Server.java

/\*

\*服务器的主要函数，实现一个简单的管理界面，多线程实现的服务器

\*/

package newServer;

import java.awt.BorderLayout;

import java.awt.Color;

import java.awt.Font;

import java.awt.GridLayout;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.io.BufferedReader;

import java.io.BufferedWriter;

import java.io.DataInputStream;

import java.io.DataOutputStream;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileOutputStream;

import java.io.FileReader;

import java.io.FileWriter;

import java.io.IOException;

import java.io.InputStreamReader;

import java.net.ServerSocket;

import java.net.Socket;

import java.text.DateFormat;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import javax.swing.JButton;

import javax.swing.JFrame;

import javax.swing.JLabel;

import javax.swing.JPanel;

/\*\*

\*

\* @author zhuqihao

\*/

public class Server extends JFrame implements Runnable {

// 约定使用的端口为8888

private int port = 8888;

private ServerSocket listen\_socket;

// 规定统一使用的字体

String font = "Xingkai SC";

Thread thread;

// 当前连接数目

static int connectionnumber = 0;

// 存储访问客户端的用户

static ArrayList clients;

// 声明界面的小部件

JPanel diary;

JPanel activeconnection;

JPanel dcontent;

JLabel diarytitle;

JLabel activeconnectiontitle;

JLabel title1;

JLabel title2;

JLabel title3;

JLabel title4;

JLabel activetitle;

// 最多显示11个活跃连接

JPanel connection[] = new JPanel[11];

// 取消连接

JButton connectionbutton[] = new JButton[11];

static JLabel connectionlabel[] = new JLabel[11];

JPanel activecontent;

static JLabel content[][] = new JLabel[10][4];

static ArrayList<String> diaryusr\_id;

static ArrayList<String> diaryusr\_mode;

static ArrayList<String> diaryusr\_result;

static ArrayList<String> diaryusr\_time;

public void ServerListen() {

try {

// 新建监听套接字

listen\_socket = new ServerSocket(port);

} catch (Exception e) {

e.printStackTrace();

}

clients = new ArrayList();

thread = new Thread(this);

// 启动服务器主线程

thread.start();

}

public Server() throws IOException {

diaryusr\_id = new ArrayList<String>();

diaryusr\_mode = new ArrayList<String>();

diaryusr\_result = new ArrayList<String>();

diaryusr\_time = new ArrayList<String>();

setBounds(250, 100, 800, 500);

diary = new JPanel();

activeconnection = new JPanel();

this.setLayout(new GridLayout(1, 2));

add(diary);

add(activeconnection);

initJPanel();

this.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

setVisible(true);

// 监听接口

ServerListen();

}

// 初始化部件的摆放和布局，左边4\*11的网格布局，右边1\*11的网格布局

public void initJPanel() {

diarytitle = new JLabel("活动日志", JLabel.CENTER);

diarytitle.setSize(400, 50);

diarytitle.setBackground(Color.white);

diarytitle.setOpaque(true);

diarytitle.setFont(new Font(font, 0, 18));

diary.setLayout(new BorderLayout());

diary.add(diarytitle, BorderLayout.NORTH);

diary.setBackground(Color.red);

dcontent = new JPanel();

diary.add(dcontent, BorderLayout.CENTER);

dcontent.setSize(400, 450);

dcontent.setBackground(Color.white);

dcontent.setLayout(new GridLayout(11, 4, 3, 3));

diary.add(dcontent);

title1 = new JLabel("客户端id", JLabel.CENTER);

title1.setOpaque(true);

title2 = new JLabel("模式", JLabel.CENTER);

title2.setOpaque(true);

title3 = new JLabel("结果", JLabel.CENTER);

title3.setOpaque(true);

title4 = new JLabel("时间", JLabel.CENTER);

title4.setOpaque(true);

dcontent.add(title1);

dcontent.add(title2);

dcontent.add(title3);

dcontent.add(title4);

for (int i = 0; i < 10; ++i) {

for (int j = 0; j < 4; ++j) {

content[i][j] = new JLabel();

content[i][j].setOpaque(true);

content[i][j].setHorizontalAlignment(JLabel.CENTER);

dcontent.add(content[i][j]);

if (j == 3) {

content[i][j].setFont(new Font("", 0, 8));

}

}

}

activetitle = new JLabel("当前活跃连接", JLabel.CENTER);

activetitle.setSize(400, 50);

activeconnection.setLayout(new BorderLayout());

activeconnection.add(activetitle, BorderLayout.NORTH);

activetitle.setBackground(Color.white);

activetitle.setOpaque(true);

activetitle.setFont(new Font(font, 0, 18));

activecontent = new JPanel();

activecontent.setBackground(Color.white);

activeconnection.add(activecontent, BorderLayout.CENTER);

activeconnection.setSize(400, 450);

activecontent.setLayout(new GridLayout(11, 1, 3, 3));

activecontent.setBackground(Color.white);

for (int i = 0; i < 11; ++i) {

connection[i] = new JPanel();

activecontent.add(connection[i]);

connection[i].setLayout(null);

connectionlabel[i] = new JLabel();

connectionlabel[i].setOpaque(true);

connection[i].add(connectionlabel[i]);

connectionlabel[i].setBounds(0, 0, 300, 40);

connectionbutton[i] = new JButton("关闭连接");

connectionbutton[i].addMouseListener(new MouseAdapter() {

public void mouseClicked(MouseEvent e) {

for (int j = 0; j < 11; ++j) {

if (e.getSource() == connectionbutton[j]) {

((Connection) clients.get(j)).close();

break;

}

}

}

});

connectionbutton[i].setOpaque(true);

connection[i].add(connectionbutton[i]);

connectionbutton[i].setBounds(300, 0, 100, 40);

}

}

// 显示最新的10个活动日志

public synchronized static void set\_diary() {

try {

int size = diaryusr\_id.size();

for (int i = size - 1; i >= 0 && i >= size - 10; --i) {

content[size - 1 - i][0].setText(diaryusr\_id.get(i));

content[size - 1 - i][1].setText(diaryusr\_mode.get(i));

content[size - 1 - i][2].setText(diaryusr\_result.get(i));

content[size - 1 - i][3].setText(diaryusr\_time.get(i));

}

} catch (Exception e) {

System.out.print(e);

}

}

// 显示连接的用户名

public synchronized static void setconnection() {

for (int i = 0; i < 11; ++i) {

if (i < clients.size()) {

Connection t = (Connection) clients.get(i);

connectionlabel[i].setText("用户名：" + t.usr\_id);

} else {

connectionlabel[i].setText("");

}

}

}

public static void main(String[] args) throws IOException {

new Server();

}

public void run() {

try {

while (true) {

// 监听窗口

Socket client\_socket = listen\_socket.accept();

Connection c = new Connection(client\_socket, connectionnumber++);

clients.add(c);

// setconnection();

}

} catch (Exception e) {

e.printStackTrace();

}

}

class Connection extends Thread {

private Socket client;

// 套接字的接受和输出字节流

private DataInputStream cin;

private DataOutputStream cout;

private int id;

// 连接用户名

String usr\_id = new String();

public Connection(Socket client, int connectionnumber) {

this.client = client;

id = connectionnumber;

try {

cin = new DataInputStream(client.getInputStream());

cout = new DataOutputStream(client.getOutputStream());

usr\_id = cin.readUTF();

} catch (Exception e) {

System.out.print(e);

}

this.start();

}

public void run() {

String mode = "";

// 循环接受固定的指令，做出相应的回应

while (true) {

if (client.isClosed()) {

break;

}

if (cin == null) {

System.out.println("end");

break;

}

try {

mode = cin.readUTF();

set\_diary();

String result = new String();

// 验证用户名和密码

if (mode.equals("usr\_password")) {

result = usr\_passwordconfirm();

cout.writeUTF(result);

}

// 创建新用户

if (mode.equals("usr\_newusr")) {

result = "success";

String new\_id = new\_usr();

cout.writeUTF(new\_id);

}

// 修改用户信息

if (mode.equals("change\_usr\_info")) {

result = "success";

usr\_change();

}

// 关闭套接字

if (mode.equals("bye")) {

result = "closed";

close();

}

// 获取用户信息

if (mode.equals("getUsr")) {

result = "success";

getusrinfo();

}

// 获取博文标题

if (mode.equals("getblogtext")) {

result = "success";

getblogtext();

}

// 获取博文主要内容

if (mode.equals("getblogmaintext")) {

result = "success";

getblogmaintext();

}

// 获取博文评论

if (mode.equals("getblogcomment")) {

result = "success";

getblogcomment();

}

// 创建博客

if (mode.equals("createblog")) {

result = "success";

createblog();

}

// 添加评论

if (mode.equals("addcomment")) {

result = "success";

addcomment();

}

// 获取当前已有的博客数目

if (mode.equals("getblognumber")) {

result = "success";

getblognumber();

}

Server.diaryusr\_id.add(usr\_id);

Server.diaryusr\_mode.add(mode);

Server.diaryusr\_result.add(result);

Date date = new Date();

DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm");

String time = format.format(date);

Server.diaryusr\_time.add(time);

// set\_diary();

} catch (Exception e) {

close();

System.out.println(mode);

e.printStackTrace();

break;

}

Server.setconnection();

}

Server.setconnection();

}

// 创建新用户

public String new\_usr() throws Exception {

String usr\_name = cin.readUTF();

String usr\_password = cin.readUTF();

String usr\_email = cin.readUTF();

// 将用户数目进行自增

File usr\_number = new File("src/usr/usr\_number.txt");

if (!usr\_number.exists()) {

usr\_number.createNewFile();

}

BufferedReader br = new BufferedReader(new FileReader(usr\_number));

String usr\_numberString = br.readLine();

br.close();

// 创建相应的用户文件

usr\_id = String.valueOf(Integer.parseInt(usr\_numberString) + 1);

BufferedWriter bw = new BufferedWriter(new FileWriter(usr\_number));

bw.write(usr\_id);

bw.flush();

bw.close();

Usr\_Info usr = new Usr\_Info(usr\_id, usr\_password, usr\_email, usr\_name);

Usr\_Info.writeUser(usr);

return usr\_id;

}

// 验证密码和用户名的对应性

public String usr\_passwordconfirm() throws Exception {

String usr\_id = cin.readUTF();

String usr\_password = cin.readUTF();

String filename = "src/usr/" + usr\_id + ".txt";

if (!(new File(filename)).exists()) {

return "not matched";

}

Usr\_Info user = Usr\_Info.readUser(filename);

if (user.getKeyword().equals(Encryption.Encrypt(usr\_password))) {

return "matched";

} else {

return "not matched";

}

}

// 修改用户信息

public void usr\_change() throws Exception {

String filename = "src/usr/" + usr\_id + ".txt";

System.out.println(filename + " in server");

String name, id, keyword, star, age, job, hobby, mail;

name = cin.readUTF();

id = cin.readUTF();

keyword = cin.readUTF();

star = cin.readUTF();

age = cin.readUTF();

job = cin.readUTF();

hobby = cin.readUTF();

mail = cin.readUTF();

/\*\*

\* Debug System.out.println(name); System.out.println(id);

\* System.out.println(keyword); System.out.println(star);

\* System.out.println(age); System.out.println(job);

\* System.out.println(hobby); System.out.println(mail);

\*/

Usr\_Info temp = Usr\_Info.readUser(filename);

temp.setAge(age);

temp.setHobby(hobby);

temp.setJob(job);

temp.setMail(mail);

temp.setName(name);

temp.setStar(star);

Usr\_Info.writeUser(temp);

}

// 获取用户信息，先读入，再写出

public void getusrinfo() throws Exception {

String usrname = cin.readUTF();

String filename = "src/usr/" + usrname + ".txt";

BufferedReader in = new BufferedReader(new FileReader(new File(filename)));

String name, id, keyword, star, age, job, hobby, mail;

// System.out.println("read begin");

name = in.readLine();

id = in.readLine();

keyword = in.readLine();

star = in.readLine();

age = in.readLine();

job = in.readLine();

hobby = in.readLine();

mail = in.readLine();

// System.out.println("readend");

cout.writeUTF(name);

cout.writeUTF(id);

cout.writeUTF(star);

cout.writeUTF(age);

cout.writeUTF(job);

cout.writeUTF(hobby);

cout.writeUTF(mail);

// System.out.print("writend");

in.close();

}

// 获取blog的标题和图片什么

public void getblogtext() throws Exception {

int kind = cin.readInt();

int id = cin.readInt();

// debug

// System.out.println(id);

String filename = "src/data/" + kind + '/' + id + "/abstract.txt";

BufferedReader in = new BufferedReader(new FileReader(new File(filename)));

String title = in.readLine();

// debug

// System.out.println(title);

String authorname = in.readLine();

String time = in.readLine();

String commentid = in.readLine();

// System.out.println(commentid);

cout.writeUTF(title);

cout.writeUTF(authorname);

cout.writeUTF(time);

cout.writeUTF(commentid);

in.close();

getfile(new File("src/data/" + kind + '/' + id), true);

}

// 获取博文主体

public void getblogmaintext() throws Exception {

int kind = cin.readInt();

int id = cin.readInt();

File f = new File("src/data/" + kind + "/" + id + "/text");

File[] files = f.listFiles();

cout.writeInt(files.length);

for (int i = 0; i < files.length; ++i) {

Thread.sleep(10);

DataInputStream fin = new DataInputStream(new FileInputStream(files[i]));

cout.writeUTF(files[i].getName());

cout.writeLong(files[i].length());

byte[] buf = new byte[8000];

while (true) {

int read = 0;

read = fin.read(buf);

if (read == -1) {

break;

}

cout.write(buf, 0, read);

}

}

}

// 获取博客评论

public void getblogcomment() throws Exception {

int kind = cin.readInt();

int blogid = cin.readInt();

int commentid = cin.readInt();

File f = new File("src/data/" + kind + "/" + blogid + "/" + commentid + ".txt");

BufferedReader in = new BufferedReader(new InputStreamReader(new FileInputStream(f)));

String author = in.readLine();

cout.writeUTF(author);

String time = in.readLine();

cout.writeUTF(time);

String ans = new String();

String buff = new String();

ans = in.readLine();

while (ans != null) {

buff += ans;

ans = in.readLine();

}

cout.writeUTF(buff);

}

// 创建博文

public void createblog() throws Exception {

int kind = cin.readInt();

BufferedReader in = new BufferedReader(

new InputStreamReader(new FileInputStream("./src/data/" + kind + "/number.txt")));

int number = Integer.parseInt(in.readLine());

++number;

FileWriter out = new FileWriter(new File("./src/data/" + kind + "/number.txt"));

out.write(String.valueOf(number));

out.flush();

File f = new File("./src/data/" + kind + "/" + number + "/abstract.txt");

if (!f.exists()) {

f.getParentFile().mkdirs();

f.createNewFile();

}

out = new FileWriter(f);

out.write(cin.readUTF() + '\n');

out.write(cin.readUTF() + '\n');

out.write(cin.readUTF() + '\n');

out.write(cin.readUTF() + '\n');

out.flush();

putfile("./src/data/" + kind + "/" + number, true);

int filenum = cin.readInt();

new File("./src/data/" + kind + "/" + number + "/text").mkdirs();

for (int i = 0; i < filenum; ++i) {

String filename = cin.readUTF();

long flength = cin.readLong();

File file = new File("./src/data/" + kind + "/" + number + "/text/" + filename);

file.createNewFile();

DataOutputStream fout = new DataOutputStream(new FileOutputStream(file));

byte buf[] = new byte[8000];

long rlen = 0;

while (true) {

int read = 0;

read = cin.read(buf);

rlen += read;

fout.write(buf, 0, read);

if (rlen == flength)

break;

}

fout.flush();

fout.close();

}

/\*

\* out = new FileWriter(new File("./src/data/" + number +

\* "/text.txt")); out.write(cin.readUTF() + '\n'); out.flush();

\*/

}

// 添加评论

public void addcomment() throws Exception {

int kind = cin.readInt();

int blogid = cin.readInt();

String author = cin.readUTF();

String commenttime = cin.readUTF();

String commenttext = cin.readUTF();

File f = new File("src/data/" + kind + "/" + blogid + "/abstract.txt");

BufferedReader in = new BufferedReader(new FileReader(f));

String title = in.readLine();

String authorname = in.readLine();

String time = in.readLine();

String commentid = in.readLine();

int cid = Integer.parseInt(commentid);

++cid;

commentid = String.valueOf(cid);

FileWriter out = new FileWriter(f);

out.write(title + '\n');

out.write(authorname + '\n');

out.write(time + '\n');

out.write(commentid + '\n');

out.flush();

f = new File("src/data/" + kind + "/" + blogid + "/" + cid + ".txt");

f.createNewFile();

out = new FileWriter(f);

out.write(author + '\n');

out.write(commenttime + '\n');

out.write(commenttext + '\n');

out.flush();

in.close();

}

// 获得服务器上blog的数目

public void getblognumber() throws Exception {

int kind = cin.readInt();

BufferedReader in = new BufferedReader(new FileReader(new File("src/data/" + kind + "/number.txt")));

cout.writeInt(Integer.parseInt(in.readLine()));

in.close();

}

public void close() {

try {

client.close();

for (int i = 0; i < Server.clients.size(); ++i) {

if (usr\_id.equals(((Connection) Server.clients.get(i)).usr\_id)) {

Server.clients.remove(i);

break;

}

}

Server.setconnection();

} catch (Exception e) {

e.printStackTrace();

}

}

// 获取文件

public void getfile(File h\_file, boolean read\_image) throws IOException {

System.out.println(h\_file.getPath());

File file = new File("");

File[] filelist = h\_file.listFiles();

for (File x : filelist) {

if (read\_image && x.getName().indexOf("icon") >= 0) {

file = x;

break;

} else if (!read\_image && x.getName().equals("text.html")) {

file = x;

break;

}

}

DataInputStream fInputStream = new DataInputStream(new FileInputStream(file));

cout.writeUTF(file.getName());

cout.writeLong(file.length());

byte buf[] = new byte[8000];

while (true) {

int read = 0;

read = fInputStream.read(buf);

if (read == -1) {

break;

}

cout.write(buf, 0, read);

}

}

// 接受客户端传输的文件

public void putfile(String filePath, boolean put\_image) throws IOException {

String filename = cin.readUTF();

long flength = cin.readLong();

String filepath = filePath;

if (put\_image) {

filepath += "/icon";

filepath += filename.substring(filename.lastIndexOf('.'));

} else

filepath += "/text.html";

File file = new File(filepath);

file.createNewFile();

DataOutputStream fout = new DataOutputStream(new FileOutputStream(file));

byte buf[] = new byte[8000];

long rlen = 0;

while (true) {

int read = 0;

read = cin.read(buf);

rlen += read;

fout.write(buf, 0, read);

if (rlen == flength)

break;

}

fout.flush();

fout.close();

}

}

}