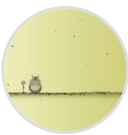


# iOS开发之AsyncSocket的使用



作者 Wy\_chris (/u/16c6a4be7e38) + 关注

2017.02.10 16:02 字数 652 阅读 181 评论 4 喜欢 11

(/u/16c6a4be7e38)

这两天看了一下公司以前的项目，之前自己都没有写过，这次也来学习一下，简单的实现一下在iOS客户端上通过 TCP / IP 来与服务器通信。先来点概念性的东西吧：

<https://git.oschina.net/wyChirs/AsyncSocket>

(<https://git.oschina.net/wyChirs/AsyncSocket>)

因为使用模拟器和手机进行通信的，所以不太好截图，要看效果的话还是下载Demo好一点。千言万语也不及一个Demo来的实在～

## 1、什么是Socket?

Socket的英文原义是“孔”或“插座”。作为BSD UNIX的进程通信机制。通常也称作“套接字”，用于描述IP地址和端口，是一个通信链的句柄。

Socket是面向C/S（客户端/服务端）模型而设计的，针对客户端和服务端提供不同的Socket。客户端有客户端的Socket，服务端有服务端的Socket；  
Socket的常用类型有：流式Socket（基于TCP协议）和 数据报式Socket（基于UDP协议）

## 2、TCP 和 UDP

TCP（Transmission Control Protocol，传输控制协议）是基于连接的协议，正式收发数据前必须和对方建立可靠的连接（“三次握手”），传输速度相对UDP较慢。  
UDP（User Data Protocol，用户数据报协议）是与TCP相对应的协议。它是面向非连接的协议，UDP适用于一次只传送少量数据、对可靠性要求不高的应用环境，但是传输速度快

### 通信流程及要点

- 1.通过 ip/端口 建立连接
- 2.连接成功开启定时发送心跳包（包内容由服务器与客户端沟通）
- 3.向服务端发送数据
- 4.接收来自服务端的数据并做业务逻辑处理
- 5.掉线重连及后台发送心跳包
- 6.关闭长连接

### 代码实现

封装了一个单例类SocketManager，负责主要的通信业务，连接、读写等。  
先贴上.h文件的代码

```

#import <Foundation/Foundation.h>
#import "AsyncSocket.h"

@protocol SocketRequestDelegate <NSObject>

/**
    获取服务器返回的数据（用于显示或逻辑处理）

    @param resultData 此处数据类型需要与后台进行沟通在进行业务逻辑
    */
-(void)socketRequestData:(NSString *)resultData;

@end

enum{
    SocketOfflineByLost,// 掉线, 默认为0
    SocketOfflineByUser,// 用户主动断开
};

@interface SocketManager : NSObject <AsyncSocketDelegate>

/**
    socket
    */
@property(nonatomic,strong)AsyncSocket *socket;

/**
    socket Host
    */
@property(nonatomic,copy )NSString *socketHost;

/**
    socket port
    */
@property(nonatomic,assign)UInt16 socketPort;

/**
    计时器
    */
@property(nonatomic,retain)NSTimer *connectTimer;

/**
    读取服务器数据
    */
@property(nonatomic,strong)id<SocketRequestDelegate> delegate;

+(SocketManager *)shareInstance;

/**
    socket连接
    */
-(void)socketConnectHost;

/**
    socket断开
    */
-(void)cutOffSocket;

/**
    心跳连接
    */
-(void)longConnectSocket;

-(void)startTimer;

/**
    向服务器提交数据

    @param data 此处数据类型需要与后台进行对接 一般是字典
    */
-(void)socketWriteData:(NSString *)data;

```

### 1.通过 ip/端口 建立连接

```
//连接
-(void)socketConnectHost{
    [self cutOffSocket]; //连接之前先手动切断连接，避免崩溃
    //连接ip 端口
    if (![self.socket connectToHost:self.socketHost onPort:self.socketPort error:nil]){
        NSLog(@"连接失败");
    };
    [self.socket readDataWithTimeout:-1 tag:100];
}
```

2.连接成功开启定时发送心跳包

连接成功60s后创建定时器并启动

```
//开始计时器 设定60s想服务器发送一次数据（数据内容与后台人员沟通）
-(void)startTimer{
    if (self.connectTimer!=nil) { //避免内存泄漏 再次开启连接
        [self.connectTimer invalidate]; //停止定时器
        self.connectTimer = nil;
    }
    self.connectTimer = [NSTimer scheduledTimerWithTimeInterval:60 target:self selector:@selector(longConnectSocket) userInfo:nil repeats:YES];
    [self.connectTimer fire]; //执行

    [[NSRunLoop currentRunLoop] addTimer:self.connectTimer forMode:NSRunLoopCommonModes];
    [[NSRunLoop currentRunLoop] run];
}

#pragma mark - 连接成功回调
-(void)onSocket:(AsyncSocket *)sock didConnectToHost:(NSString *)host port:(UInt16)port
{
    NSLog(@"连接成功");
    // [self.socket readDataWithTimeout:-1 tag:1];
    //60s后 开启定时器，并每隔60s向服务器发送一次心跳包
    [self performSelector:@selector(startTimer) withObject:self afterDelay:60];
}
```

```
//心跳包 -长连接
-(void)longConnectSocket{
    //定时向服务器发送数据，此处代码项目自行调整，这里我将当前时间传送给服务器
    [self.socket writeData:[self currentDateStr] dataUsingEncoding:NSUTF8StringEncoding] withTimeout:-1 tag:100];
}
```

3.向服务端发送数据

```
//提交数据到服务器 给外部提供接口直接写入数据
-(void)socketWriteData:(NSString *)data{
    [self.socket writeData:[data dataUsingEncoding:NSUTF8StringEncoding] withTimeout:-1 tag:100];
}
//文本编辑完成就发送给服务器
-(void)textFieldDidEndEditing:(UITextField *)textField{
    if (textField.text.length > 0) {
        //将文本信息 传给服务器
        [[SocketManager sharedInstance] socketWriteData:textField.text];
        textField.text = @"";
    }
}
```

4.接收来自服务端的数据并做业务逻辑处理

在单例中增加协议 ,用于读取服务区返回来的数据

@protocol SocketRequestDelegate <NSObject>

```
/**
    获取服务器返回的数据（用于显示或逻辑处理）
    @param resultData 此处数据类型需要与后台进行沟通在进行业务逻辑
    */
-(void)socketRequestData:(NSString *)resultData;
```

```
#pragma mark - 读取服务器数据
-(void)onSocket:(AsyncSocket *)sock didReadData:(NSData *)data withTag:(long)tag
{
    NSLog(@"收到服务端的数据");
    // 对得到的data值进行解析与转换即可 -->再通过代理传递到当前控制器
    if ([self.delegate respondsToSelector:@selector(socketRequestData:)]){
        //显示服务器发送来的消息
        [self.delegate socketRequestData:[NSString alloc] initWithData:data encoding:NSUTF8StringEncoding]];
    }
    //继续监听
    [self.socket readDataWithTimeout:-1 tag:100];
}
```

在当前控制器遵循协议方法，获取到服务器数据

```
/**
    代理回调

    @param resultData 接收服务器传来的消息
*/
-(void)socketRequestData:(id)resultData{
    self.stateLabel.text = resultData;//显示服务器发过来的数据
}
```

## 5.掉线重连及后台发送心跳包

在.h文件中。增加离线原因 根据不同情况做不同的处理

```
enum{
    SocketOfflineByLost,// 掉线，默认为0
    SocketOfflineByUser,// 用户主动断开
};
```

```
#pragma mark - 断开连接的回调
-(void)onSocketDidDisconnect:(AsyncSocket *)sock
{
    if (self.socket.userData == SocketOfflineByLost) {
        //掉线 就要重连
        [self socketConnectHost];
        NSLog(@"掉线重连");
    }else{
        //主动断开不进行重连
    }
}
```

当程序进入后台以后，如果需保持长连接状态 需要在appdelegate.m中实现唤醒app操作

```
- (void)applicationDidEnterBackground:(UIApplication *)application {
    // Use this method to release shared resources, save user data, invalidate ti
    mers, and store enough application state information to restore your applica
    tion to its current state in case it is terminated later.
    // If your application supports background execution, this method is called i
    nstead of applicationWillTerminate: when the user quits.
    //定期唤醒APP 防止长连接断开
    BOOL backgroundAccepted = [[UIApplication sharedApplication] setKeepAliveTime
    out:600 handler:^(//600s后发送一次心跳包
        [[SocketManager sharedInstance] longConnectSocket];
    )];
    if (backgroundAccepted)
    {
        NSLog(@"backgrounding accepted");
    }
}
```

## 6.关闭长连接

```
//断开
-(void)cutOffSocket{
    self.socket.userData = SocketOfflineByUser; //主动断开
    [self.socket disconnect];
}
```

在控制器中直接调用即可

```
- (IBAction)disconnet:(id)sender {
    self.stateLabel.text=@"自动下线了";
    [[SocketManager sharedInstance] cutOffSocket];
}
```

整个通信过程我所理解到的差不多就是这些，还有几个 AsyncSocket 的代理方法没有写出来，比如发生错误等。我不会后台，所以整个操作是用模拟器和手机完成的，使用 Demo是一定要将两部设备连接到同一网络，并设置正确的Host / Port 。来过的朋友觉得写的还行感谢给个喜欢～

@end



Wy\_chris (/u/16c6a4be7e38)

写了 1703 字，被 8 人关注，获得了 29 个喜欢

(/u/16c6a4be7e38)

+ 关注

如果觉得我的文章对您有用，请随意打赏。您的支持将鼓励我继续创作！

赞赏支持

🤍 喜欢 | 11


💬

🐼

🖼️

更多分享


(http://cwb.assets.jianshu.io/notes/images/9038912/



写下你的评论...

4条评论    只看作者

按喜欢排序    按时间正序    按时间倒序



电地暖商城客服 (/u/858531717e31)

2楼 · 2017.02.16 11:58


(/u/858531717e31)

受教了，感谢分享


👍 赞

💬 回复

Wy\_chris (/u/16c6a4be7e38):    @电地暖商城客服 (/users/858531717e31) 共同学习，一起进步

 2017.02.16 12:01    💬 回复

✍️ 添加新评论




逆夏流年\_d523 (/u/0e3e9921a1a0)


3楼 · 2017.02.21 17:36

(/u/0e3e9921a1a0)

~~~感谢分享

 赞    回复

Wy\_chris (/u/16c6a4be7e38):   @逆夏流年\_d523 (/users/0e3e9921a1a0) 😊😊😊  
2017.02.21 17:37    回复

 添加新评论

被以下专题收入，发现更多相似内容

-   
我的专题
- iOS学习
- IM
- 程序员
- 手机移动  
程序开发
- iOS  
Dev...