蓝色互动iOS 代码规范文档

版本1.1

上海科匠信息科技有限公司

iOS开发组编制

2014年4月16日

|  |  |  |  |
| --- | --- | --- | --- |
| 版本 | 修订内容 | 作者 | 时间 |
| 1.0 | 初稿 | 朱亮亮 | 2014-4-16 |
| 1.1 | 图片的位置做了修改 | 孟欢、严登峰、王宇 | 2015-4-20 |

# 目的

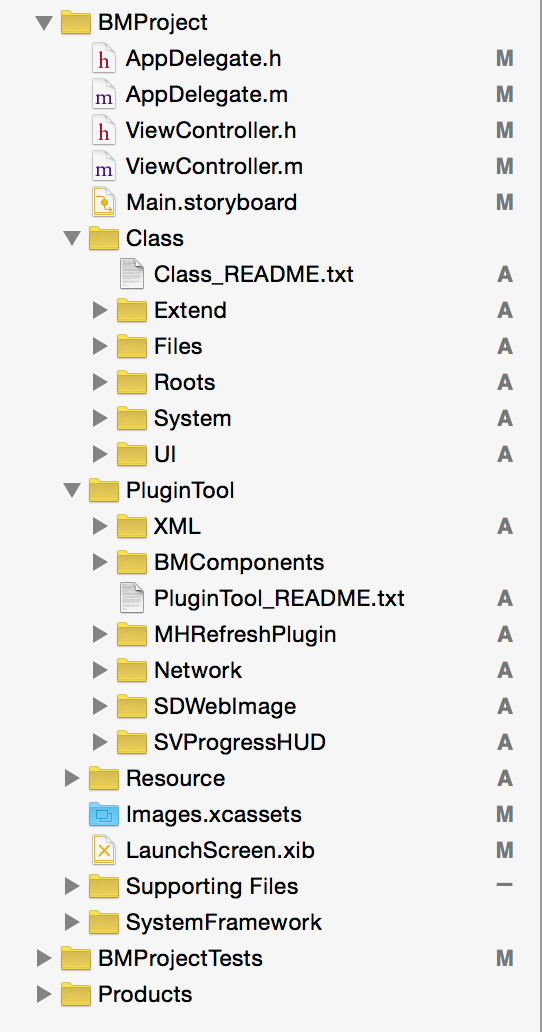
统一编程风格，提高的可读性与编码效率，避免团队开发可能带来的混乱。

# 适用范围

本规范适用于公司项目产品运用Objective-C作为开发语言的编码活动。

# 工程模板介绍

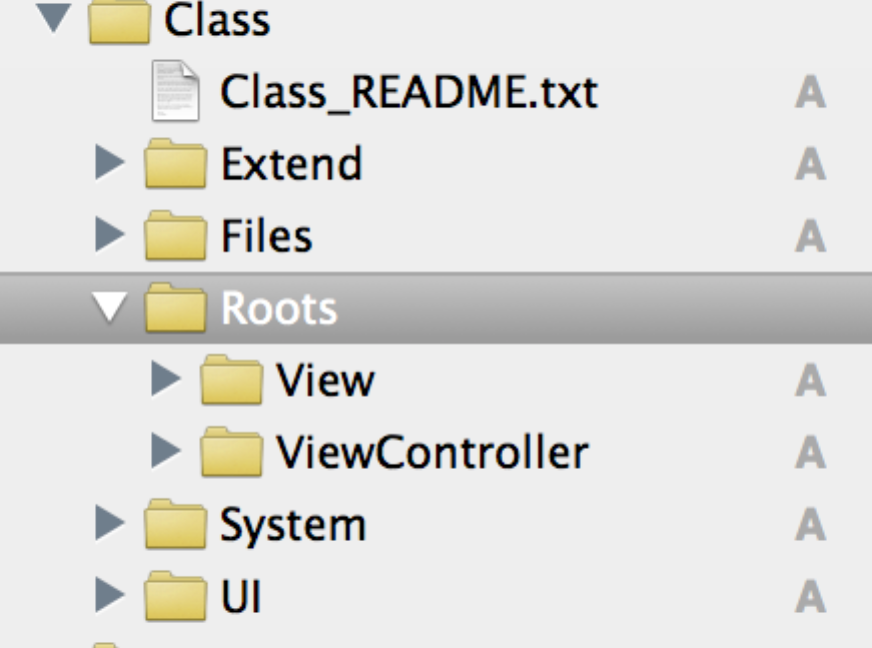
工程目录总览，所有项目都将copy模板工程，无需重新创建和配置新工程。



### 3.1 Roots

作用：只存放模块界面的根ViewController。

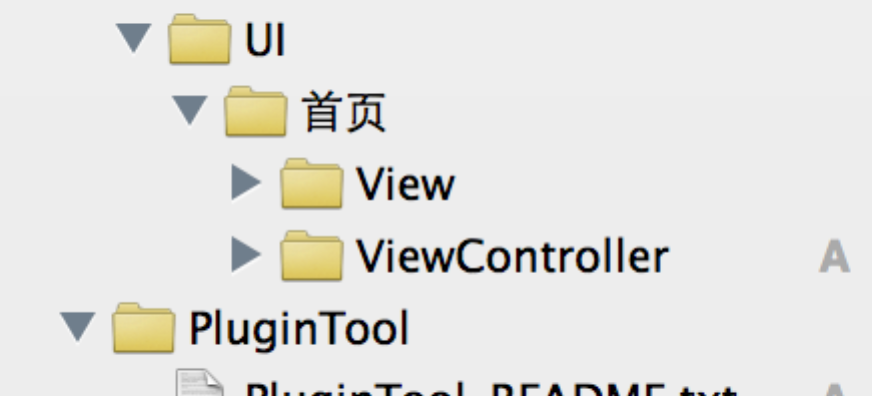
例如：项目中有个首页模块，则把首页模块的根视图控制器HomeViewController放到Roots中



### 3.2 UI

作用：存放所有新增UI界面。新的模块统一放在 UI 目录下,新建一个文件夹,然后在模块文件夹下面在新建 View 和 ViewController 两个目录,分别存放自定义的View和 ViewController。

例如：项目中有个首页模块，则在UI目录下，先创建首页目录，再在首页文件夹下创建View 和 ViewController 两个目录。首页模块中所有自定义的View全部放在此View目录中，首页模块下所有自定义的ViewController全部放在此ViewController目录中。

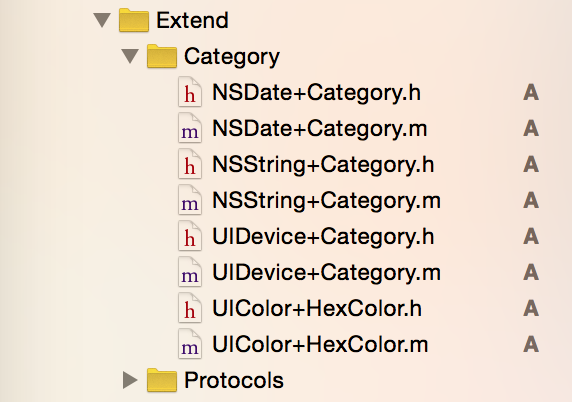


### 3.3 Extend

作用：存放所有自定义的Category和各种Util类。

例如：项目中需要自己写一个NSString+EncodingUTF8Additions，

则放在Extend中



##### **3.3.1 Protocols**

作用：存放单独创建的Protocol文件。（不准在此目录下新建任何目录和除Protocol以外的文件）

### 3.4 System

作用：存放所有系统级目录和类文件。（不准在此目录下新建任何目录和文件）

##### **3.4.1 Base**

作用：存放所有系统级父类文件 。（不准在此目录下新建任何目录和文件）

例如：BMBaseView、

BMBaseViewController、

BMBaseTableViewCell

##### **3.4.2 DB**

作用：存放所有数据库操作类。所有对数据库的操作必须新建或存放到相应的Models类和Caches类中。其中Models类放到Models目录中，Caches类放到Caches目录中。

例如：注册、登录操作完成后要将数据保存到数据库中。先在Models目录下新建User类初始化相应属性，再在Caches目录下新建UserCache类进行实际的数据库操作。

##### **3.4.3 Network**

作用：存放网络请求访问类。（不准在此目录下新建任何目录和文件）

### 3.5 Resource

作用：存放所有资源文件。（不准在此目录下新建任何目录和文件）

##### **3.5.1 Images**

作用：存放除app的Icon和开机画面以外的所有图片文件。

##### **3.5.2 Media**

作用：缩放所有多媒体文件，如各种音频、视频文件。（不准在此目录下新建任何目录和除音频、视频以外的其他文件）

##### **3.5.3 Fonts**

作用：存放系统字体外的其它字体文件。（不准在此目录下新建任何目录和除字体文件以外的其他文件）

### 3.6 PluginTool

作用：存放所有第三方库、类文件和组件目录。

# 代码规范

### 4.1 命名

##### 4.1.1 目录命名

规定： 用英文命名时，首字母大写，可以用中文命名，但是禁止使用拼音命名。

##### 4.1.2 类命名

规定： 必须采用驼峰命名法，单词首字母大写。尽量不要使用带前缀的类名，每个类都有相同前缀不能提高可读性。只有在写共用类时才 推荐使用前缀，使用的前缀必须是BM。禁止使用中文、拼音等方式来命名。

例如：BMBaseView、BMDefine等。

##### 4.1.3 类分类命名（Category）

规定： 必须采用格式：类名+分类名，即Xcode自动生成的命名格式。禁止使用中文、拼音等方式来命名。

例如：NSString+EncodingUTF8Additions。

##### 4.1.4 方法命名

4.1.4.1 一般方法命名

规定：必须使用小写开头的驼峰法命名，每个参数都应该小写开头。禁止使用中文、拼音等方式来命名。方法与方法之间必须空行。

例如：

- (BOOL)application:(UIApplication \*)application didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions;

4.1.4.2 按钮点击事件方法命名

规定：必须使用小写开头的驼峰法命名，以Click结尾 。禁止使用中文、拼音等方式来命名。方法与方法之间必须空行。

例如：

- (void)commitClick;

4.1.4.3 通知方法命名

规定：必须使用小写开头的驼峰法命名，以Notification结尾 。禁止使用中文、拼音等方式来命名。方法与方法之间必须空行。

例如：

- (void)showHUDNotification:(NSNotification \*)notification;

4.1.4.4接口方法命名

规定：必须使用api开头的驼峰法命名。禁止使用中文、拼音等方式来命名。方法与方法之间必须空行。

例如：

+ (void)apiDeleteAllCartsByMemberId:(NSString \*)memberId

password:(NSString \*)password

cartId:(NSString \*)cartId

successBlock:(void(^)(id returnData))successBlock

failureBlock:(void(^)(NSError \*error))failureBlock

showHUD:(BOOL)showHUD;

##### 4.1.5 成员变量命名

规定：尽量采用@property方式来申明变量，并且必须使用小写开头的驼峰法命名 。如申明过@property的变量，不需要在申明同名带前缀下划线变量。如果不用@property方式申明变量，则必须采用前缀下划线来申明变量。禁止使用中文、拼音等方式来命名！

例如：@property (nonatomic, strong) UITableView \*tableView; 或者UITableView \*\_tableView;但不要同时使用。

##### 4.1.6 局部变量命名

规定：必须使用小写开头的驼峰法命名 。 禁止使用中文、拼音等方式来命名！

例如：int cateParentID = 0;

##### 4.1.7 全局变量

规定：必须以k开头，使用驼峰法命名 。 禁止使用中文、拼音等方式来命名！

例如：static NSString \*kGlobalKey = @"A2HK353KH";

##### 4.1.8宏定义常量命名

规定：必须以DEF\_开头并且全部大写或单词首字母大写，单词之间可以采用下划线链接。如果是宏定义方法，方法内的参数名采用小写。 禁止使用中文、拼音等方式来命名！

例如：

#define DEF\_STRING\_SIZE(string, font, width) [string sizeWithFont:font constrainedToSize:CGSizeMake(width, MAXFLOAT) lineBreakMode:UILineBreakModeWordWrap]

### 4.2注释

##### 4.2.1 类注释

规定：所有类的.h 中，类声明的上面添加注释，说明类的基本作用。（格式必须如下，以便生成注释索引）

例如：

/\*\*

\* @brief 授权视图控制器

\*/

@interface AGAuthViewController : UIViewController <UITableViewDataSource, UITableViewDelegate>

{

}

@end

##### 4.2.2 方法注释

规定：所有类的.h 中，方法申明前添加注释，说明方法、参数、返回值的作用。@Breif后面添加方法描述。@Param后面添加参数的名称和描述，每个参数创建一个@Param。@return后面添加返回值的描述。（格式必须如下，以便生成注释索引）

例如：

/\*\*

\* @brief 设置授权凭证

\*

\* @param credential 授权凭证

\* @param type 平台类型

\*

\* @return 返回状态

\*/

+ (int)setCredential:(id<ISSPlatformCredential>)credential type:(ShareType)type;

##### 4.2.3 成员变量注释

4.2.3.1 地址型成员变量

规定：所有地址型成员变量声明前，必须采用///的方式添加注释。 （格式必须如下，以便生成注释索引）

例如：

@interface CloudSearchDemoViewController : UIViewController {

/// 百度地图搜索类

BMKCloudSearch\* \_search;

}

4.2.3.2 @property型成员变量

规定：所有@property型成员变量声明前，必须采用以下方式添加注释。 （格式必须如下，以便生成注释索引）

例如：

/\*\*

\* @brief tableView 主表单

\*/

@property (nonatomic, strong) UITableView \*tableView;

##### 4.2.4 枚举注释

规定：枚举类型声明前，必须采用以下方式添加注释。枚举中的每个值，采用/\*\*< 注释 \*/ 的方式添加注释。（格式必须如下，以便生成注释索引）

例如：

/\*\*

\* @brief 响应状态

\*/

typedef enum

{

SSResponseStateBegan = 0, /\*\*< 开始 \*/

SSResponseStateSuccess = 1, /\*\*< 成功 \*/

SSResponseStateFail = 2, /\*\*< 失败 \*/

SSResponseStateCancel = 3 /\*\*< 取消 \*/

}

SSResponseState;

##### 4.2.5 宏定义注释

4.2.5.1 宏定义常量注释

规定：所有宏定义常量声明前，必须采用以下方式添加注释 （格式必须如下，以便生成注释索引）

例如：

/\*\*

\* @brief 购买物品通知

\*/

#define DEF\_AWARD\_NOTIF\_BUY\_ITEM @"buy\_item"

4.2.5.1 宏定义方法注释

规定：所有宏定义方法声明前，必须采用以下方式添加注释 （格式必须如下，以便生成注释索引）

例如：

/\*\*

\* @brief 获取视图宽度

\*

\* @param view 视图对象

\*

\* @return 宽度

\*/

#define DEF\_WIDTH(view) view.bounds.size.width

##### 4.2.6 逻辑注释

规定：代码里的重要逻辑必须以// 的形式添加注释

例如：

if (type == 1)

{

// 一些操作

}

else

{

// 另一些操作

}

##### 4.2.7 Protocol注释

规定：所有宏定义方法声明前，必须采用以下方式添加注释 （格式必须如下，以便生成注释索引）

例如：

/\*\*

\* @brief 分享内容协议

\*/

@protocol ISSContent <NSObject>

### 4.3 规定

##### 4.3.1 dealloc方法

规定：一个类的dealloc方法如果有必要存在，必须是.m中的第一个方法，方便查找。

##### 4.3.2 #pragma mark –

4.3.2.1 #pragma mark – 点击事件

规定：所有按钮点击事件必须写在“#pragma mark – 点击事件”的下面

4.3.2.2 #pragma mark – 通知事件

规定：所有通知事件必须写在“#pragma mark – 通知事件”的下面

4.3.2.3 所有Delegate的实现

规定：所 有delegate方法的实现必须写在相应的“#pragma mark – XXXDelegate”的下面。自定义Delegate的实现也是如此。

例如：

#pragma mark - UIAlertViewDelegate

- (void)alertView:(UIAlertView \*)alertView clickedButtonAtIndex:(NSInteger)buttonIndex

{

}

### 4.4 编码风格

##### 4.4.1 必须

4.4.1.1所有.h和.m文件中，方法与方法之间必须空行

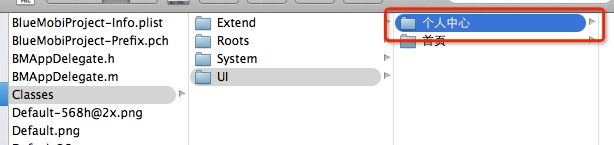
4.4.1.2 所有.m文件中变量的实现之间必须空行

4.4.1.3 所有新建的TableViewCell必须继承BMBaseTableViewCell

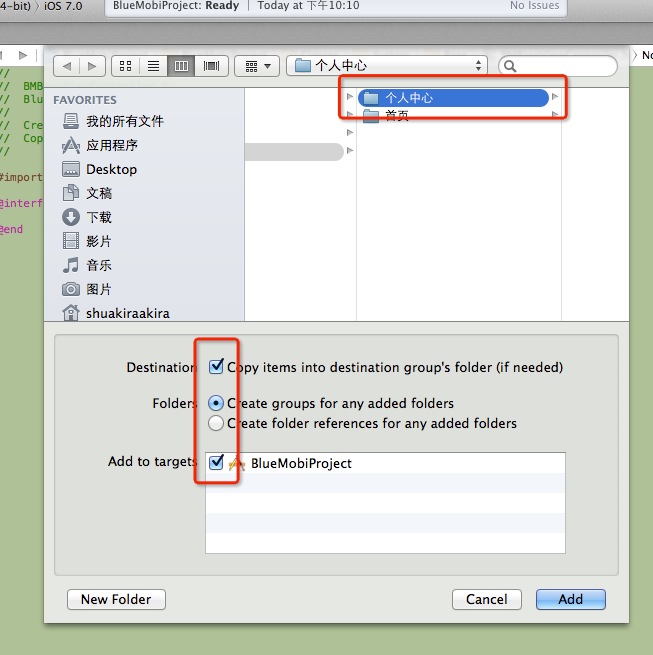
4.4.1.4 所有对TableViewCell的数据初始化，必须调用 - (void)initWithData:(id)data方法中。不准在table的- (UITableViewCell\*)tableView: cellForRowAtIndexPath:方法中直接操作TableViewCell的视图属性

4.4.1.5 所有有翻页要求的ViewController，必须继承BMBaseViewController。尤其是有上下拉刷新的，必须继承此父类

4.4.1.6 所有新建的目录，在工程下必须存在实体目录，方法是在工程目录下创建了实体目录后在将引用加入工程，例如：



步骤1



步骤2

4.4.1.7 所有的网络请求，理论上讲都应该放在viewDidAppear方法之后或之内。除非特殊原因，禁止在loadView、viewDidLoad、viewWillAppear方法中进行网络请求。

4.4.1.8 所有第三方包一律放在ThirdParty目录下，并且要先建立相应实体目录。如果添加了.a或.framework等库，必须将Search Paths下的路径改为$(SRCROOT)的相对路径。

4.4.1.9 除非特殊原因，20行代码以内必须至少有1个空行，增强代码的可读性。

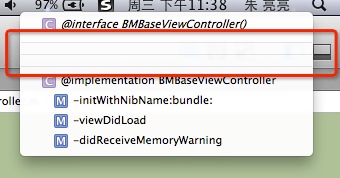
4.4.1.10 重要逻辑前必须有注释，格式参见4.2.6的说明

4.4.1.11 如果一个类文件里有多个类，或者包含了Protocol，那么类与类之间，类与Protocol之间，Protocol与Protocol之间统统要有分隔，并且Protocol一律在文件上方，类在下方。无论在.h还是.m中都是如此，请复制以下两行分隔语句，注意“—”的前和后都要有一个空格

#pragma mark -

#pragma mark –

实际效果



4.4.1.12 所有的类、Protocol、方法、成员变量、静态变量、常量，枚举都必须有注释

4.4.1.13 所有的视图控件命名规范

UIView的命名，必须以View结尾

UIButton的命名，必须以Btn结尾

UILabel的命名，必须以Lb结尾

UITextField的命名，必须以TF结尾

UITextView的命名，必须以TV结尾

UITableView的命名，必须以Table结尾

##### 4.4.2 建议

4.4.2.1 尽量采用@property方式申明成员变量，如果采用了@property方式则无需再申明相应的指针变量，且无需在.m文件中申明@synthesize，如果非要申明@synthesize，请采用@synthesize xxx = \_xxx的方式，不要采用别的方式。调用时除了getter和setter方法内，尽量用例如self.xxx的方式调用变量，少用\_xxx的方式

4.4.2.2 if语句

规定：尽量采用以下格式，if与左（之意见空一格，左右括号{}单独一行。即使if语句里只有一行代码，也请用{}括起来。

例如：

if (type == 1)

{

// 一些操作

}

else

{

// 另一些操作

}

4.4.2.3 for语句

规定：尽量采用以下格式，for与左（之意见空一格，左右括号{}单独一行。

例如：

for (int = 1; i < 2; i ++)

{

// 一些操作

}

4.4.2.4 switch语句

规定：尽量采用以下格式，switch与左（之意见空一格，左右括号{}单独一行。Case 的左右括号{}单独一行。

例如：

switch (type)

{

case 1:

{

// 一些操作

}

break;

default:

break;

}