服务器端基本操作流程

服务器Socket对象->设置服务器IP->服务器开启端口->绑定IP和端口号->开始监听->接收或发送消息（字节数组和字符串的相互转化-Enconding.UTF8）->结束接收->关闭连接

1.创建一个服务器端Socket对象：Socket serverSocket = new Socket(…)

2.设置一个本机（解读）IP：IPAddress ipAddress = IPAddress.Parse("127.0.0.1");

3.

服务器同步、异步接收客户端连接

1.同步方式 Accept Receive

2.异步方式 BeginAccept EndAccept / BeginReceive EndReceive

客户端基本操作流程

1.创建一个客户端连接的Socket对象：Socket clientSocket = new Socket(…)

2.调用clientSocket.Connect方法

**粘包和分包**

该问题是利用Socket在TCP协议下内部的优化机制导致的

粘包指当频繁发送大量短信息时，TCP将多条消息打包进行发送，而此时服务器端Receive方法只能接收到一条信息，无法自动拆解；（还有一种情况是由接收方导致的，即放在缓冲区内的数据没能被及时取走，下一条信息直接与上一条合并放入缓冲区）

分包指一次性发送大体量信息时，数据被分成多个数据流分批发送，此时服务器会将一条消息解读成多条消息

解决方案：

对于所有发送和接收（服务器端和客户端同理）的信息使用一个Message类来处理，对消息进行组拼和解读；

由客户端发往服务器端的消息由“数据长度(int32)-RequestCode(int32)-ActionCode(int32)-真实数据”四段组成，服务器接收到一条消息后首先检查数据长度即读取前四个字节，是否与整条数据长度符合，只在符合时才拆解读取后三段数据。

注：涉及一个反射调用问题（回调？）

**服务器处理消息的流程**

在服务器端，有如下对应关系：

RequestCode <-> Controller，ActionCode <-> ClientRequest

服务器首先根据RequestCode找到对应的Controller（存在一个Controller对应多个ClientRequest的情形），然后根据ActionCode找到Controller中对应的处理方法。

MyServer.Client中创建并初始化MySqlConnection对象，

UserDAO中提供AuthenticatingUser(MySqlConnection conn, string username, string password)方法用于验证传递过来的用户真实性，

服务器对于请求是调用Controller进行的，故在UserController中提供Login方法来处理登录请求，而Login方法则通过调用UserDAO的方法来访问数据库，

**游戏主体部分**

Manager集

Request集

Panel集

GameFacede

当每个具体的Request类需要调用MessagePanel中用于显示提示信息的方法时，

BaseRequest持有GameFacade的单例引用

单例模式

GameFacade

BaseRequest

RegisterRequest

LoginRequest

当服务器给与游戏响应的时候（发回数据包ActionCode + data），ClientManager进行监听并拥有顶层回调方法private void ReceiveCallback(IAsyncResult ar)

但需要注意的是，回调方法不属于Unity主线程，所有该回调方法涉及的方法调用中均无法直接访问Unity中的游戏资源。

(关于UI Canvas中Rect Transform属性Anchors和Pivot：

Anchors为锚点，表示当父物体发生大小形变时当前物体的运动参考点

Pivot为支点、中心点，其位置用于自身的形变参考)