Code Academy

Lecturer

Rokas Slaboševičius

Multi-project structure

Data

Today you will learn

01 How to split a solution into several parts

What is it?

As we develop ever larger solutions, the problem is how to keep everything in line.

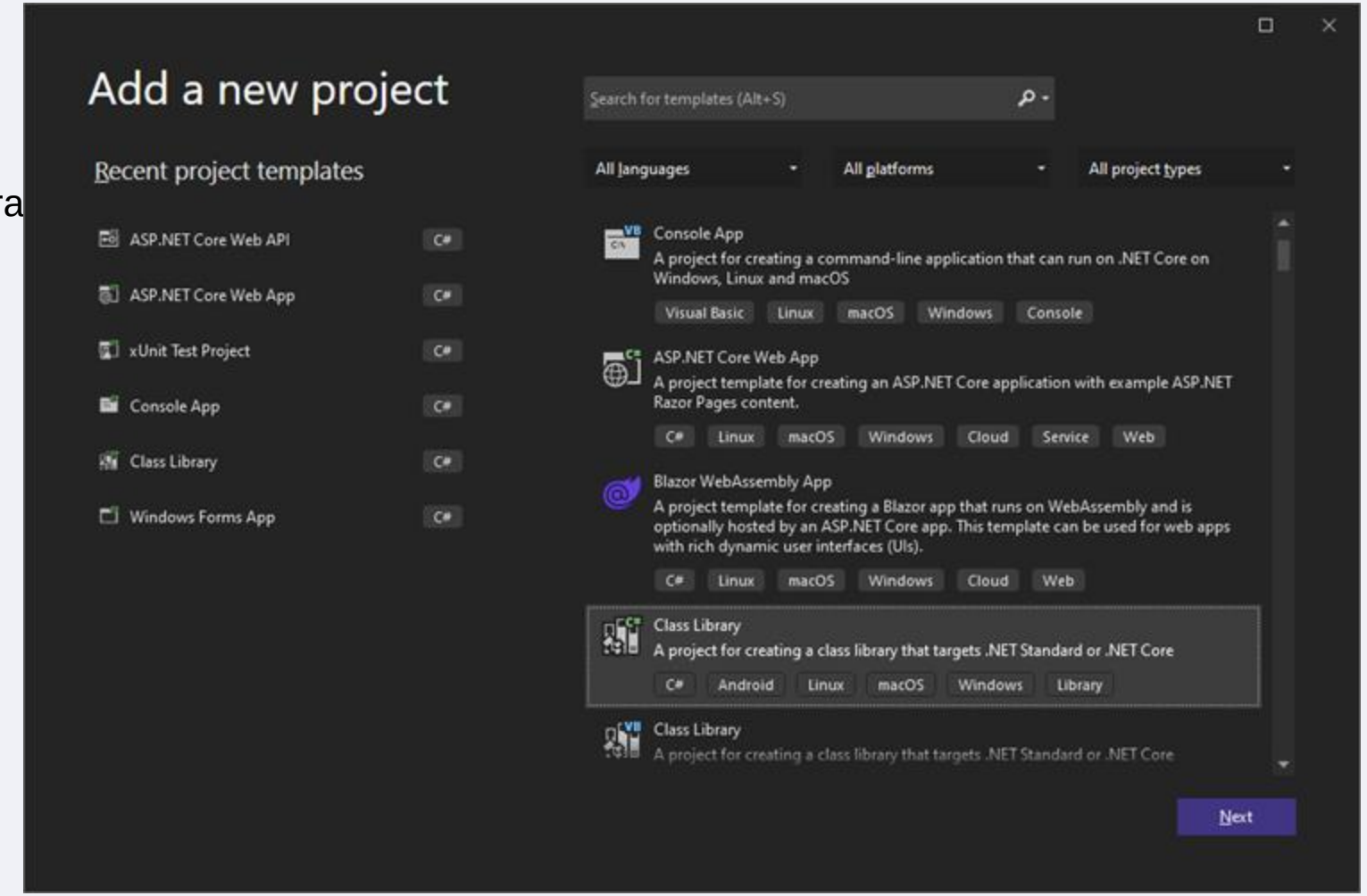
This allows you to have more than one project per solution.

The most popular way of structuring a project is called Domain Driven Design (DDD), but it is complex and could be the subject of its own module.

We will split the solution into Api, Database and BusinessLogic

The process

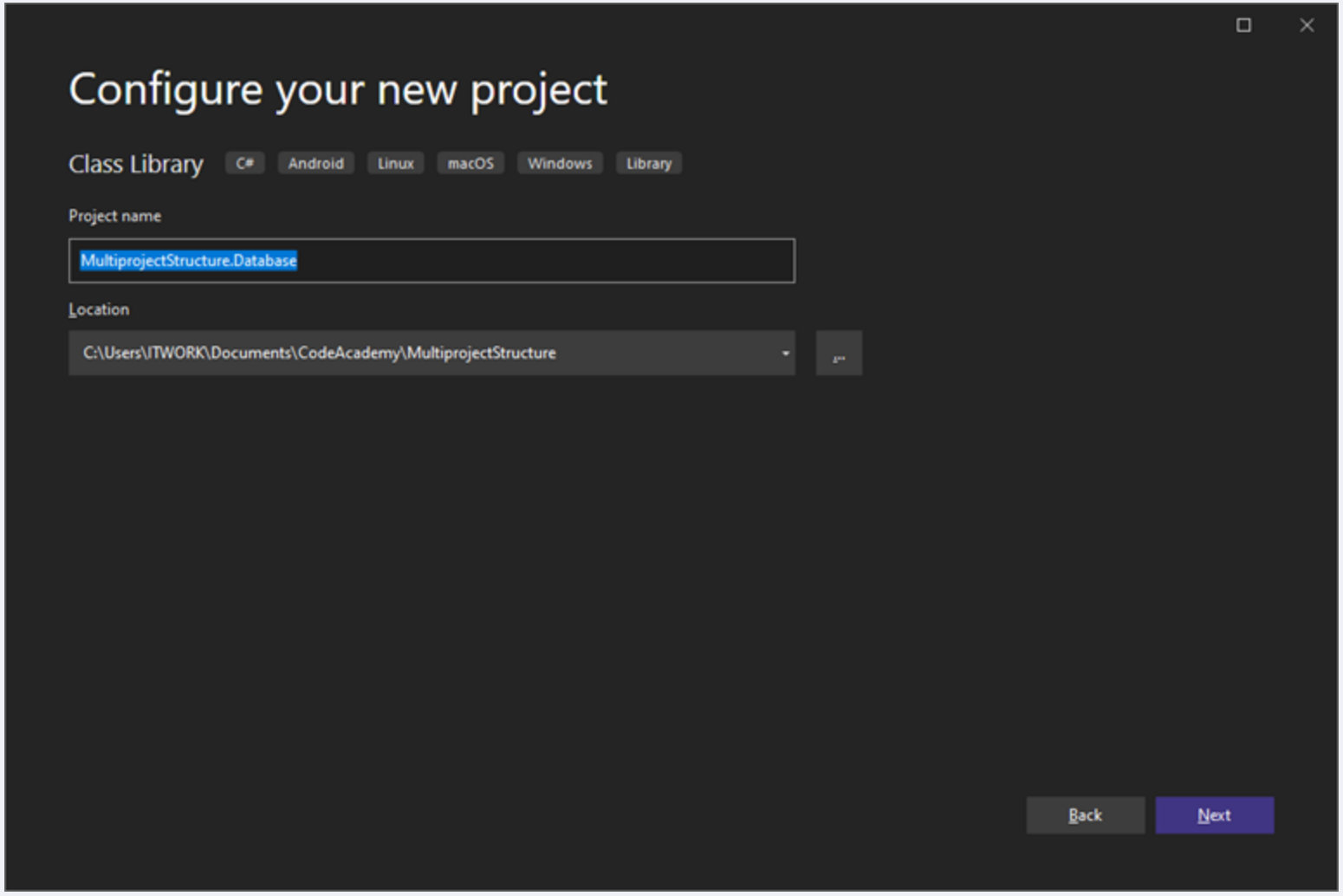
We will create a new project as Class Libra



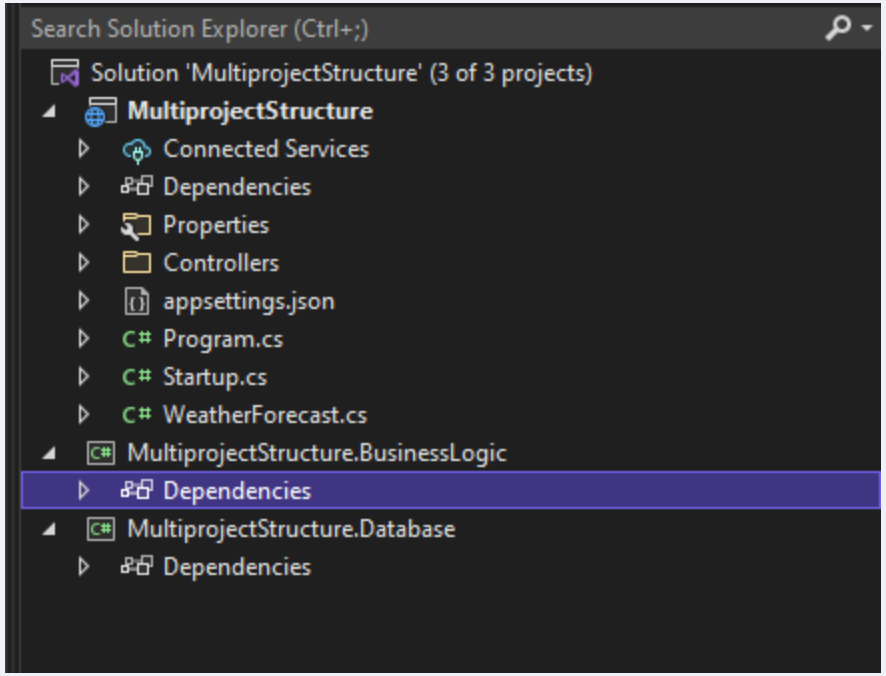
The process

Name is Solution name

point library name

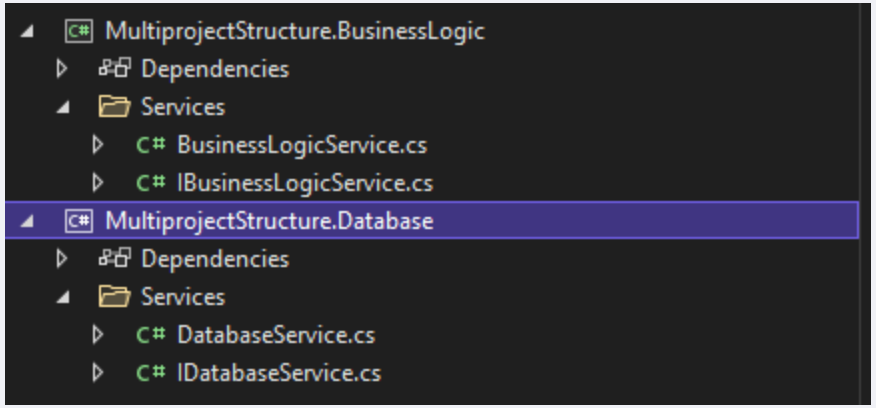


The process


Search Solution Explorer (Ctrl+;)
Solution 'MultiprojectStructure' (3 of 3 projects)
MultiprojectStructure
Connected Services
Dependencies
Properties
Controllers
appsettings.json
Program.cs
Startup.cs
WeatherForecast.cs
MultiprojectStructure.BusinessLogic
Dependencies
MultiprojectStructure.Database
Dependencies

How do I register my workshops?

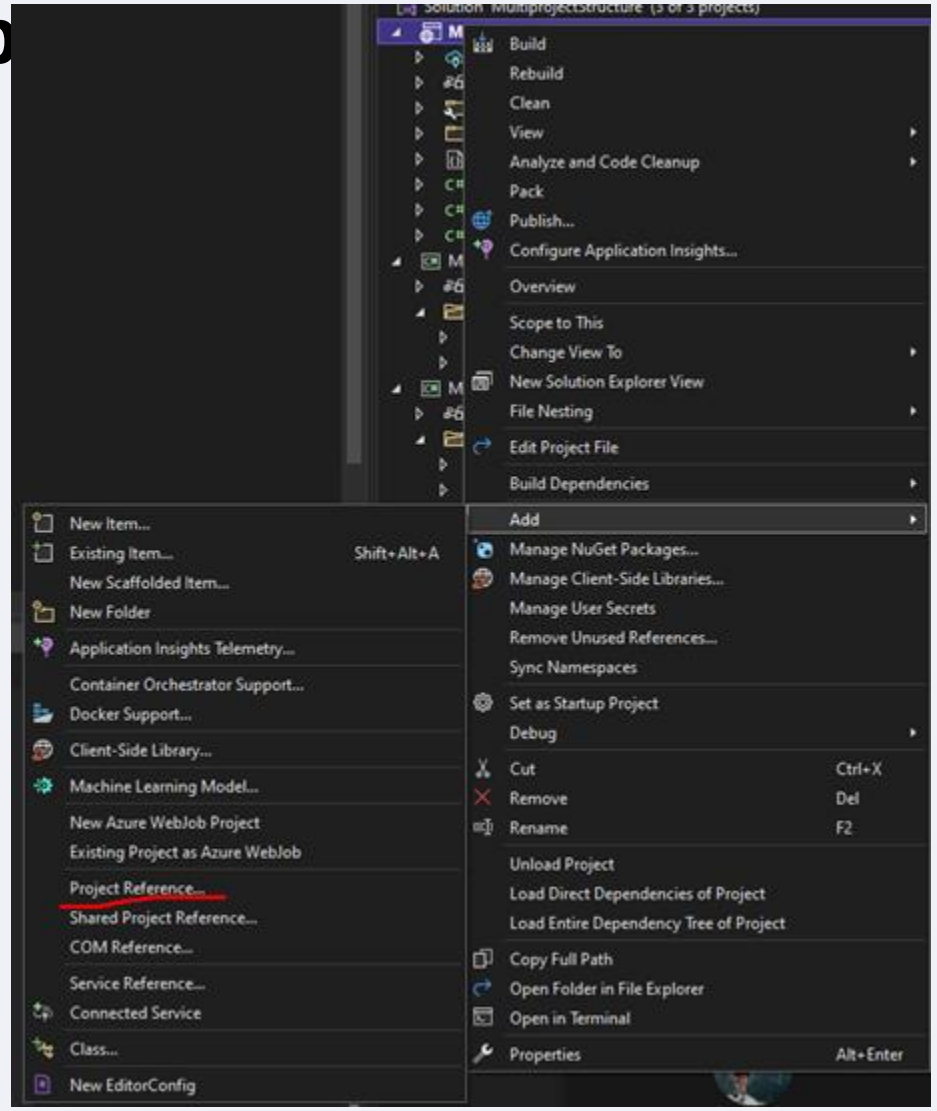
When starting to develop logic in different projects, we face the problem of registering services.



To reach others from the API project, you need to add References at the beginning

How do I register my workshop

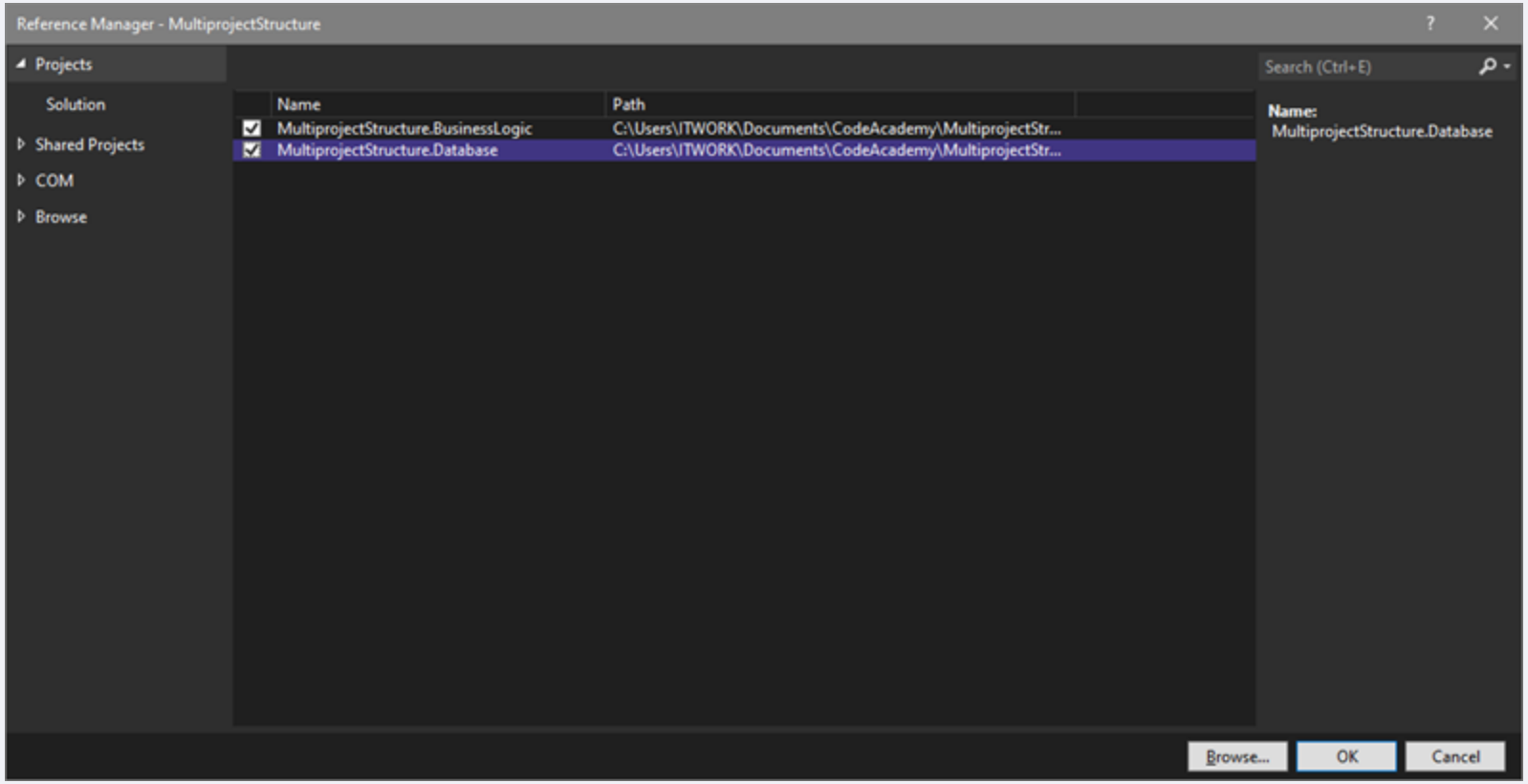
Adding a reference



How do I register my workshops?

In this case we add both

and press OK



How do I register my workshops?

The first idea that comes to mind might be to simply register services directly in the API project.

What's not good about this is that as the project grows, there will be a lot of these registrations, making the code difficult to understand.

```
services.AddScoped<IBusinessLogicService, BusinessLogicService>();
services.AddScoped<IDatabaseService, DatabaseService>();
```

How do I register my workshops?

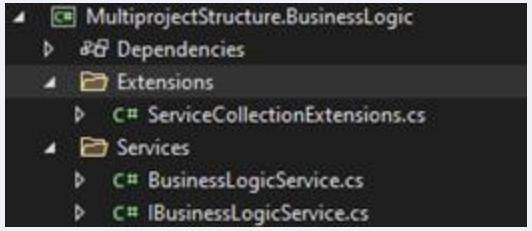
This can be avoided by writing **extension** methods for the IServiceCollection interface, since that is where we register them

```
// This method gets called by the runtime. Use this method to add services to the container.
public void ConfigureServices(IServiceCollection services)
{

    services.AddControllers();
    services.AddSwaggerGen(c =>
    {
        c.SwaggerDoc("v1", new OpenApiInfo { Title = "MultiprojectStructure", Version = "v1" });
    });
    services.AddScoped<IBusinessLogicService, BusinessLogicService>();
    services.AddScoped<IDatabaseService, DatabaseService>();
}
```

How do I register my workshops?

This is what a ServiceCollection extension class would look like with the AddBusinessLogic method



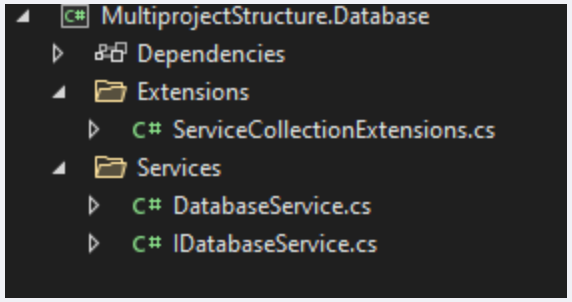

```cs
public static class ServiceCollectionExtensions
{
    public static IServiceCollection AddBusinessLogic(this IServiceCollection services)
    {
        services.AddScoped<IBusinessLogicService, BusinessLogicService>();
        return services;
    }
}
```

Now we can call this extension method from the Startup.cs class

```cs
services.AddBusinessLogic();
```

How do I register my workshops?

We do the same with the Database project




```cs
public static class ServiceCollectionExtensions
{
    public static IServiceCollection AddDatabaseServices(this IServiceCollection services)
    {
        services.AddScoped<IDatabaseService, DatabaseService>();
        return services;
    }
}
```

Calling the method

```cs
services.AddBusinessLogic();
services.AddDatabaseServices();
```

How do I register my workshops?

This allows us to group the services in a much neater way and makes it easier to read the project.

Multi-project structure

Task 1

- Try transferring yesterday's photo project to multiple projects
- Unit tests;)