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 Router & Write Messages

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

```
using AdvWorksAPI.BaseClasses;
using AdvWorksAPI.EntityLayer;

namespace AdvWorksAPI.RouterClasses;

public class LogTestRouter : RouterBase
{
  private readonly ILogger<LogTestRouter> _Logger;

  public LogTestRouter(ILogger<LogTestRouter> logger)
  {
    UrlFragment = "api/LogTest";
    TagName = "LogTest";
    _Logger = logger;
  }
}
```

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

```
protected virtual IResult 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.");

  return Results.Ok("Check Console Window");
}
```

Add the AddRoutes() method to map this WriteLogMessages as an endpoint.

```
/// <summary>
/// Add routes
/// </summary>
/// <param name="app">A WebApplication object</param>
public override void AddRoutes(WebApplication app)
{
  app.MapGet($"/{UrlFragment}/WriteMessages", () => 
WriteLogMessages())
      .WithTags(TagName)
      .Produces(200);
}
```

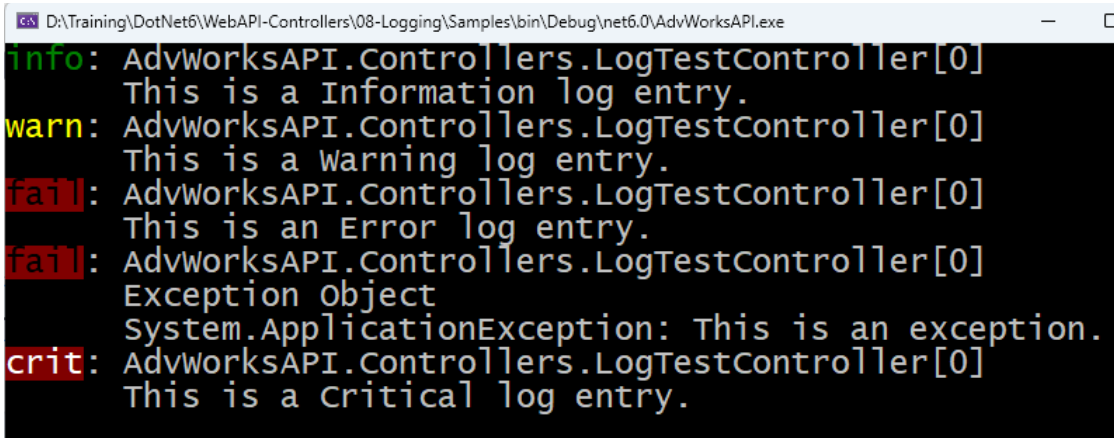
Open the **Program.cs** file and add the LogTestRouter class to DI

```
builder.Services.AddScoped<RouterBase, LogTestRouter>();
```

Try it Out

Run the application and click on the **GET /api/LogTest/WriteLogMessages** 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 show the messages in there as well

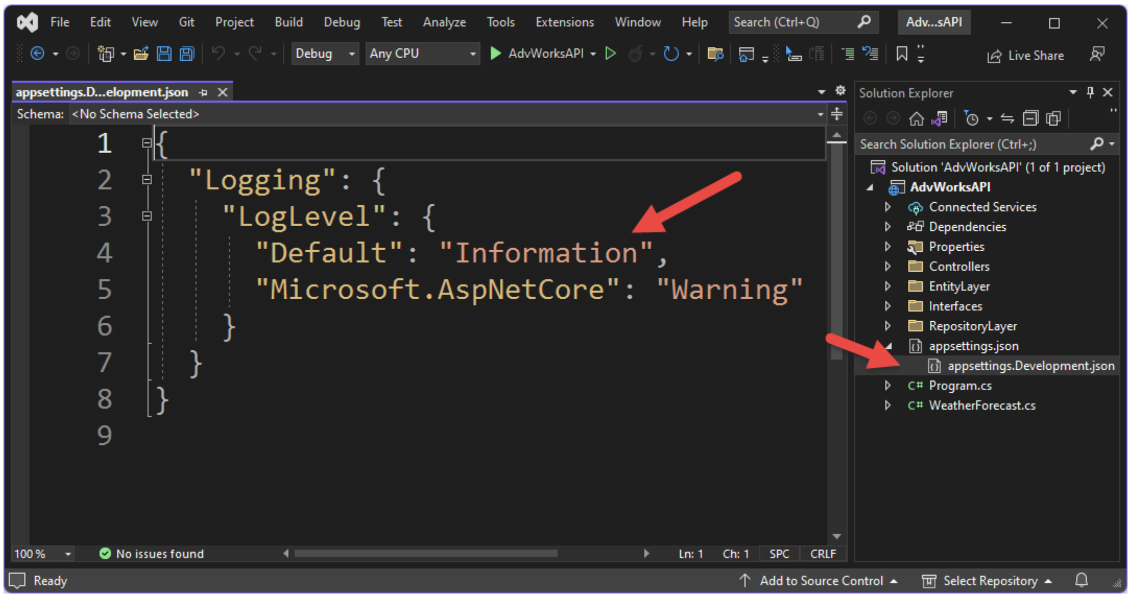
NOTE: Logging is synchronous only. Don't write directly to a slow data store such as SQL Server. Instead write to an in-memory queue and use a background worker or maybe a Windows Service to pull the messages out of the queue.

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



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

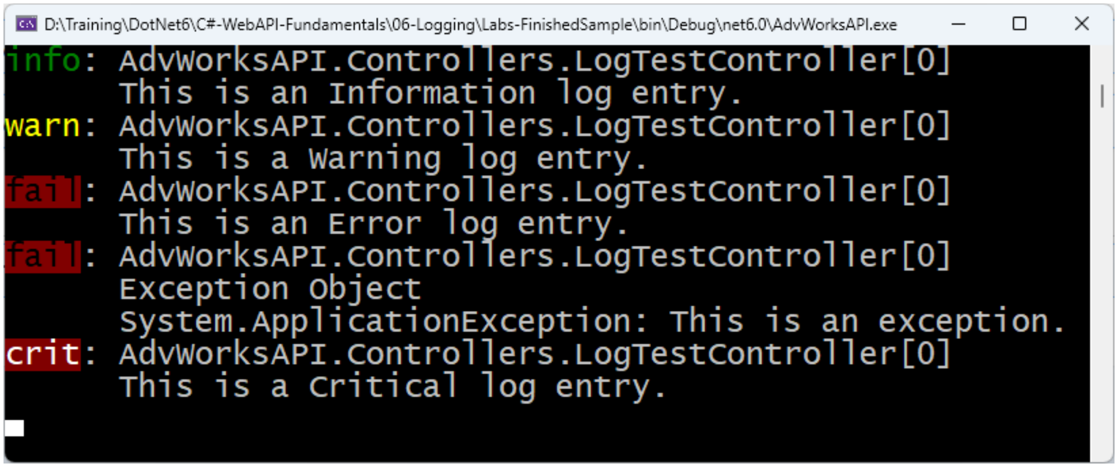
Set the "Default" property to "**Trace**"

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

Try it Out

Run the application and click on the **GET /api/LogTest/WriteLogMessages** 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/WriteLogMessages** 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 **LogTestRouter.cs** file and add a new method named **LogCustomer**().

```
protected virtual IResult LogCustomer()
{
  // Log a Customer 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);

  return Results.Ok("Check Console Window");
}
```

Add a new app.MapGet() to the AddRoutes() method

```
app.MapGet($"/{UrlFragment}/LogObject", () =>
LogCustomer())
       .WithTags(TagName)
       .Produces(200);
```

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 scoped objects

```
// 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 **LogTestRouter.cs** file and modify the return statement on the WriteLogMessages() and LogObject() methods.

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

Try it Out

Run the application.

Click on the **GET /api/LogTest/WriteLogMessages** button.

Click on the **GET /api/LogTest/LogObject** 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.