6. 스프링 데이터 JPA 분석

#1.인강/jpa활용편/datajpa/강의

- /스프링 데이터 JPA 구현체 분석
- /새로운 엔티티를 구별하는 방법

스프링 데이터 JPA 구현체 분석

- 스프링 데이터 JPA가 제공하는 공통 인터페이스의 구현체
- `org.springframework.data.jpa.repository.support.SimpleJpaRepository`

리스트 12.31 SimpleJpaRepository

```
@Repository
@Transactional(readOnly = true)
public class SimpleJpaRepository<T, ID> ...{

    @Transactional
    public <S extends T> S save(S entity) {

        if (entityInformation.isNew(entity)) {
            em.persist(entity);
            return entity;
        } else {
            return em.merge(entity);
        }
    }
    ...
}
```

- `@Repository` 적용: JPA 예외를 스프링이 추상화한 예외로 변환
- `@Transactional` 트랜잭션 적용
  - JPA의 모든 변경은 트랜잭션 안에서 동작
  - 스프링 데이터 JPA는 변경(등록, 수정, 삭제) 메서드를 트랜잭션 처리
  - 서비스 계층에서 트랜잭션을 시작하지 않으면 리파지토리에서 트랜잭션 시작
  - 서비스 계층에서 트랜잭션을 시작하면 리파지토리는 해당 트랜잭션을 전파 받아서 사용
  - 그래서 스프링 데이터 JPA를 사용할 때 트랜잭션이 없어도 데이터 등록, 변경이 가능했음(사실은 트랜잭션이 리포지토리 계층에 걸려있는 것임)

- `@Transactional(readOnly = true)`
  - 데이터를 단순히 조회만 하고 변경하지 않는 트랜잭션에서 `readOnly = true` 옵션을 사용하면 플러시를 생략해서 약간의 성능 향상을 얻을 수 있음
  - 자세한 내용은 JPA 책 15.4.2 읽기 전용 쿼리의 성능 최적화 참고

매우 중요!!!

- `save()` *메서드*
  - 새로운 엔티티면 저장(`persist`)
  - 새로운 엔티티가 아니면 병합(`merge`)

새로운 엔티티를 구별하는 방법

매우 중요!!!

- `save()` *메서드*
  - 새로운 엔티티면 저장(`persist`)
  - 새로운 엔티티가 아니면 병합(`merge`)

- 새로운 엔티티를 판단하는 기본 전략
  - 식별자가 객체일 때 `null` 로 판단
  - 식별자가 자바 기본 타입일 때 `0` 으로 판단
  - `Persistable` 인터페이스를 구현해서 판단 로직 변경 가능

*`Persistable` *

```
package org.springframework.data.domain;

public interface Persistable<ID> {
    ID getId();
    boolean isNew();
}
```

> 참고: JPA 식별자 생성 전략이 `@GenerateValue` 면 `save()` 호출 시점에 식별자가 없으므로 새로운 엔티티로 인식해서 정상 동작한다. 그런데 JPA 식별자 생성 전략이 `@Id` 만 사용해서 직접 할당이면 이미 식별자 값이 있는 상태로 `save()` 를 호출한다. 따라서 이 경우 `merge()` 가 호출된다. `merge()` 는 우선 DB를 호출해서 값을 확인하고, DB에 값이 없으면 새로운 엔티티로 인지하므로 매우 비효율 적이다. 따라서 `Persistable` 를 사용해서 새로운 엔티티 확인 여부를 직접 구현하게는 효과적이다.
> 참고로 등록시간(`@CreatedDate`)을 조합해서 사용하면 이 필드로 새로운 엔티티 여부를 편리하게 확인할 수 있다. (@CreatedDate에 값이 없으면 새로운 엔티티로 판단)

Persistable 구현

```
package study.datajpa.entity;

import lombok.AccessLevel;
import lombok.NoArgsConstructor;
import org.springframework.data.annotation.CreatedDate;
import org.springframework.data.domain.Persistable;
import org.springframework.data.jpa.domain.support.AuditingEntityListener;

import javax.persistence.Entity;
import javax.persistence.EntityListeners;
import javax.persistence.Id;
import java.time.LocalDateTime;

@Entity
@EntityListeners(AuditingEntityListener.class)
@NoArgsConstructor(access = AccessLevel.PROTECTED)
public class Item implements Persistable<String> {

    @Id
    private String id;

    @CreatedDate
    private LocalDateTime createdDate;

    public Item(String id) {
        this.id = id;
    }

    @Override
    public String getId() {
        return id;
    }
```

```
    @Override
    public boolean isNew() {
        return createdDate == null;
    }
}
```