|  |
| --- |
| **M S RAMAIAH INSTITUTE OF TECHNOLOGY**  **(Autonomous Institute Affiliated to VTU)**  **Department of Information Science and Engineering** |
| https://lh6.googleusercontent.com/LTywIt5pr2MnOeBx2D2TbPglev-Ts7H3ccZWiEZ14HZf0EWB1OjCTfj3smESo_ZXeSRctXNpIYBAOHF4HknBsQWZDdEARKrQIrQRXUOZAUGW_17HykkZ-bcnE9XFS_7ueDz8qC67  Jan-May 2018 |
| A Project on |
| **“Email Client”** |
| *Submitted in partial fulfillment of the CIE for the subject*  *Java and J2EE* |
| by |
| **Chandan Singh**  **(1MS15IS029)** |
| **Shubham Sharda**  **(1MS15IS149)** |
|  |
| *Under the guidance*  *of* |
| **Koushik S** |
| Assistant Professor |
| Department of ISE, MSRIT |
| Bengaluru-560054 |

**ABOUT THE PROJECT (PROBLEM STATEMENT)**

We all have experienced sending emails using web services such as Gmail, yahoo mail, hotmail, etc. Here in this project we have developed a email client platform where one can send and read emails through a swing based Java Desktop Application.

The further extension of this project can be used to implement other functions such as adding attachments, Sending replies, and many more.

This project on execution also reads some of the Sender’s Inbox email’s subjects and displays it.

|  |  |  |
| --- | --- | --- |
|  |  |  |

**IMPLEMENTATION DETAILS AND CODE SNIPPETS**

1. We’ve used Eclipse IDE.
2. We have used SMTP and IMAP Protocols for Sending and receiving mails respectively.
3. API used in this project: JavaMail API.
4. Strategy Pattern has been implemented on the Swing based model of the project.

**Methods and classes**

**Client.java-**

This is the main class, and it creates the objects of the class First.java and calls them.

package package1;

public class Client {

public static void main(String[] args) {

First frst = new First();

frst.RightPanel();

}}

**First.java**

Implementation of front end has been done in this class.

package package1;

import java.awt.BorderLayout;

import java.awt.GridLayout;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.util.List;

import javax.swing.\*;

import javax.swing.border.TitledBorder;

public class First extends JPanel implements ActionListener

{

/\*\*

\*

\*/

private static final long serialVersionUID = 6672594319721820776L;

TitledBorder brd = new TitledBorder("Inbox");

TitledBorder brd1 = new TitledBorder("Compose");

GmailClient newGmailClient;

protected String emailfrm;

protected String emailto;

protected String pass;

protected String subj;

protected String message;

JFrame jfrm;

JPanel lp;

JPanel rp;

JPanel irp;

JPanel Inboxirp;

JButton btnCmp;

JButton btnSend;

JButton btnInbox;

JMenuBar mnbr;

JLabel displaymsg;

JLabel frm;

JLabel to;

JLabel pw;

JLabel sub;

JLabel msg;

JTextArea afrm;

JTextArea ato;

JPasswordField apw;

JTextArea asub;

JTextArea amsg;

public First()

{

jfrm = new JFrame("Email-Xilla");

mnbr = new JMenuBar();

JMenu file = new JMenu("File");

JMenu edit = new JMenu("Edit");

JMenu help = new JMenu("Help");

mnbr.add(file);

mnbr.add(edit);

mnbr.add(help);

JSplitPane sp = new JSplitPane();

lp = new JPanel(null);

rp = new JPanel(null);

sp.setLeftComponent(lp);

sp.setRightComponent(rp);

sp.setDividerLocation(200);

btnCmp = new JButton("Compose");

btnCmp.addActionListener(this);

btnSend = new JButton("Send");

btnSend.addActionListener(this);

btnInbox = new JButton("Inbox");

btnInbox.addActionListener(this);

lp.add(btnCmp);

lp.add(btnSend);

lp.add(btnInbox);

btnCmp.setBounds(10,10,100,30);

btnSend.setBounds(10,60,100,30);

btnInbox.setBounds(10,120,100,30);

jfrm.add(sp);

jfrm.setJMenuBar(mnbr);

jfrm.setSize(1920,1080);

jfrm.setVisible(true);

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

}

public void RightPanel()

{

irp = new JPanel(null);

rp.add(irp,BorderLayout.CENTER);

brd1.setTitleJustification(TitledBorder.CENTER);

brd1.setTitlePosition(TitledBorder.TOP);

irp.setBorder(brd1);

//irp.setBackground(Color.RED);

irp.setBounds(10,10,800,700);

irp.setVisible(false);

frm = new JLabel("From:");

frm.setBounds(15,15,100,50);

to = new JLabel("To:");

to.setBounds(15,75,100,50);

pw = new JLabel("Password");

pw.setBounds(15,135,100,50);

sub = new JLabel("Subject");

sub.setBounds(15,195,100,50);

msg = new JLabel("Message");

msg.setBounds(15,255,100,50);

afrm = new JTextArea();

afrm.setBounds(150,15,200,25);

ato =new JTextArea();

ato.setBounds(150,75,200,25);

apw= new JPasswordField();

apw.setBounds(150,135,200,25);

asub = new JTextArea();

asub.setBounds(150,195,200,25);

amsg = new JTextArea();

amsg.setBounds(150,255,500,200);

irp.add(frm);

irp.add(to);

irp.add(pw);

irp.add(sub);

irp.add(msg);

irp.add(afrm);

irp.add(ato);

irp.add(apw);

irp.add(asub);

irp.add(amsg);

brd.setTitleJustification(TitledBorder.CENTER);

brd.setTitlePosition(TitledBorder.TOP);

Inboxirp = new JPanel();

Inboxirp.setBorder(brd);

rp.add(Inboxirp,BorderLayout.EAST);

//Inboxirp.setBackground(Color.RED);

Inboxirp.setBounds(850,10,850,700);

Inboxirp.setVisible(false);

}

public void actionPerformed(ActionEvent e) {

if(e.getSource()==btnCmp)

{

irp.setVisible(true);

}

if(e.getSource()==btnSend)

{

try {

newGmailClient = new SRMail();

emailfrm = afrm.getText();

emailto = ato.getText();

pass = new String(apw.getPassword());

subj = asub.getText();

message = amsg.getText();

newGmailClient.setAccountDetails(emailfrm,pass);

newGmailClient.SendGmail(emailfrm,emailto,subj,message );

}

catch(Exception er)

{

er.printStackTrace();

JOptionPane.showMessageDialog(null, "Sending email to: " + to + " failed !!!", "Falied to Send!!!", JOptionPane.ERROR\_MESSAGE);

}

}

if(e.getSource()==btnInbox)

{

displaymsg= new JLabel();

Inboxirp.setVisible(true);

Inboxirp.setLayout(new GridLayout(20,1));

newGmailClient = new SRMail();

emailfrm = afrm.getText();

pass = new String(apw.getPassword());

@SuppressWarnings("rawtypes")

List messagedis= newGmailClient.ReadGmail(emailfrm,pass);

/\*displaymsg= new JLabel();

String element = (String) messagedis.get(0);

displaymsg.setText(element);

Inboxirp.add(displaymsg);\*/

displaymsg= new JLabel();

String element1 = (String) messagedis.get(1);

displaymsg.setText(element1);

Inboxirp.add(displaymsg);

displaymsg= new JLabel();

String element3 = (String) messagedis.get(2);

displaymsg.setText(element3);

Inboxirp.add(displaymsg);

displaymsg= new JLabel();

String element4 = (String) messagedis.get(3);

displaymsg.setText(element4);

Inboxirp.add(displaymsg);

displaymsg= new JLabel();

String element5 = (String) messagedis.get(4);

displaymsg.setText(element5);

Inboxirp.add(displaymsg);

displaymsg= new JLabel();

String element6 = (String) messagedis.get(5);

displaymsg.setText(element6);

Inboxirp.add(displaymsg);

displaymsg= new JLabel();

String element7 = (String) messagedis.get(6);

displaymsg.setText(element7);

Inboxirp.add(displaymsg);

displaymsg= new JLabel();

String element8 = (String) messagedis.get(7);

displaymsg.setText(element8);

Inboxirp.add(displaymsg);

displaymsg= new JLabel();

String element9 = (String) messagedis.get(8);

displaymsg.setText(element9);

Inboxirp.add(displaymsg);

displaymsg= new JLabel();

String element10 = (String) messagedis.get(9);

displaymsg.setText(element10);

Inboxirp.add(displaymsg);

displaymsg= new JLabel();

String element11 = (String) messagedis.get(10);

displaymsg.setText(element11);

Inboxirp.add(displaymsg);

}

}

}

**GmailClient.java**

We have made GmailClient an abstract class, which will hold all the variables that are required to send and receive mail. Ex- Username, password,text etc. We have created two abstract methods which will be overridden later by the SRMail Class.

package package1;

import java.util.List;

public abstract class GmailClient {

protected String userName;

protected String password;

protected String sendingHost;

protected int sendingPort;

protected String from;

protected String to;

protected String subject;

protected String text;

protected String receivingHost;

// private int receivingPort;

public void setAccountDetails(String userName,String password){

this.userName=userName;//sender's email can also use as User Name

this.password=password;

}

public abstract void SendGmail(String mailFrom, String mailTo, String mailSubject, String mailText);

@SuppressWarnings("rawtypes")

public abstract List ReadGmail(String from , String password);

}

**SRMail.java**

We have made two functions in this class, Send mail and read mail, using SMTP and IMAP Protocols respectively.This class extends the GmailClient Class.

package package1;

import java.util.\*;

import javax.mail.Folder;

import javax.mail.Message;

import javax.mail.MessagingException;

import javax.mail.Session;

import javax.mail.Store;

import javax.mail.Transport;

import javax.mail.Message.RecipientType;

import javax.mail.internet.AddressException;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeMessage;

import javax.swing.JOptionPane;

public class SRMail extends GmailClient {

@SuppressWarnings("rawtypes")

List messagelist = new ArrayList();

@Override

public void SendGmail(String from, String to, String subject, String text) {

this.from=from;

this.to=to;

this.subject=subject;

this.text=text;

// For a Gmail account--sending mails-- host and port shold be as follows

this.sendingHost="smtp.gmail.com";

this.sendingPort=465;

Properties props = new Properties();

props.put("mail.smtp.host", this.sendingHost);

props.put("mail.smtp.port", String.valueOf(this.sendingPort));

props.put("mail.smtp.user", this.userName);

props.put("mail.smtp.password", this.password);

props.put("mail.smtp.auth", "true");

Session session1 = Session.getDefaultInstance(props);

Message simpleMessage = new MimeMessage(session1);

//MIME stands for Multipurpose Internet Mail Extensions

InternetAddress fromAddress = null;

InternetAddress toAddress = null;

try {

fromAddress = new InternetAddress(this.from);

toAddress = new InternetAddress(this.to);

} catch (AddressException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(null, "Sending email to: " + to + " failed !!!", "Falied to Send!!!", JOptionPane.ERROR\_MESSAGE);

}

try {

simpleMessage.setFrom(fromAddress);

simpleMessage.setRecipient(RecipientType.TO, toAddress);

// to add CC or BCC use

// simpleMessage.setRecipient(RecipientType.CC, new InternetAddress("CC\_Recipient@any\_mail.com"));

// simpleMessage.setRecipient(RecipientType.BCC, new InternetAddress("CBC\_Recipient@any\_mail.com"));

simpleMessage.setSubject(this.subject);

simpleMessage.setText(this.text);

//sometimes Transport.send(simpleMessage); is used, but for gmail it's different

Transport transport = session1.getTransport("smtps");

transport.connect (this.sendingHost,sendingPort, this.userName, this.password);

transport.sendMessage(simpleMessage, simpleMessage.getAllRecipients());

transport.close();

JOptionPane.showMessageDialog(null, "Mail sent successfully ...","Mail sent",JOptionPane.PLAIN\_MESSAGE);

} catch (MessagingException e) {

e.printStackTrace();

JOptionPane.showMessageDialog(null, "Sending email to: " + to + " failed !!!", "Falied to Send!!!", JOptionPane.ERROR\_MESSAGE);

}

}

@SuppressWarnings({ "unchecked", "rawtypes" })

public List ReadGmail(String from ,String password) {

this.from=from;

this.password=password;

this.receivingHost="imap.gmail.com";//for imap protocol

Properties props2=System.getProperties();

props2.setProperty("mail.store.protocol", "imaps");

// I used imaps protocol here

Session session2=Session.getDefaultInstance(props2, null);

try {

Store store=session2.getStore("imaps");

store.connect(this.receivingHost,this.from, this.password);

Folder folder=store.getFolder("INBOX");//get inbox

folder.open(Folder.READ\_ONLY);//open folder only to read

Message message[]=folder.getMessages();

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

//print subjects of all mails in the inbox

System.out.println(message[i].getSubject());

messagelist.add(message[i].getSubject());

/\*String element = (String) messagelist.get(i);

System.out.println(element);\*/

}

//close connections

folder.close(true);

store.close();

return messagelist;

} catch (Exception e) {

System.out.println(e.toString());

}

return null;}}

**Output Screenshots**







