Medium Search

Get unlimited access to the best of Medium for less than $1/week. Become a member

Clean Architecture by Uncle Bob (aka Robert C. Martin) and DDD (Domain-Driven-Design) in short

Introduction

Fahrettin Enes · Follow

3 min read · Jan 22, 2024

Listen Share More

Clean Architecture is one of the most used architectures in the modern tech stacks, especially on the back-end side. It was published in August of 2012 on Uncle Bob's (aka Robert C. Martin) blog.

About Uncle Bob

Uncle Bob is one of the persons (even maybe the only person) we realize and always mention while talking about clean code as programmers. He is also the creator of the SOLID principles which are quite important and framework/language-independent about clean-code.

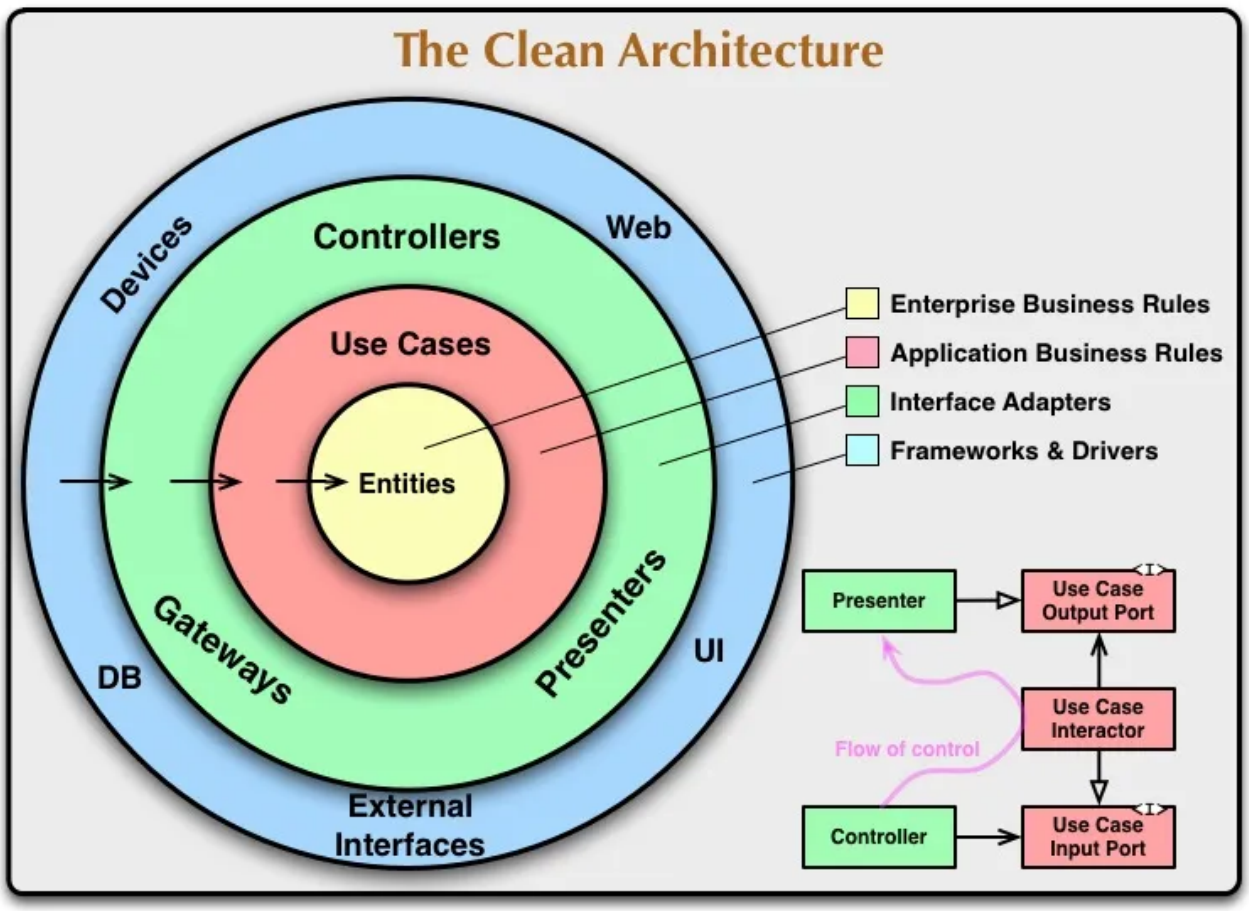
Why do people prefer Clean Architecture?

Clean Architecture is a well-fit for most of back-end or even monolith full-stack applications. It helps you to make your project testable, manageable, and easy to understand. It's also framework-independent and easy to implement in diverse of technologies (e.g. Java, NestJS, ASP.NET Web API, and more). It also mostly created pillars of DDD (Domain-Driven-Design) which was invented by **Eric Evans in 2003.**

The Layers

The layers we have in our project are quite dynamic when we choose to work with

Clean Architecture. We can think that architecture as a circle that has a couple of layers. You can find an example of that circle in this image:

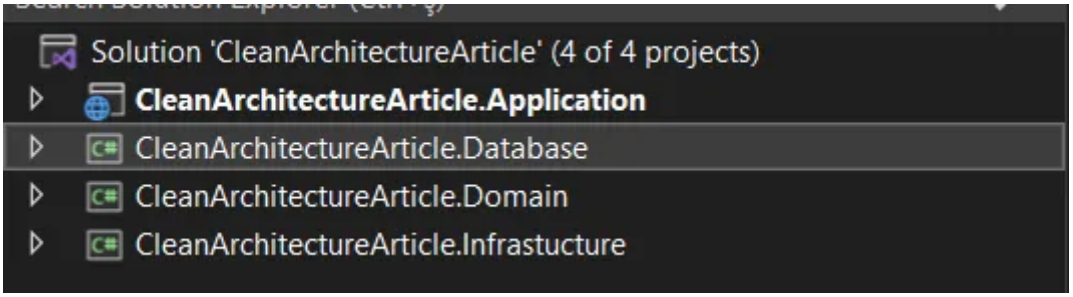


This image is about a monolith full-stack application and is quite old. You also see some of the names for each circle, these names are also may be quite dynamic for your project. In most of the modern use of this architecture, people mostly prefer different names.

Implementing it to modern-world tech-stack

Just like I said, most people prefer different names for this architecture while using it in modern tech stacks. I prefer using this on the back-end via ASP.NET Web API. I'll also follow the ASP.NET Web API in this writing while talking about file structure and layers.

I prefer and recommend using those names and file structure for each layer:



Application Layer: That layer represents the main application, it may be a console application or back-end application. So, the main project is contained in this layer. Some people also prefer using **API** naming for back-end applications.

Database Layer: If your application needs a database, you will also need that layer. You may create your repositories, related ORM models, and other things here.

Domain Layer: You can see a different naming like **Entities, and Common** instead of **Domain** in some of the applications. That layer may contain your entities like **User, Role, Product,** etc.

Infrastructure Layer: You can also see a different naming here too. Some people prefer using **the Services** name for that layer since it mostly contains your services, workers, and that kind of thing.

DDD (Domain-Driven-Design)

The Clean Architecture is already created on the pillars of DDD, just like I said in the introduction of writing.

When you choose to use DDD in your project, you are taking all entities in the center of your application.

Clean Code | Clean Architecture | Aspnet Web Api | Dotnet | Uncle Bob

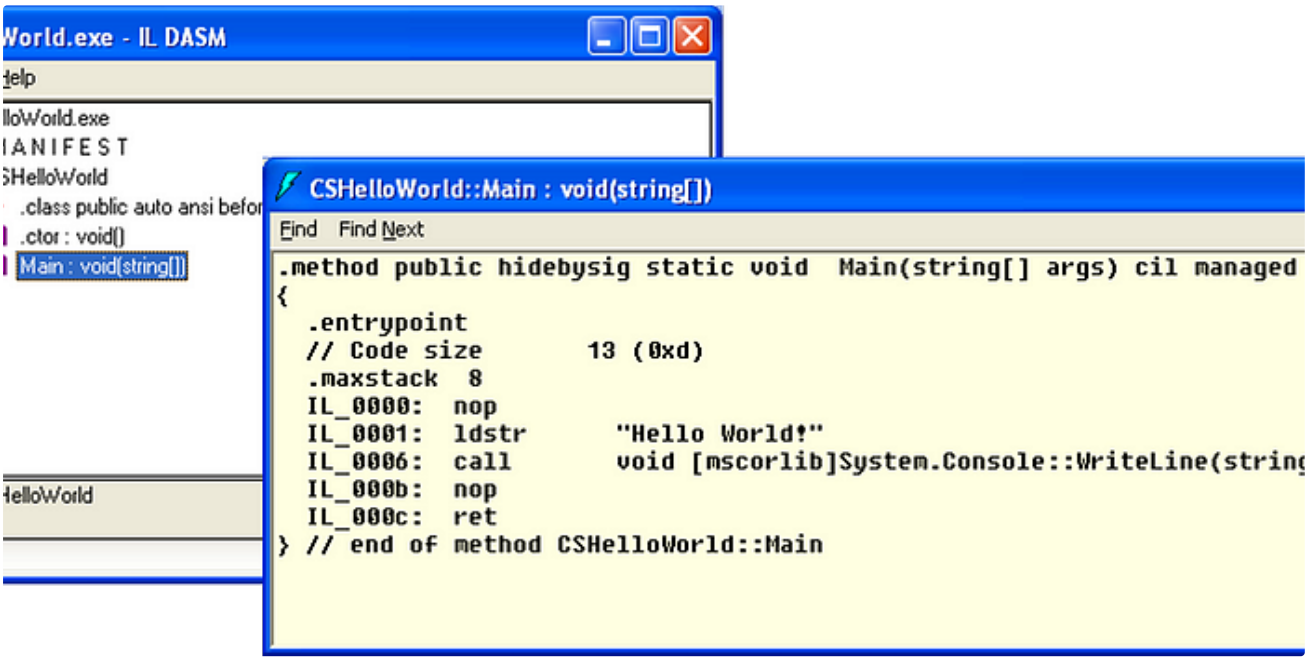
Follow

Written by Fahrettin Enes

1 Follower

problem-solver | bug-fixer | researcher

More from Fahrettin Enes



Fahrettin Enes

Journey of a .NET Project

In this article, I'll be talking about the compilation steps (a.k.a journey) of a .NET Project. I first need to talk about Common Language...

Jan 16

See all from Fahrettin Enes

Recommended from Medium

AshokReddy

Mastering Clean Architecture: Step-by-Step Guide to Building a Web API in .NET Core 8

1.Introduction

Sep 29 2

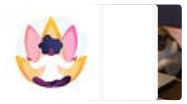
Milan Jovanović

Clean Architecture: The Missing Chapter

I see the same mistake happen over and over again.

Nov 2 33

Lists



Stories to Help You Grow as a Software Developer

19 stories · 1463 saves

Deewakar Kumar

Filters vs. Middleware in .NET Core: Key Differences, Use Cases, and Top Interview Questions

Introduction

Oct 31 15

Matt Bentley in Level Up Coding

Ultimate .NET Project Setup with Clean Architecture and Domain-Driven Design

An opinionated ASP.NET Core solution setup for creating web applications using Clean

Yohan Malshika

Rate Limiting in ASP.NET Core API

How to config Rate Limiting in ASP.NET Core Web API

Oct 21 8

Dusan Velimirovic

Understanding the Strategy Pattern in C#

The Strategy Pattern is a behavioral design pattern that allows a program to select an algorithm's behavior at runtime. It defines a family...

Aug 2 6

See more recommendations