与微软选项类似

**选项**

1. 在模块中定义选项类

public class MyOptions

{

    public int Value1 { get; set; }

    public bool Value2 { get; set; }

}

1. 在Module类的ConfigureServices方法中注册选项

public override void ConfigureServices(

    ServiceConfigurationContext context)

{

    Configure<MyOptions>(options =>

    {

        options.Value1 = 42;

        options.Value2 = true;

    });

}

1. 在需要的地方使用依赖注入

public class MyService : ITransientDependency

{

    private readonly MyOptions \_options;

    public MyService(IOptions<MyOptions> options)

    {

        \_options = options.Value; //Notice the options.Value usage!

    }

    public void DoIt()

    {

        var v1 = \_options.Value1;

        var v2 = \_options.Value2;

    }

}

**预配置**

如果我们想在模块的ConfigureServices方法使用配置，就需要预配置功能

1. 定义预配置类

public class MyPreOptions

{

    public bool MyValue { get; set; }

}

1. 在Module类中注册配置类

public override void PreConfigureServices(

    ServiceConfigurationContext context)

{

    PreConfigure<MyPreOptions>(options =>

    {

        options.MyValue = true;

    });

}

1. 在ConfigureServices方法中使用配置

public override void ConfigureServices(

    ServiceConfigurationContext context)

{

    var options = context.Services.ExecutePreConfiguredActions<MyPreOptions>();

    if (options.MyValue)

    {

        //...

    }

}