities()和htmlspecialchars()没有区别，都可以达到目的。但是，中文情况下, htmlentities()却会转化所有的html代码，连同里面的它无法识别的中文字符也给转化了。

htmlentities()和htmlspecialchars()这两个函数对单引号(')之类的字符串支持不好，都不能转化， 所以用htmlentities()和htmlspecialchars()转化的字符串只能防止XSS攻击，不能防止SQL注入攻击。

所有有打印的语句如echo，print等，在打印前都要使用htmlentities()进行过滤，这样可以防止XSS，注意中文要写出htmlentities($name,ENT\_NOQUOTES,GB2312)。

**30**、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。

**31**、mongodb基于什么开发的

MongoDB是一个基于分布式文件存储的数据库。由C++语言编写。旨在为WEB应用提供可扩展的高性能数据存储解决方案。

32、mongodb是非范式还是范式

数据表示的方式有很多种，其中最重要的问题之一就是在多大程度上对数据进行范式化。范式化（normalization）是将数据分散到多个不同的集合，不同集合之间可以相互引用数据。虽然很多文档可以引用某一块数据，但是这块数据只存储在一个集合中。所以，如果要修改这块数据，只需修改保存这块数据的那一个文档就行了。但是，MongoDB没有提供连接（join）工具，所以在不同集合之间执行连接查询需要进行多次查询。

反范式化（denormalization）与范式化相反：将每个文档所需的数据都嵌入在文档内部。每个文档都拥有自己的数据副本，而不是所有文档共同引用同一个数据副本。这意味着，如果信息发生了变化，那么所有相关文档都需要进行更新，但是在执行查询时，只需要一次查询，就可以得到所有数据。

决定何时采用范式化何时采用反范式化时比较困难的。范式化能够提高数据写入速度，反范式化能够提高数据读取速度。需要根据自己应用程序的十几需要仔细权衡。

**33**、mongodb与mysql区别

MySQL是关系型数据库。

   优势：

在不同的引擎上有不同 的存储方式。

查询语句是使用传统的sql语句，拥有较为成熟的体系，成熟度很高。

开源数据库的份额在不断增加，mysql的份额页在持续增长。

   缺点：

在海量数据处理的时候效率会显著变慢。

Mongodb是非关系型数据库(nosql ),属于文档型数据库。文档是mongoDB中数据的基本单元，类似关系数据库的行，多个键值对有序地放置在一起便是文档，语法有点类似javascript面向对象的查询语言，它是一个面向集合的，模式自由的文档型数据库。

存储方式：虚拟内存+持久化。

查询语句：是独特的Mongodb的查询方式。

适合场景：事件的记录，内容管理或者博客平台等等。

架构特点：可以通过副本集，以及分片来实现高可用。

数据处理：数据是存储在硬盘上的，只不过需要经常读取的数据会被加载到内存中，将数据存储在物理内存中，从而达到高速读写。

成熟度与广泛度：新兴数据库，成熟度较低，Nosql数据库中最为接近关系型数据库，比较完善的DB之一，适用人群不断在增长。

优点：

快速！在适量级的内存的Mongodb的性能是非常迅速的，它将热数据存储在物理内存中，使得热数据的读写变得十分快。高扩展性，存储的数据格式是json格式！

缺点：

不支持事务，而且开发文档不是很完全，完善。

   Mysql和Mongodb主要应用场景（简单了解叙述下即可）

1.如果需要将mongodb作为后端db来代替mysql使用，即这里mysql与mongodb 属于平行级别，那么，这样的使用可能有以下几种情况的考量： (1)mongodb所负责部分以文档形式存储，能够有较好的代码亲和性，json格式的直接写入方便。(如日志之类) (2)从data models设计阶段就将原子性考虑于其中，无需事务之类的辅助。开发用如nodejs之类的语言来进行开发，对开发比较方便。 (3)mongodb本身的failover机制，无需使用如MHA之类的方式实现。

2.将mongodb作为类似redis ，memcache来做缓存db，为mysql提供服务，或是后端日志收集分析。 考虑到mongodb属于nosql型数据库，sql语句与数据结构不如mysql那么亲和 ，也会有很多时候将mongodb做为辅助mysql而使用的类redis memcache 之类的缓存db来使用。 亦或是仅作日志收集分析。

**34**、写一个函数统计每一个元素出现的次数

PHP 中的 array\_count\_values() 函数可以实现 array\_count\_values() 函数用于统计数组中所有值出现的次数。 本函数返回一个数组，其元素的键名是原数组的值，键值是该值在原数组中出现的次数。

**35**、手写排序

主要从原理方面来说：重点介绍冒泡排序和选择排序

· // 冒泡排序

function BubbleSort($arr) {

    // 获得数组总长度

    $num = count($arr);

    // 正向遍历数组

    for ($i = 1; $i < $num; $i++) {

        // 反向遍历

        for ($j = $num - 1; $j >= $i ; $j--) {

            // 相邻两个数比较

            if ($arr[$j] < $arr[$j-1]) {

                // 暂存较小的数

                $iTemp = $arr[$j-1];

                // 把较大的放前面

                $arr[$j-1] = $arr[$j];

                // 较小的放后面

                $arr[$j] = $iTemp;

            }

        }

    }

    return $arr;

}

·  // 交换法排序

function ExchangeSort($arr){

    $num = count($arr);

    // 遍历数组

    for ($i = 0;$i < $num - 1; $i++) {

        // 获得当前索引的下一个索引

        for ($j = $i + 1; $j < $num; $j++) {

            // 比较相邻两个的值大小

            if ($arr[$j] < $arr[$i]) {

                // 暂存较小的数

                $iTemp = $arr[$i];

                // 把较大的放前面

                $arr[$i] = $arr[$j];

                // 较小的放后面

                $arr[$j] = $iTemp;

            }

        }

    }

    return $arr;

}

·  // 选择法排序

function SelectSort($arr) {

    // 获得数组总长度

    $num = count($arr);

    // 遍历数组

    for ($i = 0;$i < $num-1; $i++) {

        // 暂存当前值

        $iTemp = $arr[$i];

        // 暂存当前位置

        $iPos = $i;

        // 遍历当前位置以后的数据

        for ($j = $i + 1;$j < $num; $j++){

            // 如果有小于当前值的

            if ($arr[$j] < $iTemp) {

                // 暂存最小值

                $iTemp = $arr[$j];

                // 暂存位置

                $iPos = $j;

            }

        }

        // 把当前值放到算好的位置

        $arr[$iPos] = $arr[$i];

        // 把当前值换成算好的值

        $arr[$i] = $iTemp;

    }

    return $arr;

}

·  // 插入法排序

function InsertSort($arr){

    $num = count($arr);

    // 遍历数组

    for ($i = 1;$i < $num; $i++) {

        // 获得当前值

        $iTemp = $arr[$i];

        // 获得当前值的前一个位置

        $iPos = $i - 1;

        // 如果当前值小于前一个值切未到数组开始位置

        while (($iPos >= 0) && ($iTemp < $arr[$iPos])) {

            // 把前一个的值往后放一位

            $arr[$iPos + 1] = $arr[$iPos];

            // 位置递减

            $iPos--;

        }

        $arr[$iPos+1] = $iTemp;

    }

    return $arr;

}

·  // 快速排序

function QuickSort($arr){

    $num = count($arr);

    $l = $r = 0;

    $left = $right = array();

    // 从索引的第二个开始遍历数组

    for ($i = 1;$i < $num; $i++) {

        // 如果值小于索引1

        if ($arr[$i] < $arr[0]) {

            // 装入左索引数组(小于索引1的数据)

            $left[] = $arr[$i];

            $l++;

        } else {

            // 否则装入右索引中(大于索引1的数据)

            $right[] = $arr[$i];

            $r++; //

        }

    }

    // 如果左索引有值 则对左索引排序

    if($l > 1) {

        $left = QuickSort($left);

    }

    // 排序后的数组

    $new\_arr = $left;

    // 将当前数组第一个放到最后

    $new\_arr[] = $arr[0];

    // 如果又索引有值 则对右索引排序

    if ($r > 1) {

        $right = QuickSort($right);

    }

    // 根据右索引的长度再次增加数据

    for($i = 0;$i < $r; $i++) {

        $new\_arr[] = $right[$i];

    }

    return $new\_arr;

}

**36**、设计模式

   在PHP中，我主要使用了以下两种设计模式

1、单例模式

单例模式顾名思义，就是只有一个实例。作为对象的创建模式， 单例模式确保某一个类只有一个实例，而且自行实例化并向整个系统提供这个实例。

单例模式的要点有三个：

一是某个类只能有一个实例；

二是它必须自行创建这个实例；

三是它必须自行向整个系统提供这个实例。

典型的代表如框架中的基类对象。

2、简单工厂模式

①抽象基类：类中定义抽象一些方法，用以在子类中实现

②继承自抽象基类的子类：实现基类中的抽象方法

③工厂类：用以实例化所有相对应的子类

    这种我们使用最常见，基本所有的MVC框架中都是这样产生的。

**37**、用过什么PHP框架

在开发过程中，我主要使用过了这么几种框架。thinkPHP框架、CI框架，laravel框架和yii框架。我接触到的第一个框架是TP框架，我简单的说下我对这几个框架的看法：

ThinkPHP框架

  优点:

TP借鉴了Java思想，基于PHP5，充分利用了PHP5的特性，部署简单只需一个入口文件，一切搞定，简单高效，中文文档齐全，入门超级简单。自带模板引擎，具有独特的数据验证和自动填充功能，框架更新速度比较迅速。

 缺点：一个Model中可以操作多个表，但TP只能一个。  
TP默认初始化了很多配置，使用起来很方便，但自然也会影响效率。但是把一些加载配置的时间拿去研究算法，这些小影响近乎可以忽略了。

CodeIgniter框架

优点:

配置简单，上手很快，全部的配置使用PHP脚本来配置，没有使用很多太复杂的设计模式，执行性能和代码可读性上都不错，执行效率比较高，具有基本的MVC功能. 快速简洁，代码量少，框架简单，容易上手，自带了很多简单好用的library，框架适合中小型项目，大型项目也不是不可以，只是扩展能力稍差。

缺点：

1. 把Model层简单的理解为数据库操作

2. PHP框架略显简单，只能够满足小型应用，略微不太能够满足中型应用需要

laravel框架（目前最新的是5.3，要求PHP版本较高5.6）  
       优点:

   1.Laravel注重代码的模块化和可扩展性。

   2.artisan: 命令行工具，很多手动的工作都自动了

   3.可继承的模版，简化view的开发和管理

   Laravel一直是PHP开发者最受欢迎的PHP框架。这是一个年轻的框架，但是拥有优雅的语法，可简单快速开发你的应用。它拥有大多数常见的功能，如：路由，身份验证，会话，队列和缓存。

  缺点：

   laravel的中英文文档比较少 demo也比较少 有时候一个功能要试好久 甚至要看源码

YII框架（目前是2.0版本）

优点：

 1、快速，敏捷，不拖沓，给程序员飞翔的能力；

 2、有gii功能！(创建控制器,model层,crud等操作)；

 3、具有高度的可重用性和可扩展性，是纯粹的面向对象的。开发速度快，完备的文档，可重用性可高扩展,是最高效的开发框架之一。

缺点：

1、对Model层的指导和考虑较少

2、文档实例较少

3、英文太多

4、要求PHP技术精通，OOP编程要熟练！

5、要求会bootstrap

**38**、代码管理工具

   我使用过的版本控制工具有两种：早期的时候使用的是SVN，现在主要使用git，我就我个人的观点，简单的说下两者的区别：

1. Git是分布式的，SVN是集中式的，好处是跟其他同事不会有太多的冲突，自己写的代码放在自己电脑上，一段时间后再提交、合并，也可以不用联网在本地提交；

2. Git下载下来后，在本地不必联网就可以看到所有的log，很方便学习，SVN却需要联网；

3. Git鼓励分Branch（分支），而SVN，说实话，我用Branch的次数还挺少的，SVN自带的Branch merge我还真没用过，有merge时用的是Beyond Compare工具合并后再Commit的；

4. SVN在Commit前，我们都建议是先Update一下，跟本地的代码编译没问题，并确保开发的功能正常后再提交

SVN 的主要功能

    SVN属于集中化的版本控制系统，有个不太精确的比喻:SVN = 版本控制+ 备份服务器

     SVN使用起来有点像是档案仓库的感觉，支持并行读写文件，支持代码的版本化管理，功能包括取出、导入、更新、分支、改名、还原、合并等。

     功能有许多我就不一一列了，SVN大都采用图形界面操作，直观，上手快。

Git的主要功能

      Git是一个分布式版本控制系统，操作命令包括：clone，pull，push,branch ,merge ,rebas，Git擅长的是程序代码的版本化管理。

SVN 的优缺点

      SVN对中文支持好，操作简单，使用没有难度，美工人员，产品人员，测试人员，实施人员都可轻松上手。使用界面统一，功能完善，操作方便。

Git的优缺点

      对程序源代码进行差异化的版本管理，代码库占极少的空间。易于代码的分支化管理。不支持中文，图形界面支持差，使用难度大。不易推广。

SVN 和 Git 哪个更适用于项目管理？

     SVN更适用于项目管理， Git仅适用于代码管理。

     一个研发队伍的成员正常包括：需求分析、设计、美工、程序员、测试、实施、运维，每个成员在工作中都有产出物，  包括了文档、设计代码、程序代码，这些都需要按项目集中进行管理的。SVN能清楚的按目录进行分类管理， 使项目组的管理处于有序高效的状态。

现在越来越多人使用git做为版本控制工具，我以前的公司也是使用git.

**39**、手写单例模式怎么写

  三私一公。

class Example

{

//保存例实例在此属性中

private static $\_instance;

//构造函数声明为private,防止直接创建对象

private function \_\_construct()

{

echo 'I am Construceted';

}

//单例方法

public static function singleton()

{

if(!isset(self::$\_instance))

{

$c=\_\_CLASS\_\_;

self::$\_instance=new $c;

}

return self::$\_instance;

}

//阻止用户复制对象实例

public function \_\_clone()

{

trigger\_error('Clone is not allow' ,E\_USER\_ERROR);

}

function test()

{

echo("test");

}

}

// 这个写法会出错，因为构造方法被声明为private

$test = new Example;

// 下面将得到Example类的单例对象

$test = Example::singleton();

$test->test();

// 复制对象将导致一个E\_USER\_ERROR.

$test\_clone = clone $test;

?>

**40**、nosql和Mysql的区别

   也即非关系型数据库和关系型数据库。

目前世界上主流的存储系统大部分还是采用了关系型数据库，其主要有一下优点：

1.事务处理—保持数据的一致性；

2.由于以标准化为前提，数据更新的开销很小（相同的字段基本上只有一处）；

3.可以进行Join等复杂查询。

nosql在优势方面，主要体现在下面这三点：   
1. 简单的扩展：典型例子是Cassandra，由于其架构是类似于经典的P2P，所以能通过轻松地添加新的节点来扩展这个集群;  
2. 快速的读写：主要例子有Redis，由于其逻辑简单，而且纯内存操作，使得其性能非常出色，单节点每秒可以处理超过10万次读写操作;   
3. 低廉的成本：这是大多数分布式数据库共有的特点，因为主要都是开源软件，没有昂贵的License成本;   
4.   
但瑕不掩瑜，NoSQL数据库还存在着很多的不足，常见主要有下面这几个：   
1. 不提供对SQL的支持：如果不支持SQL这样的工业标准，将会对用户产生一定的学习和应用迁移成本;   
2. 支持的特性不够丰富：现有产品所提供的功能都比较有限，大多数NoSQL数据库都不支持事务，也不像 SQL Server和[Oracle](http://lib.csdn.net/base/oracle)那样能提供各种附加功能，比如BI和报表等;   
3. 现有产品的不够成熟：大多数产品都还处于初创期，和关系型数据库几十年的完善不可同日而语;

**41**、在TP中M方法与D方法的区别

虽然都是实例化模型对象，两者还是有区别的

D和M的区别主要在于

M方法不需要创建模型类文件，M方法不会读取模型类，所以默认情况下自动验证是无效的，但是可以通过动态赋值的方式实现

而D方法必须有创建模型类。

我们可以用下面两种方法去创建一个数据表的映射对象

第一种：$Test = D(‘Test’)

第二种：$Test = new Model(‘Test’)

虽然这两种都可以对数据进行select,insert,delete,udpate操作，在

数据验证上有很大的不同，

用第一种方式实例一个模型就会有数据检查功能，如果 title 没有填写的话就会提示 “请输入标题” （这个是tp提供的一个自动验证功能，当然也需要在相应的model中定义好验证条件）；

如果用第二种就没有了这个数据验证功能，需要手动验证。

D函数实例化的是你当前项目的Lib/Model下面的模块。  
如果该模块不存在的话，直接返回实例化Model的对象(意义就与M()函数相同)。  
而M只返回，实例化Model的对象。它的$name参数作为[数据库](http://lib.csdn.net/base/mysql)的表名来处理对数据库的操作。

**42**、对网站大访问量的优化方案

   提高访问速度。从硬件，最好从网站程序等等方面考虑。我给出以下几种方案：

1.尽量使用静态页，不要老使用动态信息调用。非常容易出问题  
2.图片内容与网站数据尽量放在同一个服务器或者机房内。大量外链图片是会有问题的  
3.一次又一次，一遍又一遍的分析流量走向，然后缩短浏览者浏览距离，举个例子，浏览者如果现在在你网站看一个新闻需要点5次鼠标，你就要缩短这个点击数。  
4.一次又一次，一遍又一遍的分析，修改你的网站[数据库结构](https://www.baidu.com/s?wd=%E6%95%B0%E6%8D%AE%E5%BA%93%E7%BB%93%E6%9E%84&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YvrH0kujc1mWR3PjnYuWwh0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En10knWmLrHnY)，使其更加简洁。  
5.提高网站的安防能力  
6.买个好服务器，托管在一个好的机房！

**43**、[网站高并发大流量访问的处理及解决方法](http://blog.csdn.net/haiqiao_2010/article/details/24478027)

  第一:确认服务器硬件是否足够支持当前的流量。   
普通的P4服务器一般最多能支持每天10万独立IP，如果访问量比这个还要大，那么必须首先配置一台更高性能的专用服务器才能解决问题，否则怎么优化都不可能彻底解决性能问题。  
第二：优化数据库访问   
　前台实现完全的静态化当然最好，可以完全不用访问数据库，不过对于频繁更新的网站，静态化往往不能满足某些功能。  
　缓存就是另一个解决方案，就是将动态数据存储到缓存文件中，动态网页直接调用这些文件，而不必再访问数据库，技术如果确实无法避免对数据库的访问，那么可以尝试优化数据库的查询SQL.避免使用Select \* from这样的语句，每次查询只返回自己需要的结果，避免短时间内的大量SQL查询。最好在相同字段进行比较操作,在建立好的索引字段上尽量减少函数操作，如果要做到极致的话需要代码的优化；  
　第三，禁止外部的盗链。   
   外部网站的或者文件盗链往往会带来大量的负载压力，因此应该严格限制外部对于自身的图片或者文件盗链，好在目前可以简单地通过refer来控制盗链，自己就可以通过配置来禁止盗链。当然，伪造refer也可以通过来实现盗链，不过目前蓄意伪造refer盗链的还不多，可以先不去考虑，或者使用非技术手段来解决，比如在图片上增加水印。  
第四，控制大文件的下载。   
   大文件的下载会占用很大的流量，并且对于非SCSI硬盘来说，大量文件下载会消耗CPU，使得网站响应能力下降。因此，尽量不要提供超过2M的大文件下载，如果需要提供，建议将大文件放在另外一台服务器上。  
第五，使用不同主机分流主要流量   
   将文件放在不同的主机上，提供不同的镜像供用户下载。比如如果觉得RSS文件占用流量大，那么使用FeedBurner或者FeedSky等服务将RSS输出放在其他主机上，这样别人访问的流量压力就大多集中在FeedBurner的主机上，RSS就不占用太多资源了。  
第六，使用流量分析统计软件。   
  在网站上一个流量分析统计软件，可以即时知道哪些地方耗费了大量流量，哪些页面需要再进行优化，因此，解决流量问题还需要进行精确的统计分析才可以。我推荐使用的流量分析统计软件是Analytics（Google分析）。

**45**、主要运用到哪些缓存

一、数据缓存

这里所说的数据缓存是指数据库查询缓存，每次访问页面的时候,都会先检测相应的缓存数据是否存在，如果不存在，就连接数据库，得到数据，并把查询结果序列化后保存到文件中，以后同样的查询结果就直接从缓存表或文件中获得。

用的最广的例子看Discuz的搜索功能，把结果ID缓存到一个表中，下次搜索相同关键字时先搜索缓存表。  
举个常用的方法，多表关联的时候，把附表中的内容生成数组保存到主表的一个字段中，需要的时候数组分解一下，这样的好处是只读一个表，坏处就是两个数据同步会多不少步骤，数据库永远是瓶颈，用硬盘换速度，是这个的关键点。

二、页面缓存

每次访问页面的时候，都会先检测相应的缓存页面文件是否存在，如果不存在，就连接数据库，得到数据，显示页面并同时生成缓存页面文件，这样下次访问的时候页面文件就发挥作用了。(模板引擎和网上常见的一些缓存类通常有此功能)。

三、时间触发缓存

检查文件是否存在并且时间戳小于设置的过期时间,如果文件修改的时间戳比当前时间戳减去过期时间戳大，那么就用缓存，否则更新缓存。

四、内容触发缓存

当插入数据或更新数据时，强制更新缓存。

五、静态缓存

这里所说的静态缓存是指静态化，直接生成HTML或XML等文本文件，有更新的时候重生成一次，适合于不太变化的页面，这就不说了。  
    六、内存缓存  
     Memcached是高性能的，分布式的内存对象缓存系统，用于在动态应用中减少数据库负载，提升访问速度。redis 也可以做到。

**46**、php的设计模式

1、单例模式 2、工厂模式 3、观察者模式 4、命令链模式 5、策略模式

单例模式：

一个类在整个应用中，只有一个对象实例的设计模式

类必须自行创建这个实例

必须自行向整个系统提供这个实例

三私：私有静态成员变量、构造函数、克隆函数

一公：公共的静态方法

2、工厂模式

可以根据输入的参数或者应用程序配置的不同一创建一种专门用来实例化并返回其它类的实例的类

3、观察者模式

观察者模式提供了组件之间紧密耦合的另一种方法。

该模式：一个对象通过添加一个方法（该方法允许另一个对象，即观察者注册自己）全本身变得可观察。当可观察的对象更改时，它会将消息发送到已注册的观察者。这些观察者使用该信息执行的操作与可观察的对象无关。

4、命令链模式：

以松散耦合主题为基础，发送消息、命令和请求，或通过一组处理程序发送任意内容。每个处理程序都会自行判断自己能否处理请求，如果可以，该请求被处理，进程停止。

5、策略模式：

此算法是从复杂类提取的，因而可以方便地替换。

**47.Mysql**事务的特性

事务是作为一个逻辑单元执行的一系列操作，一个逻辑工作单元必须有四个属性，称为 [ACID](https://www.baidu.com/s?wd=ACID&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3uA7WnHmLmWIWuyPbPWN-0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHTzrHR4njmY)（原子性、一致性、隔离性和持久性）属性，只有这样才能成为一个事务：  
原子性  
事务必须是原子工作单元；对于其数据修改，要么全都执行，要么全都不执行。  
一致性  
事务在完成时，必须使所有的数据都保持一致状态。在相关数据库中，所有规则都必须应用于事务的修改，以保持所有数据的完整性。  
    事务结束时，所有的内部数据结构（如 B 树索引或双向链表）都必须是正确的。  
隔离性  
由并发事务所作的修改必须与任何其它并发事务所作的修改隔离。事务查看数据时数据所处的状态，要么是另一并发事务修改它之前的状态，  
       要么是另一事务修改它之后的状态，事务不会查看中间状态的数据。这称为可串行性，因为它能够重新装载起始数据，  
       并且重播一系列事务，以使数据结束时的状态与原始事务执行的状态相同。

持久性  
事务完成之后，它对于系统的影响是永久性的。该修改即使出现系统故障也将一直保持。

begin 开始一个事务

rollback事务回滚

commit事务确认

**48**、Mysql事务的应运场景

事务处理在各种管理系统中都有着广泛的应用，比如人员管理系统，很多同步数据库操作大都需要用到事务处理。比如说，在人员管理系统中，你删除一个人员，你即需要删除人员的基本资料，也要删除和该人员相关的信息，如信箱，文章等等，这样，这些数据库操作语句就构成一个事务！

比如手机充值过程，支付宝金额减少，相应的手机话费增加，只要有一个操作不成功，则另外一个操作也不会成功

**49.Include**和require的区别

require函数通常放在PHP程序的最前面，在PHP程序执行之前，就会先读取require指定引入的文件，使它变成PHP程序网页的一部分。

include函数一般是放在流程控制的处理部分中。PHP程序在读到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。

他们两个的用途是一样的，不一定非要哪个放在最前面哪个放在中间，他们最根本的区别在于错误处理的方式不一样。

require一个文件存在错误的话，那么程序就会中断执行，并显示致命错误

而include一个文件存在错误的话，那么程序不会中断，会继续执行，并显示一个警告的错误

其它区别：include有返回值，而require没有。

**50**、索引的建立与使用

     索引就是类似书的目录，提高检索数据的效率。

索引是系统按照某个具体的算法（哈希，散列，二叉树），将数据从全部数据里进行提取，维护成一个索引文件，然后系统在进行数据查询的时候，发现如果查询条件刚好满足索引条件，就可以从索引文件中快速的定位的数据所在位置。

mysql中有以下几种索引：

主键索引（primary key效率最高的索引）

唯一索引（unique key）：不为空的情况下效率最高

普通索引（index）对数据没有要求，文件很大，效率比较低

全文索引（fulltext），对整个文章内部进行关键字索引（mysql5.5以后InnoDB支持全文索引）

英文的全文索引很简单：英文单词默认是用空格分离的

中文的全文索引很难：中文的词组成很麻烦，需要利用分词工具（sphinx）

     索引可以在创建表的同时创建索引，也可以在修改表结构时添加索引，索引主要是加在经常做为查询条件的字段上，可以使用相应的手段来检测所执行的sql语句中是否使用到了索引。

**51.**正则匹配表达式各个符文表达的意义

\ 将下一个字符标记为一个特殊字符、或一个原义字符、或一个向后引用、或一个八进制转义符。例如，’n’ 匹配字符 “n”。’\n’ 匹配一个换行符。序列 ‘\\’ 匹配 “\” 而 “\(” 则匹配 “(”。

^ 匹配输入字符串的开始位置。如果设置了 RegExp 对象的 Multiline 属性，^ 也匹配 ‘\n’ 或 ‘\r’ 之后的位置。

$ 匹配输入字符串的结束位置。如果设置了RegExp 对象的 Multiline 属性，$ 也匹配 ‘\n’ 或 ‘\r’ 之前的位置。

\* 匹配前面的子表达式零次或多次。例如，zo\* 能匹配 “z” 以及 “zoo”。\* 等价于{0,}。

+ 匹配前面的子表达式一次或多次。例如，’zo+’ 能匹配 “zo” 以及 “zoo”，但不能匹配 “z”。+ 等价于 {1,}。

? 匹配前面的子表达式零次或一次。例如，”do(es)?” 可以匹配 “do” 或 “does” 中的”do” 。? 等价于 {0,1}。

{n} n 是一个非负整数。匹配确定的 n 次。

{n,} n 是一个非负整数。至少匹配n 次。

{n,m} m 和 n 均为非负整数，其中n <= m。最少匹配 n 次且最多匹配 m 次。

. 匹配除 “\n” 之外的任何单个字符。要匹配包括 ‘\n’ 在内的任何字符，请使用象 ‘[.\n]’ 的模式。

x|y 匹配 x 或 y。

[xyz] 字符集合。匹配所包含的任意一个字符。例如， ‘[abc]’ 可以匹配 “plain” 中的 ‘a’。

[^xyz] 负值字符集合。匹配未包含的任意字符。例如， ‘[^abc]’ 可以匹配 “plain” 中的’p'。

[a-z] 字符范围。匹配指定范围内的任意字符。例如，’[a-z]’ 可以匹配 ‘a’ 到 ‘z’ 范围内的任意小写字母字符。

[^a-z] 负值字符范围。匹配任何不在指定范围内的任意字符。例如，’[^a-z]’ 可以匹配任何不在 ‘a’ 到 ‘z’ 范围内的任意字符。

\b 匹配一个单词边界，也就是指单词和空格间的位置。例如， ‘er\b’ 可以匹配”never” 中的 ‘er’，但不能匹配 “verb” 中的 ‘er’。

\d 匹配一个数字字符。等价于 [0-9]。

\D 匹配一个非数字字符。等价于 [^0-9]。

\f 匹配一个换页符。等价于 \x0c 和 \cL。

\n 匹配一个换行符。等价于 \x0a 和 \cJ。

\r 匹配一个回车符。等价于 \x0d 和 \cM。

\s 匹配任何空白字符，包括空格、制表符、换页符等等。等价于 [ \f\n\r\t\v]。

\S 匹配任何非空白字符。等价于 [^ \f\n\r\t\v]。

\t 匹配一个制表符。等价于 \x09 和 \cI。

\v 匹配一个垂直制表符。等价于 \x0b 和 \cK。

\w 匹配包括下划线的任何单词字符。等价于’[A-Za-z0-9\_]’。

\W 匹配任何非单词字符。等价于 ‘[^A-Za-z0-9\_]’。

**52.PHP**的变量类型

四种标量类型：

boolean （布尔型）：这是最简单的类型，只有两种取值，可以为 TRUE/true 或 FALSE/false ，不区分大小写。详细请查看：PHP布尔类型（boolean）

integer （整型）：在32 位操作系统中它的有效范围是：-2 147 483 648~+2 147 483 647。整型值可以使用十进制，十六进制或八进制表示，前面可以加上可选的符号（- 或者 +）。八进制表示数字前必须加上 0（零），十六进制表示数字前必须加上 0x。

float （浮点型, 也称作 double)

string （字符串）：字符型变量不同于其他编程语言有字符与字符串之分，在PHP 中，统一使用字符型变量来定义字符或者字符串。

两种复合类型：

array （数组）：数组型变量是一种比较特殊的变量类型，将在后续章节中详细说明。

object （对象）：对象也是一种特殊的数据类型。要创建object变量，请使用 new 关键字。详细请查看：PHP对象类型（object）

最后是两种特殊类型：

resource（资源）：源是一种特殊变量，保存了到外部资源的一个引用。资源是通过专门的函数来建立和使用的。详情请查看：PHP资源类型（resource）

NULL（NULL）：表示一个变量没有值。NULL 类型唯一可能的值就是 NULL。

**54.**商城秒杀的实现

  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](http://lib.csdn.net/base/mysql)，将用户入结果队列。

**55.**购物车的原理

 购物车相当于现实中超市的购物车，不同的是一个是实体车，一个是虚拟车而已。用户可以在购物网站的不同页面之间跳转，以选购自己喜爱的商品，点击购买时，该商品就自动保存到你的购物车中，重复选购后，最后将选中的所有商品放在购物车中统一到付款台结账，这也是尽量让客户体验到现实生活中购物的感觉。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。下面分析一下它们的机制及作用。

1. cookie

     cookie是由服务器产生，存储在客户端的一段信息。它定义了一种Web服务器在客户端存储和返回信息的机制，cookie文件它包含域、路径、生存期、和由服务器设置的变量值等内容。当用户以后访问同一个Web服务器时，浏览器会把cookie原样发送给服务器。通过让服务器读取原先保存到客户端的信息，网站能够为浏览者提供一系列的方便，例如在线交易过程中标识用户身份、安全要求不高的场合避免用户重复输入名字和密码、门户网站的主页定制、有针对性地投放广告等等。利用cookie的特性，大大扩展了WEB应用程序的功能，不仅可以建立服务器与客户机的联系，因为cookie可以由服务器定制，因此还可以将购物信息生成cookie值存放在客户端，从而实现购物车的功能。用基于cookie的方式实现服务器与浏览器之间的会话或购物车，有以下特点：

   1、 cookie存储在客户端，且占用很少的资源，浏览器允许存放300个cookie，每个cookie的大小为4KB，足以满足购物车的要求，同时也减轻了服务器的负荷；

   2、 cookie为浏览器所内置，使用方便。即使用户不小心关闭了浏览器窗口，只要在cookie定义的有效期内，购物车中的信息也不会丢失；

  3、 cookie不是可执行文件，所以不会以任何方式执行，因此也不会带来病毒或攻击用户的系统；

  4、  基于cookie的购物车要求用户浏览器必须支持并设置为启用cookie，否则购物车则失效；

  5、 存在着关于cookie侵犯访问者隐私权的争论，因此有些用户会禁止本机的cookie功能。

2. session

     session是实现购物车的另一种方法。session提供了可以保存和跟踪用户的状态信息的功能，使当前用户在session中定义的变量和对象能在页面之间共享，但是不能为应用中其他用户所访问，它与cookie最重大的区别是，session将用户在会话期间的私有信息存储在服务器端，提高了安全性。在服务器生成session后，客户端会生成一个sessionid识别号保存在客户端，以保持和服务器的同步。这个sessionid是只读的，如果客户端禁止cookie功能，session会通过在URL中附加参数，或隐含在表单中提交等其他方式在页面间传送。因此利用session实施对用户的管理则更为安全、有效。

     同样，利用session也能实现购物车，这种方式的特点是：

1、 session用新的机制保持与客户端的同步，不依赖于客户端设置；

2、 与cookie相比，session是存储在服务器端的信息，因此显得更为安全，因此可将身份标示，购物等信息存储在session中；

3、session会占用服务器资源，加大服务器端的负载，尤其当并发用户很多时，会生成大量的session，影响服务器的性能；

4、因为session存储的信息更敏感，而且是以文件形式保存在服务器中，因此仍然存在着安全隐患。

3. 结合数据库的方式

     这也是目前较普遍的模式，在这种方式中，数据库承担着存储购物信息的作用，session或cookie则用来跟踪用户。这种方式具有以下特点：

1、 数据库与cookie分别负责记录数据和维持会话，能发挥各自的优势，使安全性和服务器性能都得到了提高；

2、每一个购物的行为，都要直接建立与数据库的连接，直至对表的操作完成后，连接才释放。当并发用户很多时，会影响数据库的性能，因此，这对数据库的性能提出了更高的要求；

3、使cookie维持会话有赖客户端的支持。

各种方式的选择：

     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。

**56.redis**消息队列先进先出需要注意什么

   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 0  
  
list1 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

**57**、你负责的模块有哪些难题

    在我负责的B2B电商项目中，当时我负责的是订单模块，由于客户一次选择了多家商户的商品，最终生成了一个订单，这样我们平台在给商户结算时出现了不知道这比费用应该给哪个商户，这时候我们小组经过讨论，需要涉及到订单拆分，也就是说用户点击支付后,如果有多件商品,并且不是同一家店铺那么 就要用到订单的拆分,比如如果有两件商品,并且不是同一店铺 就在原来的订单号下 在生成两个子订单号 并修改订单表中两件商品的订单号。最终实现了商品的分配管理，解决了我们的难题。

    我觉得在开发过程中，遇到的难题无非是两个，一个是技术层次的，我认为，只要你有恒心，有热心，没有觉得不了的难题。另一个就是沟通问题，在任何地方任何时候沟通都是最重要的，尤其是我们做开发的，不沟通好，会影响整个项目的进度，我本人是个非常还沟通的人，所以这点上也没多大问题。

**58**、 用户下单是怎么处理的

  判断用户有没有登录，在没有登录的情况下，不允许下单。登陆后，可进行下单

  并生成唯一的订单号，此时订单的状态为未支付。

**59**、电商的登录是怎么登录的

  分为普通登录和第三方登录 这边主要说一下第三方登录吧，第三方登陆主要使用的是author协议，我就以QQ的第三方登陆为例来进行说明：当用户在我们的站点请求QQ的第三方登陆时，我们站点会引导用户跳转到QQ的登陆授权界面， 当用户输入QQ和密码成功登录以后会自动跳回到我们站点设置好的回调页面，并附带一个code参数，接着你使用code再次去请求QQ的授权页面，就可以从中获取到一个access token（访问令牌），通过这个access\_token，我们可以调用QQ提供给我们的接口，比如获取open\_id，可以获取用户的基本信息。获取到之后，我们需要拿用户的授权信息和open\_id和我们平台的普通用户进行绑定。这样不管是普通用户登陆还是第三方登陆用户，都可以实现登陆。

**60**、有负责开发app吗

     我以前有过使用 hybrid APP开发过APP，做过一个简单的广场舞APP，但我主要参与到APP的接口编写这块中。

**61**、开发app过程中遇到了什么难题

     hybrid APP开发过程中，前端知识是我的硬伤，以前，前端我一直都没有花太多精力在上面，所以在使用hybrid APP 开发过程中，页面样式总是调得很难看，后面我花了一周时间自己自学恶补了以下前端的东西，感觉收获很大。还有我记得以前在公司写接口时，我们的安卓工程师认为他们APP的分页效果，得我们接口这边事先分好页，然后他们再调用接口，其实分页的页码需要他那边提供给我们，但是他就认定了是我们这边的问题，后面经过多次沟通和测试，我们共同完成了这项任务。

**62**、linux杀死一个进程命令是什么

    首先，我们得知道要杀死的进程的进程ID，可以通过ps -ef | grep 进程名称 查到当前运行的进程ID，然后通过kill命令杀死进程 ，如 kill  -9  3329 表示强制杀死进程，当然还有不同的等级，取决于中间的数字。

63 、redis 常用的数据类型

  Redis 的数据类型主要有：

     string：字符串类型，可以包含任何数据。包括jpg图片或者序列化的对象。里面的incr方法可以实现网站计数器功能，每次访问一个就可以进行加1操作。降低了数据库的压力。

     list：是一个双向链表，通过push,pop操作从链表的头部或者尾部添加删除元素。

这使得list既可以用作栈，也可以用作队列。比如可以获取最新添加的10个商品，获取最新的登陆的10个信息，做商品的秒杀等等。都可以通过链表中的队列来实现，极大节省了各方面的资源。

     hash：hash数据类型是redis模仿数据库把一条记录信息给存储起来，这样可以把数据库中的每一条记录保存在hash中，作为缓存处理，非常接近于数据库的操作。

     set ：set是string类型的无序集合。set集合类型除了基本的添加删除操作，其他有用的操作还包含集合的取并集(union)，交集(intersection)，差集(difference)。通过这些操作可以很容易的实现sns中的好友推荐功能。比如qq好友推荐、微博系统的关注关系使用

     sorted set：和set一样sorted set也是string类型元素的集合，不同的是每个元素都会关联一个权。通过权值可以有序的获取集合中的元素，它的适用场合如:获得热门帖子(回复量)信息,根据学生成绩排序获得信息等.

    简单说下无序集合、有序集合、链表三者的主要区别：

set类型：集合类型、内部元素没有顺序，同一个集合没有重复元素

list链表类型：内部元素有彼此的先后顺序，同一个链表允许有重复元素

Sort set类型：排序集合类型，相比set类型有排序功能

**64**、你是如何测试网站的性能的

   常用的网站性能测试有：[压力测试](https://www.baidu.com/s?wd=%E5%8E%8B%E5%8A%9B%E6%B5%8B%E8%AF%95&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLm19WuWP-rHmsPvPhPhc40ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPj6YnWD1nj04)，[负载测试](https://www.baidu.com/s?wd=%E8%B4%9F%E8%BD%BD%E6%B5%8B%E8%AF%95&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLm19WuWP-rHmsPvPhPhc40ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPj6YnWD1nj04)，容量测试，并发性能测试，[兼容性](https://www.baidu.com/s?wd=%E5%85%BC%E5%AE%B9%E6%80%A7&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLm19WuWP-rHmsPvPhPhc40ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPj6YnWD1nj04)测试（不同的[操作系统](https://www.baidu.com/s?wd=%E6%93%8D%E4%BD%9C%E7%B3%BB%E7%BB%9F&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1YLm19WuWP-rHmsPvPhPhc40ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPj6YnWD1nj04)和不同的浏览器）。在项目正式上线前，我们技术部会使用压力测试工具来测试网站的性能（我们主要是进行压力测试的）。我主要用过两款软件：一个apache自带的ab压力测试工具，这个测试的最大并发量相对较小，一般1000左右就会出现请求拒绝。另一个软件是webbench,这个软件首先得安装，最大并发可以到3W。当然还有一些其他的专业的测试工具，如国外的 [Page Speed Online](https://developers.google.com/pagespeed/)、[Pingdom Tools](http://fpt.pingdom.com/)等等，我们公司有专门的测试部，我们会配合他们完成测试工作。

**65.ab**命令参数是什么

ab.exe  -n  5000  -c 50  http://www.test.com/index.php

-n是总的执行次数，-c 并发的次数， http://www.test.com/index.php要执行的文件

**66**、接口安全方面是怎么处理的

   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。

**67**、用的什么技术实现短信发送，在哪调用

  我主要用的第三方短信接口，在申请接口时进行相应信息的配置，然后在我们站点需要用到短信验证的地方进行调用，我们通常在用户注册时使用到。

**68**、上一家公司用的什么框架写的项目，还接触过什么框架?

   我的上一家公司主要使用的是XXX框架，我对该框架非常熟悉，我们公司在该框架上做了一些相应的扩展，引入了一些自己编写的类库文件和插件库。我以前还使用过yii2,ci、laravel框架，以前还自己封装过MVC框架。一个新的框架掌握起来很容易，你只要抓住其中的几个点，比如路由规则、MVC、数据库相关的操作，其他的都可以查手册，孰能生巧，通过一个小项目就可以把框架用得很熟，当然框架底层的东西，我们还是得用一些好的IDE工具去追它的底层源码。

**69**、在工作中遇到什么困难?

   总体来说：在工作我主要遇到这几个问题比较难处理：

   ①我之前工作的时候发现经常会出现一些临时需求打乱了我的计划，搞得有时候这个任务还没完成，又得去做其他的任务，最后一天下来，大大小小的东西是很多，但是没有完成得非常好的，后面我总结了一下，我会把这些都添加优先级，遇到临时需求，按照优先级重新将已有任务和临时任务进行排版，保证在规定时间内有效率的完成优先级高的任务。

  ②在做项目需求时候，遇到理解能力欠佳的人，沟通时容易被气到，影响自己的情绪，最后反倒还不能到达需要的效果。后面，每次到这种时候，我一般会借助一些纸质的、更加形象的东西，让双方都认同的、都能明白的一种方式来进行沟通，后面减少了很多不必须的麻烦。大家都知道，对于程序员来说，改需求是一件很痛苦的事情，所以前期的沟通工作很重要。

 ③还有一件事时，我以前的领导不太懂技术，所以每次出一个新的需求出来，总是要求我们在很短的时间内完成，完不成我们就会被怀疑能力有问题。当然，每个领导都希望自己的员工能够尽快的完成任务，降低成本，提高效率。这时候我会把我们的需求细化，把其中的重点、难点都列出来，做好时间规划，耐心的跟领导沟通，项目每个点的重要性和时间的花费比例，确保在这个规划的时间点内保质保量的完成任务。慢慢的也得到了领导的认可，其实领导也不是一味的不通情理，只要把东西计划好了，以最小的代价换取最高的价值，每个人都是很容易理解得。

**70**、Memcache缓存机制 以及分布式

Memcached是一个高性能的分布式内存对象缓存系统。目前全世界不少人使用这个缓存项目来构建自己大负载的网站，来分担[数据库](http://lib.csdn.net/base/mysql)的压力，通过在内在里维护一个统一的巨大的的hash表，它能够用来存储各种格式的数据，包括图像、视频、文件以及数据库检索的结果等，简单的说就是将数据调用到内存中，然后从内存中读取，从而大大提高读取速度。

 传统的查询方法是直接查询数据库，数据库将结果返回给查询语句，而当有Memcache中间缓存层时，查询的是Memcache缓存数据，下面详细了解Memcache各类数据操作原理：

1. 查询数据(select)，首先通过指定的Key查询(get)Memcache中间缓存层数据，如果存在相对应数据，则直接获取出数据结果，查询过程完全不需要查询数据库。如果不存在，则查询MySQL数据库，并以key对应value的形式将查询结果存储在Memcache缓存数据中，然后将结果返回给查询语句。  
2. 更新数据(update)，首先更新数据，然后删除相关的memcache数据(delete)。  
3. 增加数据(add)，首先删除相关缓存数据，然后增加数据。  
4. 删除数据(delete)，删除数据，并删除Memcache数据。

memcache的应用场景有：

 1.如果是一个小网站，pv值不大，就不考虑使用memcached  
 2.变化频繁，查询频繁，但是不一定写入数据库（适合memcached）（用户在线状态）  
 3.变化频繁，一变化就要入库（比如股票，金融）不适合memcached  
 4.变化不频繁，查询频繁，不管入不入库，都比较适合memcache，（新浪的新闻频道）

    分布式是指将不同的业务分布在不同的地方（几台服务器）。 而集群指的是将几台服务器集中在一起，实现同一业务。  
分布式中的每一个节点，都可以做集群。 而集群并不一定就是分布式的。

举例：就比如新浪网，访问的人多了，他可以做一个集群，前面放一个响应服务器，后面几台服务器完成同一业务，如果有业务访问的时候，响应服务器看哪台服务器的负载不是很重，就交给哪一台去完成。而分布式，从窄意上理解，也跟集群差不多，但是它的组织比较松散，不像集群，有一个组织性，一台服务器垮了，其它的服务器可以顶上来。memcache的分布式算法取决于客户端，主要用取余算法和一致性哈希算法。

**71**、 直播是怎么实现的?

   ①直播视频从技术架构角度主要分为三个部分：

     直播视频采集SDK=>直播CDN（也即直播流分发加速）=》直播视频播放器SDK

  ②音视频处理的一般流程：

   数据采集→数据编码→数据传输(流媒体服务器) →解码数据→播放显示

1、数据采集：摄像机及拾音器收集视频及音频数据，此时得到的为原始数据，要用到摄像机和拾音器。

2、数据编码:使用相关硬件或软件对音视频原始数据进行编码处理（数字化）及加工（如音视频混合、打包封装等），得到可用的音视频数据,编码的方式：CBR、VBR，编码格式分为音频和视频，音频一般有MP3、OGG、AAC，视频一般有TS、MKV、AVI、MP4等。

3、数据传输：将编码完成后的音视频数据进行传输，早期的音视频通过同轴电缆之类的线缆进行传输，IP网络发展后，使用IP网络优传输

涉及技术或协议：

传输协议：RTP与RTCP、RTSP、RTMP、HTTP、HLS（HTTP Live Streaming）等

控制信令：SIP和SDP、SNMP等

4、解码数据：

使用相关硬件或软件对接收到的编码后的音视频数据进行解码，得到可以直接显示的图像/声音

涉及技术或协议：

一般对应的编码器都会带有相应的解码器，也有一些第三方解码插件等

5、播放显示：

在显示器（电视、监视屏等）或扬声器（耳机、喇叭等）里，显示相应的图像画面或声音

涉及技术或协议：

显示器、扬声器、3D眼镜等

**72**、用户不登录，怎么直接加入购物车的

    用户在不登录的情况下，可以把要购买商品的信息（如商品的ID，商品的价格、商品的sku\_id,购买数量等关键数据）存到COOKIE里面，当登陆的情况下。把COOKIE里面的内容存到数据库，并清除cookie中的数据。

**73**、写过接口吗，怎么定义接口的

答案：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。

**74**、静态化如何实现的

答：这里要说的静态化指的是页面静态化，也即生成实实在在的静态文件，也即不需要查询数据库就可以直接从文件中获取数据，指的是真静态。它的实现方式主要有两种：

   一种是我们在添加信息入库的时候就生成的静态文件，也称为模板替换技术，这种主要用在后台，用于一些基本上很少变化的信息上，在添加信息的时候使用添加的信息来替换制定好的模板中的内容，达到生成静态文件的目的，这样在前台访问该信息时，可以直接从生成好的静态文件中获取信息，如一些CMS系统。

   另外一种是用户在访问我们的页面时先判断是否有对应的缓存文件存在，如果存在就读缓存，不存在就读数据库，同时生成缓存文件。这种实现的主要原理是基于PHP中的ob缓冲技术来实现的，当没有静态文件时，从数据库中读取，读取的数据使用OB缓存，使用相关的函数从OB缓冲中读取数据，写入到文件中，形成静态文件。当然这个过程中要考虑静态文件的缓存周期问题，我们可以根据文件的最后修改时间和当前时间及设定的缓存时间来定时更新缓存文件。

**75**、伪静态如何实现的

答：伪静态不是真正意义上的静态化，之所以使用伪静态，主要是为了SEO推广，搜索引擎对动态的文件获取难度大，不利于网站的推广。伪静态的实现原理主要是基于apache/nginx等web服务器的rewrite机制。利用Apache/nginx里面相关的服务器变量和指令来完成重写。主要有两种方式，一种是直接在配置虚拟机的位置配置伪静态，这个每次修改完成后需要重启web服务器。另一种采用分布式的，可以在网站的根目录上创建.htaccess的文件，在里面配置相应的重写规则来实现伪静态，这种每次重写时不需要重启web服务器，且结构上比较清晰。具体的配置可以参考 Apache/nginx的手册。

**76**、 mysql优化，读写分离如何实现？

答：mysql优化前面已经总结了。主要说下读写分离，当我们的数据量很大时，数据库服务器的压力变大，这时候我们需要从架构方面来解决这一问题，在一个网站中读的操作很多，写的操作很少，这时候我们需要配置读写分离，把读操作和写操作分离出来，最大程度的利用好数据库服务器。读写分离的实现原理就是在执行SQL语句的时候，判断到底是读操作还是写操作，把读的操作转向到读服务器上（从服务器，一般是多台），写的操作转到写的服务器上（主服务器，一般是一台，视数据量来看）。当然为了保证多台数据库数据的一致性，需要主从复制。主从复制的实现原理是：mysql中有一种日志，叫做bin日志（二进制日志），会记录下所有修改过数据库的sql语句。主从复制的原理实际是多台服务器都开启bin日志，然后主服务器会把执行过的sql语句记录到bin日志中，之后从服务器读取这个bin日志，把该日志的内容保存到自己中继日志里面，从服务器再把中继日志中记录的sql语句同样的执行一遍。这样从服务器上的数据就和主服务器相同了。

**77**、sku减库存

    SKU = Stock Keeping Unit (库存量单位)     
即库存进出计量的单位，可以是以件，盒，托盘等为单位。SKU是库存量单位，区分单品。     
在服装、鞋类商品中使用最多最普遍。 例如纺织品中一个SKU通常表示：规格、颜色、款式。

在设计表时，不仅仅只有商品表，商品表中有个总库存，我们还需要涉及一张SKU表，里面有SKU库存和单价字段，用户每购买一件商品，实际上购买的都是SKU商品，这样在下订单成功后，应该根据所购买的商品的唯一的SKU号来进行相应的SKU库存的减少，当然商品的总库存保存在商品主表中，也需要减少总库存中的库存量。

**78**、Linux查看日志

   Linux日志文件一般在/var/log目录下，通过查看日志，我们可以看到Linux系统内核和许多程序会产生各种错误信息、警告信息和提示信息，这些信息对于管理员了解系统的运营状态是非常有用的，这些信息都被保存在相应的日志文件中。syslog是一个历史悠久的日志系统，几乎所有的UNIX和Linux操作系统都是采用syslog进行系统日志的管理和配置。在默认的syslog配置下，日志文件通常都保存在“/var/log”目录下。默认配置的syslog日志如下：

我们可以通过下面几个命令分别进行查看：

1、cat命令：

     功能：1）显示整个文件。

          2）把文件串连接后传到基本输出，如将几个文件合并为一个文件或输出到屏幕。

2、more命令：

     以百分比的形式查看日志。

3、less命令：

     跟more功能差不多，只不过less支持前后翻阅文件。

4、head命令：

     功能：从文本文件的头部开始查看，head 命令用于查看一个文本文件的开头部分。

5、tail命令：

     功能：tail 命令用于显示文本文件的末尾几行。

**79**、库存设置？

    库存分为商品总库存和SKU库存，往往商品总库存的为SKU库存的总和。一般在商城的后台对货品设置最高库存及最低库存后，当前库存数量与最高、最低两者比较，超出库存或者低于库存的，则被统计成报表形式反映，便于用户掌握货品库存超、短缺状态及数量。

**80**、订单**、**库存**两个表**如何保证数据的一致性**？**

     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。

**81**、O2O  用户下单  c端下单  如何保证b a端 数据一致  O2O为线上和线下模式，　O2O模式奉行的是“线上支付+实体店消费”的消费模式，即消费者在网上下单完成支付后，凭消费凭证到实体店消费。O2O模式是把商家信息和支付程序放在线上进行，而把商品和服务兑现放在线下，也就是说O2O模式适用于[快递](http://www.boogle.cn/)无法送达的有形产品。数据一致性的问题是O2O行业中最常见的问题，我们可以类似于数据库的主从复制的思路来解决这个问题。O2O有个供应商系统，类似于主服务器，在C端（从服务器）下单时，数据同步更新到供应商系统端，b、a实时从供应商系统中拉取数据进行同步，比如利用定时任务，定时拉取数据进行同步。 **82**、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](http://lib.csdn.net/base/redis)是一种单线程机制的nosql[数据库](http://lib.csdn.net/base/mysql)，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。 **85**、 秒杀当中的细节你是怎么得出来的  
   通过性能测试及模拟秒杀场景。每个问题都经过反复测试，不断的发现问题，不断的解决。

**86**、开源产品的弊端

   很多公司为了快速的开发出一款产品，降低成本，往往会采用一些开源的产品来完成，一款好的开源产品可以起到事半功倍的效果，我们只需要在它的基础上做些二次开发，就可以达到我们想要的简单的效果。但是，往往这些开源产品也有自己的弊端。

      比如我以前用过的ecshop:产品线不完整，只有单用户产品，后台操作过于烦杂，无用功能比较多，使用起来有很多BUG和不便，系统功能相对薄弱，运营起来很费人力成本。被商派收购之后，宣传推广也停滞不前。而且是面向过程的，你必须按照它的规则进行修改，操作起来相对很繁琐。对PHP5.4以上的版本支持不好，jQuery的兼容性上要好好处理。

      PHPCMS：静态处理不好，SEO有待加强，访问速度也较慢。

      Shopex：代码不开源，修改很麻烦，只能通过插件的方式去改进，底层是不知道的，有漏洞更新也是很慢，当数量达到一定量的时候shopex很吃内存服务质量也不好，基本用户提的意见,很难被有效采取。最头疼的是他的后台操作，易用性很差，前台模板也不好看。并且系统各方面的压力承受能力都有不足。

系统出了问题也只能等待解决，有受制于人的感觉。

      总之，使用别人成型的东西，就要遵循别人的规范，责任在于自己。

**87**、做秒杀用什么数据库，怎么实现的。  
   因为秒杀的一瞬间，并发非常大，如果同时请求数据库，会导致数据库的压力非常大，导致数据库的性能急剧下降，更严重的可能会导致数据库服务器宕机。这时候一般采用内存高速缓存数据库redis来实现的,redis是非关系型数据库，redis是单线程的，通过redis的队列可以完成秒杀过程。

**88**、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。

**89**、什么是单点登录？

   单点登录SSO（Single Sign On）说得简单点就是在一个多系统共存的环境下，用户在一处登录后，就不用在其他系统中登录，也就是用户的一次登录能得到其他所有系统的信任。

什么情况下使用缓存

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2和应用系统3了。

   实现主要技术点：

   1、两个站点共用一个数据验证系统

   2、主要通过跨域请求的方式来实现验证及session处理。

**90**、怎么实现第三方登录？

    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。

**91**、压力测试是在什么环境下做的?

     环境分为开发环境和线上环境，我们做测试是在开发环境上进行测试的，只有测试通过了才会上线。

**92**、App数据是怎么来的

     APP数据是客户端传递相应的参数调用服务器端的接口，服务器端从数据库或者缓存数据从读取数据转换为XML/json数据，提供给APP端。

**93**、如何处理负载、高并发？(好好看看，经常问到，能回答到主要的东西即可)

  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、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。目前使用最为广泛的负载均衡软件是Nginx、LVS、HAProxy。我分别来说下三种的优缺点:

 Nginx的优点是：

1. 工作在网络的7层之上，可以针对http应用做一些分流的策略，比如针对域名、目录结构，它的正则规则比HAProxy更为强大和灵活，这也是它目前广泛流行的主要原因之一，Nginx单凭这点可利用的场合就远多于LVS了。

2. Nginx对网络稳定性的依赖非常小，理论上能ping通就就能进行负载功能，这个也是它的优势之一；相反LVS对网络稳定性依赖比较大，这点本人深有体会；

3. Nginx安装和配置比较简单，测试起来比较方便，它基本能把错误用日志打印出来。LVS的配置、测试就要花比较长的时间了，LVS对网络依赖比较大。

4. 可以承担高负载压力且稳定，在硬件不差的情况下一般能支撑几万次的并发量，负载度比LVS相对小些。

5. Nginx可以通过端口检测到服务器内部的故障，比如根据服务器处理网页返回的状态码、超时等等，并且会把返回错误的请求重新提交到另一个节点，不过其中缺点就是不支持url来检测。比如用户正在上传一个文件，而处理该上传的节点刚好在上传过程中出现故障，Nginx会把上传切到另一台服务器重新处理，而LVS就直接断掉了，如果是上传一个很大的文件或者很重要的文件的话，用户可能会因此而不满。

6. Nginx不仅仅是一款优秀的负载均衡器/反向代理软件，它同时也是功能强大的Web应用服务器。LNMP也是近几年非常流行的web架构，在高流量的环境中稳定性也很好。

7. Nginx现在作为Web反向加速缓存越来越成熟了，速度比传统的Squid服务器更快，可以考虑用其作为反向代理加速器。

8. Nginx可作为中层反向代理使用，这一层面Nginx基本上无对手，唯一可以对比Nginx的就只有 lighttpd了，不过 lighttpd目前还没有做到Nginx完全的功能，配置也不那么清晰易读，社区资料也远远没Nginx活跃。

9. Nginx也可作为静态网页和图片服务器，这方面的性能也无对手。还有Nginx社区非常活跃，第三方模块也很多。

Nginx的缺点是：

1. Nginx仅能支持http、https和Email协议，这样就在适用范围上面小些，这个是它的缺点。   
2. 对后端服务器的健康检查，只支持通过端口来检测，不支持通过url来检测。不支持Session的直接保持，但能通过ip\_hash来解决。

LVS：使用Linux内核集群实现一个高性能、高可用的负载均衡服务器，它具有很好的可伸缩性（Scalability)、可靠性（Reliability)和可管理性（Manageability)。

LVS的优点是：

1. 抗负载能力强、是工作在网络4层之上仅作分发之用，没有流量的产生，这个特点也决定了它在负载均衡软件里的性能最强的，对内存和cpu资源消耗比较低。   
2. 配置性比较低，这是一个缺点也是一个优点，因为没有可太多配置的东西，所以并不需要太多接触，大大减少了人为出错的几率。   
3. 工作稳定，因为其本身抗负载能力很强，自身有完整的双机热备方案，如LVS+Keepalived，不过我们在项目实施中用得最多的还是LVS/DR+Keepalived。  
4. 无流量，LVS只分发请求，而流量并不从它本身出去，这点保证了均衡器IO的性能不会受到大流量的影响。   
5. 应用范围比较广，因为LVS工作在4层，所以它几乎可以对所有应用做负载均衡，包括http、数据库、在线聊天室等等。

LVS的缺点是：

1. 软件本身不支持正则表达式处理，不能做动静分离；而现在许多网站在这方面都有较强的需求，这个是Nginx/HAProxy+Keepalived的优势所在。  
2. 如果是网站应用比较庞大的话，LVS/DR+Keepalived实施起来就比较复杂了，特别后面有 Windows Server的机器的话，如果实施及配置还有维护过程就比较复杂了，相对而言，Nginx/HAProxy+Keepalived就简单多了。

HAProxy的特点是：

1. HAProxy也是支持虚拟主机的。   
2. HAProxy的优点能够补充Nginx的一些缺点，比如支持Session的保持，Cookie的引导；同时支持通过获取指定的url来检测后端服务器的状态。   
3. HAProxy跟LVS类似，本身就只是一款负载均衡软件；单纯从效率上来讲HAProxy会比Nginx有更出色的负载均衡速度，在并发处理上也是优于Nginx的。  
4. HAProxy支持TCP协议的负载均衡转发，可以对MySQL读进行负载均衡，对后端的MySQL节点进行检测和负载均衡，大家可以用LVS+Keepalived对MySQL主从做负载均衡。  
5. HAProxy负载均衡策略非常多，HAProxy的负载均衡算法现在具体有如下8种：

① roundrobin，表示简单的轮询，这个不多说，这个是负载均衡基本都具备的；   
② static-rr，表示根据权重，建议关注；   
③ leastconn，表示最少连接者先处理，建议关注；   
④ source，表示根据请求源IP，这个跟Nginx的IP\_hash机制类似，我们用其作为解决session问题的一种方法，建议关注；   
⑤ ri，表示根据请求的URI；   
⑥ rl\_param，表示根据请求的URl参数’balance url\_param’ requires an URL parameter name；   
⑦ hdr(name)，表示根据HTTP请求头来锁定每一次HTTP请求；   
⑧ rdp-cookie(name)，表示根据据cookie(name)来锁定并哈希每一次TCP请求。

Nginx和LVS对比的总结：

1. Nginx工作在网络的7层，所以它可以针对http应用本身来做分流策略，比如针对域名、目录结构等，相比之下LVS并不具备这样的功能，所以Nginx单凭这点可利用的场合就远多于LVS了；但Nginx有用的这些功能使其可调整度要高于LVS，所以经常要去触碰触碰，触碰多了，人为出问题的几率也就会大。

2. Nginx对网络稳定性的依赖较小，理论上只要ping得通，网页访问正常，Nginx就能连得通，这是Nginx的一大优势！Nginx同时还能区分内外网，如果是同时拥有内外网的节点，就相当于单机拥有了备份线路；LVS就比较依赖于网络环境，目前来看服务器在同一网段内并且LVS使用direct方式分流，效果较能得到保证。另外注意，LVS需要向托管商至少申请多一个ip来做Visual IP，貌似是不能用本身的IP来做VIP的。要做好LVS管理员，确实得跟进学习很多有关网络通信方面的知识，就不再是一个HTTP那么简单了。

3. Nginx安装和配置比较简单，测试起来也很方便，因为它基本能把错误用日志打印出来。LVS的安装和配置、测试就要花比较长的时间了；LVS对网络依赖比较大，很多时候不能配置成功都是因为网络问题而不是配置问题，出了问题要解决也相应的会麻烦得多。

4. Nginx也同样能承受很高负载且稳定，但负载度和稳定度差LVS还有几个等级：Nginx处理所有流量所以受限于机器IO和配置；本身的bug也还是难以避免的。

5. Nginx可以检测到服务器内部的故障，比如根据服务器处理网页返回的状态码、超时等等，并且会把返回错误的请求重新提交到另一个节点。目前LVS中 ldirectd也能支持针对服务器内部的情况来监控，但LVS的原理使其不能重发请求。比如用户正在上传一个文件，而处理该上传的节点刚好在上传过程中出现故障，Nginx会把上传切到另一台服务器重新处理，而LVS就直接断掉了，如果是上传一个很大的文件或者很重要的文件的话，用户可能会因此而恼火。

6. Nginx对请求的异步处理可以帮助节点服务器减轻负载，假如使用 apache直接对外服务，那么出现很多的窄带链接时apache服务器将会占用大 量内存而不能释放，使用多一个Nginx做apache代理的话，这些窄带链接会被Nginx挡住，apache上就不会堆积过多的请求，这样就减少了相当多的资源占用。这点使用squid也有相同的作用，即使squid本身配置为不缓存，对apache还是有很大帮助的。

7. Nginx能支持http、https和email（email的功能比较少用），LVS所支持的应用在这点上会比Nginx更多。在使用上，一般最前端所采取的策略应是LVS，也就是DNS的指向应为LVS均衡器，LVS的优点令它非常适合做这个任务。重要的ip地址，最好交由LVS托管，比如数据库的 ip、webservice服务器的ip等等，这些ip地址随着时间推移，使用面会越来越大，如果更换ip则故障会接踵而至。所以将这些重要ip交给 LVS托管是最为稳妥的，这样做的唯一缺点是需要的VIP数量会比较多。Nginx可作为LVS节点机器使用，一是可以利用Nginx的功能，二是可以利用Nginx的性能。当然这一层面也可以直接使用squid，squid的功能方面就比Nginx弱不少了，性能上也有所逊色于Nginx。Nginx也可作为中层代理使用，这一层面Nginx基本上无对手，唯一可以撼动Nginx的就只有lighttpd了，不过lighttpd目前还没有能做到 Nginx完全的功能，配置也不那么清晰易读。另外，中层代理的IP也是重要的，所以中层代理也拥有一个VIP和LVS是最完美的方案了。具体的应用还得具体分析，如果是比较小的网站（日PV小于1000万），用Nginx就完全可以了，如果机器也不少，可以用DNS轮询，LVS所耗费的机器还是比较多的；大型网站或者重要的服务，机器不发愁的时候，要多多考虑利用LVS。

数据库优化

**96**、做秒杀时锁表考虑到没有？

考虑到了，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。

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

**97**、封装过一个简单的框架

封装过一个简单的MVC框架,主要分为3层，控制器层和模型层视图层，以及路由的分配和入口文件，模板引擎，单例模式、工厂模式，第三方类库的引入等。

**98**、你做过多大的压力测试

    我在开发环境下，通过ab压力和webbench做过相关的压力测试，当Apache最大到1000并发时，发现请求超时或者服务器积极拒绝。webbench并发到3000时，请求非常慢，5000时，服务器积极拒绝。

**99**、开发商城有什么前景？

商城创业服务平台，将创业者进行整合，将线下商家搬到线上，使线上商家摆脱地理位置的限制，降低推广成本，通过商城平台从线上快速引流，以全新的方式开展营销策略。线上交易数据可查，永久保存，商家通过平台交易行为加以分析，进行开展二次营销，多次营销，深度挖掘幼稚客户，培养忠实客户。

2.商城创业服务平台凭着创新的模式，将带领广大创业者咋移动互联网，O2O电商模式的大浪中杀出重围，取得成功！

**100.**谈谈对MVC的认识

  核心思想是：视图和用户交互通过事件导致控制器改变 控制器改变导致模型改变 或者控制器同时改变两者 模型改变 导致视图改变 或者视图改变 潜在的从模型里面获得参数 来改变自己。他的好处是可以将界面和业务逻辑分离。

Model（模型），是程序的主体部分，主要包含业务数据和业务逻辑。在模型层，还会涉及到用户发布的服务，在服务中会根据不同的业务需求，更新业务模型中的数据。  
View(视图），是程序呈现给用户的部分，是用户和程序交互的接口，用户会根据具体的业务需求，在View视图层输入自己特定的业务数据，并通过界面的事件交互，将对应的输入参数提交给后台控制器进行处理。  
Contorller（控制器），Contorller是用来处理用户 输入数据，已经更新业务模型的部分。控制器中接收了用户与界面交互时传递过来的数据，并根据数据业务逻辑来执行服务的调用和更新业务模型的数据和状态。

**101**、session与cookie的区别

1、cookie数据存放在第三方应用的浏览器上，session数据放在服务器上。

2、cookie不是很安全，别人可以分析存放在本地的COOKIE，进行COOKIE欺骗  
   考虑到安全应当使用session。

3、session会在一定时间内保存在服务器上。当访问增多，会比较占用你服务器的性能  
   考虑到减轻服务器性能方面，应当使用COOKIE。

4、单个cookie保存的数据不能超过4K，很多浏览器都限制一个站点最多保存20个cookie。

5、所以个人建议：  
   将登陆信息等重要信息存放为SESSION  
   其他信息如果需要保留，可以放在COOKIE

**102**、echo(),print(),print\_r()的区别

echo可以一次输出多个值，多个值之间用逗号分隔。echo是语言结构(language construct)，而并不是真正的函数，因此不能作为表达式的一部分使用。echo是php的内部指令，不是函数，无返回值。

print()：函数print()打印一个值（它的参数），如果字符串成功显示则返回true，否则返回false。只能打印出简单类型变量的值(如int,string)，有返回值

printf()：源于C语言中的printf()。该函数输出格式化的字符串。

[print\_r](https://www.baidu.com/s?wd=print_r&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9myuhmhmYm1bkuyD4mycY0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1fsnWn3rHn4)()和[var\_dump](https://www.baidu.com/s?wd=var_dump&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9myuhmhmYm1bkuyD4mycY0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3En1fsnWn3rHn4)()  
print\_r()可以把字符串和数字简单地打印出来，而数组则以括起来的键和值得列表形式显示，并以Array开头。但print\_r()输出布尔值和NULL的结果没有意义，因为都是打印"\n"。因此用var\_dump()函数更适合调试。print\_r是函数，可以打印出比较复杂的变量(如数组，对象)，有返回值  
var\_dump()判断一个变量的类型与长度,并输出变量的数值,如果变量有值输的是变量的值并回返数据类型。此函数显示关于一个或多个表达式的结构信息，包括表达式的类型与值。数组将递归展开值，通过缩进显示其结构。

**103**、 说一下单引号双引号？

  ①单引号内部的变量不会执行， 双引号会执行

  ②单引号解析速度比双引号快。

  ③单引号只能解析部分特殊字符，双引号可以解析所有特殊字符。

**104**、索引的优缺点

1、优点：

a）可以保证数据库表中每一行的数据的唯一性

b）可以大大加快数据的索引速度

c）加速表与表之间的连接，物别是在实现数据的参考完事性方面特别有意义

d）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

f）通过使用索引，可以在时间查询的过程中，使用优化隐藏器，提高系统的性能

2、缺点：

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

索引需要占物理空间，除了数据表占用数据空间之外，每一个索引还要占用一定的物理空间，如果需要建立聚簇索引，那么需要占用的空间会更大

以表中的数据进行增、删、改的时候，索引也要动态的维护，这就降低了整数的维护速度

建立索引的原则

在经常需要搜索的列上，可以加快搜索的速度

在作为主键的列上，强制该列的唯一性和组织表中数据的排列结构

在经常用在连接的列上，这些列主要是一外键，可以加快连接的速度

在经经常需要根据范围进行搜索的列上创建索引，国为索引已经排序，其指定的范围是连续的

在经常需要排序的列上，国为索引已经排序，这样井底可以利用索引的排序，加快排序井底时间

在经常使用在where子句中的列上，加快条件的判断速度

**105.get**和post的区别

1. get是从服务器上获取数据，post是向服务器传送数据。  
2. get是把参数数据队列加到提交表单的ACTION属性所指的URL中，值和表单内各个字段一一对应，在URL中可以看到。post是通过HTTP post机制，将表单内各个字段与其内容放置在HTML HEADER内一起传送到ACTION属性所指的URL地址。用户看不到这个过程。  
3. get传送的数据量较小，不能大于2KB。post传送的数据量较大，一般被默认为不受限制。4.. get安全性非常低，post安全性较高。但是执行效率却比Post方法好。

**106.**如何修改session的生存时间

一：在[php.ini](https://www.baidu.com/s?wd=php.ini&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1d9uHN-PhcYrH9buj63nA7-0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPHnLPjm3PHcd) 中设置 session.gc\_maxlifetime = 1440 //默认时间  
二：代码实现  
$lifeTime = 24 \* 3600;  // 保存一天   
session\_set\_cookie\_params($lifeTime);   
session\_start();

**107.PHP**版本差别

PHP5.2 以前：autoload, PDO 和 MySQLi, 类型约束  、JSON 支持  
PHP5.3：弃用的功能，匿名函数，新增魔术方法，命名空间，后期静态绑定Heredoc 和 Nowdoc, const, 三元运算符，Phar  
PHP5.4：Short Open Tag, 数组简写形式，Traits, 内置 Web 服务器，细节修改  
PHP5.5：yield, list() 用于 foreach, 细节修改  
PHP5.6: 常量增强，可变函数参数，命名空间增强

**108.**对称加密 与 非对称加密

对称加密与解密使用的是同样的密钥，但由于需要将密钥在网络传输，所以安全性不高

非对称加密使用了一对密钥，公钥与私钥，把以安全性高，但加密与解密速度慢

解决的办法是将对称加密的密钥使用非对称加密的公钥进行加密，然后发送出去，接收方使用私钥进行解密得到对称加密的密钥，然后双方可以使用对称加密来进行沟通

（一）对称加密（Symmetric Cryptography）

对称加密是最快速、最简单的一种加密方式，加密（encryption）与解密（decryption）用的是同样的密钥（secret key）,这种方法在密码学中叫做对称加密算法。

   采用单钥[密码系统](http://baike.baidu.com/view/3114489.htm)的加密方法，同一个[密钥](http://baike.baidu.com/view/934.htm)可以同时用作信息的加密和解密加密方式[DES](http://baike.baidu.com/view/7510.htm)、[3DES](http://baike.baidu.com/view/350958.htm)、TDEA、[Blowfish](http://baike.baidu.com/view/2208941.htm)、RC2、RC4、[RC5](http://baike.baidu.com/view/734720.htm)、[IDEA](http://baike.baidu.com/view/92629.htm)、SKIPJACK、AES

优缺点

对称加密算法的优点是算法公开、计算量小、加密速度快、加密效率高。

对称加密算法的缺点是在数据传送前，发送方和接收方必须商定好秘钥，然后 使双方都能保存好秘钥。其次如果一方的秘钥被泄露，那么加密信息也就不安全了。另外，每对用户每次使用对称加密算法时，都需要使用其他人不知道的唯一秘 钥，这会使得收、发双方所拥有的钥匙数量巨大，密钥管理成为双方的负担。

（二）非对称加密（asymmetric Cryptography）

允许在不安全的媒体上的通讯双方交换信息，安全地达成一致的密钥，这就是“公开密钥系统”。相对于“对称加密算法”这种方法也叫做“非对称加密算法”。

[非对称加密算法](http://baike.baidu.com/view/1490349.htm)需要两个[密钥](http://baike.baidu.com/view/934.htm)来进行加密和解密

优缺点

  非对称加密与对称加密相比，其安全性更好：对称加密的通信双方使用相同的秘钥，如果一方的秘钥遭泄露，那么整个通信就会被破解。而非对称加密使用一对秘钥，一个用来加密，一个用来解密，而且公钥是公开的，秘钥是自己保存的，不需要像对称加密那样在通信之前要先同步秘钥。

  非对称加密的缺点是加密和解密花费时间长、速度慢，只适合对少量数据进行加密。

**109.ob**系列函数

1、ob\_start() //打开缓冲区，所有输出的信息不直接发送到浏览器，而是保存在缓冲区里面

2、ob\_clean() //删除内部缓冲区的内容，不关闭缓冲区（不输出）

3、ob\_end\_clean() //删除内部缓冲区的内容，关闭缓冲区（不输出）

4、ob\_get\_clean() //返回内部缓冲区的内容，关闭缓冲区。【相当于执行ob\_get\_contents() and ob\_end\_clean()】

5、ob\_flush() //发送内部缓冲区的内容到浏览器，删除缓冲区的内容，不关闭缓冲区。

6、ob\_end\_flush() //发送内部缓冲区的内容到浏览器，删除缓冲区的内容，关闭缓冲区

7、ob\_get\_flush() //返回内部缓冲区的内容，并关闭缓冲区的内容

8、ob\_get\_contents()//返回缓冲区的内容，不输出

9、ob\_get\_length() //返回内部缓冲区的长度，如果缓冲区未被激活，该函数返回false

**110.Linux**基本命令，目录结构

|  |  |
| --- | --- |
| arch 显示机器的处理器架构(1)  uname -m 显示机器的处理器架构(2)  uname -r 显示正在使用的内核版本  dmidecode -q 显示硬件系统部件 - (SMBIOS / DMI)  hdparm -i /dev/hda 罗列一个磁盘的架构特性  hdparm -tT /dev/sda 在磁盘上执行测试性读取操作  cat /porc/cpuinfo 显示CPU info的信息  cat /porc/interrupts 显示中断  cat /porc/meminfo 校验内存使用  cat /porc/swaps 显示哪些swap被使用  cat /porc/verion 显示内核的版本  cat /porc/net/dev 显示网络适配器及统计  cat /porc/mounts 显示已加载的文件系统   date 显示系统日期  cal 2007 显示2007年的日历表  date 041217002007.00 设置日期和时间 -月日时分年.秒  clock -w 将时间修改保存到 BIOS  文件搜索  find / -name file1 从 '/'开始进入根文件系统搜索文件和目录  locate \\*.ps 寻找以 '.ps'结尾的文件 -先运行 'updatedb'命令  whereis halt 显示一个二进制文件、源码或man的位置  which halt 显示一个二进制文件或可执行文件的完整路径    挂载一个文件系统  mount /dev/hda2 /mnt/hda2 挂载一个叫做hda2的盘- 确定目录'/ mnt/hda2' 已经存在  umount /dev/hda2 卸载一个叫做hda2的盘- 先从挂载点'/ mnt/hda2' 退出  追加命令  1,linux里把文件/etc/aaa中的内容追加到/usr/bbb中的内容的后面   sudo cat /etc/aaa >>/usr/bbb 2,更改/etc/index.html的文件所有者为apache,文件群组为apache   sudo chmod apache:apache  /etc/index.html 3,更改/etc/index.html的所有者权限为读取、写入、执行。群组权限为读取。其他权限为读取   sudo chmod 744 /etc/index.html 4，删除/etc下名为hello的文件   sudo rm /etc/index.html  当然，如果你是以orot用户执行以上操作，可以去掉前边的sudo！    df -hl 查看磁盘剩余空间  df -h 查看每个根路径的分区大小  du -sh [目录名] 返回该目录的大小  du -sm [文件夹] 返回该文件夹总M数 | 关机 (系统的关机、重启以及登出)  shutdown -h now 关闭系统(1)  init 0 关闭系统(2)  telinit 0 关闭系统(3)  shutdown -h hour:minutes & 按预定时间关闭系统  shutdown -c 取消按预定时间关闭系统  shutdown -r now 重启(1)  reboot 重启(2)  logout 注销  文件和目录  pwd 显示工作路径  ls 查看目录中的文件  ls -F 查看目录中的文件  ls -l 显示文件和目录的详细资料  ls -a 显示隐藏文件  ls \*[0-9]\* 显示包含数字的文件名和目录名  tree 显示文件和目录由根目录开始的树形结构(1)  lstree 显示文件和目录由根目录开始的树形结构(2)  mkdir dir1 创建一个叫做 'dir1'的目录'  磁盘空间  df -h 显示已经挂载的分区列表  ls -lSr |more 以尺寸大小排列文件和目录  du -sh dir1 估算目录 'dir1'已经使用的磁盘空间'  下载、解压  1)对于.tar结尾的文件  　　tar -xf all.tar  2)对于.gz结尾的文件  　　gzip -d all.gz  　　gunzip all.gz  # zip all.zip \*.jpg  　　这条命令是将所有.jpg的文件压缩成一个zip包  # unzip all.zip  　　这条命令是将all.zip中的所有文件解压出来  [下载](http://download.chinaitlab.com/)命令  wget + 空格 +要[下载](http://download.chinaitlab.com/)文件的url路径  =====================================  Shell 脚本：  必须以  #!/bin/sh  开头  简单例子：判断这个目录下有没有文件（File）  #!/bin/bash Num=`[ls](https://www.baidu.com/s?wd=ls&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3nHb3uHwBn16LmHckmHmd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH01nHb1rjm3) -al /opt |grep "^-"|wc [-l](https://www.baidu.com/s?wd=-l&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3nHb3uHwBn16LmHckmHmd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH01nHb1rjm3) ` if [ $Num != 0 ]  then echo "/opt has $Num files" else  echo "/opt has none file" fi  [ls](https://www.baidu.com/s?wd=ls&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3nHb3uHwBn16LmHckmHmd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH01nHb1rjm3) -al /opt |grep "^-"|wc [-l](https://www.baidu.com/s?wd=-l&tn=44039180_cpr&fenlei=mv6quAkxTZn0IZRqIHckPjm4nH00T1Y3nHb3uHwBn16LmHckmHmd0ZwV5Hcvrjm3rH6sPfKWUMw85HfYnjn4nH6sgvPsT6KdThsqpZwYTjCEQLGCpyw9Uz4Bmy-bIi4WUvYETgN-TLwGUv3EPH01nHb1rjm3)  这个命令能够统计文件个数 为0就是没有文件 非零就是有文件 |

**111.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

**112.**魔术方法、魔术常量

1。\_\_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。

2。\_\_destruct()

当删除一个对象或对象操作终止时被调用。

3。\_\_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。

4。\_\_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。

5。\_\_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。

6。\_\_toString()

打印一个对象的时被调用。如echo $obj;或print $obj;

7。\_\_clone()

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

8。\_\_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。

9。\_\_wakeup()

unserialize时被调用，做些对象的初始化工作。

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

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

12。\_\_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。

13。\_\_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。

魔术常量：

1。\_\_LINE\_\_  
返回文件中的当前行号。

2。\_\_FILE\_\_  
返回文件的完整路径和文件名。如果用在包含文件中，则返回包含文件名。自 PHP 4.0.2 起，\_\_FILE\_\_ 总是包含一个绝对路径，而在此之前的版本有时会包含一个相对路径。

3。\_\_FUNCTION\_\_  
返回函数名称（PHP 4.3.0 新加）。自 PHP 5 起本常量返回该函数被定义时的名字（区分大小写）。在PHP 4 中该值总是小写字母的。

4。\_\_CLasS\_\_  
返回类的名称（PHP 4.3.0 新加）。自 PHP 5 起本常量返回该类被定义时的名字（区分大小写）。在PHP 4中该值总是小写字母的。

5。\_\_METHOD\_\_  
返回类的方法名（PHP 5.0.0 新加）。返回该方法被定义时的名字（区分大小写）。

\_\_set()当程序试图写入一个不存在或者不可见的成员变量时，\_\_set()方法包含两个参数，分别表示变量名称和变量值，两个参数都不可省略

\_\_get()当程序试图调用一个未定义或不可见的成员变量时，\_\_get()方法有一个参数，表示要调用的变量名

\_\_sleep() 常用于提交未提交的数据，或类似的清理操作如果有一些很大的对象，但不需要全部保存，这个功能就很好用。

\_\_construct()  在类实例化对象的同时执行该函数

\_\_distruct() 在类实例化的对象销毁时执行

\_\_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。

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

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

\_\_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)。

\_\_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。

**113.**创建索引的原则

索引是建立在数据库表中的某些列的上面。因此，在创建索引的时候，应该 仔细考虑在哪些列上可以创建索引，在哪些列上不能创建索引。一般来说，应该在这些列上创建索引， 例如：在经常需要搜索的列上，可以加快搜索的速度；在作为 主键的列上，强制该列的唯一性和组织表中数据的排列结构；在经常用在连接的列上，这些列主要是一些外键，可以加快连接的速度；在经常需要根据范围进行搜索的列上创建索引，因为索引已经排序，其指定的范围是连续的；在经常需要排序的列上创建索引，因为索引已经排序，这样查询可以利用索引的排序，加快排序查询时间；在经常使用在WHERE子句中的列上面创建索引，加快条件的判断速度。

同样，对于有些列不应该创建索引。一般来说，不应该创建索引的的 这些列具有下列特点：

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。

**114.http**的三次握手？

答：三次握手（three times handshake; three-way handshake）所谓的“三次握手”即对每次发送的数据量是怎样跟踪进行协商使数据段的发送和接收同步，根据所接收到的数据量而确定的数据确认数及数据发送、接收完毕后何时撤消联系，并建立虚连接。

TCP/IP协议中，TCP协议提供可靠的连接服务，采用三次握手建立一个连接。

（1）第一次握手：建立连接时，第三方应用端A发送SYN包（SYN=j）到服务器B，并进入SYN\_SEND状态，等待服务器B确认。

（2）第二次握手：服务器B收到SYN包，必须确认第三方应用A的SYN（ACK=j+1），同时自己也发送一个SYN包（SYN=k），即SYN+ACK包，此时服务器B进入SYN\_RECV状态。

（3）第三次握手：第三方应用端A收到服务器B的SYN＋ACK包，向服务器B发送确认包ACK（ACK=k+1），此包发送完毕，第三方应用端A和服务器B进入ESTABLISHED状态，完成三次握手。

完成三次握手，第三方应用端与服务器开始传送数据

**115**、解释一下OOP？面向对象

OOP具有三大特点

1、封装性：也称为信息隐藏，就是将一个类的使用和实现分开，只保留部分接口和方法与外部联系，或者说只公开了一些供开发人员使用的方法。于是开发人员只 需要关注这个类如何使用，而不用去关心其具体的实现过程，这样就能实现MVC分工合作，也能有效避免程序间相互依赖，实现代码模块间松藕合。

2、继承性：就是子类自动继承其父级类中的属性和方法，并可以添加新的属性和方法或者对部分属性和方法进行重写。继承增加了代码的可重用性。PHP只支持单继承，也就是说一个子类只能有一个父类。

3、多态性：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，这种技术就是多态性。多态性增强了软件的灵活性。

1、易维护

采用面向对象思想设计的结构，可读性高，由于继承的存在，即使改变需求，那么维护也只是在局部模块，所以维护起来是非常方便和较低成本的。

2、质量高

在设计时，可重用现有的，在以前的项目的领域中已被测试过的类使系统满足业务需求并具有较高的质量。

3、效率高

在软件开发时，根据设计的需要对现实世界的事物进行抽象，产生类。使用这样的方法解决问题，接近于日常生活和自然的思考方式，势必提高软件开发的效率和质量。

4、易扩展

由于继承、封装、多态的特性，自然设计出高内聚、低耦合的系统结构，使得系统更灵活、更容易扩展，而且成本较低。

**116.cookie**伪造

你的第三方应用端的cookie被恶意的用户截取到,然后向服务器端发送,并且通过验证,他们就会冒充用户进行登录,这就是cookie伪造

防cookie伪造：

现在更通用的做法是使用[session](http://zh.wikipedia.org/wiki/%E4%BC%9A%E8%AF%9D)来标识用户，也就是说我们为每个第三方应用端生成一个唯一的id，然后在服务端存储这个id所对应的状态。

这样cookie里面仅仅保存了这个id，而没有任何其他的东西。而且这个id往往还有个特性，它是随机生成，且每次登陆都会产生一个新的。这样就更降低了信息泄漏的风险。

**117.PHP**的垃圾收集机制是怎样的

ＰＨＰ作为脚本语言是页面结束即释放变量所占内存的。 当一个 PHP线程结束时，当前占用的所有内存空间都会被销毁，当前程序中所有对象同时被销毁。GC进程一般都跟着每起一个SESSION而开始运行的.gc目的是为了在session文件过期以后自动销毁删除这些文件.在PHP中，没有任何变量指向这个对象时，这个对象就成为垃圾。PHP会将其在内存中销毁；这是PHP的GC垃圾处理机制，防止内存溢出。 执行这些函数也可以起到回收作用\_\_destruct /unset/mysql\_close /fclose php对session有明确的gc处理时间设定session.gc\_maxlifetime 如果说有垃圾，那就是整体的程序在框架使用中，会多次调用同一文件等等造成的非单件模式等。所以在出来的时候，必要的用\_once引用，在声明类的时候使用单件模式。还有简化逻辑等等。

**118.**接口和抽象类的区别是什么？

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。

**119.Session**相关问题

session共享：

1.各种web框架早已考虑到这个问题，比如asp.net，是支持通过配置文件修改session的存储介质为sql server的，所有机器的会话数据都从同一个数据库读，就不会存在不一致的问题；

2.以cookie加密的方式保存在第三方应用端.优点是减轻服务器端的压力，缺点是受到cookie的大小限制，可能占用一定带宽，因为每次请求会在头部附带一定大小的cookie信息,另外这种方式在用户禁止使用cookie的情况下无效.

3.服务器间同步。定时同步各个服务器的session信息，此方法可能有一定延时，用户体验也不是很好。

php支持把会话数据存储到某台memcache服务器，你也可以手工把session文件存放的目录改为nfs网络文件系统，从而实现文件的跨机器共享。

session运行原理：

（1）当一个session第一次被启用时，一个唯一的标识被存储于本地的cookie中。

（2）首先使用session\_start()函数，PHP从session仓库中加载已经存储的session变量。

（3）当执行PHP脚本时，通过使用session\_register()函数注册session变量。

（4）当PHP脚本执行结束时，未被销毁的session变量会被自动保存在本地一定路径下的session库中，这个路径可以通过php.ini文件中的session.save\_path指定，下次浏览网页时可以加载使用。

为什么session依赖cookie：

当用户请求servlet，servlet会首先查看第三方应用端cookie中是否有sessionID，如果有则证明是旧的会话，那么就通过cookie将sessionID发送到服务器，服务器就会根据sessionID到服务器的内存中查找session对象（因为每个session都会有一个sessionID来标识session对象），找到之后然后使用。

如果cookie中没有sessionID这证明是一个新的会话。服务器就会创建一个新的Session对象，然后将SessionID存放早cookie中，通过cookie把sessionID发送到第三方应用端。第三方应用端下一次访问的时候，就会将SessionID发送到服务器以便再次找到这个session对象，完成会话跟踪所以如果用户将cookie关闭session也将会失效。session是依赖与cookie的。

与cookie的区别与联系：cookie在第三方应用端保存用户的信息，而session在服务器上保存第三方应用的信息

session依赖于cookie。如果用户关闭cookie，则session失效，原因是sessionID无法从第三方应用端传递到服务端，也不能从服务端传递到第三方应用端.

session怎么设置过期时间：

第一种方法即设置php.ini配置文件，设置session.gc\_maxlifetime和session.cookie\_lifetime节点属性值

第二种方法即设置Session时间戳

**120.**字符串反转

给定字符串abcdef,写出反转函数,将字符串反转为fedcba.

function myStrReve($str){

    $len = strlen($str);

    $result = '';

    for($i = $len - 1; $i >=0 ; $i-- ){

        $result .= $str[$i];

    }

    return $result;

}

**121.GD**库是做什么用的?gd库提供了一系列用来处理图片的API，使用GD库可以处理图片，或者生成图片。   
在网站上GD库通常用来生成缩略图或者用来对图片加水印或者对网站数据生成报表。

**122.COOKIE**、SESSION的联系和区别，多台web服务器如何共享SESSION？

使用session\_start()调用session，服务器端在生成session文件的同时，生成sessionID哈希值和默认值为PHPSESSID的sessionname，并向第三方应用端发送变量为(默认的是)PHPSESSID(sessionname)，值为一个128位的哈希 值。服务器端将通过该cookie与第三方应用端进行交互。

session变量的值经PHP内部系列化后保存在服务器机器上的文本文件中，和第三方应用端的变量名默认情况下为PHPSESSID的cookie进行对应交 互，即服务器自动发送了HTTP头：header('Set- Cookie：session\_name()=session\_id()；path=/')；即setcookie(session\_name(),session\_id())；当从该页跳转到的新页面并调用session\_start()后，PHP将检查与给定ID相关联的服务器端存贮的session数据，如果没找到，则新建一个数据集。

**123**、什么是队列？排它锁，Myisam死锁如何解决？

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

**124**、打开php.ini中的safe\_mode，会影响哪些函数？至少说出6个。

1)用户输入输出函数（fopen()file()require(),只能用于调用这些函数有相同脚本的拥有者）

2)创建新文件（限制用户只在该用户拥有目录下创建文件）

3)用户调用popen()systen()exec()等脚本，只有脚本处在safe\_mode\_exec\_dir配置指令指定的目录中才可能

4)加强HTTP认证，认证脚本拥有者的UID的划入认证领域范围内，此外启用安全模式下，不会设置PHP\_AUTH

5)mysql服务器所用的用户名必须与调用mysql\_connect()的文件的拥有者用户名相同6)

受影响的函数变量以及配置命令达到40个

**125.Smarty**的原理

smarty是个模板引擎，最显著的地方就是有可以把模板缓存起来。一般模板来说，都是做一个静态页面，然后在里面把一些动态的部分用一切分隔符切开，然后在PHP里打开这个模板文件，把分隔符里面的值替换掉，然后输出来，你可以看下PHPLib里面的template部分。

而smarty设定了缓存参数以后，第一运行时候会把模板打开，在php替换里面值的时候把读取的html和php部分重新生成一个临时的php文件，这样就省去了每次打开都重新读取html了。如果修改了模板，只要重新刷下就行了。

**126**、写出五种以上你使用过的PHP的扩展的名称（提示：常用的PHP扩展）

 mb\_sring、iconv、curl、GD、XML、socket、MySQL、PDO等

**127**、怎么把文件保存到指定目录？怎么避免上传文件重名问题？

  可以通过上传的文件名获取到文件后缀，然后使用时间戳+随机数+文件后缀的方式为文件重新命名，这样就避免了重名。可以自己设置上传文件的保存目录，与文件名拼凑形成一个文件路径，使用move\_uploaded\_file()，就可以完成将文件保存到指定目录。

**128**、在Smarty模板语法中如何获取php的全局环境变量

$smarty.get.变量#显示通过get方式传过来的指定变量的值

$smarty.post.变量#显示通过post方式传过来的指定变量的值$smarty.cookies.变量#显示通过cookie中指定变量的值

$smarty.server.SERVER\_NAME#显示server变量值，$\_SERVER系列变量$smarty.env.PATH#显示系统环境变量值，$\_ENV系列变量$smarty.session.变量#显示session中指定变量的值

$smarty.request.变量#显示通过post、get、cookie中指定变量的值

**129**、禁用COOKIE后SEESION还能用吗?

    可以，Cookie和session都是用来实现会话机制的，由于http协议是无状态的，所以要想跟踪一个用户在同一个网站之间不同页面的状态，需要有这么一个机制----会话机制。

Cookie：将会话信息的保存到浏览器端。Session：将会话信息保存到服务器端。

session默认情况下是基于cookie的，对于session来说，每生成一个sessionid，都会将其发送到浏览器端，让后将其保存到cookie当中。

如果禁用了cookie，则基于cookie的session不好使了，我们可以使用get，传递SID。

**130**、PHP7有哪些新特性？

PHP7在PHP5的基础上又做了一次质的提升，当然改变很多，我这里以我的总结简单说下，主要发生了下面这些更改：

移除了一些旧的特性

ZEND引擎升级到Zend Engine 3，也就是所谓的PHP NG

增加抽象语法树，使编译更加科学

64位的INT支持

统一的变量语法

原声的TLS - 对扩展开发有意义

一致性foreach循环的改进

新增 <=>、\*\*、??、\u{xxxx}操作符

增加了返回类型的声明

增加了标量类型的声明

核心错误可以通过异常捕获了

增加了上下文敏感的词法分析

**131**、什么是JS中的闭包，什么是PHP中的闭包函数？

**JS**的闭包：

      所谓“闭包”，指的是一个拥有许多变量和绑定了这些变量的环境的表达式（通常是一个函数），因而这些变量也是该表达式的一部分。

   先看下下面这段代码

这段代码有两个特点：1、函数b嵌套在函数a内部；2、函数a返回函数b。

这样在执行完var c=a()后，变量c实际上是指向了函数b，再执行c()后就会弹出一个窗口显示i的值(第一次为1)。这段代码其实就创建了一个闭包，为什么？因为函数a外的变量c引用了函数a内的函数b，就是说：  
  当函数a的内部函数b被函数a外的一个变量引用的时候，就创建了一个闭包。

①这里首先得说下JS的垃圾回收机制：

在Javascript中，如果一个对象不再被引用，那么这个对象就会被GC回收。如果两个对象互相引用，而不再被第3者所引用，那么这两个互相引用的对象也会被回收。因为函数a被b引用，b又被a外的c引用，这就是为什么函数a执行后不会被回收的原因。

②闭包有什么作用呢？

    1）可以在全局作用域实现对局部变量的引用

    2）可以一直保存我们的变量或函数驻留在内存中，而不会被GC回收

 ③闭包的应用场景

   1、保护函数内的变量安全。以最开始的例子为例，函数a中i只有函数b才能访问，而无法通过其他途径访问到，因此保护了i的安全性。  
  2、在内存中维持一个变量。依然如前例，由于闭包，函数a中i的一直存在于内存中，因此每次执行c()，都会给i自加1。

我眼中的闭包：

   函数中函数，且该函数捆绑了一些局部变量，又由于全局变量的引用，会导致函数与变量都不会被回收，这就是我眼中的闭包。

**PHP**中的闭包函数：

在PHP5.3以后，允许创建匿名函数，中匿名函数，也叫闭包函数（closures ），允许 临时创建一个没有指定名称的函数。最经常用作回调函数（callback）的参数。

用好闭包，可以帮我们

1 减少foreach的循环的代码

2 减少函数的参数

3 解除递归函数

PHP异步处理   redis的异步处理    数据库的死锁、悲观锁/乐观锁、排他锁/共享锁、脏读、幻读、API优化/设计、框架底层、主从延迟、redis集群、