عنوان: **ثبت لینک‌های غیرتکراری**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۹/۱۰ ۲۰:۱۵
آدرس: www.dotnettips.info
گروه‌ها: xml, LINQ to XML, OPML

ثبت لینک‌های مختلف در یک سیستم (مثلا قسمت به اشتراک گذاری لینک‌ها) در ابتدای کار شاید ساده به نظر برسد؛ خوب، هر صفحه‌ای که یک آدرس منحصربفرد بیشتر ندارد. ما هش این لینک را محاسبه می‌کنیم و بعد روی این هش، یک کلید منحصربفرد را تعریف خواهیم کرد تا دیگر رکوردی تکراری ثبت نشود. همچنین چون این هش نیز طول کوتاهی دارد، جستجوی آن بسیار سریع خواهد بود. واقعیت این است که خیر! این روش ناکارآمدترین حالت پردازش لینک‌های مختلف است.

برای مثال لینک‌های http://www.site.com و http://www.site.com/index.htm دو هش متفاوت را تولید می‌کنند اما در عمل یکی هستند. نمونه‌ی دیگر، لینک‌های http://www.site.com/index.htm و http://www.site.com/index.htm#section1 هستند که فقط اصطلاحا در یک fragment با هم تفاوت دارند و از این دست لینک‌هایی که باید در حین ثبت یکی درنظر گرفته شوند، زیاد هستند و اگر علاقمند به مرور آن‌ها هستید، می‌توانید به صفحه‌ی URL Normalization در ویکی‌پدیا مراجعه کنید.

اگر نکات این صفحه را تبدیل به یک کلاس کمکی کنیم، به کلاس ذیل خواهیم رسید:

```
using System;
using System.Web;

namespace OPMLCleaner
{
    public static class UrlNormalization
    {
        public static bool AreTheSameUrls(this string url1, string url2)
        {
            url1 = url1.NormalizeUrl();
            url2 = url2.NormalizeUrl();
            return url1.Equals(url2);
        }

        public static bool AreTheSameUrls(this Uri uri1, Uri uri2)
        {
            var url1 = uri1.NormalizeUrl();
            var url2 = uri2.NormalizeUrl();
            return url1.Equals(url2);
        }

        public static string[] DefaultDirectoryIndexes = new[]
            {
                "default.asp",
                "default.aspx",
                "index.htm",
                "index.html",
                "index.php"
            };

        public static string NormalizeUrl(this Uri uri)
        {
            var url = urlToLower(uri);
            url = limitProtocols(url);
            url = removeDefaultDirectoryIndexes(url);
            url = removeTheFragment(url);
            url = removeDuplicateSlashes(url);
            url = addWww(url);
            url = removeFeedburnerPart(url);
            return removeTrailingSlashAndEmptyQuery(url);
        }

        public static string NormalizeUrl(this string url)
        {
            return NormalizeUrl(new Uri(url));
        }

        private static string removeFeedburnerPart(string url)
        {
            var idx = url.IndexOf("utm_source=", StringComparison.Ordinal);
            return idx == -1 ? url : url.Substring(0, idx - 1);
        }

        private static string addWww(string url)
        {
            if (new Uri(url).Host.Split('.').Length == 2 && !url.Contains("://www."))
```

```
            {
                return url.Replace("://", "://www.");
            }
            return url;
        }

        private static string removeDuplicateSlashes(string url)
        {
            var path = new Uri(url).AbsolutePath;
            return path.Contains("//") ? url.Replace(path, path.Replace("//", "/")) : url;
        }

        private static string limitProtocols(string url)
        {
            return new Uri(url).Scheme == "https" ? url.Replace("https://", "http://") : url;
        }

        private static string removeTheFragment(string url)
        {
            var fragment = new Uri(url).Fragment;
            return string.IsNullOrWhiteSpace(fragment) ? url : url.Replace(fragment, string.Empty);
        }

        private static string urlToLower(Uri uri)
        {
            return HttpUtility.UrlDecode(uri.AbsoluteUri.ToLowerInvariant());
        }

        private static string removeTrailingSlashAndEmptyQuery(string url)
        {
            return url
                    .TrimEnd(new[] { '?' })
                    .TrimEnd(new[] { '/' });
        }

        private static string removeDefaultDirectoryIndexes(string url)
        {
            foreach (var index in DefaultDirectoryIndexes)
            {
                if (url.EndsWith(index))
                {
                    url = url.TrimEnd(index.ToCharArray());
                    break;
                }
            }
            return url;
        }
    }
}
```

از این روش برای تمیز کردن و حذف فیدهای تکراری در فایل‌های OPML تهیه شده نیز می‌شود استفاده کرد. عموما فیدخوان‌های نه‌چندان با سابقه، نکات یاد شده در این مطلب را رعایت نمی‌کنند و به سادگی می‌شود در این سیستم‌ها، فیدهای تکراری زیادی را ثبت کرد.

برای مثال اگر یک فایل OPML چنین ساختار XML ایی را داشته باشد:

```
<?xml version="1.0" encoding="utf-8"?>
<opml xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema" version="1.0">
  <body>
    <outline text="آی تی ایرانی">
    <outline type="rss"
                          text="فید کلی آخرین نظرات، مطالب، اشتراک‌ها و پروژه‌های .NET Tips"
                          title="فید کلی آخرین نظرات، مطالب، اشتراک‌ها و پروژه‌های .NET Tips"
                          xmlUrl="http://www.dotnettips.info/Feed/LatestChanges"
                          htmlUrl="http://www.dotnettips.info/" />
    </outline>
  </body>
</opml>
```

هر outline آن‌را به کلاس زیر می‌توان نگاشت کرد:

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

```
namespace OPMLCleaner
{
    [XmlType(TypeName="outline")]
    public class Opml
    {
        [XmlAttribute(AttributeName="text")]
        public string Text { get; set; }

        [XmlAttribute(AttributeName = "title")]
        public string Title { get; set; }

        [XmlAttribute(AttributeName = "type")]
        public string Type { get; set; }

        [XmlAttribute(AttributeName = "xmlUrl")]
        public string XmlUrl { get; set; }

        [XmlAttribute(AttributeName = "htmlUrl")]
        public string HtmlUrl { get; set; }
    }
}
```

برای اینکار فقط کافی است از LINQ to XML به نحو ذیل استفاده کنیم:

```
var document = XDocument.Load("it-92-03-01.opml");
var results = (from node in document.Descendants("outline")
                            where node.Attribute("htmlUrl") != null && node.Parent.Attribute("text") !=
null
                             && node.Parent.Attribute("text").Value == "آی تی ایرانی"
                            select new Opml
                            {
                                HtmlUrl = (string)node.Attribute("htmlUrl"),
                                Text = (string)node.Attribute("text"),
                                Title = (string)node.Attribute("title"),
                                Type = (string)node.Attribute("type"),
                                XmlUrl = (string)node.Attribute("xmlUrl")
                            }).ToList();
```

در این حالت لیست کلیه فیدهای یک گروه را چه تکراری و غیرتکراری، دریافت خواهیم کرد. برای حذف موارد تکراری نیاز است از متد Distinct استفاده شود. به همین جهت باید کلاس ذیل را نیز تدارک دید:

```
using System.Collections.Generic;

namespace OPMLCleaner
{
    public class OpmlCompare : EqualityComparer<Opml>
    {
        public override bool Equals(Opml x, Opml y)
        {
            return UrlNormalization.AreTheSameUrls(x.HtmlUrl, y.HtmlUrl);
        }

        public override int GetHashCode(Opml obj)
        {
            return obj.HtmlUrl.GetHashCode();
        }
    }
}
```

اکنون با کمک کلاس OpmlCompare فوق که از کلاس UrlNormalization برای تشخیص لینک‌های تکراری استفاده می‌کند، می‌توان به لیست بهتر و متعادل‌تری رسید:

```
var distinctResults = results.Distinct(new OpmlCompare()).ToList();
```