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.