mohamed Tayel
Posted on Jun 9, 2024 • Edited on Dec 4, 2024

Understanding Background Services in .NET 8: IHostedService and BackgroundService

#dotnetcore #aspdotnet #dotnet

C# 12 (12 Part Series)

1 Exploring Primary Constructors in C# 12: Simplifying Class...
2 Title: Simplifying Code with Type Aliases in .NET 8 and C# ...
... **8 more parts...**
11 Exploring Identity API Endpoints in .NET 8: Implementing ...
12 required Keyword in C#

.NET 8 introduces powerful features for managing background tasks with `IHostedService` and `BackgroundService`. These services enable long-running operations, such as scheduled tasks, background processing, and periodic maintenance tasks, to be seamlessly integrated into your applications. This article explores these new features and provides practical examples to help you get started. You can find the source code for these examples on my GitHub repository.

What are Background Services?

Background services in .NET allow you to run tasks in the background independently of the main application thread. This is essential for tasks that need to run continuously or at regular intervals without blocking the main application flow.

`IHostedService` Interface

The `IHostedService` interface defines two methods:

- `StartAsync(CancellationToken cancellationToken)`: Called when the application host starts.
- `StopAsync(CancellationToken cancellationToken)`: Called when the application host is performing a graceful shutdown.

Example of `IHostedService` Implementation:

```
using System;
using System.Threading;
using System.Threading.Tasks;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

public class TimedHostedService : IHostedService, IDisposable
{
    private readonly ILogger<TimedHostedService> _logger;
    private Timer _timer;

    public TimedHostedService(ILogger<TimedHostedService> logger)
    {
        _logger = logger;
    }

    public Task StartAsync(CancellationToken cancellationToken)
    {
        _logger.LogInformation("Timed Hosted Service running.");

        _timer = new Timer(DoWork, null, TimeSpan.Zero, TimeSpan.FromSeconds(5))

        return Task.CompletedTask;
    }

    private void DoWork(object state)
    {
        _logger.LogInformation("Timed Hosted Service is working.");
    }

    public Task StopAsync(CancellationToken cancellationToken)
    {
        _logger.LogInformation("Timed Hosted Service is stopping.");

        _timer?.Change(Timeout.Infinite, 0);

        return Task.CompletedTask;
    }

    public void Dispose()
    {
        _timer?.Dispose();
    }
}
```

`BackgroundService` Class

The `BackgroundService` class is an abstract base class that simplifies the implementation of background tasks. It provides a single method to override:

- `ExecuteAsync(CancellationToken stoppingToken)`: Contains the logic for the background task and runs until the application shuts down.

Example of `BackgroundService` Implementation:

```
using System;
using System.Threading;
using System.Threading.Tasks;
using Microsoft.Extensions.Hosting;
using Microsoft.Extensions.Logging;

public class TimedBackgroundService : BackgroundService
{
    private readonly ILogger<TimedBackgroundService> _logger;

    public TimedBackgroundService(ILogger<TimedBackgroundService> logger)
    {
```

mohamed Tayel
Follow
Technical Project Lead at SURE Egypt with 20+ years in software development. Specializes in .Net, and SQL Server. Passionate about delivering quality solutions

EDUCATION
computer engineering

WORK
.net Technical lead

JOINED
Feb 6, 2024

More from mohamed Tayel

Refactoring Complex Conditions: Clean Code Solutions for Nested If Statements
#csharp #nestedifstatements #codingbestpractices #dotnet

Introduction to Arrays
#csharp #arrays #collections #dotnet

Understanding the Need for Collections in Programming
#csharp #collections #dotnet #ienumerable

```
        _logger = logger;
    }

    protected override async Task ExecuteAsync(CancellationToken stoppingToken)
    {
        _logger.LogInformation("Timed Background Service running.");

        while (!stoppingToken.IsCancellationRequested)
        {
            _logger.LogInformation("Timed Background Service is working.");
            await Task.Delay(TimeSpan.FromSeconds(5), stoppingToken);
        }

        _logger.LogInformation("Timed Background Service is stopping.");
    }
}
```

Practical Usage

To utilize these background services in your .NET application, you need to register them in your dependency injection container. This can be done in the `Program.cs` file.

Registering Hosted Services:

```
using Microsoft.Extensions.DependencyInjection;
using Microsoft.Extensions.Hosting;
using System.Threading.Tasks;

public class Program
{
    public static async Task Main(string[] args)
    {
        var host = Host.CreateDefaultBuilder(args)
            .ConfigureServices(services =>
            {
                services.AddHostedService<TimedHostedService>();
                services.AddHostedService<TimedBackgroundService>();
            })
            .Build();

        await host.RunAsync();
    }
}
```

Key Differences

- **Level of Abstraction**:
  - `IHostedService`: Requires manual implementation of starting and stopping logic.
  - `BackgroundService`: Simplifies the implementation by providing a base class with a single method to override.
- **Use Cases**:
  - `IHostedService`: Suitable for more complex scenarios where you need fine-grained control over the service lifecycle.
  - `BackgroundService`: Ideal for simpler, long-running tasks that benefit from reduced boilerplate code.

Conclusion

.NET 8's background services, through `IHostedService` and `BackgroundService`, offer a robust and flexible way to manage background tasks. By choosing the appropriate abstraction based on your needs, you can efficiently implement and manage long-running operations in your applications. These new features enhance the ability to create responsive, scalable, and maintainable .NET applications.

This guide provides the foundation you need to start integrating background services into your .NET applications. For more complex scenarios, consider exploring additional capabilities and configurations offered by the .NET hosting framework.

C# 12 (12 Part Series)

1 Exploring Primary Constructors in C# 12: Simplifying Class...

2 Title: Simplifying Code with Type Aliases in .NET 8 and C# ...

... **8 more parts...**

11 Exploring Identity API Endpoints in .NET 8: Implementing ...

12 required Keyword in C#

Try the API

Top comments (3)

Subscribe

Add to the discussion

João Angelo • Jun 13 '24

Hi mohamed Tayel,
Your tips are very useful
Thanks for sharing

3 likes Reply

Jorge Yumiseba • Oct 10 '24

I am attempting to use backgroundservices on my aspnet mvc applicaiton. But it blocks the main UI thread until the backgrounservice operations are done

1 like Reply

mohamed Tayel • Oct 11 '24

Make sure your background service runs asynchronously using async/await to prevent blocking the UI thread

1 like Reply

Code of Conduct • Report abuse

Read next

Window Forms Dark mode
Karen Payne - Nov 16 '24

Você sabe o que é Message Broker?
Ramon Xavier - Nov 16 '24

Asp.Net Core and Keycloak testcontainer. Testing a secure Asp.Net Core Api using Keycloak Testcontainer
Horatiu - Nov 16 '24

iMate - Building a Mobile App for Mood Tracking
Richard - Nov 16 '24