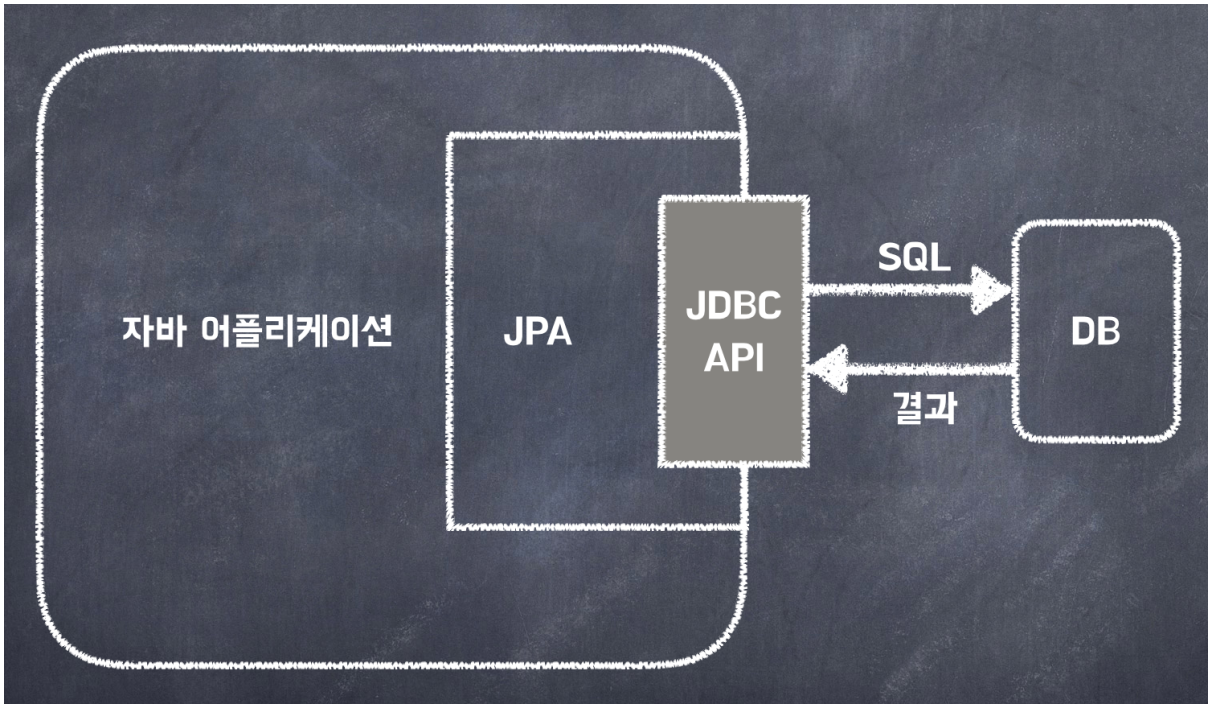
JPA에 대하여

2022.02.06

Java Persistence API

자바 표준 ORM



JPA는 어플리케이션과 JDBC사이에서 동작한다.

ORM

Object Relational Mapping

객체와 관계형 데이터베이스를 매핑한다.

ORM 프레임워크는 객체와 테이블을 매핑하여 패러다임의 불일치를 해결한다.

객체를 ORM프레임워크에 저장하면, ORM프레임워크가 SQL을 작성해서 객체를 DB에 관리

- Hibernate
- EclipseLink
- DataNucleus

대중적인 언어 대부분 ORM기술이 존재한다.

패러다임의 불일치:

객체지향 프로그래밍 언어는 메시지를 기반으로 **기능과 속성**을 한 곳에서 관리하는 기술,

관계형 데이터베이스는 어떻게 **데이터를 저장**할지에 초점이 맞춰진 기술

: 객체를 데이터베이스에 저장하려고 하니 여러 문제가 발생한다.

JPA란?

EJB ——————— Hibernate ——————— JPA

(Entity Bean) (ORM framework)

EJB 3.0에서 하이버네이트 기반으로 새로운 자바 ORM기술 표준이 만들어졌다.

JPA는 인터페이스로서 자바 표준명세서이다. 인터페이스인 JPA를 사용하기 위해서는 구현체가 필요하다.

Spring Data JPA

구현체를 좀 더 쉽게 사용하고자 추상화시킨 모듈

JPA ← Hibernate ← Spring Data JPA

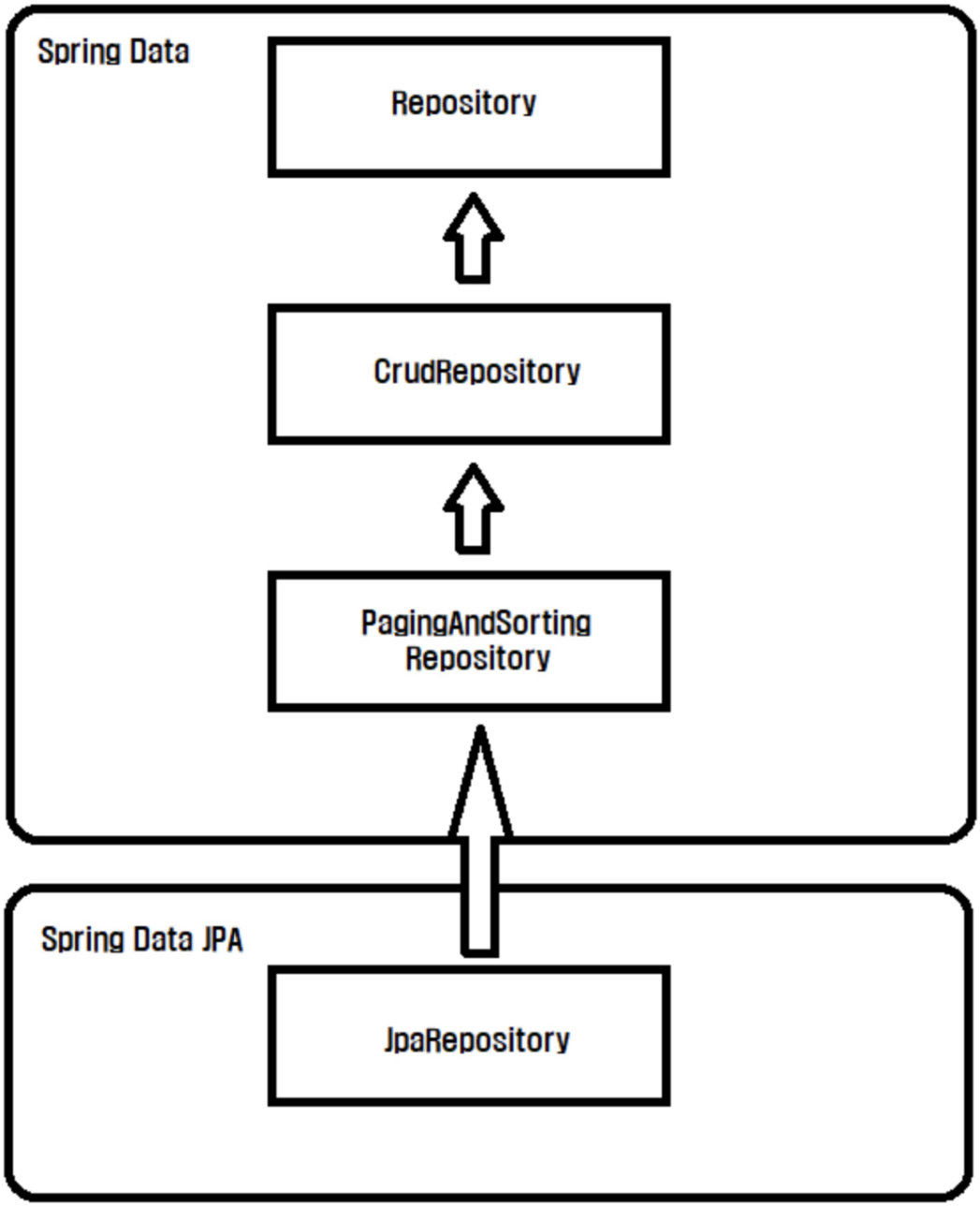
등장이유:

- 구현체 교체의 용이성 : Hibernate 외에 다른 구현체로 쉽게 교체하기 위함
- 저장소 교체의 용이성 : 관계형 데이터베이스 외에 다른 저장소로 쉽게 교체하기 위함

Spring Data의 하위 프로젝트들은 기본적인 CRUD의 인터페이스가 같다.

> save(), findAll, findOne()등을 인터페이스로 갖고 있으므로 저장소가 교체되어도 기본적인 기능은 변경할 것이 없다.

> Spring Data
>
> Spring Data is an umbrella project consisting of independent projects with, in principle, different release cadences. To manage the portfolio, a BOM (Bill of Materials - see this example) is published with a curated set of dependencies on the individual project. The release trains have names, not versions, to
>
> https://spring.io/projects/spring-data



프로젝트에 Spring Data JPA 적용하기

build.gradle

```
dependencies{
    implementation 'org.springframework.boot:spring-boot-starter-jdbc'
    implementation 'org.springframework.boot:spring-boot-starter-data-jpa'
    implementation 'org.springframework.boot:spring-boot-starter-web'
}
```

spring-boot-starter-data-jpa

- 스프링 부트용 Spring Data JPA 추상화 라이브러리
- 스프링 부트 버전에 맞춰 자동으로 JPA관련 라이브러리들의 버전을 관리

JPA 사용 시 가장 중요한 일은 엔티티와 테이블을 정확히 매핑하는 것이다

- 객체와 테이블 매핑: @Entity, @Table
- 기본 키 매핑: @Id
- 필드와 컬럼 매핑: @Column
- 연관관계 매핑: @ManyToOne, @JoinColumn

@Entity : JPA 어노테이션

- 테이블과 링크될 클래스임을 나타낸다
- 기본 생성자는 필수(파라미터가 없는 public 또는 protected 생성자)
- final 클래스, enum, interface, inner 클래스에는 사용할 수 없다
- 저장 필드에 final을 사용하면 안된다
- 기본값으로 클래스의 카멜케이스 이름을 언더스코어 네이밍으로 테이블 이름을 매칭한다

  ex) SpringBootTest.java → spring_boot_test 테이블

JpaRepository

- @Repository 추가할 필요 없다
- MyBatis등에서 DAO라고 불리는 DB Layer 접근자는 JPA에선 Repository라 하고 인터페이스로 생성한다.

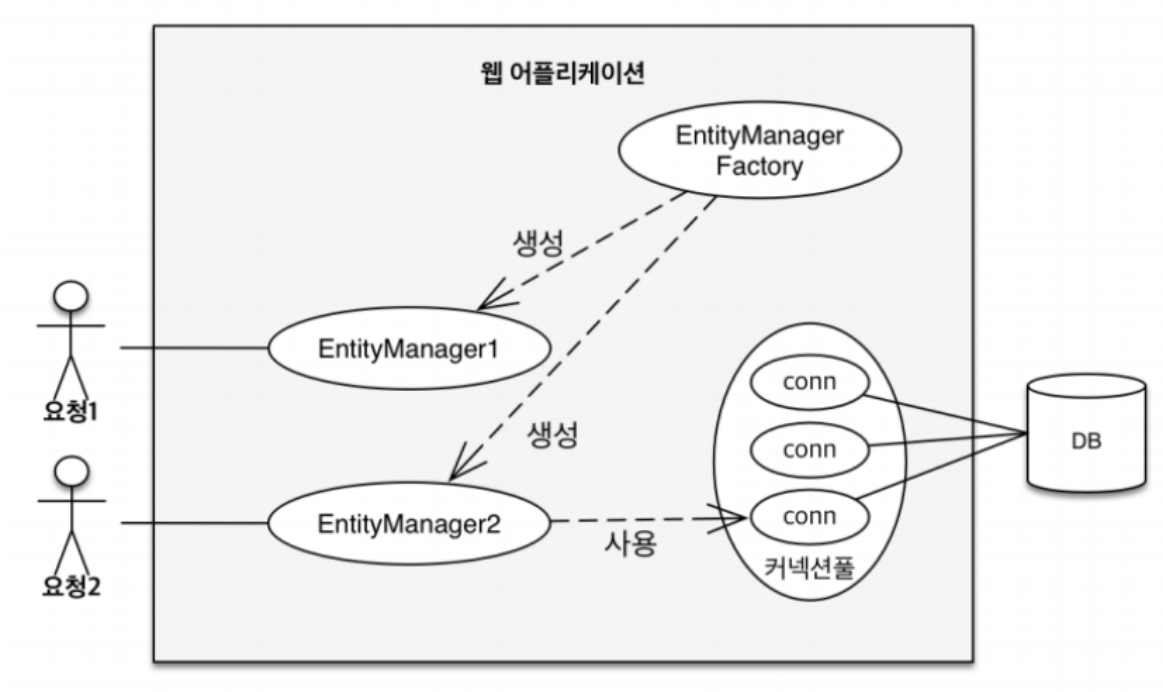
- 인터페이스 생성 후 extends `JpaRepository<Entity 클래스, PK 타입>` 상속하면 기본적인 CRUD메소드가 자동으로 생성된다.

💡 Entity 클래스와 기본 Entity Repository는 함께 위치해야 한다.

Entity class는 기본 Repository없이는 제대로 역할을 할 수가 없다.

영속성 Persistence

▼ Spring Data JPA를 사용하면 기본적으로 Entity Manger는 활성화되어있는 상태이다.

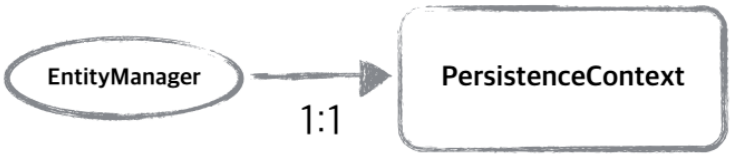


Entity Manger로 엔티티를 저장하거나 조회하면 em은 영속성 컨텍스트에 엔티티를 보관하고 관리한다.

영속성 컨텍스트(persistence context) = 엔티티를 영구 저장하는 환경

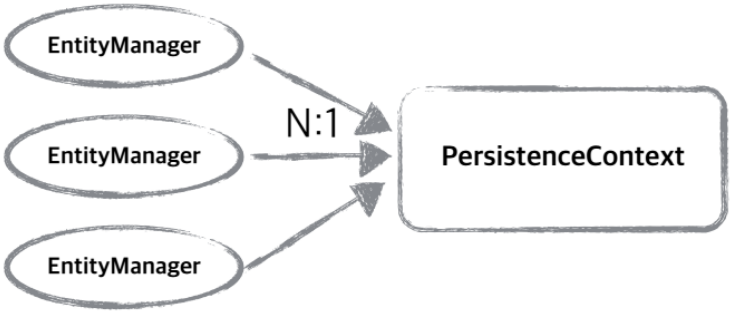
J2SE 환경

엔티티 매니저와 영속성 컨텍스트가 1:1



J2EE, 스프링 프레임워크 같은 컨테이너 환경

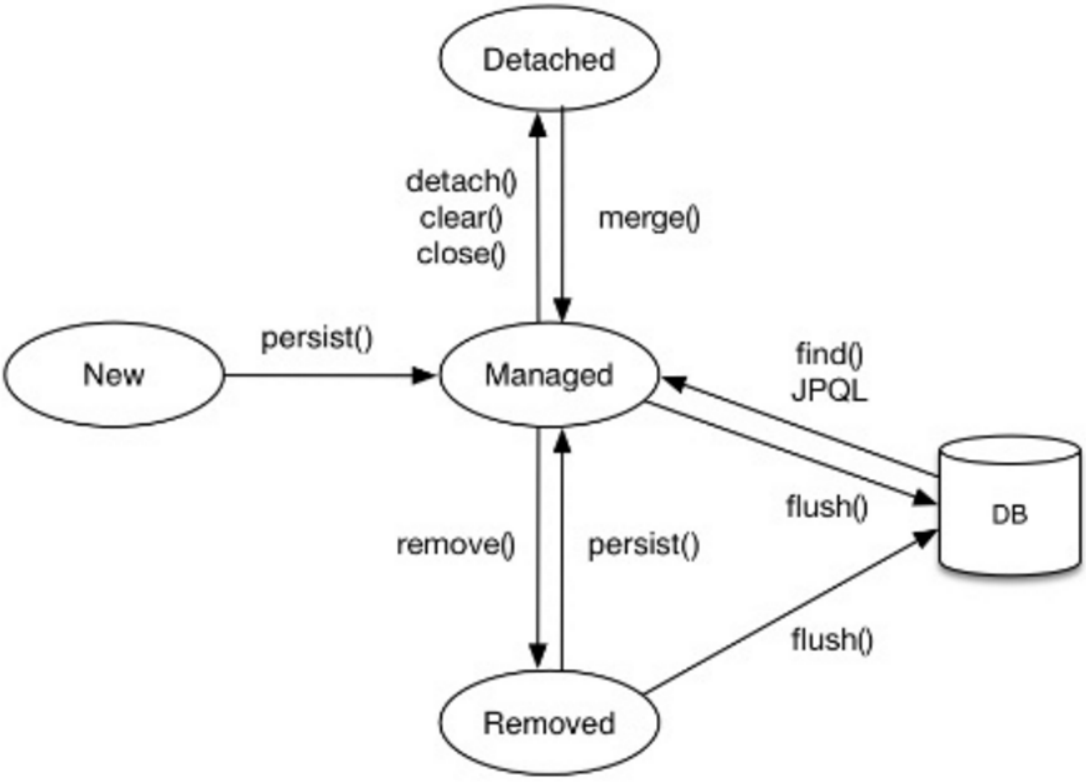
엔티티 매니저와 영속성 컨텍스트가 N:1



영속성 컨텍스트는 애플리케이션과 데이터베이스 사이에서 객체를 보관하는 가상의 데이터베이스 같은 역할을 한다.

영속성 컨텍스트가 엔티티를 관리하면 다음과 같은 장점이 있다.

- 1차 캐시,
- 동일성 보장,
- 트랜잭션을 지원하는 쓰기 지연,
- 변경 감지



엔티티 생명주기: 비영속/영속/준영속/삭제

참고자료

Spring Data JPA

Level up your Java code and explore what Spring can do for you.

https://spring.io/projects/spring-data-jpa

https://ultrakain.gitbooks.io/jpa/content/