Nitesh Singhal Follow
Mar 14 · 4 min read · Listen

Save

TRACING AND MONITORING

OpenTelemetry with Jaeger in .NET Core

A step by step guide to integrate OpenTelemetry with ASP.Net core and visualize in Jaeger

Image by Nitesh Singhal

OpenTelemetry is a collection of tools, APIs, and SDKs. Use it to instrument, generate, collect, and export telemetry data (metrics, logs, and traces) to help you analyze your software's performance and behavior.

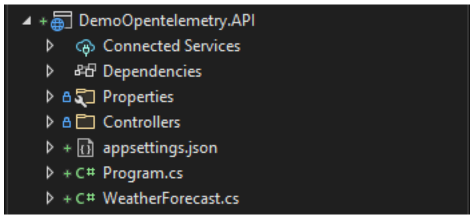
In this tutorial, we will look at how it can be integrated with ASP.NET core web project.

We will use Jaeger tool to analyze and visualize the telemetry data.

Let's get started..

🎁 **Subscribe for Nitesh Singhal's Articles**
🎁 Subscribe for Nitesh Singhal's Articles Subscribe to Nitesh Singhal's Articles to get tutorials, tips, and best...
medium.com

Let's setup API project first.



```
DemoOpentelemetry.API
    Connected Services
    Dependencies
    Properties
    Controllers
    appsettings.json
    Program.cs
    WeatherForecast.cs
```

API Project

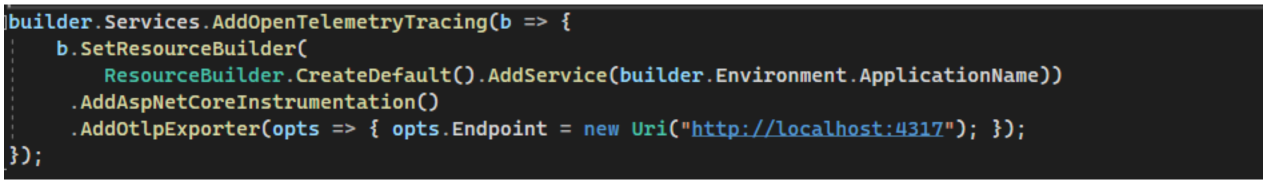
add the required nuget packages

```
<PackageReference Include="OpenTelemetry.Exporter.OpenTelemetryProtocol" Version="1.2.0-rc3" />

<PackageReference Include="OpenTelemetry.Extensions.Hosting" Version="1.0.0-rc9" />

<PackageReference Include="OpenTelemetry.Instrumentation.AspNetCore" Version="1.0.0-rc9" />
```

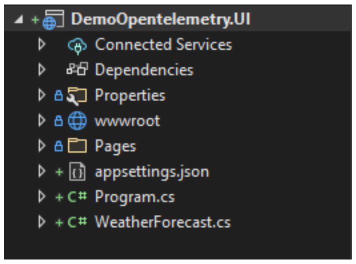
Now open Program.cs and add the following code



```
builder.Services.AddOpenTelemetryTracing(b => {
    b.SetResourceBuilder(
        ResourceBuilder.CreateDefault().AddService(builder.Environment.ApplicationName))
     .AddAspNetCoreInstrumentation()
     .AddOtlpExporter(opts => { opts.Endpoint = new Uri("http://localhost:4317"); });
});
```

here we are setting up Jaeger's URI as OLTP exporter endpoint.

now Let's setup web app.



```
DemoOpentelemetry.UI
    Connected Services
    Dependencies
    Properties
    wwwroot
    Pages
    appsettings.json
    Program.cs
    WeatherForecast.cs
```

Web App Project

and also add required nuget packages

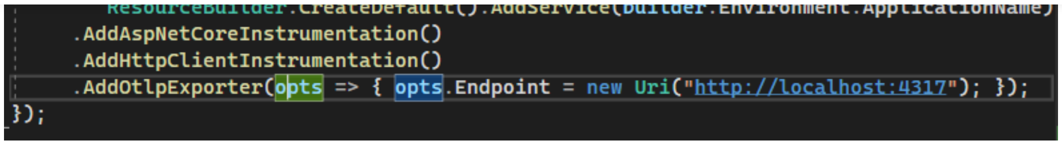
```
<PackageReference Include="OpenTelemetry.Exporter.OpenTelemetryProtocol" Version="1.2.0-rc3" />

<PackageReference Include="OpenTelemetry.Extensions.Hosting" Version="1.0.0-rc9" />

<PackageReference Include="OpenTelemetry.Instrumentation.AspNetCore" Version="1.0.0-rc9" />

<PackageReference Include="OpenTelemetry.Instrumentation.Http" Version="1.0.0-rc9" />
```

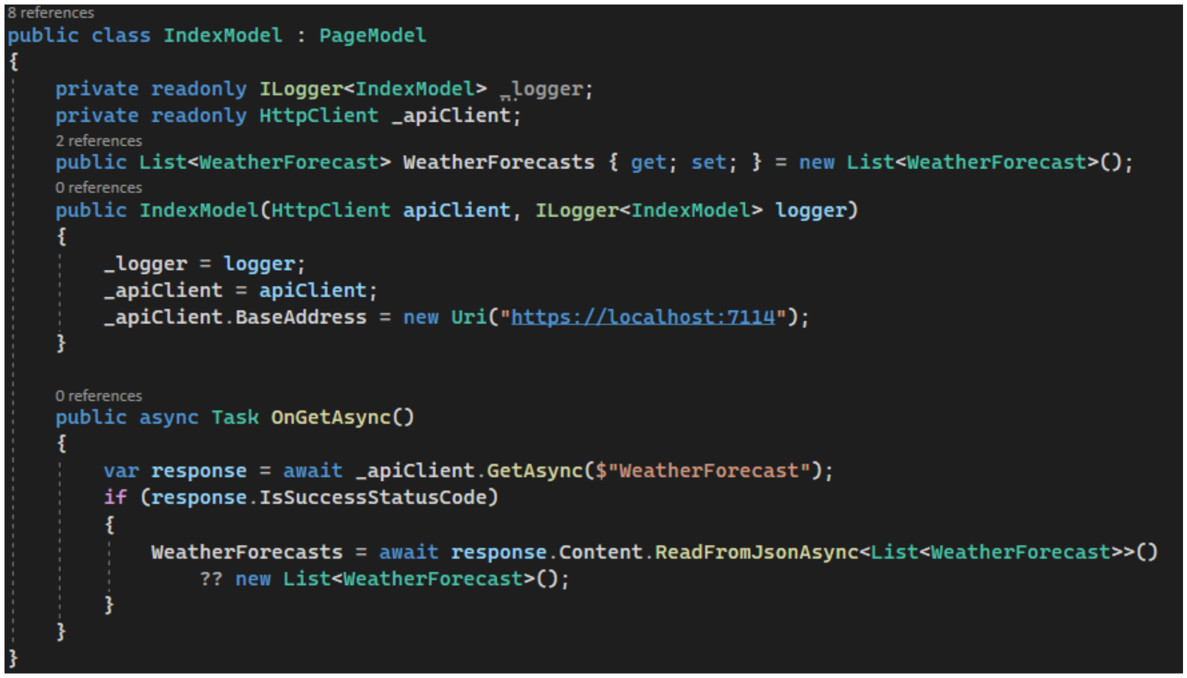
Open Program.cs and add the following code



```
        ResourceBuilder.CreateDefault().AddService(builder.Environment.ApplicationName)
    .AddAspNetCoreInstrumentation()
    .AddHttpClientInstrumentation()
    .AddOtlpExporter(opts => { opts.Endpoint = new Uri("http://localhost:4317"); });
});
```

we will need to make one more change, call the API and show the result on UI.

Open the Index.chtml.cs file and add the following code

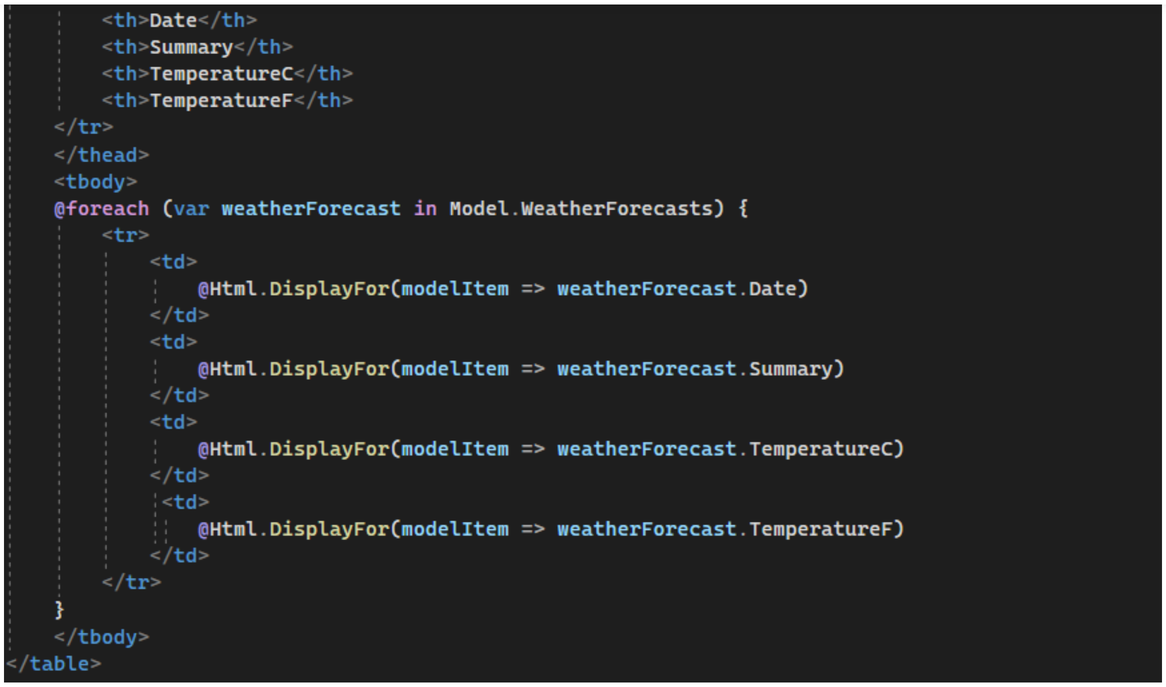


```
8 references
public class IndexModel : PageModel
{
    private readonly ILogger<IndexModel> _logger;
    private readonly HttpClient _apiClient;
    2 references
    public List<WeatherForecast> WeatherForecasts { get; set; } = new List<WeatherForecast>();
    0 references
    public IndexModel(HttpClient apiClient, ILogger<IndexModel> logger)
    {
        _logger = logger;
        _apiClient = apiClient;
        _apiClient.BaseAddress = new Uri("https://localhost:7114");
    }

    0 references
    public async Task OnGetAsync()
    {
        var response = await _apiClient.GetAsync($"WeatherForecast");
        if (response.IsSuccessStatusCode)
        {
            WeatherForecasts = await response.Content.ReadFromJsonAsync<List<WeatherForecast>>()
                ?? new List<WeatherForecast>();
        }
    }
}
```

Index.chtml.cs

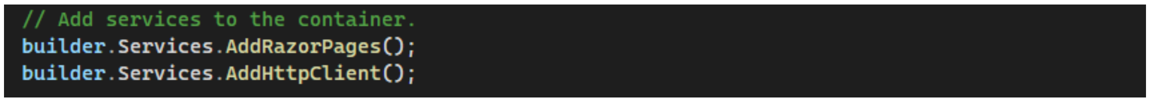
and Open Index.html and add the following table



```
        <th>Date</th>
        <th>Summary</th>
        <th>TemperatureC</th>
        <th>TemperatureF</th>
    </tr>
    </thead>
    <tbody>
    @foreach (var weatherForecast in Model.WeatherForecasts) {
        <tr>
            <td>
                @Html.DisplayFor(modelItem => weatherForecast.Date)
            </td>
            <td>
                @Html.DisplayFor(modelItem => weatherForecast.Summary)
            </td>
            <td>
                @Html.DisplayFor(modelItem => weatherForecast.TemperatureC)
            </td>
            <td>
                @Html.DisplayFor(modelItem => weatherForecast.TemperatureF)
            </td>
        </tr>
    }
    </tbody>
</table>
```

Index.html

add the following code in Program.cs



```
// Add services to the container.
builder.Services.AddRazorPages();
builder.Services.AddHttpClient();
```

Program.cs

we are ready to run the app.

Build both project and make sure they build with any errors.

Now let's setup Jaeger

Jaeger Setup

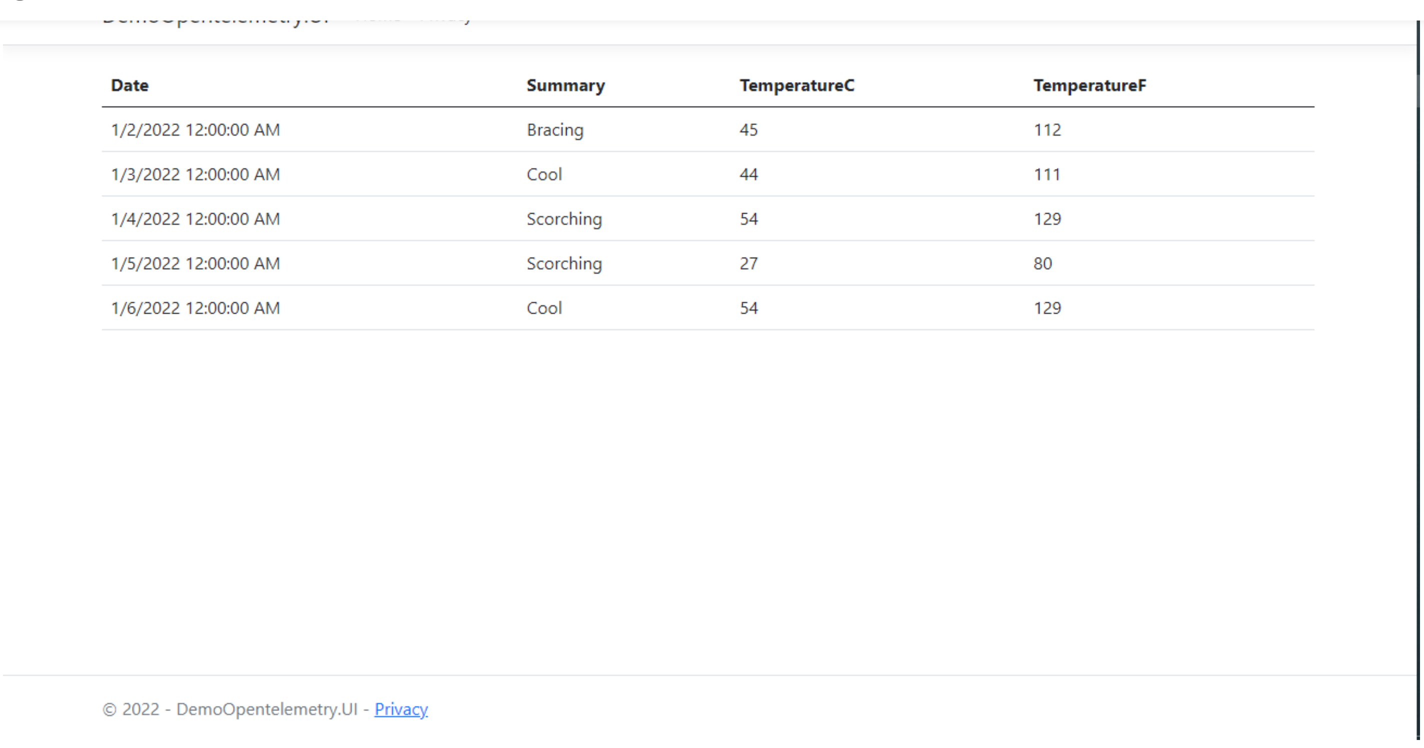
I am using docker setup.

use the following cmd to create a docker container running for Jaeger

```
docker run --name jaeger -p 13133:13133 -p 16686:16686 -p 4317:4317 -d --restart=unless-stopped
jaegertracing/opentelemetry-all-in-one
```

To verify if it running successfully, Go to browser and open UI at http://localhost:16686

Execution



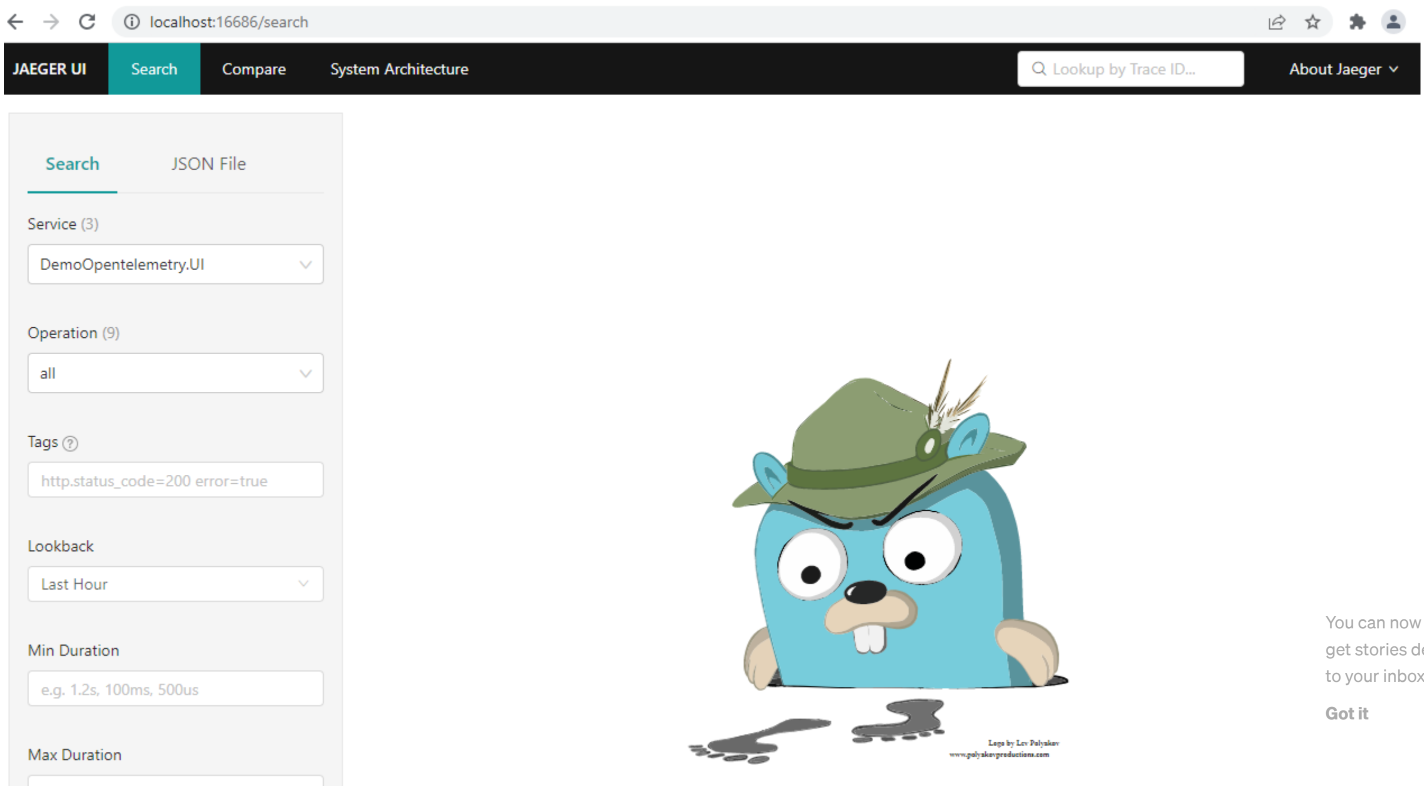
Date	Summary	TemperatureC	TemperatureF
1/2/2022 12:00:00 AM	Bracing	45	112
1/3/2022 12:00:00 AM	Cool	44	111
1/4/2022 12:00:00 AM	Scorching	54	129
1/5/2022 12:00:00 AM	Scorching	27	80
1/6/2022 12:00:00 AM	Cool	54	129

© 2022 - DemoOpentelemetry.UI - Privacy

Web App UI

We can see the list of weather forecast on UI. which is being fetched using API project and displayed on UI.

Now Let's look at the Jaeger UI.



localhost:16686/search

Service (3): DemoOpentelemetry.UI

Operation (9): all

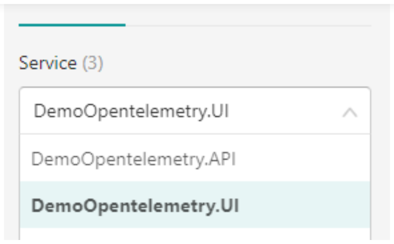
Tags: http.status_code=200 error=true

Lookback: Last Hour

Min Duration: e.g. 1.2s, 100ms, 500us

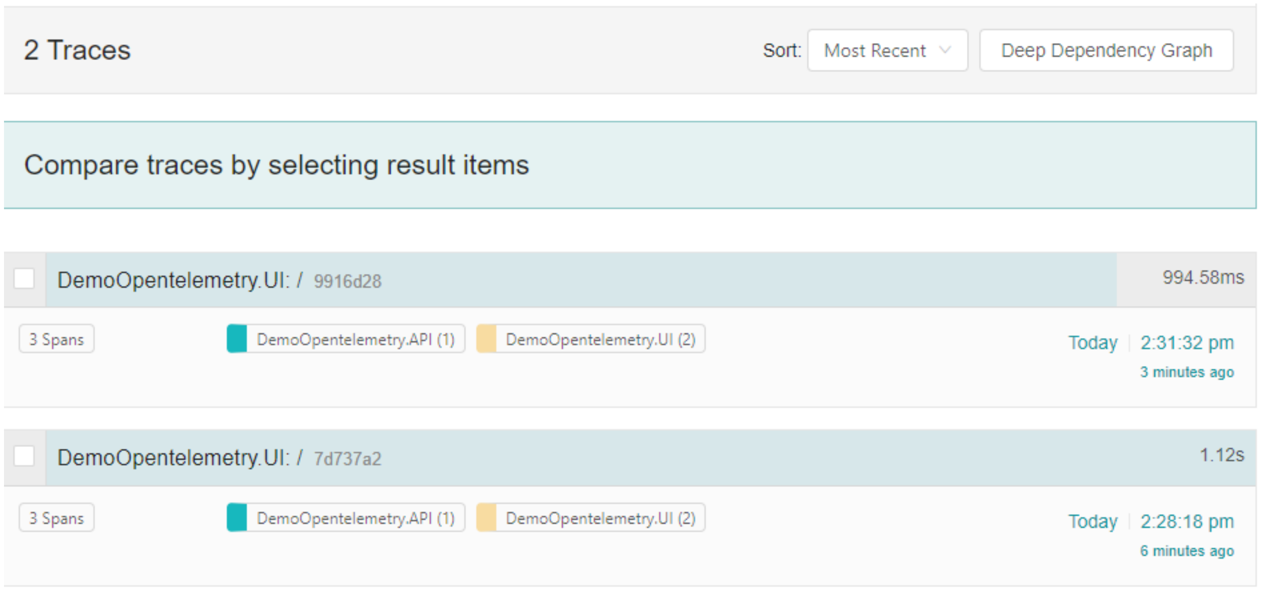
Max Duration

Jaeger UI



We can see both API and UI project is listed.

We can select operation from Operation drop down and Click on find trace.

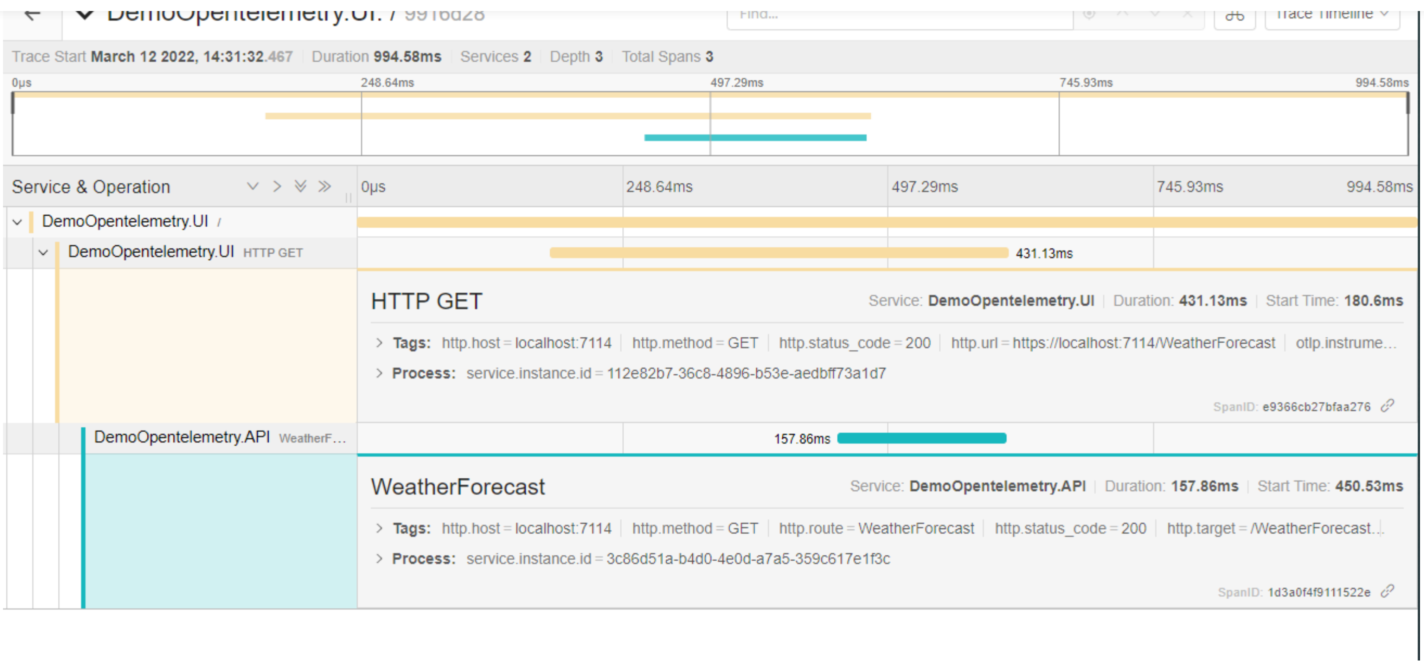


DemoOpentelemetry.UI: / 9916d28 994.58ms
3 Spans DemoOpentelemetry.API (1) DemoOpentelemetry.UI (2) Today 2:31:32 pm 3 minutes ago

DemoOpentelemetry.UI: / 7d737a2 1.12s
3 Spans DemoOpentelemetry.API (1) DemoOpentelemetry.UI (2) Today 2:28:18 pm 6 minutes ago

Trace

We can see multiple trace result, we select one for further analysis



Trace Start **March 12 2022, 14:31:32**.467 | Duration **994.58ms** | Services **2** | Depth **3** | Total Spans **3**

0µs | 248.64ms | 497.29ms | 745.93ms | 994.58ms

```
Service & Operation                        0µs    248.64ms    497.29ms    745.93ms    994.58ms
DemoOpentelemetry.UI /
  DemoOpentelemetry.UI HTTP GET                                431.13ms
```

HTTP GET — Service: **DemoOpentelemetry.UI** | Duration: **431.13ms** | Start Time: **180.6ms**

> **Tags:** http.host = localhost:7114 | http.method = GET | http.status_code = 200 | http.url = https://localhost:7114/WeatherForecast | otlp.instrume...
>
> **Process:** service.instance.id = 112e82b7-36c8-4896-b53e-aedbff73a1d7
>
> SpanID: e9366cb27bfaa276

```
    DemoOpentelemetry.API WeatherF...               157.86ms
```

WeatherForecast — Service: **DemoOpentelemetry.API** | Duration: **157.86ms** | Start Time: **450.53ms**

> **Tags:** http.host = localhost:7114 | http.method = GET | http.route = WeatherForecast | http.status_code = 200 | http.target = /WeatherForecast...
>
> **Process:** service.instance.id = 3c86d51a-b4d0-4e0d-a7a5-359c617e1f3c
>
> SpanID: 1d3a0f4f9111522e

Trace detail

We can see the whole trace chain from UI to API with duration and start time.

Demo code at Github

Summary

It is very easy to setup Opentelemetry in .NET application. although I have use jaeger tool for analysis but there are many other tools are available and if they all follow same standard then they can also be used without any other changes.

Hope is it helpful..

Happy Coding and Keep learning..!

🎁 Subscribe for Nitesh Singhal's Articles

to get tutorials, tips, and best practices directly to your email inbox.

Subscribe

Emails will be sent to myndfire@gmail.com.
Not you?