DTO vs VO

테크톡 | 김지우

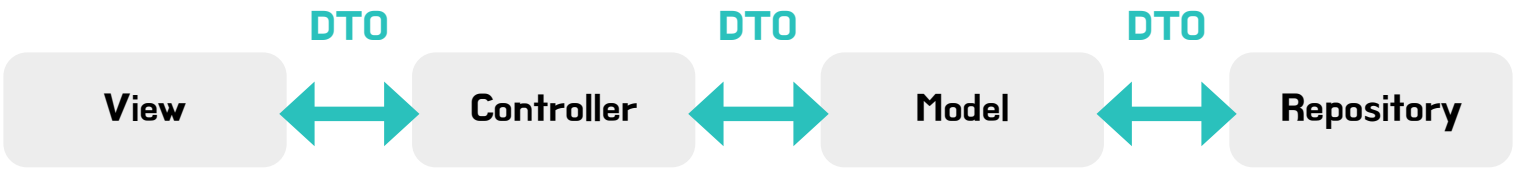
📌 목차

- DTO
- VO
- DTO vs VO

DTO

DTO (Data Transfer Object)

레이어, 프로세스 간 데이터를 전송하기 위한 객체



DTO (Data Transfer Object)

레이어, 프로세스 간 데이터를 전송하기 위한 객체

getter/setter 메서드만 포함

다른 로직을 포함하지 않음

가변 객체

DTO

도메인이 아닌 DTO를 사용해야 하는 이유?

DTO

Transfer 자원 낭비

Presentation Layer에서 필요한 데이터를

한 번의 Transfer로 전달하기 어려움

DTO

예) 로또 게임

```
구입금액을 입력해 주세요.
14000
14개를 구매했습니다.
[8, 21, 23, 41, 42, 43]
[3, 5, 11, 16, 32, 38]
[7, 11, 16, 35, 36, 44]
[1, 8, 11, 31, 41, 42]
[13, 14, 16, 38, 42, 45]
[7, 11, 30, 40, 42, 43]
[2, 13, 22, 32, 38, 45]
[23, 25, 33, 36, 39, 41]
[1, 3, 5, 14, 22, 45]
[5, 9, 38, 41, 43, 44]
[2, 8, 9, 18, 19, 21`]
[13, 14, 18, 21, 23, 35]
[17, 21, 29, 37, 42, 45]
[3, 8, 27, 30, 35, 44]
```

```java
public Map<String, Object> purchase(PurchaseInput purchaseInput) throws IllegalArgumentException {
    PurchaseCount purchaseCount = new PurchaseCount(purchaseInput.getPrice());
    LottoSet lottoSet = new LottoSet(purchaseCount);

    Map<String, Object> result = new HashMap<>();
    result.put("purchaseCount", purchaseCount);
    result.put("lottoSet", lottoSet);
    return result;
}
```

```java
@Getter
@RequiredArgsConstructor
@Builder
public class PurchaseResult {

    private final PurchaseCount purchaseCount;
    private final LottoSet lottoSet;
}
```

```java
public PurchaseResult purchase(PurchaseInput purchaseInput) throws IllegalArgumentException {
    PurchaseCount purchaseCount = new PurchaseCount(purchaseInput.getPrice());
    LottoSet lottoSet = new LottoSet(purchaseCount);

    return PurchaseResult.builder()
            .purchaseCount(purchaseCount)
            .lottoSet(lottoSet)
            .build();
}
```

DTO

도메인의 캡슐화가 깨짐

Presentation Layer에

과한 도메인 내부 메서드, 필드 노출

DTO

예) 로또 게임

```
당첨 통계
---------
3개 일치 (5000원)- 1개
4개 일치 (50000원)- 0개
5개 일치 (1500000원)- 0개
5개 일치, 보너스 볼 일치(30000000원) - 0개
6개 일치 (2000000000원)- 0개
총 수익률은 0.35입니다. // 기준이 1이기 때문에
```

```java
public class LottoStatistics {

    @Getter
    private final PrizeCount prizeCount;
    private final PurchaseCount purchaseCount;

    @Builder
    public LottoStatistics(PrizeCount prizeCount, PurchaseCount purchaseCount) {
        this.prizeCount = prizeCount;
        this.purchaseCount = purchaseCount;
    }

    public double calculateProfitRate() {
        return (double) Prize.sumOfPrizeMoney(prizeCount)
                / (purchaseCount.getPurchaseCount() * Lotto.PRICE);
    }
}
```

```java
@Getter
@RequiredArgsConstructor
@Builder
public class LottoResult {

    private final PrizeCount prizeCount;
    private final double profitRate;
}
```

DTO

도메인의 단일 책임 원칙 위반

도메인이 행동과 역할 밖의 필드와 메서드를 추가하게 되면서
시스템의 안정성이 깨짐

DTO

예) 자동차 게임

```java
public class Car {

    private static final int INITIAL_POSITION = 0;
    private static final int NAME_ALIGN_STANDARD = 5;
    private static final String POSITION_MARK = "-";
    private static final String COLON = " : ";

    private final String name;
    private int position;

    public Car(String name) {
        this.name = name;
        this.position = INITIAL_POSITION;
    }

    @Override
    public String toString() {
        StringBuilder sb = new StringBuilder();
        return sb.append(StringUtils.alignLeft(name, NAME_ALIGN_STANDARD))
                .append(COLON)
                .append(StringUtils.repeat(POSITION_MARK, position))
                .toString();
    }
}
```

DTO

DTO (Data Transfer Object)

레이어, 프로세스 간 데이터를 전송하기 위한 객체

한 번의 call에 필요한 데이터만 명확하게 담아 전송

Business 로직과 Presentation 로직을 분리

캡슐화된 도메인과 DTO가 각각의 단일 책임만 수행

V0

VO (Value Object)

값을 표현하기 위한 객체

속성 값 자체가 식별자 역할 `equals() & hashCode()`

불변 객체

도메인의 일종으로서 기능과 역할을 수행할 수 있음

VO를 사용해야 하는 이유?

객체 지향 프로그래밍

엔티티의 원시값을 VO로 포장함으로써

테이블 관점이 아닌 객체 지향적인 관점으로 프로그래밍

VO

```java
public class Point {
    private final int x;
    private final int y;

    public Point(int x, int y) {
        this.x = x;
        this.y = y;
    }

    @Override
    public boolean equals(final Object o) {
        if (this == o) return true;
        if (o == null || getClass() != o.getClass()) return false;
        final Point point = (Point) o;
        return x == point.x &&
                y == point.y;
    }

    @Override
    public int hashCode() {
        return Objects.hash(x, y);
    }
}
```

```java
public class Person {
    final PersonId id;
    final FirstName firstName;
    final LastName lastName;
    final Address address;

    public Person(PersonId id, FirstName firstNam

    // equals() & hashCode()...
}
```

```java
class Address {
    final String street;
    final String streetNo;
    final String city;
    final String postalCode;

    public Address(String street, String streetNo, String
        this.street = street;
        this.streetNo = streetNo;
        this.city = city;
        this.postalCode = postalCode;
    }

    // equals() & hashCode()...
```

VO

안전한 객체

생성자에서 제약사항을 추가하고 불변성을 보장함으로써

객체를 항상 안전하게 관리

VO

예) 로또 게임

```java
public class LottoNumber {

    public static final int LOWER_BOUND = 1;
    public static final int UPPER_BOUND = 45;

    @Getter
    private final int lottoNumber;

    public LottoNumber(int lottoNumber) throws IllegalArgumentException {
        validate(lottoNumber);

        this.lottoNumber = lottoNumber;
    }

    private void validate(int lottoNumber) throws IllegalArgumentException {...}

    // equals() & hashCode()...
```

VO

예) 로또 게임 (일급 컬렉션)

```
public class Lotto {

    public static final int PRICE = 1000;
    public static final int LOTTO_NUMBER_SIZE = 6;

    @Getter
    private final List<LottoNumber> lottoNumbers;

    public Lotto(List<Integer> numbers) throws IllegalArgumentException {
        validate(numbers);

        numbers.sort(Integer::compare);
        this.lottoNumbers = Collections.unmodifiableList(numbers.stream()
                .map(LottoNumber::new)
                .collect(Collectors.toList()));
    }

    private void validate(List<Integer> numbers) throws IllegalArgumentException {...}

    @Override
    public boolean equals(Object o) {...}

    @Override
    public int hashCode() { return Objects.hash(lottoNumbers); }
}
```

VO (Value Object)

값을 표현하기 위한 객체

원시값을 객체 지향적으로 표현

식별자 == 속성값

안전한 객체를 생성하고 불변으로 유지

DTO vs VO

DTO vs VO

DTO	VO
데이터 전달 객체	값 표현 도메인
로직 불포함	로직 포함
가변	불변

DTO를 불변객체로 사용하거나 DTO 안에 VO 객체를 담으면
데이터 전송 과정에서 값의 불변을 보장해서 더 안전!

감사합니다

참고자료

- https://martinfowler.com/eaaCatalog/dataTransferObject.html
- https://martinfowler.com/bliki/ValueObject.html
- https://velog.io/@livenow/Java-VOValue-Object%EB%9E%80
- https://woowacourse.github.io/tecoble/post/2020-06-11-value-object/
- https://woowacourse.github.io/tecoble/post/2020-05-08-First-Class-Collection/