

# Logging Lab

Perform these labs on your own computer using Visual Studio 2022 to ensure you understand the lessons presented in the corresponding videos and lectures.

## Lab 1: Inject Logger into LogTest Controller & Write Messages

Right mouse-click on the **Controllers** folder and add a new class **LogTestController**. Replace the contents of this new file with the following code.

```
using Microsoft.AspNetCore.Mvc;

namespace AdvWorksAPI.Controllers;

[Route("api/[controller]")]
[ApiController]
public class LogTestController : ControllerBase
{
    private readonly ILogger<LogTestController> _Logger;

    public LogTestController(ILogger<LogTestController>
logger)
    {
        _Logger = logger;
    }
}
```

Add a **private** method named **WriteLogMessages()** that looks like the following:

```
private void WriteLogMessages()
{
    // The following are in the Log Level order
    _Logger.LogTrace("This is a Trace log entry");
    _Logger.LogDebug("This is a Debug log entry.");
    _Logger.LogInformation("This is an Information log
entry.");
    _Logger.LogWarning("This is a Warning log entry.");
    _Logger.LogError("This is an Error log entry.");
    _Logger.LogError(new ApplicationException("This is an
exception."), "Exception Object");
    _Logger.LogCritical("This is a Critical log entry.");
}
```

Add a new action method to the `LogTestController` class named `WriteMessages()`

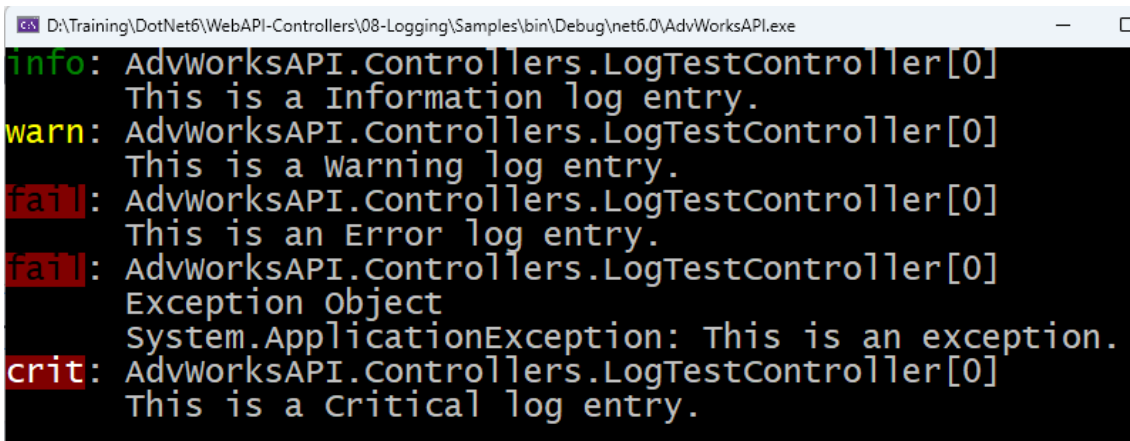
```
[HttpGet]
[Route("WriteMessages")]
public string WriteMessages()
{
    // Write Log Messages
    WriteLogMessages();

    return "Check Console Window";
}
```

## Try it Out

Run the application and click on the **GET /api/LogTest/WriteMessages** button

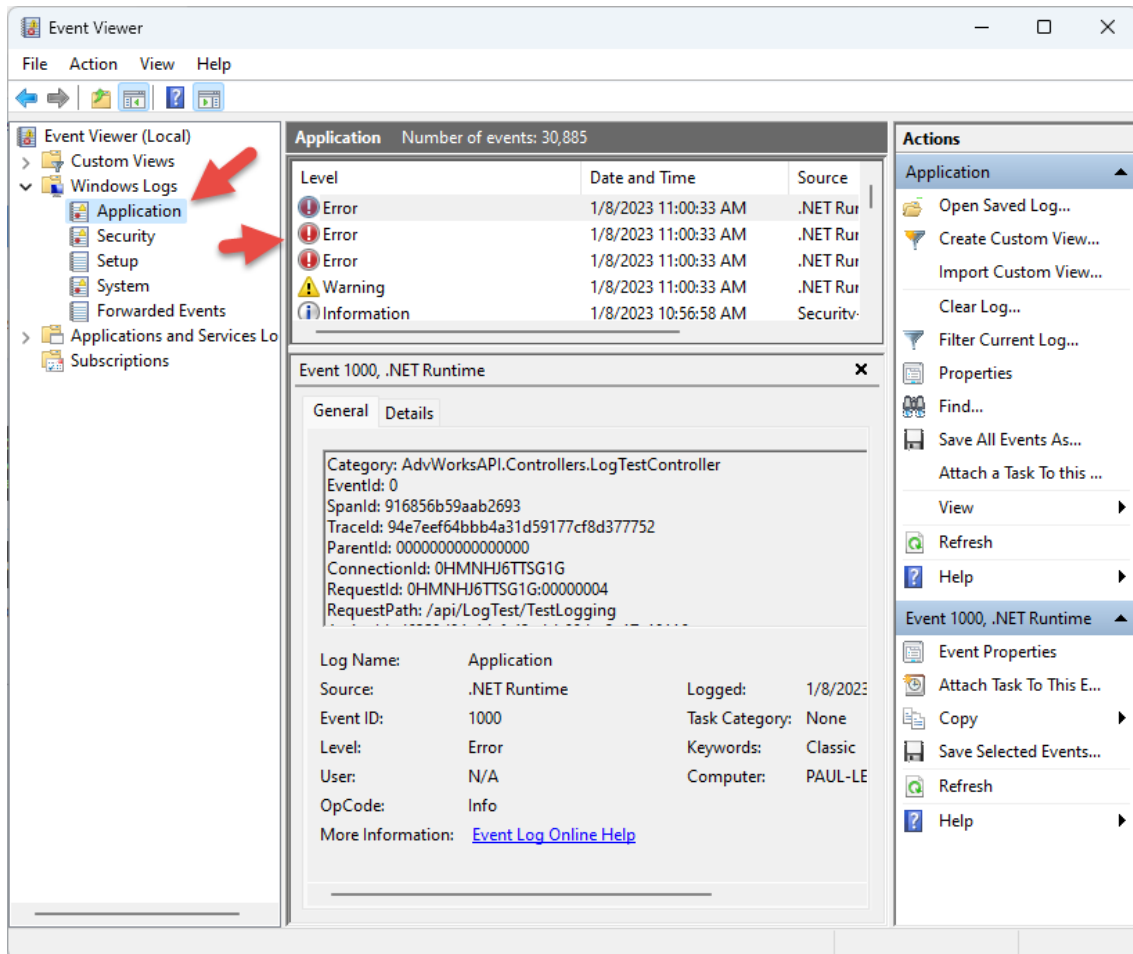
The output / console window should look similar to the following screen shot.



```
D:\Training\DotNet6\WebAPI-Controllers\08-Logging\Samples\bin\Debug\net6.0\AdvWorksAPI.exe
info: AdvWorksAPI.Controllers.LogTestController[0]
      This is a Information log entry.
warn: AdvWorksAPI.Controllers.LogTestController[0]
      This is a warning log entry.
fail: AdvWorksAPI.Controllers.LogTestController[0]
      This is an Error log entry.
fail: AdvWorksAPI.Controllers.LogTestController[0]
      Exception Object
      System.ApplicationException: This is an exception.
crit: AdvWorksAPI.Controllers.LogTestController[0]
      This is a critical log entry.
```

## Check Event Viewer

Bring up the event viewer and view the messages in there.



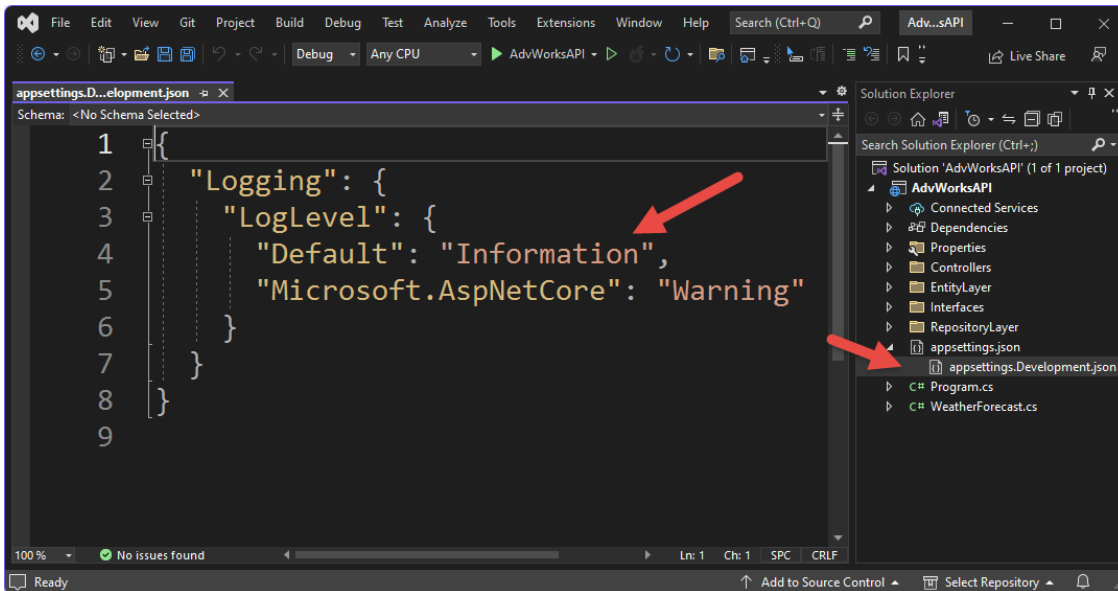
**NOTE:** Logging is synchronous only. Don't write directly to a slow data store such as SQL Server. Instead write to an in-memory queue and use a background worker or maybe a Windows Service to pull the messages out of the queue.

## Lab 2: Control Log Level

Notice in the previous example the **Trace** and **Debug** messages did **NOT** appear.

The "Default" level is set to "Information". Since Trace and Debug have a lower numeric value than Information, they do not show up.

Open the **appsettings.Development.json** file



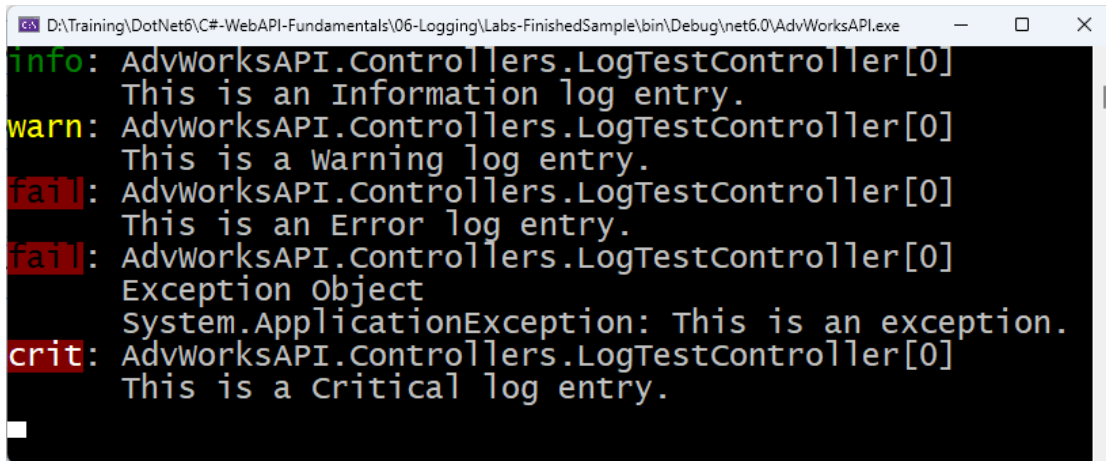
Set the "Default" property to **Trace**

```
{
  "Logging": {
    "LogLevel": {
      "Default": "Trace",
      "Microsoft.AspNetCore": "Warning"
    }
  }
}
```

## Try it Out

Run the application and click on the **GET /api/LogTest/WriteMessages** button

The output / console window should look similar to the following screen shot.



```
D:\Training\DotNet6\C#\WebAPI-Fundamentals\06-Logging\Labs-FinishedSample\bin\Debug\net6.0\AdvWorksAPI.exe
info: AdvWorksAPI.Controllers.LogTestController[0]
      This is an Information log entry.
warn: AdvWorksAPI.Controllers.LogTestController[0]
      This is a warning log entry.
fail: AdvWorksAPI.Controllers.LogTestController[0]
      This is an Error log entry.
fail: AdvWorksAPI.Controllers.LogTestController[0]
      Exception Object
      System.ApplicationException: This is an exception.
crit: AdvWorksAPI.Controllers.LogTestController[0]
      This is a critical log entry.
```

Open the **appsettings.Development.json** file and set the "Default" to "None"

Run the application and click on the **GET /api/LogTest/WriteMessages** button and notice that NO messages are now displayed.

Open the **appsettings.Development.json** file and set the "Default" back to "Information".

## Lab 3: Logging an Object

Open the **LogTestController.cs** file and add a new method named **LogCustomer()**.

```
private void LogCustomer()
{
    // Log an Object
    Customer entity = new()
    {
        CustomerID = 999,
        FirstName = "Bruce",
        LastName = "Jones",
        Title = "Mr.",
        CompanyName = "Beach Computer Consulting",
        EmailAddress =
"Jones.Bruce@beachcomputerconsulting.com",
        Phone = "(714) 555-5555",
        ModifiedDate = DateTime.Now
    };

    string json =
JsonSerializer.Serialize<Customer>(entity);

    _Logger.LogInformation("Customer = {json}", json);
}
```

### Add a new action method

```
[HttpGet]
[Route("LogObject")]
public string LogObject()
{
    // Write Customer Object to Log
    LogCustomer();

    return "Check Console Window";
}
```

## Try it Out

Run the application and click on the **GET /api/LogTest/LogObject** button.

Check the Console window to see the serialized Customer object.

## Lab 4: Log to a File

Right mouse-click on the AdvWorksAPI project and select **Manage NuGet Packages...**

Click on the **Browse** tab.

Type in **Serilog.AspNetCore**. Locate the appropriate version for the version of .NET you are running. Click the **Install** button.

Type in **Serilog.Sinks.File**. Locate the appropriate version for the version of .NET you are running. Click the **Install** button.

Open the **Program.cs** file and add a using statement.

```
using Serilog;
```

Add the following code after the code where you added the customer repository to the services container.

```
// Configure logging to Console & File using Serilog
builder.Host.UseSerilog((ctx, lc) =>
{
    // Log to Console
    lc.WriteTo.Console();
    // Log to Rolling File
    lc.WriteTo.File("Logs/Log-.txt",
        rollingInterval: RollingInterval.Day);
});
```

Right mouse-click on the **AdvWorksAPI** Project folder and add a new folder named **Logs**.

Open the **LogTestController.cs** file and modify the return statement on the WriteMessages() and LogObject() methods

```
return "Check Console Window or Log File.";
```

## Try it Out

Run the application and click on the **GET /api/LogTest/LogObject** button.

Also click on the **GET /api/LogTest/WriteMessages** button.

Close the web browser.

Go back to Visual Studio, open the log file in the **Logs** folder to see the messages and the serialized Customer object.