Pre-Requisite for this Course

Link to download .NET 7

<https://dotnet.microsoft.com/en-us/download>

Node JS

It’s a javascript runtime to run our reactjs application inside our development server.

NVM

It’s a node version manager

If u use NVM, you can switch between nodejs version very easily.

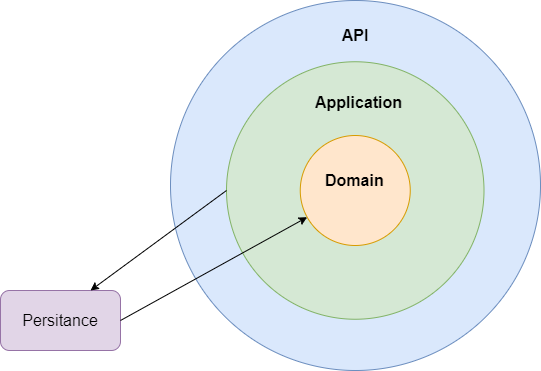
Visual Studio Code – Integrated Development Environment

Post Man – For testing our API

Resources has been forked from [TryCatchLearn/Reactivities](https://github.com/TryCatchLearn/Reactivities) to [MadOnProgramming](https://github.com/MadOnProgramming)/[.NetCoreWithReact](https://github.com/MadOnProgramming/.NetCoreWithReact)

**Clean architecture**

Here we are going to implement clean architecture. I am listing the layer here from the inner most ring to outer most ring. Outer depends on the immediate inner layer but the inner layer doesn’t depends on the outer layer.



* Domain(Entities for our Business model) – *inner most*
* Application layer (contains business login of our application)
* API (Responsible for handling http request and response) – *outer most*

**Creating a .net core project and walking through the skeleton**

Dotnet CLI commands

Dotnet –info

* To get the information about the installed sdk and runtime in our system

Dotnet new list

* To see the list of template that we can create

Dotnet new sln -n YourSolutionName

* Creates solution

Dotnet new webapi -n YourProjectName

* Creates webapi project

Dotnet new classlib -n YourProjectName

* Creates class library project

Dotnet sln add youcsproject\_file

* Adds project to the solution

Dotnet add reference csproject\_file

* Adds one project as a reference to another

**VS code command**

After opening the solution folder in vscode, open command pallete Ctrl+Shift+P

And select ‘Generate assets for build and debug’

This will generate .vscode folder to ur folder

**API Project template**

Properties\launchSettings.json

Keep only the necessary settings and change others.

Program.cs

- It’s the entry point in this application

- Creates a builder from WebApplication

- Region for adding services to the container(builder), these services can be injected while we are using it.

- Build app from the builder

- Region for adding middleware to the application pipeline

- Run the application

Some properties in .csproj

<Nullable> - In .net 6, by default string is required, You can explicitly enable or disable through this property

<ImplicitUsings>

**Starting our implementation**

Start by creating entity classes under domain project.

In our case, our project going to revolve aroung entity called Activity. So, I am creating a class called activity

*public* *class* Activity

    {

*public* Guid Id { *get*; *set*; }

*public* *string* Title { *get*; *set*; }

*public* *string* Description { *get*; *set*; }

*public* *string* Category { *get*; *set*; }

*public* DateTime Date{*get*;*set*;}

*public* *string* Venue { *get*; *set*; }

*public* *string* City { *get*; *set*; }

    }

Note: always create a property with a shortcut of typing ‘*prop’* and type enter

Always name the id property with ‘id’ instead of giving name like ‘ActivityId’. Beacause, entity framework couldn’t recognize this as key property. Unless, u explicitly specify as key property.

**Adding Entity framework to our project**

Here we are not going to use sql server. Instead, we are going to use entity framework with sql lite db.

Add nuget package “Microsoft.EntityFrameworkcore.Sqllite” to our persistent project.

Then add our dbcontext class like below

*using* Domain;

*using* Microsoft.EntityFrameworkCore;

*namespace* Persistence

{

*public* *class* DataContext : DbContext

    {

*public* DataContext(DbContextOptions *options*) : *base*(*options*)

        {

        }

*public* DbSet<Activity> Activities { *get*; *set*; }

    }

}

Then add services to our webapi project program.cs , inorder to use our datacontext class

Through dependency injection.

builder.Services.AddDbContext<DataContext>(*opt* => {

*opt*.UseSqlite(builder.Configuration.GetConnectionString("DefaultConnection"));

});

So, we have successfully integrated entityframework core within our application with sqllite.

**Entity framework migrations**

Next we are going install dotnet-ef tool

Get list of dotnet tools installed by the following command

**Dotnet tool list -g**

Here -g – represents global

If dotnet-ef is not available,then install through the following command

**Dotnet tool install –global dotnet-ef –version 7.0.0**

dotnet-ef command to add migration

dotnet-ef migrations add <<MigrationName>> -s <<.\Folder\StartupProject.csproj>> -p <<.\Folder\dbcontextHoldingProject.csproj>>