Introduction to IdentityServer

In this lesson we will learn about IdentityServer, which is the OpenID Provider we will be using across this course.

Certified OpenID Providers

Like I mentioned in the previous lesson, there are dozens of certified OpenID providers you can choose from these days, which can come from organizations mostly focused on providing identity related services like Auth0 and Okta, to giant software vendors like Google and Microsoft, which offer identity services as part of their cloud offerings.

You can certainly choose any of these providers for your microservices architecture, but our choice for this course will be IdentityServer given the unique benefits it provides to .NET based apps and services.

Introducing IdentityServer

So what exactly is IdentityServer? Well in the words of its creators, IdentityServer is an open-source, standards-compliant, and flexible OpenID Connect and OAuth 2.0 framework for ASP.NET Core.

Or in other words, IdentityServer is the framework that extends ASP.NET Core to provide OpenID Connect and OAuth 2.0 features to your services.

Why IdentityServer?

But why would we use IdentityServer instead of any of the other dozens of OpenID providers out there? Well, here for a few reasons:

- **Built for ASP.NET Core.** IdentityServer was specifically built to leverage and extend the capabilities of ASP.NET Core, like the built in ASP.NET Identity integration, making it an ideal choice for our .NET based microservices.
- **Local box experience.** You don't need to register or reach out to any cloud service to try out IdentityServer, especially for learning purposes. It can run fully in your local box and you can later containerize it to deploy it to your favorite cloud if needed.
- **Highly customizable.** There are dozens of ways to customize IdentityServer according to your needs, all of them taking advantage of both C# and the ASP.NET Configuration system.
- **Mature open source.** Not only the project is open source, which by itself is a huge benefit for us developers, but also uses the Apache 2 license which allows building commercial products on top of it. Plus, it is also part of the .NET Foundation, which brings in all the support from the .NET open-source ecosystem.
- **Broad open-source community.** The big .NET community around IdentityServer is of invaluable help if you have questions or find any issues as you start adapting it for your scenarios.
- **OpenID certified.** Like mentioned before, IdentityServer is OpenID certified, which ensures it offers all the features expected of a solid OpenID provider.

The Identity microservice revisited

Let's now take another look at our Identity microservice to understand how IdentityServer fits alongside the already implemented components.

As you know, we have already taken advantage of the built-in ASP.NET Core Identity functionality to enable the new user registration. We also introduced our Users REST API to enable a few basic user management scenarios. This REST API in turn also uses the services provided by ASP.NET Core Identity to simplify how we interact with the Users database collection.

We will now integrate IdentityServer into this same microservice via the middleware components provided in publicly available NuGet packages and which will effectively turn our microservice into an authorization server and an OpenID provider. This middleware will also be integrated to ASP.NET Core Identity so that it can work with our existing Users collection, our upcoming Roles collection and a few other built-in components.

With this our client will be able to send authentication requests to our Identity microservice, upon which the user will be redirected to the sign in page provided by the ASP.NET Core Identity integration. Then, after successfully authenticating the user, an access token, and optionally an ID token will be returned to the client.

Finally, the client will be able to use the received access token to talk to our Catalog and Inventory microservices, and even to the Users REST API in the Identity microservice, which will all now be protected to no longer allow unauthorized access.

In the next module you will learn how to properly secure your microservices in detail by using the multiple security-oriented features provided by ASP.NET Core and IdentityServer.