

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

```
01 package com.shop.dto;
02
03 import lombok.Getter;
04 import lombok.Setter;
05
06 import javax.validation.constraints.Min;
07 import javax.validation.constraints.NotNull;
08
09 @Getter @Setter
10 public class CartItemDto {
11
12     @NotNull(message = "상품 아이디는 필수 입력 값 입니다.")
13     private Long itemId;
14
15     @Min(value = 1, message = "최소 1개 이상 담아주세요")
16     private int count;
17
18 }
```

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

```
01 package com.shop.entity;
02
03 .....기존 임포트 생략.....
04
05 @Entity
06 @Table(name = "cart")
07 @Getter @Setter
08 @ToString
09 public class Cart extends BaseEntity{
10
11     .....코드 생략.....
12
13     public static Cart createCart(Member member){
14         Cart cart = new Cart();
15         cart.setMember(member);
16         return cart;
17     }
18
19 }
```

8.1 장바구니 담기

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

com.shop.entity.CartItem.java

```
01 package com.shop.entity;
02
03 .....기존 임포트 생략.....
04
05 @Entity
06 @Getter @Setter
07 @Table(name="cart_item")
08 public class CartItem extends BaseEntity{
09
10     .....코드 생략.....
11
12     public static CartItem createCartItem(Cart cart, Item item, int count){
13         CartItem cartItem = new CartItem();
14         cartItem.setCart(cart);
15         cartItem.setItem(item);
16         cartItem.setCount(count);
17         return cartItem;
18     }
19
20     public void addCount(int count){ ..... ❶
21         this.count += count;
22     }
23 }
```

8.1 장바구니 담기

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

com.shop.repository.CartRepository.java

```
01 package com.shop.repository;
02
03 import com.shop.entity.Cart;
04 import org.springframework.data.jpa.repository.JpaRepository;
05
06 public interface CartRepository extends JpaRepository<Cart, Long> {
07
08     Cart findByMemberId(Long memberId);
09
10 }
```

8.1 장바구니 담기

- 장바구니에 들어갈 상품을 저장하거나 조회하기 위해서 CartItemRepository 인터페이스를 생성

com.shop.repository.CartItemRepository.java

```
01 package com.shop.repository;
02
03 import com.shop.entity.CartItem;
04 import org.springframework.data.jpa.repository.JpaRepository;
05
06 public interface CartItemRepository extends JpaRepository<CartItem, Long> {
07
08     CartItem findByCartIdAndItemId(Long cartId, Long itemId); ①
09
10 }
```

8.1 장바구니 담기

com.shop.service.CartService.java

```
01 package com.shop.service;
02
03 import com.shop.dto.CartItemDto;
04 import com.shop.entity.Cart;
05 import com.shop.entity.CartItem;
06 import com.shop.entity.Item;
07 import com.shop.entity.Member;
08 import com.shop.repository.CartItemRepository;
09 import com.shop.repository.CartRepository;
10 import com.shop.repository.ItemRepository;
11 import com.shop.repository.MemberRepository;
12 import lombok.RequiredArgsConstructor;
13 import org.springframework.stereotype.Service;
14 import org.springframework.transaction.annotation.Transactional;
15
16 import javax.persistence.EntityNotFoundException;
17
18 @Service
19 @RequiredArgsConstructor
20 @Transactional
21 public class CartService {
22
23     private final ItemRepository itemRepository;
24     private final MemberRepository memberRepository;
25     private final CartRepository cartRepository;
26     private final CartItemRepository cartItemRepository;
27
28     public Long addCart(CartItemDto cartItemDto, String email){
29
```

8.1 장바구니 담기

```
30     Item item = itemRepository.findById(cartItemDto.getItemId()) .....❶
31         .orElseThrow(EntityNotFoundException::new);
32     Member member = memberRepository.findByEmail(email); .....❷
33
34     Cart cart = cartRepository.findById(member.getId()); .....❸
35     if(cart == null){ .....❹
36         cart = Cart.createCart(member);
37         cartRepository.save(cart);
38     }
39
40     CartItem savedCartItem =
41     cartItemRepository.findByIdAndItemId(cart.getId(), item.getId()); .....❺
42
43     if(savedCartItem != null){
44         savedCartItem.addCount(cartItemDto.getCount()); .....❻
45         return savedCartItem.getId();
46     } else {
47         CartItem cartItem =
48         CartItem.createCartItem(cart, item, cartItemDto.getCount()); .....❼
49         cartItemRepository.save(cartItem); .....❽
50         return cartItem.getId();
51     }
52
53 }
```


8.1 장바구니 담기

com.shop.controller.CartController.java

```
01 package com.shop.controller;
02
03 import com.shop.dto.CartItemDto;
04 import com.shop.service.CartService;
05 import lombok.RequiredArgsConstructor;
06 import org.springframework.http.HttpStatus;
07 import org.springframework.http.ResponseEntity;
08 import org.springframework.stereotype.Controller;
09 import org.springframework.validation.BindingResult;
10 import org.springframework.validation.FieldError;
11 import org.springframework.web.bind.annotation.PostMapping;
12 import org.springframework.web.bind.annotation.RequestBody;
13 import org.springframework.web.bind.annotation.ResponseBody;
14
15 import javax.validation.Valid;
16 import java.security.Principal;
17 import java.util.List;
18
19 @Controller
20 @RequiredArgsConstructor
21 public class CartController {
22
23     private final CartService cartService;
24
25     @PostMapping(value = "/cart")
26     public @ResponseBody
27     ResponseEntity order(@RequestBody @Valid CartItemDto cartItemDto,
28                          BindingResult bindingResult, Principal principal){
29         if(bindingResult.hasErrors()){ ..... ❶
30             StringBuilder sb = new StringBuilder();
31             List<FieldError> fieldErrors = bindingResult.getFieldErrors();
32             for (FieldError fieldError : fieldErrors) {
33                 sb.append(fieldError.getDefaultMessage());
34             }
35             return new ResponseEntity<String>
36             (sb.toString(), HttpStatus.BAD_REQUEST);
37         }
```

8.1 장바구니 담기

```
38     String email = principal.getName(); .....❷
39     Long cartItemId;
40
41     try {
42         cartItemId = cartService.addCart(cartItemDto, email); .....❸
43     } catch (Exception e) {
44         return new ResponseEntity<String>(e.getMessage(),
45         HttpStatus.BAD_REQUEST);
46     }
47     return new ResponseEntity<Long>(cartItemId, HttpStatus.OK); .....❹
48 }
49
50 }
```

8.1 장바구니 담기



[함께 해봐요 8-2] 장바구니 담기 테스트하기

com.shop.service.CartServiceTest.java

```
01 package com.shop.service;
02
03 import com.shop.constant.ItemSellStatus;
04 import com.shop.dto.CartItemDto;
05 import com.shop.entity.CartItem;
06 import com.shop.entity.Item;
07 import com.shop.entity.Member;
08 import com.shop.repository.CartItemRepository;
09 import com.shop.repository.ItemRepository;
10 import com.shop.repository.MemberRepository;
11 import org.junit.jupiter.api.DisplayName;
12 import org.junit.jupiter.api.Test;
13 import org.springframework.beans.factory.annotation.Autowired;
14 import org.springframework.boot.test.context.SpringBootTest;
15 import org.springframework.test.context.TestPropertySource;
16 import org.springframework.transaction.annotation.Transactional;
17
18 import javax.persistence.EntityNotFoundException;
19
20 import static org.junit.jupiter.api.Assertions.assertEquals;
21
22 @SpringBootTest
23 @Transactional
24 @TestPropertySource(locations="classpath:application-test.properties")
```

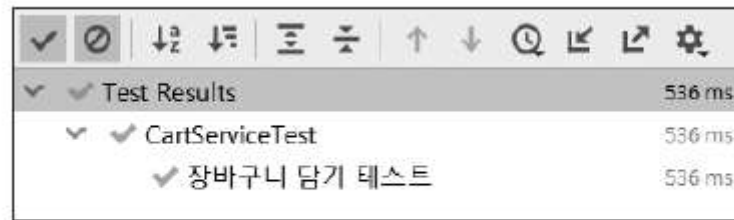
8.1 장바구니 담기

```
25 class CartServiceTest {
26
27     @Autowired
28     ItemRepository itemRepository;
29
30     @Autowired
31     MemberRepository memberRepository;
32
33     @Autowired
34     CartService cartService;
35
36     @Autowired
37     CartItemRepository cartItemRepository;
38
39     public Item saveItem(){ ..... ❶
40         Item item = new Item();
41         item.setItemNm("테스트 상품");
42         item.setPrice(10000);
43         item.setItemDetail("테스트 상품 상세 설명");
44         item.setItemSellStatus(ItemSellStatus.SELL);
45         item.setStockNumber(100);
46         return itemRepository.save(item);
47     }
48
49     public Member saveMember(){ ..... ❷
50         Member member = new Member();
51         member.setEmail("test@test.com");
52         return memberRepository.save(member);
53     }
54
55     @Test
56     @DisplayName("장바구니 담기 테스트")
57     public void addCart(){
58         Item item = saveItem();
```

8.1 장바구니 담기

```
59     Member member = saveMember();
60
61     CartItemDto cartItemDto = new CartItemDto();
62     cartItemDto.setCount(5); ..... ❸
63     cartItemDto.setItemId(item.getId()); ..... ❹
64
65     Long cartItemId = cartService.addCart(cartItemDto, member.getEmail()); ❺
66
67     CartItem cartItem = cartItemRepository.findById(cartItemId) ..... ❻
68         .orElseThrow(EntityNotFoundException::new);
69
70     assertEquals(item.getId(), cartItem.getItem().getId()); ..... ❼
71     assertEquals(cartItemDto.getCount(), cartItem.getCount()); ..... ❽
72 }
73
74 }
```

8.1 장바구니 담기



The screenshot shows a 'Test Results' window from an IDE. It contains a tree view with the following structure:

- Test Results (536 ms)
 - CartServiceTest (536 ms)
 - 장바구니 담기 테스트 (536 ms)

All tests are marked with a green checkmark, indicating they passed successfully.

Test Name	Duration
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/itemDt1.html

```
01 <script th:inline="javascript">
02
03     .....코드 생략.....
04
05     function addCart(){
06         var token = $("meta[name='_csrf']").attr("content");
07         var header = $("meta[name='_csrf_header']").attr("content");
08
09         var url = "/cart";
10         var paramData = {
11             itemId : $("#itemId").val(),
12             count : $("#count").val()
13         };
14
15         var param = JSON.stringify(paramData);
16
17         $.ajax({
18             url      : url,
19             type     : "POST",
20             contentType : "application/json",
21             data     : param,
22             beforeSend : function(xhr){
23                 /* 데이터를 전송하기 전에 헤더에 csrf 값을 설정 */
24                 xhr.setRequestHeader(header, token);
25             },
26             dataType : "json",
27             cache    : false,
28             success  : function(result, status){
29                 alert("상품을 장바구니에 담았습니다.");
30                 location.href='/';
31             },
```

8.1 장바구니 담기

```
32     error : function(jqXHR, status, error){
33
34         if(jqXHR.status == '401'){
35             alert('로그인 후 이용해주세요');
36             location.href='/members/login';
37         } else{
38             alert(jqXHR.responseText);
39         }
40
41     }
42 });
43 }
44
45 </script>
```

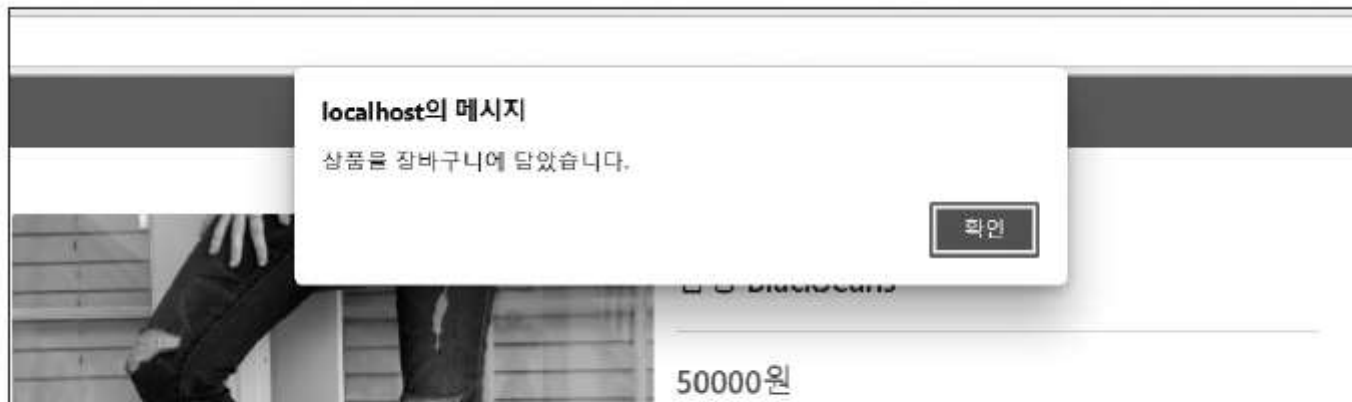

8.1 장바구니 담기

- 장바구니 담기 버튼 클릭시 addCart() 함수 호출 추가

resources/templates/item/itemDtl.html

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

8.1 장바구니 담기



[그림 8-2] 상품 장바구니 담기 성공

8.2 장바구니 조회



[함께 해봐요 8-4] 장바구니 조회하기

com.shop.dto.CartDetailDto.java

```
01 package com.shop.dto;
02
03 import lombok.Getter;
04 import lombok.Setter;
05
06 @Getter @Setter
07 public class CartDetailDto {
08
09     private Long cartItemId; //장바구니 상품 아이디
10
11     private String itemNm; //상품명
12
13     private int price; //상품 금액
14
15     private int count; //수량
16
17     private String imgUrl; //상품 이미지 경로
18
19     public CartDetailDto(Long cartItemId, String itemNm, int price,
20                          int count, String imgUrl){
21         this.cartItemId = cartItemId;
22         this.itemNm = itemNm;
23         this.price = price;
24         this.count = count;
25         this.imgUrl = imgUrl;
26     }
27 }
```

8.2 장바구니 조회

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

```
com.shop.repository.CartItemRepository.java

01 package com.shop.repository;
02
03 .....기존 임포트 생략.....
04
05 import com.shop.dto.CartDetailDto;
06 import org.springframework.data.jpa.repository.Query;
07 import java.util.List;
08
09 public interface CartItemRepository extends JpaRepository<CartItem, Long> {
10
11     .....코드 생략.....
12
13     @Query("select new com.shop.dto.CartDetailDto(ci.id, i.itemNm, i.price,
14 ci.count, im.imgUrl) " + ..... ❶
15         "from CartItem ci, ItemImg im " +
16         "join ci.item i " +
17         "where ci.cart.id = :cartId " +
18         "and im.item.id = ci.item.id " + ..... ❷
19         "and im.repimgYn = 'Y' " + ..... ❸
20         "order by ci.regTime desc"
21     )
22     List<CartDetailDto> findCartDetailDtoList(Long cartId);
23 }
```

8.2 장바구니 조회

com.shop.service.CartService.java

```
01 package com.shop.service;
02
03 .....기존 импорт 생략.....
04
05 import com.shop.dto.CartDetailDto;
06 import java.util.ArrayList;
07 import java.util.List;
08
09 @Service
10 @RequiredArgsConstructor
11 @Transactional
12 public class CartService {
13
14     .....코드 생략.....
15
16     @Transactional(readOnly = true)
17     public List<CartDetailDto> getCartList(String email){
18
19         List<CartDetailDto> cartDetailDtoList = new ArrayList<>();
20
21         Member member = memberRepository.findById(email);
22         Cart cart = cartRepository.findById(member.getId()); .....❶
23         if(cart == null){ .....❷
24             return cartDetailDtoList;
25         }
26
27         cartDetailDtoList =
28         cartItemRepository.findCartDetailDtoList(cart.getId()); .....❸
29
30         return cartDetailDtoList;
31     }
32 }
```

8.2 장바구니 조회

com.shop.controller.CartController.java

```
01 package com.shop.controller;
02
03 .....기존 임포트 생략.....
04
05 import com.shop.dto.CartDetailDto;
06 import org.springframework.ui.Model;
07 import org.springframework.web.bind.annotation.GetMapping;
08
09 @Controller
10 @RequiredArgsConstructor
11 public class CartController {
12
13     .....코드 생략.....
14
15     @GetMapping(value = "/cart")
16     public String orderHist(Principal principal, Model model){
17         List<CartDetailDto> cartDetailList =
18         cartService.getCartList(principal.getName()); .....❶
19         model.addAttribute("cartItems", cartDetailList); .....❷
20         return "cart/cartList";
21     }
22 }
```

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



[그림 8-3] 장바구니 목록

8.2 장바구니 조회

resources/templates/cart/cartList.html

```
01 <!DOCTYPE html>
02 <html xmlns:th="http://www.thymeleaf.org"
03     xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"
04     layout:decorate="~{layouts/layout1}">
05
06 <head>
07     <meta name="_csrf" th:content="${_csrf.token}"/>
08     <meta name="_csrf_header" th:content="${_csrf.headerName}"/>
09 </head>
10
11 <!-- 사용자 스크립트 추가 -->
12 <th:block layout:fragment="script">
13
14     <script th:inline="javascript">
15
16         $(document).ready(function(){
17             $("input[name=cartChkBox]").change( function(){
18                 getOrderTotalPrice();
19             });
20         });
```


8.2 장바구니 조회

```
22     function getOrderTotalPrice(){
23         var orderTotalPrice = 0;
24         $("input[name=cartChkBox]:checked").each(function() {
25             var cartItemId = $(this).val();
26             var price = $("#price_" + cartItemId).attr("data-price");
27             var count = $("#count_" + cartItemId).val();
28             orderTotalPrice += price*count;
29         });
30
31         $("#orderTotalPrice").html(orderTotalPrice+'원');
32     }
33
34     function changeCount(obj){
35         var count = obj.value;
36         var cartItemId = obj.id.split('_')[1];
37         var price = $("#price_" + cartItemId).data("price");
38         var totalPrice = count*price;
39         $("#totalPrice_" + cartItemId).html(totalPrice+"원");
40         getOrderTotalPrice();
41     }
```

8.2 장바구니 조회

```
43     function checkAll(){
44         if($("#checkall").prop("checked")){
45             $("input[name=cartChkBox]").prop("checked",true);
46         }else{
47             $("input[name=cartChkBox]").prop("checked",false);
48         }
49         getOrderTotalPrice();
50     }
51
52     </script>
53
54 </th:block>
55
56 <!-- 사용자 CSS 추가 -->
57 <th:block layout:fragment="css">
58     <style>
59         .content-mg{
60             margin-left:25%;
61             margin-right:25%;
62             margin-top:2%;
63             margin-bottom:100px;
64         }
```

8.2 장바구니 조회

```
65     .repImgDiv{
66         margin-right:15px;
67         margin-left:15px;
68         height:auto;
69     }
70     .repImg{
71         height:100px;
72         width:100px;
73     }
74     .fs18{
75         font-size:18px
76     }
77     .fs24{
78         font-size:24px
79     }
80 </style>
81 </th:block>
82 <div layout:fragment="content" class="content-mg">
83
84
85     <h2 class="mb-4">
86         장바구니 목록
87     </h2>
88
89     <div>
```

8.2 장바구니 조회

```
91     <table class="table">
92         <colgroup>
93             <col width="15%" />
94             <col width="70%" />
95             <col width="15%" />
96         </colgroup>
97         <thead>
98             <tr class="text-center">
99                 <td>
100                     <input type="checkbox" id="checkall"
101                         onclick="checkAll()"> 전체선택
102                 </td>
103                 <td>상품정보</td>
104                 <td>상품금액</td>
105             </tr>
106         </thead>
107         <tbody>
108             <tr th:each="cartItem : ${cartItems}">
109                 <td class="text-center align-middle">
110                     <input type="checkbox" name="cartChkBox"
111                         th:value="${cartItem.cartItemId}">
112                 </td>
113                 <td class="d-flex">
114                     <div class="repImgDiv align-self-center">
115                         
117                     </div>
```

8.2 장바구니 조회

```
115         <div class="align-self-center">
116             <span th:text="${cartItem.itemNm}"
117             class="fs24 font-weight-bold"></span>
118             <div class="fs18 font-weight-light">
119                 <span class="input-group mt-2">
120                     <span th:id="'price_' + ${cartItem.cartItemId}"
121                     th:data-price="${cartItem.price}"
122                     th:text="${cartItem.price} + '원'"
123                     class="align-self-center mr-2">
124                         </span>
125                         <input type="number" name="count"
126                         th:id="'count_' + ${cartItem.cartItemId}"
127                         th:value="${cartItem.count}" min="1"
128                         onchange="changeCount(this)"
129                         class="form-control mr-2" >
130                             <button type="button" class="close"
131                             aria-label="Close">
132                                 <span aria-hidden="true"
133                                 th:data-id="${cartItem.cartItemId}">&times;</span>
134                                 </button>
135                             </span>
136                         </div>
137                     </div>
138                 </td>
139                 <td class="text-center align-middle">
140                     <span th:id="'totalPrice_' + ${cartItem.cartItemId}"
141                     name="totalPrice"
142                     th:text="${cartItem.price * cartItem.count} + '원'">
143                         </span>
144                 </td>
145             </tr>
146         </tbody>
147     </table>
```

8.2 장바구니 조회

```
142     <h2 class="text-center">
143         총 주문 금액 : <span id="orderTotalPrice" class="text-danger">0원
144     </h2>
145
146     <div class="text-center mt-3">
147         <button type="button" class="btn btn-primary btn-lg">주문하기
148     </div>
149
150 </div>
151
152 </div>
153
154 </html>
```

8.2 장바구니 조회



[함께 해봐요 8-5] 장바구니 상품 수량 변경하기

com.shop.entity.CartItem.java

```
01 package com.shop.entity;
02
03 .....기존 импорт 생략.....
04
05 @Entity
06 @Getter @Setter
07 @Table(name="cart_item")
08 public class CartItem extends BaseEntity{
09
10     .....코드 생략.....
11
12     public void updateCount(int count){
13         this.count = count;
14     }
15
16 }
```

8.2 장바구니 조회

```
com.shop.service.CartService.java

01 package com.shop.service;
02
03 .....기존 임포트 생략.....
04
05 import org.thymeleaf.util.StringUtils;
06
07 @Service
08 @RequiredArgsConstructor
09 @Transactional
10 public class CartService {
11
12     .....코드 생략.....
13
14     @Transactional(readOnly = true)
15     public boolean validateCartItem(Long cartItemId, String email){
16         Member curMember = memberRepository.findByEmail(email); ❶
17         CartItem cartItem = cartItemRepository.findById(cartItemId)
18             .orElseThrow(EntityNotFoundException::new);
19         Member savedMember = cartItem.getCart().getMember(); ❷
20
21         if(!StringUtils.equals(curMember.getEmail(),
22             savedMember.getEmail())){ ❸
23             return false;
24         }
25         return true; ❹
26     }
27
28     public void updateCartItemCount(Long cartItemId, int count){ ❺
29         CartItem cartItem = cartItemRepository.findById(cartItemId)
30             .orElseThrow(EntityNotFoundException::new);
31
32         cartItem.updateCount(count);
33     }
34
35 }
```


8.2 장바구니 조회

com.shop.controller.CartController.java

```
01 package com.shop.controller;
02
03 .....기존 임포트 생략.....
04
05 import org.springframework.web.bind.annotation.PatchMapping;
06 import org.springframework.web.bind.annotation.PathVariable;
07
08 @Controller
09 @RequiredArgsConstructor
10 public class CartController {
11
12     .....코드 생략.....
13
14     @PatchMapping(value = "/cartItem/{cartItemId}") .....❶
15     public @ResponseBody ResponseEntity updateCartItem
16     (@PathVariable("cartItemId") Long cartItemId, int count, Principal principal){
17
18         if(count <= 0){ .....❷
19             return new ResponseEntity<String>
20             ("최소 1개 이상 담아주세요", HttpStatus.BAD_REQUEST);
21         } else if(!cartService.validateCartItem
22             (cartItemId, principal.getName())){ .....❸
23             return new ResponseEntity<String>
24             ("수정 권한이 없습니다.", HttpStatus.FORBIDDEN);
25         }
26
27         cartService.updateCartItemCount(cartItemId, count); .....❹
28         return new ResponseEntity<Long>(cartItemId, HttpStatus.OK);
29     }
30 }
```

8.2 장바구니 조회

resources/templates/cart/cartList.html

```
01 <script th:inline="javascript">
02
03     ....코드 생략...|.
04
05     function updateCartItemCnt(cartItemId, count){
06         var token = $("meta[name='_csrf']").attr("content");
07         var header = $("meta[name='_csrf_header']").attr("content");
08
09         var url = "/cartItem/" + cartItemId + "?count=" + count;
10
11         $.ajax({
12             url      : url,
13             type      : "PATCH",
14             beforeSend : function(xhr){
15                 /* 데이터를 전송하기 전에 헤더에 csrf 값을 설정 */
16                 xhr.setRequestHeader(header, token);
17             },
18             dataType  : "json",
19             cache      : false,
20             success    : function(result, status){
21                 console.log("cartItem count update success");
22             },
23             error      : function(jqXHR, status, error){
24
25                 if(jqXHR.status == '401'){
26                     alert('로그인 후 이용해주세요');
27                     location.href='/members/login';
28                 } else{
29                     alert(jqXHR.responseJSON.message);
30                 }
31             }
32         });
33     }
34 }
35
36 </script>
```

8.2 장바구니 조회

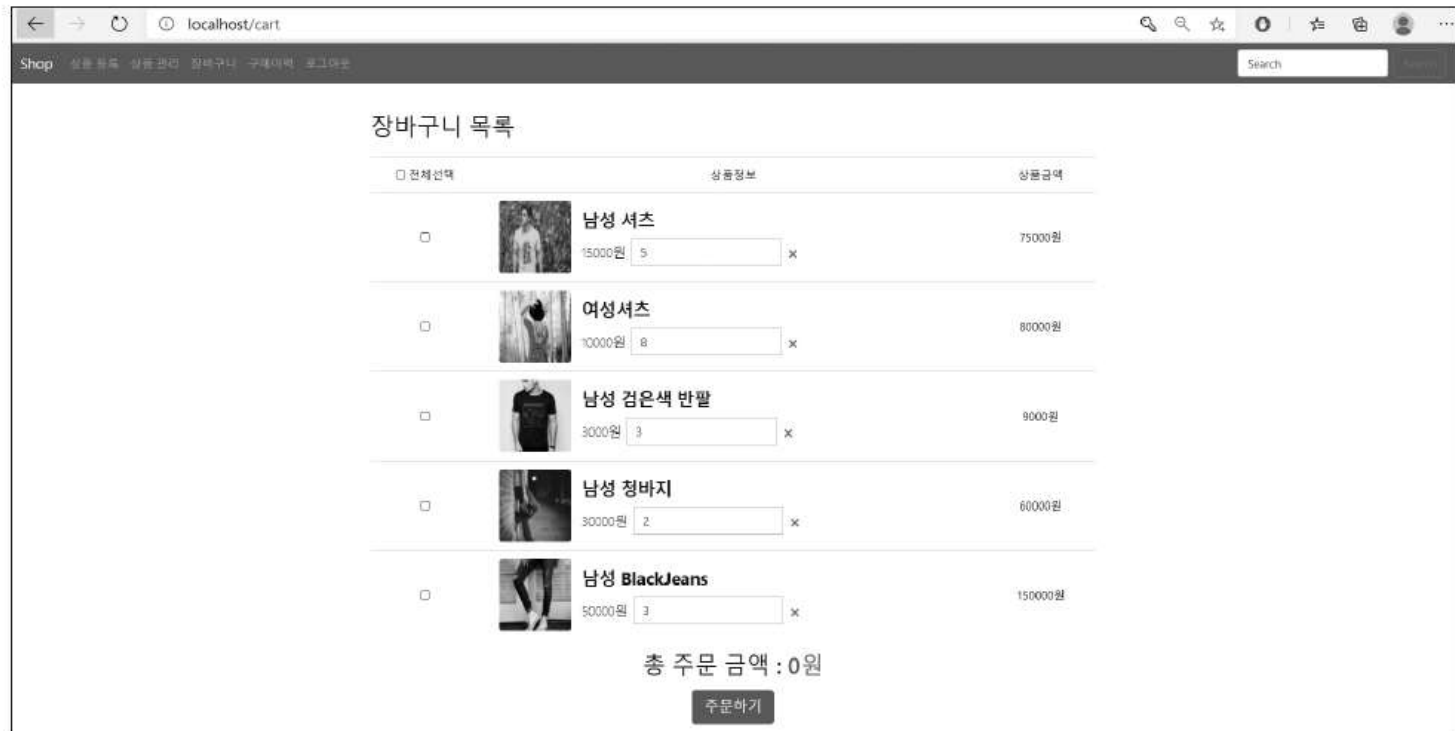
- 작성한 updateCartItemCount() 함수를 기존에 작성한 changeCount() 함수 마지막에 추가

resources/templates/cart/cartList.html

```
01 <script th:inline="javascript">
02
03     .....코드 생략.....
04
05     function changeCount(obj){
06         var count = obj.value;
07         var cartItemId = obj.id.split('_')[1];
08         var price = $("#price_" + cartItemId).attr("data-price");
09         var totalPrice = count*price;
10         $("#totalPrice_" + cartItemId).html(totalPrice+"원");
11         getOrderTotalPrice();
12         updateCartItemCount(cartItemId, count);
13     }
14
15 </script>
```

8.2 장바구니 조회

- 장바구니 상품의 수량을 변경 후 페이지 새로고침시 변경된 수량 유지



[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

```
01 package com.shop.service;
02
03 .....기존 임포트 생략.....
04
05 @Service
06 @RequiredArgsConstructor
07 @Transactional
08 public class CartService {
09
10     .....코드 생략.....
11
12     public void deleteCartItem(Long cartItemId) {
13         CartItem cartItem = cartItemRepository.findById(cartItemId)
14             .orElseThrow(EntityNotFoundException::new);
15         cartItemRepository.delete(cartItem);
16     }
17 }
```

8.2 장바구니 조회

com.shop.controller.CartController.java

```
01 package com.shop.controller;
02
03 .....기존 임포트 생략.....
04
05 import org.springframework.web.bind.annotation.DeleteMapping;
06
07 @Controller
08 @RequiredArgsConstructor
09 public class CartController {
10
11     .....코드 생략.....
12
13     @DeleteMapping(value = "/cartItem/{cartItemId}") ❶
14     public @ResponseBody ResponseEntity deleteCartItem
15     (@PathVariable("cartItemId") Long cartItemId, Principal principal){
16
17         if(!cartService.validateCartItem(cartItemId, principal.getName())){ ❷
18             return new ResponseEntity<String>("수정 권한이 없습니다.",
19                 HttpStatus.FORBIDDEN);
20         }
21
22         cartService.deleteCartItem(cartItemId); ❸
23         return new ResponseEntity<Long>(cartItemId, HttpStatus.OK);
24     }
25 }
```

8.2 장바구니 조회

(resources/templates/cart/cartList.html)

```
01 <script th:inline="javascript">
02
03     .....코드 생략.....
04
05     function deleteCartItem(obj){
06         var cartItemId = obj.dataset.id;
07         var token = $("meta[name='_csrf']").attr("content");
08         var header = $("meta[name='_csrf_header']").attr("content");
09
10         var url = "/cartItem/" + cartItemId;
11
12         $.ajax({
13             url      : url,
14             type     : "DELETE", ..... ❶
15             beforeSend : function(xhr){
```

8.2 장바구니 조회

```
16      /* 데이터를 전송하기 전에 헤더에 csrf 값을 설정 */
17      xhr.setRequestHeader(header, token);
18  },
19  dataType : "json",
20  cache   : false,
21  success : function(result, status){
22      location.href='/cart';
23  },
24  error : function(jqXHR, status, error){
25
26      if(jqXHR.status == '401'){
27          alert('로그인 후 이용해주세요');
28          location.href='/members/login';
29      } else{
30          alert(jqXHR.responseJSON.message);
31      }
32
33  }
34  });
35  }
36  </script>
```

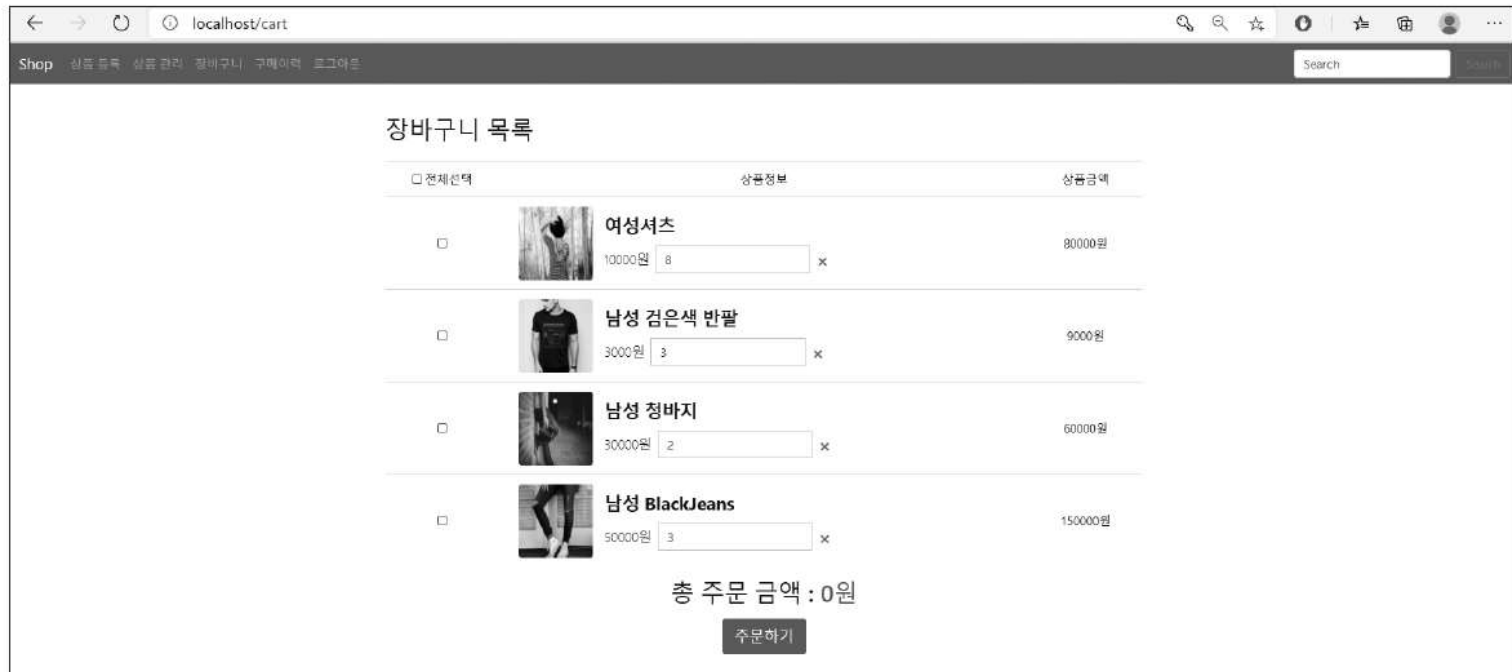

8.2 장바구니 조회

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

resources/templates/cart/cartList.html

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

8.2 장바구니 조회



[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

```
01 package com.shop.dto;
02
03 import lombok.Getter;
04 import lombok.Setter;
05
06 import java.util.List;
07
08 @Getter
09 @Setter
10 public class CartOrderDto {
11
12     private Long cartItemId;
13
14     private List<CartOrderDto> cartOrderDtoList; ①
15
16 }
```

8.3 장바구니 상품 주문

com.shop.service.OrderService.java

```
01 package com.shop.service;
02
03 .....기존 임포트 생략.....
04
05 @Service
06 @Transactional
07 @RequiredArgsConstructor
08 public class OrderService {
09
10     .....코드 생략.....
11
12     public Long orders(List<OrderDto> orderDtoList, String email){
13
14         Member member = memberRepository.findByEmail(email);
15         List<OrderItem> orderItemList = new ArrayList<>();
16
17         for (OrderDto orderDto : orderDtoList) { .....❶
18             Item item = itemRepository.findById(orderDto.getItemId())
19                 .orElseThrow(EntityNotFoundException::new);
20
21             OrderItem orderItem =
22                 OrderItem.createOrderItem(item, orderDto.getCount());
23             orderItemList.add(orderItem);
24         }
25
26         Order order = Order.createOrder(member, orderItemList); .....❷
27         orderRepository.save(order); .....❸
28
29         return order.getId();
30     }
31 }
```

8.3 장바구니 상품 주문

com.shop.service.CartService.java

```
01 package com.shop.service;
02
03 .....기존 임포트 생략.....
04
05 import com.shop.dto.CartOrderDto;
06 import com.shop.dto.OrderDto;
07
08 @Service
09 @RequiredArgsConstructor
10 @Transactional
11 public class CartService {
12
13     private final ItemRepository itemRepository;
14     private final MemberRepository memberRepository;
15     private final CartRepository cartRepository;
16     private final CartItemRepository cartItemRepository;
17     private final OrderService orderService;
18
19     .....코드 생략.....
20
21     public Long orderCartItem(List<CartOrderDto> cartOrderDtoList, String email){
22         List<OrderDto> orderDtoList = new ArrayList<>();
23         for (CartOrderDto cartOrderDto : cartOrderDtoList) { .....❶
24             CartItem cartItem = cartItemRepository
25                 .findById(cartOrderDto.getCartItemId())
26                 .orElseThrow(EntityNotFoundException::new);
27
28             OrderDto orderDto = new OrderDto();
29             orderDto.setItemId(cartItem.getItemId());
30             orderDto.setCount(cartItem.getCount());
31             orderDtoList.add(orderDto);
32         }
33
34         Long orderId = orderService.orders(orderDtoList, email); .....❷
35
36         for (CartOrderDto cartOrderDto : cartOrderDtoList) { .....❸
37             CartItem cartItem = cartItemRepository
38                 .findById(cartOrderDto.getCartItemId())
39                 .orElseThrow(EntityNotFoundException::new);
40             cartItemRepository.delete(cartItem);
41         }
42
43         return orderId;
44     }
45 }
```

8.3 장바구니 상품 주문

com.shop.controller.CartController.java

```
01 package com.shop.controller;
02
03 .....기존 임포트 생략.....
04
05 import com.shop.dto.CartOrderDto;
06
07 @Controller
08 @RequiredArgsConstructor
09 public class CartController {
10
11     .....코드 생략.....
12
13     @PostMapping(value = "/cart/orders")
14     public @ResponseBody ResponseEntity orderCartItem
15     (@RequestBody CartOrderDto cartOrderDto, Principal principal){
16
17         List<CartOrderDto> cartOrderDtoList =
18         cartOrderDto.getCartOrderDtoList();
19
20         if(cartOrderDtoList == null || cartOrderDtoList.size() == 0){ .....❶
21             return new ResponseEntity<String>("주문할 상품을 선택해주세요",
22             HttpStatus.FORBIDDEN);
23         }
24
25         for (CartOrderDto cartOrder : cartOrderDtoList) { .....❷
26             if(!cartService.validateCartItem(cartOrder.getCartItem(),
27             principal.getName())){
28
29                 return new ResponseEntity<String>("주문 권한이 없습니다.",
30                 HttpStatus.FORBIDDEN);
31             }
32
33             Long orderId = cartService.orderCartItem(cartOrderDtoList,
34             principal.getName()); .....❸
35             return new ResponseEntity<Long>(orderId, HttpStatus.OK); .....❹
36         }
37     }
38 }
```

8.3 장바구니 상품 주문

resources/templates/cart/cartList.html

```
01 <script th:inline="javascript">
02
03     function orders(){
04         var token = $("meta[name='_csrf']").attr("content");
05         var header = $("meta[name='_csrf_header']").attr("content");
06
07         var url = "/cart/orders";
08
09         var dataList = new Array();
10         var paramData = new Object();
11
12         $("input[name=cartChkBox]:checked").each(function() { .....❶
13             var cartItemId = $(this).val();
14             var data = new Object();
15             data["cartItemId"] = cartItemId;
16             dataList.push(data);
17         });
18
```

8.3 장바구니 상품 주문

```
19     paramData['cartOrderDtoList'] = dataList; ..... ②
20
21     var param = JSON.stringify(paramData);
22
23     $.ajax({
24         url      : url,
25         type     : "POST",
26         contentType : "application/json",
27         data      : param,
28         beforeSend : function(xhr){
29             /* 데이터를 전송하기 전에 헤더에 csrf 값 설정 */
30             xhr.setRequestHeader(header, token);
31         },
32         dataType  : "json",
33         cache     : false,
34         success   : function(result, status){
35             alert("주문이 완료 되었습니다.");
36             location.href='/orders'; ..... ③
37         },
38         error     : function(jqXHR, status, error){
39
40             if(jqXHR.status == '401'){
41                 alert('로그인 후 이용해주세요');
42                 location.href='/members/login';
43             } else{
44                 alert(jqXHR.responseJSON.message);
45             }
46
47         }
48     });
49 }
50
51 </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

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

8.3 장바구니 상품 주문



[그림 8-6] 장바구니 상품 선택

8.3 장바구니 상품 주문



[그림 8-7] 장바구니 상품 주문 성공

8.3 장바구니 상품 주문



[그림 8-8] 구매 이력 확인

Thank you for your attention

© 2021. 변구훈 & 로드북 all rights reserved.

이 콘텐츠의 저작권은 조휘웅과 로드북에 있습니다.
재배포가 가능하지만 저작권자 표시 및 콘텐츠 시작 부분에 나오는 표지를 반드시 실어야 합니다.
수정하여 재배포할 시에는 수정한 부분을 반드시 명시해야 합니다.