Introduction to Logging

In this lesson you will learn how logging works in ASP.NET Core and the different places where you can send logs from your microservices.

How Logging Works in ASP.NET Core

As you probably know, there will be interesting events happening all the time across your application lifecycle. And, as an application developer, you want to make sure you log these events so you can tell what's going on, especially when you need to troubleshoot issues.

In ASP.NET Core, any time your microservice needs to log any kind of information, it can make use of Logger objects, which you'll declare in your code via the ILogger interface. These logger objects are automatically registered for you during your application startup sequence, so you are free to inject them and use them in any of your microservice classes without any additional configuration.

You can use the many methods of the logger object to log events with different levels of severity, like informational events, warning events, error events and many others, according to the needs of your service.

By default, these events are sent to your console or terminal, which can tell you exactly what is going on with your microservice as you run it or debug it in your dev box. However, the Console is just one of the possible destinations of your logs, and is enabled by one of many Logging Providers, called the Console Logging Provider in this case.

There are several other logging providers that allow you to send logs to different places. For instance, .NET applications come with other built-in logging providers like the Event Source provider, which can collect ETW events that you can later explore in tools like PerfView, and the Event Log provider which, on Windows machines, can send logs to the Windows Event Log, which you can later see in the Windows Event Viewer.

There are also several third-party logging providers that can send logs to dozens of places, like the Seq provider, which sends logs to the Seq search and analysis server, the Application Insights provider, which sends logs to the Azure Application Insights service, and many other providers like elmah.io, Log4Net, NLog, Serilog, and others.

In this module you will start by learning how to send logs to your console, and later how to stand up and send logs to a Seq server, both in your box and in the Azure cloud.

Seq is a popular real-time search and analysis server for structured application log data that can detect and diagnose issues in your microservices. It's a great fit for this course because you can easily run it as a Docker container in your box or in any other box in your network, as well as in the cloud in your Azure Kubernetes Service cluster.

In the next lesson you will learn how to use the ILogger interface to start sending logs from one of your microservices to your console or terminal.