Project Guide - 설명편 (1)

[서론] Project 기본구조 설명

아키텍쳐.pptx

목표

```
main/
├─ build.gradle            // 공통적인 설정 파일
├─ approval/               // Application 없음
│  └─ build.gradle
├─ authorization/          // Application 없음
│  └─ build.gradle
└─ core/                   // Application 존재
   └─ build.gradle
```

* Application : 프로젝트 실행 주체

각 폴더별 설명

1. core project

> core 시스템은 **비즈니스 로직이 아닌, 서비스에 필요한 공통/핵심적인 기능**들이 존재합니다.

- spring 기반 시스템이 실행되기 위한 공통적인 설정

  → `Config`와 `Application`파일이 존재 (프로그램 실행의 주체)
- 서비스 통신을 위해 기본적으로 구현된 기능
  - Logging
  - 계정계에서 공통으로 필요한 전역 시스템 변수
  - token을 통한 인증/인가 로직
  - transaction 관리

2. `main project`

> 비즈니스(`authorization` & `approval`)와 core 같은 프로젝트를 하나로 묶어주는 프로젝트
> → 실행코드가 없는 껍데기 같은 존재

💡 main은 아무 기능도 없는데 왜 있어야 할까?

> 아래와 같은 기능을 합니다.

- 여러 프로젝트의 **설정 중앙 관리**

→ approval, authorization, core 에 공통적으로 필요한 dependency들을 `main/build.gradle`에서 한번만 적고, 각각의 subproject에 흘려보내어, 중복된 설정을 하지 않아도 됩니다.

→ 단, 특정 subproject에만 필요한 설정은 개별로 넣어주면 됩니다.

ex. `authorization/build.gradle`

- **core JAR 패키징**

→ 위의 내용처럼, 실제로 서버를 띄울 때는 `core/Application.java`를 사용합니다. main에 core JAR를 공통으로 둔다면 비즈니스는 중복으로 core JAR를 갖거나, 그 안의 내용을 작성하지 않아도 됩니다.

- Scalability(확장성)

→ 다른 서브 프로젝트가 늘어나도 main에서 include만 해주면 다함께 관리 가능합니다.

→ 또한 git 관리가 독립적으로 이루어져, 협업 시에도 충돌이 적습니다.

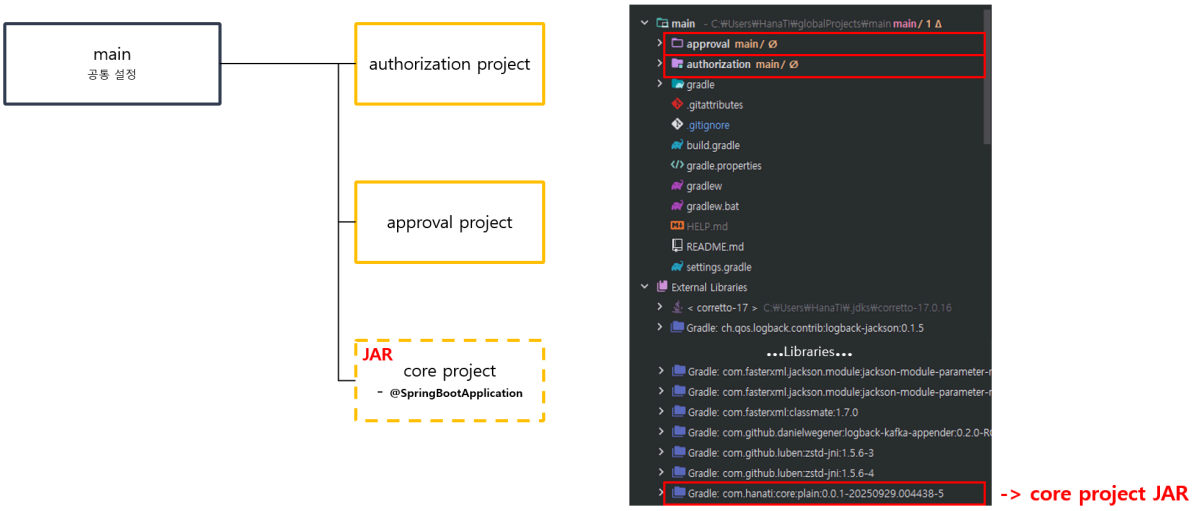
3. authorization project

> 결재 domain

4. approval project

> 여신 신청/심사/승인 domain

구조도



문서 “Project Guide - Setting편 (2)” 을 따라하면, 위 사진과 같이 구성됩니다.