Storing secrets during local development

In this lesson we will learn about the .NET Secret Manager and how you can use it to safely store secrets during your microservices local development.

The .NET Secret Manager

As you know, so far we have been trying to keep all configuration related data outside of our C# files and into the appsettings.json and appsettings.Development.json files, and we are doing this because we know that these files are just two of the possible .NET configuration sources that can easily be replaced at runtime, without touching the code base, depending on the environment where our microservice is running.

Soon our Identity microservice will need to provision an admin user as part of the service boot sequence, and in order to create that user it's going to need some configurable details like the admin user email, which we will certainly store in the appsettings.json file. And, as you may guess, alongside that email we are also going to need a password to properly identify this user.

We could go ahead and also add this password to our appsettings.json file, but that is not quite a good idea. This is because eventually you will likely push all your microservice code, including appsettings.json, into some repository in a source control system like Git. If that repository is public, you would be exposing the password to the world and even if it is private there is always the chance that the source control system gets compromised and your passwords can leak.

Luckily, there is another kind of configuration source that you can use for storing secrets, at least for local development, and that is the user secrets configuration source, which stores its contents in a file called secrets.json. This file is completely separate from your code base. In fact, it lives in your local machine user profile.

You could add values manually there, but you don't have to. In fact, you don't even need to know where this file lives because there's a built-in .NET tool designed to deal with this file: the Secret Manager. You can use the Secret Manager to create, update or remove secrets in this file while at the same time getting the secrets associated to your microservice projects.

So we will use the Secret Manager to create and store our AdminPassword and ASP.NET Core will take care of loading it, along with the AdminEmail from appsettings.json, during our microservice boot sequence, after which it will also merge the values from both configuration sources into the Configuration object we have been using in our Startup class. From our microservice code point of view, the data just shows up in the Configuration object, so it doesn't need to know where the different values came from, which is a very powerful concept.

We can then introduce a proper class to represent these admin user settings, which we will call IdentitySettings, and, just as we have done before, we will read and deserialize the values from Configuration into IdentitySettings in our Startup ConfigureServices method. Then, later in the boot

sequence, we will use IdentitySettings to create the Admin user via the ASP.NET Core Identity UserManager object.

So, by using the .NET Secret Manager:

- We are making sure that the microservice secrets remain outside of the source code, which will prevent them from making their way into our source control system
- We don't make our code base any more complicated since the microservice code doesn't need to know where the secrets really live
- And ASP.NET Core loads the user secrets configuration source by default
- But also keep in mind that the Secret Manager is only meant for local development scenarios. So don't expect the secrets you put in there will be available to the microservice once you deploy it outside of your dev box. Also, don't place Production level secrets in Secret Manager since the data is stored in plain text in th secrets.json file.

In the next lesson you will learn about the need for authorization checks in your microservices and the different ways ASP.NET Core allows you to perform authorization.