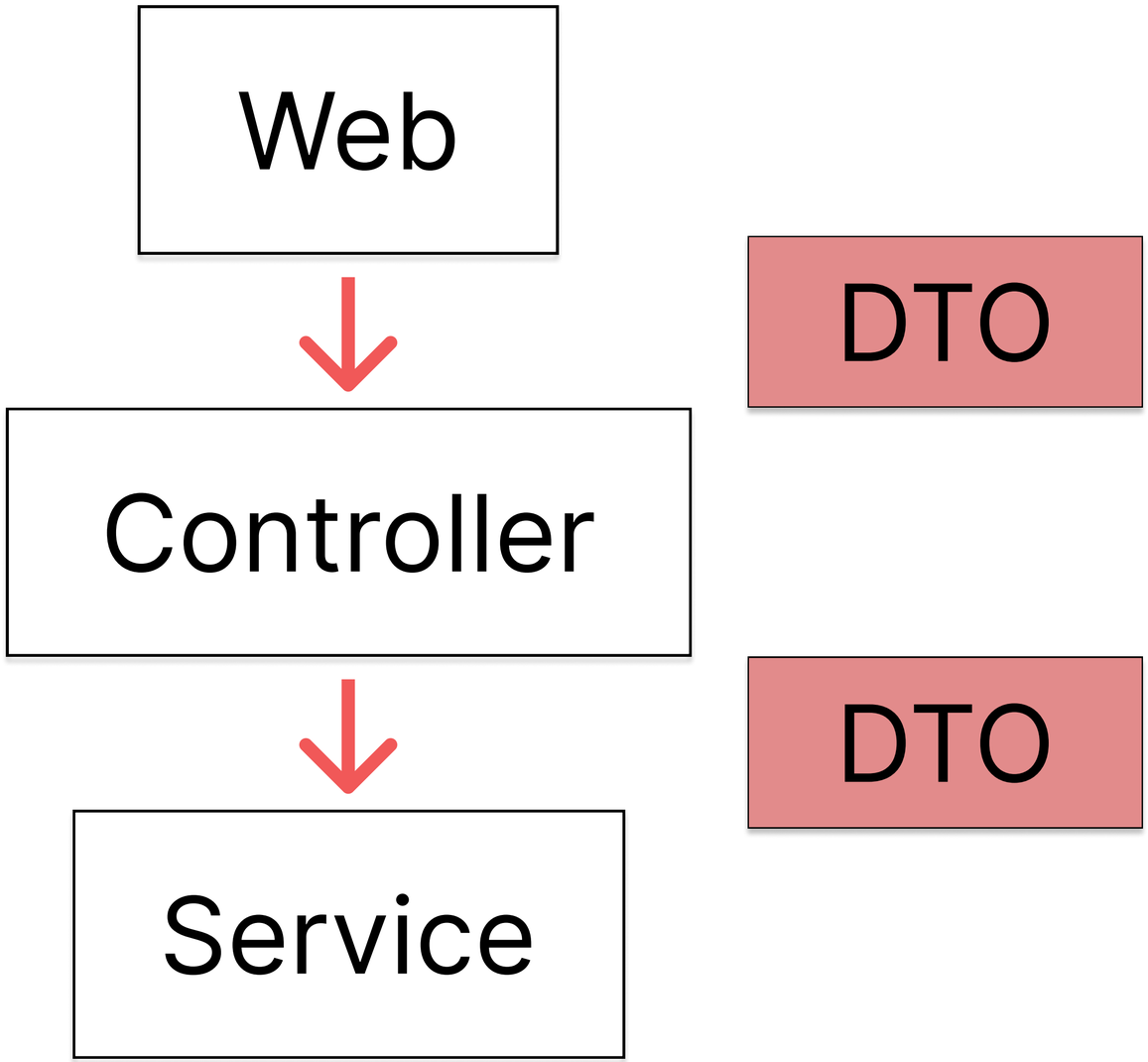
DTO와 VO

DTO (Data Transfer Object))

= 데이터를 전송하기 위한 객체

DTO



DTO

Code Blame 12 lines (10 loc) · 232 Bytes Code 55% faster with GitHub Copilot

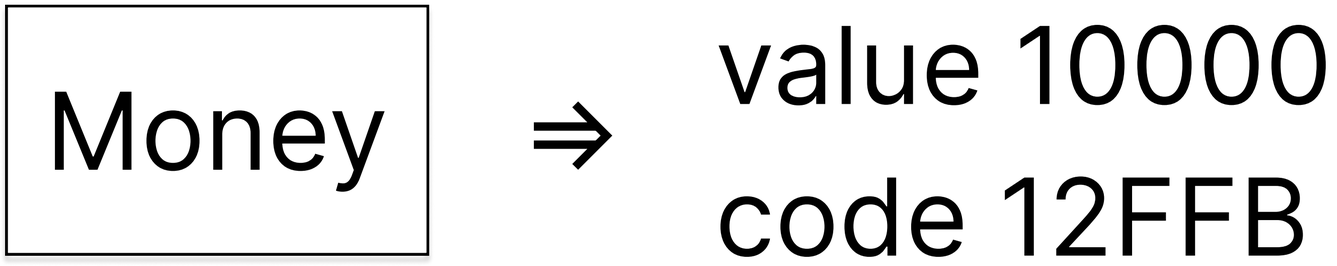
```java
package com.example.gudgement.member.dto.response;

import lombok.Builder;
import lombok.Getter;

@Getter
@Builder
public class MemberVerifyResponseDto {
    private Long id;
    private String email;
    private boolean isValid;
}
```

- getter, setter만 가진다.
  ⇒ 비즈니스 로직을 가지면 안된다.
- 가변, 불변 가능
- 계층 간 데이터 전달이 목적

VO (Value Object)

= 값을 표현하기 위한 객체

VO



VO

```java
@Embeddable
@NoArgsConstructor(access = AccessLevel.PROTECTED)
@EqualsAndHashCode
@Getter
public class Name {

    public static final int MAX_LENGTH = 20;

    @Column(name = "name", nullable = false, length = MAX_LENGTH)
    private String value;

    public Name(final String value) {
        validateNull(value);
        final String trimmedValue = value.trim();
        validate(trimmedValue);
        this.value = trimmedValue;
    }

    private void validateNull(final String value) {
        if (Objects.isNull(value)) {
            throw new NullPointerException("멤버 이름은 null일 수 없습니다.");
        }
    }

    private void validate(final String value) {
        if (value.length() > MAX_LENGTH) {
            throw new MemberNameLengthException(MAX_LENGTH, value);
        }
        if (value.isBlank()) {
            throw new MemberNameBlankException();
        }
    }

    public Name change(final String name) {
        return new Name(name);
    }
}
```

- 객체의 불변성을 보장해야한다
- 로직을 포함할 수 있다
- 서로 다른 객체이더라도 속성 값이 같다면 같은 객체이다

VO

```java
@Embeddable
@NoArgsConstructor(access = AccessLevel.PROTECTED)
@EqualsAndHashCode
@Getter
public class Name {

    public static final int MAX_LENGTH = 20;

    @Column(name = "name", nullable = false, length = MAX_LENGTH)
    private String value;

    public Name(final String value) {
        validateNull(value);
        final String trimmedValue = value.trim();
        validate(trimmedValue);
        this.value = trimmedValue;
    }

    private void validateNull(final String value) {
        if (Objects.isNull(value)) {
            throw new NullPointerException("멤버 이름은 null일 수 없습니다.");
        }
    }

    private void validate(final String value) {
        if (value.length() > MAX_LENGTH) {
            throw new MemberNameLengthException(MAX_LENGTH, value);
        }
        if (value.isBlank()) {
            throw new MemberNameBlankException();
        }
    }

    public Name change(final String name) {
        return new Name(name);
    }
}
```

- 객체의 불변성을 보장해야한다
- 로직을 포함할 수 있다
- ~~서로 다른 객체이더라도 속성 값이 같다면 같은 객체이다~~

VO

Name name = new Name(“권민우”);

이 둘은 같은가?

VO

권민우 ≠ 권민우

이 둘은 같은가? NO

VO

1. HashCode

- 함수형 파라미터를 사용할 때 이용된다.

2. equalsTo

- 값을 비교할 때 사용된다.

DTO와 VO의 차이

DTO

- 로직을 가질 수 없음
- 계층간 데이터 전송이 목적
- 가변 or 불변이 선택적

VO

- 로직을 가질 수 있음
- 의미있는 값 표현이 목적
- 불변 객체임을 보장

끝