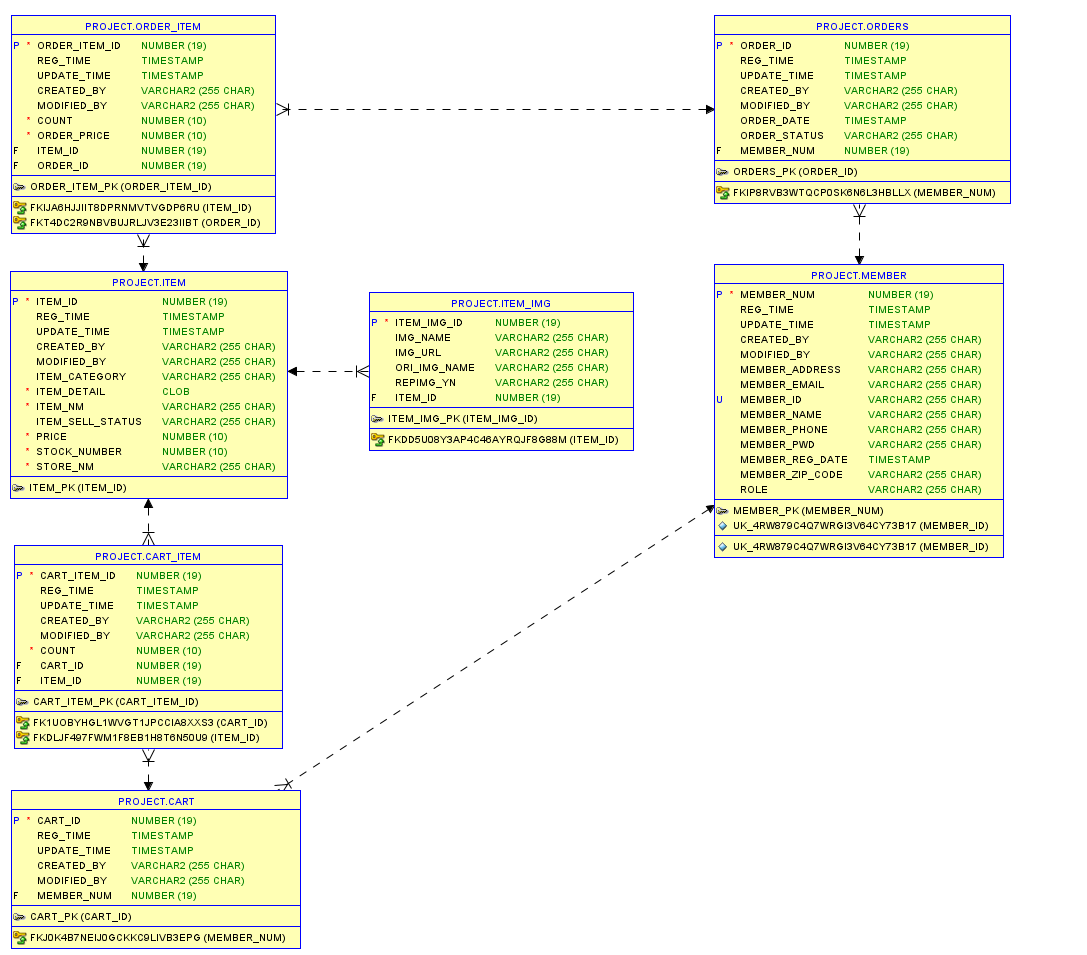
|  |
| --- |
| **쇼핑몰 프로젝트 기술소개서**  (Technical introduction) |

|  |  |
| --- | --- |
| **팀** | 1조 |
| **작성자** | 이병관 (팀장)  전용운  이지연  이승헌 |
| **참고** | 본 소개서는 프로젝트에 사용된 모든 코딩작업에  대한 설명을 담고 있음. |

**- 목 차 -**

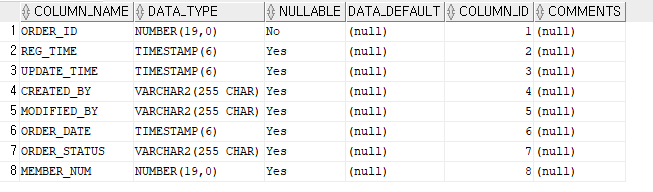
|  |  |
| --- | --- |
|  |  |
| **다이어그램** |  |
| **DB 데이터** |  |
| 1. USER\_INFO |  |
| 2. ORDERS |  |
| 3. ORDER\_ITEM |  |
| 4. MEMBER |  |
| 5. ITEM\_IMG |  |
| 6. ITEM |  |
| 7. CART\_ITEM |  |
| 8. CART |  |
| **MAIN- JAVA** |  |
| 1. 회원가입 |  |
| 2. 로그인 |  |
| 4. 마이페이지-수정 및 삭제 |  |
| 5. 장바구니 |  |
| 6. 상품 주문하기 |  |
| 7. 상품 등록하기 |  |
| **HTML** |  |
| 1. header.html |  |
| 2. footer.html |  |
| 3. main.html |  |
| 4. layout1.html |  |
| **환경 구성** |  |
| 1. build.gradle |  |
| 2. application.properties |  |
| **CSS** |  |
| **오류 및 개선사항** |  |

|  |  |
| --- | --- |
| **ER - 다이어그램**  **(ER - Diagram)** | |
|  |  |



|  |  |
| --- | --- |
| **DB 데이터**  **(DB)** | |
|  |  |

1. ORDERS



ORDER\_ID : 주문 내역 시퀀스 (PK)

ORDER\_DATE : 주문 날짜

ORDER\_STATUS : 주문 상태

MEMBER\_NUM : 멤버 시퀀스 (FK)

\* base entity \*

REG\_TIME : 생성시간

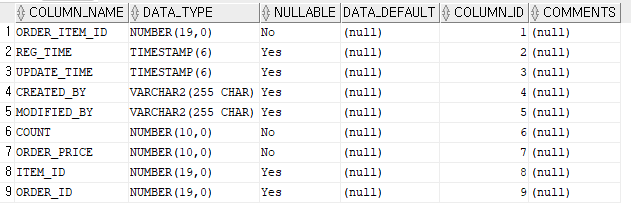
UPDATE\_TIME : 수정시간

CREATED\_BY : 생성자

MODIFIED\_BY : 수정자

**DB 데이터**

2. ORDER\_ITEM



ORDER\_ITEM\_ID : 구매 내역 시퀀스(PK)

REG\_TIME : -

UPDATE\_TIME : -

CREATED\_BY : -

MODIFIED\_BY : -

COUNT : 수량

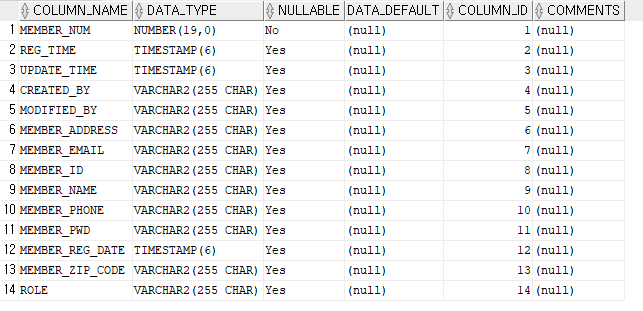
ORDER\_PRICE : 판매 가격

ITEM\_ID : 아이템 아이디 (FK)

ORDER\_ID : 주문 아이디 (FK)

**DB 데이터**

3. MEMBER



MEMBER\_NUM : 회원 시퀀스 (PK)

REG\_TIME : -

UPDATE\_TIME : -

CREATED\_BY : -

MODIFIED\_BY : -

MEMBER\_ADDRESS : 상세 주소

MEMBER\_EMAIL : 이메일

MEMBER\_ID : 아이디

MEMBER\_NAME : 이름

MEMBER\_PHONE : 핸드폰 번호

MEMBER\_PWD : 패스워드

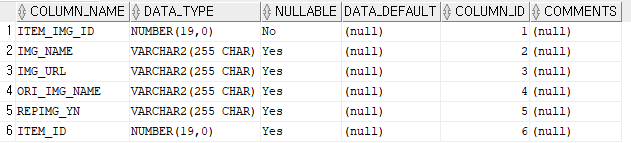
MEMBER\_REG\_DATE : -

MEMBER\_ZIP\_CODE : 주소

ROLE : 권한

**DB 데이터**

4. ITEM\_IMG



ITEM\_IMG\_ID : 이미지 시퀀스 (PK)

IMG\_NAME : 이미지 파일명

IMG\_URL : 이미지 조회 경로

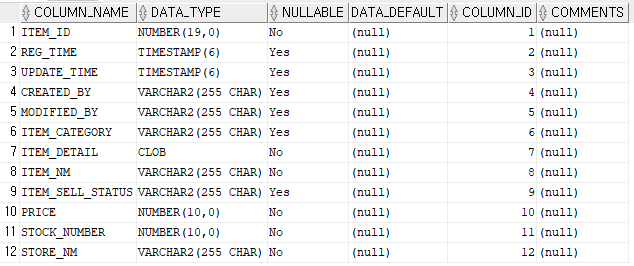
ORI\_IMG\_NAME : 원본 이미지 파일명

REPIMG\_YN : 대표 이미지 여부

ITEM\_ID : 아이템 아이디 (FK)

**DB 데이터**

5. ITEM



ITEM\_ID : 상품 시퀀스 (PK)

REG\_TIME : -

UPDATE\_TIME : -

CREATED\_BY : -

MODIFIED\_BY : -

ITEM\_CATEGORY : 상품 카테고리

ITEM\_DETAIL : 상품 상세정보

ITEM\_NM : 상품명

ITEM\_SELL\_STATUS : 상품 상태

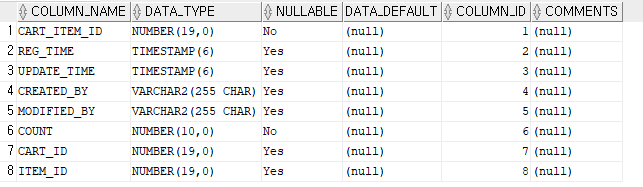
PRICE : 상품 가격

STOCK\_NUMBER : 재고 수량

STORE\_NM : 상호명

**DB 데이터**

6. CART\_ITEM



CART\_ITEM\_ID : 장바구니 시퀀스 (PK)

REG\_TIME : -

UPDATE\_TIME : -

CREATED\_BY : -

MODIFIED\_BY : -

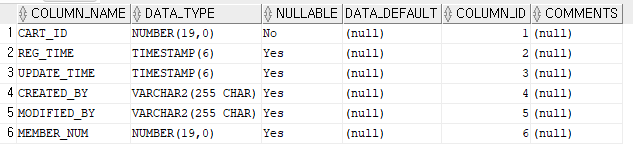
COUNT : 수량

CART\_ID :

ITEM\_ID : 아이템 아이디 (FK)

**DB 데이터**

7. CART



CART\_ID : 주문자 시퀀스 (PK)

REG\_TIME : -

UPDATE\_TIME : -

CREATED\_BY : -

MODIFIED\_BY : -

MEMBER\_NUM : 회원 시퀀스 (FK)

|  |  |
| --- | --- |
| **Main Java**  **(Main Java)** | |
|  |  |

**1-1 회원 가입**

**(Member.java)**

@Entity

@Table(name = "member")

@Getter

@Setter

@ToString

public class Member extends BaseEntity implements UserDetails {

@Id

@Column(name = "memberNum")

@SequenceGenerator(name = "MEMBER\_SEQUENCE\_GEN", sequenceName = "member\_seq", initialValue = 1, allocationSize = 1) // 시퀀스 제너레이터 이름, 시퀀스 이름, 시작 값, 할당 범위 사이즈

@GeneratedValue(strategy = GenerationType.*SEQUENCE*, generator = "MEMBER\_SEQUENCE\_GEN") // 사용할 전략을 시퀀스로 선택, 식별자 생성기를 MEMBER\_SEQUENCE\_GEN으로 설정

private Long memberNum;

@Column(unique = true)

private String memberId;

private String memberPwd;

private String memberName;

private String memberEmail;

private String memberZipCode;

private String memberAddress;

private String memberPhone;

@CreationTimestamp

private LocalDateTime memberRegDate;

private String role;

public static Member createMember(MemberRegisterFormDto memberRegisterFormDto, PasswordEncoder passwordEncoder) {

// 멤버 생성 메서드

Member member = new Member();

member.setMemberId(memberRegisterFormDto.getMemberId());

String memberPwd = passwordEncoder.encode(memberRegisterFormDto.getMemberPwd());

member.setMemberPwd(memberPwd);

member.setMemberName(memberRegisterFormDto.getMemberName());

member.setMemberEmail(memberRegisterFormDto.getMemberEmail());

member.setMemberZipCode(memberRegisterFormDto.getMemberZipCode());

member.setMemberAddress(memberRegisterFormDto.getMemberAddress());

member.setMemberPhone(memberRegisterFormDto.getMemberPhone());

member.setRole(memberRegisterFormDto.getRole());

return member;

}

@Override

public Collection<? extends GrantedAuthority> getAuthorities() {

// 복수의 권한을 콜렉션으로 받아 "," 단위로 split 하여 리턴

Set<GrantedAuthority> roles = new HashSet<>();

for (String role : role.split(",")) {

roles.add(new SimpleGrantedAuthority(role));

}

return roles;

}

@Override

public String getPassword() {

//사용자의 password 반환

return null;

}

@Override

public String getUsername() {

//사용자의 id 반환(unique 값)

return null;

}

@Override

public boolean isAccountNonExpired() {

// 계정 만료 여부 반환

return false;

}

@Override

public boolean isAccountNonLocked() {

// 계정 잠금 여부 반환

return false;

}

@Override

public boolean isCredentialsNonExpired() {

// 패스워드 만료 여부 반환

return false;

}

@Override

public boolean isEnabled() {

// 계정 사용 가능 여부 반환

return false;

}

}

**1-2 회원 가입**

**(MemberFormDto.java)**

@Getter

@Setter

public class MemberRegisterFormDto {

@NotBlank(message = "아이디는 필수 입력 값입니다")

private String memberId;

@NotEmpty(message = "비밀번호는 필수 입력 값입니다")

private String memberPwd;

@NotEmpty(message = "이름은 필수 입력 값입니다")

private String memberName;

@NotEmpty(message = "이메일은 필수 입력 값입니다")

@Email(message = "이메일 형식으로 입력하세요")

private String memberEmail;

@NotEmpty(message = "우편번호는 필수 입력 값입니다")

private String memberZipCode;

@NotEmpty(message = "상세주소는 필수 입력 값입니다")

private String memberAddress;

@NotEmpty(message = "전화번호는 필수 입력 값입니다")

private String memberPhone;

private String role;

}

**1-3 회원 가입**

**(MemberRepository.java)**

public interface MemberRepository extends JpaRepository<Member, Long> {

Member findByMemberId(String memberId);

}

**1-4 회원 가입 기능**

**(MemberService.java)**

@Service

@Transactional // 모든 작업이 성공해야만 데이터 베이스에 반영하는 어노테이션

@RequiredArgsConstructor // 빈 생성자 선언 어노테이션

public class MemberService implements UserDetailsService {

private final MemberRepository memberRepository;

public Member saveMember(Member member) {

validateDuplicateMember(member);

return memberRepository.save(member); //

}

private void validateDuplicateMember(Member member) {

// 아이디 중복 검사

Member findMember = memberRepository.findByMemberId(member.getMemberId());

if (findMember != null) {

throw new IllegalStateException("이미 가입한 회원입니다");

}

}

@Override

public UserDetails loadUserByUsername(String memberId) throws UsernameNotFoundException {

// UserDetailService 인터페이스 구현

Member member = memberRepository.findByMemberId(memberId);

if (member == null) {

throw new UsernameNotFoundException(memberId);

}

return User.*builder*()

.username(member.getMemberId())

.password(member.getMemberPwd())

.authorities(member.getAuthorities())

.build();

}

}

**1-5 회원 가입**

**(MemberController.java)**

@RequestMapping(value = "/members")

@Controller

@RequiredArgsConstructor

public class MemberController {

private final MemberService memberService;

private final PasswordEncoder passwordEncoder;

@GetMapping(value = "/new")

public String memberRegisterForm(Model model) {

model.addAttribute("memberRegisterForm", new MemberRegisterFormDto());

return "member/memberRegisterForm";

}

@PostMapping(value = "/new")

public String newMember(@Valid MemberRegisterFormDto memberRegisterFormDto, BindingResult bindingResult, Model model) {

// 검증하려는 객체에 @Vaild선언, 파라미터로 bindingResult객체 추가

if (bindingResult.hasErrors()) {

// bindingResult.hasError()를 호출 에러가 있다면 회원가입 페이지로 이동

return "member/memberRegisterForm";

}

try {

Member member = Member.*createMember*(memberRegisterFormDto, passwordEncoder);

memberService.saveMember(member);

} catch (IllegalStateException e) {

// 회원 가입 시 중복 회원 예외가 발생하면 에 메시지를 뷰로 전달

model.addAttribute("errorMessage", e.getMessage());

return "member/memberRegisterForm";

}

return "redirect:/";

}

@GetMapping(value = "/login")

public String loginMember() {

return "/member/memberLoginForm";

}

@GetMapping(value = "/login/error")

public String loginError(Model model) {

model.addAttribute("loginErrorMsg", "아이디 또는 비밀번호를 확인하세요");

return "/member/memberLoginForm";

}

}

**1-6 회원 가입**

**(memberRegisterForm.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 charset="UTF-8">

<title>sign up</title>

<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-BmbxuPwQa2lc/FVzBcNJ7UAyJxM6wuqIj61tLrc4wSX0szH/Ev+nYRRuWlolflfl" crossorigin="anonymous">

</head>

<body>

<h1>Sign Up</h1> <hr>

<div layout:fragment="content">

<form action="/members/new" role="form" method="post" th:object="${memberRegisterFormDto}">

<input type="hidden" th:name="${\_csrf.parameterName}" th:value="${\_csrf.token}">

<div class="mb-3">

<label for="exampleInputEmail1" class="form-label">Id</label>

<input type="text" class="form-control" id="exampleInputEmail1" aria-describedby="emailHelp" name="memberId">

</div>

<div class="mb-3">

<label for="exampleInputPassword1" class="form-label">Password</label>

<input type="password" class="form-control" id="exampleInputPassword1" name="memberPwd">

</div>

<div class="mb-3">

<label for="exampleInputPassword1" class="form-label">Name</label>

<input type="text" class="form-control" id="exampleInputPassword2" name="memberName">

</div>

<div class="mb-3">

<label for="exampleInputEmail1" class="form-label">Email</label>

<input type="email" class="form-control" id="exampleInputEmail2" aria-describedby="emailHelp" name="memberEmail">

</div>

<div class="mb-3">

<label for="exampleInputEmail1" class="form-label">도로명주소</label>

<input type="text" class="form-control" id="address\_kakao" aria-describedby="emailHelp" name="memberZipCode" readonly>

</div>

<div class="mb-3">

<label for="exampleInputEmail1" class="form-label">상세주소</label>

<input type="text" class="form-control" id="exampleInputEmail4" aria-describedby="emailHelp" name="memberAddress">

</div>

<div class="mb-3">

<label for="exampleInputEmail1" class="form-label">Phone</label>

<input type="text" class="form-control" id="exampleInputEmail5" aria-describedby="emailHelp" name="memberPhone">

</div>

<div class="mb-3 form-check">

<input type="radio" class="form-check-input" id="exampleCheck1" name="role" value="ROLE\_STORE,ROLE\_MEMBER"> store <br>

<input type="radio" class="form-check-input" id="exampleCheck2" name="role" value="ROLE\_MEMBER" checked="checked"> user <br>

</div>

<div class="cheak-box">

<p><span class="cheak-box1"><input type="checkbox" id="box1"><em class="checkbox"></em></span><label for=><strong>이용약관 및 개인정보수집 및 이용, 쇼핑정보 수신(선택)에 모두 동의합니다.</strong></label></p>

</div>

<button type="submit" class="btn btn-primary">Join</button>

</form> <br>

<script src="//t1.daumcdn.net/mapjsapi/bundle/postcode/prod/postcode.v2.js"></script>

<script>

window.onload = function(){

document.getElementById("address\_kakao").addEventListener("click", function(){

//주소입력칸을 클릭하면

//카카오 지도 발생

new daum.Postcode({

oncomplete: function(data) { //선택시 입력값 세팅

document.getElementById("address\_kakao").value = data.address; // 주소 넣기

document.querySelector("input[name=memberAddress]").focus(); //상세주소 포커싱

}

}).open();

});

}

</script>

</div>

</body>

<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/js/bootstrap.bundle.min.js" integrity="sha384-b5kHyXgcpbZJO/tY9Ul7kGkf1S0CWuKcCD38l8YkeH8z8QjE0GmW1gYU5S9FOnJ0" crossorigin="anonymous"></script>

</html>

**1-7 회원 가입**

[**( HYPERLINK "(MainController.java"MainController.java**]((MainController.java)**)**

@Controller

@RequiredArgsConstructor

public class MainController {

@GetMapping(value = "/")

public String main() {

return "main";

}

}

**//main.html = layout1.html 대체.**

**2-1 로그인/로그아웃**

**(MemberService.java)**

@Override

public UserDetails loadUserByUsername(String memberId) throws UsernameNotFoundException {

Member member = memberRepository.findByMemberId(memberId);

// findByMemberId(memberId)를 호출하여 member 객체에 id 저장

if (member == null) {

// member 가 null 일 경우 UsernameNotFoundException발생

throw new UsernameNotFoundException(memberId);

}

return User.*builder*()

//UserDetail를 구현하고 있는 User객체 반환

.username(member.getMemberId())

.password(member.getMemberPwd())

.authorities(member.getAuthorities())

.build();

}

**2-2 로그인/로그아웃**

**(SecurityConfig.java)**

@Configuration

@EnableWebSecurity

//시큐리티 사용을 위한 어노테이션

public class SecurityConfig extends WebSecurityConfigurerAdapter {

//WebSecurityConfigureAdapter 상속

@Autowired

MemberService memberService;

@Override

protected void configure(HttpSecurity http) throws Exception {

http.formLogin()

.loginPage("/members/login") //로그인 페이지

.defaultSuccessUrl("/") // 로그인 성공시 URL

.usernameParameter("memberId")

.passwordParameter("memberPwd")

.failureUrl("/members/login/error") // 로그인 실패시 URL

.and()

.logout()

.logoutRequestMatcher(new AntPathRequestMatcher("/members/logout"))

.logoutSuccessUrl("/");

http.authorizeRequests()

.mvcMatchers("/", "/members/\*\*", "/item/\*\*", "/images/\*\*").permitAll()

.anyRequest().authenticated(); // 인증이 되어야 함

}

@Override

public void configure(WebSecurity web) throws Exception {

//스프링 시큐리티 규칙을 무시하는 URL 설정

web.ignoring()

.antMatchers("/favicon.ico", "/resources/\*\*", "/error","/css/\*\*", "/js/\*\*", "/img/\*\*");

}

@Bean

public PasswordEncoder passwordEncoder() {

return new BCryptPasswordEncoder();

}

@Override

protected void configure(AuthenticationManagerBuilder auth) throws Exception {

//AuthenticationManagerBuilder.userDetailsService().passwordEncoder() 를 통해 패스워드 암호화에 사용될 PasswordEncoder 구현체 지정

auth.userDetailsService(memberService)

.passwordEncoder(passwordEncoder());

}

}

**2-3 로그인/로그아웃**

**(memberLoginForm.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 charset="UTF-8">

<title>StoreLoginForm</title>

</head>

<th:block layout:fragment="css">

<style>

.error {

color: #bd2130;

}

</style>

</th:block>

<body>

<div layout:fragment="content">

<form role="form" method="post" action="/members/login">

<input type="hidden" th:name="${\_csrf.parameterName}" th:value="${\_csrf.token}">

<div class="form-group">

<label th:for="memberId">아이디</label>

<input type="text" name="memberId" class="form-control" placeholder="아이디를 입력하세요">

</div>

<div class="form-group">

<label th:for="memberPwd">비밀번호</label>

<input type="password" name="memberPwd" id="memberPwd" class="form-control" placeholder="비밀번호를 입력하세요">

</div>

<p th:if="${loginErrorMsg}" class="error" th:text="${loginErrorMsg}"></p>

<button class="btn btn-primary">로그인</button>

<button type="button" class="btn btn-primary" onclick="location.href='/members/new'">회원 가입</button>

</form>

</div>

</body>

</html>

**2-4 로그인/로그아웃**

**(MemberController.java)**

@RequestMapping(value = "/members")

@Controller

@RequiredArgsConstructor

public class MemberController {

private final MemberService memberService;

private final PasswordEncoder passwordEncoder;

@GetMapping(value = "/new")

public String memberRegisterForm(Model model) {

model.addAttribute("memberRegisterForm", new MemberRegisterFormDto());

return "member/memberRegisterForm";

}

@PostMapping(value = "/new")

public String newMember(@Valid MemberRegisterFormDto memberRegisterFormDto, BindingResult bindingResult, Model model) {

if (bindingResult.hasErrors()) {

return "member/memberRegisterForm";

}

try {

Member member = Member.*createMember*(memberRegisterFormDto, passwordEncoder);

memberService.saveMember(member);

} catch (IllegalStateException e) {

model.addAttribute("errorMessage", e.getMessage());

return "member/memberRegisterForm";

}

return "redirect:/";

}

@GetMapping(value = "/login")

public String loginMember() {

return "/member/memberLoginForm";

}

@GetMapping(value = "/login/error")

public String loginError(Model model) {

model.addAttribute("loginErrorMsg", "아이디 또는 비밀번호를 확인하세요");

return "/member/memberLoginForm";

}

}

**3. 마이페이지 - 수정 및 삭제**

**5-1. 장바구니**

**(CartController.java)**

@Controller  
@RequiredArgsConstructor  
public class CartController {  
 private final CartService cartService;

@PostMapping(value = "/cart")  
 public @ResponseBody  
 ResponseEntity order(@RequestBody @Valid CartItemDto cartItemDto,  
 BindingResult bindingResult, Principal principal){  
 //장바구니에 담을 상품 정보를 cartItemDto 객체에 데이터 바인딩 시 에러 검사  
 if(bindingResult.hasErrors()){  
 StringBuilder sb = new StringBuilder();  
 List<FieldError> fieldErrors = bindingResult.getFieldErrors();  
  
 for (FieldError fieldError : fieldErrors) {  
 sb.append(fieldError.getDefaultMessage());  
 }  
  
 return new ResponseEntity<String>(sb.toString(), HttpStatus.*BAD\_REQUEST*);  
 }  
 // 현제 로그인한 회원의 이메일 정보를 변수에 저장  
 String memberId = principal.getName();  
 Long cartItemId;  
  
 try {

// 장바구니에 담을 상품 정보와 현제 로그인한 이메일 정보 이용 장바구니에 상품 담기  
 cartItemId = cartService.addCart(cartItemDto, memberId);  
 } catch(Exception e){  
 return new ResponseEntity<String>(e.getMessage(), HttpStatus.*BAD\_REQUEST*);  
 }  
 // 결과값으로 생성된 장바구니 상품 아이디와 요청이 성공하였다는 HTTP 응답 상태 코드 반환  
 return new ResponseEntity<Long>(cartItemId, HttpStatus.*OK*);  
 }

// 현재 로그인한 이메일 정보 이용 장바구니 정보를 조회 후 view로 전달  
 @GetMapping(value = "/cart")  
 public String orderHist(Principal principal, Model model){  
 List<CartDetailDto> cartDetailList = cartService.getCartList(principal.getName());  
 model.addAttribute("cartItems", cartDetailList);  
 return "cart/cartList";  
 }

// PATCH는 요청된 자원의 일부를 업데이트 할때 사용 장바구니 상품 수량만 업데이트하여 사용  
 @PatchMapping(value = "/cartItem/{cartItemId}")  
 public @ResponseBody ResponseEntity updateCartItem(@PathVariable("cartItemId") Long cartItemId, int count, Principal principal){  
// 장바구니 상품의 개수가 0개 이하로 업데이트시 에러 메시지를 반환  
 if(count <= 0){  
 return new ResponseEntity<String>("최소 1개 이상 담아주세요", HttpStatus.*BAD\_REQUEST*);

// 수정 권한을 체크한다  
 } else if(!cartService.validateCartItem(cartItemId, principal.getName())){  
 return new ResponseEntity<String>("수정 권한이 없습니다.", HttpStatus.*FORBIDDEN*);  
 }  
// 장바구니 상품 개수를 업데이트 한다.  
 cartService.updateCartItemCount(cartItemId, count);  
 return new ResponseEntity<Long>(cartItemId, HttpStatus.*OK*);  
 }

// deletemapping 자원을 삭제 할때 사용, 장바구니 상품 삭제하기 때문에 사용  
 @DeleteMapping(value = "/cartItem/{cartItemId}")  
 public @ResponseBody ResponseEntity deleteCartItem(@PathVariable("cartItemId") Long cartItemId, Principal principal){  
// 수정 권한을 체크한다.  
 if(!cartService.validateCartItem(cartItemId, principal.getName())){  
 return new ResponseEntity<String>("수정 권한이 없습니다.", HttpStatus.*FORBIDDEN*);  
 }  
// 해당 장바구니 상품을 삭제한다.  
 cartService.deleteCartItem(cartItemId);  
 return new ResponseEntity<Long>(cartItemId, HttpStatus.*OK*);  
 }  
// 상품 주문하기  
 @PostMapping(value = "/cart/orders")  
 public @ResponseBody ResponseEntity orderCartItem(@RequestBody CartOrderDto cartOrderDto, Principal principal){  
  
 List<CartOrderDto> cartOrderDtoList = cartOrderDto.getCartOrderDtoList();  
// 주문할 상품을 선택했는지 확인한다.  
 if(cartOrderDtoList == null || cartOrderDtoList.size() == 0){  
 return new ResponseEntity<String>("주문할 상품을 선택해주세요", HttpStatus.*FORBIDDEN*);  
 }  
// 주문 권한을 체크한다.  
 for (CartOrderDto cartOrder : cartOrderDtoList) {  
 if(!cartService.validateCartItem(cartOrder.getCartItemId(), principal.getName())){  
 return new ResponseEntity<String>("주문 권한이 없습니다.", HttpStatus.*FORBIDDEN*);  
 }  
 }  
// 주문 로직 호출 결과 생성된 주문 번호를 반환 받는다.  
 Long orderId = cartService.orderCartItem(cartOrderDtoList, principal.getName());

// 생성된 주문번호와 요청이 성공했다는 HTTP 응답 상태 코드를 반환  
 return new ResponseEntity<Long>(orderId, HttpStatus.*OK*);  
 }  
}

**5-2. 장바구니**

**(CartDetailDto.java)**

@Getter  
@Setter  
public class CartDetailDto {  
 private Long cartItemId;  
 private String storeNm;  
 private String itemNm;  
 private int price;  
 private int count;  
 private String imgUrl;  
  
 public CartDetailDto(Long cartItemId, String storeNm, String itemNm, int price, int count, String imgUrl) {  
 this.cartItemId = cartItemId;  
 this.storeNm = storeNm;  
 this.itemNm = itemNm;  
 this.price = price;  
 this.count = count;  
 this.imgUrl = imgUrl;  
 }  
}

**5-3. 장바구니**

**(CartItemDto.java)**

@Getter @Setter  
public class CartItemDto {  
  
 @NotNull(message = "상품 아이디는 필수 입력 값 입니다.")  
 private Long itemId;  
  
 @Min(value = 1, message = "최소 1개 이상 담아주세요")  
 private int count;  
  
}

**5-4. 장바구니**

**(CartOrderDto.java)**

@Getter  
@Setter  
public class CartOrderDto {  
 private Long cartItemId;  
 private List<CartOrderDto> cartOrderDtoList;  
}

**5-5. 장바구니**

**(Cart.java)**

@Entity  
@Table(name = "cart")  
@Getter  
@Setter  
@ToString  
public class Cart extends BaseEntity {  
 @Id  
 @Column(name = "cartId")  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private Long id;

//부모Entity를 사용할 필요가 없으면 속도를 위해 LAZY를 설정 LAZY로 지정되면 지정된 Entity객체를

//미리 가지고 있는 것이 아니라 Entity객체를 사용하려고 할 때 즉시 데이터를 db에서 가져온다  
 @OneToOne(fetch = FetchType.*LAZY*)  
 @JoinColumn(name = "memberNum")  
 private Member member;  
  
 public static Cart createCart(Member member){  
 Cart cart = new Cart();  
 cart.setMember(member);  
 return cart;  
 }  
}

**5-6. 장바구니**

**(CartItem.java)**

@Entity  
@Getter  
@Setter  
@Table(name = "cartItem")  
public class CartItem extends BaseEntity {  
 @Id  
 @GeneratedValue  
 @Column(name = "cartItemId")  
 private Long id;  
// N:1관계일때 사용한다  
 @ManyToOne(fetch = FetchType.*LAZY*)  
 @JoinColumn(name = "cartId")  
 private Cart cart;  
  
 @ManyToOne(fetch = FetchType.*LAZY*)  
 @JoinColumn(name = "itemId")  
 private Item item;  
  
 private int count;  
  
 public static CartItem createCartItem(Cart cart, Item item, int count){  
 CartItem cartItem = new CartItem();  
 cartItem.setCart(cart);  
 cartItem.setItem(item);  
 cartItem.setCount(count);  
 return cartItem;  
 }  
  
 public void addCount(int count){  
 this.count += count;  
 }  
  
 public void updateCount(int count) {  
 this.count = count;  
 }  
  
}

**5-7. 장바구니**

**(CartItemRepository.interface)**

public interface CartItemRepository extends JpaRepository<CartItem, Long> {  
 CartItem findByCartIdAndItemId(Long cartId, Long itemId);  
// 카트아이디와 상품 아이디를 이용해서 상품이 장바구니에 들어있는지 확인  
 @Query("select new com.final\_project.TodayDessert.dto.CartDetailDto(ci.id, i.storeNm, i.itemNm, i.price, ci.count, im.imgUrl) " +  
 "from CartItem ci, ItemImg im " +  
 "join ci.item i " +  
 "where ci.cart.id = :cartId " +  
 "and im.item.id = ci.item.id " +  
 "and im.repimgYn = 'Y' " +  
 "order by ci.regTime desc"  
 )  
 List<CartDetailDto> findCartDetailDtoList(Long cartId);  
}

**5-8. 장바구니**

**(CartRepository.interface)**

public interface CartRepository extends JpaRepository<Cart, Long> {  
 Cart findByMember\_MemberNum(Long memberNum);  
}

**5-9. 장바구니**

**(CartService.java)**

@Service

// begin,commit을 자동으로 해주고 예외 발생시 rollback처리를 자동으로 해준다.  
@Transactional  
@RequiredArgsConstructor  
public class CartService {  
 private final ItemRepository itemRepository;  
 private final MemberRepository memberRepository;  
 private final CartRepository cartRepository;  
 private final CartItemRepository cartItemRepository;  
 private final OrderService orderService;  
  
  
 public Long addCart(CartItemDto cartItemDto, String memberId){  
// 장바구니에 담을 상품 엔티티를 조회  
 Item item = itemRepository.findById(cartItemDto.getItemId())  
 .orElseThrow(EntityNotFoundException::new);

// 현재 로그인한 회원 엔티티를 조회  
 Member member = memberRepository.findByMemberId(memberId);  
// 현제 로그인한 회원의 장바구니 엔티티 조회  
 Cart cart = cartRepository.findByMember\_MemberNum(member.getMemberNum());

// 처음으로 장바구니에 담을 경우 해당 회원의 장바구니 엔티티를 생성  
 if(cart == null){  
 cart = Cart.*createCart*(member);  
 cartRepository.save(cart);  
 }  
// 현제 상품이 장바구니에 이미 들어가 있는지 조회  
 CartItem savedCartItem = cartItemRepository.findByCartIdAndItemId(cart.getId(), item.getId());  
  
 if(savedCartItem != null){

// 이미 장바구니에 들어있으면 기존 수량에 담을 수량을 더함  
 savedCartItem.addCount(cartItemDto.getCount());  
 return savedCartItem.getId();  
 } else {

// 장바구니 엔티티, 상품 엔티티, 장바구니에 담을 수량을 이용하여 cartitem 엔티티 생성  
 CartItem cartItem = CartItem.*createCartItem*(cart, item, cartItemDto.getCount());

// 장바구니에 들어갈 상품 저장  
 cartItemRepository.save(cartItem);  
 return cartItem.getId();  
 }  
 }  
  
 @Transactional(readOnly = true)  
 public List<CartDetailDto> getCartList(String memberId){  
  
 List<CartDetailDto> cartDetailDtoList = new ArrayList<>();  
  
 Member member = memberRepository.findByMemberId(memberId);

// 현제 로그인한 회원의 장바구니 엔티티 조회  
 Cart cart = cartRepository.findByMember\_MemberNum(member.getMemberNum());  
 if(cart == null){

// 장바구니 상품을 담은적이 없을 경우 빈 리스트 반환  
 return cartDetailDtoList;  
 }  
// 장바구니에 담겨있는 상품 정보를 조회  
 cartDetailDtoList = cartItemRepository.findCartDetailDtoList(cart.getId());  
 return cartDetailDtoList;  
 }

@Transactional(readOnly = true)  
 public boolean validateCartItem(Long cartItemId, String memberId){

// 현제 로그인한 회원 조회  
 Member curMember = memberRepository.findByMemberId(memberId);  
 CartItem cartItem = cartItemRepository.findById(cartItemId)  
 .orElseThrow(EntityNotFoundException::new);

// 장바구니 상품을 저장한 회원을 조회  
 Member savedMember = cartItem.getCart().getMember();  
// 현제 로그인한 회원과 장바구니 상품을 저장한 회원과 비교 다르면false 같으면 true  
 if(!StringUtils.*equals*(curMember.getMemberId(), savedMember.getMemberId())){  
 return false;  
 }  
  
 return true;  
 }  
// 장바구니 상품의 수량을 업데이트   
 public void updateCartItemCount(Long cartItemId, int count){  
 CartItem cartItem = cartItemRepository.findById(cartItemId)  
 .orElseThrow(EntityNotFoundException::new);  
  
 cartItem.updateCount(count);  
 }  
//cartcontroller 클래스에 장바구니 상품을 삭제 요청 처리 가능한 로직 추가  
 public void deleteCartItem(Long cartItemId) {  
 CartItem cartItem = cartItemRepository.findById(cartItemId)  
 .orElseThrow(EntityNotFoundException::new);  
 cartItemRepository.delete(cartItem);  
 }  
  
 public Long orderCartItem(List<CartOrderDto> cartOrderDtoList, String memberId){  
 List<OrderDto> orderDtoList = new ArrayList<>();  
//장바구니 페이지에서 전달받은 주문 상품 번호를 이용하여 주문로직으로 전달할 orderDto객체 만듬  
 for (CartOrderDto cartOrderDto : cartOrderDtoList) {  
 CartItem cartItem = cartItemRepository  
 .findById(cartOrderDto.getCartItemId())  
 .orElseThrow(EntityNotFoundException::new);  
  
 OrderDto orderDto = new OrderDto();  
 orderDto.setItemId(cartItem.getItem().getId());  
 orderDto.setCount(cartItem.getCount());  
 orderDtoList.add(orderDto);  
 }  
// 장바구니에 담은 상품을 주문하도록 주문 로직을 호출  
 Long orderId = orderService.orders(orderDtoList, memberId);

// 주문한 상품들을 장바구니에서 삭제  
 for (CartOrderDto cartOrderDto : cartOrderDtoList) {  
 CartItem cartItem = cartItemRepository  
 .findById(cartOrderDto.getCartItemId())  
 .orElseThrow(EntityNotFoundException::new);  
 cartItemRepository.delete(cartItem);  
 }  
  
 return orderId;  
 }  
  
}

**6-1. 상품등록**

**(ItemCategory.enum)**

public enum ItemCategory {  
 *MACARON*, *SCONE*, *CANNELE*, *BUTTERBAR*, *FINANCIER*, *COOKIES*, *VEGAN*, *ETC*}

**6-2. 상품등록**

**(ItemSellStatus.enum)**

public enum ItemSellStatus {  
 *SELL*, *SOLD\_OUT*}

**6-3. 상품등록**

**(ItemController.java)**

@Controller  
@RequiredArgsConstructor  
public class ItemController {  
  
 private final ItemService itemService;  
// ItemFormDto를 model 객체에 담아 뷰로 전달  
 @GetMapping(value = "/store/item/new")  
 public String itemForm(Model model) {  
 model.addAttribute("itemFormDto", new ItemFormDto());  
 return "item/itemRegisterForm";  
 }  
  
 @PostMapping(value = "/store/item/new")  
 public String itemNew(@Valid ItemFormDto itemFormDto, BindingResult bindingResult, Model model, @RequestParam("itemImgFile")List<MultipartFile> itemImgFileList) {

// 상품 등록시 필수 값이 없다면 다시 등록 페이지로 전환  
 if (bindingResult.hasErrors()) {  
 return "item/itemRegisterForm";  
 }  
// 첫 번째 이미지가 없다면 에러 메시지와 함께 등록 페이지로 전환 첫번째 이미지는 페이지에서 보여줄 이미지로 사용하여 필수 값으로 지정  
 if (itemImgFileList.get(0).isEmpty() && itemFormDto.getId() == null) {  
 model.addAttribute("errorMessage", "첫번째 상품 이미지는 필수 입력 값입니다");  
 return "item/itemRegisterForm";  
 }  
  
 try {

// 상품 저장 로직 호출 매개 변수로 상품 정보와 이미지 정보를 담고 itemimgfilelist를 넘겨줌  
 itemService.saveItem(itemFormDto, itemImgFileList);  
 } catch (Exception e) {  
 model.addAttribute("errorMessage", "상품 등록 중 에러가 발생했습니다");  
 return "item/itemRegisterForm";  
 }

// 정상 등록 메인 페이지로 이동  
 return "redirect:/";  
 }  
  
 @GetMapping(value = "/store/item/{itemId}")  
 public String itemDtl(@PathVariable("itemId") Long itemId, Model model) {  
 try {

// 조회한 상품 데이터를 모델에 담아 뷰로 전달  
 ItemFormDto itemFormDto = itemService.getItemDtl(itemId);  
 model.addAttribute("itemFormDto", itemFormDto);  
 } catch (EntityNotFoundException e) {

// 상품 엔티티가 존재 하지 않을 경우 상품 등록 페이지로 이동  
 model.addAttribute("errorMessage", "존재하지 않는 상품입니다");  
 model.addAttribute("itemFormDto", new ItemFormDto());  
 return "item/itemRegisterForm";  
 }  
  
 return "item/itemRegisterForm";  
 }  
  
 @PostMapping(value = "/store/item/{itemId}")  
 public String itemUpdate(@Valid ItemFormDto itemFormDto, BindingResult bindingResult, @RequestParam("itemImgFile") List<MultipartFile> itemImgFileList, Model model) {

// 상품 수정시 필수 값이 없다면 다시 등록 페이지로 전환  
 if (bindingResult.hasErrors()) {  
 return "item/itemRegisterForm";  
 }

// 첫 번째 이미지가 없다면 에러 메시지와 함께 등록 페이지로 전환 첫번째 이미지는 페이지에서 보여줄 이미지로 사용하여 필수 값으로 지정  
 if (itemImgFileList.get(0).isEmpty() && itemFormDto.getId() == null) {  
 model.addAttribute("errorMessage", "첫번째 상품 이미지는 필수 입력 값입니다");  
 return "item/itemRegisterForm";  
 }  
 try {

// 상품 수정 로직 호출 매개 변수로 상품 정보와 이미지 정보를 담고 itemimgfilelist를 넘겨줌  
 itemService.updateItem(itemFormDto, itemImgFileList);  
 }catch (Exception e) {  
 model.addAttribute("errorMessage", "상품 수정 중 에러가 발생했습니다");  
 return "item/itemRegisterForm";  
 }  
// 정상 수정시 메인 페이지로 이동  
 return "redirect:/";  
 }  
//value에 상품 관리 화면 진입 시 URL에 페이지 번호가 없는 경우와 있는 경우 두가지 매핑  
 @GetMapping(value = {"/store/items", "/store/items/{page}"})  
 public String itemManage(ItemSearchDto itemSearchDto, @PathVariable("page") Optional<Integer> page, Model model){  
// 페이징을 위해 PageRequest.of 메소드를 통해 pageable 객체를 생성 첫 번째 파라미터로는 조회할 페이지 번호 , 두번째는 한번에 가지고올 데이터 수 URL경로에 페이지 번호 있으면 해당 페이지 조회 없으면 0페이지 조회  
 Pageable pageable = [PageRequest.*of*](PageRequest.of)(page.isPresent() ? page.get() : 0, 3);  
 Page<Item> items = itemService.getStoreItemPage(itemSearchDto, pageable);  
// 조회한 상품 데이터 및 페이징 정보를 뷰에 전달  
 model.addAttribute("items", items);

// 페이지 전환 시 기본 검색 조건을 유지한 채 이동 할 수 있도록 뷰에 다시 전달  
 model.addAttribute("itemSearchDto", itemSearchDto);

// 상품 관리 메뉴 하단에 보여줄 페이지 번호의 최대 개수 5개로 했으니 5개 페이지만 보여줌  
 model.addAttribute("maxPage", 5);  
  
 return "item/itemMng";  
 }  
// 상품 상세 페이지 이동  
 @GetMapping(value = "/item/{itemId}")  
 public String itemDtl(Model model, @PathVariable("itemId") Long itemId){  
 ItemFormDto itemFormDto = itemService.getItemDtl(itemId);  
 model.addAttribute("item", itemFormDto);  
 return "item/itemDtl";  
 }  
  
 @GetMapping(value = "/item/main")  
 public String main(ItemSearchDto itemSearchDto, Optional<Integer> page, Model model) {  
  
 Pageable pageable = [PageRequest.*of*](PageRequest.of)(page.isPresent() ? page.get() : 0, 6);  
 Page<MainItemDto> items = itemService.getMainItemPage(itemSearchDto, pageable);  
 model.addAttribute("items", items);  
 model.addAttribute("itemSearchDto", itemSearchDto);  
 model.addAttribute("maxPage", 5);  
  
 return "item/itemMain";  
  
 }  
}

**6-4. 상품등록**

**(ItemFormDto.java)**

@Getter  
@Setter  
public class ItemFormDto {  
 private Long id;  
  
 @NotBlank(message = "상호명은 필수 입력 값입니다")  
 private String storeNm;  
  
 @NotBlank(message = "상품명은 필수 입력 값입니다")  
 private String itemNm;  
  
 @NotNull(message = "가격은 필수 입력 값입니다")  
 private Integer price;  
  
 @NotBlank(message = "상세 설명은 필수 입력 값입니다")  
 private String itemDetail;  
  
 @NotNull(message = "재고는 필수 입력 값입니다")  
 private Integer stockNumber;  
  
 private ItemSellStatus itemSellStatus;  
 private ItemCategory itemCategory;  
 private List<ItemImgDto> itemImgDtoList = new ArrayList<>();  
 private List<Long> itemImgIds = new ArrayList<>();  
 private static ModelMapper *modelMapper* = new ModelMapper();  
  
 public Item createItem() {  
 return *modelMapper*.map(this, Item.class);  
 }  
  
 public static ItemFormDto of(Item item) {  
 return *modelMapper*.map(item, ItemFormDto.class);  
 }  
}

**6-5. 상품등록**

**(ItemImgDto.java)**

@Getter  
@Setter  
public class ItemImgDto {  
 private Long id;  
 private String imgName;  
 private String oriImgName;  
 private String imgUrl;  
 private String repImgYn;  
 private static ModelMapper *modelMapper* = new ModelMapper();  
  
 public static ItemImgDto of(ItemImg itemImg) {  
 return *modelMapper*.map(itemImg, ItemImgDto.class);  
 }  
}

**6-6. 상품등록**

**(ItemSearchDto.java)**

@Getter  
@Setter  
public class ItemSearchDto {  
 private String searchDateType;  
 private ItemSellStatus searchSellStatus;  
 private ItemCategory searchCategory;  
 private String searchBy;  
 private String searchQuery = "";  
}

**6-7. 상품등록**

**(Item.java)**

@Entity  
@Table(name = "item")  
@Getter  
@Setter  
@ToString  
public class Item extends BaseEntity{  
 @Id  
 @Column(name = "itemId")  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private Long id;  
  
 @Column(nullable = false)  
 private String storeNm;  
  
 @Column(nullable = false)  
 private String itemNm;  
  
 @Column(nullable = false)  
 private int price;  
  
 @Column(nullable = false)  
 private int stockNumber;  
  
 @Lob  
 @Column(nullable = false)  
 private String itemDetail;  
  
 @Enumerated(EnumType.*STRING*)  
 private ItemSellStatus itemSellStatus;  
  
 @Enumerated(EnumType.*STRING*)  
 private ItemCategory itemCategory;  
  
// private LocalDateTime regTime;  
// private LocalDateTime updateTime;  
  
 public void updateItem(ItemFormDto itemFormDto) {  
 this.storeNm = itemFormDto.getStoreNm();  
 this.itemNm = itemFormDto.getItemNm();  
 this.price = itemFormDto.getPrice();  
 this.itemDetail = itemFormDto.getItemDetail();  
 this.stockNumber = itemFormDto.getStockNumber();  
 this.itemSellStatus = itemFormDto.getItemSellStatus();  
 this.itemCategory = itemFormDto.getItemCategory();  
 }  
  
 public void removeStock(int stockNumber){  
 int restStock = this.stockNumber - stockNumber;  
 if(restStock<0){  
 throw new OutOfStockException("상품의 재고가 부족합니다. (현재 재고 수량: " + this.stockNumber + ")");  
 }  
 this.stockNumber = restStock;  
 }  
  
 public void addStock(int stockNumber){  
 this.stockNumber += stockNumber;  
 }  
}

**6-8. 상품등록**

**(ItemImg.java)**

@Entity  
@Table(name = "itemImg")  
@Getter  
@Setter  
public class ItemImg {  
  
 @Id  
 @Column(name = "itemImgId")  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private Long id;  
  
 private String imgName; // 이미지 파일명  
 private String oriImgName; // 원본 이미지 파일명  
 private String imgUrl; // 이미지 조회 경로  
 private String repimgYn; // 대표 이미지 여부  
  
 @ManyToOne(fetch = FetchType.*LAZY*)  
 @JoinColumn(name = "itemId")  
 private Item item;  
  
 public void updateItemImg(String oriImgName, String imgName, String imgUrl) {  
 this.oriImgName = oriImgName;  
 this.imgName = imgName;  
 this.imgUrl = imgUrl;  
 }  
}

**6-9. 상품등록**

**(ItemImgRepository.interface)**

public interface ItemImgRepository extends JpaRepository<ItemImg, Long> {  
 List<ItemImg> findByItemIdOrderByIdAsc(Long itemId);  
  
 ItemImg findByItemIdAndRepimgYn(Long itemId, String repimgYn);  
}

**6-10. 상품등록**

**(ItemRepository.java)**

public interface ItemRepository extends JpaRepository<Item, Long>, QuerydslPredicateExecutor<Item>, ItemRepositoryCustom {  
 List<Item> findByItemNm(String itemNm);  
}

**6-11. 상품등록**

**(ItemRepositoryCustom.interface)**

public interface ItemRepositoryCustom {  
 Page<Item> getStoreItemPage(ItemSearchDto itemSearchDto, Pageable pageable);  
  
 Page<MainItemDto> getMainItemPage(ItemSearchDto itemSearchDto, Pageable pageable);  
}

**6-12. 상품등록**

**(ItemImgService.java)**

@Service  
@RequiredArgsConstructor  
@Transactional  
public class ItemImgService {

// value를 통해 application.properties파일에 등록한 itemLocation값을 불러와서 itemLocation 변수에 넣어준다.  
 @Value("${itemImgLocation}")  
 private String itemImgLocation;  
  
 private final ItemImgRepository itemImgRepository;  
 private final FileService fileService;  
  
 public void saveItemImg(ItemImg itemImg, MultipartFile itemImgFile) throws Exception {  
 String oriImgName = itemImgFile.getOriginalFilename();  
 String imgName = "";  
 String imgUrl = "";  
  
 // 파일 업로드  
 if (!StringUtils.*isEmpty*(oriImgName)) {

// 이미지를 등록했다면 저장할 경로, 이름, 파일의 바이트 배열을 파일 업로드 파라미터로 uploadfile 메소드 호출 호출 결과 로컬에 저장된 파일의 이름을 imgname변수에 저장  
 imgName = fileService.uploadFile(itemImgLocation, oriImgName, itemImgFile.getBytes());  
 imgUrl = "/images/item/" + imgName;  
 }  
  
 // 상품 이미지 정보 저장  
 itemImg.updateItemImg(oriImgName, imgName, imgUrl);  
 itemImgRepository.save(itemImg);  
 }  
  
 public void updateItemImg(Long itemImgId, MultipartFile itemImgFile) throws Exception {  
 if (!itemImgFile.isEmpty()) {  
 ItemImg savedItemImg = itemImgRepository.findById(itemImgId)  
 .orElseThrow(EntityNotFoundException::new);  
  
 // 기존 이미지 파일 삭제  
 if (!StringUtils.*isEmpty*(savedItemImg.getImgName())) {  
 fileService.deleteFile(itemImgLocation + "/" + savedItemImg.getImgName());  
 }  
  
 String oriImgName = itemImgFile.getOriginalFilename();  
 String imgName = fileService.uploadFile(itemImgLocation, oriImgName, itemImgFile.getBytes());  
 String imgUrl = "/images/item/" + imgName;  
 savedItemImg.updateItemImg(oriImgName, imgName, imgUrl);  
 }  
 }  
}

**6-13. 상품등록**

**(ItemService.java)**

@Service

@Transactional

@RequiredArgsConstructor

public class ItemService {

private final ItemRepository itemRepository;

private final ItemImgService itemImgService;

private final ItemImgRepository itemImgRepository;

public Long saveItem(ItemFormDto itemFormDto, List<MultipartFile> itemImgFileList) throws Exception {

// 상품 등록

// 상품 등록 폼으로부터 입력 받은 데이터를 이용하여 item 객체를 생성

Item item = itemFormDto.createItem();

// 상품 데이터 저장

itemRepository.save(item);

// 이미지 등록

for (int i = 0; i < itemImgFileList.size(); i++) {

ItemImg itemImg = new ItemImg();

itemImg.setItem(item);

// 첫번째 이미지일 경우 대표 상품 이미지 여부 값을 Y 세팅 나머지는 N 설정

if(i == 0) {

itemImg.setRepimgYn("Y");

} else {

itemImg.setRepimgYn("N");

}

// 상품 이미지 정보 저장

itemImgService.saveItemImg(itemImg, itemImgFileList.get(i));

}

return item.getId();

}

// 상품 데이터를 읽어오는 트랜잭션 읽기 전용 설정 JPA가 더티체킹 수행하지 않아 성능 향상

@Transactional(readOnly = true)

public ItemFormDto getItemDtl(Long itemId) {

// 해당 상품 이미지 조회 등록순으로 가져오기 위해 오름차순으로 가지고옴

List<ItemImg> itemImgList = itemImgRepository.findByItemIdOrderByIdAsc(itemId);

List<ItemImgDto> itemImgDtoList = new ArrayList<>();

for (ItemImg itemImg : itemImgList) {

ItemImgDto itemImgDto = ItemImgDto.*of*(itemImg);

itemImgDtoList.add(itemImgDto);

}

// 상품 아이디를 통해 상품 엔티티 조회

Item item = itemRepository.findById(itemId)

.orElseThrow(EntityNotFoundException::new);

ItemFormDto itemFormDto = ItemFormDto.*of*(item);

itemFormDto.setItemImgDtoList(itemImgDtoList);

return itemFormDto;

}

public Long updateItem(ItemFormDto itemFormDto, List<MultipartFile> itemImgFileList) throws Exception {

// 상품 수정

// 상품 등록 화면으로부터 전달 받은 상품 아이디를 이용하여 상품 엔티티 조회

Item item = itemRepository.findById(itemFormDto.getId())

.orElseThrow(EntityNotFoundException::new);

// 상품 등록 화면으로부터 전달 받은 ItemFormDto를 통해 상품 엔티티 업데이트

item.updateItem(itemFormDto);

// 상품 이미지 아이디 리스트 조회

List<Long> itemImgIds = itemFormDto.getItemImgIds();

// 이미지 등록

for (int i = 0; i < itemImgFileList.size(); i++) {

// 상품 이미지 업데이트 위해 updateItemImg() 메소드에 상품 이미지, 아이디, 파일정보 전달

itemImgService.updateItemImg(itemImgIds.get(i), itemImgFileList.get(i));

}

return item.getId();

}

@Transactional(readOnly = true)

public Page<Item> getStoreItemPage(ItemSearchDto itemSearchDto, Pageable pageable){

return itemRepository.getStoreItemPage(itemSearchDto, pageable);

}

@Transactional(readOnly = true)

public Page<MainItemDto> getMainItemPage(ItemSearchDto itemSearchDto, Pageable pageable){

return itemRepository.getMainItemPage(itemSearchDto, pageable);

}

}

|  |  |
| --- | --- |
| **HTML**  **(HTML)** | |
|  |  |

**1.head.html**

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org"

xmlns:sec="http://www.thymeleaf.org/extras/spring-security">

<div th:fragment="header" class="header">

<div class="first-container">

<div class="first-menu-container">

<div class="mainlogo">

<a class="logo" href="/" aria-label="오늘의디저트로고">오늘의 디저트</a>

</div>

<div class="menu-left">

<ul>

<li sec:authorize="isAnonymous()">

<a href="/members/new">회원가입</a>

</li>

<li sec:authorize="isAnonymous()">

<a href="/members/login">로그인</a>

</li>

<li sec:authorize="isAuthenticated()">

<a href="/members/logout">로그아웃</a>

</li>

</ul>

</div>

<div class="searchbox">

<form class="form-inline my-2 my-lg-0" th:action="@{/}" method="get">

<input name="searchQuery" class="form-control mr-sm-2" type="search" placeholder="Search" aria-label="Search">

<button class="btn btn-outline-success my-2 my-sm-0" type="submit">Search</button>

</form>

</div>

<div class="menu-right">

<ul>

<li sec:authorize="isAuthenticated()">

<a href="/members/myPage">마이 페이지</a>

</li>

<li sec:authorize="isAuthenticated()">

<a href="/orders">구매 이력</a>

</li>

<li sec:authorize="isAuthenticated()">

<a href="/cart">장바 구니</a>

</li>

<li sec:authorize="isAuthenticated()">

<a href="#">리뷰</a>

</li>

<li sec:authorize="hasRole('ROLE\_STORE')">

<a href="/store/item/new">상품 등록</a>

</li>

<li sec:authorize="hasRole('ROLE\_STORE')">

<a href="/store/items">상품 관리</a>

</li>

</ul>

</div>

</div>

</div>

<div class="second-container">

<div class="second-menu-container">

<ul>

<li>

<a href="/item/main">마카롱</a>

</li>

<li>

<a href="#">스콘</a>

</li>

<li>

<a href="#">까눌레</a>

</li>

<li>

<a href="#">휘낭시에</a>

</li>

<li>

<a href="#">버터바</a>

</li>

<li>

<a href="#">쿠키</a>

</li>

<li>

<a href="#">비건</a>

</li>

<li>

<a href="#">ETC</a>

</li>

</ul>

</div>

</div>

</div>

</html>

**2. footer.html**

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org">

<div class="footer" th:fragment="footer">

<footer class="page-footer font-small cyan darken-3">

<div class="footer-copyright text-center py-3">

2020 Shopping Mall Example WebSite

</div>

</footer>

</div>

</html>

**2. Main.html**

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org"

xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"

xmlns:sec="http://www.thymeleaf.org/extras/spring-security"

layout:decorate="~{layouts/layout1}">

<head>

<meta charset="UTF-8">

<title>MAIN</title>

</head>

<body>

<div layout:fragment="content">

<h1> M A I N </h1>

</div>

**4. layoutl.html**

<!DOCTYPE html>

<html xmlns:th="http://www.thymeleaf.org"

xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout">

<head>

<meta charset="UTF-8">

<title>layoutl.html</title>

<th:block layout:fragment="script"></th:block

<th:block layout:fragment="css"></th:block>

</head>

<body>

<div th:replace="fragments/header::header"></div>

<div layout:fragment="content" class="content">

</div>

<div th:replace="fragments/footer::footer"></div>

</body>

</html>

**5. CartList.html**

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org"  
 xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"  
 xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity5"  
 layout:decorate="~{layouts/layout1}">  
  
<head>  
 <meta charset="UTF-8">  
 <meta name="\_csrf" th:content="${\_csrf.token}"/>  
 <meta name="\_csrf\_header" th:content="${\_csrf.headerName}"/>  
 <script src="http://code.jquery.com/jquery-3.5.1.min.js"></script>  
</head>  
  
<!-- 사용자 스크립트 추가 -->  
<th:block layout:fragment="script">  
  
 <script th:inline="javascript">  
  
 $(document).ready(function(){  
 $("input[name=cartChkBox]").change( function(){  
 getOrderTotalPrice();  
 });  
 });  
  
 function getOrderTotalPrice(){  
 var orderTotalPrice = 0;  
 $("input[name=cartChkBox]:checked").each(function() {  
 var cartItemId = $(this).val();  
 var price = $("#price\_" + cartItemId).attr("data-price");  
 var count = $("#count\_" + cartItemId).val();  
 orderTotalPrice += price\*count;  
 });  
  
 $("#orderTotalPrice").html(orderTotalPrice+'원');  
 }  
  
 function changeCount(obj){  
 var count = obj.value;  
 var cartItemId = obj.id.split('\_')[1];  
 var price = $("#price\_" + cartItemId).data("price");  
 var totalPrice = count\*price;  
 $("#totalPrice\_" + cartItemId).html(totalPrice+"원");  
 getOrderTotalPrice();  
 updateCartItemCount(cartItemId, count);  
 }  
  
 function checkAll(){  
 if($("#checkall").prop("checked")){  
 $("input[name=cartChkBox]").prop("checked",true);  
 }else{  
 $("input[name=cartChkBox]").prop("checked",false);  
 }  
 getOrderTotalPrice();  
 }  
  
 function updateCartItemCount(cartItemId, count){  
 var token = $("meta[name='\_csrf']").attr("content");  
 var header = $("meta[name='\_csrf\_header']").attr("content");  
  
 var url = "/cartItem/" + cartItemId+"?count=" + count;  
  
 $.ajax({  
 url : url,  
 type : "PATCH",  
 beforeSend : function(xhr){  
 /\* 데이터를 전송하기 전에 헤더에 csrf값을 설정 \*/  
 xhr.setRequestHeader(header, token);  
 },  
 dataType : "json",  
 cache : false,  
 success : function(result, status){  
 console.log("cartItem count update success");  
 },  
 error : function(jqXHR, status, error){  
  
 if(jqXHR.status == '401'){  
 alert('로그인 후 이용해주세요');  
 location.href='/members/login';  
 } else{  
 alert(jqXHR.responseJSON.message);  
 }  
  
 }  
 });  
 }  
  
 function deleteCartItem(obj){  
 var cartItemId = obj.dataset.id;  
 var token = $("meta[name='\_csrf']").attr("content");  
 var header = $("meta[name='\_csrf\_header']").attr("content");  
  
 var url = "/cartItem/" + cartItemId;  
  
 $.ajax({  
 url : url,  
 type : "DELETE",  
 beforeSend : function(xhr){  
 /\* 데이터를 전송하기 전에 헤더에 csrf값을 설정 \*/  
 xhr.setRequestHeader(header, token);  
 },  
 dataType : "json",  
 cache : false,  
 success : function(result, status){  
 location.href='/cart';  
 },  
 error : function(jqXHR, status, error){  
  
 if(jqXHR.status == '401'){  
 alert('로그인 후 이용해주세요');  
 location.href='/members/login';  
 } else{  
 alert(jqXHR.responseJSON.message);  
 }  
  
 }  
 });  
 }  
  
 function orders(){  
 var token = $("meta[name='\_csrf']").attr("content");  
 var header = $("meta[name='\_csrf\_header']").attr("content");  
  
 var url = "/cart/orders";  
  
 var dataList = new Array();  
 var paramData = new Object();  
  
 $("input[name=cartChkBox]:checked").each(function() {  
 var cartItemId = $(this).val();  
 var data = new Object();  
 data["cartItemId"] = cartItemId;  
 dataList.push(data);  
 });  
  
 paramData['cartOrderDtoList'] = dataList;  
  
 var param = JSON.stringify(paramData);  
  
 $.ajax({  
 url : url,  
 type : "POST",  
 contentType : "application/json",  
 data : param,  
 beforeSend : function(xhr){  
 /\* 데이터를 전송하기 전에 헤더에 csrf값을 설정 \*/  
 xhr.setRequestHeader(header, token);  
 },  
 dataType : "json",  
 cache : false,  
 success : function(result, status){  
 alert("주문이 완료 되었습니다.");  
 location.href='/orders';  
 },  
 error : function(jqXHR, status, error){  
  
 if(jqXHR.status == '401'){  
 alert('로그인 후 이용해주세요');  
 location.href='/members/login';  
 } else{  
 alert(jqXHR.responseJSON.message);  
 }  
  
 }  
 });  
 }  
  
 </script>  
  
</th:block>  
  
<!-- 사용자 CSS 추가 -->  
<th:block layout:fragment="css">  
 <style>  
 .content-mg{  
 margin-left:25%;  
 margin-right:25%;  
 margin-top:2%;  
 margin-bottom:100px;  
 }  
 .repImgDiv{  
 margin-right:15px;  
 margin-left:15px;  
 height:auto;  
 }  
 .repImg{  
 height:100px;  
 width:100px;  
 }  
 .fs18{  
 font-size:18px  
 }  
 .fs24{  
 font-size:24px  
 }  
 </style>  
</th:block>  
  
<div layout:fragment="content" class="content-mg">  
  
 <h2 class="mb-4">  
 장바구니 목록  
 </h2>  
  
 <div>  
  
 <table class="table">  
 <colgroup>  
 <col width="15%"/>  
 <col width="70%"/>  
 <col width="15%"/>  
 </colgroup>  
 <thead>  
 <tr class="text-center">  
 <td>  
 <input type="checkbox" id="checkall" onclick="checkAll()"> 전체선택  
 </td>  
 <td>상품정보</td>  
 <td>상품금액</td>  
 </tr>  
 </thead>  
 <tbody>  
 <tr th:each="cartItem : ${cartItems}">  
 <td class="text-center align-middle">  
 <input type="checkbox" name="cartChkBox" th:value="${cartItem.cartItemId}">  
 </td>  
 <td class="d-flex">  
 <div class="repImgDiv align-self-center">  
 <img th:src="${cartItem.imgUrl}" class = "rounded repImg" th:alt="${cartItem.itemNm}">  
 </div>  
 <div class="align-self-center">  
 <span th:text="${cartItem.storeNm}" class="fs24 font-weight-bold"></span>  
 <span th:text="${cartItem.itemNm}" class="fs24 font-weight-bold"></span>  
 <div class="fs18 font-weight-light">  
 <span class="input-group mt-2">  
 <span th:id="'price\_' + ${cartItem.cartItemId}"  
 th:data-price="${cartItem.price}"  
 th:text="${cartItem.price} + '원'" class="align-self-center mr-2">  
 </span>  
 <input type="number" name="count" th:id="'count\_' + ${cartItem.cartItemId}"  
 th:value="${cartItem.count}" min="1"  
 onchange="changeCount(this)" class="form-control mr-2" >  
 <button type="button" class="close" aria-label="Close">  
 <span aria-hidden="true" th:data-id="${cartItem.cartItemId}" onclick="deleteCartItem(this)">&times;</span>  
 </button>  
 </span>  
 </div>  
 </div>  
 </td>  
 <td class="text-center align-middle">  
 <span th:id="'totalPrice\_' + ${cartItem.cartItemId}"  
 name="totalPrice" th:text="${cartItem.price \* cartItem.count} + '원'">  
 </span>  
 </td>  
 </tr>  
 </tbody>  
 </table>  
  
 <h2 class="text-center">  
 총 주문 금액 : <span id="orderTotalPrice" class="text-danger">0원</span>  
 </h2>  
  
 <div class="text-center mt-3">  
 <button type="button" class="btn btn-primary btn-lg" onclick="orders()">주문하기</button>  
 </div>  
  
 </div>  
  
</div>  
  
</html>

**6. ItemDtl.html**

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org"  
 xmlns:layout="http://www.ultraq.net.nz/thymeleaf/layout"  
 xmlns:sec="https://www.thymeleaf.org/thymeleaf-extras-springsecurity5"  
 layout:decorate="~{layouts/layout1}">  
  
<head>  
 <meta charset="UTF-8">  
 <meta name="\_csrf" th:content="${\_csrf.token}"/>  
 <meta name="\_csrf\_header" th:content="${\_csrf.headerName}"/>  
 <script src="http://code.jquery.com/jquery-3.5.1.min.js"></script>  
</head>  
  
<!-- 사용자 스크립트 추가 -->  
<th:block layout:fragment="script">  
 <script th:inline="javascript">  
 $(document).ready(function(){  
  
 calculateToalPrice();  
  
 $("#count").change( function(){  
 calculateToalPrice();  
 });  
 });  
  
 function calculateToalPrice(){  
 var count = $("#count").val();  
 var price = $("#price").val();  
 var totalPrice = price\*count;  
 $("#totalPrice").html(totalPrice + '원');  
 }  
  
 function order(){  
 var token = $("meta[name='\_csrf']").attr("content");  
 var header = $("meta[name='\_csrf\_header']").attr("content");  
  
 var url = "/order";  
 var paramData = {  
 itemId : $("#itemId").val(),  
 count : $("#count").val()  
 };  
  
 var param = JSON.stringify(paramData);  
  
 $.ajax({  
 url : url,  
 type : "POST",  
 contentType : "application/json",  
 data : param,  
 beforeSend : function(xhr){  
 /\* 데이터를 전송하기 전에 헤더에 csrf값을 설정 \*/  
 xhr.setRequestHeader(header, token);  
 },  
 dataType : "json",  
 cache : false,  
 success : function(result, status){  
 alert("주문이 완료 되었습니다.");  
 location.href='/';  
 },  
 error : function(jqXHR, status, error){  
  
 if(jqXHR.status == '401'){  
 alert('로그인 후 이용해주세요');  
 location.href='/members/login';  
 } else{  
 alert(jqXHR.responseText);  
 }  
  
 }  
 });  
 }  
  
 function addCart(){  
 var token = $("meta[name='\_csrf']").attr("content");  
 var header = $("meta[name='\_csrf\_header']").attr("content");  
  
 var url = "/cart";  
 var paramData = {  
 itemId : $("#itemId").val(),  
 count : $("#count").val()  
 };  
  
 var param = JSON.stringify(paramData);  
  
 $.ajax({  
 url : url,  
 type : "POST",  
 contentType : "application/json",  
 data : param,  
 beforeSend : function(xhr){  
 /\* 데이터를 전송하기 전에 헤더에 csrf값을 설정 \*/  
 xhr.setRequestHeader(header, token);  
 },  
 dataType : "json",  
 cache : false,  
 success : function(result, status){  
 alert("상품을 장바구니에 담았습니다.");  
 location.href='/';  
 },  
 error : function(jqXHR, status, error){  
  
 if(jqXHR.status == '401'){  
 alert('로그인 후 이용해주세요');  
 location.href='/members/login';  
 } else{  
 alert(jqXHR.responseText);  
 }  
  
 }  
 });  
 }  
  
 </script>  
</th:block>  
  
<!-- 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>  
  
<div layout:fragment="content" style="margin-left:25%;margin-right:25%">  
  
 <input type="hidden" id="itemId" th:value="${item.id}">  
  
 <div class="d-flex">  
 <div class="repImgDiv">  
 <img th:src="${item.itemImgDtoList[0].imgUrl}" class = "rounded repImg" th:alt="${item.itemNm}">  
 </div>  
 <div class="wd50">  
 <span th:if="${item.itemSellStatus == T(com.final\_project.TodayDessert.constant.ItemSellStatus).SELL}" class="badge badge-primary mgb-15">  
 판매중  
 </span>  
 <span th:unless="${item.itemSellStatus == T(com.final\_project.TodayDessert.constant.ItemSellStatus).SELL}" class="badge btn-danger mgb-15" >  
 품절  
 </span>  
 <div class="h4" th:text="${item.itemNm}"></div>  
 <hr class="my-4">  
  
 <div class="text-right">  
 <div class="h4 text-danger text-left">  
 <input type="hidden" th:value="${item.price}" id="price" name="price">  
 <span th:text="${item.price}"></span>원  
 </div>  
 <div class="input-group w-50">  
 <div class="input-group-prepend">  
 <span class="input-group-text">수량</span>  
 </div>  
 <input type="number" name="count" id="count" class="form-control" value="1" min="1">  
 </div>  
 </div>  
 <hr class="my-4">  
  
 <div class="text-right mgt-50">  
 <h5>결제 금액</h5>  
 <h3 name="totalPrice" id="totalPrice" class="font-weight-bold"></h3>  
 </div>  
 <div th:if="${item.itemSellStatus == T(com.final\_project.TodayDessert.constant.ItemSellStatus).SELL}" class="text-right">  
 <button type="button" class="btn btn-light border border-primary btn-lg" onclick="addCart()">장바구니 담기</button>  
 <button type="button" class="btn btn-primary btn-lg" onclick="order()">주문하기</button>  
 </div>  
 <div th:unless="${item.itemSellStatus == T(com.final\_project.TodayDessert.constant.ItemSellStatus).SELL}" class="text-right">  
 <button type="button" class="btn btn-danger btn-lg">품절</button>  
 </div>  
 </div>  
 </div>  
  
 <div class="jumbotron jumbotron-fluid mgt-30">  
 <div class="container">  
 <h4 class="display-5">상품 상세 설명</h4>  
 <hr class="my-4">  
 <p class="lead" th:text="${item.itemDetail}"></p>  
 </div>  
 </div>  
  
 <div th:each="itemImg : ${item.itemImgDtoList}" class="text-center">  
 <img th:if="${not #strings.isEmpty(itemImg.imgUrl)}" th:src="${itemImg.imgUrl}" class="rounded mgb-15" width="800">  
 </div>  
  
</div>  
  
</html>

**7. ItemMng.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 charset="UTF-8">  
 <script src="http://code.jquery.com/jquery-3.5.1.min.js"></script>  
</head>  
<!-- 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 searchCategory = $("#searchCategory").val();  
 var searchBy = $("#searchBy").val();  
 var searchQuery = $("#searchQuery").val();  
  
 location.href="/store/items/" + page + "?searchDateType=" + searchDateType  
 + "&searchSellStatus=" + searchSellStatus  
 + "&searchCategory=" + searchCategory  
 + "&searchBy=" + searchBy  
 + "&searchQuery=" + searchQuery;  
 }  
  
 </script>  
</th:block>  
  
<!-- 사용자 CSS 추가 -->  
<th:block layout:fragment="css">  
 <style>  
 select{  
 margin-right:10px;  
 }  
 </style>  
</th:block>  
  
<div layout:fragment="content">  
  
 <form th:action="@{'/store/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>카테고리</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="'/store/item/'+${item.id}" th:text="${item.itemNm}"></a>  
 </td>  
 <td th:text="${item.itemSellStatus == T(com.final\_project.TodayDessert.constant.ItemSellStatus).SELL} ? '판매중' : '품절'"></td>  
 <td th:text="${item.itemCategory}"></td>  
 <td th:text="${item.createdBy}"></td>  
 <td th:text="${item.regTime}"></td>  
 </tr>  
 </tbody>  
 </table>  
  
 <div th:with="start=${(items.number/maxPage)\*maxPage + 1}, end=(${(items.totalPages == 0) ? 1 : (start + (maxPage - 1) < items.totalPages ? start + (maxPage - 1) : items.totalPages)})" >  
 <ul class="pagination justify-content-center">  
  
 <li class="page-item" th:classappend="${items.first}?'disabled'">  
 <a th:onclick="'javascript:page(' + ${items.number - 1} + ')'" aria-label='Previous' class="page-link">  
 <span aria-hidden='true'>Previous</span>  
 </a>  
 </li>  
  
 <li class="page-item" th:each="page: ${#numbers.sequence(start, end)}" th:classappend="${items.number eq page-1}?'active':''">  
 <a th:onclick="'javascript:page(' + ${page - 1} + ')'" th:inline="text" class="page-link">[[${page}]]</a>  
 </li>  
  
 <li class="page-item" th:classappend="${items.last}?'disabled'">  
 <a th:onclick="'javascript:page(' + ${items.number + 1} + ')'" aria-label='Next' class="page-link">  
 <span aria-hidden='true'>Next</span>  
 </a>  
 </li>  
  
 </ul>  
 </div>  
  
 <div class="form-inline justify-content-center" th:object="${itemSearchDto}">  
 <select th:field="\*{searchDateType}" class="form-control" style="width:auto;">  
 <option value="all">전체기간</option>  
 <option value="1d">1일</option>  
 <option value="1w">1주</option>  
 <option value="1m">1개월</option>  
 <option value="6m">6개월</option>  
 </select>  
 <select th:field="\*{searchSellStatus}" class="form-control" style="width:auto;">  
 <option value="">판매상태(전체)</option>  
 <option value="SELL">판매</option>  
 <option value="SOLD\_OUT">품절</option>  
 </select>  
 <select th:field="\*{searchCategory}" class="form-control" style="width:auto;">  
 <option value="">카테고리(전체)</option>  
 <option value="MACARON">마카롱</option>  
 <option value="SCONE">스콘</option>  
 <option value="CANNELE">까눌레</option>  
 <option value="BUTTERBAR">버터바</option>  
 <option value="FINANCIER">휘낭시에</option>  
 <option value="COOKIES">쿠키</option>  
 <option value="VEGAN">비건</option>  
 <option value="ETC">그 외</option>  
 </select>  
 <select th:field="\*{searchBy}" class="form-control" style="width:auto;">  
 <option value="itemNm">상품명</option>  
 <option value="createdBy">등록자</option>  
 </select>  
 <input th:field="\*{searchQuery}" type="text" class="form-control" placeholder="검색어를 입력해주세요">  
 <button id="searchBtn" type="submit" class="btn btn-primary">검색</button>  
 </div>  
 </form>  
  
</div>  
  
</html>

**7. ItemRegister.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 charset="UTF-8">  
 <script src="http://code.jquery.com/jquery-3.5.1.min.js"></script>  
</head>  
<!-- 사용자 스크립트 추가 -->  
<th:block layout:fragment="script">  
  
 <script th:inline="javascript">  
 $(document).ready(function(){  
 var errorMessage = [[${errorMessage}]];  
 if(errorMessage != null){  
 alert(errorMessage);  
 }  
  
 bindDomEvent();  
  
 });  
  
 function bindDomEvent(){  
 $(".custom-file-input").on("change", function() {  
 var fileName = $(this).val().split("\\").pop(); //이미지 파일명  
 var fileExt = fileName.substring(fileName.lastIndexOf(".")+1); // 확장자 추출  
 fileExt = fileExt.toLowerCase(); //소문자 변환  
  
 if(fileExt != "jpg" && fileExt != "jpeg" && fileExt != "gif" && fileExt != "png" && fileExt != "bmp"){  
 alert("이미지 파일만 등록이 가능합니다.");  
 return;  
 }  
  
 $(this).siblings(".custom-file-label").html(fileName);  
 });  
 }  
  
 </script>  
  
</th:block>  
  
<!-- 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="${itemFormDto}">  
 <input type="hidden" th:name="${\_csrf.parameterName}" th:value="${\_csrf.token}">  
 <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>  
  
 <div class="form-group">  
 <select th:field="\*{itemCategory}" class="custom-select">  
 <option value="MACARON">마카롱</option>  
 <option value="SCONE">스콘</option>  
 <option value="CANNELE">까눌레</option>  
 <option value="BUTTERBAR">버터바</option>  
 <option value="FINANCIER">휘낭시에</option>  
 <option value="COOKIES">쿠키</option>  
 <option value="VEGAN">비건</option>  
 <option value="ETC">그 외</option>  
 </select>  
 </div>  
  
 <div class="input-group">  
 <div class="input-group-prepend">  
 <span class="input-group-text">상호명</span>  
 </div>  
 <input type="text" th:field="\*{storeNm}" class="form-control" placeholder="상품명을 입력해주세요">  
 </div>  
 <p th:if="${#fields.hasErrors('storeNm')}" th:errors="\*{storeNm}" class="fieldError">Incorrect data</p>  
  
 <div class="input-group">  
 <div class="input-group-prepend">  
 <span class="input-group-text">상품명</span>  
 </div>  
 <input type="text" th:field="\*{itemNm}" class="form-control" placeholder="상품명을 입력해주세요">  
 </div>  
 <p th:if="${#fields.hasErrors('itemNm')}" th:errors="\*{itemNm}" class="fieldError">Incorrect data</p>  
  
 <div class="input-group">  
 <div class="input-group-prepend">  
 <span class="input-group-text">가격</span>  
 </div>  
 <input type="number" th:field="\*{price}" class="form-control" placeholder="상품의 가격을 입력해주세요">  
 </div>  
 <p th:if="${#fields.hasErrors('price')}" th:errors="\*{price}" class="fieldError">Incorrect data</p>  
  
 <div class="input-group">  
 <div class="input-group-prepend">  
 <span class="input-group-text">재고</span>  
 </div>  
 <input type="number" th:field="\*{stockNumber}" class="form-control" placeholder="상품의 재고를 입력해주세요">  
 </div>  
 <p th:if="${#fields.hasErrors('stockNumber')}" th:errors="\*{stockNumber}" class="fieldError">Incorrect data</p>  
  
 <div class="input-group">  
 <div class="input-group-prepend">  
 <span class="input-group-text">상품 상세 내용</span>  
 </div>  
 <textarea class="form-control" aria-label="With textarea" th:field="\*{itemDetail}"></textarea>  
 </div>  
 <p th:if="${#fields.hasErrors('itemDetail')}" th:errors="\*{itemDetail}" class="fieldError">Incorrect data</p>  
  
 <div th:if="${#lists.isEmpty(itemFormDto.itemImgDtoList)}">  
 <div class="form-group" th:each="num: ${#numbers.sequence(1,5)}">  
 <div class="custom-file img-div">  
 <input type="file" class="custom-file-input" name="itemImgFile">  
 <label class="custom-file-label" th:text="상품이미지 + ${num}"></label>  
 </div>  
 </div>  
 </div>  
  
 <div th:if = "${not #lists.isEmpty(itemFormDto.itemImgDtoList)}">  
 <div class="form-group" th:each="itemImgDto, status: ${itemFormDto.itemImgDtoList}">  
 <div class="custom-file img-div">  
 <input type="file" class="custom-file-input" name="itemImgFile">  
 <input type="hidden" name="itemImgIds" th:value="${itemImgDto.id}">  
 <label class="custom-file-label" th:text="${not #strings.isEmpty(itemImgDto.oriImgName)} ? ${itemImgDto.oriImgName} : '상품이미지' + ${status.index+1}"></label>  
 </div>  
 </div>  
 </div>  
  
 <div th:if="${#strings.isEmpty(itemFormDto.id)}" style="text-align: center">  
 <button th:formaction="@{/store/item/new}" type="submit" class="btn btn-primary">저장</button>  
 </div>  
 <div th:unless="${#strings.isEmpty(itemFormDto.id)}" style="text-align: center">  
 <button th:formaction="@{'/store/item/' + ${itemFormDto.id} }" type="submit" class="btn btn-primary">수정</button>  
 </div>  
<!-- <input type="hidden" th:name="${\_csrf.parameterName}" th:value="${\_csrf.token}">-->  
  
 </form>  
  
</div>  
  
</html>

**5. Application.properties**

# SERVER

server.port = 8081

//애플리케이션의 실행할 포트를 '8081' 설정

# 데이터베이스

spring.datasource.driver-class-name=oracle.jdbc.OracleDriver

spring.datasource.url=jdbc:oracle:thin:@localhost:1521/xe

spring.datasource.username=project

spring.datasource.password=project1234

//오라클 연결 및 사용 될 계정 및 비밀번호 입력

# JPA

spring.jpa.database=oracle

spring.jpa.hibernate.ddl-auto=create

spring.jpa.generate-ddl=false

spring.jpa.show-sql=true

spring.jpa.database-platform=org.hibernate.dialect.Oracle10gDialect

spring.jpa.properties.hibernate.format\_sql=true

spring.jpa.properties.hibernate.show\_sql=true

// 오라클과 JPA 데이터베이스 연결

# LOGGING SETTING

logging.level.org.hibernate=info

# 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:/Users/EZEN-13/Desktop/img/item

#리소스 업로드 경로

uploadPath=file:///C:/Users/EZEN-13/Desktop/img/

|  |  |
| --- | --- |
| **환경 구성** | |
|  |  |

|  |
| --- |
|  |

**-build.gradle**

dependencies **{**

implementation 'org.springframework.boot:spring-boot-starter-data-jpa

implementation 'org.springframework.boot:spring-boot-starter-security'

implementation 'org.springframework.boot:spring-boot-starter-thymeleaf'

implementation 'org.springframework.boot:spring-boot-starter-validation'

implementation 'org.springframework.boot:spring-boot-starter-web'

implementation 'org.thymeleaf.extras:thymeleaf-extras-springsecurity5'

compileOnly 'org.projectlombok:lombok'

runtimeOnly 'com.oracle.database.jdbc:ojdbc8'

developmentOnly 'org.springframework.boot:spring-boot-devtools'

implementation group: 'com.github.zhanhb', name: 'thymeleaf-layout-dialect', version: '2.5.1'

annotationProcessor 'org.projectlombok:lombok'

testImplementation 'org.springframework.boot:spring-boot-starter-test'

testImplementation 'org.springframework.security:spring-security-test'

implementation 'nz.net.ultraq.thymeleaf:thymeleaf-layout-dialect'

implementation group: 'org.modelmapper', name: 'modelmapper', version: '2.4.2'

implementation "com.querydsl:querydsl-jpa:$**{**queryDslVersion**}**"

implementation "com.querydsl:querydsl-apt:$**{**queryDslVersion**}**"

**}**

**참고자료**

* 참고문헌 목혹을 적용하여 작성합니다. (참고문헌)
* 참고문헌 목혹을 적용하여 작성합니다. (참고문헌)

**[부록 1-4] 부록 스타일을 적용합니다. (‘기타제목’ 목록을 사용하여 작성합니다.)**

|  |  |  |
| --- | --- | --- |
|  |  |  |