J2EE

JavaMail API

Presented by Bartosz Sakowicz

Instalation

Java Mail API:
http://java.sun.com/products/javamail/index.html

Java Activation Framework
http://java.sun.com/products/javabeans/glasgow/jaf.html

Used for automatic recognition of data. It allows to operate in easy way on data and call data-specific methods.

Installation: put mail.jar and activation.jar to /lib directory (tomcat/lib for all applications or WEB-INF/lib for particular application)

Note:
Sun One Studio 4 (Forte for Java) already has necessary files, so using JavaMail from JSP pages is possible without any changes.

JavaMail overview

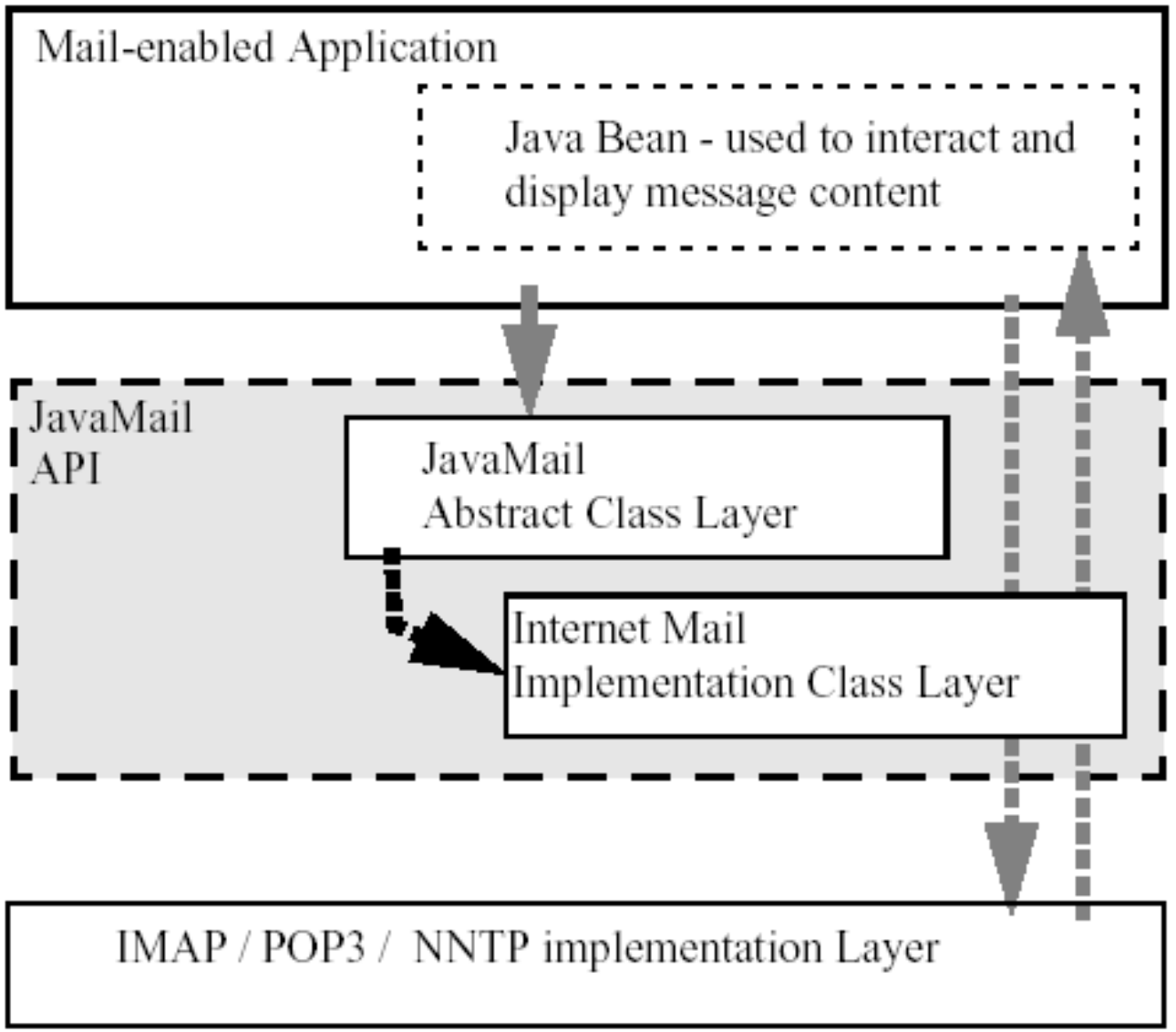
The JavaMail API is designed to make adding electronic mail capability to simple applications easy, while also supporting the creation of sophisticated user interfaces.

It includes appropriate convenience classes which encapsulate common mail functions and protocols. It fits with other packages for the Java platform in order to facilitate its use with other Java APIs, and it uses familiar programming models.

Presented by Bartosz Sakowicz

DMCS TUL

Architecture

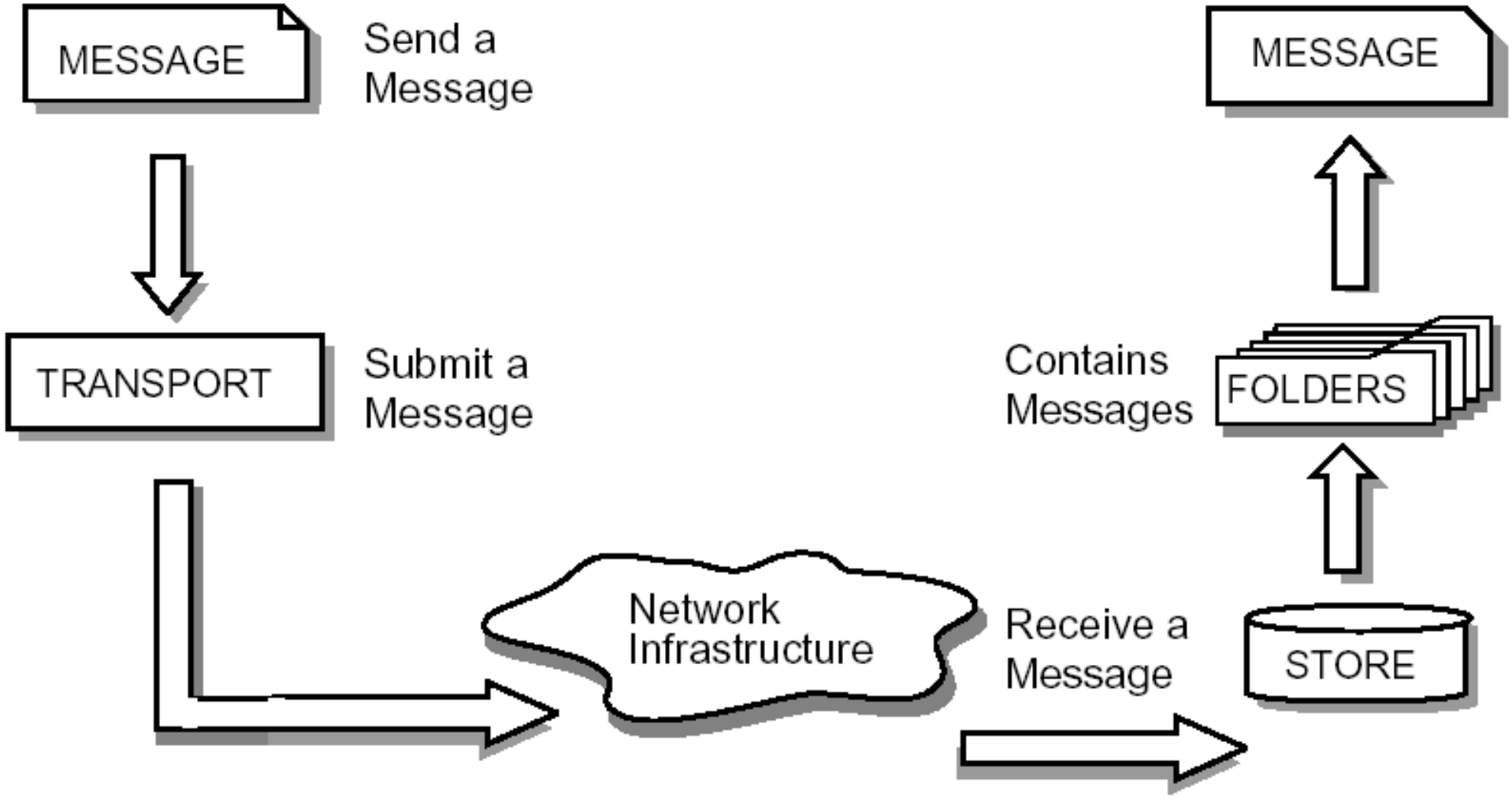


Architecture(2)

The **Abstract Layer** declares classes, interfaces and abstract methods intended to support mail handling functions that all mail systems support. API elements comprising the Abstract Layer are intended to be subclassed and extended as necessary in order to support standard data types, and to interface with message access and message transport protocols as necessary.

The **internet implementation layer** implements part of the abstract layer using internet standards.

Message Handling Process



Presented by Bartosz Sakowicz

DMCS TUL

API

The Message Class

The Message class is an abstract class that defines a set of attributes and a content for a mail message. Attributes of the Message class specify addressing information and define the structure of the content, including the content type.

Message Composition and Transport

A client creates a new message by instantiating an appropriate Message subclass. It sets attributes like the recipient addresses and the subject, and inserts the content into the Message object. Finally, it sends the Message by invoking the Transport.send method.

The Transport class models the transport agent that routes a message to its destination addresses. This class provides methods that send a message to a list of recipients.

Presented by Bartosz Sakowicz

DMCS TUL

API(2)

The Session Class

The Session class defines global and per-user mail-related properties that define the interface between a mail-enabled client and the network. JavaMail system components use the Session object to set and get specific properties.

The Session class also provides a default authenticated session object that desktop applications can share. The Session class is a final class. It cannot be subclassed.

Sending Messages

```
<%@
page import="java.util.*, javax.mail.*, javax.mail.internet.*" %>
<%
Properties props = new Properties();
props.put("mail.smtp.host", "smtp.mail.example.com");

Session s = Session.getInstance(props,null);
MimeMessage message = new MimeMessage(s);

InternetAddress from = new
        InternetAddress("you@example.com");

message.setFrom(from);
```

Sending Messages(2)

```
InternetAddress to =
        new InternetAddress("you@example.com");

message.addRecipient(Message.RecipientType.TO, to);

// Add rest of recipients in analogous way...


message.setSubject("Test from JavaMail.");

message.setText("Hello from JavaMail!");

Transport.send(message);

%>
```

Recipients

RecipientType.TO
The basic "to" address of an email.

RecipientType.CC
Carbon Copy; An address that should be sent a copy of the message. Other recipients will be aware of the copy sent.

RecipientType.BCC
Blind Carbon Copy; An address that should recieve a copy of the message, but all other addresses will not be notified of the copy.

Attachments

```
Message message = new MimeMessage(session);
// set from, to , subject ...

BodyPart messageBodyPart = new MimeBodyPart();
messageBodyPart.setText("Pardon Ideas");
Multipart multipart = new MimeMultipart();
multipart.addBodyPart(messageBodyPart);

messageBodyPart = new MimeBodyPart();
DataSource source = new FileDataSource(filename);
messageBodyPart.setDataHandler(new DataHandler(source));
messageBodyPart.setFileName(filename);
multipart.addBodyPart(messageBodyPart);

message.setContent(multipart);
```

Getting messages

```
String host = ...;
String username = ...;
String password = ...;
Properties props = new Properties();
Session session = Session.getDefaultInstance(props, null);
Store store = session.getStore("pop3");
store.connect(host, username, password);
Folder folder = store.getFolder("INBOX");
folder.open(Folder.READ_ONLY);
message[] = folder.getMessages();
for (int i=0, n=message.length; i<n; i++) {
System.out.println(i + ": " + message[i].getFrom()[0] + "\t" +
message[i].getSubject()); }
folder.close(false);
store.close();
```

Other features

With JavaMail it is also possible:

- Replying messages
- Forwarding messages
- Deleting messages
- Searching through messages