Capítulo 9

ADO.NET

- DataSet
- DataReader
- Objetos para banco de dados
- Métodos de conexão
- Considerações finais
- Referências bibliográficas
- Glossário

Integrado à plataforma .NET, o ADO.NET é uma tecnologia de acesso a banco de dados. Suas diversas classes permitem acesso a plataformas como SQL Server, MySQL, Oracle, Sybase, Access, XML e arquivos textos. Essas conexões podem ser realizadas de três maneiras: OLE DB , SQL e ODBC.

Criado para trabalhar no formato desconectado, o ADO.NET faz a conexão com a base de dados por meio de um objeto DataAdapter (SqlDataAdapter e OleDbDataAdapter), aumentando, assim, o seu desempenho. Além disso, o ADO.NET possui o objeto DataSet, que é a representação mais próxima do banco de dados. Ele carrega na memória várias tabelas representadas pelo objeto DataTable, além de permitir o relacionamento entre as tabelas por meio do objeto DataRelation. Os provedores de dados que acompanham o ADO.NET possibilitam a utilização de várias classes que interagem diretamente com a base de dados, as quais são identificadas por um prefixo, conforme mostra a tabela 14.

Caso seja necessário utilizar outros sistemas gerenciadores de banco de dados, você pode consultar o seu desenvolvedor a respeito dos serviços de conexão, como o MySQL e PostgreSQL. Para cada provedor de conexão, teremos os objetos representados a partir de seu prefixo da seguinte forma: OleDbConnection,

Tabela 14
Provedores de conexão.

PROVEDOR	DESCRIÇÃO
ODBC Data Provider API Prefixo: Odbc	Geralmente usada para banco de dados mais antigos que utilizam a interface ODBC
OleDb Data Provider API Prefixo: OleDb	Conexão do tipo OleDb, como o Access ou Excel
Oracle Data Provider API Prefixo:Oracle	Para implementação de Banco de Dados Oracle
SQL Data Provider API Prefixo:Sql	Para implementação de Banco de Dados Microsoft SQL Server

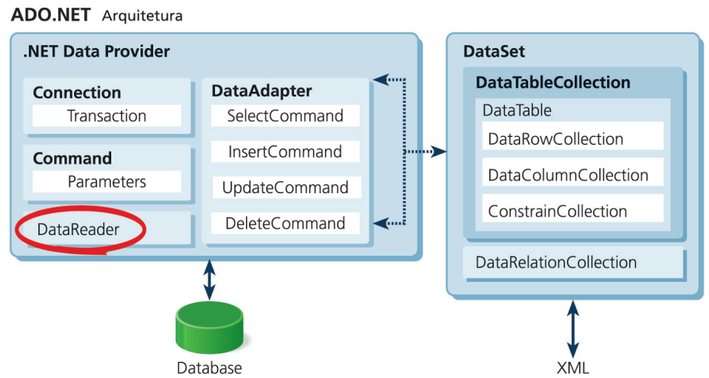


Figura 329
Estrutura geral.

SqlConnection, OdbcConnection, OleDbCommand e SqlCommand. Para termos uma visão geral dos mecanismos de conexão, observemos a figura 329, que mostra toda a estrutura.

Os principais pacotes utilizados pelo ADO.NET são:

- **System.Data**: contém as classes que representam tabelas, colunas, linhas e também a classe DataSet de todos os provedores, além das interfaces IDbCommand, IDbConnection, e IDbDataAdapter, que são usadas por todos os provedores de conexão;

- **System.Data.Commom**: define as classes para os provedores de dados DbConnection e DbDataAdapter;

- **System.Data.OleDb**: fonte de dados Ole Db usando o provedor .NET OleDb;

- **System.Data.Odbc**: fonte de dados ODBC usando o provedor .NET ODBC;

- **System.Data.SqlTypes**: dados específicos para o SQL Server.

Além disso, o ADO.NET oferece classes referenciadas:

- **Disconnected**: fornece classes capazes de armazenar dados sem a dependência da fonte de dados de determinado provedor. Por exemplo, DataTable.

- **Shared**: classes que podem ser acessadas por todos os provedores;

- **Data Providers**: classes utilizadas em diferentes fontes de dados para gerenciamento.

9.1. DataSet

O objeto recordset (ADO), que armazena somente uma coleção de tabelas, tem desvantagens em relação ao DataSet, que faz parte do System.Data. O DataSet controla uma cópia do banco de dados, representando um conjunto de informações em memória cachê que não estão conectadas com o banco de dados do sistema. Baseado em XML e independente da fonte de dados, o DataSet pode armazenar várias versões das tabelas. Apesar de trabalhar no formato desconectado, o DataSet possui mecanismos que dão suporte ao modelo conectado. Entre os métodos disponíveis, podemos destacar alguns mostrados na tabela 15.

Tabela 15
DataSet.

COLEÇÕES	DESCRIÇÃO
Tables	Uma coleção de tabelas que armazenam os dados atuais a serem manipulados

MÉTODOS	DESCRIÇÃO
AcceptChanges	Grava todas as alterações para o DataSet
Clear	Remove todas as linhas das tabelas
Clone	Faz uma cópia da estrutura, mas não copia os dados
Copy	Faz uma cópia da estrutura e dos dados
GetChanges	Retorna uma cópia do DataSet com apenas as colunas alteradas
GetXmlSchema	Retorna uma representação XML da estrutura de um DataSet
Reset	Reverte o DataSet ao seu estado original

9.2. DataReader

O DataReader permite acessar e fazer a leitura do banco de dados, percorrendo os registros de forma sequencial por meio do objeto command. Esses registros serão lidos posteriormente pelo DataReader. Diferentemente do DataSet, o DataReader não oferece acesso desconectado e não permite alterar ou atualizar a fonte de dados original. Ele possibilita apenas o acesso rápido de leitura. Entre os métodos, podemos destacar os que aparecem na tabela 16.

Tabela 16
DataReader.

MÉTODOS	DESCRIÇÃO
FieldCount	Número de colunas da linha de dados atual
IsClosed	Verifica se o objeto DataReader está fechado
Read	Avança para o próximo registro
Close	Fecha o objeto

9.3. Objetos para banco de dados

Uma das grandes vantagens do ADO.NET são os recursos oferecidos pelos objetos de manipulação de dados.

9.3.1. Objeto DataTable

O objeto DataTable pode representar uma ou mais tabelas de dados, as quais permanecem alocadas em memória. Pode ser manipulado por meio de métodos, como mostra a tabela 17.

Tabela 17
DataTable.

MÉTODOS	DESCRIÇÃO
Columns	Representa as colunas da tabela
Rows	Linhas da tabela
PrimaryKey	Chave primária
NewRow	Cria uma nova linha de dados
Copy	Faz uma cópia da estrutura e dos dados da tabela
TableName	Define o nome da tabela
Clear	Limpa dos dados da tabela

9.3.2. Objeto DataView

As operações de pesquisa, ordenação e navegação pelos dados podem ser feitas por meio do DataView, que permite a ligação da fonte de dados com a interface do usuário. Portanto, utilizamos um DataView para visualizar as informações contidas em DataTable. Uma vantagem é possuir vários DataViews para a mesma DataTable. A tabela 18 ilustra algumas das propriedades do DataView.

Tabela 18
DataView.

MÉTODOS	DESCRIÇÃO
RowFilter	Retorna a expressão usada para filtrar os dados
Item	Captura uma linha de dados específica da tabela
Sort	Ordena os dados por meio de uma coluna
Addnew	Adiciona uma nova linha
Table	Define qual DataView será visualizada
Delete	Exclui linhas de um DataView
Find	Busca uma linha de informações

9.4. Métodos de conexão

O primeiro passo para realizar a conexão é criar o objeto Connection por meio de uma string de conexão. Isso permitirá que o objeto Command receba e execute instruções SQL no formato de parâmetros. Quando o objeto Command realizar o retorno dos dados, deve-se criar um objeto DataAdapter, que preencherá um objeto DataSet ou DataTable.

9.4.1. Objeto Command

A função do Command é fazer a ligação com um banco de dados específico. Por isso, esse objeto deve conter informações necessárias para que a conexão seja estabelecida, indicando o caminho do banco, usuário, senha etc. Como foi mencionado anteriormente, cada provedor possui um objeto connection específico (figura 330).

SQL Server: Classe **SqlConnection**

OLE DB: Classe **OleDbConnection**

Figura 330
Objeto connection específico.

As principais propriedades da Classe Connection podem ser observados na tabela 19.

MÉTODOS	DESCRIÇÃO
ConnectionString	Contém a string de conexão
DataBase	Retorna o nome do banco de dados
DataSource	Retorna o nome da instância do banco de dados
State	Retorna o estado atual de conexão: Broken, Closed, Connecting, Executing, Fetching e Open

Tabela 19
Connection String.

9.4.2. Exemplo genérico de conexão

O código mostrado na figura 332, escrito em C#, representa os passos necessários para a conexão de uma base de dados SQL Server. A partir desse ponto, as operações com o banco de dados já podem ser realizadas.

```
¸ Incluindo os namespace
using System.Data
using System.Data.SqlClient

¸ Montando a string de conexão
  // definindo isoladamente cada componente da conexão
string servidor = "localhost"
string username = "usuario"
string senha = "db2009conect"
string banco = "papelaria"
// contruindo a ConnectionString
string ConnectionString = "Data Source=" + servidor + ";"
ConnectionString += "User ID=" + username + ";"
ConnectionString += "Password=" + senha + ";"
ConnectionString += "Initial Catalog=" + banco;
```

Figura 332
Código representando passos para conexão.

```
¸ Criando uma instância do objeto Connection
SqlConnection SQLConnection = new SqlConnection();

¸ Realizando a conexão
   SQLConnection.ConnectionString = ConnectionString;
SQLConnection.Open();
```

Outra forma de fazer essa conexão está representada no código mostrado na figura 333.

Figura 333
Outra forma de realizar conexão.

```
SqlConnection conexao = new SqlConnection("Data
Source=(localhost);Initial Catalog=papelaria; User ID=usuario;Passwor
d=db2009conect");
```

9.4.2.1. Implementando a leitura de dados

Ainda seguindo o exemplo anterior, vamos elaborar uma estrutura mais completa de conexão e implementar os códigos para leitura dos dados (figura 334).

Figura 334
Estrutura mais completa de conexão.

```
using System;
using System.Data;
using System.Data.SqlClient;

class ExemploConexao
{

 static void Main()
 {
   // criando a linha de conexão
   SqlConnection conexao = new SqlConnection("Data
Source=(localhost);Initial Catalog=papelaria; User ID=usuario;Passwor
d=db2009conect");
   // definindo um DataReader Nullo
   SqlDataReader drExemplo = null;
   // Abre o banco de dados
   conexao.Open();
   // Cria a linha de comando
   SqlCommand comando = new SqlCommand("select * from Cliente",
conexao);
   // Executa a leitura dos dados
   drExemplo = comando.ExecuteReader();
   // faz a leitura dos dados
   while (drExemplo.Read())
   {
    // imprime o primeiro campo da tabela
    Console.WriteLine(drExemplo[0]);
   }
   // fecha o DataReader
   drExemplo.Close();
   // fecha a conexão com o banco
   conexao.Close();
 }
}
```

Para obter mais colunas, podemos utilizar o comando da forma como é sugerida na figura 335.

Figura 335
Como obter mais colunas.

```
Console.WriteLine(drExemplo[0]);
Console.WriteLine(drExemplo[1]);
Console.WriteLine(drExemplo[2]);
```

Ou indicar o nome da coluna da qual se pretende obter a informação (figura 336).

Figura 336
Indicando o nome da coluna.

```
Console.WriteLine(drExemplo["codigo"]);
Console.WriteLine(drExemplo["nome"]);
Console.WriteLine(drExemplo["usurname"]);
```

Ao usar os métodos ligados à base de dados, é importante verificar se a conexão foi efetivamente aberta, se não ocorreu nenhum erro e fechá-la no final. O procedimento evita problemas de desempenho. Portanto, é recomendável realizar os tratamentos de erro (Try).

9.4.3. Conexão com VB.NET

Seguindo o mesmo princípio do item anterior, vamos fazer uma conexão com uma base de dados Access, porém, usando o VB.NET como plataforma de conexão.

9.4.3.1. Base de dados

Por meio do Access 2007, criamos a base de dados mostrada na figura 337 e incluímos alguns dados fictícios, para os testes iniciais. O nome do banco de dados é "Usuarios" e o da tabela, "operadores". A chave primária é o campo "cod_ID" de numeração automática. O objetivo desse banco é cadastrar o nome de login dos usuários.

Figura 337
Descrição da tabela e conteúdo.

operadores

Nome do campo	Tipo de dados
cod_ID	Numeração Automática
login	Texto
nome	Texto

operadores

cod_ID	login	nome	Adicionar Novo
2	wldy	Waldycleidson Jr.	
3	robim	Rosberwaldo Murim	
(Novo)			

9.4.3.2. Criando o Form

Em uma Solution do tipo Windows Form Application, foi criado para o Visual Basic o layout que pode ser visto na figura 338.

O layout é muito simples. Incluímos um Button com o nome de btnListar e uma ListBox chamada de lstUsers. Veja, na figura 339, como fica a descrição dos componentes.

Figura 338
Layout do Form.

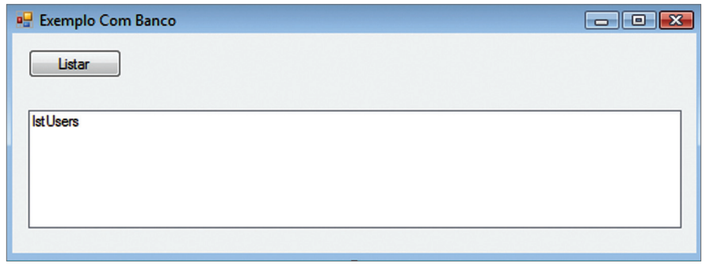
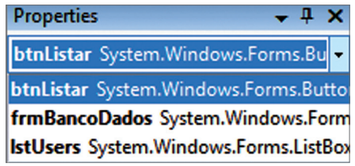


Figura 339
Nome dos componentes.



9.4.3.3. Inserindo o código

O código inserido no botão Listar deverá fazer a leitura de todos os dados contidos na tabela e inseri-los numa ListBox. Assim, haverá um registro diferente em cada linha do ListBox (figura 340).

```
Public Class frmBancoDados
Private Sub btnListar_Click(ByVal sender As System.Object, ByVal e As
System.EventArgs) Handles btnListar.Click
  Dim conexao As New OleDbConnection()
  conexao.ConnectionString = "Provider=Microsoft.Jet.OLEDB.4.0;
          Data Source=D:\Usuarios.mdb"
  Dim comando As OleDbCommand = conexao.CreateCommand
  comando.CommandText = "Select * from operadores"
  conexao.Open()
  Dim leitor As OleDbDataReader = comando.ExecuteReader()
  Try
    Dim linha As String = "Usuário:"
    While leitor.Read()
      Dim reg As Integer
      For reg = 0 To leitor.FieldCount - 1
       linha = linha & " - " & leitor.Item(reg)
      Next
      lstUsers.Items.Add(linha)
      linha = "Usuário:"
    End While
    leitor.Close()
    conexao.Close()
  Catch erro As Exception
    MsgBox("Não foi possível realizar a operação", "Erro")
  End Try
 End Sub
End Class
```

Figura 340
Registros diferentes para cada linha.

A estrutura de conexão que usa o Visual Basic, com banco de Dados Access, é semelhante ao código desenvolvido no item anterior, que adota C# e SQL Server.

9.4.3.4. Utilizando uma DataTable

Para visualizar os dados, vamos melhorar um pouco mais a nossa estrutura. Eliminamos o ListBox do exemplo anterior e colocamos um DataGrid em seu lugar, como mostra a figura 341.

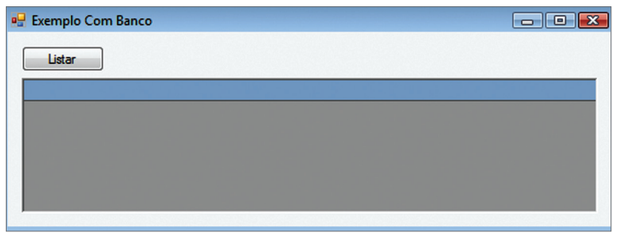


Figura 341
DataGrid.

No exemplo a seguir, vamos indicar passo a passo a construção do DataTable para visualização em um DataGrid.

• Definindo um DataTable (dtMinhaTabela), com o nome de MinhaTabela.

```
Dim dtMinhaTabela As DataTable = New DataTable("MinhaTabela")
```

• Definidos dois objetos para controlar coluna e linha do DataTable.

```
Dim dtColuna As DataColumn

Dim dtLinha As DataRow
```

• Após a definição do objeto para coluna, devemos construir a coluna, ou seja, indicar a sua nomenclatura e o tipo de dado que ela conterá.

```
dtColuna = New DataColumn()

dtColuna.DataType = System.Type.GetType("System.String")

dtColuna.ColumnName = "Login"

dtMinhaTabela.Columns.Add(dtColuna)

dtColuna = New DataColumn()

dtColuna.DataType = System.Type.GetType("System.String")

dtColuna.ColumnName = "Nome"

dtMinhaTabela.Columns.Add(dtColuna)
```

• Se utilizarmos o método Read(), serão carregadas as informações, assim como ocorreu no exemplo anterior.

```
While leitor.Read()

  dtLinha = dtMinhaTabela.NewRow

  dtLinha("Login") = leitor.Item(0)

  dtLinha("Nome") = leitor.Item(1)

  dtMinhaTabela.Rows.Add(dtLinha)

End While
```

• Com o DataTable carregado, vamos vinculá-lo a um DataSet.

```
Dim dtMinhaDataSet As DataSet = New DataSet()

dtMinhaDataSet.Tables.Add(dtMinhaTabela)
```

• Finalmente, carregamos o DataSet para dentro do DataGrid.

```
dtMinhaTabela.SetDataBinding(dtMinhaDataSet, "MinhaTabela")
```

Os comandos anteriores representam somente a criação do DataTable. O código completo ficará como se apresenta na figura 342.

Figura 342
O código completo.

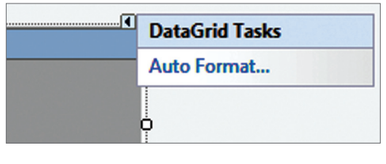
```
Imports System.Data
Imports System.Data.SqlClient
Imports System.Data.OleDb

Public Class frmBancoDados

  Private Sub btnListar_Click(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles btnListar.Click
    Dim conexao As New OleDbConnection()
    conexao.ConnectionString = "Provider=Microsoft.Jet.
OLEDB.4.0;Data Source=D:\Usuarios.mdb"

    Dim comando As OleDbCommand = conexao.CreateCommand
    comando.CommandText = "Select * from operadores"
    conexao.Open()

    Dim dtMinhaTabela As DataTable = New DataTable("MinhaTabela")
    Dim dtColuna As DataColumn
    Dim dtLinha As DataRow

    ' montando as colunas
    dtColuna = New DataColumn()
    dtColuna.DataType = System.Type.GetType("System.String")
    dtColuna.ColumnName = "Login"
    dtMinhaTabela.Columns.Add(dtColuna)

    dtColuna = New DataColumn()
    dtColuna.DataType = System.Type.GetType("System.String")
    dtColuna.ColumnName = "Nome"
    dtMinhaTabela.Columns.Add(dtColuna)

    ' inserindo os dados
    Dim leitor As OleDbDataReader = comando.ExecuteReader()
    While leitor.Read()
      dtLinha = dtMinhaTabela.NewRow
      dtLinha("Login") = leitor.Item(1)
      dtLinha("Nome") = leitor.Item(2)
      dtMinhaTabela.Rows.Add(dtLinha)
    End While
```

```
    'inclui a tabela no dataset
    Dim dtMinhaDataSet As DataSet = New DataSet()
    dtMinhaDataSet.Tables.Add(dtMinhaTabela)

    'vincula o dataset1 ao datagrid1
    dtgMinhaTabela.SetDataBinding(dtMinhaDataSet, "MinhaTabela")
  End Sub

End Class
```

9.4.3.4.1. Visual do DataGrid

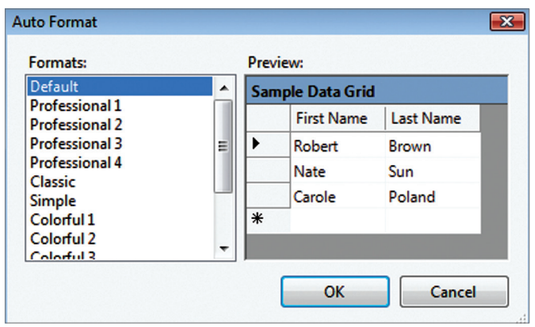
Para melhorar o visual do DataGrid, clique na seta existente no canto superior do DataGrid (figura 343) e escolha a opção AutoFormat.

Figura 343
Menu DataGrid.



Na opção Formats, teremos vários tipos de layout. Escolheremos um deles e confirmaremos com o botão OK (figura 344).

Figura 344
Formatando Layout – DataGrid.



9.4.3.5. Travando colunas

As colunas do DataGrid estão disponíveis para que o usuário possa realizar modificações diretamente na janela de dados. Se não quiser que isso seja possível, inclua o método ReadOnly como True, como mostra o código ilustrado na figura 345.

```
dtColuna = New DataColumn()
dtColuna.DataType = System.Type.GetType("System.String")
dtColuna.ColumnName = "Login"
dtColuna.ReadOnly = True
dtMinhaTabela.Columns.Add(dtColuna)
```

Figura 345
Inclusão do método ReadOnly como True.

Isso impede que o usuário faça novas inclusões. Como as colunas são montadas individualmente, podemos determinar qual vai ser o procedimento adotado em cada uma, assim como foi feito com a coluna login, deixando a alteração de dados ativa ou não. Podemos, ainda, utilizar o método Unique para informar se o valor da coluna é único ou não (figura 346). Isso significa que o método não permite que o valor registrado em uma coluna seja inserido novamente na mesma coluna em outro registro.

```
dtColuna = New DataColumn()
dtColuna.DataType = System.Type.GetType("System.String")
dtColuna.ColumnName = "Nome"
dtColuna.ReadOnly = False
dtColuna.Unique = True
dtMinhaTabela.Columns.Add(dtColuna)
```

Figura 346
Utilização do método Unique.

9.4.4. Utilizando um DataView

O DataView permite estabelecer uma ligação com a interface de usuário por meio do DataBinding, no qual podemos realizar operações como pesquisa, navegação, filtro, etc. O DataView retorna os dados contidos em um DataTable. É possível que haja vários DataView, que, aliás, não podem ser considerados tabelas. O exemplo da figura 347 mostra dois Datagrid e dois botões. No primeiro DataGrid, é carregado o conteúdo da tabela; no segundo, será um DataView, ordenado por nome.

A conexão com o banco de dados será feita de maneira diferente, para que seja possível avaliar outra forma de conexão com o DataSet. Nesse caso, todas as variáveis relacionadas à conexão com o banco de dados serão definidas dentro da classe, para que tenham visibilidade global (figura 348).

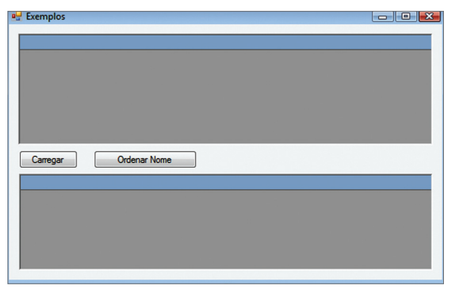


Figura 347
Construção de um DataView.

Figura 348
Variáveis definidas dentro da classe.

```
Dim conexao As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=D:\Usuarios.mdb"
Dim comando As String= "Select * from operadores"
Dim adpTabela As New OleDbDataAdapter(comando, conexao)
Dim dsTabela1 As New DataSet()
Dim dsTabela2 As New DataSet()
```

Assim, teremos: **conexão**: variável que possui as informações de provedor, caminho e nome do banco; **comando**: armazena a string referente à instrução SQL a ser executada inicialmente; **adptabela**: cria um objeto Adapter usando as variáveis comando e conexão; **dstabela1** e **dstabela2**: representa os objetos DataSet(), um para cada DataGrid do formulário.

O botão referente à opção "Carregar" deverá conter o código mostrado na figura 349.

Figura 349
Código do botão da opção Carregar.

```
dtgLista.CaptionText = "Listagem de Operadores"
adpTabela.Fill(dsTabela1, "operadores")
dtgLista.DataSource = dsTabela1
dtgLista.DataMember = "operadores"
```

É importante observar os conceitos abaixo:

- **dtgLista.CaptionText**: atribui o nome no DataGrid (dtgLista).
- **adpTabela.Fill:** preenche o objeto Adpter (adpTabela.Fill).
- **dtgLista.DataSource:** atribui o DataSet (dstabela1) no DataGrid (dtgLista).
- **dtgLista.DataMember:** associa a tabela ao DataGrid (dtgLista).

No segundo botão, referente à ordenação dos dados via campo "nome", observamos o que ilustra a figura 350.

Figura 350
Botão da ordenação dos dados.

```
dtgOrdenado.CaptionText = "Listagem de Operadores"
adpTabela.Fill(dsTabela2, "operadores")
Dim dvTabela As New DataView(dsTabela2.Tables("operadores"))
dvTabela.Sort = "nome"
dtgOrdenado.DataSource = dvTabela
```

A diferença do DataView em relação ao DataSet (dsTabela2) está na ordenação (.Sort) por meio do campo "nome". O código completo ficará como ilustra a figura 351. Como resultado, surgirá o layout mostrado na figura 352.

Figura 351
O código completo.

```
Imports System.Data.OleDb
Public Class frmExemplos
  Dim conexao As String = "Provider=Microsoft.Jet.OLEDB.4.0;Data
Source=D:\Usuarios.mdb"
  Dim comando As String = "Select * from operadores"
  Dim adpTabela As New OleDbDataAdapter(comando, conexao)
  Dim dsTabela1 As New DataSet()
  Dim dsTabela2 As New DataSet()

  Private Sub btnCarregar_Click(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles btnCarregar.Click
    dtgLista.CaptionText = "Listagem de Operadores"
    adpTabela.Fill(dsTabela1, "operadores")
    dtgLista.DataSource = dsTabela1
    dtgLista.DataMember = "operadores"
  End Sub

  Private Sub btnOrdeNome_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnOrdeNome.Click
    dtgOrdenado.CaptionText = "Listagem de Operadores"
    adpTabela.Fill(dsTabela2, "operadores")
    Dim dvTabela As New DataView(dsTabela2.Tables("operadores"))
    dvTabela.Sort = "nome"
    dtgOrdenado.DataSource = dvTabela
  End Sub
End Class
```

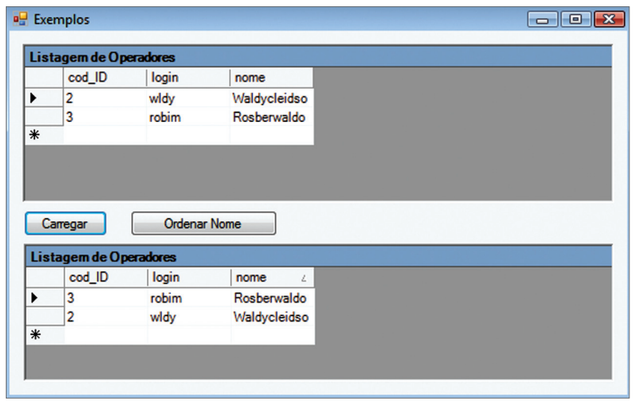


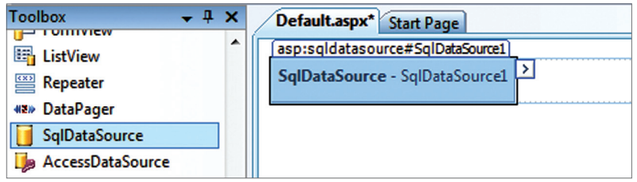
Figura 352
Utilização do DataView.

9.4.5. Conectando com ASP.NET

Podemos realizar a inclusão de DataGrid ou do GridView, utilizando Wizard (passo-a-passo) fornecedido pelo componente. Nos exemplos anteriores, realizamos a conexão por meio do Visual Basic e do C#. Agora, faremos a conexão com

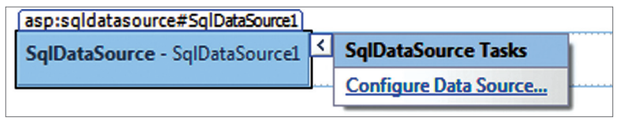
o banco via ASP.NET. O primeiro passo é criar um projeto tipo Visual Basic e um template Web Aplication. No modo Design, será adicionado o componente SqlDataSource (figura 353).

Figura 353
Componente SqlDataSource.



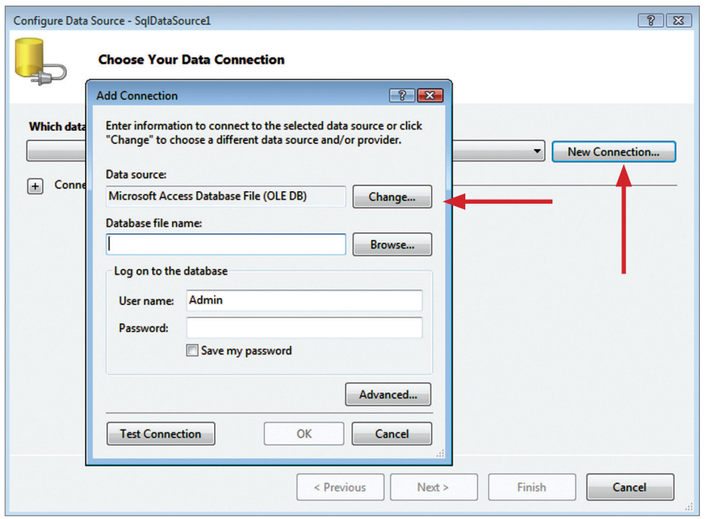
No guia do componente, escolha a opção "configure", conforme a figura 354.

Figura 354
Configuração do SqlDataSource.



O próximo passo será criar a nova conexão usando o botão New Connection. Aparecerá uma segunda janela, na qual poderemos indicar o nome e o local onde se encontra a base de dados (figura 355).

Figura 355
Localização do banco de dados.



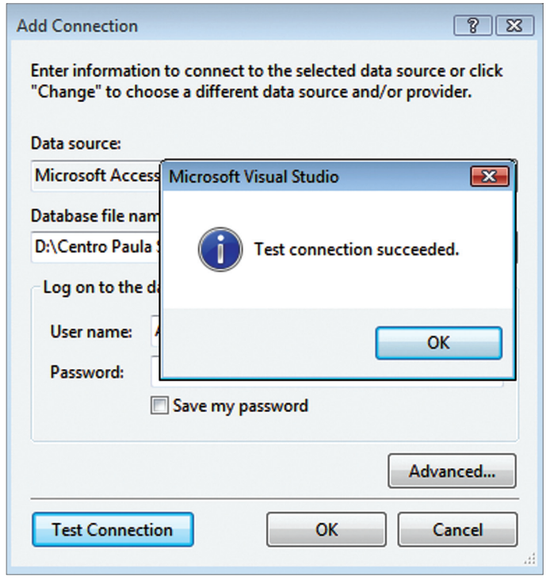


Figura 356
Testando a conexão.

Nesse ponto, será possível realizar um teste de conexão com o banco de dados, usando o botão Test Connection. Se estiver tudo OK, aparecerá uma mensagem de confirmação (figura 356).

Após o teste de conexão, finalize a tela clicando em OK. Retorne à janela anterior. Podemos disponibilizar a Connection String, como mostra a figura 357.

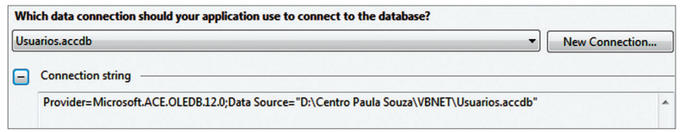
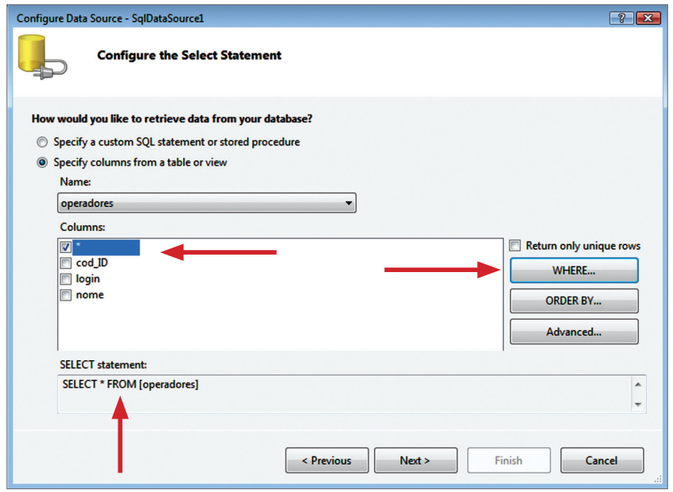


Figura 357
Visualizando a String de conexão.

Na continuação do processo, será confirmada a conexão (figura 358).

Figura 358
Confirmando a ConnectionString.

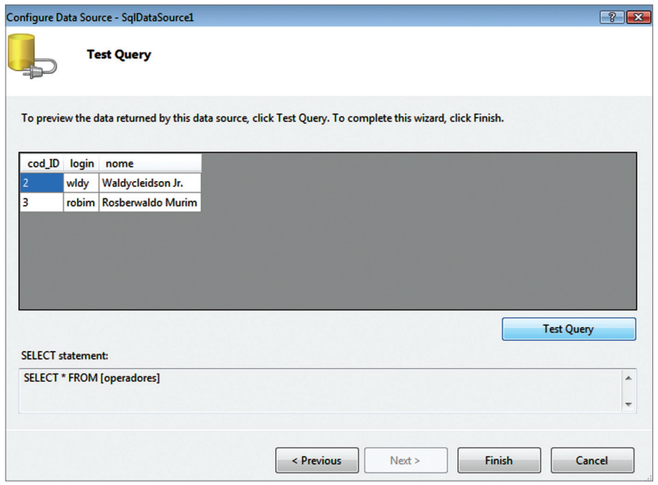
Figura 359
Configurando o SQL.



Uma nova janela será aberta para a criação da instrução SQL inicial. Observando a janela de configuração podemos verificar o nome da tabela, quais os campos a serem visualizados e a configuração das cláusulas Where e Order By. Na parte inferior da janela, nota-se a montagem da linha SQL (figura 359).

Finalizado o processo, é possível realizar o teste da Query com o banco de dados, conforme a figura 360.

Figura 360
Teste do SQL (Query).



Agora, podemos ver (figura 361) o código que foi implementado automaticamente.

```
<asp:SqlDataSource ID="SqlDataSource1" runat="server"
ConnectionString="<%$ ConnectionStrings:ConnectionString %>"
ProviderName="<%$ ConnectionStrings:ConnectionString.Provider-
Name %>"
SelectCommand="SELECT * FROM [operadores]"></
asp:SqlDataSource>
```

Figura 361
O código implementado automaticamente.

O próximo passo será a inclusão de um GridView, como mostra a figura 362.

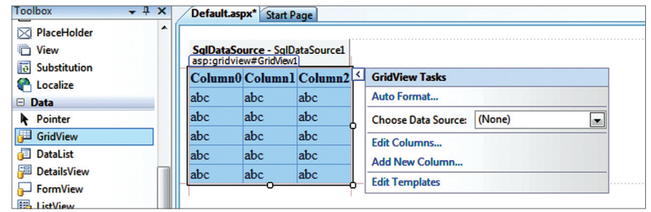


Figura 362
GridGrid View.

A partir da opção Choose Data Source, podemos apontar o SqlDataSource (figura 363).

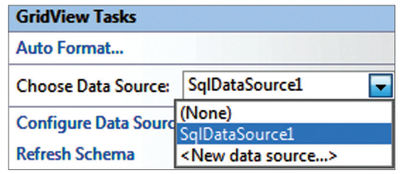


Figura 363
Selecionando o DataSource.

Finalizada essa operação, o código seguinte será o que mostra a figura 364.

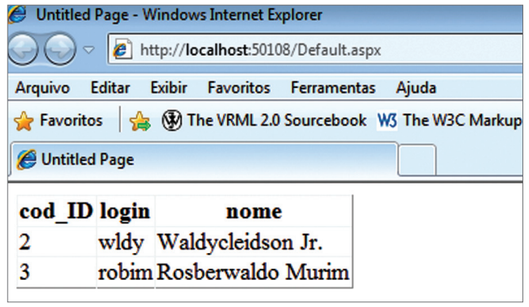
```
<asp:GridView ID="GridView1" runat="server"
AutoGenerateColumns="False"
DataKeyNames="cod_ID" DataSourceID="SqlDataSource1">
<Columns>
<asp:BoundField DataField="cod_ID" HeaderText="cod_ID"
InsertVisible="False"
```

Figura 364
Concluída a operação de selecionar o DataSource.

```
ReadOnly=”True” SortExpression=”cod_ID” />
<asp:BoundField DataField=”login” HeaderText=”login”
SortExpression=”login” />
<asp:BoundField DataField=”nome” HeaderText=”nome”
SortExpression=”nome” />
</Columns>
</asp:GridView>
```

O layout ainda não foi formatado, mas já é possível verificar o resultado pelo navegador (figura 365). Nos campos destinados aos nomes aparecem os títulos da coluna. E todos os dados são disponibilizados conforme composição da instrução SQL.

Figura 365
Visualização da Tabela.



Considerações finais

O objetivo dos autores deste livro é permitir que alunos e profissionais da área de informática possam ingressar no ramo da programação e desenvolvimento de software. Por isso, foram abordados aspectos de lógica de programação e programação orientada a objetos e apresentadas as estruturas básicas de programação como HTML, Java, C#, VB.NET e ASP.NET, que podem ser aplicadas nas plataformas desktop e Web. O livro é a ponta de um “iceberg” no que diz respeito a comandos e recursos de programação. Compete ao aluno a busca incessante pelo aprimoramento das técnicas de programação. Portanto, seguem abaixo algumas dicas.

- Trabalhar a lógica de programação é essencial. Por isso, a prática deverá ser contínua.
- Pesquisar sempre em livros especializados, revistas técnicas ou até mesmo na internet novas tecnologias e recursos que a linguagem pode oferecer e procurar aplicá-las.
- Manter-se atualizado, pois as linguagens de programação estão sempre se renovando e novas versões são lançadas a todo o momento.

Aos leitores, alunos ou profissionais da área, boa sorte e sucesso na área escolhida.

Referências bibliográficas

FORBELLONE, André. *Lógica de Programação* - A Construção de Algoritmos e Estruturas de Dados. 3ª edição. São Paulo: Ed. Makron Books, 2005.

LEISERSON, Charles E., STEIN, Clifford, RIVEST, Ronald L. e CORMEN, Thomas H. *Algoritmos: Teoria e Prática*. Rio de Janeiro: Ed. Campus, 2002.

RAY, Eric T. *Aprendendo XML*. Rio de Janeiro: Ed. Campus, 2001.

FURGERI, Sérgio. *Ensino Didático da Linguagem XML*. São Paulo: Érica, 2001.

HOLZNER, Steven. *Desvendando XML*. Rio de Janeiro: Campus, 2001.

MARCONDES, Christian Alfim. *Programando em HTML 4.0*. 7ª. edição. São Paulo: Érica, 2002