# Identity

## (Demo prep)

- Delete Identity database
- Ensure Catalog DB has only these:
```
[
  {
    "name": "Potion",
    "description": "Restores some HP",
    "price": 5
  },
  {
    "name": "Antidote",
    "description": "Cures poison",
    "price": 7
  }
]
```
- Ensure catalogitems collection in Inventory DB is in sync with Catalog
- Disable "Save and fill personal info" in browser
- Hide favorites bar
- Disable extensions in browser
- dotnet tool uninstall -g dotnet-aspnet-codegenerator
- Delete packages from %userprofile%\.nuget
- Remove Play.Identity.Service Postman collection
- Apply VS Code and PowerShell environment Settings

## Creating the Identity microservice

It's time to create our Identity microservice, and for this I have already opened a new instance of Visual Studio Code and I have opened my terminal in my projects directory which is where we have stored all the files related to the Play Economy system so far.

As a quick recap, here's what we have in each directory so far:

- Play.Catalog, contains all the source code for the Catalog microservice
- While Play.Inventory includes all the code for the Inventory microservice
- Play.Common includes our shared class library with all the generic code used across microservices
- Packages contains all the NuGet packages we have created so far
- And, Play.Infra contains the docker-compose file we use to stand up our infrastructure services

Let's now go ahead an create a new directory for our new Identity microservice

1. md Play.Identity

2. CD Play.Identity

3. code . -r

4. Create **src** directory

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

6. dotnet new webapi -n Play.Identity.Service -f net5.0

7. Select any C# file so OmniSharp starts and prompts for adding files for build/run

8. Let’s make a few small changes to these VS Code generated files so that we can more easily iterate on this microservice.

9. Add to **tasks.json** (type it):
   ```
   "group": {
      "kind": "build",
      "isDefault": true
   }
   ```

10. Remove **serverReadyAction** from launch.json.

11. Update ports in launchSettings.json:

    "applicationUrl": "https://localhost:**5003**;http://localhost:**5002**",

12. Delete WeatherForecastController and WeatherForecast

13. dotnet tool install -g dotnet-aspnet-codegenerator --version 5.0.2

14. Install NuGet packages:
    ```
    dotnet add package Microsoft.VisualStudio.Web.CodeGeneration.Design --version 5.0.2
    dotnet add package Microsoft.AspNetCore.Identity.UI --version 5.0.15
    dotnet add package Microsoft.EntityFrameworkCore.Design --version 5.0.15
    dotnet add package Microsoft.EntityFrameworkCore.SqlServer --version 5.0.15
    ```

15. dotnet aspnet-codegenerator identity --files "Account.Register"

16. Explore added directories and files

17. Delete Areas\Identity\Data directory

18. Delete Areas\Identity\IdentityHostingStartup.cs

19. Remove EntityFramework package references from csproj

20. Build the project

At this point we have a new Identity microservice that is already integrated with a few basic ASP.NET Core Identity building blocks, most of it generated by the .NET CLI and the ASP.NET Core code generator.

In the next lesson we will expand our ASP.NET Core Identity integration by using MongoDB as its backing store and we will enable both registering new users and letting them login to the system.