Institut Galilée PDJ
Master 1

T.P. 4

# Mail, JNDI, SAX

L'objectif de ce TP en deux parties est de mettre en place un logiciel ayant l'objectif suivant :

- un client se connecte à un serveur et donne un nom de répertoire et une adresse mail
- le serveur envoie par mail le contenu du répertoire

Plus techniquement, l'objectif est de vous faire pratiquer les modules suivants :

- JavaMail : pour l'envoi par mail
- JNDI : pour la lecture du répertoire
- XML (SAX) : pour le protocole d'envoi-réception

## 1 JavaMail et JNDI

Dans ce premier cas, on considèrera que le serveur est accessible par une socket (mode client-serveur). Le client connecté envoie séquentiellement

- le nom de la requête (seulement "LIST"),
- le nom de répertoire,
- l'adresse mail

Le programme pourra être fait sous Eclipse, Netbeans ou par fichier direct. De préférence, on lancera le serveur dans une console, le client dans une autre console. Les classes à écrire sont les suivantes :

- `Client.java` : Se connecte au serveur, envoie les informations dans l'ordre.
- `ServerMaitre.java` : Crée un thread avec une instance de `ServerEsclave` pour traiter les données reçues.
- `ServerEsclave.java` : Crée une instance de `ReqList` en demandant d'exécuter la requête si la commande est "LIST".
- `ReqList.java` : Crée une instance de `Repertoire` pour récupérer l'énumération (NamingEnumeration) du contenu du répertoire passé en argument. Puis crée une instance de `SendMail` pour envoyer un mail avec le contenu de l'énumération transformé en chaîne de caractères.
- `Repertoire.java` : Utilise le provider de Sun pour les systèmes de gestion de fichier pour envoyer l'énumération du contenu d'un répertoire.
- `SendMail.java` : envoi de mail

Les éléments suivants sont nécessaires :

- `fscontext.jar` et `providerutil.jar` pour JNDI
- `activation.jar` et `mail.jar` pour le JavaMail

Ces librairies peuvent être copiées à partir de mon site web (M1 PDJ).
La compilation se fait par la commande suivante :

```
javac -classpath mail.jar *java
```

Le lancement du serveur s'effectue par (le paramètre 11000 peut être changé !) :

```
java -classpath .:mail.jar:fscontext.jar:providerutil.jar ServerMaitre 11000
```

Le lancement du serveur s'effectue par (les paramètres peuvent être changés !) :

```
java Client localhost 11000 LIST . cf@lipn.univ-paris13.fr
```

# 2 XML

Faire cet exercice aussi en mode console. La structure XML requise est la suivante :

```
<Request>
   <ReqName>nom de la requête (seulement "LIST")</ReqName>
   <Dir>nom de répertoire</Dir>
   <MailAddress>adresse mail</MailAddress>
</Request>
```

On reprend la structure des classes précédentes avec les modifications suivantes :

- `Client.java` : Se connecte au serveur, envoie les informations en utilisant un envoi au format XML (on construit juste une chaîne de caractères).
- `ServerEsclave.java` : Crée une instance de `ReqList`. Analyse le flux d'entrée en créant une instance de `SAXReqReader` qui modifiera l'instance de `ReqList`. Exécute la requête si la commande est "LIST".
- `SAXReqReader.java` : Analyse le flux d'entrée en utilisant comme gestionnaire de contenu une instance de `SAXReqReaderProcess`.
- `SAXReqReaderProcess.java` : Analyse le flux d'entrée selon les mots clefs.

Les commandes de compilation et d'exécution sont identiques au cas précédent.