

# 8장. 경계 간 매핑하기

## 매핑 논쟁

 찬성파

 반대파

## 매핑 전략

- 1 매핑하지 않기
- 2 양방향 매핑 전략
- 3 완전 매핑 전략
- 4 단방향 매핑 전략

언제 어떤 매핑 전략을 사용할까?

## 결론

## 매핑 논쟁

 찬성파

- 계층 간 매핑을 하지 않으면 같은 모델을 사용한다.
- 계층이 강하게 결합된다.

 반대파

- 계층 간 매핑을 하면 보일러플레이트 코드가 너무 많아진다.
- 매핑은 과하다.

## 매핑 전략

### 1 매핑하지 않기

- 단일 책임 원칙 위반
- 간단한 CRUD 유스케이스의 경우 적합
  - 모든 계층이 정확히 같은 구조의 정확히 같은 정보를 필요로 하는 경우
  - 시간이 지나면 유효하지 않을 확률이 높다 (...)

- 애플리케이션 계층이나 도메인 계층에서 웹/영속성 문제를 다루면 곧바로 다른 전략을 취해야 함

## 2 양방향 매핑 전략

- 각 계층이 자신에게 최적화된 모델과 구조를 가질 수 있음
- 오염되지 않은 깨끗한 도메인 모델을 가질 수 있음
- 단일 책임 원칙 **만족**
- 어려워보이지만, 매핑하지 않기 전략 다음으로 가장 간단하다 !
- 너무 많은 보일러플레이트 코드가 생긴다.

## 3 완전 매핑 전략

- 연산/작업마다 별도의 입출력 모델을 사용
- 특화된 모델을 가질 수 있음
- 한 계층을 여러 개 커맨드로 매핑하기 때문에 **더 많은 코드가 필요**
- 구현 및 유지보수가 쉬움
- **전역 패턴으로 비추**
- 웹 계층 - 애플리케이션 계층 간 상태 변경 유스케이스의 경계를 명확히 할 때 가장 좋음
- 애플리케이션 계층 - 영속성 계층 간에서는 매핑 오버헤드때문에 비추

## 4 단방향 매핑 전략


- 모든 계층의 모델들이 같은 인터페이스 구현
- 도메인 모델의 행동에 접근 가능 (DDD의 팩토리 개념과 어울림)
  - **팩토리** : 어떤 특정한 상태에서부터 도메인 객체를 재구성할 책임을 가진다.
- 계층 간 모델이 비슷할 때 가장 효과적 (ex: 읽기 전용 연산)

# 언제 어떤 매핑 전략을 사용할까?

- **변경 유스케이스**를 작업한다면 웹 - 애플리케이션 계층 사이에는 **완전 매핑 전략**을 택하자.

- 변경 유스케이스 를 작업한다면 애플리케이션 - 영속성 계층 사이에는 매핑하지 않기 전략을 택하자.
- 쿼리 작업을 한다면 웹 - 애플리케이션 - 영속성 계층 사이에서 매핑하지 않기 전략을 택하자.
  - 애플리케이션 계층에서 영속성 or 웹 문제를 다루는 경우 각각 양방향 매핑 전략으로 바뀌어야 한다.

## 결론

- 인커밍 포트와 아웃고잉 포트는 서로간 통신 방법을 정의한다.
  - 매핑 여부와 매핑 전략이 여기에서 결정된다.
- 어떤 매핑 전략을 선택했더라도 언제든지 바꿀 수 있다.
- 어떤 전략도 철칙처럼 여기면 안된다.
- 유스케이스마다 적절한 전략을 택할 수 있어야 한다.
- 매핑 전략은 섞어쓸 수 있고, 섞어 써야만 한다. (전역 규칙 )
- 어떤 전략을 선택하든지 왜 해당 전략을 선택해야 하는지 설명할 수 있어야 한다.
- 매핑 가이드라인이 있다면 유지보수 하기 쉬운 코드로 팀에 보상이 될 수 있다.