Clean Code

11. 시스템 (2)

알파서클 연구원
Luna

순수한 언어 코드를 위한 AOP 도구

- 외부 라이브러리나 프레임워크를 이용하기 위한 의존 코드가 늘어나면 그에 따른 문법과 사용법을 익혀야 하며 복잡해지고 유지보수와 테스트가 어려워진다.
- (라이브러리)프레임워크가 제공해주는 AOP 도구를 이용해 순수한 언어 코드 자체를 깔끔하고 깨끗하게 유지하는 것이 좋다.

데코레이터를 이용한 외부 라이브러리 의존 코드 처리

```
@Entity()
export class User extends CoreEntity {
  @ApiProperty(...)
  @IsEmail()
  @Column({ unique: true })
  email: string;

  @ApiProperty(...)
  @IsNotEmpty()
  @Column()
  name: string;
}
```

```
// 라이브러리 의존 코드가 존재할 때 예상 코드
export class User extends CoreEntity {
  TypeORM.addEntity(this)

  email: string;

  Swagger.Property.Add(email, ...)
  Validater.isEmail(email)
  TypeORM.addColumn(email, ...)
}
```

더 좋은 시스템을 위해

- 코드에서 관점, 관심사를 분리해 단순하고 순수한 언어 코드로 남기면 변경으로부터 드는 비용도 적어지며 확장에도 유연해진다.
- 필수적인 구조, 범위, 목표를 정해놓고 단순하고 빠르게 구현한 뒤에 변하는 환경에 능동적으로 대처하는 것이 효율적이다.
- 표준을 사용하면 물론 좋지만 항상 옳지만은 않을 수도 있으며 명백한 가치를 주는지 판단하는 것이 좋다.
- 도메인 특화 언어를 사용하면 간결하지만 효과적으로 정보를 전달할 수 있다.
    - 적절한 추상화 수준에서 의도를 표현할 수 있다.
- 깨끗하지 못한 아키텍처에선 도메인 논리가 흐려지며 버그가 숨겨지고 테스트 효율이 떨어진다.
- 모든 추상화 단계에서 의도는 명확하고 단순해야 한다.