

AI SL / GLG203 – GLG204

Architectures Logicielles Java

Présentation JEE

JEE

Architecture de JEE

Objectifs de JEE

Les Composants

Les Conteneurs

Serveur JEE

Applications JEE

Architecture de JEE

Les composants logiciels ou Beans : implantent la logique des applications sous la forme de services ou d'objets orientés «métier».

Conteneurs : ils isolent les Beans accessibles par le client d'une implémentation spécifique de serveur. Ils fournissent les aspects non fonctionnels utilisés par les applications

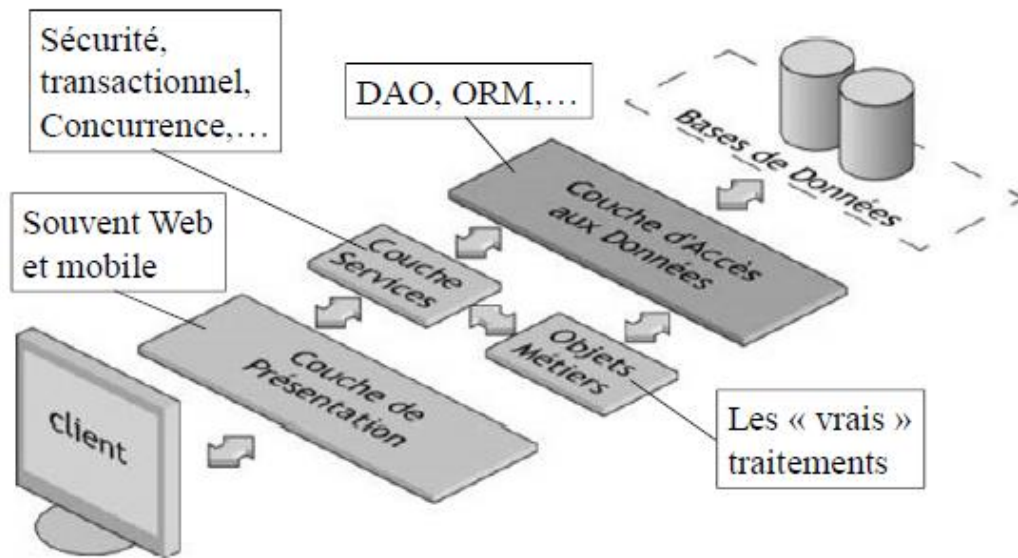
Serveurs JEE : ils assurent l'accès aux services et hébergent les conteneurs. Ils implémentent également les aspects non fonctionnels utilisés par les applications

Clients : obtiennent et utilisent les services mis à disposition

Architecture de JEE

Les Architectures actuelles : N-Tiers

JEE est juste une spécification pour l'écriture d'applications Java d'entreprise



Architecture à base de composants : **les EJB**

Couche présentation (application Web))

Java Server Faces (JSF)

Java Server Page (JSP)

Les Services non fonctionnels

Sécurité

Transactionnel,
etc.

Les objets métiers

Traitements spécifiques (Services)

Accès aux données et persistance (DAO)

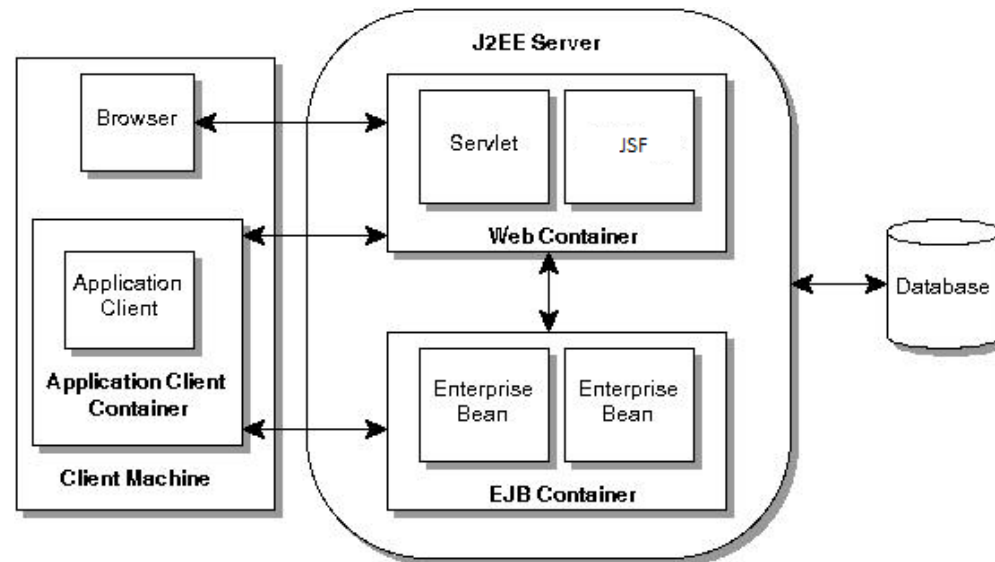
Architecture de JEE: Les clients

Les clients des applications JEE :

Client Web: **Navigateur** (JSF, JSP, Ajax)

Application Cliente JEE (nécessite API clientes JEE)

Application mobile utilisant des web services REST ou SOAP. (Mise en œuvre simplifié Depuis JEE 7)



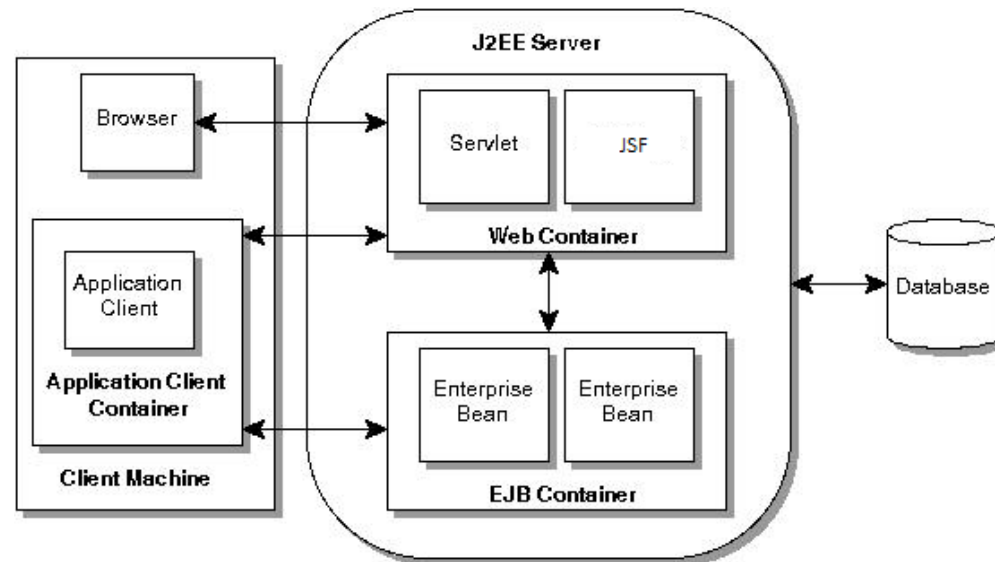
Architecture de JEE: Les EJB

EJB pour **Enterprise Java Bean**

Ils implantent la logique métier sous la forme de « **Services** » et d'objets métiers « **Entité** ».

Ils sont hébergés et gérés par les Conteneurs

Plusieurs rôles: traitements fonctionnels ou non fonctionnels, accès aux données



Objectifs de JEE

Faciliter le développement d'applications Java à base de **Composants** (EJB, JSF) :

Préserver l'interopérabilité avec le « **Legacy** » de l'entreprise (Tiers ressources) :

Fiabiliser les applications par l'exploitation de **serveurs d'applications**

JEE:

- Répartition des composants entre plusieurs serveurs
- Tolérance aux pannes: redondance logicielle et matérielle
- Supporter la montée en charge (scalability)
- Sécurité
- Transactions locales ou distribuées

Dernière version: **JEE 8** depuis septembre 2017

Les composants

Principe de « **boite noire** » : exposent uniquement leur **interface**

Réutilisables : implémenté 1 fois utilisé N fois

Configurables (composant JSF)

Respectent des interfaces (ou @marquage) définies dans les spécifications JEE

Utilisables par les applications qui respectent les spécifications JEE

Les composants : exemples

Un **EJB** chargé de calculer le tarif d'un **Produit** pour un **Client**: le « **TarifService** »

*Le **TarifService** met à disposition des méthodes pour le calcul du Tarif (Service Métier)*

*Il effectue ces traitements en utilisant les EJB « Entité » **Produit** et **Client**.*

*Il est utilisable par tous les autres **composants** de l'application.*

Un Composant **JSF** « **FicheArticle** » chargé d'afficher la fiche descriptive d'un **Produit** (Web)

*Le composant **FicheArticle** est configurable (choix de l'article à afficher)*

*Il est **réutilisable** sur toutes les pages de l'application*

*Une **instance** de ce composant est créée à **chaque utilisation***

*IL peut y avoir **plusieurs instances** de ce composant sur la même page Web*

Les composants : intérêts

Rapidité de développement

Fiabilité (code découpé en fonctions de petites tailles)

Facilité de la maintenance (composant par thématique: Service Métier)

Forte réutilisation

Encapsulation des règles métiers (utilisables sans connaissance approfondies)

Substituable par un autre composant qui implémente la même interface

Les composants : Architecture

Composants **EJB** de l'application découverts par décomposition du système
(Analyse desc.)

- **Services Applicatifs** : Réalisation des cas d'utilisation en utilisant les services
- **Services Métiers** : comportent les fonctions de base du système
- **Services d'accès aux données** : Persistance CRUD + Requêtes nécessaires aux services métiers

Couche présentation (issue de l'analyse des cas d'utilisation) :

- **Composants JSF**
- **Pages JSF**
- **Les ManagedBean** (contrôleur/modèle au sens MVC des pages et composants)

Les Conteneurs (de composants)

Rôles :

- Gérer le cycle de vie des composants
- Instrumenter les composants pour assurer les services non fonctionnels
- Interception des appels

Différents types de conteneurs : selon la type de composant (EJB, JSF):

- EJB Container
- Web Container

Intérêt : les conteneurs assurent un ensemble de fonctionnalités techniques (libère le développeur de ces traitements)

Les Conteneurs: 2 exemples

Gestion transactionnelle réalisée par le container d'EJB :

- Appel d'une méthode d'un EJB géré par le container
- Le conteneur intercepte l'appel et démarre une nouvelle transaction
- Le code de la méthode est exécuté
- A la fin de la méthode le conteneur valide la transaction

Gestion du cycle de vie d'un EJB (IOC) :

- Intercepte l'instanciation et appelle le callback associé
(**@PostConstruct** : pour permettre l'initialisation l'EJB)
- Intercepte la destruction et appelle avant le callback associé
(**@PreDestroy** : pour effectuer des libération de ressource avant la destruction de l'instance de l'EJB)

Serveur d'application JEE (1)

Héberge des applications JEE et fournit les services non fonctionnels:

JDBC (Java DataBase Connectivity) API d'accès aux bases de données (Pool de connexions)

JNDI (Java Naming and Directory Interface) accès aux services de nommage et aux annuaires d'entreprises.

JCA (JEE Connector Architecture) API de connexion au système d'information de l'entreprise comme les ERP.

JTA/JTS (Java Transaction API/Java Transaction Services) API pour la gestion de transactions.

JMX (Java Management Extension) API permettant de développer des applications de supervision d'applications

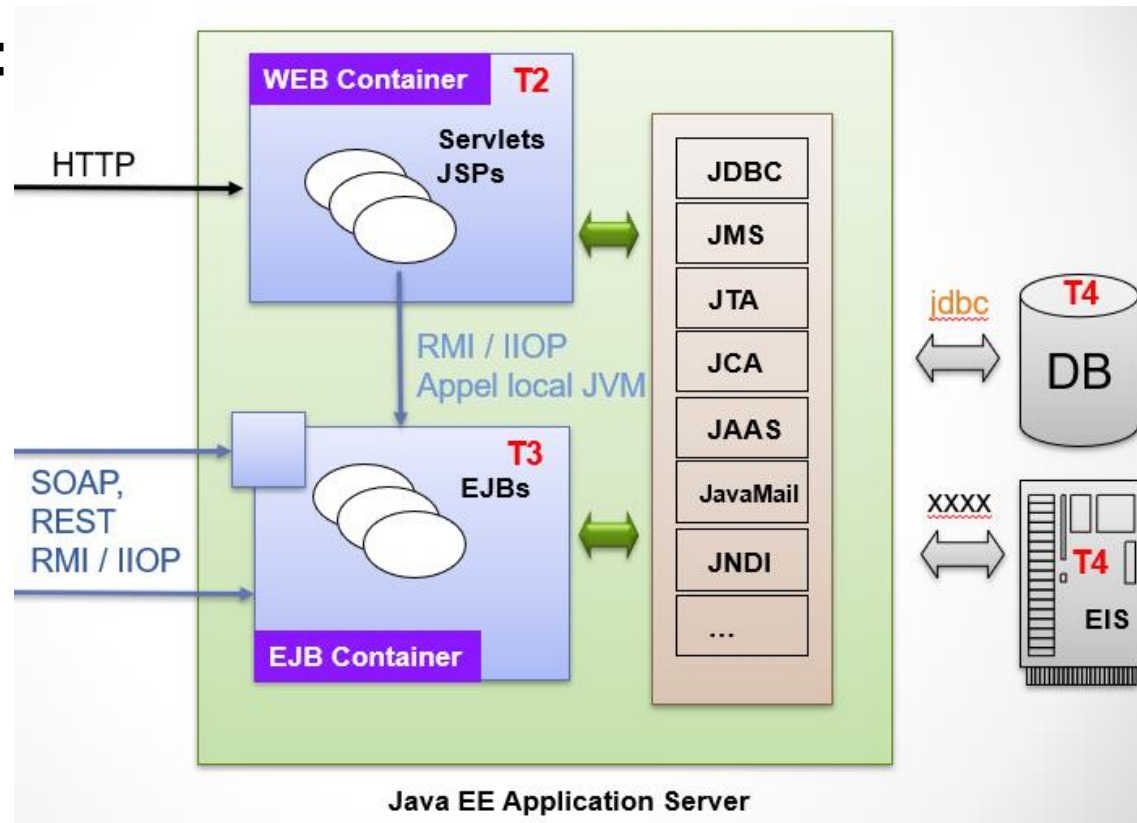
JAAS (Java Authentication and Authorization Service) API de gestion de l'authentification.

JavaMail API pour la gestion de courrier électronique.

JMS (Java Message Service) API de communication asynchrone à base de message entre applications.

Serveur d'application JEE (2)

Héberge des applications JEE et fournit les services non fonctionnels:



Serveur d'application JEE ⁽³⁾

Implémentations:

- La référence: Glassfish (JEE 7: GF 4, JEE 8: GF 5)
- Autres implémentations: JBOSS, WebSphere, JOnAS



Application JEE : Configuration ⁽¹⁾

Pour définir les stratégies à utiliser: sécurité, persistance

Pour spécifier les redirections par défaut (page d'erreur, ...)

La configuration est définie dans des **fichiers XML** (bean.xml, web.xml, faces-config.xml) ou par des **annotations** dans le code (@RolesAllowed)

Le serveur peut traiter un fichier de configuration spécifique (glassfish-web.xml)

Exemples de fichiers de configurations:

<https://antoniogoncalves.org/2013/06/04/java-ee-7-deployment-descriptors/>

Application JEE : configuration par web.xml (2)

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
    http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
  version="3.1">

  <display-name>CGR</display-name>

  <!-- PAGE PRINCIPALE -->
  <welcome-file-list>
    <welcome-file>index.jsf</welcome-file>
  </welcome-file-list>

  <!-- CONFIGURATION SECURITE -->
  <security-constraint>
    <web-resource-collection>
      <web-resource-name>ACCUEIL</web-resource-name>
      <description>Page d'accueil</description>
      <url-pattern>/index.jsf</url-pattern>
      <url-pattern>/actualites_datas/*</url-pattern>
      <url-pattern>/telechargement/*</url-pattern>
    </web-resource-collection>
    <auth-constraint>
      <role-name>SYS</role-name>
      <role-name>ADMIN</role-name>
      <role-name>GESTION_PRODUCTEUR</role-name>
      <role-name>GESTION_TIERS</role-name>
      <role-name>CONSEIL_ADMINISTRATION</role-name>
    </auth-constraint>
    <user-data-constraint>
      <transport-guarantee>NONE</transport-guarantee>
    </user-data-constraint>
  </security-constraint>
```

```
<security-constraint>
  <web-resource-collection>
    <web-resource-name>LOGIN</web-resource-name>
    <description>Pages login</description>
    <url-pattern>/login.jsf</url-pattern>
    <url-pattern>/resources/*</url-pattern>
    <url-pattern>/themes/*</url-pattern>
  </web-resource-collection>
  <user-data-constraint>
    <transport-guarantee>NONE</transport-guarantee>
  </user-data-constraint>
</security-constraint>

<security-constraint>
  <web-resource-collection>
    <web-resource-name>ACCUEIL</web-resource-name>
    <description>Page d'accueil</description>
    <url-pattern>/index.jsf</url-pattern>
    <url-pattern>/actualites_datas/*</url-pattern>
    <url-pattern>/telechargement/*</url-pattern>
  </web-resource-collection>
  <auth-constraint>
    <role-name>SYS</role-name>
    <role-name>ADMIN</role-name>
    <role-name>GESTION_PRODUCTEUR</role-name>
    <role-name>GESTION_TIERS</role-name>
    <role-name>CONSEIL_ADMINISTRATION</role-name>
  </auth-constraint>
  <user-data-constraint>
    <transport-guarantee>NONE</transport-guarantee>
  </user-data-constraint>
</security-constraint>
```

```
<!-- ROLES utilisés par l'application -->
<security-role>
  <role-name>SYS</role-name>
</security-role>

<security-role>
  <role-name>ADMIN</role-name>
</security-role>
. . .

<!-- CONFIGURATION METHODE D'AUTHENTIFICATION -->
<login-config>
  <auth-method>FORM</auth-method>
  <realm-name>CGRRealm</realm-name>
  <form-login-config>
    <form-login-page>/login.jsf</form-login-page>
    <form-error-page>/login-error.jsf</form-error-
page>
  </form-login-config>
</login-config>

<error-page>
  <error-code>403</error-code>
  <location>/errors/forbidden.jsf</location>
</error-page>

<error-page>
  <error-code>404</error-code>
  <location>/errors/notfound.jsf</location>
</error-page>
```

Application JEE : Déploiement ⁽¹⁾

Application packagée **en archive** fichier .jar avec :

- **.class** : classes et composants
- **Ressources** : pages web, images, documents, sons , etc.
- **Fichier de configuration** : configuration app web, EJB (web.xml, bean.xml, etc.)

Déploiement : téléchargement archive application avec son descripteur

Intérêt du descripteur : Fournir au serveur les directives pour déployer l'application

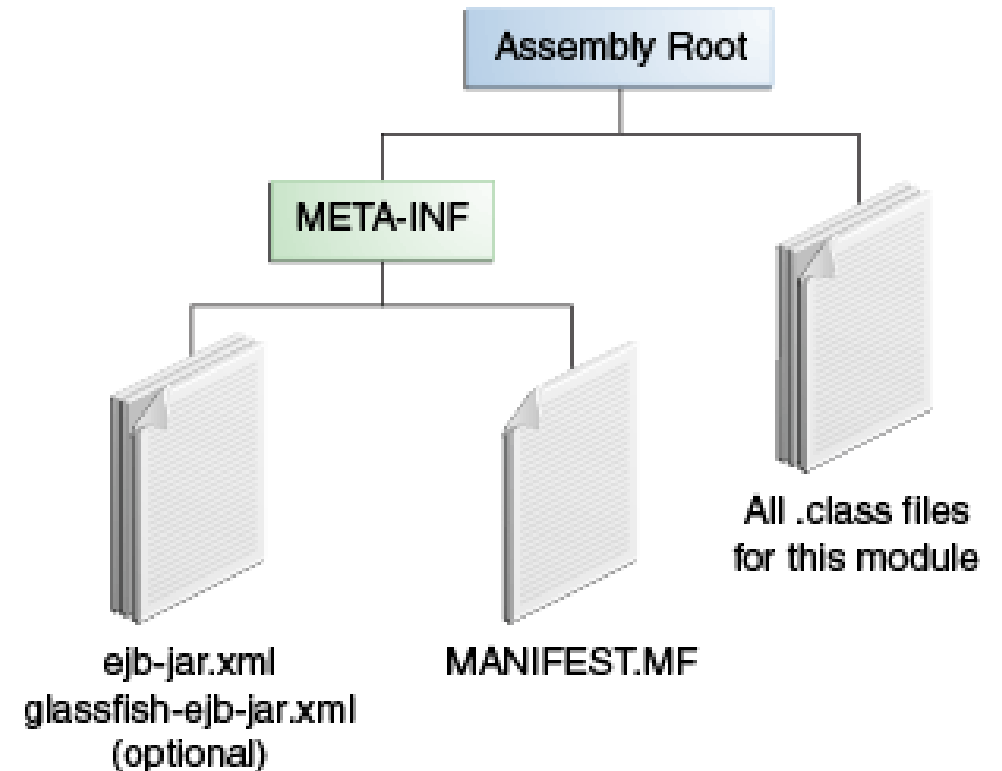
Différent types de package : .jar module d'EJB, .war, .ear

Application JEE : module d'EJB

Un .jar module d'EJB est « Portable » et peut être utilisé par différentes applications

Le package comporte :

- **Des .class** : EJB et simples classes
- **Les descripteurs**: ejb-jar.xml, manifest, etc.

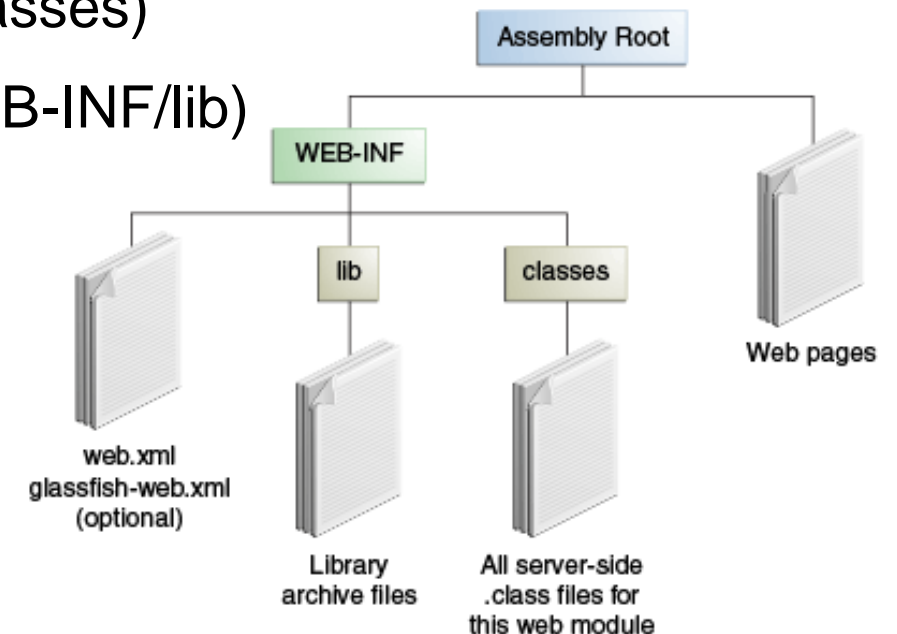


Application JEE : packagée en .war

Une application JEE web packagée

Le package comporte :

- **Des .class** : EJB et simples classes (WEB-INF/classes)
- **Les dépendances** : (lib) sous formes de .Jar (WEB-INF/lib)
- **Les descripteurs**: web.xml, etc.



Application JEE : packagée en .ear

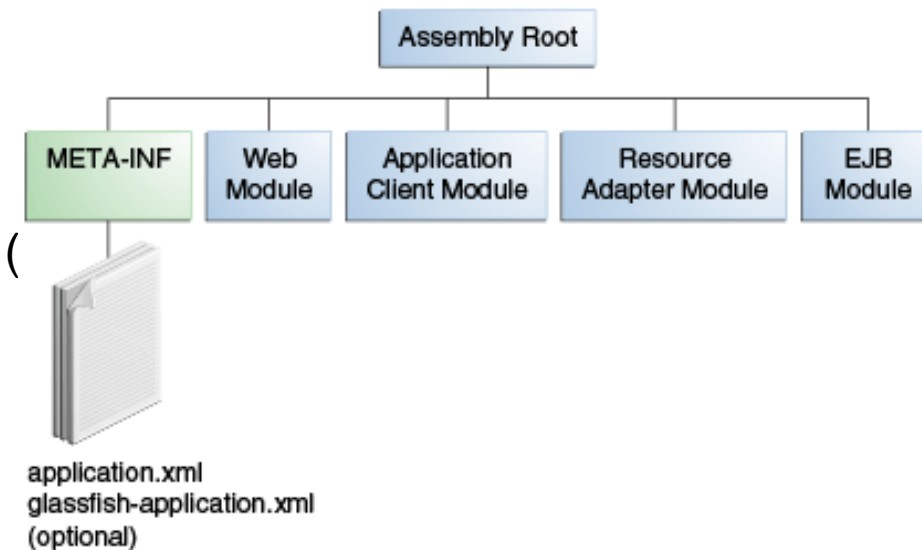
Une application JEE (Enterprise Archive): une application complète

Le package comporte :

- **Les modules:**

- modules d'EJB (.jar)
- Modules web (war)
- Resource Adapter: interfaces, lib, lib natives, connecteurs (

- **Le descripteur de l'application :** application.xml



Conclusion JEE

Spécification pour la création d'applications métiers en Java

Serveur d'application JEE : implémentation de référence Glassfish
(avec ses containers)

Intérêts:

Le développeur se focalise sur le métier, le serveur assure le non fonctionnel

Prévu pour exploiter les objets répartis (IIOP CORBA)

Haute disponibilité, support de la montée en charge