5. 상품 도메인 개발

#1.인강/jpa활용편/활용편1/강의

- /상품 엔티티 개발(비즈니스 로직 추가)
- /상품 리포지토리 개발
- /상품 서비스 개발

구현 기능

- 상품 등록
- 상품 목록 조회
- 상품 수정

순서

- 상품 엔티티 개발(비즈니스 로직 추가)
- 상품 리포지토리 개발
- 상품 서비스 개발
- 상품 기능 테스트

상품 엔티티 개발(비즈니스 로직 추가)

상품 엔티티 코드

```
package jpabook.jpashop.domain.item;

import jpabook.jpashop.exception.NotEnoughStockException;
import lombok.Getter;
import lombok.Setter;

import jpabook.jpashop.domain.Category;
import javax.persistence.*;
import java.util.ArrayList;
import java.util.List;

@Entity
@Inheritance(strategy = InheritanceType.SINGLE_TABLE)
@DiscriminatorColumn(name = "dtype")
@Getter @Setter
public abstract class Item {
```

```
    @Id @GeneratedValue
    @Column(name = "item_id")
    private Long id;

    private String name;
    private int price;
    private int stockQuantity;

    @ManyToMany(mappedBy = "items")
    private List<Category> categories = new ArrayList<Category>();

    //==비즈니스 로직==//
    public void addStock(int quantity) {
        this.stockQuantity += quantity;
    }

    public void removeStock(int quantity) {
        int restStock = this.stockQuantity - quantity;
        if (restStock < 0) {
            throw new NotEnoughStockException("need more stock");
        }
        this.stockQuantity = restStock;
    }

}
```

예외 추가

```
package jpabook.jpashop.exception;

public class NotEnoughStockException extends RuntimeException {

    public NotEnoughStockException() {
    }

    public NotEnoughStockException(String message) {
        super(message);
    }

    public NotEnoughStockException(String message, Throwable cause) {
        super(message, cause);
    }
```

```
    public NotEnoughStockException(Throwable cause) {
        super(cause);
    }

}
```

비즈니스 로직 분석

- `addStock()` 메서드는 파라미터로 넘어온 수만큼 재고를 늘린다. 이 메서드는 재고가 증가하거나 상품 주문을 취소해서 재고를 다시 늘려야 할 때 사용한다.
- `removeStock()` 메서드는 파라미터로 넘어온 수만큼 재고를 줄인다. 만약 재고가 부족하면 예외가 발생한다. 주로 상품을 주문할 때 사용한다.

상품 리포지토리 개발

상품 리포지토리 코드

```
package jpabook.jpashop.repository;

import jpabook.jpashop.domain.item.Item;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Repository;

import javax.persistence.EntityManager;
import java.util.List;

@Repository
@RequiredArgsConstructor
public class ItemRepository {

    private final EntityManager em;

    public void save(Item item) {
        if (item.getId() == null) {
            em.persist(item);
        } else {
            em.merge(item);
        }
    }

    public Item findOne(Long id) {
```

```
        return em.find(Item.class, id);
    }

    public List<Item> findAll() {
        return em.createQuery("select i from Item
i",Item.class).getResultList();
    }
}
```

기능 설명

- `save()`
  - `id`가 없으면 신규로 보고 `persist()` 실행
  - `id`가 있으면 이미 데이터베이스에 저장된 엔티티를 수정한다고 보고, `merge()`를 실행, 자세한 내용은 뒤에 웹에서 설명(그냥 지금은 저장한다 정도로 생각하자)

상품 서비스 개발

상품 서비스 코드

```
package jpabook.jpashop.service;

import jpabook.jpashop.domain.item.Item;
import jpabook.jpashop.repository.ItemRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;
import java.util.List;

@Service
@Transactional(readOnly = true)
@RequiredArgsConstructor
public class ItemService {

    private final ItemRepository itemRepository;

    @Transactional
    public void saveItem(Item item) {
        itemRepository.save(item);
    }

    public List<Item> findItems() {
```

```
        return itemRepository.findAll();
    }

    public Item findOne(Long itemId) {
        return itemRepository.findOne(itemId);
    }
}
```

상품 서비스는 상품 리포지토리에 단순히 위임만 하는 클래스

상품 기능 테스트

상품 테스트는 회원 테스트와 비슷하므로 생략