## 1、防止重复提交（例如账单的提交）？

解决方案：设置账单编号，只有在规定的时间，第一次提交时，才接受

## 2、在哪些情况下比较适合使用分布式缓存?

（1）、海量数据查询

（2）、访问的集中

（3）、数据增、删、改频繁

通过缓存数据库查询结果，减少数据访问次数，提高可扩展性。

分布式存储有利于减少其查询范围，提高访问速率；MemCached通过Hash运算，通过哈希值快速找到其位置。

MemCached主要有两点优势：

（1）、高性能，每次查询先从缓存中寻找，找到则直接返回结果；否则，从数据中查询结果并保存到内存中。

（2）、分布式，Memcached通过Hash运算到对应位置查询或缓存其查询结果

（3）、多线程

但是，Memcached只支持k/v类型数据，不支持list，set等数据结果存储，这一点不如redis；还有Memcached由于将数据存入缓存中，因此不利于数据的持久化；一旦及其发生故障，缓存中所有数据将会丢失。

## 3、防止用户数据库表被盗导致用户密码泄露？

解决方案：将用户密码与其他字符串进行一定的运算，在进行MD5加密

## 4、用户登陆周期长，大数据处理时？

解决方案：

（1）、使用memcached缓存数据

（2）、对于有几亿条数据时但只有百分之几十的数据经常用，可以新建一个表，将经常使用的数据放到这张表中，首先先从新建表中查询，然后再从其他表中查询

（3）、根据用户名的首字母（或先经过哈希运算得到哈希值）将其放入不同的表中

（4）、使用log4j日志记录排除错误

## 5、什么情况下使用网络组件？

（1）、不希望其他人知道源代码，保密性高

（2）、跨语言，不同语言之间的访问

## 6、不同公司之间登录合作？例如其他公司使用QQ账号登录？

使用OAuth V2.0 验证方法

## 7、在程序数据库操作中为什么使用消息队列？

主要原因是由于在高并发环境下，由于来不及同步处理，请求往往会发生堵塞，比如说，大量的insert，update之类的请求同时到达mysql，直接导致无数的行锁表锁，甚至最后请求会堆积过多，从而触发too many connections错误。通过使用消息队列，我们可以异步处理请求，从而缓解系统的压力。

## 8、高并发的解决方法

一种是使用缓存、另一种是使用生成静态页面；还有就是从最基础的地方优化我们写代码减少不必要的资源浪费：(

1.不要频繁的new对象,对于在整个应用中只需要存在一个实例的类使用单例模式.对于String的连接操作,使用StringBuffer或者StringBuilder.对于utility类型的类通过静态方法来访问。

2.避免使用错误的方式,如Exception可以控制方法推出,但是Exception要保留stacktrace消耗性能,除非必要不要使用instanceof做条件判断,尽量使用比的条件判断方式.使用JAVA中效率高的类,比如ArrayList比Vector性能好。)

**HTML静态化、图片服务器分离、数据库集群、库表散列、缓存、负载均衡**

## 9、各个子系统之间http api请求？

设置appCode、salt，设置密钥MD5加密之前和请求的路径加密之后是否一致。