Développement de composants distribués

Chapitre 1 : Présentation de l'architecture JEE

Ahmed JEMAL
Email :
jmlhmd@gmail.com

Soumaya Marzouk
Email :
Soumaya.marzouk@gmail.com

Plan

- Introduction à JEE
- Fonctionnement et structure d'une application JEE
- Architecture multi-tiers
- Architecture en trois couches : MVC
- Composants JEE

Introduction à JEE

Introduction à JEE (1/3): Présentation générale

- Java Enterprise Edition (JEE) est un ensemble de spécifications coordonnées et pratiques permettant :
  - le développement
  - le déploiement
  - la gestion

  … des applications **multi-tiers réparties**.
- JEE est fondée sur la plateforme *Java Standard Edition* (JSE), tout en lui ajoutant les fonctionnalités nécessaires pour fournir une plateforme :
  - complète
  - stable
  - sécurisée
  - rapide

Introduction à JEE (2/3): Services

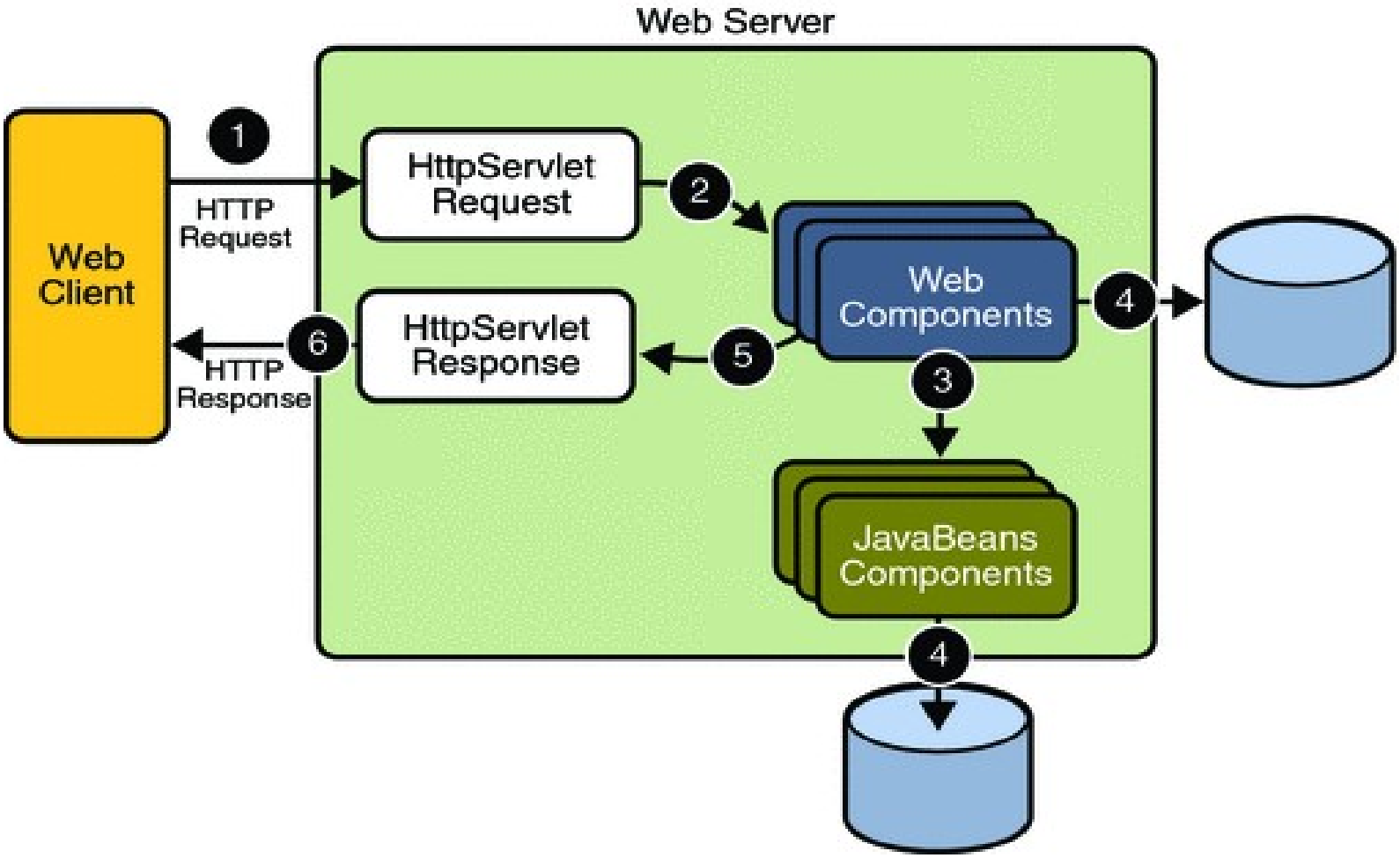
- JEE fournit un ensemble de **services** permettant aux **composants** de dialoguer entre eux. Parmi les services les plus importants :
  - Servlets
  - Java Server Pages (JSP)
  - Enterprise JavaBeans (EJB)
  - Java Transaction API (JTA)
  - Java Persistence API (JPA)
  - Java Authorization and Authentication Service (JAAS)
  - Java Message Service (JMS)
  - Java Naming and Directory Interface (JNDI)

Introduction à JEE (3/3): Bénéfices

- Fournit une approche **orientée composant** pour permettre de réaliser une application distribuée
  - L'application est un assemblage de composants
- Fondée sur l'architecture *multi-tiers*
  - Composants réutilisables
  - Séparation des préoccupations
- Composants indépendants des interfaces de programmations (APIs) et des constructeurs de technologies
  - Grâce à la notion de **conteneur** qui offre aux composants un environnement dans lequel ils peuvent s'exécuter

Fonctionnement et structure d'une application JEE

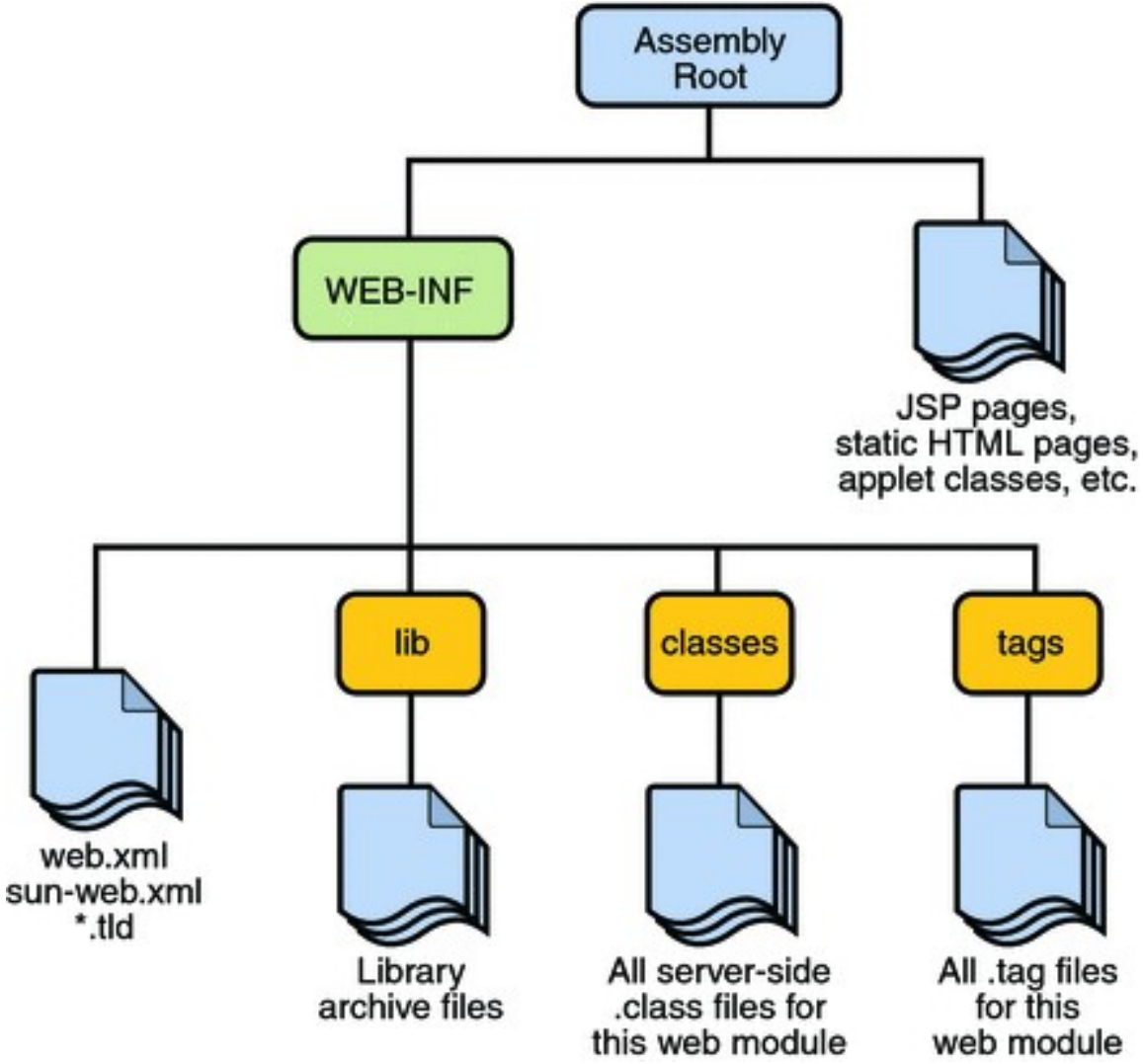
Fonctionnement et structure d'une application JEE (1/4): Traitement de la requête (1/2)



Fonctionnement et structure d'une application JEE (2/4): Traitement de la requête (2/2)

1. Le client envoie une requête HTTP au serveur web
2. Le serveur web qui implante la technologie Servlet ou JSP reçoit la requête et la convertit en objet de type **HTTPServletRequest**. Cet objet est délivré au composant web de l'application
3. Le composant web peut éventuellement interagir avec un composant EJB (Enterprise JavaBean) qui réalise la partie *métier* du comportement de l'application
4. Le composant web ou bien le composant EJB peut éventuellement interagir avec une base de données
5. Le composant web crée une réponse de type **HTTPServletResponse**
6. Le serveur web convertit l'objet de type **HTTPServletResponse** en une réponse HTTP qu'il renvoie au client

Fonctionnement et structure d'une application JEE (3/4): Structure du module web (1/2)



Fonctionnement et structure d'une application JEE (4/4): Structure du module web (2/2)

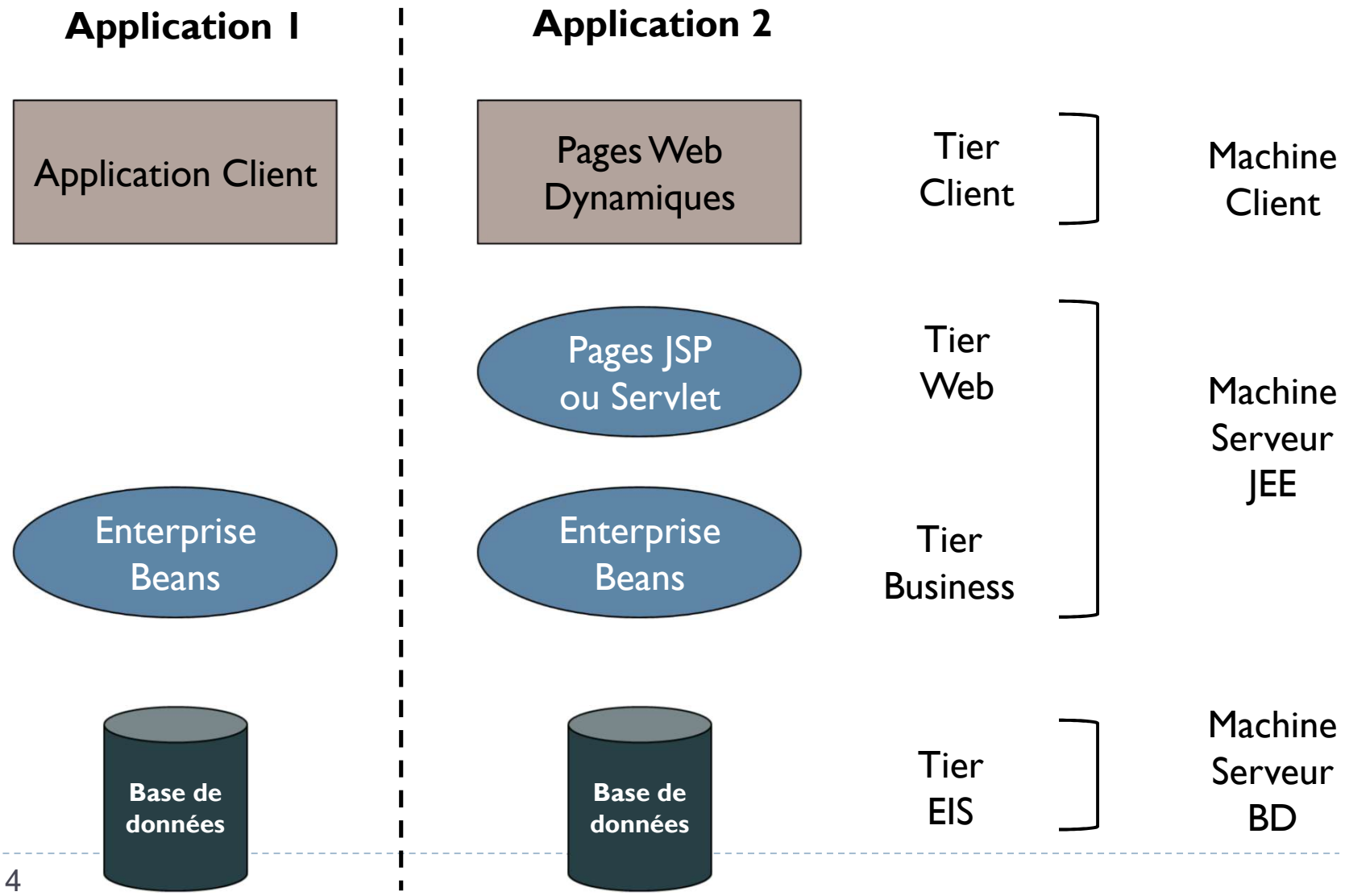
- La structure d'un module web est particulière. Le répertoire racine du module web constitue la racine de l'application qui abrite les pages JSP, HTML, les différentes classes Java et les ressources de l'application (images…).
- Le répertoire racine contient un sous répertoire appelé WEB-INF qui contient les éléments suivants :
  - **web.xml :** fichier décrivant comment l'application doit être déployée.
  - **tag :** répertoire contenant des fichiers `.tag` définissant des balises JSP personnalisées.
  - **classes :** répertoire contenant les classes situées sur le serveur (servlets, classes utilitaires, JavaBeans…).
  - **lib :** répertoire contenant les fichier JAR requis par les classes du serveur.

Architecture multi-tiers

Architecture multi-tiers (1/2)

- L'applications est divisée en composants, selon la fonctionnalité et la localité
- Ces composants sont installés sur différentes machines, appelées **tiers**
- Les tiers :
  - Le tier client : composants s'exécutant sur le client
  - Le tier web : composants s'exécutant sur le serveur
  - Le tier business : composants s'exécutant sur le serveur
  - Le tier EIS (Enterprise Information System) : composants s'exécutant sur un autre serveur (bases de données…)
- Il n'y a pas *nécessairement* 4 tiers dans une application

Architecture multi-tiers (2/2)

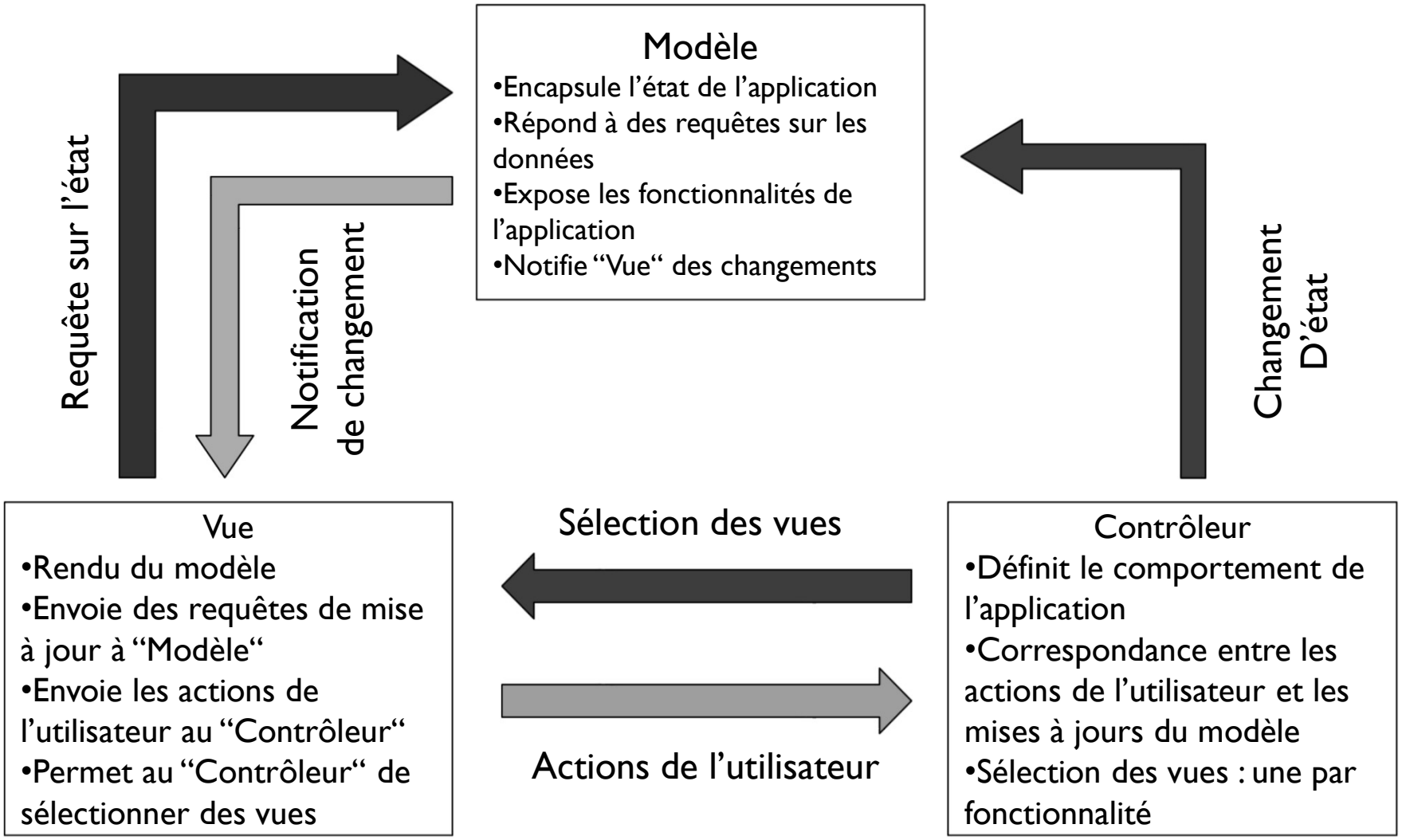


Architecture en trois couches (MVC)

Architecture en trois couches (MVC)

- Model-View-Controller (MVC) : modèle architectural de développement logiciel (Design pattern)
- Permet d'isoler la logique métier (contrôleur) de la présentation (Vue) et des données manipulées (Modèle)
- Très répandue car il permet de simplifier la distribution de fonctionnalités au sein d'une application répartie complexe
- Très populaire dans le domaine des applications web
- Favorise la réutilisation des modèles, composants et code
- Deux versions: MVC1 et MVC2
- Plusieurs implémentations sont disponibles: Struts, Spring-MVC, etc.

MVC : Architecture et interactions



MVC : Modèle

- Représente les données *métiers* et fournit un ensemble d'opérations et de règles qui pilotent l'accès et la modification de ces données
- Constitue souvent une approximation d'une grandeur du monde réel (salaire, poids, température…)
- Notifie la partie "vue" quand des changements sur les données surviennent et lui permet d'envoyer des requêtes pour interroger l'état du modèle.
- Permet au contrôleur d'accéder aux fonctionnalités de l'applications encapsulées par le modèle

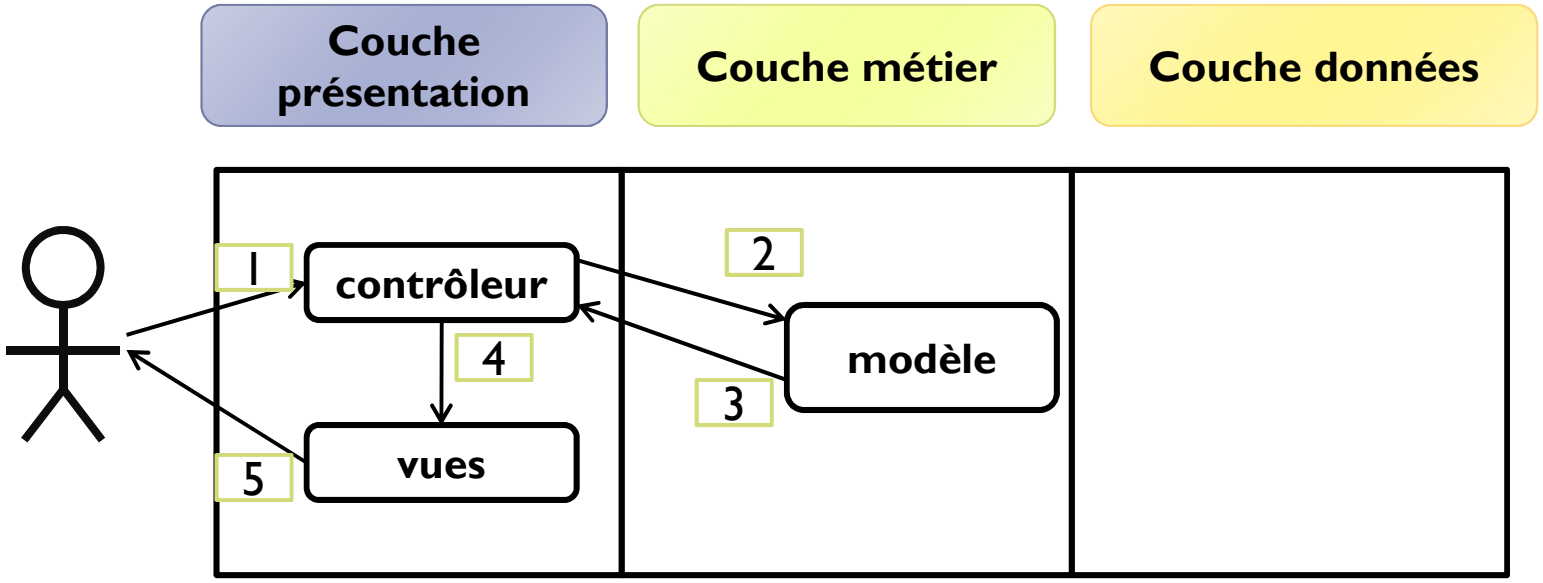
MVC : Vue

- Réalise le "rendu" de la partie "modèle" : reçoit les données à partir du modèle et spécifie comment elles devraient être représentées.
- Met à jour la représentation des données quand celles-ci sont changées dans le modèle. Deux méthodes de mise à jour :
  - **Push :** "Vue" s'enregistre auprès de "modèle" et est notifié chaque fois qu'il y a un changement
  - **Pull :** "Vue" est responsable d'interroger "modèle" avec des requêtes pour connaître les valeurs les plus récentes des données
- Transfère les actions de l'utilisateur vers la partie "Contrôleur" et permet à celui-ci de sélectionner les vues adéquates pour l'utilisateur

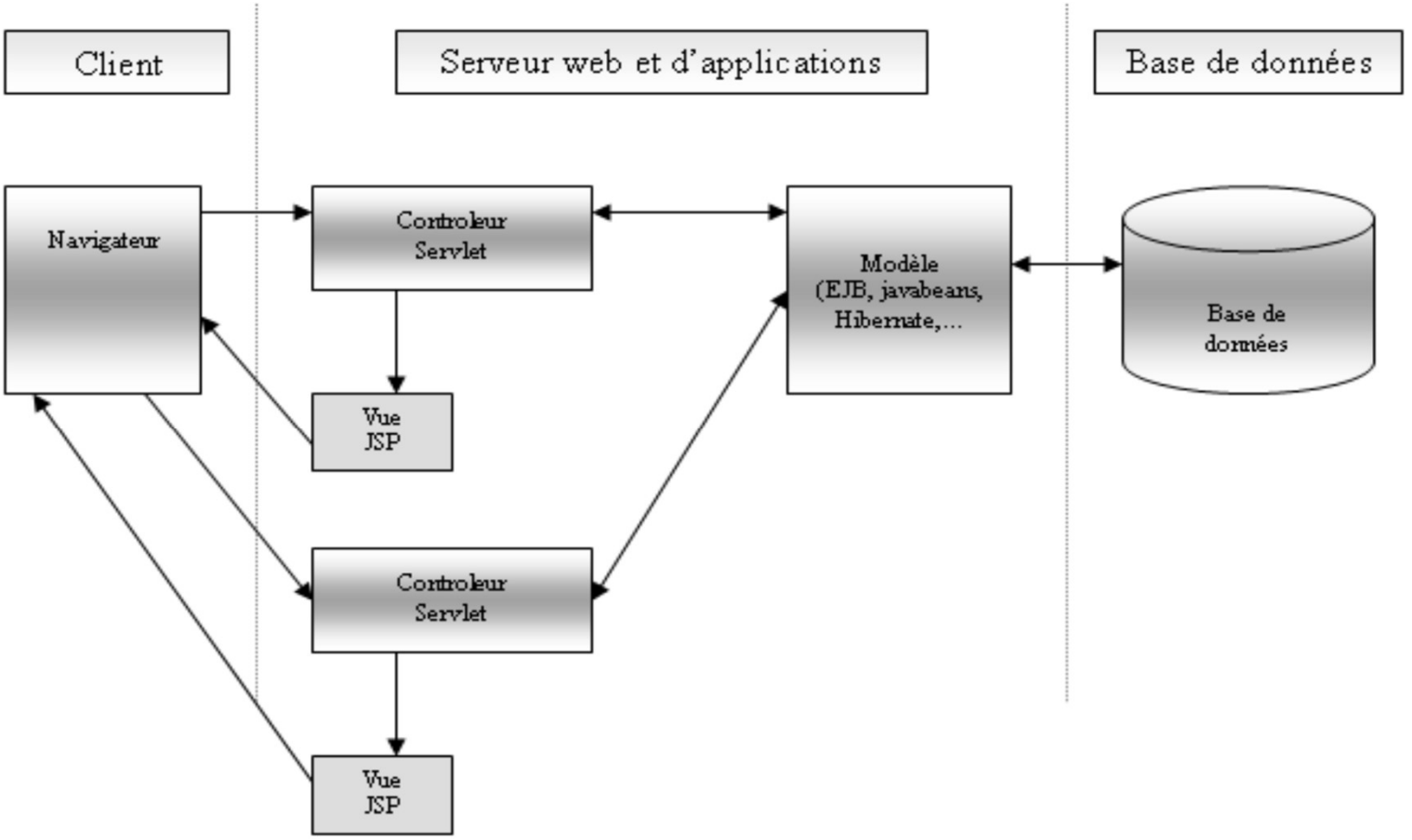
MVC : Contrôleur

- Définit le comportement *métier* de l'application
- Aiguille les requêtes de l'utilisateur et sélectionne les vues en se basant sur les actions entre l'utilisateur et les résultats des traitements sur les données
- Traduit les interactions de l'utilisateur avec la partie "Vue" en actions devant être effectuées sur le "modèle" :
  - Dans une application Web, les entrées sont des requêtes HTTP (GET ou Post)
  - Dans une application Autonome (GUI Java…) les entrées sont, par exemple, des choix de menus, des clicks sur des boutons…
- Les actions effectuées par le contrôleur sur le modèle peuvent inclure :
  - Activer un processus métier
  - Changer l'état du modèle
  - Mettre à jour les données

Illustration du modèle MVC

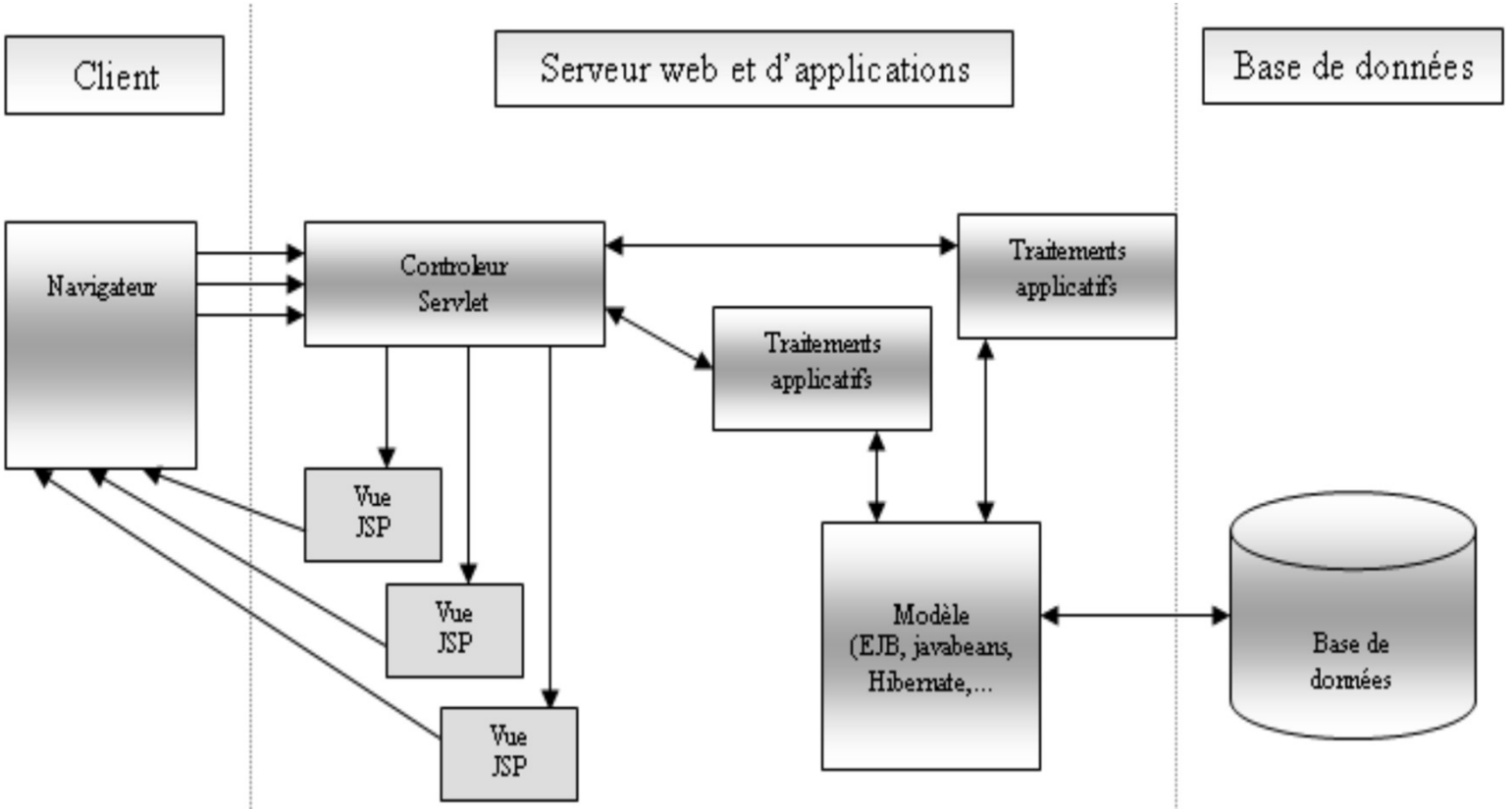


MVC1



L'inconvénient : une multiplication du nombre de servlets nécessaire à l'application : L'implémentation de plusieurs servlets nécessite beaucoup de code à produire et augmente les points d'accès à l'application ce qui augmente sa vulnérabilité.

MVC2



- **Une évolution du modèle 1: une unique servlet fait office de contrôleur**
- **Le contrôleur est divisé en deux parties:**
  - **Un point d'entrée unique à toute l'application qui oriente les requêtes vers**
  - **Les traitements applicatifs à invoquer dynamiquement**

Composants JEE

Composants JEE: Définition et motivations (1/2)

- Les applications distribuées JEE sont constituées par l'assemblages de plusieurs **composants**.
- Un composant est une unité logicielle autonome (*self-contained*).
- Un composant contient un ensemble de classes Java et d'autres fichiers de support lui permettant de communiquer avec les autres composants.
- Les composants sont écrits en Java et compilés d'une manière similaire aux programmes Java classiques, avec trois différences :
  - Les composants JEE sont assemblés pour former l'application JEE
  - Il existe une phase de vérification pour garantir que les composants sont conformes à la spécification JEE
  - Les composants sont déployés dans un environnement de production

Composants JEE: Définition et motivations (2/2)

- Il existe trois familles de composants :
  - Les clients et les applets sont des composants qui s'exécutent sur les clients d'une application JEE
  - Les Servlets, les Java Server Pages (JSP) et les Java Server Faces (JSF) sont des composants web qui s'exécutent sur le serveur
  - Les Enterprise JavaBeans (EJB) sont des composants *métier* qui s'exécutent sur le serveur
- Motivations :
  - Favoriser la réutilisation de code
  - Permettre la séparation des préoccupations
  - Permettre le développement en équipe

Composants JEE : Clients (1/2)

Un client JEE peut être un client web (avec applet ou pas) ou une application cliente autonome

- Clients web
  - Formé de deux parties : (1) un ensemble de pages web dynamiques (HTML, XML…) générées par des composants web s'exécutant sur le tier-web et (2) un navigateur qui effectue le *rendu* (affichage et organisation) de ces pages
  - Souvent appelé **client léger** car il n'effectue pas les opérations complexes (interrogation de bases de données…) lui même. Ces opérations sont souvent déléguées à la partie métier du serveur (EJB) pour plus de performance et de sécurité
- Applets
  - Une page reçue de la part du tier-web peut contenir un applet.
  - Un applet est une application Java de petite taille qui s'exécute dans une machine virtuelle attachée au navigateur web
  - Un applet peut se connecter lui même aux bases de données (**client lourd**) ou déléguer les connexions à la partie serveur (**client léger**)

Composants JEE : Clients (2/2)

- Applications clientes autonomes
  - Une application cliente est une application classique qui s'exécute sur le client
  - Fournit une alternative plus riche que les interfaces web (généralement des interfaces graphiques…). Mais les applications en mode ligne de commande existent aussi
  - Accède directement aux composants EJB qui s'exécutent sur le tier-métier… Mais peut aussi établir des connexions HTTP avec d'autres composants web (servlets…)
  - Écrite généralement en Java… Mais peut l'être dans d'autres langage de programmation pour permettre à la plateforme JEE d'interagir avec des composants non-Java

Composants JEE : Composants Web

Les composants web sont soient des servlets ou des JSP

- Les servlets sont des classes Java qui s'exécutent sur le tier-web de l'application. Il traitent des **requêtes** envoyées par le client et émettent des **réponses**.
- Les JSP sont des documents texte qui s'exécutent sur le tier-web. Il sont similaires aux servlets avec deux différences :
  - Sont compilés avant de traiter les requêtes
  - Favorisent une approche plus naturelle pour la création d'un contenu web statique/dynamique
- Les classes situées sur le serveur et qui peuvent être utilisées par les servlets et les JSP peuvent être *emballées* avec les composants web mais elles ne font pas partie de ces composants

Composants JEE : Enterprise JavaBeans

- Les EJB sont des composants métier, effectuant le comportement de l'application (banque, vente, finance…)
- Les EJB s'exécutent sur le serveur et jouent un rôle intermédiaire entre les clients et le tier EIS (Enterprise Information System)
- Un EJB reçoit des données de la part du client, les traite (si nécessaire), les envoie au tier EIS pour les stocker. Il permet aussi de retrouver des données de la part du tier EIS, de les traiter (si nécessaire) et de les renvoyer vers les clients.

Bibliographie

- **[1]** Java 2 Platform Enterprise Edition Specification, v1.4
  http://java.sun.com/j2ee/j2ee-1_4-fr-spec.pdf
- **[2]** The Java EE 5 Tutorial
  http://java.sun.com/javaee/5/docs/tutorial/doc/JavaEETutorial.pdf
- **[3]** Faq JEE
  http://java.developpez.com/faq/javaee/
- **[4]** MVC
  http://java.sun.com/blueprints/patterns/MVC-detailed.html
- **[5]** Java™ Persistence API, Linda DeMichiel et Michael Keith