عنوان: **حذف تگ‌های زاید دریافتی از متون MS-Word**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۸/۱۱ ۱۷:۳۷:۲۰
آدرس: www.dotnettips.info
برچسب‌ها: Regular expressions

یکی از مشکلاتی که من همیشه با کاربران عادی دارم بحث انتقال مطالب از Word مایکروسافت به ادیتورهای WYSWING تحت وب است. برای مثال شما سایت پویایی را درست کرده‌اید که کاربران می‌توانند مطالب آنرا ویرایش یا کم و زیاد کنند.
اگر مطلب از ابتدا در این نوع ادیتورها تایپ و آماده شود هیچ مشکلی وجود نخواهد داشت چون خروجی اکثر آنها استاندارد است، اما متاسفانه خروجی وب word بسیار مشکل‌زا است (copy/paste معمولی مطالب آن در یک ادیتور تحت وب) و خصوصا برای نمایش تایپ فارسی در وب اصلا مناسب نیست. یعنی هیچ الزامی وجود ندارد که اندازه فونت‌ها در متن نهایی نمایش داده شده در وب یکسان باشند یا خطوط در هم فرو نروند و یا عدم تناسب اندازه قلم متن صفحه با قلم استفاده شده در CSS سایت (که شکل ناهماهنگ و غیرحرفه‌ای را حاصل خواهد کرد) و امثال آن. اینجاست که کار شما زیر سؤال می‌رود! "این برنامه درست کار نمیکنه! متن من به‌هم ریخته شده و امثال این"
این کاربر عادی عموما یک تایپیست است یا یک منشی که به او گفته شده است شما از امروز موظفید مطالبی را در این سایت قرار دهید. بنابراین این کاربر حتما از word استفاده خواهد کرد (برای پیش نویس مطالب). همچنین عموما هم مرورگر "سازمانی" مورد استفاده، هنوز که هنوز است همان IE6 است (در اکثر شرکت‌ها و خصوصا ادارات) و مهم نیست که الان آخرین نگارش IE یا فایرفاکس و تمام هیاهوهای مربوطه به کجا ختم شده‌اند. حتما باید سایت با IE6 هم سازگار باشد. بنابراین از برنامه IE tester غافل نشوید.
و دست آخر شما هم نمی‌توانید به کاربر عادی ثابت کنید که این خروجی وب word اصلا استاندارد نیست (حتما کار شما است که مشکل دارد نه شرکت معظم مایکروسافت!). یا اینکه به آنها بگوئید اصلا مجاز نیستید در وب همانند یک فایل word از چندین نوع قلم مختلف فارسی غیراستاندارد استفاده کنید چون ممکن است کاربری این نوع قلم مورد استفاده شما را نداشته باشد و نمایش نهایی به هم ریخته‌تر از آنی خواهد بود که شما فکرش را می‌کنید! یا اینکه با استفاده از این روش حجم نهایی صفحه حداقل 50 کیلو بایت بیشتر خواهد شد (بدلیل حجم بالای تگ‌های زاید word) و نباید کاربران دایال آپ را فراموش کرد.
مدتی در اینباره جستجو کردم و نتیجه حاصل این بود که تمامی روش‌ها به یک مورد ختم می‌شود: حذف تگ‌های غیراستاندارد word هنگام دریافت مطلب و پیش از ذخیره سازی آن در دیتابیس
یک سری از ادیتورهای متنی تحت وب مانند FCK editor این قابلیت را به صورت خودکار اضافه کرده‌اند و حتی اگر کاربر متنی را از word در آنها Paste کند پیغامی را در همین رابطه دریافت خواهد کرد (شکل زیر) و البته کاربر می‌تواند گزینه لغو یا خیر را نیز انتخاب کند و دوباره همان وضعیت قبل تکرار خواهد شد. (یا حتی دکمه مخصوص کپی از word را هم به نوار ابزار خود اضافه کرده‌اند)

برای این منظور تابع زیر تهیه شده‌است که من همواره از آن استفاده می‌کنم و تا به امروز مشکل پاسخ پس دادن به کاربران عادی را به این صورت حل کرده‌ام!
این تابع تمامی تگ‌های اضافی و غیراستاندارد word متن دریافتی از یک ادیتور WYSWING را حذف می‌کند و به این صورت متن نهایی نمایش داده شده در سایت، تابع CSS مورد استفاده در سایت خواهد شد و نه حجم بالایی از تگ‌های غیراستاندارد word. (ممکن است کاربر در ابتدا کمی جا بخورد ولی مهم نیست! سایت باید استاندارد نمایشی خودش را از CSS آن دریافت کند و نه از تگ‌های word)

```
using System.Text.RegularExpressions;
 /// <summary>
 /// Removes all FONT and SPAN tags, and all Class and Style attributes.
 /// Designed to get rid of non-standard Microsoft Word HTML tags.
 /// </summary>
 public static string CleanMSWordHtml(string html)
 {
     try
     {
         // start by completely removing all unwanted tags
```

```
        html = Regex.Replace(html, @"<[/]?(font|span|xml|del|ins|[ovwxp]:\w )[^>]*?>", "",
RegexOptions.IgnoreCase);
        // then run another pass over the html (twice), removing unwanted attributes
        html = Regex.Replace(html, @"<([^>]*)(?:class|lang|style|size|face|[ovwxp]:\w
)=(?:'[^']*'|""[^""]*""|[^\s>] )([^>]*)>", "<$1$2>", RegexOptions.IgnoreCase);
        html = Regex.Replace(html, @"<([^>]*)(?:class|lang|style|size|face|[ovwxp]:\w
)=(?:'[^']*'|""[^""]*""|[^\s>] )([^>]*)>", "<$1$2>", RegexOptions.IgnoreCase);
        return RemoveHTMLComments(html);
    }
    catch
    {
        return html;
    }
}

public static string RemoveHTMLComments(string html)
{
    try
    {
        Regex _Regex = new Regex("((<!-- )((?!<!-- ).)*( -->))(\r\n)*", RegexOptions.Singleline);
        return _Regex.Replace(html, string.Empty);
    }
    catch
    {
        return html;
    }
}
```

متد RemoveHTMLComments را عمدا جدا قرار دادم تا مشخص‌تر باشد. پس از تمیزکاری اولیه، ممکن است دسته‌گل‌های تیم مایکروسافت به صورت کامنت باقی بمانند که باید آنها را هم تمیز کرد! :)

نظرات خوانندگان

نویسنده: Shaho
تاریخ: ۰۲:۲۶:۰۰ ۱۳۸۷/۰۸/۱۷

اقا خیلی مرسی!

نویسنده: سیدمحمدرضا فخری
تاریخ: ۱۱:۲۰:۲۰ ۱۳۸۸/۰۲/۳۱

سلام. خیلی ممنون از این کد بسیار مفید. اما یک مطلب و آن اینکه عبارت class=MsoNormal با این تابع حذف نمیشه. ممنون میشم بفرمائید چه تغییری در کد بدهیم.

نویسنده: وحید نصیری
تاریخ: ۱۴:۲۴:۱۹ ۱۳۸۸/۰۲/۳۱

class ها و lang|style|size|face|[ovwxp باید حذف بشه. ولی اگر روش فوق راضی کننده نبود از روش مقاله زیر هم می‌توان استفاده کرد:

http://www.codinghorror.com/blog/archives/000485.html

یک روش دیگر هم این است که کلا هرچی تگ html است را یکجا حذف کرد. روش کار به صورت زیر است:

http://gibbons.co.za/archive/2005/01/28/249.aspx

نویسنده: سیدمحمدرضا فخری
تاریخ: ۱۱:۴۵:۰۹ ۱۳۸۸/۰۳/۰۲

سلام. با تشکر از شما، برنامه مربوط به کدینگ هارور رو قبلا تست کردم، مشکل داشت. کدی شکه شما زحمت کشیدید هم غیر از مسئله ذکر شده درکامنت اول، خوب کار میکرد فقط مشکل اینست که اختصاصی به فرمت های ورد ندارد و همه استایل ها را پاک میکند که برای ما بعنوان یک بلاگ سرویس قابل استفاده نیست، چون کاربران پس از کپی پیست، استایل های خودشان را هم اضافه میکنند و این کد همه را پاک میکند. اگر برنامه بود که فقط بتواند اضافات ورد را پاک کند خوب بود. و اینکه بصورت جاوا اسکریپت باشد تا بتوان درکلاینت هم از آن استفاده کرد(من متاسفانه رگولار اکسپرشن را عمیق کار نکرده ام، همین رگولاررا میتوان در جاوا هم بکار برد؟).

نویسنده: وحید نصیری
تاریخ: ۱۲:۴۶:۳۷ ۱۳۸۸/۰۳/۰۲

سلام،
بله در حالت جاوا اسکریپتی توسط FCK-Editor هم کار شده که می‌شود از آن ایده گرفت:
http://dev.fckeditor.net/browser/FCKeditor/trunk/editor/dialog/fck_paste.html

به تابع CleanWord آن در صفحه فوق مراجعه نمائید.

نویسنده: سیدمحمدرضا فخری
تاریخ: ۱۴:۱۹:۴۸ ۱۳۸۸/۰۳/۰۲

ممنون

عنوان: **استخراج آدرس‌های ایمیل از یک متن**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۸/۲۷ ۲۰:۵۰:۴۸
آدرس: www.dotnettips.info
برچسب‌ها: Regular expressions

در قسمت اول بررسی نحوه برنامه نویسی افزونه outlook ، در مورد استفاده از regular expressions اندکی توضیح داده شد. امروز مثالی دیگر از همین دست را بررسی خواهیم کرد.

چند روز قبل یک ایمیل تبلیغاتی به دست من رسید که فرد ارسال کننده انبوهی از ایمیل‌ها را در قسمت To قرار داده بود (بجای قسمت BCC (رونوشت مخفی)).
خوب، برای جدا کردن انبوهی از ایمیل‌های مخلوط با سایر متون چه باید کرد؟ چند ساعت وقت گذاشت و تک تک آنها را به صورت دستی جدا کرد؟ (برای ذخیره سازی در یک دیتابیس برای مثال :))
یا برای مثال برنامه‌های download manager توانایی استخراج لینک‌های موجود در یک متن کپی شده در حافظه را دارند. آنها به چه صورتی عمل می‌کنند؟ چگونه می‌توانند لینک‌ها را با دقتی بالا و بسیار سریع از لابلای متن موجود تشخیص دهند؟

بهینه‌ترین و سریعترین راه برای این نوع جستجوها استفاده از کتابخانه regular expressions (عبارات با قاعده) در دات نت فریم ورک است. اگر نیاز به یک برگه تقلب (!) در این زمینه داشتید می‌توانید به اینجا مراجعه کنید. همچنین در همان سایت، کاربران بسیاری را خواهید یافت که الگوهای ابداعی خود را با دیگران به اشتراک می‌گذارند.

برای مثال فرض کنید فایلی را که حاوی مخلوطی از متن و ایمیل است را در یک رشته بارگذاری کرده‌اید. نحوه استخراج ایمیل‌های موجود با استفاده از این امکانات به صورت زیر خواهد بود:

```
using System.IO;
using System.Text.RegularExpressions;
using System.Text;

class CRegEx
{
   /// <summary>
   /// استخراج ایمیل‌های یک فایل متنی و ذخیره آن در فایلی جدید
   /// </summary>
   /// <param name="inFilePath">فایل ورودی</param>
   /// <param name="outFilePath">فایل خروجی</param>
   public static void ExtractEmails(string inFilePath, string outFilePath)
   {
       string data = File.ReadAllText(inFilePath); //خواندن فایل متنی
       //ایجاد شیء عبارت با قاعده بر اساس الگوی تشخیص ایمیل‌ها
       Regex emailRegex = new Regex(@"\w+([-+.]\w+)*@\w+([-.]\w+)*\.\w+([-.]\w+)*",
           RegexOptions.IgnoreCase);
       //پیدا کردن گروه تطابق یافته با الگوی ما
       MatchCollection emailMatches = emailRegex.Matches(data);
       //ایجاد شیء استرینگ بیلدر برای ذخیره سازی سریع اطلاعات دریافتی
       StringBuilder sb = new StringBuilder();
       //ذخیره ایمیل‌های استخراج شده
       foreach (Match emailMatch in emailMatches)
       {
           sb.AppendLine(emailMatch.Value);
       }
       //ذخیره کردن اطلاعات استخراج شده در فایلی جدید
       File.WriteAllText(outFilePath, sb.ToString());
   }
}
```

راستی، اگر روزی خواستید تعداد بالایی ایمیل ارسال کنید، آنها را به قسمت bcc اضافه کنید (Message.Bcc.Add)، در قالب یک ایمیل، نه چند هزار ایمیل متوالی (در طی یک حلقه برای مثال). به این صورت (استفاده از قسمت BCC) میل سرور تمام آدرس‌ها را در صف قرار خواهد داد و متحمل بار اضافی شدید نخواهد شد. در این حالت اگر میل باکس خود را چک کنید شاید بلافاصله ایمیل

مورد نظر را دریافت نکنید. نگران نباشید، انجام عملیات در صف قرار گرفته و در طی دقایق و یا حتی ساعات بعدی پردازش خواهد شد (بسته به بار سرور).

چند نکته را باید در اینجا در نظر داشت. حتما آدرس‌های اضافه شده را با استفاده از عبارات باقاعده یکبار پیش از اضافه شدن بررسی نمائید (Regex.IsMatch). در صورتیکه یکی از ایمیل‌ها فرمت غیراستانداردی داشته باشد کل کار برگشت خواهد خورد.

و همچنین باید دقت داشت که برای این موضوع حد نصاب وجود دارد. بر روی یکی از میل سرورهای یک هاست ایرانی تست کردم، حداکثر 100 رونوشت مخفی را بیشتر قبول نمی‌کرد. بنابراین هر بار می‌شود 100 ایمیل را به صورت یکجا ارسال کرد (که باز هم از روش استفاده از حلقه‌ای که 100 بار ایمیل می‌زند بسیار بهتر است و هاست دار به علت ایجاد بار اضافی شدید بر روی سرور با شما تماس نخواهد گرفت)

عنوان: **چگونه Regex سریعتری داشته باشیم؟**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۱/۰۱ ۱۳:۳۶:۰۱
آدرس: www.dotnettips.info
برچسب‌ها: Regular expressions

نکاتی را در هنگام کار با عبارات با قاعده در دات نت باید رعایت نمود تا بتوان به حداکثر کارآیی و سرعت دست یافت:
1- ایجاد اشیاء Regex هزینه بر هستند. برای مثال اگر متد شما که در آن شیء Regex را ایجاد کرده‌اید مرتبا فراخوانی می‌شود، این شیء را به صورت یک متغیر محلی خارج از بدنه تابع تعریف کنید. یا به همین صورت هرگز در یک حلقه اشیاء Regex را بارها و بارها ایجاد نکنید.
2- از گزینه RegexOptions.Compiled استفاده کنید. با اینکار زمانیکه برنامه شما اجرا می‌شود، عبارت باقاعده در حافظه کامپایل شده و به بهبود کارآیی 30 درصدی دست خواهید یافت. اگر از این گزینه استفاده نشود، هر بار که شیء Regex مورد استفاده قرار می‌گیرد، عبارت باقاعده شما همانند یک اسکریپت باید مجددا تفسیر شود.
3- اشیاء Regex را از نوع static readonly تعریف کنید تا بازهم کارآیی را افزایش دهید (اشیایی ثابت در زمان اجرا و همچنین اشاره‌گری هستند به آن شیء و نه مقدار آن).

خلاصه موارد فوق:

```
private static readonly Regex _valueFormatMatch = new Regex(@"[0-9]", RegexOptions.Compiled);
```

بعلاوه اگر نمی‌خواهید Regex شما هر بار در حین اجرای برنامه (در اولین باری که برنامه بارگذاری می‌شود)، کامپایل شود، می‌توانید آنرا به درون یک اسمبلی نیز کامپایل کنید (Precompilation). روش انجام اینکار را در این مقاله می‌توانید مشاهده نمائید.

عنوان: **عبارات باقاعده‌ای در مورد کار با تگ‌ها**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۵/۰۱ ۱۴:۵۳:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Regular expressions

حذف تمامی تگ‌های یک عبارت HTML

این تابع و عبارت باقاعده به کار رفته در آن هنگام جستجو بر روی یک فایل html که حاوی انبوهی از تگ‌ها است می‌تواند مفید باشد و یا جهت حذف هر نوع فرمت اعمالی به یک متن.

```
private static readonly Regex _htmlRegex = new Regex("<.*?>", RegexOptions.Compiled);
/// <summary>
/// حذف تمامی تگ‌های موجود
/// </summary>
/// <param name="html">ورودی اچ تی ام ال</param>
/// <returns></returns>
public static string CleanTags(string html)
{
      return _htmlRegex.Replace(html, string.Empty);
}
```

حذف یک تگ ویژه بدون حذف محتویات آن

فرض کنید می‌خواهید تمام تگ‌های script بکار رفته در یک محتوای html را حذف کنید.

```
private static readonly Regex _contentRegex = new Regex(@"<\/?script[^>]*?>", RegexOptions.Compiled |
RegexOptions.IgnoreCase);

/// <summary>
/// تنها حذف یک تگ ویژه
/// </summary>
/// <param name="html">ورودی اچ تی ام ال</param>
/// <returns></returns>
public static string CleanScriptTags(string html)
{
     return _contentRegex.Replace(html, string.Empty);
}
```

حذف یک تگ خاص به همراه محتویات آن تگ

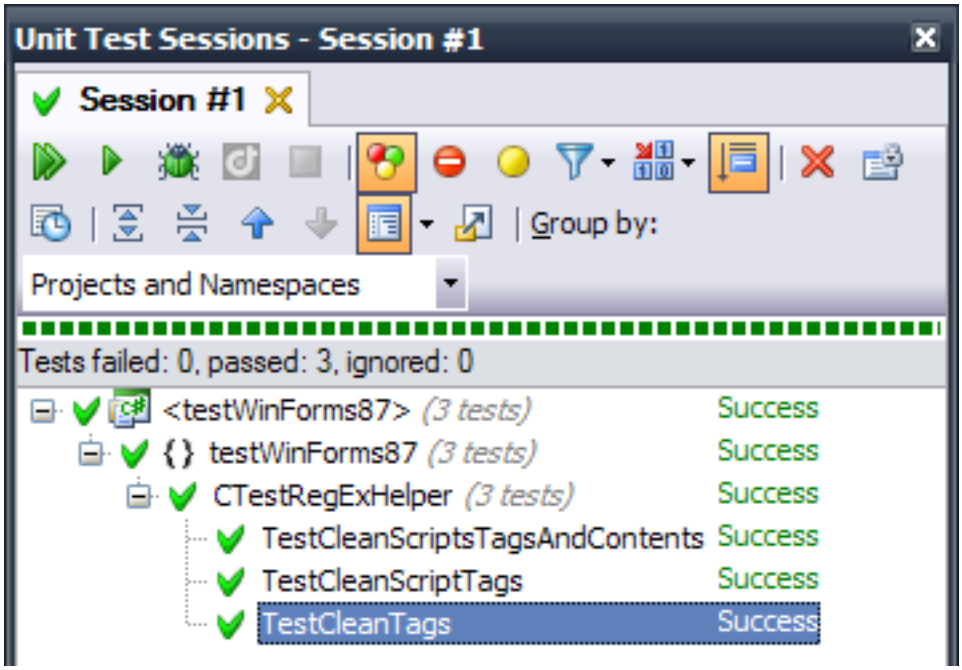
فرض کنید می‌خواهیم در محتوای html دریافتی اثری از تگ‌ها و کدهای جاوا اسکریپتی یافت نشود.

```
private static readonly Regex _safeStrRegex = new Regex(@"<script[^>]*?>[\s\S]*?<\/script>",
           RegexOptions.Compiled | RegexOptions.IgnoreCase);

/// <summary>
/// حذف یک تگ ویژه به همراه محتویات آن
/// </summary>
/// <param name="html">ورودی اچ تی ام ال</param>
/// <returns></returns>
public static string CleanScriptsTagsAndContents(string html)
{
      return _safeStrRegex.Replace(html, "");
}
```

و اگر فرض کنیم که متدهای فوق در کلاسی به نام CRegExHelper قرار گرفته‌اند، کلاس آزمون واحد آن به صورت زیر می‌تواند باشد:

```
using NUnit.Framework;
```

```
namespace testWinForms87
{
   [TestFixture]
   public class CTestRegExHelper
   {
  #region Methods (3)

  // Public Methods (3)

        [Test]
        public void TestCleanScriptsTagsAndContents()
        {
            Assert.AreEqual(
                CRegExHelper.CleanScriptsTagsAndContents("data1 <script> ... </script> data2"),
                "data1  data2");
        }

        [Test]
        public void TestCleanScriptTags()
        {
            Assert.AreEqual(
                CRegExHelper.CleanScriptTags("<b>data1</b> <script> ... </script> data2"),
                "<b>data1</b>  ...  data2");
        }

        [Test]
        public void TestCleanTags()
        {
            Assert.AreEqual(
                CRegExHelper.CleanTags("<b>data</b>"),
                "data");
        }

  #endregion Methods
   }

}
```



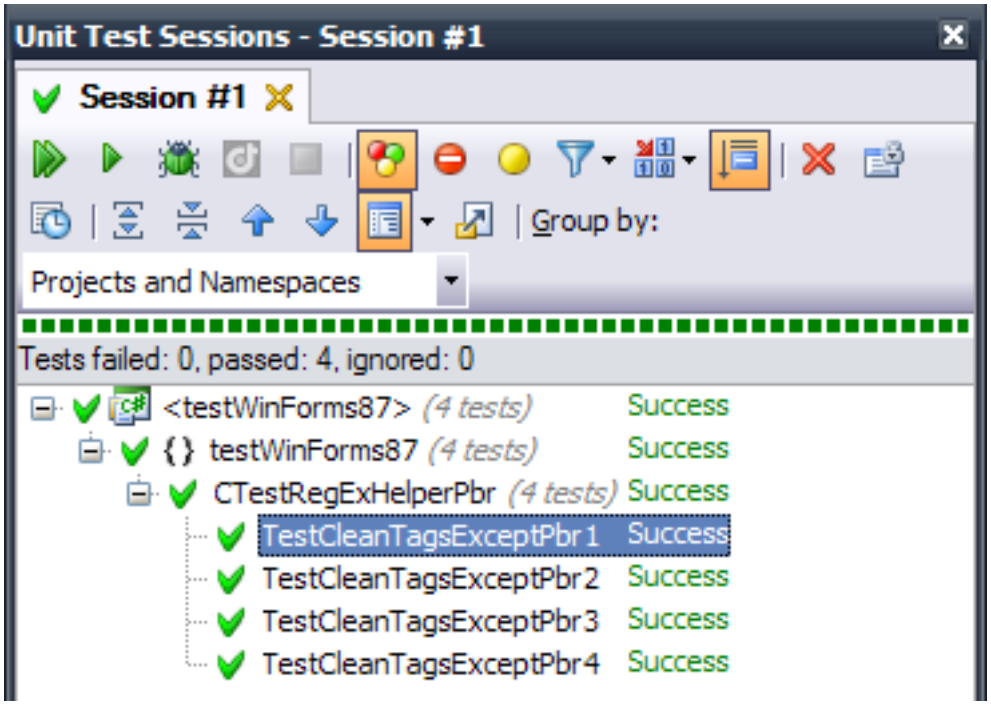
عنوان: **حذف تمامی تگ‌ها منهای چند تگ خاص از HTML دریافتی**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۶/۰۵ ۲۱:۱۲:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: Regular expressions

در ادامه مطلب " عبارات باقاعده‌ای در مورد کار با تگ‌ها " ، عبارت باقاعده مربوطه به حذف تمامی تگ‌ها برای فرمت زدایی یک متن بسیار جالب است اما مشکلی را که به وجود خواهد آورد، از بین بردن سطرهای موجود است. به عبارت دیگر با استفاده از این عبارت با قاعده، کل متن در امتداد یک سطر قرار می‌گیرد. اکنون می‌خواهیم تمامی تگ‌ها منهای دو تگ مربوط به p و br حذف شوند. چه باید کرد؟

```
private static readonly Regex _pbrRegex = new Regex(@"<(?!br|/br|p|/p).+?>",
          RegexOptions.Compiled | RegexOptions.IgnoreCase);
/// <summary>
/// حذف تمامی تگ‌ها منهای دو تگ ذکر شده
/// </summary>
/// <param name="html"></param>
/// <returns></returns>
public static string CleanTagsExceptPbr(string html)
{
    return _pbrRegex.Replace(html, string.Empty);
}
```

و اگر بخواهیم یک سری تست برای آن بنویسیم به موارد زیر می‌توان اشاره کرد:

```
using NUnit.Framework;

namespace testWinForms87
{
   [TestFixture]
   public class CTestRegExHelperPbr
   {
       [Test]
       public void TestCleanTagsExceptPbr1()
       {
           Assert.AreEqual(
               CRegExHelper.CleanTagsExceptPbr("<b>data1</b><br/>data2"),
               "data1<br/>data2");
       }

       [Test]
       public void TestCleanTagsExceptPbr2()
       {
           Assert.AreEqual(
               CRegExHelper.CleanTagsExceptPbr("<b>data1</b><br>data2"),
               "data1<br>data2");
       }

       [Test]
       public void TestCleanTagsExceptPbr3()
       {
           Assert.AreEqual(
               CRegExHelper.CleanTagsExceptPbr("<p><b>data1</b><br/>data2</p>"),
               "<p>data1<br/>data2</p>");
       }

       [Test]
       public void TestCleanTagsExceptPbr4()
       {
           Assert.AreEqual(
               CRegExHelper.CleanTagsExceptPbr("<b>data1</b><p>data2<br />"),
               "data1<p>data2<br />");
       }
   }
}
```


Unit Test Sessions - Session #1
Session #1
Group by:
Projects and Namespaces
Tests failed: 0, passed: 4, ignored: 0
<testWinForms87> (4 tests) Success
testWinForms87 (4 tests) Success
CTestRegExHelperPbr (4 tests) Success
TestCleanTagsExceptPbr1 Success
TestCleanTagsExceptPbr2 Success
TestCleanTagsExceptPbr3 Success
TestCleanTagsExceptPbr4 Success

نظرات خوانندگان

نویسنده: mdpdotnet
تاریخ: ۱۳۸۸/۰۶/۰۵ ۲۳:۲۷:۴۰

در مورد Regex ای که نوشتید یه توضیحی میدید ؟

من زیاد با رجکس ها جور نیستم.تازه دارم یاد میگیرم.

:)

نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۶/۰۶ ۰۰:۲۰:۳۵

سلام
یک debug visualizer برای VS.Net هست به نام Regular Expression Visualizer. با VS2005 و 2008 سازگار است.
آنرا از آدرس زیر دریافت کنید:
http://weblogs.asp.net/rosherove/archive/2005/11/26/AnnoucingRegexKit10.aspx
سپس فایل‌های dll آن‌را در یکی از مسیرهای زیر بسته به نگارش VS.Net خودتون کپی کنید:
My Documents\Visual Studio 2008\Visualizers
یا
My Documents\Visual Studio 2005\Visualizers

اکنون در VS.Net روی سطر return _pbrRegex یک breakpoint بگذارید و نتیجه را ملاحظه کنید.
یک مثال عملی:
http://professionalaspnet.com/archive/2008/06/18/Regular-Expression-Visualizer.aspx

برای نمونه خروجی عبارت باقاعده مثال جاری به صورت زیر است که کمک شایانی است در درک عبارت فوق و امثال آن:

>
zero-width negative lookahead
br
or
br/
or
p
or
p/
End Capture
(any character) .
+ (one or more times) (non-greedy)
<

نویسنده: DotNetCoders
تاریخ: ۱۳۸۸/۰۶/۰۶ ۰۱:۳۶:۴۹

ممنون جناب نصیری فقط من با این بخش آخر رجکس مشکل دارم.

این بخش منظورمه :

<?+.

معنیش چیه؟

اگه کتاب کامل و مفید یا منبع خوبی بهم معرفی کنید خودم پیدا میکنم وقت شما رو هم نمیگیرم.

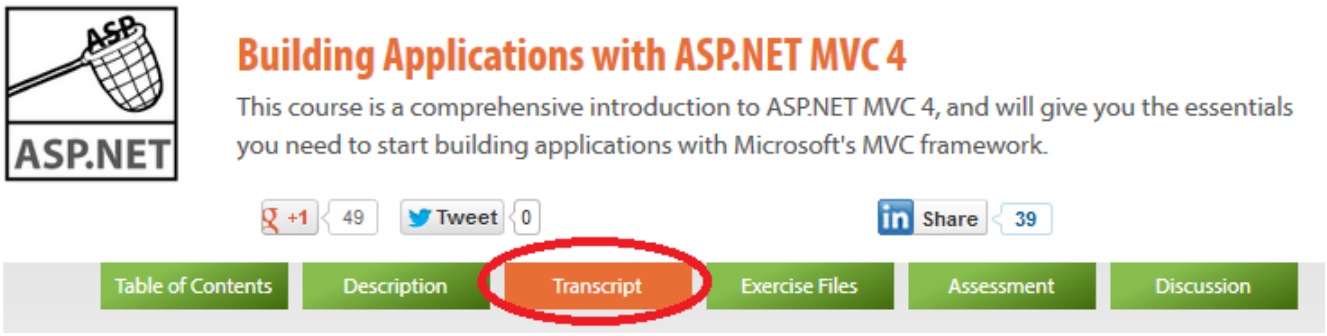
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۶/۰۶ ۰۲:۴۳:۴۰

- لطفا سه چهار سطر آخر پاسخ ارسال شده بنده را مرور بفرمائید.
- کتاب خوب هم، همه این‌را توصیه می‌کنند:

Mastering Regular Expressions از O'Reilly Media که تا به حال سه edition از آن منتشر شده.

عنوان: **تبدیل زیرنویس‌های خاص پلورال‌سایت به فرمت SRT**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۹ ۱۸:۳۵
آدرس: www.dotnettips.info
گروه‌ها: Regular expressions, SubtitleTools, Html Agility Pack

یک سری از دوره‌های پلورال‌سایت دارای زیرنویس هستند که تحت عنوان Transcript در کنار آن‌ها قرار گرفته‌اند:



این زیرنویس‌ها فرمت ویژه‌ای دارند:

```
<li class="transcript-module">
                    Introduction to ASP.NET MVC 4
                    <ul>
                                <li class="transcript-clip" data-p="author=scott-allen&amp;name=mvc4-
building-m1-intro&amp;mode=live&amp;clip=0&amp;course=mvc4-building"><a href="javascript:void(0)"
onclick="LaunchPlayerWindow('http://pluralsight.com/training', 'author=scott-allen&amp;name=mvc4-
building-m1-intro&amp;mode=live&amp;clip=0&amp;course=mvc4-building');">Introduction</a><br />
                                    <div>
                                                <a href="javascript:void(0)" onclick="p(this);" data-
s="1.636">Hi, this is Scott Allen and this is the first module in the course design</a>
                                    </div>
                                </li>
                                <li class="transcript-clip" data-p="author=scott-allen&amp;name=mvc4-
building-m1-intro&amp;mode=live&amp;clip=1&amp;course=mvc4-building"><a href="javascript:void(0)"
onclick="LaunchPlayerWindow('http://pluralsight.com/training', 'author=scott-allen&amp;name=mvc4-
building-m1-intro&amp;mode=live&amp;clip=1&amp;course=mvc4-building');">Web Platform Installer</a><br
/>
                                    <div>
                                    ...
```

در آن، هر li که دارای کلاسی به نام transcript-clip است، حاوی یک div می‌باشد و این div دارای تعدادی لینک است. این لینک‌ها توسط ویژگی datas آن‌ها که بیانگر زمان شروع گفتگو است، مشخص می‌شوند و همینطور الی آخر. بنابراین اگر بخواهیم برای آن‌ها ساختاری را تهیه کنیم، به کلاس‌های ذیل خواهیم رسید:

```
public class TranscriptClip
    {
        public string Title { set; get; }
        public IList<TranscriptItem> TranscriptItems { set; get; }
    }

    public class TranscriptItem
    {
        public double StartTime { set; get; }
        public string Text { set; get; }
    }
```

هر li دارای کلاس transcript-clip، یک شیء TranscriptClip را تشکیل می‌دهد. هر شیء TranscriptClip می‌تواند داری چندین TranscriptItem باشد.

برای استخراج این اطلاعات، یکی از بهترین ابزارها، کتابخانه HTML Agility pack است که توسط آن می‌توان به liهای یاد شده دسترسی یافت:

```
var nodes = doc.DocumentNode.SelectNodes("//li[@class='transcript-clip']/div");
```

و سپس اطلاعات آن‌ها را استخراج نمود.

```
using System;
using System.Collections.Generic;
using System.Globalization;
using System.IO;
using System.Linq;
using System.Text;
using System.Text.RegularExpressions;
using System.Web;
using HtmlAgilityPack;

namespace PluralsightTranscripts
{
    public class TranscriptClip
    {
        public string Title { set; get; }
        public IList<TranscriptItem> TranscriptItems { set; get; }
    }

    public class TranscriptItem
    {
        public double StartTime { set; get; }
        public string Text { set; get; }
    }

    public class ExtractSubtitle
    {
        public static void ConvertToSrt(string fileName)
        {
            var transcriptClips = extractItems(fileName);
            var itemNumber = 1;
            foreach (var item in transcriptClips)
            {
                transcriptClipToSrt(item, itemNumber);
                itemNumber++;
            }
        }

        private static void transcriptClipToSrt(TranscriptClip item, int itemNumber)
        {
            var count = item.TranscriptItems.Count;
            var srtFileContent = transcriptItemsToSrt(item.TranscriptItems, count);
            var fileName = removeIllegalCharacters(string.Format("{0}-{1}.srt",
itemNumber.ToString("00"), item.Title));
            File.WriteAllText(fileName, srtFileContent);
        }

        private static string transcriptItemsToSrt(IList<TranscriptItem> items, int count)
        {
            var lineNumber = 1;
            var sb = new StringBuilder();
            for (int row = 0; row < count; row++)
            {
                sb.AppendLine(lineNumber.ToString(CultureInfo.InvariantCulture));
                sb.AppendLine(getTimeLine(items, count, row));
                sb.AppendLine(items[row].Text);
                sb.AppendLine(string.Empty);
                lineNumber++;
            }
            return sb.ToString();
        }

        private static string getTimeLine(IList<TranscriptItem> items, int count, int row)
        {
            var startTs = TimeSpan.FromSeconds(items[row].StartTime);
            var endTs = row + 1 < count ? TimeSpan.FromSeconds(items[row + 1].StartTime) :
TimeSpan.FromSeconds(items[row].StartTime + 5);
            return string.Format("{0} --> {1}", timeSpanToString(startTs), timeSpanToString(endTs));
        }

        private static string timeSpanToString(TimeSpan lineTs)
```

```
        {
            return string.Format("{0}:{1}:{2},{3}", lineTs.Hours.ToString("D2"),
lineTs.Minutes.ToString("D2"), lineTs.Seconds.ToString("D2"), lineTs.Milliseconds.ToString("D3"));
        }

        private static string removeIllegalCharacters(string fileName)
        {
            string regexSearch = string.Format("{0}{1}",
                                               new string(Path.GetInvalidFileNameChars()),
                                               new string(Path.GetInvalidPathChars()));
            var r = new Regex(string.Format("[{0}]", Regex.Escape(regexSearch)));
            return r.Replace(fileName, ".");
        }

        private static IList<TranscriptClip> extractItems(string fileName)
        {
            var htmlContent = File.ReadAllText(fileName);
            var results = new List<TranscriptClip>();

            var doc = new HtmlDocument
            {
                OptionCheckSyntax = true,
                OptionFixNestedTags = true,
                OptionAutoCloseOnEnd = true,
                OptionDefaultStreamEncoding = Encoding.UTF8
            };
            doc.LoadHtml(htmlContent);

            var nodes = doc.DocumentNode.SelectNodes("//li[@class='transcript-clip']/div");

            foreach (var node in nodes)
            {
                var itemsList = new List<TranscriptItem>();
                var title = node.ParentNode.ChildNodes.First(x => x.Name == "a").InnerText;

                foreach (var childNode in node.ChildNodes)
                {
                    if (childNode.Name != "a") continue;

                    var dataS = childNode.Attributes.First(x => x.Name == "data-s");
                    itemsList.Add(new TranscriptItem
                    {
                        StartTime = double.Parse(dataS.Value),
                        Text = HttpUtility.HtmlDecode(childNode.InnerText.Trim())
                    });
                }

                results.Add(new TranscriptClip { TranscriptItems = itemsList, Title = title });
            }

            return results;
        }
    }
}
```

اگر این اطلاعات را کنار هم قرار دهیم، به کلاس کمکی فوق خواهیم رسید. کار با گره‌های li شروع می‌شود. سپس در این گره‌ها، کلیه گره‌های a یا لینک‌ها، یافت شده و سپس dataS و متن آن‌ها استخراج می‌شوند. اگر این‌ها را نهایتا کنار هم قرار دهیم، می‌توان به فرمت SRT متداول که اکثر پخش کننده‌های فایل‌های تصویری قادر به پردازش آن‌ها هستند، رسید.
فرمت SRT ساختار ساده‌ای دارد. هر گفتگوی آن حداقل از سه سطر تشکیل می‌شود. سطر اول یک شماره خود افزاینده است. سطر دوم زمان شروع و پایان گفتگو را مشخص می‌کند و سطر سوم بیانگر متن گفتگو است. برای مثال:

```
1
00:00:01,636 --> 00:00:05,616
Hi, this is Scott Allen and this is the first module in the course design
```

دریافت پروژه کامل این مطلب

PluralsightTranscripts.zip

نظرات خوانندگان

نویسنده: الهام
تاریخ: ۱۳۹۲/۰۱/۰۹ ۲۲:۲۱

سلام آقای نصیری

آیا شما عضو سایت پلورال سایت هستید؟ چقدر پرداخت کردید و این مبلغ را چطور با توجه به وضع ایران واریز کردید؟ آیا ارزش عضو شدن رو داره؟

ببخشید چون من دانشجو هستم و بدنبال یادگیری حرفه ای برنامه نویسی و زبانم هم خوبه میخواستم از منابع انگلیسی استفاده کنم.

با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۰۹ ۲۲:۳۴

سلام؛ من عضو نیستم.

نویسنده: حسین
تاریخ: ۱۳۹۲/۰۱/۱۰ ۰:۱۳

این زیرنویس‌ها فقط برای اعضای اون سایت در دسترسه.از کجا میشه بهشون دسترسی پیدا کرد ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۱/۱۰ ۰:۱۶

لطفا روی لینک مطرح شده در سطر اول مطلب فوق کلیک کنید . کل بحث جاری در مورد استخراج اطلاعات و تبدیل فرمت خاص صفحه وبی بود که ملاحظه می‌کنید. این صفحه هم عمومی است (هر چند ظاهر ساده‌ای دارد، اما پشت صحنه و سورس آن، متن زمانبندی شده کل دوره است).

نویسنده: علیرضا جهانشاهلو
تاریخ: ۱۳۹۲/۰۱/۱۴ ۲:۴۰

اتفاقا سایت Lynda هم از همین روش استفاده میکنه و من با کمی تغییر موفق شدم که فایل‌های Transcript آموزشی هاشو استخراج کنم.

ممنون مهندس

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۲/۰۷/۲۷ ۱۵:۴۳

ظاهراً ساختار عوض شده به این شکل (البته در اینجا data-s حذف شده و مقدار آن به صورت رشته ایی در انتهای مقدار ng-click اضافه شده است به صورت start=39.796) :

```
<li class="transcript-clip">
<a href="javascript:void(0)" ng-click="launchPlayerWindow('http://pluralsight.com/training',
'author=scott-allen&amp;name=mvc4-building-m1-intro&amp;mode=live&amp;clip=0&amp;course=mvc4-
building');">Introduction</a><br>
```

```
    <div>
<a href="javascript:void(0)" ng-click="launchPlayerWindow('http://pluralsight.com/training',
'author=scott-allen&amp;name=mvc4-building-m1-intro&amp;mode=live&amp;clip=0&amp;course=mvc4-
building&amp;start=39.796');">and also have an understanding of the design goals of the MVC
framework.</a>
        <a href="javascript:void(0)" ng-click="launchPlayerWindow('http://pluralsight.com/training',
'author=scott-allen&amp;name=mvc4-building-m1-intro&amp;mode=live&amp;clip=0&amp;course=mvc4-
building&amp;start=43.796');">So, let's get started.</a>
    </div>
</li>
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۷/۲۷ ۱۷:۲۱

- البته من عضو نیستم و به نظر جدیدا عنوان کردند «Sorry, transcripts are only available to subscribers».
- در کدهای فوق، فقط این چند سطر باید تغییر کنند:

```
//var dataS = childNode.Attributes.First(x => x.Name == "data-s");
                    var dataS = childNode.Attributes.First(x => x.Name == "ng-click");
                    var startTime = new Regex("(?s)start=(.+?)'").Matches(dataS.Value)
                                                                 .OfType<Match>()
                                                                 .First()
                                                                 .Groups[1]
                                                                 .Value;
                    itemsList.Add(new TranscriptItem
                    {
                        StartTime = double.Parse(startTime),
                        Text = HttpUtility.HtmlDecode(childNode.InnerText.Trim())
                    });
```

نویسنده: پویان
تاریخ: ۱۳۹۳/۰۱/۲۴ ۱۶:۲۴

با سلام
الان که حتما باید در سایت plural sight عضو باشیم راهی نیست تا زیرنویس بگیریم ؟
ایا شما زیرنویس بعضی از فیلم‌ها را دارید ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۴ ۱۶:۳۸

فایل تورنت پیوست شده حاوی مثال‌ها و زیرنویس‌های 52 دوره هست:
srt_only.zip

عنوان: **دریافت زمانبندی شده به روز رسانی‌های آنتی ویروس Symantec به کمک کتابخانه‌های Quartz.NET و Html Agility Pack**
نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۲/۰۴/۰۳ ۸:۳۰
آدرس: www.dotnettips.info
برچسب‌ها: Regular expressions, Quartz.NET, Html Agility Pack, Asynchronous Programming

در این رابطه آقای راد در دو قسمت به صورت مختصر و مفید این کتابخانه قدرتمند رو همراه با ارائه چندین مثال کاربردی معرفی کردند:
قسمت اول
قسمت دوم
در تکمیل قسمت‌های فوق بنده می‌خوام مثالی رو در این رابطه براتون بذارم، هدف از ارائه این مثال اتوماتیک سازی یک فرآیند روتین می‌باشد، به این صورت که در جایی که بنده مشغول به کار هستم یک سری لایسنس آنتی ویروس برای کلاینت‌ها در یک شبکه با مقیاس متوسط تهیه گردیده است، حال یک نسخه رایگان نیز برای کاربرانی که قصد دارند آنتی ویروس را برای سیستم شخصی خود نصب کنند نیز موجود می‌باشد که نیاز به آپدیت دارد معمولا آپدیت‌ها هر چند روز یکبار یا هر هفته در دو نسخه 64 و 32 بیتی ارائه می‌شوند، روال معمول برای دریافت آپدیت مراجعه به سایت و دانلود نسخه‌های مربوطه میباشد.

حال توسط کتابخانه قدرتمند Quartz.NET این فرآیند روتین را به صورت اتوماتیک می‌خواهیم انجام دهیم، استفاده از کتابخانه ذکر شده سخت نیست همانطور که در دو مطلب قبلی مرتبط ذکر گردیده، تنها پیاده سازی چندین اینترفیس است و بس.

```
namespace SymantecUpdateDownloader
{
    using System;
    using System.IO;
    using Quartz;
    using Quartz.Impl;
    using System.Globalization;
    public class TestJob : IJob
    {
        public void Execute(IJobExecutionContext context)
        {
            new Download().Scraping();
        }
    }
    public interface ISchedule
    {
        void Run();
    }
    public class TestSchedule : ISchedule
    {
        public void Run()
        {

            DateTimeOffset startTime = DateBuilder.FutureDate(2, IntervalUnit.Second);

            IJobDetail job = JobBuilder.Create<HelloJob>()
                                       .WithIdentity("job1")
                                       .Build();

            ITrigger trigger = TriggerBuilder.Create()
                                             .WithIdentity("trigger1")
                                             .StartAt(startTime)
                                             .WithDailyTimeIntervalSchedule(x =>
x.OnEveryDay().StartingDailyAt(new TimeOfDay(7, 0)).WithRepeatCount(0))
                                             .Build();

            ISchedulerFactory sf = new StdSchedulerFactory();
            IScheduler sc = sf.GetScheduler();
            sc.ScheduleJob(job, trigger);

            sc.Start();
        }
    }
}
```

در این کد که همانند کدهای پیشنیازهای مطلب است، در خط 33 از متد WithDailyTimeIntervalSchedule استفاده شده است

و همانطور که مشخص است وظیفه تعیین شده و هر روز ساعت 7 اجرا میشود.
مورد بعدی عملیات دانلود فایل می‌باشد که در ادامه مشاهده خواهید کرد، صفحه ایی که لینک فایل‌های دانلود را ارائه داده است دو نسخه مد نظر ما را در ابتدا لیست کرده است و با استفاده از web scrapingمی توانیم موارد تعیین شده را استخراج کنیم برای این منظور از کتابخانه htmlagilitypack استفاده میکنیم، تطبیق دو مورد(لینک) اول جهت دریافت نسخه‌های 32 و 64 بیتی به کمک Regular Expression میسر است و همانطور که در شکل زیر مشاهده میکنید از سمت چپ تاریخ به صورت 8 رقم، سه رقم قسمت دوم و ارقام و حروف قسمت سوم است به اضافه پسوند فایل مشخص است :


```
public class Download
    {
        static WebClient wc = new WebClient();
        static ManualResetEvent handle = new ManualResetEvent(true);

        private DateTime myDate = new DateTime();
        public void Scraping()
        {
            using (WebClient client = new WebClient())
            {
                client.Encoding = System.Text.Encoding.UTF8;
                var doc = new HtmlAgilityPack.HtmlDocument();
                ArrayList result = new ArrayList();

doc.LoadHtml(client.DownloadString("https://www.symantec.com/security_response/definitions/download/det
ail.jsp?gid=savce"));
                var tasks = new List<Task>();
                foreach (var href in doc.DocumentNode.Descendants("a").Select(x =>
x.Attributes["href"]))
                {
                    if (href == null) continue;
                    string s = href.Value;
                    Match m = Regex.Match(s, @"http://definitions.symantec.com/defs/(\d{8}-\d{3}-
v5i(32|64)\.exe)");
                    if (m.Success)
                    {
                        Match date = Regex.Match(m.Value, @"(\d{4})(\d{2})(\d{2})");
                        Match filename = Regex.Match(m.Value, @"\d{8}-\d{3}-v5i(32|64)\.exe");
                        int year = Int32.Parse(date.Groups[0].Value);
                        int month = Int32.Parse(date.Groups[1].Value);
                        int day = Int32.Parse(date.Groups[3].Value);

                        myDate = new DateTime(
                                Int32.Parse(date.Groups[1].Value),
                                Int32.Parse(date.Groups[2].Value),
                                Int32.Parse(date.Groups[3].Value));
                        if (myDate == DateTime.Today)
                        {
                            tasks.Add(DownloadUpdate(m.Value, filename.Value));
                        }
                        else
                        {
                            MessageBox.Show("امروز آپدیت موجود نیست");
                        }
                    }
                }
                DownloadTask = Task.WhenAll(tasks);
            }
        }
        private static Task DownloadTask;
        private Task DownloadUpdate(string url, string fileName)
        {
            var wc = new WebClient();
            return wc.DownloadFileTaskAsync(new Uri(url), @"\\10.1.0.15\SymantecUpdate\\" + fileName);
        }
```

```
    }
```

توضیح کدهای فوق :
ابتدا توسط متد LoadHtml خط 14 صفحه مورد نظر که حاوی لینک‌ها می‌باشد رو Load میکنیم، سپس توسط یک حلقه foreach خط 16 مقدار خصوصیت href تمام لینک‌های موجود در صفحه را استخراج میکنیم مثلا مقدار خصوصیت href در لینک‌ها به صورت زیر می‌باشد :

```
http://definitions.symantec.com/defs/20130622-007-v5i32.exe
```

```
http://definitions.symantec.com/defs/20130622-007-v5i64.exe
```

همانطور که مشخص است در دو مورد فوق تنها نام فایل متفاوت می‌باشد، همانطور که بحث شد برای نام فایل‌ها هم می‌توانیم یک Pattern را به صورت زیر داشته باشیم :

```
(\d{8}-\d{3}-v5i(32|64)\.exe)
```

در خط 20 نیز عملیات تطبیق تمام hrefهای موجود در صفحه را توسط Regular Expression فوق تطبیق می‌دهیم، اگر تطبیق با موفقیت انجام پذیرفت باید نام فایل و همچنین تاریخ موجود در نام فایل را نیز توسط دو Regular Expression استخراج کنیم(خط 23 و 24) در ادامه برای جدا کردن مقادیر سال ، ماه ، روز از امکان Groups در RegEx استفاده کرده ایم:

```
int year = Int32.Parse(date.Groups[0].Value);
int month = Int32.Parse(date.Groups[1].Value);
int day = Int32.Parse(date.Groups[3].Value);
```

در ادامه تاریخ استخراج شده را با تاریخ روز جاری مقایسه می‌کنیم اگر مساوی بود عملیات دانلود فایل‌ها توسط یک Task تعریف شده به صورت همزمان بر روی سرور مربوطه دانلود می‌شوند.
البته لازم به ذکر است که کدهای فوق مسلما نیاز به Refactoring دارند منتها هدف از ارائه این مثال آشنایی بیشتر با کتابخانه‌های فوق می‌باشد.
نکته آخر اینکه برنامه فوق به حالت‌های مختلفی می‌تواند اجرا گردد مثل یک برنامه وب یا یک سرویس ویندوزی و ... ، بهترین حالت یک سرویس ویندوز می‌باشد، ولی در حالت خام در حال حاضر یک ویندوز اپلیکیشن ساده می‌باشد که بر روی سرور RUN شده است که در آینده به صورت یک سرویس ویندوز ارائه خواهد شد.

نظرات خوانندگان

نویسنده: افشین
تاریخ: ۱۳۹۲/۰۴/۱۵ ۸:۱

یه سئوال دارم که همیشه ذهنم رو مشغول کرده
مگه اینترفیس فقط امضا روال‌ها رو نداره؟ پس یک کلاس نیاز داره که بتونه اون متدها رو پیاده سازی کنه و ما ازش استفاده کنیم غیر از اینه؟
پس در کد زیر

```
IJobDetail job = JobBuilder.Create<HelloJob>()
```

مجبوریم از اینترفیس به عنوان متغیر استفاده کنیم؟

نویسنده: سیروان عفیفی
تاریخ: ۱۳۹۲/۰۴/۱۵ ۱۲:۴۲

بله به همین صورته،این مطلب رو درباره اینترفیس و این مطلب رو درباره متدهای Generic بخونید،
متد Create یک متد Generic است که نام کلاسی رو که اینترفیس IJob و Implement کرده را قبول میکند، و در نهایت مقدار بازگشتی این متد از نوع IJobDetail است.

عنوان: **ارسال PingBack در ASP.NET**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۰۸ ۷:۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, Regular expressions, Html Agility Pack, SEO

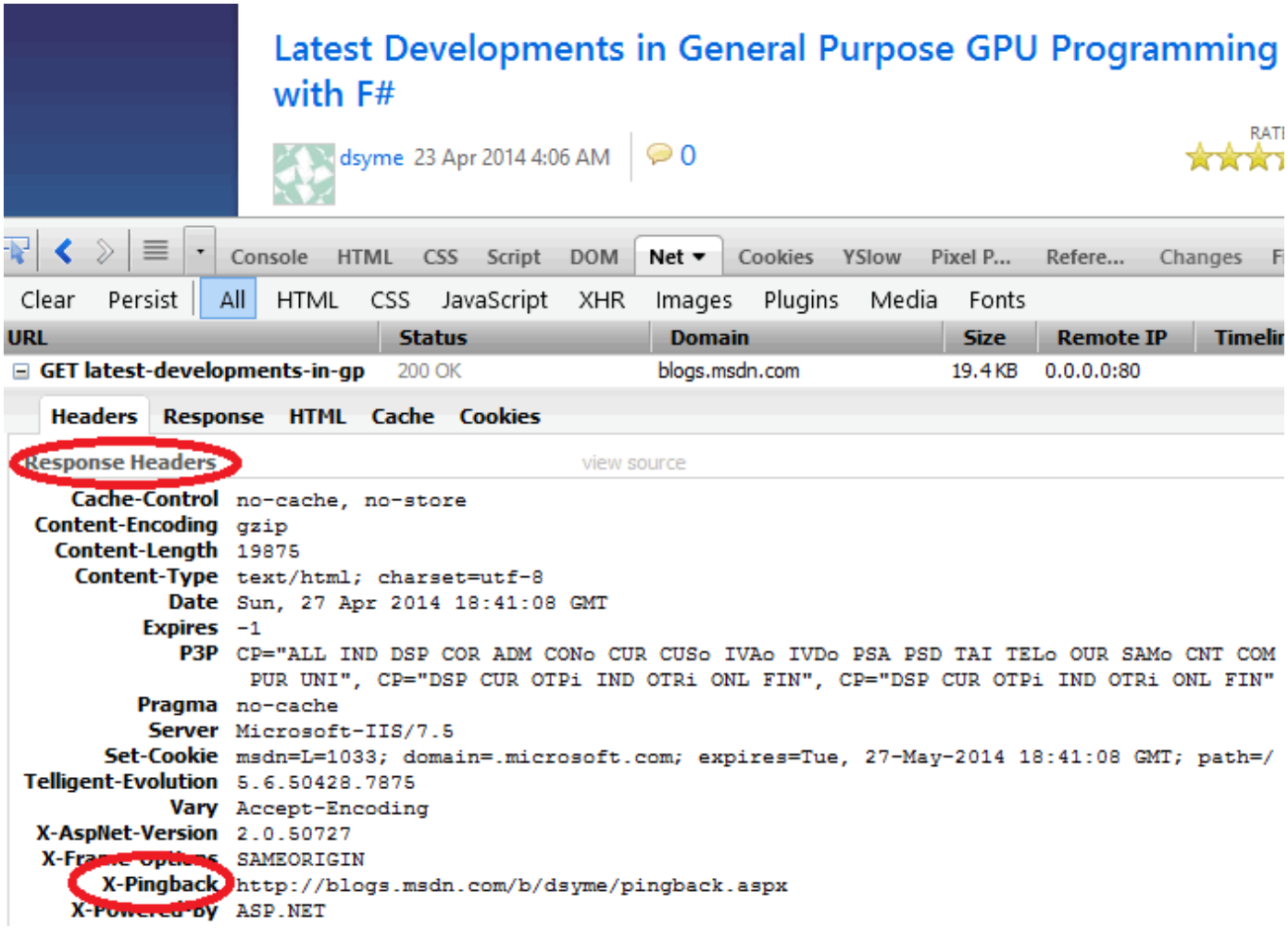
Pingback یکی از روش‌های اطلاع رسانی به سایت‌های دیگر در مورد لینک دادن به آن‌ها در سایت خود است. برای مثال من لینکی از یکی از مطالب شما را در متن جاری خودم قرار می‌دهم. سپس به وسیله‌ی ارسال یک ping، در مورد انجام اینکار به شما اطلاع رسانی می‌کنم. حاصل آن عموما قسمت معروف ping-backs سایت‌ها است. این مورد نیز یکی از روش‌های مؤثر SEO در گرفتن backlink است و تبلیغ محتوا.
کار کردن با پروتکل Ping-back آنچنان ساده نیست؛ از این جهت که تبادل ارتباطات آن با پروتکل XML-RPC انجام می‌شود. XML-RPC نیز توسط PHP کارها بیشتر مورد استفاده قرار می‌گیرد؛ بجای استفاده از پروتکل‌های استاندارد وب سرویس‌ها مانند Soap و امثال آن. پیاده سازی‌های ابتدایی Pingback نیز مرتبط است به Wordpress معروف که با PHP تهیه شده‌است. در ادامه نگاهی خواهیم داشت به جزئیات پیاده سازی ارسال ping-back توسط برنامه‌های ASP.NET.

یافتن آدرس وب سرویس سایت پذیرای Pingback

اولین قدم در پیاده سازی Pingback، یافتن آدرسی است که باید اطلاعات مورد نظر را به آن ارسال کرد. این آدرس عموما به دو طریق ارائه می‌شود:
الف) در هدری به نام x-pingback و یا pingback
ب) در قسمتی از کدهای HTML صفحه به شکل

```
<link rel="pingback" href="pingback server">
```

برای مثال اگر به وبلاگ‌های MSDN دقت کنید، هدر x-pingback را می‌توانید در خروجی وب سرور آن‌ها مشاهده کنید:



همانطور که ملاحظه می‌کنید، نیاز است Response header را آنالیز کنیم.

```
private Uri findPingbackServiceUri()
        {
            var request = (HttpWebRequest)WebRequest.Create(_targetUri);
            request.UserAgent = UserAgent;
            request.Timeout = Timeout;
            request.ReadWriteTimeout = Timeout;
            request.Method = WebRequestMethods.Http.Get;
            request.AutomaticDecompression = DecompressionMethods.GZip | DecompressionMethods.Deflate;
            using (var response = request.GetResponse() as HttpWebResponse)
            {
                if (response == null) return null;

                var url = extractPingbackServiceUriFormHeaders(response);
                if (url != null)
                    return url;

                if (!isResponseHtml(response))
                    return null;

                using (var reader = new StreamReader(response.GetResponseStream()))
                {
                    return extractPingbackServiceUriFormPage(reader.ReadToEnd());
                }
            }
        }

        private static Uri extractPingbackServiceUriFormHeaders(WebResponse response)
        {
            var pingUrl = response.Headers.AllKeys.FirstOrDefault(header =>
                                header.Equals("x-pingback", StringComparison.OrdinalIgnoreCase) ||
                                header.Equals("pingback", StringComparison.OrdinalIgnoreCase));
```

```
            return getValidAbsoluteUri(pingUrl);
        }

        private static Uri extractPingbackServiceUriFormPage(string content)
        {
            if (string.IsNullOrWhiteSpace(content)) return null;
            var regex = new Regex(@"(?s)<link\srel=""pingback""\shref=""(.+?)""",
RegexOptions.IgnoreCase);
            var match = regex.Match(content);
            return (!match.Success || match.Groups.Count < 2) ? null :
getValidAbsoluteUri(match.Groups[1].Value);
        }

        private static Uri getValidAbsoluteUri(string url)
        {
            Uri absoluteUri;
            return string.IsNullOrWhiteSpace(url) || !Uri.TryCreate(url, UriKind.Absolute, out
absoluteUri) ? null : absoluteUri;
        }

        private static bool isResponseHtml(WebResponse response)
        {
            var contentTypeKey = response.Headers.AllKeys.FirstOrDefault(header =>
                                        header.Equals("content-type",
StringComparison.OrdinalIgnoreCase));
            return !string.IsNullOrWhiteSpace(contentTypeKey) &&
                    response.Headers[contentTypeKey].StartsWith("text/html",
StringComparison.OrdinalIgnoreCase);
        }
```

نحوه‌ی استخراج آدرس سرویس Pingback یک سایت را در کدهای فوق ملاحظه می‌کنید.
targetUri، آدرسی است از یک سایت دیگر که در سایت ما درج شده‌است. زمانیکه این صفحه را درخواست می‌کنیم، response.Headers.AllKeys حاصل می‌تواند حاوی کلید x-pingback باشد یا خیر. اگر بلی، همینجا کار پایان می‌یابد. فقط باید مطمئن شد که این آدرس مطلق است و نه نسبی. به همین جهت در متد getValidAbsoluteUri، بررسی بر روی UriKind.Absolute انجام شده‌است.
اگر هدر فاقد کلید x-pingback باشد، قسمت ب را باید بررسی کرد. یعنی نیاز است محتوای Html صفحه را برای یافتن link rel=pingback بررسی کنیم. همچنین باید دقت داشت که پیش از اینکار نیاز است حتما بررسی isResponseHtml صورت گیرد. برای مثال در سایت شما لینکی به یک فایل 2 گیگابایتی SQL Server درج شده‌است. در این حالت نباید ابتدا 2 گیگابایت فایل دریافت شود و سپس بررسی کنیم که آیا محتوای آن حاوی link rel=pingback است یا خیر. اگر محتوای ارسالی از نوع text/html بود، آنگاه کار دریافت محتوای لینک انجام خواهد شد.

ارسال Ping به آدرس سرویس Pingback

اکنون که آدرس سرویس pingback یک سایت را یافته‌ایم، کافی است ping ایی را به آن ارسال کنیم:

```
public void Send()
        {
            var pingUrl = findPingbackServiceUri();
            if (pingUrl == null)
                throw new NotSupportedException(string.Format("{0} doesn't support pingback.",
_targetUri.Host));

            sendPing(pingUrl);
        }

        private void sendPing(Uri pingUrl)
        {
            var request = (HttpWebRequest)WebRequest.Create(pingUrl);
            request.UserAgent = UserAgent;
            request.Timeout = Timeout;
            request.ReadWriteTimeout = Timeout;
            request.Method = WebRequestMethods.Http.Post;
            request.ContentType = "text/xml";
            request.ProtocolVersion = HttpVersion.Version11;
            makeXmlRpcRequest(request);
            using (var response = (HttpWebResponse)request.GetResponse())
            {
                response.Close();
```

```
        }
    }

    private void makeXmlRpcRequest(WebRequest request)
    {
        var stream = request.GetRequestStream();
        using (var writer = new XmlTextWriter(stream, Encoding.ASCII))
        {
            writer.WriteStartDocument(true);
            writer.WriteStartElement("methodCall");
            writer.WriteElementString("methodName", "pingback.ping");
            writer.WriteStartElement("params");

            writer.WriteStartElement("param");
            writer.WriteStartElement("value");
            writer.WriteElementString("string", Uri.EscapeUriString(_sourceUri.ToString()));
            writer.WriteEndElement();
            writer.WriteEndElement();

            writer.WriteStartElement("param");
            writer.WriteStartElement("value");
            writer.WriteElementString("string", Uri.EscapeUriString(_targetUri.ToString()));
            writer.WriteEndElement();
            writer.WriteEndElement();

            writer.WriteEndElement();
            writer.WriteEndElement();
        }
    }
```

اینبار HttpWebRequest تشکیل شده از نوع post است و نه get. همچنین مقداری را که باید ارسال کنیم نیاز است مطابق پروتکل XML-RPC باشد. برای کار با XML-RPC در دات نت یا می‌توان از کتابخانه‌ی Cook Computing's XML-RPC.Net استفاده کرد و یا مطابق کدهای فوق، دستورات آن‌را توسط یک XmlTextWriter کنار هم قرار داد و نهایتا در درخواست Post ارسالی درج کرد.
در اینجا sourceUri آدرس صفحه‌ای در سایت ما است که targetUri ایی (آدرسی از سایت دیگر) در آن درج شده‌است. در یک pinback، صرفا این دو آدرس به سرویس دریافت کننده‌ی pingback ارسال می‌شوند.
سپس سایت دریافت کننده‌ی ping، ابتدا sourceUri را دریافت می‌کند تا عنوان آن‌را استخراج کند و همچنین بررسی می‌کند که آیا targetUri، در آن درج شده‌است یا خیر (آیا spam است یا خیر)؟
تا اینجا اگر این مراحل را کنار هم قرار دهیم به کلاس Pingback ذیل خواهیم رسید:

Pingback.cs

نحوه‌ی استفاده از کلاس Pingback تهیه شده

کار ارسال Pingback عموما به این نحو است: هر زمانیکه مطلبی یا یکی از نظرات آن، ثبت یا ویرایش می‌شوند، نیاز است Pingbackهای آن ارسال شوند. بنابراین تنها کاری که باید انجام شود، استخراج لینک‌های خارجی یک صفحه و سپس فراخوانی متد Send کلاس فوق است.
یافتن لینک‌های یک محتوا را نیز می‌توان مانند متد extractPingbackServiceUriFormPage فوق، توسط یک Regex انجام داد و یا حتی با استفاده از کتابخانه‌ی معروف HTML Agility Pack :

```
var doc = new HtmlWeb().Load(url);
var linkTags = doc.DocumentNode.Descendants("link");
var linkedPages = doc.DocumentNode.Descendants("a")
                                  .Select(a => a.GetAttributeValue("href", null))
                                  .Where(u => !String.IsNullOrEmpty(u));
```