Spring Boot 3장

2020-04-25 양시영

JPA

Java Persistence API

JAVA ORM 표준

MyBatis != ORM

MyBatis == SQL Mapper

Why JPA

JAVA == Object Oriented != RDB

Escape SQL Hell

Focus Object Oriented Programming

Escape SQL Hell

User Table : id, name, address

Create User Table CRUD SQL

-> User Table : id, name, address, email

Modify User Table CRUD SQL

If we use JPA

Modify Just Entity

```java
@Getter
@NoArgsConstructor
@Table(name = "user")
@Entity
public class User {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    @Column(unique = true)
    private String name;

    @Column
    private String password;

    @Column
    private String role;

    @Builder
    public User(Long id, String name, String password, String role) {
        this.id = id;
        this.name = name;
        this.password = password;
        this.role = role;
    }

    public void encodePassword(PasswordEncoder passwordEncoder) {
        this.password = passwordEncoder.encode(this.password);
    }

    public static void checkExistId(String id) {

    }

    public static void checkPassword(String password) {

    }

}
```

What’s mean “JAVA == Object Oriented != RDB” ?

유저 / 유저가 구매한 상품

유저 / 유저가 구매한 상품

유저 Class

```java
@Getter
@Setter
public class User {
    private Integer id;
    private String name;
    private String address;
    private String email;
}
```

상품 Class

```java
@Setter
@Getter
public class Item {
    private Integer id;
    private String itemName;
}
```

DB와 매핑할 Class

```java
@Setter
@Getter
public class UserItem {
    private Integer itemId;
    private Integer userId;
    private String itemName;
    private String userName;
    private String userAddress;
    private String userEmail;
}
```

유저 / 유저가 구매한 상품

Query

```
SELECT
    user.id as userId
    , user.name as userName
    , user.address as userAddress
    , user.email as userEmail
    , item.id as itemId
    , item.name as itemName
FROM User user, Item item
WHERE user.id = item.user_id
```

DB와 매핑할 Class

```
@Setter
@Getter
public class UserItem {
    private Integer itemId;
    private Integer userId;
    private String itemName;
    private String userName;
    private String userAddress;
    private String userEmail;
}
```

유저 / 유저가 구매한 상품

유저 Class

```java
@Getter
@Setter
public class User {
    private Integer id;
    private String name;
    private String address;
    private String email;
}
```

상품 Class

```java
@Setter
@Getter
public class Item {
    private Integer id;
    private String itemName;
}
```

DB와 매핑할 Class

```java
@Setter
@Getter
public class UserItem {
    private Integer itemId;
    private Integer userId;
    private String itemName;
    private String userName;
    private String userAddress;
    private String userEmail;
}
```

DB 때문에 만들어진 Class
사실은 불필요

What’s mean “JAVA == Object Oriented != RDB” ?

유저 / 유저가 구매한 상품

```java
@Getter
@Builder
@Entity
public class User {

    @GeneratedValue
    @Id
    private Integer id;

    private String name;

    private String address;

    private String email;

    @OneToMany(mappedBy = "user")
    private List<Item> itemList;
}
```

```java
@Getter
@Builder
@Entity
public class Item {

    @GeneratedValue
    @Id
    private Integer id;

    private String name;

    @JoinColumn(name = "user_id")
    @ManyToOne
    private User user;

}
```

user.getItemList() // item.getUser()