JavaMail(JAVA邮件服务）API详解（2） PDF转换可能丢失图片或格式，建议阅读原文 https://www.100test.com/kao_ti2020/144/2021_2022_JavaMail_J_c104_144863.htm F．Transport 在发送信息时，Transport类将被用到。这个类实现了发送信息的协议（通称为SMTP），此类是一个抽象类，我们可以使用这个类的静态方法send()来发送消息：Transport.send(message). 当然，方法是多样的。我们也可由Session获得相应协议对应的Transport实例。并通过传递用户名、密码、邮件服务器主机名等参数建立与邮件服务器的连接，并使用sendMessage()方法将信息发送，最后关闭连接：message.saveChanges(). // implicit with send()Transport transport = session.getTransport("smtp").transport.connect(host, username, password).transport.sendMessage(message, message.getAllRecipients()).transport.close(). 评论：上面的方法是一个很好的方法，尤其是在我们在同一个邮件服务器上发送多个邮件时。因为这时我们将在连接邮件服务器后连续发送邮件，然后再关闭掉连接。send()这个基本的方法是在每次调用时进行与邮件服务器的连接的，对于在同一个邮件服务器上发送多个邮件来讲可谓低效的方式。 注意：如果需要在发送邮件过程中监控mail命令的话，可以在发送前设置debug标志：session.setDebug(true)。 G．Store和Folder 接收邮件和发送邮件很类似都要用到Session。但是在获得Session后，我们需要从Session中获取特定类型的Store，然后连接到Store，这里的Store代表了存储邮件的邮件服务器。在连接Store的过程中，极有可能需要用到用户名、密码或者Authenticator。 // Store store = session.getStore("imap").Store store =

session.getStore("pop3").store.connect(host, username, password). 在连接到Store后，一个Folder对象即目录对象将通过Store的getFolder()方法被返回，我们可从这个Folder中读取邮件信息： Folder folder = store.getFolder("INBOX").folder.open(Folder.READ_ONLY).Message message[] = folder.getMessages(). 上面的例子首先从Store中获得INBOX这个Folder（对于POP3协议只有一个名为INBOX的Folder有效），然后以只读（Folder.READ_ONLY）的方式打开Folder，最后调用Folder的getMessages()方法得到目录中所有Message的数组。 注意：对于POP3协议只有一个名为INBOX的Folder有效，而对于IMAP协议，我们可以访问多个Folder（想想前面讲的IMAP协议）。而且SUN在设计Folder的getMessages()方法时采取了很智能的方式：首先接收新邮件列表，然后再需要的时候（比如读取邮件内容）才从邮件服务器读取邮件内容。 在读取邮件时，我们可以用Message类的getContent()方法接收邮件或是writeTo()方法将邮件保存，getContent()方法只接收邮件内容（不包含邮件头），而writeTo()方法将包括邮件头。 System.out.println(((MimeMessage)message).getContent()). 在读取邮件内容后，别忘记了关闭Folder和Store。 folder.close(aBoolean).store.close(). 传递给Folder.close()方法的boolean 类型参数表示是否在删除操作邮件后更新Folder。 H．继续向前进！ 在讲解了以上的七个Java Mail核心类定义和理解了简单的代码片断后，下文将详细讲解怎样使用这些类实现JavaMail API所要完成的高级功能。 五、使用JavaMail API 在明确了JavaMail API的核心部分如何工作后，本人将带领大

家学习一些使用Java Mail API任务案例。 1．发送邮件 在获得了Session后，建立并填入邮件信息，然后发送它到邮件服务器。这便是使用Java Mail API发送邮件的过程，在发送邮件之前，我们需要设置SMTP服务器：通过设置Properties的mail.smtp.host属性。 String host = ...String from = ...String to = ...// Get system propertiesProperties props = System.getProperties().// Setup mail serverprops.put("mail.smtp.host", host).// Get sessionSession session = Session.getDefaultInstance(props, null).// Define messageMimeMessage message = new MimeMessage(session).message.setFrom(new InternetAddress(from)).message.addRecipient(Message.RecipientType.TO, new InternetAddress(to)).message.setSubject("Hello JavaMail").message.setText("Welcome to JavaMail").// Send messageTransport.send(message). 由于建立邮件信息和发送邮件的过程中可能会抛出异常，所以我们需要将上面的代码放入到try-catch结构块中。 2．接收邮件 为了在读取邮件，我们获得了session，并且连接到了邮箱的相应store，打开相应的Folder，然后得到我们想要的邮件，当然别忘记了在结束时关闭连接。 String host = ...String username = ...String password = ...// Create empty propertiesProperties props = new Properties().// Get sessionSession session = Session.getDefaultInstance(props, null).// Get the storeStore store = session.getStore("pop3").store.connect(host, username, password).// Get folderFolder folder = store.getFolder("INBOX").folder.open(Folder.READ_ONLY).//

Get directoryMessage message[] = folder.getMessages().for (int i=0, n=message.length. i上面的代码所作的是从邮箱中读取每个邮件，并且显示邮件的发信人地址和主题。从技术角度讲，这里存在着一个异常的可能：当发信人地址为空时，getFrom()[0]将抛出异常。 下面的代码片断有效的说明了如何读取邮件内容，在显示每个邮件发信人和主题后，将出现用户提示从而得到用户是否读取该邮件的确认，如果输入YES的话，我们可用Message.writeTo(java.io.OutputStream os)方法将邮件内容输出到控制台上，关于Message.writeTo()的具体用法请看JavaMail API。 BufferedReader reader = new BufferedReader (new InputStreamReader(System.in)).// Get directoryMessage message[] = folder.getMessages().for (int i=0, n=message.length. i