JPA 01 자바 표준 ORM 표준 JPA 프로그래밍

김영한
2022.08.01

JPA? Java Persistence API

sql → sql mapper : jdbc, mybatis → sql 직접작성x jpa

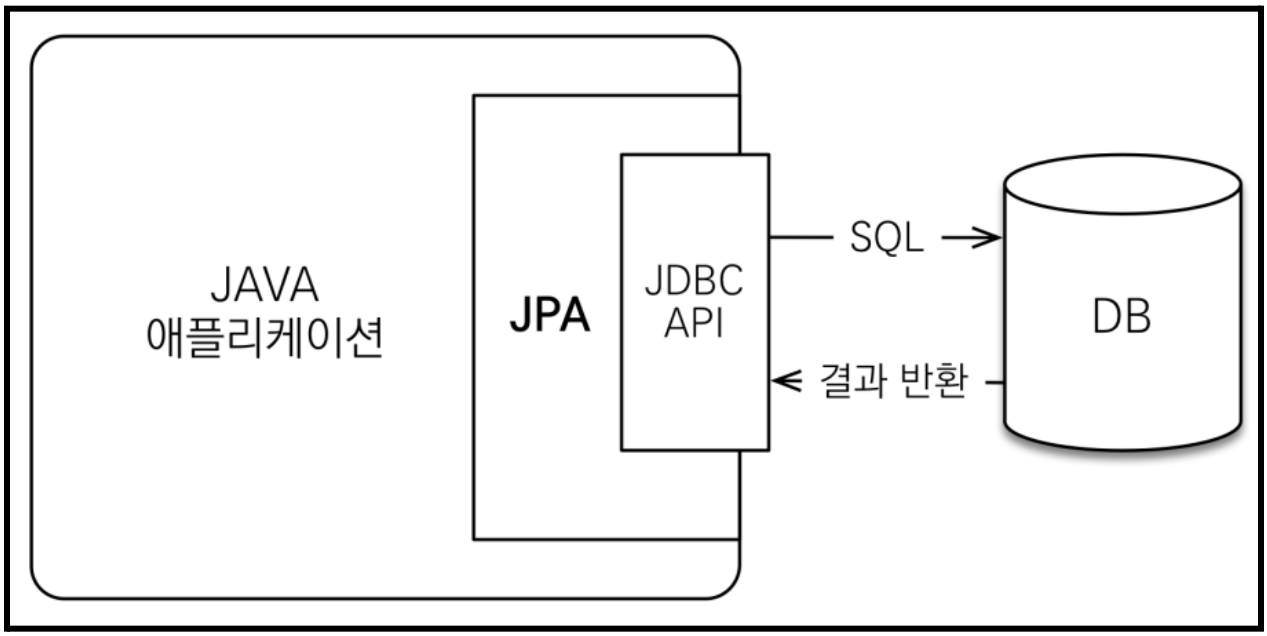
객체와 테이블 설계 매핑 정확히 매핑해서 실무에 적용해야한다.

JPA 동작 방식을 이해하고 사용해야 유지보수나 정확한 쿼리를 사용되는지 아는게 중요

ORM?

Object - relational - mapping (객체 관계 매핑)

- 객체는 객체대로 — 관계형DB는 관계형DB로 설계
- 프레임워크가 중간에 매핑 대부분 언어가 매핑은 가지고 있다.



패러다임의 불일치를 **JPA** 가 해결해준다

JPA 사용 이유

생산성

- CRUD 가 매우매우 간편해 진다. (자바 컬렉션처럼 DB를 쓰자는게 컨셉이다.)

유지 보수

패러다임의 불일치를 해소

- 상속 , 연관관계, 객체 그래프 탐색

비교**,** 성능 최적화 기능

- **1**차 캐시와 동일성**(identity)** 보장 **:** (같은 트랜잭션 안에서는 같은 엔티티를 반환), 캐시를 사용해서 중복된 sql을 동일하게 보내진다. [트랜잭션 동안 짧은 시간의 캐싱]
- 트랜잭션을 지원하는 쓰기 지연 **:** 트랜잭션을 커밋할 때까지 INSERT SQL을 모은 후에 한번에 전송한다.

지연 로딩과 즉시 로딩

- 지연 로딩 **(LAZY)**

  객체가 실제 사용 될 때 로딩, 필요한 순간에 로딩된다.

- 즉시 로딩 **(FATCH)**

  JOIN SQL로 한번에 연관된 객체까지 미리 조회

프로젝트 생성 (**Maven**)

```
<dependencies>
   <!-- JPA 하이버네이트 -->
   <dependency>
       <groupId>org.hibernate</groupId>
       <artifactId>hibernate-entitymanager</artifactId>
       <version>5.3.10.Final</version>
   </dependency>
   <!-- H2 데이터베이스 -->
   <dependency>
       <groupId>com.h2database</groupId>
       <artifactId>h2</artifactId>
       <version>1.4.200</version>
   </dependency>
</dependencies>
```

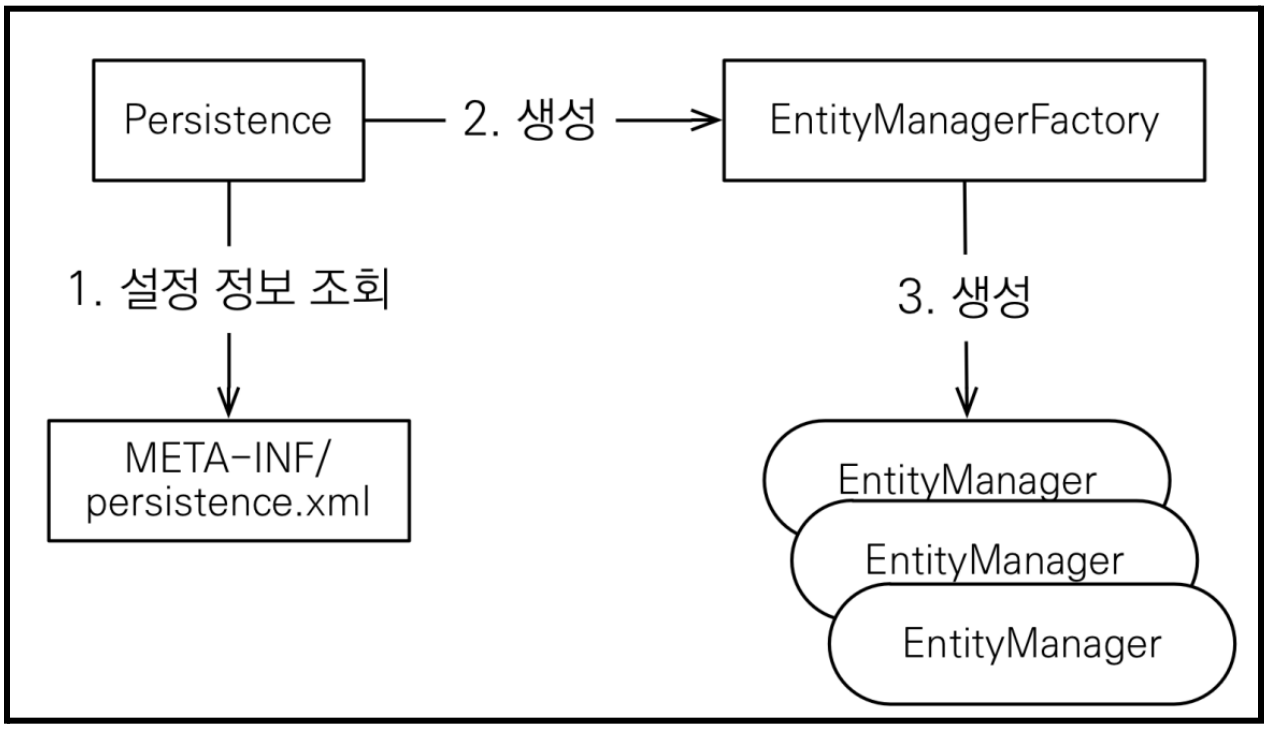
/resources/META-INF/persistence.xml 생성 해줘야 JPA 를 사용할 수 있다.

```
<?xml version="1.0" encoding="UTF-8"?>
<persistence version="2.2"
             xmlns="http://xmlns.jcp.org/xml/ns/persistence"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
             xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/persistence
http://xmlns.jcp.org/xml/ns/persistence/persistence_2_2.xsd">
    <persistence-unit name="hello">
        <properties> <!-- 필수 속성 -->
            <property name="javax.persistence.jdbc.driver"
value="org.h2.Driver"/>
            <property name="javax.persistence.jdbc.user" value="sa"/>
            <property name="javax.persistence.jdbc.password" value=""/>
            <property name="javax.persistence.jdbc.url"
value="jdbc:h2:tcp://localhost/~/test"/>
            <property name="hibernate.dialect"
value="org.hibernate.dialect.H2Dialect"/>
            <!-- 옵션 -->
            <property name="hibernate.show_sql" value="true"/>
            <property name="hibernate.format_sql" value="true"/>
            <property name="hibernate.use_sql_comments" value="true"/>
           <!--<property name="hibernate.hbm2ddl.auto" value="create" />-->
        </properties>
    </persistence-unit>
</persistence>
```

Dialect 데이터베이스 방언

여러 데이터 베이스에 따라 sql사용이 조금씩 다르기 때문에 그 속성을 맞춰주기 위해 써주는 것.

JPA 실행 과정



JPQL

JPA를 사용하면 엔티티 객체를 중심으로 개발

문제는 검색 쿼리 (join)등 테이블이 아닌 엔티티 객체를 대상으로 검색

어플리케이션이 필요한 데이터만 DB에서 불러오려면 결국 검색 조건이 포함된 SQL이 필요.

JPQL은 엔티티 객체를 대상으로 한 쿼리이다**.**

SQL을 추상화해서 데이터베이스 SQL에 의존하지 않는다.

EntityManagerFactory에서

EntityManager를 가져오고 쿼리 변경에 대해서는 무조건

Transaction 단위를 사용하기 때문 사 열어서 사용하고 닫아줘야한다.

```
public static void main(String[] args) {
       EntityManagerFactory emf = Persistence.createEntityManagerFactory("hello");

       EntityManager em = emf.createEntityManager();

       EntityTransaction tx = em.getTransaction();
       tx.begin();

       try {
//            Member findMember = em.find(Member.class, 1L);
           List<Member> result = em.createQuery("select m from Member m",
Member.class)
                   .setFirstResult(1)
                   .setMaxResults(10)
                   .getResultList();

           for (Member member : result) {
               System.out.println("member.name = " + member.getName());
           }
//            findMember.setName("HelloJPA");

           //더티 채킹 변경감지해서 쿼리를 날려준다.
           tx.commit();
       } catch (Exception e){
           tx.rollback();
       } finally {
           em.close();
       }
       emf.close();
   }
```

영속성 컨텍스트

“엔티티를 영구 저장하는 환경**”**

EntityManager.persist(entity); → 디비에 저장한다는 뜻이아니라 주어진 entity를 영속성 컨텍스트 라는 곳에 저장한다는 뜻. 엔티티를 영속화 시킨다.

의미

- 영속성 컨텍스트는 보이지 않고 논리적 개념이다.

이점**(**장점**)**

1차 캐시 가능, 동일성**(identity)**,

엔티티 등록 트랜잭션을 지원하는 쓰기 지연 [커밋하는 순간 insert] ,

변경감지 - 트랜잭션이 종료될때 [변경 감지] dirty Checking 하여 update문을 날려준다.

삭제 **-** 대상을 조회하고 remove하면 간단하게 삭제가 가능

생명 주기

- 비영속(new/transient) 전혀 관계없는 상태
- 영속(managed) 관리되는 상태
- 준영속 영속성으로 관리되다가 분리된 상태
- 삭제 객체를 삭제한 상태

팁

라이브러리 버전 선택 꿀팁

스프링 부트 Reference 에서 버전확인후 사용