Vamos usar o **SQL Server Compact 4.0 (x64)**  em uma aplicação Windows Forms usando a linguagem C#.

O Microsoft **SQL Server Compact 4.0**permite criar banco de dados compactos que podem ser distribuídos em computadores*desktop, smart devices, e Tablets.*

Quando você cria aplicações que usa o **SQL Server Compact**você pode usar tanto a linguagem Visual Basic .NET como a linguagem C# e o .NET Framework ou .NET Compact Framework para criar e gerenciar a aplicação.

O SQL Server Compact oferece os seguintes recursos que você deve considerar quando pretender usá-lo como local de armazenamento de dados para suas aplicações:

* O SQL Server Compact é baseado em arquivo, o que significa que a string de conexão é um caminho de arquivo para o arquivo de banco de dados (.SDF)
* O SQL Server Compact não é executado como um serviço. Esta é uma das principais diferenças entre o uso do SQL Server Compact e usando SQL Server ou SQL Server Express Edition.
* O SQL Server Compact dá suporte a várias conexões até o limite de conexão 256. Abrir conexões em diferentes processos também é suportado.
* A versão atual do SQL Server Compact dá suporte a arquivos de banco de dados de até 4 GB.

**Quando usar o SQL Server Compact ?**

O SQL Server Compact é um banco de dados baseado em arquivo que consiste em DLLs que ocupam cerca de 1,4 MB. A lista a seguir fornece algumas situações em que você pode querer usar o SQL Server Compact em suas aplicações:

* Para aplicativos destinados a computadores desktop e dispositivos móveis.
* Quando você precisa de um banco de dados que é livre para redistribuir.
* Quando os requisitos de tamanho de memória de aplicativos e devem ser compactos.
* Quando você quer o código de acesso de dados possa ser executado em um processo.

fonte: [http://msdn.microsoft.com/en-us/library/aa983341%28v=vs.110%29.aspx](http://msdn.microsoft.com/en-us/library/aa983341(v=vs.110).aspx) -

Neste artigo iremos abordar os seguinte assuntos:

1. Criar um banco de dados SQL Server Compact 4.0 via código;
2. Criar uma tabela e definir sua estrutura no SQL Server Compact 4.0 via código;
3. Carregar a tabela criada com dados;
4. Acessar os dados e exibir o resultado em um DataGridView;
5. Acessar os dados usando os recursos do SQL Server Compact 4.0 e exibir o resultado em uma **Label**;
6. Usar o controle **inputbox** na linguagem C#

Recursos utilizados:

1. Visual Studio 2012 Express for Windows Desktop - <http://www.microsoft.com/en-us/download/details.aspx?id=34673>
2. SQL Server Compact 4.0 SP1 - <http://www.microsoft.com/pt-br/download/details.aspx?id=30709>

**Preparando o ambiente - Criando o Projeto**

Abra o Visual Studio 2012 Express for Windows desktop e clique em**New Project;**

Selecione o template**Visual C# -> Windows** e a seguir **Windows Forms Application;**

Informe o nome **Gerenciando\_SQL\_Server\_Compact4** e clique no botão  OK;

No menu **PROJECT** clique em **Add Reference;**

Na janela Reference Manager clique em **Browse** e localize a pasta onde você instalou os arquivos do SQL Server Compact 4 selecionando a referência conforme abaixo:

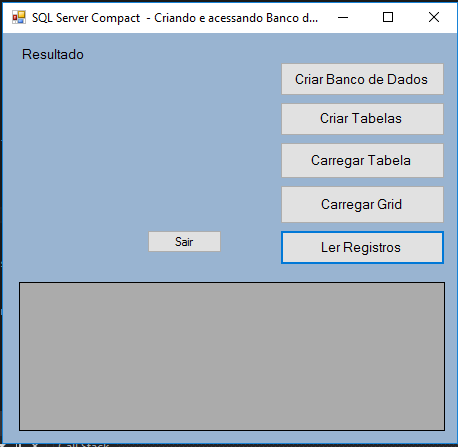
|  |
| --- |
| http://www.macoratti.net/14/03/c_cadcpt1.gif |

Atenção ! Se você tiver mais uma versão instalada verifica atentamente para estar referenciado a versão correta ou ocorreram erros no projeto.

Agora no formulário **form1.cs** inclua o seguintes controles:

* 1 Label - Text = Resultado
* 1 Label - name= lblResultado
* 5 Buttons - btnCriarBD, btnCriarTabelas, btnCarregarTabela, btnCarregarGrid e btnLerRegistros
* 1 DataGridView - name=-dgvDados

Disponha os controles conforme o leiaute da figura abaixo:



Defina os seguintes namespaces no formulário form1.cs:

using System;

using System.Data;

using System.Text;

using System.Windows.Forms;

using System.Data.SqlServerCe;

using System.IO;

using Microsoft.VisualBasic;

Agora vamos definir o código em cada evento **Click** de cada um dos botões de comando.

**1- Criar Banco de dados**

|  |
| --- |
| private void btnCriarBD\_Click(object sender, EventArgs e)  {  /\* define os parâmetros para o inputbox \*/  string Prompt = "Informe o nome do Banco de Dados a ser criado.Ex: Teste.sdf";  string Titulo = "www.macoratti.net";  **string Resultado = Interaction.InputBox(Prompt, Titulo, @"c:\dados\Agenda.sdf", 650, 350);**  /\* verifica se o resultado é uma string vazia o que indica que foi cancelado. \*/  if (Resultado != "")  {  'verifica se o nome informado contém o texto ".sdf"  if (!Resultado.Contains(".sdf"))  {  MessageBox.Show("Informe a extensão .sdf no arquivo...");  return;  }  try  {  string connectionString;  string nomeArquivoBD = Resultado;  string senha = "";  'verifica se o arquivo com o nome informado já existe  if (File.Exists(nomeArquivoBD))  {  if (MessageBox.Show("O arquivo já existe !. Deseja excluir e criar novamente ? ", "Excluir", MessageBoxButtons.YesNo,  MessageBoxIcon.Information) == DialogResult.Yes)  {  File.Delete(nomeArquivoBD);  }  else  {  return;  }  }  'monta a string de conexão com o banco de dados  **connectionString = string.Format("DataSource=\"{0}\"; Password='{1}'", nomeArquivoBD, senha);**  if (MessageBox.Show("Será criado arquivo " + connectionString +" Confirma ? ", "Criar", MessageBoxButtons.YesNo,  MessageBoxIcon.Information) == DialogResult.Yes)  {  **SqlCeEngine SqlEng = new SqlCeEngine(connectionString);**  **SqlEng.CreateDatabase();**  lblResultado.Text = "Banco de dados " + nomeArquivoBD + " com sucesso !";  }  else  {  return;  }  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  else  {  MessageBox.Show("A operação foi cancelada...");  }  } |

Neste código vamos criar um banco de dados SQL Server Compact usando o **inputbox** para que o usuário informe o nome do banco de dados.

O inputbox exibe um prompt em uma caixa de diálogo que aguarda que um texto seja informado ou um dos botões seja clicado e então retorna uma string contendo o conteúdo do TextBox.

E como usar o **inputbox** na linguagem C# ?

A saída mais fácil é fazer uma referência no seu projeto a Microsoft.VisualBasic.

No menu **Project** selecione **Add Reference** e na guia .NET selecione **Microsoft.VisualBasic** e clique em OK;

|  |
| --- |
| http://www.macoratti.net/14/03/c_cadcpt3.gif |

Após isso basta usar o método**Interaction.InputBox.**

Para detalhes em como use este recurso leia o meu artigo: [C# - Usando InputBox](http://www.google.com.br/url?sa=t&rct=j&q=&esrc=s&source=web&cd=1&ved=0CC4QFjAA&url=http%3A%2F%2Fwww.macoratti.net%2F10%2F10%2Fc_inbox.htm&ei=mUf_UqOcDpPJkAfY_4GoBA&usg=AFQjCNEjQza3nFv7Cxq_Gykf0CjJwVIZQw&sig2=2ZIJ_R4i9Jhqofy5WLWYlA)

A criação do banco de dados é feita pelo seguinte código :

**SqlCeEngine SqlEng = new SqlCeEngine(connectionString);  
SqlEng.CreateDatabase();**

O método **CreateDataBase**() cria o banco de dados.

Após informar o nome verificamos se foi informada a extensão**.sdf** no arquivo:

**if (!Resultado.Contains(".sdf"))  
{  
     MessageBox.Show("Informe a extensão .sdf no arquivo...");  
      return;  
}**

Antes de criar o banco de dados estou verificamos se o arquivo com o nome informado já existe:

 **if (File.Exists(nomeArquivoBD))**

   ....

Se o arquivo existir realizamos a exclusão do mesmo usando o método **Delete** da classe **File**:

|  |
| --- |
| **if (File.Exists(nomeArquivoBD))  {      if (MessageBox.Show("O arquivo já existe !. Deseja excluir e criar novamente ? ", "Excluir", MessageBoxButtons.YesNo, MessageBoxIcon.Information) == DialogResult.Yes)      {            File.Delete(nomeArquivoBD);      }     else {        return;    } }** |

**2- Criar tabelas**

|  |
| --- |
| private void btnCriarTabelas\_Click(object sender, EventArgs e)  {  /\* define os parâmetros para o inputbox \*/  string Prompt = "Informe o nome da tabela a ser criada.Ex: Teste";  string Titulo = "UTILIZANDO Banco Dados";  string Resultado = Interaction.InputBox(Prompt, Titulo, "Contatos", 650, 350);  /\* verifica se o resultado é uma string vazia o que indica que foi cancelado. \*/  if (Resultado != "")  {  if (Resultado.Contains(".sdf"))  {  MessageBox.Show("Não informe a extensão .sdf no arquivo...");  return;  }  //criamos um conexão  SqlCeConnection cn = new SqlCeConnection(stringConexao());  //verifica se a conexão esta fechada e abre a conexão  if (cn.State == ConnectionState.Closed)  {  cn.Open();  }  //definimos um objeto command  SqlCeCommand cmd;  //montamos uma instrução SQL usando Create Table definindo a estrutura da tabela a ser criada  **string sql = "create table " + Resultado + "("**  **+ "Nome nvarchar (60) not null, "**  **+ "Sobrenome nvarchar (80), "**  **+ "URL nvarchar (150) )";**  **cmd = new SqlCeCommand(sql, cn);**  if (MessageBox.Show("Confirma a criação da tabela ? ", "Criar Tabela", MessageBoxButtons.YesNo,  MessageBoxIcon.Information) == DialogResult.Yes)  {  try  {  //cria a tabela no banco de dados  **cmd.ExecuteNonQuery();**  lblResultado.Text = "Tabela " + Resultado + " criada com sucesso ";  }  catch (SqlCeException sqlexception)  {  MessageBox.Show(sqlexception.Message, "Caramba.", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  catch (Exception ex)  {  MessageBox.Show(ex.Message, "Caramba.", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  finally  {  cn.Close();  }  }  else  {  return;  }  }  else  {  MessageBox.Show("A operação foi cancelada...");  }  } |

Para criar uma tabela no banco de dados SQL Server Compact 4.0 montamos uma string de conexão usando a instrução **Create Table** e definindo a estrutura da tabela.

**string sql = "create table " + Resultado + "("  
                                      + "Nome nvarchar (60) not null, "  
                                      + "Sobrenome nvarchar (80), "  
                                      + "URL nvarchar (150) )";**

O método **ExecuteNonQuery**() é quem cria a tabela no banco de dados.

**3- Carregar uma Tabela com dados**

Na rotina a seguir estamos chamando o método **CarregarLinha**() e passando os valores para preencher a tabela com dados:

|  |
| --- |
| private void btnCarregarTabela\_Click(object sender, EventArgs e)  {  try  {  CarregarLinha("julio", "Sergio", @"julio@yahoo.com.br");  CarregarLinha("João", "Paulo", @"janjan@uol.com.br");  CarregarLinha("Maria", "Gabriela", @"mary@bol.com.br");  CarregarLinha("Jose", "Carlos", @"jojo@hotmail.com");  CarregarLinha("Luis", "Marcos", @"lulu@ig.com.br");    }  catch (Exception ex)  {  MessageBox.Show(ex.Message, "Oxente.", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  } |

O método **CarregarLinha**() recebe os parâmetros e monta uma instrução SQL**insert into** para incluir os valores na tabela usando o método **ExecuteNonQuery**().

|  |
| --- |
| private void CarregarLinha(string nome, string sobrenome, string url)  {  SqlCeConnection cn = new SqlCeConnection(stringConexao());  if (cn.State == ConnectionState.Closed)  {  cn.Open();  }  SqlCeCommand cmd;  **string sql = "insert into Contatos "**  **+ "(sobrenome, nome, url) "**  **+ "values (@sobrenome, @nome, @url)";**  try  {  cmd = new SqlCeCommand(sql, cn);  cmd.Parameters.AddWithValue("@sobrenome", sobrenome);  cmd.Parameters.AddWithValue("@nome", nome);  cmd.Parameters.AddWithValue("@url", url);  **cmd.ExecuteNonQuery();**  lblResultado.Text = "Linha Incluída.";  }  catch (SqlCeException sqlexception)  {  MessageBox.Show(sqlexception.Message, "Uaí Sô.", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  catch (Exception ex)  {  MessageBox.Show(ex.Message, "Uaí Sô.", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  finally  {  cn.Close();  }  } |

a

**4- Carregar Grid**

|  |
| --- |
| private void btnCarregarGrid\_Click(object sender, EventArgs e)  {  SqlCeConnection cn = new SqlCeConnection(stringConexao());  if (cn.State==ConnectionState.Closed)  {  try  {  cn.Open();  }  catch (Exception ex)  {  MessageBox.Show(ex.Message);  }  }  try  {  // define o command par ausar a tabela e não a consulta  **SqlCeCommand cmd = new SqlCeCommand("Contatos", cn);**  **cmd.CommandType = CommandType.TableDirect;**  // Pega a tabela  **SqlCeResultSet rs = cmd.ExecuteResultSet(ResultSetOptions.Scrollable);**  // carrega o resultado no grid  dgvDados.DataSource = rs;  }  catch (SqlCeException sqlexception)  {  MessageBox.Show(sqlexception.Message, "Arre Égua.", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  catch (Exception ex)  {  MessageBox.Show(ex.Message, "Arre Égua.", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  } |

Para carregar o DataGridView - **dgvDados** - usamos a classe **SqlCeResultSet** que representa um cursor que pode ser atualizado, navegado e vinculado.

A seguir usamos o método**cmd.ExecuteResultSet** que envia um **CommandText** a uma  Connection e cria um **SqlCeResultSet** usando a enumeração **ResultSetOptions**.

**SqlCeResultSet rs = cmd.ExecuteResultSet(ResultSetOptions.Scrollable);**

Os valores possíveis para **ResultSetOptions** são:

* **None** - Nenhuma opção de ResultSet foi especificada.
* **Updatable** - O ResultSet permite atualizações.
* **Scrollable** - É possível rolar o ResultSet para frente e para trás
* **Sensitive** - O ResultSet detecta as alterações feitas à fonte de dados
* **Insensitive** - O ResultSet não detecta as alterações feitas à fonte de dados.

5- Ler Registros da tabela 

|  |
| --- |
| private void btnLerRegistros\_Click(object sender, EventArgs e)  {  SqlCeConnection cn = new SqlCeConnection(stringConexao());  if (cn.State == ConnectionState.Closed)  {  cn.Open();  }  // Monta a consulta SQL  string sql = "select sobrenome, nome from Contatos ";  try  {  **SqlCeCommand cmd = new SqlCeCommand(sql, cn);**  **cmd.CommandType = CommandType.Text;**  **SqlCeResultSet rs = cmd.ExecuteResultSet(ResultSetOptions.Scrollable);**  // se você precisa atualizar o result set então use:  // SqlCeResultSet rs = cmd.ExecuteResultSet(ResultSetOptions.Scrollable | ResultSetOptions.Updatable);  **if (rs.HasRows)**  {  int ordSobrenome = rs.GetOrdinal("sobrenome");  int ordNome = rs.GetOrdinal("nome");  // trata a saida  StringBuilder saida = new StringBuilder();  // le o primeiro registro e pega os dados  rs.ReadFirst();  saida.AppendLine(rs.GetString(ordNome) + " " + rs.GetString(ordSobrenome));  while (rs.Read())  {  saida.AppendLine(rs.GetString(ordNome) + " " + rs.GetString(ordSobrenome));  }  // defina a saida  lblResultado.Text = saida.ToString();  }  else  {  lblResultado.Text = "Nenhum registro encontrado.";  }  }  catch (SqlCeException sqlexception)  {  MessageBox.Show(sqlexception.Message, "Bah Tchê.", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  catch (Exception ex)  {  MessageBox.Show(ex.Message, "Bah Tchê.", MessageBoxButtons.OK, MessageBoxIcon.Error);  }  finally  {  cn.Close();  }  } |

No código acima estamos obtendo os dados usando um **SqlResultSet** com a opção **Scrollabe**:

**SqlCeCommand cmd = new SqlCeCommand(sql, cn);  
                 cmd.CommandType = CommandType.Text;  
                 SqlCeResultSet rs = cmd.ExecuteResultSet(ResultSetOptions.Scrollable);**

A percorremos o resultado e montamos uma **StringBuilder** com as informações para sobrenome e nome da tabela e exibimos no controle Label (lblResultado) no formulário:

|  |
| --- |
| **if (rs.HasRows)    {                      int ordSobrenome = rs.GetOrdinal("sobrenome");                      int ordNome = rs.GetOrdinal("nome");                      // trata a saida                      StringBuilder saida = new StringBuilder();                      // le o primeiro registro e pega os dados                      rs.ReadFirst();                      saida.AppendLine(rs.GetString(ordNome) + " " + rs.GetString(ordSobrenome));                      while (rs.Read())                      {                          saida.AppendLine(rs.GetString(ordNome) + " " + rs.GetString(ordSobrenome));                      }                      // define a saida                      lblResultado.Text = saida.ToString();                  }                  else                  {                      lblResultado.Text = "Nenhum registro encontrado.";   }** |

Abaixo vemos uma figura do projeto em execução:

|  |
| --- |
|  |

