**ATIVIDADE 10.2 – TRABALHANDO COM OBJETOS DATASET**

Nesta atividade, você irá começar a familiarizar abaixo:

* Populando um *DataSet* de um banco de dados SQL Server.
* Editando os dados em um *DataSet*.
* Atualizando mudanças do *DataSet* para o Banco de Dados.
* Estabilizando relacionamentos entre tabelas em um *DataSet*.

**Populando um DataSet de um banco de dados SQL Server**

Para popular um *DataSet* de um banco de dados SQL Server, siga estes passos:

1. Inicie o Visual Studio. Selecione *File* -> *New* -> *Project*.
2. Escolha *Windows Forms Application*. Renomeie o projeto para *Activity10\_2* e clique no botão OK.
3. Depois de abrir o projeto, adicione uma nova classe para o projeto com o nome *Author*.
4. Abra o código da classe *Author* no editor de código. Adicione as seguintes instruções *using* no topo do arquivo:

*using System.Data;*

*using System.Data.SqlClient;*

1. Adicione o seguinte código para declarar uma variável de nível de classe *private* (privado) para a *connection string*.

*class Author*

*{*

*string \_connString = “Data Source=localhost; Initial Catalog =pubs; Integrated Security = True”;*

*}*

1. Abaixo da *connection string*, crie um método para classe *Author* chamado *GetData* que irá usar um objeto *DataAdapter* para preencher o *DataSet* e retornar isto para o cliente.

*public DataSet GetData()*

*{*

*DataSet authorDataSet;*

*using (SqlConnection pubConnection = new SqlConnection())*

*{*

*pubConnection.ConnectionString = \_connString;*

*using(SqlCommand selectCommand = new SqlCommand())*

*{*

*selectCommand.CommandText = “Select au\_id, au\_lname, au\_fname from authors”;*

*selectCommand.Connection = pubConnection;*

*using(SqlDataAdapter pubDataAdapter = new SqlDataAdapter());*

*{*

*pubDataAdapter.SelectCommand = selectCommand;*

*authorDataSet = new DataSet();*

*pubDataAdapter.Fill(authorDataSet, “Author”);*

*}*

*}*

*return authorDataSet;*

*}*

*}*

1. Rode o projeto e verifique se há alguns erros.
2. Adicione os controles listados na tabela abaixo e o conjunto de propriedades mostrados abaixo:

|  |  |  |
| --- | --- | --- |
| Control  (Controle) | Property (Propriedades) | Value  (Valores) |
| DataGridView | Name  AllowUserToAddRows  AllowUserToDeleteRows  ReadOnly | dgvAuthors  False  False  False |
| Button | Name  Text | btnGetData  Get Data |
| Button | Name  Text | btnUpdate  Update |

1. Abra o arquivo com o código da classe no editor de código. Declare um objeto nível-classe (class-level) *DataSet* depois da declaração da classe.

*public partial class Form1 : Form*

*{*

*Private DataSet \_pubDataSet;*

*}*

1. Abra o *Form1* no *Form Designer*. Dê um duplo clique no botão *(button) Get Data* para abrir o botão do método *click event* no editor de código.
2. Adicione o seguinte código para a procedimento do *click event btnGetData* definido na classe *Author*. Este *DataSet* é então carregado dentro do *Grid* usando a propriedade *DataSource.*

*private void btnGetData\_Click(object sender, EventArgs e)*

*{*

*Author author = new Author();*

*\_pubDataSet = author.GetData();*

*dgvAuthors.DataSource = \_pubDataSet.Tables[“Author”];*

*}*

1. Rode o projeto e verifique se há algum erro. Desde que o projeto é construído, rode o projeto no modo *Debug* e teste o método *GetData*. Você pode ver que o *Grid* é preenchido com as informações dos autores. Depois de testar, feche o depurador.

**Editando e Atualizando os Dados em um DataSet**

Para editar e atualizar os dados em um *DataSet*, siga estes passos:

1. Abra o código da classe do código *Author* no editor de código.
2. Criado um método da classe *Author* chamado *UpdateData* que irá usar o método *Update* do objeto *DataAdapter* para passar as atualizações feitas para o *DataSet* para o banco de dados *Pubs*.

*public void UpdateData(DataSet changedData)*

*{*

*using (SqlConnection pubConnection = new SqlConnection())*

*{*

*pubConnection.ConnectionString = \_connString;*

*using(SqlCommand updateCommand = new SqlCommand())*

*{*

*updateCommand.CommandText = “Update authors set au\_lname = @au\_lname, au\_fname = @au\_fname where au\_id = @au\_id ”;*

*updateCommand.Parameters.Add(“@au\_id”, SqlDbType.VarChar, 11, “au\_id”;*

*updateCommand.Parameters.Add(“@au\_lname”, SqlDbType.VarChar, 40, “au\_lname”);*

*updateCommand.Parameters.Add(“@au\_fname”, SqlDbType.VarChar, 40, “au\_fname”);*

*updateCommand.Connection = pubConnection;*

*using(SqlDataAdapter pubDataAdapter = new SqlDataAdapter())*

*{*

*pubDataAdapter.UpdateCommand = updateCommand;*

*pubDataAdapter.Update(changedData, “Author”);*

*}*

*}*

*}*

*}*

1. Rode o projeto e verifique se há alguns erros.
2. Abra *Form1* no *Form Designer*. Dê um duplo clique no botão *Update Data* para abrir o método do botão o evento *click* no editor de código.
3. Adicionando o seguinte código para o procedimento do evento *click btnUpdate*, com isso irá executar o método *Update Data* definido na classe *Author*. Usando o método *GetChanges* do objeto *DataSet*, apenas os dados que tinha mudado é passado para atualização.

*private void btnUpdate\_Click(object sender, EventArgs e)*

*{*

*Author author = new Author();*

*author.UpdateData(\_pubDataSet.GetChanges());*

*}*

1. Rode o projeto e verifique se há alguns erros. Desde que rode o projeto, rode o projeto no modo *Debug* e teste o método *Update*. Primeiro, clique no botão *Get Data*. Mude o último nome dos autores em geral e clique no botão *Update Button*. Clique no botão *Get Data* novamente para acessar os valores das mudanças de volta de um banco de dados (*database*). Depois de testar, pare o depurador.

**Estabelecendo relacionamentos entre tabelas em um DataSet**

1. Adicione uma nova classe nomeado *StoreSales* para o projeto.
2. Abra o código da classe *StoreSales* no editor de código. Adicione a seguinte instrução *using* no topo do arquivo:

*using System.Data;*

*using System.Data.SqlClient;*

1. Adicione o seguinte código para declarar as variáveis da classe no nível *private* para a *connection string* e *DataSet*.

class StoreSales

{

*string \_connString = “Data Source=localhost; Initial Catalog =pubs; Integrated Security = True”;*

}

1. Crie um método na classe *StoreSales* chamada *GetData* que irá selecionar a informação armazenada e informação de venda e estabelecer uma relação entre eles. Esta informação é usada para preencher um *DataSet* e retorna isto para o cliente.

*public DataSet GetData()*

*{*

*DataSet storeSalesDataSet;*

*storeSalesDataSet = new DataSet();*

*using (SqlConnection pubConnection = new SqlConnection(\_connString))*

*{*

*//Fill store table*

*using (SqlCommand storeCommand = new SqlCommand())*

*{*

*storeCommand.Connection = pubConnection;*

*storeCommand.CommandText = “SELECT [stor\_id],[stor\_name], [city],[state] FROM [stores]”;*

*using(SqlDataAdapter storeDataAdapter = new SqlDataAdapter())*

*{*

*storeDataAdapter.SelectCommand = storeCommand;*

*storeDataAdapter.Fill(storeSalesDataSet, “Stores”);*

*}*

*}*

*//Fill sales table command*

*using (SqlCommand salesCommand = new SqlCommand())*

*{*

*salesCommand.Connection = pubConnection;*

*salesCommand.CommandText = “SELECT [stor\_id], [ord\_num], [ord\_date], [qty] FROM [sales]”;*

*using(SqlDataAdapter salesDataAdapter = new SqlDataAdapter())*

*{*

*salesDataAdapter.SelectCommand = salesCommand;*

*salesDataAdapter.Fill(storeSalesDataSet, “Sales”);*

*}*

*}*

*//Create relationship between tables*

*DataColumn Store\_StoreIdColumn = storeSalesDataSet.Tables[“Store”].Column[“stor\_id”];*

*DataColumn Sales\_StoreIdColumn = storeSalesDataSet.Tables[“Sales”].Columns[“stor\_id”];*

*DataRelation StoreSalesRelation = new DataRelation (“StoresToSales”, Store\_StoreIdColumn, Sales\_StoreIdColumn);*

*storeSalesDataSet.Relations.Add(StoreSalesRelation);*

*return storeSalesDataSet;*

*}*

*}*

1. Rode o projeto e verifique se há alguns erros.
2. Adicione um segundo formulário (form) para o projeto. Adicione o controle listados na tabela abaixo para Form2 e seta as propriedades mostradas:

|  |  |  |
| --- | --- | --- |
| Control  (Controle) | Property (Propriedades) | Value  (Valores) |
| DataGridView | Name | dgvStores |
| DataGridView | Name | dgvSales |
| Button | Name  Text | btnGetData  Get Data |

1. Abra o código na classe *Form2* no editor de código. Declare um nível de acesso no objeto DataSet depois na declaração da classe.

*public partial class Form2 : Form*

*{*

*DataSet StoreSalesDataSet;*

*}*

1. Abra o *Form2* no *Form Designer*. Dê um duplo-clique no botão *Get Data* para abrir o método do botão evento *click* no editor de código.
2. Adicione o seguinte código para o procedimento do botão *btnGetData* no evento *click*, com isso irá executar o método *GetData* definido na classe *StoreSales*. Esta tabela Store é então carregada dentro do *grid Stores* usando a propriedade *DataSource*. Configurando a propriedade *DataMember* do *grid Sales* carrega isto com os dados armazenados selecionados no *grid Stores*.

*private void btnGetData\_Click(object sender, EventArgs e)*

*{*

*StoreSales storeSales = new StoreSales();*

*StoresSalesDataSet = storeSales.GetData();*

*dgvStores.DataSource = StoreSalesDataSet.Tables[“Stores”];*

*dgvSales.DataSource = StoreSalesDataSet.Tables[“Stores”];*

*dgvSales.DataMember = “StoresToSales”;*

*}*

1. Abra a classe Program no editor de código. Mude o código para lançar o Form2 quando o formulário carrega.

*Application.Run(new Form2());*

1. Quando carrega o formulário *(form)*, clique no botão *Get Data* para carregar os *grids*. Selecionando uma nova linha na *grid Stores* para mostrar as vendas na loja. Quando você está finalizando o testes, pare o depurador e saia do Visual Studio.