عنوان: **تهیه خروجی XML از یک بانک اطلاعاتی، توسط EF Code first**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۵/۱۰ ۱۱:۳۵
آدرس: www.dotnettips.info
برچسب‌ها: Entity framework, xml, LINQ to XML

نگارش کامل SQL Server امکان تهیه خروجی XML از یک بانک اطلاعاتی را دارد . اما اگر بخواهیم از سایر بانک‌های اطلاعاتی که چنین توابع توکاری ندارند، استفاده کنیم چطور؟ برای تهیه خروجی XML توسط Entity framework و مستقل از نوع بانک اطلاعاتی در حال استفاده، حداقل دو روش وجود دارد:

الف) استفاده از امکانات Serialization توکار دات نت

```
using System.IO;
using System.Xml;
using System.Xml.Serialization;

namespace DNTViewer.Common.Toolkit
{
    public static class Serializer
    {
        public static string Serialize<T>(T type)
        {
            var serializer = new XmlSerializer(type.GetType());
            using (var stream = new MemoryStream())
            {
                serializer.Serialize(stream, type);
                stream.Seek(0, SeekOrigin.Begin);
                using (var reader = new StreamReader(stream))
                {
                    return reader.ReadToEnd();
                }
            }
        }
    }
}
```

در اینجا برای نمونه، لیستی از اشیاء مدنظر خود را تهیه کرده و به متد Serialize فوق ارسال کنید. نتیجه کار، تهیه معادل XML آن است.
امکانات سفارشی سازی محدودی نیز برای XmlSerializer درنظر گرفته شده است؛ برای نمونه قرار دادن ویژگی‌هایی مانند XmlIgnore بالای خواصی که نیازی به حضور آن‌ها در خروجی نهایی XML نمی‌باشد.

ب) استفاده از امکانات LINQ to XML دات نت

روش فوق بدون مشکل کار می‌کند، اما اگر بخواهیم قسمت Reflection خودکار ثانویه آن‌را (برای نمونه جهت استخراج مقادیر از لیست دریافتی) حذف کنیم، می‌توان از LINQ to XML استفاده کرد که قابلیت سفارشی سازی بیشتری را نیز در اختیار ما قرار می‌دهد (کاری که در سایت جاری برای تهیه خروجی XML از بانک اطلاعاتی آن انجام می‌شود).

```
private string createXmlFile(string dir)
        {
            var xLinq = new XElement("ArrayOfPost",
                        _blogPosts
                            .AsNoTracking()
                            .Include(x => x.Comments)
                            .Include(x => x.User)
                            .Include(x => x.Tags)
                            .OrderBy(x => x.Id)
                            .ToList()
                            .Select(x => new XElement("Post", postXElement(x)))
                            );

            var xmlFile = Path.Combine(dir, "dot-net-tips-database.xml");
            xLinq.Save(xmlFile);
```

```
            return xmlFile;
        }

        private static XElement[] postXElement(BlogPost x)
        {
            return new XElement[]
            {
                new XElement("Id", x.Id),
                new XElement("Title", x.Title),
                new XElement("Body", x.Body),
                new XElement("CreatedOn", x.CreatedOn),
                tagElement(x),
                new XElement("User",
                                new XElement("Id", x.UserId.Value),
                                new XElement("FriendlyName", x.User.FriendlyName))
            }.Where(item => item != null).ToArray();
        }

        private static XElement tagElement(BlogPost x)
        {
            var tags = x.Tags.Any() ?
                            x.Tags.Select(y =>
                                        new XElement("Tag",
                                                new XElement("Id", y.Id),
                                                new XElement("Name", y.Name)))
                                    .ToArray() : null;
            if (tags == null)
                return null;

            return new XElement("Tags", tags);
        }
```

خلاصه‌ای از نحوه تبدیل اطلاعات لیستی از مطالب را به معادل XML آن در کدهای فوق مشاهده می‌کنید. یک سری نکات ریز نیز باید در اینجا رعایت شوند:

1) کار با یک new XElement که دارای متد Save با فرمت XML نیز هست، شروع می‌شود. مقدار آن‌را مساوی یک کوئری از بانک اطلاعاتی قرار می‌دهیم. این کوئری چون قرار است تنها اطلاعاتی را از بانک اطلاعاتی دریافت کند و نیازی به تغییر در آن‌ها نیست، با استفاده از متد AsNoTracking ، حالت فقط خواندنی پیدا کرده است.

2) اطلاعاتی را که نیاز است در فایل نهایی XML وجود داشته باشند، تنها کافی است در قسمت Select این کوئری با فرمت new XElementهای تو در تو قرار دهیم. به این ترتیب قسمت Relection خودکار XmlSerializer روش مطرح شده در ابتدای بحث دیگر وجود نداشته و عملیات نهایی بسیار سریعتر خواهد بود.

3) چون در این حالت، کار انجام شده دستی است، باید نام‌های گره‌های صحیحی را انتخاب کنیم تا اگر قرار است توسط همان XmlSerializer مجددا کار serializer.Deserialize صورت گیرد، عملیات با شکست مواجه نشود. بهترین کار برای کم شدن سعی و خطاها، تهیه یک لیست اطلاعات آزمایشی و سپس ارسال آن به روش ابتدای بحث است. سپس می‌توان با بررسی خروجی آن مثلا دریافت که روش serializer.Deserialize به صورت پیش فرض به دنبال ریشه‌ای به نام ArrayOfPost برای دریافت لیستی از مطالب می‌گردد و نه Posts یا هر نام دیگری.

4) در کوئری LINQ to Entites نوشته شده، پیش از Select، یک ToList قرار دارد. متاسفانه EF اجازه استفاده مستقیم از Select هایی از نوع XElement را نمی‌دهد و باید ابتدا اطلاعات را تبدیل به LINQ to Objects کرد.

5) در حین تهیه XElementها اگر قرار است عنصری نال باشد، باید آن‌را در خروجی نهایی ذکر نکرد. به این ترتیب serializer.Deserialize بدون نیاز به تنظیمات اضافه‌تری بدون مشکل کار خواهد کرد. در غیراینصورت باید وارد مباحثی مانند تعریف یک فضای نام جدید برای خروجی XML به نام XSI رفت و سپس به کمک ویژگی‌ها، xsi:nil را به true مقدار دهی کرد. اما همانطور که در متد postXElement ملاحظه می‌کنید، برای وارد نشدن به مبحث فضای نام xsi، مواردی که null بوده‌اند، اصلا در آرایه نهایی ظاهر نمی‌شوند و نهایتا در خروجی، حضور نخواهند داشت. به این ترتیب متد ذیل، بدون مشکل و بدون نیاز به تنظیمات اضافه‌تری قادر است فایل XML نهایی را تبدیل به معادل اشیاء دات نتی آن کند.

```
using System.IO;
using System.Xml;
using System.Xml.Serialization;

namespace DNTViewer.Common.Toolkit
{
    public static class Serializer
```

```
    {
        public static T DeserializePath<T>(string xmlAddress)
        {
            using (var xmlReader = new XmlTextReader(xmlAddress))
                {
                    var serializer = new XmlSerializer(typeof(T));
                    return (T)serializer.Deserialize(xmlReader);
                }
        }
    }
}
```