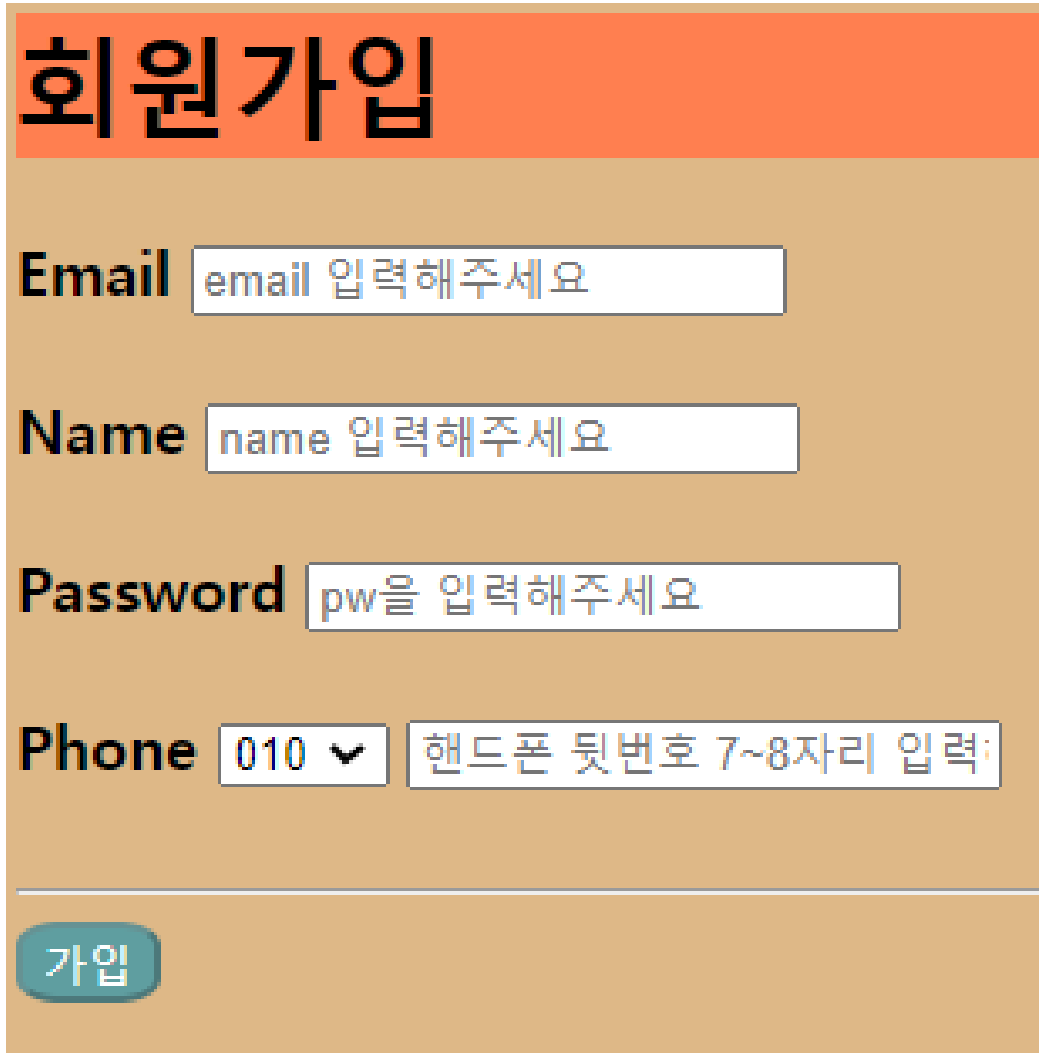
광주인력개발원

Back-end 프로젝트

박진석

1. 회원가입



1. 회원가입을 구현하여 Email과 Name, Pw, Phone 요소를 입력하게 했고
   유효성 검사를 통해 관리를 용이하게 했다.

2. Email과 Name은 중복검사 로직을 만들어 키에 준하게 만들었고 다른 로직에 사용할 수 있게 만들었다.

3. Pw는 암호화 하여 DB에 저장되게 구성하여 보안을 강화했다.

```
String passwordPattern = "^(?=.*[a-z])(?=.*[A-Z])(?=.*\\d)(?=.*[@$!%*?&])
[A-Za-z\\d@$!%*?&]{8,20}$";
if (!member.getPw().matches(passwordPattern)) {
    model.addAttribute(attributeName:"error", attributeValue:"비밀번호는 대문자,
    소문자, 숫자, 특수 문자를 최소한 하나씩 포함하고 8글자 이상 20글자 이내여야 합니다.
    ");
    return "redirect:/auth/memberup";
}
```

유효성 검사

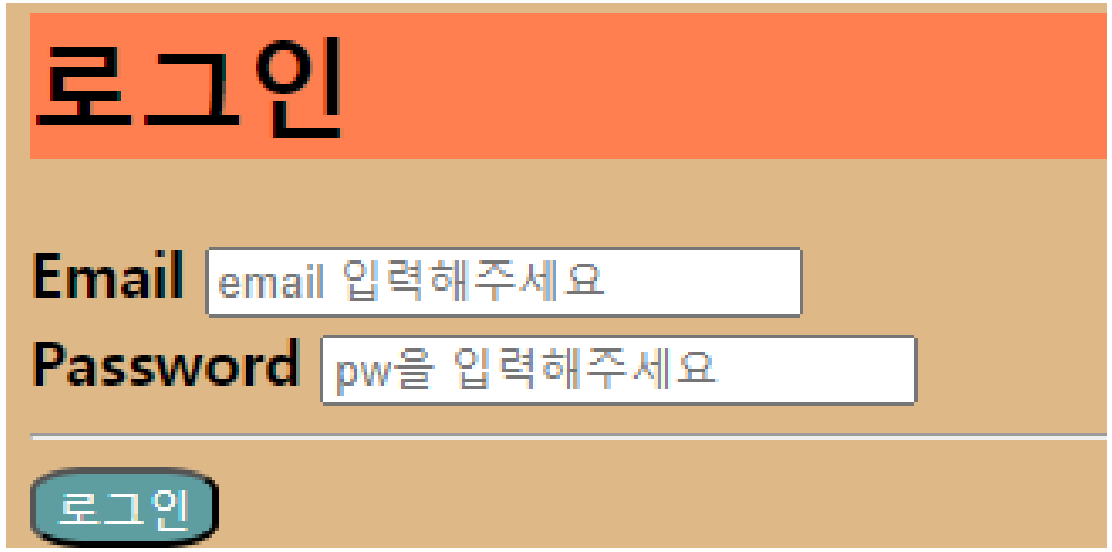
```
if (existingMemberByName != null) {
    model.addAttribute(attributeName:"error", attributeValue:"이미 사용 중인
    이름입니다.");
    return "redirect:/auth/memberup";
}
```

중복 검사

```java
@GetMapping("/eCheck")
@ResponseBody
public String eCheck(@RequestParam String email) {
    Member ec = memberRepository.findByEmail(email);
    String emailPattern = "^[a-zA-Z0-9._-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,4}$";
    if (ec != null || (!email.matches(emailPattern))) {
        return "가입불가";
    } else {
        return "사용 가능합니다.";
    }
}

@GetMapping("/nCheck")
@ResponseBody
public String nCheck(@RequestParam String name) {
    Member nc = memberRepository.findByName(name);
    String namePattern = "^[가-힣a-zA-Z0-9]{2,10}$";
    if (nc != null || (!name.matches(namePattern))) {
        return "가입불가";
    } else {
        return "사용 가능합니다.";
    }
}
```

Email , Name 체크

2. 로그인



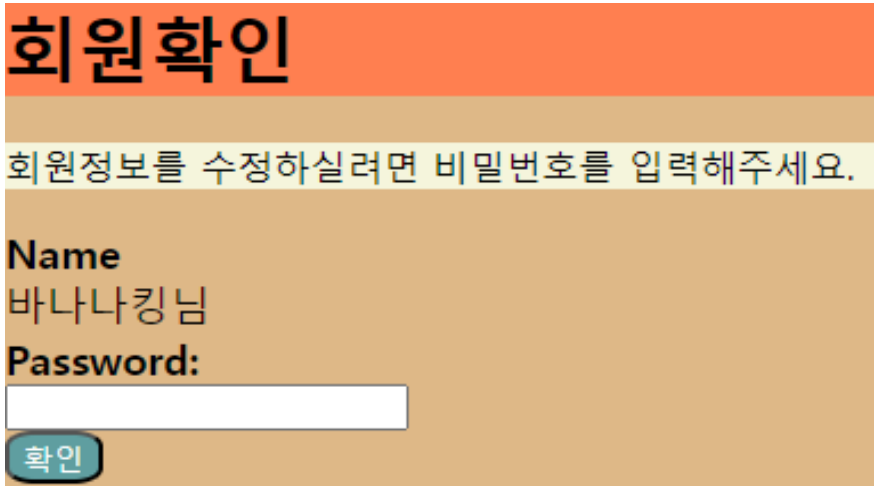
1. Email과 Pw를 입력하여 로그인이 되게 구현했다.

2. 입력된 Pw는 DB에 암호화된 Pw와 match를 사용해 비교했고 같다면 로그인이 된다.

```java
Member existingMember = memberRepository.findByEmail(member.getEmail());
if (existingMember != null) {
    String pw = existingMember.getPw();
    boolean isMatch = passwordEncoder.matches(member.getPw(), pw);
    if (isMatch) {
        session.setAttribute(name:"email", member.getEmail());
        session.setAttribute(name:"pw", member.getPw());
        session.setAttribute(name:"name", existingMember.getName());
        session.setAttribute(name:"id", existingMember.getId());

        response.put(key:"status", value:"success");
        return ResponseEntity.ok(response);
    }
}
```

Match 사용

3. 회원정보 수정



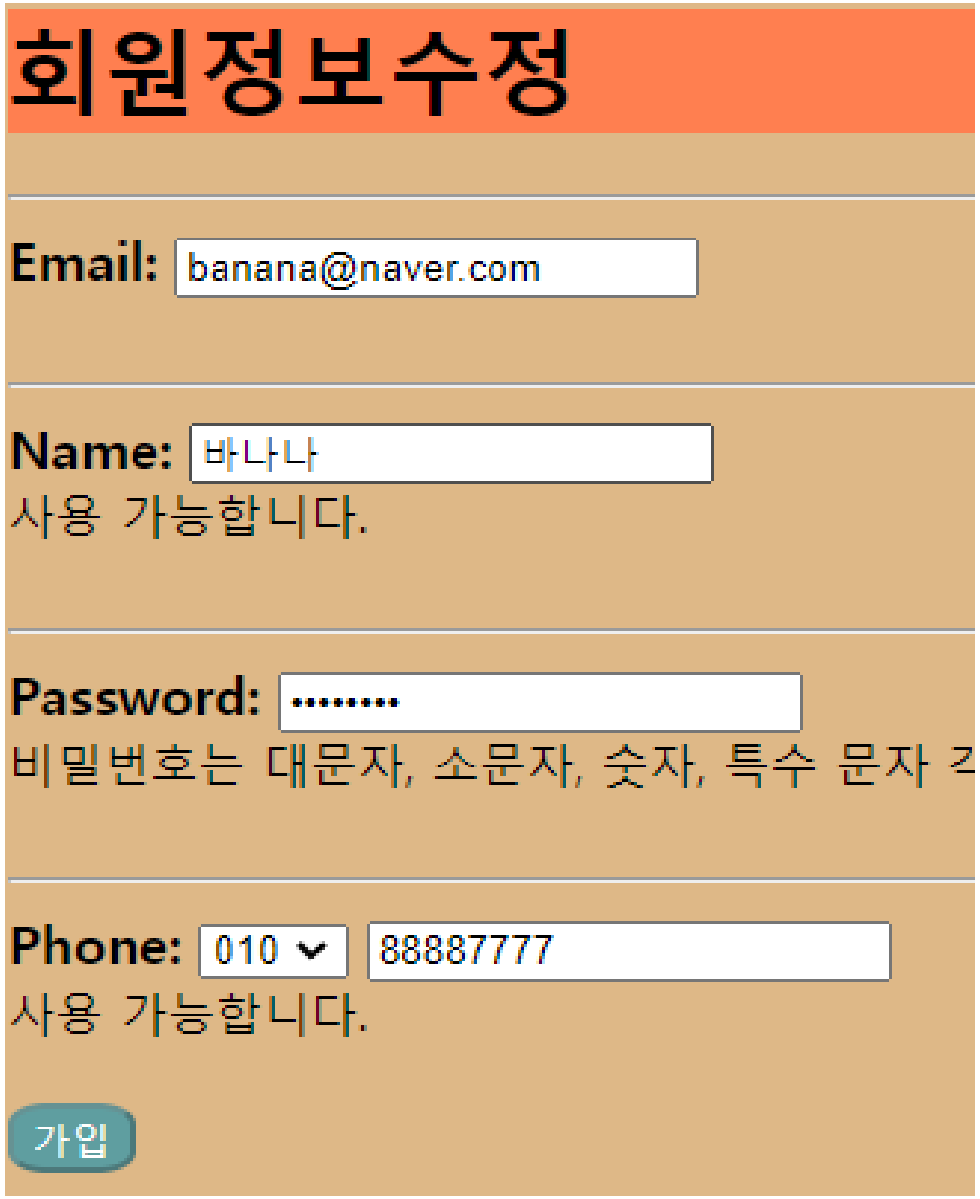
1) 수정 전 확인 페이지

1. 정보수정 전에 실수인지 아닌지 확인 하는 페이지

2. 확인 페이지에서 Pw를 입력 후 진행

3. DB의 유효성에 맞게 진행하여 다른 사람과 중복될 수 없게 로직을 구성하였다.

2. 수정페이지



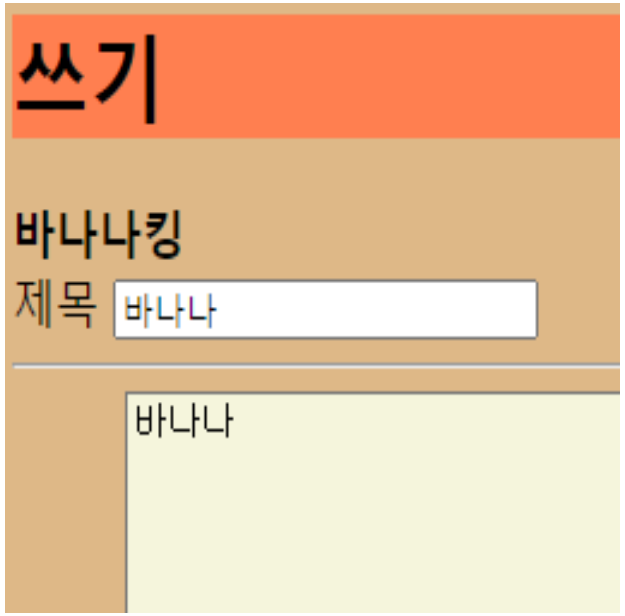
```java
if(existingMember != null) {
    String fullPhoneNumber = phonePrefix + phoneSuffix;
    existingMember.setPhone(fullPhoneNumber);

    existingMember.setName(updateMember.getName());
    if(!existingMember.getPw().equals(updateMember.getPw())) {
        existingMember.setPw(passwordEncoder.encode(updateMember.getPw()));
        session.setAttribute(name:"email", existingMember.getEmail());
        session.setAttribute(name:"pw", existingMember.getPw());
        session.setAttribute(name:"name", existingMember.getName());
    }

memberRepository.save(existingMember);
```

변경된 정보를 Save

4. Board

1) list

2) write



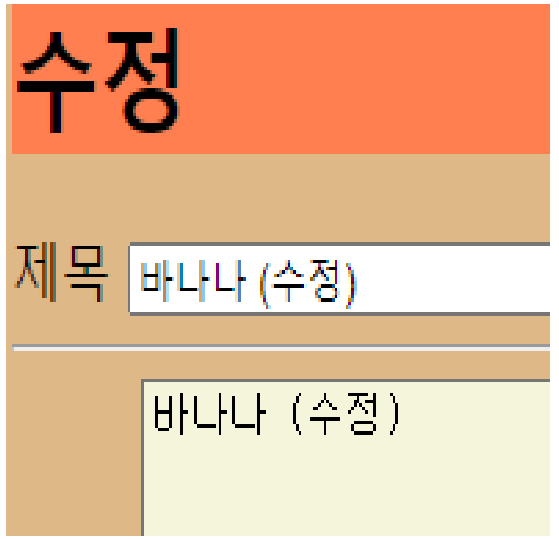
3) 쓴 후

4) 수정, 삭제

1 . 게시물을 올릴 수 있는 페이지를 만들었다.

2. 쓰기 버튼을 눌러 게시글 작성 작성자를 알아볼 수 있게 박제시켜놨다.

3. 게시물을 자기가 쓴 게시물을 클릭하면 내용물 밑에 수 수정, 삭제 버튼으로 수정, 삭제 가능
(작성자가 아니면 숨겨짐)



5) 수정

6) 삭제

```java
@PostMapping("/update")
public String updatePost(
        @ModelAttribute BoardTable board) {
    BoardTable upboard = boardTableRepository.findById(board.getId());
    int id = (int) session.getAttribute(name:"id");
    if (upboard.getMember().getId() == id) {
        session.setAttribute(name:"id", id);
        // LocalDateTime.now().truncatedTo(ChronoUnit.SECONDS);
        board.setBoardData(upboard.getBoardData());
        board.setMember(upboard.getMember());
        boardTableRepository.save(board);
```

변경된 정보를 board에 Save

```java
@GetMapping("/delet")
public String delet (
    @ModelAttribute BoardTable board) {
        int loggedName = (int) session.getAttribute(name:"id");
        BoardTable boardId = boardTableRepository.findById(board.getId());
        System.out.println(loggedName);
        System.out.println(boardId);

        int writer = boardId.getMember().getId();
        if(writer == loggedName) {
            boardTableRepository.delete(boardId);
        }
        System.out.println("erfefvrfgv"+board.getId());
```

Board,member id 와 작성자 id 가 같다면 삭제

```
<button type="button" th:if="${#session.getAttribute('id') == boardTable.member.id}" onclick="update()">수정</b
<button type="button" th:if="${#session.getAttribute('id') == boardTable.member.id}" onclick="remove()">삭제</b
```

Session id 와 board.member.id가 같다면 보여주기

```
<input id="current" type="number" th:value="${boardTable.id}" hidden>
<div th:if="${#session.getAttribute('id') != null}">
    <input id="userId" type="text" th:value="${#session.getAttribute('id')}" hidden>
</div>
<input id="authorId" type="number" th:value="${boardTable.member.id}" hidden>
```

각 키를 id로 설정하여 이용하기

```
function update() {
    if (currentId && userId && authorId && userId === authorId) {
        window.location = `/update?id=${currentId}`;
    }
}

function remove() {
    if (currentId && userId && authorId && userId === authorId) {
        window.location = `/delet?id=${currentId}`;
    }
}
```

If 문에 맞으면 수정 또는 삭제