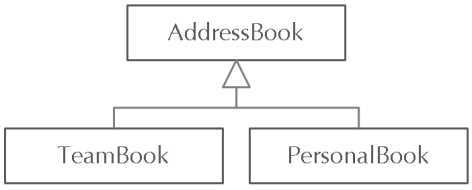
Drama-address-book 모델링/코딩 아이디어

2022. 7. 11.
송태국(tsong@nextree.io)

→ Domain Modeling Idea

1 초기 요구는 개인 주소록(Address book)을 개발하는 것이었습니다. 아마존에서 사용자에게 제공하는 주소록을 참조하여 개발하였습니다. 초기 모델은 AddressBook, Address 값객체(Value Object)를 갖고 있는 AddressPage 세 개의 도메인 객체로 시작하였습니다.

2 개발 진행 중에, NARA 플랫폼은 엔터프라이즈 환경에서 주로 사용하므로 어떤 서비스를 기획하든 팀이 사용하는 경우를 고려해야 하고, 따라서 팀 주소록(Team addressbook)이 필요하다는 의견이 나왔습니다. 그러면, 다음과 같이 모델을 확장하였습니다. AddressBook에 따라 AddressPage도 동일한 계층구조를 가집니다.



[ 상속계층을 가진 도메인 모델 ]

3 상속계층을 가진 도메인 모델을 구현할 때는 항상 고민이 따릅니다. 상속구조 매핑과 관련된 흔한 고민들이 있습니다. 상속을 계층을 테이블로 어떻게 매핑할 것인가라는 고민이 그것입니다. 하지만, NARA Way 스타일은 도메인 모델에서 상속을 최대한 자제합니다. 그렇다면 위의 상속으로 표현하는 위의 개념을 어떻게 모델링하면 좋을까요?

4 위 상속계층을 하나의 테이블로 매핑, 두 개의 테이블로 매핑, 세 개의 테이블로 매핑하는 방법이 있습니다. 우리는 그런 테이블 매핑에 대해 고민하지 않기로 했습니다. 대신 TeamBook과 PersonalBook의 특징과 차이에 대해서 고민을 했습니다. 다음과 같은 차이점들이 있습니다.
   4.1 PersonalBook 은 개인별로 하나씩 갖고 있습니다. NARA 기준으로 보면, Citizen 별로 하나씩.
   4.2 PersonalBook은 Owner 역할이 사용할 수 있습니다.
   4.3 TeamBook은 사업부별로 하나씩 갖고 있습니다. NARA 기준으로 보면, Cineroom 별로 하나씩.
   4.4 TeamBook은 TeamAdmin, TeamMember 두 가지 역할이 사용할 수 있습니다.
   4.5 회사에서 직원들이 공유하는 주소록은 그룹웨어를 사용하는 Cineroom 에서 정의하여 사용합니다.

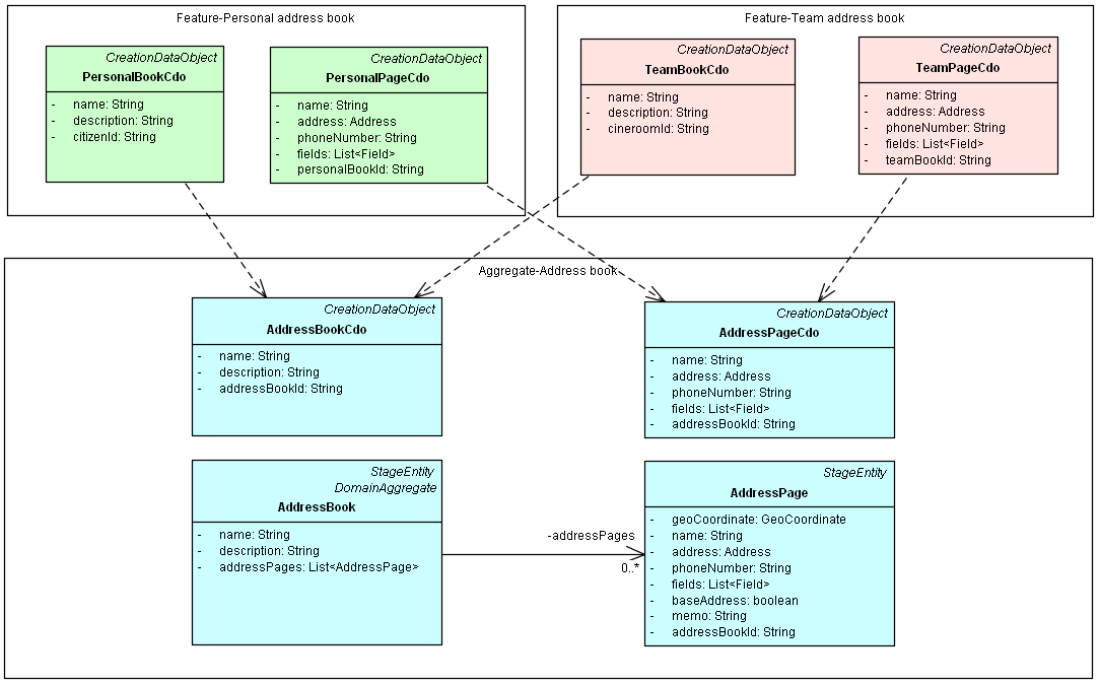
5 위 상속계층을 모두 Aggregate 모듈에서 표현하면 클래스 개수가 두 배가 되고, TeamBook의 경우 사용하지 않더라도 테이블이 생성되는 문제가 있습니다. 범용 드라마(Public drama)의 경우, 이런 부분에 주의하여야 합니다. 따라서 우리는 AddressBook은 저장을 위한 개념으로 정의하고 Aggregate에 그대로 둡니다. 대신 TeamBook과 PersonalBook은 서로 다른 정책(ID 규칙이 서로 다르고, 역할과 접근권한이 서로 다름)으로 보고, 서로 다름은 Feature 모듈에서 표현합니다.

6 그러면 Feature 모듈에 두 가지 Feature를 정의합니다. Teambook, personalbook을 정의하고 각 주소록이 가지는 차이(규칙과 권한,그리고 로직)를 각 Feature에서 처리합니다.

7 이를 위해서 각 Feature 도메인에서는 필요한 최소한의 도메인 객체(sdo, vo)를 정의합니다.
   7.1 TeamBook 피처: TeamBookCdo, TeamPageCdo
   7.2 PersonalBook 피처: PersonalBookCdo, PersonalPageCdo
   7.3 Feature에서 최소한의 도메인 정의 원칙에 따라 각 Rdo(TeamBookRdo, PersonalBookRdo)는 정의하지 않고 AddressBook을 그대로 사용합니다. 이유는 완전히 중복되기 때문입니다. 만약에 이런 부분조차 혼란스럽다고 한다면 new TeamBookRdo(AddressBook) 반식으로 의미없는 Wrapper 객체를 만들어야 합니다.
   7.4 API 클라이언트 입장에서 보면 생성할 때는 TeamBookCdo를 전송했는데 생성된 결과를 조회할 때는 AddressBook으로 리턴되는 것이 혼란스러울 수 있습니다. 하지만 “is a” 관계를에 의하여 TeamBook is an AddressBook 관계가 성립하므로 잘못된 것은 아닙니다. 생성 규칙은 다른데, 결과는

동일한 주소록이다로 해석하면 될 것 같습니다. 그리고 프론트의 한 UI 컴포넌트에서 두 가지를 동시에 사용하는 경우는 없기 때문에, AddressBook, AddressPage 등을 리턴하여도 문제가 되지 않습니다.

8 최종 Feature 도메인 모델은 다음과 같습니다. 이 결과로 TeamBook과 PersonalBook은 서로 다른 ID 규칙, 서로 다른 역할과 접근 등을 Feature 모듈 수준에서 각각 구현하므로 코드 간결성 목표를 달성하였습니다.



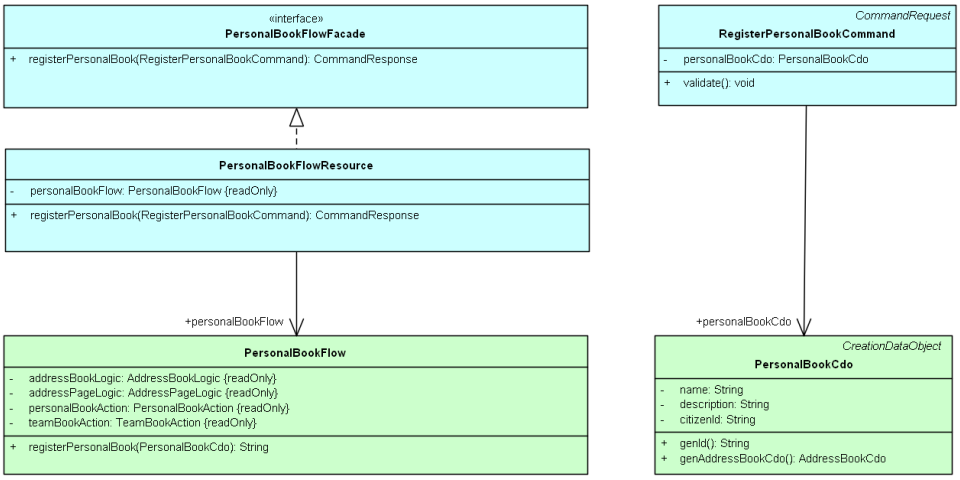

[ Address book' domain model ]

→ Command/Query Naming 규칙

1 NARA Way에서 REST API 설계는 API Operation 기반이 아니라 API 메시지 기반입니다. Command와 Qeury 메시지 모두 API Request로 간주합니다.

2 Façade 모듈에서 정의한 Command와 Query는 Feature 모듈의 Flow/Seek 로직까지 일관된 Naming 규칙의 근간이 됩니다. 따라서 Command/Query 요청 객체의 이름은 목적을 명확하게 보여줄 수 있도록 작성해야 합니다.

3 RegisterPersonalBookCommand를 예제로 REST API, 리소스, Feature 로직 간의 이름 연관성을 살펴봅니다. Command는 CUD 관련 로직을 다루며, 로직 이름은 업무처리 흐름을 표현하므로 "Flow"입니다.
- 3.1 RegisterPersonalBookCommand를 정의합니다.
- 3.2 API 오퍼레이션: PersonalBookFlowFacade.registerPersonalBook(RegisterPersonalBookCommand);
- 3.3 REST 리소스: PersonalBookFlowResource.registerPersonalBook(위와 동일);
- 3.4 API 오퍼레이션과 REST 리소스는 Realization 관계이므로 동일한 메소드 시스너처를 가집니다.
- 3.5 피처로직: PersonalBookFlow.registerPersonalBook(PersonalBookCdo);
- 3.6 피처로직의 패러미터는 RegisterPersonalBookCommand 안에 있는 PersonalBookCdo 입니다.
- 3.7 Command의 속성을 정의할 때, 한 번에 사용하는 속성이 세 개를 넘지 않도록 주의합니다.




4 FindPersonalBookQuery 역시 위와 동일한 Naming 규칙을 사용합니다. 찾기 로직은 Seek를 사용합니다.
- 4.1 FindPersonalBookQuery를 정의합니다.
- 4.2 API 오퍼레이션: PersonalBookSeekFacade.findPersonalBook(FindPersonalBookQuery);
- 4.3 REST 리소스: PersonalBookSeekResource.findPersonalBook(위와 동일);
- 4.4 API 오퍼레이션과 REST 리소스는 Realization 관계이므로 동일한 메소드 시스너처를 가집니다.
- 4.5 피처로직: PersonalBookSeek.findPersonalBook(String personalBookId);
- 4.6 피처로직의 패러미터는 FindPersonalBookQuery 안에 있는 personalBookId 입니다.

5 이런 스타일의 Naming 규칙은 코드 자동화를 가능하게 합니다. 다음과 같은 두 가지 자동화가 가능합니다.
- 5.1 Feature 이름을 받아서 CUD 커맨드(Register/Modify/Remove)와 Find 쿼리를 자동으로 생성할 수 있습니다. 개발자들이 쉽게 잊을 수 있는 부분을 자동으로 생성하여, 시간도 절약하고 품질도 높일 수 있습니다.
- 5.2 Command/Query를 선택하여 세 가지 클래스와 오퍼레이션 시그너처를 생성할 수 있습니다.
  - 5.2.1 [Feature이름]FlowFacade.java 와 오퍼레이션 시그너처
  - 5.2.2 [Feature이름]FlowResource.java 와 오퍼레이션 바디, 그리고 //FixMe 주석
  - 5.2.3 [Feature이름]Flow.java 와 오퍼레이션 바디, 그리고 //FixMe 주석

[끝]