TP Partie 1 : MIME, URL, Mail Server, et http

Documentation :

1. http://www.tutorialspoint.com/javamail_api/
2. http://www.jmdoudoux.fr/java/dej/chap-javamail.htm

1. MIME depuis JAVA

Pour récupérer le MIME d'un document, il existe plusieurs solutions (plus ou moins efficace). La liste des solutions est renseignée dans le document "Get the Mime Type from a File - Real's Java How-to.html" qui se trouve dans le dossier web du TP.
Tester le type MIME d'un fichier de votre choix puis d'un fichier html en ligne avec le code suivant (essaies de le commenter d'abord) :

```
import java.net.FileNameMap;
import java.net.URLConnection;

public class FileUtils {

 public static String getMimeType(String fileUrl)
         throws java.io.IOException
 {
      FileNameMap fileNameMap = URLConnection.getFileNameMap();
      String type = fileNameMap.getContentTypeFor(fileUrl);

      return type;
 }
m
 public static void main(String args[]) throws Exception {

  System.out.println(FileUtils.getMimeType("file://c:/temp/test.txt
  "));
      // output : text/plain
    }
  }
```

2. Création d'un URL en java

L'objectif de cet exercice est la réalisation un simple programme permettant d'ouvrir un URL avec détection d'erreur (e.g http://www.polytech2go.fr/index.htm)

Corriger et tester le code suivant (bien sûr il faut créer la classe) :

```
try {
        URL monUrl = new URL("http://www.polytech2go.fr/index.htm");

} catch (MalformedURLException e) {
        // exception handler code here
}

//analyser un URL
```

```
        System.out.println("protocol = " + aURL.getProtocol());
        System.out.println("host = " + aURL.getHost());
        System.out.println("filename = " + aURL.getFile());
        System.out.println("port = " + aURL.getPort());
        System.out.println("ref = " + aURL.getRef());

        //lire directement le contenu d'un URL ouvert dans l'application
        //(code HTML)

        BufferedReader in = new BufferedReader(
                                new InputStreamReader(monUrl.openStream()));

        String inputLine;

        while ((inputLine = in.readLine()) != null)
                System.out.println(inputLine); // sortie standard

        in.close();
```

3. Connexion avec un serveur PHP

Dans le cas d'exécution d'un programme script serveur (exemple de PHP), il est important de pouvoir lui fournir les arguments et/ou les données et de lire la réponse généré par le script.

Tester et commenter le code suivant (il faut lui donner une URL d'une page pour stringtosend) :

```
      import java.net.*;
      import java.util.*;
      import java.io.*;

      public class URLConnectionEx {

        public static void main (String args[]) throws IOException {

          if (args.length != 1)
             throw new RuntimeException ("Syntaxe: java URLConnectionEg
<url>");

        String stringtosend = URLEncoder.encode(args[0]);

        URL url = new URL("http://www.polytech2go.fr/cgi-bin/montest.php");
        URLConnection c = url.openConnection();

        c.setDoOutput (true);

       PrintWriter out = new PrintWriter(c.getOutputStream());
       out.println("monstring=" + stringtosend); // envoi
       out.close();

       /// Ajouter le code pour lire la réponse du serveur


        }
    }
```

4. Protocole http en Java

4.1. Que fait le code suivant

```
import java.net.*;
import java.io.*;

public class jhttp
{
  public static void main ( String[] args ) throws IOException
  {
    try
    {
      URL url = new URL( args[0] );

      URLConnection c = url.openConnection();

      for (int i=0; ; i++)
      {
            String name = c.getHeaderFieldKey(i);
            String value = c.getHeaderField(i);

            if (name == null && value == null)     // end of headers
            {
              break;
            }

            if (name == null)     // first line of headers
            {
              System.out.println("Server HTTP version, Response code:");
              System.out.println(value);
              System.out.print("\n");
            }
            else
            {
              System.out.println(name + "=" + value);
            }
      }
    }
    catch (Exception e) {}
  }
}
```

4.2. Code de réponse : classe HttpURLConnection

Tester et commenter le code suivant :

```
import java.net.*;
import java.io.*;

public class hrc
{
  public static void main ( String[] args ) throws Exception
  {
    try
    {
      URL url = new URL( args[0] );

      HttpURLConnection  c = (HttpURLConnection) url.openConnection();

      System.out.println( c.getResponseCode() );
    }
```

```
    catch (Exception e) {}
  }
}
```

5. Envoie d'un mail

5.1. Configuration pour l'envoi du mail avec le serveur smtp de gmail

5.1.1. Activation du compte Gmail pour autorisé les envois de mail par les applications (il faut se loger chez gmail avant, vous pouvez utiliser votre mail académique) : https://www.google.com/settings/security/lesssecureapps

5.1.2. Installation de l'API javax.mail.jar

a. Téléchargement https://javaee.github.io/javamail/#Download_JavaMail_Release

b. Sous Windows : inclure le jar dans le projet

c. Sous linux (on va pas le faire): en root

i. [root@iga ext]# update-alternatives --display java
(par exemple /usr/lib/jvm/jre-1.6.0-openjdk/)
Choisir le meilleur endroit (*)

ii. Puis placer le jar dans lib/ext de ce chemin

d. Désactiver l'anti-virus (plutôt le parefeu)

5.1.3. Conditions d'utilisation des serveurs mails de google

e. A lire https://support.google.com/a/answer/176600?hl=fr Et a commenter. Quelle est la solution que nous allons utiliser

5.1.4. Code a adapter avec votre compte

```
import java.util.Properties;
import javax.mail.Message;
import javax.mail.MessagingException;
import javax.mail.PasswordAuthentication;
import javax.mail.Session;
import javax.mail.Transport;
import javax.mail.internet.InternetAddress;
import javax.mail.internet.MimeMessage;

public class mail {
   public static void main(String[] args) {



      // Recipient's email ID needs to be mentioned.
    String to = "XXXXX";

      // Sender's email ID needs to be mentioned
           String from = "XXXX";
      final String username = "XXXX";//change accordingly
      final String password = "XXXX";//change accordingly

      // Assuming you are sending email through relay.jangosmtp.net
      String host = "XXXXX";

      Properties props = new Properties();
  // Nous supposons ici que le serveur smtp de l'entreprise demain
  //l'authentification et utilise un tunnel SSL (port 465) ou
  //TLS (port 587)
```

```
        props.put("mail.smtp.auth", "true");
        props.put("mail.smtp.starttls.enable", "true");
        props.put("mail.smtp.host", host);
        props.put("mail.smtp.port", "587");

        // Get the Session object.
        Session session = Session.getInstance(props,
        new javax.mail.Authenticator() {
           protected PasswordAuthentication getPasswordAuthentication() {
              return new PasswordAuthentication(username, password);
           }
        });

        try {
           // Create a default MimeMessage object.
           Message message = new MimeMessage(session);

           // Set From: header field of the header.
           message.setFrom(new InternetAddress(from));

           // Set To: header field of the header.
           message.setRecipients(Message.RecipientType.TO,
           InternetAddress.parse(to));

           // Set Subject: header field
           message.setSubject("Testing !");

           // Now set the actual message
           message.setText("Hello, this is sample for to check send "
              + "email using JavaMailAPI ");

           // Send message
           Transport.send(message);

           System.out.println("Sent message successfully....");

        } catch (MessagingException e) {
              throw new RuntimeException(e);
        }
     }
  }
```

Remplir les champs et tester

5.2. Ajouter dans le Body du mail, un contenu Text et un contenu HTML

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

Multipart multiPart = new MimeMultipart("alternative");

MimeBodyPart textPart = new MimeBodyPart();
textPart.setText(text, "utf-8");

MimeBodyPart htmlPart = new MimeBodyPart();
htmlPart.setContent(html, "text/html; charset=utf-8");

multiPart.addBodyPart(textPart); <-- first
multiPart.addBodyPart(htmlPart); <-- second
message.setContent(multiPart);
```

5.3. En cherchant sur google, ajouter une pièce jointe au mail.

TP Partie 2 : mise en place de résolution DNS, PING, Socket et Thread

Documentation

1. https://openclassrooms.com/courses/java-et-la-programmation-reseau
2. Fichier du TP : IP-M1-2013204.pdf
3. http://www.computing.dcu.ie/~humphrys/Notes/Networks/java.html
4. http://cs.lmu.edu/~ray/notes/javanetexamples/

A. IP Address

1. **Exercice : Lecture**
   - Commencer par lire IP-M1-20132014.pdf
2. **Exercice : Get IP address of hostname**
   - Commencer par éditer et commenter le fichier ip.java
   - Compilation et exécution sous linux
     - Pour compiler : javac ip.java
     - Pour exécuter : java ip www.computing.dcu.ie
   - Compiler et exécuter la classe. Faire plusieurs tests sur des sites différents
3. **Exercice : Get the hostname of an IP addres**
   - Utilisation de l'objet `InetAddress` pour récupérer, en fonction de ce que vous allez saisir, soit l'adresse IP d'une adresse internet, soit le nom de domaine d'une adresse IPv4.
     Write program to find hostname given IP. Note getByName() is flexible in its input.

4. **Exercice : Ping**
   - Faire un sort de ping avec la methode isReachable() : tente un envoi de ECHO REQUEST en ICMP (si privilège ok), ou alors tente d'établir une connexion TCP avec le port 7 (Echo).

B. Socket

1. **Exercice : Date Server**
   - Commencer par éditer et commenter les lignes de la classe DateServer.java.
   - Compiler et Exécuter cette classe. Commenter le résultat.
   - Editer et commenter les lignes de la classe DateClient.java
   - Compiler et Exécuter cette classe. Commenter le résultat.
   - Est-ce que le serveur peut traiter plusieurs clients simultanément ?

2. **Exercice : Port Scanner**
   - En lisant la classe ports.java, proposer un programme qui permet de lister l'ensemble des ports ouverts sur une machine passée en argument. Vous pouvez faire une petite console pour saisir le hostname cible

C. Thread

1. **Exercice : PrintThread1**
   - Commencer par éditer et commenter les lignes de la classe PrintThread1.java.
   - Compiler et Exécuter cette classe. Commenter le résultat.

2. **Exercice : PrintThread2**
   - Commencer par éditer et commenter les lignes de la classe PrintThread2.java.
   - Compiler et Exécuter cette classe. Commenter le résultat. Expliquer la différence entre les deux classes.

3. **Exercice : Ping-Pong**
   - Ecrire une classe Ping qui réalise un Thread qui affiche « Ping » à intervalle irrégulier
   - Ecrire une classe Pong qui réalise un Thread qui affiche « Pong » à intervalle irrégulier
   - Ecrire une classe Go qui lance les Threads Ping et Pong

D. Threaded Server

1. **Exercice : Date Server**
   - Proposer une version parallèle du serveur de temps précédent. Procéder aux tests.

2. **Exercice : Hello Server**
   - Editer, commenter et exécuter le serveur-client Hello