列表数据本地化思想

1. 简述：App 开发过程通常为界面的搭建配合数据的请求和展示两大部分。在界面搭建好后，数据是通过联网进行网络请求获得的，也就是说，一般来说，app 从打开开始进入一个列表，列表的初始状态是无数据的。数据请求给用户的直观反映就是列表从空到展示数据。那么问题来了，如果网络断了呢？那列表就一直空了？很多 app 像雪球等，很多页面列表就有做本地数据缓存，进入页面时若列表为空，则先从本地取缓存数据出来展示，然后进行网络请求并更新列表数据，这样能提供不错的用户体验，不至于出现断网情况下进入 app 列表无数据的尴尬现象。以下就华讯股大师 app 的自选股列表为例进行列表本地化思想的讲解。

2. 自选股列表本地化思想

   (1) 为什么要对列表创建本地缓存？当然是为了更好的用户体验，其次是避免多次请求用户自选表接口，只请求一次，然后列表增删改查让本地保证和服务器同步就好了。那为啥不用内存缓存来做呢？首先，理解下内存缓存和本地缓存的区别。所谓内存缓存，举个简单的例子：单例。程序运行期间只会被实例化一次的对象。假如，程序重新运行了的话，单例里的数据就清空了，这就是内存缓存，和 app 的生命周期保持一致。如果用单例来做的话，那么 app 第一次打开进入自选股列表的时候，可想而知，第一次打开，列表数据为空，需要网络请求。也就是说，使用内存缓存还是会有空列表的情况。要避免这种情况，那就得使用本地缓存了。想想是不是这样的，比如偏好设置 NSUserDefaults，里面存了个登陆状态，那么 app 下一次打开 app 还是可以获取到当前本地存储的登陆状态，这个状态不会像单例一样数据重新初始化。

   (2) 怎么实现自选股列表本地化？

   1) 创建单例对象，在单例对象里定义自选股列表属性。

```
@interface MYMemoryDefaults : NSObject//此类用来存放非持久化的用户偏好数据(仅存于内存)

+ (instancetype)standardUserDefaults;


//----------------------------------------属性及相关说明----------------------------------------

/**
 * 用户自选股列表
 *
 * @note 用来存放自选股列表。
 */
@property (strong, nonatomic) NSMutableArray <GuPiaoDaiMaiName *> *stocksList;
```

2）viewWillAppear 里判断是否未请求过自选列表即单例内属性是否为空，为空的话就先从本地数据库取自选股数据出来展示。当自选股请求成功后，更新至单例，那么下一次进入页面，都会从单例取数据给列表展示，不从本地数据库取。也就是说每次进页面，要么从单例要么从本地数据库取，只要用户有自选股那么列表几乎不可能为空。

```
if(![[MYMemoryDefaults standardUserDefaults].stocksList count]){
   //未请求过自选列表，先从数据库取
   self.tableViewModel.data = [self.tableViewModel getAllZiXuanGuPiaoFromLocalSqlite];//从本地数据库初始化自选股数据
}else{
   self.tableViewModel.data = [MYMemoryDefaults standardUserDefaults].stocksList;
}

if(![self.tableViewModel getItemCount]){
   [self showEmptyLabel];
}else{
   [self hideEmptyLabel];
   [self.tableViewModel.data enumerateObjectsUsingBlock:^(GuPiaoDaiMaiName *obj, NSUInteger idx, BOOL * _Nonnull stop) {
      obj.cellsColorChangeType = 0;
   }];
}
[self.tableView reloadData];

[self refreshTableView];
```

3）refreshTableView 逻辑必须处理好，比如什么时候进行服务器和本地列表数据同步。数据请求 model 里可以定义一个 userZiXuanXuanData 属性，如果单例自选股数据为空则进行网络请求用户自选股列表。请求成功后数据保存至单例。如果单例自选股数据不为空，那么 userZiXuanXuanData 为单例里的自选股数据，请求完自选股实时行情后将新数据更新至单例中。

```
__weak typeof(self) weakSelf = self;
NSMutableArray *zixuan = [MYMemoryDefaults standardUserDefaults].stocksList;

if([zixuan count]){
   self.userZiXuanXuanData = zixuan;

   [self requestRealDetail:self.userZiXuanXuanData success:^(NSArray<GuPiaoDaiMaiName *> *responseObject) {
      NSMutableArray *selfStocks = [MYMemoryDefaults standardUserDefaults].stocksList;
      if(![responseObject count]){//返回为空或返回股票数和请求不一致 || [responseObject count] != [selfStocks count]
         completion(YES, [NSError errorWithDomain:@"返回为空" code:-1002 userInfo:nil]);
         return ;
      }

      [responseObject enumerateObjectsUsingBlock:^(GuPiaoDaiMaiName * _Nonnull obj, NSUInteger idx, BOOL * _Nonnull stop) {
         //相比上一次请求涨跌情况
         CGFloat dif_rate_pre = [[(GuPiaoDaiMaiName *)[selfStocks objectAtIndex:idx]diff_rate] floatValue];
         CGFloat dif_rate_now = [obj.diff_rate floatValue];
         if(dif_rate_pre != dif_rate_now)
            obj.cellsColorChangeType = dif_rate_pre < dif_rate_now ? 2: 1;
         [selfStocks replaceObjectAtIndex:idx withObject:obj];
      }];

      weakSelf.data = responseObject;
      completion(YES, nil);
   }];
   return;
}
```

```
    [self.userZiXuanXuanDataremoveAllObjects];
    [self getZiXuanGuListDataWithResult:^(BOOL isLastData, NSError *error) {//isLastData表示userZiXuanXuanData是否为空
        if(isLastData == YES){//请求成功，userZiXuanXuanData不为空
            [self requestRealDetail:weakSelf.userZiXuanXuanDatasuccess:^(NSArray<GuPiaoDaiMaiName *> *responseObject) {
                if(![responseObject count]){
                    completion(YES, [NSError errorWithDomain:@"返回为空" code:-1002 userInfo:nil]);
                    return ;
                }

                [responseObject enumerateObjectsUsingBlock:^(GuPiaoDaiMaiName * _Nonnull obj, NSUInteger idx, BOOL * _Nonnull stop) {
                    obj.cellsColorChangeType = 0;
                }];

                //将新数据存入内存
                [MYMemoryDefaults standardUserDefaults].stocksList = [responseObject mutableCopy];

                weakSelf.data = responseObject;
                completion(YES, nil);
            }];
        }else{//请求成功，userZiXuanXuanData为空
            //将新数据存入内存
            [[MYMemoryDefaults standardUserDefaults].stocksList removeAllObjects];
            [weakSelf deleteAllZiXuanGuPiaoIntoLocalSqlite];
            [weakSelf.data removeAllObjects];
            completion(YES, nil);
        }
    }];
}
```

4）确保单例里自选股列表操作保持和服务器自选股列表操作保持一致，比如添加了自选股，要确保单例也添加一项；删除了自选股，单例对应项也得删除；置顶自选股，单例对应项也得置顶等。

5）在合适的地方将单例的自选股列表本地化到 sqlite 数据库。什么地方需要进行此项操作呢？首先是用户登陆状态发生变化，比如退出登陆，被人登陆顶掉了，需要保存这个用户的自选股列表，并且清除单例自选股列表数据。

```
//账号登录，将内存的自选股列表更新到本地数据库
NSMutableArray *array = [MYMemoryDefaults standardUserDefaults].stocksList;
if([array count]){
    //清空本地数据库
    BOOL flag = [PeopleDao deleteAllTableStockcodeWithUserId:[[NSUserDefaults standardUserDefaults] objectForKey:@"userid"]
        withroomId:@"10"];
    if(flag){
        [array enumerateObjectsUsingBlock:^(GuPiaoDaiMaiName *obj, NSUInteger idx, BOOL * _Nonnull stop) {
            obj.row = [NSString stringWithFormat:@"%ld", idx];
            NSString *userid = [[NSUserDefaults standardUserDefaults] objectForKey:@"userid"];
            [PeopleDao insertIntoStockcodeWithName:obj.Stockcode withStockname:obj.Stockname withRoomId:@"10" withUserId:userid
                withnowPrice:obj.nowPrice withdiff_rate:obj.diff_rate withMarket:obj.market withRow:[NSString stringWithFormat:@"%d",
                [PeopleDao getCountOfRowsOfStockTableWithUserId:userid withroomId:@"10"]]];
        }];
    }
    //清除内存的自选股列表
    [[MYMemoryDefaults standardUserDefaults].stocksList removeAllObjects];
}
```

比如 app 退到后台或 app 关闭后需要保存这个用户的自选股列表，并且清除单例自选股列表数据。

```
- (void)applicationDidEnterBackground:(UIApplication *)application {
    // Use this method to release shared resources, save user data, invalidate timers, and store enough application state information to restore your
       application to its current state in case it is terminated later.
    // If your application supports background execution, this method is called instead of applicationWillTerminate: when the user quits.

    [[UIApplication sharedApplication] setApplicationIconBadgeNumber:0];


    //应用退出或退入后台，将内存的自选股列表更新到本地数据库
    NSMutableArray *array = [MYMemoryDefaults standardUserDefaults].stocksList;
    if([array count]){
        //清空本地数据库
        BOOL flag = [PeopleDao deleteAllTableStockcodeWithUserId:[[NSUserDefaults standardUserDefaults] objectForKey:@"userid"] withroomId:@"10"
           ];
        if(flag){
            [array enumerateObjectsUsingBlock:^(GuPiaoDaiMaiName *obj, NSUInteger idx, BOOL * _Nonnull stop) {
                obj.row = [NSString stringWithFormat:@"%ld", idx];
                NSString *userid = [[NSUserDefaults standardUserDefaults] objectForKey:@"userid"];
                [PeopleDao insertIntoStockcodeWithName:obj.Stockcode withStockname:obj.Stockname withRoomId:@"10" withUserId:userid withnowPrice:obj
                   .nowPrice withdiff_rate:obj.diff_rate withMarket:obj.market withRow:[NSString stringWithFormat:@"%d", [PeopleDao
                   getCountOfRowsOfStockTableWithUserId:userid withroomId:@"10"]]];
            }];
        }
    }
}
```