عنوان: **معرفی Microsoft.Data.dll یا WebMatrix.Data.dll**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۷/۱۵ ۱۴:۲۵:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: ADO.NET

مایکروسافت اخیرا علاوه بر تکمیل ORM های خود مانند LINQ to SQL و همچنین Entity framework ، لایه دیگری را نیز بر روی ADO.NET جهت کسانی که به هر دلیلی دوست ندارند با ORMs کار کنند و از نوشتن کوئری‌های مستقیم SQL لذت می‌برند، ارائه داده است که Microsoft.Data library نام دارد و از قابلیت‌های جدید زبان سی شارپ مانند واژه کلیدی dynamic استفاده می‌کند.

در ادامه قصد داریم جهت بررسی توانایی‌های این کتابخانه از بانک اطلاعاتی معروف Northwind استفاده کنیم. این بانک اطلاعاتی را از اینجا می‌توانید دریافت کنید.

مراحل استفاده از Microsoft.Data library:
الف) این اسمبلی جدید به همراه پروژه WebMatrix ارائه شده است. بنابراین ابتدا باید آن‌را دریافت کنید: +
لازم به ذکر است که این کتابخانه اخیرا به WebMatrix.Data.dll تغییر نام یافته است. (اگر وب را جستجو کنید فقط به Microsoft.Data.dll اشاره شده است)

ب) پس از نصب، ارجاعی را از اسمبلی WebMatrix.Data.dll به پروژه خود اضافه نمائید. این اسمبلی در صفحه‌ی Add References ظاهر نمی‌شود و باید کامپیوتر خود را برای یافتن آن جستجو کنید که عموما در آدرس زیر قرار دارد:

```
C:\Program Files\Microsoft ASP.NET\ASP.NET Web Pages\v1.0\Assemblies\WebMatrix.Data.dll
```

ج) اتصال به بانک اطلاعاتی
پیش فرض اصلی این کتابخانه بانک اطلاعاتی SQL Server CE است. بنابراین اگر قصد استفاده از پروایدرهای دیگری را دارید باید به صورت صریح آن‌را ذکر نمائید:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
<appSettings>
 <add key="systemData:defaultProvider" value="System.Data.SqlClient" />
</appSettings>
<connectionStrings>
 <add name="Northwind"
     connectionString="Data Source=(local);Integrated Security = true;Initial Catalog=Northwind"
     providerName="System.Data.SqlClient" />
</connectionStrings>
</configuration>
```

این تعاریف در فایل web.config و یا app.config برنامه وب یا ویندوزی شما قرار خواهند گرفت.

د) نحوه‌ی تعریف کوئری‌ها و دریافت اطلاعات

```
using System;
using WebMatrix.Data;

namespace TestMicrosoftDataLibrary
{
 class Program
 {
     static void Main(string[] args)
     {
```

```
        getProducts();

        Console.Read();
        Console.WriteLine("Press a key ...");
    }

    private static void getProducts()
    {
        using (var db = Database.Open("Northwind"))
        {
            foreach (var product in db.Query("select * from products where UnitsInStock < @0", 20))
            {
                Console.WriteLine(product.ProductName + " " + product.UnitsInStock);
            }
        }
    }
 }
}
```

پس از افزودن ارجاعی به اسمبلی WebMatrix.Data و مشخص سازی رشته‌ی اتصالی به بانک اطلاعاتی، استفاده از آن جهت دریافت اطلاعات کوئری‌ها همانند چند سطر ساده‌ی فوق خواهد بود که از امکانات dynamic زبان سی شارپ 4 استفاده می‌کند؛ به این معنا که product.ProductName و product.UnitsInStock در زمان اجرا مورد ارزیابی قرار خواهند گرفت.
همچنین نکته‌ی مهم دیگر آن نحوه‌ی تعریف پارامتر در آن است (همان 0@ ذکر شده) که نسبت به ADO.NET کلاسیک به شدت ساده شده‌است (و نوشتن کوئری‌های امن و SQL Injection safe را تسهیل می‌کند).
در اینجا Database.Open کار گشودن name ذکر شده در فایل کانفیگ برنامه را انجام خواهد داد. اگر بخواهید این تعاریف را در کدهای خود قرار دهید (که اصلا توصیه نمی‌شود)، می‌توان از متد Database.OpenConnectionString استفاده نمود.

یا مثالی دیگر: استفاده از LINQ حین تعریف کوئری‌ها:

```
private static void getCustomerFax()
{
        using (var db = Database.Open("Northwind"))
        {
            var product = db.Query("SELECT * FROM [Customers] WHERE City=@0",
"Paris").FirstOrDefault();
            if (product != null)
                Console.WriteLine(product.Fax);
            else
                Console.WriteLine("not found.");
        }
}
```

ه) اجرای کوئری‌ها بر روی بانک اطلاعاتی

```
private static void ExecQuery()
{
        using (var db = Database.Open("Northwind"))
        {
           int affectedRecords = db.Execute("UPDATE [Customers] SET fax = fax + '*' WHERE  City = @0",
"Paris");
           Console.WriteLine("Affected records: {0}", affectedRecords);
        }
}
```

با استفاده از متد Execute آن می‌توان کوئری‌های دلخواه خود را بر روی بانک اطلاعاتی اجرا کرد. خروجی آن تعداد رکورد تغییر کرده است.

و) نحوه‌ی اجرای یک رویه ذخیره شده و نمایش خروجی آن

```
private static void ExecSPShowResult()
{
```

```
        using (var db = Database.Open("Northwind"))
        {
            var customer = db.Query("exec CustOrderHist @0", "ALFKI").FirstOrDefault();
            if (customer != null)
            {
                Console.WriteLine(customer.ProductName);
            }
        }
}
```

در این مثال رویه ذخیره شده CustOrderHist در بانک اطلاعاتی Northwind اجرا گردیده و سپس اولین خروجی آن نمایش داده شده است.

ز) اجرای یک تابع و نمایش خروجی آن

```
private static void useFuncs()
{
        using (var db = Database.Open("Northwind"))
        {
            var query = db.Query("SELECT dbo.FN_GET_CATEGORY_TREE(@0) as Rec1", 3);
            foreach(var tree in query)
            {
                Console.WriteLine(tree.Rec1);
            }
        }
}
```

در اینجا تابع FN_GET_CATEGORY_TREE موجود در بانک اطلاعاتی Northwind انتخاب گردیده و سپس خروجی آن به کمک یک نام مستعار (برای مثال Rec1) نمایش داده شده است.

سؤال : آیا WebMatrix.Data.dll بهتر است یا استفاده از ORMs ؟

در اینجا چون از قابلیت‌های داینامیک زبان سی شارپ 4 استفاده می‌شود، کامپایلر درکی از اشیاء خروجی و خواص آن‌ها برای مثال tree.Rec1 (در مثال آخر) ندارد و تنها در زمان اجرا است که مشخص می‌شود آیا یک چنین ستونی در خروجی کوئری وجود داشته است یا خیر. اما حین استفاده از ORMs این طور نیست و Schema یک بانک اطلاعاتی پیشتر از طریق نگاشت‌های جداول به اشیاء دات نتی، به کامپایلر معرفی می‌شوند و همین امر سبب می‌شود تا اگر ساختار بانک اطلاعاتی تغییر کرد، پیش از اجرای برنامه و در حین کامپایل بتوان مشکلات را دقیقا مشاهده نمود و سپس برطرف کرد.
ولی در کل استفاده از این کتابخانه نسبت به ADO.NET کلاسیک بسیار ساده‌تر بوده، می‌توان اشیاء و خواص آن‌ها را مطابق نام جداول و فیلدهای بانک اطلاعاتی تعریف کرد و همچنین تعریف پارامترها و برنامه نویسی امن نیز در آن بسیار ساده‌تر شده است.

برای مطالعه بیشتر:
Introduction to Microsoft.Data.dll

نظرات خوانندگان

نویسنده: سامان نام نیک
تاریخ: ۱۳۸۹/۰۷/۱۷ ۱۳:۳۳:۴۵

سلام
خیلی مفید بود
من که مدت هاس به دلیل استفاده از linq دیگه از ado.net استفاده نمیکنم
ولی اگه روزی بخوام استفاده کنم قطعا از کتابخانه فوق استفاده می کنم راستی فک کنم به نسبت ado.net performance بالاتری داشته باشه
دیگه از دست datatable , reader, adpater, ... خلاص میشیم
نظر شما چیه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۷/۱۷ ۱۴:۱۵:۴۷

بله. طراحی ADO.NET مربوط به دات نت یک است و از هیچکدام از پیشرفت‌های اخیر بدیهی است که استفاده نمی‌کند. به همین جهت است که در این کتابخانه ترکیبی از LINQ و قابلیت‌های dynamic زبان سی شارپ 4 را مشاهده می‌کنید.

نویسنده: Meysam
تاریخ: ۱۳۸۹/۰۷/۱۷ ۲۲:۲۶:۴۱

EF یا LINQ خودشون رو ADO.Net کار میکنن

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۷/۱۸ ۰۰:۴۳:۵۵

بله. همانطور که در مقدمه بحث عنوان شد، WebMatrix.Data.dll هم لایه‌ای است روی ADO.NET . مابقی هم به همین صورت؛ به این جهت که از پیشرفت‌های زبان‌های دات نتی استفاده کنند. زمانیکه ADO.NET ارائه شد نه Generics وجود داشت، نه LINQ نه قابلیت‌های پویای زبان و نه ...

نویسنده: وحید نصیری
تاریخ: ۱۳۸۹/۰۷/۲۲ ۰۱:۲۹:۰۵

کم کم داره از این دست پروژه‌ها زیاد میشه. دقیقا بر همین اساس و ایده استفاده از قابلیت‌های پویای زبان:
Kynetic ORM: An ORM without configuration using C# 4.0 Dynamics, Generics and Reflection

عنوان: **استفاده از DbProviderFactory**
نویسنده: فرهاد فرهمندخواه
تاریخ: ۱۳۹۱/۰۵/۱۹ ۱۹:۳۲
آدرس: www.dotnettips.info
برچسب‌ها: ADO.NET

استفاده از DbProviderFactory امکان اتصال به دیتابیس‌های مختلف با یک کد واحد را برای شما فراهم می‌سازد،بطوریکه اگر بخواهید برنامه ای بنویسید که قابلیت اتصال به Oracle و SqlServer و دیگر دیتابیس‌ها را داشته باشد، استفاده از DbProviderFactory ، کار شما را تسهیل می‌نماید.

DbProviderFactory در .Net Framework 2.0 ارائه شده است.برای درک و چگونگی استفاده از DBProviderFactory مثالی را بررسی می‌نماییم.
ابتدا کد زیر را درون یک فرم کپی نمایید:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Windows.Forms;
using System.Data.Common;

namespace DBFactory
{
    public partial class Form1 : Form
    {
        private string _MySQLProvider = "MySql.Data.MySqlClient";
        private string _SQLProvider="System.Data.SqlClient";
        private string _OracleProvider ="System.Data.OracleClient";
        private DbProviderFactory _DbProviderFactory;
        private DbConnection _DbConnection = null;
        private DbCommand _DbCommand = null;
        private DbDataAdapter _DbDataAdapter = null;

        public Form1()
        {
            InitializeComponent();
        }

        private void Form1_Load(object sender, EventArgs e)
        {
            try
            {

             string _SQLconnectionstring = "Integrated Security=SSPI;Persist Security
Info=False;Initial Catalog=Test;Data Source=FARHAD-PC";
             string _Oracleconnectionstring = "Data Source=ServiceName;User
Id=Username;Password=Password";

            _DbProviderFactory = DbProviderFactories.GetFactory(_SQLProvider);
            _DbConnection = _DbProviderFactory.CreateConnection();
            _DbConnection.ConnectionString = _SQLconnectionstring;

            _DbConnection.Open();

            if (_DbConnection.State == ConnectionState.Closed)
            {

                MessageBox.Show("اتصال با دیتابیس برقرار نشده است");
            }
            else
            {
                MessageBox.Show("اتصال با دیتابیس با موفقیت بر قرار شده است");
            }
            }
            catch (System.Exception excep)
            {
                MessageBox.Show(excep.Message.ToString());
            }
```

```
        }
    }
}
```

برای استفاد از DBProviderFactory می‌بایست از فضای نامی System.Data.Common استفاده نمایید. بعد از اعلان کلاس فرم تعدادی آبجکت تعریف شده است، که سه آبجکت ابتدایی آن، بیانگر Provider دیتابیس‌های MySQL،SQLSERVER و Oracle می‌باشد:

```
private string _MySQLProvider = "MySql.Data.MySqlClient";
 private string _SQLProvider="System.Data.SqlClient";
 private string _OracleProvider ="System.Data.OracleClient";
```

Providerهای بیان شده، جهت استفاده DBFactory برای تشخیص نوع Database می‌باشد، تا بتواند آبجکت‌های مربوط به دیتابیس را ایجاد و در اختیار برنامه نویس قرار دهد. در این مثال ارتباط با دیتابیس SQLSERVER را امتحان می‌کنیم. بنابراین خواهیم داشت:

```
_DbProviderFactory = DbProviderFactories.GetFactory("System.Data.SqlClient");
```

در کد بالا، Provider، دیتابیس SQLSERVER به DbProviderFactory به عنوان ورودی داده شده است، بنابراین آبجکتهای مربوط به دیتابیس SQL Server ایجاد و در اختیار شما قرار می‌گیرد.

اگر به نام فضای نامی System.Data.Common توجه نمایید،از کلمه Common استفاده شده است و منظور این است که تمامی کلاسهایی را که این فضای نامی ارائه می‌دهد، در هر دیتابیسی قابل استفاده می‌باشد. برای تشخیص، کلاسهای مربوط به این فضای نامی نیز در ابتدای نام آنها از دو حرف DB استفاده شده است. تمامی کلاسهای زیر در فضای نامی System.Data.Common قابل ارائه و استفاده می‌باشد:

```
DbCommand
DbCommandBuilder
DbConnection
DbDataAdapter
DbDataReader
DbException
DbParameter
DbTransaction
```

جهت اطلاع: ممکن است سئوالی در ذهن شما ایجاد شود که دات نت چگونه براساس نام Provider نوع دیتابیس را تشخیص می‌دهد؟
جواب: زمانی که دیتابیس‌های مختلف روی سیستم شما نصب می‌شود، Providerهای مربوط به هر دیتابیس درون فایل Machine.config که مربوط به دات نت میباشد، درج می‌شود. و دات نت براساس اطلاعات مربوط به همین فایل آبجکت‌های دیتابیس را ایجاد می‌نماید.

امیدوارم مطلب فوق مفید واقع شود.

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۱۹ ۱۹:۵۸

من توصیه می‌کنم که ADO.NET رو به شکل خام آن فراموش کنید. این نوع روش‌ها هرچند پایه و اساس تمام ORMهای نوشته شده هستند، اما فقط ابتدای کار را به شما نشان می‌دهند. واقعیت این است که سوئیچ کردن بین بانک‌های اطلاعاتی مختلف نیاز به تولید SQL قابل فهم برای آن موتور خاص را نیز دارد. اینجا است که ORMها در وقت شما صرفه جویی می‌کنند. شما کوئری LINQ می‌نویسید اما در پشت صحنه بر اساس پروایدر مورد استفاده، این کوئری LINQ به معادل SQL قابل فهم برای بانک اطلاعاتی مورد نظر ترجمه می‌شود. خیلی از توابع هستند که در بانک‌های اطلاعاتی مختلف تفاوت می‌کنند و این SQL ایی که مورد بحث است ... در عمل آنچنان استاندارد نیست. توابع تاریخ در SQLite با SQL Server فرق می‌کند. نوع‌های داده‌ای این‌ها عموما تطابق ندارد و مسایل دیگر. ORMها می‌توانند این مسایل را به خوبی مدیریت کنند بدون اینکه شما آنچنان درگیر این جزئیات شوید.

نویسنده: فرهاد فرهمندخواه
تاریخ: ۱۳۹۱/۰۵/۱۹ ۲۱:۱۲

سلام
با تشکر از توصیه شما
تا حدودی با نظر شما موافق هستم، اگر بخواهیم با امکانات جدید مایکروسافت نرم افزاری ایجاد نماییم. قطعا، روش بیان شده ضرورتی ندارد، اما برای پروژه هایی که با امکانات قدیمی‌تر نوشته شده اند و بدلایلی امکان بازنویسی آنها وجود، ندارد، و از طرفی میبایست با دیتابیس‌های مختلف نیز کار کند، روش فوق می‌تواند مفید باشد،
در مورد اینکه دیتابیس‌ها با هم متفاوت می‌باشند، نیز با شما موافقم، حتی معتقدم که Provider ی را که مایکروسافت برای Oracle ارائه داده است،در مقایسه با Provider شرکت Oracle بسیار ضعیف‌تر عمل می‌نماید، به عنون مثال در جاهایی که مدت زمان درج اطلاعات زیادی بصورت Batch بسیار اهمیت دارد،Provider، شرکت Oracle برای دیتابیس Oracle سازگارتر و کاراتر میباشد.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۱/۰۵/۱۹ ۲۱:۳۱

- اگر به هر دلیلی مجبور هستید که از دات نت 2 استفاده کنید، NHibernate می‌تونه پیشنهاد خوبی باشه و نسخه مخصوص دات نت 2 هم دارد (به آرشیو قدیمی آن سایت مراجعه کنید). (پایه زبان فعلی جاوا از خیلی از جهات شبیه به دات نت 2 است)
- مایکروسافت کلا توسعه پروایدر ADO.NET مخصوص اوراکل را رسما متوقف کرده و خود اوراکل الان داره این کار رو ادامه می‌ده . خلاصه از پروایدر مایکروسافت برای کار با اوراکل استفاده نکنید.