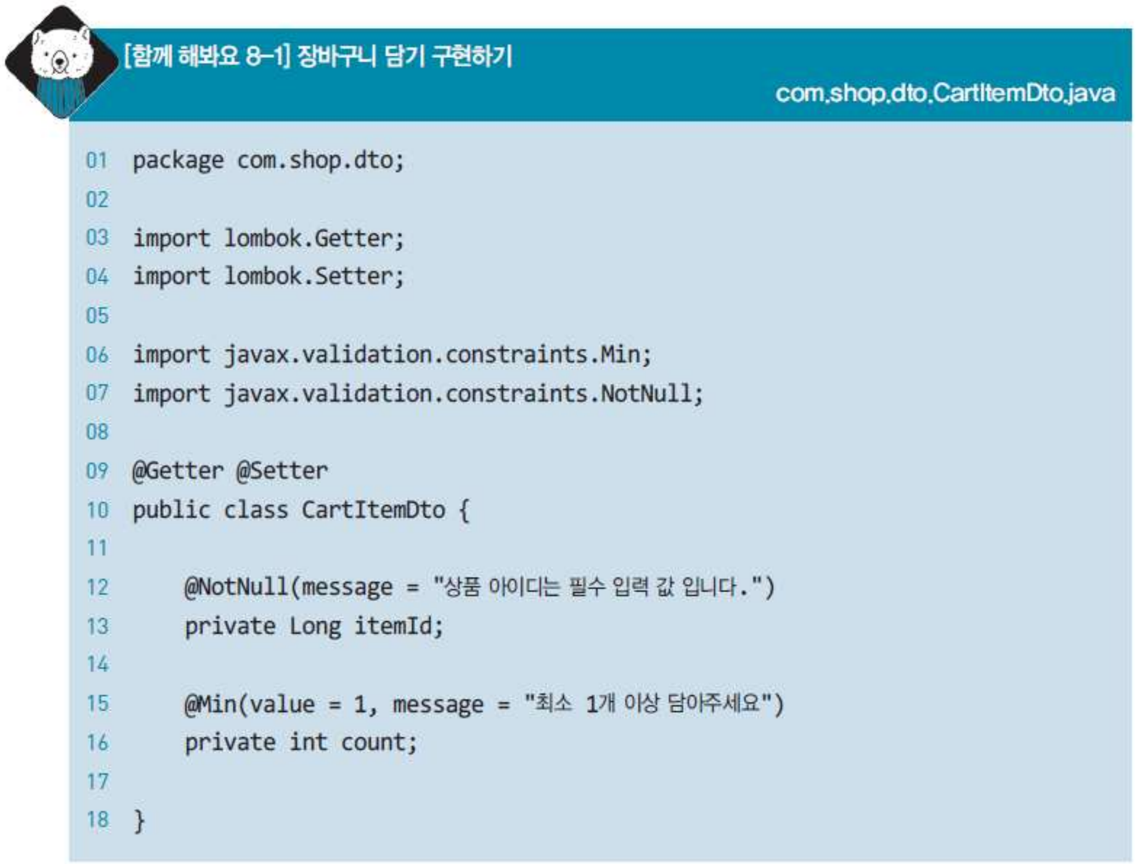
학습 목표

- 1. 장바구니 기능으로 상품을 장바구니에 담거나 주문, 삭제하는 프로세스를 학습한다

8.1 장바구니 담기

- 상품 상세 페이지에서 장바구니에 담을 수량을 선택하고 장바구니 담기 버튼을 클릭할 때 상품이 장바구니에 담기는 기능 구현



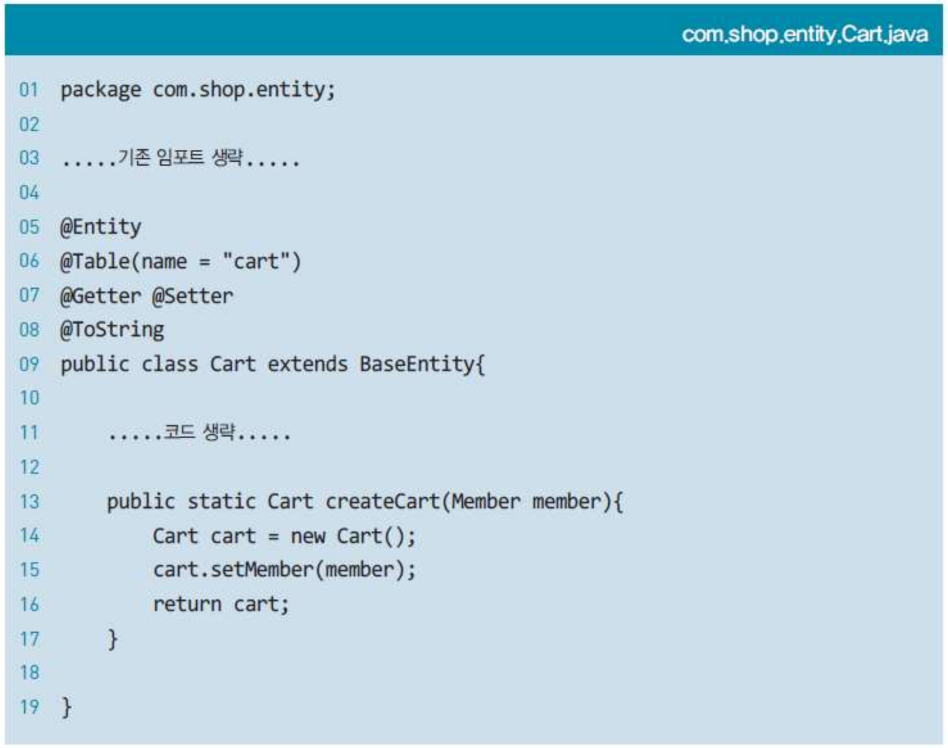
[함께 해봐요 8-1] 장바구니 담기 구현하기

com.shop.dto.CartItemDto.java

```java
package com.shop.dto;

import lombok.Getter;
import lombok.Setter;

import javax.validation.constraints.Min;
import javax.validation.constraints.NotNull;

@Getter @Setter
public class CartItemDto {

    @NotNull(message = "상품 아이디는 필수 입력 값 입니다.")
    private Long itemId;

    @Min(value = 1, message = "최소 1개 이상 담아주세요")
    private int count;

}
```

8.1 장바구니 담기

- 회원 한 명당 1개의 장바구니를 갖으므로 처음 장바구니에 상품을 담을 때는 해당 회원의 장바구니
- Cart 엔티티에 회원 엔티티를 파라미터로 받아서 장바구니 엔티티를 생성 로직 추가

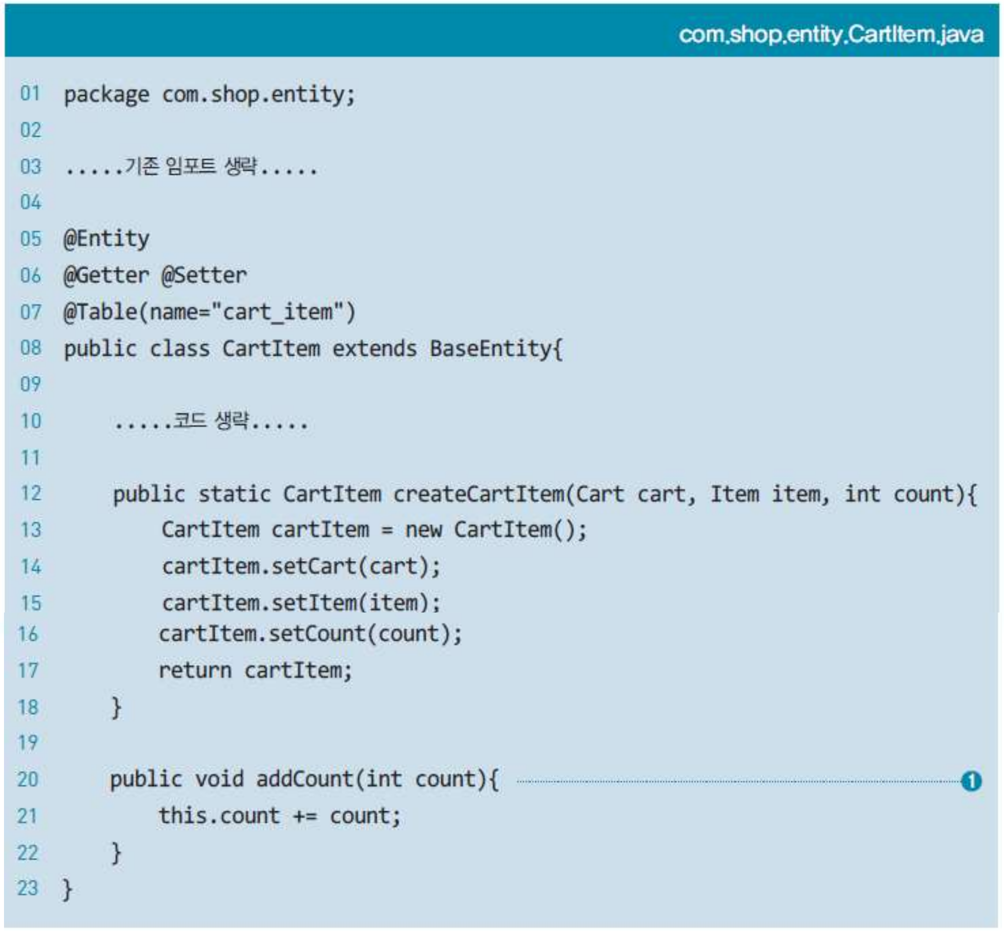


com.shop.entity.Cart.java

```java
package com.shop.entity;

.....기존 임포트 생략.....

@Entity
@Table(name = "cart")
@Getter @Setter
@ToString
public class Cart extends BaseEntity{

    .....코드 생략.....

    public static Cart createCart(Member member){
        Cart cart = new Cart();
        cart.setMember(member);
        return cart;
    }

}
```

8.1 장바구니 담기

- 장바구니에 담을 상품 엔티티를 생성하는 메소드와 장바구니에 담을 수량을 증가시켜 주는 메소드 추가



com.shop.entity.CartItem.java

```
package com.shop.entity;

.....기존 임포트 생략.....

@Entity
@Getter @Setter
@Table(name="cart_item")
public class CartItem extends BaseEntity{

    .....코드 생략.....

    public static CartItem createCartItem(Cart cart, Item item, int count){
        CartItem cartItem = new CartItem();
        cartItem.setCart(cart);
        cartItem.setItem(item);
        cartItem.setCount(count);
        return cartItem;
    }

    public void addCount(int count){ ❶
        this.count += count;
    }
}
```

8.1 장바구니 담기

- 현재 로그인한 회원의 Cart 엔티티를 찾기 위해서 CartRepository에 쿼리 메소드 추가

com.shop.repository.CartRepository.java

```java
package com.shop.repository;

import com.shop.entity.Cart;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CartRepository extends JpaRepository<Cart, Long> {

    Cart findByMemberId(Long memberId);

}
```

8.1 장바구니 담기

- 장바구니에 들어갈 상품을 저장하거나 조회하기 위해서 CartItemRepository 인터페이스를 생성

com.shop.repository.CartItemRepository.java

```java
package com.shop.repository;

import com.shop.entity.CartItem;
import org.springframework.data.jpa.repository.JpaRepository;

public interface CartItemRepository extends JpaRepository<CartItem, Long> {

    CartItem findByCartIdAndItemId(Long cartId, Long itemId);

}
```

8.1 장바구니 담기

com.shop.service.CartService.java

```java
package com.shop.service;

import com.shop.dto.CartItemDto;
import com.shop.entity.Cart;
import com.shop.entity.CartItem;
import com.shop.entity.Item;
import com.shop.entity.Member;
import com.shop.repository.CartItemRepository;
import com.shop.repository.CartRepository;
import com.shop.repository.ItemRepository;
import com.shop.repository.MemberRepository;
import lombok.RequiredArgsConstructor;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import javax.persistence.EntityNotFoundException;

@Service
@RequiredArgsConstructor
@Transactional
public class CartService {

    private final ItemRepository itemRepository;
    private final MemberRepository memberRepository;
    private final CartRepository cartRepository;
    private final CartItemRepository cartItemRepository;

    public Long addCart(CartItemDto cartItemDto, String email){

```

8.1 장바구니 담기

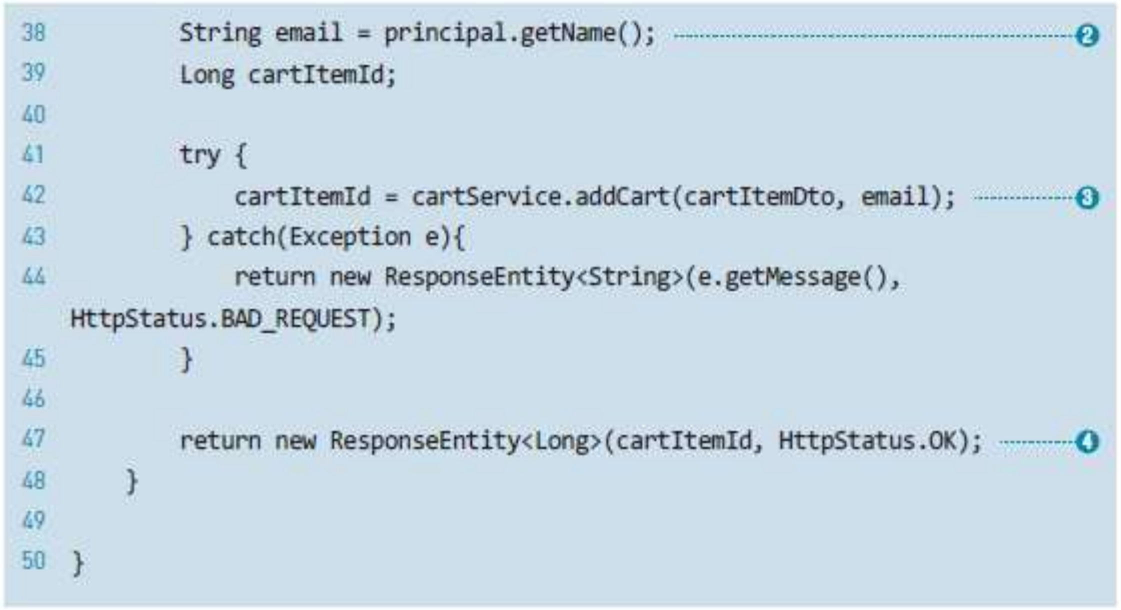
```java
        Item item = itemRepository.findById(cartItemDto.getItemId()) ❶
                .orElseThrow(EntityNotFoundException::new);
        Member member = memberRepository.findByEmail(email); ❷

        Cart cart = cartRepository.findByMemberId(member.getId()); ❸
        if(cart == null){ ❹
            cart = Cart.createCart(member);
            cartRepository.save(cart);
        }

        CartItem savedCartItem =
        cartItemRepository.findByCartIdAndItemId(cart.getId(), item.getId()); ❺

        if(savedCartItem != null){
            savedCartItem.addCount(cartItemDto.getCount()); ❻
            return savedCartItem.getId();
        } else {
            CartItem cartItem =
CartItem.createCartItem(cart, item, cartItemDto.getCount()); ❼
            cartItemRepository.save(cartItem); ❽
            return cartItem.getId();
        }

    }

}
```

8.1 장바구니 담기

com.shop.controller.CartController.java

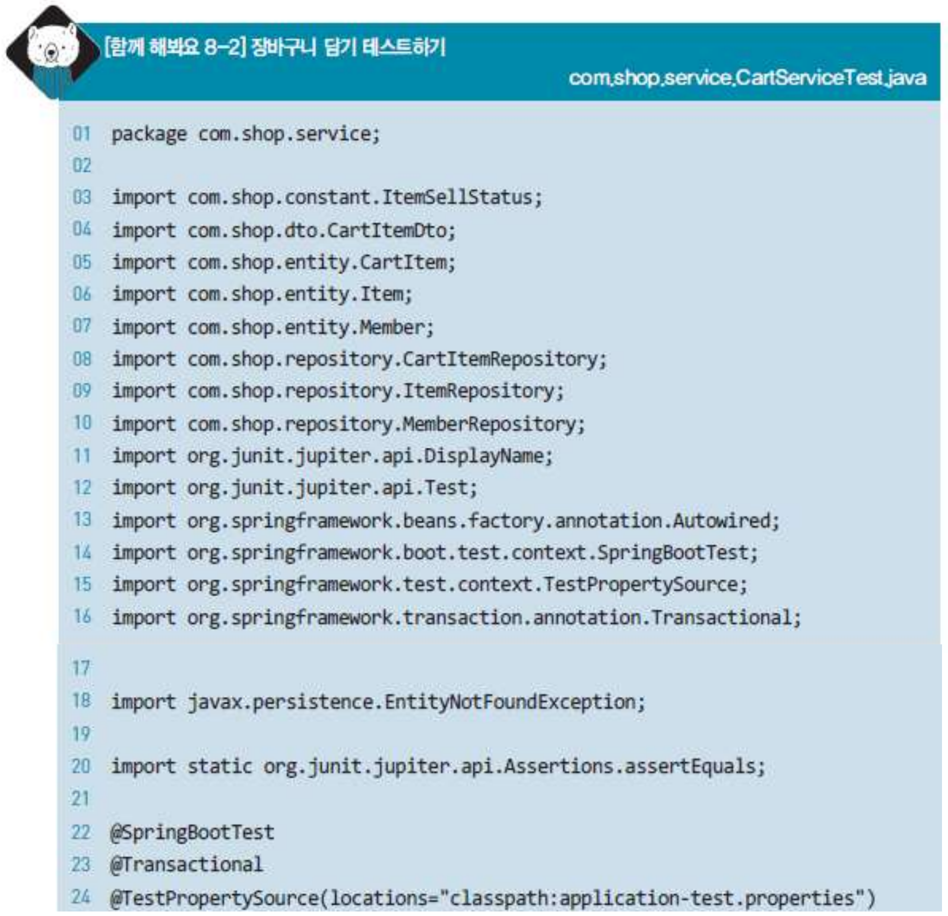
```java
package com.shop.controller;

import com.shop.dto.CartItemDto;
import com.shop.service.CartService;
import lombok.RequiredArgsConstructor;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
import org.springframework.validation.FieldError;
import org.springframework.web.bind.annotation.PostMapping;
import org.springframework.web.bind.annotation.RequestBody;
import org.springframework.web.bind.annotation.ResponseBody;

import javax.validation.Valid;
import java.security.Principal;
import java.util.List;

@Controller
@RequiredArgsConstructor
public class CartController {

    private final CartService cartService;

    @PostMapping(value = "/cart")
    public @ResponseBody
    ResponseEntity order(@RequestBody @Valid CartItemDto cartItemDto,
BindingResult bindingResult, Principal principal){

        if(bindingResult.hasErrors()){ ❶
            StringBuilder sb = new StringBuilder();
            List<FieldError> fieldErrors = bindingResult.getFieldErrors();
            for (FieldError fieldError : fieldErrors) {
                sb.append(fieldError.getDefaultMessage());
            }
            return new ResponseEntity<String>
(sb.toString(), HttpStatus.BAD_REQUEST);
        }

```

8.1 장바구니 담기



```
        String email = principal.getName(); ❷
        Long cartItemId;

        try {
            cartItemId = cartService.addCart(cartItemDto, email); ❸
        } catch(Exception e){
            return new ResponseEntity<String>(e.getMessage(),
HttpStatus.BAD_REQUEST);
        }

        return new ResponseEntity<Long>(cartItemId, HttpStatus.OK); ❹
    }

}
```

8.1 장바구니 담기

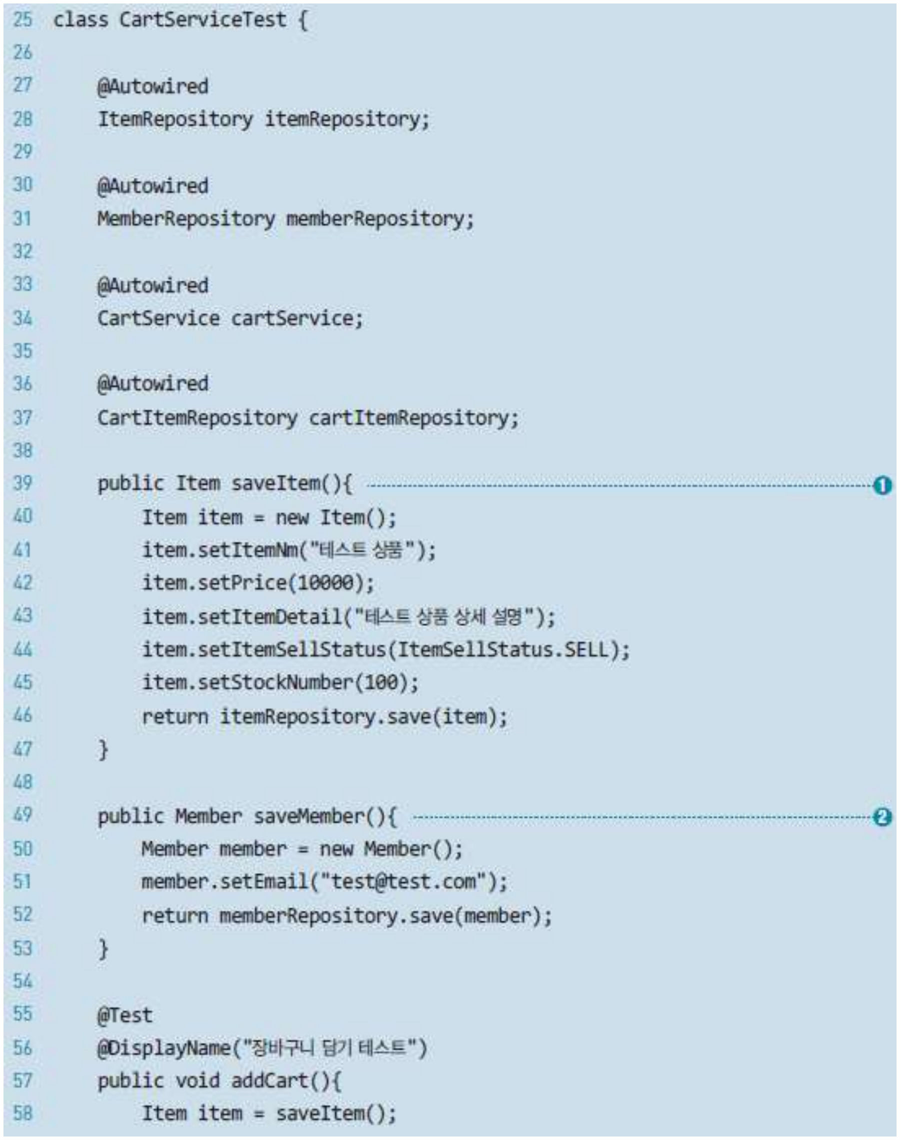


[함께 해봐요 8-2] 장바구니 담기 테스트하기

com.shop.service.CartServiceTest.java

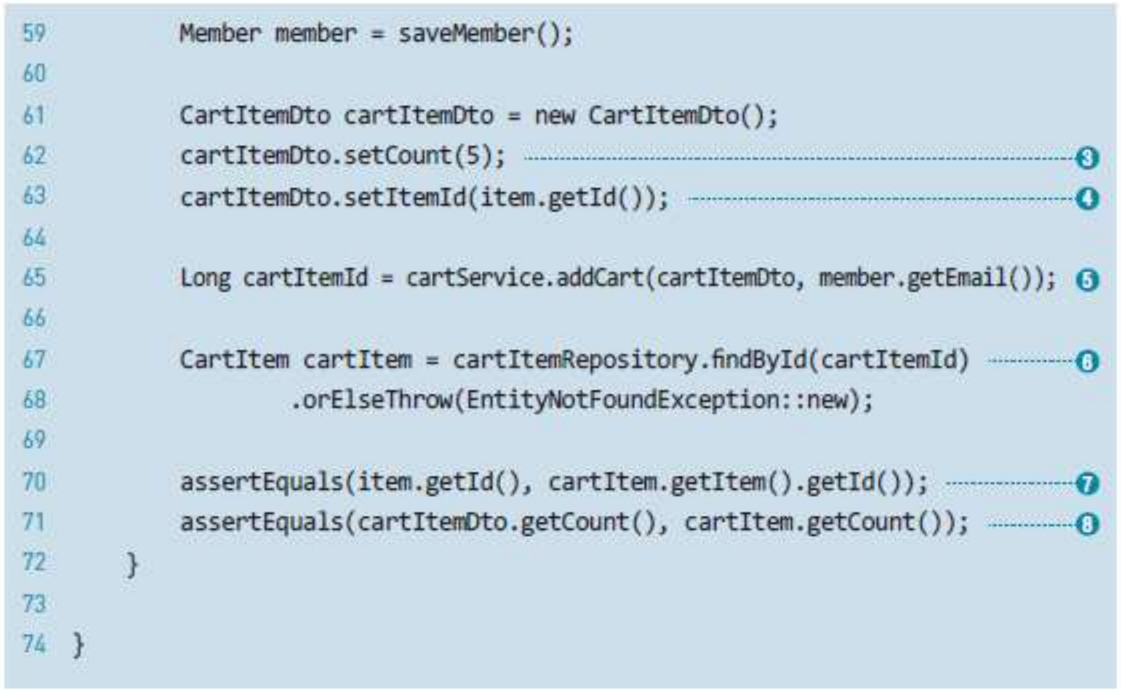
```java
package com.shop.service;

import com.shop.constant.ItemSellStatus;
import com.shop.dto.CartItemDto;
import com.shop.entity.CartItem;
import com.shop.entity.Item;
import com.shop.entity.Member;
import com.shop.repository.CartItemRepository;
import com.shop.repository.ItemRepository;
import com.shop.repository.MemberRepository;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import org.springframework.test.context.TestPropertySource;
import org.springframework.transaction.annotation.Transactional;

import javax.persistence.EntityNotFoundException;

import static org.junit.jupiter.api.Assertions.assertEquals;

@SpringBootTest
@Transactional
@TestPropertySource(locations="classpath:application-test.properties")
```

8.1 장바구니 담기



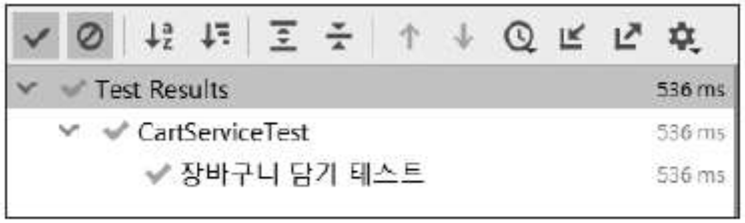
```
class CartServiceTest {

    @Autowired
    ItemRepository itemRepository;

    @Autowired
    MemberRepository memberRepository;

    @Autowired
    CartService cartService;

    @Autowired
    CartItemRepository cartItemRepository;

    public Item saveItem(){ ❶
        Item item = new Item();
        item.setItemNm("테스트 상품");
        item.setPrice(10000);
        item.setItemDetail("테스트 상품 상세 설명");
        item.setItemSellStatus(ItemSellStatus.SELL);
        item.setStockNumber(100);
        return itemRepository.save(item);
    }

    public Member saveMember(){ ❷
        Member member = new Member();
        member.setEmail("test@test.com");
        return memberRepository.save(member);
    }

    @Test
    @DisplayName("장바구니 담기 테스트")
    public void addCart(){
        Item item = saveItem();
```

8.1 장바구니 담기



```
        Member member = saveMember();

        CartItemDto cartItemDto = new CartItemDto();
        cartItemDto.setCount(5);                                          ❸
        cartItemDto.setItemId(item.getId());                              ❹

        Long cartItemId = cartService.addCart(cartItemDto, member.getEmail()); ❺

        CartItem cartItem = cartItemRepository.findById(cartItemId)       ❻
                .orElseThrow(EntityNotFoundException::new);

        assertEquals(item.getId(), cartItem.getItem().getId());           ❼
        assertEquals(cartItemDto.getCount(), cartItem.getCount());        ❽
    }

}
```

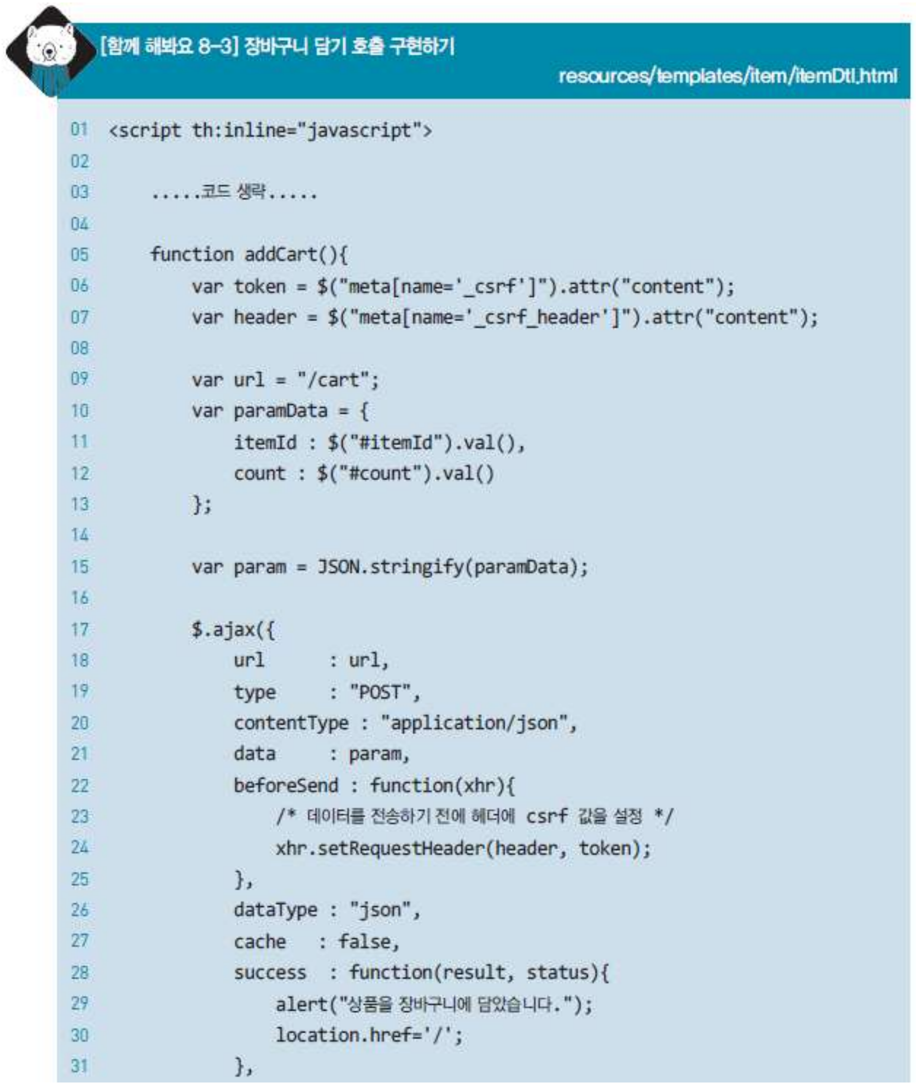
8.1 장바구니 담기



```
Test Results                536 ms
  CartServiceTest           536 ms
    장바구니 담기 테스트      536 ms
```

[그림 8-1] 장바구니 담기 테스트 코드 실행 결과

8.1 장바구니 담기



[함께 해봐요 8-3] 장바구니 담기 호출 구현하기

resources/templates/item/itemDtl.html

```
<script th:inline="javascript">

    .....코드 생략.....

    function addCart(){
        var token = $("meta[name='_csrf']").attr("content");
        var header = $("meta[name='_csrf_header']").attr("content");

        var url = "/cart";
        var paramData = {
            itemId : $("#itemId").val(),
            count : $("#count").val()
        };

        var param = JSON.stringify(paramData);

        $.ajax({
            url      : url,
            type     : "POST",
            contentType : "application/json",
            data     : param,
            beforeSend : function(xhr){
                /* 데이터를 전송하기 전에 헤더에 csrf 값을 설정 */
                xhr.setRequestHeader(header, token);
            },
            dataType : "json",
            cache   : false,
            success  : function(result, status){
                alert("상품을 장바구니에 담았습니다.");
                location.href='/';
            },
```

8.1 장바구니 담기

```
            error : function(jqXHR, status, error){

                if(jqXHR.status == '401'){
                    alert('로그인 후 이용해주세요');
                    location.href='/members/login';
                } else{
                    alert(jqXHR.responseText);
                }


            }
        });
    }

</script>
```

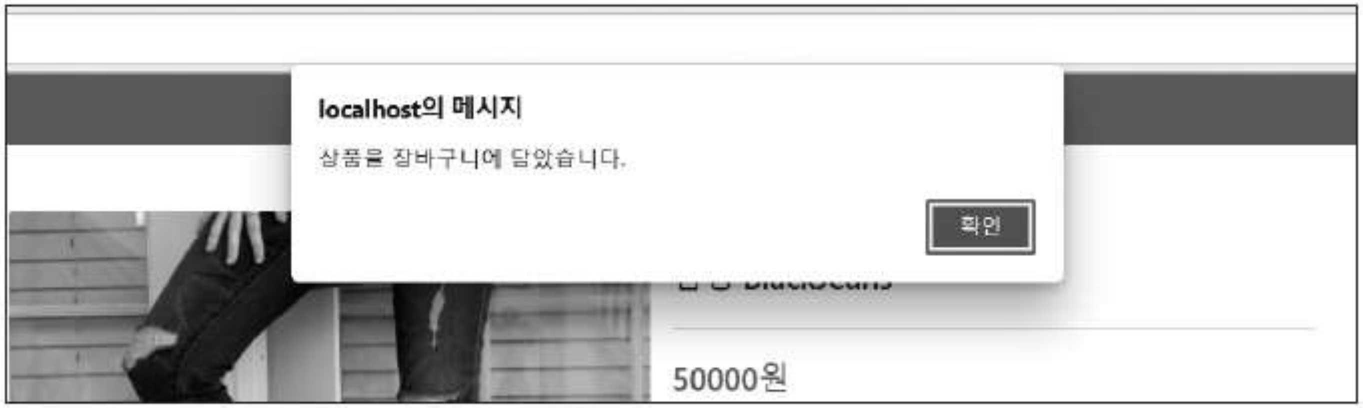
8.1 장바구니 담기

- 장바구니 담기 버튼 클릭시 addCart() 함수 호출 추가

resources/templates/item/itemDtl.html

```
<button type="button" class="btn btn-light border border-primary btn-lg"
 onclick="addCart()">장바구니 담기</button>
```

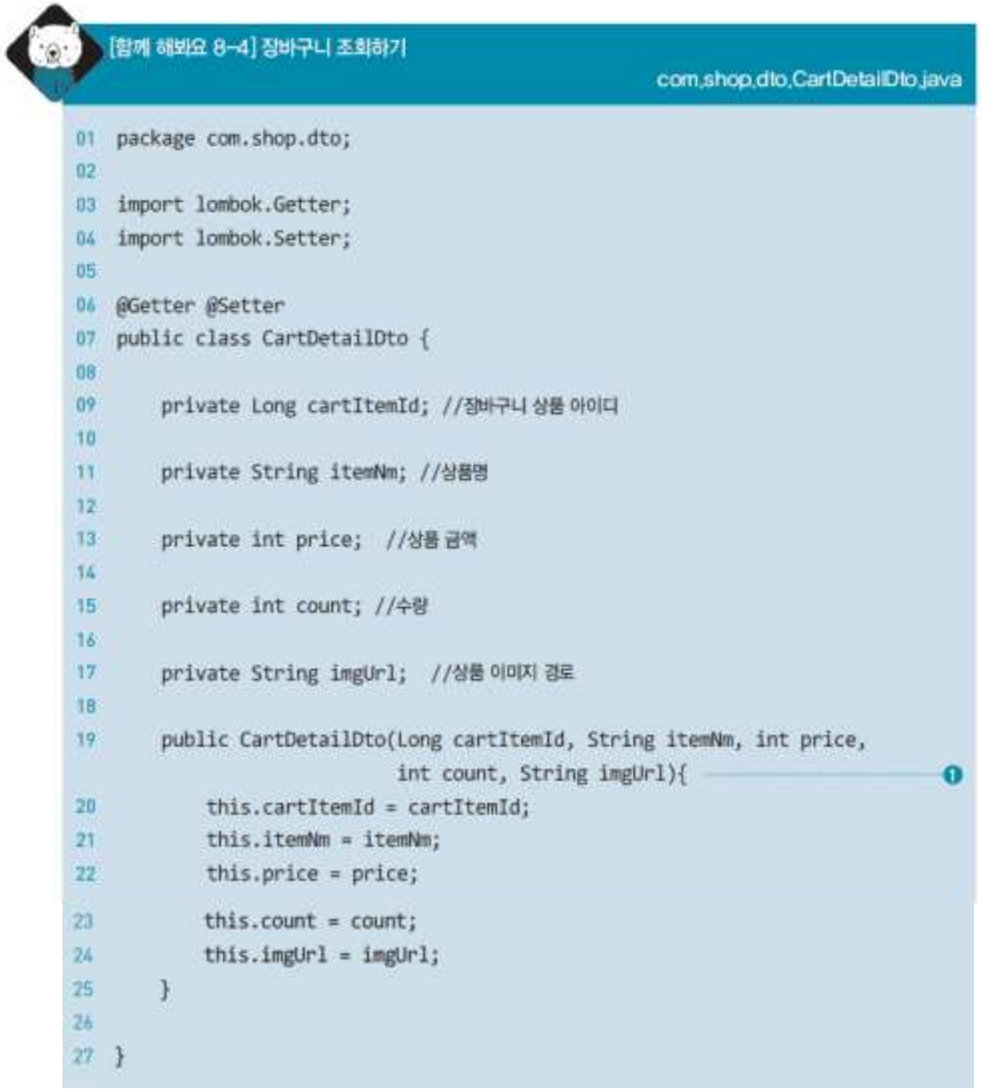
8.1 장바구니 담기


localhost의 메시지

상품을 장바구니에 담았습니다.

[그림 8-2] 상품 장바구니 담기 성공

8.2 장바구니 조회



[함께 해봐요 8-4] 장바구니 조회하기

com.shop.dto.CartDetailDto.java

```java
package com.shop.dto;

import lombok.Getter;
import lombok.Setter;

@Getter @Setter
public class CartDetailDto {

    private Long cartItemId; //장바구니 상품 아이디

    private String itemNm; //상품명

    private int price;  //상품 금액

    private int count; //수량

    private String imgUrl;  //상품 이미지 경로

    public CartDetailDto(Long cartItemId, String itemNm, int price,
                         int count, String imgUrl){        ❶
        this.cartItemId = cartItemId;
        this.itemNm = itemNm;
        this.price = price;
        this.count = count;
        this.imgUrl = imgUrl;
    }

}
```

8.2 장바구니 조회

- 최적화가 필요할 경우 아래 코드와 같이 DTO의 생성자를 이용하여 반환 값으로 DTO 객체를 생성하여 조회

com.shop.repository.CartItemRepository.java

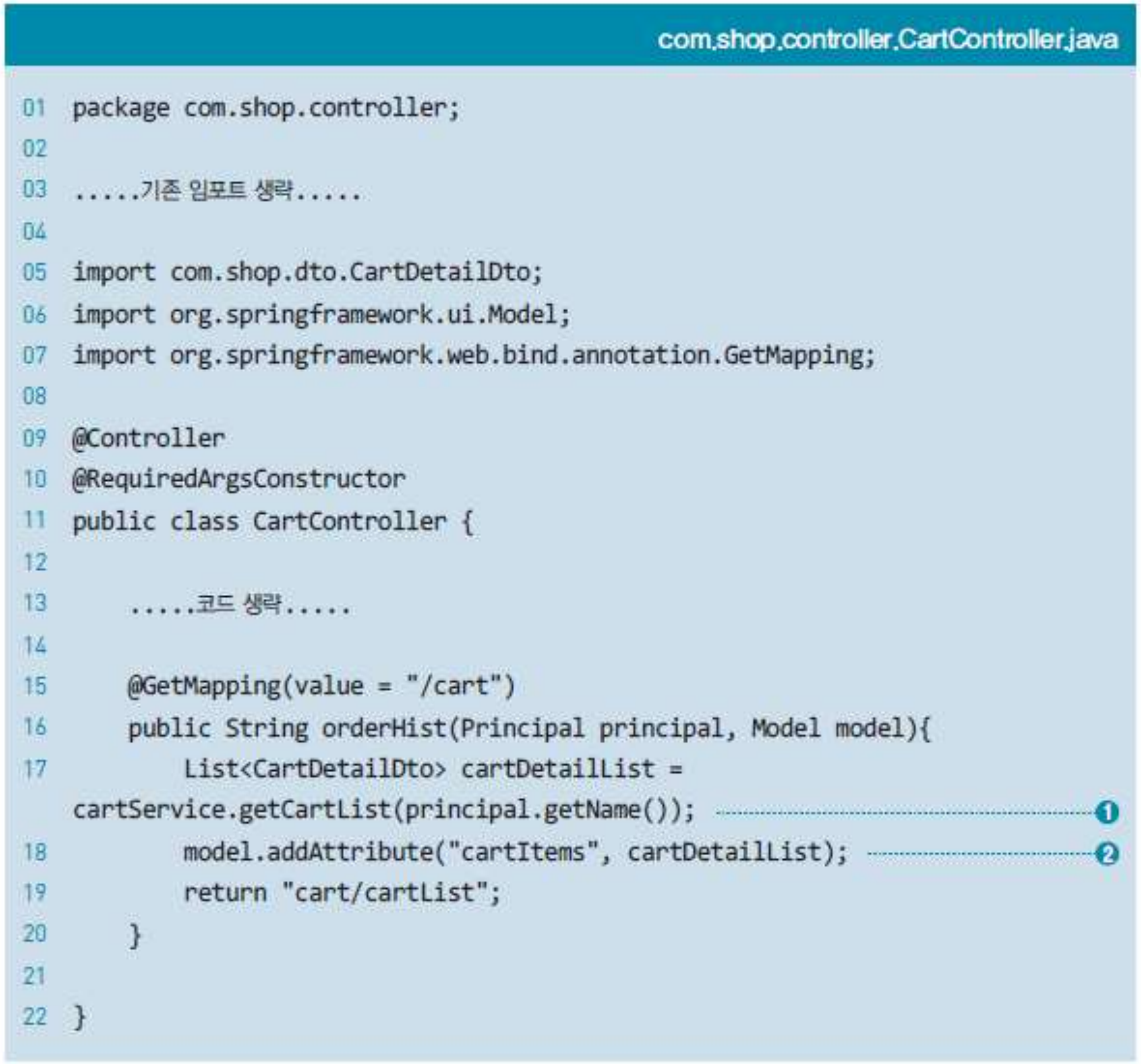
```java
package com.shop.repository;

.....기존 임포트 생략.....

import com.shop.dto.CartDetailDto;
import org.springframework.data.jpa.repository.Query;
import java.util.List;

public interface CartItemRepository extends JpaRepository<CartItem, Long> {

    .....코드 생략.....

    @Query("select new com.shop.dto.CartDetailDto(ci.id, i.itemNm, i.price,
ci.count, im.imgUrl) " + ❶
            "from CartItem ci, ItemImg im " +
            "join ci.item i " +
            "where ci.cart.id = :cartId " +
            "and im.item.id = ci.item.id " + ❷
            "and im.repimgYn = 'Y' " + ❸
            "order by ci.regTime desc"
    )
    List<CartDetailDto> findCartDetailDtoList(Long cartId);

}
```

8.2 장바구니 조회

com.shop.service.CartService.java

```java
package com.shop.service;

.....기존 임포트 생략.....

import com.shop.dto.CartDetailDto;
import java.util.ArrayList;
import java.util.List;

@Service
@RequiredArgsConstructor
@Transactional
public class CartService {

    .....코드 생략.....

    @Transactional(readOnly = true)
    public List<CartDetailDto> getCartList(String email){

        List<CartDetailDto> cartDetailDtoList = new ArrayList<>();

        Member member = memberRepository.findByEmail(email);
        Cart cart = cartRepository.findByMemberId(member.getId()); ❶
        if(cart == null){ ❷
            return cartDetailDtoList;
        }

        cartDetailDtoList =
cartItemRepository.findCartDetailDtoList(cart.getId()); ❸

        return cartDetailDtoList;
    }

}
```

8.2 장바구니 조회

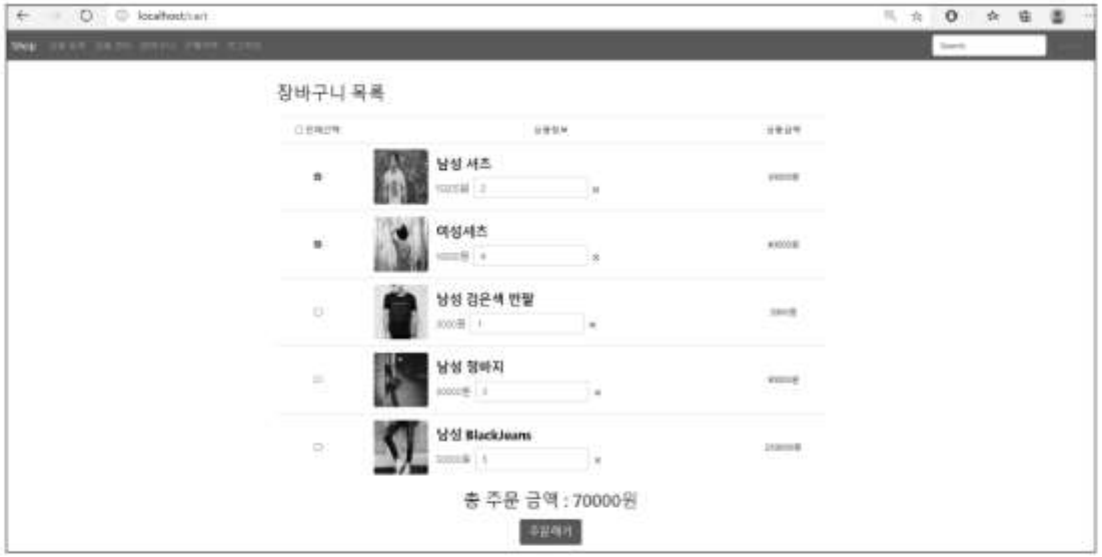


com.shop.controller.CartController.java

```java
package com.shop.controller;

.....기존 임포트 생략.....

import com.shop.dto.CartDetailDto;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.GetMapping;

@Controller
@RequiredArgsConstructor
public class CartController {

    .....코드 생략.....

    @GetMapping(value = "/cart")
    public String orderHist(Principal principal, Model model){
        List<CartDetailDto> cartDetailList =
cartService.getCartList(principal.getName()); ❶
        model.addAttribute("cartItems", cartDetailList); ❷
        return "cart/cartList";
    }

}
```

8.2 장바구니 조회

- 구현할 기능 리스트
  - 장바구니 상품 선택 시 총 주문 금액 계산
  - 버튼 클릭 시 장바구니에 담긴 상품 삭제
  - 장바구니 상품 수량 변경 시 상품 금액 계산
  - 장바구니 상품 수량 변경 시 장바구니에 담긴 상품 수량 업데이트
  - 장바구니 상품 주문하기
  - 장바구니 상품 전체 선택


localhost/cart

장바구니 목록

☐ 전체선택	상품정보	상품금액
■	남성 셔츠 [illegible] 2 ×	[illegible]
■	여성셔츠 [illegible] 4 ×	[illegible]
☐	남성 검은색 반팔 [illegible] 1 ×	[illegible]
☐	남성 청바지 [illegible] ×	[illegible]
☐	남성 BlackJeans [illegible] 1 ×	[illegible]

총 주문 금액 : 70000원

[그림 8-3] 장바구니 목록

8.2 장바구니 조회

resources/templates/cart/cartList.html

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org"
      xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
      layout:decorate="~{layouts/layout1}">

<head>
    <meta name="_csrf" th:content="${_csrf.token}"/>
    <meta name="_csrf_header" th:content="${_csrf.headerName}"/>
</head>

<!-- 사용자 스크립트 추가 -->
<th:block layout:fragment="script">

    <script th:inline="javascript">

        $(document).ready(function(){
            $("input[name=cartChkBox]").change( function(){
                getOrderTotalPrice();
            });
        });
```

8.2 장바구니 조회

```
        function getOrderTotalPrice(){
            var orderTotalPrice = 0;
            $("input[name=cartChkBox]:checked").each(function() {
                var cartItemId = $(this).val();
                var price = $("#price_" + cartItemId).attr("data-price");
                var count = $("#count_" + cartItemId).val();
                orderTotalPrice += price*count;
            });

            $("#orderTotalPrice").html(orderTotalPrice+'원');
        }

        function changeCount(obj){
            var count = obj.value;
            var cartItemId = obj.id.split('_')[1];
            var price = $("#price_" + cartItemId).data("price");
            var totalPrice = count*price;
            $("#totalPrice_" + cartItemId).html(totalPrice+"원");
            getOrderTotalPrice();
        }
```

8.2 장바구니 조회

```
        function checkAll(){
            if($("#checkall").prop("checked")){
                $("input[name=cartChkBox]").prop("checked",true);
            }else{
                $("input[name=cartChkBox]").prop("checked",false);
            }
            getOrderTotalPrice();
        }

    </script>

</th:block>

<!-- 사용자 CSS 추가 -->
<th:block layout:fragment="css">
    <style>
        .content-mg{
            margin-left:25%;
            margin-right:25%;
            margin-top:2%;
            margin-bottom:100px;
        }
```

8.2 장바구니 조회

```
        .repImgDiv{
            margin-right:15px;
            margin-left:15px;
            height:auto;
        }
        .repImg{
            height:100px;
            width:100px;
        }
        .fs18{
            font-size:18px
        }
        .fs24{
            font-size:24px
        }
    </style>
</th:block>
<div layout:fragment="content" class="content-mg">

    <h2 class="mb-4">
        장바구니 목록
    </h2>

    <div>
```

8.2 장바구니 조회

```
        <table class="table">
            <colgroup>
                <col width="15%"/>
                <col width="70%"/>
                <col width="15%"/>
            </colgroup>
            <thead>
            <tr class="text-center">
                <td>
                    <input type="checkbox" id="checkall"
                           onclick="checkAll()"> 전체선택
                </td>
                <td>상품정보</td>
                <td>상품금액</td>
            </tr>
            </thead>
            <tbody>
            <tr th:each="cartItem : ${cartItems}">
                <td class="text-center align-middle">
                    <input type="checkbox" name="cartChkBox"
                           th:value="${cartItem.cartItemId}">
                </td>
                <td class="d-flex">
                    <div class="repImgDiv align-self-center">
                        <img th:src="${cartItem.imgUrl}"
class = "rounded repImg" th:alt="${cartItem.itemNm}">
                    </div>
```

8.2 장바구니 조회

```
                <div class="align-self-center">
                    <span th:text="${cartItem.itemNm}" class="fs24 font-weight-bold"></span>
                    <div class="fs18 font-weight-light">
                        <span class="input-group mt-2">
                            <span th:id="'price_' + ${cartItem.cartItemId}"
                                  th:data-price="${cartItem.price}"
                                  th:text="${cartItem.price} + '원'" class="align-self-center mr-2">
                            </span>
                            <input type="number" name="count" th:id="'count_' + ${cartItem.cartItemId}"
                                   th:value="${cartItem.count}" min="1"
                                   onchange="changeCount(this)" class="form-control mr-2" >
                            <button type="button" class="close" aria-label="Close">
                                <span aria-hidden="true" th:data-id="${cartItem.cartItemId}">&times;</span>
                            </button>
                        </span>
                    </div>
                </div>
            </td>
            <td class="text-center align-middle">
                <span th:id="'totalPrice_' + ${cartItem.cartItemId}"
                   name="totalPrice" th:text="${cartItem.price * cartItem.count} + '원'">
                </span>
            </td>
        </tr>
        </tbody>
    </table>
```

8.2 장바구니 조회

```
        <h2 class="text-center">
            총 주문 금액 : <span id="orderTotalPrice" class="text-danger">0원</span>
        </h2>

        <div class="text-center mt-3">
            <button type="button" class="btn btn-primary btn-lg">주문하기</button>
        </div>

    </div>

</div>

</html>
```

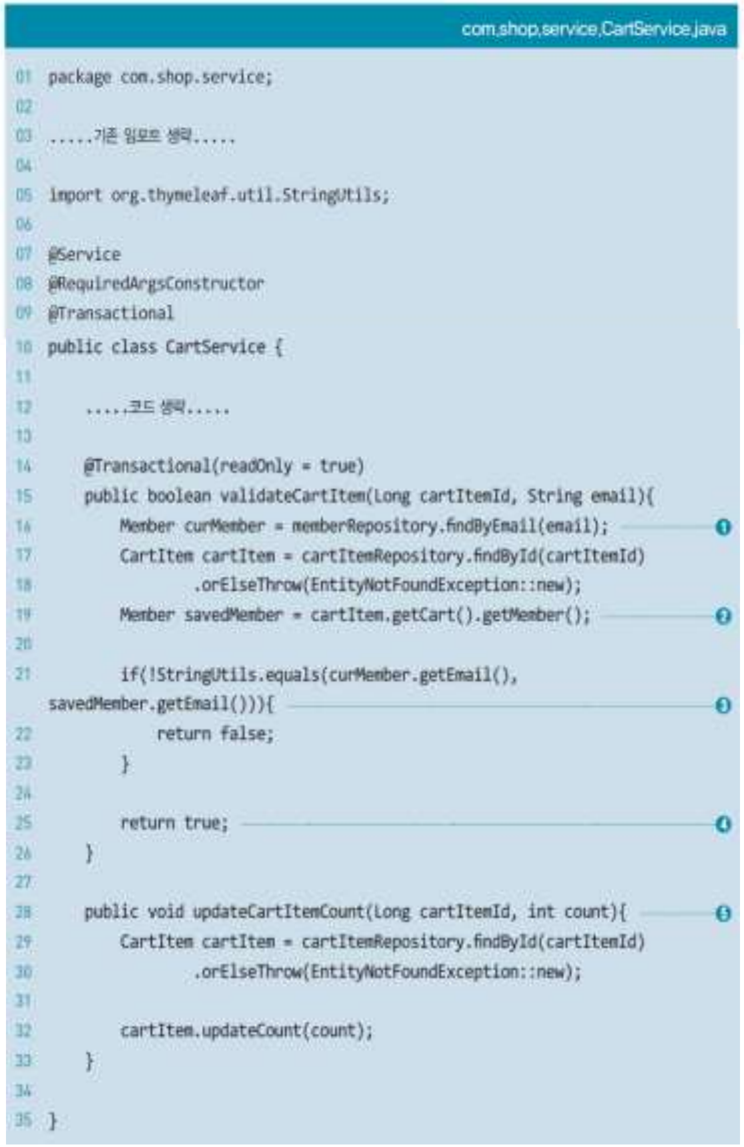
8.2 장바구니 조회

[함께 해봐요 8-5] 장바구니 상품 수량 변경하기

com.shop.entity.CartItem.java

```java
package com.shop.entity;

.....기존 임포트 생략.....

@Entity
@Getter @Setter
@Table(name="cart_item")
public class CartItem extends BaseEntity{

    .....코드 생략.....

    public void updateCount(int count){
        this.count = count;
    }

}
```

8.2 장바구니 조회



com.shop.service.CartService.java

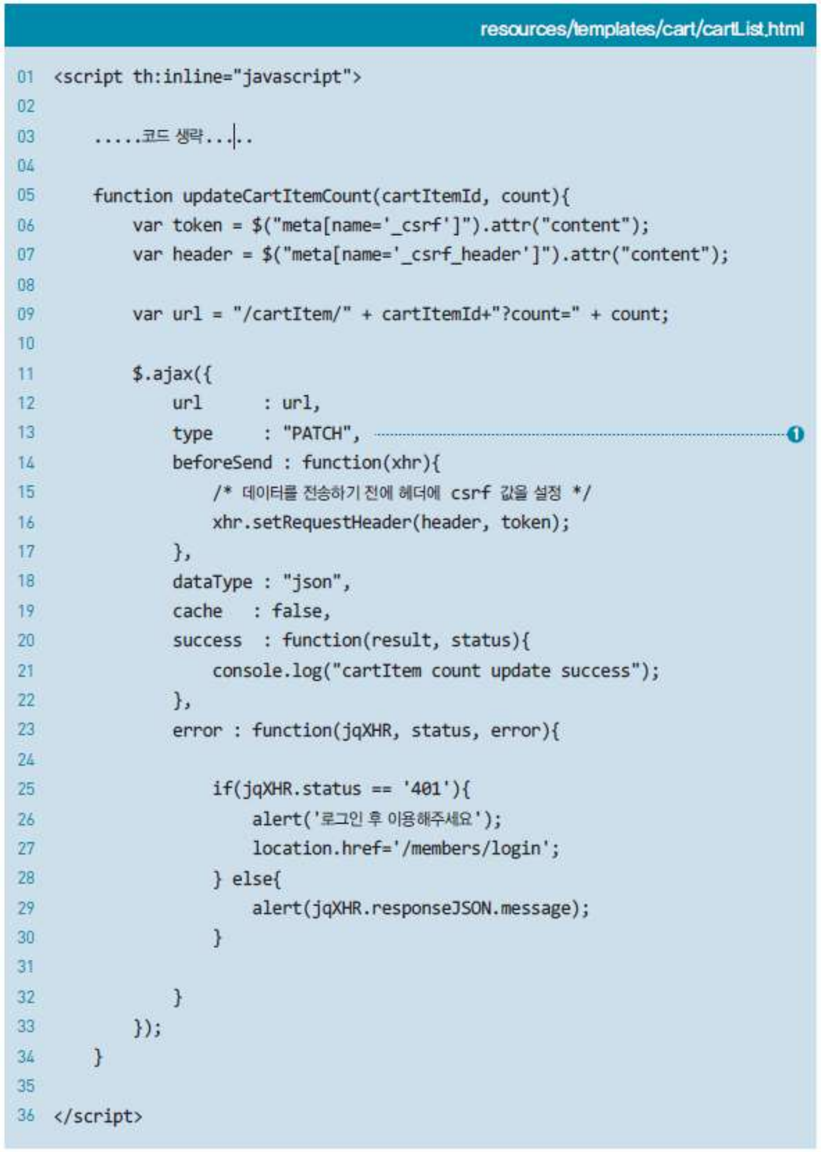
```java
package com.shop.service;

.....기존 임포트 생략.....

import org.thymeleaf.util.StringUtils;

@Service
@RequiredArgsConstructor
@Transactional
public class CartService {

    .....코드 생략.....

    @Transactional(readOnly = true)
    public boolean validateCartItem(Long cartItemId, String email){
        Member curMember = memberRepository.findByEmail(email);          ❶
        CartItem cartItem = cartItemRepository.findById(cartItemId)
                .orElseThrow(EntityNotFoundException::new);
        Member savedMember = cartItem.getCart().getMember();             ❷

        if(!StringUtils.equals(curMember.getEmail(),
savedMember.getEmail())){                                                ❸
            return false;
        }

        return true;                                                     ❹
    }

    public void updateCartItemCount(Long cartItemId, int count){        ❺
        CartItem cartItem = cartItemRepository.findById(cartItemId)
                .orElseThrow(EntityNotFoundException::new);

        cartItem.updateCount(count);
    }

}
```

8.2 장바구니 조회

com.shop.controller.CartController.java

```java
package com.shop.controller;

.....기존 임포트 생략.....

import org.springframework.web.bind.annotation.PatchMapping;
import org.springframework.web.bind.annotation.PathVariable;

@Controller
@RequiredArgsConstructor
public class CartController {

    .....코드 생략.....

    @PatchMapping(value = "/cartItem/{cartItemId}") ❶
    public @ResponseBody ResponseEntity updateCartItem
(@PathVariable("cartItemId") Long cartItemId, int count, Principal principal){

        if(count <= 0){ ❷
            return new ResponseEntity<String>
                    ("최소 1개 이상 담아주세요", HttpStatus.BAD_REQUEST);
        } else if(!cartService.validateCartItem
                (cartItemId, principal.getName())){ ❸
            return new ResponseEntity<String>
                    ("수정 권한이 없습니다.", HttpStatus.FORBIDDEN);
        }

        cartService.updateCartItemCount(cartItemId, count); ❹
        return new ResponseEntity<Long>(cartItemId, HttpStatus.OK);
    }

}
```

8.2 장바구니 조회

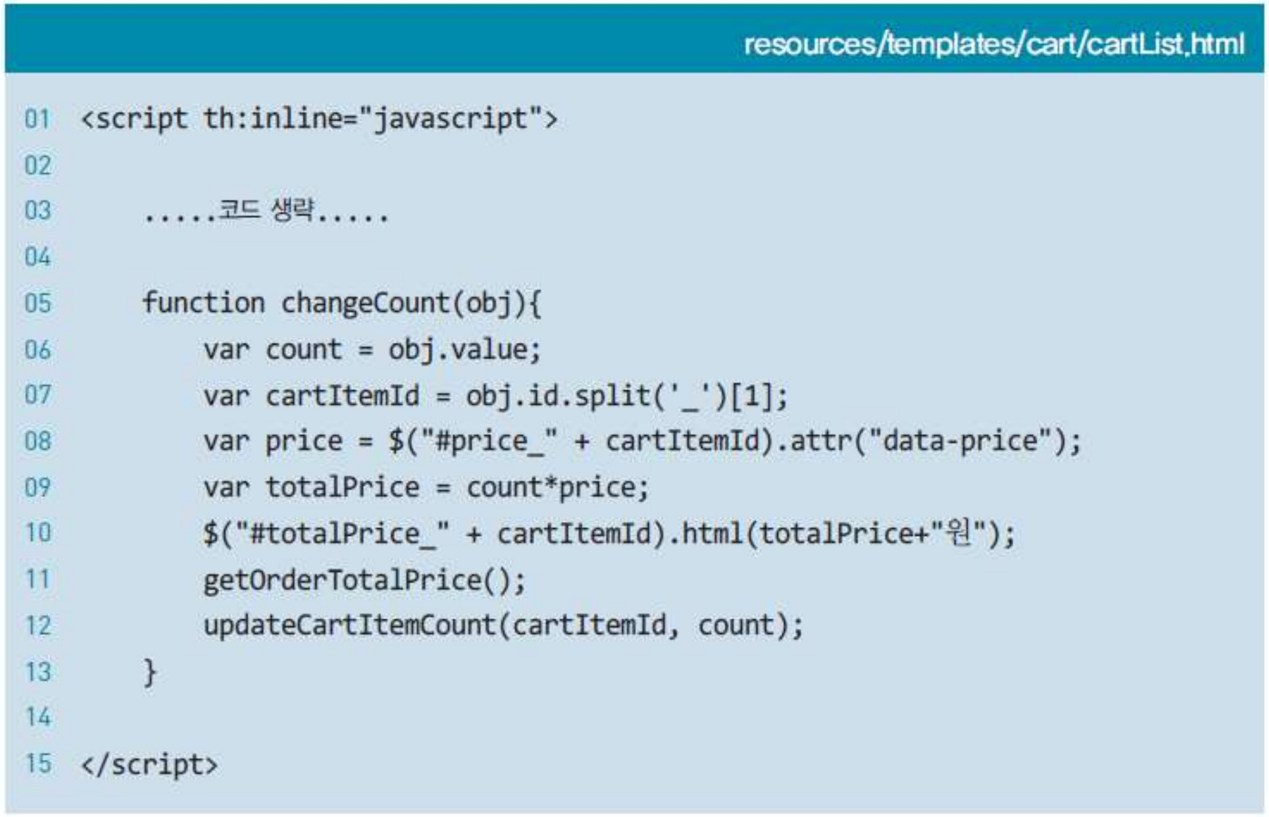


resources/templates/cart/cartList.html

```
<script th:inline="javascript">

    .....코드 생략.....

    function updateCartItemCount(cartItemId, count){
        var token = $("meta[name='_csrf']").attr("content");
        var header = $("meta[name='_csrf_header']").attr("content");

        var url = "/cartItem/" + cartItemId+"?count=" + count;

        $.ajax({
            url      : url,
            type     : "PATCH",  ❶
            beforeSend : function(xhr){
                /* 데이터를 전송하기 전에 헤더에 csrf 값을 설정 */
                xhr.setRequestHeader(header, token);
            },
            dataType : "json",
            cache   : false,
            success  : function(result, status){
                console.log("cartItem count update success");
            },
            error : function(jqXHR, status, error){

                if(jqXHR.status == '401'){
                    alert('로그인 후 이용해주세요');
                    location.href='/members/login';
                } else{
                    alert(jqXHR.responseJSON.message);
                }

            }
        });
    }

</script>
```

8.2 장바구니 조회

- 작성한 updateCartItemCount() 함수를 기존에 작성한 changeCount() 함수 마지막에 추가

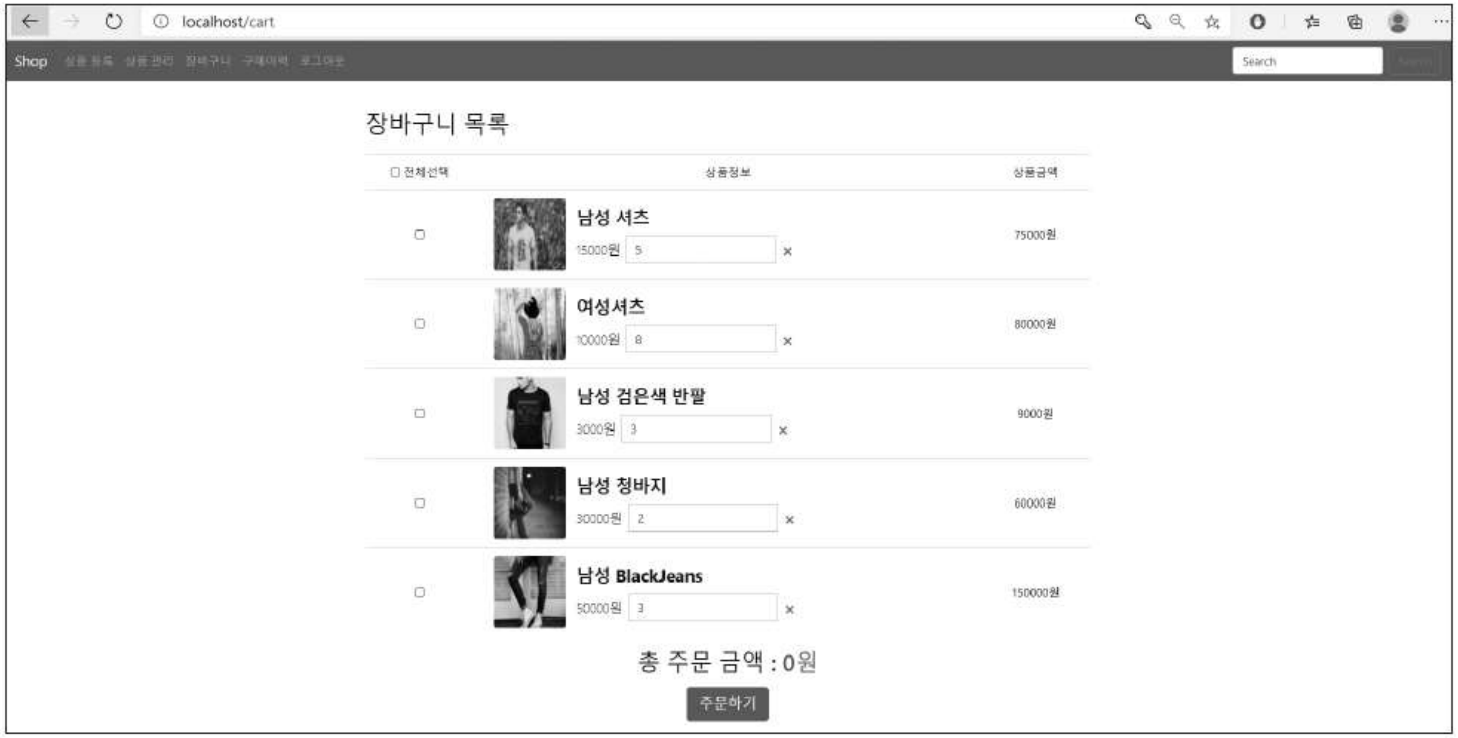


resources/templates/cart/cartList.html

```
<script th:inline="javascript">

    .....코드 생략.....

    function changeCount(obj){
        var count = obj.value;
        var cartItemId = obj.id.split('_')[1];
        var price = $("#price_" + cartItemId).attr("data-price");
        var totalPrice = count*price;
        $("#totalPrice_" + cartItemId).html(totalPrice+"원");
        getOrderTotalPrice();
        updateCartItemCount(cartItemId, count);
    }

</script>
```

8.2 장바구니 조회

- 장바구니 상품의 수량을 변경 후 페이지 새로고침시 변경된 수량 유지



localhost/cart

Shop 상품 등록 상품 관리 장바구니 구매이력 로그아웃

장바구니 목록

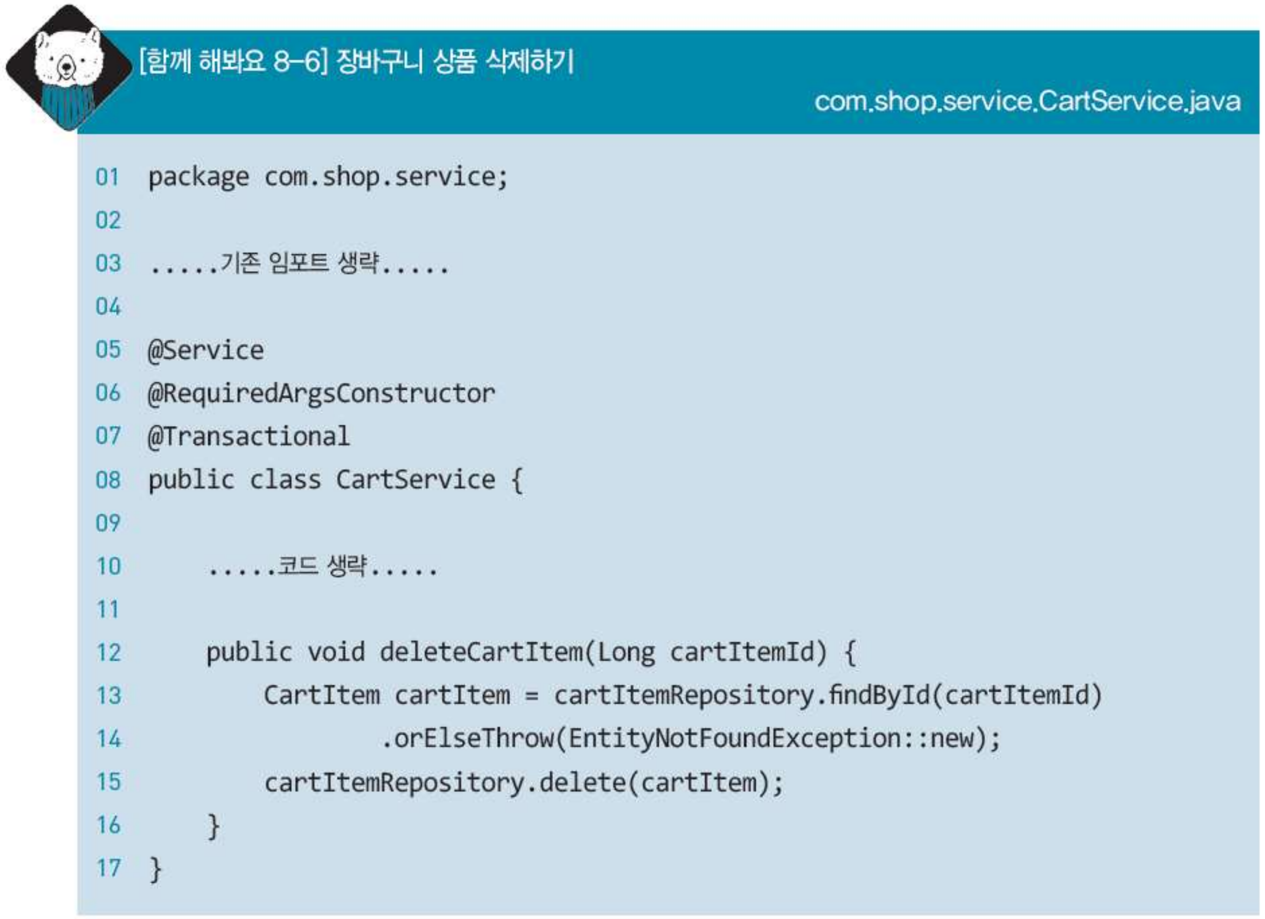
☐ 전체선택	상품정보	상품금액
☐	남성 셔츠 15000원 5 ×	75000원
☐	여성셔츠 10000원 8 ×	80000원
☐	남성 검은색 반팔 3000원 3 ×	9000원
☐	남성 청바지 30000원 2 ×	60000원
☐	남성 BlackJeans 50000원 3 ×	150000원

총 주문 금액 : 0원

[그림 8-4] 장바구니 목록

8.2 장바구니 조회

- 상품정보에 있는 버튼을 클릭할 때 장바구니에 넣어 놓은 상품을 삭제

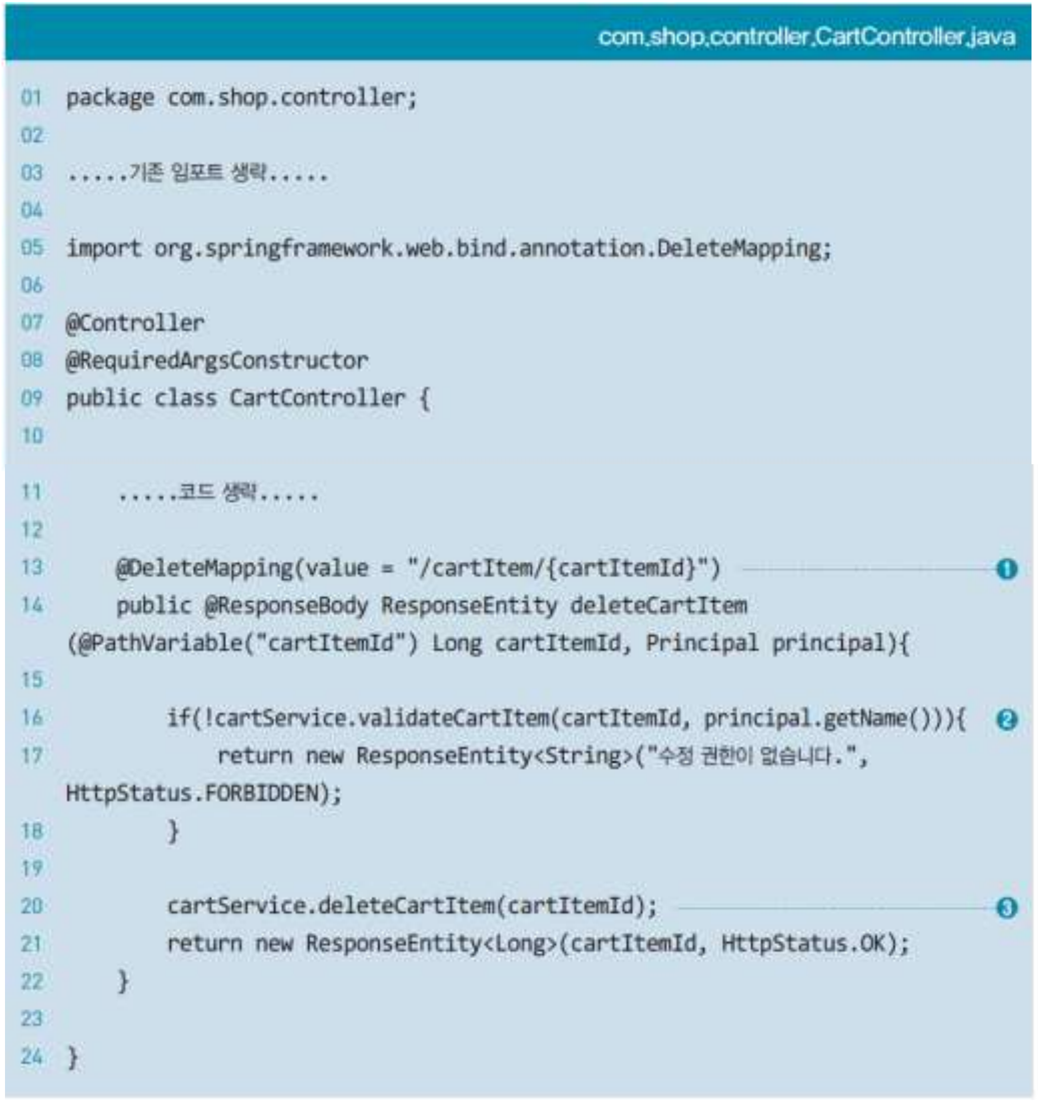


[함께 해봐요 8-6] 장바구니 상품 삭제하기

com.shop.service.CartService.java

```java
package com.shop.service;

.....기존 임포트 생략.....

@Service
@RequiredArgsConstructor
@Transactional
public class CartService {

    .....코드 생략.....

    public void deleteCartItem(Long cartItemId) {
        CartItem cartItem = cartItemRepository.findById(cartItemId)
                .orElseThrow(EntityNotFoundException::new);
        cartItemRepository.delete(cartItem);
    }
}
```

8.2 장바구니 조회



com.shop.controller.CartController.java

```
package com.shop.controller;

.....기존 임포트 생략.....

import org.springframework.web.bind.annotation.DeleteMapping;

@Controller
@RequiredArgsConstructor
public class CartController {

    .....코드 생략.....

    @DeleteMapping(value = "/cartItem/{cartItemId}")                        ❶
    public @ResponseBody ResponseEntity deleteCartItem (@PathVariable("cartItemId") Long cartItemId, Principal principal){

        if(!cartService.validateCartItem(cartItemId, principal.getName())){   ❷
            return new ResponseEntity<String>("수정 권한이 없습니다.", HttpStatus.FORBIDDEN);
        }

        cartService.deleteCartItem(cartItemId);                               ❸
        return new ResponseEntity<Long>(cartItemId, HttpStatus.OK);
    }

}
```

8.2 장바구니 조회

(resources/templates/cart/cartList.html)

```
<script th:inline="javascript">

    .....코드 생략.....

    function deleteCartItem(obj){
        var cartItemId = obj.dataset.id;
        var token = $("meta[name='_csrf']").attr("content");
        var header = $("meta[name='_csrf_header']").attr("content");

        var url = "/cartItem/" + cartItemId;

        $.ajax({
            url      : url,
            type     : "DELETE",                                    ❶
            beforeSend : function(xhr){
```

8.2 장바구니 조회

```
                /* 데이터를 전송하기 전에 헤더에 csrf 값을 설정 */
                xhr.setRequestHeader(header, token);
            },
            dataType : "json",
            cache   : false,
            success  : function(result, status){
                location.href='/cart'; ❷
            },
            error : function(jqXHR, status, error){

                if(jqXHR.status == '401'){
                    alert('로그인 후 이용해주세요');
                    location.href='/members/login';
                } else{
                    alert(jqXHR.responseJSON.message);
                }

            }
        });
    }
</script>
```

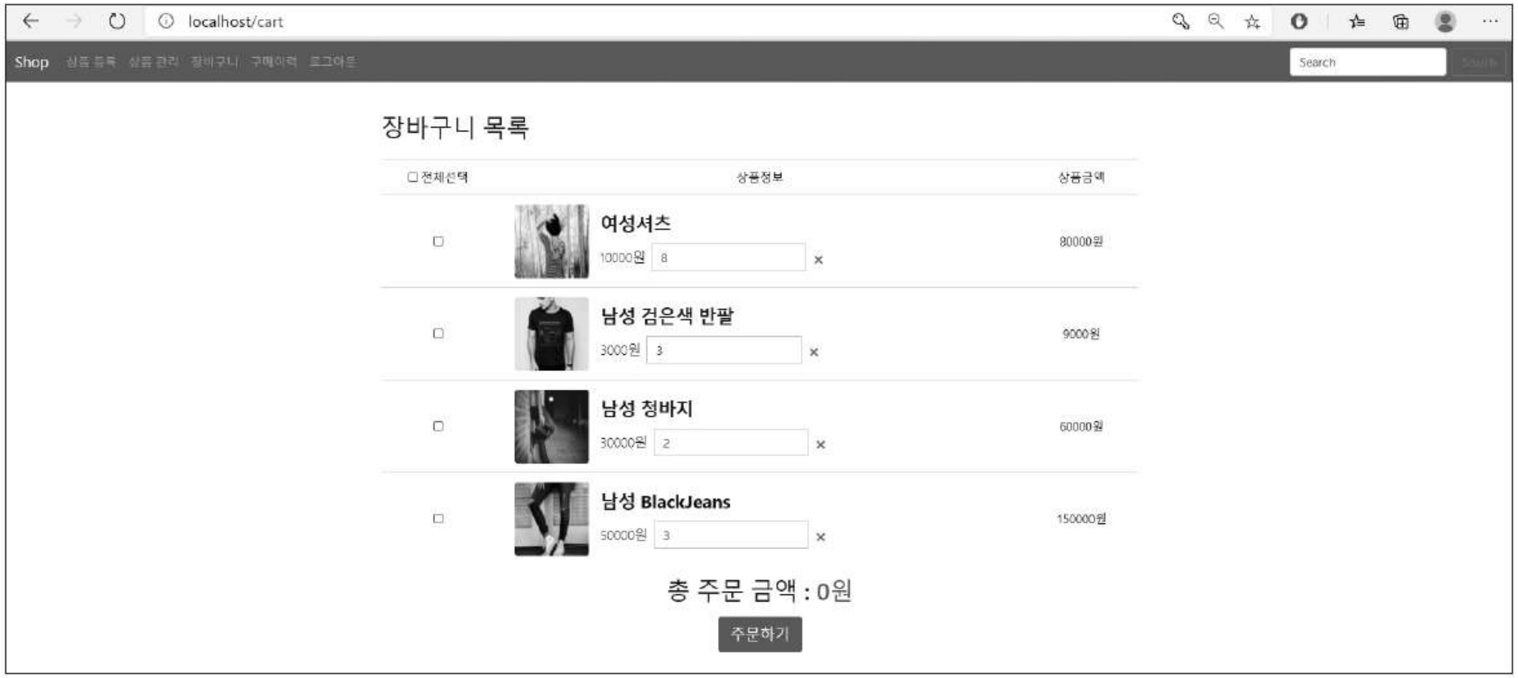
8.2 장바구니 조회

- <삭제> 버튼을 누르면 deleteCartItem() 함수가 호출되도록 onclick 속성을 추가

resources/templates/cart/cartList.html

```
<button type="button" class="close" aria-label="Close">
        <span aria-hidden="true" th:data-id="${cartItem.cartItemId}" onclick="deleteCartItem(this)">&times;</span>
</button>
```

8.2 장바구니 조회



localhost/cart

장바구니 목록

☐ 전체선택	상품정보	상품금액
☐	여성셔츠 10000원 8 ×	80000원
☐	남성 검은색 반팔 3000원 3 ×	9000원
☐	남성 청바지 30000원 2 ×	60000원
☐	남성 BlackJeans 50000원 3 ×	150000원

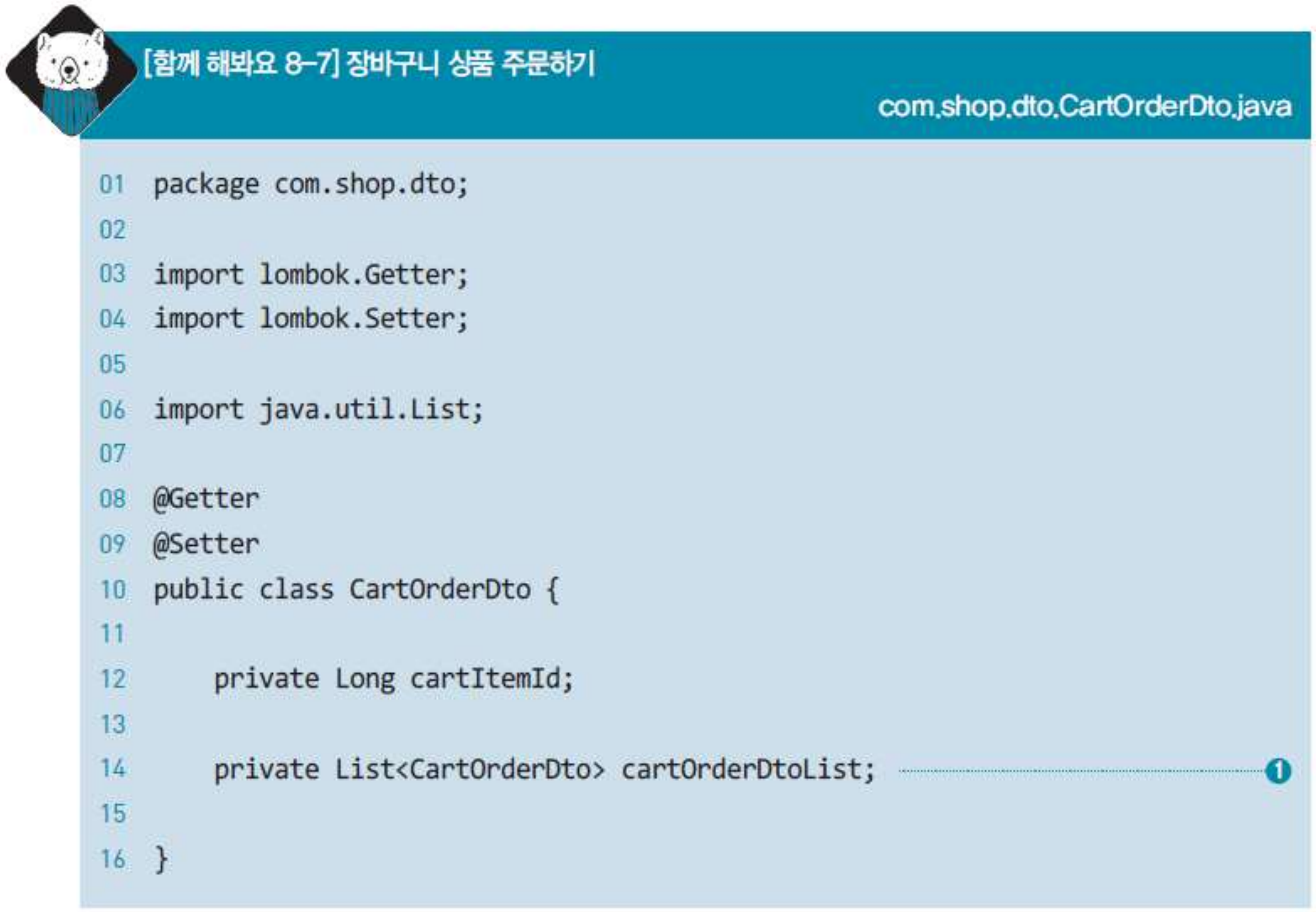
총 주문 금액 : 0원

주문하기

[그림 8-5] 장바구니 상품 삭제 후 목록 확인

8.3 장바구니 상품 주문

- 장바구니에서 주문을 하면 기존 주문 로직과의 차이점은 여러 개의 상품을 하나의 주문에 담을 수 있음
-

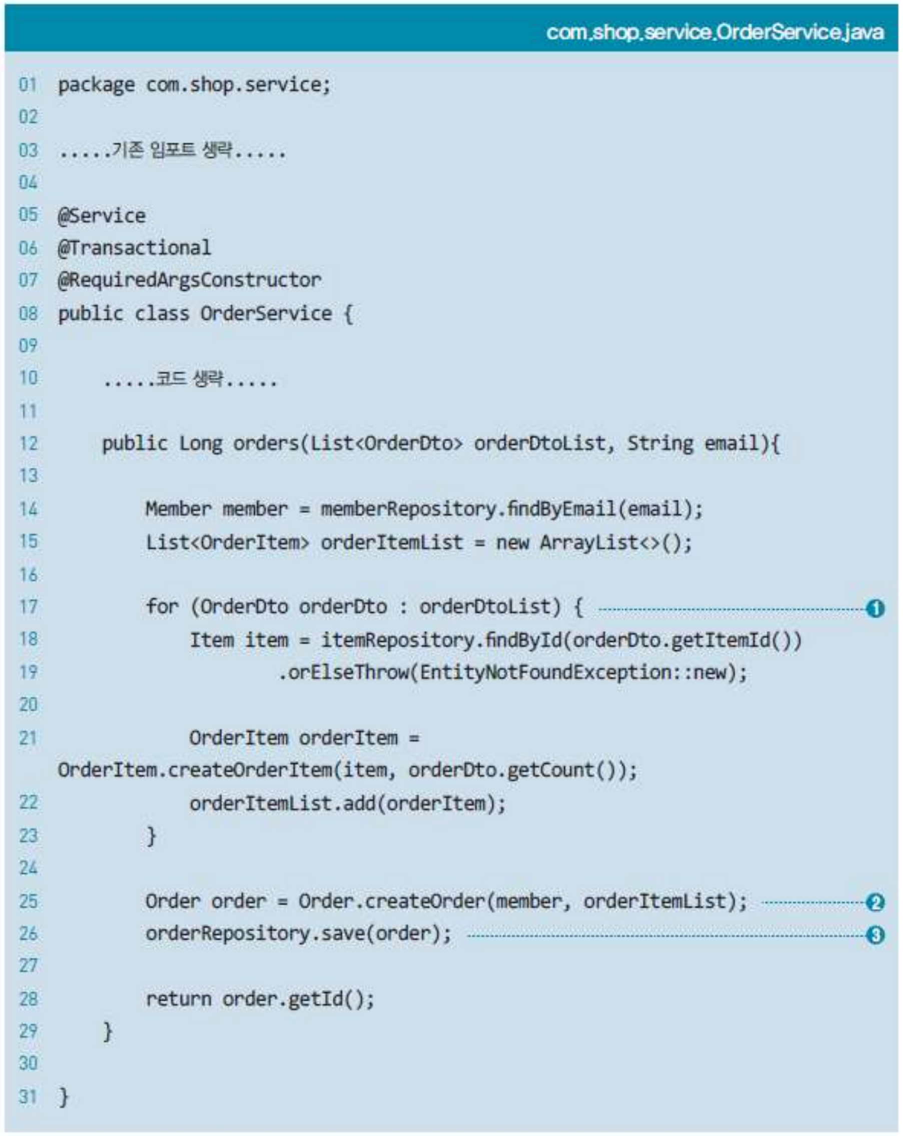


[함께 해봐요 8-7] 장바구니 상품 주문하기

com.shop.dto.CartOrderDto.java

```java
package com.shop.dto;

import lombok.Getter;
import lombok.Setter;

import java.util.List;

@Getter
@Setter
public class CartOrderDto {

    private Long cartItemId;

    private List<CartOrderDto> cartOrderDtoList; ❶

}
```

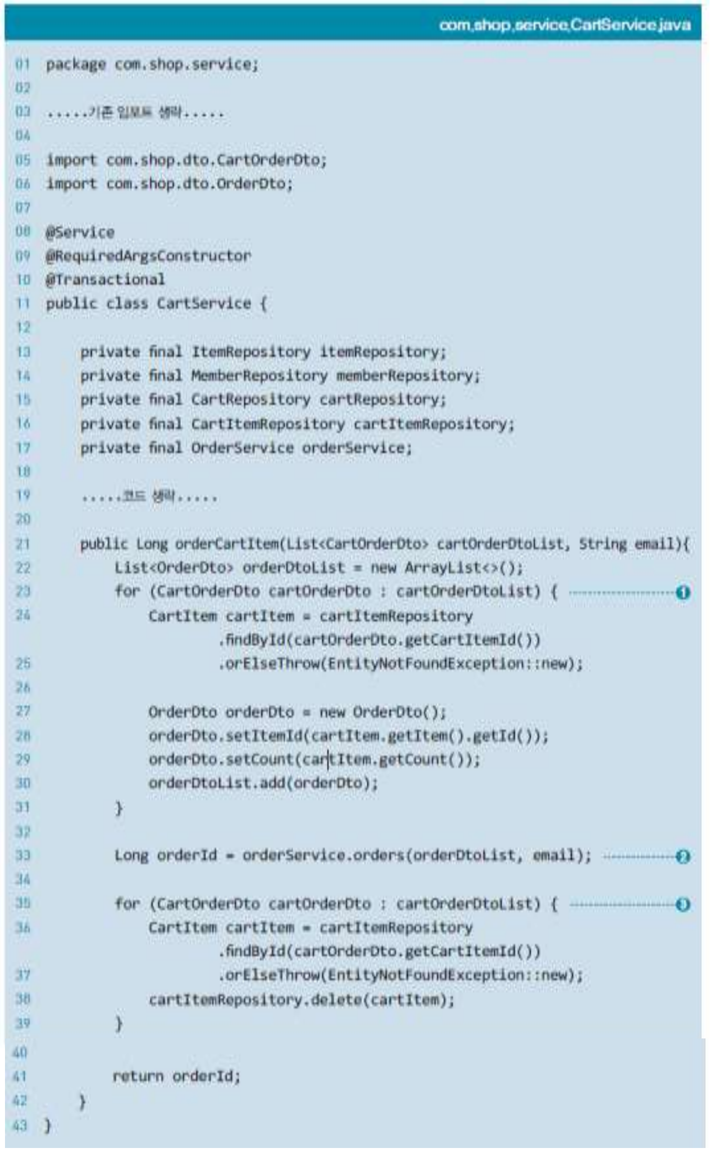
8.3 장바구니 상품 주문



com.shop.service.OrderService.java

```
package com.shop.service;

.....기존 임포트 생략.....

@Service
@Transactional
@RequiredArgsConstructor
public class OrderService {

    .....코드 생략.....

    public Long orders(List<OrderDto> orderDtoList, String email){

        Member member = memberRepository.findByEmail(email);
        List<OrderItem> orderItemList = new ArrayList<>();

        for (OrderDto orderDto : orderDtoList) { ························ ❶
            Item item = itemRepository.findById(orderDto.getItemId())
                    .orElseThrow(EntityNotFoundException::new);

            OrderItem orderItem =
OrderItem.createOrderItem(item, orderDto.getCount());
            orderItemList.add(orderItem);
        }

        Order order = Order.createOrder(member, orderItemList); ········· ❷
        orderRepository.save(order); ·································· ❸

        return order.getId();
    }

}
```

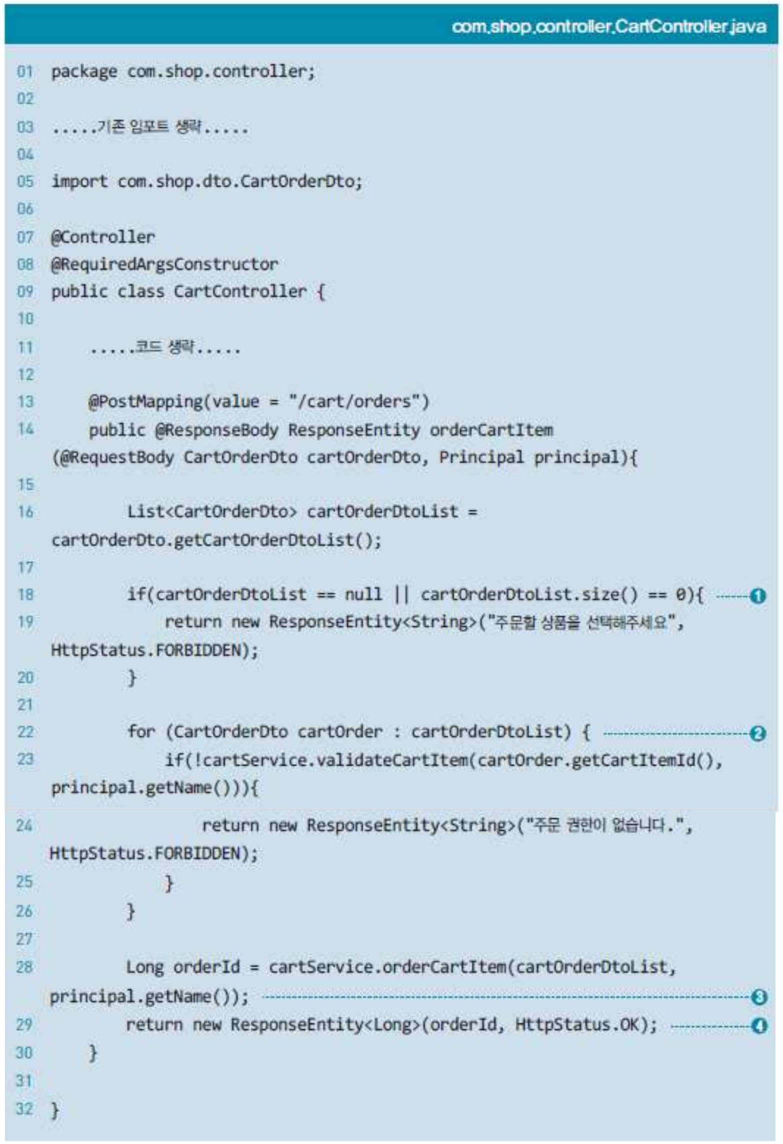
8.3 장바구니 상품 주문



com.shop.service.CartService.java

```java
package com.shop.service;

.....기존 임포트 생략.....

import com.shop.dto.CartOrderDto;
import com.shop.dto.OrderDto;

@Service
@RequiredArgsConstructor
@Transactional
public class CartService {

    private final ItemRepository itemRepository;
    private final MemberRepository memberRepository;
    private final CartRepository cartRepository;
    private final CartItemRepository cartItemRepository;
    private final OrderService orderService;

    .....코드 생략.....

    public Long orderCartItem(List<CartOrderDto> cartOrderDtoList, String email){
        List<OrderDto> orderDtoList = new ArrayList<>();
        for (CartOrderDto cartOrderDto : cartOrderDtoList) {  ············❶
            CartItem cartItem = cartItemRepository
                    .findById(cartOrderDto.getCartItemId())
                    .orElseThrow(EntityNotFoundException::new);

            OrderDto orderDto = new OrderDto();
            orderDto.setItemId(cartItem.getItem().getId());
            orderDto.setCount(cartItem.getCount());
            orderDtoList.add(orderDto);
        }

        Long orderId = orderService.orders(orderDtoList, email);  ············❷

        for (CartOrderDto cartOrderDto : cartOrderDtoList) {  ············❸
            CartItem cartItem = cartItemRepository
                    .findById(cartOrderDto.getCartItemId())
                    .orElseThrow(EntityNotFoundException::new);
            cartItemRepository.delete(cartItem);
        }

        return orderId;
    }
}
```

8.3 장바구니 상품 주문



com.shop.controller.CartController.java

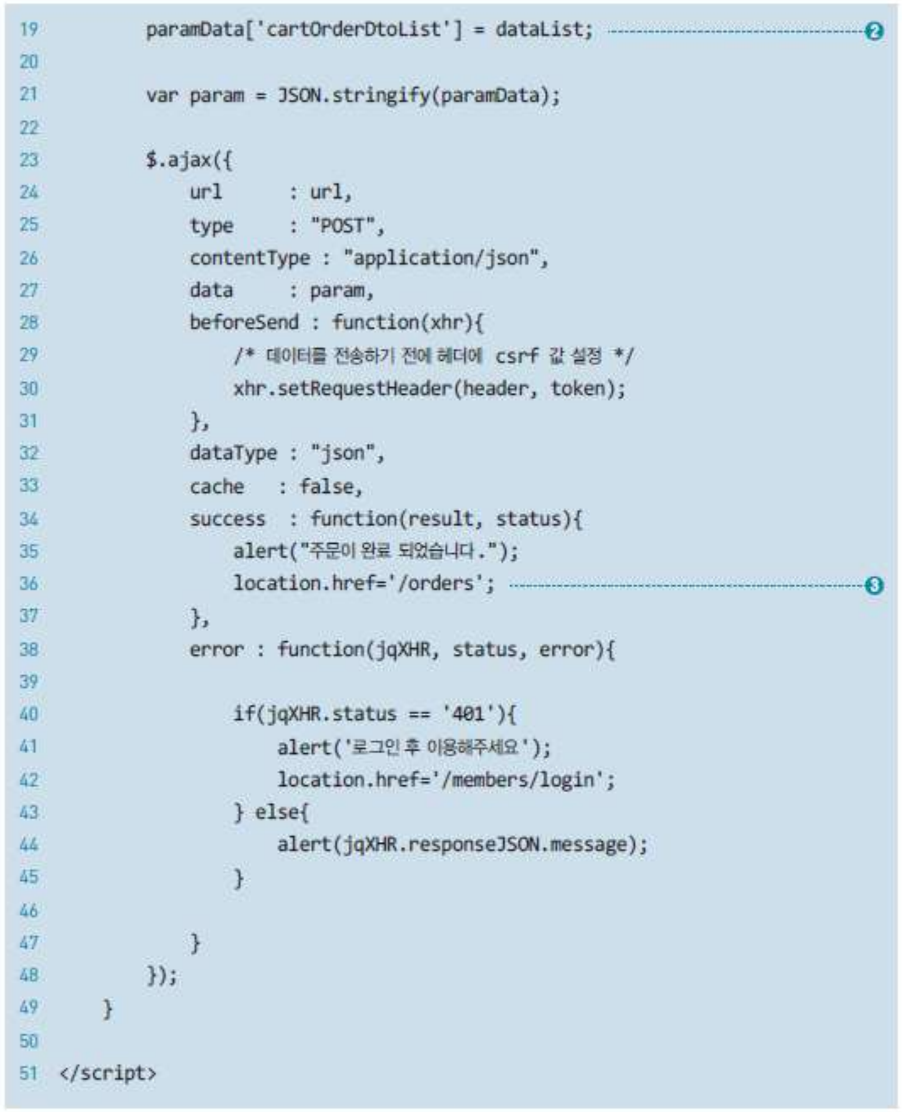
```java
package com.shop.controller;

.....기존 임포트 생략.....

import com.shop.dto.CartOrderDto;

@Controller
@RequiredArgsConstructor
public class CartController {

    .....코드 생략.....

    @PostMapping(value = "/cart/orders")
    public @ResponseBody ResponseEntity orderCartItem
(@RequestBody CartOrderDto cartOrderDto, Principal principal){

        List<CartOrderDto> cartOrderDtoList =
cartOrderDto.getCartOrderDtoList();

        if(cartOrderDtoList == null || cartOrderDtoList.size() == 0){ ······❶
            return new ResponseEntity<String>("주문할 상품을 선택해주세요",
HttpStatus.FORBIDDEN);
        }

        for (CartOrderDto cartOrder : cartOrderDtoList) { ······❷
            if(!cartService.validateCartItem(cartOrder.getCartItemId(),
principal.getName())){
                return new ResponseEntity<String>("주문 권한이 없습니다.",
HttpStatus.FORBIDDEN);
            }
        }

        Long orderId = cartService.orderCartItem(cartOrderDtoList,
principal.getName()); ······❸
        return new ResponseEntity<Long>(orderId, HttpStatus.OK); ······❹
    }

}
```

8.3 장바구니 상품 주문

resources/templates/cart/cartList.html

```
<script th:inline="javascript">

    function orders(){
        var token = $("meta[name='_csrf']").attr("content");
        var header = $("meta[name='_csrf_header']").attr("content");

        var url = "/cart/orders";

        var dataList = new Array();
        var paramData = new Object();

        $("input[name=cartChkBox]:checked").each(function() { ------------------ ❶
            var cartItemId = $(this).val();
            var data = new Object();
            data["cartItemId"] = cartItemId;
            dataList.push(data);
        });

```

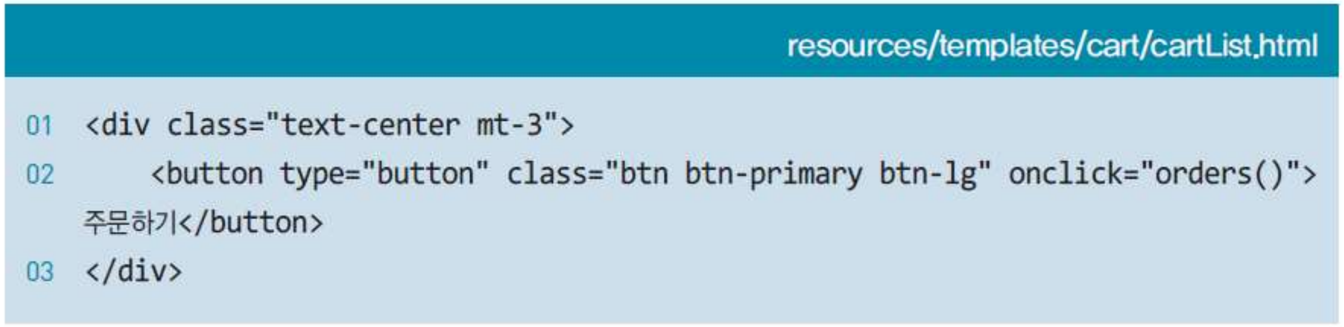
8.3 장바구니 상품 주문



```
        paramData['cartOrderDtoList'] = dataList; ❷

        var param = JSON.stringify(paramData);

        $.ajax({
            url      : url,
            type     : "POST",
            contentType : "application/json",
            data     : param,
            beforeSend : function(xhr){
                /* 데이터를 전송하기 전에 헤더에 csrf 값 설정 */
                xhr.setRequestHeader(header, token);
            },
            dataType : "json",
            cache   : false,
            success  : function(result, status){
                alert("주문이 완료 되었습니다.");
                location.href='/orders'; ❸
            },
            error : function(jqXHR, status, error){

                if(jqXHR.status == '401'){
                    alert('로그인 후 이용해주세요');
                    location.href='/members/login';
                } else{
                    alert(jqXHR.responseJSON.message);
                }

            }
        });
    }

</script>
```

8.3 장바구니 상품 주문

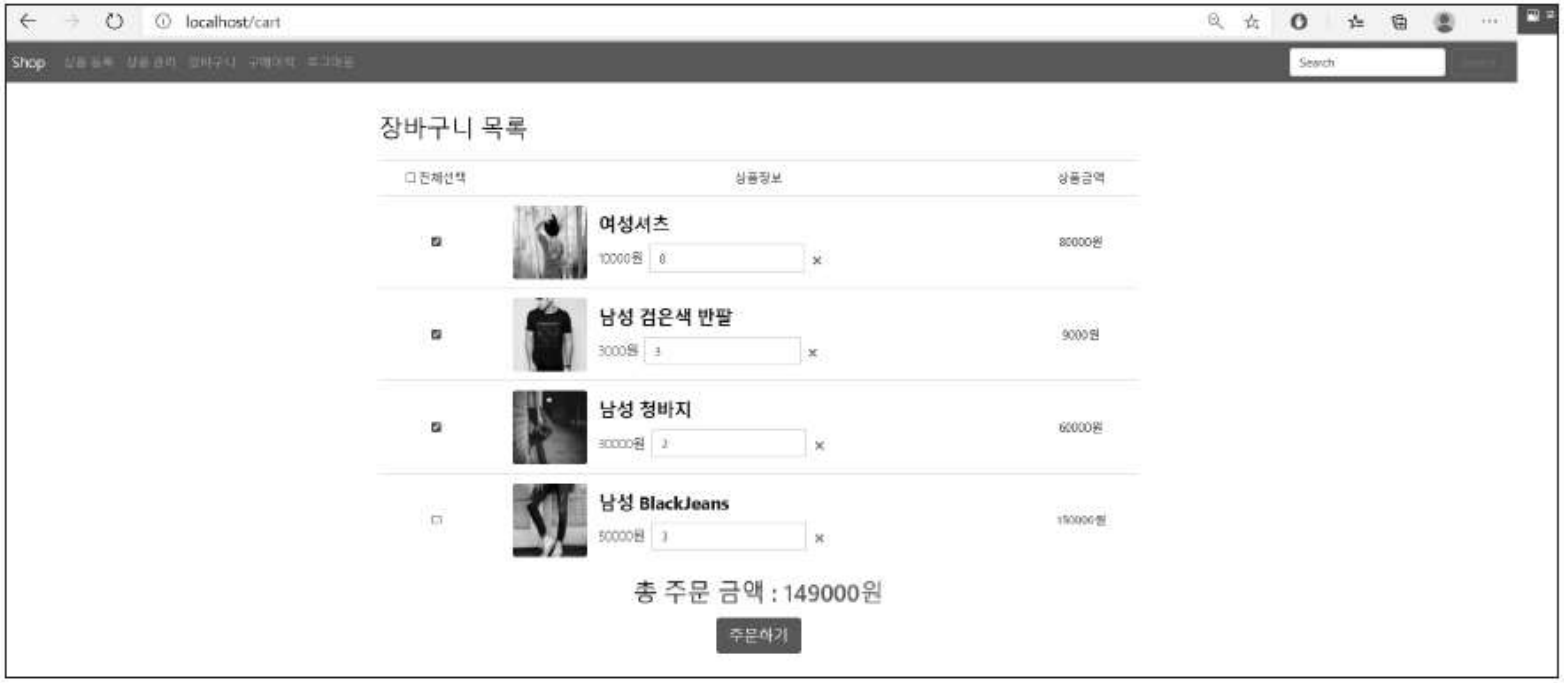
- <주문하기> 버튼을 클릭하면 구현한 orders() 함수가 실행될 수 있도록 주문하기 버튼의 onclick 속성에 추가



resources/templates/cart/cartList.html

```
<div class="text-center mt-3">
    <button type="button" class="btn btn-primary btn-lg" onclick="orders()">주문하기</button>
</div>
```

8.3 장바구니 상품 주문



장바구니 목록

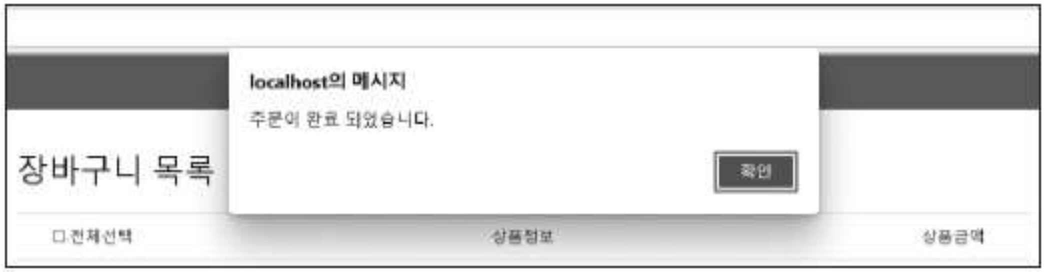
☐ 전체선택	상품정보	상품금액
☑	여성셔츠 10000원 8 ×	80000원
☑	남성 검은색 반팔 3000원 3 ×	9000원
☑	남성 청바지 30000원 2 ×	60000원
☐	남성 BlackJeans 50000원 3 ×	150000원

총 주문 금액 : 149000원

주문하기

[그림 8-6] 장바구니 상품 선택

8.3 장바구니 상품 주문

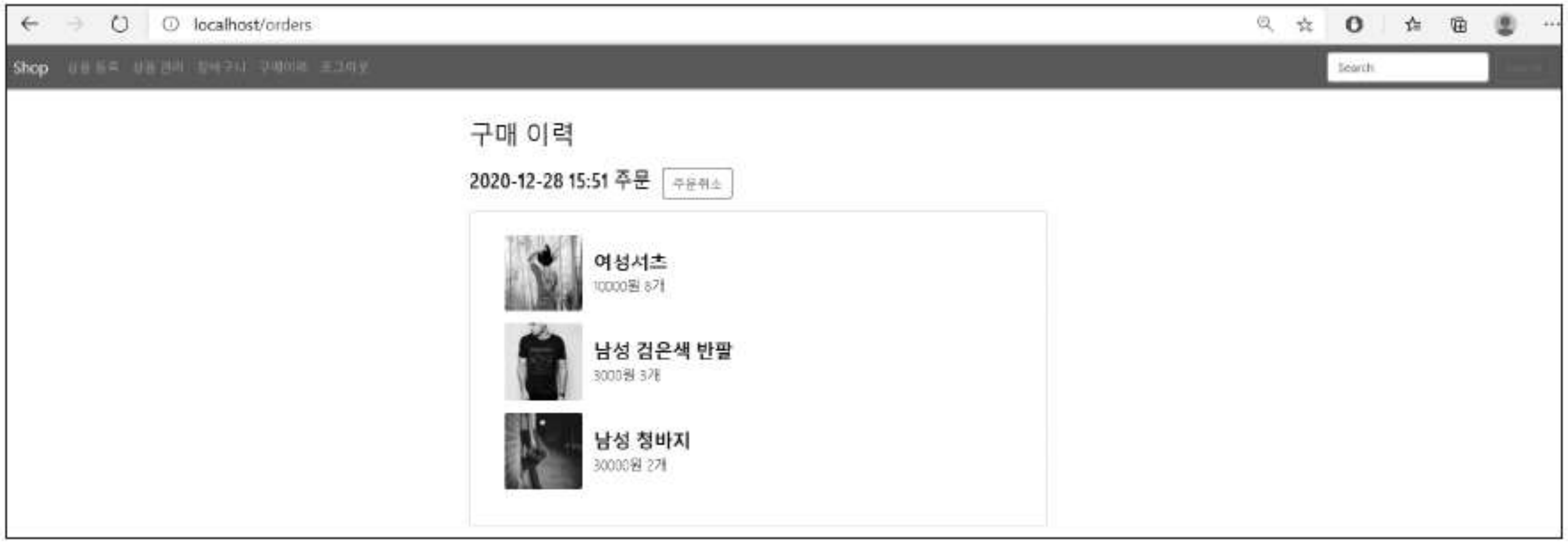


localhost의 메시지

주문이 완료 되었습니다.

[그림 8-7] 장바구니 상품 주문 성공

8.3 장바구니 상품 주문



구매 이력

2020-12-28 15:51 주문

여성셔츠
10000원 8개

남성 검은색 반팔
3000원 3개

남성 청바지
30000원 2개

[그림 8-8] 구매 이력 확인

Thank you for your attention