



Heitor Gonzaga de Moura Neto, Técnico em Processamento de Dados pela ETF-TO ( Escola Técnica Federal do Estado do Tocantins ) , cursando Sistemas de Informação - Faculdade Católica, atualmente na Phoenix Desenvolvimento e Consultoria, desenvolvendo aplicações comerciais em plataforma Java há mais de 2 anos.

## API JAVAMAIL

O JavaMail é uma API opcional da plataforma Java, que nos proporciona funcionalidades de correios eletrônico através do pacote javax.mail, nos permitindo realizar desde simples tarefas como enviar e receber e-mails até tarefas mais complexas.

O JavaMail por padrão suporta os protocolos de envio e recebimento SMTP, IMAP, POP3 e suas versões seguras.

Neste artigo termos como principal objetivo abordar alguns dos principais aspectos da API JavaMail de forma que ao termino deste artigo conseguiremos criar uma classe que nos permitirá enviar um e-mail.

## DOWNLOAD E INSTALAÇÃO

A API do JavaMail pode ser baixada a partir do seguinte endereço <http://java.sun.com/products/javamail/downloads/index.html> e sua documentação em <http://java.sun.com/products/javamail/reference/api/index.html>.

Ao descompactar o arquivo baixado, dentro da pasta /lib deve conter 5 arquivos, sendo eles:

- **Mail.jar:** Que contem toda a implementação da API JavaMail e todos protocolos como SMTP, IMAP e POP3 assim como suas versões seguras SMTPS, IMAPS e POP3S.
- **MailApi.jar:** Contem a implementação da API JavaMail, porém não possui a implementação dos protocolos.
- **Imap.jar:** Possui a implementação do protocolo imap.
- **Pop3.jar:** Possui a implementação do protocolo pop3.
- **Smtp.jar:** Possui a implementação do protocolo smtp.

Apenas o uso do arquivo **mail.jar** é suficiente para nossa aplicação, a não ser é claro que queiramos reduzir o tamanho da mesma, neste caso utilizaríamos o arquivo **mailapi.jar** com os arquivos referentes aos protocolos que utilizaríamos para envio e recebimento.

## PREPARANDO O CAFÉ

Neste primeiro exemplo estarei apenas mostrando como enviar um e-mail de texto plano contendo apenas dados básico como assunto e corpo de mensagem, para isto faremos uso do protocolo SMTP para envio de mensagem.

Para a criação de nossas classes de envio de e-mail utilizando a API JavaMail utilizaremos como ferramenta de desenvolvimento o Eclipse.

Primeiramente vamos criar nosso projeto no Eclipse (Figura 1) em seguida vamos importar a API do JavaMail mail.jar ao nosso projeto (Figura 2).

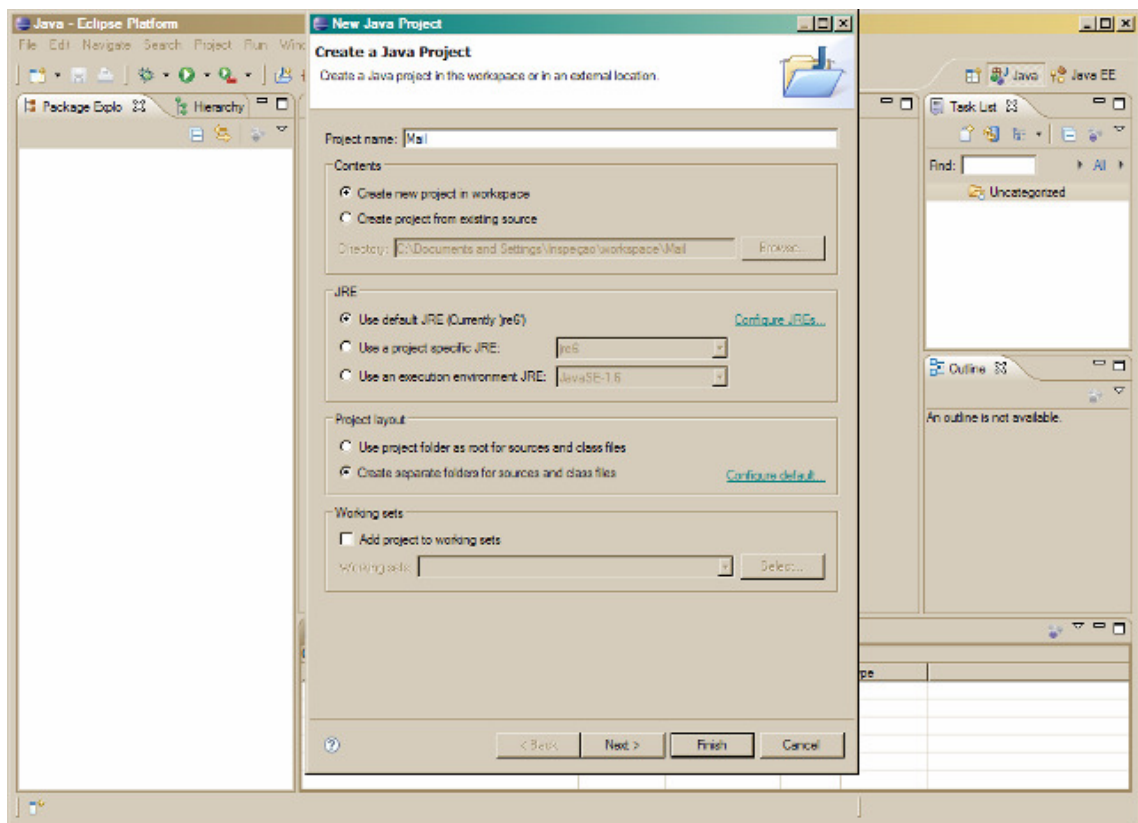


Figura 1 – Mostra a criação de um novo projeto no Eclipse.

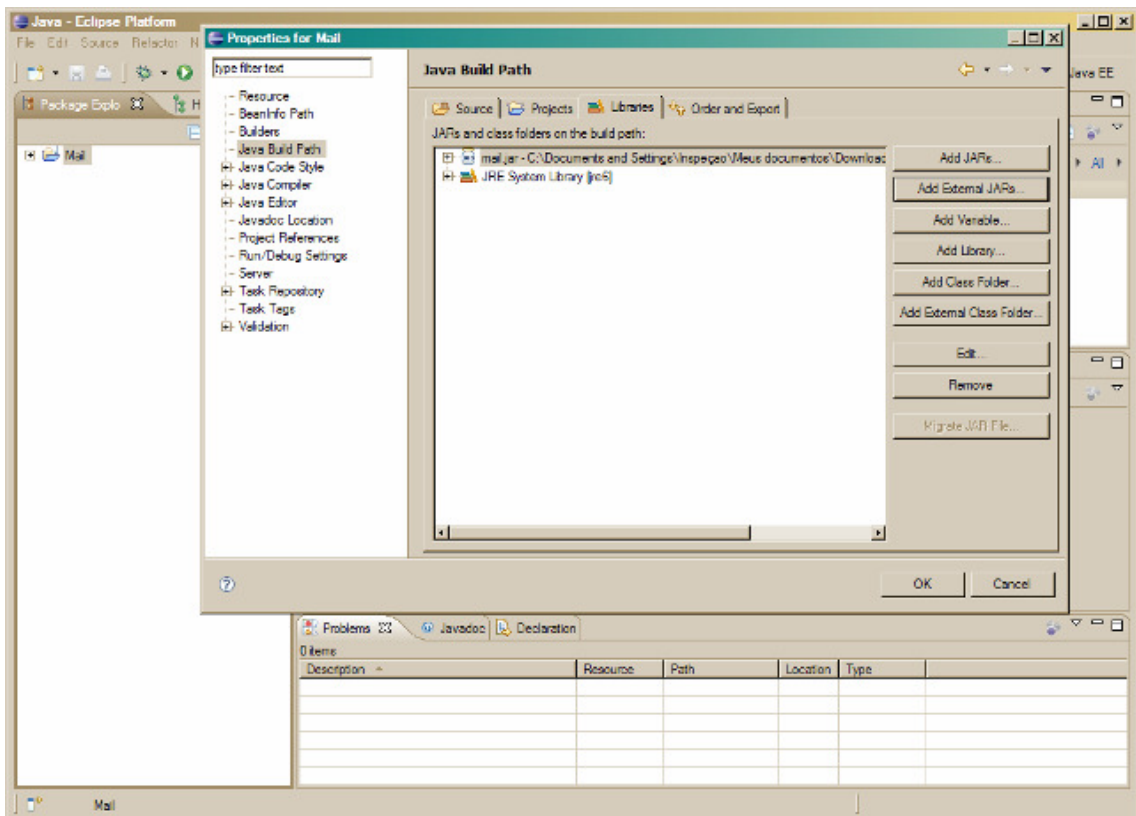


Figura 2 – Mostra a adição da API JavaMail mail.jar ao projeto no Eclipse.

O processo básico para envio de e-mail utilizando o JavaMail consiste em três passos:

1. Criar uma sessão e passar as propriedades da conexão, tais como servidor smtp, usuário e senha.
2. Criar uma mensagem e setar seus campos, tais como assunto, texto da mensagem, emissor e destinatário.
3. Enviar a mensagem.

Agora iniciando o desenvolvimento, vamos criar uma classe para envio de e-mail, irei criar uma classe chamada Envio que ira se responsabilizar por todo envio de e-mails utilizando a API do JavaMail.

Certo pessoal vamos colocar a mão na massa e começar a preparar o nosso café.

```

package email;

import java.util.Properties;

import javax.mail.*;
import javax.mail.internet.*;

public class Enviar {

    private String remetente;
    private String destinatario;
    private String smtpHost;
    private String porta;
    private String assunto;
    private Properties propriedades;

```

```

private Session sessao;
private static String usuario;
private static String senha;

private static class Autenticacao extends Authenticator
{
    public PasswordAuthentication getPasswordAuthentication()
    {
        return new PasswordAuthentication(usuario, senha);
    }
}

public Enviar(String remetente, String destinatario,
              String assunto, String smtpHost, String porta,
              String usuario, String senha)
{
    this.remetente = remetente;
    this.destinatario = destinatario;
    this.assunto = assunto;
    this.smtpHost = smtpHost;
    this.porta = porta;
    this.usuario = usuario;
    this.senha = senha;

    this.propriedades = System.getProperties();
    this.propriedades.put("mail.smtp.host", this.smtpHost);
    this.propriedades.put("mail.smtp.auth", "true");
    //this.propriedades.put("mail.debug", "true");
    //this.propriedades.put("mail.smtp.debug", "true");
    this.propriedades.put("mail.smtp.port", this.porta);
    this.propriedades.put("mail.smtp.starttls.enable", "true");
    this.propriedades.put("mail.smtp.socketFactory.port", this.porta);
    this.propriedades.put("mail.smtp.socketFactory.fallback",
    "false");
    this.propriedades.put("mail.smtp.socketFactory.class",
    "javax.net.ssl.SSLSocketFactory");
    //this.propriedades.put("mail.smtp.user",
    //"heitorgonzaga@gmail.com");

    Authenticator auth = new Autenticacao();
    this.sessao = Session.getDefaultInstance(this.propriedades, auth);

    try
    {
        Message mensagem = new MimeMessage(this.sessao);
        mensagem.setSubject(this.assunto);
        mensagem.setFrom(new InternetAddress(this.remetente));
        mensagem.addRecipient(Message.RecipientType.TO,
            new InternetAddress(this.destinatario));
        mensagem.setText("Este é o corpo da mensagem");
        System.out.println("Enviando mensagem");
        Transport.send(mensagem);
        System.out.println("Mensagem enviada");
    }
    catch (Exception err)
    {
        System.out.println("Erro ao enviar mensagem");
    }
}

```

```

public static void main(String args[])
{
    Enviar env = new
    Enviar("heitorgonzaga@gmail.com", "heitorgonzaga@gmail.com", "teste
    email", "smtp.gmail.com", "heitorgonzaga@gmail.com", "senhaaqui");
}
}

```

### **Agora vamos aos detalhes do nosso café.**

Primeiramente vamos aos nossos imports da vida, o pacote **javax.mail.\*** é o pacote que contem as classes que modelam ou descrevem as funcionalidades de um correio eletrônico genérico, o que quer dizer que possui características que são comuns a qualquer sistema de correio eletrônico, o pacote **javax.mail.internet.\***, possui as classes responsáveis por modelar um sistema de correio eletrônico utilizando conexão com internet como por exemplo o endereço de um destinatário de e-mail, e por ultimo o pacote **java.util.Properties** que irá representar um conjunto de propriedades tais como conexão com o propriedades de conexão com o servidor smtp.

Em seguida é criada uma Inner Class Autenticacao que extend de **javax.mail.Authenticator**. A Classe **javax.mail.Authenticator** serve para representar um objeto que necessite de uma autenticação em rede, porém a classe **Authenticator** é **abstrata** o que quer dizer que a subclasse que extender deve implementar seus métodos abstratos, no nosso caso o método é o **getPasswordAuthentication()** que nos irá retornar um objeto do tipo **PasswordAuthentication**, em seguida criamos o nosso método construtor onde devemos informar todas as propriedades referentes a conexão tais como servidor smtp, porta do servidor smtp, usuario, senha, remetente e destinatário entre outros, logo dentro no nosso método construtor nos iniciamos o nosso objeto propriedades que é uma instancia de Properties e informamos as propriedades de conexão com o servidor de e-mail tais como **“mail.smtp.host”** que identifica o nome do servidor smtp, **“mail.smtp.auth”** que identifica que o meu servidor requer autenticação, **“mail.smtp.port”** que identifica que nosso servidor estará recebendo conexões na porta 465, **“mail.smtp.starttls.enable”** informa que nosso servidor estará recebendo uma conexão segura, outras propriedades com **“mail.debug”** mostra com um log ou debug do processo corrente a conexão. Em seguida é criada uma **sessão** que recebe como parâmetros o objeto do tipo Properties que contém definida as nossas configurações do servidor smtp, assim como também o objeto que possui a autenticação do servidor, lembrando que se for um servidor de e-mail que não necessite de autenticação, não é obrigatório o uso do objeto com autenticação, basta apenas que seja passado como parâmetro o objeto do tipo Properties em seguida passe **null** referente ao objeto de autenticação, após isso é criada a nossa mensagem, para isso passamos como parâmetro a nossa sessão em seguida setamos as informações referentes a nossa mensagem como assunto, remetente, destinatário e texto da mensagem e utilizamos o método **send()** da classe Transport para enviar a nossa mensagem.

Como resultado é mostrado a saída Mensagem Enviada no console (Figura 3)

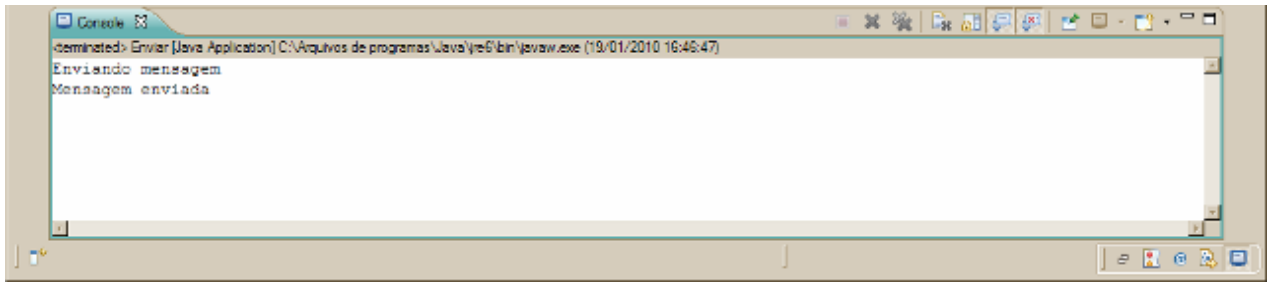


Figura 3 – Mostra o resultado da execução da classe Enviar.

Todos os testes foram feitos utilizando como serviço de e-mail gmail, e utilizando protocolo smtp. Em breve estarei postando um artigo sobre como receber e-mails utilizando a API JavaMail.

Espero que tenham gostado qualquer duvida me encontro a disposição para devidos esclarecimentos por e-mail [heitorgonzaga@gmail.com](mailto:heitorgonzaga@gmail.com), <http://heitormoura.blogspot.com>.

Até o próximo café.