Google Developer Student Clubs

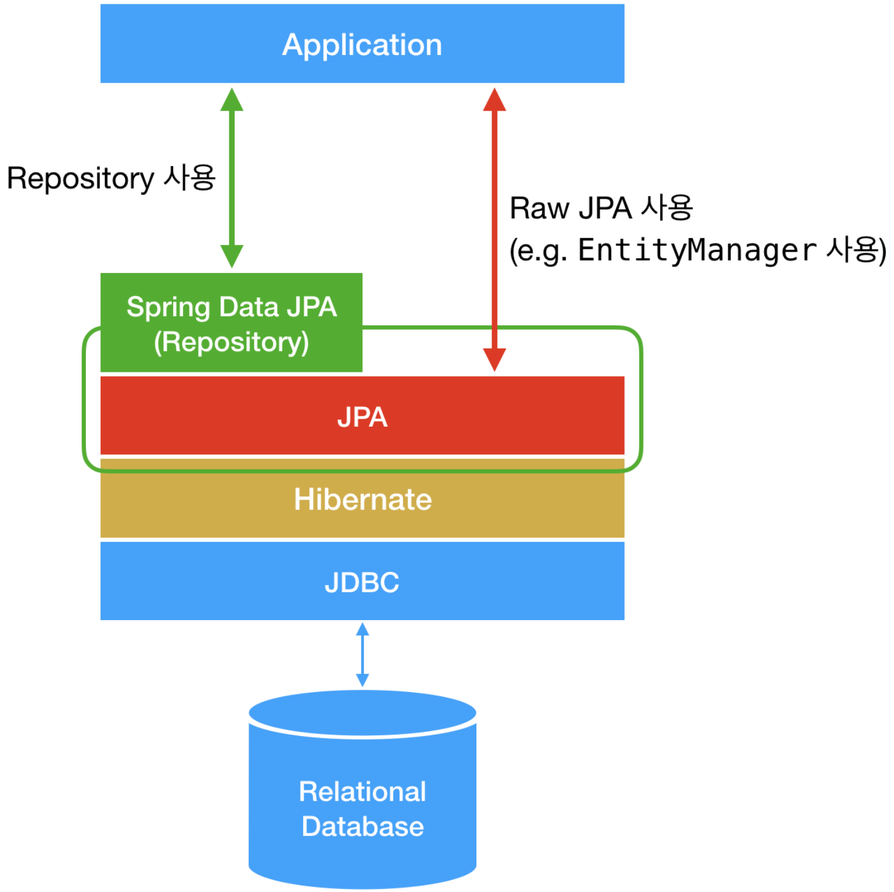
Backend 7th Study

Google Developer Student Clubs

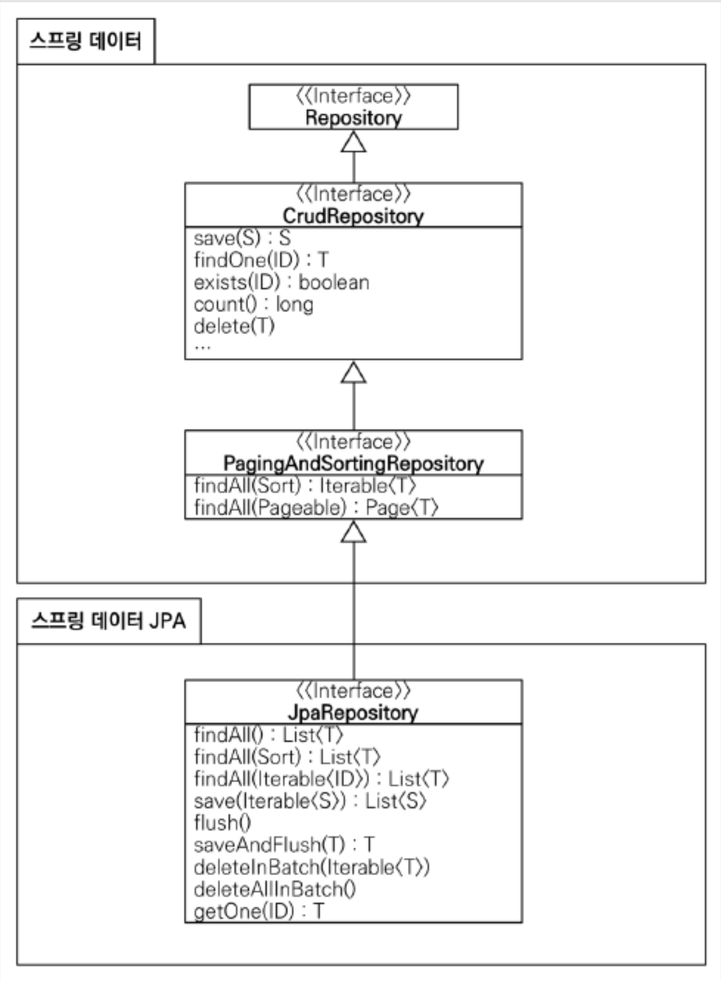
Contents

우리가 앞으로 사용하게 될 것.
Data JPA InterFace

Data JPA


Application
Repository 사용
Raw JPA 사용
(e.g. EntityManager 사용)
Spring Data JPA
(Repository)
JPA
Hibernate
JDBC
Relational
Database

Data JPA


스프링 데이터
<<Interface>>
Repository
<<Interface>>
CrudRepository
save(S) : S
findOne(ID) : T
exists(ID) : boolean
count() : long
delete(T)
...
<<Interface>>
PagingAndSortingRepository
findAll(Sort) : Iterable<T>
findAll(Pageable) : Page<T>
스프링 데이터 JPA
<<Interface>>
JpaRepository
findAll() : List<T>
findAll(Sort) : List<T>
findAll(Iterable<ID>) : List<T>
save(Iterable<S>) : List<S>
flush()
saveAndFlush(T) : T
deleteInBatch(Iterable<T>)
deleteAllInBatch()
getOne(ID) : T

Data JPA

주요 메서드

- `save(S)` : 새로운 엔티티는 저장하고 이미 있는 엔티티는 병합한다.
- `delete(T)` : 엔티티 하나를 삭제한다. 내부에서 `EntityManager.remove()` 호출
- `findById(ID)` : 엔티티 하나를 조회한다. 내부에서 `EntityManager.find()` 호출
- `getOne(ID)` : 엔티티를 프록시로 조회한다. 내부에서 `EntityManager.getReference()` 호출
- `findAll(…)` : 모든 엔티티를 조회한다. 정렬(`Sort`)이나 페이징(`Pageable`) 조건을 파라미터로 제공할 수 있다.

Data JPA

```java
@NoRepositoryBean
public interface JpaRepository<T, ID> extends PagingAndSortingRepository<T, ID>, QueryByExampleExecutor<T> {

    /*
     * (non-Javadoc)
     * @see org.springframework.data.repository.CrudRepository#findAll()
     */
    1 implementation
    @Override
    List<T> findAll();

    /*
     * (non-Javadoc)
     * @see org.springframework.data.repository.PagingAndSortingRepository#findAll(org.springframework.data.domain.Sort)
     */
    1 implementation
    @Override
    List<T> findAll(Sort sort);

    /*
     * (non-Javadoc)
     * @see org.springframework.data.repository.CrudRepository#findAll(java.lang.Iterable)
     */
    1 usage   1 implementation
    @Override
    List<T> findAllById(Iterable<ID> ids);
```

Data JPA

```java
package com.gdsc.homework.thirdstudy.domain.member;

import org.springframework.data.jpa.repository.JpaRepository;

import java.util.Optional;

2 usages  LeeJejune
public interface MemberRepository extends JpaRepository<Member, Long> {
    1 usage  LeeJejune
    Optional<Member> findMemberByName(String name);
}
```

Data JPA

https://docs.spring.io/spring-data/jpa/docs/current/reference/html/#repositories.query-methods.query-creation

- COUNT: count...By 반환타입 `long`
- EXISTS: exists...By 반환타입 `boolean`
- 삭제: delete...By, remove...By 반환타입 `long`
- DISTINCT: findDistinct, findMemberDistinctBy
- LIMIT: findFirst3, findFirst, findTop, findTop3

Data JPA

```
@Query("select m from Member m where m.name= :username and m.age = :age")
List<Member> findUser(@Param("username") String username, @Param("age") int age);
```

Data JPA

```
@Query("select m from Member m where m.name= :username and m.age = :age")
List<Member> findUser(@Param("username") String username, @Param("age") int age);
```

Data JPA

Page 인터페이스

```
public interface Page<T> extends Slice<T> {
    int getTotalPages();     //전체 페이지 수
    long getTotalElements(); //전체 데이터 수
    <U> Page<U> map(Function<? super T, ? extends U> converter); //변환기
}
```

Slice 인터페이스

```
public interface Slice<T> extends Streamable<T> {
  int getNumber();            //현재 페이지
  int getSize();              //페이지 크기
  int getNumberOfElements();  //현재 페이지에 나올 데이터 수
  List<T> getContent();       //조회된 데이터
  boolean hasContent();       //조회된 데이터 존재 여부
  Sort getSort();             //정렬 정보
  boolean isFirst();          //현재 페이지가 첫 페이지 인지 여부
  boolean isLast();           //현재 페이지가 마지막 페이지 인지 여부
  boolean hasNext();          //다음 페이지 여부
  boolean hasPrevious();      //이전 페이지 여부
  Pageable getPageable();     //페이지 요청 정보
  Pageable nextPageable();    //다음 페이지 객체
  Pageable previousPageable();//이전 페이지 객체
  <U> Slice<U> map(Function<? super T, ? extends U> converter); //변환기
}
```

Data JPA

사용자 정의 인터페이스 구현 클래스

```
@RequiredArgsConstructor
public class MemberRepositoryImpl implements MemberRepositoryCustom {

    private final EntityManager em;

    @Override
    public List<Member> findMemberCustom() {
        return em.createQuery("select m from Member m")
                .getResultList();
    }
}
```

사용자 정의 인터페이스 상속

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

사용자 정의 메서드 호출 코드

```
List<Member> result = memberRepository.findMemberCustom();
```

사용자 정의 구현 클래스

- 규칙: 리포지토리 인터페이스 이름 + `Impl`
- 스프링 데이터 JPA가 인식해서 스프링 빈으로 등록

사실 정말 중요한건...

Google Developer Student Clubs

시험 잘 보세요!