CENTRALE
L Y O N

Architectures

En cours de construction

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

BTD/GL/Archi 1

CENTRALE
L Y O N

Bertrand DAVID : Génie Logiciel

Pourquoi des architectures ?

- **Mettre en évidence un « cadre » de travail**
- **Identifier des grands principes organisationnels et de manipulation**
- **Dégager une démarche de travail**
- **Accompagner cette démarche par des outils**

BTD/GL/Archi 2

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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**

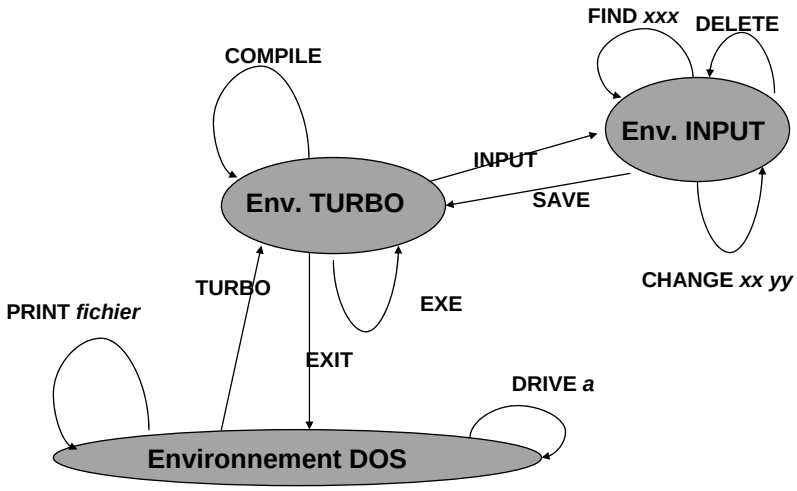
BTD/GL/Archi

3

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Spécification à l'aide de graphe



BTD/GL/Archi

4

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Grands principes

- Grammaire de suivi et de reconnaissance
- Grammaire d'action

- Approche moteur
- Exécution dirigée par les données

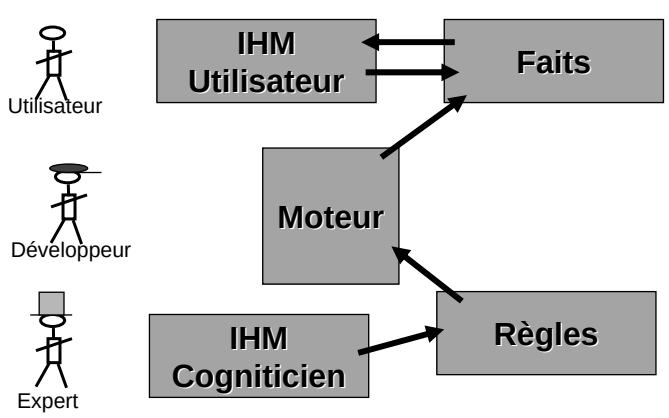
BTD/GL/Archi

5

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Systèmes experts



BTD/GL/Archi

6

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Grands principes

- Séparation entre identification de grands principes réutilisables (règles) et d'éléments caractérisant une situation particulière (faits)
- Séparation entre expression déclarative et traitement algorithmique (moteur)
- Différents niveau d'intervention :
  - ➔Création d'un système support : 1 fois
  - ➔Création d'une base de règles : pour chaque nouveau domaine d'application
  - ➔Création d'une base de faits : pour chaque nouveau cas

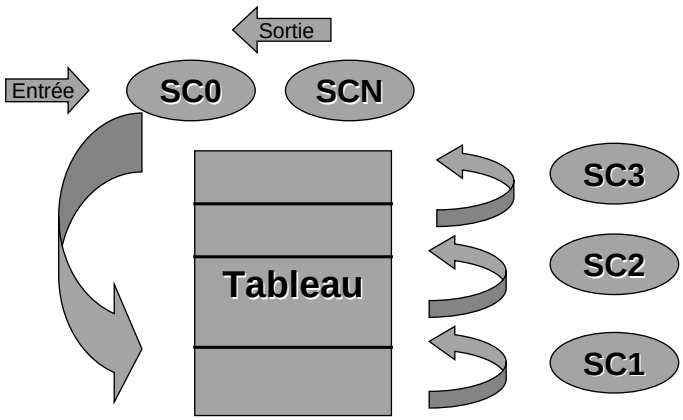
BTD/GL/Archi 7

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Architecture black board

- SCi :Source de Connaissance i



BTD/GL/Archi 8

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Grands principes

- **Organisation à responsabilités clairement définies**
- **Chaque SC (source de connaissance) a des droits de lecture (d'observation) et d'écriture (d'action) sur des zones précises**
- **Processus d'observation et d'action avec changement de zone**
- **Autonomie des acteurs**
- **Existence ou non d'un coordonnateur explicite**

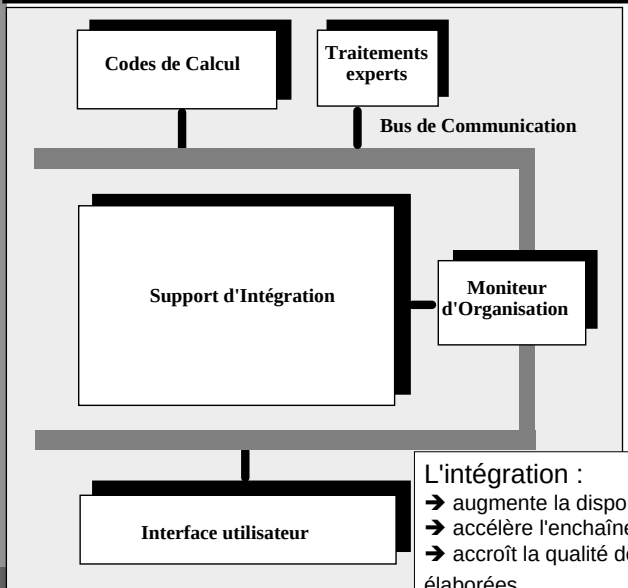
BTD/GL/Archi

9

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.

BTD/GL/Archi

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Grands principes

- **Séparation des 4 aspects :**
  - ➔**IHM**
  - ➔**Coordination**
  - ➔**Gestion des données**
  - ➔**Traitements**
- **Dépendance versus indépendance**
- **Ouvert ou fermé**

BTD/GL/Archi

11

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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, …)**

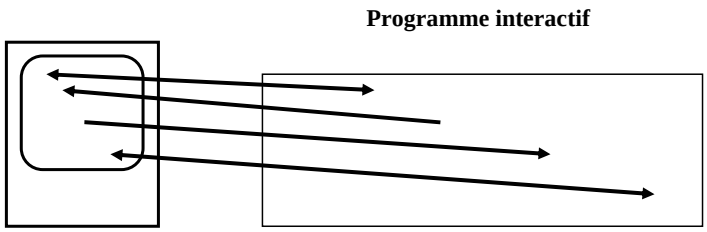
BTD/GL/Archi

12

CENTRALE L Y O N

Bertrand DAVID : Génie Logiciel

Architecture magmatique



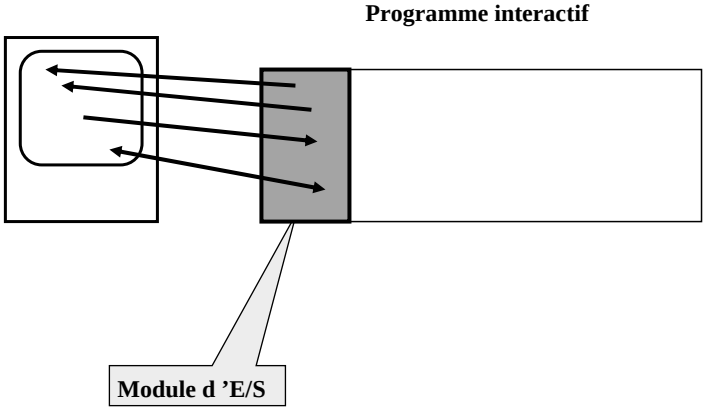
BTD/GL/Archi

13

CENTRALE L Y O N

Bertrand DAVID : Génie Logiciel

Module Entrées/Sorties



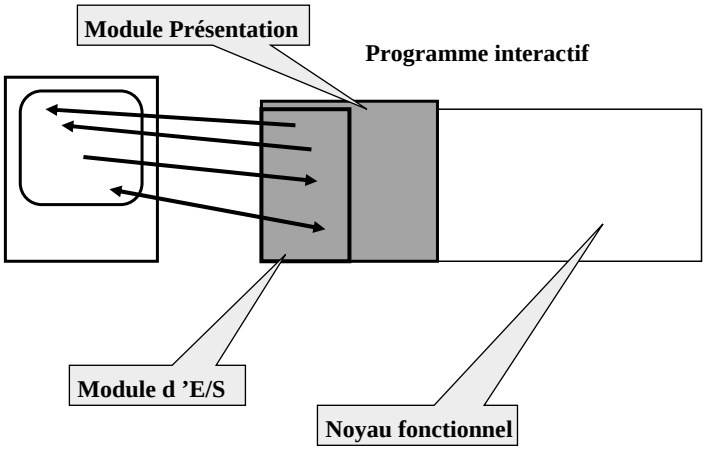
BTD/GL/Archi

14

CENTRALE L Y O N

Bertrand DAVID : Génie Logiciel

Module Présentation



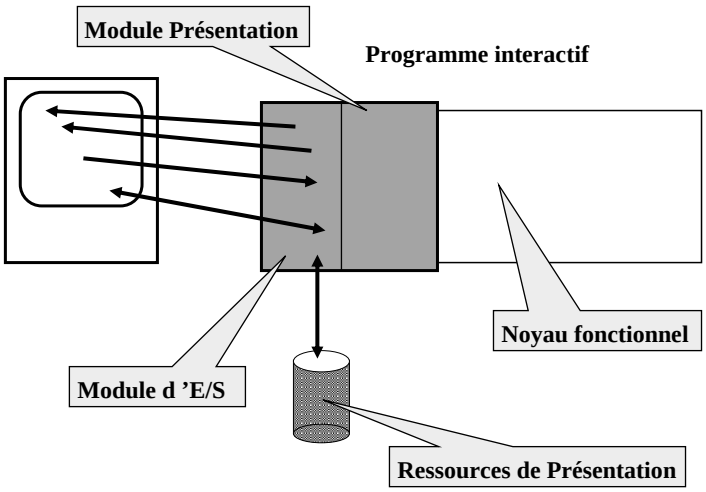
BTD/GL/Archi

15

CENTRALE L Y O N

Bertrand DAVID : Génie Logiciel

Ressources de présentation



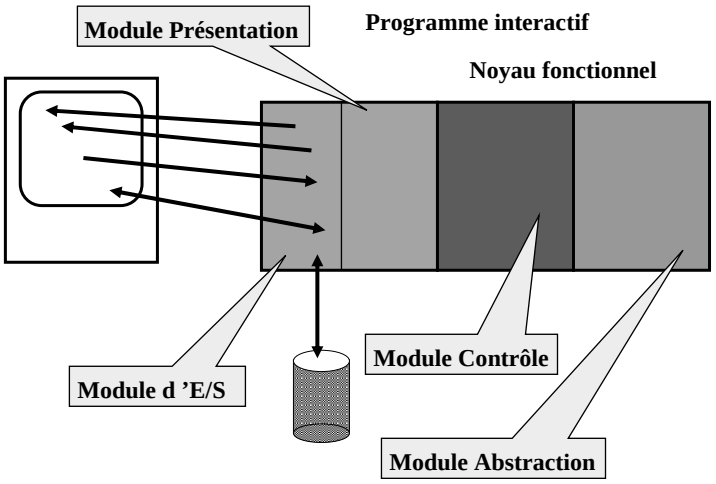
BTD/GL/Archi

16

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Module Contrôle



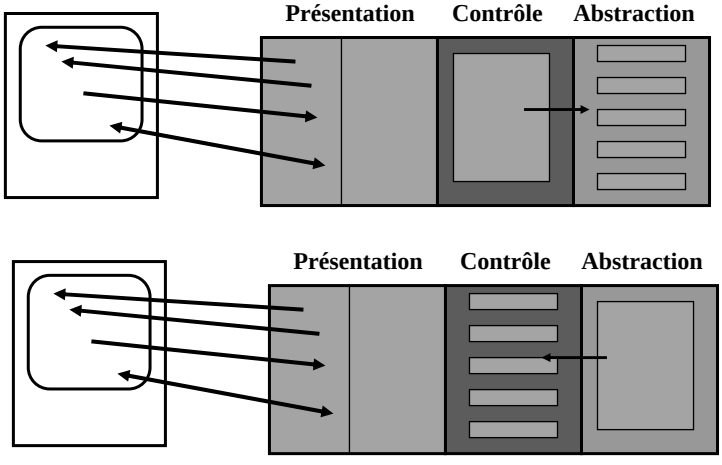
BTD/GL/Archi

17

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Contrôle Interne /Contrôle Externe



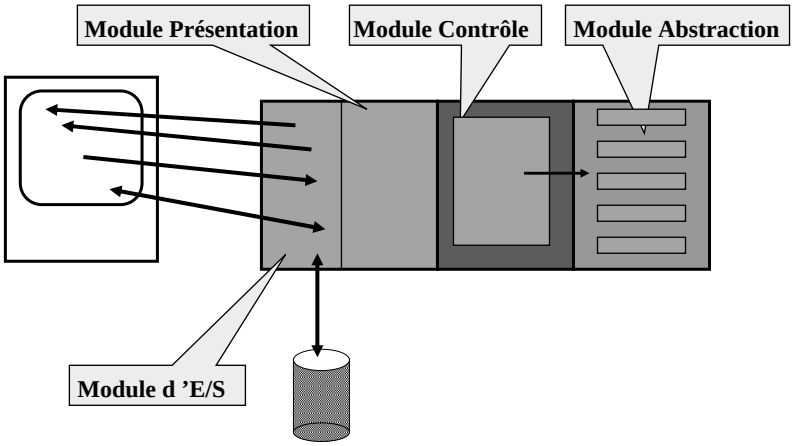
BTD/GL/Archi

18

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Contrôle Interne /Contrôle Externe



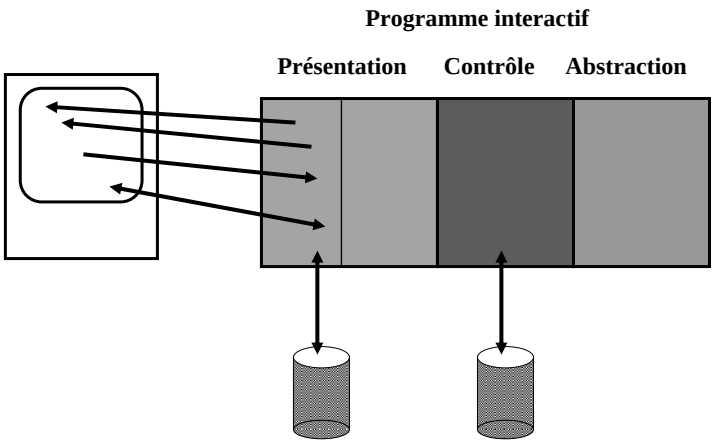
BTD/GL/Archi

19

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Ressources de contrôle



BTD/GL/Archi

20

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Portabilité - adaptabilité

BTD/GL/Archi

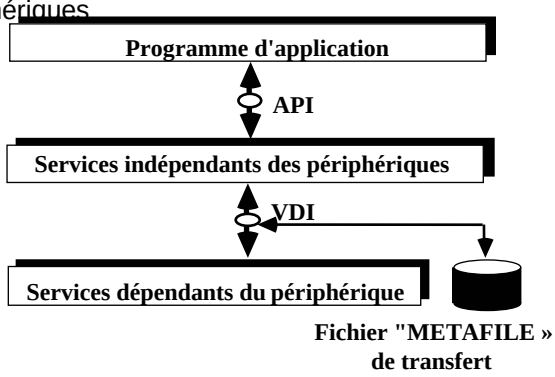
21

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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)

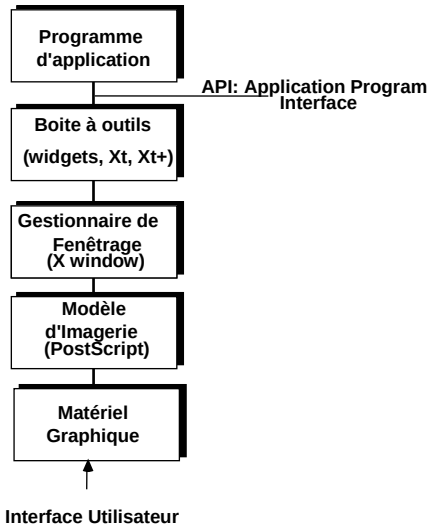
BTD/GL/Archi

22

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Pipeline d'affichage



BTD/GL/Archi

23

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.

BTD/GL/Archi

24

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.

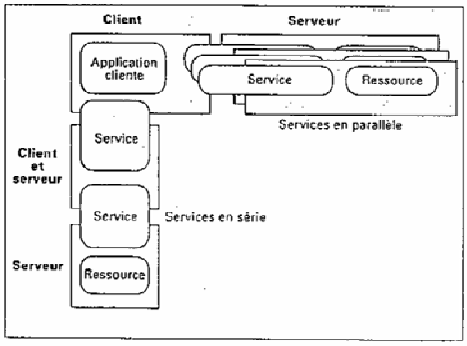
BTD/GL/Archi

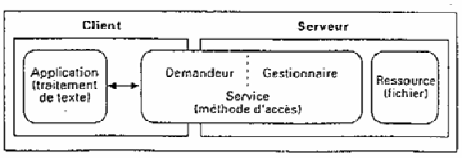
25

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Client - Serveur





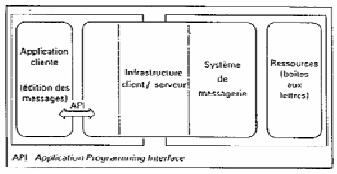
BTD/GL/Archi

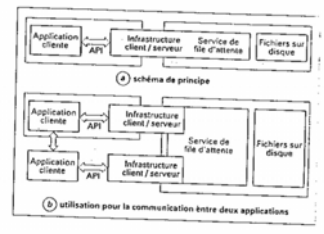
26

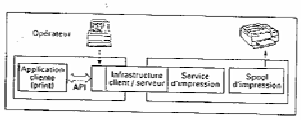
CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Client - Serveur







BTD/GL/Archi

27

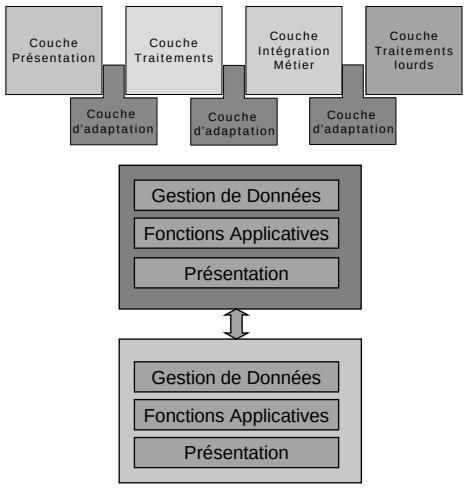
CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Client - Serveur

- **Architecture**

•4 couches logiques
•3 couches d'adaptation



BTD/GL/Archi

28

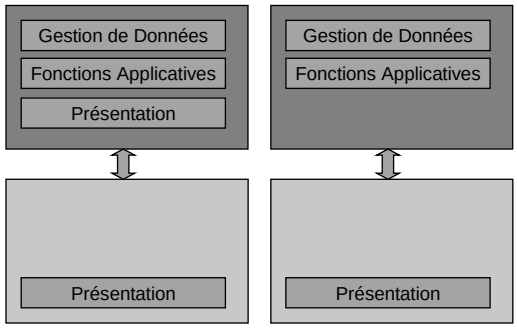
CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.**



BTD/GL/Archi

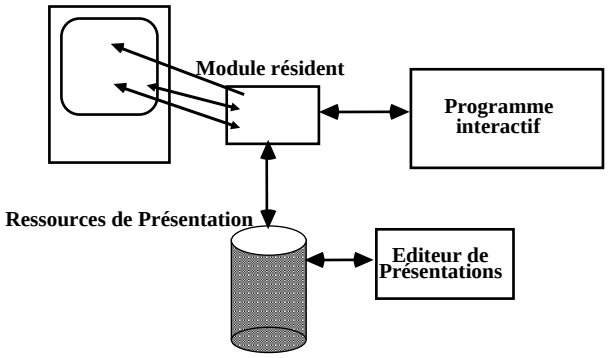
29

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Amélioration externe

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



BTD/GL/Archi

30

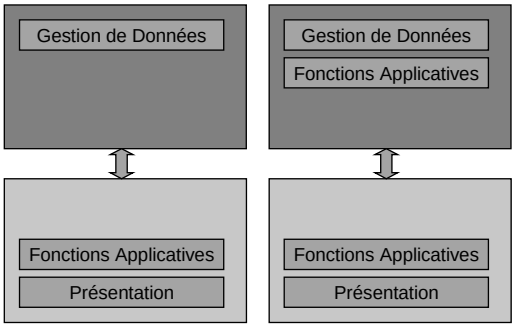
CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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).**



BTD/GL/Archi

31

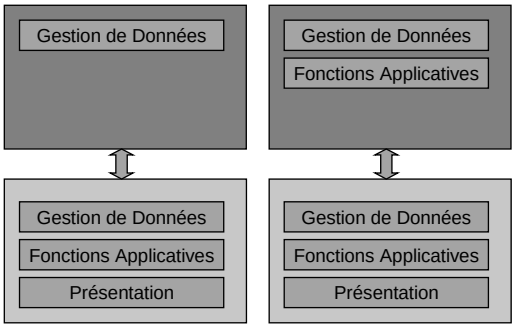
CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.**



BTD/GL/Archi

32

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.

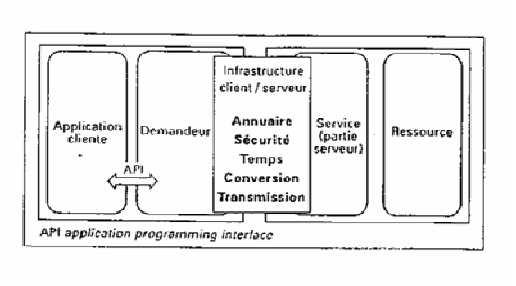
BTD/GL/Archi

33

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Client - Serveur



BTD/GL/Archi

34

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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**

BTD/GL/Archi

35

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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**

BTD/GL/Archi

36

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.

BTD/GL/Archi

37

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.**

BTD/GL/Archi

38

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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

BTD/GL/Archi

39

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.

BTD/GL/Archi

40

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.

BTD/GL/Archi

41

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.**

BTD/GL/Archi

42

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.**

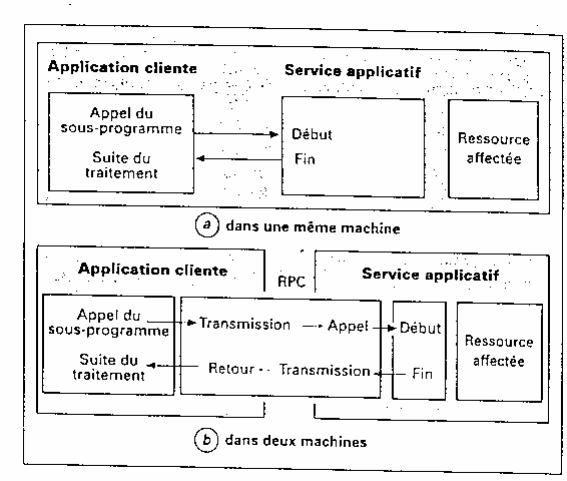
BTD/GL/Archi

43

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Client - Serveur



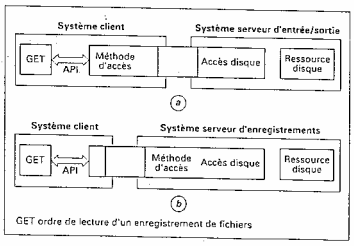
BTD/GL/Archi

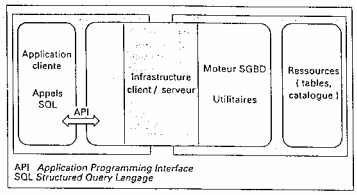
44

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Serveur de fichiers et de BD





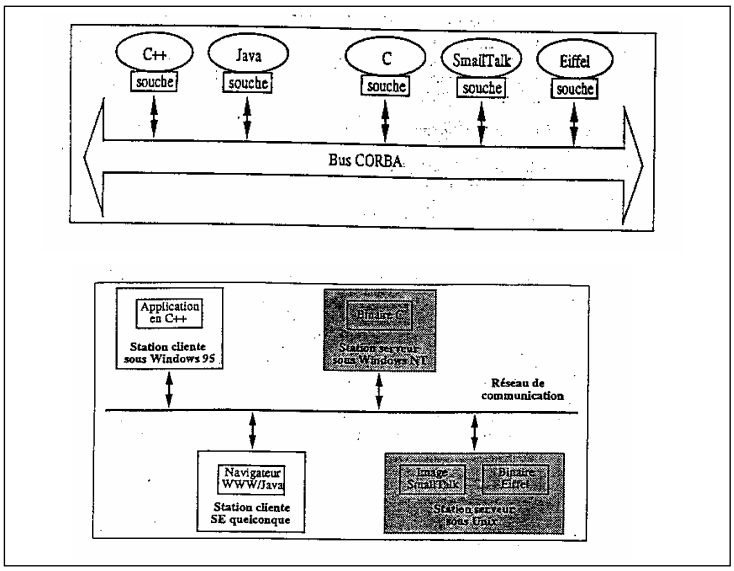
BTD/GL/Archi

45

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

CORBA (1/4)



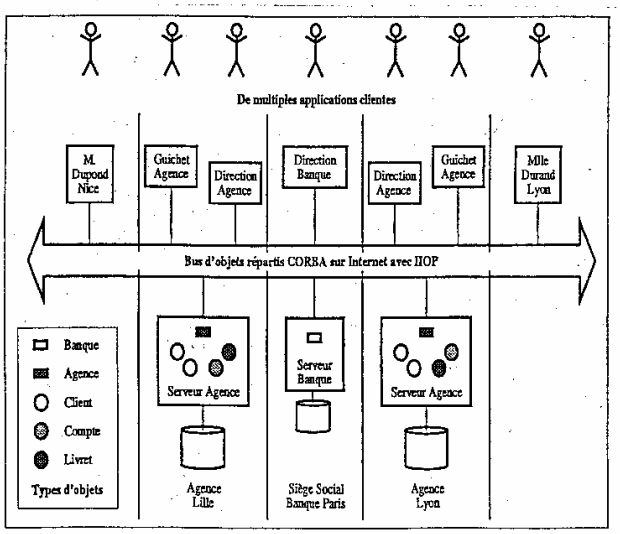
BTD/GL/Archi

46

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

CORBA (2/4)



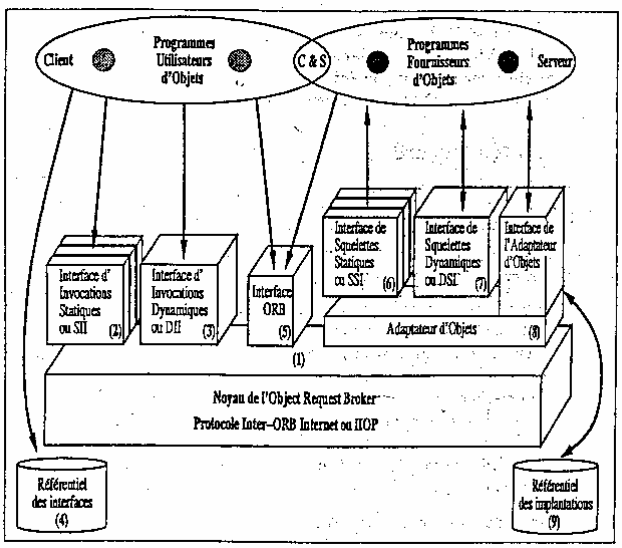
BTD/GL/Archi

47

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

CORBA (3/4)



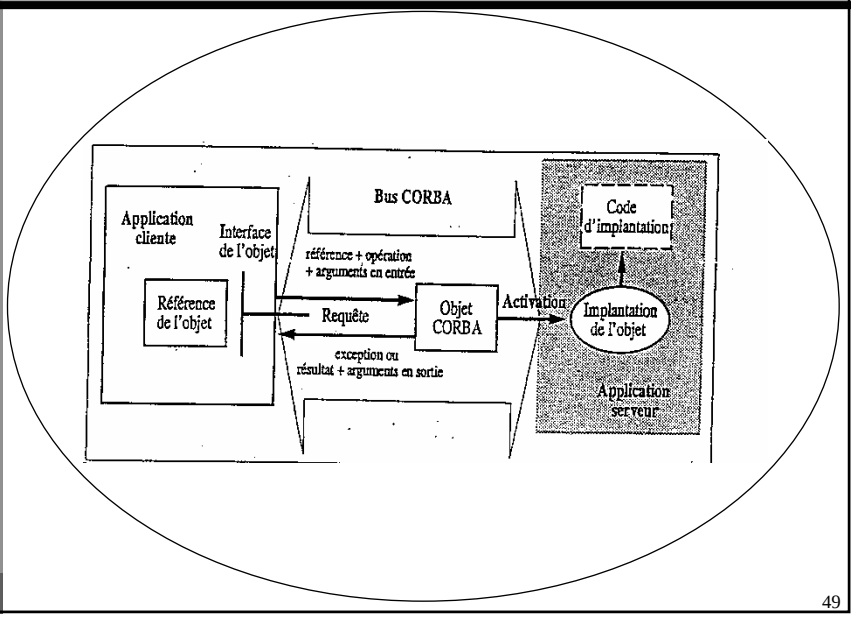
BTD/GL/Archi

48

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

CORBA (4/4)



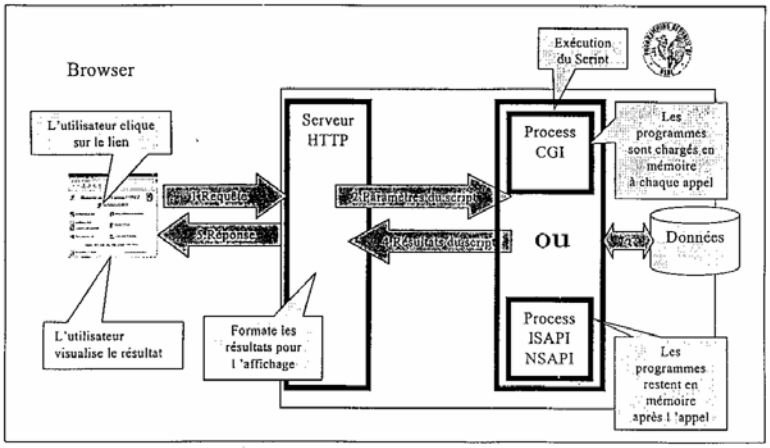
BTD/GL/Archi

49

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Application Web



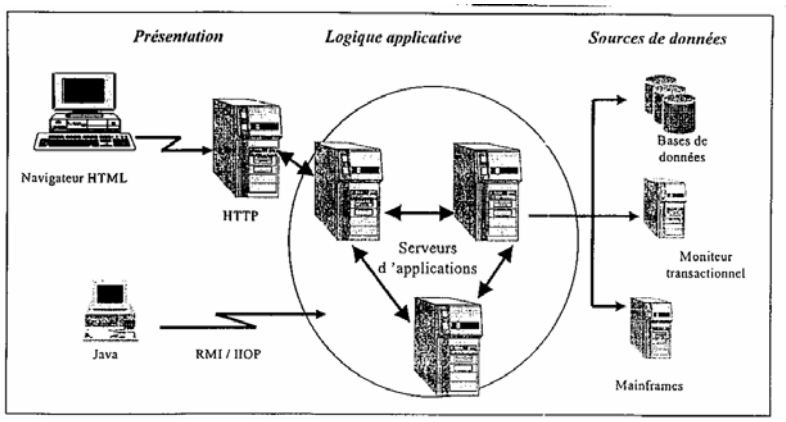
BTD/GL/Archi

50

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Client-Serveur multi-tiers



BTD/GL/Archi

51

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Java - ses particularités

- **Java** supporte la programmation parallèle **Programmation de processus concourants par l'intermédiaire de threads**
- **Java s**upporte 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
  et permet le développement d'applications conséquentes.

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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**

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.

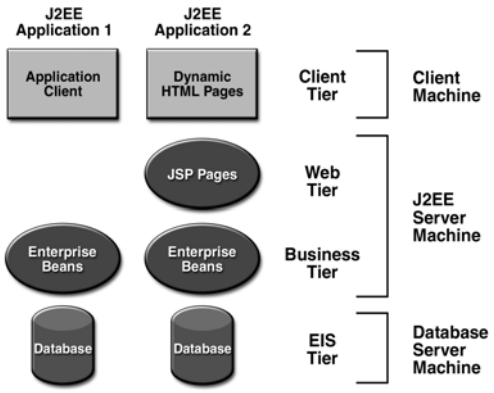
BTD/GL/Archi

57

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

J2EE: Les applications distribuées multi-tiers

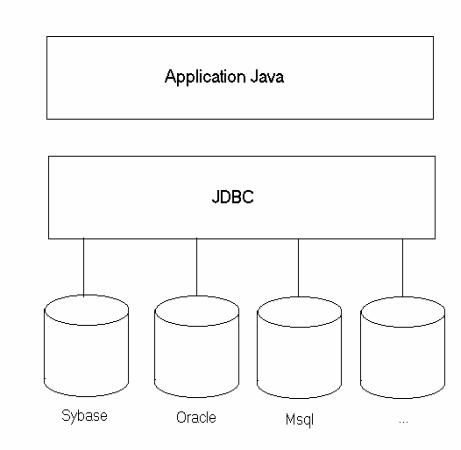


CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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**



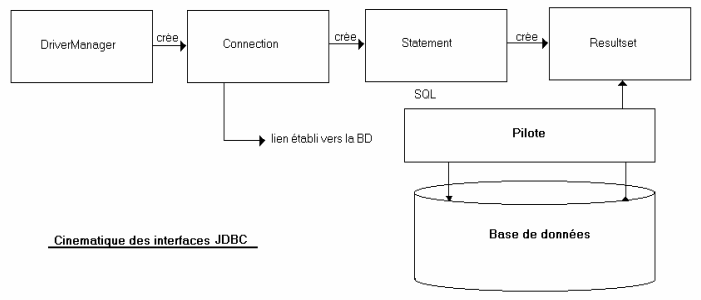
CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Mise en œuvre de JDBC

Les différentes étapes :

1. **Importer le package java.sql**
2. **Enregistrer le driver JDBC**
3. **Etablir la connexion à la base de données**
4. **Créer une zone de description de requête**
5. **Exécuter la requête**
6. **Traiter les données retournées**
7. **Fermer les différents espaces**



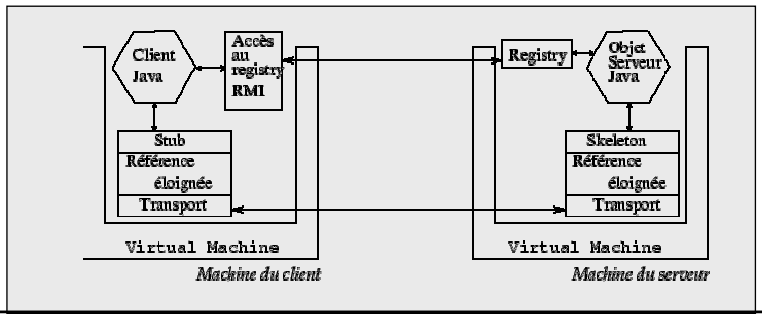
Cinematique des interfaces JDBC

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

RMI (Remote Method Invocation)

- **RMI (*Remote Method Invocation*) désigne un mécanisme qui permet la conception d'applications distribuées basées sur la technologie Java.**
- **Grâce à RMI, deux entités Java localisées chacune dans une machine virtuelle distincte peuvent communiquer.**
- **Dans le modèle client-serveur classique, le client est un programme Java tournant sur une machine et le serveur est composé d'un ou plusieurs objets distants.**



BTD/GL/Archi

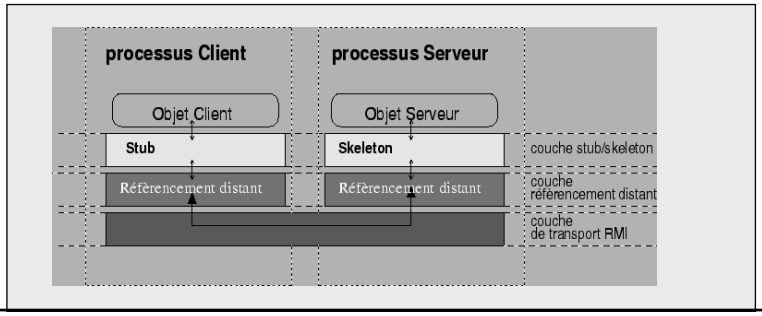
61

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Architecture et fonctionnement

- **Pour invoquer une méthode d'un objet serveur, le client Java interroge un outil qui se comporte tel un serveur de noms. Le *Registry* fournit la référence et l'interface de l'objet serveur. Alors, un mécanisme permet au client d'invoquer une méthode de l'objet distant comme s'il était local.**
- **Lors de cet appel local, l'objet s'adresse en fait à une instance de la classe *stub* générée par la compilation RMI. Cet objet possède exactement la même interface que l'objet distant. Le stub sérialise les paramètres et les envoie en même temps qu'il invoque la méthode. Ainsi, le client croit manipuler le véritable objet.**
- **Le *skeleton* a un rôle symétrique au stub du côté serveur, il dé-sérialise les paramètres, invoque la méthode puis retourne la réponse.**



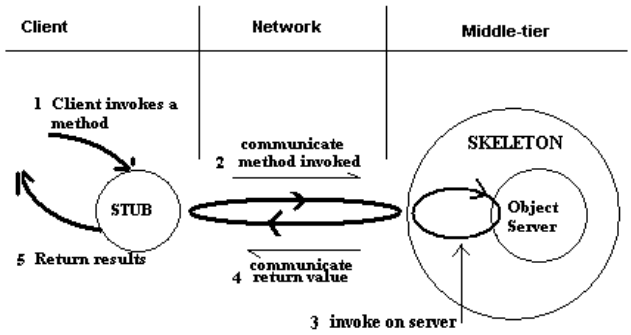
BTD/GL/Archi

62

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Rmi (Remote Method Invocation)



CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Architecture en trois couches

- Lors de l'exécution d'une application distribuée par RMI, on distingue trois couches dans l'architecture, comme on peut le voir sur la figure ci dessous.
- La couche stub/skeleton qui offre les interfaces que les clients et serveur utilisent pour interagir ;
- la couche de référencement éloigné qui constitue le *middleware* entre la couche stub/skeleton et le protocole de transport. Cette couche s'occupe de la création et du maintien des références à des objets distants ;
- la couche de protocole de transport est un protocole de bas niveau d'envoi et réception de requêtes sur le réseau.

BTD/GL/Archi

64

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Sérialisation

- **L'objet stub possède une référence interne vers l'objet distant qu'il représente et propage la requête à travers la couche de référencement distant. Il sérialise les arguments et demande au bus d'envoyer la requête et ses arguments au bon objet. La sérialisation est obligatoire, elle consiste en la conversion des objets locaux dans une forme portable de manière à pouvoir les transmettre physiquement sur un réseau.**
- *Fonctionnement d'une application distribuée*
- **Un serveur est indépendant des clients et les clients sont indépendants les uns des autres. La plupart du temps, ces programmes tournent sur des machines virtuelles différentes. Dans la conception séquentielle, l'exécution d'une opération concernant un objet métier implique création d'une instance de cet objet. En RMI, il n'est pas possible de faire un *new* sur un objet distant. En effet, les applications clientes et serveurs tournent sur des processus distincts et ont des espaces mémoire différents.**
- *Généralités*
- **Les objets qui transitent sur un réseau doivent être sérialisables. Un objet est sérialisable si sa classe implémente l'interface *Serializable*. Par exemple, un *String* l'est, car il est une instance de la classe *String* qui implante l'interface *Serializable*. En fait, toute variable de type simple, comme *Int* est par nature sérialisable.**

BTD/GL/Archi

65

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Conception client-serveur 2-tiers

- **L'idée est de transférer tous les objets qui concernent les données et le fonctionnel de l'implantation séquentielle du côté serveur. Seule l'interface utilisateur reste du côté client.**
- **Deux techniques permettent à l'interface utilisateur de communiquer avec les objets de l'application côté serveur selon deux considérations.**
- **La première et la plus simple à réaliser est d'utiliser un objet intermédiaire, le ``*conteneur*''. Le côté client possède l'interface de cet objet et le côté serveur l'implantation. Le client invoque les méthodes de son interface et du côté serveur, le conteneur invoque les méthodes des autres objets. C'est uniquement à lui que s'adresse le client, et son rôle est ensuite de gérer l'exécution de l'application.**
- **La deuxième consiste en la distribution des objets métier sur le bus de communication. Ainsi, pour chacun d'eux, le client possède une interface et invoque les méthodes par son intermédiaire.**

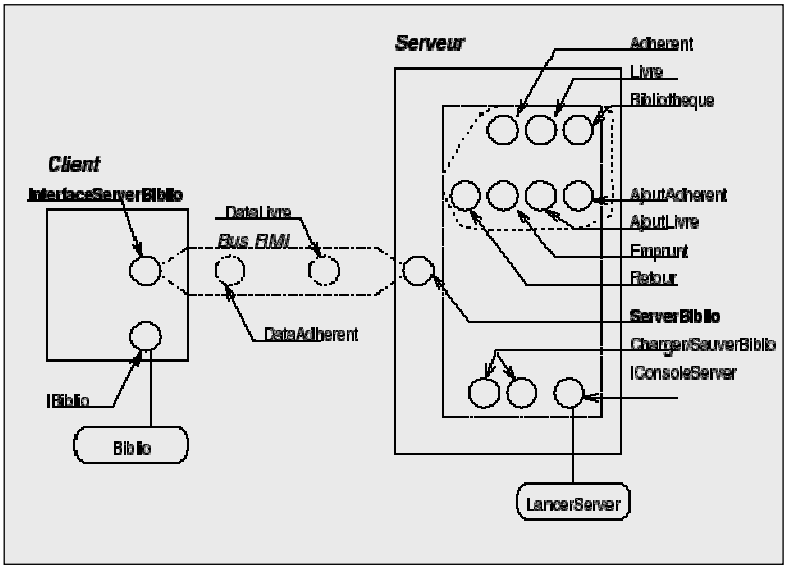
BTD/GL/Archi

66

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Client-serveur RMI avec conteneur



BTD/GL/Archi

67

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Utilisation d'un objet « conteneur »

- *Principe*
- **L'idée est de transférer les objets de l'implantation séquentielle du côté serveur. Seule l'interface utilisateur reste du côté client.**
- **Ici, l'objet *ServerBiblio* remplace l'interface utilisateur du modèle centralisé.**
- **Les objets qui transitent sont les renseignements que fournissent les utilisateurs. Ces données doivent être encapsulées dans des objets sérialisables *DataAdhérent* et *DataLivre* visibles des deux côtés. Après transmission sur le réseau, elles sont désérialisées du côté serveur par l'objet *ServerBiblio*. Celui-ci interroge l'objet *Bibliotheque* pour retrouver les bonnes instances des objets *Adhérent* et *Livre*, ou en créer de nouvelles si elles n'existent pas. Cela garantit l'intégrité des données. Ces objets sont indispensables, car il n'est plus possible comme en séquentiel de créer une nouvelle instance à distance.**

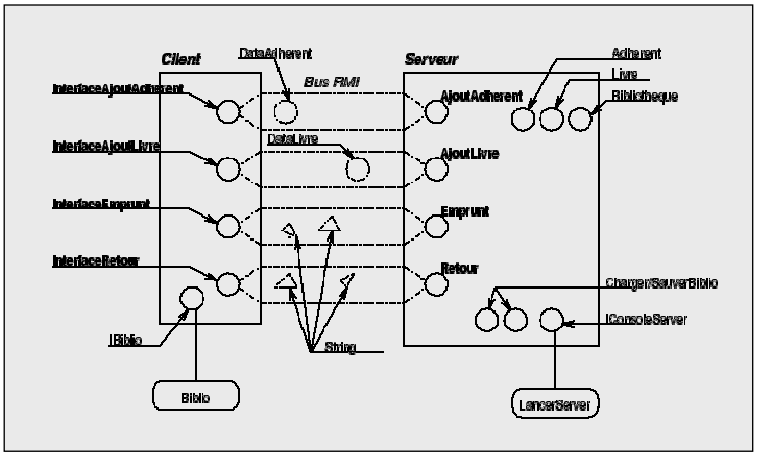
BTD/GL/Archi

68

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Distribution des objets métier Client-serveur RMI sans conteneur



BTD/GL/Archi

69

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Discussion sur les deux schémas

- **Le cas du conteneur est à priori plus facile à concevoir. Le protocole est simple, le client invoque des méthodes de l'interface du conteneur. Celui-ci s'occupe de propager la requête aux objets de l'application qui sont du côté serveur.**
- **Dans le cas du développement sans conteneur, chaque objet peut être déployé sur une machine différente. Alors, le client dispose de chaque interface et s'adresse au bus afin de communiquer avec chaque objet. Il est ainsi possible, en 3-tiers, d'avoir des machines spécialisées qui accueillent les objets métier pour lesquels elles ont été conçues.**

BTD/GL/Archi

70

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Conception client-serveur 3-tiers

- **Principe**
- **La conception d'applications client-serveur à trois niveaux avec RMI reprend les mécanismes de la conception à deux niveaux.**
- **Entre chacun des niveaux client, logique de traitement et serveur de données, l'une des deux propositions précédentes est applicable suivant les besoins techniques et logiques.**
- **L'idée est d'obtenir un niveau réellement indépendant des niveaux client et serveur de données. Dans celui-ci, la logique de traitement est encapsulée dans des objets qui peuvent évoluer et même être complètement changés sans mettre en péril le reste de l'application.**

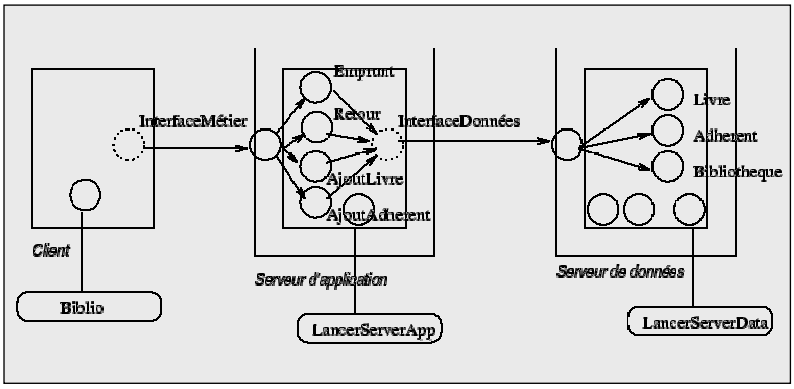
BTD/GL/Archi

71

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Client-serveur 3-tier RMI avec deux conteneurs



BTD/GL/Archi

72

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Exemple : bibliothèque

- **Dans l'exemple de la bibliothèque, la partie cliente ne contient que l'interface utilisateur. Un menu demande à l'utilisateur quelle opération sur la bibliothèque il veut effectuer. La partie serveur de données est constituée des objets données *Livre*, *Adhérent* et *Bibliotheque*. La partie logique de traitement concerne tous les objets métier.**
- **Les trois parties sont indépendantes et des interfaces permettent d'accéder aux objets de chacune d'elles. Si les interfaces ne sont pas modifiées, on peut changer radicalement le contenu technique ou logique sans mettre en péril l'application toute entière.**

BTD/GL/Archi

73

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Communication entre les trois niveaux

- **Les besoins en terme de communication dépendent de l'application à concevoir. Dans l'exemple de la bibliothèque, il y a deux sortes d'actions. Des actions de création de livres ou d'adhérents, et des actions plus complexes comme l'emprunt et le retour de livres.**
- *Les objets méthodes proches de la logique de traitement*
- **Les objets méthodes qui contiennent la logique de traitement sont amenés à être modifiés suivant les besoins de l'entreprise qui possède le système d'information. Le client invoque leurs méthodes par leurs interfaces ou grâce à un objet conteneur.**

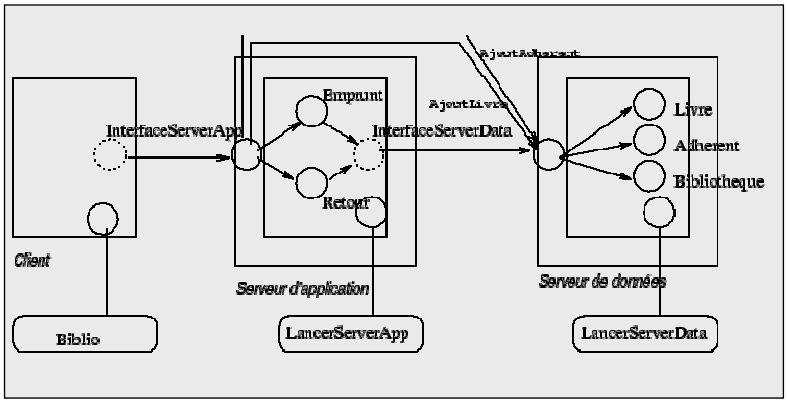
BTD/GL/Archi

74

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Client-serveur 3-tier RMI avec deux conteneurs simplifié



- *Les objets méthodes proches des données*
- **Les objets *AjoutLivre* et *AjoutAdherent* ne contiennent pratiquement pas de logique métier et s'apparentent plus à des méthodes liées aux données. Le fait de mettre ses objets sur un niveau intermédiaire n'apporterait qu'une lourdeur dans la conception et l'administration.**

BTD/GL/Archi

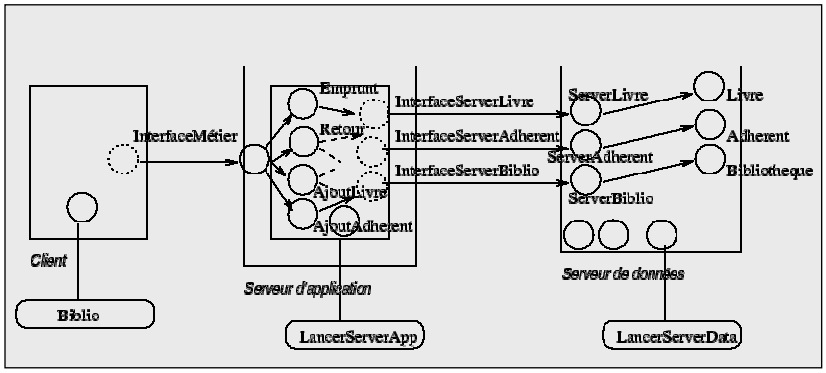
75

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Client-serveur 3-tier RMI avec un conteneur simplifié

- **Une solution de facilité voudrait que chaque objet donnée ait une interface dans le serveur de traitement de la logique métier.**



BTD/GL/Archi

76

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Approche 3-tiers avec le protocole de communication RMI

- Le protocole RMI permet une première découpe en trois entités serveur communicantes : la partie cliente, le serveur de traitement et le serveur de données.
- L'application présentée est très simple, mais les mécanismes énoncés sont applicables à plus grande échelle. Néanmoins la maintenance d'une application de très grande taille peut être ardue. En effet, pour chaque objet distant auquel un client veut accéder, il faut fournir une interface RMI. Il semble que la solution d'un objet conteneur soit la plus simple à réaliser et à administrer.
- La méthode présentée s'adapte bien à cette technologie. La séparation entre les objets fonctionnels et les objets de stockage semble naturelle.

BTD/GL/Archi

77

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

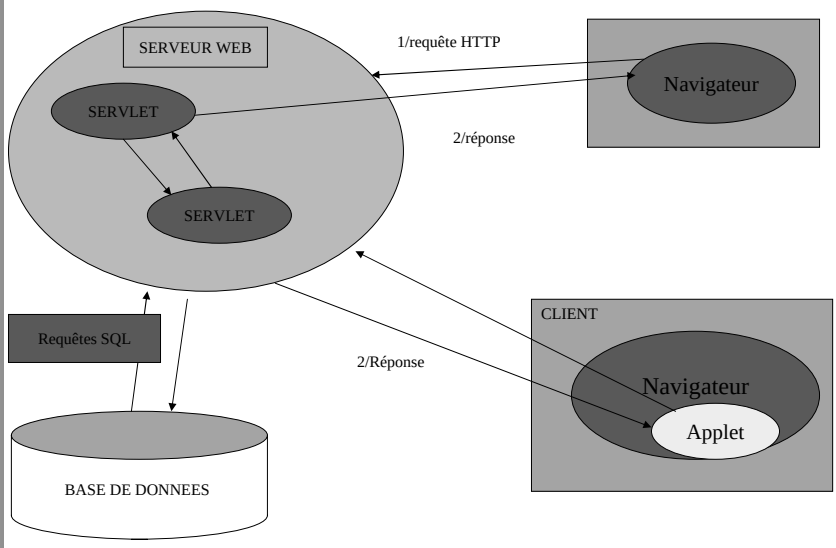
Servlet et génération de pages dynamiques Comparaison servlets/Applets

- 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

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Les Servlets et leur accès



BTD/GL/Archi

79

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.**

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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>

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

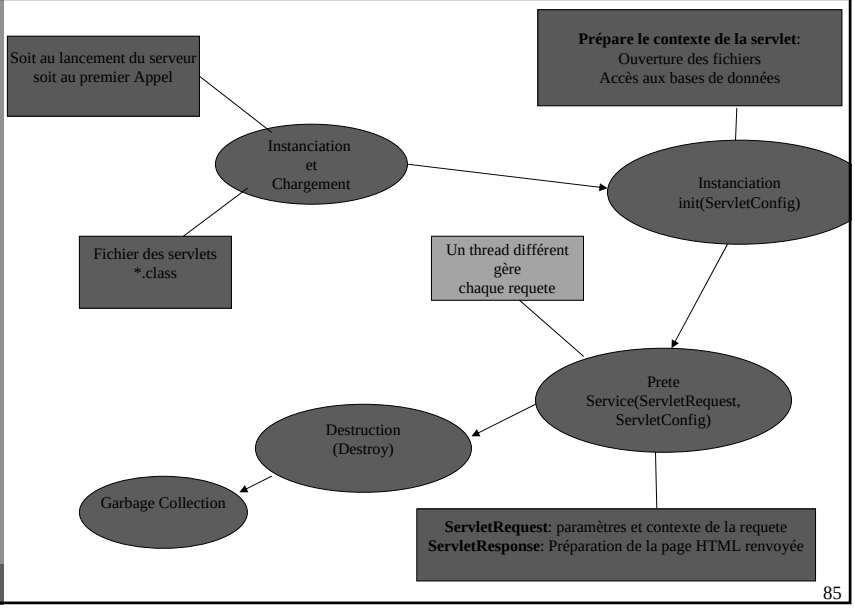
Servlet et génération de pages dynamiques
Exemple de génération de page

```
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> »);
}}
```

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Cycle de vie d'une servlet



BTD/GL/Archi

85

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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**

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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)**

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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**

CENTRALE LYON

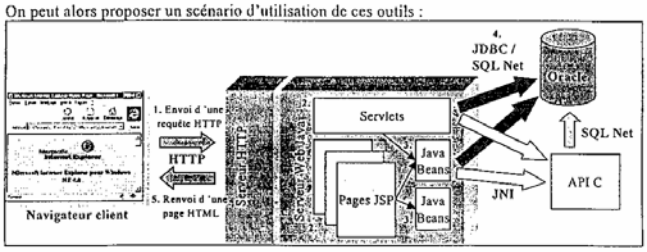
Bertrand DAVID : Génie Logiciel

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 :



BTD/GL/Archi

89

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Exemple de page JSP: les éléments

<%@page info =« exemple.jsp compilé »%>	DIRECTIVE JSP
<HTML ><HEAD><TITLE>La date par jsp</TITLE> Démonstration de génération de page dynamique	CODE HTML
<%@page import =« Date Bean »%>	DIRECTIVE JSP
<jsp:use Bean id=« ladate » class « Date Bean » scope «session »/	Balise spécifique JSP
Recherche du message. Recherche de la date. / <%=la date.getMessage()%> <%=ladate.getDate()%> / . 	EXPRESSIONS CODE HTML
<%! String[ ]semaine={« lundi », «mardi », «mercredi », «jeudi », «vendredi », «samedi »}; %>	DECLARATION

<%
int jour=ladate.getJour();
out.println(semaine[jour]);
if (jour!=1) out.println(« jour travaillé »);
else out.println(« jour férié »);
%>
	SCRIPTLET

BTD/GL/Archi

90

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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>
```

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

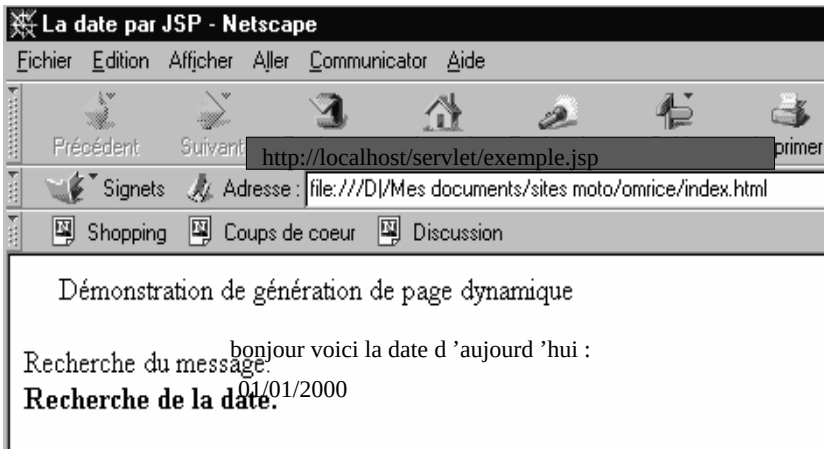
Programme Java de l’exemple de JSP

```
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);
    }
}
```

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Page JSP générée



Lundi.jour travaillé

BTD/GL/Archi

93

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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**

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Enterprise JavaBeans

Les composants JavaBeans

- **JavaBeans est un modèle de *composants logiciels réutilisables* spécifié par SUN MICROSYSTEMS en collaboration avec des industriels. Il permet au concepteur d'applications de bénéficier des avantages de Java et notamment de l'indépendance vis à vis d'une plateforme. L'idée est de programmer une seule fois de petits composants logiciels, et de pouvoir les utiliser et les réutiliser n'importe où.**

- *Différence entre les Enterprise JavaBeans et les beans*
- **Les EJB sont des *composants logiciels réutilisables*, c'est-à-dire des bouts de logiciels indépendants pouvant être assemblés pour construire une application à base d'objets métiers distribués dans un environnement Java. L'EJB typique est constitué de méthodes qui encapsulent la logique liée à un métier. A la différence des JavaBeans, ils ne possèdent pas d'interface graphique et sont destinés à être placés sur un serveur.**

- *Conteneur*
- **Un client n'accède pas directement à un EJB, en fait, les spécifications *EJB* décrivent le conteneur : fournisseur de services aux Enterprise JavaBeans. Ces services sont de l'ordre de la transaction, de la persistance et de l'administration des multiples instances d'un bean. Tant qu'un bean respecte les spécifications, il est capable de tourner dans n'importe quel conteneur sur n'importe quelle plateforme.**

- **Un conteneur simplifié a été présenté dans la partie *Java RMI*. Il gère l'interface des objets métiers distants. C'est à lui que s'adresse les clients pour invoquer les méthodes de ces objets. Le comportement des conteneurs EJB est similaire mais il faut ajouter de nombreux services qui simplifient grandement la programmation d'applications 3-tiers. Ainsi, les programmeurs n'ont plus à se soucier des détails concernant la gestion des transactions et des états, la concurrence, la sécurité, les unités d'exécution multiples, le regroupement des ressources et autres API complexes de bas niveau.**

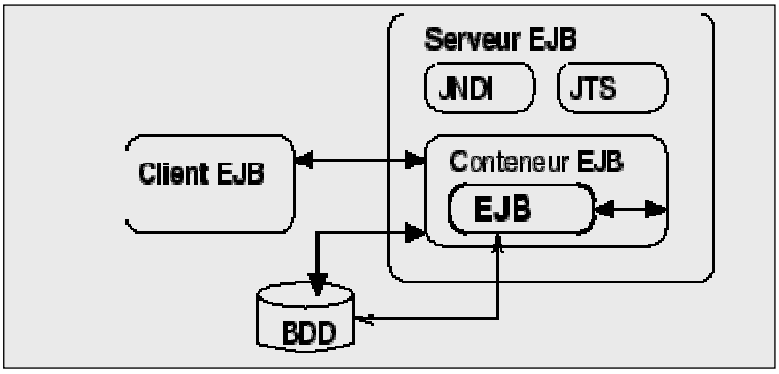
BTD/GL/Archi

95

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Architecture des Enterprise JavaBeans



Les serveurs EJB fournissent tout l'environnement énoncé par les spécifications EJB.
Les conteneurs EJB sont des interfaces entre les EJB et le monde extérieur. Ils font ``vivre les EJB'', invoquent leurs méthodes, gèrent les aspects techniques tels que la sécurité, les transactions, la concurrence et tout autre service. Un conteneur est responsable de la mise à disposition des classes EJB aux clients.
Les clients EJB sont des programmes qui permettent d'accéder aux EJB. Ils peuvent être lancés à partir d'un servlet, d'une applet, d'une page HTML, d'une page Java Server (JSP), de JavaScript, de DHTML, d'une application JAVA.
Les EJB sont des composants réutilisables orientés métier. Ils contiennent des méthodes sur des données de l'entreprise.
Une instance d'EJB est gérée par un conteneur. Tout accès à l'EJB passe par un conteneur.
JNDI *(Java Naming Directory Interface)* est une interface qui permet de retrouver les noms des beans déployés à partir d'un nom notoire.
JTS *(Java TransactionService)* permet au conteneur de gérer l'ensemble des transactions de la façon spécifiée dans le descripteur de déploiement de chacun des beans déployés. Le JTS est une encapsulation de *CORBA OTS*.

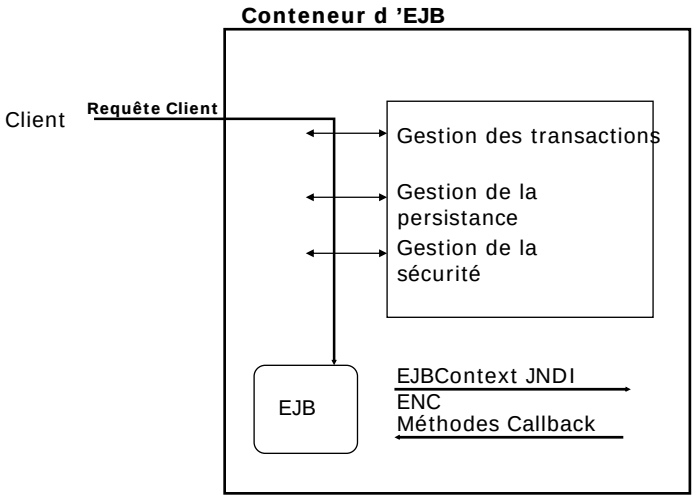
BTD/GL/Archi

96

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Le conteneur d'EJB



BTD/GL/Archi

97

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

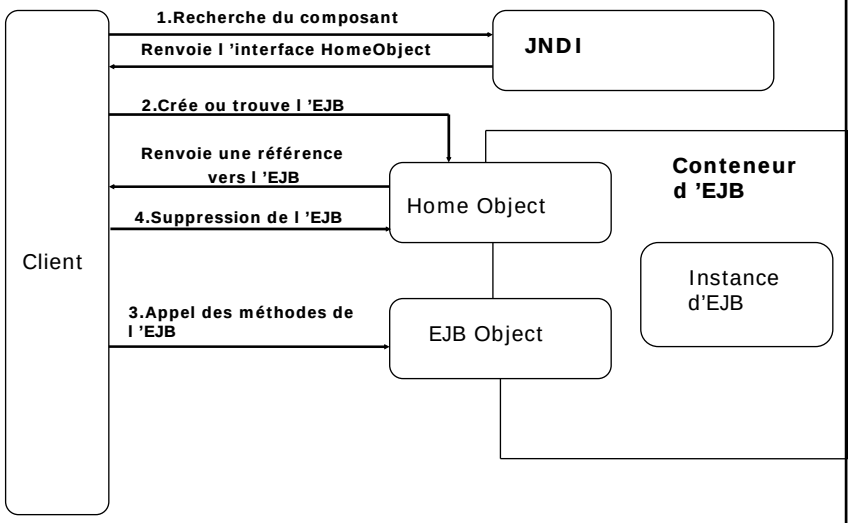
EJB: Les mécanismes du conteneur

- **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...

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Le conteneur d'EJB vu par le client



BTD/GL/Archi

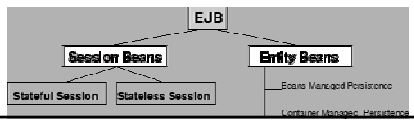
99

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Hiérarchie des différents EJB

- *Les beans session*
- **Un bean session a une durée de vie assez courte, limitée au temps d'une session client. Il est créé en réponse à la requête d'un seul client. Il peut être transactionnel. Il est détruit en cas de panne du serveur et le client doit établir un nouveau bean pour continuer les opérations. Il ne représente pas les données devant être stockées dans une base de données. Il offre un environnement d'exécution ajustable pour exécuter simultanément un grand nombre de Beans Session.**
- **Généralement les beans session sont utilisés pour des données transitoires pouvant être perdues.**
- **<u>Beans session sans état</u>**
  Un Bean session sans état ne garde pas trace des informations échangées d'un appel de méthode à un autre. C'est un bean sans variable d'instance et deux de ses instances sont équivalentes. Dans l'exemple de la bibliothèque, l'objet *Emprunt* peut être codé par un Bean Session sans état. En effet, il ne possède pas de variable d'instance et n'est créé que pour une requête d'un client.
- **Un conteneur peut facilement gérer ce type de bean car il peut être créé ou détruit à tout moment sans se soucier de son état.**
- **<u>Beans session avec état</u>**
  Un Bean session avec état possède une variable d'instance et donc change d'état au cours d'une transaction. L'exemple classique est le caddy dans un site de vente sur Internet où le client enregistre ses achats et à tout moment peut décider d'en payer le contenu.
- **Le cycle de vie de ce type de bean session est plus compliqué à gérer pour le conteneur, car s'il veut le suspendre temporairement, il doit stocker son état afin de le restaurer.**



BTD/GL/Archi

100

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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

BTD/GL/Archi

101

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Chaque Bean Session doit fournir trois classes :

- **La classe EJB :**
  - → **une implantation de l'interface *SessionBean*;**
  - → **les implantations des méthodes métiers ;**
  - → **les implantations des méthodes *ejbCreate*().**
- **L'interface home :**
  - → **l'héritage de l'interface *EJBHome*;**
  - → **les signatures des méthodes *create*().**
- **L'interface remote :**
  - → **l'héritage de l'interface *EJBObjet*;**
  - → **les signatures des méthodes métier.**
- Les beans entity
- **Les beans entity ont un cycle de vie long. Ils existent au travers de plusieurs sessions clientes et sont partagés par plusieurs clients. Ils survivent aux pannes diverses du serveur. Ils représentent un ensemble de données dans un système de stockage persistent comme une base de données. Dans l'exemple de la bibliothèque, l'objet *Bibliotheque* est typiquement un bean entité. Il contient deux tables dans lesquelles sont stockées les adhérents et les livres et un système d'entrée-sortie permet de sauvegarder ses données dans un fichier lors de l'extinction du serveur.**

BTD/GL/Archi

102

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Chaque Bean Entité doit fournir trois classes

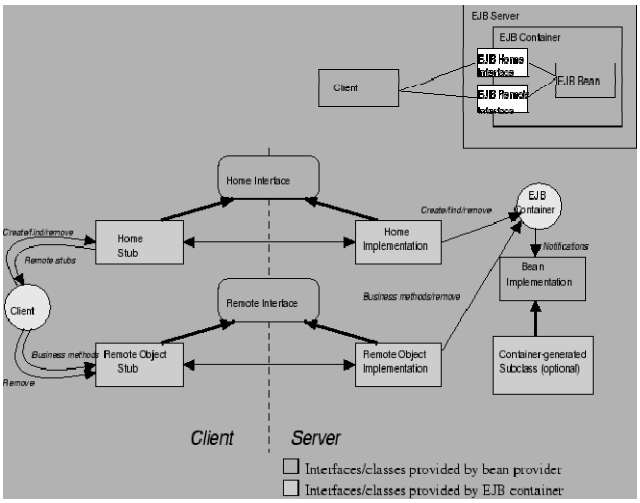
- **La classe EJB :**
  - **une implantation de l'interface *EntityBean*;**
  - **les implantations des méthodes métiers ;**
  - **les implantations des méthodes *ejbCreate*();**
  - **la méthode *ejbPostCreate*() correspondant à chaque méthode *ejbCreate*() implantée ;**
  - **une implantation de *ejbFindByPrimaryKey*() pour la persistance gérée par bean ;**
  - **les autres méthodes de recherches implantées.**
- **L'interface home :**
  - **l'héritage de l'interface *EJBHome*;**
  - **les signatures des méthodes *create*();**
  - **la signature de la méthode *findByPrimaryKey*();**
  - **les signatures des méthodes de recherche.**
- **L'interface remote :**
  - **l'héritage de l'interface EJBObject;**
  - **les signatures des méthodes métier.**
- **Les informations stockées dans les EJB sont liées aux informations de la base de données sous-jacente. Hormis le fait que les données sont conservées entre plusieurs sessions, les spécifications EJB prévoient deux types de synchronisation entre l'EJB et la base de donnée qu'il représente :**
- **La *persistance* gérée par bean spécifie que le bean doit lui-même contenir les primitives de sauvegarde et restauration vers une base de données persistante. Les méthodes *ejbLoad*() et *ejbStore*() sont implantées dans le bean. Le conteneur appelle l'une pour restaurer l'état de l'EJB depuis une base de données, et l'autre pour le sauvegarder.**
- **Ainsi, le programmeur possède la main sur la gestion de la persistance, mais l'utilisation de la persistance gérée par bean ne permet pas un portage sur d'autres environnements que celui pour lequel il a été créé.**
- **La persistance gérée par le conteneur évite d'avoir à implanter les méthodes *ejbLoad*() et *ejbStore*(). La persistance gérée par conteneur rend le code du bean portable sur n'importe quel environnement puisque la génération s'effectue en fonction de l'environnement opérationnel sur lequel le bean est déployé.**
- **En RMI, dans la partie serveur de données, la gestion de la persistance serait déléguée au conteneur.**

BTD/GL/Archi

103

CENTRALE LYON

Bertrand DAVID : Génie Logiciel



BTD/GL/Archi

104

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Le déploiement sur un serveur d'applications

- *Principales caractéristiques d'un serveur d'application*
  - **La plupart des serveurs d'application possède les caractéristiques suivantes :**
  - **administration de composants ;**
  - **bus logiciel ;**
  - **administration de données ;**
  - **services de sécurité ;**
  - **services d'administration ;**
  - **service de transaction.**
  - **A cela, il faut ajouter un moniteur de transaction, ou un gestionnaire de file d'attente, un outil de développement d'application à base de composants, un serveur Internet qui gère les documents HTML et les scripts CGI.**
- *Les serveurs d'applications EJB*
  - **Les serveurs d'applications EJB supportent l'accès de composants Java, d'applets et de clients EJB par l'intermédiaire de serveurs internet ou directement.**
  - **Les serveurs d'application EJB implantent des modèles de composants. Ils possèdent en leur noyau un certain nombre de frameworks et protocoles destinés à accueillir les objets métiers et à en faciliter l'accès et la gestion.**
- *Terminologie*
  - **Le composant EJB est développé sans se soucier de l'environnement dans lequel il sera instancié.**
  - **Le conteneur EJB accueille le composant EJB. Les personnes chargées du déploiement de l'EJB doivent garantir l'accueil du composant par le container et l'accueil du conteneur par le serveur d'application EJB. coordonne une transaction, mais le contrôle de la transaction est délégué à l'administration des transactions qui peut faire partie du serveur d'application.**
  - **Le serveur d'application propose souvent un mécanisme de files d'attente de messages et de répartition de charge.**

BTD/GL/Archi

105

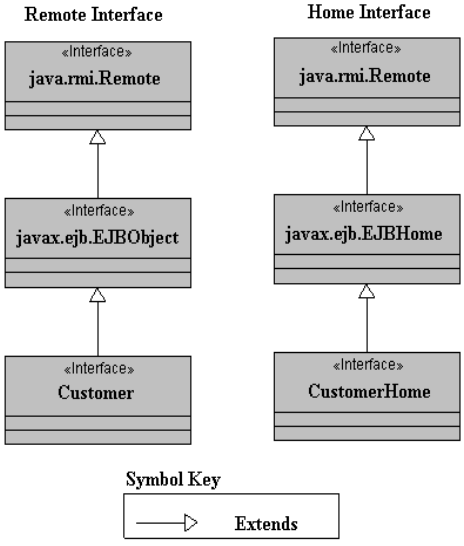
CENTRALE LYON

Bertrand DAVID : Génie Logiciel

EJB: Les 2 interfaces

Interface distante:

Concerne les méthodes de l'instance de l'EJB



Interface locale:

(cycle de vie de l'EJB)

créer

détruire

trouver

Obtention d'une référence sur l'interface distante

Class Diagram of Remote and Home Interfaces

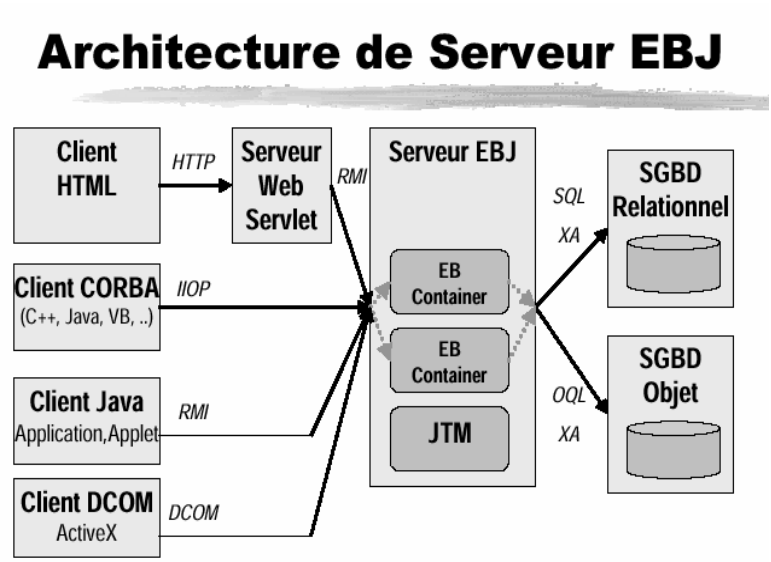
BTD/GL/Archi

106

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Les modes d'accès aux EJB



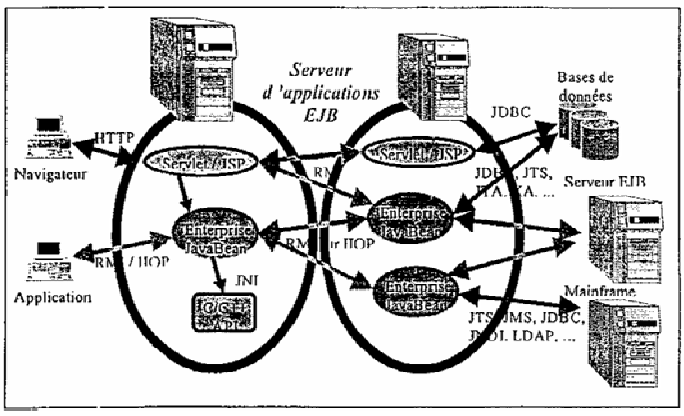
BTD/GL/Archi

107

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

EJB – Entreprise Java Beans



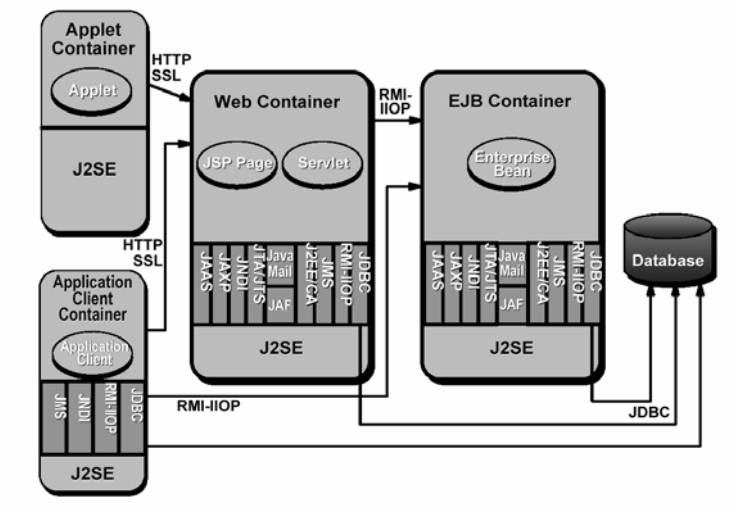
BTD/GL/Archi

108

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

J2EE Platform



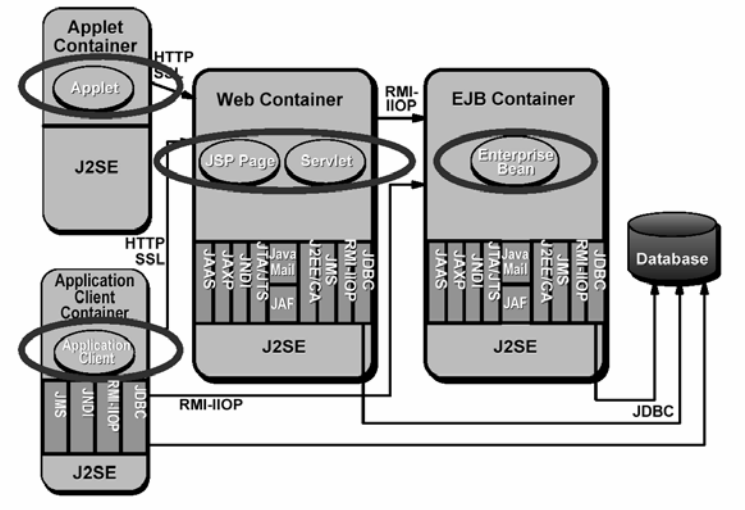
BTD/GL/Archi

109

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

J2EE Application Components



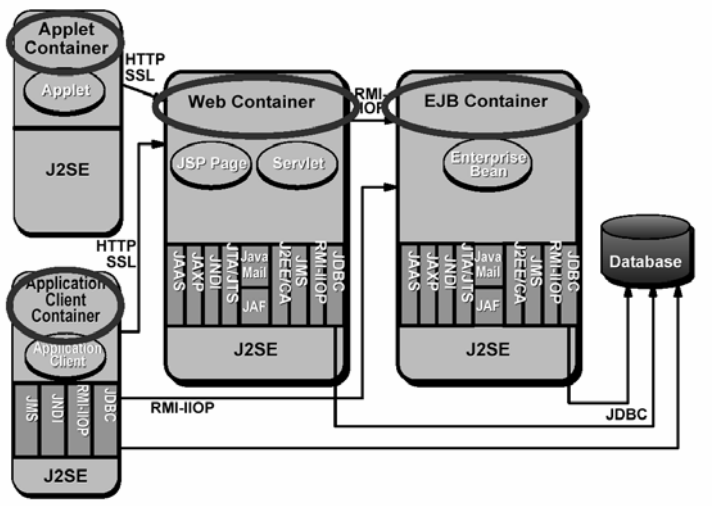
BTD/GL/Archi

110

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

J2EE Containers



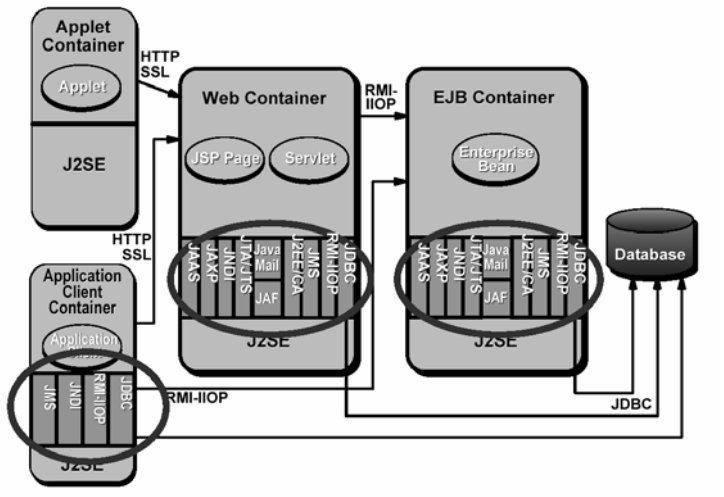
BTD/GL/Archi

111

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

J2EE Standard Services



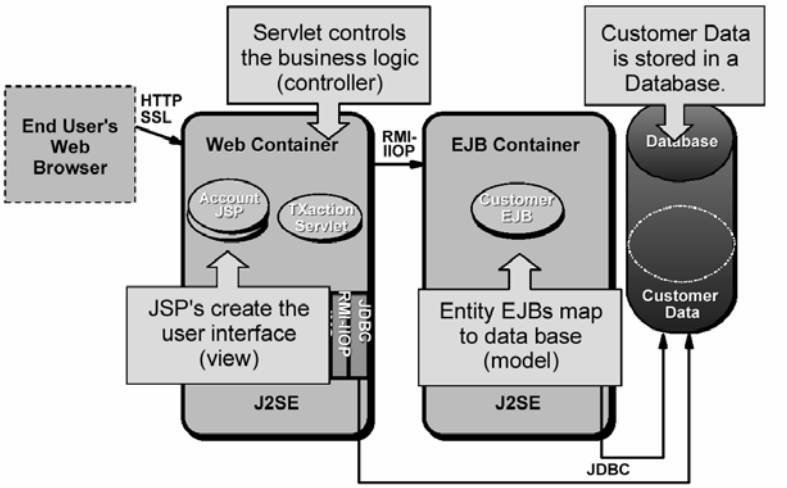
BTD/GL/Archi

112

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Server based J2EE application



BTD/GL/Archi

113

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.**

BTD/GL/Archi

114

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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**

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.**
-

BTD/GL/Archi

116

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.

BTD/GL/Archi

117

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.

BTD/GL/Archi

118

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.**

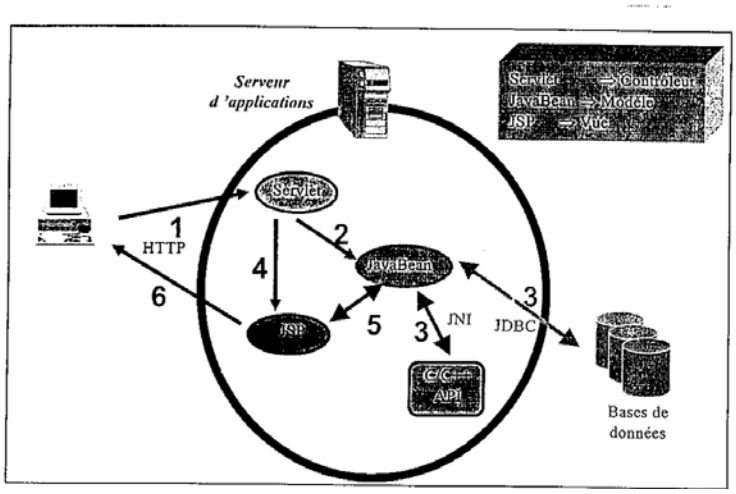
BTD/GL/Archi

119

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Modèle - MVC



BTD/GL/Archi

120

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.**

BTD/GL/Archi

121

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.**

BTD/GL/Archi

122

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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.**

BTD/GL/Archi

123

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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 !!!**

BTD/GL/Archi

124

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

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**

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Lexique

- **EAI Entreprise Application Intégration**
- **Middleware**
- **ODBC Open DataBase Connectivity**
- **IDL Interface Definition Language**
- **ISAPI API permettant d'ajouter des fonctions au serveur Internet de Microsoft (Internet Information Server IIS)**
- **NSAPI idem pour Netscape (Netscape Entreprise Server)**
- **DTP Distributed Transaction Processing**
- **ACID**
- **OMG Object Management Group**

BTD/GL/Archi

126

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

Définitions

- COM Component Object Model
- DCOM Distributed Component Object
- ActiveX composants s'appuyant sur COM
- OLE Object Linking and Embedding
- HTTP
- RMI Remote Method Invocation
- JAR fichier archive de Java
- CAB

BTD/GL/Archi

127

CENTRALE LYON

Bertrand DAVID : Génie Logiciel

OLE - COM - DCOM ActiveX

- En construction

BTD/GL/Archi

128