Curso C# Conceitos Básicos

Diogo Cezar Teixeira Batista

diogocezar@utfpr.edu.br

Universidade Tecnológica Federal do Paraná
Campus Cornélio Procópio
UTFPR-CP

Cornélio Procópio - 2008

Agenda I

Agenda II

- O que é um DataAdapter?
- Criando um DataSet e um DataAdapter
- Criando e preenchendo um DataSet

O que é o ADO.NET ?

- Tecnologia baseada no ADO (*Active Data Objects*);
- Criado para trabalhar com um ambiente desconectado;
- Camada de persistência em XML.

Os namespaces relacionados ao ADO.NET

- Os namespaces utilizados para trabalhar com ADO.NET:
  - *System.Data*: Infra-estrutura básica para trabalharmos com qualquer base de dados relacional;
  - *System.Data.Common*: Interfaces comuns a todos os bancos de dados;
  - *System.Data.SqlClient*: Biblioteca de acesso ao SQL Server;
  - *System.Data.OleDb*: Biblioteca de acesso para bancos de dados que suportam OleDb;
  - *System.Data.SqlTypes*: Definição dos tipos nativos do SQL Server;
  - *System.XML*: Contém as classes para manipulação de documentos XML.

O modelo de execução do ADO.NET I

- Múltiplas bases de dados simultaneamente;
- É possível armazenar duas tabelas de diferentes bancos de dados;
- Estrutura responsável pelo armazenamento dos dados: *DataSet*;
- Os *DataSet* contém um conjunto de objetos (*DataTables*) que representam resultados tabulares extraídos da base de dados.

O modelo de execução do ADO.NET II

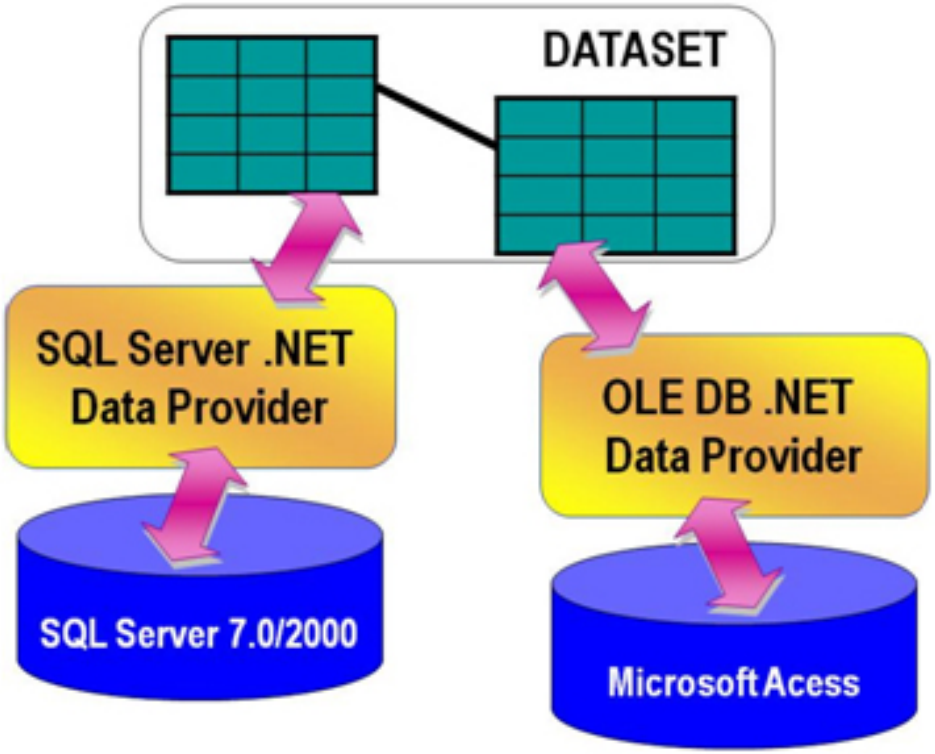

Figura: Esquema acesso ao banco de dados

O modelo de execução do ADO.NET III

- Para extrair os dados: *.NET Data Providers*;
- *Data Providers*: bibliotecas de classes especializadas para o acesso a um tipo de banco de dados relacional;
- Por serem uma implementação específica, são mais eficientes que bibliotecas genéricas como OLEDB ou ODBC;
- Apesar de cada implementação ser específica para o banco de dados, possuem uma estrutura em comum.

O modelo de execução em um ambiente conectado

- O ADO.NET é capaz de trabalhar com dois modelos, o modelo conectado e o modelo desconectado;
- No modelo conectado: necessário manter a conexão aberta enquanto são realizadas as operações de leitura e gravação;
- Para trabalharmos com o modelo conectado do ADO.NET devemos seguir a seguinte ordem:
  1. *XxxConnection*: utilizado para estabelecer a conexão com o banco;
  2. *XxxCommand*: É um objeto utilizado para enviar comandos a base de dados;
  3. *XxxDataReader*: É um objeto utilizado para ler dados de um comando executado.

O modelo de execução em um ambiente desconectado I

- Utiliza outros objetos;
- *DataSet*: armazena e manipula os dados em memória;
- *XxxDataAdapter*: extrai e envia as alterações ao banco de dados;
- Os passos para extração e manipulação dos dados em um ambiente desconectado são:
  1. É aberta uma conexão utilizando um objeto *XxxConnection*;
  2. É criado um objeto do tipo *XxxDataAdapter*: dados para memoria $\rightarrow$ aramazena alterações;
  3. método *Fill* do *XxxDataAdapter* para extrair os dados da base e armazenar em um *DataSet*;
  4. Fechamos a conexão com o banco pois os dados;
  5. É possível inserir, remover ou alterar registros do *DataSet*;

O modelo de execução em um ambiente desconectado II

6. Ao finalizar as alterações, restabelecemos a conexão com o banco de dados para enviar as alterações;
7. Utilizase o comando *XxxCommandBuilder* para gerar as strings sql que vão alterar o *XxxDataAdapter*;
8. Utilizando o método *Update* do *DataAdapter*, enviamos as alterações para o banco de dados;
9. Ao finalizar o processo, fechamos a conexão com o banco de dados.

Estabelecendo uma conexão com um banco de dados

- Primeiro passo para uma aplicação que acessa dados de um banco;
- Cria-se uma instância (objeto) da classe que faz a conexão com o banco;
- Ao criar essa instância informa-se uma *Connection String* que contém parâmetros para conexão no banco, como usuário e senha;
- A *string* de conexão possui uma série de parâmetros que podem variar de acordo com o banco de dados utilizado;
- Os parâmetros são separados por ponto e virgula.

Código 1: Padrão para Connection Strings

```
1  "Nome do Parâmetro = Valor do Parâmetro"
```

Exemplos de Connection Strings

Código 2: Exemplos de Connection Strings

```
1  class Global
2  {
3          public static string cnmysql = "database = dados;
               data source = localhost; user id = root;
               password = password";
4
5          public static string cnfirebird = "User=SYSDBA;
               Password = masterkey; Database = C:\\dados.fdb;
               DataSource = localhost; Dialect=3; Charset=
               WIN1252; Role=; Connection lifetime=15; Pooling=
               true; MinPoolSize=0; MaxPoolSize=50; Packet Size
               =4096; ServerType=0";
6
7          public static string cnpostgree = "User ID=postgres;
               Password=password;Host=localhost;Port=5432;
               Database=dados;Pooling=true;Min Pool Size=0;Max
               Pool Size=100;Connection Lifetime=0";
8  }
```

Criando comandos I

- É possível executar comandos no banco através da classe: *SqlCommand*;
- Ao se criar uma instância deve-se informar a consulta SQL bem como a Conexão com o banco;
- Esses parâmetros podem ser informados no construtor ou através das propriedades *CommandText* e *Connection*;
- Os SQL's informados podem ser de qualquer tipo:
  - Retornando um conjunto de registros;
  - Retornando um valor específico;
  - Sem retorno.
- Para cada um desses casos existe um método específico para execução.

Criando comandos II

Código 3: Exemplo de utilização do comando SqlCommand

```
1  SqlCommand oCmd = New SqlCommand("UPDATE Products SET
       UnitPrice=UnitPrice*1.1");
2  oCmd.Connection = oConn;
3  oCmd.CommandText = "UPDATE Products SET UnitPrice=UnitPrice
       *1.05";
```

Executando comandos

- Os métodos de execução variam de acordo com a natureza do comando executado;
- Os três métodos mais comuns são:
  - *ExecuteNonQuery*: Para comandos que não executam consultas (querys);
  - *ExecuteScalar*: Para comandos que executam resultados escalares;
  - *ExecuteReader*: Para comandos que retornam conjuntos de dados.

O método ExecuteNonQuery

Definição

É utilizado quando queremos executar um comando que não retorna como resultado um conjunto de dados.

- Utilizado para executar DCL (*Data Control Language*) suportados pelo banco de dados;
- Opcionalmente podemos informar um parâmetro para este método para obter o número de linhas afetadas pelo comando executado.

Código 4: Exemplo de utilização do comando ExecuteNonQuery

```
1  SqlCommand oCmd = new SqlCommand("UPDATE Products SET
       UnitPrice=UnitPrice*1.1");
2  oCmd.Connection = oConn;
3  oCmd.CommandText = "UPDATE Products SET UnitPrice=UnitPrice
       *1.05";
4  oCmd.ExecuteNonQuery();
```

O método ExecuteScalar

Definição

É utilizado para comandos que retornam valores escalares, ou seja, valores únicos.

- Em geral é utilizado para comandos que retornam uma contagem de registros;
- Este comando pode retornar qualquer tipo de dado.

Código 5: Exemplo de utilização do comando ExecuteScalar

```
1  SqlCommand oCmd = new SqlCommand("UPDATE Products SET
       UnitPrice=UnitPrice*1.1");
2  oCmd.Connection = oConn;
3  oCmd.CommandText = "SELECT COUNT(*) FROM Products";
4  int iNumProdutos;
5  iNumProdutos = oCmd.ExecuteScalar();
```

O método ExecuteReader I

Definição

É utilizado para executar consultas (*querys*) que retornam um conjunto de dados.

- Este método tem como resultado um objeto do tipo *SqlDataReader*.
- A classe *SqlDataReader* representa um cursor aberto no banco de dados com os dados retornados;
- Para lermos os dados de um *DataReader*, é necessário executamos o método *Read*;

O método ExecuteReader II

- Com o *DataReader* não é possivel executar nenhuma outra operação com a mesma conexão aberta, por isso deve-se fechar ao término da execução.

Código 6: Exemplo de utilização do comando ExecuteReader

```
1  SqlCommand oCmd = new SqlCommand("SELECT ProductName,
       ProductId FROM Products", oConn);
2  SqlDataReader oDr;
3  oDr = oCmd.ExecuteReader();
4  while (oDr.Read()) {
5          Debug.WriteLine(oDr("ProductName").ToString());
6  }
```

Passando parâmetros I

- É possível passar parâmetros para os objetos da classe *SqlCommand*;
- Para indicarmos parâmetros nas *querys* utilizamos o símbolo @ como prefixo para indicar um parâmetro;
- Esta sintaxe pode variar de acordo com o banco de dados utilizado (o Oracle utiliza ":" por exemplo);
- Depois de indicar os parâmetros na query, é preciso adicionar objetos do tipo *SqlParameter* na coleção de parâmetros do *SqlCommand*.

Passando parâmetros II

Código 7: Exemplo de utilização de parâmetros

```
1  SqlConnection oConn = new SqlConnection(sConnString);
2  SqlCommand oCmd = new SqlCommand();
3  oCmd.Connection = oConn;
4  oCmd.CommandText = "UPDATE Products " + " SET UnitPrice=
       @UnitPrice " + " WHERE ProductId=@ProductId";
5  SqlParameter oParam = new SqlParameter("@UnitPrice", 1.5);
6  oCmd.Parameters.Add(oParam);
7  oParam = new SqlParameter();
8  oParam.ParameterName = "@ProductId";
9  oParam.DbType = DbType.Int32;
10 oParam.Value = 1;
11 oCmd.Parameters.Add(oParam);
12 oCmd.ExecuteNonQuery();
```

O que é um DataSet?

Definição

O *DataSet* é uma classe capaz de armazenar múltiplos resultados tabulares em uma mesma estrutura.

- Composto por *DataTables* que representam estes resultados tabulares;
- Para extrairmos dados da base de dados e preenchermos o *DataSet* utilizamos a classe *DataAdapter*;

O que é um DataAdapter?

Definição

O *DataAdapter* é a classe responsável por fazer a interação entre a base de dados e o *DataSet*.

- Capaz de executar os quatro comandos básicos de um banco de dados (*Insert*, *Update*, *Delete*, *Select*);
- Para extrairmos dados da base de dados e preenchermos o *DataSet* utilizamos a classe *DataAdapter*;

Criando um DataSet e um DataAdapter

Instruções

Quando criamos um *DataAdapter* é possível informar uma *query* e uma conexão para a extração dos dados.

Código 8: Criando um DataSet e um DataAdapter

```
1  SqlDataAdapter daProduct = new SqlDataAdapter("SELECT * FROM
       Products", oConn);
2  SqlDataAdapter daOrders = new SqlDataAdapter();
3  SqlCommand oCmd = new SqlCommand("SELECT * FROM Orders",
       oConn);
4  daOrders.SelectCommand = new SqlCommand(oCmd);
```

Criando e preenchendo um DataSet

Instruções

Para criar um novo *DataSet* basta utilizar o comando New e criar um novo objeto. Para preencher um *dataset* utilizando um *DataAdapter*, devemos utilizar o método *Fill* do *DataAdapter*, informando o *DataSet* e nome da tabela a ser criada no *DataSet*.

Código 9: Criando e preenchendo um DataSet

```
1  SqlDataAdapter daProduct = new SqlDataAdapter("SELECT * FROM
       Products", oConn);
2  DataSet ds = new DataSet();
3  daProduct.Fill(ds, "Products");
```