禪

دسترسی به منابع دادهای با LINQ

# تکنیکهای کاربردی استخراج دادهها با لینک (بخش پنجم)


.NET Language-Integrated Query (LINQ)
C#
VB
Others…
.NET Language-Integrated Query
LINQ enabled data sources
LINQ enabled ADO.NET
LINQ To Objects
LINQ To Datasets
LINQ To SQL
LINQ To Entities
LINQ To XML
Objects
Relational
<book>
<title/>
<author/>
<price/>
</book>
XML

در شماره گذشته، با تعدادی از محاورههای کاربردی لینک در ارتباط با فایلهای XML آشنا شدیم . در شماره گذشته نحوه دریافت دادهها از اسناد XML، جستوجو در اسناد بهمنظور پیدا کردن فیلد خاص، ویرایش اسناد XML و نحوه ساخت فایلهای XML با استفاده از آرایهها را همراه با مثالهای عملی مشاهده کردیم . در این مقاله تعداد دیگری از تکنیکهای پیشرفته کار با اسناد XML را مورد بررسی قرار خواهیم داد .

## چگونه میتوانیم یک سند **XML** را بارگذاری کنیم؟

XElements و XDocuments دو کلاس پرکاربرد هستند که متدهای لازم بهمنظور بارگذاری یا تجزیه یک سند را در اختیارتان قرار میدهند . Parse() از جمله متدهای پرکاربردی است که قادر است یک رشته فرمتبندی شده برمبنای ساختار XML را بهعنوان ورودی دریافت کند و نمونهای از کلاس XElements را بهعنوان مقدار خروجی برگرداند . متد Load() نیز به شما اجازه میدهد یک سند XML را بارگذاری کنید . این متدها به شما اجازه میدهد نام یک فایل یا استریم را بهعنوان ورودی مورد استفاده قرار دهید . فهرست 1 نحوه بهکارگیری این دو متد را نشان میدهد .

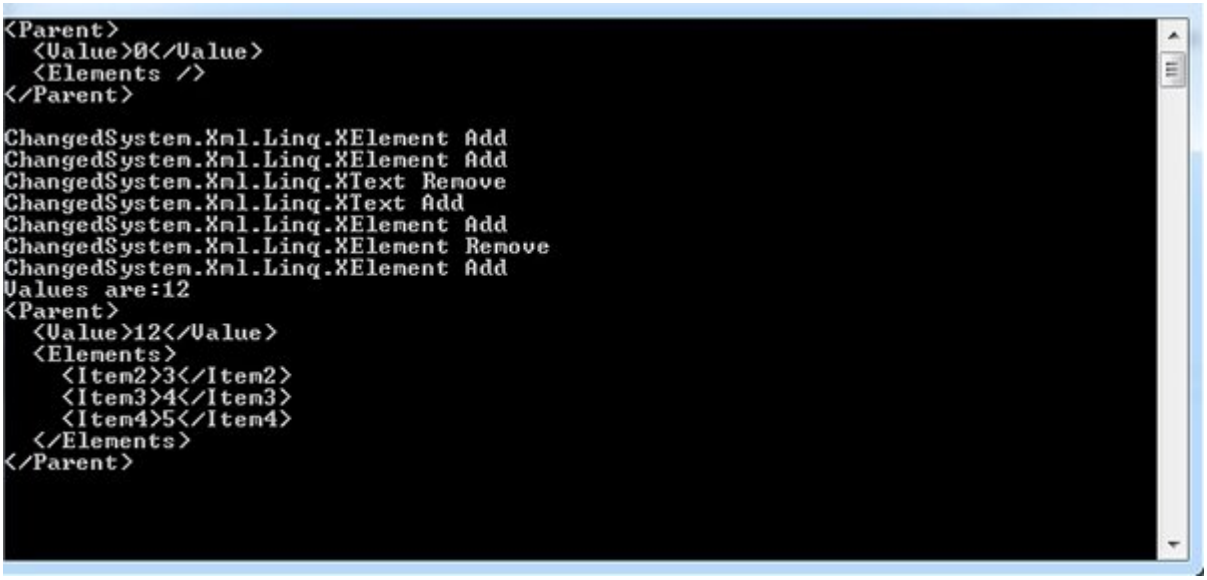
```
static void Main(string[] args)
{
```

```
        string myElement =
        @"<Language ID ='1'>
<HighLevel>Yes</HighLevel>
<Programming>CSharp</Programming>
</Language>";
        XElement newElement = XElement.Parse(myElement);
        Console.WriteLine(newElement);
        Console.WriteLine();
XDocument Doc = XDocument.Load("myxml.xml");
        Console.WriteLine(Doc);
    }
```

## XML [illegible]

[illegible]

```
    static void Main(string[] args)
    {
        XElement root = new XElement("Parent",
  new XElement("Value", "0"),
  new XElement("Elements"));
 XElement total = root.Element("Value");
XElement items = root.Element("Elements");
        items.Changed += (object sender, XObjectChangeEventArgs objectchangeevent) =>
        {
            switch (objectchangeevent.ObjectChange)
            {
                case XObjectChange.Add:
                    if (sender is XElement)
total.Value = ((int)total + (int)(XElement)sender).ToString();
                    if (sender is XText)
total.Value = ((int)total + (int)((XText)sender).Parent).ToString();
                    break;
                case XObjectChange.Remove:
                    if (sender is XElement)
 total.Value = ((int)total - (int)(XElement)sender).ToString();
                    if (sender is XText)
total.Value = ((int)total - Int32.Parse(((XText)sender).Value)).ToString();
                    break;
            }
            Console.WriteLine("Changed{0}{1}",sender.GetType().ToString(),
objectchangeevent.ObjectChange.ToString());
        };
        Console.WriteLine(root);
        Console.ReadLine();
        items.SetElementValue("Item1", 1);
```



```
           items.SetElementValue(“Item2”, 2);
           items.SetElementValue(“Item2”, 3);
           items.SetElementValue(“Item3”, 4);
          items.SetElementValue(“Item1”, null);
           items.SetElementValue(“Item4”, 5);
Console.WriteLine(“Values are:{0}”, (int)total);
           Console.WriteLine(root);
           Console.ReadLine();
       }
```



```
<Parent>
  <Value>0</Value>
  <Elements />
</Parent>

ChangedSystem.Xml.Linq.XElement Add
ChangedSystem.Xml.Linq.XElement Add
ChangedSystem.Xml.Linq.XText Remove
ChangedSystem.Xml.Linq.XText Add
ChangedSystem.Xml.Linq.XElement Add
ChangedSystem.Xml.Linq.XElement Remove
ChangedSystem.Xml.Linq.XElement Add
Values are:12
<Parent>
  <Value>12</Value>
  <Elements>
    <Item2>3</Item2>
    <Item3>4</Item3>
    <Item4>5</Item4>
  </Elements>
</Parent>
```

شکل *1* - لینک اجازه می دهد از زمان اعمال تغییرات روی گره ها اطلاع پیدا کنید .

## سریالیز کردن دادهها از طریق اعلان **XML**

سریالیز یا همان ذخیرهسازی دادهها درون یک فایل از طریق کلاس XElement و متد System.Xml.Linq.XElementSave یا از طریق کلاس XDocument و متد System.Xml.Linq.XDocument.Save که قادر به تولید یک اعلان XML است، انجام میشود. هر زمان فرآیند سریالیز کردن روی یک استریم XmlWriter به مرحله اجرا درمیآید، ابتدا این موضوع بررسی میشود که آیا یک اعلان XML وجود دارد یا خیر . در مرحله آخر متد Save فراخوانی میشود . البته به این نکته توجه داشته باشید نحوه بهکارگیری متد Save برای هریک از کلاسهای فوق متفاوت است.

متد Save در کلاس XDocument امکان ذخیرهسازی دادهها درون یک فایل را برمبنای یک استریم TextWriter یا XmlWriter امکانپذیر میسازد .

متد Save در کلاس XElement امکان ذخیرهسازی دادهها درون یک فایل را برمبنای یک استریم TextWriter، XMLTextWriter یا XmlWriter فراهم میآورد . فهرست 3 یک نمونه از کلاس XElement را تعریف و سند ایجاد شده را در یک فایل ذخیره میکند و سپس محتوای فایل ساخته شده را روی صفحهنمایش نشان میدهد.

```
using System.Xml.Linq;
using System.IO;
       static void Main(string[] args)
       {
           XElement root = new XElement(“Parent”,
     new XElement(“Child”, “child content”) );
           root.Save(“MyXml.xml”);
           string str = File.ReadAllText(“MyXml.xml”);
           Console.WriteLine(str);
       }
```

## ذخیرهسازی بدون تعریف اعلان یک سند **XML**

در مثال قبل، نحوه ذخیرهسازی دادهها از طریق تعریف اعلان XML را مشاهده کردیم . اما این امکان وجود دارد تا فرآیند ذخیرهسازی را بدون نیاز به تعریف اعلان XML انجام دهید . در این روش ابتدا باید نمونهای از کلاس XmlWriterSettings را تعریف کنید و در ادامه مقدار پیشفرض خاصیت OmitXmlDeclaration را در حالت true قرار دهید . تنظیم این خاصیت به حالت true به معنای آن است که ما به اعلان XML نیاز نداریم . پس از تعریف این کلاس و تنظیم خاصیت مربوط در ادامه متد Create از کلاس XmlWriter را فراخوانی میکنیم . متد فوق دو کلاس StringBuiler و XmlSettings را بهعنوان ورودی دریافت کرده است و کلاس XmlWriter را بهعنوان خروجی برمیگرداند . پس از این کار عناصر والد و فرزند XML ایجاد شده و در انتها خروجی کلاس XmlWriter بهعنوان پارامتری برای متد Save ارسال میشود. فهرست 4 نحوه بهکارگیری این متدها را نشان میدهد .

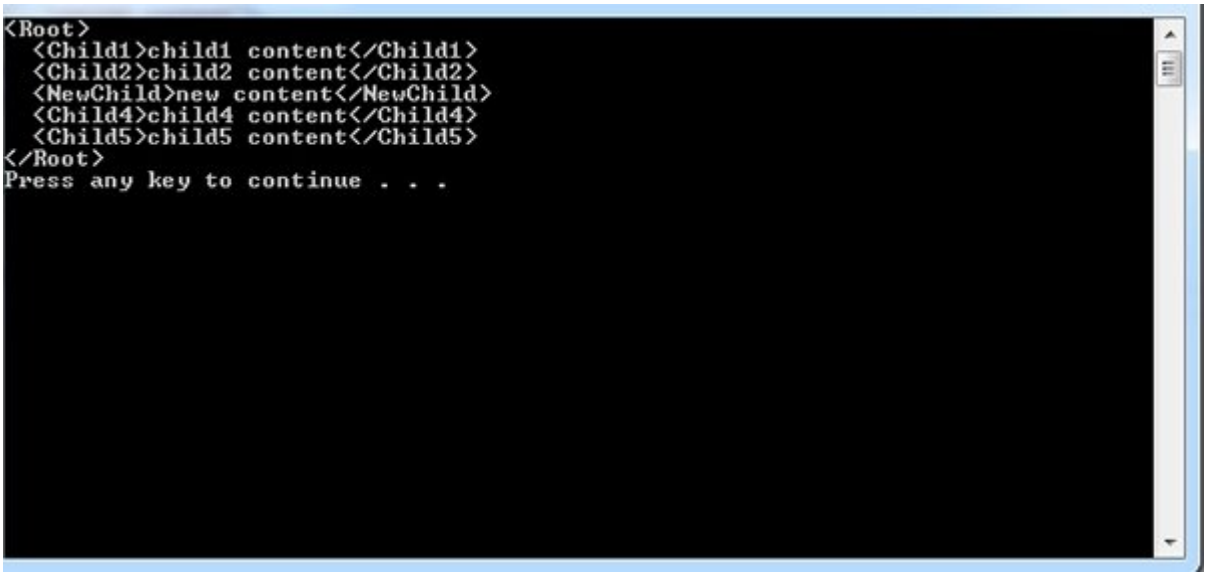
```
        static void Main(string[] args)
        {
  StringBuilder sb = new StringBuilder();
            XmlWriterSettings xmlwritersettings = new XmlWriterSettings();
xmlwritersettings.OmitXmlDeclaration = true;

            using (XmlWriter xmlwriter = XmlWriter.Create(sb, xmlwritersettings))
            {
              XElement root = new XElement(“Parent”,new XElement(“Child”, “child content”));
                root.Save(xmlwriter);
            }
            Console.WriteLine(sb.ToString());
        }
        }
```

## چگونه میتوانیم گرههایی را از یک عنصر **XElement** حذف کنیم؟

اگر بهدنبال حذف همه گرهها یا خصلتهای متعلق به یک شی XElement هستید، لینک متد RemoveAll() را در اختیار شما قرار میدهد . فهرست 5 عنصری را همراه با خصلتها و عناصر فرزند ایجاد میکند . در ادامه متد فوق را فراخوانی و عناصر فرزند و خصلتها را حذف میکند . خروجی قطعه کد زیر برابر با <Root /> خواهد بود . فهرست 5:

```
static void Main(string[] args)
        {
XElement root = new XElement(“Root”,
    new XAttribute(“Att1”, 1),
    new XAttribute(“Att2”, 2),
    new XAttribute(“Att3”, 3),
    new XElement(“Child1”, 1),
    new XElement(“Child2”, 2),
    new XElement(“Child3”, 3)
);
root.RemoveAll();
Console.WriteLine(root);
        }
```



```
<Root>
  <Child1>child1 content</Child1>
  <Child2>child2 content</Child2>
  <NewChild>new content</NewChild>
  <Child4>child4 content</Child4>
  <Child5>child5 content</Child5>
</Root>
Press any key to continue . . .
```

شکل 2 - جایگزین کردن مقدار جدید به جای مقدار فعلی

## جایگزین کردن مقدار جدید بهجای مقدار فعلی

بعضی مواقع لازم است محتوای یک گره خاص تغییر پیدا کند . لینک متد ReplaceWith را که متعلق به کلاس XNode است در اختیار برنامهنویسان قرار داده است . فهرست 6 نحوه جایگزین کردن محتوای فعلی با محتوای جدید را نشان میدهد . شکل 2 خروجی فهرست 6 را نشان میدهد .

```
static void Main(string[] args)
       {
   XElement xmlTree = new XElement(“Root”,
   new XElement(“Child1”, “child1 content”),
   new XElement(“Child2”, “child2 content”),
   new XElement(“Child3”, “child3 content”),
   new XElement(“Child4”, “child4 content”),
   new XElement(“Child5”, “child5 content”)

);
XElement child3 = xmlTree.Element(“Child3”);
child3.ReplaceWith(
   new XElement(“NewChild”, “new content”)
);
Console.WriteLine(xmlTree);
       }
```

## چگونه میتوانیم مقادیری را درون یک سند کدگذاری بنویسیم یا مقادیری را از درون یک سند کدگذاری بخوانیم؟

برای آنکه بتوانید یک سند XML کدگذاری شده را ایجاد کنید، باید یک نمونه از کلاس XDeclaration را به یک درخت XML اضافه و استاندارد کدگذاری (Encoding) مورد نظر خود را انتخاب کنید . در فهرست 7 دو سند ساخته میشود. سند اول برمبنای سیستم کدینگ utf-8 و دومی برمبنای سیستم کدگذاری utf-16 ساخته میشود. در ادامه این اسناد بارگذاری شده و محتوای آنها روی صفحهنمایش نشان داده میشود . اگر از متد Encoding در ارتباط با سندی استفاده کردید، در زمان سریالیز کردن دادهها، سند XML برمبنای انکدینگ تعیین شده ساخته میشود. شکل 3 خروجی فهرست 7 را نشان میدهد .

```
Console.WriteLine(“Creating a document with utf-8 encoding”);
XDocument encodedDoc8 = new XDocument(
   new XDeclaration(“1.0”, “utf-8”, “yes”),
   new XElement(“Root”, “Content”)
```

```
);
encodedDoc8.Save(“EncodedUtf8.xml”);
Console.WriteLine(“Encoding is:{0}”, encodedDoc8.Declaration.Encoding);
Console.WriteLine();

Console.WriteLine(“Creating a document with utf-16 encoding”);
XDocument encodedDoc16 = new XDocument(
   new XDeclaration(“1.0”, “utf-16”, “yes”),
   new XElement(“Root”, “Content”)
);
encodedDoc16.Save(“EncodedUtf16.xml”);
Console.WriteLine(“Encoding is:{0}”, encodedDoc16.Declaration.Encoding);
Console.WriteLine();
 XDocument newDoc8 = XDocument.Load(“EncodedUtf8.xml”);
Console.WriteLine(“Encoded document:”);
Console.WriteLine(File.ReadAllText(“EncodedUtf8.xml”));
Console.WriteLine();
Console.WriteLine(“Encoding of loaded document is:{0}”, newDoc8.Declaration.Encoding);
Console.WriteLine();
 XDocument newDoc16 = XDocument.Load(“EncodedUtf16.xml”);
Console.WriteLine(“Encoded document:”);
Console.WriteLine(File.ReadAllText(“EncodedUtf16.xml”));
Console.WriteLine();
Console.WriteLine(“Encoding of loaded document is:{0}”, newDoc16.Declaration.Encoding);
```



```
Creating a document with utf-8 encoding
Encoding is:utf-8

Creating a document with utf-16 encoding
Encoding is:utf-16

Encoded document:
<?xml version="1.0" encoding="utf-8" standalone="yes"?>
<Root>Content</Root>

Encoding of loaded document is:utf-8

Encoded document:
<?xml version="1.0" encoding="utf-16" standalone="yes"?>
<Root>Content</Root>

Encoding of loaded document is:utf-16
```

شکل 3 - تعیین *Encoding* برای سندی که ساخته خواهد شد.

## XML به منظور تبدیل یک درخت XSLT به کارگیری

XSLT (سرنام Extensible Stylesheet Language Transformations) زبانی است که بهمنظور تبدیل ساختار یک سند XML به انواع دیگری از اسناد مورد استفاده قرار میگیرد . این فناوری از سوی کنسرسیوم وب ابداع شده است. XSLT در مقایسه با استاندارد XML از انعطافپذیری بیشتری برخوردار بوده و قادر است یک سند XML را به یک سند HTML یا اسناد دیگر تبدیل کند . بهعنوان یک برنامهنویس این توانایی را دارید تا یک درخت XML را ایجاد کرده، یک نمونه از کلاس XmlReader را با فراخوانی متد createReader و برمبنای درخت XML موجود تعریف کرده، سند جدیدی ایجاد کرده و درنهایت یک شی

XmlWriter را که بهمنظور ساخت سند جدیدی مورد استفاده قرار میگیرد، تعریف کنید . در ادامه این توانایی را دارید تا از مبدل XSLT بهمنظور تبدیل یک سند XML به فرمتهای دیگر استفاده کنید . فهرست 8 نحوه بهکارگیری این تکنیک را نشان میدهد .

```
    static void Main(string[] args)
    {
        string xslstring= @”<?xml version=’1.0’?>
<xsl:stylesheet xmlns:xsl=’http://www.w3.org/1999/XSL/Transform’ version=’1.0’>
  <xsl:template match=’/Root’>
    <Parent>
        <A1>
        <xsl:value-of select=’Child1’/>
        </A1>
        <A2>
        <xsl:value-of select=’Child2’/>
        </A2>
    </Parent>
  </xsl:template>
</xsl:stylesheet>”;

        XDocument xmlTree = new XDocument(
            new XElement(“Root”,
  new XElement(“Item1”, “Item1 data”),
    new XElement(“Item2”, “Item2 data”)));

        XDocument newTree = new XDocument();
        using (XmlWriter writer = newTree.CreateWriter())
        {
            XslCompiledTransform xslt = new XslCompiledTransform();
            xslt.Load(XmlReader.Create(new StringReader(xslstring)));
            xslt.Transform(xmlTree.CreateReader(), writer);
        }
        Console.WriteLine(newTree);
    }
```

کلاس XslCompiledTransform بهمنظور تبدیل دادههای XML مورد استفاده قرار میگیرد . این کلاس از مفسر XSLT بهمنظور تبدیل دادهها استفاده میکند . متد Load موجود در کلاس XslCompiledTransform بهمنظور بارگذاری و کامپایل نمونهای که با استفاده از متد Create ساخته خواهد شد مورد استفاده قرار میگیرد . البته به این نکته توجه داشته باشید که متد Load موجود در کلاس xslCompiledTransform در مقایسه با متد Load موجود در کلاس XslTransform از سرعت پایینتری برخوردار است، بهواسطه آنکه XLST پیش از آنکه فرآیند بارگذاری را انجام دهد، ابتدا فرآیند کامپایل را اجرا میکند .

نویسنده :
حمیدرضا_تائبی
دسته بندی:
کارگاه
تاریخ انتشار :
11:28 - 17/03/1396
برچسب:
کدگذاری - بارگذاری - عناصر - ذخیره سازی - متد - استریم - LINQ

□□□□□
□□□□:https://www.shabakeh-mag.com/workshop/7998/%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-%D9%BE%D9%86%D8%AC%D9%85