# 第13讲 RMI程序设计（二）

**教学与实践目的**：进一步掌握RMI架构软件中的分布式程序设计技术。

## 一、回顾

远程调用是一种非常好的程序设计思想和技术，是现代WEB应用的前驱，也是目前许多基于C/S架构的APP应用软件的主流技术。RMI作为远程调用的方式之一，上一讲已经介绍了其编程的范式。

客户端可以调用服务端提供的远程服务，服务端也可以回调客户端的远程服务对象，从而实现真正的分布式、协同式软件。一些典型的应用需要服务器调用客户端的对象，如聊天系统中服务器需要刷新客户端的信息显示窗口；股票系统中服务器需要刷新客户端的实时交易数据信息等，因这些动作的事件触发在服务器端，而客户端是被动的。服务器主动刷新客户端的方式，是重要的推送服务程序设计技术。

在上一讲，已经了解了服务器如何主动调用客户端的远程服务，实现方式是客户端发布远程服务，由服务端“lookup”其远程服务，进行远程调用；其实还有另外一种使用方式：就是服务端定义的远程服务方法中，提供客户端的远程接口类型参数，客户端采用本地方法调用的方式，将自己的远程接口实现类作为参数传递给服务端，而不需要特意进行远程服务的发布。

## 二、程序任务描述

使用RMI的方式，实现简单的群聊功能。

关键技术：将客户端窗口的信息刷新方法定义成远程接口中的方法，由服务器远程调用实现刷新，从而实现由服务器控制客户聊天窗口信息的显示进程。

## 三、聊天程序设计

### 1. 创建远程接口

创建两个远程接口ClientService和ServerService，分别由客户端和服务端实现。接口定义如以下代码所示（再次强调，服务端和客户端都各自拥有相同的远程接口，特别要注意package的一致）。

**客户端远程接口ClientService：**

package **chapter13.rmi;**import java.rmi.Remote;  
import java.rmi.RemoteException;  
  
*/\*\*  
 \* 客户端远程对象接口，该接口为服务端提供回调服务  
 \*/*public interface ClientService extends Remote  
{  
 */\*\*  
 由服务端主动推送消息到客户端，客户端刷新聊天信息的方法；  
 该方法由客户端实现，服务端调用  
 \*/* public void showMsgToClient(String msg) throws RemoteException;  
}

**服务端远程接口ServerService：**

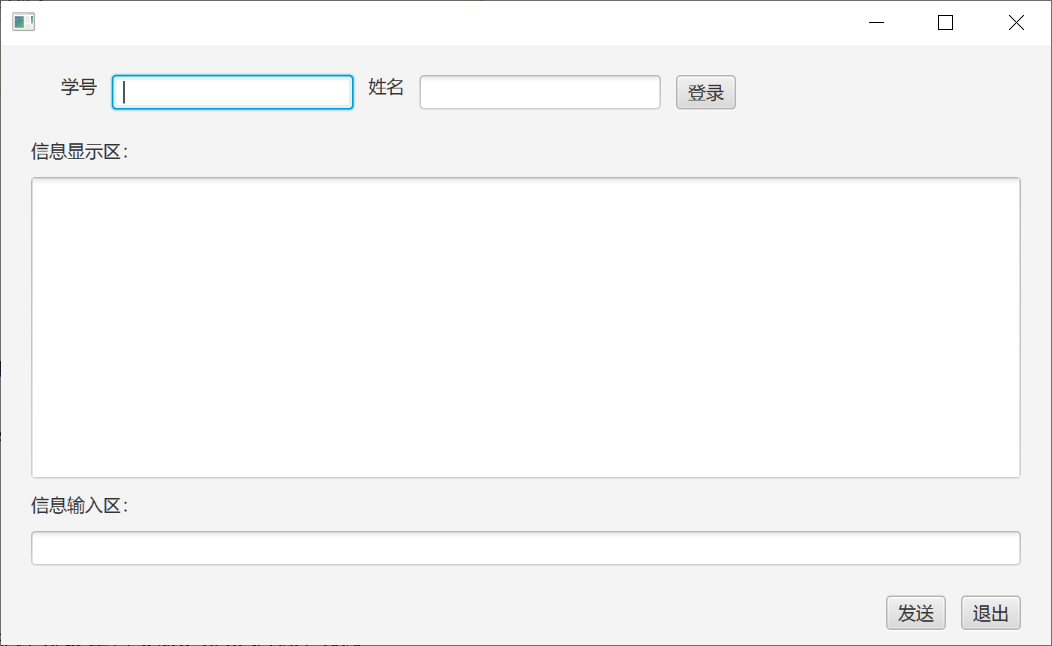
package **chapter13.rmi;**  
  
import java.rmi.Remote;  
import java.rmi.RemoteException;  
  
*/\*\*  
 \* 服务端远程对象接口，该接口为客户端提供服务  
 \* 由服务端实现以下远程方法  
 \*/*public interface ServerService extends Remote  
{  
 */\*\*  
 \* 客户加入群组的远程方法  
 \* @param client 格式为学号-姓名的字符串  
 \* @param clientService 用于将客户端的远程对象注入在线列表  
 \* @return 返回相关信息  
 \* @throws RemoteException  
 \*/* public String addClientToOnlineGroup(String client, ClientService clientService) throws RemoteException;  
  
  
 */\*\*  
 \* 客户退出群组的远程方法  
 \*/* public String removeClientFromOnlineGroup(String client,ClientService clientService) throws RemoteException;  
  
 */\*\*  
 \* 客户发送群聊信息的远程方法  
 \* @param client 格式为学号-姓名的字符串  
 \* @param msg 要发送的信息  
 \* @throws RemoteException  
 \*/* public void sendPublicMsgToServer(String client,String msg) throws RemoteException;  
}

### 2. 创建客户端远程接口实现类

package chapter13.client;  
  
import chapter13.rmi.ClientService;  
import java.rmi.RemoteException;  
import java.rmi.server.UnicastRemoteObject;  
  
public class **ClientServiceImpl** extends UnicastRemoteObject implements ClientService  
{  
 *//获取客户端窗体变量（第3步创建该客户端窗体）* private RmiClientFX rmiClientFX;  
 public ClientServiceImpl(RmiClientFX rmiClientFX) throws RemoteException  
 {  
 this.rmiClientFX = rmiClientFX;  
 }  
  
 @Override  
 public void showMsgToClient(String msg) throws RemoteException  
 {  
 *//调用客户端窗体中定义的刷新窗体信息显示的方法* rmiClientFX.appendMsg(msg);  
 }  
}

### 3. 创建客户端窗体（包含课堂计分）

该程序命名为RmiClientFX，参考界面如图13.1所示，**注意：**这个程序中包含课堂计分的操作，具体见下面程序中的相关注释。



13.1 群聊参考界面

部分参考代码如下：

package chapter13.client;  
......  
public class RmiClientFX extends Application  
{  
 *//定义成员变量*  
 private Button btnExit = new Button("退出");  
 private Button btnSend = new Button("发送");  
 private Button btnLogin = new Button("登录");  
 *//学号和姓名的文本框*  
 private TextField tfNO = new TextField();  
 private TextField tfName = new TextField();  
 private TextField tfMsg = new TextField();  
 *//显示信息的文本区域* private TextArea taDisplay = new TextArea();  
  
 *//服务端远程接口* private ServerService serverService;  
 *//客户端远程接口*  
 private ClientService clientService;  
 private String client;*//学号-姓名的格式* public static void main(String[] args)  
 {  
 *launch*(args);  
 }  
  
 @Override  
 public void start(Stage primaryStage)  
 {  
 *//初始化界面元素* initComponents(primaryStage);

*//初始化rmi相关操作* new Thread(() ->  
 {  
 initRmi();  
 }).start();

*//初始化事件* initEvent(primaryStage);  
 }  
  
 *//供外部调用的方法，用于在显示区添加信息，刷新窗体信息显示* public void appendMsg(String msg)  
 {  
 taDisplay.appendText(msg + "\n");  
 }

*//初始化界面元素* private void initComponents(Stage primaryStage)  
 {  
 BorderPane mainPane = new BorderPane();  
 *//登录区域* HBox hBoxLogin = new HBox();  
 hBoxLogin.setSpacing(10);  
 hBoxLogin.setPadding(new Insets(10, 20, 10, 20));  
 hBoxLogin.getChildren().addAll(new Label("学号"),  
 tfNO, new Label("姓名"), tfName, btnLogin);  
  
 *//内容显示区域* VBox vBox = new VBox();  
 vBox.setSpacing(10);*//各控件之间的间隔  
 //VBox面板中的内容距离四周的留空区域* vBox.setPadding(new Insets(10, 20, 10, 20));  
  
 ......

}

*//初始化rmi操作*  
 private void initRmi()  
 {  
 try  
 {  
 String ip = "202.116.195.71";

*//为了不和上一讲端口冲突，临时修改为8008，一般不做特别说明，是使用1099*  
 int port = 8008;  
 *//获取RMI注册器* Registry registry = LocateRegistry.*getRegistry*(ip, port);  
 for (String name : registry.list())  
 {  
 System.*out*.println(name);  
 }  
  
 *//客户端(调用端)到注册器中使用助记符寻找并创建远程服务对象的客户端(调用端)stub,  
 // 之后本地调用serverService的方法，实质就是调用了远程同名接口下的同名方法* serverService = (ServerService) registry.lookup("ServerService");  
 *//实例化本地客户端的远程对象* clientService = new ClientServiceImpl(this);  
 }  
 catch (Exception e)  
 {  
 e.printStackTrace();  
 }  
 }

*//初始化事件* private void initEvent(Stage primaryStage)  
 {  
  
 *//登录，加入在线用户群，格式为学号-姓名* btnLogin.setOnAction(event ->  
 {  
 try  
 {  
 String NO = tfNO.getText().trim();  
 String name = tfName.getText().trim();  
 if (!NO.equals("") && !NO.equals(""))  
 {  
 client = NO + "-" + name;

**//以下为课堂计分操作，加入在线用户组成功，完成任务得5分** *//调用服务端的远程方法，将客户加入在线用户组，并将客户端的远程对象注入，用于服务器跟踪客户*  
 String retStr = serverService.addClientToOnlineGroup(client, clientService);  
 taDisplay.appendText(retStr + "\n");  
 }  
 }  
 catch (RemoteException e)  
 {  
 e.printStackTrace();  
 }  
 });

*//发送按钮* btnSend.setOnAction(event ->  
 {  
 String sendMsg = tfMsg.getText();  
 try  
 {

*//调用服务端的远程服务，群发消息*  
 serverService.sendPublicMsgToServer(client, sendMsg);  
 }  
 catch (RemoteException e)  
 {  
 e.printStackTrace();  
 }  
 });  
  
 *//退出按钮* btnExit.setOnAction(event ->  
 {  
 *//调用退出方法*  
 exit();  
 });  
  
  
 *//响应窗体关闭* primaryStage.setOnCloseRequest(event ->  
 {  
 *//调用退出方法*  
 exit();  
 });  
  
 }  
   
 *//退出方法，一定要记住调用服务端远程方法清除登录记录*  
 private void exit()  
 {  
 try  
 {  
 serverService.removeClientFromOnlineGroup(client, clientService);  
 }  
 catch (RemoteException e)  
 {  
 e.printStackTrace();  
 }  
 finally  
 {  
 System.*exit*(0);  
 }  
 }  
}

### 4. 创建服务端远程接口实现类

前面的操作是调用教师服务器实现的远程服务，完成以上课堂得分相关操作。接下来进入自己编写服务端程序的阶段。

创建一个服务端远程接口实现类，该类命名为ServerServiceImpl，在这个类中，使用并发安全的map作为在线用户的存储集合，map的key使用学号-姓名格式的字符串，value使用客户端的远程对象实例。部分代码如下：

package chapter13.server;  
......  
public class **ServerServiceImpl** extends UnicastRemoteObject implements ServerService  
{  
 *//存储在线用户的map* private static ConcurrentHashMap<String,ClientService> *onlineGroup* = new ConcurrentHashMap<>();  
  
 protected ServerServiceImpl() throws RemoteException  
 {  
 }  
  
 @Override  
 *//客户加入群组的远程方法，在client中包含自己的学号和姓名，格式为学号-姓名* public String addClientToOnlineGroup(String client, ClientService clientService) throws RemoteException  
 {  
 if (client == null)  
 return "From 服务器：学号姓名信息为空";  
 if(client.split("-").length != 2)  
 return "From 服务器：学号姓名格式不正确";  
  
 boolean isLogin = false;  
 *//避免反复登录，关键是判断在线map中是否已经存在相同的clientService  
 // 自行实现判断代码......*

*......*  
 if (!isLogin)  
 {  
 *onlineGroup*.put(client.trim(), clientService);  
 isLogin = true;

*// 群发新用户上线的信息*  
 sendPublicMsgToServer(client, "加入到群聊！");  
 return "From 服务器：" + client.trim() + " 加入到群聊！";  
 }  
 else  
 {  
 return "From 服务器：不要反复登录";  
 }  
 }  
  
 @Override  
 *//客户退出群组的远程方法* public String removeClientFromOnlineGroup(String client,ClientService clientService) throws RemoteException  
 {  
 *//删除的判断依据应该是clientService，因为client可能被用户修改了*

*//此部分代码自行实现......*

*// 群发用户离线的信息*  
 sendPublicMsgToServer(client, "退出群聊！");  
 return "From 服务器：" + client.trim() + " 退出群聊！";  
 }  
  
 @Override  
 *// //客户发送群聊信息的远程方法* public void sendPublicMsgToServer(String client, String msg) throws RemoteException  
 {  
 if (msg != null)  
 {  
 *//遍历在线map，获得所有客户端远程对象，进行消息群发*

*//可能有客户端程序，退出时没有调用removeClientFromOnlineGroup，造成onlineGroup存在无效的用户值，群发消息时候，就会去连接这些不存在客户端的远程对象，造成socket超时错误，要考虑这种异常情况，否则可能导致所有客户端都无法使用* for (String onlineUser : *onlineGroup*.keySet())  
 {  
 *//调用客户端远程对象的刷新信息方法，推送信息到客户端* ClientService clientService = *onlineGroup*.get(onlineUser);  
 if (clientService != null)  
 {

*//此部分代码自行实现......*  }  
 }  
 }  
 }  
}

### 5. 创建服务端的远程服务发布程序

部分参考代码如下：

package chapter13.server;  
  
......  
public class ServerServiceServer  
{  
 public static void main(String[] args)  
 {

*//为了避免和上一讲1099端口冲突，临时使用8008*  
 int port = 8008;  
 try  
 {  
 *//对于有多个网卡的机器，建议用下面的命令指绑定固定的ip，因为默认是绑定到0.0.0.0  
 //System.setProperty("java.rmi.server.hostname",本机器的ip地址);  
 //第一步，启动RMI注册器* Registry registry = LocateRegistry.*createRegistry*(port);  
 *//第二步，实例化远程服务对象* ServerService serverService = new ServerServiceImpl();  
 *//第三步，用助记符来注册及发布远程服务对象,助记符建议和远程服务接口命名相同，方便使用* registry.rebind("ServerService",serverService);  
 System.*out*.println("服务端发布了ServerService远程服务");  
 }  
 catch (RemoteException e)  
 {  
 e.printStackTrace();  
 }  
 }  
}

### 6. 本地测试

（1）本地测试首先修改RmiClientFX客户端中initRmi方法中的ip地址，从服务器ip修改为本地的ip，表示使用本机的服务端远程服务。

（2）先运行服务端的远程服务发布程序ServerSerivceServer，然后运行多个客户端RmiClientFX（关于在idea下，如何并行运行多个同名程序，参见第5讲的图5.1），模拟多个用户同时登录，同时测试用户反复登录使用有没有异常；

特别要测试这种情况：退出时不调用removeClientFromOnlineGroup清除在线记录的情况。即客户端登录使用，不清除在线记录关闭窗体，再开客户端窗体登录使用，看自己的的代码能不能应付这种情况（这是非常常见的情况，毕竟无法控制所有客户端的情况。如果代码处理不当，客户端再次登录发送信息，会出现异常造成客户端无法使用）。

请思考RMI方式的实现和TCP/UDP实现上有和区别？

## 四、扩展练习

这个聊天程序是实现群聊，尝试修改对应的服务端、客户端，实现一对一、一对多的私聊功能，并相应完善使用界面；

思考RMI方式如何实现类似第4讲的文件下载。