Chapitre 1

Architecture MVC et JEE

Patron de conception

- Les patrons de conception décrivent des solutions standard pour répondre à des problèmes d'**architecture** et de **conception** des logiciels.
- Un Patron de conception (design pattern) est une solution standard pour répondre à type ou une catégorie de problèmes.
- C'est une **architecture logicielle** pour structurer une application.
- Les **patrons de conception** sont basés sur le paradigme objet.

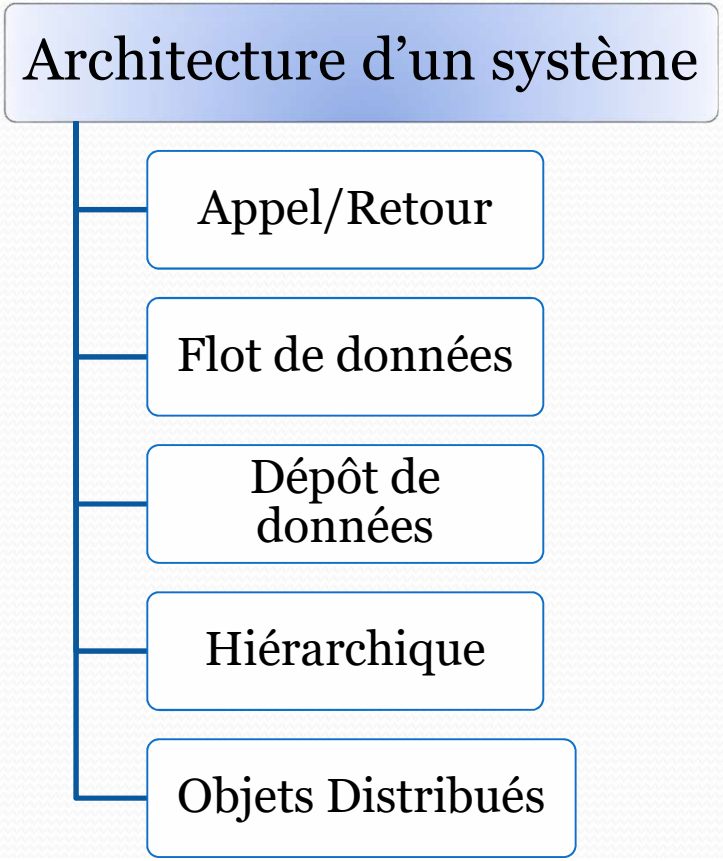
Définition de l'architecture d'un système

L'architecture d'un système logiciel décrit son organisation générale :

- ses composants
- les relations entre **ses composants** et avec **l'environnement**
- les principes de conception et d'évolution du système

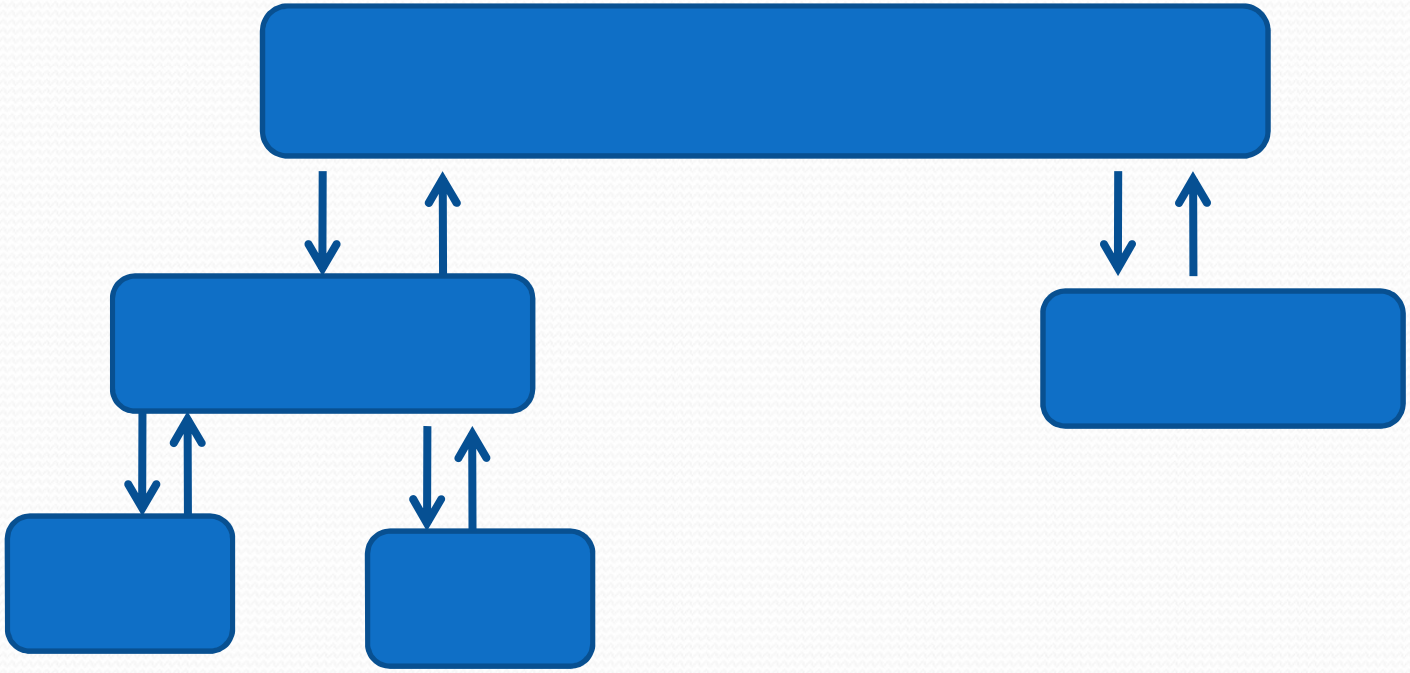
Styles d'Architecture

- Un style d'architecture est caractérisé par une forme particulière d'**interaction** entre ses composants.
- Cinq catégories principales d'architectures :



Appel/Retour

Les composants utilisent les appels de procédure pour transférer le contrôle et échanger des données(RPC).



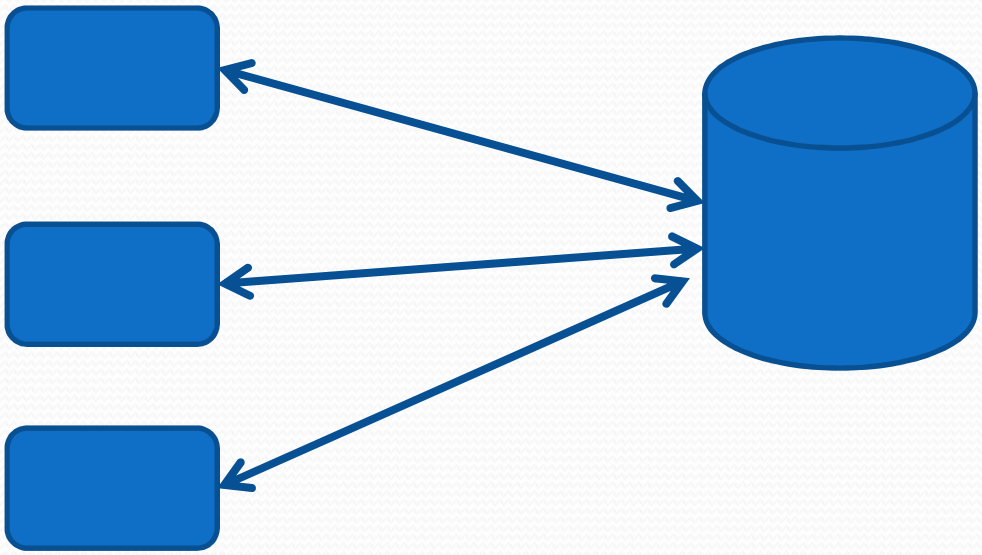
Flot de contrôle

- Des composants autonomes qui interagissent exclusivement via les flots de données qui circulent entre eux.
- Exemples : traitement batch, tubes, filtres

Dépôt de données

Se caractérise par l'existence d'un dépôt de données central manipulé par tous les composants.

Exemples : bases de données, tableau noir



Hiérarchique

Le système est un empilement de niveaux dont les composants interagissent selon des règles bien définies.

Exemples:

- Architectures en couches,
- Architectures en tiers

Composants distribués

- Des composants autonomes coopèrent via des échanges d'**évènement** ou de **messages**.
- Il comporte cinq sous-catégories :
  - **Architectures client/serveur**
  - Architectures pair à pair
  - Architectures à base de messages ou événements
  - Objets distribués
  - Architectures orientées services (SOA)

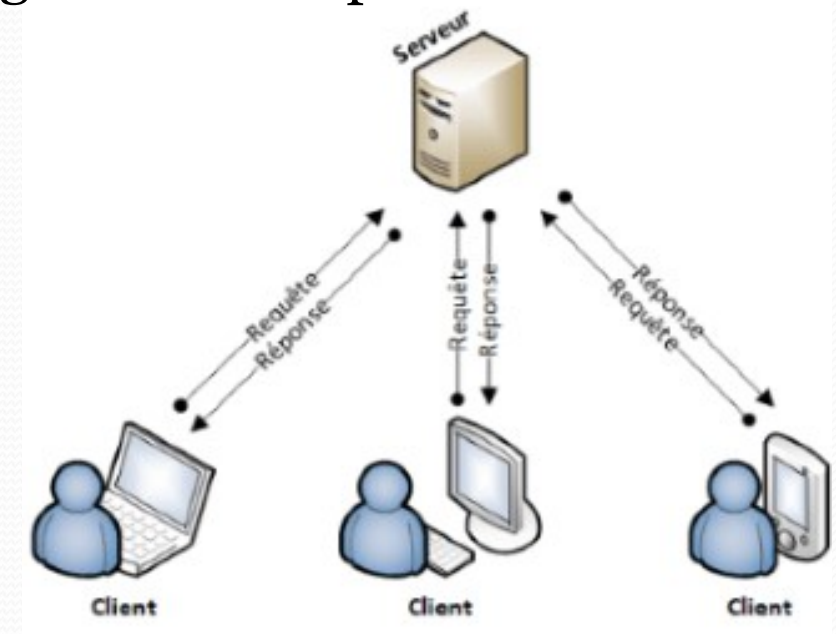
Architecture d'une application web

Architecture d'une application web

- Une application Web est généralement constituée de trois parties ou tiers :
  - Présentation : les pages Web
  - Métier : regroupe tous les traitements de l'application Web
  - Données ou persistance : regroupe toutes les données manipulées par l'application Web ainsi que leur représentation.
- Le patron de conception adopté pour les applications web est le modèle **Client/Serveur**

Architecture d'une application web

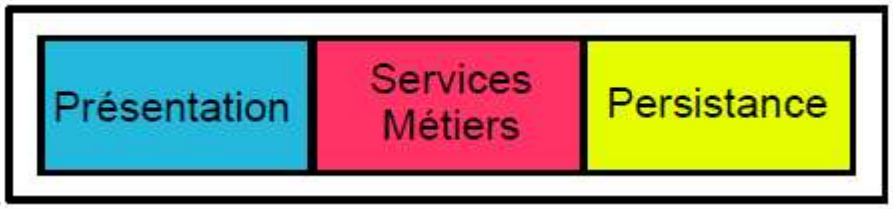
- Le modèle Client/serveur est une architecture logicielle dans laquelle les programmes d'application, appelés clients, font appel, dans un cadre réseau, à des services distants fournis par un serveur.
- L'interaction entre les clients et le serveur est réalisée via des messages de requêtes et de réponse.
- Cette architecture est largement adoptée dans les applications Web



Arhitecture d'une application Web

- De nombreuses variantes de placement des tiers d'une application web sont possibles :

- Modèle 1-tiers
  - Tout est sur la même machine

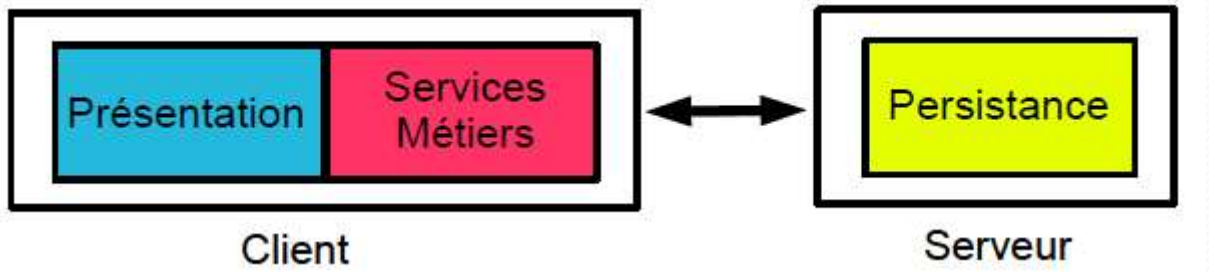


Architecture 2 – tiers

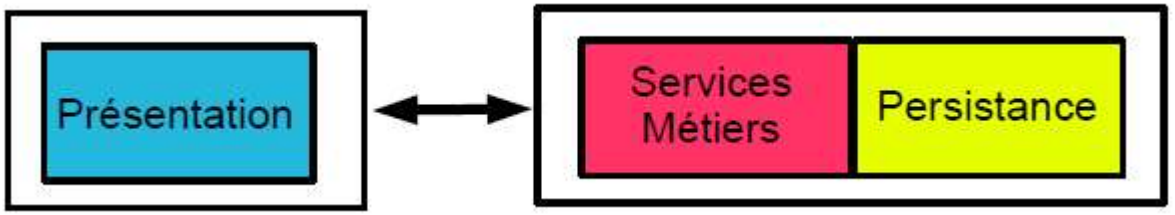
- **Client / serveur de base, avec 2 éléments**
  - Client : présentation, interface utilisateur
  - Serveur : partie persistance, gestion physique des données
- **Les services métier / la partie applicative :** peuvent être
  1. Soit entièrement coté client, intégrés avec la présentation
     - La partie serveur ne gère que les données
       - Ex : traitement de texte avec serveur de fichiers distants
       - Ex : application accédant à une BDD distante
  2. Soit entièrement coté serveur
     - La partie client ne gère que l'interface utilisateur
     - L'interface utilisateur peut même être exécutée sur le serveur
       - Fonctionnement mode terminal / mainframe
       - L'utilisateur a un simple écran / clavier / souris pour interagir à distance avec l'application s'exécutant entièrement sur le serveur
  3. Soit découpés entre la partie serveur et la partie client

Architecture 2 – tiers

Client : Présentation + applicatif

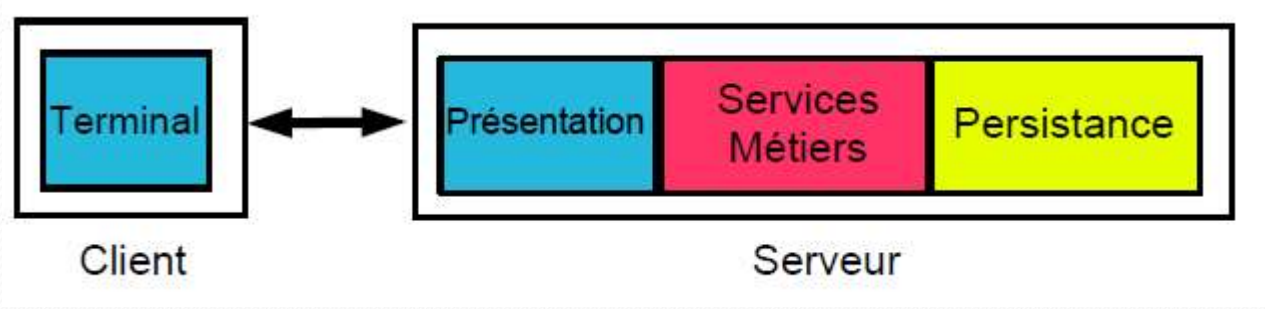


Serveur : Applicatif + gestion des données

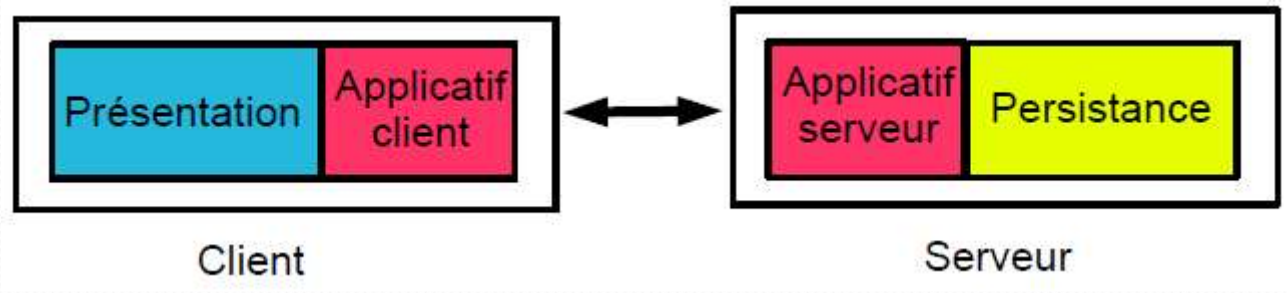


Architecture 2-tiers

- Terminal : le client intègre un minimum de la partie présentation



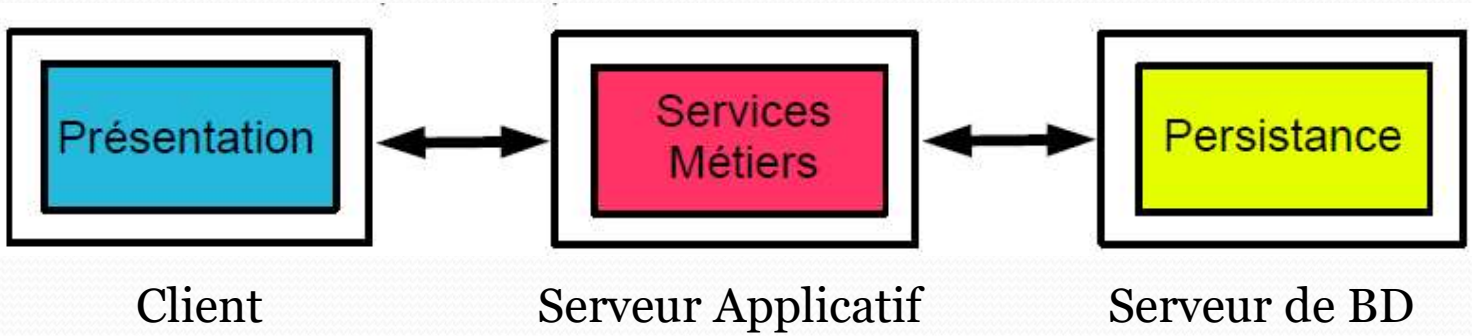
- Applicatif : découpé entre client et serveur



Architecture 3 – tiers

Les 3 principaux tiers s'exécutent chacun sur une machine différente

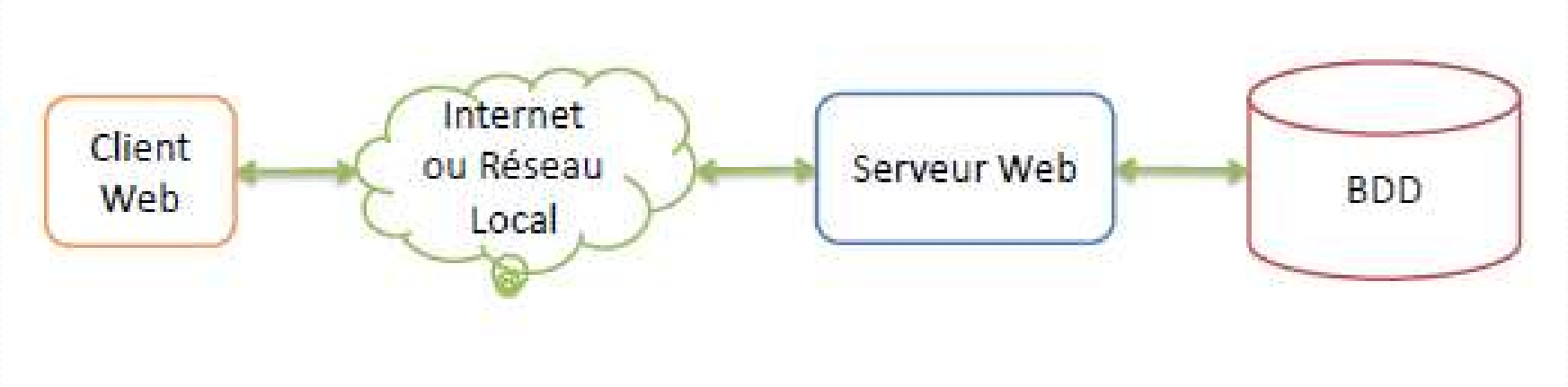
- Présentation
  - Machine client
- Applicatif / métier
  - Serveur d'applications
- Persistance
  - Serveur de base de données



Modèle MVC

Motivation du modèle MVC

• Une application Web suit le principe d'une architecture client-serveur :



• L'interaction est faite via des messages de **Requête/Réponse**
•Le modèle MVC est basé sur une architecture Client/Serveur.

Motivation du modèle MVC

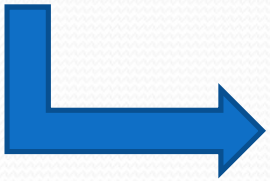
Application interactive :

- **Architecture simpliste** : regroupement du **code**, des **données** et de la **visualisation** dans une même page :
  - Problème de **duplication** des **données**, des **visualisations** et des **traitements**
  - Manque de flexibilité (un seul développeur peut intervenir sur la page, la modification de la visualisation peut entraîner une modification du code et des données)

Motivation du modèle MVC

Application Interactive : Deux grandes parties :

- **la partie visible** (front End) : l'IHM (Interface Homme-Machine), concerne deux parties :
  - **les entrées** : actions de l'utilisateur
  - **les sorties** : la visualisation des résultats de l'action de l'utilisateur
- **la partie caché** (back End) : le stockage, l'accès et le traitement des données



cœur de l'application

- Décomposition de l'application en trois parties :
  - le Modèle, la Vue, le Contrôleur

Modèle MVC

- Le patron architectural MVC concerne les applications interactives et vise à séparer les trois éléments :
  - **Traitements**
  - **Données**
  - **Présentation**
- Ce découplage facilite :
  - l'évolution des interfaces utilisateur
  - Proposer plusieurs versions d'interfaces adaptées à différents contextes.

Modèle MVC

- **But**
  - Isoler la donnée elle-même de sa présentation
  - Distinguer la consultation de la modification
- **Principe**
- Les 3 composantes suivantes d'une donnée sont distinguées et séparées :
  - Le modèle (la structure de la donnée)
  - La vue (la représentation de la donnée pour affichage)
  - Le contrôleur (les moyens de modifier la valeur)

La couche Modèle

Définit les fonctionnalités de l'application :

- Représentation des données
- Accès aux données
- Traitement des données

- **Rôle**
  - Encapsuler les propriétés d'une donnée
  - Être indépendant des vues et contrôleurs
- **Conséquences**
  - Définir les accesseurs aux propriétés de cette donnée

La couche Vue

- **La vue** affiche les données, provenant exclusivement du modèle, pour l'utilisateur et/ou reçoit ses actions.
- **Aucun traitement autre que la gestion de la présentation n'y est réalisé.**

➔ **Sorties de l'application**

- Visualisation(s) des données issues du Modèle
- Assure la consistance de la visualisation avec la représentation des données dans le modèle

- **Rôle**
  - Représenter la donnée encapsulée via un modèle
  - Se maintenir à jour lorsque le modèle est modifié
- **Conséquences**
  - Doit s'enregistrer comme écouteur au niveau du modèle

La couche contrôle

- son rôle est de **traiter les événements** en provenance de l'interface utilisateur pour :
  - les transmettre au modèle pour le faire évoluer
  - ou à la vue pour modifier son aspect visuel (pas de modification des données affichées mais des modifications de présentation (couleur de fond, affichage ou non de la légende d'un graphique, …).

➔**Gestion des entrées de l'application**

- Comportement de l'application face aux actions de l'utilisateur (programmation événementielle (ex: clic sur un bouton submit)
- Traduction des actions de l'utilisateur en actions sur le modèle
- Traduction des résultats des traitements du modèle (réactions) et des actions de l'utilisateur en action sur la vue (modification appropriée de la vue)

Avantages du modèle MVC

- Structuration de l'application
- Indépendance entre les données, la visualisation et le comportement de l'application,
- Modularité et réutilisation des éléments : Plusieurs vues possibles,
- Changement/échange de contrôleurs, contrôleurs multiples,
- Changement de représentation ou de moyens de/stockage des données au niveau du modèle,
- Solution particulièrement adaptée à la programmation orientée objet,
- Synchronisation vue/modèle a travers le contrôleur.

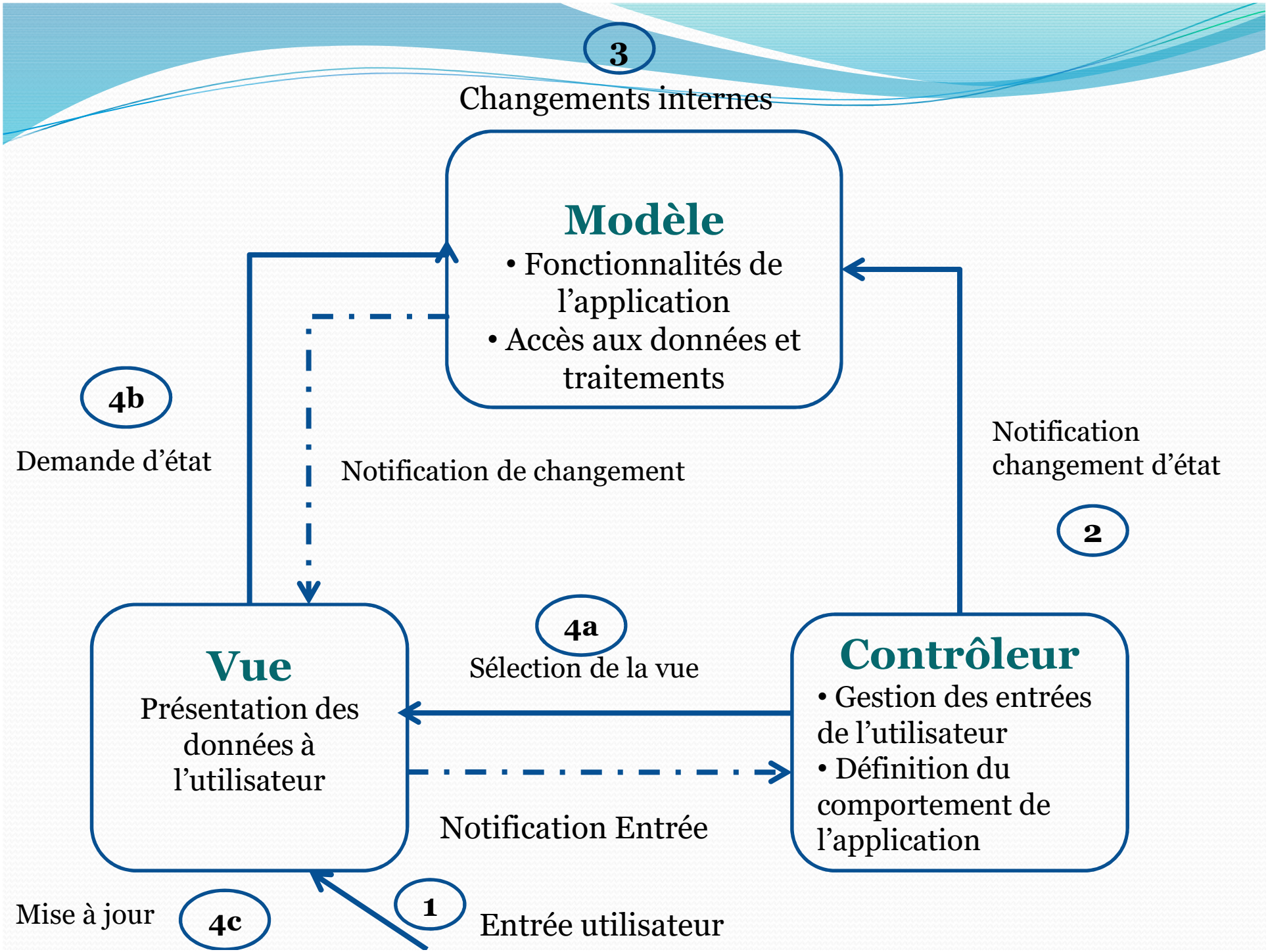
Inconvénients

- Complexité accrue
- Mises à jour potentiellement trop nombreuses
- Augmentation du temps d'exécution
- Le contrôleur et la vue sont souvent lies au modèle

MVC et applications web

Principe de fonctionnement :

1. Par le biais d'une page web, l'utilisateur émet une requête HTTP au serveur web en cliquant sur un lien ou sur un bouton. Cette requête est prise en charge par le contrôleur.
2. Le **contrôleur** déclenche les traitements associés à la requête de l'utilisateur définis dans le **modèle**.
3. Le contrôleur sélectionne ensuite la page web qui sera en charge de la construction ou l'affichage de la réponse et lui transmet les entités contenant les données à afficher.
4. La **page web** construit la réponse en faisant appel aux accesseurs des entités.
5. La réponse HTTP est transmise au navigateur qui l'affiche sous forme de page web.


3
Changements internes
Modèle
• Fonctionnalités de l'application
• Accès aux données et traitements
4b
Demande d'état
Notification de changement
Notification changement d'état
2
4a
Sélection de la vue
Vue
Présentation des données à l'utilisateur
Contrôleur
• Gestion des entrées de l'utilisateur
• Définition du comportement de l'application
Notification Entrée
Mise à jour
4c
1
Entrée utilisateur

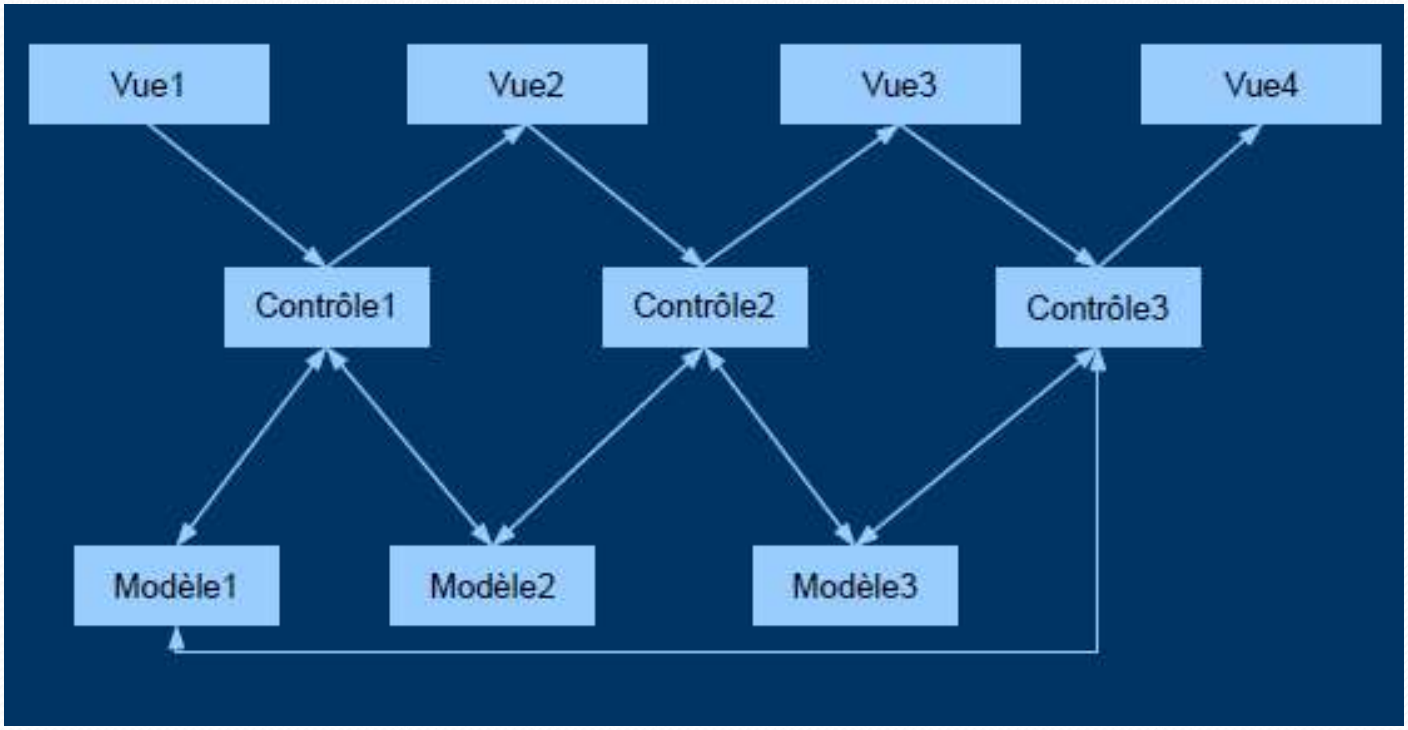
MVC et applications web

MVC1

- Modèle
  - Des classes
- Vue
  - Les pages web
- Contrôleur
  - Une classe de contrôle qui traite l'information de la page source en modifiant les données si nécessaire
  - Affiche la nouvelle page

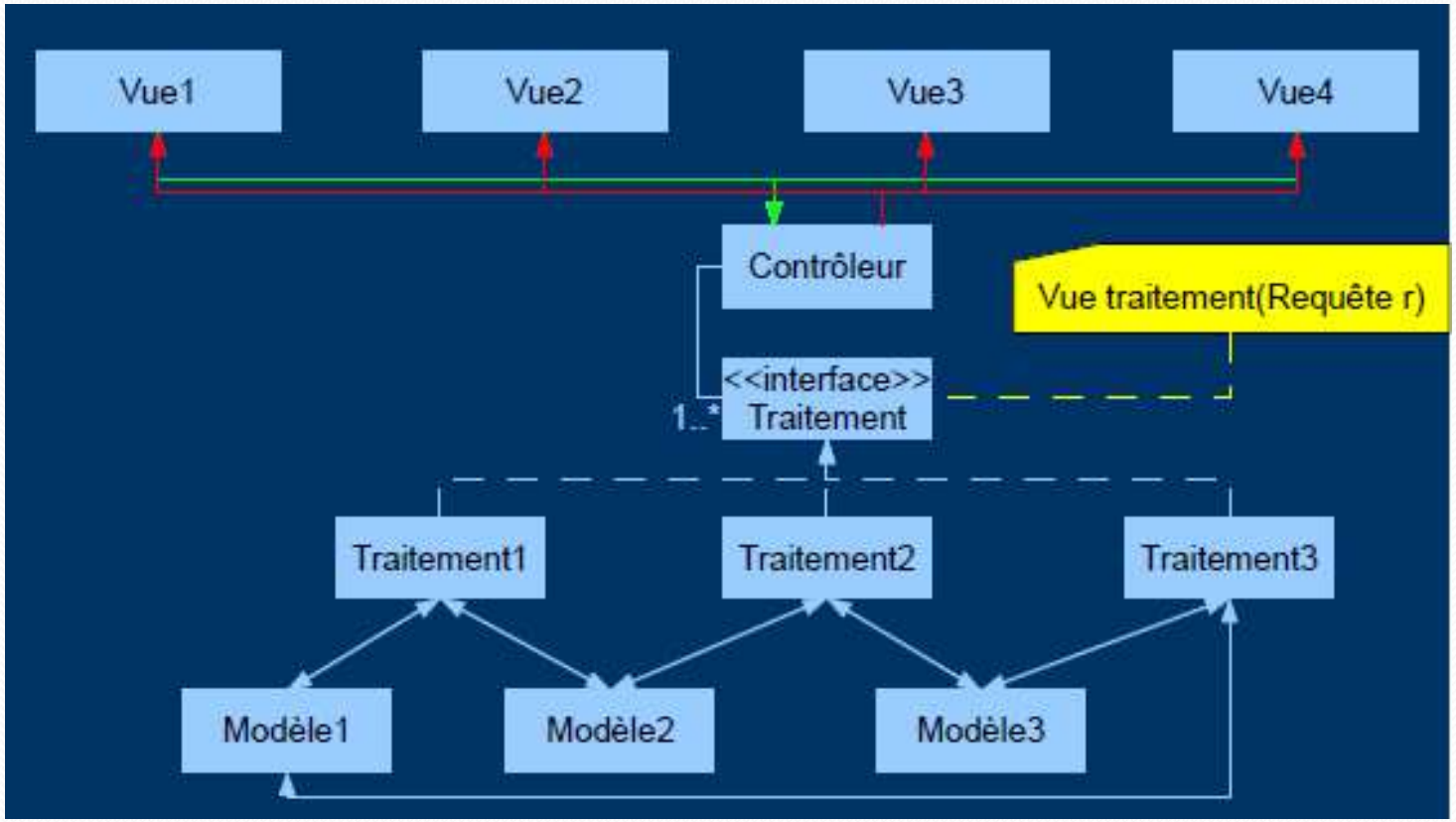
➔ **La vue sert à configurer le contrôle**

➔ Plusieurs Contrôleurs


Vue1
Vue2
Vue3
Vue4
Contrôle1
Contrôle2
Contrôle3
Modèle1
Modèle2
Modèle3

- **Inconvénient MVC1** :
  - Multiplication du nombre de contrôleurs
  - Lourdeur de déclaration au niveau du Serveur d'Application
  - Mises à jour peu faciles
- **Modèle MVC2**
  - Un seul contrôleur servant d'aiguilleur vers le modèle et la vue
  - Elle a l'architecture simple :

```
switch(pageSource) {
    case page1 : pageSuite=traitement1();afficher(pageSuite);break;
    case page2 : pageSuite=traitement2();afficher(pageSuite);break;
...
}
```

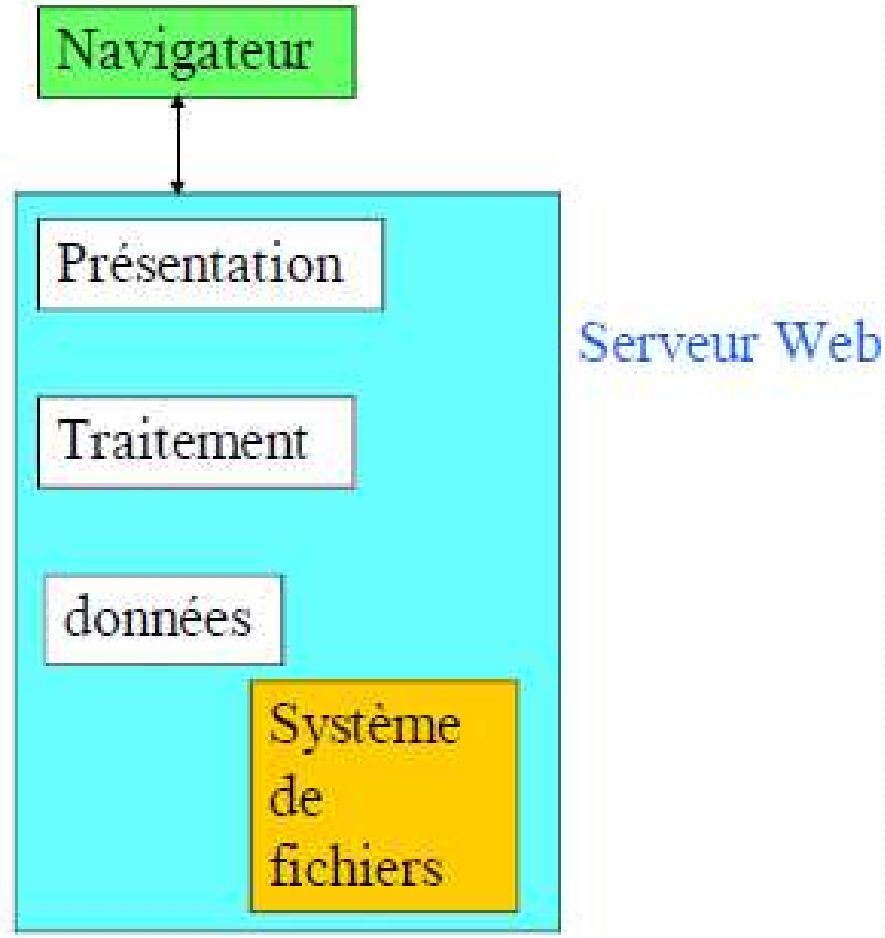

Vue1
Vue2
Vue3
Vue4
Contrôleur
Vue traitement(Requête r)
<<interface>>
Traitement
1..*
Traitement1
Traitement2
Traitement3
Modèle1
Modèle2
Modèle3

Implémentation JEE de MCV

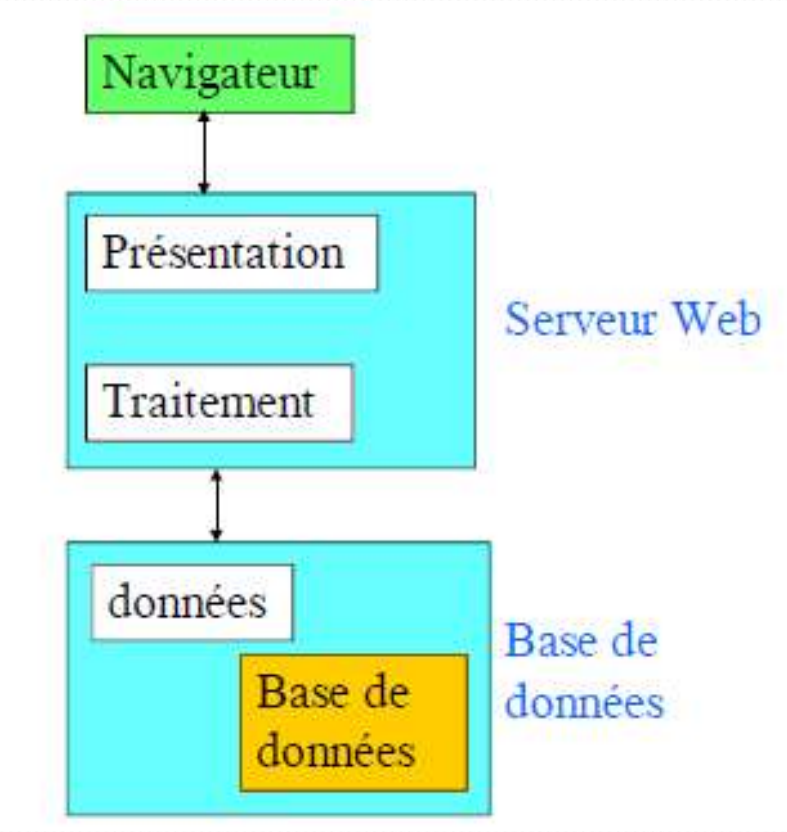
JEE : **J**ava **E**nterprise **E**dition

- L'environnement JEE fournit un ensemble d'APIs permettant de développer des sites Web dynamiques avec une technologie Java.
- définie par Sun http://java.sun.com/
- basée sur Java : Standard Edition, Enterprise Edition, Micro Edition
- applications types : systèmes d'information entreprise, commerce électronique…
- ensemble de technologies pour construire des applications réparties
- implantation de référence : JEE 5 SDK

Architecure JEE(1 tiers)



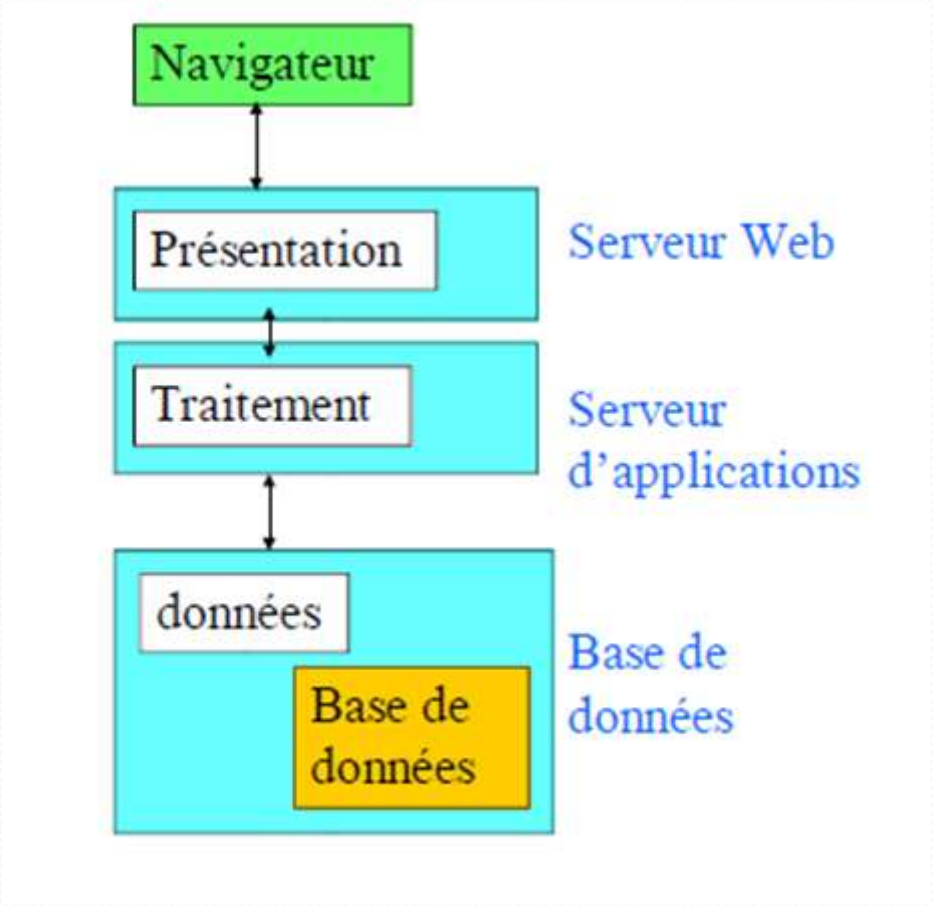
Architecure J2EE(2 tiers)



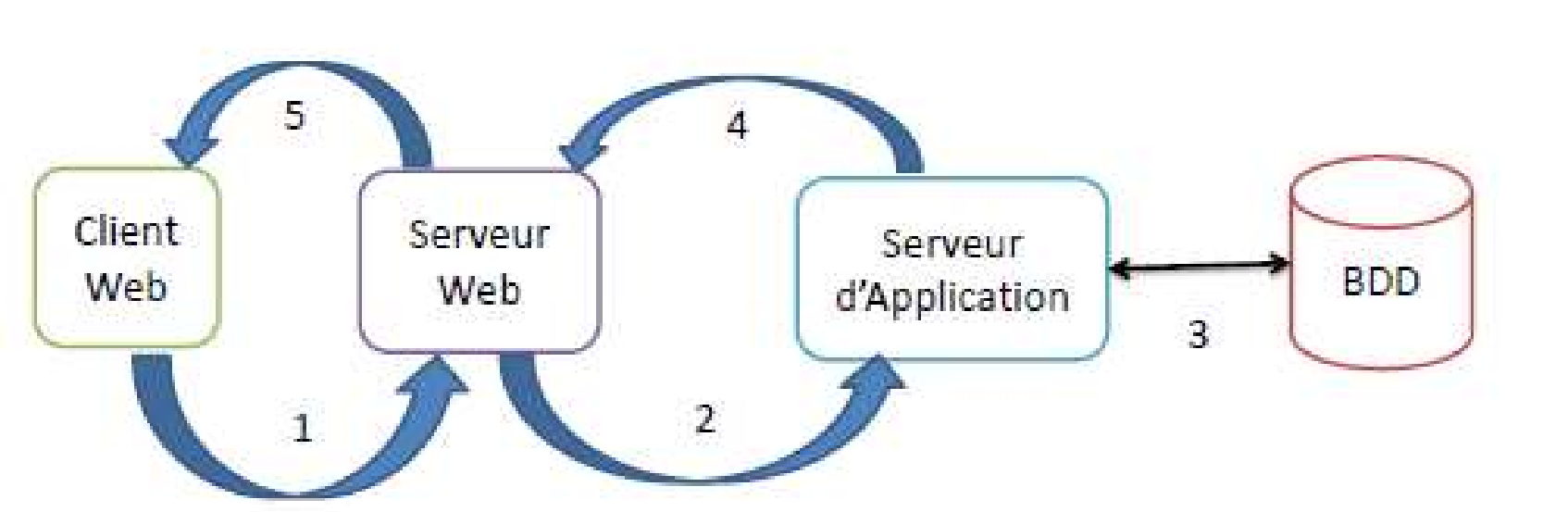
- Dans le cas d'une application JEE (ou J2E), l'architecture est quelque peu différente et utilise un **serveur d'applications**, situé entre le **serveur Web** et la **base de données**

Architecure JEE(3 tiers)

- JEE se base sur une architecture Client/Serveur à 3 tiers :



Principe de fonctionnement

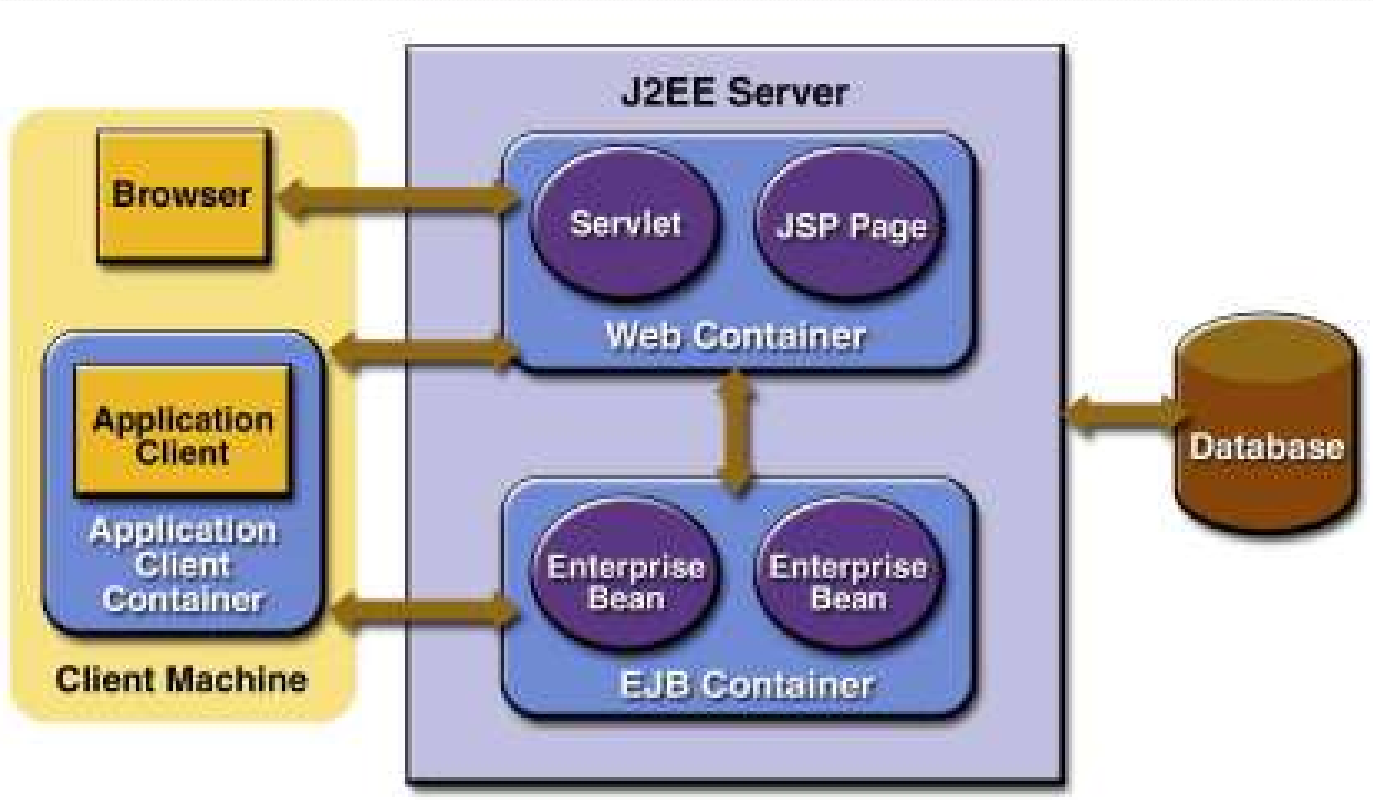


- Le client envoie une requête au serveur
- (2) Le serveur Web transmet la requête au serveur d'applications
- (3) Le serveur d'applications traite la requête en manipulant les données stockées dans la base de données
- (4) Le serveur d'applications envoie au serveur Web le résultat de la requête
- (5) Le serveur Web transmet le résultat au client Web, qui peut enfin l'afficher

Architecture JEE(3 tiers)

Le modèle MVC est implémenté dans JEE par les entités :

- La vue : les pages Web (HTML & JSP)
- Le contrôle : des servlets
- Le modèle : des classes java (des EJB et des JavaBean)



Conteneur JEE

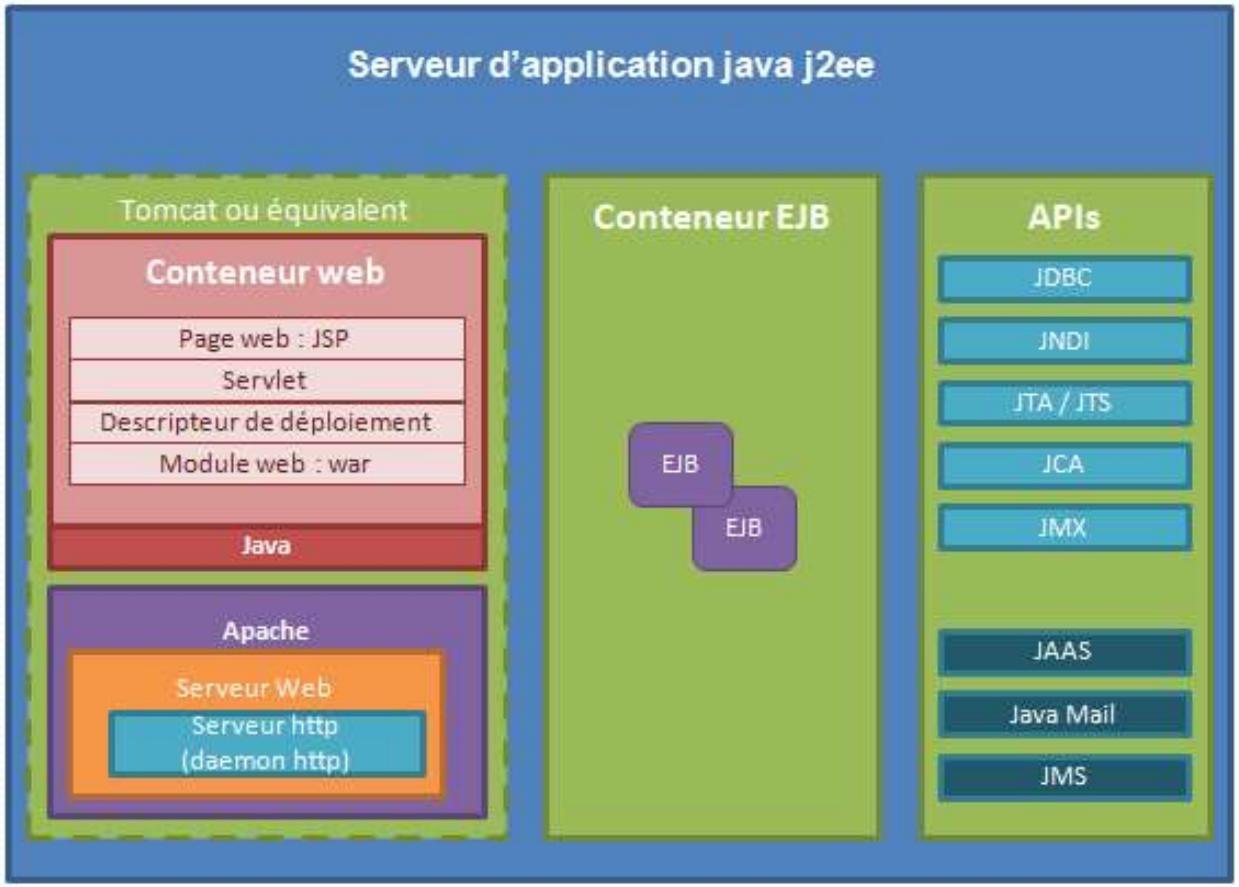
Un conteneur est un composant logiciel système qui contrôle d'autres composants, dits métier

Conteneur JEE : Environnement d'exécution Java permettant d'héberger des composants applicatifs et de contrôler leur exécution. Il existe deux types de container :

- Conteneur JEE Web : utilisés pour héberger des servlets ou des pages JSP
- Conteneur JEE EJB : supportant l'exécution des composants EJB

- Tomcat est un exemple de conteneur
- Il n'est autre qu'un serveur Apache couplé avec un moteur web java pour exécuter le code java
- Les servlets n'ont pas de méthode main(), ils sont contrôlés par le conteneur Tomcat
- Les requêtes ne sont pas adressées aux servlets mais au conteneur dans lequel ils sont déployés
- Tous les conteneurs de servlets doivent supporter le protocole HTTP et peuvent aussi supporter le protocole HTTPS.
- Un conteneur de JSP (JSP container) fournit les mêmes services qu'un conteneur de servlets.

Serveur d'application



- **Un conteneur EJB** qui encapsule les traitements des Entreprise JavaBeans.
- Un ensemble de services répartie en :
- **Des services d'infrastructures**
  - **JDBC** (Java DataBase Connectivity) API d'accès aux bases de données relationnelles.
  - **JNDI** (Java Naming and Directory Interface) API d'accès aux services de nommage et aux annuaires d'entreprises.
  - **JTA/JTS** (Java Transaction API/Java Transaction Services) API pour la gestion de transactions.
  - **JCA** (JEE Connector Architecture) API de connexion au système d'information de l'entreprise comme les ERP.
  - **JMX** (Java Management Extension) API permettant de développer des applications web de supervision d'applications
- **Des services de communication:**
  - **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 entre application.