列表数据请求框架搭建

1. 简述：开发应用难免和列表打交道，可能是 UITableView 或 UICollectionView，正常开发思路是：配置数据源，定义列表展示数据，使用 MJRefresh 进行上下拉刷新，请求服务器返回数据更新，刷新列表展示新数据。这出现一个问题：控制器代码过于复杂。所以应该把某些逻辑抽象出来，如数据和请求，将其定义为一个 Model，负责提供列表的数据源，提供数据并维护数据。
2. 框架搭建(以 UICollectionView 为例)

   (1) 定义 BaseManagerModel

```
@protocol ManagerStateObserver <NSObject>

@optional
- (void)managerDidLoadedData:(BaseManager *)manager;
- (void)managerDidFailedData:(BaseManager *)manager;

@end

typedef void(^managerFinsihBlock)(BaseManager *manager);

@interface BaseManager : NSObject

@property (nonatomic, assign) id<ManagerStateObserver> delegate;
@property (nonatomic, assign) NSInteger errorCode;
@property (nonatomic, strong) NSString *errorMessage;
@property (nonatomic, strong) id data;
@property (nonatomic, assign) BOOL isLoading;
@property (nonatomic, assign) JCRequestID requestId;
@property (nonatomic, assign) JCNetworkResponseStatus status;

- (void)performSuccess:(BaseManager *)manager;
- (void)perFormFail:(BaseManager *)manager;

@end
```

```
#import "BaseManager.h"

@implementation BaseManager

@synthesize delegate = _delegate;
@synthesize data = _data;
@synthesize isLoading = _isLoading;
@synthesize requestId = _requestId;
```

```
#pragma mark self method

- (void)performSuccess:(BaseManager *)manager
{
    if ([self.delegate respondsToSelector:@selector(managerDidLoadedData:)]) {
        [self.delegate managerDidLoadedData:manager];
    }
}

- (void)perFormFail:(BaseManager *)manager
{
    if ([self.delegate respondsToSelector:@selector(managerDidFailedData:)]) {
        [self.delegate managerDidFailedData:manager];
    }
}

@end
```

```
#import "BaseManager.h"

typedef void (^MYCollectionCellConfigureBlock)(id collectionViewCell, id item, NSIndexPath *indexPath);
typedef void (^MYCollectionReusableViewBlock)(UICollectionReusableView *reusableView, NSString *elementKind,
    NSIndexPath *indexPath);

@interface MYBaseManagerModel : BaseManager <UICollectionViewDataSource>

@property (nonatomic, assign) NSInteger total;
@property (nonatomic, assign) NSInteger itemCount;

@property (nonatomic, assign) NSInteger startRow;
@property (nonatomic, assign) NSInteger pages;
@property (nonatomic, strong) NSString *keywords;

@property (nonatomic, assign) BOOL isReset;

@property (nonatomic, strong) NSString *cellIdentifier;
@property (nonatomic, strong) NSString *headerIdentifier;
@property (nonatomic, strong) NSString *footerIdentifier;

@property (nonatomic, copy) MYCollectionReusableViewBlock reusableViewBlcok;
@property (nonatomic, copy) MYCollectionCellConfigureBlock configureCellBlock;
```

```
- (id)initWithCellIdentifier:(NSString *)aCellIdentifier
      withHeaderIdentifier:(NSString *)aHeaderIdentifier
       configureCellBlock:(MYCollectionCellConfigureBlock)aConfigureCellBlock
           withCellClass:(NSString *)className;

- (id)initWithCellIdentifier:(NSString *)aCellIdentifier
      withFooterIdentifier:(NSString *)aFooterIdentifier
       configureCellBlock:(MYCollectionCellConfigureBlock)aConfigureCellBlock
           withCellClass:(NSString *)className;
```

```
- (id)initWithCellIdentifier:(NSString *)aCellIdentifier
      withHeaderIdentifier:(NSString *)aHeaderIdentifier
      withFooterIdentifier:(NSString *)aFooterIdentifier
       configureCellBlock:(MYCollectionCellConfigureBlock)aConfigureCellBlock
           withCellClass:(NSString *)className;

- (NSInteger)getItemCount;
- (id)getItemDataWithIndex:(NSInteger)index;
- (BOOL)isMoreData;
- (NSInteger)getItemTotal;

- (void)cancelRequest;

- (void)reSetData;

@end
```

```
#import "MYBaseManagerModel.h"
#import "NSArray+ExtraMethod.h"
#import "List.h"

@interface MYBaseManagerModel ()

@property (nonatomic, strong) NSString *cellClassName;

@end

@implementation MYBaseManagerModel

- (id)init {
    self = [super init];
    if (self) {
    }
    return self;
}

- (id)initWithCellIdentifier:(NSString *)aCellIdentifier
        withHeaderIdentifier:(NSString *)aHeaderIdentifier
          configureCellBlock:(MYCollectionCellConfigureBlock)aConfigureCellBlock
               withCellClass:(NSString *)className {
    self = [super init];
    if (self) {
        self.cellIdentifier = aCellIdentifier;
        self.headerIdentifier = aHeaderIdentifier;
        self.cellClassName = className;
        self.configureCellBlock = [aConfigureCellBlock copy];
    }

    return self;
}
```

```
-(id)initWithCellIdentifier:(NSString *)aCellIdentifier withFooterIdentifier:(NSString *)aFooterIdentifier
   configureCellBlock:(MYCollectionCellConfigureBlock)aConfigureCellBlock withCellClass:(NSString *)className
{
   self = [super init];
   if (self) {
      self.cellIdentifier = aCellIdentifier;
      self.footerIdentifier = aFooterIdentifier;
      self.cellClassName = className;
      self.configureCellBlock = [aConfigureCellBlock copy];
   }
   return self;
}

- (id)initWithCellIdentifier:(NSString *)aCellIdentifier
      withHeaderIdentifier:(NSString *)aHeaderIdentifier
      withFooterIdentifier:(NSString *)aFooterIdentifier
        configureCellBlock:(MYCollectionCellConfigureBlock)aConfigureCellBlock
            withCellClass:(NSString *)className {
   self = [self initWithCellIdentifier:aCellIdentifier withHeaderIdentifier:aHeaderIdentifier configureCellBlock:
      aConfigureCellBlock withCellClass:className];

   self.footerIdentifier = aFooterIdentifier;

   return self;
}
```

```
#pragma self private method

- (id)itemAtIndexPath:(NSIndexPath *)indexPath {
   return self.data[(long)indexPath.row];
}

- (NSInteger)getItemCount {
   return self.itemCount;
}

- (NSInteger)getItemTotal {
   return self.total;
}

- (id)getItemDataWithIndex:(NSInteger)index {
   return [self.data objectSafetyAtIndex:index];
}

- (BOOL)isMoreData {
   if ([self.data isKindOfClass:[NSArray class]]) {
      return self.total > self.itemCount ? YES : NO;
   }
   return NO;
}
```

```
- (void)cancelRequest {
    [[JCRequestProxy sharedInstance] cancelRequestID:self.requestId];
    if (self.isLoading) {
        self.isLoading = NO;
    }
}

- (void)reSetData {
    self.keywords = @"";
    self.startRow = 0;
    self.isReset = YES;

    self.total = 0;
    self.itemCount = 0;
}

#pragma mark UICollectionViewDataSource

- (NSInteger)numberOfSectionsInCollectionView:(UICollectionView *)collectionView {
    return 1;
}

- (NSInteger)collectionView:(UICollectionView *)collectionView numberOfItemsInSection:(NSInteger)section {
    return [self.data count];
}
```

```
- (UICollectionViewCell *)collectionView:(UICollectionView *)collectionView cellForItemAtIndexPath:(NSIndexPath
    *)indexPath {
    UICollectionViewCell *cell = [collectionView dequeueReusableCellWithReuseIdentifier:self.cellIdentifier
        forIndexPath:indexPath];

    id item = [self itemAtIndexPath:indexPath];
    self.configureCellBlock(cell, item, indexPath);

    return cell;
}
```

```
- (UICollectionReusableView *)collectionView:(UICollectionView *)collectionView
    viewForSupplementaryElementOfKind:(NSString *)kind atIndexPath:(NSIndexPath *)indexPath {
    UICollectionReusableView *reuseableView;

    if ([kind isEqualToString:UICollectionElementKindSectionHeader]) {
        UICollectionReusableView *headerView = [collectionView dequeueReusableSupplementaryViewOfKind:
            UICollectionElementKindSectionHeader withReuseIdentifier:self.headerIdentifier forIndexPath:indexPath];
        reuseableView = headerView;
    } else {
        UICollectionReusableView *footerView = [collectionView dequeueReusableSupplementaryViewOfKind:
            UICollectionElementKindSectionFooter withReuseIdentifier:self.footerIdentifier forIndexPath:indexPath];
        reuseableView = footerView;
    }

    if (self.reusableViewBlcok) {
        self.reusableViewBlcok(reuseableView, kind, indexPath);
    }

    return reuseableView;
}
```

(2) 定义具体列表 Model

1) 根据实际情况重写-initWith 方法和 dataSource 相关方法，添加必要请求参数

2) 添加数据请求方法，处理数据更新

```
#import "MYBaseManagerModel.h"
#import "MYESiteInfo.h"

@interface MYESiteManagerModel : MYBaseManagerModel

- (id)initWithCellIdentifier:(NSString *)aCellIdentifier
        configureCellBlock:(MYCollectionCellConfigureBlock)aConfigureCellBlock
            withCellClass:(NSString *)className;

- (void)loadESiteDataWithBlock:(managerFinsihBlock)finishBlock;

- (void)removeListData:(MYESiteUnitInfo *)infoData;
- (void)reSetAllCondition;

- (BOOL)isNeedRefresh;

@end
```

```
#import "MYESiteManagerModel.h"
#import "MYRequestProxy.h"
#import "MYJobForwordingIndexRequest.h"
#import "NSArray+ExtraMethod.h"

@interface MYESiteManagerModel ()

@property (atomic, copy) managerFinsihBlock managerBlock;
@property (nonatomic, assign) NSTimeInterval lastTimeInterval;

@end

@implementation MYESiteManagerModel

@synthesize isLoading = _isLoading;

- (id)init {
    self = [super init];

    if (self) {
    }

    return self;
}
```

```
- (id)initWithCellIdentifier:(NSString *)aCellIdentifier
         configureCellBlock:(MYCollectionCellConfigureBlock)aConfigureCellBlock
             withCellClass:(NSString *)className {
    self = [super initWithCellIdentifier:aCellIdentifier withHeaderIdentifier:nil configureCellBlock:
        aConfigureCellBlock withCellClass:className];

    if (self) {
        self.isLoading = NO;
        self.isReset = NO;
        self.data = [NSMutableArray array];
        self.requestId = -1;
        self.startRow = 0;
        self.total = 0;
        self.itemCount = 0;
    }

    return self;
}
```

```objc
- (void)loadESiteDataWithBlock:(managerFinsihBlock)finishBlock {
    if (self.isLoading) { //在载入的话就不动
        return;
    }

    if (_managerBlock != finishBlock) {
        _managerBlock = finishBlock;
    }

    MYJobForwordingIndexRequest *request = [[MYJobForwordingIndexRequest alloc] init];
    request.keywords = self.keywords;
    request.startIndex = [NSString stringWithFormat:@"%ld", (long)self.startRow];
    request.stepLength = @"10";

    __weak typeof(self) weakSelf = self;
    self.isLoading = YES;
    self.requestId = [[MYRequestProxy sharedInstance] httpPostWithRequest:request entityClass:@"MYESiteInfo"
        withBlock:^(JCNetworkResponse *response){
        weakSelf.isLoading = NO;
         if (response.status == JCNetworkResponseStatusSuccess) {
            weakSelf.startRow += 10;
            MYESiteInfo *infoData = response.content;

            if (weakSelf.isReset) {
                weakSelf.data = infoData.eSiteArrary;
                weakSelf.isReset = NO;
            } else {
                [weakSelf.data addObjectsFromArray:infoData.eSiteArrary];
            }
```

```
            weakSelf.total = infoData.pageInfo.totalCount;
            weakSelf.itemCount = [weakSelf.data count];
            weakSelf.errorCode = 1;
            weakSelf.errorMessage = infoData.msg;
        } else {
            weakSelf.errorCode = response.error.code;
            weakSelf.errorMessage = [response.error localizedDescription];
        }
        finishBlock(weakSelf);
    }];
}

#pragma mark self method

- (void)removeListData:(MYESiteUnitInfo *)infoData {
    if ([self.data containsObject:infoData]) {
        [self.data removeObject:infoData];
    }
    self.itemCount = [self.data count];
    self.total -= 1;
}

- (void)reSetAllCondition {
    self.isReset = YES;
    self.requestId = -1;
    self.startRow = 0;
    self.total = 0;
    self.itemCount = 0;
}

- (BOOL)isNeedRefresh {
    //超过500秒刷新
    NSTimeInterval currentTimeInterval = [[NSDate date] timeIntervalSince1970];
    if (currentTimeInterval - self.lastTimeInterval >= 600) {
```

(3) viewController 中的配置

```
self.eSiteManagerModel = [[MYESiteManagerModel alloc] initWithCellIdentifier:@"microShopCell"
    withHeaderIdentifier:nil configureCellBlock:^(id collectionViewCell, id item, NSIndexPath *indexPath) {
    [(MYMicroShopCollectionViewCell *)collectionViewCell setDelegate:selfWeak];
    [(MYMicroShopCollectionViewCell *)collectionViewCell loadESiteInfo:item];
} withCellClass:nil];

[self.microShopCollectionView setDataSource:self.eSiteManagerModel];
```

```
self.microShopCollectionView.footer = [MYRefreshFooter footerWithRefreshingBlock: ^{
    if([selfWeak.eSiteManagerModel isMoreData]){
     [selfWeak.eSiteManagerModel loadESiteDataWithBlock:^(BaseManager *manager) {
        [selfWeak handlerNetworkInfo:manager];
        [selfWeak.microShopCollectionView.footer endRefreshing];
        [selfWeak.microShopCollectionView reloadData];
     }];
      return ;
    }
    [selfWeak.microShopCollectionView.footer endRefreshingWithNoMoreData];
}];
```

```
- (void)refreshMicoShop {
    [self.eSiteManagerModel reSetData];
    [self.microShopCollectionView.footer resetNoMoreData];

    if(!self.eSiteManagerModel.isLoading) {
        [self.microShopCollectionView setScrollEnabled:NO];
    }
    [self.microShopCollectionView hide];

    @WeakObj(self);
    [self.eSiteManagerModel loadESiteDataWithBlock:^(BaseManager *manager) {
        [selfWeak.progressView stopAnimating];
        [selfWeak hideProgress];
        [selfWeak handlerNetworkInfo:manager];
        [selfWeak.microShopCollectionView setScrollEnabled:YES];
        if([selfWeak.eSiteManagerModel getItemCount] != 0){
            [selfWeak.microShopCollectionView hide];

        }else{
            [selfWeak.microShopCollectionView showMessage:@"您还没有添加任何招聘信息"];
            CGFloat height = WINDOW_WIDTH*328/720 + 44;
            if (iPhone4s) {
                [selfWeak.microShopCollectionView setImageViewCenterYConstraintEqualToSuperViewCenterY:
                    WINDOW_HEIGHT*0.12+2 - height];
            }else
                [selfWeak.microShopCollectionView setImageViewCenterYConstraintEqualToSuperViewCenterY:
                    WINDOW_HEIGHT*0.1 - height];

            [selfWeak.microShopCollectionView showButtonWithContent:@"添加订单"];
            [selfWeak.microShopCollectionView.showButton addTarget:self action:@selector(goRecruit:)
                forControlEvents:UIControlEventTouchUpInside];
        }
        [selfWeak.microShopCollectionView reloadData];
    }];
```