

스프링 부트 3 백엔드 개발자 되기 (2판)

JPA+OAuth2+JWT+AWS와 배우는 스프링 부트 3
Java 백엔드 입문자를 위한 풀 패키지

05 데이터베이스 조작이 편해지는 ORM

5.1 데이터베이스란?

- 데이터를 효율적으로 보관하고 꺼낼 수 있는 곳
- 데이터베이스 관리자 = DBMS
 - 흔히 MySQL, 오라클과 같이 부르는 것은 데이터베이스가 아니라 데이터베이스 관리자
- 관계형 DBMS = RDBMS
 - 흔히 떠올리는 표 형태의 데이터베이스

회원 테이블		
ID	이메일	나이
1	a@test.com	10
2	b@test.com	20
3	c@test.com	30

- 이 책에서 사용하는 데이터베이스
 - H2, MySQL

5.1 데이터베이스란?

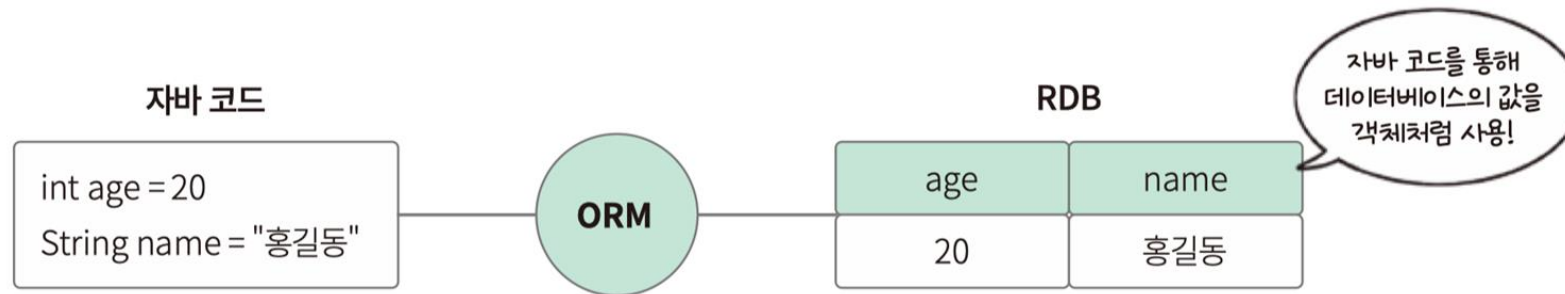
- 꼭 알아야 할 데이터베이스 용어

- 테이블 : 데이터베이스에서 데이터를 구성하기 위한 가장 기본적인 단위
- 행 = row = 테이블의 가로로 배열된 데이터의 집합
- 열 = column = 행의 속성을 결정, 무결성 보장
- 기본키 = 행을 구분할 수 있는 식별자(열)
- 쿼리 = 데이터베이스에서 데이터를 처리할 때 쓰는 명령문

회원 테이블		
ID	이메일	나이
1	a@test.com	10
2	b@test.com	20
3	c@test.com	30


5.2 ORM이란?

- 자바의 객체와 데이터베이스를 연결하는 프로그래밍 기법
 - ORM이 없다면 데이터베이스를 조작하기 위한 언어(SQL)를 새로 배워야 함
 - ORM이 있으면 스프링 부트에서 사용하는 자바 언어로 데이터베이스를 조작할 수 있음
 - 개발자의 러닝커브 down



5.2 ORM이란?(cont.)

- ORM의 장점과 단점

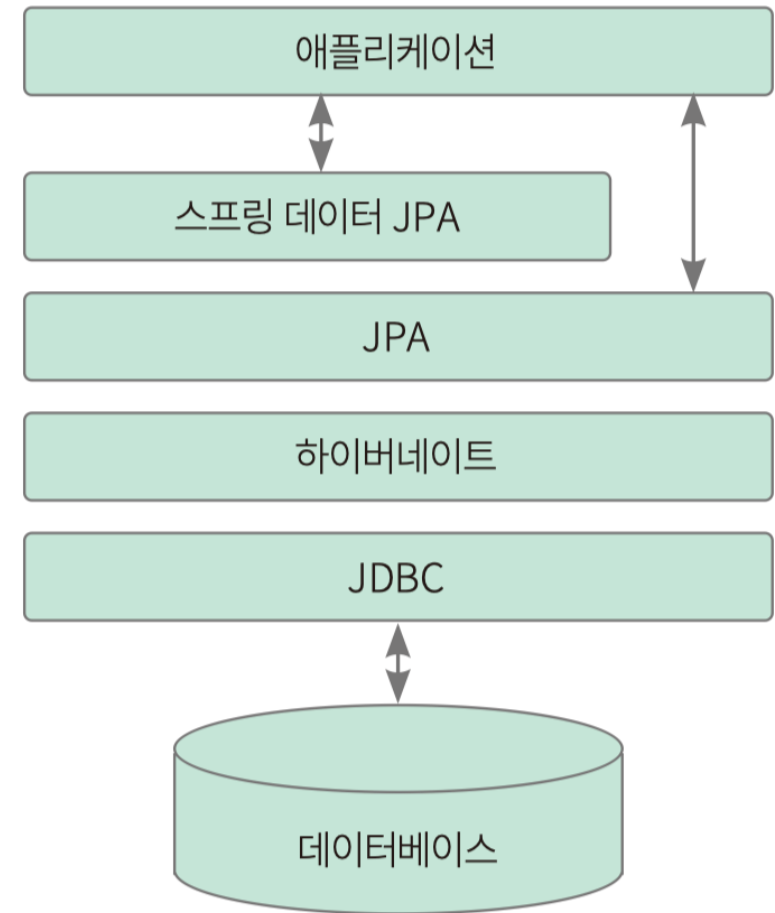


ORM의 장점과 단점

- 장점 1. SQL을 직접 작성하지 않고 사용하는 언어로 데이터베이스에 접근할 수 있습니다.
- 장점 2. 객체지향적으로 코드를 작성할 수 있기 때문에 비즈니스로직에만 집중할 수 있습니다.
- 장점 3. 데이터베이스 시스템이 추상화되어 있기 때문에 MySQL에서 PostgreSQL로 전환한다고 해도 추가로 드는 작업이 거의 없습니다. 즉, 데이터베이스 시스템에 대한 종속성이 줄어듭니다.
- 장점 4. 매핑하는 정보가 명확하기 때문에 ERD에 대한 의존도를 낮출 수 있고 유지보수할 때 유리합니다.
- 단점 1. 프로젝트의 복잡성이 커질수록 사용 난이도도 올라갑니다.
- 단점 2. 복잡하고 무거운 쿼리는 ORM으로 해결이 불가능한 경우가 있습니다.

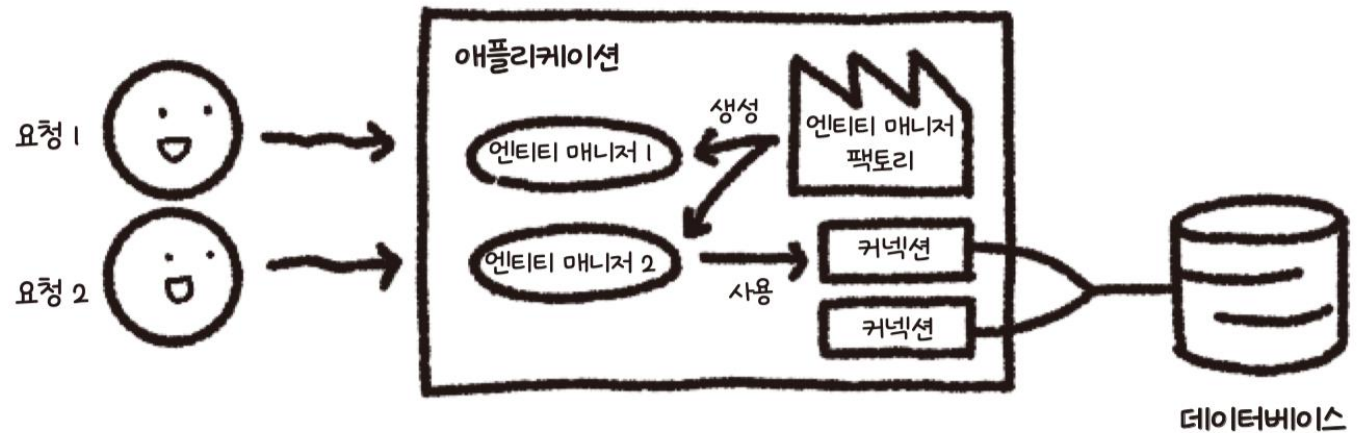
5.3 JPA와 하이버네이트

- ORM의 종류는 매우 다양
- 자바는 JPA를 표준으로 사용
 - JPA = java persistence API
 - 자바에서 관계형 데이터베이스를 사용하는 방식을 정의한 인터페이스
 - 실제 사용을 위해서는 추가로 ORM 프레임워크를 선택해야 함
- ORM 프레임워크, 하이버네이트
 - JPA 인터페이스를 구현한 구현체 = 이것을 실제 코드에 사용



5.3 JPA와 하이버네이트(cont.)

- 엔티티
 - 데이터베이스의 테이블과 매핑되는 객체
- 엔티티 매니저
 - 엔티티를 관리하는 역할
 - 엔티티 매니저는 엔티티 매니저 팩토리에서 생성
 - 엔티티 매니저는 Spring Data JPA에서 관리하므로 직접 생성하거나 관리할 필요가 없음



5.3 JPA와 하이버네이트(cont.)

- 영속성 컨텍스트
 - 엔티티를 관리하는 가상의 공간
 - 데이터베이스에서 효과적으로 데이터를 가져올 수 있는 이유
- 1차 캐시
 - 데이터베이스를 직접 거치지 않고 빠르게 데이터를 조회할 수 있게 해줌
- 쓰기 지연
 - 트랜잭션을 커밋하기 전까지 쿼리를 모았다가 보내 실행하는 것
- 변경 감지
 - 1차 캐시에 저장되어 있는 엔티티와 현재 엔티티 비교하여 변경 사항을 감지
- 지연 로딩
 - 쿼리로 요청한 데이터를 즉시 로딩하지 않음
 - 자원 효율을 고려하여 데이터 조회

5.3 JPA와 하이버네이트(cont.)

- 엔티티의 상태에는 총 4가지가 있음
- 비영속 상태
 - 엔티티 매니저가 엔티티를 관리하지 않는 상태
- 관리 상태
 - 엔티티가 관리되는 상태
- 분리 상태
 - 엔티티가 분리된 상태
- 삭제 상태
 - 엔티티가 삭제된 상태

5.4 스프링 데이터와 스프링 데이터 JPA

- 리포지터리 역할을 하는 인터페이스를 만들어 간단히 CRUD 작업 가능

- MemberRepository.java

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

MemberRepository.java

- MemberService.java

- 데이터 생성
- 데이터 조회
- 데이터 삭제

```
@Service
public class MemberService {

    @Autowired
    MemberRepository memberRepository;

    public void test() {
        // ❶ 생성(Create)
        memberRepository.save(new Member(1L, "A"));
    }
}
```

MemberService.java

...

5.5 예제 코드 살펴보기

- 117쪽부터 참고

```
Member.java
@Entity // ❶ 엔티티로 지정
@Getter
@NoArgsConstructor(access = AccessLevel.PROTECTED) // ❷ 기본 생성자
@AllArgsConstructor

public class Member {

    @Id // ❸ id 필드를 기본키로 지정
    @GeneratedValue(strategy = GenerationType.IDENTITY) // ❹ 기본키 자동으로 1씩 증가
    @Column(name = "id", updatable = false)
    private Long id;

    @Column(name = "name", nullable = false) // ❺ name이라는 not null 컬럼과 매핑
    private String name;
}
```

5.5 예제 코드 살펴보기(cont.)

- 전체 관계 그림 참고(사용자, 리포지토리, DB, 엔티티)

