



Intergiciels

CORBA



Introduction

CORBA : *Common Object Request Broker Architecture*

- Un modèle d'intergiciel orienté objets répartis
- Spécifications standards pour un environnement d'applications réparties
- Support multi-systèmes, multi-langages et non propriétaires
- Infrastructure de communication pour des objets hétérogènes
- Services généraux pour des applications réparties



Object Management Group

L'OMG (*Object Management Group*) est un consortium international

- Objectif :
 - Promouvoir la technologie objets répartis
 - spécifications standards pour l'interopérabilité des applications des entreprises;
- fondé en 1989 par 11 entreprises
- regroupe environ 700 membres (IBM, Sun Microsystems, HP, Motorola, Nokia, Siemens, universités, ...)

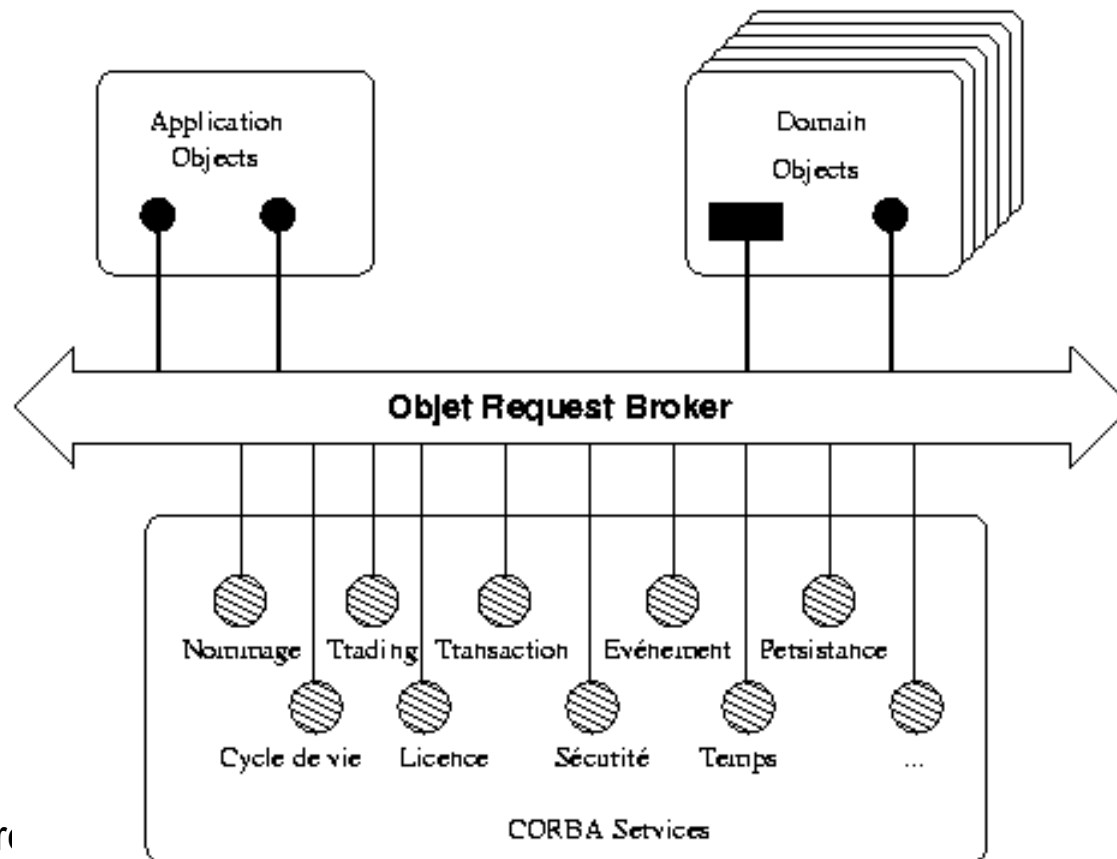


Principales productions de l'OMG

- OMA (Object Management Architecture) 1990
- CORBA (Common Object Request Broker Architecture) 1991
- Services CORBA (Common Object Services) 1997
- IIOP (Internet Inter-ORB Protocol) 1996
- UML (Unified Modeling Language) 1997
- CCM (Corba Component Model) 2002
- MDA (Model Driven Architecture) en cours

Object Management Architecture (1)

Le modèle CORBA s'inscrit dans un modèle plus global de gestion des objets distribués : OMA (*Object Management Architecture*)





Object Management Architecture (2)

Cette architecture spécifie un environnement standard pour le déploiement d'objets distribués regroupant :

- Un bus logiciel (ORB : *Object Resource Broker*) qui assure la connexion entre les objets;
- Les services CORBA gèrent la création et l'accès aux objets déployés;
- Les *Domain Objects* sont des composants qui encapsulent certaines fonctionnalités propres à un domaine d'application (ex : gestion de production, finance, ...);
- Les objets d'applications sont développés pour une application particulière et tirent parti de l'ensemble des autres composants.



Services CORBA (1)

- **Nommage** : Recherche d'un objet par son nom (pages blanches)
- **Trading** : Recherche d'un objet par ses propriétés (pages jaunes)
- **Transaction** : Permet des échanges transactionnels entre objets (propriétés ACID)
- **Evénement** : Gestion d'événements en mode producteur / consommateur
- **Notification** : Filtrage des événements
- **Persistance** : Stocke et restaure l'état des objets
- **Cycle de vie** : Créé, copie, déplace, détruit des objets
- **Licence** : Contrôle l'utilisation des objets



Services CORBA (2)

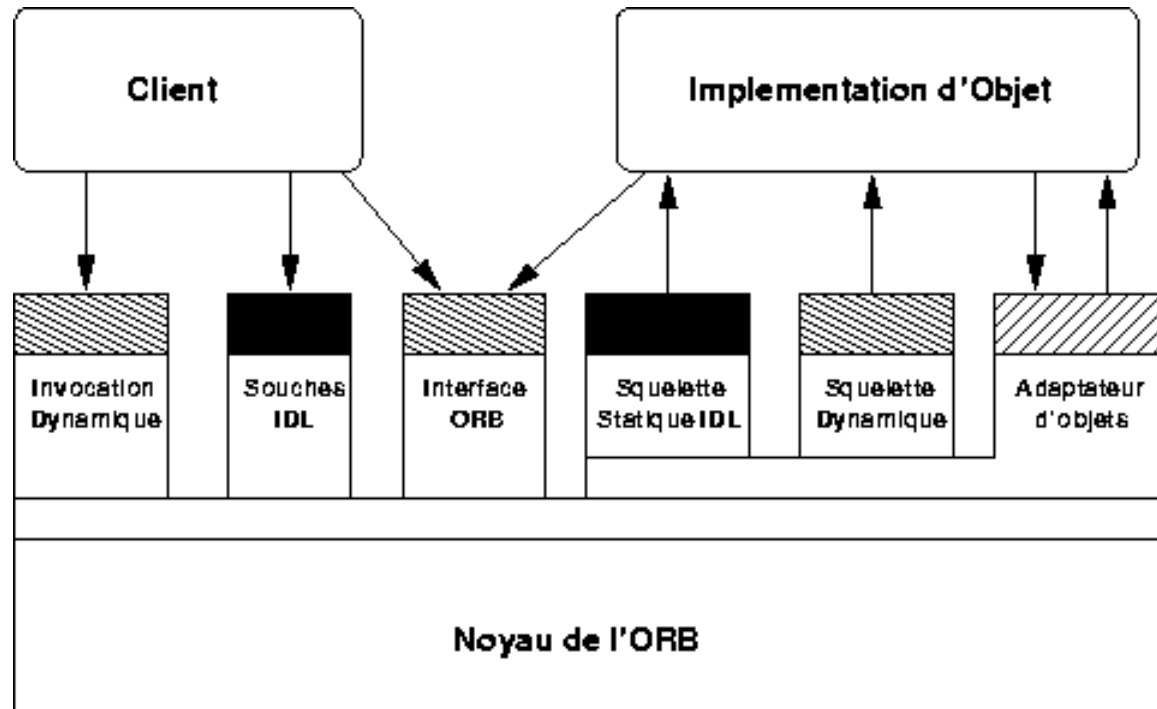
- **Sécurité** : Authentification, contrôle d'accès, chiffrement, ...
- **Temps** : Synchronisations d'horloge
- **Propriétés** : Affecte des attributs à des objets
- **Interrogation** : Envoi de requêtes vers les objets
- **Concurrence** : Gestion de verrous pour les accès concurrents
- **Relations** : Gestion d'association entre objets
- **Collection** : Gestion de collection d'objets



Spécifications CORBA

- Un langage de définition des interfaces (IDL CORBA)
- Projection de l'IDL vers d'autres langages
 - Ada, C, C++, Cobol, CORBA Scripting Language, Java, Lisp, Python, SmallTalk
- Fonctionnalités de l'ORB
- Interopérabilité entre ORBs
 - IIOP (Internet Inter-ORB Protocol)
 - GIOP (General Inter-ORB Protocol)

Vue Générale de CORBA



Interface indépendante de l'implémentation de l'ORB



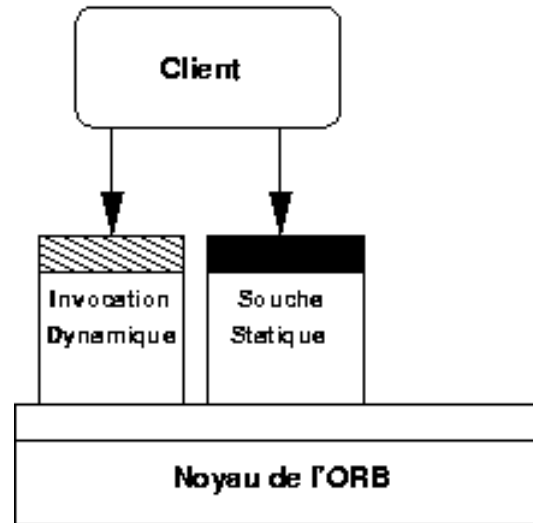
Des souches et squelettes existent pour chaque type d'objet



Il peut y avoir plusieurs adaptateurs d'objets



Objet Client



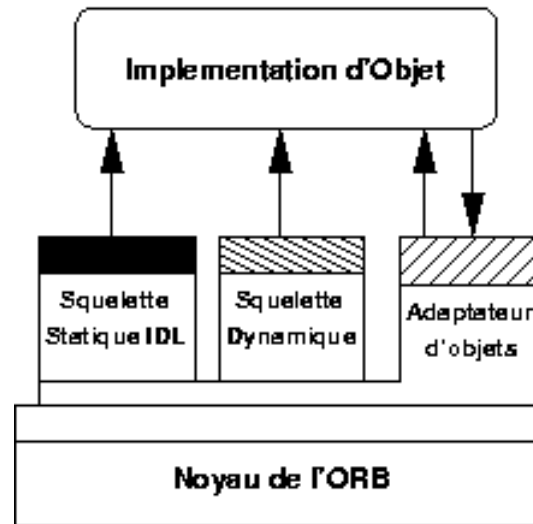
Souche statique

Une par type d'objet serveur
Générée à partir d'une interface IDL
Similaire aux souches RMI

Souche dynamique

Souche générique construisant dynamiquement tout type de requête
Permet d'invoquer des objets serveurs découverts à l'exécution

Objet fournisseur de service (serveur)



Squelette

Invoque un service pour un client
 Renvoie une valeur de retour à une souche
Statique: associé à un type d'objet serveur, généré à partir d'un IDL
Dynamique: squelette générique, accès à des objets serveurs inconnus

Adaptateur d'objet

Réceptacle pour les objets serveurs (servants)
 Créé les références d'objets
 Plusieurs adaptateurs peuvent exister dans des espaces différents



Adaptateur d'objet

Un adaptateur d'objet est le mécanisme qui connecte des requêtes à un objet implémenté dans un langage de programmation en utilisant une référence (IOR).

Portable Object Adapter (POA)

- Assure la portabilité entre différents ORB
- Supporte la persistance d'identité
- Permet à un servant d'avoir plusieurs identités



Interfaces IDL (1)

Les services fournis par un objet CORBA sont décrits dans une interface au format IDL (*Interface Definition Language*).

- une interface regroupe un ensemble d'attributs et d'opérations;
- des relations d'héritage entre interfaces peuvent exister;
- correspondance entre IDL et un langage de programmation.



Interfaces IDL (2)

- Structure
- Syntaxe
- Exceptions
- Modules
- Interfaces



IDL – Structure (1)

Une interface est décrite dans un fichier IDL

```
<types>
<constantes>
<exceptions>

<modules>
  <interfaces>
    <attributes>
    <operations>
```

Les types, constantes et exceptions peuvent également être déclarés localement à un module ou à une interface.



IDL – Structure (2)

```
exception invalidName {} ;

module MasterWI {
    interface ChatServeur {
        attribute string user ;
        attribute string message ;
        string addChat(in string localisation) ;
        void publishString(in string n, in string s)
            raises (invalidName) ;
    } ;
} ;
```



IDL – Syntaxe (1)

- Syntaxe proche de C++ ou Java
- La casse des identificateurs n'est pas discriminante
- Mots-clés de l'IDL :

any	double	interface	raises	TRUE
attribute	enum	long	readonly	typedef
boolean	exception	module	sequence	union
case	FALSE	Object	short	unsigned
char	float	octet	string	void
const	in	oneway	struct	wchar
context	inout	out	switch	wstring
default				



IDL – Syntaxe (2)

Type de données simples:

- `short` : codé sur 2 octets
- `long` : codé sur 4 octets
- `long long` : codé sur 8 octets
- `float` : codé sur 4 octets
- `double` : codé sur 8 octets
- `long double` : codé sur 16 octets
- `char` : codé sur 1 octet
- `wchar` : codé sur 2 octets



IDL – Syntaxe (3)

exemples de types de données complexes :

```
struct personne {  
    string <24> nom;  
    short age  
    string adresse;  
};  
enum sexe {masculin, feminin};  
union pers switch(sexe) {  
    case feminin: string nom;  
                string nomJeuneFille;  
    default: string nom;  
};  
sequence <long,20> vecteur;
```



IDL – Syntaxe (4)

Les constantes sont déclarées par le mot-clé `const`

```
const <type> <ident> = <expr>;
```

L'expression `<expr>` est soit une valeur constante soit un calcul n'utilisant que des valeurs constantes.

```
const short joursParSemaine = 7;
```

```
const short heuresParJour = 24;
```

```
const short heuresParSemaine = joursParSemaine  
* heuresParJour;
```



IDL - Exception

Dans un fichier IDL les exceptions :

- sont similaires aux exceptions Java;
- décrivent un comportement anormal d'un traitement;
- sont définies dans le fichier de l'interface;
- ou appartiennent à des exceptions systèmes;

```
exception <ident> { <donnee>* };
```

```
exception invalidName {  
    string value;  
};
```

```
void publishString(in string n, in string s) raises  
    (invalidName);
```



IDL - Modules

Un module regroupe un ensemble de types, constantes, exceptions et interfaces de même « thème » (similaire à la notion de package en Java).

```
module <ident> { ... }
```

```
module MasterWI {  
    module TPCorba {  
        interface Chat {  
            ...  
        };  
    };  
};
```

Toutes les définitions d'un module sont accessibles en précisant leur chemin :

```
MasterWI::TPCorba::Chat
```



IDL – Interfaces (1)

Attributes

- Accès public
`attribute string message;`
- Peut être en lecture seule
`readonly attribute string user`
- Une opération de lecture (*getter*) y est associé
`string user();`
- Une opération d'écriture (*setter*) peut y être associé
`void message(string);`



IDL – Interfaces (2)

Opérations

- définit le nom d'une opération, les types des paramètres d'entrée et de retour et les exceptions susceptibles d'être levées

```
<type retour> <ident>(<liste parametres>)  
raises (<exceptions>);
```

- différents types de passage de paramètres

```
void tri(in long taille, inout sequence  
<long> tab, out boolean doublon);
```

- opérations asynchrones (pas de paramètres de sortie)

```
oneway void publishString(in string s);
```

Mode d'invocation des opérations

- Synchrones
 - valeurs de retour possibles
 - bloquant pour le client
- Asynchrone (*oneway*)
 - pas valeurs de retour possibles
 - non bloquant pour le client
- Semi-synchrone



IDL – Interfaces (3)

Héritage

- Syntaxe

```
interface <ident>:<ident_interface_mere>
    {...};
```

- Héritage multiple autorisé (attention aux losanges)

```
interface Chat : User, Communication {...}
```

- Surcharge interdite, sauf pour les constantes et exceptions



De l'IDL à un langage de programmation

Des utilitaires permettent de générer automatiquement du code objet à partir d'une interface écrite en IDL.

Des outils dédiés à un langage de programmation réalisent les traductions :

- mapping des types de données IDL vers ceux du langage;
- transformation de types complexes en objets;
- génération d'une interface, de souches et de squelettes.

Pour Java l'utilitaire de génération de code est `idlj`.



Mapping vers Java et C++

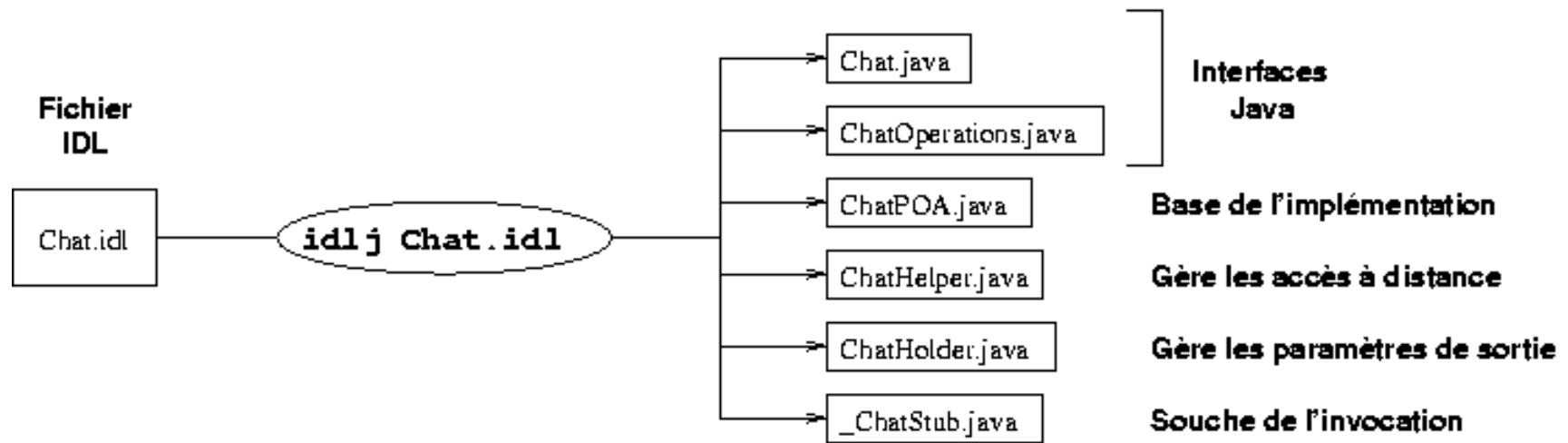
IDL	Java	C++
module	package	namespace
interface	interface	abstract class
opération	méthode	member function
attribut	paire de méthodes	paire de fonctions
exception	exception	exception



Mapping vers les types Java

Type IDL	Type Java
boolean	boolean
char/wchar	char
octet	byte
short	short
long	int
long long	long
float	float
double	double

Génération de code en Java



- Les classes nécessaires au client sont créés avec l'option `-fclient`
- Les classes seulement nécessaires au serveur sont créés avec l'option `-fserver` (mais l'utilisation de `-fall` est conseillée)

Développement d'une application

- Ecriture de l'interface IDL
- Génération du code côtés client et serveur
- Ecriture de l'implémentation du serveur
 - 1) Accès à l'ORB
 - 2) Création de l'objet implémentation
 - 3) Enregistrement de l'objet implémentation
- Ecriture de l'implémentation du client
 - 1) Accès à l'ORB
 - 2) Découverte des objets implémentations
- Exécution du serveur
- Exécution du client



Référence d'objets (IOR) (1)

L'ORB attribue un IOR (*Interoperable Object Reference*) aux objets connectés.

Accès à un objet à partir d'une référence :

```
org.omg.CORBA.ORB orb =  
    org.omg.CORBA.ORB.init(args,null);  
org.omg.CORBA.Object obj =  
    orb.string_to_object(args[0]);  
Chat CS = ChatHelper.narrow(obj);
```

Dans cet exemple, args[0] doit être l'IOR d'une implémentation de Chat.



Référence d'objets (IOR) (2)

Accès à une référence à partir de l'objet :

```
org.omg.CORBA.ORB orb = org.omg.CORBA.ORB.init(args,null);

POA rootpoa = POAHelper.narrow(
    orb.resolve_initial_references("RootPOA"));
rootpoa.the_POAManager().activate();

ChatImpl obj = new ChatImpl();

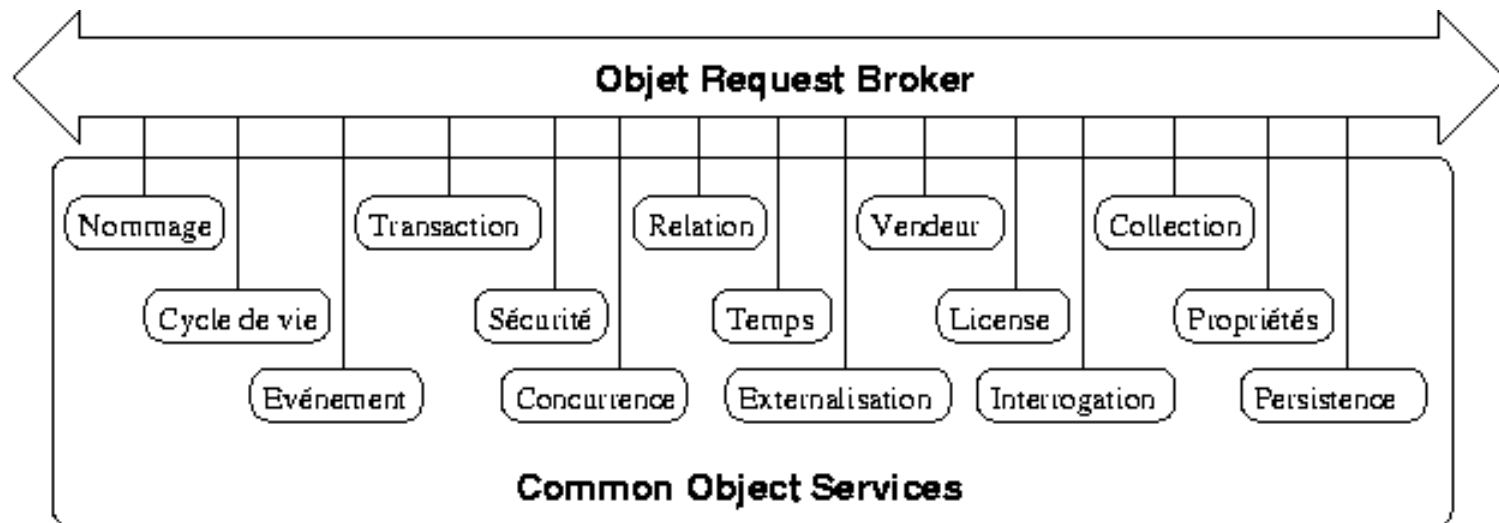
org.omg.CORBA.Object ref = rootpoa.servant_to_reference(obj);
String ior = orb.object_to_string(ref)

orb.run();
```



Services COS

Plusieurs services génériques sont accessibles via l'ORB



Lancement des services avec le SDK 1.6 :

orbd -ORBInitialPort <port> -ORBInitialHost <hote>



Service de nommage

- Service de pages blanches

- Accès au service

```
org.omg.CORBA.Object objRef =  
    orb.resolve_initial_references("NameService");  
NamingContextExt ncRef = NamingContextExtHelper.narrow(objRef);
```

- Enregistrement

```
ChatImpl obj = new ChatImpl();  
org.omg.CORBA.Object ref = rootpoa.servant_to_reference(obj);  
Chat chatRef = ChatHelper.narrow(ref);  
NameComponent path[] = ncRef.to_name(«chat»);  
ncRef.rebind(path, chatRef);
```

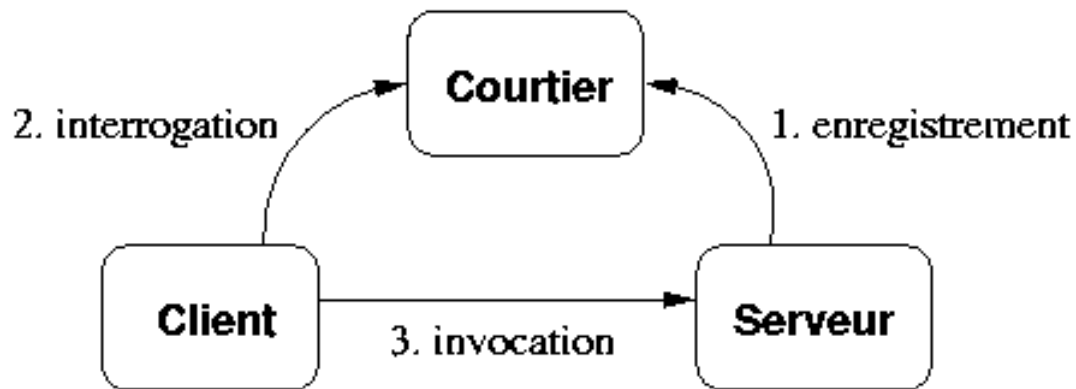
- Recherche

```
Chat ch = ChatHelper.narrow(ncRef.resolve_str(«chat»));
```

Service de courtage

Recherche d'un type d'objet plutôt qu'un objet spécifique

- Service de pages jaunes
- Les objets enregistrent leur fonction auprès du courtier





Liens utiles

- D'autres cours ou tutoriaux
 - <http://www.lifl.fr/~merle/corba/cours/>
 - <http://www.cs.wustl.edu/~schmidt/corba.html>
 - <http://java.sun.com/developer/codesamples/idl.html>
- CORBA IDL
 - <http://java.sun.com/j2se/1.5.0/docs/guide/idl/>
 - <http://developer.gnome.org/doc/guides/corba/html/idl-basics.html>
 - <http://java.sun.com/j2se/1.5.0/docs/guide/idl/jidlGlossary.html>
- Détails sur le service de nommage
 - <http://java.sun.com/j2se/1.5.0/docs/guide/idl/jidlNaming.html>