C:\Documents and Settings\Administrator\Application Data\Tencent\Users\339838619\QQ\WinTemp\RichOle\J_4XP5{GRN5K(%G]MTS(6%X.jpg

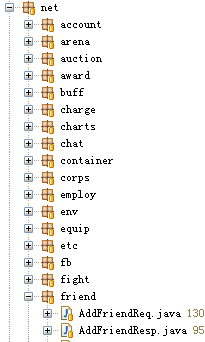
比如添加好友协议：

客户端发送以0x0823协议号为报头

byte添加方式+具体数据为报体

的“请求”

服务器端处理这个请求的类是



## AddFriendReq类

@Component

@Scope("prototype")

public class AddFriendReq extends BaseProtocalAction {

@Override

public boolean needSavePlayer() {

return true;

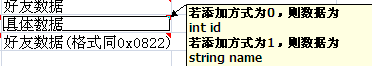
}

private static final int R\_CODE=0x0823;**// 这地方是协议号，也就是客户端请求的那个协议**

@Autowired

private FriendsService friendsService;

**//通过协议可知道客户端发送过来的具体数据会有id和name**

****

**因此这个地方声明2个变量playerId，playerName用来接受客户端传过来的数据，具体接受数据的方法是decodeBody**，**会根据数据类型进行接受，getInt()/getPrefixeString()，如果双方数据类型错误，就可能导致在这个地方出错。**

private int playerId;

private String playerName;

@Override

public void **decodeBody**(GBuffer in) {

byte type=in.get();

if(type == 0) {

playerId=in.getInt();

} else {

playerName=in.getPrefixedString();

}

}

// **execute这个方法是执行体，执行业务逻辑的部分。通过得到客户端传过来的数据，进行下一步，比如new一个好友，放入好友名姓名，调用好友Service类进行保存至数据库等操作。**

**如果需要返回客户端数据，就把数据通过return new xxxResp（）的方式传递给与Req相对应的Resp类进行处理**

@Override

public GMessage **execute**(IoSession session, GProtocalMessage req) {

Player me=this.getPlayer();

Player myFriend;

if(playerId != 0) {

myFriend=onlinemanager.getPlayerById(playerId);

} else {

myFriend=onlinemanager.getPlayerByPlayerName(playerName);

}

if(myFriend == null) {

if(playerId != 0) {

myFriend=playerservice.loadPlayerFromDBForAddFriend(playerId);

} else {

myFriend=playerservice.loadPlayerFromDBForAddFriend(playerName);

}

}

int r=friendsService.addFriend(me, myFriend);

return new AddFriendResp(r, myFriend);

}

@Override

public int getMsgCode() {

return R\_CODE;

}

}

## AddFriendResp类：

public class AddFriendResp extends S2CMessage {

private static final int R\_CODE=0x0824;**//协议号**

private Player myFriend;**//通过req传过来的对象，会在构造函数中接收**

private int r;**// 同上**

private AddFriendResp() {

setMsgCode(R\_CODE);

}

public AddFriendResp(int r, Player myFriend) {

this();

this.myFriend=myFriend;

this.r=r;

}

**//encodeBody这个方法主要用来返回给客户端数据，按照指定数据格式返回数据，需要注意每个数据的数据类型**

@Override

protected void **encodeBody**(GBuffer out) {

out.put((byte)r);

if(myFriend != null && r == 0) {

out.putPrefixedString(myFriend.getName());

out.putInt(myFriend.getUid());

out.put((byte)myFriend.getFightHeadId());

out.put((byte)myFriend.getSex().getId());

out.put((byte)myFriend.getLv());

out.put((byte)0);

}

}

}

这些协议码和相对应的处理类会在服务器启动的时候注册到内存中去。

可以通过net包下的GameServerHandler类的registActions方法去了解

这个GameServerHandler类会在初始化bean的时候执行registActions



Entity层就不用说了。

Service层就是写各种业务逻辑的。

Dao层也就是提供持久化的。

只要底层通信写好之后，再写就只涉及entity，net，service，dao这四层。  
基本流程就是上面所说的：

客户端和服务器定义协议，客户端请求相应的协议，服务端会由相应的类去接收客户端发来的数据，通过相关service层来处理业务逻辑，得到结果数据，通过resp返回给客户端。