Timo.GG DB 사용

초기 세팅

- 1. MariaDB 의존성 설치
- 2. QueryDSL 의존성 설치

DB 초기화

MariaDB 설치

- 윈도우로 MariaDB 설치

터미널 세팅

- `mariadb -u root`
- `CREATE DATABASE _데이터베이스명_;`
- `CREATE USER '_아이디_'@'%' IDENTIFIED BY '_비밀번호_';`
- `GRANT ALL PRIVILEGES ON _데이터베이스명_.* TO '_아이디_'@'%';`
- `FLUSH PRIVILEGES;`
- `USE demoTimo;`

Application.properties 설정

```
spring.datasource.driver-class-name=org.mariadb.jdbc.Driver
spring.datasource.url=jdbc:mariadb://localhost:3306/demoTimo
spring.datasource.username=사용자계정 아이디
spring.datasource.password=사용자계정 비밀번호

spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.show-sql=true
spring.jpa.generate-ddl=true
```

Repository 생성

```
@Repository
public interface PostRepository extends JpaRepository<Post, Long>,
QuerydslPredicateExecutor<Post> {
}
```

JPAQuery Method

Config 설정

```
@Configuration
@EnableJpaAuditing
@RequiredArgsConstructor
public class Config {
    private final EntityManager em;

```

```
    @Bean
    public JPAQueryFactory jpaQueryFactory() {
        return new JPAQueryFactory(em);
    }
}
```

BaseEntity 생성

```
@MappedSuperclass
@EntityListeners(AuditingEntityListener.class)
@Getter
public class BaseEntity {
    @CreatedDate
    @Column(name = "createdAt", updatable=false)
    private LocalDateTime createdAt;

    @LastModifiedDate
    @Column(name="modifiedAt")
    private LocalDateTime modifiedAt;
}
```

PostEntity 생성

```
@Entity
@AllArgsConstructor
@NoArgsConstructor
@Data
@Builder
public class PostEntity extends BaseEntity {
    @Id
    @GeneratedValue(strategy= GenerationType.IDENTITY)
    private Long id;

    @Column(length=100, nullable=false)
    private String title;

    @Column(length=1500, nullable=false)
    private String content;

    @Column(length=50, nullable=false)
    private String writer;
}
```

더미 데이터 생성

```

@SpringBootTest
class PostRepositoryTest {
    @Autowired private PostRepository postRepository;

    @Autowired private JPAQueryFactory queryFactory;

    @Test
    public void insertDummies(){
        IntStream.rangeClosed(1, 300).forEach(i -> {
```

```
            PostEntity post = PostEntity.builder()
                    .title("Title..." + i)
                    .content("Content..." + i)
                    .writer("user"+ (i % 10))
                    .build();
            System.out.println(postRepository.save(post));
        });
    }
    @Test
    public void testQuery1(){
        Pageable pageable = PageRequest.of(0, 10, Sort.by("id").descending());
        Page<PostEntity> page = postRepository.findAll(pageable);
        //TODO: findById, findByIdGreaterThan, findByIdContains
        page.stream().forEach( PE -> System.out.println(PE));
    }
    @Test
    public void testQuery2(){
        List<PostEntity> result = queryFactory.select(QPostEntity.postEntity)
                .from(QPostEntity.postEntity)
//                .where(QGuestbook.guestbook.id.eq(1L))
                .where(
                        QPostEntity.postEntity.title.contains("1")
                                .or(QPostEntity.postEntity.content.contains("1")))
                .fetch();
        result.stream()
                .forEach(guestbook -> {
                    System.out.println(guestbook);
                });
    }
}
```