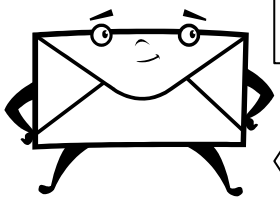
Java: Elektronische Post

Ziele

- Verstehen welche Protokolle zum Senden und Empfangen von Nachrichten verwendet werden.
- Authentifizierungs- und Sicherungsarten der Übermittlung unterscheiden und einsetzen können.
- Reine Text- und HTML-Nachrichten verschicken können.
- Nachrichten empfangen können und dabei die einzelnen Teile der Nachricht auslesen können.
- Nachrichten am Server löschen können.
- In den Übungen werden Netzwerk- und Thread-Programmierung mit den Inhalten dieses Kapitels kombiniert.

Konzept



Simple Mail Transfer Protocol (SMTP)

Post Office Protocol (POP)

Multipurpose Internet Mail Extensions (MIME)

- Ä, å, 德, ض
- Audio, Bilder, Videos

Codierung der Binärdateien in ASCII

Einstellungsmöglichkeiten

- Postein- und -ausgangsserver erfordert Authentifizierung
  - Eigener Benutzername/Passwort oder gleiche Einstellungen wie Posteingangsserver (SMTP-Auth)
  - Vor dem Senden bei Posteingangsserver anmelden (SMTP-After-POP)[3)]
- Postein- und -ausgangsserver erfordern verschlüsselte Verbindung (SSL-Verschlüsselung)[1)]
  ACHTUNG: Aber nur vom Client zum Server
- Anmeldung mithilfe der gesicherten Kennwortauthentifizierung (SPA) erforderlich[2)]
  Benutzername, Passwort verschlüsselt vom Client zum Server geschützt
- Nachrichten am Server belassen

Internet Message Access Protocol (IMAP)

- Sowohl für Senden als auch Empfangen einsetzbar
- Hierarchische Mailboxen
- Belässt Nachrichten am Server, verwaltet diese dort
- SSL-Verschlüsselung

Notwendige Zutaten

Die JavaMail API

`Javamail-1_4.zip` enthält Java-Archiv `mail.jar`
http://java.sun.com/products/javamail/

JavaBeans Activation Framework

`jaf-1_0_2-upd2.zip` enthält Java-Archiv `activation.jar`
http://java.sun.com/products/javabeans/glasgow/jaf.html

Nachrichten mittels SMTP senden

Festlegen der Verbindungseigenschaften, Authentifizierungs- und Verschlüsselungsart	`Properties`, `Authenticator`
⬇	
Erstellen einer Benutzersitzung	`Session`
⬇	
Erstellen der Nachricht • Absender und Empfänger festlegen `setFrom()`, `setRecipient()` • Betreff festlegen `setSubject()` • Inhalttyp und Inhalt festlegen `setContent()`	`MimeMessage` `InternetAddress`
⬇	
Versenden der Nachricht	`Transport`

```
import javax.mail.*;
…
// Falls notwendig vorher mit Posteingangsserver verbinden[3)]
...
// Festlegen der Verbindungseigenschaften, Authentifizierungs- und
// Verschlüsselungsart
Properties properties = new Properties();
properties.setProperty("mail.smtp.host", "smtp.test.net");
properties.setProperty("mail.smtp.user", "resi");
properties.setProperty("mail.smtp.password", "ruckzuck");
// Anmeldung mithilfe der gesicherten Kennwortauthentifizierung
// (SPA) erforderlich[2)]
Authenticator authenticator = new Authenticator() {
  protected PasswordAuthentication getPasswordAuthentication(){
    return new PasswordAuthentication("resi","ruckzuck");
  } };
// Postausgangsserver erfordert verschlüsselte Verbindung[1)]
properties.setProperty("mail.smtp.socketFactory.class",
  "javax.net.ssl.SSLSocketFactory");
properties.setProperty("mail.smtp.port", "25");
properties.setProperty("mail.smtp.socketFactory.port", "25");
properties.setProperty("mail.smtp.auth", "true");
// Erstellen der Benutzersitzung
Session session = Session.getInstance(properties,authenticator);
// Erstellen der Nachricht
MimeMessage message = new MimeMessage(session);
try { message.setFrom(new InternetAddress("resi@test.net"));
  message.setRecipient(MimeMessage.RecipientType.TO,
    new InternetAddress("sepp@mail.net"));
  message.setSentDate(new Date());
  message.setSubject("Urlaub wohin?");
  message.setContent(
    "<html><head></head><body>Hallo Sepp, …</body></html>",
    "text/html; charset=ISO-8859-1");
  // Versenden der Nachricht
  Transport.send(message);
} catch (AddressException e) {
  // Ausgelöst vom Konstruktor InternetAddress()
} catch (MessagingException e) {
  // Ausgelöst von setFrom() oder von send()
}
```

Nachrichtentypen

`message.isMimeType("text/*")`

`text/plain`	ASCII-Textnachricht ohne Anlage
`text/html`	HTML-Nachricht
`multipart/alternative`	HTML- oder RTF-Nachricht gemeinsam mit gleicher ASCII-Textnachricht verschickt (Microsoft Outlook)
`multipart/mixed`	Nachricht mit Anlagen
`multipart/report`	Statusnachricht z. B. bei fehlgeschlagener Übermittlung

Nachrichten mittels POP abrufen

Festlegen der Verbindungseigenschaften, Authentifizierungs- und Verschlüsselungsart — `Properties, Authenticator`

↓

Erstellen einer Benutzersitzung — `Session`

↓

Festlegen des Übertragungsprotokolls (POP oder IMAP) und Verbinden — `Store`

↓

Öffnen des Posteingangsordner zum Lesen und Abfrage der enthaltenen Nachrichtenanzahl — `Folder`

↓

Lesen der Nachrichten der Reihe nach — `Message`

- Nachrichtennummer `getMessageNumber()`
- Absender `getFrom()[0].toString()`
- Empfänger `getHeader("To")[0]`
- Sendedatum `getSentDate()`
- Betreff `getSubject()`

```
// Festlegen der Verbindungseigenschaften, Authentifizierungs- und
// Verschlüsselungsart
// Erstellen der Benutzersitzung
…
properties.setProperty("mail.pop3.host", "pop3.test.net");
…
// Statusmeldungen werden in die Standardausgabe geschrieben
session.setDebug(true);
Store store=null; Folder folder=null; BufferedReader in=null;
try {
  // Festlegen des Übertragungsprotokolls (POP oder IMAP)
  // und Verbinden
  store = session.getStore("pop3");
  store.connect();
  // Öffnen des Posteingangsordner zum Lesen und Abfrage der
  // enthaltenen Nachrichtenanzahl
  folder = store.getFolder("INBOX");
  folder.open(Folder.READ_ONLY);
  // Lesen der Nachrichten der Reihe nach
  for (int i = 1; i <= folder.getMessageCount(); i++) {
    Message message = folder.getMessage(i);
    System.out.println(message.getMessageNumber());
    System.out.println(message.getFrom()[0].toString());
    System.out.println(message.getHeader("To")[0]);
    System.out.println(message.getSentDate());
    System.out.println(message.getSubject());
    System.out.println(
      new ContentType(message.getContentType()).getBaseType());
    // Lesen der Nachrichteninhalte
    (siehe hinten)
  }
} catch (NoSuchProviderException e) {
  // Ausgelöst von getStore()
} catch (MessagingException e) {
  // Ausgelöst von connect()
} catch (IOException e) {
  // Ausgelöst von getContent() und readLine()
} finally {
  try { in.close(); } catch (Exception e) { ; }
  // Schließen des Ordners
  try { folder.close(false); } catch (Exception e) { ; }
  try { store.close(); } catch (Exception e) { ; }
}
```

Lesen der Nachrichteninhalte

- Nachrichtentyp ist `text/plain` oder `text/html`
  ➔ Auslesen des Inhaltes mit `getContent()`

- Nachrichtentyp ist `multipart/*`
  ➔ Auslesen der Nachrichtentexte
  ➔ Auslesen der Anlagen

`Multipart`, `Part`, `MimeBodyPart`

```
if (message.isMimeType("text/*")) {
  // ASCII- oder HTML-Nachricht ohne Anlage
  System.out.println((String)message.getContent());
}
if (message.isMimeType("multipart/*")) {
  Multipart multipart = (Multipart)message.getContent();
  int anzahlAnlagen = 0;
  for (int j = 0; j < multipart.getCount(); j++) {
    Part part = multipart.getBodyPart(j);
    if (part.getDisposition() == null) {
      // Nachrichtentext
      MimeBodyPart mimeBodyPart = (MimeBodyPart)part;
      System.out.println(mimeBodyPart.getContentType());
      in = new BufferedReader(
        new InputStreamReader(mimeBodyPart.getInputStream()));
      String inhalt = "";
      for (String line; (line = in.readLine()) != null; )
        inhalt += line;
      System.out.println(inhalt);
      in.close();
    }
    if (part.getDisposition() != null &&
      (part.getDisposition().equals(Part.ATTACHMENT) ||
      part.getDisposition().equals(Part.INLINE))) {
      // Anhang erkannt
      anzahlAnlagen++;
    }
  }
  System.out.println(anzahlAnlagen);
}
```

Mails löschen

```
// Inbox kann verändert werden
folder.open(Folder.READ_WRITE);
for (int i = 1; i <= folder.getMessageCount(); i++) {
     Message message = folder.getMessage(i);
     message.setFlag(Flags.Flag.DELETED, true);
}
// Im Ordner durchgeführten Änderungen werden wirksam
folder.close(true);
```