TD sur JMS

Questions de cours :

1) Qu'est-ce qu'un middleware orienté message (MOM) ? Quelles différences faites-vous entre un MOM et JMS ?

MOM : Message Oriented Middleware
Intergiciels orientés Messages
But :

- Création de messages complexes (pas uniquement des tableaux d'octets)
- Emission/Réception de messages
- Eventuellement : stockage de messages (message queuing)

JMS est une API Java qui permet de travailler avec des MOM fonctionnant en mode Message Queuing

2) Qu'est-ce que JNDI ? A quoi sert-il lorsqu'il est utilisé avec JMS ?

JNDI (Java Naming and Directory Interface) est une API pour gérer des annuaires de données ou d'objets. Les annuaires peuvent être gérés par des fichiers ou par des systèmes d'annuaires répartis comme LDAP.

Pour JMS il sert à nommer :

- les usines de connexion (ConnectionFactory) qui servent à créer des connexions avec des fournisseurs JMS ;
- les files et les sujets de discussions.

3) Parmi les applications suivantes, indiquez si l'utilisation d'un middleware orienté message est adapté en justifiant vos décisions :

1. Visiophonie ;

Pas adapté au temps-réel et aux flux audio/vidéo

2. Consultation d'infos météo à distance ;

Adapté en utilisant le mode requête/réponse mais un MOCS comme CORBA est tout aussi adapté.

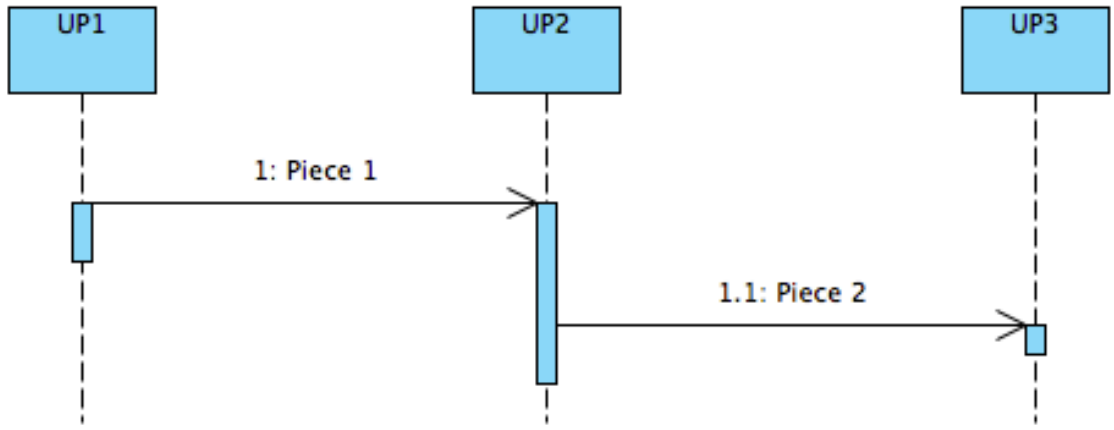
3. Messagerie d'entreprise (courrier électronique, fax, messages vocaux) ;

Adapté grâce à la gestion de l'asynchronisme.

Exercice 1 : le but de cet exercice est de simuler trois unités de production dans une ligne automatisée d'usine.

La première unité produit des pièces qui sont consommées par la seconde. Cette dernière produit ensuite une nouvelle pièce qui est consommée par la troisième.

1.1 Représentez les interactions entre entités réparties à l'aide d'un diagramme de séquences UML.



1.2 Représentez sur un schéma l'architecture JMS : c'est-à-dire, les différentes entités logicielles et les communications en utilisant des sujets de discussion (*topics*) et/ou des files de messages (*queues*).

1.3 Quel type de message utiliseriez-vous pour le corps des messages échangés ?

ObjectMessage transportant des objets de type Piece (sérialisés).

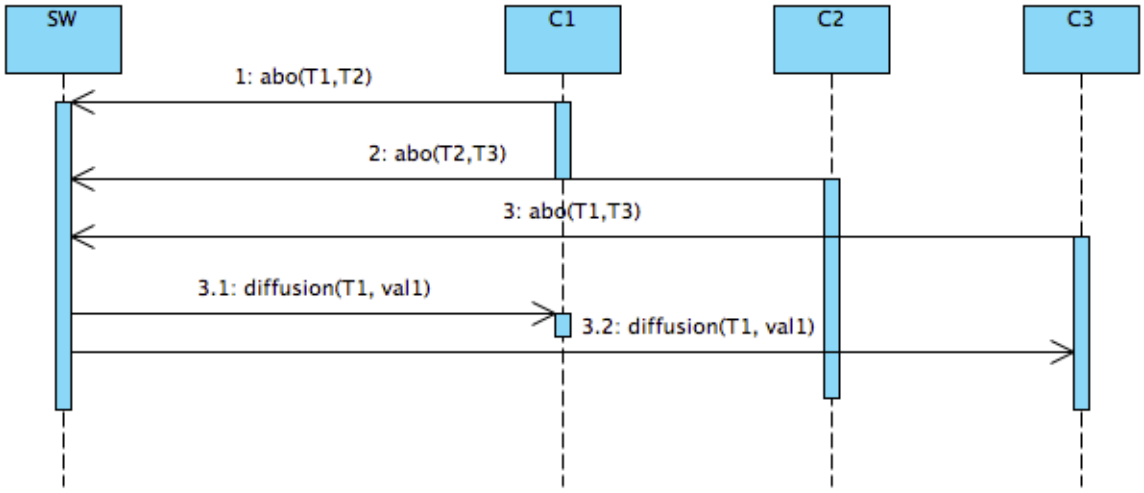
1.4 On veut maintenant gérer le système avec une seule file de message. Comment gérer cela avec JMS ?

On va typer les messages en utilisant l'entête JMSType (type INIT et TRANSFO). A2 utilisera le filtre : "JMSType = 'INIT'" et A3, le filtre : "JMSType = 'TRANSFO'"

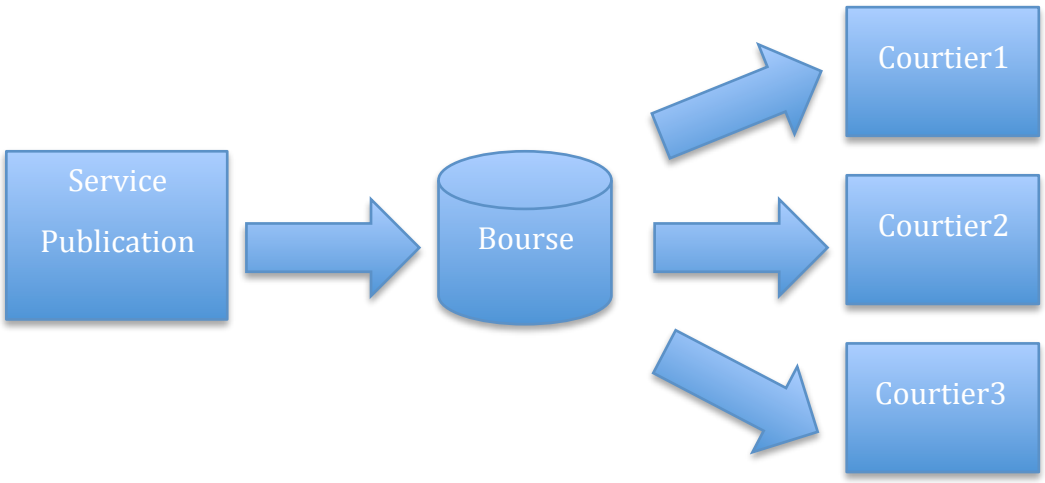
Exercice 2 : le but de cet exercice est de simuler un service web qui diffuse les cours de la bourse d'un certain nombre de titres à un ensemble de courtiers en bourse.

Note : bien sûr un courtier peut s'abonner à plusieurs titres.

2.1 Représentez les interactions entre entités réparties à l'aide d'un diagramme de séquences UML.



2.2 Sur un schéma, placez les différents acteurs et les communications en utilisant des topics et/ou des files de messages.



2.3 Quel type de message utiliseriez-vous pour le corps des messages échangés ?

MapMessage contenant le nom de l'entreprise ou du titre (String), la cotation (float) et la variation de la cotation (float).

Exercice 3 : Une application de type eCommerce en JMS.

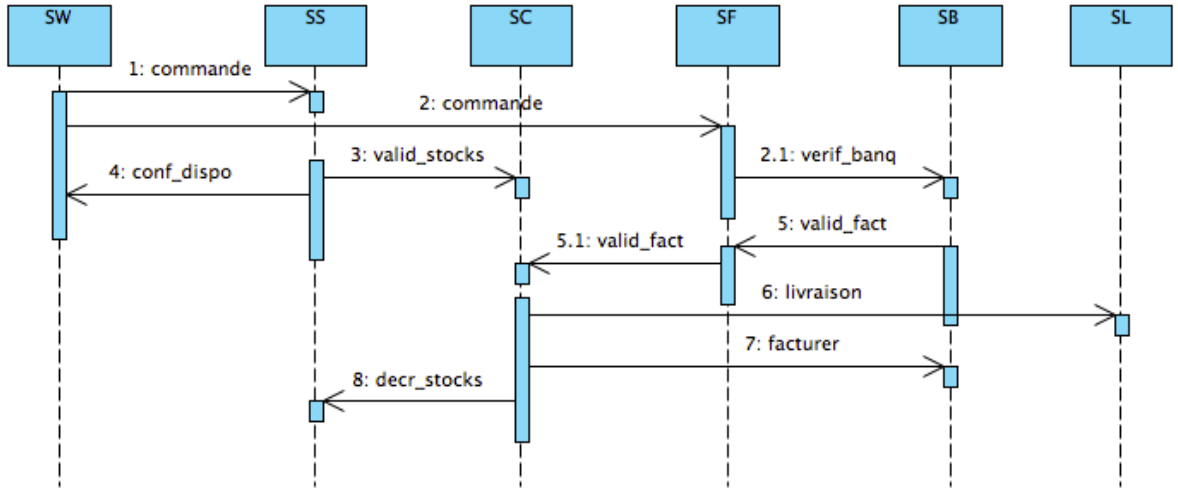
Cette application simule une application complète de eCommerce avec les participants suivants :

- ServeurWeb : un serveur web sur lequel le client choisit des objets à acheter ;
- ServeurClientèle : centralise le traitement des commandes ;
- ServeurStocks : vérifie la disponibilité des objets commandés ;
- ServeurFacturation : centralise le traitement des paiements bancaires ;
- ServeurBanque : vérifie les références bancaires des clients et gère les ordres de paiement ;
- ServeurLivraison : s'occupe de la livraison des commandes.

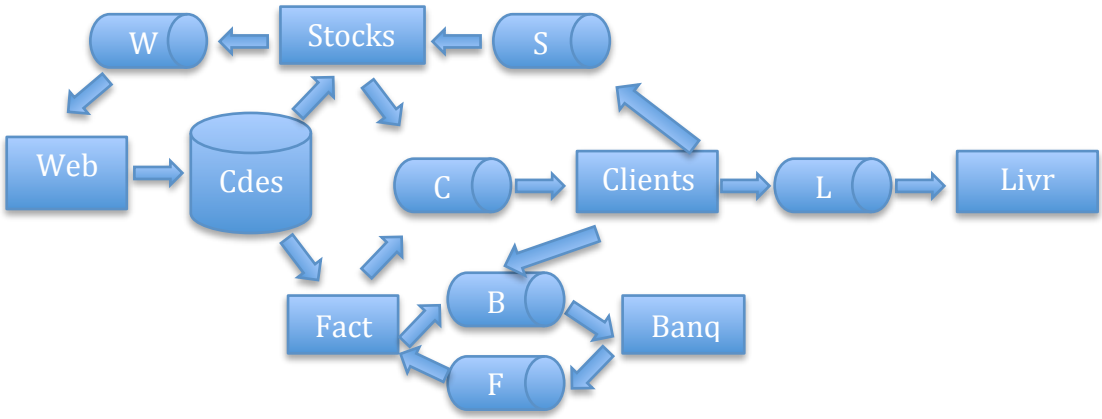
Remarques :

1. Pour qu'une commande soit livrée par le ServeurLivraison, il faut que le ServeurStocks et le ServeurFacturation l'aient validée ;
2. Le ServeurClientèle s'occupe de cette validation ;
3. Le ServeurFacturation contacte le ServeurBanque pour s'assurer que les références bancaires sont correctes et pour demander un ordre de paiement ;

3.1 Représentez les interactions entre entités réparties à l'aide d'un diagramme de séquences UML



3.2 Sur un schéma, placez les différents acteurs et les communications en utilisant des topics et des files de messages.

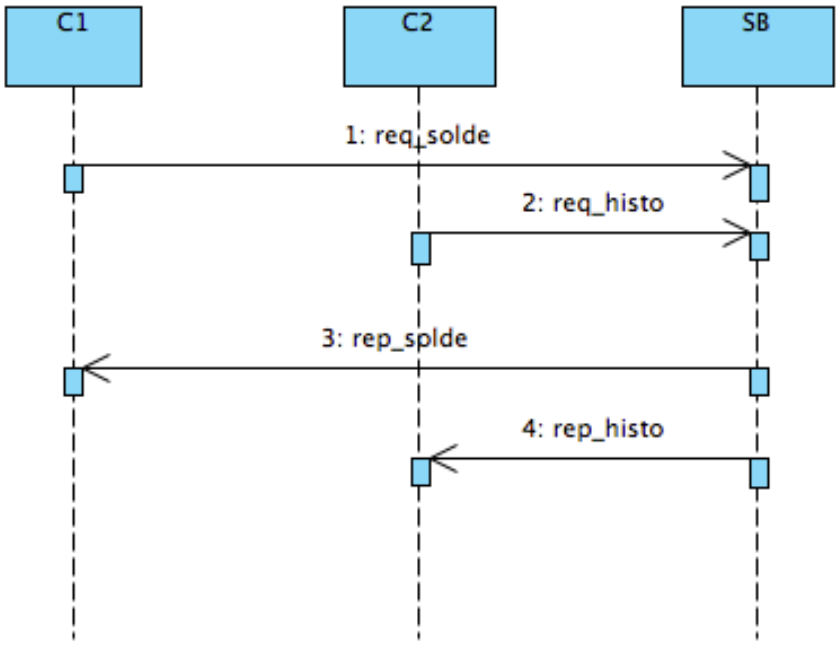


3.2 : Faites un scénario en indiquant quels messages transitent par quelles files/topics.

1 Un client valide sa commande. Web envoie un message commande (id client, id commande, liste des produits et quantités) via le topic Cdes vers Stocks et Fact.
3 Stocks valide la commande et envoie un message valid_stocks (id commande, liste des produits et quantités commandées et disponibles) via la file C vers Clients.
4 Stocks envoie aussi un message de confirmation conf_dispo(id commande, liste des produits et quantités commandées et disponibles) de disponibilité au serveur Web via la file W.
5 Fact récupère les infos bancaires du client et les envoie dans un message verif_banq (id verif, infos bancaires) via la file B à Banq.
6 Banq vérifie les infos bancaires et envoie un message valid_fact (id commande, true ou false) via la file F à Fact.
7 Fact envoie le message valid_fact (id commande, true ou false) via C à Clients.
8 Clients, après la réception des 2 messages valid_stocks et valid_fact d'une même commande, envoie le message livraison(id commande, id client, adresse client, liste des produits et quantités à livrer) via L à Livr ainsi que le message facturer(id client, infos bancaires, somme à prélever) via B à Banque. Il envoie aussi un message decr_stocks(liste des produits et quantités envoyées) pour décrémenter les stocks disponibles à Stocks via S.

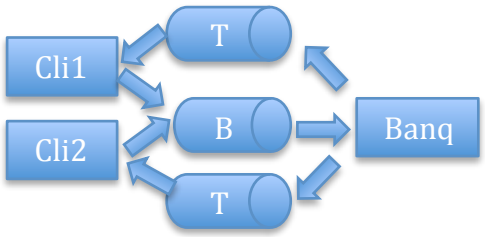
Exercice 4 : On se propose de réaliser une application répartie avec JMS permettant la consultation à distance du solde et de l'historique d'un compte bancaire.

4.1 : Diagramme de séquence



4.2 : Présentez sur un schéma, l'architecture de l'application répartie (applications, files et/ou topics).

Ici on peut directement utiliser le mode requête/réponse présenté à la fin du cours.



B est une file nommée utilisée pour envoyer les requêtes. Les files T sont des files temporaires utilisées pour envoyer les réponses.

4.2 : Décrivez les messages qui seront échangés entre les applications.

Les clients envoient des messages :

Req_solde (id compte) de type StreamMessage

Req_histo (id compte) de type StreamMessage

Le serveur envoie des messages :

Rep_solde (id compte, solde) de type StreamMessage

Rep_histo (id compte, nb opérations, liste opérations) de type StreamMessage.

Exercice 5 : On se propose de développer une application répartie avec JMS permettant la consultation et la gestion d'une bibliothèque (permettant : l'ajout, le retrait, l'emprunt et le retour d'un livre).

5.1 : Décrivez, avec un schéma, l'application que vous proposez en précisant les files de messages, sujets de discussion (topics) et applications que vous allez utiliser.

Le schéma est identique au précédent.

5.2 : Donnez la liste exhaustive des messages que vous allez utiliser (et leurs types JMS) ainsi que leur contenu.

Les clients envoient des messages :

Req_recherche (mots) de type TextMessage

Req_ajout_livre (auteur, titre, année, ref) de type StreamMessage

Req_retrait_livre (ref) de type StreamMessage

Req_emprunt_livre (ref, id emprunteur) de type StreamMessage

Req_retour_livre (ref, id emprunteur) de type StreamMessage

Le serveur envoie des messages :

Rep_recherche (nombre de refs, liste de refs) de type StreamMessage

Rep_ajout_livre (ref, nouveau nombre de livres) de type StreamMessage.

Rep_retrait_livre (ref, nouveau nombre de livres) de type StreamMessage.

Rep_emprunt_livre (ref, id emprunteur, true ou false) de type StreamMessage.
