عنوان: **تبدیل برنامه‌های کنسول ویندوز به سرویس ویندوز ان تی**
نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۲۵ ۱۶:۰
آدرس: www.dotnettips.info
گروه‌ها: Windows Service

در ویژوال استودیو، قالب پروژه ایجاد سرویس‌های ویندوز ان تی از پیش تدارک دیده شده است؛ اما کار کردن با آن ساده نیست به علاوه امکان دیباگ این نوع سرویس‌ها نیز به صورت پیش فرض درنظر گرفته نشده است و نیاز به تمهیدات و نکات خاصی دارد. جهت سهولت ایجاد سرویس‌های ویندوز ان تی، کتابخانه‌ای به نام TopShelf ایجاد شده است که یک برنامه ویندوزی را به سادگی تبدیل به یک سرویس ویندوز ان تی می‌کند. در ادامه جزئیات نحوه استفاده از آن‌را مرور خواهیم کرد.

الف) دریافت TopShelf

TopShelf یک کتابخانه سورس باز است و علاوه بر آن، آخرین فایل‌های باینری آن‌را از طریق نیوگت نیز می‌توان دریافت کرد:

```
PM> Install-Package Topshelf
```

ب) فعال سازی TopShelf

یک برنامه ساده کنسول را ایجاد کنید. سپس با استفاده از نیوگت و اجرای فرمان فوق، ارجاعی را به اسمبلی TopShelf اضافه نمائید.

```
using Topshelf;

namespace MyService
{
    class Program
    {
        static void Main(string[] args)
        {
            HostFactory.Run(config =>
            {
                config.Service(settings => new TestService());
                config.EnableServiceRecovery(recovery => recovery.RestartService(delayInMinutes: 1));
                config.EnableShutdown();
                config.EnablePauseAndContinue();
                config.SetDescription("MyService Desc.");
                config.SetDisplayName("MyService");
                config.SetServiceName("MyService");
                config.RunAsLocalSystem();
            });
        }
    }
}
```

کدهای آغازین کار با TopShelf همین چندسطر فوق هستند. در آن وهله‌ای از کلاس سرویس مشتق شده از ServiceControl را دریافت کرده و سپس نام سرویس و سطح دسترسی اجرای آن مشخص می‌شوند. EnableServiceRecovery مربوط به حالتی است که سرویس کرش کرده است و ویندوز این قابلیت را دارد تا یک سرویس را به صورت خودکار راه اندازی مجدد کنند.

```
using Topshelf;
using Topshelf.Logging;

namespace MyService
{
    public class TestService : ServiceControl
    {
        static readonly LogWriter _log = HostLogger.Get<TestService>();

        public bool Start(HostControl hostControl)
        {
            _log.Info("TestService Starting...");

            return true;
```

```
        }

        public bool Stop(HostControl hostControl)
        {
            _log.Info("TestService Stopped");

            return true;
        }

        public bool Pause(HostControl hostControl)
        {
            _log.Info("TestService Paused");

            return true;
        }

        public bool Continue(HostControl hostControl)
        {
            _log.Info("TestService Continued");

            return true;
        }
    }
}
```

در اینجا امضای کلی کلاس سرویس را مشاهده می‌کند که می‌تواند شامل چهار متد استاندارد آغاز، پایان، مکث و ادامه باشد.

ج) نصب TopShelf

در همین حالت اگر برنامه را اجرا کنید، سرویس ویندوز ان تی تهیه شده، شروع به کار خواهد کرد (مزیت مهم آن نسبت به قالب توکار تهیه سرویس‌های ویندوز در ویژوال استودیو).
برای نصب این سرویس تنها کافی است در خط فرمان با دسترسی مدیریتی، دستور نصب `your_exe install` و یا عزل `your_exe uninstall` صادر شوند.

نظرات خوانندگان

نویسنده: محسن جمشیدی
تاریخ: ۱۳۹۲/۰۸/۲۶ ۷:۵۱

نسخه سرور Quartz.Net هم از همین کامپوننت برای ساختن ویندوز سرویس استفاده می‌کند

نویسنده: Hamid NCH
تاریخ: ۱۳۹۲/۰۸/۲۶ ۱۲:۸

خیلی ممنون از مطلب خوبتون.
ممکنه یه مثال کوچیک عملی از ویندوز سرویس ارائه بفرمایید.

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۲۶ ۱۲:۳۹

من برای شرکتی چندسال قبل یک چنین برنامه‌ای رو تهیه کردم:
یک سرویس ویندوز ان تی که روزهای اول به پورت سریال دستگاه کارتخوان متصل می‌شد و بعدها این پورت تبدیل شد به پورت شبکه و ریموت هم می‌شد به آن وصل شد؛ اما باز هم این پورت شبکه تبدیلگری بود برفراز سیستم اصلی RS232 آن دستگاه. کار این سرویس که نیاز به دسترسی بالایی برای اتصال به پورت‌های سیستم داشت و همچنین همیشه در حال اجرا بود و راه اندازی آن نیازی به لاگین شخصی و اجرای دستی برنامه نداشت (مزیت سرویس‌های ویندوز ان تی)، اتصال هر 5 دقیقه یکبار به دستگاه کارتخوان، تخلیه اطلاعات آن و ثبت آن‌ها در بانک اطلاعاتی یک برنامه ASP.NET بود. به این ترتیب کارکنان آن سازمان می‌توانستند از ورود و خروج خودشان با یک برنامه تحت وب (در شبکه داخلی شرکت) گزارشگیری کنند.

نویسنده: سعید ستوده
تاریخ: ۱۳۹۲/۰۸/۲۶ ۱۹:۲۶

سلام؛ برای سایت mvc که نوشتیم و روی هاست آپلود کردیم اگه بخواهیم سرویسی داشته باشیم که هر یک ساعت یکبار یا بازه کمتر یا بیشتر از داخل دیتابیس یک چیزی را چک کنه و اس ام اسی ارسال کنه به نظر شما چه راهکارهایی وجود داره و کدوم بهتره البته با توجه به اینکه ما سرور اختصاصی نداریم و تنها یک هاست معمولیست؟

نویسنده: وحید نصیری
تاریخ: ۱۳۹۲/۰۸/۲۶ ۲۰:۵۴

« اجرای وظایف زمان بندی شده با Quartz.NET - قسمت اول »

عنوان: **هاست سرویس های Asp.Net Web Api با استفاده از OWIN و TopShelf**
نویسنده: مسعود پاکدل
تاریخ: ۱۳۹۳/۰۲/۲۷ ۲۳:۱۵
آدرس: www.dotnettips.info
گروه‌ها: ASP.NET Web API, Self-Hosting, Windows Service, OWIN

زمانیکه از Templateهای پیش فرض تدارک دیده شده در VS.Net برای اپلیکیشن‌های وب خود استفاده می‌کنید، وب اپلیکیشن و سرور با هم یکپارچه هستند و تحت IIS اجرا می‌شوند. به وسیله Owin می‌توان این دو مورد را بدون وابستگی به IIS به صورت مجزا اجرا کرد. در این پست قصد داریم سرویس‌های Web Api را در قالب یک Windows Service با استفاده از کتابخانه‌ی TopShelf هاست نماییم.

پیش نیاز ها:

» Owin چیست

» تبدیل برنامه‌های کنسول ویندوز به سرویس ویندوز ان تی

برای شروع یک برنامه Console Application ایجاد کرده و اقدام به نصب پکیج‌های زیر نمایید:

```
Install-Package Microsoft.AspNet.WebApi.OwinSelfHost
Install-Package TopShelf
```

حال یک کلاس Startup برای پیاده سازی Configurationهای مورد نیاز ایجاد می‌کنیم

در این قسمت می‌توانید تنظیمات زیر را پیاده سازی نمایید:

»سیستم Routing؛

»تنظیم Dependency Resolver برای تزریق وابستگی کنترلرهای Web Api؛

»تنظیمات hubهای SignalR(در حال حاضر SignalR به صورت پیش فرض نیاز به Owin برای اجرا دارد)؛

»رجیستر کردن Owin Middlewareهای نوشته شده؛

»تغییر در Asp.Net PipeLine؛

»و...

```
public class Startup
    {
        public void Configuration(IAppBuilder appBuilder)
        {
            HttpConfiguration config = new HttpConfiguration();
            config.Routes.MapHttpRoute(
                name: "DefaultApi",
                routeTemplate: "api/{controller}/{id}",
                defaults: new { id = RouteParameter.Optional }
            );

            appBuilder.UseWebApi(config);
        }
    }
```

* به صورت پیش فرض نام این کلاس باید Startup و نام متد آن نیز باید Configuration باشد.

در این مرحله یک کنترلر Api به صورت زیر به پروژه اضافه نمایید:

```
public class ValuesController : ApiController
    {
        public IEnumerable<string> Get()
        {
            return new string[] { "value1", "value2" };
        }

        public string Get(int id)
        {
```

```
            return "value";
        }

        public void Post([FromBody]string value)
        {
        }

        public void Put(int id, [FromBody]string value)
        {
        }
    }
```

کلاسی به نام ServiceHost ایجاد نمایید و کدهای زیر را در آن کپی کنید:

```
public class ServiceHost
    {
        private IDisposable webApp;

        public static string BaseAddress
        {
            get
            {
                return "http://localhost:8000/";
            }
        }

        public void Start()
        {
            webApp = WebApp.Start<Startup>(BaseAddress);
        }

        public void Stop()
        {
            webApp.Dispose();
        }
    }
```

واضح است که متد Start در کلاس بالا با استفاده از متد Start کلاس WebApp، سرویس‌های Web Api را در آدرس مورد نظر هاست خواهد کرد. با فراخوانی متد Stop این سرویس‌ها نیز dispose خواهند شد.
در مرحله آخر باید شروع و توقف سرویس‌ها را تحت کنترل کلاس HostFactory کتابخانه TopShelf در آوریم. برای این کار کافیست کلاسی به نام ServiceHostFactory ایجاد کرده و کدهای زیر را در آن کپی نمایید:

```
public class ServiceHostFactory
    {
        public static void Run()
        {
            HostFactory.Run( config =>
            {
                config.SetServiceName( "ApiServices" );
                config.SetDisplayName( "Api Services ]" );
                config.SetDescription( "No Description" );

                config.RunAsLocalService();

                config.Service<ServiceHost>( cfg =>
                {
                    cfg.ConstructUsing( builder => new ServiceHost() );

                    cfg.WhenStarted( service => service.Start() );
                    cfg.WhenStopped( service => service.Stop());
                } );
            } );
        }
    }
```

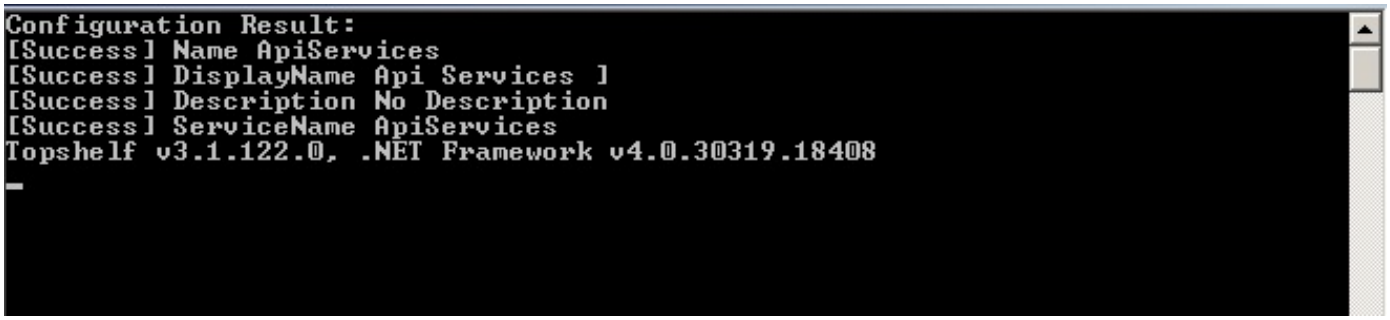
توضیح کدهای بالا:

ابتدا با فراخوانی متد Run سرویس مورد نظر اجرا خواهد شد. تنظیمات نام سرویس و نام مورد نظر جهت نمایش و همچنین توضیحات در این قسمت انجام می‌گیرد.

با استفاده از متد ConstructUsing عملیات وهله سازی از سرویس انجام خواهد گرفت. در پایان نیز متد Start و Stop کلاس ServiceHost، به عنوان عملیات شروع و پایان سرویس ویندوز مورد نظر تعیین شد.

حال اگر در فایل Program پروژه، دستور زیر را فراخوانی کرده و برنامه را ایجاد کنید خروجی زیر قابل مشاهده است.

```
ServiceHostFactory.Run();
```



در حالیکه سرویس مورد نظر در حال اجراست، Browser را گشوده و آدرس http://localhost:8000/api/values/get را در AddressBar وارد کنید. خروجی زیر را مشاهده خواهید کرد: