

悟家

版本号V001R001

悟家框架学习文档

**深圳市银河风云网络系统股份有限公司**

**Shenzhen GALAXYWIND Network Systems Co., Ltd.**

地址：深圳市高新技术产业园区北区新西路5号银河风云大厦

Address: GALAXYWIND building, No.5 Xinxi road, Shenzhen High-Tech Industry Park, NanShan district, China

邮编 P.C.: 518055

电话 Tel: +86-755-83400088

传真 FAX: +86-755-86139063

网站 Web: [http://www.galaxywind.com](http://www.galaxywind.com/)

客服 Hotline: 400 000 9879

企业QQ: 4000009879

微信公众号 WeChat:



版权声明

版权所有©深圳市银河风云网络系统股份有限公司2016。深圳市银河风云网络系统股份有限公司（以下简称"银河风云"）对本资料进行版权声明，未经银河风云书面许可，任何单位及个人不得以任何方式或理由对本资料的任何内容进行复制、修改、抄录、传播。

商标声明

"银河风云"，"GALAXYWIND"等是深圳市风云实业有限公司的注册商标，本文档提及的其他商标、服务标志、注册商标及注册服务标志均为其所有者拥有。

目录

**1** 标题1（快捷方式 Ctrl+F1） 2

1.1 标题2（快捷方式F2） 2

1.1.1 标题3（快捷方式F3） 2

# 标题1（快捷方式 Ctrl+F1）

## 标题2（快捷方式F2）

### 标题3（快捷方式F3）

#### 标题4（快捷方式F4）

#### 标题5（快捷方式F5）

（快捷方式F6）

（快捷方式F7）

|  |
| --- |
| **表格标题（15cm宽度）** |
|  |

例子：函数说明示例：

|  |  |  |
| --- | --- | --- |
| **macbee\_app\_version\_set** | | |
| 函数原型 | macbee\_ret\_t  macbee\_app\_version\_set(uint8\_t maj, uint8\_t min, uint8\_t rev) | |
| 函数描述 | 设置macbee设备的版本号 | |
| 参数 | maj | 主版本号 |
| min | 次版本号 |
| rev | 修正版本号 |
| 返回值 | 见表3.1 | |
| 备注 |  | |

正文内容格式（快捷方式F8）

1. 列表1（快捷方式F9）

* 子列表1（快捷方式F11）
* 子列表2
* 列表2（快捷方式F10）
* 子列表1
* 子列表2

警告警告（快捷键CTRL+F9）

警告采用该样式

注意注意（快捷键CTRL+F10）

注意采用该样式

备注备注（快捷键CTRL+F11）

备注信息采用该样式

# 悟家系列-林肯温控器

## LinkonTimerListPage

@interface LinkonTimerListPage : CommonTimerListPage

林肯温控器定时器列表页，继承自定时器列表公共类CommonTimerListPage

#define LINKON\_MAX\_TEMP 350

#define LINKON\_MIN\_TEMP 50

温度使用的是整数而不是浮点数，可以节省空间（定义为short只需要2个字节）

在viewDidLoad中:

[self.view adjustFrameHeightHasStateBar:YES hasNavigationBar:YES hasTabBar:NO];

通过状态栏，导航栏以及底栏，修正Frame

-(NSMutableArray \*)commonTimerListgetDataSource

重写父类方法，返回定时器列表数据源。

？疑问self.devInfo.linkonInfo.comm\_timer\_head.timerArray

类的设计没猜到

-(void)commonTimerListSetImage:(CommonTimerListTableCell \*)cell indexPath:(NSIndexPath \*)indexPath

重写父类方法，为指定的Cell填充数据，图片等。

子类不需要关心Cell的重用与初始化，父类作为TableView的数据源提供Cell时，完成Cell的初始化与重用，并在返回Cell前调用子类的该方法，由子类完成Cell的数据填充。

GxTimer \* timer = self.timerListAry[indexPath.row];

实际上父类存放定时器的容器是timerListAry

[cell setSwitchOnOff:timer.enable];

更新当前Cell的Switch开关状态

[cell setAshingTimerCellFromEnable:timer.enable];

当定时器enable属性为NO时，图片做灰度处理。

[self getTempWithTimer:timer]

格式化返回定时器的温度字符串，其他模式，风速，情景类似。

[self BigTen:timer.hour]

格式化返回时间，猜测主要是前面补0，关于定时器的时间，用的是 hour，minute，duration三个字段来存储的。

[self periodTimerHour:timer.hour mintue:timer.minute dur:timer.duration isNextDayBuffer:&isNextDay]

格式化结束时间，同时计算是否跨天。

定时器的类别直接分为“阶段定时器”、“开机定时器”、“关机定时器”，比我之前的设计可以节省一个字段。

IJStrLocalized(@"关")

多语言话字符串

最终通过

[cell setIconImage:@"定时器\_阶段定时开启" timerBeginString:timerBegin timerEndString:timerEnd isNextDay:isNextDay week:week state:stateStr]

或

[cell setIconImage:@"定时器\_定时开启" timerString:timerBegin week:week state:stateStr]

更新了Cell

-(void)commonTimerPushToEditpageWithTitle:(NSString \*)title

页面跳转，父类收到列表点击或者侧滑点击编辑按钮应该会调用该函数。

？疑问：被编辑的定时器应该是放在self.theparam中传递到定时器编辑页面。

[self pushViewController:editPage animated:YES displayTabBar:NO];

父类统一页面跳转接口。

[[LinkonTimerEditPage alloc]initWithParam:self.theparam withHandle:self.handle];

最上层的自定义父类，统一参数传递。

-(void)commonTimerConfigSWonOff:(GxTimer \*)timer

定时器Switch开关状态改变后回调此函数。

？疑问：cl\_comm\_timer\_t具体结构是什么样的？

cl\_comm\_timer\_t \* air\_timer = [timer makeCommonTimerPktWithStatus:timer.enable timerType:linkon\_timer];

cl\_misc\_comm\_timer\_modify\_add(self.handle, airTimer)

先由GxTimer对象生成一个cl\_comm\_timer\_t，再通过cl\_misc\_comm\_timer\_modify\_add添加定时器修改任务。处理返回的回调由基类父控件定义的self.handle接收。

猜测就会触发：

-(void)eventHandle:(u\_int32\_t)event withUerHandle:(void\*)userHandle withCallbackHandle:(void\*)callbackHandle

本类没有重写该方法，猜测在父类中会判断修改是否成功，则会做出提示或重新刷新列表的操作。

## LinkonTimerEditPage

@interface LinkonTimerEditPage : CommonTimerEditPage

林肯温控器定时器编辑页，继承自定时器编辑公共类CommonTimerEditPage

-(NSArray \*)commonTimerConfigurationDataSource

重写父类方法，需要返回一个二维数组，父类根据该配置数组，生成界面。

子项可以表示为【样式，具体值，对齐方式】

猜测会因为不同的样式使用不同的重用标记。

CommonTimerEditItemTypePlaceholder 占位行，父类很可能并不是使用Header或者Footer而是使用空白行来占位。

CommonTimerEditItemTypeONOFFTime 开关定时器设定时间

CommonTimerEditItemTypeBeginTime 阶段定时器起始时间

CommonTimerEditItemTypeEndTime 阶段定时器结束时间

CommonTimerEditItemTypeWeek 周期选择

CommonTimerEditItemTypeFunction 开关定时器开关设定

CommonTimerEditItemTypeTemp 温度设定

CommonTimerEditItemTypeWind 风速设定

CommonTimerEditItemTypeMode 模式设定

CommonTimerEditItemTypeScene 情景设定

self.timer.linkonTimer.tmp

定时器基类对象timer下有个属性linkonTimer，猜测其他设备的定时器也会采用类似的命名方法，通过懒加载的方式不会造成内存浪费，同时让一个基类对象可以联系上所有的设备定时器。

DevInformation \* devInfo = [APP\_SHARED\_USERMANAGER getDeviceByHandle:self.handle];

获取当前设备

疑问：没猜到实现方法，尤其是参数居然是self.handle。

LinkonThermostatStat \* linkonStat = devInfo.linkonInfo.stat;

设备下有linkonInfo属性，对于其他设备，肯定有别的属性，通过懒加载可以分设备存储不同的属性字段了。

LinkonThermostatStat林肯温控器状态参数类，最初看到这个类的时候以为定时器那边也会复用，但是并没有。

疑问：为什么要将LinkonThermostatStat独立出来呢？

-(void)cellItemSelected:(CommonTimerEditItemType)itemType indexPath:(NSIndexPath \*)indexPath

重写父类方法，处理Cell选中逻辑。

[self userCommonItemChose:itemType indexPath:indexPath];

使用通用的选择器

[self pickerChose:itemType index:indexPath.row];

使用了自定义的选择器，该函数在本类自己实现的。

-(void)commonTimerCellMakeForCustom:(ihomeTableViewCell \*)cell itemType:(CommonTimerEditItemType)itemType indexPath:(NSIndexPath \*)indexPath

自定义Cell样式

[cell setSwitch:isOnTimer];

懒加载方式，直接通过设置Switch的状态达到样式配置。

-(void)commonCellSwitchSelected:(ihomeTableViewCell \*)cell itemType:(CommonTimerEditItemType)itemType indexPath:(NSIndexPath \*)indexPath onOff:(BOOL)onoff

Cell中的Switch开关状态改变后触发此函数。更改数据源，重新加载列表[self tableViewReloadData]。

-(NSString\*)commonTimerCellCustomTitle:(CommonTimerEditItemType)itemType indexPath:(NSIndexPath \*)indexpath

重写父类方法，返回列表标题。

-(void)commonTimeConfigTimerStruct:(cl\_comm\_timer\_t \*)timer\_t compaterTimer:(GxTimer \*)compater

保存定时器，点击保存按钮后，会触发此函数。

timer\_t->extended\_data\_u.linkon\_timer.tmp = temp;

cl\_misc\_comm\_timer\_modify\_add(self.handle, airTimer)

cl\_comm\_timer\_t已经以参数形式传入，无需再生成，将用户设定的值保存到cl\_comm\_timer\_t中，最后调用sdk提交本次修改。

self.timer.tid

定时器的唯一标识tid应该是提交到SDK后，若成功，才会生成的。

-(void)pickerChose:(CommonTimerEditItemType)itemType index:(NSInteger)index

针对Cell的选中逻辑部分自定义选择器。

初始化一个普通的UIPickerView，数据源与回调均为本类

初始一个GXBottomCommonView对象，并通过setPicker函数加入UIPickerView，通过setTitle函数，设置标题，通过show函数展现出来。点击确认按钮回调设置在confirmBlock。

在确认回调里面，修改列表数据源的数据

self.DataSource[index][EditDataTypeDataValue] = [self getSceneByModeIndex:sceneIndex];

最后重新加载列表。

self.hourNumberAry

self.mintureNUmberAry

父类关于小时与分钟的数据源。

[GxAppConfig sharedInstance].isCentigrade

全局配置，是否摄氏度

总结

1. 尽可能的节省空间，字段类型刚好够就好，能用字节空间小的就用小的。
2. 对于页面控制器，尽量在父类完成界面元素的生成，样式控制，在子类中进行数据填充，与业务逻辑处理。

# 项目结构

整个项目包含很多个Target。Target之间相互独立，并没有依赖关系，但是资源文件有穿插共享。

## 文件层次结构

1. common（通用文件）
   1. 必选类
      1. include

必要的头文件，包含很多宏定义，结构体定义，调试控制

* + 1. appHelpers

工具类的扩展，包含网络库，钥匙串，加密，队列，颜色，日期，图片（缓存，处理），字符串，通讯录，压缩等等。

|  |  |
| --- | --- |
| AFNetworking+helper | 对第三方网络库AFNetworking的扩展 |
| ihomeKeychain | 钥匙串的增、删、查 |
| Cipher | 加密与解密 |
| Queue | 自定义的队列 |
| Colours | 颜色扩展类 |
| NSDate-Utilities | 日期类扩展 |
| GxImageCache | 自定义图片缓存 |
| NSString+helper | 字符串扩展类 |
| GxAppHelper | 针对整个项目提供了很多全局函数。回头还需要仔细看看 |
| UIImage+helper | 图片扩展类，包括剪裁，旋转，着色，模糊以及生成渐变扇形图片等等 |
| AddressBook | 通讯录 |
| ZipArchive+helper | 压缩类扩展，提供压缩诊断报告 |

* + 1. configes

|  |  |
| --- | --- |
| GxFileManager | 文件管理器 |
| GxSysInformation | 当前设备软硬件信息，设备类型，尺寸，系统版本 |
| GxAppConfig | 程序配置，诸如语言，反馈，温度单位，提示相关，UUID，Banner相关，自动登录，诊断，缓存等。还有很多没太看明白 |

* + 1. viewHelpers

|  |  |
| --- | --- |
| SWTableViewCell | 第三方列表Cell侧滑库 |
| MMDrawerController | 第三方抽屉侧滑库 |
| ThemeManager | 主题管理类，还提供了3个基础主题界面类，自身监听主题切换通知，更改主题资源或颜色 |
| gmGridView | 第三方表格视图库 |
| EqImageView |  |
| GxBaseBButtonViewController |  |
| GxBaseTabBarController | 基础TabBar页面控制类 |
| GxBaseTableViewController | 已被注释掉了 |
| GxBaseViewController | 贯穿整个项目的基础界面控制类 |
| ihomeTableViewCell | 贯穿整个项目的基础列表Cell类 |
| IMButton |  |
| MBProgressHUD | 第三方HUD窗口类 |
| MLNavigationController | 第三方手势返回导航控制类 |
| TTTAttributedLabel | 第三方富文本库 |
| UIButton+helper | 按钮扩展类 |
| UILabel+helper | 文本扩展类，用于计算文本宽高 |
| UIMenuBar | 第三方目录菜单库 |
| UINumberView | 专门用于显示数字的界面类 |
| UISlider+helper | 滑动条扩展类 |
| UISwitch+helper | 开关扩展类 |
| UITableView+helper | 列表扩展类 |
| UITextField+helper | 文本输入框扩展类，扩展了一个pickerArray属性，想干什么？ |
| UIView+helper | UIView的扩展类，提供界面层次匹配查询，大小位置的便捷查询与设置等 |
| UIAlertView+helper | 警告框扩展类，提供修改文本对齐方式 |
| UIImageView+helper | UIImageView扩展类，主要提供扇形渐变着色 |
| UIActionSheet+helper | UIActionSheet扩展类 |
| GXBottomCommonView | 项目常用的底部弹窗界面类，提供了取消与确认按钮，可以自定义加入选择器等 |
| GxShowPopView | 全屏弹窗界面 |
| GxTableViewCell | 基础Cell类，需要与ihomeTableViewCell进行比较下 |

* + 1. sdkSouces

针对不同的产品有各自的头文件。

* + 1. modules

|  |  |
| --- | --- |
| AFNetworking | 第三方网络库 |
| DXAlertView | 第三方警告框库，目前项目主要就使用这个警告框框架 |
| EnvironmentInfo | 天气数据实例类 |
| MCSoundBoard | 第三方音效播放管理类 |
| Reachability | 第三方网络状态库，有新增自定义函数 |
| GxTimer | 定时器类 |
| AudioRecorder | 录音类 |
| AudioPlayer | 音频播放类 |
| SharedData | 全局共享数据存储类，主要提供事件中转分发功能等 |
| UserManager | 当前登录用户管理类。包含用户信息等 |
| DevInformation | 设备信息类 |
| AreaInformation | 区域信息类。设备按区域分组管理？ |
| SceneInformation | 情景类，就是诸如回家，离家，休息等可编辑的情景 |
| EquipmentInformation | 设备信息类，得好好的与DevInformation联系起来看看 |
| VideoPlayer | 视频播放控制类 |
| AlarmLog | 报警日志管理类 |
| ProbeLanDeviceList | 局域网设备列表 |
| BindPhone | 手机绑定管理类 |
| Appliance |  |
| HttpComm | HTTP请求二次封装 |
| FansBox |  |
| MsgLog | 消息与推送管理 |
| DeviceUpgradeInformation | 设备固件升级管理类 |
| CmtLog |  |
| Health | 健康类，主要管理家庭成员健康信息统计 |
| ComUdpDevData | 设备基础信息类 |
| AirEbInfo | 空调类设备信息 |
| APPDebugInformation | 设备与软件信息管理类 |
| RfDeviceInfo | 射频设备信息类 |

* 1. 可选类

|  |  |
| --- | --- |
| GTMBASE64 | Base64编码 |
| MJExtension | 第三方模型与字典转换库 |
| SDWebImage | 第三方网络图片下载库 |
| SceneLinkage | 情景联动 |
| notifyView | 第三方通知栏 |
| EGOTableViewPullRefresh | 第三方列表下拉刷新 |
| ihomePickerView | 自定义选择控制器 |
| Linechart | 第三方的曲线绘制库 |
| MagicPieLayer | 第三方饼状图绘制库 |
| codeBar | 二维码扫描界面控制类 |
| iCarousel | 第三方轮播图片库 |
| BackgroundRunner | 后台运行管理 |
| RemotePushManager | 设备订阅管理？ |
| BBCyclingLabel | 第三方跑马灯效果的文本控件类 |
| ihomeChartView | 曲线界面类 |

* 1. miscFiles

|  |  |
| --- | --- |
| ZipArchive | 压缩库 |
| ShareSDK | 第三方分享，包括QQ，新浪微博，微信 |
| colorPicker | 调色盘，提供颜色的选择 |
| sound | 声音资源文件 |
| ZBarSDK | 第三方二维码扫码库 |

1. 产品或系列

|  |  |
| --- | --- |
| Skins | 皮肤资源文件。是以文件形式加入工程 |
| InfoPlist.strings | 项目配置文件多语言配置设置 |
| Info.plist | 项目配置文件 |
| Language-name.strings | 多语言翻译 |
| theme.plist | 皮肤配置文件 |
| Images.xcassets | 主要包含AppIcon与LaunchImage |
| viewControllers | 产品自定义界面控制类 |

# 总结-枚举与宏

# 总结-数据类

# 总结-管理类

# 总结-界面类

# 总结-界面控制类

## GxBaseViewController （基础界面控制类）

1. 初始化接口

-(instancetype)initWithParam:(id)param withHandle:(cl\_handle\_t)t\_handle;

参数param一般是一个字典型数据。

参数t\_handle就是当前设备的一个句柄属性。DevInformation.handle

这里有个小疑问，为什么没有直接用设备序列号去定位一个设备？

初始化函数会完成页面通知注册，监听从GX\_NOTIFY\_LANGUAGE\_CHANGE到GX\_NOTIFY\_APP\_WILLTERMINATE之间的通知，最后通过notifyHandleDelegate接收通知回调。管理通知的类为SharedData

1. 导航栏按钮的添加与回调处理

-(void)addBarItemLeftButton:(UIButton\*)btn;左侧添加一个按钮

-(void)addBarItemRightButton:(UIButton\*)btn;右侧添加一个按钮

-(void)addBarItemRightButtons:(NSArray \*)btns;右侧添加一组按钮

-(void)addLeftBackButton;左侧添加返回按钮

-(void)addLeftBackButton:(UIImage \*)image;左侧添加指定图片返回按钮

-(void)addLeftBackButton:(UIImage \*)image himage:(UIImage \*)himage;左侧添加指定图片与高亮返回

-(void)addLeftBackThemeButton;左侧添加主题控制返回按钮

-(void)onBackButtonClicked:(id)sender;点击“返回”

-(void)addRightMoreButton;添加“更多”

- (void)onBtnMoreClicked:(id)sender;点击“更多”

1. 系统通知回调处理

-(void)onLanguageChanged:(NSObject\*)obj;切换语言

-(void)onBackgroundChanged:(NSObject\*)obj;切换背景

-(void)onNetworkStatusChanged:(NSObject\*)obj;切换网络状态

-(void)onThemeChanged:(NSObject\*)obj;切换主题

-(void)onAlarmShow:(NSObject\*)obj;报警通知

-(void)onNotifyCReceived:(GxNotifyType)type withObject:(NSObject\*)obj;分类型获取到通知

1. 分屏幕尺寸加载界面

-(void)viewDidLoadWith35InchScreen;专为3.5寸设备添加子页面

-(void)viewDidLoadWith4InchScreen; 专为4.0寸设备添加子页面

1. HUD显示等待信息

-(void)waitviewShow:(NSString \*)prompt toview:(UIView \*)parentView;

-(void)waitviewShow:(NSString \*)prompt dimBackground:(BOOL)dimBackground toview:(UIView \*)parentView;

-(void)waitviewShow:(NSString \*)prompt;

-(void)waitviewShow:(NSString \*)prompt dimBackground:(BOOL)dimBackground;

-(void)waitviewStop;

-(BOOL)waitviewState;

-(void)waitviewChangeMsg:(NSString \*)msg;

1. 页面导航push操作

-(void)pushViewController:(UIViewController\*)viewController

                 animated:(BOOL)animated

            displayTabBar:(BOOL)displayTabBar;

-(void)pushRightViewController:(UIViewController\*)viewController

                      animated:(BOOL)animated;

1. 状态与连接提示

-(void)setStatusImage:(UIImage \*)image Label:(NSString \*)label;

-(void)setConnectStatusLabel:(NSString \*)text;

-(void)setConnectStatusLabelFrame:(CGRect)frame;

1. 是否支持侧滑返回

-(void)setNavigationPanCtrlEnable:(BOOL)enable;

1. 侧边栏控制

-(GxLeftActionListPage\*)getLeftActionListPage;获取侧边栏界面对象

-(void)ajustLeftActionByPageArray:(NSArray\*)pagenames;配置侧边栏

-(void)ajustLeftPageCmt:(DevInformation \*)devInfo;配置小区

-(void)adjustLeftPageRowBindPhone:(cl\_handle\_t)devHandle;配置隐私设置

-(void)adjustLeftPageRowResetDevice:(cl\_handle\_t)devHandle;配置远程回复出厂设置

-(void)adjustLeftPageRowChangeDeviceWiFi:(cl\_handle\_t)devHandle;配置设备重置无线连接

1. 播放声音与振动

- (void)sound;

- (void)sound:(NSString \*)audiofile;

- (void)sound:(NSString \*)audiofile volume:(float)volume;

- (void)shake;

- (void)feedback:(NSString \*)audio volume:(float)volume;

- (void)feedback:(NSString \*)audio;

1. 设置空数据提示

-(void)setNoDataLabel:(NSString \*)label;

- (void) setNoDataViewVerticalAlignmentCenter;

- (void) setNoDataViewCenter:(CGPoint)center;

1. 设备序列号无法登录时，显示帮助提示界面

- (void)addDeviceHelpViewWithType:(NSInteger)type;

- (void)onBtnKnowClicked:(id)sender;

1. 操作提示界面控制

- (void)tipViewInit:(CGFloat)heightTip;

- (void)tipViewPresented:(UIView \*)addView;

- (void)tipViewDismiss;

1. 底部选择界面控制，实际使用时需要在皮肤下parameter.plist中设置。

- (void)showBottomOptionsView;

- (void)hiddenBottomOptionsView;

通过重写- (void)onBottomOptionClick:(UIView \*)view 函数获取点击事件。

1. 提示网络不可用

- (BOOL)networkNoneButtonClick:(id)sender;

当点击操作需要网络时，先通过该函数检测一下网络。

## GxLeftActionListPage （侧边栏基类）

## CommonTimerListPage （通用定时器列表）

## CommonTimerEditPage （通用定时器编辑与新建）

# 总结-程序流程

## 程序主入口

/smartHome/main.m（以iHome-808为例）

1. /smartHome /AppDelegate
   1. 注册分享
   2. 设置默认皮肤
   3. 监听网络状态
   4. 判断是否由系统通知中心打开App，获取通知信息。
   5. 初始化rootView（分情况）
   6. 加载启动页（启动页会自动消失）
   7. 注册申请APNS权限。

自此进入rootView

1. rootView先是IntroductionPage，介绍完毕后重新设置为MMDrawerController。
2. MMDrawerController结构大致如下：

MMDrawerController : {

center : (GxBaseTabBarController)[

0 : MLNavigationController -> PhoneUserDeviceListPage

1 : MLNavigationController -> DeviceGroupListPage

2 : MLNavigationController -> SceneLinkageHomepage

3 : MLNavigationController -> RFHistoryListPage

]

left : SLleftPage

}

## 从点击首页设备列表到林肯温控器控制页面

1. 若设备未完成添加，则弹添加设备界面。
2. 排除SlaveDeviceInformation 类型设备。
3. 调用cl\_user\_set\_click 判断设备是否可控。
4. 判断设备是否正在升级，登录失败信息，绑定错误信息，是否登录成功，设备是否在线
5. 设置当前设备是否可编辑。
6. 准备页面跳转参数。
7. 调用cl\_user\_scan\_db\_rf 进行双向扫描
8. 判断设备版本是否过低。
9. 区分设备类型，加载对应的设备控制界面。
10. 林肯温控器控制界面结构如下：

MMDrawerController : {

center : (GxBaseTabBarController)[

0 : MLNavigationController -> I8CtrlPage

1 : MLNavigationController -> SmartControlViewController

]

right: iHome808LeftActionListPage

}

# 总结-释义

|  |  |  |  |
| --- | --- | --- | --- |
| 简写 | 全拼 | 举例 | 释义 |
| cmt | community | [devInfo.cmtLog getCmtsNum] | 小区，社区 |
| cl | client | cl\_lede\_ctrl\_stat(cl\_handle\_t dev\_handle,cl\_lede\_led\_state\_t\* uconfig) | 客户端，多用于SDK网络访问 |
|  | slave | SlaveDeviceInformation |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

# 总结-SDK

|  |  |  |
| --- | --- | --- |
| API | 使用时机 | 备注 |
| cl\_user\_set\_click(cl\_handle\_t dev\_handle, u\_int8\_t \*stat) | 点击某个设备的时候，调用这个接口 | 如果返回非RS\_OK，不能进入设备控制界面 |
| cl\_user\_scan\_db\_rf(cl\_handle\_t user\_handle) | 每次登录都需要扫描一次双向设备 | 有返回值，但是没关心 |
|  |  |  |