Institut Informatique Claude Chappe

**Master Informatique**
**Unité 178UD02**
**Programmation distribuée**

# TP n°1
# Middleware à message - JMS

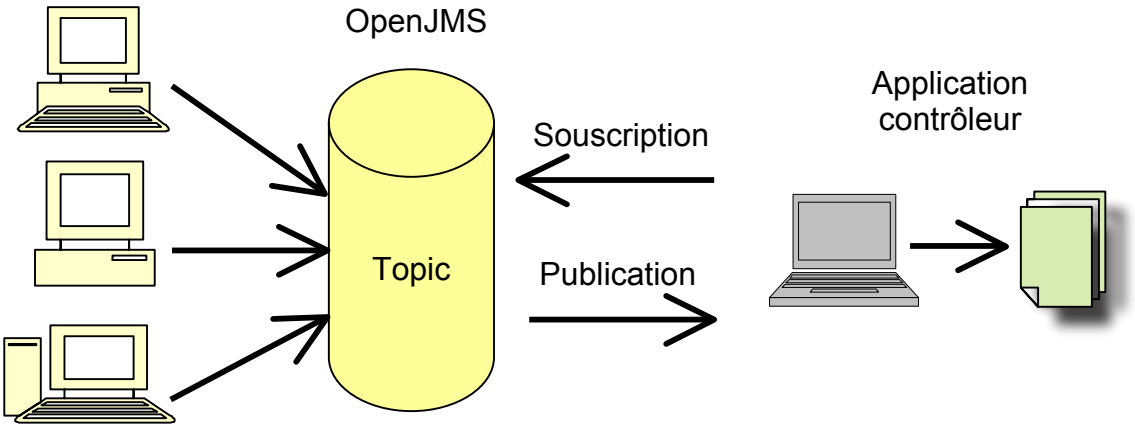
## Objectifs du TP :

- Introduction au middleware à message
- Familiarisation avec l'API JMS
- Mise en œuvre du provider OpenJMS

## Sujet : un système de journalisation centralisé

### Présentation

L'objectif est de développer un logiciel de journalisation centralisé (et simplifié). Un tel système permet de garder un historique des événements survenant sur un équipement ou un logiciel. Les événements peuvent être normaux, par exemples : début d'une session de travail, fin d'une session de travail, exécution normale d'une tâche, etc. Les événements peuvent être également anormaux comme par exemples : exécution anormale d'une tâche, absence d'une ressource, etc.




On va utiliser OpenJMS pour gérer de façon centralisée les événements à journaliser. Afin de ne pas multiplier le nombre de ressources utilisées, un provider JMS recevra les événements des différentes applications surveillées sur un seul topic. Les informations supplémentaires (comme par exemple la date d'émission de l'événement, la machine émettrice) seront transférés en utilisant l'entête du message et ses propriétés. Le corps du message sera un texte décrivant l'événement lui-même.

Une application « contrôleur » s'abonnera à ce topic est aura pour double rôle l'affichage en clair et en continu des événements reçus et la sauvegarde de ses mêmes événements sur disque.

Le logiciel devra avoir les caractéristiques suivantes :

- L'application contrôleur n'est pas une application graphique. L'affichage se fait en mode texte uniquement.
- L'application contrôleur ne sera pas constamment en fonctionnement. Il faut donc faire attention à ce qu'elle ne perde aucun événement entre deux mises en fonctionnement.
- La sauvegarde des événements sera faite dans un simple fichier texte ou un fichier structuré XML.
- L'application contrôleur doit faire l'affichage en même temps que la sauvegarde.
- Chaque événement affiché et stocké par l'application contrôleur devra comporter au minimum les informations suivantes dans cet ordre : la date de l'événement, le nom de la machine concernée, le login de l'utilisateur, le nom de l'application concernée si le message concerne une application et l'événement lui-même.
- Utilisation des propriétés des messages pour transférer les informations (sauf l'événement lui-même transféré dans le corps du message).

## Travail à réaliser

Le travail consiste à configurer OpenJMS et à finir de développer l'application contrôleur. Vous partirez d'un code existant mais incomplet disponible sur UMTICE.

Première partie : installation et configuration du provider OpenJMS

1. Installation du provider OpenJMS
    - Téléchargez l'archive d'installation d'OpenJMS (référence ci-après)
    - Décompressez le fichier d'archive
2. Dans le répertoire `docs`, lisez l'*Administrator Guide* (partie *Running OpenJMS*)
3. Lisez le *User Guide* (partie *Using OpenJMS* et partie *Examples*)
4. Lisez et exécutez les exemples de codes du répertoire `examples/basic`. Pour visualiser l'état du provider OpenJMS, vous utiliserez l'outil d'administration `bin/admin.sh` (référence ci-après).
5. Configurez le provider pour l'application de journalisation (modifiez le fichier `config/openjms.xml` puis relancer le provider) :
    - Un *topic* nommé « journal_topic »
    - Un *subscriber* nommé « journal »

Seconde partie : conception et programmation de l'application contrôleur

1. Installation :
    a. Téléchargez le fichier zip du code source se trouvant sur UMTICE.
    b. Décompressez le fichier et placer le répertoire `Archive` au même niveau que le répertoire `openjms-0.7.7-beta-1`.
2. Développement :
    a. Finir le développement des classes `JournalListener` et `JournalApp`
3. Compilation :
    a. Dans le répertoire `Archive`, compilez l'application avec la commande :
    ```
    javac -cp ../openjms-0.7.7-beta-1/lib/jms-1.1.jar
    journal/*.java journaltest/*.java
    ```
4. Exécution :
    a. Dans un terminal, dans `bin`, lancer l'outil d'administration (cf. ci-après).
    b. Dans un terminal, dans `bin`, lancer le provider : `./startup.sh`
    c. Dans un terminal, dans `Archive`, lancer le test : `./essai.sh`
    (attendre la fin de l'exécution en regardant les processus avec la commande `ps`)

d. Dans un terminal, dans `Archive`, lancer le contrôleur : `./controleur.sh` (touche [`Enter`] pour arrêtez)
e. Dans un autre terminal dans le répertoire `Archive` lancer le test : `./essai.sh`

**Remarques :**

- Pour exécuter les scripts sh d'un répertoire lancer d'abord la commande :
  `chmod +x *.sh`

- L'exécution des scripts sh nécessite de positionner la variable d'environnement `JAVA_HOME` avec la commande : `export JAVA_HOME=/usr`
  (exemple sous Mac OS X ; à modifier selon votre environnement d'exécution).

- Vous pouvez utiliser l'outil d'administration `admin.sh` du répertoire `bin` pour observer l'évolution du nombre de messages dans les files et les topics utilisés. Pour cela, il faut auparavant effectuer les actions suivantes :
  - rendre exécutable les scripts sh du répertoire `bin`
  - positionner la variable d'environnement `JAVA_HOME`
  - modifier le fichier `config/openjms.xml` de la manière suivante :
    `<AdminConfiguration script="${openjms.home}/bin/startup.sh"/>`

## Travail à rendre

Le travail donne lieu à un compte-rendu à déposer sur UMTICE ***à la fin de la séance***. Il sera rendu uniquement sous forme de fichier compressé (format zip) contenant :

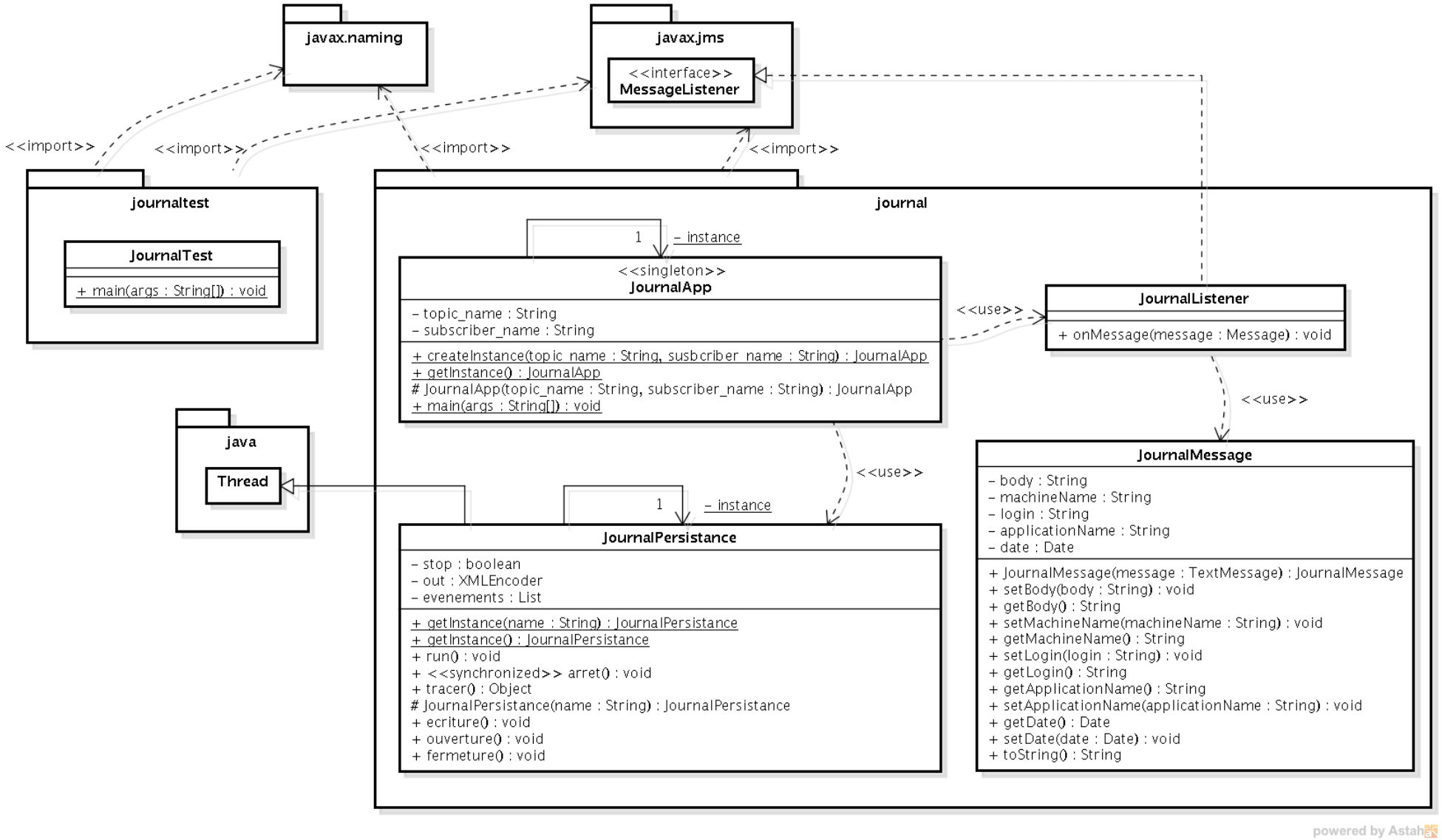
- L'ensemble des fichiers (fichiers sources, de configuration, d'exécution, etc.) utiles pour compiler et exécuter l'application répartie ;
- Un document comportant la réponse à la question suivante : comment avez-vous pris en compte le fait que l'application contrôleur n'est pas toujours en fonctionnement ?

## Références web

Site de l'implémentation OpenJMS http://openjms.sourceforge.net/

Site de la spécification JMS http://www.oracle.com/technetwork/java/jms/index.html

Site de l'API JMS 2.0 https://jms-spec.java.net/2.0/apidocs/


javax.naming
javax.jms
<<interface>>
MessageListener
<<import>>
<<import>>
<<import>>
<<import>>
journaltest
JournalTest
+ main(args : String[]) : void
journal
1
- instance
<<singleton>>
JournalApp
- topic_name : String
- subscriber_name : String
+ createInstance(topic_name : String, susbcriber_name : String) : JournalApp
+ getInstance() : JournalApp
# JournalApp(topic_name : String, subscriber_name : String) : JournalApp
+ main(args : String[]) : void
<<use>>
JournalListener
+ onMessage(message : Message) : void
<<use>>
java
Thread
<<use>>
1
- instance
JournalPersistance
- stop : boolean
- out : XMLEncoder
- evenements : List
+ getInstance(name : String) : JournalPersistance
+ getInstance() : JournalPersistance
+ run() : void
+ <<synchronized>> arret() : void
+ tracer() : Object
# JournalPersistance(name : String) : JournalPersistance
+ ecriture() : void
+ ouverture() : void
+ fermeture() : void
JournalMessage
- body : String
- machineName : String
- login : String
- applicationName : String
- date : Date
+ JournalMessage(message : TextMessage) : JournalMessage
+ setBody(body : String) : void
+ getBody() : String
+ setMachineName(machineName : String) : void
+ getMachineName() : String
+ setLogin(login : String) : void
+ getLogin() : String
+ getApplicationName() : String
+ setApplicationName(applicationName : String) : void
+ getDate() : Date
+ setDate(date : Date) : void
+ toString() : String
powered by Astah