

Serviço de Nomes CORBA e Interoperabilidade de ORBs

Serviço de Nomes CORBA

■ Páginas Brancas

- Permite encontrar objetos através de nomes
- Nomes → Referência de Objeto
- Essa associação é denominada **name binding**
- Um **name context** é o espaço onde o nome do objeto é único
- Nomes são relativos a **name contexts**

Serviço de Nomes CORBA

■ Object Name Service

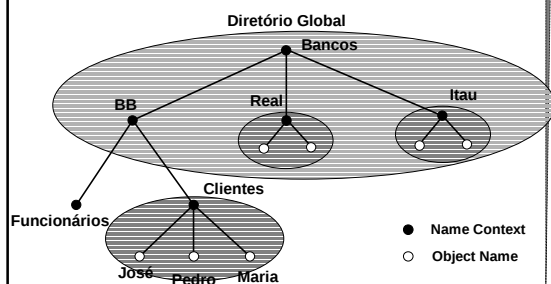
- Padrão OMG desde 1993
- Encapsula os conceitos de serviços conhecidos, como X.500 e DNS
- A idéia é permitir a criação de hierarquias de nomes
- Clientes podem navegar em contextos de nomes procurando objetos

Serviço de Nomes CORBA

■ CORBA Object Name

- Uma sequência de nomes que formam uma hierarquia (**Compound Name**)
- Cada elemento é um **nome de contexto**, exceto o último que é o **nome simples do objeto**
- Os contextos são manipulados através da interface **NamingContext**

Serviço de Nomes CORBA



Serviço de Nomes CORBA

■ CORBA Object Name

- Um **Compound Name** define o caminho a se seguir para se "resolver" nomes

Bancos BB Clientes José

- Realizar um **bind** é criar uma associação nome/objeto em um determinado contexto
- Os nomes são manipulados através de uma Struct IDL denominada **NameComponent**

Serviço de Nomes CORBA

■ CORBA Object Name

- Cada componente do nome é uma estrutura com dois atributos:
 - Identificador - String com o nome do Objeto
 - Tipo - String que define o tipo do componente

© 2002-2003 Carlos A. G. Ferraz

7

Funcionamento do Serviço de Nomes

■ Interface NamingContext

- bind, rebind
 - Associa um objeto a um nome dentro de um contexto;
- unbind
 - Remove um objeto de um contexto;
- new_context
 - Retorna um contexto;

© 2002-2003 Carlos A. G. Ferraz

8

Funcionamento do Serviço de Nomes

■ Interface NamingContext

- bind_new_context
 - Cria um contexto e associa com um nome fornecido;
- destroy
 - Apaga um contexto de nome;
- resolve
 - Recupera um objeto através de um nome em um determinado contexto;

© 2002-2003 Carlos A. G. Ferraz

9

Funcionamento do Serviço de Nomes

■ Estrutura de Dados NameComponent

```
struct NameComponent{  
    string id;  
    string kind;  
};
```

© 2002-2003 Carlos A. G. Ferraz

10

Funcionamento do Serviço de Nomes

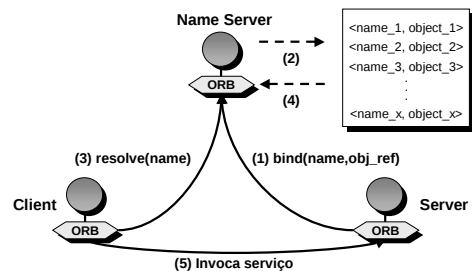
■ Pacote Java

- O pacote (package) Java que implementa o serviço de nomes é o CosNaming
- Todos os ORBs Java têm que utilizar este package como interface para o serviço de nomes (org.omg.CosNaming)

© 2002-2003 Carlos A. G. Ferraz

11

Funcionamento do Serviço de Nomes

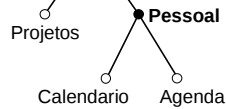


© 2002-2003 Carlos A. G. Ferraz

12

Cenários - Criando Namespace

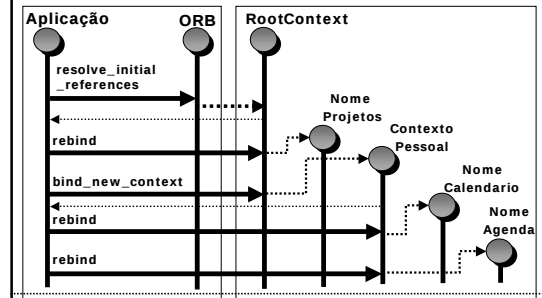
Contexto de Nomes Inicial (root)



© 2002-2003 Carlos A. G. Ferraz

13

Cenários - Criando Namespace



© 2002-2003 Carlos A. G. Ferraz

14

Cenários - Criando Namespace

■ Obter o contexto de nomes inicial

- Através do comando `resolve_initial_references()`;

```
org.omg.CORBA.Object objRef=
orb.resolve_initial_references("NameService");
```

```
NamingContext rootContext =
NamingContextHelper.narrow(objRef);
```

© 2002-2003 Carlos A. G. Ferraz

15

Cenários - Criando Namespace

■ Criar o nome chamado "Projetos"

- Inicialmente deve-se criar um `NameComponent` para Projetos
- Deve-se utilizar o construtor que recebe duas strings (id, kind)
- Será usado o tipo "Text" para esse nome

© 2002-2003 Carlos A. G. Ferraz

16

Cenários - Criando Namespace

■ Criar o nome chamado "Projetos"

```
NameComponent comp1 = new
NameComponent("Projetos","Text");
```

```
NameComponent[] name1 = {comp1};
```

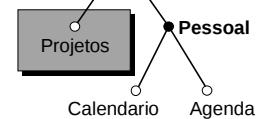
```
// Associação da referência do objeto (gerada
// pelo POA) ao nome "Projetos"
rootContext.rebind(name1,objref);
```

© 2002-2003 Carlos A. G. Ferraz

17

Cenários - Criando Namespace

Contexto de Nomes Inicial (root)



© 2002-2003 Carlos A. G. Ferraz

18

Cenários - Criando Namespace

■ Criar contexto chamado "Pessoal"

- Criar um contexto chamado "Pessoal" relativo ao rootContext
- Inicialmente deve-se criar o nome "Pessoal" e conectá-lo ao contexto rootContext
- bind_new_context deve ser invocado para se criar um novo NameContext
- Será utilizado o tipo "diretorio" para esse contexto

© 2002-2003 Carlos A. G. Ferraz

19

Cenários - Criando Namespace

■ Criar contexto chamado "Pessoal"

```
NameComponent comp2 = new
    NameComponent("Pessoal", "diretorio");

NameComponent[] name2 = {comp2};

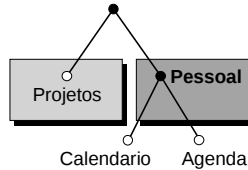
NamingContext PessoalContext =
    rootContext.bind_new_context(name2);
```

© 2002-2003 Carlos A. G. Ferraz

20

Cenários - Criando Namespace

Contexto de Nomes Inicial (root)



© 2002-2003 Carlos A. G. Ferraz

21

Cenários - Criando Namespace

■ Criar os nomes "Calendario" e "Agenda" e associá-los ao contexto "Pessoal"

- As referências dos objetos devem ser criadas inicialmente
 - No exemplo serão usados os valores objref3 e objref4 para representar essas referências (geradas pelo POA a partir dos Servants)
- Os nomes devem ser criados e associados às referências e depois conectados ao contexto devido

© 2002-2003 Carlos A. G. Ferraz

22

Cenários - Criando Namespace

■ Criar os nomes "Calendario" e "Agenda" e associá-los ao contexto "Pessoal"

```
NameComponent comp3 = new
    NameComponent("Calendario", "text");
NameComponent[] name3 = {comp3};

NameComponent comp4 = new
    NameComponent("Agenda", "text");
NameComponent[] name4 = {comp4};

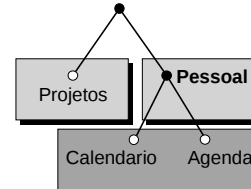
PessoalContext.rebind(name3, objref3);
PessoalContext.rebind(name4, objref4);
```

© 2002-2003 Carlos A. G. Ferraz

23

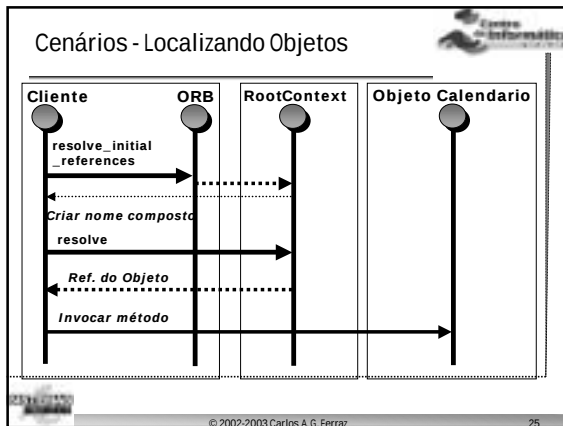
Cenários - Criando Namespace

Contexto de Nomes Inicial (root)



© 2002-2003 Carlos A. G. Ferraz

24



Cenários - Localizando Objetos

- Obter o contexto de nomes inicial
 - Através do comando resolve_initial_references();

```

org.omg.CORBA.Object objRef =
    orb.resolve_initial_references("NameService");

NamingContext rootContext =
    NamingContextHelper.narrow(objRef);
  
```

© 2002-2003 Carlos A. G. Ferraz 26

Cenários - Localizando Objetos

- Construir o nome composto a ser procurado

```

NameComponent comp1 = new
    NameComponent("Pessoal", "diretorio");

NameComponent comp2 = new
    NameComponent("Calendario", "text");

NameComponent[] name = {comp1, comp2};
  
```

© 2002-2003 Carlos A. G. Ferraz 27

Cenários - Localizando Objetos

- Encontrar o objeto com o nome especificado

```

org.omg.CORBA.Object CalendarioRef =
    rootContext.resolve(name);

Calendario CalendarioObj =
    CalendarioHelper.narrow(CalendarioRef);
  
```

© 2002-2003 Carlos A. G. Ferraz 28

Cenários - Localizando Objetos

- Invocar o método desejado
 - O método do objeto deve ser invocado como se este fosse local

```

// Utilizando Invocação Estática

<tipo_resultado> resultado =
    CalendarioObj.<método>(<params>);
  
```

© 2002-2003 Carlos A. G. Ferraz 29

Interoperabilidade entre ORBS CORBA

© 2002-2003 Carlos A. G. Ferraz

Interoperabilidade

- A idéia de interoperabilidade entre ORBs CORBA surgiu com o CORBA 2.0
- A partir dessa versão foi introduzido um protocolo para permitir interoperabilidade

© 2002-2003 Carlos A. G. Ferraz

31

Interoperabilidade



General Inter-ORB Protocol

© 2002-2003 Carlos A. G. Ferraz

32

Interoperabilidade

- GIOP (General Inter-ORB Protocol) permite a interação entre ORBs de fabricantes diferentes
- GIOP é independente da arquitetura do ORB
- GIOP é executado sobre um protocolo de transporte orientado a conexão
- GIOP também é neutro com relação a esse protocolo de transporte

© 2002-2003 Carlos A. G. Ferraz

33

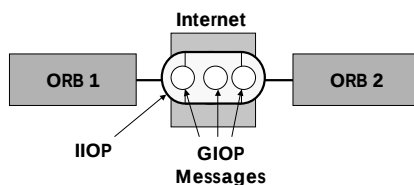
GIOP e IIOP

- O IIOP (Internet Inter-ORB Protocol) é uma especialização do GIOP para TCP/IP
- IIOP define como agentes abrem conexões TCP/IP e as utiliza para transferir mensagens GIOP

© 2002-2003 Carlos A. G. Ferraz

34

GIOP e IIOP



© 2002-2003 Carlos A. G. Ferraz

35

GIOP e IIOP

- GIOP e IIOP resolvem problemas de interoperabilidade da seguinte forma:
 - Permitindo que diferentes ORBs interajam escondendo fatores de implementação
 - Suportando todas as funcionalidades de CORBA através de diferentes ORBs
 - Preservando o conteúdo e a semântica das informações específicas entre os ORBs

© 2002-2003 Carlos A. G. Ferraz

36

O IIOP

- Formado por duas partes distintas
 - Common Data Representation (CDR)
 - Tipos de Dados CORBA
 - Representação de baixo nível
 - Mensagens GIOP
 - Handshaking de comunicação

© 2002-2003 Carlos A. G. Ferraz 37

Common Data Representation

- CDR é a sintaxe utilizada para transferência de informações em comunicações IIOP
- Usada para mapear os tipos de dados CORBA IDL em uma representação comum de mais baixo nível
- Suporta:
 - Tipos Primitivos – char, short, float, boolean, etc.
 - Tipos Construídos – union, array, sequence etc.
 - Pseudo-objetos – exceptions
 - Referências de Objetos

© 2002-2003 Carlos A. G. Ferraz 38

Mensagens GIOP

- 8 formatos de mensagens no total
- Mensagens geradas apenas pelo Cliente
 - Request msg
 - LocateRequest msg
 - CancelRequest msg
- Mensagens geradas apenas pelo Servidor
 - Reply msg
 - LocateReply msg
 - CloseConnection msg

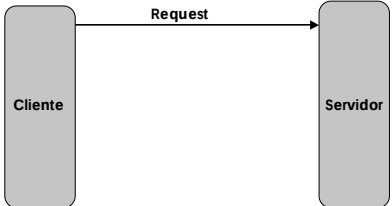
© 2002-2003 Carlos A. G. Ferraz 39

Mensagens GIOP

- Cliente e Servidor suportam ambas
 - MessageError msg
 - Fragment msg

© 2002-2003 Carlos A. G. Ferraz 40

Mensagens GIOP



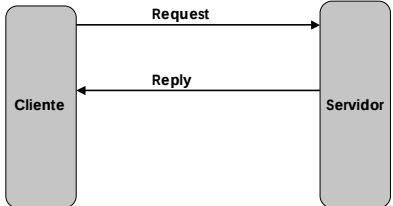
```

graph LR
    Cliente[Cliente] -- Request --> Servidor[Servidor]
  
```

TCP/IP : GIOP message header, Request header (ID), Request body (parâmetros)

© 2002-2003 Carlos A. G. Ferraz 41

Mensagens GIOP

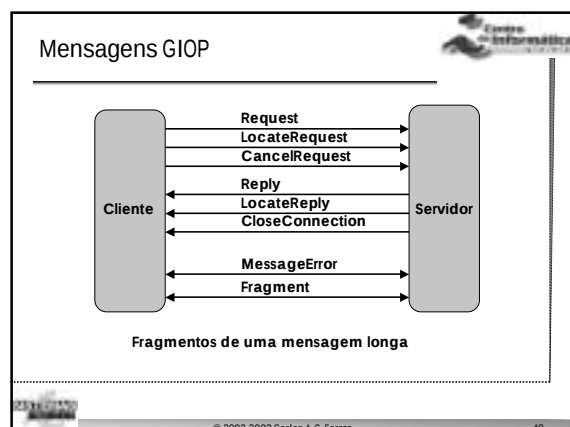
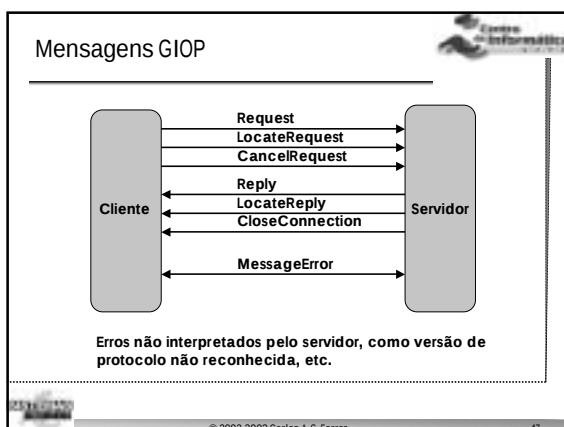
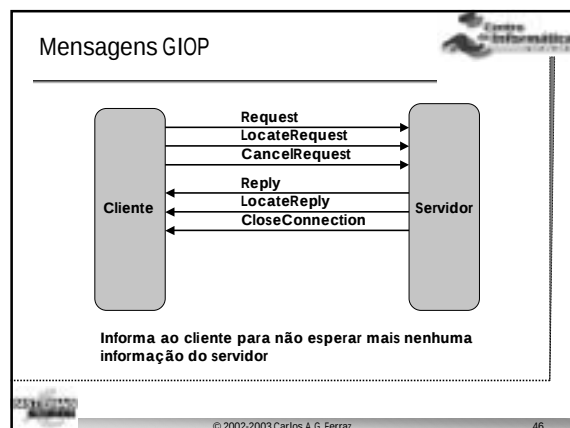
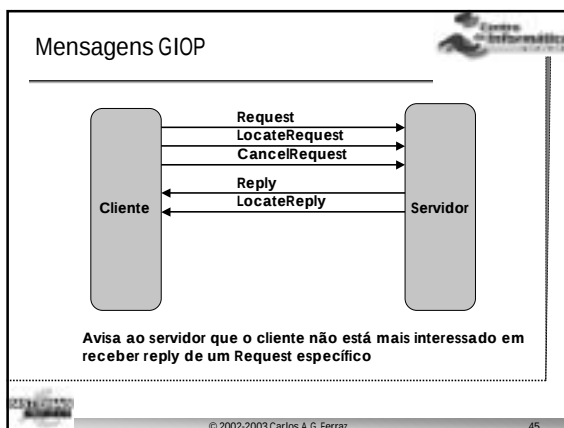
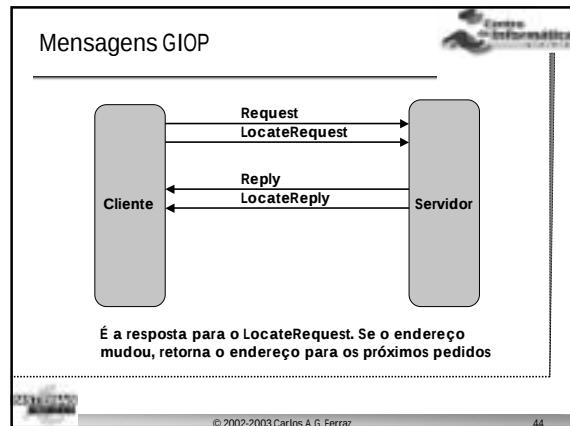
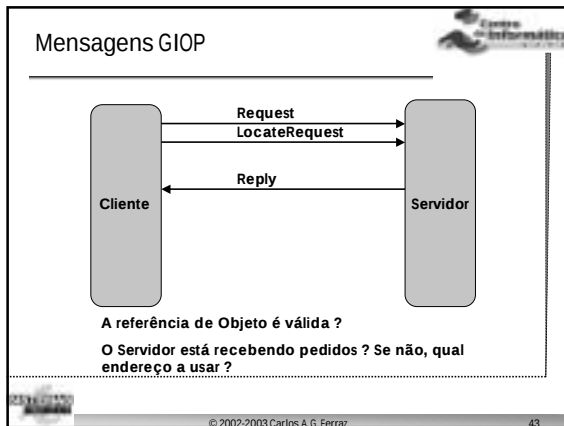


```

graph LR
    Cliente[Cliente] -- Request --> Servidor[Servidor]
    Servidor -- Reply --> Cliente
  
```

GIOP header, Reply header (ID do Request), Status Code (se no_exception retorna os parâmetros out e inout)

© 2002-2003 Carlos A. G. Ferraz 42



Conclusão

- GIOP é um protocolo independente de qualquer tipo de rede particular
- GIOP por si só é insuficiente para permitir comunicação cliente-servidor
- IIOP é o mapeamento de GIOP em redes TCP/IP
- IIOP é o protocolo que o OMG adotou como padrão e deve ser suportado por TODOS os ORBs CORBA, ou de forma nativa ou através de pontes

Conclusão

- IIOP requer que agentes possam aceitar pedidos de objetos ou fornecer localização de objetos através de seus endereços IP
- Esses endereços são publicados através de IORs (Interoperable Object References)
- Clientes requerem serviços de objetos inicializando uma conexão através de um IOR específico