

1조 프로젝트_2차

팀원:동경민

깃허브 주소: https://github.com/quackkkoduck/team1_project.git

```

@PostMapping("/signup")
@Transactional
public String signupPost(@ModelAttribute User user, BindingResult bindingResult) {
    // 이메일 중복 검사
    List<User> existingUsers = userRepository.findByEmail(user.getEmail());
    if (!existingUsers.isEmpty()) {
        bindingResult.rejectValue(field:"email", errorCode:"error.user",
            defaultMessage:"이미 가입된 이메일입니다.");
        return "signup";
    }

    // 이메일 검사
    String emailPattern = "[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\\.[a-zA-Z]{2,4}$";
    if (!user.getEmail().matches(emailPattern)) {
        bindingResult.rejectValue(field:"email", errorCode:"error.user",
            defaultMessage:"유효하지 않은 이메일 형식입니다.");
        return "signup";
    }
}

```

Email:

aa@naver.com

이미 가입된 이메일입니다

SELECT * FROM "user";

id	email	name	pwd
1	asdf@naver.com	aa	\$2a\$10\$d7s3rVfxKSc
2	aa@naver.com	me	\$2a\$10\$Wxau/Be09C
3	qq@naver.com	qwer	\$2a\$10\$5BVyTOpRR

(3 rows, 2 ms)

Edit

Email:

Password:

! 이메일 주소에 '@'를 포함해 주세요. 'aa'에 '@'가 없습니다.

회원가입

- 가입시 정규식을 활용하여 유효성검사를 통해 이미 가입한 이메일이거나 형식에 맞지 않은 이메일 형태는 가입되지 않도록 설정하였고 이미 데이터베이스에 저장되어있는 이메일은 중복 가입이 불가 하도록 하였습니다.

회원가입

유사한 방식으로 비밀번호도
정규식을 통해 주어진 조건에
맞을 경우에만 가입
가능하도록 하였으며
비밀번호는 encode를
활용하여 암호화시켜 저장
되도록 하였습니다

```
// 비밀번호 검사
String pwdPattern = "^(?=.*[A-Za-z])(?=.*\\d)(?=.*[@$!%*#?&])
[A-Za-z\\d@$!%*#?&]{8,}$";
if (!user.getPwd().matches(pwdPattern)) {
    bindingResult.rejectValue(field:"pwd", errorCode:"error.user",
    defaultMessage:"영문 숫자 특수기호 조합 8자리 이상 입력해 주세요");
    return "signup";
}

// 비밀번호 암호화
String userPwd = user.getPwd();
String encodedPwd = passwordEncoder.encode(userPwd); // 비밀번호
암호화
user.setPwd(encodedPwd); // 암호화된 비밀번호를 설정

userRepository.save(user);

return "redirect:/";
}
```

Email:

asdf@naver.com

Password:

영문 숫자 특수기호 조합 8자리 이상 입력해 주세요

Name:

asdf

Sign Up

SELECT * FROM "user";

id	email	name	pwd
1	asdf@naver.com	aa	\$2a\$10\$d7s3rVfxKScxyd8creM.ceKiKO.kCYxuoMd33.0Od40hYgG9RQ.l

```

@GetMapping("/signin")
public String signin() {
    return "signin";
}

@PostMapping("/signin")
public String signinPost(@ModelAttribute User user, Model model) {
    List<User> dbUsers = userRepository.findByEmail(user.getEmail());

    if (dbUsers != null && !dbUsers.isEmpty()) {
        User dbUser = dbUsers.get(index:0); // 첫 번째 사용자를 선택


        String encodedPwd = dbUser.getPwd();
        String userPwd = user.getPwd();
        boolean isMatch = passwordEncoder.matches(userPwd, encodedPwd);

        if (isMatch) {
            session.setAttribute(name:"user_id", dbUser.getId());
            session.setAttribute(name:"user_email", dbUser.getEmail());

            return "redirect:/"; // 로그인 성공 시 홈 페이지로 리다이렉트
        } else {
            model.addAttribute(attributeName:"error", attributeValue:"Invalid email or password");
            return "signin";
        }
    } else {
        model.addAttribute(attributeName:"error", attributeValue:"User not found");
        return "signin";
    }
}

```

.....


 이메일 주소에 '@'를 포함해 주세요. 'asdf'에 '@'가 없습니다.

Sign In

로그인

로그인은 사용자의 이메일 주소를 사용하여 데이터베이스에서 사용자 정보를 검색해서 dbUsers라는 항목으로 저장 후 null이 아니면서 비어있지 않을 때를 참값으로 설정하였으며 이때 데이터베이스에서 가져온 암호화된 비밀번호와 사용자가 입력한 비밀번호가 일치하였을 때 참값이면 홈으로 이동되게 하였고 로그인 정보가 맞지 않을 때에는 로그인 창에 머물도록 하였습니다.

로그인 또한 이메일 형식에 맞춰서 작성해야 합니다.

```
public class SignInCheckInterceptor implements HandlerInterceptor {
    @Override
    public boolean preHandle(
        HttpServletRequest request, HttpServletResponse response,
        Object handler) throws Exception {
        log.warn(msg:"preHandle");
        HttpSession session = request.getSession();
        Long user = (Long) session.getAttribute(name:"user_id");
        if (user == null) {
            // 사용자가 로그인하지 않은 경우, 로그인 페이지로 이동
            response.sendRedirect(location:"/signin");
            return false;
        }
        return true;
    }
}
```

```
@GetMapping("/usercheck")
public String usercheck() {
    return "usercheck";
}

@PostMapping("/usercheck")
public String usercheck(@ModelAttribute User user, Model model) {
    System.out.println(user);
    String userEmail =(String)session.getAttribute(name:"user_email");
    if (userEmail!=null|| user.getPwd() != null) {
        List<User> dbUser = userRepository.findByEmail(userEmail);
        System.out.println(dbUser);

        String encodedPwd = dbUser.get(index:0).getPwd();
        ;
        String userPwd = user.getPwd();
        boolean isMatch = passwordEncoder.matches(userPwd, encodedPwd);

        if (isMatch) {
            System.out.println(x:1);
            return "redirect:/myPage";
        } else {
            model.addAttribute(attributeName:"error", attributeValue:"Invalid email or password");
            System.out.println(x:2);
            return "usercheck";
        }
    } else {
        model.addAttribute(attributeName:"error", attributeValue:"User not found");
        System.out.println(x:3);
        return "usercheck";
    }
}
```

회원정보수정

회원정보를 수정하기 위해서는
비밀번호를 입력해야 들어갈
수 있게 하였으며 로그인이
안되어 있을 시에는 회원정보
창으로 못들어가도록
하였습니다.

또한 비밀번호 정보를
가져와서 일치하는 지 확인
후 일치했을 시에만 접속
가능 하도록 하였으며 일치
하지 않으면
/usercheck페이지에 머물도록
하였습니다.

password

비밀

확인하기

Email:

asdf@naver.com

Password:

Name:

aa

수정하기 @^v^@

SELECT * FROM "user";

id	email	name	pwd
1	asdf@naver.com	aa	\$2a\$10\$8L8
2	aa@naver.com	me	\$2a\$10\$Wx
3	qq@naver.com	qwer	\$2a\$10\$5B

Email:

asdf@naver.com

Password:

.....

Name:

aa수정

수정하기 @^v^@

SELECT * FROM "user";

id	email	name	pwd
1	asdf@naver.com	aa수정	\$2a\$10\$3e5
2	aa@naver.com	me	\$2a\$10\$Wx
3	qq@naver.com	qwer	\$2a\$10\$5B

회원정보수정

회원정보를 수정하기 위해서는
비밀번호를 입력해야 들어갈
수 있게 하였으며 로그인
안되어 있을 시에는 회원정보
창으로 못들어가도록
하였습니다.

또한 비밀번호 정보를
가져와서 일치하는 지 확인
후 일치했을 시에만 접속
가능 하도록 하였습니다.

게시판_보기

타임리프 식을 적용하여
로그인 하지 않은 경우에
null값이 적용되도록 하여
내용이 보이지 않게
하였습니다.

```
<tr>
  <th>번호</th>
  <th>제목</th>
  <!-- <th>내용</th> -->
  <th>작성자</th>
  <th>내용보기</th>
</tr>
</thead>

<tbody>
<tr th:each="board : ${list}" th:attr="id=${board.id}" th:unless="${session.user_id == null}"
  th:href="@{'/board/update/' + ${board.id}}">
  <td th:text="${board.id}"></td>
  <td th:text="${board.title}"></td>
  <!-- <td th:text="${board.content}"></td> -->
  <td th:text="${board.user.name}"></td>
  <td>
    <a th:unless="${session.user_id == null}" th:href="@{'/board/update/' + ${board.id}}"
      style="color: #4169E1;">상세내용 보기</a>
  </td>
  <td>
    <a th:unless="${session.user_id == null}" th:href="@{'/board/delete/' + ${board.id}}"
      style="color: #DC143C;"
      onclick="return confirm('정말로 삭제하시겠습니까?');">✖</a>
  </td>
</tr>
<tr th:if="${session.user_id == null}">
  <td colspan="5">로그인 후 이용해 주세요</td>
</tr>
</tbody>
</table>

회원가입 menu4 로그인 후 사용해주세요.
```

글쓰기

번호	제목	작성자	내용보기
----	----	-----	------

로그인 후 이용해 주세요			
---------------	--	--	--

게시판_작성 및 수정

```
@GetMapping("/board/update/{id}")
public String boardUpdateForm(Model model, @PathVariable("id") long id) {
    Optional<Board> data = boardRepository.findById(id);
    Board board = data.orElse(other:null);
    if (board == null) {
        return "error-page";
    } else {
        model.addAttribute(attributeName:"board", board);
        return "board/update";
    }
}

@PostMapping("/board/update/{id}")
public String boardUpdate(
    @ModelAttribute Board updatedBoard, @PathVariable("id") long id) {
    Optional<Board> data = boardRepository.findById(id);
    if (data.isPresent()) {
        Board originalBoard = data.get();

        String userEmail = (String) session.getAttribute(name:"user_email");
        if (userEmail != null && userEmail.equals(originalBoard.getUser().getEmail())) {
            originalBoard.setTitle(updatedBoard.getTitle());
            originalBoard.setContent(updatedBoard.getContent());
            boardRepository.save(originalBoard);
            return "redirect:/board/list";
        } else {
            return "redirect:/board/update/{id}";
        }
    } else {
        return "redirect:/board/update/{id}";
    }
}
```

SELECT * FROM "board";

id	content	title	user_id
1	asdf	aa	1
2	aa	hi	3
3	ho!	wow!	3
4	22	11	2

(4 rows, 0 ms)

SELECT * FROM "board";

id	content	title	user_id
1	asdf수정	aa수정	1
2	aa	hi	3
3	ho!	wow!	3
4	22	11	2

(4 rows, 0 ms)

글쓰기

번호	제목	작성자	내용보기	
4	11	me	상세내용 보기	✖
3	wow!	qwer	상세내용 보기	✖
2	hi	qwer	상세내용 보기	✖
1	aa	aa	상세내용 보기	✖

글쓰기

번호	제목	작성자	내용보기	
4	11	me	상세내용 보기	✖
3	wow!	qwer	상세내용 보기	✖
2	hi	qwer	상세내용 보기	✖
1	aa수정	aa	상세내용 보기	✖

Title:
aa수정

Content:
asdf수정

수정완료

Id를 사용하여 수정하려는
게시물을 데이터베이스에서
가져온 후 가져온 게시물을
originalBoard 객체로 저장 하고
현재 사용자의 이메일 주소를
String userEmail =
(String)session.getAttribute("use
r_email")을 통해 가져옵니다.

이후 if문을 통해 현재 사용자의
이메일 주소가 게시물 작성자의
이메일 주소와 일치하는 지
확인합니다. 메소드를 사용하여
제목과 내용을 업데이트하여
데이터 베이스에 저장합니다.

게시판_작성 및 수정

```
@GetMapping("/board/delete/{id}")
public String boardDeleteform(@PathVariable("id") long id) {

    Optional<Board> data = boardRepository.findById(id);
    if (data.isPresent()) {
        Board board = data.get();
        // 현재 사용자의 이메일
        String userEmail = (String) session.getAttribute(name:"user_email");

        if (userEmail != null && userEmail.equals(board.getUser().getEmail())) {
            boardRepository.deleteById(id);
        }
    }
    return "redirect:/board/list";
}

@PostMapping("/board/delete/{id}")
public String boardDelete(@PathVariable("id") long id) {
    System.out.println(id);

    return "redirect:/board/list";
}
```

SELECT * FROM "board";				SELECT * FROM "board";			
id	content	title	user_id	id	content	title	user_id
1	asdf수정	aa수정	1	2	aa	hi	3
2	aa	hi	3	3	ho!	wow!	3
3	ho!	wow!	3	4	22	11	2
4	22	11	2	(3 rows, 0 ms)			
(4 rows, 1 ms)							

Edit

localhost 내용:
정말로 삭제하시겠습니까?

확인 취소

번호	제목	작성자	내용보기	
4	11	me	상세내용 보기	✖
3	wow!	qwer	상세내용 보기	✖
2	hi	qwer	상세내용 보기	✖
1	aa수정	aa	상세내용 보기	✖

회원가입 menu4 로그아웃. asdf

번호	제목	작성자	내용보기	
4	11	me	상세내용 보기	✖
3	wow!	qwer	상세내용 보기	✖
2	hi	qwer	상세내용 보기	✖

또한 boardRepository를 사용하여 주어진 ID에 해당하는 게시글 정보를 검색하여 data객체에 데이터가 존재하는 경우 board객체를 가져옵니다. 이후 세션에서 사용자의 이메일을 가져온 다음 현재 이메일 주소가 게시글 작성자의 이메일과 일치하는 경우에 삭제가 가능하도록 구성하였습니다.

맵

```
<script type="text/javascript"
  src="//dapi.kakao.com/v2/maps/sdk.js?
  appkey=40c033f19396b0dda04ec4450190c95e&libraries=services,
  clusterer,drawing"></script>

<script>
  // 마커를 클릭하면 장소명을 표시할 인포윈도우입니다
  var infowindow = new kakao.maps.InfoWindow({ zIndex: 1 });

  var mapContainer = document.getElementById('map'), // 지도를
  표시할 div
    mapOption = {
      center: new kakao.maps.LatLng(35.1582761, 126.
        7954074), // 지도의 중심좌표
      level: 3 // 지도의 확대 레벨
    };

  // 지도를 생성합니다
  var map = new kakao.maps.Map(mapContainer, mapOption);

  // 장소 검색 객체를 생성합니다
  var ps = new kakao.maps.services.Places();

  // 키워드로 장소를 검색합니다
  ps.keywordSearch('광주 맛집', placesSearchCB);
```

```
// 키워드 검색 완료 시 호출되는 콜백함수 입니다
function placesSearchCB(data, status, pagination) {
  if (status === kakao.maps.services.Status.OK) {

    // 검색된 장소 위치를 기준으로 지도 범위를 재설정하기위해
    // LatLngBounds 객체에 좌표를 추가합니다
    var bounds = new kakao.maps.LatLngBounds();

    for (var i = 0; i < data.length; i++) {
      displayMarker(data[i]);
      bounds.extend(new kakao.maps.LatLng(data[i].y,
        data[i].x));
    }

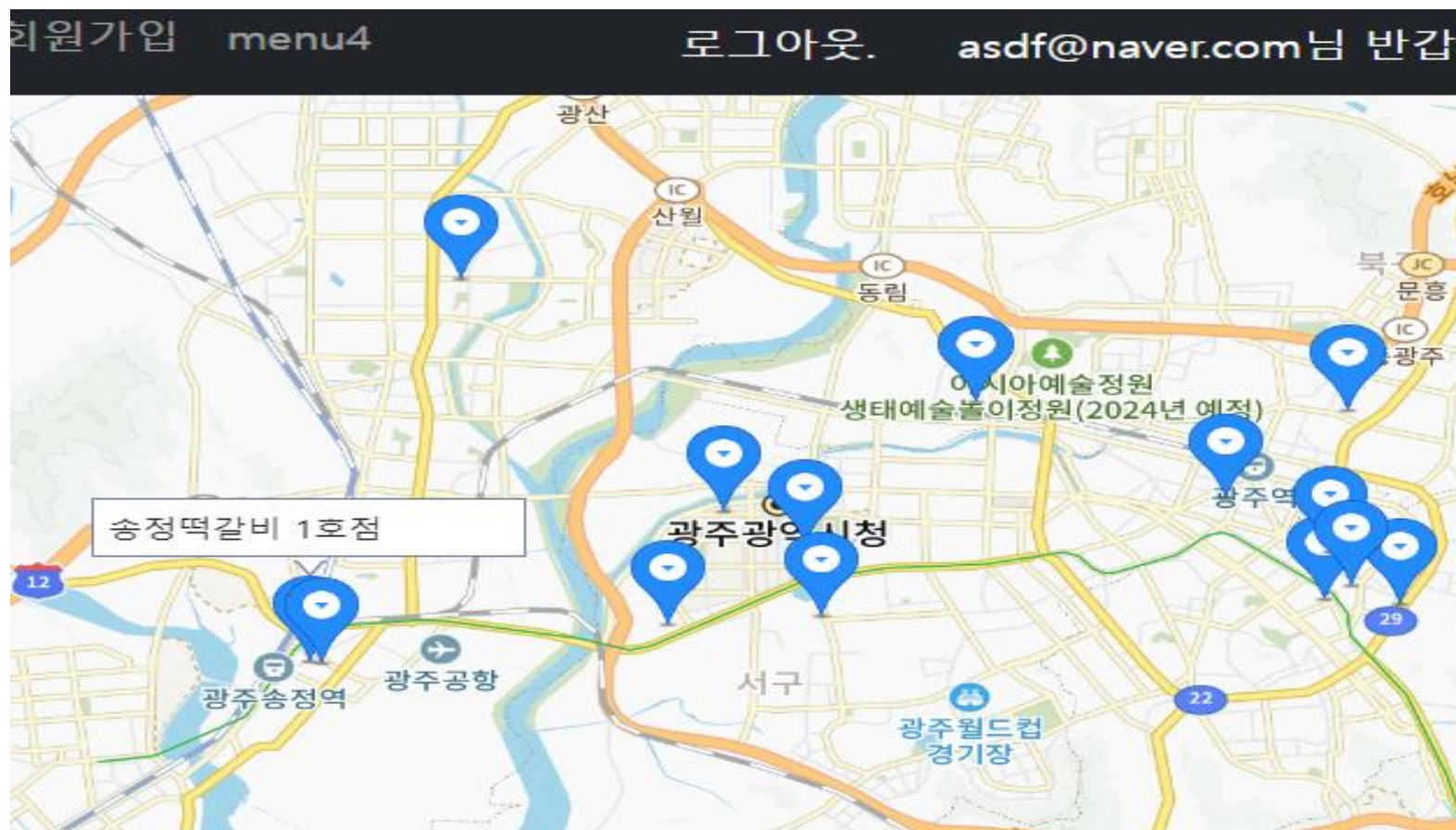
    // 검색된 장소 위치를 기준으로 지도 범위를 재설정합니다
    map.setBounds(bounds);
  }
}

// 지도에 마커를 표시하는 함수입니다
function displayMarker(place) {

  // 마커를 생성하고 지도에 표시합니다
  var marker = new kakao.maps.Marker({
    map: map,
    position: new kakao.maps.LatLng(place.y, place.x)
  });

  // 마커에 클릭이벤트를 등록합니다
  kakao.maps.event.addListener(marker, 'click', function
  () {
    // 마커를 클릭하면 장소명이 인포윈도우에 표출됩니다
    infowindow.setContent('<div style="padding:5px;
      font-size:12px;">' + place.place_name + '</div>');
    infowindow.open(map, marker);
  });
}
```

맵- 호출



1조 프로젝트_2차

The end