〈 2022 . 03 .14 〉

어제 Project 를 합치고 회원가입 기능과

아이디 . 비밀번호 찾기 기능을 구현하였다 .

오늘의 목표는 로그인 기능 method 를 정리하고.

통합하고 있는 project 에도 로그인이 적용 되게 하자

1. 먼저 login 에 관련된 method 정리부터 하자

——— 수정 전

〈LoginService. java〉

```
package com.team.team_project.service.login;

        import com.team.team_project.dto.PasswordDTO.PasswordDTO;
        import org.springframework.ui.Model;
        import org.springframework.validation.Errors;
        import org.springframework.validation.FieldError;

        import javax.crypto.BadPaddingException;
        import javax.crypto.IllegalBlockSizeException;
        import javax.crypto.NoSuchPaddingException;
        import java.security.InvalidAlgorithmParameterException;
        import java.security.InvalidKeyException;
        import java.security.NoSuchAlgorithmException;
        import java.security.spec.InvalidKeySpecException;
        import java.util.HashMap;
        import java.util.Map;

public interface LoginService {


    Map<String, Object> forloginUpdate(String account, PasswordDTO dto) throws
InvalidAlgorithmParameterException, NoSuchPaddingException,
IllegalBlockSizeException, NoSuchAlgorithmException, InvalidKeySpecException,
BadPaddingException, InvalidKeyException;

}
```

〈 LoginServiceImpl.java 〉

```java
@Service
@Log4j2
@RequiredArgsConstructor
public class LonginServiceImpl implements LoginService {
    private final UserRepository userRepository;

    private final ValidateHandling validateHandling;


    @Override
    public Map<String, Object> forloginUpdate(String account, PasswordDTO dto) throws
InvalidAlgorithmParameterException, NoSuchPaddingException, IllegalBlockSizeException,
NoSuchAlgorithmException, InvalidKeySpecException, BadPaddingException,
InvalidKeyException {
        Map<String,Object> result = validateHandling.idAndEmailValidCheck(account);
        /*** idAndEmailValidCheck Key
         * // map 에 유효성 검사가 성공했는지 여부를 삽입
         *          validCheckResult.put("result",validFinalCheck);
         *
         *
         *          // 값들이 유효한 경우
         *          if(validFinalCheck==true) {
         *              validCheckResult.put("account",account);
         *              // 유효한 값이 email 일 경우\
         *              if (isItEmail == true) {
         *                  validCheckResult.put("validComplete_Value", "email");
         *                  validCheckResult.put("code",code);
         *                  validCheckResult.put("pw",password);
         *                  validCheckResult.put("nick",nick);
         *                  System.out.println(code);
         *                  System.out.println(password);
         *                  System.out.println(nick);
         *              }
         *              // 유효한 값이 id 일 경우
         *              else if (isItId == true) {
         *                  validCheckResult.put("validComplete_Value", "id");
         *                  validCheckResult.put("code",code);
         *                  validCheckResult.put("pw",password);
         *                  validCheckResult.put("nick",nick);
         *                  System.out.println(code);
         *                  System.out.println(password);
         *                  System.out.println(nick);
         *              }
         *          }
         * **/

        /**
         *
-------------------------------------------------------------------------------------
         * method 시작
         * **/

        /**
         * complete = true ----> 아이디 혹은 이메일이 존재하는 값
         * complete = false ----> error
         * */
        boolean complete = (boolean)result.get("result");

        /**
         * dto 에서 pw 값을 가져온다
         * */
        String pw = dto.getPw();

        /***
         * 최종적으로 pw 까지 맞는지 확인하기 위한 boolean 값
         * */
        boolean loginComplete = false;

        /** pwError Message 를 받을 String
         * **/
        String pwErrorMessage = null;
```

```
        if(complete==true){
            /**
             * complete 이 true 이면 nick, pw, code 값이 존재함
             * validComplete_value = email ----> 해당 값이 email
             * validComplete_value = id -------> 해당 값이 id
             * **/

            String pwResult = (String)result.get("pw");

            if(BCrypt.checkpw(pw, pwResult)){
                loginComplete = true;
            }else{
                /***
                 *  else 는 비밀번호가 서로 다르다는 말
                 *  pwError 에 값을 부여
                 * */
                pwErrorMessage = "Wrong Password ! Pleas check it";
            }
        }
        result.put("loginResult", loginComplete);

        if(loginComplete==false){
            result.put("passwordError", pwErrorMessage);
        }

        return result;
    }
}
```

⟨ PageController.java ⟩

```java
@PostMapping("login")
public String login(String account,@Valid PasswordDTO dto, Errors errors, Model model,
HttpSession session) throws Exception {

    if(errors.hasErrors()){
        model.addAttribute("dto", dto);
        Map<String, String> validatorResult = validateHandling.validateHandling(errors);
        for (String key : validatorResult.keySet()) {
            model.addAttribute(key, validatorResult.get(key));
        }

        return "checkplan/mainpage";
    }

    String url = null;

    Map<String, Object> loginResult = loginService.login(account,dto);

    boolean complete = (boolean) loginResult.get("loginResult");


    if(complete==true){
        session.setAttribute("code", (Long)loginResult.get("code"));
        session.setAttribute("nick",(String)loginResult.get("nick"));
        if(loginResult.get("status").equals("7day")){
            url = "checkplan/forUser/retire/UnscribingCancle";
        }else if(loginResult.get("status").equals("회원")){
            url =  "checkplan/logincomplete";
        }
    }else if(complete==false) {
        if ((boolean) loginResult.get("result") == false) {
            model.addAttribute("accountError", (String) loginResult.get("validErrorMessage"));
            return "checkplan/mainpage";
        } else if ((boolean) loginResult.get("result") == true) {
            model.addAttribute("passwordError", (String) loginResult.get("passwordError"));
            return "checkplan/mainpage";
        }
    }
    return url;
}
```

수정 후

〈 LoginService. java 〉

```java
package com.team.team_project.service.login;

import com.team.team_project.dto.passwordDTO.PasswordDTO;
import java.util.Map;

public interface LoginService {
    /**
     *  로그인을 위한 method
     */
    Map<String, Object> login(String account, PasswordDTO dto) throws Exception;
}
```

〈 LoginServiceImpl. java 〉

```java
@Service
@Log4j2
@RequiredArgsConstructor
public class LonginServiceImpl implements LoginService {

    private final ValidateHandling validateHandling;

    /**------------------- 로그인을 위한 method----------------------------------------------------------
     */
    @Override
    public Map<String, Object> login(String account, PasswordDTO dto) throws Exception {
        Map<String,Object> result = validateHandling.accountValidCheck(account);
        /**
         * accountValidCheck
         *  값이 유효하면-----
         *  result.put("code",(Long)info[0]);
         *  result.put("pw",(String)info[2]);
         *  result.put("nick",(String)info[3]);
         *  result.put("status",(String)info[4]);
         *  result.put("birthday",(LocalDate)info[5]);
         *
         *  값이 유효하지 않다면 -------
         *  Map 에 errorMessage
         *  key 값을 가진다
         */

        // false ---> 로그인이 불가능
        // true ----> 로그인이 가능하게 끔
        boolean isAvailable = false;

        // map 에 "errorMessage" 라는 key 가 존재하지 않을 때
        // -----> 입력한 id  와  email 이 존재한다
        if(result.get("errorMessage")==null)
        {

            if(BCrypt.checkpw(dto.getPw(), (String)result.get("pw")))
            {
                // 입력한 비밀번호와 등록한 비밀번호가 맞다면
                isAvailable = true;
            }

            else
            {
                /***
                 *  else 는 비밀번호가 서로 다르다는 말
                 *  pwError 에 값을 부여
                 * */
                result.put("pwError", "Wrong Password ! Pleas check it");
            }
        }
        // result ( map ) 에 로그인을 가능하게끔 하는 결과를 입력한다
        result.put("loginResult", isAvailable);
        return result;
    }
}
```

⟨ PageController. java ⟩

```java
/**
 * Login 을 하기 위한 method ==========================================================================
 */
@PostMapping("login")
public String login(String account,
                    @Valid PasswordDTO dto,
                    Errors errors,
                    Model model,
                    HttpSession session)
        throws Exception


{
    /**
     *  Password dto가 유효한지 확인하는 method ---------------------------------------------
     */
    if(errors.hasErrors()){
        model.addAttribute("dto", dto);
        Map<String, String> validatorResult = validateHandling.validateHandling(errors);
        for (String key : validatorResult.keySet()) {
            model.addAttribute(key, validatorResult.get(key));
        }

        return "checkplan/mainpage";
    }
    /**
     *  Password dto가 유효한지 확인하는 method ---------------------------------------------
     */

    String url = null;

    Map<String, Object> loginResult = loginService.login(account,dto);

    /**
     * login
     *  공통 값
     *  key = loginResult
     *
     *  true ----------
     *  result.put("code",(Long)info[0]);
     *  result.put("pw",(String)info[2]);
     *  result.put("nick",(String)info[3]);
     *  result.put("status",(String)info[4]);
     *  result.put("birthday",(LocalDate)info[5]);
     *
     *  false ------------
     *  key = errorMessage
     *  key = pwError
     */
```

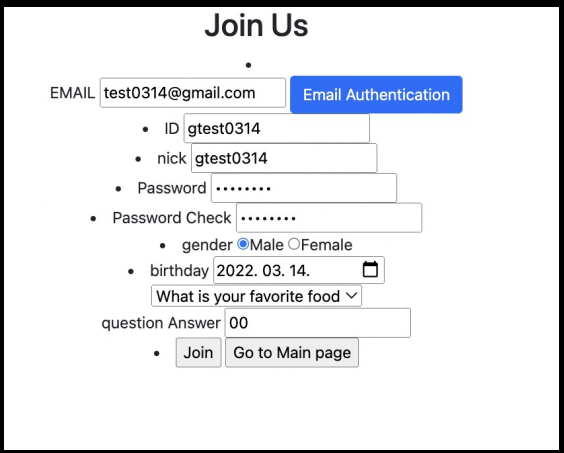
```
    boolean isLogin = (boolean) loginResult.get("loginResult");

    // isLogin ---> ture : 로그인 값이 유효 하다면
    if(isLogin)
    {
        session.setAttribute("code", (Long)loginResult.get("code"));
        session.setAttribute("nick",(String)loginResult.get("nick"));
        if(loginResult.get("status").equals("7day"))
        {
            url = "checkplan/forUser/retire/UnscribingCancle";
        }
        else if(loginResult.get("status").equals("회원"))
        {
            url =  "checkplan/logincomplete";
        }
    }
    else
    {
        String pwError = (String)loginResult.get("pwError");
        /**
         *  pwError 가 존재 ---> pw 가 틀렸음
         *  pwError 가 존재 하지 않음  ---> 입력한 id 혹은 email 이 틀렸을 때
         */
        if(pwError==null)
        {
            model.addAttribute("accountError", (String) loginResult.get("errorMessage"));
            return "checkplan/mainpage";
        }
        else
        {
            model.addAttribute("passwordError", (String) loginResult.get("pwError"));
            return "checkplan/mainpage";
        }
    }
    return url;
}
```

TEST————————————

〈 1 . Test 용 계정 2개를 만든다 〉

- 1 개는 탈퇴한 상태
- 나머지는 탈퇴하지 않은 상태로 설정

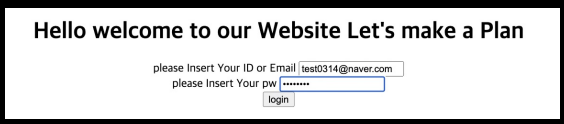


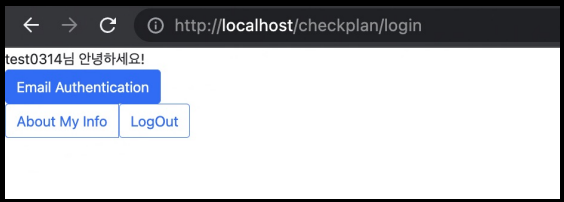
————〉 두 계정 모두 비밀번호는 Qwer!234

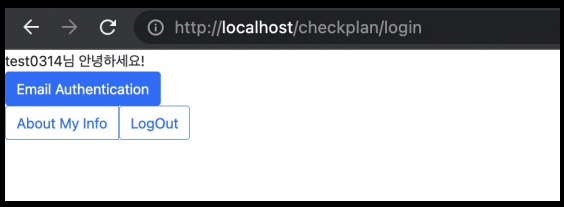
〈 2. 우선 계정 중 하나를 선택해서 유효성이 제대로 잘 작동하는 지 확인한다. 〉

case 1. 올바르게 입력했을 경우

email 을 사용해서 로그인

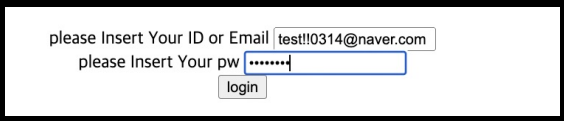


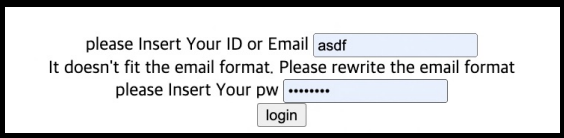




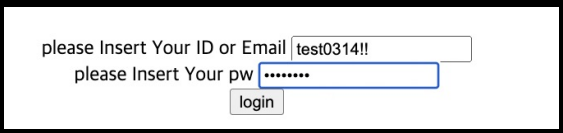
case 2. id 혹은 email 을 형식에 맞지 않게 입력 했을 때

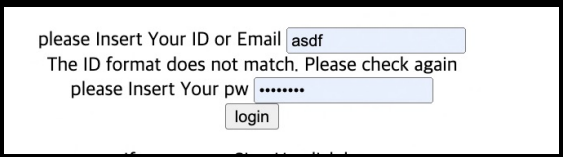
email





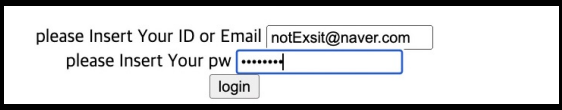
id

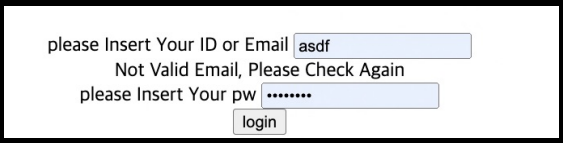




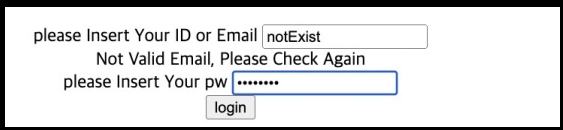
case 3. 존재하지 않는 id, email 을 입력 하였을 때

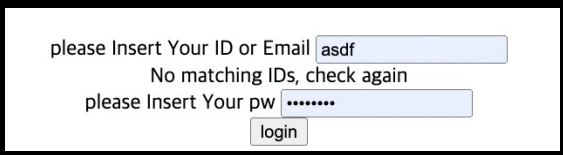
email





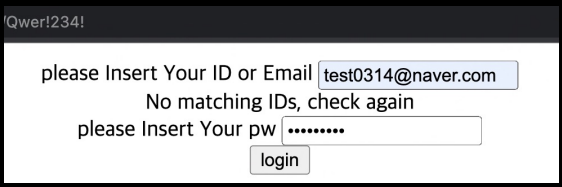
id

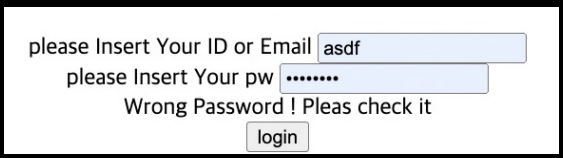




case4. 비밀번호를 틀렸을 때

email

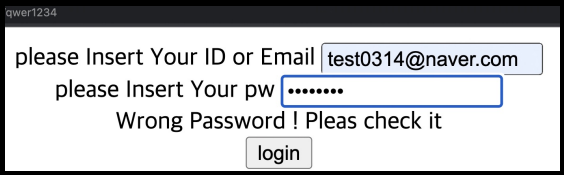


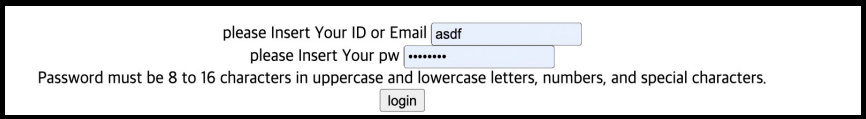


id

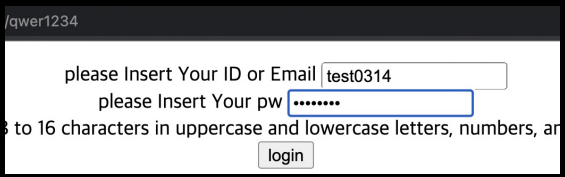
case 5. 비밀번호 형식에 맞지 않게 입력 했을 때

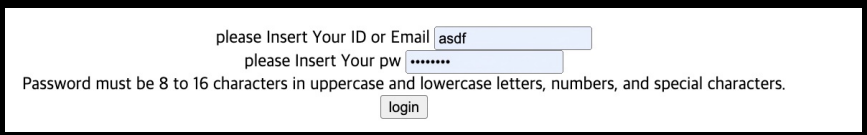
email





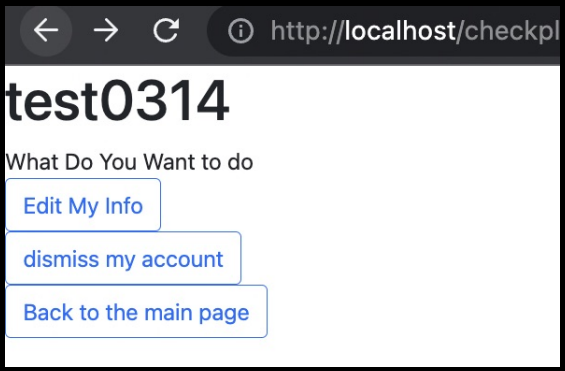
id

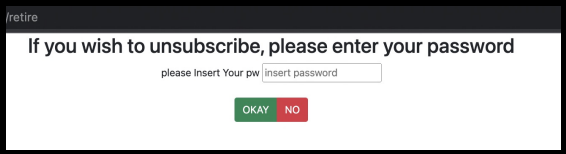


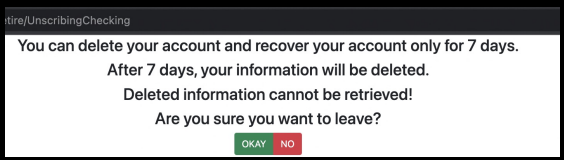


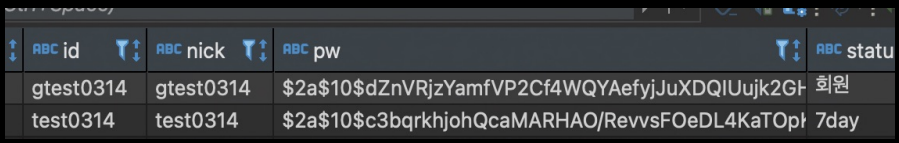
〈 3. 2개중 하나의 계정을 탈퇴 〉

(탈퇴 하면 status 칼럼에 7day 라는 값으로 변경)
——〉 7 일 이후에 expired 라는 값으로 자동 변경할 예정







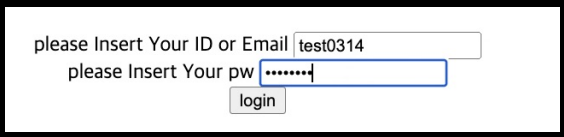


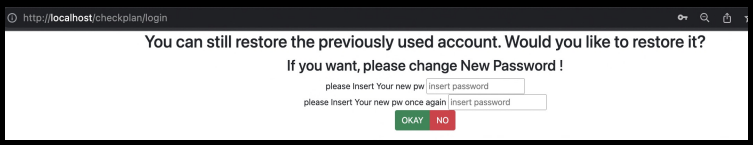
ABC id	ABC nick	ABC pw	ABC statu
gtest0314	gtest0314	$2a$10$dZnVRjzYamfVP2Cf4WQYAefyjJuXDQlUujk2GH	회원
test0314	test0314	$2a$10$c3bqrkhjohQcaMARHAO/RevvsFOeDL4KaTOpH	7day

gtest0314 의 status ——〉 회원

test0314 의 status ——〉 7day

case 1. id : test0314 로 로그인 할 경우 ——〉 탈퇴한 아이디





case 2. id: gtest0314 로 로그인 한 경우 ——〉 탈퇴하지 않음