Published in JavaScript in Plain English

Edson Moisinho Follow
Dec 30, 2022 · 3 min read · · Listen

DTO vs VO vs Entity vs POCO

Learn the differences between these concepts

Photo by Mathew Schwartz on Unsplash

Classes are one of the most basic concepts of software development, and they are used for an infinite number of different things, but when we talk about data and how to technically

keep and transfer data we can have difficulties understanding and using the best approaches.

In this article, I will explain nces, and where how they should be best used.

Data Transfer Object (DTO)

I think DTO is the most common object nowadays, a DTO is an object used to interchange data between layers, processes, or applications, a DTO is usually immutable and must not have any logic or behavior, it is a data-only object, and must be optimized to serialization and deserialization.

> *As the DTO should not have any business logic, it also does not have any validation methods or validation attributes, usually, the data transported is not reliable from a business perspective, which means that maybe a string property will be null, or an int property will be negative or invalid.*

A DTO does not reflect business entities, for example, a `person`, or a `product`, instead, it reflects what the consumer needs, it must be built for a specific purpose and most of the time this purpose combines data from multiples sources, it must have all data needed to serve the caller's needs, and their properties could be of any type, `string`, `bool`, `int`, `double`, etc.

Value Object (VO)

A VO is also an immutable object that is used to hold data but as the opposite of a DTO, the data in a VO is valid and reliable, it represents independent entities, such as a `product`, `user`, `contact` but it should not be based on identity, which means it should not be used to persist data or connect to databases or any other kind of shared storage mechanism.

> *Comparison made between VOs must consider the* **values** *of their properties and not their* **reference** *as in the case of DTOs.*

Basically, a VO must ensure that it has all the logic and structure to keep the data in a valid state, compared by values, being immutable, and providing only the information that is needed by its consumers.

Entity

Entities are mutable objects and are directly related to the tables in a database or a domain unit in a real-world application.

They must have all the behaviors needed to normalize the data, such as validation, CRUD operations, fetching related data from different sources, applying business logic, and so on.

Changing a value in an Entity, and after running a successful validation allows the persistence of the change and makes this change visible to others.

> *It is very frequent to have frameworks that use entities to map the tables in a database, where each table will be represented by an entity class and all the persistence and validation logic are part of the entity.*

POCO

POCO classes are a little bit close to Entity, but the main difference is that they do not have any behavior, they are just simple classes that keep the data that represent business models and will be used for other classes to be persisted.

A POCO class does not depend on any framework or system, since they only map business models, such as database tables, and all the validation logic or persistent rules should not be part of a POCO class.

> *POCO means Plain Old CLR Object, and it is an adaptation of POJO that means Plain Old Java Objects*

Map of Responsibilities

As the types of classes have similar concepts but different responsibilities we can think of a responsibilities map as below:

	Immutable	Id	
DTO	✅		
VO	✅	❌	❌
Entity	❌	✅	✅
POCO	❌	✅	❌

type of classes.md hosted with ❤ by GitHub view raw

Final Considerations

Although the differences between all these classes are very subtle they also can change over the implementations and depending on the language, framework, architecture, or other concepts and of course, the authors.

We can also find other variances but I think those in this article are the most common and important.

I hope it can help you on understanding better these concepts, and organize the classes inside your projects.

Thank you again for reading and see you next time.

More content at PlainEnglish.io.

Sign up for our free weekly newsletter. Follow us on Twitter, LinkedIn, YouTube, and Discord.

Looking to scale awareness and adoption for your tech startup? Check out Circuit.

Value Objects Entity Data Transfer Object Poco Architecture