**武汉大学国际软件学院**

**实验报告**

**课程名称 网络及分布式计算实验**

**专业年级 2014级软件工程**

**姓 名**

**学 号**

**实验学期 2016-2017　 学年 第1 学期**

**课堂时数 课外时数**

**填写时间 2017 年 5 月 12 日**

**实验任务二**

邮局协议(Post Office Protocol Version 3, POP3)提供了一种对邮件消息进行排队的标准机制，这样接收者以后才能检索邮件。POP3服务器也运行在TCP/IP之上，并且在默认端口110上监听（SMTP协议运行在25端口上）。现在，大部分邮件服务器都采用SMTP发送邮件，同时使用POP3接收电子邮件信息。

**解法一**

1. 握手

客户端向服务器端发送建立一个TCP连接请求。

例如，在建立连接后，某个服务器发回的响应应是这样的信息：

+OK Hello there

这是服务器发回的欢迎信息，+OK代表正确的状态。如果此时接收的响应行开头为-ERR,说明服务器此时为某些问题困扰。

2.认证

客户端向服务器标明身份的主要命令是USER命令和PASS命令。

USER <username>

PASS <password>

服务器对USER命令可能的响应是：

+OK　name is a valid mailbox

-ERR never heard of mailbox name

例如：

C: USER mrose

S: +OK mrose is a real hoopy frood

...

C: USER frated

S: -ERR sorry, no mailbox for frated here

服务器对PASS命令可能的响应是：

+OK maildrop locked and ready

-ERR invalid password

-ERR unable to lock maildrop

例如：

C: USER mrose

S: +OK mrose is a real hoopy frood

C: PASS secret

S: +OK mrose's maildrop has 2 messages (320 octets)

...

C: USER mrose

S: +OK mrose is a real hoopy frood

C: PASS secret

S: -ERR maildrop already locked

客户端首先使用USER命令向服务器发送用户名，服务器接收到命令后从中取出用户名，在数据库中核对，如果没有这个用户名存在，就发送-ERR给客户端。认证失败后，客户端不能继续进入事务状态，只能以QUIT命令结束会话。如果用户名存在，那么发送+OK给客户端，提示客户端继续使用PASS命令提供密码。如果密码和用户名配对成功，那么发送+OK到客户端。会话进入事务阶段。

3.接收邮件

邮件服务器打开用户的邮箱后，它赋给每一个邮件（也称消息）一个十进制编码。第一个消息为1号消息，第二个消息为2号消息，以此类推。

接收邮件的命令是RETR，这个命令的格式是RETR msg。

参数msg是消息的编号，msg是必需的。这个命令只能在会话处于事务状态时才能使用。使用这个命令服务器的可能应答有：

-ERR no such message

当msg指定的消息不存在时，服务器发回-ERR的响应。

+OK message follows

消息存在，服务器发回+OK响应。在发送状态信息行后，接着发送包含消息数据的信息。例如：

C: RETR 1

S: +OK 120 octets

S: <the POP3 server sends the entire message here>

S: .

这是一个典型的返回多行响应信息的命令。上面的信息显示客户端发送命令RETR 1要求服务器发回1号信息。服务器收到请求后，确认1号信息存在，首先发回响应+OK 120 octets, 120 octets表示这个消息长度为120字节。由于多行响应以CRLF.CRLF结束，所以客户端也可以不理会长度信息。

4.其他操作

（1）STAT

这个命令没有参数，返回当前对话对应的信箱中的信息个数和这些消息总的字节数。对于这个命令，可能的响应如：

+OK nn mm

其中，nn为消息个数，mm为消息的总字节数。

（2）LIST

LIST [msg]

这个命令的作用和STAT命令类似，不同的是，LIST命令作用于邮件，而STAT命令作用于整个信箱。

LIST命令可能的响应有如下两种：

+OK scan listing follows

-ERR no such message

下面的例子显示了如何使用LIST命令：

C: LIST

S: +OK ２ messages (320 octets)

S: 1 120

S: 2 200

S: .

...

C: LIST 2

S: +OK 2 200

...

C: LIST 3

S: -ERR no such message, only 2 messages in maildrop

LIST命令后的参数可选，如果没有指明消息，那么LIST命令默认作用于每一个消息。这是返回多行的响应。响应中的+OK 2 200代表第二个消息，消息长度200字节。

（3）DELE

DELE msg

参数msg为要删除的消息编号，执行这个命令后可能的响应为：

+OK message deleted

-ERR no such message

例如：

C: DELE 1

S: +OK message 1 deleted

...

C: DELE 2

S: -ERR message 2 already deleted

5.断开连接

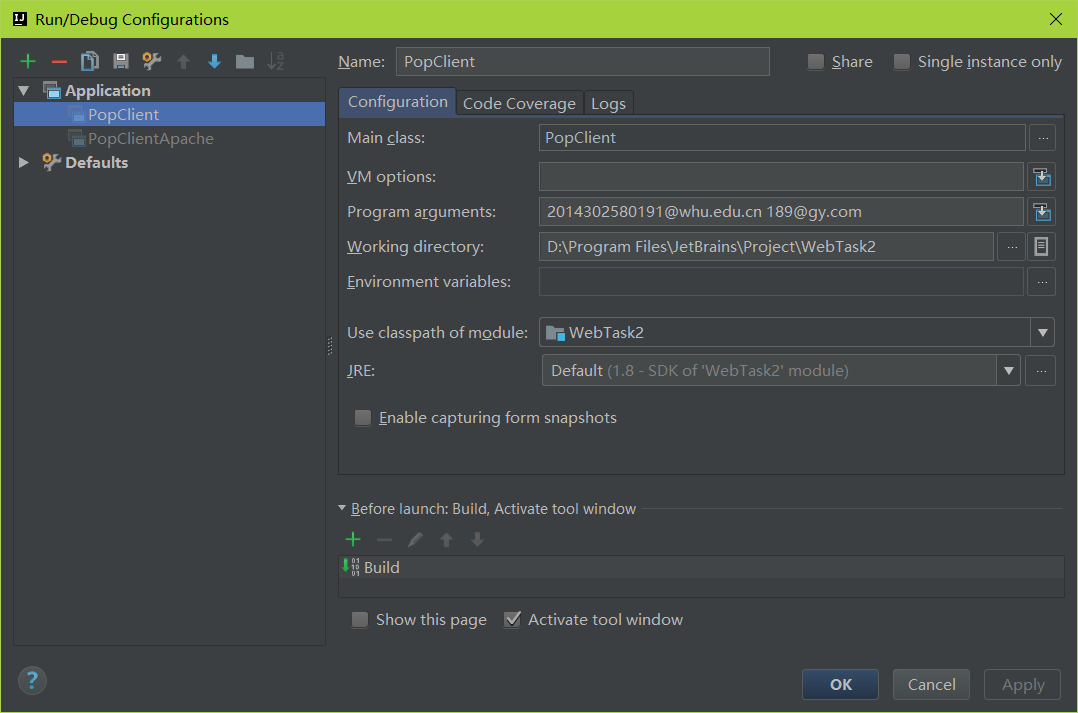
QUIT命令可能的响应只有+OK，例如：

C: QUIT

S: +OK dewey POP3 server signing off

import java.io.BufferedReader;  
import java.io.IOException;  
import java.io.InputStreamReader;  
import java.io.PrintStream;  
import java.net.Socket;  
  
*/\*\*  
 \* Created by BG on 2017/4/24.  
 \*/*public class PopClient {  
  
 private static BufferedReader *in*; // 从套接字中读取文本  
 private static PrintStream *out*; // 将文本写入套接字  
  
 public static void main(java.lang.String[] args) {  
 // 用户名  
 String username = "";  
 // 密码  
 String password = "";  
 // Pop服务器名，这里我用的是武汉大学电子邮箱的服务器  
 String popServer = "pop.whu.edu.cn";  
 // 设置标志位  
 boolean delete = false;  
 boolean secure = false;  
 Socket socket;  
 String line;  
 String numberMessages;  
 try {  
 if (args.length == 2) {  
 // 从命令行中取用户名和密码  
 username = args[0];  
 password = args[1];  
 } else  
 System.*out*.println("Usage:\r\n: java PopExample <PopServer> <username> <password>");  
 System.*out*.println("Connecting to " + popServer + " ...");  
 // Pop3服务默认使用110端口  
 // 建立TCP连接  
 socket = new Socket(popServer,110);  
 System.*out*.println("Connected\nAuthentication in progress...");  
 // 连接建立成功，获得关联的输入流和输出流  
 *in* = new BufferedReader(new InputStreamReader(socket.getInputStream()));  
 *out* = new PrintStream(socket.getOutputStream());  
 // 这一行响应是服务器的欢迎信息，可省略  
 line = *readFromInputStream*();  
 // 发送用户名  
 *printToOutputStream*("USER " + username);  
 // 如果用户存在，则响应信息开头有+OK字样。否则退出  
 if (!*readFromInputStream*().startsWith("+OK")) {  
 System.*out*.println("Wrong user name， disconnecting");  
 socket.close();  
 System.*exit*(1);  
 }  
 // 用户名存在，则发送密码  
 *out*.println("PASS " + password);  
 System.*out*.println("R: PASS \*\*\*\*\*\*\*\*");  
 // 读服务器响应信息  
 line = *readFromInputStream*();  
 // 如果密码核对成功，响应信息中开头+OK字样  
 if (!line.startsWith("+OK")) {  
 // 密码不正确，无法继续  
 System.*out*.println("Wrong password， disconnecting");  
 socket.close();  
 System.*exit*(1);  
 }  
 // 命令STAT  
 *printToOutputStream*("STAT");  
 // 读取响应  
 line = *readFromInputStream*();  
 if (!line.startsWith("+OK")) {  
 //如果命令执行失败，则退出  
 System.*out*.println("Error:" + line);  
 socket.close();  
 System.*exit*(1);  
 }  
 // 从统计信息中抽取消息个数  
 int i = line.lastIndexOf(' ');  
 numberMessages = line.substring(4, i);  
 System.*out*.println("You have " + numberMessages + " message(s) in your mailbox");  
 // 获取消息个数  
 int n = Integer.*parseInt*(numberMessages);  
 int numberBytes;  
 // 取出服务器上的每一个消息  
 for (int msg = 1; msg <= n; msg++) {  
 System.*out*.println("Retreaving message " + msg);  
 // 命令RETR 用于收取消息  
 *printToOutputStream*("RETR " + msg);  
 // 从服务器读取消息  
 line = *readFromInputStream*();  
 if (!line.startsWith("+OK")) {  
 System.*out*.println("Error: " + line);  
 socket.close();  
 System.*exit*(1);  
 }  
 // 消息以“.”号结束，一行中连续两个"."代表"."  
 line = *in*.readLine();  
 while (line.compareTo(".") != 0) {  
 if (line.compareTo("..") == 0)  
 System.*out*.println(".");  
 else  
 System.*out*.println(line);  
 // 读取下一行  
 line = *in*.readLine();  
 }  
 // 消息已读取到本地，从服务器删除消息  
 // 删除消息的命令是DELE [messagenumber]  
 *printToOutputStream*("DELE " + msg);  
 *readFromInputStream*();  
 }  
 // 准备退出，使用QUIT命令  
 *printToOutputStream*("QUIT");  
 *readFromInputStream*();  
 // 关闭套接字  
 socket.close();  
 } catch (IOException e) {  
 e.printStackTrace(System.*err*);  
 }  
 }  
 // 方法printToOutputStream()向SMTP服务器发送命令  
 private static void printToOutputStream(String s) throws IOException {  
 System.*out*.println("S : " + s);  
 *out*.println(s);  
 return;  
 }  
 // 方法readFromInputStream()接收从SMTP服务器发回的响应信息  
 private static String readFromInputStream() throws IOException {  
 String s = *in*.readLine();  
 if (s != null)  
 System.*out*.println("R :" + s);  
 return s;  
 }  
  
}

这里可以使用命令行参数来设置邮箱和密码，我使用的是武汉大学电子邮箱的服务器。



**解法二**

我在查找网络资料对解法一进行编码的时候，看到了apache的一个包的API，发现使用这个包可以更方便快捷的实现这个客户端，所以我使用这个包实现了一下，完成了最基本的功能。

添加包依赖：[commons-net-3.6](http://mirror.bit.edu.cn/apache/commons/net/binaries/commons-net-3.6-bin.tar.gz)

以下是代码：

import org.apache.commons.net.pop3.POP3Client;  
import org.apache.commons.net.pop3.POP3MessageInfo;  
  
import java.io.BufferedReader;  
import java.io.Reader;  
  
*/\*\*  
 \* Created by BG on 2017/4/25.  
 \*/*public class PopClientApache {  
  
 public static void main(String[] args) {  
 POP3Client pop3 = new POP3Client();  
 try {  
  
 pop3.setDefaultPort(110);  
 pop3.connect("pop.whu.edu.cn");  
 // 我测试的是QQ邮件～  
 // 输入你的QQ号作为油箱名称 QQ密码作为邮箱密码  
 if (pop3.login("2014302580191@whu.edu.cn", "189@gy.com")) {  
  
 POP3MessageInfo[] p3m = pop3.listMessages();  
  
 System.*out*.println("您一共有" + p3m.length + "封信件");  
 for (POP3MessageInfo obj : p3m) {  
 int id = obj.number;// 获得信件在服务器端的唯一编码  
  
 Reader red = pop3.retrieveMessage(id);  
 BufferedReader br = new BufferedReader(red);  
 while (br.readLine() != null) {  
 System.*out*.println(br.readLine());  
 }  
 }  
 }  
  
 pop3.logout();  
 pop3.disconnect();  
  
 } catch (Exception e) {  
 // *TODO Auto-generated catch block* System.*out*.println("连接失败");  
 e.printStackTrace();  
 }  
  
 }  
}

注：对于这个邮箱帐号和密码嘛，希望老师和助教调试完就忘了吧，个人邮箱又有点懒得删掉了，但是毕竟以后怕可能会有用，还是有点隐私的。。。