عنوان: **انجام کارهای زمانبندی شده در برنامه‌های ASP.NET توسط DNT Scheduler**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۲۶ ۱۱:۴۰
آدرس: www.dotnettips.info
گروه‌ها: ASP.Net, MVC, Schedule

اگر به دو مطلب استفاده از Quartz.Net (^ و ^) و خصوصا نظرات آن دقت کرده باشید به این نتیجه خواهید رسید که ... این کتابخانه‌ی در اصل جاوایی گنگ طراحی شده‌است. در سایت جاری برای انجام کارهای زمانبندی شده (مانند ارسال ایمیل‌های روزانه خلاصه مطالب، تهیه خروجی PDF و XML سایت، تبدیل پیش نویس‌ها به مطالب، بازسازی ایندکس‌های جستجو و امثال آن) از یک Thread timer استفاده می‌شود که حجم نهایی کتابخانه‌ی محصور کننده و مدیریت کننده‌ی وظایف آن جمعا 8 کیلوبایت است؛ متشکل از ... سه کلاس. در ادامه کدهای کامل و نحوه‌ی استفاده از آن را بررسی خواهیم کرد.

دریافت کتابخانه DNT Scheduler و مثال آن

DNTScheduler.7z
در این بسته، کدهای کتابخانه‌ی DNT Scheduler و یک مثال وب فرم را، ملاحظه خواهید کرد. از این جهت که برای ثبت وظایف این کتابخانه، از فایل global.asax.cs استفاده می‌شود، اهمیتی ندارد که پروژه‌ی شما وب فرم است یا MVC. با هر دو حالت کار می‌کند.

نحوه‌ی تعریف یک وظیفه‌ی جدید

کار با تعریف یک کلاس و پیاده سازی ScheduledTaskTemplate شروع می‌شود:

```
public class SendEmailsTask : ScheduledTaskTemplate
```

برای نمونه :

```
using System;

namespace DNTScheduler.TestWebApplication.WebTasks
{
    public class SendEmailsTask : ScheduledTaskTemplate
    {
        /// <summary>
        /// اگر چند جاب در یک زمان مشخص داشتید، این خاصیت ترتیب اجرای آن‌ها را مشخص خواهد کرد
        /// </summary>
        public override int Order
        {
            get { return 1; }
        }

        public override bool RunAt(DateTime utcNow)
        {
            if (this.IsShuttingDown || this.Pause)
                return false;

            var now = utcNow.AddHours(3.5);
            return now.Minute % 2 == 0 && now.Second == 1;
        }

        public override void Run()
        {
            if (this.IsShuttingDown || this.Pause)
                return;

            System.Diagnostics.Trace.WriteLine("Running Send Emails");
        }

        public override string Name
        {
            get { return "ارسال ایمیل"; }
        }
    }
}
```

- در اینجا Order، ترتیب اجرای وظیفه‌ی جاری را در مقایسه با سایر وظیفه‌هایی که قرار است در یک زمان مشخص اجرا شوند، مشخص می‌کند.
- متد RunAt ثانیه‌ای یکبار فراخوانی می‌شود (بنابراین بررسی now.Second را فراموش نکنید). زمان ارسالی به آن UTC است و اگر برای نمونه می‌خواهید بر اساس ساعت ایران کار کنید باید 3.5 ساعت به آن اضافه نمائید. این مساله برای سرورهایی که خارج از ایران قرار دارند مهم است. چون زمان محلی آن‌ها برای تصمیم گیری در مورد زمان اجرای کارها مفید نیست.
در متد RunAt فرصت خواهید داشت تا منطق زمان اجرای وظیفه‌ی جاری را مشخص کنید. برای نمونه در مثال فوق، این وظیفه هر دو دقیقه یکبار اجرا می‌شود. یا اگر خواستید اجرای آن فقط در سال 23 و 33 دقیقه هر روز باشد، تعریف آن به نحو ذیل خواهد بود:

```
public override bool RunAt(DateTime utcNow)
        {
            if (this.IsShuttingDown || this.Pause)
                return false;

            var now = utcNow.AddHours(3.5);
            return now.Hour == 23 && now.Minute == 33 && now.Second == 1;
        }
```

- خاصیت IsShuttingDown موجود در کلاس پایه ScheduledTaskTemplate، توسط کتابخانه‌ی DNT Scheduler مقدار دهی می‌شود. این کتابخانه قادر است زمان خاموش شدن پروسه‌ی فعلی IIS را تشخیص داده و خاصیت IsShuttingDown را true کند. بنابراین در حین اجرای وظیفه‌ای مشخص، به مقدار IsShuttingDown دقت داشته باشید. اگر true شد، یعنی فقط 30 ثانیه وقت دارید تا کار را تمام کنید.
خاصیت Pause هر وظیفه را برنامه می‌تواند تغییر دهد. به این ترتیب در مورد توقف یا ادامه‌ی یک وظیفه می‌توان تصمیم گیری کرد. خاصیت ScheduledTasksCoordinator.Current.ScheduledTasks، لیست وظایف تعریف شده را در اختیار شما قرار می‌دهد.
- در متد Run، منطق وظیفه‌ی تعریف شده را باید مشخص کرد. برای مثال ارسال ایمیل یا تهیه‌ی بک آپ.
- Name نیز نام وظیفه‌ی جاری است که می‌تواند در گزارشات مفید باشد.

همین مقدار برای تعریف یک وظیفه کافی است.

نحوه‌ی ثبت و راه اندازی وظایف تعریف شده

پس از اینکه چند وظیفه را تعریف کردیم، برای مدیریت بهتر آن‌ها می‌توان یک کلاس ثبت و معرفی کلی را مثلا به نام ScheduledTasksRegistry ایجاد کرد:

```
using System;
using System.Net;

namespace DNTScheduler.TestWebApplication.WebTasks
{
    public static class ScheduledTasksRegistry
    {
        public static void Init()
        {
            ScheduledTasksCoordinator.Current.AddScheduledTasks(
                new SendEmailsTask(),
                new DoBackupTask());

            ScheduledTasksCoordinator.Current.OnUnexpectedException = (exception, scheduledTask) =>
            {
                //todo: log the exception.
                System.Diagnostics.Trace.WriteLine(scheduledTask.Name + ":" + exception.Message);
            };

            ScheduledTasksCoordinator.Current.Start();
        }

        public static void End()
        {
```

```
            ScheduledTasksCoordinator.Current.Dispose();
        }

        public static void WakeUp(string pageUrl)
        {
            try
            {
                using (var client = new WebClient())
                {
                    client.Credentials = CredentialCache.DefaultNetworkCredentials;
                    client.Headers.Add("User-Agent", "ScheduledTasks 1.0");
                    client.DownloadData(pageUrl);
                }
            }
            catch (Exception ex)
            {
                //todo: log ex
                System.Diagnostics.Trace.WriteLine(ex.Message);
            }
        }
    }
}
```

- شیء ScheduledTasksCoordinator.Current، نمایانگر تنها وهله‌ی مدیریت وظایف برنامه است.
- توسط متد ScheduledTasksCoordinator.Current.AddScheduledTasks، تنها کافی است کلاس‌های وظایف مشتق شده از ScheduledTaskTemplate، معرفی شوند.
- به کمک متد ScheduledTasksCoordinator.Current.Start، کار Thread timer برنامه شروع می‌شود.
- اگر در حین اجرای متد Run، استثنایی رخ دهد، آن‌را توسط یک Action delegate به نام ScheduledTasksCoordinator.Current.OnUnexpectedException می‌توانید دریافت کنید. کتابخانه‌ی DNT Scheduler برای اجرای وظایف، از یک ترد با سطح تقدم Below normal استفاده می‌کند تا در حین اجرای وظایف، برنامه‌ی جاری با اخلال و کندی مواجه نشده و بتواند به درخواست‌های رسیده پاسخ دهد. در این بین اگر استثنایی رخ دهد، می‌تواند کل پروسه‌ی IIS را خاموش کند. به همین جهت این کتابخانه کار try/catch استثناهای متد Run را نیز انجام می‌دهد تا از این لحاظ مشکلی نباشد.
- متد ScheduledTasksCoordinator.Current.Dispose کار مدیر وظایف برنامه را خاتمه می‌دهد.
- از متد WakeUp تعریف شده می‌توان برای بیدار کردن مجدد برنامه استفاده کرد.

استفاده از کلاس ScheduledTasksRegistry تعریف شده

پس از اینکه کلاس ScheduledTasksRegistry را تعریف کردیم، نیاز است آن‌را به فایل استاندارد global.asax.cs برنامه به نحو ذیل معرفی کنیم:

```
using System;
using System.Configuration;
using DNTScheduler.TestWebApplication.WebTasks;

namespace DNTScheduler.TestWebApplication
{
    public class Global : System.Web.HttpApplication
    {
        protected void Application_Start(object sender, EventArgs e)
        {
            ScheduledTasksRegistry.Init();
        }

        protected void Application_End()
        {
            ScheduledTasksRegistry.End();
            //نکته مهم این روش نیاز به سرویس پینگ سایت برای زنده نگه داشتن آن است
            ScheduledTasksRegistry.WakeUp(ConfigurationManager.AppSettings["SiteRootUrl"]);
        }
    }
}
```

- متد ScheduledTasksRegistry.Init در حین آغاز برنامه فراخوانی می‌شود.
- متد ScheduledTasksRegistry.End در پایان کار برنامه جهت پاکسازی منابع باید فراخوانی گردد.

همچنین در اینجا با فراخوانی ScheduledTasksRegistry.WakeUp، می‌توانید برنامه را مجددا زنده کنید! IIS مجاز است یک سایت ASP.NET را پس از مثلا 20 دقیقه عدم فعالیت (فعالیت به معنای درخواست‌های رسیده به سایت است و نه کارهای پس زمینه)، از حافظه خارج کند (این عدد در application pool برنامه قابل تنظیم است). در اینجا در فایل web.config برنامه می‌توانید آدرس یکی از صفحات سایت را برای فراخوانی مجدد تعریف کنید:

```
<?xml version="1.0"?>
<configuration>
  <appSettings>
      <add key="SiteRootUrl" value="http://localhost:10189/Default.aspx" />
  </appSettings>
</configuration>
```

همینکه درخواست مجددی به این صفحه برسد، مجددا برنامه توسط IIS بارگذاری شده و اجرا می‌گردد. به این ترتیب وظایف تعریف شده، در طول یک روز بدون مشکل کار خواهند کرد.

گزارشگیری از وظایف تعریف شده

برای دسترسی به کلیه وظایف تعریف شده، از خاصیت ScheduledTasksCoordinator.Current.ScheduledTasks استفاده نمائید:

```
var jobsList = ScheduledTasksCoordinator.Current.ScheduledTasks.Select(x => new
{
   TaskName = x.Name,
   LastRunTime = x.LastRun,
   LastRunWasSuccessful = x.IsLastRunSuccessful,
   IsPaused = x.Pause,
}).ToList();
```

لیست حاصل را به سادگی می‌توان در یک Grid نمایش داد.

نظرات خوانندگان

نویسنده: محمد رعیت پیشه
تاریخ: ۱۳۹۲/۱۲/۲۶ ۱۹:۲۲

آیا یکی از کاربردهای این مطلب می‌تونه مثلا ارسال یک ایمیل یک هفته قبل از اتمام زمان شارژ کاربری باشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۲۶ ۱۹:۳۷

بله. می‌توانید یک وظیفه‌ی جدید تعریف کنید که هر شب ساعت مثلا 11 و 15 دقیقه اجرا شود (نحوه‌ی تعریف متد RunAt). سپس در متد Run آن یک کوئری از دیتابیس گرفته، لیست موارد مدنظر را واکشی کرده و به آن‌ها ایمیل بزنید.

نویسنده: پیمان مهربانی
تاریخ: ۱۳۹۲/۱۲/۲۷ ۱۱:۴۷

کتابخانه quartz scheduler سبکی بود اما نگرانی هایی مانند پیش آمدن همزمانی در اجرای وظایف را دارم.پیش از این از کتابخانه Scheduler خود ویندوز و یا Jonهای SQL Server هم بهره برد.

این روش چه مزایا و معایبی نسبت به روش‌های موجود دارد و آیا توصیه شده برای استفاده در پروژه‌های بزرگ هست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۲۷ ۱۲:۲

- «پیش آمدن همزمانی در اجرای وظایف»
خاصیت Order را برای وظایفی که قرار است در یک زمان مشخص اجرا شوند، مقدار دهی کنید. 1و 2 و 3 و الی آخر.
- «حتی می‌توان از Scheduler خود ویندوز و یا Jonهای SQL Server هم بهره برد».
بله. به شرطی‌که سرور در اختیار شما باشد و دسترسی کافی برای انجام اینکار را داشته باشید. البته در این حالت خاص، مدیریت آن یکپارچه با یک برنامه‌ی وب نیست.
در سرورهای اشتراکی روش ارائه شده در این مطلب بدون نیاز به سطح دسترسی خاصی کار می‌کند. ضمنا برای ASP.NET نوشته شده است و این قابلیت را دارد که به شما اعلام کند مثلا تا 30 ثانیه دیگر برنامه از سرور unload می‌شود؛ توسط خاصیت IsShuttingDown. همچنین حق تقدم ترد آن طوری تنظیم شده که سبب اختلال در عملیات و عملکرد متداول سایت نشود.
- «آیا توصیه شده برای استفاده در پروژه‌های بزرگ هست؟»
یک به اشتراک گذاری بود از قسمتی از کدهای زیر ساخت سایت جاری که هم اکنون مورد استفاده است (مقدمه بحث).

نویسنده: سلمان کاظمی
تاریخ: ۱۳۹۲/۱۲/۲۸ ۱:۳۲

سوالی که واسه من پیش اومده اینه که من یک نرم افزار Web Form دارم . هدفم اینه که یک روز قبل از تاریخ تولد اعضا ایمیلی با عنوان تولدتان مبارک ارسال شه .حالا من باید این کدهای گفته شده را در برنامه خود بیاورم یا یک وب سرویس بنویسم که اینکارو انجام بده؟اگه بخوام که در برنامه انجام بشه خوب کجا باید نوشت ؟ یا به عبارتی دیگه باید حتما برنامه اجرا بشه و اگه برنامه اصلا Run نشه چی میشه؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۱۲/۲۸ ۸:۵۹

- وب سرویس فقط با یک درخواست رسیده کار می‌کند. کار کتابخانه‌ی فوق، اجرا در پس زمینه‌ی برنامه به صورت مداوم است.
- فقط دو حالت وجود دارد که برنامه اجرا نشود:

الف) protected void Application_End فراخوانی شود. متد WakeUp نوشته شده برای این منظور و راه اندازی مجدد برنامه توسط آن، کافی است.
ب) کل سرور ری استارت شود (نه فقط برنامه). در این حالت کافی است آدرس برنامه را به یکی از سرویس‌هایی که هر از چندگاهی برنامه را ping می‌کنند، معرفی کنید.

نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۳/۰۱/۰۱ ۲۳:۲۳

خیلی خیلی ممنون ... کتابخانه‌ی ساده و مفیدی است.
سوال من اینه که میشه کاری کرد که آدرس روت تعریف شده در فایل کانفیگ جهت بیدار شدن IIS را در کد نوشت مثل کد زیر :

```
var urlToWakeup = Request.Url.Scheme + Uri.SchemeDelimiter + Request.Url.Host +
            (Request.Url.IsDefaultPort ? "" : ":" + Request.Url.Port);
```

که مجبور نباشی در هر سایتی که طراحی میکنیم آدرس رو در فایل کانفیگ ست کنیم !
پی نوشت : من این کار رو در متد Application_End نوشتم و خطای در دسترس نبودن Request رو دریافت کردم ! (هر چند که میدونستم این خطا رو میده !)

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۰۲ ۰:۵۱

می‌شود در Application_BeginRequest اطلاعات آدرس ریشه سایت را در یک متغیر استاتیک ذخیره کرد. مقدار آن در Application_End قابل استفاده است:

```
namespace TestApp
{
    public static class App
    {
        public static string SiteRootUrl;
    }

    public class TestApplication : HttpApplication
    {
        protected void Application_BeginRequest(object sender, EventArgs e)
        {
            if (string.IsNullOrWhiteSpace(App.SiteRootUrl))
            {
                App.SiteRootUrl = Request.Url.GetLeftPart(UriPartial.Authority) +
Request.ApplicationPath;
            }
        }

        protected void Application_End()
        {
            // use App.SiteRootUrl
        }
    }
}
```

نویسنده: امیرحسین جلوداری
تاریخ: ۱۳۹۳/۰۱/۰۲ ۱۳:۲

خیلی خیلی ممنون :) ... اگه میشه نسخه‌ی nuget کتابخانه را هم بذارید

نویسنده: مهدی پایروند
تاریخ: ۱۳۹۳/۰۱/۲۵ ۱۴:۳۱

1.برای زنده نگهداشتن سایت میشه از خود همین کتابخانه استفاده کرد؟
2.برای چه تعداد جاب بنظرتون منطقیه که استفاده بشه؟
ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۵ ۱۶:۴۳

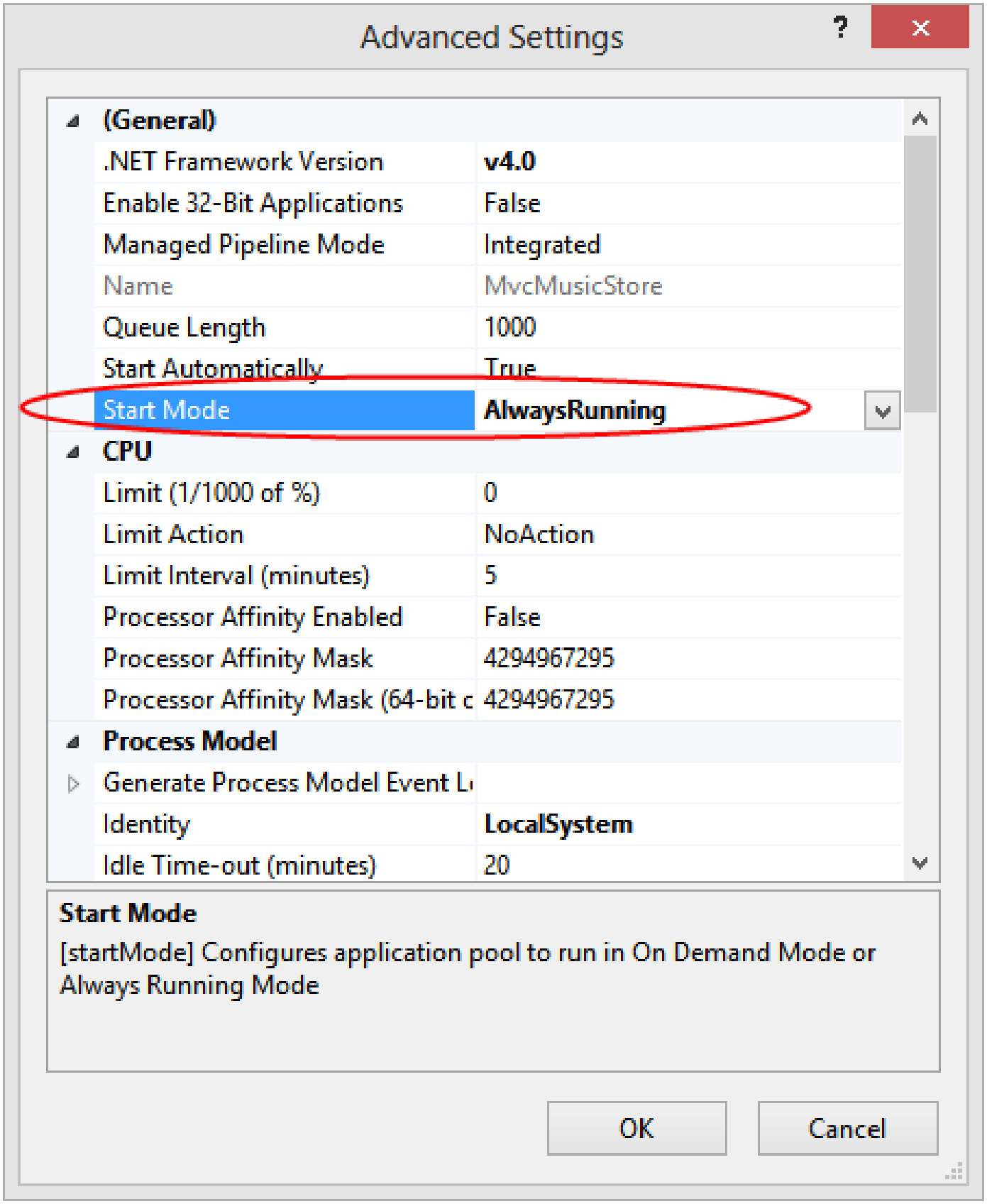
- کمی بالاتر توضیح دادم « فقط دو حالت وجود دارد که برنامه اجرا نشود: ... »
- این مورد فقط بستگی به توان سرور شما دارد.

نویسنده: fss
تاریخ: ۱۳۹۳/۰۱/۲۸ ۹:۶

در حالتی که "کل سرور ری استارت شود (نه فقط برنامه) " ، بعد از بالا آمدن سرور و شروع به کار IIS، تابع Application_Start کار شروع برنامه را خود به خود انجام نمی‌دهد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۱/۲۸ ۹:۱۴

نه تا زمانیکه اولین درخواستی به برنامه برسد.
کل سرور ری استارت شده. IIS برنامه را فقط زمانی مجددا بارگذاری می‌کند که درخواست نمایش یکی از قسمت‌های سایت به آن ارسال شود.
البته IISهای جدید قابلیت Auto-Start هم دارند؛ ولی باید در تنظیمات Application pool برنامه انتخاب شود:



همچنین Application Initialization Module نیز برای اجرا خودکار برنامه پس از ری‌استارت سرور طراحی شده .

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۱۷ ۹:۶

یک نکته‌ی تکمیلی

در دات نت *4.5.2* ، متدی به نام HostingEnvironment.QueueBackgroundWorkItem اضافه شده‌است تا درخواست اجرای کارهای پس زمینه در ASP.NET به سادگی و همچنین با اطمینان بیشتری قابل انجام باشد.

نویسنده: مهرداد
تاریخ: ۱۳۹۳/۰۲/۱۷ ۱۳:۴۰

یعنی با استفاده از این متد جدیدی که در دات نت 4.5.2 اضافه شده ، دیگه نیازی به Quartz.net و یا DNT scheduler نیست ؟ و میتوان برای کارهایی که نیاز به زمانبندی دارند از این متد استفاده کرد ؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۱۷ ۱۳:۴۲

فقط قسمت مدیریت تردهای آن با یک سطر QueueBackgroundWorkItem جایگزین می‌شود. مابقی قسمت‌های آن (مانند اینکه یک WorkItem چه زمانی در Queue قرار گیرد) تفاوتی نمی‌کند و مانند قبل است.

نویسنده: مهرداد
تاریخ: ۱۳۹۳/۰۲/۱۷ ۱۴:۱۰

منظور شما این خط هست ؟

```
ScheduledTasksCoordinator.Current.AddScheduledTasks(
                new SendEmailsTask(),
                new DoBackupTask());
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۱۷ ۱۴:۲۳

این‌ها باقی خواهند ماند. قسمت new Thread و مدیریت آن جایگزین می‌شود.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۲/۲۹ ۹:۲۵

یک نکته

مثالی از نحوه‌ی استفاده از متد جدید HostingEnvironment.QueueBackgroundWorkItem
+ یک مثال رسمی

نویسنده: Aria
تاریخ: ۱۳۹۳/۰۳/۱۹ ۲۳:۱۰

با سلام
من روی یه پروژه مالی دارم با ASP WebForm کار میکنم و میخواستم در اول هر ماه یک سری دستورات ویرایش یا update را که به صورت رکورد در یک جدول ذخیره می‌کنم انجام بشن. سوالم اینه که برای این که مجموعه دستورات من انجام بشن باید برنامه جتما Run باشه و چه جوری میشه اونو به صورت پس زمینه در حال اجرا نگه داشت.
با تشکر

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۲۰ ۰:۱۹

- به اندازه کافی در نظرات این بحث در مورد زنده نگه داشتن یک برنامه ASP.NET بحث شده. کمی وقت بگذارید و آن‌ها را مطالعه کنید.
+ اگر برنامه مالی است، احتمالا دسترسی کاملی به سرور و همچنین SQL Server (اگر با آن کار می‌کنید) دارید. در این حالت برای به روز رسانی زمانبندی شده‌ی چند رکورد شاید بهتر باشد از سرویس معروف و همیشه در حال اجرای SQL Server agent استفاده کنید. در اینجا نیز می‌شود یک job را که متشکل از دستورات T-SQL است، در فواصل زمانی مشخصی اجرا کرد.

نویسنده: میثم
تاریخ: ۱۳۹۳/۰۳/۲۵ ۱۳:۴۸

سلام.
منظورتون فقط همین یک خط توی متد Start کلاس ScheduledTasksCoordinator بود که تغییر میکنه؟
یعنی فقط متد Start به شکل زیر بازنویسی میشه دیگه؟

```
public void Start()
        {
            _timer.OnTimerCallback = () =>
            {
                var now = DateTime.UtcNow;
                var taskToRun = _tasks.Where(x => !x.IsRunning && x.RunAt(now)).OrderBy(x =>
x.Order).ToList();
                if (_isShuttingDown || !taskToRun.Any())
                    return;

                HostingEnvironment.QueueBackgroundWorkItem(x => taskAction(taskToRun));
            };
        }
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۳/۲۵ ۱۵:۱

- بله. قسمت‌های HostingEnvironment.RegisterObject و IRegisteredObject آن هم باید حذف شوند چون در QueueBackgroundWorkItem وجود دارند و یک CancellationToken را تنظیم می‌کند.
+ زمانیکه از DNTScheduler استفاده می‌کنید، عملا نیازی به QueueBackgroundWorkItem ندارید. چون نکته‌ی HostingEnvironment.RegisterObject و IRegisteredObject در آن لحاظ شده. این نکته که خاموش شدن IIS را گزارش می‌کند، چند سال قبل، توسط یکی از اعضای قبلی تیم ASP.NET منتشر شده بود . دقیقا از همین نکته در QueueBackgroundWorkItem استفاده شده.

به صورت خلاصه، DNTScheduler با دات نت 4 به بعد سازگار است و نکات QueueBackgroundWorkItem دات نت 4.5.2 را به صورت توکار پیاده سازی کرده‌است.

نویسنده: حسین
تاریخ: ۱۳۹۳/۰۴/۱۷ ۱۱:۲

سلام. اگر بخواهیم یک کار نسبتاً زمانبر که IO هم هست را توسط این کتابخانه در فواصل زمانی معین اجرا کنیم، میشه از async و await استفاده کرد؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۴/۱۷ ۱۱:۲۰

بله. از روش و کتابخانه معرفی شده در مطلب « استفاده از async و await در برنامه‌های کنسول و سرویس‌های ویندوز NT » استفاده کنید.

نویسنده: حمید حسین وند
تاریخ: ۱۳۹۳/۰۵/۰۹ ۱۰:۱۹

سلام
من برای این سری کارها از ویندوز سرویس استفاده می‌کنم. مثلا ویندوز سرویس من از ساعت 8 صبح شروع به کار می‌کنه و رویدادهایی مثل سالروز تولد رو با استفاده از پیامک به کاربران پیام تبریک ارسال می‌کنه.
مهمترین عاملی که باعث شد من از ویندوز سرویس استفاده کنم اجرای مداوم وهمیشگی بدون ارسال درخواست به وب سایت من بود. ولی فکر می‌کنم این کتابخانه شما هم مثل ویندوز سرویس عمل می‌کنه و خودش همیشه در حال اجراست.
حالا به نظرتون آیا از ویندوز سرویس استفاده کنم بهتره و یا اینکه از این کتابخانه استفاده کنم؟
ممنون

نویسنده: محسن خان
تاریخ: ۱۳۹۳/۰۵/۰۹ ۱۱:۳

فرض کن داری از یک هاست اشتراکی استفاده می‌کنی. دسترسی ادمین هم روی سرور نداری برای اجرا سرویس ویندوز یا فرض کن در یک سازمان بهت گفتن ما فقط اجازه می‌دیم فایل‌های سایتت رو روی سرور کپی کنی. دسترسی بیشتری بهت نمی‌دیم. اون وقت چکار می‌کنی؟

نویسنده: Mohammad
تاریخ: ۱۳۹۳/۰۵/۱۵ ۲:۵۶

سلام. ممنون.

دوستان برای من وقتی به صورت ساعت می‌دم درست کار نمیکنه.مثلا من می‌خوام در ساعت دو و بیست و چهار دقیقه بامداد یه کاری رو انجام بده اینجوری نوشتم:در صورتی اگه بگم دو دقیقه دو دقیقه درست انجام میشه.به نظرتون مشکل از کجاست؟ممنون

```
var now = utcNow.AddHours(3.5);
    return now.Hour == 2 && now.Minute == 24 && now.Second == 1;
```

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۵/۱۵ ۸:۴۶

مشکل از تنظیم نبودن ساعت سرور است. مقدار DateTime.UtcNow را روی سرور بررسی کنید و با مقدار واقعی تطابق بدید. بعد اختلافش را باید در همینجا اعمال کنید. مثلا اگر پس از بررسی متوجه شدید ساعت سرور یک ساعت عقب هست، now.Hour == 2 می‌شود now.Hour == 1 و امثال این نوع محاسبات.

نویسنده: علی
تاریخ: ۱۳۹۳/۰۶/۱۵ ۱۸:۳۴

ممنون. مفید بود. اگه بخواهیم یه کاری مثلا هر سه روز یکبار انجام بشه باید چه جوری زمان رو تعیین کنیم؟ ممنون

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۶/۱۵ ۱۹:۵۷

```
public override bool RunAt(DateTime utcNow)
        {
            if (this.IsShuttingDown || this.Pause)
                return false;

            var now = utcNow.AddHours(3.5);
            return (now.Day % 3 == 0) && (now.Hour == 0 && now.Minute == 1 && now.Second == 1);
        }
```

نویسنده: مهدی
تاریخ: ۱۳۹۳/۰۶/۱۷ ۲۲:۲۲

یه سوال:اختلاف زمانی ساعت گرینویچ با تهران همیشه 3:30 هست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۶/۱۷ ۲۳:۸

اگر سرور شما تنظیمات daylight saving time صحیحی داشته باشد، بله. اگر نه، خودتان این یک ساعت تفاوت را 6 ماه یکبار باید محاسبه و اعمال کنید.

نویسنده: آرام
تاریخ: ۱۳۹۳/۰۶/۱۸ ۹:۵۷

سلام.
چرا هیچ ارجاعی به متد Stop در کلاس JobsRunnerTimer وجود نداره؟ برای این کار دلیلی دارید؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۳/۰۶/۱۸ ۱۰:۱۴

نیازی نیست. چون طول عمر کل این ماژول دقیقا معادل طول عمر برنامه‌ی وب است. خاتمه‌ی آن هم به صورت خودکار با از حافظه خارج کردن AppDomain برنامه توسط IIS انجام می‌شود. تا زمانیکه برنامه در حال اجرا است این ماژول هم به همین ترتیب. هر زمان که IIS تصمیم به خاتمه‌ی برنامه گرفت، نه این ماژول، هیچ ماژول دیگری هم فرصت مقاومت پیدا نمی‌کند و راسا به همراه AppDomain جاری خاتمه می‌یابد.