Travaux Dirigés CORBA n°2

Master 2 Ingénierie Informatique

—2007-2008—

Le service de nommage

L'objectif de ce td est d'apprendre à utiliser le service de nommage CORBA (`NamingService`). Ce service va gérer les références de vos objets applicatifs au sein d'une hiérarchie d'objets.

▸ Exercice 1. *Un service de nommage simple*

Jusqu'à présent, l'échange des références d'objet s'est fait par l'intermédiaire d'un fichier de nom `ObjectRef`*qui doit être commun au serveur et au client. Cette technique est, comme vous l'avez certainement constaté, peu adaptée à des applications distribuées sur Internet. Le service de nommage CORBA (*`NamingService`*) est fait pour palier à ce problème.*

Le SDK de Sun fournit par défaut un application serveur de nommage : `orbd`*. Son lancement se fait en lui précisant un numéro de port comme dans l'exemple suivant :*

```
orbd -ORBInitialPort 1234
```

Pour pouvoir accéder au service de nommage, il faut indiquer au serveur ou au client où le trouver. Pour cela, on passe en argument de l'ORB le numéro de port et le nom de l'hôte sur lesquels le serveur de nommage attend les requêtes. Ceci se fait, par exemple, de la manière suivante si la référence de l'orb est obtenue dans la classe `App` *en appelant* `orb.init(args,null)`[1] *:*

```
java App -ORBInitialPort 1234 -ORBInitialHost host
```

Dans ce premier exercice nous allons enregistrer simplement la référence d'un objet de type `Forum` *dans la racine du service de nommage sous le nom* `Forum`*.*

Les étapes à suivre sont les suivantes :

– *du coté serveur :*

1. *récupérer la référence de l'objet mandataire (proxy) du service de nommage en utilisant sa référence initiale* `"NameService"` *comme vous l'avez déjà fait pour le POA racine ;*

[1] La variable de nom `args` correspond au paramètre de la fonction `main()`

2. *la convertir vers un type utilisable au moyen de la méthode* `narrow()` *de la classe* `NamingContextExtHelper` *;*
3. *associer à un nom la référence de l'objet que vous souhaitez faire connaître via la méthode* `bind()`*, en prenant soin de vérifier que le nom n'est pas déjà présent. Si ce n'est pas le cas, il faudra écraser l'association déjà présente au moyen de la méthode* `rebind()`*.*

   Pour la construction du nom associé à la référence de l'objet, on utilisera la méthode `to_name()` *de la classe* `NamingContextExt`*. Rappelons qu'un nom CORBA est constitué d'un ensemble (tableau) de* `NameComponent` *qui possède chacun un champ identificateur (le nom)* `id` *et un champ type* `kind`*.*

– *du coté client*

1. *procéder comme du coté serveur pour la construction de l'objet mandataire du service de nommage ;*
2. *pour obtenir une référence à partir du nom d'un objet utiliser la méthode* `resolve()`[(2)] *de la classe* `NamingContextExt` *;*
3. *une fois la référence obtenue le client peut convertir la référence vers le type* `Forum` *en utilisant la méthode* `narrow()` *de la classe* `ForumHelper`*.*

▸ Exercice 2. *Une arborescence de services de nommage*

Dans cet exercice nous allons créer une arborescence de stockage de références (arbre d'objets de type `NamingContext`*).*

Créer sous la racine un nouveau contexte de nom `CTX` *via la méthode* `bind_new_context()`*, puis enregistrer une référence d'objet dans ce nouveau contexte via la méthode* `bind()`

Il est possible de désenregister les références en détruisant les liaisons d'objet avec la méthode `unbind()` *et de modifier l'architecture en détruisant une liaison de contexte au moyen de la méthode* `destroy()` *si le contexte est vide.*

▸ Exercice 3. *Recherche dans la structure*

Dans l'arborescence de contexte, la résolution de nom se fait avec un nom absolu ce qui n'est pas toujours très simple.

Développer une méthode de recherche récursive `explore()` *qui parcourt les conteneurs du service de nommage pour rechercher un nom. Pour cela, lister le contexte racine pour identifier les sous-contextes puis les parcourir récursivement.*

Rappelons que la méthode `list()` *de la classe* `NamingContext` *permet d'obtenir la liste des éléments d'un conteneur dans un tableau de* `Binding`*. Un élément de type*

(2) Il est également possible d'utiliser la méthode `resolve_string()`.

`BindingIterator` est également obtenu, il permet de récupérer les éléments suivants. Le nombre d'éléments retournés dans la liste initiale est fixé lors de l'appel à `list()`.

Rappelons également que les objets manipulés par l'intermédiaire des objets de type `Holder` correspondent à des paramètres `out` et que les séquences CORBA (`BindingList`) sont représentées en Java par des tableaux.