Artigo – http://mballem.wordpress.com/

Enviando email com a API JavaMail

Autor

Marcio Ballem: é formado em Sistemas de Informação e possui certificação Oracle Certified Professional, Java SE 6 Programmer. Trabalhou profissionalmente com desenvolvimento em Delphi 7 e Java JEE.

Introdução

Muitas vezes vejo em tópicos, criados em fóruns sobre Java, a dúvida do tipo "Como enviar email com Java". Na *internet* existem vários tutoriais sobre isso, mas mesmo assim, sempre existe algum ponto em que o usuário encontra alguma dificuldade e não consegue resolve-la.

Neste artigo irei demonstrar como criar uma pequena e simples aplicação de envio de email utilizando a API *JavaMail*. Demonstrarei como enviar email em texto simples e em formato *HTML*, também como anexar no email uma lista de arquivos em anexos e como enviá-lo a mais de um destinatário.

1. API JavaMail

A API *JavaMail* foi projetada para facilitar a criação de aplicações simples ou sofisticadas quando se faz necessário o uso de correio eletrônico, o famoso email.

O *JavaMail* possui todas as classes necessárias para criação de um serviço de email, desde a parte de conexão com a internet, o servidor de email, os protocolos necessários, a utilização de conexão por *Proxy* e a adição de mensagens e anexos. Desta forma não precisamos de nada além da biblioteca *JavaMail* para criar uma aplicação de envio ou recebimento de email.

Na grande parte das vezes o *JavaMail* é utilizado para envio de email através do protocolo *SMTP*, mas a API também da suporte ao protocolo *POP3* para recebimento de email e a protocolos como *DNS* e *IMAP*, todos os que precisamos para realizar o envio ou recebimento de um email.

Neste artigo o foco será o envio de email pela API *JavaMail* que deve ser baixada no portal da Oracle e adicionada ao projeto. Para baixá-la acesse: http://www.oracle.com/technetwork/java/javaee/index-138643.html, e adicione ao projeto o arquivo ***mail.jar***.

2. Classe MailJava

Na classe *MailJava* vamos criar todos os atributos que serão necessários para a criação e envio de um email. Teremos que adicionar propriedades como usuário e senha de um servidor de email, para a autenticação no servidor. Também propriedades de conexão com o servidor de email e é claro os dados que serão enviados como o corpo da mensagem, assunto e anexos.

Crie uma classe conforme a listagem 1 e leia atentamente os comentários do código para saber qual o papel de cada atributo na classe.

Listagem 1. Classe MailJava

```
package br.mb.tutorialJavaMail;
```

```java
public class MailJava {
    //indica se o formato de texto será texto ou html
    public static final String TYPE_TEXT_PLAIN = "text/plain";
    public static final String TYPE_TEXT_HTML = "text/html";
    //indica qual será o servidor de email(gmail, hotmail...)
    private String smtpHostMail;
    //indica a porta de acesso ao servidor
    private String smtpPortMail;
    //indica que a necessidade de autenticação no servidor(true ou false)
    private String smtpAuth;
    //indica ao servidor que ele está recebendo uma conexão segura
    private String smtpStarttls;
    //nome do remetente do email
    private String fromNameMail;
    //email do remetente
    private String userMail;
    //senha do email do remetente
    private String passMail;
    //assunto do email
    private String subjectMail;
    //corpo do email, onde estará o texto da mensagem
    private String bodyMail;
    //lista com email e nome dos destinatários
    private Map<String, String> toMailsUsers;
    //lista contendo os arquivos anexos
    private List<String> fileMails;
    //charset, no caso de html é necessário
    private String charsetMail;
    //tipo do formato da mensagem, texto ou html
    private String typeTextMail;

        // gere os métodos getters and setters
}
```

Algumas configurações são diferentes para cada servidor de email. Configurações como a *smtpHostMail* e *smtpPortMail* serão diferentes para cada servidor e alguns exigirão que a configuração do *smtpStarttls* seja setada como *true* e alguns não.

Veja na tabela 1 algumas configurações para servidores distintos.

Servidor	*smtpHostMail*	*smtpPortMail*	*smtpStarttls*
Gmail	smtp.gmail.com	587	true
Bol	smtps.bol.com.br	587	true
Ibest	smtp.ibest.com.br	587	true
IG	smtp.ig.com.br	587	true
Hotmail	smtp.live.com	25	true

Tabela 1 – Configuração de servidores

A maioria dos servidores exige autenticação segura, porém servidores de email de empresas com domínio próprio, muitas vezes não exigem essa segurança. Caso deseje testar com mais algum servidor faça uma busca no *Google* por acesso *SMTP* + o servidor desejado, assim vai encontrar facilmente os dados necessários para a conexão.

3. Classe de Envio

Esta próxima classe que será criada é a classe responsável por receber as configurações, criar os objetos necessários para o email e então enviá-lo. É nesta classe que iremos trabalhar com a API *JavaMail*, então tome muito cuidado ao importar as classes do *JavaMail* para não importar classes com o mesmo nome de outros pacotes.

A listagem 2 contém a classe MailJavaSender.

Listagem 2. Arquivo jndi.properties

```
package br.mb.tutorialJavaMail;

import javax.activation.DataHandler;
import javax.activation.FileDataSource;
import javax.mail.Authenticator;
import javax.mail.PasswordAuthentication;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.Session;
import javax.mail.Multipart;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeBodyPart;
import javax.mail.internet.MimeMessage;
import javax.mail.internet.MimeMultipart;
import java.io.UnsupportedEncodingException;
import java.util.Map;
import java.util.Properties;

public class MailJavaSender {

    //cria as propriedades necessárias para o envio de email
    public void senderMail(final MailJava mail) throws
        UnsupportedEncodingException, MessagingException {

        Properties props = new Properties();
        props.setProperty("mail.transport.protocol", "smtp");
        props.setProperty("mail.host", mail.getSmtpHostMail());
        props.setProperty("mail.smtp.auth", mail.getSmtpAuth());
        props.setProperty("mail.smtp.starttls.enable", mail.getSmtpStarttls());
        props.setProperty("mail.smtp.port", mail.getSmtpPortMail());
        props.setProperty("mail.mime.charset", mail.getCharsetMail());

        //classe anonima que realiza a autenticação
        //do usuario no servidor de email.
        Authenticator auth = new Authenticator() {
            public PasswordAuthentication getPasswordAuthentication() {
                return new PasswordAuthentication(
                                mail.getUserMail(), mail.getPassMail()
                 );
            }
        };

        // Cria a sessao passando as propriedades e a autenticação
        Session session = Session.getDefaultInstance(props, auth);
        // Gera um log no console referente ao processo de envio
        session.setDebug(true);
```

```
        //cria a mensagem setando o remetente e seus destinatários
        Message msg = new MimeMessage(session);
        //aqui seta o remetente
        msg.setFrom(new InternetAddress(
                        mail.getUserMail(), mail.getFromNameMail())
        );
        boolean first = true;
        for (Map.Entry<String, String> map : mail.getToMailsUsers().entrySet()) {
            if (first) {
                //setamos o 1° destinatario
                msg.addRecipient(Message.RecipientType.TO,
                        new InternetAddress(map.getKey(), map.getValue())
                 );
                first = false;
            } else {
                //setamos os demais destinatarios
                msg.addRecipient(Message.RecipientType.CC,
                        new InternetAddress(map.getKey(), map.getValue())
                 );
            }
        }

        // Adiciona um Assunto a Mensagem
        msg.setSubject(mail.getSubjectMail());

        // Cria o objeto que recebe o texto do corpo do email
        MimeBodyPart textPart = new MimeBodyPart();
        textPart.setContent(mail.getBodyMail(), mail.getTypeTextMail());

        // Monta a mensagem SMTP  inserindo o conteudo, texto e anexos
        Multipart mps = new MimeMultipart();
        for (int index = 0; index < mail.getFileMails().size(); index++) {

            // Cria um novo objeto para cada arquivo, e anexa o arquivo
            MimeBodyPart attachFilePart = new MimeBodyPart();
            FileDataSource fds = new FileDataSource(
                 mail.getFileMails().get(index)
            );
            attachFilePart.setDataHandler(new DataHandler(fds));
            attachFilePart.setFileName(fds.getName());

            //adiciona os anexos da mensagem
            mps.addBodyPart(attachFilePart, index);

        }

        //adiciona o corpo texto da mensagem
        mps.addBodyPart(textPart);

        //adiciona a mensagem o conteúdo texto e anexo
        msg.setContent(mps);

        // Envia a Mensagem
        Transport.send(msg);
    }

}
```

Podemos setar três configurações para o envio de email referente aos destinatários.

O BCC (com cópia oculta) que serve para enviar para o destinatário um email com cópias do email sem que os outros fiquem sabendo.

O CC (com cópia) envia uma cópia para a lista de destinatários, mas ao mesmo tempo os outros destinatários ficam sabendo do envio entre si.

O TO (para) por sua vez é o mais comum e sempre é usado para enviar para os destinatários. Todos ficam sabendo de todos os outros destinatários também.

4. Criando a classe de teste

Criaremos uma terceira classe que será responsável por informar todos os dados referentes ao envio do email. Nesta classe temos dois métodos que serão responsáveis pelo tipo de conteúdo do corpo da mensagem, um do tipo texto e outro do tipo *HTML*. Basta alterar a propriedade referente para enviar como um ou outro.

Sete as configurações referentes ao seu servidor de email. Insira quantos destinatários quiser na lista e também quantos anexos desejar.

Listagem 3. Classe MailTester

```
package br.mb.tutorialJavaMail;

import javax.mail.MessagingException;
import java.io.UnsupportedEncodingException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

public class MailTester {
    public static void main(String[] args) {
        MailJava mj = new MailJava();
        //configuracoes de envio
        mj.setSmtpHostMail("smtp.gmail.com");
        mj.setSmtpPortMail("587");
        mj.setSmtpAuth("true");
        mj.setSmtpStarttls("true");
        mj.setUserMail("seuEmail@gmail.com");
        mj.setFromNameMail("seuNome");
        mj.setPassMail("suaSenha");
        mj.setCharsetMail("ISO-8859-1");
        mj.setSubjectMail("JavaMail");
        mj.setBodyMail(htmlMessage());
        mj.setTypeTextMail(MailJava.TYPE_TEXT_HTML);

        //sete quantos destinatarios desejar
        Map<String, String> map = new HashMap<String, String>();
        map.put("destinatario1@bol.com.br", "email bol");
        map.put("destinatario2@msn.com", "email msn");
        map.put("destinatario3@ig.com.br", "email ig");

        mj.setToMailsUsers(map);

        //seta quatos anexos desejar
        List<String> files = new ArrayList<String>();
        files.add("C:\\images\\ajax_loader.gif");
```

```
        files.add("C:\\images\\hover_next.png");
        files.add("C:\\images\\hover_prev.png");

        mj.setFileMails(files);

        try {
            new MailJavaSender().senderMail(mj);
        } catch (UnsupportedEncodingException e) {
            e.printStackTrace();
        } catch (MessagingException e) {
            e.printStackTrace();
        }
    }

    private static String textMessage() {
        return  "Leia o novo tutorial JavaMail do Programando com Java.\n" +
                "Saiba como enviar emails com anexo, em formato texto e html.\n" +
                "Envie seu email para mais de um destinatario.";
    }

    private static String htmlMessage() {
        return
"<html>\n" +
"\t<head>\n" +
"\t\t<title>Email no formato HTML com Javamail!</title> \n" +
"\t</head>\n" +
"\t<body>\n" +
"\t\t<div style='background-color:orange; width:28%; height:100px;'>\n" +
"\t\t\t<ul>\n" +
"\t\t\t\t<li>Leia o novo tutorial JavaMail do Programando com Java.</li>\n" +
"\t\t\t\t<li>Aprenda como enviar emails com anexos.</li>\n" +
" \t\t\t\t<li>Aprenda a enviar emails em formato texto simples ou html.</li> \n" +
"\t\t\t\t<li>Aprenda como enviar seu email para mais de um destinátario.</li>\n" +
"\t\t\t</ul>\n" +
"\t\t\t<p>Visite o blog \n" +
"\t\t\t\t<a href='http://mballem.wordpress.com/'>Programando com Java</a>\n" +
"\t\t\t</p>\n" +
"\t\t</div>\t\n" +
"\t\t<div style='width:28%; height:50px;' align='center'>\n" +
"\t\t\tDownload do JavaMail<br/>\n" +
"\t\t\t<a href='http://www.oracle.com/technetwork/java/javaee/index-
138643.html'>\n" +
"\t\t\t\t<img
src='http://www.oracleimg.com/admin/images/ocom/hp/oralogo_small.gif'/>\n" +
"\t\t\t</a> \n" +
"\t\t</div>\t\t\n" +
"\t</body> \n" +
"</html>";
    }
}
```

Conclusão

Vimos como configurar uma classe de envio de email pela API *JavaMail*. Essa classe que postei no artigo não precisa ser exatamente desta forma, você pode encontrar outras maneiras de programá-la, mas as configurações e objetos criados na classe, com certeza serão sempre usadas.

No arquivo baixado para adquirir a biblioteca do *JavaMail*, você pode encontrar a documentação do código fonte de toda a API e ainda um arquivo em PDF que contém exemplos e explicações de como utilizar a API.