

# **Material para prova**

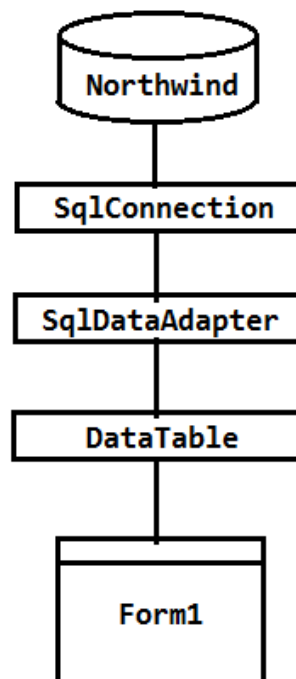
## **P1 de ILP20**

## 1. ADO.Net

**ADO.NET** (ActiveX Data Objects) é um conjunto de classes definidas pela .NET framework (localizadas no namespace System.Data) que pode ser utilizado para acessar dados armazenados numa base de dados remota.

O modelo desconectado ADO.NET utiliza dois tipos de objetos para acessar a base de dados: os objetos Dataset, que podem conter um ou mais Data Table, e os .NET Data Provider que é um conjunto de componentes que inclui os objetos Connection, Command, DataReader, and DataAdapter.

Pode ser usado para acessar base de dados para qual há um provedor específico .NET, ou, via um provedor ponte .NET, para o qual existe um provedor específico OLE DB, Driver ODBC , ou Driver JDBC. ADO.NET é algo considerado uma evolução da tecnologia ADO, mas é importante notar que importantes mudanças foram feitas entre os dois.



- **System.Data** - (Dados do sistema) - contém as classes fundamentais para gerenciar dados como DataSet e DataRelation.
  - **System.Data.OleDb** - Possui classes usadas para realizar conexão com o provedor OLE DB, que acessa o banco de dados Access.
  - **System.Data.SqlClient** - Possui classes para conexão com um banco de dados SQL Server.
- Abaixo está sendo exibido o esquema de como se conectam as camadas de objetos para o acesso do banco de dados.

## 2. SqlConnection

O objeto [Connection](#) têm a função de gerar uma conexão com um banco de dados sendo, portanto o objeto 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.

A [ConnectionString para Access](#) possui 2 propriedades importantes:

- Provider: define o tipo de banco de dados que será acessado, no caso do access ãMicrosoft.Jet.OLEDB.4.0ö.
- Data Source: Define o endereço e nome do arquivo de banco de dados.

```
string strconexao = "provider=Microsoft.jet.oledb.4.0;" +
    "data source=" + Application.StartupPath +
    "\\receitas.mdb";

conexao = new OleDbConnection(strconexao);
conexao.Open();
```

A ConnectionString para SQL SERVER possui as propriedades:

- **Initial catalog**: define qual o banco de dados que será aberto.
- **Integrated security**: define que será feita a conexão através do usuário do Windows.
- **Data Source**: Define o servidor de banco de dados que será executado.
- **User id**: define usuário para conexão.
- **Password**: define senha para a conexão.

```
string strconexao = "Data source=(local); INITIAL CATALOG =  
                    NORTHWIND; integrated security = SSPI";  
conexao = new SqlConnection(strconexao);  
conexao.Open();
```

Se o servidor for definido como local será usado o da própria máquina, pode ser informado também por IP ou nome do servidor.

Após realizada a conexão com a fonte de dados podemos usar objetos para receber e enviar dados para a fonte de dados, dentre estes objetos podemos citar: [Command](#) e [DataAdapter](#).

### 3. [SQLDataAdapter](#)

O objeto **Data Adapter** é parte integrante dos provedores gerenciados da ADO.NET, e são um conjunto de objetos usados para efetuar a comunicação entre a fonte de dados e o DataTable.

Geralmente, isto significa efetuar a leitura dos dados de um banco de dados e preencher um DataTable, efetuar a manutenção de dados e, em seguida, escrever de volta no banco de dados com

as informações atualizadas do DataTable. Na verdade, um data adapter pode mover dados entre qualquer fonte de dados e um DataTable.

Na maioria dos casos, adaptadores são configuráveis, permitindo que você defina qual dados deseja mover para o DataTable, e a partir do DataTable exibí-los no formulário.

Com frequência, isto é realizado com a utilização de sentenças SQL ou procedimentos armazenados que são invocados para ler e escrever para a fonte de dados.

- **Fill** é função que preenche o DataTable com os dados filtrados da tabela pelo DataAdapter.

Veja abaixo um exemplo de DataAdapter preenchendo um DataTable:

```
//configuração do oledbdataadapter
//define string de sql; instancia objeto; abre a conexão
string strsql = "select * from autores";
adapter = new SqlDataAdapter(strsql, conexao);

adapter.Fill(tblquestoes); //o adapter preencher o datatable
```

## 4. DataTable

O **DataTable** funciona como um conjunto de linhas e colunas na memória que armazenam os dados filtrados e descarregados pelo DataAdapter, e pode ser considerado a principal classe do ADO.NET para armazenar dados desconectados.

Como já dissemos um objeto DataTable representa uma ou mais tabelas de dados em memória. Os objetos DataTable estão contidos no objeto DataSet e/ou DataView. Abaixo temos uma relação das principais propriedades do objeto DataTable:

- **Columns** - representa as colunas da tabela através da coleção de objetos DataColumn (DataColumnCollection);
- **Rows** - representa as linhas da tabela através de uma coleção de objetos DataRow (DataRowCollection);

- **Rows.Count** - conta a quantidade de linhas do DataTable.

Veja no quadro abaixo como ler uma linha do DataTable:

```
lblcidade.Text = tblclientes.rows[linha]["coluna"].ToString();
```

## 5. DataSet.

O DataSet é a classe da biblioteca ADO.NET feita especialmente para representar um banco de dados em memória. No entanto, o *DataSet* é mais interessante porque pode conter uma coleção de objetos *DataRelation* e *DataTable*.

*DataSet* que pode representar na memória muitas tabelas. Os objetos *DataTable* são usados para representar e tratar estas tabelas ; além disto podemos criar relacionamentos entre estas tabelas através de objetos *DataRelation*. Desta forma o *DataSet* consegue ser uma representação mais próxima do banco de dados.

## 6. SqlCommand

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

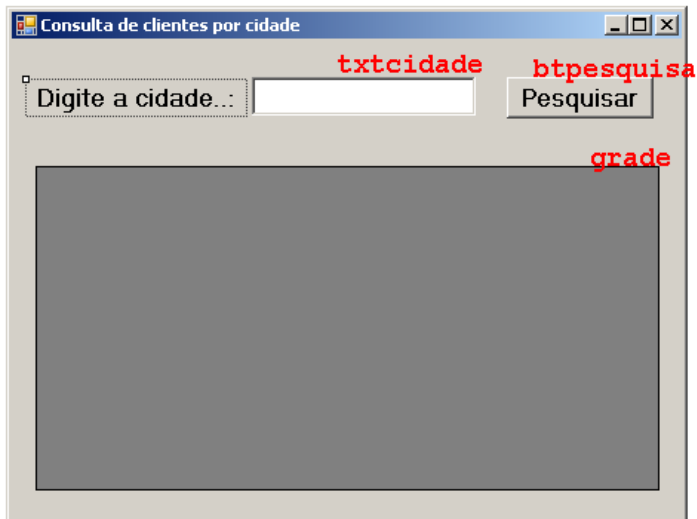
- *ExecuteNonQuery* - executa declarações SQL que não retornam dados , tais como **INSERT** , **UPDATE** , **DELETE** e **SELECT**

Para criar um comando você já deve ter uma conexão criada. Assim para um banco de dados SQL Server devemos usar um objeto [SqlCommand](#), já se usarmos provedores OLE DB deveremos usar o objeto [OLEDBCommand](#). Vejamos um exemplo de criação de um comando:

```
string sql = "insert into autores (autor,cidade,email) values ('"
            + txtautor.Text + "','"
            + txtcidade.Text + "','"
            + txtemail.Text + "')";

comando = new OleDbCommand(sql, conexao);
comando.ExecuteNonQuery();
```

7. Veja abaixo o exemplo do formulário para Consultar clientes por cidade.



```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Data.SqlClient;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace PesquisaClientes
{
    public partial class Form1 : Form
    {
        private SqlConnection conexao;
        private SqlDataAdapter adapter;
        private DataTable tblclientes;
        private string strsql, strconex;
```



```
public Form1()  
{  
    InitializeComponent();  
}  
  
private void btpesquisar_Click(object sender, EventArgs e)  
{
```

```
    //abre uma conexao com o banco de dados  
    strconex = "data source=(local);initial  
                catalog=locadora;integrated  
                security=sspi";  
    conexao = new SqlConnection(strconex);  
    conexao.Open();  
  
    //Define o que será filtrado (puxado) do banco de  
        dados  
    // define qual tabela e quais registros  
    strsql = "Select * from Clientes where cidade like '"  
                +txtcidade.Text  
                +"%'";  
    adapter = new SqlDataAdapter(strsql, conexao);  
  
    //instancia o datatable  
    tblclientes = new DataTable();  
  
    //preenche o DataTable  
    adapter.Fill(tblclientes);  
    grade.DataSource = tblclientes;
```

```
}
```

## 8. Veja abaixo o exemplo do formulário de Cadastrar Categorias.

Montar um Programa de cadastro de categorias para o banco de dados ãLocadoraã, observando as colunas da tabela abaixo:

CadCategorias


Categoria txtcategoria

Descrição txtdescricao

Valor txtvalor

btgravar btcancelar

Gravar Cancelar

Categorias	
	Codcategoria
	Categoria
	Descricao
	valor

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
```

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

```
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
namespace locadoraManha
{
    public partial class CadCategorias : Form
    {
```

```
        private SqlConnection conexao;
        private SqlCommand comando;
        private string strconex, strsql;
```

```
        public CadCategorias()
        {
            InitializeComponent();
        }
```

```
private void btgravar_Click(object sender, EventArgs e)
{
```

```
        strconex = "data source=(local);" +
                    "initial catalog = locadora; Integrated
security=SSPI";
        conexao = new SqlConnection(strconex);
        conexao.Open();
        //para inserção ou deleção use sql conexao e sql
command
        strsql = "insert into categorias
(categoria,descricao,valor)      "+
                " values ('"+txtcategoria.Text +"',
'"+txtdescricao.Text +"',"+txtvalor.Text+" )";
        comando = new SqlCommand(strsql, conexao);
        comando.ExecuteNonQuery();

        MessageBox.Show("Registro gravado com
sucesso!", "aviso",
                        MessageBoxButtons.OK,
                        MessageBoxIcon.Information);
```

```
    }
```

```
}
```