04 도메인의 격리

모델의 각 요소(객체)를 하나하나 구분하지 않고 **하나의 시스템**으로 바라 볼 수 있도록 시스템에서 도메인과 **관련이 적은 기능**으로부터 도메인 객체를 분리

Domain-Driven Design, Modeling, Design

Layered Architecture

Pattern-Oriented Software Architecture Volume 1: A System of Patterns. pp. 31 ~ 51 참고



Model-Driven Design을 가능케 하는 것은 결정적으로 **도메인 계층을 분리**하는 데 있다.



복잡한 프로그램을 여러 개의 계층으로 나눠라 (같은 계층에서는) 응집력 있고 오직 "**아래**"에 위치한 계층에만 의존하는 각 계층에서 설계를 발전시켜라 표준 아키텍처 패턴에 따라 상위 계층과의 결합을 느슨하게 유지하라 **도메인 모델과 관련된 코드는 모두 한 계층에 모으고** 사용자 인터페이스 코드나 애플리케이션 코드, 인프라스트럭처 코드와 **격리**하라

계층화의 핵심 원칙

계층화의 핵심 원칙은

한 계층의 모든 요소는 **오직 같은 계층에 존재하는 다른 요소** 혹은 **계층상 "아래"에 위치한 요소**에만 의존한다는 것이다.

위로 거슬러 올라가는 의사소통은 반드시 간접적인 메커니즘을 거쳐야 한다.

계층화의 가치

계층화의 가치는 각 계층에서 컴퓨터 프로그램의 **특정 측면만을 전문적을 다룬다**는 데 있다. …

거듭 말하지만 경험과 관례를 바탕으로 널리 받아들여지는 계층화가 어느 정도 정해졌다. … 대다수의 성공적인 아키텍처에서는 아래의 네 가지 개념적 계층으로 나뉜다.

계층	설명
User Interface (or Presentation Layer)	사용자에게 정보를 **보여주고** 사용자의 명령을 **해석**하는 일을 책임진다. 간혹 사람이 아닌 **다른 컴퓨터 시스템이 외부 행위자**가 되기도 한다.
Application Layer	소프트웨어가 **수행할 작업을 정의**하고 표현력 있는 도메인 객체가 **문제를 해결하게 한다.** 이 계층에서 책임지는 작업은 **업무상 중요**하거나 다른 시스템의 **응용 계층과 상호작용**하는 데 필요한 것이다. 이 계층은 얇게 유지된다. 여기에는 **업무 규칙이나 지식이 포함되지 않으며**, 오직 작업을 **조정**하고 아래에 위치한 계층에 포함된 도메인 객체의 협력자에게 작업을 **위임**한다. 응용 계층에서는 업무 상황을 반영하는 상태는 없지만 사용자나 프로그램의 작업에 대한 **진행상황을 반영하는 상태**를 가질 수는 있다.

계층	설명
Domain Layer (or Model Layer)	**업무 개념**과 **업무 상황에 관한 정보**, **업무 규칙**을 표현하는 일을 책임진다. 이 계층에서는 **업무 상황을 반영하는 상태를 제어하고 사용하며**, 그와 같은 상태 저장과 관련된 기술적인 세부사항은 **인프라스트럭처에 위임**한다. 이 계층은 **업무용 소프트의 핵심**이다.
Infrastructure Layer	상위 계층을 지원하는 **일반화된 기술적 기능**을 제공한다. 이러한 기능에는 애플리케이션에 대한 메시지 전송, 도메인 영속화, UI에 위젯을 그리는 것 등이 있다. 또한 인프라스트럭처 계층은 아키텍처 프레임워크를 통해 **네 가지 계층에 대한 상호작용 패턴을 지원**할 수도 있다.

계층 간 관계 설정



각 계층은 설계 의존성을 **오직 한 방향**으로만 둬서 느슨하게 결합된다. (상위 → 하위)

- 상위 → 하위 : 하위 계층에 대한 참조를 갖고(최소한 임시로라도) 직접 호출
- 하위 → 상위 : Callback 이나 Obeserver Pattern으로 간접 호출

응용 계층 → 도메인 계층

응용 계층과 도메인 계층에 UI를 연결하는 패턴은 MVC에서 유래한다. 1970년 Smalltalk 분야에서 발견되어 MVC를 따르는 여러 UI 아키텍처에 영감을 준다.

- 패턴과 참고 서적
  - MVC
    - Patterns of Enterprise Application Architecture (Martin Fowler 2003)
  - MVP
    - Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (Craig Larman 1998)
      - Application coordinator pattern ☞ 애플리케이션 계층을 연결하는 접근법 가운데 하나

도메인 계층 → 인프라스트럭처 계층

인프라스트럭처 계층은 도메인 계층의 하위 계층이므로 **도메인의 어떤 지식**을 갖고, **도메인에 관련된 어떤 활동**이 일어나면 안된다. 사실 도메인의 지식/활동 같은 기술적인 기능은 대개 **SERVICE**로 제공된다. 분리의 주된 이점은 애플리케이션 계층이 **단순**해져서 애플리케이션 **본연의 책임에만 집중**하게 되는 것이며 이로써 메시지를 **"언제"** 보내는지는 알아도 **"어떻게"** 보내는지는 알 필요가 없어진다.

응용 계층과 도메인 계층에서는 인프라스트럭처 계층에서 제공하는 SERVICE를 요청한다. 그러나 모든 인프라스트럭처 계층이 상위 계층에서 호출할 수 있는 SERVICE의 형태로 만들어지는 것은 아니다. 어떤 기술적인 구성요소는 다른 계층의 기본적인 기능을 직접적으로 지원하도록 만들어져(이를 테면, 모든 도메인 객체에 대한 Abstract class를 제공하는 것과 같이) 그러한 계층과 관계를 맺는 메커니즘(MVC 및 그와 비슷한 패턴의 구현과 같은)을 제공하기도 한다. 이러한 'Architectural Framework'는 프로그램의 다른 부분의 설계에 훨씬 더 많은 영향을 미친다.

Architectural Framework

도메인 계층은 모델이 살아가는 곳

The Smart UI "Anti-Pattern"

지능형 UI “안티 패턴”

다른 종류의 격리

From:
https://wiki.ledyx.xyz/ - **Ledyx WIKI**

Permanent link:
https://wiki.ledyx.xyz/doku.php?id=doamin-driven_design:04_isolating_the_domain&rev=1691753561

Last update: **2026/10/24 07:03**