UNIVERSITÉ DE BOURGOGNE
UFR Sciences et Techniques

Cours: BD et Environnement Distribuées

TP 6 - Serveur d'applications

Préparé par:

MATTA Elie et al.

Table des matières

Introduction

Avec l'évolution des nouvelles technologies, les entreprises, pour la plus part, utilisent de plus en plus l'outil informatique, devenu presque indispensable. Quelques soit le domaine d'activité des entreprises, des applications informatiques appropriées sont développées, pour favoriser et améliorer toutes les étapes du processus métier, allant de la production à comptabilité.
Ces applications deviennent elles aussi de plus en plus importantes, nécessitant beaucoup de ressources, et par conséquent, des systèmes plus puissants.
Aujourd'hui, ces applications sont centralisées sur de gros systèmes et accessibles à partir des machines clientes. Ces gros systèmes sont appelés Serveur d'application, et peuvent contenir plusieurs applications pour l'entreprise. On peut également avoir plusieurs serveurs d'application au niveau de l'entreprise.

L'objectif de ce TP est de détailler les fonctionnalités d'un serveur d'application et d'expérimenter leurs avantages et utilités en développent un exemple d'application sur JONAS. Dans un deuxième temps, on a encore développer une mini-application qui gère le compte d'un solde bancaire en exploitant le composant EJB.

Question 1: Étude des fonctionnalités

A) Les serveurs d'applications

Selon **Wikipedia**[(1)]:
Un serveur d'applications est un logiciel d'infrastructure offrant un contexte d'exécution pour des composants applicatifs. Le terme est apparu dans le domaine des applications web. Dans un sens strict les composants hébergés par le serveur d'applications ne sont pas de simples procédures ou scripts mais de réels composants logiciels conformes à un modèle de composants (EJB, COM, Fractal, etc.).

Les clients des serveurs d'application sont : des programmes autonomes (stand alone application), des applets ou d'autres composants.

La structuration en couches des différents composants mis à disposition par le serveur d'application permet une prise en compte des besoins métier, des interactions avec les utilisateurs, des connexions avec les bases de données, etc.

Les serveurs d'applications sont des logiciels occupant la couche centrale dans une architecture multicouche, qu'elle soit classique 3-tiers (postes clients, serveur de données, serveur d'applications) ou étendue (n-tiers) lorsqu'elle intègre des serveurs d'acquisition (données de terrain, données de process, de back-office, etc.) et/ou des serveurs d'interface (gateways, systèmes coopérants externes, etc.).

Dans un sens plus large, un serveur d'application peut être une machine servant à héberger des applications soit pour permettre leur exécution depuis un poste client (mode client serveur de données, généralement partage de fichiers et politiques de gestion des accès) ou pour déporter l'affichage sur le poste client (mode client serveur d'affichage).

Critiques
Cette definition n'est pas parfaitement exacte car elle très globale et ne precise pas les differents cas d'utilisation que le serveur d'application peut y contriber.

Notre definition
Les serveurs d'applications sont des frameworks qui hébergent des composants modulaires réutilisables (des objets qui peuvent être assembler pour former la fonction nécessaire) et distribués (capable de travailler avec d'autres serveurs). Ces composants doivent répondre à un modèle (EJB, COM,...).
Les serveurs d'applications donc fournisse un "cadre" d'exécution et un ensemble de service permettant la bonne exécution de ces composants.

B) Les grandes fonctionnalités assurés par un serveur d'applications

On va énumérer les fonctionnalités les plus importants et nécessaires fonctionnalités que le serveur d'application doit assurer[(2)]:

- Gestion de ressource: Améliore l'exécution d'un système et simplifie son codage. Ceci élimine le besoin de créer et de maintenir un code de gestion complexe de ressource dans l'application.

- Gestion de transaction: Ils permettent d'employer un API simple pour contrôler des transactions.
- Gestion d'activation des objets: permet d'activer et désactiver les objets et services.

C) Les services hébergés dans un serveur d'application

Pour assurer le bon fonctionnement des fonctionnalités cité ci-dessous, les serveurs d'applications fournissent un certain nombre de service de base:

- ✓ L'optimisation de l'accès aux ressources (locales et distantes)
- ✓ La gestion transactionnelle
- ✓ L'accès aux composants
- ✓ La répartition de la charge
- ✓ Services de sécurité : Fournir les services, tels que l'authentification, l'autorisation, et un transport bloqué, qui crée un environnement bloqué aux ressources déployées.

En optionnel, les serveurs d'applications peuvent offrir encore des services, en particulier:

- ✓ Un service de registre
- ✓ Un service mail
- ✓ Un service de sécurité

D) Les services hébergés dans un serveur d'application

Un composant est un objet qui intègre la logique métier réutilisable et qui adhère à un modèle[3],[4]:

- ✓ Support d'un ensemble de méthodes normées pour :
  - être exécuté dans un serveur d'applications,
  - être intégré dans un AGL en phase de développement, grâce aux méthodes auto descriptives (introspection).
- ✓ Regroupement des méthodes normées dans des interfaces :
  - Le composant donne un descriptif de ses services,
  - La description des services est indépendante de l'implémentation.

Il faut noter que la structure du composant doit permettre eventuellement à un assemblage réel avec d'autres composant, sans se soucier des implémentations.

Donc le modèle de composants apporte un ensemble d'objets et de méthodes afin de mieux gérer les aspects transactionnels et de sécurité.

Composant vs Objet

La notion d'objet est très proche à celle du composant, la difference principale est simple: Ils ne sont pas concus dans le même objectif. En effet, un composant se trouve sur une machine

distante de celle qui aura besoin d'elle, alors qu'un objet est une instance de classe dont la creation et l'execution des methodes sera fait sur la machine qui l'a instancié. D'où on peut deduire que le composant possedera des caracteristiques liées au deploiement et il diminura donc la compléxité d'une application.

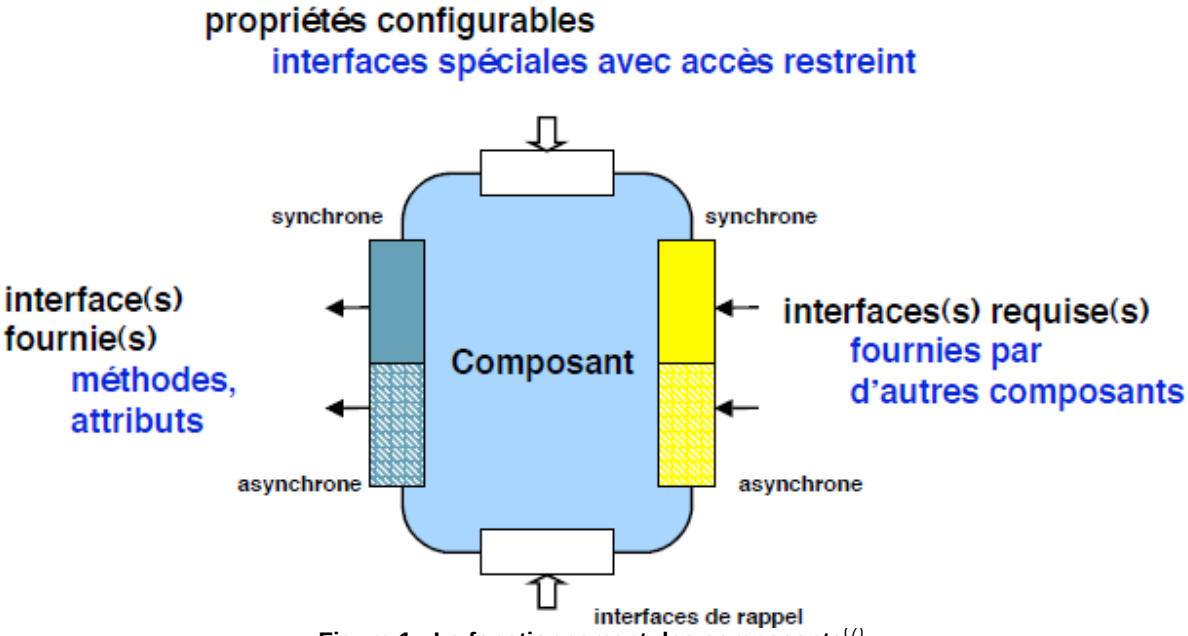

Figure 1 - Le fonctionnement des composants[1]

E) Les modèles de composants existants

Quatre modèles (architectures) de composants se partagent le marché [3],[5]:

1. Le modèle COM (Component Object Model, 1996) de Microsoft, se vent indépendant du langage (VB, C++, C#, Visual J++, etc.)
2. Le modèle EJB (Enterprise JavaBean, 1998), spécification multi éditeur (Sun et IBM à l'origine), se vent multi plateforme et uniquement en langage Java (peut être utilisé comme un service OSGi).
3. Corba-CCM (OMG), la norme CORBA 3 (CORBA Component Model, CCM) définit un standard ouvert qui est une alternative sérieuse aux EJB.
4. De nombreux autres composants: Fractal, Spring, Web services, Avalon (Apache), XPCOM (Mozilla), K-Component, Comet, Kilim, OpenCOM, FuseJ, Jiazzi, SOFA, ArticBeans, PECOS, Draco, Wcomp, Rubus, Koala, PACCPin, OLAN, Newton, COSMOS, Java/A, HK2, SOFA

1. **Modèle COM** (Microsoft): Approche boite noire : aucune information en dehors des interfaces. Assemblage par satisfaction d'interface.

- Le modèle COM définit un standard pour la communication des objets : assemblage de composants multi éditeurs et multi langages.
  - ✓ Il désigne les services de base du broker d'objet.

- La technologie ActiveX a été développée par dessus COM, pour désigner :
  - ✓ les composants destinés à Internet,
  - ✓ des contrôles graphiques (OCX),
  - ✓ et permettre le pilotage d'application.

- Le modèle COM+ est une évolution de COM à partir de Windows 2000.
  - ✓ Il intègre de nouvelles fonctionnalités :
    - La gestion d'événement en mode publish et subscribe,
    - des communications asynchrones et sûres entre composants (MSMQ : Message Queuing 3.0),
    - Les transactions distribuées (MTS).

- Du point de vue architecture de déploiement, les composants métiers peuvent être sollicités par :
  - ✓ des clients lourds *via* :
    - le protocole DCOM,
    - Le protocole HTTP, au moyen d'IIS.
  - ✓ Des clients légers, tels que des pages HTML via les ASP.

Difference avec les autres modèle de composants
Le choix de telle ou telle architecture doit se faire par rapport aux besoins de l'application. Pour une application dédiée au système d'exploitation de Microsoft, .NET est le choix évident. Si un composant doit obligatoirement présenter plusieurs interfaces, les efforts devront s'orienter vers une implantation CCM. Si le composant doit être déployé sur plusieurs machines différentes (ou se déplacer entre ces machines), EJB remporte haut la main.
Enfin, l'architecture .NET est plus que quant à la spécification du composant, et nous la situons plutôt entre EJB et CCM : l'objet distant .NET peut être issu d'une composition, mais n'a pas la notion de ports comme les CCM. Bien qu'étant plus complète, la spécification CCM est complexe comparée aux EJB : plusieurs vues du composant, plusieurs descripteurs, et une technologie basée sur le bus Corba qui est parfois considéré comme assez lourd.

2. **Modèle EJB** (Sun): Orienté trois tiers, coté serveur. Mono-langage
- Le modèle Enterprise JavaBeans est basé sur le concept *Write Once, Run Everywhere* pour les serveurs.
- Le modèle EJB repose sur l'architecture en couches suivante :
  - ✓ L'EJB Server contient l'EJB Container et lui fournit les services de bas niveau.
  - ✓ L'EJB Container est l'environnement d'exécution des composants Enterprise JavaBeans (interface entre le bean et l'extérieur).
  - ✓ Les clients ne se connectent pas directement au bean, mais à une représentation fournie par le conteneur. Celui-ci route les requêtes vers le bean.

- Un Bean Enterprise (EJB ou beans) est un composant chargé des opérations métiers de l 'application.
- L 'architecture est découpée en quatre tâches:
  - ✓ le développement du bean enterprise,
  - ✓ l 'assemblage de l'EJB (JAR),
  - ✓ le déploiement de l'EJB dans un environnement d 'exécution,
  - ✓ l 'administration et la configuration de l'EJB

Difference avec les autres modèle de composants
L'architecture la plus utilisée, et la plus ancienne est sans conteste EJB. Cependant, la plus complète reste la spécification CCM, où nous avons véritablement la notion qu'un composant

peut présenter plusieurs ports ; le modèle du composant reste plat, même s'il peut être "segmenté" en plusieurs classes d'implantations, cette segmentation étant visible de l'extérieur ; les EJB et .NET n'offrent pas cette transparence. Cependant, CCM et EJB offrent la possibilité de déployer un assemblage de composants, mais ce n'est pas aussi complet qu'un composant composite.
Un point négatif pour EJB est que la spécification contraint le langage de programmation; ce n'est pas le cas pour CCM, ni pour .NET.

3. **Corba-CCM** (OMG): Amélioration du précédent sur le bus à objet CORBA
Le CORBA COMPONENT MODEL est un modèle pour definir un composant dans une architecture distribuée CORBA.Il est compatible avec les structure EJB (Enterprise Java Beans).

Ce modèle propose aussi bien une architecture pour le composant qu'une API pour l'implémentation CORBA. Il integre aussi des interfaces pour la configuration, la definition de la composition des composants et un modele pour le deploiement

Quel est l'interet pour un composant de s'integrer dans une architecture CORBA ? D'une part cela lui permet d'etre independant de la plate forme et du langage, compatible avec le modèle EJB. D'autre part ce composant peut etre distribué sur un bus CORBA (description du déploiement) en bénéficiant de services CORBA ;la gestion d'événements par exemple .

Le CCM propose toute une structure pour définir un composant, son comportement, son intégration dans un conteneur (application) et son déploiement dans l'environnement distribué CORBA.
Le CCM a été intégré dans la version de CORBA 3 .

Difference avec les autres modèle de composants[(6)]

- CCM fournit un cadre global pour la construction d'applications à base de composants distribués
    - Spécification, implantation, conditionnement, assemblage, déploiement et exécution des composants
- Répartition des assemblages de composants CORBA
    - Peuvent être déployés et s'exécutés sur différentes machines

- Segmentation de l'implantation des composants
    - Plusieurs classes au lieu d'une seule
    - Un état persistant par segment possible

4. **Autre composants**

Fractal
Fractal impose une séparation stricte entres les interfaces et leurs implémentations
Il est suffisamment générale pour être appliqué à l'ingénierie des systèmes et du middleware et à tout autre domaine. Il est conçu dans le but de faciliter et d'unifier la conception, développement, déploiement et l'administration

Difference avec les autres modèle de composants

- ✓ Hièrarchie de composition
- ✓ Réflexion
- ✓ Pas de granularité à priori.

- ✓ Idéalement, compatible avec d'autres
- ✓ implémentations de composants (par ex. CCM).
- ✓ Une orientation Java assez marquée, bien que les
- ✓ concepteurs s'en défendent.

F) L'impact du paradigme des objets distribues dans les serveurs d'applications

Le paradigme des objets distribués dans les serveurs d'applications peut permettre la séparation des informations concernant un même objet , chaque composant ayant alors accès aux informations qui lui sont nécessaire, améliorant ainsi la sécurité des informations.

G) La plateforme PHP, ZOPE et Tomcat

On peut classifier ces plateformes en tent que serveurs d'applications web, specifiquement PHP et ZOPE (qui implémente le langage Python). Tomcat se semble être limiter à un conteneur web, car il implement les servlets et JSP mais non pas les EJB. Pour son bon fonctionnement, il necessite donc d'etre accompagne par un second logiciel qui assura cette fonctiionalité.

Pour PHP, il s'agit globalement d'une plateforme qui offre un langage et un environnement d'execution implementant plusieurs bibiliotheques pour developper des sites web et des applications.

Question 2: Expérimentations

A) Les differents types de composants EJB 2

Les composants EJB se classent dans trois catégories :

1. Session Beans (Stateless/Stateful): Les EJB sessions sont des objets proposant des services à leur appelant. Ils proposent un certain nombre de méthodes écrites par le développeur. Il y a deux types d'EJB session: les EJB sessions ne conservant pas leur état entre deux appels (EJB dit "state less"), et ceux le conservant (EJB dit "stateful"). Il n'y a aucune garantie qu'entre deux appels au même EJB l'instance de l'objet soit la même.
2. Entity Beans: Les EJB entité sont des beans ayant majoritairement pour vocation d'être persistants, c'est-à-dire pouvant être stockés sur un support physique entre deux sessions. Ils peuvent être de deux sortes : BMP (Bean Managed Persistence) ou CMP (Container Managed Persistence)
3. Message Driven Beans: Depuis la norme EJB 3.0, cette architecture propose un troisième type de composant : les EJB message permettant de déclencher un processus côté serveur applicatif lors de la publication d'un message asynchrone.

B) Installation et deploiment de Jonas

Le déploiement, configuration et test de Jonas a été effectué en suivant les étapes précisé dans le site web[8].

C) Developpement d'un EJB simple

On a divisé les etapes de réalisation d'un EJB permettant d'afficher bonjour en mode console de la manière suivante:

1. Etape 1 - Definition de l'interface distante

Cette interface définit toutes les fonctions rattachées à l'EJB dont le client peut acceder. C'est utile de noter que l'interface Hello contient seuelement une seule fonction *hello()* qui doit etre **public** et doit heriter de l'interface **javax.ejb.EJBObject**.

```
import java.rmi.*;
import javax.ejb.*;
public interface Hello extends EJBObject {
public String hello()
throws RemoteException;
}
```

Figure 2 - Interface Hello

2. Etape 2 - Definiton de l'interface locale

L'interface locale *HelloHome* va permettre au client a communiquer avec l'interface distante en creant une référence sur *HelloHome*.


{EM}
Contact me for the full version
em@eliematta.com