بسم الله الرحمن الرحیم

جزوه آموزشی مباحث ویژه

زبان C# و SQL

نویسنده : استاد رجب کیانی

نمایه

* پیش‌گفتار

مواد درسی: زبان C# و SQL

انتخاب موضوعی دلخواه که با سیستم آن آشنا هستید.

گروه حداکثر 2 نفره

انتخاب یک موضوع برای پروژه عملی

مطالب هر جلسه را بر روی پروژه خود انجام می‌دهید.

برخی موضوعات پیشنهادی:

کتابخانه - حسابداری - انبارداری - فروشگاه - آژانس املاک - فروش بلیط - سیستم ورود و خروج اداره - آموزشگاه - بانک - پذیرش بیمه - دبیرخانه - آزمایشگاه - مدیریت کلاسها - آژانس تاکسی سرویس - مدیریت سی دی کلوپ - پذیرش بیمارستان - مدیریت هتل - مدیریت باشگاه - سیستم دانش آموزان مدرسه - آژانس رزرو بلیط قطار - صندوق بانک - انبارداری - کتابفروشی - مدیریت رستوران - سیستم هم اتاقی خوابگاه

هر جلسه آنچه کار می‌شود در جلسه بعد می‌خواهیم (پرسیده می شود.)

جزوه‌ای که هم اکنون در اختیار شما است، شامل شش بخش می‌باشد که برای درس مباحث ویژه و انجام پروژه‌ی آن (stdTable) لازم می‌باشد.

در این جزوه فرض بر این بوده که دانشجو با حداقل مفاهیم اصولی برنامه نویسی (روش کارکرد دستورات کنترلی، حلقه های تکرار، انواع متغیرها، متدها، پارامتر و . . .) آشنایی داشته و مفاهیم اولیه برنامه نویسی ذکر نشده است اما حتی الامکان سعی شده است مراحل ایجاد پروژه stdTable با جزئیات کافی ذکر شده و نکات مهم آنها توضیح داده شود.

بخش اول

برخی از دستورات مهم و کاربردی SQL Server

SQL به معنی زبان پرس‌وجوی ساخت‌یافته است که با آن می‌توان با پایگاه داده یا بانک اطلاعاتی خود ارتباط برقرار کرد. یک پایگاه داده (database) از یک یا چند جدول (table) تشکیل می‌شود. جدول، داده‌ها را در ساختاری شبیه به جداول معمولی (سطر و ستون) نگهداری می‌کند. ستون جدول، فیلد و هر سطر جدول یک رکورد گفته می‌شود.

ساختن دیتابیس:

```
create database databasename;
```

ساختن جدول:

در زیر جدولی به نام clstable با فیلدهایی به نام clsname به تعداد ده کاراکتر و clscode از نوع عددی ایجاد کردیم و کلید اصلی را فیلد clscode در نظر گرفتیم:

```
Create table clstable
(
clsname char(10),
clscode int,
units int,
clsun int
primary key (clscode)
```

```
);
```

جدول مربوط به دانشجویان:

```
Create table stdenttable
(
stdname char(20),
stdfam char(10),
stdcode int,
stdun char(20)
primary key (stdcode)
);
```

حذف جدول:

```
Drop table tablname;
```

مثال: حذف جدول std از بانک.

```
Drop table std;
```

درج در جدول:

```
Insert into tablename(fieldname1,fieldname2) Values ('  ',  );
```

مثال: درج رکوردی در جدول std مقدار نام را برابر احمد و کد را برابر 32 قرار دادیم.

```
INSERT INTO std  (stdname, stdcode) VALUES     (' ahmadi ', 32)
```

انتخاب و نمایش:

استخراج اطلاعات fieldname1 و fieldname2 از جدول tablename :

select fieldname1,fieldname2 from tablename where شرط ;

حذف رکورد یا رکوردهای خاص از جدول:

delete from tablename where شرط ;

مثال: حذف رکوردهایی از جدول std که نام آنها علی می‌باشد:

delete from std where stdName='ali'

ویرایش فیلدهای یک یا چند رکورد:

update tablename set field1=value1 , . . . where شرط

شرط: در قسمت شرط می‌توان شرط یا شرط‌های خاص بر اساس فیلد(های) یک یا چند جدول تعیین نمود.

مثال: در شرط زیر کسانی که نام آنها ahmad و دارای مینترم 2 هستند، انتخاب می‌شوند:

Where name='ahmad' And minterm=2

در پایان دستور می توان نتایج را براساس فیلد (فیلدهایی) خاص مرتب کرد.

Order by نام فیلدخاص

Group by نام فیلد خاص

عملگرهای مجموعه‌ای:

اجتماع Union

تفاضل Except

اشتراک Intersection

نکته: فیلتر distinct برای حذف تکرارهای خروجی بر اساس یکی از فیلدها استفاده می‌شود.

Select sname, s#,score

From stdtbl,clstbl

Where (شرط یا شرایط)

SELECT dbo.std.stdname, dbo.std.stdcode, dbo.clstable.clsname

FROM dbo.std CROSS JOIN dbo.clstable

WHERE (std.stcode =1001)

الحاق دو جدول:

SELECT dbo.std.stdname, dbo.std.stdcode, dbo.clstble.clstime

FROM dbo.std INNER JOIN

dbo.clstble ON dbo.std.clsid = dbo.clstble.clsi

که خروجی های استخراج شده از جداول به شرح زیر است.

eslami mehdi 221 11

Iman 222 11

aahngar 223 11

katanbaf 224 11

Shirali 225 11

Rakhasi 226 8

Ahmad 232 8

Navid 235 8

algorithm 1 8

software 2 11

ood 3 14

eslami mehdi 221 NULL NULL NULL NULL 2

Iman 222 NULL NULL NULL NULL 2

aahngar 223 NULL NULL NULL NULL 2

katanbaf 224 NULL NULL NULL NULL 2

Shirali 225 NULL NULL NULL NULL 2

Rakhasi 226 NULL NULL NULL NULL 1

ahmad 232 NULL NULL NULL NULL 1

navid 235 NULL NULL NULL NULL 1

بخش دوم

ایجاد بانک اطلاعاتی

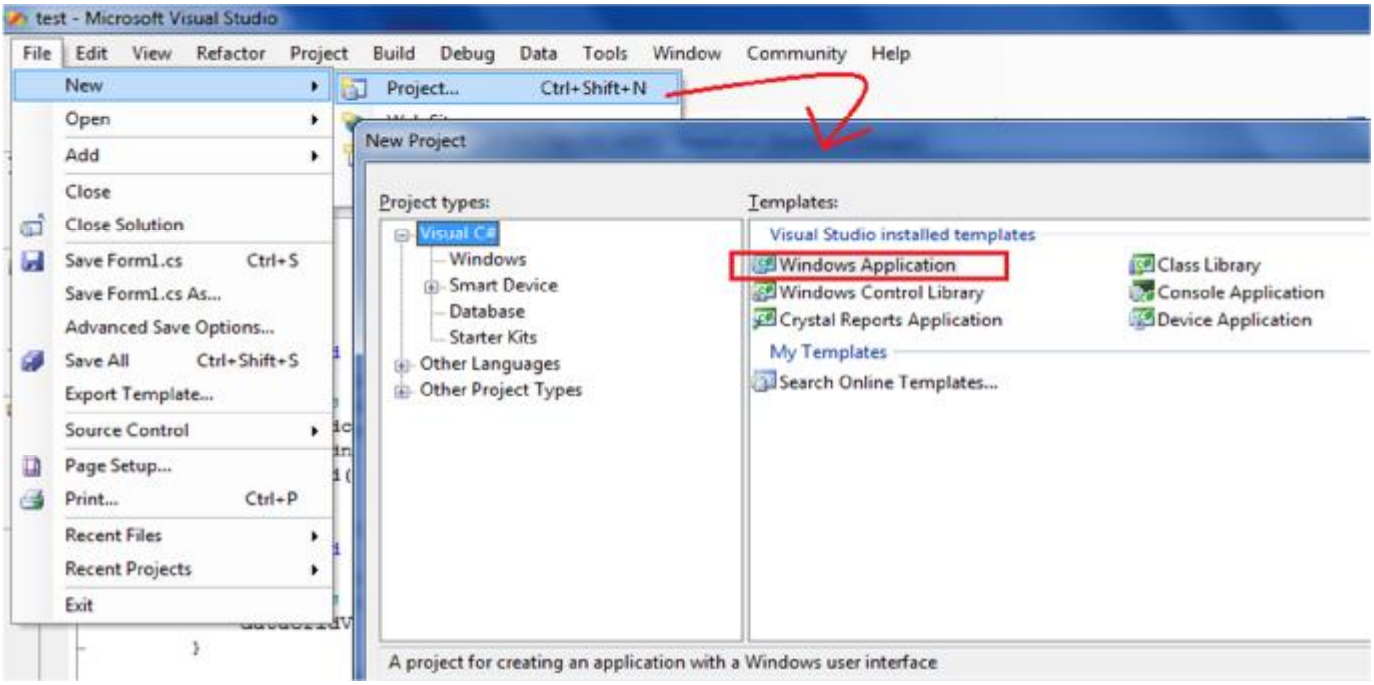
برای انجام این پروژه باید برنامه‌های زیر روی کامپیوتر شما نصب باشند:

- Microsoft Visual Studio 2005 or later OR Microsoft Visual C# Express Edition
- Microsoft SQL server 2003 or later (Express Edition)

قبل از هر چیز باید بانک اطلاعاتی خود را درست کنیم. برای این کار مراحل زیر را دنبال کنید:

1. پروژه‌ی جدید ایجاد کنید (از نوع Windows Application).

File > New > Project ... > Visual C# > Windows



2. اکنون مسیر زیر را دنبال کنید:

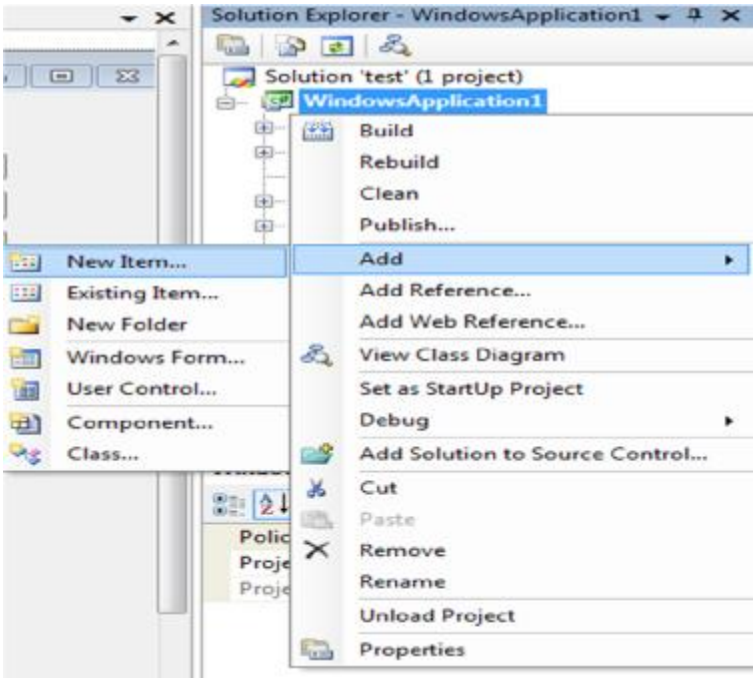
View > Solution Explorer

تا اینجا پروژه‌ی ساخته شده است ولی باید بانک خود را به پروژه اضافه کنیم.برای اینکار:

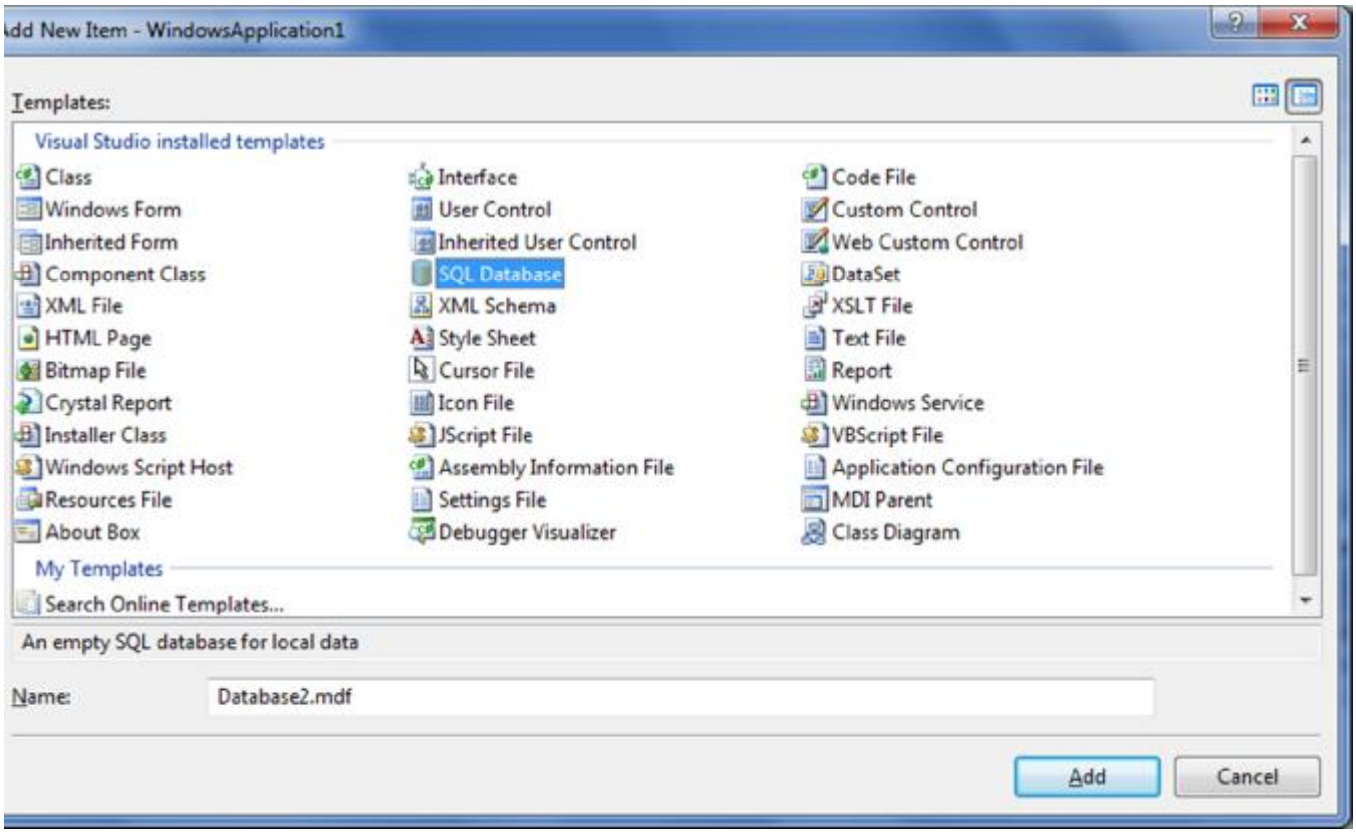
3. در پنجره Solution Explorer روی نام پروژه خود کلیک راست نمایید و مسیر زیر را دنبال کنید:

Add > New Item ...

مانند شکل زیر :



4.اکنون کادر محاوره‌ای مانند شکل زیر برای شما باز می‌شود:

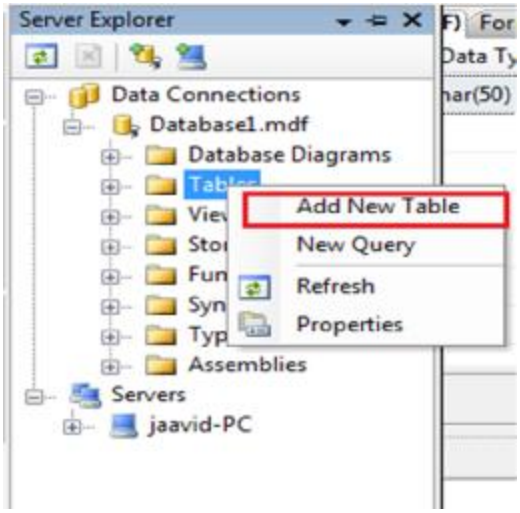


5. در این کادر محاوره‌ای گزینه‌ی SQL Database را برگزینید.در پایین این کادر محاوره‌ای نام بانک اطلاعاتی خود را وارد کنید و دکمه add را کلیک کنید.توجه شود که پسوند فایل بانک اطلاعاتی در اینجا MDF است.(Microsoft-SQL Master Database File)

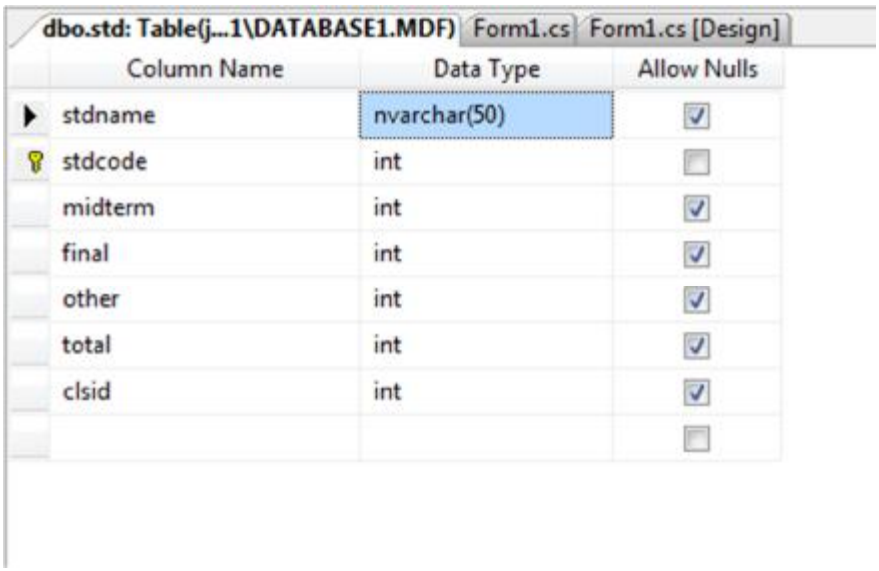
6. در اینجا اگر SQL server Express Edition بر روی کامپیوتر شما نصب باشد بانک بدون هیچ مشکلی ساخته می‌شود و آیکان آن در پنجره Solution Explorer نمایش داده می‌شود.بلافاصله بعد از اینکه بانک شما ساخته شد، کادر محاوره‌ای به نام Data Source Configuration Wizard ظاهر می‌شود، و از شما می‌خواهد که جداول و عناصر دیگر بانک اطلاعاتی خود را انتخاب کنید تا به دیتا سورس اضافه شوند.چون بانک ما فاقد هر گونه جدول است از این ویزارد صرف نظر می‌کنیم.

7. در این مرحله باید قبل از هرچیز جداول برنامه تعریف شوند.ما قصد داریم جدول std را تعریف نماییم.بصورت زیر:

View > Server Explorer > DataBase1.Mdf > Tables (Right-Click) > Add New Table



8. در این مرحله فرمی به صورت زیر باز می‌شود.که باید در آن نام فیلدهای جدول خود را تعریف کنیم:



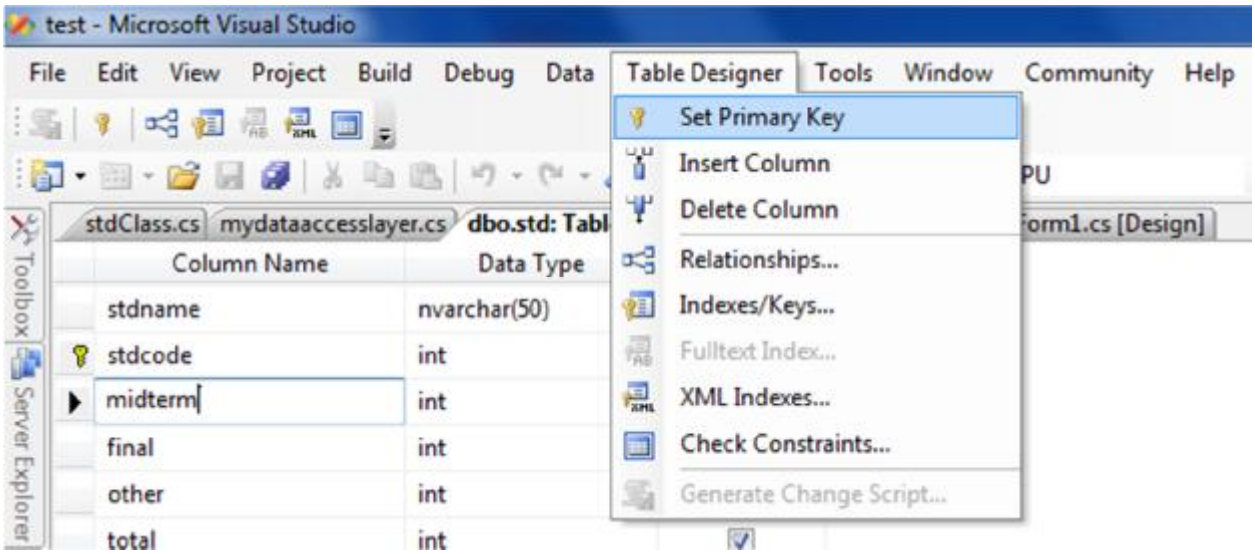
در این فرم :

ستون Column Name نام فیلد را تعیین می‌کند.

ستون Data Type نوع فیلد را تعیین می‌کند.

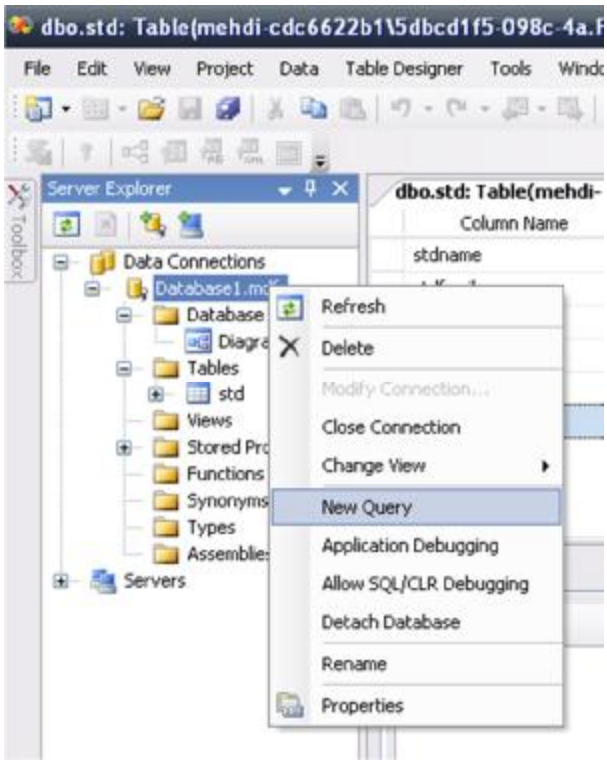
ستون Allow Nulls تعیین می‌کند که آیا فیلد تعریف شده می‌تواند مقدار Null بپذیرد یا خیر.

برای تعیین فیلد کلید ، ابتدا فیلدی که قرار است کلید شود را انتخاب کنید سپس از منوی Table Designer گزینه Set Primary Key را انتخاب نمایید.با انجام این‌کار یک آیکان کلید کوچک در کنار نام فیلد ظاهر می‌شود.



ساخت جدول از طریق Query :

برای این کار روی Database1.mdf از پنجره Server Explorer کلیک راست کرده و گزینه New Query را انتخاب می کنیم.



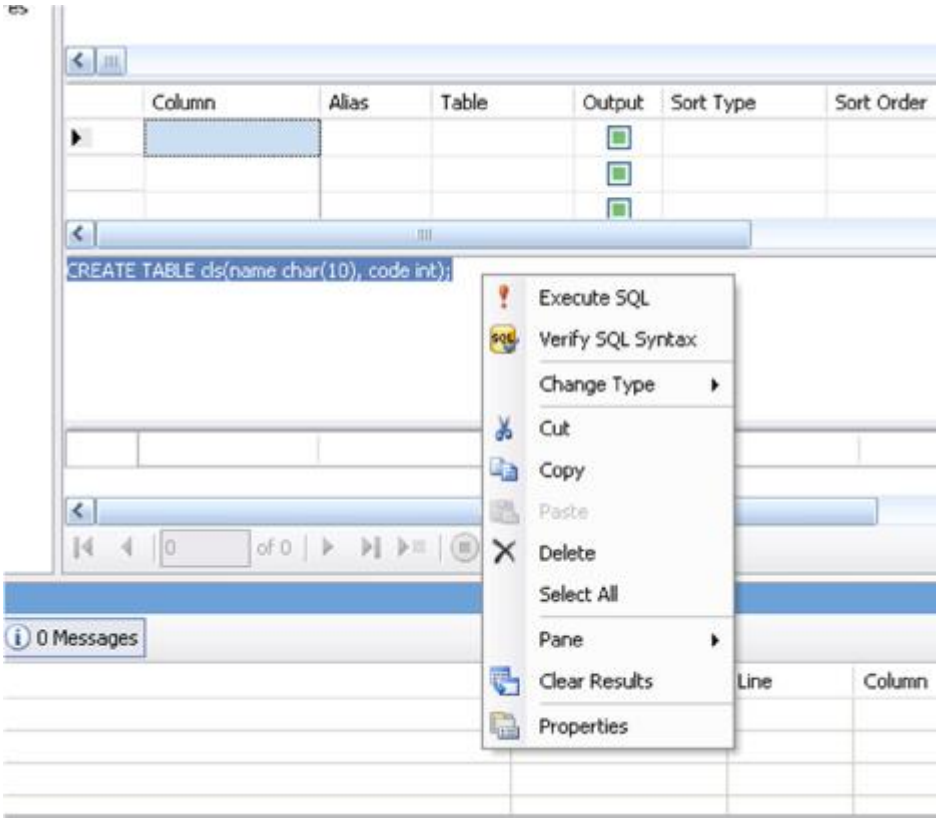
اگر پنجره انتخاب جداول باز شد آن را می‌بندیم و در پنجره جدید دستور SQL زیر را وارد می‌کنیم:

CREATE TABLE نام جدول (نام فیلد1 نوع فلید1 , نام فیلد2 نوع فیلد 2 , . . .);

مثلا:

```
CREATE TABLE cls(name char(10), code int);
```

سپس روی پنجره ویرایشگر Query راست کلیک کرده و گزینه Execute SQL را انتخاب می‌کنیم.



سپس پیغام اجرای موفقیت‌آمیز Query نمایش داده می‌شود و جدول جدید به Database ما اضـافه می‌شود.

ایجاد ارتباط بین جداول:

جداول

stdtable(stdcode,stdname,stdfam,unname),

clastable(clascode,clasname,units,unname) ,

listtable(stdcode, clascode,scor1,scor2,scor3,scor4,final)

را فرض کنید:

در جدول listtable ترکیب دو فیلد stdcode و clascode کلید اصلی می باشد.

فیلد stdcode در جدول listtable یک کلید خارجی محسوب می شود زیرا در جدول دیگری کلید اصلی است.

فیلد clascode در جدول listtable یک کلید خارجی محسوب می شود زیرا در جدول دیگری کلید اصلی است.

می‌خواهیم listtable از داده‌های stdtable و clastable استفاده کند. برای این منظور از database diagrams استفاده می‌کنیم.

رسم خط ارتباطی <- انتخاب جداول مرتبط <- RightClick ->Add New Diagram

و تایید

با این دستور کلید خارجی برای جدول ترکیبی تعریف می‌شود.

کار با کلید خارجی مفید ولی بسیار حساس می‌باشد مثلاً اگر از رکوردهایی در سایر جدول‌ها استفاده شده باشد نمی‌توان در جدول مبدا آن را حذف کرد برای انجام صحیح این کار باید ابتدا رکوردهای مربوطه را در جدول مقصد حذف کرد تا خطای کلید خارجی رخ ندهد.

بخش سوم

آشنایی مفهومی با کلاس‌های استفاده شده در پروژه

لازم به ذکر است که در این بخش فقط کلیات مفهومی کلاس‌های مورد استفاده شده، روش تعریف شیءای از نوع آن کلاس‌ها، کاربرد متدها و خاصیت‌های استفاده شده‌ی آن کلاس‌ها ذکر گردیده است ولی توضیحات تکمیلی و پیاده‌سازی آنها به طور دقیق در بخش بعد انجام خواهد شد.

هدف از این بخش تنها آشنایی مقدماتی با کلاس‌های استفاده شده و خاصیت‌ها و متدهای مهم آن‌هاست.

در تکنولوژی ADO.NET مهمترین اشیاء به دو دسته تقسیم می‌شوند:

1- اشیایی برای نگهداری و مدیریت داده‌ها: مانند کلاس‌های dataset و datatable

2- اشیایی برای ارتباط با منبع داده (datasource): مانند کلاس‌های connections و commands

این دسته اشیا را می‌توان به دو نوع تقسیم کرد:

1- اشیاء OlbDB که بیشتر برای بانک‌های غیر SQL مانند Access و Oracle استفاده می‌شود.

2- اشیاء SQL که وقتی برای MS SQL Server استفاده می‌شوند، دارای سرعت بالاتری هستند.

فضای نام System.Data :

کلاسهای اصلی ADO.net در فضای نام System.Data قرار دارد. این فضای نام خود نیز شامل چند فضای نام دیگر است که مهمترین آنها عبارتند از:

System.Data.OleDb - System.Data.SqlClient

System.Data.Odbc - System.Data.OracleClient

فضای نام System.Data.SqlClient شامل کلاس‌هایی است که برای دسترسی به بانک‌های اطلاعاتی ایجاد شده به وسیله‌ی SQL Server به کار می‌رود.

برای استفاده از این کلاس باید با استفاده از using آنرا به برنامه ضمیمه کرد.

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

برای استفاده از کلاس‌های پایه ADO.NET مانند DataSet و DataTable باید فضای نام System.Data را نیز به برنامه اضافه کنیم.

```
using System.Data;
```

DataTable چیست؟

ما می‌توانیم با استفاده از این کلاس داده‌های موجود در جدولی از بانک خود را به حافظه (RAM) منتقل کنیم تا بتوانیم روی داده‌های آن کار کنیم. در این کلاس داده‌ها در قالب سطر و ستون نگهداری می‌شوند.

نحوه تعریف شیءای به نام dt از نوع این کلاس:

```
DataTable dt = new DataTable();
```

DataSet چیست؟

دیتاست مجموعه‌ای از جداول است، که در حافظه قرار دارند. در واقع یک دیتاست شامل یک یا چند DataTable است. یک دیتاست همانند یک مخزن در حافظه عمل می‌کند و اطلاعات جدول (جدول‌ها) را در خود نگه می‌دارد.

نحوه تعریف شیءای به نام ds از نوع این کلاس:

```
DataSet  ds = new DataSet ();
```

کلاس SqlConnection:

این کلاس در فضای نام System.Data.SqlClient قرار دارد و برای ایجاد ارتباط بین برنامه و پایگاه داده مورد استفاده قرار می‌گیرد. پس از ایجاد شیای از این کلاس می‌توانیم با استفاده از خاصیت ConnectionString که یک خاصیت رشته‌ای است، مسیر و مهمترین ویژگیهای بانک اطلاعاتی موردنظر را به آن اختصاص دهیم.

برای ایجاد شیءای از این کلاس و اختصاص مسیر موردنظر از کدی مشابه قطعه کد زیر استفاده می‌کنیم. (البته روش‌های مختلفی برای این کار وجود دارد.)

```
SqlConnection  con;
con = new SqlConnection();
s = @"Data Source=(local)\SQLEXPRESS;AttachDbFilename =";
s+= @"مسیربانک اطلاعاتی\(theDBName).mdf;";
s += ";Integrated Security=True;Connect Timeout=30;User Instance=True";
con.ConnectionString=s;
```

بدیهی است که بجای .(theDBName) باید نام بانک بانک اطاعاتی خود را وارد کنیم.

کلاس SqlConnection دو متد برای شروع و اتمام ارتباط برنامه با بانک اطلاعاتی دارد.

برای باز کردن ارتباط con.Open();

برای بستن ارتباط con.Close();

کلاس SqlCommand

این کلاس نیز در فضای نام System.Data.SqlClient قرار دارد. این کلاس حاوی یک دستور SQL برای اجرا روی داده‌های بانک اطلاعاتی است. این دستور می‌تواند یک دستورSELECT،یک دستورINSERT و یا دیگر دستورات SQL باشد. حتی می‌تواند نام یک پروسیجر ذخیره شده در بانک اطلاعاتی باشد.

برای ایجاد شیءای از نوع این کلاس از دستورات زیر استفاده می‌کنیم:

```
SqlCommand com;
com = new SqlCommand();
```

خاصیت Connection

برای استفاده از شیء ای از نوع SqlCommand باید خاصیت Connection آن را تنظیم کنیم. مقدار این خاصیت را برابر مقدار شیءای که از نوع کلاسSqlConnection برای ارتباط با بانک مورد نظر ایجاد کردیم قرار می‌دهیم.

```
com.Connection=con;
```

خاصیت CommandText

این خاصیت متنی را دریافت می کند که می‌تواند حاوی یک دستور SQL (یا نام یک پروسیجر ذخیره شده) باشد که باید روی داده اجرا شود. به عنوان مثال قطعه کد زیر یک نمونه از دستور SQL که در این خاصیت قرارداده شده است را نمایش می دهد. این قطعه کد برای درج یک رکورد با مقادیر مورد نظر مورد استفاده قرار می گیرد.

```
Com.CommandText= "insert into std(stdcode,stdname) values(123,'ALi')"
```

متد ExecuteNonQuery( ):

بعد از انجام مراحل فوق برای اجرای دستور با استفاده از فراخوانی متد ExecuteNonQuery() دستور موجود در شی SqlCommand را اجرا کنید. همانطور که از نام این متد مشخص است زمانی کاربرد دارد که بخواهیم دستوری را روی بانک اجرا کرده و مانند دستور insert هیچ داده‌ای ر ا برنگرداند. ولی در صورتی که بخواهیم داده (داده‌هایی) را از بانک برگردانیم همانند زمان استفاده از دستور SELECT نمی‌توانیم برای اجرای آن از این متد استفاده کنیم.

حال روش ایجاد ارتباط و انجام عملیات به وسیله کلاس‌هایی را که در بالا ذکر شد، انجام می‌دهیم. در زیر مثال ساده‌ای از ایجاد ارتباط و درج رکورد قید شده است. تمام کلاس‌ها و خاصیت و متدهای استفاده شده مربوط به آنها در صفحات قبلی ذکر گردیده است.

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data.SqlClient;
using System.Data;
namespace WindowsApplication1
{
  class mydataaccesslayer
    {
      private SqlConnection con;
      private SqlCommand com;
      con = new SqlConnection();
      com = new SqlCommand();
      strDirectory = "پوشه ی حاوی فایل بانک" ;
      con.ConnectionString=
```

```
@"DataSource=.\SQLEXPRESS;AttachDbFilename="+strDirectory+@\DBName.mdf;Integrated Security=True;User Instance=True";
        com.Connection = con;
       com.CommandText="insert into std(stdcode,stdname) values(123,'ALi')
        ;
       con.Open();
        com.ExecuteNonQuery();
      con.Close();
   }
}
```

کلاس SqlDataAdapter

این کلاس در فضای نام System.Data قرار دارد. این کلاس همانند پلی بین جداول بانک اطلاعاتی و نیز داده های موجود در حافظه که به وسیله ی DataSet و یا DataTable نگهداری میشوند، عمل می کنند. این کلاس برای دسترسی به بانک اطلاعاتی از کلاس SqlConnection و SqlCommand استفاده می کند.

برای ایجاد شیء ای از نوع این کلاس از دستور زیر استفاده می کنیم:

```
SqlDataAdapter da;
da = new SqlDataAdapter();
```

خاصیت SelectCommand

این خاصیت در کلاس **DataAdapter** برای دریافت داده های مورد نیاز در برنامه از بانک اطلاعاتی و نیز قرار دادن آنها در **DataSet** و یا **DataTable** به کار می رود.

قبل از ااینکه بخواهید به وسیله ی **DataAdapter** اطلاعات را از بانک بگیرید باید خاصیت SelectCommand را تنظیم کنید. این خاصیت شیءای از نوع SqlCommand دریافت کرده که

این شیء مشخص می کند داده ها چگونه باید از بانک اطلاعاتی انتخاب شده و نیز چه داده هایی باید انتخاب شوند.

همانطور که در بخش قبل نیز گفته شده بود اشیایی که از کلاس SqlCommand ایجاد می شوند خود نیز خاصیت هایی دارند که قبل از استفاده باید مقدار دهی شوند یعنی خاصیت‌های Connection و CommandText .

متد Fill( )

با استفاده ا ز این متد در کلاس DataAdapter می‌توانید دستور SQL موجود در خاصیت SelectCommand را در بانک اطلاعاتی اجرا کرده، سپس داده‌های برگشتی از اجرای این دستور از درون یک DataSet در حافظه قرار دهید. این متد type و format فیلدها را نیز بازیابی می-کند.البته قبل از استفاده از این متد، باید شیءای از نوع DataSet و یا DataTable ایجاد کنید.

```
DataTable dt = new DataTable();
da.Fill(dt);
```

حال با مثالی دلیل استفاده از این کلاس و کلاس DataTable را در پروژه‌ی stdtable بیان می‌کنیم. در مثال قبل، ما تنها می‌توانستیم عملیاتی را در بانک خود اعمال کنیم حال ممکن است ما بخواهیم داده (داده هایی) را به عنوان پاسخ از بانک خود دریافت کنیم در این صورت نمی‌توانیم از متد ExecuteNonQuery مربوط به کلاس SqlCommand استفاده کنیم.

در مثال زیر با استفاده از دستور SELECT مربوط به SQL ، داده‌های جدولمان را انتخاب می‌کنیم و در یک DataTable قرار می‌دهیم.

```
using System;
```

```
using System.Collections.Generic;
using System.Text;
using System.Data.SqlClient;
using System.Data;
namespace WindowsApplication1
{
  class mydataaccesslayer
    {
      private SqlConnection con;
      private SqlCommand com;
      SqlDataAdapter da;
      con = new SqlConnection();
      com = new SqlCommand();
      strDirectory = "پوشه ی حاوی فایل بانک" ;
      con.ConnectionString=
@"DataSource=.\SQLEXPRESS;AttachDbFilename="+strDirectory+@\Database
.mdf;Integrated Security=True;User Instance=True";
       con.Open();
       com.Connection = con;
       com.CommandText= "select * from std ";
       da = new SqlDataAdapter();
      da.SelectCommand = com;
      DataTable dt = new DataTable();
      da.Fill(dt);
```

```
    con.Close();
  }
}
```

کنترل DataGridView

این کنترل برای نمایش داده‌های موجود در پایگاه داده به کار می‌رود. این کنترل خاصیتی به نام DataSource دارد که می‌تواند با یک DataSet و یا یک DataTable مقداردهی گردد.

در صورتی که با یک DataTable مقدار دهی گردد تنها کد زیر کافی است.

```
dataGridView1.DataSource = dt;
```

اما در صورتی که به یک DataSet متصل گردد باید علاوه بر DataSource که منبع داده های ما را مشخص می‌کند، خاصیت DataMember دیتاگرید خود را مقداردهی کنیم. زیرا دیتاست مجموعه‌ای از جداول را شامل می‌شود و ما باید با تعیین DataMember با نام جدول موردنظر ارتباط دیتاگرید خود را با یکی از جداول مشخص نماییم.

```
dataGridView1.DataSource = ds;
dataGridView1.DataMember = " std";
```

DataBindings

با استفاده از این خاصیت می‌توان کنترل TextBox را به رکورد جاری table خود متصل کرد.

```
textBox1.DataBindings.Add("text", dt, "stdCode");
textBox2.DataBindings.Add("text", dt, "stdName");
```

که با استفاده از رویداد Add در خاصیت DataBindings توانستیم این کنتـرل را بـه DataTable متصل کنیم. همان‌طور که ملاحضه می‌کنید این متد دارای 3 پارامتر است.

پارامتر اول خاصیتی را مشخص می‌کند که می‌خواهیم مقادیر فیلدها به آن نسبت داده شود.

پارامتر دوم DataTable ی را مشخص می‌کند که می‌خواهیم اطلاعات را از آن بخوانیم.

پارامتر سوم نام فیلدی را مشخص می‌کند که می‌خواهیم مقادیر آن در TextBox نمایش داده شوند.

اتصال کنترل ComboBox :

کنترل comboBox می‌تواند به گونه‌ای تنظیم شود که هر کدام از عناصر لیستش شامل مقـدار فیلـدی از رکوردهای بانک باشد.برای اتصال کنترل comboBox خود نیز می توان به شیوه ی زیر عمل کرد:

```
comboBox1.DisplayMember = "stdname";
comboBox1.ValueMember = "stdcode";
comboBox1.DataSource = dt;
```

همان‌طور که مشاهده می‌کنید در اینجا سه خاصیت comboBox را مقداردهی کرده‌ایم.

DisplayMember: فیلدی را تعیین می‌کنیم که می‌خواهیم مقدار comboBox بر اساس آن باشد.

ValueMember: فیلد کلید جدول‌مان را باید مشخص کنیم.

DataSource: باید جدولی را مشخص کنیم که می‌خواهیم از آن اطلاعات را بخوانیم.

بخش چهارم

پیاده سازی عملی پروژه‌ی StdTable

همان‌طور که دانشجویان عزیز اطلاع دارند، زبان C# یک زبان کاملاً شیء گراست. این بدین معنی است که این زبان با اشیاء و کلاس‌ها سروکار دارد و برنامه‌هایی که به این زبان نوشته می‌شوند نیز باید از این قاعده تبعیت کنند و بر مبنای شیءگرایی نوشته شوند. ما نیز در این پروژه از روش شیءگرا برای نوشتن پروژه خود استفاده می‌کنیم.

پروژه‌ی ما دارای یک فرم است. درون آن فرم تعدادی دکمه وجود دارند که به وسیله آن دکمه کاربر می‌تواند داده‌های بانک اطلاعاتی را دستکاری (درج، حذف، اصلاح و ...) کند. همچنین کاربر با کلیک بر روی دکمه "نمایش" می‌تواند اطلاعات دستکاری شده بانک را مشاهده کند.

تا این مرحله جداول مورد نیاز ما تعریف شده‌اند و باید کد نویسی را برای دستکاری جداول شروع کنیم. قبل از هر چیز باید کلاسی ایجاد کنیم که ارتباط ما با جدول فیزیکی بانک را برقرار سازد. نام دلخواه این کلاس را MyDataAccessLayer می‌گذاریم.

این کلاس دارای چهار خاصیت و چهار متد می‌باشد.

خاصیت‌ها :

```
public string database = "database1.mdf";
private SqlConnection con;
private SqlCommand com;
private SqlDataAdapter da;
```

خاصیت اول نام بانک اطلاعاتی ما را مشخص می‌کند.

خاصیت‌های بعدی برای کار با بانک اطلاعاتی استفاده می‌شوند که توضیحات آنها پیشتر ارائه گردید.

و همچنین متدها:

```
public void connect()
{
   string cs = @"Data Source=.\SQLEXPRESS;AttachDbFilename=D:\New folder (2)\test\test\WindowsApplication1\Database1.mdf;Integrated Security=True;User Instance=True";
```

```
    cs = string.Format(cs, this.database);
    con.ConnectionString = cs;
    con.Open();
}
public void disconnect()
{
        con.Close();
}
public DataTable select(string s)
{
    com.CommandText = s;
    DataTable dt = new DataTable();
    da.Fill(dt);
    return dt;
}
public void docommand(string s)
{
    com.CommandText = s;
    com.ExecuteNonQuery();
}
```

متد اولی برای اتصال به بانک استفاده می‌شود.

متد دومی برای قطع اتصال از بانک استفاده می‌شود.

متد سوم برای انتخاب داده‌ها از جداول بانک استفاده می‌شود.

متد سومی برای اجرای یک دستور SQL غیر دستور انتخاب استفاده می‌شود.

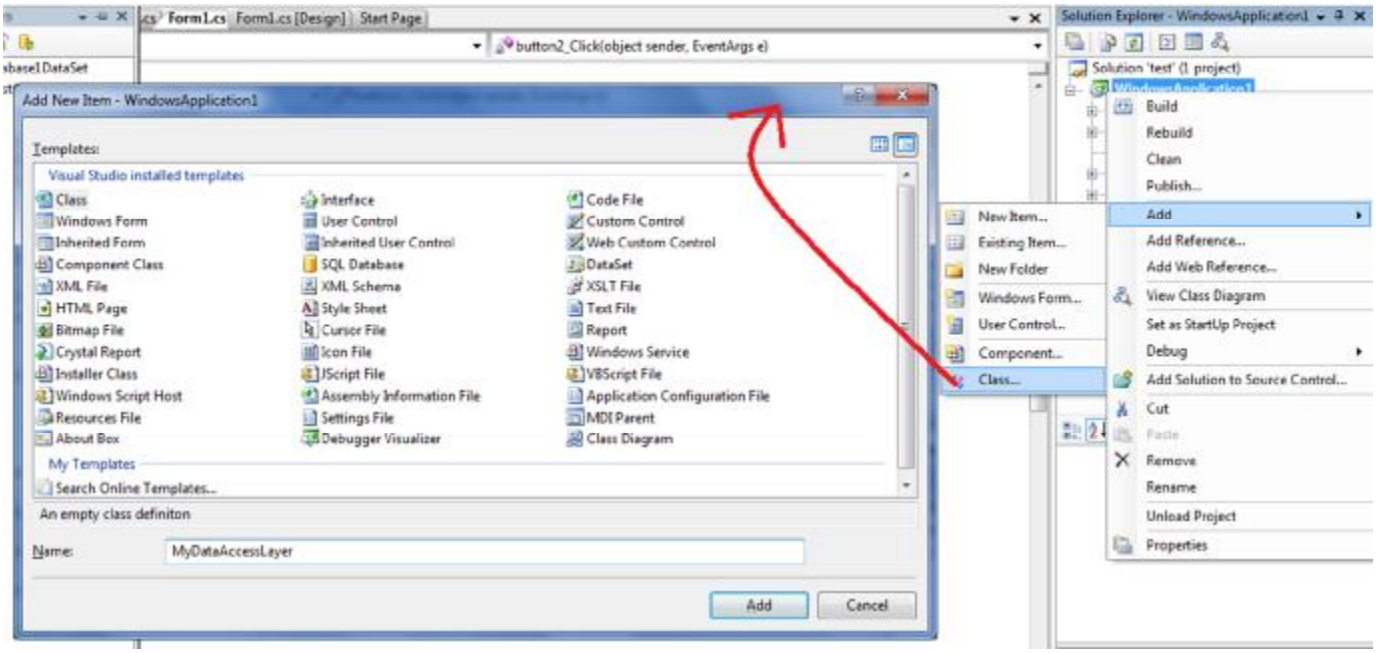
کـــــــد کامـــــــل ایـــــــن کـــــــلاس در زیـــــــر آورده شـــــــده اســـــــت:

```
using System;
using System.Collections.Generic;
using System.Text;
using System.Data.SqlClient;
using System.Data;

namespace WindowsApplication1
{
    class mydataaccesslayer
    {
        public string database = "database1.mdf";

        private SqlConnection con;
        private SqlCommand com;
        private SqlDataAdapter da;
        public mydataaccesslayer()
        {
            con = new SqlConnection();
            da = new SqlDataAdapter();
            com = new SqlCommand();
            com.Connection = con;
            da.SelectCommand = com;
        }
        public void connect()
        {
```

دقت شود که این کلاس دارای متدی همنام با خودش است. به این متد، متد سازنده کلاس می‌گوییم.کار این متد مقداردهی اولیه خاصیت‌های کلاس است.

برای تعریف این کلاس در پروژه مسیر زیر را دنبال کنید:

Solution Explorer > (right click on project name) > Add > Class...



در کادر محاوره ظاهر شده نام کلاس را MyDataAccessLayer بگذارید.

با این کار شما یک فایل کلاس خالی به پروژه خود اضافه کردید. اکنون می‌توانید خاصیت‌ها و متدهای خود را به آن اضافه کنید. برای راحتی کار می‌توانید خاصیت ها و متدهایی که در بالا لیست شده‌اند را به فایل کلاس خود اضافه کنید.

در این مرحله باید برای هر جدول،کلاس متناظر آن را ایجاد کنیم. تا این کـلاس فقـط روی داده‌هـای همان جدول کار کند. مثلا برای جدول STD کلاس STDClass را تعریف می‌کنیم که دارای خاصـیت و متدهایی به شکل زیر می‌باشد:

خاصیت‌ها :

```
public string stdname;
public  int stdcode;
public  int midterm;
public int final;
```

متدها

```
public DataTable selectall()
        {
            string sql = "select * from std ";
            _mydal.connect();
            DataTable dt = _mydal.select(sql);
            _mydal.disconnect();
            return dt;
        }
        public void upd()
        {
            string sql = "update std set stdName='{0}' WHERE stdCode = {1}";
            sql = string.Format(sql, this.stdname, this.stdcode);
            _mydal.connect();
            _mydal.docommand(sql);
            _mydal.disconnect();

        }
```

اگر دانشجویان عزیز دقت کنند متوجه می‌شوند که خاصیت‌ها دقیقا متناظر با فیلدهای جدول Std هستند. البته این خاصیت‌ها حتما باید به صورت Public تعریف شوند تا بتوان در کلاسهای دیگر از این خاصیت‌ها استفاده کرد. یکی از خاصیت‌های تعریف شده، شیئی از کلاس MyDataAccessLayer است. همانطور که گفته شد پروژه‌ی ما با استفاده از این کلاس به داده‌ها و جداول بانک اطلاعاتی خود دسترسی پیدا می‌کند.

تابع Format() :

این تابع که متدی از کلاس string است برای دستکاری رشته‌ها و قالب‌بندی آنها استفاده می‌شود.

این تابع در برنامه‌ها کاربرد بسیاری دارد.

```
string sql="insert into std(stdcode,stdname) values({0},'{1}')";
sql = string.Format(sql, this.stdcode, this.stdname);
```

در دو دستور بالا ابتدا درون رشته‌ی sql رشته‌ای قرار داده شد. سپس با استفاده از تابع format این رشته دستکاری گردیده است.

پارامتر اول رشته‌ای است که قرار است دستکاری گردد.

پارامترهای بعدی مقادیری را مشخص می‌کنند که قرار است درون رشته به جای پارامترها جایگزین شوند.

پارامترها با {} مشخص می‌شوند.

بدیهی است که پارامتر دوم به جای {0} و همینطور پارامتر سوم به جای {1} جایگزین می‌شود.

در '{1}' ، نماد کوتیشن که {1} که درون آن قرار دارد، رشته بودن آن را مشخص می کند.

اکنون می‌خواهیم ابتدا داده‌های جدول را بخوانیم و در یک DataGridView نمایش دهیم.

برای این‌کار یک کنترل DataGridView را به فرم اضافه کنید. سپس در رویداد کلیک دکمه نمـایش کدهای زیر را وارد نمایید:

```
private void button1_Click(object sender, EventArgs e)
{
        stdClass _s10 = new stdClass();
        dt = _s10.selectall();
        dataGridView1.DataSource = dt;
}
```

خط اول شیئی از کلاس متناظر با جدول STD می‌سازد.

خط دوم تمام داده‌های جدول STD را با استفاده از کلاس متناظر انتخـاب مـی‌کنـد و درون dt قـرار می‌دهد. dt در اینجا یک DataTable است.

خط سوم dt را به DataGridView1 نسبت می‌دهد.

در این قطعه کد از متد SelectAll کلاس StdClass استفاده شده است.این کلاس ابتدا با اسـتفاده از کلاس MyDataAccessLayer دستور select را اجرا می‌کند سپس داده‌هـای انتخـاب شـده را بـه صورت یک DataTable برمی‌گرداند.

اکنون می‌خواهیم با کلیک روی دکمه "اضافه" که به پروژه‌ی خـود اضـافه کـردیم. رکـوردی را بـه جدول STD اضافه کنیم. برای این‌کار در رویداد کلیک دکمه، کدهای زیر را وارد کنید:

```
private void button3_Click(object sender, EventArgs e)
    {
      stdClass _std = new stdClass();
      _std.stdcode = Convert.ToInt32(textBox2.Text);
      _std.stdname = textBox1.Text;
      _std.addfromform();
    }
```

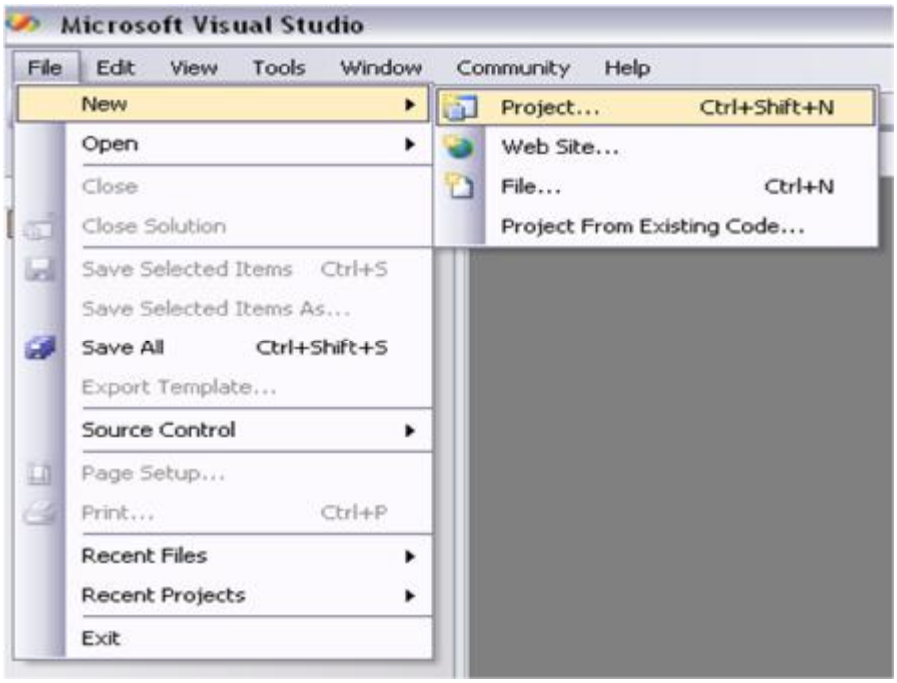
این کد و سایر کدهای دیگر بسیار ساده بوده و نیازی به توضیح ندارند.

بخش پنجم

ایجاد فایل اجرایی از پروژه

برای ایجاد فایل اجرایی از پروژه ی خود ابتدا باید مسیر زیر را رفته که یک پروژه ی جدید ایجاد نمایید.

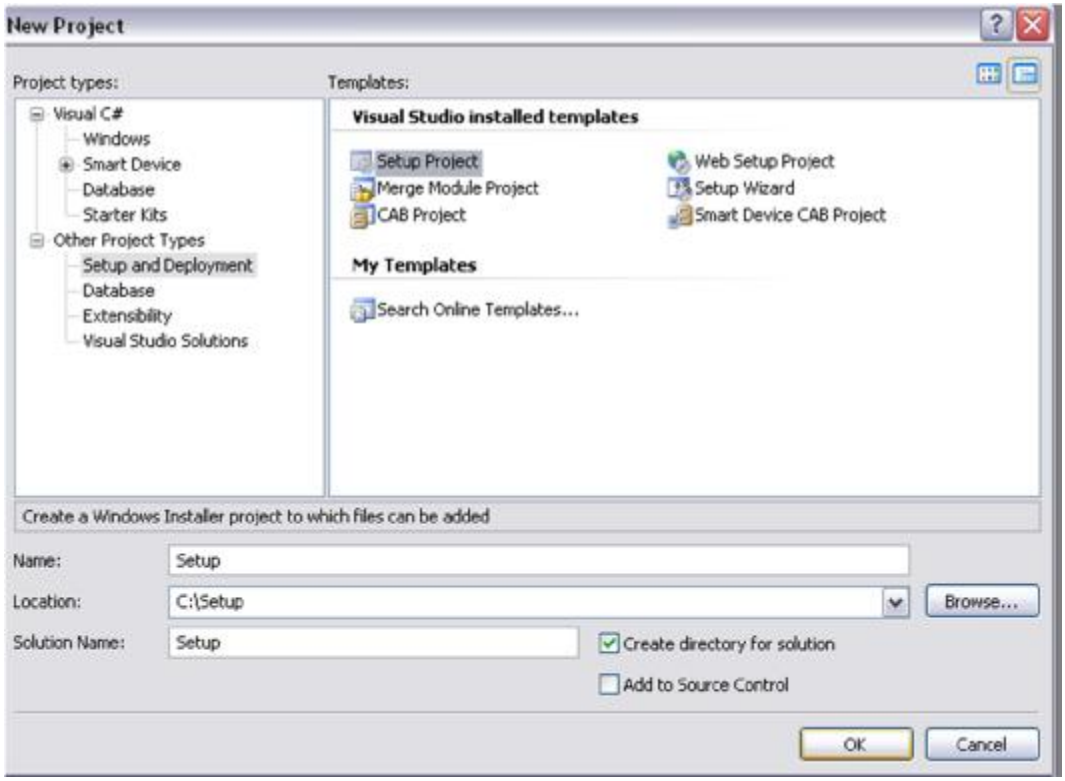
File > New > Project ...

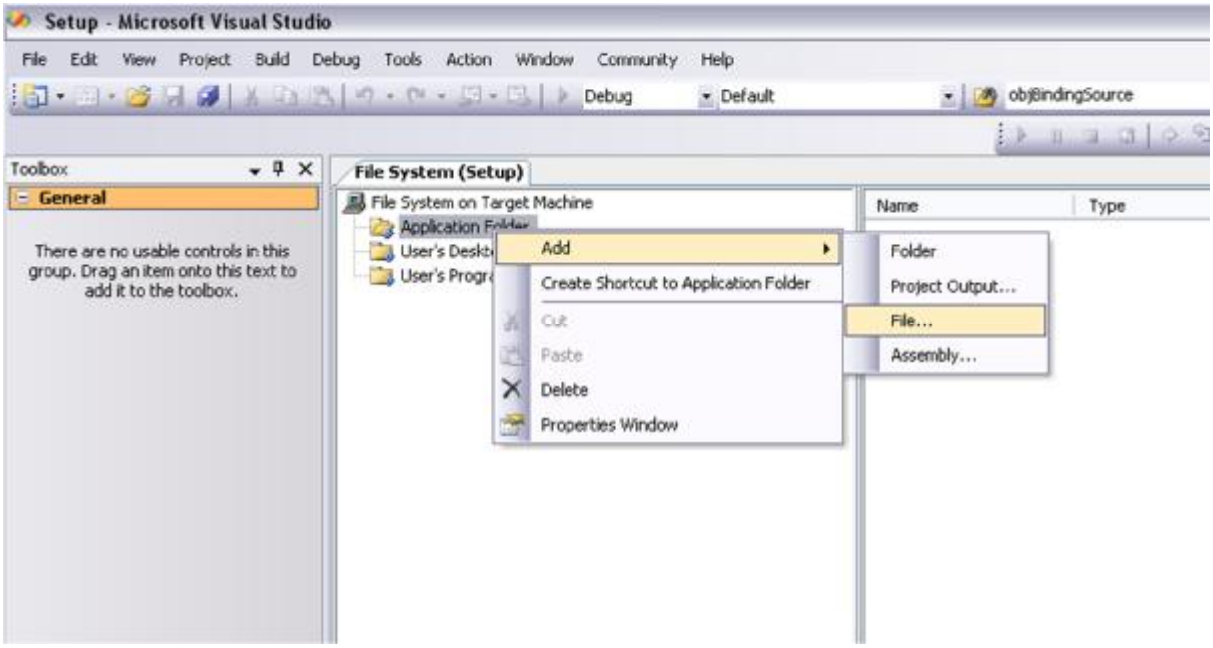


سپس همان‌طور که مشاهده می کنید پروژه‌ای از نوع Setup Project ایجاد می‌کنیم.

Other Project Types > Setup and Deployment > Setup Project

سپس مسیر و نامی را که برای ذخیره شدن فایل اجرایی مد نظر داریم را وارد می‌کنیم.



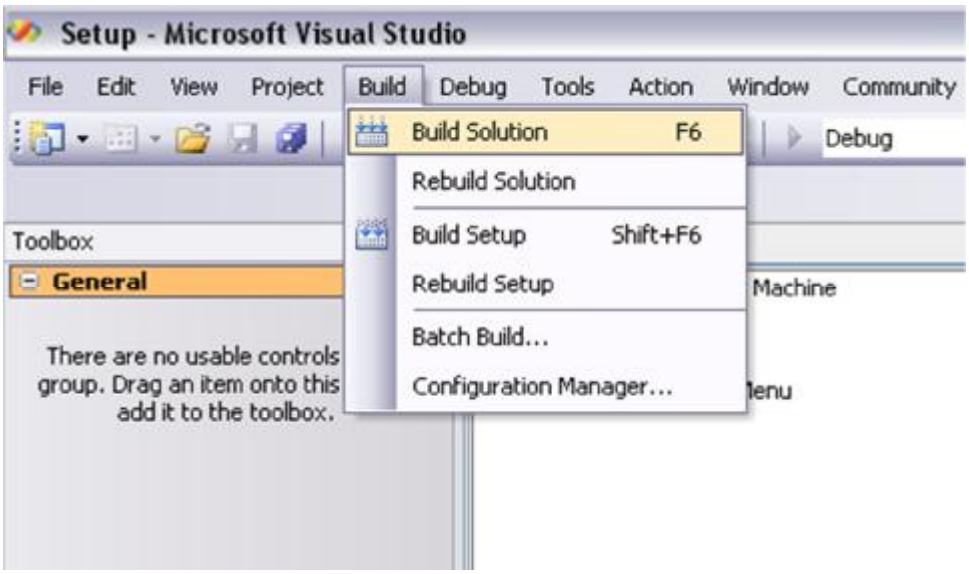


پس از اینکه این پروژه ایجاد شد در قسمت File System(setup) بـر روی Apllication Folder کلیک راست نمود پس از آن گزینه‌ی Add و درنهایت ...File را انتخاب می‌کنیم.

در کادری که ظاهر میگردد به سراغ مسیر پروژه ی برنامه ای که می خواهیم ازآن فایل اجرای درست کنیم رفته ، به قسمت bin>debug پروژه رفته و فایل های موجود در آن را انتخاب میکنیم.

فایل exe و همچنین فایل بانک اطلاعاتی مورد استفاده شده، تصاویر و یا دیگر فایلهای پروژه ی مورد استفاده در پروژه ی اصلی خود را انتخاب تا به Project Setup اضافه گردد

در نهایت از منوی Build گزینه ی Build Soulation را انتخاب کرده (یا تنها F6 را بزنید) تا فایل اجرایی ساخته شود.



فایل اجرایی ایجاد شده در پوشه ی Debug مربوط به مسیر پروژه ی Setup که در اول همین بخش ایجاد کردیم ساخته می شود.

بخش ششم

استفاده از XML در برنامه

Xml یک زبان انتقال اطلاعات است. در محیط‌های تجاری برای اینکه برنامه‌ها بتوانند با یکدیگر ارتباط برقرار کنند باید از یک زبان انتقال استفاده کنند. منظور از برنامه‌های تجاری نرم‌افزارهای معمولی مانند آفیس و ... نیست. بلکه نرم‌افزارهایی مانند نرم‌افزار انبارداری یا حسابداری و... است.

ممکن است فکر کنید که برای انتقال اطلاعات چرا باید از xml استفاده کرد.

* چرا از بانک اطلاعاتی استفاده نکنیم؟

در جواب باید گفت که xml فرمت استانداردی است و توسط همه شناخته شده است. Xml شباهت بسیاری به html دارد. یعنی برای انتقال اطلاعات از تگ استفاده می‌شود.

تفاوت xml و html در چیست ؟

html داده‌ها را نمایش می‌دهد در حالی که XML داده‌ها را ارائه می‌دهد.

مثلا کد زیر نمونه‌ای از چند عنصر xml است :

```
<Book>
<Title>Learning Visual C# 2005</Title>
<ISBN>XXXX-XX-XXX</ISBN>
<Subject>Programming</Subject>
</Book>
```

ذخیره رکوردهای یک جدول در یک فایل XML :

برای اینکار متد زیر را در فایل MyDataAccessLayer کپی کنید:

```
public DataSet select2DS(string s, string sTableName)
    {
       da = new SqlDataAdapter(com);
       DataSet ds = new DataSet();
       da.SelectCommand = new SqlCommand(s);
       da.SelectCommand.Connection = con;
       da.Fill(ds, sTableName);
       return ds;
    }
```

با استفاده از این متد ما اطلاعات را از بانک می‌گیریم و در یک **Dataset** ذخیره می‌کنیم.سپس باید متدهای زیر را در کلاس متناظر با جدول خود اضافه کنید.مثلا در پروژه‌ی **STD** باید متدهای زیر را اضافه کرد:

```
public void writeAllToXML(string sFileName)
    {
       string sql = "select * from std ";
       _mydal.connect();
       DataSet ds= _mydal.select2DS(sql, "std");
       ds.WriteXml(sFileName);
       _mydal.disconnect();
    }

public DataTable readAllFromXML(string sFileName)
    {
       _mydal.connect();
       DataSet ds = new DataSet();
       ds.ReadXml(sFileName);
       _mydal.disconnect();

       return ds.Tables[0];
    }
```

وظیفه‌ی متد اول این است که مسیر یک فایل **XML** را گرفته سپس رکوردهای جدول **STD** را در یک دیتاست به نام **ds** ذخیره می‌کند. در آخر همه‌ی اطلاعات را آن فایل xml ذخیره می‌کند. وظیفه‌ی متد دوم این است که مسیر یک فایل xml را گرفته سپس رکوردها را از آن فایل xml خوانده و در

یک دیتاست کمکی (ds) ذخیره می‌کند .در آخر تنها جدول موجود در آن دیتاست کمکی را به عنوان خروجی برمی‌گردان. سپس در فرم خود باید دکمه‌ای اضافه کنیم که دکوردهای جدول STD را برای ما در یک فایل xml ذخیره می‌کند.کدهای زیر را در رویداد کلیک دکمه اضافه می‌کنیم:

```
stdClass _s10 = new stdClass();

_s10.writeAllToXML("c:\\stdTable.XML");
```

کدهای بالا رکوردهای جدول STD را در فایل stdTable.xml ذخیره می‌کند.

برای خواندن اطلاعات ذخیره شده در فایل فوق دکمه‌ای را به فرم اضافه می‌کنیم سپس کدهای زیر را به رویداد کلیک آن دکمه اضافه می‌کنیم:

```
stdClass _s10 = new stdClass();

DataTable dt = _s10.readAllFromXML("c:\\stdTable.XML");

dataGridView1.DataSource = dt;
```

با آرزوی توفیق و سربلندی، خواهشمند است هر نوع نظر و پیشنهادی دارید، به ایمیل r.k.shahvandi@gmail.com ارسال فرمایید.