

---

# Les architectures N-tiers

# SOMMAIRE DU COURS XML ET LES ARCHITECTURES N-TIER

---

- Introduction aux architectures N-tier
- Serveurs d'applications
- Déploiement d'applications J2EE
- *Tiers applicatif* : servlets
- *Tiers présentation* : JSP
- *Tiers métier* : accès aux bases de données
- Les APIs pour lire des documents XML
- Les APIs pour transformer des documents XML
- Modèles de conception et frameworks
- Conclusion : transformation client ou serveur ?

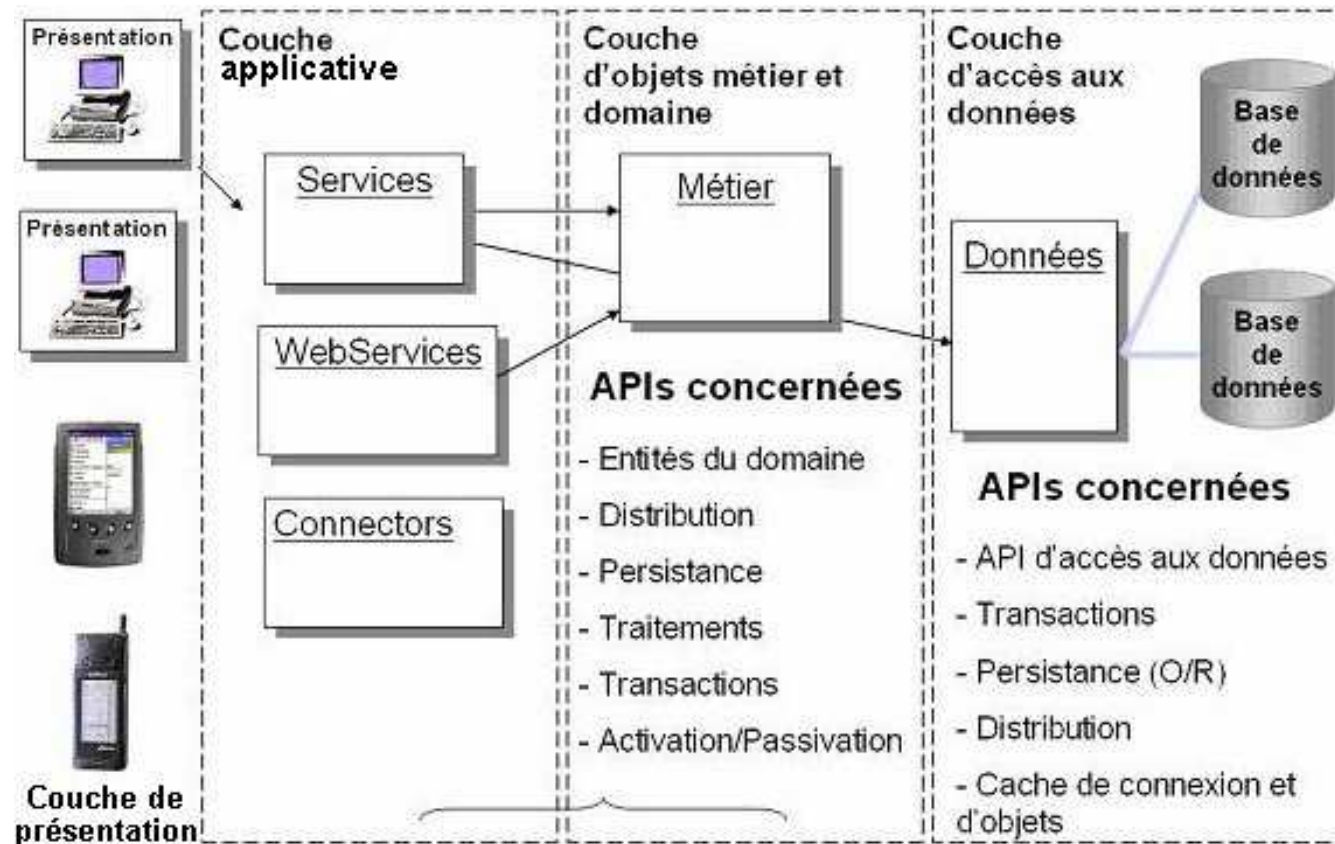
# INTRODUCTION AUX ARCHITECTURES N-TIER 1/9

---

- L'architecture N-tier (anglais *tier* : étage, niveau), ou encore appelée multi-tier, est une architecture client-serveur dans laquelle une application est exécutée par plusieurs composants logiciels distincts.
- Exemple d'architecture 3-tier :
  - Tier de présentation : interfaces utilisateurs sur un PC poste de travail, qui s'adressent à des applications serveur
  - Tier des règles de gestion : applications serveur qui contiennent la logique de gestion et accèdent aux données stockées dans des bases de données
  - Tier de base de données : serveurs de bases de données
- Avantages des architectures N-tier :
  - Le lien entre les niveaux est défini et limité à des interfaces
  - Les interfaces assurent la modularité et l'indépendance technologique et topologique de chaque niveau

# INTRODUCTION AUX ARCHITECTURES N-TIER 2/9


## ■ Les différentes couches d'une architecture 4-tier :



# INTRODUCTION AUX ARCHITECTURES N-TIER 3/9

---

## ■ Les différentes couches d'une architecture 4-tier :

- **La couche de présentation** contient les différents types de clients, léger (ASP, JSP) ou lourd (Applet)
- **La couche applicative** contient les traitements représentant les règles métier (créer un compte de facturation, calculer un amortissement ... ) 
- **La couche d'objets métier** est représentée par les objets du domaine, c'est à dire l'ensemble des entités persistantes de l'application (Facture, Client ... )
- **La couche d'accès aux données** contient les usines d'objets métier, c'est à dire les classes chargées de créer des objets métier de manière totalement transparente, indépendamment de leur mode de stockage (SGBDR, Objet, Fichiers, ...)

# INTRODUCTION AUX ARCHITECTURES N-TIER 4/9

---

## ■ La valeur ajoutée des architectures n-tier :

- Cette **séparation par couches de responsabilités** sert à découpler au maximum une couche de l'autre afin d'éviter l'impact d'évolutions futures de l'application.
- Par exemple : si l'on est amené à devoir changer de base de données relationnelle, seule la couche d'accès aux données sera impactée, la couche de service et la couche de présentation ne seront pas concernées car elles auront été découplées des autres.

# INTRODUCTION AUX ARCHITECTURES N-TIER 5/9

- **Les différentes technologies côté client :**

Type de données	Présentation	Traitement	Informations
HTML	X		X
XML			X
CSS	X		
XSL	X (XSL FO)	X (XSLT)	
JavaScript		X	
VBScript		X	
Applet Java	X	X	X
Active X	X	X	X

- HTML, XML, XSL sont des **langages de marquage**/balisage.
- HTML, CSS, XML, XSL sont **des standards du W3C**
- JavaScript et Java sont **des langages standards**
- VBScript est un **langage propriétaire**
- Active X est une **technologie objet propriétaire**<sup>7</sup>

# INTRODUCTION AUX ARCHITECTURES N-TIER 6/9

---

## ■ Les différentes technologies côté serveur :

### – JSP (Java Server Pages de Sun)

Comme la plupart de ses concurrents, il permet d'intégrer des scripts, ici sous forme de code Java, dans les pages html. Lorsqu'une page JSP est appelée pour la première fois, elle est compilée et transformée en servlet (programme côté serveur). Ce servlet est exécuté et produit un contenu au format html.

### – Java / J2EE (Java 2 Enterprise Edition, Sun)

Promu par la société Sun, l'avantage principal de java est d'être **indépendant du système d'exploitation** (interprété par une machine virtuelle). Java offre de plus la particularité de pouvoir être exécuté côté client (applets) ou côté serveur (servlets). **Il nécessite une bonne connaissance technique et des concepts objet.**



# INTRODUCTION AUX ARCHITECTURES N-TIER 7/9

---

## ■ Les différentes technologies côté serveur :

### – ASP (Active Server Pages de Microsoft)

Cette technologie est basée sur des scripts côté serveur, écrits en Vbscript ou Javascript. Ces scripts sont exécutés par le serveur et leur résultat est produit sous forme de pages html standard. Un des avantages d'ASP est sa **facilité de mise en œuvre**. Largement répandue, cette technologie présente toutefois le désavantage d'être **intimement liée à l'environnement** Windows Server et au serveur IIS.

### – C# / .Net (Microsoft)

Cette technologie **ressemble en de nombreux points à la technologie Java / J2EE**. Elle présente cependant encore de nombreuses **faiblesses en terme de portabilité serveur, multi-plateformes, scalabilité ...**

# INTRODUCTION AUX ARCHITECTURES N-TIER 8/9

---

## ■ Les différentes technologies côté serveur :

### – PHP (Hypertext PreProcessor)

PHP connaît un succès toujours croissant sur le Web et se positionne comme un rival important pour ASP et JSP.

L'environnement Linux est sa plateforme de prédilection. Combiné avec le serveur Web Apache et la base de données MySQL, PHP offre une **solution particulièrement robuste, stable et efficace, offrant en outre l'avantage d'être gratuite**, tous ces logiciels venant du monde des logiciels libres (Open Source).

# INTRODUCTION AUX ARCHITECTURES N-TIER 9/9

---

## ■ Les différents réseaux :

- **Internet** : Réseau de portée mondiale interconnectant des ordinateurs et des réseaux personnels et professionnels.
- **Intranet** : Réseau de portée locale interconnectant des ordinateurs et des réseaux réservés à une entreprise.
- **Extranet** : Réseau interconnectant plusieurs ordinateurs et réseaux de plusieurs entreprises.

# SERVEURS D'APPLICATION

1/4

## ■ Principales fonctionnalités d'un serveur Web :

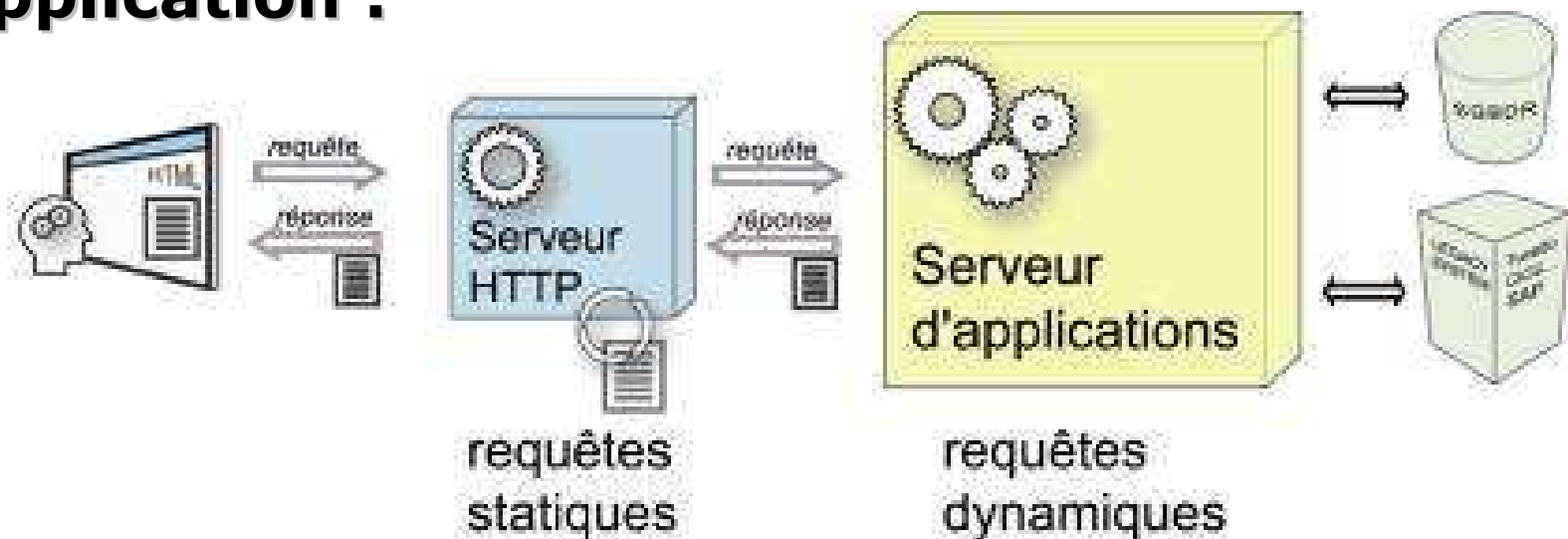


- Réceptionner la requête
- Re-router les requêtes dynamiques
- Rechercher les pages statiques
- Encapsuler les pages dans la réponse
- Émettre la réponse

# SERVEURS D'APPLICATION

2/4

## ■ Principales fonctionnalités d'un serveur d'application :



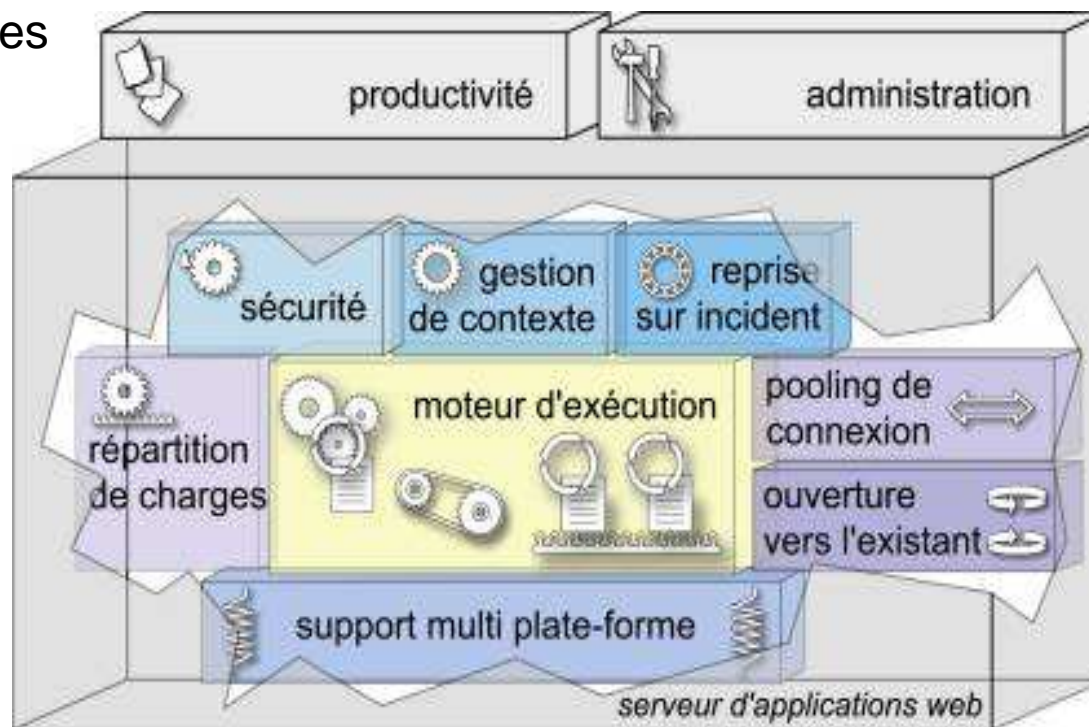
- Réceptionner la requête
- Construire la réponse dynamique
- Renvoyer la réponse au serveur Web

# SERVEURS D'APPLICATION

3/4

## ■ Les fonctionnalités d'un serveur d'application :

- La production de contenu dynamique
- Le support des plates-formes
- L'ouverture vers l'existant
- Le pooling de connexions
- Le respect des standards
- L'administration
- La reprise sur incident
- La répartition de charges
- La sécurité
- La gestion de contexte



# SERVEURS D'APPLICATION

4/4

---

## ■ **L'architecture mise en œuvre dans le cadre des TD :**

- Système d'exploitation Linux et Windows
- Serveur d'application Tomcat
- Programmation J2EE

# RAPPEL SUR J2EE 1/6

---

## ■ J2EE s'appuie sur des concepts objet :

- **Classe** : type d'objet caractérisé par sa structure de données (attributs) et son comportement (méthodes).
- **Objet** : instance de classe.
- **Héritage** : Mécanisme permettant à une classe d'objets de bénéficier de la structure de données et du comportement d'une classe "mère", tout en lui permettant de les affiner et ce, afin de prendre en compte les spécificités de la classe "fille", sans avoir cependant à redéfinir ce que les deux classes ont de commun.
- **Abstraction** : Mécanisme permettant la dissociation entre la déclaration d'une classe et son implémentation.
- **Polymorphisme** : Mécanisme permettant d'associer à un comportement, une implémentation différente en fonction de l'objet auquel on se réfère.
- **Encapsulation** : Mécanisme permettant de dissimuler les détails du fonctionnement interne d'une classe aux autres classes.



# RAPPEL SUR J2EE 2/6

---

## ■ J2EE s'appuie sur le langage Java :

- Java est un **langage orienté objet** dont la syntaxe est dérivé du C et dont la conception résulte de l'expérience de divers langages (Smalltalk, Ada, C++ ...)
- Java est un langage **semi-compilé**. Le code obtenu après compilation s'appelle du byte-code et ce code est interprétable par une JVM (Java Virtual Machine).
- Cependant, Java peut également être complètement **compilé** (transformé en langage machine) si nécessaire.
- Java est **portable** sur toutes les plate-formes puisqu'il existe des machines virtuelles pour chacune. (Les navigateurs intègrent des machines virtuelles java).
- Java est un langage conçu, à l'origine, pour être utilisé via un **réseau**.

# RAPPEL SUR J2EE 3/6

---

## ■ J2EE s'appuie sur le langage Java (suite) :

- Java est un langage intégrant différent mécanisme de **sécurité** (il permet de définir des stratégies de sécurité permettant par exemple d'interdire l'accès aux ressources locales de la machine).
- Java intègre un **ramasse-miette automatique** facilitant de ce fait le codage et diminuant les risques de mauvaise gestion de la mémoire.
- Java est un langage **multi-thread** (il permet la gestion en parallèle de plusieurs processus).
- **Les produits de développement :**
  - ☐ IBM RAD
  - ☐ Borland Jbuilder
  - ☐ Oracle JDeveloper
  - ☐ NetBeans (Sun)
  - ☐ Eclipse
  - ☐ BEA Workshop

# RAPPEL SUR J2EE 4/6

---

## ■ J2EE est une architecture de composants :

### ● Objectif des composants :

- ☐ avoir des briques de bases réutilisables.

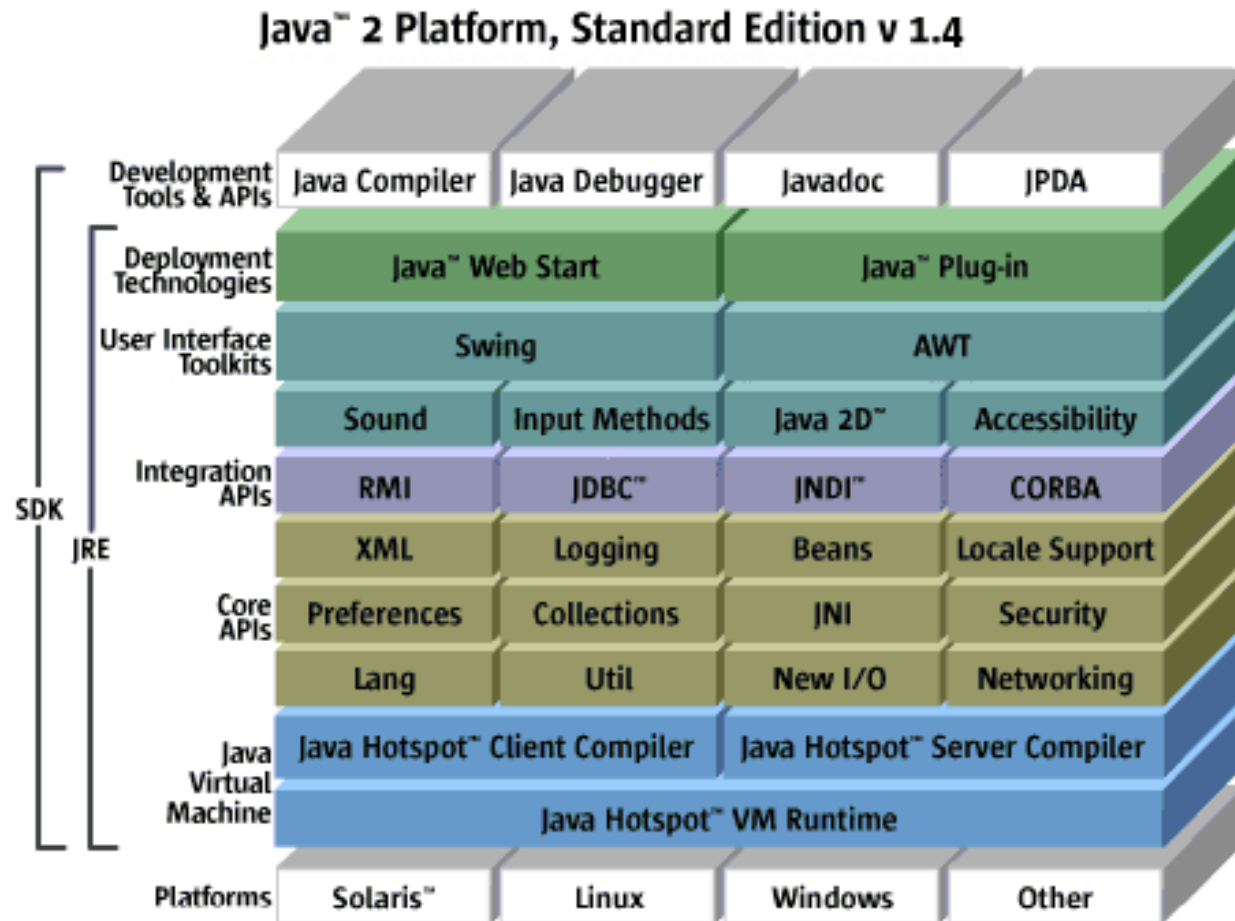
### ● Définition d'un composant :

- ☐ **module logiciel,**
- ☐ **exporte différents attributs, propriétés et méthodes,**
- ☐ **est prévu pour être configuré,**
- ☐ **est prévu pour être installé,**
- ☐ **fournit un mécanisme lui permettant de s'auto-décrire.**

● Composant = objet + configurateur + installateur.

# RAPPEL SUR J2EE 5/6

## ■ Les composants de la plate-forme J2EE :



Version  
1.5

# RAPPEL SUR J2EE 6/6

---

## ■ En résumé :

- **Java 2 Enterprise Edition** est la définition d'un ensemble de standards, relatifs à des services techniques développés en Java dont l'objectif est de fournir une architecture logicielle permettant le déploiement d'applications transactionnelles critiques.
- C'est aujourd'hui un standard du marché car il offre :
  - *une simplification de l'architecture, du développement et de la maintenance*
  - *un support du transactionnel et de la scalabilité*
  - *une intégration homogène avec les SI existants*
  - *une indépendance sur le choix des serveurs, des outils et des composants – nouveau JEE*

# ASSEMBLAGE ET DEPLOIEMENT

## D'APPLICATIONS J2EE 1/6

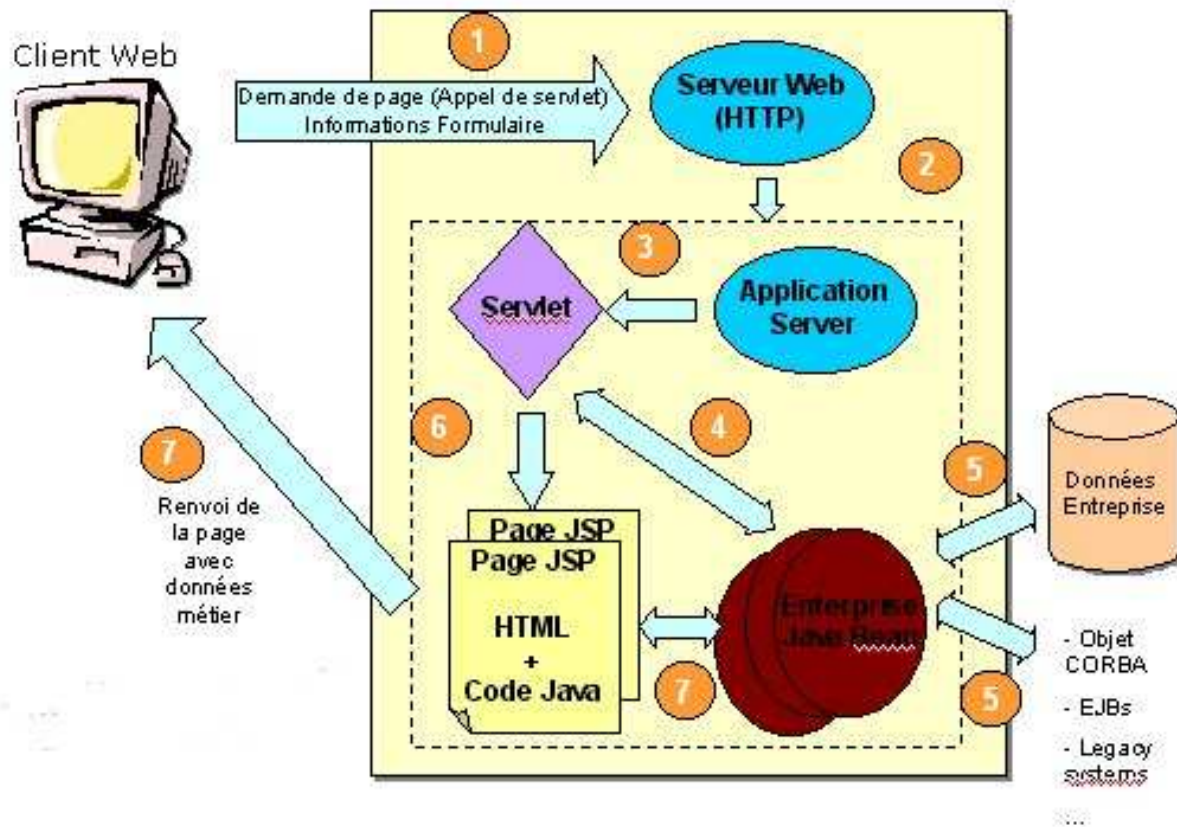
---

### ■ Le développement d'applications Web repose sur trois composants J2EE principaux :

- **Les servlets** : ce sont des programmes Java exécutés sur un serveur (par sa JVM). Ils permettent d'étendre le comportement du serveur dynamiquement.
- **Les JSP** : ce sont des pages HTML incluant du code JAVA (stocké à l'intérieur de balises).
- **Les EJB** : ce sont des entités de traitement s'exécutant dans un environnement adapté (conteneur) et possédant des mécanismes de configuration et d'installation.

# ASSEMBLAGE ET DEPLOIEMENT D'APPLICATIONS J2EE 2/6

## ■ Mécanisme d'une application Web J2EE :



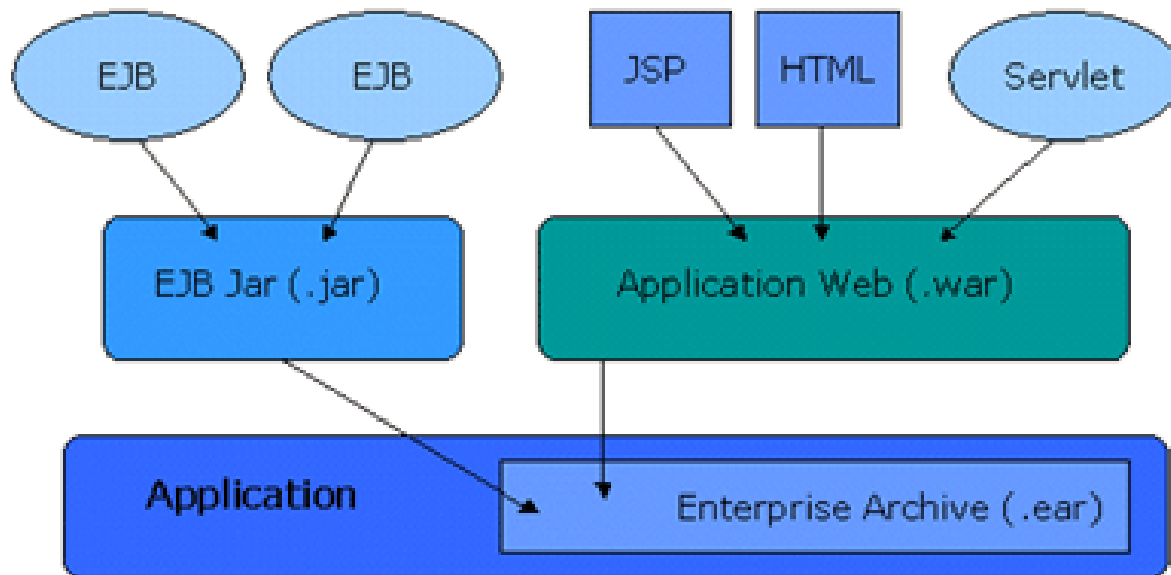
**3 - La servlet** contrôle la validité de la requête HTTP.

**4 - Elle** instancie les beans de données pour accéder aux données.

**6 - Elle** invoque la JSP pour générer la page HTML qui contient le résultat de la requête.

# ASSEMBLAGE ET DEPLOIEMENT D'APPLICATIONS J2EE 3/6

## ■ Architecture d'une application J2EE :



### • 3 couches :

❑ *Les composants.*

❑ *Les modules regroupant les composants*

❑ *Les applications regroupant les modules*

- Les modules et les applications correspondent physiquement à des fichiers d'archives : archive EJB JAR (.jar) pour un module EJB, archive WAR pour un module web, archive EAR pour une application.



# ASSEMBLAGE ET DEPLOIEMENT

## D'APPLICATIONS J2EE 4/6

---

- **Module Web (.war).** Selon la spécification J2EE, une application Web doit avoir la structure suivante:
- un répertoire racine public contenant les pages HTML, les pages JSP, les images...
  - un repertoire *WEB-INF* situé dans le répertoire racine de l'application web.
  - un fichier *web.xml* situé à la racine de *WEB-INF* : c'est le descripteur de déploiement de l'application web.
  - un répertoire *WEB-INF/classes* contenant les classes compilées de l'application (servlets, classes auxiliaires...).
  - un répertoire *WEB-INF/lib* contenant les fichiers JAR de l'application (drivers JDBC, frameworks empaquetés...).

Le tout peut être empaqueté dans une archive sous la forme d'un fichier WAR (réalisé avec l'utilitaire jar du JDK).

# ASSEMBLAGE ET DEPLOIEMENT

## D'APPLICATIONS J2EE 5/6

---

- **Module EJB (.jar).** Selon la spécification J2EE 1.2, un fichier JAR doit avoir la structure suivante :
- un répertoire *META-INF/* contenant un descripteur de déploiement XML du module EJB, nommé *ejb-jar.xml*
- les fichiers *.class* correspondant aux interfaces locale (home interface) et distante (remote interface), à la classe d'implémentation, et aux classes auxiliaires (classes d'exception par exemple) des EJBs, situées dans leur package.

Le tout peut être empaqueté dans une archive sous la forme d'un fichier JAR.

Nouvelle version JEE 1.5

# ASSEMBLAGE ET DEPLOIEMENT

## D'APPLICATIONS J2EE 6/6

---

- **Application d'entreprise (.ear).** Selon les spécifications J2EE, une application d'entreprise doit avoir la structure suivante :
  - un répertoire *META-INF/* contenant le descripteur de déploiement XML de l'application J2EE nommé *application.xml*. C'est dans ce descripteur que l'on définit les modules web et EJB qui constituent l'application d'entreprise. On y précise par exemple sur quelle racine du serveur web (placé en frontal devant le serveur d'application) doit résider l'application web.
  - les fichiers archives .JAR et .WAR correspondant aux modules EJB et aux modules Web de l'application d'entreprise.

Le tout peut être empaqueté dans une archive sous la forme d'un fichier EAR.