# .Net Core Windows Service with Topself and Nlog

This is a project sample of a .Net Core Console application that can run and install as windows service with all new features in .net core like DI, Logging etc using TopSelf for automatic schedule support.

## TopSelf

Topself is use to initializes HostFactory Like Below

var services = ServiceDependency.ConfigureServices();

var serviceProvider = services.BuildServiceProvider();

var rc = HostFactory.Run(x =>

{

x.Service<Startable>(s =>

{

s.ConstructUsing(name => serviceProvider.GetService<Startable>());

s.WhenStarted(tc => tc.Start());

s.WhenStopped(tc => tc.Stop());

});

x.RunAsLocalSystem();

x.SetDescription("Windows Service Demo");

x.SetDisplayName("WinServiceDemo");

x.SetServiceName("WinServiceDemo");

});

var exitCode = (int)Convert.ChangeType(rc, rc.GetTypeCode());

Environment.ExitCode = exitCode;

## Quartz

Quart Library is used to Schedule services like below

public JobTrigger GetCommonJob()

{

ReportProcessorJobKey = JobKey.Create("CommonJob", "CommonJob");

var job = JobBuilder.Create<CommonJob>().WithIdentity("CommonJob", "CommonJob").Build();

int ConfigFrequency;

int frequency = int.TryParse(ConfigurationManager.AppSettings["CommonJobFrequencyInSecond"], out ConfigFrequency) ? ConfigFrequency : 5;

var jobTrigger = TriggerBuilder.Create()

.WithSimpleSchedule(x => x.WithIntervalInSeconds(frequency).RepeatForever())

.Build();

return new JobTrigger { Job = job, Trigger = jobTrigger };

}

## For DI & Logging please refer the source code