عنوان: **مدیریت دانلودهای همزمان از یک سایت و بحث تایم آوت**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۸/۰۷ ۰۱:۲۶:۳۱
آدرس: www.dotnettips.info
برچسب‌ها: system.net

یک سرویس ویندوز ان تی با سی شارپ نوشته‌ام که کارش مراجعه به یک سری آدرس RSS و ذخیره سازی آنها به صورت آنالیز شده در یک دیتابیس SQL server است (این مورد ضعفی است که اکثر برنامه‌های فیدخوان دارند و پس از مدتی کار با آنها این احساس را دارید که اطلاعات گذشته را از دست داده‌اید).
در طی آزمایش اولیه این سرویس، به مشکل عجیب timeout پس از باز کردن برای مثال سومین یا چهارمین thread همزمان برای دانلود کردن اطلاعات بر خوردم. همه چیز درست بود، از کلاس‌ها، دریافت اطلاعات از وب و غیره، اما برنامه کار نمی‌کرد. این مشکل فقط هم با feedburner.com رخ می‌داد (همانطور که مطلع هستید feedburner.com سرویسی را جهت پیگیری آمار مشترکین فیدهای شما ارائه می‌دهد که بسیار جالب است. برای مثال چند نفر مشترک دارید، یا یک سری نمودار و غیره. به همین جهت رسم شده است که اکثر سایت‌ها فیدهای خودشان را در این سایت نیز ثبت می‌کنند).
پس از مدتی جستجو به نکته جالب زیر برخوردم که شاید برای شما هم در آینده مفید باشد:
مطابق RFC2068 - Hypertext Transfer Protocol -- HTTP/1.1 ، شما تنها مجازید 2 کانکشن فعال به یک سایت باز کنید. این علت تایم آوت در سومین thread ایجاد شده بود. برای مثال IE این مورد را محترم می‌شمارد. در دات نت نیز به صورت پیش فرض این محدودیت قرار داده شده است که به‌سادگی می‌توان آنرا تغییر داد. برای این منظور باید یک فایل app.config به پروژه اضافه کرد و سپس خطوط زیر را به آن افزود:

```
<configuration>
<system.net>
<connectionManagement>
 <add address="*" maxconnection="100" />
</connectionManagement>
</system.net>
</configuration>
```

بعد از این تغییر مشکل timeout برنامه حل شد.

برای مدیریت چندین ترد همزمان دانلود کننده و در صف قرار دادن آنها در این پروژه، از کتابخانه سورس باز زیر استفاده کردم:
http://www.codeplex.com/smartthreadpool

مآخذ:
http://msdn.microsoft.com/en-us/library/fb6y0fyc.aspx
http://www.faqs.org/rfcs/rfc2068.html
http://vahidnasiri.blogspot.com
http://odetocode.com/Blogs/scott/archive/2004/06/08/272.aspx

پ.ن.
برای اینکه در بلاگر بتوانید متون حاوی xml را ارسال کنید باید از سرویس زیر استفاده کنید
/http://www.elliotswan.com/postable

نظرات خوانندگان

نویسنده: ناصر حاجلو :- khatmikhi
تاریخ: ۱۳۸۷/۰۸/۰۵ ۱۸:۳۷:۰۰

برای دور زدن دانلوداز سرورهای مختلف که مشکل محدودیت کانکشن فعال دارند هم می شود از اینها استفاده کرد ؟ البته من این کار رو برای دانلود نمی خوام ، بلکه قصد نهایی باز کردن تعداد بسیار زیاد کانکشن به یک سرور خاص و نهایتا از کار انداختن آن است . چیزی مثل حملات DDOS .
البته من فکر نمی کنم این حرفی که شما زدید در این مورد که من گفتم کار کند چون معمولا Firewallهایی که روی سرور هست این اجازه رو به شما نمیده . و نهایتا همه رو می بنده .
در این مورد می تونید راهنمایی کنه ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۰۸/۰۵ ۱۹:۲۵:۰۰

در این مورد خیر.
اگر سرور یا حتی برنامه برای این نوع حملات آماده نشده باشند، بله. این روش‌ها می‌تونه عملیات سایت رو مختل کنه. البته هدف من فقط دریافت فید از یک سایت مادر بود :)
مطلبی رو چند وقت پیش در سایت آقای Omar AL Zabir دیدم که در همین ارتباط بود. نحوه ایجاد این نوع حملات و نحوه دفاع توسط یک برنامه هوشمند که برای این موارد آماده شده:
مشاهده مطلب:
http://msmvps.com/blogs/omar/archive/2007/03/24/prevent-denial-of-service-dos-attacks-in-your-web-application.aspx
مثالی هم که در سایت ایشون برای حمله عنوان شده عملا با پیش فرض‌های دات نت (حداکثر 2 کانکشن همزمان) کار خاصی رو انجام نمیده و نهایتا timeout خواهند گرفت، مگر اینکه ...

نویسنده: hajloo
تاریخ: ۱۳۸۷/۰۸/۰۸ ۱۲:۳۳:۰۰

لینک خوبی بود اما کافی نبود اگر بعدا چیز بهتری پیدا کردی یکجایی توی وبلاگت بنویس (چون این وبلاگ رو دنبال می کنم و حتما پیگیری می کنم)
این رو هم بگم که اصولا من این کار رو واقعا برای هک یا خوابوندن سرور نمی خوام اما برام جالب شد که اگر این اتفاق بیفته خیلی خطات دیگه هم می تونه بوجود بیاد .
زیاد وقتت رو نمی گیرم ولی اگر تونستی در ارتباط با سوال من یک پست بنویس فکر کنم برای خیلی ها جالب باشه . ممنون

نویسنده: مهدی
تاریخ: ۱۳۸۷/۱۱/۳۰ ۱۵:۰۶:۰۰

سلام
من دنبال یه راهی میگشتم که با استفاده از خود سی شارپ بشه تشخیص داد که به اینترنت کانکشن داریم یا نه؟
;()System.Net.NetworkInformation.NetworkInterface.GetAllNetworkInterfaces
این فقط شبکه ها رو برمیگردونه ولی تشخیص کانکت بودن به اینترنت رو نمیده.

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۱/۳۰ ۱۵:۴۳:۰۰

راهی که در تمام زبان‌های برنامه نویسی برای چک کردن مطمئن دسترسی به اینترنت وجود دارد دانلود کردن یک صفحه از اینترنت است. عموما سایت گوگل پیشنهاد می‌شود چون احتمال داون بودن آن در حد صفر است.
با استفاده از HttpWebResponse صفحه اول گوگل را دانلود کنید. اگر StatusCode مربوط به GetResponse مساوی OK بود یعنی

دسترسی به اینترنت دارید.

نویسنده: مهدی
تاریخ: ۱۳۸۷/۱۲/۰۱ ۱۱:۲۹:۰۰

خیلی ممنونم

عنوان: **مونیتور کردن وضعیت یک سایت در دات نت**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۱/۱۵ ۱۴:۰۷:۴۰
آدرس: www.dotnettips.info
برچسب‌ها: system.net

فرض کنید می‌خواهیم وضعیت یک سایت را از لحاظ قابلیت دسترسی مونیتور کنیم، آیا Up است، Down است و امثال آن. یک سری از وب سرورها ping را بسته‌اند (ICMP Replies). بنابراین الزاما با استفاده از این روش ساده نمی‌توان به مقصود رسید.
خوشبختانه انجام این‌کار با استفاده از فضای نام استاندارد System.Net و کلاس HttpWebRequest ، بدون نیاز به هیچگونه کلاس یا کامپوننت خارجی، به سادگی قابل انجام است. کلاس زیر به همین منظور تهیه شده است:

```
using System;
using System.Net;

  public class CSiteMonitor
  {
      public struct UrlHeaderInfo
      {
          public DateTime _lastModified;
          public string _statusCode;
          public string _errorMessage;
      }

      /// <summary>
      /// آیا آدرس اینترنتی وارد شده معتبر است؟
      /// </summary>
      /// <param name="url">آدرس مورد نظر جهت بررسی</param>
      /// <returns></returns>
      public static bool IsValidURL(string url)
      {
          try
          {
              Uri uri = new Uri(url);
              return (uri.Scheme == Uri.UriSchemeHttp) || (uri.Scheme == Uri.UriSchemeHttps);
          }
          catch { return false; }
      }
      /// <summary>
      /// آدرس اینترنتی جهت بررسی
      /// </summary>
      /// <param name="url"></param>
      /// <returns></returns>
      /// <exception cref="ArgumentException">آدرس اینترنتی وارد شده معتبر نیست</exception>
      public static UrlHeaderInfo GetSiteHeaderInfo(string url)
      {
          if (!IsValidURL(url))
              throw new ArgumentException("آدرس اینترنتی وارد شده معتبر نیست", "url");

          UrlHeaderInfo hhi = new UrlHeaderInfo { _lastModified = DateTime.Now, _statusCode = "NOK!",
_errorMessage = string.Empty };

          HttpWebRequest request = (HttpWebRequest)WebRequest.Create(url);
          //request.Proxy
          request.Method = "HEAD";
          request.AllowAutoRedirect = true;
          request.UserAgent = "Mozilla/5.0 (Windows; U; Windows NT 5.0; ; rv:1.8.0.7) Gecko/20060917
Firefox/1.9.0.1";
          request.Timeout = 1000 * 300;
          request.AutomaticDecompression = DecompressionMethods.GZip | DecompressionMethods.Deflate;

          try
          {
              using (HttpWebResponse response = (HttpWebResponse) request.GetResponse())
              {
                  hhi._statusCode = response.StatusCode.ToString();
                  hhi._lastModified = response.LastModified;
              }
          }
          catch (Exception ex)
          {
              hhi._errorMessage = ex.Message;
          }
```

```
            return hhi;
        }
    }
```

توضیحات:

- در متد GetSiteHeaderInfo نیاز بود تا از یک تابع بیش از یک خروجی داشته باشیم. راه‌های زیاد برای انجام این‌کار هست.برای مثال:

الف)ارائه خروجی‌ها به صورت یک آرایه. زیاد جالب نیست، چون اگر شخصی دقیقا مستندات متد شما را مطالعه نکند نمی‌داند که ترتیب خروجی‌ها چگونه است و هر کدام چه معنایی دارند.

ب)ارائه خروجی‌ها با استفاده از آرگومان‌هایی از نوع out یا ref . در دنیای شیء گرایی این نوع روش‌ها را باید منسوخ شده در نظر گرفت و صرف سازگاری با زبان‌هایی مانند C که این روش در آن‌ها رواج دارد (استفاده از آرگومان‌هایی از نوع اشاره‌گر) باید به آن نگاه کرد و نه بیشتر.

ج)خروجی‌ها را به صورت یک کلاس یا struct درنظر گرفت تا استفاده کننده دقیقا بداند که فیلد خروجی چه معنایی دارد و هم چنین دقیقا چه تعداد خروجی مد نظر است.

- حتما باید از try/finally جهت اطمینان حاصل نمودن از بسته شدن response استفاده شود، در غیر اینصورت پس از دو خطای متوالی حاصل شده عملا دیگر نمی‌توان از شیء response استفاده کرد. البته همانطور که پیش تر نیز ذکر شد، عبارت using توسط کامپایلر به try/finally بست داده می‌شود، بنابراین جهت خوانایی بیشتر کد بهتر است از این روش استفاده شود.

- جهت بلاک نشدن درخواست بهتر است از یک UserAgent کمک گرفته شود.

- جهت بررسی اعتبار یک آدرس اینترنتی یا می‌توان از Regular expressions استفاده کرد یا از شیء Uri که روش آن‌را ملاحظه می‌کنید.

- اگر در شبکه داخلی خود از پروکسی استفاده می‌شود، می‌توان قسمت request.Proxy را با شیء پروکسی تنظیم شده مطابق مشخصات پروکسی سرور خود، بکار برد.

- در این مثال بیشتر هدف پیاده سازی کلاس دریافت اطلاعات هدر سایت بود و از ارائه کدهای مربوط به تایمر یا یک ترد جهت بررسی متوالی وضعیت سایت صرفنظر شد.

مثالی در مورد نحوه‌ی استفاده از کلاس فوق:

```
CSiteMonitor.UrlHeaderInfo info = CSiteMonitor.GetSiteHeaderInfo("http://www.google.com");
MessageBox.Show(info._statusCode);
```

نظرات خوانندگان

نویسنده: Anonymous
تاریخ: ۱۳۸۷/۱۱/۱۸ ۰۰:۰۹:۰۰

بسیار عالی
ممنون

عنوان: **سریع‌تر کردن عملیات دریافت اطلاعات از وب، توسط یک برنامه ASP.Net**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۱/۲۱ ۱۸:۴۷:۵۵
آدرس: www.dotnettips.info
گروه‌ها: system.net

فرض کنید یک برنامه ASP.Net نوشته‌اید که کار آن نمایش یک سری فید از سایت‌های مختلف است یا دریافت وضعیت آب و هوا از یک وب سرویس و نمایش آن در سایت می‌باشد و امثال آن (استفاده از XmlDom/Reader ، WebClient ، WebRequest و ...).
اگر همین عملیات را یکبار با ASP.Net 1.1 و بار دیگر با ASP.Net 2.0 به بالا انجام دهید، متوجه تفاوت سرعت دریافت قابل تاملی خواهید شد (ASP.Net 2.0 به بعد حدودا تا 10 ثانیه کندتر عمل می‌کند). علت چیست؟
در دات نت 1.1 ، تنظیمات پروکسی پیش فرض در کتابخانه‌های مربوطه وجود نداشت و به نال تنظیم شده بود. در دات نت 2 به بعد این مورد به پروکسی پیش فرض سیستم، تنظیم شده است. پروکسی پیش فرض سیستم همان تنظیماتی است که در internet explorer صورت می‌گیرد.
کاربر پیش فرض ASP.Net (مثلا NETWORK SERVICE) دسترسی خواندن این اطلاعات را از رجیستری ویندوز ندارد. علت این وقفه هم همین مورد است! (حتی اگر برنامه ویندوزی شما هم دسترسی خواندن اطلاعات کلیدهای HKLM رجیستری ویندوز را نداشته باشد، باز هم این مساله رخ خواهد داد)
دات نت فریم ورک سعی می‌کند تا این تنظیمات را از رجیستری یا مکان‌های میسر دیگر بخواند و در آخر پس از شکست کلیه حالات مختلف، کلاینت را به صورت مستقیم متصل خواهد کرد.
خوشبختانه این عملکرد پیش فرض قابل تغییر است. تنها کافی است چند سطر زیر را به فایل config برنامه خود اضافه کنید:

```
<system.net>
    <defaultProxy>
      <proxy bypassonlocal="true" usesystemdefault="false" />
    </defaultProxy>
</system.net>
```

با این‌کار تشخیص خودکار پروکسی سیستم غیرفعال شده و وقفه‌ی معرفی شده دیگر وجود نخواهد داشت.
راه دیگر انجام آن، نسبت دادن نال به خاصیت Proxy شیء HttpWebRequest است که همین اثر را خواهد داشت.

عنوان: **اضافه کردن قابلیت از سرگیری مجدد (resume) به HttpWebRequest**
نویسنده: وحید نصیری
تاریخ: ۰۹:۵۵:۰۰ ۱۳۸۷/۱۲/۱۰
آدرس: www.dotnettips.info
برچسب‌ها: system.net

مطابق RFC مربوطه ، اگر هدر درخواست ارسالی به سرور را کمی تغییر دهیم می‌توان بجای شروع از اولین بایت، از بایت مورد نظر شروع به دریافت فایل نمود. (البته این به شرطی است که سرور آن‌را پشتیبانی کند)

یعنی نیاز داریم که به هدر ارسالی سطر زیر را اضافه کنیم:

```
Range: bytes=n-
```

که n در اینجا حجم فایل ناقص دریافتی موجود بر حسب بایت است.
برای بدست آوردن اندازه‌ی فایل ناقص موجود می‌توان از دستور زیر استفاده کرد:

```
using System.IO;
long brokenLen = new FileInfo(fileNamePath).Length;
```

سپس اگر شیء webRequest ما به صورت زیر تعریف شده باشد:

```
HttpWebRequest webRequest = (HttpWebRequest)WebRequest.Create(url);
```

فقط کافی است سطر زیر را جهت افزودن قابلیت از سرگیری مجدد دریافت فایل به این شیء افزود:

```
دانلود از ادامه//
webRequest.AddRange((int)brokenLen); //resume
```

نکته:
اگر علاقمند باشید که ریز فعالیت‌های انجام شده توسط فضای نام System.Net را ملاحظه کنید، به فایل config خود (مثلا فایل app.config برنامه)، چند سطر زیر را اضافه کنید:

```
<?xml version="1.0" encoding="utf-8" ?>
<configuration>
 <system.diagnostics>
   <trace autoflush="true" />
   <sources>
     <source name="System.Net">
       <listeners>
         <add name="MyTraceFile"/>
       </listeners>
     </source>
   </sources>

   <sharedListeners>
     <add
       name="MyTraceFile"
       type="System.Diagnostics.TextWriterTraceListener"
       initializeData="System.Net.trace.log"
              />
   </sharedListeners>

   <switches>
     <add name="System.Net" value="Verbose" />
```

```
   </switches>

 </system.diagnostics>
</configuration>
```

به این صورت کلیه هدرهای ارسالی به سرور و ریز فعالیت‌های انجام شده در پشت صحنه‌ی کلاس‌های موجود در فایلی به نام System.Net.trace.log برای شما ثبت خواهد شد.

ملاحظات:
بدیهی است پیاده سازی قابلیت resume نیاز به موارد زیر خواهد داشت:
الف) در نظر گرفتن مسیری پیش فرض برای ذخیره سازی فایل‌ها
ب) پیدا کردن اندازه‌ی فایل موجود بر روی یک سرور و مقایسه‌ی آن با حجم فایل موجود بر روی هارد
امکان پیدا کردن اندازه‌ی یک فایل هم بدون دریافت کامل آن میسر است. خاصیت ContentLength مربوط به شیء HttpWebResponse بیانگر اندازه‌ی یک فایل بر روی سرور است و صد البته پیش از استفاده از این عدد، مقدار StatusCode شیء نامبرده را بررسی کنید. اگر مساوی OK بود، یعنی این عدد معتبر است.

نظرات خوانندگان

نویسنده: نیما
تاریخ: ۱۳۸۷/۱۲/۱۱ ۰۰:۱۴:۰۰

سلام استاد نصیری
مقاله جالبی بود.یعنی این نرم افزارهای Download Manager هم از این روش استفاده میکنند؟
یه سری از سایتها هستند که فقط یه بار میشه به این صورت ازشون دانلود کرد.یعنی اگر الان بزنیم تو Queue اون دانلود منیجر وحتی قسمتیش هم دانلود بشه اما اون دانلود رو متوقف کرده و بعد از مثلا یک اعت دوباره Resume کنیم پیغام میده که فایل پیدا نشد.آیا آدرسها تغییر میکنند؟
آیا مثال بالا رو میشه برای گرفتن یه استریم از یک صفحه هم تعمیم داد مثلا نوشتن یک دانلودر برا youtube ؟
ممنون از لطف شما استاد عزیز

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۲/۱۱ ۰۲:۰۲:۰۰

- بله. این یک استاندارد است.
- بسته به سیاست یک سایت ممکن است فایل را پس از مدتی پاک کنند. یک سری از دانلود سنترهای عمومی به همین صورت هستند.
- تمام فایل‌های youtube لینک مستقیم به فایل flv هم دارند. می‌شود به این صورت فایل‌های آن‌ها را دریافت کرد با قابلیت از سرگیری مجدد.

نویسنده: نیما
تاریخ: ۱۳۸۷/۱۲/۱۱ ۱۶:۵۱:۰۰

سلام
جناب نصیری در مورد دوم باید عرض کنم که فایله پاک نمیشه و اگر دوباره بک بار صفحه رو رفرش کنیم دوباره میشه فایل رو دانلود کرد به نظر میرسه فلدرهای حاوی فایلها احتمالا تغییر نام پیدا میکنند.
در مورد سوم که فرمودین آیا ما دسترسی مستقیم به فایل FLV رو داریم؟ کلا در اینگونه سایتها راهی برای پیدا کردنه لینکه مستقیم وجود داره؟
ببخشید من زیاد سوال پرسیدم اما یه مسئله ای به ذهنه من رسید اینکه چطور نمیشه از RapidShare با یوزر رایگان با استفاده از نرم افزارهای مدیریت دانلود چیزی دانلود کرد؟از کجا میفهمه که الان ویندوز داره میگیره یا دانلود مینیجر؟
ممنون از لطف شما

نویسنده: وحید نصیری
تاریخ: ۱۳۸۷/۱۲/۱۱ ۱۸:۴۶:۰۰

- بله. به سورس اسکریپت زیر مراجعه کنید:
http://userscripts.org/scripts/show/39917

- رپیدشیر به کلاینت‌های دانلودی از اکانت پرمیوم استفاده نمی‌کنند (اکانت پرمیوم از basic authentication استفاده میکنه)، اجازه‌ی بازکردن بیشتر از یک ترد را نمی‌دهد. مرورگرها هم عموما فقط از یک ترد برای دانلود استفاده می‌کنند. اگر دانلود منیجر خودتون رو هم به یک ترد محدود کنید مثل مرورگر عمل می‌کند.

عنوان: **دریافت اطلاعات از سایت‌های غیر استاندارد**
نویسنده: وحید نصیری
تاریخ: ۱۳۸۸/۰۴/۲۷ ۱۷:۵۷:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: system.net

اساسا از آنجائیکه ما در یک دنیای کامل زندگی نمی‌کنم و بقولی همه چیزمان باید با همه چیزمان جور دربیاید، ممکن است هنگام استفاده از یک httpWebRequest به خطای زیر برخورد کرده و عملیات دریافت اطلاعات متوقف شود:

```
The server committed a protocol violation. Section=ResponseHeader Detail=CR must be followed by LF
```

و یا حالتی دیگر:

```
The underlying connection was closed: The server committed an HTTP protocol violation.
```

بعضی از وب سروها ممکن است پاسخ ارسالی خود را دقیقا مطابق سطر به سطر RFC های مربوطه ارائه ندهند و کلاس httpWebRequest دات نت هم تعارفی با آن‌ها نداشته و به دلایل امنیتی پردازش پاسخ دریافتی را نیمه کاره رها می‌کند.
برای مثال content-length دقیقا باید به همین شکل ارسال شود و اگر به صورت content length (با یک فاصله در میان کلمات ارسال گردد) به عنوان یک HTTP response split attack در نظر گرفته شده و خطاهای HTTP protocol violation حاصل می‌شوند.

اما می‌توان آگاهانه دات نت فریم ورک را وادار کرد که از این مساله چشم پوشی کند و این نوع سایت‌ها را نیز بررسی و دریافت نماید. برای این منظور در فایل app.config برنامه ویندوزی خود و یا web.config یک برنامه تحت وب، چند سطر زیر را اضافه کنید:

```
<configuration>
 <system.net>
  <settings>
   <httpWebRequest useUnsafeHeaderParsing="true" />
  </settings>
 </system.net>
</configuration>
```

عنوان: **یافتن آدرس نهایی یک Url پس از Redirect**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۰/۰۷/۰۳ ۰۹:۳۹:۰۰
آدرس: www.dotnettips.info
برچسب‌ها: system.net

برای مثال آدرس http://feedproxy.google.com/~r/nettuts/~3/tWCksueANyY/ را در نظر بگیرید. گوگل پس از خرید feedburner ، از feedproxy.google.com جهت ردیابی آدرس‌های فیدها استفاده می‌کند؛ مثلا فید شما چند نفر خواننده دارد، کدام موارد بیشتر خوانده شده، با چه مرورگرهایی، از چه مکان‌هایی و مواردی از این دست. البته می‌توان در تنظیمات فیدبرنر این نوع آدرس دهی را خاموش کرد ولی به صورت پیش فرض فعال است. مشکل feedproxy.google.com هم این است که در ایران فیلتر است. بنابراین یافتن آدرس اصلی این نوع لینک‌ها پس از Redirect نهایی می‌تواند جهت ارائه عمومی آن‌ها مفید باشد. با استفاده از قطعه کد زیر می‌توان این آدرس را یافت:

```
using System;
using System.Net;

namespace Linq2Rss
{
    public class RedirectFinder
    {
        public CookieContainer Cookies { set; get; }
        public string Url { get; set; }

        public string GetRedirectUrl()
        {
            var hops = 1;
            const int MaxRedirects = 20;

            do
            {
                var request = (HttpWebRequest)WebRequest.Create(Url);
                request.UserAgent = "MyUA";
                request.KeepAlive = true;
                request.Referer = Url;
                if (Cookies == null) Cookies = new CookieContainer();
                request.CookieContainer = Cookies;
                request.AllowAutoRedirect = false;

                using (var webResp = request.GetResponse() as HttpWebResponse)
                {
                    if ((webResp.StatusCode == HttpStatusCode.Found) ||
                    (webResp.StatusCode == HttpStatusCode.Redirect) ||
                    (webResp.StatusCode == HttpStatusCode.Moved) ||
                    (webResp.StatusCode == HttpStatusCode.MovedPermanently))
                    {
                        var newLocation = webResp.Headers["Location"];
                        if (newLocation.StartsWith("/"))
                        {
                            var uri = new Uri(Url);
                            Url = string.Format("{0}://{1}:{2}{3}", uri.Scheme, uri.Host, uri.Port,
newLocation);
                        }
                        else
                            Url = newLocation;
                    }
                    else
                    {
                        if (webResp.StatusCode == HttpStatusCode.OK)
                            return Url;
                    }
                }

                hops++;
            } while (hops <= MaxRedirects);
            return Url;
        }
    }
}
```

برای یافتن آدرس واقعی یک Url پس از Redirect راهی بجز درخواست آن از وب سرور اولیه وجود ندارد. سپس وضعیت پاسخ داده شده بررسی می‌شود؛ اگر حاوی Moved ، Found یا Redirect بود، به این معنا است که باید آدرس جدید را از هدر پاسخ دریافتی استخراج کنیم. این آدرس، در کلید Location ذخیره می‌شود. اکنون یکبار دیگر نیاز است تا این آدرس جدید بررسی شود، زیرا ممکن است این مورد هم به آدرس دیگری اشاره کند. در کل، کد فوق 20 بار این بررسی را انجام خواهد داد (هر چند عموما در دو یا سه سعی به جواب خواهیم رسید).

عنوان: **نحوه کار با ftp - بخش اول**
نویسنده: بهمن آبادی
تاریخ: ۱۳۹۲/۰۲/۱۱ ۱۹:۵۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, C#, system.net, ftp

امروز می‌خوام نحوه کار با FTP بصورت ساده برای کاربران و برنامه نویسان مبتدی رو آموزش بدم.

برای استفاده از FTP نیاز به یک اکانت FTP در سایت مورد نظر بهمراه دسترسی به پوشه ای مشخص می‌باشد.
برای مثال ما یک اکانت FTP در سایت dotnettip.info داریم که به پوشه upload دسترسی داره.

ابتدا در فایل Web.config و در بین تگ های appSettings مقادیر زیر را برای دسترسی به اکانت و نام کاربری و رمز عبور ذخیره می‌کنیم.

```
<add key="FtpAddress" value="ftp://ftp.dotnetips.info" />
<add key="FtpUser" value="uploadcenter" />
<add key="FtpPass" value="123123" />
<add key="FolderPath" value="~/Upload/" />
```

*نکته : برای امنیت بیشتر و دسترسی به اطلاعات اکانت می‌شود از روش‌های دیگری نیز استفاده کرد.

در ادامه یک کلاس در App_code پروژه خود با نام FTPHelper ایجاد می‌کنیم و کد زیر را در آن قرار می‌دهیم:
تکه کد بالا برای ست کردن مقادیر نام کاربری و رمز عبور و آدرس FTP در کلاس مذکور که بصورت پیش‌فرض از web.config پر می‌شود ایجاد و بکار خواهد رفت.

```
using System.Net;
using System.IO;
using System.Configuration;


public class FtpHelper
{
public FtpHelper()
{
       //Default Value Set From Application
         _hostname = ConfigurationManager.AppSettings.GetValues("FtpAddress")[0];
        _username = ConfigurationManager.AppSettings.GetValues("FtpUser")[0];
        _password = ConfigurationManager.AppSettings.GetValues("FtpPass")[0];
}

    #region "Properties"
    private string _hostname;
    /// <summary>
    /// Hostname
    /// </summary>
    /// <value></value>
    /// <remarks>Hostname can be in either the full URL format
    /// ftp://ftp.myhost.com or just ftp.myhost.com
    /// </remarks>
    public string Hostname
    {
        get
        {
            if (_hostname.StartsWith("ftp://"))
            {
                return _hostname;
            }
            else
            {
                return "ftp://" + _hostname;
            }
        }
        set
        {
            _hostname = value;
        }
    }
    private string _username;
```

```
    /// <summary>
    /// Username property
    /// </summary>
    /// <value></value>
    /// <remarks>Can be left blank, in which case 'anonymous' is returned</remarks>
    public string Username
    {
        get
        {
            return (_username == "" ? "anonymous" : _username);
        }
        set
        {
            _username = value;
        }
    }
    private string _password;
    public string Password
    {
        get
        {
            return _password;
        }
        set
        {
            _password = value;
        }
    }


    #endregion
}
```

سپس فضای نام‌های زیر را در کلاس خود قرار می‌دهیم.

```
using System.Net;
using System.IO;
```

حالا برای بارگذاری فایل می‌توانیم از یک تابع بصورت shared استفاده کنیم که بتوان با دادن آدرس فایل بصورت فیزیکی به تابع و مشخص کردن پوشه مورد نظر آنرا در هاست مقصد (FTP) بارگذاری کرد.توجه داشته باشید که تابع فوق نیازی به قرار گرفتن در کلاس بالا (FtpHelper) ندارد.یعنی می‌توان آنرا در هرجای برنامه پیاده سازی نمود.

```
public static bool Upload(string fileUrl)
        {
            if (File.Exists(fileUrl))
            {
                FtpHelper ftpClient = new FtpHelper();
                string ftpUrl = ftpClient.Hostname + System.IO.Path.GetFileName(fileUrl);


                FtpWebRequest ftp = (FtpWebRequest)FtpWebRequest.Create(ftpUrl);
                ftp.Credentials = new NetworkCredential(ftpClient.Username, ftpClient.Password);

                ftp.KeepAlive = true;
                ftp.UseBinary = true;
                ftp.Timeout = 3600000;
                ftp.KeepAlive = true;
                ftp.Method = WebRequestMethods.Ftp.UploadFile;

                const int bufferLength = 102400;
                byte[] buffer = new byte[bufferLength];
                int readBytes = 0;

                //open file for reading
                using (FileStream fs = File.OpenRead(fileUrl))
                {
                    try
                    {
                        //open request to send
                        using (Stream rs = ftp.GetRequestStream())
```

```
                    {
                        do
                        {
                            readBytes = fs.Read(buffer, 0, bufferLength);
                            fs.Write(buffer, 0, readBytes);
                        } while (!(readBytes < bufferLength));
                        rs.Close();
                    }

                }
                catch (Exception)
                {
                 //Optional Alert for Exeption To Application Layer
                   //throw (new ApplicationException("بارگذاری فایل با خطا  رو به رو شد"));

                }
                finally
                {
                    //ensure file closed
                    //fs.Close();
                }

            }

            ftp = null;
            return true;
        }
        return false;


    }
```

تکه کد بالا فایل مورد نظر را در صورت وجود به صورت تکه‌های 100 کیلوبایتی بر روی ftp بارگذاری می‌کند، که می‌توانید مقدار آنرا نیز تغییر دهید.
اینکار باعث افزایش سرعت بارگذاری در فایل‌های با حجم بالا برای بارگذاری می‌شود.

در بخش‌های بعدی نحوه ایجاد پوشه ، حذف فایل ، حذف پوشه و دانلود فایل از روی FTP را بررسی خواهیم کرد.

نظرات خوانندگان

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۲/۱۱ ۲۲:۳۸

ممنون از مطلب شما.
چند نکته جزئی در مورد کدهای تهیه شده:
- وجود این try/catch در اینجا هیچ هدفی رو برآورده نکرده. از قسمت throw ex هم توصیه می‌شود که استفاده نکنید. از thow خالی استفاده کنید تا stack trace پاک نشه. به علاوه زمانیکه مشغول به طراحی یک کتابخانه هستید تا حد ممکن از ذکر try/catch خودداری کنید. وظیفه بررسی این مسایل مرتبط است به لایه‌های بالاتر استفاده کننده و نه کتابخانه پایه.
- if ابتدای متد هم ضرورتی ندارد. اگر قرار است باشد، باید به استفاده کننده در طی یک استثناء اعلام شود که چرا فایل درخواستی او آپلود نشده. در کل استفاده از متد File.Exists به همراه صدور استثناء در صورت عدم وجود فایل، در اینجا مناسب‌تر است.
- نامگذاری‌هایی مانند obj_ftp مربوط به دوران C است. در سی‌شارپ روش دیگری رو باید استفاده کنید که در مطلب اصول نامگذاری در دات نت به تفصیل بررسی شده.
- بررسی صفر بودن readBytes بهتر است پیش از فراخوانی متد Write انجام شود.
- یک سری از اشیاء در دات نت پیاده سازی کننده IDispoable هستند. به این معنا که بهتر است از using برای استفاده از آن‌ها کمک گرفته شود تا کامپایلر قسمت finally به همراه آزاد سازی منابع را به صورت خودکار اضافه کند. این نکته برای مواردی که در بین کار استثنایی رخ می‌دهد جهت آزاد سازی منابع لازم است. یعنی بهتر بود بجای try/catch از try/finally و یا using در مکان‌های مناسب استفاده می‌شد.
- علت استفاده از شیء Application دراینجا چه چیزی بوده؟ AppSettings خوانده شده از وب کانفیگ برنامه و کل اطلاعات آن در آغاز به کار یک برنامه ASP.NET به صورت خودکار کش می‌شوند. به همین جهت است که اگر حتی یک نقطه در فایل وب کانفیگ تغییر کند برنامه ASP.NET ری استارت می‌شود (تا دوباره تنظیمات را بخواند). بنابراین مستقیما از همان امکانات ConfigurationManager بدون انتساب آن به شیء سراسری Application استفاده کنید. اینکار سربارای آنچنانی هم ندارد؛ چون از حافظه خوانده می‌شود و نه از فایل. هر بار فراخوانی ConfigurationManager.AppSettings به معنای مراجعه به فایل web.config نیست. فقط بار اول اینکار انجام می‌شود؛ تا زمانیکه این فایل تغییر کند یا برنامه ری استارت شود.

نویسنده: بهمن آبادی
تاریخ: ۱۳۹۲/۰۲/۱۲ ۳:۳۰

با تشکر از نظر شما. تمام صحبت‌های شما منطقی است. اولا من این بخش رو می‌خواستم بعدا تکمیل کنم ولی بدلایلی زودتر ارسال کردم.

مواردی هم که اعلام کزدید کاملا صحیح می‌باشد.فقط بخش تابعی که درون آن از try/catch استفاده شده رو میخواستم ابتدا در تیکه کد دیگری از یک صفحه aspx بنویسم که وقتی برای بررسی مجدد تابع پیدا نکردم وگرنه اونجا try/catch بلا استفاده است.

موارد متذکر شده برای سهولت بیشتر کاربران در کدهای فوق لحاظ و ویرایش گردید.
با تشکر.

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۱۲ ۱۱:۴

```
.locals init ([0] class [mscorlib]System.IO.TextWriter w)
  IL_0000:  ldstr      "log.txt"
  IL_0005:  call       class [mscorlib]System.IO.StreamWriter
      [mscorlib]System.IO.File::CreateText(string)
  IL_000a:  stloc.0

 .try
  {
    IL_000b:  ldloc.0
```

```
  IL_000c:  ldstr      "This is line one"
  IL_0011:  callvirt   instance void [mscorlib]
    System.IO.TextWriter::WriteLine(string)
  IL_0016:  leave.s    IL_0022
} // end .try
finally
{
  IL_0018:  ldloc.0
  IL_0019:  brfalse.s  IL_0021
  IL_001b:  ldloc.0
  IL_001c:  callvirt   instance void [mscorlib]
    System.IDisposable::Dispose()
  IL_0021:  endfinally
} // end handler
```

ماخذ

زمانیکه از using statement استفاده می‌کنید، خود کامپایلر try/finally رو اضافه می‌کنه. یعنی قسمت close الان بهش نیازی نیست چون استریم مورد استفاده حتما در اینجا dispose میشه.

نویسنده: بهمن آبادی
تاریخ: ۱۳۹۲/۰۲/۱۲ ۱۲:۵۰

از دستم در رفته ...شما ببخشید.

بیشتر هدفم روند بارگذاری و تکنیک بارگذاری اون بود ولی خوب درست کد نوشتن کاملاً صحیحه و من می‌خواستم برای کاربرای آماتور بنویسم ولی مجبورم کردین که کاربرای حرفه ای هم مثل شما به کد ایرادی نباشه.

بازم ممنون از دقت و راهنماییتون .

حالا بهتر شد نسبت به قبلیه ؟

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۲/۱۲ ۱۷:۲۸

من فکر می‌کنم اگر ReSharper رو نصب کنید، پیش از ارائه یک مطلب یا پروژه خیلی از ایرادات رو با خط کشیدن زیر اون یا نمایش یک علامت زرد کنار صفحه گوشزد می‌کنه. مثلا می‌گه که این نوع نامگذاری درست نیست یا این شیء رو میشه با using محصور کرد. خلاصه از دستش ندید، حیفه!

نویسنده: سمیرا قادری
تاریخ: ۱۳۹۲/۰۶/۱۳ ۱۲:۳۷

با تشکر از مطلب خوبتان
ببخشید این روش کار کردن با Ftp چه مزایایی دارد و چه کاربردی دارد؟ چون من تا حالا فقط از Cute Ftp استفاده کردم آن هم برای Upload سایت بعد از Publish

نویسنده: محسن خان
تاریخ: ۱۳۹۲/۰۶/۱۳ ۱۲:۵۱

یه خورده وقت بذاری، باهاش می‌تونی یک Cute Ftp بنویسی.

نویسنده: رضا منصوری
تاریخ: ۱۳۹۲/۱۲/۲۰ ۱۵:۵۰

با تشکر از مطلب خوبتون ،
برای اینکه به طور مستقیم از file.SaveAs استفاده کنیم و فایل را مستقیم به Ftp آپلود کنیم (نه اینکه از فضای ذخیره شده ای کپی کنیم) چه پیشنهادی می‌دهید ؟ مثلا ما هاست دانلود داریم و میخواهیم فایل‌های ارسالی را در هاست دانلود ذخیره کنیم.

عنوان: **راه‌های کم کردن احتمال اسپم شدن ایمیل‌های ارسالی توسط SMTP Client**
نویسنده: حسین پناهلو
تاریخ: ۱۳۹۲/۰۶/۱۷ ۱۷:۵۰
آدرس: www.dotnettips.info
گروه‌ها: system.net, C#.NET, internet, Email, SMTP

1. فرستادن ایمیل‌ها با وقفه زمانی
2. نفرستادن پشت سر ایمیل‌ها به یک host خاص
3. استفاده نکردن از کلمه هایی که احتمال اسپم شناخته شدن ایمیل را افزایش می‌دهند در قسمت Subject Email

در لینک‌های زیر لیست بعضی از این کلمات را می‌توانید مشاهده کنید:

http://blog.hubspot.com/blog/tabid/6307/bid/30684/The-Ultimate-List-of-Email-SPAM-Trigger-Words.aspx

http://www.inmotionhosting.com/support/edu/everything-email/spam-prevention-techniques/common-spam-words

4. فعال نکردن high priority

با فعال شدن این گزینه ایمیل شما مورد بررسی‌های بیشتری قرار می‌گیرد و شانس اسپم شناخته شدن آنرا افزایش می‌دهد.

```
mailMessage.Priority = MailPriority.High ; //not good
```

5. Set کردن Encoding صحیح

```
mailMessage.BodyEncoding = System.Text.Encoding.GetEncoding(“utf-8”);
```

6. استفاده از حداکثر سه image در متن ایمیل
7. اضافه کردن هم htmlview و هم plainview به view ایمیل ارسالی

با مثال نحوه این کار را نشان می‌دهم:

```
System.Net.Mail.AlternateView plainView = System.Net.Mail.AlternateView.CreateAlternateViewFromString
System.Text.RegularExpressions.Regex.Replace(BodyText, @”<(.|\n)*?>”, string.Empty), null,
“text/plain”);
System.Net.Mail.AlternateView htmlView =
System.Net.Mail.AlternateView.CreateAlternateViewFromString(BodyText, null, “text/html”);
mailMsg.AlternateViews.Add(plainView);
mailMsg.AlternateViews.Add(htmlView);
```

منابع:

http://stackoverflow.com/questions/5042309/email-messages-going-to-spam-folder

/http://www.andreas-kraus.net/blog/tips-for-avoiding-spam-filters-with-systemnetmail

نظرات خوانندگان

نویسنده: حمید حسین وند
تاریخ: ۱۳۹۳/۰۱/۱۴ ۱۱:۴۷

سلام
لطفا کدهای فوق رو به صورت زیر اصلاح کنید. هنگام استفاده مشکل دارند.

```
AlternateView plainView = AlternateView.CreateAlternateViewFromString(
                        Regex.Replace(EmailBody, @"<(.|\n)*?>", string.Empty), null, "text/plain");
AlternateView htmlView = AlternateView.CreateAlternateViewFromString(EmailBody, null, "text/html");
MyMsg.AlternateViews.Add(plainView);
MyMsg.AlternateViews.Add(htmlView);
```

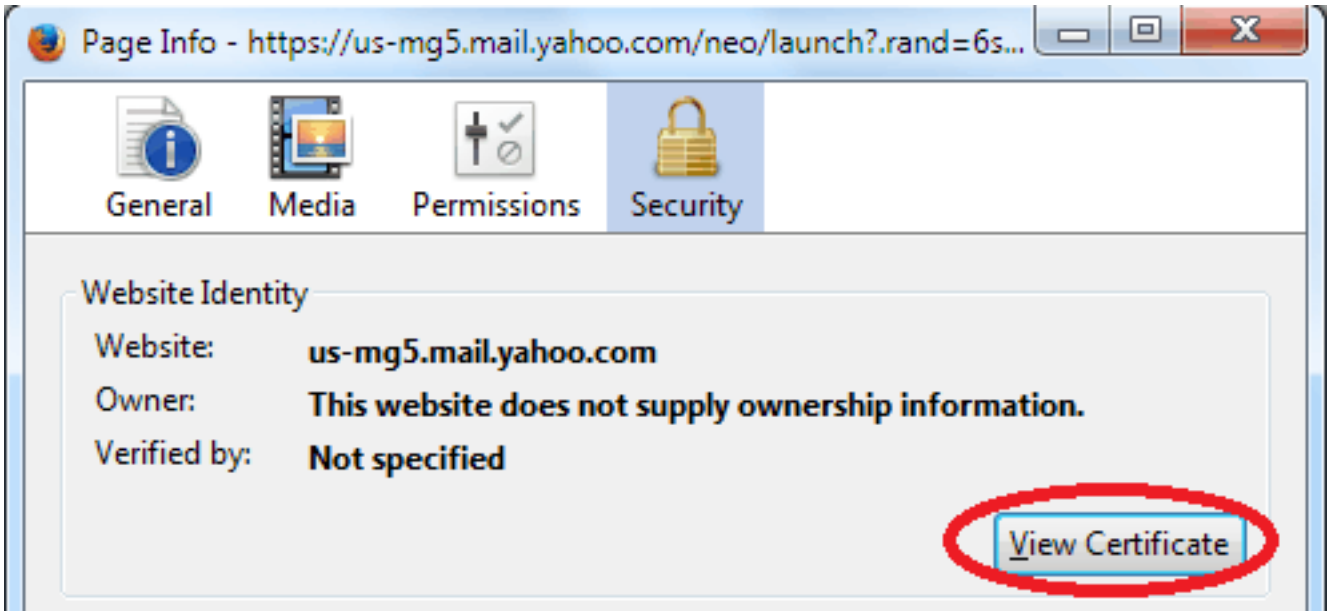
البته من namespace هارو در بالای کلاس تعریف کردم و از اینجا حذف کردم.

```
using System.Net.Mail;
using System.Text.RegularExpressions;
```

با تشکر

عنوان: دانلود مجوز SSL یک سایت HTTPS
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۷/۲۱ ۰:۵۵
آدرس: www.dotnettips.info
گروه‌ها: system.net, Cryptography, https, SSL

اگر به مرورگرها دقت کرده باشید، امکان نمایش SSL Server Certificate یک سایت استفاده کننده از پروتکل HTTPS را دارند. برای مثال در فایرفاکس اگر به خواص یک صفحه مراجعه کنیم، در برگه امنیت آن، امکان مشاهده جزئیات مجوز SSL سایت جاری فراهم است:



سؤال: چگونه می‌توان این مجوزها را با کدنویسی دریافت یا تعیین اعتبار کرد؟

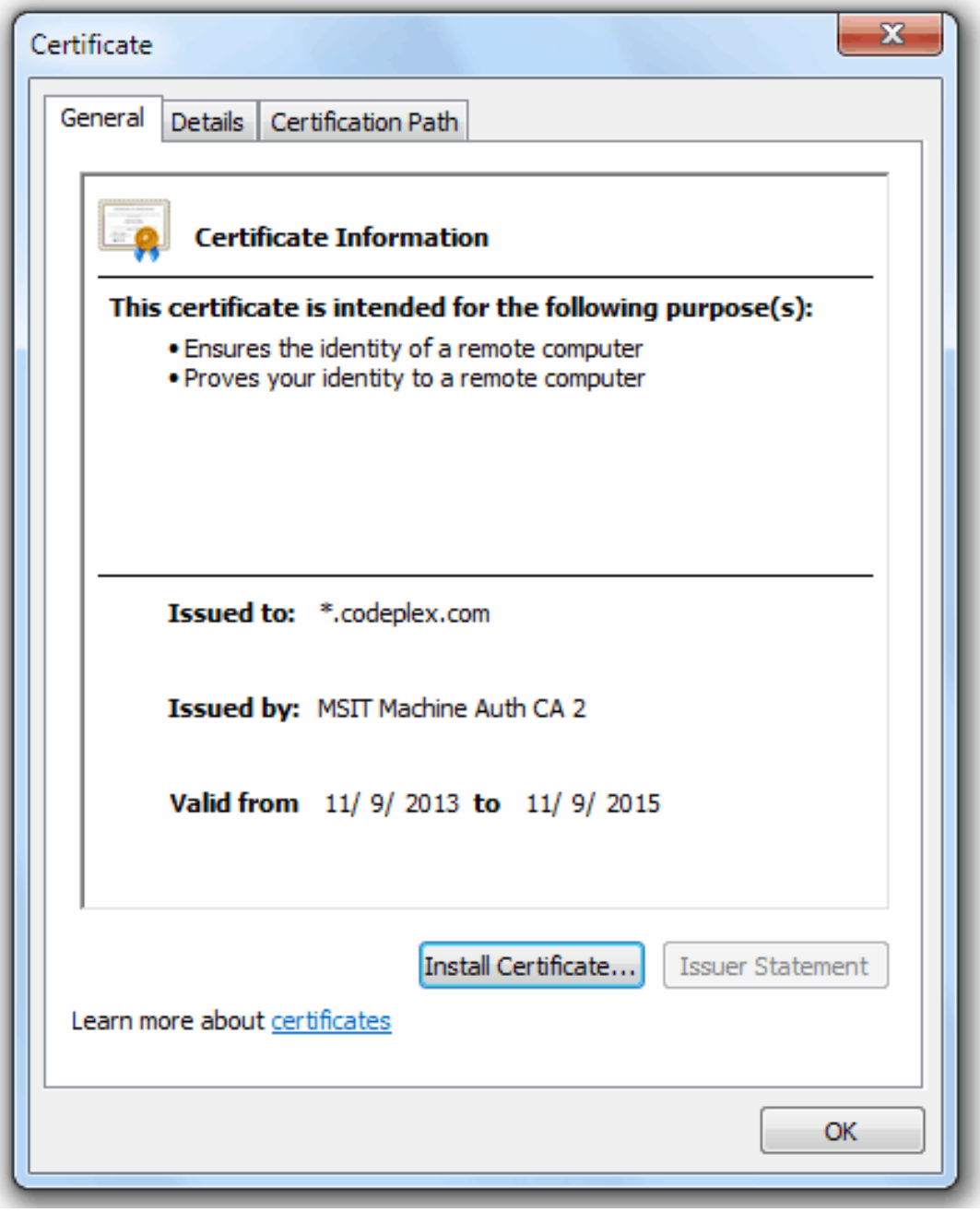
قطعه کد زیر، نحوه دریافت مجوز SSL یک سایت را نمایش می‌دهد:

```
using System;
using System.Diagnostics;
using System.IO;
using System.Net;
using System.Security.Cryptography.X509Certificates;

namespace DownloadCerts
{
    class Program
    {
        static void Main(string[] args)
        {
            // صرفنظر از خطاهای احتمالی مجوز
            ServicePointManager.ServerCertificateValidationCallback = delegate { return true; };

            var url = "https://pdfreport.codeplex.com";
            var request = WebRequest.Create(url) as HttpWebRequest;
            request.Method = WebRequestMethods.Http.Head;
            using (var response = request.GetResponse())
            { /* در اینجا مجوز، در صورت وجود دریافت شده */  }

            if (request.ServicePoint.Certificate == null)
                return;

            // ذخیره سازی مجوز در فایل
            var cert = new X509Certificate2(request.ServicePoint.Certificate);
            Console.WriteLine("Expiration Date: {0}", cert.GetExpirationDateString());
            var data = cert.Export(X509ContentType.Cert);
```

```
            File.WriteAllBytes("site.cer", data);
            Process.Start(Environment.CurrentDirectory);
        }
    }
}
```

ممکن است مجوز یک سایت معتبر نباشد. کلاس WebRequest در حین مواجه شدن با یک چنین سایت‌هایی، یک WebException را صادر می‌کند. از این جهت که می‌خواهیم حتما این مجوز را دریافت کنیم، بنابراین در ابتدای کار، ServerCertificateValidation را غیرفعال می‌کنیم.
سپس یک درخواست ساده را به آدرس سرور مورد نظر ارسال می‌کنیم. پس از پایان درخواست، خاصیت request.ServicePoint.Certificate با مجوز SSL یک سایت مقدار دهی شده است. در ادامه نحوه ذخیره سازی این مجوز را با فرمت cer مشاهده می‌کنید.

Name	Date modified	Type
DownloadCerts.exe	۲۱ مهر ۱۳۹۲ ۱۲:۲۷ ق.ظ	Application
DownloadCerts.pdb	۲۱ مهر ۱۳۹۲ ۱۲:۲۷ ق.ظ	PDB File
DownloadCerts.vshost.exe	۲۱ مهر ۱۳۹۲ ۱۲:۲۹ ق.ظ	Application
DownloadCerts.vshost.exe.manifest	۲۶ اسفند ۱۳۸۸ ۰۹:۳۹ ب.ظ	MANIFEST File
site.cer	۲۱ مهر ۱۳۹۲ ۱۲:۲۹ ق.ظ	Security Certificate



نظرات خوانندگان

نویسنده: حمید حسین وند
تاریخ: ۱۳۹۳/۰۱/۲۲ ۱۶:۲۸

سلام؛ وقتی این گواهی یا certificate رو دانلود کردیم به چه دردمون میخوره؟ یعنی کارااییش برای ما چیه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۲ ۱۶:۵۸

جهت بررسی اعتبار آن می‌تواند مفید باشد. مثلا نوشتن برنامه‌ای مانند SSL Certificate Verifier