Adding OpenTelemetry

Start

There's plenty of tracing information that is currently being emitted in our microservices by the ASP.NET Core runtime and by MassTransit libraries. However, to make use of it we need to start collecting, aggregating, and exporting that tracing information. And, for that, we can take advantage of OpenTelemetry.

So, let's now add the OpenTelemetry libraries to our Trading microservice.

In Trading repo

1. Add the OpenTelemetry NuGet packages to Play.Trading.Service.csproj:

```
dotnet add package OpenTelemetry --version 1.2.0
dotnet add package OpenTelemetry.Extensions.Hosting --version 1.0.0-rc9.2
dotnet add package OpenTelemetry.Exporter.Console --version 1.2.0
dotnet add package OpenTelemetry.Instrumentation.AspNetCore --version 1.0.0-rc9.2
dotnet add package OpenTelemetry.Instrumentation.Http --version 1.0.0-rc9.2
```

2. Register OpenTelemetry in Startup.ConfigureServices:

```
public void ConfigureServices(IServiceCollection services)
{
  …
  services.AddSeqLogging(Configuration);

  services.AddOpenTelemetryTracing(builder =>
  {
    var serviceSettings = Configuration.GetSection(nameof(ServiceSettings))
                                       .Get<ServiceSettings>();
    builder.AddSource(serviceSettings.ServiceName)
           .AddSource("MassTransit")
           .SetResourceBuilder(
             ResourceBuilder.CreateDefault()
                     .AddService(serviceName: serviceSettings.ServiceName))
           .AddHttpClientInstrumentation()
           .AddAspNetCoreInstrumentation()
           .AddConsoleExporter();
  });
```

3. Run all microservices

4. Try a purchase

5. Notice Trading telemetry in console logs

6. Commit and push

In the next lesson you will learn how to run a Jaeger server in your box via Docker.