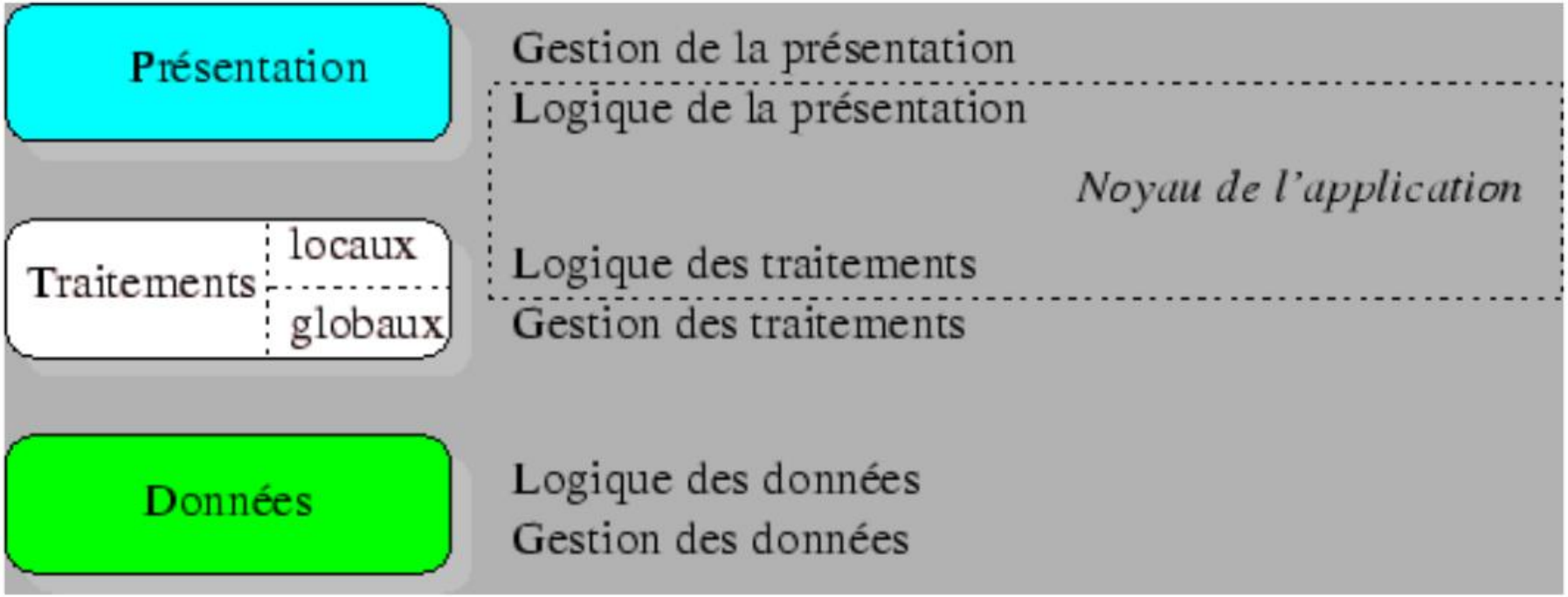
C3I-SI / NSY102

Intergiciels à objets répartis
Conception et implantation

Architectures N -Tiers

Architecture des applications (1)

Les 3 niveaux d'abstraction d'une application



La couche de présentation: Fournit et gère les IHM pour permettre l'interaction avec l'utilisateur.

Architecture des applications (2)

La logique applicative: (les traitements)

Les traitements locaux:

- Mise en forme des données dans l'IHM
- Contrôle et aide à la saisie

Les traitements globaux:

Fonctionnalités mises à disposition par l'application

- Contient et applique les règles de gestion
- C'est la couche incluant la « Logique Métier »

Architecture des applications (3)

La gestion des données :

Ensemble des mécanismes permettant l'exploitation et le stockage des données manipulées par l'application.

« Persistance des données métiers »

Architecture des applications (4)

- Les trois niveaux : **Présentation**, **Traitements et Données** peuvent être imbriqués ou répartis de différentes manières entre **une ou plusieurs machines physiques**.

- Le découpage et la répartition de ces niveaux conduisent aux architectures suivantes :

Architectures 1 tiers

Architectures 2 tiers

Architectures 3 tiers et N-tiers

Architectures 1 Tiers (1)

Application sur ordinateur central

- Déployée et développée sur « Mainframe »
- Utilisation à l'aide de terminaux passifs.

Avantages :

- Utilisation optimale des ressources
- Simplicité de mise en oeuvre
- Haute fiabilité

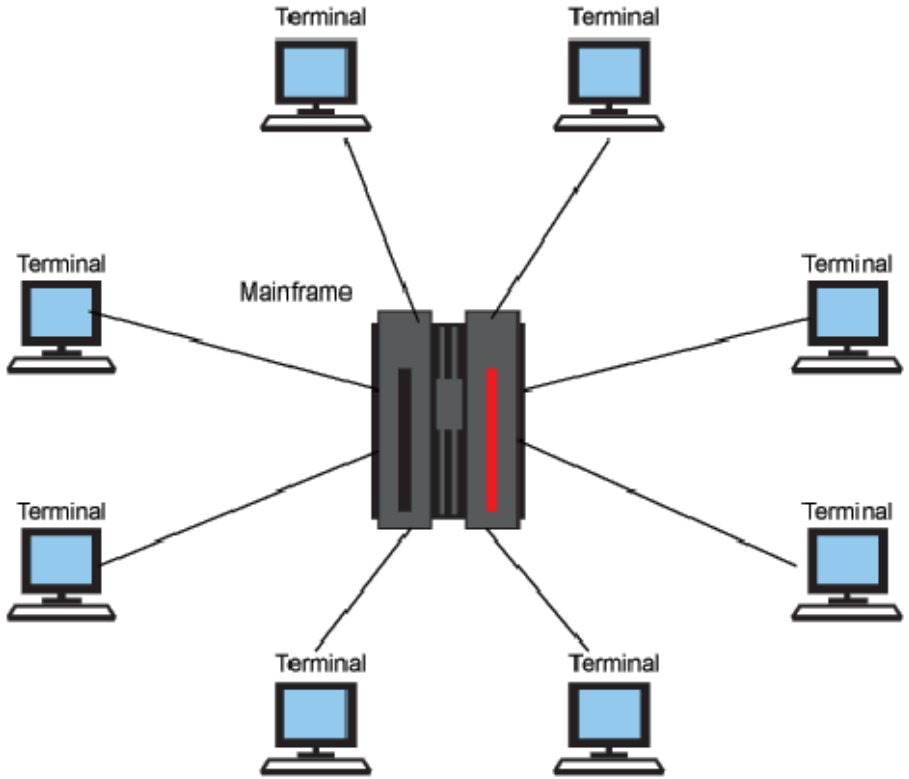
Inconvénients :

- Interface graphique en mode caractères.

Architectures 1 Tiers (2)

- **Applications déployées et développées sur « Mainframe » :**

Données, SGBD et traitements sur même ordinateur (Ordinateur ou Mini Ordinateur)



Architectures 1 Tiers (3)

Arrivée des micro-ordinateurs :

- **« Revamping » :** ré-habillage d'applications centralisées (Mainframe) avec des interfaces graphiques modernes.
- **Applications complètes fonctionnant sur micro-ordinateur.**
- **Applications multi-utilisateurs :** même application déployée sur plusieurs machines, partage de fichiers de données. (chaque application exécute son propre moteur de SGBD)

Avantages :

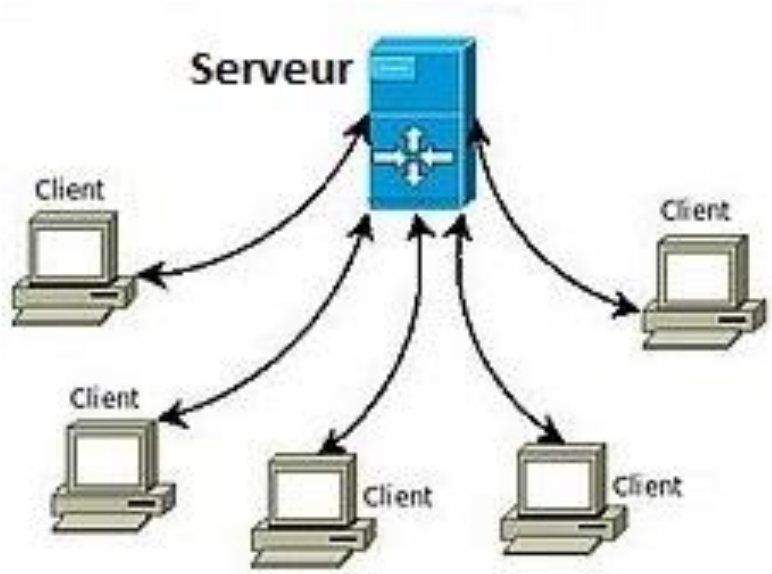
- Interfaces graphiques plus attrayantes

Inconvénients :

- Accès aux données conduit à une forte utilisation du réseau
- Mode multi-utilisateur est générateur de nouvelles problèmatiques.

Architectures 1 Tiers (4)

- **Applications multi-utilisateurs :** Les données sont sur un serveur, les traitements sur les postes clients utilisent les même données (partagées).



- **Problèmatiques :**
  - Accès simultanés *(comment garantir la consistance des données ?)*
  - Forte utilisation du reseau

Architectures 1 Tiers (5)

Recherche de solutions pour concilier :

- La fiabilité de la gestion centralisée des données
- L'interface utilisateur moderne

Solution: scinder l'application en 2 parties distinctes

- Gestion centralisée des données (SGBD)
- Gestion locale de l'application.

Naissance du concept de **client-serveur** et **d'architecture distribuée**

Le Mode Client-Serveur (1)

2 entités communicantes :

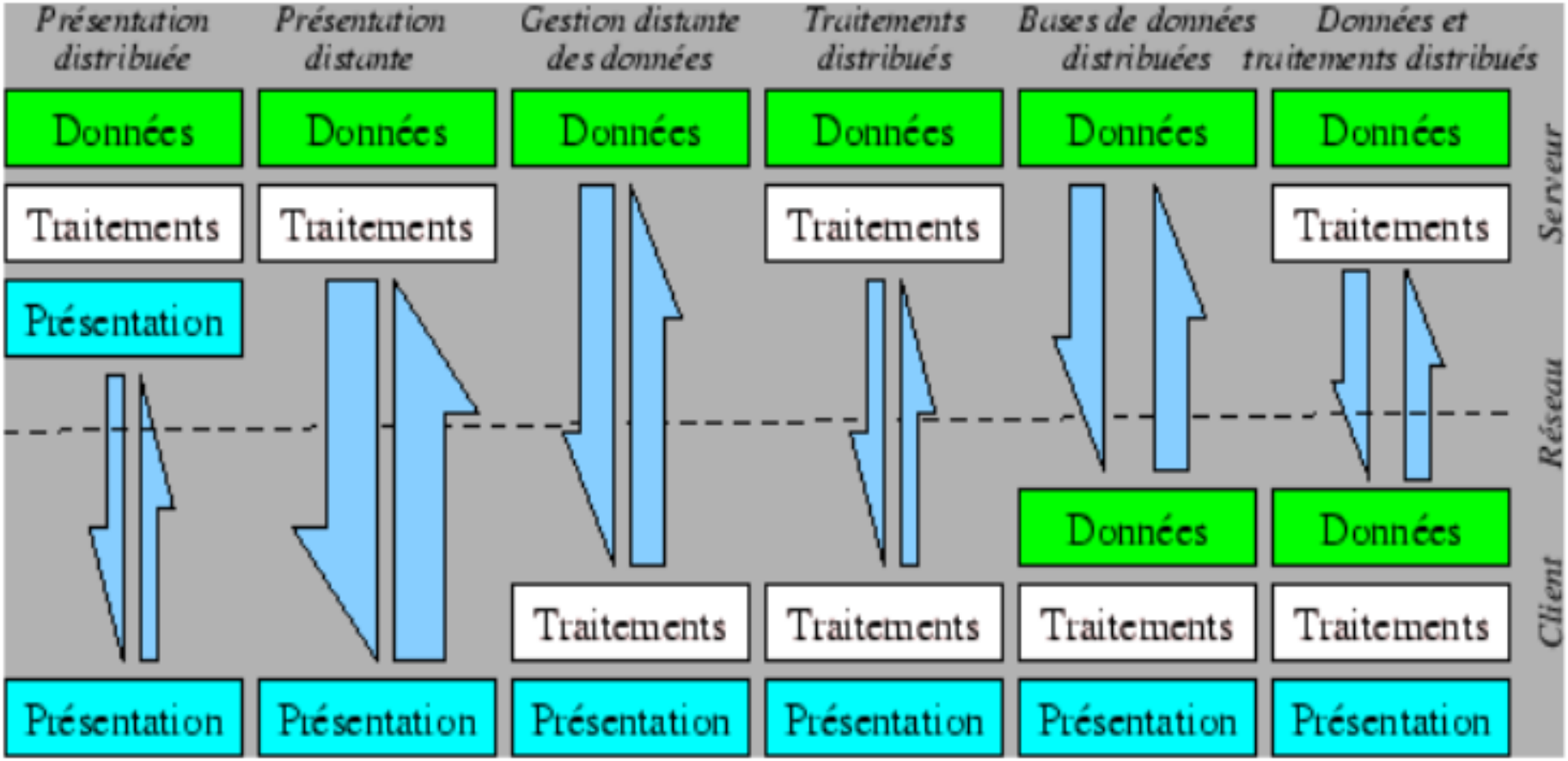
- **Le serveur :** met à disposition des services accessibles à distance.
- **Le client :** il est consommateur (utilisateur) de ces services

Intéraction Client / Serveur:

- Le serveur attend les demandes de service.
- Le client envoie au serveur une demande de service.
- Le serveur traite la demande et renvoie le résultat.
- Le client reçoit le résultat et l'exploite.

Le Mode Client-Serveur (2)

Différentes architectures client-serveur (schéma du Gartner Group)



Architectures 2 Tiers (1)

Nommée aussi « Client-Serveur de données »

Côté client :

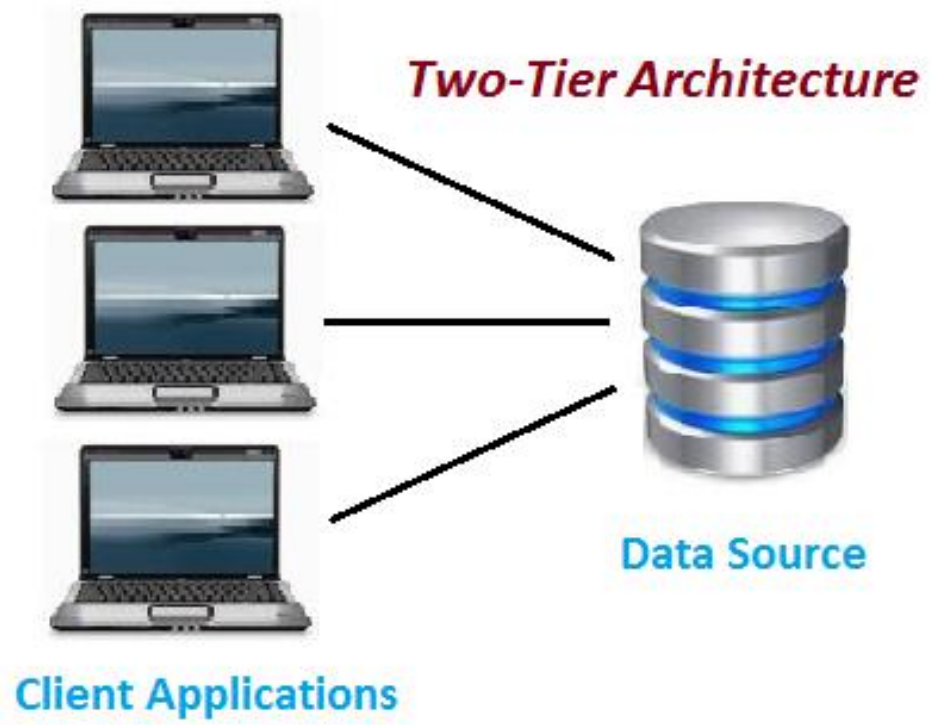
- Prise en charge de la présentation
- Traitements (logique métier) implantée sur poste client.

Serveur :

- Gestion des données déléguée à un SGBD centralisé. (Serveur Bdd dédié)
- Communication entre client et serveur assurée par un **« Middleware »** ou **« Élément du milieu»**. (pilote SGBD, protocole échange de données etc.)

Architectures 2 Tiers (2)

« Client-Serveur de données »



- Présentation et traitements dans l'application cliente en exploitant les données du SGBD (Data Source)

Architectures 2 Tiers (3)

Avantage :

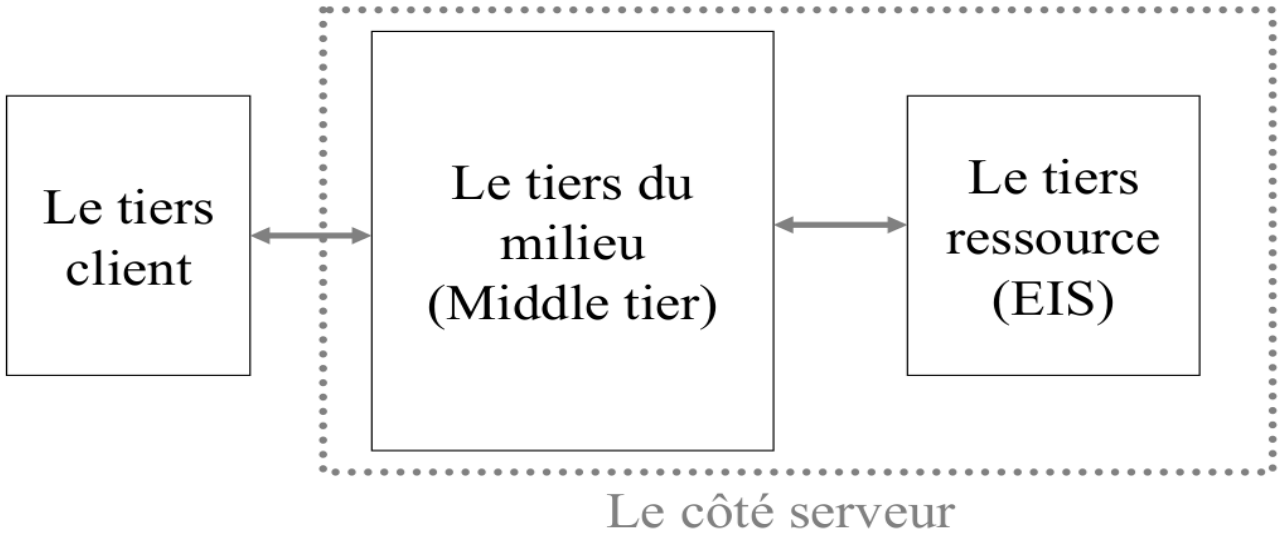
- Gestion centralisée des données

Inconvénients :

- Le **« Client lourd »** supporte la majorité des traitements applicatifs, problème de performances.
- Conversation entre client et serveur assez **« bruyante »**
- Architecture statique : Difficulté pour **réagir à la montée en charge**.
- Le fort couplage entre le client et le serveur complique les évolutions.

Recherche d'une architecture **plus évoluée** facilitant les **forts déploiements à moindre coût** : **Architecture 3 Tiers**

Architectures 3 Tiers (1)



- La **présentation** est prise en charge par le poste client
- La **logique applicative** est prise en charge par le serveur intermédiaire.
- Les **données** sont gérées de façon centralisée (SGBD)

Architectures 3 Tiers (2)

Séparation en 3 niveaux distincts:

- **Premier niveau :** Affichage + traitements locaux (poste client)
- **Second niveau :** Traitements applicatifs globaux => Services applicatifs (serveur applicatif)
- **Troisième niveau :** Les services d'accès et de gestion des données (serveur de donnée : SGBD)

Ces niveaux sont indépendants → Implantation possible sur des machines différentes.

Possibilité d'utiliser **un client « Léger » :** le client n'assure que la présentation de l'application (éventuellement quelques contrôles locaux). Il peut être réduit à un **simple navigateur Web**.

Serveur applicatif : Un serveur reçoit et traite toutes les demandes de services des clients de l'application (+ services non fonctionnels)

Architectures 3 Tiers (3)

Apports de l'architecture 3 Tiers:

- Meilleure **séparation entre les 3 couches**.
- Serveur applicatif peut prendre en charge les besoins non fonctionnels (session, persistance, transaction, sécurité), le développeur **se concentrer sur la logique métier** de son application.
- **Facilité de maintenance :** action corrective peut se limiter à quelques composants.

Nouvelle problématique :

- Le serveur applicatif devient l'acteur principal de l'application, il est fortement sollicité : problème de charge.

Pour résoudre ce problème : Répartition des services entre plusieurs serveurs, Redondance matérielle et logicielle, Bdd répliquées:

Les architectures N-Tiers

Architectures N Tiers (1)

- **N Tiers: ne signifie pas N couches :**
  - **Tiers physiques :**

    Serveurs, redondance matérielle, réplication
  - **Tiers logiques :**

    Découpage en composants coopérants (les services) pas forcément liés à la répartition physique.
  - **Distribution de la logique applicative** entre plusieurs composants pouvant être déployés sur plusieurs serveurs.
  - **Mise en œuvre d'une approche objet :**
    - **Services et Objets métiers :** composants logiciel réutilisables
    - **ORB** pour rendre transparente la communication entre objets (**CORBA, RMI, DCOM**)

Architectures N Tiers (2)

- **Apports des architectures N Tiers :**
  - Vision cohérente du système d'information
  - Services représentés par des objets interchangeables, implantés en fonction des besoins (les composants)
  - Permet de concevoir des applications puissantes, modulaires, évolutives et simples à maintenir.

Une **bonne conception**, et une **bonne répartition** de l'application sont **INDISPENSABLES** pour en bénéficier.

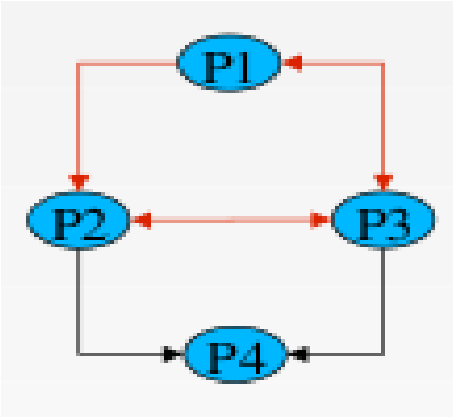
Pour qu'un logiciel soit **extensible et réutilisable** il doit être **découpé en modules ou composants faiblement couplés** et à forte cohésion.

Couplage (1)

Définition: « Une entité (fonction, module, classe, package, composant) est couplée à une autre si elle dépend d'elle. »

Couplage fort :

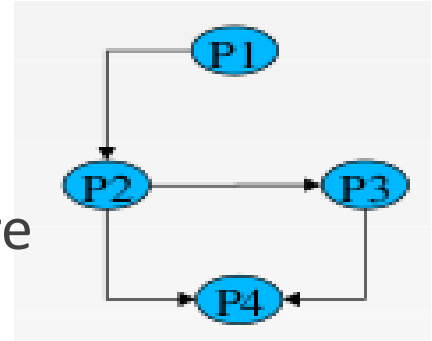
- Inter-dépendance entre les composants de l'application



- Induit des problèmes pour le développement, les tests et la maintenance

Couplage (2)

Couplage faible :



- Pas d'inter-dépendance entre les composants :
- Permet de maitriser la complexité de l'architecture

Couplage très faible :

- Les composants sont interchangeables, indépendants, de différentes technologies, conçus en toute indépendance.
- Permet de faciliter la construction, l'évolution, l'adaptation, la maintenance.
- Application construite par l'assemblage de composants collaborants

Architecture orientée service (SOA) : composants répartis collaborant pour «s'échanger des services »

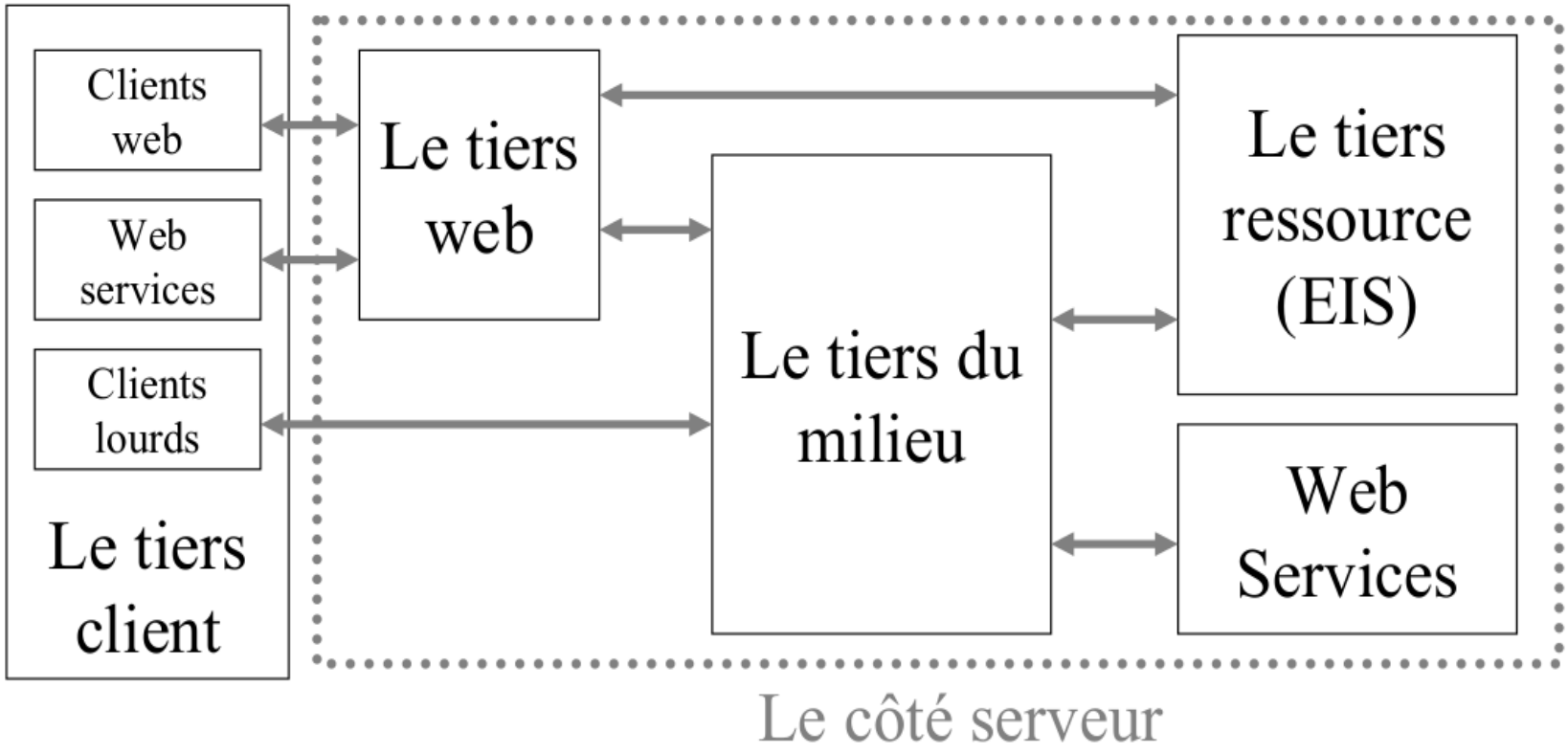
Couplage (3)

Comment parvenir à réduire le couplage ?

- Chaque composant doit être un **fournisseur de services** autour d'une thématique métier ou technique.
- **Il doit exposer ce qu'il sait faire mais jamais comment il le fait**. La collaboration entre composants se base sur des «contrats de service»
- Lorsqu'un composant en utilise un autre, il n'a pas besoin de (ne doit pas) connaître son implémentation: il n'est donc pas en mesure de décider de l'implémentation qu'il va utiliser.
- Des Design Pattern permettent de réduire le couplage tout en permettant la collaboration entre composants :

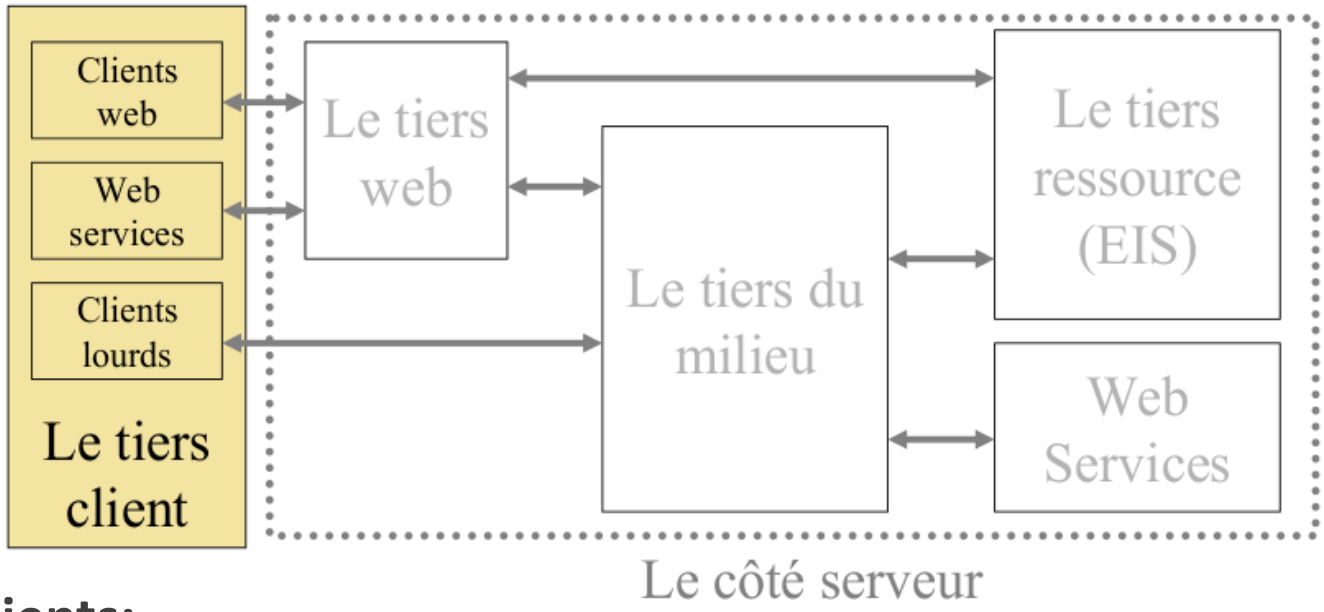
  IOC, Injection de dépendance, Commande, Proxy. (NFP121)

Architectures Web (4 Tiers)



Un nouveau tiers : le tiers Web

Architectures Web: le tiers client



Types de clients:

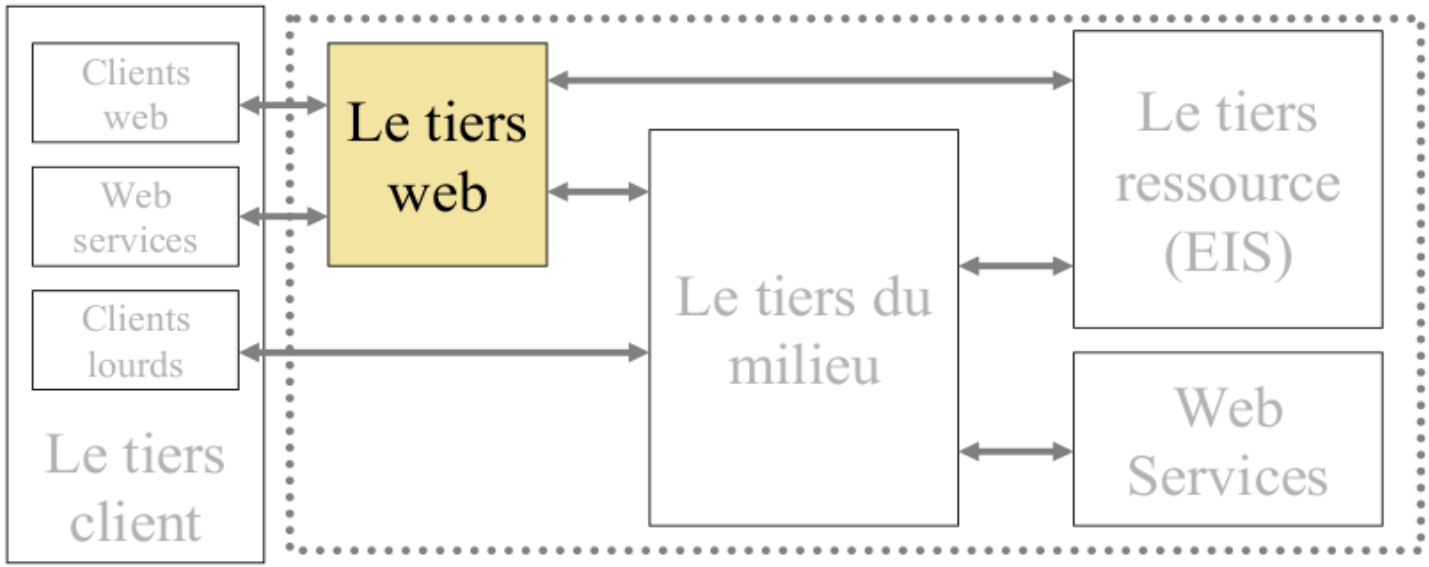
Navigateur Web, PDA, Téléphone mobile : HTTP / HTTPS , HTML / XML

Client lourd : application, applet : **RMI**, CORBA, **IIOP**, MOM, **JMS**, autres.

Consommateur Web-service : HTTP / HTTPS, **SOAP**, **REST**

Les clients obtiennent les services mis à disposition côté serveur à partir de requêtes dans divers protocoles.

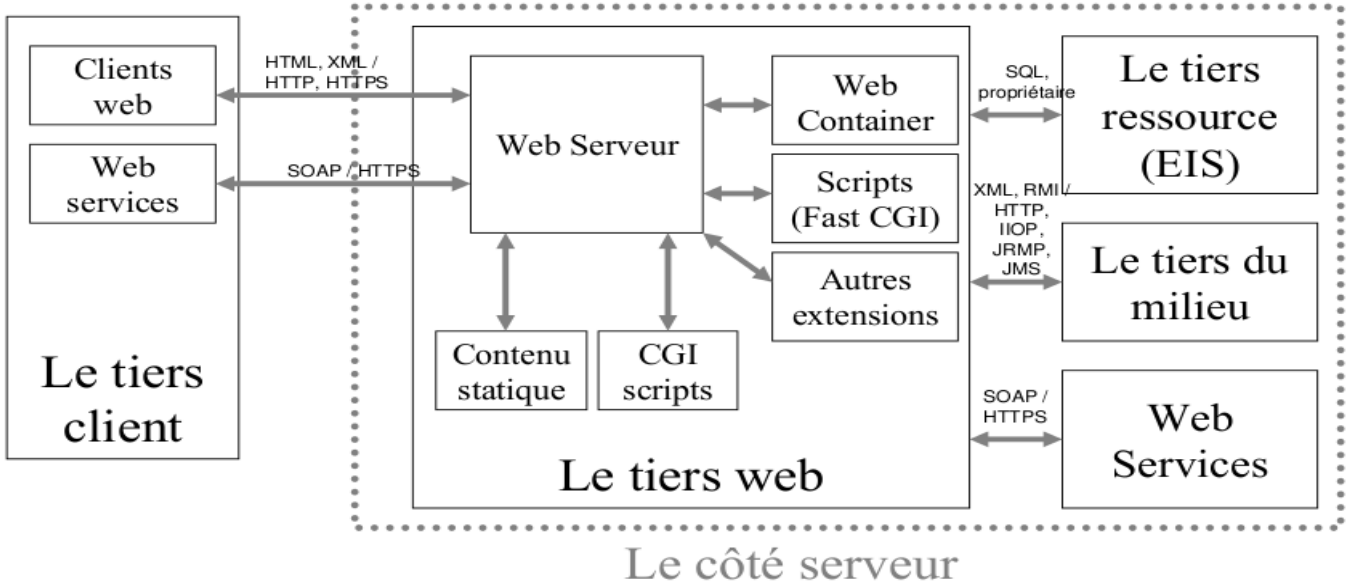
Architectures Web: le tiers Web (1)



Rôles du tiers Web:

- Reçoit les requêtes Http des clients, les fait traiter et renvoie la réponse.
- Assure une séparation entre la présentation de l'application cliente et la logique métier.
- Transforme les requêtes Http dans un format compris par l'application.
- Assure le suivi des sessions utilisateur.

Architectures Web: le tiers Web (2)



Technologies utilisées dans le tiers Web:

CGI (Common gateway interface): génération de page web. (Java, C, C++, Perl, autres.)

ASP : Scripting interprété dans des pages Html (Microsoft)

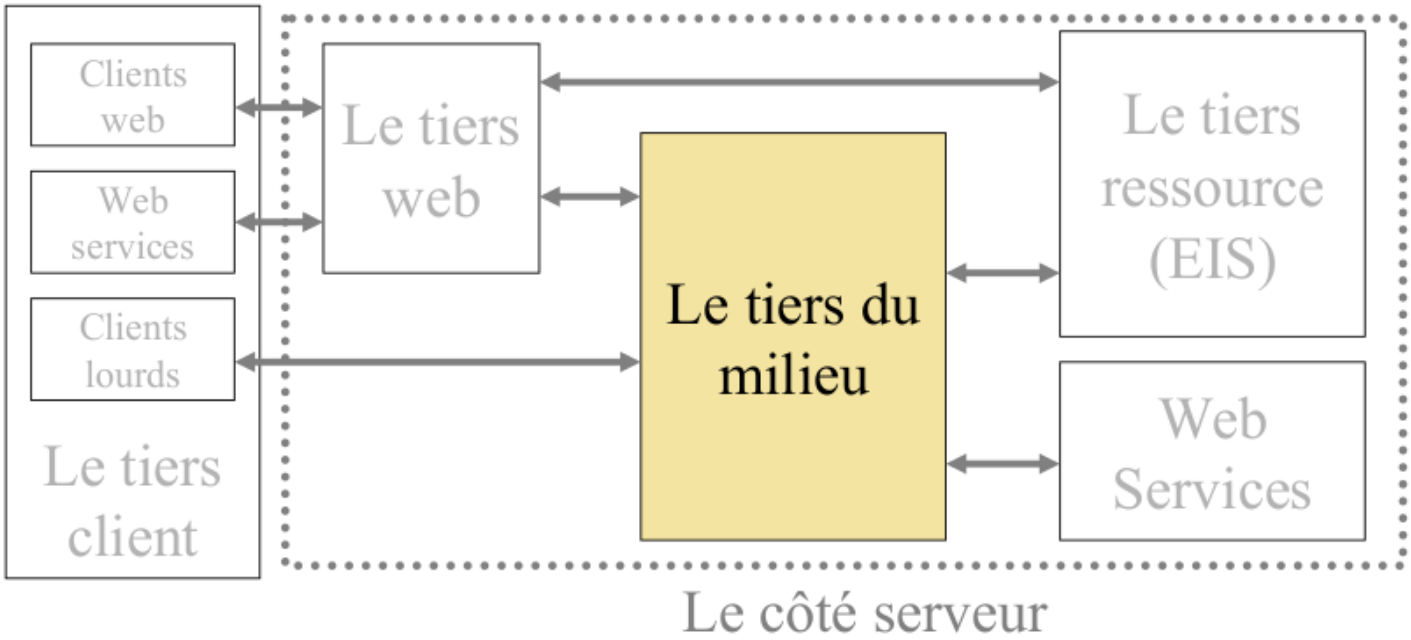
Servlets : technologie Java (gérées par un conteneur)

JSP : Scripting dans des pages Html (Java) compilé en Servlet.

PHP, Python, Extensions spécifiques.

en JEE: Servlet, JSP, JSF

Architectures Web: le tiers du milieu (1)



Rôles du tiers du milieu (1):

Gestion des composants :

- **Composants système :** gestionnaire de transaction, gestionnaire MOM, administration, sécurité, etc.
- **Composants de l'application :** services et objets métiers

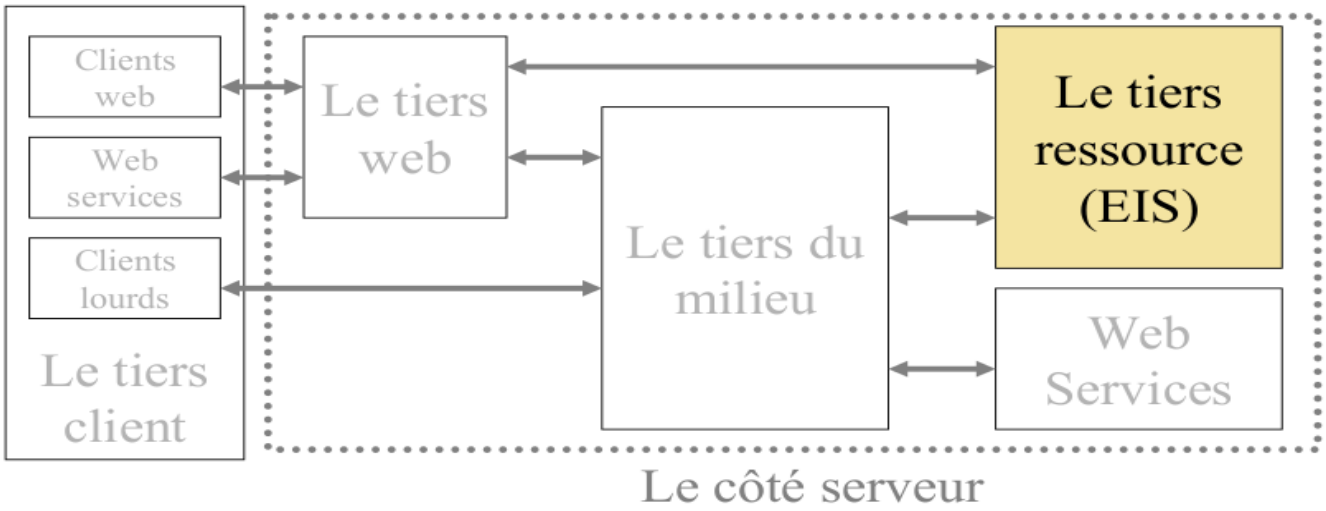
Qualités attendues : Tolérance aux fautes, haute disponibilité

Architectures Web: le tiers du milieu (2)

Rôles du tiers du milieu (2):

- **Réaliser la logique métier :** traitements des applications métiers déployées.
- **Passage à l'échelle :** capacité à accroître ses ressources pour faire face à un nombre de demandes accru sans dégrader les temps de réponse (scalability)
- **Répartition de charge :** Capacité à rediriger les demandes vers d'autres serveurs en fonction de leur disponibilité (redondance matérielle et logicielle).
- **Pooling de ressources :** limite la sollicitation du tiers ressources en utilisant le même accès pour plusieurs clients (exemple : Pool de connexions à une Bdd).
- **Gestion de transaction :** Une transaction est une unité indivisible de travail comprenant plusieurs opérations, dont toutes ou aucune doivent être effectuées pour protéger l'intégrité des données (ACID). Transaction BDD et Transaction applicatives manipulant les objets métier. (**JTA**)
- **Sécurité :** gestion de l'authentification et des autorisations.
- **Console de gestion :** système permettant de contrôler l'ensemble du système et de tous ses serveurs.

Architectures Web: le tiers ressource



Rôle du tiers ressource: Stockage et accès des données de l'application.

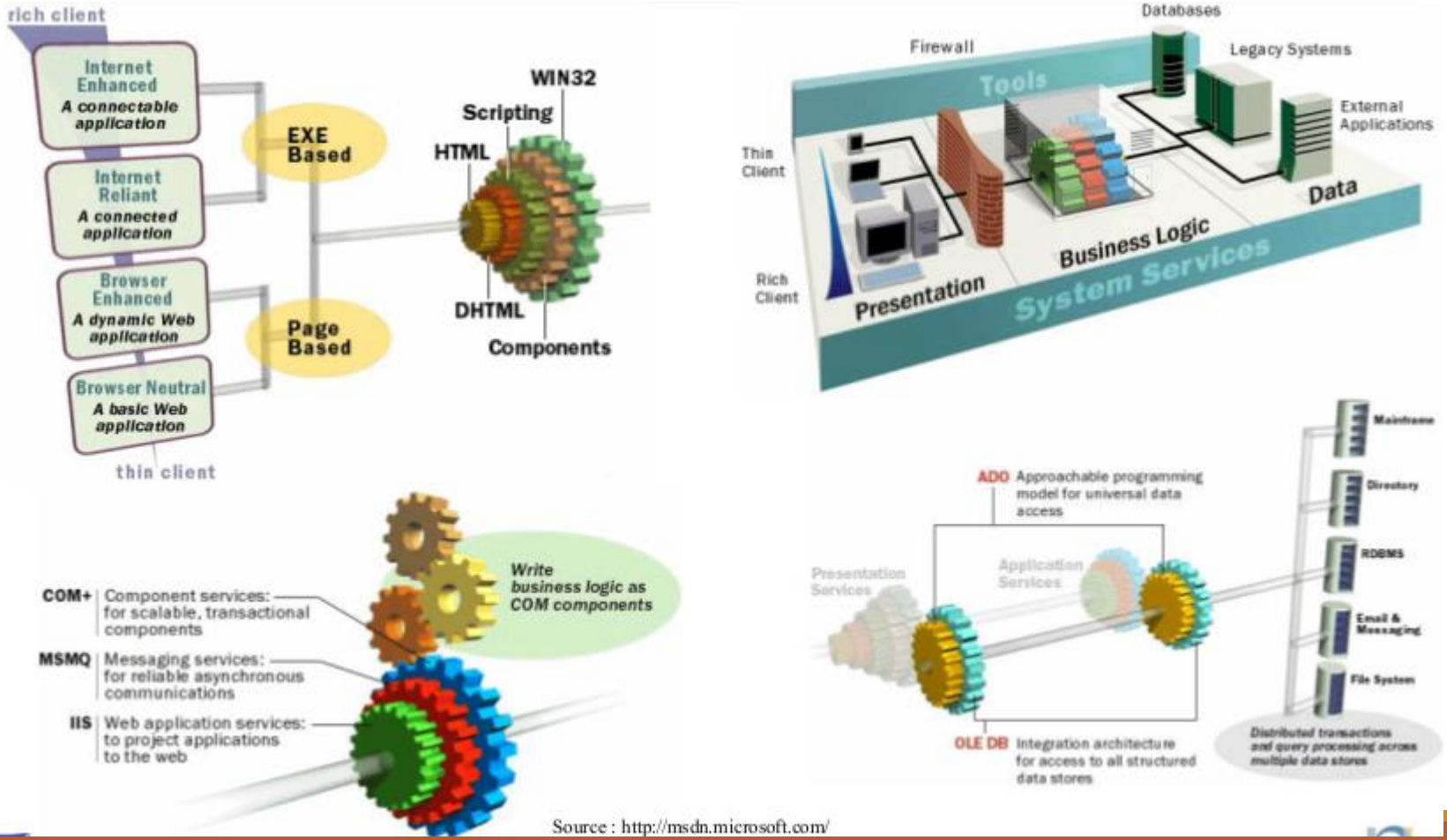
Connecteurs vers bases de données : JDO, JDBC, ADO.NET

Connecteurs vers données de systèmes propriétaires par protocoles propriétaires : ERP, EAI

Design Pattern: **DAO**, **DTO**
Persistence : **Spécification JPA, implementations EclipseLink et Hibernate**

Architectures Distribuées de Microsoft

Microsoft DNA (Distributed internet Architecture)

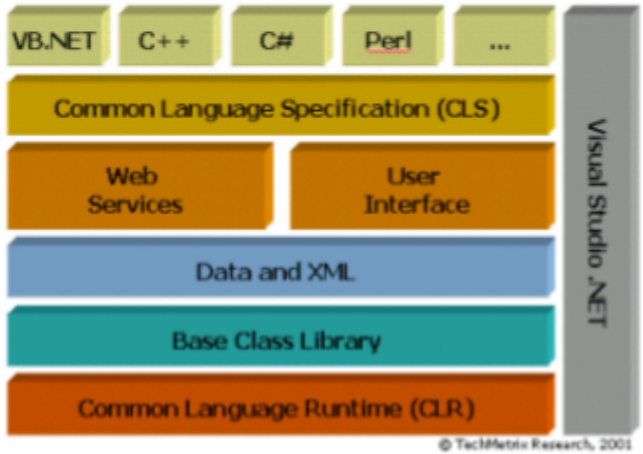


Source : http://msdn.microsoft.com/

Architectures Distribuées de Microsoft

Plate-forme Microsoft.NET

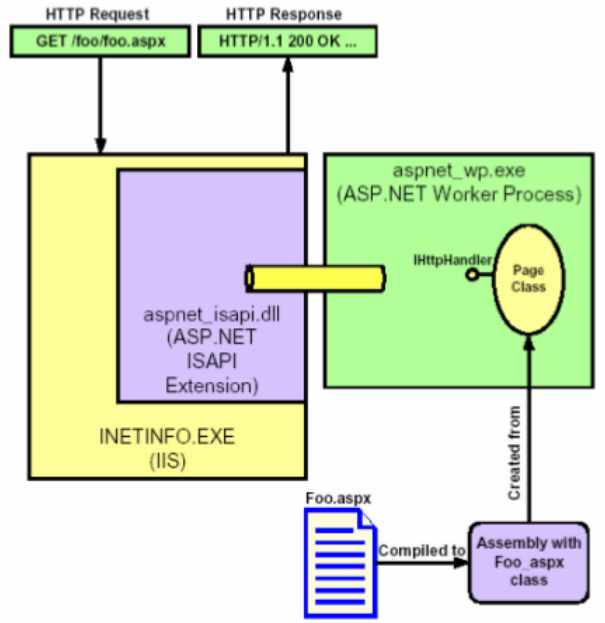
- .NET est une stratégie de produits M$
- Remplacement de Microsoft DNA
- Composé de 3 parties :
  - CLR (Common Language Runtime)
  - BCL (Base Class Library)
  - ASP.NET
- CLS (Common Language Specification)
- CTS (Common Type System)
- MSIL (Microsoft Intermediate Language)



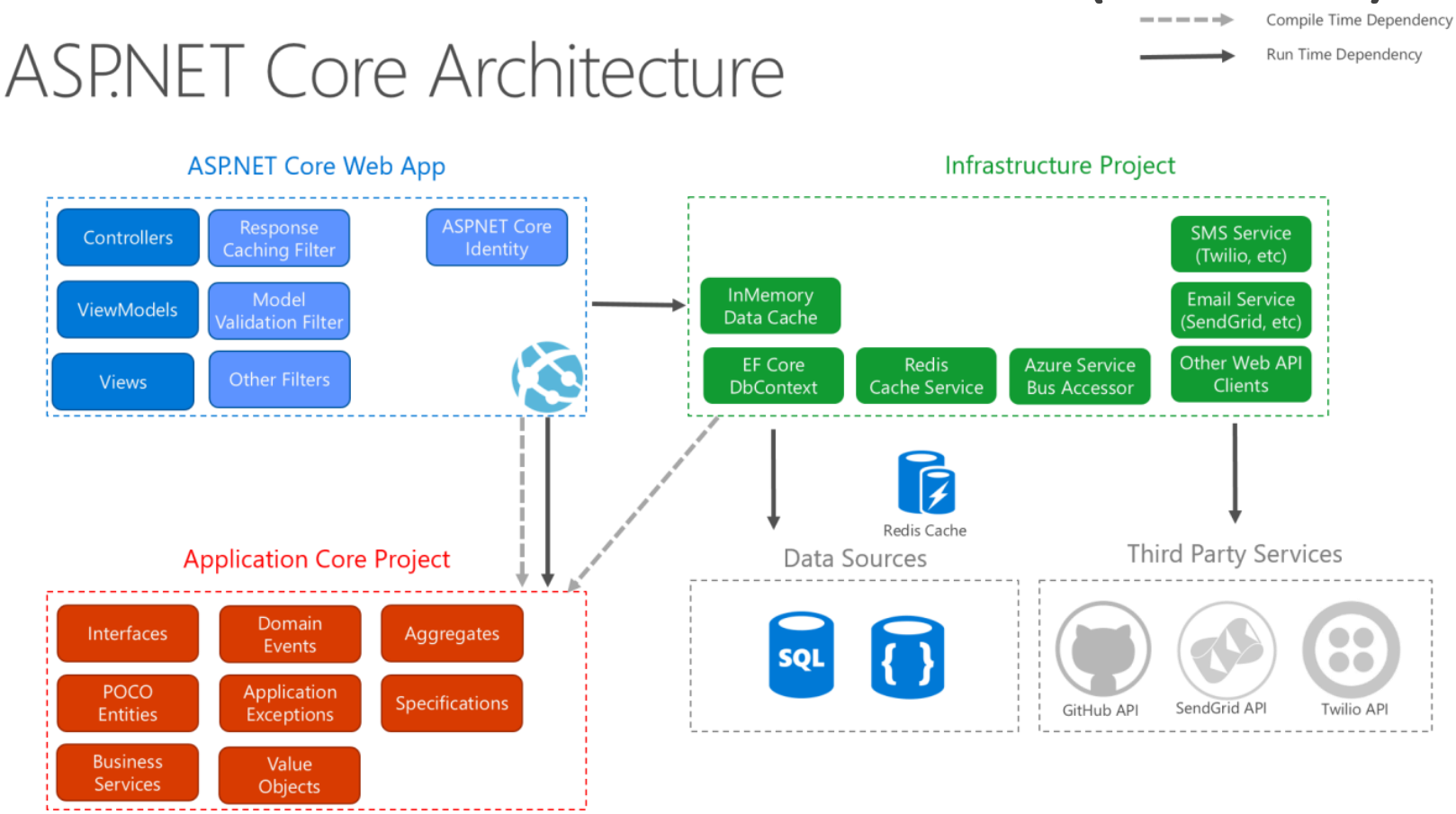
Architecture Distribuée de Microsoft

Plate-forme Microsoft.NET : ASP.NET

- ASP.NET est une abstraction de HTTP
  - 3 abstractions : context, handler, module
  - pages compilées et exécutées dans le CLR
    - modèle de développement basé sur les WebForms qui permet de développer une interface graphique Web comme une interface graphique VB
  - sépare traitements et présentation
    - le formulaire représente la page Web
    - les traitements sont contenus dans une seconde page appelée Code Behind
    - Cette page peut être codée dans n'importe quel langage du Framework .NET et implémente les événements liés à cette page. La page HTML finale qui sera générée au client intègre la présentation et le Code Behind en ciblant différents navigateurs.



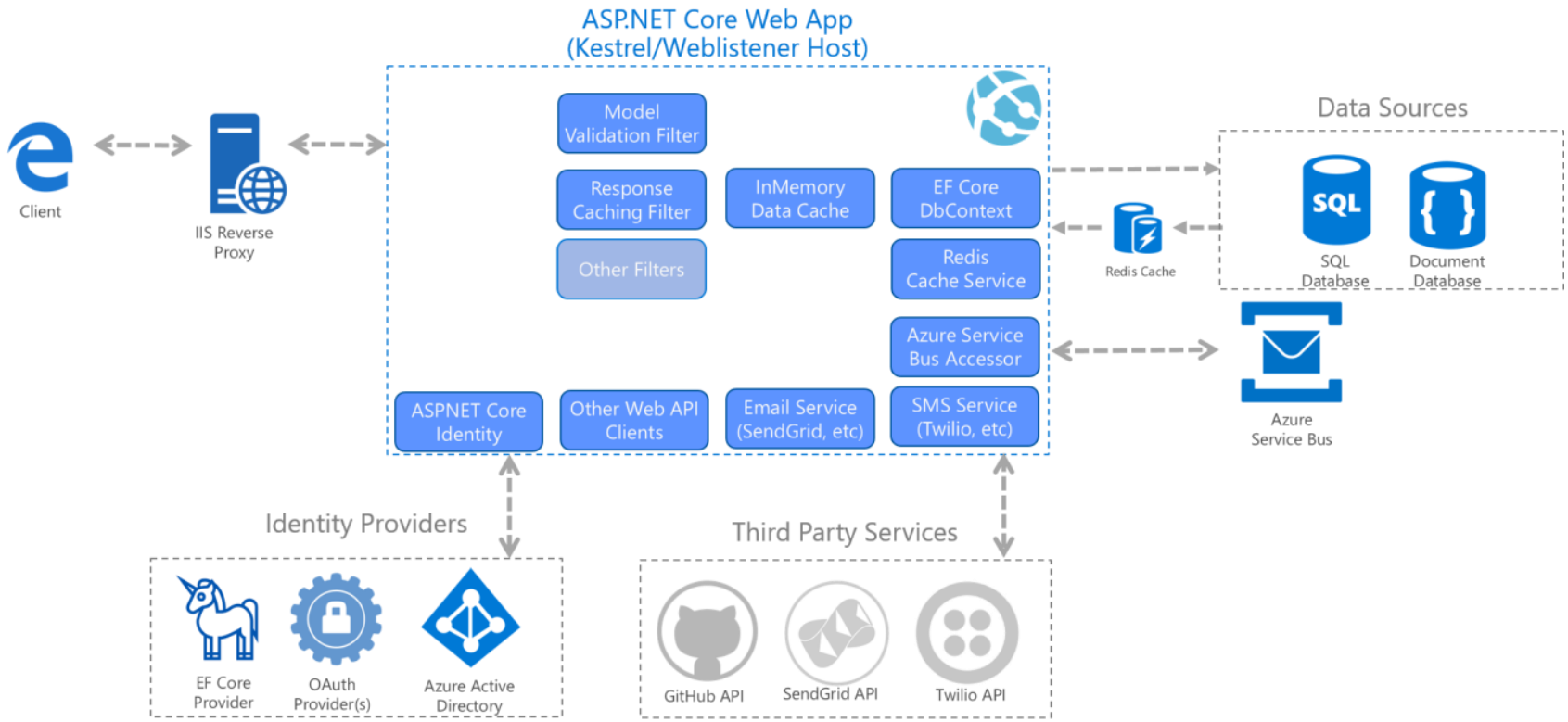
Architecture Distribuée de Microsoft (actuelle)



https://docs.microsoft.com/en-us/dotnet/standard/modern-web-apps-azure-architecture/common-web-application-architectures

Architecture Distribuée de Microsoft (actuelle)

ASP.NET Core Architecture

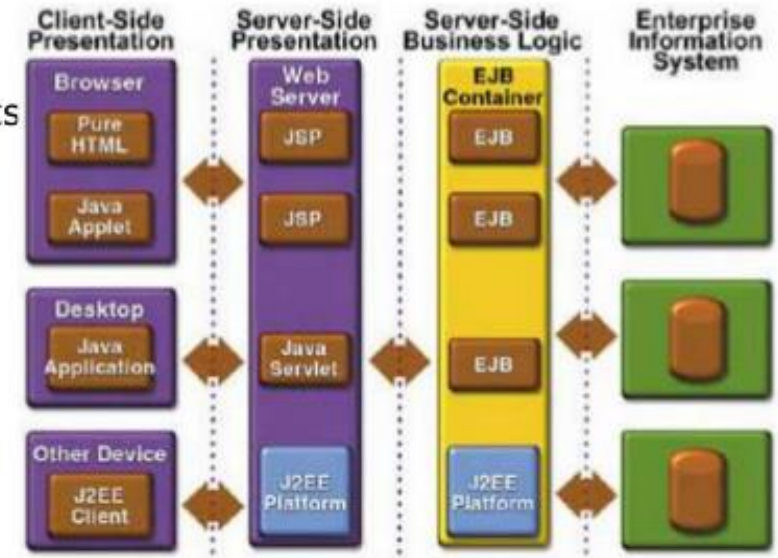


https://docs.microsoft.com/en-us/dotnet/standard/modern-web-apps-azure-architecture/common-web-application-architectures

Architecture Distribuée JAVA

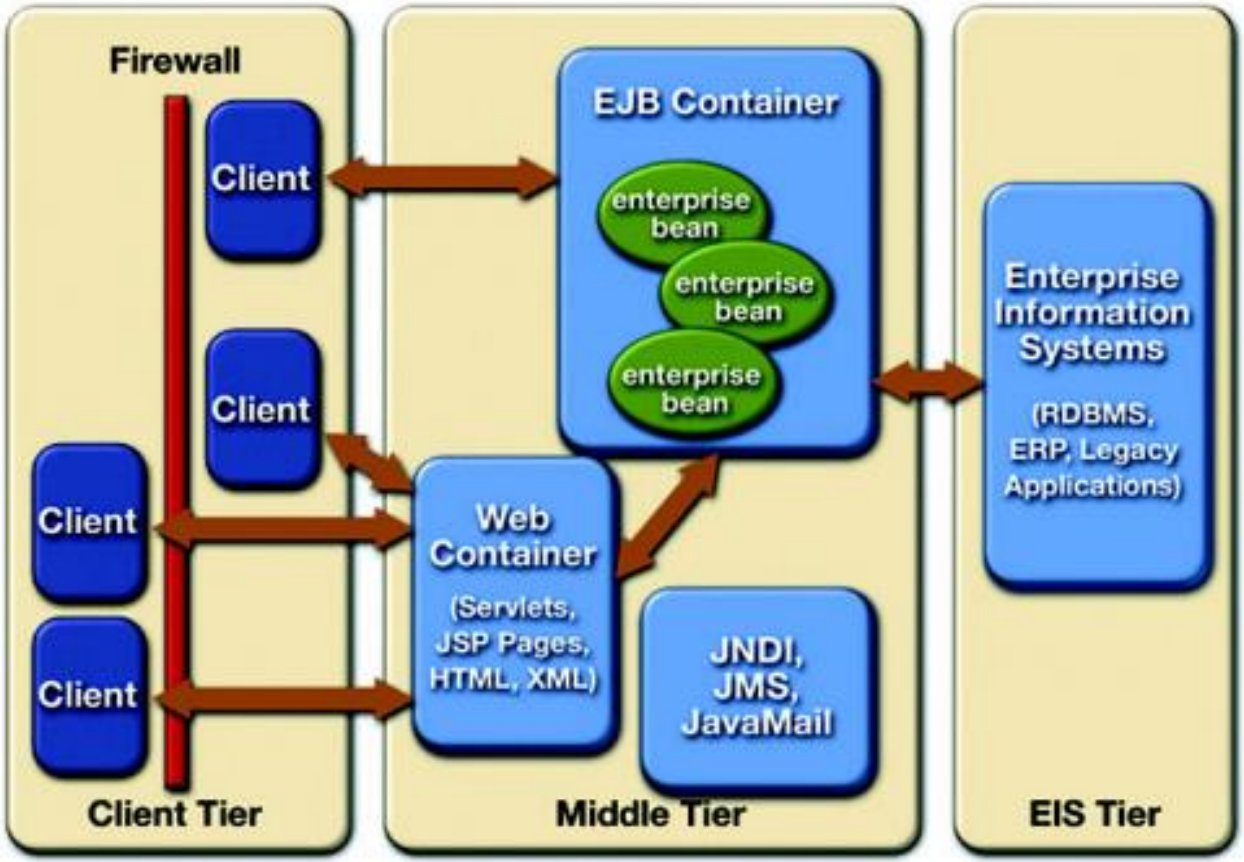
La plate-forme JEE (Java Enterprise Edition)

- J2EE est un standard industriel
  - contrairement à .net c'est une spécification
- Une application J2EE assemble des composants
  - composants clients : applications clients applets
  - composants web : servlet et JSP
  - composants business : EJB
  - écrit en Java compilé en bytecode
  - assemblés dans l'application J2EE
  - déployés dans un serveur J2EE
  - Le serveur J2EE fournit des conteneurs qui permettent de simplifier les composants et d'offrir tous les services nécessaires



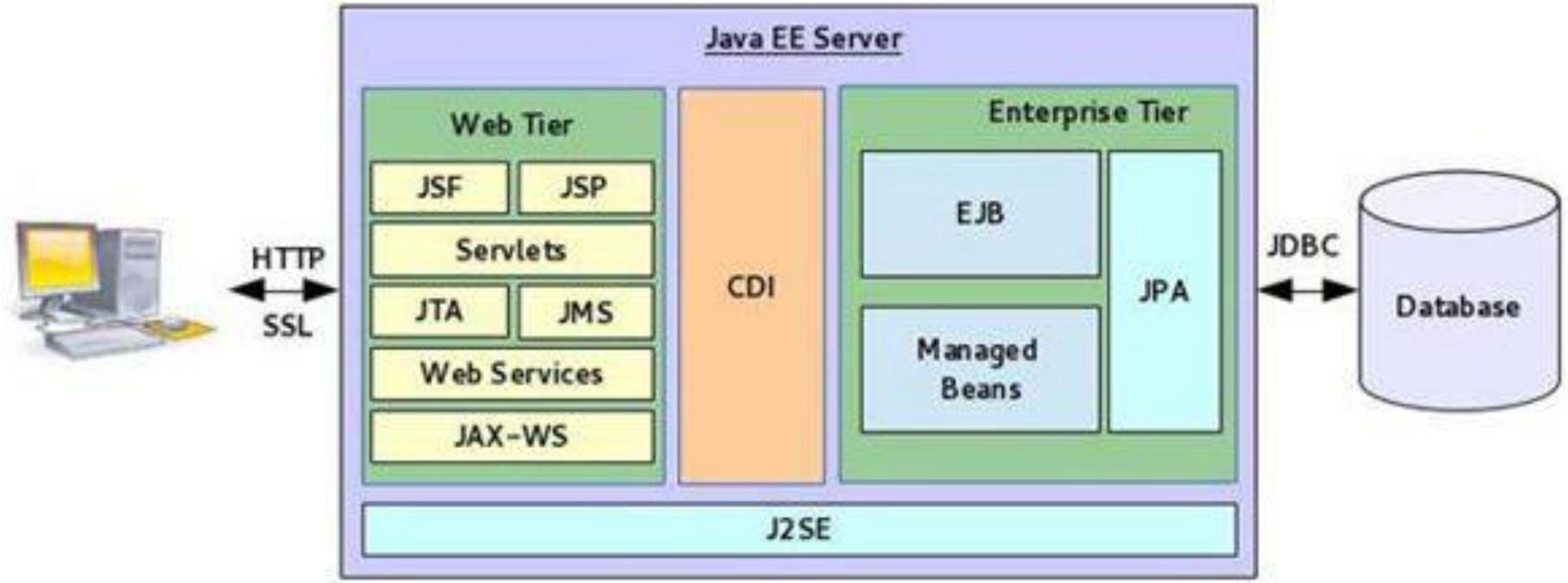
Architecture Distribuée JAVA

JEE (Java Enterprise Edition)



Architecture Distribuée JAVA

JEE (Java Enterprise Edition)



EJB : Enterprise Java Bean, les composants (Session, Entity, Timer, Message Driven)
CDI : Context Dependency Injection

Architecture Distribuée JAVA

Composants de la plate-forme JEE

Java Servlet Technology : prise en charge des servlets Requête / Réponse HTTP traitées en Java.

Java Server Pages (JSP) : page html contenant du code java (compilées en Servlet)

Java Server Faces (JSF) : page en xhtml contenant des références à des ManagedBean (compilées en Servlet)

Java Messaging Service (JMS) : MOM, communication asynchrone

Java Transaction API (JTA) : service de gestion de transaction

JavaMail : Service pour l'envoi de Mails par les composants.

Java Authentification and Authorization Service (JAAS)

Architecture Distribuée JAVA

Design patterns dans JEE

Tiers Présentation :

MVC, Composite view

Tiers applicatif :

Service locator, **Dependency Injection**, Business delegate, **Proxy**, Value object et Value object assembler (**DTO**), session façade, composite entity, value liste handler.

Tiers ressource :

Data Access Object (**DAO**)