Specification

Created	@Mar 9, 2021 11:42 PM
Tags	DDD

🧐 Specification이란...

Specification에 대해서 한번 알아보고자 한다

```
public interface Specification<T> {
  Predicate toPredicate(Root<T> root, CriteriaQuery query, CriteriaBuilder cb);
}
```

Specification은 Spring Data JPA에서 개발자가 Predicate 생성만 해주면 나머지 쿼리를 알아서 만들어주는 좋은 친구이다 🙂

`Specification`이라는 것은 `Interface`로써 `Predicate`를 반환하는 `toPredicate()` 만 구현하면 된다

`toPredicate()` 가 받는 인자값을 보면 JPQL 사용을 쉽게 할 수 있도록 도와주는 Criteria api를 사용하는 것을 볼 수 있다 그래서 본인이 Criteria를 사용해본적이 있다면 큰 무리없이 사용할 수 있을 것이다

`toPredicate()`의 인자값

- `Root<T> root` : 조회를 시작하는 객체를 주입받는다
- `CriteriaQuery query` : query를 주입받는다 (어떤 쿼리를 주입받는지는 모
- `CriteriaBuilder cb` : query를 생성할 수 있는 빌더를 주입받는 곳이다

🖌 `toPredicate()` 사용하기

User 엔티티 작성

```
@Entity
class User {
  @Id
  private Long id;
  private String password;
  private String email;
}
```

위와 같은 User 엔티티가 있다고 하겠습니다

우리는 email이 `peach@kakao.com`을 가지는 User를 찾아보도록 하겠습니다

Spring Data Jpa를 이용해서 DAO 작성

```
public interface UserDao extends
    JpaRepository<User, Long>, JpaSpecificationExecutor {

}
```

위와 같이 `JpaSpecificationExecutor`이라는 인터페이스를 상속받아도 됩니다

`JpaSpecificationExecutor` 에는

`findOne()`, `findAll()`, `count()` ... 메서드가 존재하는데

```
public interface UserDao extends JpaRepository<User, Long> {
  public List<T> findAll(Specification<T> specification);
}
```

위와 같이 상속받지 않고 필요한 메서드만 작성해도 상관없이 동작한다

참고로 직접 코드를 작성해주는 것이면 반드시 findAll()이라는 이름으로 생성해야 한다 임의로 findByEmail 이런식으로 생성하면 엄청난 오류 메세지를 보게될 것이다

테스트 해보기

```
@Test
void testGetUserSameEmail() {
  String email = "peach@kakao.com"
  Specification<User> spec = ((root, query, cb) -> cb.equal(root.get("email"), email);
```

```
  User user = userDao.findAll(spec);

}
```

정상 작동하는 것을 볼 수 있지만 사실상 서비스 계층에서 Specification을 작성해줄거면 저렇게 작성하기 보다

DAO 계층에서 Criteria나 JPQL로 작성하는 것이 더 나을 수 도 있다

👏 Specification을 이용하는 이유

위에서 우리는 간단하게 Specification을 이용방법을 알아보았다

하지만 위에서 처럼 사용하게 된다면 뭔가 2% 모자란 사용이다

우리는 Specification 코드 생성하는 것을 User 엔티티에다가 static 메서드로 옮길 것이다

```
@Entity
class User {
  @Id
  private Long id;
  private String password;
  private String email;

  public static Specification<User> findByEmail(String email) {
    return ((root, query, cb) -> cb.equal(root.get("email"), email);
  }
}
```

```
@Test
void testGetUserSameEmail() {
  String email = "peach@kakao.com"
  List<User> user = userDao.findAll(User.findByEmail(email));

}
```

User 엔티티로 메서드를 옮김으로써 위와 같이 코드를 작성할 수 있을 것이다

위와 같이 리팩토링을 진행함으로써 우리는 Specification을 이용해서 좀 더 도메인 관점에서 개발을 할 수 있게 되었다

결론 : 도메인 관점에서 개발을 하기위해서 Specification을 사용한다...