

5조

FORNUAL

전문가를 위한 우리들의 쉼터



목차





서론

구성 ~~~

BACK

JAVA

SpringFramework

Security

SocialLogin

세미 발표때까지 사용하였던 JAVA와 SpringFramework
에 더해

Security에 필요한 소스 코드들

/

SocialLogin 기능을 활용하여 토큰을 얻어오고,
로그인 + 회원 가입을 할 수 있는 기능의 추가

FRONT

JavaScript (ECMA6)

BOOTSTRAP

Security

API

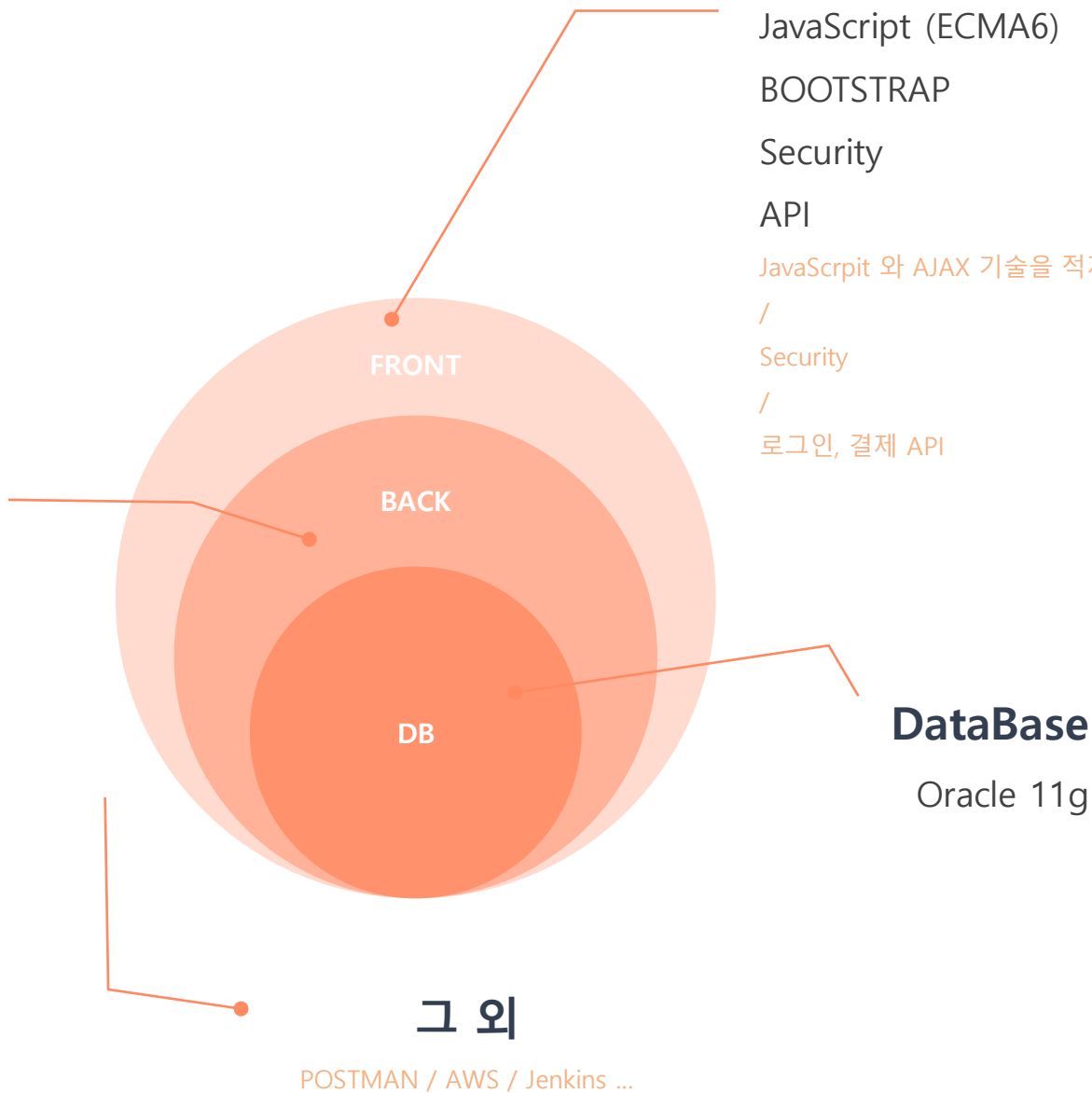
JavaScript 와 AJAX 기술을 적재적소에 혼합하여 사용

/

Security

/

로그인, 결제 API





본론

추가된 페이지 주요 기술



AWS

1

인스턴스 (1/4) 정보

 인스턴스를 속성 또는 (case-sensitive) 태그로 찾기

	Name ▾	인스턴스 ID	인스턴스 상태 ▾	인스턴스 유형 ▾	상태 검사
<input type="checkbox"/>	My WebServer	i-018464e7b457e730b	⊖ 중지됨	t2.micro	-
<input checked="" type="checkbox"/>	Web	i-04005ada1db950c4c	✔ 실행 중	t2.micro	✔ 2/2개 검사 통과...

2

▼ 인스턴스 세부 정보 정보

플랫폼

☒ Ubuntu(추론)

플랫폼 세부 정보

☒ Linux/UNIX

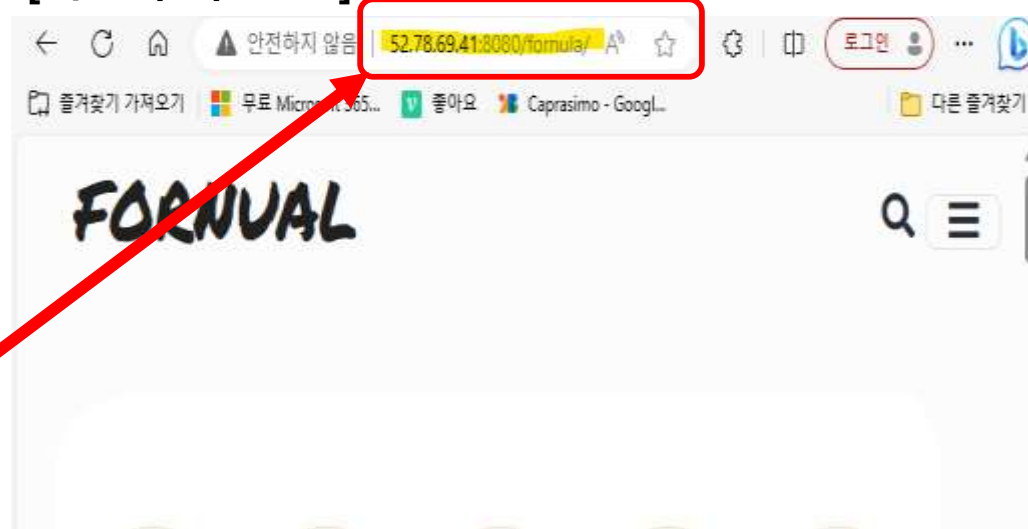
3

IP 버전 ▾	유형 ▾	프로토콜 ▾	포트 범위 ▾	소스 ▾
IPv4	SSH	TCP	22	0.0.0.0/0
IPv4	사용자 지정 TCP	TCP	8080	0.0.0.0/0
IPv4	HTTPS	TCP	443	0.0.0.0/0
IPv4	HTTP	TCP	80	0.0.0.0/0

주요 기능

1. AWS EC2를 이용한 인스턴스 생성
2. Ubuntu 플랫폼 이용
3. 보안그룹 설정 - 모든 접근 허용
4. .war 파일 배포

[배포시 화면 url]

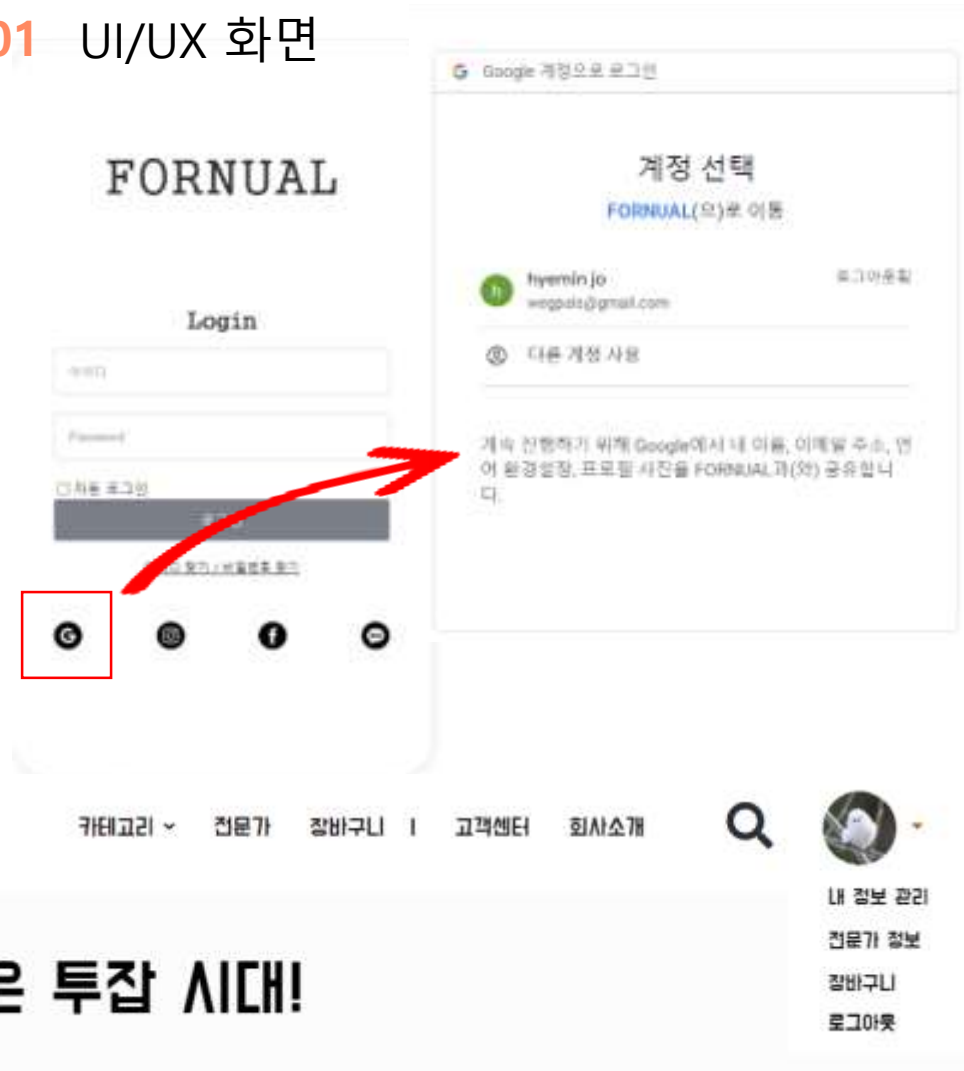




CONTENTS

구글 소셜 로그인

01 UI/UX 화면



주요 기능

1. '구글' 아이콘 클릭시 구글 계정 로그인 화면으로 이동
2. 메인 헤더 '로그인 회원' 상태로 변경

큰 투잡 시대!



회원가입

01 UI/UX 화면

주요 기능

1. RestController 중복 확인
2. Back/Front 정규표현식 검증
3. Valid 적용

1

```
public class MemberJoinRestController {
    private final MemberJoinService memberJoinService;

    private boolean isValid(String id) {
        return id.matches("[a-zA-Z0-9]{6,20}$");
    }

    @PostMapping("/idcheck")
    public Message idCheck(@RequestBody IdCheckForm form) {
        if (StringUtils.isEmpty(form.getId())) {
            return new Message("아이디를 입력해 주세요.");
        }

        Log.info("Id 중복검사 id = {}", form.getId());
        if (!isValid(form.getId())) {
            return new Message("아이디는 6-20자 영문 숫자 조합이어야 합니다.");
        }

        Member idCheck = memberJoinService.idCheck(form.getId());
        Log.info("idCheck isNull? = {}", idCheck);
        if (ObjectUtils.isEmpty(idCheck)) {
            return new Message("사용가능한 아이디입니다.");
        }

        return new Message("사용 불가능한 아이디입니다.");
    }
}
```

2

```
@NotEmpty(message = "")
@Pattern(regexp = "^(?=.*[A-Za-z])(?=.*[0-9])(?=.*[!@#$%^&*])[a-zA-Z0-9!@#$%^&]{8,20}$",
message = "8-20자의 영문, 숫자, 특수문자를 모두 포함한 비밀번호를 입력해 주세요")
private String password;
```

3

```
// 회원가입
@PostMapping("/join")
public String joinForm(@ModelAttribute @Valid Member member, Errors errors,
    if(errors.hasErrors()) {
        return "join";
    }

    memberSecurityService.addSecurityMember(member);
    memberSecurityService.addAuth(auth);
    redirectAttributes.addFlashAttribute("message", "회원가입 성공");
    return "redirect:/";
}
```



구글 소셜 로그인

01

```
// 로그인 관련 api 정보 제공하는 클래스
public class GoogleLoginApi extends DefaultApi20 {
    //싱글톤 사용
    public GoogleLoginApi() {
        // TODO Auto-generated constructor stub
    }
    private static class InstanceHolder{
        private static final GoogleLoginApi INSTANCE = new GoogleLoginApi();
    }
    public static GoogleLoginApi instance() {
        return InstanceHolder.INSTANCE;
    }

    //사용자 접근 토큰 제공을 위한 api의 url 주소 반환하는 메소드
    @Override
    public String getAccessTokenEndpoint() {
        return "https://accounts.google.com/o/oauth2/token";
    }

    //로그인 처리를 위한 API의 URL 주소를 반환하는 메소드
    @Override
    protected String getAuthorizationBaseUrl() {
        return "https://accounts.google.com/o/oauth2/v2/auth";
    }
}
```

02

```
@Component
//구글 로그인을 위한 빈 등록
public class GoogleLoginBean {
    private static final String GOOGLE_CLIENT_ID = "746123676134-e1542ag1abucm152bt5n6s2l3q9s0reh.apps.googleusercontent.com";
    private static final String GOOGLE_CLIENT_SECRET = "GOCSPX-9ZofM3Y1C4PyhE9H39xxQy6B516";
    private static final String GOOGLE_REDIRECT_URI = "http://localhost:9003/google/callback";
    private static final String GOOGLE_SESSION_STATE = "googleState";
    // 토큰으로 가져올 값의 범위 - 이메일, id, email 가져옴
    private final static String GOOGLE_SCOPE = "email openid profile";
    // 프로필 조회 API URL
    private final static String GOOGLE_PROFILE_API_URL = "https://www.googleapis.com/oauth2/v2/userinfo";
}
```

주요 코드

1. 로그인 관련 API 정보를 제공하는 클래스 작성
2. 구글 로그인을 위한 빈 등록
3. 접근 토큰을 획득

03

```
// 구글 callback 처리 및 AccessToken 획득 메소드
public OAuth2AccessToken getAccessToken(HttpSession session, String code, String state) throws IOException {

    // 클라이언트로 전달받은 세션검증용 난수값과 세션에 저장되어 있는 값이 일치하는지 확인
    String sessionState = getSession(session);

    if (StringUtils.pathEquals(sessionState, state)) {
        // 사용자가 접근 토큰을 받기위해 저장된 oauth2IService 객체 생성
        OAuth20Service oAuth20Service = new ServiceBuilder()
            .apiKey(GOOGLE_CLIENT_ID)
            .apiSecret(GOOGLE_CLIENT_SECRET)
            .callback(GOOGLE_REDIRECT_URI).state(state)
            .scope(GOOGLE_SCOPE)
            .build(GoogleLoginApi.instance());

        OAuth2AccessToken accessToken = oAuth20Service.getAccessToken(code);
        return accessToken;
    }

    return null;
}
```




구글 소셜 로그인

04

```
// 구글 로그인 성공시 Callback URL 페이지를 처리하기 위한 컨트롤러 메소드
@RequestMapping("/callback")
public String login(@RequestParam String code, @RequestParam String state, HttpSession session, Model model)
    throws IOException, ParseException {
    OAuth2AccessToken accessToken = googleLoginBean.getAccessToken(session, code, state);

    // 사용자 접근 토큰을 이용해 로그인 사용자의 정보를 반환 객체인 'accessToken'을 사용하여 사용자 정보를 apiResult에 저장
    String apiResult = googleLoginBean.getUserProfile(accessToken);
    log.info("apiResult:{}", apiResult);

    // JSONparser객체를 이용해 문자열을 json 객체로 변환
    JSONParser parser = new JSONParser();
    Object object = parser.parse(apiResult);
    JSONObject responseObject = (JSONObject) object;

    String id = (String) responseObject.get("id");
    String email = (String) responseObject.get("email");

    log.info("id:{}", id);
    System.out.println("aaaaa");

    // 반환받은 구글 사용자 정보를 이용해 'auth' 객체를 생성하여 'auth' 객체를 'authList'에 저장
    Auth auth = new Auth();
    auth.setId("google_" + id);
    auth.setRole("ROLE_MEMBER");
    log.info("auth:{}", auth);

    // authList에 'auth' 객체를 저장한 'auth' 객체를 추가
    List<Auth> authList = new ArrayList<Auth>();
    authList.add(auth);

    Member member = new Member();
    member.setId("google_" + id);
    member.setPassword(UUID.randomUUID().toString());
    member.setEmail(email);
    member.setEnabled("1");
    member.setMemberStatus(1);
    member.setCategoryOne(1);
    member.setCategoryTwo(1);
    member.setCategoryThree(1);

    member.setMemberAuthList(authList);

    // 데이터베이스에 'authList'에 저장된 'id' 값을 찾아서 비교
    List<Member> findSecurityMemberById = memberSecurityService.findSecurityMemberById(member.getId());
    Member loginMember = findSecurityMemberById.stream().findAny().orElse(null);

    if(loginMember == null) {
        memberSecurityService.addAuth(auth);
        memberSecurityService.addSecurityMember(member);
    }

    // 구글 로그인 성공시 정보를 이용하여 UserDetails 객체(로그인 사용자)를 생성하여 저장
    CustomMemberDetails customMemberDetails = new CustomMemberDetails(loginMember);

    Authentication authentication = new UsernamePasswordAuthenticationToken(customMemberDetails, null, customMemberDetails.getAuthorities());
    SecurityContextHolder.getContext().setAuthentication(authentication);

    session.setAttribute(SessionConst.LOGIN_MEMBER, customMemberDetails);

    log.info("loginMember = {}", customMemberDetails.getMemberStatus());

    return "redirect:/";
}
```

주요 코드

4. 구글 로그인 성공시, Callback URL 페이지 요청

- 받아온 토큰을 이용해 사용자의 정보를 저장
- JSON 객체로 변환
- 반환받은 정보를로 권한을 바꿔줌
- 해당 로그인한 사용자의 id 와 DB의 id를 비교하여 회원정보가 없을 경우에 권한과 사용자의 정보를 insert하는 명령 실행
- 구글 로그인 사용자 정보를 사용하여 UserDetails 객체(로그인 사용자)를 생성하여 저장



전문가 등록

```

1  @PreAuthorize("hasRole('ROLE_MEMBER')")
   @PostMapping("/expert-join")
   public String join(@Valid @ModelAttribute("expert") Expert expert, Errors errors,
   @RequestParam("uploadFile") MultipartFile uploadFile, Model model,
   HttpSession session, RedirectAttributes redirectAttributes)
   throws IllegalStateException, IOException, ExistsExpertException {

   if(errors.hasErrors()) {
       model.addAttribute("expert", expert);
       Log.info("errors :{}", errors);
       return "expert-join";
   }

   Log.info("expert:{}", expert);
   Log.info("file:{}", uploadFile);

   //회원가입 처리
   CustomMemberDetails loginMember = (CustomMemberDetails) session.getAttribute(SessionConst.Login_Member);
   Member member = memberSecurityService.getSecurityMember(loginMember.getId());
   expert.setMemberIdx(member.getMemberIdx());
   Log.info("MemberIdx:{}", member.getMemberIdx());

3  // 업로드한 파일이 pdf 파일이 아닌 경우
   if (!uploadFile.isEmpty() && !uploadFile.getContentType().equals("application/pdf")) {
       Log.info("File:{}", uploadFile);
       model.addAttribute("message", "pdf 파일만 업로드해주세요.");
       return "expert-join";
   }

   if(!uploadFile.isEmpty()) {
       String uploadDirectory = context.getServletContext().getRealPath("/resources/upload");
       Log.info("filepath =" + uploadDirectory);

       String expertfileName = extracted(uploadFile);
       Log.info("filename =" + expertfileName);

       expert.setExpertfileName(expertfileName);
   }

   expertJoinService.addExpertInfo(expert);

   // 등록처리에 성공하였을때 memberIdx 변경
   Auth auth = new Auth();
   auth.setId(member.getId());
   auth.setRole("ROLE_EXPERT");
   Log.info("auth:{}", auth);

4  expertJoinService.updateExpertStatus(auth);
   expertJoinService.updateStatus(member.getMemberIdx());

```

주요 코드

1. @PreAuthorize 어노테이션을 이용하여 권한이 'ROLE_MEMBER'인 회원만 접근을 허용
2. Validation 적용
3. MultipartFile 업로드 처리
4. 등록 처리가 성공하면 해당 사용자의 auth 권한에 'ROLE_EXPERT'를 INSERT하는 서비스와 MEMBER 테이블의 MEMBERIDX를 2(=EXPERT)로 UPDATE하는 서비스를 호출

```


2  @NotEmpty(message = "전화번호를 반드시 입력해주세요.")
   @Pattern(regexp="^\d{3}-\d{3,4}-\d{4}$", message="전화번호 형식에 맞게 입력해주세요.")
   private String phone;

```



전문가 랭킹

판매금액 랭킹




16736100원

전문가 ID: qkrqkds95

종목어

안녕하세요 박하윤입니다 안녕하세요 박하...




12522000원

전문가 ID: sgwonbe212

증권

자기소개를 작성하지 않은 전문가입니다



20000원

전문가 ID: sgwonbe9854

개장 세무

안녕

주요 기능

```
<select id="selectTotalMoneyList" resultType="ExpertMoneyRanking">
  SELECT *
  FROM (
    SELECT
      ROWNUM AS rn,
      COALESCE(totalMoney, 0) AS totalMoney,
      e.EXPERT_IDX,
      e.INTEREST,
      e.INTRODUCE,
      m.ID
    FROM
      EXPERT e
    LEFT JOIN
      (
        SELECT
          i.EXPERT_IDX,
          SUM(i.PRICE) AS totalMoney
        FROM
          SALES s
        JOIN
          ITEM i ON s.ITEM_IDX = i.ITEM_IDX
        GROUP BY
          i.EXPERT_IDX
      ) t ON e.EXPERT_IDX = t.EXPERT_IDX
    LEFT JOIN
      MEMBER m ON e.MEMBER_IDX = m.MEMBER_IDX
    WHERE totalMoney != 0
    ORDER BY
      COALESCE(totalMoney, 0) DESC
  )
  WHERE rn BETWEEN 1 AND 3
</select>
```

1. EXPERT,SALES,MEMBER 테이블을 outer join하여 가상의 컬럼 totalPrice를 생성 및 조회
이 때, COALESCE 함수를 사용하여 판매 금액이 0인 전문가들은 랭킹에서 제외
2. 필수 항목이 아닌 자기소개는 전문가의 상태에 따라 JavaScript를 사용하여 메시지 출력
3. JavaScript를 사용하여 ID, 자기소개를 클릭 시 해당 전문가의 포트폴리오 항목으로 페이지 이동

전문가 랭킹

전체 전문가

전문가 번호: 1

전문가 ID: sgwonbe212

경력: 3년 차

자기소개: 안녕하세요

영어

전문가 번호: 3

전문가 ID: asd123

경력: 1년 차

자기소개:

안녕하세요! "보고서"라고 하면 보통 보고서라고 생각하는데, 저는 보고서 작성에 관심이 있습니다. "보고서"라는 용어에 대해 궁금한 점이 있으신가요? 궁금하시면 언제든지 물어봐주세요. 감사합니다.

그레픽

전문가 번호: 11

전문가 ID: qwe123

경력: 7년 차

자기소개:

안녕하세요! 저는 7년 차의 전문가입니다. 주로 프론트엔드 개발을 담당하고 있습니다. 다양한 프로젝트를 경험하고 있으며, 팀워크와 커뮤니케이션에 중점을 둡니다. 궁금한 점이 있으시면 언제든지 물어봐주세요. 감사합니다.

한국어

전문가 번호: 13

전문가 ID: qkrqkbs95

경력: 4년 차

자기소개:

안녕하세요! 저는 4년 차의 전문가입니다. 주로 백엔드 개발을 담당하고 있습니다. 다양한 프로젝트를 경험하고 있으며, 팀워크와 커뮤니케이션에 중점을 둡니다. 궁금한 점이 있으시면 언제든지 물어봐주세요. 감사합니다.

한국어

전문가 번호: 14

전문가 ID: 789ssy

경력: 3년 차

자기소개:

안녕하세요! 저는 3년 차의 전문가입니다. 주로 프론트엔드 개발을 담당하고 있습니다. 다양한 프로젝트를 경험하고 있으며, 팀워크와 커뮤니케이션에 중점을 둡니다. 궁금한 점이 있으시면 언제든지 물어봐주세요. 감사합니다.

SNS호부

전문가 번호: 15

전문가 ID: sgwonbe985d

경력: 1년 차

자기소개:

안녕

개인정보

주요 기능

```
<select id="selectExpertList" resultType="ExpertMoneyRanking">
  SELECT
    e.*,
    m.ID
  FROM (
    SELECT
      rownum AS rn,
      expert_idx,
      member_idx,
      phone,
      interest,
      introduce,
      career,
      company_one,
      company_two,
      company_three
    FROM (
      SELECT *
      FROM expert
      ORDER BY expert_idx
    ) subquery
  ) e
  LEFT JOIN MEMBER m ON e.member_idx = m.member_idx
  WHERE rn BETWEEN #{startRow} AND #{endRow}
</select>
```

1. MEMBER 테이블, EXPERT 테이블을 JOIN하여 전문가 등록 순서대로 전문가 페이지를 나열
2. 등록된 전문가의 숫자만큼 페이징 처리 및 전문가 번호, 전문가 ID 클릭 시 포트폴리오 화면으로 페이지 이동
3. 전문가 관심사 항목 클릭 시 해당 항목의 검색결과출력



CONTENTS

전문가 정보 수정

전문가 정보

전문가 번호

13

전화번호

01012331332

자기소개

안녕하세요 박하윤입니다
안녕하세요 박하윤입니다
안녕하세요 박하윤입니다
안녕하세요 박하윤입니다
안녕하세요 박하윤입니다

직무 및 연차

중국어



4년

경력 사항

Vina

Viettel

ALS

포트폴리오

38452f44-38c0-41c5-8feb-b22504c96488_processfolio.pdf

파일 선택 선택된 파일 없음

[PDF 파일로 업로드 해주세요]

수정완료

주요 기능

1. 전문가의 포트폴리오 파일인 PDF 파일을 올리는 과정 수정
2. PDF 파일을 올릴 때 파일의 이름에 UUID를 자동으로 적용하여 서버 컴퓨터의 portfolio 경로에 자동 저장하는 메소드 추가

```
@PostMapping("/input")
public String modifyExpert(@ModelAttribute Expert expert, Model model,
    @RequestParam MultipartFile uploadFile) throws IOException {
    log.info("Modifying expert information for expertIdx: {}", expert.getExpertIdx());

    /*
     * if (errors.hasErrors()) { model.addAttribute("originalExpert", expert);
     * log.info("errors :{}", errors); return "expert-input"; }
     */

    if (!uploadFile.isEmpty() && !uploadFile.getContentType().equals("application/pdf")) {
        log.info("file:{}", uploadFile);
        model.addAttribute("message", "pdf 파일만 업로드해주세요.");
        return "expert-input";
    }

    if (!uploadFile.isEmpty()) {
        String uploadDirectory = context.getServletContext().getRealPath("/resources/images/portfolio");
        log.info("filepath =" + uploadDirectory);

        String expertfileName = UUID.randomUUID().toString() + "_" + uploadFile.getOriginalFilename();
        log.info("filename =" + expertfileName);

        expert.setExpertfileName(expertfileName);

        File file=new File(uploadDirectory, expertfileName);

        uploadFile.transferTo(file);
        model.addAttribute("uploadDirectory", uploadDirectory);
        model.addAttribute("expertfileName", expertfileName);
    }
}
```




전문가 - 판매관리

1

```
<!-- 판매내역 출력-->
<select id="selectSalesList" resultType="SaleItemExpert">
    SELECT * FROM(SELECT rownum rn, saleslist.* FROM
        (SELECT S.sales_idx,
            S.sales_date,
            S.sales_status,
            I.item_idx,
            I.item_name,
            I.price,
            I.item_content,
            P.itemfile_name,
            ST.status
        FROM item I
        LEFT JOIN sales S ON I.item_idx = S.item_idx
        LEFT JOIN photo P ON I.item_idx = P.item_idx
        LEFT JOIN status ST ON S.sales_status = ST.status_idx
        WHERE I.expert_idx = #{expertIdx} AND I.item_status=1 AND S.sales_idx is not null
        ORDER BY S.sales_date desc
        ) saleslist
    WHERE rn BETWEEN #{startRow} AND #{endRow}
</select>
```

2

```
@Slf4j
@RestController
@RequestMapping("/expert")
@RequiredArgsConstructor
public class ExpertSalesRestController {
    private final ExpertSalesService expertSalesService;

    @PreAuthorize("hasRole('ROLE_EXPERT')")
    @PutMapping("/sales/update")
    public String modifyStatus(@RequestBody Sales sales)
```

3

```
    Log.info("idx:{0}", sales.getSalesIdx());
    Log.info("status:{0}", sales.getSalesStatus());
    Log.info("sales:{0}", sales);
```

```
    expertSalesService.modifySalesStatus(sales);
    expertSalesService.modifyPurchaseStatus(sales);
```

주요 코드

1. 판매 내역 출력을 위해 4개의 테이블 조인 (ITEM, SALES, PHOTO, STATUS 테이블 JOIN)
2. Rest 방식을 이용하여 프론트와 백 분리 구매와 판매 상태를 변경하는 서비스 호출
3. 시큐리티를 적용하여 'ROLE_EXPERT'만 접근 허용
4. 버튼을 클릭 시, 변경에 성공하면 상태에 따라 TEXT를 변경하고 속성을 추가

4

```
.ajax({
    url: "${pageContext.request.contextPath}/expert/sales/update",
    contentType: "application/json",
    type: "PUT",
    data: JSON.stringify({ "salesIdx": salesIdx, "salesStatus": salesStatus }),
    dataType: "text",
    success: function (result) {
        if (result === "success") {
            if (salesStatus == 3) {
                //alert("주문진 상태를 변경하였습니다.");
                currentStatus.text("제작완료"); // #Btn1의 text 변경
                currentStatus.closest('.post-card').find('#statusCheck').text('제작중');
                currentStatus.data('salesstatus', salesStatus); // salesstatus 값을 업데이트
            } else if (salesStatus == 4) {
                // alert("주문중 상태를 변경하였습니다.");
                currentStatus.prop("disabled", true);
                currentStatus.css({ "color": "gray" });
                currentStatus.closest('.post-card').find('#statusCheck').text('제작완료');
            }
        }
    }
})
```



CONTENTS

상품 관리

상품 관리

상품수정



2023-09-05 22:08:33 / ₩ 10000

SNS 홍보 마케팅 전문 업체 "파라미디어"

상품수정



2023-09-05 22:04:11 / ₩ 150000

곧지 아픈 개인 세무. 전문가에게 맡기세요!

상품수정



2023-09-05 21:58:58 / ₩ 2000000

각 나라의 문화를 이해하라!

주요 기능

1. Session에서 사용자의 정보를 가져올 때 Security를 적용
2. 등록일이 빠른 순서로 상품들을 1 페이지당 3개씩 나열하도록 설정
3. 상품 수정 버튼 클릭 시 상품 수정 폼으로 이동
4. 상품 페이지 클릭 시 해당 상품의 상세페이지로 이동

```
@GetMapping("/board")
public String getBoardList(@RequestParam(defaultValue = "1") int pageNum,
                           HttpSession session,
                           Model model) {
    String originalFileName;
    int pos;

    CustomMemberDetails loginMember = (CustomMemberDetails) session.getAttribute(SessionConst.LOGIN_MEMBER);
    Expert expert = itemDetailService.findByMemberId(loginMember.getMemberId());
    int expertId = expert.getExpertId();
    log.info("expertId={}", expertId);

    Map<String, Object> resultMap=expertBoardService.getBoardList(pageNum, expertId);
    log.info("list={}", resultMap);

    List<ItemPhotoForExpert> resultList=(List<ItemPhotoForExpert>) resultMap.get("boardList");
    log.info("resultList={}", resultList);

    for(ItemPhotoForExpert itemBoardList : resultList) {
        pos=itemBoardList.getItemfileName().lastIndexOf("_");
        originalFileName=itemBoardList.getItemfileName().substring(pos+1);
        itemBoardList.setItemfileName(originalFileName);
    }

    log.info("pager={}", resultMap.get("pager"));
    model.addAttribute("boardList", resultMap.get("boardList"));
    model.addAttribute("pager", resultMap.get("pager"));
    log.info("boardList={}", resultList);
    return "expert-list";
}
```



CONTENTS

상품 관리 中 상품 수정

주요 기능





사이트 이용시 주의사항

1. 구매자는 판매자에게 2번의 수정 요청을 제안할 수 있다
 - 1-1. 판매자는 이를 성실히 응답해야 할 의무가 있다
 - 1-2. 구매자와의 의견조율에 있어 불성실한 태도를 보이거나, 구매자와의 의견조율이 불가능한 상황이 온다면 구매자는 구매 취소를 선택할 수 있다
2. 상품의 상태를 제작시작 상태로 바꾼 순간부터 판매자는 구매자와의 의사 소통이 가능해지며 제작기간이 길어지거나 주문이 누락될 경우 구매자는 구매 취소 / 환불이 가능하다
3. 판매자가 제품을 구매자에게 전달한 이후 일정기간이 지나면 판매한 상품의 상태는 자동으로 구매확정이 이루어진다

제1조(목적) 이 규칙은 「전자상거래 등에서의 소비자보호에 관한 법률」 및 같은 법 시행령에서 위임된 사항과 그 시행에 필요한 사항을 규정함을 목적으로 한다.

[전문개정 2012. 8. 17.]

제2조(통신판매에 관한 정보의 제공방법 등) 「전자상거래 등에서의 소비자보호에 관한 법률」(이하 “법”이라 한다) 제2조제2호 본문에서 “총리령으로 정하는 방법”이란 다음 각 호의 방법을 말한다.

1. 광고물 · 광고시설물 · 전단지 · 방송 · 신문 및 잡지 등을 이용하는 방법
2. 판매자와 직접 대면하지 아니하고 우편환 · 우편대체 · 지로 및 계좌이체 등을 이용하는 방법

전자상거래법 보기 ->

☐ 동의 시 상품등록이 가능합니다

상품등록

주요 기능

1. JavaScript를 통하여 전자 상거래법을 볼 수 있는 파일로의 이동 기능 추가
2. 로그인한 사용자의 정보를 @PreAuthorize 어노테이션을 사용하여 상품 등록 폼으로 전달

```
<script type="text/javascript">
function toggleSubmitButton(checkbox) {
    var submitButton = document.getElementById('checkSubmit');
    submitButton.disabled = !checkbox.checked;
}
</script>
```

```
@PreAuthorize("hasRole('ROLE_EXPERT')")
@PostMapping("/add/{expertIdx}")
@Transactional(rollbackFor = Exception.class)
public String insert(@ModelAttribute @Valid ItemForm itemForm, Errors errors, @PathVariable Integer expertIdx,
    Model model, RedirectAttributes redirectAttributes) {
    //log.info("itemForm", itemForm);
```



CONTENTS

상품 등록

상품등록

규정을 준수하여 상품을 등록해주세요

0

1000 이상의 값이어야 합니다.

상품제목을 입력해주세요

상품 이름은 무조건 입력하세요

상품설명

관심사

대 카테고리 ▼ 중 카테고리 ▼

상품 사진 등록하러 가기

주요 기능

1. DTO에서정규표현식 검증
2. Validation 적용
3. 에러 메시지 출력 / Front에서의 검증

1

```
private int expertIdx;  
private int categoryIdx;  
  
@NotEmpty(message="상품 이름은 무조건 입력하세요")  
@Size(max = 30, message = "30 글자 이상 입력할 수 없습니다.")  
private String itemName;  
  
private String itemContent;  
  
@Min(value = 1000, message = "1000 이상의 값이어야 합니다.")  
private int price;  
  
}
```

2

```
if (errors.hasErrors()) {  
    System.out.println(errors.getErrorCount());  
    log.info("Validation errors: {}", errors);  
    return "item-add";  
}
```

3

```
<script type="text/javascript">  
    $("#itemaddForm").submit(function() {  
        var submitResult = true;  
        $(".error").hide();  
  
        if($("#itemName").value()==""){  
            $("#itemNameMsg").html("상품 제목을 입력해 주세요");  
            submitResult = false;  
        }  
        $(".error").show();  
  
        if($("#price").value()==""){  
            $("#priceMsg").html("상품 가격을 입력해 주세요");  
            submitResult = false;  
        }  
        $(".error").show();  
  
        return submitResult = true;  
    });  
</script>
```

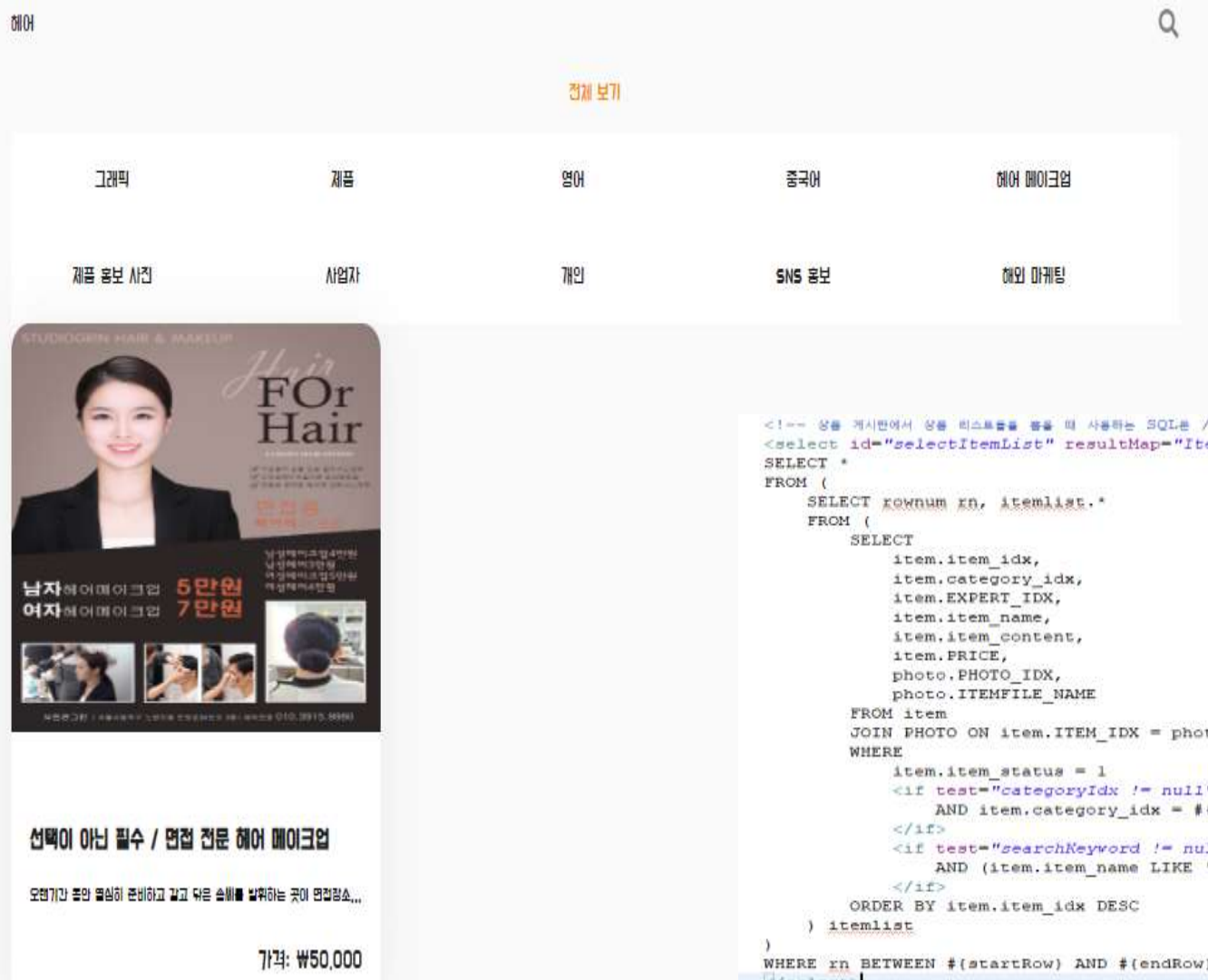


CONTENTS

검색 기능 (header, 상품 게시판)

주요 기능

Header에서의 검색 기능 /
상품 게시판에서의 검색 기능에서
사용이 가능하고, 데이터 손실을 최소화
하기 위하여
상품 게시판 출력 Mapper에서 If 태그를
사용하여 데이터 효율성 증가



```
<!-- 상품 게시판에서 상품 리스트를 출력 할 때 사용하는 SQL문 / categoryId를 입력 받으면 해당 categoryId가 포함된 상품만 출력되도록 변경 -->
<select id="selectItemList" resultMap="ItemPhotoCategoryCart">
SELECT *
FROM (
    SELECT rownum rn, itemlist.*
    FROM (
        SELECT
            item.item_idx,
            item.category_idx,
            item.EXPERT_IDX,
            item.item_name,
            item.item_content,
            item.PRICE,
            photo.PHOTO_IDX,
            photo.ITEMFILE_NAME
        FROM item
        JOIN PHOTO ON item.ITEM_IDX = photo.ITEM_IDX
        WHERE
            item.item_status = 1
            <if test="categoryId != null">
                AND item.category_idx = #{categoryId}
            </if>
            <if test="searchKeyword != null">
                AND (item.item_name LIKE '%' || #{searchKeyword} || '%' OR item.item_content LIKE '%' || #{searchKeyword} || '%')
            </if>
        ORDER BY item.item_idx DESC
    ) itemlist
    )
WHERE rn BETWEEN #{startRow} AND #{endRow}
</select>
```



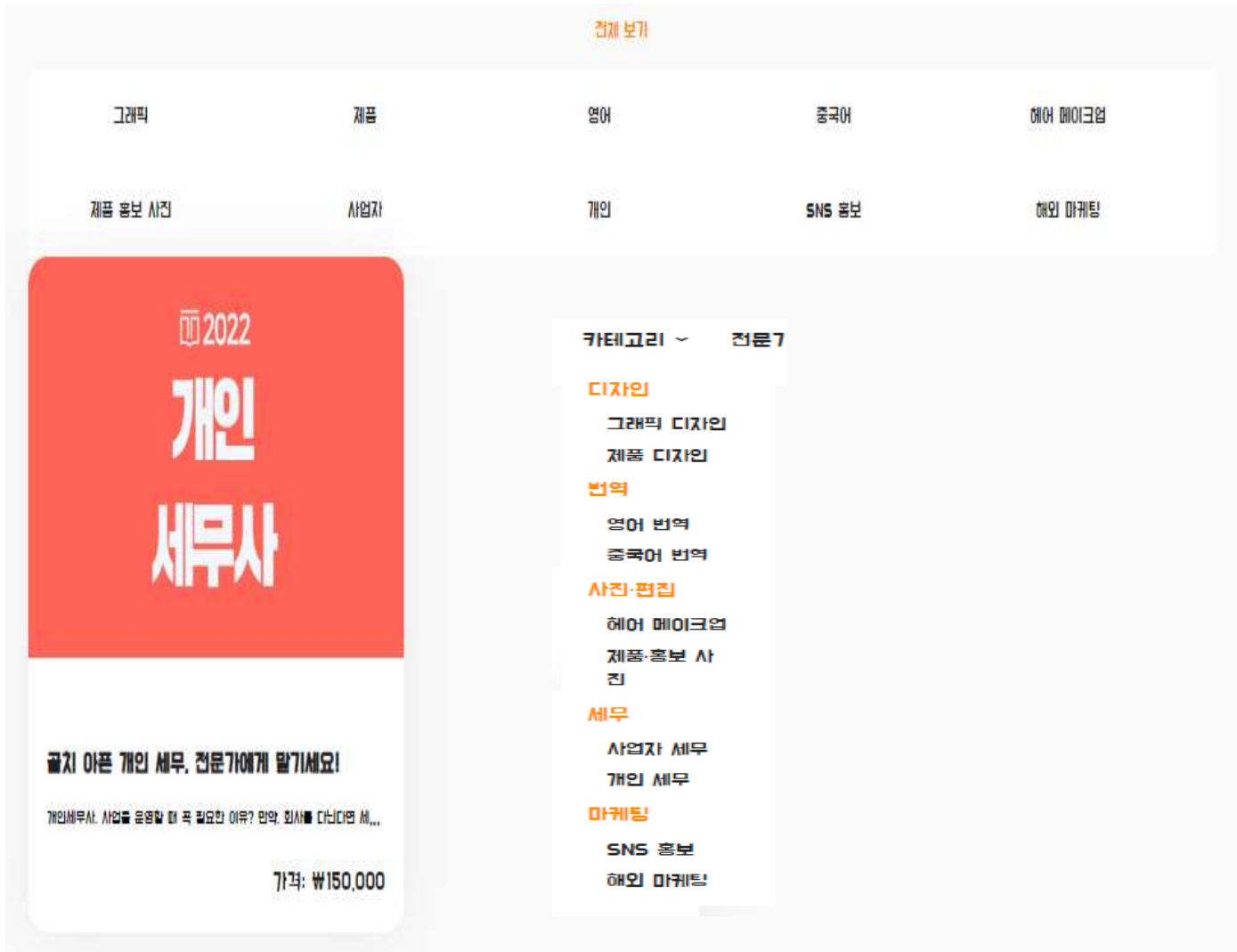
CONTENTS

카테고리별 출력 기능 (header, 상품 게시판)

주요 기능

Header에서의 카테고리별 출력 기능 / 상품 게시판에서의 카테고리별 출력 기능에서 사용이 가능하며, 데이터 손실을 최소화 하기 위하여 상품 게시판 출력 Mapper에서 If 태그를 사용하여 데이터 효율성 증가

```
item.item_status = 1
<if test="categoryIdx != null">
    AND item.category_idx = #{categoryIdx}
</if>
<if test="searchKeyword != null">
    AND item.search_keyword = #{searchKeyword}
</if>
</search>
<:when test="${category == 1}">그래픽</c:when>
<:when test="${category == 2}">제품</c:when>
<:when test="${category == 3}">영어</c:when>
<:when test="${category == 4}">중국어</c:when>
<:when test="${category == 5}">헤어 메이크업</c:when>
<:when test="${category == 6}">제품 홍보 사진</c:when>
<:when test="${category == 7}">사업자</c:when>
<:when test="${category == 8}">개인</c:when>
<:when test="${category == 9}">SNS 홍보</c:when>
<:when test="${category == 10}">해외 마케팅</c:when>
</search>
```



상품 상세페이지 - 카카오톡 공유하기

01 UI/UX 화면

1



주요 기능

1. 카카오톡 공유하기 기능 추가

```
var itemIdx = ${itemIdx};

// SDK를 초기화 합니다. 서문할 일의 JavaScript 키를 설정해 주세요.
Kakao.init('b8e41eee0cc466dfb4360alb5f92e9a7');

// SDK 초기화 여부를 판단합니다.
console.log(Kakao.isInitialized());

function kakaoShare() {
  Kakao.Link.sendDefault({
    objectType : 'feed',
    content : {
      title : 'FORNUAL',
      description : '전문가 외주 플랫폼 "FORNUAL"입니다',
      imageUrl : '<url value="/pictures/placeholder/광고 1.jpg"/>',
      link : {
        mobileWebUrl : 'http://52.78.69.41:8080/formula/',
        webUrl : 'http://52.78.69.41:8080/formula/',
      },
    },
  },
  buttons : [ {
    title : '웹으로 보기',
    link : {
      mobileWebUrl : 'http://52.78.69.41:8080/formula/item/'+itemIdx+'/1',
      webUrl : 'http://52.78.69.41:8080/formula/item/'+itemIdx+'/1',
    },
  }, ],
});
// 카카오톡 모듈이 시 카카오톡 설치 완료임을
installTalk : true,
```

상품 상세페이지 - 작성자 정보 확인하기

01 UI/UX 화면

1

[25년 경력] 사업자 전문 세무 관리

포트폴리오

2

프로세스관리

백화문

1. 주제에 대한 고찰

나는 학생시절부터 컴퓨터에서 보내는 시간이 꽤나 많고, 공학, 경영, 디자인, 건축, 문화, 예술 등 다양한 분야에 관심이 있다. 특히, 디자인 분야에 관심이 많고, 디자인을 공부하는 동안에는 디자인 관련 책을 많이 읽었다. 디자인을 공부하는 동안에는 디자인 관련 책을 많이 읽었다. 디자인을 공부하는 동안에는 디자인 관련 책을 많이 읽었다.

앞서 말한 주제 후반은 디자인 관련 책을 많이 읽었다. 디자인을 공부하는 동안에는 디자인 관련 책을 많이 읽었다. 디자인을 공부하는 동안에는 디자인 관련 책을 많이 읽었다.

2. 주제 선정 이유

오늘날 사회는 매우 빠르게 변화하고 있다. 디자인 분야는 특히 그렇다. 디자인 분야는 특히 그렇다. 디자인 분야는 특히 그렇다. 디자인 분야는 특히 그렇다. 디자인 분야는 특히 그렇다.

제임스

전문가 정보

전화번호

01012331332

직무 및 경력

직무: 세무

경력: 25년

경력 사항

Vina

Vietnam

ALS

지정사항

안녕하세요 박하문입니다

안녕하세요 박하문입니다

안녕하세요 박하문입니다

안녕하세요 박하문입니다

안녕하세요 박하문입니다

전문가 번호 : 13

주요 기능

1. 전문가 번호 클릭 기능 수정
2. 세션에서 로그인 사용자를 가져와 사용하는 것이 아닌, 해당 글을 작성한 전문가의 전문가 번호를 가져와 Mapper에서 사용
3. 해당 전문가의 포트폴리오 정보 출력

```
<a href="<c:url value="expertoutput/${item.expertIdx }"/>">전문가  
번호 : ${item.expertIdx }</a>
```

```
@GetMapping("/{itemIdx}/expertoutput/{expertIdx}")  
public String goExpertOutput(@PathVariable Integer itemIdx, @ModelAttribute Expert originalExpert, Model model) {  
    Item item = itemDetailsService.getItem(itemIdx);  
    Expert expert = expertInputService.getOriginalExpert(item.getExpertIdx());  
    int expertIdx = expert.getExpertIdx();  
  
    originalExpert = expertInputService.getOriginalExpert(expertIdx);  
    model.addAttribute("item", item);  
    model.addAttribute("expert", expert);  
    model.addAttribute("originalExpert", originalExpert);  
  
    log.info("Showing modify form for expertIdx: {}", expertIdx);  
    log.info("Showing modify form for originalExpert: {}", originalExpert);  
  
    return "expert-output";  
}
```

```
@Override  
public Expert findExpertById(int memberIdx) {  
    Expert findExpert = itemDetailsService.findExpertById(memberIdx);  
  
    if (ObjectUtils.isEmpty(findExpert)) {  
        throw new NotFoundException("해당 전문가 번호가 없습니다.");  
    }  
  
    return findExpert;  
}
```




장바구니

1

```
$.ajax({
  type: "POST",
  url: "${pageContext.request.contextPath}/item/" + itemIdx + "/1",
  success: function(response) {
    if (response === "success") {
      alert("장바구니에 상품이 추가되었습니다.");
      cartButton.attr("src", afterPhotoURL); // 이미지 변경
    } else {
      alert("로그인 사용자만 가능합니다.");
    }
  },
  $.ajax({
    type: "DELETE",
    url: "${pageContext.request.contextPath}/item/" + itemIdx + "/delete",
    dataType: "text",
    success: function(result) {
      if (result === "success") {
        alert("장바구니를 삭제하였습니다.");
        cartButton.attr("src", beforePhotoURL); // 이미지 변경
      } else {
        alert("장바구니 삭제 중 오류가 발생했습니다.");
      }
    },
    error: function(xhr) {
      alert("장바구니 삭제 중 오류가 발생했습니다." + xhr.status);
    }
  })
},

```

2

```
@Slf4j
@RestController
```

```
@RequiredArgsConstructor
```

```
public class CartRestController {
```

3

```
// 아이템 페이지에서 장바구니 삭제
```

```
@PreAuthorize("hasRole('ROLE_MEMBER')")
```

```
@DeleteMapping("/item/{itemIdx}/delete")
```

```
public String removeItemCart(@PathVariable int itemIdx, HttpSession session) {
```

```
// 세션에서 memberId 추출
```

```
CustomMemberDetails member = (CustomMemberDetails) session.getAttribute(SessionConst.Login_Member);
```

```
int memberId = member.getMemberIdx();
```

```
log.info("deletemapping의 memberId:{}", memberId);
```

```
log.info("deletemapping의 itemIdx :{}", itemIdx);
```

```
cartService.removeCart(itemIdx, memberId);
```















```
return "success";
```

주요 코드

1. Ajax 통신을 이용해 상품의 추가 및 삭제 처리 작성
2. Rest방식을 이용하여 프론트와 백 분리하여 장바구니 삭제 및 추가 처리
3. Security를 적용하여 'ROLE_MEMBER'만 접근 가능하도록 작성



회의록

 20230801 사전준비, 첫번째 회의	2023-08-03 오전 10:07	텍스트 문서	2KB
 20230803 디자인 결정, 두번째 회의	2023-08-03 오후 8:13	텍스트 문서	3KB
 20230807 화면 정의서, 세번째 회의	2023-08-07 오후 11:25	텍스트 문서	2KB
 20230809 ERD와 화면 정의서의 결합, ...	2023-08-09 오후 11:43	텍스트 문서	3KB
 20230811 ERD 수정(1차), 다섯번째 회의	2023-08-11 오후 8:44	Microsoft Word ...	17KB
 20230817 DB에 테이블 생성,여섯번째 ...	2023-08-17 오후 5:18	텍스트 문서	1KB
 20230818 프론트 작업, 일곱번째 회의	2023-08-18 오후 4:29	Microsoft Word ...	20KB
 20230818 프론트 작업, 여덟번째 회의	2023-08-24 오후 3:13	Microsoft Word ...	20KB
 20230830 프로그램 발표 전날, 아홉번...	2023-08-30 오후 5:38	Microsoft Word ...	16KB
 20230901 파이널 프로젝트 준비, 첫번...	2023-09-01 오후 6:22	Microsoft Word ...	17KB
 20230904 파이널 프로젝트 작업 시작, ...	2023-09-04 오후 5:56	Microsoft Word ...	17KB
 20230906 파이널 프로젝트, 세번째 회의	2023-09-08 오후 1:35	텍스트 문서	1KB
 20230914 파이널 프로젝트, 두번째 회의	2023-09-14 오후 3:12	Microsoft Word ...	18KB
 20230918 파이널 프로젝트, 세번째 회의	2023-09-18 오후 5:24	Microsoft Word ...	13KB




회의 주요 토론 내용

1. 역할 분담
2. 각 기능 별 연결점 생성 및 페이지 이동 시 필요한 Data에 대한 합의
3. 새로운 기능에 대한 회의
4. 기능별 마감일자 설정 및 회의 시 자신의 역할에 대한 브리핑
5. 기능 별 개발자가 보지 못한 문제점, 개선방안에 대한 토론



결론

FeedBack

 20230908 파이널 프로젝트, 첫번째 피...	2023-09-08 오후 6:43	Microsoft Word ...	21KB
 20230914 파이널 프로젝트, 두번째 피...	2023-09-14 오후 6:39	Microsoft Word ...	20KB
 20230921 파이널 프로젝트, 세번째 피...	2023-09-25 오후 3:31	Microsoft Word ...	13KB

1. AWS를 사용하여 서버 개발에 대한 FeedBack 적용
2. POSTMAN을 사용하여 Back, Front를 분리하여 테스트 진행
3. 또 뭐있지
4. 3번째엔 뭐했음? 암튼 그런거 생각나는 사람이 채워넣어

5조

감사합니다