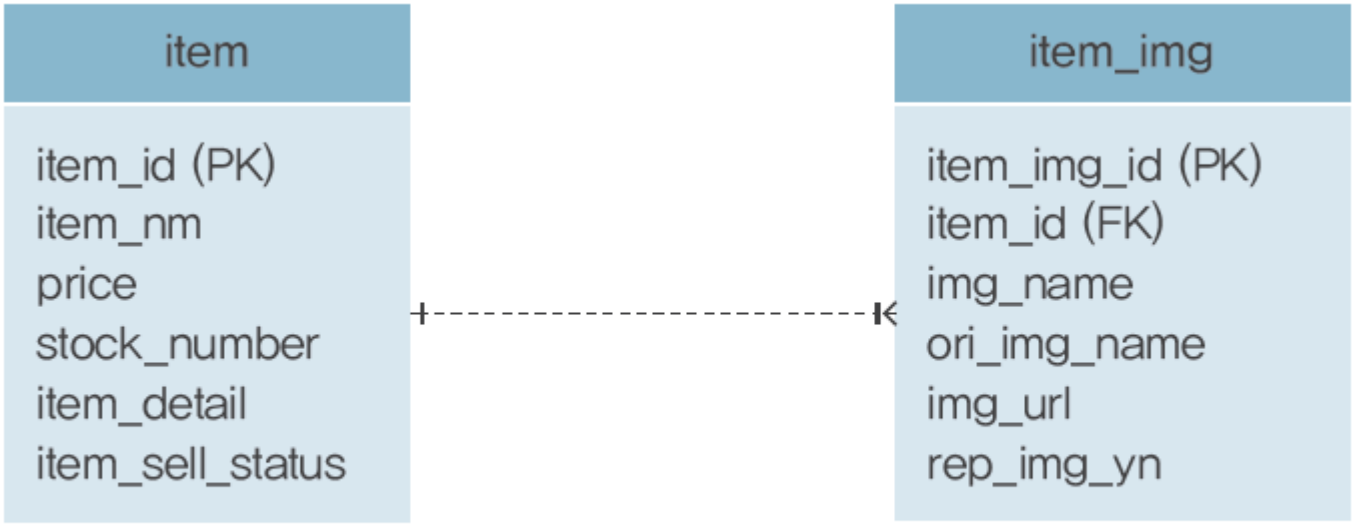
학습 목표

- 1. 상품 등록 및 수정 기능을 구현하면서 Spring DATA JPA를 이용해 데이터를 처리하는 방법을 학습한다.
- 2. Querydsl을 이용해 등록한 상품 데이터를 다양한 조건에 따라서 조회하는 방법을 학습한다.

6.1 상품 등록하기

1. 상품 이미지 엔티티는 이미지 파일명, 원본 이미지 파일명, 이미지 조회 경로, 대표 이미지 여부를 갖도록 설계

2. 대표 이미지 여부가 "Y"인 경우 메인페이지에서 상품을 보여줄 때 사용



[그림 6-1] 상품-상품 이미지 ERD

6.1 상품 등록하기

[함께 해보요 6-1] 상품 등록 구현하기

com.shop.entity.ItemImg.java

```
01 package com.shop.entity;
02
03 import lombok.Getter;
04 import lombok.Setter;
05
06 import javax.persistence.*;
07
08 @Entity
09 @Table(name="item_img")
10 @Getter @Setter
11 public class ItemImg extends BaseEntity{
12
13     @Id
14     @Column(name="item_img_id")
15     @GeneratedValue(strategy = GenerationType.AUTO)
16     private Long id;
17
18     private String imgName;  //이미지 파일명
19
20     private String oriImgName;  //원본 이미지 파일명
21
22     private String imgUrl;  //이미지 조회 경로
23
24     private String repimgYn; //대표 이미지 여부
25
26     @ManyToOne(fetch = FetchType.LAZY) ❶
27     @JoinColumn(name = "item_id")
28     private Item item;
29
30     public void updateItemImg(String oriImgName, String imgName, String imgUrl){ ❷
31         this.oriImgName = oriImgName;
32         this.imgName = imgName;
33         this.imgUrl = imgUrl;
34     }
35
36 }
```

6.1 상품 등록하기

- DTO 객체와 Entity객체의 변환을 도와주는 modelmapper라이브러리 추가
- 서로 다른 클래스의 값을 필드의 이름과 자료형이 같으면 getter, setter를 통해 값을 복사해서 객체를 반환

pom.xml

```
<dependency>
    <groupId>org.modelmapper</groupId>
    <artifactId>modelmapper</artifactId>
    <version>2.3.9</version>
</dependency>
```

6.1 상품 등록하기

com.shop.dto.ItemImgDto.java

```
01 package com.shop.dto;
02
03 import com.shop.entity.ItemImg;
04 import lombok.Getter;
05 import lombok.Setter;
06 import org.modelmapper.ModelMapper;
07
08 @Getter @Setter
09 public class ItemImgDto {
10
11     private Long id;
12
13     private String imgName;
14
15     private String oriImgName;
16
17     private String imgUrl;
18
19     private String repImgYn;
20
21     private static ModelMapper modelMapper = new ModelMapper();  ❶
22
23     public static ItemImgDto of(ItemImg itemImg){
24         return modelMapper.map(itemImg,ItemImgDto.class);  ❷
25     }
26
27 }
```

6.1 상품 등록하기

com.shop.dto.ItemFormDto.java

```
01 package com.shop.dto;
02
03 import com.shop.constant.ItemSellStatus;
04 import com.shop.entity.Item;
05 import lombok.Getter;
06 import lombok.Setter;
07 import org.modelmapper.ModelMapper;
08
09 import javax.validation.constraints.NotBlank;
10 import javax.validation.constraints.NotNull;
11 import java.util.ArrayList;
12 import java.util.List;
13
14 @Getter @Setter
15 public class ItemFormDto {
16
17     private Long id;
18
19     @NotBlank(message = "상품명은 필수 입력 값입니다.")
20     private String itemNm;
21
22     @NotNull(message = "가격은 필수 입력 값입니다.")
23     private Integer price;
24
25     @NotBlank(message = "이름은 필수 입력 값입니다.")
26     private String itemDetail;
27
```

6.1 상품 등록하기

```
28      @NotNull(message = "재고는 필수 입력 값입니다.")
29      private Integer stockNumber;
30
31      private ItemSellStatus itemSellStatus;
32
33      private List<ItemImgDto> itemImgDtoList = new ArrayList<>();  ①
34
35      private List<Long> itemImgIds = new ArrayList<>();  ②
36
37      private static ModelMapper modelMapper = new ModelMapper();
38
39      public Item createItem(){
40          return modelMapper.map(this, Item.class);  ③
41      }
42
43      public static ItemFormDto of(Item item){
44          return modelMapper.map(item,ItemFormDto.class);  ④
45      }
46
47  }
```

6.1 상품 등록하기

com.shop.ItemController.java

```
01 package com.shop.controller;
02
03 .....기존 임포트 생략.....
04
05 import org.springframework.ui.Model;
06 import com.shop.dto.ItemFormDto;
07
08 @Controller
09 public class ItemController {
10
11     @GetMapping(value = "/admin/item/new")
12     public String itemForm(Model model){
13         model.addAttribute("itemFormDto", new ItemFormDto());
14         return "item/itemForm";
15     }
16
17 }
```

6.1 상품 등록하기

- 상품 등록 같은 관리자 페이지에서 중요한 것은 데이터의 무결성을 보장하는 것이 중요
- 잘못된 값이 저장되지 않도록 밸리데이션(validation)
- 데이터끼리 서로 연관이 있으면 어떤 데이터가 변함에 따라서 다른 데이터도 함께 체크를 해야 하는 경우가 많음
- 소스 양이 많기 때문에 깃허브에서 복사해서 사용
- 깃허브 주소 : https://github.com/roadbook2/shop

6.1 상품 등록하기

resources/templates/item/itemForm.html

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

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

    <script th:inline="javascript">
        $(document).ready(function(){
            var errorMessage = [[${errorMessage}]];
            if(errorMessage != null){
                alert(errorMessage);
            }

            bindDomEvent();

        });
```

6.1 상품 등록하기

```
20          function bindDomEvent(){
21              $(".custom-file-input").on("change", function() {
22                  var fileName = $(this).val().split("\\").pop();  //이미지 파일명
23                  var fileExt = fileName.substring(fileName.lastIndexOf(".")+1);
                    // 확장자 추출
24                  fileExt = fileExt.toLowerCase(); //소문자 변환
25
26                  if(fileExt != "jpg" && fileExt != "jpeg" && fileExt != "gif"
                       && fileExt != "png" && fileExt != "bmp"){
27                      alert("이미지 파일만 등록이 가능합니다.");
28                      return;
29                  }
30
31                  $(this).siblings(".custom-file-label").html(fileName);
32              });
33          }
34
35      </script>
36
37  </th:block>
```

6.1 상품 등록하기

```
<!-- 사용자 CSS 추가 -->
<th:block layout:fragment="css">
    <style>
        .input-group {
            margin-bottom : 15px
        }
        .img-div {
            margin-bottom : 10px
        }
        .fieldError {
            color: #bd2130;
        }
    </style>
</th:block>

<div layout:fragment="content">

    <form role="form" method="post" enctype="multipart/form-data"
          th:object="${itemFormDto}">

        <p class="h2">
            상품 등록
        </p>

        <input type="hidden" th:field="*{id}">

        <div class="form-group">
            <select th:field="*{itemSellStatus}" class="custom-select">
                <option value="SELL">판매중</option>
                <option value="SOLD_OUT">품절</option>
            </select>
        </div>
```

6.1 상품 등록하기

```
<!-- 사용자 CSS 추가 -->
<th:block layout:fragment="css">
    <style>
        .input-group {
            margin-bottom : 15px
        }
        .img-div {
            margin-bottom : 10px
        }
        .fieldError {
            color: #bd2130;
        }
    </style>
</th:block>

<div layout:fragment="content">

    <form role="form" method="post" enctype="multipart/form-data"
          th:object="${itemFormDto}">

        <p class="h2">
            상품 등록
        </p>

        <input type="hidden" th:field="*{id}">

        <div class="form-group">
            <select th:field="*{itemSellStatus}" class="custom-select">
                <option value="SELL">판매중</option>
                <option value="SOLD_OUT">품절</option>
            </select>
        </div>
```

6.1 상품 등록하기

```
71          <div class="input-group">
72              <div class="input-group-prepend">
73                  <span class="input-group-text">상품명</span>
74              </div>
75              <input type="text" th:field="*{itemNm}" class="form-control"
                       placeholder="상품명을 입력해주세요">
76          </div>
77          <p th:if="${#fields.hasErrors('itemNm')}" th:errors="*{itemNm}"
             class="fieldError">Incorrect data</p>
78
79          <div class="input-group">
80              <div class="input-group-prepend">
81                  <span class="input-group-text">가격</span>
82              </div>
83              <input type="number" th:field="*{price}" class="form-control"
                       placeholder="상품의 가격을 입력해주세요">
84          </div>
85          <p th:if="${#fields.hasErrors('price')}" th:errors="*{price}"
               class="fieldError">Incorrect data</p>
86
87          <div class="input-group">
88              <div class="input-group-prepend">
89                  <span class="input-group-text">재고</span>
```

6.1 상품 등록하기

```
90              </div>
91              <input type="number" th:field="*{stockNumber}"
                       class="form-control" placeholder="상품의 재고를 입력해주세요">
92          </div>
93          <p th:if="${#fields.hasErrors('stockNumber')}" th:errors="*{stockNumber}"
               class="fieldError">Incorrect data</p>
94
95          <div class="input-group">
96              <div class="input-group-prepend">
97                  <span class="input-group-text">상품 상세 내용</span>
98              </div>
99              <textarea class="form-control" aria-label="With textarea"
                          th:field="*{itemDetail}"></textarea>
100         </div>
101         <p th:if="${#fields.hasErrors('itemDetail')}" th:errors="*{itemDetail}"
               class="fieldError">Incorrect data</p>
102
103         <div th:if="${#lists.isEmpty(itemFormDto.itemImgDtoList)}">
104             <div class="form-group" th:each="num: ${#numbers.sequence(1,5)}">
105                 <div class="custom-file img-div">
106                     <input type="file" class="custom-file-input"
                               name="itemImgFile">
107                     <label class="custom-file-label"
                               th:text="상품이미지 + ${num}"></label>
108                 </div>
109             </div>
110         </div>
```

6.1 상품 등록하기

```
120         </div>
121
122         <div th:if="${#strings.isEmpty(itemFormDto.id)}"
                 style="text-align: center">
123             <button th:formaction="@{/admin/item/new}" type="submit"
                        class="btn btn-primary">저장</button>
124         </div>
125         <div th:unless="${#strings.isEmpty(itemFormDto.id)}"
                 style="text-align: center">
126             <button th:formaction="@{'/admin/item/' + ${itemFormDto.id} }"
                        type="submit" class="btn btn-primary">수정</button>
127         </div>
128         <input type="hidden" th:name="${_csrf.parameterName}"
                   th:value="${_csrf.token}">
129
130     </form>
131
132 </div>
133
134 </html>
```

6.1 상품 등록하기

- 현재는 application을 재실행하면 데이터가 삭제되므로 데이터베이스에 저장된 데이터가 삭제되지 않도록 ddl-auto 설정 변경
- application.prperties의 ddl-auto 속성을 validate로 변경하면 애플리케이션 실행 시점에 엔티티와 테이블이 매핑이 정상적으로 되어있는지만 확인

application.properties 설정 변경하기

```
spring.jpa.hibernate.ddl-auto=validate
```

application-test.properties 설정 추가하기

```
spring.jpa.hibernate.ddl-auto=create
```

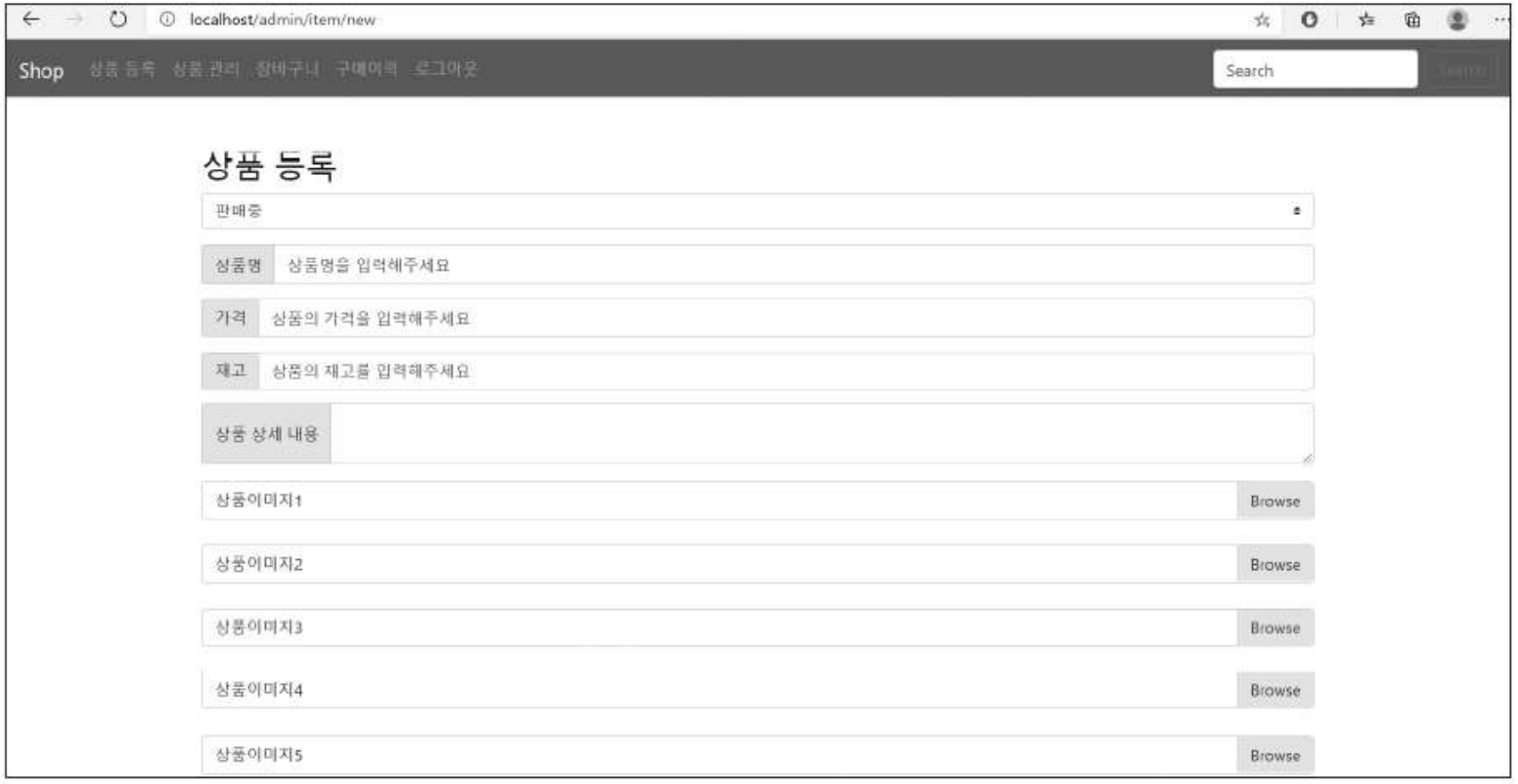
6.1 상품 등록하기

- 회원 가입 후 다시 로그인을 하였다면 ADMIN ROLE로 가입을 진행하였기 때문에 상품 등록 메뉴가 보이는 것을 확인

[그림 6-2] ADMIN ROLE 회원 로그인

6.1 상품 등록하기

- 상품 등록 메뉴를 클릭하면 방금 전에 만들었던 상품 등록 페이지가 보이는 것을 확인 가능



[그림 6-3] 상품 등록 페이지

6.1 상품 등록하기

- 이미지 파일을 등록할 때 서버에서 각 파일의 최대 사이즈와 한번에 다운 요청할 수 있는 파일의 크기 지정
- 어떤 경로에 저장할지를 관리하기 위해서 프로퍼티에itemImgLocation을 추가
- 프로젝트 내부가 아닌 자신의 컴퓨터에서 파일을 찾는 경로로 uploadPath 프로퍼

application.properties 설정 추가하기

```
#파일 한 개당 최대 사이즈
spring.servlet.multipart.maxFileSize=20MB

#요청당 최대 파일 크기
spring.servlet.multipart.maxRequestSize=100MB

#상품 이미지 업로드 경로
itemImgLocation=C:/shop/item

#리소스 업로드 경로
uploadPath=file:///C:/shop/
```

6.1 상품 등록하기

- addResourceHandlers 메소드를 통해서 자신의 로컬 컴퓨터에 업로드한 파일을 찾을 위치를 설정

com.shop.config.WebMvcConfig.java

```
01 package com.shop.config;
02
03 import org.springframework.beans.factory.annotation.Value;
04 import org.springframework.context.annotation.Configuration;
05 import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;
06 import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;
07
08 @Configuration
09 public class WebMvcConfig implements WebMvcConfigurer {
10
11     @Value("${uploadPath}")                                          ①
12     String uploadPath;
13
14     @Override
15     public void addResourceHandlers(ResourceHandlerRegistry registry) {
16         registry.addResourceHandler("/images/**")                    ②
17                 .addResourceLocations(uploadPath);                   ③
18     }
19 }
```

6.1 상품 등록하기

com.shop.service.FileService.java

```
01 package com.shop.service;
02
03 import lombok.extern.java.Log;
04 import org.springframework.stereotype.Service;
05
06 import java.io.File;
07 import java.io.FileOutputStream;
08 import java.util.UUID;
09
10 @Service
11 @Log
12 public class FileService {
13
14     public String uploadFile(String uploadPath, String originalFileName,
                                byte[] fileData) throws Exception{
15         UUID uuid = UUID.randomUUID();                                    ①
16         String extension = originalFileName.substring(originalFileName
                    .lastIndexOf("."));
17         String savedFileName = uuid.toString() + extension;              ②
18         String fileUploadFullUrl = uploadPath + "/" + savedFileName;
19         FileOutputStream fos = new FileOutputStream(fileUploadFullUrl);  ③
20         fos.write(fileData);                                              ④
21         fos.close();
22         return savedFileName;                                             ⑤
23     }
```

- 파일 업로드와 삭제 처리를 담당하는 FileService 작성

6.1 상품 등록하기

```
25      public void deleteFile(String filePath) throws Exception{
26          File deleteFile = new File(filePath);                    ⑥
27
28          if(deleteFile.exists()) {                                ⑦
29              deleteFile.delete();
30              log.info("파일을 삭제하였습니다.");
31          } else {
32              log.info("파일이 존재하지 않습니다.");
33          }
34      }
35
36  }
```

6.1 상품 등록하기

- 상품의 이미지 정보를 저장하는 ItemImgRepository 인터페이스 작성

com.shop.repository.ItemImgRepository.java

```
01 package com.shop.repository;
02
03 import com.shop.entity.ItemImg;
04 import org.springframework.data.jpa.repository.JpaRepository;
05
06 public interface ItemImgRepository extends JpaRepository<ItemImg, Long> {
07
08 }
```

6.1 상품 등록하기

- 상품 이미지를 업로드하고, 상품 이미지 정보를 저장하는 ItemImgService 클래스 작성

com.shop.service.ItemImgService.java

```
01 package com.shop.service;
02
03 import com.shop.entity.ItemImg;
04 import com.shop.repository.ItemImgRepository;
05 import lombok.RequiredArgsConstructor;
06 import org.springframework.beans.factory.annotation.Value;
07 import org.springframework.stereotype.Service;
08 import org.springframework.transaction.annotation.Transactional;
09 import org.springframework.web.multipart.MultipartFile;
10 import org.thymeleaf.util.StringUtils;
11
12 @Service
13 @RequiredArgsConstructor
14 @Transactional
15 public class ItemImgService {
16
17     @Value("${itemImgLocation}")
18     private String itemImgLocation;
19
20     private final ItemImgRepository itemImgRepository;
21
22     private final FileService fileService;
```

6.1 상품 등록하기

```
24      public void saveItemImg(ItemImg itemImg, MultipartFile itemImgFile)
    throws Exception{
25          String oriImgName = itemImgFile.getOriginalFilename();
26          String imgName = "";
27          String imgUrl = "";
28
29          //파일 업로드
30          if(!StringUtils.isEmpty(oriImgName)){
31              imgName = fileService.uploadFile(itemImgLocation, oriImgName,
                            itemImgFile.getBytes());                    ..... ❷
32              imgUrl = "/images/item/" + imgName;                     ..... ❸
33          }
34
35          //상품 이미지 정보 저장
36          itemImg.updateItemImg(oriImgName, imgName, imgUrl);         ..... ❹
37          itemImgRepository.save(itemImg);                            ..... ❺
38      }
39
40  }
```

6.1 상품 등록하기

```
24      public void saveItemImg(ItemImg itemImg, MultipartFile itemImgFile)
    throws Exception{
25          String oriImgName = itemImgFile.getOriginalFilename();
26          String imgName = "";
27          String imgUrl = "";
28
29          //파일 업로드
30          if(!StringUtils.isEmpty(oriImgName)){
31              imgName = fileService.uploadFile(itemImgLocation, oriImgName,
                            itemImgFile.getBytes());                    ..... ❷
32              imgUrl = "/images/item/" + imgName;                     ..... ❸
33          }
34
35          //상품 이미지 정보 저장
36          itemImg.updateItemImg(oriImgName, imgName, imgUrl);         ..... ❹
37          itemImgRepository.save(itemImg);                            ..... ❺
38      }
39
40  }
```

6.1 상품 등록하기

- 상품을 등록하는 ItemService 클래스 작성

com.shop.service.ItemService.java

```
01 package com.shop.service;
02
03 import com.shop.dto.ItemFormDto;
04 import com.shop.entity.Item;
05 import com.shop.entity.ItemImg;
06 import com.shop.repository.ItemImgRepository;
07 import com.shop.repository.ItemRepository;
08 import lombok.RequiredArgsConstructor;
09 import org.springframework.stereotype.Service;
10 import org.springframework.transaction.annotation.Transactional;
11 import org.springframework.web.multipart.MultipartFile;
12
13 import java.util.List;
14
15 @Service
16 @Transactional
17 @RequiredArgsConstructor
18 public class ItemService {
19
20     private final ItemRepository itemRepository;
21     private final ItemImgService itemImgService;
22     private final ItemImgRepository itemImgRepository;
23
24     public Long saveItem(ItemFormDto itemFormDto,
   List<MultipartFile> itemImgFileList) throws Exception{
```

6.1 상품 등록하기

com.shop.service.ItemService.java

```
01 package com.shop.service;
02
03 import com.shop.dto.ItemFormDto;
04 import com.shop.entity.Item;
05 import com.shop.entity.ItemImg;
06 import com.shop.repository.ItemImgRepository;
07 import com.shop.repository.ItemRepository;
08 import lombok.RequiredArgsConstructor;
09 import org.springframework.stereotype.Service;
10 import org.springframework.transaction.annotation.Transactional;
11 import org.springframework.web.multipart.MultipartFile;
12
13 import java.util.List;
14
15 @Service
16 @Transactional
17 @RequiredArgsConstructor
18 public class ItemService {
19
20     private final ItemRepository itemRepository;
21     private final ItemImgService itemImgService;
22     private final ItemImgRepository itemImgRepository;
23
24     public Long saveItem(ItemFormDto itemFormDto,
   List<MultipartFile> itemImgFileList) throws Exception{
```

- 상품을 등록하는 ItemService 클래스 작성

6.1 상품 등록하기

```
26          //상품 등록
27          Item item = itemFormDto.createItem();                    ①
28          itemRepository.save(item);                               ②
29
30          //이미지 등록
31          for(int i=0;i<itemImgFileList.size();i++){
32              ItemImg itemImg = new ItemImg();
33              itemImg.setItem(item);
34              if(i == 0)                                           ③
35                  itemImg.setRepimgYn("Y");
36              else
37                  itemImg.setRepimgYn("N");
38              itemImgService.saveItemImg(itemImg, itemImgFileList.get(i));  ④
39          }
40
41          return item.getId();
42      }
43
44  }
```

6.1 상품 등록하기

com.shop.controller.ItemController.java

```
01 package com.shop.controller;
02
03 .....기존 임포트 생략.....
04
05 import com.shop.service.ItemService;
06 import lombok.RequiredArgsConstructor;
07 import org.springframework.web.bind.annotation.PostMapping;
08 import javax.validation.Valid;
09 import org.springframework.validation.BindingResult;
10 import org.springframework.web.bind.annotation.RequestParam;
11 import org.springframework.web.multipart.MultipartFile;
12 import java.util.List;
13
14 @Controller
15 @RequiredArgsConstructor
16 public class ItemController {
17
18     private final ItemService itemService;
19
20     .....코드 생략.....
21
```

6.1 상품 등록하기

```
22      @PostMapping(value = "/admin/item/new")
23      public String itemNew(@Valid ItemFormDto itemFormDto, BindingResult
        bindingResult, Model model, @RequestParam("itemImgFile") List<MultipartFile>
        itemImgFileList){
24
25          if(bindingResult.hasErrors()){ ..... ❶
26              return "item/itemForm";
27          }
28
29          if(itemImgFileList.get(0).isEmpty() && itemFormDto.getId() == null){ ❷
30              model.addAttribute("errorMessage", "첫번째 상품 이미지는 필수 입력 값 입니다.");
31              return "item/itemForm";
32          }
33
34          try {
35              itemService.saveItem(itemFormDto, itemImgFileList); ..... ❸
36          } catch (Exception e){
37              model.addAttribute("errorMessage", "상품 등록 중 에러가 발생하였습니다.");
38              return "item/itemForm";
39          }
40
41          return "redirect:/"; ..... ❹
42      }
43
44  }
```

6.1 상품 등록하기

- 상품 저장 로직 테스트 코드를 작성
- 테스트 코드를 작성하기 위해서 ItemImgRepository 인터페이스에 findByItemIdOrderByIdAsc 메소드를 추가
- 상품 이미지 아이디의 오름차순으로 가져오는 쿼리 메소드

com.shop.repository.ItemImgRepository.java

```
01 package com.shop.repository;
02
03 .....기존 임포트 생략.....
04
05 import java.util.List;
06
07 public interface ItemImgRepository extends JpaRepository<ItemImg, Long> {
08
09     List<ItemImg> findByItemIdOrderByIdAsc(Long itemId);
10
11 }
```

6.1 상품 등록하기

com.shop.service.ItemServiceTest.java

```
01 package com.shop.service;
02
03 import com.shop.constant.ItemSellStatus;
04 import com.shop.dto.ItemFormDto;
05 import com.shop.entity.Item;
06 import com.shop.entity.ItemImg;
07 import com.shop.repository.ItemImgRepository;
08 import com.shop.repository.ItemRepository;
09 import org.junit.jupiter.api.DisplayName;
10 import org.junit.jupiter.api.Test;
11 import org.springframework.beans.factory.annotation.Autowired;
12 import org.springframework.boot.test.context.SpringBootTest;
13 import org.springframework.mock.web.MockMultipartFile;
14 import org.springframework.security.test.context.support.WithMockUser;
15 import org.springframework.test.context.TestPropertySource;
16 import org.springframework.transaction.annotation.Transactional;
17 import org.springframework.web.multipart.MultipartFile;
18
19 import javax.persistence.EntityNotFoundException;
20 import java.util.ArrayList;
21 import java.util.List;
22
```

6.1 상품 등록하기

```
23 import static org.junit.jupiter.api.Assertions.assertEquals;
24
25 @SpringBootTest
26 @Transactional
27 @TestPropertySource(locations="classpath:application-test.properties")
28 class ItemServiceTest {
29
30     @Autowired
31     ItemService itemService;
32
33     @Autowired
34     ItemRepository itemRepository;
35
36     @Autowired
37     ItemImgRepository itemImgRepository;
38
39     List<MultipartFile> createMultipartFiles() throws Exception{ ①
40
41         List<MultipartFile> multipartFileList = new ArrayList<>();
42
43         for(int i=0;i<5;i++){
44             String path = "C:/shop/item/";
45             String imageName = "image" + i + ".jpg";
46             MockMultipartFile multipartFile =
                                    new MockMultipartFile(path, imageName,
                                    "image/jpg", new byte[]{1,2,3,4});
47             multipartFileList.add(multipartFile);
48         }
49
50         return multipartFileList;
51     }
```

6.1 상품 등록하기

```
53      @Test
54      @DisplayName("상품 등록 테스트")
55      @WithMockUser(username = "admin", roles = "ADMIN")
56      void saveItem() throws Exception{
57          ItemFormDto itemFormDto = new ItemFormDto();                    ❷
58          itemFormDto.setItemNm("테스트상품");
59          itemFormDto.setItemSellStatus(ItemSellStatus.SELL);
60          itemFormDto.setItemDetail("테스트 상품 입니다.");
61          itemFormDto.setPrice(1000);
62          itemFormDto.setStockNumber(100);
63
64          List<MultipartFile> multipartFileList = createMultipartFiles();
65          Long itemId = itemService.saveItem(itemFormDto, multipartFileList);  ❸
66
67          List<ItemImg> itemImgList =
                        itemImgRepository.findByItemIdOrderByIdAsc(itemId);
68          Item item = itemRepository.findById(itemId)
69                  .orElseThrow(EntityNotFoundException::new);
70
71          assertEquals(itemFormDto.getItemNm(), item.getItemNm());        ❹
72          assertEquals(itemFormDto.getItemSellStatus(),
                        item.getItemSellStatus());
73          assertEquals(itemFormDto.getItemDetail(), item.getItemDetail());
74          assertEquals(itemFormDto.getPrice(), item.getPrice());
75          assertEquals(itemFormDto.getStockNumber(), item.getStockNumber());
76          assertEquals(multipartFileList.get(0).getOriginalFilename(),
                        itemImgList.get(0).getOriImgName());                ❺
77      }
78  }
```

6.1 상품 등록하기

[그림 6-4] 상품 등록 테스트

6.1 상품 등록하기

[그림 6-5] 상품 데이터 입력

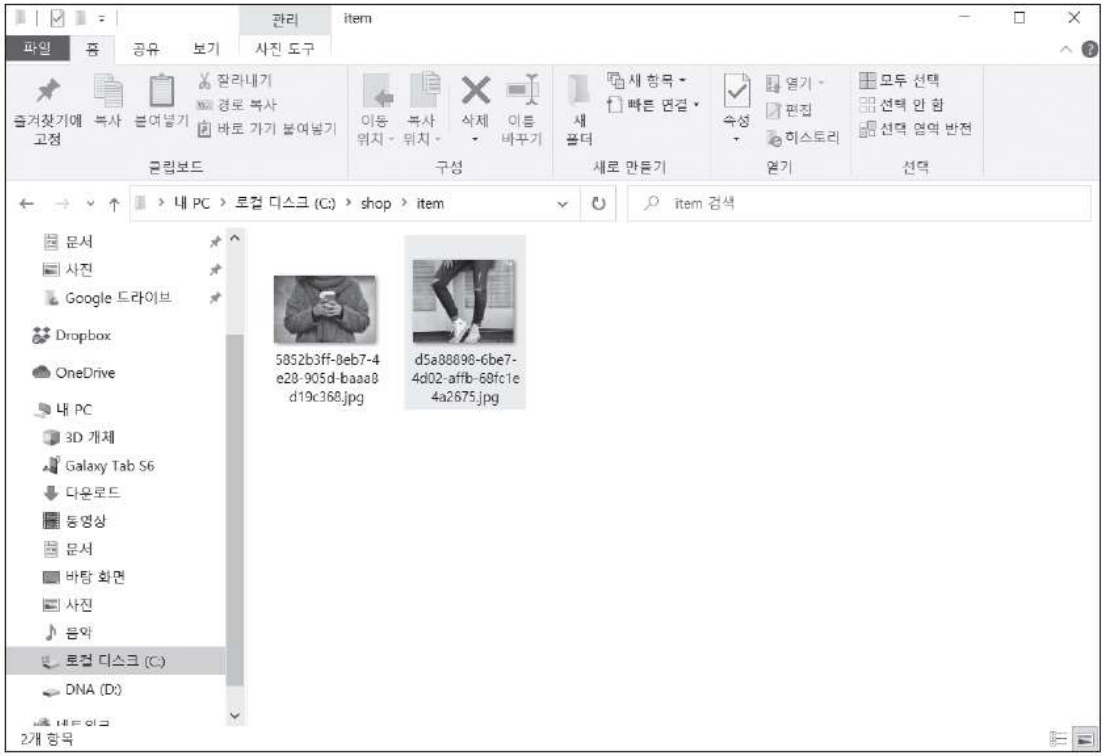
6.1 상품 등록하기

- <저장> 버튼을 눌렀을 때 상품이 정상적으로 저장됐다면 다음과 같이 메인 페이지로 이동

[그림 6-6] 상품 저장 시 메인 페이지 이동

6.1 상품 등록하기

- 파일 업로드 경로인 C:/shop/item 경로에 들어가보면 업로드한 청바진 사진과 스웨터 사진이 올라온것을 확인



[그림 6-7] 상품 등록 후 이미지 저장 결과

6.2 상품 수정하기

- 상품 등록 후 콘솔창을 보면 insert into item 쿼리문에서 item_id에 들어가는 binding parameter 값 확인
- 해당 상품 아이디를 이용해서 상품 수정 페이지에 진입 예제 진행

```
Hibernate:
    insert
    into
        item
        (reg_time, update_time, created_by, modified_by, item_detail, item_nm, item_sell_status, price, stock_number, item_id)
    values
        (?, ?, ?, ?, ?, ?, ?, ?, ?, ?)
2021-03-05 13:48:36.908 TRACE 8936 --- [nio-80-exec-10] o.h.type.descriptor.sql.BasicBinder      : binding parameter [1] as [TIMESTAMP] - [2021-03-05T13:48:36.823002800]
2021-03-05 13:48:36.908 TRACE 8936 --- [nio-80-exec-10] o.h.type.descriptor.sql.BasicBinder      : binding parameter [2] as [TIMESTAMP] - [2021-03-05T13:48:36.823002800]
2021-03-05 13:48:36.909 TRACE 8936 --- [nio-80-exec-10] o.h.type.descriptor.sql.BasicBinder      : binding parameter [3] as [VARCHAR] - [test@test.com]
2021-03-05 13:48:36.909 TRACE 8936 --- [nio-80-exec-10] o.h.type.descriptor.sql.BasicBinder      : binding parameter [4] as [VARCHAR] - [test@test.com]
2021-03-05 13:48:36.909 TRACE 8936 --- [nio-80-exec-10] o.h.type.descriptor.sql.BasicBinder      : binding parameter [5] as [CLOB] - [상품 등록 테스트입니다.]
2021-03-05 13:48:36.910 TRACE 8936 --- [nio-80-exec-10] o.h.type.descriptor.sql.BasicBinder      : binding parameter [6] as [VARCHAR] - [테스트상품]
2021-03-05 13:48:36.911 TRACE 8936 --- [nio-80-exec-10] o.h.type.descriptor.sql.BasicBinder      : binding parameter [7] as [VARCHAR] - [SELL]
2021-03-05 13:48:36.911 TRACE 8936 --- [nio-80-exec-10] o.h.type.descriptor.sql.BasicBinder      : binding parameter [8] as [INTEGER] - [10000]
2021-03-05 13:48:36.911 TRACE 8936 --- [nio-80-exec-10] o.h.type.descriptor.sql.BasicBinder      : binding parameter [9] as [INTEGER] - [100]
2021-03-05 13:48:36.911 TRACE 8936 --- [nio-80-exec-10] o.h.type.descriptor.sql.BasicBinder      : binding parameter [10] as [BIGINT] - [2]
```

[그림 6-8] 상품 등록 후 이미지 저장 결과

6.2 상품 수정하기

[함께 해봐요 6-2] 상품 수정하기

com.shop.service.ItemService.java

```
01 package com.shop.service;
02
03 .....기존 임포트 생략.....
04
05 import com.shop.dto.ItemImgDto;
06 import javax.persistence.EntityNotFoundException;
07 import java.util.ArrayList;
08
09 @Service
10 @Transactional
11 @RequiredArgsConstructor
12 public class ItemService {
13
14     .....코드 생략.....
```

6.2 상품 수정하기

```
19          List<ItemImg> itemImgList =
    itemImgRepository.findByItemIdOrderByIdAsc(itemId);                    ②
20          List<ItemImgDto> itemImgDtoList = new ArrayList<>();
21          for (ItemImg itemImg : itemImgList) {                          ③
22              ItemImgDto itemImgDto = ItemImgDto.of(itemImg);
23              itemImgDtoList.add(itemImgDto);
24          }
25
26          Item item = itemRepository.findById(itemId)                    ④
27                  .orElseThrow(EntityNotFoundException::new);
28          ItemFormDto itemFormDto = ItemFormDto.of(item);
29          itemFormDto.setItemImgDtoList(itemImgDtoList);
30          return itemFormDto;
31      }
32
33  }
```

6.2 상품 수정하기

com.shop.controller.ItemController.java

```
01 package com.shop.controller;
02
03 .....기존 임포트 생략.....
04
05 import org.springframework.web.bind.annotation.PathVariable;
06 import javax.persistence.EntityNotFoundException;
07
08 @Controller
09 @RequiredArgsConstructor
10 public class ItemController {
11
12     .....코드 생략.....
13
14     @GetMapping(value = "/admin/item/{itemId}")
15     public String itemDtl(@PathVariable("itemId") Long itemId, Model model){
16
17         try {
18             ItemFormDto itemFormDto = itemService.getItemDtl(itemId); ------❶
19             model.addAttribute("itemFormDto", itemFormDto);
20         } catch(EntityNotFoundException e){ ----------------------------------❷
21             model.addAttribute("errorMessage", "존재하지 않는 상품 입니다.");
22 model.addAttribute("itemFormDto", new ItemFormDto());
23             return "item/itemForm";
24         }
25
26         return "item/itemForm";
27     }
28 }
```

6.2 상품 수정하기

- 상품 아이디는 insert into item 쿼리문에서 각자 확인한 상품 아이디 입력
- 저장한 상품을 조회하기 위해서 웹 브라우저에 http://localhost/admin/item/{상품번호} 입력

6.2 상품 수정하기

- 상품 아이디가 2번으로 발급됐을 경우 localhost/admin/item/2 입력

[그림 6-9] 상품 수정 페이지

6.2 상품 수정하기

com.shop.service.ItemImgService.java

```
01 package com.shop.service;
02
03 .....기존 임포트 생략.....
04
05 import javax.persistence.EntityNotFoundException;
06
07 @Service
08 @RequiredArgsConstructor
09 @Transactional
10 public class ItemImgService {
11
12     .....코드 생략.....
13
14     public void updateItemImg(Long itemImgId, MultipartFile itemImgFile)
   throws Exception{
15         if(!itemImgFile.isEmpty()){ ..... ❶
16             ItemImg savedItemImg = itemImgRepository.findById(itemImgId) ... ❷
17                     .orElseThrow(EntityNotFoundException::new);
18
```

6.2 상품 수정하기

com.shop.service.ItemImgService.java

```
01 package com.shop.service;
02
03 .....기존 임포트 생략.....
04
05 import javax.persistence.EntityNotFoundException;
06
07 @Service
08 @RequiredArgsConstructor
09 @Transactional
10 public class ItemImgService {
11
12     .....코드 생략.....
13
14     public void updateItemImg(Long itemImgId, MultipartFile itemImgFile)
   throws Exception{
15         if(!itemImgFile.isEmpty()){ ..... ❶
16             ItemImg savedItemImg = itemImgRepository.findById(itemImgId) ... ❷
17                     .orElseThrow(EntityNotFoundException::new);
18
```

6.2 상품 수정하기

```
19              //기존 이미지 파일 삭제
20              if(!StringUtils.isEmpty(savedItemImg.getImgName())) {          ③
21                  fileService.deleteFile(itemImgLocation+"/"+
                                           savedItemImg.getImgName());
22              }
23
24              String oriImgName = itemImgFile.getOriginalFilename();
25              String imgName = fileService.uploadFile(itemImgLocation,
                                 oriImgName, itemImgFile.getBytes());          ④
26              String imgUrl = "/images/item/" + imgName;
27              savedItemImg.updateItemImg(oriImgName, imgName, imgUrl);       ⑤
28          }
29      }
30
31  }
```

6.2 상품 수정하기

```
19              //기존 이미지 파일 삭제
20              if(!StringUtils.isEmpty(savedItemImg.getImgName())) {          ③
21                  fileService.deleteFile(itemImgLocation+"/"+
                                           savedItemImg.getImgName());
22              }
23
24              String oriImgName = itemImgFile.getOriginalFilename();
25              String imgName = fileService.uploadFile(itemImgLocation,
                                 oriImgName, itemImgFile.getBytes());          ④
26              String imgUrl = "/images/item/" + imgName;
27              savedItemImg.updateItemImg(oriImgName, imgName, imgUrl);       ⑤
28          }
29      }
30
31  }
```

6.2 상품 수정하기

com.shop.entity.Item.java

```
01 package com.shop.entity;
02
03 .....기존 임포트 생략.....
04
05 import com.shop.dto.ItemFormDto;
06
07 @Entity
08 @Table(name="item")
09 @Getter @Setter
10 @ToString
11 public class Item extends BaseEntity{
12
13     .....코드 생략.....
14
15     public void updateItem(ItemFormDto itemFormDto){
16         this.itemNm = itemFormDto.getItemNm();
17         this.price = itemFormDto.getPrice();
18         this.stockNumber = itemFormDto.getStockNumber();
19         this.itemDetail = itemFormDto.getItemDetail();
20         this.itemSellStatus = itemFormDto.getItemSellStatus();
21     }
22
23 }
```

- 상품 업데이트 로직 구현

6.2 상품 수정하기

com.shop.service.ItemService.java

```
01 package com.shop.service;
02
03 .....기존 임포트 생략.....
04
05 @Service
06 @Transactional
07 @RequiredArgsConstructor
08 public class ItemService {
09
10     .....코드 생략.....
11
12     public Long updateItem(ItemFormDto itemFormDto,
   List<MultipartFile> itemImgFileList) throws Exception{
13
14         //상품 수정
15         Item item = itemRepository.findById(itemFormDto.getId())  ❶
16                 .orElseThrow(EntityNotFoundException::new);
17         item.updateItem(itemFormDto);  ❷
18
19         List<Long> itemImgIds = itemFormDto.getItemImgIds();  ❸
20
21         //이미지 등록
22         for(int i=0;i<itemImgFileList.size();i++){
23             itemImgService.updateItemImg(itemImgIds.get(i),
   itemImgFileList.get(i));  ❹
24         }
25
26         return item.getId();
27     }
28
29 }
```

- 상품 업데이트 시 변경 감지 기능 활용

6.2 상품 수정하기

com.shop.controller.ItemController.java

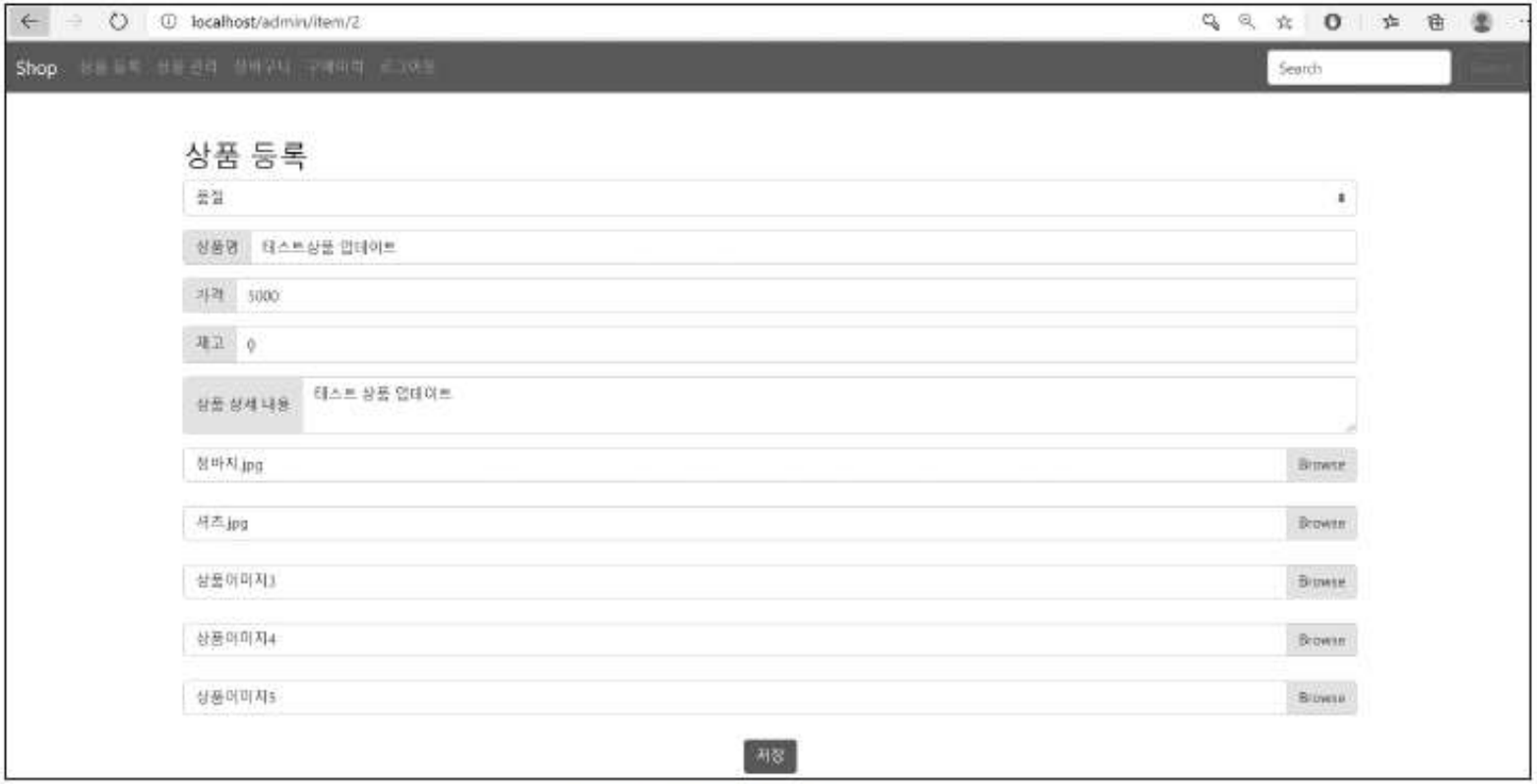
```
01 package com.shop.controller;
02
03 .....기존 임포트 생략.....
04
05 @Controller
06 @RequiredArgsConstructor
07 public class ItemController {
08
09     .....코드 생략.....
10
11     @PostMapping(value = "/admin/item/{itemId}")
12     public String itemUpdate(@Valid ItemFormDto itemFormDto,
   BindingResult bindingResult, @RequestParam("itemImgFile") List<MultipartFile>
   itemImgFileList, Model model){
13
14         if(bindingResult.hasErrors()){
15             return "item/itemForm";
16         }
17
```

6.2 상품 수정하기

```
18          if(itemImgFileList.get(0).isEmpty() && itemFormDto.getId() == null){
19              model.addAttribute("errorMessage", "첫번째 상품 이미지는 필수 입력 값 입니다.");
20              return "item/itemForm";
21          }
22
23          try {
24              itemService.updateItem(itemFormDto, itemImgFileList);  ①
25          } catch (Exception e){
26              model.addAttribute("errorMessage", "상품 수정 중 에러가 발생하였습니다.");
27              return "item/itemForm";
28          }
29
30          return "redirect:/";
31      }
32
33  }
```

6.2 상품 수정하기

- 상품 수정 시 데이터가 변경되는 것을 확인 가능



[그림 6-10] 상품 수정 결과

6.3 상품 관리하기

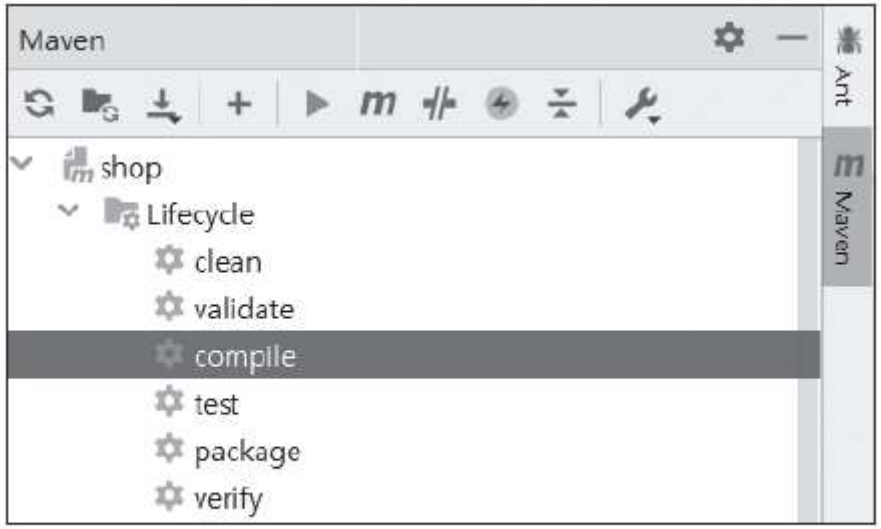
- 상품 관리 화면에서는 상품을 조회하는 조건을 설정 후 페이징 기능을 통해 일정 개수의 상품만 불러오며, 선택한 상품 상세 페이지로 이동할 수 있는 기능까지 구현
- 조회 조건
  - 상품 등록일
  - 상품 판매 상태
  - 상품명 또는 상품 등록자 아이디

6.3 상품 관리하기

- 조회 조건이 복잡한 경우 Querydsl을 이용해 조건에 맞는 쿼리를 동적으로 쉽게 생성
- Querydsl을 사용하면 비슷한 쿼리를 재활용할 수 있음
- 자바 코드로 작성하기 때문에 IDE의 도움을 받아서 문법 오류를 바로 수정 가능

6.3 상품 관리하기

- QDomain을 생성하기 위해서 메이븐의 컴파일 명령 실행



[그림 6-13] 메이븐 컴파일 실행

6.3 상품 관리하기

[함께 해봐요 6-3] 상품 관리 메뉴 구현하기

com.shop.dto.ItemSearchDto.java

```
01 package com.shop.dto;
02
03 import com.shop.constant.ItemSellStatus;
04 import lombok.Getter;
05 import lombok.Setter;
06
07 @Getter @Setter
08 public class ItemSearchDto {
09
10     private String searchDateType; ..... ❶
11
12     private ItemSellStatus searchSellStatus; ..... ❷
13
14     private String searchBy; ..... ❸
15
16     private String searchQuery = ""; ..... ❹
17
18 }
```

6.3 상품 관리하기

- Querydsl을 Spring Data Jpa과 함께 사용하기 위해서 사용자 정의 리포지토리를 정의. 총 3단계의 과정으로 구현
  - 사용자 정의 인터페이스 작성
  - 사용자 정의 인터페이스 구현
  - Spring Data Jpa 리포지토리에서 사용자 정의 인터페이스 상속

6.3 상품 관리하기

- 사용자 정의 인터페이스 작성

com.shop.repository.ItemRepositoryCustom.java

```
01 package com.shop.repository;
02
03 import com.shop.dto.ItemSearchDto;
04 import com.shop.entity.Item;
05 import org.springframework.data.domain.Page;
06 import org.springframework.data.domain.Pageable;
07
08 public interface ItemRepositoryCustom {
09
10     Page<Item> getAdminItemPage(ItemSearchDto itemSearchDto, Pageable pageable); ❶
11
12 }
```

6.3 상품 관리하기

- 사용자 정의 인터페이스 구현

com.shop.repository.ItemRepositoryCustomImpl.java

```
01 package com.shop.repository;
02
03 import com.querydsl.core.QueryResults;
04 import com.querydsl.core.types.dsl.BooleanExpression;
05 import com.querydsl.jpa.impl.JPAQueryFactory;
06 import com.shop.constant.ItemSellStatus;
07 import com.shop.dto.ItemSearchDto;
08 import com.shop.entity.Item;
09 import com.shop.entity.QItem;
10
11 import org.springframework.data.domain.Page;
12 import org.springframework.data.domain.PageImpl;
13 import org.springframework.data.domain.Pageable;
14 import org.thymeleaf.util.StringUtils;
15
16 import javax.persistence.EntityManager;
17 import java.time.LocalDateTime;
18 import java.util.List;
```

6.3 상품 관리하기

```
20 public class ItemRepositoryCustomImpl implements ItemRepositoryCustom{ ❶
21
22     private JPAQueryFactory queryFactory; ❷
23
24     public ItemRepositoryCustomImpl(EntityManager em){ ❸
25         this.queryFactory = new JPAQueryFactory(em);
26     }
27
28     private BooleanExpression searchSellStatusEq(ItemSellStatus searchSellStatus){ ❹
29         return searchSellStatus ==
   null ? null : QItem.item.itemSellStatus.eq(searchSellStatus); ❺
30     }
31
32     private BooleanExpression regDtsAfter(String searchDateType){ ❻
33         LocalDateTime dateTime = LocalDateTime.now();
34
35         if(StringUtils.equals("all", searchDateType) || searchDateType == null){
36             return null;
37         } else if(StringUtils.equals("1d", searchDateType)){
38             dateTime = dateTime.minusDays(1);
39         } else if(StringUtils.equals("1w", searchDateType)){
40             dateTime = dateTime.minusWeeks(1);
41         } else if(StringUtils.equals("1m", searchDateType)){
42             dateTime = dateTime.minusMonths(1);
43         } else if(StringUtils.equals("6m", searchDateType)){
44             dateTime = dateTime.minusMonths(6);
45         }
46
47         return QItem.item.regTime.after(dateTime);
48     }
```

6.3 상품 관리하기

```
50      private BooleanExpression searchByLike(String searchBy, String searchQuery){ 6
51
52          if(StringUtils.equals("itemNm", searchBy)){
53              return QItem.item.itemNm.like("%" + searchQuery + "%");
54          } else if(StringUtils.equals("createdBy", searchBy)){
55              return QItem.item.createdBy.like("%" + searchQuery + "%");
56          }
57
58          return null;
59      }
60
61      @Override
62      public Page<Item> getAdminItemPage(ItemSearchDto itemSearchDto,
    Pageable pageable) {
63          QueryResults<Item> results = queryFactory ..... 7
64                  .selectFrom(QItem.item)
65                  .where(regDtsAfter(itemSearchDto.getSearchDateType()),
66                         searchSellStatusEq(itemSearchDto.getSearchSellStatus()),
67                         searchByLike(itemSearchDto.getSearchBy(),
    itemSearchDto.getSearchQuery()))
68                  .orderBy(QItem.item.id.desc())
69                  .offset(pageable.getOffset())
70                  .limit(pageable.getPageSize())
71                  .fetchResults();
72
73          List<Item> content = results.getResults();
74          long total = results.getTotal();
75          return new PageImpl<>(content, pageable, total); ..... 8
76      }
77  }
```

6.3 상품 관리하기

[표 6-1] Querydsl 조회 결과를 반환하는 메소드

메소드	기능
QueryResults<T> fetchResults()	조회 대상 리스트 및 전체 개수를 포함하는 QueryResults 반환
List<T> fetch()	조회 대상 리스트 반환
T fetchOne()	조회 대상이 1건이면 해당 타입 반환. 조회 대상이 1건 이상이면 에러 발생
T fetchFirst()	조회 대상이 1건 또는 1건 이상이면 1건만 반환
long fetchCount()	해당 데이터 전체 개수 반환. count 쿼리 실행

6.3 상품 관리하기

- Spring Data Jpa 리포지토리에서 사용자 정의 인터페이스 상속

com.shop.repository.ItemRepository.java

```
01 package com.shop.repository;
02
03 .....기존 임포트 생략.....
04
05 public interface ItemRepository extends JpaRepository<Item, Long>,
   QuerydslPredicateExecutor<Item>, ItemRepositoryCustom{
06
07     .....코드 생략.....
08
09 }
```

6.3 상품 관리하기

com.shop.service.ItemService.java

```
01 package com.shop.service;
02
03 .....기존 임포트 생략.....
04
05 import com.shop.dto.ItemSearchDto;
06 import org.springframework.data.domain.Page;
07 import org.springframework.data.domain.Pageable;
08
09 @Service
10 @Transactional
11 @RequiredArgsConstructor
12 public class ItemService {
13
14     .....코드 생략.....
15
16     @Transactional(readOnly = true)
17     public Page<Item> getAdminItemPage(ItemSearchDto itemSearchDto,
   Pageable pageable){
18         return itemRepository.getAdminItemPage(itemSearchDto, pageable);
19     }
20
21 }
```

- 상품 조회 조건과 페이지 정보를 파라미터로 받아서 상품 데이터를 조회하는 getAdminItemPage() 메소드를 추가
- 데이터의 수정이 일어나지 않으므로 최적화를 위해 @Transactional(readOnly=true) 어노테이션을 설정

6.3 상품 관리하기

com.shop.controller.ItemController.java

```
01 package com.shop.controller;
02
03 .....기존 임포트 생략.....
04
05 import com.shop.dto.ItemSearchDto;
06 import com.shop.entity.Item;
07 import org.springframework.data.domain.Page;
08 import org.springframework.data.domain.PageRequest;
09 import org.springframework.data.domain.Pageable;
10 import java.util.Optional;
11
12 @Controller
13 @RequiredArgsConstructor
14 public class ItemController {
15
16     .....코드 생략.....
17
18     @GetMapping(value = {"/admin/items", "/admin/items/{page}"}) ····· ❶
19     public String itemManage(ItemSearchDto itemSearchDto,
   @PathVariable("page") Optional<Integer> page, Model model){
20         Pageable pageable = PageRequest.of(page.isPresent() ? page.get() : 0, 3); ❷
21         Page<Item> items =
   itemService.getAdminItemPage(itemSearchDto, pageable); ····· ❸
22         model.addAttribute("items", items); ····· ❹
23         model.addAttribute("itemSearchDto", itemSearchDto); ····· ❺
24         model.addAttribute("maxPage", 5); ····· ❻
25         return "item/itemMng";
26     }
27 }
```

6.3 상품 관리하기

- 상품 관리 페이지 html 코드 작성

resources/templates/item/itemMng.html

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

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

        $(document).ready(function(){
            $("#searchBtn").on("click",function(e) {
                e.preventDefault();
                page(0);
            });
        });

        function page(page){
            var searchDateType = $("#searchDateType").val();
            var searchSellStatus = $("#searchSellStatus").val();
            var searchBy = $("#searchBy").val();
            var searchQuery = $("#searchQuery").val();

            location.href="/admin/items/" + page +
"?searchDateType=" + searchDateType
            + "&searchSellStatus=" + searchSellStatus
            + "&searchBy=" + searchBy
            + "&searchQuery=" + searchQuery;
        }

    </script>
</th:block>

```

6.3 상품 관리하기

```
<!-- 사용자 CSS 추가 -->
<th:block layout:fragment="css">
    <style>
        select{
            margin-right:10px;
        }
    </style>
</th:block>

<div layout:fragment="content">

    <form th:action="@{'/admin/items/' +
${items.number}}" role="form" method="get" th:object="${items}">
        <table class="table">
            <thead>
            <tr>
                <td>상품아이디</td>
                <td>상품명</td>
                <td>상태</td>
                <td>등록자</td>
                <td>등록일</td>
            </tr>
            </thead>
            <tbody>
            <tr th:each="item, status: ${items.getContent()}">
                <td th:text="${item.id}"></td>
                <td>
                    <a th:href="'/admin/item/'+${item.id}"
                       th:text="${item.itemNm}"></a>
                </td>
                <td th:text="${item.itemSellStatus ==
T(com.shop.constant.ItemSellStatus).SELL} ? '판매중' : '품절'"></td>
                <td th:text="${item.createdBy}"></td>
                <td th:text="${item.regTime}"></td>
            </tr>
            </tbody>
        </table>

```

6.3 상품 관리하기

```
67          <div th:with="start=${(items.number/maxPage)*maxPage + 1},
    end=(${(items.totalPages == 0) ? 1 : (start + (maxPage - 1)
    < items.totalPages ? start + (maxPage - 1) : items.totalPages)})" >
68              <ul class="pagination justify-content-center">
69
70                  <li class="page-item" th:classappend=
    "${items.first}?'disabled'">
71                      <a th:onclick="'javascript:page(' + ${items.number - 1}
    + ')'" aria-label='Previous' class="page-link">
72                          <span aria-hidden='true'>Previous</span>
73                      </a>
74                  </li>
75
76                  <li class="page-item" th:each=
    "page: ${#numbers.sequence(start, end)}" th:classappend=
    "${items.number eq page-1}?'active':''">
77                      <a th:onclick="'javascript:page(' + ${page - 1} + ')'"
    th:inline="text" class="page-link">[[${page}]]</a>
78                  </li>
79
80                  <li class="page-item" th:classappend="${items.last}?'disabled'">
81                      <a th:onclick="'javascript:page(' + ${items.number + 1}
    + ')'" aria-label='Next' class="page-link">
82                          <span aria-hidden='true'>Next</span>
83                      </a>
84                  </li>
85
86              </ul>
87          </div>
```

6.3 상품 관리하기

```
67          <div th:with="start=${(items.number/maxPage)*maxPage + 1},
    end=(${(items.totalPages == 0) ? 1 : (start + (maxPage - 1)
    < items.totalPages ? start + (maxPage - 1) : items.totalPages)})" >
68              <ul class="pagination justify-content-center">
69
70                  <li class="page-item" th:classappend=
    "${items.first}?'disabled'">
71                      <a th:onclick="'javascript:page(' + ${items.number - 1}
    + ')'" aria-label='Previous' class="page-link">
72                          <span aria-hidden='true'>Previous</span>
73                      </a>
74                  </li>
75
76                  <li class="page-item" th:each=
    "page: ${#numbers.sequence(start, end)}" th:classappend=
    "${items.number eq page-1}?'active':''">
77                      <a th:onclick="'javascript:page(' + ${page - 1} + ')'"
    th:inline="text" class="page-link">[[${page}]]</a>
78                  </li>
79
80                  <li class="page-item" th:classappend="${items.last}?'disabled'">
81                      <a th:onclick="'javascript:page(' + ${items.number + 1}
    + ')'" aria-label='Next' class="page-link">
82                          <span aria-hidden='true'>Next</span>
83                      </a>
84                  </li>
85
86              </ul>
87          </div>
```

6.3 상품 관리하기

```
89          <div class="form-inline justify-content-center" th:object="${itemSearchDto}">
90              <select th:field="*{searchDateType}" class="form-control"
                        style="width:auto;">
91                  <option value="all">전체기간</option>
92                  <option value="1d">1일</option>
93                  <option value="1w">1주</option>
94                  <option value="1m">1개월</option>
95                  <option value="6m">6개월</option>
96              </select>
97              <select th:field="*{searchSellStatus}" class="form-control"
                        style="width:auto;">
98                  <option value="">판매상태(전체)</option>
99                  <option value="SELL">판매</option>
100                 <option value="SOLD_OUT">품절</option>
101             </select>
102             <select th:field="*{searchBy}" class="form-control" style="width:auto;">
103                 <option value="itemNm">상품명</option>
104                 <option value="createdBy">등록자</option>
105             </select>
106             <input th:field="*{searchQuery}" type="text"
    class="form-control" placeholder="검색어를 입력해주세요">
107             <button id="searchBtn" type="submit"
    class="btn btn-primary">검색</button>
108         </div>
109     </form>
110
111 </div>
112
113 </html>
```

6.3 상품 관리하기

[그림 6-15] 상품 관리 페이지

6.4 메인 화면

- 등록한 상품을 메인 페이지에서 고객이 볼 수 있도록 구현
- 메인 페이지 구현도 상품 관리 메뉴 구현과 비슷하며, 동일하게 QuerydsI을 사용하여 페이징 처리 및 네비게이션바에 있는 Search 버튼을 이용하여 상품명으로 검색이 가능하도록 구현

6.4 메인 화면

- @QueryProjection을 이용하여 상품 조회 시 DTO 객체로 결과를 받는 예제 진행

[함께 해봐요 6-4] 메인 페이지 구현하기

com.shop.dto.MainItemDto.java

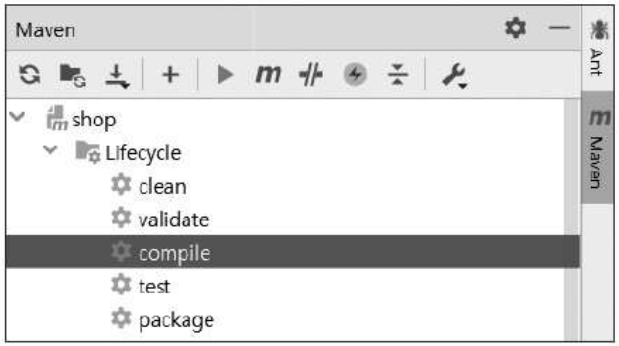
```
01 package com.shop.dto;
02
03 import com.querydsl.core.annotations.QueryProjection;
04 import lombok.Getter;
05 import lombok.Setter;
06
07 @Getter @Setter
08 public class MainItemDto {
09
10     private Long id;
11
12     private String itemNm;
13
14     private String itemDetail;
15
```

6.4 메인 화면

```
16      private String imgUrl;
17
18      private Integer price;
19
20      @QueryProjection ①
21      public MainItemDto(Long id, String itemNm, String itemDetail,
                           String imgUrl,Integer price){
22          this.id = id;
23          this.itemNm = itemNm;
24          this.itemDetail = itemDetail;
25          this.imgUrl = imgUrl;
26          this.price = price;
27      }
28
29  }
```

6.4 메인 화면

- @QueryProjection을 사용할 때 [maven compile]을 실행해 QDto 파일을 생성



[그림 6-17] maven compile

[그림 6-18] QMainItemDto 생성 확인

6.4 메인 화면

- 메인 페이지 상품 리스트 조회 쿼리 작성

com.shop.repository.ItemRepositoryCustom.java

```
01 package com.shop.repository;
02
03 .....기존 임포트 생략.....
04
05 import com.shop.dto.MainItemDto;
06
07 public interface ItemRepositoryCustom {
08
09     .....코드 생략.....
10
11     Page<MainItemDto> getMainItemPage(ItemSearchDto itemSearchDto,
                                         Pageable pageable);
12
13 }
```

6.4 메인 화면

com.shop.repository.ItemRepositoryCustomImpl.java

```
01 package com.shop.repository;
02
03 .....기존 임포트 생략.....
04
05 import com.shop.dto.MainItemDto;
06 import com.shop.dto.QMainItemDto;
07 import com.shop.entity.QItemImg;
08
09 public class ItemRepositoryCustomImpl implements ItemRepositoryCustom{
10
11     .....코드 생략.....
12
13     private BooleanExpression itemNmLike(String searchQuery){ ..... ❶
14         return StringUtils.isEmpty(searchQuery) ?
   null : QItem.item.itemNm.like("%" + searchQuery + "%");
15     }
```

6.4 메인 화면

```
17      @Override
18      public Page<MainItemDto> getMainItemPage(ItemSearchDto itemSearchDto,
                                                 Pageable pageable) {
19          QItem item = QItem.item;
20          QItemImg itemImg = QItemImg.itemImg;
21
22          QueryResults<MainItemDto> results =  queryFactory
23                  .select(
24                          new QMainItemDto( ------------------------------------❷
25                          item.id,
26                          item.itemNm,
27                          item.itemDetail,
28                          itemImg.imgUrl,
29                          item.price)
30                  )
31                  .from(itemImg)
32                  .join(itemImg.item, item) ----------------------------------❸
33                  .where(itemImg.repimgYn.eq("Y")) ---------------------------❹
34                  .where(itemNmLike(itemSearchDto.getSearchQuery()))
35                  .orderBy(item.id.desc())
36                  .offset(pageable.getOffset())
37                  .limit(pageable.getPageSize())
38                  .fetchResults();
39
40          List<MainItemDto> content = results.getResults();
41          long total = results.getTotal();
42          return new PageImpl<>(content, pageable, total);
43      }
44  }
```

6.4 메인 화면

com.shop.service.ItemService.java

```
01 package com.shop.service;
02
03 .....기존 임포트 생략.....
04
05 import com.shop.dto.MainItemDto;
06
07 @Service
08 @Transactional
09 @RequiredArgsConstructor
10 public class ItemService {
11
12     .....코드 생략.....
13
14     @Transactional(readOnly = true)
15     public Page<MainItemDto> getMainItemPage(ItemSearchDto itemSearchDto,
                                                Pageable pageable){
16         return itemRepository.getMainItemPage(itemSearchDto, pageable);
17     }
18
19 }
```

6.4 메인 화면

com.shop.controller.ItemController.java

```
01 package com.shop.controller;
02
03 import com.shop.dto.ItemSearchDto;
04 import com.shop.dto.MainItemDto;
05 import com.shop.service.ItemService;
06 import lombok.RequiredArgsConstructor;
07 import org.springframework.data.domain.Page;
08 import org.springframework.data.domain.PageRequest;
09 import org.springframework.data.domain.Pageable;
10 import org.springframework.stereotype.Controller;
11 import org.springframework.ui.Model;
12 import org.springframework.web.bind.annotation.GetMapping;
13
14 import java.util.Optional;
15
16 @Controller
17 @RequiredArgsConstructor
18 public class MainController {
19
20     private final ItemService itemService;
21
22     @GetMapping(value = "/")
23     public String main(ItemSearchDto itemSearchDto, Optional<Integer> page,
   Model model){
24         Pageable pageable = PageRequest.of(page.isPresent() ? page.get() : 0, 6);
25         Page<MainItemDto> items =
   itemService.getMainItemPage(itemSearchDto, pageable);
26         model.addAttribute("items", items);
27         model.addAttribute("itemSearchDto", itemSearchDto);
28         model.addAttribute("maxPage", 5);
29         return "main";
30     }
31
32 }
```

6.4 메인 화면

- main.html 코드 작성

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

<!-- 사용자 CSS 추가 -->
<th:block layout:fragment="css">
    <style>
        .carousel-inner > .item {
            height: 350px;
        }
        .margin{
            margin-bottom:30px;
        }
        .banner{
            height: 300px;
            position: absolute; top:0; left: 0;
            width: 100%;
            height: 100%;
        }
        .card-text{
            text-overflow: ellipsis;
            white-space: nowrap;
            overflow: hidden;
        }
        a:hover{
            text-decoration:none;
        }
        .center{
            text-align:center;
        }
    </style>
</th:block>
```

6.4 메인 화면

```
35 <div layout:fragment="content">
36
37     <div id="carouselControls" class="carousel slide margin"
                data-ride="carousel">                                    ❶
38         <div class="carousel-inner">
39             <div class="carousel-item active item">
40                 <img class="d-block w-100 banner"
src=" https://user-images.githubusercontent.com/13268420
/112147492-1ab76200-8c20-11eb-8649-3d2f282c3c02.png" alt="First slide">  ❷
41             </div>
42         </div>
43     </div>
44
45     <input type="hidden" name="searchQuery"
           th:value="${itemSearchDto.searchQuery}">                      ❸
46     <div th:if="${not #strings.isEmpty(itemSearchDto.searchQuery)}"
          class="center">
47         <p class="h3 font-weight-bold"
th:text="${itemSearchDto.searchQuery} + '검색 결과'"></p>                ❹
48     </div>
49
50     <div class="row">
```

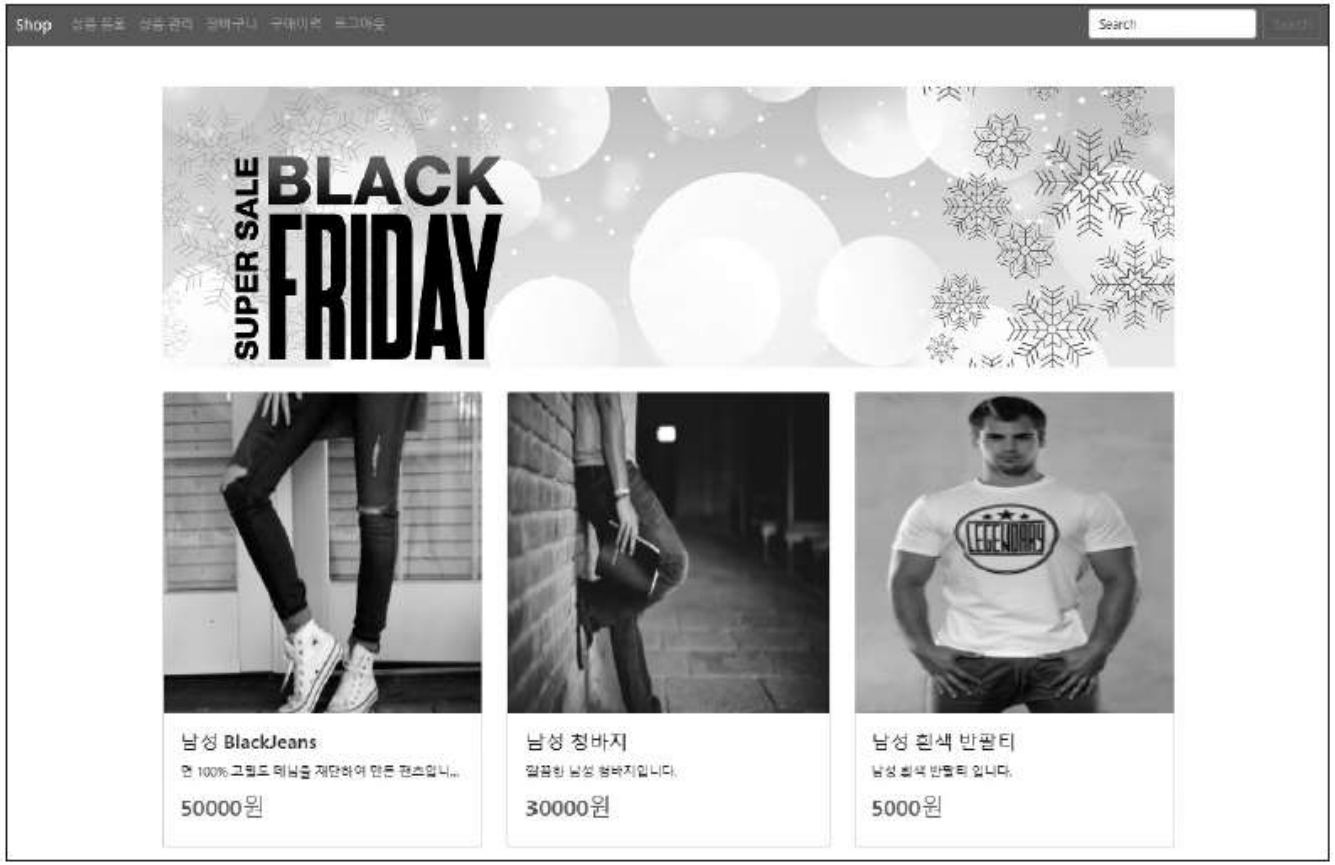
6.4 메인 화면

```
51          <th:block th:each="item, status: ${items.getContent()}">
52              <div class="col-md-4 margin">
53                  <div class="card">
54                      <a th:href="'/item/' +${item.id}" class="text-dark">
55                          <img th:src="${item.imgUrl}" class="card-img-top"
                                th:alt="${item.itemNm}" height="400">
56                          <div class="card-body">
57                              <h4 class="card-title">[[${item.itemNm}]]</h4>
58                              <p class="card-text">[[${item.itemDetail}]]</p>
59                              <h3 class="card-title text-danger">
                                 [[${item.price}]]원</h3>
60                          </div>
61                      </a>
62                  </div>
63              </div>
64          </th:block>
65      </div>
66
67      <div th:with="start=${(items.number/maxPage)*maxPage + 1},
    end=(${(items.totalPages == 0) ? 1 : (start + (maxPage - 1)
    < items.totalPages ? start + (maxPage - 1) : items.totalPages)})" >
68          <ul class="pagination justify-content-center">
69
70              <li class="page-item"
                    th:classappend="${items.number eq 0}?'disabled':''">
71                  <a th:href="@{'/' + '?searchQuery=' +
    ${itemSearchDto.searchQuery} + '&page=' + ${items.number-1}}"
    aria-label='Previous' class="page-link">
72                      <span aria-hidden='true'>Previous</span>
73                  </a>
74              </li>
```

6.4 메인 화면

```
76              <li class="page-item" th:each="page: ${#numbers.sequence(start, end)}"
th:classappend="${items.number eq page-1}?'active':''">
77                  <a th:href="@{'/' +'?searchQuery=' +
${itemSearchDto.searchQuery} + '&page=' + ${page-1}}" th:inline="text"
class="page-link">[[${page}]]</a>
78              </li>
79
80              <li class="page-item"
th:classappend="${items.number+1 ge items.totalPages}?'disabled':''">
81                  <a th:href="@{'/' +'?searchQuery=' +
${itemSearchDto.searchQuery} + '&page=' + ${items.number+1}}"
saria-label='Next' class="page-link">
82                      <span aria-hidden='true'>Next</span>
83                  </a>
84              </li>
85
86          </ul>
87      </div>
88
89  </div>
```

6.4 메인 화면



[그림 6-19] 메인 페이지

6.5 상품 상세 페이지

- 메인 페이지에서 상품 이미지나 상품 정보를 클릭 시 상품의 상세 정보를 보여주는 페이지 구현

com.shop.controller.ItemController.java

```
01 package com.shop.controller;
02
03 .....기존 임포트 생략.....
04
05 @Controller
06 @RequiredArgsConstructor
07 public class ItemController {
08
09     .....코드 생략.....
10
11     @GetMapping(value = "/item/{itemId}")
12     public String itemDtl(Model model, @PathVariable("itemId") Long itemId){
13         ItemFormDto itemFormDto = itemService.getItemDtl(itemId);
14         model.addAttribute("item", itemFormDto);
15         return "item/itemDtl";
16     }
17
18
19 }
```

6.5 상품 상세 페이지

resources/templates/item/itemDtl.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(){

            calculateToalPrice();

            $("#count").change( function(){
                calculateToalPrice();
            });
        });

        function calculateToalPrice(){ ------------------------------ 1
            var count = $("#count").val();
            var price = $("#price").val();
            var totalPrice = price*count;
            $("#totalPrice").html(totalPrice + '원');
        }

    </script>
</th:block>

```

6.5 상품 상세 페이지

```
<!-- 사용자 CSS 추가 -->
<th:block layout:fragment="css">
    <style>
        .mgb-15{
            margin-bottom:15px;
        }
        .mgt-30{
            margin-top:30px;
        }
        .mgt-50{
            margin-top:50px;
        }
        .repImgDiv{
            margin-right:15px;
            height:auto;
            width:50%;
        }
        .repImg{
            width:100%;
            height:400px;
        }
        .wd50{
            height:auto;
            width:50%;
        }
    </style>
</th:block>
```

6.5 상품 상세 페이지

```
61 <div layout:fragment="content" style="margin-left:25%;margin-right:25%">
62
63 <input type="hidden" id="itemId" th:value="${item.id}">
64
65     <div class="d-flex">
66         <div class="repImgDiv">
67             <img th:src="${item.itemImgDtoList[0].imgUrl}"
   class = "rounded repImg" th:alt="${item.itemNm}">
68         </div>
69         <div class="wd50">
70             <span th:if="${item.itemSellStatus ==
   T(com.shop.constant.ItemSellStatus).SELL}" class="badge badge-primary mgb-15">
71                 판매중
72             </span>
```

6.5 상품 상세 페이지

```
73              <span th:unless="${item.itemSellStatus ==
    T(com.shop.constant.ItemSellStatus).SELL}" class="badge btn-danger mgb-15" >
74                  품절
75              </span>
76              <div class="h4" th:text="${item.itemNm}"></div>
77              <hr class="my-4">
78
79              <div class="text-right">
80                  <div class="h4 text-danger text-left">
81                      <input type="hidden" th:value="${item.price}"
                               id="price" name="price">
82                      <span th:text="${item.price}"></span>원
83                  </div>
84                  <div class="input-group w-50">
85                      <div class="input-group-prepend">
86                          <span class="input-group-text">수량</span>
87                      </div>
88                      <input type="number" name="count" id="count"
                               class="form-control" value="1" min="1">
89                  </div>
90              </div>
91              <hr class="my-4">
92
```

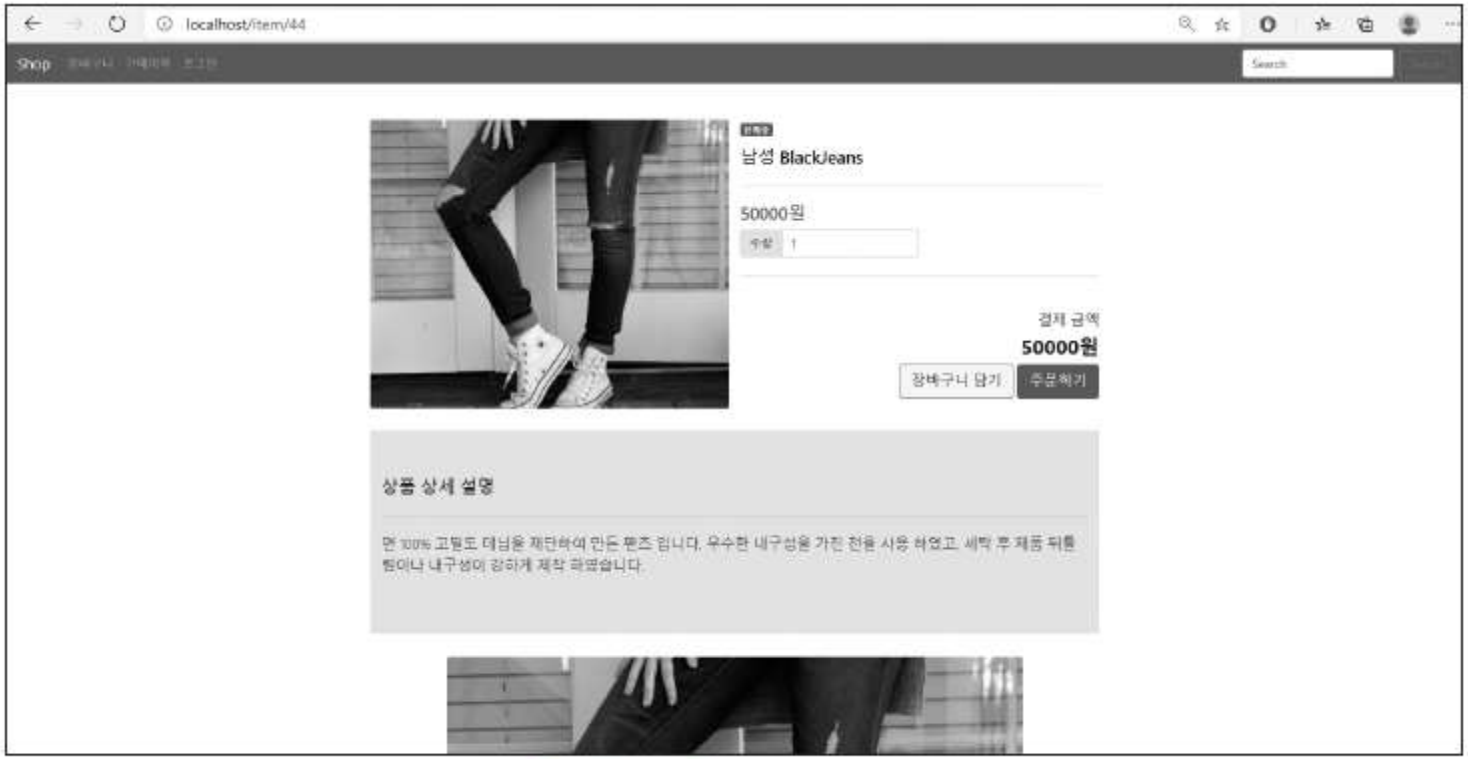
6.5 상품 상세 페이지

```
93              <div class="text-right mgt-50">
94                  <h5>결제 금액</h5>
95                  <h3 name="totalPrice" id="totalPrice"
                        class="font-weight-bold"></h3>
96              </div>
97              <div th:if="${item.itemSellStatus ==
    T(com.shop.constant.ItemSellStatus).SELL}" class="text-right">
98                  <button type="button"
    class="btn btn-light border border-primary btn-lg">장바구니 담기</button>
99                  <button type="button" class="btn btn-primary btn-lg">주문하기
                    </button>
100             </div>
101             <div th:unless="${item.itemSellStatus ==
    T(com.shop.constant.ItemSellStatus).SELL}" class="text-right">
102                 <button type="button" class="btn btn-danger btn-lg">품절
                    </button>
103             </div>
104         </div>
105     </div>
```

6.5 상품 상세 페이지

```
106
107     <div class="jumbotron jumbotron-fluid mgt-30">
108         <div class="container">
109             <h4 class="display-5">상품 상세 설명</h4>
110             <hr class="my-4">
111             <p class="lead" th:text="${item.itemDetail}"></p>
112         </div>
113     </div>
114
115     <div th:each="itemImg : ${item.itemImgDtoList}" class="text-center">
116         <img th:if="${not #strings.isEmpty(itemImg.imgUrl)}"
                 th:src="${itemImg.imgUrl}" class="rounded mgb-15" width="800">
117     </div>
118
119 </div>
120
121 </html>
```

6.5 상품 상세 페이지



[그림 6-20] 상품 상세 페이지

Thank you for your attention