عنوان: **خلاصه‌ای از LINQ to XML**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۵/۲۱ ۱۲:۵۷:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: LINQ

در این مقاله مروری سریع و کاربردی خواهیم داشت بر توانایی‌های مقدماتی LINQ to XML .

فایل Employee.XML را با محتویات زیر در نظر بگیرید:

```
<Employees>
     <Employee>
         <Name>Vahid</Name>
         <Phone>11111111</Phone>
         <Department>IT</Department>
  <Age>52</Age>
     </Employee>
     <Employee>
         <Name>Farid</Name>
         <Phone>124578963</Phone>
         <Department>Civil</Department>
        <Age>35</Age>
     </Employee>
     <Employee>
         <Name>Mehdi</Name>
         <Phone>1245788754</Phone>
         <Department>HR</Department>
        <Age>30</Age>
     </Employee>
 </Employees>
```

1 - چگونه یک فایل XML را جهت استفاده توسط LINQ بارگذاری کنیم؟

قبل از شروع، اسمبلی System.Xml.Linq باید به ارجاعات برنامه اضافه شود. سپس:

```
using System.Xml.Linq;

XDocument xDoc = XDocument.Load("Employee.xml");
```

2 - اگر محتویات XML دریافتی به صورت رشته بود (مثلا از یک دیتابیس دریافت شد)، اکنون چگونه باید آن‌را بارگذاری کرد؟

این‌کار را با استفاده از یک StringReader به صورت زیر می‌توان انجام داد:

```
// loading XML from string
StringReader sr = new StringReader(stringXML);
XDocument xDoc = XDocument.Load(sr);
```

3- چگونه یک کوئری ساده شامل تمامی رکوردهای Employee مجموعه Employees را تهیه کنیم؟

```
using System.Collections;

IEnumerable<XElement> empList = from e in xDoc.Root.Elements("Employee") select e;
```

توسط کوئری فوق، تمامی رکوردهای کارکنان در یک Collection در اختیار ما خواهند بود. نکته‌ی مهم عبارت LINQ فوق،

xDoc.Root.Elements("Employee") می‌باشد. به این صورت از xDoc بارگذاری شده، ابتدا Root و یا همان محتوای فایل XML را جهت بررسی انتخاب کرده و سپس گره‌های مرتبط با کارکنان را انتخاب می‌کنیم.
اکنون که مجموعه کارکنان توسط متغیر empList در اختیار ما است، دسترسی به محتویات آن به سادگی زیر خواهد بود:

```
foreach (XElement employee in empList)
{
    foreach (XElement e in employee.Elements())
    {
         Console.WriteLine(e.Name + " = " + e.Value);
    }
}
```

در این‌جا حلقه خارجی اطلاعات کلی تمامی کارکنان را باز می‌گرداند و حلقه داخلی اطلاعات یک گره دریافت شده را نمایش می‌دهد.

4 - کوئری بنویسید که اطلاعات تمامی کارکنان بخش HR را باز گرداند.

```
IEnumerable<XElement> hrList = from e in xDoc.Root.Elements("Employee")
                                                  where e.Element("Department").Value == "HR"
                                                  select e;
```

همانطور که ملاحظه می‌کنید همانند عبارات SQL ، در تمامی عناصر متعلق به کارکنان، عناصری که دپارتمان آن‌ها مساوی HR است بازگشت داده می‌شود.

-5 کوئری بنویسید که لیست تمامی کارکنان بالای 30 سال را ارائه دهد.

```
IEnumerable<XElement> tList = from e in xDoc.Root.Elements("Employee")
                                                   where int.Parse(e.Element("Age").Value) > 30
                                                   select e;
```

چون حاصل e.Element("Age").Value یک رشته است، برای اعمال فیلترهای عددی باید این رشته‌ها تبدیل به عدد شوند. به همین جهت از int.Parse استفاده شده است.

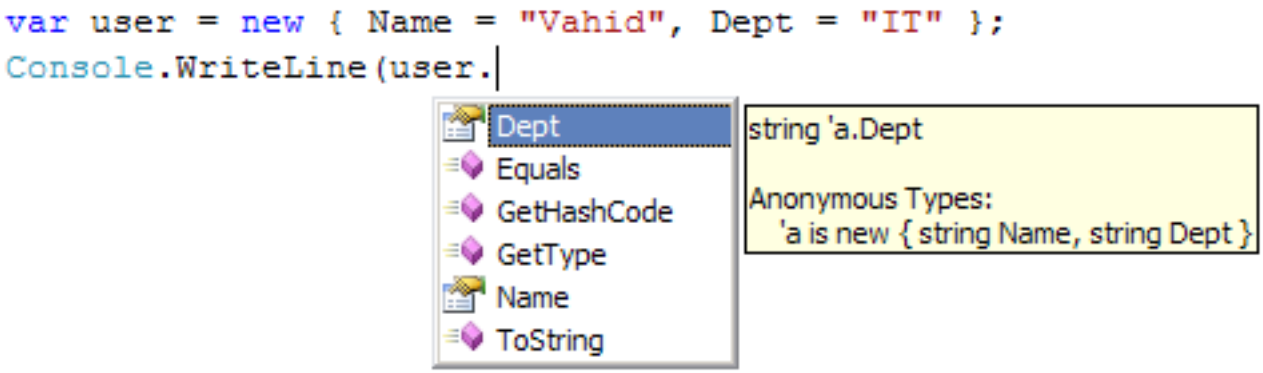
6 - کوئری بنویسید که لیست تمامی کارکنان بالای 30 سال را مرتب شده بر اساس نام باز گرداند.

```
IEnumerable<XElement> tList = from e in xDoc.Root.Elements("Employee")
                                                  where int.Parse(e.Element("Age").Value) > 30
                                                  orderby e.Element("Name").Value
                                                  select e;
```

در اینجا همانند عبارات SQL از orderby جهت مرتب سازی بر اساس عناصر نام استفاده شده است.

7 - تبدیل نتیجه‌ی یک کوئری LINQ به لیستی از اشیاء

مفهومی به سی شارپ 3 اضافه شده است به نام anonymous types . برای مثال:



توسط این قابلیت می‌توان یک شیء را بدون نیاز به تعریف ابتدایی آن ایجاد کرد و حتی از intelliSense موجود در IDE نیز بهره مند شد. این نوع‌های ناشناس توسط واژه‌های کلیدی new و var تولید می‌شوند. کامپایلر به صورت خودکار برای هر anonymous type یک کلاس ایجاد می‌کند.

دقیقا از همین توانایی در LINQ نیز می‌توان استفاده نمود:

```
var empList = from e in xDoc.Root.Elements("Employee")
                          orderby e.Element("Name").Value
                          select new
                          {
                              Name = e.Element("Name").Value,
                              Phone = e.Element("Phone").Value,
                              Department = e.Element("Department").Value,
                              Age = int.Parse(e.Element("Age").Value)
                          };
```

در این‌جا حاصل کوئری، تبدیل به لیستی از اشیاءanonymous می‌شود. اکنون برای نمایش آن‌ها نیز می‌توان از واژه کلیدی var استفاده نمود که از هر لحاظ نسبت به روش اعمال foreach بر روی Xelement ها که در مثال 3 مشاهده کردیم خواناتر است:

```
foreach (var employee in empList)
            {
                Console.WriteLine("Name = " + employee.Name);
                Console.WriteLine("Dep = " + employee.Department);
                Console.WriteLine("Phone = " + employee.Phone);
                Console.WriteLine("Age = " + employee.Age);
            }
```

و البته بدیهی است که می‌توان از anonymous types استفاده نکرد و دقیقا تعریف شیء را پیش از انتخاب آن نیز مشخص نمود. برای مثال:

```
public class Employee
   {
        public string Name { get; set; }
        public string Phone { get; set; }
        public string Department { get; set; }
        public int Age { get; set; }
  }
```

در این حالت، قسمت select new عبارت LINQ ما به select new Employee تغییر خواهد کرد.

برای مثال اگر بخواهیم لیست دریافتی را به صورت یک لیست جنریک بازگشت دهیم خواهیم داشت:

```
public class Employee
        {
            public string Name { get; set; }
            public string Phone { get; set; }
```

```
        public string Department { get; set; }
        public int Age { get; set; }
    }

    List<Employee> Get()
    {
        XDocument xDoc = XDocument.Load("Employee.xml");
        var items =
            from e in xDoc.Root.Elements("Employee")
            orderby e.Element("Name").Value
            select new Employee
            {
                Name = e.Element("Name").Value,
                Phone = e.Element("Phone").Value,
                Department = e.Element("Department").Value,
                Age = int.Parse(e.Element("Age").Value)
            };
        return items.ToList();
    }
```

نظرات خوانندگان

نویسنده: Anonymous
تاریخ: ۱۳۸۸/۰۵/۲۱ ۱۳:۰۸:۰۷

برای تکمیل بحث با اجازه آقای نصیری این لینک هم من اضافه می کنم که قابلیت های خارق العاده LINQ رو برای کار با XML نشون می ده :

http://windowsclient.net/learn/video.aspx?v=6895

موفق باشید.

نویسنده: علی اقدم
تاریخ: ۱۳۸۹/۰۷/۰۴ ۱۲:۲۷:۳۵

بسیار مفید بود.ممنون.