Seoul Software ACademy

웹 개발자 부트캠프 과정

SeSAC x CODINGOn

With. 팀 뤼쳐드

ORM

ORM 이란?

- Object Relational Mapping
  - Object : 객체 지향 언어의 객체
  - Relational : 관계형 데이터베이스 (Relational Database) 의 관계
  - Mapping : 객체 지향 언어의 객체와 관계형 데이터를 서로 변환해준다.
- Ex) JPA, Hibernate 등등

JPA

JPA 란?

- Java Persistence API
- 자바 진영의 ORM 기술 표준
- JPA 가 개발자 대신 적합한 SQL을 생성하고 DB 에 전달하고, 객체를 자동으로 Mapping 해주기에 SQL 을 직접 작성할 필요가 없다.
- EX) Hibernate (JPA 를 구현한 대표적 오픈소스)

JPA 장단점

- 장점
  - 생산성이 뛰어나고 유지보수가 용이하다.
  - DBMS에 대한 종속성이 줄어든다.

- 단점
  - JPA의 장점을 살려 잘 사용하기 위해서는 학습 비용이 높고, 복잡한 쿼리를 사용할 때 불리하다.
  - 잘못 사용할 경우 SQL을 직접 사용하는 것보다 성능이 떨어질 수 있다.

JPA 사용하기

build.gradle

```
implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
```

- JPA 를 사용하기 위해 dependency(라이브러리) 추가

application.properties 수정

```
# mysql connect
spring.jpa.database=mysql
spring.jpa.database-platform=org.hibernate.dialect.MySQL5InnoDBDialect

# DB function use
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=create
# none: not create / create: auto create / update: only update structure

logging.level.org.hibernate=info

spring.jpa.properties.hibernate.show_sql=true
spring.jpa.properties.hibernate.format_sql=true
spring.jpa.properties.hibernate.use_sql_comments=true
```

Entity 란?

- 데이터베이스에 쓰일 필드와 여러 Entity 간의 관계를 설정하는 것
- @Entity 를 이용해 해당 클래스가 Entity임을 알려주고, JPA에 정의된 필드를 바탕으로 데이터베이스에 테이블을 만들 수 있다.

domain/UserEntity

```java
@Entity
@Table(name="user")
public class UserEntity {
    2 usages
    @Id
    @GeneratedValue
    private int id;

    2 usages
    @Column(length=10, nullable=false)
    private String name;

    2 usages
    @Column(length=10, nullable=false)
    private String nickname;


    kyuriiii
    public int getId() { return id; }
    kyuriiii
    public void setId(int id) { this.id = id; }
    kyuriiii
    public String getName() { return name; }
    1 usage  kyuriiii
    public void setName(String name) { this.name = name; }
    kyuriiii
    public String getNickname() { return nickname; }
    1 usage  kyuriiii
    public void setNickname(String nickname) { this.nickname = nickname; }
}
```

Entity 관련 Annotation

- @Entity : 데이터베이스의 Table 임을 의미한다.
- @Table : 테이블의 이름 명시
- @Id : primary key를 의미
- @GeneratedValue : primary key의 생성 전략
- @Column : 테이블의 컬럼을 의미

Repository란?

- Entity에 의해 생성된 DB에 접근하는 메소드를 사용하기 위한 인터페이스
- JpaRepository를 상속받으면 기본적인 DB 접근 메소드를 사용할 수 있다.
  - findAll()
  - findById()
  - findBy컬럼명()
  - save()

repository/UserRepository

```
2 usages  kyuriiii
@Repository
public interface UserRepository extends JpaRepository<UserEntity, Integer> {
    1 usage  kyuriiii
    Optional<UserEntity> findByName(String name);
}
```

Optional이란?

- Null 일 수도 있는 객체를 감싸는 Wrapper 클래스
- Optional<T> option
  - Option 변수 내부에는 null이 아닌 T 객체가 있을 수도 있고 null이 있을 수도 있다.
  - 즉, Optional 클래스는 여러 가지 API를 제공해 null일 수도 아닐 수도 있는 객체를 다룰 수 있다.

service/MainService

```java
@Service
public class MainService {
    3 usages
    @Autowired
    private UserRepository userRepository;

    1 usage  kyuriiii
    public List<UserDTO> getUserList(){
        List<UserEntity> result = userRepository.findAll();
        List<UserDTO> users = new ArrayList<~>();

        for ( int i = 0; i < result.size(); i++ ) {
            UserDTO user = new UserDTO();
            user.setId(result.get(i).getId());
            user.setName(result.get(i).getName());
            user.setNickname(result.get(i).getNickname());
            user.setNo(i+1);

            users.add(user);
        }
        return users;
    }
    1 usage  kyuriiii *
    public ArrayList<UserDTO> getUserName(String name){
        Optional<UserEntity> user = userRepository.findByName(name);
        ArrayList<UserDTO> userList = new ArrayList<>();

        if ( user.isPresent() ) {
            UserDTO dto = new UserDTO();
            dto.setId(user.get().getId());
            dto.setNo(0);
            dto.setName(user.get().getName());
            dto.setNickname(user.get().getNickname());
            userList.add(dto);
        }
        return userList;
    }
    1 usage  kyuriiii
    public void addUser(UserEntity user) { userRepository.save(user); }
```

controller/MainController

```java
@Controller
public class MainController {
    3 usages
    @Autowired
    MainService mainService;

    new *
    @GetMapping("/users")
    public String getUsers(Model model) {
        ArrayList<UserDTO> userList = (ArrayList<UserDTO>) mainService.getUserList();
        model.addAttribute( attributeName: "list", userList);
        return "user";
    }
    new *
    @GetMapping("/user")
    public String getUser(@RequestParam String name, Model model) {
        ArrayList<UserDTO> userList = mainService.getUserName(name);

        model.addAttribute( attributeName: "list", userList);
        return "user";
    }

    new *
    @GetMapping("/user/insert")
    public String getInsertUser(@RequestParam String name, @RequestParam String nickname, Model model) {
        UserEntity user = new UserEntity();
        user.setName(name);
        user.setNickname(nickname);

        mainService.addUser(user);

        model.addAttribute( attributeName: "list",  attributeValue: null);
        return "user";
    }
```

실습. 사용자 조회

- 지금 수업에서 다룬 내용에 추가로 아래 작업들 진행하기
  - 새로운 사용자 이름, 닉네임으로 user 추가
  - 사용자 이름으로 조회 (n명)
  - 검색어를 보냈을 때 사용자 이름과 일치하거나 닉네임과 일치할 경우 조회 (n명)
  - 이름이 존재하는지 조회

실습. 게시판 시스템

- MyBatis 로 만들었던 게시판 실습 JPA로 구현하기