JSP 블로그 구현 포트폴리오

작성자 : 김혜련

이 프로젝트는 Spring을 사용하지 않고 직접 MVC 구조를 구현하여
유사 MVC 구조로 제작된 블로그 프로젝트 입니다.

전체적인 디자인은 HTML, CSS, JavaScript, Jquery를 사용했으며
게시물 작성, 수정은 TOAST UI Editor를 이용했습니다.

소스 코드 : https://github.com/khr777/myblog

프로젝트 위키 : https://to2.kr/byU

memome Service address : https://harry.ouo.nz

차례

개발환경

Front-End

- HTML
- CSS
- JavaScript
- jQuery 3.5
- Lodash 4.1
- Ajax(jQuery 활용)

Back-End

기술스택

- CentOS 7
- Tomcat 9
- JDK 1.8
- Servlet 4.0/JSP 3.1
- JDBC
- Spring 5 / Spring Boot 2

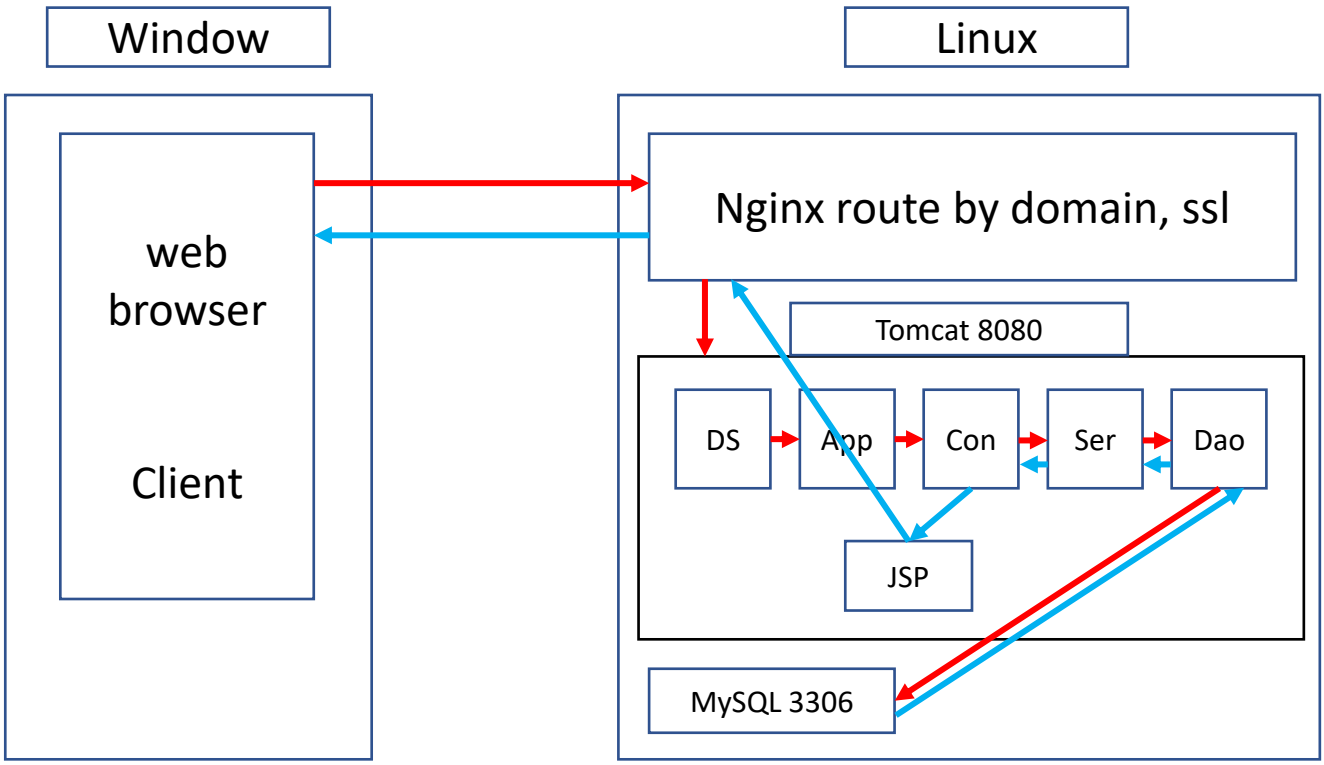
개발툴

- STS 4.4
- SQLYog Community Edition
- Maven
- Git, GitHUB
- Visual Studio Code
- Chrome
- Window 10
- Putty
- Filezilla

프로젝트 프레임워크

- 유사 MVC 구조
- 스프링을 사용하지 않고 직접 MVC 구조 만듬

동작과정



1. Client가 보낸 요청을 Nginx가 받아 기본 요청인 http 80을 proxy 설정을 통해 tomcat 8080으로 연결을 해줍니다.

2. Tomcat으로 요청이 들어오면 제일 먼저 DispatcherServlet에서 App을 실행시킵니다.

3. App에서는 먼저 제대로 된 db연결 정보인지 확인한 후 들어온 요청이 어떤 Controller의 요청인지를 확인하여 경로를 지정하고 Controller를 상속받은 각 Controller는 알맞은 액션이 실행되며 비즈니스 로직에 관련된 부분은 Service에 위임합니다.

4. Service에선 Controller에게 받은 데이터를 분석/처리해주고 데이터 보관, 수정, 삭제를 위해 DAO에게 위임합니다.

5. 계층간 데이터 교환에 DTO가 사용됩니다. (lombok을 사용하여 클래스에 getter/setter/생성자 등을 자동 생성해줍니다.)

6. DAO로 전달된 데이터는 Mybatis의 annotation /.xml 파일을 사용하여 객체들을 연결시킵니다.

7. .xml 파일에서 쿼리를 읽어 JDBC에 넘깁니다.

8. Service는 요청했던 데이터를 DAO에게 받아 성공과 실패를 구분해 Controller에게 보내줍니다.

9. Controller는 상황에 맞는 메시지를 Service에게 받아와 JSP를 통해 사용자에게 보여줍니다.

```java
@WebServlet("/s/*")
public class DispatcherServlet extends HttpServlet {


    protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        req.setCharacterEncoding("UTF-8");

        new App(req, resp).start();


    }


    protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
        doGet(req, resp);

    }

}
```

DispatcherServlet

```java
public class App {
    private HttpServletRequest req;
    private HttpServletResponse resp;
    private boolean isDevelServer = true;

    public App(HttpServletRequest req, HttpServletResponse resp) {
        this.req = req;
        this.resp = resp;

        String profilesActive = System.getProperty("spring.profiles.active");

        if ( profilesActive != null && profilesActive.equals(("production"))) {
            isDevelServer = false;
        }
    }
    private void loadDriver(HttpServletRequest req, HttpServletResponse resp) throws IOException  {


        // DB 커넥터 로딩 시작
        String driverName = "com.mysql.cj.jdbc.Driver";

        try {
            Class.forName(driverName);
        } catch (ClassNotFoundException e) {
            System.err.printf("[ClassNotFoundException 예외, %s]\n", e.getMessage());
            resp.getWriter().append("DB 드라이버 클래스 로딩 실패");

            return;
        }
        // DB 커넥터 로딩 성공
    }
    private String getDbUri() {


        if ( isDevelServer ) {
            return "jdbc:mysql://localhost:3306/harry.blog?serverTimezone=Asia/"
                    + "Seoul&useOldAliasMetadataBehavior=true&zeroDateTimeBehavior=convertToNull";
        }
        return "jdbc:mysql://localhost:3306/harry?serverTimezone=Asia/"
                + "Seoul&useOldAliasMetadataBehavior=true&zeroDateTimeBehavior=convertToNull";

    }
```

App

```
public void start() throws ServletException, IOException  {

    if (req.getServletContext().getInitParameter("gmailId") != null ) {
        Config.gmailId = (String)req.getServletContext().getInitParameter("gmailId");
    }

    if ( req.getServletContext().getInitParameter("gmailPw") != null ) {
        Config.gmailPw = (String)req.getServletContext().getInitParameter("gmailPw");
    }

    // [ DB드라이버 로딩 ]
    loadDriver(req, resp);

    // [ DB 접속정보 세팅 ]
    String Uri = getDbUri();
    String user = getId();
    String password = getDbPassword();

    Connection dbConn = null;

    try  {
        // = DB 접속 성공 =
        dbConn = DriverManager.getConnection(Uri, user, password);

        // [ 올바른 컨트롤러로 라우팅 ] : 올바른 길로 인도한다
        route(dbConn, req, resp);
    } catch (SQLException e) {
        Util.printEx("SQL 예외(커넥션 열기)", resp, e);
    } catch (SQLErrorException e) {
        Util.printEx(e.getMessage(), resp, e.getOrigin());
    } catch (Exception e) {
        Util.printEx("기타 예외", resp, e);
    }
    finally {
        if ( dbConn != null ) {
            try {
                dbConn.close();
            } catch (SQLException e) {
                Util.printEx("SQL 예외(커넥션 닫기)", resp, e);
            }
        }
    }
```

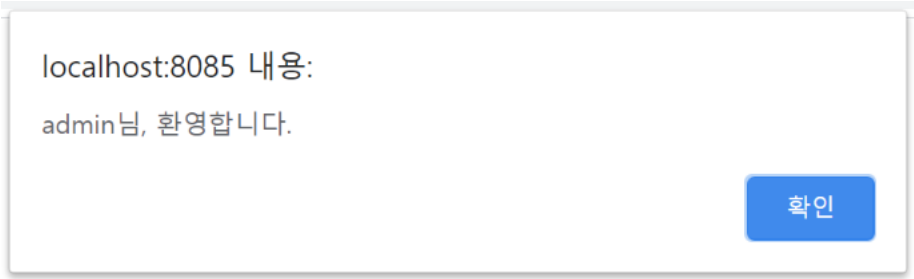
```
private void route(Connection dbConn, HttpServletRequest req, HttpServletResponse resp)
        throws IOException, ServletException {
    resp.setContentType("text/html; charset=UTF-8");
    req.setCharacterEncoding("UTF-8");
    String contextPath = req.getContextPath();
    String requestURI = req.getRequestURI();
    String actionStr = requestURI.replace(contextPath + "/s/", "");
    String[] actionStrBits = actionStr.split("/");
    String controllerName = actionStrBits[0];
    String actionMethodName = actionStrBits[1];
    Controller controller = null;
    switch (controllerName) {
    case "article":
        controller = new ArticleController(dbConn, actionMethodName, req, resp);
        break;
    case "member":
        controller = new MemberController(dbConn, actionMethodName, req, resp);
        break;
    case "home":
        controller = new HomeController(dbConn, actionMethodName, req, resp);
        break;
    case "test":
        controller = new TestController(dbConn, actionMethodName, req, resp);
    }
    if (controller != null) {
        String actionResult = controller.executeAction();
        if ( actionResult.equals("")) {
            resp.getWriter().append("액션의 결과가 없습니다.");
        } else if ( actionResult.endsWith(".jsp")) {
            String  viewPath = "/jsp/" + actionResult;
            req.getRequestDispatcher(viewPath).forward(req, resp);
        } else if ( actionResult.startsWith("html:")) {
            resp.getWriter().append(actionResult.substring(5));
        } else if ( actionResult.startsWith("json:")) {
            resp.setContentType("application/json; charset=UTF-8");
            resp.getWriter().append(actionResult.substring(5));
        } else {
            resp.getWriter().append("처리할 수 없는 액션 결과입니다.");
        }
    } else {
        resp.getWriter().append("존재하지 않는 페이지 입니다.");
    }
}
```

회원관리

회원가입


로그인 아이디
로그인 아이디를 입력해주세요.
로그인 비밀번호
로그인 비밀번호를 입력해주세요.
로그인
취소
회원가입
아이디 찾기
비밀번호 찾기
harry.ouo.nz


로그인 아이디
admin
사용할 수 있는 아이디 입니다.
이름
admin
닉네임
admin
로그인 비밀번호
로그인 비밀번호 확인
이메일
admin@admin.com
취소
회원가입 정보 제출
Welcome to my blog!


localhost:8085 내용:
admin님, 환영합니다.
확인

회원가입

```java
@Override
public String doAction() {
    switch (actionMethodName) {
    case "join":
        return actionJoin();
    case "doJoin":
        return actionDoJoin();
```

MemberController

```java
private String actionJoin() {
    return "member/join.jsp";
}

private String actionDoJoin() {
    String loginId = req.getParameter("loginId");
    String name = req.getParameter("name");
    String nickName = req.getParameter("nickname");
    String loginPw = req.getParameter("loginPwReal");
    String email = req.getParameter("email");

    boolean isJoinableLoginId = memberService.isJoinableLoginId(loginId);

    if (isJoinableLoginId == false) {
        return String.format("html:<script> alert('%s은(는) 이미 사용중인 아이디 입니다.'); history.back(); </script>", loginId);
    }

    boolean isJoinableNickName = memberService.isJoinableNickName(nickName);

    if (isJoinableNickName == false) {
        return String.format("html:<script> alert('%s은(는) 이미 사용중인 닉네임 입니다.'); history.back(); </script>", nickName);
    }

    boolean isJoinableEmail = memberService.isJoinableEmail(email);

    if (isJoinableEmail == false) {
        return String.format("html:<script> alert('%s은(는) 이미 사용중인 이메일 입니다.'); history.back(); </script>", email);
    }

    String ENGLISH_LOWER = "abcdefghijklmnopqrstuvwxyz";
    String ENGLISH_UPPER = ENGLISH_LOWER.toUpperCase();
    String NUMBER = "0123456789";
    // 랜덤을 생성할 대상 문자열
    String DATA_FOR_RANDOM_STRING = ENGLISH_LOWER + ENGLISH_UPPER + NUMBER;
    // 랜덤 문자열 길이
    int random_string_length = 10;
    String authCode = generate(DATA_FOR_RANDOM_STRING, random_string_length);
    int id = memberService.join(loginId, name, nickName, loginPw, email, authCode);
    return String.format("html:<script> alert('%s님, 환영합니다.'); location.replace('../home/main'); </script>", name);
}
```

MemberController

회원가입

```
public int join(String loginId, String name, String nickName, String loginPw, String email, String authCode) {
    int id = memberDao.join(loginId, name, nickName, loginPw, email, authCode);

    String emailTitle = "harry's life 회원가입을 축하드립니다. 이메일 인증 후 활동해주세요.";
    String emailBody = "";
    emailBody += "<h1> 환영합니다. 회원님 ^^</h1><br>";
    emailBody += "<h2>테스트 중입니다. 회원님????</h2><br>";
    emailBody += "<h3>아래 '인증하기' 버튼을 클릭한 후 회원활동을 하실 수 있습니다.</h3><br>";
    emailBody += "<html><body><h4><a href=" + "https://harry.ouo.nz/blog/s/member/";
    emailBody += "doAuthEmail?email=" + email + "&authCode=" + authCode + "&memberId=" + id
            + "> 인증하기</a></h4></body></html>";

    attrService.setValue("member__" + id + "__extra__emailAuthCode", authCode);

    mailService.send(email, emailTitle, emailBody);

    return id;
}
```

MemberService

```
public int join(String loginId, String name, String nickName, String loginPw, String email, String authCode) {
    SecSql secSql = new SecSql();

    secSql.append("INSERT INTO `member` ");
    secSql.append("SET regDate = NOW() ");
    secSql.append(", updateDate = NOW() ");
    secSql.append(", loginId = ? ", loginId);
    secSql.append(", name = ? ", name);
    secSql.append(", nickname = ?", nickName);
    secSql.append(", loginPw = ?", loginPw);
    secSql.append(", email = ?", email);
    secSql.append(", mailAuthCode =?", authCode);
    secSql.append(", mailAuthStatus = 0");
    return DBUtil.insert(dbConn, secSql);

}
```

MemberDao

회원가입

```
public boolean isJoinableLoginId(String loginId) {
    return memberDao.isJoinableLoginId(loginId);
}

public boolean isJoinableNickName(String nickName) {
    return memberDao.isJoinableNickName(nickName);
}
```

MemberService

```
public boolean isJoinableLoginId(String loginId) {
    SecSql sql = SecSql.from("SELECT COUNT(*) AS cnt");
    sql.append("FROM `member`");
    sql.append("WHERE loginId = ?", loginId);


    return DBUtil.selectRowIntValue(dbConn, sql) == 0 ;
}

public boolean isJoinableNickName(String nickName) {
    SecSql sql = SecSql.from("SELECT COUNT(*) AS cnt");
    sql.append("FROM `member`");
    sql.append("WHERE nickname = ?", nickName);


    return DBUtil.selectRowIntValue(dbConn, sql) == 0 ;
}
```

MemberDao

- isJoinableLoginId, NickName 메서드를 통해서 이미 가입 이력이 있는 일치하는 정보가 있는지를 MemberDao가 DB에서 찾아 MemberService에 전달을 해줍니다.
- MemberService가 가입 가능 여부를 MemberController에게 전달해주면 MemberContoller는 가입을 받거나 거절합니다.

회원가입

```java
// 회원가입
@RequestMapping("/usr/member/join")
public String showMain() {
    return "member/join";
}

// 회원가입
@RequestMapping("/usr/member/doJoin")
public String doJoin(@RequestParam Map<String, Object> param, Model model) {

    Util.changeMapKey(param, "loginPwReal", "loginPw");

    int newMemberId = memberService.join(param);

    String nickname = Util.getAsStr(param.get("nickname"));
    model.addAttribute("alertMsg", nickname + " 님, 회원가입을 감사드립니다.");

    String redirectUri = (String) param.get("redirectUri");
    model.addAttribute("redirectUri", redirectUri);

    return "common/redirect";
}
```

MemberController

```java
// 회원가입
public int join(Map<String, Object> param) {
    memberDao.join(param);
    int id = Util.getAsInt(param.get("id"));
    attrService.setValue("member", id, "extra", "lastPasswordUpdateDate", "1", null);
    sendJoinCompleteMail((String) param.get("email"));

    return id;
}
```

MemberService

로그인

```
@Override
public String doAction() {
    switch (actionMethodName) {
    case "join":
        return actionJoin();
    case "doJoin":
        return actionDoJoin();
    case "login":
        return actionLogin();
    case "doLogin":
        return actionDoLogin();
```

MemberController

```
private String actionLogin() {

    return "member/login.jsp";
}

private String actionDoLogin() {

    String loginId = req.getParameter("loginId");
    String loginPw = req.getParameter("loginPwReal");
    int loginedMemberId = memberService.getMemberIdByLoginIdAndLoginPw(loginId, loginPw);

    if (loginedMemberId == -1) {
        return "html:<script> alert('일치하는 정보가 없습니다.'); history.back(); </script>";
    }

    Member member = memberService.getMemberById(loginedMemberId);

    String emailAuthed = attrService.getValue("member__" + loginedMemberId + "__extra__emailAuthed");

    if (loginedMemberId != -1 && emailAuthed.length() == 0) {
        emailAuthed = member.getEmail();
        return "html:<script> alert('이메일 미인증 회원으로 인증 후 이용해주세요.'); location.replace('../member/emailAuthed?id=" +
                loginedMemberId + "'); </script>";
    }
    session.setAttribute("loginedMemberId", loginedMemberId); // 최초 키값을 설정하는 코드(개별 저장소 생성)


    boolean isNeedToChangePasswordForTemp = memberService.isNeedToChangePasswordForTemp(loginedMemberId);


    String redirectUri = Util.getString(req, "redirectUri", "../home/main");

    req.setAttribute("jsAlertMsg", "로그인 되었습니다.");

    if ( isNeedToChangePasswordForTemp ) {
        req.setAttribute("jsAlertMsg2", "현재 임시패스워드를 사용중입니다. 비밀번호를 변경해주세요.");
    }

    req.setAttribute("redirectUri", redirectUri);

    return "common/data.jsp";
}
```

MemberController

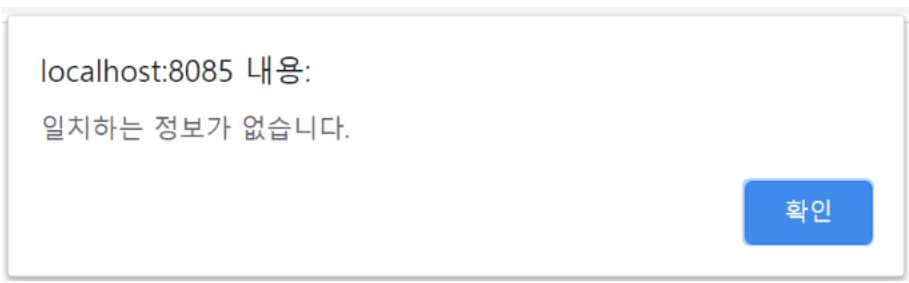
로그인

```
public int getMemberIdByLoginIdAndLoginPw(String loginId, String loginPw) {
    return memberDao.getMemberIdByLoginIdAndLoginPw(loginId, loginPw);
}
```

MemberService

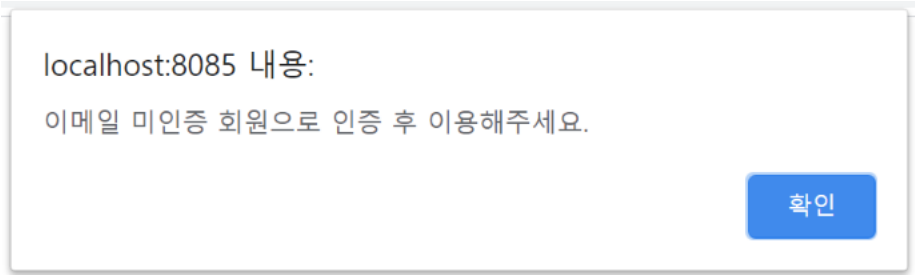
```
public Member getMemberFromMemberId(int loginedMemberId) {
    SecSql sql = SecSql.from("SELECT *");
    sql.append("FROM `member`");
    sql.append("WHERE id = ?", loginedMemberId);
     return new Member(DBUtil.selectRow(dbConn, sql));
}
```

MemberDao



- 일치하는 회원정보가 존재하지 않아 로그인 실패를 알려주는 화면입니다.

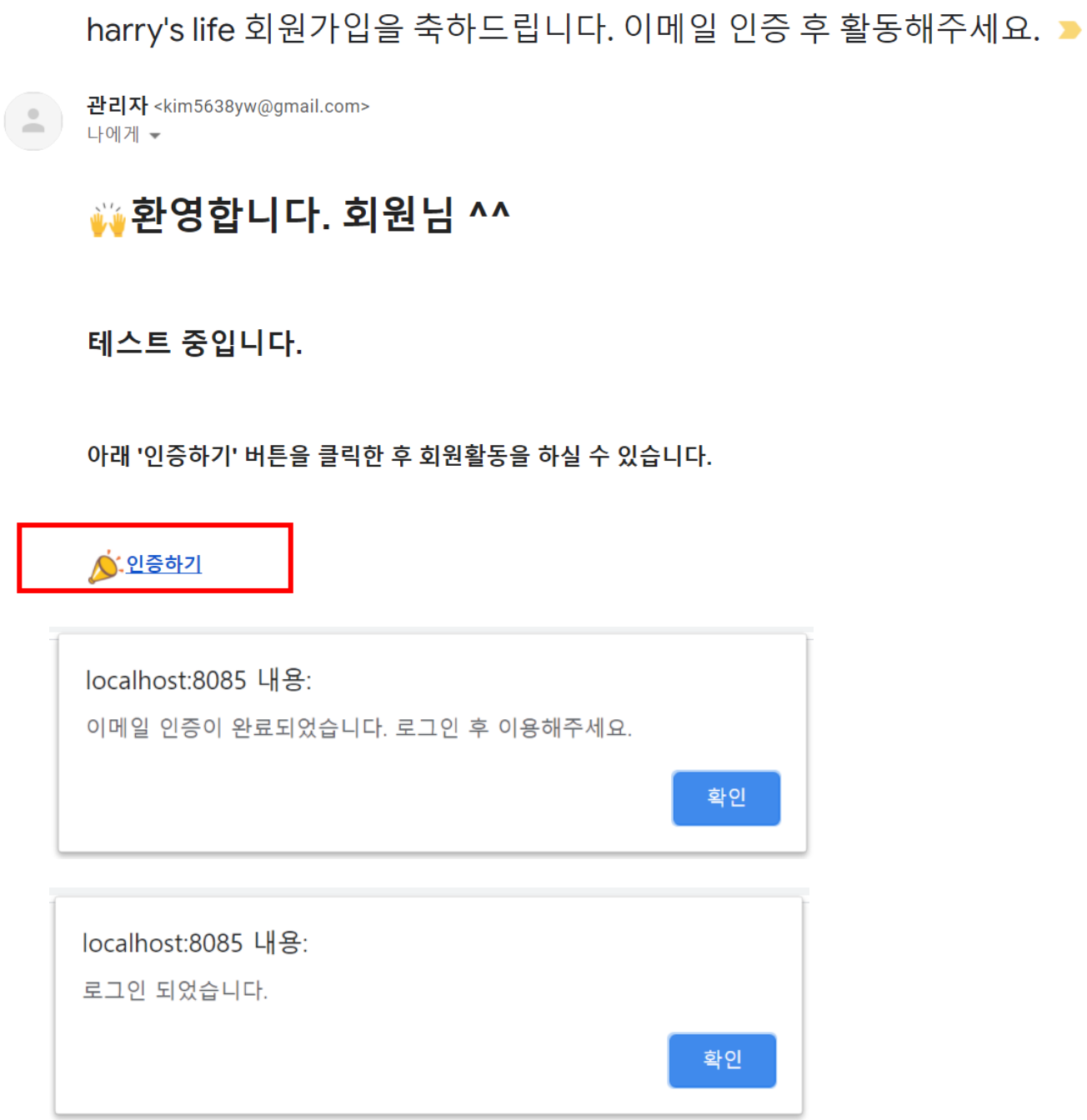
로그인



- 처음 회원가입 후 이메일 인증을 하지 않은 채로 로그인을 할 경우, 이메일 인증을 권유하는 알림과 '확인' 을 클릭하면 이메일 인증을 해달라는 알림을 한번 더 페이지에서 보여줍니다.

- 이메일 인증을 하지 않은 경우 로그인 및 로그인이 필요한 활동을 할 수 없습니다.

로그인



- 인증 이메일을 받아 '인증하기' 버튼을 클릭하면 인증완료 알림과 로그인 페이지로 이동합니다.

로그아웃

```
case "doLogout":
    return actionDoLogout();
```

MemberController

```
private String actionDoLogout() {

    HttpSession session = req.getSession();
    int loginedMemberId = 0;

    if (session.getAttribute("loginedMemberId") != null) {
        loginedMemberId = (int) session.getAttribute("loginedMemberId");
    }

    session.removeAttribute("loginedMemberId");

    String redirectUri = Util.getString(req, "redirectUri", "../home/main");

    return String.format("html:<script> alert('로그아웃 되었습니다.'); location.replace('" + redirectUri + "'); </script>");
}
```

MemberController

- 현재 session에 저장된(접속된) 회원이 정보를 얻어 remove 합니다.

회원정보 변경

```
private String actionMyPage() {
    return "member/myPage.jsp";
}


private String actionMemberDataForPrivate() {

    return "member/memberDataForPrivate.jsp";

}
```

MemberController

- myPage에서 회원정보 변경 또는 비밀번호 변경을 클릭하면 페이지 이동하기 전에 ‘로그인 비밀번호‘ 확인을 입력해야 합니다.

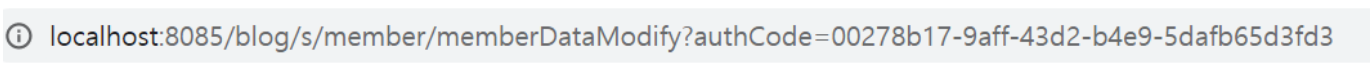
회원정보 변경

```
private String actionDoMemberDataForPrivate() {
    String loginPw = req.getParameter("loginPwReal");

    Member loginedMember = (Member) req.getAttribute("loginedMember");
    int loginedMemberId = (int) req.getAttribute("loginedMemberId");

    if (loginedMember.getLoginPw().equals(loginPw)) {
        // 감히 controller가 직접 authCode를 만들 수 없다. 데이터를 구워달라고 service한테 부탁해야 한다.
        String authCode = memberService.genModifyPrivateAuthCode(loginedMemberId);

        return String
                .format("html:<script> location.replace('memberDataModify?authCode=" + authCode + "'); </script>");
    }

    return String.format("html:<script> alert('비밀번호를 다시 입력해주세요.'); history.back(); </script>");
}
```

MemberController

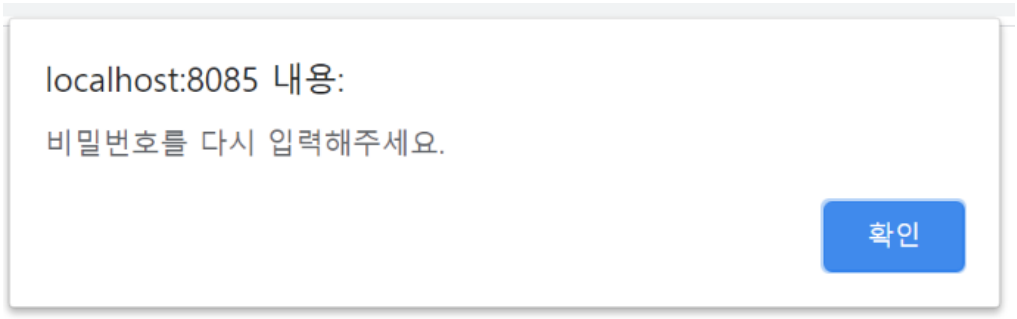
```
public String getModifyPrivateAuthCode(int actorId) {
    String authCode = UUID.randomUUID().toString();

    attrService.setValue("member__" + actorId + "__extra__modifyPrivateAuthCode", authCode);

    return authCode;

}
```

MemberService



- '회원정보 변경' 페이지로 이동할 때의 uri로 MemberService가 생성해준 authCode를 가지고 있습니다.

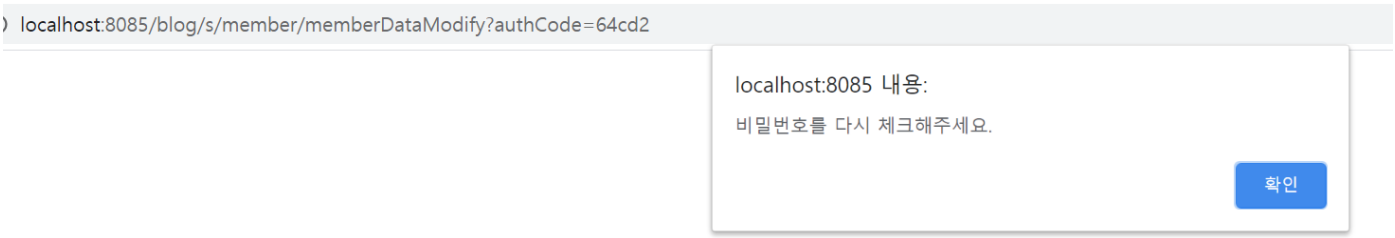
회원정보 변경



- 회원정보를 변경하기 위해 현재 비밀번호를 입력했을 때 패스워드가 일치 하지 않는 경우 알림

- MemberService가 생성해준 authCode를 제대로 가지고 있을 때.

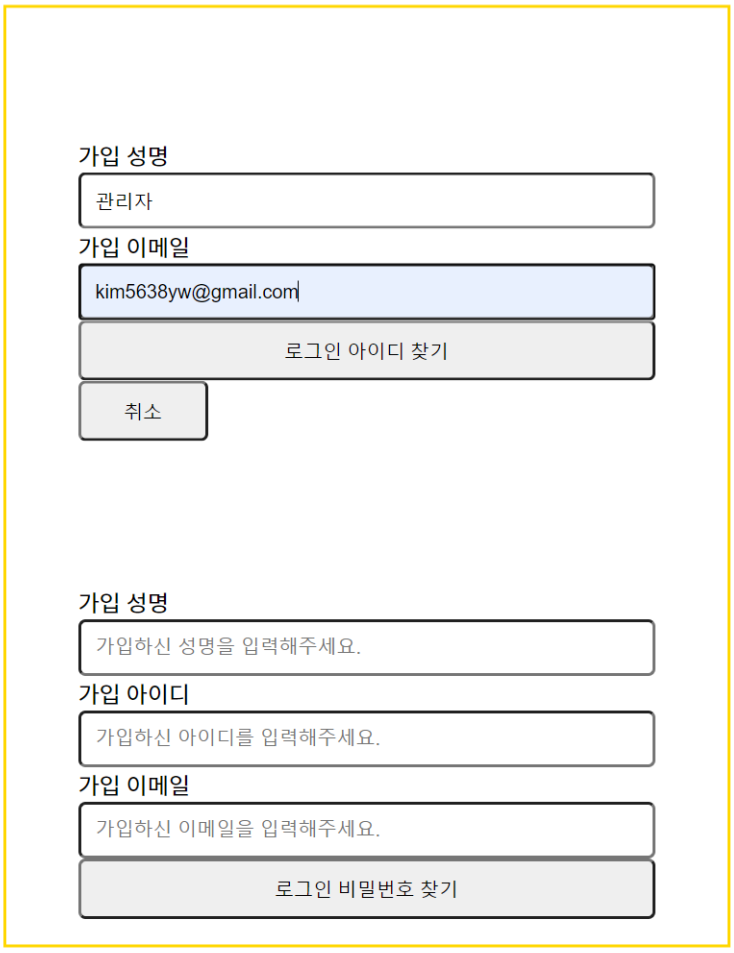
회원정보 변경



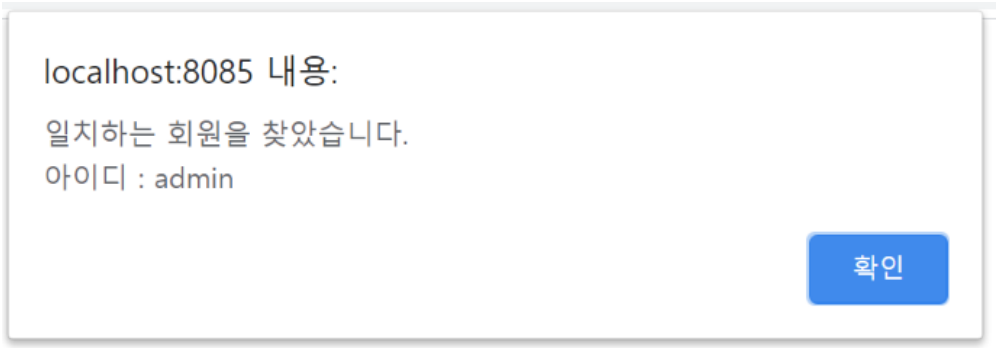
- 회원정보 변경을 위하여 현재 패스워드를 입력할 때 부여 받고 attr에 저장해 놓은 authCode가 일치하지 않은 채로 페이지에 접속하게 되면 비밀번호를 다시 입력해달라는 경고와 함께 패스워드 입력 화면으로 이동합니다.

- 패스워드 변경 상기 동일

로그인 아이디 찾기



harry.ouo.nz



로그인 비밀번호 찾기

```
case "findAccount":
    return actionFindAccount();



private String actionFindAccount() {
    return "member/findAccount.jsp";
}



case "doFindLoginId":
    return actionDoFindLoginId();
case "doFindLoginPw":
    return actionDoFindLoginPw();
```

```
 private String actionDoFindLoginId() {

     String name = Util.getString(req, "name");
     String email = Util.getString(req, "email");



     Member member = memberService.getMemberByNameAndEmail(name, email);

     if ( member == null ) {
         req.setAttribute("jsAlertMsg", "일치하는 회원이 없습니다.");
         req.setAttribute("jsHistoryBack", true);
         return "common/data.jsp";
     }


     req.setAttribute("jsAlertMsg", "일치하는 회원을 찾았습니다.\\n아이디 : " + member.getLoginId());
     req.setAttribute("jsHistoryBack", true);
     return "common/data.jsp";

 }
```

MemberController

로그인 비밀번호 찾기

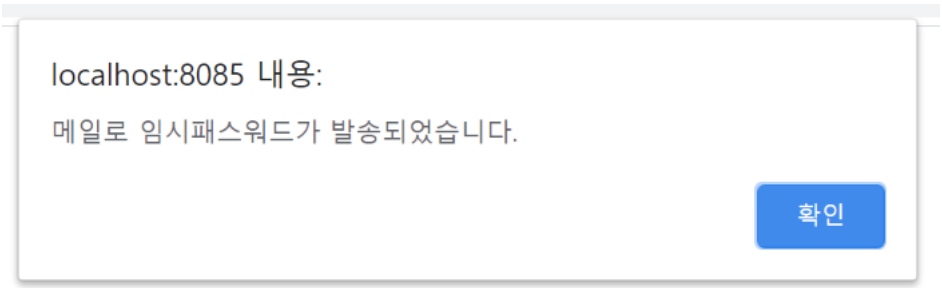
```
public Member getMemberByNameAndEmail(String name, String email) {
    return memberDao.getMemberByNameAndEmail(name, email);
}
```

MemberService

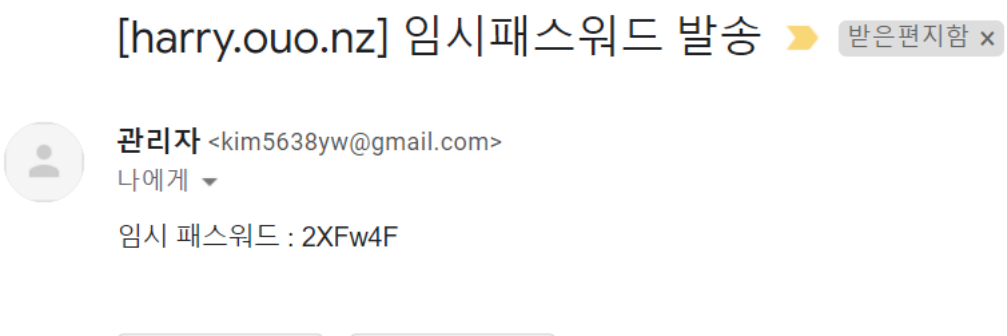
```
public Member getMemberByNameAndEmail(String name, String email) {
    SecSql sql = SecSql.from("SELECT *");
    sql.append("FROM `member`");
    sql.append("WHERE name = ? ",name);
    sql.append("AND email = ?", email);

    Map<String, Object> row = DBUtil.selectRow(dbConn, sql);

    if ( row.isEmpty()) {
        return null;
    }

    return new Member(row);
}
```

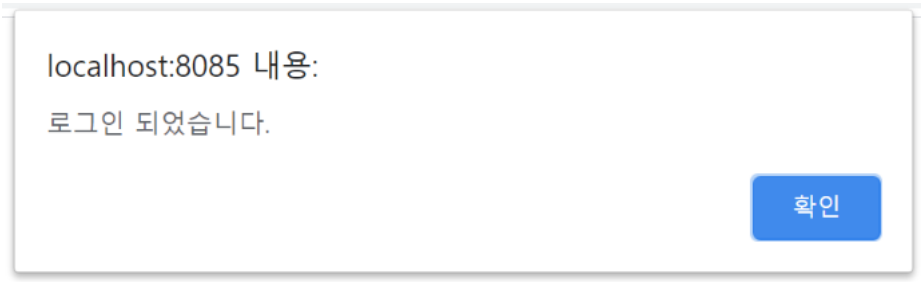
MemberDao

로그인 비밀번호 찾기



- 비밀번호 찾기 정보를 입력한 후 '확인' 버튼을 누르면 이메일로 임시 패스워드가 발송됩니다.





- 메일로 받은 임시 패스워드로 로그인을 하면 접속이 가능합니다.

로그인 비밀번호 찾기

```
    private String actionDoFindLoginPw() {

        String name = Util.getString(req, "name");
        String loginId = Util.getString(req, "loginId");
        String email = Util.getString(req, "email");


        Member member = memberService.getMemberByLoginId(loginId);

        if ( member == null || member.getEmail().equals(email) == false ) {
            req.setAttribute("jsAlertMsg", "일치하는 회원이 없습니다.");
            req.setAttribute("jsHistoryBack", true);
            return "common/data.jsp";
        }


        memberService.notifyTempLoginPw(member);

        req.setAttribute("jsAlertMsg", "메일로 임시패스워드가 발송되었습니다.");
        req.setAttribute("redirectUri", "../member/login");
        return "common/data.jsp";
    }
```

MemberController

```
public void notifyTempLoginPw(Member member) {

    String to = member.getEmail();
    String tempPasswordOrigin = Util.getTempPassword(6);
    String tempPassword = Util.sha256(tempPasswordOrigin);

    memberDao.modify(member.getId(), tempPassword);
    attrService.setValue("member", member.getId(), "extra", "useTempPassword", "1");

    String title = String.format("[%s] 임시패스워드 발송", Config.getSiteName());
    String body = String.format("<div>임시 패스워드 : %s</div>\n", tempPasswordOrigin);
    mailService.send(to, title, body);
}
```

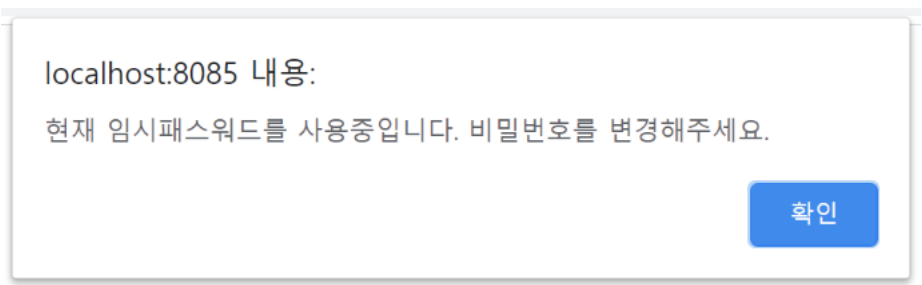
MemberService

로그인 비밀번호 찾기

```
public void modify(int actorId, String loginPw) {

    SecSql sql = SecSql.from("UPDATE `member`");
    sql.append("SET updateDate = NOW()");
    sql.append(", loginPw = ?", loginPw);
    sql.append("WHERE id = ? ", actorId);

    DBUtil.update(dbConn, sql);
}
```

MemberDao

임시 패스워드 사용 여부 알림



- 임시 패스워드를 사용하고 있다면 로그인 시, 변경 권유를 합니다.

임시 패스워드 사용 여부 알림

```
private String actionDoLogin() {

    String loginId = req.getParameter("loginId");
    String loginPw = req.getParameter("loginPwReal");
    int loginedMemberId = memberService.getMemberIdByLoginIdAndLoginPw(loginId, loginPw);

    if (loginedMemberId == -1) {
        return "html:<script> alert('일치하는 정보가 없습니다.'); history.back(); </script>";
    }

    Member member = memberService.getMemberById(loginedMemberId);

    String emailAuthed = attrService.getValue("member__" + loginedMemberId + "__extra__emailAuthed");

    if (loginedMemberId != -1 && emailAuthed.length() == 0) {
        emailAuthed = member.getEmail();
        return "html:<script> alert('이메일 미인증 회원으로 인증 후 이용해주세요.'); location.replace('../member/emailAuthed?id=" +
                loginedMemberId + "'); </script>";
    }
    session.setAttribute("loginedMemberId", loginedMemberId); // 최초 키값을 설정하는 코드(개별 저장소 생성)

    boolean isNeedToChangePasswordForTemp = memberService.isNeedToChangePasswordForTemp(loginedMemberId);

    String redirectUri = Util.getString(req, "redirectUri", "../home/main");

    req.setAttribute("jsAlertMsg", "로그인 되었습니다.");

    if ( isNeedToChangePasswordForTemp ) {
        req.setAttribute("jsAlertMsg2", "현재 임시패스워드를 사용중입니다. 비밀번호를 변경해주세요.");
    }

    req.setAttribute("redirectUri", redirectUri);

    return "common/data.jsp";
}
```

MemberController

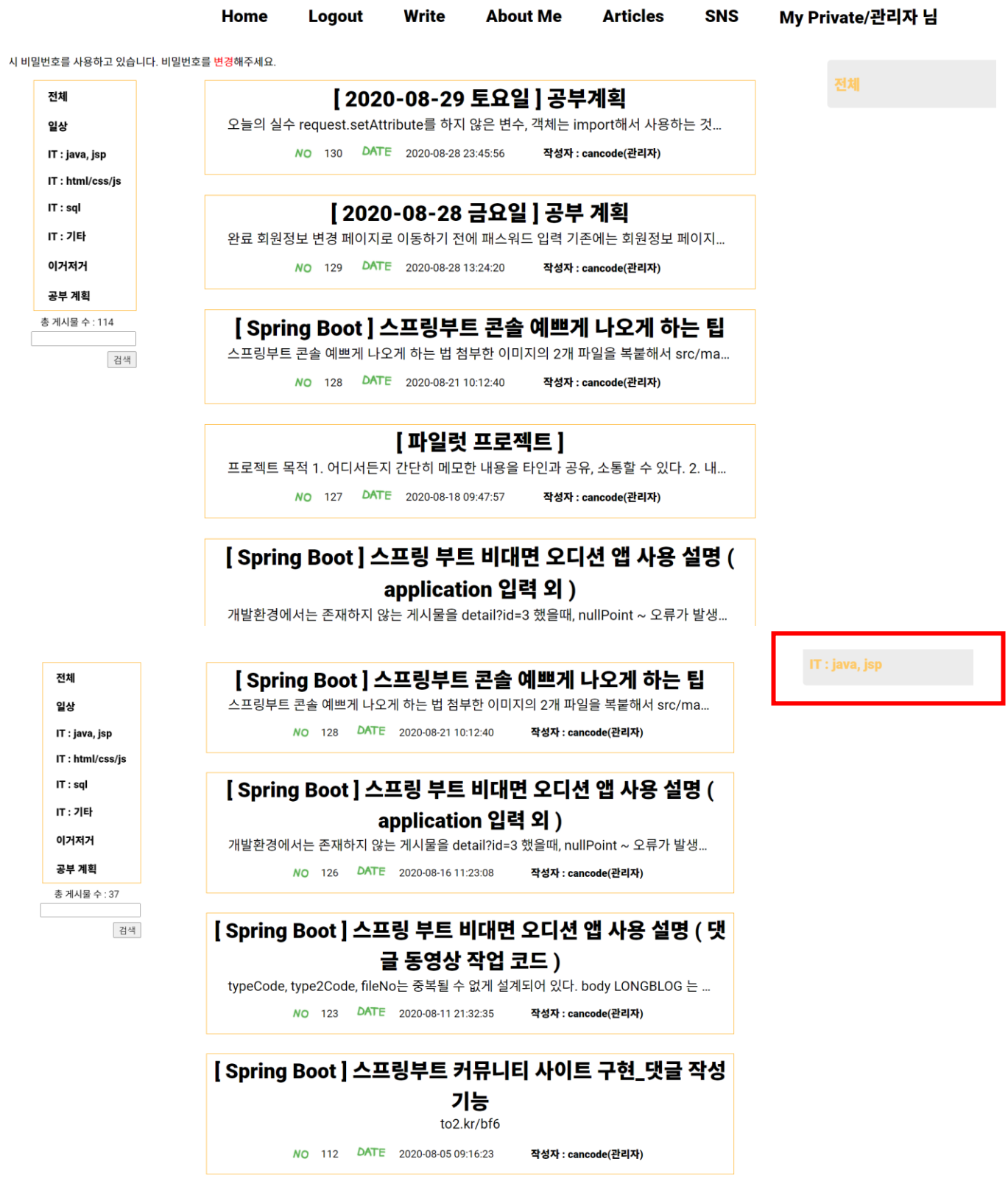
```
public boolean isNeedToChangePasswordForTemp(int actorId) {
    return attrService.getValue("member", actorId, "extra", "useTempPassword").equals("1");
}
```

MemberService

- 비밀번호 찾기를 할 때, attr에 저장해둔 값을 얻어 임시 패스워드 사용 여부를 판별합니다.
- MemberService는 그 결과를 MemberController한테 전달하고 MemberController가 로그인을 받아들일 때 사용자에게 결과를 보여줍니다.

게시판

멀티게시판


Home
Logout
Write
About Me
Articles
SNS
My Private/관리자 님
시 비밀번호를 사용하고 있습니다. 비밀번호를 변경해주세요.
전체
일상
IT : java, jsp
IT : html/css/js
IT : sql
IT : 기타
이거저거
공부 계획
총 게시물 수 : 114
검색
[2020-08-29 토요일] 공부계획
오늘의 실수 request.setAttribute를 하지 않은 변수, 객체는 import해서 사용하는 것...
NO 130 DATE 2020-08-28 23:45:56 작성자 : cancode(관리자)
[2020-08-28 금요일] 공부 계획
완료 회원정보 변경 페이지로 이동하기 전에 패스워드 입력 기존에는 회원정보 페이지...
NO 129 DATE 2020-08-28 13:24:20 작성자 : cancode(관리자)
[Spring Boot] 스프링부트 콘솔 예쁘게 나오게 하는 팁
스프링부트 콘솔 예쁘게 나오게 하는 법 첨부한 이미지의 2개 파일을 복붙해서 src/ma...
NO 128 DATE 2020-08-21 10:12:40 작성자 : cancode(관리자)
[파일럿 프로젝트]
프로젝트 목적 1. 어디서든지 간단히 메모한 내용을 타인과 공유, 소통할 수 있다. 2. 내...
NO 127 DATE 2020-08-18 09:47:57 작성자 : cancode(관리자)
[Spring Boot] 스프링 부트 비대면 오디션 앱 사용 설명 (application 입력 외)
개발환경에서는 존재하지 않는 게시물을 detail?id=3 했을때, nullPoint ~ 오류가 발생...
전체
IT : java, jsp
전체
일상
IT : java, jsp
IT : html/css/js
IT : sql
IT : 기타
이거저거
공부 계획
총 게시물 수 : 37
검색
[Spring Boot] 스프링부트 콘솔 예쁘게 나오게 하는 팁
스프링부트 콘솔 예쁘게 나오게 하는 법 첨부한 이미지의 2개 파일을 복붙해서 src/ma...
NO 128 DATE 2020-08-21 10:12:40 작성자 : cancode(관리자)
[Spring Boot] 스프링 부트 비대면 오디션 앱 사용 설명 (application 입력 외)
개발환경에서는 존재하지 않는 게시물을 detail?id=3 했을때, nullPoint ~ 오류가 발생...
NO 126 DATE 2020-08-16 11:23:08 작성자 : cancode(관리자)
[Spring Boot] 스프링 부트 비대면 오디션 앱 사용 설명 (댓글 동영상 작업 코드)
typeCode, type2Code, fileNo는 중복될 수 없게 설계되어 있다. body LONGBLOG 는 ...
NO 123 DATE 2020-08-11 21:32:35 작성자 : cancode(관리자)
[Spring Boot] 스프링부트 커뮤니티 사이트 구현_댓글 작성 기능
to2.kr/bf6
NO 112 DATE 2020-08-05 09:16:23 작성자 : cancode(관리자)

멀티게시판, 카테고리별 게시물 리스팅, 제목&내용 검색

```
public String doAction() {
    switch (actionMethodName) {
    case "list":
        return actionList();
```

```
private String actionList() {

    int page = 1;

    if (!Util.empty(req, "page") && Util.isNum(req, "page")) {
        page = Util.getInt(req, "page");
    }

    int cateItemId = 0;

    if (!Util.empty(req, "cateItemId") && Util.isNum(req, "cateItemId")) { // cateItemId가 없지 않고 숫자가 맞으면

        cateItemId = Util.getInt(req, "cateItemId");

    }

    String cateItemName = "전체";

    if (cateItemId != 0) {
        CateItem cateItem = articleService.getCateItem(cateItemId);
        cateItemName = cateItem.getName();
    }
    req.setAttribute("cateItemName", cateItemName);

    String searchKeywordType = ""; // keywordType이 더 중요하니까 searchkeyword보다 위에 써주는게 낫다.

    if (!Util.empty(req, "searchKeywordType")) { // cateItemId가 없지 않고 숫자가 맞으면

        searchKeywordType = Util.getString(req, "searchKeywordType");

    }

    String searchKeywordTypeBody = "";

    if (!Util.empty(req, "searchKeywordTypeBody")) { // cateItemId가 없지 않고 숫자가 맞으면

        searchKeywordTypeBody = Util.getString(req, "searchKeywordTypeBody");

    }

    String searchKeyword = "";
```

ArticleController

멀티게시판, 카테고리별 게시물 리스팅, 제목&내용 검색

```
    String searchKeyword = "";

    if (!Util.empty(req, "searchKeyword")) { // cateItemId가 없지 않고 숫자가 맞으면

        searchKeyword = Util.getString(req, "searchKeyword");

    }

    int itemsInAPage = 10; // 게시물 리스트에 보여줄 게시물 개수
    int totalCount = articleService.getForPrintListArticlesCount(cateItemId, searchKeywordType,
            searchKeywordTypeBody, searchKeyword);
    int totalPage = (int) Math.ceil(totalCount / (double) itemsInAPage);

    req.setAttribute("totalCount", totalCount);
    req.setAttribute("totalPage", totalPage);
    req.setAttribute("cPage", page);

    int loginedMemberId = (int) req.getAttribute("loginedMemberId");

    List<Article> articles = articleService.getForPrintListArticles(loginedMemberId, page, cateItemId, itemsInAPage,
            searchKeywordType, searchKeywordTypeBody, searchKeyword);
    req.setAttribute("articles", articles);

    return "article/list.jsp";
}
```

ArticleController

```
  public List<Article> getForPrintListArticles(int actorId, int page, int cateItemId, int itemsInAPage,
          String searchKeywordType, String searchKeywordTypeBody,  String searchKeyword) {
      List<Article> articles = articleDao.getForPrintListArticles(page, cateItemId,
              itemsInAPage, searchKeywordType, searchKeywordTypeBody, searchKeyword );

      for ( Article article : articles ) { // 꼬리표를 달아준다. 이 게시물을 이 활동자가 삭제할 수 있는지에 대한.
          updateArticleExtraDataForPrint(article, actorId);
      }

      return articles;
  }

   private void updateArticleExtraDataForPrint(Article article, int actorId) {
       boolean deleteAvailable = Util.isSuccess(getCheckRsDeleteAvailable(article, actorId));
       article.getExtra().put("deleteAvailable", deleteAvailable);


       boolean modifyAvailable = Util.isSuccess(getCheckRsModifyAvailable(article, actorId));
       article.getExtra().put("modifyAvailable", modifyAvailable);


   }
```

ArticleService

멀티게시판, 카테고리별 게시물 리스팅, 제목&내용 검색

```
public List<Article> getForPrintListArticles(int page, int cateItemId, int itemsInAPage, String searchKeywordType,
        String searchKeywordTypeBody, String searchKeyword) {

    int itemInAPage = 10;
    this.itemInAPage = itemInAPage;
    int limitFrom = (page - 1) * itemInAPage;
    SecSql sql = SecSql.from("SELECT A.*, M.nickname AS extra__writer");
    sql.append("FROM article AS A");
    sql.append("INNER JOIN `member` AS M");
    sql.append("ON A.memberId = M.id");
    sql.append("WHERE A.displayStatus = 1");
    if (cateItemId != 0) {
        sql.append("AND cateItemId = ?", cateItemId);
    }
    if (searchKeywordType.equals("title") && searchKeywordTypeBody.equals("body") && searchKeyword.length() > 0) {
        sql.append("AND title LIKE CONCAT('%', ? , '%')", searchKeyword);
        sql.append("OR body LIKE CONCAT('%', ? , '%')", searchKeyword);
    }
    sql.append("ORDER BY A.id DESC");
    sql.append("LIMIT ?, ? ", limitFrom, itemInAPage);

    List<Map<String, Object>> rows = DBUtil.selectRows(dbConn, sql);
    List<Article> articles = new ArrayList<>();

    for (Map<String, Object> row : rows) {
        articles.add(new Article(row));
    }

    return articles;
}
```

ArticleDao

- 기본적으로 카테고리 전체의 게시물을 리스팅 합니다.
- 입력 받은 카테고리가 있다면 해당 카테고리 번호로 작성된 게시물을 불러오는 구조입니다.

게시판 게시물(제목&내용 검색) 페이징

```java
public int getForPrintListArticlesCount(int cateItemId, String searchKeywordType, String searchKeywordTypeBody,  String searchKeyword) {
    return articleDao.getForPrintListArticlesCount(cateItemId, searchKeywordType, searchKeywordTypeBody, searchKeyword);
}
```

ArticleService

```java
public int getForPrintListArticlesCount(int cateItemId, String searchKeywordType, String searchKeywordTypeBody,
        String searchKeyword) {
    SecSql sql = new SecSql();

    sql.append("SELECT COUNT(*) AS cnt ");
    sql.append("FROM article ");
    sql.append("WHERE displayStatus = 1 ");
    if (cateItemId != 0) {
        sql.append("AND cateItemId = ? ", cateItemId);
    }

    if (searchKeywordType.equals("title") && searchKeywordTypeBody.equals("body") && searchKeyword.length() > 0) {
        sql.append("AND title LIKE CONCAT('%', ?, '%')", searchKeyword);
        sql.append("OR body LIKE CONCAT('%', ?, '%')", searchKeyword);
    }

    int count = DBUtil.selectRowIntValue(dbConn, sql);

    return count;

}
```

ArticleDao

- 게시물 리스팅을 할 때, 카테고리별로 게시물 전체 개수를 ArticleDao한테 전달을 받아 한 페이지에서 보여줄 게시물 개수를 구하여 ArticleController에서 JSP파일로 전달을 해줍니다.

- 제목&내용으로 검색을 하면 검색어를 받아 포함하는 게시물을 ArticleDao가 찾아서 ArticleService한테 전달을 해줍니다.

- ArticleService는 ArticleController에 전달을 해주며 ArticleController가 JSP로 해당 게시물들을 전달하여 사용자에게 보여줍니다.

게시물 작성

```
case "write":
    return actionWrite();
case "doWrite":
    return actionDoWrite();
```

```
private String actionWrite() {

    return "article/write.jsp";
}

private String actionDoWrite() {
    String title = req.getParameter("title");
    String body = req.getParameter("body");

    int cateItemId = Util.getInt(req, "cateItemId");
    int displayStatus = Util.getInt(req, "displayStatus");

    int loginedMemberId = (int) req.getAttribute("loginedMemberId");

    int id = articleService.write(cateItemId, displayStatus, title, body, loginedMemberId);

    return "html:<script> alert('" + id + "번 게시물이 생성되었습니다.'); location.replace('list'); </script>";

}
```

ArticleController

```
public int write( int cateItemId, int displayStatus, String title, String body, int memberId) {

    return articleDao.write( cateItemId, displayStatus,  title,  body, memberId);

}
```

ArticleService

```
public int write(int cateItemId, int displayStatus, String title, String body, int memberId) {
    SecSql secSql = new SecSql();

    secSql.append("INSERT INTO article");
    secSql.append("SET regDate = NOW()");
    secSql.append(", updateDate = NOW()");
    secSql.append(", title = ? ", title);
    secSql.append(", body = ? ", body);
    secSql.append(", displayStatus = ? ", displayStatus);
    secSql.append(", cateItemId = ?", cateItemId);
    secSql.append(", memberId = ?", memberId);

    return DBUtil.insert(dbConn, secSql);
}
```

ArticleDao

게시물 수정

```
case "modify":
    return actionModify();
case "doModify":
    return actionDoModify();


private String actionModify() {

    int id = 0;

    if (Util.empty(req, "id")) {
        return "html:id를 입력해주세요.";
    }
    if (Util.isNum(req, "id") == false) {
        return "html:id를 정수로 입력해주세요.";
    }

    id = Util.getInt(req, "id");

    int loginedMemberId = (int) req.getAttribute("loginedMemberId");

    if (!Util.empty(req, "id") && Util.isNum(req, "id")) { // 게시물 번호

        id = Util.getInt(req, "id");

    }

    Article article = articleService.getForPrintArticle(id, loginedMemberId);

    req.setAttribute("article", article);
    return "article/modify.jsp";
}
```

ArticleController

게시물 수정

```
private String actionDoModify() {

    if (Util.empty(req, "id")) {
        return "html:id를 입력해주세요.";
    }

    if (Util.isNum(req, "id") == false) {
        return "html:id를 정수로 입력해주세요.";
    }

    int id = Util.getInt(req, "id");

    int loginedMemberId = (int) req.getAttribute("loginedMemberId");

    Map<String, Object> getCheckRsModifyAvailableRs = articleService.getCheckRsModifyAvailable(id, loginedMemberId);

    if (Util.isSuccess(getCheckRsModifyAvailableRs) == false) {
        return "html:<script> alert('" + getCheckRsModifyAvailableRs.get("msg") + "'); history.back(); </script>";
    }

    String title = req.getParameter("title");
    String body = req.getParameter("body");
    int cateItemId = Util.getInt(req, "cateItemId");
    int displayStatus = Util.getInt(req, "displayStatus");

    articleService.modifyArticle(id, cateItemId, displayStatus, title, body);

    return "html:<script> alert('" + id + "번 게시물이 수정되었습니다.'); location.replace('detail?id=" + id + "'); </script>";

}
```

ArticleController

```
public void modifyArticle(int id, int cateItemId, int displayStatus, String title, String body) {
    articleDao.modifyArticle(id, cateItemId, displayStatus, title, body);
}
```

ArticleService

```
public void modifyArticle(int id, int cateItemId, int displayStatus, String title, String body) {

    SecSql sql = SecSql.from("UPDATE article");

    sql.append("SET updateDate = NOW()");
    sql.append(", cateItemId = ?", cateItemId);
    sql.append(", title = ?", title);
    sql.append(", body = ? ", body);
    sql.append(", displayStatus = ?", displayStatus);
    sql.append(" WHERE id = ?", id);

    DBUtil.update(dbConn, sql);

}
```

ArticleDao

게시물 수정

```
public Map<String, Object> getCheckRsDeleteAvailable(int id, int actorId) {
    Article article = articleDao.getForPrintArticle(id);

    return getCheckRsDeleteAvailable(article, actorId);
}

public Map<String, Object> getCheckRsModifyAvailable(int id, int actorId) {

    return getCheckRsDeleteAvailable(id, actorId);
}


private Map<String, Object> getCheckRsDeleteAvailable(Article article, int actorId) {
    Map<String, Object> rs = new HashMap<>();


    if ( article == null ) {
        rs.put("resultCode", "F-1");
        rs.put("msg", "존재하지 않는 게시물 입니다.");

        return rs;
    }

    if ( article.getMemberId() != actorId ) {
        rs.put("resultCode", "F-2");
        rs.put("msg", "게시물 삭제 권한이 없습니다.");

        return rs;
    }

    rs.put("resultCode", "S-1");
    rs.put("msg", "작업이 가능합니다.");

    return rs;
}
```

ArticleDao

- ArticleService에서 게시물 작성자와 현재 로그인한 회원 정보가 일치하는지 확인한 후 ArticleController에 게시물 수정, 삭제 권한을 알려줍니다.

게시물 삭제

```
case "doDelete":
    return actionDoDelete();
```

```
public void deleteArticle(int id) {
    articleDao.deleteArticle(id);
}
```

ArticleController

```
private String actionDoDelete() {

    if (Util.empty(req, "id")) {
        return "html:id를 입력해주세요.";
    }

    if (Util.isNum(req, "id") == false) {
        return "html:id를 정수로 입력해주세요.";
    }

    int id = Util.getInt(req, "id");

    int loginedMemberId = (int) req.getAttribute("loginedMemberId");

    Map<String, Object> getCheckRsDeleteAvailableRs = articleService.getCheckRsDeleteAvailable(id, loginedMemberId);

    if (Util.isSuccess(getCheckRsDeleteAvailableRs) == false) {
        return "html:<script> alert('" + getCheckRsDeleteAvailableRs.get("msg") + "'); history.back(); </script>";
    }

    articleService.deleteArticle(id);

    String redirectUri = Util.getString(req, "redirectUri", "list");

    return "html:<script> alert('" + id + "번 게시물이 삭제되었습니다.'); location.replace('" + redirectUri + "'); </script>";
}
```

ArticleService

```
public int deleteArticle(int id) {
    SecSql sql = SecSql.from("DELETE FROM article");

    sql.append("WHERE id = ? ", id);

    return DBUtil.delete(dbConn, sql);
}
```

ArticleDao

게시물 삭제

게시물 작성 테스트

번호 : 131
등록일 : 2020-10-03 12:14:43
수정일 : 2020-10-03 12:14:43

조회수 : 1
카테고리 : 일상
작성자 : 관리자

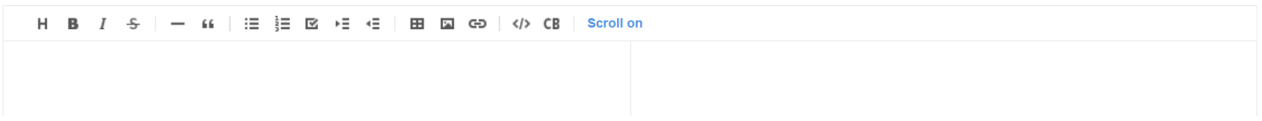
수정
목록

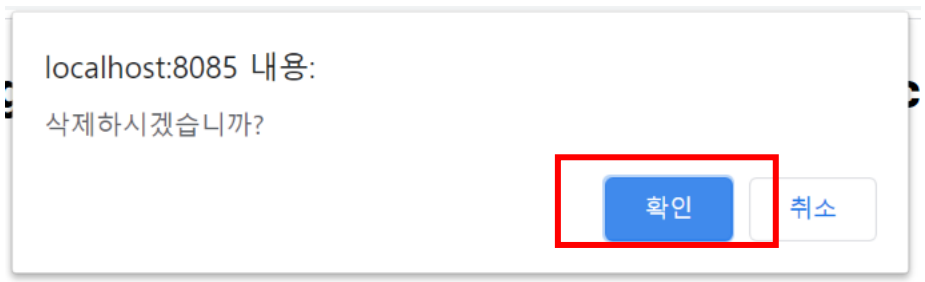
게시물 작성 테스트

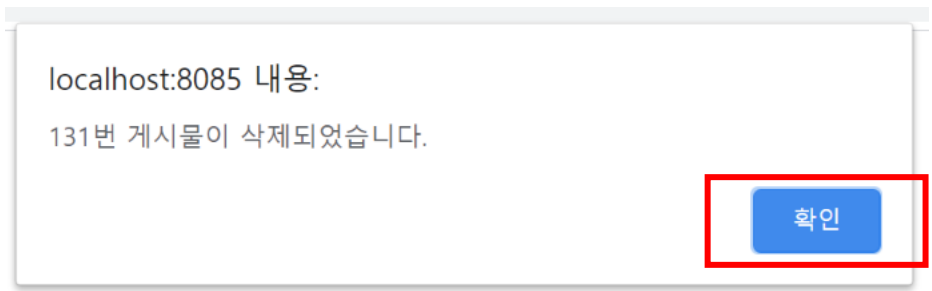
이전글

게시물 삭제

댓글 입력







게시물 상세보기

```
case "detail":
    return actionDetail();
```

```
private String actionDetail() {
    if (Util.empty(req, "id")) {
        return "html:id를 입력해주세요.";
    }
    if (Util.isNum(req, "id") == false) {
        return "html:id를 정수로 입력해주세요.";
    }

    int id = Util.getInt(req, "id");

    int cateItemId = 0;

    if (!Util.empty(req, "cateItemId") && Util.isNum(req, "cateItemId")) { // cateItemId가 없지 않고 숫자가 맞으면

        cateItemId = Util.getInt(req, "cateItemId");
    }

    articleService.increaseHit(id);

    // 게시물 작성자만 삭제버튼 보이게 하기 위한 현재 접속자 확인용
    int loginedMemberId = (int) req.getAttribute("loginedMemberId");

    Article article = articleService.getForPrintArticle(id, loginedMemberId);

    // 항목 모두 불러오는 메서드 네임
    int beforeId = articleService.getForPageMoveBeforeArticle(id, cateItemId);
    int afterId = articleService.getForPageMoveAfterArticle(id, cateItemId);
    CateItem cateItem = articleService.getCateItem(article.getCateItemId());
    Member member = memberService.getMemberFromMemberId(article.getMemberId());
    req.setAttribute("member", member);
    req.setAttribute("beforeId", beforeId);
    req.setAttribute("afterId", afterId);
    req.setAttribute("article", article);
    req.setAttribute("cateItem", cateItem);
    req.setAttribute("cateItemId", cateItemId);
    List<ArticleReply> articleReplies = articleService.getForPrintArticleReplies(id, loginedMemberId);
    req.setAttribute("articleReplies", articleReplies);

    return "article/detail.jsp";
}
```

ArticleController

게시물 상세보기

```
public Article getForPrintArticle(int id, int actorId) {
    Article article = articleDao.getForPrintArticle(id);
    updateArticleExtraDataForPrint(article, actorId);

    return article;
}
```

ArticleService

```
public Article getForPrintArticle(int id) {
    SecSql sql = SecSql.from("SELECT A.*");
    sql.append(", M.name AS extra__writer");
    sql.append("FROM article AS A");
    sql.append("INNER JOIN member AS M");
    sql.append("ON A.memberId = M.id");
    sql.append("WHERE A.displayStatus = 1");
    sql.append("AND A.id = ?", id);


    return new Article(DBUtil.selectRow(dbConn, sql));

}
```

ArticleDao

게시물 상세보기

```
public Article getForPrintArticle(int id, int actorId) {
    Article article = articleDao.getForPrintArticle(id);
    updateArticleExtraDataForPrint(article, actorId);

    return article;
}
```

ArticleService

```
public Article getForPrintArticle(int id) {
    SecSql sql = SecSql.from("SELECT A.*");
    sql.append(", M.name AS extra__writer");
    sql.append("FROM article AS A");
    sql.append("INNER JOIN member AS M");
    sql.append("ON A.memberId = M.id");
    sql.append("WHERE A.displayStatus = 1");
    sql.append("AND A.id = ?", id);


    return new Article(DBUtil.selectRow(dbConn, sql));

}
```

ArticleDao

게시물 이전글, 다음글, 조회수

```
public int getForPageMoveBeforeArticle(int id, int cateItemId) {
    return articleDao.getForPageMoveBeforeArticle(id, cateItemId);
}

public int getForPageMoveAfterArticle(int id, int cateItemId) {
    return articleDao.getForPageMoveAfterArticle(id, cateItemId);
}

public void increaseHit(int id) {
    articleDao.increaseHit(id);

}
```

ArticleService

게시물 이전글, 다음글, 조회수

```
public int getForPageMoveBeforeArticle(int id, int cateItemId) {
    SecSql sql = new SecSql();
    sql.append("SELECT id ");
    sql.append("FROM article ");
    sql.append("WHERE id < ? ", id);
    sql.append("AND displayStatus = 1 ");
    if (cateItemId != 0) {
        sql.append("AND cateItemId = ? ", cateItemId);
    }
    sql.append("ORDER BY id DESC ");
    sql.append("LIMIT 1");

    int articleId = DBUtil.selectRowIntValue(dbConn, sql);

    return articleId;
}

public int getForPageMoveAfterArticle(int id, int cateItemId) {
    SecSql sql = new SecSql();
    sql.append("SELECT id");
    sql.append("FROM article");
    sql.append("WHERE id > ?", id);
    sql.append("AND displayStatus = 1");
    if (cateItemId != 0) {
        sql.append("AND cateItemId = ?", cateItemId);
    }

    sql.append("ORDER BY id DESC ");
    sql.append("LIMIT 1");

    int articleId = DBUtil.selectRowIntValue(dbConn, sql);

    return articleId;
}

public int increaseHit(int id) {
    SecSql sql = SecSql.from("UPDATE article");
    sql.append("SET hit = hit + 1");
    sql.append("WHERE id = ?", id);

    return DBUtil.update(dbConn, sql);
}
```

ArticleDao

댓글

댓글 작성

```
case "doWriteReply":
    return actionDoWriteReply();
```

```
private String actionDoWriteReply() {

    if (Util.empty(req, "articleId")) {
        return "html:articleId를 입력해주세요.";
    }

    if (Util.isNum(req, "articleId") == false) {
        return "html:articleId를 정수로 입력해주세요.";
    }

    int articleId = Util.getInt(req, "articleId");

    int loginedMemberId = (int) req.getAttribute("loginedMemberId");
    String body = req.getParameter("body");

    String redirectUri = Util.getString(req, "redirectUri");
    System.out.println(redirectUri);

    int id = articleService.writeArticleReply(body, articleId, loginedMemberId);

    redirectUri = Util.getNewUri(redirectUri, "lastWorkArticleReplyId", id + "");

    return "html:<script> alert('" + articleId + "번 게시물 댓글을 작성하셨습니다.'); location.replace('" + redirectUri
            + "'); </script>";
}
```

ArticleController

```
public int writeArticleReply(String body, int articleId, int memberId) {
    return articleDao.writeArticleReply(body, articleId, memberId);
}
```

ArticleService

댓글 작성

```java
public int writeArticleReply(String body, int articleId, int memberId) {
    SecSql sql = SecSql.from("INSERT INTO articleReply");
    sql.append("SET regDate = NOW()");
    sql.append(", updateDate = NOW()");
    sql.append(", articleId = ?", articleId);
    sql.append(", `body` = ?", body);
    sql.append(", displayStatus = 1 ");
    sql.append(", memberId = ?", memberId);

    return DBUtil.insert(dbConn, sql);
}
```

ArticleDao

```javascript
function WriteReplyList__showTop() {
    var top = $('.cancel').offset().top;
    $(window).scrollTop(top);
}

function WriteReplyList__showDetail() {
    WriteReplyList__showTop();

    var $tr = $('.border .replyList[data-id="'
            + param.lastWorkArticleReplyId + '"]');
    $tr.addClass('high');
    setTimeout(function() {
        $tr.removeClass('high');
    }, 1000);
}
```

detail.jsp

- 댓글을 작성하면 좀 전에 작성된 댓글에 하이라이트로 색상을 1초간 주어 알아보기 쉽도록 구현했습니다.

댓글 수정

```
case "modifyReply":
    return actionModifyReply();
case "doModifyReply":
    return actionDoModifyReply();
```

```
private String actionModifyReply() {

    if (Util.empty(req, "id")) {
        return "html:id를 입력해주세요.";
    }

    if (Util.isNum(req, "id") == false) {
        return "html:id를 정수로 입력해주세요.";
    }

    int id = Util.getInt(req, "id"); // 댓글 번호

    int loginedMemberId = (int) req.getAttribute("loginedMemberId");

    ArticleReply articleReply = articleService.getArticleReply(id);
    req.setAttribute("articleReply", articleReply);

    Article article = articleService.getForPrintArticle(articleReply.getArticleId(), loginedMemberId);
    req.setAttribute("article", article);

    return "article/modifyReply.jsp";
}
```

ArticleController

- 수정할 댓글의 게시물과 댓글을 ArticleService에 요청해서 받아옵니다.

댓글 수정

```
private String actionDoModifyReply() {

    if (Util.empty(req, "id")) {
        return "html:id를 입력해주세요.";
    }
    if (Util.isNum(req, "id") == false) {
        return "html:id를 정수로 입력해주세요.";
    }

    int id = Util.getInt(req, "articleReplyId"); // 댓글 번호
    String body = Util.getString(req, "body");

    System.out.println("id : " + id);

    int loginedMemberId = (int) req.getAttribute("loginedMemberId");

    Map<String, Object> getReplyCheckRsModifyAvailable = articleService.getReplyCheckRsModifyAvailable(id,
            loginedMemberId);

    if (Util.isSuccess(getReplyCheckRsModifyAvailable) == false) {
        return "html:<script> alert('" + getReplyCheckRsModifyAvailable.get("msg")
                + "'); history.back(); </script>";
    }

    articleService.modifyArticleReply(id, body);

    int articleId = 0; // 게시물 id

    if (!Util.empty(req, "id") && Util.isNum(req, "id")) {
        articleId = Util.getInt(req, "id");
    }

    String redirectUri = Util.getString(req, "redirectUri", "list");

    redirectUri = Util.getNewUri(redirectUri, "lastWorkArticleReplyId", id + "");

    return "html:<script> alert('" + articleId + "번 게시물의 " + id + "댓글을 수정하셨습니다.'); location.replace('" + redirectUri
            + "'); </script>";
}
```

ArticleController

- jsp에서 입력 받은 댓글 수정 내용을 사용자에게 전달받아 ArticleService에 저장을 요청합니다.

댓글 수정

```
public int modifyArticleReply(int id, String body) {
    return articleDao.modifyArticleReply(id, body);
}
```

ArticleService

```
public int modifyArticleReply(int id, String body) {
    SecSql sql = SecSql.from("UPDATE articleReply ");
    sql.append("SET updateDate = NOW()");
    sql.append(", body = ? ", body);
    sql.append(" WHERE id = ?", id);

    return DBUtil.update(dbConn, sql);

}
```

ArticleDao

댓글 삭제

```
 case "doDeleteReply":
     return actionDoDeleteReply();
 }
```

```
private String actionDoDeleteReply() {

    int loginedMemberId = (int) req.getAttribute("loginedMemberId");
    int replyId = 0;

    if (!Util.empty(req, "replyId") && Util.isNum(req, "replyId")) { // 댓글 번호
        replyId = Util.getInt(req, "replyId");
    }
    int id = 0;

    if (!Util.empty(req, "id") && Util.isNum(req, "id")) { // 게시물 번호
        id = Util.getInt(req, "id");
    }

    Map<String, Object> getReplyCheckRsDeleteAvailable = articleService.getReplyCheckRsDeleteAvailable(replyId,
            loginedMemberId);
    if (Util.isSuccess(getReplyCheckRsDeleteAvailable) == false) {
        return "html:<script> alert('" + getReplyCheckRsDeleteAvailable.get("msg")
                + "'); history.back(); </script>";
    }

    articleService.deleteArticleReply(replyId);
    String redirectUri = Util.getString(req, "redirectUri", "list");

    return "html:<script> alert('댓글이 삭제되었습니다.'); location.replace('" + redirectUri + "'); </script>";
}
```

ArticleController

```
public int deleteArticleReply(int id) {
    return articleDao.deleteArticleReply(id);
}
```

ArticleService

```
public int deleteArticleReply(int id) {
    SecSql sql = SecSql.from("DELETE FROM articleReply ");
    sql.append("WHERE id = ? ", id);

    return DBUtil.update(dbConn, sql);
}
```

ArticleDao

댓글 리스팅

```
        List<ArticleReply> articleReplies = articleService.getForPrintArticleReplies(id, loginedMemberId);
        req.setAttribute("articleReplies", articleReplies);

        return "article/detail.jsp";
    }
```

ArticleController

```
public List<ArticleReply> getForPrintArticleReplies(int articleId, int actorId) {
    List<ArticleReply> articleReplies = articleDao.getForPrintArticleReplies(articleId, actorId);

    for ( ArticleReply articleReply : articleReplies ) {
        updateArticleReplyExtraDataForPrint(articleReply, actorId);
    }

    return articleReplies;
}
```

ArticleService

```
public List<ArticleReply> getForPrintArticleReplies(int articleId, int actorId) {


    SecSql sql = SecSql.from("SELECT A.*, M.nickname AS extra__writer");
    sql.append("FROM articleReply AS A");
    sql.append("INNER JOIN `member` AS M");
    sql.append("ON A.memberId = M.id");
    sql.append("WHERE articleId = ?", articleId);
    sql.append("ORDER BY A.id DESC");


    List<Map<String, Object>> rows = DBUtil.selectRows(dbConn, sql);
    List<ArticleReply> articleReplies = new ArrayList<>();

    for (Map<String, Object> row : rows) {
        articleReplies.add(new ArticleReply(row));
    }
    return articleReplies;
}
```

ArticleDao

댓글 리스팅

```
        List<ArticleReply> articleReplies = articleService.getForPrintArticleReplies(id, loginedMemberId);
        req.setAttribute("articleReplies", articleReplies);

        return "article/detail.jsp";
    }
```

ArticleController

```
public List<ArticleReply> getForPrintArticleReplies(int articleId, int actorId) {
    List<ArticleReply> articleReplies = articleDao.getForPrintArticleReplies(articleId, actorId);

    for ( ArticleReply articleReply : articleReplies ) {
        updateArticleReplyExtraDataForPrint(articleReply, actorId);
    }

    return articleReplies;
}
```

ArticleService

- 리스팅 할 댓글을 불러올 때, 수정 권한을 extra에 담아 ArticleController에 전달해줍니다.

```
public List<ArticleReply> getForPrintArticleReplies(int articleId, int actorId) {


    SecSql sql = SecSql.from("SELECT A.*, M.nickname AS extra__writer");
    sql.append("FROM articleReply AS A");
    sql.append("INNER JOIN `member` AS M");
    sql.append("ON A.memberId = M.id");
    sql.append("WHERE articleId = ?", articleId);
    sql.append("ORDER BY A.id DESC");


    List<Map<String, Object>> rows = DBUtil.selectRows(dbConn, sql);
    List<ArticleReply> articleReplies = new ArrayList<>();

    for (Map<String, Object> row : rows) {
        articleReplies.add(new ArticleReply(row));
    }
    return articleReplies;
}
```

ArticleDao

```java
public class SecSql {
    private StringBuilder sqlBuilder;
    private List<Object> datas;

    @Override
    public String toString() {
        return "sql=" + getFormat() + ", data=" + datas;
    }

    public SecSql() {
        sqlBuilder = new StringBuilder();
        datas = new ArrayList<>();
    }

    public boolean isInsert() {
        return getFormat().startsWith("INSERT");
    }

    public SecSql append(Object... args) {
        if (args.length > 0) {
            String sqlBit = (String) args[0];
            sqlBuilder.append(sqlBit + " ");
        }

        for (int i = 1; i < args.length; i++) {
            datas.add(args[i]);
        }
        return this;
    }

public PreparedStatement getPreparedStatement(Connection dbConn) throws SQLException {
    PreparedStatement stmt = null;

    if (isInsert()) {
        stmt = dbConn.prepareStatement(getFormat(), Statement.RETURN_GENERATED_KEYS);
    } else {
        stmt = dbConn.prepareStatement(getFormat());
    }

    for (int i = 0; i < datas.size(); i++) {
        Object data = datas.get(i);
        int parameterIndex = i + 1;

        if (data instanceof Integer) {
            stmt.setInt(parameterIndex, (int) data);
        } else if (data instanceof String) {
            stmt.setString(parameterIndex, (String) data);
        }
    }

    return stmt;
}

public String getFormat() {
    return sqlBuilder.toString();
}

//빌더 패턴
public static SecSql from(String sql) {
    return new SecSql().append(sql);
}
```

SecSql

```
public class Util {

    //똑같은 메서드는 여러개가 있어도 상관없다.
    public static boolean empty(HttpServletRequest req, String paramName) {
         String paramValue = req.getParameter(paramName);
        return empty(paramValue);
    }
    public static  boolean empty(Object obj) {
         if ( obj == null ) {
             return true;
         }
         if ( obj instanceof String) {
             return ((String)obj).trim().length() == 0 ;
         }
        return true;
    }
    public static boolean isNum(HttpServletRequest req, String paramName) {
        String paramValue = req.getParameter(paramName);
        return isNum(paramValue);
    }
    public static boolean isNum(Object obj) {
        if ( obj == null ) {
            return false;
        }
        if ( obj instanceof Long ) {
            return true;
        }
        else if ( obj instanceof Integer ) {
            return true;
        }
        else if ( obj instanceof String ) {
            try {
                Integer.parseInt((String)obj);
                return true;
            }
            catch ( NumberFormatException e ) {
                return false;
            }
        }
        return false;
    }
```

```
public static int sendMail(String smtpServerId, String smtpServerPw, String from, String fromName, String to
        String title, String body) {
    Properties prop = System.getProperties();
    prop.put("mail.smtp.starttls.enable", "true");
    prop.put("mail.smtp.host", "smtp.gmail.com");
    prop.put("mail.smtp.auth", "true");
    prop.put("mail.smtp.port", "587");

    Authenticator auth = new MailAuth(smtpServerId, smtpServerPw);

    Session session = Session.getDefaultInstance(prop, auth);

    MimeMessage msg = new MimeMessage(session);

    try {
        msg.setSentDate(new Date());

        msg.setFrom(new InternetAddress(from, fromName));
        msg.setRecipient(Message.RecipientType.TO, new InternetAddress(to));
        msg.setSubject(title, "UTF-8");
        msg.setText(body, "UTF-8");
        msg.setContent(body, "text/html; charset=UTF-8");




        Transport.send(msg);

    } catch (AddressException ae) {
        System.out.println("AddressException : " + ae.getMessage());
        return -1;
    } catch (MessagingException me) {
        System.out.println("MessagingException : " + me.getMessage());
        return -2;
    } catch (UnsupportedEncodingException e) {
        System.out.println("UnsupportedEncodingException : " + e.getMessage());
        return -3;
    }

    return 1;
}
```

```
public static String getTempPassword(int length) {
    int index = 0;
    char[] charArr = new char[] { '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'A', 'B', 'C', 'D', 'E', 'F',
    'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z', 'a',
    'b', 'c', 'd', 'e', 'f', 'g', 'h', 'i', 'j', 'k', 'l', 'm', 'n', 'o', 'p', 'q', 'r', 's', 't', 'u', 'v',
    'w', 'x', 'y', 'z' };

    StringBuffer sb = new StringBuffer();

    for (int i = 0; i < length; i++) {
    index = (int) (charArr.length * Math.random());
    sb.append(charArr[index]);
    }

    return sb.toString();
}
public static String sha256(String base) {
    try {
        MessageDigest digest = MessageDigest.getInstance("SHA-256");
        byte[] hash = digest.digest(base.getBytes("UTF-8"));
        StringBuffer hexString = new StringBuffer();

        for ( int i = 0; i < hash.length; i++ ) {
            String hex = Integer.toHexString(0xff & hash[i]);
            if ( hex.length() == 1 )
                hexString.append('0');
            hexString.append(hex);
        }
        return hexString.toString();
    } catch (Exception ex) {
        return "";
    }
}
```

Util

Util을 끝으로 포트폴리오를 마치겠습니다.
긴 글 읽어 주셔서 감사합니다.

소스 코드 : https://github.com/khr777/myblog