

초격차 패키지 Online.

# 안녕하세요. Final 구글 캘린더 클론 프로젝트 정채균입니다.

## Chapter 1 | 강의 소개

강사 소개 및 프로젝트 개요

## Chapter 4 | 2차 요구사항 개발

캘린더 공유하기 요구사항을 추가 구현합니다.

## Chapter 2 | 1차 요구사항 개발

기획서를 이해하고 핵심 도메인을 개발합니다.

## Chapter 3 | 알람 배치 시스템 개발

시간에 맞춰 이메일을 전송하는 배치 시스템을 개발합니다.

# 강의 소개

## 1 강사 소개 및 프로젝트 개요

# 강사 소개

## 강사 소개

백엔드 개발자로 약 3년차 주니어  
현재 카카오 자회사에서 근무  
자바 개발자로 시작, 현재는 코틀린 Lover

객체지향, 함수형 둘다 좋아하고 함수형으로 넘어가기 위해  
열심히 공부중

**실무자 관점에서 고민하는 부분을 솔직하게 나눠보려 합니다.**  
**강의를 통해 함께 배워가는 유익한 시간이 됩시다!**

# 프로젝트 소개

구글 캘린더 기능 구현 맛보기



## 구글 캘린더 살펴보기

The image displays the Google Calendar interface. On the left, a vertical app drawer lists various Google services: Account, Search, Maps, YouTube, Play, News, Gmail, Meet, Chat, Contacts, Drive, and Calendar. The 'Calendar' icon, which shows a calendar page with the number 31, is highlighted with a red rectangular box. On the right, the main calendar view is shown for the date 2021년 7월 24일 (July 24, 2021). The view includes a weekly calendar grid on the left and a detailed hourly view on the right. The hourly view shows a green event bar for '오전 8시 ~ 9시' (8 AM ~ 9 AM) and a red event bar for '오전 10시' (10 AM). The interface also includes a search bar, a '만들기' (Create) button, and a list of '내 캘린더' (My Calendars) and '다른 캘린더' (Other Calendars).

구글 캘린더

오늘 < > 2021년 7월 24일

만들기

2021년 7월

GMT+09

알림 2개

오전 1시

오전 2시

오전 3시

오전 4시

오전 5시

오전 6시

오전 7시

오전 8시 ~ 9시

오전 9시

오전 10시

오전 11시

오후 12시

내 캘린더

- ☒ [Event]
- ☒ 생일
- ☒ 알림
- ☐ Tasks

다른 캘린더

- ☒ [Event]
- ☒ 대한민국의 휴일
- ☒ Holidays in South Korea

## 사실은 맛보기 입니다. & 몇가지 문답

10시간에 캘린더 api를 다 만들 수 있다면?

- 구글에 취업하겠지..?

이 캘린더 기능구현을 해보는 이유가 뭘니까?

- 흔한 주제는 아니기 때문입니다. (단골주제: 게시판, 주문시스템, 도서관리)
- 맛보기라 할지라도 실제 프로젝트의 요구사항을 만들어보기 때문
- 쉬워보이는데 그렇게 쉽지 않다..!
- 준비하는 저도 고민할 부분이 많았습니다.

## 사실은 맛보기 입니다. & 몇가지 문답

무엇을 배울 수 있을까요?

- 강사의 코딩 습관..?
- 스프링 사용 방법(스프링의 구조를 뜯어보고 파악하는 습관)
- 자바로 코드 추상화 방식
- 도메인 설계 과정 엿보기
- 코드 리팩토링 포인트, 구조적인 개선 사항 도출해보기
- 트러블 슈팅 능력
- 추가 요구사항 대응하는 방식

이런건 배워가지 마세요!

- 라이브 코딩할때 머뭇머뭇 할 때..? 헛소리..?
- 강사의 주관적인 관점이 주입된다고 느껴질 때
- 더 나은 방식이 있다면 시도해도 좋습니다!



## 학습 개념

### Java

- 초급~중급 수준의 OOP, 디자인패턴
- **Java Stream**을 사용한 함수형 프로그래밍 예제
- 테스트 기반한 도메인 설계

### Spring

- MVC 개발 방식
- 계층 구조, 계층 간 의존성 관리
- 템플릿 엔진을 이용한 화면 구현
- **Custom ArgumentResolver** 을 통한 인증처리
- 글로벌 에러 처리
- **JPA**를 이용한 테이블 모델링
- **MailSender** 를 사용한 메일발송
- **Spring Validator** 를 사용한 데이터 검증 & 에러 처리

### 기타

- 암호화 데이터 저장, 검증

## 학습도 중요하지만 개발자는..

주어진 요구사항을 쿵쾅 잘 만들되,

- 함께 일하는 기획자와 원활한 소통
- 협업하는 개발자들과 가벼운, 심도있는 개발 이야기
- 내가 만들고 있는 프로젝트를 잘 가꿔내기
  - 견고한 시스템 만들기
  - 기술부채 줄이기

## 강의에 쓰일 기술 스택

### JDK 11

### Spring Boot 2.5.2

- Spring MVC
- Spring JPA
- Spring Batch
- Spring Mail

### 기타

- thymeleaf (템플릿 엔진)
- jbcrypt (패스워드 암호화)
- lombok
- junit (테스트)
- gradle (프로젝트 관리)
- docker (mysql)
- git

## 로컬 환경

OS: MAC (window로 하셔도 무관합니다.)

IDE: IntelliJ

## 먼저 요구되는 지식 (입문 강의 수준)

자바

Spring MVC

Spring JPA

Spring Batch

MySQL

docker

HTTP

html, css, js 기초

git

## 강의에 사용되는 소스

github: <https://github.com/LarryJung/final-project-calendar>

# 1차 요구사항 개발

1 기획서 이해

# 요구사항 분석



## 캘린더의 핵심 기능 구현

회원: 가입, 로그인, 로그아웃

The screenshot displays the Fast Campus calendar application. At the top, there's a navigation bar with a menu icon, the text '캘린더', a date selector for '2021년 7월', and a view selector '조회: 일, 주, 월 별 조회'. A user profile icon is circled in red in the top right corner. On the left, a sidebar shows '2021년 7월' and a list of calendar types: '내 캘린더' (My Calendar) with checkboxes for '생일' (Birthday), '알림' (Notification), and 'Tasks', and '다른 캘린더' (Other Calendar) with checkboxes for '대한민국의 휴일' (Korean Holidays) and 'Holidays in South Korea'. The main calendar grid shows dates from 28 to 31. A modal window is open for creating an event, with the title '제목 및 시간 추가' circled in red. The modal includes options for '이벤트' (Event), '할 일' (To-do), and '알림' (Notification), a date range selector for '7월 9일 (금요일) - 7월 10일 (금요일)', a '시간 추가' (Add Time) button, a '참석자 추가' (Add Attendees) section with a 'Google Meet 화상 회의 추가' (Add Google Meet Video Conference) button, a '위치 추가' (Add Location) section, a '설명 또는 첨부파일 추가' (Add Description or Attachment) section, and a list of participants including 'Larry Jung'. The modal also has '옵션 더보기' (More Options) and '저장' (Save) buttons. A red box highlights the main calendar area and the event creation modal. Text overlays on the image include '생성, 수정, 취소: 이벤트, 할 일, 알림' (Create, Edit, Cancel: Event, To-do, Notification) and '공유: 캘린더 공유(양방향, 단방향)' (Share: Calendar Share (Two-way, One-way)).

이벤트 수락, 거부 기능  
메일 전송 기능  
이벤트 알림 배치 기능  
등등..

## 요구사항

### 회원

- 간단한 회원가입이 가능하다. (email, pw, name, birthday)
- 비밀번호는 **암호화** 하여 저장한다.
- 수정 및 탈퇴는 불가능하다.

### 권한

- 가입을 해야 서비스 이용 가능하다.
- 자신의 캘린더만 조회 가능하다.

## 요구사항

### 일정

- 3가지 일정을 만들 수 있다. (이벤트, 할일, 알림)

### 이벤트

- 기간으로 등록
- **참석자 여러명**
- 위치정보는 없음
- 등록하면 참석자에게 **초대 메일**이 전송된다.
- 참석자는 메일을 통해 **수락, 거절**이 가능하다.
- 수락여부 파악 가능하다.
- **시간이 겹치지 않는 회원**만 초대할 수 있다.

제목 및 시간 추가

이벤트 할 일 알림

7월 9일 (금요일) - 7월 9일 (금요일) 시간 추가  
반복 안함

시간 찾기

참석자 추가 가능한 참석자 조회 API (스펙 아웃)

Google Meet 화상 회의 추가

위치 추가 가능한 위치(회의실 등) 정보 조회 API (스펙 아웃)

설명 또는 첨부파일 추가

Larry Jung  
한가함 · 기본 공개 설정 · 알림 2개

옵션 더보기 저장

저장 클릭 시 생성 API  
호출

## 요구사항

### 할일

- 할일은 자신만 등록 가능.
- 시간 등록(optional), 디폴트는 자정

제목 추가

이벤트 할 일 알림

🕒 2021년 7월 9일 시간 추가

≡ 설명 추가

☰ My Tasks ▼

저장

저장 클릭 시 생성 API 호출

## 요구사항

### 알림

- 할일과 비슷하나 알림 반복기능이 추가된다.
- **반복기능**
  - **반복 주기**에 따라 입력 데이터가 다름 (일,주,개월)
  - 예) 반복주기 2,주 / 요일 금 / 종료 13회
  - => 2주차 금요일마다 알림전송 13회
  - 월 반복주기는 요일별,일별 두가지 조건이 있으나 일별로만..

반복 설정

반복 주기 1 일 ▼

종료

☒ 없음

☐ 날짜: 2021년 8월 8일

☐ 다음 30 회 반복

취소 완료

반복 설정

반복 주기 1 주 ▼

반복 요일

월 화 수 목 금 토 일

종료

☒ 없음

☐ 날짜: 2021년 10월 8일

☐ 다음 13 회 반복

취소 완료

반복 설정

반복 주기 1 개월 ▼

매월 9일 ▼

종료

☒ 없음

☐ 날짜: 2022년 7월 9일

☐ 다음 12 회 반복

취소 완료

### 제목 추가

- 이벤트 할 일 **알림**
- 🕒 2021년 7월 9일
- 🔄 반복 안함 ▼ ☒ 종일

저장

# 테스크 산정

개발을 시작하기 앞서 태스크를 산정해봅니다.

## 태스크 관리방식

- 글로 관리하는 것보다 도구를 쓰면 좋다.
- 스프레드 시트
- 노션(추천), 트렐로 등.. (kanban board 검색!)

## 캘린더 프로젝트

+ Add a view

Properties Group by Status Filter Sort Search ... New

No Status 0 ... + Not started 2 ... + In progress 2 ... + Completed 2 ... + Add a group

+ New

|          |           |        |
|----------|-----------|--------|
| 이메일 연동하기 | 배치 시스템 개발 | 기획서 분석 |
| 프로젝트 회고  | 할일 버그 픽스  | 도메인 설계 |

+ New + New + New

## 테스크 산정

(할일 범위 정의, 소요 시간 예측)

기획

설계

구현

- 기술 스택 정하기 (5m)
- 프로젝트 셋팅 (20m)
- 도메인 설계 (1h)
- API 스펙 설계 (스펙을 미리 전달해야할 상황인 경우) (20m)
- 도메인 구현 (3h)
  - 회원 기능
  - 스케줄 생성
  - 스케줄 조회
  - 이메일 연동
- 배치 프로젝트 셋팅 (1h)
- 알람 기능 개발 (1h)

세분화 할 수 있으면 더 좋다. 그만큼 일정이나 범위를 가시적으로 볼 수 있다는 의미!



# 1차 요구사항 개발

2 설계하기

# 기술 스택 정하기

## 회원가입, 로그인, 인증

### 스프링 MVC의 세션 방식으로 회원가입 사용

#### 스프링 MVC 세션 vs 스프링 시큐리티

- 요구사항에 비해 다루기 어렵다. 잘 사용할 수 있으면 좋지만, 학습곡선이 꽤 높은 편

#### 세션 vs JWT(json web token) 토큰 헤더 방식

- JWT는 **stateless** 하게 서버를 개발할 수 있는 장점이 있으나, JWT 강의를 통해 적용해볼 것. 세션방식이 아직까지도 많이 사용됨

#### 회원가입 vs OAuth 소셜 로그인

- 요구사항의 범위를 벗어난다고 판단

## 회원가입, 로그인 화면

### 스프링 MVC + *thymleaf* 템플릿 엔진으로 사용

템플릿 엔진 vs frontend framework (vue, react...)

- **react** 등을 사용해도 된다. (더 재미있다!)
- 하지만 초기 셋팅 비용이 많이 든다.

## 데이터 영속화

### 스프링 JPA 사용

관계형 데이터베이스 & ORM 프레임워크를 찾고 있다면,  
JPA는 선택이 아닌 필수가 되버린 요즘..

다른 대안도 존재

- Spring Data JDBC
- Mybatic..
- jdbc template..

보조 라이브러리

- QueryDSL (복잡한 쿼리를 코드로 관리하기 위해 사용됨. 필수는 아님. 사용하지 않음)

다음은 프로젝트 셋팅!

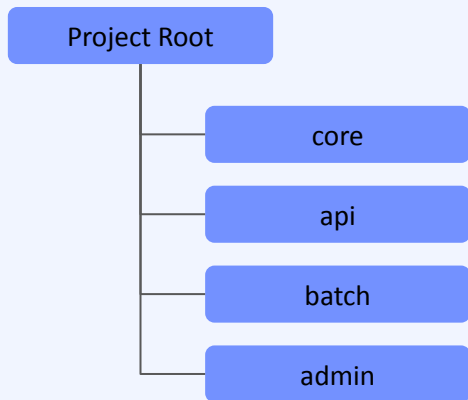
# 프로젝트 셋팅

## Gradle multi module

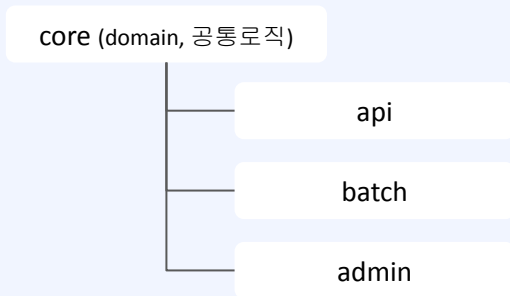
### 멀티 모듈을 만드는 이유

라이브러리를 가져다 쓰는 것처럼 우리가 만든 소스를 레고처럼 조립하며 써야 할 때 여러개 모듈을 만들면 좋다. 모듈(프로젝트)의 **상위, 하위 관계**를 맺어줄 수 있다.

프로젝트 디렉토리 구조



모듈 의존성 구조





## 코딩 시간

### 할 일

- 멀티 모듈 셋팅
- 도커 셋팅
- hello world 찍어보기

git branch: project-setting



# 도메인 설계 - 1

User, Event, Task, Notification 도메인 설계

## 도메인 설계 방식은 주로..

1. 도메인(객체) 관계를 파악(정의) : 그림 등으로..
- 2-1. 요구사항에 기반한 erd 를 설계를 먼저 하기도..
- 2-2. 도메인(객체) 관계를 먼저 그려보고, 코드로 짜보고 erd 설계..
3. 정규화, 비정규화 장단점 파악 후 보완

ORM을 사용한다면 두 가지를 **복합적으로 생각할 수 있어야 한다.**

- ERD 위주의 모델링을 하게 되면 ORM을 사용하는 장점을 활용하기 어려울 수도 있고,
- 객체 관계에만 집중하다보면 서비스가 커졌을 때 도메인 간 의존성 관리가 어려워 질 수도 있다. (객체관계가 거미줄처럼 엮여질수도..)

**2-2번으로 선택 (이유는 코드를 짜보면서 출발하는 개인적인 선호도)**

## 도메인 정의 - 유저

- 간단한 회원가입이 가능하다. (email, pw, name, birthday)
- 비밀번호는 **암호화** 하여 저장한다.

유저

- id
- name
- email
- password
- birthday

## 도메인 정의 - 이벤트

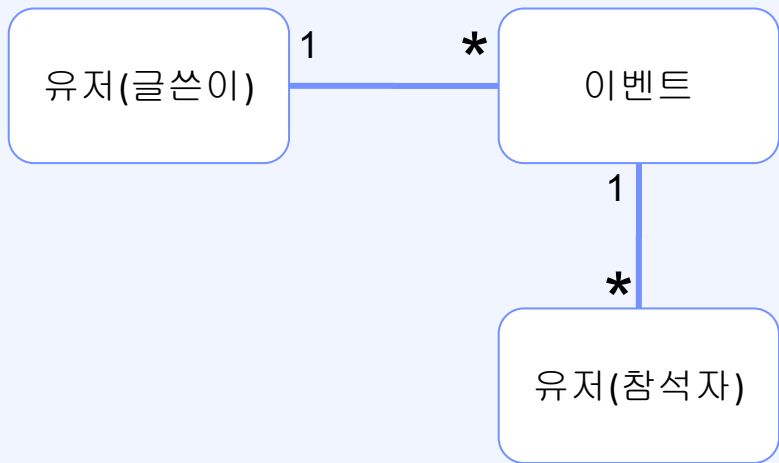
- 기간으로 등록
- 참석자 여러명
- 등록하면 참석자에게 **초대 메일**이 전송된다.
- 참석자는 메일을 통해 **수락, 거절**이 가능하다.
- 수락여부 파악 가능하다.



- id
- startAt
- endAt
- title
- description
- writer
- List<참석자>
- 수락 여부?
- 메일 전송 여부?

## 도메인 정의 - 이벤트

- 기간으로 등록
- 참석자 여러명
- 등록하면 참석자에게 **초대 메일**이 전송된다.
- 참석자는 메일을 통해 **수락, 거절**이 가능하다.
- 수락여부 파악 가능하다.



- id
- startAt
- endAt
- title
- description
- writer
- List<참석자>
- 수락 여부?
- 메일 전송 여부?

## 도메인 정의 - 이벤트

- 기간으로 등록
- 참석자 여러명
- 등록하면 참석자에게 **초대 메일**이 전송된다.
- 참석자는 메일을 통해 **수락, 거절**이 가능하다.
- 수락여부 파악 가능하다.



- id
- startAt
- endAt
- title
- description
- writer
- List<참석자>
- 수락 여부?
- 메일 전송 여부?

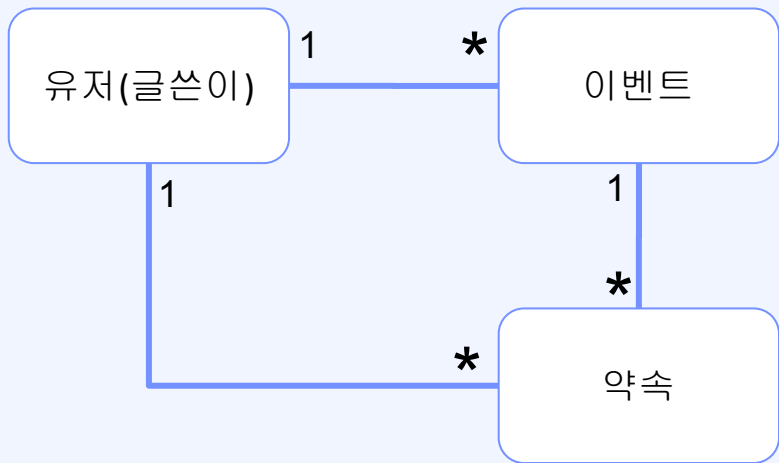
이 관계가 **어색한** 이유..

글쓴이와 참석자라는 관점에서는 위 관계가 맞으나,  
유저 엔티티(혹은 테이블) 입장에서는 이벤트와 **ManyToMany**(다대다) 관계가 된다.

**ManyToMany** 관계는 중간에 조인테이블을 추가하여, **일대다 + 다대일 관계**로 설계한다.

## 도메인 정의 - 약속

- 기간으로 등록
- 참석자 여러명
- 등록하면 참석자에게 **초대 메일**이 전송된다.
- 참석자는 메일을 통해 **수락, 거절**이 가능하다.
- 수락여부 파악 가능하다.



- id
- startAt
- endAt
- title
- description
- writer

이벤트의 초대원들마다 약속(초대장) 을 만들자.

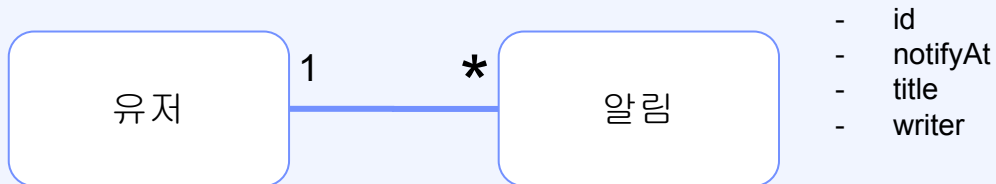
- id
- 이벤트(event\_id)
- 참석자(user\_id)
- 요청 상태(요청, 참석, 거부, 보류..?)



## 도메인 정의 - 할일



## 도메인 정의 - 알림



### 반복 기능 구현

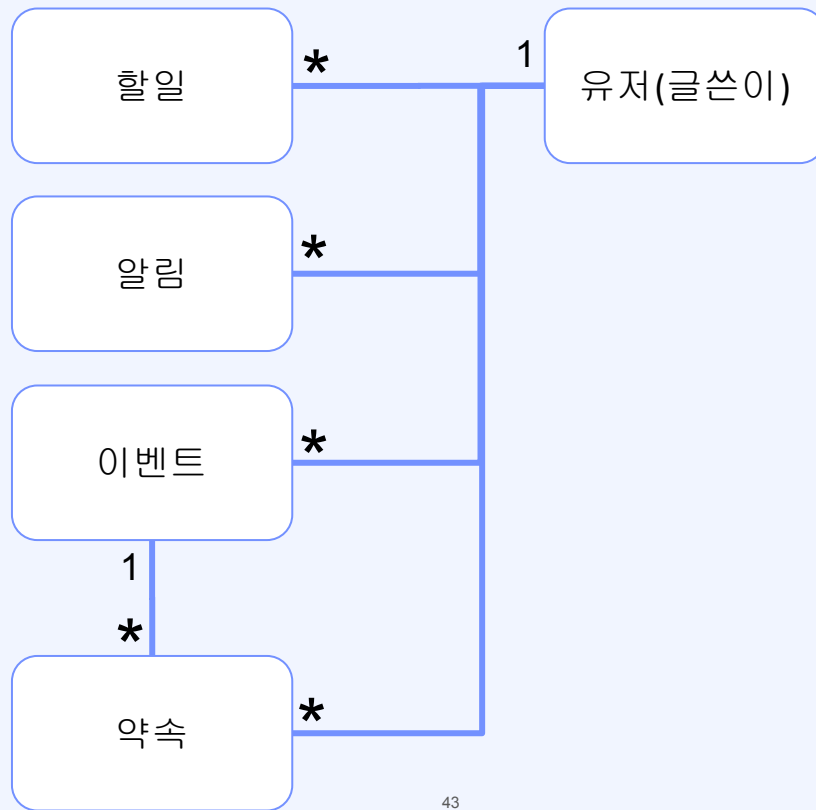
1. 반복 정보에 대한 것을 도메인에 녹여낼지?
2. 생성 시점에 반복 내용에 대해 여러개를 저장할 지?

1번 - 반복 정보는 구조 만들어서 저장하면 될건데.. 알림 로직이 많이 복잡해질 것 같다.

2번 - 알림 생성 요점에 계산해서 다 저장해두면 될듯. 하지만 무한 반복 알림설정은 대응 불가능.

**2번 채택!** (무한 알림은 스펙 아웃. 하고 싶으면 반복 정보까지 저장해놓고 기간을 두고 미리미리 찍어내기)

## 도메인 정의 - 최종 관계도



## 코딩 시간

### 할 일

- 도메인 객체 생성, 관계  
    맞어보기

git branch: domain-1



# 도메인 설계 - 2

Schedule 엔티티 도출해보기

## 엔티티 간의 데이터 중복

도메인 관계를 기반으로 erd를 확정하려다 보니

데이터가 중복된다..

분명 비슷한 데이터 구조인데 테이블을 여러개 관리해야 할지..? 에 대한 고민이 필요하다.

## 엔티티 간의 데이터 중복

1. 타입 속성을 추가하여 **events** 테이블을 확장해서 사용한다. (schedule)

장점: 테이블을 하나로 관리

단점: - **notifyAt**, **taskAt** 과 같은 도메인 변수명 사용 불가능

- 할일, 알림의 경우 불필요한 **null** 속성 다수 존재
- 로직 내에서 타입체크가 필수적이다.

```
Schedule schedule;
```

```
if (schedule.type() == Schedule.EVENT)
```

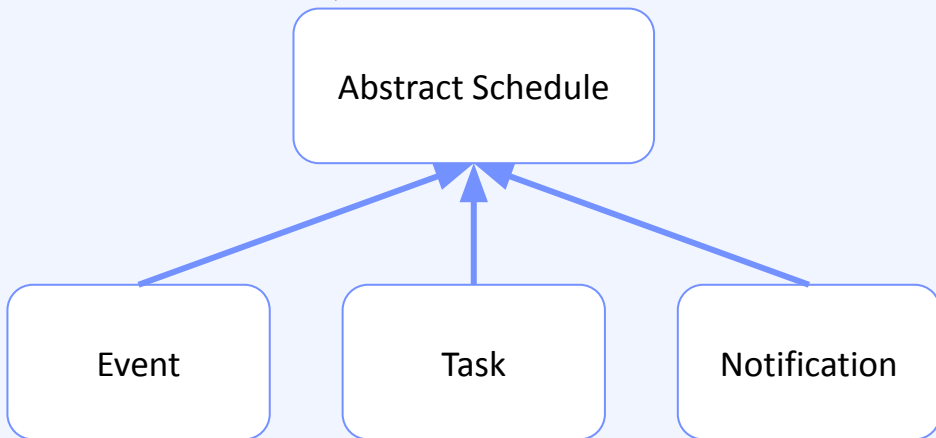
## 엔티티 간의 데이터 중복

2. **events** 테이블을 확장하되, 공통 속성을 부모 클래스로 분리하는 구조를 적용해본다.

(상속 관계 맵핑 전략!)

장점: **JPA**가 해당 기능을 제공해준다.

단점: - 편리하지만, **JPA** 의존도가 상당히 높아진다고 생각..





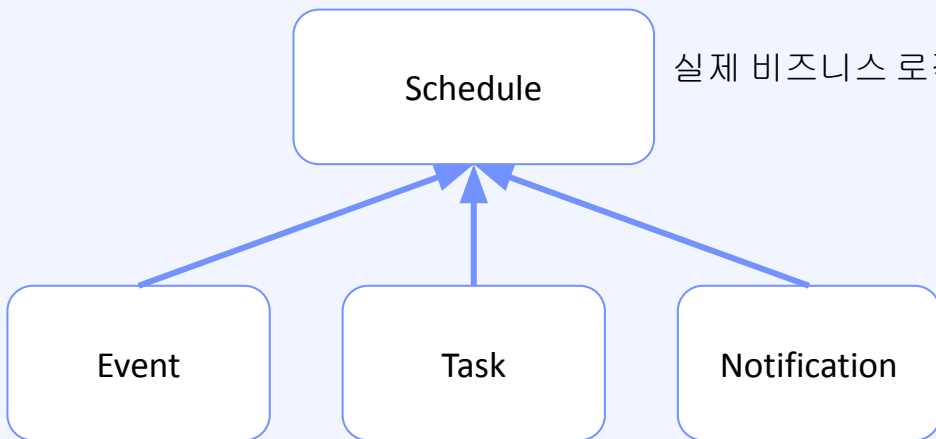
## 엔티티 간의 데이터 중복

순수하게 코드로 해결해볼 수도 있다.

테이블은 하나로 두되,

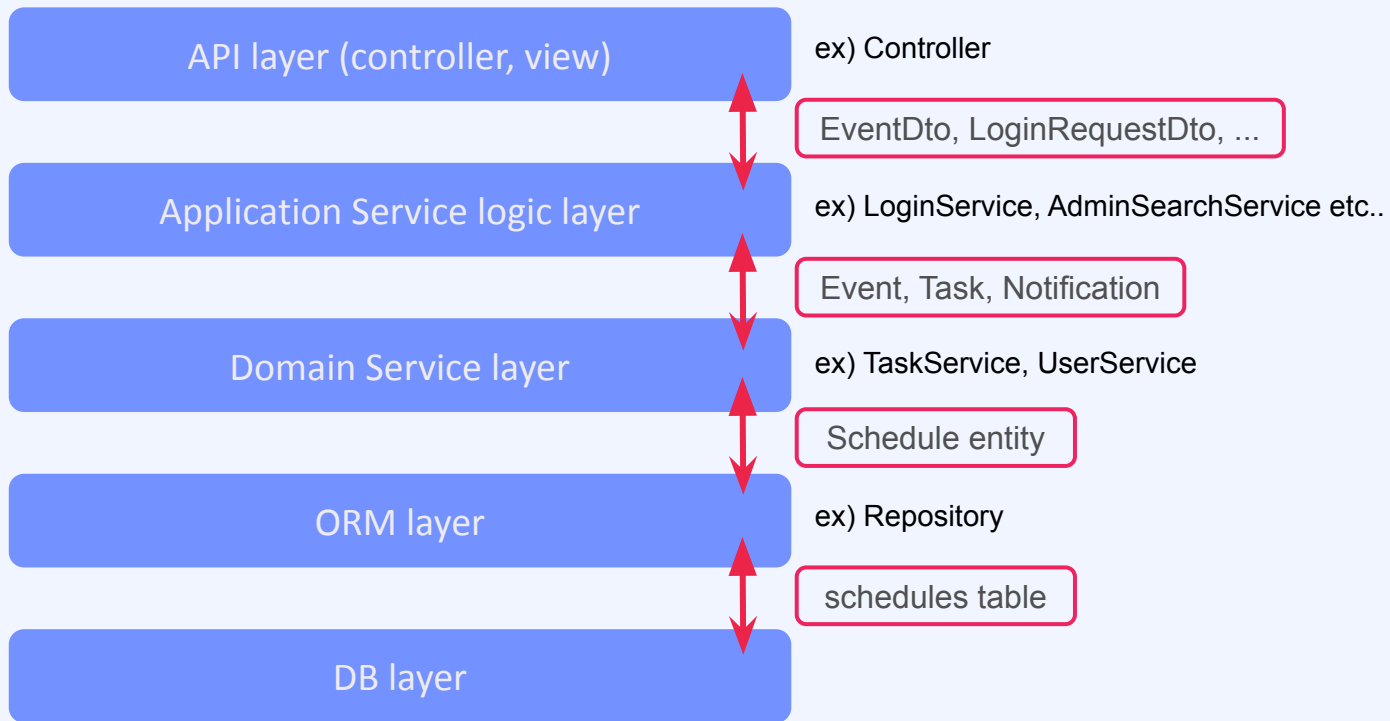
코드로 직접 감싸서 용도에 맞는 도메인 3개로 쪼갤 수 있지 않을까?

- 장점: 테이블을 하나로 둔다 (때로는 단점일수도.. 데이터가 한 테이블로 몰리므로)
- 단점: 구현의 복잡도가 생길 수도 있다. (프레임워크 의존도를 벗어나려다보니)



실제 비즈니스 로직에서는 스케줄에 대한 존재는 숨긴다.

## 지켜줘야 할 레이어 간 의존성 관리



## 코딩 시간

### 할 일

- 스케줄 엔티티로 통합
- 인텔리제이로 diagram 추출해보기

git branch : domain-2



# 도메인 설계 - 3

BaseEntity 적용 및 Auditing 이해하기

## 엔티티 간의 데이터 중복 - 2

User, Engagement, Schedule 간 공통 속성이 존재

자주 사용하는 기법인 JPA Auditing 을 사용하여 createdAt, updatedAt 컬럼을 추가해본다.

### JPA Auditing 이란?

- 엔티티의 각종 이벤트(엔티티 생명주기 그림..) 시점에 미리 등록해둔 Listener를 통해 특정 로직을 수행할 수 있는 기능이다.
- 예를들어 @PrePersist, @PreUpdate, @PreRemove
- 위 어노테이션을 이용하여 더 다양한 Auditing 기능을 추가할 수 있다.
- AuditingEntityListener.java 수스 참고

```
@PrePersist  
public void touchForCreate(Object target) {}
```

## 코딩 시간

### 할 일

- id, createdAt, updated 공통화
- Auditing 기능 코드 파악해보기



# API 스펙 설계 - 1

## API 스펙 관리에 대해

1. 코드로 관리
  - Swagger (자바 어노테이션 기반)
  - Spring Rest Docs (스프링 테스트 코드 기반)
2. Rest Client Tool 이용
  - Postman 등
3. 문서로 관리 (이가 아니면 잇몸으로..)
  - 스프레드 시트
  - 어떤 표 형식이든..

모든 경우 **장단점**이 존재한다.

- 복잡하거나, 간단하거나
- 빠르거나, 오래걸리거나
- 상황에 맞게 진행하되 지속적으로 개선해나가는게 중요!

**지금은 장표로 관리!**



## RESTful API 에 대해

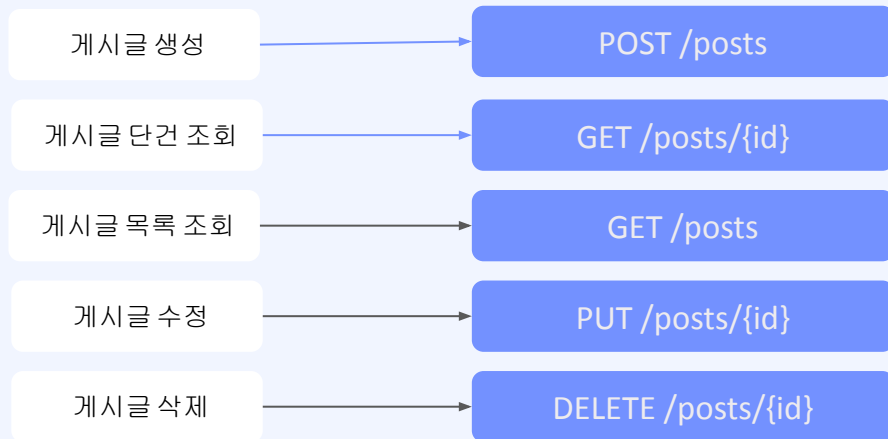
REST API 를 구성하는 요소가 정말 많다.

- 그것들을 모두 적용해야 할지는 실무 상황에 따라 많이 달라진다.
- (현재 강의의 범위를 벗어날 수도!)

몇가지 특징을 가지도록 설계하기도 한다.

- 한 리소스의 **CRUD** 에 대한 **http method, path** 를 정의하는 컨벤션을 따라 하겠다.

예) **post** (게시글 리소스 이름)



## 유저 인증, 권한관리 API 설계

### 회원가입

요청

POST /api/sign-up

```
{  
  "name": "name", // required, string  
  "email": "email", // required, string  
  "password": "password", // required, string, min_size=6  
  "birthday": "2021-07-01" // required, string, ISO-8601 format  
}
```

응답

200 OK Cookie: JSESSIONID=xxxx

## 유저 인증, 권한관리 API 설계

### 로그인

요청

POST /api/sign-in

```
{  
  "email": "email", // required, string  
  "password": "password", // required, string, min_size=6  
}
```

응답

200 OK Cookie: JSESSIONID=xxxx

### 로그아웃

요청

POST /api/sign-out

Cookie: JSESSIONID=xxxx

응답

200 OK

## 스케줄 3종 생성 API

### 이벤트 생성

#### 요청

POST /api/schedules/events

```
{  
  "title": "긴급회의", // required, string  
  "description": "아젠다 blabla" // not required, string  
  "startAt": "2021-08-15T23:00:00", // required, ISO-8601  
  "endAt": "2021-08-17T12:00:00", // required, ISO-8601  
  "attendeelds": [1, 3] // not required, int array  
}
```

#### 응답

200 OK

## 스케줄 3종 생성 API

### 할일 생성

#### 요청

POST /api/schedules/tasks

```
{  
  "title": "방 청소", // required, string  
  "description": "꼼꼼하게" // not required, string  
  "taskAt": "2021-08-17T12:00:00", // required, ISO-8601  
}
```

#### 응답

200 OK

## 스케줄 3종 생성 API

### 알림 생성

#### 요청

POST /api/schedules/notifications

```
{  
  "title": "아침 수영", // required, string  
  "notifyAt": "2021-08-17T08:00:00", // required, ISO-8601,  
  "repeatInfo": { // required  
    "interval": {  
      "intervalValue": 3,  
      "timeUnit": "DAY" // DAY/WEEK/MONTH/YEAR  
    },  
    "times": 4  
  }  
}
```

#### 응답

200 OK

반복예시) 3일에 한번씩 4번 반복

# API 스펙 설계 - 2

## Engagement 수정 API

### 이벤트 수락,거절 api (역등성 지원)

요청

```
PUT /api/schedules/events/engagements/{engagementId}
```

```
{
```

```
  "type": "ACCEPT" // ACCEPT/REJECT
```

```
}
```

응답

```
200 OK
```



## 스케줄 일,주,월별 조회 API

|        |   |   |                  |   |   |    |     |   |
|--------|---|---|------------------|---|---|----|-----|---|
| 오늘     | < | > | 2021년 7월 17일     | 🔍 | ? | ⚙️ | 일 ▾ | ⋮ |
|        |   |   | 토                |   |   |    |     |   |
|        |   |   | 17               |   |   |    |     |   |
| GMT+09 |   |   | 제한절              |   |   |    |     |   |
|        |   |   | Constitution Day |   |   |    |     |   |
| 오전 10시 |   |   |                  |   |   |    |     |   |
| 오전 11시 |   |   |                  |   |   |    |     |   |
| 오후 12시 |   |   |                  |   |   |    |     |   |
| 오후 1시  |   |   |                  |   |   |    |     |   |
| 오후 2시  |   |   |                  |   |   |    |     |   |
| 오후 3시  |   |   |                  |   |   |    |     |   |
| 오후 4시  |   |   |                  |   |   |    |     |   |

|        |    |    |          |    |    |    |           |   |
|--------|----|----|----------|----|----|----|-----------|---|
| 오늘     | <  | >  | 2021년 7월 | 🔍  | ?  | ⚙️ | 주 ▾       | ⋮ |
|        | 일  | 월  | 화        | 수  | 목  | 금  | 토         |   |
|        | 11 | 12 | 13       | 14 | 15 | 16 | 17        |   |
| GMT+09 |    |    |          |    |    |    | 제한절       |   |
|        |    |    |          |    |    |    | Constitut |   |
| 오전 1시  |    |    |          |    |    |    |           |   |
| 오전 2시  |    |    |          |    |    |    |           |   |

## 스케줄 일,주,월별 조회 API

|         |         |                                     |          |            |        |                               |     |   |
|---------|---------|-------------------------------------|----------|------------|--------|-------------------------------|-----|---|
| 오늘      | <       | >                                   | 2021년 7월 | 🔍          | ?      | ⚙️                            | 월 ▾ | ⋮ |
| 일<br>27 | 월<br>28 | 화<br>29<br>○ 오후 10:30 [ 연사미팅] 정재균 ▾ | 수<br>30  | 목<br>7월 1일 | 금<br>2 | 토<br>3                        |     |   |
| 4       | 5       | 6                                   | 7        | 8          | 9      | 10                            |     |   |
| 11      | 12      | 13                                  | 14       | 15         | 16     | 17<br>제한일<br>Constitution Day |     |   |
| 18      | 19      | 20                                  | 21       | 22         | 23     | 24                            |     |   |
| 25      | 26      | 27                                  | 28       | 29         | 30     | 31                            |     |   |

응답 데이터 형식은 일,주,월 다르게 할 필요까진 없어 보인다.

## 스케줄 일,주,월별 조회 API

### 일별 조회

#### 요청

```
GET /api/schedules/day?date=2021-07-03
{
  "type": "ACCEPT" // ACCEPT/REJECT
}
```

#### 응답

200 OK

```
{
  "userId": 1, "name": "홍길동",
  "scheduleResList": [
    {
      "scheduleId": 1,
      "startAt": "2021-08-15T23:00:00",
      "endAt": "2021-08-17T12:00:00",
      "title": "제주도 가기",
      "description": null,
      "scheduleType": "EVENT"
    },
    {
      "scheduleId": 2,
      "taskAt": "2021-08-15T23:00:00",
      "title": "방청소",
      "description": null,
      "scheduleType": "TASK"
    },
    {
      "scheduleId": 3,
      "notifyAt": "2021-08-15T23:00:00",
      "title": "방청소",
      "description": null,
      "scheduleType": "NOTIFICATION"
    }
  ]
}
```

문서 관리의 단점.. (코드 스니펫도 없을 뿐더러 모든게  
수동이다..)

## 스케줄 일,주,월별 조회 API

### 주별 조회

요청

GET /api/schedules/week?startOfWeek=2021-07-03

### 월별 조회

요청

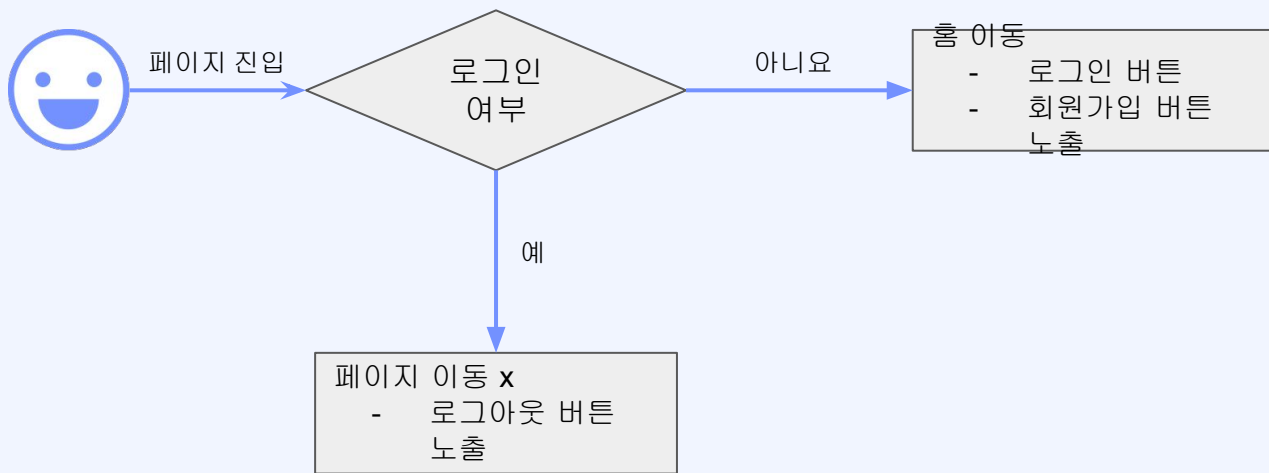
GET /api/schedules/month?yearMonth=2021-07 // yyyy-MM

### 응답

일 조회와 동일

## 필요한 화면 정의 (로그인, 회원가입, 이벤트 수락,거절 화면)

간단하지만 유저 플로우를 만들어보고자 하니, 유저단에 대해서는 화면이 있으면 좋겠다.



캘린더 화면을 구현하지 않으니 사실상 필요한 화면은 ‘홈’ 하나뿐이다.  
홈에서 버튼을 제어할 수 있게 구현한다. (화면 소스는 제공)

## 코딩 시간

이제 상세 구현을 들어가봅시다!



# 1차 요구사항 개발

3 상세 기능 구현

# 가입, 로그인, 로그아웃 구현



## 코딩 시간

### 할 일

- LoginService, UserService 분리
- HttpSession 사용

git branch: login-service



# 패스워드 암호화 적용

## 암호화 간단 개념

- 평문을 알아볼 수 없게 바꿔 놓음
- 양방향, 단방향의 개념이 있다.
  - 양방향 : 복호화 가능 (대칭키, 공개키 방식)
  - 단방향 : 복호화 불가능 (해시함수)

### 패스워드 암호화 요구조건

- 복호화 불가능
  - 해시함수 사용
- 특정 패스워드의 해시값이 노출되어도, 같은 패스워드인 다른 계정도 탈취당하면 안된다.
  - salt 필요
- Brute-force 공격에 대비가 가능해야 한다. (연산속도가 너무 빠르면 안된다.)

*BCrypt 알고리즘을 많이 사용한다.*

## 코딩 시간

### 할 일

- jbcrypt 의존성 추가
- 암호화 테스트
- UserService 에 적용

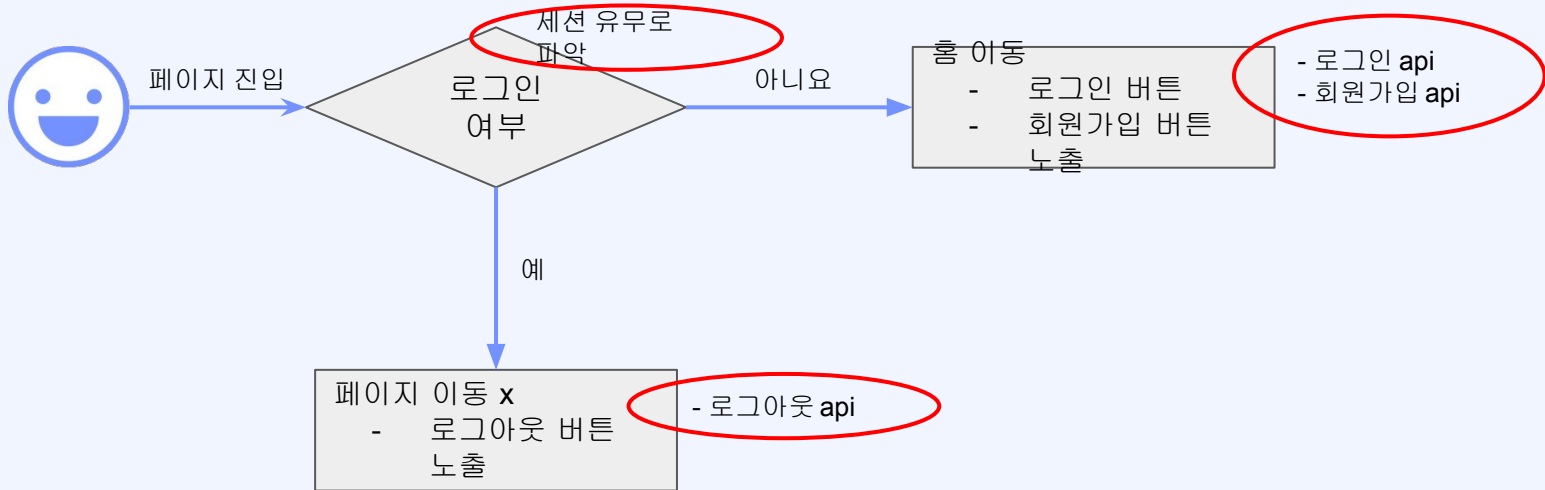
git branch: password-encoding



# 화면 구현

## 필요한 화면 정의 (로그인, 회원가입, 이벤트 수락,거절 화면)

간단하지만 유저 플로우를 만들어보고자 하니, 유저단에 대해서는 화면이 있으면 좋겠다.



캘린더 화면을 구현하지 않으니 사실상 필요한 화면은 ‘홈’ 하나뿐이다.  
홈에서 버튼을 제어할 수 있게 구현한다. (화면 소스는 제공)

## 코딩 시간

### 할 일

- 로그인 관련 api 개발
- 화면 개발
- 브라우저 테스트

git branch: login-view



# 핵심 도메인 개발 - Task



## 코딩 시간

### 할 일

- 생성 로직 개발
- **ArgumentResolver** 사용
- API 테스트

git branch: create-task, auth-resolver



# 핵심 도메인 개발 - Event

## 코딩 시간

### 할 일

- Event, Engagement 생성 로직 개발
- API 테스트

github branch: create-event



# 핵심 도메인 개발 - Notification

## 코딩 시간

### 할 일

- 반복 알림 요구사항 구현
- API 테스트

github branch: create-notification



# 조회 로직 개발 - 일별 조회

## 스케줄 일,주,월별 조회 API

### 일별 조회

#### 요청

```
GET /api/schedules/day?date=2021-07-03
{
  "type": "ACCEPT" // ACCEPT/REJECT
}
```

#### 응답

200 OK

```
{
  "userId": 1, "name": "홍길동",
  "scheduleResList": [
    {
      "scheduleId": 1,
      "startAt": "2021-08-15T23:00:00",
      "endAt": "2021-08-17T12:00:00",
      "title": "제주도 가기",
      "description": null,
      "scheduleType": "EVENT"
    },
    {
      "scheduleId": 2,
      "taskAt": "2021-08-15T23:00:00",
      "title": "방청소",
      "description": null,
      "scheduleType": "TASK"
    },
    {
      "scheduleId": 3,
      "notifyAt": "2021-08-15T23:00:00",
      "title": "방청소",
      "description": null,
      "scheduleType": "NOTIFICATION"
    }
  ]
}
```

문서 관리의 단점.. (코드 스니펫도 없을 뿐더러 모든게  
수동이다..)

## 코딩 시간

### 할 일

- 일별 조회 로직 추가
- Period 객체 활용

github branch: query-day





# 조회 로직 개발 - 일,주,월 별 조회

## 코딩 시간

### 할 일

- 일,주,월별 조회 로직 완성 및 코드 개선

github branch: query-remains, query-function



# API 개발 마무리 - Validation 사용

## 코딩 시간

### 할 일

- Validation 어노테이션 추가 및 테스트

git branch: api-validation



# API 개발 마무리 - 예외처리

## 코딩 시간

### 할 일

- CustomException 생성
- Global Handler 를 이용한 일관된 예외 처리 방식 적용

git branch: api-exceptions



# 이메일 발송 기능 개발

gmail smtp 서버를 이용하기 위한 등록 절차

## 메일 발송에 필요한 준비물

- Gmail SMTP 접속용 계정 생성
- 테스트를 위한 실제 이메일 준비.. (1~3개)

### 대략적인 개발 순서

- gmail smtp 계정 생성 및 메일 전송을 위한 앱 등록
- spring mail starter 의존성 추가
- 스프링 설정에 smtp 추가
- 이메일 발송 기능 개발

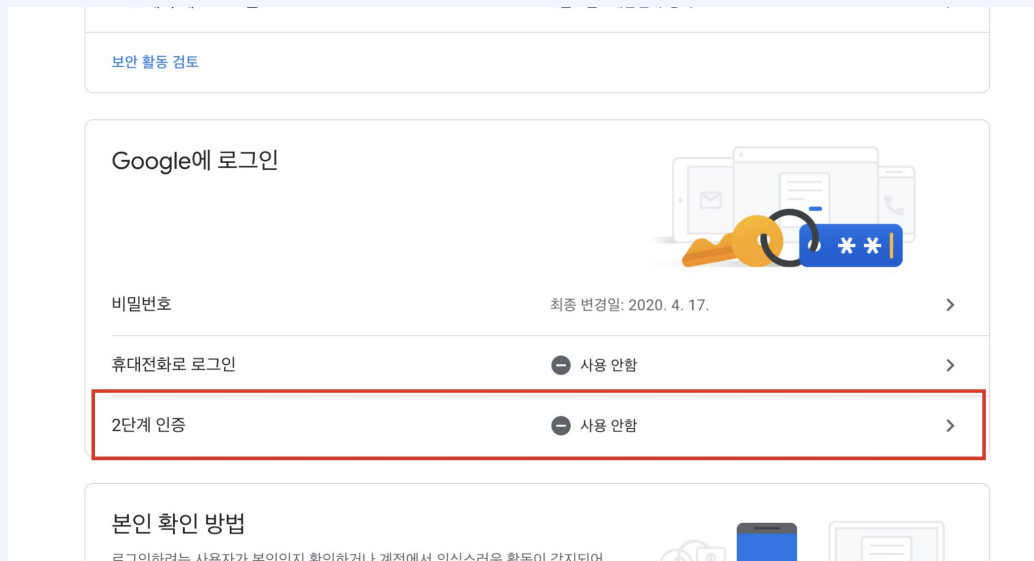
blog: <https://thisdev.tistory.com/6>



## Gmail SMTP 용 설정하기

구글 계정의 보안 설정 접속 후 2단계 인증 진행

<https://myaccount.google.com/u/2/security>



## Gmail SMTP 용 설정하기

구글 계정의 보안 설정 접속 후 2단계 인증 진행

← 2단계 인증



### 2단계 인증으로 계정 보호

Google 계정에 로그인할 때마다 비밀번호 및 인증 코드가 필요합니다. 자세히 [알아보기](#)



보안을 강화하세요.

비밀번호와 휴대전화로 전송된 고유 인증 코드를 입력하세요.



계정 도용 방지

누군가 내 비밀번호를 알게 되더라도 내 계정에 로그인할 수 없습니다.

시작하기

← 2단계 인증

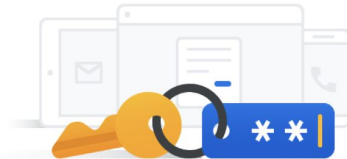
2021. 8. 5.에 2단계 인증을 사용하도록 설정했습니다.

사용 안함

## Gmail SMTP 용 설정하기

### 앱 비밀번호 클릭

Google에 로그인



|        |  |   |
|--------|--|---|
| 비밀번호   | 최종 변경일: 2020. 4. 17.                   | > |
| 2단계 인증 | <input checked="" type="checkbox"/> 사용 | > |
| 앱 비밀번호 | 없음                                     | > |

## Gmail SMTP 용 설정하기

### 앱 비밀번호 설정

앱 설정 -> 메일

기타 설정 -> 기타(맞춤 이름)

#### < 앱 비밀번호

앱 비밀번호를 사용하면 2단계 인증을 지원하지 않는 기기의 앱에서 Google 계정에 로그인할 수 있습니다. 비밀번호를 한 번만 입력하면 기억할 필요가 없습니다. [자세히 알아보기](#)

앱 비밀번호가 없습니다.

앱 비밀번호를 생성할 앱 및 기기를 선택하세요.

메일

기기 선택

iPhone

iPad

BlackBerry

Mac

Windows 휴대전화

Windows 컴퓨터

기타(맞춤 이름)

생성

## Gmail SMTP 용 설정하기

### 앱 이름 추가 후 생성

#### ← 앱 비밀번호

앱 비밀번호를 사용하면 2단계 인증을 지원하지 않는 기기의 앱에서 Google 계정에 로그인할 수 있습니다. 비밀번호를 한 번만 입력하면 기억할 필요가 없습니다. [자세히 알아보기](#)

앱 비밀번호가 없습니다.

앱 비밀번호를 생성할 앱 및 기기를 선택하세요.

test-calendar-app X

생성

## Gmail SMTP 용 설정하기

앱 비밀번호 확인

-> 스프링 설정에 추가

앱 비밀번호

생성된 앱 비밀번호

기기용 앱 비밀번호

Email

securesally@gmail.com

Password

••••••••••

사용 방법

설정하려는 애플리케이션 또는 기기의 Google 계정 설정으로 이동합니다. 비밀번호를 위에 표시된 16자리 비밀번호로 교체합니다.

일반적인 비밀번호와 마찬가지로 이 앱 비밀번호는 Google 계정에 대한 완전한 액세스 권한을 부여합니다. 비밀번호를 기억하지 않아도 되므로 적어 놓거나 다른 사용자와 공유하지 마세요.

[확인](#)

# 이메일 발송 기능 개발

## Java mail 사용한 이메일 발송 테스트

## 코딩 시간

### 할 일

- 스프링 설정 추가
- 메일 발송 테스트

git branch : mail-test





# 이메일 발송 기능 개발

## thymeleaf 를 사용한 템플릿 메일 발송 - 1

## 코딩 시간

### 할 일

- 이메일 템플릿 추가
- Engagement update api 추가
- 이메일 서비스 구현
- 테스트

git branch: mail-template-1



# 이메일 발송 기능 개발

## thymeleaf 를 사용한 템플릿 메일 발송 - 2

## 코딩 시간

### 할 일

- 메일 날짜 포맷 로직 개발

git branch: mail-template-2



# 1차 요구사항 개발

4 마무리

# 개선사항 도출, 도전과제 안내

## 개선사항 도출 & 도전과제 안내

테스트 코드 부족

-> 가능한 테스트는 많이 만들기

도메인 서비스에 외부 서비스 디펜던시 (이메일 - 이벤트)

-> **Spring Event Publisher** 기능을 고려해 볼 것

이메일 발송 기능 성능 개선

-> **Spring Async** 로 비동기 적용

# 알람 배치 시스템 개발

## 1 기획서 이해



# 알람 요구사항 분석 및 테스트 산정

## 알람 요구사항

1분 간격으로 **스케줄이 있는 대상**에게 이메일 발송 (배치 프로그램)

스케줄 대상

- **schedules** 테이블에 있는 **writer** 기준
- **engagements** 테이블에 있는 **attendee** 기준

## 테스크 산정

- 기술 스택 정하기 (Spring Batch)
- 배치 프로젝트 셋팅
- 알람 대상 조회 로직 구현
- 알람 대상 이메일 발송 로직 구현
  - **Batch** 모듈에서 이메일을 발송할 지?
  - **Batch -> Api** 모듈로 **api** 를 통해 발송할 지? (의존성 관리를 이유로.. 선택)
- 주기적으로 실행하도록 **cron** 설정 및 실행

# 알람 배치 시스템 개발

## 2 설계하기

# 배치 프로젝트 셋팅

## 코딩 시간

### 할 일

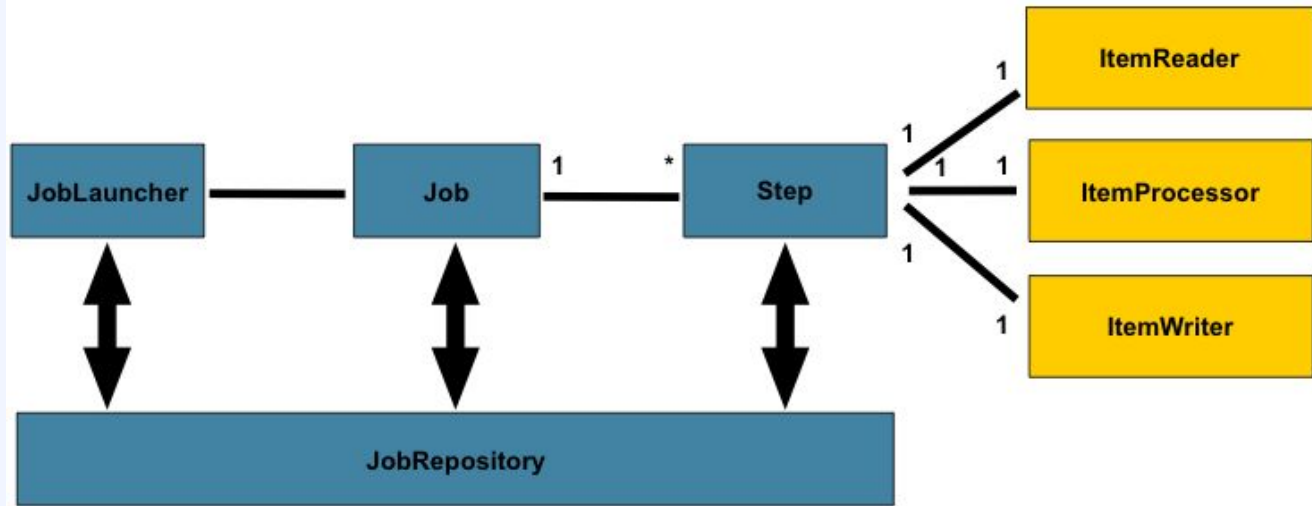
- 배치 프로젝트 셋팅
- 데이터 소스 설정, 배치 테이블 생성
- 간단한 잡 실행

git branch : batch-setting



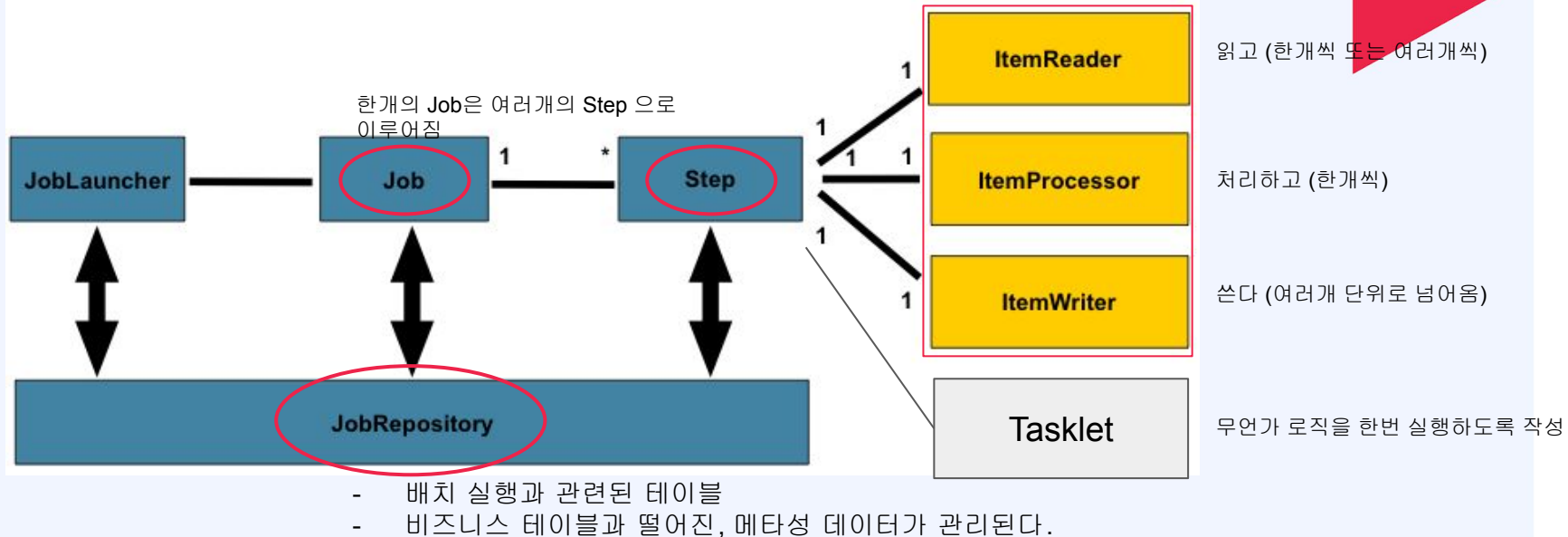
# 배치 아키텍처 설명

## 스프링 배치 구조





## 스프링 배치 구조



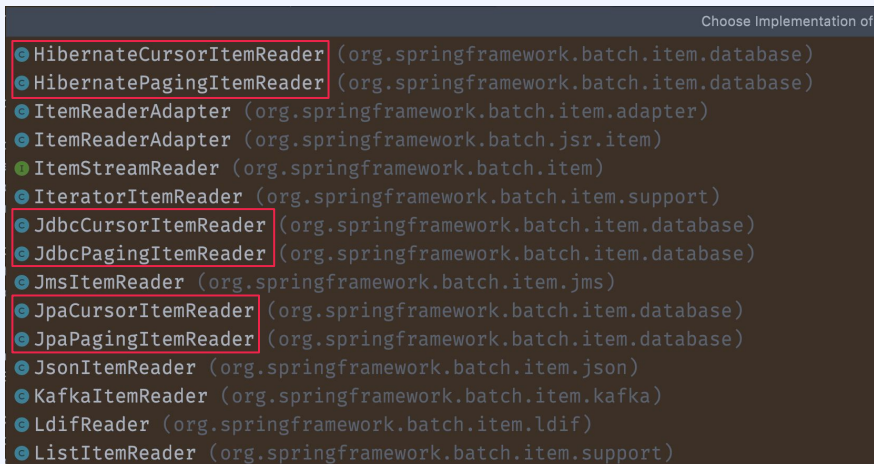
# ItemReader

```
public interface ItemReader<T> {  
  
    /** Reads a piece of input data and advance to the next one. Implementations ... */  
    @Nullable  
    T read() throws Exception, UnexpectedInputException, ParseException, NonTransientResourceException;  
  
}
```

단순하게 아이템 하나를 읽는 **Strategy** 이다.

배치 어플리케이션은 데이터를 읽는 것으로 시작하기 때문에,  
다양한 데이터 소스 (**꼭 db를 말하는 것은 아님!!**)로부터  
데이터를 읽을 수 있는 구현체를 잘 정의하는게 중요하다.

저희는 어떤 **Reader**를 선택해서 만들어야 할까요?



# ItemProcessor

```
public interface ItemProcessor<I, O> {

    /** Process the provided item, returning a potentially
     * @Nullable
     * @param item the item to process
     * @return the processed item
     */
    O process(@NonNull I item) throws Exception;
}
```

Reader로부터 받아온 Item을 가공하는 담당.

Item이 <I> 타입이었다면 <O> 타입으로 변경해서 넘길 수 있음.

배치 어플리케이션의 핵심 비즈니스 로직이 들어가게 됨.

하지만 따로 가공할 로직이 없다면 **Processor** 는 만들지 않아도 된다.  
(Optional)

프로세서는 개발자가 직접 구현하는 영역이기 때문에 제공되는 구현체의 종류도 Reader에 비해 별로 없음.

```

● BeanValidatingItemProcessor (org.springframework.batch.item
● ClassifierCompositeItemProcessor (org.springframework.batch.item
● CompositeItemProcessor (org.springframework.batch.item
● FunctionItemProcessor (org.springframework.batch.item
● ItemProcessorAdapter (org.springframework.batch.item
● ItemProcessorAdapter (org.springframework.batch.jsr.item
● PassThroughItemProcessor (org.springframework.batch.item
● ScriptItemProcessor (org.springframework.batch.item
● ValidatingItemProcessor (org.springframework.batch.item

```

# ItemWriter

```
public interface ItemWriter<T> {  
  
    /** Process the supplied data element. Will not be called  
    void write(List<? extends T> items) throws Exception;  
  
}
```

Reader, Processor로부터 받아온 Item에 대한 마지막 처리 단계 (마지막이니 void)

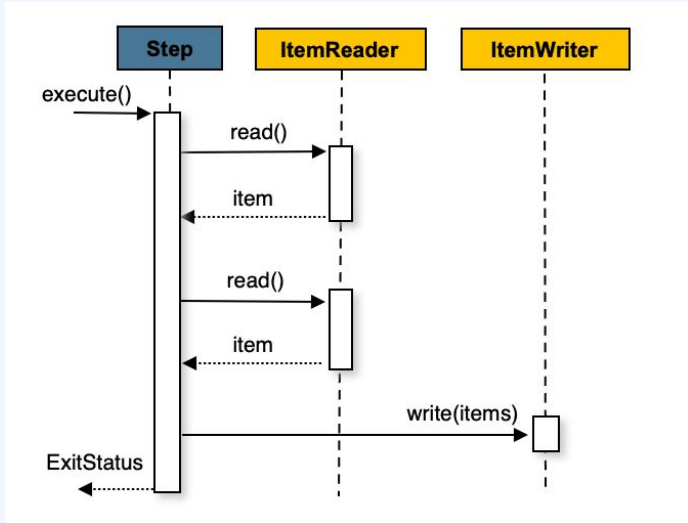
- db에 저장하거나, 파일로 쓰거나, 이벤트를 발행하거나 등등..

Step을 만들 때 지정하는 **Chunk 갯수만큼의 인자**로 받게된다.

예시)

- Paging Reader의 page\_size = 10, Chunk = 100 이라면
- 
- Reader가 10번 Page Read를 하고,
- Processor 100번 처리할 때마다
- Writer를 실행한다.

## Chunk Orientation, Batch Transaction 에 대해



스프링배치는 **Chunk** 지향 프로세싱

- 읽어야 할 또는, 처리할 아이템이 굉장히 많을때 트랜잭션을 한번에 이어가는 것 보다 일정 주기로 여러번 실행하는 것이 **안정적**이다.

```

List items = new ArrayList();
for(int i = 0; i < commitInterval; i++){
    Object item = itemReader.read();
    if (item != null) {
        items.add(item);
    }
}
itemWriter.write(items);
    
```

안정적?

- **Fault Tolerant** (장애 허용)
- 실패 시 다양한 처리를 할 수 있게 해준다.

<https://docs.spring.io/spring-batch/docs/current/reference/html/step.html#chunkOrientedProcessing>

## Chunk Orientation, Batch Transaction 에 대해

TaskletStep.doExecution(StepExecution stepExecution)

```
stepOperations.iterate(new StepContextRepeatCallback(stepExecution) {  
  
    @Override  
    public RepeatStatus doInChunkContext(RepeatContext repeatContext, ChunkContext chunkContext)  
        throws Exception {  
  
        StepExecution stepExecution = chunkContext.getStepContext().getStepExecution();  
  
        // Before starting a new transaction, check for  
        // interruption.  
        interruptionPolicy.checkInterrupted(stepExecution);  
  
        RepeatStatus result;  
        try {  
            result = new TransactionTemplate(transactionManager, transactionAttribute)  
                .execute(new ChunkTransactionCallback(chunkContext, semaphore));  
        }  
    }  
}
```

# Reader 구경해보기

## 코딩 시간

### 할 일

- PagingReader 와 CursorReader 차이점 파악
- 간단히 모든 row를 읽어들이는 배치 잡 살펴보기

git branch : batch-reader





# 알람 배치 시스템 개발

## 3 상세 기능 구현

# 알람 대상 조회 로직 개발

## 코딩 시간

### 할 일

- Engagement, Schedule
- 알람 대상 Read Job 개발

git branch : batch-implement



# 배치 이메일 발송 기능 개발

## 코딩 시간

### 할 일

- Api module에 이메일 발송 api 추가
- cron 으로 배치 테스트

git branch : batch-mail

Windows OS 크론식 참고

<https://decdream08.tistory.com/67>



# 알람 배치 시스템 개발

4 마무리

# 개선사항 도출, 도전과제 안내

## 개선사항 도출 & 도전과제 안내

### 테스트 코드 부족

-> 가능한 테스트는 많이 만들기! 배치도 테스트 가능하며 배치 테스트를 잘 짜는 것도 중요하다.

### 배치 실패시 처리

-> 해피케이스만 고려했지만 배치도 실패 사유가 다양하며, 실패 시 후처리를 고려해야 한다.

-> 재시도 할 것인지? (전체, 부분)

-> 재시도 하지 않고 다른 제 2의 처리? (JobListener, StepListener 고려)

-> 개발팀에 noti를 즉각 줄 수 있을지?

### 기타

-> 쿼리 베이스가 아닌 queryDSL 등을 사용하여 쿼리를 코드로 관리할 수도?

-> 배치를 중단하고 싶거나, 주기를 변경하고 싶을 때 크론식을 변경해야 한다.

-> 젠킨스 등을 이용하여 잡 트리거링 가능(현업에서 많이 사용)

-> 알람 대상 범위를 변경하고 싶을때 (10분전이 아니라 10~20분 전 사이의 대상으로 한다던지..)

-> 배치 실행 할때 JobParameter 를 넘겨주기



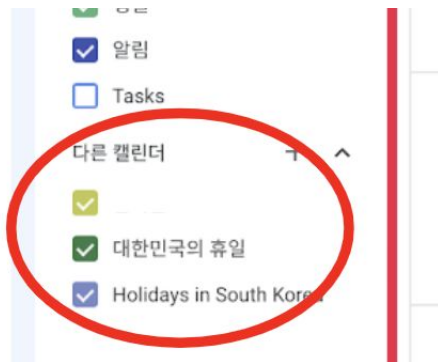
# 캘린더 공유하기 추가 기능 개발

## 1 기획서 이해

# 공유하기 기능 분석 및 테스트 산정

## 공유하기 기능

유저는 다른 유저에게 스케줄 공유하기 요청을 할 수 있고  
공유하기 요청에 수락하면 서로의 스케줄을 확인할 수  
공유에는 방향이 존재한다. (단방향, 양방향)



예) **A**유저가 **B**유저에게 단방향 공유 요청 (**B**야 내 스케줄까지 확인해~)  
결과)

- **A** 유저는 자신의 스케줄만 확인 가능
- **B** 유저는 **A** 스케줄까지 확인 가능

### 필요한 API ?

- 공유하기 요청(생성) **api** (생성 시점에 이메일 발송)
- 공유하기 수락,거절 **api** (like engagement)
- 공유하기 조회 **api**

## 테스크 산정

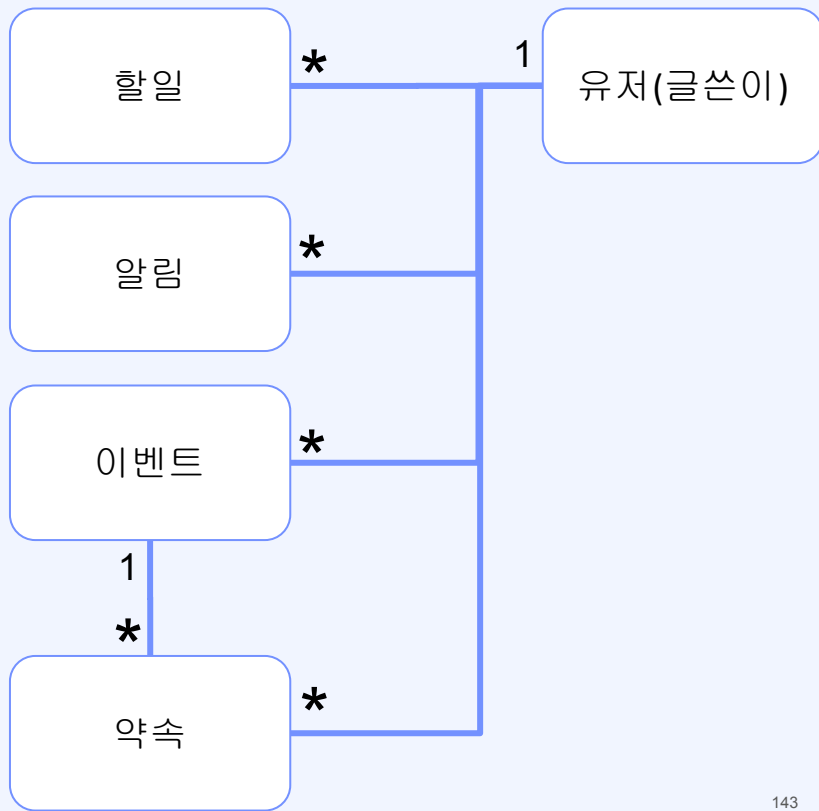
- 테이블 설계
- **API** 설계
- 생성, 수정 **api** 개발
- 조회 **api** 개발
- 테스트

# 캘린더 공유하기 추가 기능 개발

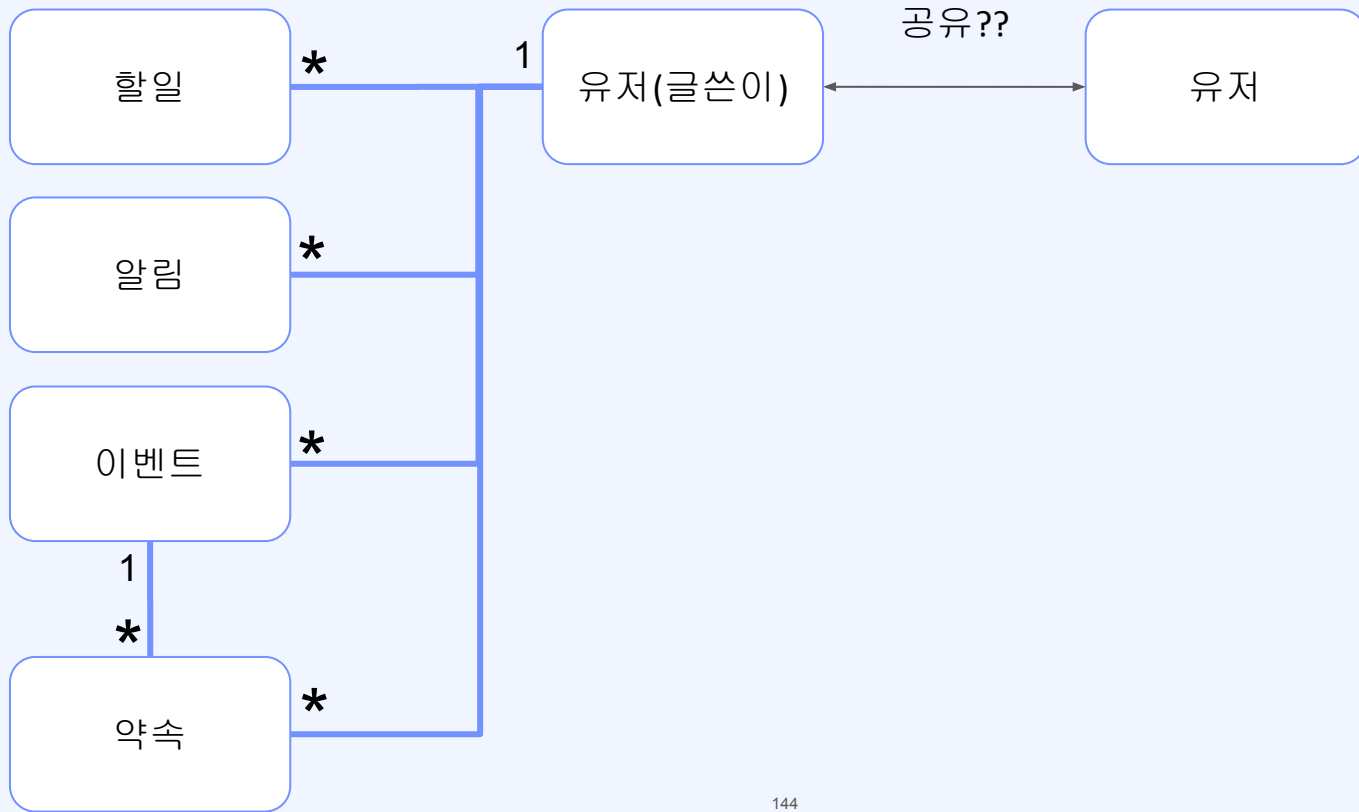
## 2 설계하기

# 공유하기 테이블, API 스펙 설계

## 도메인 정의 - 최종 관계도

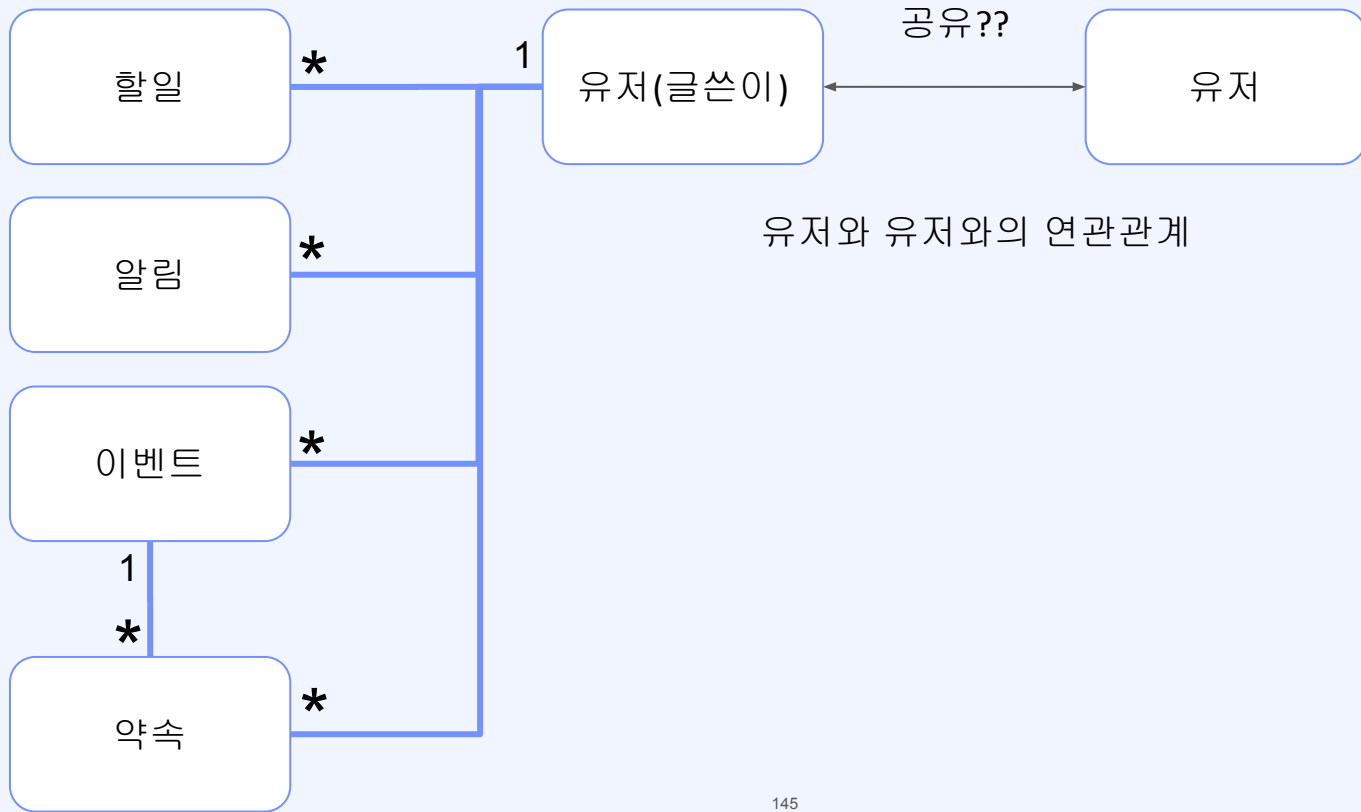


## 도메인 정의 - 최종 관계도

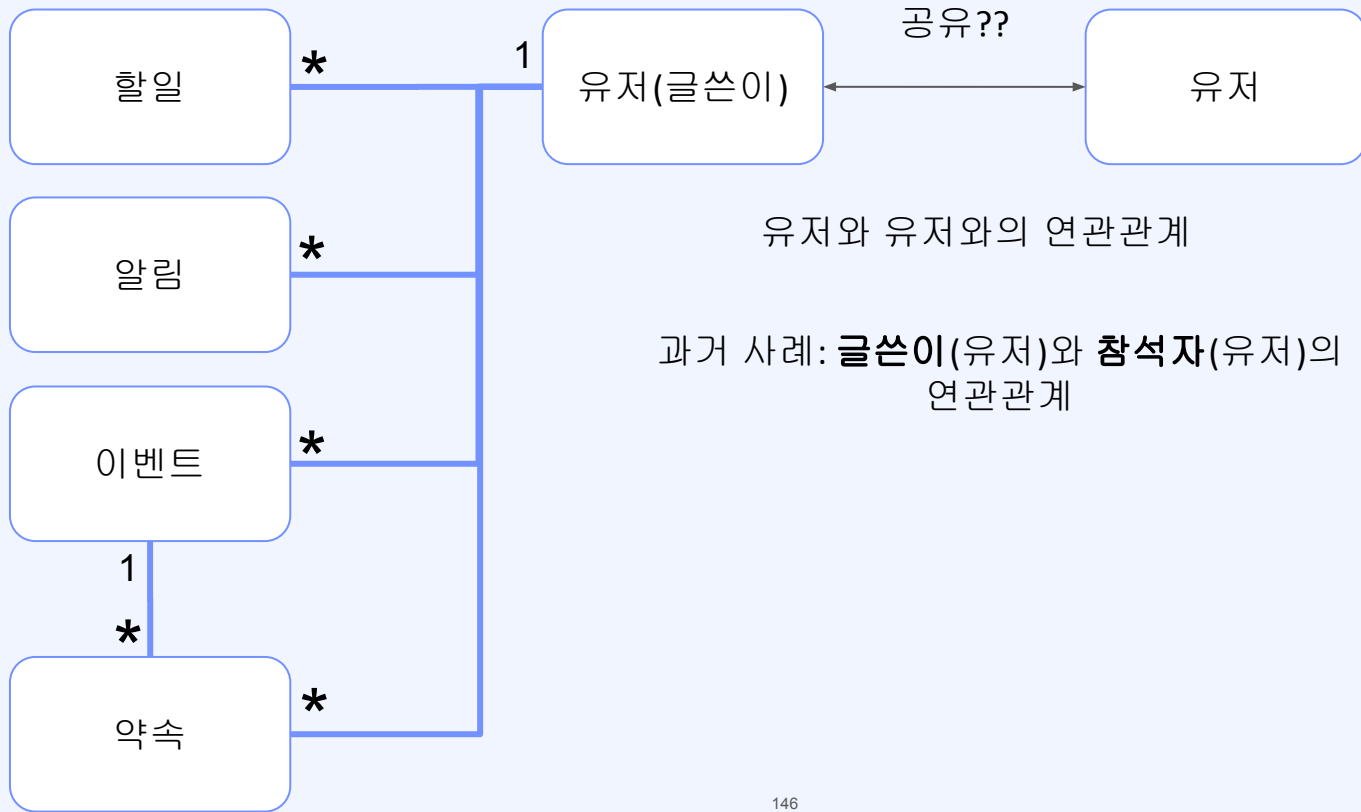




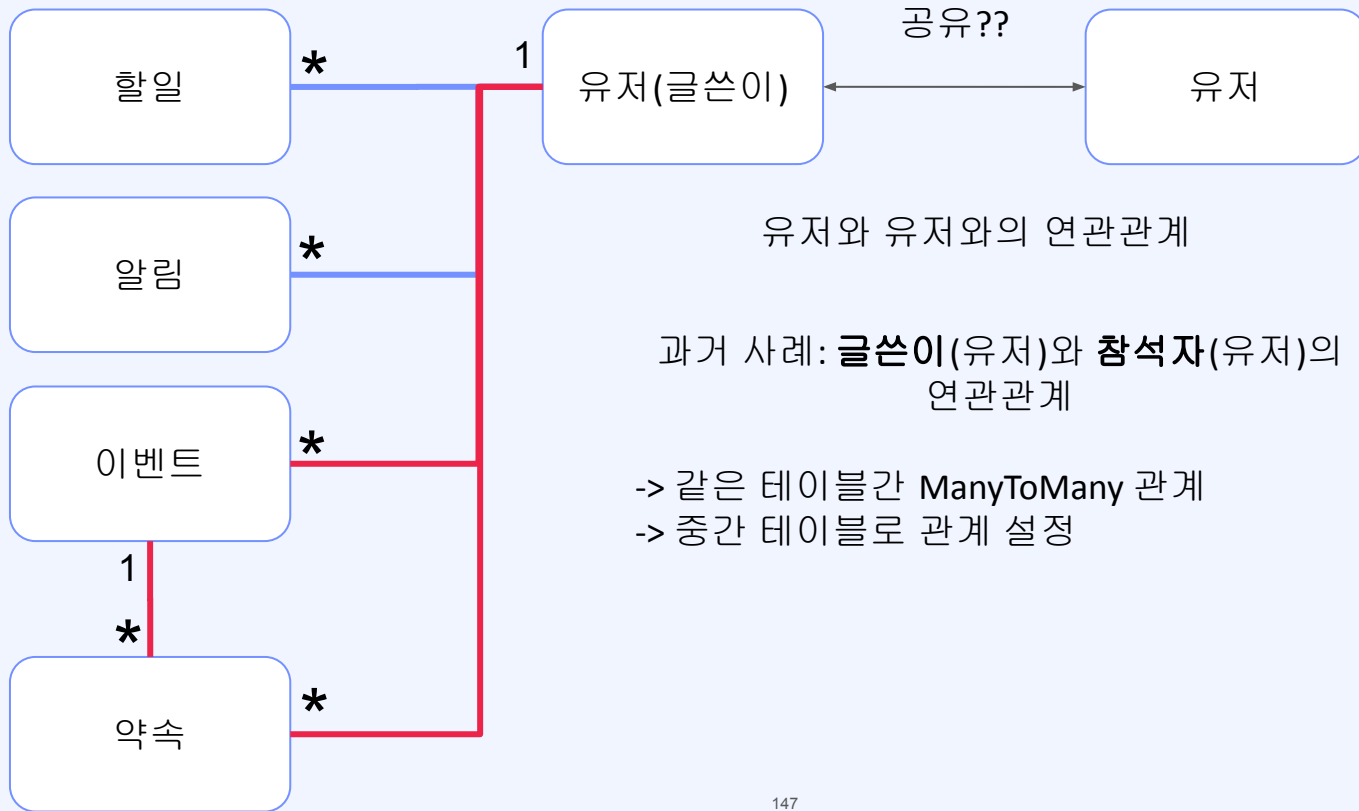
## 도메인 정의 - 최종 관계도



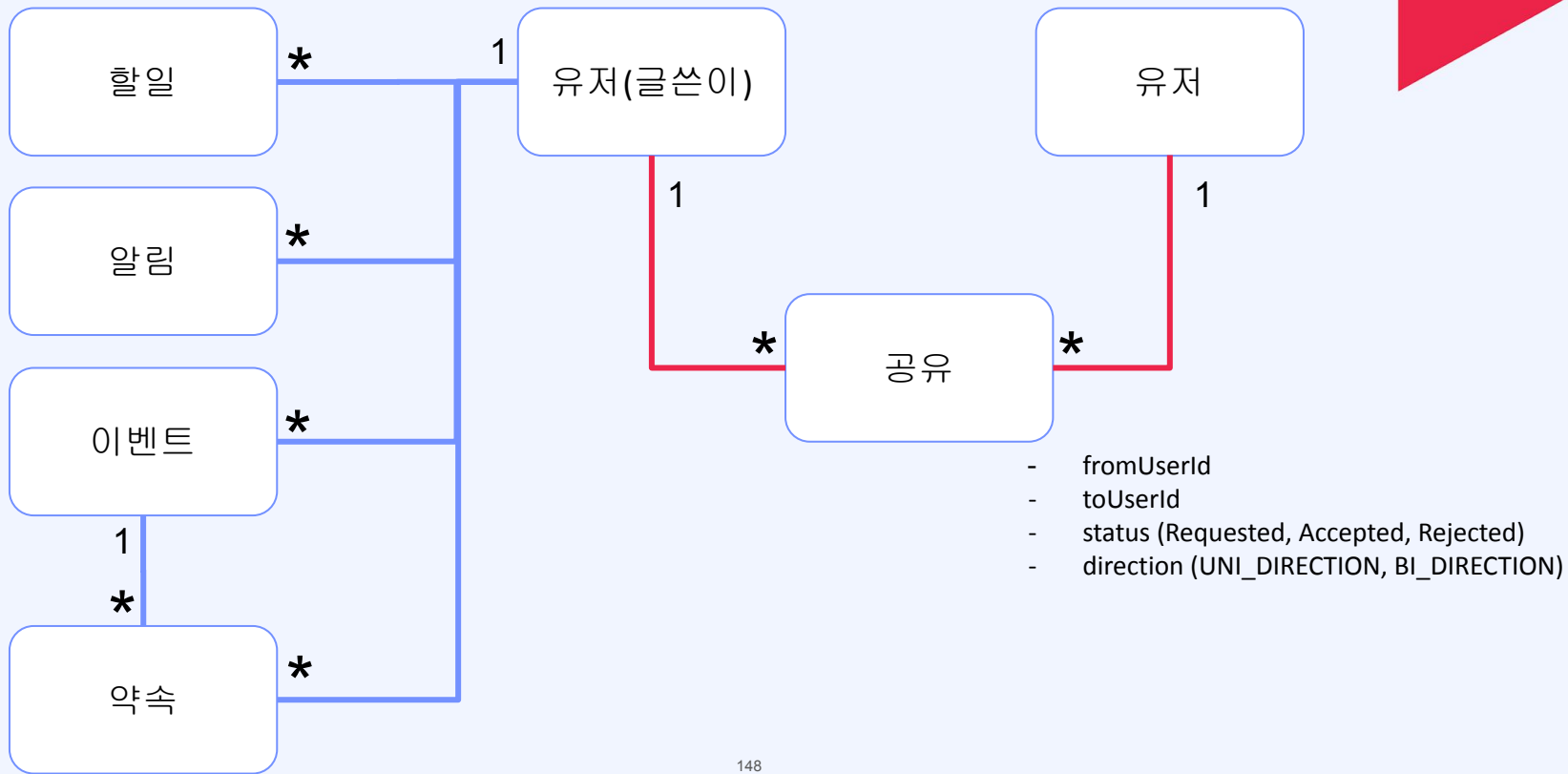
## 도메인 정의 - 최종 관계도



## 도메인 정의 - 최종 관계도



## 도메인 정의 - 최종 관계도



## API 스펙

### 공유하기 요청

요청

POST /api/schedules/shares/{shareId}

```
{  
  "toUserId": 1,  
  "direction": "BI_DIRECTION"  
}
```

응답

200 OK

### 공유하기 응답

요청

PUT /api/schedules/shares/{shareId}

```
{  
  "type": "REJECT"  
}
```

응답

200 OK

# API 스펙

## 스케줄 조회 공통 응답

```
[[ // 여러명의 유저 스케줄을 보여줘야 하기 때문에 리스트
  {
    "userId": 3,
    "name": "name1",
    "me": false, // 자기 자신 여부
    "scheduleResList": [[ // 이하 기존 응답
      {
        "scheduleId": 3,
        "startAt": "2021-08-15T23:00:00",
        "endAt": "2021-08-17T12:00:00",
        "title": "제주도 여행",
        "description": null,
        "writer": {
          "id": 4,
          "name": "다두미",
          "email": "jchaevv@gmail.com",
          "birthday": null
        },
        "scheduleType": "EVENT"
      }
    ]
  }
]]
```

# 캘린더 공유하기 추가 기능 개발

3 상세 기능 구현

# 공유하기 도메인 서비스 로직 개발



## 코딩 시간

### 할 일

- Share 엔티티 추가
- 생성, 수정 로직 구현

git branch : share-domain



# 공유하기 적용된 스케줄 조회 api 개발

## 코딩 시간

### 할 일

- 스케줄 api 수정

git branch : share-query



# 캘린더 공유하기 추가 기능 개발

4 마무리

# 개선사항 도출, 도전과제 안내

## 개선사항 도출 & 도전과제 안내

### 테스트 코드 부족

- 가능한 테스트는 많이 만들기!

### 공유하기 생성 Validation

- A -> B 단방향 요청이 있고, 이후 B -> A 로 다시 요청을 한다면? (fromUserId, toUserId 를 묶어서 복합유니크 키 필요)

# 프로젝트 개발 회고

## 사용했던 기술들

java (oop, stream(no-forloop), functional)

bcrypt

lombok

springboot

spring jpa (entity, repository, entity listener, jpql, transaction)

spring mvc (session, resolver, exception handler, restTemplate, profile, template engine)

spring batch (job, step, reader, cursor, pageReader, chunk)

spring email

gradle (multi module)

bash (batch script)

cron

http client tool (insomnia)

docker

mysql



## Next up!

### For Dev

- Spring Security
- QueryDSL
- Logging Framework
- MicroService
- Async, NonBlocking
- 설정에 집착해보기 (특히나 스프링 부트는 기본으로 설정해 주는 부분이 크기 때문에, 디테일한 설정을 놓치기 쉽습니다.)

### For Infra

- CI,CD (젠킨스 등)

### CS

- RDBMS (query, index)
- Network (http)

# 축하합니다!

길고 긴 강의 따라오시느라  
정말 수고 많으셨습니다! 🙌🙌🙌

수강생 여러분들  
저보다 훌륭한 개발자 될 수 있기를 바랍니다!

