## **如何记录日志**

注入日志记录器

public class TodoController : Controller

{

    private readonly ILogger \_logger;

    public TodoController(

        ILogger<TodoController> logger)

    {

        \_todoRepository = todoRepository;

        \_logger = logger;

    }

然后在该记录器对象上调用日志记录方法

public IActionResult GetById (string id) {

    \_logger.LogInformation (LoggingEvents.GetItem, "Getting item {ID}", id);

    \_logger.LogWarning (LoggingEvents.GetItemNotFound, "GetById({ID}) NOT FOUND", id);

    ...

}

## **如何添加提供程序**

public static void Main (string[] args) {

    var webHost = new WebHostBuilder ()

        .UseKestrel ()

        .UseContentRoot (Directory.GetCurrentDirectory ())

        // 配置 配置文件

        .ConfigureAppConfiguration ((hostingContext, config) => {

            var env = hostingContext.HostingEnvironment;

            config.AddJsonFile ("appsettings.json", optional : true, reloadOnChange : true)

                .AddJsonFile ($"appsettings.{env.EnvironmentName}.json", optional : true, reloadOnChange : true);

            config.AddEnvironmentVariables ();

        })

        // 添加日志提供者

        .ConfigureLogging ((hostingContext, logging) => {

            logging.AddConfiguration (hostingContext.Configuration.GetSection ("Logging"));

            logging.AddConsole ();

            logging.AddDebug ();

        })

        .UseStartup<Startup> ()

        .Build ();

    webHost.Run ();

}

## **配置**

日志记录配置通常由应用设置文件的 Logging 部分提供。

{

    "Logging": {

        // 要使用的日志级别，大于或等于该级别的日志将会被记录

        "LogLevel": {

            "Default": "Debug",

            "System": "Information",

            "Microsoft": "Information"

        },

        "Console": {

            "IncludeScopes": "true"

        }

    }

}

## **日志类别**

在创建 ILogger 对象时指定类别。 类别可以是任意字符串，但约定使用写入日志的类的完全限定名称。

如下方示例所示，在大多数情况下使用 ILogger<T> 更简单。

public class TodoController : Controller

{

    private readonly ILogger \_logger;

    public TodoController(ILogger<TodoController> logger)

    {

        \_logger = logger;

    }

## **日志级别**

ASP.NET Core 定义了以下[日志级别](https://docs.microsoft.com/zh-cn/dotnet/api/microsoft.extensions.logging.loglevel)（按严重性从低到高排列）。

Trace = 0

表示仅对于开发人员调试问题有价值的信息。不要在生产环境中启用它们。 默认情况下禁用。

Debug = 1

表示在开发和调试过程中短期有用的信息。 通常不会在生产中启用 Debug 级别日志，因为日志数量过多。

Information = 2

用于跟踪应用程序的常规流。 这些日志通常有长期价值。

Warning = 3

表示应用程序流中的异常或意外事件。

Error = 4

表示无法处理的错误和异常。

Critical = 5

需要立即关注的失败。 例如数据丢失、磁盘空间不足。

## **日志事件 ID**

每次写入日志时都可指定一个事件 ID。

### **筛选日志规则**

配置数据根据 提供程序 和 类别 指定 最低日志级别 ，如下方示例所示：

{

    "Logging": {

        // Debug 提供器使用的配置

        "Debug": {

            "LogLevel": {

                "Default": "Information"

            }

        },

        // 控制台提供器使用的配置

        "Console": {

            "IncludeScopes": false,

            "LogLevel": {

                "Microsoft.AspNetCore.Mvc.Razor.Internal": "Warning",

                "Microsoft.AspNetCore.Mvc.Razor.Razor": "Debug",

                "Microsoft.AspNetCore.Mvc.Razor": "Error",

                "Default": "Information"

            }

        },

        // 其他提供器使用的配置

        "LogLevel": {

            "Default": "Debug"

        }

    }

}

## **内置日志记录提供程序（提供器）**

ASP.NET Core 提供以下提供程序：

* [控制台](https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/logging/?view=aspnetcore-2.1" \l "console-provider)
* [调试](https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/logging/?view=aspnetcore-2.1" \l "debug-provider)
* [EventSource](https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/logging/?view=aspnetcore-2.1" \l "eventsource-provider)
* [EventLog](https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/logging/?view=aspnetcore-2.1" \l "windows-eventlog-provider)
* [TraceSource](https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/logging/?view=aspnetcore-2.1" \l "tracesource-provider)
* [Azure 应用服务](https://docs.microsoft.com/zh-cn/aspnet/core/fundamentals/logging/?view=aspnetcore-2.1" \l "azure-app-service-provider)