Microsoft Visual C# 2010 Express

Acesso a Dados através da Tecnologia ADO.NET

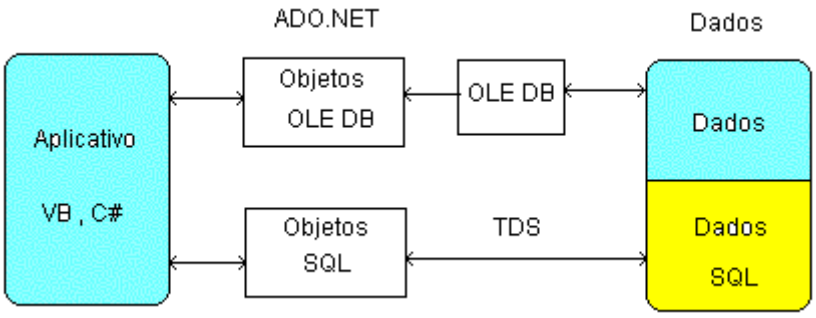
- Uma visão geral (I): Objectos Connection , Command e DataReader

Neste documento deve-se entender por fonte de dados, os dados que se pode aceder de um servidor que dispõe de um serviço de gestão de base de dados – em geral de um SGBD, por exemplo: SQL SERVER (Microsoft), MYSQL (open source), ORACLE, ou em dimensão menor mas cada vez com mais recursos o Microsoft Office Access. O objetivo principal é mostrar como utilizar os recursos de acesso a dados, do ADO.NET via código, utilizando as linguagens de programação do Microsoft Visual Studio, e neste caso através do "Visual C#.Net".

Todos estes recursos estão presentes no **Namespace** System.Data do ADO.NET.

Os componentes ADO.NET foram desenhados para tratar o acesso aos dados e a manipulação dos mesmos. Os componentes que podemos chamar de pilares do ADO.NET são : DataSet e o "provider"**.NET** que é um conjunto de componentes que inclui os objetos **Connection**, **Command**, **DataReader**, and **DataAdapter.**

O ADO.NET fornece acesso consistente a fonte de dados, como por exemplo o SQL Server, assim como a outras fontes acessíveis, via OLE DB, XML ou ODBC. A cada fonte de dados implica ter objetos apropriados, por exemplo : *OleDataAdapter ou o SqlDataAdapter (ver esquema abaixo).*



Todos os recursos ADO.NET são oferecidos através dos *Namespaces* (espaços de nomes) da biblioteca de nomes da classe .NET. (Em C# devem ser incluídos com "`using`")

- System.Data - (Dados do sistema) - contém as classes fundamentais para gerir dados como DataSet e DataRelation .
- System.Data.Common - (Dados comuns de Sistema) - Possui classes bases que são herdadas por outras classes.
- System.Data.OleDb - Possui classes usadas para realizar conexão com o "provider" OLE DB.
- System.Data.SqlClient - Possui classes para conexão com um banco de dados SQL Server via interface TDS (*Tabular Data Stream*)
- System.Data.SqlTypes - (Dados de sistema Tipos SQL) - inclui tipos adicionais de dados que não são fornecidos pelo .NET.

Em primeiro veremos o objecto **Connection.**

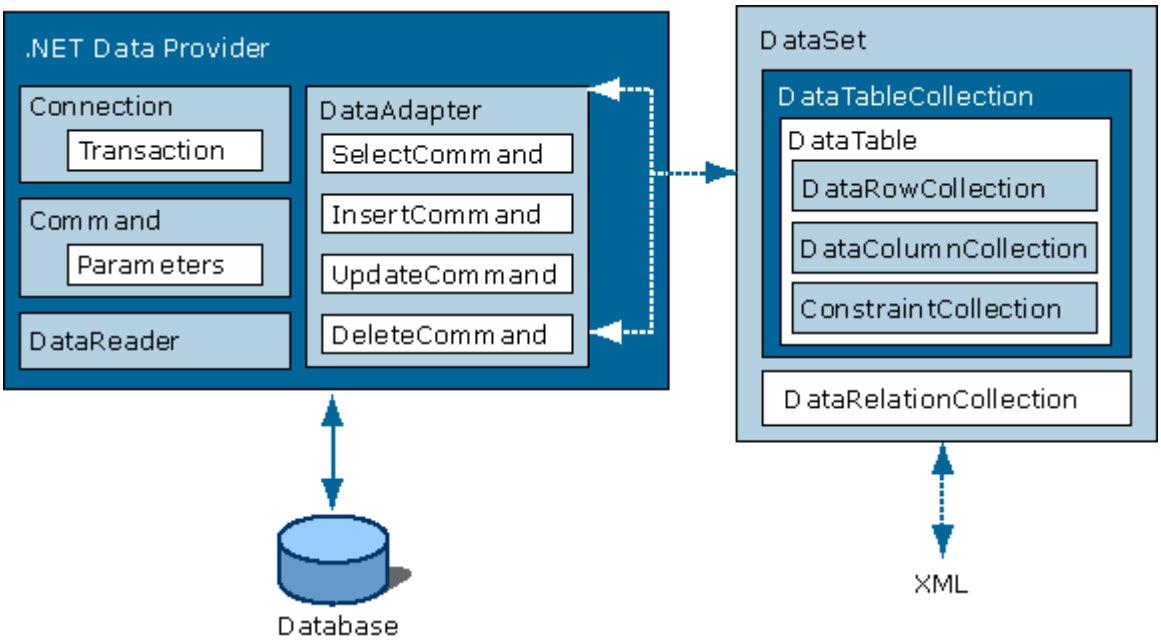
ADO.NET - Objecto Connection

O objecto Connection têm a função de gerar uma conexão com uma fonte de dados sendo portanto o objecto fundamental no acesso a dados.

Para estabelecer uma conexão com uma fonte de dados, o objeto Connection usa a propriedade ConnectionString que é a string de conexão que deverá ser informada para que a conexão seja efetivamente aberta.

Após realizada a conexão com a fonte de dados podemos usar objectos para receber e enviar dados para a fonte de dados, dentre estes objetos podemos citar : Command e DataAdapter.

Diagrama com os componentes da arquitetura ADO.NET



Como exemplo vamos criar um projeto que utiliza os objetos OleDBConnection e OleDbCommand para inserir uma linha na tabela Disciplinas da base de dados **HistoricoEscolar.accdb** Abaixo está a estrutura da tabela Disciplinas e uma visualização dos seus dados atuais:

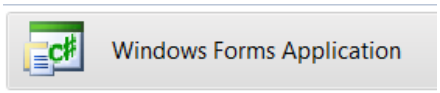
Disciplinas

Nome do campo	Tipo de dados	
ID	Numeração autom	Código Sequencial - Identificador da Disciplina
Disciplina	Texto	Descritivo do nome da disciplina
Abreviatura	Texto	Abreviatura do Nome da Disciplina
AnoCurricular	Número	Indica se é do 10º , 11º ou 12º

Disciplinas

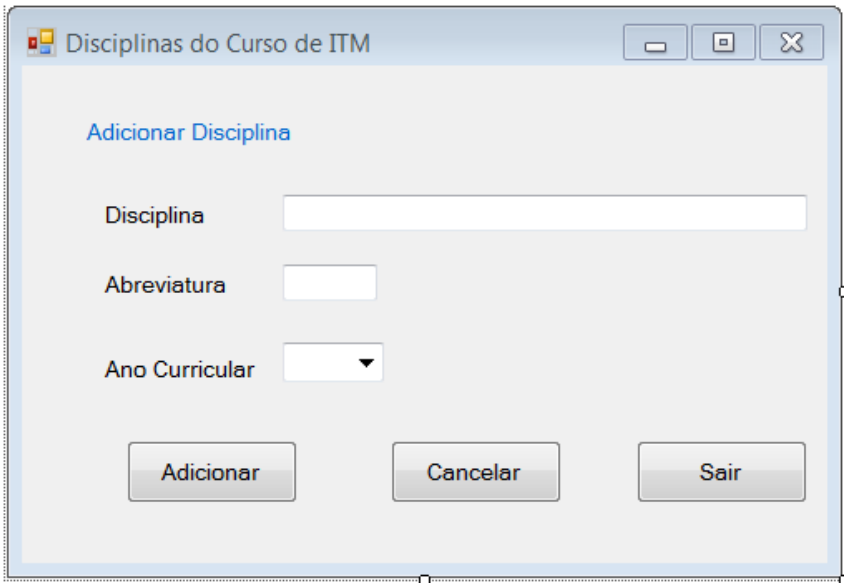
ID	Disciplina	Abreviatura	AnoCurricular
1	Tecnicas de Programação	TP	10
2	Fundamentos e Arq. Comp.	FAC	10
3	Aplicações Informáticas	AI	10

Inicie um novo projeto no Visual Studio.NET com as seguintes características

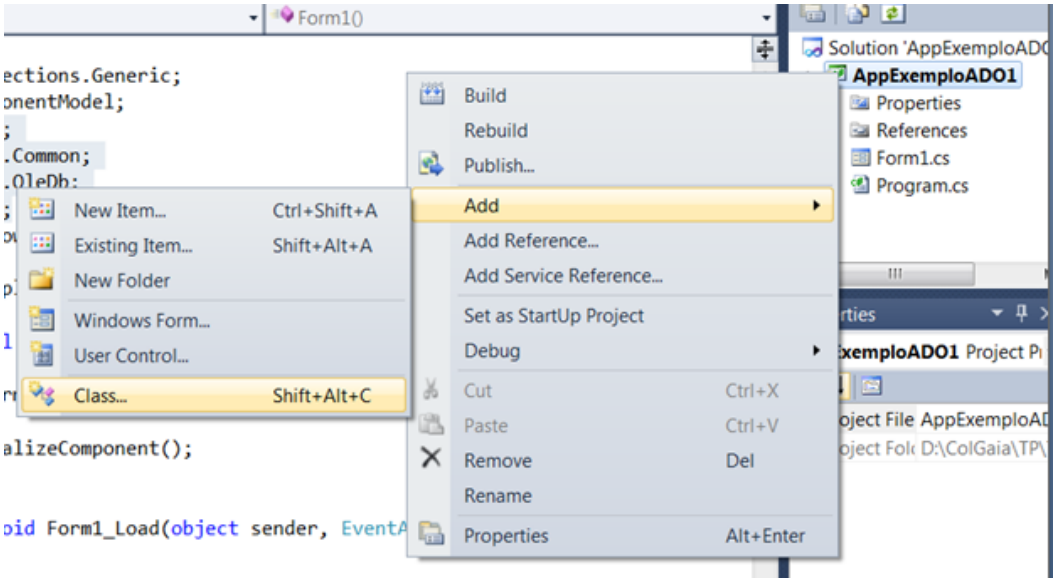


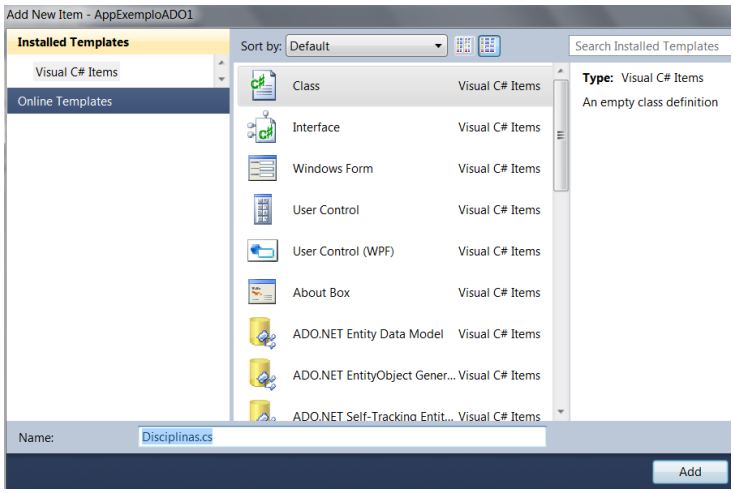
- Project Types : ***Windows Forms Application***
- Name : **AppExemploADO1**
-

1. Crie um formulário - **form1.cs** - conforme layout abaixo:



2. Terminado o interface vamos continuar a metodologia de trabalhar com classes – vamos criar o código que nos permite criar a Classe "Disciplina", a qual incluirá uma método que nos permitirá gravar os dados na base de dados citada acima. Começando por adicionar ao projeto um item do tipo Class – siga a sequência da figuras que se seguem:





2.1 Adicionar a classe Disciplina, e escrever o código com as propriedades e métodos para criar e manipular os respetivos dados e ações para um objeto desta classe:

```
//------- código do ficheiro Disciplina.cs ------------------

using System;
using System.Collections.Generic;
using System.Text;
// ---- Para permitir trabalhar com o provider "Microsoft Access"
using System.Data;
using System.Data.Common;
using System.Data.OleDb;
//-----------------------------
namespace AppExemploADO1
{
    class Disciplina
    {
        private int id;
        private string disciplina;
        private string abreviatura;
        private int anoCurricular;

        public void setID(int id)
        {
            this.id = id;
        }
        public int getid()
        {
            return id;
        }


        public void setDados(string p_disciplina, string p_abreviatura, int p_anoCurricular)
        {
            disciplina = p_disciplina;
            abreviatura = p_abreviatura;
            anoCurricular = p_anoCurricular;
        }



        public string getDisciplina()
        {
            return disciplina;
        }
```

```
        public string getAbreviatura()
        {
            return abreviatura;
        }

        public int AnoCurricular()
        {
            return anoCurricular;
        }

        // --- Método que insere um novo registo na tabela Disciplina da base de dados,
        // a partir dos dados que estarão no objecto – para este método funcionar recebe a
        // string de conexão que indica a base de dados (configurada por um objeto da classe
        // OleDbConnection propriedade ConnectionString)

        public int Adiciona_BD(string conexao)
        {
            int status = 100; // Não há erro

            OleDbConnection bd = new OleDbConnection();
            //atribui a string para a conexão a base de dados - deve ser indicado
correctamente o
            //Provider e localização da base de dados(Data Source)
            bd.ConnectionString = conexao;

            OleDbCommand cmd = bd.CreateCommand();

 string strInsert = "INSERT INTO Disciplinas (Disciplina,Abreviatura,AnoCurricular) Values (";

            if (_disciplina.Length > 0)
            {
                      strInsert = strInsert + "'" + _disciplina + "',";
            }
            else
            {
                  status = -1;
            }

             if (_abreviatura.Length > 0)
            {
                      strInsert = strInsert + "'" + _abreviatura + "',";
            }
            else
            {
                  status = -2;
            }

            if (_anoCurricular.ToString() != "" )
            {
                strInsert = strInsert + _anoCurricular.ToString() + ")";
            }

            cmd.CommandText = strInsert;

            if (status == 100)
            {
                try
                {
                          bd.Open();
                          cmd.ExecuteNonQuery();
                          bd.Close();
                }
                catch (Exception erro)
                {
```

```
                    status = -200;
                }
                  }

            return status;
        }
    }
}
```

3. Agora vamos adicionar o código que irá inserir um novo registo na base de dados no evento Click do botão "Adicionar" (btnAdicionar) conforme o descrito abaixo:

```
private void btnAdicionar_Click(object sender, EventArgs e)
        {
            string strProvider;
            int resultado;

            Disciplina oDisciplina = new Disciplina();

            strProvider = "Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=D:\\ColGaia\\TP\\Tp11\\ProjectosC#\\FichasdeTrabalhoADO\\HistoricoEscolar.accdb;P
ersist Security Info=False";

            if (txtDisciplina.Text == "")
            {
                MessageBox.Show("Dado Obrigatório", "Erro", MessageBoxButtons.OK);
                txtDisciplina.Focus();
                return;
            }

            if (txtAbreviatura.Text == "")
            {
                MessageBox.Show("Dado Obrigatório", "Erro", MessageBoxButtons.OK);
                txtAbreviatura.Focus();
                return;
            }

            if (cbxAnoCurricular.Text == "")
            {
                MessageBox.Show("Dado Obrigatório", "Erro", MessageBoxButtons.OK);
                cbxAnoCurricular.Focus();
                return;
            }

            oDisciplina.setDados(txtDisciplina.Text, txtAbreviatura.Text,
Convert.ToInt32(cbxAnoCurricular.Text));
            resultado = oDisciplina.Adiciona_BD(strProvider);
            if (resultado == 100)
            {
                MessageBox.Show("Registo adicionado com sucesso!");
                txtAbreviatura.Clear();
                txtDisciplina.Clear();
                cbxAnoCurricular.SelectedIndex = -1;
            }
            else
                MessageBox.Show("Erro " + resultado.ToString());
        }
```

ADO.NET - Objetos Command

Os objetos Command são utilizados para executar declarações SQL e procedimentos armazenados (*stored procedures*). Os métodos para realizar estas tarefas são:

- ExecuteReader - executa declarações SQL que **retornam** linhas de dados, tais como **SELECT**
- ExecuteNonQuery - executa declarações SQL **que não retornam** dados, tais como **INSERT, UPDATE , DELETE e SET**
- **ExecuteScalar -** retorna um valor único como resultado de uma função agregada: **SUM, AVG, COUNT, MAX E MIN.**

Para criar um comando deve-se ter uma conexão criada. Assim para uma base de dados m **SQL Server** devemos utilizar um objeto SqlCommand, já se usarmos providers **OLE DB** deveremos usar o objeto OleDbCommand . Vejamos um exemplo de criação de um comando (para SQL SERVER):

```
string SqlString = "Select * from Clientes Where Codigo > 100"

SqlCommand cmd = new SqlCommand(SqlString, conexao);
```

No exemplo acima, é utilizado um objeto SqlCommand onde especificamos a conexão já existente que será utilizada para seleccionar registos de uma tabela clientes onde o código seja maior que 100. Outra forma de obter o mesmo resultado:

```
string SqlString = "Select * from Clientes Where Codigo > 100"
SqlCommand cmd = new SqlCommand();

cmd.CommandText = SqlString;

cmd.Connection = conexao;
```

Se precisarmos receber e manipular os dados retornados pelos métodos acima (ExecuteReader) precisamos utilizar os objectos **DataReader** : OleDbDataReader ou SqlDataReader.

Objetos DataReader

A utilização de um objecto DataReader é uma das maneiras mais fáceis para ler os dados retornados pelo objeto Command . Eles permitem aceder e percorrer os registos recebido através do DataReader, no modo de somente leitura e somente para frente - **forward-only**. **Não oferecem acesso desconectado** e **não permitem alterar ou atualizar a fonte de dados original sendo utilizados para obter rapidamente dados de leitura**. Apresenta poucos recursos mas seu desempenho é muito melhor do que o oferecido pelos DataSet (a ver mais adiante).

As propriedades e métodos mais usadas dos objetos DataReader são :

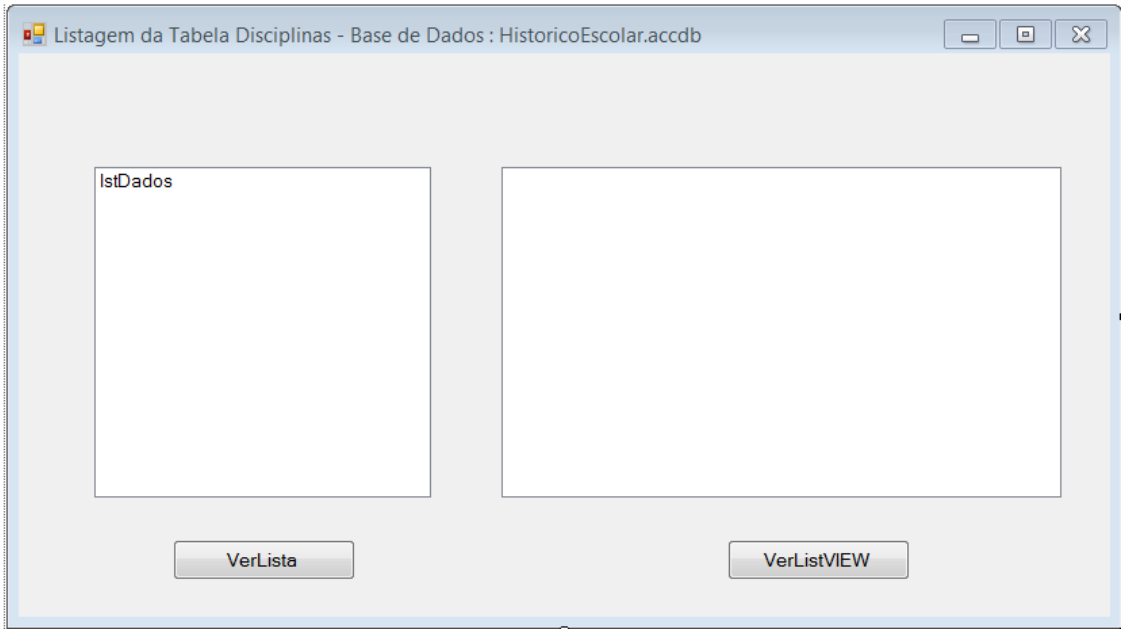
1. FieldCount - informa o **número de colunas** da linha de dados actual

2. IsClosed - Indica se o objeto DataReader esta fechado.
3. RecordsAffected - especifica o **número de linhas alteradas**, excluídas ou incluídas na execução de uma declaração SQL
4. Close - Método que fecha o objeto
5. GetName - Método que retorna o **nome** da n-ésima coluna.
6. Read - método que permite ao DataReader avançar para o próximo registo
7. IsDbNull - método que informa se a n-ésima coluna possui um valor nulo.

Para criar um objecto DataReader utilizamos o método **ExecuteReader** de um objeto Command. Abaixo um exemplo simples de como fazer isto:

```
SqlDataReader leitor = cmd.ExecuteReader()
```

Vamos mostrar um exemplo completo usando o **DataReader** para ler a tabela Disciplinas da base de dados **HistoricoEscolar.accdb** . Os dados serão exibidos por dois controlos: **Listbox e ListView.**



Código do botão VerLista : `btnListarListbox`

```
private void btnListarListbox_Click(object sender, EventArgs e)
{
    string strProvider
    OleDbConnection bd = new OleDbConnection();

    OleDbCommand cmd;
    OleDbDataReader leitor;

    strProvider = "Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=D:\\ColGaia\\TP\\Tp11\\ProjectosC#\\FichasdeTrabalhoADO\\HistoricoEscolar.accdb;Per
sist Security Info=False";

    //atribui a string para a conexão a base de dados - deve ser indicado corretamente o
    //Provider e localização da base de dados(Data Source)
    bd.ConnectionString = strProvider;

    cmd = bd.CreateCommand();
```

```
    cmd.CommandText = "Select * from Disciplinas";

    Try
     {
          bd.Open();

          leitor = cmd.ExecuteReader()

          While leitor.Read()
          {
                int i;

                for (i = 0;i <= leitor.FieldCount – 1; i++)
                            ListBox1.Items.Add(leitor[i]));
           }

           leitor.Close();
           bd.Close();
     }
     Catch (Exception erro)
      {
        MessageBox.Show(erro.Message)
      }
    }
}
```