Scheduled jobs made easy – Topshelf and Quartz.NET

I often work on applications whose sole task is to execute a script at certain time or day. You may approach it in several ways, eg.: Task Scheduler (Windows), SQL Job (if it is a SQL task) or CRON in Linux. You may also write application, which would run in background and execute a script at certain time. The only question is – do you really want do it?

In this post, I would like to introduce other solution to tasks like this. The combination of two frameworks: **Topshelf** (Windows host) and **Quartz.NET** (free-for-business company Task Scheduler).

Topshelf

Topshelf is a framework that makes it easy to launch Windows services written in .NET. Thanks to this, a developer working on a Windows Service can focus solely on building business logic instead of complex configuration service.

Quartz.NET

Quartz.NET is a fully functional framework for creating tasks in time. It is written from scratch in .NET based on a popular framework written in Java – Quartz.

How do they work together?

To connect Topshelf and Quartz.NET you've to get packages first:

```
install-package Topshelf
install-package Topshelf.Quartz
```

job. Simple hosting console app looks like this:

```
1.  namespace mpustelak.TopShelfAndQuartz
2.  {
3.      class Program
4.      {
5.          static void Main(string[] args)
6.          {
7.              HostFactory.Run(x =>
8.              {
9.                  x.Service<MyService>(s =>
10.                 {
11.                     s.WhenStarted(service => service.OnStart
12.                     s.WhenStopped(service => service.OnStop(
13.                     s.ConstructUsing(() => new MyService());
14.
15.                     s.ScheduleQuartzJob(q =>
16.                         q.WithJob(() =>
17.                             JobBuilder.Create<MyJob>().Build
18.                             .AddTrigger(() => TriggerBuilder
19.                                 .WithSimpleSchedule(b => b
20.                                     .WithIntervalInSeconds(1
21.                                     .RepeatForever())
22.                                 .Build()));
23.                 });
24.
25.                 x.RunAsLocalSystem()
26.                     .DependsOnEventLog()
27.                     .StartAutomatically()
28.                     .EnableServiceRecovery(rc => rc.RestartS
29.
30.                 x.SetServiceName("My Topshelf Service");
31.                 x.SetDisplayName("My Topshelf Service");
32.                 x.SetDescription("My Topshelf Service's desc
33.             });
34.         }
35.     }
36.
37.     public class MyService
38.     {
39.         public void OnStart()
40.         {
41.         }
42.
43.         public void OnStop()
44.         {
```

```
        public class MyJob : IJob
        {
            public void Execute(IJobExecutionContext context)
            {
                Console.WriteLine($"[{DateTime.UtcNow}] Welcome
            }
        }
    }
```

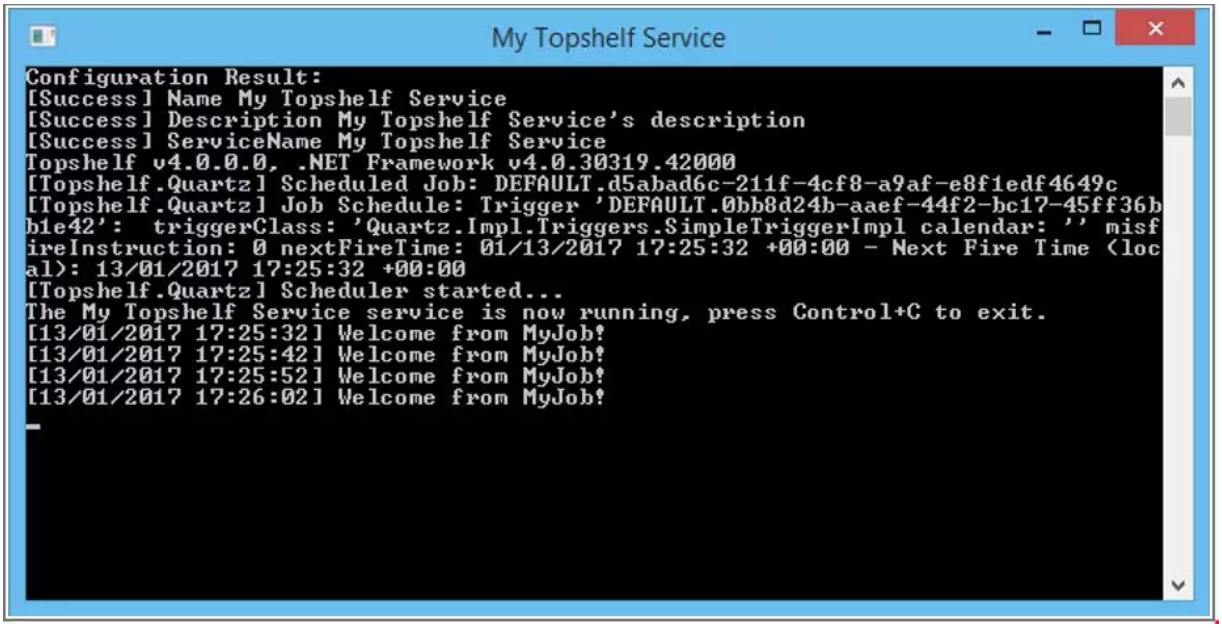
First step is to execute *HostFactory.Run(...)* code which is responsible for hosting the Windows service (a console application).

Second step is to define how our service should behave. In this case I called it *MyService*. As an addition we may set up what should happen when the service starts and stops (*OnStart* and *OnStop* methods).

The next step is to initialize which constructor Quartz should be using. In this case it's the default constructor. At this point, there is also the possibility to transfer that dependency onto the inversion of control container, and use i.e. *ConstructUsingNinject* for NInject (for more info check i.e. Topshelf.Ninject or Topshelf.StructureMap Nuget packages).

The last step is to define job scheduler's execution. In presented example, it will run every 10 seconds (*WithIntervalInSeconds(10)*), without stopping (*RepeatForever*). *MyJob* implementation of *IJob* interface shows what's going to happen every 10 seconds. It' display current time and "*Welcome from MyJob*" text.

Check below how it looks like in the console once started:



Topshelf and Quartz.NET allows us, developers, to rapidly build small (or even large) Windows services to execute scheduled jobs in .NET. As shown in this post, developers don't need to understand the complexity of setting up Windows services and installing them on the machine via InstallUtil.

For more info about Topshelf and Quartz.NET, please reffer to documentation: Topshelf and Quartz.NET.

3

PUBLISHED BY

Mateusz Pustelak
Software Developer with several years of commercial experience, TDD practitioner, DDD/CQRS fan. Currently working for Universal Music Group in London.

16th January 2017 Mateusz Pustelak Good Practices CSharp, DotNet, Micro Service, QuartzNET, Topshelf, Windows Service

5 thoughts on "Scheduled jobs made easy – Topshelf and Quartz.NET"

Paweł

20th January 2017 at 8:30 am

You could also check Hangfire (http://hangfire.io/). Quite nice, it has its own server and a web dashboard.

Mateusz Pustelak

20th January 2017 at 8:43 am

Thanks for info. I've heard about it but never had a chance to use it on production.

MarkusR

loop gracefully?

Anthony

31st January 2018 at 7:17 am

What happens when the Job takes longer than the scheduled time.
Example: in your sample code the job takes 1 min to execute and its scheduled to run after every 1O secs

---

Clarke

31st October 2018 at 11:31 am

I have to voice my passion for your kindness giving support to those people that should have guidance on this important matter.
