Architectures n-tiers et déploiement d'applications Web

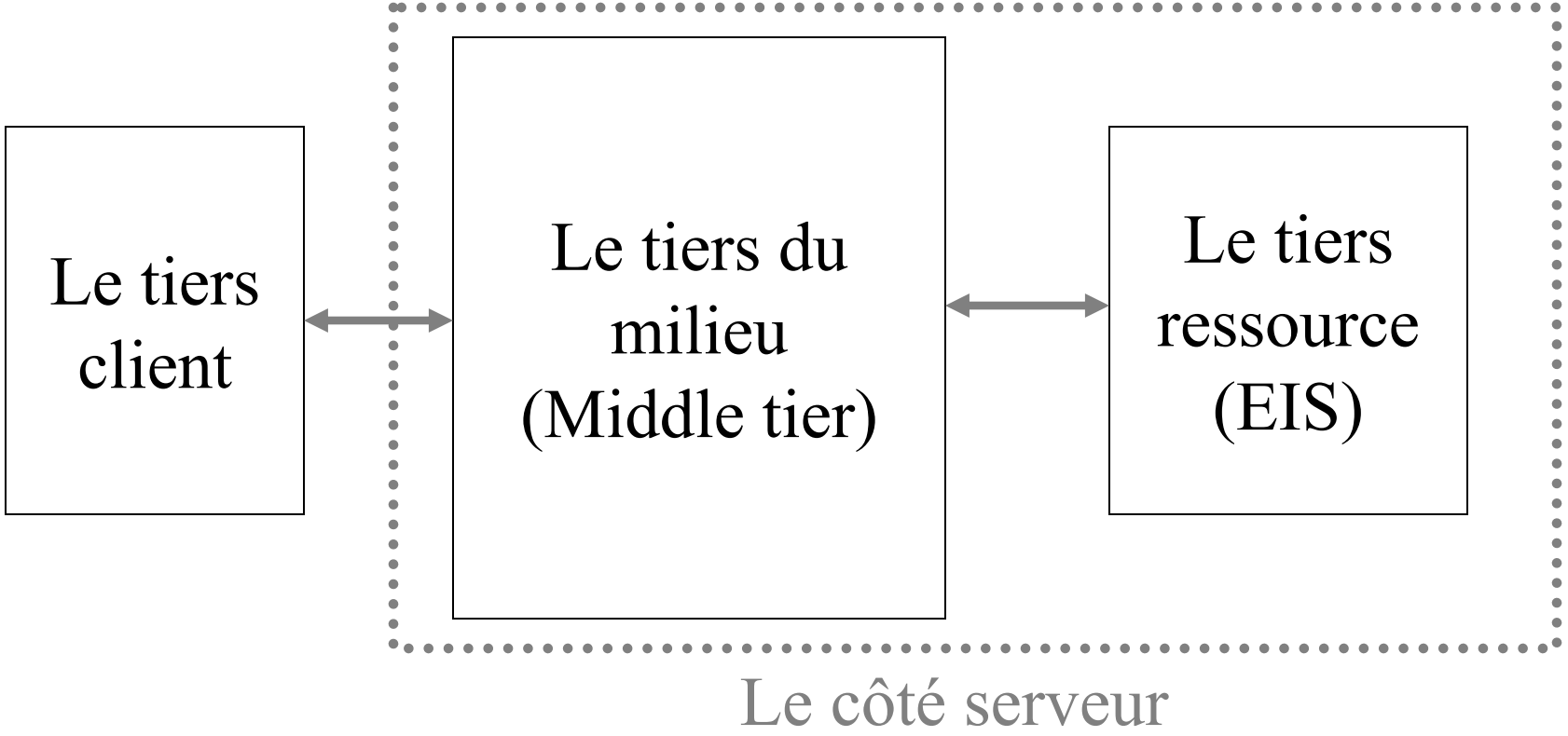
Plan

- Les architectures n-tiers pour le Web
- Plate-formes Microsoft DNA, .NET
- Plate-forme J2EE
- Les Web Services
- Exemple

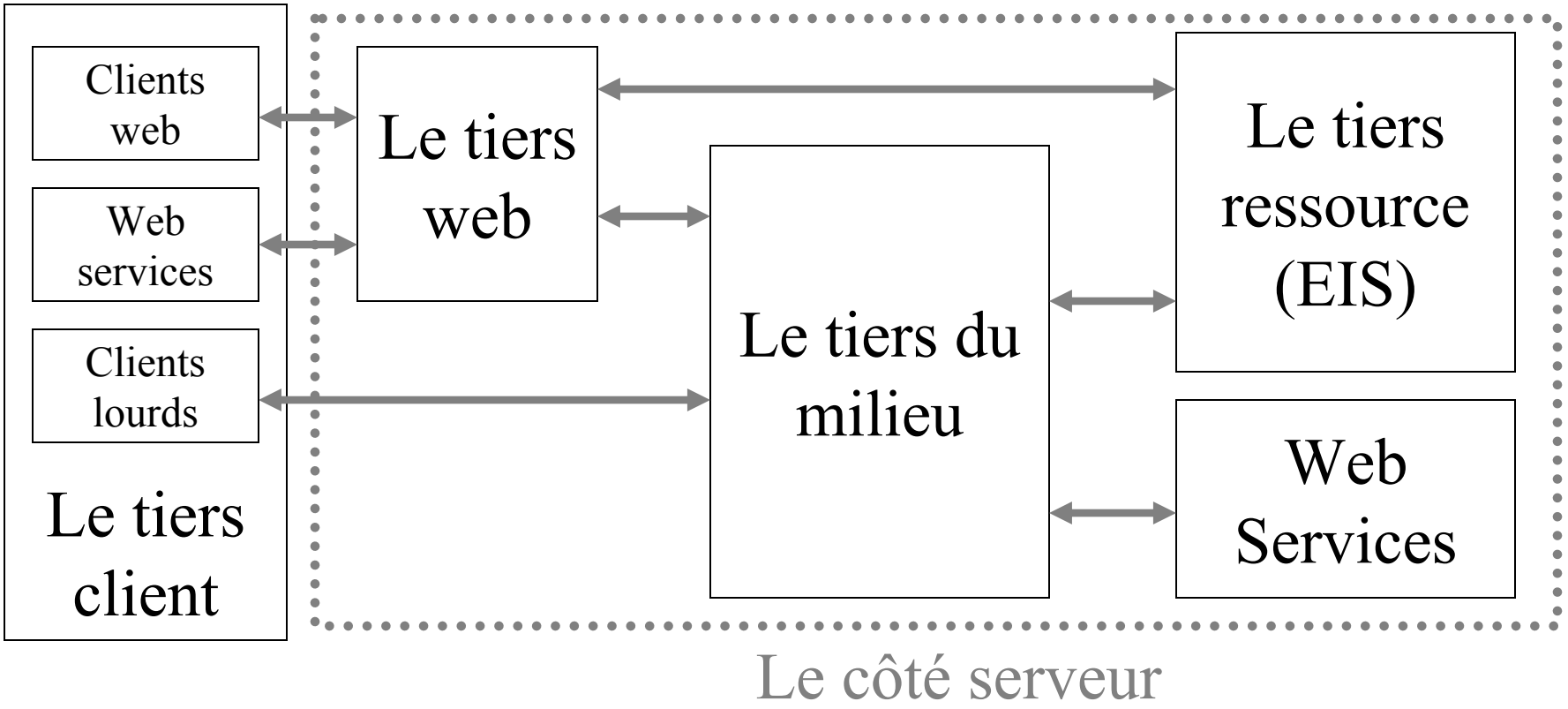
Les architectures n-tiers pour le Web

- Les architectures 3-tiers classiques
- Les architectures Web
- Le tiers client
- Le tiers Web
- Le tiers du milieu
- Le tiers ressource (EIS)

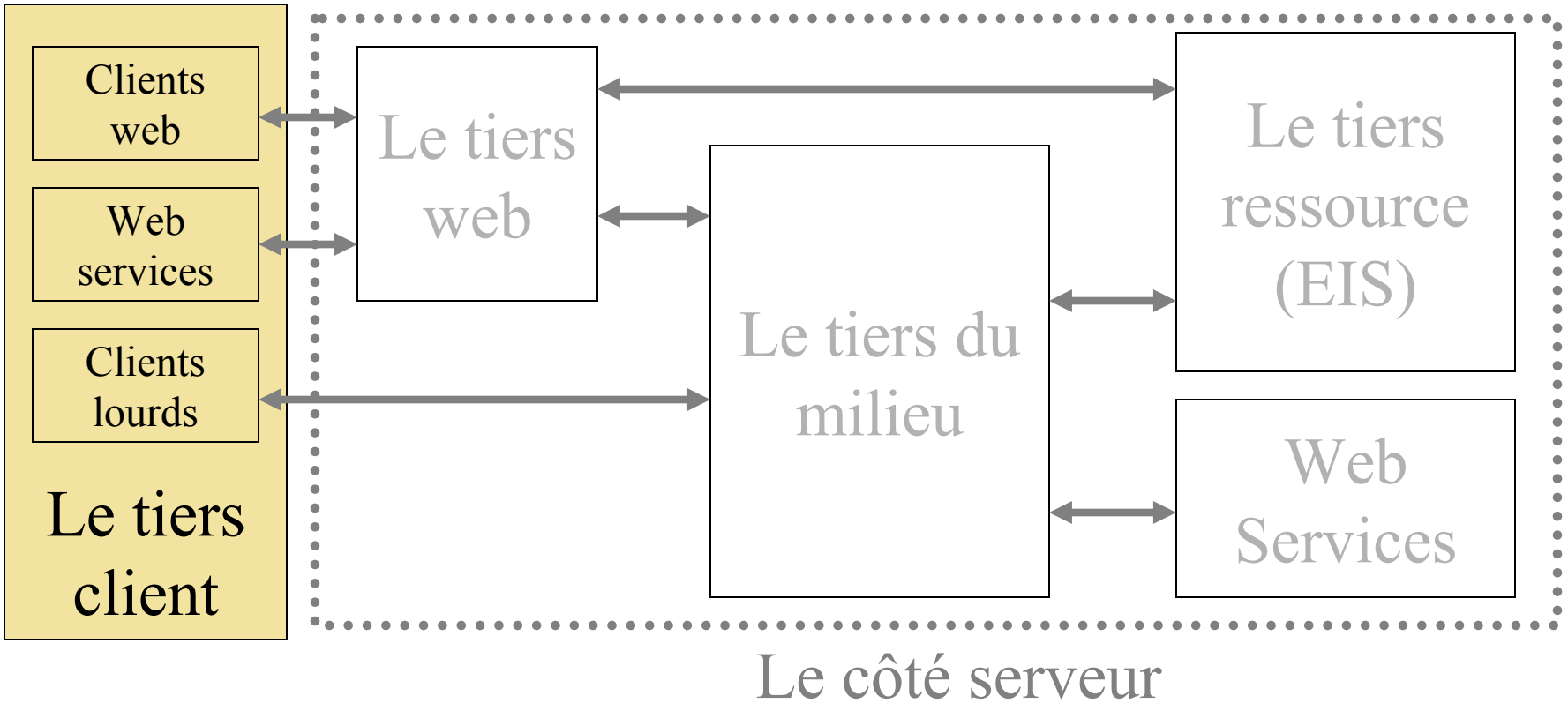
Les architectures 3-tiers classiques



Les architectures web



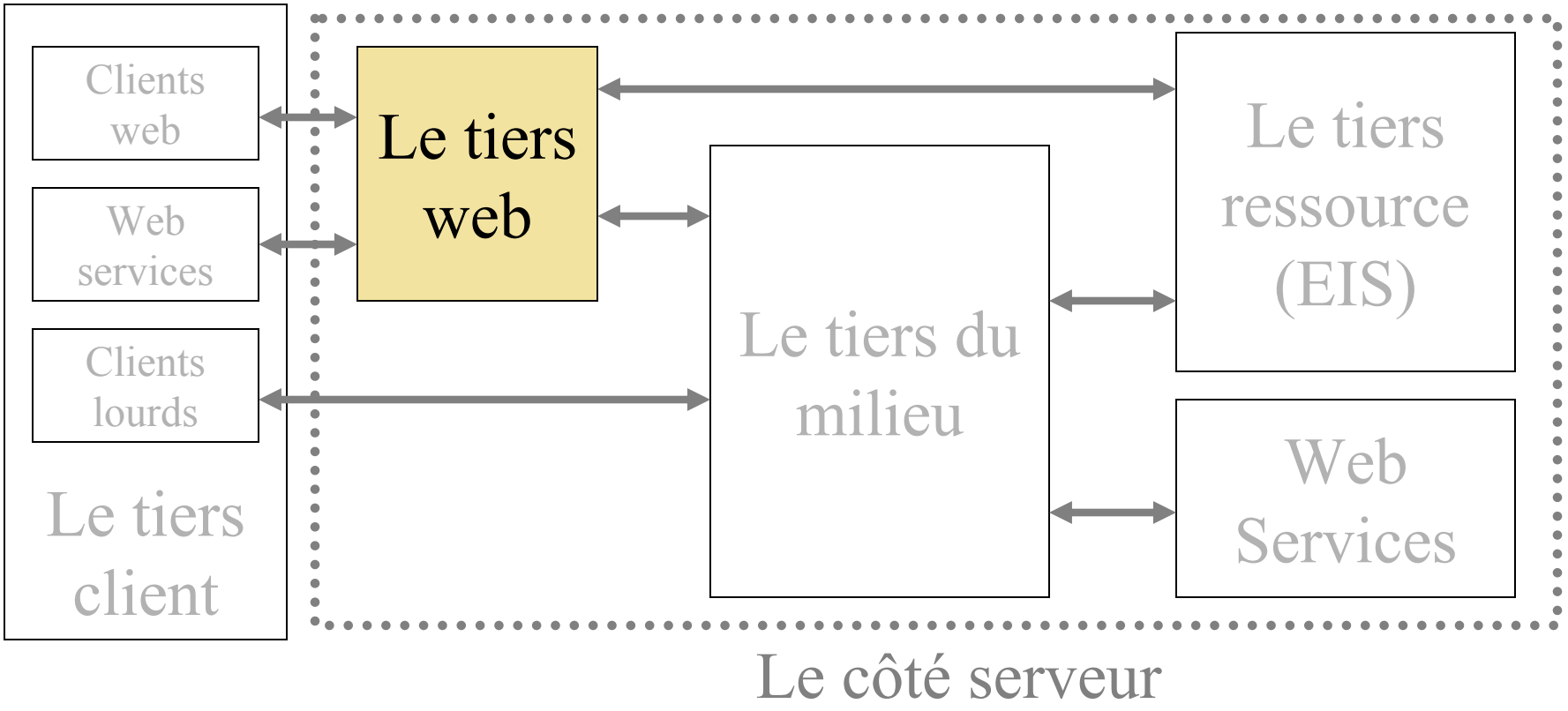
Les architectures web : le tiers client



Le tiers client

- Un web browser
  - HTTP, HTTPS / HTML, XML
- Un PDA
  - HTTP , HTTPS / WML, XML
- Un client lourd (fat client), applets, apps
  - IIOP / RMI, CORBA
  - JRMP / RMI
  - autres…
- Un Web-service
  - HTTP , HTTPS / ebXML (Elect. Business Exchange Specification , SOAP (XML)

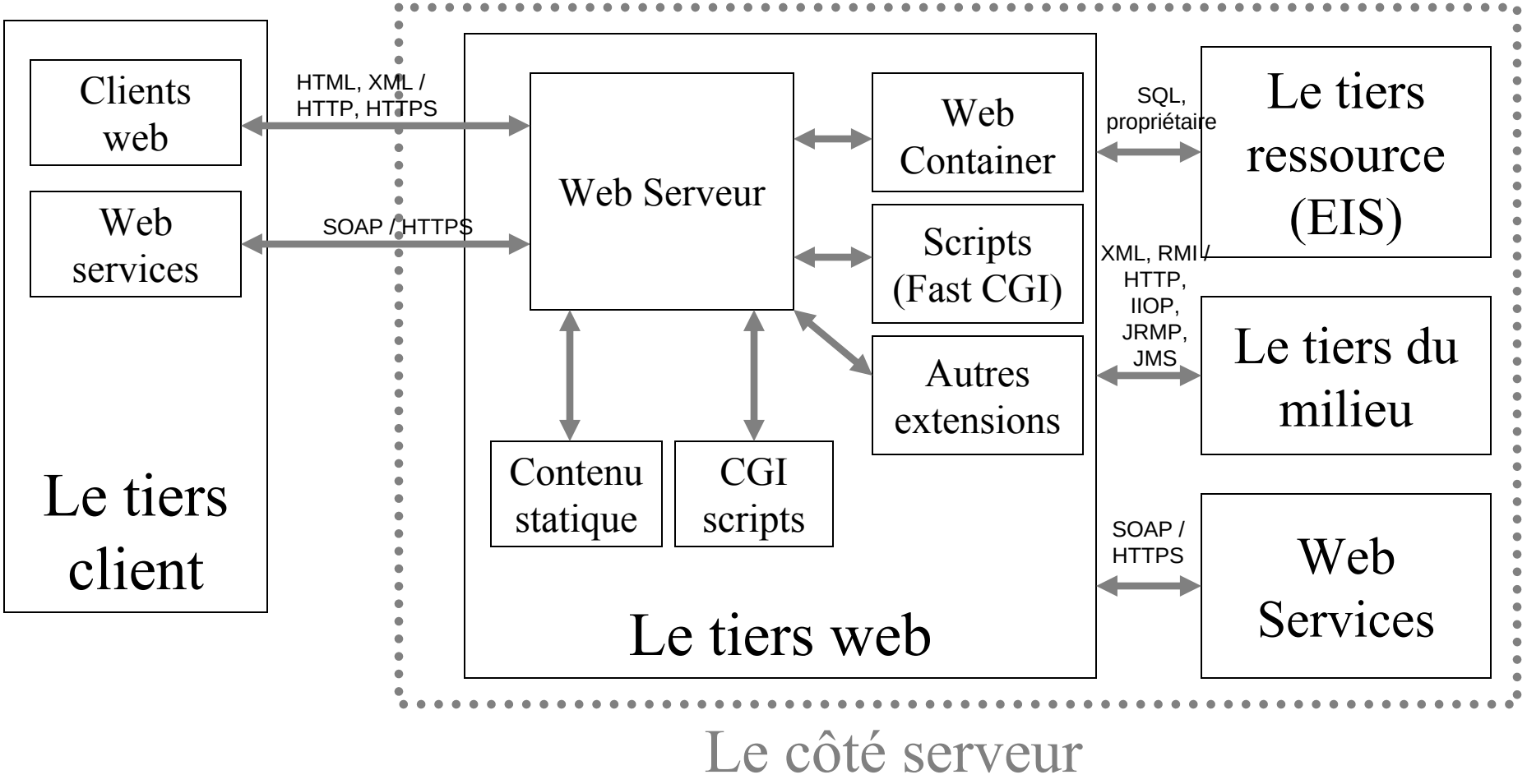
Les architectures web : le web tiers



Le tiers web : rôle

- reçoit les requêtes HTTP des clients et renvoie les réponses
- permet la séparation entre présentation (spécifique au client) et «business logic»
- génère du contenu dynamiquement
- transforme des requêtes HTTP dans un format compris par l'application
- contient la logique du flot de présentation
- identifie la session de l'utilisateur
- supporte plusieurs types de clients

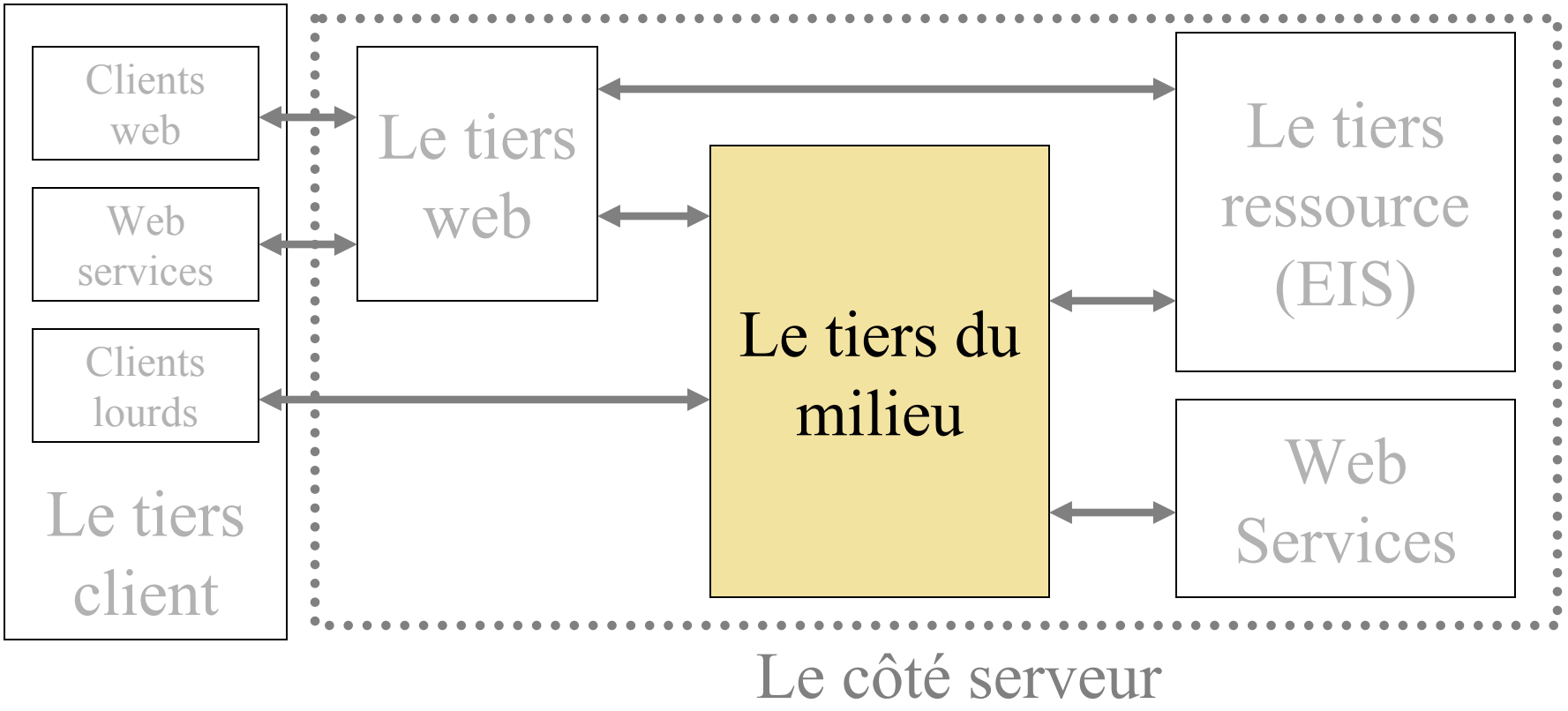
Le tiers web : architecture



Technologies utilisées dans le tiers web

- CGI/FastCGI (Common Gateway Interface)
  - Peut-être écrit en JAVA, C, C++, Perl...
- ASP (Active Server Pages)
  - Scripting interprété dans des pages HTML (Microsoft)
- Java Servlets
  - nécessite un conteneur Java
- JSP (Java Server Pages)
  - Scripting dans des pages HTML compilé en Servlet
- PHP, Python
- JavaScript (Server Side)

Les architectures web : le tiers du milieu



Le tiers du milieu : rôle

- Gestion de composants
  - fourni tous les services et outils pour gérer les composants du système et l implémentation de la «business logic» comme
    - » management de session
    - » synchrone et asynchrone notification
- Tolérance de fautes, haute disponibilité
  - Capacité de l'application de résister à une possible panne sans point unique de panne. Définie les polices de récupération.
  - 24-7
- Passage à l'échelle
  - Capacité pour le système d'accroître ses ressources matérielles pour supporter un nombre accru d'utilisateur avec un temps de réponse constant
- Balance de charge
  - Capacité d 'envoyer une requête a différents serveurs en fonction de la disponibilité des serveurs

Le tiers du milieu : rôle

- Ressources pooling
  - Protège le tiers ressource en utilisant des groupes de connections partagées entre tous les clients
- Transaction Management
  - Une transaction est une unité indivisible de travail comprenant plusieurs opérations, dont toutes ou aucune doivent être effectuées pour protéger l'intégrité des données
  - Assure les propriétés ACID des transactions (atomicité, consistance, isolation and durabilité)
- Console de management
  - Unique point de management permettant de contrôle l'ensemble du système incluant tous les serveurs
- Sécurité
  - Authentification
  - Autorisation

Propriétés des transactions : ACID

ACID (atomicity, consistency, isolation and durability)

- Atomicity
  - » This implies indivisibility; any indivisible operation (one which will either complete fully or not at all) is said to be atomic.
- Consistency
  - » A transaction must transition persistent data from one consistent state to another. If a failure occurs during processing, the data must be restored to the state it was in prior to the transaction.
- Isolation
  - » Transactions should not affect each other. A transaction in progress, not yet committed or rolled back (these terms are explained at the end of this section), must be isolated from other transactions. Although several transactions may run concurrently, it should appear to each that all the others completed before or after it; all such concurrent transactions must effectively end in sequential order.
- Durability
  - » Once a transaction has successfully committed, state changes committed by that transaction must be durable and persistent, despite any failures that occur afterwards.

Niveau d'isolation des transactions

The isolation level measures concurrent transactions' capacity to view data that have been updated, but not yet committed, by another transaction. If other transactions were allowed to read data that are as-yet uncommitted, those transactions could end up with inconsistent data were the transaction to roll back, or end up waiting unnecessarily were the transaction to commit successfully. A higher isolation level means less concurrence and a greater likelihood of performance bottlenecks, but also a decreased chance of reading inconsistent data. A good rule of thumb is to use the highest isolation level that yields an acceptable performance level. The following are common isolation levels, arranged from lowest to highest:

- ReadUncommitted
  - » Data that have been updated but not yet committed by a transaction may be read by other transactions.
- ReadCommitted
  - » Only data that have been committed by a transaction can be read by other transactions.
- RepeatableRead
  - » Only data that have been committed by a transaction can be read by other transactions, and multiple reads will yield the same result as long as the data have not been committed.
- Serializable
  - » This, the highest possible isolation level, ensures a transaction's exclusive read-write access to data. It includes the conditions of ReadCommitted and RepeatableRead and stipulates that all transactions run serially to achieve maximum data integrity. This yields the slowest performance and least concurrency.

Type de serveurs : Web Information Serveurs

- Web Information Serveurs
  - A la frontière du tiers web et du tiers du milieu
  - Pas de transactions
  - Serveurs sans états
  - Utilise des templates et un langage de script pour générer les pages HTML dynamiquement tout en accédant le tiers ressource
  - Exemples
    - IIS + ASP
    - Web serveur + PHP, Python, CGI

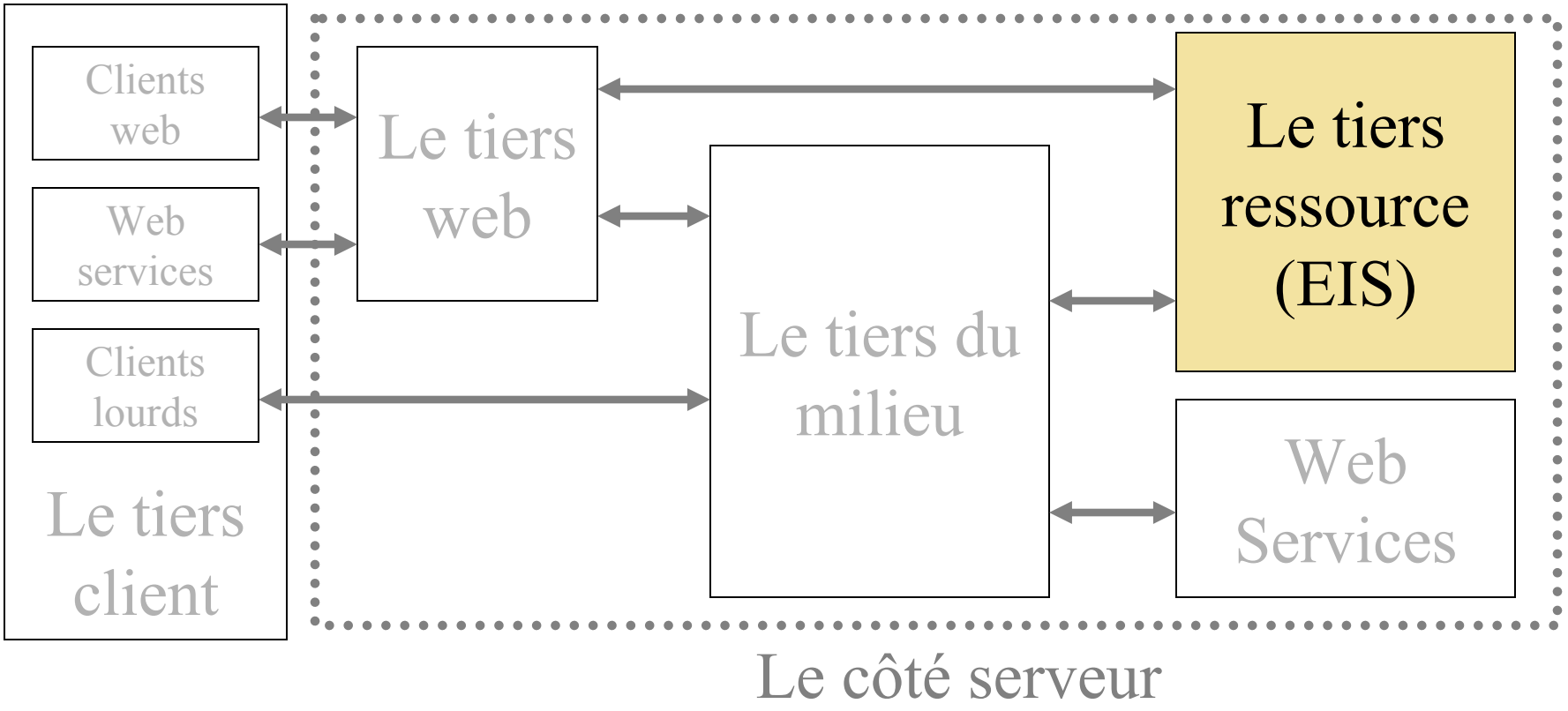
Type de serveurs : Serveurs de composants

- Serveurs de composants
  - Permet l'accès au tiers ressource
  - Gère les transactions
  - Serveurs sans états
  - Sont maintenant au cœur des serveurs d'applications
  - Exemples
    - Microsoft Transaction Server (MTS, dans .net)
    - Sybase Jaguar (dans Sybase EAServer)
    - IBM Component broker (dans WebSphere)

Type de serveurs : Serveurs d'applications

- Serveur d'applications
  - Environnement complet de développement coté serveur
  - Comprends toujours un serveur de composants
  - Serveurs avec états
  - Supporte «business logic» décrite à l 'aide d 'objets, de règles et de composants
  - Exemples
    - Microsoft .net Enterprise Servers
    - J2EE Serveurs : IBM WebSphere, BEA WebLogic, JBoss
    - ORB Corba Servers : Borland VisiBroker, IONA ORBacus
    - Notez que les serveurs d'applications Corba complet intègrent J2EE.
    - Pour une comparaison MTS-EJB regardez
      - » http://www.execpc.com/~gopalan/misc/ejbmts/ejbmtscomp.html

Les architectures web : le tiers ressource



Le tiers ressource (EIS : Enterprise Information Systems)

- Base de données (databases)
  - JDO, SQL/J, JDBC, ADO.NET
- Anciens systèmes (legacy systems)
  - J2EE Connector, protocoles propriétaires
- ERP (Enterprise Resource Planning)
  - J2EE Connector, protocoles propriétaires
- EAI (Enterprise Application Integration)
  - J2EE Connector, protocoles propriétaires

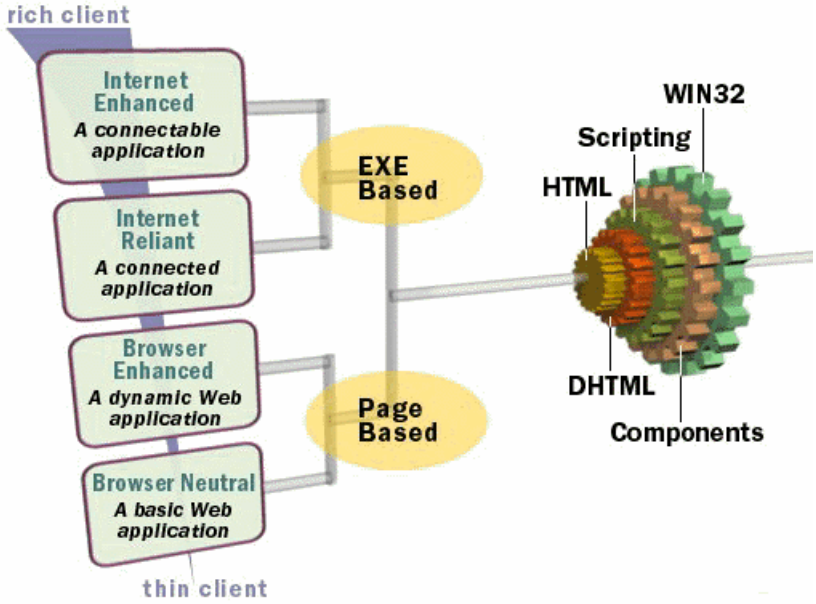
Plate-formes Microsoft DNA, .NET

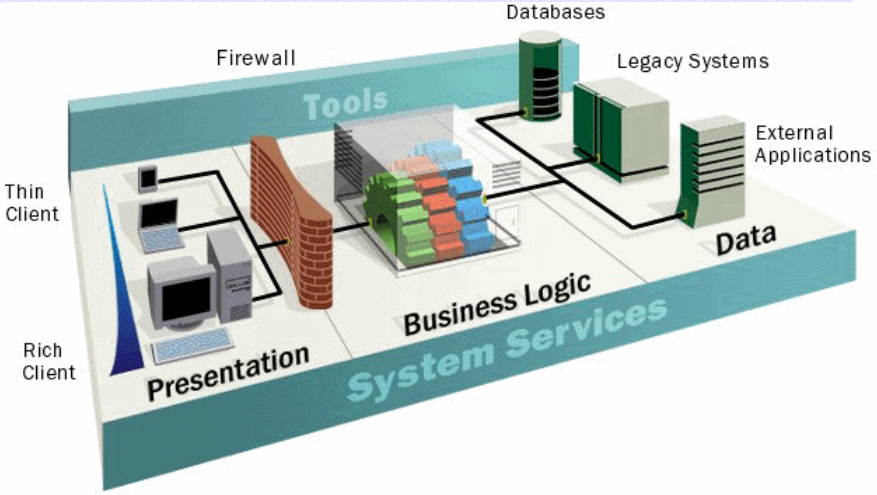
Deux mondes ?

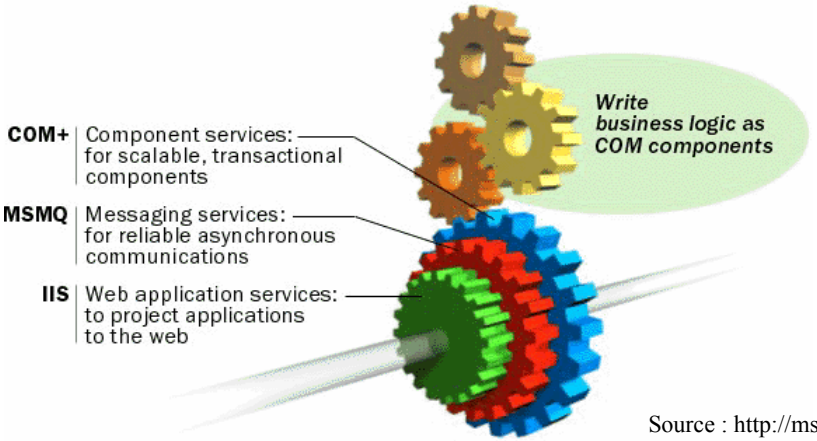
Microsoft DNA (Distributed interNet Architecture)

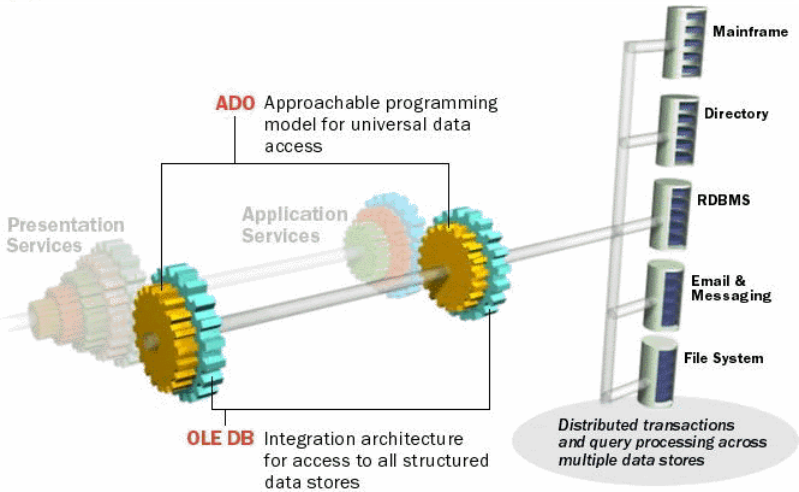
- Méthodologie, software pattern
  - apporte une solution à un ensemble de problème rencontrés dans les applications distribuées
- Ensemble de technologies Microsoft
  - Microsoft apporte les outils logiciels pour implémenter cette méthodologie
    - Présentation : Internet Explorer
    - Moteur de rendu : IIS (Internet Information Server)
    - Rendu<->Business : ASP (Active Server Pages), Scripting (VBScript)
    - Communication entre composants : COM, MSMQ, COM+
    - Composants : COM (Common Object Model) avec/sans MTS
    - Business<->Data : ActiveX Data Objects (More COM), ODBC
    - Accès aux données : OLEDB, Universal Data Access and ADSI (ADO)
    - Persistance : SQL Server, Exchange, Active Directory et NTFS

Microsoft DNA (Distributed interNet Architecture)





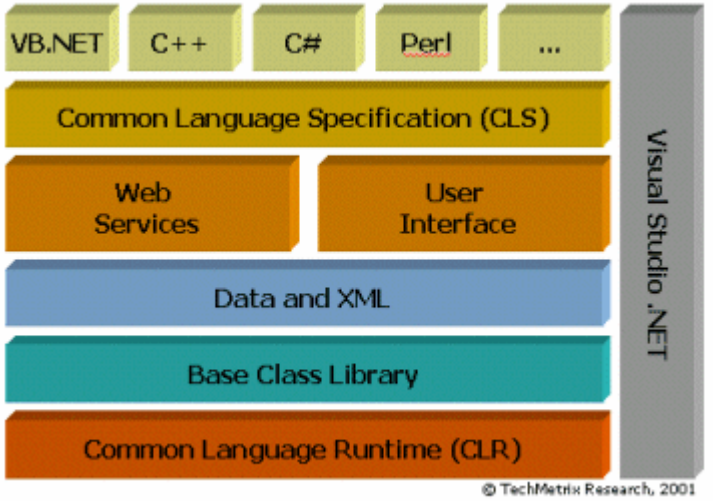




Source : http://msdn.microsoft.com/

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)



CLR : Common Language Runtime

- C'est le moteur d'exécution pour les application du .NET Framework
  - Equivalent a une JVM du monde Java
  - Code management (chargement et exécution)
  - Isolement de la mémoire par application
  - Vérification des types
  - Conversion du MSIL en code natif (pas d'interprétation)
  - Accès aux méta-data (information sur les types)
  - Gestion de la mémoire pour les objets (garbage collection)
  - Contrôle des accès du code (Security Manager)
  - Gestion des exceptions, incluant les exceptions inter-langages
  - Interopération entre managed code, COM objects, and pre-existing DLLs (unmanaged code and data)
  - Services aux développeurs (profiling, debugging, etc...)

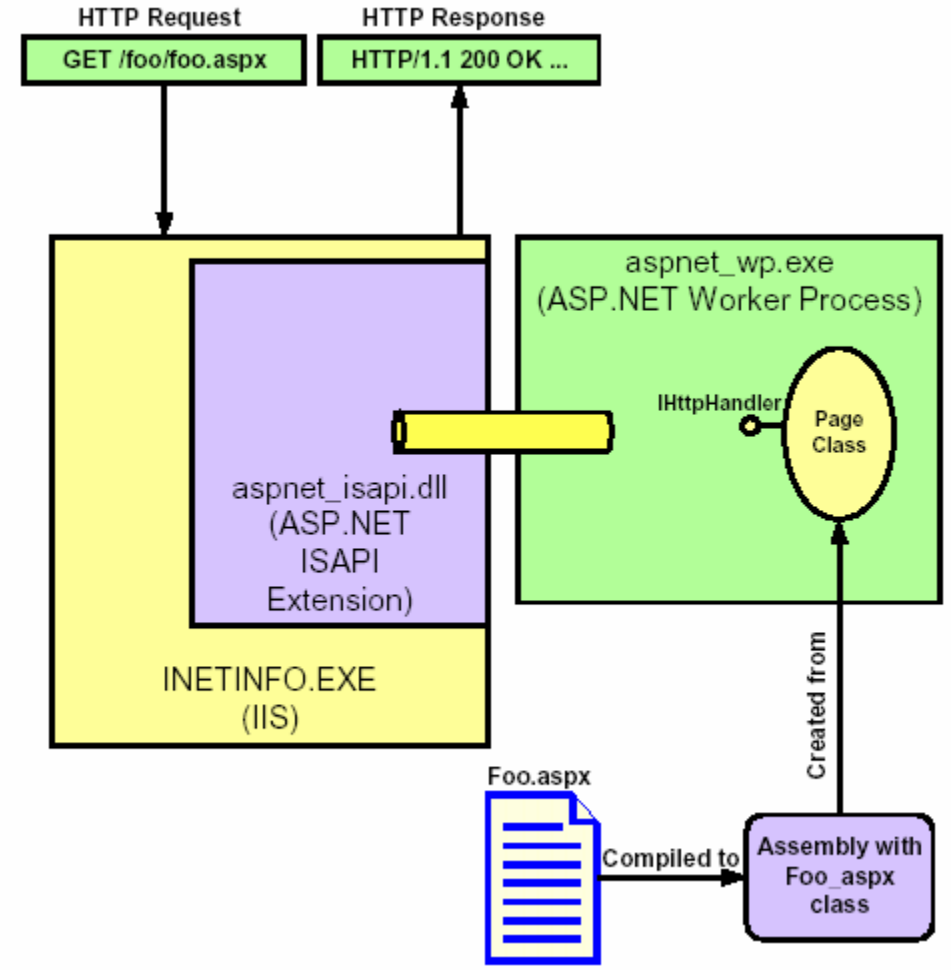
CTS / CLS / MSIL

- The Common Type System
  - » Type system, built into the Common Language Runtime, that supports the types and operations found in most programming languages. The common type system supports the complete implementation of a wide range of programming languages.
- The Common Language Specification
  - » Set of constructs and constraints that serves as a guide for library writers and compiler writers. It allows libraries to be fully usable from any language supporting the CLS, and for those languages to integrate with each other. The Common Language Specification is a subset of the common type system. The Common Language Specification is also important to application developers who are writing code that will be used by other developers. When developers design publicly accessible APIs following the rules of the CLS, those APIs are easily used from all other programming languages that target the Common Language Runtime.
- The Microsoft Intermediate Language
  - » CPU-independent instruction set into which .NET Framework programs are compiled. It contains instructions for loading, storing, initializing, and calling methods on objects. Combined with metadata and the common type system, MSIL allows for true cross-language integration.
  - » **Prior to execution, MSIL is converted to machine code. It is not interpreted.**

ASP.NET / WebForms

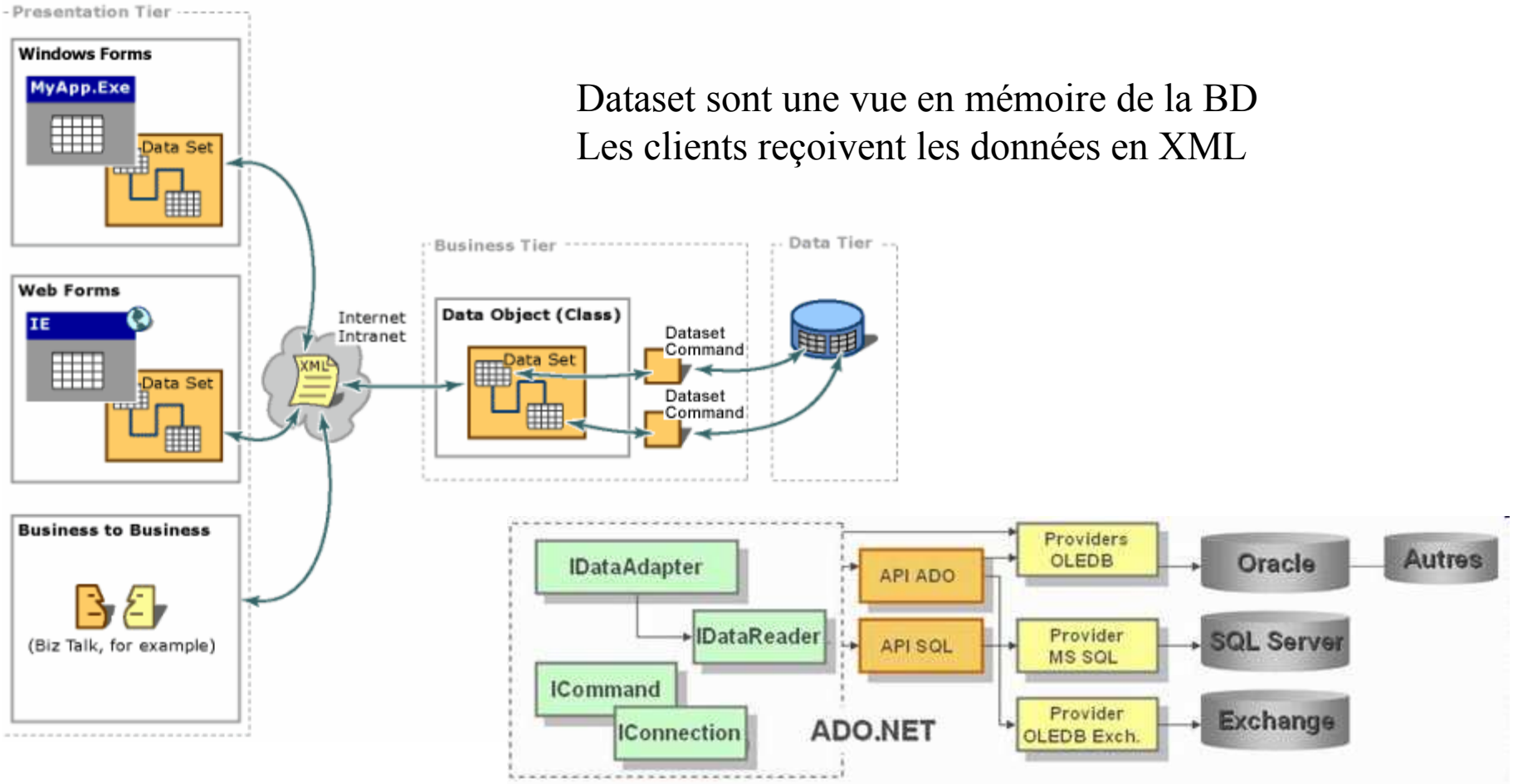
- ASP.NET est une abstraction de HTTP
  - 3 abstractions : context, handler, module
  - pages compilées et exécutées dans le CLR
    - modèle de développement basé sur les WebForms qui permet de développer une interface graphique Web comme une interface graphique VB
  - sépare traitements et présentation
    - 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.

ASP.NET architecture



Microsoft ADO.NET

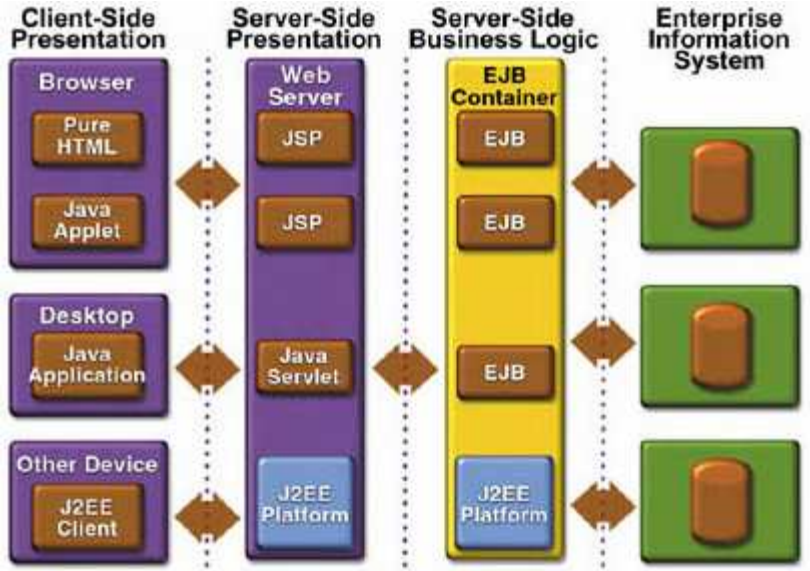
Dataset sont une vue en mémoire de la BD
Les clients reçoivent les données en XML



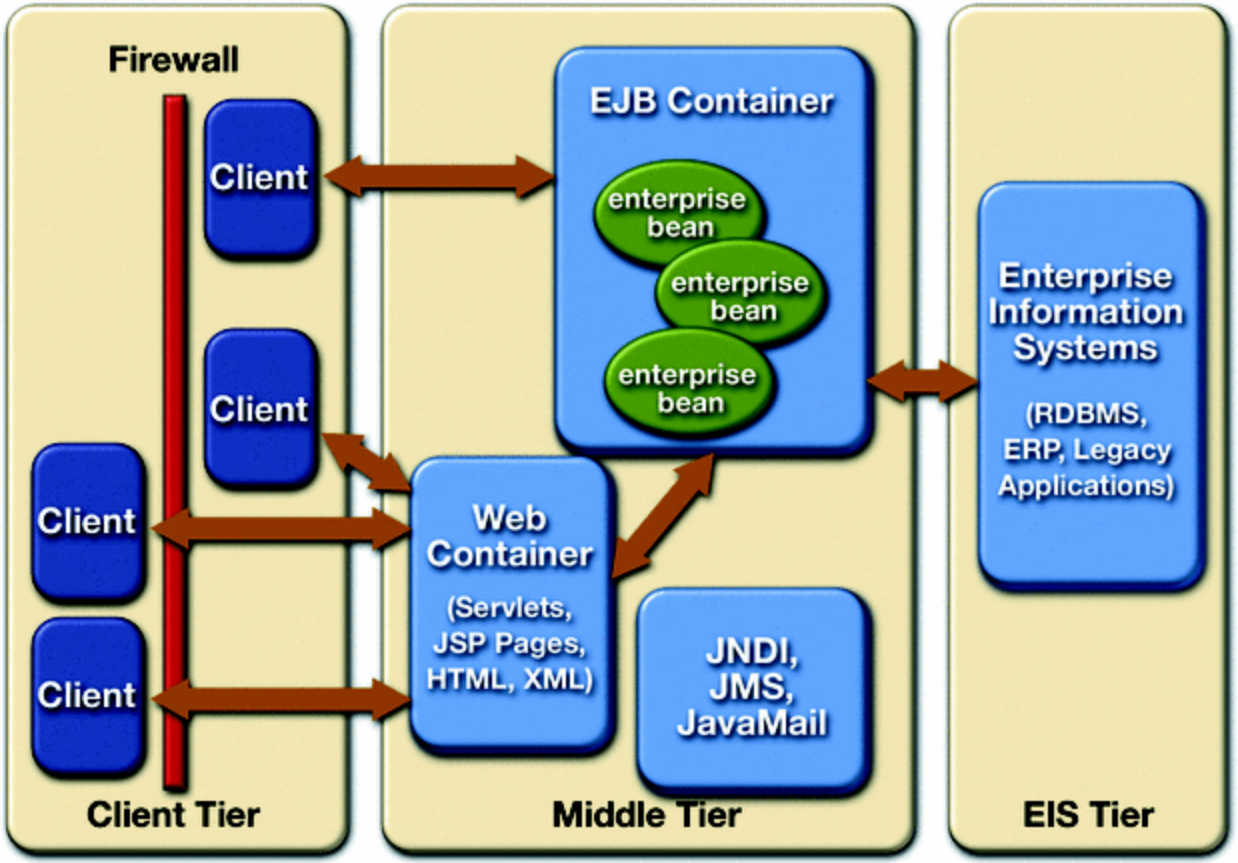
Source : msdn.microsoft.com/vstudio/nextgen/technology/adoplus.asp

La plate-forme J2EE

- 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 d'un serveur J2EE



Source : http://java.sun.com/blueprints/guidelines

APIs de la plate-forme J2EE

- Enterprise JavaBeans Technology (EJB) 2.0
  - Brique de base pour construire l'application
  - 3 types de beans
    - session beans
    - entity beans
    - message-driven beans
- JDBC 4.0 API
  - Permet l'exécution de commandes SQL
  - 2 parties dans l'API
    - la partie application pour accéder à la BD
    - la partie fournisseur de services pour attacher le driver JDBC à la plate-forme J2EE
- Java Data Objects (JDO)

APIs de la plate-forme J2EE

- Java Servlet Technology 2.4
  - introduit un modèle Requête/Réponse dans Java et étends la capacité d'un serveur HTTP
- JavaServer Pages (JSP) Technology 2.0
  - mélange texte statique (HTML, XML) avec constructions dynamiques de contenue à l'aide de tags JSP
- Java Message Service (JMS) 1.1
  - introduit un modèle de messages entre composants (MOM : Messages Oriented Middleware)
  - permet des communications asynchrones, fiable et indépendantes entre composants

APIs de la plate-forme J2EE

- Java Transaction API (JTA) 1.0.1
  - fournit les services nécessaires pour contrôler le niveau d'isolation des transactions
- JavaMail Technology 1.3.1
  - Permet l'envoie d'emails. 2 parties dans l'API
    - la partie application pour contrôle l'envoie d'emails
    - la partie fournisseur de services emails
- JavaBeans Activation Framework(JAF)1.0.2
  - Service de découverte et d'encapsulation de composants a l 'aide de JavaBean

APIs de la plate-forme J2EE

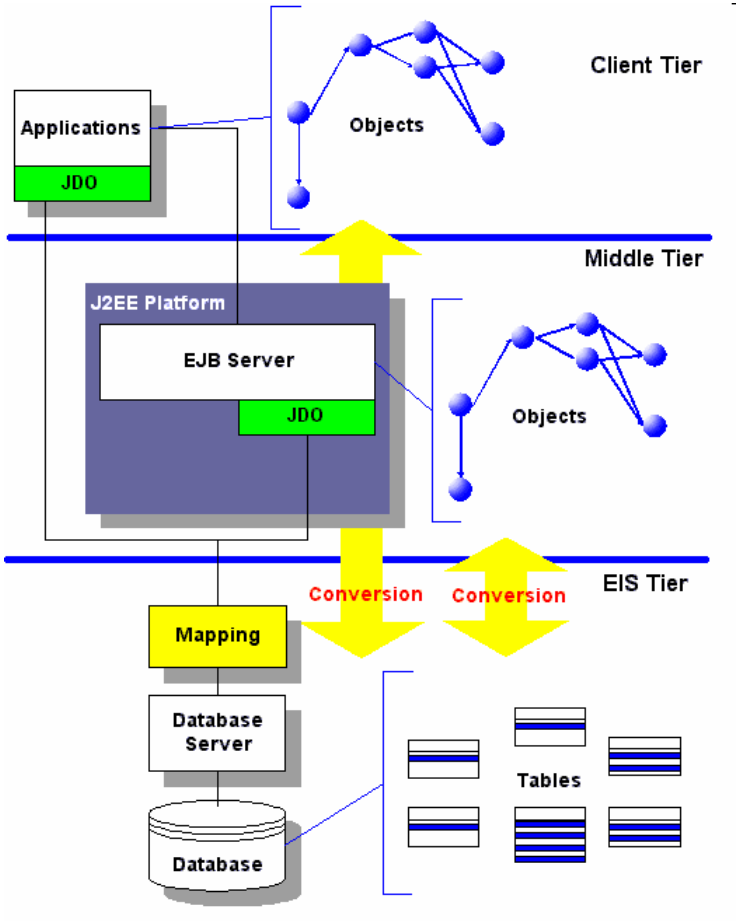
- Java API for XML (JAXP) 1.2.4
  - Standard API pour parser et transformer les données XML (DOM/SAX/XSTL/TrAX)
- J2EE Connector API 1.5
  - Permet la création d'adaptateurs de ressources pour permettre l'accès aux systèmes EIS
- Java Authentication and Authorization Service (JAAS) 1.0
  - Permet la gestion de la securite dans les applications J2EE

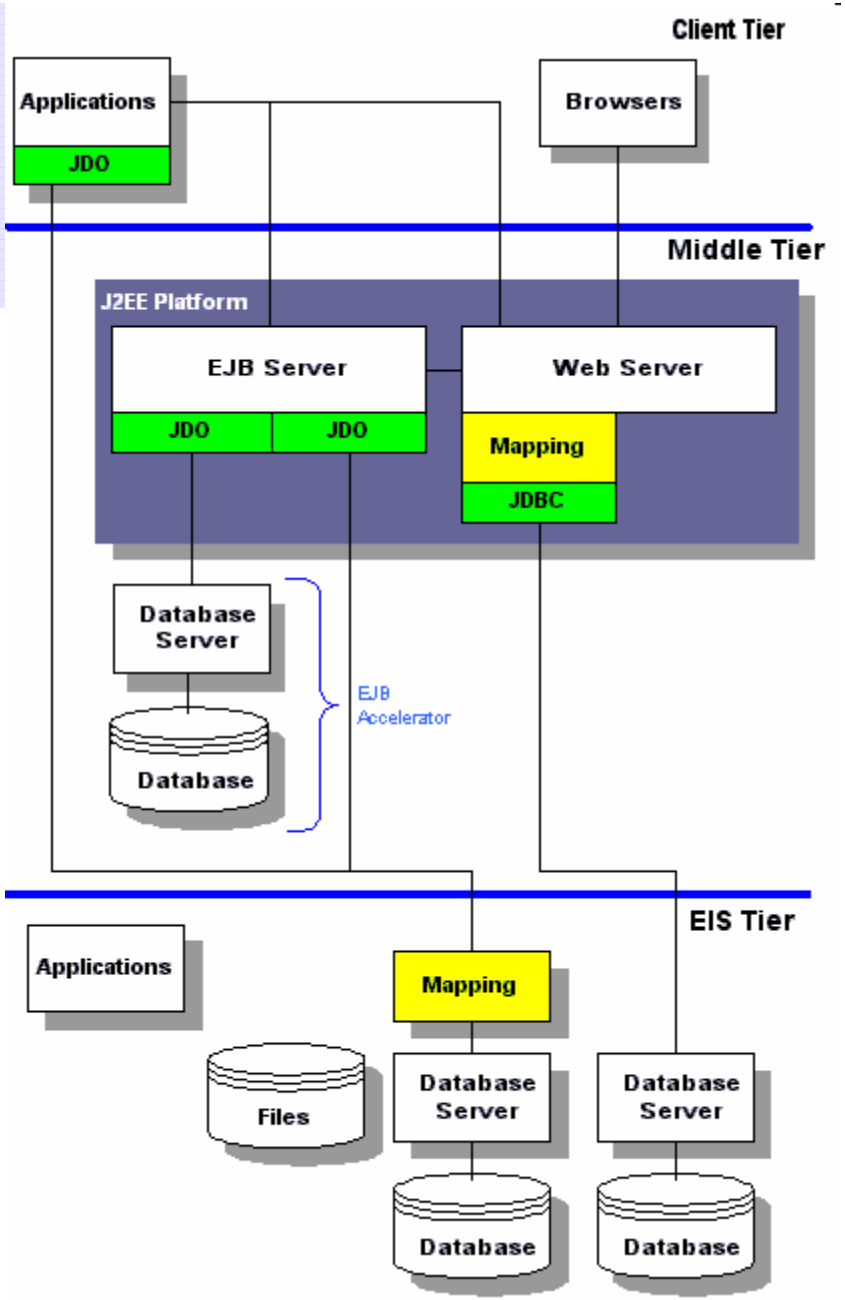
APIs de la plate-forme J2EE : WebServices = WSDP 1.3

Java Web Services Developer Pack

- Java API for XML Binding (JAXB, 1.0.2)
- Java APIs for XML Messaging 1.0 (JAXM)
- Java API for XML Registries 1.0.5 (JAXR)
- Java API for XML-based RPC 1.1 (JAX-RPC)
- JavaServer Faces (JSF)
- XML WS Security 1.0
- JAXP Java API for XML processing 1.2.4
- SOAP with attachments API 1.2.4 (SAAJ)
- Java Server Pages Std. Tag Library 1.1 (JSTL)

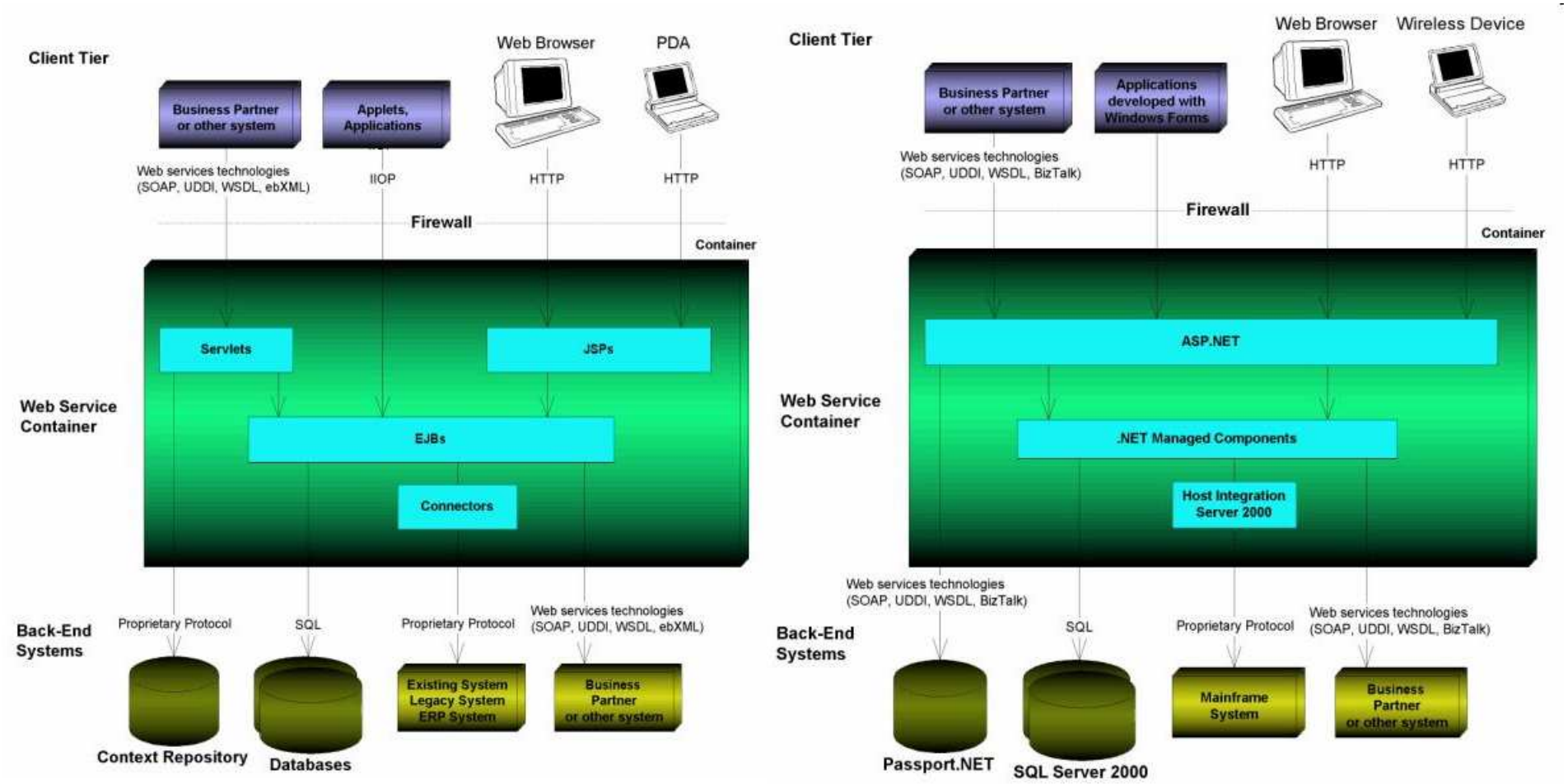
JDO / JDBC





Source : http://www.java-application-servers.com/articles/app_servers.html

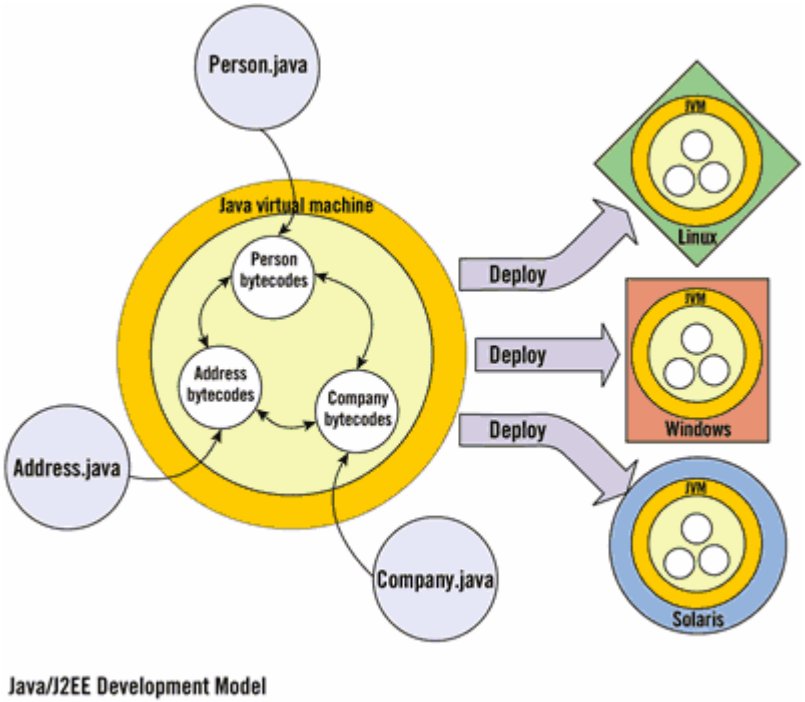
Comparaison de l'architecture J2EE et .NET



Source : http://www.theserverside.com/resources/article.jsp?l=J2EE-vs-DOTNET

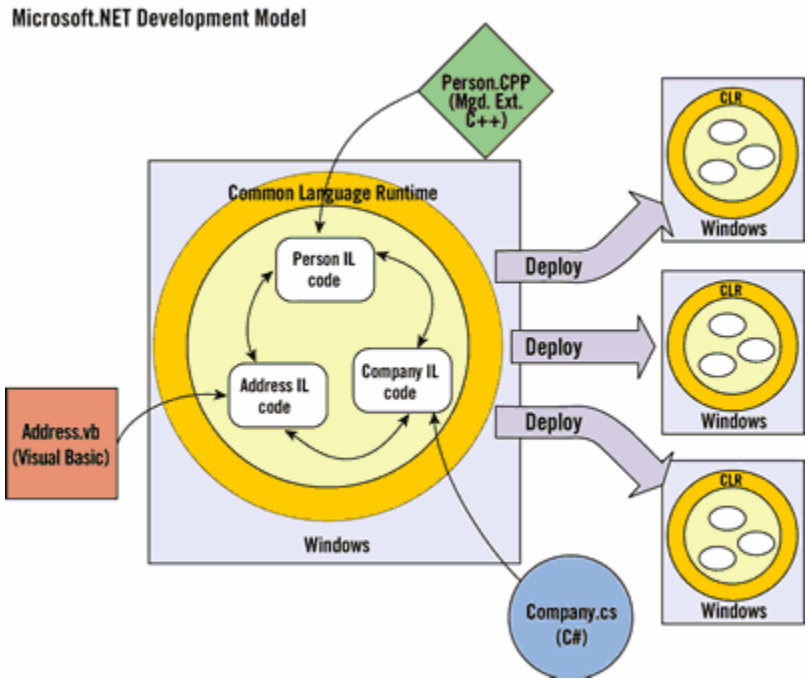
Comparaison du modèle de développement de J2EE et .NET

Un langage
Plusieurs plate-formes



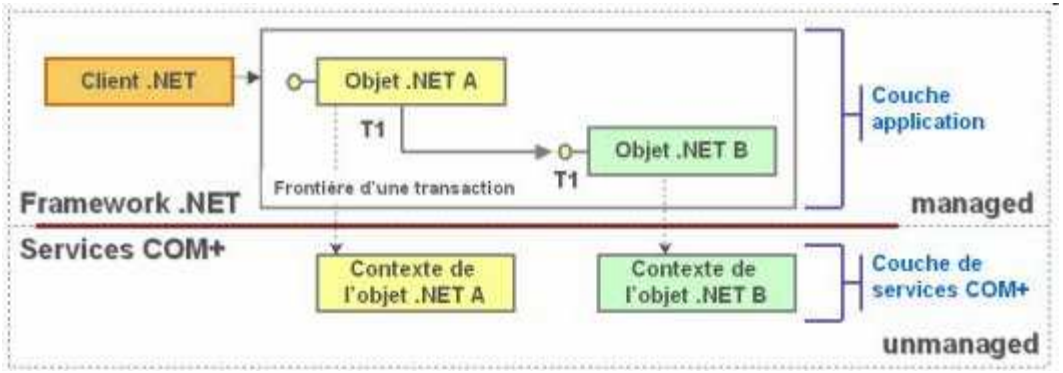
Java/J2EE Development Model

Plusieurs langages
Une plate-forme

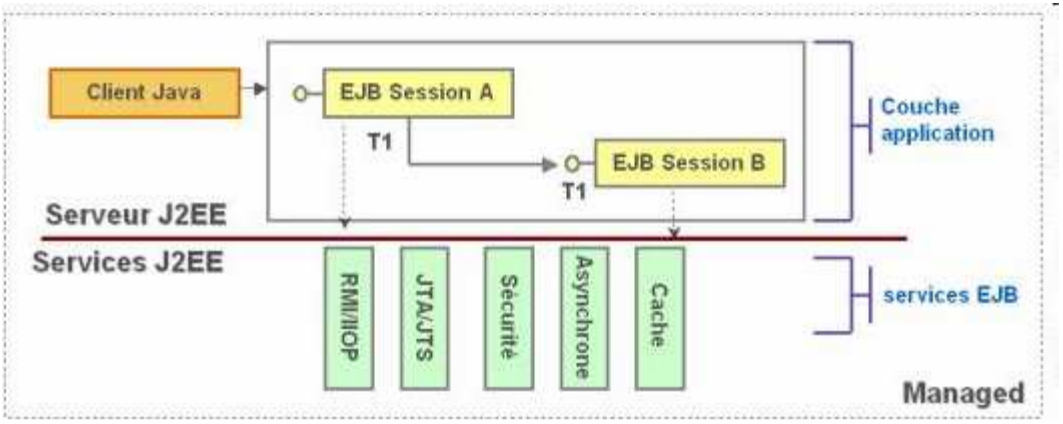


Source : http://www.sdmagazine.com/documents/s=733/sdm0103a/0103a.htm

Gestion des composants : managed / unmanaged



.NET propose le même ensemble de services sous l'appellation de ServicedComponent. Le conteneur utilisé dans le Framework est COM+ COM+ fonctionne dans un environnement non managé avec une gestion de type différente de celle de .NET (Common Type System)



Dans J2EE les transactions sont gérées en Java à l'aide des API JTA/JTS et l'ensemble des composants s'exécutent dans un seul et même environnement managé.

Source : http://www.dotnetguru.org/articles/architecturedotnet.htm

Tableau comparatif des plate-formes .NET et J2EE

	Microsoft. NET	J2EE	différences essentielles
Langage	C#, Multi-Langage	Java	C# a certains des JavaBeans et ajoute les metadata tags. L'intégration dans la syntaxe est différente. J2EE est plate-forme indépendant mais langage spécifique, .NET est langage indépendant mais plate-forme spécifique.
Services	BCL	Java core API	Similaire services
Présentation	ASP.NET	Servlet JSP	ASP.NET utilise tout les langages supportes dans .NET et est compile en code natif par le CLR. JSPs utilisent Java code (snippets, ou JavaBean références), compile en bytecodes.
Interprète	CLR	JVM	CLR permet a du code de plusieurs langages d'utiliser un ensemble de composants partages.
GUI composants	Win Forms Web Forms	Swing	Composants Web similaire ne sont pas disponible en Java. WinForms et WebForms sont complètement intègre a VisualStudio .net
DB accès	ADO.NET	JDBC, JDO, SQL/J	ADO.NET est construit a partir d'une architecture XML
WebServices	oui	oui	.NET web services supposent un model de message base sur SOAP tandis que J2EE laisse le choix au developpeur.
Implicit middleware	oui	oui	
Technologie	Produit	Standard	J2EE est une specification, .NET est une strategie de produits

Définition des Web Services

- Collection de fonctions packagées dans une même entité et publiées pour etre utilisée sur le reseau
  - Interface XML sur un ensemble de services
  - Evolution naturelle des systèmes distribués
  - le fondement des web services est l'utilisation de messages XML transportés sur des protocoles standard comme HTTP
  - « Old technologies wearing a new hat »
- Protocoles simples et standards permettant une utilisation universelle
  - UDDI, WSDL, SOAP : Publish, Bind, Find, Invoke
- Interface au-dessus des architectures n-tiers existante (.NET, J2EE)

Motivation des Web Services

- L'intégration de modules distribués est très difficile à cause de l'hétérogénéité des systèmes
  - CORBA était une solution mais sa complexité a freiné son développement
  - Microsoft avait sa propre technologie COM
- Les Web Services sont simples basés sur XML
- Les protocoles sont simples et standards
- Une énorme synergie est née derrière eux

Technologies des Web Services : WSDL

- WSDL (Web Services Description Language)
  - format de représentation des interfaces de Service Web en XML
  - spécifie le prototype des services (signatures des méthodes, différents types utilisés)
  - WSDL est la représentation XML du langage IDL (Interface Definition Language) ou MIDL (Microsoft) de description des interfaces
  - rôles
    - sert de référence à la génération de proxies
    - assure que le couplage entre le client et le serveur se fait via les interfaces
    - toute évolution dans l'implémentation d'un service n'aura aucun impact sur le client tant que les interfaces restent inchangées et compatibles.

Exemple de WSDL

```
<?xml version="1.0"?>
<definitions name="StockQuote" targetNamespace="http://example.com/stockquote.wsdl" xmlns:tns="http://example.com/stockquote.wsdl"
      xmlns:xsd1="http://example.com/stockquote.xsd" xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" xmlns="http://schemas.xmlsoap.org/wsdl/">
   <types>
      <schema targetNamespace="http://example.com/stockquote.xsd" xmlns="http://www.w3.org/2000/10/XMLSchema">
         <element name="TradePriceRequest">
            <complexType>
               <all>
                  <element name="tickerSymbol" type="string"/>
               </all>
            </complexType>
         </element>
         <element name="TradePrice">
            <complexType>
               <all>
                  <element name="price" type="float"/>
               </all>
            </complexType>
         </element>
      </schema>
   </types>
   <message name="GetLastTradePriceInput">
      <part name="body" element="xsd1:TradePriceRequest"/>
   </message>
   <message name="GetLastTradePriceOutput">
      <part name="body" element="xsd1:TradePrice"/>
   </message>
   <portType name="StockQuotePortType">
      <operation name="GetLastTradePrice">
         <input message="tns:GetLastTradePriceInput"/>
         <output message="tns:GetLastTradePriceOutput"/>
      </operation>
   </portType>
   <binding name="StockQuoteSoapBinding" type="tns:StockQuotePortType">
      <soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
      <operation name="GetLastTradePrice">
         <soap:operation soapAction="http://example.com/GetLastTradePrice"/>
         <input>
            <soap:body use="literal"/>
         </input>
         <output>
            <soap:body use="literal"/>
         </output>
      </operation>
   </binding>
   <service name="StockQuoteService">
      <documentation>My first service</documentation>
      <port name="StockQuotePort" binding="tns:StockQuoteBinding">
         <soap:address location="http://example.com/stockquote"/>
      </port>
   </service>
</definitions>
```

The WSDL definition shown in the example below contains the following key pieces of information:

- A description/format of the messages that can be passed (via embedded XML Schema Definitions) within the <types> and <message> elements

- The semantics of the message passing (e.g. Request-only, request-response, response-only) within the <portType> element

- A specified encoding (various encodings over a specified transport such as HTTP, HTTPS, or SMTP) within the <binding> element
The endpoint for the service (a URL) within the <service> element

Source : http://dcb.sun.com/practices/webservices/overviews/overview_wsdl.jsp

Technologies des Web Services : UDDI

- UDDI (Universal Description, Discovery, and Integration)
  - fournie un annuaire permettant de retrouver des web services sur le même principe que les pages jaunes
  - UDDI implique que les différents fournisseurs de web services parviennent à s'entendre sur la définition de critères communs et de catégories "métier" bien déterminées
  - mise en œuvre dans le cadre de places de marché collaboratives ou dans des domaines très spécifiques
  - Microsoft et IBM proposent des solutions plus légères à mettre en œuvre telles que WS-Inspection (Web Services Inspection Language)

Pb.: - Lack of moderation in public UDDI repository (old WS)

- Inadequate QoS, sec. Information (NFP for communication)
- Business model (still need to negociate, agree on contract …)

SOAP : Simple Object Access Protocol

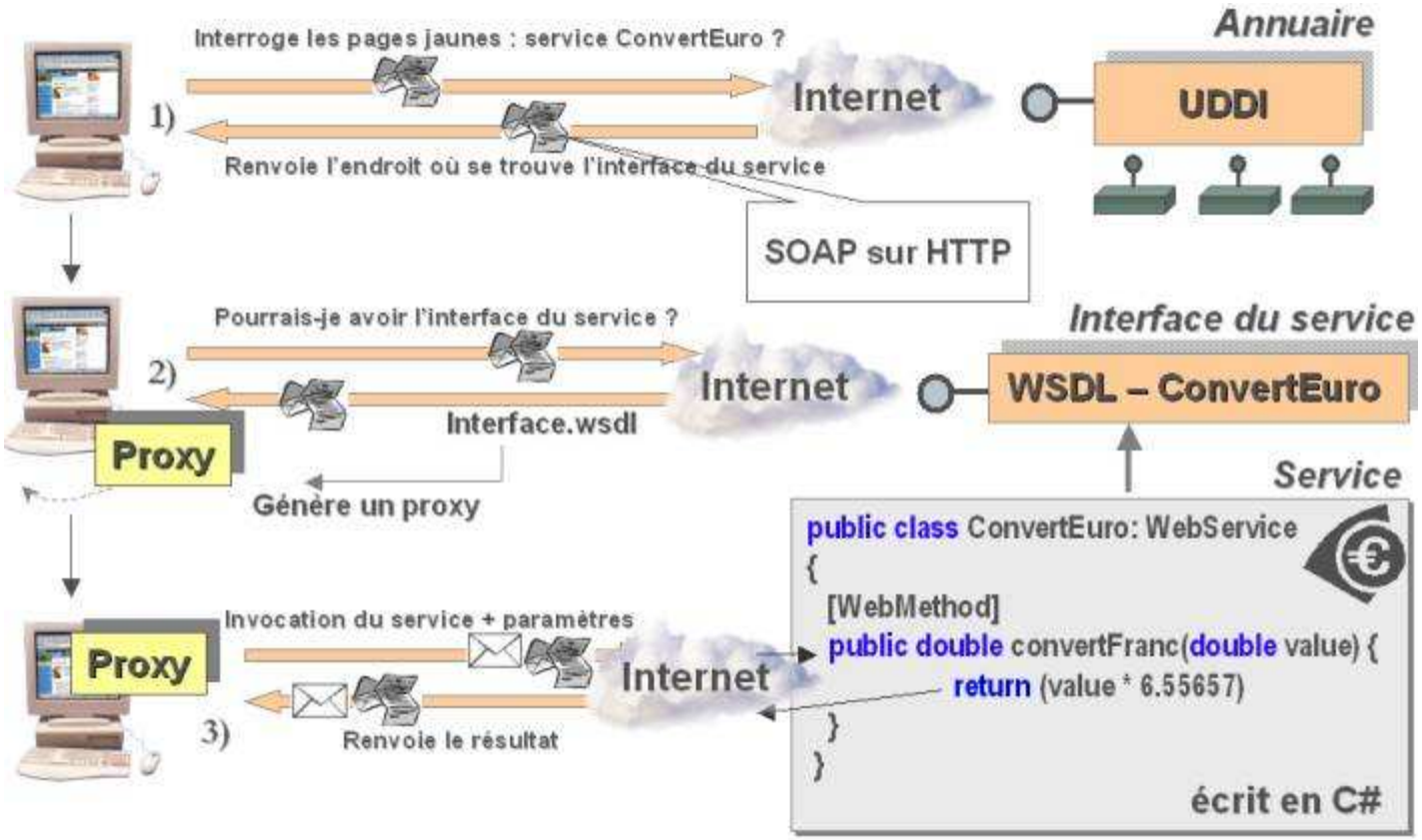
- SOAP est un protocole minimal pour faire du RPC basé sur XML
- SOAP est indépendant d'un protocole de transport particulier
- C'est le IIOP de Corba ou le JRMP de RMI
- Structure :
  - Une déclaration XML (optionnelle)
  - une Enveloppe SOAP (l'élément racine) <SOAP-ENV:Envelope> qui est composée de:
    - Un en-tête SOAP (optionnel) <SOAP-ENV:Header>
    - Un corps SOAP <SOAP-ENV:Body>

Exemple de requêtes SOAP

```
<!-- Request -->
<SOAP-ENV:Envelope
            SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
            xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/"
            xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
            xmlns:xsd="http://www.w3.org/1999/XMLSchema">
   <SOAP-ENV:Body>
      <ns1:doubleAnInteger xmlns:ns1="urn:MySoapServices">
         <param1 xsi:type="xsd:int">123</param1>
      </ns1:doubleAnInteger>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>


<!-- Response -->
<SOAP-ENV:Envelope
            xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
            xmlns:xsi="http://www.w3.org/1999/XMLSchema-instance"
            xmlns:xsd="http://www.w3.org/1999/XMLSchema">
   <SOAP-ENV:Body>
      <ns1:doubleAnIntegerResponse
            xmlns:ns1="urn:MySoapServices"
            SOAP-ENV:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
         <return xsi:type="xsd:int">246</return>
      </ns1:doubleAnIntegerResponse>
   </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

Exemple d'invocation d'un Web Service



Source : http://www.dotnetguru.org/articles/webservices/WebServices.htm

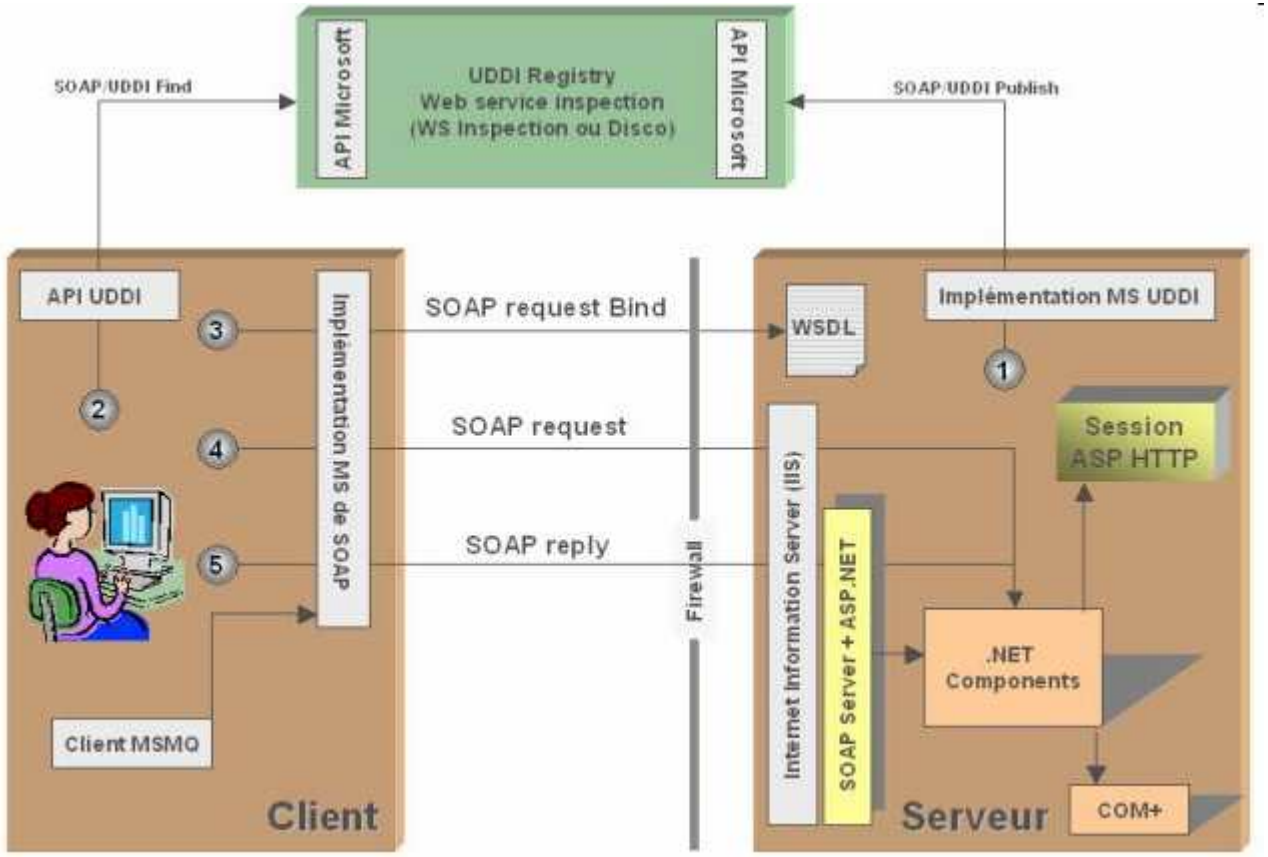
Exemple d'invocation d'un Web Service

- Le client demande un service et fait une recherche sémantique dans un annuaire UDDI qui donne la liste des prestataires habilités à répondre à la requête
- une fois la réponse reçue (en XML), recherche de l'interface du composant référencé dans l'annuaire
- l'interface WSDL décrit l'ensemble des services implémentés par l'objet distribué et il est possible de vérifier si l'interface correspond a la demande
- invocation du service : l'invocation est prise en charge par un Proxy SOAP généré coté client à l'aide de l'interface WSDL

Axis from Apache:

proxy (client, web service -> Java object, client --> Java object)

L'architecture Web Services .NET



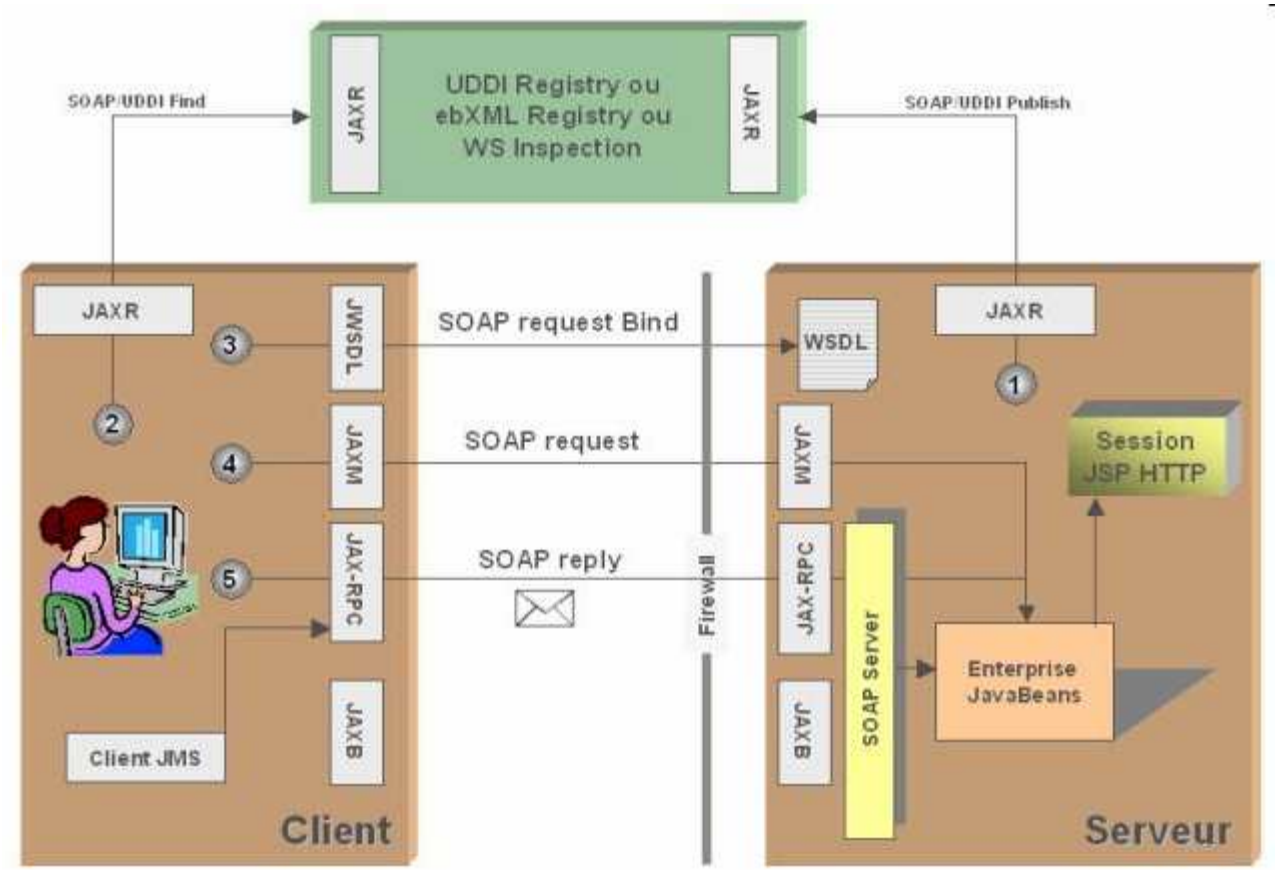
Environnement totalement intégré

Avantages :
- homogénéité de l'ensemble de l'architecture permettant d'utiliser pleinement les ressources du système Windows et du Framework .NET
- impact positif sur les performances et sur l'intégration de l'outil Visual Studio avec la plate-forme

Inconvénients
implique de posséder l'ensemble des produits de la gamme Microsoft .NET dont IIS

Source : http://www.dotnetguru.org/articles/webservices/WebServices.htm

L'architecture Web Services J2EE



J2EE fournie un ensemble d'APIs et d'interfaces dans le but de proposer plusieurs implémentations différentes. A l'heure actuelle, excepté le WebService Pack de Sun faisant office de RI (Reference Implementation), il n'existe aucun produit intégrant toutes ces APIs.

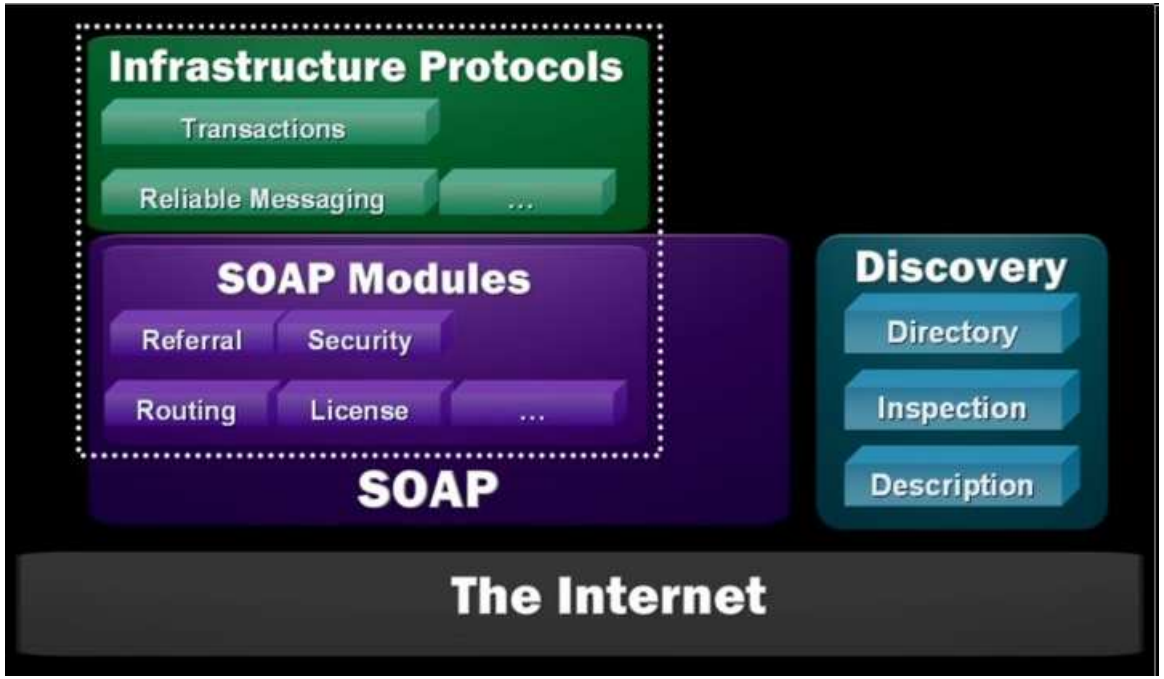
Source : http://www.dotnetguru.org/articles/webservices/WebServices.htm

Conclusion sur les web services

- Avantages
  - simple, présent sur toutes les plate-formes dans tous les langages
  - déployable partout (Internet, Intranet, Extranet)
  - couche légère sur des systèmes existant
- Problèmes
  - **Manque d'interopérabilite entre SOAP implémentations !!!!**
  - Les applications sont indépendantes et contrôlées par des organisations différentes
  - La fiabilité n'est pas garantie
  - Les interactions peuvent être synchrones ou asynchrones (one way)
  - L'implémentation de la sécurité est complexe
  - Les transactions ne sont pas supportés
  - Le protocole est trop verbeux

Requirements pour les futures Web Services

- GXA: Global XML Web Services Architecture
  - Sécurité : (WS-Security, WS-License)
  - Routage : (WS-Routing, WS-Addressing, WS-Referral)
  - Fiabilité : (Reliable HTTP : HTTPR)
  - Transactions : BPSS (Business Process Specification Schema : ebXML),



XLANG (Microsoft),
WSFL (Web Services Flow Language : IBM),
BPEL: replace WSFL (IBM) and XLANG (Microsoft), at OASIS Std. body

WSEL (Web Services Endpoint Language : IBM)

WSRF: Web Service Resource Framework for the GRID

Source : http://gotdotnet.com/team/xmlwebservices/gxa_overview.aspx

Exemple: Objets, Composants, Web Services

myDog Code Example

```
public class Dog                    // Define Dog class
 {
   public string SpeakToMe()        // Define SpeakToMe method
   {
     return "woof";                 // Dogs SpeakToMe by
   }                                // returning "woof"
 }
 public class TestDriver            // Test program
 {
   public static void Main ()
   {
     Dog myDog;                     // Declare myDog to be a Dog
     string whatHeSaid;             // Declare a local string variable
     myDog = new Dog();             // Instantiate myDog
     whatHeSaid = myDog.SpeakToMe();        // Invoke SpeakToMe on myDog
     Console.WriteLine(whatHeSaid);                 // Print out result
   }
 }
```

FIG 1

L'objet Dog peut être aisément transformé en composant ou en web service.

PB: Quand doit on utiliser un objet, ou bien un composant ou encore un web service ?

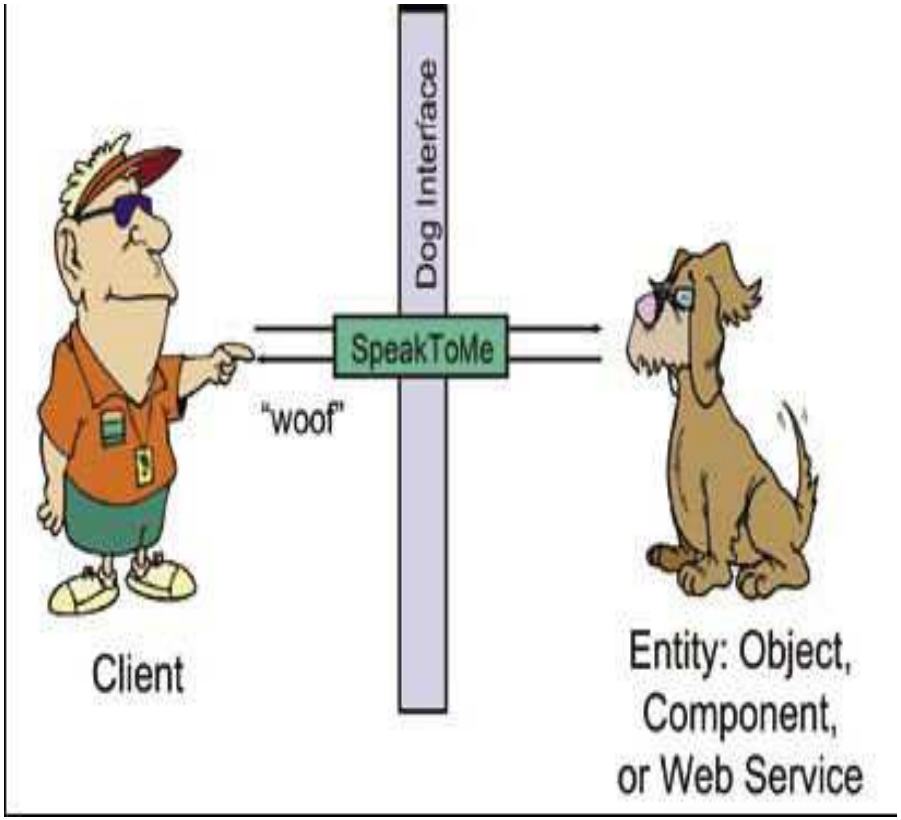
http://www-sop.inria.fr/oasis/Romain.Quilici/fuzzy.pdf

Aspects communs/Différences

- Code qui effectue une action
- Fournissent des interfaces pour décrire les services
- Vivent dans un processus
- Sont connectés a un client
- Le client invoque des méthodes pour effectuer des actions

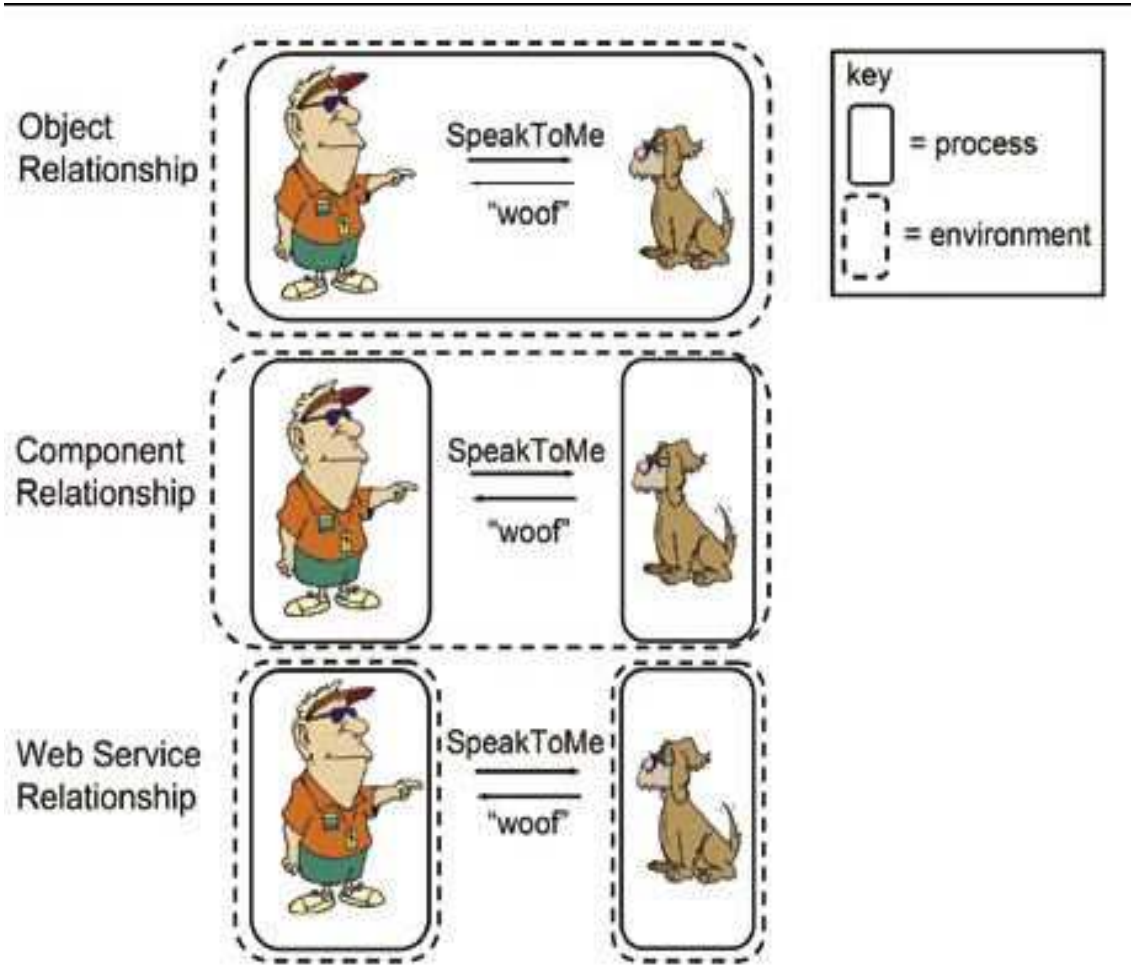
Les différences entre ces 3 notions viennent principalement de:

- La localisation des processus dans lesquels l'entité et le client s'exécutent
- Environnement d'exécution de l'entité et du client (ex serveur J2EE ou la plateforme .Net)



http://www-sop.inria.fr/oasis/Romain.Quilici/fuzzy.pdf

Relation entre entités

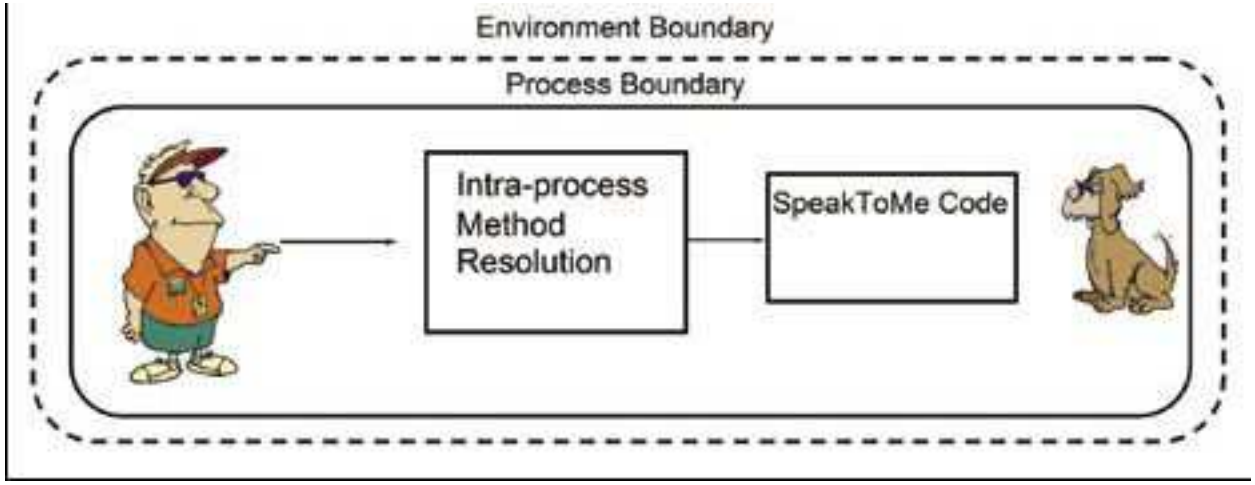


- Web Service solution globale ?

Problème de performance, en effet les web services sont plus lents que les composants qui sont plus lents que les objets

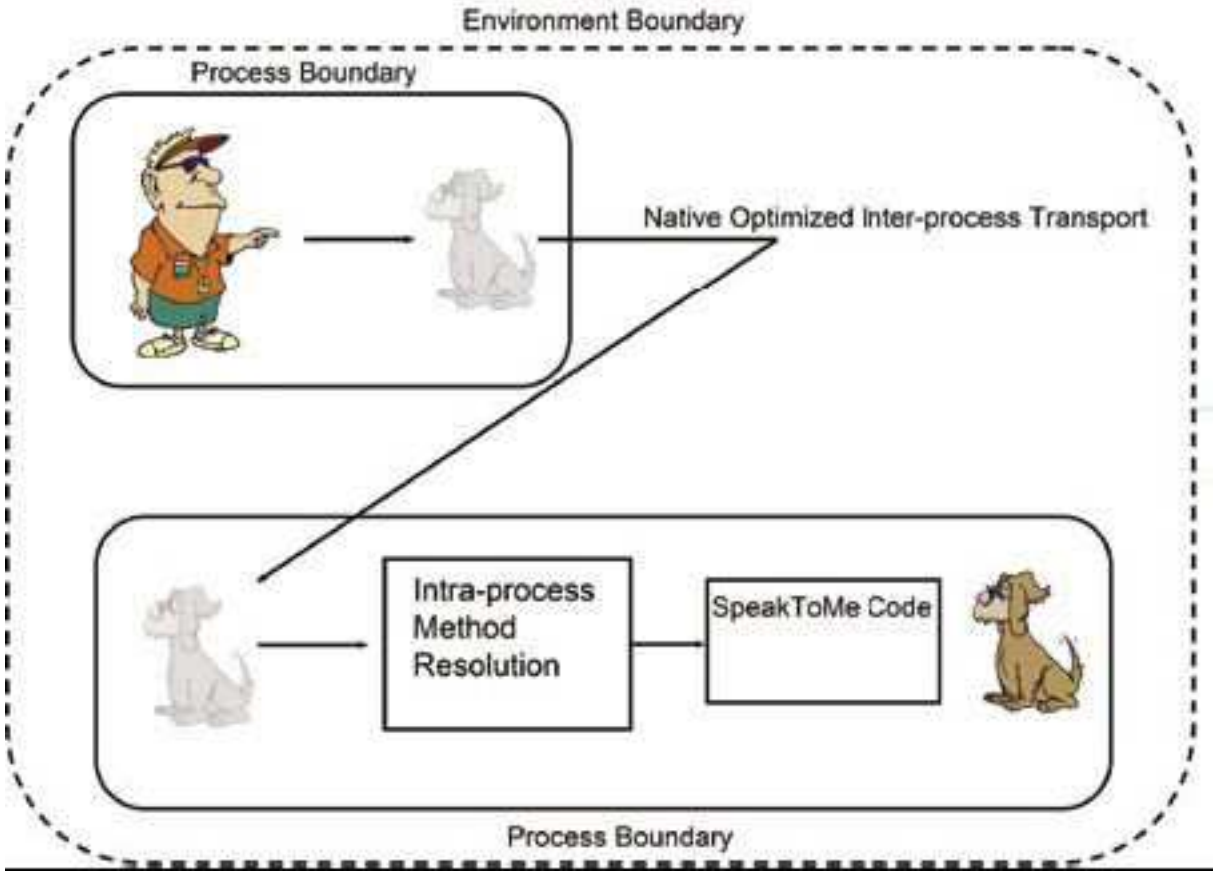
http://www-sop.inria.fr/oasis/Romain.Quilici/fuzzy.pdf

Résolution de méthode: Objets



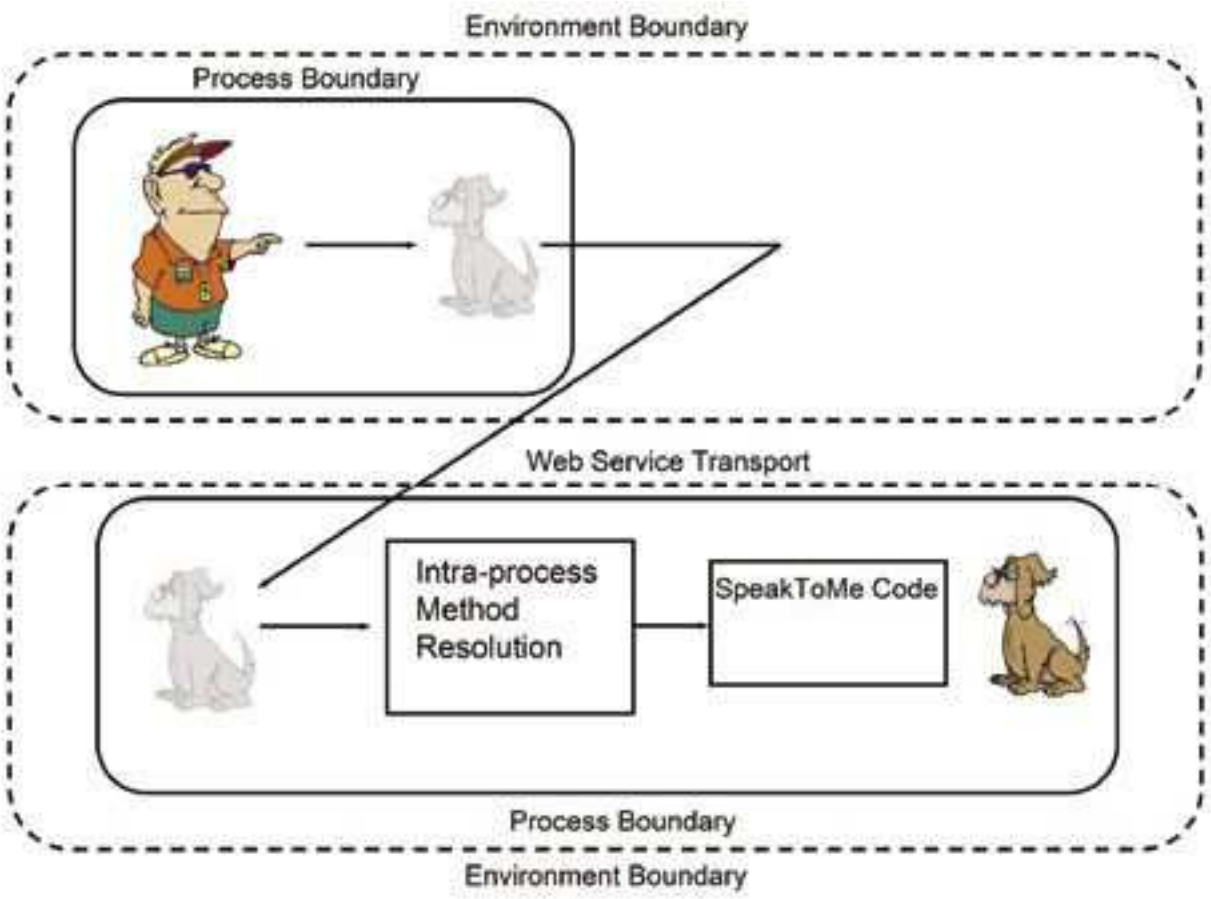
http://www-sop.inria.fr/oasis/Romain.Quilici/fuzzy.pdf

Résolution de méthode: Composants



http://www-sop.inria.fr/oasis/Romain.Quilici/fuzzy.pdf

Résolution de méthode: Web Services



http://www-sop.inria.fr/oasis/Romain.Quilici/fuzzy.pdf

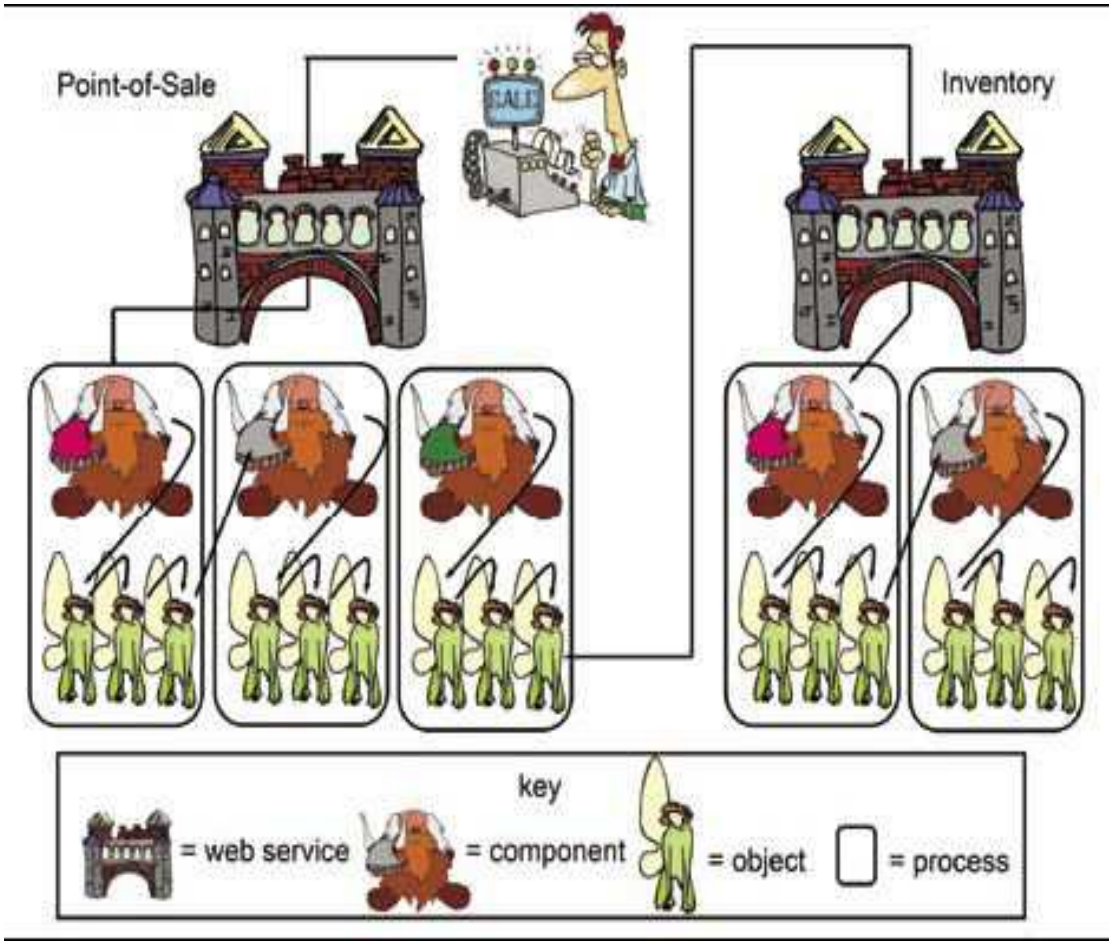
Comparaisons

TABLE 1 Comparison of Objects, Components, and Web Service Attributes

	Objects	Components	Web services
Locality	In the same process as its client	In a different process than its client	In a different organization than its client
Environment	In the same environment as its client	In the same environment as its client	In a different environment than its client
Speed	Very fast communications with its client	Slow communications with its client	Very slow communications with its client
Builder relationship	Probably built by same person that built its client	Probably built by the same group that built its client	Probably built by a different company than built its client
Quantity	Tons—every place you look!	Quite a few—at least one for each process	Hardly any—perhaps one per major software system

http://www-sop.inria.fr/oasis/Romain.Quilici/fuzzy.pdf

Exemple: Un point de vente et système d'inventaire



Deux systèmes développés séparément:

- Le point de vente
- Le système inventaire

  → Web Services

Point de vente

- Accessible depuis plusieurs clients différents

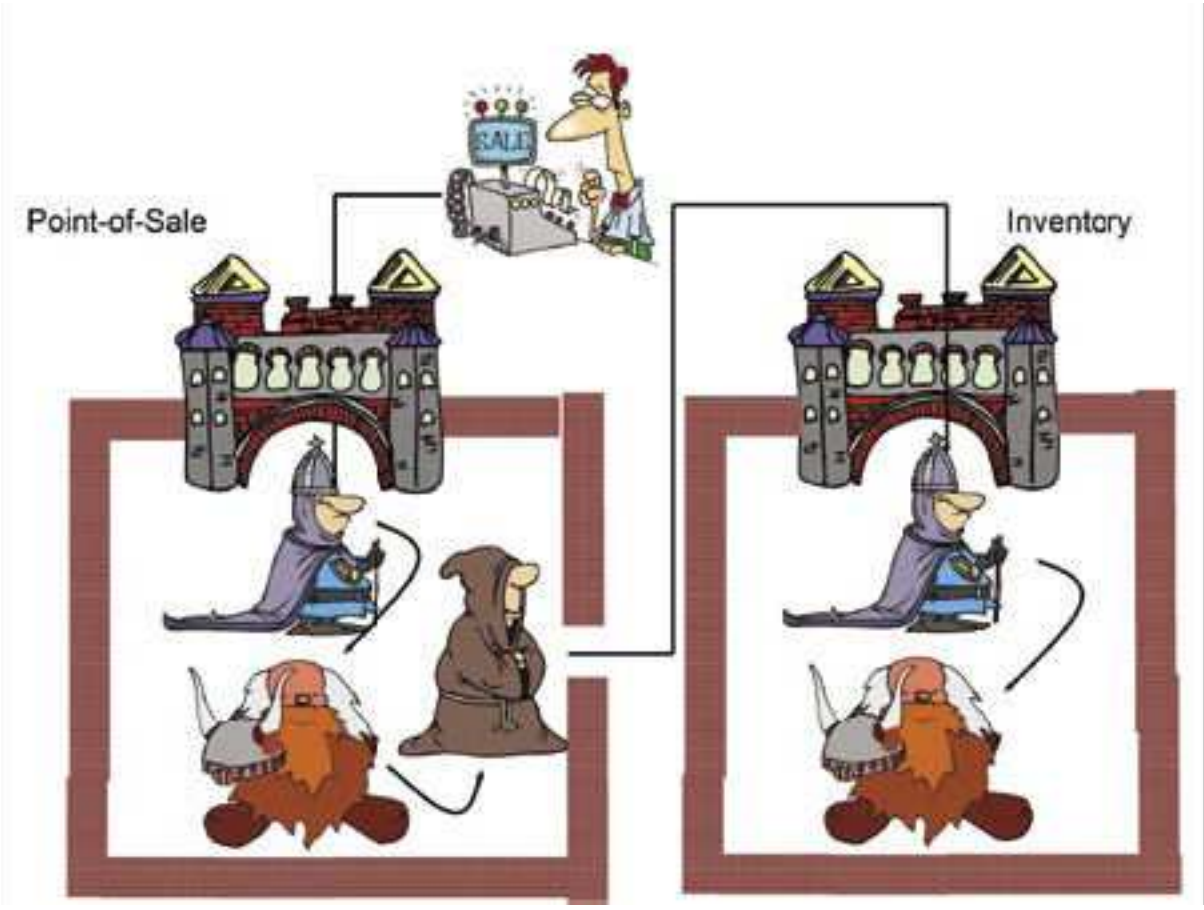
  → Web Services
- Différentes machines
- Différentes parties

  → Composants
- Exécution de chaque partie

  → Objets

http://www-sop.inria.fr/oasis/Romain.Quilici/fuzzy.pdf

Exemple: Un point de vente et système d'inventaire



- Garde: point d'entrée
- Travailleur: application
- Envoyeur: point de sortie

http://www-sop.inria.fr/oasis/Romain.Quilici/fuzzy.pdf