

Cocos开发者平台Cocos引擎中文官网Cocos商店

耀敬业我的帖子我的问答退出

任务(2)权限积分隐私帐号



首页

资讯

代码库

工具库

论坛

招聘

活动

每日签到

论坛搜索

OS X开发

Swift专区

iOS开发

App Store经验交流

美术区

游戏开发区

议事大厅

我的主题我的回复我的收藏好友近况书签

好友邀请会员排行社区应用最新帖子精华区社区服务会员列表统计排行平板模式

热门标签: ios 开发 问题 app 游戏

CocoaChina » iOS开发 » 项目功能大全，让你的项目一天搞定（给新手们一点干货）RSS

上一主题 下一主题 « 1 2 3 4 » Pages: 1/4 Go

主题：项目功能大全，让你的项目一天搞定（给新手们一点干货）使用道具 回复

小耗子



级别: 新手上路
状态: 未签到 - [1天]
UID: 492973
精华: 0
发帖: 2
可可豆: 28 CB
威望: 17 点
在线时间: 21(时)
注册时间: 2015-07-28
最后登录: 2015-09-15

楼主：发表于: 2015-08-22 16:29 发自: Web Page



倒序阅读 | 只看楼主

项目功能大全，让你的项目一天搞定（给新手们一点干货）来源于 已删除

工作了好久一直很忙，好不容易有些时间，今天有人问我如何开始写一个项目，不禁回忆起自己写第一个项目的时候，到现在，突然感觉自己写的还是在重复，有些感想，特此写下这篇文章，给想入这行的新手们一些参考

1.写项目之前首先我们要确定需求，明确项目需要实现哪些功能（吐槽下：好多项目的大部分功能都是一样的）

2，美工，后台啥的我都直不一一说了情况都不一样

废话不说了直接开写程序

很重要（1）**首先我们要确定明确开发需要的框架，一个好的框架可以让我们轻松很多**

框架可以自己写，也可以从网上找，多对比一下会有惊喜的

页面

（1）一般每个详情页面都有相应的导航栏;如果有直接参考（见导航Deno，直接输入图片网址或者本地图片，设置坐标一键搞定）；如果没有直接跟

（2）然后就到了图文布局了：（推荐xib或storyboard）强大快速（尤其项目比较急的时候更是不二选择）；不熟练的话那就老实的算坐标，布局吧

情况太多，一般难度也不大，费点心基本都可以搞定（我在这里就不详细说了。）

我这里就按功能说了（不需要就直接跳过）

功能

搜索

第一种搜索：

UISearchBar 遵守协议

//将要进入编辑模式调用

- (BOOL)searchBarShouldBeginEditing:(UISearchBar *)searchBar {

//显示cancel 按钮

[searchBar setShowsCancelButton:YES animated:YES];

return YES;

}

//将要退出编辑模式调用

- (BOOL)searchBarShouldEndEditing:(UISearchBar *)searchBar {

1 | 请把代码粘贴在这里

//隐藏cancel 按钮

[searchBar setShowsCancelButton:NO animated:YES];

return YES;

}

```
// 点击 搜索按钮的时候调用
- (void)searchBarSearchButtonClicked:(UISearchBar *)searchBar {
    //写入搜索内容（根据需要进行搜索）

}

//点击cancel 被调用
- (void)searchBarCancelButtonClicked:(UISearchBar *)searchBar {
    //清空内容
    searchBar.text = @"";
    [searchBar resignFirstResponder]; //收键盘
}

第二种搜索
iOS8新出的感觉很强大，大家以后尽量都用它吧，紧跟版本呀！
    UISearchController：它的其中一个属性就是searchBar
需要先设置它的搜索结果视图
//nil为和当前视图总用一个视图
self.searchVC = [[UISearchController alloc] initWithResultsController:nil];
//如果不需要刻意再创建一个如：
UITableViewController *tableVC = [[UITableViewController alloc] initWithStyle:UITableViewStylePlain];

    tableVC.tableView.delegate = self;
    tableVC.tableView.dataSource = self;
    [tableVC.tableView registerClass:[UITableViewCell class] forCellReuseIdentifier:@"Cell"];
self.searchVC = [[UISearchController alloc] initWithResultsController:tableVC];

注意需要两个协议
//搜索协议
delegate
//更新搜索内容的代理
searchResultsUpdater
//必须要加上 自适应才能显示 搜索条
[self.searchVC.searchBar sizeToFit];
UISearchResultsUpdating 协议
- (void)updateSearchResultsForSearchController:(UISearchController *)searchController {
    NSLog(@"searchBar 更新内容");
//搜索内容
    //检索推荐谓词检索，快准狠有没有！详细的去百度吧度娘威武！

//最后搜索数据源变了 ->让搜索控制器 内部的结果视图控制器的tableView的刷新
    UITableViewController *tableVC = (UITableViewController *)searchController.searchResultsController;
    [tableVC.tableView reloadData];
}

- (void)willPresentSearchController:(UISearchController *)searchController {
    NSLog(@"searchController 将要 显示");
}

- (void)didPresentSearchController:(UISearchController *)searchController {
    NSLog(@"searchController 已经 显示");
}

- (void)willDismissSearchController:(UISearchController *)searchController {
    NSLog(@"searchController 将要 消失");
}

- (void)didDismissSearchController:(UISearchController *)searchController {
    NSLog(@"searchController 已经 消失");
}

！！如果需要页面跳转
在进行页面跳转的时候要注意现在有两个视图呀需要区分开
UIViewController *vc = nil;
```

```

//self.presentedViewController获取已经模态跳转上册的视图控制器，如果dismiss 之后 这个值会变成nil
if (self.presentedViewController) {
    //判断一下 当前视图控制器有没有 模态跳转 的视图，如果有 那么 做另外一个模态跳转的时候 应该用 上一个已经模态跳转的控制器进行 模态跳
    个
    vc = self.presentedViewController;
}else {
    vc = self;
}
//模态跳转
[vc presentViewController:animated:YES completion:nil];

```

下载

首先判断是否已经下载过，然后告知服务器从哪里下载，

(1) 下载前需要先确定路径

获取文件在沙盒中Documents下的全路径

//我们把url作为文件名-> 但是url 中可能存在一些非法字符不能作为文件名，这时我们可以用md5 对文件名进行加密 产生一个唯一的字符串 (十六数字+A-F表示)，这样就可以保证文件名不出现非法字符

```

NSString *fileName = [url MD5Hash]; //MD5
//获取Documents
NSString *docPath = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES)lastObject];
//拼接路径
NSString *filePath = [docPath stringByAppendingPathComponent:fileName];
NSLog(@"path:%@",filePath);
***这里需要用到OC文件管理的知识！
//创建文件（首先检测有没有存在，如果没有在创建）
if ([[NSFileManager defaultManager] fileExistsAtPath:filePath]) {
    //检测文件是否存在
    //不存在那么要创建
    //NSFileManager 文件管理句柄
    [[NSFileManager defaultManager] createFileAtPath:filePath contents:nil attributes:nil];
}
//如果已存在获取已近下载的大小，如果不存在那么大小为0；
NSDictionary *fileDict = [[NSFileManager defaultManager] attributesOfItemAtPath:filePath error:nil];
//保存已经下载文件的大小
self.loadedFileSize = fileDict[NSFileSize];
//下载前需要打开文件
self.fileHandle = [NSFileHandle fileHandleForWritingAtPath:filePath];
*****(如果服务器支持可变断点续传可以照用，如果不支持请忽略 头域)
//把文件大小告知服务器
//创建可变请求 增加请求头
NSMutableURLRequest *request = [NSMutableURLRequest requestWithURL:[NSURL URLWithString:url]];
//增加头域 告知服务器 从 哪个字节之后开始下载（不了解头域的还是那句话百度）， 不支持头域的可以直接跳过
[request setValue:[NSString stringWithFormat:@"%bytes=%llu-",fileSize] forHTTPHeaderField:@"Range"];
//创建请求连接 开始异步下载
NSURLSession *session = [NSURLSession alloc] initWithRequest:request delegate:self;
NSURLSessionDataTask *task = [session dataTaskWithURL:[NSURL URLWithString:url] completionHandler:^(NSData *data, NSURLResponse *response, NSError *error) {
    //接收服务器响应
    - (void)connection:(NSURLSessionConnection *)connection didReceiveResponse:(NSURLSessionResponse *)response {
        在这里我们可以计算文件的总大小，获取数据的类型： httpResponse.MIMETYPE， 数据的大小： NSHTTPURLResponse *httpResponse = (NSHTTPURLResponse *)response
        NSLog(@"url:%@",httpResponse.URL.absoluteString);
        //计算文件总大小 = 已经下载的+服务器将要发的
        ( self.loadedFileSize和上面关联着 数据是一段一段下载的)
        self.totalFileSize = self.loadedFileSize+httpResponse.expectedContentLength;
    }
}];
//接收数据过程 一段一段接收
- (void)connection:(NSURLSessionConnection *)connection didReceiveData:(NSData *)data {
    //都是OC文件管理的知识，我就不细说了吧
    //下载一段 写一段数据
    //先把文件偏移量定位到文件尾
    [_fileHandle seekToEndOfFile];

```

```

//写文件
[_fileHandle writeData:data];
//立即同步到磁盘
[_fileHandle synchronizeFile];
//记录已经下载数据大小
self.loadedFileSize += data.length
}
//下载完成一定要关闭呀
- (void)connectionDidFinishLoading:(NSURLConnection *)connection {
    [self stopDownload];//停止下载
}
//下载失败也要关闭呀
- (void)connection:(NSURLConnection *)connection didFailWithError:(NSError *)error {
    [self stopDownload];
}
- (void)stopDownload {
    if (_httpRequest) {
        [_httpRequest cancel];
        _httpRequest = nil;
    }
    [_fileHandle closeFile];//关闭文件
}
}

收藏（关注）
这和数据库有关联（数据库下面我会说的，不懂得也可以先看后面的数据库）
点击收藏按钮的时候相当于在数据库里面增加了一条数据
和关注按钮基本一样，有时候只是表现形似不同罢了本质都是一样
这里我用的是DBManager
//关于数据库的方法，自己写的，就是这里面记录的都是收藏过得数据
//获取所有的收藏过得数据，放在数组里
self.favoriteArr = [[DBManager sharedManager]fetchall];
然后遍历数组获取相关图片的的URL下载图片
for (int i = 0; i<_self.favoriteArr.count; i++) {
    AppModel *model = _favoriteArr;
//我这里建个button 来显示收藏（具体问题具体分析说白了就是把你收藏的东西展现出来）图片下载用的是SDImage第三方库
    UIButton *btn = [UIButton buttonWithType:UIButtonTypeCustom];
    [btn sd_setImageWithURL:[NSURL URLWithString:model.iconUrl] forState:UIControlStateNormal placeholderImage:[UIImage imageNamed:@"_candou"]];
    。
    。
    。
}

登录和注册（每个App必备的功能）

分为，自身登录*注册（自己起的名字知道大概意思就行了哈，别在意哈）和第三方登录（微博，QQ，微信，等外国的不常用，基本就是那几个社
忘了还有人人（话说没多少人用了吧），第三方遵循的基本原则：那个人多就用那个）
主要说下自身登录*注册
大多数登录和注册都用的是post请求{这里需要我们和做服务器的协调好，登陆成功，登录失败返回什么
我们根据这些来提示用户是否登陆成功，！
如果用户登录成功我们需要记录用户登录成功这个状态，以防止用户多次登录，重复登录
这时我们需要定义一个全局变量
在登录成功后记录登录状态
如
extern：引入外部变量
extern BOOL isLogin;
        isLogin=YES;

在其他页面再需要登录的先判断登录状态，如果为YES就不需要在登陆了，如果没有提示用户需要登录后才可以进入
}
第三方登录直接到开放平台下demo；

```

清除缓存

这里我用的SDImage库的

```
#import "UIImageView+WebCache.h"
```

```
-(double)getCachesSize
```

```
{
```

```
//SDImage 缓存
```

```
NSInteger sdfileSize=[[SDImageCache sharedImageCache]getSize]; NSString*caches=[NSSearchPathForDirectoriesInDomains(NSCachesDirectoryDomainMask, YES)]lastObject];
```

```
NSString*mycacehs=[caches stringByAppendingPathComponent:@"MyCaches"];
```

```
NSDirectoryEnumerator*enumor=[[NSFileManager defaultManager]enumeratorAtPath:mycacehs];
```

```
NSUInteger mysize=0;
```

```
for (NSString *filename in enumor) {
```

```
    NSString*filepath=[mycacehs stringByAppendingPathComponent:filename];
```

```
    NSDictionary*filedict=[[NSFileManager defaultManager]attributesOfItemAtPath:filepath error:nil];
```

```
    //自身缓存
```

```
    mysize+=filedict.fileSize;
```

```
}
```

```
return (mysize+sdfileSize)/1024.0/1024.0;
```

```
}
```

分享

：（系统的和第三方的： 第三方的推荐友盟（简单呀两句话sdk:直接抄不用改省事哈哈，不过就一个微博可以用，如果想用QQ，微信，等，自有大的去对着文档来吧，））

系统的：

协议：MFMessageComposeViewControllerDelegate（信息）,MFMailComposeViewControllerDelegate（邮箱）

1（信息）

//需要真机，虚拟机没效果

```
if ([MFMessageComposeViewController canSendText]) {
```

//检测 当前应用 是否支持短信功能

//支持的话 创建 具有短信模块的界面

```
MFMessageComposeViewController *message = [[MFMessageComposeViewController alloc] init];
```

```
    //设置联系人(可以群发)
```

```
    message.recipients = @[@"10086",@"10011"];
```

```
    //设置短信的内容
```

```
    message.body = [NSString stringWithFormat:@"快来下载，这里有惊喜-%@",@"网址"];
```

```
    message.messageComposeDelegate = self;
```

```
    //模态跳转(内部有导航)
```

```
    [self presentViewController:message animated:YES completion:nil];
```

```
}
```

//协议

```
-(void)messageComposeViewController:(MFMessageComposeViewController *)controller didFinishWithResult:(MessageComposeResult)result {
```

```
    switch (result) {
```

```
        case MessageComposeResultCancelled:
```

```
        {
```

```
            NSLog(@"取消");
```

```
        }
```

```
        break;
```

```
        case MessageComposeResultSent:
```

```
        {
```

```
            NSLog(@"短信已发送");
```

```
        }
```

```
        break;
```

```
        case MessageComposeResultFailed:
```

```
        {
```

```
            NSLog(@"短信失败");
```

```
        }
```

```
        break;
```

```
    default:
```

```

        break;
    }
    //最后要模态跳转返回
    [controller dismissViewControllerAnimated:YES completion:nil];
}

(2) 邮箱
if ([MFMailComposeViewController canSendMail]) {
    //检测是否支持邮箱功能
    //如果支持 创建界面
    MFMailComposeViewController *mail = [[MFMailComposeViewController alloc] init];
    //设置联系人
    [mail setToRecipients:@[@"xxxxx@qq.com",@"zzzz@163.com"]];
    //设置抄送
    [mail setCcRecipients:@[@"xxx@sina.com"]];
    //设置标题
    [mail setSubject:@"分享爱限免应用"];
    //设置内容
    NSString *str = [NSString stringWithFormat:@"点击有惊喜:%@",@"网址"];
    //第二个参数 是否以HTML格式
    [mail setMessageBody:str isHTML:YES];

    //添加附件
    NSData *data = UIImagePNGRepresentation([UIImage imageNamed:@"account_candou"]);
    //第一个参数 文件二进制 2 文件的类型 3 文件的名称
    [mail addAttachmentData:data mimeType:@"image/png" fileName:@"account_candou"];

    //设置代理
    mail.mailComposeDelegate = self;
    //模态跳转
    [self presentViewController:mail animated:YES completion:nil];
}

//协议
- (void)mailComposeController:(MFMailComposeViewController *)controller didFinishWithResult:(MFMailComposeResult)result error:(NSError *)error {
    switch (result) {
        case MFMailComposeResultCancelled:
            NSLog(@"邮件取消");
            break;
        case MFMailComposeResultSaved:
            NSLog(@"邮件保存");
            break;
        case MFMailComposeResultSent:
            NSLog(@"邮件发送");
            break;
        case MFMailComposeResultFailed:
            NSLog(@"邮件失败");
            break;

        default:
            break;
    }
    //模态跳转返回
    [self dismissViewControllerAnimated:YES completion:nil];
}

//友盟的(不懂可以看官网)建立使用UM的，系统的太坑了，在咱们这没人用呀
//协议
UMSocialUIDelegate
[UMSocialSnsService presentSnsIconSheetView:self appKey:@"507fcab25270157b37000010" shareText:str shareImage:[UIImage imageNamed:@"nt_candou"] shareToSnsNames:[NSArray arrayWithObjects:UMShareToSina,UMShareToSms,UMShareToEmail,UMShareToWechatTimeline,nil] delegate:self];
//（写了这么多也就前三个有用微信需要自己去注册）
在appdelegate.m中（其他的，哎都是泪看文档把，你妹的呀）

```

```

//初始化UM
- (void)initUM {

    //初始化
    [UMSocialData setAppKey:@"507fcab25270157b37000010"];
}

地图
又到了都是泪的地方，文档走起吧，大苹果太渣
推荐百度，高德，腾讯（百度最好，但是那啥注册一把泪呀）
（1）定位：
//1.头文件
#import <CoreLocation/CoreLocation.h>
协议 CLLocationManagerDelegate
//必须要强引用 在定位之前不能释放
@property (nonatomic,strong) CLLocationManager *manager;
kCLLocationAccuracyBestForNavigation -->最好的精度 用于导航
kCLLocationAccuracyBest;//精度高的
kCLLocationAccuracyNearestTenMeters; 10m
kCLLocationAccuracyHundredMeters; 100m
kCLLocationAccuracyKilometer; 1000m
kCLLocationAccuracyThreeKilometers; 3000m
//精度越高 越耗电
- (void)initLocationManager {
    //用的时候才创建 懒加载
    if (!self.manager) {
        //实例化 管理器
        self.manager = [[CLLocationManager alloc] init];
        //设置精度类型
        self.manager.desiredAccuracy = kCLLocationAccuracyBest;
        //设置 精度的大小
        self.manager.distanceFilter = 10;
        //获取定位的数据 必须要设置代理
        self.manager.delegate = self;
        //iOS8之后 必须要向用户申请授权
        /*
        1.在Info.plist中 添加选项
        Privacy - Location Usage Description(可选)
        NSLocationAlwaysUsageDescription(和代码要对应)
        或者
        NSLocationWhenInUseUsageDescription(代码要对应)
        2.在代码中 添加
        [self.manager requestAlwaysAuthorization];
        或者
        [self.manager requestWhenInUseAuthorization];
        */

        double v = [UIDevice currentDevice].systemVersion.doubleValue;
        if (v >= 8.0) { //判断版本
            //设置一个就可以
            //始终允许授权打开定位 (前后台都允许)
            [self.manager requestAlwaysAuthorization];
            //使用的时候允许 (前台允许)
            [self.manager requestWhenInUseAuthorization]; //
            //允许之后 第一次 会弹出一个警告框 选择允许
        }
        //下面的条件编译也可以 判断当前系统 版本
        #ifdef __IPHONE_8_0 //如果定义过这个宏 __IPHONE_8_0，iOS8.0之后就会定义宏 __IPHONE_8_0

        #endif

    }
}

```

```
}
//开始定位
- (void)startLocation:(UIBarButtonItem *)item {
    //判断是否具备定位功能
    if ([CLLocationManager locationServicesEnabled]) {
        //懒加载管理器
        [self initLocationManager];
        //开始定位
        [self.manager startUpdatingLocation];
    }
}
//停止定位
- (void)stopLocation:(UIBarButtonItem *)item {
    [self.manager stopUpdatingLocation];
}
定位协议
//当定位的位置 发生改变的时候 一会一直调用
//会把定位的地理位置 传入
//locations 存放的就是地理位置
//数组中就一个元素
//
- (void)locationManager:(CLLocationManager *)manager didUpdateLocations:(NSArray *)locations {
    if (locations.count) {
        //获取定位 位置
        CLLocation *location = [locations lastObject];
        //得到经纬度
        CLLocationCoordinate2D coordinate = location.coordinate;
        //打印经纬度
        NSLog(@"location:%f %f",coordinate.longitude,coordinate.latitude);
        //地理反编码
        //把经纬度装化为具体的地址
    #if 0
        [self reverseGeocoderWithBaidu:coordinate];
    #else
        [self reverseGeocoderWithSystem:location];
    #endif

    }
}
//百度的
//需要OCJson解析
- (void)reverseGeocoderWithBaidu:(CLLocationCoordinate2D)coordinate {
    //用多线程 异步下载
    dispatch_async(dispatch_get_global_queue(DISPATCH_QUEUE_PRIORITY_DEFAULT, 0), ^{
        NSData *data = [NSData dataWithContentsOfURL:[NSURL URLWithString:[NSString stringWithFormat:kPathUrl,coordinate.latitude,coordinate.longitude]]];
        //json解析
        NSDictionary *dict = [NSJSONSerialization JSONObjectWithData:data options:NSJSONReadingMutableContainers error:nil];
        NSDictionary *resultDict = dict[@"result"];
        NSLog(@"address:%@",resultDict[@"formatted_address"]);//获取地址
    });
}
//系统 地理反编码
- (void)reverseGeocoderWithSystem:(CLLocation *)location {
    //创建对象
    CLGeocoder *geocoder = [[CLGeocoder alloc] init];
    //根据location内部的经纬度 进行 地理反编码
    [geocoder reverseGeocodeLocation:location completionHandler:^(NSArray *placemarks, NSError *error) {
        //placemarks 我们要的反编码信息
    }];
}
```



```

        for (CLPlacemark *placemark in placemarks) {
            NSLog(@"country:%@",placemark.country);
            NSLog(@"name:%@",placemark.name);
            //遍历地址字典
            for (NSString *key in placemark.addressDictionary) {
                NSLog(@"%@",placemark.addressDictionary[key]);
            }
        }
    }
};

}

- (void)locationManager:(CLLocationManager *)manager didFailWithError:(NSError *)error {
    NSLog(@"定位失败");
}

```

(2) 导航：在定位的基础的延伸

(3) 查询：在定位的基础的延伸

(4) 地图：

我把系统的说了，其他的百度和高德，腾讯的自己下（）

和定位很类似

```
#import <MapKit/MapKit.h>
```

协议

MKMapViewDelegate

```
@property (nonatomic, strong) MKMapView *mapView;//地图
```

```
(void)initMapView {
```

```
    [self initManager];//需要定位 就调用
```

```
//实例化 地图
```

```
self.mapView = [[MKMapView alloc] initWithFrame:self.view.bounds];
```

```
self.mapView.mapType = MKMapTypeStandard;
```

```
//设置地图显示的 区域 (给一个中心位置)
```

```
// 给一个 经纬度 和 缩放比例(0.01---0.05)
```

```
//34.77274892, 113.67591140
```

```
self.mapView.region = MKCoordinateRegionMake(CLLocationCoordinate2DMake(34.77274892, 113.67591140),MKCoordinateSpanMake(0.01,
```

```
//是否显示 用户位置
```

```
self.mapView.showsUserLocation = YES;
```

```
//设置代理
```

```
self.mapView.delegate = self;//可以操作点标注
```

```
//粘贴地图
```

```
[self.view addSubview:self.mapView];
```

```
//增加大头针 （需要就创建不需要可以无视）
```

```
[self createAnnotation];
```

```
}
```

协议方法

```
- (MKAnnotationView *)mapView:(MKMapView *)mapView viewForAnnotation:(id<MKAnnotation>)annotation {
```

```
if ([annotation isKindOfClass:[MKPointAnnotation class]]) {
```

```
    //判断是哪一类点标注数据
```

```
    //创建 点标注视图 (采用复用机制)
```

```
    //队列获取空闲的
```

```
    MKPinAnnotationView *pinView = (MKPinAnnotationView *)[mapView dequeueReusableAnnotationViewWithIdentifier:@"MKPinAnnotationView"]
```

```
    if (pinView == nil) {
```

```
        //没有那么创建新的
```

```
        pinView = [[MKPinAnnotationView alloc] initWithAnnotation:annotation reuseIdentifier:@"MKPinAnnotationView"];
```

```
    }
```

```

//设置属性（不需要的可以不设置）
//是否显示气泡
pinView.canShowCallout = YES;
//是否有掉落的动画
pinView.animatesDrop = YES;//(点标注视图子类MKPinAnnotationView才可以设置)
//设置大头针视图的颜色 红 绿 紫三种
pinView.pinColor = MKPinAnnotationColorPurple;

//设置气泡的左右侧附件
UIView *redView = [[UIView alloc] initWithFrame:CGRectMake(0, 0, 30, 30)];
redView.backgroundColor = [UIColor redColor];
pinView.leftCalloutAccessoryView = redView;

UIButton *button = [UIButton buttonWithType:UIButtonTypeInfoLight];
button.frame = CGRectMake(0, 0, 30, 30);
//气泡附件 如果是UIControl的子类 不需要再增加事件，有一个协议的方法可以替代
//[button addTarget:<#(id)#> action:<#(SEL)#> forControlEvents:<#(UIControlEvents)#>];
pinView.rightCalloutAccessoryView = button;
return pinView;//返回对象地址
}
}
}

如果需要的还可以添加手势
- (void)createLongPress {
    UILongPressGestureRecognizer *longPress = [[UILongPressGestureRecognizer alloc] initWithTarget:self action:@selector(longPress)];
    [self.mapView addGestureRecognizer: longPress];
}

根据手势做出不同的调用
- (void)longPress:(UILongPressGestureRecognizer *)press {

}

数据库
sqlite(一般我们不直接向数据库进行操作，使用 第3 方库来操作数据库，方便快捷，但不能认为这些库就能存储数据，简单点理解这玩意就是中介)
常用的有FMDB（开源库）coredata(官方系统库)
常用的数据库操作，增删改查
首先导入第三方库FMDB
封装一个类来管理数据库
方法：
代码操作数据库 用fmdb 第三库操作 sqlite
fmdb 就是 通过对C语言的底层函数封装 对 数据库进行创建 增删改查数据

步骤
1. 导入fmdb
2. 导入libsqlite3.dylib
3. 导入头文件 #import "FMDatabase.h"
4. 创建 数据库
    4.1 打开数据库 创建表
    4.2 增删改查数据 代码执行sql 语句
//导入头文件
#import "FMDatabase.h"
创建数据库
- (void)createDataBase {
//沙盒路径
NSString *docPath = [NSSearchPathForDirectoriesInDomains(NSDocumentDirectory, NSUserDomainMask, YES) lastObject];
//拼接 数据库的路径后面的数据库的名字
NSString *dataPath = [docPath stringByAppendingPathComponent:@"myData.sqlite"];
//实例化一个 fmdb 对象
NSLog(@"%@",dataPath);
_database = [[FMDatabase alloc] initWithPath:dataPath];

//如果打开成功 返回yes
//调用open 的时候 如果数据库不存在那么就会先创建再打开，如果存在直接打开

```

```

if ([_database open]) { //打开数据库
    //打开成功之后创建表
    [self createTable];
}else{
    NSLog(@"open error:%@",[_database lastErrorMessage]); //最近一次错误
}
//数据库 一般 只打开一次 这样可以提高效率
}

创建表格
//其实在数据库里面创建表是一样只不过用库来代创建方便（和用中介找房子一样的）
//user是表名（后面的表的属性）
- (void)createTable {
    NSString *sql = @"CREATE TABLE if not exists user (serial integer Primary Key Autoincrement,num integer,name Varchar(256),mydate datetime
age blob)";
    //执行sql 语句 成功返回yes 失败返回no
    BOOL isSuccess = [_database executeUpdate:sql];
    if (!isSuccess) {
        NSLog(@"create table error:%@",_database.lastErrorMessage);
    }
}

//增加数据
- (void)insertData {
    NSInteger num = arc4random()%69+1;
    NSString *name = [NSString stringWithFormat:@"xiaohong%u",arc4random()%100];
    //时间
    NSDate *date = [NSDate date];

    //图片要存成二进制
    NSData *imageData = UIImagePNGRepresentation([UIImage imageNamed: @"0"]);
    //要用？ 占位符 在sql 中？ 表示的是对象的占位符
    //？ 对应的必须是一个OC的对象的地址

    //增加 sql 语句
    NSString *sql = @"insert into user(num,name,mydate,headimage) values (?, ?, ?, ?)";

    //执行sql
    BOOL isS = [_database executeUpdate:sql, @(num), name, date, imageData];
    if (!isS) {
        NSLog(@"insert into error :%@",_database.lastErrorMessage);
    }
}

//查询
- (void)fetchAllData{
    //1.sql
    NSString *sql = @"select num,name,mydate,headimage from user";
    //2.执行
    //会返回一个结果集合 查找的结果都在 FMResultSet 中
    FMResultSet *rs = [_database executeQuery:sql];
    //遍历集合
    while ([rs next]) { //表示 FMResultSet 中还有没有记录
        NSInteger num = [rs intForColumnIndex:0]; //根据字段索引获取值
        NSInteger num2 = [rs intForColumn:@"num"]; //根据字段名字获取值
        NSLog(@"num:%ld num2:%ld", num, num2);
        NSLog(@"name:%@", [rs stringForColumn:@"name"]);
        NSLog(@"date:%@", [rs dateForColumn:@"mydate"]);
        NSLog(@"image_length:%ld", [rs dataForColumn:@"headimage"].length);
        NSLog(@"-----");
        //[rs objectForKeyForColumnName:<#(NSString *)#>]; //通用
    }
    //第一循环 遍历第0条记录
    //第二循环 1

```

```
//...
//直到最后没有了记录 循环退出

}

//更新数据
-(void)updateData
{
    NSString*sql=@"update user name=? where num<50 ";
    if ([_database executeUpdate:sql:@"小红" ]) {
        NSLog(@"update error:%@",_database.lastErrorMessage);
    }
}

//删除数据
-(void)deletedatawithNUM:(NSInteger)num
{
    NSString*sql=@"delete from user where num=?";
    if ([_database executeUpdate:sql:@(num)]) {
        NSLog(@"%@",_database.lastErrorMessage);
    }
}
}
```

coredata 苹果官方的效果棒棒哒，也很好用

首先右键—> newfile—> (iOS)core data—>DataModel 完成后会有类名.xcdatamodel 文件，可视化操作，创建表（注意改下表名字，默认的有可不上），然后在表里面添加属性，完成后右键—> newfile—> (iOS)core data—>NSManagedObject subclass ,然后和你的表关联一下model层就创建

下面就是封装一个类来专门管理coredata, 方便我们对数据库尽心操作
导入头文件

```
#import <CoreData/CoreData.h>
//设计一个单例类 管理数据库
@interface CoreDataManager : NSObject
//非标准单例
+ (instancetype)defaultManager;

//上下文管理对象
@property (nonatomic,strong) NSManagedObjectContext *context;
```

```
//增删改查
//增加一个数据
- (void)insertDataWithName:(NSString *)name age:(int)age;
```

```
//根据名字删除
- (void)deleteDataWithName:(NSString *)name;
```

```
//修改数据 根据名字修改年龄
- (void)updateDataWithName:(NSString *)name age:(int)age;
```

```
//查询
//查询所有的数据
- (NSArray *)fetchAllData;
```

```
//根据名字查找
- (NSArray *)fetchDataWithName:(NSString *)name;
```

```
实现方法：
//（创建单例类的方法网上有好多）
+ (instancetype)defaultManager {
    static CoreDataManager *manager = nil;
```

```
@synchronized(self) {
    manager = [[self alloc] init];
```

```
}
return manager;
}

//初始化准备工作
//1.导入头文件 CoreData/CoreData.h
//2.创建一个数据模型文件(和数据库中的表类似)，里面创建一些数据模型(设计属性)
//3.设计一个数据模型类(根据数据模型文件)
//术语不明白的度娘走起（我就不唠叨了）
- (instancetype)init {
    if (self = [super init]) {
        //1.将数据模型文件中的模型放入 modelFile 指向的对象中
        //关联数据模型
        NSManagedObjectModel *modelFile = [NSManagedObjectModel mergedModelFromBundles:nil];

        //2.设置存储协调器(协调底层和上层)
        //2.1让协调器和modelFile产生关联
        NSPersistentStoreCoordinator *coordinator = [[NSPersistentStoreCoordinator alloc] initWithManagedObjectModel:modelFile];

        //2.2设置数据库文件的路径
        NSString *path = [NSHomeDirectory() stringByAppendingPathComponent:@"Documents/Mydata.sqlite"];
        NSError *error = nil;
        //2.3设置存储方式 根据路径创建数据库文件
        ///将coreData数据 映射到数据库

        NSPersistentStore *store = [coordinator addPersistentStoreWithType:NSSQLiteStoreType configuration:nil URL:[NSURL fileURLWithPath:path]
s:nil error:&error];
        if (!store) {
            //创建失败
            NSLog(@"creat store failed:%@",error.localizedDescription);
            return nil;
        }

        //3.托管对象/上下文管理对象
        self.context = [[NSManagedObjectContext alloc] init];
        //托管对象和协调器产生关联
        self.context.persistentStoreCoordinator = coordinator;
        //_context 对数据库进行增删改查
    }
    return self;
}

下面就是增删改查了
//增加一个数据
- (void)insertDataWithName:(NSString *)name age:(int)age {
    //1.给_context 操作的数据增加一个UserModel实例对象

    //用 NSEntityDescription 来增加
    UserModel *model = (UserModel *)[NSEntityDescription insertNewObjectForEntityForName:@"UserModel" inManagedObjectContext:self.conte
model.name = name;
model.age = @(age);
model.fName = [name substringToIndex:1];

    //保存数据
    [self saveDataWithType:@"addData"];
}

- (void)saveDataWithType:(NSString *)type {
    NSError *error = nil;
    //回写 保存到数据库文件
    if (![self.context save:&error]) {
        //保存失败
        NSLog(@"%@: %@",type,error.localizedDescription);
    }
}
```

```
}
//根据名字删除
- (void)deleteDataWithName:(NSString *)name {
    //根据名字 找到对象
    NSArray *arr = [self fetchDataWithName:name];
    //遍历数组
    for (UserModel *model in arr) {
        [self.context deleteObject:model];
    }
    //保存数据
    [self saveDataWithType:@"deleteData"];
}

//修改数据 根据名字修改年龄
- (void)updateDataWithName:(NSString *)name age:(int)age{
    //1.根据名字 找到对象
    NSArray *arr = [self fetchDataWithName:name];
    //2.遍历数组
    for (UserModel *model in arr) {
        model.age = @age;
    }
    //3.保存数据
    [self saveDataWithType:@"updateData"];
}

//查询
//查询所有的数据
- (NSArray *)fetchAllData {
    return [self fetchDataWithName:nil];
}

//根据名字 在数据库中 查找 数据模型对象
//根据名字查找
- (NSArray *)fetchDataWithName:(NSString *)name {
    //1.先设置查找请求
    NSFetchRequest *request = [[NSFetchRequest alloc] init];
    //2.设置 查找的数据模型对象
    request.entity = [NSEntityDescription entityForName:@"UserModel" inManagedObjectContext:_context];
    //3.设置 谓词 (根据条件 找要设置谓词)
    if (name) {
        //name 不是nil 那么就根据名字找 设置谓词
        //要查询 一个对象的 匹配的属性 那么需要设置谓词
        NSPredicate *predicate = [NSPredicate predicateWithFormat:@"name like %@",name];
        request.predicate = predicate;
    }
    //还可以设置排序 从小到大 或者从大到小
    //按照年龄降序 的一个描述
    NSSortDescriptor *sort1 = [NSSortDescriptor sortDescriptorWithKey:@"age" ascending:NO];
    //按照 name 进行 升序排列
    NSSortDescriptor *sort2 = [NSSortDescriptor sortDescriptorWithKey:@"name" ascending:YES];
    #if 0
        request.sortDescriptors = @[sort1]; //按照一个准则排序 age
    #else
        //先按照 age 进行降序排， 如果出现age 相同 那么 再按照name 升序排序
        request.sortDescriptors = @[sort1,sort2];
    #endif

    //不设置 谓词 那么找所有

    //5.执行 查询请求 返回一个数组
    NSArray *resultArr = [_context executeFetchRequest:request error:nil];
    return resultArr;
}
```

封装以后直接调就行了

二维码（多关注点github好东西很多的）

第三方库ZBarSDK

导入库

导入系统库

libz.dylib

libiconv.dylib

QuartzCore.framework

CoreVideo.framework

CoreMedia.framework

AVfoundation.framework

原理示例：

二维码编译顺序

Zbar编译

需要添加AVFoundation CoreMedia CoreVideo QuartzCore libiconv

```
ZCZBarController*vc=[[ZCZBarController alloc] initWithBlock:^(NSString *str, BOOL isSucceed) {  
    if (isSucceed) {  
        NSLog(@"扫描后的结果~%@",str);  
    }  
}];  
[self presentViewController:vc animated:YES completion:nil];
```

生成二维码

拖拽libqrencode包进入工程，注意点copy

添加头文件#import "QRCodeGenerator.h"

```
imageView.image=[QRCodeGenerator qrlImageForString:@"这个是什么" imageSize:imageView.bounds.size.width];
```

```
#import "ZCZBarController.h"
```

```
#import "QRCodeGenerator.h"
```

```
UIButton*button=[UIButton buttonWithType:UIButtonTypeSystem];
```

```
button.frame=CGRectMake(0, 70, 100, 100);
```

```
[button addTarget:self action:@selector(btn:) forControlEvents:UIControlEventTouchUpInside];
```

```
[button setTitle:@"扫描二维码" forState:UIControlStateNormal];
```

```
[self.view addSubview:button];
```

```
UIImageView*imageview=[[UIImageView alloc] initWithFrame:CGRectMake(100, 200, 200, 200)];
```

```
imageview.image=[QRCodeGenerator qrlImageForString:@"生成二维码" imageSize:300];
```

```
[self.view addSubview:imageview];
```

```
-(void)btn:(UIButton*)button
```

```
{
```

```
ZCZBarController*vc=[[ZCZBarController alloc] initWithBlock:^(NSString *str, BOOL isSucceed) {  
    if (isSucceed) {  
        NSLog(@"扫描后的结果~%@",str);  
    }  
}];
```

```
};
```

```
[self presentViewController:vc animated:YES completion:nil];
```

```
}
```

推送

本地推送，网络推送，激光推送（要钱呀！屌丝伤不起）

网络推送：

应用场景

提醒业务，比如一些秀场，女主播可以通知他们的土豪（比如我），赶紧来撒钱

每天晚上8点影视剧的推送

小说更新

游戏活动推送等

//在这个方法里面写

```
-(BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
```

```
if ([[UIDevice currentDevice] systemVersion] floatValue)>=8.0) {
```

```
[[UIApplication sharedApplication] registerUserNotificationSettings:[UIUserNotificationSettings settingsForTypes:(UIUserNotificationTypeAlert|  
otificationTypeSound|UIUserNotificationTypeBadge) categories:nil]];
```

```
//开启通知
```

```
[[UIApplication sharedApplication] registerForRemoteNotifications];
```

```

    }

}

//当我们接到通知之后，如何去处理，首先去处理一个标识
-(void)application:(UIApplication *)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken
{
    //我们首先获取一个token值，相当于我们用的QQ，需要一个QQ号码，那么这个QQ是谁，是苹果服务器，我们自己通过自己的设备向苹果服务器发
    请求，告诉他们我们应用的一个标示，作为他们的联系
    //获取token需要进行处理，把这个标示发给我们服务器端做记录，当我们的服务器需要给用户发消息的时候，使用这个标示符+我们要发送的消息
    服务器，拼过会根据这个标示符发到对应的手机的里面
    //因为在有网的情况下，手机是一直和苹果服务器保持者联系，从理论上来说苹果可以控制任何一台手机的情况下进行相关的操作
    //最明显的就是，在有网的情况下，你收不到任何消息，但是在有网的情况下会弹出很多消息
    NSLog(@"%@",deviceToken);
}

//出错处理
-(void)application:(UIApplication *)application didFailToRegisterForRemoteNotificationsWithError:(NSError *)error
{
    NSLog(@"%@",error);
}

-(void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo
{
    //接受推到消息的是一个字典，是规定的格式
}

本地推送：
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {

if ([UIApplication instancesRespondToSelector:@selector(registerUserNotificationSettings:)]) {
    [application registerUserNotificationSettings:[UIUserNotificationSettings settingsForTypes:UIUserNotificationTypeAlert|UIUserNotificationType
    UIUserNotificationTypeSound categories:nil]];
}

/创建一个本地推送
UILocalNotification*local=[[UILocalNotification alloc]init];
//设置推送内容
local.alertBody=@"亲，什么时候约;
//设置声音
local.soundName=@"au_gameover.wav";
//设置推送数目
local.applicationIconBadgeNumber=1000;
local.fireDate=[NSDate dateWithTimeIntervalSinceNow:10];
//把推送任务增加到推送队列中，需要注意，推送通知后，程序就算被杀掉，推送通知任然可以运行
//虚拟机上如果要看效果的话
//如果要弹出推送通知，需要程序退出后台，快捷键Command+Shift+h
[[UIApplication sharedApplication]scheduleLocalNotification:local];
/*
//删除通知
NSArray*localArray=[UIApplication sharedApplication].scheduledLocalNotifications;
//遍历通知
for (UILocalNotification* notification in localArray) {
    if ([notification.alertBody isEqual:@"亲，什么时候约"]) {
        [[UIApplication sharedApplication]cancelLocalNotification:notification];
    }
}
//删除所有的通知
[[ UIApplication sharedApplication]cancelAllLocalNotifications];
/*

本地推送的加入方式，比如判断。3天没来，每次程序启动，把原来的旧通知，并且计算出3天后的时间

*/

// NSInteger num=[UIApplication sharedApplication].applicationIconBadgeNumber;
// num=num+1;

```



```
// local.applicationIconBadgeNumber=num;
*/

}

//激光有Demo
VLC(网上有教程)
VLC集成指南
添加libMobileVLCKit
添加库
libstdc++
libiconv
libbz2
Security
QuartzCore
CoreText
CFnetWork
OpenGLS
AudioToolbox
修改C++编译器为stdC++

聊天 (tcp-udp)

(socket库)
#import "AsyncSocket.h"
协议
AsyncSocketDelegate
//建立发送端
    AsyncSocket * sendSocket;
//建立服务端
    AsyncSocket * severSocket;
//建立一个数组保存连接
@property(nonatomic,strong)NSMutableArray * socketArray;

/*
    建立一个群聊，学生向教师端发送消息，教师端显示所有消息
*/

- (void)CreatSocket
{
    sendSocket=[[AsyncSocket alloc] initWithDelegate:self];
    severSocket=[[AsyncSocket alloc] initWithDelegate:self];
    //服务端绑定端口，监听该端口接收的数据
    /*
        端口最大为65535，其中建议设置为5000以上，另外还有一些特殊的端口，例如8080为视频端口，建议不要占用
    */
    [severSocket acceptOnPort:5678 error:nil];

}

- (void)onSocket:(AsyncSocket *)sock didAcceptNewSocket:(AsyncSocket *)newSocket
{
    //接收的一个新连接，这个连接需要保存一下，然后持续保持连接
    [self.socketArray addObject:newSocket];

    //其中-1标示持续观察，如果设置为300，那么300秒以后就不在观察
    [newSocket readDataWithTimeout:-1 tag:100];

}

//协议方法
- (void)onSocket:(AsyncSocket *)sock didReadData:(NSData *)data withTag:(long)tag
```

```
{
    //接收到的数据
    NSString * message=[[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding];
    if (_textView) {
        //在原来的旧数据上面，追加新的数据
        _textView.text=[NSString stringWithFormat:@"%s%@",_textView.text,message];
    }

    [sock readDataWithTimeout:-1 tag:100];

}

- (void)onSocket:(AsyncSocket *)sock didWriteDataWithTag:(long)tag
{
    //发送成功
}

//textfield 协议
//发送数据
-(BOOL)textFieldShouldReturn:(UITextField *)textField{
    if (textField.text.length>0) {
        // 发送数据
        if (![sendSocket isConnected]) {
            //确定是否连接，如果没有连接，则开始连接host:后面是IP地址
            [sendSocket connectToHost:@"192.168.2.7" onPort:5678 error:nil];
        }
        //当连接完成以后，发送数据
        //拼接数据是谁说，我希望获得当前设备的名称
        // [[UIDevice currentDevice]systemName];该方法只有在真机上才有效，在模拟器上无效
        NSString * message=[NSString stringWithFormat:@"%s说： %s\n",@"房鸢",textField.text];
        [sendSocket writeData:[message dataUsingEncoding:NSUTF8StringEncoding] withTimeout:-1 tag:100];

    }
    return YES;
}
```

屏幕截图

ZCScreenShot库

此类用于屏幕截图

添加库：无

代码示例 为截取全屏

```
[BeginInitImageContext beginImageContext:self.view.frame View:self.view];
```

2个参数 第一个参数用于截取的范围，第二个参数截取哪个view上

//示例代码

```
[ZCScreenShot beginImageContext:self.view.frame View:self.view];
```

//第一个参数是截取图片的范围，第二个参数是截取的那一层

```
#import "ZCScreenShot.h"
```

```
UIImage *image=[ZCScreenShot beginImageContext:self.view.frame View:self.view];
```

（上面两个加起来就是一个小型教学客户端呀）

语音

（科大讯飞）

讯飞官方demo吧，实在是太全了我就不献丑了

关键词: ios 开发 应用 软件开发	
<div><div>回复</div><div>引用</div><div>收藏</div><div>分享</div></div>	
<div><div>ygq_2wang</div><div></div><div><div>级别: 新手上路</div><div>状态: 未签到 - [5天]</div><div>UID: 492541</div><div>精华: 0</div><div>发帖: 54</div><div>可可豆: 76 CB</div><div>威望: 76 点</div><div>在线时间: 126(时)</div><div>注册时间: 2015-07-27</div><div>最后登录: 2016-03-02</div></div></div>	<div>沙发： 发表于: 2015-08-22 18:16 发自: Web Page<div>只看该作</div></div> <div>mark</div> <div><div>回复</div><div>引用</div><div>分享</div></div>
<div><div>迷糊小红发</div><div>拐了个弯，重新出发</div><div></div><div><div>级别: 新手上路</div><div>状态: 未签到 - [35天]</div><div>UID: 493716</div><div>精华: 0</div><div>发帖: 15</div><div>可可豆: 210 CB</div><div>威望: 113 点</div><div>在线时间: 71(时)</div><div>注册时间: 2015-07-31</div><div>最后登录: 2016-03-04</div></div></div>	<div>板凳： 发表于: 2015-08-22 18:25 发自: Web Page<div>只看该作</div></div> <div>楼主好人，好人一生平安</div> <div><div>回复</div><div>引用</div><div>分享</div></div>
<div><div>lebut</div><div>找一群志同道合的人共同学习，挑战各种疑难问题，成为当今时代大牛</div><div></div><div><div>级别: 侠客</div><div>状态: 未签到 - [3天]</div><div>UID: 238297</div></div></div>	<div>3 楼： 发表于: 2015-08-22 22:21 发自: Web Page<div>只看该作</div></div> <div>学习了</div>

<div>精华: 0 发帖: 223 可可豆: 250 CB 威望: 247 点 在线时间: 205(时) 注册时间: 2013-08-19 最后登录: 2016-03-03</div>	<div></div> <div>回复 引用 分享</div>
<div>cskevincs</div> <div></div> <div>级别: 新手上路 UID: 301141 精华: 0 发帖: 23 可可豆: 23 CB 威望: 23 点 在线时间: 217(时) 注册时间: 2014-03-18 最后登录: 2016-03-03</div>	<div>4 楼: 发表于: 2015-08-23 10:29 发自: Web Page 只看该作</div> <div></div> <div>回复 引用 分享</div>
<div>AE86</div> <div></div> <div>级别: 新手上路 状态: 未签到 - [1天] UID: 464038 精华: 0 发帖: 7 可可豆: 14 CB 威望: 8 点 在线时间: 96(时) 注册时间: 2015-04-20 最后登录: 2016-03-01</div>	<div>5 楼: 发表于: 2015-08-23 12:19 发自: Web Page 只看该作</div> <div>很好，mark</div> <div>回复 引用 分享</div>
<div>op航海</div> <div></div> <div>级别: 新手上路 UID: 407220 精华: 0 发帖: 34 可可豆: 178 CB 威望: 178 点 在线时间: 566(时) 注册时间: 2014-11-27 最后登录: 2016-03-03</div>	<div>6 楼: 发表于: 2015-08-23 14:20 发自: Web Page 只看该作</div> <div>mark下.....</div> <div>回复 引用 分享</div>
<div>yangdreaming</div> <div></div> <div>级别: 新手上路  状态: 未签到 - [27天] UID: 63064 精华: 0 发帖: 37 可可豆: 342 CB 威望: 273 点 在线时间: 484(时) 注册时间: 2011-04-11 最后登录: 2016-03-02</div>	<div>7 楼: 发表于: 2015-09-09 17:04 发自: Web Page 只看该作</div> <div></div> <div>回复 引用 分享</div>
<div>lpx1520</div>	<div>8 楼: 发表于: 2015-09-09 21:36 发自: Web Page 只看该作</div> <div> 太有用了,mark</div>

« 1 2 3 4 » Pages: 1/4 Go

✚ CocoaChina » iOS开发 版块浏览记录? 帖

<div><div><div><div><div><div></div><div>Cocoa China</div></div></div><div>苹果开发中文站</div><div><div><div>» 网站地图</div><div>» 联系我们</div><div>» 清除Cookies</div><div>» 手机浏览</div></div></div></div></div></div>	<div>首页</div> <div>游戏开发</div> <div>App Store研究</div> <div>iOS开发</div> <div>游戏开发</div> <div>Cocos引擎</div> <div>业界动态</div> <div>产品设计</div> <div>程序人生</div>	<div>开发者服务</div> <div>论坛</div> <div>技术问答</div> <div>开发者中心</div> <div>代码库</div> <div>工具库</div>	<div>Cocos商店</div> <div>上传大赛</div>	<div>论坛板块</div> <div>OS X开发</div> <div>Swift专区</div> <div>iOS开发</div> <div>App Store经验交流</div> <div>美术区</div> <div>游戏开发区</div> <div>议事大厅</div>	<div>开发者中心</div> <div>App Store应用排行</div> <div>发码</div> <div>签名</div> <div>FAQ</div>	<div>关注CocoaChina</div> <div>关注微信 每日推荐</div> <div></div> <div>扫一扫 浏览移动版</div> <div></div>
--	--	---	------------------------------------	--	--	---

						客服邮箱
©2015 Chukong Technologies,Inc. Cocoa China 苹果开发中文站 Powered by phpwind Certificate Code © 2003-2010 phpwind.com Corporation. 京ICP证 100954号 京公网安备11010502011183 京ICP备 11006519号						