Corba :

Nommez et décrivez les 5 types de composants d'une architecture Corba

Object common services :Un ensemble de spécification d'interfaces IDL-Leurs fonctionnalités peuvent être étendues ou spécialisées par héritage-Interfaces indépendantes des domaines d'application-Centralise des services répétitifs que l'on retrouve dans la plupart des développements

common facilities : Ensemble de services de plus haut niveau fournissant des fonctionnalités utiles dans de nombreuses applications.-ndépendants du domaine d'application-Exemple : gestionnaire d'impression.

Domain objects :Objets utilitaires destinés à un secteur d'activité donné-Leur objectif : assurer l'interopérabilité sémantique entre les SI d'entreprises d'un même métier.

Application Objects :Ce sont des interfaces développées en IDL spécifiquement pour une application donnée-Ne sont pas standardisées par l'OMG

ORB :Transporte les requêtes vers les objets.-Facilite la communication entre les objets d'un système réparti.-Prend en charge la communication entre les objets serveurs et les différents clients.-Permet l'interopérabilité et la portabilité des applications.

Quels sont les 2 éléments dans Corba qui permettent l'interopérabilité ?

- ✓ L'ORB : Car il s'occupe du transport des requêtes/réponses entre clients et serveur malgré l'hétérogénéité du matériel et de l'applicatif met en œuvre
- ✓ L'Object Adapter : permet l'adaptation à la nature des objets implantés

Mentionner et comparer les différentes façons qu'a un c,lient CORBA pour obtenir la référence d'un servant CORBA ?

- ✓ Diffusion de l'IOR non gérée par le bus :
  - o Utilisation de object_ti_string coté serveur et string_to_object coté client
  - o Fichier partagé de références objets
  - o Problèmes :
    - ▪ Ne supporte pas le cas où le serveur support plusieurs servants
    - ▪ Ne peut pas supporter l'accès de plusieurs instances de serveurs au même fichier partagé de références
  - o Adapté à des cas simple
- ✓ Utilisation du service de nommage
  - o Permet la désignation d'objets CORBA à l'aide de noms
  - o Equivalent à la notion de path
  - o Peut être persistent
  - o Adapté à tout les cas de figure non supportés par la diffusion non gérée par le bus

Qu'est ce que le référentiel d'interfaces dans Corba ?

C'est un composant corba qui stocke toutes les définitions IDL des interfaces des objets du bus, des méthodes qu'ils supportent et des paramètres qu'ils nécessitent.

Décrire les étapes de développement d'une application corba ?

1. Ecriture de l'interface de l'objet en IDL
2. Compilation de l'IDL • Génération des modules stub et skeleton
3. Implantation de l'objet • Dérivation d'une classe depuis le skeleton généré

4. Rédaction ou génération du code de l'application serveur • Ce code sert à lancer l'objet et à le mettre à disposition des différentes applications clientes
5. Compilation du serveur • Génération de l'exécutable de l'application serveur avec liaison au module ORB
6. Réalisation de l'application cliente • Cette application se connecte à l'objet distant pour lui demander l'exécution de méthodes
7. Compilation du client • Tout comme le serveur, cette application doit inclure l'ORB

IDL :

Un compilateur C génère du code objet. Qu'en est-il d'un compilateur IDL ? Que fait-on avec les résultats de la compilation IDL ?
un compilateur idl permet de projeter les définitions écrite en idl vers le langage voulu. Les résultats de la compilation IDL sont utilisés pour lier aux objets CORBA (objets fictifs) des objets d'implantation (servant) qui vont évoluer au sein d'un serveur.

De quelle façon peut-on écrire deux interfaces IDL dans le même fichier ayant exactement le même nom sans pour autant provoquer une erreur de compilation?

Les placer dans des modules différents.

Supposons qu'une application comporte une partie client réalisée en Java et une partie serveur réalisée en C++. Indiquez combien de contrats IDL, de compilateurs IDL et d'ORB différents sont nécessaires au minimum pour réaliser cette application?

- ✓ CONTRATS IDL :1
- ✓ Compilateur IDL :2
- ✓ ORB : 1 (si sur la même machine) sinon 2

Le langage IDL ne contient aucune instruction d'affectation. Pourquoi?

Car IDL est langage déclaratif et non un langage de programmation.

SII DII SSI DSI:

expliquer la difference entre appel statique et dynamique. La difference est-elle du cote du client, du serveur, ou des deux ? quels sont les avantages et les inconvénients de l'appel dynamique ?

• **SII** (Static Invocation Interface) est l'interface d'invocations statiques permettant de soumettre des requêtes contrôlées à la compilation des programmes. Cette interface est générée à partir de définitions OMG-IDL.
• **DII** (Dynamic Invocation Interface) est l'interface d'invocations dynamiques permettant de construire dynamiquement des requêtes vers n'importe quel objet CORBA sans générer/utiliser une interface SII.

• **SSI** (Skeleton Static Interface) est l'interface de squelettes statiques qui permet à l'implantation des objets de recevoir les requêtes leur étant destinées. Cette interface est générée comme l'interface SII.
• **DSI** (Dynamic Skeleton Interface) est l'interface de squelettes dynamiques qui permet d'intercepter dynamiquement toute requête sans générer une interface SSI. C'est le pendant de DII pour un serveur.

IOR : ORB : POA :

Quelle information est contenue dans un IOR ?

C'est l'information nécessaire pour l'aiguillage de la requête du client au serveur.

Quelle est l'utilité de l'opération:

interface ORB { Object resolve initial references (in Objectld identifier) raises (InvalidName); }

Elle retourne la référence d'objet corba associéé au service identifié.

quelle est l'utilité d'utiliser plus d'un portable object adapter (POA à l'interieur d'une meme application serveur ?:

Séparer les POA suivant les métiers et chaque POA gère un ensemble de servants.

Objet:

Quelles sont les différentes façons de mise en oeuvre d'un objet d'implantation. Expliquez les motivations de chacune des approches.

Il existe deux façons de mise en œuvre d'un objet d'implantation :

Par héritage : Motivation ➔plus performante (vu qu'elle ne nécessite pas la création d'une instance supplémentaire- l'appel de méthode est direct)

Par délégation :Motivation➔souplesse(Permet de choisir la superclasse de l'objet servant)

Expliquez la différence entre objet distribué, servant, object id et IOR ?

IOR :suite d'octets identifiant l'objet chez le serveur-Object id : suite d'octets identifiant l'objet chez le POA-Servant : entité réelle programmée en un langage, implante un objet distribué et évolue dans le cadre d'un serveur.-Objet distribué : entité virtuelle qui peut être localisée par un bus ORB et qui reçoit les requêtes clients qui lui sont destinés

Cobol :

Comment peut-on rendre une application COBOL accessible depuis Internet?

En exportant ces services sous la forme d'un objet CORBA, ainsi les différents clients peuvent utiliser ses services à travers l'ORB et en particulier l'IIOP.

Serveur :

Que doit faire le code serveur ?

- ✓ Initialiser son environnement d'exécution :Initialiser l'ORB et l'adaptateur d'objets
- ✓ Rendre les objets servants publics :Exporter leurs IOR
- ✓ Attendre les requêtes des clients
- ✓ Arrêter l'environnement d'exécution

Quel est la différence entre un servant et une application serveur :

Un serveur est une structure d'accueil des objets d'implantation(servant) et des exécutions des opérations

Au niveau Code :

Indiquer comment obtenir une référence sur un objet *bibliothèque* distant.

Par la mise à la disposition des clients un fichier partagé sur lequel sera sérialiser le IOR de l'objet bibliothèque.

Indiquer comment faire un appel à la méthode TrouverAdhérent. ?

1. Le client récupère la IOR de l'objet bibliothèque sur lequel on veut chercher l'adhérent.
2. Utiliser la méthode string_to_object afin de se lier à l'objet distant
3. Appeler la méthode par l'objet récupéré

Indiquer comment faire un appel à l'attribut *nom.*

1. Le client récupère la IOR de l'objet adhérent sur lequel on veut accéder au nom.
2. Utiliser la méthode string_to_object afin de se lier à l'objet distant
3. Appeler la méthode nom()

Code :

---adherentImpl ad = new adherentImpl(nom,prenom,adresse,numcarte); ListeAdherents.add(ad);
--- adherentImpl ad=null; ListIterator it=this.ListeAdherents.listIterator(); while(it.hasNext()){ ad=(adherentImpl)it.next(); if(ad.numcarte()==numcarte) it.remove(); }
--- adherentImpl ad=null; ListIterator it=this.ListeAdherents.listIterator(); while(it.hasNext()){ ad=(adherentImpl)it.next(); if(ad.numcarte()==numcarte) break; }

--- interface HotelItf{exception ReservationImpossible{string raison;};exception NumeroIncorrect{}; unsigned short nbChambreLibres(in Date dateCheck); unsigned short reserver(in Date dateRes,in unsigned short nombreChambre,in unsigned short nombreDeNuit); raises(ReservationImpossible);

void annuler(in unsigned short numReservation); raises(NumeroIncorrect); };

--- struct Date { unsigned short Day; unsigned short Month; short Year;};

---package agence; import agence.HotelItfPOA; public class HotelImpl extends HotelItfPOA{ short nbChambreLibres(Date dateCheck){} short reserver(Date dateRes,short nombreChambre,short nombreDeNuit) throws agence.HotelItfPackage.ReservationImpossible{} void annuler(short numReservation) throws agence.HotelItfPackage.NumeroIncorrect{}}