Reusing common code via Nuget

(Demo prep)

- Remove package sources:
  dotnet nuget remove source "PlayEconomy"
- Reopen Postman

Extracting into a class library

It's time to move the common code we refactored from our Catalog microservice to a new shared class library that can be used by both Catalog and any future microservice. So, to start with I have opened here a new Visual Studio Code instance.

1. Open new VS Code instance

2. md Play.Common

3. Open Play.Common dir in VS Code

4. Create src dir

5. Add the Common project:

   dotnet new classlib -n Play.Common

6. View → Command Palette → .NET: Generate assets for build and debug

7. Add to tasks.json:

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

8. Delete Class1.cs

9. dotnet add package MongoDB.Driver

10. dotnet add package Microsoft.Extensions.Configuration

11. dotnet add package Microsoft.Extensions.Configuration.Binder

12. dotnet add package Microsoft.Extensions.DependencyInjection

13. Add Settings directory

14. Add to Settings dir the settings.cs files from Catalog

15. Fix namespaces to Play.Common.Settings

16. Add to root dir IEntity.cs and IRepository.cs from Catalog

17. Fix namespaces to Play.Common

18. Add a MongoDB dir

19. Add to MongoDB dir Extensions.cs, and MongoRepository.cs from Catalog

20. Fix namespaces to Play.Common.MongoDB

And with that we should be ready to build our project and produce our NuGet package. But before that let's also add a couple of new functions to MongoRepository that our future microservices will use to query for entities based on a filter.

21. Add to IRepository:

```
Task<IReadOnlyCollection<T>> GetAllAsync(Expression<Func<T, bool>> filter);
Task<T> GetAsync(Expression<Func<T, bool>> filter);
```

22. Add to MongoRepository:

```
public async Task<IReadOnlyCollection<T>> GetAllAsync(Expression<Func<T, bool>> filter)
{
   return await dbCollection.Find(filter).ToListAsync();
}

public async Task<T> GetAsync(Expression<Func<T, bool>> filter)
{
   return await dbCollection.Find(filter).FirstOrDefaultAsync();
}
```

23. CTRL + SHIFT + B to ensure everything builds

24. Create the nuget package:

```
dotnet pack -o ..\..\..\packages\
```

BACK IN CATALOG PROJECT

25. dotnet nuget add source D:\projects\packages -n PlayEconomy

26. Remove MongoDB.Driver package referernce

27. dotnet add package Play.Common

28. Remove IEntity.cs, Repositories and Settings dirs

29. Fix imports

30. Run Catalog service and try queries

In the next lesson we will learn about Docker compose and how it can help us simplify the way we configure and start our infrastructure services.