Asp.NetCore

## 基础

### 1、IOC

控制反转: 转移具体流程的控制权。对于asp.netcore来说，流程是指从接收到用户请求开始、到解析出Controll、执行Action、输出View的整个流程。Asp.netcore，允许用户参与到流程执行的各个环节中，将流程的控制权转交给用户。类似于比较熟悉的模板方法。

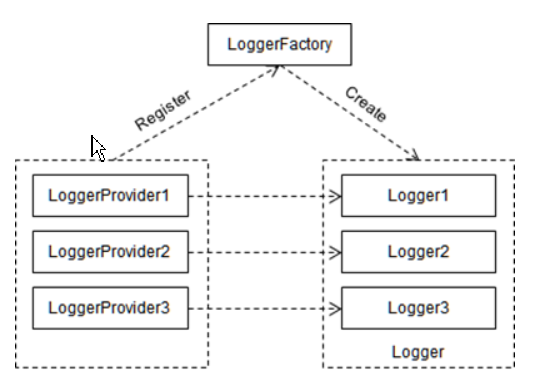
### 2、Dependency Injection

依赖注入: 在asp.netcore中，DI被广泛使用。它自身维护一个ID容器,在Startup中将所有依赖注入。Demo中演示的是从框架、数据层、业务层全部通过注入。

### 3、配置文件

### 4、日志

NetCore日志架构图



Logger1~n: 处理日志的真实写入操作。

LoggerProvider: 负责创建Logger1~n。

LoggerFactroy: 负责注册LoggerProvider，并创建一个主的ILogger对象，这个对象可以理解为所有由注册到LoggerFactroy中所有LoggerProvider创建的ILogger对象的集合，调用这个主的ILogger对象会依次调用ILogger1~ILoggerN的所有写入方法。

#### 简单的Demo:

添加Nuget包

Install-Package Microsoft.Extensions.Logging

Install-Package Microsoft.Extensions.Logging.Console

Install-Package Microsoft.Extensions.Logging.Debug

其中: Logging.Console, Logging.Debug分别为实现了ILoggerProvider的控制台输出与调试窗口输出。

在代码中使用:

// 控制各个ILogger写入日志的级别

Func<string, LogLevel, bool> filter = (category, level) => true;

ILoggerFactory loggerFactory = new LoggerFactory();

loggerFactory.AddProvider(new ConsoleLoggerProvider(filter, false));

loggerFactory.AddProvider(new DebugLoggerProvider(filter));

// nameof(Program): 事件源

ILogger logger = loggerFactory.CreateLogger(nameof(Program));

// 事件Id

int eventId = 3721;

logger.LogInformation(eventId, "Hello World NewCore Logger:{auther}", "cnf");

logger.LogWarning(eventId, "Hello World NewCore Logger:{auther}", "cnf");

logger.LogError(eventId, "Hello World NewCore Logger:{auther}", "cnf");