PARTIE 1 : J2EE - JAVA 2 ENTERPRISE EDITION

1. Introduction

Le «**Java Framework**» (*Java 2 Platform*) est composé de trois éditions, destinées à des usages différents :

- **J2SE** : *Java 2 Standard Edition* est destiné au développement d'applications pour ordinateurs personnels ;
- **J2EE** : *Java 2 Enterprise Edition*, destiné à un usage professionnel avec la mise en oeuvre de serveurs.
- **J2ME** : *Java 2 Micro Edition* est prévu pour le développement d'applications embarquées, notamment sur des assistants personnels et terminaux mobiles ;

Chaque édition propose un environnement complet pour le développement et l'exécution d'applications basées sur Java et comprend notamment une machine virtuelle Java (Java virtual machine) ainsi qu'un ensemble de classes.

J2EE(*Java 2 Enterprise Edition*) est une plate-forme fortement orientée serveur pour le développement et l'exécution d'applications distribuées (applications d'entreprises basées sur des composants).

. Elle est composée de deux parties essentielles :

- un ensemble de spécifications pour une infrastructure dans laquelle s'exécutent les composants écrits en Java : un tel environnement se nomme serveur d'application.
- un ensemble d'API qui peuvent être obtenu et utilisé séparément. Pour être utilisées, certaines nécessitent une implémentation de la part d'un fournisseur tiers.

L'utilisation de J2EE pour développer et exécuter une application propose plusieurs avantages :

- une architecture d'application basée sur les composants qui permet un découpage de l'application et donc une séparation des rôles lors du développement
- la possibilité de s'interfacer avec le système d'information existant grâce à de nombreuses API : JDBC, JNDI, JMS.. ...
- la possibilité de choisir les outils de développement et le ou les serveurs d'applications utilisés qu'ils soient commerciaux ou libres

L'architecture d'une application se découpe idéalement en au moins trois tiers :

- la partie cliente : c'est la partie qui permet le dialogue avec l'utilisateur. Elle peut être composée d'une application standalone, d'une application web ou d'applets
- la partie métier : c'est la partie qui encapsule les traitements (dans des EJB ou des JavaBeans)
- la partie données : c'est la partie qui stocke les données

On parle généralement de «plate-forme J2EE» pour désigner l'ensemble constitué des services (API) offerts et de l'infrastructure d'exécution. J2EE comprend notamment :

- Les spécifications du **serveur d'application**, c'est-à-dire de l'environnement d'exécution : J2EE définit finement les rôles et les interfaces pour les applications ainsi que l'environnement dans lequel elles seront exécutées. Ces recommandations permettent ainsi à des entreprises tierces de développer des serveurs d'application conformes aux spécifications ainsi définies, sans avoir à redévelopper les principaux services.
- Des services, au travers d'**API**, c'est-à-dire des extensions Java indépendantes permettant d'offrir en standard un certain nombre de fonctionnalités. *Sun* fournit une implémentation minimale de ces API appelée **J2EE SDK** (*J2EE Software Development Kit*).

2. Les API de J2EE

Les API de J2EE peuvent se répartir en trois grandes catégories :

- Les **composants**. On distingue habituellement deux familles de composants :
  - Les composants web : **Servlets et JSP** (Java Server Pages). Il s'agit de la partie chargée de l'interface avec l'utilisateur (on parle de *logique de présentation*).

- Les composants métier : **EJB (Enterprise Java Beans**). Il s'agit de composants spécifiques chargés des traitements des données propres à un secteur d'activité (on parle de *logique métier* ou de *logique applicative*) et de l'interfaçage avec les bases de données.

- Les **services**, pouvant être classés par catégories :
  - Les **services d'infrastructures** : il en existe un grand nombre, définis ci-dessous :
    - **JDBC** (*Java DataBase Connectivity*) est une API d'accès aux bases de données relationnelles.
    - **JNDI** (*Java Naming and Directory Interface*) est une API d'accès aux services de nommage et aux annuaires d'entreprises tels que DNS, NIS, LDAP, etc.
    - **JTA**/**JTS** (*Java Transaction API*/*Java Transaction Services*) est un API définissant des interfaces standard avec un gestionnaire de transactions.
    - **JPA : Java Persistence API**
    - **JMX** (*Java Management Extension*) fournit des extensions permettant de développer des applications web de supervision d'applications.
    - **JAX-WS : Java API for XML Web services**
    - **Java API for XML Parsing (JAXP)** Analyse et exploitation de données au format XML
    - **Java API for XML-based RPC (JAXP-RPC)**
    - **SOAP with Attachments API for Java (SAAJ**)
  - Les **services de communication** :
    - **JAAS** (*Java Authentication and Authorization Service*) est une API de gestion de l'authentification et des droits d'accès.
    - **JavaMail** est une API permettant l'envoi de courrier électronique.
    - **JMS** (*Java Message Service*) fournit des fonctionnalités de communication asynchrone (appelées *MOM* pour *Middleware Object Message*) entre applications.
    - **RMI-IIOP** est une API permettant la communication synchrone entre objets.
      RMI permet d'utilisation d'objet Java distribué. RMI-IIOP est une extension de RMI pour utiliser avec CORBA.

L'architecture J2EE permet ainsi de séparer la couche présentation, correspondant à l'interface homme-machine (IHM), la couche métier contenant l'essentiel des traitements de données en se basant dans la mesure du possible sur des API existantes, et enfin la couche de données correspondant aux informations de l'entreprise stockées dans des fichiers, dans des bases de données relationnelles ou XML, dans des annuaires d'entreprise ou encore dans des systèmes d'information complexes.

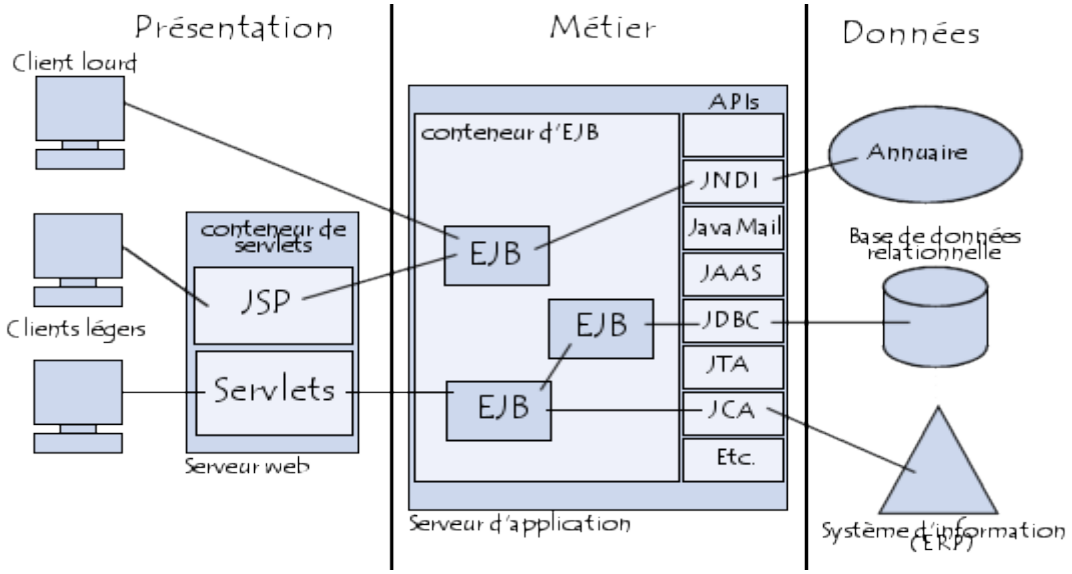


Figure 1. Architecture d'une plate-forme J2EE

Quelques serveurs certifiés J2EE : GlassFish (implémentation de référence réalisé par Sun Microsystems), **JBoss**, JOnAS , **Weblogic** ,**Websphere**

3. L'environnement d'exécution des applications J2EE

J2EE propose des spécifications pour une infrastructure dans laquelle s'exécutent les composants. Ceci permet de séparer les applications et l'environnement dans lequel il s'exécute. Pour exécuter ces composants de natures différentes, J2EE définit des **conteneurs p**our chacun de ces composants. Il définit pour chaque composant des interfaces qui leur permettront de dialoguer avec les composants lors de leur exécution. Les conteneurs permettent aux applications d'accéder aux ressources et aux services en utilisant les API.

Les appels aux composants se font par des clients via les conteneurs. Les clients n'accèdent pas directement aux composants mais sollicite le conteneur pour les utiliser.

4. Les conteneurs

Les conteneurs fournissent des services qui peuvent être utilisés par les applications lors de leur exécution.

Il existe plusieurs conteneurs définit par J2EE:

- conteneur web : pour exécuter les servlets et les JSP
- conteneur d'EJB : pour exécuter les EJB
- conteneur client : pour exécuter des applications standalone sur les postes qui utilisent des composants J2EE

Les serveurs d'applications peuvent fournir un conteneur web uniquement (exemple : Tomcat) ou un conteneur d'EJB uniquement (exemple : JBoss, Jonas, ...) ou les deux (exemple : Websphere, Weblogic, ...).

Pour déployer une application dans un conteneur, il faut lui fournir deux éléments :

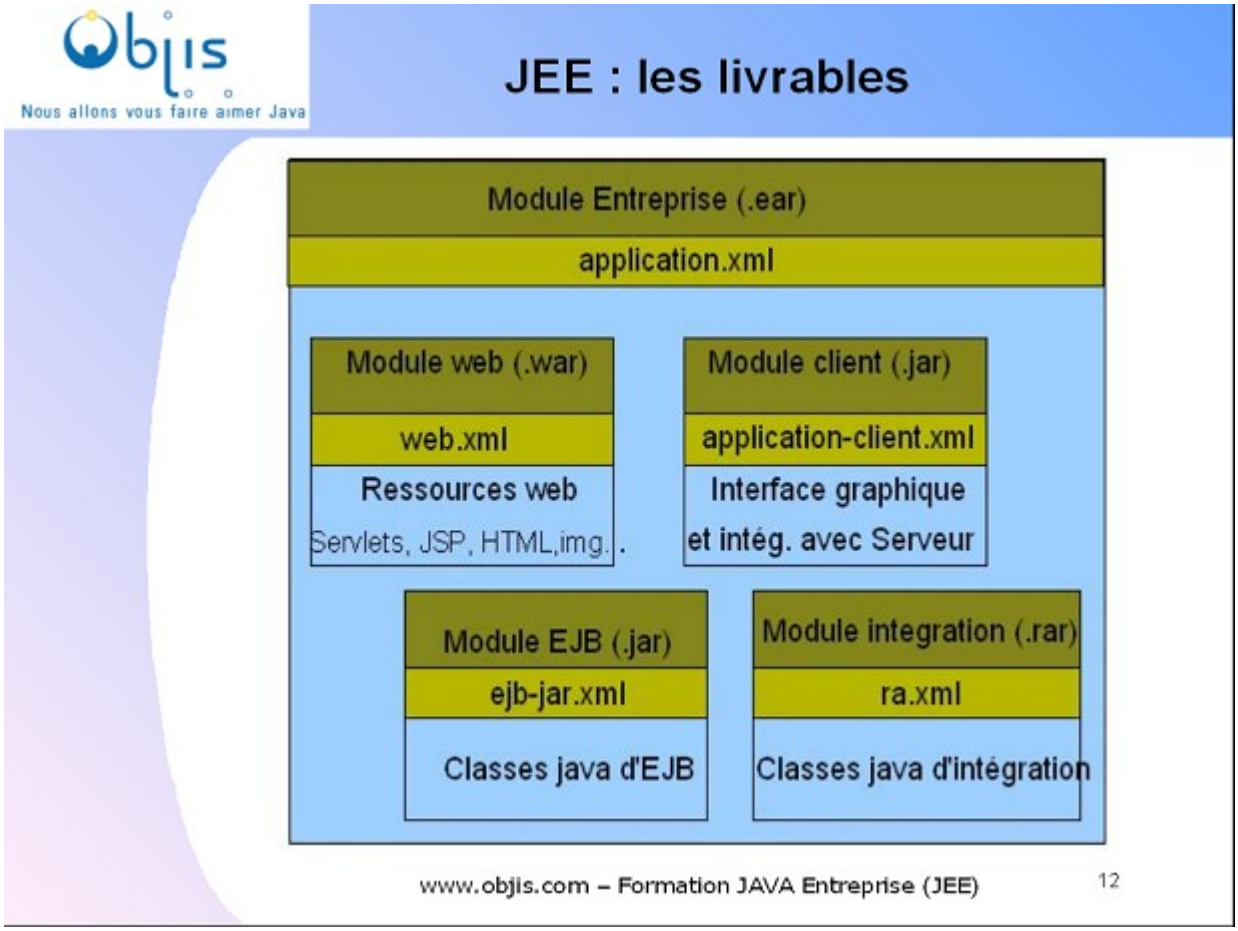
- l'application avec tous les composants (classes compilées, ressources ...) regroupée dans une archive ou module. Chaque conteneur possède son propre format d'archive.
- un fichier descripteur de déploiement contenu dans le module qui précise au conteneur des options pour exécuter l'application

Il existe trois types d'archives :

Archive / module	Contenu	Extension	Descripteur de déploiement
bibliothèque	Regroupe des classes	jar	
application client	Regroupe les ressources nécessaires à leur exécution (classes, bibliothèques, images, ...)	jar	application-client.jar
web	Regroupe les servlets et les JSP ainsi que les ressources nécessaires à leur exécution (classes, bibliothèques de balises, images, ...)	war	web.xml
EJB	Regroupe les EJB et leurs composants (classes)	jar	ejb-jar.xml

Une application est un regroupement d'un ou plusieurs modules dans un fichier EAR (Entreprise ARchive). L'application est décrite dans un fichier application.xml lui même contenu dans le fichier EAR

Les livrables J2EE



Le descripteur de déploiement J2EE

Pour déployer des applications J2EE sur le serveur d'applications, toutes les informations doivent être rassemblées dans un et un seul descripteur de déploiement contenu dans un fichier au format XML
Ce descripteur de déploiement doit être enregistré avec le nom *META-INF/application.xml* dans le fichier ear.
Le descripteur de déploiement standard doit contenir les informations structurelles suivantes :

- Composants EJB,
- Composants Web,
- Composants Client,
- Descripteur de déploiement alternatif pour ces composants,
- Rôle de sécurité.

Exemple simple de descripteur de déploiement d'application

```
<?xml version="1.0" encoding="UTF-8"?>

<application xmlns="http://java.sun.com/xml/ns/j2ee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
        xsi:schemaLocation="http://java.sun.com/xml/ns/j2ee
        http://java.sun.com/xml/ns/j2ee/application_1_4.xsd"
        version="1.4">

 <display-name>Simple example of application</display-name>
 <description>Simple example</description>

 <module>
  <ejb>ejb1.jar</ejb>
 </module>
 <module>
  <ejb>ejb2.jar</ejb>
 </module>

 <module>
  <web>
   <web-uri>web.war</web-uri>
   <context-root>web</context-root>
  </web>
 </module>
</application>
```

OUTILS DE DEVELOPPEMENT DES APPLICATIONS J2EE

- la plateforme j2ee (j2eesdk)
- l'EDI Eclipse WTP (Web Tools Platform)
- Eclipse version jee