转 JavaMail 发送邮件的实例

2013年10月12日 13:17:45　阅读数：2085

JavaMail是提供给开发者处理电子邮件相关的编程接口。它是Sun发布的用来处理email的API。今天学习了一下JavaMail，javamail发送邮件确实是一个比较麻烦的问题。为了以后使用方便，自己写了段代码，打成jar包，以方便以后使用。呵呵

以下三段代码是我的全部代码，朋友们如果想用，直接复制即可。
第一个类：MailSenderInfo.java

```java
/**
* 发送邮件需要使用的基本信息
*/
import java.util.Properties;
public class MailSenderInfo {
    // 发送邮件的服务器的IP和端口
    private String mailServerHost;
    private String mailServerPort = "25";
    // 邮件发送者的地址
    private String fromAddress;
    // 邮件接收者的地址
    private String toAddress;
    // 登陆邮件发送服务器的用户名和密码
    private String userName;
    private String password;
    // 是否需要身份验证
    private boolean validate = false;
    // 邮件主题
    private String subject;
    // 邮件的文本内容
    private String content;
    // 邮件附件的文件名
    private String[] attachFileNames;
    /**
      * 获得邮件会话属性
      */
    public Properties getProperties(){
      Properties p = new Properties();
      p.put("mail.smtp.host", this.mailServerHost);
      p.put("mail.smtp.port", this.mailServerPort);
      p.put("mail.smtp.auth", validate ? "true" : "false");
      return p;
    }
    public String getMailServerHost() {
      return mailServerHost;
    }
    public void setMailServerHost(String mailServerHost) {
      this.mailServerHost = mailServerHost;
    }
    public String getMailServerPort() {
      return mailServerPort;
    }
    public void setMailServerPort(String mailServerPort) {
      this.mailServerPort = mailServerPort;
    }
    public boolean isValidate() {
      return validate;
    }
    public void setValidate(boolean validate) {
      this.validate = validate;
    }
    public String[] getAttachFileNames() {
      return attachFileNames;
    }
    public void setAttachFileNames(String[] fileNames) {
      this.attachFileNames = fileNames;
    }
    public String getFromAddress() {
      return fromAddress;
    }
    public void setFromAddress(String fromAddress) {
      this.fromAddress = fromAddress;
    }
    public String getPassword() {
      return password;
    }
    public void setPassword(String password) {
      this.password = password;
    }
    public String getToAddress() {
      return toAddress;
    }
    public void setToAddress(String toAddress) {
      this.toAddress = toAddress;
```

```
    }
    public String getUserName() {
      return userName;
    }
    public void setUserName(String userName) {
      this.userName = userName;
    }
    public String getSubject() {
      return subject;
    }
    public void setSubject(String subject) {
      this.subject = subject;
    }
    public String getContent() {
      return content;
    }
    public void setContent(String textContent) {
      this.content = textContent;
    }
}
```

第二个类：SimpleMailSender.java

```java
import java.util.Date;
import java.util.Properties;
import javax.mail.Address;
import javax.mail.BodyPart;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Multipart;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;

/**
* 简单邮件（不带附件的邮件）发送器
*/
public class SimpleMailSender  {
/**
  * 以文本格式发送邮件
  * @param mailInfo 待发送的邮件的信息
  */
    public boolean sendTextMail(MailSenderInfo mailInfo) {
      // 判断是否需要身份认证
      MyAuthenticator authenticator = null;
      Properties pro = mailInfo.getProperties();
      if (mailInfo.isValidate()) {
      // 如果需要身份认证，则创建一个密码验证器
        authenticator = new MyAuthenticator(mailInfo.getUserName(), mailInfo.getPassword());
      }
      // 根据邮件会话属性和密码验证器构造一个发送邮件的session
      Session sendMailSession = Session.getDefaultInstance(pro,authenticator);
      try {
      // 根据session创建一个邮件消息
      Message mailMessage = new MimeMessage(sendMailSession);
      // 创建邮件发送者地址
      Address from = new InternetAddress(mailInfo.getFromAddress());
      // 设置邮件消息的发送者
      mailMessage.setFrom(from);
      // 创建邮件的接收者地址，并设置到邮件消息中
      Address to = new InternetAddress(mailInfo.getToAddress());
      mailMessage.setRecipient(Message.RecipientType.TO,to);
      // 设置邮件消息的主题
      mailMessage.setSubject(mailInfo.getSubject());
      // 设置邮件消息发送的时间
      mailMessage.setSentDate(new Date());
      // 设置邮件消息的主要内容
      String mailContent = mailInfo.getContent();
      mailMessage.setText(mailContent);
      // 发送邮件
      Transport.send(mailMessage);
      return true;
      } catch (MessagingException ex) {
          ex.printStackTrace();
      }
      return false;
    }

    /**
      * 以HTML格式发送邮件
      * @param mailInfo 待发送的邮件信息
      */
    public static boolean sendHtmlMail(MailSenderInfo mailInfo){
      // 判断是否需要身份认证
      MyAuthenticator authenticator = null;
```

```
        Properties pro = mailInfo.getProperties();
        //如果需要身份认证，则创建一个密码验证器
        if (mailInfo.isValidate()) {
          authenticator = new MyAuthenticator(mailInfo.getUserName(), mailInfo.getPassword());
        }
        // 根据邮件会话属性和密码验证器构造一个发送邮件的session
        Session sendMailSession = Session.getDefaultInstance(pro,authenticator);
        try {
        // 根据session创建一个邮件消息
        Message mailMessage = new MimeMessage(sendMailSession);
        // 创建邮件发送者地址
        Address from = new InternetAddress(mailInfo.getFromAddress());
        // 设置邮件消息的发送者
        mailMessage.setFrom(from);
        // 创建邮件的接收者地址，并设置到邮件消息中
        Address to = new InternetAddress(mailInfo.getToAddress());
        // Message.RecipientType.TO属性表示接收者的类型为TO
        mailMessage.setRecipient(Message.RecipientType.TO,to);
        // 设置邮件消息的主题
        mailMessage.setSubject(mailInfo.getSubject());
        // 设置邮件消息发送的时间
        mailMessage.setSentDate(new Date());
        // MiniMultipart类是一个容器类，包含MimeBodyPart类型的对象
        Multipart mainPart = new MimeMultipart();
        // 创建一个包含HTML内容的MimeBodyPart
        BodyPart html = new MimeBodyPart();
        // 设置HTML内容
        html.setContent(mailInfo.getContent(), "text/html; charset=utf-8");
        mainPart.addBodyPart(html);
        // 将MiniMultipart对象设置为邮件内容
        mailMessage.setContent(mainPart);
        // 发送邮件
        Transport.send(mailMessage);
        return true;
        } catch (MessagingException ex) {
            ex.printStackTrace();
        }
        return false;
    }
}
```

第三个类：MyAuthenticator.java

```
[java]
import javax.mail.*;

public class MyAuthenticator extends Authenticator{
    String userName=null;
    String password=null;

    public MyAuthenticator(){
    }
    public MyAuthenticator(String username, String password) {
        this.userName = username;
        this.password = password;
    }
    protected PasswordAuthentication getPasswordAuthentication(){
        return new PasswordAuthentication(userName, password);
    }
}
```

下面给出使用上面三个类的代码：

注：这段不是转载的，自己试过确实好用~~

```java
public class test {
    public static void main(String[] args){
        //这个类主要是设置邮件
      MailSenderInfo mailInfo = new MailSenderInfo();
      mailInfo.setMailServerHost("smtp.163.com");
      mailInfo.setMailServerPort("25");
      mailInfo.setValidate(true);
      mailInfo.setUserName("07gdg");
      mailInfo.setPassword("xxx");//邮箱密码 ，此处省略
      mailInfo.setFromAddress("07gdg@163.com");//发送邮箱，拿我们系的公共邮箱试一试~
      mailInfo.setToAddress("leixiaohua1020@126.com"); //接收邮箱
      mailInfo.setSubject("2013年存储展望:大数据、云依然是主旋律");
      mailInfo.setContent("又到了年底，对于存储领域来说，这一年发生了太多的事，大数据迅速崛起成为IT热词，而与大数据相关的大数据衍生行业也得到了繁荣发展。2012年大数据从天而降，迅速占领科技报端，混合云存储初现端倪，NAS存储重现胜景，闪存技术和融合式基础架构也跻身主流，可以说2012年存储市场繁荣异常。");
        //这个类主要来发送邮件
      SimpleMailSender sms = new SimpleMailSender();
         sms.sendTextMail(mailInfo);//发送文体格式
         sms.sendHtmlMail(mailInfo);//发送html格式
    }

}
```

文章标签： JavaMail 发送邮件 实例

个人分类： J2EE