主从列表封装思想

1. 简述：项目中经常需要用到主从列表，比如城市选择啊，筛选等功能，如果每一级都是单独都一个页面，那么只要做好页面间传值即可。但这种往往少见，通常是需要同个页面展示多级列表，如果每次都写一遍，那么完全没必要的，有一些通用的地方是可以进行封装的。
2. 以下就主从列表封装提出几个问题：
   (1) 是否封装 cell 的加载动画？个人觉得没必要，cell 显示前丢给外界就完事，怎么动画让开发者自己动脑筋就好。
   (2) 是否封装 cell 样式？个人觉得没必要，怎么好看的自定义 cell，让外界自己定义，封装只需要加载显示即可，顶多外界没定义时加载显示系统默认 cell 样式就好
3. DoubleTableView 封装如下：

```
static NSInteger const MYDOUBLETABLEVIEW_Main_Tag = 1000;//mainTable的tastaticg
static NSInteger const MYDOUBLETABLEVIEW_Sub_Tag = 1001;//subTable的tag

typedef void (^MYDoubleTableCellConfigBlock)(UITableView *tableView, id tableCell, id item, NSIndexPath *indexPath);


@protocol  MYDoubleTableViewDelegate<NSObject>

@optional;
- (void)doubleTableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath;
- (CGFloat)doubleTableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath;

- (void)scrollViewWillBeginDragging:(UIScrollView *)scrollView;

@end


@interface MYDoubleTableView : UIView

@property (strong, nonatomic, readonly) UITableView *mainTable;
@property (strong, nonatomic, readonly) UITableView *subTable;

@property (copy, nonatomic) MYDoubleTableCellConfigBlock cellConfigBlock;
@property (assign, nonatomic) id <MYDoubleTableViewDelegate> delegate;

-(instancetype)initWithFrame:(CGRect)frame mainTableCellClass:(Class)mainCell subTableCellClass:(Class)subCell;

- (void)reloadMainTable:(NSMutableArray *)data;
- (void)reloadSubTable:(NSMutableArray *)data;

- (void)clickMainTable:(NSInteger)index; //此方法调用后会调用代理方法doubleTableView:didSelectRowAtIndexPath:

@end
```

```objc
#import "MYDoubleTableView.h"
#import "MYDoubleTableModel.h"

static NSString *const kMainTableCellIdentifier = @"mainCell";
static NSString *const kSubTableCellIdentifier = @"subCell";

@interface MYDoubleTableView ()<UITableViewDelegate>

@property (weak, nonatomic) Class mainCellClass;
@property (weak, nonatomic) Class subCellClass;

@property (strong, nonatomic) MYDoubleTableModel *mainTableDataSource;
@property (strong, nonatomic) MYDoubleTableModel *subTableDataSource;

@end

@implementation MYDoubleTableView

- (instancetype)initWithFrame:(CGRect)frame{
    if(self = [super initWithFrame:frame]){
        _mainCellClass = [UITableViewCell class];
        _subCellClass = [UITableViewCell class];
        [self configDoubleTable];
    }
    return self;
}

- (instancetype)initWithFrame:(CGRect)frame mainTableCellClass:(Class)mainCell subTableCellClass:(Class)subCell{
    if(self = [super initWithFrame:frame]){
        _mainCellClass = mainCell;
        _subCellClass = subCell;
        [self configDoubleTable];
    }
    return self;
}
```

```
- (void)configDoubleTable{
    __weak typeof(self) weakSelf = self;
    _mainTable = [[UITableView alloc] initWithFrame:CGRectMake(0, 0, 100, CGRectGetHeight(self.bounds)) style:UITableViewStylePlain];
    [_mainTable registerClass:_mainCellClass forCellReuseIdentifier:kMainTableCellIdentifier];
    _mainTable.separatorStyle = UITableViewCellSeparatorStyleNone;
    _mainTable.tableFooterView = [[UIView alloc] init];
    _mainTable.backgroundColor = [UIColor whiteColor];
    _mainTable.showsVerticalScrollIndicator = NO;
    _mainTable.tag = MYDOUBLETABLEVIEW_Main_Tag;
    [self addSubview:_mainTable];
    _mainTableDataSource = [[MYDoubleTableModel alloc] initWithCellIdentifier:kMainTableCellIdentifier headerIdentifier:nil cellConfigBlock:^(id tableCell, id item,
        NSIndexPath *indexPath) {
        if(weakSelf.cellConfigBlock){
            weakSelf.cellConfigBlock(weakSelf.mainTable, tableCell, item, indexPath);
        }
    }];
    _mainTable.dataSource = _mainTableDataSource;
    _mainTable.delegate = self;

    _subTable = [[UITableView alloc] initWithFrame:CGRectMake(100, 0, CGRectGetWidth(self.bounds) - 100, CGRectGetHeight(self.bounds)) style:UITableViewStylePlain]
        ;
    [_subTable registerClass:_subCellClass forCellReuseIdentifier:kSubTableCellIdentifier];
    _subTable.separatorStyle = UITableViewCellSeparatorStyleNone;
    _subTable.tableFooterView = [[UIView alloc] init];
    _subTable.backgroundColor = [UIColor whiteColor];
    _subTable.showsVerticalScrollIndicator = NO;
    _subTable.tag = MYDOUBLETABLEVIEW_Sub_Tag;
    [self addSubview:_subTable];
    _subTableDataSource = [[MYDoubleTableModel alloc] initWithCellIdentifier:kSubTableCellIdentifier headerIdentifier:nil cellConfigBlock:^(id tableCell, id item,
        NSIndexPath *indexPath) {
        [tableCell setSelectionStyle:UITableViewCellSelectionStyleNone];
        if(weakSelf.cellConfigBlock){
            weakSelf.cellConfigBlock(weakSelf.subTable, tableCell, item, indexPath);
        }
    }];
    _subTable.dataSource = _subTableDataSource;
    _subTable.delegate = self;
```

```
#pragma mark UITableViewDelegate

- (void)tableView:(UITableView *)tableView didSelectRowAtIndexPath:(NSIndexPath *)indexPath{
    if(self.delegate && [self.delegate respondsToSelector:@selector(doubleTableView:didSelectRowAtIndexPath:)]){
        [self.delegate doubleTableView:tableView didSelectRowAtIndexPath:indexPath];
    }
}

- (CGFloat)tableView:(UITableView *)tableView heightForRowAtIndexPath:(NSIndexPath *)indexPath{
    if(self.delegate && [self.delegate respondsToSelector:@selector(doubleTableView:heightForRowAtIndexPath:)]){
        return [self.delegate doubleTableView:tableView heightForRowAtIndexPath:indexPath];
    }
    return 44;
}

#pragma mark UIScrollViewDelegate

- (void)scrollViewWillBeginDragging:(UIScrollView *)scrollView{
    if(self.delegate && [self.delegate respondsToSelector:@selector(scrollViewWillBeginDragging:)]){
        [self.delegate scrollViewWillBeginDragging:scrollView];
    }
}
```

```
#pragma mark public methods

- (void)reloadMainTable:(NSMutableArray *)data{
    _mainTableDataSource.data = data;
    [_mainTable reloadData];
}

- (void)reloadSubTable:(NSMutableArray *)data{
    _subTableDataSource.data = data;
    [_subTable reloadData];
}

- (void)clickMainTable:(NSInteger)index{
    [self.mainTable selectRowAtIndexPath:[NSIndexPath indexPathForRow:index inSection:0] animated:NO scrollPosition:UITableViewScrollPositionNone];
    [self tableView:_mainTable didSelectRowAtIndexPath:[NSIndexPath indexPathForRow:index inSection:0]];
}
```

MYDoubleTableModel 封装如下：

```
#import <Foundation/Foundation.h>
#import <UIKit/UIKit.h>

typedef void (^MYCellConfigBlock)(id tableCell, id item, NSIndexPath *indexPath);

@interface MYDoubleTableModel : NSObject<UITableViewDataSource>

@property (strong, nonatomic) NSMutableArray *data;

- (instancetype)initWithCellIdentifier:(NSString *)cellIdentifier headerIdentifier:(NSString *)headerIdentifier cellConfigBlock:(MYCellConfigBlock)configBlock;

@end


#import "MYDoubleTableModel.h"

@interface MYDoubleTableModel ()

@property (strong, nonatomic) NSString *cellIdentifier;
@property (strong, nonatomic) NSString *headerIdentifier;
@property (copy, nonatomic) MYCellConfigBlock configBlock;

@end

@implementation MYDoubleTableModel

- (instancetype)initWithCellIdentifier:(NSString *)cellIdentifier headerIdentifier:(NSString *)headerIdentifier cellConfigBlock:(MYCellConfigBlock)configBlock{
    if(self = [super init]){
        self.data = [NSMutableArray array];
        self.cellIdentifier = cellIdentifier;
        self.headerIdentifier = headerIdentifier;
        self.configBlock = [configBlock copy];
    }
    return self;
}
```

```
#pragma mark UITableViewDataSource

- (NSInteger)numberOfSectionsInTableView:(UITableView *)tableView{
    return 1;
}

- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section{
    return [self.data count];
}

- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexPath{
    UITableViewCell *cell = [tableView dequeueReusableCellWithIdentifier:self.cellIdentifier forIndexPath:indexPath];
    if(self.configBlock){
        self.configBlock(cell, [self itemAtIndexPath:indexPath], indexPath);
    }
    return cell;
}

#pragma mark private methods

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

@end
```

4. 总结：

封装只要获取数据源 data 数组，然后加载出来之后，能准确把用户点击那个列表的第几个 cell 告诉外界即可。至于 cell 长什么样，大小如何，需不需要动画等等。我想这不是封装过程中该考虑的事。考虑的多了，封装逻辑就更繁琐，不利于其他人理解，而且有时候会使封装变的不够灵活(如 cell 样式写进了封装：“玛德，UI 设计要求的样式不是这样的，怎么自定义？封装代码多看不懂…这不麻烦了嘛 T T”)