Visual Basic .NET

ADO .NET

Professor: Danilo Giacobo

Página pessoal: www.danilogiacobo.eti.br
E-mail: danilogiacobo@gmail.com

Objetivos da aula

- ✓ Introdução
- ✓ A biblioteca de classes do ADO.NET
- ✓ Principais conceitos do ADO.NET
- ✓ O Objeto Connection
- ✓ O Objeto Command
- ✓ O Objeto DataReader
- ✓ O Objeto DataAdapter
- ✓ O Objeto DataTable
- ✓ O Objeto DataView
- ✓ O Objeto DataSet

Introdução

- O **ADO.NET** (*ActiveX Data Objects*) consiste de um conjunto de classes definidas pelo .NET Framework que podem ser utilizadas para acessar dados de um banco de dados local ou remoto.
- Utiliza uma arquitetura de dados desconectada:
  - ✓ A aplicação se conecta a fonte de dados apenas para salvar ou recuperar registros;
  - ✓ Garante uma ótima utilização do sistema porque os recursos **ADO.NET** não mantém bloqueios de banco de dados ou conexões ativas para períodos de tempo prolongados.
- O .NET framework fornece um conjunto de classes organizadas no espaço nome **System.Data**. Essa coleção de classes é comumente referida como **ADO.NET**.

Biblioteca

- Todos os recursos **ADO.NET** são oferecidos por meio dos seguintes *Namespaces*:

- **System.Data:** contém as classes fundamentais para gerenciar dados;
- **System.Data.Common:** possui classes bases que são herdadas por outras;
- **System.Data.Odbc:** Possui classes para realizar conexão com provedor ODBC;
- **System.Data.OleDb:** Possui classes para realizar conexão com provedor OLE DB;
- **System.Data.Sql:** Possuem classes que suportam funcionalidade específicas do banco de dados SQL Server.
- **System.Data.SqlClient:** Possui classes para conexão com banco de dados SQL Server via interface TDS (*Tabular Data Stream*).
- **System.Data.SqlTypes:** Fornece classes para tipos de dados nativos em SQL Server. Essas classes fornecem uma alternativa mais segura, mais rápida para tipos de dados fornecidos pelo *Common Language Runtime* (CLR) do .NET Framework.

Escalabilidade

- A arquitetura de dados desconectada permite que os aplicativos sirvam a mais usuários de forma mais eficiente em comparação com uma arquitetura de dados conectada.
- Isto porque na arquitetura de dados desconectada, a conexão com a fonte de dados é encerrada assim que os dados são recuperados da fonte de dados, deixando a fonte de dados disponível para outros usuários.
- Em outras palavras, as aplicações criadas no **ADO.NET** pode gerir de modo eficaz vários usuários.

Desempenho

- No **ADO .NET** usa-se **XML** (*eXtensible Markup Language*) para transferir dados entre aplicações sendo nenhuma conversão de tipos de dados é requerido e dessa forma tem-se um melhor desempenho.

Programação

- O **ADO.NET** permite uma programação rápida e fácil com um mínimo de erros permitindo que se use a programação "tipada", a qual torna o código mais legível.
- Com a programação "tipada" tem-se a verificação da existência de erros em tempo de compilação, a qual aumenta a segurança do código.

Interoperabilidade

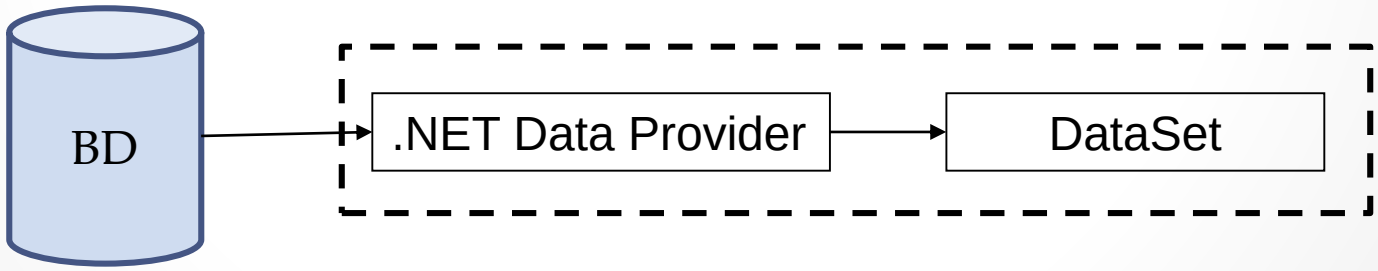
- Com o suporte ao **XML** pelo **ADO.NET** para transferir dados, qualquer aplicativo que interpreta **XML** pode trocar dados com uma aplicação **ADO.NET**.
- Portanto, as aplicações criadas no **ADO.NET** são interoperáveis com outros aplicativos.

Manutenibilidade

- Usando **ADO.NET**, pode-se criar aplicativos que são logicamente divididos em camadas.
- Por exemplo, pode-se criar um aplicativo com camadas separadas para a interface do usuário, lógica de negócios e acesso a dados.
- Dividindo uma aplicação em camadas lógicas simplifica a manutenção da aplicação.
- Além disso, pode-se adicionar camadas para uma aplicação **ADO.NET** como e quando necessário.

Principais conceitos do ADO.NET

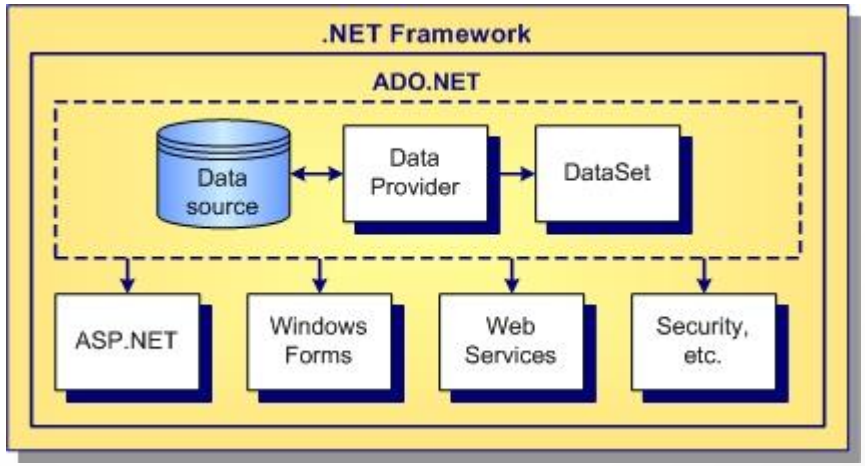
- O **ADO.NET** permite que sejam criadas várias camadas distribuídas e aplicações de compartilhamento de dados permitindo que sejam acessados os dados de um banco de dados relacional, uma fonte **XML** ou outra aplicação.
- Para acessar e gerenciar dados, **ADO.NET** fornece dois componentes:
  - .NET Data Provider
  - DataSet
- O **.NET Data Provider** (ou provedor de dados .NET) é usado para acessar os dados.
- O **DataSet** é usado para manipular os dados.



Principais conceitos do ADO.NET

- Para permitir que um aplicativo possa se comunicar com uma fonte de dados, o **.NET Data Provider** utiliza quatro objetos:

  - Objeto **Connection**
  - Objeto **Command**
  - Objeto **DataReader**
  - Objeto **DataAdapter**



Objeto Connection

- O objeto **Connection** têm a função de gerar uma conexão com uma fonte de dados sendo portanto o objeto fundamental no acesso a dados.
- Para estabelecer uma conexão com uma fonte de dados o objeto **Connection** usa a propriedade **ConnectionString** que é a *string* de conexão que deverá ser informada para que a conexão seja efetivamente aberta.
- Após realizada a conexão com a fonte de dados podemos usar objetos para receber e enviar dados para a fonte de dados.

Exemplo de String de Conexão com o Microsoft Access:

Provider = Microsoft.ACE.OLEDB.12.0; Data Source = {arquivo .accdb}; Persist Security Info = False;

Dica: Para outros tipos de strings de conexão acesse o site:
http://www.connectionstrings.com/

Na pasta:
Exemplo_Connection.sln

Objeto Command

- Os objetos **Command** são usados para executar declarações SQL e procedimentos armazenados (Stored Procedures).
- Os métodos usados para realizar estas tarefas são:

- **ExecuteReader**: executa declarações SQL que retornam linhas de dados, tais como **SELECT**;

- **ExecuteNonQuery**: executa declarações SQL que não retornam dados, tais como **INSERT**, **UPDATE** e **DELETE**;

- **ExecuteScalar**: retorna um valor único como resultado de uma função agregada: **SUM**, **AVG**, **COUNT**, **MAX** e **MIN**.

Na pasta:
Objeto_Command.sln

Objeto DataReader

- O objeto **DataReader** é uma das maneiras mais fáceis para ler os dados retornados pelos objetos **Command**.
- Ele permite acessar e percorrer os registros no modo de somente leitura e apenas para frente (*forward-only*).
- Não oferece acesso desconectado e não permite alterar ou atualizar a fonte de dados original.
- Apresenta poucos recursos mas seu desempenho é muito melhor do que o oferecido pelo objeto **DataSet**.
- Para criar um objeto **DataReader** usa-se o método **ExecuteReader** de um objeto **Command**.

Objeto DataReader

As propriedades e métodos mais usadas dos objetos **DataReader** são:

- **FieldCount:** informa o número de colunas da linha de dados atual.
- **IsClosed:** Indica se o objeto **DataReader** esta fechado.
- **RecordsAffected:** especifica o número de linhas alteradas, excluídas ou incluídas na execução de uma declaração SQL.
- **Item(n):** obtêm o valor da *n-ésima* coluna no seu formato nativo.
- **Close:** método que fecha o objeto.
- **GetName(n):** método que retorna o nome da *n-ésima* coluna.
- **Read:** método que permite ao **DataReader** avançar para o próximo registro.
- **IsDbNull(n):** método que informa se a *n-ésima* coluna possui um valor nulo.
- **HasRows:** indica se o **DataReader** contém uma ou mais linhas.

Objeto DataAdapter

- O **ADO. NET** usa a arquitetura de dados desconectado e essa arquitetura é implementada em um aplicativo usando **DataSets**.
- Um **DataSet** é um banco de dados virtual que armazena os dados recuperados a partir de qualquer fonte de dados.
- O objeto **DataAdapter** age como um elo entre o conjunto de dados e a fonte de dados, ele permite a transferência de dados do **DataSet** para a fonte de dados e vice-versa, e possibilita acessar e manipular os dados.
- Sua função basicamente é a seguinte:
  - Acessar a fonte de dados através de uma conexão prévia;
  - Fazer a consulta na base de dados;
  - Obter os dados; e
  - Preencher o **DataSet**.

Objeto DataAdapter

- Para acessar o banco de dados, executar o comando **SQL** via **DataAdapter**, trazer os dados e preencher o **DataSet**, usa-se o método **Fill**.
- O método **Fill** retorna a linhas de uma fonte de dados usando a declaração **SELECT** definida por uma propriedade **SelectCommand** associada.
- O objeto **Connection** associado com a declaração **SELECT** precisa ser válido mas não precisa estar aberto.
  - Se a conexão for fechada antes da chamada do método **Fill**, ela será aberta para que os dados possam ser retornados e, em seguida, fechada novamente;
  - Se a conexão estiver aberta, ela permanecerá aberta após o uso do método **Fill**.

Objeto DataTable

- Um objeto **DataTable** representa uma ou mais tabelas de dados em memória.
- Os objetos **DataTable** estão contidos no objeto **DataSet** e/ou **DataView**.

Objeto DataTable

Principais propriedades:

- **Columns:** representa as colunas da tabela através da coleção de objetos **DataColumn** (**DataColumnCollection**).
- **Rows:** representa as linhas da tabela através de uma coleção de objetos **DataRow** (**DataRowCollection**).
- **PrimaryKey:** representa a chave primária da tabela através dos objetos **DataColumn**.
- **TableName:** define o nome do objeto **DataTable** via coleção **DatatableCollection** em um objeto **DataSet**.
- **AcceptChanges:** efetiva as alterações realizadas no **DataTable** no banco de dados.
- **NewRow:** gera um novo objeto **DataRow** que representa uma linha de dados;
- **Copy:** copia os dados e a estrutura do **DataTable**.
- **Clear:** limpa os dados de um **DataTable**.
- **RejectChanges:** ignora as alterações feitas no **DataTable**.

Objeto DataView

- Usa-se o **DataView** para mostrar uma visão dos dados contidos em um **DataTable**.
- Pode-se ter vários **DataViews** ligados a um mesmo **DataTable**, sendo que cada um exibe um visão diferente dos dados.
- O objeto **DataTable** possui um **DataView** padrão que é acessado através da propriedade **DefaultView**.

Principais propriedades:

- **RowFilter:** retorna uma expressão usada para filtrar os dados a serem exibidos pelo **DataView**.
- **Count:** informa o número de linhas no **DataView** após a aplicação dos filtros: **RowFilter** e **RowStateFilter**.
- **Item:** obtêm uma linha de dados de um tabela especificada.
- **Sort:** define a coluna que irá ordenar o **DataView** e o tipo da ordenação (ASC ou DESC).
- **Addnew:** inclui uma nova linha no **DataView**.

Objeto DataView

Principais propriedades (continuação):

- **RowStateFilter:** define a versão dos dados que serão exibidos pelo **DataView**. Oferece as seguintes opções:
  - **CurrendRows:** linhas de dados atuais (linhas não alteradas , novas).
  - **Added:** linhas de dados novas.
  - **Deleted:** Linha excluída pelo método **Delete**.
  - **None:** Nenhuma linha.
  - **ModifiedCurrent:** linhas de dados que foram modificadas (versão atual).
  - **OriginalRows:** linhas originais.
  - **Unchanged:** Linhas não modificadas.
  - **ModifiedOriginal:** linhas de dados que foram modificadas (versão original).

- **Table:** define qual é o objeto **DataTable** de origem para o **DataView**.
- **Delete:** exclui uma linha do **DataView**.
- **Find:** busca por uma linha no **DataView**.

Objeto DataSet

- O objeto **DataSet** representa o primeiro dos dois maiores componentes da arquitetura ADO.NET; outro membro são os **Providers**.
- Pode-se resumir os atributos como segue:
  - É baseado em XML;
  - É um conjunto de dados em cache que não esta conectado ao banco de dados;
  - É independente da fonte de dados;
  - Pode armazenar dados em múltiplas tabelas que podem ser relacionadas;
  - Armazena múltipla versões de dados para coluna e para cada linha em cada tabela.

Referências Bibliográficas

- HOLZNER, Steven. **Visual Basic .NET: Black Book**. Arizona: Coriolis Group Books, 2002. xxxviii, 1144 p ISBN 1-57610-835-X.