Spring Boot

JPA(1)

신온유
정보보호학과
2017111683

JPA 소개

ORM이란?

- Object-Relational Mapping (객체와 관계형데이터베이스 매핑, 객체와 DB의 테이블이 매핑을 이루는 것)
- 객체가 테이블이 되도록 매핑 시켜주는 프레임워크 이다.
- 프로그램의 복잡도를 줄이고 자바 객체와 쿼리를 분리할 수 있으며 트랜잭션 처리나 기타 데이터베이스 관련 작업들을 좀 더 편리하게 처리할 수 있는 방법
- SQL Query가 아닌 직관적인 코드(메서드)로서 데이터를 조작할 수 있다.

ex) 기존쿼리 : SELECT * FROM MEMBER; 이를 ORM을 사용하면 Member테이블과 매핑된 객체가 member라고 할 때, member.findAll()이라는 메서드 호출로 데이터 조회가 가능하다.

JPA 소개

1. 관계형 데이터 베이스가 중요시 되는 상황에서 객체를 관리하는 것의 중요성이 부각 되었음.

2. 관계형 데이터베이스가 SQL 만 인식할 수 있기 때문에 SQL 중심이 되었고 그에 따라 애플리케이션 코드보다 SQL 코드로 가득하게 되었음.

3. SQL 사용을 위해서는 각 테이블마다 기본적인 CRUD(Create,Read,Update,Delete)코드를 반복적으로 생성해야 하는 문제가 발생함.

4. 어떻게 데이터를 저장할 지에 초점을 맞춘 관계형 데이터 베이스와 달리 객체 지향 프로그래밍 언어는 기능과 속성을 한 곳에서 관리하는 것에 초점을 맞춤.

5. 즉,객체지향 < -/- > 관계형데이터베이스 두가지는 서로를 표현할 수 없다.

6. 이러한 객체와 데이터베이스 패러다임 불일치를 해결하기 위해 나온 것이 바로 JPA 이다.

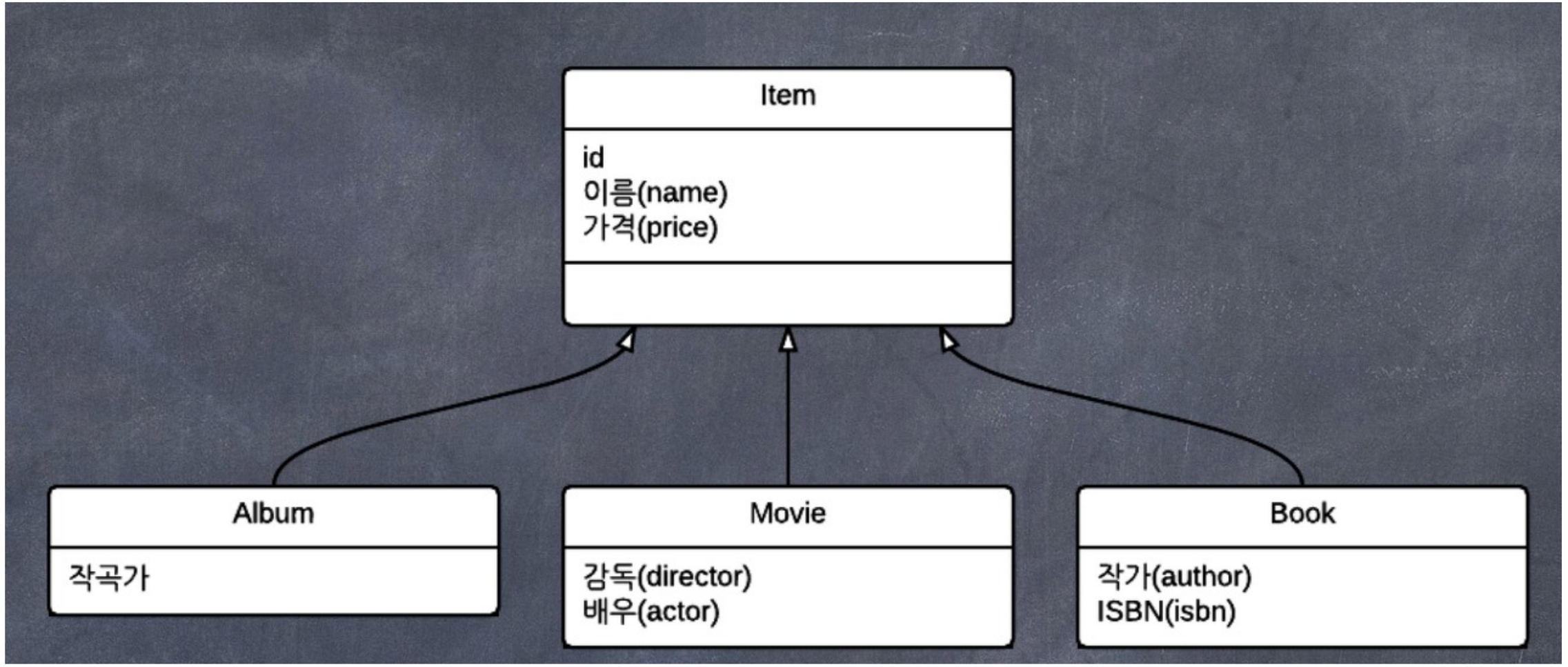
JPA 소개

JPA란?

- Java Persistence API (자바 ORM 기술에 대한 API 표준 명세)
- 한마디로 ORM을 사용하기 위한 인터페이스를 모아둔 것 이라고 볼 수 있다.
- 자바 어플리케이션에서 관계형 데이터베이스를 사용하는 방식을 정의한 인터페이스이다.
- ORM에 대한 자바 API 규격이며 Hibernate, OpenJPA 등이 JPA를 구현한 구현체 이다. (ORM을 사용하기 위한 인터페이스를 모아둔 것)
- Hibernate 이외에도 EcipseLink, DataNucleus, OpenJPA, TopLink 등이 있습니다.

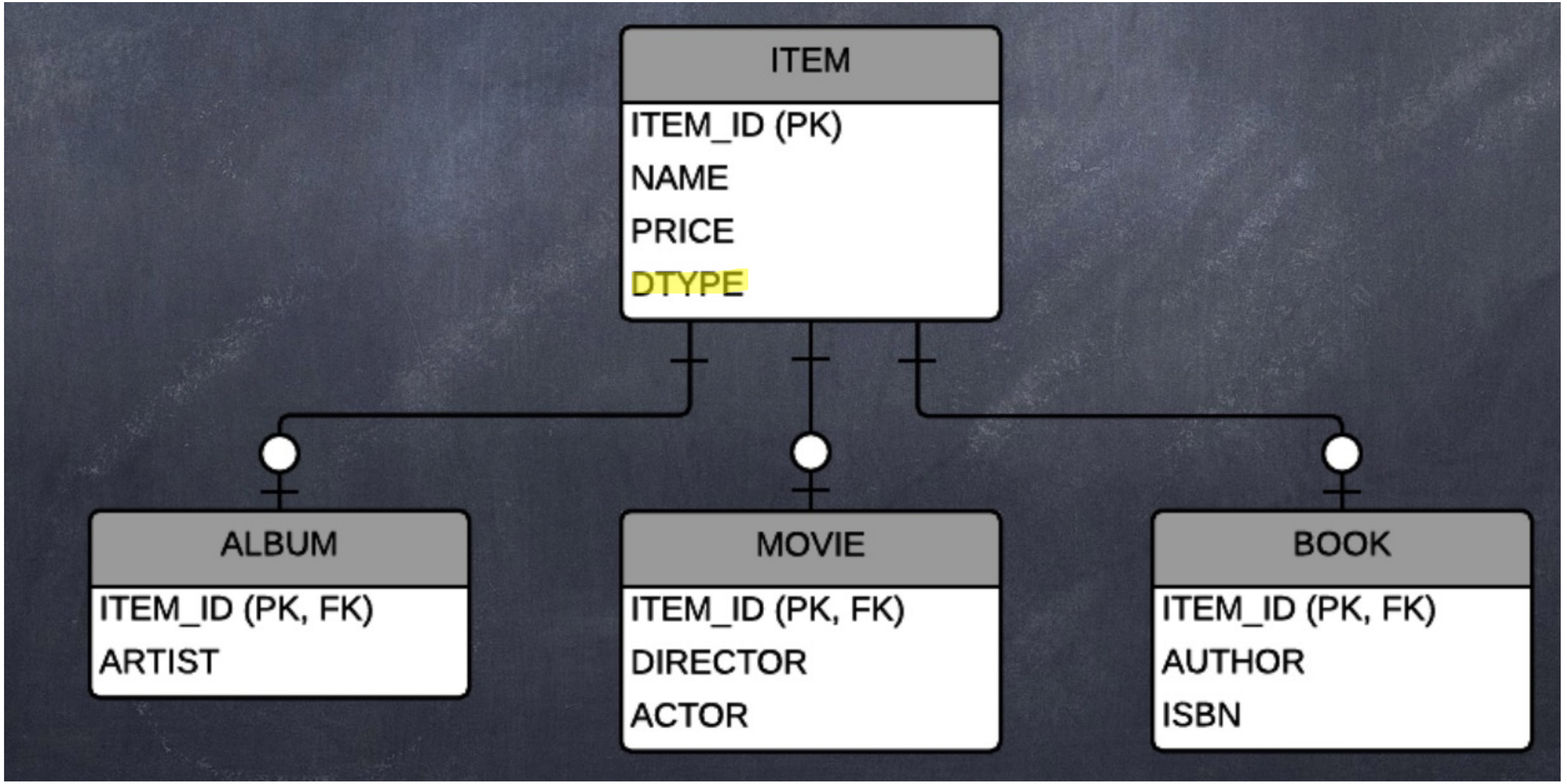
※결국 인터페이스이기 때문에 JPA를 사용하기 위해서는 JPA를 구현한 Hibernate, EclipseLink, DataNucleus 같은 ORM 프레임워크를 사용해야 한다.

JPA 소개



객체는 이러한 상속기능을 지닌다

JPA 소개



Item 테이블의 DTYPE컬럼을 사용하면 어떤 자식 테이블과 관계가 있는지 정의할 수 있다.

JPA 소개

- 이러한 경우 자식 데이터만을 위한 SQL문을 추가로 작성하는 등의 비용이 발생한다.
- 예) 부모 객체에서 부모 데이터만 꺼내 ITEM용 SQL을 작성하고 자식 객체에서는 자식 데이터만 꺼네 ALBUM용 SQL을 작성해야한다. 작성해야할 코드도 많고 자식타입에따라 DTYPE도 추가로 저장해야함.

만약, 데이터가 자바 컬렉션에 보관된다면 부모, 자식이나 타입에 대한 고민없이 컬렉션을 조회하면된다.

```
list.add(album);
list.add(movie);

Album albun = list.get(albumId);
```

예제

```java
//실제 DB테이블과 매칭될 클래스
//DB 데이터에 작업 할 경우에 실제 쿼리를 날리기보다는 엔티티 클래스의 수정을 통해 작업을 함
public class Posts {
    @Id
    //테이블의 PK 필드를 나타냄
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    //PK의 생성규칙을 나타낸다
    private Long id;

    @Column(length = 500,nullable = false)
    //테이블의 칼럼을 나타냄 선언하지 않더라도 해당 클래스의 필드는 모두 칼럼이 된다
    //기본값 외의 추가로 변경이 필요하면 사용함
    private String title;

    @Column(columnDefinition = "TEXT",nullable = false)

    private String content;
    private String author;

    @Builder
    //해당 클래스의 빌더 패턴 클래스를 생성
    public  Posts(String title,String content,String author){
        this.title = title;
        this.content = content;
        this.author = author;
    }
}
```

예제

```java
import org.springframework.data.jpa.repository.JpaRepository;
//DB Layer 접근자
//데이터에 접근을 위한 객체 생성
public interface PostsRepository extends JpaRepository<Posts,Long>{

}
```

예제

```java
@Test
public void 게시글저장_불러오기(){
    String title = "테스트 게시글";
    String content = "테스트 본문";

    postsRepository.save(Posts.builder()
            .title(title)
            .content(content)
            .author("onyoo@gmail.com")
            .build()
    );
    //테이블 posts에 insert/update 쿼리를 실행함
    //id값이 있다면 update 없다면 insert 쿼리가 실행 된다

    //when
    List<Posts> postsList = postsRepository.findAll();
    //posts 테이블에 있는 모든 데이터를 조회해오는 메소드

    //then
    Posts posts = postsList.get(0);
    assertThat(posts.getTitle()).isEqualTo(title);
    assertThat(posts.getContent()).isEqualTo(content);
```

예제

```
Hibernate: insert into posts (id, author, content, title) values (null, ?, ?, ?)
2021-11-11 12:45:57.051  INFO 1311 --- [    Test worker] o.h.h.i.QueryTranslatorFactoryInitiator  : HHH000397: Using ASTQueryTranslatorFactory
Hibernate: select posts0_.id as id1_0_, posts0_.author as author2_0_, posts0_.content as content3_0_, posts0_.title as title4_0_ from posts posts0_
Hibernate: select posts0_.id as id1_0_, posts0_.author as author2_0_, posts0_.content as content3_0_, posts0_.title as title4_0_ from posts posts0_
```

데이터가 insert 되는 모습

다음발표

JPA를 이용하여
등록/수정/조회 API 작성하기

감사합니다