.NET 的一个重要的概念IHost（主机），其由HostBuilder生成

**示例**

var builder = new HostBuilder();

// 生成Host

builder.Build()

    // 运行Host

    .Run();

**HostBuilder提供的功能**

HostBuilder提供了以下功能

1. 依赖关系注入容器 (DI)
2. Logging注册
3. Configuration配置注册
4. IHostedService 注册
5. Host生命周期管理

**依赖关系注入容器**

HostBuilder 提供已了默认的依赖注入容器，如果我们想使用其他依赖注入容器，则可以调用UseServiceProviderFactory替换

var builder = new HostBuilder();

builder.UseServiceProviderFactory(new AutofacServiceProviderFactory());

**Logging注册**

HostBuilder默认注册了如下日志提供者，如果需要其他日志则需要自己注册

- 控制台

- 调试

- EventSource

- EventLog（仅当在 Windows 上运行时）

日志的注册五花八门，这里不示例了

**Configuration配置注册**

我们可以使用如下方法给Host添加配置

builder.ConfigureHostConfiguration((config) =>

{

    // 配置Host的路径

    config.SetBasePath(Directory.GetCurrentDirectory());

    // 添加Json配置文件

    config.AddJsonFile("hostsettings.json", optional: true);

    // 添加环境变量配置

    config.AddEnvironmentVariables(prefix: "PREFIX\_");

    // 添加命令行配置

    config.AddCommandLine(args);

    Dictionary<string, string> source = new Dictionary<string, string>

    {

        ["format:dateTime:longDatePattern"] = "dddd, MMMM d, yyyy",

        ["format:dateTime:longTimePattern"] = "h:mm:ss tt",

        ["format:dateTime:shortDatePattern"] = "M/d/yyyy",

        ["format:dateTime:shortTimePattern"] = "h:mm tt",

    };

    // 添加内存配置

    config.AddInMemoryCollection(source);

});

**IHostedService 注册**

我们可以通过如下方法注册IHostedService

builder.ConfigureServices((\_, services) =>

{

    services.AddHostedService<ExampleHostedService>()

});

当Host启动时，会调用IHostedService的StartAsync方法，当Host停止时会IHostedService的调用StopAsync方法

**HostBuilder注册的服务**

HostBuilder注册了以下服务，我们可以通过依赖注入获取

1. IHostApplicationLifetime

通过IHostApplicationLifetime我们可以向Host注册生命周期函数

public class ExampleHostedService

{

    private readonly ILogger \_logger;

    public ExampleHostedService(

        ILogger<ExampleHostedService> logger,

        IHostApplicationLifetime appLifetime)

    {

        \_logger = logger;

        // Host 启动后

        appLifetime.ApplicationStarted.Register(OnStarted);

        // Host 即将停止

        appLifetime.ApplicationStopping.Register(OnStopping);

        // Host 停止后

        appLifetime.ApplicationStopped.Register(OnStopped);

}

    private void OnStarted()

    {

        \_logger.LogInformation("2. OnStarted has been called.");

}

    private void OnStopping()

    {

        \_logger.LogInformation("3. OnStopping has been called.");

}

    private void OnStopped()

    {

        \_logger.LogInformation("5. OnStopped has been called.");

    }

}

1. IHostLifetime

IHostLifetime用于控制Host何时启动和停止

1. IHostEnvironment

IHostEnvironment保存Host的环境信息

// App 的名称

IHostEnvironment.ApplicationName

// 程序的根目录文件提供器

IHostEnvironment.ContentRootFileProvider

// 程序的根目录

IHostEnvironment.ContentRootPath

// 当前环境（开发开始生产环境）

IHostEnvironment.EnvironmentName

**Asp.net core站点与HostBuilder**

Asp.net core站点其实就是一个Host，这个Host包含了一个IHostedService（GenericWebHostService），GenericWebHostService监听我们的端口并将请求映射到我们的控制器

builder.ConfigureServices((context, services) => services.AddHostedService<GenericWebHostService>());

**HostBuilder不局限于站点**

HostBuilder并不一定要用于站点搭建，还可以用于控制台和后台服务等

示例1：如下我们使用HostBuilder构建控制台应用

namespace TestWeb2

{

    public class MyService : IHostedService

    {

        private IHttpClientFactory \_httpClientFactory { get; }

        // 依赖注入 IHttpClientFactory

        public MyService(IHttpClientFactory httpClientFactory) {

            this.\_httpClientFactory = httpClientFactory;

        }

        public async Task StartAsync(CancellationToken cancellationToken)

        {

            while (true) {

                Console.WriteLine("请输入命令");

                Console.WriteLine("1: 请求 \\ 2: 待开发");

                string cmd = Console.ReadLine();

                if (cmd == "1")

                {

                    Console.WriteLine("执行命令1");

                }

                else {

                    Console.WriteLine("无效的命令");

                }

            }

        }

        public Task StopAsync(CancellationToken cancellationToken)

        {

            return Task.CompletedTask;

        }

}

    class Program

    {

        static void Main(string[] args)

        {

            IHostBuilder builder = new HostBuilder();

            builder.ConfigureServices(services =>

            {

                services.AddHostedService<MyService>();

            });

            builder.Build().Run();

            Console.WriteLine("程序已退出");

        }

    }

}

示例2：如下是 .NET 后台服务的模板代码

public class Program

{

    public static void Main(string[] args)

    {

        CreateHostBuilder(args).Build().Run();

    }

    public static IHostBuilder CreateHostBuilder(string[] args) =>

        Host.CreateDefaultBuilder(args)

            .ConfigureServices((hostContext, services) =>

            {

                // 注册Host服务

                services.AddHostedService<Worker>();

            });

}