翻页Pagination操作只针对list方法，其它的方法不需要翻页。

为什么要使用cache，因为cache的访问速度快，Memcached所有数据都存放在内存中，显然访问内存的速度比访问硬盘的速度要快很多。

数据库本身会自带一些cache，但是无法控制那些数据cache，那些不cache，另外数据库为了保证数据一致性，会导致cache hit rate即命中率(就是访问某个数据中从缓存中访问到的概率)不一定很高，而且数据库cache会有容量限制，这样就不可控。

cache访问速度和数据库访问速度基本是百倍到千倍的差距。

Facebook曾对Memcached进行了优化，其中有一个机制是防止“缓存穿透”，所谓缓存穿透指的是如果有大量的query都同时去访问cache中的某个key，但该key不存在或失效了，这样大量的query都会去访问DB，这就是缓存穿透，会导致DB挂掉，为此Facebook设计了一个机制，只让第一个query去访问DB并回填到memcached中，其它query等待一个很短的时间后，再去访问cache，这样通常就有了key。

TTL(Time to Live)，存活时间，配置中是TIMEOUT，单位是秒，也就是缓存中某个key自动的存活时间，超过这个时间后，缓存会自动失效。

删除key同样会带来数据不一致的问题，删除key后，会重新从数据库读取数据库并回填到cache中，如果这时有request更新了数据库，同样会造成数据不一致的问题。

当某个key对应的内容发生变化，删除key带来的数据不一致的问题远低于修改key。

对于社交系统而言，数据不一致性通常是通过TIMEOUT来解决。

Memcached通常适用于社交Web系统，不适用于金融系统，金融系统对数据一致性要求很高。

如果希望某个key常驻在Memcached中，应该怎么处理？

Memcached和web服务器通常不在一台服务器上，如果加锁需要加分布式锁，不建议使用分布式锁，效率非常低，原本使用Memcached就是为了提高效率。

Memcached更适合缓存一个一个的对象，User信息不会经常改变，例如你的用户名和头像不会经常改变，在真实系统中，cache hit rate往往高达98%以上，对User表单的请求，98%都可以使用cache，不需要去访问数据库。

Django的代码中，models.py一般是底层的代码，它不要再去依赖其他的代码，像services.py, serializers.py都是去依赖models.py。

Model中的方法一是不需要其他参数，只和它自身有关，二是代码量不要太长，如果太长就放到Service中。