Servicio de Nombres

Sistemas Distribuidos: Programación con CORBA

Juan Pavón Mestras
Dep. Sistemas Informáticos y Programación
Universidad Complutense Madrid

Referencias a objetos

- Para poder utilizar un objeto CORBA es necesario tener su referencia:
  - como resultado de una operación invocada en algún objeto:
    - por ejemplo, una Factoría
    - el Servicio de Nombres o el Servicio de Trading (estándar)
  - leyendo la referencia en forma de string y convirtiéndola con el método *string_to_object()* proporcionado por el ORB
- *string_to_object()* es limitado para sistemas distribuidos (el string tiene que leerse de alguna parte, por ejemplo en un fichero o una dirección URL)
- El servicio de Nombres es la mejor manera de hacer aplicaciones CORBA portables y completamente distribuidas

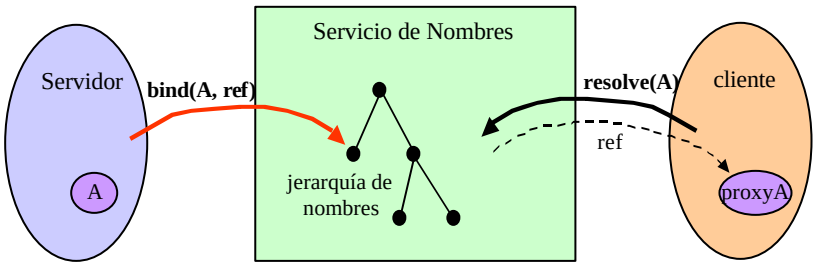
Servicios CORBA bien conocidos

- Como hay que empezar por alguna parte, el ORB está configurado para dar las referencias de los servicios básicos, los servicios CORBA "bien conocidos":
  - Servicio de Nombres ("NameService")
  - Servicio de Trader ("TradingService")
  - Repositorio de Interfaces ("InterfaceRepository")
- Para obtener la referencia a uno de estos servicios (por ejemplo el Servicio de Nombres):

```
NamingContext ns = null;
org.omg.CORBA.Object objeto = null;
objeto = orb.resolve_initial_references("NameService");
ns = NamingContextHelper.narrow(objeto);
```


Uso del Servicio de Nombres

- El Servicio de Nombres guarda pares
  <**nombre, referencia a objeto**>
  - Los nombres están organizados en una jerarquía
- El Servicio de Nombres es usado por cliente y servidor:
  - El servidor asocia (*bind*) en el Servicio de Nombres una referencia a objeto con un nombre
  - El cliente puede pedirle al Servicio de Nombres que a partir de un nombre le dé (*resolve*) una referencia a un objeto CORBA

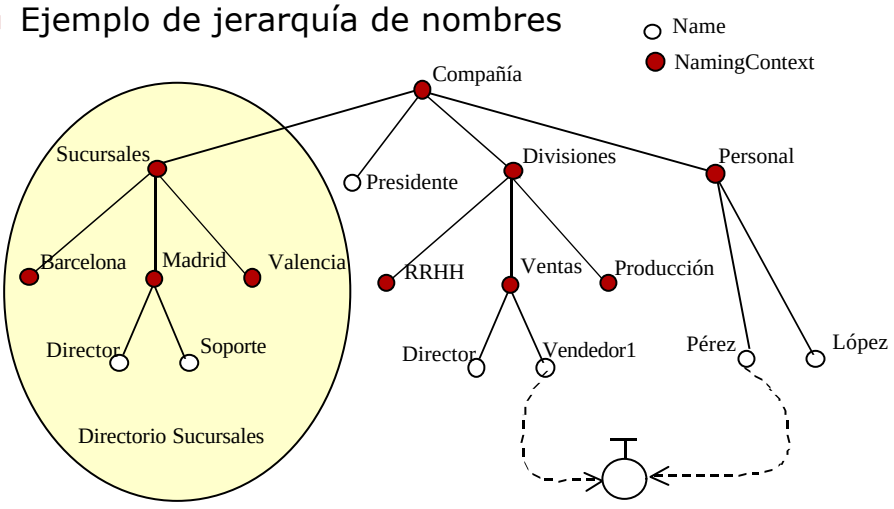


Jerarquía de nombres

- Los nombres en el Servicio de Nombres están organizados jerárquicamente
  - Ejemplos de jerarquías: sistema de ficheros, direcciones y dominios en Internet, etc.
- Cada nodo en la jerarquía de nombres puede ser:
  - NamingContext
    - Define un espacio de nombres
  - Name
    - Puede tener asociado una referencia a un objeto
- Un objeto puede tener asociados varios nombres
- La organización de la jerarquía de nombres es la que quiera el programador
    - El Servicio de Nombres de CORBA es suficientemente flexible para adaptar otros servicios de directorio fácilmente (por ejemplo, DCS CDS, ISO X.500, Sun NIS+, Internet LDAP)

Jerarquía de nombres

- Ejemplo de jerarquía de nombres



Nombres

- Cada Name está formado por varios NameComponent
- Cada NameComponent es un par: <identificador, clase>

```
typedef string Istring;
typedef sequence <NameComponent> Name;
struct NameComponent {
        Istring     id;
        Istring     kind;
};
```

- id es el nombre que identifica el objeto NameComponent
- kind cualifica id (no es obligatorio darle un valor)
- Ejemplos:
  ("C:","volumen")("usuario","dir")("juan","dir")("ejemplo.idl","archivo idl")
  ("Compañía","")("Sucursal","")("Madrid","Ciudad")("Director","cargo")

NamingContext

- La interfaz NamingContext define las operaciones principales del servicio de Nombres
  - Buscar y listar referencias a objetos a partir de nombres
  - Crear la jerarquía de nombres y asociaciones nombre-objeto

```
interface NamingContext {

    Object resolve (in Name n)
                raises (NotFound, CannotProceed, InvalidName);

    void list (in unsigned long how_many,
                out BindingList b, out BindingIterator bi);

// continúa
```

NamingContext

```
        void destroy() raises (NotEmpty);
        NamingContext new_context();
        void unbind(in Name n) raises (NotFound, CannotProceed, InvalidName);
        void bind(in Name n, in Object obj)
          raises (NotFound, CannotProceed, InvalidName, AlreadyBound);
        void rebind(in Name n, in Object obj)
          raises (NotFound, CannotProceed, InvalidName);
        void bind_context(in Name n, in NamingContext nc)
          raises (NotFound, CannotProceed, InvalidName, AlreadyBound);
        void rebind_context(in Name n, in NamingContext nc)
          raises (NotFound, CannotProceed, InvalidName);
        void bind_new_context(in Name n)
          raises (NotFound, CannotProceed, InvalidName, AlreadyBound);
    };
```


BindingList

- La operación list devuelve una lista de bindings:

```
enum BindingType { nobject, ncontext };
struct Binding {
    Name binding_name;
    BindingType binding_type;
};
typedef sequence<Binding> BindingList;
void list (in unsigned long how_many, out BindingList bl, out BindingIterator bi);
```

- La interfaz BindingIterator permite recorrer la lista de bindings:

```
interface BindingIterator {
    boolean next_one(out Binding b);
    boolean next_n(in unsigned long how_many, out BindingList bl);
    void destroy();
};
```

Creación de un nombre y asociación a un objeto

- Normalmente es el servidor quien crea un nombre y le asocia a una referencia a objeto
  1) Obtiene el NamingContext raíz llamando a *resolve_initial_references("NameService")*
  2) Crea los NamingContext necesarios (si no existen) usando *bind_new_context()*
  3) Crea las asociaciones a las referencias a objetos usando *bind()*

Ejemplo de programación del servidor

```
// Crea el objeto Contador
ImplContador unObjetoContador = new ImplContador();
Contador c = unObjetoContador._this(orb);

// Consigue la referencia al servicio de Nombres
org.omg.CORBA.Object nsObj =
                        orb.resolve_initial_references("NameService");

NamingContext ns = NamingContextHelper.narrow(nsObj);

// registra el objeto c en el servicio de Nombres con el nombre "Contador 1"
NameComponent nc = new NameComponent("Contador 1","");
NameComponent[] nombreContador = { nc };

ns.rebind(nombreContador, c);

// el servidor pasa a esperar peticiones ...
```

Excepciones en el Servicio de Nombres

- En NamingContext define las siguientes excepciones:
  - NotFound — Algún componente del nombre especificado no está en la jerarquía de nombres
  - InvalidName — El nombre especificado no es válido
  - AlreadyBound — El objeto ya está asociado con el nombre dado
  - NotEmpty — El NamingContext tiene al menos un binding
- Por lo tanto el código anterior debería estar en un bloque *try...catch* para tratarlas

Ejemplo de programación del servidor (continuación)

Excepciones

```
try {
        // código de uso del NS
}
catch(org.omg.CosNaming.NamingContextPackage.CannotProceed cnp) {
        System.err.println("Excepción en ns.rebind: " + cnp);
}
catch(org.omg.CosNaming.NamingContextPackage.NotFound nf) {
        System.err.println(" Excepción en ns.rebind : " + nf);
}
catch(org.omg.CosNaming.NamingContextPackage.InvalidName in) {
        System.err.println(" Excepción en ns.rebind : " + in);
}
catch(org.omg.CORBA.ORBPackage.InvalidName in) {
        System.err.println(" Excepción en resolve_initial_references : " + in);
}
catch(org.omg.CORBA.SystemException e) {
        System.err.println("System Exception: " + e);
}
```

Encontrar un nombre

- El cliente busca un nombre y obtiene el objeto asociado
  1) Obtiene el NamingContext raíz llamando a *resolve_initial_references("NameService")*
  2) Crea un Name, que puede ser compuesto por uno o más NameComponent
  3) Obtiene la referencia a un objeto invocando *resolve()* sobre el NamingContext raíz
  4) Hace *narrow* de la referencia a objeto conseguida

Ejemplo de programación del cliente

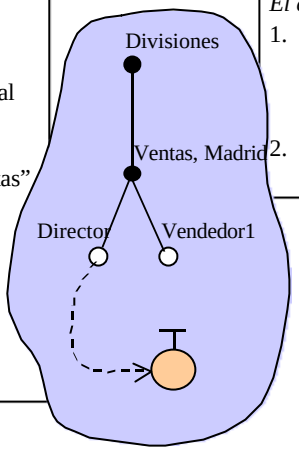
```
// Consigue la referencia al servicio de Nombres
org.omg.CORBA.Object nsObj =
                        orb.resolve_initial_references("NameService");

NamingContext ns = NamingContextHelper.narrow(nsObj);

// obtiene la referencia al objeto "Contador 1" en el servicio de Nombres
NameComponent nc = new NameComponent("Contador 1","");
NameComponent[] nombreContador = { nc };

Contador c = ContadorHelper.narrow(ns.resolve(nombreContador));

// sigue la ejecución, con el objeto Contador1 ya disponible
```

Usando una jerarquía de nombres

- Consideramos cómo registrar el Director de la división de ventas

El servidor tendrá que:

1. Definir un contexto de nombrado llamado "Divisiones" y asociarlo al raíz
2. Definir un contexto de nombrado llamado "Ventas" y asociarlo al anterior
3. Definir un nombre "Director"
4. Asociar el nombre a una referencia al objeto del Director en el contexto anterior



El cliente tendrá que:

1. Construir un nombre compuesto para "Divisiones"-"Ventas"-"Director"
2. Obtener la referencia al objeto asociado al nombre

Usando una jerarquía de nombres

El servidor

```
// Consigue la referencia al servicio de Nombres
org.omg.CORBA.Object nsObj =
                        orb.resolve_initial_references("NameService");
NamingContext ns = NamingContextHelper.narrow(nsObj);

// Primer contexto: "Divisiones"
NameComponent[] divisiones = { new NameComponent("Divisiones",""); };
NamingContext divisionesNC = ns.bind_new_context(divisiones);

// Segundo contexto: "Ventas"
NameComponent[] ventas = { new NameComponent("Ventas","Madrid"); };
NamingContext ventasNC = divisionesNC.bind_new_context(ventas);

// Nombre: "Director"
NameComponent[] director = { new NameComponent("Director",""); };
Persona p = new ImplementacionPersona();  // se crea la persona del Director...
ventasNC.bind (director, p);
```

Usando una jerarquía de nombres

El cliente

```
// Consigue la referencia al servicio de Nombres
org.omg.CORBA.Object nsObj =
                        orb.resolve_initial_references("NameService");
NamingContext ns = NamingContextHelper.narrow(nsObj);

// Prepara el nombre compuesto: “Divisiones” “Ventas” “Director”
NameComponent[] nombreDirector = new NameComponent[3];
nombreDirector[0] =  new NameComponent(“Divisiones","");
nombreDirector[1] =  new NameComponent(“Ventas ",“Madrid");
nombreDirector[2] =  new NameComponent(“Director ","");

// Resuelve el nombre en el contexto raíz
org.omg.CORBA.Object objetoDirector = ns.resolve(nombreDirector);
// Hace narrow a Persona:
Persona p = PersonaHelper.narrow(objetoDirector);
```


Resumen

- El servicio de nombres permite asociar nombres a los objetos CORBA
  - Los clientes pueden conseguir la referencia a un objeto CORBA utilizando un nombre, totalmente independiente de la distribución física del sistema
  - Los nombres pueden ser compuestos y están organizados jerárquicamente
- Al ser el servicio de Nombres un servicio CORBA bien conocido es posible que cualquier aplicación CORBA acceda a él usando
  orb.resolve_initial_references(“NameService”)
- Hay que tener cuidado al usar el servicio de nombres:
  - Destruir correctamente los contextos
  - Destruir los iteradores cuando no sean necesarios
  - Publicar referencias que no sean nulas ni transitorias
  - No conviene utilizar caracteres especiales (“/”, “.” , “*”) en los nombres