

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 منحصر بفرد بیشتر ندارد. ما هس این لینک را محاسبه می‌کنیم و بعد روی این هس، یک کلید منحصر بفرد را تعریف خواهیم کرد تا دیگر رکوردی تکراری ثبت نشود. همچنین چون این هس 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/index.htm> و <http://www.site.com/index.htm#section1> هسند که فقط اصطلاحا در یک fragment با هم تفاوت دارند و از این دست لینک‌هایی که باید در حین ثبت یکی در نظر گرفته شوند، زیاد هسند 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."))
            {
                url = "http://www." + url;
            }
        }
    }
}
```

```

        {
            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.IsNullOrEmpty(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. عموماً 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=".NET Tips. فید کلی آخرین نظرات، مطالب، اشتراک‌ها و پروژه‌های"
                title=".NET Tips. فید کلی آخرین نظرات، مطالب، اشتراک‌ها و پروژه‌های"
                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);
        }

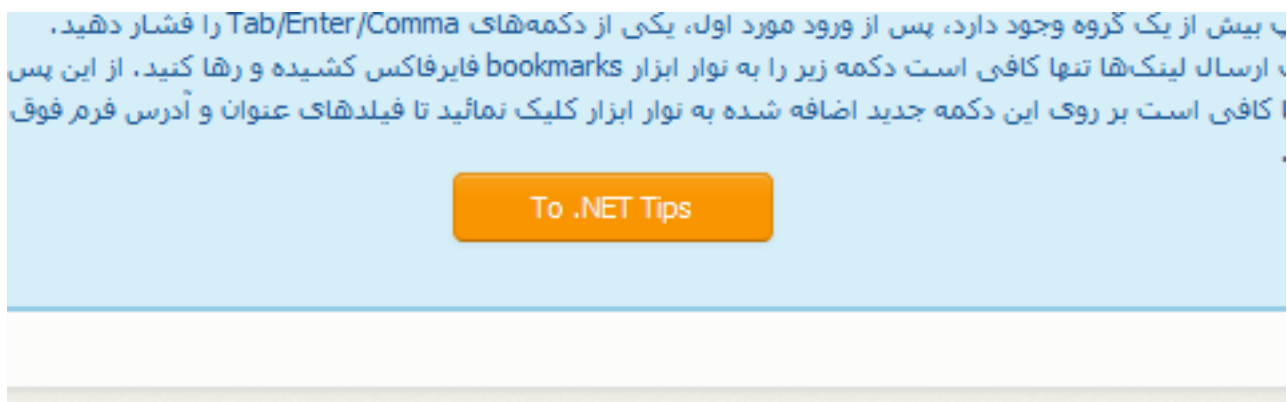
        public override int GetHashCode(Opml obj)
        {
            return obj.HtmlUrl.GetHashCode();
        }
    }
}
```

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();
```

اگر بعنوان فردی که روزانه بیش از 300 عنوان خبری مربوط به آی تی را مانند دیگر توسعه‌دهندگان، قسمتی از فعالیت روزانه‌ی خود کرده‌اید، مطمئناً دنبال راه‌های ساده‌ی اشتراک گذاری و یا به قول آقای هنسلمن [Tif this thenhat](#) یا [IFTT باختصار](#) هستید. من برای مرور و دسته بندی فیدها از [Inoreader.com](#) ([دارای یک API برای توسعه](#)) استفاده می‌کنم و برای [اشتراک مطالب در این سایت](#) از امکان موجود در هر دو سایت، استفاده می‌کنم.

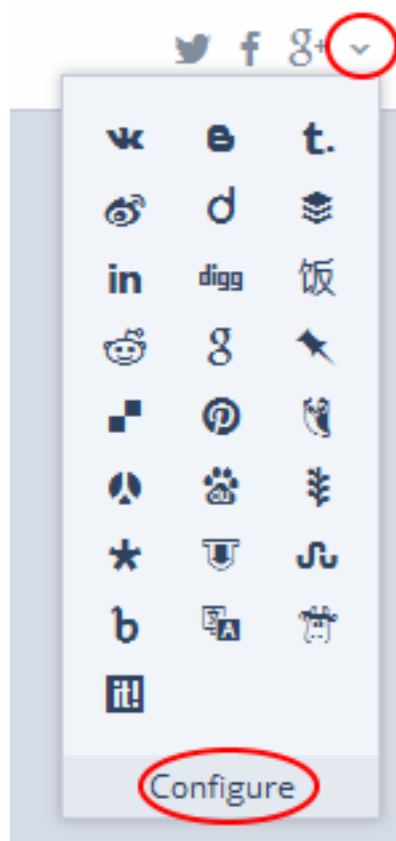
در سایت جاری و [در قسمت ارسال لینک‌های خبری](#) ، امکانی فراهم شده‌است تا کسانی که از فایرفاکس استفاده می‌کنند بتوانند به راحتی صفحه‌ی در حال تماشا را برای اشتراک ارسال کنند. تصویر زیر قسمتی از صفحه‌ی ارسال لینک‌های خبری است که دکمه‌ی اشتراک فایرفاکس در آن وجود دارد:



که لینک آن دارای یک دستور جاوااسکریپتی است که کار ارسال لینک و عنوان encode شده‌ی مطلب را انجام می‌دهد:

```
javascript:location.href='http://www.dotnettips.info/dailylinks?url='+encodeURIComponent(location.href)+'&title='+encodeURIComponent(document.title)
```

در Inoreader نیز میتوان برای اشتراک آیتم‌ها، لینکی سفارشی را ایجاد و استفاده کرد تا کار اشتراک گذاری به راحتی در تمام مرورگرهای متداول و نه فقط فایرفاکس، انجام پذیرد. به این صورت که در گوشه‌ی سمت راست و پایین آیتم باز شده، تعداد زیادی از شبکه‌های اجتماعی برای اشتراک وجود دارند و نیز قسمتی هم برای معرفی یک Custom Site:



که برای تعریف یک مقصد اشتراک، پارامترهای زیر را در اختیار می‌گذارد:

[TITLE]: عنوان آیتم با انکدینگ

[TITLE_NOENC]: عنوان آیتم بدون انکدینگ

[URL]: آدرس آیتم

[CONTENT]: محتوای آیتم

[SOURCE]: منبع، که در اینجا کلمه‌ی Inoreader جایگزین میشود.

حالا با استفاده از الگوی معرفی شده‌ی لینک اشتراک سایت DNT و جایگزینی با آیتمهای بالا میتوان یک Custom Site را به Inoreader اضافه کرد:

```
http://www.dotnettips.info/dailylinks?url=[URL]&title=[TITLE]
```

که البته همانطور که مشخص است سایت جاری فقط دو پارامتر از پنج پارامتر را قبول می‌کند (پارامترهای دارای encoding):

Use this dialog to reorder your sharing buttons. The first 3 buttons will always be visible after you click the Share button. The rest will be available in the quick menu.

☐ Hide sharing options behind "Send to" button

☐ Shorten URLs before sharing

Add custom site

Name:



URL:

More settings

URL Example:

`http://you.wordpress.com/wp-admin/press-this.php?u=[URL]&t=[TITLE]&s=[CONTENT]`
To send an article to your Wordpress blog

URL Variables: [TITLE], [TITLE_NOENC], [URL], [CONTENT], [SOURCE]

 Twitter 

حالا این مورد نیز به لیست اشتراک گذاری در Inoreader اضافه شده و می‌توانید نتیجه‌ی تست آن را در زیر ببینید:






























www.dotnettips.info/dailylinks?url=http%3A%2F%2Fwindowscenter.ir%2F%251

لینک خبری

آدرس مطلب

عنوان

توضیح کوتاه حداکثر ۳ سطری



۶/۶