

JavaMail

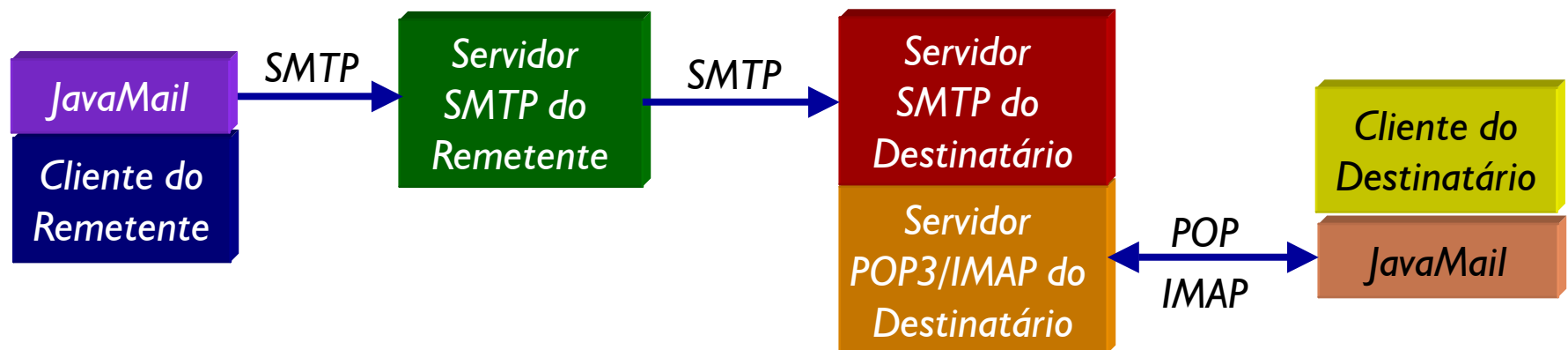
*Helder da Rocha (helder@acm.org)
argonavis.com.br*

O que é JavaMail?

- API genérica para construir aplicações que manipulam **correio eletrônico** (não necessariamente Internet e-mail) e **messaging** como clientes
 - Serve para estender aplicações existentes para lidar com mensagens, cabeçalhos, attachments, etc.
 - Permite um nível de abstração mais alta (em comparação com sockets)
 - Permite independência de implementação (em comparação alternativas como "com.sun.net.smtp.*")
- Distribuição oferece suporte a MIME, SMTP, IMAP, POP, etc. mas pode ser estendida

Arquitetura de sistemas de e-mail

- **SMTP - Simple Mail Transfer Protocol**
 - Principal meio de transporte para e-mail na Internet
- **POP - Post Office Protocol**
 - Lida com repositórios (Stores) de e-mails recebidos
 - Cliente POP pode se conectar a servidor para baixar mensagens recebidas
- **IMAP**
 - Pode ser usado no lugar do POP



- *Oferece uma camada abstrata sobre diversas implementações de correio eletrônico*
 - *Permite criar clientes genéricos*
- *Principais classes da API*
 - *Sessões: `javax.mail.Session`*
 - *Serviço de Armazenamento: `javax.mail.Store`*
 - *Serviço de transporte: `javax.mail.Transport`*
 - *Mensagens: `javax.mail.Message`, `Part`, `MultiPart`*
- *Maior parte da API são interfaces e classes abstratas*
 - *Há implementações para SMTP, IMAP, NNTP*, POP3*
- *JARs*
 - *`activation.jar` (JavaBeans Activation Framework)*
 - *`mail.jar`*

- *Dois papéis*
 - *guardar propriedades acessíveis via construtores*
 - *oferece métodos de fábrica para objetos Store e Transport*

- *Dois tipos de objetos Session*

```
Session session = Session.getDefaultInstance(props)  
Session session Session.getInstance(props)
```

- *Autenticadores*

- *Passados como parâmetro nos métodos getInstance()*

Propriedades como
servidor de email,
usuário, etc.

```
public class MyAuthenticator extends Authenticator {  
    protected PasswordAuthentication getPasswordAuthentication() {  
        return new PasswordAuthentication("scott", "tiger");  
    }  
}  
(...)  
Properties props = System.getProperties();  
Session session =  
    Session.getDefaultInstance(props, new MyAuthenticator());
```

- São encapsuladas na classe abstrata **Message**, que possui
 - Conjunto de cabeçalhos RFC822 (Headers)
 - Conteúdo (`javax.activation.DataHandler`)
- Há métodos get/set para conteúdo e cabeçalhos
- Distribuição `JavaMail` traz uma implementação da classe abstrata `Message`: **MimeMessage**

```
Message msg = new MimeMessage(session);
```

- Destinos e origens são definidos em objetos do tipo **Address** e adicionados nos cabeçalhos `To` e `From`

```
msg.setFrom(new InternetAddress("fulano@destino.org"));  
msg.setRecipients(Message.RecipientType.TO,  
                  InternetAddress.parse(recv, false));  
msg.setSubject("JavaMail Test Message");  
msg.setHeader("X-Mailer", "My Simple JavaMail Client");  
msg.setText("This is the sample message text");
```

Envio de mensagens

- *Depois de criada a mensagem, basta passá-la para um meio de transporte para enviá-la*
- *Forma mais simples de fazer é passar para o método (estático) `send()`, de `javax.mail.Transport`*

```
Transport.send(msg);
```

- *Pode-se também controlar o tipo de transporte usado (de acordo com o protocolo)*

```
Transport smtp = session.getTransport("smtp");  
smtp.send(msg);
```

```
Transport nntp = session.getTransport("nntp");  
nntp.send(msg);
```

Recuperação de Mensagens

- *JavaMail 1.2 possui implementações para POP3 e IMAP*
- *A classe Store permite a conexão a um repositório de mensagens (message store)*

```
Store popStore = session.getStore("pop3");  
popStore.connect("localhost", null, null);
```

- *Obtendo-se um objeto Store, é preciso buscar a pasta default e, a partir dela, a pasta desejada (por exemplo: INBOX).*
- *Depois é preciso abrir a pasta para leitura ou leitura/gravação*

```
Folder defaultPopFolder = popStore.getDefaultFolder();  
inboxFolder = defaultPopFolder.getFolder("INBOX");  
inboxFolder.open(Folder.READ_ONLY);
```

- *A partir daí, pode-se ler as mensagens*

```
Message[] msgs = inboxFolder.getMessages();
```


- A classe abstrata *SearchTerm* permite fazer buscas em mensagens dos folders com base em critérios dependentes da implementação de uma subclasse
 - *SubjectTerm*, faz buscas baseadas no assunto
 - *FromStringTerm* busca no cabeçalho *From*
 - *AndTerm*, *OrTerm*, *NotTerm* aceitam outros *SearchTerm*

```
SearchTerm st =  
    new AndTerm(new FromStringTerm("elfodon@gurus.org"),  
                new SubjectTerm("[java-list]));  
Message[] msgs = folder.search(st);
```

- Utilizando *SearchTerms*, pode-se filtrar mensagens indesejadas e apagá-las dos Folders

```
SearchTerm st = new SubjectTerm("Ganhe dinheiro!");  
Message[] msgs = folder.search(st);  
msgs[0].setFlag(Flags.Flag.DELETED, true);  
inboxFolder.close(true);
```

← Sinaliza para remoção de mensagem

Recebimento de mensagens com anexos

- Mensagens podem ter uma ou várias partes. Pode-se compor uma mensagem com várias outras mensagens
- Mensagens multipartes (Multipart) consistem de uma série de objetos **BodyPart**.
 - Cada BodyPart tem seu próprio tipo de dados MIME e conteúdo.
 - A mensagem tem tipo MIME que começa com "multipart/"
- Ao obter uma mensagem do folder, verifica-se se a mensagem é do tipo "multipart/*". Se for
 - Obtenha o conteúdo (objeto Multipart)
 - Use `mp.getBodyPart(idx)` para obter cada MimeBodyPart

```
if (msg.isMimeType("multipart/*")) {  
    Multipart mp = (Multipart) msg.getContent();  
    for (int i = 0; i < mp.getCount(); i++) {  
        MimeBodyPart bp = (MimeBodyPart)mp.getBodyPart(i);  
        Object o = p.getContent();  
    } (...)
```

Envio de mensagens com anexos

- Para enviar anexos enviamos a mensagem como *Multipart*
 - Objeto *MimeMultipart* serve de container para tipos MIME diferentes
 - Para texto da mensagem e cada anexo é preciso criar um objeto *MimeBodyPart* e preenche-lo através do método *setContent()*
 - No final, coloca-se o *MimeMultipart* em uma mensagem e envia-se

```
Message msg = new MimeMessage(session);
(...)
Multipart mp = new MimeMultipart();

MimeBodyPart mbp1 = new MimeBodyPart();
mbp1.setContent(args.length + " files attached.", "text/plain");
mp.addBodyPart(mbp1);

for (int i = 0; i < args.length; i++) {
    File f = new File(args[i]);
    MimeBodyPart mbp = new MimeBodyPart();
    mbp.setFileName (f.getName());
    mbp.setDataHandler(new DataHandler(new FileDataSource(f)));
    mp.addBodyPart(mbp);
}
msg.setContent(mp);
Transport.send(msg);
```

Execução dos exemplos

- **Configuração do ambiente**
 - *Primeiro configure o arquivo `smtp.properties` (em `lib/`) e informe o endereço do servidor de e-mail utilizado, nome e senha.*
 - *Se quiser usar um servidor local, você pode usar o Jakarta James que suporta POP, NNTP e SMTP. Inicie o James.*
- **Execução (em `cap17/nut/`)**
 - *Envio de mensagens simples*
 - > **`ant sendmail`**
 - *Envio de mensagens com attachment: passe a lista de arquivos entre aspas para o argumento `"-Dattach"` no `build.xml`*
 - > **`ant sendmail -Dattach="java.gif build.xml"`**
 - *Listagem das mensagens no INBOX*
 - > **`ant getmail`**
 - *Para ver o conteúdo (e possivelmente esvaziar o INBOX) use seu cliente de e-mail (ou escreva um que faça isto!)*

- *JavaMail oferece uma API que permite transformar qualquer aplicação Java em um cliente IMAP, SMTP, POP3 ou SMTP*
- *Permite capacitar servlets, JSPs, EJBs e outros componentes com recursos de e-mail*
 - *Pode ser usado para messaging XML (Web Services)*
 - *Pode ser usado para transferir arquivos de forma assíncrona (usando attachments)*

- *1. Escreva uma página JSP simples que ofereça uma interface para*
 - *Escrever uma mensagem*
 - *Fazer upload de um arquivo (attachment). Use a biblioteca cos.jar em cap06\hunter\upload\lib (veja exemplos na mesma pasta)*
 - *Enviar mensagem com arquivo atachado*
- *2. Escreva uma página JSP que*
 - *Conecte-se ao servidor de e-mail*
 - *Liste as mensagens disponíveis: Data de envio, remetente, assunto e link para attachments.*
- *Aproveite os esqueletos disponíveis!*

[1] Jim Farley, William Crawford e David Flanagan. *Java Enterprise in a Nutshell*. 2nd. Edition, 2002. O'Reilly.

helder@ibpinet.net

www.argonavis.com.br