Creating a microservice via the .NET CLI

In this lesson we will create our first microservice, the Catalog microservice, using the .NET CLI and Visual Studio Code.

(Demo prep)

- Delete %userprofile%/.dotnet
- Remove the ASP.NET Core HTTPS dev certificate from Trusted Root Certification Authorities

Creating the project

To create our microservice project we will take advantage of the .NET command line interface, also known as .NET CLI, which comes installed with the .NET SDK.

1. In a Terminal, switch to D:\projects

2. md play.catalog

3. Open folder in VS Code

4. Crate **src** directory

5. Right click on src and **Open Terminal**

6. dotnet new webapi -n Play.Catalog.Service

   Now, if by any chance you have multiple versions of the .NET SDK installed in your machine, and you are not sure that .NET 5 is your default version, please add the --framework argument to the dotnet new command, specifying the value of net5.0 for .NET 5.

   That will make sure that you are using the ASP.NET Core 5 templates when creating projects, which will produce the same files that you will see in this and in future lessons.

7. Close terminal

8. Explain:
   - Csproj (declares how to build our application, so in this case we will be using the typical tasks and tools to build web projects)
   - Program.cs (the main entry point)
       - Add VS Code missing artifacts
       - CreateDefaultBuilder: default values for the hosting process, including reading appsettings.json

    - ConfigureWebHostDefaults: defaults for asp.net core app, including the use of the Kestrel web server
- launchSettings.json → applicationUrl

9. Open Terminal (View → Terminal)

10. Switch to src\Play.Catalog.Service

11. Build via **dotnet build**

12. Build via **Terminal → Run Build Task**

13. Add to **tasks.json** (type it):

    ```
    "group": {
       "kind": "build",
       "isDefault": true
    }
    ```

14. Build via **CTRL + SHIFT + B**

15. Run the service via **dotnet run**

16. Run the service via the **Run View**

17. Explain the cert trust issue and then:
    dotnet dev-certs https --trust

18. Run again and go to **/swagger**

19. Close browser and stop server

In the next lesson we will define the REST API for our Catalog microservice.