iOS 数据库离线缓存思路和网络层封装

2015-11-23 09:10 编辑：lansekuangtu 分类：iOS开发 来源：shelin 投稿 18 4318

数据库 iOS 离线缓存 网络层封装

招聘信息：iOS高级开发工程师（中国排名第一的企业级移动互联网云计算公司 和创科技 红圈营销）

- 作者：shelin 投稿。

一直想总结一下关于iOS的离线数据缓存的方面的问题，然后最近也简单的对AFN进行了再次封装，所有想把这两个结合起来写一下。数据展示型的页面做离线缓存可以有更好的用户体验，用户在离线环境下仍然可以获取一些数据，这里的数据缓存首选肯定是SQLite，轻量级，对数据的存储读取相对于其他几种方式有优势，这里对AFN的封装没有涉及太多业务逻辑层面的需求，主要还是对一些方法再次封装方便使用，解除项目对第三方的耦合性，能够简单的快速的更换底层使用的网络请求代码。这篇主要写离线缓存思路，对AFN的封装只做简单的介绍。

关于XLNetworkApi

XLNetworkApi的一些功能和说明：

- 使用XLNetworkRequest做一些GET、POST、PUT、DELETE请求，与业务逻辑对接部分直接以数组或者字典的形式返回。
- 以及网络下载、上传文件，以block的形式返回实时的下载、上传进度，上传文件参数通过模型XLFileConfig去存取。

热门资讯


【译】17个提升iOS开发效率的必用工具
点击量 11641


iOS开发——UI组件（个人整理）
点击量 9156


Objective-C 编码建议
点击量 7095


我已经写了48年代码了，我感觉我还能写下
点击量 6354


ViewController 瘦身的另一种解决方案
点击量 6088


新手向：五分钟搭建App设置页面_纯代码
点击量 5646

出大事了！A商要联合程序员们打淘宝分京东
点击量 5468


让我们来搞崩 Cocoa 吧 （黑暗代码）
点击量 4892


深入理解 dispatch_queue
点击量 4707


详解 iOS 上机题！附个人见解
点击量 4296

综合评论

数据还有顺序的，而且更新服务器数据也是个问题，建议数据库只存储第一页
longdawei1988 评论了 iOS 数据库离线缓存思路和网络层封装...

为什么要用数据库做缓存
redoca 评论了 iOS 数据库离线缓存思路和网络层封装...

mark
liuyongchun 评论了 UIImagePickerController从拍...

- 通过继承于XLDataService来将一些数据处理，模型转化封装起来，于业务逻辑对接返回的是对应的模型，减少Controllor处理数据处理逻辑的压力。
- 自定义一些回调的block

```
/**
请求成功block
*/
typedef void (^requestSuccessBlock)(id responseObj);
/**
请求失败block
*/
typedef void (^requestFailureBlock) (NSError *error);
/**
请求响应block
*/
typedef void (^responseBlock)(id dataObj, NSError *error);
/**
监听进度响应block
*/
typedef void (^progressBlock)(int64_t bytesWritten, int64_t totalBytesWritten, int64_t totalBytesEx
```

- XLNetworkRequest.m部分实现

```
#import "XLNetworkRequest.h"
#import "AFNetworking.h"
@implementation XLNetworkRequest
+ (void)getRequest:(NSString *)url params:(NSDictionary *)params success:(requestSuccessBlock)succe
//网络不可用
  if (![self checkNetworkStatus]) {
      successHandler(nil);
      failureHandler(nil);
      return;
  }
  AFHTTPRequestOperationManager *manager = [self getRequstManager];
  [manager GET:url parameters:params success:^(AFHTTPRequestOperation * _Nonnull operation, id  _No
      successHandler(responseObject);
  } failure:^(AFHTTPRequestOperation * _Nullable operation, NSError * _Nonnull error) {
      XLLog(@"------请求失败-------%@",error);
      failureHandler(error);
  }];
}
```

- 下载部分代码

```
//下载文件，监听下载进度
+ (void)downloadRequest:(NSString *)url successAndProgress:(progressBlock)progressHandler complete:
  if (![self checkNetworkStatus]) {
```

网络请求前就向数据库读取数据, 数据比网络上的旧怎么办?
zhangslnote 评论了 **iOS 数据库离线缓存思路和网络层封装...**

mark一下
q470 评论了 **必看：游戏开发者必知的21个免费学习资源...**

我去 GitHub 看了你的源码才知道怎么用....声明属性要这
maginawin 评论了 **使用OC链式调用方式简化SpriteKit的动画调...**

这个人可能是看了MJ的视频跑出来写的吧。
sharlier 评论了 **iOS 数据库离线缓存思路和网络层封装...**

理论上说的再直白明了,不懂的人也还是不懂,有人的会认为这是装逼,有的人只
a6270067 评论了 **百度贴吧前负责人：做产品16年，我有9条心得...**

挺不错的文章，虽然现在又很多工具，但自己实现一遍还是很有意思的
小包子_01 评论了 **使用大图+脚本，生成各种size的app icon...**

mark
莫问前程凶吉 评论了 **必看：游戏开发者必知的21个免费学习资源...**

相关帖子

- 今天看到一个好消息，google 2016年2月重返中国!
- 五年土木工程转行iOS
- cocos2d-js中关于xml解析问题
- 关于ZipZap和ZipArchive的使用
- cocos2d-x 3.8 shader 变灰写法
- 坦克大战超难版，玩这么多年没见过这么变态的1990
- 界面返回问题
- 怎样用xib形式的cell上的imageview添加点击事件跳转另一个界面
- 如何不跳出当前应用发短信

```
        progressHandler(0, 0, 0);
        completionHandler(nil, nil);
        return;
    }
    NSURLSessionConfiguration *sessionConfiguration = [NSURLSessionConfiguration defaultSessionConfig
    AFURLSessionManager *manager = [[AFURLSessionManager alloc] initWithSessionConfiguration:sessionC
    NSURLRequest *request = [NSURLRequest requestWithURL:[NSURL URLWithString:url]];
    NSProgress *kProgress = nil;
    NSURLSessionDownloadTask *downloadTask = [manager downloadTaskWithRequest:request progress:&kProg
        NSURL *documentUrl = [[NSFileManager defaultManager] URLForDirectory :NSDocumentDirectory inD
        return [documentUrl URLByAppendingPathComponent:[response suggestedFilename]];
    } completionHandler:^(NSURLResponse * _Nonnull response, NSURL * _Nonnull filePath, NSError * _No
        if (error) {
            XLLog(@"------下载失败-------%@",error);
        }
        completionHandler(response, error);
    }];
    [manager setDownloadTaskDidWriteDataBlock:^(NSURLSession * _Nonnull session, NSURLSessionDownload
        progressHandler(bytesWritten, totalBytesWritten, totalBytesExpectedToWrite);
    }];
    [downloadTask resume];
}
```

- 上传部分代码

```
//上传文件，监听上传进度
+ (void)updateRequest:(NSString *)url params:(NSDictionary *)params fileConfig:(XLFileConfig *)file
    if (![self checkNetworkStatus]) {
        progressHandler(0, 0, 0);
        completionHandler(nil, nil);
        return;
    }
    NSMutableURLRequest *request = [[AFHTTPRequestSerializer serializer] multipartFormRequestWithMeth
        [formData appendPartWithFileData:fileConfig.fileData name:fileConfig.name fileName:fileConfig
    } error:nil];
    //获取上传进度
    AFHTTPRequestOperation *operation = [[AFHTTPRequestOperation alloc] initWithRequest:request];
    [operation setUploadProgressBlock:^(NSUInteger bytesWritten, long long totalBytesWritten, long lo
        progressHandler(bytesWritten, totalBytesWritten, totalBytesExpectedToWrite);
    }];
    [operation setCompletionBlockWithSuccess:^(AFHTTPRequestOperation * _Nonnull operation, id  _Nonn
        completionHandler(responseObject, nil);
    } failure:^(AFHTTPRequestOperation * _Nonnull operation, NSError * _Nonnull error) {
        completionHandler(nil, error);
        if (error) {
            XLLog(@"------上传失败-------%@",error);
        }
```

```
    }];
    [operation start];
}
```

- XLDataService.m部分实现

```
+ (void)getWithUrl:(NSString *)url param:(id)param modelClass:(Class)modelClass responseBlock:(resp
        [XLNetworkRequest getRequest:url params:param success:^(id responseObj) {
        //数组、字典转化为模型数组
        dataObj = [self modelTransformationWithResponseObj:responseObj modelClass:modelClass];
        responseDataBlock(dataObj, nil);
    } failure:^(NSError *error) {
        responseDataBlock(nil, error);
    }];
}
```

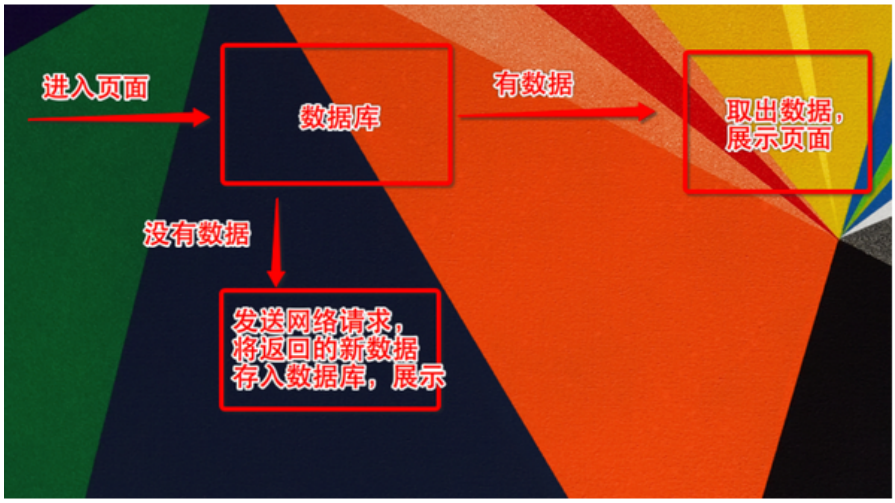
- （关键）下面这个方法提供给继承XLDataService的子类重写，将转化为模型的代码写在这里，相似业务的网络数据请求都可以用这个子类去请求数据，直接返回对应的模型数组。

```
/**
数组、字典转化为模型
*/
+ (id)modelTransformationWithResponseObj:(id)responseObj modelClass:(Class)modelClass {
      return nil;
}
```

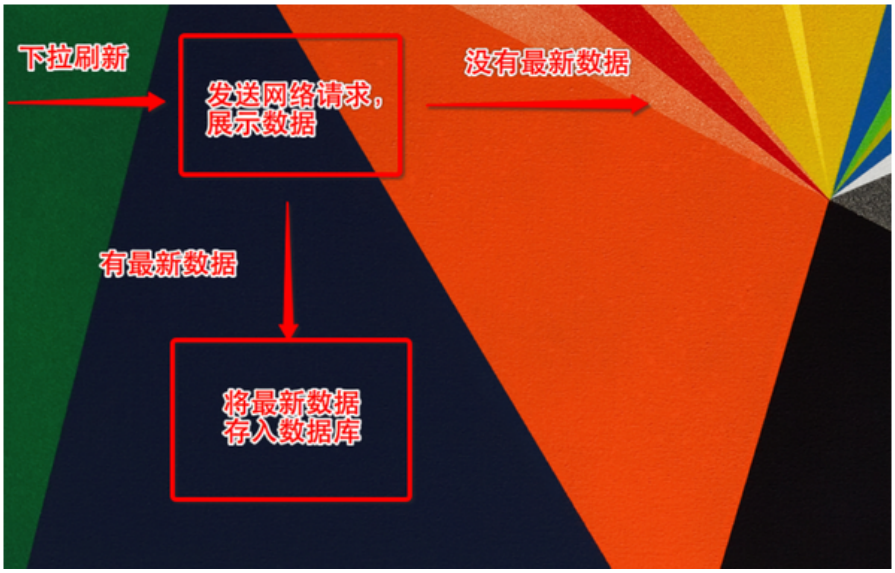
关于离线数据缓存

当用户进入程序的展示页面，有三个情况下可能涉及到数据库存取操作，简单画了个图来理解，思路比较简单，主要是一些存取的细节处理。

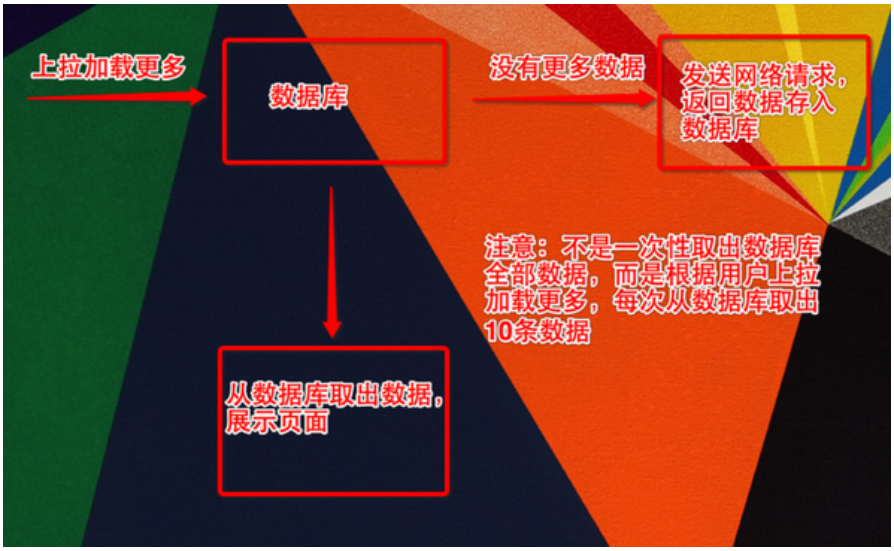
- 进入展示页面



下拉刷新最新数据



上拉加载更多数据



- 需要注意的是，上拉加载更多的时候，每次从数据库返回一定数量的数据，而不是一次性将数据全部加载，否则会有内存问题，直到数据库中没有更多数据时再发生网络请求，再次将新数据存入数据库。这里存储数据的方式是将服务器返回每组数据的字典归档成二进制作为数据库字段直接存储，这样存储在模型属性比较多的情况下更有好处，避免每一个属性作为一个字段，另外增加了一个idStr字段用来判断数据的唯一性，避免重复存储。
- 首先定义一个工具类XLDataBase来做数据库相关的操作，这里用的是第三方的FMDB。

```
#import "XLDataBase.h"
#import "FMDatabase.h"
#import "Item.h"
#import "MJExtension.h"
@implementation XLDataBase
static FMDatabase *_db;
+ (void)initialize {
    NSString *path = [NSString stringWithFormat:@"%@/Library/Caches/Data.db",NSHomeDirectory()];
    _db = [FMDatabase databaseWithPath:path];
    [_db open];
```

```
    [_db executeUpdate:@"CREATE TABLE IF NOT EXISTS t_item (id integer PRIMARY KEY, itemDict blob N
}
//存入数据库
+ (void)saveItemDict:(NSDictionary *)itemDict {
    //此处把字典归档成二进制数据直接存入数据库，避免添加过多的数据库字段
    NSData *dictData = [NSKeyedArchiver archivedDataWithRootObject:itemDict];
    [_db executeUpdateWithFormat:@"INSERT INTO t_item (itemDict, idStr) VALUES (%@, %@)",dictData,
}
//返回全部数据
+ (NSArray *)list {
    FMResultSet *set = [_db executeQuery:@"SELECT * FROM t_item"];
    NSMutableArray *list = [NSMutableArray array];
    while (set.next) {
        // 获得当前所指向的数据
        NSData *dictData = [set objectForColumnName:@"itemDict"];
        NSDictionary *dict = [NSKeyedUnarchiver unarchiveObjectWithData:dictData];
        [list addObject:[Item mj_objectWithKeyValues:dict]];
    }
    return list;
}
//取出某个范围内的数据
+ (NSArray *)listWithRange:(NSRange)range {
    NSString *SQL = [NSString stringWithFormat:@"SELECT * FROM t_item LIMIT %lu, %lu",range.locatio
    FMResultSet *set = [_db executeQuery:SQL];
    NSMutableArray *list = [NSMutableArray array];
    while (set.next) {
        NSData *dictData = [set objectForColumnName:@"itemDict"];
        NSDictionary *dict = [NSKeyedUnarchiver unarchiveObjectWithData:dictData];
        [list addObject:[Item mj_objectWithKeyValues:dict]];
    }
    return list;
}
//通过一组数据的唯一标识判断数据是否存在
+ (BOOL)isExistWithId:(NSString *)idStr
{
    BOOL isExist = NO;
    FMResultSet *resultSet= [_db executeQuery:@"SELECT * FROM t_item where idStr = ?",idStr];
    while ([resultSet next]) {
        if([resultSet stringForColumn:@"idStr"]) {
            isExist = YES;
        }else{
            isExist = NO;
        }
    }
    return isExist;
}
```

```
@end
```

- 一些继承于XLDataService的子类的数据库存储和模型转换的逻辑代码

```
#import "GetTableViewData.h"
#import "XLDataBase.h"
@implementation GetTableViewData
//重写父类方法
+ (id)modelTransformationWithResponseObj:(id)responseObj modelClass:(Class)modelClass {
    NSArray *lists = responseObj[@"data"][@"list"];
    NSMutableArray *array = [NSMutableArray array];
    for (NSDictionary *dict in lists) {
        [modelClass mj_setupReplacedKeyFromPropertyName:^NSDictionary *{
            return @{ @"ID" : @"id" };
        }];
        [array addObject:[modelClass mj_objectWithKeyValues:dict]];
        //通过idStr先判断数据是否存储过，如果没有，网络请求新数据存入数据库
        if (![XLDataBase isExistWithId:dict[@"id"]]) {
            //存数据库
            NSLog(@"存入数据库");
            [XLDataBase saveItemDict:dict];
        }
    }
    return array;
}
```

下面是一些控制器的代码实现：

```
#import "ViewController.h"
#import "GetTableViewData.h"
#import "Item.h"
#import "XLDataBase.h"
#import "ItemCell.h"
#import "MJRefresh.h"
#define URL_TABLEVIEW @"https://api.108tian.com/mobile/v3/EventList?cityId=1&step=10&theme=0&page=%
@interface ViewController () {
    NSMutableArray *_dataArray;
    UITableView *_tableView;
    NSInteger _currentPage;//当前数据对应的page
}
@end
@implementation ViewController
#pragma mark Life cycle
- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup after loading the view, typically from a nib.
    [self createTableView];
    _dataArray = [NSMutableArray array];
```

```
}
- (void)viewWillAppear:(BOOL)animated {
    [super viewWillAppear:animated];
    NSRange range = NSMakeRange(0, 10);
    //如果数据库有数据则读取，不发送网络请求
    if ([[XLDataBase listWithRange:range] count]) {
        [_dataArray addObjectsFromArray:[XLDataBase listWithRange:range]];
        NSLog(@"从数据库加载");
    }else{
        [self getTableViewDataWithPage:0];
    }
}
#pragma mark UI
- (void)createTableView {
    _tableView = [[UITableView alloc] initWithFrame:self.view.bounds];
    _tableView.delegate = self;
    _tableView.dataSource = self;
    _tableView.rowHeight = 100.0;
    [self.view addSubview:_tableView];
    _tableView.mj_header = [MJRefreshNormalHeader headerWithRefreshingBlock:^{
        [self loadNewData];
    }];
    _tableView.mj_footer = [MJRefreshAutoNormalFooter footerWithRefreshingBlock:^{
        [self loadMoreData];
    }];
}
#pragma mark GetDataSoure
- (void)getTableViewDataWithPage:(NSInteger)page {
    NSLog(@"发送网络请求！");
    NSString *url = [NSString stringWithFormat:URL_TABLEVIEW, page];
    [GetTableViewData getWithUrl:url param:nil modelClass:[Item class] responseBlock:^(id dataObj,
        [_dataArray addObjectsFromArray:dataObj];
        [_tableView reloadData];
        [_tableView.mj_header endRefreshing];
        [_tableView.mj_footer endRefreshing];
    }];
}
- (void)loadNewData {
    NSLog(@"下拉刷新");
    _currentPage = 0;
    [_dataArray removeAllObjects];
    [self getTableViewDataWithPage:_currentPage];
}
- (void)loadMoreData {
    NSLog(@"上拉加载");
    _currentPage ++;
```

```
    NSRange range = NSMakeRange(_currentPage * 10, 10);
    if ([[XLDataBase listWithRange:range] count]) {
        [_dataArray addObjectsFromArray:[XLDataBase listWithRange:range]];
        [_tableView reloadData];
        [_tableView.mj_footer endRefreshing];
        NSLog(@"数据库加载%lu条更多数据",[[XLDataBase listWithRange:range] count]);
    }else{
        //数据库没更多数据时再网络请求
        [self getTableViewDataWithPage:_currentPage];
    }
}
#pragma mark UITableViewDataSource
- (NSInteger)tableView:(UITableView *)tableView numberOfRowsInSection:(NSInteger)section {
    return _dataArray.count;
}
- (UITableViewCell *)tableView:(UITableView *)tableView cellForRowAtIndexPath:(NSIndexPath *)indexP
    ItemCell *cell = [ItemCell itemCellWithTableView:tableView];
    cell.item = _dataArray[indexPath.row];
    return cell;
}
@end
```

最后附上代码的下载地址，重要的部分代码中都有相应的注释和文字打印，运行程序可以很直观的表现。

https://github.com/ShelinShelin/OffLineCache.git

希望大家能提出一些意见，很乐意与大家互相交流。

上一篇：源码推荐(11.20)：自定义视频播放器，模仿智联招聘滑动字母表动画

相关资讯

详解 iOS 上机题！附个人见解
使用大图+脚本，生成各种size的app icon和图片素材
iOS系统的这些小功能会影响到你的APP哦~你测了没呢?
【译】详细讲述iOS自定义转场
iOS hybrid App 的实现原理及性能监测
详述iOS国际化
watchOS 2教程(一):开始吧
iOS 9的 Core Image新滤镜
iOS 开发之 ReactiveCocoa 下的 MVVM（干货分享）
【译】17个提升iOS开发效率的必用工具

我来说两句

您还没有登录！请 登录 或 注册

所有评论（18）

longdawei1988 5分钟前

数据还有顺序的，而且更新服务器数据也是个问题，建议数据库只存储第一页的内容，可以参考网易新闻的做法。

0 0 回复

redoca 10分钟前

为什么要用数据库做缓存

0 0 回复

zhangslnote 2015-11-23 12:32:51

网络请求前就向数据库读取数据, 数据比网络上的旧怎么办?

1 0 回复

sharlier 2015-11-23 10:20:05

这个人可能是看了MJ的视频跑出来写的吧。

0 1 回复

ymoo_ 2015-11-23 07:09:54

下载文件进度那里有问题，如果我同时有多个下载任务呢？ [manager setDownloadTaskDidWriteDataBlock:^(NSURLSession * _Nonnull session, NSURLSessionDownloadTask * _Nonnull downloadTask, int64_t bytesWritten, int64_t totalBytesWritten, int64_t totalBytesExpectedToWrite) { progressHandler(bytesWritten, totalBytesWritten, totalBytesExpectedToWrite); }]; 这样是不对的。因为这个方法是manager的，manager是个单例。这个进度回调永远都是你最后调用下载方法那个的进度回调。 正确的应该是在你调用你封装的下载方法里面把进度block以downloadTaskId为键缓存起来（比如字典progressDic），在你这个单例初始化的时候调用setDownloadTaskDidWriteDataBlock这个方法，在这个方法的block里面根据downloadTaskId去 progerssDic里面取下载任务的进度回调然后调用。下载完成后把这个进度block从progressDic里面移除就是了。

0 0 回复

wsw_ios 2015-11-23 06:34:11

[modelClass mj_setupReplacedKeyFromPropertyName:^NSDictionary *{ return @{ @"ID" : @"id" }; }];这段干什么用的?

0 0 回复

danxinni 2015-11-23 06:04:14

感觉有个问题啊。 后台数据更新，而触底加载取得是本地缓存，并非实时数据。是不是应该做一个请求，来判断后台数据是否有更新？ 觉得这个缓存可以用在聊天历史功能上，因为一般聊天记录是不能更改的，只能增加。

1 0 回复

xiaodu1233 2015-11-23 05:04:48

我之前也做个了sqlite的离线换存，但是怎么搞数据缓存的时候创建不同的表或者库网布太懂，数据库懂得少

0 0 回复

xiaodu1233 2015-11-23 04:39:00

henhaogood!!!!!!!!!!!!!!!!!!!!!!!!

0 0 回复

qq272600 2015-11-23 04:03:34

mark

0 0 回复

amor猫猫 2015-11-23 03:56:09

当数据库有数据时就不发送网络请求，如何保证我进入程序时数据是最新的?

0 0 回复

sharlier 2015-11-23 04:16:26

回复：amor猫猫 后台可以来个时间标记，你可以对比一下，如果有最新的，就刷新最新的插入

0 0 回复

amor猫猫 2015-11-23 05:35:50

回复：sharlier 好的，谢谢~

0 0 回复

a2341178s 2015-11-23 03:17:23

给力，已收藏

0 0 回复

陌路叶子 2015-11-23 02:56:58

数据显示出来如果需要有一定的排序组织，文章里的做法是不是不方便了?

0 0 回复

glbfor 2015-11-23 04:01:32

回复：陌路叶子 这种情况下，在处理回调的前面加一个拦截器方法，然后对数据处理一下再返回就是了。

0 0 回复

ai罗 2015-11-23 02:53:32

不错，值得学习，6666

0 0 回复

allen1324 2015-11-23 03:15:05

回复：ai罗 NICE，正好需要这个功能，学习学习

0 0 回复