

ADO.NET – بخش دوم: آشنایی با DataSet

در مقاله قبلی با مقدمات استفاده از ADO.NET و همچنین با DataReader آشنا شدیم و با استفاده از روش Connected اطلاعات مورد نیاز را از Database خواندیم. در این مقاله با روش Disconnected آشنا می شویم.

یک پروژه از نوع Windows Application و با نام TestDataSet بسازید. یک Button و یک ListBox روی فرم قرار دهید و نام آنها را به btnGetPublishers و lstPublishers تغییر دهید. Text کنترل Button را به Get Publishers تغییر دهید. از بخش Events رویداد Click را برای Button انتخاب نمایید. کد زیر را درون این رویداد بنویسید:

```
private void btnGetPublishers_Click(object sender, EventArgs e)
{
    string strConnectionString = @"...";
    string strSQL = "SELECT * FROM Publishers ORDER BY Pub_Name";
    System.Data.SqlClient.SqlConnection oConnection = null;
    System.Data.SqlClient.SqlDataAdapter oDataAdapter = null;
    System.Data.DataSet oDataSet = null;
    oConnection = new
        System.Data.SqlClient.SqlConnection(strConnectionString);
    oDataSet = new DataSet();
    oDataAdapter = new
        System.Data.SqlClient.SqlDataAdapter(strSQL, oConnection);

    try
    {
        oDataAdapter.Fill(oDataSet, "Publishers");
        foreach (System.Data.DataRow oDataRow in
            oDataSet.Tables["Publishers"].Rows)
        {
            lstPublishers.Items.Add(oDataRow[1].ToString());
        }
    }
    catch (System.Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.Message);
    }
    finally
    {
        if (oDataSet != null)
        {
            oDataSet.Dispose();
            oDataSet = null;
        }

        if (oDataAdapter != null)
        {
            oDataAdapter.Dispose();
            oDataAdapter = null;
        }

        if (oConnection != null)
        {
            oConnection.Dispose();
            oConnection = null;
        }
    }
}
```

}

در این کد ابتدا دو متغیر رشته ای برای نگهداری `ConnectionString` و دستور `SQL` تعریف کرده ایم. توجه کنید که در اینجا `ConnectionString` برابر ... قرار داده شده است که شما باید با توجه به `Database` خودتان مقدار آن را تعیین کنید.

نکته

در کلیه مثالهای این سری مقالات از بانک اطلاعاتی Pubs که بطور پیش فرض در SQL Server 2000 وجود دارد استفاده می شود. در صورتی که از SQL Server 2005 استفاده می کنید یا این Database را خودتان باید بسازید و با آن را Import کنید.

پس از آن کدهای مربوط به تعریف اشیایی از نوع `SqlConnection` و `SqlDataAdapter` نوشته شده است. با `SqlConnection` در مقاله قبلی آشنا شده اید. `SqlDataAdapter` رابط بین `Database` و `DataSet` می باشد در واقع وظیفه خواندن اطلاعات از `Database` و قرار دادن آن در `DataSet` و همچنین `Update` کردن `Database` از روی `DataSet` وظیفه `DataAdapter` می باشد. می توان آن را به پلی بین `DataSet` و `DataAdapter` برای ذخیره و بازیابی اطلاعات تشبیه کرد. وقتی یک `SqlDataAdapter` یک `DataSet` را پر می کند `Table` ها و فیلدهای مورد نیاز را در `DataSet` خواهد ساخت.

پس از آن یک شی از نوع `DataSet` تعریف شده است. `DataSet` نسخه ای از داده های ذخیره شده در حافظه می باشد.

نکته

هر `DataSet` حاوی یک یا بیشتر `DataTable` می باشد و هر `DataTable` حاوی صفر یا بیشتر `DataRow` می باشد. هر `DataTable` معادل یک جدول و هر `DataRow` بیانگر یک رکورد است.

برای تخصیص حافظه مربوط به `DataAdapter` آن را با دو پارامتر `strSQL`(دستور `SQL` مورد نظر) و `oConnection` (شی `Connection` تعریف شده) `New` می کنیم.

از آنجایی که بخشهای مربوط به کار با `Database` از نقاط بحرانی کد محسوب می شوند آنها را در `Try..Catch..finally` قرار می دهیم تا اگر خطایی رخ داد بتوانیم آن را کنترل کنیم. (بزودی در وبلاگ درباره نحوه کنترل خطاها صحبت خواهیم کرد)

در بخش `Try` متد `Fill` از شی `oDataAdapter` را فراخوانی کرده ایم. با فراخوانی این متد ساختار لازم در `DataSet` ساخته شده (`DataTable` و فیلدهای مورد نیاز) و داده ها از `Database` خوانده و در آن کپی می شوند. پارامترهایی که به متد `Fill` ارسال کرده ایم عبارتند از نام `DataSet` و نام `Table`. ممکن است این سوال برای شما پیش آمده باشد که با توجه به اینکه نام `Table` در دستور `SQL` ذکر شده است پس ارسال آن به متد `Fill` چه لزومی دارد؟ جواب: `DataSet` از این نام برای نامگذاری `DataTable` موجود در خود استفاده می کند. برای روشن شدن این موضوع به خط بعد توجه کنید:

```
foreach (System.Data.DataRow oDataRow in oDataSet.Tables["Publishers"].Rows)
```

ما برای دسترسی به رکورد های موجود در DataTable درون DataSet از نام DataTable یعنی Publishers استفاده کرده ایم. در اینجا با استفاده از یک حلقه foreach رکوردهای (Row های) موجود در DataTable با نام Publishers را تک تک مرور کرده و مقدار فیلد دوم آن را (با اندیس یک) در ListBox اضافه می کنیم.

در بخش Catch در صورت رخدادن هرگونه خطایی پیغام آن را نمایش می دهیم. در بخش finally کلیه اشیایی را که ساخته ایم با فراخوانی متد Dispose از بین می بریم. حالا برنامه را اجرا نموده و Button را کلیک نمایید.

تا اینجا ما فقط با استفاده از DataAdapter اطلاعات را از Database بازیابی نموده و آن را در DataSet ذخیره کرده ایم اما ما باید توانایی اضافه نمودن اطلاعات جدید به Database و تغییر و حذف اطلاعات موجود را هم داشته باشیم.

یک Button به فرم اضافه کنید و نام آن را به btnAddPublisher و Text آن را به Add a Publisher تغییر دهید. در بخش Events رویداد Click مربوط به این Button را انتخاب می کنیم و کد زیر را برای آن می نویسیم:

```
private void btnAddPublisher_Click(object sender, EventArgs e)
{
    string strConnectionString = @"...";
    string strSQL = "SELECT * FROM Publishers ORDER BY Pub_Name";
    System.Data.SqlClient.SqlConnection oConnection = null;
    System.Data.SqlClient.SqlDataAdapter oDataAdapter = null;
    System.Data.SqlClient.SqlCommandBuilder oCommandBuilder = null;
    System.Data.DataSet oDataSet = null;

    oConnection = new
        System.Data.SqlClient.SqlConnection(strConnectionString);
    oDataSet = new DataSet();
    oDataAdapter = new
        System.Data.SqlClient.SqlDataAdapter(strSQL, oConnection);
    oCommandBuilder =
        new System.Data.SqlClient.SqlCommandBuilder(oDataAdapter);
    try
    {
        oDataAdapter.Fill(oDataSet, "Publishers");
        System.Data.DataRow oDataRow =
            oDataSet.Tables["Publishers"].NewRow();
        oDataRow["pub_id"] = "9998";
        oDataRow["pub_name"] = "CSharpBlog";
        oDataRow["city"] = "Tehran";
        oDataRow["country"] = "Iran";
        oDataSet.Tables["Publishers"].Rows.Add(oDataRow);

        oDataAdapter.Update(oDataSet, "Publishers");
        foreach (System.Data.DataRow oCurDataRow in
            oDataSet.Tables["Publishers"].Rows)
            oCurDataRow.AcceptChanges();
    }
    catch (System.Exception ex)
    {
        System.Windows.Forms.MessageBox.Show(ex.Message);
    }
}
```

```
finally
{
    if (oDataSet != null)
    {
        oDataSet.Dispose();
        oDataSet = null;
    }

    if (oCommandBuilder != null)
    {
        oCommandBuilder.Dispose();
        oCommandBuilder = null;
    }

    if (oDataAdapter != null)
    {
        oDataAdapter.Dispose();
        oDataAdapter = null;
    }

    if (oConnection != null)
    {
        oConnection.Dispose();
        oConnection = null;
    }
}
}
```

قسمت عمده ای از کد شبیه به کد قبلی می باشد. در کد جدید یک شی جدید از نوع SqlCommandBuilder تعریف شده است. برای آشنایی با SqlCommandBuilder باید با SqlDataAdapter بیشتر آشنا شویم. هر SqlDataAdapter چهار نوع دستور SQL دارد:

- Select Command: دستور SQL برای بازیابی داده ها از Database
- InsertCommand: دستور SQL برای ذخیره داده های جدید در Database
- UpdateCommand: دستور SQL برای بروز نمودن داده های Database از روی داده های تغییر یافته در DataSet
- DeleteCommand: دستور SQL برای حذف داده ها از Database از روی داده های حذف شده در DataSet

به نظر کار کمی مشکل شد اما اینطور نیست. مایکروسافت برای راحتی کار برنامه نویسان کلاسی به نام SqlCommandBuilder را در اختیار آنها قرار داده است. همانطور که حدس زده اید از این کلاس برای ساخت دستورات SQL مورد نیاز استفاده می شود. از آنجا که SqlCommandBuilder بصورت هوشمندانه اقدام به ساخت دستورات SQL مورد نیاز می نماید فقط کافی است DataSet مورد نظر را به آن معرفی کنیم!!! که این کار را در زمان New کردن SqlCommandBuilder انجام داده ایم:

```
oCommandBuilder = new System.Data.SqlClient.SqlCommandBuilder(oDataAdapter);
```

نکته

SqlCommandBuilder بطور ذاتی دارای محدودیتهایی می باشد ولی در اغلب موارد نیاز برنامه نویسیها را رفع می کند. برای آشنایی بیشتر با آن به MSDN مراجعه کنید.

برای ایجاد یک رکورد جدید یک شی از نوع DataRow تعریف می کنیم و آن را برابر با مقدار برگشتی متد NewRow از DataTable مورد نظر که در DataSet قرار دارد، قرار می دهیم. سپس فیلدهای مورد نیاز را مقداردهی کرده و آن را به Row های موجود در DataTable اضافه می کنیم:

```
oDataSet.Tables["Publishers"].Rows.Add(oDataRow);
```

با فراخوانی متد Update از DataAdapter اطلاعات نیز Update می شود.

پس از هر فراخوانی متد Update باید از کد زیر برای ارسال اطلاعات به Database استفاده کنیم (Commints :all changes)

```
foreach (System.Data.DataRow oCurDataRow in oDataSet.Tables["Publishers"].Rows)
    oCurDataRow.AcceptChanges();
```

فراخوانی متد AcceptChanges باعث Commit شدن تغییرات خواهد شد.

برای دو حالت دیگر یعنی تغییر در داده ها و حذف داده ها نیز کدهای زیر را می نویسیم. البته در اینجا فقط کدهای مربوط به بلاک Try نوشته شده است:

کد کلید تغییر داده ها:

```
oDataAdapter.Fill(oDataSet, "Publishers");
```

```
oDataSet.Tables["Publishers"].Rows[oDataSet.Tables["Publishers"].Rows.Count -
1]["country"] = "UK";
oDataAdapter.Update(oDataSet, "Publishers");
foreach (System.Data.DataRow oCurDataRow in oDataSet.Tables["Publishers"].Rows)
    oCurDataRow.AcceptChanges();
```

کد کلید Delete:

```
oDataAdapter.Fill(oDataSet, "Publishers");
```

```
oDataSet.Tables["Publishers"].Rows[oDataSet.Tables["Publishers"].Rows.Count -
1].Delete();
oDataAdapter.Update(oDataSet, "Publishers");
oDataAdapter.Update(oDataSet.Tables["Publishers"]);
foreach (System.Data.DataRow oCurDataRow in oDataSet.Tables["Publishers"].Rows)
    oCurDataRow.AcceptChanges();
```

آدرس وبلاگ : <http://Csharpblog.blogfa.com>

Email : Csharpblog@yahoo.com

نویسنده: ققنوس

حقوق این مقاله متعلق به وبلاگ <http://Csharpblog.blogfa.com> می باشد.