# JAVA ET LE WEB

Licence Devops
Java
2019/2020

Fayçal BRAÏKI
etudiantspsp@free.fr

# Sommaire

I. Compléments sur les servlets

II. Les javabeans

III. Le *design pattern* Modèle Vue Contrôleur (MVC)

# Compléments sur les servlets

**Cycle de vie d'un servlet**

# Compléments sur les servlets

- Lors de l'initialisation, la méthode init() n'est appelée qu'une seule fois, à l'image d'un constructeur
  ➔Le servlet n'est donc chargé qu'une seule fois
- Cette méthode contient des éléments d'initialisation du servlet tel que des connexions base de données, des connexions réseau, …
- Elle peut aussi récupérer les paramètres d'initialisation et de contexte de l'application

# Compléments sur les servlets

- Chaque requête d'un client donne lieu à un appel à la méthode service()
- Elle assure le routage selon le type de requête (GET ou POST)
- La méthode destroy() assure la destruction de la servlet lorsqu'elle est déchargée par le moteur ou lorsque le serveur s'arrête
- Elle n'est appelée qu'une seule fois et contient, par exemple, la fermeture d'une connexion à une base de données

# Compléments sur les servlets

- Une servlet n'étant chargée qu'une seule fois, se pose la question de la gestion des accès concurrents aux ressources de la servlet
- Un thread est généré à chaque requête
- Les attributs de la servlet sont partagés et les méthodes accèdent à des ressources partagées
  ➔gestion des accès concurrents sur les sections critiques du code
- Plusieurs possibilités :
  - *Utiliser le mot clé « synchronized »*
  - *Implémenter l'interface SingleThreadModel (obsolète) qui restreint l'exécution d'une méthode par un seul thread à la fois*
    *➔potentielle perte de performance*
  - *L'exclusion mutuelle par sémaphore à l'aide de la classe Semaphore*

# Compléments sur les servlets

- HTML est un langage peu évolué or une application web a besoin de connaître divers éléments :
  - *utilisateur émettant la requête*
  - *Historique des requêtes*
  - *Nombre global de requêtes*
- Les servlets vont donc supporter ces fonctions à l'aide de divers mécanismes
  - *Les cookies*
  - *Utilisation des objets HttpSession*
  - *La réécriture d'url*
  - *…*

# Compléments sur les servlets

- La gestion des cookies
  - *Fichier texte comportant diverses informations relatives aux échanges entre le client et le serveur*
  - *Ils sont stockés sur le poste client et permettent de l'identifier*
  - *Le client a la possibilité de les désactiver dans le navigateur web*
  - *Structure d'un cookie :*
    - *Partie : Description*
    - Name : Identité du cookie
    - Value : Valeur du cookie
    - Domain : Nom du domaine qui l'a défini
    - Path : Information sur le chemin
    - MaxAge : Date d'expiration du cookie
    - Secure : Vrai si le cookie est transmis avec SSL

# Compléments sur les servlets

- La gestion des cookies
  - *Manipulation des cookies :*
    - Package javax.servlet.http.Cookie permettant de créer des objets cookie sur la base de paire nom/valeur
    - Public Cookie (String name, String valeur)
    - Les autres attributs sont accessibles :
    - void setMaxAge(int expSec):
      - *nombre de secondes depuis le 01/01/1970,*
      - *par défaut -1 (expire quand on ferme le client),*
      - *0 supprime le cookie*
    - void setPath(String path)
    - void setSecure(boolean flag)
    - void setDomain(String domain)

# Compléments sur les servlets

- La gestion des cookies

Exemple d'envoi de cookie :

```
protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {


    response.setContentType("text/html;charset=UTF-8");
     Cookie a = new Cookie("toto","20");
     response.addCookie(a);
```

# Compléments sur les servlets

- La gestion des cookies

Exemple de lecture de cookie :

```
Cookie [] tabcookie =request.getCookies();

        for (int i=0;i<tabcookie.length;i++)
//on pourra faire une recherche dans le tableau sur un critère nom, valeur, etc…
```

# Compléments sur les servlets

- Utilisation des objets HttpSession
  - *Une session représente l'ensemble des interactions (requêtes HTTP) entre un serveur et un client sur un temps donné*
  - *Les objets HttpSession mémorisent les sessions de chaque utilisateur*
  - *Chaque session est désignée par un ID de session donnant accès à une table contenant l'ensemble des données qui lui sont liées*
  - *Ces objets se trouvent sur le serveur et sont accessibles par la méthode request.getSession() (retourne la session, si elle existe, associée à la requête, en créer une sinon) qui retourne un objet de type HttpSession*
  - *Cet objet nous permet d'accéder aux informations de la session par des méthodes telles que getAttribute ou modifier la session par setAttribute*

# Compléments sur les servlets

- Exemple :

```
HttpSession session = request.getSession();
session.setAttribute("compteur", compteur);
Integer leCompteur = (Integer)session.getAttribute("compteur");
```

# Les javabeans

*Les javabeans sont des classes publiques qui obéissent à certaines règles :*

- *Ne doivent compter aucun attribut public*
- *Doivent avoir un constructeur sans argument (ou aucun constructeur)*
- *Tous les attributs doivent disposer d'accesseur et de modificateur (setXXXX, getXXXX, isXXXXX)*
- *Doivent être sérialisables*

*L'utilisation des beans dans une page jsp obéit à une syntaxe particulière : la balise <jsp:useBean..../>*

# Les javabeans

*<jsp:useBean*

*id=« nom du bean instancié »*

*class =« nom qualifié de la classe du bean instancié »*
*scope =« request|session|page|application »*

*/>*

Attribut *scope :*

- *page (mode par défaut): idem request sauf que le Bean ne sort pas de la page*
- *request: le Bean est valide pour la requête, il est détruit à la fin*
- *session: il est stocké dans la session de l'utilisateur et il dure tant que la session n'est pas terminée*
- *application: le Bean est valide pour l'application courante (créé une fois et partagé par tous les clients de l'application)*

# Les javabeans

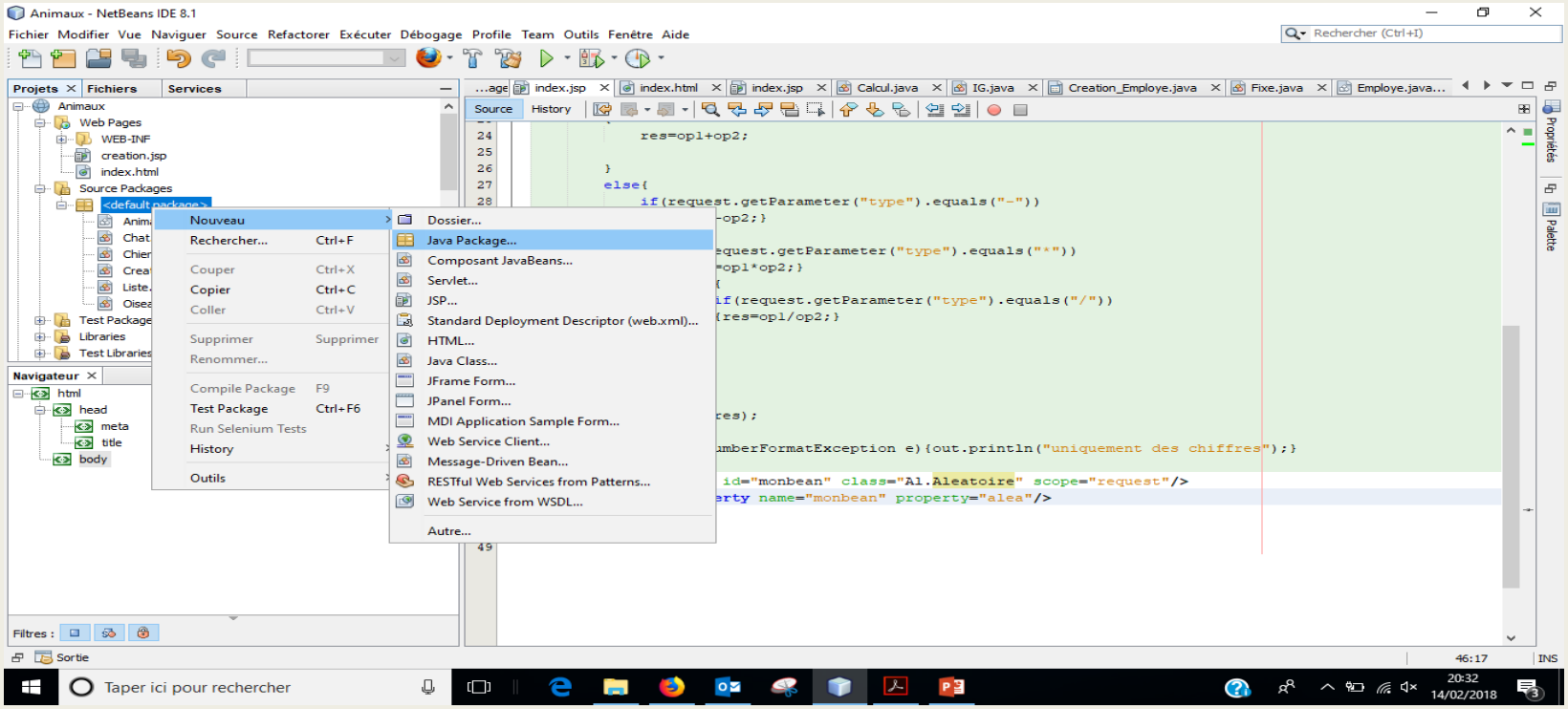
*Exemples d'utilisation des beans :*

- *Génération d'un nombre aléatoire*
- *Affichage et conservation d'un nom de client le long d'une session*
- *Affichage et conservation d'une donnée partagée par plusieurs clients*

# Les javabeans

*Exemples d'utilisation des beans :*

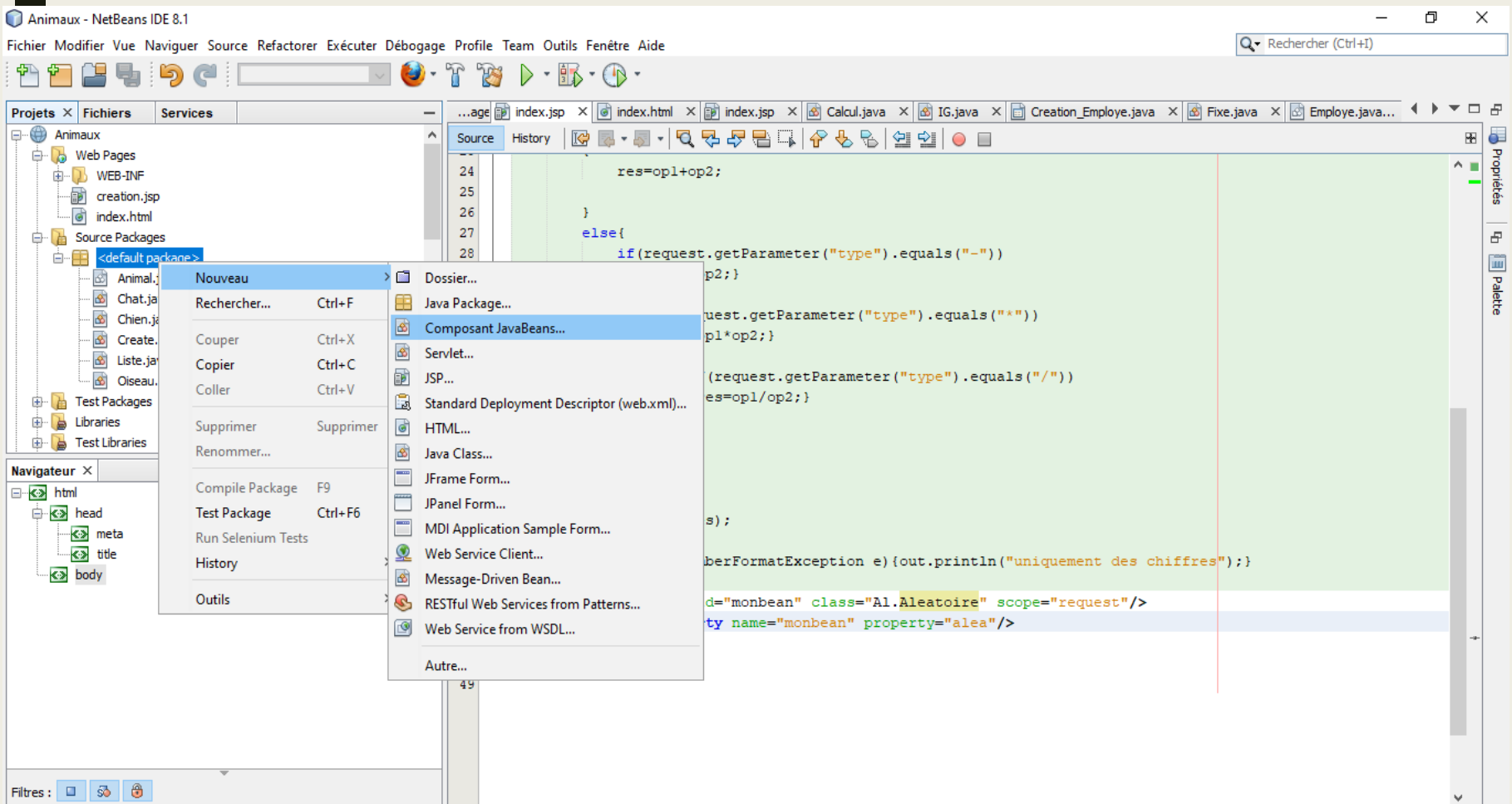
- *Génération d'un nombre aléatoire :*
- *Il faut au préalable créer dans l'application un package dans lequel le javabean sera inséré puis créer la classe sous la forme d'un javabean*



# Les javabeans

*Exemples d'utilisation des beans :*

- *Génération d'un nombre aléatoire :*



# Les javabeans

*Exemples d'utilisation des beans :*

- *Génération d'un nombre aléatoire :*
- *Le principe de fonctionnement est le suivant :*
  - Le bean sera créé dans la servlet
  - Il est ensuite redirigé dans la requête et dans la jsp
  - La page jsp se chargera d'afficher le nombre aléatoire généré par le bean

# Les javabeans

*Exemples d'utilisation des beans :*

- *Génération d'un nombre aléatoire :*

```
package Al;
import java.beans.*;
import java.io.Serializable;
public class Aleatoire implements Serializable {//définition d'un bean
    private  double alea;
    public Aleatoire() {
       alea=Math.random();
    }
    public double getAlea() {
        return alea;
    }
    public void setAlea(double value) {
        alea = value;
    }
}
```

# Les javabeans

*Exemples d'utilisation des beans :*

- *Génération d'un nombre aléatoire :*

```
//dans la servlet
 protected void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {
...
Aleatoire a =new Aleatoire();//création d'un bean
request.setAttribute("monbean", a);//affectation du bean à un attribut de la requête
request.getRequestDispatcher("index.jsp").forward(request, response);//redirection de la requête dans ma jsp
...
}
```

# Les javabeans

*Exemples d'utilisation des beans :*

- *Génération d'un nombre aléatoire :*

*//dans la jsp*

*<jsp:useBean id="monbean" class="Al.Aleatoire" scope="request"/>//mon bean est de type request*

*<jsp:getProperty name="monbean" property="alea"/> //l'invocation de la propriété alea de mon bean provoquera l'utilisation de l'accesseur s'y référant*

# Les javabeans

*Exemples d'utilisation des beans :*

- *Affichage et conservation d'un nom de client le long d'une session*

```
//dans la servlet
TestBean test = new TestBean(...);
HttpSession session = request.getSession();
session.setAttribute("monBean", test);
request.getRequestDispatcher ("Affiche.jsp").forward(request, response);

//dans la jsp
<jsp:useBean id="monBean" type=« client.TestBean » scope="session" />
<jsp:getProperty name="monBean" property=« nom » />
```

# Les javabeans

*Exemples d'utilisation des beans :*

- *Affichage et conservation d'une donnée partagée par plusieurs clients*

```
//dans la servlet
synchronized (this) {//pour la gestion des accès concurrents

TestBean test = new TestBean(…);

getServletContext.setAttribute("monBean", test);

RequestDispatcher dispatcher = request.getRequestDispatcher ("AfficheJSP.jsp");

dispatcher.forward(request, response);

}

//dans la jsp
<jsp:useBean id="monBean" type=« client.TestBean »  scope="application" />

<jsp:getProperty name="monBean" property=« nbclients »/>
```

# Les javabeans

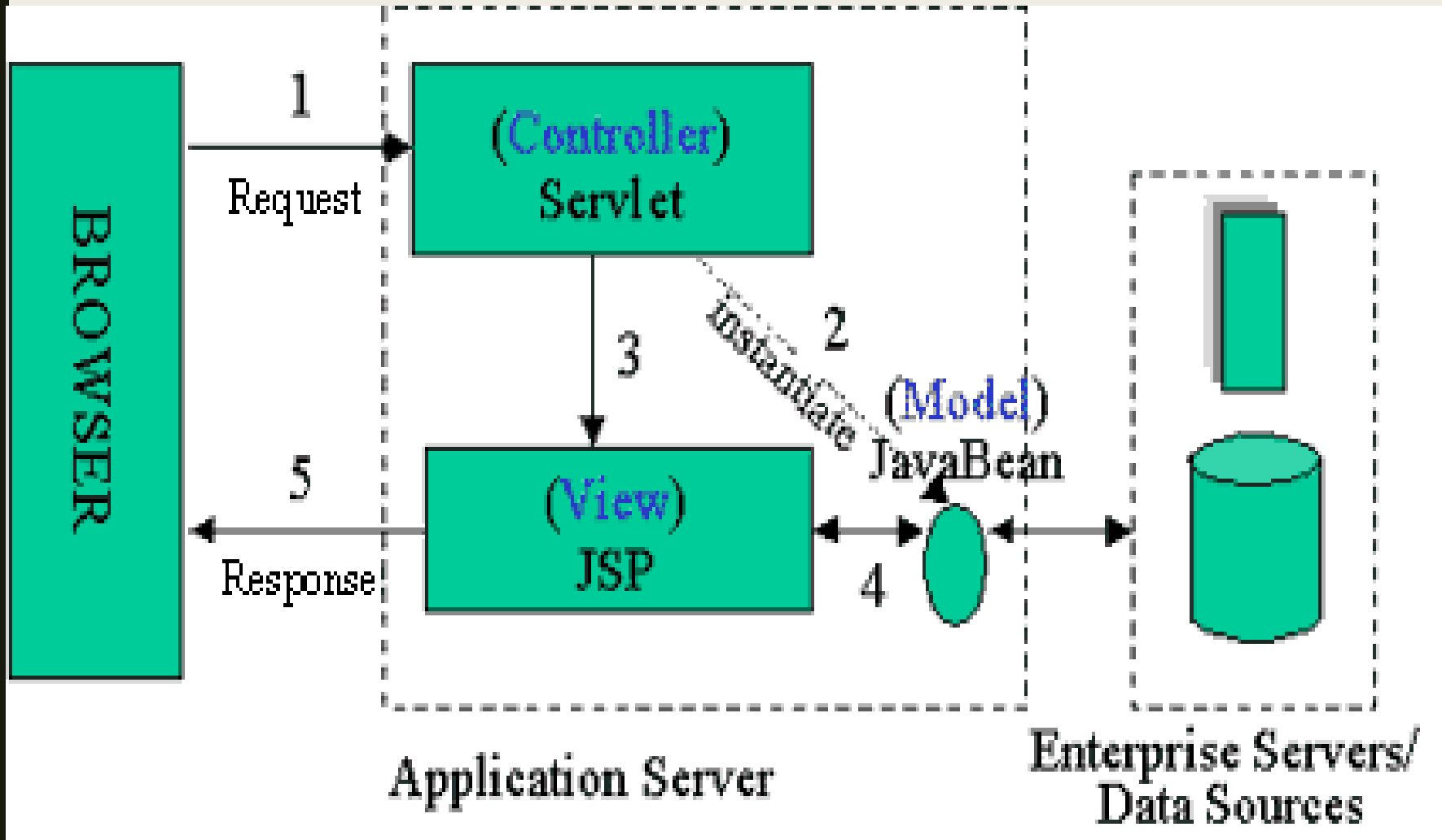
- *Un bean est aussi paramétrable à l'aide de la balise <jsp:setProperty … />*
- *et selon la syntaxe suivante :*
  *<jsp:useBean id="monBean" class=" client.TestBean " scope="request"/>*
  *<jsp:setProperty name="monBean" property=« nom » value=« Jojo »/>*

- *Bien évidemment cela suppose l'existence d'un modificateur dans le bean pour l'attribut que l'on souhaite modifier*

# Le *design pattern* Modèle Vue Contrôleur (MVC)

- *MVC (Rappel) :*
  - Modèle de conception (design pattern) des interfaces graphiques introduit à la fin des années 70
  - Le modèle est la partie relative aux données traitées par l'application
  - La vue est partie relative aux interactions avec l'utilisateur
  - Le contrôleur est la partie relative à la gestion de l'ensemble de l'application (M et V)
- *Dans une application web java, l'utilisation des servlets, des jsp et des javabeans permettent d'implémenter MVC en respectant toutefois certaines règles :*
  - Les servlets jouent le rôle de contrôleur
  - Les jsp jouent le rôle de vue
  - Les beans permettent l'accès au modèle

# Le *design pattern* Modèle Vue Contrôleur (MVC)

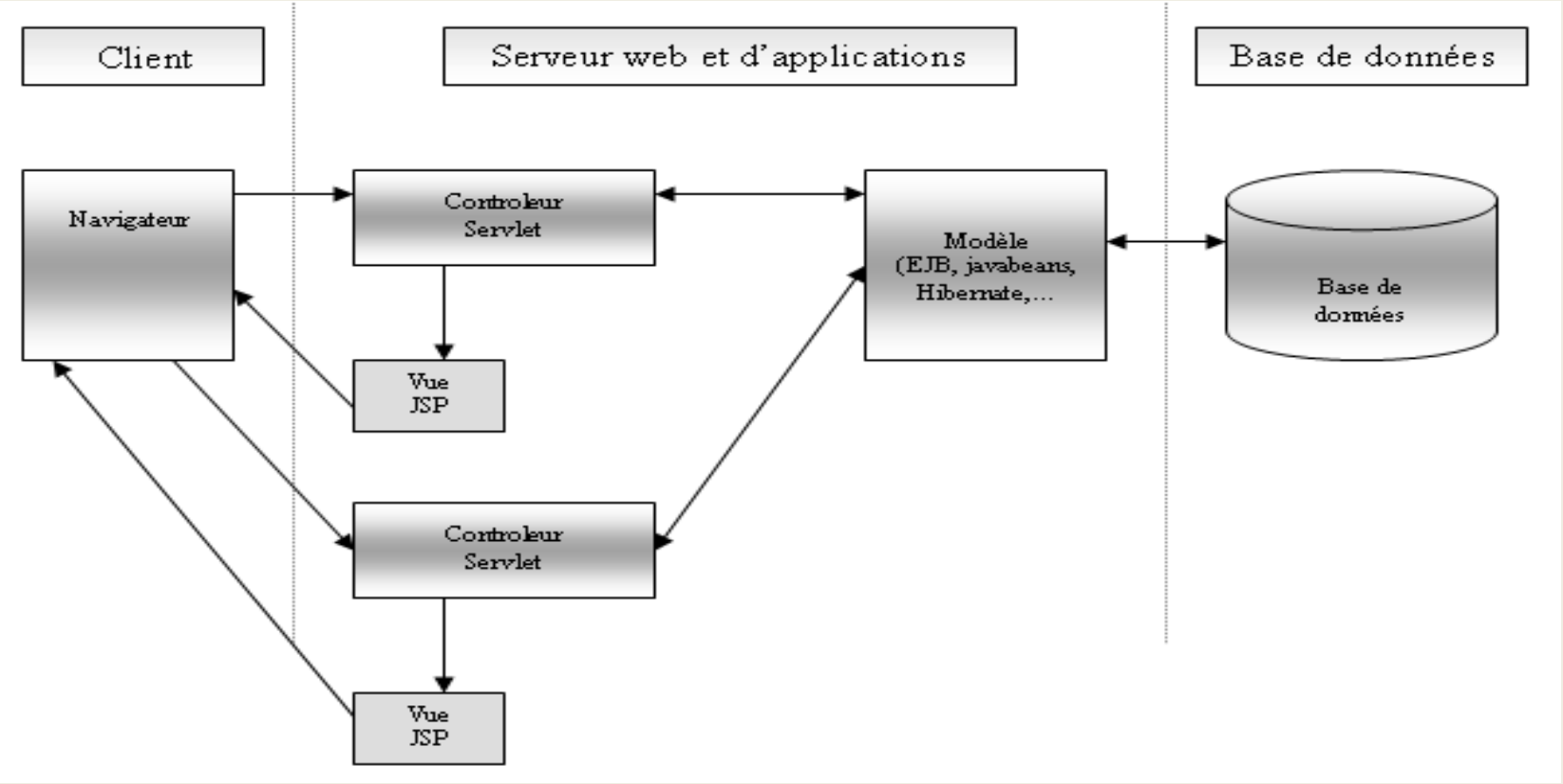


# Le *design pattern* Modèle Vue Contrôleur (MVC)

- *Bonne pratique :*
  - Créer un package pour chacun les rôles modèle  (voire pour les beans) et contrôleur

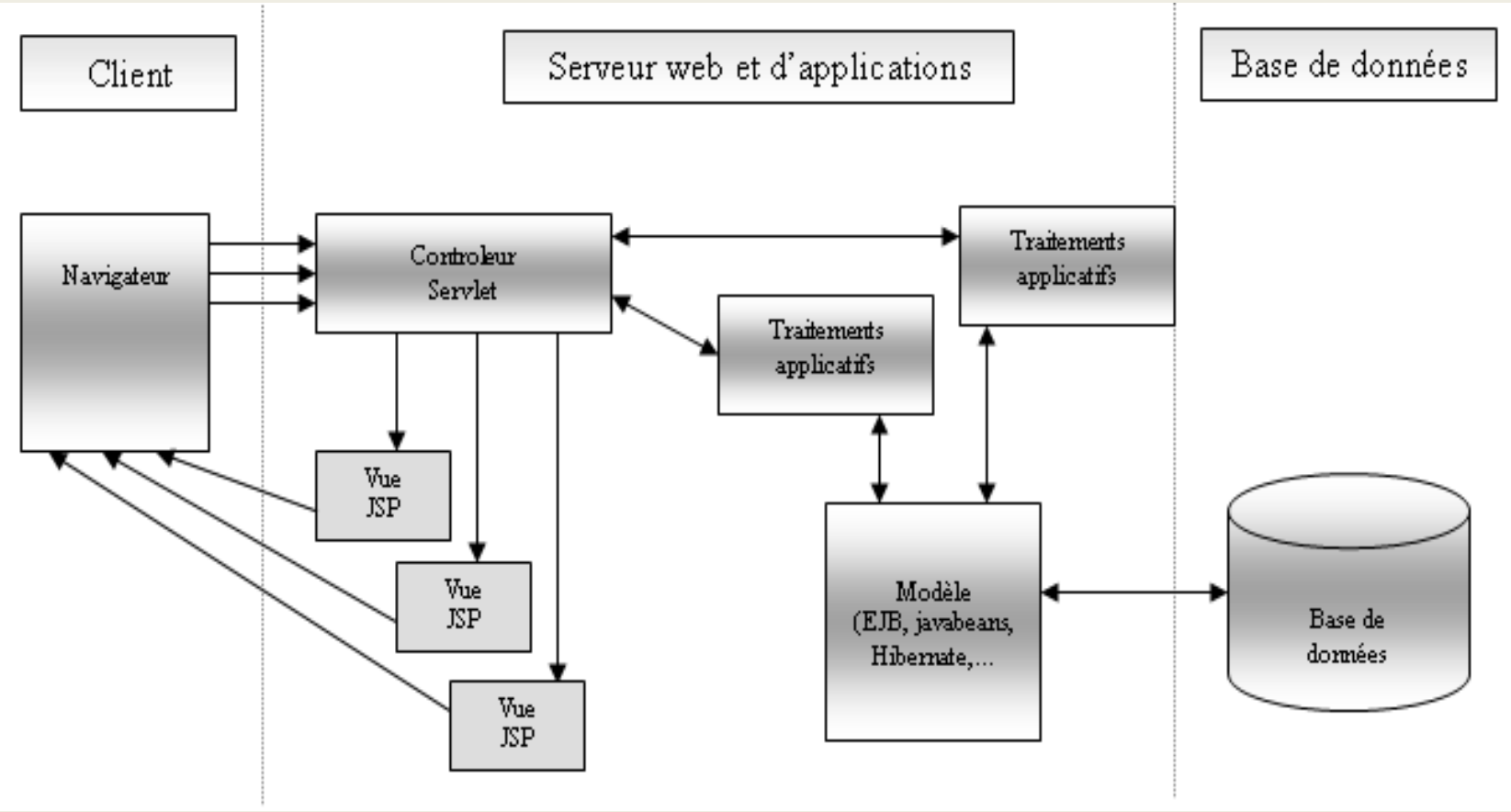
# Le *design pattern* Modèle Vue Contrôleur (MVC)

- *MVC 1 : plusieurs servlets de contrôle possibles*



# Le *design pattern* Modèle Vue Contrôleur (MVC)

- *MVC 2 : une seule servlet de contrôle*



# Le *design pattern* Modèle Vue Contrôleur (MVC)

- *Les outils d'aide à la conception MVC*
  - Communément appelés « FrameWorks »
  - Fournissent des éléments pré-construits, des squelettes d'applications conformes aux recommandations MVC
  - Leur usage n'est pas forcément systématique et est principalement conditionné par la nature et la complexité du projet ainsi que par la capacité des équipes de développeurs à les utiliser
  - Il en existe de 2 types (requête ou composant)
  - Les + connus : Spring MVC, Struts, Tapestry, JSF, etc…

# Le *design pattern* Modèle Vue Contrôleur (MVC)

- *Exercice :*
  *reprendre l'application Ménagerie pour la rendre conforme à MVC 2 en utilisant un ou des beans*