Chapitre 2 : Architecture N-Tiers

1. Introduction

Le modèle Client/Serveur a ses limites et inconvénients :

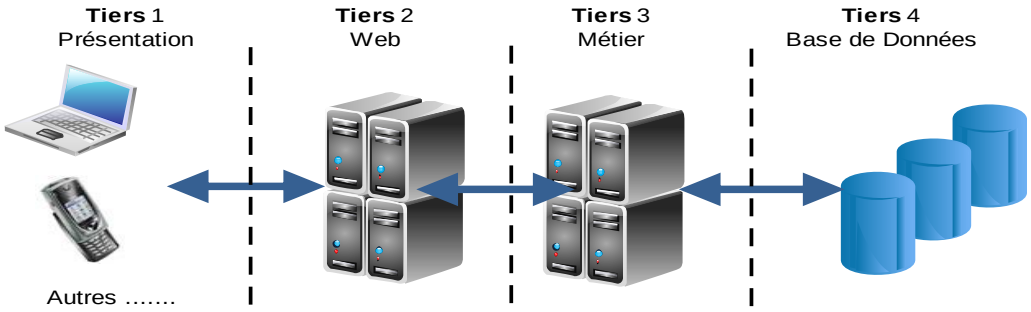
- Le client est en général lourd
- Il supporte la majorité de la logique applicative en plus de la présentation
- Il est très sollicité
- Il nécessite une mise à jour et maintenance régulière afin de répondre aux besoins des utilisateurs
- La tache du serveur en général se réduit à la gestion de données (SGBD)

➤ Nouvelle architecture : Architecture N-Tiers (Multitiers, Multiniveaux, Multiétage)

2. Architecture N-Tiers

- Les Tiers sont issus d'un découpage logique d'application
- L'architecture N-Tiers est conforme au découpage
- Les Tiers sont distribués logiquement et éventuellement physiquement
- Le développement, en général, est a base de composants softwares (Beans, …)
- Exemples d'architectures : J2EE, .NET, iOSGI, …

- Principaux Tiers : Présentation, Web, Métier (Business), Base de données
- Présentation : La logique présentation de l'application, dépend du type de client
- Web : Médiateur entre couche présentation et métier
- Métier : La logique de l'application métier (Code propre à l'application)
- Base de données ou EIS : Données nécessaires à l'application



3. Architecture 3-Tiers

- Différentes architectures
- Architecture 1 : Client léger
  Tier 1 : Présentation Tier 2 : Web + Métier Tier 3 : BD
  Exemple : Navigateur Web (Explorer,Firefox, …)
  Serveur Web (Apache, IIS, …)
  Serveur de base de données (Oracle, MySql, …)

- Architecture 2 : Client lourd
  Tier 1 : Présentation + Web Tier 2 : Métier Tier 3 : BD
  Exemple : Applet Java (JVM)
  Serveur Web
  Serveur de base de données
- Inconvénients et limites : Similaires au Client/Serveurs (Surcharge d'un tier)

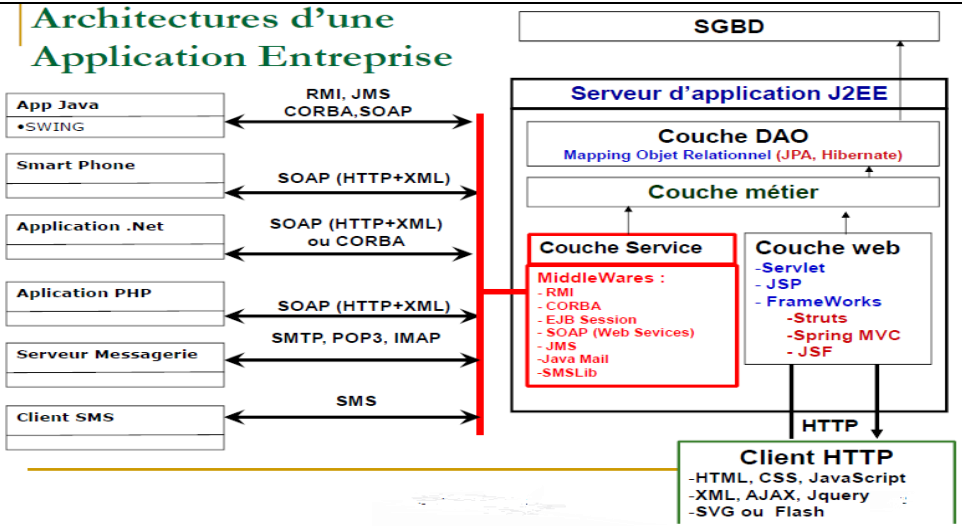
4. Exemples Architectures N-Tiers

- Plusieurs Architectures : Dépendent du tiers et le propriétaire
- J2EE et .NET (Microsoft) sont les architectures n-tiers les plus utilisées

Architecture	Web	Métier	DAO	BD
J2EE	Servlet,JSP, JSF, …	EJB, Java classe	JDBC	ORACLE SQL-Server

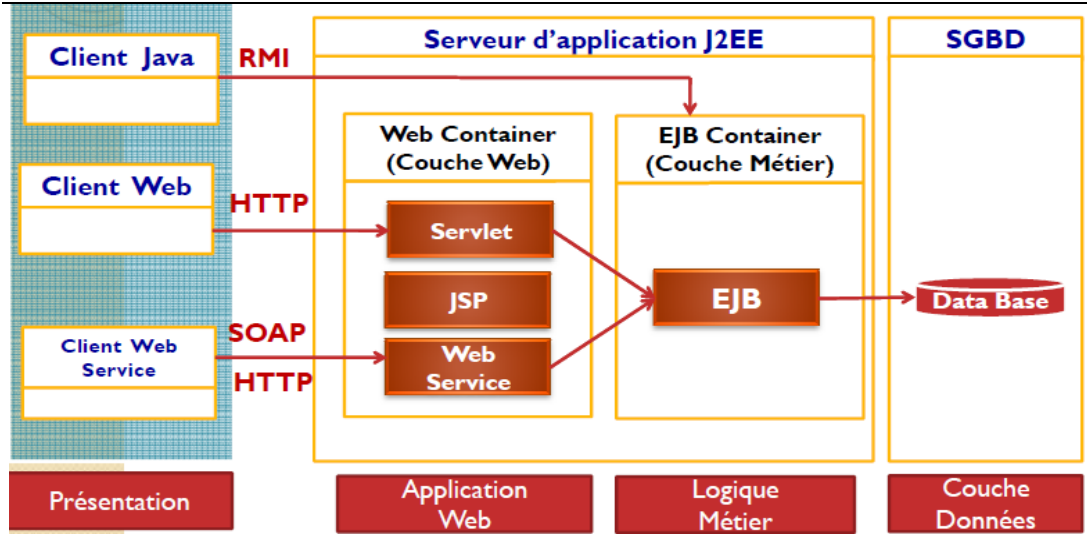
.NET	ASP, …	C#, VB,C++	ADO.NET	MySQL ……

Architectures d'une Application Entreprise



5. Architectures J2EE

- J2EE : Architecture N-Tiers a base de Java
- J2EE offre un environnement pour développer, déployer et exécuter des applications réparties dans le monde de l'entreprise
- Une architecture à base de composants (Un ou plusieurs objets) : EJB (Enterprise Java Beans)
- La couche métier : Un ensemble de composants
- Les composants peuvent êtres reparties sur des machines différentes
- Le conteneur fournit l'infrastructure d'exécutions des composants : Creation, Destruction, Activation, …
- Le conteneur est similaire au système d'exploitation pour les processus (J2SE : Un conteneur)
- Le serveur d'application prend en charge : Couche Web et Couche Métier (Serveur EJB)
- Serveurs d'application : Weblogic, JBoss, Jonas, Glassfish, Tomcat, Websphere, …
- Les clients EJB sont de divers types : EJB ou autres.
- Le client EJB peut être situé dans la même machine (Local) ou une autre machine (Remote)
- Les clients sont de divers types et utilisent diverses technologies d'interaction


Client Java
RMI
Serveur d'application J2EE
SGBD
Web Container
(Couche Web)
EJB Container
(Couche Métier)
Client Web
HTTP
Servlet
JSP
EJB
Data Base
Client Web
Service
SOAP
HTTP
Web
Service
Présentation
Application
Web
Logique
Métier
Couche
Données

5.1- Couche Présentation

La couche présentation est liée au type de client utilisé :

◦ Client Lourd java Desktop:

- Interfaces graphiques java SWING, AWT, SWT.
- Ce genre de client peut communiquer directement avec les composants métiers déployés dans le conteneur EJB en utilisant le middleware RMI

◦ Client Leger Web

- HTML, Java Script, CSS.
- Un client web communique avec les composants web Servlet déployés dans le conteneur web du serveur d'application en utilisant le protocole HTTP.

◦ Un client .Net, PHP, C++, …

- Ce genre de clients développés avec un autre langage de programmation autre que java, communiquent généralement avec les composants Web Services déployés dans le conteneur Web du serveur d'application en utilisant le protocole SOAP (HTTP+XML)

◦ Client Mobile

- Android, iPhone, Tablette etc..
- Généralement ce genre de clients communique avec les composants Web Services en utilisant le protocole HTTP ou SOAP

5.2- Couche Web

La couche Web sert de médiateur entre la couche présentation et la couche métier.

- Elle contrôle l'enchainement des tâches offertes par l'application
  - Elle reçoit les requêtes http clientes
  - Assure le suivi des sessions
  - Vérifier les autorisations d'accès de chaque session
  - Assure la validation des données envoyées par le client
  - Fait appel aux composants métier pour assurer les traitements nécessaires
  - Génère une vue qui sera envoyée à la couche présentation.
- Elle utilise les composants web Servlet et JSP
- Elle respecte le modèle MVC (Modèle Vue Contrôleur)
- Des framework comme JSF, SpringMVC ou Struts sont généralement utilisés dans cette couche.

5.3- Couche Métier

La couche métier est la couche principale de toute application

- Elle implémente la logique métier d'une entreprise
- Elle se charge de récupérer, à partir des différences sources de données, les données nécessaires pour assure les traitements métiers déclenchés par la couche Web.
- Elle assure la gestion du WorkFlow (Processus de traitement métier en plusieurs étapes)

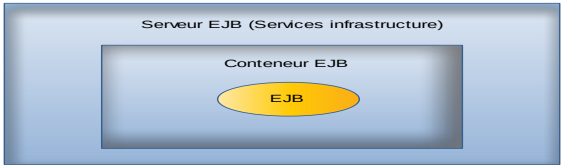
Il est cependant important de séparer la partie accès aux données (Couche DAO) de la partie traitement de la logique métier (Couche Métier) pour les raisons suivantes :
Ne pas se perdre entre le code métier, qui est parfois complexe, et le code d'accès aux données qui est élémentaire mais conséquent.

- Ajouter un niveau d'abstraction sur l'accès aux données pour être plus modulable et par conséquent indépendant de la nature des unités de stockage de données.
- La couche métier est souvent stable. Il est rare qu'on change les processus métier. Alors que la couche DAO n'est pas stable. Il arrive souvent qu'on est contrait de changer de SGBD ou de répartir et distribués les bases de données.
- Faciliter la répartition des tâches entre les équipes de développement.
- Déléguer la couche DAO à un framework spécialisé dans l'accès aux données (Hibernate, Toplink, etc…)

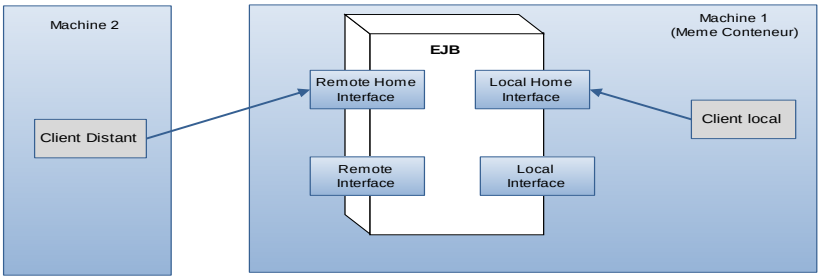
6. Le serveur EJB

- Le conteneur gère le cycle de vie de l'EJB
- Le serveur EJB fournit les services d'infrastructure au conteneur
- Parmi les services

  JDBC : Middleware d'accès aux BD
  JTA : Service de gestion des transactions
  JCA : Service de connexion à des systèmes externes
  JMS : Service de communication asynchrone de messages
  JPA : Service de persistance
  Etc …



7. Le composant EJB



- EJB : Classe java
- EJB possède des interfaces : Local et Remote
- Local invoqué par un client local, situé dans la même machine et même conteneur
- Remote invoqué par un client distant, situé dans une autre machine
- Remote et Local : les méthodes Métier
- Remote Home et Local Home : Gestion du cycle de vie de l'EJB (Gestion du composant)

8. Types de composant EJB

- Session Bean
  - Modélise un traitement métier
  - Une classe Java et interface (Remote/Local) qui expose des méthodes métier
  - Deux types de session bean

    Stateless (Sans état) : Ne mémorise pas l'état entre appels consécutifs

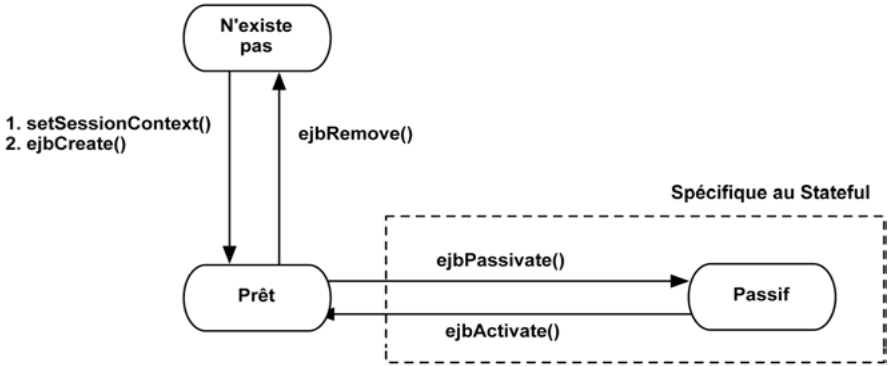
    Ex : Traiter des requêtes de plusieurs clients (Liste des produits)

    Statefull (Avec état) : Mémorise l'état

    Ex : Maintenir une conversation avec un client

    Remplir plusieurs formulaires (Plusieurs pages Web) avant la validation
- Entity Bean
  - Modélise les objets persistant de l'application (JPA)
  - Classe représentant des objets de la BD : Table de la BD
- Message Driven Bean
  - Traitement de messages d'autres applications (Messagerie)
  - Communication grâce à JMS
  - Sans état
  - Une seule méthode et pas d'interfaces

9. Cycle de vie d'un EJB session



10. Exemple de EJB session

```
public class CalculatriceBean implements javax.ejb.SessionBean
{

   /* Méthodes de l'interface Remote. */
   public double add(double v1,double v2) {return v1+v2;}
   public double sub(double v1,double v2) {return v1-v2;}
   public double mul(double v1,double v2) {return v1*v2;}
   public double div(double v1,double v2) {return v1/v2;}

/* Méthodes de l'interface RemoteHome. */
   public void ejbCreate() throws CreateException {}
   public void ejbActivate() {}
   public void ejbPassivate() {}
   public void ejbRemove() {}
   public void setSessionContext(SessionContext sc) {}
}
```