
채팅 어플

이거 완전 세미나 시간이잖아~

진행 목차

01 언어 설명

02 현 진행 상황

03 코드 설명

04 마무리

01 Kotlin?



JetBrains에서 만든 언어로 자바와 완벽 호환되며 앱 개발에 주로 사용됨

02 현 진행 상황

```
entity
├── Custom.kt
├── Event
└── ExecutingEvent.kt
```

기초가 되는 데이터 클래스

```
model
├── CustomModel.kt
└── EventModel.kt
```

SQL 구문 별로 필요한 데이터 클래스

```
resources
├── config.sql
├── mappers
│   ├── Common.mapper.xml
│   ├── Event.mapper.xml
│   └── ExecutingEvent.mapper.xml
├── mybatis.config.xml
├── static.sql
├── templates
├── application.yml
├── log4jdbc.log4j2.properties
└── logback.xml
```

.xml -> Mybatis들을 위한 sql 구문 저장소

02 현 진행 상황

```
▼ repo
  CustomerRepo.kt
  EventRepo
  ExcutingEventRepo
```

DB에 직접적으로 접근하는 클래스

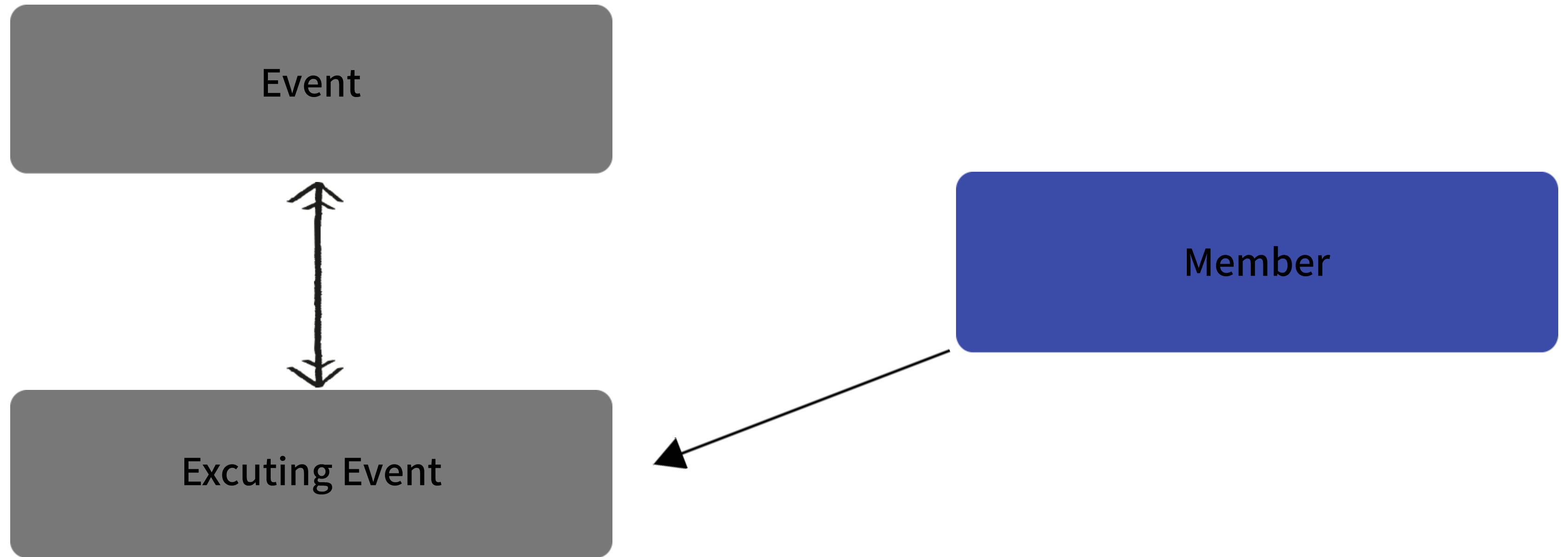
```
▼ service
  CustomService
  EventService
  ExcutingEventService.kt
```

로직들을 관리하는 클래스

```
▼ web
  ▼ member
    MemberController
    EventController
    ExcutingEventController
```

sql 구문별로 명령어를
서버에 전달하는 코드

02 현 진행 상황



03 코드설명

```

9 data class Event( ㉸ Park SeungBin
10     val id : Long?,
11     var title : String,
12     var eventType : String,
13     @JsonFormat(shape = JsonFormat.Shape.STRING, pattern= DATE)
14     var regDate: LocalDate,
15     var description: String,
16     @JsonFormat(shape = JsonFormat.Shape.STRING, pattern= DATETIME)
17     var createDt : LocalDateTime,
18     @JsonFormat(shape = JsonFormat.Shape.STRING, pattern= DATETIME)
19     var updateDt : LocalDateTime?,
20     var createUserId : Long,
21     var updateUserId : Long?
22 )

```

```

7 data class SearchEvent( ㉸ Park SeungBin +1
8     val title : String?,
9     @JsonFormat(shape = JsonFormat.Shape.STRING, pattern= DATETIME)
10     val fromFromDate : LocalDateTime?,
11     @JsonFormat(shape = JsonFormat.Shape.STRING, pattern= DATETIME)
12     val fromToDate : LocalDateTime?,
13     val statusOfEvent : String?// 진행상태 ( before, ing, after)
14 )
15
16
17 data class UpdateEvent( ㉸ Park SeungBin +1
18     var id: Long,
19     var title: String,
20     var eventType : String,
21     var description : String,
22     @JsonFormat(shape = JsonFormat.Shape.STRING, pattern= DATETIME)
23     var updateDt : LocalDateTime,
24     var updateUserId : Long
25 )
26
27
28 data class InsertEvent( ㉸ Park SeungBin
29     var title : String,
30     var eventType : String,
31     @JsonFormat(shape = JsonFormat.Shape.STRING, pattern= DATETIME)
32     var regDate: LocalDateTime?,
33     var description: String,
34     @JsonFormat(shape = JsonFormat.Shape.STRING, pattern= DATETIME)
35     var createDt : LocalDateTime,
36     var createUserId : Long
37 )

```

03 코드설명

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

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

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

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

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

    fun updateEvent(updatingEvent: UpdateEvent)=
        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.updateEvent(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

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

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

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

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

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

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

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>

