WORKSHOPS

INTRODUÇÃO AO .NET

LABORATÓRIO

Acesso a dados usando o ADO.NET (C#)

Data de revisão: Fevereiro de 2004

Índice

LAB 05: Acesso a dados usando o ADO.NET – C#

Este Lab fornece uma visão geral do ADO.NET. O primeiro exercício provê uma breve introdução ao ADO.NET e exercícios subseqüentes que apresentam em detalhes como realizar diferentes operações em bancos de dados como inserir, atualizar e excluir registros, executado tais operações, inclusive, de forma transacional. O Lab discute também a forma de trabalho com o controle DataGridView e a criação programática de objetos DataSet, DataAdapter, DataReader e Command.

Preparação do Lab

A solução trabalhada e os arquivos associados e este Lab estão localizados em **%SystemDrive%\CentrosXML\Workshops\FirstLook\Lab05\Starter Solution**..

Objetivos do Lab

Tempo estimado para completar este Lab: 60 minutos

Após completar este Lab você saberá sobre:

- Entendendo o ADO.NET
- Trabalhando com os objetos DataSet e SqlDataAdapter
- Trabalhando com os objetos SqlDataReader e SqlCommand
- Inserindo, Atualizando e Excluindo registros usando um DataSet
- Carregando um Recordset nativo ADO em um DataSet

Exercício1 – Introdução ao ADO.NET

A biblioteca ADO.NET provê acesso consistente a fontes de dados como o Microsoft SQL Server, Oracle, fontes de dados que utilizam OLE DB ou ODBC para se conectar e arquivos XML. Aplicações voltadas a dados podem usar ADO.NET para conectar com fontes de dados, capturar e manipular os dados capturados.

Tarefa 1 – Visão Geral dos Provedores de Dados (Data Providers) ADO.NET

Os Data Providers (Provedores de Dados) constituem a seção núcleo da arquitetura ADO.NET, permitindo comunicação entre as aplicações e fontes de dados. Um Data Provider permite que você conecte com uma fonte de dados, capture e manipule dados e,por fim, atualize a fonte de dados. Os Data Providers no .NET Framework servem, portanto, como uma ponte entre uma aplicação e a fonte de dados.

Os ADO.NET Data Providers foram projetados para serem leves, constituindo uma camada mínima entre a fonte de dados e seu código, aumentando a performance sem sacrificar a funcionalidade.

Os quatro principais objetos que compõem um Data Provider ADO.NET são:

Connection – Estabelece uma conexão para uma fonte de dados específica

Command – Executa um comando na fonte de dados

DataReader – Provê acesso rápido, forward-only, read-only a dados

DataAdapter – Preenche um DataSet e resolve atualizações com a fonte de dados

O .NET Framework apresenta 4 Data Providers já embutidos.

- Data Provider SQL Server

O Data Provider para SQL Server usa o protocolo SQL específico de servidor para transferência de dados (Tabular Data Stream), permitindo, assim, a comunicação com o SQL Server. Este Data Provider é otimizado para acessar o SQL Server diretamente sem adicionar uma camada de conectividade OLE DB ou ODBC. Este provedor é recomendado para aplicações utilizando o Microsoft SQL Server 7.0 ou superior.

Para usar este Data Provider inclua a seguinte importação no seu código:

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

- Data Provider OLEDB

O Data Provider fornecido pelo .NET Framework para OLEDB usa OLEDB nativo através de uma camada de interoperabilidade COM para habilitar acesso a dados. Este Data Provider suporta tanto transação manual como automática. Este provedor é recomendado para aplicações de camada intermediária usando Microsoft SQL Server 6.5 ou anterior, ou qualquer outro provedor compatível com OLE DB.

Para usar este Data Provider inclua a seguinte importação no seu código:

```
using System.Data.OleDb;
```

- Data Provider ODBC

O Data Provider fornecido pelo .NET Framework para ODBC utiliza um driver manager nativo para ODBC para habilitar acesso a dados a fonte de dados ODBC. Este Data Provider suporta tanto transação local quanto distribuída. Este provedor é recomendado para aplicações usando fontes de dados ODBC.

Para usar este Data Provider inclua o a seguinte importação no seu código:

```
using System.Data.Odbc;
```

- Data Provider Oracle

O Data Provider fornecido pelo .NET Framework para Oracle habilita acesso a dados para fontes de dados para o Oracle usando o software de conectividade cliente para Oracle. Este Data Provider suporta software cliente para Oracle nas versões 8.1.7 e superior. Este provedor é recomendado para aplicações utilizando fontes de dados Oracle.

Nota: Para usar este provedor você precisará importar o namespace usando o comando de menu **Project | Add Reference…** e selecione o assembly **System.Data.OracleClient.dll**.

Para usar este Data Provider inclua a seguinte importação no seu código:

```
using System.Data.OracleClient;
```

Tarefa 2 – String de Conexão (Connection String)

A string de conexão no ADO.NET é similar a uma string de conexão OLEDB. As aplicações precisam especificar uma string de conexão para conectar a uma fonte de dados. A propriedade ConnectionString do objeto Connection pode ser configurada somente quando a conexão estiver fechada. A string de conexão é uma série de pares chave/valor (key/value) delimitados por ponto-e-vírgula. Para conectar a um SQL Server rodando na máquina local, especifique "(local)" para o servidor. Os seguintes elementos constituem a propriedade ConnectionString para um objeto SqlConnection:

Data Source-or- Server	O nome ou endereço de rede da instância do SQL server para o qual conectar
Integrated Security -or- Trusted_Connection	Quando false, User ID e Password são especificados na conexão. Quando true, a conta atual do Windows é utilizada para autenticação
User ID	A conta de login do SQL Server
Password -or- Pwd	A senha para o SQL Server logar
Initial Catalog -or- Database	O nome do banco de dados

Exercício2 – Trabalhando com os objetos SqlCommand e SqlDataReader

Neste exercício você irá executar tarefas comuns de bancos de dados incluindo conectar a um banco de dados SQL server utilizando a classe SqlConnection e inserir, atualizar, e excluir dados usando a classe SqlCommand. Você irá também utilizar a classe SqlDataReader para capturar um lote de dados read-only, forward-only de um banco de dados. O uso do DataReader pode aumentar a performance da aplicação e reduzir a sobrecarga de sistema visto que somente uma linha por vez fica armazenada em memória. Finalmente você entenderá como realizar transações de banco de dados.

Tarefa 1 – Armazenando string de conexão em um arquivo de configuração de aplicação

É muito comum armazenar uma string de conexão de banco de dados em um único local para simples manutenção da aplicação. Nesta tarefa você irá criar um projeto e então criará um arquivo de configuração de aplicação, que armazenará uma string de conexão.

- Clique em **Start**, selecione **All Programs | Microsoft Visual C# Express**
- Selecione o comando de menu **File | New Project**

- Na lista **Templates** selecione **Windows Application**
- Configure a propriedade **Name** para “Lab05-CS”
- Clique em **OK**
- Para adicionar um arquivo de configuração de aplicativo, selecione o comando de menu **Project | Add New Item…**
- Ou clique com o botão direito no projeto **Lab05-CS** e acesse **Add | New Item.**
- Na lista **Templates** selecione **Application Configuration File** e clique em **Add.**
- Escreva o seguinte conteúdo XML em destaque dentro da tag <configuration> no arquivo App.config:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
<appSettings>
   <add key="ConnectionString"
     value="server=NomeDoComputador\SQLEXPRESS;Trusted_Connection=yes;
database=AdventureWorks"/>
</appSettings>
</configuration>
```

Tarefa 2 – Usando o SqlCommand para inserir, atualizar, e excluir dados

O objeto SqlCommand representa um comando Transact-SQL ou uma stored procedure, que será executada contra um banco de dados SQL Server. Nesta tarefa você criará uma conexão com o SQL Server e então vai inserir, atualizar e excluir dados utilizando o objeto SqlCommand.

- Para abrir **Solution Explorer,** selecione o comando de menu **View | Solution Explorer**
- Dê um clique com o botão direito do mouse em Form1.cs na Solution Explorer e selecione **View Code**
- Adicione o seguinte código em destaque para o código no topo do código:

```
...
...
using System.Windows.Forms;
using System.Data;
using System.Data.SqlClient;
```

- Declare uma variável **SqlConnection** na seção de declaração geral na classe Form1

```
public class Form1 : System.Windows.Forms.Form
{
   SqlConnection mySqlConnection;
...
...
```

- Para acessar a visualização de projeto do **Form1**, selecione o comando de menu **View | Designer**

- Dê um duplo-clique na visualização de projeto do **Form1,** o Visual C# irá navegar para o arquivo fonte do **Form1** e irá criar um manipulador de evento (**Form1_Load**) para o evento **Load** do formulário.

- Coloque um valor para a variável mySqlConnection no manipulador de evento **Form1_Load**:

```
private void Form1_Load(object sender, System.EventArgs e)
{
   mySqlConnection = new SqlConnection (
System.Configuration.ConfigurationSettings.AppSettings.Get("ConnectionString"));
}
```

- A linha de código acima, lê o valor da chave “ConnectionString” do arquivo de configuração de aplicação (App.Config)

- Para acessar a visualização de projeto do **Form1**, selecione o comando de menu **View | Designer**

- Para abrir a **Toolbox**, selecione o comando de menu **View | Toolbox**

- Arraste um **Button** da toolbox e solte o mesmo no formulário

- Dê um clique com o botão direito do mouse no button e selecione o comando de menu **Properties**

- Configure as seguintes propriedades para o controle usando a janela **Properties**:

  Name - “btnInsert”

  Text - “Insert Command”

- Aumente a largura do controle button para caber o texto

- Arraste outro controle **Button** da toolbox e solte o mesmo no formulário

- Dê um clique com o botão direito do mouse no button e selecione o comando de menu **Properties**

- Configure as seguintes propriedades para o controle usando a janela **Properties**:

  Name - “btnUpdate”

  Text - “Update Command”

- Aumente a largura do controle button para caber o texto

- Arraste outro controle **Button** da toolbox e solte o mesmo no formulário

- Dê um clique com o botão direito do mouse no button e selecione o comando de menu **Properties**

- Configure as seguintes propriedades para o controle usando a janela **Properties**:

  Name - “btnDelete”

Text - “Delete Command”

- Aumente a largura do controle button para caber o texto
- Dê um duplo-clique no controle button **Insert Command**. O Visual C# Express irá navegar para a visualização de código do **Form1** (Form1.cs) e criará um manipulador de evento para o evento **Click** do button **Insert Command**
- Adicione o seguinte código no método **btnInsert_Click** para inserir dados sobre um novo cliente usando o objeto SqlCommand:

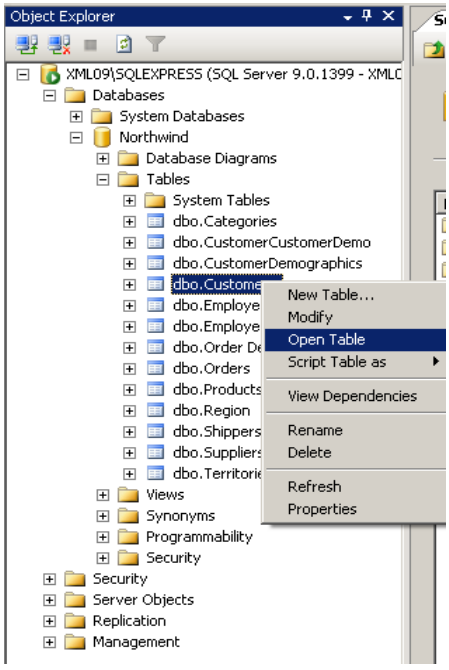
```
private void btnInsert_Click(object sender, EventArgs e)
{
   SqlCommand mySqlCommand = new SqlCommand();
   // Open the connection
   mySqlConnection.Open();

   //Assign the connection property.
   mySqlCommand.Connection=mySqlConnection;
   try
   {
         // Insert new record for XYZ customer
         mySqlCommand.CommandText = "INSERT INTO Production.Location (Name,
CostRate) Values ( 'XYZ', '55,12')";

         mySqlCommand.ExecuteNonQuery();
         MessageBox.Show("New Location data recorded!");
   }
   catch(Exception ex)
   {
         MessageBox.Show("Error occurred, customer data could not be recorded: "
+ ex.Message);
   }
   finally
   {
         // Close the connection if it is open
         if (mySqlConnection.State==ConnectionState.Open)
         {
               mySqlConnection.Close();
         }
   }
}
```

- A propriedade **State** do objeto SqlConnection indica o estado da conexão, se está aberta ou fechada. Esta propriedade pode ser usada para abrir a conexão se estiver fechada para minimizar os recursos do servidor consumido pela aplicação
- O método **ExecuteNonQuery** do objeto SqlCommand é usado quando a query está sendo executada, Insert neste caso, não é esperado retorno de resultado
- Para compilar e rodar sua aplicação selecione o comando de menu **Debug | Start Without Debugging** ou pressione **CTRL+F5**
- Clique no button **Insert Command** para realizar a operarão de inserção

- Uma caixa de mensagem aparecerá dizendo “New Location data recorded!”
- Clique em **OK** para fechar a caixa de mensagem
- Feche o **Form1**
- Para verificar se o dado inserido foi de fato inserido com sucesso
    - Acesse **Start | All Programs | Microsoft SQL Server | SQL Server Management Studio.**
    - Em **Server Name** coloque **NomeDoComputador\NomeDaInstancia.**
    - Em **Authentication** coloque **Windows Authentication.**
    - Aperte o botão **Connect.**
    - Acesse **View | Object Explorer.**
    - Expanda o nó **Databases.**
    - Expanda o nó **AdventureWorks** e o nó **Tables.**
    - Clique com o botão direito em **Production.Location** e acesse **Open Table**



    - Verifique no final da tabela que um novo cliente com o Nome de ‘XYZ’ foi armazenado
    - Volte para o **Visual C# Express**
- Para abrir **Solution Explorer,** selecione o comando de menu **View | Solution Explorer**
- De um duplo clique em **Form1.cs** na **Solution Explorer**.

- Dê um duplo-clique no controle button **Update Command**. O Visual C# Express irá navegar para a visualização de código do **Form1** (Form1.cs) e irá criar um manipulador de eventos para o evento **Click** do button **Update Command**

- Adicione o código no método **btnUpdate_Click** para atualizar o dado inserido usando o objeto SqlCommand:

```
private void btnUpdate_Click(object sender, EventArgs e)
{
   SqlCommand mySqlCommand = new SqlCommand();
   // Open the connection
   mySqlConnection.Open();

   //Assign the connection property.
   mySqlCommand.Connection=mySqlConnection;
   try
   {
         // Update XYZ customer
         mySqlCommand.CommandText = "UPDATE Production.Location SET CostRate='10.00'
WHERE Name='XYZ'";
         mySqlCommand.ExecuteNonQuery();
         MessageBox.Show("Location Data successfully updated.");
   }

   catch(Exception ex)
   {
         MessageBox.Show("Error occurred, location data could not be updated: "
+ ex.Message);
   }
   finally
   {
         // Close the connection if it is open
         if (mySqlConnection.State==ConnectionState.Open)
         {
               mySqlConnection.Close();
         }
   }
}
```

- Para compilar e rodar sua aplicação selecione o comando de menu **Debug | Start Without Debugging** ou pressione **CTRL+F5**

- Clique no button **Update Command** para realizar toda a atualização da linha inserida usando o objeto SqlCommand

- Uma caixa de mensagem aparecerá dizendo “Location Data successfully updated.”

- Clique em **OK** para fechar a caixa de mensagem

- Feche o **Form1**

- Para verificar se o dado foi atualizado com sucesso

    - Volte para o **SQL Server Management Studio.**

- Se a tabela **Production.Location** estiver aberta, feche-a, clique com o botão direito no nó **Production.Location** e acesse **Open Table.**
- Verifique no final da tabela que o CostRate mudou para “10,00”.
- Volte para o **Visual C# Express.**

- Para abrir o **Solution Explorer,** selecione o comando de menu **View | Solution Explorer**
- Dê um clique com o botão direito do mouse em Form1.cs na Solution Explorer e selecione **View Code**
- Dê um duplo-clique no controle button **Delete Command**. O Visual C# Express irá navegar para a visualização de código do **Form1** (Form1.cs) e criará um manipulador de evento para o evento **Click** do button **Delete Command**.
- Adicione o seguinte código no método **btnDelete_Click** para apagar o dado inserido utilizando o objeto SqlCommand:

```
private void btnDelete_Click(object sender, EventArgs e)
{
   SqlCommand mySqlCommand = new SqlCommand();
   // Open the connection
   mySqlConnection.Open();

   //Assign the connection property.
   mySqlCommand.Connection=mySqlConnection;
   try
   {
         // Delete XYZ customer
         mySqlCommand.CommandText = "DELETE FROM Production.Location WHERE
Name='XYZ'";
         mySqlCommand.ExecuteNonQuery();
         MessageBox.Show("Customer Data successfully deleted.");
   }
   catch(Exception ex)
   {
         MessageBox.Show("Error occurred, data could not be deleted: " +
ex.Message);
   }
   finally
   {
         // Close the connection if it is open
         if (mySqlConnection.State==ConnectionState.Open)
         {
               mySqlConnection.Close();
         }
   }
}
```

- Para compilar e rodar sua aplicação selecione o comando de menu **Debug | Start Without Debugging** ou pressione **CTRL+F5**
- Clique no button **Delete Command** para realizar a operação de exclusão

- Uma caixa de mensagem aparecerá dizendo “Customer Data successfully deleted.”.
- Clique em **OK** para fechar a caixa de mensagem
- Feche o **Form1**
- Para verificar se o dado foi excluído com sucesso
    - Volte para o **SQL Server Management Studio.**
    - Se a tabela **Production.Location** estiver aberta, feche-a, clique com o botão direito no nó **Production.Location** e acesse **Open Table.**
    - Verifique no final da tabela que o registro foi excluído.
    - Arraste até o final da tabela para verificar que o cliente com o ID ‘XYZ’ não está presente
    - Volte para o **Visual C# Express.**

Tarefa 3 – Trabalhando com o objeto SqlTransaction

Uma transação é uma unidade de trabalho em qual uma série de operações ocorrem entre os comandos BEGIN TRANSACTION e END TRANSACTION de uma aplicação. Uma transação executa exatamente uma vez e é atômica – todo o trabalho ou nenhum é realizado. A classe SqlTransaction representa uma transação Transact-SQL para ser feita em um banco de dados SQL Server. Nesta tarefa, você trabalhará com o objeto SqlTransaction.

- O seguinte exemplo de código representa a estrutura de uma operação transacional:

```
SqlTransaction myTrans;

myTrans = mySqlConnection.BeginTransaction();
SqlCommand.Transaction = myTrans;
try
{
      ...
      ...
      ...
      myTrans.Commit();
}
catch(Exception ex)
{
      myTrans.Rollback();
}
```

- Para abrir **Solution Explorer,** selecione o comando de menu **View | Solution Explorer**
- Dê um duplo clique em **Form1.cs**.
- Para abrir a **Toolbox**, selecione o comando de menu **View | Toolbox**
- Arraste um **Button** da toolbox e solte o mesmo no formulário

- Dê um clique com o botão direito do mouse no button e selecione o comando de menu **Properties**

- Configure as seguintes propriedades para o controle usando a janela **Properties**:

  Name - "btnTransaction"

  Text - "Transaction"

- Aumente a largura do controle button para caber o texto

- Dê um duplo - clique no controle button **Transaction**. O Visual C# Express irá navegar para a visualização de código do **Form1** (Form1.cs) e criará um manipulador de evento para o evento **Click** do button **Transaction**

- Adicione o seguinte código no método **btnTransaction_Click** para realizar uma operação transacional de inserção, atualização, e exclusão usando o objeto SqlCommand:

```
private void btnTransaction_Click(object sender, EventArgs e)
{
   SqlTransaction myTrans;
   SqlCommand mySqlCommand = new SqlCommand();
   // Open the connection
   mySqlConnection.Open();

   //Assign the connection property.
   mySqlCommand.Connection=mySqlConnection;
   myTrans = mySqlConnection.BeginTransaction();
   mySqlCommand.Transaction = myTrans;
   try
   {
         // Insert new record for XYZ customer
         mySqlCommand.CommandText = "INSERT INTO Production.Location (Name,
CostRate) Values ('XYZ', '10,00')";
         mySqlCommand.ExecuteNonQuery();

         // Update XYZ customer
         mySqlCommand.CommandText = "UPDATE Production.Location SET
CostRate='55,00' WHERE Name='XYZ'";
         mySqlCommand.ExecuteNonQuery();

         // Delete XYZ customer
         mySqlCommand.CommandText = "DELETE FROM Customers WHERE Name='XYZ'";

         mySqlCommand.ExecuteNonQuery();
         MessageBox.Show("Committing transaction");

         myTrans.Commit();
         MessageBox.Show("Data successfully updated.");
   }

   catch(Exception ex)
   {
         myTrans.Rollback();
         MessageBox.Show("Error occurred, data has not been successfully
updated: " + ex.Message);
```

```
    }

    finally
    {
          // Close the connection if it is open
          if (mySqlConnection.State==ConnectionState.Open)
          {
                mySqlConnection.Close();
          }
    }
}
```

- Para compilar e rodar sua aplicação selecione o comando de menu **Debug | Start Without Debugging** ou pressione **CTRL+F5**
- Clique no button **Transaction** para realizar todas as operações, ou todas as operações ocorrem ou nenhuma ocorre
- Uma caixa de mensagem dizendo “Committing transaction”
- Clique em **OK** para fechar a caixa de mensagem
- Uma caixa de mensagem dizendo “Data successfully updated.”
- Clique em **OK** para fechar a caixa de mensagem
- Feche o **Form1**
- Para verificar se a transação ocorreu com sucesso
- Para verificar se o dado foi excluído com sucesso
    - Volte para o **SQL Server Management Studio.**
    - Se a tabela **Customers** estiver aberta, feche-a, clique com o botão direito no nó **Customers** e acesse **Open Table.**
    - Arraste até o final da tabela para verificar que o cliente com o ID ‘XYZ’ não está presente
    - Volte para o **Visual C# Express.**

Tarefa 4 – Usando o SqlDataReader

O SqlDataReader fornece um mecanismo para leitura de um lote de dados forward-only de linhas retornadas em uma query de banco de dados SQL Server. Nesta tarefa você usará o objeto SqlDataReader. Você usará o parâmetro CommandBehavior.CloseConnection do DataReader para que a conexão se feche automaticamente quando a instância do objeto SqlDataReader é fechada.

- Para abrir **Solution Explorer,** selecione o comando de menu **View | Solution Explorer**
- Dê um clique com o botão direito do mouse em Form1.cs na Solution Explorer e selecione **View Designer**

- Para abrir a **Toolbox**, selecione o comando de menu **View | Toolbox**
- Arraste outro **Button** da toolbox e solte o mesmo sobre o formulário como mostrado na Figura 2.3 a seguir
- Dê um clique com o botão direito do mouse no button e selecione Properties
- Configure as seguintes propriedades para o controle usando a janela **Properties**:

  Name - "btnDataReader"

  Text - "Data Reader"

- Aumente a largura do controle button para caber o texto
- Arraste uma **ComboBox** da toolbox e solte o mesmo sobre o formulário como mostrado na Figura 2.3 abaixo
- Dê um clique com o botão direito do mouse na combo box e selecione Properties
- Configure as seguintes propriedades para o controle usando a janela **Properties**:

  DropDownStyle - DropDownList (selecione da lista de opções)

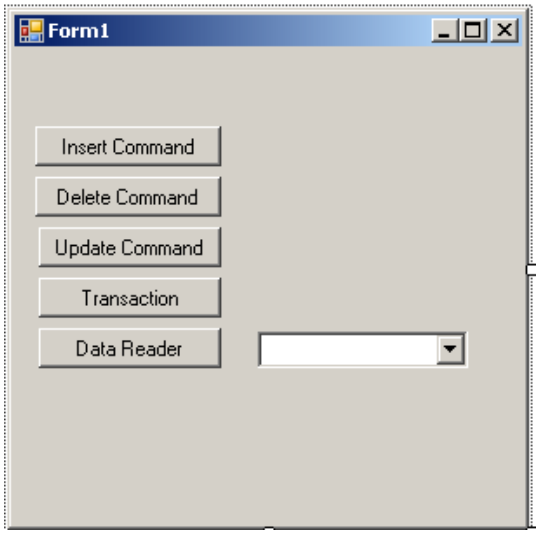


Figure 2.3 Visualização de desenho do Form1

- Dê um duplo-clique no controle button **Data Reader**. O Visual C# Express irá navegar para a visualização de código do **Form1** (Form1.cs) e criará um manipulador de evento para o evento **Click** do button **Data Reader**
- Abra uma conexão de banco de dados e leia o dado no SqlDataReader usando o objeto SqlCommand no método **btnDataReader_Click** como mostrado abaixo:

```
private void btnDataReader_Click(object sender, EventArgs e)
{
   SqlDataReader myReader = null;
```

```
   SqlCommand mySqlCommand = new SqlCommand("SELECT Name FROM
Production.Location", mySqlConnection);
   try
   {
         mySqlConnection.Open();
         myReader = mySqlCommand.ExecuteReader(CommandBehavior.CloseConnection);
         while (myReader.Read())
         {
               comboBox1.Items.Add(myReader["Name"].ToString());
         }
         MessageBox.Show("All names loaded successfully!");
   }
   catch(Exception ex)
   {
         MessageBox.Show("Error occurred: " + ex.Message);
   }
   finally
   {
         if (myReader != null)
               myReader.Close();
   }
}
```

- Para compilar e rodar sua aplicação selecione o comando de menu **Debug | Start Without Debugging** ou pressione **CTRL+F5**
- Clique no button **Data Reader** para carregar os nomes de contato de todos os clientes na caixa combo box
- Uma caixa de mensagem aparecerá dizendo “All names loaded successfully!”
- Clique em **OK** para fechar a caixa de mensagem. A combo Box é populada com o dado
- Expanda a combo box para verificar que os nomes de contatos estão listados
- Feche o **Form1**

Nota: Uma vez que você usou o parâmetro CommandBehavior.CloseConnection no método ExecuteReader do objeto SqlCommand, o estado de conexão é fechado quando o DataReader é fechado. Você não precisa, portanto, fechar a conexão explicitamente.

Exercício3 – Trabalhando com o objeto DataSet

Neste exercício você vai popular um DataSet utilizando o objeto SqlDataAdapter com o intuito de recuperar dados para a camada de interface. O DataSet (ADO.NET) é uma representação residente em memória de dados que provê um modelo de programação relacional independente da fonte de dados. O objeto DataSet representa um conjunto completo de dados incluindo tabelas, regras, e relacionamentos entre tabelas. Sendo o DataSet independente da fonte de dados, um DataSet pode incluir dados de múltiplas fontes. Interação com as fontes de dados existentes é controlada através do DataAdapter.

Neste exercício você irá:

- Popular um DataSet com dados usando o objeto SqlDataAdapter.

- Usar os objetos SqlCommandBuilder e SqlDataAdapter para inserir, atualizar e excluir dados.
- Chamar uma stored procedure e persistir o DataSet em Xml.
- Criar uma DataRelation

Tarefa 1 – Recuperando dados com o DataSet

Nesta tarefa você recuperará dados usando um objeto SqlDataAdapter e irá preencher um objeto DataSet. Finalmente você irá acoplar o DataSet a um controle DataGridView .

- Para abrir **Solution Explorer,** selecione o comando de menu **View | Solution Explorer**
- Dê um clique com o botão direito do mouse no projeto (Lab04-CS) na Solution Explorer e selecione **Add | Windows Form…** A janela de diálogo **Add New Item** aparecerá
- Clique em **Add**. O Form2.cs é adicionado ao projeto e está visível na Solution Explorer
- Dê um clique com o botão direito do mouse em Form2.cs na Solution Explorer e selecione **View Code**
- Adicione o seguinte código em destaque no topo do código

```
...
...
using System.Windows.Forms;
using System.Data.SqlClient;
```

- Declare variáveis para **SqlConnection**, **DataSet** e **SqlDataAdapter** na seção de declaração geral da classe Form2. O **SqlDataAdapter** irá gerenciar as atualizações para a tabela **Production.Location**

```
public partial class Form2 : Form
{
   SqlConnection mySqlConnection;
   DataSet myDataSet;
   SqlDataAdapter mySqlDataAdapter;
...
...
```

- Para acessar a visualização de projeto do **Form2**, selecione o comando de menu **View | Designer**
- Dê um duplo-clique na visualização de projeto do **Form2,** O Visual C# Express irá navegar para a visualização de código do **Form2** (Form2.cs) e criará um método **Form2_Load** para o formulário
- Configure a propriedade **ConnectionString** da variável declarada no manipulador de eventos **Form2_Load**:

```
private void Form2_Load(object sender, System.EventArgs e)
{
   mySqlConnection = new SqlConnection (
System.Configuration.ConfigurationSettings.AppSettings.Get("ConnectionString"));
}
```

- Para acessar a visualização de projeto do **Form2**, selecione o comando de menu **View | Designer**
- Para abrir a **Toolbox**, selecione o comando de menu **View | Toolbox**
- Arraste um **Button** da toolbox e solte o mesmo no Form2
- Dê um clique com o botão direito do mouse no button e selecione Properties
- Configure as seguintes propriedades para o controle usando a janela **Properties**:

  Name - "btnGet"

  Text - "Get Data"

- Para abrir a **Toolbox**, selecione o comando de menu **View | Toolbox**
- Arraste um controle **DataGridView** da seção **Data** da toolbox e solte o mesmo no formulário como mostrado na Figura 3.1 abaixo. Um controle DataGridView é adicionado com o nome **DataGridView 1**

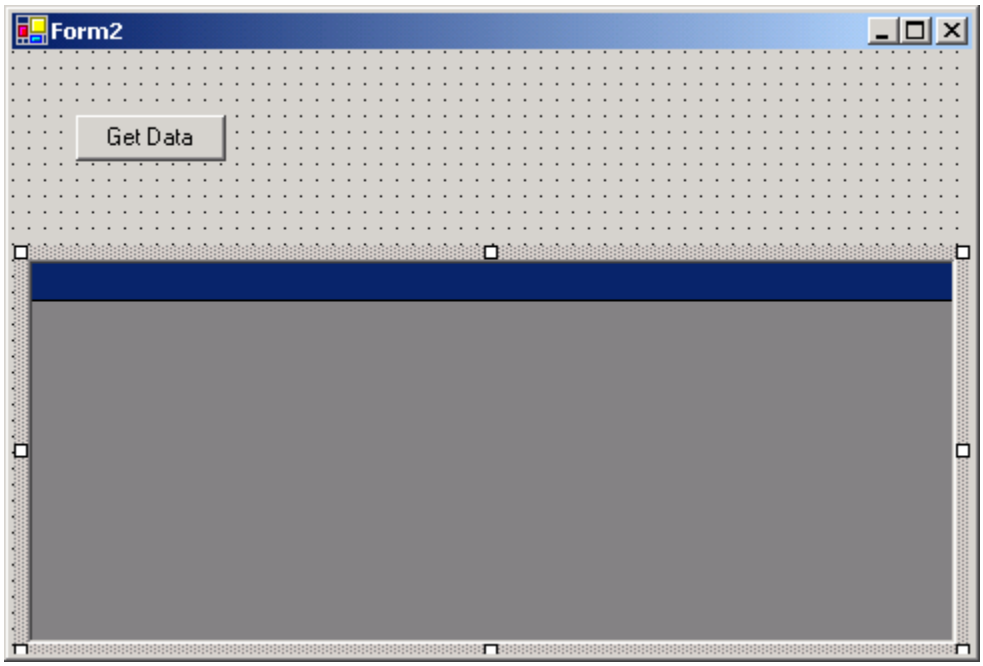


Figura 3.1 Visualização de projeto do Form2

- Dê um duplo-clique no botão **Get Data**
- Adicione o seguinte código em destaque ao método **btnGet_Click**

```
private void btnGet_Click(object sender, EventArgs e)
{
   mySqlDataAdapter = new SqlDataAdapter("SELECT * FROM Production.Location",
mySqlConnection);
   try
   {
```

```
         myDataSet = new DataSet();
         mySqlDataAdapter.Fill(myDataSet," Production.Location");
         dataGridView1.DataSource=
myDataSet.Tables["Production.Location"].DefaultView;
   }



   catch(Exception ex)
   {
         MessageBox.Show("Unable to retrieve Production.Location data: " +
ex.Message);
   }
}
```

- Para carregar o **Form2** durante a inicialização da aplicação
    - Abra o **Solution Explorer,** selecionando o comando de menu **View | Solution Explorer**
    - Dê um duplo clique no arquivo **Program.cs**
    - No arquivo **Program.cs**, localize o método **Main()**
    - Para carregar **Form2** quando a aplicação é executada, mude o parâmetro do método **Application.Run** para Form2()

```
static void Main()
{
   Application.Run(new Form2());
}
```

- Para compilar e rodar sua aplicação selecione o comando de menu **Debug | Start Without Debugging** ou pressione **CTRL+F5**
- Para carregar dados no controle DataGridView , clique no button **Get Data**
- Feche o **Form2**

Tarefa 2 – Persistindo o DataSet em um documento XML

Nesta tarefa você irá persistir um DataSet em um documento XML e irá carregar dados originalmente no formato XML no mesmo DataSet.

- Para abrir **Solution Explorer,** selecione o comando de menu **View | Solution Explorer**
- Para abrir visualização de projeto do **Form2**, dê um duplo-clique em Form2.cs na Solution Explorer
- Para abrir a **Toolbox**, selecione o comando de menu **View | Toolbox**
- Arraste um **Button** da toolbox e solte o mesmo no Form2
- Dê um clique com o botão direito do mouse no button e selecione Properties
- Configure as seguintes propriedades para o controle usando a janela **Properties**:

Name - “btnSave”

Text - “Save Xml”

- Arraste um **Button** da toolbox e solte o mesmo no Form2
- Dê um clique com o botão direito do mouse no button e selecione Properties
- Configure as seguintes propriedades para o controle usando a janela **Properties**:

Name - “btnLoad”

Text - “Load Xml”

- Para visualizar o **Form2** no modo de edição de código, selecione o comando de menu **View | Code**
- Declare uma variável **String** na seção de declaração geral da classe Form2. Esta string irá armazenar o caminho do arquivo XML no qual o DataSet será persistido

```
public class Form2 : Form
{
   SqlConnection mySqlConnection;
   DataSet myDataSet;
   SqlDataAdapter mySqlDataAdapter;
   string xmlFilename=@"C:\Location.xml";
   ...
   ...
```

- Para acessar a visualização de projeto do Form2, selecione o comando de menu **View | Designer**
- Dê um duplo-clique no controle button **Save Xml**
- Adicione o seguinte código em destaque ao método **btnSave_Click**

```
private void btnSave_Click(object sender, EventArgs e)
{
   myDataSet.WriteXml(xmlFilename);
   MessageBox.Show("Production.Location details stored in: " + xmlFilename);
   myDataSet.Clear(); //this will clear the grid also
}
```

- Para acessar a visualização de projeto do Form2, selecione o comando de menu **View | Designer**
- Dê um duplo-clique no controle button **Load Xml**
- Adicione o seguinte código em destaque ao método **btnLoad_Click**

```
private void btnLoad_Click(object sender, EventArgs e)
{
   MessageBox.Show("Reading data from XML file... ");
   myDataSet.ReadXml(xmlFilename);
   dataGridView1.DataSource=
myDataSet.Tables["Production.Location"].DefaultView;
}
```

- Para compilar e rodar sua aplicação selecione o comando de menu **Debug | Start Without Debugging** ou pressione **CTRL+F5**

- Clique em **Get Data**. A aplicação irá carregar a tabela Production.Location do banco de dados AdventureWorks

- Clique em **Save Xml** para escrever os dados de cliente e pedidos no arquivo XML C:\Location.xml. Clique em **OK** para fechar a caixa de mensagem "Location details stored in C:\ Location.xml"

- Clique em **Load Xml** para ler os dados de cliente e pedidos do arquivo XML. Clique em **OK** para fechar a caixa de mensagem "Reading data from XML File". A aplicação irá ler o dado do arquivo C:\Location.xml e irá preencher os dados no controle DataGridView usando o objeto DataSet

- Feche o **Form2**

- Abra o C:\Location.xml no Internet Explorer e note como todos os dados do DataSet estão persistidos neste arquivo

Tarefa 4– Usando Stored Procedures

Nesta tarefa você fará a chamada a uma stored procedure usando o objeto SqlDataAdapter

- Para acessar a visualização de projeto do **Form2**, selecione o comando de menu **View | Designer**

- Para abrir a **Toolbox**, selecione o comando de menu **View | Toolbox**

- Arraste um **Button** da toolbox e solte o mesmo na visualização de projeto do formulário como mostrado na Figura 3.9 a seguir

- Dê um clique com o botão direito do mouse no button e selecione Properties

- Configure as seguintes propriedades para o controle usando a janela **Properties**:

  Name - "btnSP"

  Text - "Call SP"

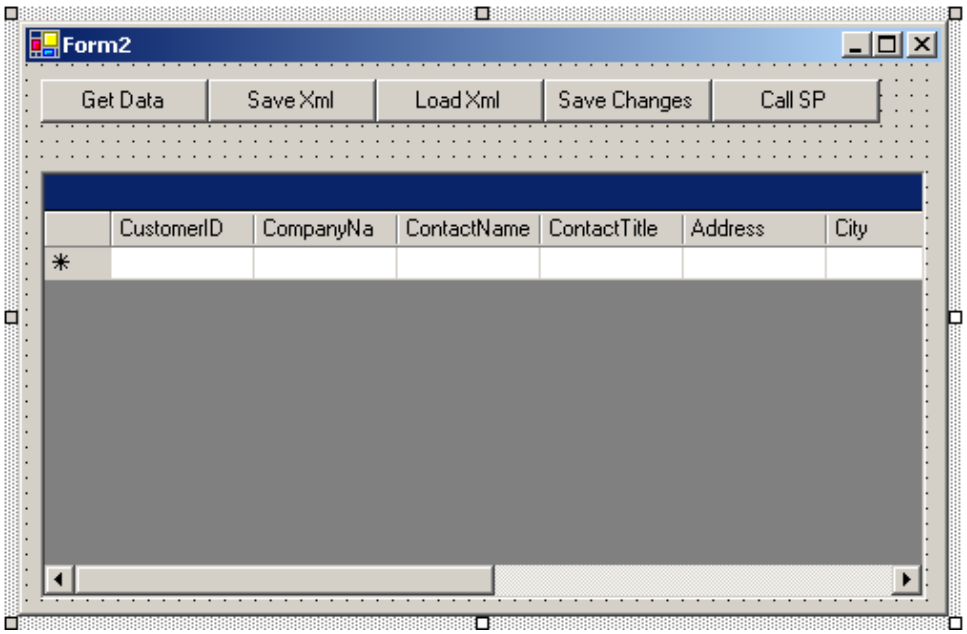


Figure 3.9: Visualização de desenho do Form2

- Para visualizar as propriedades do controle DataGridView , dê um clique com o botão direito do mouse no controle DataGridView e selecione o comando de menu **Properties**
- Dê um duplo-clique no controle button **Call SP**
- Adicione o seguinte código em destaque ao método **btnSP_Click**:

```
private void btnSP_Click(object sender, EventArgs e)
{
try
{
   SqlDataAdapter myDataAdapter = new SqlDataAdapter("uspGetEmployeeManagers",
mySqlConnection);
   myDataAdapter.SelectCommand.CommandType = CommandType.StoredProcedure;
   myDataAdapter.SelectCommand.Parameters.Add(new SqlParameter("@EmployeeID",
SqlDbType.Int));
   myDataAdapter.SelectCommand.Parameters["@EmployeeID"].Value = "5";
   DataSet ds = new DataSet();
   myDataAdapter.Fill(ds, " HumanResources.Employee");
   dataGridView1.DataSource=ds.Tables["HumanResources.Employee"].DefaultView;
}
catch(Exception ex)
{
   MessageBox.Show("Error occurred: " + ex.Message);
}
}
```

- Este código de exemplo recupera os históricos de pedidos de um customer ID específico através da chamada da stored procedure CustOrderHist
- Para compilar e rodar sua aplicação selecione o comando de menu **Debug | Start Without Debugging** ou pressione **CTRL+F5**

- Clique em **Call SP**. A lista dos empregados e seus respectivos gerentes será mostrada no controle DataGridView
- Feche o **Form2**

Exercício4 – Carregando um Recordset nativo ADO em um DataSet

Neste exercício você usará um objeto Recordset ADO 2.x em conjunto com um Dataset (ADO.NET) . Isto será feito usando o overload do método OleDbDataAdapter.Fill do Data Provider OLEDB, que aceita um objeto Recordset ADO.

Tarefa 1 – Adicionando uma referência ao Microsoft ActiveX Data Objects

- De maneira a usar o ADO em uma aplicação .NET, uma referência à biblioteca Microsoft ActiveX Data Object type é necessária. Para configurar esta referência, abra o **Solution Explorer** usando o comando de menu **View | Solution Explorer**
- Dê um clique com o botão direito do mouse no nó **References** na Solution Explorer e selecione **Add Reference**. A janela de diálogo **Add Reference** aparecerá
- Mude para a aba **COM** na janela de diálogo **Add Reference**
- Selecione **Microsoft ActiveX Data Objects 2.7 Library** na lista **Component Name**
- Clique em **Select** para adicionar uma referência a biblioteca de tipos selecionada
- Click em **OK** para fechar a janela de diálogo **Add Reference**

Tarefa 2 – Carregando um Recordset nativo ADO em um DataSet

- Para abrir **Solution Explorer,** selecione o comando de menu **View | Solution Explorer**
- Dê um clique com o botão direito do mouse no projeto (Lab05-CS) na Solution Explorer e selecione **Add | Windows Form…**
- Para adicionar o Form3.cs ao projeto corrente, clique em **Add**
- Dê um clique com o botão direito do mouse em Form3.cs na Solution Explorer e selecione **View Code**
- Na parte superior da seção de código, adicione o seguinte código em destaque:

```
..
...
using System.Windows.Forms;
using System.Data.SqlClient;
using System.Data.OleDb;
using ADODB;
```

- Para acessar a visualização de projeto do **Form3**, selecione o comando de menu **View | Designer**
- Para abrir a **Toolbox**, selecione o comando de menu **View | Toolbox**

- Arraste um controle **DataGridView** da toolbox e solte o mesmo sobre a visualização de projeto do Form3
- Arraste um controle **Button** da toolbox e solte o mesmo sobre a visualização de projeto do Form3
- Dê um clique com o botão direito do mouse no button e selecione Properties
- Configure as seguintes propriedades para o controle usando a janela **Properties**:

  Name - "btnLoadRS"

  Text - "Load Recordset"

- Aumente a largura do controle button para caber o texto

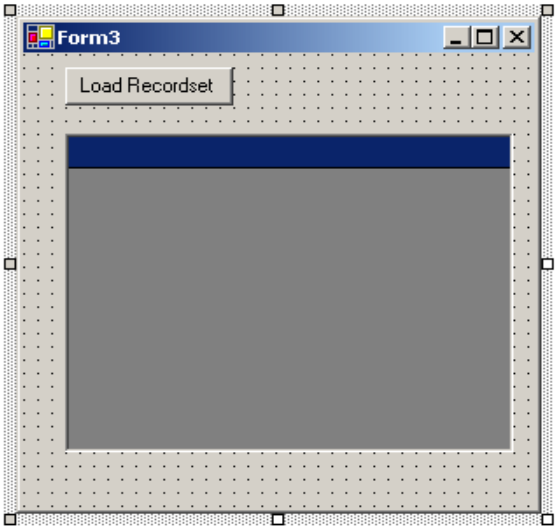


Figure 4.1: Visualização de desenho do Form3

- Dê um duplo-clique no botão **Load Recordset**
- Adicione o seguinte código em destaque ao método **btnLoadRS_Click**:

```
private void btnLoadRS_Click(object sender, EventArgs e)
{
   try
   {
         RecordsetClass rsObj = new RecordsetClass();
         string constr =
"provider=sqloledb;server=NomeDaMaquina\\SQLEXPRESS;database=AdventureWorks;
trusted_connection=yes";

   rsObj.Open("HumanResources.Employee",constr,ADODB.CursorTypeEnum.adOpenForwar
dOnly,ADODB.LockTypeEnum.adLockReadOnly,0x200);

         DataSet myDataSet = new DataSet();
```

```
            OleDbDataAdapter adapter = new OleDbDataAdapter();
            adapter.MissingSchemaAction = MissingSchemaAction.AddWithKey;
            adapter.Fill(myDataSet, rsObj, "Production.Location");
            dataGridView1.DataSource                                                    =
myDataSet.Tables["Production.Location"].DefaultView;
      }
      catch(Exception ex)
      {
            MessageBox.Show("Error occured while retrieving the data: " +
ex.Message);
      }
}
```

- Para carregar **Form3** durante a inicialização da aplicação
    - Abra o **Solution Explorer,** selecionando o comando de menu **View | Solution Explorer**
    - Dê um clique com o botão direito do mouse em Program.cs na Solution Explorer e selecione **View Code**
    - No arquivo fonte localize o método **Main()**
    - Para carregar **Form3** quando a aplicação é executada, mude o parâmetro do método **Application.Run** para Form3()

```
static void Main()
{
   Application.Run(new Form3());
}
```

- Para compilar e rodar sua aplicação selecione o comando de menu **Debug | Start Without Debugging** ou pressione **CTRL+F5**
- Para carregar os dados de cliente no controle DataGridView , clique no button **Load Recordset**
- Feche o **Form3**
- Par fechar a solução do Visual Studio, selecione o comando de menu **File | Exit**

Resumo do Laboratório

Neste Lab você realizou os seguintes Exercícios:

- Entendendo o ADO.NET
- Trabalhando com os objetos DataSet e SqlDataAdapter
- Trabalhando com os objetos SqlDataReader e SqlCommand
- Carregando um Recordset nativo ADO em um DataSet