3조 다목적 쇼핑몰 웹 사이트 발표

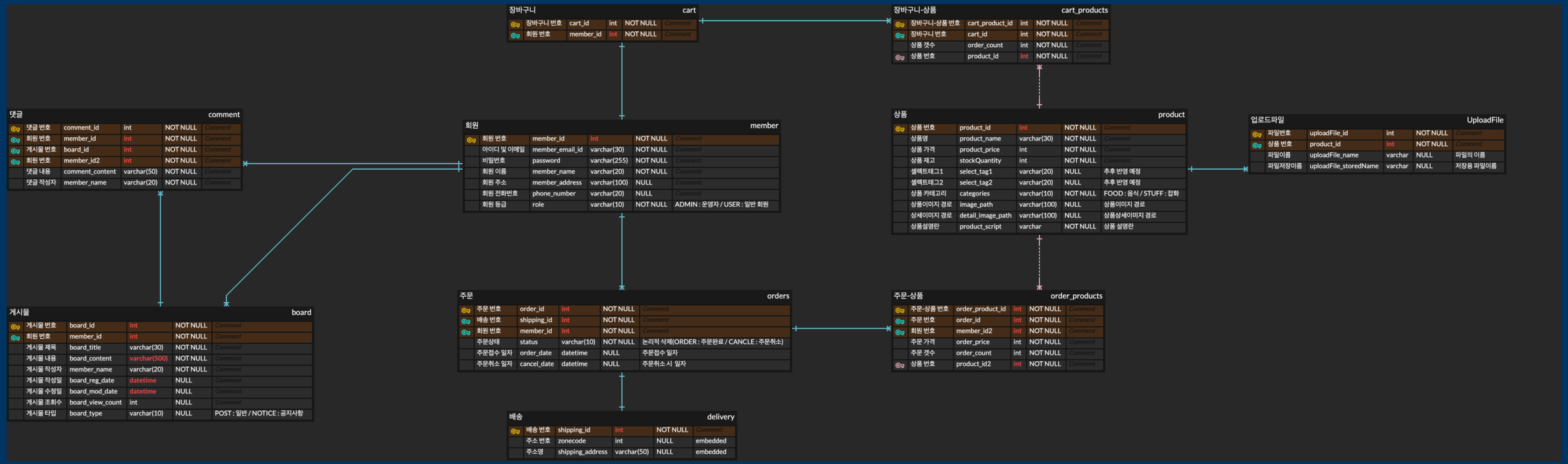
권동혁, 서정국

2023.10.10 (화)

제작 목표
이번 팀 프로젝트를 진행하며…

- 다목적상품을 판매하는 쇼핑몰 웹사이트 제작하기.
- Spring FrameWork 구조와 WEB 기초 이해
- 단위 및 통합 테스트 코드의 이해
- 협업을 위한 Git Skill Up 및 커뮤니케이션 능력 학습

DATA BASE 설계



장바구니 — cart

장바구니 번호	cart_id	int	NOT NULL	Comment
회원 번호	member_id	int	NOT NULL	Comment

장바구니-상품 — cart_products

장바구니-상품 번호	cart_product_id	int	NOT NULL	Comment
장바구니 번호	cart_id	int	NOT NULL	Comment
상품 갯수	order_count	int	NOT NULL	Comment
상품 번호	product_id	int	NOT NULL	Comment

댓글 — comment

댓글 번호	comment_id	int	NOT NULL	Comment
회원 번호	member_id	int	NOT NULL	Comment
게시물 번호	board_id	int	NOT NULL	Comment
회원 번호	member_id2	int	NOT NULL	Comment
댓글 내용	comment_content	varchar(50)	NOT NULL	Comment
댓글 작성자	member_name	varchar(20)	NOT NULL	Comment

회원 — member

회원 번호	member_id	int	NOT NULL	Comment
아이디 및 이메일	member_email_id	varchar(30)	NOT NULL	Comment
비밀번호	password	varchar(255)	NOT NULL	Comment
회원 이름	member_name	varchar(20)	NOT NULL	Comment
회원 주소	member_address	varchar(100)	NULL	Comment
회원 전화번호	phone_number	varchar(20)	NULL	Comment
회원 등급	role	varchar(10)	NOT NULL	ADMIN : 운영자 / USER : 일반 회원

상품 — product

상품 번호	product_id	int	NOT NULL	Comment
상품명	product_name	varchar(30)	NOT NULL	Comment
상품 가격	product_price	int	NOT NULL	Comment
상품 재고	stockQuantity	int	NOT NULL	Comment
셀렉트태그1	select_tag1	varchar(20)	NULL	추후 반영 예정
셀렉트태그2	select_tag2	varchar(20)	NULL	추후 반영 예정
상품 카테고리	categories	varchar(10)	NOT NULL	FOOD : 음식 / STUFF : 잡화
상품이미지 경로	image_path	varchar(100)	NULL	상품이미지 경로
상세이미지 경로	detail_image_path	varchar(100)	NULL	상품상세이미지 경로
상품설명란	product_script	varchar	NOT NULL	상품 설명란

업로드파일 — UploadFile

파일번호	uploadFile_id	int	NOT NULL	Comment
상품 번호	product_id	int	NOT NULL	Comment
파일이름	uploadFile_name	varchar	NULL	파일의 이름
파일저장이름	uploadFile_storedName	varchar	NULL	저장용 파일이름

게시물 — board

게시물 번호	board_id	int	NOT NULL	Comment
회원 번호	member_id	int	NOT NULL	Comment
게시물 제목	board_title	varchar(30)	NOT NULL	Comment
게시물 내용	board_content	varchar(500)	NOT NULL	Comment
게시물 작성자	member_name	varchar(20)	NOT NULL	Comment
게시물 작성일	board_reg_date	datetime	NULL	Comment
게시물 수정일	board_mod_date	datetime	NULL	Comment
게시물 조회수	board_view_count	int	NULL	Comment
게시물 타입	board_type	varchar(10)	NULL	POST : 일반 / NOTICE : 공지사항

주문 — orders

주문 번호	order_id	int	NOT NULL	Comment
배송 번호	shipping_id	int	NOT NULL	Comment
회원 번호	member_id	int	NOT NULL	Comment
주문상태	status	varchar(10)	NOT NULL	논리적 삭제(ORDER : 주문완료 / CANCLE : 주문취소)
주문접수 일자	order_date	datetime	NULL	주문접수 일자
주문취소 일자	cancel_date	datetime	NULL	주문취소 시 일자

주문-상품 — order_products

주문-상품 번호	order_product_id	int	NOT NULL	Comment
주문 번호	order_id	int	NOT NULL	Comment
회원 번호	member_id2	int	NOT NULL	Comment
주문 가격	order_price	int	NOT NULL	Comment
주문 갯수	order_count	int	NOT NULL	Comment
상품 번호	product_id2	int	NOT NULL	Comment

배송 — delivery

배송 번호	shipping_id	int	NOT NULL	Comment
주소 번호	zonecode	int	NULL	embedded
주소명	shipping_address	varchar(50)	NULL	embedded

주문 내역의 삭제는 논리적 삭제 진행

다목적 쇼핑몰 웹사이트 제작하기

쇼핑몰 웹사이트의 구현 기능목록

1. 상품 CRUD API
2. 상품 장바구니, 결제 API
3. 회원 CRUD API
4. 고객 문의 게시판 CRUD API

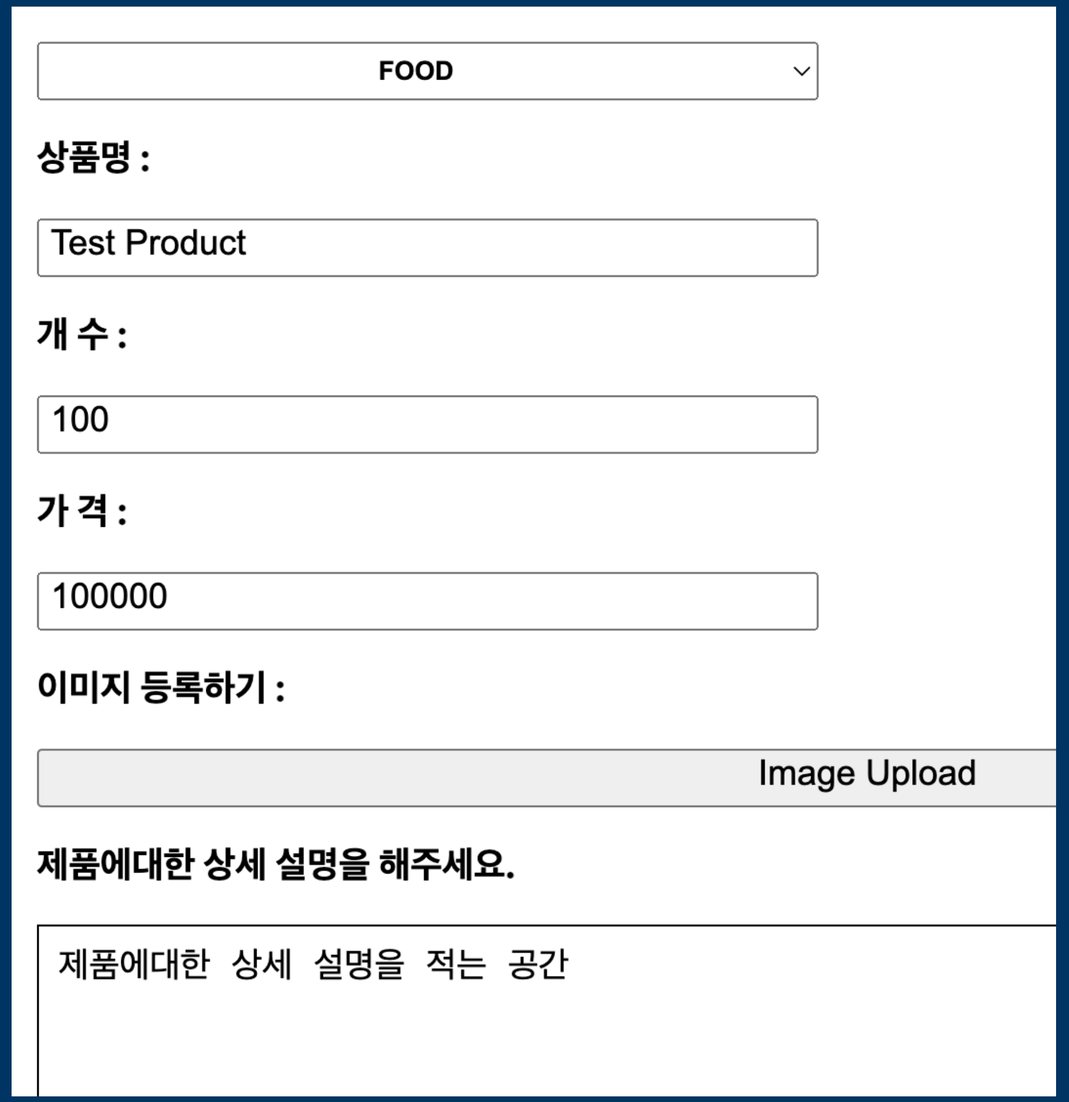
외부 API

1. KAKAO Address API
2. Iamport API

다목적 쇼핑몰 웹사이트 제작하기
상품 CRUD API

```java
@RequiredArgsConstructor
@Slf4j
public class ProductsAPIController {

    private final ProductsService productsService;

    no usages  kwon93 +1
    @PostMapping(value = "/api/products")
    public ResponseEntity<ProductsResponseDTO> postProducts(@RequestBody Pro
        ProductsResponseDTO responseDTO = productsService.productSaveByReque
        log.info("dto = {}", responseDTO.getProductScript());
        return ResponseEntity.status(HttpStatus.CREATED).body(responseDTO);
    }

    no usages  kwon93
    @PutMapping(value = "/api/products/{id}", consumes = MediaType.APPLICATI
    public ResponseEntity updateProducts(@RequestBody ProductsRequestDTO req
        productsService.productUpdateByRequestAndId(requestDTO, productId);
        return ResponseEntity.noContent().build();
```

API 와 View Controller의 분리



실제 웹사이트 상품 등록 화면

다목적 쇼핑몰 웹사이트 제작하기
상품 장바구니 및 결제 API

```
no usages  wjdrnrdl97
@PostMapping("/api/cart")
public ResponseEntity<CartResponseDTO> postCartByRequest(@Request
    CartResponseDTO response = cartService.createCartWithProducts
    return ResponseEntity.status(HttpStatus.CREATED).body(respons
}

no usages  wjdrnrdl97
@GetMapping("/api/cart/{memberId}")
public ResponseEntity<List<CartProductsResponseDTO>>getCartWithPr
    List<CartProductsResponseDTO> response = cartService.getCartW
    return ResponseEntity.ok().body(response);
}

no usages  wjdrnrdl97
@DeleteMapping("/api/cart/{id}")
public ResponseEntity deleteCartProductsById(@PathVariable("id")L
    cartService.deleteCartProductsById(id);
    return ResponseEntity.noContent().build();
}
no usages  wjdrnrdl97
@DeleteMapping("/api/cart/all/{memberId}")
```

설명

```
if(cart_btn){
    cart_btn.addEventListener('click',e => {
        fetch(`/api/cart`,
            {
                method: 'POST',
                headers: {"Content-Type": "application/json"},
                body : JSON.stringify({
                    productsId : document.querySelector('.productId').value,
                    memberId : document.getElementById('memberId').value,
                    count : quantity.innerHTML
                })
            }
        )
        .then(()=>{
            alert("장바구니에 담겼습니다.");
        })
    })
}
```

VanillaScript를 활용해 fetch API로 API 요청 진행

다목적 쇼핑몰 웹사이트 제작하기
회원 CRUD API

현재 Spring Interceptor 기능 활용

추후 Spring Security 적용 예정

설명

다목적 쇼핑몰 웹사이트 제작하기
고객 문의 게시판 API

test 고객문의

1 / WRITER : kwon / 2023-10-10 / 1

test

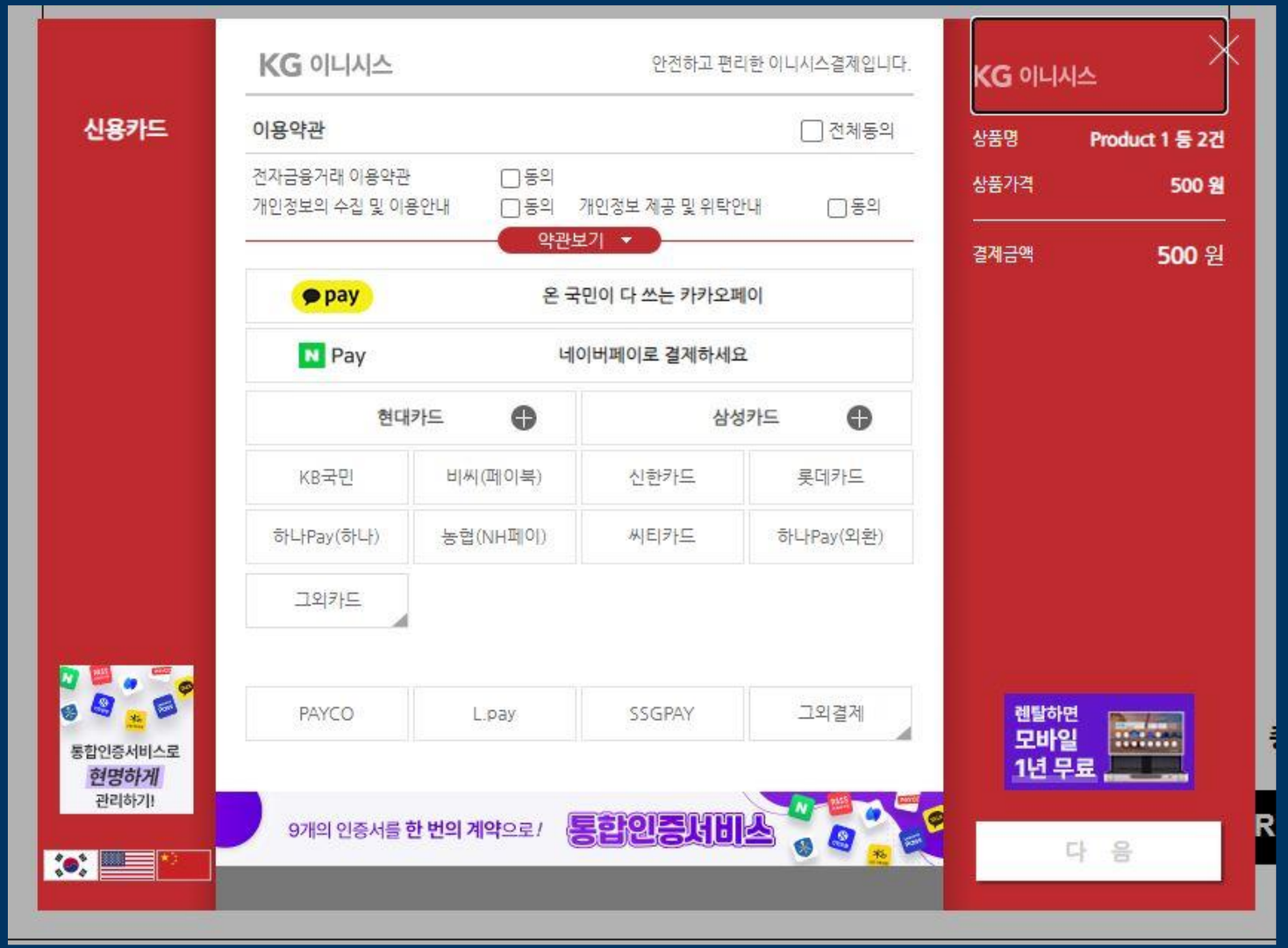
LIST UPDATE DELETE

Comment

Please enter your comment.

COMMNET SUBMIT

다목적 쇼핑몰 웹사이트 제작하기
결제 외부 API



다목적 쇼핑몰 웹사이트 제작하기
주문-재고 기능 / 동시성 제어

```java
public void decreaseStockQuantity(Integer count){  Complexity is 5 Everything is cool!
    Integer checkStock = this.stockQuantity = stockQuantity - count;
    if (checkStock < 0){
        throw new RuntimeException("재고가 부족합니다.");
    }
    this.stockQuantity = checkStock;
}
1 usage  wjdrnrdl97
public void increaseStockQunatity(Integer count){
    this.stockQuantity += count;
}
```

주문 생성 -> 상품 재고 감소

주문 취소 -> 상품 재고 증가

다목적 쇼핑몰 웹사이트 제작하기
주문-재고 기능 / 동시성 제어

```
@Test
@DisplayName("주문 요청을 받아 주문을 생성에 성공한다.")
public void save(){
```

```
OrderProductsRequestDTO dto1 = OrderProductsRequestDTO.builder()
        .productId(1L)
        .count(4)
        .build();
```

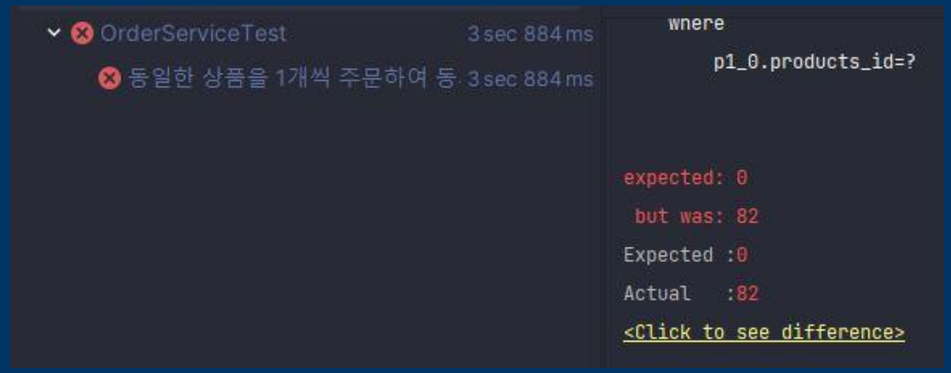
```
    //when
    OrderResponseDTO result = orderService.save(request);
    //then
    List<OrderProducts> orderProducts = orderProductsRepository.findAll();
    Products productResult = productsRepository.findById(1L).orElse( other: null);
    assertThat(result).isNotNull();
    assertThat(orderProducts).hasSize( expected: 2);
    assertThat(productResult.getStockQuantity()).isEqualTo( expected: 100);
}
```

4개 주문 -> 상품 재고 4개 감소

다목적 쇼핑몰 웹사이트 제작하기
주문-재고 기능 / 동시성 제어

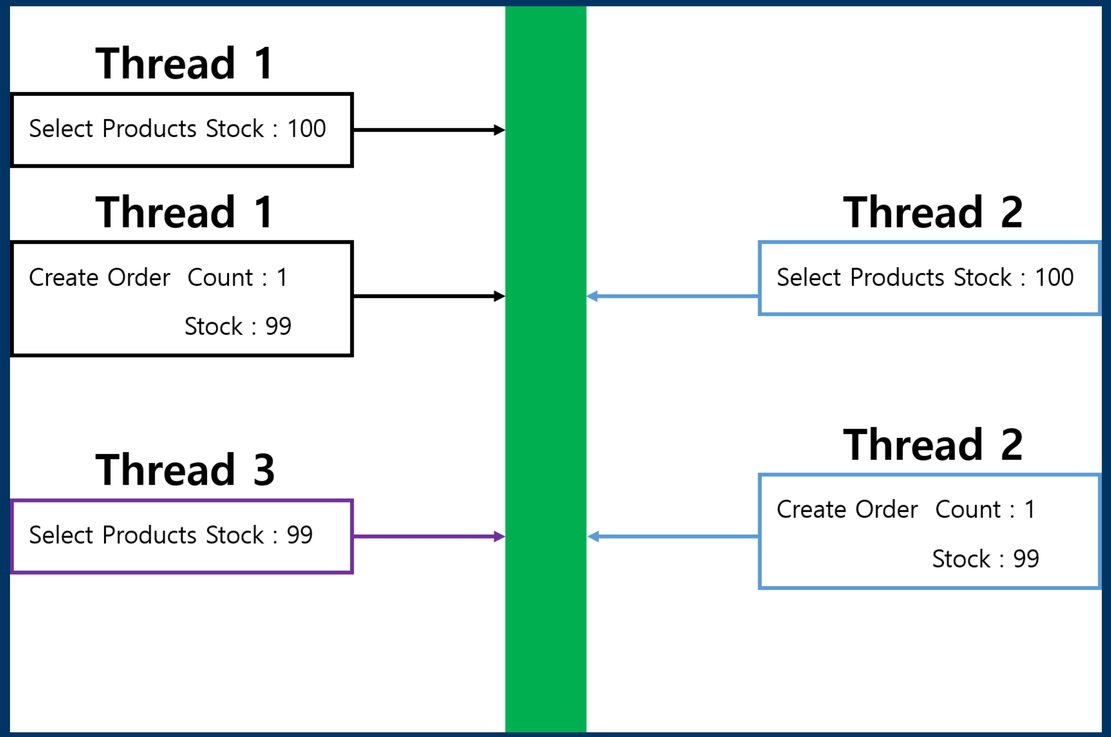
```
@Test
@DisplayName("동일한 상품을 1개씩 주문하여 동시에 100번 요청하였을 때 상품의 재고수량이 0이여야한다.")
public void saveWithPessimisticLock() throws InterruptedException {  Complexity is 4 Everything is cool!
```

```
int threadCount = 100;
ExecutorService executorService = Executors.newFixedThreadPool( nThreads: 10);
CountDownLatch latch = new CountDownLatch(threadCount);
for(int i = 0; i < threadCount; i++){
    executorService.submit(()→ {
        try {
            orderService.save(request);
        }
        finally {
            latch.countDown();
        }
    });
}
latch.await();
//then
Products result = productsRepository.findById(1L).orElse( other: null);
assertThat(result.getStockQuantity()).isEqualTo( expected: 0);
```

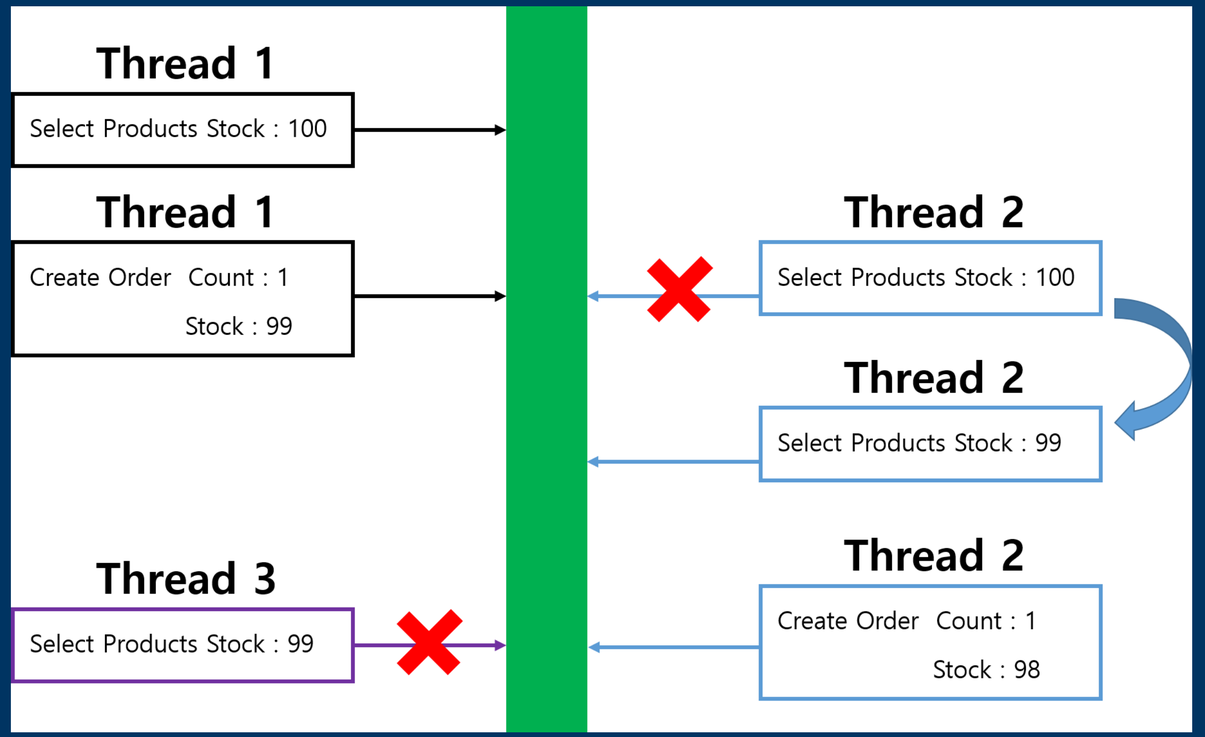


기댓값 : 1개 주문 X 100 -> 상품 재고 100개 감소

다목적 쇼핑몰 웹사이트 제작하기
주문-재고 기능 / 동시성 제어



< 일반 Thread >



< Pessimistic Lock >

다목적 쇼핑몰 웹사이트 제작하기
주문-재고 기능 / 동시성 제어

```
@Test
@DisplayName("동일한 상품을 1개씩 주문하여 동시에 100번 요청하였을 때 상품의 재고수량이 0이여야한다.")
public void saveWithPessimisticLock() throws InterruptedException {  Complexity is 4 Everything is cool!
```

```
int threadCount = 100;
ExecutorService executorService = Executors.newFixedThreadPool( nThreads: 10);
CountDownLatch latch = new CountDownLatch(threadCount);
for(int i = 0; i < threadCount; i++){
    executorService.submit(()→ {
        try {
            orderService.save(request);
        }
        finally {
            latch.countDown();
        }
    });
}
latch.await();
//then
Products result = productsRepository.findById(1L).orElse( other: null);
assertThat(result.getStockQuantity()).isEqualTo( expected: 0);
```

```
@Lock(LockModeType.PESSIMISTIC_WRITE)
@Query("select p from Products p where p.id = :id")
Optional<Products> findAndPessimisticLockById(Long id);
```

Pessimistic Lock 설정

OrderServiceTest 3 sec 348 ms

동일한 상품을 1개씩 주문하여 동 3 sec 348 ms

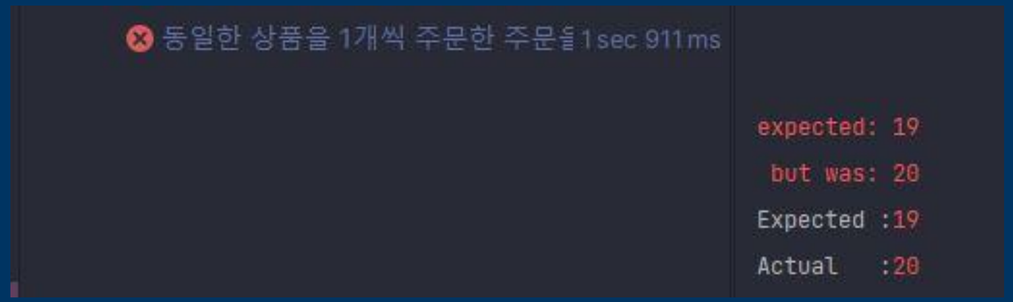
```
p1_0.products_id=?

expected: 1
 but was: 0
Expected :1
Actual   :0
<Click to see difference>
```

기댓값 : 1개 주문 X 100 -> 상품 재고 100개 감소

다목적 쇼핑몰 웹사이트 제작하기
주문-재고 기능 / 동시성 제어

```
@Test
@DisplayName("동일한 상품을 1개씩 주문한 주문을 동시에 20개 취소하였을 때 재고수량이 20이 증가해야한다. ")
public void test() throws InterruptedException {...}
```

```
int threadCount = 20;
ExecutorService executorService = Executors.newFixedThreadPool( nThreads: 5);
CountDownLatch latch = new CountDownLatch(threadCount);
for(int i = 0; i < threadCount; i++){
    Long orderId = 1L + i;
    executorService.submit(()→ {
        try {
            orderService.deleteByOrdersIds(orderId);
        }
        finally {
            latch.countDown();
        }
    });
}
latch.await();
```



기댓값 : 1개 X 20개 주문취소 -> 상품 재고 20 증가

다목적 쇼핑몰 웹사이트 제작하기

실제 Local Server Application 시연