**com.seeyon.oa.M1IPad**

**网络数据交互：**

**以登录为例：**

1、参数封装类，

文件目录为：/src/M1Core/Biz/Login/Paramete

类的继承关系：SyLoginBizParam🡪SyBaseBizParam->SyBaseObject

SyLoginBizParam：封装网络请求参数。

SyBaseBizParam:没有任何通用的参数被封装，可做扩展。

2.网络请求，

类的继承关系：SyLoginBiz->SyBaseBiz

文件目录为：/src/M1Core/Biz/Login/

syLoginBiz: 封装网络请求，

**a、请求方法request**：

//移除原来登录所保留的cookie消息，以便登录成功后使用新的cookie

// remove NSHTTPCookie

NSArray \*array = [[NSHTTPCookieStorage sharedHTTPCookieStorage] cookies];

for (NSHTTPCookie \*aCookie in array) {

[[NSHTTPCookieStorage sharedHTTPCookieStorage] deleteCookie:aCookie];

}

// remove NSHTTPCookie end

//--------------------end----------------------

//------------**登录时所需要的请求参数-**-----------------

SyLoginBizParam \*aParam = (SyLoginBizParam \*)self.param;

self.delegate = aParam.delegate;

这里的self.param,在SyloginBiz被实例化时，就已经被传值，代码如下：

\_loginBiz = [SyBizManager instanceLoginBiz:loginBaseBizParam];

+ (id)instanceObjWithKey:(NSString \*)aKey parameter:(id)aParameter

{

NSString \*aServerType = [SyServerManager defaultManager].serverType;

NSString \*aServerVersion = [SyServerManager defaultManager].serverVersion;

Class aClass = [self objectWithKey:aKey

serverType:aServerType

serverVersion:aServerVersion

objType:kObjectType\_Class];

id obj = [[aClass alloc] init];

SEL aSel = NSSelectorFromString(@"setParam:");(此处已经将参数请求参数初始化)

if ([obj respondsToSelector:aSel]) {

[obj performSelector:aSel withObject:aParameter];

}

return obj; //在此处不做自动释放，使用后直接release

}

//-------------------end-------------------------

//-----------------**参数转换,转换为syLoginRequest所封装的参数-**------------------

MLoginParameter \*param = [[MLoginParameter alloc] init];

param.username = aParam.userName;

param.password = aParam.password;

param.loginType = aParam.loginType;

param.token = aParam.deviceToken;

//-------------------end---------------------

//------------------**初始化request**-----------------

SyLoginRequest \*aRequest = [[SyLoginRequest alloc] initWithLoginParameter:param];

类的继承关系：SyLoginRequest->SyBaseRequest

syLoginRequest：- (id)initWithLoginParameter:(MLoginParameter \*)loginParameter方法中

对managerName 、managerMethod进行自定义，后台的数据请求接口，每个接口这两个属性都不一样。

同时，在请求数据成功后，会将数据转换成SyLoginResponse，代码如下：

文件路径为：/mobile\_ios-trunk/src/M1Core/Common/ProviderSupport/ SyNormalRequestProvider.m

NSString \*responseStr = [NSStringFromClass([request class]) replaceCharacter:@"Request" withString:@"Response"];

response = [[NSClassFromString(responseStr) alloc] init];

[response setContentValue:valueContent];

//---------------------end-----------------------

// -----------------请求数据----------------------

类的继承关系：SyNormalRequestProvider-> SyBaseProvider

\_provider = [[SyNormalRequestProvider alloc] initWithDelegate:self];

\_provider.useCookiePersistence = YES; //是否储存cookie值，并使用

[\_provider request:aRequest];

syNormalRequestProvide中：- (void)request:(SyBaseRequest \*)aRequest url:(NSString \*)aUrl

自定义请求所需参数，url,是否使用cookie

数据请求成功之后，最终会使用delegate到SyLoginBiz中的 ：

* (void)providerDidFinishLoad:(SyBaseProvider \*)aProvider

再进行返回数据的具体操作。

然后再回调SyLoginBiz的代理SyBaseBizDelegate

- (void)bizDidFinishLoad:(SyBaseBiz \*)aBiz

//--------------------end------------------------

**本地数据交互：**

数据操作是一个单例：SyDBConnection.m,数据库的创建，表的创建，数据查询接口都在这儿

//存储用户信息

1、- (void)setDBUserInformation

{

SyDBConnection \*dbConnection = [SyDBConnection connection];

//-----------------将要储存的数据用对象表示出来--------------------

SyDBUserInformation \*dbUserInformation = [[SyDBUserInformation alloc] init];

dbUserInformation.userName = [SyCore sharedSyCore].sySetting.loginName;

dbUserInformation.userPassword = [SyCore sharedSyCore].sySetting.loginPassword;

dbUserInformation.serverID = [SyCore sharedSyCore].sySetting.serverIdentifier;

dbUserInformation.owerID = [NSString stringWithLongLong:[SyCore sharedSyCore].sySetting.currentUser.orgID ];

//-------------------------------end------------------------------

//-----------------------------插入消息-------------------------

BOOL insert = [dbConnection insertOfflineUserInformation:dbUserInformation];

if (insert) {

NSLog(@"insertSyUserInformation sucess");

}

SY\_RELEASE\_SAFELY(dbUserInformation);

[dbConnection closeConnection];

//--------------------------------end------------------------------------

}

1、2 插入数据

// 插入消息记录

- (BOOL)insertOfflineUserInformation:(SyDBUserInformation \*)aUserInformation

这里通过SQLITE语句将数据插入到数据库，具体方式参照示例代码。

其他查询，删除，更新，也操作示例代码即可。

1.3 配置文件数据

[SyCore sharedSyCore].sySetting.loginName;

**页面切换：**

**一、首页菜单**

1、登录成功后，调用 M1IPhoneAppDelegate.m：

- (void)showMainView

{

if ([self sharedSyCore].isOutLineLogin) {

[self showMainViewInner];

return;

}

//-----------获取用户拥有哪些功能模块---------------

\_getMemberResourcesByIdBiz = [self requestMemberResourcesWithDelegate:self];

//--------------------end------------------------

}

2、SyAppDelegate.m

- (SyBaseBiz \*)requestMemberResourcesWithDelegate:(id<SyBizDelegate>)aDelegate

{

SyGetMemberResourcesByIdBizParam \*aParam = [[SyGetMemberResourcesByIdBizParam alloc] init];

aParam.delegate = aDelegate;

//---------------根据用户orid来去得相应的服务模块-----------

aParam.memberID = self.currentUser.orgID;

aParam.accountID = self.currentUser.account.orgID;

[\_getMemberResourcesByIdBiz cancel];

[\_getMemberResourcesByIdBiz release];

//---------------------------end------------------------

//------------------实例化功能模块网络请求类-----------------

\_getMemberResourcesByIdBiz = [SyBizManager allocAndInitSyGetMemberResourcesByIdBiz:aParam];

[aParam release];

[\_getMemberResourcesByIdBiz request];

//--------------------------end-----------------------

//---------------返回网络请求实例，方便M1IPhoneAppDelegate.m中的 - (void)bizDidFinishLoad:(SyBaseBiz \*)aBiz，判断加载相应的页面。------------------

return \_getMemberResourcesByIdBiz;

//----------------end------------------------

}

2.1、getMemberResourcesByIdBiz为SyGetMemberResourcesByIdBiz类的实例，在SyGetMemberResourcesByIdBiz中的：

- (void)providerDidFinishLoad:(SyBaseProvider \*)aProvider

{

SyGetMemberResourcesByIdResponse \*aResponse = (SyGetMemberResourcesByIdResponse \*)aProvider.response;

MList \*aList = aResponse.mList;

[\_getMemberResourcesBO release];

//-----------------------储存用户所拥有的功能，模块信息，在menuView和其他子功能模块中进行使用------

\_getMemberResourcesBO = [[SyGetMemberResourcesBO alloc] initWithResources:aList.value];

//----------------------------保存到全局，程序运行时一直存在-----------------

self.sharedSyCore.memberResourcesBO = \_getMemberResourcesBO;

//----------------------------------end---------------------------------

[super providerDidFinishLoad:aProvider];

}

3、M1IPhoneAppDelegate.m中

- (void)bizDidFinishLoad:(SyBaseBiz \*)aBiz if (aBiz == \_getMemberResourcesByIdBiz) {

[self showMainViewInner];

[self showWelcomeView];

}

3.1 M1IPhoneAppDelegate.m中：

//---------加载登录后的功能模块菜单页面---------

- (void)showMainViewInner｛

。 。 。

else {

if (!\_menuViewController) {

\_menuViewController = [[SyHomeMenuViewController alloc] init];

\_menuViewController.view.frame = CGRectMake(mainScreenSize.width, statusBarHeight, mainScreenSize.width, \

mainScreenSize.height - statusBarHeight);

}

aToViewController = \_menuViewController;

}

[self.window addSubview:aToViewController.view];

。 。 。

｝

//--------------------------end-----------------------------------

3.1.1 类的继承：

SyHomeMenuViewController—>SyPhoneBannerViewController->SyBannerViewController—>SyBaseViewController

其中SyBaseViewController.m中，自动加载相应子类的view,命名方式为SyHomeMenuView，即去掉“Controller”之后的类名,代码如下：

- (void)loadView

{

[super loadView];

. . .

CGRect frame = [self mainFrame];

if (!\_mainView) {

NSString \*aClassName = NSStringFromClass([self class]);

aClassName = [aClassName replaceCharacter:kControllerSuffix withString:@""];

Class aClass = NSClassFromString(aClassName);

SyBaseView \*aView = [(SyBaseView \*)[aClass alloc] initWithFrame:frame];

if ([aView respondsToSelector:@selector(setViewController:)]) {

[aView setViewController:self];

}

self.mainView = aView;

. . . .

4、 加载模块，在SyMenuView.m中的

所有功能模块保存在Setting.plist"中;

- (void)setup

{

…..

}

该方法对，服务器传回来的功能木块权限信息进行判断，删除的功能模块进行判断，，新增的功能模块等，然后把需要的加入到homeMenu中。

二、功能模块页面切换

1. SyMenuView.m中

- (void)touchUpInsideAction:(SyMenuItem \*)button

{

. . .

//点击Menu菜单后，发送代理到M1IPhoneAppDelegate.m,进行页面切换

// 其中button.target保存要切换的页面类名，然后根据类名实例化

1. if ([\_delegate respondsToSelector:@selector(didSelectedMenuItem:)]) {

[(NSObject \*)\_delegate performSelector:@selector(didSelectedMenuItem:) withObject:button afterDelay:0.13];

}

//---------------------------------end --------------------------

. . .

}

* 1. 、M1IPhoneAppDelegate.m中

- (void)didSelectedMenuItem:(SyMenuItem \*)aMenuItem

{

// aMenuItem.cmpInvokeAddr = @"<http://www.baidu.com>";

//--------------------创建切换视图------------------

[self changeContentViewControllerWithTarget:aMenuItem.target userInfo:[NSDictionary dictionaryWithObjectsAndKeys:aMenuItem.cmpInvokeAddr, @"appPath", aMenuItem.cmpAppName, @"appName", aMenuItem.cmpInfoID, @"infoID", nil] tag:aMenuItem.menuTag];

//--------------------end-------------------

}

1.2 创建和切换视图

- (void)changeContentViewControllerWithTarget:(NSString \*)aTarget userInfo:(NSDictionary \*)aUserInfo tag:(NSString \*)aTag

{

//----------------每一个子功能模块，都有一个navigationcontroller,---------

//----------------先判断是否已经创建过该Navi,没有创建则进行创建-------------

//----------------由于切换到其他功能模块时，并没有移除Navi的子视图，所以重新进入之前选中的功能模块时会保持之前的状态------------

UINavigationController \*navController = [\_menuDictionary objectForKey:aTag];

if (!navController) {

Class class = NSClassFromString(aTarget);

UIViewController \*viewController = [[class alloc] init] ;

if ([viewController isKindOfClass:[SyWebAppFrameViewController class]]) {

SyWebAppFrameViewController \*aVController = (SyWebAppFrameViewController \*)viewController;

aVController.appPath = aAppPath;

aVController.appName = aAppName;

}

if (viewController) {

UINavigationController \*aNavController = [[UINavigationController alloc] initWithRootViewController:viewController];

[\_menuDictionary setObject:aNavController forKey:aTag];

navController = aNavController;

[viewController release];

[aNavController release];

}

}

// ------------------------------end--------------------------

if (navController) {

if (!\_contentViewController) \_contentViewController = \_menuViewController;

if (\_contentViewController != navController) {

//---重新将window的rootViewController设为一个naviCongroller---

self.window.rootViewController = navController;

//---------将原来首页主菜单页面移除-----------------

[\_contentViewController.view removeFromSuperview];

SyMenuView \*aMenuView = [self menuView];

[aMenuView removeFromSuperview];

UIViewController \*topViewController = [self topViewController:navController];//[navController topViewController];

//实现自定义NaviBar的Controller，点击Title可以下拉出现MenuView----------

if ([topViewController respondsToSelector:@selector(updateToOldBannerState)]) {

SyBannerViewController \*aBannerViewController = (SyBannerViewController \*)topViewController;

//将menuView放在当前子页面的下层

[aBannerViewController.view insertSubview:aMenuView belowSubview:aBannerViewController.bannerNavigationBar];

}

else {

[topViewController.view addSubview:aMenuView];

}

}

}

}

2.一般的页面切换

//参数为SyCollaborationDetailViewControllerParam--🡪 SyBaseParam

//所有的参数类需继承与SyBaseParam这样，继承之后自己扩展参数

SyCollaborationDetailViewControllerParam \*aParam =

[[SyCollaborationDetailViewControllerParam alloc] init];

aParam.collListItem = aItem;

aParam.refColSummaryId = @"-1";

aParam.docResId = @"-1";

aParam.baseObjectId = @"-1";

aParam.from = [SyPojoTransformUtils fromWithAffairState:aItem.affairState];

//------------------------------end----------------------------

SyBaseViewController \*viewController = [SyViewControllerManager instanceCollaborationDetailViewController:aParam];

[aParam release];

[self.navigationController pushViewController:viewController animated:YES];

[viewController release];