MOM

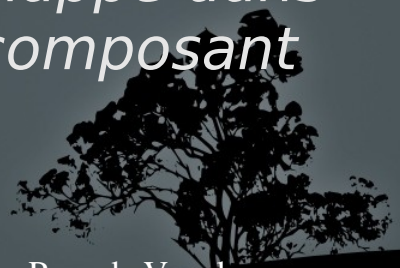
- Message Oriented Middleware
  - *Principes : concepts*
    - *Communication par messages*
      - *asynchrone*
    - *Couplage faible des composants*

MOM

- Principes : modèle de communication
  - *2 modes :*
    - *Point à point : un composant produit les messages, un autre les consomme*
    - *Abonnement (Publish Subscribe):*
      - *les consommateurs s'abonnent à un Topic*
      - *Les messages du topic sont dans une file d'attente de la messagerie*

MOM

- Principes : les messages
  - *Transport : en-tête technique*
  - *Routage : passage des messages entre 2 instances d'un MOM*
  - *Persistance des messages : sauvegarde des messages en cas d'interruption du service de messagerie*
  - *Fiabilité : accusés de réception, persistance*
  - *Transformation : le message est mappé dans l'environnement natif de chaque composant*



MOM

- *Implémentations*
  - *Peu de standards...*
  - *AMQP : protocole basé sur JMS*
    - *CISCO, IONA*
    - *2006- [2011]*
  - *XMPP : Extensible Messaging and Presence Protocol*
    - *Basé sur XML, opensource*
    - *JABBER (2000), MMS (Microsoft Messenger service)*
  - *JMS : JAVA Messaging Service*

JMS

- *C'est quoi ?*
  - *Une API fournie par SUN.*
- *Pourquoi faire ?*
  - *Pour utiliser des services de messagerie dans des applications Java*
- *Intégré à J2EE à partir de la version 1.3 (vue comme un service)*

JMS

- Les acteurs :
  - *Provider JMS* : outils qui implémente JMS pour échanger des messages
  - *Client JMS* : composant JAVA qui produit ou consomme des messages
  - La *ConnectionFactory* : usine de messages
  - Les destinations : *Queue* et *Topic*

JMS

- La communication :
  - Point à point : message produit par un producteur, consommé par un consommateur (*Queuing*)
  - *Publish/subscribe* (abonnement): messages envoyés et reçus par des groupes de producteurs/consommateurs

JMS

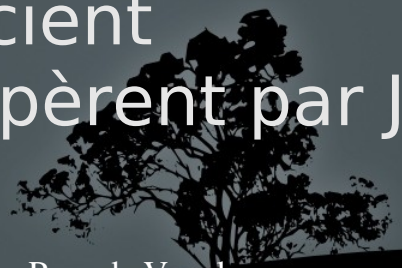
- La communication :
  - Asynchrone : c'est le principe de la com. Par messages !
  - Synchrone : la méthode *receive()* une resynchronisation retardée
  - Type de messages : paquet d'octets, texte brute, objets sérialisés

JMS

- Les interfaces du package JMS :
  - Connection
  - Session
  - Message
  - MessageProducer
  - MessageListener

JMS

- La fabrique de connections :
  - Rôle : fabrique des objets qui permettent de se connecter à un *broker* (bus, service)
  - 2 types :
    - *QueueConnectionFactory*
    - *TopicConnectionFactor*
  - Les connections s'instancient directement ou se récupèrent par JNDI

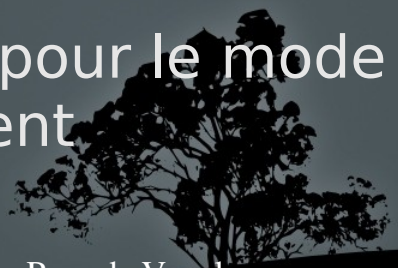


JMS

- Les connections
  - Constructeurs : *createQueueConnection()* ou *createTopicConnection()*.
  - *connection.start()* : démarre la connection
  - *connection.stop()* : suspend la connection
  - *connection.close()* : la ferme

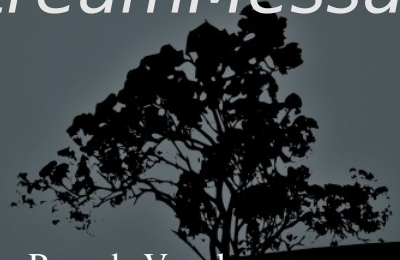
JMS

- Les sessions
  - C'est quoi ? Contexte transactionnel pour une connexion donnée
  - Créées à partir d'une connexion
  - *Single thread*
  - Permet de créer des messages (à émettre ou à recevoir) :
    - interface *QueueSession* pour le mode point à point
    - interface *TopicSession* pour le mode publication/abonnement



JMS

- Les messages
  - Composés de 3 parties : en-tête, propriétés (données fonctionnelles), du corps du message
  - Méthodes *createXXXMessage()* pour créer des messages au format XXX
  - Interfaces dérivées :
    - *BytesMessage, MapMessage, ObjectMessage, StreamMessage, TextMessage*



JMS

- Messages: réception
  - Synchrone, asynchrone (*thread*)
  - C'est un objet de type *message* qui est reçu
  - → il faut faire un cast :

```
Message message = ...
If (message instanceof TextMessage) {
TextMessage textMessage = (TextMessage) message;
System.out.println("message: " + textMessage.getText());
}
```

JMS

- Messages: réception
  - interface *MessageConsumer*
  - *2 dérivées : QueueReceiver et TopicSubscriber*
  - *Quelques méthodes* :
    - *Message receive()* : attendre et retourner le message à son arrivée
    - *Message receive(long n) : attendre le message n millisecondes*
    - *Message receiveNoWait() : renvoyer un message sans attendre s'il y en a un de présent*
    - *setMessageListener(MessageListener) : associer un listener capable du traitment asynchrone du message*

JMS

- Messages: envoi
  - interface *MessageProducer*
  - *2 dérivées :*
    - *QueueSender*
    - *TopicPublisher*
  - *Créées avec :*
    - *createSender()*
    - *createPublisher()*

JMS

- Messages: interface *TopicPublisher*
-
  - *void publish(Message)*
    - *Envoyer le message dans le topic défini dans l'objet de type TopicPublisher*
  - *void publish(Topic, Message)*
    - *Envoyer le message dans le topic fourni en paramètre*

ActiveMQ

- MOM qui implémente/étend JMS
  - s'appuie sur :
    - Apache Camel : implémentation des *Entreprises Integration Patterns*
    - *Jetty : serveur d'application (intégré à ActiveMQ)*
  - *et qui est utilisé par :*
    - *Apache Service Mix, Mule*
    - *Geronimo (serveur d'applications)*
    - *Axis, CXF : extensions d'AMQ*
- Support *:*
  - *Langages : C, C++ Ajax, Rest, SOAP, Delphi, Perl, PHP, Python, Ruby...Support par des passerelles/wrappers*
  - *Protocoles : AMQP, OpenWire, STOMP, XMPP*

ActiveMQ

- Open source
- Techno JAVA
- Composants orientés métiers
- Communications entre groupes de clients/serveurs
- Solutions spécifiques :
  - Support des *clusters* de machines
  - Support du temps réel
  - ...

ActiveMQ

- Gestion des messages :
  - Groupes de messages :
    - Envoi d'un paquet de messages à un destinataire
    - Choix automatique d'une autre destinataire si le premier est défaillant
  - Pas de priorité des messages
  - Destination virtuelles : *topics, queues* redirigeant vers des composants du domaine
  - *Total ordering*

ActiveMQ

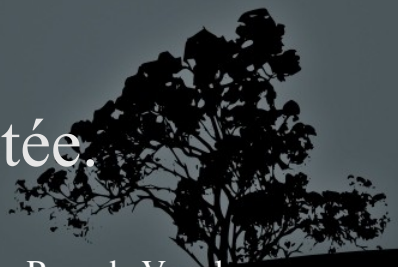
- Traitement des messages :
  - *Modèle d'intégration d'Entreprise EIP*
  - Fonctionnalités de routage/transformation des requêtes
- Erreurs :
  - Journalisation des transactions, reprise sur incidents, etc.

ActiveMQ

- Persistance :
  - *Active MQ Message Store*
    - Stockage dans un fichier avec journalisation et cache
    - Gestion très efficace du cache
- Répartition de charge, disponibilité
  - *Cluster* de *Brokers* : tolérance aux pannes et à la montée en charge
  - Réseau de *Brokers* : *Queue* et *Topics*

ActiveMQ

- Répartition de charge, disponibilité
  - Réplication :
    - Maître-esclave : redondance avec un esclave par maître
    - Message Store partagé
    - Fédérations de *brokers* (même domaine)
- Sécurité : pas le point fort d'AMQ
  - Service d'authentification intégré sous forme de *plugin*
  - L'interconnexion peut être cryptée.



ActiveMQ

- Administration
  - Monitoring des plate-formes :
    - Interface Web, par des messages (voir par XMPP)
  - *Advisory Messages* : messages d'information relatifs à :
    - Connections clients
    - Files d'attentes créées, détruites
    - Messages traités, expirés, etc.
  - Interface d'administration par défaut : *HTTP://0.0.0.0:8161/admin*