구인구직사이트(Rest 서버)

6조(박인우 / 이인화 / 김유현 / 강은희)

Contents

01 개발환경 및 기술

개발 언어

Java

개발 환경

Vscode · Spring · GitHub · Mybatis · H2DB · POSTMAN

소통 환경

Notion · Discord

02 목표 및 달성률

프로젝트 목표

- 코드 리팩토링을 통해 코드 수정
- RestAPI 서버 및 문서 구축
- Entity, DTO 변환 로직
- JWT 토큰 인증 방식 구현
- Base64를 적용하여 파일 송수신
- Validation Check 및 AOP
- 클라우드 배포
- Junit 컨트롤러 레이어 테스트
- Sentry.io 모니터링 구성(에러로그 log4j)

달성률

- Sentry.io 모니터링 구성하기(에러로그 log4j) 실패

03 Bulider

```java
@Getter @NoArgsConstructor
public static class PhotoUpdateDto{
    private String fileName;
    private String photo;

    @Builder
    public PhotoUpdateDto(String fileName, String photo) {
        this.fileName = fileName;
        this.photo = photo;
    }
}
```

04 Refactoring Controller

```
@GetMapping("/comp/comphome")
public String compMyhome(Model model) {
    Comp compSession = (Comp)session.getAttribute(name:"compSession");
    if ( compSession == null ){
        return "redirect:/comp/login";
    }
    List<JobsManageJobsRespDto> jDtos = jobsRepository.findByIdtoManageJobs(compSession.getCompId());
    model.addAttribute(attributeName:"jDtos", jDtos);
    Comp compPS = compRepository.findByCompId(compSession.getCompId());
    model.addAttribute(attributeName:"comp", compPS);


    Set<String> set = new HashSet<>();
    List<JobsIdRespDto> jobsIdList = jobsRepository.findJobsIdByCompId(compSession.getCompId());
    for (JobsIdRespDto jobsId : jobsIdList) {
        List<RequiredSkillWriteReqDto> rSkillList = skillRepository.findByJobsSkill(jobsId.getJobsId());
        for (RequiredSkillWriteReqDto skill : rSkillList) {
            set.add(skill.getSkill());
        }
    }

    RequiredSkillByCompRespDto rSkillList = new RequiredSkillByCompRespDto();
    List<String> skillList = new ArrayList<>(set);
    rSkillList.setSkillList(skillList);

    model.addAttribute(attributeName:"sDto", rSkillList);

    List<ResumeMatchRespDto> fiveMatchList = new ArrayList<>();
    List<ResumeMatchRespDto> fourMatchList = new ArrayList<>();
    List<ResumeMatchRespDto> threeMatchList = new ArrayList<>();
    List<ResumeMatchRespDto> twoMatchList = new ArrayList<>();
    List<ResumeMatchRespDto> oneMatchList = new ArrayList<>();

    List<ResumeMatchRespDto> rDtos = resumeRepository.findMatchResumeByCompId(compSession.getCompId());
    for (ResumeMatchRespDto rDto : rDtos) {
```

(메소드를 분리 시켜 캡슐화)

```
@GetMapping("/comp/comphome")
public ResponseEntity<?> compMyhome(@LoginComp LComp comp) {
    CompHomeOutDto compResult = compService.기업홈정보와매칭이력서(comp
    return new ResponseEntity<>(new ResponseDto<>(code:1, msg:"기업 홈
}
```

04 Refactoring Dto

(변경 후 하나의 Dto로 추가)

```java
@Getter
@Setter
public static class JobsManageJobsRespDto {
    private Integer num;
    private Integer jobsId;
    private String title;
    private String position;
    private String career;
    private Timestamp endDate;
    private List<String> skillList;
    private Long leftTime;
}
```

```java
@Getter
@Setter
@ToString
public static class ResumeMatchRespDto {
    private Integer resumeId;
    private String photo;
    private String name;
    private String title;
    private String address;
    private String education;
    private String career;
    private Integer state;
    private List<String> skillList;
    private Integer compScrapId;
}
```

```java
@Getter
@Setter
public static class CompHomeOutDto {
    private List<JobsManageJobsRespDto> jDto;
    private List<ResumeMatchOutDto> rDto;
    private List<String> skillList;

    @Getter
    @Setter
    public static class JobsManageJobsRespDto {
        private Integer jobsId;
        private Integer num;
        private String title;
        private String position;
        private String career;
        private Timestamp endDate;
        private List<String> skillList;
        private Long leftTime;
    }

    @Getter
    @Setter
    public static class ResumeMatchOutDto {
        private Integer resumeId;
        private String title;
        private String address;
        private String education;
        private String career;
        private Integer state;
        private List<String> skillList;
        private UserDto userDto;

        @Getter
        @Setter
        public static class UserDto {
            private Integer userId;
            private String name;
            private String photo;
        }
    }
}
```

05 AOP @LoginUser / @LoginComp

```java
@Target(ElementType.PARAMETER)
@Retention(RetentionPolicy.RUNTIME)
public @interface LoginComp {
}
```

```java
@Target(ElementType.PARAMETER)
@Retention(RetentionPolicy.RUNTIME)
public @interface LoginUser {
}
```

```java
@GetMapping("/comp/comphome")
public ResponseEntity<?> compMyhome(@LoginComp LComp comp) {
    CompHomeOutDto compResult = compService.기업홈정보와매칭이력서(comp.getId());
    return new ResponseEntity<>(new ResponseDto<>(code:1, msg:"기업 홈 조회 성공", compResult), HttpStatus.OK);
}
```

```java
@GetMapping("/user/myhome")
public @ResponseBody ResponseEntity<?> myhome(@LoginUser LUser user) {
    UserHomeOutDto userResult = userService.마이홈조회(user);
    return new ResponseEntity<>(new ResponseDto<>(code:1, msg:"마이홈 보기 성공", userResult), HttpStatus.OK);
}
```

05 AOP @Valid

```java
@Target({ METHOD, FIELD, CONSTRUCTOR, PARAMETER, TYPE_USE })
@Retention(RUNTIME)
@Documented
public @interface Valid {
}
```

```java
@PostMapping("/compjoin")
public ResponseEntity<?> join(@Valid CompJoinReqDto compJoinReqDto, BindingResult bindingResult) {
    CompJoinReqDto compJoinOutDto = compService.회원가입(compJoinReqDto);
    return new ResponseEntity<>(new ResponseDto<>(code:1, msg:"회원가입완료", compJoinOutDto), HttpStatus.OK);
}
```

```java
@Aspect
@Component
public class ValidAdvice {

    @Pointcut("@annotation(org.springframework.web.bind.annotation.PostMapping)")
    public void postMapping() {
    }

    @Pointcut("@annotation(org.springframework.web.bind.annotation.PutMapping)")
    public void putMapping() {
    }

    @Around("postMapping() || putMapping()")
    public Object validationAdvice(ProceedingJoinPoint proceedingJoinPoint) throws Throwable {
        Object[] args = proceedingJoinPoint.getArgs();
        for (Object arg : args) {
            if (arg instanceof BindingResult) {
                BindingResult bindingResult = (BindingResult) arg;

                if (bindingResult.hasErrors()) {
                    Map<String, String> errorMap = new HashMap<>();

                    for (FieldError error : bindingResult.getFieldErrors()) {
                        errorMap.put(error.getField(),error.getDefaultMessage());
                    }
                    throw new MyValidationException(errorMap);
                }
            }
        }
        return proceedingJoinPoint.proceed(); // 정상적으로 해당 메서드를 실행해라!!
    }
}
```

```java
    @Getter
    @Setter
    @AllArgsConstructor
    @NoArgsConstructor
    public static class CompJoinReqDto {
        private Integer compId;
        @NotBlank(message = "이메일은 필수 입력 값입니다/")
        private String email;
        @NotBlank(message = "비밀번호는 필수 입력 값입니다/")
        private String password;
        @NotBlank(message = "회사이름은 필수 입력 값입니다/")
        private String compName;
        @NotBlank(message = "대표자명은 필수 입력 값입니다/")
        private String representativeName;
        @NotBlank(message = "회사번호는 필수 입력 값입니다/")
        private String businessNumber;
        private Timestamp createdAt;
    }

    @Getter
    @Setter
    public static class CompLoginReqDto {
        private Integer compId;
        @NotBlank(message = "이메일은 필수 입력 값입니다/")
        private String email;
        @NotBlank(message = "비밀번호는 필수 입력 값입니다/")
        private String password;
        private String rememberEmail;
    }
```

06 JWT

```java
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);
        String jwt = prefixJwt.replace(JwtProvider.TOKEN_PREFIX, replacement:"");
        try {
            DecodedJWT decodedJWT = JwtProvider.verify(jwt);
            int id = decodedJWT.getClaim(name:"id").asInt();
            String email = decodedJWT.getClaim(name:"email").asString();
            String role = decodedJWT.getClaim(name:"role").asString();
            // 내부에서 공현자리 세션
            if (role.equals(anObject:"user")){
                HttpSession session =  req.getSession();
                LUser loginUser = LUser.builder().id(id).email(email).role(role).build();
                session.setAttribute(name:"principal", loginUser);
                chain.doFilter(req, resp);
            }else{
                HttpSession session =  req.getSession();
                LComp loginComp = LComp.builder().id(id).email(email).role(role).build();
                session.setAttribute(name:"compSession", loginComp);
                chain.doFilter(req, resp);
            }
        }catch (SignatureVerificationException sve){
            resp.setStatus(sc:401);
            resp.setContentType(type:"text/plain; charset=utf-8");
            resp.getWriter().println(x:"로그인 다시해1");
        }catch (TokenExpiredException tee){
            resp.setStatus(sc:401);
            resp.setContentType(type:"text/plain; charset=utf-8");
            resp.getWriter().println(x:"로그인 다시해2");
        }
    }
}
```

```java
public class JwtProvider {
    private static final String SUBJECT = "JWT_HMAC";
    private static final int EXP = 1000 * 60 * 60 * 24;
    public static final String TOKEN_PREFIX = "Bearer ";
    public static final String HEADER = "Authorization";

    public static String create(Object obj) {

        if (obj instanceof User) {
            User user = (User) obj;
            String jwt = JWT.create().withSubject(SUBJECT)
                    .withExpiresAt(new Date(System.currentTimeMillis() + EXP))
                    .withClaim(name:"id", user.getUserId())
                    .withClaim(name:"email", user.getEmail())
                    .withClaim(name:"role", value:"user")
                    .sign(Algorithm.HMAC512(SecretKey.KEY));
            return TOKEN_PREFIX + jwt;
        } else {
            Comp comp = (Comp) obj;
            String jwt = JWT.create().withSubject(SUBJECT)
                    .withExpiresAt(new Date(System.currentTimeMillis() + EXP))
                    .withClaim(name:"id", comp.getCompId())
                    .withClaim(name:"email", comp.getEmail())
                    .withClaim(name:"role", value:"comp")
                    .sign(Algorithm.HMAC512(SecretKey.KEY));
            return TOKEN_PREFIX + jwt;
        }
    }

    public static DecodedJWT verify(String jwt) throws SignatureVerificationException, TokenExpiredException {
        // when
        DecodedJWT decodedJWT = JWT.require(Algorithm.HMAC512(SecretKey.KEY))
                .build().verify(jwt);
        return decodedJWT;
    }
}
```

06 JWT 인증 방식

· 인증이 필요한 부분 : 주소 앞 /user 와 /comp로 설정

```java
@Configuration
public class FilterRegisterConfig {

    @Bean
    public FilterRegistrationBean<?> jwtVerifyFilterRegister(){
        FilterRegistrationBean<JwtVerifyFilter> registration = new FilterRegistrationBean<>();
        registration.setFilter(new JwtVerifyFilter());
        registration.addUrlPatterns(...urlPatterns:"/user/*");
        registration.addUrlPatterns(...urlPatterns:"/comp/*");
        registration.setOrder(order:1);
        return registration;
    }
}
```

07 Test

```java
@AutoConfigureMockMvc
@SpringBootTest(webEnvironment = WebEnvironment.MOCK)
public class ApplyControllerTest {

    @Autowired
    private MockMvc mvc;
    @Autowired
    private ObjectMapper om;

    MockHttpServletRequest request = new MockHttpServletRequest();
    String token ;

    @BeforeEach
    public void mockUserToken(){
        User mockUser = User.builder()
                    .userId(userId:1)
                    .build();
        token = JwtProvider.create(mockUser);
    }

    @Test
    public void applyResume_test() throws Exception {
        // given
        ApplyReqDto aDto = ApplyReqDto.builder()
                            .resumeId(resumeId:1)
                            .jobsId(jobsId:1)
                            .userId(userId:1)
                            .applyId(applyId:1)
                            .build();
        String requestBody = om.writeValueAsString(aDto);

        // when
        MockHttpServletRequestBuilder requestBuilder = MockMvcRequestBuilders.post(urlTemplate:"/user/apply/resume")
                                                .header(JwtProvider.HEADER, token)
                                                .content(requestBody)
                                                .contentType(MediaType.APPLICATION_JSON_VALUE);
        ResultActions result = mvc.perform(requestBuilder);

        // then
        System.out.println("테스트 : "+ result.andReturn().getResponse().getContentAsString());
    }
}
```

```java
@AutoConfigureMockMvc
@SpringBootTest(webEnvironment = WebEnvironment.MOCK)
public class ResumeDtoControllerTest {

    @Autowired
    private MockMvc mvc;

    @Autowired
    private ObjectMapper om;

    MockHttpServletRequest request = new MockHttpServletRequest();
    String token;

    @BeforeEach
    public void mockUserToken() {
        User mockUser = User.builder()
                .userId(userId:1)
                .build();
        token = JwtProvider.create(mockUser);
    }

    @Test
    public void writeResume_test() throws Exception {
        // given
        ResumeWriteReqDto rDto = ResumeWriteReqDto.builder()
                .resumeId(resumeId:1).title(title:"제목")
                .content(content:"내용").education(education:"대졸").career(career:"신입")
                .link(link:"노션").state(state:1)
                .skillList(Arrays.asList(...a:"spring", "java")).build();

        String requestBody = om.writeValueAsString(rDto);

        // when
        MockHttpServletRequestBuilder requestBuilder = MockMvcRequestBuilders.post(urlTemplate:"/user/resume/write")
                .header(JwtProvider.HEADER, token)
                .content(requestBody)
                .contentType(MediaType.APPLICATION_JSON_VALUE);

        ResultActions result = mvc.perform(requestBuilder);

        // then
        System.out.println("테스트 : " + result.andReturn().getResponse().getContentAsString());
    }
}
```

08 API Get

1. 공통코드

code : 1 통신 정상

code : -1 통신 실패

msg: 메세지

data : response Body 값

3. 이력서 상세보기

요청 주소 (GET)

- http://localhost:8080/resume/1

요청 헤더

Authorization : Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJKV1RfSE1BQyIsInJvbG
.f1MKJ9ynBbTQcn9Zekvh-e-KGxv3XZx9yxhyQArCaLexMhCfkNp3PcbxXFOdAwhUXEbKUzp1sFMvHs3P-qRvsg

응답 바디

- application/json


```
{
    "code": 1,
    "msg": "이력서 상세보기 완료",
    "data": {
        "resumeId": 1,
        "title": "끊임없이 개선하려는 개발자",
        "content": "백엔드 이력서의 내용입니다.",
        "education": "고졸",
        "career": "신입",
        "link": "블로그 주소",
        "skillList": [
            "HTML/CSS",
            "JavaScript",
            "Oracle",
            "Spring",
            "Vue.js"
        ],
        "user": {
            "userId": "1",
            "photo": "/images/ham.png",
            "name": "박민우",
            "birth": "1994-12-14",
            "address": "서울특별시"
        },
        "compScrap": {
            "compScrapId": "1"
        },
        "apply": {
            "applyId": 6,
            "applyState": 1
        },
        "suggest": {
            "suggestId": 1,
            "suggestState": 1
        }
    }
}
```

08 API Post

2. 기업 회원가입

요청 주소 (POST)
- http://localhost:8080/compjoin

요청 바디
- application/json

```
{
    "email": "grean@nate.com",
    "password": "1234",
    "compName": "그린컴퓨터",
    "representativeName": "김그린",
    "businessNumber": "0511111111"
}
```

응답 바디
- application/json


```
{
    "code": 1,
    "msg": "회원가입완료",
    "data": {
        "compId": 9,
        "email": "grean@nate.com",
        "password": "ea32961dbd579ef5697c367f9267921ee07f14d77fb2d4fb9500d4221d615695",
        "compName": "그린컴퓨터",
        "representativeName": "김그린",
        "businessNumber": "0511111111",
        "createdAt": "2023-03-25T03:14:22.184+00:00"
    }
}
```

08 API Put

9. 구직자 회원수정

요청 주소 (PUT)
- http://localhost:8080/user/update

요청 헤더
Authorization : Bearer eyJ0eXAiOiJKV1QiLCJhbG

요청 바디
- application/json

```
{
    "userId": "1",
    "password": "2580",
    "name": "김그린",
    "birth": "20110101",
    "tel": "01022222222",
    "address": "부산광역시"
}
```

응답 바디
- application/json

```
{
    "code": 1,
    "msg": "수정완료",
    "data": {
        "userId": 1,
        "password": "fdbb42d1f6357d07eda8a62358160c9109278b14c8747b70f7aa011e9f5a0ad5",
        "name": "김그린",
        "birth": "20110101",
        "tel": "01022222222",
        "address": "부산광역시"
    }
}
```

08 API Delete

9. 이력서 삭제

요청 주소 (DELETE)
- http://localhost:8080/user/resume/1/delete

요청 헤더
Authorization : Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzUxMiJ9.eyJzdWIiOiJjb3PthqDtgbAiLCJpZC

응답 바디
- application/json


```
{
    "code": 1,
    "msg": "삭제성공",
    "data": null
}
```

09 소감

박인우	김유현	이인화	강은희
스프링의 새로운 기술들을 사용해봐서 좋았고, REST API를 직접 구현하면서 왜 REST아키텍처를 이용하는지 조금 알게 되어서 좋았다.	2차 프로젝트 까지 하면서, 아직도 개념이 잡히지 않거나 이해가 어려웠던 부분을 팀원들과의 조언과 공부하는 시간을 통해 조금 더 성장하는 시간이 였고 부족한 부분을 더 채워 나가고 싶다.	1,2차 프로젝트를 하면서 부족함을 많이 느꼈지만, 많이 배웠고 이것을 토대로 더 열심히 할 것이다.	지난 시간에 못한 부분을 채울 수 있는시간 같아서 다행이라고 생각했습니다. 부족하기도 했지만, 많이 배운 시간이라고 생각합니다.