Programme du module: Environnement JEE

1. Chapitre 1: Introduction JEE
2. Chapitre 2: Les servlets
3. Chapitre 3: Les JSP et Java Bean
4. Chapitre 4 : JSTL et JDBC
5. Chapitre 5: Les EJB
6. Chapitre 6 : Développement d'application JEE avec Spring;

(Spring IoC, Spring AOP, Spring MVC, Spring Boot, Spring DATA/JPA)

JEE : introduction

Mohammed OUANAN

m.ouanan@umi.ac.ma

Plan

Qu’est ce que Java EE?

Le besoin de JEE?

- Dynamiser le contenu web
  - Apparition de génération dynamique de code (ASP,-1996- 2002, PHP-1997 , JEE - 1999)
  - Apparition des applets (exécution code java sur navigateur)
- Echanger des informations entre applications
  - RMI, Corba, EJB
- Création de véritables d'architectures logicielles
  - Architectures 3tiers, Modèle-Vue-Contrôleur
- Harmonisation des infrastructures

Jakarta EE

JEE

- 1999 : inclus dans **Java 2** sous le nom **J2EE** pour **J**ava 2 Platform **E**nterprise **E**dition
- 2006 : renommé depuis **Java 5** en **JEE** pour **J**ava Platform **E**nterprise **E**dition
- 2017 : **Oracle** confie le projet à **Eclipse Foundation** qui décide de le renommer **Jakarta EE**

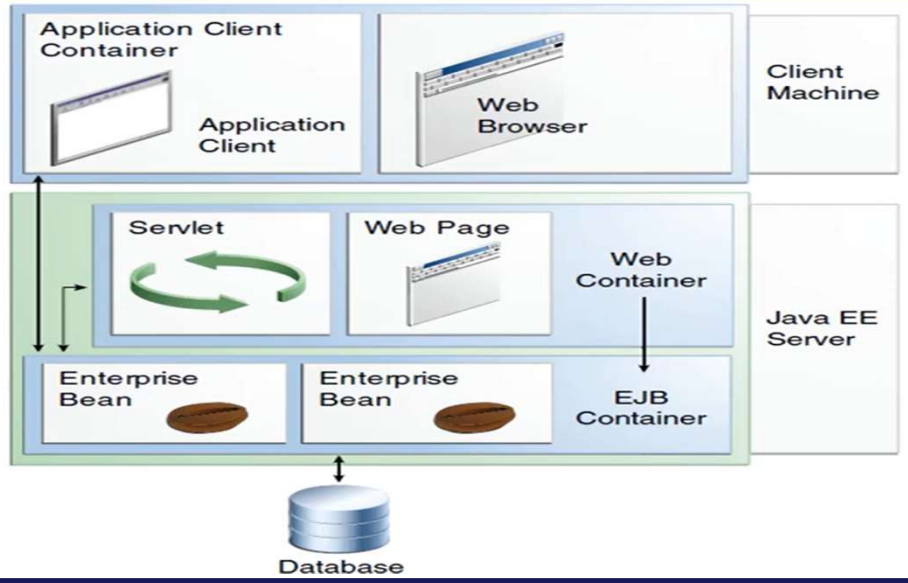
Jakarta EE

JEE

- Plateforme facilitant le développement d'applications d'entreprise distribuées
- Développée par **Sun** puis **Oracle**
- **JEE** = **JSE** + plusieurs autres **API**
  - **JSE** : **J**ava **S**tandard **E**dition (anciennement J2SE)
  - **API** : Application Programming Interface

Jakarta EE

Positionenment des serveurs JEE



Jakarta EE côté client

- Un client Java EE peut être :
  - une application console écrite en Java
  - une application dotée d'une interface graphique Swing.

  Ce type de client est appelé client lourd, en raison de la quantité importante de code qu'il met en œuvre.
- Un client Java EE peut également être conçu pour être utilisé à partir du Web.
  - Ce type de client fonctionne à l'intérieur d'un navigateur Web.
  - La plus grande partie du travail est reportée sur le serveur et le client ne comporte que très peu de code. Pour cette raison, on parle de client léger.

  Un client léger peut être une simple interface HTML, une page contenant des scripts JavaScript, ou encore une applet Java si une interface un peu plus riche est nécessaire.

Jakarta EE côté Serveur

Les API de Java EE peuvent se répartir en deux grandes catégories :

- Les composants déployés sur le serveur
  - Les composants web qui sont réalisés à l'aide de servlets, de JavaServer Pages (JSP) ou de Java server face (JSF).
  - Les composants métier sont des Entreprise JavaBeans (EJB). Il s'agit de composants spécifiques chargés des traitements des données et de l'interfaçage avec les bases de données.
- Les services
  - Les services d'infrastructures : JDBC, JNDI, JTA, JCA, JMX
  - Les services de communication : RMI-IIOP, JavaMail, JAAS

Les principaux composants Jakarta EE

- **Servlet**: Utilise les mécanismes de requetes-reponses http afin d'effectuer des tâches ou de générer des pages html
- **EJB** : définit la façon dont les composants doivent être écrits et le contrat qu'ils doivent respecter avec le serveur d'application
- **RMI** : communication inter procédés
- **JNDI** : C'une API d'accès aux services de nommage et aux annuaires d'entreprises tels que DNS, NIS, LDAP, etc.
- **JDBC** : connexion vers les bases de données
- **JTA** : service de transaction
- **JMS** : service de messagerie
- **JSP** : Java Server Page
- **Java IDL** : intégration aux autres langages (ex: CORBA) JavaMail
- **Connectors** : intégration à des middlewares existants XML

Les conteneurs JEE

Un conteneur (container) est l'interface entre le composant et les services de bas niveaux nécessaires

- Pour pouvoir être exécuté, un composant / application web doit être :

(1) assemblé dans un module Java EE

(2) déployé dans son conteneur.

Les conteneurs JEE

Les conteneurs sont les éléments fondamentaux de l'architecture Java EE.

- Ils fournissent un ensemble de services permettant aux développeurs d'applications de se concentrer sur la logique métier du problème à résoudre sans se préoccuper de toute l'infrastructure interne.
- Les conteneurs assurent la gestion du cycle de vie des composants qui s'exécutent en eux. Les conteneurs fournissent des services qui peuvent être utilisés par les applications lors de leur exécution.

  La plate-forme Java EE disposent de conteneurs pour les composants Web et les composants métiers.
- Ces conteneurs possèdent des interfaces leur permettant de communiquer avec les composants qu'ils hébergent.
- Il existe plusieurs conteneurs définit par Java EE:

  • conteneur web : pour exécuter les servlets, les JSP et JSF

  • conteneur d'EJB : pour exécuter les EJB

  • conteneur client : pour exécuter des applications autonomes sur les postes qui utilisent des composants Java EE

Comparaison des fichiers JAR, WAR et EAR

sont des formats d'archives utilisés pour regrouper et distribuer des applications. Chacun de ces formats a un objectif spécifique, lié à l'environnement dans lequel l'application sera déployée (standalone ou serveur d'applications)

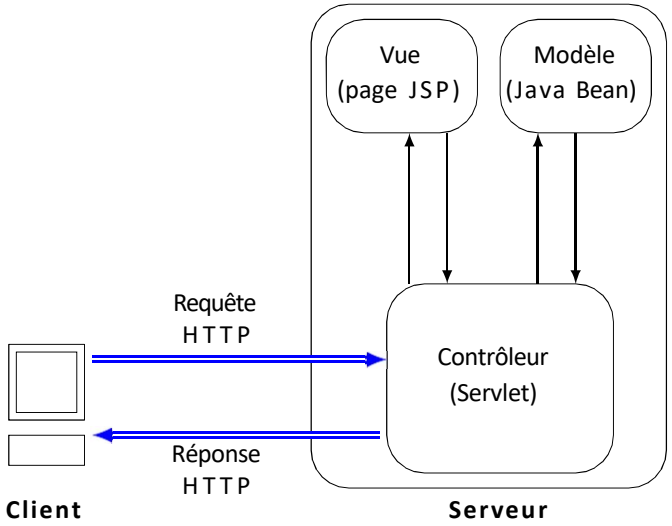
Type	Contenu	Utilisation principale	Environnement de déploiement
JAR	Classes Java, bibliothèques, métadonnées	Applications autonomes ou bibliothèques Java	JVM standalone
WAR	Application web, fichiers JSP/HTML, classes	Applications web Java (Servlets, JSP, Spring)	Serveur d'application web (Tomcat, Jetty)
EAR	Modules JAR et WAR, fichiers de configuration	Applications d'entreprise Java EE (EJB, Web) complexes	Serveur d'application Java EE (WildFly, GlassFish)

Déploiement d'une application

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
- 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

Jakarta EE


Vue
(page JSP)
Modèle
(Java Bean)
Requête
HTTP
Contrôleur
(Servlet)
Réponse
HTTP
Client
Serveur

Jakarta EE

Déroulement

- L'échange entre le client et le serveur s'effectue via le modèle **HTTP** (requête - réponse)
- Lorsque l'utilisateur saisit l'adresse d'une page de notre site, cette dernière sera envoyée sous forme de requête **HTTP** au contrôleur
- Le contrôleur demande au modèle de lui fournir certaines données
- Ensuite il renvoie ces données à la vue pour qu'elle construise la page **HTML**
- Enfin le client reçoit la réponse sous forme de page **JSP**

Jakarta EE

De quoi on a besoin

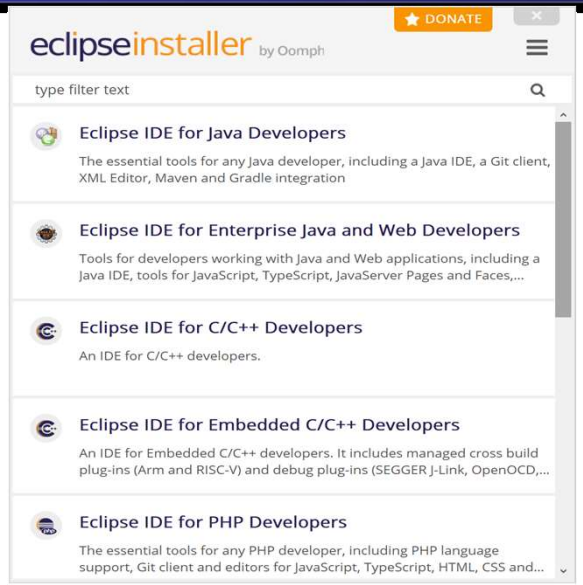
- **JDK** : **J**ava **D**evelopment **K**it
- **IDE** : **I**ntegrated **D**evelopment **E**nvironment (Environnement de Développement Intégré)
- Un serveur **HTTP**

Jakarta EE

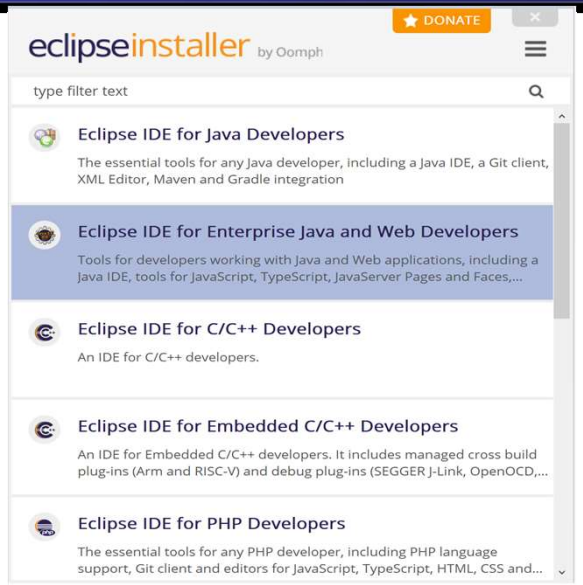
Environnement de Dévloppement Intégré

- **Eclipse**
- Netbeans
- JDeveloper
- IntelliJ IDEA
- JBuilder
- JCreator...
- ...

Jakarta EE


DONATE
eclipseinstaller by Oomph
type filter text
Eclipse IDE for Java Developers
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration
Eclipse IDE for Enterprise Java and Web Developers
Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces,...
Eclipse IDE for C/C++ Developers
An IDE for C/C++ developers.
Eclipse IDE for Embedded C/C++ Developers
An IDE for Embedded C/C++ developers. It includes managed cross build plug-ins (Arm and RISC-V) and debug plug-ins (SEGGER J-Link, OpenOCD,...
Eclipse IDE for PHP Developers
The essential tools for any PHP developer, including PHP language support, Git client and editors for JavaScript, TypeScript, HTML, CSS and...

Jakarta EE


DONATE
eclipseinstaller by Oomph
type filter text
Eclipse IDE for Java Developers
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration
Eclipse IDE for Enterprise Java and Web Developers
Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces,...
Eclipse IDE for C/C++ Developers
An IDE for C/C++ developers.
Eclipse IDE for Embedded C/C++ Developers
An IDE for Embedded C/C++ developers. It includes managed cross build plug-ins (Arm and RISC-V) and debug plug-ins (SEGGER J-Link, OpenOCD,...
Eclipse IDE for PHP Developers
The essential tools for any PHP developer, including PHP language support, Git client and editors for JavaScript, TypeScript, HTML, CSS and...

Jakarta EE

Modifier l'encodage par défaut

- Aller dans le menu `Window` et cliquer sur `Preferences`
- Dans la zone de recherche, écrire `encoding`
- Pour toutes les sections, mettre `UTF-8` à la place de l'encodage par défaut (généralement Cp1252 ou ISO-8859-1)
- Ensuite appliquer et fermer

Java

Quelques raccourcis **Eclipse**

- `Ctrl` + `Shift` + `:` : commenter/décommenter le code
- `Ctrl` + `Shift` + `f` : formater le code
- `Ctrl` + `Alt` + ↓ ou `Ctrl` + `Alt` + ψ : dupliquer la ligne sélectionnée
- `Ctrl` + `Shift` + `o` : gérer les importer
- `Ctrl` + `Alt` + `l` : afficher la liste des raccourcis
- `Alt` + `Shift` + `R` : faire une sélection multiple
- `Shift` + `K` : aller à l'occurrence suivante
- `Ctrl` + `Shift` + `K` : aller à l'occurrence précédente

Serveur **HTTP**

- **Apache Tomcat**
- WebLogic Server (Serveur payant d'oracle utilisé par JDeveloper)
- JBoss
- GlassFish (Open Source de Oracle)
- ...

Serveur **HTTP**

- **Apache Tomcat**
- WebLogic Server (Serveur payant d'oracle utilisé par JDeveloper)
- JBoss
- GlassFish (Open Source de Oracle)
- ...

Pourquoi **Apache Tomcat** ?

- gratuit
- multi-plateforme
- léger
- ...

Jakarta EE

Mise en place d'**Apache Tomcat**

- Pour télécharger, aller sur https://tomcat.apache.org/download-10.cgi
- Aller dans la section `Core`
- Cliquer sur `32-bit/64-bit Windows Service Installer`
- Attendre la fin du téléchargement puis lancer l'installation
- Installer **Apache Tomcat** à la racine de votre disque dur (`C:` pour Windows) et vérifier que le nom du dossier destination ne contient pas d'espace
- Décocher la case de la dernière fenêtre d'installation proposant de démarrer **Apache Tomcat**

Jakarta EE

Les étapes

- Aller dans le menu `File > New` et cliquer sur `Other` (ou CTRL + N)
- Ensuite choisir `Dynamic Web Project` situé dans le répertoire `Web`
- Saisir `cours-jee` dans `Project name`
- Ensuite cliquer sur le bouton `New Runtime`, choisir la dernière version d'Apache Tomcat et cocher la case juste en-dessous `create a new local server`
- Cliquer sur `Next` et préciser le répertoire d'installation de **Apache Tomcat**
- Cliquer sur `Finish` ensuite deux fois sur `Next`, puis cocher la case `Generate web.xml deployment descriptor`

Jakarta EE

Les étapes

- Faire un clic droit sur le projet et choisir `Run As` ensuite `Run on Server`
- Sélectionner le serveur **Apache Tomcat** et cocher la case `Always use this server when running this project`
- Cliquer sur `Next` et vérifier que notre projet figure dans la liste `Configured`
- Et enfin valider en cliquant sur `Finish`

Jakarta EE

Structure d'une application **JEE**

- Le nom du projet définit la racine de l'arborescence
- La racine contient principalement trois répertoires
  - `Java Resources` est réservé aux classes **Java**
  - `src` contient un dossier `main` contenant deux répertoires `java` et `webapp`
    - `java` pointe vers `Java Resources`
    - `webapp` contient tous les fichiers web
  - `build` contient les fichiers compilés (d'extension `.class`)

Jakarta EE

Contenu de `webapp`

Les fichiers définis directement dans `webapp` sont accessibles aux visiteurs (sans avoir besoin de passer par une **Servlet**). C'est ici qu'on place les fichiers **CSS**, **JavaScript**...

- `WEB-INF` (inaccessible aux visiteurs) a un dossier `lib` (pour les librairies externes d'extension `.jar`) et un fichier `web.xml` (premier fichier consulte´ par le serveur **HTTP**).
- `META-INF` contient les méta-données du projet.

Jakarta EE

Création d'une page **HTML**

- Clic droit sur le projet, aller dans le menu `new` et choisir `HTML File`
- Placer le fichier directement dans `webapp` (pas dans `WEB-INF`)
- Nommer le fichier `index.html` et générer la page en cliquant sur Finish
- Mettre `Hello world` entre les balises `<body>`
- Démarrer le serveur
- Aller à l'adresse `http://localhost:8080/cours-jee/` ou `http://localhost:8080/cours-jee/index.html`

Remarque

En cas d'erreur lors du lancement d'**Apache Tomcat**

- Faire double clic sur le serveur
- Aller dans la section **Ports** et vérifier que
  - **Tomcat admin port** : 8005
  - **HTTP/1.1** : 8080
  - **AJP/1.3** : 8009

`WEB-INF` est inaccessible aux visiteurs

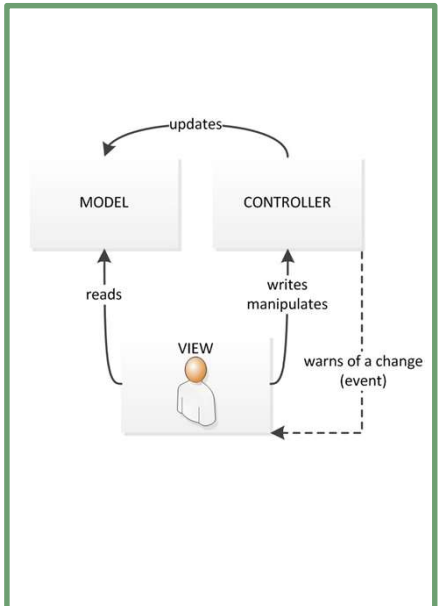
- Déplacer `index.html` dans `WEB-INF`
- Redémarrer le serveur
- Vérifier que `http://localhost:8080/cours-jee/` et `http://localhost:8080/cours-jee/index.html` ne sont plus accessibles.

Architecture MVC Modèle-Vue-Contrôleur

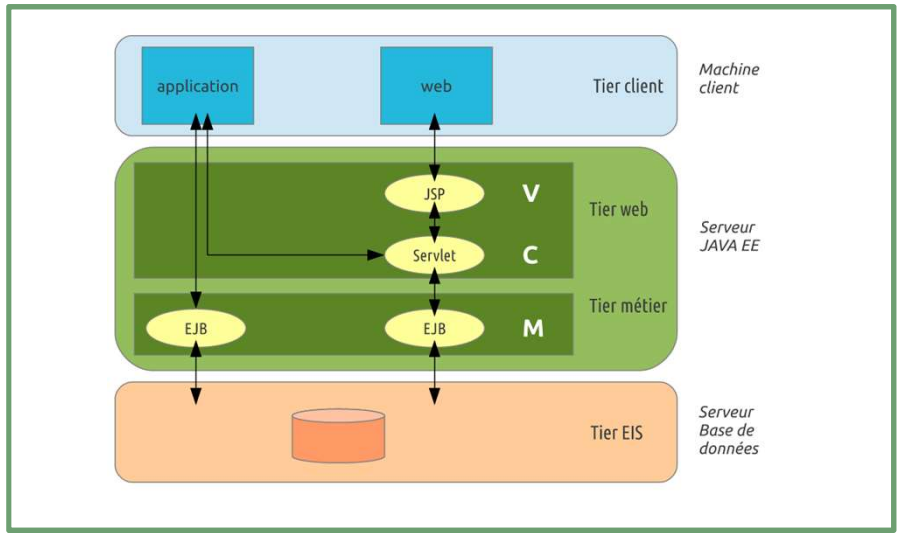
- **Définition**

Séparation des problématiques liées aux applications interactives

- Modèle
  - Stockage des données
- Vue
  - Affichage des données
  - Gestion d'interactions
- Contrôleur
  - Traitement des demandes de modification des données
  - Modification/validation des données
  - Orchestrateur des pages et données à afficher



- **JEE et modèle MVC**



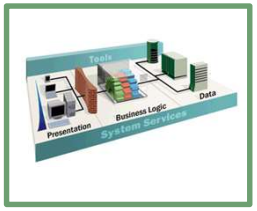
Frameworks JEE

sont des outils ou des bibliothèques conçus pour simplifier et accélérer le développement d'applications d'entreprise en Java. Ils fournissent des fonctionnalités prêtes à l'emploi pour gérer des tâches courantes telles que la gestion des transactions, la persistance des données, la sécurité, et bien plus encore.

Architecture multi-tiers

- **Définition**

Découpage fonctionnel et/ou physique d'un programme



- Objectif:

  Diviser les responsabilités et charges de travail pour un passage à l'échelle, la modularité de programmation, la sécurisation adaptée à la fonction d'un tiers.

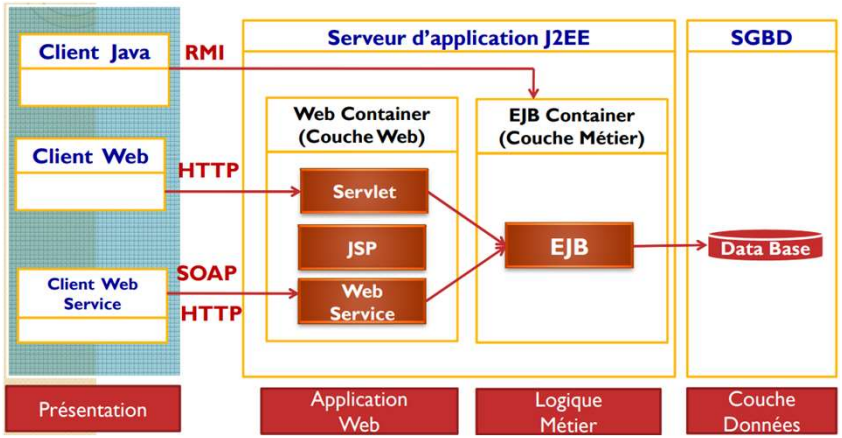
- Exemples:
  - Architecture 1tier

    → Un seul composant applicatif gère la présentation, le fonctionnement, la persistance et l'espace de stockage
  - Architecture 2tiers

    → Client/serveur:
  - Architecture 3tiers/4tiers

    → Applications J2E actuelles

Architecture Multi-Tiers



Couche présentation

- Elle implémente la logique présentation de l'application
- 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 (Remote Method Invocation)
  - 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
    - Androide, iPhone, Tablette etc..
    - Généralement ce genre de clients communique avec les composants Web Services en utilisant le protocole HTTP ou SOAP

Couche d'application

- Appelée également couche web.
- La couche application 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 suivie des sessions
  - Vérifier les autorisations d'accès de chaque session
  - Assure la validation des données envoyées par le client
  - Fait appel au 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.

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 traitement métiers déclenchés par la couche application.
  - 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 frameworks spécialisés dans l'accès aux données (Hibernate, Toplink, etc…)