

SOC-第一次作业

-1140310606 张茗帅

一、源码学习&错误修改

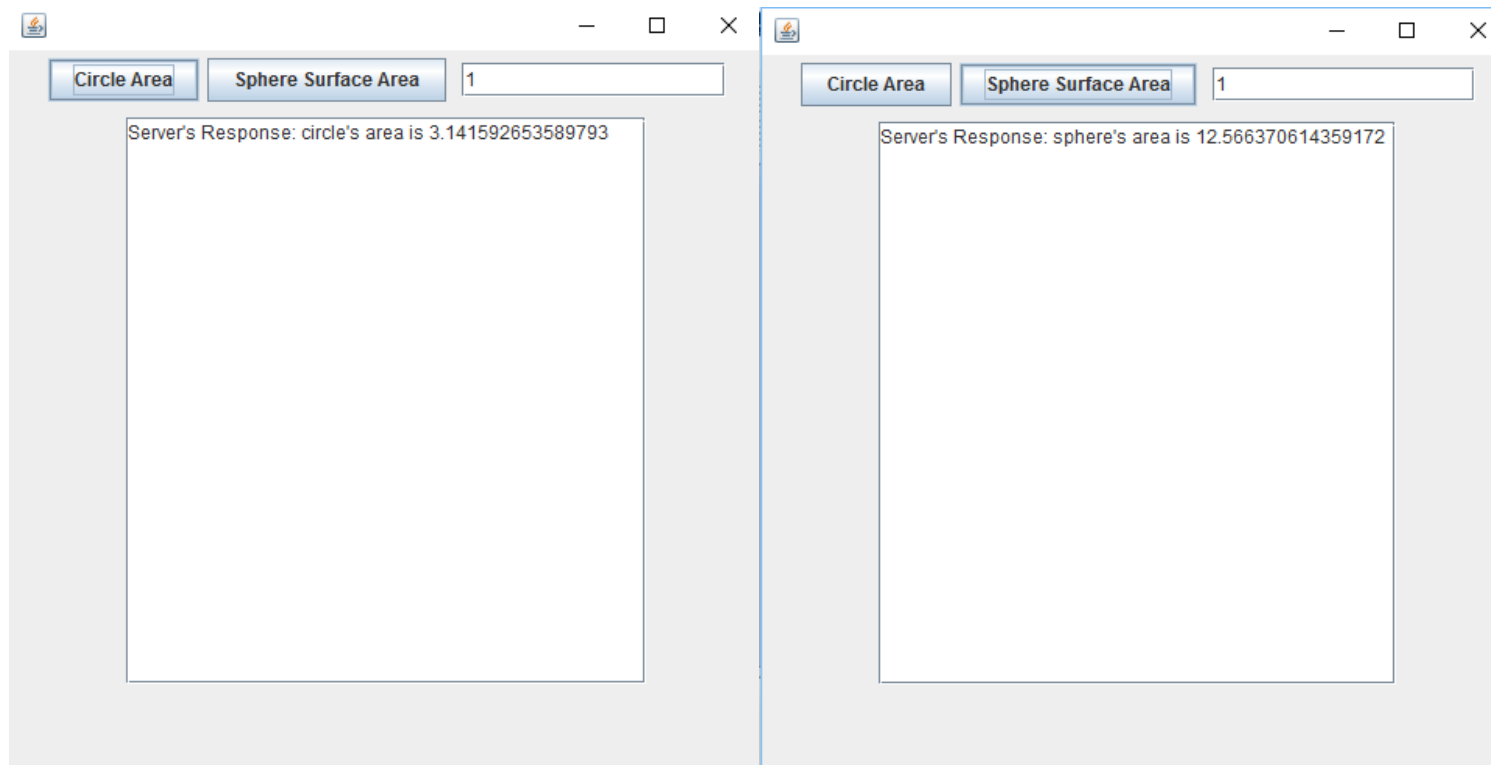
错误改正：

老师故意将 XML-RPC 端的注册类的别名和 Client 端调用类时所用的名字弄成不一致，而且第一个方法调用 circle 还多打了个数字 1 变成了 circ1le。这些错误被我发现了并加以改正。如下图：

```
if (type.trim().equals("circle"))
    result = (Double) client.execute("areaHandler.circleArea", new Object[] { radius });
else if (type.trim().equals("sphere"))
    result = (Double) client.execute("sphereAreaHandler.sphereArea", new Object[] { radius });

phm.addHandler("areaHandler", AreaHandler.class);
phm.addHandler("sphereAreaHandler", SphereAreaHandler.class);
```

改正完毕后，程序得到正确执行：



XML-RPC 服务器端：

(1) 首先建立两个操作类 AreaHandler 和 SphereAreaHandler 用于在服务器端进行运算，在获取客户端传入的半径 r 之后分别计算圆的面积和球的表面积。

.....

(2) 通过 XML-RPC 使用这两个方法必须需要两个步骤：

- 1 该方法必须用 XML-RPC 程序包注册
- 2 某个服务器可以使这个程序包通过 HTTP 进行访问

因此，AreaServer 类可以完成这两个步骤

.....

(3) AreaServer:

首先实例化一个内置服务器 public WebServer web_server, 然后设置服务器的地址和端口号

```
this.web_server = new WebServer(10080,  
    InetAddress.getByAddress(new byte[] { 127, 0, 0, 1 }));
```

接下来通过 web_server 实例化一个 XmlRpcServer

然后建立一个 Handler 实例，通过它来注册 AreaHandler 类和

SphereAreaHandler 类，并设置了别名 areaHandler 和

sphereAreaHandler，从而客户端可以通过这两个别名在远程分别调用这两个方法。（如下图）

```
XmlRpcServer xmlRpcServer = web_server.getXmlRpcServer(); //get XMLRPC Server Instance  
PropertyHandlerMapping phm = new PropertyHandlerMapping(); //create Handler instance  
try  
{  
    //register the class of AreaHandler as area  
    phm.addHandler("areaHandler", AreaHandler.class);  
    phm.addHandler("sphereAreaHandler", SphereAreaHandler.class);  
}
```

然后启动服务器即可 web_server.start()

Client 客户端：

首先构造了一个 java 的图形化界面，用于输入半径 r 并进行相关操作和展示计算的结果

然后创建客户端实例，标识服务器（配置的 URL 指向相应的服务器，因此我们必须让这里的 URL 与服务器端设置的 URL 相同）

```
XmlRpcClient client = new XmlRpcClient();
XmlRpcClientConfigImpl config = new XmlRpcClientConfigImpl();
try
{
    config.setServerURL(new URL("http://127.0.0.1:10080"));
}
```

接下来，根据我们触发了不同的按钮来选择执行与其相对应的方法。触发 circle 按钮，client 的 execute 方法通过 areaHandler.circleArea 这个（别名.方法）调用远程服务器端的 AreaHandler().circleArea 这个方法，并且通过 Object 类型传入了一个参数 r ，将返回值保存在 result 中用于之后的显示。另一个按钮 sphere 也是同样的道理。

二、实战演练

遗留系统描述：

我设计的遗留系统是一个登入与注册程序，用户通过图形界面输入自己的账号和密码，然后与注册表中的账号信息和密码信息进行匹配，返回相应的匹配结果。同样用户也可以注册一个新的账户和密码，然后系统会将新的用户密码信息保存到注册表中。其中注册

表的内容在文件中存储。

服务化思路：

利用 XMI-RPC 技术将该遗留系统进行服务化。

首先服务器端用于进行计算。定义一个 FileData 类，该类中有两个方法，分别为 writeLog()和 readLog()来控制对文件的读取和写入。

其中 writeLog 的接收参数为 str1 和 str2，表示从用户端接收的“用户名”和“密码”这两个参数。返回值 0 和 1 分别表示写入失败和写入成功。

而 readLog 的接受参数为 id 和 password，同样也表示从用户端接收的“用户名”和“密码”这两个参数。用来验证这两个信息是否匹配且“用户名”是否在文件中存在。返回值 0 表示“用户名”未在文件中找到，返回值 1 表示“用户名”已经找到且“密码”和“用户名”匹配成功，返回值 2 表示“用户名已经找到”，但是“用户名”和“密码”不匹配，返回值-1 表示出现了异常，文件读取失败

然后我建立一个 FileServer 类，用于配置 XML-RPC 服务器，然后将 FileData.class 注册，设置别名为 fileHandler，用于客户端进调用执行。

最后编写客户端的 login.class，该类通过图形化界面接受用户输入的参数“用户名”，和“密码”，然后通过两个不同的按钮来进行对应

的操作。

点击“登入”按钮，则调用服务器端 FileData 类中的 readLog 方法（通过之前设置的别名 fileHandler.readLog 进行调用），然后客户端按照不同的返回值，来判断“用户名”和“密码”的匹配情况，之后在图形界面中给出相应的返回信息，如“登入成功！”、“密码与用户名不符合”、“该用户不存在”、“发生异常”。

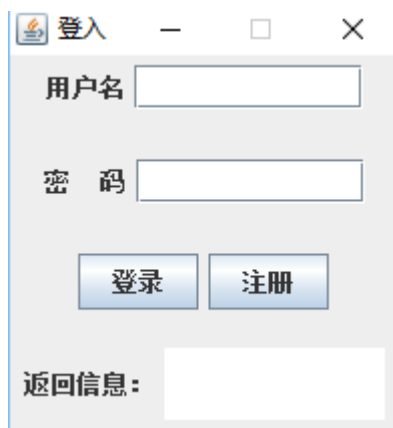
点击“注册”按钮，则调用服务器端 FileData 类中的 writeLog 方法（通过之前设置的别名 fileHandler.writeLog 进行调用），然后客户端按照不同的返回值，来判断“用户名”和“密码”的是否注册成功，之后在图形界面中给出相应的返回信息，如“注册成功！”、“发生异常”。

操作演示：

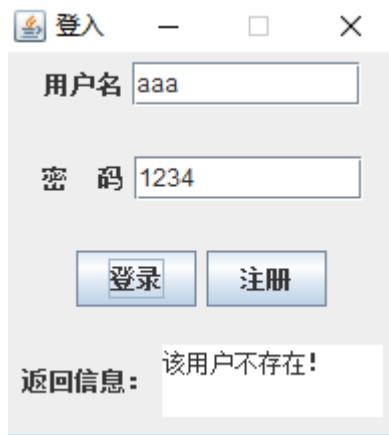
首先启动服务器



然后运行客户端，显示如下界面



输入 aaa 作为用户名，1234 作为密码，点击登入，因为此时是空表，则一定不存在



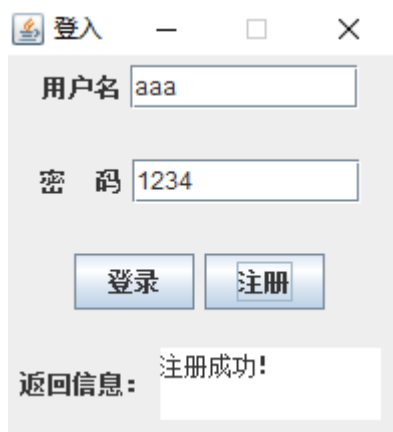
登入

用户名

密 码

返回信息: 该用户不存在!

进行注册，得到返回信息“注册成功！”



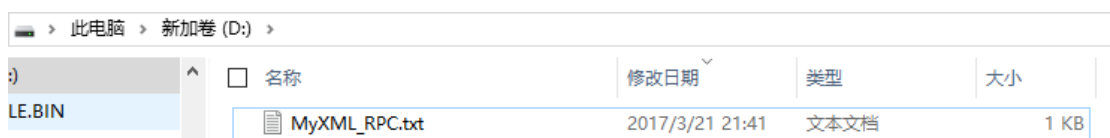
登入

用户名

密 码

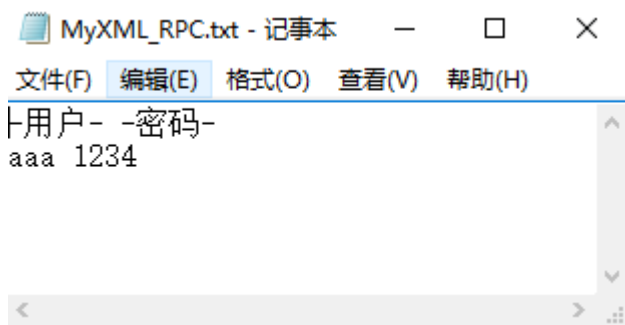
返回信息: 注册成功!

这时我们去 D 盘，发现生成了 MyXML_RPC.txt 这个文件



此电脑 > 新加卷 (D:) >			
名称	修改日期	类型	大小
MyXML_RPC.txt	2017/3/21 21:41	文本文档	1 KB

文件的内容包含我们刚才注册的用户名和密码这一项信息



MyXML_RPC.txt - 记事本

文件(F) 编辑(E) 格式(O) 查看(V) 帮助(H)

用户- 密码-
aaa 1234

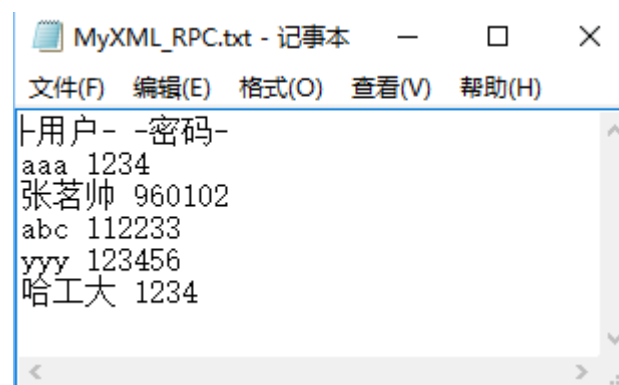
我们同样来测试一下中文

然后登入

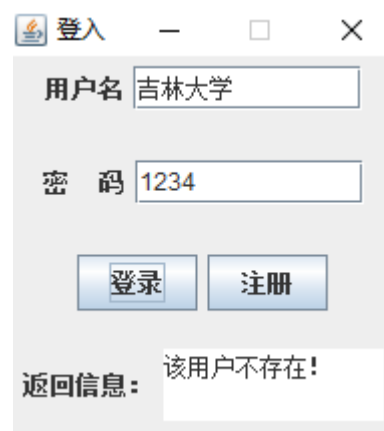
这时，文件并没有出现中文乱码（在设置文件的编码格式应该与 java 中相同为 utf-8）



接下来输入多组数据，用于接下来的测试



输入表中不存在的用户名“吉林大学”，返回信息为“该用户不存在！”



当用户名和密码不匹配式，同样能够得到正确的返回信息。



测试输入不合法的情况：有一项为空

不合法情况：输入含有空格

三、源码（含注释）

FileData.java:

```
package com.zhangmingshuai;
import java.io.*;
import java.text.SimpleDateFormat;
import java.util.*;

public class FileData{
    //写文件，支持中文字符
    public int writeLog(String str1,String str2)
    {
        try
        {
```

```

String path="D:/MyXML_RPC.txt";
File file=new File(path);
if(!file.exists()){ //当文件不存在时, 创建一个
    file.createNewFile();
    PrintStream ps = new PrintStream(new
FileOutputStream(file));
    ps.println("--用户--密码--");// 往文件里写入字符串
}
FileOutputStream out=new FileOutputStream(file,true); //采
用追加方式用true
StringBuffer sb=new StringBuffer();
sb.append(str1+" "+str2+"\r\n"); //windows中换行符为
\r\n,linux中换行符为\r,Mac换行符为\n
out.write(sb.toString().getBytes("utf-8")); //转换对应的字符集
utf-8,防止中文乱码。
out.close();
return 1; //返回值为1时表示写入成功
}
catch(IOException ex)
{
    System.out.println(ex.getStackTrace());
    return 0; //返回值为0时表示写入失败
}
}
//读文件, 支持中文字符
public int readLog(String id,String password)
{
    StringBuffer sb=new StringBuffer();
    String tempstr=null;
    String[] a = null; //字符串数组用于保存从文件中读出的内容
    try
    {
        String path="D:/MyXML_RPC.txt";
        File file=new File(path);
        if(!file.exists()){
            file.createNewFile();
            PrintStream ps = new PrintStream(new
FileOutputStream(file));
            ps.println("-用户- -密码-");// 往文件里写入字符串
        }
        FileInputStream fis=new FileInputStream(file); //文件输入
流
        BufferedReader br=new BufferedReader(new
InputStreamReader(fis,"utf-8")); //设置编码规则, 放置中文乱码

```

```

        while ((tempstr=br.readLine())!=null) {
            sb.append(tempstr);
            // System.out.println(tempstr);
            sb.append(" "); //加入空格为了今后的分割
        }
        a = sb.toString().split("\\s"); //将缓冲区的字符串以空格进行
分割保存至a中
        if (a.length == 0) {
            return 0; //返回值为0的时候代表匹配失败，即用户未存在于注册
表中
        }
        for(int i=0;i<a.length-1;i=i+2){ //文件的格式为：用户名 密码
\n,因此以i=i+2来匹配用户名
            //System.out.println(a[i]);
            if(a[i].equals(id)&&(a[i+1].equals(password))){
                return 1; //返回值为1代表匹配成功，即找到了该用户且密码
输入正确
            }
            else
if(a[i].equals(id)&&(!a[i+1].equals(password))){
                return 2; //返回值时代表找到该用户，但是密码不正确
            }
        }
        return 0; //未找到用户，即此用户未注册
    }
    catch(IOException ex)
    {
        System.out.println(ex.getStackTrace());
        return -1; //意外情况时返回-1
    }
}

/* public static void main(String[] args) {
    // TODO Auto-generated method stub
    writeLog("this","hi");
    writeLog("啊啊","好的");
    int a;
    a = readLog("啊啊","好");
    System.out.println(a);
}
*/
}

```

.....

FileServer.java:

```
package com.zhangmingshuai;
import java.io.IOException;
import java.net.InetAddress;

import org.apache.xmlrpc.XmlRpcHandler;
import org.apache.xmlrpc.server.PropertyHandlerMapping;
import org.apache.xmlrpc.server.XmlRpcServer;
import org.apache.xmlrpc.webserver.WebServer;

public class FileServer {

    public WebServer    web_server; //实例化一个内置服务器

    public FileServer()
    {
        try
        {
            this.web_server = new WebServer(10080, //配置服务器的IP地址
和端口号
                InetAddress.getByAddress(new byte[] { 127, 0, 0,
1 }));
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }

    public void initServer()
    {
        XmlRpcServer xmlRpcServer = web_server.getXmlRpcServer();
//获取一个XML-RPC的实例
        PropertyHandlerMapping phm = new PropertyHandlerMapping();
//建立Handler实例
        try
        {
            //注册FileData类, 设置别名为fileHandler
            phm.addHandler("fileHandler", FileData.class);
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
    }
}
```

```

        e.printStackTrace();
    }
    xmlRpcServer.setHandlerMapping(phm);
    try
    {
        // 启动服务器
        System.out.println("Attempting to start XML-RPC
Server...");
        web_server.start();
        System.out.println("Server is ready, waiting for client
calling...");
    }
    catch (Exception e)
    {
        e.printStackTrace();
    }
}

public static void main(String[] args)
{
    // TODO Auto-generated method stub
    FileServer mySer = new FileServer();
    mySer.initServer();
}
}

```

login.java

```

package com.zhangmingshuai;

import java.awt.BorderLayout;
import java.awt.GridLayout;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.net.URL;

import javax.swing.JButton;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JPanel;
import javax.swing.JScrollPane;
import javax.swing.JTextArea;

```

```

import javax.swing.JTextField;

import org.apache.xmlrpc.client.XmlRpcClient;
import org.apache.xmlrpc.client.XmlRpcClientConfigImpl;

public class login extends JFrame{
    //定义组件
    static JPanel jp1, jp2, jp3, jp4;
    static JLabel jl1, jl2, jl3;
    static JTextField jtf1, jtf2;
    static JButton jb1, jb2;
    static JTextArea ta;
    static JScrollPane js;

    //构造函数 初始化组件
    public login(){
        jp1 = new JPanel();
        jp2 = new JPanel();
        jp3 = new JPanel();
        jp4 = new JPanel();

        jl1 = new JLabel("用户名");
        jl2 = new JLabel("密    码");
        jl3 = new JLabel("返回信息：");

        jtf1 = new JTextField(10);
        jtf2 = new JTextField(10);

        jb1 = new JButton("登录");
        jb2 = new JButton("注册");

        ta = new JTextArea(2,10); //这个组件用于显示返回信息
        //js = new JScrollPane(ta);

        jp1.add(jl1);
        jp1.add(jtf1);
        jp2.add(jl2);
        jp2.add(jtf2);
        jp3.add(jb1);
        jp3.add(jb2);
        jp4.add(jl3);
        jp4.add(ta);

        this.add(jp1);
    }
}

```

```

        this.add(jp2);
        this.add(jp3);
        this.add(jp4);

        this.setLayout(new GridLayout(4, 1));
        this.setTitle("登入");
        this.setSize(200,220);
        this.setLocation(100,200);
        this.setResizable(false);
        this.setVisible(true);
        this.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        ta.setEditable(false); //设置输出信息为不可编辑状态
    }

    public String invoke(String id,String password, String type)
    {
        int result = 0;
        XmlRpcClient client = new XmlRpcClient(); //实例化XML-RPC客
        户端
        XmlRpcClientConfigImpl config = new
        XmlRpcClientConfigImpl();
        try
        {
            config.setServerURL(new URL("http://127.0.0.1:10080"));
            //设置端口, 和对应服务器的端口向对应
        }
        catch (Exception e)
        {
            e.printStackTrace();
        }
        client.setConfig(config); //配置client服务器
        String out = null;
        try
        {
            if (type.trim().equals("login")){ //触发“登入”按钮, 调用
            readLog方法, 传入两个参数id和password
                result =
                (Integer)client.execute("fileHandler.readLog", new Object[]
                { id,password });
                if(result == 1){ //根据不同的返回值, 输入不同的信息
                    out = "登入成功!";
                }
                else if(result == 2){

```

```

        out = "密码与用户名不符合！";
    }
    else if(result == 0){
        out = "该用户不存在！";
    }
    else{
        out = "发生异常！";
    }
}
else if (type.trim().equals("register")){ //触发“登入”按钮，调用readLog方法，传入两个参数id和password
    result = (Integer)
client.execute("fileHandler.writeLog", new Object[]
{ id,password });
    if(result == 1){ //根据不同的返回值，输入不同的信息
        out = "注册成功！";
    }
    else{
        out = "发生异常！";
    }
}
}
catch (Exception e)
{
    e.printStackTrace();
}
return out;
}

```

```

public static void main(String[] args) {

    final login mylogin = new login();

    jb1.addActionListener(new ActionListener(){ //判断哪个按钮被触
发
        public void actionPerformed(ActionEvent e) {
            // TODO Auto-generated method stub
            String id = jtf1.getText();
            String password = jtf2.getText();
            if (id.equals("")||id.contains("
")||password.equals("")||password.contains(" ")) { //这个条件表示用户

```


输入为空或者含有空格

```
        ta.setText("输入不合法（含空格或为空）");
    }else{
        String result = mylogin.invoke(id,password,
"login");
        ta.setText(result);
    }
}
});

jb2.addActionListener(new ActionListener() {    //判断哪个按钮
被除法
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        String id = jtf1.getText();
        String password = jtf2.getText();
        if (id.equals("")||id.contains("
")||password.equals("")||password.contains(" ")){//这个条件表示用户
输入为空或者含有空格
            ta.setText("输入不合法（含空格或为空）");
        }else{
            String result =
mylogin.invoke(id,password,"register");
            ta.setText(result);
        }
    }
});
}
```

四、补充说明

我提交的作业以.rar 格式进行存储，其中包含此报告，和两个文件夹->MyXMLRPC_Server 和 MyXMLRPC_Client 这两个工程文件，直接导入至 myeclipse 即可以使用，不过其中的 jar 包需要自己导入。