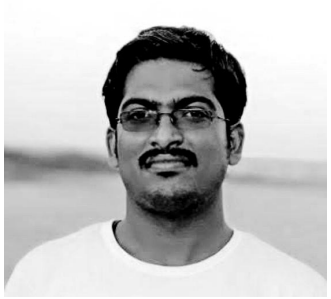
Background Tasks



Sowndarrajan Jayapal

https://www.linkedin.com/in/sowndarrajan/

Examples

- Clean up database or file system every day
- Perform some CPU intensive work asynchronously
- Process messages from a queue every X minutes
- Refresh cache every X minutes
- Send payslip to employees every month
- Send performance reports to stakeholders every day

Agenda

- IHostedService
- BackgroundService
- WorkerService
- Hangfire
- Quartz

Why?

- The IHostedService background task execution is coordinated with the lifetime of the application
- You register tasks when the application starts and you have the opportunity to do graceful clean-up when the application is shutting down.

IHostedService

IHostedService

The fundamental building block of creating a Hosted Service

```
public interface IHostedService
{
    /// <summary>
    /// Triggered when the application host is ready to start the service.
    /// </summary>
    /// <param name="cancellationToken">Indicates that the start process has been aborted.</param>
    Task StartAsync(CancellationToken cancellationToken);

    /// <summary>
    /// Triggered when the application host is performing a graceful shutdown.
    /// </summary>
    /// <param name="cancellationToken">Indicates that the shutdown process should no longer be graceful.</param>
    Task StopAsync(CancellationToken cancellationToken);
}
```

IHostedService

StartAsync blocks the rest of the application from starting

UNLESS, you truly don’t want your app to boot before it finishes

IHostedService

By default, It starts before the application pipeline is configured

IHostedService

To change the default behavior, configure hosted service like below

```
public static IHostBuilder CreateHostBuilder(string[] args) =>
    Host.CreateDefaultBuilder(args)
        .ConfigureWebHostDefaults(webBuilder =>
        {
            webBuilder.UseStartup<Startup>();

        })
        .ConfigureServices(services :IServiceCollection =>
        {
            services.AddHostedService<RefreshCacheHostedService>();
        }); // IHostBuilder
```

IHostedService

Cancellation Token:

- Default timeout 5 seconds
- Remaining operations after timeout should be aborted

```
services.Configure<HostOptions>(option :HostOptions =>
{
    option.ShutdownTimeout = TimeSpan.FromSeconds(20);
});
```

Demo

BackgroundService

BackgroundService

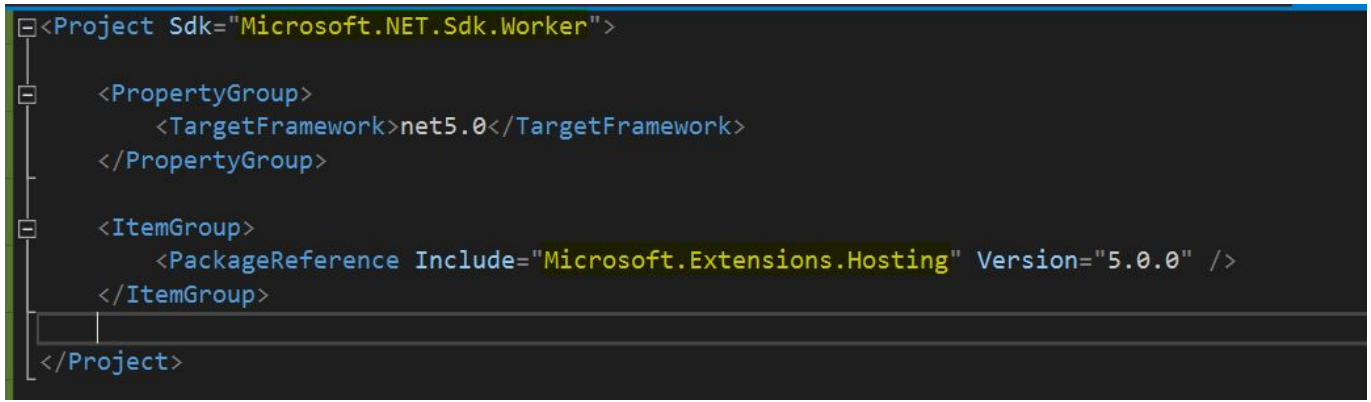
- Abstract class, implements IHostedService
- Exposes **ExecuteAsync()** abstract method
- Handles Starting and Stopping
  - Can still override StartAsync and StopAsync

Demo

WorkerService

WorkerService

- Enhanced .NET Core Console App template
- Allows you to have an IHost



WorkerService

Hosting

- Windows Service
- Systemd
- Windows Scheduler
- Azure Web Job
- K8s Cron Job

WorkerService

Windows: Microsoft.Extensions.Hosting.WindowsService

Linux: Microsoft.Extensions.Hosting.Systemd

```
public static IHostBuilder CreateHostBuilder(string[] args) =>
    Host.CreateDefaultBuilder(args)
        .UseWindowsService()
        .ConfigureServices((hostContext, services) =>
        {
            services.AddHostedService<DailyReportService>();
        }); // IHostBuilder
```

Demo

Hangfire

Hangfire

- Full featured library for running jobs in .NET Core
- Comes with UI for monitoring and history
- Supports Cron and ad-hoc running of jobs
- Automatic retries
- Supports scaling
- Simple to use

Demo

Quartz

Quartz

- More features to deal with recurring jobs
  - Triggers are more powerful
  - suitable for complex scheduling logic
- We should always use **IJob** interface to create scheduler jobs
  - But hangfire can accept any method
- No default dashboard
  - Can use other 3rd Party NuGet package (if required)

Demo

References

- Background tasks with hosted services in ASP.NET Core
- Implement background tasks in microservices
- The Background on Background Tasks in .NET Core

- https://github.com/jsowndarrajan/BackgroundTasks

Q & A

Thank You!