

KH FINAL PROJECT

스토리보드

풍당

김연중, 김인곤, 윤성훈, 이소윤, 황선애

문의하기

배송은 언제 도착하나요?

배송문의

답변완료

언제쯤?

2022-07-28 00:41:14

삭제하기

내일 도착합니다

문의합니다

결제문의

답변대기

상실하게 답변해주세요

2022-07-28 00:41:03

삭제하기

답변을 기다리는 중입니다.

```

<c:forEach items="${ask}" var="ask">
  <form action="deleteAsk.do" method="post" id="deleteAsk">
    <div class="ask_box" style="position: relative;">
      <li style="display: flex; margin-top: 30px; margin-bottom: 10px;">
        <div class="ask01">${ask.ask_title}</div> <span class="ask_sort">${ask.ask_category}</span>
        <c:if test="${ask.reply_yn eq 'N'}">
          <span class="reply_end">답변대기</span>
        </c:if> <c:if test="${ask.reply_yn eq 'Y'}">
          <span class="reply_end2">답변완료</span>
        </c:if> <span class="toggle tg1">89660</span> <span class="toggle tg2">89650</span>
      </li>
      <li class="content">
        <div style="display: flex;">
          <span style="width: 75%;">${ask.ask_content}</span>
          <div class="ask_date">
            <p>${ask.ask_date}</p>
            <p class="ask_delete" id="ask_delete${ask.ask_no}">삭제하기</p>
            <input type="hidden" name="ask_no" value="${ask.ask_no}">
          </div>
        </div>
        <hr style="width: 100%; margin: 20px auto 20px;">
        <div style="width: 75%;"><c:if test="${ask.reply_yn eq 'N'}">답변을 기다리는 중입니다. </c:if></div>
        <div><c:if test="${ask.reply_yn eq 'Y'}">${ask.ans_content}</c:if></div>
      </li>
    </div>
  </form>
</c:forEach>

```


- session에 저장된 email 값을 사용하여 현재 로그인한 회원이 문의한 내역만 조회 되도록 함
- jstl을 사용해 문의 내용, 답변, 날짜, 문의 카테고리, 답변 상태를 화면에 나타냄




```
<c:if test="${empty ask}">
  <div style="position: relative; top: 60px;">
    <h4>등록된 1:1 문의가 없습니다.</h4>
    <h4>궁금하거나 건의할 사항이 있다면 언제든지 문의해주세요!</h4>
  </div>
  <a href="<%=request.getContextPath()%>/customerCenter/doAsk"><button
    class="btn btn-fill-fcolor" style="position: relative; top: 90px;">문의하기</button></a>
</c:if>
```

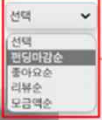
등록된 1:1 문의가 없습니다.
궁금하거나 건의할 사항이 있다면 언제든지 문의해주세요!

문의하기

- 등록된 문의사항이 없다면 문의가 없다는 문구와 함께 문의하기 버튼이 중앙에 정렬되어 화면에 보여짐



```


@GetMapping("/list/cateC1")
public ModelAndView fundingCatelist1(ModelAndView mv, String C1, HttpSession session) {
    mv.addObject("allProducts", service.selectCateProducts1(C1));
    mv.addObject("category_id", "C1");
    Member member = (Member)session.getAttribute("loginInfo");
    if (member != null) {
        String email = member.getEmail();
        mv.addObject("alarm", alarmService.countAlarm(email));
    }

    mv.setViewName("funding/fundinglist");
    return mv;
}


```



반려동물 | 주식회사 다르텍
373% 500,000원
11일 남음



반려동물 | 오티피
8% 1,500,000원
11일 남음



반려동물 | 주식회사 맥파이테크
12% 1,000,000원
5일 남음

```

@GetMapping("/list")
public ModelAndView fundinglist(ModelAndView mv, @RequestParam(value = "category_id", defaultValue = "") String category_id,
    @RequestParam(value = "cateSelect", defaultValue = "") String cateSelect, HttpSession session) {

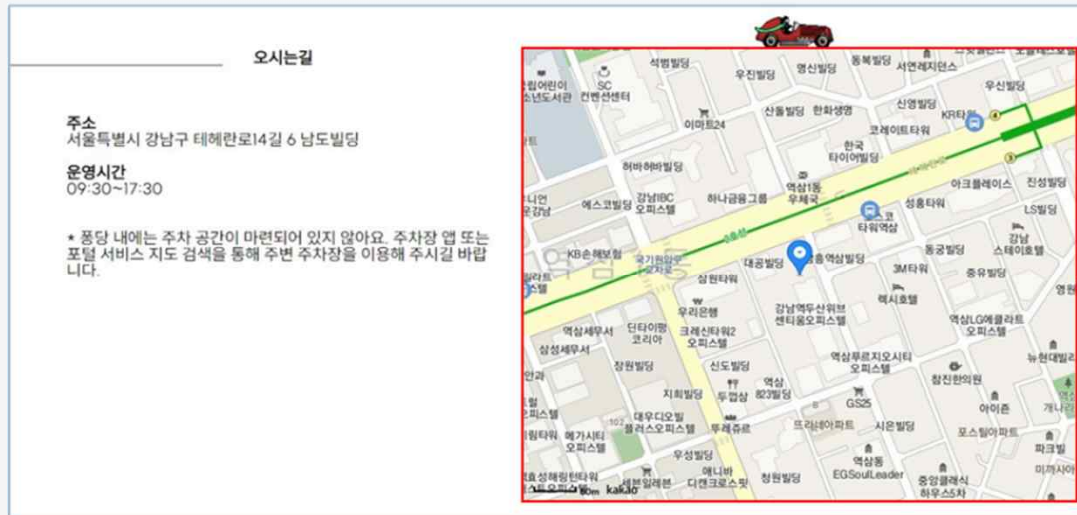
    Member member = (Member)session.getAttribute("loginInfo");
    if (member != null) {
        String email = member.getEmail();
        mv.addObject("alarm", alarmService.countAlarm(email));
    }

    Map<String, String> map = new HashMap<String, String>();
    map.put("category_id", category_id);
    map.put("cateSelect", cateSelect);

    mv.addObject("category_id", category_id);
    mv.addObject("allProducts", service.selectAllProducts(map));
    mv.setViewName("funding/fundinglist");
    return mv;
}

```

- 선택한 카테고리에 맞는 상품 목록 조회
 - DB 삽입 시 설정한 카테고리 ID에 맞는 상품 목록만을 조회
=> C1 : 가전, C2 : 뷰티, C3 : 식품, C4 : 캠핑, C5 : 반려동물, C6 : 기타
- 상품 정렬 기준을 선택
 - sql문 실행 시 ORDER BY 조건으로 선택한 정렬 값을 넣어 상품을 조회



```
<script type="text/javascript"
  src="//dapi.kakao.com/v2/maps/sdk.js?
  appkey=c2c1f7266c7882e29f054c8c9c9fa5fc"></script>
<script>
  var container = document.getElementById('map'); //지도 표시 div
  var options = {
    center : new kakao.maps.LatLng(37.498933398783876,
      127.03280117765928), //지도의 중심좌표
    level : 3//지도의 확대 레벨
  };
  var map = new kakao.maps.Map(container, options);
  // 마커가 표시될 위치
  var markerPosition = new kakao.maps.LatLng
    (37.498933398783876,127.03280117765928);

  // 마커 생성
  var marker = new kakao.maps.Marker({
    position : markerPosition
  });
  // 마커가 지도 위에 표시되도록 설정
  marker.setMap(map);

  // 아래 코드는 지도 위의 마커를 제거하는 코드
  // marker.setMap(null);
</script>
```

- 카카오맵 API를 사용하여 지도의 중심좌표를 위도와 경도로 찾고, 마커를 지도에 표시

아이디 찾기

로그인

회원가입

아이디·비밀번호 찾기

아이디 찾기

비밀번호 찾기

동당은 이메일을 아이디로 사용합니다.
소유하고 계신 계정을 입력해보세요.
가입여부를 확인해드립니다.

이메일 계정

확인

```
function emailValidate(email) {  
  // 이메일 유효성 정규식검사  
  var filter = /^[^(\w-\.]+)@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.)(\[[\w-]+\.\.))|([a-zA-Z]{2,4}|[0-9]{1,3})(\.)?$/;  
  
  if(email == '') {  
    $(".error_comment").html("이메일 주소를 다시 입력해주세요.");  
    return false;  
  } else if(!filter.test(email)) {  
    // test() : 주어진 문자열이 정규 표현식을 만족하는지 판별, return값 [true | false]  
    $(".error_comment").html("이메일 형식에 맞게 다시 입력해주세요.");  
    return false;  
  } else {  
    $(".error_comment").html("");  
    return true;  
  }  
}
```

- 정규 표현식을 사용해 이메일 유효성 확인

- 이메일을 입력하지 않고 확인 버튼 클릭 시, 이메일을 다시 입력해달라는 문구를 화면에 표시하고 false를 반환
- 형식에 맞지 않는다면 형식에 맞게 다시 입력해달라는 문구를 화면에 표시하고 false를 반환
- 형식에 맞는다면 true를 반환하고 ajax를 통해 가입한 이메일을 찾아줌

```

$("#submit_btn").click(function() {
    var val = $("#email").val();
    // 이메일 유효성 체크
    var emailValidity = emailValidate(val);

    if(emailValidity == false) {
        return 0;
    } else {
        $.ajax({
            url: "<%=request.getContextPath()%>/member/find/email",
            type: "post",
            data: {
                email: $("#email").val()
            },
            success: function(result) {
                var email = result.email;

                if(email == null) {
                    console.log("email: " + $("#input_email").val());
                    var email = $("#email").val();
                    var html = '는 유효한 이메일이 아닙니다.';
                    var onclick_val = "location.href='"+ "<%=request.getContextPath()%>/member/register'";

                    $("#result_email").html(email);
                    $("#comment").html(html);

                    $("#login_btn").removeAttr("onclick");
                    $("#login_btn").attr("onclick", onclick_val);
                    $("#login_btn").html("회원가입하기");

                    $("#id_vision").hide();
                    $("#result_content").show();
                } else {
                    console.log("email: " + email);
                    var email = result.email;
                    var html = '회원으로 등록된 이메일 아이디입니다.';
                    html += "<br>해당 이메일로 로그인하고 비밀번호를 이용하세요!";

                    $("#id_vision").hide();
                    $("#result_email").html(email);
                    $("#comment").html(html);
                    $("#result_content").show();
                }
            },
            error: function(request, status, error) {
                console.log(request);
                console.log(status);
                console.log(error);
            }
        });
    }
});

```

- Ajax를 통한 저장된 정보 조회

- 입력한 이메일이 존재한다면 입력한 이메일 정보와 함께 로그인 또는 아이디·비밀번호 찾기 페이지로 이동할 수 있는 버튼이 보여짐
- 입력한 이메일이 존재하지 않는다면 회원가입 버튼이 보여짐

로그인 회원가입

아이디·비밀번호 찾기

아이디 찾기 비밀번호 찾기

가장자릿수 이메일 계정을 입력하시면
비밀번호를 이메일로 발송해드립니다.

```
} else {  
    alert("인증번호가 전송되었습니다.");  
    $.ajax({  
        url: "<%=request.getContextPath()%>/member/authentication",  
        type: "post",  
        data : {  
            email: $("#pwd_email").val()  
        },  
        success : function(authNumber) {  
            var authNumber = authNumber;  
            var html = "";  
            $("#pwd_vision").hide();  
            $("#auth_vision").show();  
  
            html += '<p id="guide">';  
            html += ' 가입하셨던 이메일 계정을 입력하시면, <br>';  
            html += ' 비밀번호를 이메일로 발송해드립니다. ';  
            html += '</p>';  
            html += '<form id="memberInfo_form">';  
            html += ' <div><input type="text" name="auth_number" class="input_email" id="auth_number" placeholder="인증번호">';  
            html += ' <p class="error_comment"></p>';  
            html += ' <div id="submit_btn_wrap">';  
            html += ' <button type="button" id="auth_chk">확인</button> ';  
            html += ' </div>';  
            html += '</form>';  
        }  
    });  
}
```

```
@RequestMapping(value="/authentication", method=RequestMethod.POST, produces="application/text;charset=utf8")  
@ResponseBody  
public String mailCheck(  
    @RequestParam(value="email", required = false) String email) {  
    System.out.println("[이메일 인증요청]");  
    System.out.println("email:\t" + email);  
    return mailService.joinEmail(email);  
}
```

☆ [종당] 웹사이트 비밀번호 인증 안내[웹 제작 테스트용]

보낸사람 <yjk3856@gmail.com>

받는사람 <yjk3856@naver.com>

홈이자를 방문해주셔서 감사합니다.

인증 번호는 [118290]입니다.
해당 인증번호를 인증번호 확인란에 입력해주세요.

- Ajax를 통한 인증번호 전송
 - 사용자가 입력한 이메일은 controller로 전달되며 전달된 이메일로 인증번호를 전송

가입하셨던 이메일 계정을 입력하시면,
비밀번호를 이메일로 발송해드립니다.

118290

확인

```

$("#auth_vision").html(html);
$("#auth_chk").click(function() {
    console.log("email_val: " + email_val);
    var emailValue = email_val;
    var input_num = $("#auth_number").val();

    if(input_num == authNumber) {
        $.ajax({
            url: "%request.getContextPath()%/member/find/password",
            type: "get",
            data: {
                email : email_val
            },
            success: function(result) {
                console.log("[프론트에서 가져온값]");

                if(result.email == null) {
                    alert("이메일이 존재하지 않습니다.");
                    location.href = "%request.getContextPath()%/member/findInfo";
                } else {
                    var html = "";
                    var email = result.email;
                    var password = result.password;
                    const onclick_val = "location.href='%request.getContextPath()%/member/login'";

                    html += "<p id='result_password'>";
                    html += "email: " + email + "의 비밀번호는 ";
                    html += "<span style='color:blue'>" + password + "</span> 입니다.";
                    html += "</p>";
                    html += "<div id='btn_wrap'>";
                    html += "<div><button type='button' class='result_btn', id='login_btn'>로그인 하러가기</button></div>";
                    html += "</div>";

                    $("#auth_vision").hide();
                    $("#result_content").html(html);
                    $("#result_content").show();

                    $("#login_btn").removeAttr("onclick");
                    $("#login_btn").attr("onclick", onclick_val);
                }
            },
            error : function(request, status, error) {
                console.log(request);
                console.log(status);
                console.log(error);
            }
        });
    } else {
        console.log("잘못 입력하셨습니다.");
        alert("인증번호가 일치하지 않습니다.");
    }
});

```

son789@naver.com의 비밀번호는 000입니다.

로그인 하러가기

- Ajax를 통한 인증번호 비교

- 사용자가 입력한 인증번호와 발송한 인증번호를 비교하여 두 값이 불일치한다면 인증번호가 일치하지 않는다는 팝업 문구를 띄움
- 사용자가 입력한 인증번호와 발송한 인증번호를 비교하여 두 값이 일치한다면 해당 이메일의 정보의 비밀번호를 html형식으로 보여줌

```
<!-- Gmail -->
<bean class="org.springframework.mail.javamail.JavaMailSenderImpl" id="mailSender">
    <property name="host" value="smtp.gmail.com"/>
    <property name="port" value="587"/>
    <property name="username" value="${email.account}"/>
    <property name="password" value="${email.password}"/>
    <property name="javaMailProperties">
        <props>
            <prop key="mail.transport.protocol">smtp</prop>
            <prop key="mail.smtp.auth">true</prop>
            <!-- gmail의 경우 보안문제 업데이트로 인해 SSLSocketFactory를 추가해야 smtp 사용 가능. -->
            <prop key="mail.smtp.socketFactory.class">javax.net.ssl.SSLSocketFactory</prop>
            <prop key="mail.smtp.starttls.enable">true</prop>
            <prop key="mail.debug">true</prop>
            <prop key="mail.smtp.ssl.trust">smtp.gmail.com</prop>
            <prop key="mail.smtp.ssl.protocols">TLSv1.2</prop>
        </props>
    </property>
</bean>
```

- makeRandomNumber()는 난수값을 생성해 인증번호를 만들
- joinEmail()은 생성한 인증번호를 포함한 이메일 양식으로 mailSend()에 매개변수 값으로 전달
- mailSend()는 매개변수(발신자, 수신자, 제목, 내용)를 받아 MimeMessage클래스를 통해 html형식으로 전송

```
package kh.spring.fongdang.common;

import java.util.Random;

@Component
@PropertySource("classpath:fongdang.properties")
public class MailSendUtil {

    @Autowired
    private JavaMailSenderImpl mailSender;
    private int authNumber;

    public void makeRandomNumber() {
        Random r = new Random();
        int checkNum = r.nextInt(888888) + 111111;
        System.out.println("-----");
        System.out.println("인증번호:\t" + checkNum + "\n");
        System.out.println("-----\n");
        authNumber = checkNum;
    }


    // 이메일 보낼 양식
    public String joinEmail(String email) {
        makeRandomNumber();
        String setFrom = "${email.account}";
        String toMail = email;
        String title = "[풍당] 웹사이트 비밀번호 인증 안내[원격발행소통]"; // 이메일 제목
        String content = "홈페이지를 방문해주셔서 감사합니다."
            + "<br><br>"
            + "인증 번호는 [" + authNumber + "]입니다."
            + "<br>"
            + "해당 인증번호를 인증번호 확인란에 입력해주세요.";
        mailSend(setFrom, toMail, title, content);
        return Integer.toString(authNumber);
    }

    // 이메일 전송 메소드
    private void mailSend(String setFrom, String toMail, String title, String content) {
        MimeMessage message = mailSender.createMimeMessage();
        //true 매개변수 전달하면 multipart 형식의 메시지 전달이 가능. 문자 인코딩 설정도 가능
        try {
            MimeMessageHelper helper = new MimeMessageHelper(message, true, "utf-8");
            helper.setFrom(setFrom);
            helper.setTo(toMail);
            helper.setSubject(title);
            // true 전달 > html 형식으로 전송, 작성하지 않으면 단순 텍스트로 전달.
            helper.setText(content, true);
            mailSender.send(message);
        } catch (MessagingException e) {
            e.printStackTrace();
        }
    }
}
```

로그인

☒ 아이디 저장 아이디·비밀번호 찾기

로그인

 카카오로 시작하기

 네이버로 시작하기

아직 풍당 계정이 없나요? 회원가입

```
<script>
/* 저장된 쿠키값 불러오기 */
var remember_email = "${member.email}";
if (remember_email != '') {
    $("#chk_id").attr("checked", true);
    $("#email").val(remember_email);
}

function kakaoLogin() {
    var url = 'https://kauth.kakao.com/oauth/authorize?client_id=' + '<spring:eval expression="@property[\'kakao.api_key\']"/>'
        + '&redirect_uri=' + '<spring:eval expression="@property[\'kakao.redirect_uri\']"/>' + '&response_type=code';
    location.href= url;
}

function naverLogin() {
    // ajax를 통해 네이버 로그인 URL을 얻고, 얻은 url로 이동
    $.ajax({
        url: '<%=request.getContextPath()%>/member/login/getNaverAuthUrl',
        type: 'get',
    }).done(function (res) {
        location.href = res;
    });
}
</script>
```


kakao

카카오에일 아이디, 이메일, 전화번호

☒ 로그인 상태 유지

로그인

또는

 QR코드 로그인

회원가입 카카오계정 : 비밀번호 찾기

- 카카오로 시작하기 버튼 클릭 시, 카카오 API가 호출되고 로그인 정보를 입력 받아 문제없이 전달하면 서버는 인증 코드를 발급해 앱에 등록된 redirect uri로 전달함

```

@RequestMapping(value="/login/oauth/kakao", method=RequestMethod.GET)
public String kakaoLogin(Model model)
{
    SnsInfo sns_info
    , Member new_snsUser
    , @RequestParam(value="code", required = false) String code
    , HttpSession session) {
    String msg = "";

    System.out.println("\n[code]\n" + code);
    String access_token = kakaoLoginBO.getAccessToken(code);

    HashMap<String, Object> userInfo = kakaoLoginBO.getUserInfo(access_token);
    if(userInfo == null) {
        msg = "불의장측에 오류 문의해야 소셜 계정을 이용할 수 있습니다. 계속해서 오류 발생시 관리자에게 문의해주세요";
        model.addAttribute("msg", msg);
        return "member/errorPage";
    }
    System.out.println("#####[KAKAO LOGIN]#####");
    System.out.println("\t\t[KAKAO]access_token:" + access_token);
    System.out.println("\t\tnickname:" + userInfo.get("nickname"));
    System.out.println("\t\temail:" + userInfo.get("email") + "\n");

    Gson gson = new GsonBuilder()
        .setPrettyPrinting()
        .create();
    System.out.println("[userInfo]\n" + gson.toJson(userInfo));

    // TODO : 카카오 로그인 세션에 저장 이후 내 정보 구현하기
    String email = (String)userInfo.get("email");
    String nickname = (String)userInfo.get("nickname");
    String name = (String)userInfo.get("nickname");
    String sns_type = "KAKAO";

    sns_info.setSns_email(email);
    sns_info.setSns_name(name);
    sns_info.setSns_nickname(nickname);
    sns_info.setSns_type(sns_type);
    
```

```

public String getAccessToken(String authorize_code) {
    String access_token = "";
    String refresh_token = "";
    String reqURL = "https://kauth.kakao.com/oauth/token";

    try {
        URL url = new URL(reqURL);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        // POST 요청을 위해 기본값이 false인 setDoOutput을 true로
        conn.setRequestMethod("POST");
        conn.setDoOutput(true);
        // POST 요청에 필요로 요구하는 파라미터 스트림을 통해 전송
        BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(conn.getOutputStream()));
        StringBuilder sb = new StringBuilder();
        sb.append("grant_type=authorization_code");

        sb.append("&client_id=" + api_key); //본인이 발급받은 key
        sb.append("&redirect_uri=" + redirect_uri); //본인이 설정한 주소

        sb.append("&code=" + authorize_code);
        bw.write(sb.toString());
        bw.flush();

        // 결과 코드가 200이라면 성공
        int responseCode = conn.getResponseCode();
        System.out.println("responseCode : " + responseCode);

        // 요청을 통해 얻은 JSON타입의 Response 데이터를 읽어오기
        BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));
        String line = "";
        String result = "";

        while ((line = br.readLine()) != null) {
            result += line;
        }
        System.out.println("response body : " + result);

        // Gson 라이브러리에 포함된 클래스로 JSON파싱 객체 생성
        @SuppressWarnings("deprecation")
        JsonParser parser = new JsonParser();
        @SuppressWarnings("deprecation")
        JsonElement element = parser.parse(result);

        access_token = element.getAsJsonObject().get("access_token").getAsString();
        refresh_token = element.getAsJsonObject().get("refresh_token").getAsString();

        System.out.println("access_token : " + access_token);
        System.out.println("refresh_token : " + refresh_token);

        br.close();
        bw.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
    return access_token;
}
    
```

- 카카오 서버에서 받은 code(인가코드)값을 통해 함수getAccessToken()는 접근 토큰 발급을 요청하고 서버는 응답결과로 알맞는 토큰을 전송

```
@RequestMapping(value="/login/oauth/kakao", method=RequestMethod.GET)
public String kakaoLogin(Model model
    , SnsInfo sns_info
    , Member new_SnsUser
    , @RequestParam(value="code", required = false) String code
    , HttpSession session) {
    String msg = "";

    System.out.println("\n[code]\n\t" + code);
    String access_token = kakaoLoginBO.getAccessToken(code);

    HashMap<String, Object> userInfo = kakaoLoginBO.getUserInfo(access_token);
    if(userInfo == null) {
        msg = "동의항목에 모두 동의해야 소셜 계정을 이용할 수 있습니다. 계속해서 오류 발생시 관리자에게 문의해주세요요";
        model.addAttribute("msg", msg);
        return "member/errorPage";
    }
    System.out.println("#####[KAKAO LOGIN]#####");
    System.out.println("\t\t>[KAKAO]access_Token:\t" + access_token);
    System.out.println("\t\t>nickname:\t" + userInfo.get("nickname"));
    System.out.println("\t\t>email:\t" + userInfo.get("email") + "\n");

    Gson gson = new GsonBuilder()
        .setPrettyPrinting()
        .create();
    System.out.println("[userInfo]\n\t"+gson.toJson(userInfo));

    // TODO : 카카오 토큰 세션에 저장 이후 내 정보 구편하기
    String email = (String)userInfo.get("email");
    String nickname = (String)userInfo.get("nickname");
    String name = (String)userInfo.get("nickname");
    String sns_type = "KAKAO";

    sns_info.setSns_email(email);
    sns_info.setSns_name(name);
    sns_info.setSns_nickname(nickname);
    sns_info.setSns_type(sns_type);
}
```

```
public HashMap<String, Object> getUserInfo(String access_token) {
    // 요청하는 클라이언트마다 가진 정보가 다를 수 있기에 HashMap타입으로 선언
    HashMap<String, Object> userInfo = new HashMap<String, Object>();
    String reqURL = "https://kapi.kakao.com/v2/user/me";
    try {
        URL url = new URL(reqURL);
        HttpURLConnection conn = (HttpURLConnection) url.openConnection();
        conn.setRequestMethod("GET");

        // 요청에 필요한 Header에 포함될 내용
        conn.setRequestProperty("Authorization", "Bearer " + access_token);

        int responseCode = conn.getResponseCode();
        System.out.println("responseCode : " + responseCode);

        BufferedReader br = new BufferedReader(new InputStreamReader(conn.getInputStream()));

        String line = "";
        String result = "";

        while ((line = br.readLine()) != null) {
            result += line;
        }
        System.out.println("response body : " + result);

        @SuppressWarnings("deprecation")
        JsonParser parser = new JsonParser();
        @SuppressWarnings("deprecation")
        JsonElement element = parser.parse(result);

        JsonObject properties = element.getAsJsonObject().get("properties").getAsJsonObject();
        JsonObject kakao_account = element.getAsJsonObject().get("kakao_account").getAsJsonObject();

        String nickname = properties.getAsJsonObject().get("nickname").getString();
        String email = kakao_account.getAsJsonObject().get("email").getString();

        userInfo.put("nickname", nickname);
        userInfo.put("email", email);

    } catch (Exception e) {
        return null;
    }
    return userInfo;
}
```

- 발급받은 접근 토큰을 통해 함수 getUserInfo()는 카카오 서버에 사용자 로그인 정보를 요청하고 응답결과로 받은 사용자의 정보를 SnsInfo객체에 저장


```

/** 1) 기존 회원인지 체크 */
Member member = null;
member = member_service.selectMember(email);
System.out.println("\n***** 기존 회원인지 체크 *****");
if(member == null) {
    /** 2) 기존 회원이 아닌 경우 */
    System.out.println("\t=>동당 회원이 아닙니다.\n");
    new_snsUser.setEmail(email);
    new_snsUser.setNickname(nickname);
    new_snsUser.setName(name);

    /** 2-1) 우선 테이블(MEMBER)에 받아오 소셜 정보를 추가 */
    int result1 = member_service.insertMember(new_snsUser);
    if(result1 == 0) {
        System.out.println(">>>>동당 회원 가입 실패\n" + result1);
    } else {
        System.out.println(">>>>동당 회원 가입 성공\n" + result1 + "\n");
    }

    /** 2-2) 소셜 테이블(SNS_INFO)에 소셜정보를 따로 보관 */
    int result2 = sns_service.insertSnsUser(sns_info);
    if(result2 == 0) {
        System.out.println("#####KAKAO 회원정보 입력 실패");
    } else {
        System.out.println("#####KAKAO 회원정보 입력 성공");
    }

    member = member_service.selectMember(email);
} else {
    /** 3) 기존 회원인 경우 */
    System.out.println("\t=>기존에 가입한 동당 회원입니다.\n" + member + "\n");
    System.out.println("#####메인 페이지로 이동합니다.");

}
System.out.println("\n세션에 가져갈 회원정보\n[member]\n\t=" + member + "\n");

session.setAttribute("kakaoToken", access_token); // 페이지에 보낼 소셜 정보가 담긴 세션
session.setAttribute("loginInfo", member); // 페이지에 보낼 회원 정보가 담긴 세션

return "redirect:/";
}

```

```

<insert id="insertMember" parameterType="Member">
<choose>
<when test="name == null and password == null">
    insert into member(
        email, nickname
    ) values(
        #{email}, #{nickname}
    )
</when>
<when test="password == null">
    insert into member(
        email, name, nickname
    ) values(
        #{email}, #{name}, #{nickname}
    )
</when>
<otherwise>
    insert into member(
        email, name, password, nickname
    ) values(
        #{email}, #{name}, #{password}, #{nickname}
    )
</otherwise>
</choose>
</insert>

```

```

<insert id="insertSnsUser" parameterType="Oauth">
<if test="sns_name != null">
    insert into sns_info(
        sns_email,
        sns_name,
        sns_nickname,
        sns_type,
        sns_connect_date
    ) values(
        #{sns_email},
        #{sns_name},
        #{sns_nickname},
        #{sns_type},
        default
    )
</if>
<if test="sns_name == null">
    insert into sns_info(
        sns_email,
        sns_nickname,
        sns_type,
        sns_connect_date
    ) values(
        #{sns_email},
        #{sns_nickname},
        #{sns_type},
        default
    )
</if>
</insert>

```

- 카카오 계정으로 입력한 이메일을 통해 기존에 가입한 회원인지 조회
 - 신규 회원이 아닐 경우 사용자 카카오 로그인 정보가 담긴 SnsInfo객체를 SNS_INFO 테이블에 저장함
 - 신규 회원인 경우 MEMBER 테이블에 사용자의 정보를 저장함

로그인

son789@naver.com

비밀번호

☒ 아이디 저장

아이디·비밀번호 찾기

로그인

카카오로 시작하기

N

네이버로 시작하기

아직 풍당 계정이 없나요? [회원가입](#)

```
<script>
/* 저장된 쿠키값 불러오기 */
var remember_email = "${member.email}";
if (remember_email != '') {
    $("#chk_id").attr("checked", true);
    $("#email").val(remember_email);
}

function kakaoLogin() {
    var url = 'https://kauth.kakao.com/oauth/authorize?client_id=' + '<spring:eval expression="@property[\'kakao.api_key\']"/>'
        + '&redirect_uri=' + '<spring:eval expression="@property[\'kakao.redirect_uri\']"/>' + '&response_type=code';
    location.href= url;
}

function naverLogin() {
    // ajax를 통해 네이버 로그인 URL을 얻고, 얻은 url로 이동
    $.ajax({
        url: '<%=request.getContextPath()%>/member/login/getNaverAuthUrl',
        type: 'get',
    }).done(function (res) {
        location.href = res;
    });
}
</script>
```

NAVER

☒ ID 로그인 ☐ 일회용 번호 ☐ QR코드네이버에 로그인하여 **fongdang** 서비스를
이용하실 수 있습니다.공용 PC에서 사용하시는 경우 보안을 위해 서비스 이용 후
네이버에서도 반드시 로그아웃해 주세요.

아이디

비밀번호

☒ 로그인 상태 유지IP보안 ☒

로그인

[비밀번호 찾기](#) | [아이디 찾기](#) | [회원가입](#)

- 네이버로 시작하기 버튼 클릭 시, ajax의 get방식으로 서버에 네이버 로그인 url값을 요청하고 결과값을 받아 네이버 로그인 페이지로 이동


```

@RequestMapping("/login/getNaverAuthUrl")
@ResponseBody
public String getNaverAuthUrl(HttpSession session) {
    // 네이버 로그인 URL을 반환
    String naverAuthUrl = naverLoginBO.getAuthorizationUrl(session);
    System.out.println("\n=====naverAuthUrl:\t" + naverAuthUrl);

    return naverAuthUrl;
}

@RequestMapping(value="/login/oauth/naver/callback", method=RequestMethod.GET)
public String naverCallback(Model model
    , @RequestParam String code
    , @RequestParam String state
    , SnsInfo sns_info
    , Member new_snsUser
    , HttpSession session) throws IOException, ParseException {
    /* 네이버 인종이 성공적으로 완료되면 code 파라미터가 전달되며 이를 통해 access token을 발급 */
    OAuth2AccessToken oauthToken = naverLoginBO.getAccessToken(session, code, state);
    String access_token = oauthToken.getAccessToken(); // 토큰
    String msg = "";

    // 1. 로그인 사용자 정보를 읽어옴, json구조
    String apiResult = naverLoginBO.getUserProfile(oauthToken);
    if(apiResult == null) {
        msg = "통일장부에 오류 발생하여 소셜 계정을 이용할 수 없습니다. 계속해서 오류 발생시 관리자에게 문의해주세요";
        model.addAttribute("msg", msg);
        return "member/errorPage";
    }

    // 2. String형식인 apiResult를 json형태로 바꿈
    JSONParser parser = new JSONParser();
    Object obj = parser.parse(apiResult);
    JSONObject jsonObj = (JSONObject)obj;

    // 3. 객체화 객상
    JSONObject response_obj = (JSONObject)jsonObj.get("response");
    // response의 nickname 값 파싱

    String nickname = (String)response_obj.get("nickname");
    String email = (String)response_obj.get("email");
    String name = (String)response_obj.get("name");
    String sns_type = "NAVER";

    System.out.println("\n=====NAVER LOGIN=====");
    System.out.println("\t>[NAVER]accessToken:\t" + access_token);
    System.out.println("\t>nickname:\t" + nickname);
    System.out.println("\t>email:\t" + email);
    System.out.println("\t>name:\t" + name + "\n");

    sns_info.setSns_email(email);
    sns_info.setSns_name(name);
    sns_info.setSns_nickname(nickname);
    sns_info.setSns_type(sns_type);
}

```

```

<!-- OAuth2.0 Java OpenSource Library 설정 -->
<dependency>
    <groupId>com.github.scribejava</groupId>
    <artifactId>scribejava-core</artifactId>
    <version>2.8.1</version>
</dependency>

```

```

/* 네아로 인증 URL 생성 Method */
public String getAuthorizationUrl(HttpSession session) {

    /* 세션 유효성 검증을 위하여 난수를 생성 */
    String state = generateRandomString();
    /* 생성한 난수 값을 session에 저장 */
    setSession(session, state);

    /* Scribe에서 제공하는 인증 URL 생성 기능을 이용하여 네아로 인증 URL 생성 */
    OAuth2Service oauthService = new ServiceBuilder()
        .apiKey(CLIENT_ID)
        .apiSecret(CLIENT_SECRET)
        .callback(REDIRECT_URI)
        .state(state) //앞서 생성한 난수값을 인증 URL생성시 사용할
        .build(NaverLoginApi.instance());

    return oauthService.getAuthorizationUrl();
}

```

- OAuth2.0라이브러리인 scribejava 사용하기 위해 pom.xml 라이브러리에 scribejava 등록
- getAuthorizationUrl()는 scribe에서 제공하는 인증 URL을 통해 네이버 로그인 페이지의 url값을 반환

```

@RequestMapping("/login/getNaverAuthUrl")
@ResponseBody
public String getNaverAuthUrl(HttpSession session) {
    // 네이버 로그인 url 반환
    String naverAuthUrl = naverLoginBO.getAuthorizationUrl(session);
    System.out.println("\n#####naverAuthUrl:\t" + naverAuthUrl);

    return naverAuthUrl;
}

@RequestMapping(value="/login/oauth/naver/callback", method=RequestMethod.GET)
public String naverCallback(Model model
    , @RequestParam String code
    , @RequestParam String state
    , SnsInfo sns_info
    , Member new_snsUser
    , HttpSession session) throws IOException, ParseException {
    /* 네이버 로그인 성공 시 네이버 서버로부터 code, state, state를 받아 access token을 발급 */
    OAuth2AccessToken oauthToken = naverLoginBO.getAccessToken(session, code, state);
    String access_token = oauthToken.getAccessToken(); // 토큰
    String msg = "";

    // 1. 로그인 사용자 정보를 받아옴. json구조
    String apiResult = naverLoginBO.getUserProfile(oauthToken);
    if(apiResult == null) {
        msg = "로그인 후에도 로그아웃된 사용자입니다. 계속해서 요청 발생시 관리자에게 문의하십시오";
        model.addAttribute("msg", msg);
        return "member/errorPage";
    }

    // 2. String형식인 apiResult를 json형태로 바꿈
    JSONParser parser = new JSONParser();
    Object obj = parser.parse(apiResult);
    JSONObject jsonObj = (JSONObject)obj;

    // 3. 객체화 과정
    JSONObject response_obj = (JSONObject)jsonObj.get("response");
    // response의 nickname 값 추출

    String nickname = (String)response_obj.get("nickname");
    String email = (String)response_obj.get("email");
    String name = (String)response_obj.get("name");
    String sns_type = "NAVER";

    System.out.println("\n#####[NAVER LOGIN]#####");
    System.out.println("\t>[NAVER]accessToken:\t" + access_token);
    System.out.println("\t>nickname:\t" + nickname);
    System.out.println("\t>email:\t" + email);
    System.out.println("\t>name:\t" + name + "\n");

    sns_info.setSns_email(email);
    sns_info.setSns_name(name);
    sns_info.setSns_nickname(nickname);
    sns_info.setSns_type(sns_type);
}

```

```

/* 네이버 Callback 처리 및 AccessToken 획득 Method */
public OAuth2AccessToken getAccessToken(HttpSession session
    , String code
    , String state) throws IOException{

    /* Callback으로 전달받은 세션검증용 난수값과 세션에 저장되어있는 값이 일치하는지 확인 */
    String sessionState = getSession(session);
    if(StringUtils.pathEquals(sessionState, state)){
        OAuth20Service oauthService = new ServiceBuilder()
            .apiKey(CLIENT_ID)
            .apiSecret(CLIENT_SECRET)
            .callback(REDIRECT_URI)
            .state(state)
            .build(NaverLoginApi.instance());

        /* Scribe에서 제공하는 AccessToken 획득 기능으로 네이버 Access Token을 획득 */
        OAuth2AccessToken accessToken = oauthService.getAccessToken(code);
        return accessToken;
    }
    return null;
}

```

```

/* Access Token을 이용하여 네이버 사용자 프로필 API를 호출 */
public String getUserProfile(OAuth2AccessToken oauthToken) throws IOException{
    OAuth20Service oauthService = new ServiceBuilder()
        .apiKey(CLIENT_ID)
        .apiSecret(CLIENT_SECRET)
        .callback(REDIRECT_URI).build(NaverLoginApi.instance());

    OAuthRequest request = new OAuthRequest(Verb.GET, PROFILE_API_URL, oauthService);
    oauthService.signRequest(oauthToken, request);
    Response response = request.send();

    return response.getBody();
}

```

- 로그인에 성공 시 네이버 서버는 사용자의 로그인 정보를 callback url로 전달
- 네이버 서버에서 받은 callback url의 값들(session, code, state)을 통해 함수 getAccessToken()는 접근 토큰 발급을 요청하고 서버는 응답결과로 알맞는 토큰을 전송
- getUserProfile()은 서버로부터 발급받은 접근 토큰을 통해 사용자의 로그인 정보를 서버에 요청

```

/** 1) 기존 회원인지 체크 */
Member member = null;
member = member_service.selectMember(email);
System.out.println("\n***** 기존 회원인지 체크 *****");
if(member == null) {
    /** 2) 기존 회원이 아닌 경우 */
    System.out.println("\t>플랫폼 회원이 아닙니다.\n");
    new_snsUser.setEmail(email);
    new_snsUser.setNickname(nickname);
    new_snsUser.setName(name);

    /** 2-1) 우선 테이블(MEMBER)에 받아온 소셜 정보를 추가 */
    int result1 = member_service.insertMember(new snsUser);
    if(result1 == 0) {
        System.out.println(">>>>플랫폼 가입 실패\n" + result1);
    } else {
        System.out.println(">>>>플랫폼 회원 가입 성공\n" + result1 + "\n");
    }

    /** 2-2) 소셜 테이블(SNS_INFO)에 소셜정보들을 따로 보관 */
    int result2 = sns_service.insertSnsUser(sns_info);
    if(result2 == 0) {
        System.out.println("#####KAKAO 회원정보 입력 실패");
    } else {
        System.out.println("#####KAKAO 회원정보 입력 성공");
    }

    member = member_service.selectMember(email);
} else {
    /** 3) 기존 회원인 경우 */
    System.out.println(">>기존에 가입한 플랫폼 회원입니다.\n" + member + "\n");
    System.out.println("#####에이인 페이지로 이동합니다.");

}
System.out.println("\n세션에 가져갈 회원정보\n[member]\n\t>" + member + "\n");

session.setAttribute("kakaoToken", access_token); // 페이지에 보낼 소셜 정보가 담긴 세션
session.setAttribute("loginInfo", member); // 페이지에 보낼 회원 정보가 담긴 세션

return "redirect:/" ;
}

```

```

<insert id="insertMember" parameterType="Member">
<choose>
<when test="name == null and password == null">
insert into member(
email, nickname
) values(
#{email}, #{nickname}
)
</when>
<when test="password == null">
insert into member(
email, name, nickname
) values(
#{email}, #{name}, #{nickname}
)
</when>
<otherwise>
insert into member(
email, name, password, nickname
) values(
#{email}, #{name}, #{password}, #{nickname}
)
</otherwise>
</choose>
</insert>

```

```

<insert id="insertSnsUser" parameterType="Oauth">
<if test="sns_name != null">
insert into sns_info(
sns_email,
sns_name,
sns_nickname,
sns_type,
sns_connect_date
) values(
#{sns_email},
#{sns_name},
#{sns_nickname},
#{sns_type},
default
)
</if>
<if test="sns_name == null">
insert into sns_info(
sns_email,
sns_nickname,
sns_type,
sns_connect_date
) values(
#{sns_email},
#{sns_nickname},
#{sns_type},
default
)
</if>
</insert>

```

- 네이버 계정으로 입력한 이메일을 통해 기존에 가입한 회원인지 조회
 - 신규 회원이 아닐 경우 사용자 네이버 로그인 정보가 담긴 SnsInfo객체를 SNS_INFO 테이블에 저장함
 - 신규 회원인 경우 MEMBER 테이블에 사용자의 정보를 저장함

```

/* 메이커 등록 */
@PostMapping("/insert")
public ResponseEntity<String> insertMaker(ModelAndView mv
    , Maker maker
    , HttpServletRequest req
    , HttpSession session
    , RedirectAttributes rtrr
    , MultiPartHttpRequest multipart
    ) {
    if (multipart.getFile("logo_file").getSize() > 0) {
        MultiPartFile makerLogoFile = multipart.getFile("logo_file");
        String logoFile = fileUpload.saveFile(makerLogoFile, req);
        maker.setMaker_license_copy(logoFile);
    }
    if (multipart.getFile("license_copy_file").getSize() > 0) {
        MultiPartFile makerLicenseCopyFile = multipart.getFile("license_copy_file");
        String licenseFile = fileUpload.saveFile(makerLicenseCopyFile, req);
        maker.setMaker_license_copy(licenseFile);
    }
    Member member = (Member) session.getAttribute("loginInfo");
    if (member != null) {
        member.setEmail(member.getEmail());
    }
    try {
        String result = "";
        int i = makerService.insertMaker(maker);
        if (i == 1) {
            result = "success";
        } else {
            result = "fail";
        }
        return new ResponseEntity<>(result, HttpStatus.OK);
    } catch (Exception e) {
        return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_ACCEPTABLE);
    }
}

```

```

/* 메이커 수정등록 */
@PostMapping("/update")
public ResponseEntity<String> updateMaker(ModelAndView mv
    , Maker maker
    , HttpServletRequest req
    , HttpSession session
    , RedirectAttributes rtrr
    , MultiPartHttpRequest multipart
    ) {
    if (multipart.getFile("logo_file").getSize() > 0) {
        MultiPartFile makerLogoFile = multipart.getFile("logo_file");
        String logoFile = fileUpload.saveFile(makerLogoFile, req);
        maker.setMaker_license_copy(logoFile);
    }
    if (multipart.getFile("license_copy_file").getSize() > 0) {
        MultiPartFile makerLicenseCopyFile = multipart.getFile("license_copy_file");
        String licenseFile = fileUpload.saveFile(makerLicenseCopyFile, req);
        maker.setMaker_license_copy(licenseFile);
    }
    Member member = (Member) session.getAttribute("loginInfo");
    if (member != null) {
        member.setEmail(member.getEmail());
    }
    try {
        String result = "";
        int i = makerService.updateMaker(maker);
        if (i == 1) {
            result = "success";
        } else {
            result = "fail";
        }
        return new ResponseEntity<>(result, HttpStatus.OK);
    } catch (Exception e) {
        return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_ACCEPTABLE);
    }
}

```

- Ajax를 통한 입력한 정보 DB에 저장 및 수정
 - 메이커 정보가 DB에 이미 저장되어 있는 경우에는 기존 데이터들을 새로 입력한 값으로 수정으로 해줌
 - 메이커 정보가 DB에 저장되지 않은 경우에는 새로 입력한 값을 DB에 저장함

메이커 정보

메이커(가업)명:

사업자구분:

문해전화번호:

문해 이메일:

주소(사업장 주소):

사업자등록번호:

사업자등록증 사진:

메이커 프로필 이미지:

홈페이지 (선택사항):

카카오채널 (선택사항):

가입하기

```
//API 사업자등록 번호 확인
$("#license_button").click(function () {
    var form = {
        bNo: $("#b_no").val()
    }
    $.ajax({
        type: "POST"
        ,url: "<%=request.getContextPath()%>/maker/licenseCheck"
        ,data : form
        ,dataType:"JSON"
        ,success: function (result) {
            console.log("result : ", result);
            let str = result;
            let strArr = str.data;
            alert(strArr[0].tax_type);
            if (strArr[0].b_stt_cd != '') {
                $("#maker_register_num_yn").val("Y");
            }else{
                $("#maker_register_num_yn").val("N");
            }
        }
        ,error : function (result) {
            console.log(result);
        }
    });
    $("#b_no").on("onchange", function() {
        $("#maker_register_num_yn").val("N");
    });
});
```

```
/* 사업자등록번호 API */
@PostMapping("/licenseCheck")
public ResponseEntity<String> inserMaker(ModelAndView mv, HttpServletRequest req) {

    String url = "https://api.odcloud.kr/api/nts-businessman/v1/"
        + "status/servicekey=yokAhitJw0vyyvU9zRtnifTovmD2218CR57jk85VqArcRRe%2FdbuK2B1AgTt2BNt2FU75Xyn04NukTFd4qE4k5K2FNGRQKJ0K30";
    JSONObject jo = new JSONObject();
    JSONArray jarr = new JSONArray();

    jarr.put(req.getParameter("bNo"));
    jo.put("b_no", jarr);

    String result = apiRequestUtil.requestPost(url, jo.toString());

    try {
        return new ResponseEntity<>(result, HttpStatus.OK);
    } catch (Exception e) {
        return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_ACCEPTABLE);
    }
}
```

- 사업자등록 번호 확인

- 사업자등록번호를 조회하는 오픈 API를 사용하여 입력된 값의 진위여부를 확인하여 팝업 문구로 사용자에게 알려줌



```
<!-- sun start -->
<c:if test="${pick_yn eq 'N'}">
<button id="btn_like" type="button">
좋아요(<span id="pick_cnt">${pick_cnt}</span>)
</button>
</c:if>
<c:if test="${pick_yn eq 'Y'}">
<button id="btn_like" type="button">
싫어요(<span id="pick_cnt">${pick_cnt}</span>)
</button>
</c:if>
<input type="hidden" id="pick_yn" name="pick_yn" value="${pick_yn}" />
<input type="hidden" id="pick_update_yn" name="pick_update_yn" value="${pick_update_yn}" />
<input id="email" type="hidden" value="${loginInfo.email}">
<!-- sun end -->
```

- 펀딩 상세페이지로 이동 시, 좋아요 여부를 mv에 담아서 이동
 - 좋아요를 한 경우에는 좋아요 옆의 사진이 색이 채워진 하트로 표시됨
=> pickYn = 'N'
 - 좋아요를 하지 않은 경우에는 좋아요 옆의 사진이 색이 채워지지 않은 하트로 표시됨
=> pickYn = 'Y'

4일 남음

2022.08.04 종료

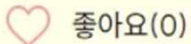
달성률 66% 목표금액 1,000,000원

누적 펀딩액 658,000원

서포터 10명



펀딩하기



종아요(0)



공유하기



판매자에게 문의하기



신고하기

```
$("#btn_like").on('click', function(){
    if($("#email").val() == ''){
        alert('로그인 해주세요.');
```

```
// pick insert
@PostMapping("/insert")
public ResponseEntity<String> insertPick(Pick pick) {
    try {
        pickService.insertPick(pick);
        JSONObject jo = new JSONObject();
        jo.put("pick_yn", "Y");
        jo.put("pick_cnt", pickService.countPick(pick));
        jo.put("pick_update_yn", "Y");
        String result = jo.toString();
        return new ResponseEntity<>(result, HttpStatus.OK);
    } catch (Exception e) {
        return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_ACCEPTABLE);
    }
}
```

- Ajax를 통한 데이터 DB에 저장
 - 종아요 업데이트 여부 값을 확인하여 종아요 값을 저장함
 - email, p_no, pickYn 값을 json 타입으로 변경하여 controller에 전달하고 DB에 저장함
 - => 전달된 값을 DB에 저장하고 종아요 여부, 종아요 수, 종아요 업데이트 여부를 리턴함
 - 종아요 사진이 색이 채워진 하트로 변경되고, 종아요 한 사람 수가 1증가함

4일 남음 2022.08.04 종료
달성률 66% 목표금액 1,000,000원
누적 펀딩액 658,000원
서포터 10명



펀딩하기



좋아요(1)



공유하기



판매자에게 문의하기



신고하기

```

$("#btn_like").on('click', function(){
    if($("#email").val() == ''){
        alert('로그인 해주세요.');
```

 }
 return;

 var urlStr;
 if (\$("#pick_update_yn").val() == 'Y') {
 urlStr = "<%=request.getContextPath()%>/pick/update";
 } else {
 urlStr = "<%=request.getContextPath()%>/pick/insert";
 }

 var pickYn = \$("#pick_yn").val();
 if (pickYn == 'N') {
 pickYn = 'Y';
 } else {
 pickYn = 'N';
 }

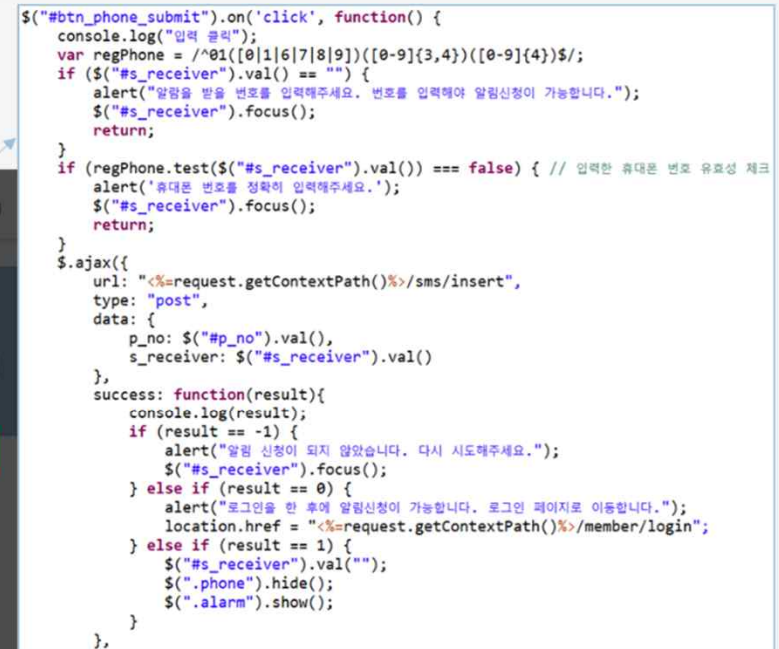
 \$.ajax({
 url: urlStr,
 type: "post",
 data: {
 email: \$("#email").val(),
 p_no: \$("#p_no").val(),
 pick_yn: pickYn
 },
 datatype: "JSON",
 success: function(result){
 console.log(result);
 var data = JSON.parse(result);
 console.log(data);
 if (data.pick_yn == 'Y') {
 \$("#pick_yn").val('Y');
 var src = "<%=request.getContextPath()%>/resources/images/heart_color.png";
 \$("#pick_img").attr('src', src);
 } else {
 var src = "<%=request.getContextPath()%>/resources/images/heart.png";
 \$("#pick_yn").val('N');
 \$("#pick_img").attr('src', src);
 }
 \$("#pick_cnt").text(data.pick_cnt);
 \$("#pick_update_yn").val(data.pick_update_yn);
 },
 error: function(request, status, error) {
 console.log("code:" + request.status + "\n" + "message:" + request.responseText + "\n" + "error:" + error);
 }
 });
});

```

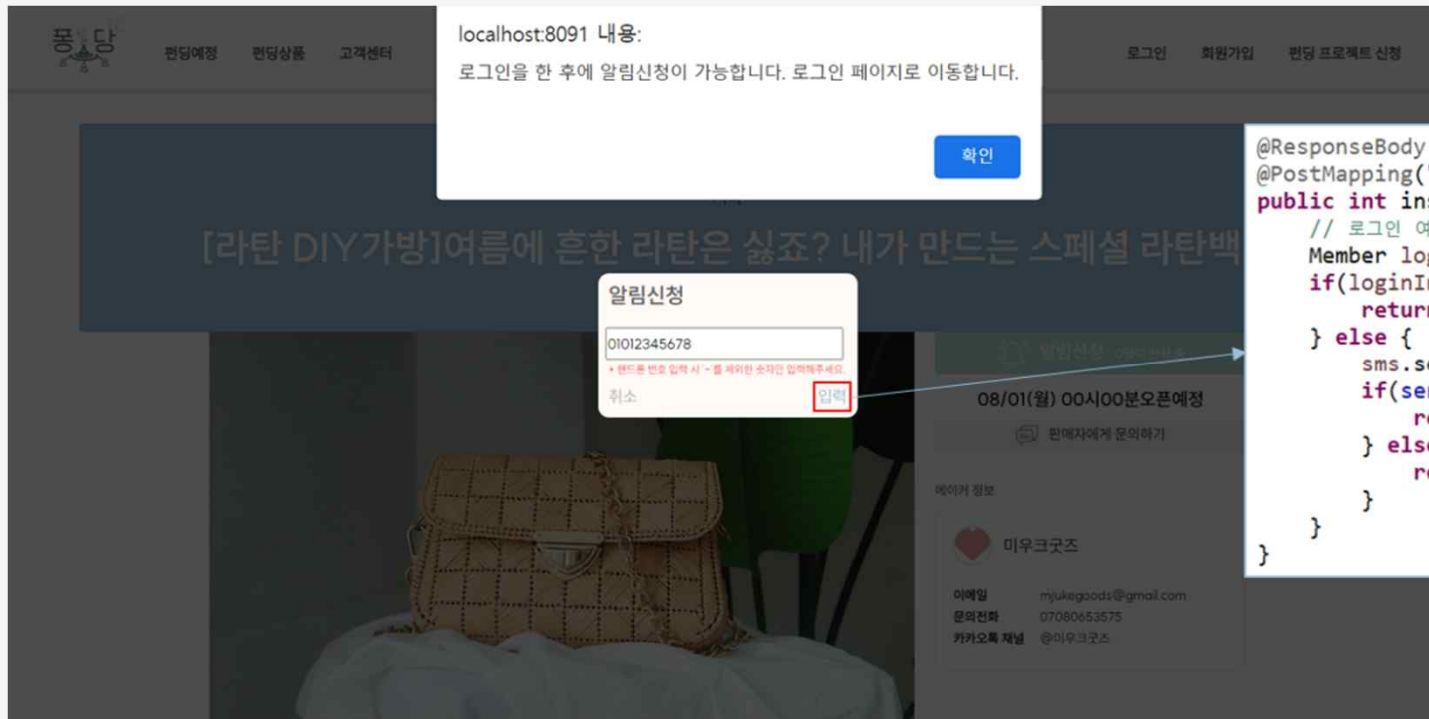
// pick delete
@PostMapping("/update")
public ResponseEntity<String> updatePick(Pick pick){
    try {
        pickService.updatePick(pick);
        JSONObject jo = new JSONObject();
        jo.put("pick_yn", pick.getPick_yn());
        jo.put("pick_cnt", pickService.countPick(pick));
        jo.put("pick_update_yn", "Y");
        String result = jo.toString();
        return new ResponseEntity<>(result, HttpStatus.OK);
    } catch (Exception e) {
        return new ResponseEntity<>(e.getMessage(), HttpStatus.NOT_ACCEPTABLE);
    }
}

```

- Ajax를 통한 데이터 DB에서 수정
 - 좋아요 업데이트 여부 값을 확인하여 좋아요 값을 수정함
 - email, p_no, pickYn 값을 json 타입으로 변경하여 controller에 전달하고 DB에서 수정함
=> 전달된 값을 DB에 저장하고 좋아요 여부, 좋아요 수, 좋아요 업데이트 여부를 리턴함
 - 좋아요 사진이 색이 비워진 하트로 변경되고, 좋아요 한 사람 수가 1감소함

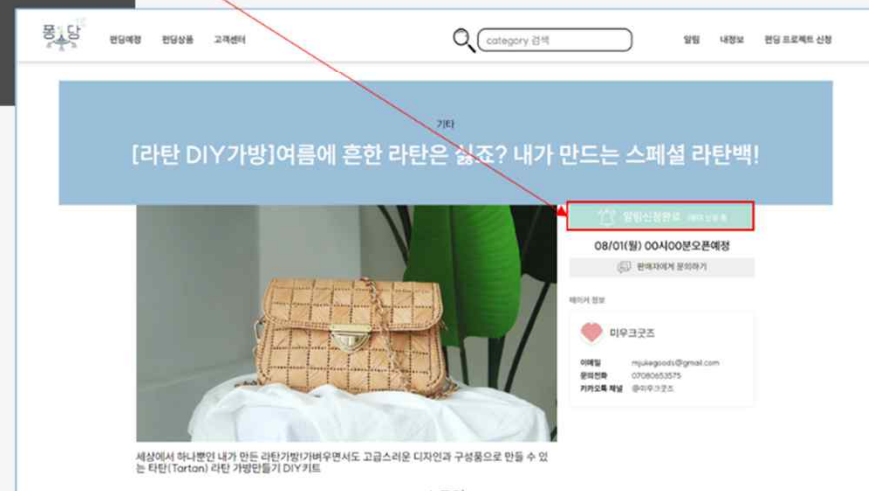
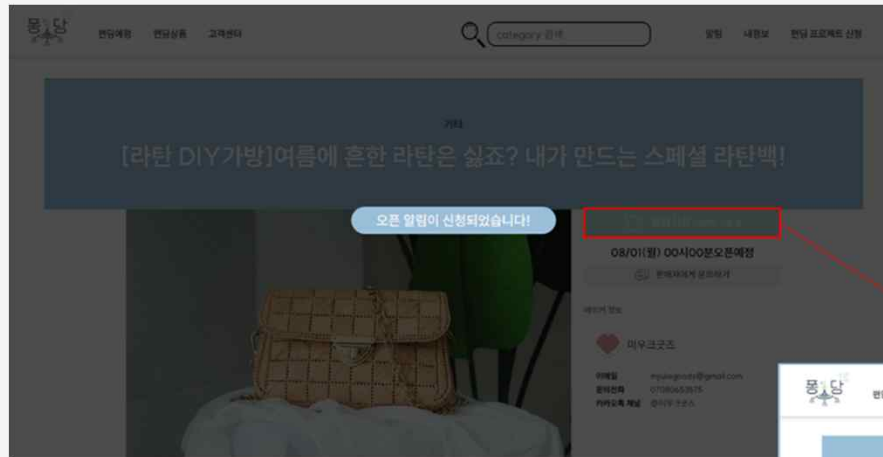


- 정규 표현식을 사용해 전화번호 유효성 확인
 - 전화번호를 입력하지 않고 입력 버튼 클릭 시, 팝업 문구를 띄우고 번호 입력 창에 focus가 맞춰지도록 설정
 - 형식에 맞지 않는다면 팝업 문구를 띄우고 번호 입력 창에 focus가 맞춰지도록 설정
 - 형식에 맞는다면 ajax를 통해 정보를 DB에 저장



```
@ResponseBody
@PostMapping("/insert")
public int insertSms(Sms sms, HttpSession session) {
    // 로그인 여부 확인
    Member loginInfo = (Member)session.getAttribute("loginInfo");
    if(loginInfo == null) { // 로그아웃 상태일 때
        return 0;
    } else {
        sms.setEmail(loginInfo.getEmail());
        if(service.insertSms(sms) < 1) {
            return -1;
        } else {
            return 1;
        }
    }
}
```

- Ajax를 통한 입력한 정보 DB에 저장
 - 로그인하지 않고 알림 신청을 할 경우 로그인이 필요하다는 팝업 문구를 띄우고, 로그인페이지로 이동
=> controller에서 session정보 확인을 통해 로그인 여부를 확인하고 0을 리턴함
 - 로그인을 했고 정보가 DB에 저장 됐을 경우 알림 신청이 되었다는 모달창을 띄움
=> controller에서 session정보 확인을 통해 로그인 여부를 확인하고 1을 리턴함



- DB에 저장 성공 후, 모달창이 뜨고 이를 닫으면 해당 펀딩 상품의 알림 신청자 수가 1 증가함

```

$("#btn_funding_alarm_cancel").on('click', function(){
    console.log("알림신청완료 클릭");
    $.ajax({
        url: "<%=request.getContextPath()%>/sms/delete",
        type: "post",
        data: {
            s_no: $("#s_no").val(),
        },
        success: function(result){
            console.log(result);
            if (result == -1) {
                alert("알림신청 취소가 되지 않았습니다. 다시 시도해주세요.");
                $("#s_receiver").focus();
            } else if (result == 0) {
                alert("로그인을 한 후에 알림신청 취소가 가능합니다. 로그인 페이지로 이동합니다.");
                location.href = "<%=request.getContextPath()%>/member/login";
            } else if (result == 1) {
                $(".cancel_alarm").show();
            }
        },
        error: function(request, status, error) {
            console.log("code:" + request.status + "\n" + "message:" + request.responseText + "\n" + "error:" + error);
        }
    });
});

```

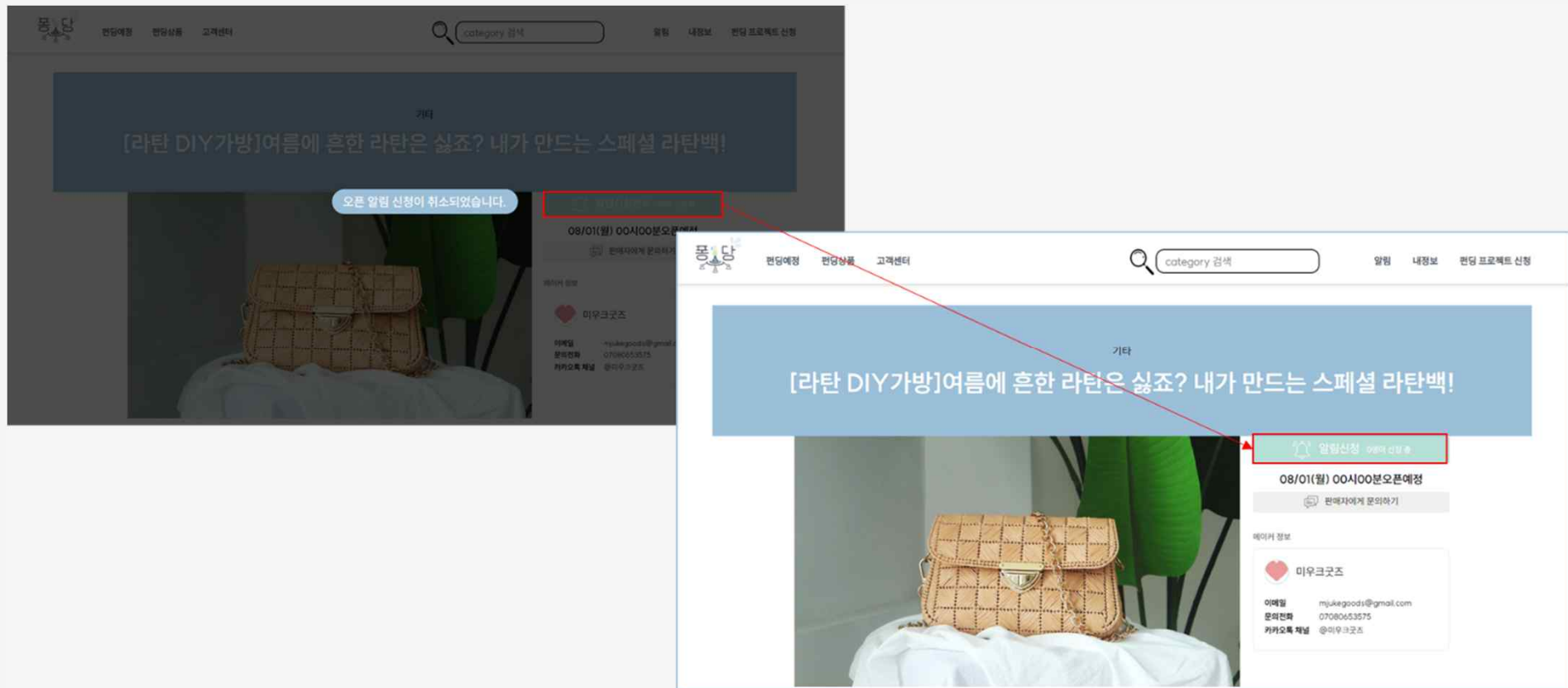
```

@ResponseBody
@PostMapping("/delete")
public int deleteSms(int s_no, HttpSession session) {
    // 로그인 여부 확인
    Member loginInfo = (Member)session.getAttribute("loginInfo");
    if(loginInfo == null) { // 로그아웃 상태일 때
        return 0;
    } else {
        if(service.deleteSms(s_no) < 1) {
            return -1;
        } else {
            return 1;
        }
    }
}

```

- Ajax를 통한 데이터 DB에서 삭제

- 로그인하지 않고 알림 신청을 취소할 경우 로그인이 필요하다는 팝업 문구를 띄우고, 로그인페이지로 이동
=> controller에서 session정보 확인을 통해 로그인 여부를 확인하고 0을 리턴함
- 로그인을 했고 정보가 DB에서 삭제 됐을 경우 알림 신청이 취소되었다는 모달창을 띄움
=> controller에서 session정보 확인을 통해 로그인 여부를 확인하고 1을 리턴함



- DB에서 삭제 성공 후, 모달창이 뜨고 이를 닫으면 해당 펀딩 상품의 알림 신청자 수가 1 감소함

```

@Service
@PropertySource("classpath:fongdang.properties")
public class SmsServiceImpl implements SmsService{
    @Value("${sms.api_key}")
    private String api_key;

    @Value("${sms.api_secret}")
    private String api_secret;

    @Value("${sms.from_num}")
    private String from_num;

```

```

@Override
@Scheduled(cron = "0 0 0/1 * * *") //1시간 마다 실행
public void sendMessage() {
    Message coolsms = new Message(api_key, api_secret);

```

```

    List<Sms> smsList = dao.selectSmsList();
    for(Sms sendList : smsList) {
        HashMap<String, String> params = new HashMap<String, String>();
        params.put("to", sendList.getS_receiver()); // 수신자 번호
        params.put("from", from_num); // 송신자 번호
        params.put("type", "LMS"); // 메시지 타입
        params.put("text", "알림 신청하신 <" + sendList.getP_name() + ">프로젝트 오픈 1시간 전입니다."); // 문자 메시지 내용

        try {
            coolsms.send(params); // 메시지 전송
        } catch (CoolsmsException e) {
            e.printStackTrace();
        }
    }
}

```

```

<select id="selectSmsList" resultType="Sms">
    SELECT s_receiver, p_name
    FROM sms JOIN product USING(p_no)
    WHERE TO_CHAR(SYSDATE, 'YYYY/MM/DD HH24') = TO_CHAR(s_time, 'YYYY/MM/DD HH24')
</select>

```

- 스프링 스케줄러를 사용해 입력한 번호로 알림 신청한 프로젝트 오픈 1시간 전에 오픈 안내 문자 발송
 - properties 파일에 미리 입력해 놓은 api_key, api_secret, from_num(송신자 번호) 값을 Value 어노테이션을 사용해 가져옴
 - Scheduled 어노테이션에 cron 표현식을 사용해 sendMessage()(오픈 안내 문자 발송 메소드)가 1시간마다 실행되도록 설정
 - SmsDao에서 문자를 보낼 목록을 뽑아와 입력한 번호에 안내 문구를 발송
 - => 테이블 저장된 문자를 보낼 시간의 시와 현재 시간의 시를 비교해 두 값이 일치하는 목록을 조회

localhost:8091 내용:
jpg, jpeg, png 파일만 업로드 할수 있습니다. 파일 확인 후 다시 시도해 주세요.

확인

신고사유 선택: 허위사실

신고 내용 입력

dd

호주날씨예측.jpynb

파일 첨부

• 동일 상을 신고는 불가능합니다.
• 증명자료는 png, jpg, jpeg 파일만 업로드 가능합니다.
• 신고 내용이 사실과 다르거나 허위인 경우, 이용 제재를 받을 수 있습니다.
• 신고자의 정보 및 신고 내용은 안전하게 보호되며 외부에 제공되지 않습니다.

취소 신고

```

$("#btn_report_send").on('click', function(){
    console.log("신고하기 클릭");
    if ($("#report_content").val() == "") {
        alert("신고 내용을 입력해주세요. 내용을 입력해야 리포트 등록할 수 있습니다.");
        $("#report_content").focus();
        return;
    }
    if ($("#uploadfile").val()) { // 파일이 있다면
        var ext = $("#uploadfile").val().split('.').pop().toLowerCase(); //확장자 분리하여 소문자로 변경
        if ($.inArray(ext, ['jpg', 'jpeg', 'png']) == -1) { //확장자가 있는지 체크
            alert('jpg, jpeg, png 파일만 업로드 할수 있습니다. 파일 확인 후 다시 시도해주세요.');
            return;
        }
    }
    var form = $("#report_form")[0]; //form에 작성된 모든 것 가져오기
    var formData = new FormData(form);

    $.ajax({
        url: "%request.getContextPath()%/report/insert",
        type: "post",
        processData: false, //데이터 객체를 문자열로 바꿀지에 대한 값 true=일반문자/ false=데이터객체
        contentType: false, //일반 text인지 구분하는 값 true이면 일반 텍스트로 구분
        data: formData,
        success: function(result){
            console.log(result);
            if (result == -1) {
                alert("신고가 되지 않았습니다. 다시 시도해주세요.");
                $("#report_content").focus();
            } else if (result == 0) {
                alert("로그인을 한 후에 신고가 가능합니다. 로그인 페이지로 이동합니다.");
            }
        }
    });
}

```

- 파일 첨부 시, 파일의 확장자 확인

- 첨부한 파일이름에서 확장자만 분리하여 소문자로 변경 후 jpg, jpeg, png 중에 확장자가 있는 확인
- 허용된 확장자가 아니라면 팝업 문구를 띄움
- 허용된 확장자라면 ajax를 통해 입력한 정보를 DB에 저장



- Ajax를 통한 입력한 정보 DB에 저장
 - 로그인하지 않고 신고를 할 경우 로그인이 필요하다는 팝업 문구를 띄우고, 로그인페이지로 이동
=> controller에서 session정보 확인을 통해 로그인 여부를 확인하고 0을 리턴함

The screenshot displays a web application interface for submitting a report. A modal window shows a confirmation message: "localhost:8091 내용: 상품을 신고했습니다." (Content: Reported the product). The background shows a product page for a lamp with a price of 658,000 KRW. A report form is overlaid, featuring a dropdown for "신고사유 선택" (Select reason for report) set to "허위사실" (False facts), a text area for "신고 내용 입력" (Enter report content) with the text "허위 사실로 기재로 신고합니다." (Reporting as false facts), and a file upload section with a "heart.png" file and a "파일 첨부" (Attach file) button. A red box highlights the "신고" (Report) button. Below the form, a list of rules for reporting is provided.

- 동일 상품 신고는 불가능합니다.
- 증명자료는 png, jpg, jpeg 파일만 업로드 가능합니다.
- 신고 내용이 사실과 다르거나 허위인 경우, 이용 제재를 받을 수 있습니다.
- 신고자의 정보 및 신고 내용은 안전하게 보호되며 외부에 제공되지 않습니다.

Below the form, the Java controller code for the report submission is shown:

```
@ResponseBody
@PostMapping("/insert")
public int insertReport(Report report, @RequestParam(name = "uploadfile", required = false) MultipartFile file, HttpServletRequest req, HttpSession session) {
    // 로그인 여부 확인
    Member loginInfo = (Member)session.getAttribute("loginInfo");
    if(loginInfo == null) { // 로그인 안함
        return 0;
    } else {
        report.setEmail(loginInfo.getEmail());

        if(service.checkReport(report) > 0) { // 신고 갯수 확인
            return 2;
        } else {
            if(!file.isEmpty()) { // 파일 존재 여부 확인
                String report_file = commonfile.saveFile(file, req);
                if(report_file != null) { //저장에 성공하면 실행
                    report.setReport_file(report_file);
                }
            }

            if(service.insertReport(report) < 1) {
                return -1;
            } else {
                return 1;
            }
        }
    }
}
```

- Ajax를 통한 입력한 정보 DB에 저장

• 로그인을 했고 정보가 DB에 저장 됐을 경우 신고가 되었다는 팝업 문구를 띄움

=> controller에서 session정보 확인을 통해 로그인 여부를 확인하고 신고 여부를 확인한 다음 파일 존재여부를 확인해 파일을 업로드 한 후, 1을 리턴함

The screenshot displays a web application interface for filing a complaint. A modal window titled '신고하기' (Report) is open, showing a form with a dropdown for '신고사유 선택' (Select reason for report), a text area for '신고 내용 입력' (Enter report content), and a '파일 첨부' (Attach file) button. A red box highlights the '신고' (Report) button. A tooltip above the form indicates 'localhost:8091 내용: 이미 신고한 상품입니다.' (Content: Product already reported).

Below the form, a list of rules is provided:

- 동일 상품을 신고는 불가능합니다.
- 증명자료는 png, jpg, jpeg 파일만 업로드 가능합니다.
- 신고 내용의 사실과 다르거나 허위인 경우, 이용 제재를 받을 수 있습니다.
- 신고자의 정보 및 신고 내용은 안전하게 보호되며 외부에 제공되지 않습니다.

On the right, a code block shows the backend controller logic for the report:

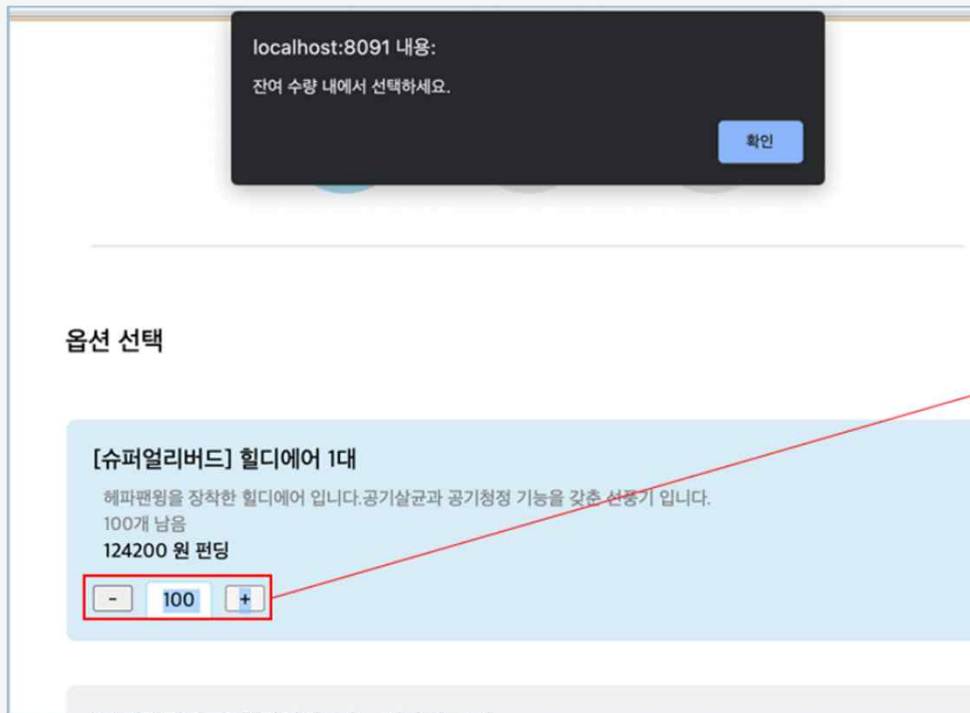
```
@ResponseBody
@PostMapping("/insert")
public int insertReport(@RequestParam(name = "uploadfile", required = false) MultipartFile file, HttpServletRequest req, HttpSession session) {
    // 로그인 여부 확인
    Member loginInfo = (Member)session.getAttribute("loginInfo");
    if(loginInfo == null) { // 로그인 상태일 때
        return 0;
    } else {
        report.setEmail(loginInfo.getEmail());

        if(service.checkReport(report) > 0) { // 신고 갯수 확인
            return 2;
        } else {
            if(!file.isEmpty()) { // 파일 존재 여부 확인
                String report_file = commonfile.saveFile(file, req);
                if(report_file != null) { //저장에 성공하면 실행
                    report.setReport_file(report_file);
                }
            }

            if(service.insertReport(report) < 1) {
                return -1;
            } else {
                return 1;
            }
        }
    }
}
```

- Ajax를 통한 입력한 정보 DB에 저장

- 로그인을 했고 이미 신고한 상품이라면 이미 신고했다는 팝업 문구를 띄움
=> controller에서 session정보 확인을 통해 로그인 여부를 확인하고 신고 여부를 확인하여 2를 리턴함



```

423
424 function fnCalCount(type, no){
425
426     var $input = $("input[name='pop_out']");
427     var tCount = Number($input.val());
428     var tEqCount = $("input[name='limit']").val();
429     var ss = $("input[name='optionNo']").val();
430     if(ss == no){
431         if(type=='p'){
432             if(tCount < tEqCount){
433                 $input.val(Number(tCount)+1);
434             }else{
435                 alert("잔여 수량 내에서 선택하세요.")
436             }
437         }
438     }else{
439         if(tCount > 0) $input.val(Number(tCount)-1);
440     }
441     }else{
442         console.log("failed")
443     }
444 }
445
    
```

- 구매할 수량을 증가시킴

- 증가시킨 수량과 옵션의 제한 수량을 비교
 - => 증가시킨 수량이 제한 수량보다 적을 경우, 화면에 보이는 구매 수량이 1 증가함
 - => 증가시킨 수량이 제한 수량보다 많을 경우, 수량을 초과했다는 팝업 문구를 띄움

[얼리버드] 헐디에어 1대 + 헤파팬wing 2개

헐디에어 1대+헤파팬wing 2개
200개 남은
182700 원 펀딩

- 0 +

[선풍기 200년만의 혁신, 공기청정 공기살균되는 선풍기]에 펀딩합니다.
배송 예정일은 2022-08-15 입니다.

next step

```

689 function order(){
690     const data={
691         order_no :createOrderNo(),
692         p_no : $("input[name='p_no']").val(),
693         option_no : '3',//$("#").val(),
694         total_price : $("input[name='option_price']").val() *5,//$("#amount").val(),//5
695         amount : '5',//$("#amount").val(),
696         payment_plan :$("#plan").val()
697     }
698     if(!data.option_no){
699         swal("옵션을 선택해주세요.");
700         return;
701     }
702     if(!data.amount){
703         swal("수량은 0 이상이어야 합니다.");
704         return;
705     }
706     orderNext(data)
707 }
708 function orderNext(data) {
709     $.ajax({
710         type: "post",
711         url: "<%=request.getContextPath()%>/order/insert",
712         data: data,
713         success: function (result) {
714             console.log(result);
715             if(result == 0){
716                 alert("로그인을 한 후에 주문이 가능합니다. 로그인 페이지로 이동합니다.");
717                 location.href = "<%=request.getContextPath()%>/member/login";
718             }else if(result == -1){
719                 alert("주문실패")
720             }else if(result == 1){
721                 alert("결제 페이지로 이동합니다.");
722                 pay(data.order_no);
723             }
724         },
725         error : function (result) {
726             alert("에러 발생.");
727             console.log(result);
728         }
729     })
730 }

```

- Ajax를 통한 데이터 DB에 저장

- 로그인하지 않고 주문을 할 경우 로그인이 필요하다는 팝업 문구를 띄우고, 로그인페이지로 이동
=> controller에서 session정보 확인을 통해 로그인 여부를 확인하고 0을 리턴함
- 로그인을 했고 정보가 DB에 저장 됐을 경우 결제페이지로 이동하는 pay() 메소드 호출
=> controller에서 session정보 확인을 통해 로그인 여부를 확인하고 1을 리턴함



```
function paymentCard(data){
    IMP.init('imp04215728');
    IMP.request_pay({
        pg : 'html5_inicis',
        pay_method : 'card',
        merchant_uid : data.pay_no,
        name : $("input[name='p_name']").val(), //결제창에서 보여질 이름
        amount : data.total_sum, //실제 결제되는 가격
        buyer_email : data.email,
        buyer_name : $("input[name='supporter1']").val(),
        buyer_tel : data.phone,
        buyer_addr : data.order_address,
        buyer_postcode : $("input[name='memberaddr1']").val(),
    }, function(rsp) {
        console.log(rsp);
        if ( rsp.success ) {
            var msg = '결제가 완료되었습니다.';
            data.impUid = rsp.imp_uid;
            data.merchant_uid = rsp.merchant_uid;
            paymentComplete(data);
        } else {
            var msg = '결제에 실패하였습니다.';
            msg += '에러내용 : ' + rsp.error_msg;
        }
    });
}
```

```
function paymentComplete(data){
    $.ajax({
        url: "<=>request.getContextPath()>/pay/insert",
        method: "POST",
        data: data,
    })
    .done(function(result){
        messageSend();
        swal({
            text: result,
            closeOnClickOutside :false
        })
        .then(function(){
            location.replace("/");
        })
    })
    .fail(function(){
        alert("에러");
        location.replace("/")
    })
}
```

- 결제 성공 시 토큰을 통해 실제 결제되어야하는 가격과 결제된 가격 매치, 다를 경우 결제 취소

감사합니다.