Introduction JAVA EE

QUE VEUT DIRE JAVA EE ?

- Java EE reprend tout ces concepts de la programmation réseaux en rajoutant la sécurité.

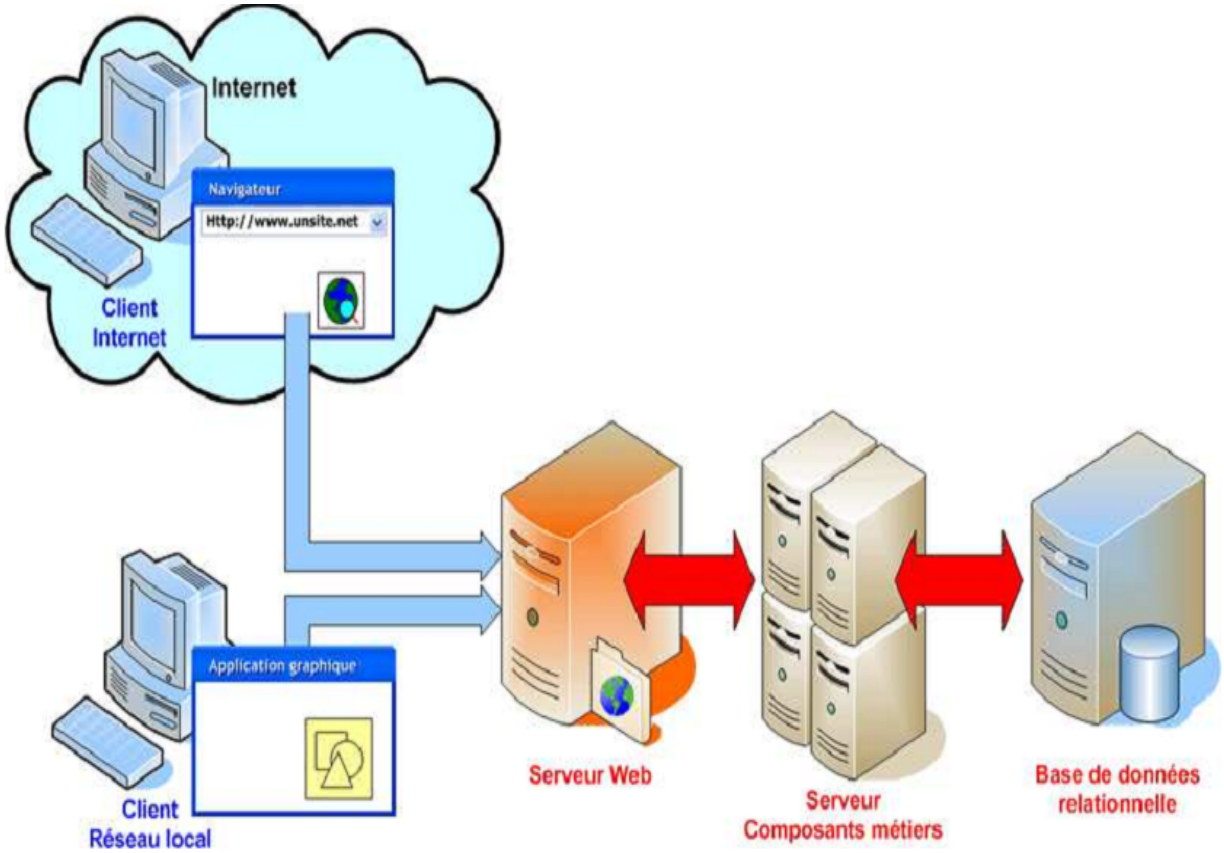
- Java EE est une architecture aboutie et performante, elle a été mise en œuvre pour le monde des entreprises qui ont besoin d'un système stable qui accepte facilement la montée en charge sans poser de problème de sécurité.

- Java EE exploite au maximum la technologie des objets distribués, ce qui permet d'avoir un système simple à utiliser côté clients.

QUE VEUT DIRE JAVA EE ?

- Signifie Java Entreprise Edition et représente essentiellement des applications d'entreprise.
- Inclut le stockage sécurisé des informations, ainsi que leur manipulation et leur traitement : factures clients, calculs d'amortissement, réservation de vols, etc.
- Ces applications peuvent avoir des interfaces utilisateurs multiples, par exemple une interface Web pour les clients, accessible sur Internet et une interface graphique fonctionnant sur les ordinateurs de l'entreprise sur le réseau privé de celle-ci.

EXEMPLE 1


Internet
Navigateur
Http://www.unsite.net
Client
Internet
Application graphique
Client
Réseau local
Serveur Web
Serveur
Composants métiers
Base de données
relationnelle

SERVEURS D'APPLICATIONS

- Tout comme les bibliothèques d'interfaces graphiques comme Swing fournissent les services nécessaires au développement d'application graphiques.
- Les serveurs d'applications mettent à disposition les fonctionnalités permettant de réaliser des applications d'entreprise
- Exemples : communication entre ordinateurs,
  mis en place de protocole adaptés,
  gestion des connexions avec une DB,
  présentation de pages Web,
  gestion des transactions, etc

SERVEURS D'APPLICATIONS

- Java EE propose un ensemble de bibliothèques avec des objets de très haut niveau pour mettre en oeuvre facilement ses serveurs.

- les développeurs n'ont pas à partir d'une feuille blanche et surtout Java EE permet d'avoir une démarche standardisée.

QU'EST-CE QUE JAVA EE ?

- Java EE est souvent synonyme de Entreprise JavaBeans pour de nombreux développeurs.
- Mais, Java EE est beaucoup plus que cela.
- Java EE est une collection de composants, de conteneurs et de services permettant de créer et de déployer des applications distribuées au sein d'une architecture standardisée.

QU'EST-CE QUE JAVA EE ?

- Java EE est logiquement destiné aux gros systèmes d'entreprise.
- les applications doivent être constituées de plusieurs composants pouvant être déployés sur des plate-formes multiples afin de disposer de la puissance de calcul nécessaire.
- C'est la raison d'être des applications distribuées.
- Java EE fournit un ensemble de composants standardisés facilitant le déploiement des applications, des interfaces définissant la façon dont les modules logiciels peuvent être interconnectés, et les services standards, avec leur protocole associé, grâce auxquels ces modules peuvent communiquer.

ARCHITECTURE MULTITIERS

- Un des thèmes du développement d'applications Java EE est la décomposition de celles-ci en plusieurs niveaux ou tiers.

- Généralement, une application d'entreprise est composée de trois couches fondamentales (d'où le terme décomposition en trois tiers) :

ARCHITECTURE MULTITIERS

La première : présentation

- a pour rôle d'afficher les données pour l'utilisateur et de collecter les informations qu'il saisit.
- Cette interface est souvent appelée couche de présentation car sa fonction consiste à présenter les données à l'utilisateur et à lui permettre de fournir des informations au système.
- Est la partie de l'application responsable de la création et du contrôle de l'interface présentée à l'utilisateur et de la validation de ses actions.

ARCHITECTURE MULTITIERS

La deuxieme : la logique métier

- permet à l'application de fonctionner et de traiter les données.
- La logique métier est mise en oeuvre à partir des règles métier.

- Exemple : Dans une application de paye, par exemple, la logique métier multiplie les heures travaillées par le salaire horaire pour déterminer combien chaque employé doit toucher.

ARCHITECTURE MULTITIERS

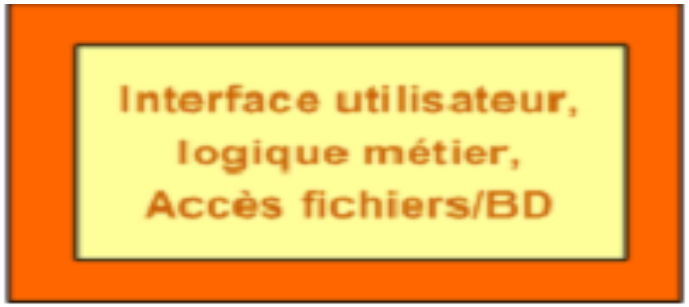
La troisieme : Persistance

- Toutes les applications d'entreprise ont besoin d'écrire et de lire des données.

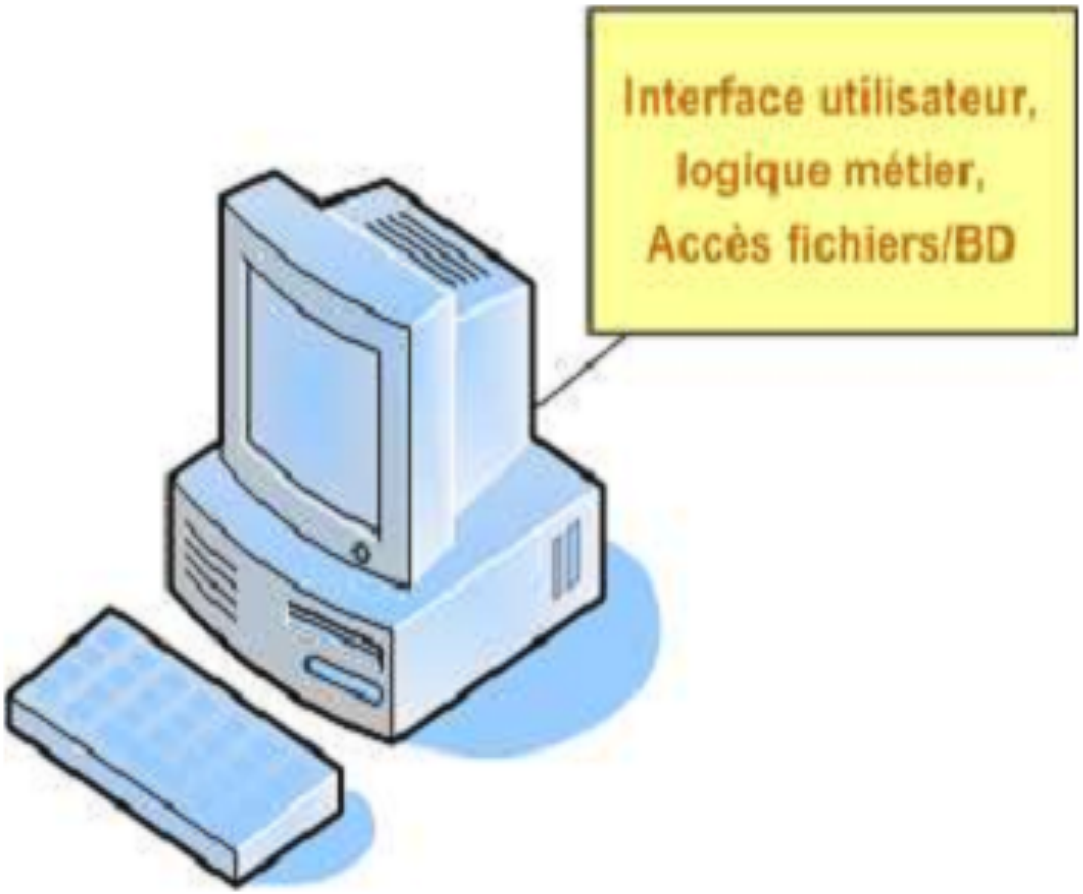
- Cette fonctionnalité est assurée par la couche d'accès de données, également appelée couche de persistance, qui assure la lecture, l'écriture à partir des différentes sources.

ARCHITECTURE MULTITIERS

Architecture 1 tier

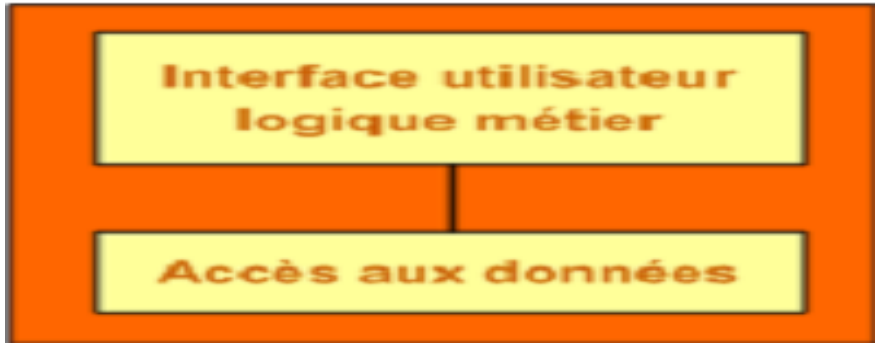


- Les applications bureautiques sont conçues pour fonctionner sur un ordinateur unique.
- Cette architecture monolitique est appelée simple tiers car toutes les fonctionnalités sont comprises dans une seule couche logicielle.

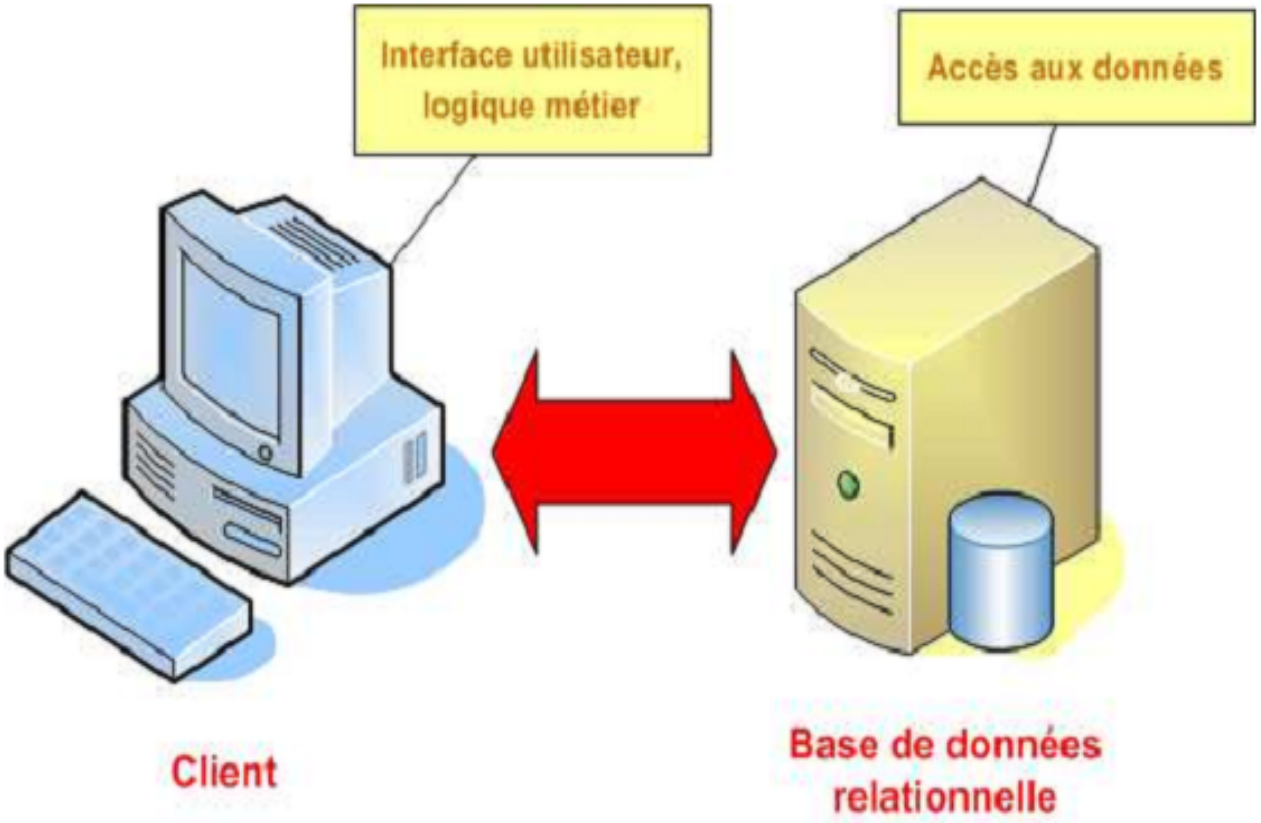

Interface utilisateur,
logique métier,
Accès fichiers/BD

ARCHITECTURE MULTITIERS

Architecture 2 tiers



- La raison de cette approche est de centraliser les données afin de permettre à plusieurs utilisateurs d'y accéder simultanément.
- Les données peuvent ainsi être partagées entre plusieurs utilisateurs de l'application.
- Cette architecture est appelée client-serveur, qui dans notre approche peut être représentée en deux tiers.


Interface utilisateur,
logique métier
Accès aux données
Client
Base de données
relationnelle

ARCHITECTURE MULTITIERS

Inconvénient Architecture 2 tiers

- la logique chargée de la manipulation des données et de l'application des règles métiers afférentes est incluse dans l'application elle-même.
- Un Contre Exemple :

lorsque plusieurs applications doivent partager l'accès à une base de données. Il peut y avoir, par exemple, une règle stipulant qu'un client affichant un retard de paiement de plus de 90 jours verra son compte suspendu. Il n'est pas compliqué d'implémenter cette règle dans chaque application accédant aux données client.

Toutefois, si la règle change et qu'un délai de 60 jours est appliqué, il faudra mettre à jour toutes les applications, ce qui peut être contraignant.

ARCHITECTURE MULTITIERS

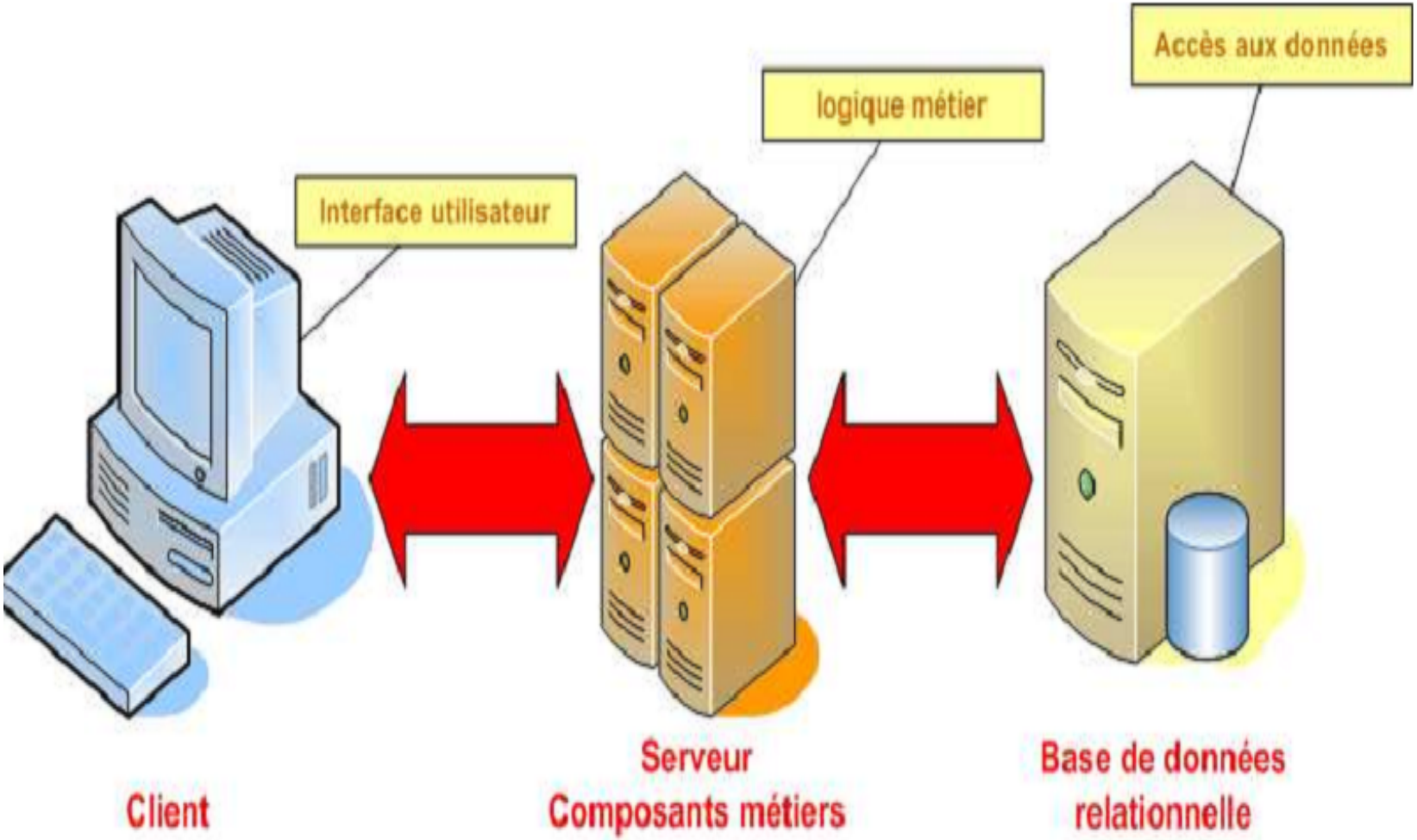
Inconvenient Architecture 2 tiers

- Pour éviter ce genre de problème
- la solution consiste à séparer physiquement les règles métier en les plaçant sur un serveur où elles n'auront à être remise à jour qu'une seule fois
- Cette solution ajoute un troisième tiers à l'architecture client-serveur

ARCHITECTURE MULTITIERS

Architecture 3 tiers

- la logique métier est extraite de l'applicationcliente.
- Elle n'est plus responsable que de la présentation de l'interface à l'utilisateur et de la communication avec le tiers médian.
- Son rôle est réduit à la couche présentation.


Accès aux données
logique métier
Interface utilisateur
Client
Serveur
Composants métiers
Base de données
relationnelle

CONCEPTS ET SPÉCIFICITÉS DE J2EE

Côté client

- peut être une application console (texte seulement) écrite en Java.
- une application dotée d'une interface graphique développée en Swing.

en raison de la quantité importante de code, ce client est appelé client lourd.

- peut également être conçu pour être utilisé à partir du Web.
- Ce type de client fonctionne à l'intérieur d'un navigateur Web.

Vu que la plus grande partie du travail est reportée sur le serveur le client est léger.

CONCEPTS ET SPÉCIFICITÉS DE J2EE

Côté Serveur

- Les composants déployés sur le serveur peuvent être classés en deux groupes.
- **Les composants Web sont réalisés à l'aide de servlets ou de JavaServer Pages (JSP)**
- **Les composants métiers, dans le contexte Java EE, sont des Entreprise JavaBeans (EJB).**

LES CONTENEURS

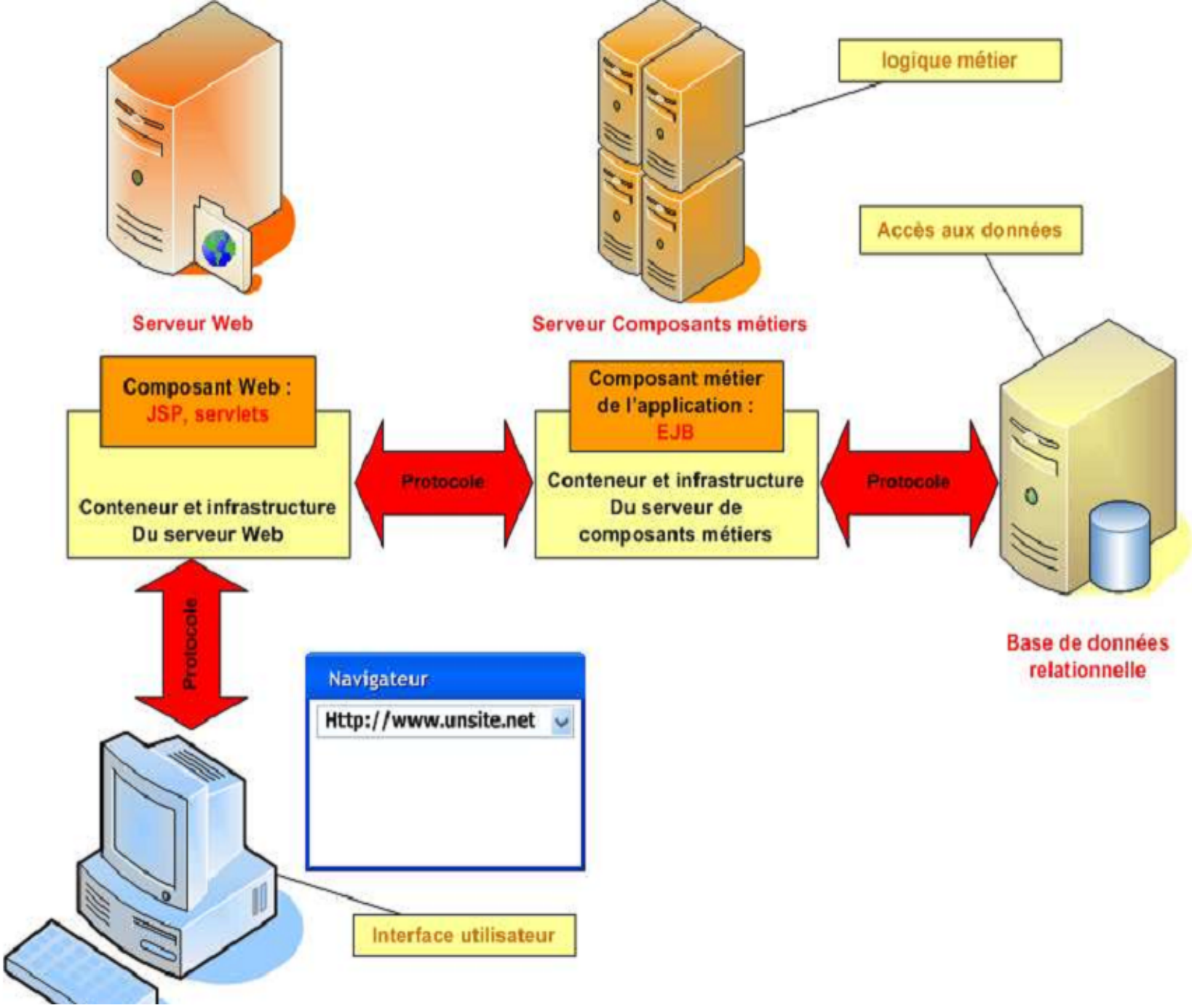
- Les conteneurs sont les éléments fondamentaux de l'architecture Java EE
- fournissent :

une interface parfaitement définie, un ensemble de services permettant aux développeurs d'applications de se concentrer sur la logique métier. sans qu'ils aient à se préoccuper de toute l'infrastructure interne.

LES CONTENEURS

Les conteneurs s'occupent :

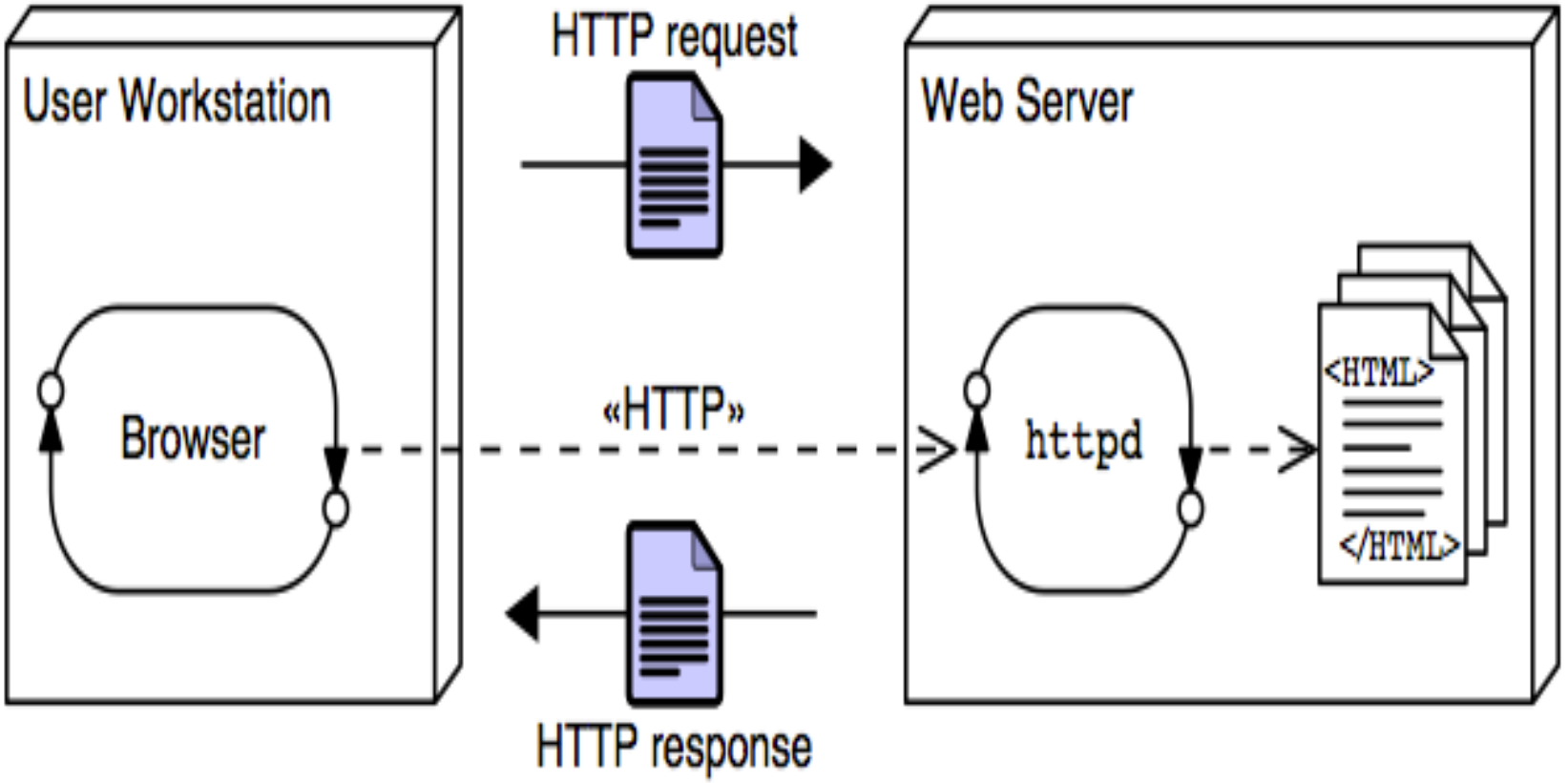
- toutes les tâches fastidieuses liées au démarrage des services sur le serveur
- l'activation de la logique applicative.
- **la gestion des protocoles de communication.**
- **la libération des ressources utilisées.**


logique métier
Accès aux données
Serveur Web
Serveur Composants métiers
Composant Web :
JSP, servlets
Conteneur et infrastructure
Du serveur Web
Protocole
Composant métier
de l'application :
EJB
Conteneur et infrastructure
Du serveur de
composants métiers
Protocole
Base de données
relationnelle
Protocole
Navigateur
Http://www.unsite.net
Interface utilisateur

LES SERVLETS JAVA

Statiques :

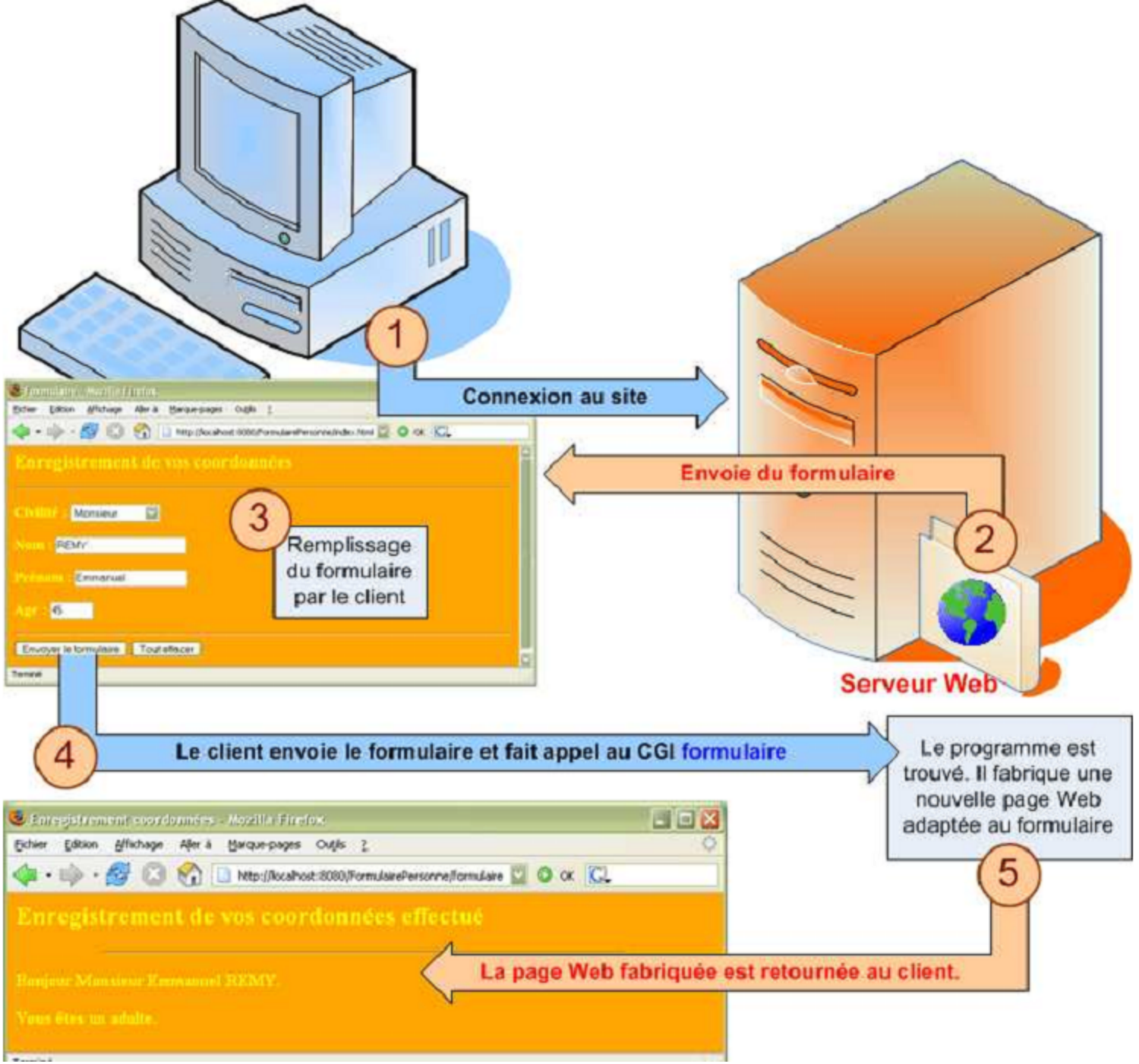
- Vous avez sans doute l'habitude d'accéder à des pages HTML statiques à l'aide d'un navigateur envoyant une requête à un serveur Web
- Ce serveur renvoi cette page qui est stockée sur son disque.
- **le serveur ici : joue le rôle d'un bibliothécaire virtuel qui renvoie le document demandé.**


User Workstation
HTTP request
Web Server
Browser
«HTTP»
httpd
<HTML>
</HTML>
HTTP response

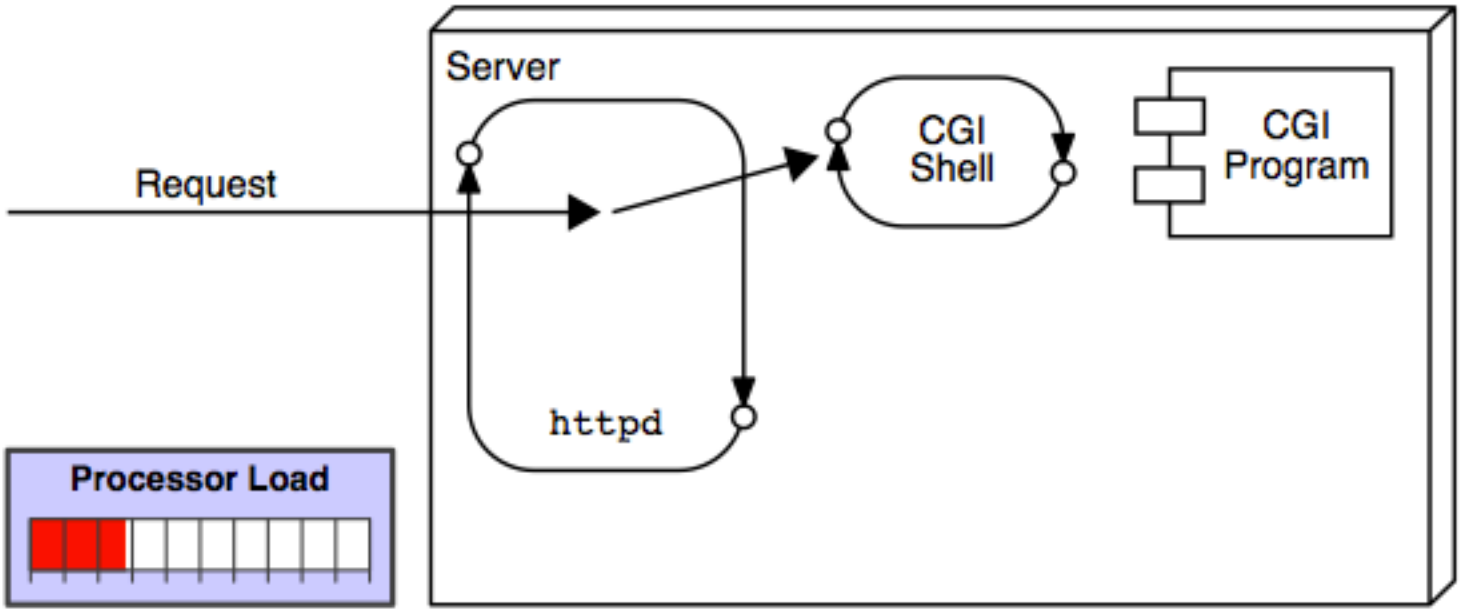
LES SERVLETS JAVA

Dynamiques :

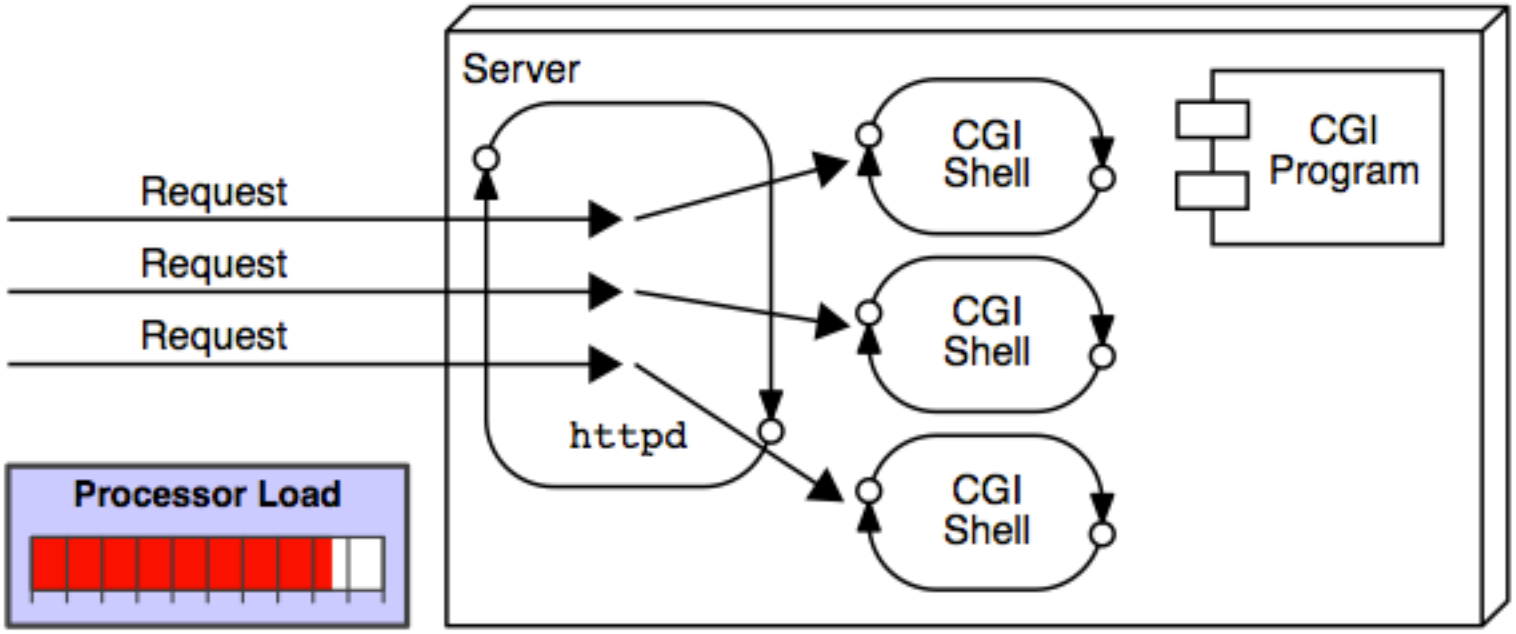
- **le modèle précèdent ne permet pas d'accéder à des pages dynamiques, dont le contenu serait créé à la demande.**
- Supposons par exemple que le client souhaite obtenir une liste des documents HTML correspondant à certains critères
- **Dans ce cas, il est nécessaire de créer une page HTML différente en fonction des critères spécifiés par le client.**


1
Connexion au site
Envoie du formulaire
2
Serveur Web
3
Remplissage du formulaire par le client
4
Le client envoie le formulaire et fait appel au CGI formulaire
Le programme est trouvé. Il fabrique une nouvelle page Web adaptée au formulaire
5
La page Web fabriquée est retournée au client.
Enregistrement de vos coordonnées effectué

LES SERVLETS JAVA vs CGI



LES SERVLETS JAVA vs CGI



LES SERVLETS JAVA vs CGI

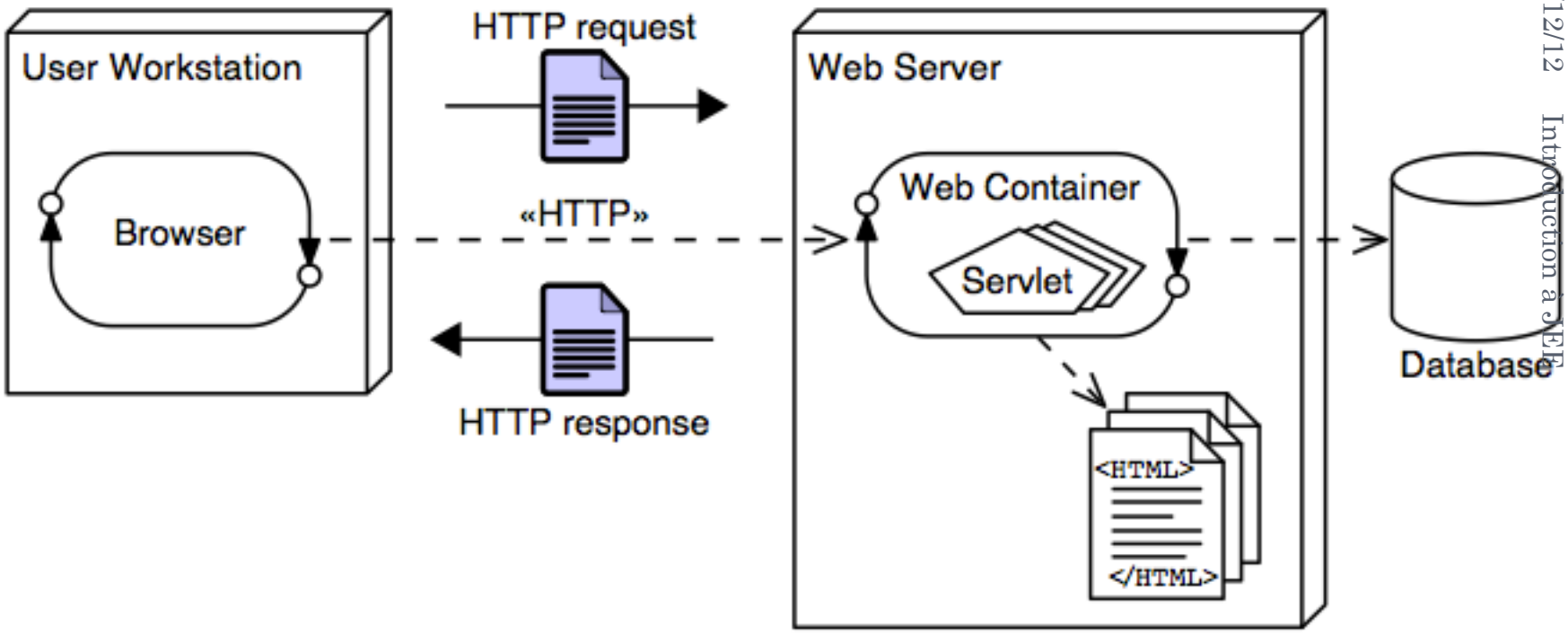
Avantages CGI

- Écrit par plusieurs langages de programmation.
- Relativement facile pour un WEB designer.

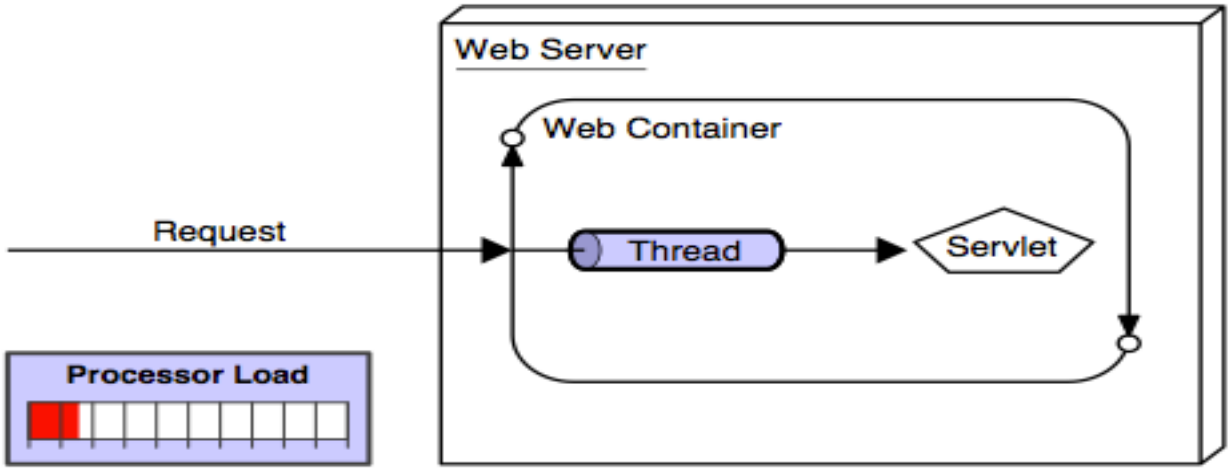
Inconvénients

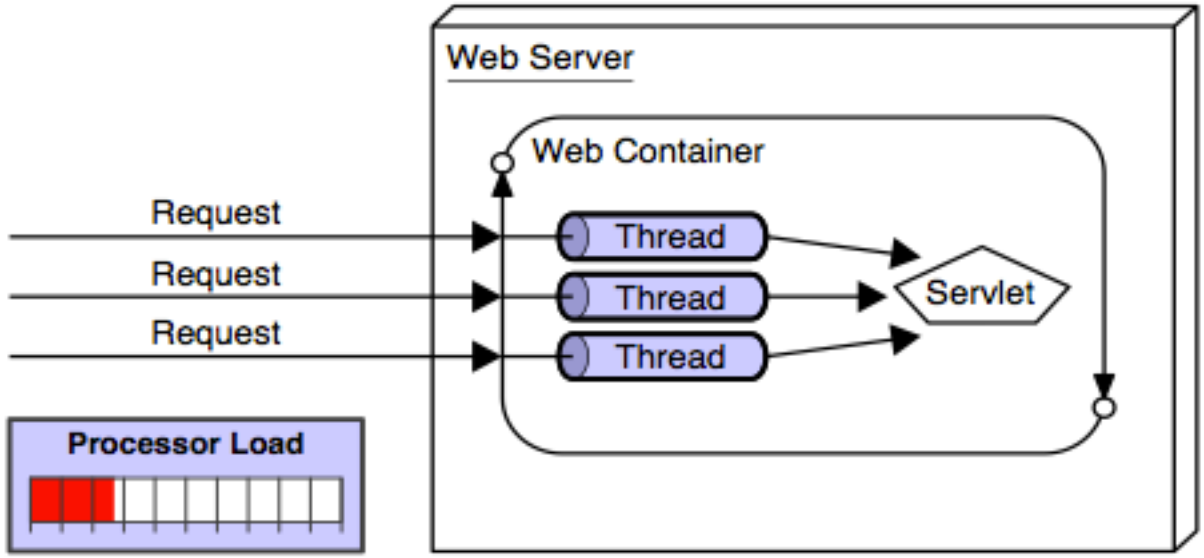
- Chaque Shell est trop lourde.
- Logique métier est mélangée avec HTML (présentation).
- Le langage n'est pas tjrs sécurisé ou OO.
- Le langage n'est pas tjrs de plateforme indépendante.

LES SERVLETS JAVA



LES SERVLETS JAVA





LES SERVLETS JAVA

- Une servlet est un composant implémentant l'interface javax.servlet.Servlet.
- Son invocation est la conséquence de la requête du client, dirigé vers cette servlet
- Le serveur Web reçoit une demande adressée à une servlet sous la forme d'une requête HTTP
- Il transmet la requête à la servlet concernée, puis renvoie la réponse fournie par celle du client .
- La servlet construit la réponse et renvoie sous forme de code HTML.

LES SERVLETS JAVA

[Grasp 3] * - jGRASP CSD (Java)

```
import javax.servlet.*;
import javax.servlet.http.*;
import java.io.*;
import java.util.*;
public class Simple extends HttpServlet {
  //Traiter la requête HTTP Get
  public void doGet(HttpServletRequest request, HttpServletResponse response) throws ServletEx
    response.setContentType("text/html");  // type MIME pour l'en-tête http --> Page HTML
    PrintWriter out = response.getWriter();
    out.println("<html>");
    out.println("<head><title>Enregistrement coordonnées</title></head>");
    out.println("<body bgcolor=orange text=yellow>");
    out.println("<h2>Enregistrement de vos coordonnées effectué</h2>");
    out.println("<hr width=75%>");
    out.print("<p><b>Bonjour "+ request.getParameter("civilite")+" ");
    out.print(request.getParameter("prenom")+" ");
    out.println(request.getParameter("nom")+".");
    out.println("</p></b></body></html>");
  }
  //Traiter la requête HTTP post
  public void doPost(HttpServletRequest request, HttpServletResponse response) throws ServletE
      doGet(request, response);
  }
}
```

LES JAVA SERVER PAGES (JSP)

- PHP (PHP Hypertext Preprocessor)

```
<? for ( $i=0; $i<10; $i++ ) { ?>
  <TR><TD><? echo $i ?></TD><TD><? echo ($i * $i) ?></TD></TR>
<? } ?>
```

- ASP (Active Server Pages)

```
<% FOR I = 0 TO 10 %>
  <TR><TD><%= I %></TD><TD><%= (I * I) %></TD></TR>
<% NEXT %>
```

- JSP (JavaServer Pages)

```
<% for ( int i=0; i<10; i++ ) { %>
  <TR><TD><%= i %></TD><TD><%= (i * i) %></TD></TR>
<% } %>
```

LES JAVA SERVER PAGES (JSP)

- Les JavaServer Pages, ou JSP, servent, comme les servlets, à créer du contenu Web de manière dynamique.
- Ces deux types de composants représentent à eux seuls un très fort pourcentage du contenu des applications Web.
- Les JSP sont des documents de type texte, contenant du code HTML ainsi que des scriptlets (et/ou des expressions), c'est-à-dire des morceaux de code Java.
- Les développeur des pages JSP peuvent mélanger du contenu statique et du contenu dynamique

LES JAVA SERVER PAGES (JSP)

```
<HTML>
<HEAD>
<TITLE>La date par Jsp</TITLE>
</HEAD>
</BODY>
<font size="+3">
<p> Démonstration de la génération de page dynamique </p>
<%@ page language="java" import = "mesdates.*" %>
<font size=6 color="red">
  <%= ladate.getMessage() %>
   <br>
  <%= ladate.getDate() %>
 </font> <br>
    <%!
      String [] semaine = {"dimanche","lundi", "mardi", "mercredi", "jeudi", "vendredi",  "samedi"};
    %>
<%
int jour = ladate.getJour() - 1 ;
out.println (semaine[jour]);
if (jour != 1) out.println (" , on travaille aujourd'hui");
else out.println (" , c'est férie aujourd'hui");
%>
<font>
<form method="get">
changer le message : </br>
    <input type="text" name="message" size="50">
    <input type="submit" value="Valider">
</form>
<IMG SRC="<%= ladate.getDrapeau() %> "/>
<form method="get">
 choisissez votre pays :
</br>
 <select NAME="pays">
    <option VALUE=0> france 
    <option VALUE=1 selected>angleterre 
    <option VALUE=2>bretagne 
```

LES JAVA SERVER PAGES (JSP)

- Les pages JSP s'exécutent, en fait, sous la forme de servlets.
- Donc elles disposent du même cycle de vie.
- Les pages JSP simplifient la création de pages générées dynamiquement sur un serveur HTTP, en utilisant une démarche opposée à celle des servlets.
- Dans la démarche de la plateforme J2EE :

les servlets serviront à traiter les requêtes des clients

les pages jSP servirons plus à la présentation.

LES ENTREPRISE JAVABEANS

- Généralement, lorsque parlons de Java EE, nous pensons immédiatement au EJB
- Les EJB sont des composants Java qui implémentent la logique métier de l'application.
- L'architecture Java EE comporte un serveur qui sert de conteneur pour les EJB.
- Ce conteneur charge tous les composants à la demande et invoque les opérations qu'ils exposent, en appliquant les règles de sécurité et en contrôlant les transactions.
- Cette architecture est très complexe mais heureusement totalement transparente au développeur.

LES ENTREPRISE JAVABEANS

- Le conteneur d'EJB fournit automatiquement toutes la plomberie et le câblage nécessaire pour la réalisation d'applications d'entreprise.
- La création des EJB ressemble beaucoup à celle des objets RMI.
- Ils existent plusieurs types d' EJB

LES ENTREPRISE JAVABEANS

Les Beans sessions

- Comme leur nom l'indique, ont une durée de vie correspondant à celle de la "conversation" ou "session" entre l'application cliente et le composant.
- Servent à fournir à l'application cliente divers services conçus par le développeur.
- Selon le choix du dev., un Bean session peut maintenir un état pendant toute la durée de la session.
- Mais ne conserve aucun résultat auquel le client pourrait faire référence ultérieurement.

LES ENTREPRISE JAVABEANS

Les beans entités

- Représentent des objets métier du domaine de l'application tels que clients, factures, produits.

- Ces objets persistent de façon à pouvoir être réutilisés.
- Le conteneur de l'architecture J2EE s'occupe de tous les détails de cette tâche de persistance.
- Avec un bean entité, le développeur ne voit pas du tout la base de données et donc ne s'en occupe pas.

LES ENTREPRISE JAVABEANS

Les beans contrôlés par messages

- Fournit un modèle de composant permettant d'écouter un service de messages.
- La plateforme J2EE définit une queue de message, qui est une sorte de file d'attente dans laquelle les applications peuvent placer des messages.
- L'avantage de cette architecture est que les composants qui utilisent les messages n'ont pas besoin de savoir qui les a envoyés.
- Il leur suffit d'identifier la queue contenant les messages.

L'ARCHITECTURE MVC

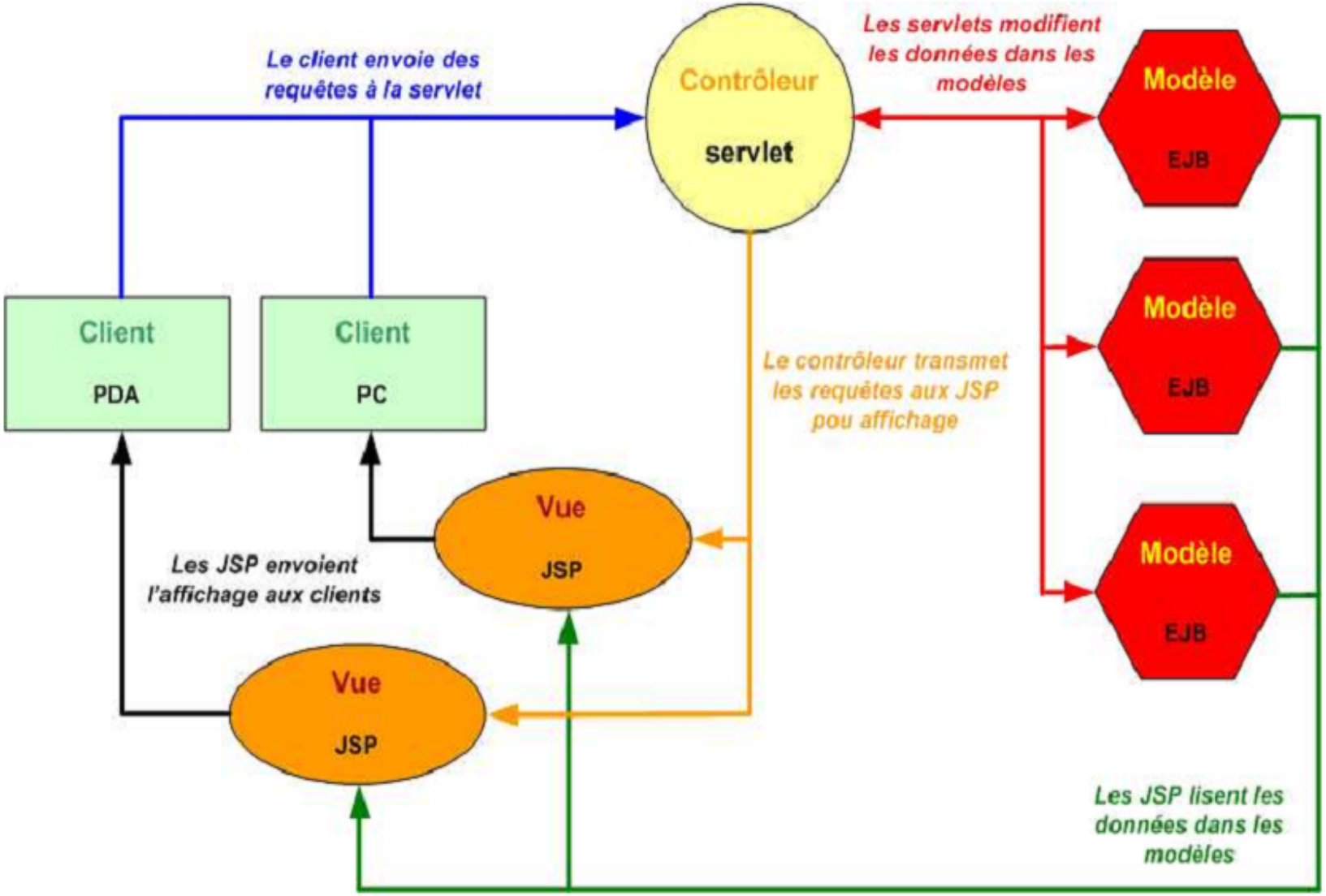
Modèle 1

- La logique métier, la logique d'affichage et la manipulation des requêtes sont mélangés dans un même composant.
- Cette architecture conduit à placer du code Java dans des JSP ou du code HTML dans les servlets.
- Lorsqu'il s'agit d'une petite application, cela ne pose pas de problème, et le sujet est correctement traité.
- Si c est une vraie application, la maintenance est super compliquée.

L'ARCHITECTURE MVC

Modèle 2

- un composant est chargé de recevoir les requêtes.
- un autre de traiter les données.
- un troisième de préparer l'affichage.
- Si les interfaces entre ces trois composants sont clairement définies, il devient facile d'en modifier un sans toucher aux deux autres.
- Dans ce contexte, il est d'ailleurs possible de prévoir plusieurs affichages, un pour un PC par exemple, et un autre pour un PDA.


Le client envoie des requêtes à la servlet
Contrôleur
servlet
Les servlets modifient les données dans les modèles
Modèle
EJB
Modèle
EJB
Modèle
EJB
Client
PDA
Client
PC
Le contrôleur transmet les requêtes aux JSP pou affichage
Vue
JSP
Les JSP envoient l'affichage aux clients
Vue
JSP
Les JSP lisent les données dans les modèles

L'ARCHITECTURE MVC

ROLE

- Web Designer : créé la Vue.
- Web Component Developer : créé le Contrôleur.
- Business Component Developer : créé le Modèle.
- Data Access Developer : créé l'accès à la base de donnée.

L'ARCHITECTURE MVC

Modèle

- Le modèle englobe à la fois la logique métier et les données sur lesquelles il opère.
- Toute classe Java peut jouer le rôle du modèle et de nombreuses applications Web utilisent uniquement des servlets ou des JSP et des classes ordinaires implémentant cette logique.
- Toutefois, les EJB sont des composants parfaits pour ce rôle.

L'ARCHITECTURE MVC

Vue

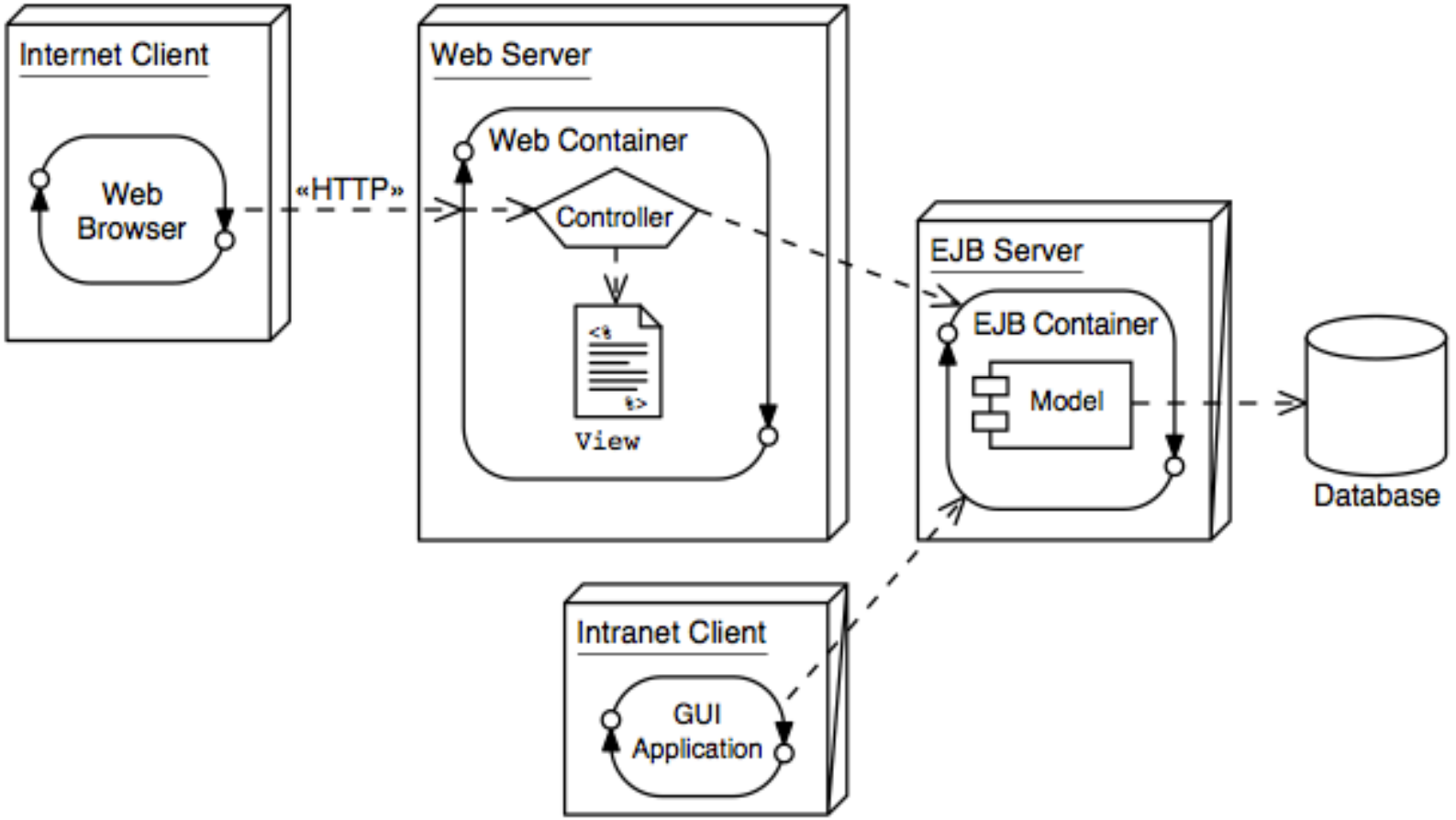
- Une fois la requête traitée, le contrôleur détermine quel composant doit être employé pour afficher les données.
- Dans les applications les plus simples, le composant contrôleur peut aussi jouer le rôle de vue.
- Dans les systèmes plus complexes, la vue et le contrôleur sont des composants distincts.
- Les pages JSP sont parfaitement adaptées à la vue puisque ces dernières accueillent de façon naturelle le balisage HTML.

L'ARCHITECTURE MVC

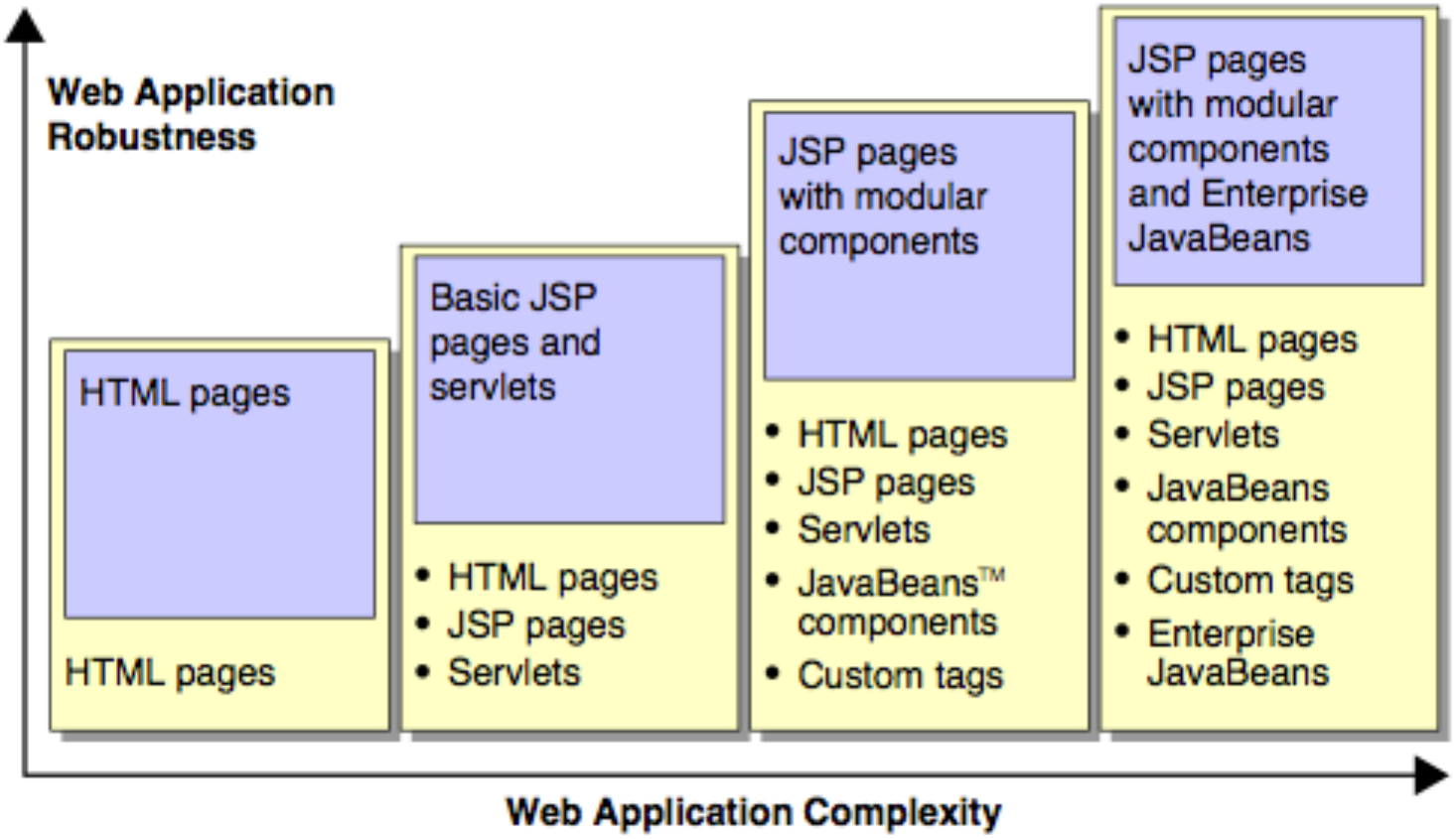
Contrôleur

- Les composants de cette catégorie reçoivent les requêtes des clients, les traitent et les transmettent aux composants chargés de traiter les données.
- Ils les dirigent ensuite vers les composants responsables de la vue.
- Tous les composants Web, tel un EJB, une servlet, ou une JSP peut jouer ce rôle.
- Toutefois, les servlets sont des composants dont la structure est la plus adaptée.
- Une servlet est conçue pour recevoir les requêtes des clients et leur retourner une réponse, ce qui est précisément le rôle du contrôleur

L'ARCHITECTURE MVC



L'ARCHITECTURE MVC



Blog GeekExplains

Difference between a Web Server, Web Container, and an Application Server

A **Web Server** is a server capable of receiving HTTP requests, interpreting them, processing the corresponding HTTP Responses and sending them to the appropriate clients (Web Browsers). **Example:** Apache Web Server. Read more about Web Servers and their working>>

A **Web Container** is a J2EE compliant implementation which provides an environment for the Servlets and JSPs to run. Putting it differently we can say that a Web Container is combination of a Servlet Engine and a JSP Engine. If an HTTP Request refers to a Web Component (typically a Servlet or a JSP) then the request is forwarded to the Web Container and the result of the request is sent back to Web Server, which uses that result to prepare the HTTP Response for the particular HTTP Request. **Example:** Tomcat is a typical Web Container. A typical setup would be to have Apache HTTP Server as the Web Server and Tomcat as the Web Container.

An **Application Server** is a complete server, which provides an environment for running the business components (EJBs, ADF BCs, etc.) in addition to providing the capabilities of a Web Container as well as of a Web Server. **Example:** Bea WebLogic, IBM WebSphere, Oracle Application Server, etc.