**IHostedService**

یکی از interface های مهم در .net است که برای ساختن سرویس های long running کاربرد دارد از نوع backGroundService

هر وقت بخواهیم کاری در پشت صحنه همیشه همراه برنامه اجرا شود مثل ( زمانبندی ، مانیتورینگ، ارسال پیام) می توان از آن استفاده کرد.

2 متد اصلی دارد

public interface IHostedService

{

Task StartAsync(CancellationToken cancellationToken); // وقتی برنامه اجرا میشه

Task StopAsync(CancellationToken cancellationToken); // وقتی برنامه متوقف میشه

}

مثال لاگ گرفتن :

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("سرویس زمان‌بندی شروع شد.");

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

return Task.CompletedTask;

}

private void DoWork(object? state)

{

\_logger.LogInformation("کار در حال اجراست: {time}", DateTimeOffset.Now);

}

public Task StopAsync(CancellationToken cancellationToken)

{

\_logger.LogInformation("سرویس زمان‌بندی متوقف شد.");

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

return Task.CompletedTask;

}

public void Dispose()

{

\_timer?.Dispose();

}

}

در نهایت نیاز به ثبت سرویس در program.cs داریم

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddHostedService<TimedHostedService>();

var app = builder.Build();

app.Run();

**🔑 نکته:**

* هر چیزی که با IHostedService بسازی، عمرش به عمر اپ وابسته‌ست.
* برای راحتی، مایکروسافت یه کلاس انتزاعی به اسم BackgroundService هم داده که کار با IHostedService رو ساده‌تر می‌کنه. (کافیه ExecuteAsync رو override کنی).

مثال با BackgroundService

همان قبلی است با این تفاوت کلاس Abstract است که باید متد execute آن را override کرد.

و در نهایت در program.cs باید ثبت شود

public class TimedBackgroundService : BackgroundService

{

private readonly ILogger<TimedBackgroundService> \_logger;

public TimedBackgroundService(ILogger<TimedBackgroundService> logger)

{

\_logger = logger;

}

protected override async Task ExecuteAsync(CancellationToken stoppingToken)

{

\_logger.LogInformation("سرویس Background شروع شد.");

while (!stoppingToken.IsCancellationRequested)

{

\_logger.LogInformation("کار در حال اجراست: {time}", DateTimeOffset.Now);

await Task.Delay(TimeSpan.FromSeconds(5), stoppingToken);

}

\_logger.LogInformation("سرویس Background متوقف شد.");