Third Libaray

2016年5月21日 星期六　　下午1:10

[常用框架集合]

Xcode插件

github、CocoaPods

2016年5月16日 星期一

下午2:41

git init 初始化本地仓库

git add . 将文件放入缓存区

git commit -m "提交注释内如"

git push origin master 推送到主分支

git clone 网址

git pull 将远程仓库的最新版本下拉到本地

CocoaPods

1、安装

$ sudo gem install cocoapods

$ pod setup

2、更改repo源

gem sources --remove https://rubygems.org/

gem sources -a https://ruby.taobao.org/

gem sources -l

2、使用镜像索引

3、使用镜像源

pod repo remove master

pod repo add master https://gitcafe.com/akuandev/Specs.git

https://github.com/CocoaPods/Specs.git

pod repo update

pod init 自动创建Podfile

pod install

pod update	->Podfile.lock

pod search

4、使用私有pods

pod 'MyCommon', :podspec => 'https://yuantiku.com/common/myCommon.podspec'

原理：

所有的依赖库都放到一个名为Pods项目中，然后让主项目依赖Pods项目，Pods最终会编译成一个名为libPods.a文件，主项目只需依赖这个.a文件即可。CocoaPods提供一个Pods-resources.sh脚本，项目编译时会执行，将三方库资源复制到目标目录。Pods.xcconfig设置依赖和参数。

5、常见错误，卸载重装

```
sudo rm -fr ~/.cocoapods/repos/master
sudo gem uninstall cocoapods
sudo gem update
sudo gem instal cocoapods
pod setup
```

Cocoa Pods

Masonry

pod 'Masonry'

一个
sh脚

```
mas_makeConstraints
mas_updateConstraints
mas_remakeConstraints

UIEdgeInsets padding = UIEdgeInsetsMake(10, 10, 10, 10);

[view1 mas_makeConstraints:^(MASConstraintMaker *make) {
    make.top.equalTo(superview.mas_top).with.offset(padding.top);
    make.right.equalTo(superview.mas_right).with.offset(-padding.right);
     ......
}];


[view1 mas_makeConstraints:^(MASConstraintMaker *make) {
    make.edges.equalTo(superview).with.insets(padding);
}];
```

make.center.equalTo(ws.view);//将sv居中

make.size.mas_equalTo(CGSizeMake(300, 300));//将size设置成(300,300)

equalTo 和 mas_equalTo的区别

mas_equalTo是一个MACRO，是equalTo的封装 ，CGSize NSNumber UIEdgeinsets equalTo适用于基本数据类型，而mas_equaalTo适用于类似UIEdgeInsetsMake 等复杂类型本上它可以替换equalTo

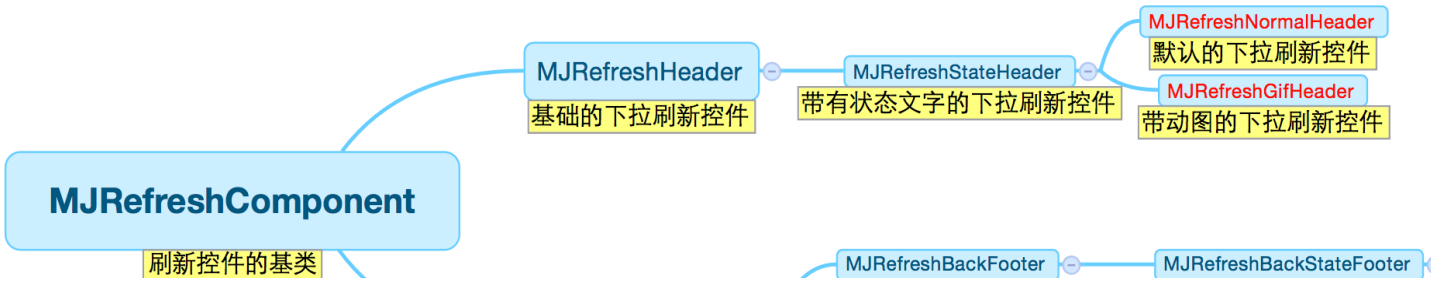
MJRefresh

用法简单的下拉刷新框架

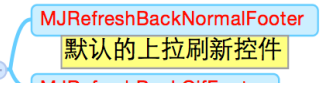
1、支持的控件：UIScrollView、UITableView、UICollectionView、UIWebView

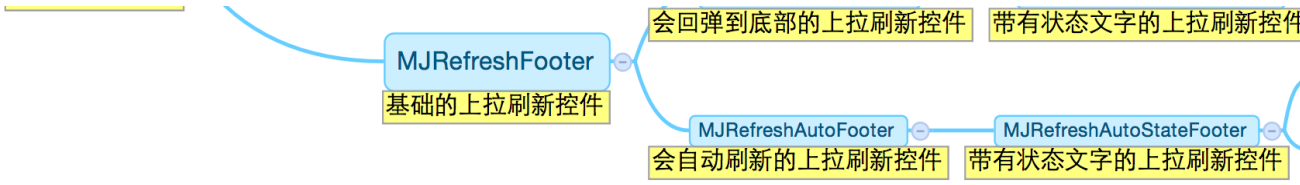
2、pod 'MJRefresh'

3、类结构图



，基





下拉刷新

默认（Normal）：MJRefreshNormalHeader

动图（Gif）：MJRefreshGifHeader

上拉刷新

自动刷新（Auto）：MJRefreshAutoNormalFooter、MJRefreshAutoGifFooter

自动回弹（Back）：MJRefreshBackNormalFooter、MJRefreshBackGifFooter

```
- (void)beginRefreshing;/** 进入刷新状态 */

- (void)endRefreshing;/** 结束刷新状态 */

- (BOOL)isRefreshing;/** 是否正在刷新 */

BOOL automaticallyChangeAlpha;/** 根据拖拽比例自动切换透明度 */
```

创建实例

```
footer/headerWithRefreshingBlock:
footer/headerWithRefreshingTarget: refreshingAction:
- (void)endRefreshingWithNoMoreData; /** 提示没有更多的数据 */
- (void)resetNoMoreData; /** 重置没有更多的数据（消除没有更多数据的状态） */
```

属性

```
lastUpdatedTime /** 上一次下拉刷新成功的时间 */
ignoredScrollViewContentInsetTop /** 忽略多少scrollView的contentInset的top
automaticallyHidden /** 自动根据有无数据来显示和隐藏（有数据显示,无数据隐藏）
automaticallyRefresh /** 是否自动刷新(默认为YES) */
/** 当底部控件出现多少时就自动刷新(默认为1.0，即底部控件完全出现时会自动刷新) *
triggerAutomaticallyRefreshPercent
```

MJRefreshBackGifFooter
带动图的上拉刷新控件
MJRefreshAutoNormalFooter
默认的上拉刷新控件
MJRefreshAutoGifFooter
带动图的上拉刷新控件

```
*/
*/

/
```

设置动画图片

```
// 设置正在刷新状态的动画图片，状态(MJRefreshStateIdle，MJRefreshStatePullin
[header setImages:refreshingImages forState:MJRefreshStateRefreshing];


// 隐藏时间 可以自定义label
header.lastUpdatedTimeLabel.hidden = YES;
// 隐藏状态
header.stateLabel.hidden = YES;
```

SDCycleScrollView

[图片轮播原理]

pod 'SDCycleScrollView','~> 1.64'

```
// 网络加载图片的轮播器
 SDCycleScrollView *cycleScrollView = [cycleScrollViewWithFrame:frame
delegate:delegate placeholderImage:placeholderImage];
 cycleScrollView.imageURLStringsGroup = imagesURLStrings;
// 本地加载图片的轮播器
 SDCycleScrollView *cycleScrollView = [SDCycleScrollView cycleScrollViewW
imagesGroup:图片数组];

// 在iOS 7以后，controller 会对其中唯一的scrollView或其子类调整内边距
self.automaticallyAdjustsScrollViewInsets = NO

// 设置pageControl居右，默认居中
cycleScrollView.pageControlAliment = SDCycleScrollViewPageContolAlimentRight;


// 如果设置title数组，则会在图片下面添加标题
cycleScrollView.titlesGroup = 标题数组（数组元素个数必须和图片数组元素个数保持一致）

// 如需监听图片点击，请设置代理，实现代理方法
cycleScrollView.delegate = ;

// 自定义轮播时间间隔
```

```
g)
```

```
ithFrame:
```

cycleScrollView.autoScrollTimeInterval = ;

IQKeyboardManager

自动键盘回收，配置IQKeyboardManager单利可放在AppDelegate或第一次使用时

```
IQKeyboardManager *manager = [IQKeyboardManager sharedManager];
manager.enable = YES;
manager.shouldResignOnTouchOutside = YES;
manager.shouldToolbarUsesTextFieldTintColor = YES;
manager.enableAutoToolbar = YES;
manager.toolbarManageBehaviour = IQAutoToolbarByTag;
```

设置最后一个输入框的ReturnKey关键字

```
IQKeyboardReturnKeyHandler *retuenKeyHandler =
[[IQKeyboardReturnKeyHandleralloc] initWithViewController:self];
retuenKeyHandler.lastTextFieldReturnKeyType = UIReturnKeyDone;
```

百度地图SDK

申请秘钥key

Key:S2x0Ge2tYnSrekFkolKc5SGY

cn.tarena.test

百度地图集成

POI（Point Of Interest）搜索关键词

流程：给定城市名字和关键词，发送请求，返回结果，在地图上添加大头针

MJExtension

A fast, convenient and nonintrusive conversion between JSON and model.

```
pod 'MJExtension'
```

1、JSON/JSONString -> Model（兼容模型中包含模型）

```
[User objectWithKeyValues:dict]
[User objectWithKeyValues:jsonString]
```

2、模型中有数组属性，数组里又有模型

```
[StatusResult setupObjectClassInArray:^NSDictionary *{
    return @{
                @"statuses" : @"Status",
                // @"statuses" : [Status class],
                @"ads" : @"Ad"
                // @"ads" : [Ad class]
            };
}];
```

3、模型中的属性名和字典中的key不相同（需要多级映射）

```
[Student setupReplacedKeyFromPropertyName:^NSDictionary *{
    return @{
                @"ID" : @"id",//@"属性名":@"字典key"
                @"oldName" : @"name.oldName",
            };
}];
```

4、将字典数组转换为模型数组

```
[User objectArrayWithKeyValuesArray:dictArray]
```

5、模型转字典，模型数组转字典数组

```
NSDictionary *stuDict = stu.keyValues;
[User keyValuesArrayWithObjectArray:userArray]
```

6、Core Data

```
NSManagedObjectContext *context = nil;
User *user = [User objectWithKeyValues:dict context:context];
[context save:nil];
```

7、Coding

```
// what properties not to be coded  忽略不序列化的属性
[User mj_setupIgnoredCodingPropertyNames:^NSArray *{
    return @[@"name"];
}];
[NSKeyedArchiver archiveRootObject:user toFile:file];
[NSKeyedUnarchiver unarchiveObjectWithFile:file];
```

8、过滤字典值（NSString->NSDate,nil->@""）

重写该方法

```
- (id)mj_newValueFromOldValue:(id)oldValue property:(MJProperty *)propert
```

FDFullscreenPopGesture(全屏返回手势)

An UINavigationController's category to enable fullscreen pop gesture in an iOS7+ sys
with AOP.

```
pod 'FDFullscreenPopGesture', '~> 1.1'
```

UINavigationController 管理了串行的 N 个 UIViewController 栈式的 push 和 pop

iOS7 之后为 vc 控制自己的 status bar，UINavigationBar依然是全局的

```
preferredStatusBarStyle
prefersStatusBarHidden
preferredStatusBarUpdateAnimation
```

```
导入工程即可，不用写一行代码
- (void)viewDidLoad
[super viewDidLoad];
self.navigationController.fd_prefersNavigationBarHidden = YES;
}
```

MBProgressHUD

pod 'MBProgressHUD', '~> 0.9.2'

```
hud.removeFromSuperViewOnHide = YES;//隐藏时候从父控件中移除
hud.dimBackground = YES; //YES代表需要蒙版效果
```

用法1：

```
MBProgressHUD *hud = [MBProgressHUD showHUDAddedTo:self.view animated:YES];
hud.mode = MBProgressHUDModeText;
hud.labelText.text = @"网络繁忙，请稍后再试";
hud.margin = 10;
[hud hideAnimated:YES afterDelay:3];
[hud removeFromSuperViewOnHide];
```

用法2：

y

tem style

```
self.hud = [MBProgressHUD showHUDAddedTo:self.view animated:YES];
[self.hud hideAnimated:YES];
```

```
MBProgressHUD *hud = [MBProgressHUD showHUDAddedTo:self.view animated:YES
hud.mode = MBProgressHUDModeAnnularDeterminate;
hud.labelText = @"Loading";
[self doSomethingInBackgroundWithProgressCallback:^(float progress) {
    hud.progress = progress;
    // 从self.view中获取HUDView,如果没有声明变量可以用下面的方式更新进度
    [MBProgressHUD HUDForView:self.view].progress=progress;
} completionCallback:^{
    [hud hide:YES];
}];
```

AFNetworking

```
pod 'AFNetworking', '~> 3.0'
```

AFNetworkReachabilityManager

- AFURLSessionManager
  1、**Download Task**
  `downloadTaskWithRequest`
  2、**Upload Task**
  `uploadTaskWithRequest`
  3、**Upload Task for a Multi-Part Request, with Progress**
  `[AFHTTPRequestSerializer serializer] multipartFormRequestWithMetho`
  `uploadTaskWithStreamedRequest`
  4、**Data Task**
  `[manager dataTaskWithRequest`

- AFHTTPSessionManager
  `is a subclass of `AFURLSessionManager` with convenience methods for making HTTP req`
  `GET`
  `POST`

CocoaLumberjack（替换NSLog -> log4j）

```
];
```

```
d:@"POST"
```

```
uests
```

YYKit (YYKit is a collection of iOS components)

pod 'YYKit'

- YYModel 、YYCache 、YYImage 、YYWebImage 、YYText 、YYKeyboardManager
- YYDispatchQueuePool 、YYAsyncLayer 、YYCategories

JTSImageViewController

An interactive iOS image viewer that does it all: double tap to zoom, flick to dismiss,

图像点击查看大图

HMSegmentedControl、SVSegmentedControl

标签滚动

ARSegmentPager、TwitterCover

tableView下拉、上拉，header图片放大或缩小

NJKWebViewProgress

通过计算需要加载的请求的个数来现实加载进度

webViewDidStartLoad ->webViewDidFinishLoad/didFailLoadWithError

动态控制资源请求数量 finished->+1,error->-1

progress = loadingCount / maxLoadCount

readyState -> interactive 在body中加入一个隐藏的iframe，判断DOM是否渲染完成

et cetera.

readyState -> interactive 往body中加入一个隐藏的iframe，判断DOM是否渲染完成
所有资源渲染完成 -> 进度1.0