Reusing common code via Nuget

In this lesson we will learn about the need to reuse common code across microservices and how the NuGet package manager can be used to package and share code across services in a straightforward way.

Reusing common code

One of the things you will quickly realize as you start building more than one microservice is how much code starts getting repeated between each service implementation.

For instance, our Catalog microservice currently has a repository implementation for querying and storing data in its MongoDB database. It also has a couple of settings classes to more easily use the different service configurations that are stored in our appsettings.json file. Eventually we will also add classes to interact with our service broker, code to add instrumentation to the services and likely some more code that is not directly related to the purpose of the microservice.

The problem with this is that as soon as we bring in our second microservice, the Inventory service, we would likely need to copy to it that same code, with very little modifications. This is time consuming and goes against the Don't Repeat Yourself principle, which states that "every piece of knowledge must have a single, unambiguous, authoritative representation within a system", which in our case means that our common code should live and be maintained in a single place.

Therefore, we could be tempted to keep our common code in one of our microservices and have all the other services simply reference that project to get access to the common code. However, this is not a good idea because microservices should be independent of each other to ensure each of them can evolve quickly and have no ties of the internal implementation of others.

We can solve this problem by introducing a new library project that we will call the Common library and then we can extract all the common code into it. The Common library becomes the single place where we have all the code that is not specific to any microservice and therefore is the single place where we will perform any updates to it when needed.

But then, how can we make the code in this new Common library available to our microservices? We could have our microservices add a direct reference on the library project file, but that is not a good idea because as new members join our team each of them might want to work on one microservice or the other, or even in the Common library, and therefore they will only grab the code base for the project they are interested in, but not the code bases of all projects all the time. Plus, eventually each microservice and the Common library should get their own independent source control repository where they can be tracked and built independently.

So, if the microservice projects can't access the Common project directly, what can they do? Here is where we can use NuGet, which is the package manager for .NET. With NuGet we can execute a simple command like *dotnet pack* to bundle all the output files from the Common project into a NuGet package.

A NuGet package is nothing more than a zip file with the nupkg extension that contains all the files that are to be shared with other projects. Now each of our microservices can reference the NuGet package to get access to everything that the common library has to offer. And the good thing is that microservice projects don’t need to know where this NuGet packages are hosted. We will initially place the package in our local machine but eventually you will likely want to host them in a cloud based location either accessible just by your team or by anyone on the web. The important thing is that regardless of the location you won’t have to change how the microservices access the common code.

With our new Common NuGet package our common code is now maintained in a single place and the time to build new microservices is significantly reduced.

In the next lesson we will extract our current common code into a new Common library, and we will produce our first NuGet package that the Catalog microservice and every future service will be able to reference to get quick access to the common code.