بسم الله الرحمن الرحیم

آموزشگاه تحلیل داده

تخصصی ترین مرکز برنامه نویسی و دیتابیس در ایران

ASP.NET و LINQ

مدرس : مهندس افشین رفوآ

ASP.NET و LINQ

اغلب اپلیکیشن ها مبتنی بر معماری data-centric هستند (بدین معنا که پایگاه داده در آن نقش اساسی ایفا می کند) ، با این وجود بیشتر data repository (انباره های داده)، پایگاه داده های رابطه ای (relational) هستند. طی سالیان دراز طراحان و برنامه سازان سعی کردند برنامه ها یا اپلیکشن های خود را بر مبنای object model (مدل شئ گرایی) تعبیه و طراحی کنند.

اشیاء مسئول اتصال به کامپوننت های دسترسی به داده (data access components) هستند. این کامپوننت ها Data Access Layer (لایه ی دسترسی به داده) نیز اطلاق می گردند.

حال به سه نکته ی زیر دقت کنید:

کلیه ی داده های مورد نیاز یک اپلیکیشن لزوماً در یک منبع (source) ذخیره نمی شوند. منبع می تواند یک پایگاه داده ی رابطه ای، یک business object، فایل XML و یا یک سرویس وب باشد.

دسترسی به یک شئ که در حافظه ی اصلی کامپیوتر ذخیره شده باشد (in-memory object) به مراتب آسان تر از دستیابی به یک شئ از پایگاه داده یا فایل XML هست.

داده هایی که دسترسی به آن صورت گرفته باید ابتدا مرتب و گروه بندی شده یا اصلاح شوند.

بنابراین اگر ابزاری وجود دارد که دسترسی به داده را آسان ساخته و امکان متصل کردن داده ها را از چندین منبع داده ی متفاوت و نیز اجرای عملیات متعارف پردازش داده را در تنها چند خط کد فراهم بیاورد، کمک بزرگی خواهد بود.

Language Integrated Query و یا به اختصار LINQ چنین ابزاری است. لینک یک زبان برای تقاضا از هر گونه مجموعه داده (بانک اطلاعاتی، آرایه‌ها، Xml و...) می‌باشد.

هدف اصلی خلق LINQ ارائه یک زبان یکپارچه جهت تقاضا، برنامه ریزی و فیلتر اطلاعات ذخیره شده در اشیاء مختلف از جمله پایگاه داده، اشیا، آرایه‌ها و از همه مهتر XML است، این زبان قابلیت اشکال زدایی به صورت Runtime را دارست و همچنین بسیار انعطاف پذیر می باشد.

لینک در واقع یک مجموعه افزونه برای زبان های پشتیبانی شده تحت .Net Framework 3.5 بوده که query (پرسمان از داده یا پایگاه داده) را به عنوان شئ تنظیم (set) می کند. این زبان یک دستور نگارش (syntax) و الگوی برنامه نویسی (programming model) مشترک و یکپارچه برای query گرفتن از انواع مختلف داده با استفاده از یک زبان مشترک ارائه می دهد (تعریف می کند).

عملگرهای رابطه ای (relational operators) مانند Select، Project، Join، Group، Partition، Set در LINQ و کامپایلرهای VB و C# در .Net Framework 3.5 پیاده سازی می شوند. این عملگرها از دستور نگارش LINQ پشتیبانی می کنند که قابلیت و امکان کار با انباره های داده (data store) پیکربندی شده را بدون متوسل شدن به ADO.NET فراهم می آورد.

به عنوان مثال برای query گرفتن از جدول Customers موجود در پایگاه داده ای به نام Northwind، با استفاده از LINQ query در C#، لازم است از کدی به صورت زیر استفاده کنید:

```
var data = from c in dataContext.Customers
where c.Country == "Spain"
select c;
```

که در آن:

کلیدواژه ی 'from' به صورت منطقی کل محتویات مجموعه را تکرار می کند (در آن حلقه می زند).

عبارت دربردانده ی کلیدواژه ی 'where' به ازای (برای) هر شئ موجود در مجموعه (collection) ارزیابی می شود.

دستور 'select' شئ ارزیابی شده را گزنیش کرده و آن را به لیست برگشتی اضافه می کند.

کلیدواژه ی 'var' به منظور تعریف یک متغیر جدید مورد استفاده قرار می گیرد. به این خاطر که نوع دقیق شئ بازگشتی مشخص نیست، بیانگر این است که اطلاعات به صورت پویا (dynamic) برداشت شده یا تعریف می شوند.

LINQ query می تواند به هر کلاس حامل داده ای که از IEnumerable<T> به ارث می برد اعمال شود. در اینجا T می تواند هر نوع داده ای باشد، برای مثال List<Book>.

برای درک بهتر به ذکر مثالی خواهیم پرداخت. این مثال از کلاس Books.cs استفاده می کند:

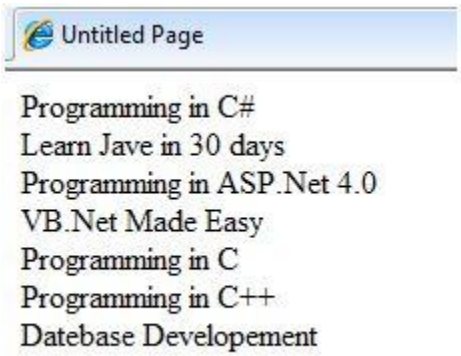
```
using System;
using System.Collections.Generic;
public class Books
{
    public string ID { get; set; }
    public string Title { get; set; }
    public decimal Price { get; set; }
    public DateTime DateOfRelease { get; set; }

    public static List<Books> GetBooks()
    {
        List<Books> list = new List<Books>();
        list.Add(new Books
        {
            ID = "001",
            Title = "Programming in C#",
            Price = 634.76m,
            DateOfRelease = Convert.ToDateTime("2010-02-05")
        });

        list.Add(new Books
        {
            ID = "002",
            Title = "Learn Jave in 30 days",
            Price = 250.76m,
            DateOfRelease = Convert.ToDateTime("2011-08-15")
        });

        list.Add(new Books
        {
            ID = "003",
            Title = "Programming in ASP.Net 4.0",
            Price = 700.00m,
            DateOfRelease = Convert.ToDateTime("2011-02-05")
        });

        list.Add(new Books
        {
            ID = "004",
            Title = "VB.Net Made Easy",
            Price = 500.99m,
            DateOfRelease = Convert.ToDateTime("2011-12-31")
        });

        list.Add(new Books
        {
            ID = "005",
            Title = "Programming in C",
            Price = 314.76m,
            DateOfRelease = Convert.ToDateTime("2010-02-05")
        });
```

```
        list.Add(new Books
        {
            ID = "006",
            Title = "Programming in C++",
            Price = 456.76m,
            DateOfRelease = Convert.ToDateTime("2010-02-05")
        });

        list.Add(new Books
        {
            ID = "007",
            Title = "Datebase Developement",
            Price = 1000.76m,
            DateOfRelease = Convert.ToDateTime("2010-02-05")
        });

        return list;
    }
}
```

صفحه ی وبی که این کلاس را مورد استفاده قرار می دهد دارای یک کنترل label ساده است که به وسیله ی آن عنوان کتاب ها را نمایش می دهد. رخداد Page_Load فهرستی از کتاب ها ایجاد کرده و عنوان آن ها را با بهره گیری از LINQ query بازمی گرداند:

```
public partial class simplequery : System.Web.UI.Page
    {
        protected void Page_Load(object sender, EventArgs e)
        {
            List<Books> books = Books.GetBooks();
            var booktitles = from b in books select b.Title;

            foreach (var title in booktitles)
                lblbooks.Text += String.Format("{0} <br />", title);
        }
     }
```

هنگامی که صفحه اجرا می شود، کنترل label نتایج پرسمان (query) را اینگونه نمایش می دهد:



عبارت LINQ:

```
var booktitles =
from b in books
select b.Title;
```

که دقیقاً معادل SQL query زیر می باشد:

```
SELECT Title from Books
```

عملگرهای LINQ

جدا از عملگرهایی که تاکنون به کار برده شدند، عملگرهای دیگری نیز وجود دارند که تمامی query clause ها (عبارت و دستور های پرسمان) را پیاده سازی می کنند. در این مبحث به معرفی و تشریح برخی از این عملگرها و دستورها خواهیم پرداخت.

Join clause (دستور join)

دستور join (پیوند) به منظور پیوند دادن دو جدول داده و نمایش ستون هایی که دربردارنده ی داده از هر دو جدول هستند بکار می رود. البته زبان LINQ نیز توانایی انجام همین عملیات را دارد. برای بررسی این موضوع کافی است کلاسی دیگری با نام Saledetails.cs در پروژه ی قبلی وارد کنید:

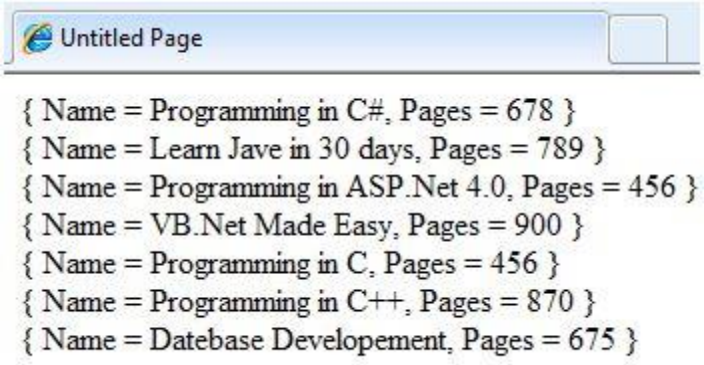
```
using System;
using System.Collections.Generic;
using System.Data;
public class Salesdetails
{
    public int sales { get; set; }
    public int pages { get; set; }
    public string ID { get; set; }

    public static IEnumerable<Salesdetails> getsalesdetails()
    {
        Salesdetails[] sd =
{
new Salesdetails { ID = "001", pages=678, sales = 110000},
new Salesdetails { ID = "002", pages=789, sales = 60000},
new Salesdetails { ID = "003", pages=456, sales = 40000},
new Salesdetails { ID = "004", pages=900, sales = 80000},
new Salesdetails { ID = "005", pages=456, sales = 90000},
new Salesdetails { ID = "006", pages=870, sales = 50000},
new Salesdetails { ID = "007", pages=675, sales = 40000},
};
        return sd.OfType(Salesdetails)();
    }
}
```

کدهای موجود در مدیریت کننده ی رخداد / Page_Load event handler را با استفاده از دستور join به query در هر دو جدول اضافه کنید.

```
        protected void Page_Load(object sender, EventArgs e)
        {
            IEnumerable<Books> books = Books.GetBooks();
```

```
    IEnumerable<Salesdetails> sales = Salesdetails.getsalesdetails();
    var booktitles = from b in books
     join s in sales on b.ID equals s.ID
     select new { Name = b.Title, Pages = s.pages };
    foreach (var title in booktitles)
   lblbooks.Text += String.Format("{0} <br />", title);
}
```

نتیجه:

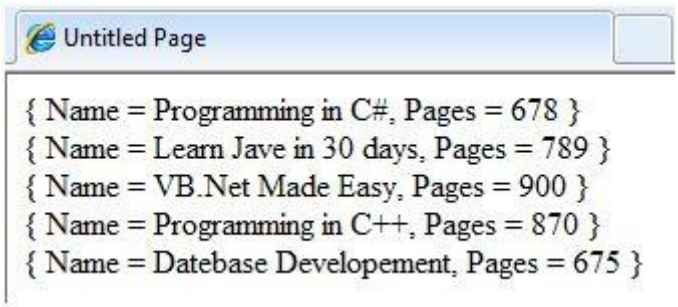


دستور Where

دستور Where برای افزودن شرط به منظور محدود کردن نتایج حاصل از جستجو و استخراج نتایج دقیقتر مورد استفاده قرار می گیرد. با استفاده از دستور Where می توان تعدادی فیلتر شرطی (conditional filter) به query (پرسمان) افزود. برای مثال اگر قصد مشاهده ی کتاب ها را داشته باشید که تعداد صفحات آن بالغ بر 500 صفحه است باید Page_Load event handler را به ترتیب زیر اصلاح کنید:

```
var booktitles = from b in books
join s in sales on b.ID equals s.ID
where s.pages > 500
select new { Name = b.Title, Pages = s.pages };
```

query (پرسمان) تنها آن دسته از سطرهایی را برمی گرداند که تعداد صفحات آن بالغ بر 500 باشد:

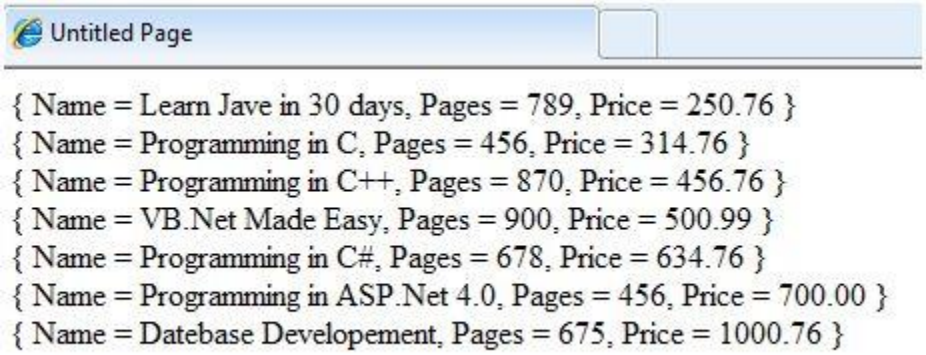


دستورهای Orderby و Orderbydescending

دستورها ی نام برده امکان مرتب سازی نتایج query را فراهم می آورد. به منظور query گرفتن از عنوان ها، تعداد صفحات و قیمت کتاب ها، کد زیر را در Page_Load event handler وارد کنید:

```
var booktitles = from b in books
join s in sales on b.ID equals s.ID
orderby b.Price
select new { Name = b.Title, Pages = s.pages, Price = b.Price };
```

تاپل های بازگشتی به صورت زیر می باشند:



دستور Let

دستور let امکان تعریف متغیر و تخصیص یک مقدار (مقداری که از مقادیر داده ها محاسبه شده) به آن متغیر را فراهم می کند. از این دستور برای مقدار دهی به یک خانه حافظه استفاده می کنیم. به عنوان مثال برای محاسبه ی فروش کل از دو فروش قبلی باید بدین ترتیب عمل کنید:

```
TotalSale = Price of the Book * Sales
```

برای نیل به این هدف لازم است تکه کدهای زیر را به داخل Page_Load event handler اضافه کنید:

```
var booktitles = from b in book
join s in sales on b.ID equals s.ID
let totalprofit = (b.Price * s.sales)
select new { Name = b.Title, TotalSale = totalprofit };
```

نتیجه:

{ Name = Programming in C#, TotalSale = 69823600.00 }
{ Name = Learn Jave in 30 days, TotalSale = 15045600.00 }
{ Name = Programming in ASP.Net 4.0, TotalSale = 28000000.00 }
{ Name = VB.Net Made Easy, TotalSale = 40079200.00 }
{ Name = Programming in C, TotalSale = 28328400.00 }
{ Name = Programming in C++, TotalSale = 22838000.00 }
{ Name = Datebase Developement, TotalSale = 40030400.00 }