تکنیک‌های کاربردی استخراج داده‌ها با لینک (بخش چهارم)

در شماره گذشته با تعدادی از محاوره‌های کاربردی لینک در ارتباط با فایل‌های XML آشنا شدیم. اسناد XML با توجه به آنکه بر مبنای یک استاندارد جهانی آماده می‌شوند، همواره به یک شکل مورد استفاده قرار می‌گیرند. نکته جالب توجهی که در ارتباط با فایل‌های XML وجود دارد این است که حتی کاربران عادی نیز می‌توانند این اسناد را مشاهده یا ایجاد کنند. اما برنامه‌نویسان با توجه به حوزه کاری خود باید با شیوه ساخت و به‌کارگیری این اسناد در برنامه‌های کاربردی آشنایی داشته باشند. در شماره گذشته تاحدی با تکنیک‌های مربوط به XML آشنا شدیم. در این شماره با چند تکنیک کوتاه اما مهم که در ارتباط با اسناد XML وجود دارد، آشنا خواهیم شد.

دریافت داده‌ها از فایل‌های XML

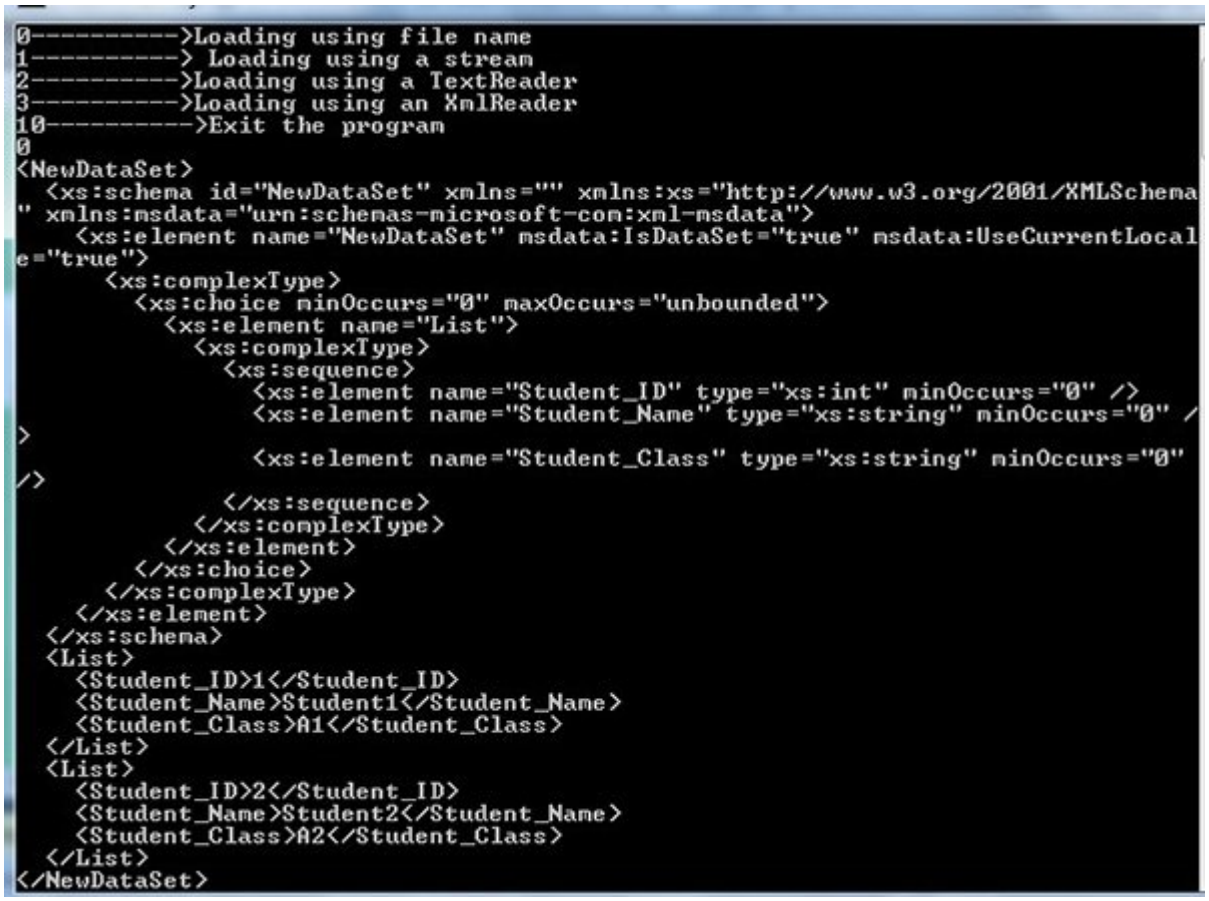
لینک کلاس‌های خاص خود را برای دستکاری داده‌های XML در اختیار دارد. این کلاس‌ها برای آنکه در زمان کار کردن با محاوره‌ها بهترین کارایی را داشته باشند، به‌شکل خوبی بهینه‌سازی شده‌اند. System.Xml.Linq.XElement یکی از کاربردی‌ترین کلاس‌های لینک است که برای کار کردن با اسناد XML مورد استفاده قرار می‌گیرد. این کلاس متشکل از توابع ایستایی است که به شما اجازه می‌دهد با فایل‌های XML در تعامل باشید. کلاس XElement به‌منظور نمایش یک عنصر XML مورد استفاده قرار می‌گیرد. برنامه‌نویسان می‌توانند از این کلاس به‌منظور ساخت عناصر؛ تغییر محتوای عناصر؛ اضافه، تغییر یا حذف عناصر فرزند؛ اضافه کردن خصلت‌هایی به یک عنصر؛ سریالیز کردن محتوای یک عنصر در یک فرمت متنی و... استفاده کنند. Load یکی از متدهای پرکاربرد کلاس فوق است. این متد به شما اجازه بارگذاری و تجزیه یک سند XML را می‌دهد. خروجی این متد یک نمونه کلاس XElement است که گره ریشه را مشخص می‌کند. برنامه‌نویسان قادر هستند به روش‌های مختلفی از این متد استفاده کنند. فهرست یک نحوه خواندن محتوای یک فایل XML را با استفاده از متد Load به شکل‌های مختلف نشان می‌دهد. ما برای راحتی کار این متد را درون یک بلوک Switch مورد استفاده قرار داده‌ایم. (در این مثال، فرض شده است فایلی به‌نام myList.xml درون مسیر تعیین شده وجود دارد.)

فهرست یک:

```
using System;
using System.Xml;
using System.Xml.Linq;
using System.IO;
        static void Main(string[] args)
        {

            string filename = @"c:\myList.xml";
            Byte m_case;
            XElement root;
```

```
            while (true)
            {
            Console.WriteLine("0---------->Loading using file name");
            Console.WriteLine("1----------> Loading using a stream");
            Console.WriteLine("2---------->Loading using a TextReader");
            Console.WriteLine("3---------->Loading using an XmlReader");
            Console.WriteLine("10---------->Exit the program");
            m_case= Convert.ToByte(Console.ReadLine());
            switch(m_case)
            {
                case 0:
            root = XElement.Load(filename);

            Console.WriteLine(root);
                    break;
                case 1:
            FileStream filestream = File.OpenRead(filename);
            root = XElement.Load(filestream);

            Console.WriteLine(root);
                    break;
                case 2:
                    TextReader reader = new StreamReader(filename);
                    root = XElement.Load(reader);
                    Console.WriteLine(root);
                    break;
                case 3:
                    XmlReader xmlreader = new XmlTextReader(new StreamReader(filename));
                    root = XElement.Load(xmlreader);
                    Console.WriteLine(root);
                    break;
                case 10:
                    return;
            }
        }
}
```

در فهرست یک، فرض بر این بوده است که فایل myList.xml در مسیر درایو C قرار دارد. در ادامه بلوک Switch به روش‌های مختلف این فایل را فراخوانی می‌کند و محتوای درون فایل‌ها را نشان می‌دهد. برای آنکه بتوانید این قطعه کد را آزمایش کنید، کافی است مسیر و نام فایل XML خود را جایگزین مقدار پیش‌فرض کنید. شکل 1 خروجی فهرست یک را نشان می‌دهد.



□□□ *1-* □□□□□□□□□ □□□□□□□□ *XML* □□ □□□□□□□ □□□□□.

جست‌وجو به‌منظور پیدا کردن مقداری خاص در یک سند XML

بعضی مواقع لازم است یک سند XML را برای پیدا کردن مقدار خاصی مورد جست‌وجو قرار دهید. این جست‌وجو ممکن است بر مبنای نام یا خصیصه یا مجموعه‌ای از خصایص انجام شود. در چنین زمان‌هایی به‌سادگی می‌توانید یک درخت XML را در قالب یک منبع داده‌ای برای یک محاوره لینک تعریف کنید. برای آنکه بتوانید یک درخت XML را به‌عنوان یک منبع داده‌ای برای یک محاوره لینک تعریف کنید، باید به‌ترتیبی که در ادامه به آن اشاره می‌کنیم عمل کنید.

ابتدا فایل مورد نظر را فراخوانی کنید. در ادامه محاوره لینک را همراه با عناصر و مقادیری که به‌دنبال آن‌ها هستید، تعریف کنید. در این محاوره شرطی که باید اعمال شود با استفاده از کلمه کلیدی where تعریف می‌شود. در ادامه عناصری که به‌عنوان نتیجه محاوره باید بازگردانده شود را با استفاده از کلمه کلیدی select تعریف کنید. در نهایت الگویی که بر مبنای آن نتایج مرتب می‌شوند را با استفاده از کلمه کلیدی orderby مشخص کنید. البته مرتب‌سازی یک گزینه اختیاری است.

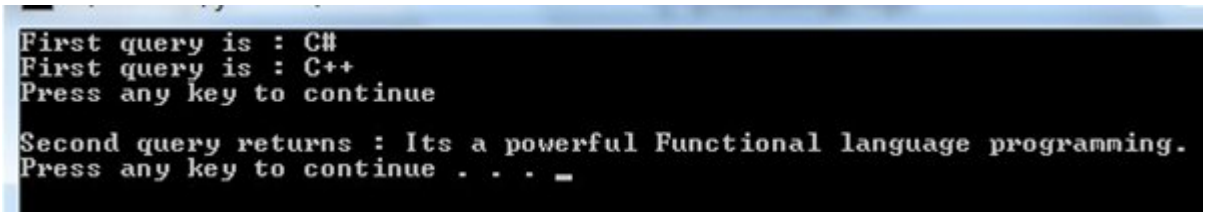
فهرست دو نحوه بارگذاری یک درخت XML را که در فایل myxml.xml ذخیره شده است نشان می‌دهد. در این مثال عناصری که در زیرشاخه برنامه‌نویسی قرار داشته و مقدار فیلد Rate آن‌ها برابر با 1 باشد انتخاب شده و در ادامه مقدار قرار گرفته در خصلت LangaugeName چاپ می‌شود. در محاوره دوم عناصری که Rate آن‌ها برابر با 2 باشد انتخاب شده و این مرتبه خصلت Describ آن‌ها چاپ می‌شود. اما برای آنکه دستورات فهرست دو به‌درستی اجرا شوند، ابتدا فایل myxml.xml را در سیستم خود بسازید و دستوراتی که در فهرست دو مشاهده می‌کنید را در برنامه خود وارد کنید.

فهرست دو:

```
<?xml version=”1.0” encoding=”utf-8”?>
<ComputersWorld>
```

```
  <Programming>
    <LanguageName>C#</LanguageName>
    <Describ>Its a powerful language programming of .Net’s family.</Describ>
    <Rate>1</Rate>
    <Rich>true</Rich>
  </Programming>
  <Programming>
    <LanguageName>C++</LanguageName>
    <Describ>Its a powerful language programming</Describ>
    <Rate>1</Rate>
    <Rich>true</Rich>
  </Programming>
  <Programming>
    <LanguageName>F#</LanguageName>
    <Describ>Its a powerful Functional language programming.</Describ>
    <Rate>2</Rate>
    <Rich>true</Rich>
  </Programming>
  <Database>
    <DatabaseName>SqlServer</DatabaseName>
    <Describ>Its a good and secure DBMS.</Describ>
    <Rate>1</Rate>
    <Rich>true</Rich>
  </Database>
  <Security>
    <OperatingSystem>Windowsr</OperatingSystem>
    <Describ>It has high security feature</Describ>
    <Rate>1</Rate>
    <Rich>true</Rich>
  </Security>
</ComputersWorld>
```

حال برای آنکه بتوانید داده‌های درون فایل XML را مشاهده کنید دستوراتی که در فهرست سه مشاهده می‌کنید را در برنامه کاربردی خود وارد کنید. خروجی فهرست سه را در شکل 2 مشاهده می‌کنید.



□□□ 2- □□□□□□□□ □□□□□□□ □□□□□ □□□□□□□ □□□□□□□□ XML.

فهرست سه:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
```

```
using System.Xml.Linq;
        static void Main(string[] args)
        {

            XElement rootElement = XElement.Load(@"c:\myxml.xml");
                IEnumerable<string> query1 = from elem in rootElement.Elements()
where (elem.Name == "Programming" && ((string)elem.Element("Rate"))== "1")
                   select ((string)elem.Element("LanguageName"));
                foreach (string str1 in query1)
                {
                    Console.WriteLine("First query is : {0}", str1);
                }

                Console.WriteLine("Press any key to continue");
                Console.ReadLine();

                IEnumerable<string> query2 = rootElement.Elements().Where(e
                => e.Name == "Programming"
                && (string)e.Element("Rate") == "2").Select(
                e => (string)e.Element("Describ"));
                foreach (string str1 in query2)
                {
         Console.WriteLine("Second query returns : {0}", str1);
                }
}
```

ویرایش یک درخت XML با استفاده از محاوره‌های لینک

در اکثر مواقع نیاز داریم تا عملیاتی همچون اضافه، حذف یا ویرایش عناصر درون یک درخت XML را در برنامه خود وارد کنیم. برای این منظور متدهایی همچون Add یا Remove که در کلاس XElement قرار دارند مورد استفاده قرار می‌گیرند. برای انجام چنین کارهایی در گام نخست باید عنصر مورد نظر خود را پیدا کنید. با فراخوانی متدهای Attribute و Element این فرآیند به‌سادگی انجام می‌شود.

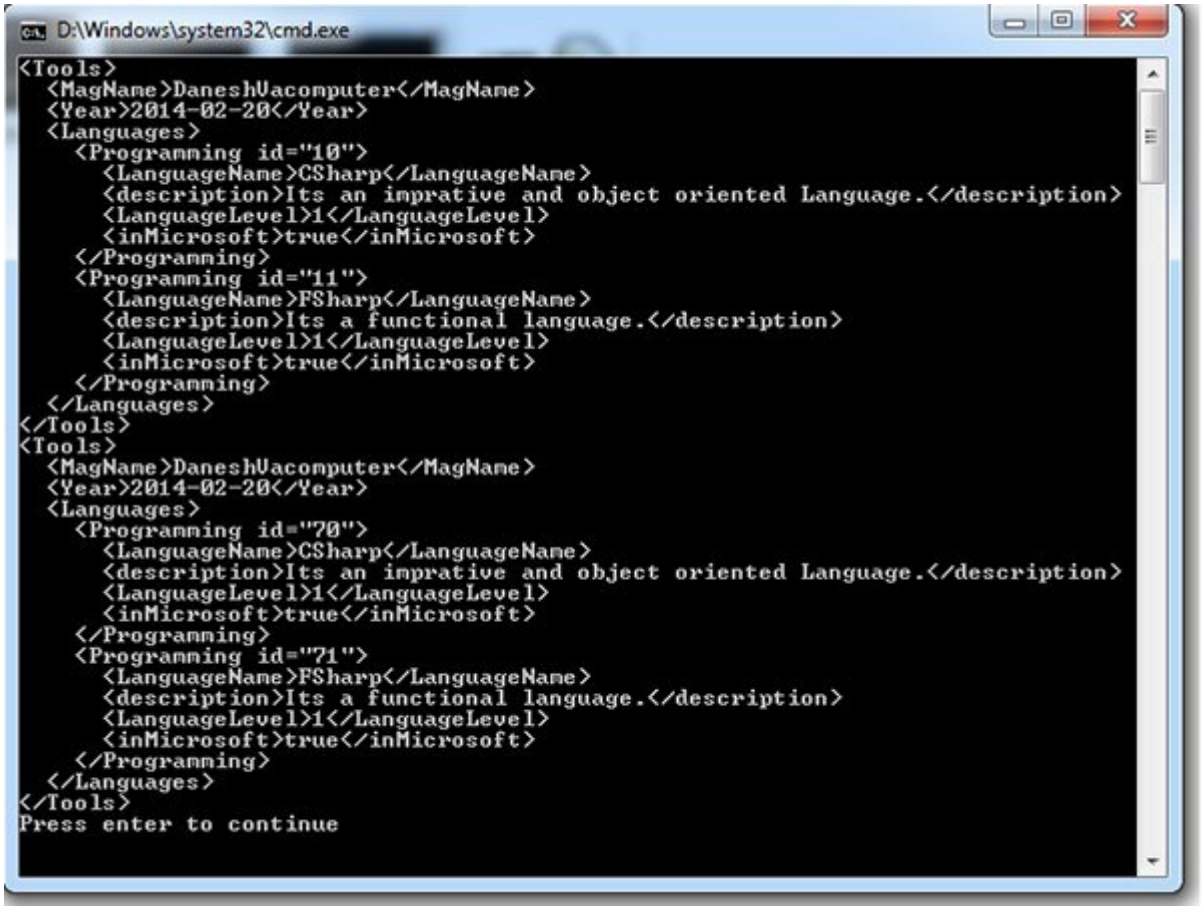
برای ویرایش یک عنصر ابتدا محاوره لینک را در ارتباط با عنصر مورد نظر تعریف کنید و در مرحله بعد متدهای ReplaceAttributes یا ReplaceNodes را برای ویرایش عنصر پیدا شده فراخوانی کنید. همچنین، با فراخوانی متد ReplaceWith می‌توانید مقدار عنصر جاری را با مقدار جدید تعویض کنید. برای حذف یک عنصر کافی است متد Remove را روی یک نمونه از کلاس XElement فراخوانی و عنصر پیدا شده را حذف کنید. همانند مثال قبل برای آنکه در اجرای کدهای این بخش مشکلی نداشته باشید، ابتدا فایلی به‌نام myxml.xml را در درایو C ایجاد و دستورات فهرست پنج را در این فایل وارد کنید و سپس آن را ذخیره‌سازی کنید. (دقت کنید فایل را با پسوند xml ذخیره کنید.)

فهرست پنج:

```
<?xml version="1.0" ?>
<!--An easy eaxmple of XML-->
<Tools>
<MagName>DaneshVacomputer</MagName>
<Year>2014-02-20</Year>
<Languages>
<Programming id="10">
<LanguageName>CSharp</LanguageName>
```

<description>Its an imprative and object oriented Language.</description>
<LanguageLevel>1</LanguageLevel>
<inMicrosoft>true</inMicrosoft>
</Programming>
<Programming id=”11”>
<LanguageName>FSharp</LanguageName>
<description>Its a functional language.</description>
<LanguageLevel>1</LanguageLevel>
<inMicrosoft>true</inMicrosoft>
</Programming>
</Languages>
</Tools>

حال زمان آن رسیده است تا مجموعه دستوراتی که در فهرست شش مشاهده می‌کنید را در برنامه خود وارد کنید. این دستورات به‌منظور جست‌وجو، ویرایش و اضافه کردن عناصر در یک سند XML مورد استفاده قرار می‌گیرند. خروجی قطعه کد فوق را در شکل 3 مشاهده می‌کنید.



□□□ *3-* □□□□□□□ □□□□□ □□□□□ □□□□□□□□ *XML* □□□□□□□□ □□□□□ □□□□□□ □□□□□□□ □□□□□.

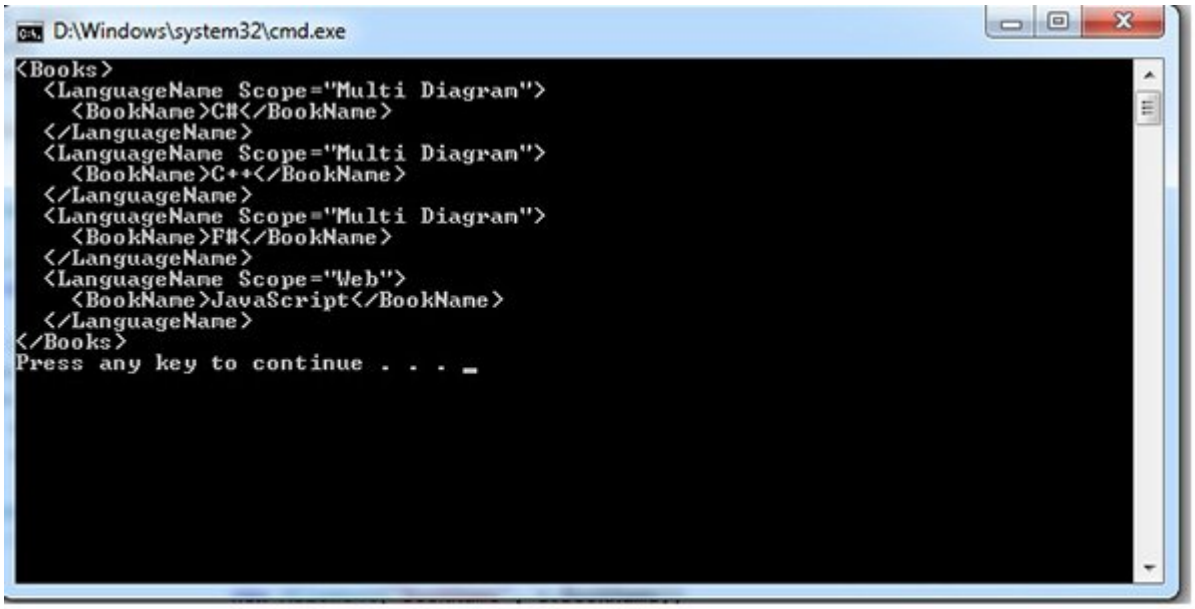
فهرست شش:

```
using System.Collections.Generic;
using System.Data.Linq.Mapping;
using System.Data.Linq;
using System.Linq;
using System;
```

```
using System.Xml;
using System.Xml.Linq;

namespace ConsoleApplication8
{
    class Program
    {
        static void Main(string[] args)
        {
            XElement root = XElement.Load(@"c:\myxml.xml");
            Console.WriteLine(root);

            IEnumerable<XElement> Elements
            = from elem in root.Element("Languages").Elements()
              where (elem.Name == "Programming")
              select elem;
            foreach (XElement elem in Elements)
            {
                int Value = Byte.Parse((string)elem.Attribute("id"));
       elem.ReplaceAttributes(new XAttribute("id", Value + 60));
            }
            Console.WriteLine(root);
            Console.WriteLine("Press enter to continue");
            Console.ReadLine();
    IEnumerable<XElement> descrpElement = from elem in
                                              root.Element("Languages").Elements()
                                          where
(((string)elem.Element("description")).Contains("Language"))
                                          select elem;
            foreach (XElement elem in descrpElement)
            {
                elem.Remove();
            }
            Console.WriteLine(root);
            Console.WriteLine("Press enter to continue");
            Console.ReadLine();

            XElement newElement = new XElement("Programming",
            new XAttribute("id", 12),
            new XElement("LanguageName", "C++ Language"),
            new XElement("description",
            "Ita an high level language"),
            new XElement("inMicrosoft", true));
            root.Element("Languages").Add(newElement);
            Console.WriteLine(root);
        }
    }
}
```

نحوه ساخت یک سندXML با استفاده از آرایه‌ها

نکته‌ای که لازم است به آن توجه داشته باشید این است که یک سند XML همیشه به‌شکل ایستا ایجاد نمی‌شود. در بسیاری از موارد ممکن است یک فایل XML از دل یک برنامه کاربردی به وجود آید. به‌طور مثال، ممکن است به‌عنوان یک برنامه‌نویس داده‌های خود را از منابعی همچون آرایه‌ها، اشیا ADO.Net، فایل‌های داده‌ای یا هر گونه منبعی که اطلاع دقیقی درباره آن ندارید دریافت کنید. راهکارهای مختلفی برای دریافت این داده‌ها وجود دارد و به شکل‌های مختلفی این داده‌ها را می‌توان در LINQ to XML مورد استفاده قرار داد. لینک به برنامه‌نویسان این توانایی را می‌دهد تا محاوره‌های لینک را به‌طور مستقیم درون سازنده XElement قرار دهند. فهرست هفت شیوه ساخت یک فایل XML را با استفاده از مقادیر قرار گرفته درون آرایه‌ها نشان می‌دهد. شکل 4 خروجی این قطعه کد را نشان می‌دهد.



□□□ *4-* □□□□ □□□□□□□□ □□ *XML* □□□□□□□□ □□ □□□□□□□□ □□□□ □□□□□□□□□.

فهرست هفت:

```
using System.Data.Linq.Mapping;
using System.Data.Linq;
using System.Linq;
using System;
using System.Xml;
using System.Xml.Linq;
        static void Main(string[] args)
        {

var Languages = new[] {
new { BookName = “C#”, Scope = “Multi Diagram”},
new { BookName = “C++”,Scope = “Multi Diagram” },
new { BookName  = “F#”, Scope = “Multi Diagram” },
new { BookName  = “JavaScript”,Scope = “Web”}
};
XElement newelement =
new XElement(“Books”, from c in Languages
select new XElement(“LanguageName”, new XAttribute(“Scope”, c.Scope),
new XElement(“BookName”, c.BookName))
```

```
);
Console.WriteLine(newelement);
}
```

در شماره آینده تکنیک‌های دیگر مرتبط با محاوره‌های Linq to XML را مورد بررسی قرار خواهیم داد.

تاریخ انتشار:

12 اردیبهشت 1396

نشانی منبع:
https://www.shabakeh-mag.com/workshop/7707/%D8%AA%DA%A9%D9%86%DB%8C%DA%A9%E2%80%8C%D9%87%D8%A7%DB%8C-%DA%A9%D8%A7%D8%B1%D8%A8%D8%B1%D8%AF%DB%8C-%D8%A7%D8%B3%D8%AA%D8%AE%D8%B1%D8%A7%D8%AC-%D8%AF%D8%A7%D8%AF%D9%87%E2%80%8C%D9%87%D8%A7-%D8%A8%D8%A7-%D9%84%DB%8C%D9%86%DA%A9-%D8%A8%D8%AE%D8%B4-%DA%86%D9%87%D8%A7%D8%B1%D9%85