45주차 발표_민서연

Hibernate

01 Hibernate

자바 진영의 다양한 ORM 프레임워크 중 가장 많이 사용되는 프레임워크

Hibernate 기반으로 만들어진 ORM 기술 표준이 JPA

JDBC API를 내부적으로 사용하고 JPA의 특징, 장점을 그대로 가지고 직접 사용 할 수 있게 해준다.

ORM(Object-Relational Mapping):
객체 지향 프로그래밍 언어와 관계형 데이터베이스 간의 데이터 변환을 자동으로 처리하는 기술

JDBC(Java Database Connectivity):
애플리케이션에서 데이터베이스에 데이터를 저장하거나 업데이트, 접근할 수 있도록 도와주는 자바 API

01 Hibernate

Hibernate 사용 방법

1. 엔티티 클래스 정의

JPA 어노테이션을 사용해 데이터베이스 테이블과 매핑되는 엔티티 클래스를 정의한다.

@Entity, @Table, @Id, @Column

```java
@Entity
@Getter
@NoArgsConstructor(access = AccessLevel.PROTECTED)
public class User {

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

    private String userEmail;

    private String userPassword;

    private String userNickname;

    private LocalDate userAge;

    private String userProfileImg;
```

01 Hibernate

Hibernate 사용 방법

2. EntityManager 생성

EntityManagerFactory를 통해 EntityManager를 생성한다.

데이터베이스 연결, 트랜잭션 관리를 수행한다.

```
EntityManagerFactory emf = Persistence.createEntityManagerFactory("myPersistenceUnit");
EntityManager em = emf.createEntityManager();
```

01 Hibernate

Hibernate 사용 방법

3. CRUD 작업 수행

EntityManager 를 구현한 메소드를 사용해서 엔티티에 대한 CRUD 작업을 수행하면 된다.

```
// Create
em.getTransaction().begin();
User user = new User("Chaeyami", "chaeyami@example.com");
em.persist(user);
em.getTransaction().commit();

// Read
User foundUser = em.find(User.class, user.getId());

// Update
em.getTransaction().begin();
foundUser.setEmail("chaeyami02@example.com");
em.merge(foundUser);
em.getTransaction().commit();

// Delete
em.getTransaction().begin();
em.remove(foundUser);
em.getTransaction().commit();
```

01 Hibernate

Hibernate 사용 방법

4. 쿼리 작성

JPQL, 네이티브 SQL, Querydsl 등을 사용할 수 있다.

```
// JPQL
List<User> users = em.createQuery("SELECT u FROM User u", User.class).getResultList();

// Native SQL
List<User> users = em.createNativeQuery("SELECT * FROM users", User.class).getResultList();
```

02 Hibernate와 JPA와 Spring Data JPA

JPA = 기술 명세

JPA는 특정 기능을 하는 라이브러리가 아닌 관계형 db를 사용하는 방식을 정의한 인터페이스이다.

즉 자바 어플리케이션에서 관계형데이터베이스를 어떻게 사용해야 하는지를 정의하는 한 방법이다.

Hibernate = JPA의 구현체

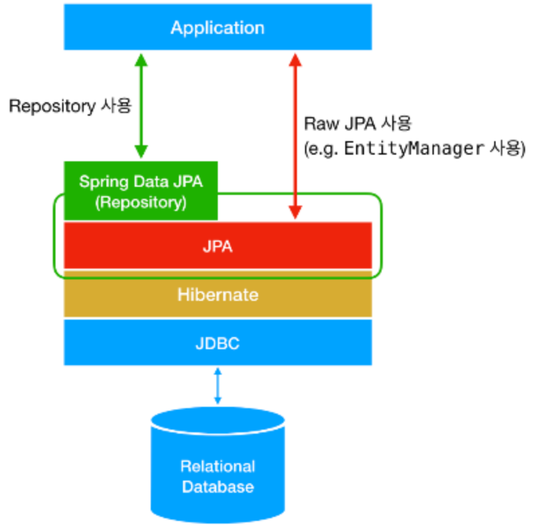
java.persistence.EntityManager와 같은 인터페이스를 직접 구현한 라이브러리이다.

02 Hibernate와 JPA와 Spring Data JPA

Spring Data JPA = JPA를 쓰기 편하게 만들어 둔 모듈

Spring Data JPA는 JPA를 더 쉽고 편하게 사용할 수 있도록 추상화시킨 Repository라는 인터페이스를 제공한다.

사용자가 Repository 인터페이스에 정해진 규칙대로 메소드를 입력하면 Spring이 알아서 해당 메소드 이름에 적합한 쿼리를 날리는 구현체를 만들어서 Bean으로 등록해준다.



03 Fetch Join 최적화

쿼리의 N+1 발생 문제

```
select m from Member m join m.team
```

join을 써서 데이터를 조회하게 되는 경우

Member 전체 조회하는 쿼리 호출 + Memeber와 연관있는 Team을 조회하는 쿼리 호출

총 두번의 쿼리가 발생한다.

엔티티의 연관관계는 지연로딩으로 fetch타입을 설정을 따로 해주는데 `@ManyToOne(fetch = FetchType.LAZY)`

여기서 일반 Join을 사용할 경우 N+1 문제가 발생한다.

03 Fetch Join 최적화

Fetch Join

```
// TeamRepository.java
@Query("SELECT distinct t FROM Team t join fetch t.members")
public List<Team> findAllWithMemberUsingFetchJoin();
```

연관 관계의 엔티티나 컬렉션을 프록시(가짜 객체) 가 아닌 진짜 데이터를 같이 조회하는 기능이다.

JPQL에서 지원하는 기능으로, 성능 최적화를 위해 사용된다.

연관된 엔티티나 컬렉션을 SQL로 한번에 조회한다.

03 Fetch Join 최적화

- 일반 JOIN
  - Fetch JOIN과 달리 연관 엔티티에 Join을 걸어도 실제 쿼리에서 조회하는 엔티티는 오직 JPQL에서 조회하는 주최가 되는 엔티티만 조회해서 영속화한다.
- FETCH JOIN
  - 조회의 주최가 되는 엔티티 이외에 Fetch Join이 걸린 연관 엔티티도 함께 조회해서 모두 영속화 한다.

03 Fetch Join 최적화

- 일반 JOIN

```
TeamRepository.java
@Query("SELECT distinct t FROM Team t JOIN t.members")
public List<Team> findAllWithMembersUsingJoin();
```

```
select
     distinct team0_.id as id1_1_,
     team0_.name as name2_1_
from team team0_
inner join
    member members1_
        on team0_.id=members1_.team_id
```

- Fetch JOIN

```
// TeamRepository.java
@Query("SELECT distinct t FROM Team t JOIN FETCH t.members")
public List<Team> findAllWithMembersUsingFetchJoin();
```

```
Hibernate:
    select
         distinct team0_.id as id1_1_,
         members1_.id as id1_0_1_,
         team0_.name as name2_1_,
         members1_.age as age2_0_1,
         members1_.name as name3_0_1_,
         members1_.team_id as team_id4_0_1_,
         members1_.team_id as team_id4_0_1_,
         members1_.id as id1_0_1_
    from team team0_
    inner join
        member members1_
            on team0_.id=members1_.team_id
```

04 출처

https://velog.io/@luke9701/JPA-Hibernate%EB%9E%80-%EB%AC%B4%EC%97%87%EC%9D%B4%EA%B3%A0-%EC%99%9C-%EC%82%AC%EC%9A%A9%ED%95%98%EB%8A%94%EA%B0%80
https://chaeyami.tistory.com/263
https://chaeyami.tistory.com/256
https://suhwan.dev/2019/02/24/jpa-vs-hibernate-vs-spring-data-jpa/
https://velog.io/@guns95/Fetch-Join%ED%8C%A8%EC%B9%98-%EC%A1%B0%EC%9D%B8-EntityGraphe