# MYSliderView 封装

1. 简述：MYSliderView 是一个有着和 UIScrollView 类似轮播效果的视图，不一样的是，在展示 N 个列表时内存、性能比 scrollview 好很多，而且能使控制器和列表相关代码逻辑解耦。
2. 实现：

```
#import <UIKit/UIKit.h>
@class MYSliderView;

@protocol  MYSliderViewDatasource<NSObject>

@required;
- (UIViewController *)baseViewControllerOfSiderView:(MYSliderView *)sliderView;
- (NSInteger)numberOfViewControllersInSliderView:(MYSliderView *)sliderView;
- (UIViewController *)sliderView:(MYSliderView *)sliderView viewControllerAtIndex:(NSInteger)index;

@end

@protocol MYSliderViewDelegate <NSObject>

@optional;
- (void)sliderView:(MYSliderView *)sliderView switchingFrom:(NSInteger)fromIndex to:(NSInteger) toIndex percent:(CGFloat)percent;

@end

@interface MYSliderView : UIView
```

```
/**
* 概述：
* 结合之前独立navigationBar和bannerScrollView两者的封装思想完成了此次sliderView的实现，sliderView类似ScrollView，但能更高效，避免很多情况下内存的浪费，比如类似网易新闻顶部导航：如果有30个TableView或CollectionView列表的数据，由scrollView来嵌套30个列表，再假设每个列表都有banner轮播100张图片，那内存消耗是相当可怕的。此次封装的sliderView，结合NSCache的使用，可以做到，滑动过程最多load三个视图，并且滑动结束只保留一个视图，极大节省了内存的消耗。此次封装是解耦思想的一大体现。
*
*
* 实现：
* 1.baseVC使用NSCache存储相应列表对应的VC，然后作为sliderview的数据源为其提供baseVC、滑动结束显示的当前VC以及相应列表对应的VC总数
* 2.初始化sliderView：获取到数据源后调用showViewControllerAtIndex:0，将第1r个VC即currentVC添加到baseVC作为子控制器，将视图添加到sliderView，设置约束，其中有个动态约束用来控制VC的view移动，每次调用showViewControllerAtIndex都会重置sliderview
* 3.给sliderview添加pan手势，在手势触发方法里处理相应逻辑：
*  3.1 滑过程如果判断往右则通过代理向baseVC获取上一ViewController，添加到baseVC作为子控制器，将视图添加到sliderView，约束紧贴currentVC左边沿；左滑获取下一个viewController，添加到baseVC作为子控制器，将视图添加到sliderView，约束紧贴currentVC右边沿
*  3.2 滑动结束判断最终显示的是上一页还是下一页，实现动画将currentVC的约束左或右移一个sliderView宽度，然后调用showViewControllerAtIndex:页数，将所有vc移除掉，并将最终显示的作为仅存的currentVC
*
```

```
*
* 注意事项:
* 1.实现的3.1步骤，将所有vc移除掉，会走viewDisappear，然后将最终列表重新显示出来，即最终列表连续走了viewDisappear和viewAppear，而另外两个列表vc则只走viewDisappear
* 2.baseVC初始化sliderView，以及所有列表VC，建议将列表VC添加至NSCache，并设置cache限制数，假如5个列表VC,限制数为4，那么1-4列表第一次显示时会走viewDidLoad加载视图，5出来后1会dealloc，因为到了Cache限制，会将前面的cache干掉，然后让5进来。从5移回2，所有视图只会走viewDisappear和viewAppear，而移回1时，会重新viewDidLoad加载视图5又被干掉。网易有时候滑到后面专题列表回前面会重新刷新，可能就是类似原理吧。
* 3.本次封装仅实现了slider视图，将滑动进度通过代理传到baseVC，如果需要实现像网易那样切换的顶部导航，导航UI随滑动进度改变，则可以实现该代理方法获取进度
*
*/


@property (nonatomic, assign) id<MYSliderViewDatasource> dataSource;
@property (nonatomic, assign) id<MYSliderViewDelegate> delegate;

- (void)showViewController:(NSInteger)index;
```

```
#import "MYSliderView.h"
#import "UIView+YXHView.h"

typedef NS_ENUM(NSInteger, MYSliderViewControllerRemoveOption) {
    MYSliderViewControllerRemoveOptionAll,
    MYSliderViewControllerRemoveOptionLastAndCurrent,
    MYSliderViewControllerRemoveOptionNextAndCurrent
};

@interface MYSliderView ()

@property (nonatomic, weak) UIViewController *lastViewController;
@property (nonatomic, weak) UIViewController *currentViewController;
@property (nonatomic, weak) UIViewController *nextViewController;

@property (nonatomic, assign) NSInteger currentPageIndex;
@property (nonatomic, assign) CGFloat lastPanTranslation;

@property (nonatomic, strong) UIPanGestureRecognizer *panGestureRecognizer;

@property (nonatomic, strong) NSLayoutConstraint *currentLeftConstraint;

@property (nonatomic, assign) NSInteger numberOfViewControllers;
@property (nonatomic, weak) UIViewController *baseViewController;
- (UIViewController *)viewControllerAtIndex:(NSInteger)index;

@end
```

```objc
@implementation MYSliderView

- (id)initWithCoder:(NSCoder *)aDecoder {
    self = [super initWithCoder:aDecoder];

    if (self) {
        [self setUp];
    }

    return self;
}

- (id)initWithFrame:(CGRect)frame {
    self = [super initWithFrame:frame];

    if (self) {
        [self setUp];
    }

    return self;
}

- (void)setUp {
    self.currentPageIndex = -1;
    _lastPanTranslation = 0;
    self.panGestureRecognizer = [[UIPanGestureRecognizer alloc] initWithTarget:self action:@selector(handlePanGesture:)];
    [self addGestureRecognizer:self.panGestureRecognizer];
}
```

```
- (void)setDataSource:(id<MYSliderViewDatasource>)dataSource{
    if(!_dataSource){
        _dataSource = dataSource;
        [self showFinalViewControllerOfIndex:0];//初始化第一个视图控制器
        return;
    }
    _dataSource = dataSource;
}

#pragma mark Hanle PanGestureRecognizer

- (void)handlePanGesture:(UIPanGestureRecognizer *)pan {
    CGFloat translationx = [pan translationInView:self].x;

    BOOL showRightVC;
    if (pan.state == UIGestureRecognizerStateBegan) {
        _lastPanTranslation = translationx;
        if(translationx != 0.0){//有可能begin状态已经有transition了，得处理一下相关逻辑
            showRightVC = _lastPanTranslation > 0 ? NO : YES;
            [self addViewControllerIfNeeded:showRightVC];

            self.currentLeftConstraint.constant = _lastPanTranslation;
            if(self.currentPageIndex == 0)
                if(_lastPanTranslation > 0)
                    self.currentLeftConstraint.constant = 0;

            if(self.currentPageIndex == self.numberOfViewControllers - 1)
                if(_lastPanTranslation < 0)
                    self.currentLeftConstraint.constant = 0;


            [self layoutIfNeeded];
        }
    } else if (pan.state == UIGestureRecognizerStateChanged) {
```

```
} else if (pan.state == UIGestureRecognizerStateChanged) {
    [self layoutIfNeeded];
    CGFloat realTranslation = translationx - _lastPanTranslation;

    if(realTranslation != 0.0){//考虑可能移动方向发生改变，左右两个vc都需要显示
        showRightVC = realTranslation > 0 ? NO : YES;
        [self addViewControllerIfNeeded:showRightVC];
    }

    if(self.currentPageIndex == 0)
        if(realTranslation > 0)
            realTranslation = 0;

    if(self.currentPageIndex == self.numberOfViewControllers - 1)
        if(realTranslation < 0)
            realTranslation = 0;

    self.currentLeftConstraint.constant = realTranslation;
    if(realTranslation == 0)
        return;

    CGFloat leftConstant = self.currentLeftConstraint.constant;
    if(self.delegate && [self.delegate respondsToSelector:@selector(sliderView:switchingFrom:to:percent:)]){
        if(realTranslation > 0){
            [self.delegate sliderView:self switchingFrom:self.currentPageIndex to:self.currentPageIndex - 1 percent:fabs(leftConstant / 100) > 1?1 :
                fabs(leftConstant / 100)];
        }else{
            [self.delegate sliderView:self switchingFrom:self.currentPageIndex to:self.currentPageIndex + 1 percent:fabs(leftConstant / 100) > 1?1 :
                fabs(leftConstant / 100)];
        }
    }
} else if (pan.state == UIGestureRecognizerStateEnded) {
```

```
} else if (pan.state == UIGestureRecognizerStateEnded) {
    _lastPanTranslation = 0;
    __weak typeof(self) weakSelf = self;
    CGFloat distance = self.currentLeftConstraint.constant;

    if(distance == 0)//不滑动或禁止滑动
        return;
    else if(distance > 0){//右滑可能显上一个
        if(fabs(distance) > 100){//右滑距离足够显示上一个
            self.currentLeftConstraint.constant = self.frame.size.width;
            [UIView animateWithDuration:0.33 animations:^{
                [weakSelf layoutIfNeeded];
            }completion:^(BOOL finished) {
                NSInteger lastPageIndex = weakSelf.currentPageIndex - 1;
                [weakSelf showFinalViewControllerOfIndex:lastPageIndex];
                if(weakSelf.delegate && [weakSelf.delegate respondsToSelector:@selector(sliderView:switchingFrom:to:percent:)]){
                    [weakSelf.delegate sliderView:weakSelf switchingFrom:lastPageIndex +1 to:lastPageIndex percent:1];
                }
            }];
        }else{//右滑距离不足，视图复位
            weakSelf.currentLeftConstraint.constant = 0;
            [UIView animateWithDuration:0.33 animations:^{
                [weakSelf layoutIfNeeded];
            }];
            if(weakSelf.delegate && [weakSelf.delegate respondsToSelector:@selector(sliderView:switchingFrom:to:percent:)]){
                [weakSelf.delegate sliderView:self switchingFrom:self.currentPageIndex to:self.currentPageIndex  percent:1];
            }
        }
    }else{//左滑可能显下一个
```

```
    }else{//左滑可能显下一个
        if(fabs(distance) > 100){//左滑距离足够显示下一个
            self.currentLeftConstraint.constant = -self.frame.size.width;
            [UIView animateWithDuration:0.33 animations:^{
                [weakSelf layoutIfNeeded];
            }completion:^(BOOL finished) {
                NSInteger lastPageIndex = weakSelf.currentPageIndex + 1;
                [weakSelf showFinalViewControllerOfIndex:lastPageIndex];
                if(weakSelf.delegate && [weakSelf.delegate respondsToSelector:@selector(sliderView:switchingFrom:to:percent:)]){
                    [weakSelf.delegate sliderView:weakSelf switchingFrom:lastPageIndex -1 to:lastPageIndex percent:1];
                }
            }];
        }else{//左滑距离不足，视图复位
            weakSelf.currentLeftConstraint.constant = 0;
            [UIView animateWithDuration:0.33 animations:^{
                [weakSelf layoutIfNeeded];
            }];
            if(weakSelf.delegate && [weakSelf.delegate respondsToSelector:@selector(sliderView:switchingFrom:to:percent:)]){
                [weakSelf.delegate sliderView:weakSelf switchingFrom:weakSelf.currentPageIndex to:weakSelf.currentPageIndex  percent:1];
            }
        }
    }
  }
}
```

```
#pragma mark publicMethods

- (void)showViewController:(NSInteger)index{
  if(self.currentPageIndex == index)
    return;

  self.lastPanTranslation = 0;
  [self removeViewControllersWithOption:MYSliderViewControllerRemoveOptionAll];
  [self showFinalViewControllerOfIndex:index];
}

#pragma mark privateMethod

- (void)showFinalViewControllerOfIndex:(NSInteger)index{
  if(self.currentPageIndex == index)
    return;

  if(self.currentViewController){
    UIViewController *controller = index > self.currentPageIndex ? _nextViewController : _lastViewController;
    [self removeViewControllersWithOption:controller == _nextViewController ? MYSliderViewControllerRemoveOptionLastAndCurrent :
      MYSliderViewControllerRemoveOptionNextAndCurrent];//移除另外两个视图控制器
    [self removeConstraints:self.constraints];

    [controller.view setLayoutTopFromSuperViewWithConstant:0];
    [controller.view setLayoutBottomFromSuperViewWithConstant:0];
    [controller.view setLayoutWidth:self multiplier:1 constant:0];
    NSLayoutConstraint *leftConstraint = [NSLayoutConstraint constraintWithItem:controller.view attribute:NSLayoutAttributeLeft
      relatedBy:NSLayoutRelationEqual toItem:self attribute:NSLayoutAttributeLeft multiplier:1 constant:0];
    [self addConstraint:leftConstraint];
    self.currentLeftConstraint = leftConstraint;
```

```
    self.currentPageIndex = index;
    self.currentViewController = controller;

    if(controller == _lastViewController)
        _lastViewController = nil;
    else
        _nextViewController = nil;
    return;
}

//初始化currentViewController
UIViewController *currentViewController = [self viewControllerAtIndex:index];
[self.baseViewController addChildViewController:currentViewController];//懒加载前执行，保证currentVC在loadView也能访问其
    parrentVC
[currentViewController.view setTranslatesAutoresizingMaskIntoConstraints:NO];
 [self addSubview:currentViewController.view];

[currentViewController.view setLayoutTopFromSuperViewWithConstant:0];
[currentViewController.view setLayoutBottomFromSuperViewWithConstant:0];
[currentViewController.view setLayoutWidth:self multiplier:1 constant:0];
self.currentLeftConstraint = [NSLayoutConstraint constraintWithItem:currentViewController.view attribute:NSLayoutAttributeLeft
    relatedBy:NSLayoutRelationEqual toItem:self attribute:NSLayoutAttributeLeft multiplier:1 constant:0];
[self addConstraint:self.currentLeftConstraint];

[currentViewController didMoveToParentViewController:currentViewController.parentViewController];//addChild之后手动调用告诉ios已
    完成添加子视图控制器的操作，removeChild之后系统自动调用

self.currentPageIndex = index;
self.currentViewController = currentViewController;
}
```

```
- (void)addViewControllerIfNeeded:(BOOL)showRightVC{
  if(showRightVC == YES){
    NSInteger nextPageIndex = self.currentPageIndex + 1;
    if(nextPageIndex == self.numberOfViewControllers)
      return;
    if(!self.nextViewController){//不存在才添加
      _nextViewController = [self viewControllerAtIndex:nextPageIndex];
      [self.baseViewController addChildViewController:_nextViewController];//添加目标视图控制器，懒加载前执行，保证在loadView也
        能访问其parrentVC
      [_nextViewController.view setTranslatesAutoresizingMaskIntoConstraints:NO];
      [self addSubview:_nextViewController.view];

      [_nextViewController.view setLayoutTopFromSuperViewWithConstant:0];
      [_nextViewController.view setLayoutBottomFromSuperViewWithConstant:0];
      [_nextViewController.view setLayoutWidth:self multiplier:1 constant:0];
      NSLayoutConstraint *nextLeftConstraint = [NSLayoutConstraint constraintWithItem:_nextViewController.view attribute:
        NSLayoutAttributeLeft relatedBy:NSLayoutRelationEqual toItem:self.currentViewController.view attribute:NSLayoutAttributeLeft
        multiplier:1 constant:self.frame.size.width];
      [self addConstraint:nextLeftConstraint];

      [_nextViewController didMoveToParentViewController:_nextViewController.parentViewController];
    }
  }else{
```

```
    }else{
        NSInteger lastPageIndex = self.currentPageIndex - 1;
        if(lastPageIndex == -1)
            return;
        if(!self.lastViewController){//不存在才添加
            _lastViewController = [self viewControllerAtIndex:lastPageIndex];
            [self.baseViewController addChildViewController:_lastViewController];//添加目标视图控制器，懒加载前执行，保证在loadView也
                能访问其parrentVC
            [_lastViewController.view setTranslatesAutoresizingMaskIntoConstraints:NO];
            [self addSubview:_lastViewController.view];

            [_lastViewController.view setLayoutTopFromSuperViewWithConstant:0];
            [_lastViewController.view setLayoutBottomFromSuperViewWithConstant:0];
            [_lastViewController.view setLayoutWidth:self multiplier:1 constant:0];
            NSLayoutConstraint *lastLeftConstraint = [NSLayoutConstraint constraintWithItem:_lastViewController.view attribute:
                NSLayoutAttributeLeft relatedBy:NSLayoutRelationEqual toItem:self.currentViewController.view attribute:NSLayoutAttributeLeft
                multiplier:1 constant:-self.frame.size.width];
            [self addConstraint:lastLeftConstraint];

            [_lastViewController didMoveToParentViewController:_lastViewController.parentViewController];
        }
    }
}
```

```
- (void)removeViewControllersWithOption:(MYSliderViewControllerRemoveOption)option{
    BOOL last, current, next;
    last = current = next = NO;

    switch (option) {
        case MYSliderViewControllerRemoveOptionAll:
            last = current = next = YES;
            break;
        case MYSliderViewControllerRemoveOptionLastAndCurrent:
            last = current = YES;
            break;
        case MYSliderViewControllerRemoveOptionNextAndCurrent:
            current = next = YES;
            break;
        default:
            break;
    }

    if (self.currentViewController && current){
        [self removeViewController:self.currentViewController];
        self.currentViewController = nil;
    }

    if(self.lastViewController && last){
        [self removeViewController:self.lastViewController];
        self.lastViewController = nil;
    }

    if(self.nextViewController && next){
        [self removeViewController:self.nextViewController];
        self.nextViewController = nil;
    }
}
```

```
- (void)removeViewController:(UIViewController  *)controller{
    [controller willMoveToParentViewController:nil];//removChild之前手动调用，且参数为nil ，addChild之前系统自动调用
    [controller.view removeFromSuperview];
    [controller removeFromParentViewController];
}

#pragma mark getDatasFromDataSource

- (NSInteger)numberOfViewControllers {
    if (!_numberOfViewControllers)
        _numberOfViewControllers = [_dataSource numberOfViewControllersInSliderView:self];
    return _numberOfViewControllers;
}

- (UIViewController *)baseViewController{
    if(!_baseViewController)
        _baseViewController = [_dataSource baseViewControllerOfSiderView:self];
    return _baseViewController;
}

- (UIViewController *)viewControllerAtIndex:(NSInteger)index{
    return [_dataSource sliderView:self viewControllerAtIndex:index];
}

@end
```