

# Apache James

## The Java Mail Server

Henning P. Schmiedehausen  
<henning@apache.org>

ApacheCon EU 2005, July 22th 2005  
Stuttgart, Germany

# An Apache Mail Server?

- Did you know that the ASF offers a complete Mail Server Solution?
- That it does so since 1999?
- A server that actually runs on **all** available platforms?

(that have a Java 1.3 compliant JRE)

- That is embeddable and extensible?

# Meet James

---

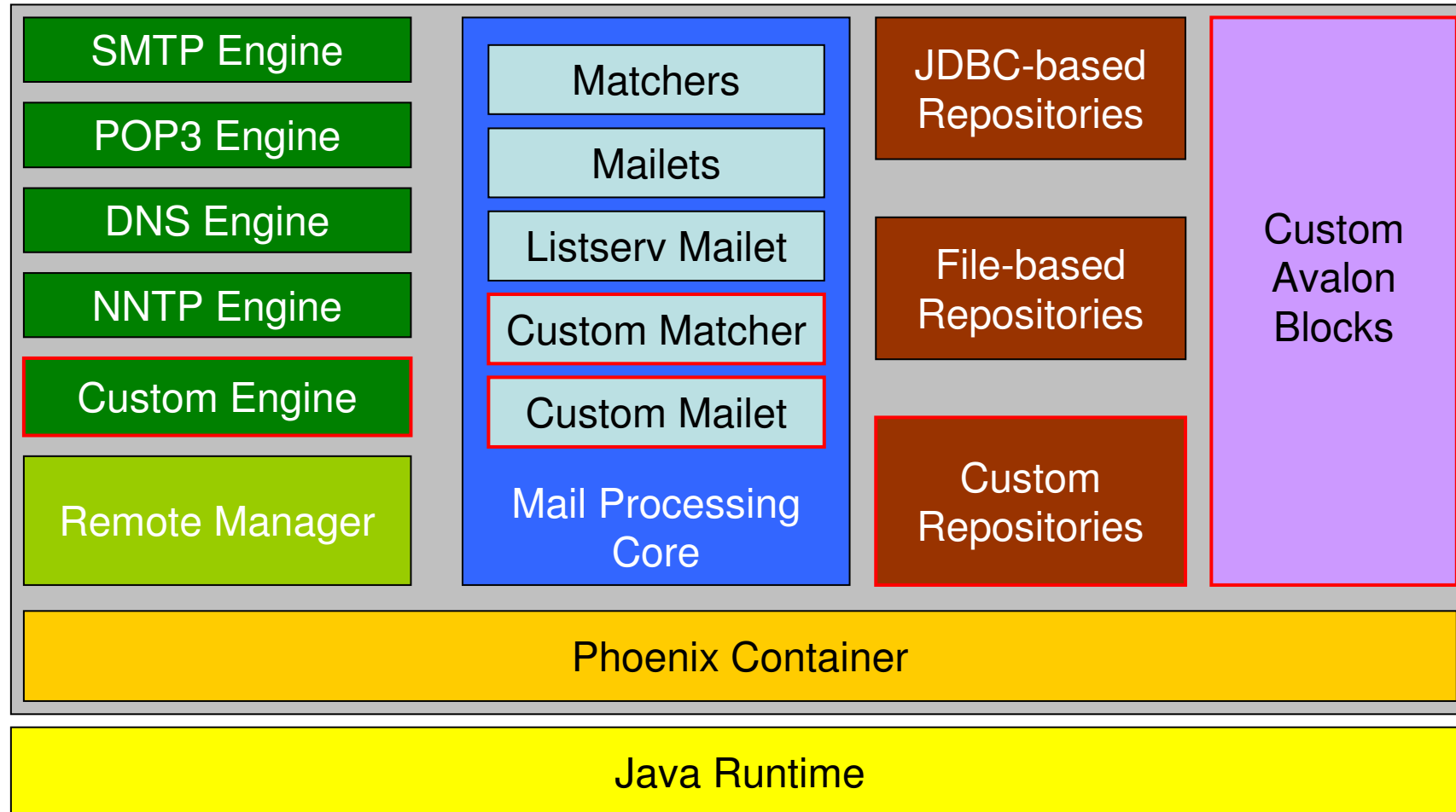
- The Apache James project was incepted in 1999 and became a TLP in late 2002
- “Java Apache Mail Enterprise Server”
- 100% pure Java, JDK 1.3 or better, JavaMail API
- Supports SMTP, POP3, NNTP and fetching messages from other mail servers
- JDBC and JNDI for Storage and Users

# James Overview

---

- component-based, using Apache Avalon
- container based, using Apache Phoenix
- runs standalone (from command line)
- can be embedded (e.g. in JBoss)

# James Architecture



# James vs. Avalon

---

- Last Release June 15, 2004: 2.2.0
- Apache Avalon closed in 2004, removing a lot of documentation
- Apache Phoenix development ceased, (resurrected as Loom @ Codehaus)
- Current release version (2.2.0) uses some „well hidden“ Avalon Code

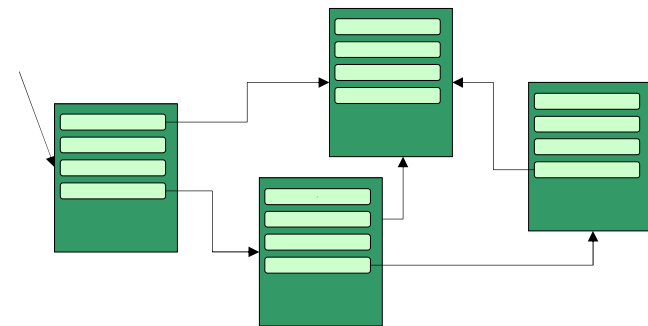
# James applications

---

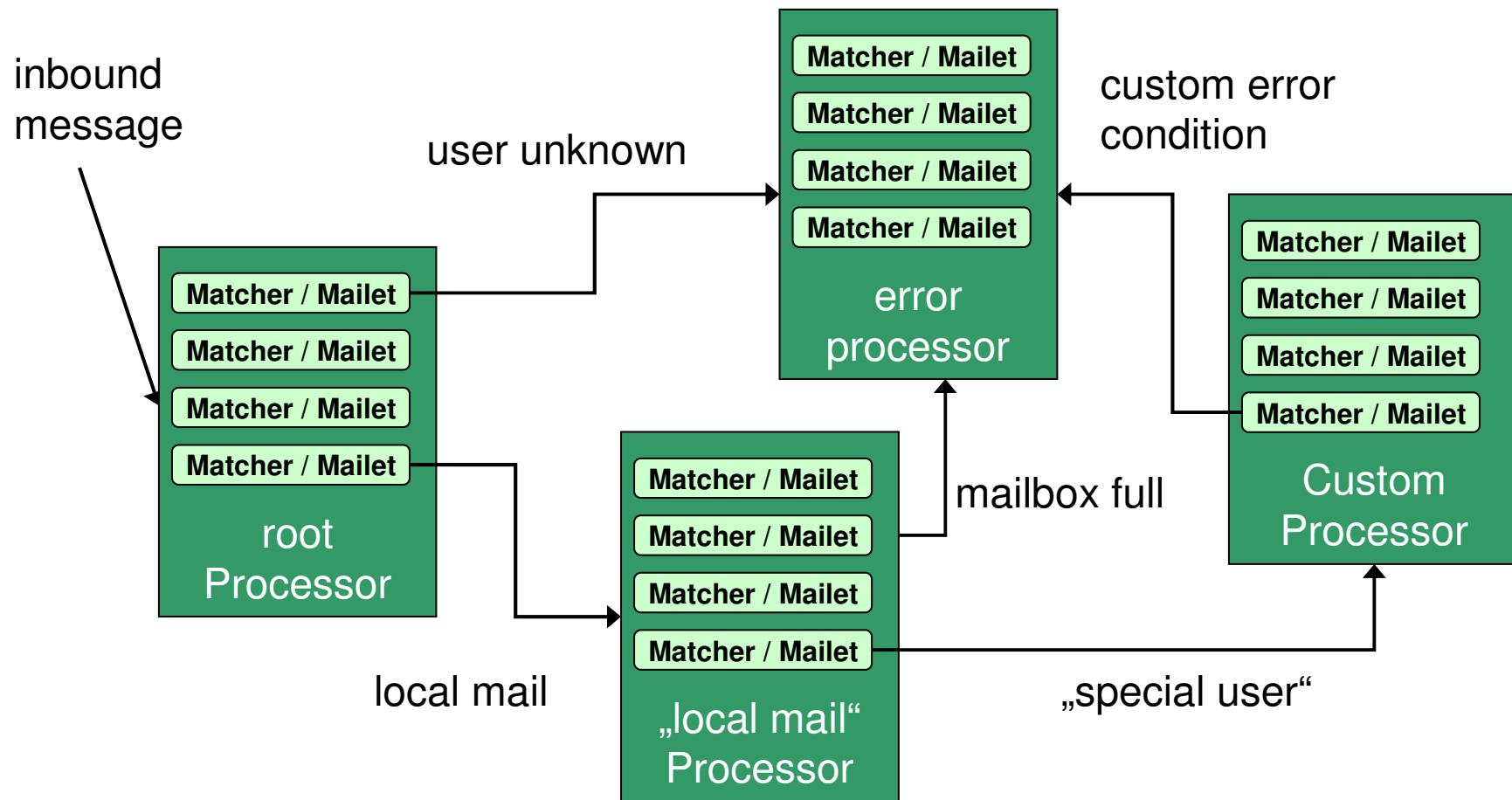
- Just another SMTP / POP3 server?
- James tries to be the „Tomcat for mail applications“
- Multi-purpose mail application container
- Embedded extension mechanisms
  - Mailets and Matchers
  - Custom code
  - Avalon Blocks

# James Message processing

- mail goes through a „tree of processors“
- Each processor contains one or more Matcher/Mailet pairs
- Messages can be split for different recipients (e.g. unknown users)
- Processor-to-Processor forwarding with special Mailet (**ToProcessor**)



# Message processing tree



# James Demo

---

- Starting and stopping from command line
- Adding an user through Admin console
- Sending and receiving mail

# Mailet/Matcher API

---

- Mailet (Mail Servlet) & Matcher
- They always come in pairs
- Matcher – Message selection code
- Mailet – Message processing code
- Released API at 1.0 level, a second revision is planned

# Matcher API

---

```
public interface Matcher {  
    void init(MatcherConfig cfg);  
    void destroy();  
    Collection match(Mail mail);  
  
    MatcherConfig getMatcherConfig();  
    String getMatcherInfo();  
  
};
```

# Included Matchers

- **Matcher Examples**
  - **All** – match all messages
  - **HasAttachment** – multipart/mixed messages
  - **HasHeader** – finds a header included
  - **HostIs, HostIsLocal** – matches hostname
  - **RecipientIs, UserIs** – recipient name
- James distribution contains more than 25 „ready to use“ matchers

# Mailet API

---

```
public interface Mailet {  
    void init(MailetConfig cfg);  
    void destroy();  
    void service(Mail mail);  
  
    MailetConfig getMailetConfig();  
    String getMailetInfo();  
  
};
```

# Included Mailables

---

- **Mailet examples**
  - **AddHeader, AddFooter** – modify contents
  - **Forward** – send message to recipients
  - **ToProcessor** – continue with other processor
  - **LocalDelivery, RemoteDelivery**
  - **JDBCALias** – Mail aliasing through database
- James contains ~ 20 „ready to use“  
Mailables

# Writing a custom Maillet

---

Task: „*The mail system should answer every mail to a local user with a confirmation mail*“

- **Matcher: `RecipientIsLocal`**
- **Custom Maillet**

# Writing a custom Maillet

```
public class NotifyMaillet
    extends GenericMaillet
    implements Maillet
{
    public void service(Mail mail) {
        MailAddress sender    = mail.getSender();
        Collection rcpts      = mail.getRecipients();
        MailletContext context = getMailletContext();

        for (Iterator it = rcpts.iterator(); it.hasNext(); ) {

            MailAddress rcpt = (MailAddress) it.next();
            MimeMessage msg = createNotifyMsg(sender, rcpt);
            context.sendMail(msg);
        }
    }
}
```

# Configuring the Maillet

---

config.xml (James main configuration)

[...]

```
<processor name="root">
```

[...]

```
<mailet match="RecipientIsLocal"  
    class="de.intermeta.james.mailet.NotifyMailet"/>
```

[...]

```
</processor>
```

[...]

---

# Mailet Demo

---

# Extending James

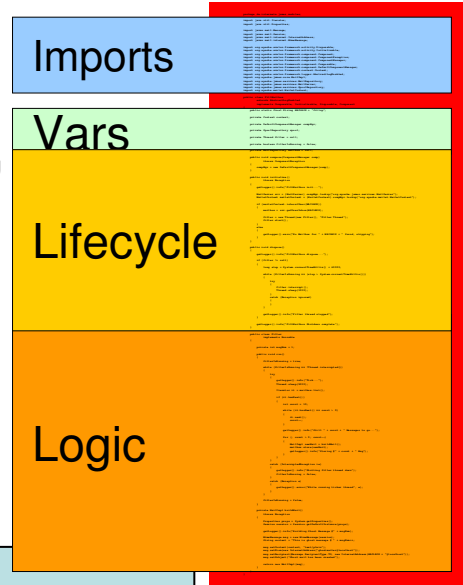
---

Task: *„If the mailbox is not empty, it should always contain at least ten messages“*

- Extension through Avalon block
- Managed and connected to James by the Phoenix container
- Must be deployed as Jar  
(Phoenix won't load it as a class file)

# A custom James Extension

- 200 Lines of Code
- **Composable, Initializable, Disposable** Lifecycle
- Access to James through `assembly.xml` configuration



```
<block name="fillmailbox"
  class="de.intermeta.james.modules.FillMailbox">
  <provide name="James"
    role="org.apache.mailet.MailetContext"/>
  <provide name="James"
    role="org.apache.james.services.MailServer"/>
</block>
```

---

# Extension Demo

---

# James future

---

- James V3 is under development
  - IMAP support
  - better Spam Filtering capabilities
  - Fast fail support for SMTP
- Full container independence is discussed
- Continuous development, though at a leisurely pace
- Noel promised a “release this month!”

# (My) James Wish list

---

- IMAP Support
- RSS protocol engine (in and out)
- Apache Project and no web engine?
- Instant Messaging gateway?
- Standards compliant STARTTLS
- Support for Non-Phoenix containers (Loom, Excalibur)
- (better) J2EE integration

# Where to go from here?

- Apache James Homepage

- <http://james.apache.org/>

- This talk (slides & code)

- <http://henning.schmiedehausen.org/james/>

- James Resources

- <http://www.java201.com/resources/browse/31-all.html>

---

Any questions?

---

---

Thanks for your attention!

---