



Institut Informatique
Claude Chappe

Master STS Mention Informatique Parcours ISI

Programmation distribuée
M1 / 178EN003

C4.1 – Introduction à Java EE

Thierry Lemeunier

thierry.lemeunier@univ-lemans.fr

www-lium.univ-lemans.fr/~lemeunie

Plan du cours

- Communication par composant distribué :
 - Principes : des objets aux composants
 - Introduction à Java EE
 - Identité
 - Architecture et conteneurs
 - APIs

Composant distribué – Principes (1/3)

- Défauts de la distribution par objets
 - Rappel : les objets distants sont distribués sur les serveurs
 - *Middleware* basique :
 - Des services non-fonctionnels (sécurité, persistance, transaction, etc.) sont absents ou disponibles pour les développeurs...
 - ... mais doivent être explicitement programmés (généralement « en dur » dans le code)
 - Les applications sont trop monolithiques (mélange du code fonctionnel et non-fonctionnel) et donc peu évolutives
 - Idem pour le déploiement et la configuration des applications
 - Absence de description globale de l'application
 - Difficultés pour la configuration en développement et en exploitation
 - Absence d'outils d'administration
- ➔ Charge encore trop importante pour les programmeurs
- ➔ Une partie du cycle de vie n'est pas couverte par le *middleware*
- ➔ Incidence importante sur la qualité et la portabilité des applications distribuées

Composant distribué – Principes (2/3)

■ Distribution par composants

□ Le middleware

- propose les services non-fonctionnels nécessaires
- et les prend en charge au maximum

□ Le développeur :

- décrit le déploiement, la configuration et l'assemblage
- et programme uniquement la partie métier de l'application

→ Notion de composant et de serveurs d'application

□ Exemples de *middleware* à composants :

- Norme *Corba Component Model* (Corba 3.x) de l'OMG
- Spécification *EJB* intégrée à *Java EE* de Sun
- Le *framework WCF* de la plateforme *.Net 3.0* de Microsoft

Composant distribué – Principes (3/3)

■ Notion de composant

□ Module logiciel paramétrable et interchangeable

- Spécifie des interfaces fournies pour les autres composants ou le client
- Spécifie des interfaces requises provenant d'autres composants
- Spécifie des interfaces de communication avec son conteneur
- Implémentent des services métiers

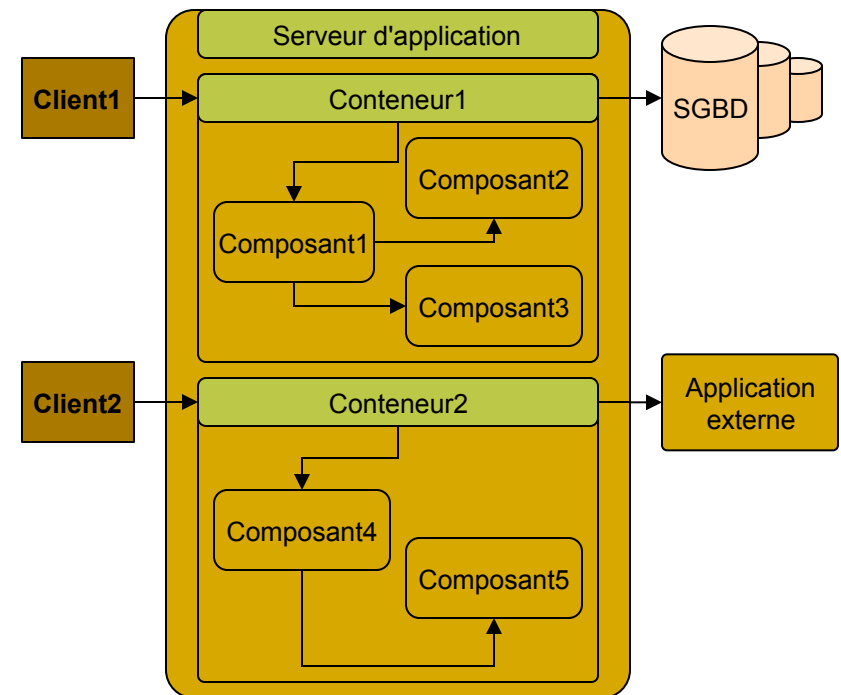
→ Middleware à composants = architecture (plate ou emboîtée) « puzzle » de composants

■ Conteneur

- Les composants sont installés dans des conteneurs d'indirection
- Le conteneur prend en charge les services tels que :
 - environnement (dépendances...)
 - vie des composants (*pooling*...)
 - la persistance
 - les transactions
 - la sécurité (authentification, autorisation)

■ Serveur d'application

- Espace d'exécution des conteneurs et des composants
- Lien avec le système d'exploitation



Plan du cours

- Communication par composant distribué :
 - ✓ Principes : des objets aux composants
 - Introduction à Java EE
 - Introduction à *EJB 3.0*
 - Introduction à *Java Persistence API*

Java EE - Identité

■ Identité :

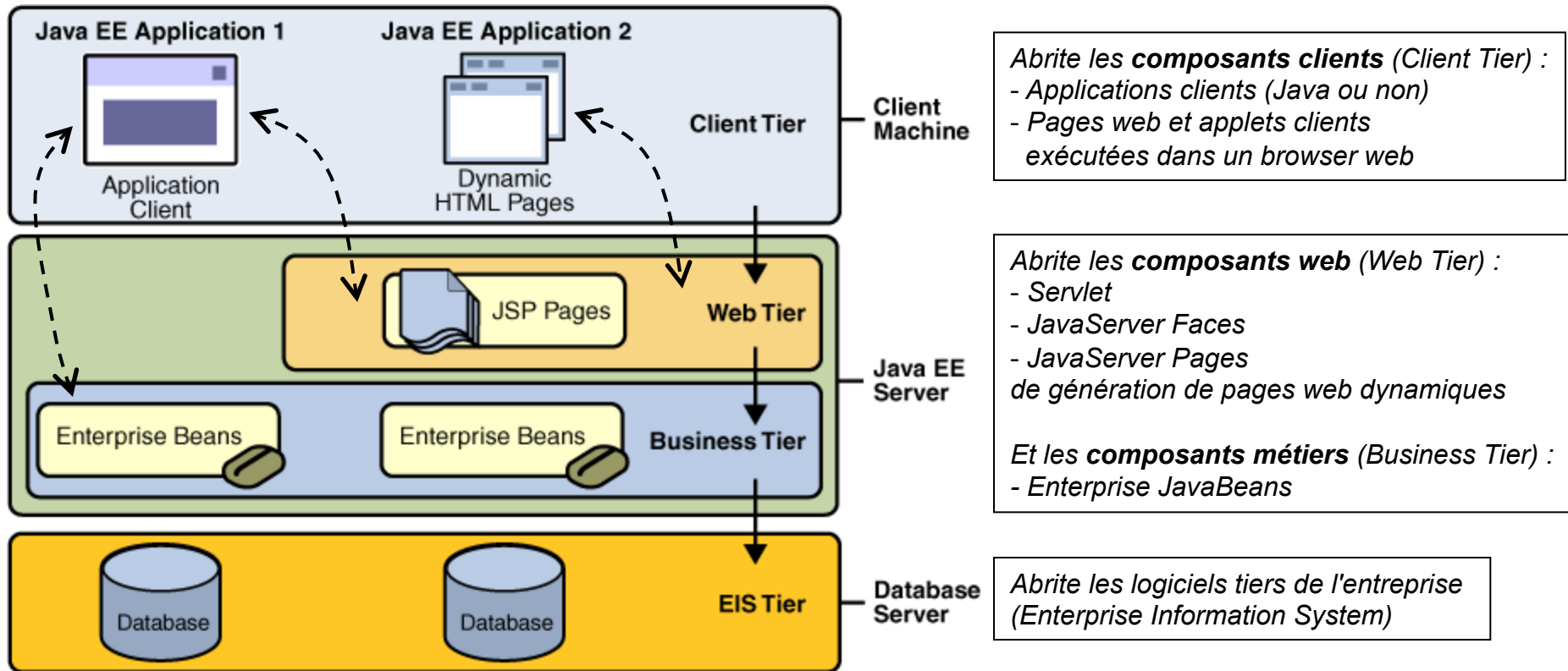
- Java EE (*Java Enterprise Edition*) est une spécification pour les applications distribuées d'entreprises côté serveur
- Elle est constituée d'un ensemble d'API spécifiant les services techniques des applications distribuées
- Elle propose un modèle d'architecture distribuée et un modèle de déploiement à base de composants

■ Le monde Java en très court :

- Java SE (*Java Standard Edition*) : plateforme standard pour les applications généralement graphiques côté client avec les API telles que :
 - AWT, Swing, Java2D pour la partie interface utilisateur
 - JDBC, JNDI, RMI, RMI-IIOP, IDL pour la partie intégration d'applications
 - Collections, réflexivité, multithreading, sérialisation, JAXP...
- Java ME (*Java Micro Edition*) : plateforme pour les applications embarquées sur matériels mobiles ou « enfouis »
- Les implémentations de Java EE sont une extension de Java SE
 - 1999 : J2EE 1.2
 - 2003 : J2EE 1.4 (*Java Specification Requests 151*)
 - 2006 : Java EE 5 (*JSR 244*) ➔ **Simplification substantielle !**
 - 2009 : Java EE 6 (*JSR 316*)
 - 2013 : Java EE 7 (*JSR 342*)

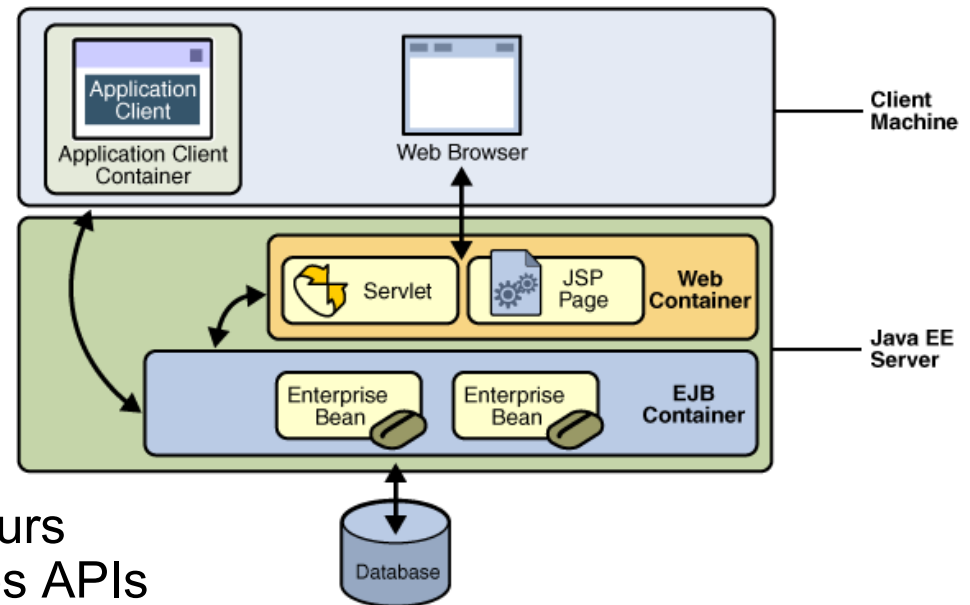
Java EE – Architecture

➔ Application multi-tiers à composants distribués



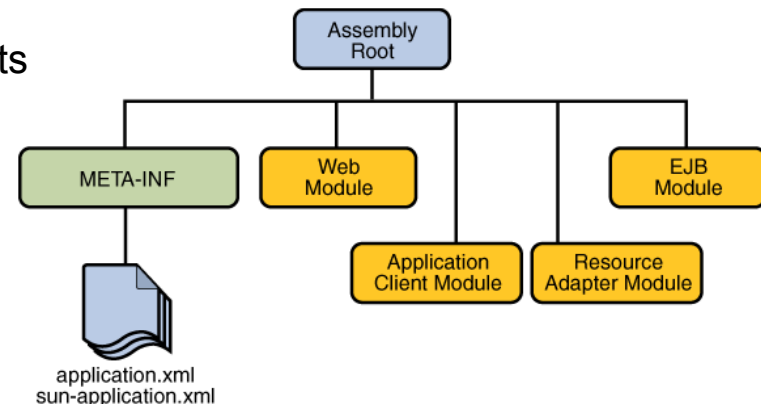
Java EE – Les conteneurs

- Les composants sont assemblés dans un module qui est déployé dans un conteneur
- Un conteneur est configurable et fournit des services tels que :
 - ❑ La sécurité
 - ❑ La gestion transactionnelle d'opérations
 - ❑ La recherche de services via l'API *JNDI*
 - ❑ Les connections distantes
 - ❑ Le cycle de vie des composants
 - ❑ Le recyclage des ressources
 - ❑ La persistance
 - ❑ L'accès aux APIs de Java EE
- Les types de conteneurs :
 - ❑ *Enterprise JavaBeans container*
 - ❑ *Web container*
 - ❑ *Application client container*
 - ❑ *Applet container*
- Les différents types de conteneurs n'ont pas tous accès aux mêmes APIs



Java EE – Assemblage et déploiement

- Une application Java EE est un assemblage :
 - de modules archivés dans une fichier *Enterprise Archive* unique (fichier JAR avec l'extension .ear)
 - et de descripteurs de déploiement (fichiers XML)
 - Il y a 2 types de descripteur :
 - Les descripteurs standards JEE (application.xml)
 - Les descripteurs spécifiques à la plateforme (sun-application.xml pour la plateforme Sun)
- Assemblage
 - L'assemblage consiste à grouper des modules (fichiers JAR) d'une même application
 - Un module comprend un ou plusieurs composants (insérables dans un conteneur de même type) et éventuellement un descripteur
 - Extension des modules :
 - Module Web : .war
 - Module EJB : .jar
 - Module client : .jar
 - Module ressource adaptateur : .rar
- Déploiement
 - Le déploiement consiste à configurer les composants par rapport à l'environnement d'exploitation (spécifier la localisation des ressources, les utilisateurs, les autorisations, etc.)
 - Java EE contrôle l'exécution selon les informations lues dans les descripteurs de l'application, des modules et des composants.



Java EE – Développement d'une application

- L'approche par composants permet de distinguer différents métiers pour le cycle de vie d'une application Java EE :
 - Le fournisseur du serveur d'application et des containers
 - Il implémente une plateforme qui suit la spécification Java EE
 - Exemples : *WebLogic, Websphere, Geronimo, JBoss, Tomcat, GlassFish...*
 - Les vendeurs d'outils
 - Ils développent des outils pour développer, maintenir et déployer les composants
 - Principaux IDEs gratuits : *Eclipse, NetBeans, JDeveloper*
 - Le fournisseur de composants
 - Trouve les composants utiles en les achetant ou en les construisant
 - Il fournit les modules de l'application (comprenant leurs descripteurs)
 - L'assembleur d'application
 - C'est l'architecte de l'application
 - Il décide de la combinaison des modules (le descripteur général)
 - Le déployeur
 - Il déploie l'application sur le serveur d'application en l'ajustant à l'environnement d'exploitation (sécurité, connexion aux ressources, etc.)
 - L'administrateur
 - Contrôle le fonctionnement de l'application en exploitation (outils de *monitoring*)

Java EE – Tour d'horizon des APIs (1/3)

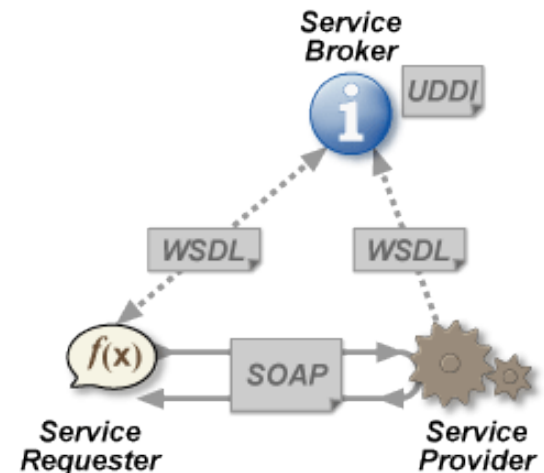
- Servlet (la version 1.0 date de Juin 1997) :
 - Composant côté serveur d'interaction HTTP avec le client
 - Une classe qui étend la classe `javax.servlet.http.HttpServlet`
- JSP (*JavaServer Pages*) :
 - API de construction dynamique côté serveur de contenu Web
 - Documents XMLs mixant :
 - des éléments statiques (balises HTML, WML, XML, etc.)
 - et du code JSP d'accès aux Java beans et de génération à la volée de contenu
 - JSTL (*JavaServer Pages Standard Tag Library*) :
 - Spécification des tags pour écrire des JSP portables (cf. <http://jstl.java.net/>)
- JSF (*JavaServer Faces*) :
 - Framework côté serveur de création de contenu web à la volée
 - Création de l'interface graphique qui sera générée par différents *renderers* (HTML, WML, XML).
 - Création de classes Java Bean contrôleurs utiles aux traitements côté serveur
 - Navigation entre pages du site web
 - Documents XML plus complexe que JSP car c'est un modèle MVC (avec gestion des événements, composants graphiques réutilisables, Ajax...)
- Remarques :
 - Les pages JSP et JSF ont le même type de cycle de vie
 - Lors d'une requête du client, elles sont transformées à la volée par le serveur en classe servlet qui la compile puis l'exécute !

Java EE – Tour d'horizon des APIs (2/3)

- *EJB (Enterprise JavaBeans)* : composant métier côté serveur
- *JMS (Java Message Service)*
- *JavaMail API* : API d'envoi d'emails par les applications
- *JAXP (Java API for XML Processing)* : API de Java SE pour la lecture et l'enregistrement de documents XML (normes DOM ou SAX)
- *JAX-WS (Java API for XML Web Services)* : API des services web
- *JAXB (Java Architecture for XML Binding)* : API de transformation de documents XML en classes Java et vice-versa
- *SAAJ (SOAP with Attachments API for Java)* : API de prise en charge des messages du protocole SOAP
- *JAXR (Java API for XML Registries)* :
 - API d'accès aux services de nom par le web
 - Standard web de service de nom : ebXML et UDDI

Java EE – Zoom sur les services Web

- Définition W3C du service Web : système logiciel supportant une interaction entre machines différentes au dessus d'un réseau
- Service Web et Java EE
 - Java EE fournit les APIs XML pour les services web qui se chargent de la transformation des données en flux XML
 - Échange du flux XML :
 - Le flux XML est échangé entre le client et le *web tier* avec le protocole SOAP (*Simple Object Access Protocol*) au dessus de *HTTP*.
 - Description des services web :
 - Java EE fournit un outil pour générer la description d'un service web avec le langage *WSDL* (*Web Services Description Language*)
 - La description inclut le nom du service, la localisation du service et le protocole de communication
 - La description WSDL peut être publiée et trouvée via le protocole *UDDI* (*Universal Description, Discovery and Integration*)
- Remarque : SOAP, WSDL et UDDI sont des normes du W3C



Java EE – Tour d'horizon des APIs (3/3)

- *JCA (J2EE Connector Architecture) :*
 - API pour connecter une application Java EE avec d'autres applications d'entreprise de technologies différentes
 - L'intégration s'effectue en développant un *resource adapter*
- *JDBC (Java Database Connectivity) :*
 - API de Java SE de connexion à une base de données
 - Une partie est utilisée par l'application (standard SQL) et une autre par la plateforme pour attacher un driver JDBC spécifique
- *JPA (Java Persistence API) :*
 - API de persistance basée sur un *mapping* entre le modèle objet et le modèle relationnel
 - Utilise un langage de requête *JPQL* (dérivé de SQL)
- *JTA (Java Transaction API) :*
 - API de gestions des opérations transactionnelles
- *JNDI (Java Naming and Directory Interface) :*
 - API d'accès à des services de nommage (LDAP, RMIRegistry, ...)
- *JAAS (Java Authentication and Authorization Service)*
 - API de Java SE d'authentification et d'autorisation
 - Remarque : la sécurité dans JEE n'est pas abordée dans ce cours !

Java EE – Zoom sur JNDI (1/2)

■ JNDI : *Java Naming and Directory Interface*

- API d'accès à des services de nommage et d'annuaire pour les appli. Java
- Permet de localiser un objet ou une ressource distribuée
- Permet l'accès à des services extérieurs (LDAP, DNS, NIS, RMIRegistry, etc)
- JNDI doit connaître les paramètres d'accès au service utilisé :
 - Soit à partir d'un fichier de propriétés (jndi.properties)
 - Soit "en dur" dans le code source
 - Soit par injection de dépendance dans le cas d'une exécution dans un conteneur d'application client (ACC) avec glassfish/bin/appclient

```
public class CategoryCrudFrame extends JFrame {
```

```
    public void findActionPerformed(EventObject evt) {
```

```
        Context initialContext = new InitialContext();
```

```
        CatalogRemote catalogRemote = (CatalogRemote) initialContext.lookup("ejb/stateless/Catalog");
```

```
        Category category = catalogRemote.findCategory(identifiant);
```

```
        model.setIdentifiant(category.getId());
```

```
        model.setName(category.getName());
```

```
        model.setDescription(category.getDescription());
```

```
    }
```

```
}
```

Correspond à la racine de l'arbre de l'annuaire

Recherche d'un objet nommé

Java EE – Zoom sur JNDI (2/2)

// Exemple de codage en dur avec les paramètres d'accès au service de nommage de GlassFish

```
public class CategoryCrudFrame extends JFrame {

    public void findActionPerformed(EventObject evt) {

        Properties props = new Properties();
        props.setProperty("java.naming.factory.initial", "com.sun.enterprise.naming.SerialInitContextFactory");
        props.setProperty("java.naming.factory.url.pkgs", "com.sun.enterprise.naming");
        props.setProperty("java.naming.factory.state", "com.sun.corba.ee.impl.presentation.rmi.JNDIStateFactoryImpl");
        props.setProperty("java.naming.provider.url", "localhost");

        Context initialContext = new InitialContext(props);
        CatalogRemote catalogRemote = (CatalogRemote) initialContext.lookup("ejb/stateless/Catalog");
        ...
    }
}
```

// Exemple avec exécution dans l'ACC glassfish/bin/appclient

```
public class CategoryCrudFrame extends JFrame {

    public void findActionPerformed(EventObject evt) {
        @EJB(mappedName="ejb/stateless/Catalog") CatalogRemote catalogRemote;
        ...
    }
}
```