# Log4Net

## Package

Install-Package Microsoft.Extensions.Logging.Log4Net.AspNetCore -Version 3.0.0

Se si vuole scrivere il log su db Sql Server installare il seguente package

Install-Package MicroKnights.Log4NetAdoNetAppender -Version 1.0.2

## Log4net.config e appsettings.json

<log4net debug="true">

<appender name="RollingLogFileAppender" type="log4net.Appender.RollingFileAppender">

<file value="LOGS/CoreWebApi.log"/>

<appendToFile value="true" />

<rollingStyle value="Size" />

<maxSizeRollBackups value="10" />

<maximumFileSize value="25MB" />

<staticLogFileName value="true" />

<layout type="log4net.Layout.PatternLayout">

<!--<conversionPattern value="%-5p %d %5rms %-22.22c{1} - %m%n" />-->

<conversionPattern value="%date %level %message%newline" />

</layout>

</appender>

<appender name="AdoNetAppender" type="MicroKnights.Logging.AdoNetAppender, MicroKnights.Log4NetAdoNetAppender">

<bufferSize value="1" />

<connectionType value="System.Data.SqlClient.SqlConnection, System.Data" />

<connectionStringName value="log4net" />

<connectionStringFile value="appsettings.json" />

<commandText value="INSERT INTO Log ([Date],[Thread],[Level],[Logger],[Message],[Exception]) VALUES (@log\_date, @thread, @log\_level, @logger, @message, @exception)" />

<parameter>

<parameterName value="@log\_date" />

<dbType value="DateTime" />

<layout type="log4net.Layout.RawTimeStampLayout" />

</parameter>

<parameter>

<parameterName value="@thread" />

<dbType value="String" />

<size value="255" />

<layout type="log4net.Layout.PatternLayout">

<conversionPattern value="%thread" />

</layout>

</parameter>

<parameter>

<parameterName value="@log\_level" />

<dbType value="String" />

<size value="50" />

<layout type="log4net.Layout.PatternLayout">

<conversionPattern value="%level" />

</layout>

</parameter>

<parameter>

<parameterName value="@logger" />

<dbType value="String" />

<size value="255" />

<layout type="log4net.Layout.PatternLayout">

<conversionPattern value="%logger" />

</layout>

</parameter>

<parameter>

<parameterName value="@message" />

<dbType value="String" />

<size value="4000" />

<layout type="log4net.Layout.PatternLayout">

<conversionPattern value="%message" />

</layout>

</parameter>

<parameter>

<parameterName value="@exception" />

<dbType value="String" />

<size value="2000" />

<layout type="log4net.Layout.ExceptionLayout" />

</parameter>

</appender>

<root>

<!-- minimum level to log -->

<level value="DEBUG" />

<appender-ref ref="RollingLogFileAppender" />

<appender-ref ref="AdoNetAppender" />

</root>

</log4net>

<!--Creazione della tabella su SQL-Server

CREATE TABLE [dbo].[Log] (

[Id] [int] IDENTITY (1, 1) NOT NULL,

[Date] [datetime] NOT NULL,

[Thread] [varchar] (255) NOT NULL,

[Level] [varchar] (50) NOT NULL,

[Logger] [varchar] (255) NOT NULL,

[Message] [varchar] (4000) NOT NULL,

[Exception] [varchar] (2000) NULL

)-->

Se si vuole utilizzare AdoNetAppender occorre specificare la connectionstring in appsettings.json come segue :

"ConnectionStrings": {

"DefaultConnection": "Data Source=DESKTOP-5G6VD0K\\SQLEXPRESS;Initial Catalog=ACL; User ID=sa; Password=xxx;",

"log4net": "Data Source=DESKTOP-5G6VD0K\\SQLEXPRESS;Initial Catalog=ACL; User ID=sa; Password=xxxx;"

},

All’interno del tag root specificare il level da tracciare che può essere, in ordine gerarchico :

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| ALL | DEBUG | INFO | WARN | ERROR | FATAL | OFF |
| ALL |  |  |  |  |  |  |
| DEBUG | DEBUG |  |  |  |  |  |
| INFO | INFO | INFO |  |  |  |  |
| WARN | WARN | WARN | WARN |  |  |  |
| ERROR | ERROR | ERROR | ERROR | ERROR |  |  |
| FATAL | FATAL | FATAL | FATAL | FATAL | FATAL |  |
| OFF | OFF | OFF | OFF | OFF | OFF | OFF |

Mentre con il tag <appender-ref indicare con quale/i modalità deve essere scritto il log.

## Startup.cs

Nel metodo Configure aggiungere la seguente riga :

loggerFactory.AddLog4Net();

## Logger.cs

public static class Logger

{

private static readonly string LOG\_CONFIG\_FILE = @"log4net.config";

private static readonly log4net.ILog \_log = GetLogger(typeof(Logger));

public static ILog GetLogger(Type type)

{

return LogManager.GetLogger(type);

}

public static void Info(object message)

{

SetLog4NetConfiguration();

\_log.Info(message);

}

public static void Debug(object message)

{

SetLog4NetConfiguration();

\_log.Debug(message);

}

public static void Warning(object message)

{

SetLog4NetConfiguration();

\_log.Warn(message);

}

public static void Error(object message)

{

SetLog4NetConfiguration();

\_log.Error(message);

}

private static void SetLog4NetConfiguration()

{

XmlDocument log4netConfig = new XmlDocument();

log4netConfig.Load(File.OpenRead(LOG\_CONFIG\_FILE));

var repo = LogManager.CreateRepository(

Assembly.GetEntryAssembly(), typeof(log4net.Repository.Hierarchy.Hierarchy));

log4net.Config.XmlConfigurator.Configure(repo, log4netConfig["log4net"]);

}

public static string GetLogFileName()

{

String filename = null;

IAppender[] appenders = \_log.Logger.Repository.GetAppenders();

// Check each appender this logger has

foreach (IAppender appender in appenders)

{

Type t = appender.GetType();

// Get the file name from the first FileAppender found and return

if (t.Equals(typeof(FileAppender)) || t.Equals(typeof(RollingFileAppender)))

{

filename = ((FileAppender)appender).File;

break;

}

}

return filename;

}

}

## Scrivere un log

Logger.Debug(string.Concat("Errore durante esecuzione script SQL ", file));