Spedizione email in Java

Web Programming

AA 2013-2014 Gino Perna

JavaMail

JavaMail implementano il supporto generico alla posta elettronica, tramite un insieme di API ben disegnate, espressive ed espandibili. Tramite l'architettura a provider, è possibile supportare diversi standard; SUN infatti offre i provider per i protocolli SMTP, POP3 ed IMAP

Introduzione - 1

- All'interno delle funzionalità offerte dalla piattaforma Java, si trova anche il supporto alla posta elettronica, realizzato tramite le API JavaMail. Questa libreria offre le funzionalità necessarie ad implementare applicazioni di email completamente funzionanti, come Outlook o Thunderbird.
- Non essendo realizzata con componenti visuali - la creazione di una interfaccia utente è interamente a carico dello sviluppatore - è utilizzabile sia in applicazioni desktop che sul lato server, ad esempio per implementare una WebMail.

Introduzione - 2

- Come noto, esistono diversi protocolli che consentono ad una applicazione di realizzare la posta elettronica. I principali sono tre:
  - SMTP (Simple Mail Transfer Protocol). Consente l'invio di un messaggio di posta;
  - POP (Post Office Protocol). Protocollo comunemente utilizzato per la ricezione di messaggi di posta da un server;
  - IMAP (Internet Message Access Protocol). Protocollo di posta orientato al server e di tipo drop-and-store, più evoluto rispetto a POP e dotato di funzionalità di gestione del server che facilita la gestione della posta off-line.

Il protocollo SMTP (spedire mail)

- After establishing a connection between the sender (the client) and the receiver (the server), the following is a legal SMTP session. In the following conversation, everything sent by the client is prefaced with C: and everything sent by the server is prefaced with S:. On most computer systems, a connection can be established using the telnet command on the client machine, for example.

```
telnet mail.unitn.it 25
```

which opens a TCP connection from the sending machine to the MTA listening on port 25 on host mail.unitn.it.

Il protocollo SMTP – tipica sessione

Inizio con: `telnet mail.unitn.it 25`
S: 220 www.example.com ESMTP Postfix
C: HELO mydomain.com
S: 250 Hello mydomain.com
C: MAIL FROM:<sender@mydomain.com>
S: 250 Ok
C: RCPT TO:<friend@example.com>
S: 250 Ok
C: DATA
S: 354 End data with <CR><LF>.<CR><LF>
C: Subject: test message
C: From: sender@mydomain.com
C: To: friend@example.com
C:
C: Hello,
C: This is a test.
C: Goodbye.
C: .
S: 250 Ok: queued as 12345
C: QUIT
S: 221 Bye

Le Piattaforme

- Il package JavaMail è ottenibile in due diversi modi, per prima cosa è possibile scaricare l'implementazione di riferimento e la documentazione dal sito ufficiale ospitato da SUN, che si trova all'indirizzo http://www.oracle.com/technetwork/java/javamail/index.html.
- Il package è utilizzabile all'interno di un ambiente J2SE a partire dalla versione 1.1.7, e richiede anche il JavaBeans Activation Framework (http://www.oracle.com/technetwork/java/javase/index-jsp-136939.html) un package che ha lo scopo di abilitare la gestione, all'interno di applicazioni Java, di blocchi di dati di tipo arbitrario, incapsulandoli ed accedendovi, e di scoprire le operazioni effettuabili su di essi. Esempi sono le immagini JPEG o GIF, oppure documenti Word od Excel;

Le Piattaforme - cont 2

- l'Activation Framework è utilizzato all'interno di JavaMail per supportare gli allegati binari ai messaggi di posta ed anche per scoprire a runtime il tipo di dato memorizzato all'interno di un blocco binario.
- Se invece si sta utilizzando la piattaforma Java2 Enterprise Edition (J2EE), a partire dalla versione 1.3, non è necessario scaricare né l'una, né l'altra, in quanto i servizi legati alla posta elettronica sono parte fondamentale e integrante della piattaforma enterprise.
- Quindi per le applicazioni con tomcat non e' necessario scaricare i jar

Invio di un Messaggio

- Per illustrare con semplicità un utilizzo pratico di JavaMail, si osservi l'esempio presentato oltre, che implementa un semplice programma di invio di un messaggio di posta, forse il più semplice realizzabile con queste API.
- La prima cosa che si nota è la presenza dell'importazione dei package javax.mail e javax.mail.internet, due componenti principali di JavaMail.
- All'interno del primo package si trovano le classi principali, come Session e Message, utilizzati nel programma per l'invio del messaggio; la prima classe - di tipo final - implementa una sessione di posta ed ha lo scopo di raccogliere proprietà e configurazioni e di fornire le sessioni alle classi client.

Invio di un Messaggio - 2

- Le sessioni possono essere di due tipi: condivise o meno; nel primo caso una unica sessione viene utilizzata da parte di più sezioni del programma, tipicamente in una applicazione desktop. In ambiente server è invece preferibile utilizzare sessioni non condivise, ottenute in modo similare a come in JDBC si ottiene una connessione da un DataSource.
- Per ottenere una sessione condivisa è necessario utilizzare il metodo Session.getDefaultInstance() a cui è indispensabile passare un oggetto Properties con i parametri di configurazione necessari ad operare con il protocollo di posta.

Invio di un Messaggio - 3

- L'unico parametro indispensabile in questo caso è **mail.smtp.host**, che indica il nome o l'indirizzo del server SMTP di invio.
- Il programma poi crea un oggetto Message di tipo MimeMessage e ne imposta le proprietà. In particolare:
  - il mittente;
  - il destinatario;
  - l'oggetto;
  - la data di invio;
  - il testo del messaggio.

Invio di un Messaggio - 4

- Questi parametri sono impostati tramite i metodi riassunti in tabella 1 che nella classe Message sono astratti, mentre in MimeMessage sono effettivamente implementati.

Metodo	Scopo
setFrom()	Imposta il mittente sotto forma di oggetto InternetAddress
setRecipients()	Aggiunge un destinatario all'elenco dei destinatari. Il tipo può essere TO, BCC o CC definiti come campi statici nella classe Message.RecipientType.
setSubject()	Imposta l'oggetto del messaggio
setSentDate()	Imposta la data di invio del messaggio
setText()	Imposta il testo del messaggio
setReplyTo()	Imposta l'indirizzo a cui rispondere
setFlag()	Imposta un flag (utilizzato ad esempio per le priorità)

Invio di un Messaggio - 5

- A questo punto è possibile inviare il messaggio utilizzando il metodo Transport.send(). Questo metodo si occuperà dunque di individuare tutti i destinatari (tramite Message.getAllRecipients()) ed ad inviargli il messaggio, utilizzando il trasporto appropriato per ciascun destinatario (alcuni di questi potrebbero non essere destinatari Internet).
- Gli indirizzi sono infatti stati rappresentati con oggetti InternetAddress (classe presente nel package javax.mail.internet), classe che rappresenta indirizzi nella forma nome@server.dominio e che estende javax.mail.Address.

Invio di un Messaggio - 6

- Il metodo send() può sollevare due eccezioni nel caso l'invio non vada a buon fine: SendFaildException e MessagingException entrambi presenti nel package javax.mail.
- Si noti che send() può eseguire un invio parziale. Se ad esempio un messaggio è indirizzato a più destinatari, chiamiamoli Mario, Giovanni, Andrea, Luca, Elisa e Maurizio nell'ordine e l'invio fallisce per Giovanni, Luca e Maurizio, l'invio a Mario, Andrea ed Elisa (chi mancava dall'elenco precedente) viene comunque fatto. Al termine delle operazioni viene sollevata una eccezione che riassume quanto successo: fornisce l'elenco dei destinatari per cui l'invio è andato a buon fine e l'elenco dei destinatari per cui l'invio non è riuscito.

Invio di un Messaggio - esempio

```
package it.gino.mail;
import java.util.*;
import javax.mail.*;
import javax.mail.internet.*;
/**
 *
 * @author Administrator
 */
public class mail01 {

    /** Creates a new instance of mail01 */
    public mail01() {
    }
```

Invio di un Messaggio - esempio

```
public static void main( String[] args ) {
  try {
    Properties props = System.getProperties();
    props.put( "mail.smtp.host", "nome_del_mio_host_smtp" );
// al posto del nome in rosso va il mio server o smarthost provider
    props.put( "mail.debug", "true" );

    Session session = Session.getDefaultInstance( props );
    Message message = new MimeMessage( session );

    InternetAddress from = new InternetAddress( "info@unitn.it" );
    InternetAddress to[] =
   InternetAddress.parse( "gino@enginsoft.it" );
```

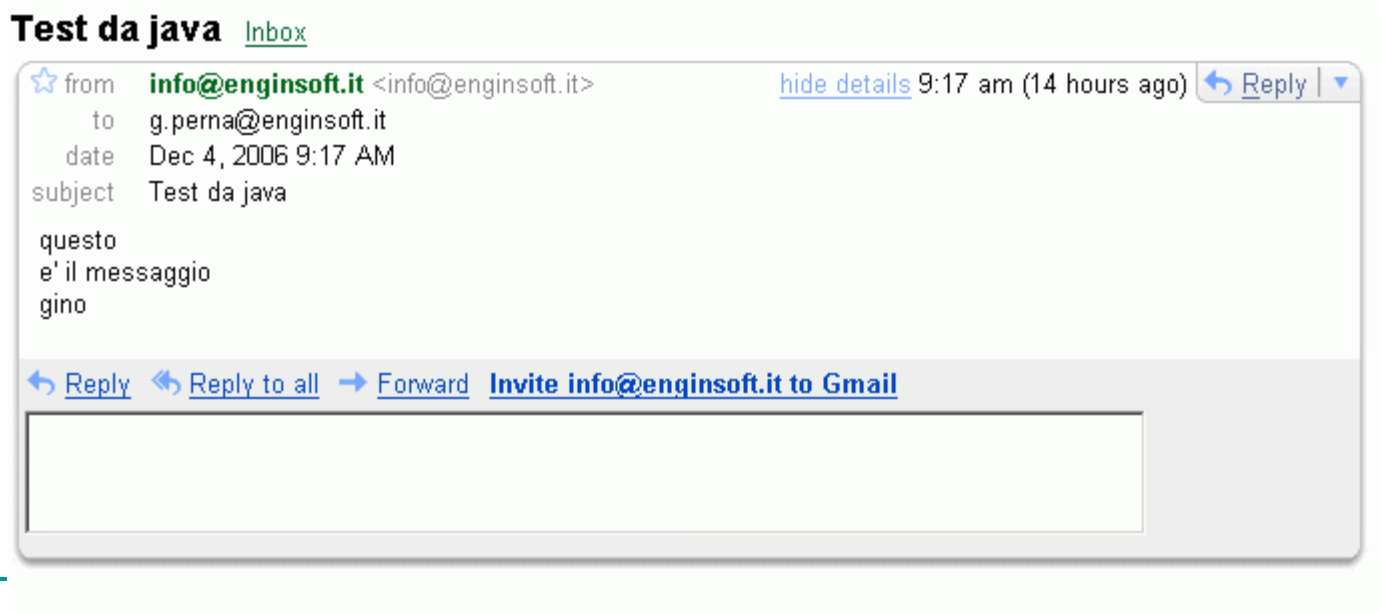
Invio di un Messaggio - esempio

```
    message.setFrom( from );
    message.setRecipients( Message.RecipientType.TO, to );
    message.setSubject( "Test da java" );
    message.setSentDate( new Date() );
    message.setText( "questo\ne' il messaggio\ngino\n" );

    Transport.send(message);
    } catch(MessagingException e) {
      e.printStackTrace(); // USARE LOG4J!!!
    }
  }
}
```

Invio di un Messaggio - esempio

- Se tutto funziona correttamente, si dovrebbe essere in grado di ricevere il messaggio ed ottenere qualcosa simile al quanto presente in figura



Cosa contiene il messaggio in realta'

```
X-Spam-Status: No, score=-2.5 required=3.5 tests=AWL,BAYES_00,NO_REAL_NAME autolearn=no
    version=3.1.0
Received: from mail4-out.unitn.it (mail4-out.unitn.it [193.205.206.45]) by poldo.enginsoft.it
    (8.12.7/8.12.7/SuSE Linux 0.6) with ESMTP id kB48I0bV012094 for <g.perna@enginsoft.it>; Mon, 4
    Dec 2006 09:18:00 +0100
Received: from mail4-out.unitn.it (unknown [127.0.0.1]) by mail4-out.unitn.it (Symantec Mail
    Security) with ESMTP id 8319D152CDA for <g.perna@enginsoft.it>; Mon,  4 Dec 2006 09:16:51
    +0100 (CET) X-AuditID: c1cdce2d-a9893bb00000763a-cb-4573d973a23a
Received: from science.unitn.it (alpha.science.unitn.it [193.205.194.23]) by mail4-out.unitn.it
    (Symantec Mail Security) with ESMTP id 54252EAD43 for <g.perna@enginsoft.it>; Mon,  4 Dec 2006
    09:16:51 +0100 (CET)
Received: from hyvals3 (gate [193.205.194.28]) by science.unitn.it (8.12.11.20060308/8.12.11)
    with ESMTP id kB48GnEF013271 for <g.perna@enginsoft.it>; Mon, 4 Dec 2006 09:16:50 +0100
Date: Mon, 4 Dec 2006 09:17:26 +0100 (CET)
From: info@enginsoft.it
To: g.perna@enginsoft.it
Message-ID: <26956691.01165220246959.JavaMail.Administrator@hyvals3>
Subject: Test da java
MIME-Version: 1.0
Content-Type: text/plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
X-Virus-Scanned: by AMaViS - amavis-milter (http://www.amavis.org/)
Resent-From: gigi@poldo.enginsoft.it


questo

e' il messaggio

gino
```

SMTP universita'

- Il server da utilizzare e' mail.unitn.it
- Questo server e' completamente loggato, si sa cosa passa, da dove a chi, il mittente e cosa spedite!
- Non GIOCATE!!!!!!!
- Accetta solo indirizzi ***from*** dal dominio unitn.it

SMTP Server

- I server SMTP normalmente sono chiusi ovvero non fanno relay (cioe' posso contattarli solo per mandare messaggi per i quali sono autorizzati a fare il delivery)
- Cosa fare per utilizzarlo da casa?
  - Google mail (se avete un account) (oppure Yahoo)
  - http://www.softstack.com/freesmtp.html free smtp server windows
  - Postfix su linux (configurare smarthost in main.cf)

SMTP Server google

- Google mette a disposizione un smtp server in TLS sulla porta 465 (la 25 criptata!)

Configuring other mail clients

You can use the following information to configure POP with many mail clients. If you encounter difficulties, we suggest contacting your mail client's customer support department for further instructions -- we're not able to provide assistance with configuring mail clients not listed here.

Incoming Mail (POP3) Server - requires SSL:	`pop.gmail.com` **Use SSL**: Yes **Port**: 995
Outgoing Mail (SMTP) Server - requires TLS:	`smtp.gmail.com` (use authentication) **Use Authentication**: Yes **Use STARTTLS**: Yes (some clients call this SSL) **Port**: 465 or 587
Account Name:	your Gmail username (including `@gmail.com`)
Email Address:	your full Gmail email address (`username@gmail.com`)
Password:	your Gmail password

Please note that if your client does not support SMTP authentication, you won't be able to send mail through your client using your Gmail address.

SMTP Server google – gli header

```
package it.gino.mail;
import java.security.Security;
import java.util.Date;
import java.util.Properties;
import javax.mail.Authenticator;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.AddressException;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;
```

SMTP Server google – connessione

```
public static void main(String[] args) throws AddressException, MessagingException {
  Security.addProvider(new com.sun.net.ssl.internal.ssl.Provider());
  final String SSL_FACTORY = "javax.net.ssl.SSLSocketFactory";
  // Get a Properties object
  Properties props = System.getProperties();
  props.setProperty("mail.smtp.host", "smtp.gmail.com");
  props.setProperty("mail.smtp.socketFactory.class", SSL_FACTORY);
  props.setProperty("mail.smtp.socketFactory.fallback", "false");
  props.setProperty("mail.smtp.port", "465");
  props.setProperty("mail.smtp.socketFactory.port", "465");
  props.put("mail.smtp.auth", "true");
      props.put( "mail.debug", "true" );
```

SMTP Server google – connessione

```
final String username = "gigi@gmail.com";
final String password = "**********";
Session session = Session.getDefaultInstance(props, new Authenticator(){
    protected PasswordAuthentication getPasswordAuthentication() {
        return new PasswordAuthentication(username, password);
    }});

     // – Create a new message –
Message msg = new MimeMessage(session);

// – Set the FROM and TO fields –
msg.setFrom(new InternetAddress(username + ""));
msg.setRecipients(Message.RecipientType.TO,
  InternetAddress.parse("gino@enginsoft.it",false));
msg.setSubject("Hello da gmail");
msg.setText("come stai "+ new Date());
msg.setSentDate(new Date());
Transport.send(msg);
```

SMTP Server yahoo – connessione

- Analoga configurazione come gmail
  - smtp.mail.yahoo.com
  - mail.smtps.auth=true
  - Utente, password
  - user@yahoo.com

Here are the basic server settings for Yahoo! Mail:

Incoming Mail (POP3) Server:	pop.mail.yahoo.com (Use SSL, port: 995)
Outgoing Mail (SMTP) Server:	smtp.mail.yahoo.com (Use SSL, port: 465, use authentication)
Account Name/Login Name:	Your Yahoo! Mail ID (your email address without the "@yahoo.com")
Email Address:	Your Yahoo! Mail address (e.g., user@yahoo.com)
Password:	Your Yahoo! Mail password

SMTP Server sull porta 587 (unitn da fuori universita')

```
Analoga configurazione come gmail
import="java.util.*;
import="java.io.*;
import="javax.mail.*;
import="javax.event.*;
import="javax.mail.internet.*;
import="java.security.Security.*;
import="javax.activation.*;
Properties props = new Properties();
props.put("mail.smtp.host", "smtp_server_universita_da_fuori");
props.put("mail.smtp.port","587");
props.put("mail.debug", "true");
props.put("mail.smtp.debug", "true");
props.put ("mail.smtp.starttls.enable", "true");
props.put("mail.transport.protocol", "smtp");
props.put("mail.smtp.socketFactory.port", "587");
props.put("mail.smtp.socketFactory.class", "javax.net.ssl.SSLSocketFactory");
props.put("mail.smtp.socketFactory.fallback", "false");
java.security.Security.addProvider(new com.sun.net.ssl.internal.ssl.Provider());
Authenticator auth = new javax.mail.Authenticator() {
protected PasswordAuthentication getPasswordAuthentication(){
return new PasswordAuthentication("myUsername", "myPassword");
}
};
```

SMTP Server 587 (unitn)

```
#
Session s = Session.getInstance(props,auth);
MimeMessage message = new MimeMessage(s);
InternetAddress from = new InternetAddress("myFriendAddress");
message.setFrom(from);
InternetAddress to = new InternetAddress("myAddress");
message.addRecipient(Message.RecipientType.TO, to);
message.setSubject("Test from JavaMail.");
message.setText("Hello from JavaMail!");
message.saveChanges();
Transport transport = s.getTransport("smtp");
transport.connect("smtp-auth.bris.ac.uk","myUsername","myPassword");
transport.sendMessage(message,message.getAllRecipients());
transport.close();
```

Messaggi Multipart

- In precedenza è stato utilizzato il metodo setText() per impostare il contenuto di un messaggio che effettua alcune operazioni per impostare il tipo di contenuto (MIME) a text/plain.
- JavaMail supporta infatti messaggi di tipo multipart, composti cioè da contenuti multipli di tipi eterogenei. Un messaggio di posta potrebbe essere composto infatti da un contenuto in testo semplice e da un contenuto HTML; spesso questi messaggi di tipo ricco possono essere visualizzati in un modo o nell'altro in funzione delle capacità dei singoli programmi di client email.

Messaggi Multipart

- Un altro uso dei messaggi multipart è quello legato agli allegati; quando si allega un file di Word o Excel, un file compresso od un'immagine ad un messaggio di posta, il client crea un messaggio multipart creando una parte per ciascun allegato associando alla parte il tipo MIME dell'archivio (p.e. image/gif) ed il contenuto binario del file.
- E' in questo momento che entra in gioco il JavaBeans Activation Framework, che offre appunto il supporto richiesto a queste operazioni con dati binari, permettendo addirittura l'invio di oggetti Java serializzati!

Messaggi Multipart: invio allegati

- Per inviare un messaggio multipart in JavaMail le operazioni di connessione al provider ed inizializzazione del messaggio sono le medesime di quelle viste nel listato precedente; le novità intervengono infatti solo nella composizione del corpo da inviare.
- E' necessario infatti utilizzare un oggetto Multipart, che consente di memorizzare diverse parti di corpo (oggetti BodyPart) e che fornisce metodi per impostarle e ritornarle; anche in questo caso Multipart è una classe astratta che demanda alle sottoclassi l'effettiva realizzazione di parte dei suoi servizi; per la posta Internet è disponibile la classe MimeMultipart che si basa appunto sulle convenzioni MIME.

Messaggi Multipart: invio allegati

- La classe MimeMultipart definisce diversi sottotipi, conformi alle specifiche MIME, che possono essere "mixed", "alternative", "related" e così via; il tipo principale è invece "multipart". Questo approccio consente una più agevole corrispondenza nell'Activation Framework, disaccoppiando il processo di fornitura dei gestori delle diverse parti dalle API JavaMail.
- Nel listato seguente è presente il codice completo di un programma di prova che esegue un invio di un messaggio con allegato, sviluppato a partire dalla classe Invio presentata nel listato 1.

Messaggi Multipart: invio allegati

- Come si nota osservando il codice, viene per prima cosa istanziato un oggetto MimeMultipart, che ospiterà il corpo del messaggio e subito dopo un oggetto MimeBodyPart (sottoclasse di BodyPart) che fornisce una implementazione MIME di una singola parte. Nell'esempio vengono utilizzate due parti, una per il testo del messaggio ed un'altra per l'allegato Word. Se nel primo caso è sufficiente chiamare il metodo setText() per impostare il contenuto della parte, che è solo testuale, nel secondo caso è necessario utilizzare le classi di Activation Framework per ottenere un blocco dati binario che contenga il documento Word.

Messaggi Multipart: invio allegati

- In particolare, viene creato un nuovo DataSource a partire dal file Documento.doc che viene letto tramite una istanza di DataHandler:

```
DataSource source = new FileDataSource( "Documento.doc" );
messageBodyPart = new MimeBodyPart();
messageBodyPart.setDataHandler( new DataHandler(source) );
messageBodyPart.setFileName( "Allegato.doc" );
```

- I due oggetti MimeBodyPart vengono poi aggiunti all'oggetto MimeMultipart tramite il metodo **`addBodyPart()`**.

Messaggi Multipart: invio allegati

```
import java.util.*;

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

public class InvioMultipart {

public static void main( String[] args ) {
try {
Properties props = System.getProperties();
props.put( "mail.smtp.host", args[0] );

Session session = Session.getDefaultInstance( props );
Message message = new MimeMessage( session );

InternetAddress from = new InternetAddress( "max@bigatti.it" );
InternetAddress to[] = InternetAddress.parse( "gigi@bagigi.it" );
```

Messaggi Multipart: invio allegati

```
message.setFrom( from );
message.setRecipients( Message.RecipientType.TO, to );
message.setSubject( "[Mokabyte] - Messaggio multipart di prova" );
message.setSentDate( new Date() );

Multipart multipart = new MimeMultipart();

//crea la parte testuale
BodyPart messageBodyPart1 = new MimeBodyPart();
messageBodyPart1.setText("Invio in allegato un documento Word");

//crea l'allegato Word
DataSource source = new FileDataSource( "Documento.doc" );
BodyPart messageBodyPart2 = new MimeBodyPart();
messageBodyPart2.setDataHandler( new DataHandler(source) );
messageBodyPart2.setFileName( "Documento.doc" );
```

Messaggi Multipart: invio allegati

```
//aggiunge le parti all'oggetto multipart
multipart.addBodyPart( messageBodyPart1 );
multipart.addBodyPart( messageBodyPart2 );

//imposta come contenuto del messaggio l'oggetto multipart
message.setContent(multipart);

Transport.send(message);
} catch(MessagingException e) {
e.printStackTrace();
}
}
}
```

Mail da JSP – il form

```
/*mailF.htm  --------------*/
 <html>
 <body>
   <form action="sendMail.jsp" method="post">
     <table cellspacing="2" cellpadding="2"
    border="1">
       <tr>
         <td>To:</td>
         <td>
           <input type="text" name="to"
    size="30" maxlength="30">
         </td>
       </tr>
       <tr>
         <td>From:</td>
         <td>
           <input type="text" name="from"
    size="30" maxlength="30">
         </td>
 </tr>
    <tr>
      <td>Subject</td>
      <td>
        <input type="text"
name="subject" size="30" maxlength="30">
      </td>
    </tr>
    <tr>
      <td colspan="2">
        <textarea cols="40" rows="10"
name="body"></textarea>
      </td>
    </tr>
    <tr>
      <td>
        <input type="submit"
name="submit" value="Submit">
        <input type="Reset">
      </td>
    </tr>
  </table>   </form> </body> </html>
```

Mail da JSP – jsp

```
<html>    <head>      <title>JSP JavaMail Example </title>    </head>
 <body>

 <%@ page import="java.util.*" %>
 <%@ page import="javax.mail.*" %>
 <%@ page import="javax.mail.internet.*" %>
 <%@ page import="javax.activation.*" %>

 <%
     String host = "mail.rex.com"; //mettere il vostro
     String to = request.getParameter("to");
     String from = request.getParameter("from");
     String subject = request.getParameter("subject");
     String messageText = request.getParameter("body");
     boolean sessionDebug = false;

     Properties props = System.getProperties();
     props.put("mail.host", host);
     props.put("mail.transport.protocol", "smtp");
```

Mail da JSP – jsp - cont

```
    Session mailSession = Session.getDefaultInstance(props, null);
     mailSession.setDebug(sessionDebug);
     Message msg = new MimeMessage(mailSession);

     msg.setFrom(new InternetAddress(from));
     InternetAddress[] address = {new InternetAddress(to)};
     msg.setRecipients(Message.RecipientType.TO, address);
     msg.setSubject(subject);
     msg.setSentDate(new Date());
     msg.setText(messageText);

     Transport.send(msg);

     out.println("Mail was sent to " + to);
     out.println(" from " + from);
     out.println(" using host " + host + ".");

%>
    </table>
  </body>
</html>
```

Esercizi

- Testare l'invio dal proprio PC
- Automatizzare una servlet/JSP/procedura qualsiasi vista le scorse lezioni con l'invio di una mail con allegato.

Bibliografia

[1] JavaMail QuickStart - http://www.javaworld.com/javaworld/jw-10-2001/jw-1026-javamail.html

[2] Managing ezines with JavaMail and XSLT, Part 2 - http://www-106.ibm.com/developerworks/xml/library/x-xmlist2/?open&l=842%2Ct=gr%2Cp=JavaMail2

[3] Fundamentals of the JavaMail API short course - http://developer.java.sun.com/developer/onlineTraining/JavaMail/contents.html

[4] JavaMail Homepage - http://java.sun.com/products/javamail/

[5] JavaMail API documentation - http://java.sun.com/products/javamail/javadocs/index.html

[6] JavaBeans Activation Framework - http://java.sun.com/products/javabeans/glasgow/jaf.html