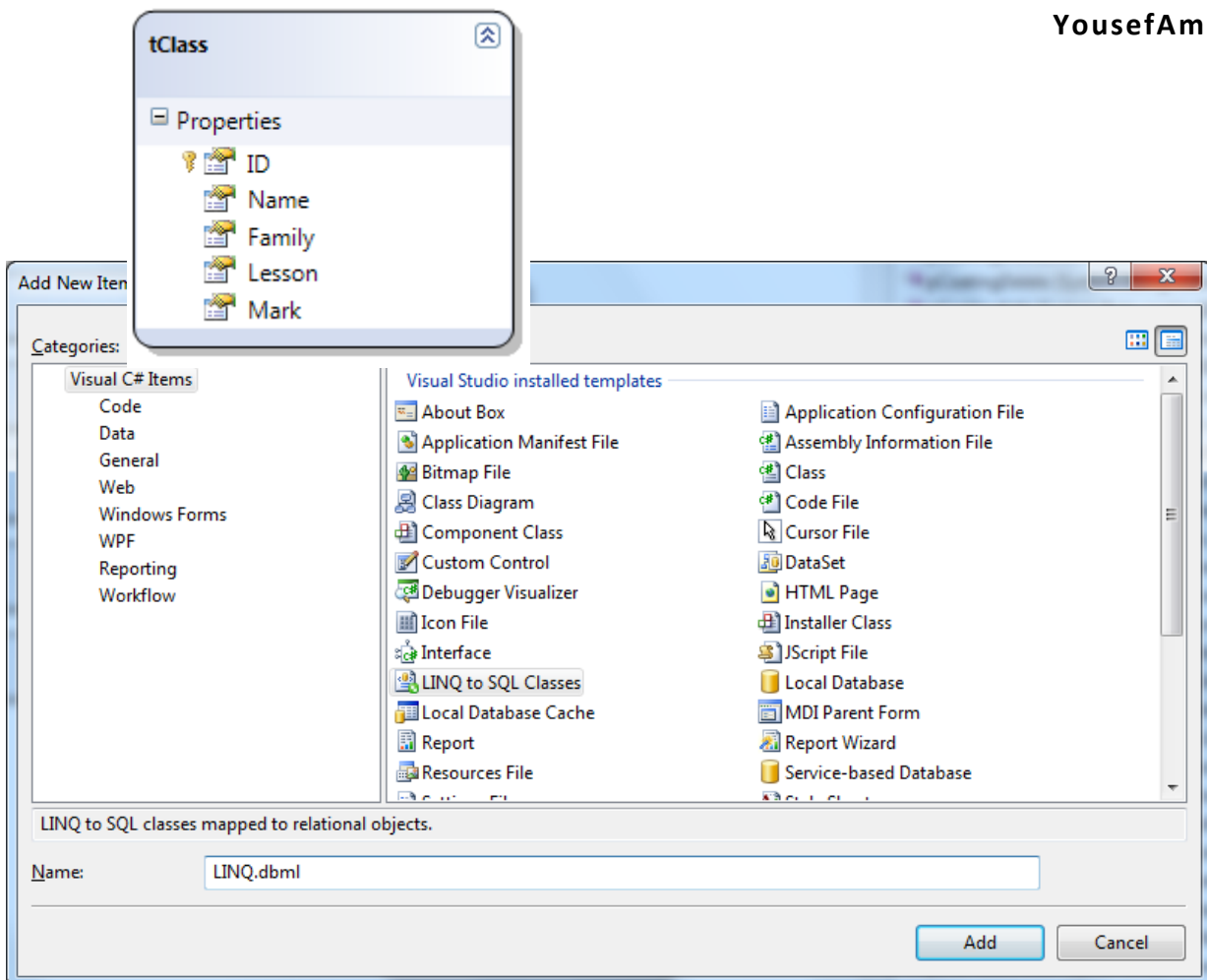


# Using LINQ in C# 2

## استفاده از LINQ در C# بخش ۲

ویرایش: مردادماه ۱۳۸۹

YousefAmiri.ir



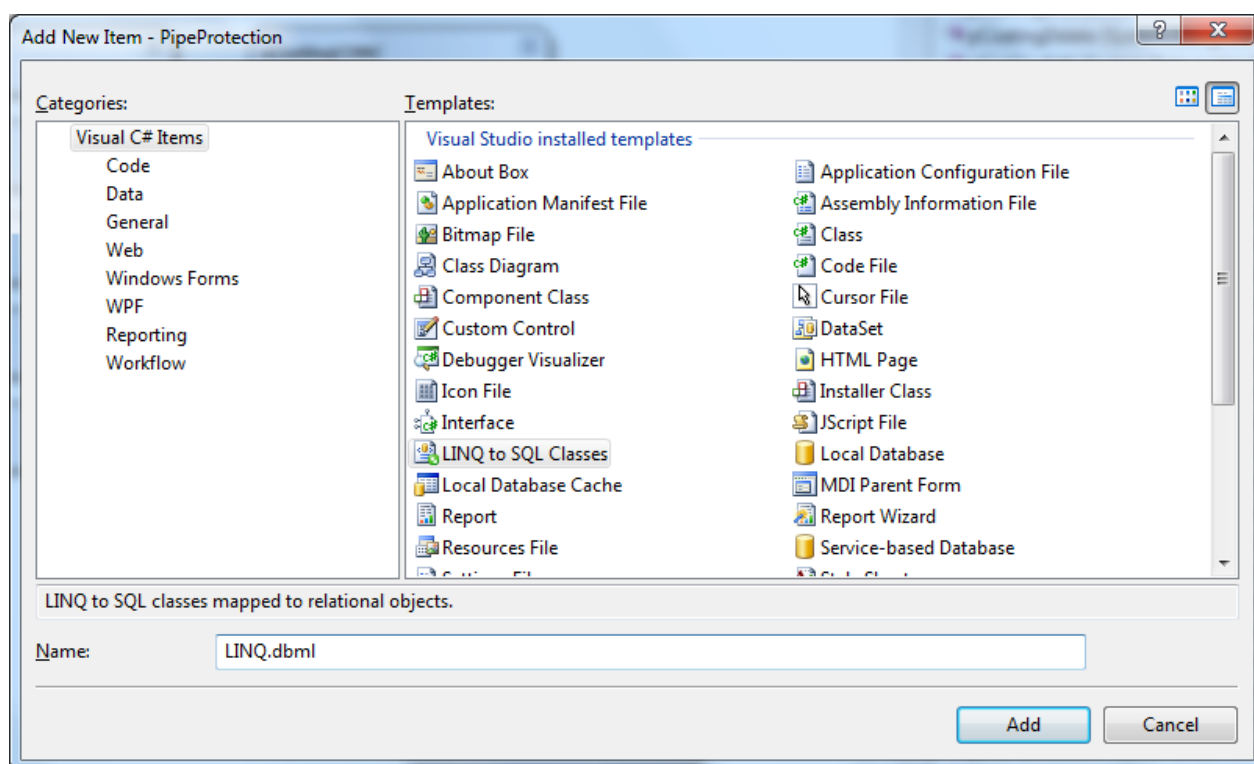
در این مقاله بصورت کاربردی و سریع به پرکاربردترین دستورات LINQ پرداخته خواهد شد. با ترکیب کردن موارد ذکر شده قادر خواهید بود به راحتی، عمده ترین پرس و جوها را با دیتابیس داشته باشید. قبل از شروع، چنانکه هیچ آشنایی اولیه ای با LINQ ندارید، بخش اول این مطلب را از سایت [itnee.com](http://itnee.com) دانلود و مطالعه نمایید.

## شروع

۱. در Visual Studio پروژه ای به نام LINQFast بسازید.

۲. محیط LINQ را به پروژه خود اضافه کنید:

Project > Add New Item



شکل ۱ - با انتخاب LINQ to SQL Classes، محیط لینک را به پروژه اضافه کنید

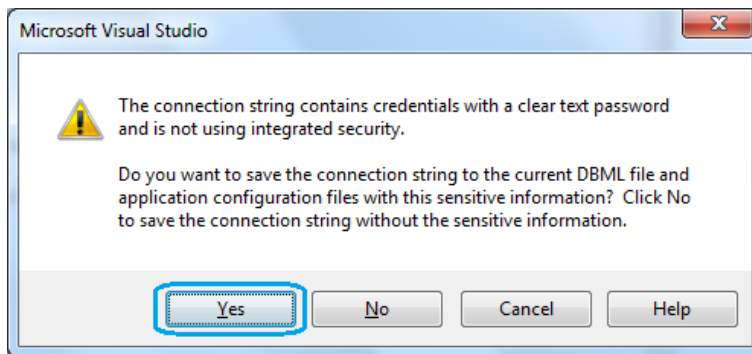
۳. با فرض داشتن جدول زیر در بانک اطلاعاتی، پانل View > Server Explorer را فعال کنید و جدول tClass را به محیط LINQ بکشید.

tClass		
ID		int
Name		nvarchar(20)
Family		nvarchar(30)
Lesson		nvarchar(30)
Mark		real

**توجه:** ویژوال استودیو پیامی نشان خواهد داد تا از شما تایید بگیرد که کلمه عبور اتصال به بانک اطلاعاتی را در

Connection String قرار دهد یا نه! این کار امنیت بانک اطلاعاتی را پایین می آورد. در این پروژه، برای حفظ سادگی کار،

دکمه Yes را کلیک میکنیم



۴. در Form1 دکمه ای با نام `btRun`، برچسبی با نام `lbResult`، کومبویی با نام `cbLesson` و گرایدی با نام `gvClass` قرار دهید. دکمه، دستورات LINQ را اجرا می کند. برچسب، برگشتی های تک مقداری را نشان می دهد. (تعداد، میانگین) گراید، برگشتی های چند مقداری را نشان می دهد. (رکورد، فیلد یا یک ستون از جدول)

## یادآوری

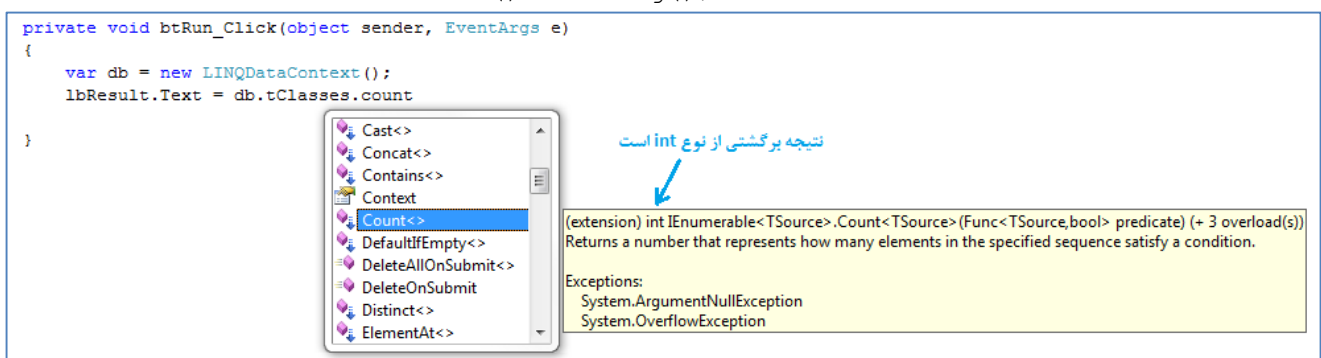
دستور زیر کلیه داده های جدول را در گراید می ریزد:

```
var db = new LINQDataContext();
gvClass.DataSource = db.tClass;
```

## تعداد رکوردها

دستور زیر تعداد رکوردهای جدول را بر می گرداند و آنرا در Label قرار می دهد:

```
var db = new LINQDataContext();
lbResult.Text = db.tClasses.Count().ToString();
```



شکل ۲- فراخوانی متد `Count()`

## تعداد رکوردها با شرط

```
var db = new LINQDataContext();
lbResult.Text = db.tClasses.Where(c => c.Lesson == "Math").Count().ToString();
```

قبل از `Count()` شرط را قرار می دهیم. اگر تعداد شرطها بیشتر بود عبارت آن مانند زیر خواهد بود:

```
Where(c => c.Lesson == "Math" || c.Lesson == "Computer")
```

```
Where(c => c.Name == "نام" && c.Family == "خانوادگی نام")
```

توجه: اگر هیچ رکوردی در جدول موجود نباشد، متد `Count()` صفر برمی گرداند.

### ماکزیمم نمره

اگر هیچ رکوردی در جدول موجود نباشد یا فیلدهای `Mark` تهی باشند، متد `Max()` مقدار `null` بر می گرداند. پس بهتر است همیشه نتیجه آنرا در یک متغیر `object` قرار دهیم تا از بروز خطا جلوگیری کنیم.

```
var db = new LINQDataContext();
object mx = db.tClasses.Select(c => c.Mark).Max();

if (mx == null)
    lbResult.Text = "No Mark";
else
    lbResult.Text = mx.ToString();
```

### میانگین و جمع نمرات

```
var db = new LINQDataContext();
object mx = db.tClasses.Select(c => c.Mark).Average();

if (mx == null)
    lbResult.Text = "No Mark";
else
    lbResult.Text = mx.ToString();
```

برای جمع، به جای `Average()` از متد `Sum()` استفاده کنید

### یک رکورد

```
var db = new LINQDataContext();
gvClass.DataSource = db.tClasses.Where(c => c.ID == 1).Select(c => c).First();
```

نکته ۱: در عبارت `Select` که فقط `c` را ذکر کرده ایم، همه فیلدها برگردانده می شوند.

نکته ۲: متد `First()` به معنی اینست که اولین رکورد را از یافته ها برگردان. البته چون در اینجا شرط ما روی کلید اولیه است، همیشه یک رکورد برگردانده می شود. اما LINQ متوجه این کار نمی شود! پس همیشه وقتی می خواهید یک رکورد یا فیلد را برگشت دهید، بعد از `Select` از `First` استفاده کنید.

### چند رکورد

کافیست از `First` استفاده نکنید و شرطی را قرار دهید که صفر، یک یا چند رکورد را برگشت دهد.

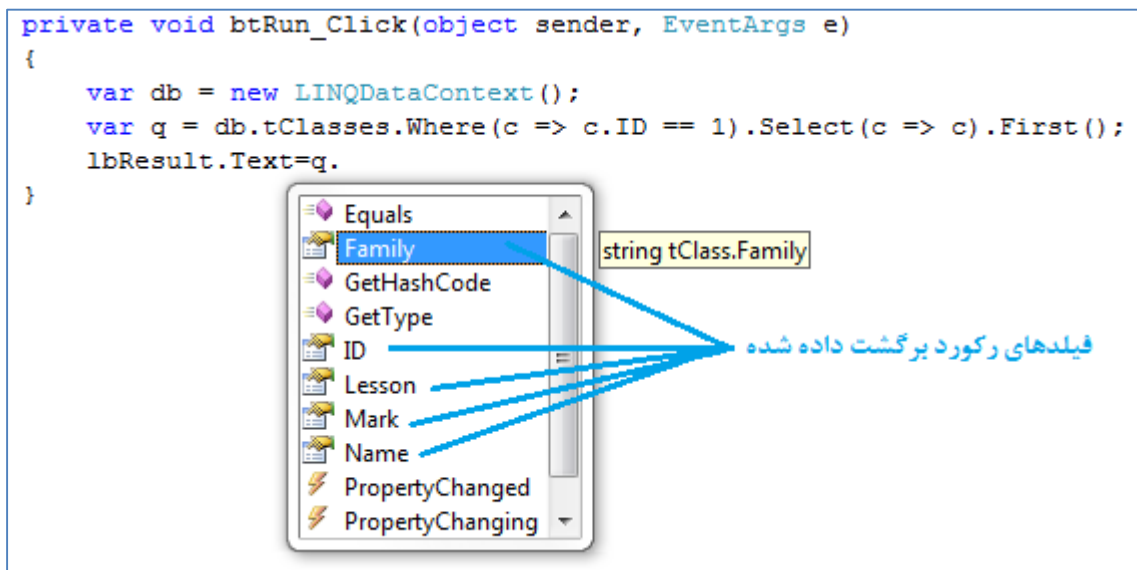
```
gvClass.DataSource = db.tClasses.Where(c => c.Mark < 10).Select(c => c);
```

### فیلدهای یک رکورد

این دستور مانند مثال قبلی است اما به جای قرار دادن نتیجه در یک گراید، می خواهیم به تک تک فیلدها دسترسی داشته باشیم.

`q` متغیری از نوع نامشخص است که ماهیت آن با توجه به مقدار برگشتی تعیین می شود. در اینجا به معنی یک رکورد است. و `q.Family` به معنی فیلد `Family` از رکورد `q` است.

```
var db = new LINQDataContext();
var q = db.tClasses.Where(c => c.ID == 1).Select(c => c).First();
lbResult.Text = q.Family;
```



شکل ۳ - دسترسی به فیلدهای یک رکورد

### تعدادی از فیلدها

عبارت `Select(c => c)` همیشه راه حل خوبی نیست! چون همه فیلدها را برمی گرداند و در کارهای عملی که با حجم بالایی از اطلاعات سروکار داریم، باعث کند شدن سرعت لود شدن اطلاعات می شود. پس فقط فیلدهای مورد نیاز را لود می کنیم.

```
var q = db.tClasses.Where(c => c.ID == 1).Select(c => new { c.Name, c.Family }).First();
```

در اینجا رکورد `q` شامل ۲ فیلد است

### مرتب سازی صعودی

```
gvClass.DataSource = db.tClasses.Where(c => c.Mark < 10).Select(c => c).OrderBy(c => c.Family);
```

### مرتب سازی نزولی

```
gvClass.DataSource = db.tClasses.Select(c => c).OrderByDescending(c => c.Family);
```

### مرتب سازی پله‌ای

```
gvClass.DataSource = db.tClasses.Select(c => c).OrderBy(c=>c.Family).ThenBy(c=>c.Name);
```

### حلقه For

گاهی اوقات لازم است رکوردها را به ترتیب و در یک حلقه `for` مورد جستجو یا پیمایش قرار دهیم برای این کار از دو متد `Skip()` و `Take()` استفاده می کنیم. در این مثال، سه نفری که بیشترین نمره را گرفته اند، برمی گردانیم.

`Skip()` پرش به رکورد بعدی (معمولا پارامتر آن، اندیس حلقه است)

`Take()` تعداد رکورد برای برگشت مانند `TOP` در `SQL` (معمولا پارامتر آن عدد ۱ است)

اسامی را به ترتیب بیشترین نمره مرتب می کنیم و سه نفر بالای لیست را بر می گردانیم

```
var q = db.tClasses.Select(c => c).OrderByDescending(c => c.Mark).Take(3);
```

برای جلوگیری از خطا، ابتدا چک می‌کنیم که تعداد رکوردهای مورد درخواست، موجود باشد.

```
if (3 <= db.tClasses.Count())
    for (int i = 0; i < 3; i++)
        MessageBox.Show(q.Select(c => c.Family).Skip(i).Take(1).First());
```

### چک کردن کلمه عبور

```
if (db.tLogins.Where(c => c.Password == txPassword.Text).Count() == 0)
    MessageBox.Show("Invalid password!");
```

### مقادیر غیر تکراری

می‌خواهیم ستونی از جدول را برگردانیم، مقادیر تکراری آنرا حذف و نتیجه را در جایی مانند یک **ComboBox** بریزیم.

```
cbLesson.DataSource = db.tClasses.Select(c => c.Lesson).Distinct();
```

**Distinct()** نتایج تکراری را حذف می‌کند.

