1. **大致模式有几种**

整体就三个问题，怎么何时缓存，缓存怎么存，读取是否要穿透。

有个关键问题，缓存的目的是减少数据库压力并顺便加速。同时需要缓存的东西有三种情况：

一种是在背后代码某些验证、运算时，从缓存读，这种情况，一般都是变更很不频繁的简单的关系数据库映射，如动态参数设定表。这种情况只要后台跑个东西，定期更新一下就行了。

还有一种是用于特殊用途，不仅仅是作为缓存了，比如生成唯一编号功能。比如处理接口异步队列。 这些个别的功能，需要特殊设计，已不再是缓存范畴。

另一个是用户看到的数据，这种数据的的处理。 当关联业务发生时，只在缓存记录下，某用户的某个业务变更了，哪些查看模块分别需要更新缓存了。 当试图调用这些模块获取数据的接口时， 若发现这个人的这个模块被标记成了需要更新，那么就把这次取到数据缓存下来，只要没有再被标记更新，下次访问就优先从缓存读取数据。

但是这种模式中，如果用户展示数据中，有不受自己控制的，比如xxx最低价啥的，只能在关键业务点进行同步缓存更新。

**通知更新集合：** S:CUN:登陆账号 -> a模块,b模块,c模块

含义，某登陆账号，abc三个模块需要更新了。

**个人数据缓存散列：** H:CUC:登陆账号 -> filed(模块名) value(缓存内容)

含义，某登陆账号，某模块当前的缓存内容。

**当更新点触发时：**

1.（可视结果类型）若该触发点触发后，则使用SADD S:CUN:登陆账号 a模块,b模块 命令, 将a,b模块名追加到集合（相同模块名不会重复插入）

2.（内部变量类型）直接调用“更新类库”操作对应的映射更新缓存。不存在任何通知的概念。

原来获取a模块数据的接口内，首先根据当前用户，调用SREM命令，

如SREM S:CUN:登陆账号 a模块 ， 命令会将a模块删除，返回成功删除的数量。

若命令返回值为1，代表此模块需要更新， 否则，代表不用更新直接从缓存读取就行了。

若需要更新，正常执行原有代码得到数据，成功得到数据后，将数据更新到“个人数据缓存散列”中。使用命令如: HSET H:CUC:登陆账号 a模块 缓存内容

若不需要更新，直接从“个人数据缓存散列”中获得数据，缓存获取失败时，才执行原有代码得到数据。

规则规范

1、键名一律采用: 类型：数据表或菜单名：数据 的规则。如S:PTJRSDB:2014，代表平台设定的2014年的假期信息，是使用集合类型存储的。

其中，类型表示分别为：str-字符串类型；H-hash类型；L-列表类型（list），S-集合类型；Z-有序集合类型。

命名的基本原则是通过键名可以看出在redis中采用的存储类型，以及键值包含的基本内容是什么。

**二． Redis操作类库二次封装**

1. 封装在“FMDBHelperClass”中的“RedisClass”类中。 全静态，RedisClient必须要存到静态字典中（Hashtable），以连接配置字符串内容为key，主要是为了防止每次都去实例化(极大的影响效率和无谓的增加连接数和出口端口占用)。

2. 重要的几个方法

2.1 获取连接实例的公开方法。用于调用原封装类库时（一次封装），取已存在的连接实例。

2.2 通知更新的方法。用于在关键业务点操作时，通知缓存需要更新。通知失败要写日志。

2.3 根据key和field获取dataset格式缓存记录。专用于获取存入的数据集类型缓存。

3. 在使用时，原从数据库获取数据的场景上，先获取缓存数据，若返回null，才穿透到数据库，从数据库获取数据后，如果获取到的不是null并至少有一个表，则需要写入一份到缓存，作为初始缓存数据。

1. **各种业务缓存更新真正的操作类：**

开始更新缓存每个业务对应的类库，通常有一个方法，也可能多个。

这是封装好的一些列针对业务的DLL，固定步骤： 先删除对应键值，再呼叫正常业务接口获取数据。最后把数据更新至缓存。最后删除G\_开头的key和field对应的缓存。

1. **具体业务可行性或方案分析**
2. **默认大盘数据(优先级高)**

暂时保留现有模式，只是原txt不再生成，把原txt内容存入Redis， 客户端通过访问一个Handler.ashx(一般处理程序)，从Redis读取byte[]后，直接返回到客户端。 到了客户端仍然还是转化成dataset供程序使用。

1. 大盘详情页面

暂不处理，目前已去掉了定时刷新。

1. 大盘分类下对应数据

保持现状吧，目测刻意看分类的访问量不会很大。大了再说。

1. **自选商品(优先级高)**

因为这个功能必须要定时刷新，所以，原获取自选商品数据集的接口(业务名称是”自选商品临时列表”)，只需要换成： 优先尝试从缓存取出数据集返回，找不到再穿透到数据库中正常查找，从数据库找到后，再更新一次缓存。

而缓存的更新，在用户添加、删除自选商品时更新就行。 缓存使用哈希类型。 Key固定为“H:xxsp”，filed为用户登录邮箱， value为用户要显示的自选商品列表完整数据集的byte[]。

1. **成交详情(优先级高)**

写个接口，从监控里面调用，直接每小时更新一次缓存就行。 直接从缓存读取全部数据（一个键值记录全部数据即可）,也不用穿透，缓存没有也不去看数据库了。为减少数据量，只取最近200条就行了，原来取得太多了。 并且取消这个栏目的自动实时刷新功能。

1. **底部统计分析(优先级高)**

同大盘。将txt替换成缓存读取。

1. 商品分类

暂不处理应该也没问题。

1. **动态参数设定表(优先级高)**

在监控中调用一个新接口， 每天凌晨跑其他监控前，更新一下缓存就行了。 用一个key， 一个hash的filed，存一个字段正好。只有一行数据。取得时候，一次取出来用。 目前应该没有单独取某个字段的地方，都是统一用了方法一次取所有。

1. 平台节假日表**(优先级高)**

这个可以放入监控，也是每天凌晨更新缓存一次， 这个用集合存应该比较方便，把节假日日期全部存入集合就行了。

1. 用户基本资料和关联经纪人资料。**(优先级高)**

这个需要具体分析。比较复杂了。

1. 通知记录和提醒连同强制踢人**(优先级高)**

通知的列表查询，预计使用频率不会很高，不使用缓存，仍然从数据库读取。也就是说原来的业务代码中，正常往数据库插入提醒。

要解决的是客户端定时去找有没有新提醒的问题。 让定期的查询，只去缓存找，不穿透数据库,找到的取出来用的的同时数据就删掉。 为了减少代码修改量（到处都插入了提醒），直接在监控中更新缓存。 每轮跑监控，都找出“未查看”的提醒，一个人一个提醒一条写入缓存（由于要按照时间排序，缓存数据结构要用list比较合适）。 写入后，从数据库标记成“已查看”。 踢人功能同理。

1. 默认C区第一个列表。**(优先级高)**
2. 交易账户中间的余额统计。**(优先级高)**
3. 框架中的异步接口处理**(优先级中)**

取消txt保存，存入缓存。 使用list。按顺序异步执行。