



It's Your Life

with





간단한

DAO 구현 실습



실습 1

테이블, 데이터 쿼리

실습 1. 회원 테이블 만들기 & 데이터 삽입하기

- 아래의 조건을 만족하는 DATABASE 와 TABLE 을 만들어 주세요
- DATABASE 이름 : user_ex
- TABLE 이름 : users
- user 테이블의 컬럼과 조건
 - id : 숫자, 자동 생성 및 자동 숫자 증가, PRIMARY KEY
 - email : 최대 100 자의 문자, 중복 허용 X, 필수 값
 - password : 최대 100 자의 문자, 필수 값

실습 1. 회원 테이블 만들기 & 데이터 삽입하기

- 만들어진 users 테이블에 아래의 조건을 만족하는 회원 데이터를 넣어 주세요
 - 회원 1, email : tetz / password : 1234
 - 회원 2, email : siwan / password : 1234
 - 회원 3, email : na / password : 1234
- 테이블이 정상적으로 생성이 되면 아래와 같은 결과가 나와야 합니다!

	id	email	password
1	1	tetz	1234
2	2	siwan	1234
3	3	na	1234

SELECT * FROM users;
쿼리를 실행했을 때 왼쪽과 같은
테이블이 나와야 합니다!

▼ user_ex

▼ Tables

▶ users

워크 벤치의 데이터베이스 스키마 화면

Result Grid | Filter Rows:

	id	email	password
▶	1	tetz	1234
	2	siwan	1234
	3	na	1234
•	NULL	NULL	NULL

SELECT * FROM users; 를
워크 벤치에서 실행한 화면



실습 2

Gradle 프로젝트 생성 및 DB 접속하기

실습 2. Gradle 프로젝트 생성 & DB 접속



- 인텔리제이를 이용하여 새로운 프로젝트를 생성해 주세요
- 프로젝트 이름 : jdbc_dao_ex
- 빌드 시스템 : Gradle
- JDK : JDK 17 버전 아무거나
- Gradle DSL : Groovy
- Add Sample code 옵션만 체크 / Tip 은 체크 불필요

실습 2. Gradle 프로젝트 생성 & DB 접속



- 프로젝트 루트 폴더에 user.sql 파일을 만들고 실습 1 에서 진행한 SQL 쿼리를 복사 붙여넣기 해주세요
- 그리고 data source 를 실습 1에서 생성한 user_ex 로 설정해 주세요
- SELECT * FROM users; 쿼리 실행 시 아래와 같이 테이블이 나와야 합니다

	id	email	password
1	1	tetz	1234
2	2	siwan	1234
3	3	na	1234

실습 2. Gradle 프로젝트 생성 & DB 접속



- build.gradle 파일을 열고 Dao 작성을 위한 라이브러리를 추가해 주세요
- 라이브러리 정보는 기존에 쓰시던 것이 있으면 그것을 사용, 없으시면 슬랙 게시물에 있는 라이브러리 코드 블록의 코드를 dependencies 에 추가해 주세요
- 필요 라이브러리는 MySQL 접속용 mysql-connector 와 Lombok 입니다

실습 2. Gradle 프로젝트 생성 & DB 접속



- UserDao 클래스를 만들고 UserDao 클래스를 이용해 방금 만든 user_ex 데이터 베이스에 접속해 주세요.
 - 접속 방법은 기존 수업에서 사용했던 JDBCUtil 을 사용하지 말고, 직접 문자열 형태로 DB 서버 접속 정보를 Driver 에 전달하는 형식으로 접속해 주세요.
 - 접속 시점(static 시점, 인스턴스화 되는 시점 등등)은 편하신 방법으로 선택

실습 2. Gradle 프로젝트 생성 & DB 접속



- UserDao 클래스 운영을 위한 UserMain 클래스를 만들어 주세요
- 아래의 UserMain 코드를 실행했을 때, 아래와 같은 결과 화면이 나오면 됩니다!

```
3 ▶ public class UserMain {  
4 ▶     public static void main(String[] args) {  
5         UserDao userDao = new UserDao();  
6     }  
7 }
```

✓ jdbc_dao_ex [:Us 1 sec, 178 ms

오후 1:46:16: Executing ':UserMain.main()'...

> Task :compileJava

> Task :processResources NO-SOURCE

> Task :classes

> Task :UserMain.main()

DB 접속에 성공!



실습 3

UserVo(Value
Object) 구현하기

실습 3. UserVo 구현하기



- users 테이블에 대응되는 UserVo 클래스를 구현하세요
- Getter / Setter / equals / toString / 생성자는 Lombok 의 어노테이션을 사용하세요

실습 3. UserVo 구현하기



- 아래의 UserMain 코드 실행 시 아래와 같은 결과가 나오면 됩니다!

```
3 ▶ public class UserMain {  
4 ▶   public static void main(String[] args) {  
5       UserDao userDao = new UserDao();  
6  
7       UserVo tetz = new UserVo(id: 1, email: "tetz", password: "1234");  
8  
9       System.out.println(tetz);  
10    }  
11 }
```

```
> Task :UserMain.main()  
DB 접속에 성공!  
UserVo(id=1, email=tetz, password=1234)
```



실습 4

UserDao 에

회원 CRUD 구현하기

실습 4. 회원 CRUD 구현하기



- users 테이블에 회원을 생성, 조회, 업데이트, 삭제하는 CRUD 기능을 UserDao 에 구현해 주세요
- 아래와 같은 UserMain 코드가 실행 되었을 때, 다음과 같은 결과 화면이 뜨도록 구현해 주시면 됩니다!



```
3 ▶ public class UserMain {  👤 kdtTetz *  
4 ▶     public static void main(String[] args) {  👤 kdtTetz *  
5         UserDao userDao = new UserDao();  
6         // 회원 추가  
7         userDao.create(email: "tetz2", password: "1234");  
8         // 추가 회원 조회  
9         userDao.getAllUsers();  
10  
11        // 회원 수정 메서드 실행  
12        userDao.updateUser(id: 4, newEmail: "lhs", newPassword: "abcd");  
13        // 수정 회원 조회  
14        userDao.getAllUsers();  
15  
16        // id가 4인 회원 삭제 메서드 실행  
17        userDao.deleteUser(id: 4);  
18        // 삭제 회원 조회  
19        userDao.getAllUsers();  
20    }  
21 }
```



```
> Task :UserMain.main()
```

```
DB 접속에 성공!
```

```
회원 추가 성공!
```

```
UserVo(id=1, email=tetz, password=1234)
```

```
UserVo(id=2, email=siwan, password=1234)
```

```
UserVo(id=3, email=na, password=1234)
```

```
UserVo(id=4, email=tetz2, password=1234)
```

```
회원 정보 수정 성공!
```

```
UserVo(id=1, email=tetz, password=1234)
```

```
UserVo(id=2, email=siwan, password=1234)
```

```
UserVo(id=3, email=na, password=1234)
```

```
UserVo(id=4, email=lhs, password=abcd)
```

```
회원 삭제 성공!
```

```
UserVo(id=1, email=tetz, password=1234)
```

```
UserVo(id=2, email=siwan, password=1234)
```

```
UserVo(id=3, email=na, password=1234)
```



실습 5

JOIN 을 사용하여
회원 이름 출력하기

실습 5. JOIN 을 사용하여 회원 이름 출력



- 먼저 JOIN 을 위한 테이블을 추가 합시다!

```
28  # user_info 테이블 생성
29  # 해당 테이블은 user 테이블의 id 를 외래키(FOREIGN KEY) 로 가지며 회원 id 를 가지고
30  # 회원의 이름을 알 수 있는 목적을 가지는 테이블 입니다
31  • CREATE TABLE user_info
32  (
33      id    INT PRIMARY KEY,
34      name  VARCHAR(50) NOT NULL,
35      # user 테이블의 id 와 user_info 의 id 가 서로 참조하는 관계임을 외래키로 설정
36
37      FOREIGN KEY (id) REFERENCES users (id) ON DELETE CASCADE
38      # JOIN 문 연습을 위해, 억지로 만든 테이블이며 해당 테이블은 제2 정규형을(2NF)를 위배합니다
39      # name 컬럼도 id 에 종속이기 때문에 해당 테이블은 굳이 따로 나눌 필요가 없기 때문입니다
40  );
```

- SQL 코드가 필요하신 분은 아래 링크로 고고고

- https://github.com/xenosign/jdbc_dao_ex/blob/main/user.sql

실습 5. JOIN 을 사용하여 회원 이름 출력



• 테스트를 위한 데이터 추가

```
47 # 각각의 테이블에 필요한 데이터 삽입
48 • INSERT INTO user_info (id, name)
49 VALUES ('1', '이효석'),
50          ('2', '김시완'),
51          ('3', '나건우');
```

Result Grid		
	id	name
▶	1	이효석
	2	김시완
	3	나건우
●	NULL	NULL

실습 5. JOIN 을 사용하여 회원 이름 출력



- users 테이블과 user_info 테이블을 합쳐서 회원 목록 조회 시, 회원의 이름 정보가 같이 출력되는 getAllUsersWithName 메서드를 UserDao 에 추가해 주세요!
- 아래의 UserMain 코드 실행 시, 다음과 같은 결과 화면이 나와야 합니다!



```
public class UserMain {  👤 kdtTetz *  
    public static void main(String[] args) {  👤 kdtTe  
        UserDao userDao = new UserDao();  
  
        // 이름이 출력 안되는 회원 조회 메서드  
        userDao.getAllUsers();  
        // 이름이 출력되는 회원 조회 메서드  
        userDao.getAllUsersWithName();  
    }  
}
```

```
> Task :UserMain.main()
```

DB 접속에 성공!

UserVo(id=1, email=tetz, password=1234)

UserVo(id=2, email=siwan, password=1234)

UserVo(id=3, email=na, password=1234)

ID: 1, Email: tetz, Password: 1234, Name: 이효석

ID: 2, Email: siwan, Password: 1234, Name: 김시완

ID: 3, Email: na, Password: 1234, Name: 나건우