# Easy Mode Guide LoggerSQL

## Download the Example Project

<https://github.com/Davidfcr/LoggerWorkshop>

## Prepare SQL Structure

Serilog can create the table using the property “autoCreateSqlTable”: “true”.

But if you want to create the structure manually, you can use the next script. This allow to have more control over the schema, like use the xml format instead of nvarchar for the Properties column.

CREATE TABLE [Log] (

   [Id] int IDENTITY(1,1) NOT NULL,

   [Message] nvarchar(max) NULL,

   [MessageTemplate] nvarchar(max) NULL,

   [Level] nvarchar(128) NULL,

   [TimeStamp] datetimeoffset(7) NOT NULL,

   [Exception] nvarchar(max) NULL,

   [Properties] nvarchar(max) NULL,

   [LogEvent] nvarchar(max) NULL

   CONSTRAINT [PK\_Log]

     PRIMARY KEY CLUSTERED ([Id] ASC)

)

## Review the Project

First, install the next NuGets:

1. Serilog.AspNetCore
2. Serilog.Settings.Configuration
3. Serilog.Sinks.MSSqlServer

Second, head to program.cs and updated it with this lines in order to setup the logger.

public class Program

{

    public static IConfiguration Configuration { get; } = new ConfigurationBuilder()

        .SetBasePath(Directory.GetCurrentDirectory())

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

        .AddJsonFile($"appsettings.{Environment.GetEnvironmentVariable("ASPNETCORE\_ENVIRONMENT") ?? "Production"}.json", optional: true)

        .Build();

    public static void Main(string[] args)

    {

        Log.Logger = new LoggerConfiguration()

            .ReadFrom.Configuration(Configuration)

            .CreateLogger();

        try

        {

            Log.Information("Getting the motors running...");

            BuildWebHost(args).Run();

        }

        catch (Exception ex)

        {

            Log.Fatal(ex, "Host terminated unexpectedly");

        }

        finally

        {

            Log.CloseAndFlush();

        }

    }

    public static IWebHost BuildWebHost(string[] args) =>

        WebHost.CreateDefaultBuilder(args)

               .UseStartup<Startup>()

               .UseConfiguration(Configuration)

               .UseSerilog()

               .Build();

}

This code add the configuration to use the environment setup up in the appsettings.json, and the 3 lines at the start of the Main method tells the application to activate Serilog as the logger. In the BuildWebHost method we need to add the Serilog service as well.

Next up, add a new appsettings.json file and configure Serilog using this parameters, if you created the table manually, wipe out the autoCreateSqlTable property:

{

  "Serilog": {

    "MinimumLevel": "Information",

    "WriteTo": [

      {

        "Name": "MSSqlServer",

        "Args": {

          "connectionString": "<connection string>",

          "tableName": "Log",

"autoCreateSqlTable": "true"

        }

      }

    ]

  },

}

All ready. Start logging using the next line wherever you want to send messages into the Database.

Log.Information("Pong received");

You can use the logging structured objects like {@object} in the message template.

Log.Information("Pong couldn’t be sended back, Error: {@ex.Message}", ex.Message);