CORBA em Java

Introdução

- CORBA – *Common Object Request Broker Architecture*
- Norma (da indústria) destinada a possibilitar a invocação remota de objetos (http://www.omg.org/)
- Desenvolvida pelo designado *Object Management Group* (OMG), um consórcio de entidades da indústria

Introdução

- Compatível com uma grande variedade de tipos de redes, sistemas operativos e linguagens
- Serviços e clientes CORBA podem ser realizados recorrendo a diversas linguagens de programação
- Não é uma linguagem de programação em si, mas sim uma forma de definir a interacção entre objetos/serviços distribuídos

Introdução

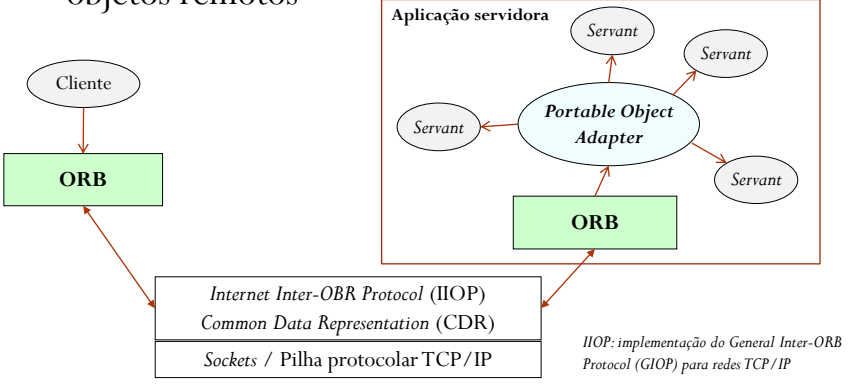
- Os serviços CORBA, i.e., objetos exportados para utilização remota, são descritos através de um esquema, recorrendo à linguagem IDL (*Interface Definition Language*)
- IDL: especificação de interfaces para RPC entre componentes desenvolvidos em linguagens distintas e possivelmente situados em máquinas distintas

Introdução

- É possível passar um esquema IDL para qualquer linguagem de programação para a qual esteja definido um mapeamento: C, C++, Java, COBOL, Python, etc.

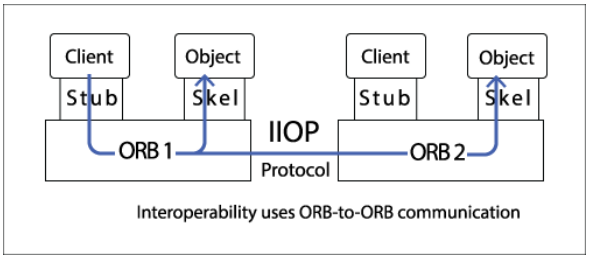
Arquitetura CORBA

- *Object Request Broker* (ORB): objetos/componentes que operam como intermediários entre clientes e objetos remotos



IIOP: implementação do General Inter-ORB Protocol (GIOP) para redes TCP/IP

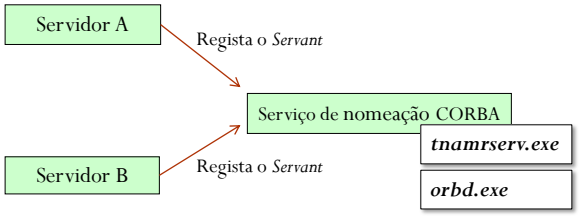
Arquitetura CORBA



https://www.corba.org/faq.htm

Serviços CORBA

- Descritos por esquemas
- Realizados por *servants*
- Os *servants* CORBA são criados e registados num serviço de nomeação por aplicações servidoras
- Serviço de nomeação ⇒ independência de localização



Interface Definition Language

- Linguagem IDL ⇒ descreve os serviços CORBA
- Suporta tipos de dados primitivos e complexos

Java	IDL
void	void
boolean	boolean
char	wchar
byte	octet
short	short
int	long
long	long Long
float	float
double	double
java.lang.String	string / wstring

Interface Definition Language

index in sequence of bytes	← 4 bytes →	*Notes on representation*
0–3	5	*Length of string*
4–7	"Smit"	*'Smith'*
8–11	"h___"	
12–15	6	*Length of string*
16–19	"Lond"	*'London'*
20-23	"on__"	
24–27	1984	*unsigned long*

This CDR message represents a *Person* struct with value: {'Smith', 'London', 1984}

Coulouris, Dollimore, Kindberg, and Blair, "Distributed Systems: Concepts and Design", Ed. 5, Addison-Wesley, 2012

Interface Definition Language

- Interfaces: descrevem os objetos remotos, os métodos que expõem e variáveis membro

```
interface UserAccount
{
   float getBalance();
   void setBalance(in float amount);
};
```


Interface Definition Language

- Módulos: agrupamento lógico de interfaces
  - Resulta em *packages* quando se opta por mapeamentos para Java

```
module AccountTypes
{
   interface UserAccount{
      // ...
   };
   interface UserID{
      // ...
   };
   interface Subscriptions{
      // ...
   };
};
```

Interface Definition Language

- Atributos
  - Possibilitam não recorrer a *getters/setters*
  - Podem ser mutáveis ou imutáveis

```
interface UserAccount
{
   // balance is not protected, and can be easily changed
   attribute float balance;

   // account id may be read but not remotely modified
   readonly attribute long long accountid;
};
```


Interface Definition Language

- Operações
  - Funções que podem aceitar parâmetros e retornar resultados
  - Existem três tipos de parâmetros: ***in*** (i.e., apenas de entrada); ***out*** (i.e., pode ser modificado); e ***inout*** (i.e., propriedades ***in*** + ***out***)

```
interface UserBalance
{
   float getBalance();
   float addBalance(in float amount);
   float subtractBalance(in float amount);
};
```

Interface Definition Language

- Excepções

```
module BankingSystem
{
    // Account inactive exception
    exception AccountInactive
    {
        string reason;
    };
    // Account overdrawn exception
    exception AccountOverdrawn
    {
        string reason;
    };
    interface BankAccount
    {
        double withdrawMoney(in double amount) raises (AccountInactive,
                                                       AccountOverdrawn);
        // ...
    };
};
```


IDL ⇒ Java

- Definição do esquema

```
exception BulbBrokenException
{
};

interface LightBulb
{
    void on() raises (BulbBrokenException);
    void off();
    boolean isOn() raises (BulbBrokenException);
};
```

Ficheiro *LightBulb.idl*

- Conversão do esquema IDL para Java

```
idlj -fclient LightBulb.idl ⇒ gera apenas o mapeamento para o cliente
idlj -fserver LightBulb.idl ⇒ gera apenas o mapeamento para o servidor

idlj -fall LightBulb.idl ⇒gera os mapeamentos relativos ao cliente e ao
                          servidor
```

IDL ⇒ Java

- Codificação do *Servant* + Servidor

```
public interface LightBulbOperations
{
   void on () throws BulbBrokenException;
   void off ();
   boolean isOn () throws BulbBrokenException;
} // interface LightBulbOperations
```


IDL ⇒ Java

- *LightBulbPOA.java*
  - O código fonte de um *servant* deve estender esta classe e realizar os métodos definidos em *LightBulbOperations.java*

```
import java.util.Random;
import org.omg.CORBA.*;
import org.omg.CosNaming.*;
import org.omg.CosNaming.NamingContextPackage.*;

public class LightBulbServant extends LightBulbPOA
{
   private boolean bulbOn;

   // BOOLEAN FLAG REPRESENTING LIGHT BULB WORKING
   private boolean bulbOk = true;
```

IDL ⇒ Java

```
public LightBulbServant()
{
   bulbOn = false;
}

public void on () throws BulbBrokenException
{
   if (bulbOn)
      return;

   if (!bulbOk)
      throw new BulbBrokenException();

   // CHECK TO SEE IF BULB WILL BREAK
   Random r = new Random();
```


IDL ⇒ Java

```
   // ONE IN TEN CHANCE TO BREAK
   int chance = r.nextInt() % 10;

   if (chance == 0){
      bulbOk = false; bulbOn = false;
      throw new BulbBrokenException();
   }else{
      bulbOn = true;
   }
}

public void off ()
{
   bulbOn = false;
}
```

IDL ⇒ Java

```
public boolean isOn () throws BulbBrokenException
{
   if ( !bulbOk )
      throw new BulbBrokenException();

   return bulbOn;
}

public static void main(String args[])
{
   System.out.println ("Loading ORB");

   try{
      // CREATE AN INITIALIZE AN ORB OBJECT
      ORB orb = ORB.init(args, null);
      // GET A REFERENCE TO THE ROOT POA (PORTABLE OBJECT ADAPTER)
      POA rootpoa = POAHelper.narrow(
               orb.resolve_initial_references("RootPOA"));
```


IDL ⇒ Java

```
public boolean isOn () throws BulbBrokenException
{
   if ( !bulbOk )
      throw new BulbBrokenException();

   return bulbOn;
}

public static void main(String args[])
{
   System.out.println ("Loading ORB");

   try{
      // CREATE AN INITIALIZE AN ORB OBJECT
      ORB orb = ORB.init(args, null);
      // GET A REFERENCE TO THE ROOT POA (PORTABLE OBJECT ADAPTER)
      POA rootpoa = POAHelper.narrow(
               orb.resolve_initial_references("RootPOA"));
```

A classes geradas com nomes terminados em ***Helper*** incluem métodos utilitários. Um dos mais relevantes é o método *narrow()*. Este passa a referência de um objecto CORBA genérico para uma referência de um tipo específico de interface. É comparável aos *casts* para referências remotas específicas em Java RMI.

IDL ⇒ Java

```
        // AND ACTIVATE THE POA MANAGER
        rootpoa.the_POAManager().activate();

        // CREATE A NEW LIGHT BULB SERVANT
        LightBulbServant servant = new LightBulbServant();
        // GET AN OBJECT REFERENCE ASSOCIATED WITH THE SERVANT
        org.omg.CORBA.Object ref =
                    rootpoa.servant_to_reference(servant);

        LightBulb href = LightBulbHelper.narrow(ref);

        // OBJECT REQUEST BROKER READY
        System.out.println ("Light bulb service loaded");

        // NEXT STEP : EXPORT THE ORB TO THE NAMING SERVICE
        // GET A (GENERIC CORBA OBJECT) REFERENCE TO THE NAMING SERVICE
        org.omg.CORBA.Object object =
              orb.resolve_initial_references("NameService");
```


IDL ⇒ Java

```
        // NARROW (I.E, CAST) TO A NAMINGCONTEXTEXT OBJECT
        NamingContextExt namingContext =
                 NamingContextExtHelper.narrow(object);

        // REGISTER THE SERVENT WITH THE NAMING SERVICE
        String name = "LightBulb";
        NameComponent path[] = namingContext.to_name(name);
        namingContext.rebind(path, href);

        System.out.println ("Servant registered");

        // WAIT INDEFINITELY FOR CLIENTS TO USE THE SERVANT.
        orb.run();
      }catch (Exception e){
        System.err.println ("Error - " + e);
      }
   }
}
```

A *thread* actual bloqueia até que outra invoque o método *shutdown* de *orb* (e.g., se for passada uma referência para *orb* ao *servant*, o método *shutdown* pode ser invocado a partir de qualquer um dos seus métodos)

IDL ⇒ Java

- Código do Cliente

```
import org.omg.CORBA.*;
import org.omg.CosNaming.*;

public class LightBulbClient
{
   public static void main(String args[])
   {
      try{
         // First step : initialize an ORB object
         ORB orb = ORB.init(args, null);

         // Second step: lookup the light bulb using a naming service
         System.out.println ("Looking for lightbulb");

         // Get a reference to the (generic CORBA object) naming service
         org.omg.CORBA.Object object =
               orb.resolve_initial_references("NameService");
```


IDL ⇒ Java

```
         // Narrow to a NamingContextExt object
         NamingContextExt namingContext =
                     NamingContextExtHelper.narrow(object);

         // Resolve the (generic CORBA) object Reference in the naming
         // service and narrow it to a LightBulb object
         String name = "LightBulb";
         LightBulb lb = LightBulbHelper.narrow(
                     namingContext.resolve_str(name));

         System.out.println ("Found light bulb");

         // Turn bulb on and off till it fails, demonstrating exceptions
         while (true){
            try{

               lb.on();
               System.out.print ("Light on.... ");
```

LightBulbHelper.java gerado pela aplicação ***idlj.exe***
Actua como proxy.

IDL ⇒ Java

```
                lb.off();
                System.out.print ("Light off.... ");

             }catch (BulbBrokenException e){
                System.err.println();
                System.err.println ("Bulb broken exception");
                break;
             }
          }

       }catch (Exception e){
          System.err.println ("Error - " + e);
       }
    }
}
```


Execução

- Lançar o serviço de nomeação

```
orbd.exe                                  (porto TCP 900)   ou
tnameserv.exe                             (porto TCP 900)   ou
orbd.exe -ORBInitialPort 1050                               ou
tnameserv.exe -ORBInitialPort 1050
```

- Lançar a aplicação servidora

```
java LightBulbServant                                                    ou
java LightBulbServant -ORBInitialPort 1050 -ORBInitialHost localhost
```

- Correr os clientes

```
java LightBulbClient                                                     ou
java LightBulbClient -ORBInitialPort 1050 -ORBInitialHost localhost
```