**Université Bretagne Sud – UFR SSI**

Master Informatique 1re année – 2023/2024

# INF 2165 – Introduction aux systèmes distribués

# TP MOM 1 : Introduction à OpenJMS

L'objectif du TP est de construire et exécuter une application distribuée simple en Java utilisant l'API JMS. Cette application fera usage des communications sur une file de messages ainsi que des communications basées-contenu sur des topics.

## 1 Prise en main d'OpenJMS

### 1.1 Installation et environnement

OpenJMS est une implémentation libre de la spécification JMS 1.1. La documentation relative à OpenJMS est accessible à l'URL `http://openjms.sourceforge.net`.

Pour pouvoir utiliser OpenJMS en TP, décompresser l'archive `openjms.tgz` (donnée dans le répertoire de forum `/home/forum/m1info/INF2165/JMS`) dans votre répertoire personnel.

La compilation et l'exécution nécessite que les points suivants soient respectés :
- La variable `JAVA_HOME` contient le répertoire d'installation du JDK 8 (`/opt/jdk8`)
- la variable `OPENJMS_HOME` contient le répertoire d'installation d'OpenJMS (e.g. `$HOME/openjms-0.7.7-beta-1`)
- Les trois fichiers jar suivants sont dans votre CLASSPATH :

```
$OPENJMS_HOME/lib/openjms-0.7.7-beta-1.jar
$OPENJMS_HOME/lib/jms-1.1.jar
$OPENJMS_HOME/lib/jndi-1.2.1.jar
```

### 1.2 Lancement et configuration de la plate-forme avec l'outil graphique

#### Outil graphique d'aministration

Un outil graphique est fourni dans OpenJMS pour administrer la plate-forme, il est lancé par le script `$OPENJMS_HOME/bin/admin.sh`.

Avec cet outil vous pourrez tout d'abord lancer un serveur JMS. La configuration de ce serveur est définie dans le fichier XML `$OPENJMS_HOME/config/openjms.xml`. Vous trouverez notamment dans ce fichier sous quel nom est enregistré dans JNDI la *ConnectionFactory* créée par le serveur.

Puis il sera possible d'établir une connexion à ce serveur ; l'outil graphique devient alors un client d'administration connecté au serveur JMS. Via la connexion établie, vous pouvez alors, en manipulant une arborescence d'icones, ajouter ou supprimer une file, un topic ou un utilisateur, voir le nombre de messages contenu dans chaque file ou topic...

#### Connecteurs

OpenJMS autorise la connexion des clients (clients producteurs et consommateurs ainsi que clients d'administration) via des connecteurs supportant plusieurs protocoles. Vous utiliserez les connecteurs basés sur TCP.

Un fichier `jndi.properties` accessible via le CLASSPATH permet au client de spécifier le connecteur utilisé. Il pourra, par exemple dans le cas où le serveur tourne sur la même machine que le client, contenir les lignes suivantes :

```
java.naming.provider.url=tcp://localhost:3035
java.naming.factory.initial=org.exolab.jms.jndi.InitialContextFactory
```

# 2 Applications de communication

## 2.1 Communications en mode point à point

À l'aide de l'outil d'administration graphique, créez une file de message « file1 ».

Écrivez un programme producteur Java utilisant l'API JMS. Ce programme se connecte au serveur OpenJMS et envoie une suite de messages (chaînes de caractères) sur cette file « file1 ». Le nombre de messages à émettre sera passé en paramètre du programme.

Écrivez un programme consommateur Java utilisant l'API JMS. Ce programme se connecte au serveur OpenJMS et reçoit une suite de messages (chaînes de caractères) depuis la file « file1 ». Le nombre de messages à recevoir sera passé en paramètre du programme. Vous écrirez deux versions de ce programme :

— Dans une première version, vous utiliserez le mode *pull* (réception synchrone) dans lequel le consommateur bloque en attendant les messages, ces messages pouvant avoir été émis avant le lancement du consommateur. Vous vérifierez les différentes possiblités d'ordonnancement des émissions et réceptions en testant différents ordres de lancement de vos programmes. Vous pourrez jouer sur le paramètre du programme (nombre de messages à recevoir) pour consommer les messages en lançant plusieurs fois le programme consommateur.

— Dans une seconde version, vous utiliserez le mode *push* (réception asynchrone) dans lequel le consommateur peut faire un traitement (vous lui ferez afficher un caractère « . » toutes les secondes) en étant interrompu pour traiter les messages qui arrivent (seuls les messages émis après le lancement du consommateur peuvent être reçus par celui-ci).

Fabriquez et employez un scénario de test simple vous permettant de vérifier que le mode de communication est bien point à point : un message émis ne peut être reçu que par un seul consommateur.

## 2.2 Communications en mode publish-subscribe

À l'aide de l'outil d'administration graphique, créez un topic « topic1 ».

Écrivez un programme Java utilisant l'API JMS qui se connecte au serveur OpenJMS et envoi une suite de messages (chaînes de caractères) sur ce topic « topic1 ». Le nombre de messages à émettre sera passé en paramètre du programme.

Écrivez un autre programme Java utilisant l'API JMS qui se connecte au serveur OpenJMS et qui reçoit un une suite de messages (chaînes de caractères) depuis le topic « topic1 ». Le nombre de messages à recevoir sera passé en paramètre du programme. Vous construirez deux versions de ce programme, l'un fonctionnant en mode *push* et l'autre en mode *pull*.

Fabriquez et employez un scénario de test simple vous permettant de vérifier que la communication via les topics est bien une communication multi-points : plusieurs consommateurs concurrents reçoivent bien tous les messages.