BCSCHOOL

Empowering Digital Skills For The Jobs Of The Future

by

BC Soft
consulenza sistemi informativi

AiRED
Artificial Intelligence Ricerca e Sviluppo

Academy .NET

Generic Host

Sommario

- Utilizzo di host generici
- Configurazione host
- Avvio e Arresto Host
- CancelationToken
- Hosted Services
- HostBuilder

I Microservizi

I **microservizi** potrebbero dover eseguire **thread indipendenti** allo scopo di svolgere operazioni differenti.

I thread richiedono risorse differenti tra cui:

- Connessioni a database
- Canali di comunicazione
- Moduli specializzati

I thread necessitano l'inizializzazione quando il microservizio viene avviato e fermati quando il microservizio non è più operativo.

Host

Un **host** fornisce le condizioni adeguate per eseguire attività differenti chiamati **hosted services**.

Fornisce agli **hosted services:**

risorse

Impostazioni comuni

Start/stop

Host

Tutte le funzionalità di **host** sono contenute nel package **Microsoft.Extensions.Hosting NuGet**.

Configurazione host

Per configurare un **host** è necessario:

- Creazione istanza **HostBuilder**
- Chiamata al motodo **Build**

Il metodo **Build** si occupa della configurazione dell'host con le informazioni fornite:

```
var myHost=new HostBuilder()
//Several chained calls
//defining Host configuration
.Build();
```

Configurazione host

Configurazione **host** include:

- Definizione **risorse comuni**
- Definizione cartella predefinita per file
- Caricamento parametri configurazione da diverse sorgenti(es. file JSON)

Avvio e Arresto Host

Avvio Host:

```
host.Start();
```

Arresto Host:

```
await host.StopAsync (timeout).
```

Tempo massimo prima che i servizi si interrompano correttamente

CancelationToken

L’arresto di un microservizio viene segnalato da un **CancelationToken**.

Il **CancelationToken** viene passato quando il microservizio viene avviato dall’orchestrator.

Accade quando i microservizi sono ospitati in **Azure Service Fabric**.

In questo caso non si utilizza host.Start ma **RunAsync**.

```
await host.RunAsync(cancelationToken)
```

Hosted Services

Gli hosted services implementano l'interfaccia **IHostedService.**

Metodi:

- **StartAsync(CancellationToken)**
- **StopAsync(CancellationToken)**

StartAsync segnala una richiesta di arresto

StopAsync segnala che il timeout di arresto è scaduto

ConfigureServices

- Gli **hosted services** vengono dichiarati nel metodo **ConfigureServices**
- **ConfigureServices** viene utilizzato per definire opzioni host:

```
using  Microsoft.Extensions.DependencyInjection;


services.AddHostedService<MyHostedService>();
```

Entensione DependencyInjection

IHostedService

L’interfaccia **IHostedService**, se non implementata direttamente, può essere ereditata da classe astratta **BackgroundService**.

BackgroundService presenta il metodo **ExecuteAsync(CancellationToken)**

Il metodo **ExecuteAsync(CancellationToken)** permette la gestione dell’intera logica del servizio.

Arresto Host

Necessaria dichiarazione interfaccia **IApplicationLifetime**:

```
public class MyHostedService: BackgroundService
{
private applicationLifetime;
public MyHostedService(IApplicationLifetime applicationLifetime)
{
this.applicationLifetime=applicationLifetime;
}
protected Task ExecuteAsync(CancellationToken token)
{
...
applicationLifetime.StopApplication();
...
}
}
```

Quando il servizio ospitato viene creato, verrà automaticamente passata un'implementazione di **IApplicationLifetime**

HostBuilder

HostBuilder presenta un metodo che è possibile utilizzare per definire la cartella predefinita:

```
.UseContentRoot("c:\\<deault path>")
```

Presenta metodi che è possibile usare per aggiungere **logging targets**:

```
.ConfigureLogging((hostContext, configLogging) =>
{
configLogging.AddConsole();
configLogging.AddDebug();
})
```

HostBuilder

Infine, HostBuilder presenta metodi che è possibile usare per leggere i parametri di configurazione da diverse fonti:

```
.ConfigureHostConfiguration(configHost =>
{
configHost.AddJsonFile("settings.json", optional: true);
configHost.AddEnvironmentVariables(prefix: "PREFIX_");
configHost.AddCommandLine(args);
})
```

Domande & approfondimenti

Academy .NET