相关技术及开发流程介绍

11计算机1班 王立鹏

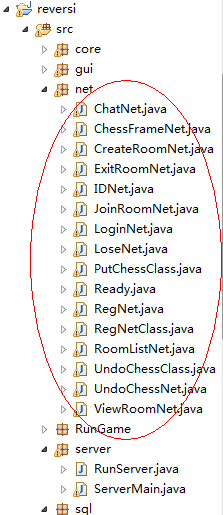
201103130138

刚接到题目的时候我就在思考怎么实现整个项目，刚开始几天对整体架构没有什么概念，就先写了一份单机版，当然最后的完成版本是从服务器开始写的，采用C/S架构，但单机版作为纪念我没有移除，在完整版中运行登陆界面后如果用户名为single，密码忽略不填则会进入单机版。

好了，下面开始详细说一下完整版的各种技术：

1. server工作在被动模式中，即server不会主动向任何客户端（登陆界面，注册界面，大厅，房间）主动发送任何信息。所有信息均有客户端首先发起，然后server收到信息后判断是何种信息然后做相应处理返回给客户端。

为了实现客户端和server端能够实现各种通信（下棋、聊天、建立房间的等）我实现了很多net类：



Server收到客户端的信息后会解开信息，查看信息标记，然后分情况处理：

**public** **void** run()

{

**try** {

in = **new** ObjectInputStream(**new** BufferedInputStream(incoming.getInputStream()));

out = **new** ObjectOutputStream(**new** BufferedOutputStream(incoming.getOutputStream()));

**try**{

IDNet cmd = (IDNet) in.readObject();

**int** cmdType = cmd.getID();

**switch** (cmdType)

{

**case** 0:

{

………………………………………………..

break;

}

…………………………………….

**case** 12:

{

………………………………………………..

break;

}

}

}

1. 用户信息管理使用成熟的小型关系型数据库mysql，小巧轻便，同时做到server方便安全管理用户信息。
2. 心跳检测：server维护着一个当前在线用户的列表***private******static*** *Vector<User> userList;* 一个当前房间的列表：***private******static*** *Vector<Room> roomList;* ， server有一个线程定期清楚所有在线用户：

/\*

\* 定时清除登录的用户列表，用于"心跳检测"

\*/

**class** ClearUsers **extends** Thread

{

**public** **void** run()

{

**while**(**true**)

{

***userList*.clear();**

**try** {

*sleep*(30000);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

}

}

然后用户在大厅界面每秒会向服务器申请一次当前房间列表：

**class** refreshTable **extends** Thread

{

**public** **void** run()

{

**while**(**true**)

{

**refreshFunc();**

**try** {

*sleep*(1000);

} **catch** (InterruptedException e) {

// **TODO** Auto-generated catch block

//e.printStackTrace();

}

}

}

}

**refreshFunc() 功能向server发送一个申请包，里面包含此用户信息，服务器接受到后给客户端回复一个包，包含着当前房间列表，同时把此用户重新加到在线用户列表中：**

**case** 5://房间列表

{

RoomListNet roo = (RoomListNet) cmd;

**boolean** inUserList = **false**;

**for**(User u:*userList*)

{

**if**(u.getUsername().equals(roo.getUser().getUsername()))

{

inUserList = **true**;

**break**;

}

}

**if**(!inUserList)

{

*userList*.add(roo.getUser());

}

RoomListNet net = **new** RoomListNet(*roomList*);

out.writeObject(net);

out.flush();

**break**;

}

这样就可以大致维护一个在线用户列表，同时**客户端大厅房间列表采用Jtable，获取到房间列表后会自动刷新显示房间列表。**

1. 此项目实现了维护注册唯一性、登陆唯一性、创建房间唯一性、进入房间唯一性。

**注册唯一性：**一个用户名只允许注册一次。

**登陆唯一性：**一个用户名只允许同时登录一次。

**创建房间唯一性：**一个用户在当前登录状态下只允许同时创建一个房间。

**进入房间唯一性：**当前用户只允许在一个房间中或在大厅中等待。

**5．**在房间界面，棋手图片由性别决定，不同性别的图片不相同。

**public** **void** setImage(**int** u1,**int** u2)

{

**this**.u1 = **new** ImageIcon("res"+File.*separator*+u1+".png");

**this**.u2 = **new** ImageIcon("res"+File.*separator*+u2+".png");

}

**6.**无上限限制当前房间数量，可在服务器内存允许下创建尽量多的房间，允许**百万级**用户同时在线，每个房间不限制观看游戏人数数量，无上限限制注册用户数量，可在服务器数据库承受能力下创建尽量多的用户。

**7.**客户端下棋界面设计成丢弃所有用户无效输入（点击棋盘无效位置等），这样就减少了客户端与server之间的网络通讯，减少网络流量，减少服务器负载。

**if**(x <= startx || y <= starty || x >= 628 || y >= 612) **return** ;

**if**(chessManList.havaChessman(pos\_x, pos\_y) || !canPlace.havaChessman(pos\_x, pos\_y)) **return** ;

检测到鼠标点击事件后判断是否在有效位置点击的鼠标，如果不是的话直接丢弃。

**8.**server会自动统计当前房间人数，如果发现此房间没有用户则会自动销毁此房间，减少垃圾空间占用，同时释放此房间名，允许在线用户再次创建这个名称的房间。

/\*

\* 用来从房间或大厅中清除用户,清除没有玩家的房间

\*/

**private** **void** removeUser(User user)

{

**int**[] needRemove = **new** **int**[MAXUSER];

needRemoveNum = 0;

**int** num = 0;

**for**(User u:*userList*)

{

**if**(u.getUsername().equals(user.getUsername()))

{

needRemoveNum++;

needRemove[needRemoveNum] = num;

}

num++;

}

**for**(**int** i = 1;i<=needRemoveNum;i++)

{

*userList*.remove(needRemove[i]);

}

**int**[] needRemoveRoom = **new** **int**[MAXUSER];

**int**[] needRemoveUser = **new** **int**[MAXUSER];

**int** needRemoveRoomNum = 0;

**int** needRemoveUserNum = 0;

**int** numRoom = 0;

**for**(Room r:*roomList*)

{

Vector<User> ul = r.getUserList();

needRemoveUserNum = 0;

num = 0;

**for**(User u:ul)

{

**if**(u.getUsername().equals(user.getUsername()))

{

needRemoveUserNum++;

needRemoveUser[needRemoveUserNum] = num;

}

num++;

}

**for**(**int** i = 1;i<=needRemoveUserNum;i++)

{

**if**(ul.get(needRemoveUser[i]).getUsername().equals(r.getPlayer1()))

{

r.setPlayer1(**new** User("",3,"","",0,0));

}

**if**(ul.get(needRemoveUser[i]).getUsername().equals(r.getPlayer2()))

{

r.setPlayer2(**new** User("",3,"","",0,0));

}

ul.remove(needRemoveUser[i]);

}

**if**(ul.isEmpty())

{

needRemoveRoomNum++;

needRemoveRoom[needRemoveRoomNum] = numRoom;

}

numRoom++;

}

**for**(**int** i = 1;i<=needRemoveRoomNum;i++)

{

*roomList*.remove(needRemoveRoom[i]);

}

needRemoveRoomNum = 0;

num = 0;

**for**(Room r:*roomList*)

{

**if**(r.getPlayer1().equals("")&&r.getPlayer2().equals(""))

{

needRemoveRoomNum++;

needRemoveRoom[needRemoveRoomNum] = num;

}

num++;

}

**for**(**int** i = 1;i<=needRemoveRoomNum;i++)

{

*roomList*.remove(needRemoveRoom[i]);

}

}

**9.**当客户端房间关闭时，客户端会向服务器发送信息，告诉服务器此用户退出房间，方便服务器统计房间人数。

**this**.addWindowListener(**new** WindowAdapter() {

**public** **void** windowClosed(WindowEvent e){

user.setPlayer(**false**);

ExitRoomNet exit = **new** ExitRoomNet(user);

**try** {

**if**(t!=**null**)

t.~~stop~~();

connect();

ObjectOutputStream out = **new** ObjectOutputStream(**new** BufferedOutputStream(*s*.getOutputStream()));

out.writeObject(exit);

out.flush();

out.close();

} **catch** (Exception e1) {

// **TODO** Auto-generated catch block

e1.printStackTrace();

}

}

**10.**用户密码储存在数据库中并用md5($password)方式储存，这样即使发生安全事故，比如数据库泄露也不会直接泄漏用户密码，防止发生csdn泄漏用户资料的问题发生。相关链接（<http://tech.qq.com/a/20111223/000056.htm>）

String passwd = GenMD5.*getMD5*(pass);

其中GenMD5.getMD5()是一个静态方法。返回输入字符串的md5值。

**11.**游戏结束后会更新用户积分信息

/\*

\* 下棋结束后更新用户信息

\*/

**private** **void** upUserInfo(String name,**int** Score)

{

String cm = "UPDATE users SET score = "+Score+" WHERE username = \'"+name+"\'";

**try** {

ExecSql.*state*.execute(cm);

} **catch** (SQLException e) {

// **TODO** Auto-generated catch block

e.printStackTrace();

}

}

**12.**大厅列表中房间状态自动更新：当创建房间后房间状态为**等待玩家**，当开始游戏后房间状态会自动变为**游戏中。。。**所有用户大厅中房间列表都会得到更新

**13**.服务器配置与客户端配置文件化：在项目文件夹中的config文件夹下有两个配置文件，database.props里面储存服务器用到的相关数据库信息，方便在不改动代码的情况下切换数据库，也可以实现server与数据库server分开独立运行，增加项目稳定性。serverAddress.props是客户端读取的服务器地址，方便用户修改服务器地址。

**private** **static** Connection getConnection() **throws** IOException, SQLException

{

Properties props = **new** Properties();

FileInputStream in = **new** FileInputStream("config"+File.*separator*+"database.props");

props.load(in);

in.close();

String drivers = props.getProperty("jdbc.drivers");

**if**(drivers != **null**) System.*setProperty*("jdbc.drivers", drivers);

String url = props.getProperty("jdbc.url");

String username = props.getProperty("jdbc.username");

String passwd = props.getProperty("jdbc.password");

**return** DriverManager.*getConnection*(url, username, passwd);

}

上面这个是数据库连接时使用的读取配置文件方面的代码

/\*

\* 连接服务器

\*/

**private** **void** connect() **throws** Exception

{

Properties props = **new** Properties();

FileInputStream in = **new** FileInputStream("config"+File.*separator*+"serverAddress.props");

props.load(in);

in.close();

String serveraddr = props.getProperty("Server");

*s* = **new** Socket();

**try** {

*s*.connect(**new** InetSocketAddress(serveraddr, 8090),3000);

} **catch** (IOException e) {

// **TODO** Auto-generated catch block

//e.printStackTrace();

}

}

上面这个是客户端与服务器连接时使用的读取配置文件方面的代码

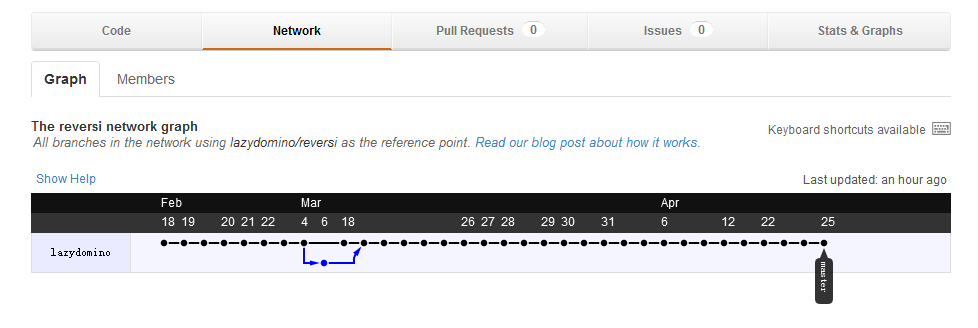
**14.**使用了java虚拟机环境检测，可以在不修改源代码的情况下在windows和linux等其他系统下正常工作：

**this**.u1 = **new** ImageIcon("res"+File.*separator*+u1+".png");

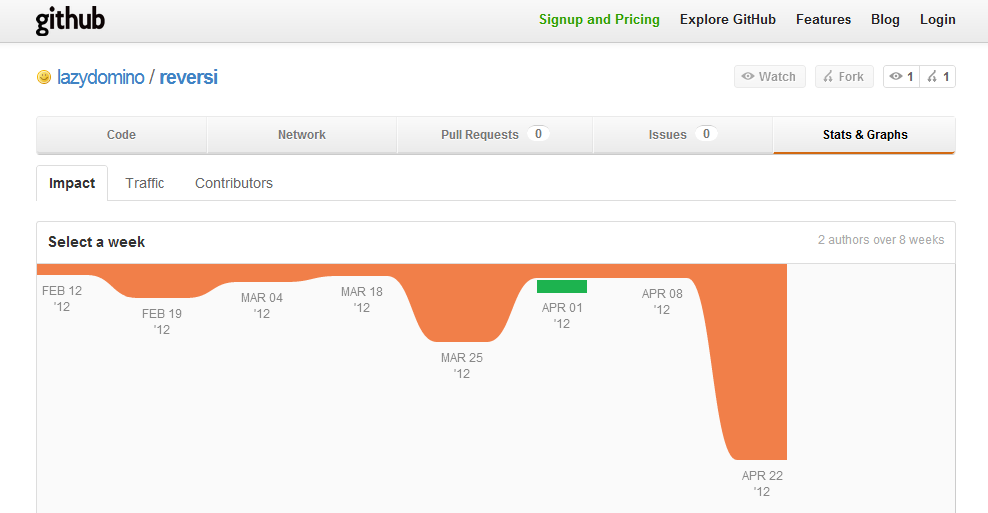
**this**.u2 = **new** ImageIcon("res"+File.*separator*+u2+".png");

**15.**开发过程使用git（一个分布式版本管理工具）配合github.com管理代码，每次作出重大代码改动或实现某些功能后就将代码作为新版本提交到github上，这样就保存了开发历史，可以恢复到任何一个代码变动版本，方便代码改动，不会出现代码写乱后无法恢复到上一版本的情况，具体项目主页，请移步到：<https://github.com/lazydomino/reversi>

在github上还可以看到此项目的发展状况，例如开发进度：



可以看到项目从建立到现在，中间没有出现大分支，蓝色分支是有一次我在机房用机房电脑递交后合并到master主开发线路上的。



这张图代表的是代码变动，中间我重装过电脑系统，所以github的ssh的rsa验证信息有所变动，不同颜色代表不同认证信息对代码变动做的贡献。

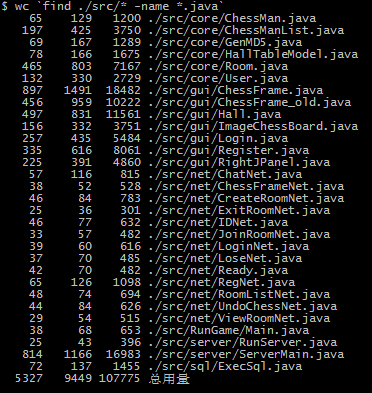
令：*可以用git工具从github代码仓库中获取当前最新开发版本*

*Windows用户建议使用msysgit工具（<http://code.google.com/p/msysgit/>），linux/Mac 及其他类unix用户使用相应开发版本的git工具均可以从github上colon该项目所有代码及历史变动。*

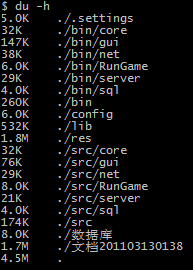
*可以使用：git clone https://github.com/lazydomino/reversi.git* *命令克隆该项目所有代码及其他附件*

**16.统计：**

该项目到提交课程设计时代码量达到5000+级别：



项目总文件大小（包含图片，配置文件，数据库，文档）为4.5MB



***----王立鹏（lazydomino）完成与2012.4.25***