Acessando banco de dados de forma genérica usando o ADO.NET
Por Marcio Belo (http://www.mbelo.hpg.com.br) em 13/08/2004

Lembro-me que no DevDays de 2001 um ouvinte usou a palavra durante a sessão de perguntas e reclamou o fato de que no .NET, mais precisamente no ADO.NET, existia um namespace específico para acesso a bancos Sql Server (SqlClient) e outro genérico para outros bancos (OleDb). Esse participante citou que achava tendenciosa essa atitude da Microsoft de criar uma API específica para o banco dela e criar outra para os outros bancos. Devo dizer que concordei, embora o palestrante tenha rebatido dizendo que os outros fornecedores poderiam criar APIs específicas para os seus bancos, tirando o maior proveito possível dos recursos e performance. Hoje podemos testemunhar que foi de fato isso que ocorreu, a citar: Oracle Provider, Firebird Provider, etc.

No II DevAspNet TechConference surgiu uma colocação interessante a partir da pergunta de um ouvinte: como fazer para criar uma aplicação que acesse de forma transparente dois banco de dados de fornecedores diferentes? No ADO, bastava que eu mudasse a string de conexão. O problema que dificulta essa independência no ADO.NET se deve ao fato de que quando utilizo objetos das classes de dados diretamente, por exemplo SqlConnection, não posso usar esse mesmo objeto para acessar um banco Oracle. Precisarei, nesse caso, mudar as referências dos objetos para a classe OraConnection (usando a API da Oracle) ou OleDbConnection (usando OleDb Provider for Oracle).

Entretanto, na mesma conferência citada acima, o palestrante Dennes Torres mostrou que existe uma maneira de criar código para acessar de forma transparente qualquer tipo de banco relacional, mudando apenas a string de conexão da aplicação. Obviamente, para permitir essa indepêndencia, deve-se usar comandos SQL padronizados pela norma ANSI SQL'92.

A forma de criação envolve a utilização das interfaces usadas para implementar as classes específicas. Dessa forma, podemos referenciar a interface ao invés de referenciar uma classe específica para acesso ao banco.

Por exemplo, ao invés de criar um objeto da classe SqlConnection, crio uma referência para a interface IDbConnection, do namespace System.Data. Essa interface é implementada por todas as classes específicas, dessa forma consigo ter uma referência genêrica que funcionará com qualquer instância específica de conexão para um determinado banco de dados. Veja o exemplo:

```
Dim cn as IDbConnection
```

Após a criação dessa referência, podemos instanciar, dinâmicamente, o objeto de conexão que desejamos.

Para instanciar uma conexão Sql Server:

```
cn=New SqlClient.SqlConnection
```

Para instanciar uma conexão Oracle:

```
cn=New OleDb.OleDbConnection
```

O que vimos acima é uma referência genêrica para um objeto de conexão (que implementa a interface IDbConnection). Para usar os outros objetos necessários para interagir com o banco usamos de forma correspondente a interface adequada (todas definidas na namespace System.Data).Exemplos:

```
Dim cmd as IDbCommand

Dim dr as IDataReader

Dim da as IDbDataAdapter
```

Dim tr as IDbTransaction

Para exemplificar essa situação, fiz um pequeno código ASP.NET em VB usando o Web Matrix. Nessa aplicação, dependendo a opção selecionada, o aplicativo conecta o banco Northwind do Access ou do SqlServer, sem nenhuma alteração no código. Seria muito simples adicionar o Oracle e qualquer outro banco de bancos.

Arquivo GenAdoNet.aspx

```
<%@ Page Language="VB" %>
<script runat="server">
Sub btnPopula_Click(sender As Object, e As EventArgs)
    ' Instancia objetos dinamicamente
    Dim cn As System.Data.IDbConnection=GetConnection()
    Dim da As System.Data.IDbDataAdapter=GetDataAdapter(cn)

    ' Executa data adapter genérico e exibe na grid
    Dim ds As System.Data.DataSet=New System.Data.DataSet()
    da.Fill(ds)
    dgCustomers.DataSource=ds.Tables(0)
    dgCustomers.DataBind()
End Sub

Private Function GetConnection() as System.Data.IDbConnection
    if rblTipo.SelectedItem.Value="Access" then
    return New System.Data.OleDb.OleDbConnection( _
    "Provider=Microsoft.Jet.OLEDB.4.0;Password=;User ID=Admin;" & _
    "Data Source=C:\Arquivos de Programas\Microsoft Visual Studio\" & _
    "VB98\NWIND.MDB")
    else if rblTipo.SelectedItem.Value="SqlServer" then
    return New System.Data.SqlClient.SqlConnection("Data Source=.;" & _
        "Initial Catalog=Northwind;Integrated Security=True")
    end if
End Function

Private Function GetDataAdapter(ByRef cn as System.Data.IDbConnection) _
    as System.Data.IDbDataAdapter
    dim sql as String="select * from Customers"
    if rblTipo.SelectedItem.Value="Access" then
    return New System.Data.OleDb.OleDbDataAdapter(sql,cn)
    else if rblTipo.SelectedItem.Value="SqlServer" then
    return New System.Data.SqlClient.SqlDataAdapter(sql,cn)
    end if
End Function
</script>

<html>
<head>
</head>
<body>
<form runat="server">
<p>
Tipo de Banco:
<asp:RadioButtonList id="rblTipo" runat="server" Width="162px" RepeatDirection="Horizontal">
<asp:ListItem Value="Access" Selected="True">Access</asp:ListItem>
<asp:ListItem Value="SqlServer">SqlServer</asp:ListItem>
</asp:RadioButtonList>
<asp:Button id="btnPopula" onclick="btnPopula_Click" runat="server" Text="Popula
Grid"></asp:Button>
</p>
<p>
<asp:DataGrid id="dgCustomers" runat="server" EnableViewState="False"></asp:DataGrid>
</p>
</form>
</body>
</html>
```