09. 도메인 모델과 BOUNDED CONTEXT

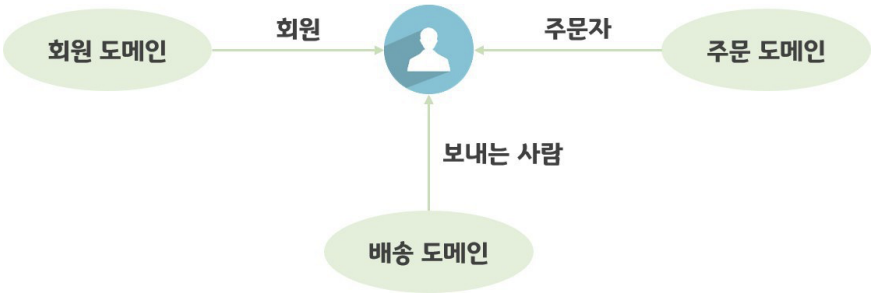
version 2018.36

도메인 모델과 경계

1장에서 말한 것처럼 한 도메인은 다시 여러 하위 도메인으로 구분되기 때문에 한 개의 모델로 여러 하위 도메인을 모두 표현하려고 시도하게 되면 모든 하위 도메인에 맞지 않는 모델을 만들게 됩니다.

예를 들어, 상품이라는 모델을 생각해봅시다. 카탈로그에서의 상품, 재고 관리에서의 상품, 주문에서의 상품, 배송에서의 상품은 이름만 같지 실제로 의미하는 것이 다릅니다. 또한 카탈로그에서 물리적으로 한 개인 상품이 재고 관리에서는 여러 개 존재할 수 있습니다.

논리적으로 같은 존재처럼 보이지만 하위 도메인에 따라 다른 용어를 사용하는 경우도 있습니다. 시스템을 사용하는 사람을 회원 도메인에서는 회원이라고 부르지만, 주문 도메인에서는 주문자라고 부르고, 배송 도메인에서는 보내는 사람이라 부르기도 합니다.



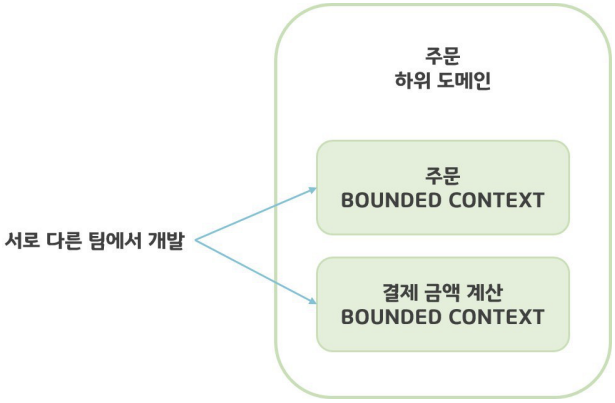
올바른 도메인 모델을 개발하려면 하위 도메인마다 모델을 만들어야 합니다. 각 모델은 명시적으로 구분되는 경계를 가져서 섞이지 않도록 해야 합니다. 여러 하위 도메인이 섞이기 시작하면 각 하위 도메인별로 다르게 발전하는 요구사항을 모델에 반영하기 어려워집니다.

모델은 특정한 컨텍스트(문맥)하에서 완전한 의미를 갖습니다. 같은 제품이라도 카탈로그 컨텍스트와 재고 컨테스트에서의 의미가 서로 다릅니다. 이렇게 **구분되는 모델의 경계를 갖는 컨텍스트를 BOUNDED CONTEXT**라고 부릅니다.

BOUNDED CONTEXT

BOUNDED CONTEXT는 모델의 경계를 결정하며 한 개의 BOUNDED CONTEXT는 논리적으로 한 개의 모델을 갖습니다. 또한 BOUNDED CONTEXT는 용어를 기준으로 구분합니다. 카탈로그 컨텍스트와 재고 컨텍스트는 서로 다른 용어를 사용하므로 이 용어를 기준으로 컨텍스트를 분리할 수 있습니다. 또한, **BOUNDED CONTEXT는 실제로 사용자에게 기능을 제공하는 물리적 시스템**으로 BOUNDED CONTEXT안에서 도메인 모델은 도메인을 구현합니다.

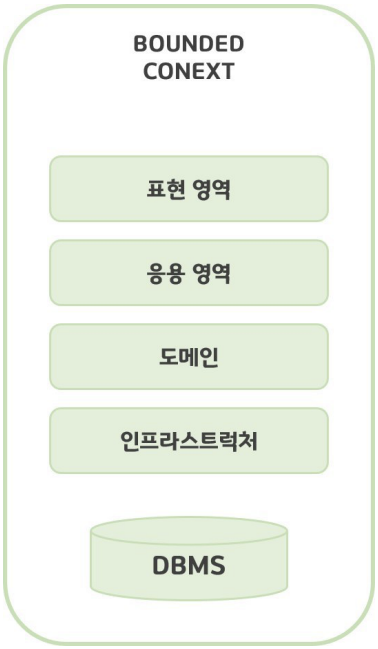
이상적으로 하위 도메인과 BOUNDED CONTEXT가 일대일 관계를 가지면 좋겠지만 현실은 그렇지 않을 때가 많습니다. BOUNDED CONTEXT는 기업의 팀 조직 구조에 따라 결정되기도 합니다. 예를 들어, 하나의 주문 하위 도메인이라도 주문을 처리하는 팀과 복합한 결제 금액 계산 로직을 구현하는 팀이 따로 있다고 해봅시다. 이 경우 주문 하위 도메인에 주문 BOUNDED CONTEXT와 결제 금액 계산 BOUNDED CONTEXT가 존재하게 됩니다.



규모가 작은 기업은 모든 하위 도메인을 한 개의 팀에서 구현할 때도 있습니다. 이러한 경우 하나의 시스템에서 모든 기능을 제공하므로 한 개의 BOUNDED CONTEXT에서 여러 하위 도메인을 구현하게 됩니다. 이 때 물리적인 BOUNDED CONTEXT가 한 개 이더라도 내부적으로 패키지를 활용하여 논리적인 BOUNDED CONTEXT를 만들어야 합니다.

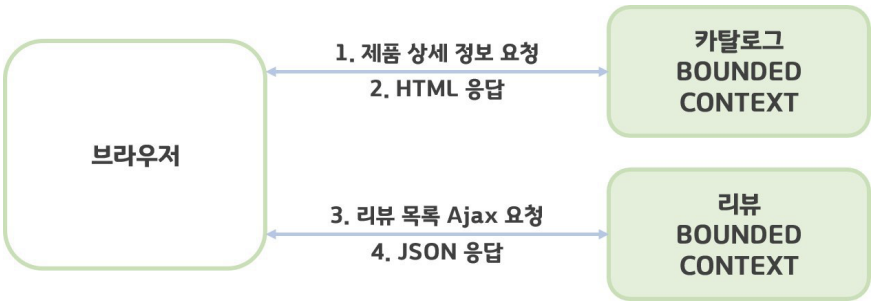
BOUNDED CONTEXT의 구현

BOUNDED CONTEXT는 도메인 모델뿐만 아니라 도메인 기능을 사용자에게 제공하는데 필요한 표현 영역, 응용 서비스, 인프라 영역 등을 모두 포함합니다. 도메인 모델의 데이터 구조가 바뀌면 DB 테이블 스키마도 함께 변경해야 하므로 해당 테이블도 BOUNDED CONTEXT에 포함됩니다.

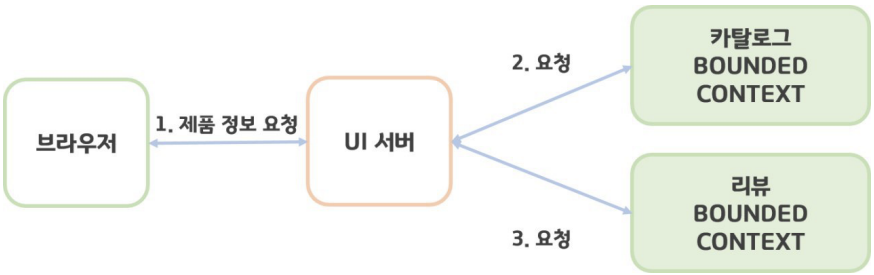


모든 BOUNDED CONTEXT를 반드시 도메인 주도로 개발할 필요는 없습니다. 상품의 리뷰는 복잡한 도메인 로직을 가지 않기 때문에 CRUD방식으로 구현해도 됩니다. 즉, DAO와 데이터 중심의 밸류 객체를 이용해서 리뷰 기능을 구현해도 기능을 유지보수하는 데 큰 문제가 없습니다.

BOUNDED CONTEXT가 반드시 사용자에게 보여지는 UI를 가져야 하는 것은 아닙니다. 예를 들어, 상품 상세 정보를 보여주는 페이지를 생각해봤을 때, 웹 브라우저는 아래의 그림처럼 동작할 수 있습니다.



아래의 그림과 같이 UI를 처리하는 서버를 두고 UI 서버에서 BOUNDED CONTEXT와 통신하여 사용자 요청을 처리하는 방법도 있습니다.



이 구조에서 UI 서버는 각 BOUNDED CONTEXT를 위한 출입구 역할을 수행합니다.

BOUNDED CONTEXT 간 통합

온라인 쇼핑 사이트에서 매출 증대를 위해 카탈로그 하위 도메인에 개인화 추천 기능을 도입하기로 했다고 가정 해봅시다. 기존 카탈로그 시스템을 개발하던 팀과 별도로 추천 시스템을 담당하는 팀이 새로 생겨서 이 팀에서 주도적으로 추천 시스템을 만들기로 했습니다. 이렇게 되면 카탈로그 하위 도메인에는 **기존 카탈로그를 위한 BOUNDED CONTEXT**와 **추천 기능을 위한 BOUNDED CONTEXT**가 생깁니다.

두 팀이 관련된 BOUNDED CONTEXT를 개발하면 자연스럽게 두 BOUNDED CONTEXT 간 통합이 발생합니다. 통합이 필요한 기능은 다음과 같습니다.

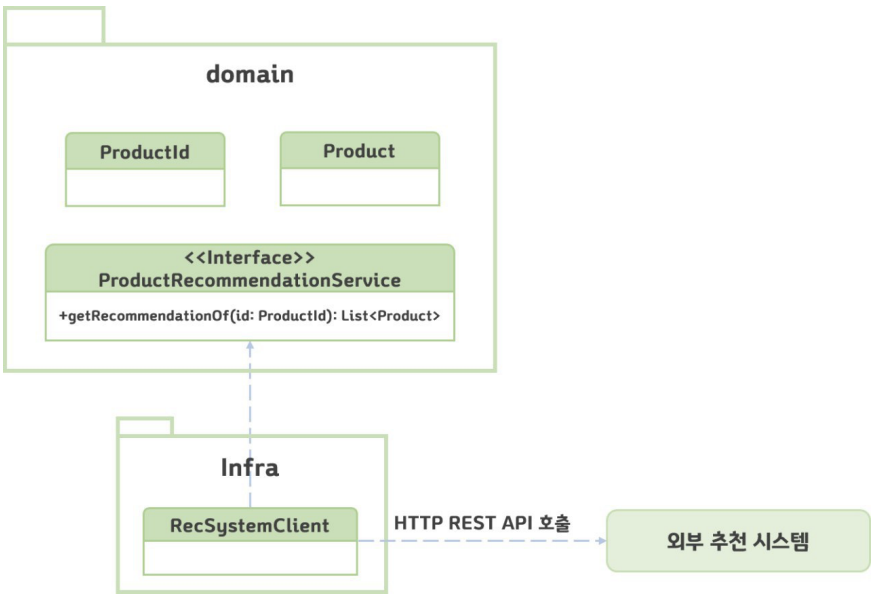
- 사용자가 제품 상세 페이지를 볼 때, 보고 있는 상품과 유사한 상품 목록을 하단에 보여준다.

사용자가 `카탈로그 BOUNDED CONTEXT`에 추천 제품 목록을 요청하면 `카탈로그 BOUNDED CONTEXT` 시스템에서는 `추천 BOUNDED CONTEXT`로 부터 추천 정보를 읽어와 추천 제품 목록을 제공합니다. 이 때 각 BOUNDED CONTEXT에서 **각각의 목적이 다르므로 구현된 도메인 모델은 다릅니다.**

카탈로그 시스템은 추천 시스템으로부터 추천 데이터를 받아오지만, 카탈로그 시스템에서 추천의 도메인 모델을 사용하기보다는 카탈로그 도메인 모델을 사용해서 추천 상품을 표현해야 합니다. 즉, 다음과 같이 카탈로그 모델을 기반으로 하는 도메인 서비스를 이용해서 상품 추천 기능을 표현해야 합니다.

```
public interface ProductRecommendationService {
  public List<Product> getRecommendationOf(ProductId id);
}
```

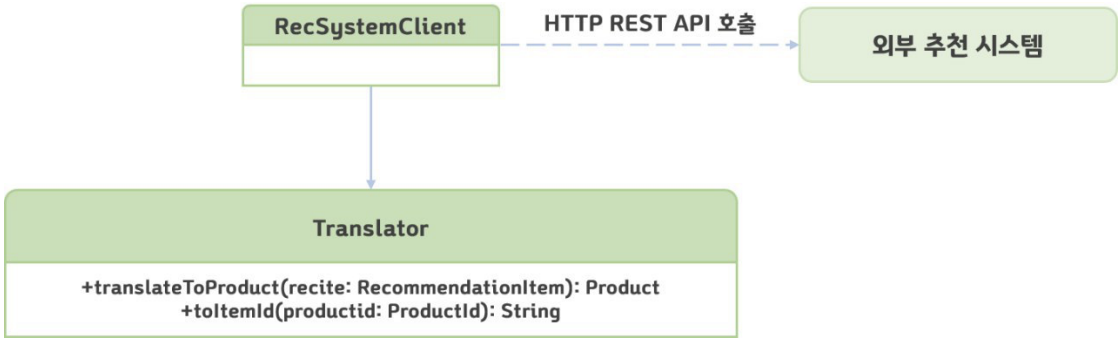
도메인 서비스를 구현한 클래스는 인프라스트럭처 영역에 위치합니다. **이 클래스는 외부 시스템과의 연동을 처리하고 외부 시스템의 모델과 현재 도메인 모델 간의 변환을 책임집니다.**



도메인 서비스를 구현한 클래스인 RecSystemClient는 외부 추천 시스템이 제공하는 REST API를 이용해서 특정 상품을 위한 추천 상품 목록을 로딩합니다. RecSystemClient는 REST API로부터 데이터를 읽어와 자신이 속한 카탈로그 BOUNDED CONTEXT(카탈로그 도메인)에 맞는 상품 모델로 변환합니다. 다음은 일부 코드를 가상으로 만들어 본 것입니다.

```
public class RecSystemClient implements ProductRecommendationService {

  private ProductRepository productRepository;

  @Override
  public List<Product> getRecommendationOf(ProductId id) {
    List<RecommendationItem> items = getRecItems(id.getValue());
    return toProducts(items);
  }

  private List<RecommendationItem> getRecItems(String itemId) {
    // externalRecClient는 외부 추천 시스템을 위한 클라이언트라고 가정
    return externalRecClient.getRecs(itemId);
  }

  ...
}
```

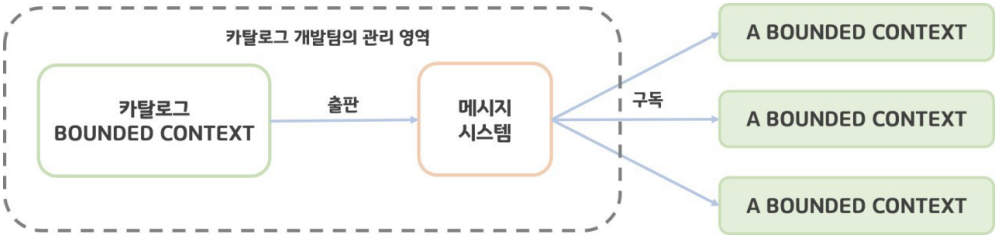
RecSystemClient는 추천 시스템의 모델을 받아와 toProducts() 메서드를 이용해서 카탈로그 도메인의 Product 모델로 변환하는 작업을 처리합니다. 두 모델 간의 변환 과정이 복잡하면 아래 그림과 같이 변환 처리를 위한 별도 클래스를 이 클래스에서 변환을 처리해도 됩니다.



REST API를 호출하는 것은 두 BOUNDED CONTEXT를 직접 통합하는 방법입니다. 이에 반해 간접 통합 방식인 **메시지 큐를 사용하는 방법**이 있습니다. 메시지 큐는 보통 비동기로 메시지를 처리하기 때문에 메시지를 송신한 BOUNDED CONTEXT는 다른 BOUNDED CONTEXT가 메시지를 처리하는것을 기다리지 않고 바로 이어서 자신의 처리를 계속합니다.

두 BOUNDED CONTEXT를 개발하는 팀은 메시지 큐에 담을 데이터의 구조를 협의 하게 되는데 그 큐를 누가 제공하느냐에 따라 데이터 구조가 결정됩니다. 예를 들어, 카탈로그 시스템에서 큐를 제공한다면 담기는 내용은 카탈로그 도메인을 따릅니다. 그렇기 때문에 추천 시스템에서 이 큐로부터 필요한 메시지를 수신한다면 추천 도메인 모델에 맞게 변형하여 사용해야합니다.

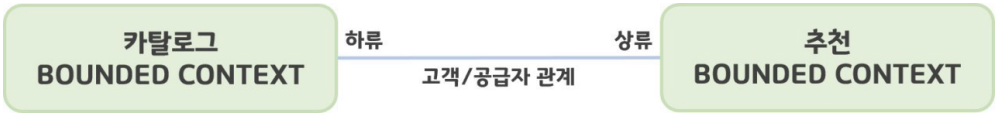
하나의 시스템에서 큐로 데이터를 송신하고 다른 시스템들에서 이를 수신해가는 특징으로 인해 이 방식은 **출판/구독 모델**을 따릅니다.



BOUNDED CONTEXT 간 관계

고객/공급자 관계

BOUNDED CONTEXT는 어떤 식으로든 연결되기 때문에 두 BOUNDED CONTEXT는 다양한 방식으로 관계를 맺습니다. 두 BOUNDED CONTEXT 간 관계 중 가장 흔한 관계는 한쪽에서 API를 제공하고 다른 한쪽에서 그 API를 호출하는 관계입니다. 이 관계에서 API를 사용하는 BOUNDED CONTEXT는 API를 제공하는 BOUNDED CONTEXT에 의존하게 됩니다.

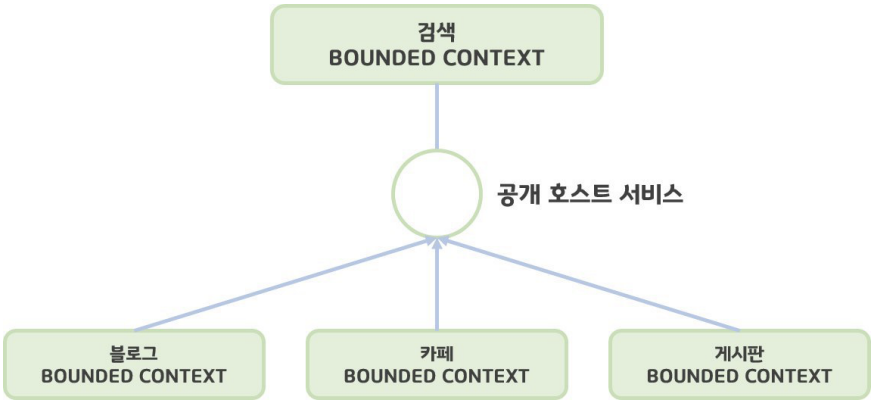


위의 그림에서 **하류(downstream) 컴포넌트**인 카탈로그 컨텍스트는 **상류(upstream) 컴포넌트**인 추천 컨텍스트가 제공하는 데이터와 기능에 의존합니다. 때문에 추천 시스템이 제공하는 REST API의 인터페이스가 바뀌면 카탈로그 시스템의 코드도 바뀌게 됩니다.

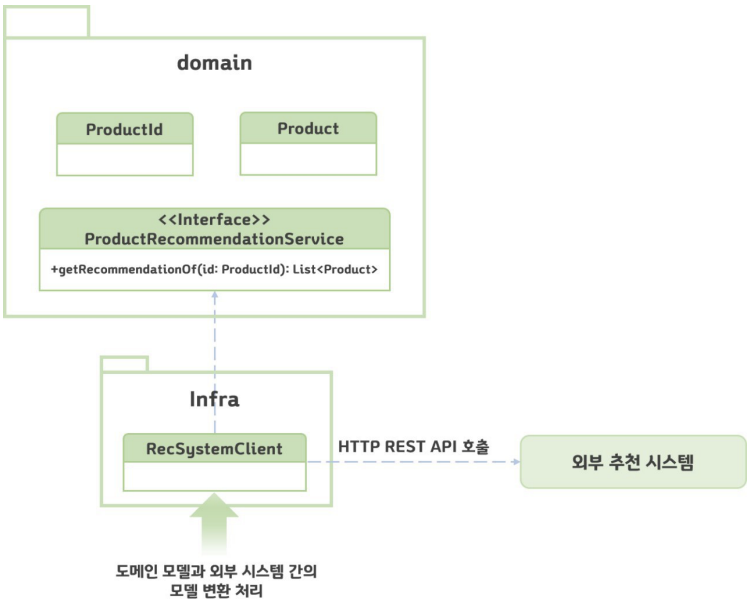
상류 컴포넌트는 일종의 서비스 **공급자 역할**을 하며, 하류 컴포넌트는 그 서비스를 사용하는 **고객 역할**을 합니다. 고객과 공급자 관계에 있는 두 팀은 의존 관계에 있기 때문에 상류 팀과 하류 팀은 개발 계획을 서로 공유하고 일정을 협의해서 결정해야 합니다.

상류 팀의 고객인 하류 팀이 다수 존재하면 상류 팀은 여러 하류 팀의 요구사항을 수용할 수 있는 API를 만들고 이를 서비스 형태로 공개해서 서비스의 일관성을 유지할 수 있습니다. 이런 서비스를 가리켜 **공개 호스트 서비스(OPEN HOST SERVICE)**라고 합니다.

공개 호스트 서비스의 대표적인 예가 검색입니다. 블로그, 카페, 게시판과 같은 서비스를 제공하는 포탈은 각 서비스별로 검색 기능을 구현하기보다는 검색을 위한 전용 시스템을 구축하고 검색 시스템과 각 서비스를 통합 합니다. 이때, 검색 시스템은 상류 컴포넌트가 되고 블로그, 카페, 게시판은 하류 컴포넌트가 됩니다. 상류 팀은 각 하류 컴포넌트의 요구사항을 수용하는 단일 API를 만들어 이를 공개하고 각 하류 팀은 공개된 API를 사용해서 검색 기능을 구현하게 됩니다.



아래 그림에서 RecSystemClient는 외부 시스템과의 연동을 처리를 함으로써 외부 시스템의 도메인 모델이 내 도메인 모델을 침범하지 않도록 막아주는 역할을 합니다. 즉, 내 모델이 깨지는 것을 막아 주는 **안티코럽션 계층(Anticorruption Layer)**이 됩니다. 이 계층에서 두 BOUNDED CONTEXT 간의 모델 변환을 처리해 주기 때문에 다른 BOUNDED CONTEXT의 모델에 영향을 받지 않고 내 도메인 모델을 유지할 수 있습니다.



공유 커널

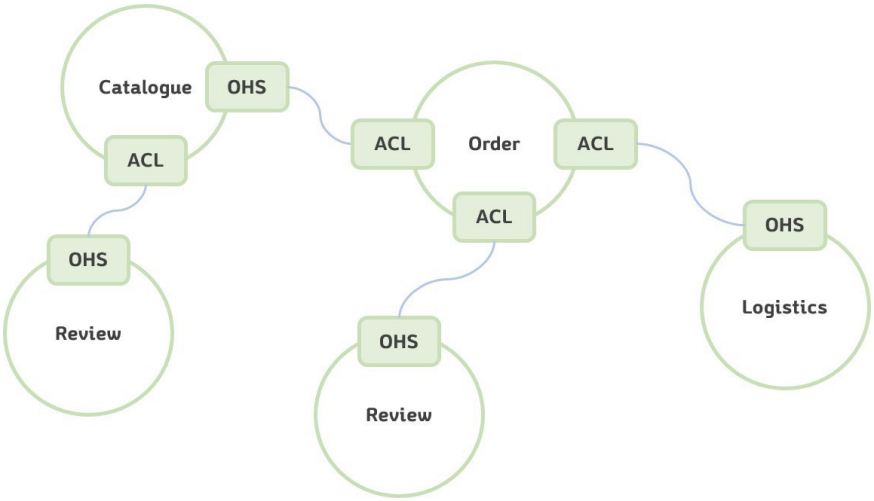
예를 들어, 운영자를 위한 주문 관리 도구를 개발하는 팀과 고객을 위한 주문 서비스를 개발하는 팀이 다르다고 가정합시다. 이 경우, 두 팀은 주문을 표현하는 모델을 공유함으로써 주문과 관련된 중복 개발을 막을 수 있습니다. 이렇게 두 팀이 공유하는 모델을 **공유 커널(SHARED KERNEL)**이라고 부릅니다.

독립 방식 관계

독립 방식 관계(SEPERATE WAY)는 그냥 서로 통합하지 않는 방식입니다. 예를 들어, 온라인 쇼핑몰 솔루션과 외부의 ERP 서비스를 사용하고 있다고 합시다. 온라인 쇼핑몰 솔루션은 외부 ERP 서비스와의 연동을 지원하지 않으므로 온라인 쇼핑몰에서 판매가 발생하면 쇼핑몰 운영자는 쇼핑몰 시스템에서 판매 정보를 보고 ERP 시스템에 입력해야 합니다.

컨텍스트 맵

개별 BOUNDED CONTEXT에 집중하다 보면 전체를 보지 못할 때가 있습니다. 나무만 보고 숲을 보지 못하는 상황을 방지하려면 전체 비즈니스를 조망할 수 있는 지도가 필요한데, 그것이 바로 **컨텍스트 맵**입니다. 컨텍스트 맵은 아래의 그림처럼 BOUNDED CONTEXT 간의 관계를 표시한 것입니다.



BOUNDED CONTEXT 영역에 주요 애그리거트를 함께 표시하면 모델에 대한 관계가 더 명확히 드러납니다. 그림에는 오픈 호스트 서비스(OHS)와 안티코럽션 계층(ACL)만 표시했는데 하위 도메인이나 조직 구조를 함께 표시하면 도메인을 포함한 전체 관계를 이해하는 데 도움이 됩니다.