**http://www.macoratti.net/maco10.gifC# - ADO .NET para Iniciantes - I**

Este é um mini-curso sobre **ADO .NET** para iniciantes usando a linguagem **C#.**

Os requisitos mínimos para que você aproveite este mini-curso é possuir um pouco de conhecimento sobre lógica de programação.

Se você não conhece a linguagem C# pode ler os seguintes artigos :

* [**Introdução bem básica ao C#**](http://www.macoratti.net/08/08/c_bas1.htm)
* [**C# - Sintaxe e conceitos básicos**](http://www.macoratti.net/cshp_cb1.htm)

Se você já conhece VB .NET e esta querendo a aprender C# sugiro que você leia o meu artigo:

* [**VB.NET e C# - Guia rápido de referência comparativa**](http://www.macoratti.net/vbnxcshp.htm)

O material necessário para acompanhar o curso é: (*Irei utilizar o Visual C# 2008 Express Edition)*

**1-** [**Visual C# 2008 Express Edition**](http://www.microsoft.com/express/vcsharp/) **2-** [**SharpDevelop 2.2**](http://sharpdevelop.net/OpenSource/SD/Download/)[**3- SQL Server 2005 Express Edition**](http://www.microsoft.com/downloads/details.aspx?familyid=220549b5-0b07-4448-8848-dcc397514b41&displaylang=en)

Todos esses programas são gratuitos e portanto você não vai gastar um centavo neste curso.

Durante o curso vamos desenvolver uma aplicação com acesso a dados usando C# de forma que você irá fazendo e aprendendo na prática os principais conceitos básicos envolvidos.

Mas chega de papo e vamos ao que interessa...

**O que é ADO .NET**

ADO .NET é a nova tecnologia para acesso a dados da plataforma .NET estando integrada ao .NET Framework e oferecendo diversas classes que permitem realizar praticamente todas as tarefas relacionadas com o acesso e manutenção de dados.

**ADO .NET** oferece suporte a uma variedade de opções para desenvolvimento de soluções com acesso a dados que permitem a comunicação com qualquer fonte de dados, desde os já conhecidos gerenciadores de banco de dados relacionais (**SGBD**) como : **SQL Server, MySQL, FireBird, Oracle, Sybase, Access, XML**, arquivos textos, etc.

Os componentes considerados os pilares da ADO.NET são o **DataSet** e os **provedores .NET** que são um conjunto de componentes que incluem os objetos **:**

* **Connection - responsável por efetuar a conexão com o banco de dados**
* **Command - responsável por executar comandos contra o banco de dados;**
* **DataAdapter - é utilizado para preencher o objeto DataSet;**

Através da **ADO.NET** podemos acessar dados de três maneiras básicas: **OLE DB , SQL e ODBC**.

Os provedores de dados ADO .NET são livrarias de classes que permitem uma maneira comum de interagir com uma fonte específica de dados. Cada livraria possui um prefixo que indica qual provedor ela suporta. Veja abaixo os principais provedores:

|  |  |  |
| --- | --- | --- |
| **Nome do Provedor** | **API prefixo** | **Descrição** |
| ODBC Data Provider | **Odbc** | Fonte de dados com uma interface ODBC interface. Geralmente usada para banco de dados antigos; |
| OleDb Data Provider | **OleDb** | Fonte de dados que expõe uma interface OleDb interface, ou seja: Access ou Excel; |
| Oracle Data Provider | **Oracle** | Para banco de dados Oracle; |
| SQL Data Provider | **Sql** | Para interação com o Microsoft SQL Server; |

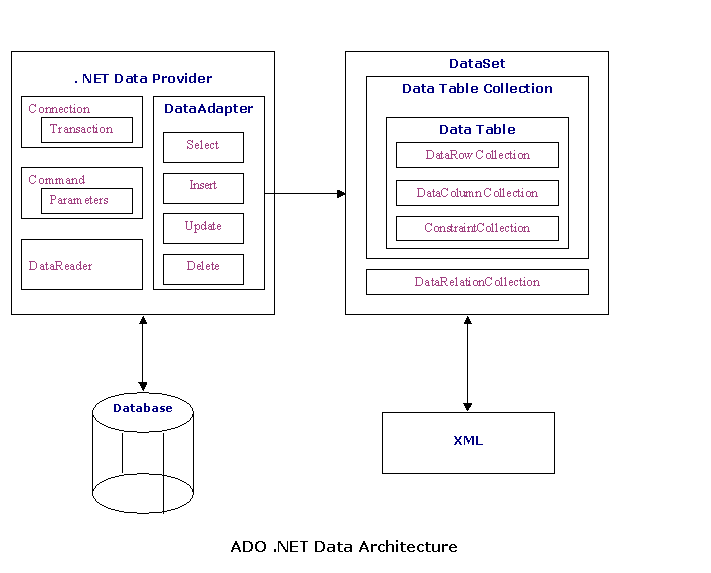
**Nota: Também existem provedores fornecidos por terceiros para MySQL, PostGreeSQL, FireBird, etc.**

Cada objeto possui uma versão para cada uma das maneiras aqui mencionadas, assim temos os objetos :

* ***OleDbConnection, OleDbCommand, OleDbDataReader, OleDataAdapter;***
* ***SqlConnection, SqlCommand, SqlDataReader, SqlDataAdapter;***
* ***OdbcConnection, OdbcCommand,etc.***

***Nota: Temos também os provedores fornecidos por terceiros como o .NET Connector para o MySQL.***

Abaixo uma figura ilustrando isto*:*



Existem duas maneiras básicas de você realizar a conexão com uma fonte de dados com ADO .NET:

**1- ) Usando um DataSet**

Este modo é conhecido como modo desconectado.

O objeto **DataSet** veio para substituir com vantagens o **objeto recordset** (ADO) e, guarda poucas similaridades com o objeto **recordset**. Enquanto o objeto *recordset* representa uma coleção de tabelas de dados O objeto DataSet representa uma cópia do banco de dados em memória.

A classe **DataSet** é membro do namespace System.Data e representa o primeiro dos dois maiores componentes da arquitetura ADO.NET os outros membros seriam os provedores Data .NET. Podemos resumir os atributos do **DataSet** como segue:

* **É baseado em XML**
* **É um conjunto de dados em cache que não esta conectado ao banco de dados**
* **É independente da fonte de dados**
* **Pode armazenar dados em múltiplas tabelas que podem ser relacionadas**
* **Armazena múltipla versões de dados para coluna e para cada linha em cada tabela**

O **DataSet** fornece as principais funcionalidades para criar aplicações para banco de dados desconectados , embora suporte também o modelo conectado através de leitores de dados (**DataReader**).

A classe **DataSet** é derivada da classe **System.ComponentModel.MarshalByValueComponent** da qual ela recebe a habilidade de ser *serializada* , incluída na caixa de ferramentas do VS.NET e visualmente desenhada em um descritor. Os principais métodos da classe **DataSet** são :

|  |  |
| --- | --- |
| **Membro** | **Descrição** |
| **Coleções** | |
| Relations | Uma coleção de relações hospedadas em um objeto **DataRelationCollection** que liga tabelas através de chaves estrangeira |
| Tables | Uma coleção de tabelas que armazena os dados atuais |
| **Métodos** | |
| **AcceptChanges** | Grava todas as alterações para o DataSet |
| **Clear** | Remove todas as linhas de todas as tabelas |
| **Clone** | Faz uma cópia da estrutura mas não os dados de um DataSet |
| **Copy** | Faz uma cópia a estrutura e os dados de um DataSet |
| **GetChanges** | Retorna uma cópia do DataSet com apenas as colunas alteradas ou aquelas que coincidem com o filtro definido em DataRowState |
| **GetXml** | Retorna uma representação exm XML dos dados |
| **GetXmlSchema** | Retorna uma representação XML da estrutura de um DataSet |
| **HasChanges** | Retorna um valor indicando que existe mudanças pendentes |
| **InferXmlSchema** | Infere a estrutura de um DataSet baseada em um arquivo ou fluxo |
| **Merge** | Mescla o DataSet com o provedor |
| **ReadXml** | Carrega um esquema XML e dados para um DataSet |
| **ReadXmlSchema** | Carrega um esquem XML para um DataSet |
| **Reset** | Reverte o DataSet ao seu estado original |
| **WriteXML** | Escreve os dados e o esquema XML para um arquivo ou fluxo de dados |
| **WriteXmlSchema** | Escreve o esquema XML para um arquivo ou fluxo |

**2- Usando um DataReader**

Os objetos DataReader é uma das maneiras mais fáceis para ler os dados retornados pelos objetos Command . Eles permitem acessar e percorrer os registros no modo de **somente leitura e somente para frente** - forward-only .

O **DataReader** **não** oferece o acesso desconectado e não permite alterar ou atualizar a fonte de dados original sendo usado para obter rapidamente dados de apenas leitura. Apresenta poucos recursos mas seu desempenho é muito melhor do que o oferecido pelos DataSet.

As propriedades e métodos mais usadas dos objetos DataReader são :

1. FieldCount - informa o número de colunas da linha de dados atual
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. Item (n) - obtêm o valor da n-ésima coluna no seu formato nativo.
5. Close - Método que fecha o objeto
6. GetName - Método que retorna o nome da n-ésima coluna.
7. Read - método que permite ao DataReader avançar para o próximo registro
8. IsDbNull - método que informa se a n-ésima coluna possui um valor nulo.

Para criar um objeto DataReader usamos o método **ExecuteReader** de um objeto Command.

**O objeto DataTable**

Ao tratar com banco de dados não podemos deixar de pensar em tabelas e o 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*)
* PrimaryKey - representa a chave primária da tabela atraves dos objetos *DataColumn*
* TableName - define o nome do objeto DataTable via coleção DatatableCollection em um objeto DataSet
* AcceptChanges - Efetiva as alterações realizadas no DataTable no banco de dados.
* NewRow - gera um novo objeto *DataRow* que representa uma linha de dados
* Copy - copia os dados e a estrutura do DataTable.
* Clear - limpa os dados de um DataTable.
* RejectChanges - ignora as alterações feitas no DataTable.

**O objeto DataView**

O DataView tem a função de permitir a ligação dos dados de uma fonte de dados com a interface do usuário através do DataBinding . Através do DataView podemos filtrar, ordenar, pesquisar e navegar pelos dados oferecendo diversas visões de um mesmo conjunto de dados ao usuário.

Usamos o DataView para mostrar uma visão dos dados contidos em um DataTable , com isto você pode ter vários **DataViews** ligados a um mesmo DataTable , sendo que cada um exibe um visão diferente dos dados.

O objeto **DataTable** possui um **DataView** padrão que é acessado através da propriedade DefaultView.

As principais propriedades do objeto DataView são :

* RowFilter - retorna uma expressão usada para filtrar os dados a serem exibidos pelo DataView.
* RowStateFilter - Define a versão dos dados que serão exibidos pelo DataView. Oferece as seguintes opções :
  + CurrendRows - linhas de dados atuais (linhas não alteradas , novas)
  + Added - linhas de dados novas.
  + Deleted - Linha excluída pelo método Delete
  + None - Nenhuma linha
  + ModifiedCurrent - linhas de dados que foram modificadas - versão atual
  + OriginalRows - linhas originais
  + Unchanged - Linhas não modificadas
  + ModifiedOriginal - linhas de dados que foram modificadas - versão original
* Count - informa o número de linhas no DataView após a aplicação dos filtros : *RowFilter e RowStateFilter*
* Item - obtêm uma linha de dados de um tabela especificada.
* Sort - define a coluna que irão ordenar o DataView e o tipo da ordenação ( *ASC - ascendente ou DESC - descendente*)
* Addnew - Inclui uma nova linha no DataView
* Table - Define qual o objeto DataTable de origem para o DataView
* Delete - exclui uma linha do DataView
* Find - Busca por uma linha no DataView

Creio que para um início já temos muita informação, na [**sequência**](http://www.macoratti.net/08/11/c_adn_2.htm) irei mostrar como realizar uma conexão com um banco de dados usando ADO .NET com C#.

Eu sei é apenas C# mas eu gosto... http://www.macoratti.net/1_positivo.gif

referências:

* [ADO.NET - Uma visão geral : Objetos Connection](http://www.macoratti.net/ado_net1.htm)
* [ADO.NET - Uma visão geral II : O objeto DataTable](http://www.macoratti.net/ado_net2.htm)
* [ADO.NET - Uma visão geral III : O objeto DataView](http://www.macoratti.net/ado_net3.htm)
* [ADO.NET - Uma visão geral IV : O objeto DataSet...](http://www.macoratti.net/ado_net4.htm)
* [DataSet x DataReader - Umna questão de desempenho ?](http://www.macoratti.net/adon_dsr.htm)
* [Criando objetos básicos de dados com ADO.NET](http://www.macoratti.net/vbn_cobd.htm)
* [ADO.NET - Boas Práticas de Programação](http://www.macoratti.net/adn_bpu1.htm)
* [C#- ADO .NET para iniciantes - II](http://www.macoratti.net/08/11/c_adn_2.htm)

José Carlos Macoratti