Exemplo de Cliente e de Servidor CORBA Escritos em C++

Voltando ao Nosso Exemplo

```
// IDL
module Stock {
    exception UnknownStock {
        string name;
    };
    interface Quoter {
        string name();
        long value(in string stock)
            raises(UnknownStock);
    };
};
```

Esqueleto C++ para a Interface Quoter

```
// Código C++ gerado pelo tradutor IDL
namespace POA_Stock {
    class Quoter: public virtual
                  PortableServer::ServantBase {
    public:
        virtual char* name()
            throw(CORBA::SystemException) = 0;
        virtual CORBA::Long value(const char*)
            throw(Stock::UnknownStock,
                  CORBA::SystemException) = 0;
    };
};
```


Classe Servente para a Interface Quoter

```
class QuoterImpl : public POA_Stock::Quoter {
public:
    QuoterImpl(const char* my_name);
    char* name()
        throw(CORBA::SystemException);
    CORBA::Long value(const char* name)
        throw(Stock::UnknownStock,
              CORBA::SystemException);
private:
    string m_name; // classe string ANSI/ISO
    map<string,CORBA::Long> m_map; // map da STL
};
```

QuoterImpl::name()

```
char*
QuoterImpl::name()
    throw(CORBA::SystemException)
{
    char* nm = CORBA::string_dup(m_name.c_str());
    if (nm == 0)
        throw CORBA::NO_MEMORY();
    return nm;
}
```


QuoterImpl::value()

```
CORBA::Long
QuoterImpl::value(const char* stock)
    throw(Stock::UnknownStock,
          CORBA::SystemException)
{
    map<string,CORBA::Long>::iterator iter =
        m_map.find(stock);
    if (iter == m_map.end())
        throw Stock::UnknownStock(stock);
    return (*iter).second;
}
```

QuoterImpl::QuoterImpl()

```
QuoterImpl::QuoterImpl(const char* my_name)
    : m_name(my_name)
{
    // inicializa m_map com informações lidas
    // de um arquivo ou banco de dados
    // ...

    //     (num quoter real, m_map deveria
    //      ser atualizado periodicamente)
}
```


Mainline do Servidor de Cotações

```
int main(int argc, char* argv[]) {
   using namespace CORBA; using namespace Stock;
   ORB_ptr orb = ORB_init(argc, argv);
   Object_ptr obj=
      orb->resolve_initial_references("RootPOA");
   POA_ptr poa = POA::_narrow(obj);
   QuoterImpl quoter_impl("DowJones");
   Quoter_ptr quoter = quoter_impl->_this();
   poa->the_POAManager()->activate();
   orb->run();
   release(quoter); release(poa);
   release(obj); release(orb);
}
```

Ficou faltando uma coisa...

- A mainline do servidor deveria ter criado um arquivo contendo uma object reference (convertida para string) do Quoter.
  - Como fizemos no exemplo em Java
  - Para escrever o cliente C++, vamos supor que existe um arquivo chamado **`quoter.ref`** contendo uma referência para o Quoter

Exemplo de Cliente C++

```
int main(int argc, char** argv)
{
    using namespace CORBA;
    using namespace Stock;
    string objstr; // classe string ANSI/ISO
    ifstream is("quoter.ref");
    is >> objstr;
    ORB_ptr orbref = ORB_init(argc, argv);
    Object_ptr objref =
        orbref->string_to_object(objstr.c_str());
```

Exemplo de Cliente C++ (cont.)

```
    Quoter_ptr quoter = Quoter::_narrow(objref);
    char* name = quoter->name();
    Long value = quoter->value(argv[1]);
    cout << "Cotação de " << argv[1]
         << ", fornecida por " << name << ": "
         << value << endl;
    string_free(name);
    release(quoter);
    release(objref);
    release(orbref);
    return 0;
}
```


Tratando Exceções no Cliente

```
try {
    char* name = quoter->name();
    Long value = quoter->value(argv[1]);
    cout << "Cotação de " << argv[1]
         << ", fornecida por " << name << ": "
         << value << endl;
} catch(UnknownStock& us) {
    cerr << "Não disponível cotação de "
         << us.name << endl;
} catch(SystemException&) {
    cerr << "Exceção de sistema!" << endl;
}
```

_ptrs e _vars

- O mapeamento de IDL para C++ define dois tipos de referências para objetos CORBA:
  - Dada uma interface **A**, o compilador IDL gera definições de dois tipos C++, **A_ptr** e **A_var**
  - Ambos tem semântica de ponteiros
  - Ambos podem ser usados para para invocar operações sobre objetos remotos
  - Uma referência tipo **_ptr** requer gerenciamento de memória explícito
  - Uma referência tipo **_var** faz gerenciamento de memória de modo automático (smart pointer)

Usando referências _var

```
ORB_var orbref = ORB_init(argc, argv);
Object_var objref =
    orbref->string_to_object(objstr.c_str());
Quoter_var quoter = Quoter::_narrow(objref);
String_var name = quoter->name();
Long value = quoter->value(argv[1]);

// não é mais preciso chamar release para
// quoter, objref ou orbref, nem chamar
// string_free para name
```

Observações

- O cliente C++ pode ser usado com o servidor Java
- Se completarmos o servidor C++ (fazendo-o criar o arquivo **`quoter.ref)`**, ele poderá ser acessado pelo cliente Java.
- CORBA provê interoperabilidade entre
  - linguagens de programação
  - sistemas operacionais
  - arquiteturas de hardware