5-3 : Spring Data JPA

들어가기 전에

이번 시간에는…

Spring Data JPA 개요, 개념, 원리
쿼리 메서드 기능

test 도메인 부분 JPA에서 Spring Data JPA로 변경 실습
Optional, lambda식 활용

과제 설명

Spring Data JPA

Spring Data JPA

스프링 프레임워크 + JPA 기반 하에 JPA를 편리하게 사용하게 해주는 라이브러리

Spring Data JPA

Spring Data JPA 활용

Spring Data JPA는 JpaRepository를 상속받는 인터페이스만 구현해주면 곧바로 활용할 수 있다.
JpaRepository 인터페이스는 기본적인 CRUD 및 필요한 메서드를 제공

```
public interface MemberRepository extends JpaRepository<Member, Long> {
}
```

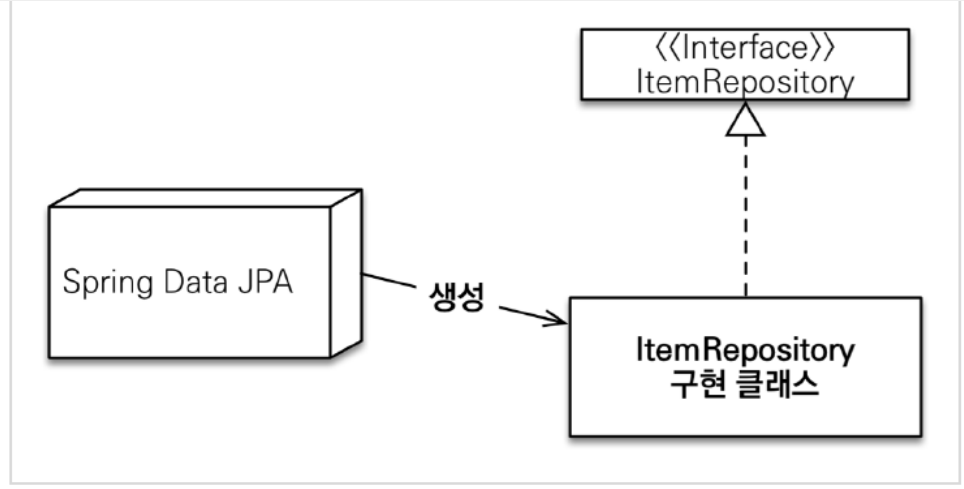
제네릭은 <엔티티 클래스 타입, 식별자(PK) 타입>으로 설정

Spring Data JPA

Spring Data JPA 원리

Spring Data JPA는 인터페이스만 구현해도
내부적으로 Spring Data JPA가 구현 클래스를 대신 생성해준다.

위와 같은 원리로 인터페이스만 구현해도 기본 구현 메서드를 활용할 수 있다.



Spring Data JPA

Spring Data JPA 주요메서드

save(S) : 새로운 엔티티는 저장하고 이미 있는 엔티티는 병합한다.

findAll(...) : 모든 엔티티를 조회한다. 정렬(Sort)이나 페이징(Pageable) 조건을 파라미터로 제공할 수 있다.

findById(ID) : 엔티티 하나를 조회한다. 내부에서 EntityManager.find() 호출

delete(T) : 엔티티 하나를 삭제한다. 내부에서 EntityManager.remove() 호출

쿼리 메서드

Spring Data JPA 쿼리 메서드 기능

주요 공통 메서드 외에 다른 메서드를 구현하고자 할 때 사용

Spring Data JPA 인터페이스를 상속 받아 추가 메서드를 구현하기에는
공통 메서드를 모두 오버라이딩 받아야하기 때문에 매우 비효율적

따라서, Spring Data JPA에서는 다음과 같은 쿼리 메서드 기능을 제공한다.

주요 쿼리 메서드 기능 종류

1. 메서드 이름으로 쿼리 생성
2. @Query
3. @EntityGraph

쿼리 메서드

Spring Data JPA 쿼리 메서드 기능
– 메서드 이름으로 쿼리 생성

메서드 이름을 분석해서 JPQL 쿼리를 실행
조건이 간단하고, 짤막한 쿼리 생성 시 효과적

- **조회**: find…By ,read…By ,query…By get…By
- **COUNT**: count…By 반환타입 long
- **EXISTS**: exists…By 반환타입 boolean
- **삭제**: delete…By, remove…By 반환타입 long
- **DISTINCT**: findDistinct, findMemberDistinctBy
- **LIMIT**: findFirst3, findFirst, findTop, findTop3

공식 문서: https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#jpa.query-methods.query-creation

쿼리 메서드

Spring Data JPA 쿼리 메서드 기능
– 메서드 이름으로 쿼리 생성

메서드 이름을 분석해서 JPQL 쿼리를 실행
조건이 간단하고, 짤막한 쿼리 생성 시 효과적

```
public interface MemberRepository extends JpaRepository<Member, Long> {
    List<Member> findByUsernameAndAgeGreaterThan(String username, int age);
}
```

쿼리 메서드

Spring Data JPA 쿼리 메서드 기능
– @Query

@Query 어노테이션을 활용
JPA Repository에 메서드와 쿼리를 동시에 정의하여 활용할 수 있다.

기본적으로 메서드 이름 쿼리 생성으로 사용 후, 조건이 많아지면 @Query를 주로 사용

```
public interface MemberRepository extends JpaRepository<Member, Long> {
    @Query("select m from Member m where m.username= :username and m.age = :age")
    List<Member> findUser(@Param("username") String username, @Param("age") int age);
}
```

쿼리 메서드

Spring Data JPA 쿼리 메서드 기능
– @Query

@Query 어노테이션을 활용
JPA Repository에 메서드와 쿼리를 동시에 정의하여 활용할 수 있다.

Entity 조회
+
파라미터 바인딩

```
public interface MemberRepository extends JpaRepository<Member, Long> {
    @Query("select m from Member m where m.username= :username and m.age = :age")
    List<Member> findUser(@Param("username") String username, @Param("age") int age);
}
```

DTO 조회

```
// DTO로 직접 조회
// DTO 패키지 경로를 모두 적어줘야 한다는 번거로움이 있다. -> QueryDSL 사용으로 해결 가능.
@Query("select new study.datajpa.dto.MemberDto(m.id, m.username, t.name) " +
        "from Member m join m.team t")
List<MemberDto> findMemberDto();
```

쿼리 메서드

Spring Data JPA 쿼리 메서드 기능
– @EntityGraph

연관된 엔티티들을 SQL 한 번에 조회하는 방법
@EntityGraph 어노테이션을 활용하여 Fetch Join을 간단하게 구현할 수 있는 방법

```
//JPQL + 엔티티 그래프
@EntityGraph(attributePaths = {"team"})
@Query("select m from Member m")
List<Member> findMemberEntityGraph();

//메서드 이름으로 쿼리에서 특히 편리하다.
@EntityGraph(attributePaths = {"team"})
List<Member> findByUsername(String username)
```

실습 & 과제 안내

test 도메인 부분 JPA를 Spring Data JPA로 변경

1. 요구사항에 맞게 test 도메인의 POST, GET, PATCH API
JDBC Template에서 Spring Data JPA 로 변경

2. Optional, Lambda 식을 활용한 코드 구현(선택 사항)

API 명세서: https://documenter.getpostman.com/view/16596703/2s8YzQX4SU#introduction

과제 안내

Spring Data JPA로 리팩토링

4주차에 구현한 SpringBoot 템플릿 코드를 Spring Data JPA를 활용하여 다시 재구현

과제 유의사항

최대한 Spring Data JPA를 활용하여 구현(JDBC Template 지양)

N + 1 설정(지연 로딩, Fetch Join, BatchSize 설정) 1개 이상씩 구현

수업에서 배운 Spring Data JPA 쿼리 메서드 기능 1개 이상씩 구현

최대한 Optional, Lambda 식을 활용해서 구현해보기

Hello World!