M2 : Intégration et travail coopératif

Bertrand DAVID

Ecole Doctorale ED IIS
DEA ISCE

Architectures :
Approche grammaticale
Systèmes experts
Architecture black board
Systèmes intégrés industriels
Applications interactives
Client – Serveur
CORBA
J2EE
JavaBeans
.NET
Serveurs d'applications : CGI, JSP, EJB
OLE - COM - DCOM ActiveX

Spécification à l'aide de grammaire

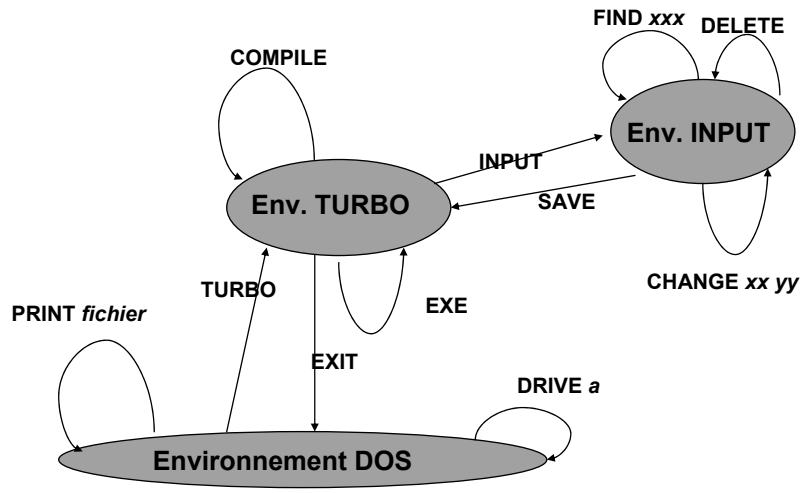
Grammaire : G = (Vn, Vt, S, {règles})

Vn = (envDOS, envTURBO, envINPUT)
Vt = (TURBO, DRIVE, PRINT, COMPILE, EXE, EXIT, INPUT, SAVE, CHANGE, DELETE, FIND, a, f, x, y)
S Axiome (Starter)

Règles

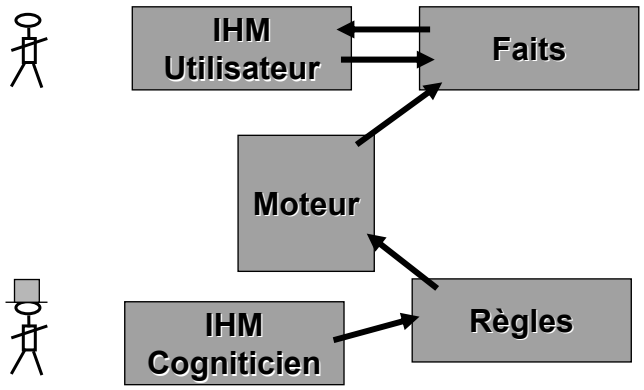
S :: envDOS
envDOS :: TURBO envTURBO | DRIVE a envDOS | PRINT f envDOS
envTURBO :: Input envINPUT | EXIT envDOS | COMPILE envTURBO | EXE envTURBO
envINPUT :: FIND x envINPUT | CHANGE x y envINPUT | DELETE envINPUT | SAVE envTURBO

Spécification à l'aide de graphe



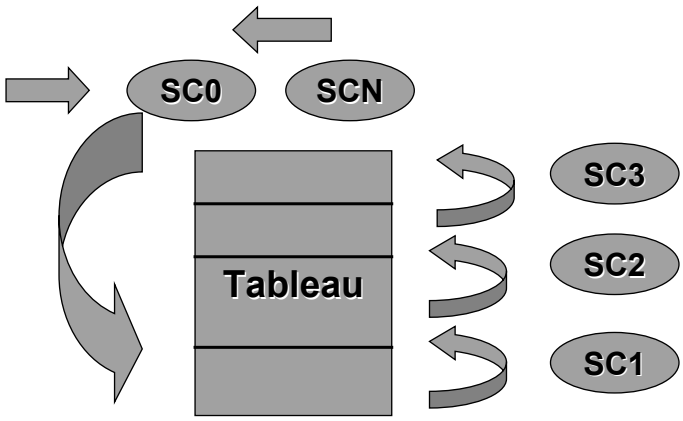
ED IIS - DEA ISCE Intégration & Travail Coopératif 3

Systèmes experts

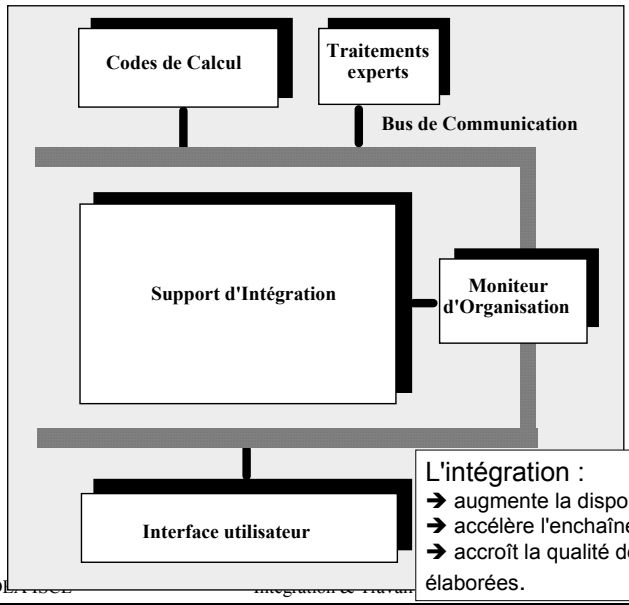


ED IIS - DEA ISCE Intégration & Travail Coopératif 4

Architecture black board



Systèmes intégrés industriels



Intégration par :

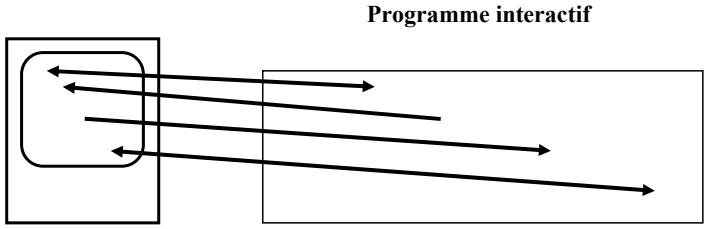
- **des traitements**
- **des données**
- **des processus**

L'intégration :
➔ augmente la disponibilité des données,
➔ accélère l'enchaînement des traitements,
➔ accroît la qualité des informations élaborées.

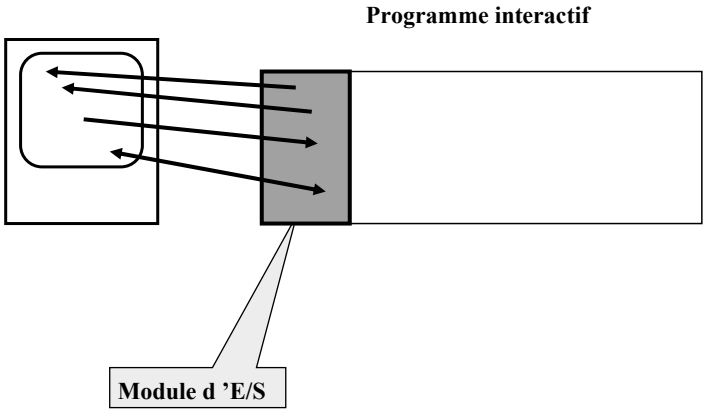
Applications interactives

- Des modèles d'architectures proposés pour des applications interactives (voir ci-après leur évolution) ont pour but de proposer un cadre (framework) de développement.
- Elles peuvent être à couches (SEEHEIM, Modèle Langage, ARCH,…) ou à objets ou agents (PAC, AMF, …)

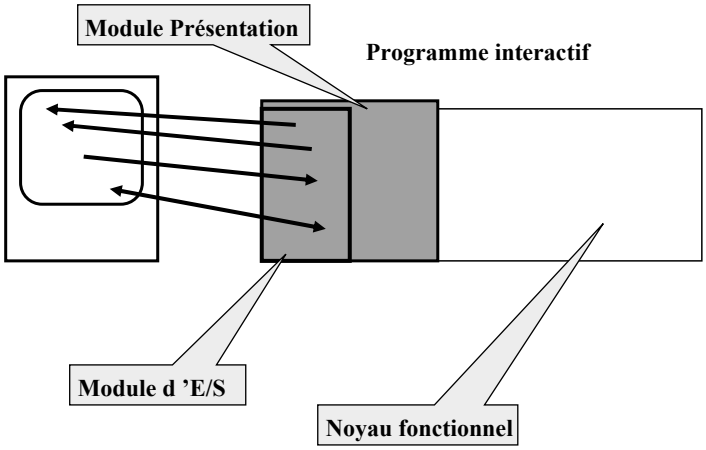
Architecture magmatique



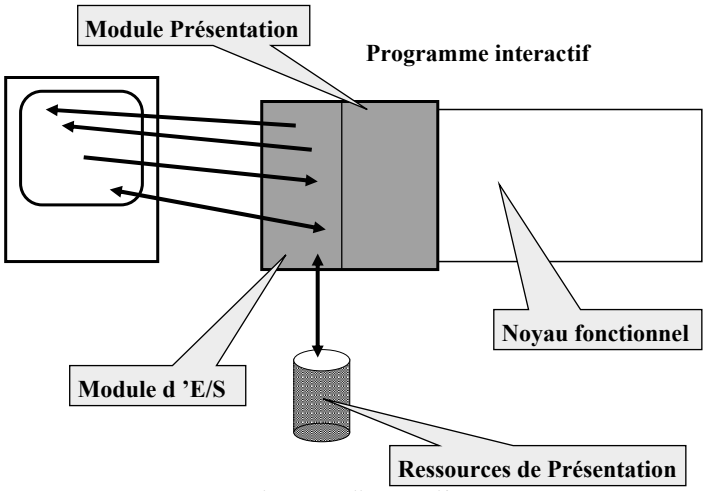
Module Entrées/Sorties



Module Présentation

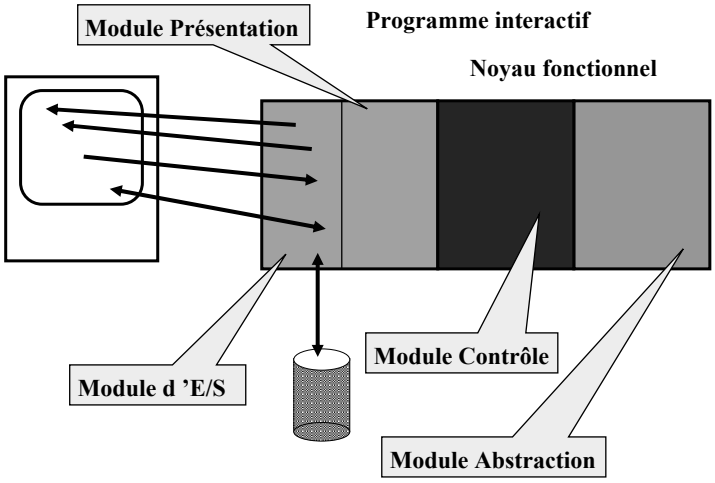


Ressources de présentation



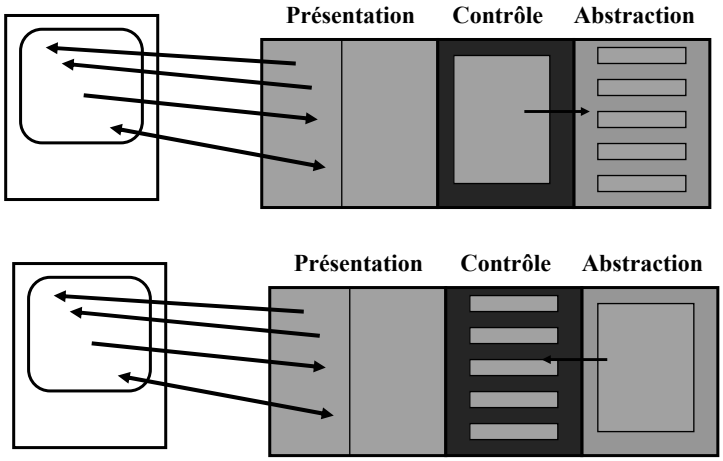
ED IIS - DEA ISCE Intégration & Travail Coopératif 11

Module Contrôle



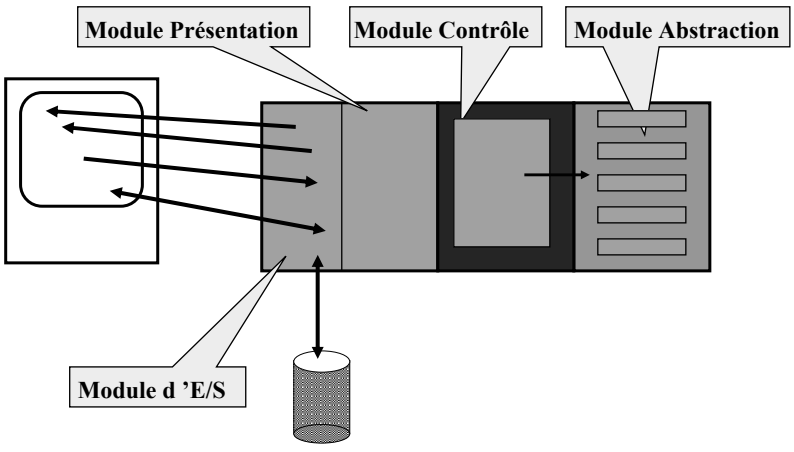
ED IIS - DEA ISCE Intégration & Travail Coopératif 12

Contrôle Interne /Contrôle Externe



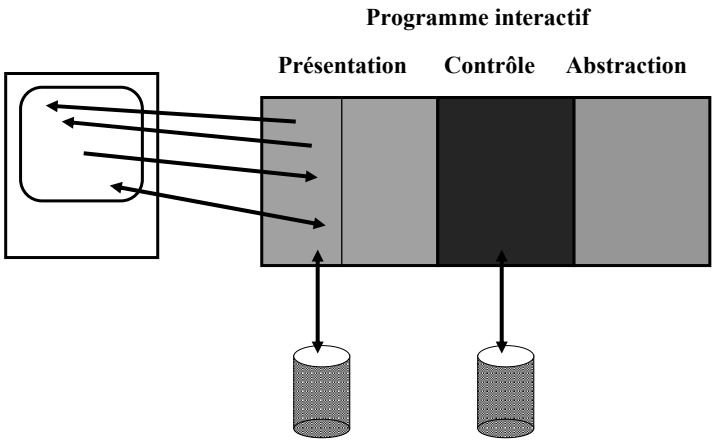
ED IIS - DEA ISCE Intégration & Travail Coopératif 13

Contrôle Interne /Contrôle Externe



ED IIS - DEA ISCE Intégration & Travail Coopératif 14

Ressources de contrôle



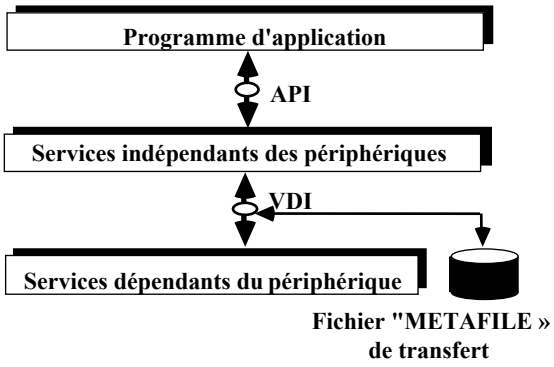
ED IIS - DEA ISCE Intégration & Travail Coopératif 15

Portabilité - adaptabilité

ED IIS - DEA ISCE Intégration & Travail Coopératif 16

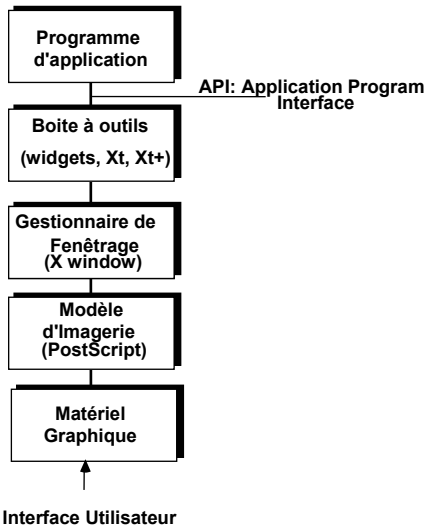
Portabilité

- Approche portabilité : **structuration à deux niveaux fait apparaître des primitives dépendantes et indépendantes des périphériques**



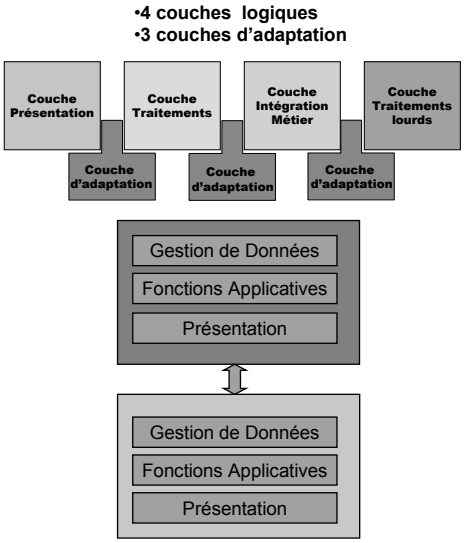
- **La portabilité est assurée par une architecture en couches avec des notions d'interface standard: VDI (Virtual Device Interface) et API (Application Program Interface)**

Pipeline d'affichage



Client - Serveur

- Architecture



ED IIS - DEA ISCE Intégration & Travail Coopératif 19

Client - Serveur

Définitions :

- **Un service est un ensemble de fonctions mises à la disposition des programmes d'application et généralement partageable entre ces programmes.**
- **Très souvent, un service logiciel gère des ressources.**
- **La notion de service n'est pas nouvelle, ce qui est nouveau c'est la possibilité de dissocier la localisation de l'application appelante et la localisation d'une partie ou de la totalité des fonctions de service appelé.**
- **On dira que l'on est en mode client / serveur lorsque le service appelé incorpore un dispositif logiciel qui lui permet d'être exécuté localement ou à distance et de façon transparente pour l'application qui a recours à ce service.**
- **On observe que le service comporte en fait deux parties : une partie proche de l'application appelante nommée demandeur et une partie éloignée chargée de gérer une ressource appelée gestionnaire de ressource.**
- **Par extension, on appelle client la machine qui contient l'application appelante et la partie demandeur, la machine qui contient la portion déportée du service est nommée serveur.**

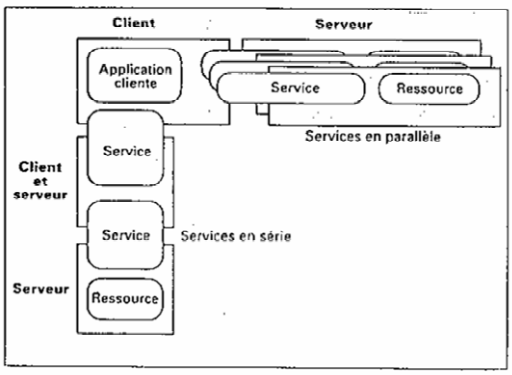
ED IIS - DEA ISCE Intégration & Travail Coopératif 20

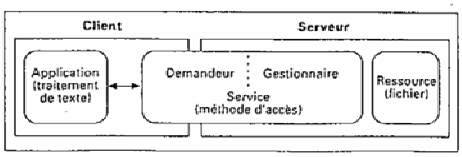
Client – Serveur : Configurations logicielles

Trois schémas sont envisageables :

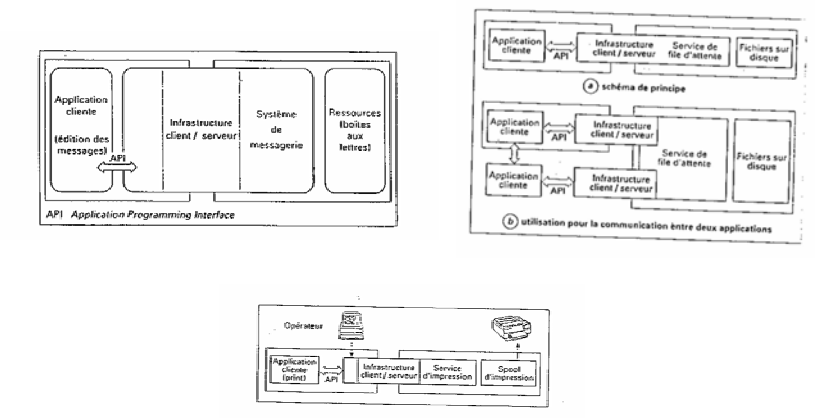
- **le cas simple dans lequel une application fait appel à un seul serveur (base de données par exemple),**
- **le cas où l'application fait appel à plusieurs services qui peuvent être localisés ou non sur le même serveur (accès aux fichiers et à la base de données). On parle de configuration logique en parallèle,**
- **le cas se services imbriqués où un service est lui-même client d'un autre service (accès à la base de données fait appel à des services d'accès aux fichiers). On parle de configuration logique en série.**

Client - Serveur





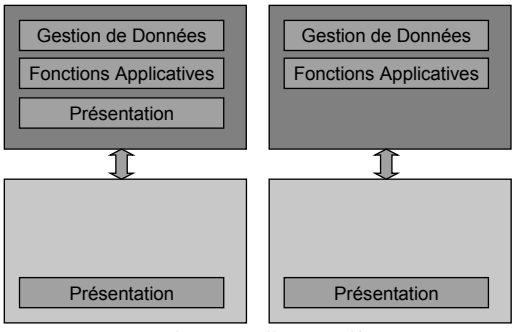
Client - Serveur



Distribution de la présentation (Types 1 et 2)

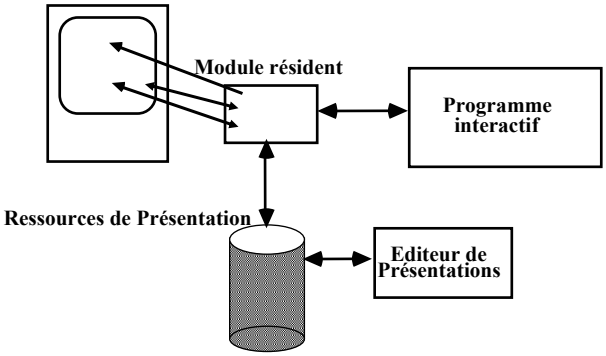
La présentation est une des formes du client/serveur consistant à ne localiser sur le poste Client que les actions de présentation (décentralisation des fonctions d'interface utilisateur).

- Type 1 : l'affichage seulement est déporté. Ceci est utilisé notamment pour la réhabilitation d'applications existantes (rewamping) conçues initialement en mode caractère.
- Type 2 : l'interface est entièrement (affichage et gestion des interactions) sur le poste client. Ceci correspond aux nouvelles applications développées sur le réseau local.



Amélioration externe

- **Amélioration des programmes existants par une présentation définie de façon externe :**

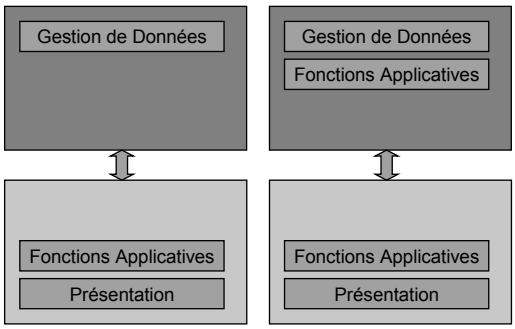


ED IIS - DEA ISCE Intégration & Travail Coopératif 25

Distribution des fonctions applicatives
(Types 3 et 4)

La distribution des fonctions applicatives est la forme du client/serveur consistant à partager les fonctions de logique fonctionnelle.

- Type 3 : correspond à la séparation de fonctions d'accès aux données et de la logique fonctionnelle.
- Type 4 : les traitements liés aux interactions utilisateur sont sur le poste client alors que les traitements liés aux manipulations de données sont sur le serveur de manière à réduire l'utilisation du réseau (en volume de données et en fréquence).

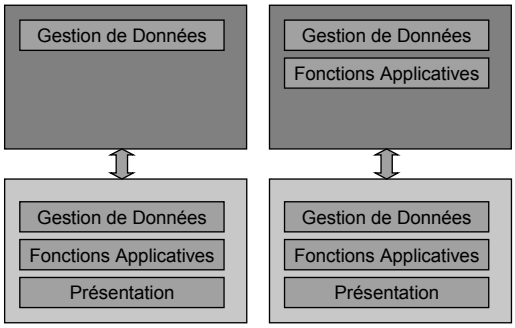


ED IIS - DEA ISCE Intégration & Travail Coopératif 26

Distribution des données (Type 5 et 6)

La distribution des donnée peut couvrir des situations différentes:

- Type 5 :le poste client gère partiellement des données.
- Type 6 :les données peuvent être distribuées sur plusieurs serveurs qui capables ou pas mettre en oeuvre des fonctions applications.



ED IIS - DEA ISCE Intégration & Travail Coopératif 27

Infrastructure client / serveur

- Les fonctions qui permettent la délocalisation transparente du service par rapport à l'application appelante constituent l'infrastructure client/serveur. Cette infrastructure est désignée par le terme d'environnement de traitements répartis appelé également middleware.
- L'organisation vue précédemment décomposant le service en deux parties: une locale (près de l'application), l'autre distante (située sur le serveur) gérant les ressources. Entre les deux, le middleware vise à assurer la transparence de localisation.

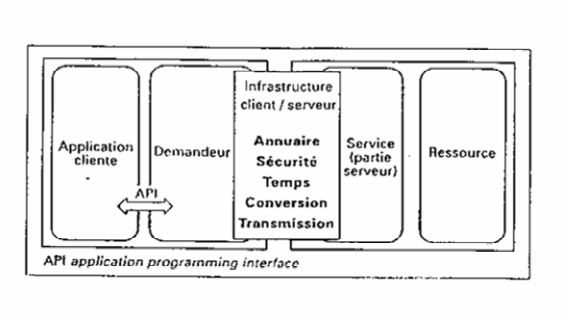
Les principales fonctions du middleware sont les fonctions suivantes :

- adressage : gérer l'accès aux services vie une table d'adressage (service - localisation) pour optimiser le fonctionnement,
- conversion : transformer le codage des informations,
- communication : acheminer des requêtes et des réponses entre les différentes machines,
- sécurité : garantir la confidentialité et l'intégrité des données échangées,
- gestion du temps : synchroniser les horloges des machines,

Ces fonctions peuvent elles-mêmes être réparties sur plusieurs systèmes.

ED IIS - DEA ISCE Intégration & Travail Coopératif 28

Client - Serveur



Architecture technique Client / Serveur

Application

API - Application Programming Interfaces
Par exemple : SQL

FAP - Format and Protocols (protocole de communication et format de données)
Par exemple :
DCE ou APPC (Application Program to Program Communication)
RDA (Remote Data Access) norme ISO pour l'accès distant aux bases de données
RPC (Remote Procedure CAll)

Transport
Par exemple : TCP-IP

Typologie d'échanges client/serveur :

Contexte synchrone

Mode d'échange où le client suspend son exécution en attendant la réponse du serveur.

Le client est passif à partir du moment où il a transmis son appel jusqu'à la réception de la réponse du serveur. Le temps d'inactivité est la somme de trois périodes :

- ➔ le temps de transmission de la requête,
- ➔ le temps d'exécution de la procédure,
- ➔ le temps de transmission de la réponse

Typologie d'échanges client/serveur :

Contexte asynchrone

Une fois l'ordre envoyé au serveur, le client poursuit l'exécution de son application. Cette méthode rend le temps de réponse apparent pratiquement négligeable. La mise en oeuvre est plus délicate, car il est nécessaire que l'application reste à l'écoute de l'événement de retour de la requête.

Applications dans un environnement interactif

Deux contextes différents d'utilisation interactive :

conversationnel et transactionnel

Mode conversationnel

Les applications doivent être écrites indépendamment de la gestion de la location. Les fonctions de type "appel de procédure distante" plus connues sous leur signe anglais (Remote Procedure Call) prennent en charge le déport des appels de sous-programmes et constituent la base de développements interactifs.

La fonction "appel de procédure distante" s'interpose entre le programme appelant et le programme appelé et joue deux rôles principaux :

- transformer les paramètres des appels de sous-programmes et les réponses de ces sous-programmes en messages transmissibles grâce aux services de transmission,
- faire appel aux fonctions de l'infrastructure client/serveur pour :
  - déterminer l'adresse du serveur grâce à la table d'adressage,
  - convertir les paramètres,
  - transmettre les messages contenant les paramètres.

ED IIS - DEA ISCE Intégration & Travail Coopératif 33

Le service RPC est divisé en deux parties

- **Une partie localisée dans la** machine client **se substitue au programme appelé; c'est elle qui récupère les paramètres émis par l'appelant, les convertit (si besoin), les insère dans un message et fait appel aux fonctions de routage et de communication pour les envoyer à la machine serveur ;**
- **Une partie localisée dans la** machine serveur **reçoit le message, appelle le programme d'application et lui fournit les paramètres de sorte que ce programme "croit" que l'appelant est un programme local**

ED IIS - DEA ISCE Intégration & Travail Coopératif 34

Mode transactionnel

Dans le mode transactionnel, d'autres impératifs sont à prendre en compte :

- **un grand nombre d'utilisateurs sont susceptibles d'accéder aux données et aux traitements,**
- **les traitements sont regroupés en transactions. Une transaction est formée d'une suite d'interactions entre le système et l'utilisateur qui doit répondre à une série de caractéristiques que l'on dénomme propriétés ACID (atomicité, cohérence, isolation, durabilité) définies par ISO/IEC 10026/1 :**
  - l'atomicité signifie que le découpage de la transaction en étapes de traitements plus élémentaires ne doit pas être possible ;
  - la cohérence signifie que les ressources accédées par une transaction doivent être laissées dans un état cohérent par cette transaction ;
  - l'isolation signifie que les déroulements d'autres traitements extérieurs à la transaction ne doivent pas interférer sur le déroulement de la transaction ;
  - la durabilité signifie que les effets de la transaction sur le ressources doivent être persistants après la fin de la transaction et ne peuvent être effacés qu'explicitement par un autre traitement.

ED IIS - DEA ISCE Intégration & Travail Coopératif 35

Mode transactionnel

Le traitement transactionnel en mode réparti exige la mise en place de mécanismes qui vont permettre le respect de ces exigences :

- le verrouillage des ressources utilisées, pour empêcher leur usage momentané par d'autres transactions ;
- la journalisation des modifications de fichiers pour restaurer leur état en cas de dysfonctionnement du système ;
- le déclenchement des actions sur les ressources e, fin de transaction, mécanisme connu sous le nom de validation en deux phases (two phase commit) ;
- le déclenchement du traitement associé à un message provenant d'une autre machine ;
- la relance d'une transaction après interruption non planifiée du système.

Le rôle du moniteur de transactions est de gérer le déroulement des transactions et en particulier les synchronisations entre les deux machines.

ED IIS - DEA ISCE Intégration & Travail Coopératif 36

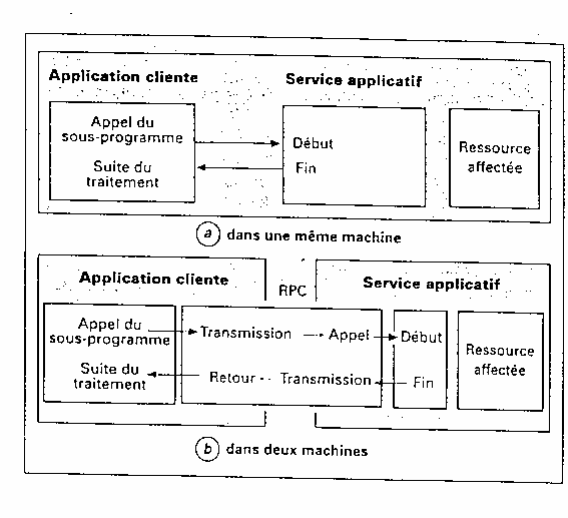
Accès aux fichiers

- Plusieurs approches sont disponibles selon l'endroit où la méthode d'accès est scindée entre la partie client et la partie serveur.
- Une mise en oeuvre utilise le mécanisme de redirection d'entrée-sortie :une fonction particulière est chargée de rediriger la demande d'entrée-sortie sur la machine qui héberge le fichier et son serveur. La localisation du fichier est transparente à l'application.
- Une autre technique consiste à partager le fichier par déport des requêtes d'accès à un enregistrement logique du fichier distant plutôt que par déport des accès à un bloc physique. Ce procédé permet de limiter la quantité d'information qui circule sur le réseau et sera donc recommandé pour l'accès déporté à des fichiers à travers des réseaux longue distance.

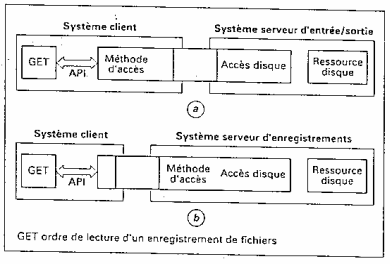
Bases de données

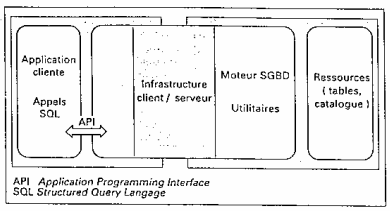
- Il est important de distinguer les bases de données réparties et l'accès répartis aux bases de données.
- Dans le cas d'une base de données répartie la base est découpée en plusieurs sous-ensembles eux-mêmes résidents sur les machines différentes.
- Dans le traitement client/serveur d'accès répartis aux bases de données, le gestionnaire de base de données est capable de rediriger des requêtes d'accès à une base de données vers une autre machine et de fournir la réponse à cette requête sur le système du demandeur.
- Les requêtes peuvent être simples (requêtes à distance). On peut également avoir des transactions à requêtes multiples à distance mettant en oeuvre éventuellement plusieurs bases de données localisées sur des machines différentes. On peut également concevoir un accès distant à des bases de données reparties.

Client - Serveur

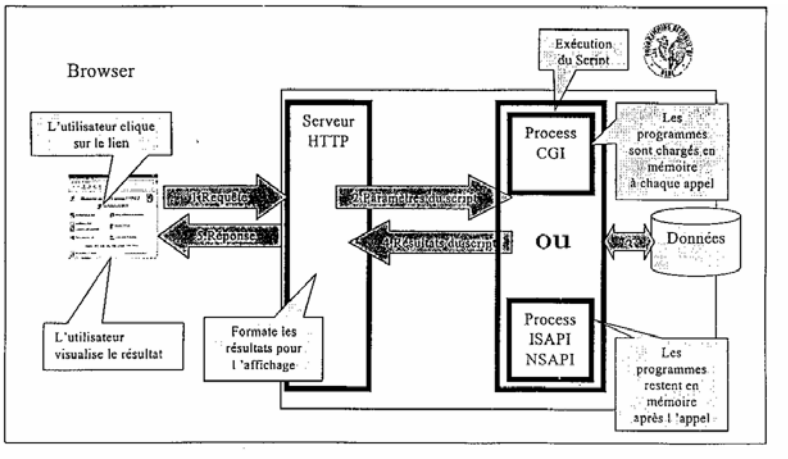


Serveur de fichiers et de BD

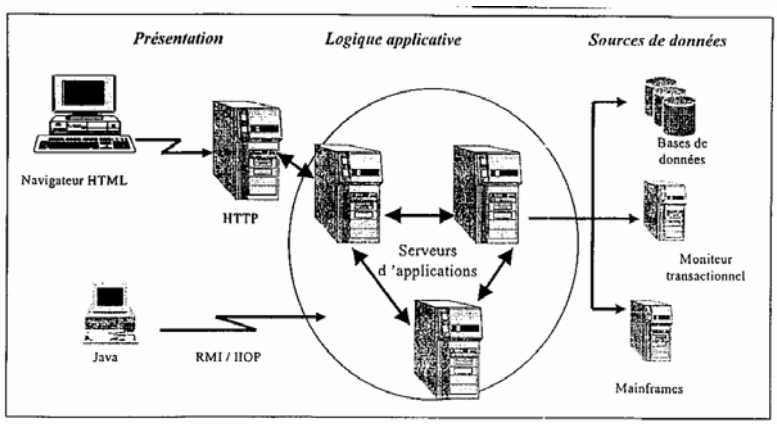




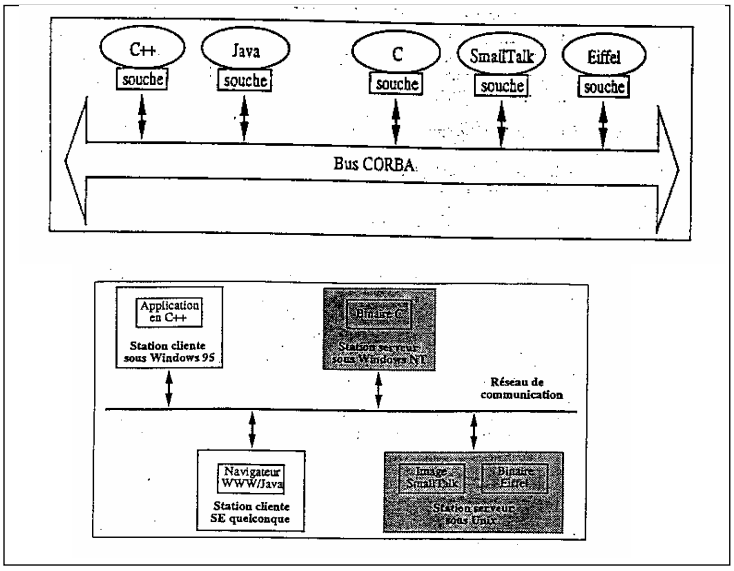
Application Web



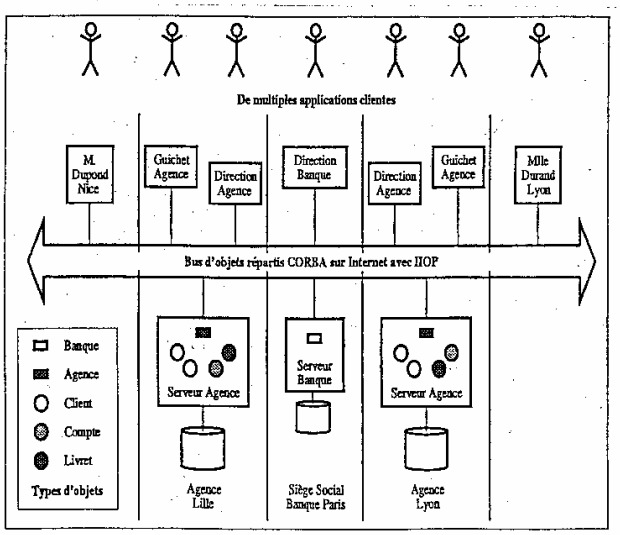
Client-Serveur multi-tiers



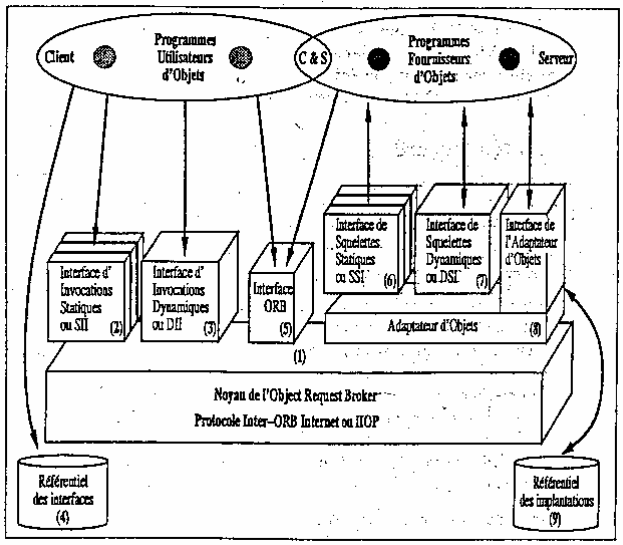
CORBA (1/4)



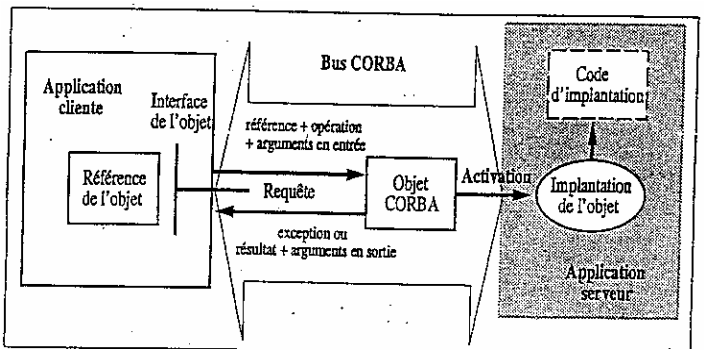
CORBA (2/4)



CORBA (3/4)



CORBA (4/4)



Plateforme Java Entreprise Edition

1. Qu'est-ce que Java
2. Pourquoi la plateforme J2EE ?
3. J2ee (Java 2 Platform Enterprise Edition) & Les Applications distribuées multi-tiers
4. JDBC (Java DataBase Connectivity)
5. Rmi (Remote Method Invocation)
6. Les Servlets
7. Les JSP(Java Server Pages)
8. Les EJB (Enterprise Java Beans)
9. Conclusions

Qu'est-ce que Java ?

- Un Langage indépendant de la plate forme
  Le code source est compilé en un code intermédiaire ou byte code qui sera soumis à une couche logicielle , appelée machine virtuelle(VM) .
  Cette machine virtuelle interprète le byte code pour l' exécuter sur la plate forme cible .
- Un langage de programmation orienté objets
  La définition des objets a lieu dans des classes
  Regroupées dans des bibliothèques (les packages)
- Un ensemble d'API variées
  Standard: Structures de programmation classiques (listes, dictionnaires)
  Spécialisées: bibliothèques normalisées couvrant de nombreux domaines allant de l'Audio/vidéo au graphisme 3d

Qu'est-ce que Java /C++?

- **Langage réellement portable**
  A la différence de C++, java n'emploie pas d'instructions spécifiques à la machine cible.(COMPILE UNE FOIS,EXECUTE PARTOUT)
- **Langage proche de C++**
  Même syntaxe, mêmes structures de contrôle
  Permet l'utilisation de code C++ à partir de programmes écrits en langage Java
- **Langage différent du C++pour la gestion de la mémoire**
  Les allocations_libération de mémoire sont gérées par la machine virtuelle
  Au travers d'un mécanisme appelé GarbageCollector ou ramasse miettes
- **Langage simplifiant la programmation**
  Un tableau véhicule sa dimension, tandis que le système gère les éventuels débordements
  gestion transparente des pointeurs

Qu'est-ce que Java ?
Ses particularités

- Java **Supporte la programmation parallèle** Programmation de processus concourants par l'intermédiaire de threads
- Java **Supporte la programmation réseau**
  Bibliothèques de gestion des objets répartis sur le réseau.
  Bibliothèques de gestion de la sécurité des applications réseau
- Java **supporte le développement d'applications N tiers**
  Permet d'envisager le développement d'applications conséquentes

Qu'est-ce que Java ?
Un ensemble d'outils

- **Environnements de développement** :
  - Sun JDK 1.3.x (compilateur, interpréteur, *appletviewer*,...)
  - IDE : IBM Visual Age, Symantec Visual Café, Borland Jbuilder, Java WorkShop, Visual J++, ...

Des serveurs d'application

- IBM Websphere,Bea Weblogic,Allaire Jrun,…
- Tomcat

Pourquoi la plateforme J2EE ?

Il existe deux versions du kit de développement

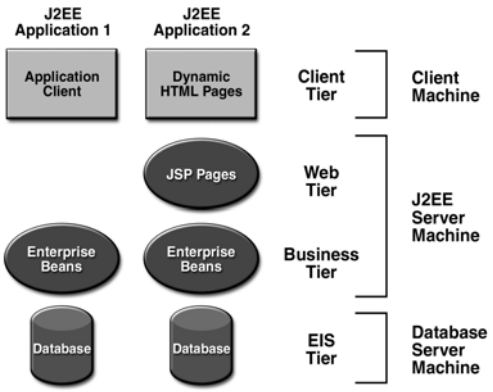
1/ La version standard du kit de développement Java

La version standard permet de développer toutes sortes d'applications, des programmes simples jusqu'aux serveurs.

2/ La version enterprise du kit de développement Java

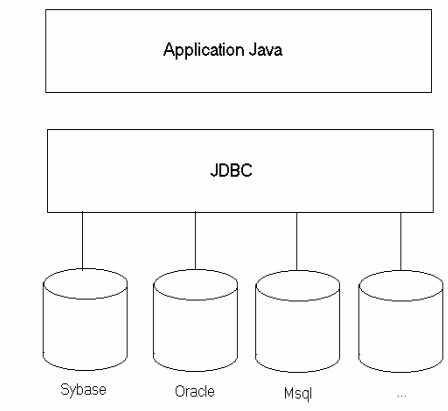
La version appelée J2EE(Java 2 Edition Professionnelle) possède des bibliothèques spécifiques permettant le développement d'application réseau complexes.

J2EE: Les applications distribuées multi-tiers



JDBC (Java DataBase Connectivity)

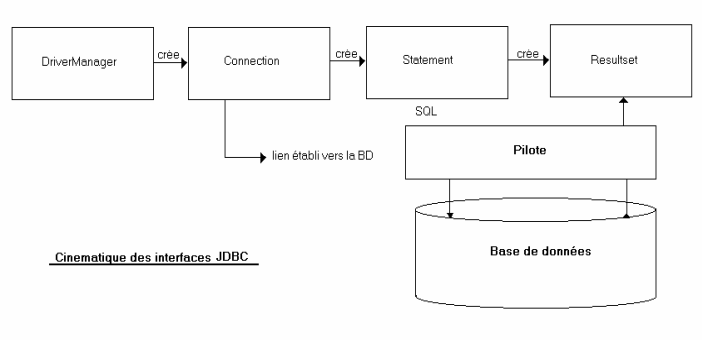
- Il s'agit d'un ensemble de classes ou d'interfaces permettant l'utilisation sur le réseau d'un ou plusieurs SGBDR à partir d'un programme Java



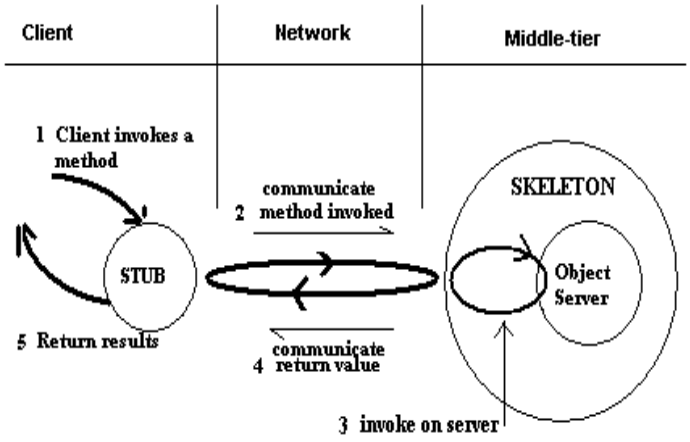
Mise en œuvre de JDBC

Les différentes étapes :

- Importer le package java.sql
- Enregistrer le driver JDBC
- Etablir la connexion à la base de données
- Créer une zone de description de requête
- Exécuter la requête
- Traiter les données retournées
- Fermer les différents espaces



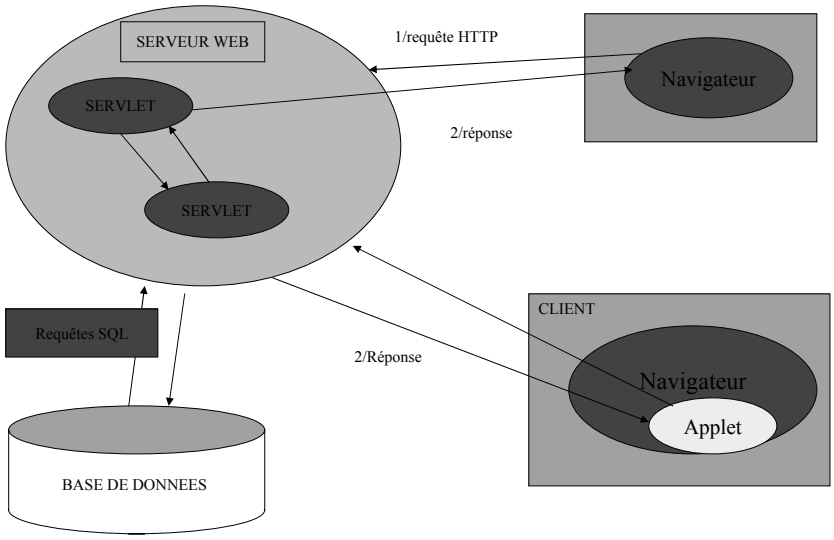
Rmi (Remote Method Invocation)



Servlet et génération de pages dynamiques Comparaison servlets/Applets

- Une servlet est un programme Java qui s'exécute sur un serveur d'application pour répondre aux requêtes des systèmes clients
- Une servlet est au serveur ce qu'une Applet est a un Navigateur
- En général même si la notion de servlet n'est pas liée Au protocole HTTP, c'est ce protocole qui est utilisé. D'où la dénomination <<http SERVLET>>
- Paquetages et classes associés
  - Javax.servlet
  - Javax.servlet.http

Les Servlets et leur accès



ED IIS - DEA ISCE Intégration & Travail Coopératif 58

Servlet et génération de pages dynamiques Fonctionnalités offertes par les servlets

- Génère une partie dynamique dans une page web Html statique existante
- Crée et envoie une page web Html dynamique qui peut être fonction des paramètres de la requête ou de la nature de la requête et du résultat de la requête à une base de données.
- Peut gérer concurremment la connexion avec plusieurs clients en partageant des données communes.
- Contrôle les sessions avec un client particulier en sauvegardant son contexte et en le restaurant à l 'aide des cookies.

Servlet et génération de pages dynamiques Avantages apportés par les servlets/CGI

- Langage indépendant des systèmes d 'exploitation(NT,UNIX) et des serveurs(Apache,Sun,Netscape..) .
- Une servlet ne s 'exécute pas dans un processus séparé ,il n 'y a pas de création de processus à chaque requête.
- Une servlet contient une mémoire persistante entre chaque requête, ceci permet de partager les informations entre clients.
- Prend en charge les connexions multiples(plusieurs clients)par le biais du multithread avec mémoire partagée et persistante.
- Offre par le mécanisme des moniteurs Java la gestion des accès concurrents.
- Permet comme pour les applets d 'appliquer des règles d 'accès restrictives pour assurer la sécurité.

Servlet et génération de pages dynamiques Autres avantages

- Rapidité d ’exécution du fait que la servlet n’est chargée qu ’une seule fois.
- Facilité de communication avec des servlets présentes sur le même serveur
- Équilibrage de charge sur les serveurs:Possibilité d ’appeler des servlets sur d ’autres serveurs pour déporter des traitements

Servlet et génération de pages dynamiques Appel d ’une Servlet

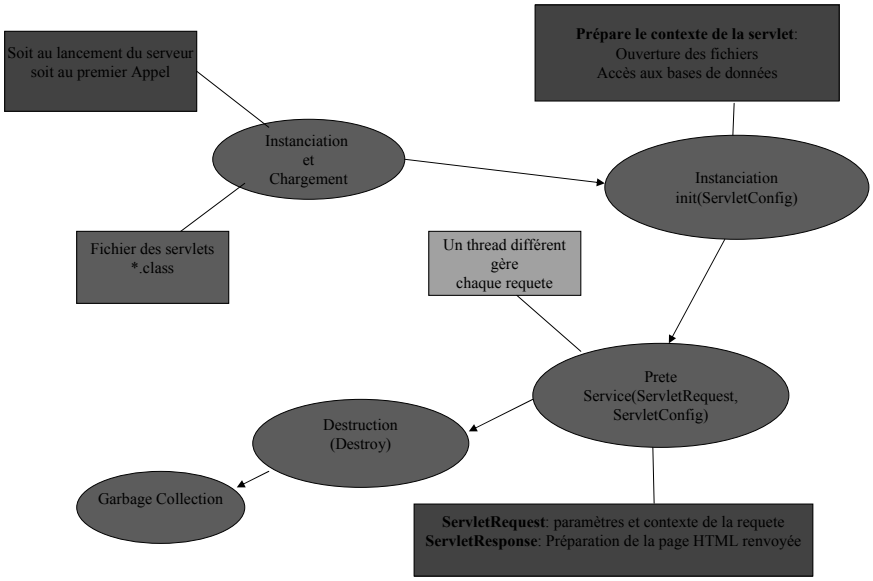
- **A partir du navigateur** :
  - En entrant l ’URL de la servlet et les paramètres de la requete
- **A partir d ’une page HTML** :
  - En utilisant les balises <FORM> et</FORM>
  - Puis en spécifiant les méthodes GET ou POST
  - Le formulaire HTML fournissant les paramètres utilisateur

- Dans une page HTML par l’emploi de la balise <SERVLET>

Servlet et génération de pages dynamiques Exemple de génération de page

```
import java.io.*;
import java.servlet.*;
import java.servlethttp.*;
public class HelloWorld extends HttpServlet{
public void doGet(HttpServletRequest requete,HttpServletResponse reponse)
  throws IOException,ServletException{
reponse.setContentType(« text html »);
PrintWriter out=reponse.getWriter();
out.println(« <html> »);   /* écriture du source de la page HTML
out.println(« <body> »);
out.println(« <head> »);
out.println(« <title> page simple</title>»);
out.println(« </head> »);
out.println(« </body> »);
out.println(« </html> »);
}}
```

Cycle de vie d'une servlet



ED IIS - DEA ISCE Intégration & Travail Coopératif 64

Servlet et génération de pages dynamiques
Configuration nécessaire

- JDK 2 Java Development Kit
- JSDK 1.2 Java Servlet Development Kit
- Serveur HTTP avec extension Servlet
- JDBC dataBase Access
- Driver JDBC
- Serveur SGBD

JSP JAVA SERVER PAGES

- **Les JSP** sont une extension de l 'utilisation des Servlets, elles permettent d'inclure directement du code Java dans une page HTML..
  --Instructions Java : scriptlets
  --Utilisation des Beans
  --Utilisation de servlets existantes
- **Technologie** comparable aux ASP de Microsoft **(**avec une plus grande indépendance vis à vis des plateformes)

Principe des JSP

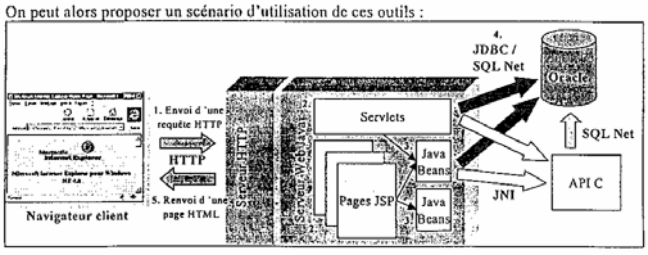
- Le serveur reçoit une demande de page JSP
- A partir du nom d'extension de fichier le serveur identifie que la demande concerne une page JSP.
- Routage de cette demande au processus de traitement des JSP
- Analyse de la page et génération d'une servlet (code java puis compilation)
- Renvoi de la page générée vers l'auteur de la demande

JSP

Le principe de fonctionnement des JSP est le suivant :

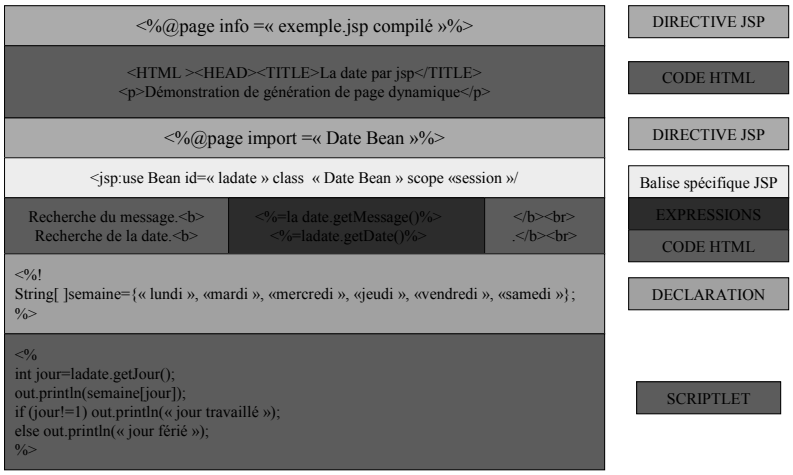
1. Le client envoie une requête HTTP concernant la page JSP concernée au Serveur Web qui redirige la demande vers le serveur applicatif.
2. Si la page JSP est invoquée pour la première fois, elle est compilée dynamiquement (en servlet) avant d'être exécutée.
3. La page JSP invoque un ou plusieurs Java Beans.
4. Les JavaBeans peuvent accéder à la base de données, exécuter des requêtes, utiliser des API, ...
5. La page JSP formate les informations reçues du Java Bean (résultats des requêtes) dans une page HTML renvoyée au client.

On peut alors proposer un scénario d'utilisation de ces outils :



ED IIS - DEA ISCE Intégration & Travail Coopératif 68

Exemple de page JSP: les éléments



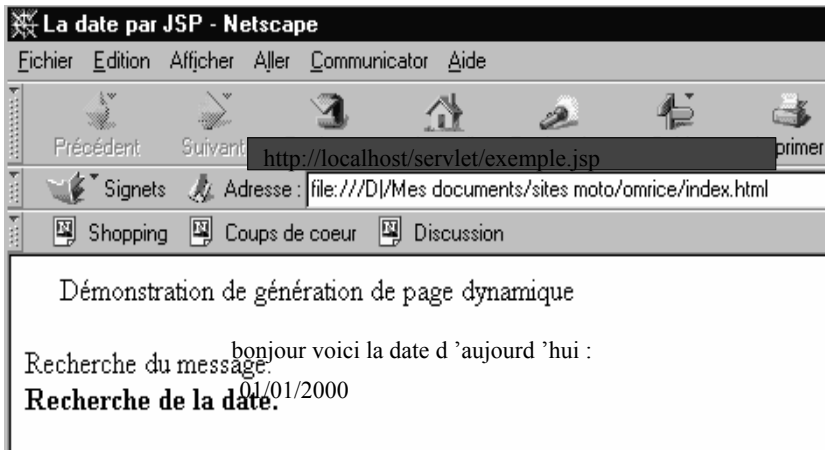
Référence à un fichier JSP

```
<HTML><HEAD>
<TITLE>Affichage de la date à l 'aide d 'une JSP</TITLE>
</HEAD>
<BODY>
<H1>Accès à la date à l 'aide de JSP
</H1>
La date du jour va etre:
<a href=« num/afficheDate.jsp » </a>
```

Programme Java de l 'exemple de JSP

```
Public class DateBean{
GregorianCalendar laDate=new GregorianCalendar();
public String getDate(){
return «  »+LaDate.get(Calendar.DAY_OF_MONTH)
+ «  \ »+(LaDate.get(Calendar.MONTH)+1)
+ «  \ »+LaDate.get(Calendar.YEAR);
}
public String getMessage(){
     return «  bonjour voici la date d 'aujourd 'hui :»;
     }
public String getJour(){
     return  laDate.get(Calendar.DAY_OF_WEEK);
     }
}
```

Page JSP générée



Lundi.jour travaillé

Les Enterprise Java Beans (EJB)

L 'architecture

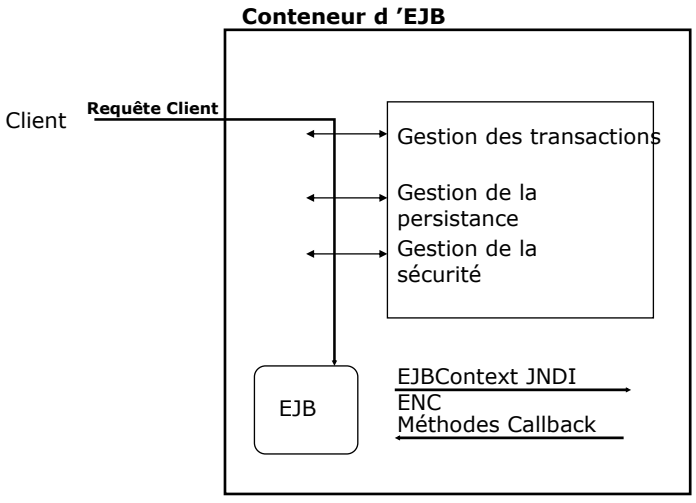
- Elle permet la création d ' **applications réparties**
- Elle utilise des **composants** exécutés sur une serveur appelé client distant

Ces composants n 'ont rien à voir avec les JavaBeans qui sont des composants situés coté client

Les objectifs

- **Rendre une application** facile à développer, déployer, administrer indépendamment de la plate forme permettant son exécution

Le conteneur d'EJB



ED IIS - DEA ISCE Intégration & Travail Coopératif 74

EJB: Les mécanismes du conteneur

- Les méthodes callback: Préviennent l 'Ejb d 'un événement concernant son cycle de vie.
- L 'objet EJBContext: Est une référence sur le conteneur permettant aux EJB d 'obtenir des informations sur l 'identité des clients,l 'état d 'une transaction...
- L'interface JNDI(Java Naming Directory Interface)

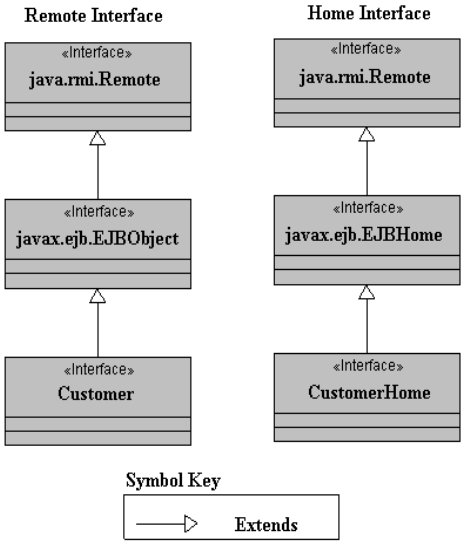
  ENC (Environment naming Context):

  Permet à un EJB d 'accéder à diverses ressources: des connexions JDBC,d 'autres EJB...

EJB: Les 2 interfaces

Interface distante:

Concerne les méthodes de l 'instance de l 'EJB



Class Diagram of Remote and Home Interfaces

Interface locale:

(cycle de vie de l 'EJB)

créer

détruire

trouver

Obtention d 'une référence sur l 'interface distante

EJB: Les 2 types d'EJB

Les **Session beans**

•Sont non persistants

•Associés à un seul client

•Un flot d 'exécution est créé pour:

a/chaque appel de méthode

stateless Session bean

b/plusieurs appels de méthodes en provenance du meme client

stateful Session bean

•Sont détruits après arrêt du serveur

Les **Entity beans**

•Sont persistants

•La persistance est gérée par:

l 'EJB lui meme **BMP**

(Bean Managed Persistance)

le conteneur **CMP**

(Container Managed Persistance)

• Représentent les données d 'une base de données

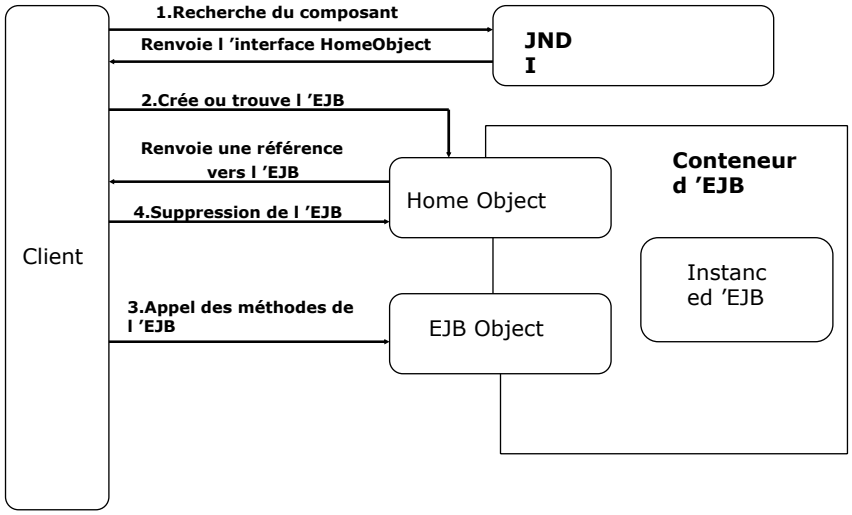
•Acceptent les accès multiples effectués par plusieurs clients

•Gestion d 'accès concourants

•Leur état est restauré après redémarrage du serveur

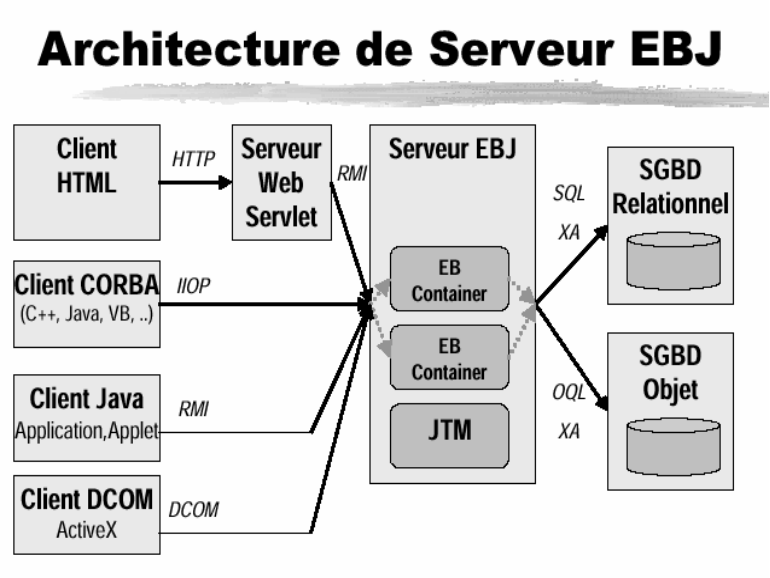
ED IIS - DEA ISCE Intégration & Travail Coopératif 77

Le conteneur d 'EJB vu par le client

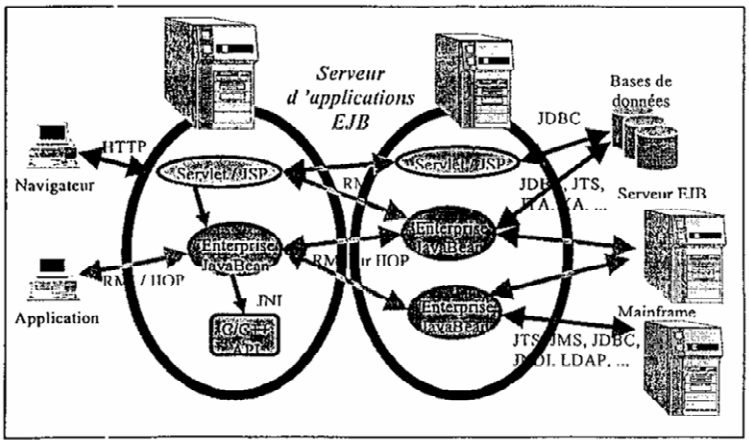


ED IIS - DEA ISCE Intégration & Travail Coopératif 78

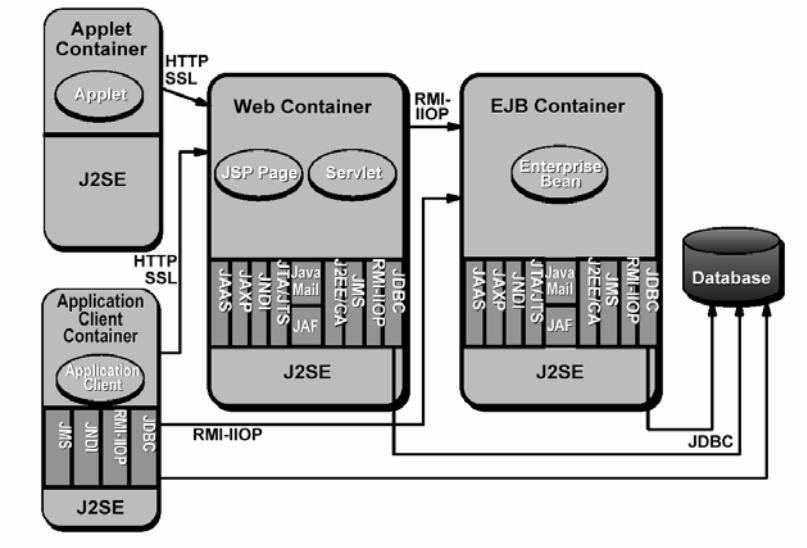
Les modes d'accès aux EJB



EJB – Entreprise Java Beans

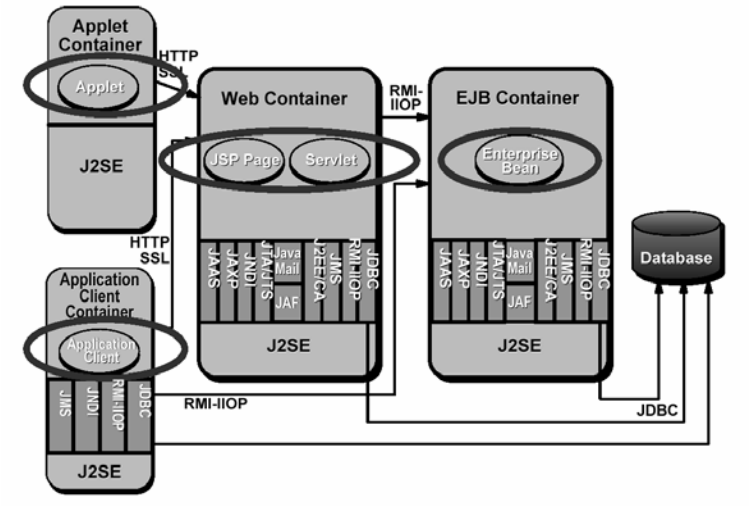


J2EE Platform



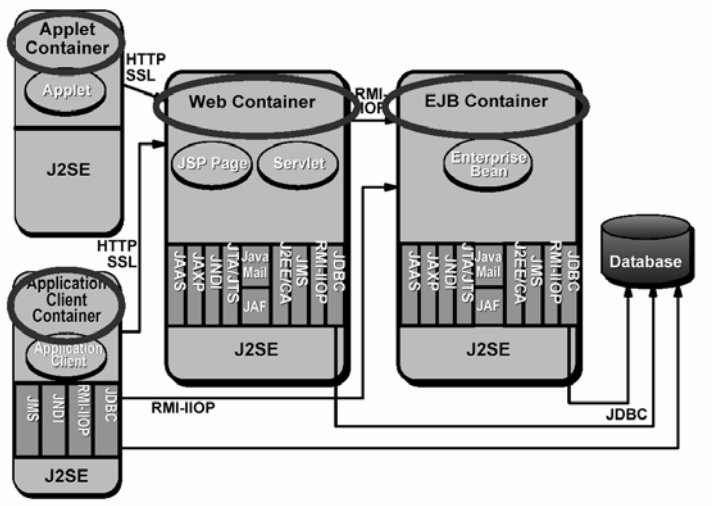
ED IIS - DEA ISCE Intégration & Travail Coopératif 81

J2EE Application Components

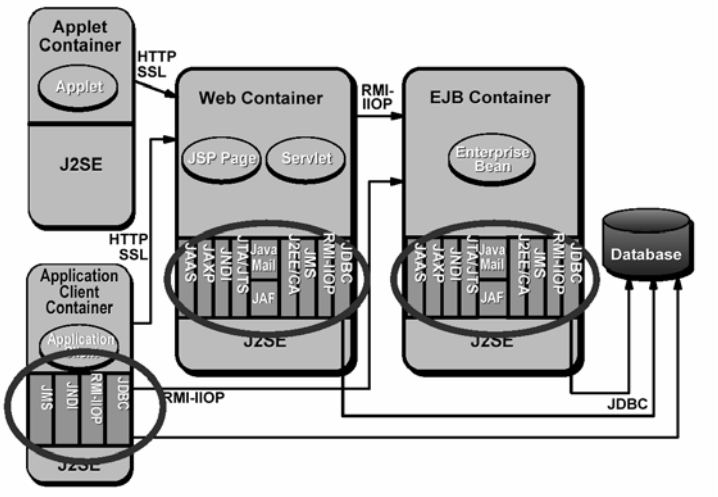


ED IIS - DEA ISCE Intégration & Travail Coopératif 82

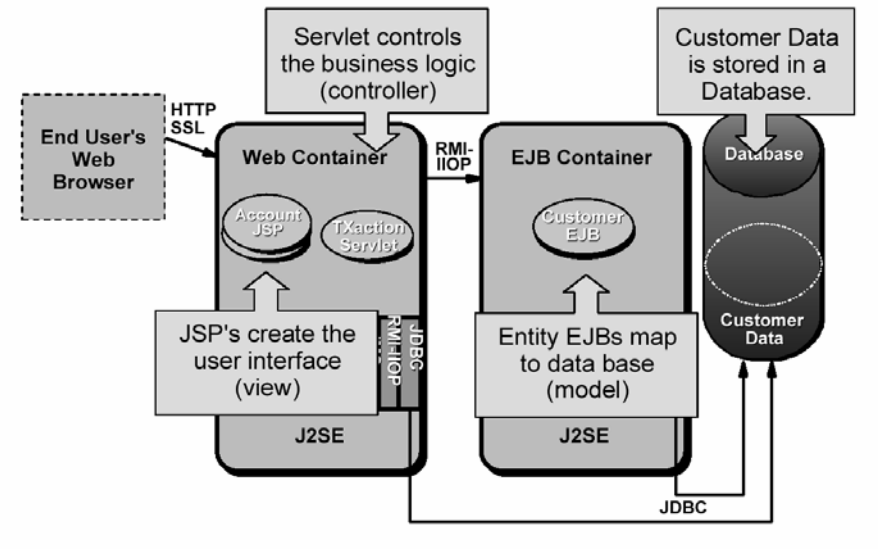
J2EE Containers



J2EE Standard Services



Server based J2EE application



ED IIS - DEA ISCE Intégration & Travail Coopératif 85

JDBC+RMI+SERVLET+JSP+EJB = Système Distribué 100% JAVA

L 'ensemble des technologies qui viennent d 'être présentées permet d 'affirmer que :

La **plateforme Java2** offre un ensemble cohérent de mécanismes permettant de diminuer la complexité associée au développement de **systèmes distribués** grâce à une architecture uniformisant le développements de **composants modulaires** pouvant être aisément déployés sur le **réseau.**

ED IIS - DEA ISCE Intégration & Travail Coopératif 86

Références

- Site Web :
  - http://www.com.sun.java : *Site officiel Java (JDK et doc.)*
  - http://www.javaworld.com : *Info sur Java*
  - http//www.gamelan.com : *applications, applets, packages, ...*
  - http://www.jars.com : *idem*
  - http://www.blackdown.com : *Java pour linux*
- *Livres :*
  - D. Flanagan :Java in a nutshell, O'Reilly.
  - P. Niemeyer, J. Peck : Exploring Java, O'Reilly.
  - B. Eckel. : Thinking in Java, http://www.EckelObject.com/Eckel

Qu'est-ce qu'un JavaBean ?

- Les JavaBeans sont des objets Java qui se comportent conformément à la spécification JavaBeans.
- Les JavaBeans (ou beans) sont des composants logiciels réutilisables qui peuvent être manipulés dans un environnement de développement comme VisualAge pour Java.
- Le modèle des signatures de méthode et de définition de classe d'un bean permet à des environnements tels que VisualAge pour Java de déterminer leurs propriétés et comportement.
- La faculté d'un environnement de beans à déterminer les caractéristiques d'un bean est appelée introspection.
- L'éditeur de composition visuelle, par exemple, permet de sélectionner les beans à partir d'une palette, spécifier leurs caractéristiques et connecter les beans entre eux.
- Les beans peuvent contenir d'autres beans ainsi que des connexions à des beans.
-

ED IIS - DEA ISCE Intégration & Travail Coopératif 88

Caractéristiques des beans (1)

- Les beans possèdent trois sortes de caractéristiques :
  - Evénements
  - Méthodes
  - Propriétés
- Un bean expose une caractéristique lorsqu'il rend cette caractéristique disponible pour les autres beans.

Caractéristiques des beans (2)

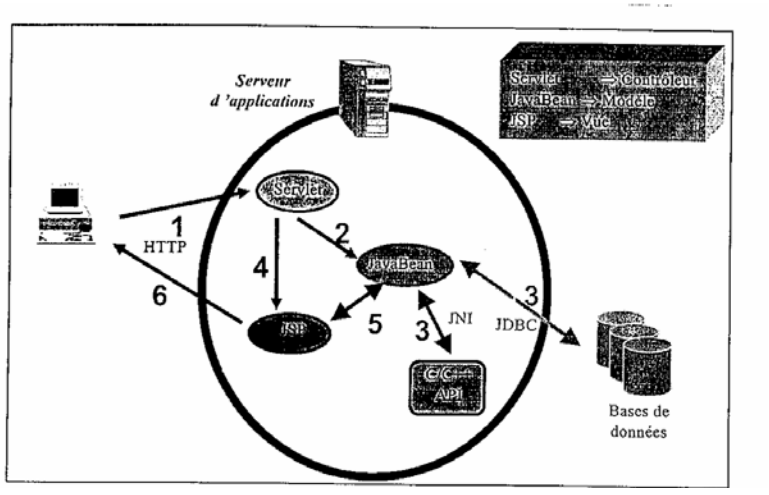
- Brève description de ces trois types de caractéristiques :

1. Evénements sont les événements que le bean génère. D'autres beans peuvent signaler que ces événements les intéressent et être prévenus lorsqu'ils se produisent.
2. Méthodes sont les actions qu'un bean expose pour que les autres beans les appellent. Les méthodes bean sont un sous-ensemble des méthodes publiques de la classe Java qui constitue le bean.
3. Les propriétés sont les attributs exposés par un bean. Les propriétés peuvent être lues, écrites ou les deux. Elles peuvent présenter les caractéristiques décrites ci-après.

Caractéristiques des beans (3) :

- Une propriété liée déclenche l'événement propertyChange lorsque sa valeur est modifiée.
- Une propriété contrainte permet à d'autres beans de déterminer si la valeur de la propriété peut être modifiée (elle déclenche l'événement vetoableChange).
- Une propriété indexée est un tableau. Elle expose donc des méthodes supplémentaires à des éléments d'adresse individuels.
- Une propriété cachée n'est pas visible. Elle est utilisable par des outils adaptés aux beans uniquement.
- Une propriété expert doit être manipulée uniquement par des utilisateurs experts.
- Une propriété normale est une propriété ni cachée, ni expert.

Modèle - MVC



Beans (1)

- L'interface publique d'un bean détermine les interactions avec les autres composants du même type. Elle se compose des éléments suivants :
  - Les Propriétés qui sont des données accessibles par d'autres beans. Ces données peuvent correspondre aux propriétés logiques d'un bean, comme le solde d'un compte, la taille d'un envoi, ou le libellé d'un bouton.
  - Les Evénements qui correspondent à des signaux qui indiquent que quelque chose s'est passé. L'ouverture d'une fenêtre ou la modification de la valeur d'une propriété, par exemple, déclenchent un événement.
  - Les Méthodes qui correspondent aux opérations qu'un bean peut exécuter. Elles peuvent être déclenchées par des connexions à d'autres beans.

Beans (2)

- Deux types de beans sont utilisés dans l'éditeur de composition visuelle :
  - Le bean visuel qui s'affiche dans le programme en cours d'exécution. Les beans visuels, tels que les fenêtres, les boutons et les zones de texte, constituent l'interface graphique utilisateur d'un programme.
  - Le bean non visuel qui n'apparaît pas dans le programme au moment de son exécution. Il représente un objet qui encapsule des données et implémente un comportement dans un programme.

Connexions

- Editeur de composition visuelle

  L'éditeur de composition visuelle est l'outil de programmation visuelle intégré de VisualAge for Java.

- Dans l'éditeur de composition visuelle, les connexions définissent les interactions entre les beans. On peut connecter des beans entre eux et des connexions entre elles.

- Une connexion est constituée d'une source et d'une cible. L'extrémité de départ de la connexion constitue la source de la connexion et l'extrémité d'arrivée la cible.

Classes BeanInfo

- Les beans peuvent être associés à des classes BeanInfo. Ces classes décrivent explicitement les événements, méthodes et propriétés exposés par un bean.

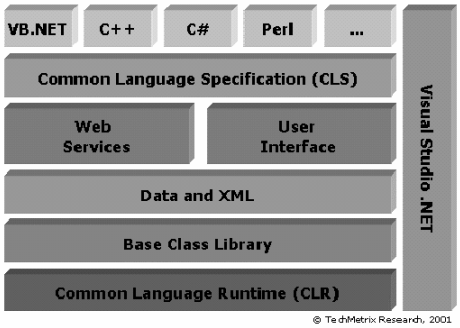
- VisualAge pour Java peut générer des classes BeanInfo pour les beans. Le nom de la classe BeanInfo est le même que le nom du bean avec le suffixe "BeanInfo".

- La classe BeanInfo contient des méthodes publiques qui renvoient des informations relatives au bean, notamment la classe du bean, le nom de cette classe, ainsi que des informations relatives aux événements, les méthodes et les propriétés du bean !!!

Références

- Site Web :
  - http://www.com.sun.java : *Site officiel Java (JDK et doc.)*
  - http://www.javaworld.com : *Info sur Java*
  - http//www.gamelan.com : *applications, applets, packages, ...*
  - http://www.jars.com : *idem*
  - http://www.blackdown.com : *Java pour linux*
- *Livres :*
  - D. Flanagan :Java in a nutshell, O'Reilly.
  - P. Niemeyer, J. Peck : Exploring Java, O'Reilly.
  - B. Eckel. : Thinking in Java, http://www.EckelObject.com/Eckel

L'architecture globale de MS.NET

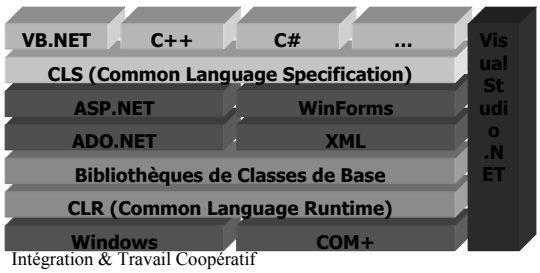


Les Web Services constituent une nouvelle vision très pertinente des services distribués. Ils permettent en effet de mettre des objets en ligne très simplement, avec une API totalement neutre (architecture et système d'exploitation réellement indifférents) aussi facile à programmer que XML.

Visual Studio.NET permet de concevoir des services ou des pages Web de la même manière que l'on concevrait une application classique.

La programmation autour de .NET supporte un grand nombre de langages, comme Visual Basic (VB.NET), Perl, Eiffel (Eiffel#), Python, JavaScript, C++… et même Java (J#).

MS.NET intègre un compilateur pour chacun de ces langages qui génèrera du MSIL.



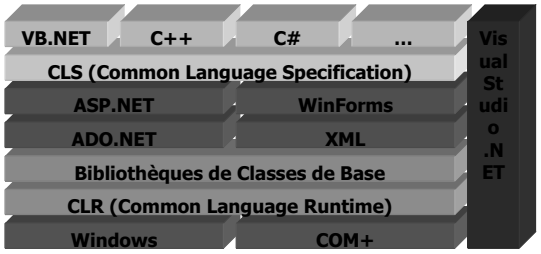
L'architecture de .NET

- Les composants de .NET s'organisent de la manière suivante :
  - C#, un nouveau langage orienté objet destiné à faciliter la programmation dans .NET, notamment les composants, qui intègre des éléments de C, C++ et Java en apportant quelques innovations comme les méta-données.
  - Un environnement d'exécution commun (*Common Language Runtime* - CLR) qui exécute un *bytecode* écrit dans langage intermédiaire (*Microsoft Intermediate Language* - MSIL ou IL). Du code et des objets écrits dans un langage quelconque peuvent être compilés en IL et exécutés par le CLR si un compilateur IL existe pour ce dernier.
  - Une grande bibliothèque de composants et d'objets de base accessibles par le CLR, qui fournissent les fondations pour écrire rapidement un programme (accès réseau, graphisme, accès aux bases de données).
  - ASP.NET, une nouvelle version d'ASP (Active Server Pages) qui supporte une véritable compilation en IL, alors qu'ASP était interprété auparavant. On peut également écrire les pages ASP dans n'importe quel langage disposant d'un compilateur IL.
  - Visual Studio .NET, une refonte de l'environnement Visual Studio et de Visual InterDev permettant aussi bien le développement d'applications et de composants classiques que Web.
  - WinForms et WebForms, un ensemble de composants graphiques accessibles dans Visual Studio .NET.
  - ADO.NET, une nouvelle génération de composants d'accès aux bases de données ADO qui utilise XML et SOAP pour l'échange de données.

ED IIS - DEA ISCE Intégration & Travail Coopératif 99

L'environnement d'exécution commun

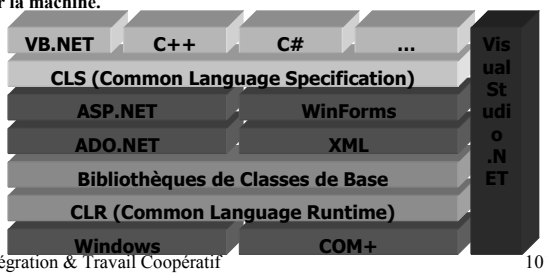
- **Tout comme Java, MS .NET est muni :**
-
  - **d'une machine virtuelle nommée CLR (Common Language Runtime)**
  - **d'un code intermédiaire, le MSIL (MicroSoft Intermediate Language).**
  - **Ainsi, ce code généré par compilation est indépendant de l'architecture sous-jacente et peut s'exécuter sur n'importe quelle machine comportant la machine virtuelle .NET.**
  - **Pour l'instant, seul les OS de Windows le supportent.**
  - **Le CLR comporte aussi un ramasse-miettes ainsi qu'un système de droits d'accès au code (sécurité).**



ED IIS - DEA ISCE Intégration & Travail Coopératif 100

Common Language Specification (CLS) – Common Type System (CTS)

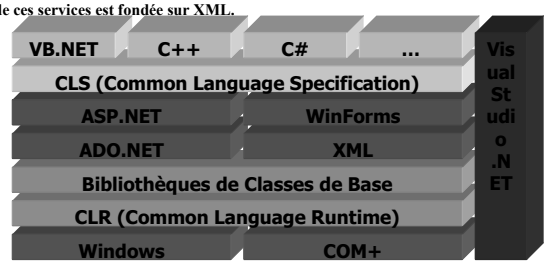
- Le langage MSIL présente des fonctionnalités intéressantes. Il constitue une base commune, un support à tous les langages compatibles avec l'environnement .NET, entraînant parfois l'ajout ou la suppression de quelques caractéristiques propres à chacun.
- Par exemple, la surcharge d'opérateurs est à présent possible en Eiffel#. En revanche, le debugger n'accepte plus l'héritage multiple du C++ ou le type *unsigned int*.
- Le concept de CLS (Common Language Specification, sous-ensemble commun à tous les langages) est issu de ce principe et offre des fonctionnalités comme un mécanisme commun de débuggage, la généricité, la surcharge d'opérateur, des gestions des exceptions ou la manipulation des pointeurs. La base commune contenant également tous les types admis, le transtypage ne pose plus de problèmes : en effet, les types sont communs à tous les langages modifiés (CTS : Common Type System).
- Les fichiers écrits en MSIL sont compilés à nouveau par un compilateur JIT en code natif pour donner un exécutable valide pour la machine.



ED IIS - DEA ISCE Intégration & Travail Coopératif 101

Les briques de base (Base Class Library)

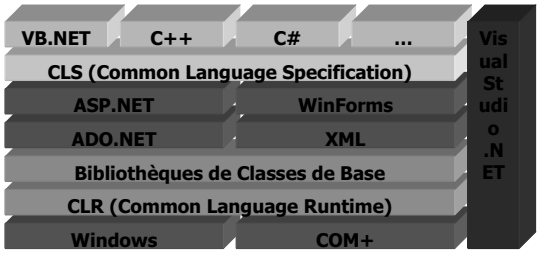
- Dans l'optique d'aider le développeur et de favoriser la réutilisation de composants, .NET propose une grande bibliothèque de classes, similaire en structure à celle de Java.
- Cette bibliothèque est commune à tous les langages, puisque écrite en MSIL, et rapproche d'autant les différents langages utilisés.
- Ainsi, dans une application, des composants en C++ peuvent faire appel à d'autres créés en Cobol…
- D'autre part, on remarque que dans cette bibliothèque commune, la gestion des fichiers XML est très présente.
- En parallèle à ce système de briques, Microsoft souhaite vendre ou louer des composants plus élaborés réutilisables pour constituer des applications Web complexes.
- L'interopérabilité forte de ces services est fondée sur XML.



ED IIS - DEA ISCE Intégration & Travail Coopératif 102

ASP.NET

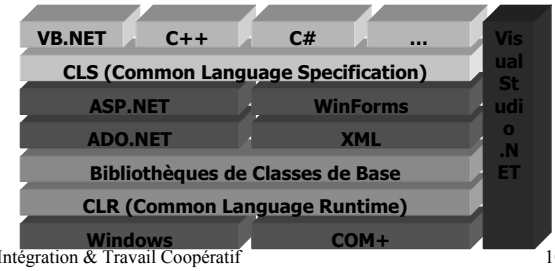
- ASP.NET est une nouvelle version d'ASP, un langage interprété produisant des pages HTML et WML très comparable à PHP en de nombreux points.
- La mise à jour introduit une vraie rupture dans le paradigme d'ASP avec l'ajout de deux nouveaux concepts comme la compilation des pages et les WebForms.
- Comme les Java Server Pages (JSP), le code ASP.NET est toujours compilé (en IL puis code natif avec le JIT) et peut être écrit dans un langage de haut niveau.
- ASP.NET supporte tous les langages du CLR, ce qui inclut C# et VB.NET.



ED IIS - DEA ISCE Intégration & Travail Coopératif 103

WebForms

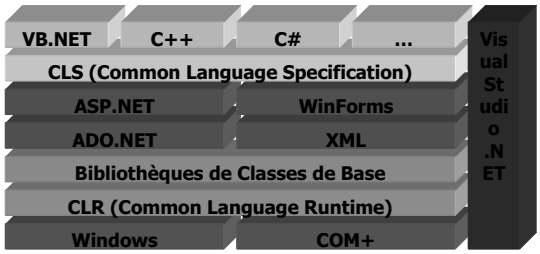
- Les WebForms sont une couche d'abstraction ajoutée pour permettre une programmation composite d'interface homme-machine orientée Web.
- Des composants génériques tels les formulaires, tableaux, boutons et zones de textes peuvent être assemblés afin de générer les pages ASP.NET.
- S'ils sont utilisés avec soin (aucun ajout de code HTML en dur par exemple), ces composants nommés WebForms répondent à deux problématiques des développeurs Web :
  - la gestion des différences entre les navigateurs et leurs nombreuses versions
  - et la reconnaissance à l'exécution du terminal et/ou du navigateur client afin d'utiliser un équivalent natif se rapprochant le plus possible de l'effet voulu.



ED IIS - DEA ISCE Intégration & Travail Coopératif 104

Visual Studio .NET

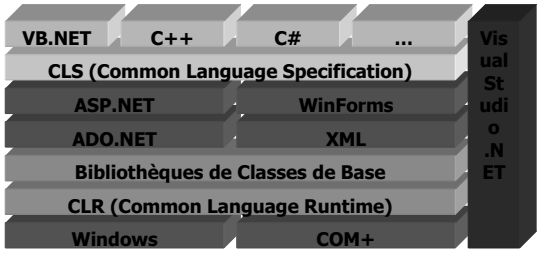
- **Microsoft Visual Studio .NET est l'outil de référence des développements en .NET.**
- **Cette nouvelle version de Visual Studio apporte réellement de nouvelles fonctionnalités intéressantes (comme UML) pour les développeurs Windows.**
- **En particulier, il est possible d'écrire et de déployer des applications Web en des temps très courts.**



ED IIS - DEA ISCE Intégration & Travail Coopératif 105

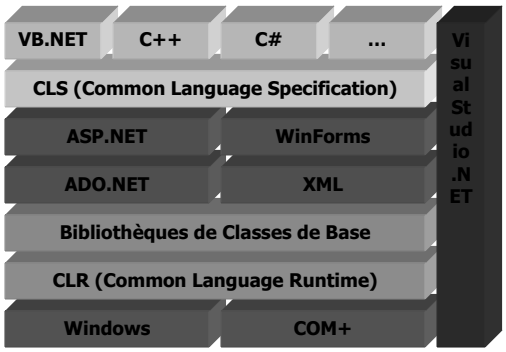
Les services Web et ADO.NET (Data & XML)

- Les services Web tels qu'ils sont introduits dans .NET sont une petite révolution dans l'informatique, non pas dans l'avance technologique, mais dans la simplicité de son approche, ce qui est intéressant dans un domaine réputé complexe :
  - **Un élément important de .NET appelé à être utilisé au sein des services Web est ADO.NET, un modèle orienté réseau pour les bases de données, qui permettra des prouesses comme la consultation et la mise à jour par HTTP et XML.**
  - **Conformément à l'esprit de .NET, cette évolution d'ADO (ActiveX Data Objects) met XML au cœur des bases de données qu'elles soient relationnelles ou pas.**
  - **L'innovation centrale d'ADO.NET est le DataSet, un modèle XML déconnecté des données.**
  - **Un objet DataSet peut ainsi effectuer des requêtes sur la base et traduire les résultats en XML.**
  - **Le grand intérêt réside dans le fait que les manipulations ultérieures sur le DataSet s'effectuent sans connexion à la base. L'API est très riche, elle permet notamment d'exprimer des contraintes, des relations entre tables, des insertions, des mises à jour…**
  - **L'objectif d'ADO.NET est à proprement parler de libérer les données en leur permettant d'être manipulées et échangées sur le Web avec une grande simplicité.**



ED IIS - DEA ISCE Intégration & Travail Coopératif 106

L'architecture de .NET Framework selon Microsoft



Essai de comparaison avec J2EE

- **L'un des objectifs évidents de Microsoft avec ce nouveau produit est de concurrencer Sun sur le marché des serveurs.**
- **C'est tout naturellement que .NET s'inspire largement de la technologie J2EE, référence dans le domaine.**
- **Il est intéressant d'établir un parallèle entre les deux architectures pour voir si .NET a su reprendre les points forts de J2EE et palier à ses défauts.**
- **La comparaison que nous effectuons ici concerne tous les domaines, du langage de programmation à l'agencement des différents éléments d'architecture.**

Le modèle Objet : Les langages C# et Java (1/3)

- **Les langages Java et C# sont respectivement les langages de référence de J2EE et .NET. Le tout nouveau C# s'appuie en partie sur Java, en partie sur C++, ce qui explique les ressemblances.**
- **Ce que C# a repris à Java :**
  - **les interfaces sont des classes totalement abstraites,**
  - **l'héritage est simple, mais plusieurs interfaces peuvent être implémentées pour la même classe,**
  - **les variables doivent être initialisées par défaut,**
  - **le concept de packages / espaces de nom est très présent pour regrouper et classer les classes dépendantes,**
  - **on retrouve également les mêmes attributs de visibilité (public, private),**
  - **les classes intérieures, classes définies à l'intérieur des classes,**
  - **la gestion des exceptions,**
  - **la génération de documentation…**
  - **en ce qui concerne les méta-données, les deux langages proposent une API permettant de manipuler des attributs sur la classe en cours (capacité d'introspection), mais il est possible dans .NET d'ajouter d'autres attributs de méta-informations,**
  - **le multithreading est présent des deux côtés mais bien mieux intégré dans Java (communication, synchronisation),**

Le modèle Objet : Les langages C# et Java (2/3)

- **En revanche, C# innove dans les aspects suivants :**
  - Les événements : **Chaque classe peut recevoir ou exporter des événements.**
  - Système de type unifié : **A la différence de Java, les types primitifs comme int et float héritent de Object, ce qui évite l'encapsulation en Java par les classes comme Int et Float.**
  - Les fonctions déléguées (*delegates*) : **Elles sont un équivalent au typage sûr des références aux fonctions de C ou de C++.**
  - Les propriétés (*properties*) : **Elles contribuent à simplifier la syntaxe en encapsulant les accesseurs (les get et set parfois désagréables de Java) dans la déclaration d'un membre.**
  - Les structures (*struct*) : **elles sont similaires aux classes, sans héritage, et sont allouées sur la pile. Elles contiennent directement leurs valeurs (*value type*) à la différence des classes classiques dont on ne manipule en réalité que le pointeur vers une instance. L'usage des structures permet d'améliorer la performance du code produit.**
  - Les types énumérés (*enum*) : **Restaurés de C++, ils permettent d'associer des noms avec des valeurs numériques, ce qui évite les static final int FIELD de Java.**

Le modèle Objet : Les langages C# et Java (3/3)

- La surcharge d'opérateur (*operator overloading*) : **Héritée de C++, elle simplifie également la syntaxe pour la manipulation des objets.**
- La généricité (*templates*) : **C# reprend le principe des *templates* de C++. Pour rendre cela possible, le code MSIL n'a pas, comme Java, de fonction AddInt ou AddFloat mais une fonction Add générique (*type neutral*) qui est remplacée par du code natif approprié en fonction du contexte.**
- Les méthodes virtuelles **: Comme en C++, les méthodes ne sont pas virtuelles par défaut, c'est-à-dire qu'il faut déclarer une méthode virtual dans la classe mère et ajouter le mot clé override dans la méthode redéfinie pour pouvoir surcharger – c'est exactement le contraire en Java. Outre un gain faible en performance, ceci permet de voir clairement quelles méthodes peuvent et sont redéfinies.**
- Les indexers : **Les objets implémentant l'interface IEnumerable peuvent voir leur contenu parcouru par le mot clé foreach, ce qui simplifie également la syntaxe.**
- Syntaxe de passage des paramètres riche : **Les mots clés in, out, ref et params permettent un interfaçage aisé avec les autres langages.**
- Les espaces de nommage (*namespaces*) : **Les paquetages (*packages*) de Java qui étaient très fortement liés à une structure sous forme de répertoire sont remplacés par une notion plus souple d'espace de nommage.**

Les machines virtuelles : CLR et JVM

- **Les composants de base sont les mêmes.**
- **Le langage de programmation est compilé en un langage intermédiaire (*bytecode* / *MSIL*), et sont destinés à fonctionner dans une machine virtuelle.**
- **On note également l'existence d'un ramasse-miettes, d'un gestionnaire de sécurité, de multithreading.**
- **Mais le Java bytecode est interprété alors que MSIL est recompilé.**
- **Le CLR supporte plusieurs langages précompilés, dont ceux qui ne sont pas orientés objet, et permet la création de composants partagés.**
- **Il faut également insister sur la portabilité qui est valable surtout chez Java.**

Les bibliothèques de données

- **Les API proposés pour les deux langages (On part du principe que .NET correspond surtout à C#) sont très semblables.**
- **Il faut reconnaître que .NET propose un panel de fonctions de manipulation de SOAP et XML très étendu.**

L'archivage, les composants

- **Java et .NET comportent des composants permettant le déploiement des archives.**
- **La gestion de la propriété et des versions est en revanche visible uniquement du côté de l'outil de Microsoft.**
- **L'interopérabilité des composants est présente dans les deux camps, mais les EJBs restent un modèle beaucoup plus sûr et plus mûr.**

Le remoting (accès à distance)

- **Le remoting est suffisamment bien implémenté dans les deux camps pour le rendre simple d'utilisation.**
- **Mais .NET a l'avantage car il est plus flexible en ce qui concerne son extensibilité.**
- **Il permet également les invocations asynchrones.**

Du côté Web : ASP.NET / JSP

- **ASP.NET et JSP sont toutes deux écrites en langage de haut niveau (Java pour JSP, les langages du CLS pour ASP.NET) et sont compilées (en bytecode JVM pour Java, en IL puis code natif pour ASP.NET) pour accélérer l'accès. La syntaxe et les concepts sont similaires.**
- **Mais, surtout dans ce domaine, on peut dire que ASP.NET est orienté langage car il autorise un très grand nombre de scripts (pour chaque langage compatible .NET) alors que JSP est orienté plate-forme puisqu'il fonctionne sur de nombreux serveurs.**
- **Les WebForms apportent une couche d'abstraction par rapport au langage cible de présentation (HTML, WML…) dont JSP ne dispose pas.**

Connexion à la base de données : JDBC / ADO.NET

- **Si dans les deux plates-formes, on dénote bien le découplage entre le modèle de données concret et l'utilisateur (en utilisant DataSet et ResultSet), on s'aperçoit que :**
  - **ADO.NET se base sur XML et travaille toujours en non-connecté. A l'aide de HTTP et XML, les services Web et ADO.NET permettent de mettre en ligne des services, de partager et mettre à jour des bases de données simplement et indépendamment de l'architecture cible.**
  - **L'architecture J2EE est moins ouverte de par sa dépendance envers Java (RMI, messagerie Java). Un pont CORBA existe, mais n'est pas directement utilisable.**
  - **Le modèle de données de JDBC est connecté à la différence d'ADO.NET.**

WinForms et Java Swing

- **Bien que fortement orientés réseau, .NET et J2EE permettent le développement d'applications graphiques classiques.**
- **Swing et WinForms sont orientés-objets en respectant le paradigme Modèle-Vue-Contrôleur (MVC).**
- **Après longue maturation (échec de AWT), Swing fonctionne de manière presque similaire sur toutes les plates-formes cibles.**
- **WinForms est pour l'instant restreint à Windows uniquement.**

Les terminaux mobiles

- **Les deux architectures comportent également un support pour terminaux mobiles**
  - ➔ **.NET Mobile SDK pour .NET (limité à Windows CE pour l'instant)**
  - ➔ **J2ME (Micro Edition) pour Java.**

Bref…

- **Malgré la différence d'âge entre les deux technologies, elles ne présentent pas de différences majeures ni de révolution.**
- **C'est dans une optique d'approche du problème que .NET fait preuve de plus de simplicité. Mais si ce dernier se démarque franchement de ce que Microsoft nous a habitué, il reste à voir si le produit saura se vendre sur le marché.**
- **Il est clair que, avec ce nouveau produit, Microsoft propose un moyen de construire des Web Services qui s'appuient sur les technologies les plus en vogue comme XML et SOAP et en reprenant certains points forts de J2EE.**
- **Cette offre se veut avant tout developer-friendly, parce qu'il prétend pouvoir supporter une vingtaine de langages. En revanche, la portabilité laisse à désirer, et reste encore à l'état de promesse.**
- **Il semble donc que .NET est un produit qui concurrencera Java dans le domaine des serveurs et surtout des Web Services dès qu'il sera complet.**
- **L'architecture du .NET Framework est semblable à celle de J2EE, et l'outil Visual Studio .NET permet d'agir à tous les niveaux. Il est donc facile et rapide de construire des Web Services, mais la maîtrise du développement en est d'autant plus limitée.**
- **Mais, laissons le temps montrer de lui-même les qualités et les défauts de Microsoft .NET, ses succès et ses échecs.**

Bibliographie

- ***Jean-François Bobier Microsoft .NET : architecture et services*Mémoire d'option IDL, www.developpez.biz/downloads/dotnet/MemoireDotNet.pdf**
- ***Markus Völter, CTO, Mathema AG / Michael Stal, SIEMENS : Comparing J2EE with .NET*Présentation PowerPoint, www.voelter.de/data/presentations/J2EE_vs_NET_MV.ppt**
- ***Kevin Hoffman, Jeff Gabriel, Denise Gosnell, Jeff Hasan, Christian Holm, Ed Musters, Jan Narkiewickz, John Schenken, Thiru Thangarathinam, Scott Wylie, Jonothon Ortiz : .NET Framework Professionnel,* Edition Wrox Press France SARL**
- ***IBM Competitive Technology Laboratory : J 2EE vs. .NET Comparison,* Presentation Powerpoint, ftp://ftp.software.ibm.com/software/websphere/partners/roadshow/scene1-j2ee-dotnet.pdf**

OLE - COM - DCOM ActiveX

- En construction

Serveurs d'applications : CGI, JSP, EJB

- En construction