Team Project

Shopping Mall

최원익, 이상암

2024.06.17 ~ 2024.08.19

Junior 백앤드 개발자

프로젝트 기획

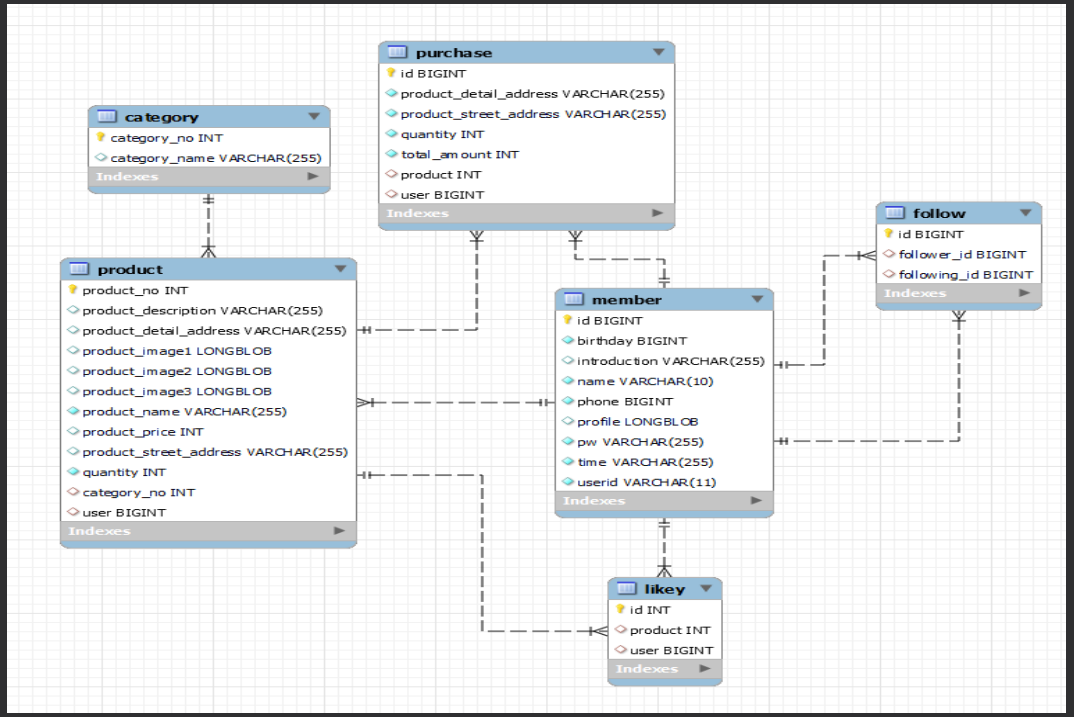
- **[서비스 기능]**
  - 판매자는 판매하고 싶은 상품을 개시할 수 있다.
  - 구매자는 개시 되어 있는 상품을 구매 할 수 있다.
- [db 테이블 구성]

테이블	구성 계획
유저	Id, pw, 이름, 생일, 프로필사진, 휴대폰 번호, 생성 날짜
상품	상품이름, 주소, 상세주소, 이미지 3개, 상품설명, 가격, 상품수량, 유저, 카테고리
카테고리	카테고리 이름, 상품
팔로우	팔로워, 팔로잉
좋아요	상품, 유저
상품구매	주소, 상세주소, 수량, 가격, 상품, 유저

기능 구현

- **[로그인]**
  - Spring Security를 사용한 보안 강화
  - 아이디 중복 확인 및 비밀번호 패턴 정형화
  - 로그인 시 로그인 하기 전 페이지로 넘어갈 수 있도록 설정
  - 회원 소개 설정 및 프로필 사진을 변경할 수 있다.
  - 유저간 팔로우 가능
- [상품 등록]
  - 주소api를 이용한 주소 설정
- [상품 구매]
  - 카카오페이를 이용하여 상품 구매 가능
  - 상품에 '좋아요' 버튼을 누를 수 있다.

db 구축



데이터베이스의 join문을 배우지 않아 구현에 어려움을 겪었다.

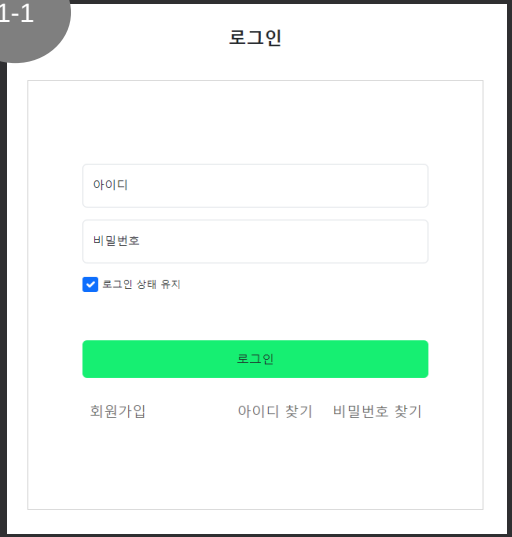
로그인

1

```
    SecurityFilterChain filterChain(HttpSecurity http) throws Exception {
CsrfTokenRequestAttributeHandler requestHandler = new CsrfTokenRequestAttributeHandler();
requestHandler.setCsrfRequestAttributeName("_csrf");

return http.csrf(csrf -> csrf.csrfTokenRequestHandler(requestHandler)
        .csrfTokenRepository(CookieCsrfTokenRepository.withHttpOnlyFalse()).ignoringRequestMatchers("/header"))
        .authorizeHttpRequests(request -> request
                .requestMatchers("/follow","product/productADD","/kakaoPay","/likey").authenticated()
                .requestMatchers(PathRequest.toStaticResources().atCommonLocations()).permitAll()
                .requestMatchers("join/**").permitAll()
                .anyRequest().permitAll())
        .formLogin(login -> login
                .loginPage("/login/login")
                .loginProcessingUrl("/login/login")
                .usernameParameter("userid")
                .passwordParameter("pw")
                .successHandler(new LoginSuccessHandler())
                .failureHandler(new LoginFailHandler())
                .permitAll()
                )
        .rememberMe(customizer -> customizer
                .rememberMeParameter("remember")
                .tokenValiditySeconds(2678400)
                .userDetailsService(myUserDetailsService)
                .authenticationSuccessHandler(new LoginSuccessHandler()))
        .logout(logout -> logout
                .logoutUrl("/logout")
                .logoutSuccessUrl("/index")
                .deleteCookies("remember")
                .permitAll())
        .addFilterAfter(new CsrfCookieFilter(), BasicAuthenticationFilter.class)
        .httpBasic(Customizer.withDefaults())
        .build();
}
```

1-1



- 허용되지 않은 페이지에 접속을 시도하면 security가 동작하여 커스텀 로그인 페이지로 넘어가도록 설정함.
- 로그인에 성공한다면 SuccessHandler가, 실패한다면 FailHander가 동작한다.

로그인

1-2

```
s LoginSuccessHandler extends SimpleUrlAuthenticationSuccessHandler {

    ide
    c void onAuthenticationSuccess(HttpServletRequest request, HttpServletResponse response,
            Authentication authentication) throws IOException, ServletException {
        super.clearAuthenticationAttributes(request);


        RequestCache requestCache = new HttpSessionRequestCache();
        SavedRequest savedRequest = requestCache.getRequest(request, response);

        if (savedRequest != null) {
            String url = savedRequest.getRedirectUrl();
            log.info(url);
            if (url == null || url.equals("")) {
                url = "/index";
            } else if (url.contains("/join")) {
                url = "/index";
            }
            requestCache.removeRequest(request, response);
            getRedirectStrategy().sendRedirect(request, response, url);
        }
        else if(savedRequest==null&&MemberService.backPage.equals("http://localhost:8080/login/login")){
            String url = "http://localhost:8080/index";
            requestCache.removeRequest(request, response);
            getRedirectStrategy().sendRedirect(request, response, url);
        }
        else {
            String url =MemberService.backPage;
            requestCache.removeRequest(request, response);
            getRedirectStrategy().sendRedirect(request, response, url);
        }

        super.onAuthenticationSuccess(request, response, authentication);

        UserHandler.loginSession(request);

        // 컨트롤러에 전달
        request.setAttribute("userid", authentication.getName());
    }

}
```

1-3

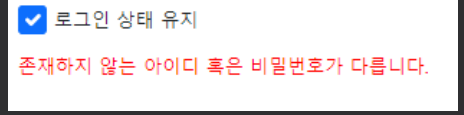
```
lass LoginFailHandler extends SimpleUrlAuthenticationFailureHandler {

@Override
public void onAuthenticationFailure(HttpServletRequest request, HttpServletResponse response,
        AuthenticationException exception) throws IOException, ServletException {

    String message = getMessage(exception);
    String redirectUrl = "/login/login?hasMessage=true&message=" + message;
    setDefaultFailureUrl(redirectUrl);
    super.onAuthenticationFailure(request, response, exception);
}

private static String getMessage(AuthenticationException exception) throws UnsupportedEncodingException {
    String message = exception.getMessage();
    String encodeMessage = URLEncoder.encode(message, "UTF-8");

    return encodeMessage;
}
```



과정 1-2 success

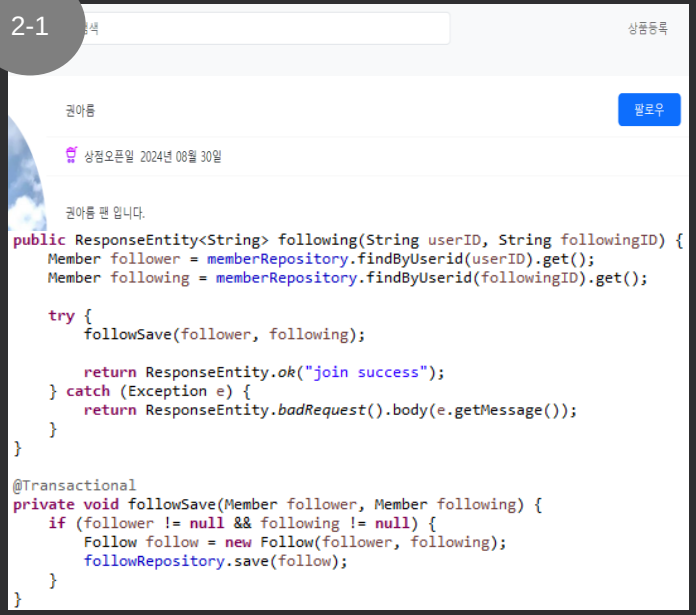
- 로그인에 성공한다면 이전에 접속되어 있던 페이지로 넘어 갈 수 있도록 설정

과정 1-3 fail

- 로그인에 실패한다면 실패 메시지와 함께 다시 로그인 할 수 있도록 설정

팔로우

2-1



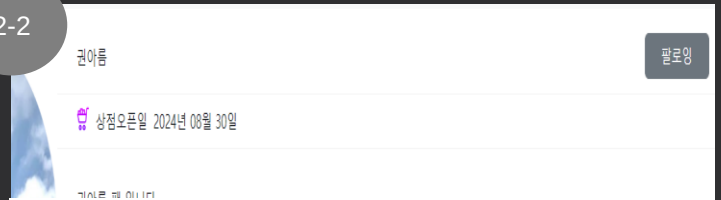

```
public ResponseEntity<String> following(String userID, String followingID) {
    Member follower = memberRepository.findByUserid(userID).get();
    Member following = memberRepository.findByUserid(followingID).get();

    try {
        followSave(follower, following);

        return ResponseEntity.ok("join success");
    } catch (Exception e) {
        return ResponseEntity.badRequest().body(e.getMessage());
    }
}

@Transactional
private void followSave(Member follower, Member following) {
    if (follower != null && following != null) {
        Follow follow = new Follow(follower, following);
        followRepository.save(follow);
    }
}
```

2-2




```
@Transactional
public boolean unfollow(String userID, String followingID) {
    Member follower = memberRepository.findByUserid(userID).get();
    Member following = memberRepository.findByUserid(followingID).get();

    Long followerId = follower.getId();
    Long followingId = following.getId();

    if (isFollowing(followerId, following.getUserid())) {
        followRepository.deleteByFollowerAndFollowing(followerId, followingId);
        return true;
    }
    return false;
}
```

과정 2-1 팔로우

- 상대가 팔로우 되어 있지 않을 경우 팔로우 할 수 있다.

과정 2-2 언팔로우

- 상대가 팔로우 되어 있을 경우 상대를 언팔로우 할 수 있다.


```
public boolean add(Optional<Product> product,Optional<Member> member) {
    boolean bool = (likeyRepository.findUserCountByProduct(member.get(), product.get()) == 1) ? true : false;

    Likey likey = Likey.builder()
            .user(member.get())
            .product(product.get())
            .build();
        ol==false) {
        likeyRepository.save(likey);

        turn true;
        {
        likeyRepository.deleteByUserAndProduct(likey.getUser(),likey.getProduct());

        return false;
    }
}
```

3-1

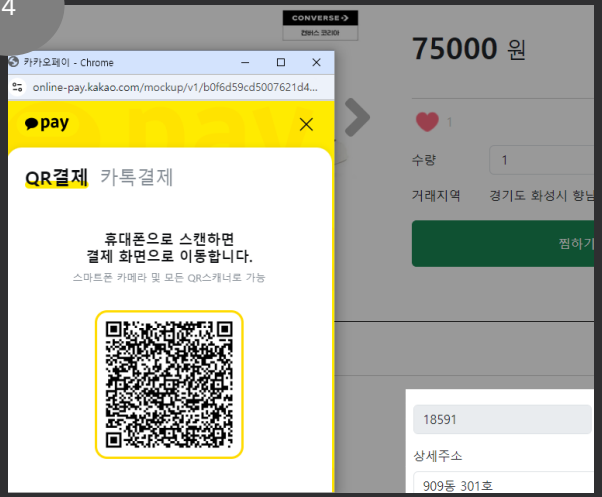
3-2

과정 3 좋아요 버튼
- 좋아요 버튼으로 장바구니 역할을 구현

과정 3-1 좋아요 추가
- 유저가 상품을 아직 '좋아요' 추가를 안 했을 경우 db에 추가 됨

과정 2-2 좋아요 취소
- 유저가 이미 상품을 '좋아요' 추가를 했을 경우 db에서 삭제 됨

4



4-1

```
/kakaoPay',
    'post',
reSend: function(xhr) {
    xhr.setRequestHeader(header, token);
},
headers: { 'Content-Type': 'application/json' },
dataType: 'json',
data: JSON.stringify({
    partnerOrderId: kakaoPay.partnerOrderId,
    partnerUserId: kakaoPay.partnerUserId,
    itemName: kakaoPay.itemName,
    quantity: $quantity.value,
    totalAmount: kakaoPay.totalAmount*$quantity.value,
    taxFreeAmount: kakaoPay.taxFreeAmount,
    productStreetaddress: $productStreetaddress.value,
    productDetailaddress: $productDetailaddress.value
}),
success: (data) => {
    var kakaopay = {
        ref: null,
    };

    kakaopay.ref = window.open('', 'paypopup', 'width=426,height=510,toolbar=no');
    if (kakaopay.ref) {
        setTimeout(function() {
            kakaopay.ref.location.href = data.next_redirect_pc_url;
        }, 0);

        var checkPopupClosed = setInterval(function() {
            if (kakaopay.ref.closed) {
                clearInterval(checkPopupClosed);

                window.location.href="http://localhost:8080/complete"
            }
        }, 1000);
    } else {
        throw new Error("popup을 열 수 없습니다!(cannot open popup)");
    }
```

과정 4 상품 구매
- 상품 구매 시 카카오페이 popup 실행

과정 4-1 카카오 페이
- 카카오페이 api에 구매 과정에서 필요한 정보 제공
- 상품 이름, 구매자 아이디, 상품 총액, 구매자 주소, 상품 개 수

4-2

```
ide
    ReadyResponse ready(KakaoPayDTO kakaoPay) {
    // Request header
    HttpHeaders headers = new HttpHeaders();
    // 여기까진 ok
    headers.add("Authorization", "DEV_SECRET_KEY " + kakaopaySecretKey);
    headers.setContentType(MediaType.APPLICATION_JSON);

    product = kakaoPay;

    // Request param
    ReadyRequest readyRequest = ReadyRequest.builder().cid(cid).partnerOrderId(kakaoPay.getPartnerOrderId()
            .partnerUserId(kakaoPay.getPartnerUserId()).itemName(kakaoPay.getItemName())
            .quantity(kakaoPay.getQuantity()).totalAmount(kakaoPay.getTotalAmount())
            .taxFreeAmount(kakaoPay.getTaxFreeAmount()).approvalUrl(sampleHost + "/approve")
            .cancelUrl(sampleHost + "/cancel").failUrl(sampleHost + "/fail").build();

    // Send reqeust
    HttpEntity<ReadyRequest> entityMap = new HttpEntity<>(readyRequest, headers);

    ResponseEntity<ReadyResponse> response = new RestTemplate()
            .postForEntity("https://open-api.kakaopay.com/online/v1/payment/ready", entityMap, ReadyRespons

    ReadyResponse readyResponse = response.getBody();

    // 주문번호와 TID를 매핑해서 저장해놓는다.
    // Mapping TID with partner_order_id then save it to use for approval request.
    this.tid = readyResponse.getTid();
    return readyResponse;
}
```

4-3

```
e
    String approve(String pgToken) {
    // ready할 때 저장해놓은 TID로 승인 요청
    // Call "Execute approved payment" API by pg_token, TID mapping to the current
    // payment transaction and other parameters.
    HttpHeaders headers = new HttpHeaders();
    headers.add("Authorization", "SECRET_KEY " + kakaopaySecretKey);
    headers.setContentType(MediaType.APPLICATION_JSON);

    // Request param
    ApproveRequest approveRequest = ApproveRequest.builder().cid(cid).tid(tid)
            .partnerOrderId(product.getPartnerOrderId()).partnerUserId(product.getPartnerUserId()).pgToken(pgToken)
            .build();

    // Send Request
    HttpEntity<ApproveRequest> entityMap = new HttpEntity<>(approveRequest, headers);
    try {
        ResponseEntity<String> response = new RestTemplate()
                .postForEntity("https://open-api.kakaopay.com/online/v1/payment/approve", entityMap, String.class);

        // 승인 결과를 저장한다.
        // save the result of approval
        String approveResponse = response.getBody();

        String productCODE = product.getPartnerOrderId();
        String result = productCODE.replace("product", "");
        int productID = Integer.parseInt(result);

        Optional<Product> Op = productRepository.findByProductNo(productID);
        // 수량 체크
        productUpdate(Op.get(), product.getQuantity());

        // 구매내역 저장
        Purchase purchase = Purchase.builder().product(Op.get()).quantity(product.getQuantity())
                .totalAmount(product.getTotalAmount()).productStreetaddress(product.getProductStreetaddress())
                .productDetailaddress(product.getProductDetailaddress()).build();
        purchaseRepository.save(purchase);

        return approveResponse;
    } catch (HttpStatusCodeException ex) {
        return ex.getResponseBodyAsString();
    }
}
```

과정 4-2 카카오페이 api

- 이 메서드는 카카오페이 결제를 준비하기 위한 요청을 보내는 기능 수행
- 사용자의 주문 정보 (KakaoPayDTO)를 받아서, 그 정보에 기반해 카카오페이 API에 결제 준비 요청을 보내고, 그 결과로 받은 TID(거래 ID)를 저장한 뒤, 응답(ReadyResponse)을 반환

과정 4-3 구매성공

- 구매 내역 db에 저장

Thank You

최원익
010-9013-6753
dyd975@naver.com