채팅 어플

이거 완전 세미나 시간이잖아~

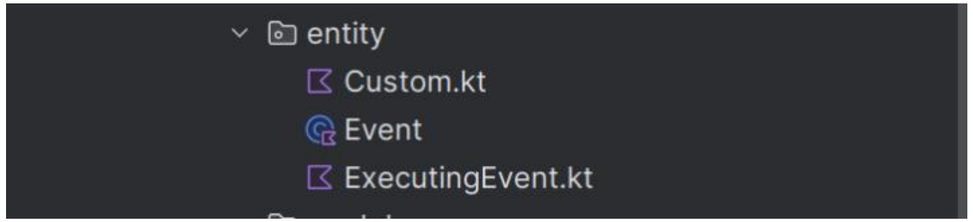
박승빈

진행 목차

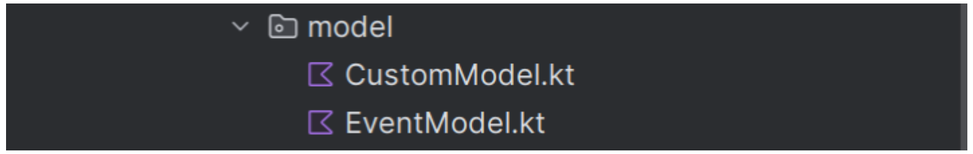
01 Kotlin?

JetBrains에서 만든 언어로 자바와 완벽 호환되며 앱 개발에 주로 사용됨

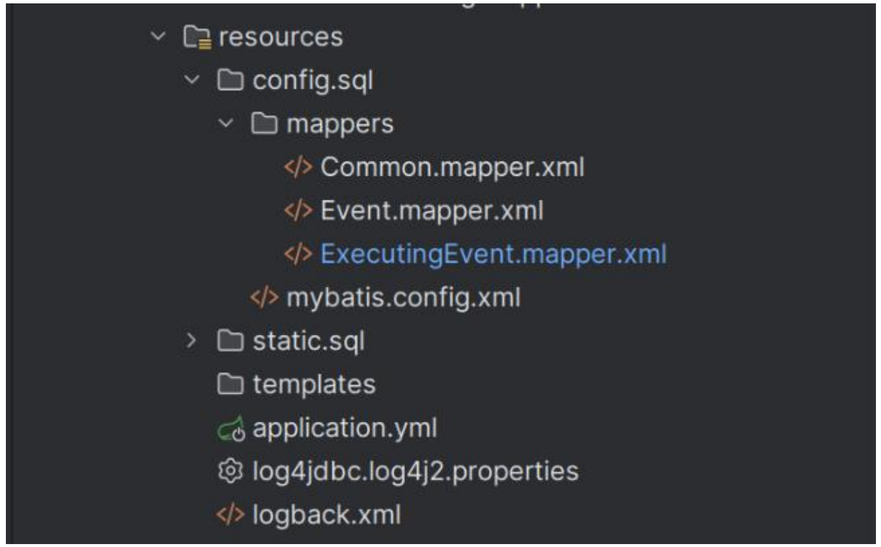
02 현 진행 상황



기초가 되는 데이터 클래스

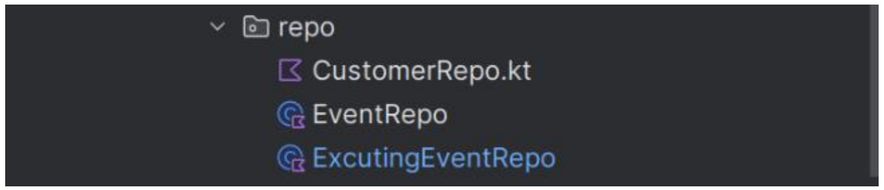


SQL 구문 별로 필요한 데이터 클래스

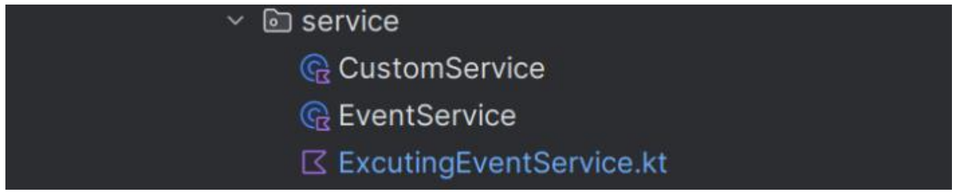


.xml -> Mybatis들을 위한 sql 구문 저장소

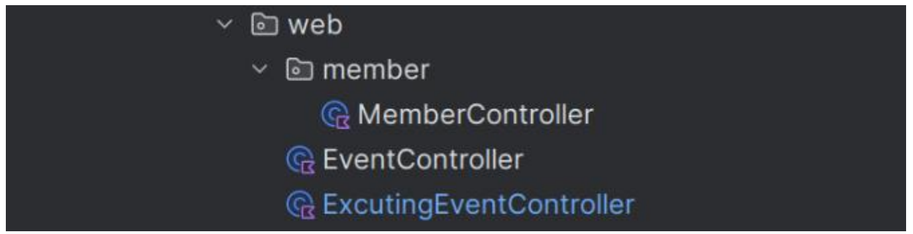
02 현 진행 상황



DB에 직접적으로 접근하는 클래스

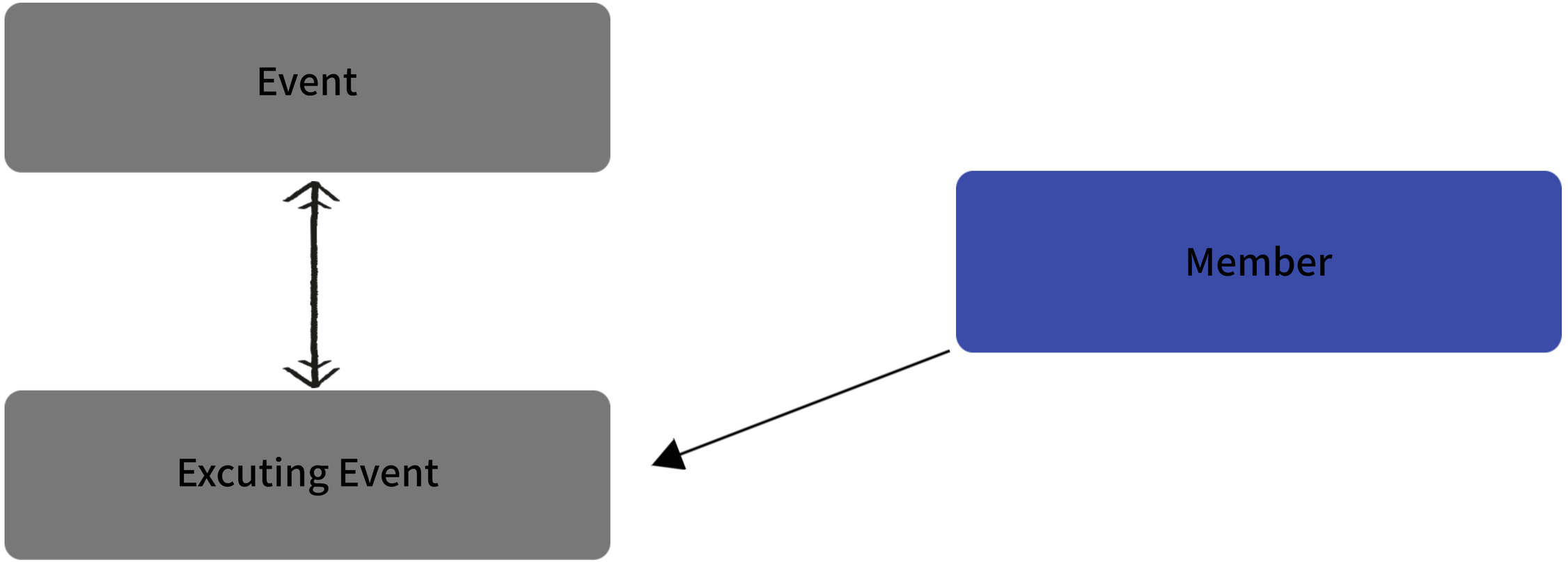


로직들을 관리하는 클래스



sql 구문별로 명령어를
서버에 전달하는 코드

02 현 진행 상황



03 코드설명

```
data class Event(  Park SeungBin
    val id : Long?,
    var title : String,
    var eventType : String,
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern= DATE)
    var regDate: LocalDate,
    var description: String,
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern= DATETIME)
    var createDt : LocalDateTime,
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern= DATETIME)
    var updateDt : LocalDateTime?,
    var createUserId : Long,
    var updateUserId : Long?
)
```

```
data class SearchEvent(  Park SeungBin +1
    val title : String?,
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern= DATETIME)
    val fromFromDate : LocalDateTime?,
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern= DATETIME)
    val fromToDate : LocalDateTime?,
    val statusOfEvent : String?// 진행상태 ( before, ing, after)
)


data class UpdateEvent(  Park SeungBin +1
    var id: Long,
    var title: String,
    var eventType : String,
    var description : String,
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern= DATETIME)
    var updateDt : LocalDateTime,
    var updateUserId : Long
)


data class InsertEvent(  Park SeungBin
    var title : String,
    var eventType : String,
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern= DATETIME)
    var regDate: LocalDateTime?,
    var description: String,
    @JsonFormat(shape = JsonFormat.Shape.STRING, pattern= DATETIME)
    var createDt : LocalDateTime,
    var createUserId : Long
)
```

03 코드설명

```
@Repository  Park SeungBin
class EventRepo(
    private val sqlSessionTemplate: SqlSessionTemplate
) {
    fun getEvent(id: Long) =  Park SeungBin
        sqlSessionTemplate.selectOne<Event>( statement: "EVENT.selectEvent", id)

    fun getListEvents(searchEvent: SearchEvent) =  Park SeungBin
        sqlSessionTemplate.selectList<Event>( statement: "EVENT.selectListEvents", searchEvent)

    fun insertsEvents(event: InsertEvent) = sqlSessionTemplate.insert( statement: "EVENT.insertingEvent", event)  Pa


    fun selectListEventsByStatus(searchEvent: SearchEvent) =  Park SeungBin
        sqlSessionTemplate.selectList<ExecutingEvent>( statement: "EVENT.selectListEventsByStauts", searchEvent)

    fun deleteEvent(id : Long)=  Park SeungBin
        sqlSessionTemplate.selectOne<Event>( statement: "EVENT.deleteEvent", id)

    fun updateEvent(updatingEvent: UpdateEvent)=  Park SeungBin
        sqlSessionTemplate.selectList<Event>( statement: "EVENT.updateEvent", updatingEvent)
}
```

03 코드설명

```
@Service  Park SeungBin
class EventService(
    private val eventRepo: EventRepo,
    repo: EventRepo
){
    fun getEvent(id: Long) = eventRepo.getEvent(id)  Park SeungBin

    fun getListEvents(searchEvent: SearchEvent) = eventRepo.getListEvents(searchEvent)  Park SeungBin

    fun insertListEvent(insertEvent: InsertEvent) = eventRepo.insertsEvents(insertEvent)  Park SeungBin

    fun selectListEventsByStatus(searchEvent: SearchEvent) = eventRepo.selectListEventsByStatus(searchEvent)

    fun deleteEvent(id : Long)= eventRepo.deleteEvent(id)  Park SeungBin

    fun updatesEvent( updating:UpdateEvent) = eventRepo.updateEvent(updating)  Park SeungBin

}
```

03 코드설명

```
@RestController  Park SeungBin
@RequestMapping("/v1/events")
class EventController(private val eventService: EventService) {
    @GetMapping("/{id}")  Park SeungBin
    fun getEvent(@PathVariable id: Long) = eventService.getEvent(id)

    @PostMapping("/list")  Park SeungBin
    fun getListEvents(@RequestBody searchEvent: SearchEvent) = eventService.getListEvents(searchEvent)

    @PostMapping("/insert")  Park SeungBin
    fun insertListEvent(@RequestBody insertEvents : InsertEvent) = eventService.insertListEvent(insertEvents)

    @GetMapping("/status")  Park SeungBin
    fun selectListEventsByStatus(@RequestBody searchEvent: SearchEvent) = eventService.selectListEventsByStatus(sea

    @DeleteMapping("/{id}")  Park SeungBin
    fun deleteEvent(@PathVariable id: Long) = eventService.deleteEvent(id)

    @PostMapping("/update")  Park SeungBin
    fun updateEvents(@RequestBody updateEVent: UpdateEvent) = eventService.updatesEvent(updateEVent)

}
```

03 코드설명

```
<!--    </insert>-->
<select id="selectEvent" resultType="event">
    SELECT
        id,
        title,
        event_type,
        reg_date,
        description,
        create_dt,
        update_dt,
        create_user_id,
        update_user_id
    FROM
        event
    where id = #{value}
</select>
```

```
<insert id="insertingEvent">
    INSERT
    INTO
        event
    (title,
     event_type,
     reg_date,
     description,
     create_dt,
     create_user_id)
    VALUES (
                #{title},
                #{eventType},
                #{regDate} ,
                #{description},
                #{createDt},
                1)
</insert>
```

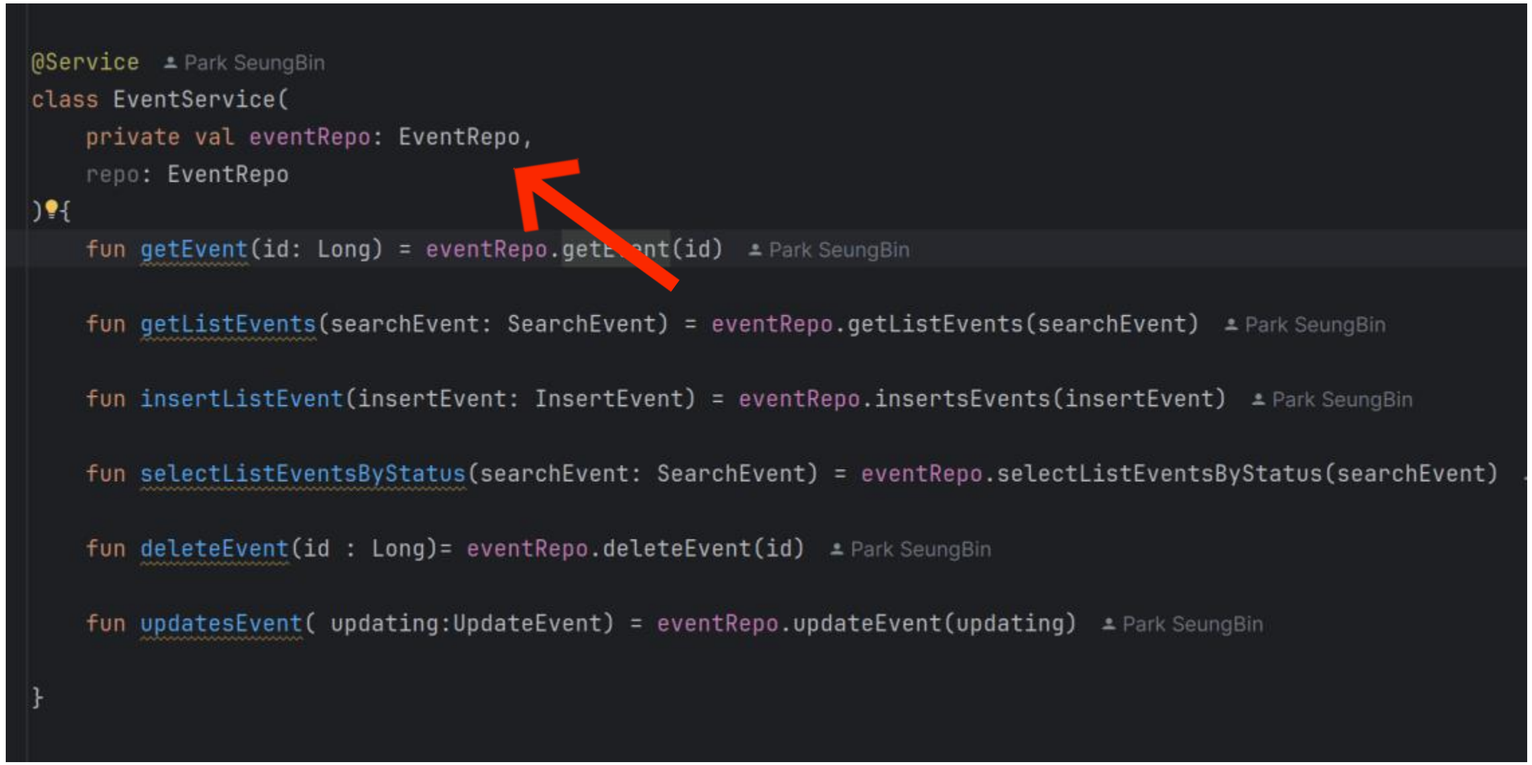
```
<delete id="deleteEvent">
    DELETE FROM event
    WHERE id = #{id}
</delete>


<update id="updateEvent">
    UPDATE event
    SET title = #{title},
        event_type = #{eventType},
        description = #{description},
        update_dt = #{updateDt},
        update_user_id = #{updateUserId}
    WHERE id = #{id};
</update>
```

느낀점

이 프로젝트를 통해서 코틀린에 대한 지식도 얻은것이 있지만
어떠한 프로젝트를 할때 환경 설정들은 어떻게 해야되는지, 그리고 프로젝트에 대한 계획을 왜 만들어서
체계적으로 진행해야되는지 알게되는 프로젝트였습니다.

느낀점



만들면서 여러 에러들, 그중에서는 며칠을 고민한 에러들도 있었지만
해결하는것에 있어서 행복감을 느껴볼 수 있었던 것 같습니다.
그리고 개발하는데에 있어서는 개발툴이 좋으면 퀄리티가 25만배정도 향상된다는것을 깨우칠 수 있었습니다.

https://github.com/sb-sfamily/spring-kotlin