Value Objects

> An object that represents a descriptive aspect of the domain with no conceptual identity is called a VALUE OBJECT.
>
> (Eric Evans)

Two Entities with the same properties but with different `Id` s are considered as different entities. However, Value Objects have no `Id` s and they are considered as equals if they have the same property values.

The ValueObject Class

`ValueObject` is an abstract class that can be inherited to create a Value Object class.

Example: An Address class

```
public class Address : ValueObject
{
    public Guid CityId { get; private set; }

    public string Street { get; private set; }

    public int Number { get; private set; }

    private Address()
    {

    }

    public Address(
        Guid cityId,
        string street,
        int number)
    {
        CityId = cityId;
        Street = street;
        Number = number;
    }

    protected override IEnumerable<object> GetAtomicVal
    {
        yield return Street;
        yield return CityId;
        yield return Number;
    }
}
```

- A Value Object class must implement the `GetAtomicValues()` method to return the primitive values.

ValueEquals

4.1 (latest) English

Filter topics

- Getting Started
- Startup Templates
- Tutorials
- Fundamentals
- Infrastructure
- Architecture
  - Modularity
  - Domain Driven Design
    - Overall
    - Domain Layer
      - Entities & Aggregate Roots
      - Value Objects
      - Repositories
      - Domain Services
      - Specifications
    - Application Layer
    - Guide: Implementing DDD
  - Multi Tenancy
  - Microservices
- API
- User Interface
- Data Access
- Real Time
- Testing
- Samples
- Application Modules
- Release Information
- Reference
- Contribution Guide

`ValueObject.ValueEquals(...)` method is used to check if two Value Objects are equals.

Example: Check if two addresses are equals

```
Address address1 = ...
Address address2 = ...

if (address1.ValueEquals(address2)) //Check equality
{
    ...
}
```

Best Practices

Here are some best practices when using Value Objects:

- Design a value object as **immutable** (like the Address above) if there is not a good reason for designing it as mutable.
- The properties that make up a Value Object should form a conceptual whole. For example, CityId, Street and Number shouldn't be separate properties of a Person entity. This also makes the Person entity simpler.

See Also

- Entities

Share on :

In this document