فصل هشتم :
طریقه دستیابی و کار با داده ها در ASP.NET – قسمت اول

مقدمه :

VS.NET با استفاده از مجموعه ای از ابزارها و فضاهای نام که به آنها ADO.NET اطلاق می شود به دیتابیس دسترسی دارد. طریقه دستیابی به داده ها در ADO.NET تقریبا مستقل از منبع داده است و پس از اینکه Connection و ارتباط ایجاد شد جدای از نوع دیتابیس ، طرز کار و رفتار با آنها بوسیله ی یک سری از اشیاء ، خواص و روش های موجود که برای تمام آنها یکسان است صورت می گیرد.

در طی این فصل و فصول آتی نحوه ی اتصال و استفاده از بانک های اطلاعاتی با استفاده از ابزارهای جدید فراهم شده را خواهیم آموخت.

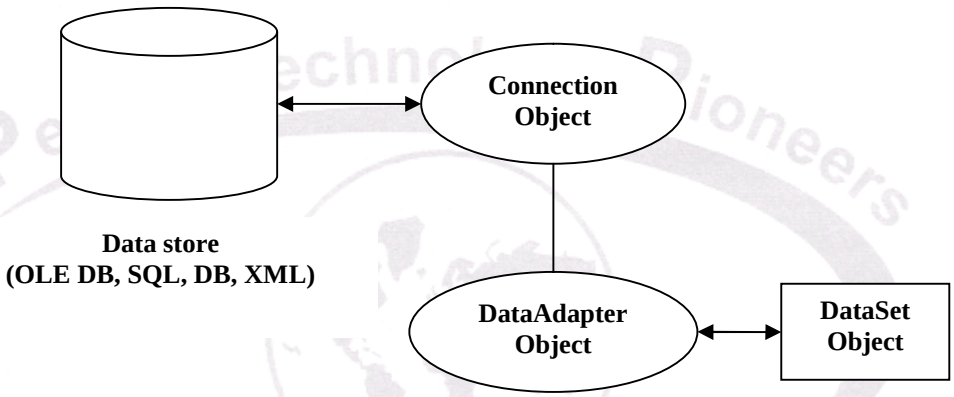
درک پایه ای از ADO.NET :

برای دستیابی به داده ها در ADO.NET سه لایه وجود دارد:

- مکان فیزیکی نگهداری داده ها : می تواند یک دیتابیس OLE ، دیتابیس SQL و یا یک فایل XML باشد .

- فراهم کننده ی داده : این مرحله شامل شیء Connection و اشیاء Command است که نمایش دهنده ی درون حافظه ای داده ها هستند.
- Data Set : نمایش دهنده ی درون حافظه ای جداول و ارتباطات بین آنها است. پس از اینکه Data Set ایجاد شد اینکه از کجا آمده است و یا کجا ذخیره گشته است اهمیت ندارد. به این نوع معماری Disconnected هم می گویند زیرا Data Set مستقل از ذخیره داده ها است .

شکل ۱- مدل شیءایی ADO.NET را در عمل نمایش می دهد.



شکل ۱- اشیاء Connection و Command در ADO.NET .

دو نوع اتصال داده به دیتابیس در ADO.NET وجود دارد :

§ استفاده از OleDbConnection برای اتصال به پایگاه داده محلی. کانکشن از نوع پایگاه داده Ole از شیء OleDataAdaptor برای انجام دستورات و بازگشت داده استفاده می شود.
§ استفاده از SqlDbConnection برای اتصال به یک دیتابیس بر روی سرور.

ADO.NET خواص ، اشیاء و متدش را به صورت سه فضای نام که در جدول ۱ بیان شده اند، ارائه می دهد.

جدول ۱- فضاهای نام ADO.NET :

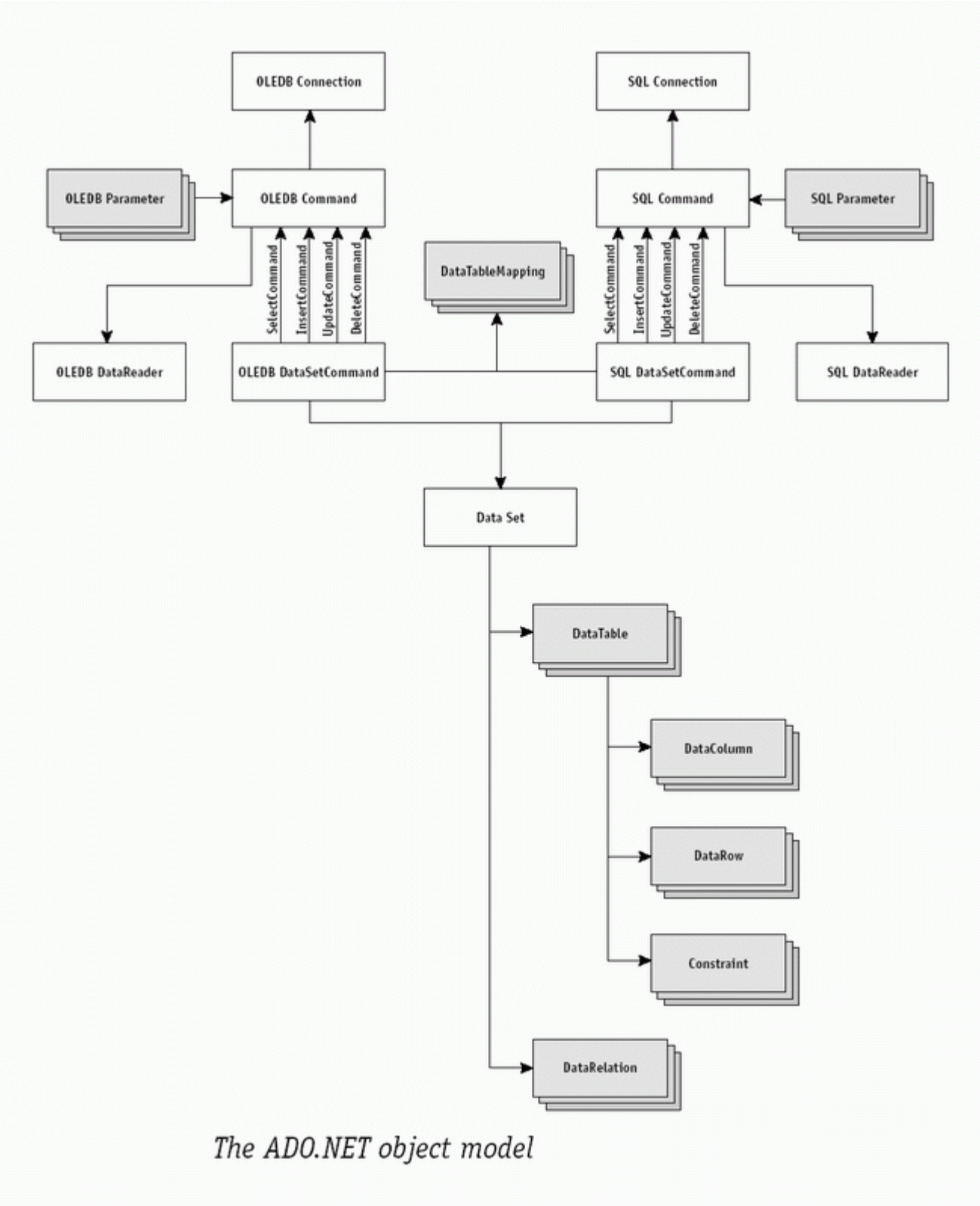
فضای نام	مواردی را که مهیا می کند
System.Data	کلاس ها، نوع ها و سرویس ها برای ایجاد و دستیابی به **Data** Sets و اشیاء مشتق شده از آن
System.Data.SqlClient	نوع ها و کلاس هایی را برای دستیابی به دیتابیس های SQL-Server
System.Data.OleDb	نوع ها و کلاس ها برای دستیابی به دیتابیس OLE

بدیهی است که مطابق جدول فوق هنگامیکه می خواهید از این فضاهای نام استفاده کنید باید آنها را به برنامه خود به صورت زیر Import کنید :

```
using System.Data;
// برای کانکشن های پایگاه داده اس کیو ال سرور
using System.Data.SqlClient;
// OLE DB برای کانکشن های پایگاه داده
using System.Data.OleDb;
```

برای دستیابی به یک دیتابیس بوسیله ی ADO.NET ، این مراحل را طی کنید :

۱- ایجاد یک کانکشن به یک بانک اطلاعاتی با استفاده از شیء کانکشن.
۲- استفاده از یک دستور برای ایجاد یک DataSet با استفاده از شیء Adapter .
۳- استفاده از شیء DataSet در کد برای نمایش داده ها یا تغییر داده ها در دیتابیس.
٤- اجرای دستورات برای به روز رسانی پایگاه داده از DataSet با استفاده از شیء Adapter .
٥- بستن کانکشن - اگر شما به صورت صریح در مرحله ی ۲ با استفاده از متد Open آنرا گشوده اید. استفاده از دستورات بدون اجرای دستور Open در ابتدا به صورت ضمنی سبب باز و بسته شدن هر کانکشن با هر در خواست می شود.



شکل ۲- مدل شیءایی ADO.NET از دیدگاهی دیگر.

در ادامه تمام این مراحل به صورت مفصلی توضیح داده خواهند شد. عموما در محصولات مایکروسافت دو روش برای کار با دیتابیس وجود دارد : ۱- استفاده از ابزارهای ویژوال در زمان طراحی . ۲- کد نویسی مستقیم .

با توجه به اینکه روش اول فقط برای تازه کاران جذاب و مفید می باشد و هیچگونه دید عمیقی را از کار ارائه نداده و با کوچکترین خطایی در برنامه رفع آن واقعا مشکل می باشد از آن استفاده نخواهد شد. برای مثال در روش اول شما با استفاده از ابزارهای ویژوال می توانید یک ConnectionString را برای اتصال به پایگاه داده ایجاد کنید ولی بدیهی است که با کدنویسی انعطاف پذیری و توانایی بیشتری وجود خواهد داشت و واقعا کسی مبحث را درست درک کرده که بتواند برای این مجموعه کد بنویسد.

نمایش داده های SQL-Server :

کنترل DataGrid برای کار با داده ها بسیار انعطاف پذیر می باشد . آن از ویژگی های پیشرفته ای مانند Paging ، ویرایش داده ها و مرتب کردن داده ها بهره می برد که مباحث مفصل تر آن و همچنین سایر کنترل های اینگونه در فصلی جداگانه ارائه خواهد گردید. در این فصل برای معرفی مفاهیم کار با داده ها صرفا از آن برای نمایش داده ها استفاده خواهیم کرد. برای نمایش داده ها حداقل از سه شیء استفاده می شود : SQLConnection ، SQLDataAdapter و SQLDataSet . شیء SQLConnection در فضای نام System.Data.SqlClient برای خلق یک کانکشن از سرور وب به بانک داده ی SQL-Server بکار برده می شود. شیء SQLDataAdapter نیز در همین فضای نام قرار داشته و بیانگر کانکشن و دستوراتی است که روی دیتابیس اجرا می شوند. شیء SQLDataSet در فضای نام System.Data موجود بوده و SQLDataAdapter را بکار می گیرد تا داده ها را از منبع داده ای SQL-Server بدست آورد.

مثال ۱

می خواهیم از بانک اطلاعاتی همراه SQL-Server به نام Pubs رکوردهای جدول Titles را بخـوانیم و در یک DataGrid نمایش دهیم .

ابتـدا فضـاهای نـام System.Data و System.Data.SqlClient را وارد کنیـد. یـک DataGrid روی صفحه قرار دهید. سپس بر روی صفحه دوبار کلیک نموده و کد زیر را داخل آن بنویسید:
یک SqlConnection برای ایجاد ارتباط با بانک اطلاعـاتی بـاز مـی کنـیم . در ConnectionString آن همانطور که ملاحظه می کنید ، نام سرور ، ID و Pass کاربر تعریف شده روی آن و نام دیتابیسی را کـه برنامه می خواهد به آن متصل شود را مشخص می کنیم.

```
SqlConnection sqlconnectionPubs = new
                SqlConnection("server=(local);uid=sa;pwd=;database=pubs");
```

سپس با ایجاد یک SQLDataAdapter و پاس کردن یک عبارت SQL بـه آن ، بـرای دریافـت داده هـا اقدام می نماییم و در ادامه یک شیء DataSet ایجاد می کنیم :

```
SqlDataAdapter sqldataadapterTitles = new
SqlDataAdapter(
    "select title, notes, price from titles where type='business'",
    sqlconnectionPubs);

DataSet datasetTitles = new DataSet();
sqldataadapterTitles.Fill(datasetTitles, "titles");
```

SQLDataAdapter را اضافه می کنیم تا DataSet را پیمایش کند. DataSource مربوط به دیتاگرید را به DataSet ایجاد شده تنظیم می کنیم و در نهایت دیتاگرید را به دیتاست متصل می نماییم.

```
DataGrid1.DataSource=datasetTitles.Tables["titles"].DefaultView;
DataGrid1.DataBind();
```

اضافه کردن داده ها به بانک اطلاعاتی SQL-Server :

پس از مطالعه ی فصول پیشین در مورد بکار گیری دستورات SQL برای مثال استفاده از Insert در اینجا به سادگی صورت خواهد گرفت .

هنگامیکه کانکشن ایجاد شد ، با استفاده از شیء Command می توان دستور SQL را برای اجرای Insert کردن داده ها مورد استفاده قرار داد و نهایتا کانکشن باید بسته شود.

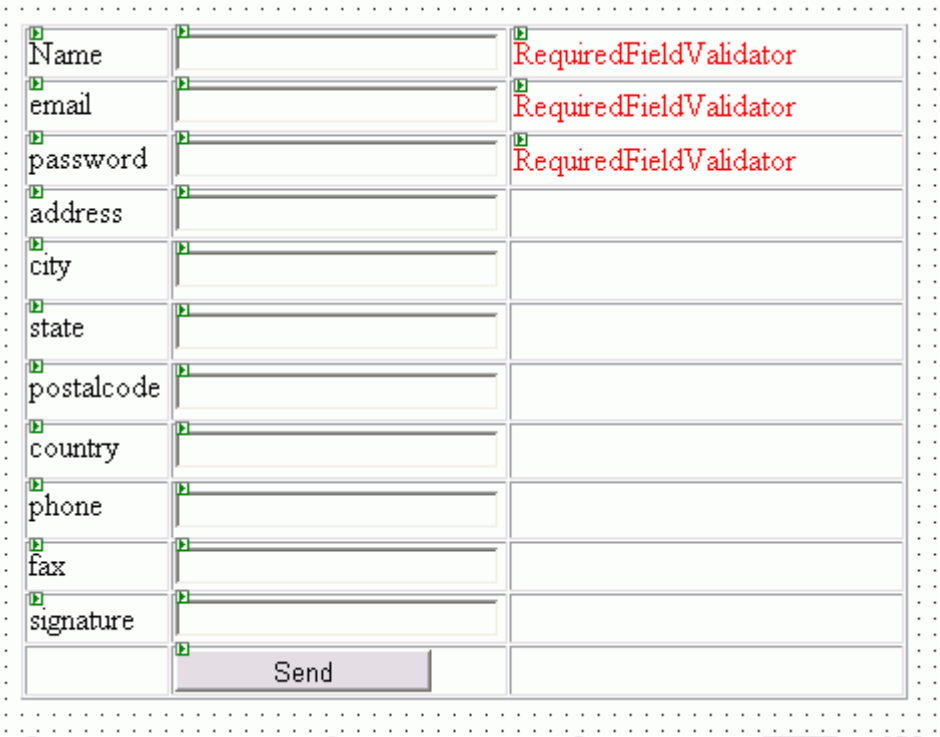
مثال ۲ :

بانک اطلاعاتی مربوط به فوروم دوره به نام AspNetForum و جدول tblUsers مربوط به ثبت نام یک کاربر جدید را به صورت زیر ایجاد کنید :

```
use AspNetForum

CREATE TABLE [dbo].[tblUsers] (
	[user_id] [int] IDENTITY (1, 1) NOT NULL ,
	[user_type_id] [smallint] NULL ,
	[user_name] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[email] [varchar] (60) COLLATE SQL_Latin1_General_CP1_CI_AS NOT NULL ,
	[password] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[addr1] [varchar] (100) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[city] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[state] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[postalcode] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[country] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[phone] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[fax] [varchar] (50) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
	[signature] [varchar] (1024) COLLATE SQL_Latin1_General_CP1_CI_AS NULL ,
) ON [PRIMARY]
GO
```

سپس فرم برنامه را به شکل زیر طراحی کنید :



شکل ۱- تصویر مثال یک در حالت طراحی . می توان از کنترل Table در قسمت HTML Controls برای نظم بخشیدن به محل قرار گیری کنترل ها استفاده کرد.

نام عناصر آن را برای درک و ساده سازی کار برنامه نویسی به نامهایی مفهوم تغییر دهید و سپس کنترل های ReqiuredFieldValidator را برای سه آیتم اول تنظیم کنید.

برای زیبا سازی بیشتر می توان روی کنترل Table کلیک راست کرد و سپس خواص آنرا انتخاب نمود و سایز حاشیه (Border) آنرا مساوی صفر قرار داد تا در زمان اجرا نامرئی باشد.

فیلد user_id به صورت Auto Increment تعریف شده است پس خود SQL-Server کار اضافه کردن و مدیریت آنرا به صورت خودکار انجام می دهد.

فیلد user_type_id را به صورت پیش فرض صفر وارد کرده و در ابتدای ورود همه را کاربر معمولی در نظر می گیریم. در قسمت مدیریت فوروم می توان این عدد را برای مثال به ۲ برای مدیر سایت تغییر داد.

اکنون با استفاده از کد زیر می توان داده ها را به آن اضافه کرد و سپس با یک DataGrid لیست کاربرها را نمایش داد. فعلا یک گرید ساده در کنار این مجموعه روی صفحه برای آزمایش کارمان اضافه می کنیم و در هنگام تکمیل نهایی برنامه آنرا در یک صفحه ی دیگر برای مثال به نام لیست کاربران نمایش خواهیم داد.

ابتدا فضای نام مربوطه به برنامه ملحق می شود :

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

مانند معمول یک Connection باز می شود :

```
SqlConnection sqlconnectionForum =  new
      SqlConnection("server=(local);uid=sa;pwd=;database=AspNetForum");
```

در ادامه شیء SQLDataAdapter را ایجاد و با یک عبارت SQL داده هایی را که می خواهیم با آنها کار کنیم انتخاب می نماییم.

```
SqlDataAdapter sqldataadapterUsers = new
    SqlDataAdapter("select * from tblUsers", sqlconnectionForum);
```

یک رشته که بیانگر دستور Insert می باشد را خلق می کنیم :

```
String insertCmd =
      "INSERT INTO tblUsers(user_type_id, user_name, email,"+
      "password, addr1, city, state, postalcode, country,"+
      "phone,fax,signature) VALUES("+
       "0,"+
       txtName.Text.Trim() +","+
       txtEmail.Text.Trim()+","+txtPassWord.Text.Trim()+","+
       txtAddress.Text.Trim() +","+
       txtCity.Text.Trim()+","+txtState.Text.Trim()+","+
       txtPostalCode.Text.Trim() +","+
      txtCountry.Text.Trim()+","+txtPhone.Text.Trim()+","+
      txtFax.Text.Trim() +","+ txtSig.Text.Trim() +  ")";
```

رشته ی ایجاد شده را برای خلق SQLCommand مورد استفاد قرار می دهیم :

```
SqlCommand sqlcommandUsers = new
SqlCommand(insertCmd, sqlconnectionForum);
```

و عاقبت این Command را Open ، Execute و می بندیم.

```
sqlcommandUsers.Connection.Open();
sqlcommandUsers.ExecuteNonQuery();
sqlcommandUsers.Connection.Close();
```

برای نمایش داده های آن هم می توان به صورت زیر عمل کرد :

```
DataSet datasetUsers = new DataSet();
sqldataadapterUsers.Fill(datasetUsers, "tblUsers");

DataGrid1.DataSource=datasetUsers.Tables["tblUsers"].DefaultView;
DataGrid1.DataBind();
```

به روز رسانی داده ها و ویرایش آنها :

با اجرای دستور SQL Update که در فصول قبلی در مورد آن توضیح داده شد می توان اینکار را انجام داد.

چون نحوه ی عملیات دقیقا همانند به عملیات Insert است می توان به مثال زیر برای تکمیل مبحث مراجعه کرد :

مثال ۳

می خواهیم در دیتابیس Pubs ، هنگامیکه Title_ID در جدول titles مساوی BU9999 است قیمت آنرا به 35.00 تغییر دهیم و سپس حاصل را در یک دیتاگرید نمایش دهیم.

ابتدا باید اضافه کردن فضای نام لازم به پروژه صورت گیرد :

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

یک کانکشن را به دیتا بیس Pubs برقرار می کنیم:

```
SqlConnection sqlconnectionPubs =new
      SqlConnection("server=(local);uid=sa;pwd=;database=pubs");
```

در ادامه شیء SQLDataAdapter را ایجاد و با یک عبارت SQL داده هایی را که می خواهیم با آنها کار کنیم انتخاب می نماییم.

```
SqlDataAdapter sqldataadapterTitles =
     new SqlDataAdapter("select title, notes,"+
     "price from titles where type='business'", sqlconnectionPubs);
```

یک رشته که بیانگر دستور Insert می باشد را خلق می کنیم و رشته ی ایجاد شده را برای خلق SQLCommand مورد استفاده قرار می دهیم :

```
String updateCmd =
        "UPDATE titles SET price = 35.00 WHERE title_id = 'BU9999'";
SqlCommand sqlcommandTitles =
        new SqlCommand(updateCmd, sqlconnectionPubs);
```

و عاقبت این Command را Open ، Execute و می بندیم.

```
sqlcommandTitles.Connection.Open();
sqlcommandTitles.ExecuteNonQuery();
sqlcommandTitles.Connection.Close();
```

برای نمایش داده های آن هم می توان به صورت زیر عمل کرد :

```
DataSet datasetTitles = new DataSet();
sqldataadapterTitles.Fill(datasetTitles, "titles");

DataGrid1.DataSource=datasetTitles.Tables["titles"].DefaultView;
DataGrid1.DataBind();
```

حذف اطلاعات از جدول :

مهمترین و معمولترین حالت حذف اطلاعات معمولا مربوط به مدیریت دیتابیس می شود و بهتر است کاربران دسترسی به یک چنین امکاناتی را نداشته باشند .

روال کار در اینجا نیز همانند اجرای سایر دستورات SQL ذکر شده می باشد و در ابتدا با شیء SQLConnection برای ایجاد یک ارتباط به دیتابیس شروع می کنیم. پس از ایجاد کانکشن یک شیء SqlCommand ایجاد می شود و یک عبارت SQL که باید روی دیتابیس اجرا شود. در ادامه ، دستور اجرا شده و کانکشن خاتمه پیدا می کند.

مثال ۴ :

در این مثال می خواهیم در جدول Titles مربوط بـه دیتـابیس Pubs ، title_id مسـاوی BU9999 را حذف کنیم و حال را در یک گرید نمایش دهیم.

به صورت خلاصه :

```
private void Page_Load(object sender, System.EventArgs e)
      {
       SqlConnection sqlconnectionPubs =  new
       SqlConnection("server=(local);uid=sa;pwd=;database=pubs");

       SqlDataAdapter sqldataadapterTitles = new
       SqlDataAdapter("select title, notes,"+
       "price from titles where type='business'", sqlconnectionPubs);

      String deleteCmd =
             "DELETE FROM titles WHERE title_id = 'BU9999'";

      SqlCommand sqlcommandTitles =
                  new SqlCommand(deleteCmd, sqlconnectionPubs);

      sqlcommandTitles.Connection.Open();
      sqlcommandTitles.ExecuteNonQuery();
      sqlcommandTitles.Connection.Close();

      DataSet datasetTitles = new DataSet();
      sqldataadapterTitles.Fill(datasetTitles, "titles");

                  DataGrid1.DataSource=datasetTitles.Tables["titles"].DefaultView;
      DataGrid1.DataBind();

      }
```

احتمالا از شکل خروجی بی روح برنامه زیاد راضی نیستید! روی دیتا گرید کلیک راست نمایید و سـپس Auto Format آنرا انتخاب کنید. در اینجا می توان انواع و اقسام حالت هـای خروجـی را بـرای دیتاگریـد انتخاب کرد.

مرتب کردن داده ها در یک بانک SQL-server :

هنگامیکه با حجم زیادی از اطلاعات سروکار داریم بسیار لازم است که قابلیت مرتب کردن داده ها نیز به بانک اطلاعاتی اضافه شوند.

برای Sort کردن داده ها ابتدا باید با استفاده از DataSet داده ها را به DataView منتقل کرد. پس از آن می توان با استفاده از خاصیت Sort ، ستون (فیلدی) را که باید این عملیات روی آن صورت گیرد را مشخص کنید.

راه دومی که عموما ساده تر از روش ذکر شده می باشد استفاده از عبارت Order By در عبارت SQL می باشد که دقیقا کار دستورات فوق را انجام می دهد .

مثال ۵ :

در این مثال می خواهیم در جدول titles و ستون title دیتابیس Pubs را مرتب کنیم.

به صورت خلاصه با توجه به اینکه توضیحات آن مانند قسمت های قبل می باشد :

```
private void Page_Load(object sender, System.EventArgs e)
   {
      SqlConnection sqlconnectionPubs = new
      SqlConnection("server=(local);uid=sa;pwd=;database=pubs");

      SqlDataAdapter sqldataadapterTitles =
        new SqlDataAdapter("select title, notes,"+
        "price from titles where type='business'", sqlconnectionPubs);

      DataSet datasetTitles = new DataSet();
      sqldataadapterTitles.Fill(datasetTitles, "titles");

      DataView dataviewTitle =
          datasetTitles.Tables["titles"].DefaultView;
      dataviewTitle.Sort = "title";

      DataGrid1.DataSource=dataviewTitle;
      DataGrid1.DataBind();
    }
```

در فصلی که گذشت با یک سری از اساسی ترین اعمالی که در هنگام برنامه نویسی بانک های اطلاعاتی سروکار داریم آشنا شدیم. این موارد تقریبا ٦٠ درصد کار نرمال را شامل می شوند و بقیه آنها یک سری موارد پیشرفته تر در مورد کار با Stored Procedures و نمایش پیشرفته داده ها است .

تمرین :

۱- یک کلاس جدید ایجاد کنید برای بایند کردن اطلاعات یک دیتابیس به دیتاگرید. این کلاس موارد زیر را به صورت خاصیت دریافت می کند : `strConnectinString` ، `strSQL` ، `strTableNameToBind` و یک متد هم به نام `bindToDataGrid` که کار نهایی اتصال را انجام خواهد داد .

۲- کلاسی جدید برای اضافه کردن داده ها به بانک SQL-Server ایجاد کنید. در این کلاس موارد زیر به صورت خاصیت دریافت می شوند : `strConnectinString` ، `strTableNameToAdd` و آرایه ای از موارد ورودی که باید اضافه شوند به نام `InputsToAdd` و در آخر با استفاده از فراخوانی متد `addRecordsToSQLdb` کاراضافه کردن رکورد به دیتابیس صورت گیرد. در این کلاس بدیهی است که عملیات ایجاد رشته ی SQL برای Insert کردن به صورت خودکار صورت خواهد گرفت. این تمرین ها در حقیقت ایده ای برای شما هستند که بعضی از کارها باید یکبار برای همیشه انجام شوند و با Encapsulation یک سری از توابع دنیا چقدر زیباتر خواهد شد!