前言:
XMPPFramework简介

XMPPFramework是一个OS X/iOS平台的开源项目，使用Objective-C实现了XMPP协议（RFC-3920），同时还提供了用于读写XML的工具，大大简化了基于XMPP的通信应用的开发。

- XMPPStream：xmpp基础服务类
- XMPPRoster：好友列表类
- XMPPRosterCoreDataStorage：好友列表（用户账号）
- XMPPvCardCoreDataStorage：好友名片（昵称，签名，性别，年龄等信息）
- xmppvCardAvatarModule：好友头像
- XMPPReconnect：如果失去连接,自动重连
- XMPPRoom：提供多用户聊天支持

pod 'XMPPFramework'
pod ‘CocoaAsyncSocket’

一.登陆和注册

```
#ifndef XMPPSample_XMPPConfig_h
#define XMPPSample_XMPPConfig_h

//openfire服务器IP地址
#define  kHostName
@"127.0.0.1"
//openfire服务器端口 默认5222
#define  kHostPort      5222
//openfire域名
```

```
#define kDomin @"127.0.0.1"
//resource
#define kResource @"iOS"
#endif



  #import "XMPPFramework.h"

@interface XMPPManager :
NSObject<XMPPStreamDelegate>

// 通信通道对象
@property (nonatomic, strong)
XMPPStream *xmppStream;
//单例
+ (XMPPManager *)sharedManager;
//登陆
- (void)loginWithUserName:
(NSString *)userName
                    password:
(NSString *)password;
//注册
- (void)registerWithUserName:
```

```
(NSString *)userName
                        password:
(NSString *)password;
@end


// 枚举
//连接服务器类型
typedef NS_ENUM(NSInteger,
ConnectToServerPurpose)
{

ConnectToServerPurposeLogin, //
登陆

ConnectToServerPurposeRegister
//注册
};

@interface XMPPManager ()

@property (nonatomic, copy)
NSString *password;//密码
@property (nonatomic, assign)
```

```
ConnectToServerPurpose
connectToServerPurpose;

@end

@implementation XMPPManager

/**
 * 创建单例
 */
+ (XMPPManager *)sharedManager
{
    static XMPPManager *manager
= nil;
    static dispatch_once_t
onceToken;
    dispatch_once(&onceToken,
^{
        manager = [[XMPPManager
alloc] init];
    });
    return manager;
}

/**
```

```
 * 初始化方法
 */
- (instancetype)init
{
    if (self = [super init]) {
        // 创建通信通道对象
        self.xmppStream =
[[XMPPStream alloc] init];
        // 设置服务器IP地址

self.xmppStream.hostName =
kHostName;
        // 设置服务器端口

self.xmppStream.hostPort =
kHostPort;
        // 添加代理
        [self.xmppStream
addDelegate:self
delegateQueue:dispatch_get_main
_queue()];

    }
    return self;
```

```
}


/**
 * 登陆方法
 */
- (void)loginWithUserName:
(NSString *)userName password:
(NSString *)password
{
    self.connectToServerPurpose
= ConnectToServerPurposeLogin;
    self.password = password;
    // 连接服务器
    [self
connectToServerWithUserName:use
rName];
}

/**
 * 注册方法
 */
- (void)registerWithUserName:
(NSString *)userName password:
```

```
(NSString *)password
{
    self.connectToServerPurpose
=
ConnectToServerPurposeRegister;
    self.password = password;
    [self
connectToServerWithUserName:use
rName];
}

/**
 * 连接服务器
 */
-
(void)connectToServerWithUserNa
me:(NSString *)userName
{
    // 创建XMPPJID对象
    XMPPJID *jid = [XMPPJID
jidWithUser:userName
domain:kDomin
resource:kResource];
    // 设置通信通道对象的JID
```

```
    self.xmppStream.myJID =
jid;

    // 发送请求
    if ([self.xmppStream
isConnected] ||
[self.xmppStream isConnecting])
{
        // 先发送下线状态
        XMPPPresence *presence
= [XMPPPresence
presenceWithType:@"unavailable"
];
        [self.xmppStream
sendElement:presence];

        // 断开连接
        [self.xmppStream
disconnect];
    }

    // 向服务器发送请求
    NSError *error = nil;
```

```
    [self.xmppStream
connectWithTimeout:-1
error:&error];

    if (error != nil) {
        NSLog(@"%s__%d__%@| 连
接失败", __FUNCTION__, __LINE__,
[error localizedDescription]);
    }
}

/**
 * 连接超时方法
 */
-
(void)xmppStreamConnectDidTimeo
ut:(XMPPStream *)sender
{
    NSLog(@"%s__%d__| 连接服务器
超时", __FUNCTION__, __LINE__);
}

/**
 * 连接成功
```

```
 */
- (void)xmppStreamDidConnect:
(XMPPStream *)sender
{
    switch
(self.connectToServerPurpose) {
        case
ConnectToServerPurposeLogin:
            [self.xmppStream
authenticateWithPassword:self.p
assword error:nil];//认证
            break;
        case
ConnectToServerPurposeRegister:
            [self.xmppStream
registerWithPassword:self.passw
ord error:nil];//注册

        default:
            break;
    }



}
```

viewController

```
- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup
after loading the view,
typically from a nib.
    // 添加代理
    [[XMPPManager
sharedManager].xmppStream
addDelegate:self
delegateQueue:dispatch_get_main
_queue()];
}

/**
 * 登陆按钮方法
 */

- (void) login
{
    NSString
*strUserName=@"gaosng";
    NSString *strPwd=@"123456";
    [[XMPPManager
```

```
sharedManager]
loginWithUserName:strUserName
password:strPwd];

}

/**
 * 验证成功
 */
-
(void)xmppStreamDidAuthenticate
:(XMPPStream *)sender
{
    NSLog(@"%s__%d__| 登陆成功",
__FUNCTION__, __LINE__);
    XMPPPresence *presence =
[XMPPPresence
presenceWithType:@"available"];
    [[XMPPManager
sharedManager].xmppStream
sendElement:presence];
}

/**
```

```
 * 登陆失败
 */
- (void)xmppStream:(XMPPStream
*)sender didNotAuthenticate:
(DDXMLElement *)error
{
    NSLog(@"%s__%d__|",
__FUNCTION__, __LINE__);
}

- (void)didReceiveMemoryWarning
{
    [super
didReceiveMemoryWarning];
    // Dispose of any resources
that can be recreated.
}

RegisterViewController

- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup
after loading the view.
```

```
    [[XMPPManager
sharedManager].xmppStream
addDelegate:self
delegateQueue:dispatch_get_main
_queue()];
}
-(IBAction) registerBtn:
(id)sender{
    [self registerAccount];
}
-(void) registerAccount{
    NSString
*strUserName=@"test001";
    NSString *strPwd=@"123456";
    [[XMPPManager
sharedManager]
registerWithUserName:strUserNam
e password:strPwd];
}

- (void)xmppStreamDidRegister:
(XMPPStream *)sender
{
    NSLog(@"%s__%d__|",
__FUNCTION__, __LINE__);
```

```
    [self.navigationController
popViewControllerAnimated:YES];
}

- (void)xmppStream:(XMPPStream
*)sender didNotRegister:
(DDXMLElement *)error
{
    NSLog(@"%s__%d__| 注册失败",
__FUNCTION__, __LINE__);
}
```

二.获取联系人(花名册)

```
#import "XMPPManager.h"
#import "XMPPConfig.h"
@interface RosterViewController
()<XMPPRosterDelegate>
@property (nonatomic, strong)
NSMutableArray *dataArray;
@end

@implementation
RosterViewController
```

```
- (void)viewDidLoad {
    [super viewDidLoad];
    // Do any additional setup
after loading the view.
    //显示标题为当前登陆的用户
    self.title = [XMPPManager
sharedManager].xmppStream.myJID
.user;

    self.dataArray =
[NSMutableArray array];
    //请求获取花名册
    [[XMPPManager
sharedManager].xmppRoster
addDelegate:self
delegateQueue:dispatch_get_main
_queue()];

}

/**
 * 检索到好友
 */
- (void)xmppRoster:(XMPPRoster
```

```
*)sender didRecieveRosterItem:
(DDXMLElement *)item
{
    // 取到JID字符串
    NSString *jidStr = [[item
attributeForName:@"jid"]
stringValue];
    // 创建JID对像
    XMPPJID *jid = [XMPPJID
jidWithString:jidStr];
    // 把jid添加到数组中
    if ([self.dataArray
containsObject:jid]) {
        return;
    }
    [self.dataArray
addObject:jid];

    NSIndexPath *indexPath =
[NSIndexPath
indexPathForRow:self.dataArray.
count - 1 inSection:0];
//    [self.tableView
insertRowsAtIndexPaths:@[indexP
```

```
ath]
withRowAnimation:UITableViewRow
AnimationAutomatic];
}

/**
 * 检索好友结束
 */
-
(void)xmppRosterDidEndPopulatin
g:(XMPPRoster *)sender
{
    NSLog(@"%s__%d__| 检索好友结
束", __FUNCTION__, __LINE__);
}
```

三.添加好友

在xmppManager.h中

```
#import "XMPPRoster.h"
#import
"XMPPRosterCoreDataStorage.h"
@interface XMPPManager :
NSObject<XMPPStreamDelegate,XMP
```

```
PRosterDelegate>
// 好友花名册管理对象
@property (nonatomic, strong)
XMPPRoster *xmppRoster;
@property (nonatomic, strong)
XMPPJID *fromJID;
//1.好友请求添加我为联系人时
- (void)xmppRoster:(XMPPRoster
*)sender
didReceivePresenceSubscriptionR
equest:(XMPPPPresence *)presence
{
    self.fromJID =
presence.from;
    UIAlertView *alertView =
[[UIAlertView alloc]
initWithTitle:@"好友请求"
message:presence.from.user
delegate:self
cancelButtonTitle:@"拒绝"
otherButtonTitles:@"同意",
nil];
    [alertView show];
}
```

```
- (void)alertView:(UIAlertView
*)alertView
clickedButtonAtIndex:
(NSInteger)buttonIndex
{
    switch (buttonIndex) {
        case 0:
            // 拒绝添加此好友
            [self.xmppRoster
rejectPresenceSubscriptionReque
stFrom:self.fromJID];
            break;
        case 1:
            // 同意添加此好友
            [self.xmppRoster
acceptPresenceSubscriptionReque
stFrom:self.fromJID
andAddToRoster:YES];

        default:
            break;
    }
}
```

```
//2.添加好友时
- (IBAction)addFriendAction:
(UIBarButtonItem *)sender
{
    UIAlertView *alertView =
[[UIAlertView alloc]
initWithTitle:@"提示"
message:@"添加好友"
delegate:self
cancelButtonTitle:@"取消"
otherButtonTitles:@"确定",
nil];
    alertView.alertViewStyle =
UIAlertViewStylePlainTextInput;
    [alertView show];
}

- (void)alertView:(UIAlertView
*)alertView
clickedButtonAtIndex:
(NSInteger)buttonIndex
{
```

```
    if (buttonIndex) {
        UITextField *textField
= [alertView textFieldAtIndex:
0];
        //添加好友
        XMPPJID *jid = [XMPPJID
jidWithUser:textField.text
domain:kDomin
resource:kResource];
        [[XMPPManager
sharedManager].xmppRoster
addUser:jid withNickname:nil];
    }
}
```

三.收发消息时

XMPPManager.h文件

```
#import
"XMPPMessageArchiving.h"
#import
"XMPPMessageArchivingCoreDataSt
orage.h"
XMPPRosterDelegate
```

```
// 信息归档对象
@property (nonatomic, strong)
XMPPMessageArchiving
*xmppMessageArchiving;
// 创建一个数据管理器
@property (nonatomic, strong)
NSManagedObjectContext
*context;

@interface ChatViewController :
ViewController

@property (nonatomic, strong)
XMPPJID *friendJID;

@end

#import "XMPPManager.h"

@interface ChatViewController
()<XMPPStreamDelegate,
UIAlertViewDelegate>
@property (nonatomic, strong)
```

```
NSMutableArray *messageArray;
@end

@implementation
ChatViewController

- (void)viewDidLoad
{
    [super viewDidLoad];

    // 初始化数组
    self.messageArray =
[NSMutableArray array];

    // 给通信通道对象添加代理
    [[XMPPManager
sharedManager].xmppStream
addDelegate:self
delegateQueue:dispatch_get_main
_queue()];

    // 检索信息
    [self reloadMessages];
```

```
}

- (void)reloadMessages
{
    NSManagedObjectContext
*context = [XMPPManager
sharedManager].context;

    // 创建查询类
    NSFetchRequest
*fetchRequest =
[[NSFetchRequest alloc] init];

    // 创建实体描述类
    NSEntityDescription
*entityDescription =
[NSEntityDescription
entityForName:@"XMPPMessageArch
iving_Message_CoreDataObject"
inManagedObjectContext:context]
;

    [fetchRequest
setEntity:entityDescription];
```

```
    // 创建谓词
    NSPredicate *predicate =
[NSPredicate
predicateWithFormat:@"bareJidSt
r == %@ and streamBareJidStr ==
%@", self.friendJID.bare,
[XMPPManager
sharedManager].xmppStream.myJID
.bare];

    // 创建排序类
    NSSortDescriptor
*sortDescriptor =
[NSSortDescriptor
sortDescriptorWithKey:@"timesta
mp" ascending:YES];
    [fetchRequest
setPredicate:predicate];
    [fetchRequest
setSortDescriptors:@[sortDescri
ptor]];
```

```
    // 从临时数据库中查找聊天信息
    NSArray *fetchArray =
[context
executeFetchRequest:fetchReques
t error:nil];

    if (fetchArray.count != 0)
{

        if
(self.messageArray.count != 0)
{
            [self.messageArray
removeAllObjects];
        }

        [self.messageArray
addObjectsFromArray:fetchArray]
;

        //[self.tableView
reloadData];


        if
```

```
(self.messageArray.count != 0)
{
                // 动画效果
                NSIndexPath
*indexPath = [NSIndexPath
indexPathForRow:self.messageArr
ay.count - 1 inSection:0];
              // [self.tableView
scrollToRowAtIndexPath:indexPat
h
atScrollPosition:UITableViewScr
ollPositionBottom
animated:YES];
            }
    }

}

/**
 * 消息发送成功的方法
 */
- (void)xmppStream:(XMPPStream
*)sender didSendMessage:
(XMPPMessage *)message
{
```

```
    [self reloadMessages];
}

/**
 * 消息接收成功
 */
- (void)xmppStream:(XMPPStream
*)sender didReceiveMessage:
(XMPPMessage *)message
{
    [self reloadMessages];
}

- (IBAction)sendAction:
(UIBarButtonItem *)sender
{
    UIAlertView *alertView =
[[UIAlertView alloc]
initWithTitle:@"发送消息"
message:@"" delegate:self
cancelButtonTitle:@"取消"
otherButtonTitles:@"发送",
nil];
    alertView.alertViewStyle =
```

```
UIAlertViewStylePlainTextInput;
    [alertView show];
}

- (void)alertView:(UIAlertView
*)alertView
clickedButtonAtIndex:
(NSInteger)buttonIndex
{
    if (buttonIndex) {
        UITextField *textField
= [alertView textFieldAtIndex:
0];
        XMPPMessage *message =
[XMPPMessage
messageWithType:@"chat"
to:self.friendJID];
        [message
addBody:textField.text];
        [[XMPPManager
sharedManager].xmppStream
sendElement:message];
    }
}
```