

JOBARA

구인구직사이트

2조

조장 : 김태훈

조원 : 이상현

강민호

김지윤

CONTENTS

1. 주식처리

2. Filter

3. Controller

4. DTO 변환

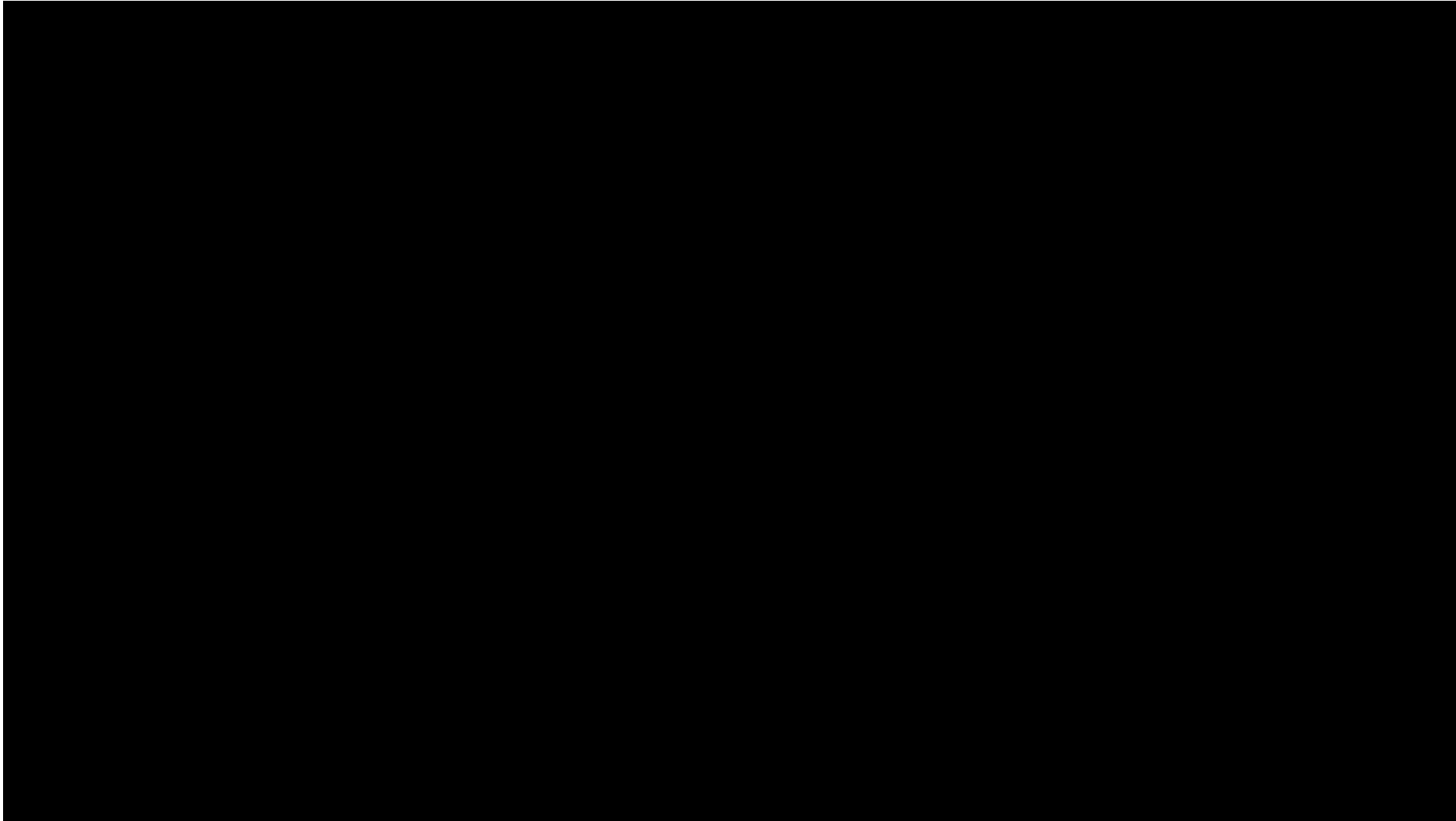
5. Base64

6. 테스트 코드

7. 배포

8. 후기

0. 시연영상



01. 주식처리

Controller

```
@PostMapping("/company/{id}")
@CompanyCheckApi
public ResponseEntity<?> update(@PathVariable Integer id, @RequestBody @Valid CompanyUpdateReqDto compan
    BindingResult bindingResult) {
    // 1. 기능 : 기업 회원 수정 기능을 구현하는 메소드
    // 2. Arguments :
    // - CompanyUpdateReqDto
    // (password, email, address, detailAddress, tel, profile, companyName, companyScale, companyField
    // password : 최소 문자열길이 2, 최대 문자열길이 32, empty
    // email : 최소 문자열길이 2, 최대 문자열길이 32, empty
    // address : 최소 문자열길이 2, 최대 문자열길이 32, empty
    // detailAddress : 최소 문자열길이 2, 최대 문자열길이 64, empty
    // tel : 최소 문자열길이 2, 최대 문자열길이 64, empty
    // profile : empty
    // companyName : 최소 문자열길이 1, 최대 문자열길이 16, empty
    // companyScale : 최소 문자열길이 1, 최대 문자열길이 8, empty
    // companyField : 최소 문자열길이 1, 최대 문자열길이 16, empty

    // 3. Return :

    // 작성자 : ---
    // 작성일 : 2023-03-24
    // 수정자 : -
    // 수정일 : -
```

01. 주식처리

Service

```
@Transactional(readOnly = true)
public BoardDetailRespDto getDetail(Integer boardId, LoginUser principal) {
    // @GetMapping("/boards/{id}")에 의해 호출됨.
    // 기능 : 게시글 상세보기 View에 필요한 데이터를 가져와 Controller에 전달
    // 사용되는 요소 : skill 파싱, 좋아요(스크랩) 파싱, career/jobType/education 파싱












    // 진행 과정 :
    // 1. DB에서 필요 데이터 가져오기
    // - findBoardDetailByJoin 메서드를 통해 요청 데이터(게시글 상세보기 View)를 가져온다.
    // - principalId은 love 테이블, resume 테이블 join에 사용된다.
    // (게시글 상세보기 내의 특정 유저 좋아요 활성화 상태, 이력서 가져오는데 활용)
    // - boardId는 요청 게시물의 board 테이블의 where 절에 거는데 사용된다.
    // 2. 파싱과정
    // - skillParse : DB에 들어가있는 skill 정보를 STRING_AGG query 문법을 사용해
    // 한 속성에 1,2,3 문자열로 가져온다. 이 문자열을 List<Integer> skill에 담기위한 파싱
    // - faSoild : 좋아요한 적이 있다면 LoveDto의 id에 0, css 변수에 "" 빈문자열,
    // 없다면 LoveDto의 id에 해당 id값, css 변수에 "fa-solid"을 저장
    // (View에서는 태그에 .id, .css를 활용해 뿌리기만 할 수 있음)
    // - parseIntegerInfo : career, jobType, education은 DB에 숫자로 저장되며, View에는 문자열로
    // 뿌려진다.
    // View와 DB에서의 표현 방식이 다른것을 파싱한다.
    // (이렇게 다르게 저장한 이유는 설계당시 query의 조건절에 걸 때의 용이성과 DB의 부하를 조금이나마 신경쓰자는 취지였


    // 작성자 : ---
    // 작성일 : 2023-03-24
    // 수정자 : -
    // 수정일 : -
```

1-2. 협업 전략

Commit log

Commits on Mar 25, 2023


Resume test topic
 K-Minho committed 2 days ago
import 정리
 K-Minho committed 2 days ago
resume delete test
 K-Minho committed 2 days ago
resume update test
 K-Minho committed 2 days ago
resumesave test
 K-Minho committed 2 days ago
resumelist test
 K-Minho committed 2 days ago
Employee test 완료 ...
 K-Minho committed 2 days ago
update test
 K-Minho committed 2 days ago
skillupdate test
 K-Minho committed 2 days ago
detail test
 K-Minho committed 2 days ago
updateForm test
 K-Minho committed 2 days ago

 71 branches

 276 commits



167개의
commit 추가

 27 branches

 443 commits

02.

Filter

로그인 - JWT Token 응답

```
@PostMapping("/login")
public ResponseEntity<?> login(@RequestBody UserLoginReqDto userLoginReqDto, String remember,
    HttpServletResponse response) {
    Verify.validateString(userLoginReqDto.getUsername(), msg: "유저네임을 입력하세요.");
    Verify.validateString(userLoginReqDto.getPassword(), msg: "암호를 입력하세요.");
    User userPS = userService.getUser(userLoginReqDto);
    Cookie cookie = new Cookie(name: "remember", userPS.getUsername());
    response.addCookie(cookie);
    String jwt = JwtProvider.create(userPS);
    return ResponseEntity.ok().header(JwtProvider.HEADER, jwt).body(body: "로그인 성공");
}
```

JWT Token 생성

```
public class JwtProvider {
    private static final String SUBJECT = "jwt";
    private static final int EXP = 1000 * 60 * 60;
    public static final String TOKEN_PREFIX = "Bearer ";
    public static final String HEADER = "Authorization";
    private static final String SECRET = "thisIsSecretCode";

    public static String create(User user) {
        String jwt = JWT.create()
            .withSubject(SUBJECT)
            .withExpiresAt(new Date(System.currentTimeMillis() + EXP))
            .withClaim(name: "id", user.getId())
            .withClaim(name: "role", user.getRole())
            .sign(Algorithm.HMAC512(SECRET));
        return TOKEN_PREFIX + jwt;
    }
}
```

02. Filter

토큰검증 & Session 저장

```
public class JwtVerifyFilter implements Filter {
    @Override
    public void doFilter(ServletRequest request, ServletResponse response, FilterChain chain)
        throws IOException, ServletException {
        HttpServletRequest req = (HttpServletRequest) request;
        HttpServletResponse resp = (HttpServletResponse) response;
        String prefixJwt = req.getHeader(JwtProvider.HEADER);

        if (prefixJwt == null) {
            resp.setStatus(sc:401);
            resp.setContentType(type:"text/plain; charset=utf-8");
            resp.getWriter().println(x:"로그인이 필요합니다.");
        } else {
            String jwt = prefixJwt.replace(JwtProvider.TOKEN_PREFIX, replacement:"");
            try {
                DecodedJWT decodedJWT = JwtProvider.verify(jwt);
                int id = decodedJWT.getClaim(name:"id").asInt();
                String role = decodedJWT.getClaim(name:"role").asString();

                // 내부적으로 권한처리
                HttpSession session = req.getSession();
                LoginUser loginUser = LoginUser.builder().id(id).role(role).build();
                session.setAttribute(name:"loginUser", loginUser);
            } catch (Exception e) {
                // JWT 검증 실패
            }
        }
        chain.doFilter(request, response);
    }
}
```

Session 활용

```
@GetMapping("/company/boards/myList/{id}")
@CompanyCheckApi
public ResponseEntity<?> myBoardList(@PathVariable int id) {
    LoginUser principal = (LoginUser) session.getAttribute(name:"loginUser");

    List<BoardMyListRespDto> myBoardListPS = boardService.getMyBoardList(id, principal.getId());
    return ResponseEntity.ok(myBoardListPS);
}
```


02. Filter

Filter 활용 - Url 등록

```
@Configuration
public class FilterRegisterConfig {

    @Bean
    public FilterRegistrationBean<?> jwtVerifyFilterAdd() {
        FilterRegistrationBean<JwtVerifyFilter> registraion = new FilterRegistrationBean<>();
        registraion.setFilter(new JwtVerifyFilter());
        registraion.addUrlPatterns(...urlPatterns: "/employee/*");
        registraion.addUrlPatterns(...urlPatterns: "/company/*");
        registraion.addUrlPatterns(...urlPatterns: "/loves/*");
        registraion.setOrder(order:1);
        return registraion;
    }
}
```

```
@GetMapping("/company/boards/myList/{id}")
@CompanyCheckApi
public ResponseEntity<?> myBoardList(@PathVariable
```

```
@GetMapping("/employee/boards/myScrapList/{id}")
@EmployeeCheckApi
public ResponseEntity<?> myScrapBoardList(@PathVa
```

```
@PostMapping("/loves")
@EmployeeCheck
public ResponseEntity<?> save(@RequestBody LoveS
```

인터셉터를 사용하지 않은 이유?

03. Controller

URL 설계

공통 - {id}는 해당되는 요소 바로 뒤에 붙인다.

ex) employee PK id일경우 employee 바로 뒤에 작성

로그인 필요 - 필요한 role을 앞에 작성

ex) employee/{id}, company/{id}

board에서 권한이 필요하지 않은 내용은 단순히 board만 작성함.

ex) boards/{id} (상세보기), boards (전체보기)

단, board에서 인증이 필요한 내용들은 앞에 company를 작성함.

ex) company/board/{ (board의) id }

love는 양쪽다 로그인이 필요하기에 개별적인 /love로 필터링함.

로그인이 불필요하되, 역할의 구분이 필요하다면 해당 엔드포인트 + role

ex) joinEmployee

03. Controller

URL 설계

apply는 현 사용자의 role이 맨앞에 오는 방식으로 함.

∴ ex) 지원자가 지원 상황을 볼때, /employee/{ (employee의) id }/apply
기업이 지원자를 관리할때, /company/apply/{ (apply의) id }

resume은 앞에 employee를 붙인다.

ex) employee/resume/{id}

employee 보유기술 수정은 다음과 같이 작성함.

ex) /employee/{id}/skill

employee 상세보기는 다음과 같이 작성함.

ex) /user/{id}

03. Controller

Rest API

4. 구직자 유저 정보 수정

요청

- 메서드 : put
- 경로(url) : employee/{id}

응답

- MIME -> application/json

요청 값

```
{
  "id" : Integer
  "password" : String
  "email": String
  "address" : String
  "detailAddress" : String
  "tel": "String
  "employeeDto" : {
    "realName": String
    "education": String
    "career" : Integer
  }
}
```

출력 예시

```
{
  "code": 1,
  "msg": "수정 완료",
  "data": null
}
```

03. Controller

유효성 검사

```
@NotEmpty(message = "제목을 입력하세요")
@Size(max = 16, message = "제목은 16자 이내로 입력해주세요")
private String title;

    public ResponseEntity<?> save(@RequestBody @Valid BoardInsertReqDto boardInsertReqDto,
    @NotEmpty(message = " " BindingResult bindingResult) {
@Size(max = 65536, message = "내용이 너무 겁니다.")
private String content;

private Integer career;

@NotEmpty(message = "경력을 선택하세요")
private String careerString;

private Integer education;

@NotEmpty(message = "학력을 선택하세요")
private String educationString;
```

03. Controller

ResponseDto

```
@Getter
@Setter
@AllArgsConstructor
public class ResponseDto<T> {
    private Integer code;
    private String msg;
    private T data;
}
```

```
return new ResponseEntity<>(new ResponseDto<>(code:1, msg:"게시글 등록 성공", data:null), HttpStatus.CREATED);
```

```
return new ResponseEntity<>(new ResponseDto<>(code:1, msg:"등록 게시글 목록", myBoardListPS), HttpStatus.OK);
```

04. DTO 변환

Version 3 활용

Version 1

```
@Getter @Setter @ToString
public class BoardJoinUserDtoV1 {
    private Integer id;
    private String title;
    private String content;
    private UserDto user;
    private Timestamp createdAt;

    public BoardJoinUserDtoV1(BoardJoinUserDtoV1Flattern board) {
        this.id = board.getId();
        this.title = board.getTitle();
        this.content = board.getContent();
        this.user = new UserDto(
            board.getUserId(),
            board.getUserUsername(),
            board.getUserPassword(),
            board.getUserEmail(),
            board.getUserCreatedAt()
        );
        this.createdAt = board.getCreatedAt();
    }
}
```

Version 2

```
@Getter @Setter
public class BoardJoinUserDtoV2 {
    private BoardDto board;
    private UserDto user;
}
```

Version 3

```
@Getter @Setter
public class BoardJoinUserDtoV3 {
    private Integer id;
    private String title;
    private String content;
    private UserDto user;
    private Timestamp createdAt;

    @Getter @Setter
    public static class UserDto {
        private Integer id;
        private String username;
        private String password;
        private String email;
        private Timestamp createdAt;
    }
}
```

04.

DTO 변환

Mapping

```
<resultMap id="boardDetailJoin" type="shop.mtcoding.jobara.board.dto.BoardDetailRespDto">
  <id property="id" column="id"/>
  <result property="title" column="title"/>
  <result property="content" column="content"/>
  <result property="careerInteger" column="career"/>
  <result property="jobTypeInteger" column="job_type"/>
  <result property="educationInteger" column="education"/>
  <result property="favor" column="favor"/>
  <association property="company" javaType="shop.mtcoding.jobara.board.dto.BoardDetailRespDto$CompanyD
    <id property="userId" column="company_user_id"/>
    <result property="companyName" column="company_name"/>
    <result property="companyScale" column="company_scale"/>
    <result property="companyField" column="company_field"/>
  </association>
  <association property="user" javaType="shop.mtcoding.jobara.board.dto.BoardDetailRespDto$UserDto">
    <id property="id" column="user_id"/>
    <result property="profile" column="profile"/>
  </association>
  <association property="love" javaType="shop.mtcoding.jobara.board.dto.BoardDetailRespDto$LoveDto
    <id property="id" column="love_id"/>
    <id property="boardId" column="love_board_id"/>
    <id property="userId" column="love_user_id"/>
  </association>
  <collection property="resume" javaType="java.util.ArrayList" ofType="shop.mtcoding.jobara.board.
    <id property="id" column="resume_id"/>
    <result property="userId" column="resume_user_id"/>
    <result property="title" column="resume_title"/>
    <result property="content" column="resume_content"/>
    <result property="createdAt" column="resume_created_at"/>
  </collection>
  <collection property="needParse" ofType="java.lang.String">
    <result column="skills" />
  </collection>
</resultMap>
```

```
@Getter
@Setter
public class BoardDetailRespDto {
    private Integer id;
    private String title;
    private String content;
    @JsonIgnore
    private Integer careerInteger;
    @JsonIgnore
    private Integer jobTypeInteger;
    @JsonIgnore
    private Integer educationInteger;
    private String career;
    private String jobType;
    private String education;
    private String favor;

    private CompanyDto company;
    private UserDto user;
    private LoveDto love;
    private List<ResumeDto> resume;
    @JsonIgnore
    private String needParse;
    private List<Integer> skill;

    public void skillParse(String needParse) {
```


05. Base64

디코딩&하드디스크 저장과정

```
public static String saveImage(String base64Image) throws IOException {
    checkExist(base64Image);
    byte[] decodedData = decode(base64Image);
    String mimeType = checkImage(decodedData);
    String staticFolder = System.getProperty(key:"user.dir") + "\\src\\main\\resources\\static\\";
    UUID uuid = UUID.randomUUID();
    String filePath = "\\images\\" + uuid + "_" + System.currentTimeMillis() + "."
        + mimeType.split(regex:"/")[1];

    // filePath :
    // \images\ uuid값_시간.프로필사진.png
    Path imageFilePath = Paths.get(staticFolder + "\\" + filePath);
    Files.write(imageFilePath, decodedData);

    return imageFilePath.toString();
}
```

```
private static void checkExist(String base64Image) {
    if (base64Image.isEmpty()) {
        throw new CustomApiException(msg:"사진이 전송되지 않았습니다");
    }
}
```

```
private static byte[] decode(String base64Image) {
    String[] parts = base64Image.split(regex:",");
    // base64Data : data:image/png;base64, 없애 base64 String 값
    String base64Data = parts[1];
    // 하드 디스크는 이진데이터를 읽어 저장하므로 base64 문자셋 -> 이진 데이터 디코딩
    byte[] decodedData = Base64.getDecoder().decode(base64Data);
    return decodedData;
}
```

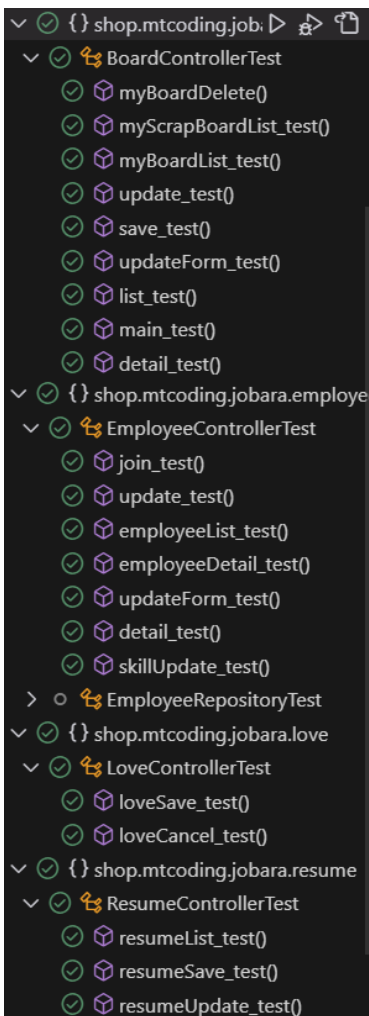
```
private static String checkImage(byte[] decodedData) throws IOException {
    String mimeType = null;
    mimeType = URLConnection.guessContentTypeFromStream(new ByteArrayInputStream(decodedData));

    if (!mimeType.startsWith(prefix:"image/")) {
        throw new CustomApiException(msg:"사진 파일만 업로드 할 수 있습니다.");
    }

    return mimeType;
}
```

06. 테스트 코드

Controller Test



```
@WebMvcTest(ApplyController.class) @MockBean
public class ApplyControllerTest { private ApplyService applyService;
```

```
@Test
public void companyApplyList_test() throws Exception {
    // given
    Integer id = 6;

    // mock
    List<ApplyJoinBoardAndUser> applyListPS = new ArrayList<>();
    ApplyJoinBoardAndUser applyJoinBoardAndUser = ApplyJoinBoardAndUserBuilder.makeApplyJoinBoardAndUser
        ApplyJoinBoardAndUserBuilder.makeUser(id:1, realName:"김일"),
        ApplyJoinBoardAndUserBuilder.makeBoard(id:1, title:"제목1"),
        ApplyJoinBoardAndUserBuilder.makeResume(id:1));
    applyListPS.add(applyJoinBoardAndUser);
    applyJoinBoardAndUser = ApplyJoinBoardAndUserBuilder.makeApplyJoinBoardAndUser(id:2, state:0,
        ApplyJoinBoardAndUserBuilder.makeUser(id:2, realName:"김이"),
        ApplyJoinBoardAndUserBuilder.makeBoard(id:2, title:"제목2"),
        ApplyJoinBoardAndUserBuilder.makeResume(id:2));
    applyListPS.add(applyJoinBoardAndUser);
    given(applyService.getApplyForCompany(id)).willReturn(applyListPS);

    // when
    ResultActions resultActions = mvc.perform(get("/company/" + id + "/apply"));
    String resp = resultActions.andReturn().getResponse().getContentAsString();
    System.out.println("테스트 : " + resp);

    // then
    resultActions.andExpect(status().is2xxSuccessful());
    resultActions.andExpect(jsonPath(expression:"$.code").value(expectedValue:1));
    resultActions.andExpect(jsonPath(expression:"$.msg").value(expectedValue:"지원자 리스트 불러오기 성공"));
    resultActions.andExpect(jsonPath(expression:"$.data[0].id").value(expectedValue:1));
    resultActions.andExpect(jsonPath(expression:"$.data[0].state").value(expectedValue:"0"));
    resultActions.andExpect(jsonPath(expression:"$.data[0].board.id").value(expectedValue:1));
    resultActions.andExpect(jsonPath(expression:"$.data[0].board.title").value(expectedValue:"제목1"));
    resultActions.andExpect(jsonPath(expression:"$.data[0].user.id").value(expectedValue:1));
    resultActions.andExpect(jsonPath(expression:"$.data[0].user.realName").value(expectedValue:"김일"));
    resultActions.andExpect(jsonPath(expression:"$.data[0].resume.id").value(expectedValue:1));
}
```

07.

배포



배포환경

- AWS EC2 (Elastic Compute Cloud)
- Linux (Amazon Linux 2 또는 다른 Linux 배포판)
- Windows에서 MobaXterm을 사용하여 SSH 접속 및 명령어 실행

서버

- Spring Boot RESTful API 서버
- H2 데이터베이스와 연동
- HTTP 프로토콜을 사용하여 클라이언트 요청을 처리

배포 방식

- Spring Boot 애플리케이션을 JAR 파일 형태로 빌드
- EC2 인스턴스에 JAR 파일을 전송하여 애플리케이션 배포
- EC2 인스턴스 내에서 H2 데이터베이스 실행 및 연동
- EC2 인스턴스 내에서 Spring Boot 애플리케이션 실행

```
improve startup time and JSP compilation time.
2023-03-26 13:39:55.065 INFO 4410 --- [main] o.a.c.c.C.[Tomcat].[localhost].[/] : Initializing Spring embedded WebApplicationContext
2023-03-26 13:39:55.065 INFO 4410 --- [main] w.s.c.ServletWebServerApplicationContext : Root WebApplicationContext: initialization completed in 553ms
2023-03-26 13:39:55.229 INFO 4410 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Starting...
2023-03-26 13:39:55.739 INFO 4410 --- [main] com.zaxxer.hikari.HikariDataSource : HikariPool-1 - Start completed.
2023-03-26 13:39:55.780 INFO 4410 --- [main] o.s.b.a.h2.H2ConsoleAutoConfiguration : H2 console available at '/h2-console'. Database available at 'jdbc:h2:mem:testdb'
2023-03-26 13:40:00.820 INFO 4410 --- [main] o.s.b.w.embedded.tomcat.TomcatWebServer : Tomcat started on port(s): 8080 (http) with context path ''
2023-03-26 13:40:00.845 INFO 4410 --- [main] shop.mtcoding.jobara.JobaraApplication : Started JobaraApplication in 12.94 seconds (JVM running for 14.202)
^C
root@ip-172-31-42-95:/home/ubuntu/mini-project_rest-human-cloud# ps -ef | grep *.jar
root      4410      1454   9 13:39 pts/1    00:00:14 java -jar jobara-0.0.1-SNAPSHOT.jar
root      4448      1454   0 13:42 pts/1    00:00:00 grep --color=auto m*.jar
root@ip-172-31-42-95:/home/ubuntu/mini-project_rest-human-cloud#
```

07.

배포



← → ↻ ▲ 주의 요함 | 43.201.68.141:8080/home

```
{
  code: 1,
  msg: "메인 페이지 출력 성공",
  data: null
}
```

← → ↻ ▲ 주의 요함 | 43.201.68.141:8080/boards

```
{
  code: 1,
  msg: "광고 리스트 불러오기 성공",
  data: {
    keyword: null,
    blockCount: 5,
    currentBlock: 0,
    currentPage: 0,
    startPageNum: 1,
    lastPageNum: 2,
    totalCount: 9,
    totalPage: 2,
    board: [
      {
        id: 12,
        title: "계약직 - 웹 개발(백엔드)",
        companyName: "KBS미디어",
        dday: "50",
        user: {
          id: 15,
          profile: "/images/kbs.png"
        },
        love: null
      },
      {
        id: 11,
        title: "계약직 - 웹 개발(백엔드)",
        companyName: "KBS미디어",
        dday: "50",
        user: {
          id: 15,
          profile: "/images/kbs.png"
        },
        love: null
      },
      {
        id: 10,
        title: "웹 개발(백엔드)",
        companyName: "프로그래"
      }
    ]
  }
}
```

← → ↻ ▲ 주의 요함 | 43.201.68.141:8080/boards/1

```
{
  code: 1,
  msg: "광고 상세정보 불러오기 성공",
  data: {
    id: 1,
    title: "인공지능 솔루션 (AI Solution) 개발",
    content: "<p style='margin-right: 0px; margin-bottom: system, system-ui, BlinkMacSystemFont, Roboto, 'Sego
Emoji'", &quot;Segoe UI Emoji'", &quot;Segoe UI
0px; padding: 0px; border: 0px; font-family: inherit; fc
입증되었습니다.<br>우리의 현재 산업을 지탱하고 있는 모든
career: "1년이상 ~ 3년미만",
jobType: "인턴",
education: "고졸이상",
favor: "프로젝트 유경험자",
company: {
  userId: 6,
  companyName: "스윙비(Swingvy)",
  companyScale: "대기업",
  companyField: "IT업"
},
user: {
  id: 6,
  profile: "/images/swingb.png"
},
resume: [ ],
love: {
  id: 0,
  boardId: null,
  userId: null,
  css: ""
},
skill: [
  2,
  3,
  4
]
}
}
```

후기

Q&A

JOBARA

Thank you