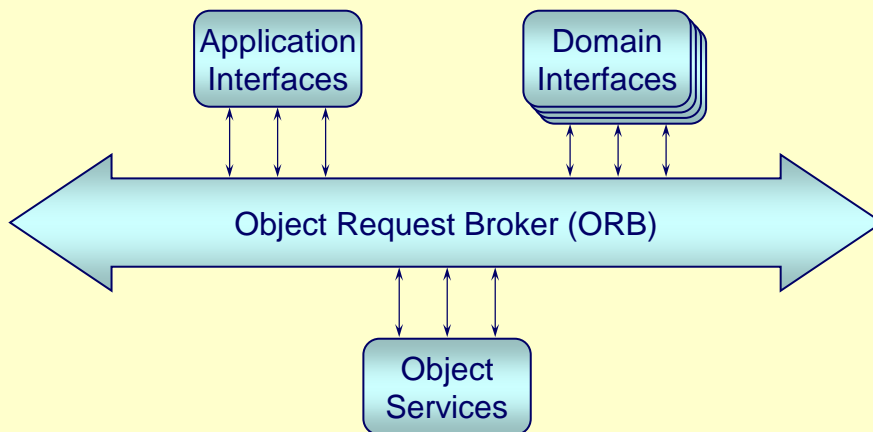


O Serviço de Nomes CORBA

Reverendo a OMA

OMA (Object Management Architecture):



Object Services

- Naming service
 - Dado o nome do objeto, obter a object reference
 - Analogia: lista telefônica
- Trading service
 - Analogia: lista de páginas amarelas
- Event service
 - Propagação de eventos assíncronos
- Security service
 - Segurança
- Transaction service
 - Transações distribuídas

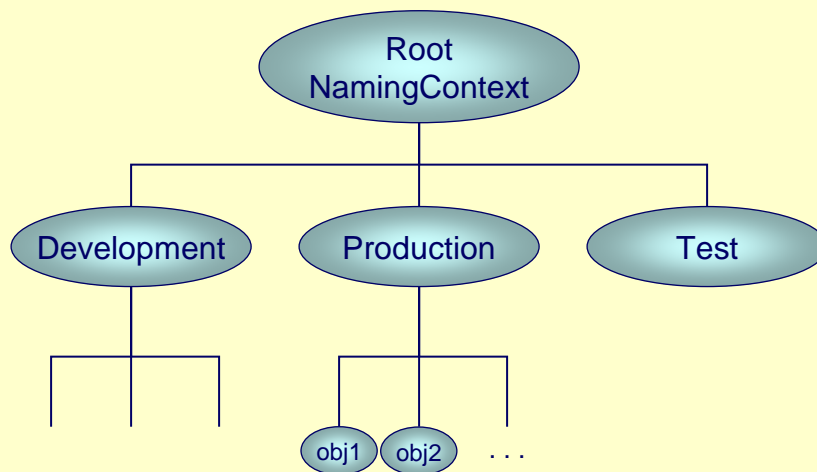
Naming Service

- Permite que nomes sejam associados a object references
- Clientes obtém object references consultando o naming service
 - Eles não precisam se basear em object references convertidas para strings
- Como o clientes de um objeto chegam a ele pelo nome, você pode trocar o objeto sem precisar alterar os clientes
 - Registre o novo objeto com o naming service

Names e NamingContexts

- Um **NamingContext** é um objeto que contém um conjunto de **Names**
- Cada **Name** pode estar associado a um objeto ou a um **NamingContext**
 - o **Name** é relativo ao **NamingContext** que o contém
 - os **Names** de um **NamingContext** podem estar associados a outros **NamingContexts**, permitindo a criação de grafos de nomes

Hierarquias de Nomes



Estrutura do Módulo CosNaming

```
// File: CosNaming.idl
#pragma prefix "omg.org"
module CosNaming {
    // Definições de tipos (structs, sequences, etc...)
    ...
    // Definições de interfaces
    interface NamingContext {
        ...
    };
    interface BindingIterator {
        ...
    };
    interface NamingContextExt {
        ...
    };
};
```



Estrutura dos Nomes

- Um **Name** é uma sequência de **NameComponents**
- Cada **NameComponent** é um par (**id**, **kind**)
- Analogia: nomes de arquivos da forma nome.ext
 - o nome corresponde ao **id**
 - a extensão (ext) corresponde ao **kind**
- Muitos consideram que a distinção entre **id** e **kind** é uma má idéia
- Recomendação:
 - Sempre deixe o campo **kind** vazio
 - Se quiser um nome com extensão, coloque ambos no campo **id**, usando o caracter “.” como separador



Estrutura dos Nomes (cont.)

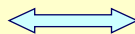
```
// File: CosNaming.idl
module CosNaming {
    // Definições de tipos
    typedef string Istring;
    struct NameComponent {
        Istring id;
        Istring kind;
    };
    typedef sequence<NameComponent> Name;

    // Mais definições de tipos...
};
```

Resolução de Nomes

- A resolução de um nome é sempre relativa a um contexto (o contexto inicial).
- O serviço de nomes procura nesse contexto uma associação com a primeira componente do nome
- Se o nome tiver mais de uma componente, essa associação deve ser com outro contexto
- A resolução do restante do nome prossegue no contexto associado à primeira componente

`cxt->op([c1, c2,... ,cn])`



`cxt->resolve([c1])->op([c2, c3,..., cn])`

A Interface NamingContext

```
// File: CosNaming.idl
module CosNaming {
    ...
    interface NamingContext {
        ...
        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);
    }
};
```



A Interface NamingContext (cont.)

```
Object resolve(in Name n)
    raises(NotFound, CannotProceed, InvalidName);
void unbind(in Name n)
    raises(NotFound, CannotProceed, InvalidName);
NamingContext new_context();
NamingContext bind_new_context(in Name n)
    raises(NotFound, CannotProceed, InvalidName,
        AlreadyBound);
void destroy() raises(NotEmpty);
void list(in unsigned long how_many,
    out BindingList bl,
    out BindingIterator bi);
};
...
};
```



Registrando um Nome (em Java)

```
// Cria o objeto quoter
QuoterImpl quoterServant = new QuoterImpl("Reuters");
org.omg.CORBA.Object quoter =
    quoterServant._this_object(orb);

// Registra o quoter no serviço de nomes
org.omg.CORBA.Object obj =
    orb.resolve_initial_references("NameService");
NamingContext nc = NamingContextHelper.narrow(obj);
NameComponent[] name = new NameComponent[1];
name[0] = new NameComponent("ServiçoDeCotações", "");
nc.bind(name, quoter);
```

Obtendo um **NamingContext** Inicial

- Um problema do tipo “ovo e galinha”: como um programa obtém um **NamingContext** inicial?
 - No exemplo anterior, como o objeto obteve uma referência para um **NamingContext**?
- Solução: a operação **resolve_initial_references** é suportada diretamente pelo ORB
 - Ela é um naming service minimal

Resolvendo um Nome (em Java)

```
org.omg.CORBA.Object obj =
    orb.resolve_initial_references("NameService");
NamingContext nc = NamingContextHelper.narrow(obj);
NameComponent[] name = ...    // o nome a resolver;
try {
    obj = nc.resolve(name);
}
catch (NotFound nf) {
    System.out.println("Nome não encontrado... ");
}
```

Tipos Auxiliares para a Operação **list**

```
// File: CosNaming.idl
module CosNaming {
    // Definições de tipos
    ...
    enum BindingType {nobject, ncontext};
    struct Binding {
        Name binding_name; // devia ser NameComponent!!!
        BindingType binding_type;
    };
    typedef sequence<Binding> BindingList;

    // Definições de interfaces
    ...
};
```


Interface Auxiliar para a Operação **list**

```
// File: CosNaming.idl
module CosNaming {
    ...
    interface BindingIterator {
        boolean next_one(out Binding b);
        boolean next_n(in unsigned long how_many,
                      out BindingList bl);
        void destroy();
    };
    ...
};
```

- Problema:
 - E se os clientes esquecerem de chamar **destroy**?




Defeito da Interface **NamingContext**

- A maioria de suas operações recebem **Names** (seqüências de **NameComponents**)
 - É inconveniente lidar com **Names**
- Solução (na nova versão no serviço de nomes):
 - Uma convenção para nomes “stringficados”:
 - O caracter “/” é usado para separar os componentes do nome
 - O caracter “.” é usado para separar os campos **id** e **kind** de cada componente
 - O caracter “\” é usado como “escape”
 - Exemplos: “a/b/c”, “id1.kind1/id2.kind2/id3”
 - Uma nova interface (**NamingContextExt**) com operações de conversão entre **Names** e nomes “stringficados”.



A Interface **NamingContextExt**

```
module CosNaming {
    ...
    interface NamingContextExt : NamingContext {
        typedef string StringName;
        typedef string Address;
        typedef string URLString;
        StringName to_string(in Name n) raises(InvalidName);
        Name to_name(in StringName sn) raises(InvalidName);
        exception InvalidAddress {};
        URLString to_url(in Address addr, in StringName sn)
            raises(InvalidAddress, InvalidName);
        Object resolve_str(in StringName n)
            raises(NotFound, CannotProceed, InvalidName,
                AlreadyBound);
    };
};
```

 IME-USP
Departamento de Ciência da Computação

19

Copyright © 1998, 1999 Francisco Reverbel

Naming Service: Sumário

- Serviço básico e bastante utilizado
- Operações **bind**, **unbind** e **rebind** para object references e para **NamingContexts**
- Operações para resolver nomes e para listar o conteúdo de um **NamingContext**
- Operações para criar e destruir um **NamingContext**
- Flexibilidade: não impõe ao usuário uma política de definição de nomes

O CosNaming Completo

```
// File: CosNaming.idl
#pragma prefix "omg.org"
module CosNaming {
    typedef string Istring;
    struct NameComponent {
        Istring id;
        Istring kind;
    };
    typedef sequence<NameComponent> Name;
    enum BindingType {nobject, ncontext};
    struct Binding {
        Name binding_name;      // should be NameComponent!!!
        BindingType binding_type;
    };
    typedef sequence<Binding> BindingList;
    interface BindingIterator; // forward declaration
}
```



O CosNaming Completo (cont.)

```
interface NamingContext {
    enum NotFoundReason {
        missing_node, not_context, not_object
    };
    exception NotFound {
        NotFoundReason why;
        Name rest_of_name;
    };
    exception CannotProceed {
        NamingContext cxt;
        Name rest_of_name;
    };
    exception InvalidName {};
    exception AlreadyBound {};
    exception NotEmpty {};
}
```



O CosNaming Completo (cont.)

```
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);
Object resolve(in Name n)
    raises(NotFound, CannotProceed, InvalidName);
void unbind(in Name n)
    raises(NotFound, CannotProceed, InvalidName);
NamingContext new_context();
```



O CosNaming Completo (cont.)

```
NamingContext bind_new_context(in Name n)
    raises(NotFound, CannotProceed, InvalidName,
           AlreadyBound);
void destroy() raises(NotEmpty);
void list(in unsigned long how_many,
          out BindingList bl,
          out BindingIterator bi);
}; // end of interface NamingContext

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



O CosNaming Completo (cont.)

```
interface NamingContextExt : NamingContext {
    typedef string StringName;
    typedef string Address;
    typedef string URLString;
    StringName to_string(in Name n) raises(InvalidName);
    Name to_name(in StringName sn) raises(InvalidName);
    exception InvalidAddress {};
    URLString to_url(in Address addr, in StringName sn)
        raises(InvalidAddress, InvalidName);
    Object resolve_str(in StringName n)
        raises(NotFound, CannotProceed, InvalidName,
            AlreadyBound);
};
}; // end of module CosNaming
```