Ambientes de Desenvolvimento Avançados

ADO – ActiveX Data Objects

Teófilo Matos

2001.11.27

Estrutura da Aula

- Introdução aos MDAC´s
  - ADO
- Cliente COM no Browser
  - Utilização do ADO

MDAC´s - *Universal Data Access*

- Estratégia desenvolvida pela *Microsoft®*
- Permitir aos utilizadores um acesso eficiente a qualquer fonte de dados
  - servidores de base de dados
  - correio electrónico
  - documentos de produtividade pessoal – relatórios, apresentações, folhas de cálculos –
  - servidores de informação baseados na *Web*
  - etc ...

MDAC´s - *Universal Data Access*

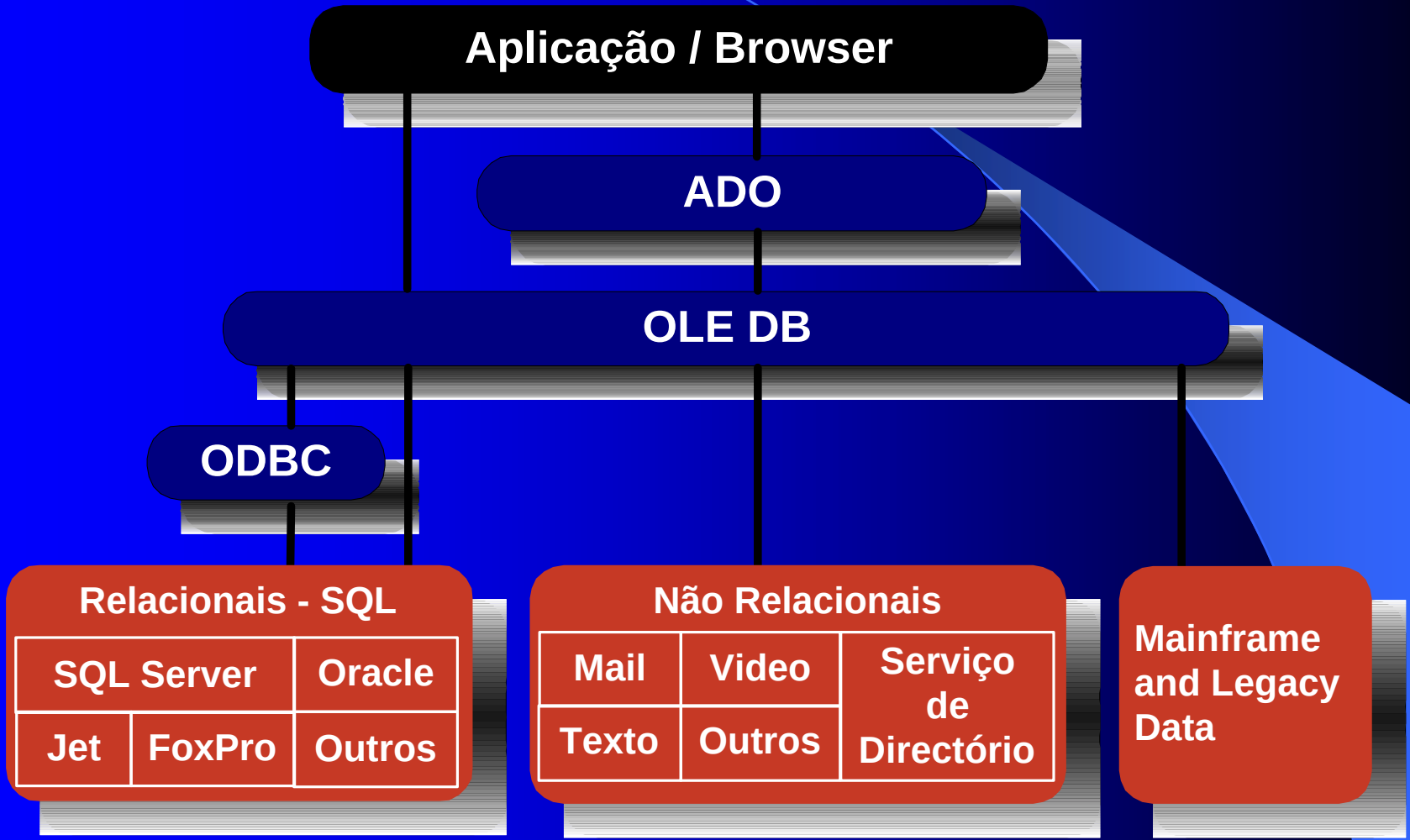
- Disponibiliza um *interface standard*
  - Acesso aos dados – consultas e/ou alterações – de forma transparente
- Especificação independente do armazenamento de dados, ferramentas e linguagens.

  - interface de alto nível (ADO) ⇨ Fácil utilização
  - interface baixo nível (OLE DB) ⇨ Alta performance

MDAC´s - *Universal Data Access*

- Muita flexibilidade do *UDA*
  - integrar vários repositórios
  - utilização de ferramentas, aplicações e plataformas, consideradas mais adequadas
  - assenta, na sua maioria, em ***objectos COM***
  - modelo consistente de programação

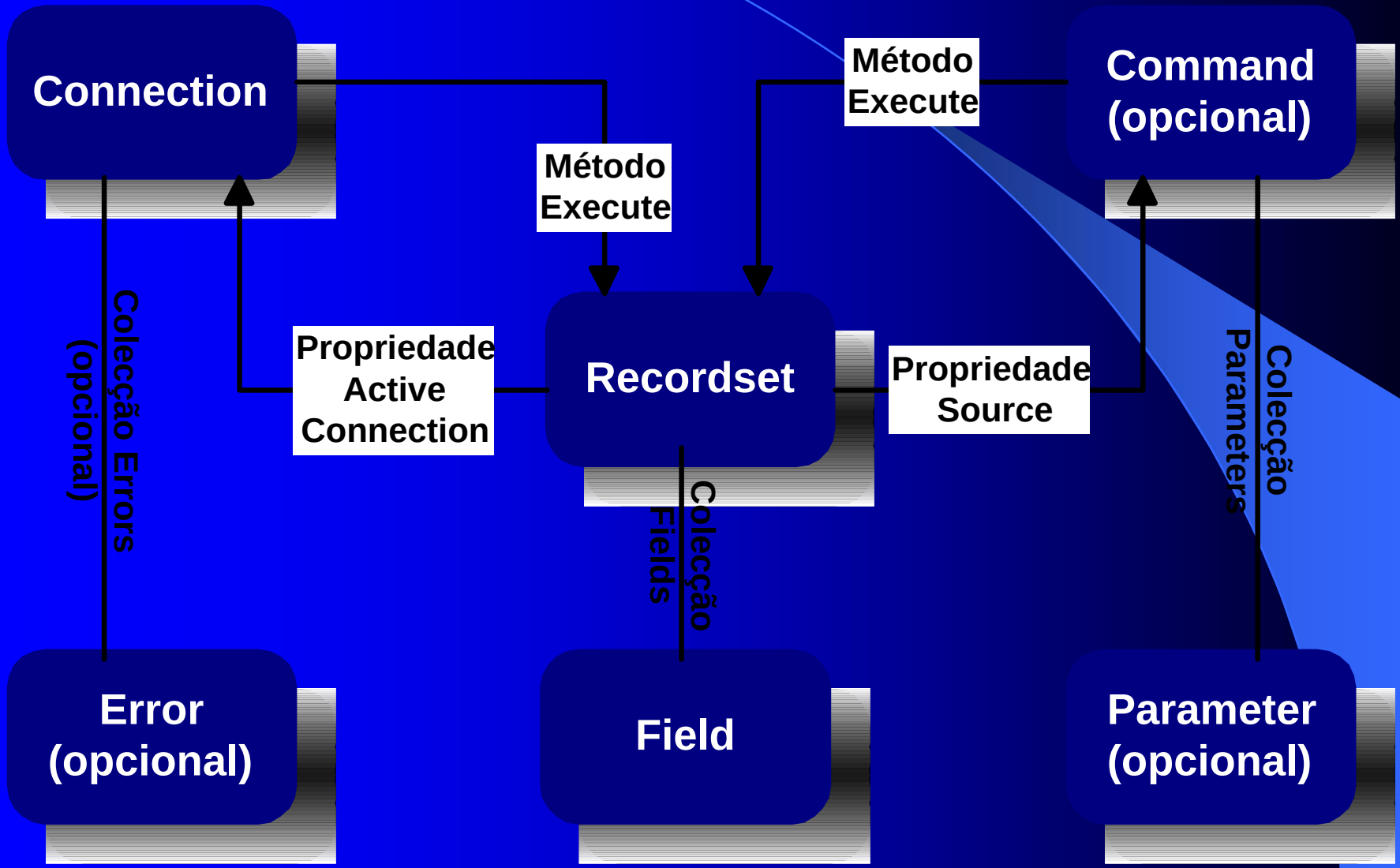
MDAC´s, principais componentes



Componentes do MDAC

- ODBC
  - interface para acesso a bases de dados SQL (relacionais) heterogéneas.
- ADO
  - *interface* usado para aceder aos dados mantém-se constante.
- OLE DB
  - compreendem novos formatos (não relacionais) que residem em novos locais, tais como a *Internet* .
  - Utilização de providers.

ADO – Modelo Objects



ADO – ActiveX Data Objects

- O modelo de objectos do ADO define uma colecção de objectos programáveis que podem ser utilizados com qualquer linguagem de programação em plataformas que suportem COM. Este modelo de objectos foi desenhado para expor as funções mais utilizadas do OLE DB.

ADO – *Connection*

- O objecto *connection* permite que seja estabelecida uma ligação a uma fonte de dados

```
_bstr_t strConn=T("Provider=Microsoft.Jet.OLEDB.4.0;
                  Data Source=lojainform.mdb;User ID=;Password=;");
try{
        ADODB::_ConnectionPtr spConn=NULL;

        HRESULT hr=spConn.CreateInstance( __uuidof(ADODB::Connection) );

        if (FAILED(hr)) com_issue_error(hr);

        spConn->Open(strConn, "", "", adConnectUnspecified );
}

catch( _com_error &e )
{ . . . }
```

ADO – *Error*

- O objecto *Error* permite que seja adquirida informação sobre possíveis erros que ocorram no *provider*.

```
_bstr_t strConn=T("Provider=Microsoft.Jet.OLEDB.4.0;
                    Data Source=lojainform.mdb;User ID=;Password=;");

ADODB::_ConnectionPtr spConn=NULL;
ADODB::ErrorsPtr spErs;
ADODB::ErrorPtr spEr;
try{
        HRESULT hr=spConn.CreateInstance( __uuidof(ADODB::Connection) );
        if (FAILED(hr)) _com_issue_error(hr);
        spConn->ConnectionTimeout = 5;
        spConn->Open(strConn, "", "", adConnectUnspecified );
}
catch( ... )
{
 spErs = spConn->Errors;
 _bstr_t erro="";
 for( long i=0; i<spErs->Count ; i++){
    spEr = spErs->GetItem(i);
    erro+=spEr->SQLState+(_bstr_t)" : "+(_bstr_t)spEr->NativeError+(_bstr_t)" : "+\
          spEr->Description+(_bstr_t)"\n";
 }
 ::MessageBox( NULL , erro , "Erro COM!" , MB_ICONSTOP|MB_OK);
}
```

ADO – *Command*

- O objecto *Command* permite que sejam feitos inquéritos à base de dados e por conseguinte sejam retornados registos num objecto Recordset como resposta ao inquérito.

```
_bstr_t strConn=T(“Provider=Microsoft.Jet.OLEDB.4.0;
                    Data Source=lojainform.mdb;User ID=;Password=;");

ADODB::_ConnectionPtr spConn=NULL;
ADODB::_CommandPtr spComm=NULL;
ADODB::_RecordsetPtr spRs=NULL;
 try{
          HRESULT hr=spConn.CreateInstance( __uuidof(ADODB::Connection) );
          if (FAILED(hr)) _com_issue_error(hr);

          spConn->ConnectionTimeout = 10;
          spConn->Open(strConn, "", "", adConnectUnspecified );

          hr=spComm.CreateInstance( __uuidof(ADODB::Command));
          if (FAILED(hr)) _com_issue_error(hr);

          spComm->ActiveConnection = spConn;
          spComm->CommandText = _bstr_t("SELECT * FROM clientes");
          spComm->CommandTimeout = 15;

           spRs = spComm->Execute(NULL,NULL,ADODB::adCmdText);
}catch( _com_error &e ){ . . . }
```

ADO – *Recordset*

- O objecto Recordset pode representar tanto um conjunto de registos de uma base de dados como o resultado obtido através da execução de um objecto *Command.*
- Documentação de Apoio – página 18.

```
_bstr_t strConn=T("Provider=Microsoft.Jet.OLEDB.4.0;
                Data Source=lojainform.mdb;User ID=;Password=;");

ADODB::_RecordsetPtr rs;
_bstr_t field;
 try{
          HRESULT hr = rs.CreateInstance( __uuidof(ADODB::Recordset) );
          if (FAILED(hr)) _com_issue_error(hr);

          // biblioteca de cursor local
          rs->PutCursorLocation( ADODB::adUseClient );

          // abrir um cursor
          hr = rs->Open("Select * from clientes",
                    strConn, ADODB::adOpenStatic, ADODB::adLockBatchOptimistic,
                    ADODB::adCmdUnspecified);

          // desligar o recordset do servidor
          rs->PutRefActiveConnection(NULL);
...
```

ADO – *Recordset*

```
...
// colocar-se no primeiro registo
 rs->MoveFirst();

 // percorrer todos os registos do recordset
 for(int i=0;i<rs->RecordCount(); i++) {
            // retirar o conteúdo do campo "nome"
            field=rs->Fields->GetItem("nome")->Value;
            // avançar um registo
            rs->MoveNext();
 }
  // fechar o objecto recordset
 rs->Close();
  // destruir o objecto
 rs=NULL;
}
```

```
. . .
  // apontador para o recordset filho Produto
 _RecordsetPtr spRsProd=spRS->Fields->GetItem("PRODUTO")->Value;

 // adiciona um registo ao recordset filho Produto
 spRsProd->AddNew();
 spRsProd->Fields->GetItem("Descricao")->PutValue( “Computador” );
 spRsProd->Fields->GetItem("CustoUnitario")->PutValue( 100L );
 spRsProd->Fields->GetItem("QuantidadeStock")->PutValue( 20L );
 . . .
```

ADO – *Recordset*

```
. . .

spRs->PutCursorLocation(adUseClient);

// abrir o recordset com a estrutura pretendida, mas vazio
spRs->Open( _T("SELECT TOP 0 * FROM clientes'"), strConnect,
                adOpenStatic, adLockOptimistic, -1 );

spRs->AddNew();
spRs->Fields->GetItem("NOME")->PutValue( “João Antunes” );
spRs->Fields->GetItem("MORADA")->PutValue( “Rua das carmelitas” );
spRs->Fields->GetItem("TELEFONE")->PutValue( “221234567” );
. . .
spRs->Update();

. . .
```

Como Integrar o ADO numa aplicação em Microsoft Visual C++

- Incluir a DLL do ADO

Deve ser incluído o ficheiro **msado15.dll**, através da directiva **#import**.

```
#import "C:\Program Files\Common Files\System\ado\msado15.dll" no_namespace \
        rename("EOF", "EndOfFile")
ou
#import "msado15.dll" no_namespace rename("EOF", "EndOfFile")
```

se tivermos colocado previamente o caminho em options|directories|executable files.

- Iniciar o subsistema COM

```
// iniciar o subsistema COM
struct InitOle
{
        InitOle() {::CoInitialize(NULL);}
        ~InitOle() {::CoUninitialize();}
} _init_InitOle_;
```

Cliente COM

- Utilização do Browser
  - Verificar se um determinado cliente existe (uso do ADO)

Exemplificar

Dúvidas

Documentação:

http://www/~tmatos/ADAV/2002_2003/ADAV_ADO.pdf