로그인 및 로그아웃

```java
@Configuration
@EnableWebSecurity // springSecurity를 사용하기위한 어노테이션
public class SpringSecurityConfig extends WebSecurityConfigurerAdapter {

    private final AuthenticationService authenticationService;
    @Autowired
    public SpringSecurityConfig(AuthenticationService authenticationService) {
        this.authenticationService = authenticationService;
    }

    @Bean
    public PasswordEncoder passwordEncoder(){

        return new BCryptPasswordEncoder();
    }

    @Override
    protected void configure(AuthenticationManagerBuilder auth) throws Exception {
        auth.userDetailsService(authenticationService).passwordEncoder(passwordEncoder());
    }

    @Override
    public void configure(WebSecurity web){

        /* 인증 무시 설정 */
        web.ignoring().antMatchers("/css/**", "/js/**", "/images/**", "/lib/**");
    }

    /* HTTP요청에 대한 권한 설정 */
    @Override
    protected void configure(HttpSecurity http) throws Exception {
        //권한별 접근 페이지 설정
        http
                .authorizeRequests() // 리소스의 권한 설정
                .mvcMatchers("/**", "/member/**").permitAll() // antMatchers 설정한 리소스의 접근을 인증절차
없이 허용한다는 의미
                .and()
                .csrf().disable();

        //로그인 로그아웃 설정
        http
                .formLogin()
                .loginPage("/member/login")             // 커스텀 로그인 페이지 사용
                .defaultSuccessUrl("/member/loginSuccess") //로그인 성공시 이동 페이지
                .failureUrl("/member/loginFail") // 로그인 실패시 이동 페이지
                .usernameParameter("membeId")           // 아이디 파라미터명 설정
                .passwordParameter("memberPwd")         // 패스워드 파라미터명 설정
                .and()
                .logout()
                .logoutRequestMatcher(new AntPathRequestMatcher("/member/logout"))
                .deleteCookies("JSESSIONID")
                .invalidateHttpSession(true)
                .logoutSuccessUrl("/");
    }
```

```java
public static void invalidateSession(HttpServletRequest request, HttpServletResponse response){
        boolean isSecure = false;
        String contextPath = null;
        if (request != null) {
            HttpSession session = request.getSession(false);
            if (session != null) {
                session.invalidate();
            }
            isSecure = request.isSecure();
            contextPath = request.getContextPath();
        }
        SecurityContext context = SecurityContextHolder.getContext();
        SecurityContextHolder.clearContext();
        context.setAuthentication(null);
        if (response != null) {
            Cookie cookie = new Cookie("JSESSIONID", null);
            String cookiePath = StringUtils.hasText(contextPath) ? contextPath : "/";
            cookie.setPath(cookiePath);
            cookie.setMaxAge(0);
            cookie.setSecure(isSecure);
            response.addCookie(cookie);
        }
    }

/* 로그인 아이디 찾기 */
    @Override
    public UserDetails loadUserByUsername(String memberId) {

        log.info("[AuthenticationService] ================================================");
        log.info("[AuthenticationService] memberId : " + memberId);

        MemberDTO member = memberMapper.findByMemberId(memberId);

        log.info("[AuthenticationService] member : " + member);

        return member;
    }
```

```xml
<!-- 로그인 아이디 찾기 -->
    <select id="findByMemberId" resultType="MemberDTO">
        SELECT
                MEMBER_NO
              , MEMBER_ID
              , MEMBER_PWD
              , NICKNAME
              , PHONE
              , EMAIL
              , ADDRESS
              , ENROLL_DATE
              , MEMBER_YN
              , MEMBER_ROLE
          FROM MEMBER
         WHERE MEMBER_YN = 'Y'
           AND MEMBER_ID = #{ memberId
    </select>
```

- Spring Security config 및 Session 설정을 하였습니다.
- SessionUtil.invalidateSession()를 사용하여 현재 로그인중인 세션을 초기화 하였습니다.
- UserDetailsService를 재정의하여 로그인 메소드를 구현하였습니다.

회원가입

```java
/* 회원가입 */
    @PostMapping("/regist")
    public String memberRegist(@ModelAttribute MemberDTO member, RedirectAttributes rttr,
HttpServletRequest request){

        /* 주소API를 사용하여 받아온 주소를 합쳐준다.*/
        String address = request.getParameter("zipCode") + " " + request.getParameter("address1") + " "
+ request.getParameter("address2");
        /* 핸드폰번호 '-' 삭제 */
        member.setPhone(member.getPhone().replace("-", ""));
        member.setAddress(address);
        /* 비밀번호 인코딩 */
        member.setMemberPwd(passwordEncoder.encode(member.getMemberPwd()));

        log.info("");
        log.info("");
        log.info("[MemberController] member : " + member);

        memberService.registMember(member);

        rttr.addFlashAttribute("message", "회원 가입에 성공하였습니다.");

        return "redirect:/";
    }

/* 회원가입시 아이디 중복체크 */
    @PostMapping("/idDupCheck")
    public ResponseEntity<String> idDupCheck(@RequestBody MemberDTO member){

        log.info("");
        log.info("");
        log.info("[MemberController] member ID check : " + member.getMemberId());

        String result = "사용 가능한 아이디 입니다.";
        log.info("[MemberController] Request Check ID : " + member.getMemberId());

        if("".equals(member.getMemberId())) {
            log.info("[MemberController] 아이디를 입력해 주세요");
            result = "아이디를 입력해 주세요.";
        } else if(memberService.selectMemberById(member.getMemberId())) {
            log.info("[MemberController] 중복된 아이디가 존재합니다.");
            result = "중복된 아이디가 존재합니다.";
        }

        return ResponseEntity.ok(result);
    }
```

```javascript
<script>
  if(document.getElementById("duplicationCheck")) {

    const $duplication = document.getElementById("duplicationCheck");
    const $id = document.getElementById("memberId");
    $duplication.onclick = function() {

      if($id.value.length >= 5) {
        let memberId = document.getElementById("memberId").value.trim();

        fetch("/member/idDupCheck", {
          method: "POST",
          headers: {
            'Content-Type': 'application/json;charset=UTF-8'
          },
          body: JSON.stringify({memberId: memberId})
        })
                .then(result => result.text())
                .then(result => alert(result))
                .catch((error) => error.text().then((res) => alert(res)));
      } else {

        alert("아이디를 5글자 이상 입력해주세요")
        return false;
      }
    }
  }

$('.btn-submit').click(function(){

    let id = $('#memberId').val().trim();
    let memberPwd = $('#memberPwd').val().trim();
    let nickname = $('#nickname').val().trim();
    let phone = $('#phone').val();
    let email = $('#email').val();
    let searchZipCode = $('#searchZipCode').val();
    let address1 = $('#address1').val();
    let address2 = $('#address2').val();

    if(id == "" || id == null){

      alert("아이디를 입력하세요.")
      id.focus();
      return false;

    } else if(memberPwd == "" || memberPwd == null){

      alert("비밀번호를 입력하세요.");
      memberPwd.focus();
      return false;
    } else if(nickname == "" || nickname == null){

      alert("닉네임을 입력하세요.");
      nickname.focus();
      return false;
    } else if(phone == "" || phone == null){

      alert("핸드폰 입력하세요.");
      phone.focus();
      return false;
    } else if(email == "" || email == null){

      alert("이메일을 입력하세요.");
      email.focus();
      return false;
    } else if(searchZipCode == "" || searchZipCode == null){

      alert("우편번호를 입력하세요.");
      searchZipCode.focus();
      return false;
    } else if(address1 == "" || address1 == null){

      alert("주소를 입력하세요.");
      address1.focus();
      return false;
    } else if(address2 == "" || address2 == null){

      alert("상세주소를 입력하세요.");
      address2.focus();
      return false;
    }

    return true;
});
</script>
```

```javascript
<script th:inline="javascript">

  const $searchZipCode = document.getElementById("searchZipCode");
  const $goMain = document.getElementById("goMain");

  $searchZipCode.onclick = function() {

    /* 다음 우편번호 검색 창을 오픈하면서 동작할 콜백 메소드를 포함한 객체를 매개변수로 전달한다. */
    new daum.Postcode({
      oncomplete: function(data){
        /* 팝업에서 검색결과 항목을 클릭했을 시 실행할 코드를 작성하는 부분 */
        document.getElementById("zipCode").value = data.zonecode;
        document.getElementById("address1").value = data.address;
        document.getElementById("address2").focus();
      }
    }).open();
  }

  $goMain.onclick = function() {
    location.href = "/";
  }

</script>
```

- PasswordEncoder를 사용하여 비밀번호를 보안화 하였습니다.
- 회원가입시 아이디 중복체크를 할 수 있도록 만들었고, 주소 등록은 카카오 지도 API를 사용하였습니다.

비밀번호 재설정 (1 / 2)

```java
@Autowired
    public MailHandler(JavaMailSender mailSender) throws MessagingException{

        this.mailSender = mailSender;
        message = this.mailSender.createMimeMessage();
        messageHelper =new MimeMessageHelper(message, true, "UTF-8");
    }

    // 제목
    public void setSubject(String subject) throws  MessagingException{
        messageHelper.setSubject(subject);
    }

    // 내용
    public void setText(String htmlContent) throws MessagingException{
        messageHelper.setText(htmlContent, true);
    }

    // 발송지
    public void setFrom(String email, String name) throws UnsupportedEncodingException,
MessagingException{
        messageHelper.setFrom(email, name);
    }

    // 수신지
    public void setTo(String email) throws MessagingException{
        messageHelper.setTo(email);
    }

    public void addInline(String contentId, DataSource dataSource) throws MessagingException{
        messageHelper.addInline(contentId, dataSource);
    }

    // 보내기
    public void send(){
        mailSender.send(message);
    }

/* 랜덤 인증번호 발송 */
    private String init(){
        Random ran = new Random();
        StringBuffer sb = new StringBuffer();
        int num = 0;
        do{
            num = ran.nextInt(75) + 48;
            if((num >= 48 && num <= 57) || (num >= 65 && num <= 90) || (num >= 97 && num <= 122)){
                sb.append((char) num);
            } else {
                continue;
            }
        } while (sb.length() < size);
        if(lowerCheck){
            return sb.toString().toLowerCase();
        }


        return sb.toString();
    }
```

```yaml
spring:
    mail:
      host: smtp.naver.com
      port: 465
      username: 개인정보
      password: 개인정보
      properties:
        mail.smtp.auth: true
        mail.smtp.ssl.enable: true
        mail.smtp.ssl.trust: smtp.naver.com
```

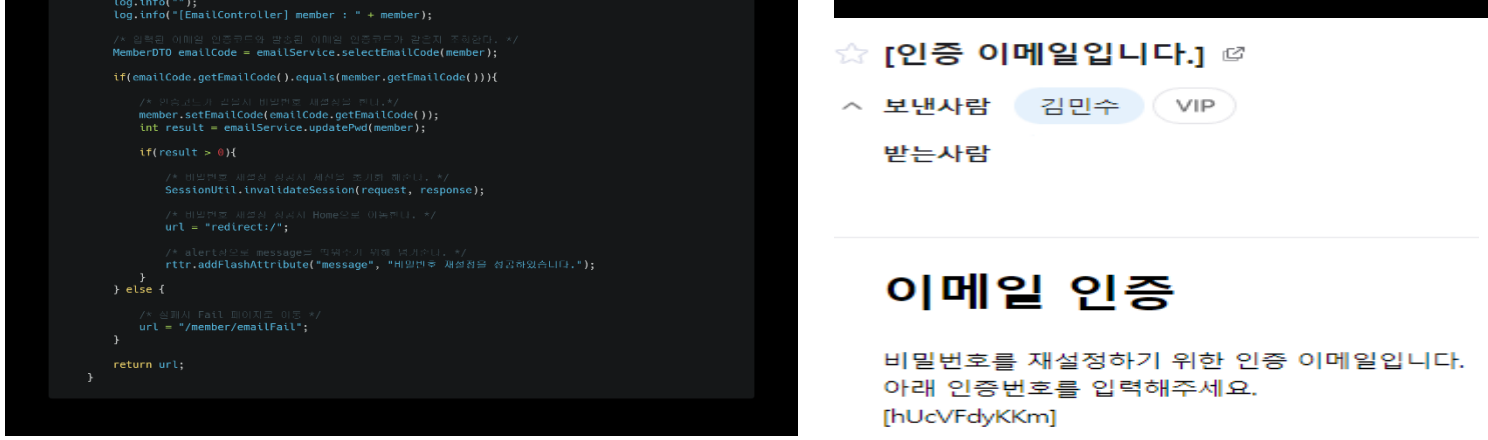
- JavaMailConfig 설정을 하였습니다.
- 메일 인증을 위한 정보를 application.yml에 입력하였습니다

비밀번호 재설정 (2 / 2)

```java
/* 비밀번호 재설정 이메일 전송 */
    @PostMapping("/confirmEmail")
    public ModelAndView confirmEmail(ModelAndView mv, MemberDTO member, HttpServletRequest request)
throws MessagingException, UnsupportedEncodingException {

        member.setMemberId(request.getParameter("memberId"));
        member.setEmail(request.getParameter("email"));

        /* 비밀번호를 수정 하기 위해 해당 ID DB에 저장된 email값을 가져온다. */
        MemberDTO selectEmail = emailService.selectEmail(member);

        log.info("");
        log.info("");
        log.info("[EmailController] email : " + selectEmail);

        /* 해당 ID의 email과 입력한 email이 같은지 조회한다 */
        if(selectEmail.getEmail().equals(member.getEmail())){

            /* email이 같을시 이메일 인증코드를 발송한다. */
            int result = emailService.updateEmailCode(member);

            /* 이메일 인증코드 발송 성공시 */
            if(result > 0){

                mv.addObject("mvMemberId", member.getMemberId());
                mv.addObject("mvEmail", member.getEmail());

                log.info("");
                log.info("");
                log.info("[EmailController] mvMemberId : " + member.getMemberId());
                log.info("[EmailController] mvEmail : " + member.getEmail());

                /* 비밀번호 재설정 페이지로 이동한다. */
                mv.setViewName("/member/updatePwd");
            }

        } else {

            /* 실패시 Fail페이지로 이동 */
            mv.setViewName("/member/emailFail");
        }

        return mv;
    }

    /* 비밀번호 재설정 */
    @PostMapping("/updatePwd")
    public String sendEmail(MemberDTO member, HttpServletRequest request, HttpServletResponse response,
RedirectAttributes rttr){

        /* 비밀번호 재설정을 위하여 setting 해준다. */
        member.setMemberId(request.getParameter("memberId"));
        member.setEmail(request.getParameter("email"));
        member.setMemberPwd(passwordEncoder.encode(request.getParameter("newPwd")));
        member.setEmailCode(request.getParameter("code"));

        String url = "";

        log.info("");
        log.info("");
        log.info("[EmailController] member : " + member);

        /* 입력한 이메일 인증코드와 발송된 이메일 인증코드가 같은지 조회한다. */
        MemberDTO emailCode = emailService.selectEmailCode(member);

        if(emailCode.getEmailCode().equals(member.getEmailCode())){

            /* 인증코드 가 같을시 비밀번호 재설정을 한다.*/
            member.setEmailCode(emailCode.getEmailCode());
            int result = emailService.updatePwd(member);

            if(result > 0){

                /* 비밀번호 재설정 성공시 세션을 초기화 해준다. */
                SessionUtil.invalidateSession(request, response);

                /* 비밀번호 재설정 성공시 Home으로 이동한다. */
                url = "redirect:/";

                /* alert창으로 message를 띄워주기 위해 넘겨준다. */
                rttr.addFlashAttribute("message", "비밀번호 재설정을 성공하였습니다.");
            }
        } else {

            /* 실패시 Fail 페이지로 이동 */
            url = "/member/emailFail";
        }

        return url;
    }
```

```java
/* DB 이메일 인증코드 설정 */
    @Override
    @Transactional
    public int updateEmailCode(MemberDTO member) throws MessagingException,
UnsupportedEncodingException {

        /* 랜덤 인증코드를 member EmailCode에 set한다. */
        String emailCode = new TempKey().getKey(10, false);
        member.setEmailCode(emailCode);

        log.info("");
        log.info("");
        log.info("[EmailServiceImpl] memberEmailCode " + member.getEmailCode());

        /* 이메일 인증코드 설정한다. */
        int result = memberMapper.updateEmailCode(member);

        /* 이메일 인증코드 설정 성공시 */
        if(result > 0){
            MailHandler sendMail = new MailHandler(mailSender);
            sendMail.setSubject("[인증 이메일입니다.]");
            sendMail.setText(
                    "<h1>이메일 인증</h1>" +
                            "<br>비밀번호를 재설정하기 위한 인증 이메일입니다." +
                            "<br>아래 인증번호를 입력해주세요." +
                            "<br>[" + emailCode + "]"
            );
            sendMail.setFrom("minsukim0743@naver.com","김민수");
            sendMail.setTo(member.getEmail());
            sendMail.send();
        }

        return result;
    }
```

```xml
<!-- DB 이메일 인증코드 설정 -->
    <update id="updateEmailCode" parameterType="MemberDTO">
        UPDATE
               MEMBER
           SET EMAIL_CODE = #{ emailCode }
         WHERE EMAIL = #{ email }
           AND MEMBER_ID = #{ memberId }
    </update>

    <!-- 이메일 인증코드 조회 -->
    <select id="selectEmailCode" resultMap="memberResultMap" resultType="String">
        SELECT
               EMAIL_CODE
          FROM MEMBER
         WHERE MEMBER_ID = #{ memberId }
           AND EMAIL = #{ email }
    </select>

    <!-- 비밀번호 재설정 -->
    <update id="updatePwd" parameterType="MemberDTO">
        UPDATE
               MEMBER
           SET MEMBER_PWD = #{ memberPwd }
         WHERE EMAIL_CODE = #{ emailCode }
    </update>

    <!-- 이메일 조회 -->
    <select id="selectEmail" resultMap="memberResultMap">
        SELECT
               EMAIL
          FROM MEMBER
         WHERE MEMBER_ID = #{ memberId }
    </select>
```



- JavaMail SMTP API를 사용하여 이메일 인증코드를 받아 비밀번호를 잊어버렸을 때 비밀번호 재설정을 할 수 있도록 만들었습니다.

댓글 조회

```java
// 댓글 리스트 조회
    @GetMapping("/commentList")
    @ResponseBody
    public String commentList(){

        Gson gson = new Gson();

        List<CommentDTO> commentList = commentService.commentList();

        log.info("");
        log.info("");
        log.info("[commentList]" + commentList);

        return gson.toJson(commentList);
    }
```

```javascript
<script th:inline="javascript">
  // 댓글 리스트
  function commentList(){

      $.ajax({
          url : "/member/commentList",
          type : "GET",
          contentType : 'application/json',

          success : function(data){

              let commentList = JSON.parse(data);
              console.log(commentList);

              let $table = $("#commentTable tbody");
              $table.html("");

              for(let idx in commentList){

                  let $tr = $("<tr>");
                  let $td = $('<td class="writer-regDate">');
                  let $writer = $('<p class="writer">').text(commentList[idx].writer);
                  let $regDate = $('<p class="regDate">').text(commentList[idx].regDate);
                  let $content = $('<td class="content">').text(commentList[idx].content);
                  let $delete = $('<td style="text-align: right"><a class="commentDelete"
onclick="commentDelete('+ commentList[idx].commentNo + ','+commentList[idx].writer+'\');">삭제</a>
</td>');
                  let $commentNo = $('<input type="hidden"
class="commentNo">').text(commentList[idx].commentNo);

                  let writer = commentList[idx].writer;

                  $tr.append($td);
                  $td.append($writer);
                  $td.append($regDate);
                  $tr.append($content);
                  $tr.append($delete)
                  $tr.append($commentNo);

                  if(loginWriter != writer){
                      $delete.css("display", "none");
                  }

                  $table.append($tr);

              }
          },
          error : function(){

              alert('오류!')
          }

      });
  }
<script>
```

```xml
<!-- 댓글 리스트 조회하기 -->
    <select id="commentList" resultMap="commentResultMap">
        SELECT
                COMMENT_NO
              , CONTENT
              , WRITER
              , REG_DATE
          FROM POST_COMMENT
         ORDER BY REG_DATE ASC
    </select>

    <!-- 댓글 개수 조회하기 -->
    <select id="selectTotalCount" resultType="_int">
        SELECT
                COUNT(*)
          FROM POST_COMMENT
    </select>
```

- Ajax를 사용하여 댓글 리스트 조회 할 수 있도록 만들었습니다.

댓글 등록 및 삭제

```java
// 댓글 등록하기
    @PostMapping("/commentInsert")
    @ResponseBody
    public int commentInsert(@ModelAttribute CommentDTO comment){

        log.info("");
        log.info("");
        log.info("[commentInsert]" + comment);

        int result = commentService.commentInsert(comment);

        return result;
    }

// 댓글 삭제
    @PostMapping("/comments/{commentNo}")
    @ResponseBody
    public int commentDelete(@PathVariable int commentNo){

        log.info("");
        log.info("");
        log.info("[commentDelete]" + commentNo);

        int result = commentService.commentDelete(commentNo);

        return result;
    }
```

```javascript
<script th:inline="javascript">
    /*<![CDATA[*/
    let loginWriter = $('#writer').val();

$('[name=commentInsertBtn]').click(function(){ // 댓글 등록 버튼 클릭시

    let insertData = $('[name=commentInsertForm]').serialize(); // commentInsertForm의 내용을 가져옴
    commentInsert(insertData);
});

// 댓글 등록
function commentInsert(insertData) {
    $.ajax({
        url: '/member/commentInsert',
        type: 'POST',
        data: insertData,

        success: function() {
            commentList();
            $('.content').val(''); // 댓글 등록 성공시 foam 내용 초기화
        },

        error: function(error){
            console.log(error);
        }
    });
}

$(document).ready(function(){
   commentList(); // 페이지 로딩시 댓글 목록 출력
});

// 댓글 삭제
function commentDelete(commentNo, writer){

    if(loginWriter == writer){

        $.ajax({

            url : "/member/comments/" + commentNo,
            type : "POST",
            success : function(){

                commentList();
            }
        });
    }

}
    /*]]>*/
</script>
```

```xml
<!-- 댓글 등록하기 -->
    <insert id="commentInsert">
        INSERT
          INTO POST_COMMENT
        (
          COMMENT_NO
        , CONTENT
        , WRITER
        , REG_DATE
        )
        VALUES
        (
          SEQ_COMMENT_NO.NEXTVAL
        , #{ content }
        , #{ writer }
        , TO_CHAR(SYSDATE, 'YYYY-MM-DD HH24:MI:SS')
        )
    </insert>

    <!-- 댓글 삭제 -->
    <delete id="commentDelete" parameterType="_int">
        DELETE
          FROM POST_COMMENT
         WHERE COMMENT_NO = #{ commentNo }
    </delete>
```

- Ajax를 사용하여 댓글 등록 및 삭제를 할 수 있도록 만들었습니다.