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

نظرات خوانندگان

نویسنده: علی یگانه مقدم
تاریخ: ۱۳۹۳/۱۲/۲۹ ۱:۵۰

یه مشکلی که الان من با \wبهش برخوردم این هست که حروف فارسی رو هم برمیگردونه

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۱۲/۲۹ ۹:۴۸

- اگر می‌خواهید \w حروف یونیکد را در نظر نگیرد، باید ویژگی ECMAScript را فعال کنید.
- یک روش دیگر تعیین اعتبار ایمیل، استفاده از کلاس MailAddress دات نت است. اگر ایمیل وارد شده‌ی به آن معتبر نباشد، یک استثناء را صادر می‌کند:

```
public static bool IsValidEmail(string to)
        {
            if (string.IsNullOrWhiteSpace(to))
                return false;

            try
            {
                var toEmail = new MailAddress(to);
                return toEmail != null;
            }
            catch
            {
                return false;
            }
        }
```