2022 03 21 mon

TEAM Project

오늘 해야할 일

1. 원래 Project에 회원 탈퇴 기능 Refactoring

2. 회원 탈퇴기능 calendarCRUR Project 에 적용 시키기

〈 1. 원래 Project 에서 회원 탈퇴기능 Refactoring 〉

해당 기능에 관련된 method

Controller

```
@GetMapping("retire/retire")
public void retire(){
}

@PostMapping("retire/UnscribingChecking")
public String unscribingChecking(@Valid PasswordDTO dto, Errors errors,Model model, HttpSession
session){
    String url = null;
    String userNick = (String) session.getAttribute("nick");
    String pw = dto.getPw();

    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/forUser/retire/retire";
    }

    boolean userPwCollectOrNot = editService.bringPwForRetire(pw, userNick);
    if(userPwCollectOrNot==true){
        url = "checkplan/forUser/retire/UnscribingDoing";
    }else if(userPwCollectOrNot==false){

        model.addAttribute("errorMessage","Wrong Password! , Please Check again");
        return  "checkplan/forUser/retire/retire";
    }
    return url;
}
@Autowired
private UnscribeService unscribeService;


@GetMapping("retire/UnscribingComplete")
public void unscribingComplete(HttpSession sessioin){
    String userNick = (String) sessioin.getAttribute("nick");
    String userStatus = "7day";
    boolean userUnscribeComplete = unscribeService.userUnscribecomplete(userStatus, userNick);
    if(userUnscribeComplete==true){
        unscribeService.userModDateUpdate(userNick);
        sessioin.invalidate();
    }
}
@PostMapping("unScribeCancle")
public String unScribeCancle(HttpSession session, @Valid PasswordDTO dto,Errors errors, Model model,
String pwCheck){
    Long code = (Long) session.getAttribute("code");
    String url = null;

    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/forUser/retire/UnscribingCancle";
    }
    /***
     * dto 에서 pw 를 받아온다.
     */
    String pw = dto.getPw();

    int unScribeCancleResult = editService.unSubScribeCancle(pw, pwCheck, code);
    if(unScribeCancleResult>0){
        url = "redirect:/checkplan/mainpage";
    }else{
        model.addAttribute("ErrorMessage", "The passwords entered do not match each other.");
        return "checkplan/forUser/retire/UnscribingCancle";
    }
    return url;
}
```

1.

```
// --------------- 회원 탈퇴 시 회원 정보를 한번 더 확인하는 method ------
@PostMapping("retire/UnscribingChecking")
public String unscribingChecking(@Valid PasswordDTO dto, Errors errors,Model model, HttpSession
session){
    String url = null;
    String userNick = (String) session.getAttribute("nick");
    String pw = dto.getPw();

    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/forUser/retire/retire";
    }

    boolean userPwCollectOrNot = editService.bringPwForRetire(pw, userNick);
    if(userPwCollectOrNot==true){
        url = "checkplan/forUser/retire/UnscribingDoing";
    }else if(userPwCollectOrNot==false){

        model.addAttribute("errorMessage","Wrong Password! , Please Check again");
        return  "checkplan/forUser/retire/retire";
    }
    return url;
}
```

Script

탈퇴 하려고 할 때, 탈퇴하려는 사용자가

사용자 자신이 맞는지 확인하기 위해서

비밀번호를 한번 더 입력받아서 본인 확인을 수행

본인 확인 결과 사용자 본인이 맞다면 탈퇴 page 로 이동

해당 controller method 를 수정하기 위해, 수정해야할 method

——〉 수정 전

1. Repository

```
// ========== 회원 탈퇴 ========================================

// -------- 회원 닉네임을 가지고 회원 계정 상태를 변경
@Modifying
@Transactional
@Query(value = "update User set status=:status where nick=:nick")
int unSubscribe(@Param("status") String status, @Param("nick")String nick);
```

2. Service

```
boolean userUnscribecomplete(String status, String nick);
```

3. ServiceImpl

```
    @Override
    public boolean userUnscribecomplete(String status, String nick) {
        boolean successOrNot =false;
        int unscribeResult = userRepository.unSubscribe(status, nick);
        if(unscribeResult>0){
            successOrNot = true;
        }
        return successOrNot;
    }
```

해당 controller method 를 수정하기 위해. 수정해야할 method

——〉 수정 후

1. Repository

```
@Modifying
@Transactional
@Query(value = "update User set status=:status where code=:code")
int unSub(@Param("status") String status, @Param("code")Long code);
```

Script

1. method 이름 수정

2. code 를 받아서 status 값을 수정하는 형태로 변경

이유 :

1. code 는 pid 값여서 유일 무이 하기 때문입니다 .

2. nick 과 code 둘 다 . user 고유의 값이지만

nick 은 회원 정보 수정을 통해 변경 될 수 있기 때문에

변경 할 수 없는 code 값을 사용

2. Service

```
/** 회원 탈퇴를 진행하는 method ----------
 */
boolean unSubscribing(String status, Long code);
```

Script

1. method 이름 변경

2. 어떤 기능을 수행하는지 주석 처리

3. ServiceImpl

```
/** 회원 탈퇴를 진행하는 method
 */
@Override
public boolean unSubscribing(String status, Long code)
{
    boolean isUnSub =false;

    // 어떠한 오류로 인해 Update 가 안될 경우를 방지
    if(userRepository.unSub(status, code)>0)
        {
            isUnSub = true;
        }
    return isUnSub;
}
```

Script

1. method 이름 변경

2. 어떤 기능을 수행하는지 주석 처리

3. 필요 없는 member 변수 값 줄이고 , 변수 이름 재설정

Controller 수정

```
/** 회원 탈퇴 시 비밀번호를 입력 받아, 탈퇴 할 것인지 확인 받는 method
----------------------------------------
 */
@PostMapping("retire/UnscribingChecking")
public String unSubCheck(@Valid PasswordDTO dto, Errors errors,Model model, HttpSession session)
{
    // ---- 유효성 검사 ----------
    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/forUser/retire/retire";
    }

    /** 입력한 비밀번호와, user 의 비밀번호가 같은지 확인
     */

    // ---  계정을 탈퇴 ( 휴면 ) 계정으로 전환 확인 page 로 이동  ----------
    if( editService.isPwEqual(dto.getPw(), (Long)session.getAttribute("code")) )
        {
            return  "checkplan/forUser/retire/UnscribingDoing";
        }
    // --- 비밀번호가 서로 맞지 않음으로 현재 page로 이동 ----------
    else
        {
            model.addAttribute("errorMessage","Wrong Password! , Please Check again");
            return  "checkplan/forUser/retire/retire";
        }
}
```

Script ———

1. method 이름 변경

2. 불필요한 member 변수 선언 제거

* login . logout. 회원 정보 수정 시 modDate 를 변경하는 method ──〉 수정 전

1. Repository

```
// -------- 회원 닉네임을 삽입해서 마지막 수정 날짜를 변경 --------------
@Modifying
@Transactional
@Query(value = "update User set modDate=:modDate where nick=:nick")
int updateModate(@Param("modDate") LocalDateTime modDate, @Param("nick")String nick);
```

2 . Service

```
void userModDateUpdate(String nick);
```

3 . ServiceImpl

```
    @Override
    public void userModDateUpdate(String nick) {
        LocalDateTime localDateTime = LocalDateTime.now();
        userRepository.updateModate(localDateTime, nick);
    }
```

1. Repository

```
@Modifying
@Transactional
@Query(value = "update User set modDate=:modDate where code=:code")
int updateModDateTemp(@Param("modDate") LocalDateTime modDate, @Param("code")Long
code);
```

Script —

1. 현재 적용 된 곳이 많아 Temp를 붙여서 새로 생성

2. code 를 받아서, modDate 를 수정하게 변경

2. Service

```
/** modeDate를 변경하는 method
 */
void updateModDate(Long code );
```

Script

1. 어떤 기능을 수행하는지 주석 추가

2. Code 를 받아서 구현하게 끔 수정

3. ServiceImpl

```
/** mode date를 변경하는 Method
 */
@Override
public void updateModDate(Long code)
{
    LocalDateTime localDateTime = LocalDateTime.now();
    userRepository.updateModDateTemp(localDateTime,code);
}
```

Script

1. Code 를 받아서 구현하게 끔 수정

2.

```
// ------------ 회원 탈퇴를 진행하는 method
-----------------------
@GetMapping("retire/UnscribingComplete")
public void unscribingComplete(HttpSession sessioin){
    String userNick = (String) sessioin.getAttribute("nick");
    String userStatus = "7day";
    boolean userUnscribeComplete =
unscribeService.userUnscribecomplete(userStatus, userNick);
    if(userUnscribeComplete==true){
        unscribeService.userModDateUpdate(userNick);
        sessioin.invalidate();
    }
}
```

Script —

사용자 본인이 맞다면 session 에서 nick Name 을 받아

사용자 계정 상태를 '7day' 로 변경

해당 controller 를 변경하기 위해서

변경해야하는 method ⟶ 수정 전

〈 1. UserRepository 〉

```
@Query(value = "select pw from User where nick=:nick")
String brinUserPw(@Param("nick")String nick);
```

Script —

user nick 을 받아서 user 의 현재 비밀번호를 받아오는 method

〈 2. EditService 〉

```
boolean bringPwForRetire(String pw, String nick);
```

Script —

user nick 으로 user의 현재 비밀번호와 입력한 비밀번호가

일치하는지 확인하기 위한 method

〈 3. EditServiceImpl 〉

```
@Override
public boolean bringPwForRetire(String pw,String nick) {
    boolean userConfirm = false;
    String userPw = userRepository.brinUserPw(nick);
    if(BCrypt.checkpw(pw, userPw)){
        userConfirm = true;
    }
    /**
     * nick name 으로 user 의 pw 를 가져오는 것이라서
     * pw 가 없을 수가 없다 .
     */


    return userConfirm;
}
```

Script —

위와 동일

수정 후 —

〈 1. UserRepository 〉

—

```
// --------------------------- 사용자의 비밀번호를 가져오는 method
@Query(value = "select pw from User where code=:code")
String getPw(@Param("code")Long code);
```

Script —

1. code 를 입력 받고 현재 pw 를 리턴받도록 수정

2. method 기능에 대한 주석 추가

3. method 이름 변경

〈 2. EditService 〉

```
/** 입력한 비밀번호가, 사용자의 비밀번호와 같은지 확인하는 method
*/
boolean isPwEqual(String pw, Long code);
```

Script —

1. method 를 설명하는 주석 추가

2. 입력한 비밀번호와 사용자의 code 를 입력 받도록 수정

〈 3. EditServiceImpl 〉

```
/** 입력한 비밀번호가, 사용자의 비밀번호와 같은지 확인하는 method
 */
@Override
public boolean isPwEqual(String pw,Long code)
{
    boolean isValid = false;

    String userPw = userRepository.getPw(code);

    if(BCrypt.checkpw(pw, userPw))
    {
        isValid = true;
    }

    return isValid;
}
```

Script —

1. method 를 설명하는 주석 추가

2. member 변수명 변경, method 이름 변경

Controller

```
// ------------ 회원 탈퇴를 진행하는 method
-----------------------------------------------------------------
@PostMapping("retire/UnscribingComplete")
public String unSubComplete(HttpSession sessioin)
{
    Long code = (Long)sessioin.getAttribute("code");

    // 정상적으로 만료가 된다면
    if ( unscribeService.unSubscribing("휴면",code))
        {
            unscribeService.updateModDate(code);
            sessioin.invalidate();
        }
    return "checkplan/forUser/retire/UnscribingComplete";
}
```

Script —

1. method 를 설명하는 주석 추가

2. 멤버 변수 개수를 줄임

3. 데이터 전달 방식 Get —〉 Post 로 변경

3.

```
// ------------------------ 탈퇴한 회원 계정을 복구하는 method
@PostMapping("unScribeCancle")
public String unScribeCancle(HttpSession session, @Valid PasswordDTO dto,Errors errors, Model model,
String pwCheck){
    Long code = (Long) session.getAttribute("code");
    String url = null;

    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/forUser/retire/UnscribingCancle";
    }
    /***
     * dto 에서 pw 를 받아온다.
     */
    String pw = dto.getPw();

    int unScribeCancleResult = editService.unSubScribeCancle(pw, pwCheck, code);
    if(unScribeCancleResult>0){
        url = "redirect:/checkplan/mainpage";
    }else{
        model.addAttribute("ErrorMessage", "The passwords entered do not match each other.");
        return "checkplan/forUser/retire/UnscribingCancle";
    }
    return url;
}
```

Script —

휴면 상태인 사용자가 만료 기간 내에 해당 계정으로

login 을 하였을 때, 휴면 상태를 해지하는 method

— 해당 controller method 수정 시 수정이 필요한 method

수정 전

〈 1. UserRepository 〉

```
// 계정 복구를 위한 method
@Modifying
@Transactional
@Query(value = "update User set status=:status, pw=:pw, modDate=:modDate where code=:code")
int unScribeCancle(@Param("status")String status,
                   @Param("pw")String pw,
                   @Param("modDate") LocalDateTime modDate,
                   @Param("code")Long code);
```

Script —

user nick 을 받아서 user 의 현재 비밀번호를 받아오는 method

〈 2. EditService 〉

```
int unSubScribeCancle(String pw, String pwCheck, Long code);
```

Script —

user nick 을 입력해서 return 받은 현재 password 와

비밀번호 일치 확인용 pwCheck 가

맞으면 비밀번호와 User state 를 '회원' 으로 변경하는 method

〈 3. EditServiceImpl 〉

—

```
@Override
public int unSubScribeCancle(String pw,  String pwCheck, Long code) {
    int result = -1;
    String pwresult = null;
    String status = "회원";

    /*** 새로운 pw 가 pwcheck 와 맞는지 확인 ----> 이거 method 따로 빼서 만들자  많이 사용됨
     * */
    if(pwAndDupCheck.pwAndPwCheck(pw,pwCheck)){
        pwresult = BCrypt.hashpw(pw,BCrypt.gensalt());
        LocalDateTime modDate = LocalDateTime.now();
        result = userRepository.unScribeCancle(status, pwresult, modDate, code);
    }



    return result;
}
```

Script —

user nick 으로 user의 현재 비밀번호와 입력한 비밀번호가

일치하는지 여부에 따라 user state 를 변경하는 method

〈 4. pwAndDupCheckService 〉

—

```
boolean pwAndPwCheck(String pw, String pwCheck);
```

Script —

입력 받은 nick 으로 현재 사용자의 Password 를 받아오고

확인 용 Password (pw) 와 비교해서

서로 일치하는지 확인하는 method

⟨ 5. pwAndDupCheckServiceImpl ⟩

—

```
@Override
public boolean pwAndPwCheck(String pw, String pwCheck) {
    boolean pwSameCheck = false;
    if(pw.equals(pwCheck)){
        pwSameCheck = true;
    }
    return pwSameCheck;
}
```

Script —

입력 받은 nick 으로 현재 사용자의 Password 를 받아오고

확인 용 Password (pw) 와 비교해서

서로 일치하는지 확인하는 method

수정 후 —

〈 1. UserRepository 〉

—

```
/** 휴면 계정인 사용자의 state 를 '회원' 으로 변경하기 위한 method ----------
     */
    @Modifying
    @Transactional
    @Query(value = "update User set status=:status, pw=:pw, modDate=:modDate where code=:code")
    int unSubCancel(@Param("status")String status,
                    @Param("pw")String pw,
                    @Param("modDate") LocalDateTime modDate,
                    @Param("code")Long code);
}
```

Script —

1. method 에 대한 주석 추가

2. method 이름 변경

〈 2. EditService 〉

```
/** 비밀번호가 서로 일치하면 user 상태를 회원으로 변경하는 method
-------------------------------------------------------------
 */
int unSubCancel(String pw, String pwCheck, Long code);
```

Script —

1. method 이름 변경

2. 해당 Method의 기능 주석 추가

〈 3. EditServiceImpl 〉

—

```
/** 비밀번호가 서로 일치하면 user 상태를 회원으로 변경하는 method
------------------------------------------------------------
 */
@Override
public int unSubCancel(String pw,  String pwCheck, Long code) {
    // 비밀번호가 서로 일치할 때 -------------------
    if( pwAndDupCheck.isPwEqual(pw,pwCheck) )
        {

            LocalDateTime modDate = LocalDateTime.now();
            // 회원 정보를 "회원" 으로 변경 -------------------
        return userRepository.unSubCancel("회원", BCrypt.hashpw(pw,BCrypt.gensalt()), modDate, code);
        }
    // 비밀번호가 서로 일치하지 않을 때 -----------------------
    else
        {
     return -1;
        }
}
```

Script —

1. method 이름 변경

2. method 기능에 대한 주석 추가

3. 불필요한 method 제거

〈 4. pwAndDupCheckService 〉

—

```
/** pw 와 pwCheck 가 일치하는지 판단하는 method
 */
boolean isPwEqual(String pw, String pwCheck);
```

Script —

1. method 이름 변경

2. method 기능에 대한 주석 추가

⟨ 5. pwAndDupCheckServiceImpl ⟩

—

```
/** pw 와 pwCheck 가 일치하는지 판단하는 method
 */
@Override
public boolean isPwEqual(String pw, String pwCheck)
{
    boolean isValid = false;
    if(pw.equals(pwCheck))
        {
            isValid = true;
        }
    return isValid;
}
```

Script —

1. method 기능에 대한 주석 추가

2. method 이름 변경

Controller - 수정 후

```
/** 회원 상태가 expiring 인 계정 state 를 "회원"으로 변경
-------------------------------------------------------
 *  < 회원 계정 복구 >
 */
@PostMapping("unScribeCancle")
public String unSubCancel(
        HttpSession session,
        @Valid PasswordDTO dto,
        Errors errors,
        Model model,
        String pwCheck)
{

    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/forUser/retire/UnscribingCancle";
    }

    // 비밀번호 값과 비밀번호 확인 값이 같다면 --------------------
    if( editService.unSubCancel(dto.getPw(), pwCheck, (Long) session.getAttribute("code")) > 0 )
        {
            return "redirect:/checkplan/mainpage";
        }
    else
        {
            model.addAttribute("ErrorMessage", "The passwords entered do not match each other.");
        return "checkplan/forUser/retire/UnscribingCancle";
        }
}
```

Script —

1. method 이름 변경

2. method 기능에 대한 주석 추가

3. 불필요한 member 변수 제거