

```

@EnableJpaRepositories(basePackageClasses = RepositoryBase.class)
@Configuration
public class JpaConfig {
    @Bean
    LocalContainerEntityManagerFactoryBean entityManagerFactory(DataSource
dataSource){
        LocalContainerEntityManagerFactoryBean emf = new
LocalContainerEntityManagerFactoryBean();
        emf.setDataSource(dataSource);

        emf.setPackagesToScan("com.nhnacademy.springboard.spirngmvcboard.entity");
        emf.setJpaVendorAdapter(jpaVendorAdapter());
        emf.setJpaProperties(jpaProperties());
        return emf;
    }
    private JpaVendorAdapter jpaVendorAdapter(){
        HibernateJpaVendorAdapter hibernateJpaVendorAdapter = new
HibernateJpaVendorAdapter();
        hibernateJpaVendorAdapter.setDatabase(Database.MYSQL);
        return hibernateJpaVendorAdapter;
    }
    private Properties jpaProperties(){
        Properties jpaProperties= new Properties();
        jpaProperties.setProperty("hibernate.show_sql", "true");
        jpaProperties.setProperty("hibernate.format_sql", "true");
        jpaProperties.setProperty("hibernate.use_sql_comments", "true");
        jpaProperties.setProperty("hibernate.globally_quoted_identifiers",
"true");
        jpaProperties.setProperty("hibernate.temp.use_jdbc_metadata_defaults",
"false");

        return jpaProperties;
    }
    @Bean
    public PlatformTransactionManager
transactionManager(EntityManagerFactory entityManagerFactory){
        JpaTransactionManager jpaTransactionManager = new
JpaTransactionManager();
        jpaTransactionManager.setEntityManagerFactory(entityManagerFactory);
        return jpaTransactionManager;
    }
}

```

- com/nhnacademy/springboard/spirngmvcboard/init/DATABASE\_INITIALIZER.java
- init

```
@Slf4j
@Component
public class DatabaseInitializer {

    private UserService userService;
    private BoardService boardService;
    private PostService postService;

    @Autowired
    public DatabaseInitializer(UserService userService, BoardService
boardService, PostService postService) {
        this.userService = userService;
        this.postService=postService;
        this.boardService=boardService;
    }

    @Bean
    public void initializeDatabase() {
        Board board1 = new Board();
        board1.setBoardName("공지사항");
        board1.setDescription("공지사항 게시판입니다.");
        board1.setBoardId(1L);
        if (boardService.getBoard(1L) == null) {
            board1 = boardService.createBoard(board1);
        }

//        if (boardService.findByBoardName("공지사항") == null) {
//
//        }

        log.info("board1 id: {}", board1);
        Board board2 = new Board();
        board2.setBoardName("자유게시판");
        board2.setDescription("자유롭게 글을 올리는 게시판입니다.");
        board2.setBoardId(2L);

        if (boardService.getBoard(2L ) == null) {
            board2 = boardService.createBoard(board2);
        }

        log.info("board2 id: {}", board2);

        User user = new User();
        user.setEmail("admin@nhnacademy.com");
        user.setUsername("admin");
        user.setPassword("1234");
        user.setBirthDate(LocalDate.now());
        user.setCreatedAt(LocalDateTime.now());
        if (userService.getUserByEmail(user.getEmail()) == null) {
            userService.createUser(user);
        }
    }
}
```

```
    } else {

user.setUserId(userService.getUserByEmail(user.getEmail()).getUserId());
    }

    Post post1 = new Post();
    post1.setPostId(1L);
    post1.setAuthor(user);
    post1.setBoard(board1);

    post1.setTitle("서비스 이용 안내");
    post1.setContent("안녕하세요. NHN Academy 웹 사이트를 이용해주시는 여러분께 감사드립니다.");
    post1.setCreatedAt(LocalDateTime.now().minusDays(7));
    post1.setUpdatedAt(LocalDateTime.now().minusDays(7));

    Post post2 = new Post();
    post2.setPostId(2L);
    post2.setAuthor(user);
    post2.setBoard(board2);

    post2.setTitle("나만의 공부법");
    post2.setContent("여러분은 자신만의 공부법이 있으신가요? 제가 추천하는 공부법을 소개합니다.");
    post2.setCreatedAt(LocalDateTime.now().minusDays(5));
    post2.setUpdatedAt(LocalDateTime.now().minusDays(5));

    Post post3 = new Post();
    post3.setPostId(3L);
    post3.setAuthor(user);
    post3.setBoard(board2);
    post3.setTitle("자유롭게 글을 올려보세요");
    post3.setContent("이 게시판은 자유롭게 글을 올리는 공간입니다.");
    post3.setCreatedAt(LocalDateTime.now().minusDays(3));
    post3.setUpdatedAt(LocalDateTime.now().minusDays(3));

    Post post4 = new Post();
    post4.setPostId(4L);
    post4.setAuthor(user);
    post4.setBoard(board2);
    post4.setTitle("NHN IT 교육 서비스 이용 후기");
    post4.setContent("NHN에서 제공하는 IT 교육 서비스를 이용하셨나요? 후기를 남겨보세요!");
    post4.setCreatedAt(LocalDateTime.now().minusDays(1));
    post4.setUpdatedAt(LocalDateTime.now().minusDays(1));
    savePostIfNotExists(post1);
    savePostIfNotExists(post2);
    savePostIfNotExists(post3);
    savePostIfNotExists(post4);

    }

private void savePostIfNotExists(Post post) {
    Post existingPost = postService.getPost(post.getPostId());
```

```

        if (existingPost == null) {
            postService.createPost(post);
        }
    }
}

```

- controller
- com/nhnacademy/springboard/spirngmvcboard/controller/BoardController.java
- com/nhnacademy/springboard/spirngmvcboard/controller/ControllerBase.java
- com/nhnacademy/springboard/spirngmvcboard/controller/LoginController.java
- com/nhnacademy/springboard/spirngmvcboard/controller/MainController.java
- com/nhnacademy/springboard/spirngmvcboard/controller/ManagementUserController.java
- com/nhnacademy/springboard/spirngmvcboard/controller/PostController.java
- com/nhnacademy/springboard/spirngmvcboard/controller/RegistController.java

```

@Controller
@RequestMapping("/board")
public class BoardController implements ControllerBase {
    private final BoardService boardService;
    private final UserService userService;
    private final PostService postService;

    public BoardController(BoardService boardService, PostService
postService, UserService userService) {
        this.boardService = boardService;
        this.postService = postService;
        this.userService=userService;
    }

    @GetMapping
    public String doCategory(Model model) {
        try {
            model.addAttribute("boards", boardService.findAll());
        } catch (Exception e) {
            e.getMessage();
        }
        return "category/category";
    }

    @GetMapping("/{id}")
    public String doBoard(Model model, @PathVariable("id") Long id) {
        log.info("id={}", id);
        Board board = boardService.getBoard(id);
        List<Post> posts = postService.findByBoardId(board.getBoardId());
        model.addAttribute("posts", posts);
        model.addAttribute("board", board); // 이 부분을 추가해줍니다.
        log.info("posts={}", posts);
        return "board/posts";
    }
}

```

```

    }

    @PostMapping("/{boardId}/write")
    public String createPost(@ModelAttribute Post post, HttpSession session,
        @PathVariable Long boardId) {
        User user = userService.getUserById(((User)
        session.getAttribute("user")).getUserId());
        Board board = boardService.getBoard(boardId);
        post.setAuthor(user);
        post.setBoard(board);
        postService.createPost(post);
        return "redirect:/board/" + boardId;
    }

    @GetMapping("/{boardId}/write")
    public String showWriteForm(@PathVariable Long boardId, Model model,
        HttpSession session) {
        User user = (User) session.getAttribute("user");
        if (user == null) {
            try {
                throw new AccessDeniedException("Access is Denied");
            } catch (AccessDeniedException e) {
                throw new RuntimeException(e);
            }
        }
        Post post = new Post();
        model.addAttribute("post", new Post());
        model.addAttribute("board", boardService.getBoard(boardId));
        return "board/write";
    }
}

```

```

package com.nhnacademy.springboard.spirngmvcboard.controller;

public interface ControllerBase {

}

```

```

@Slf4j
@RequestMapping("/")
@Controller
@RequiredArgsConstructor
public class LoginController implements ControllerBase {
    private final LoginService loginService;
    private final UserService userService;
}

```

```
@GetMapping("/login")
public String isLogin(Model model, HttpServletRequest request) {
    HttpSession session = request.getSession(true);
    User user = (User) session.getAttribute("user");
    if (Objects.isNull(user)) {
        return "login/loginForm";
    }
    model.addAttribute("user", user);

    return "main";
}

@PostMapping("/login")
public String doLogin(@ModelAttribute("LoginRequest") LoginRequest
loginRequest, Model model, HttpServletRequest request) {
    try {
        if (loginService.isValidate(loginRequest)) {
            User user = userService.getUserByEmail(loginRequest.getEmail());
            if (user == null) {
                model.addAttribute("errorMessage", "해당 계정이 존재하지 않습니다.");
                return "login/loginForm";
            }
            HttpSession session = request.getSession(false);
            session.setAttribute("user", user);
            return "redirect:/login";
        } else {
            // 로그인 실패 시 에러 메시지를 추가합니다.
            model.addAttribute("errorMessage", "이메일 또는 비밀번호가 잘못되었습니다.");
            return "login/loginForm";
        }
    } catch (Exception e) {
        log.error("Login failed: {}", e.getMessage(), e);
        model.addAttribute("errorMessage", e.getMessage());
        return "login/loginForm";
    }
}

@GetMapping("/logout")
public String doLogout(HttpServletRequest request, HttpServletResponse
response) {
    HttpSession session = request.getSession(false);
    session.removeAttribute("user");
    Cookie cookie = new Cookie("JSESSIONID", "");
    cookie.setMaxAge(0);
    cookie.setValue(null);
    response.addCookie(cookie);
    return "redirect:/login";
}

@GetMapping("/find-password")
public String findPasswordForm(Model model) {
```

```

        model.addAttribute("FindPasswordRequest", new FindPasswordRequest());
        return "user/find-password";
    }

    @PostMapping("/find-password")
    public String findPassword(@ModelAttribute FindPasswordRequest
findPasswordRequest, Model model) {
        User user =
userService.getUserByEmail(findPasswordRequest.getEmail());
        if (user == null) {
            model.addAttribute("errorMessage", "해당 이메일로 가입한 사용자가 존재하지 않습
니다.");
            return "user/find-password";
        }
        model.addAttribute("message", "해당 이메일로 가입한 사용자의 비밀번호는 " +
user.getPassword() + " 입니다.");
        return "user/find-password-result";
    }

    @GetMapping("/find-id")
    public String findIdForm(Model model) {
        model.addAttribute("FindIdRequest", new FindIdRequest());
        return "user/find-id";
    }

    @PostMapping("/find-id")
    public String findId(@ModelAttribute FindIdRequest findIdRequest, Model
model) {
        User user = userService.getUserByEmail(findIdRequest.getEmail());
        if (user == null) {
            model.addAttribute("errorMessage", "해당 이메일로 가입한 사용자가 존재하지 않습
니다.");
            return "user/find-id";
        }
        model.addAttribute("message", "해당 이메일로 가입한 사용자의 아이디는 " +
user.getUsername() + " 입니다.");
        return "user/find-id-result";
    }
}

```

```

@Controller
@RequestMapping("/")
public class MainController implements ControllerBase {
    @GetMapping("/")
    public String mainPage(Model model, HttpServletRequest request) {
        HttpSession session = request.getSession(false);
        if (session != null) {
            User user = (User) session.getAttribute("user");
            if (user != null) {
                model.addAttribute("user", user);
            }
        }
    }
}

```

```
    }  
    }  
    return "main";  
    }  
}
```

```
@Slf4j  
@Controller  
@RequestMapping("/usermanage")  
public class ManagementUserController implements ControllerBase {  
  
    private final UserService userService;  
  
    public ManagementUserController(UserService userService) {  
        this.userService = userService;  
    }  
  
    @GetMapping("/list")  
    public String doUserManage(Model model) {  
        List<User> userList = userService.findAll();  
        model.addAttribute("userlist", userList);  
        return "managementUser";  
    }  
  
    @GetMapping("/delete/{id}")  
    public String doDeleteUser(@PathVariable("id") Long id, Model model) {  
        userService.deleteUserById(id);  
        model.addAttribute("userList", userService.findAll());  
        return "redirect:/usermanage/list";  
    }  
  
    @GetMapping("/update/{userId}")  
    public String validateUpdateUser(@PathVariable("userId") Long userId,  
    Model model) {  
        User user = userService.getUserById(userId);  
        if (user == null) {  
            model.addAttribute("error", "해당 사용자가 없습니다.");  
            return "managementUser";  
        }  
        model.addAttribute("user", user);  
        return "userUpdate";  
    }  
    @PostMapping("/update/{userId}")  
    public String  
doUpdate(@ModelAttribute("ModifyUserRequest")ModifyUserRequest  
modifyUserRequest, @PathVariable("userId") Long userId, Model model) {  
        User existingUser = userService.getUserById(userId);  
        if (existingUser == null) {  
            model.addAttribute("error", "해당 사용자가 없습니다.");  
        }  
    }  
}
```



```

        return "managementUser";
    }
    existingUser.setUsername(modifyUserRequest.getUserName());
    log.info("UserName = {}", modifyUserRequest.getUserName());
    log.info("Password = {}", modifyUserRequest.getPassword());
    existingUser.setPassword(modifyUserRequest.getPassword());
    userService.modifyUser(existingUser);
    model.addAttribute("userList", userService.findAll());
    return "redirect:/usermanage/list";
}
}

```

```

@Controller
@RequestMapping("/board")
public class PostController implements ControllerBase {

    private final PostService postService;
    private final BoardService boardService;
    private final UserService userService;

    public PostController(PostService postService, BoardService
boardService, UserService userService) {
        this.postService = postService;
        this.boardService = boardService;
        this.userService = userService;
    }

    @GetMapping("/{boardId}/post/{postId}")
    public String viewPost(@PathVariable("boardId") Long boardId,
@PathVariable("postId") Long postId, Model model) {
        Post post = postService.getPost(postId);
        if (post == null) {
            try {
                throw new NotFoundException("Post not found with id " + postId);
            } catch (NotFoundException e) {
                throw new RuntimeException(e);
            }
        }

        User author = userService.getUserById(post.getAuthor().getUserId());
        post.setAuthor(author);

        Board board = boardService.getBoard(boardId);
        if (board == null) {
            try {
                throw new NotFoundException("Board not found with id " + boardId);
            } catch (NotFoundException e) {
                throw new RuntimeException(e);
            }
        }
    }
}

```

```
}

post.setBoard(board);
model.addAttribute("board", board);
model.addAttribute("post", post);
return "board/post";
}

@GetMapping("/{boardId}/post/{postId}/edit")
public String showEditForm(@PathVariable("boardId") Long boardId,
@PathVariable("postId") Long postId, Model model, HttpSession session) {
    User user = (User) session.getAttribute("user");
    Post post = postService.getPost(postId);

    // 로그인한 유저와 게시물 작성자가 같은 경우에만 수정 가능
    if (user != null && post.getAuthor().equals(user)) {
        model.addAttribute("post", post);
        return "board/edit";
    } else {
        try {
            throw new AccessDeniedException("Access is Denied");
        } catch (AccessDeniedException e) {
            throw new RuntimeException(e);
        }
    }
}

@PostMapping("/{boardId}/post/{postId}/edit")
public String updatePost(@PathVariable("boardId") Long boardId,
@PathVariable("postId") Long postId, @ModelAttribute Post updatedPost,
HttpSession session) {
    User loginUser = (User) session.getAttribute("user");
    Post post = postService.getPost(postId);

    // 로그인한 유저와 게시물 작성자가 같은 경우에만 수정 가능
    if (loginUser != null && post.getAuthor().equals(loginUser)) {
        post.setTitle(updatedPost.getTitle());
        post.setContent(updatedPost.getContent());
        postService.modifyPost(post);
        return "redirect:/board/" + boardId + "/post/" + postId;
    } else {
        try {
            throw new AccessDeniedException("Access is Denied");
        } catch (AccessDeniedException e) {
            throw new RuntimeException(e);
        }
    }
}

@PostMapping("/{boardId}/post/{postId}/delete")
public String deletePost(@PathVariable("boardId") Long boardId,
@PathVariable("postId") Long postId, HttpSession session) {
    User loginUser = (User) session.getAttribute("user");
    Post post = postService.getPost(postId);
```

```
// 로그인한 유저와 게시물 작성자가 같은 경우에만 삭제 가능
if (loginUser != null && post.getAuthor().equals(loginUser)) {
    postService.deletePostById(post.getPostId());
    return "redirect:/board/" + boardId;
} else {
    try {
        throw new AccessDeniedException("Access is Denied");
    } catch (AccessDeniedException e) {
        throw new RuntimeException(e);
    }
}
}
}
```

```
@Slf4j
@Controller
@RequestMapping("/")
public class RegistController {
    private final UserService userService;

    public RegistController(UserService userService) {
        this.userService = userService;
    }

    @PostMapping("/regist")
    public String doRegist(@ModelAttribute("user") @DateTimeFormat(pattern = "yyyy-MM-dd") User user, Model model) {
        User existingUser = userService.getUserByEmail(user.getEmail());
        if (existingUser != null) {
            model.addAttribute("error", "이미 존재하는 사용자 이메일입니다. 다른 사용자 이메일을 선택해주세요.");
            log.info("User already exists: {}", user.getUsername());
            return "userRegist";
        }
        User newUser = new User();
        newUser.setUsername(user.getUsername());
        newUser.setEmail(user.getEmail());
        newUser.setPassword(user.getPassword());
        newUser.setCreatedAt(user.getCreatedAt());
        newUser.setBirthDate(user.getBirthDate());
        try {
            userService.createUser(newUser);
        } catch (Exception e) {
            model.addAttribute("error", "등록 중 오류가 발생했습니다. 다시 시도해주세요.");
            return "userRegist";
        }

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

```
@GetMapping("/regist")
public String doRegist() {
    return "userRegist";
}
```

- service
- com/nhnacademy/springboard/spirngmvcboard/service/BoardService.java
- com/nhnacademy/springboard/spirngmvcboard/service/PostService.java
- com/nhnacademy/springboard/spirngmvcboard/service/UserService.java

```
public interface UserService {
    User createUser(User user);
    User getUser(String email);
    User modifyUser(User user);
}
```

```
public interface PostService {
    Post createPost(Post post);
    Post getPost(String postId);
    Post modifyPost(Post post);
    List<Post> findByUserId(String userId);
    List<Post> findByBoardId(String boardId);
}
```

```
public interface BoardService {
    Board createBoard(Board board);
    Board getBoard(String boardId);
    Board modifyBoard(Board board);
    List<Board> findAll();
}
```

- service.impl
- com/nhnacademy/springboard/spirngmvcboard/service/impl/BoardServiceImpl.java
- com/nhnacademy/springboard/spirngmvcboard/service/impl/PostServiceImpl.java
- com/nhnacademy/springboard/spirngmvcboard/service/impl/UserServiceImpl.java

```
@Service
@RequiredArgsConstructor
public class BoardServiceImpl implements BoardService {
    private final BoardRepository boardRepository;

    @Override
    public Board createBoard(Board board) {
        return boardRepository.save(board);
    }

    @Override
    public Board getBoard(Long boardId) {
        Optional<Board> boardOptional = boardRepository.findById(boardId);
        return boardOptional.orElse(null);
    }

    @Override
    public Board modifyBoard(Board board) {
        Optional<Board> boardOptional =
boardRepository.findById(board.getBoardId());
        if (boardOptional.isPresent()) {
            Board existingBoard = boardOptional.get();
            existingBoard.setBoardName(board.getBoardName());
            existingBoard.setDescription(board.getDescription());
            return boardRepository.save(existingBoard);
        } else {
            return null;
        }
    }

    @Override
    public Board findByBoardName(String boardName) {
        Board board=boardRepository.findByBoardName(boardName).orElse(null);
        log.info("findByBoardName: {}", board.getBoardName());
        return boardRepository.findByBoardName(boardName).orElse(null);
    }

    @Override
    public List<Board> findAll() {
        return boardRepository.findAll();
    }
}
```

## PostServiceImpl.java

---

```
@Service
@RequiredArgsConstructor
@Transactional
public class PostServiceImpl implements PostService {
```

```
private final PostRepository postRepository;
private final UserRepository userRepository;
private final BoardRepository boardRepository;

@Override
public Post createPost(Post post) {

    return postRepository.save(post);
}
@Override
public Post getPost(Long postId) {
    Optional<Post> postOptional = postRepository.findById(postId);
    return postOptional.orElse(null);
}

@Override
public Post modifyPost(Post post) {
    Optional<Post> postOptional =
postRepository.findById(post.getPostId());
    if (postOptional.isPresent()) {
        Post existingPost = postOptional.get();
        existingPost.setTitle(post.getTitle());
        existingPost.setContent(post.getContent());
        return postRepository.save(existingPost);
    } else {
        return null;
    }
}

@Override
public List<Post> findByUserId(Long userId) {
    List<Post> allPosts = postRepository.findByAuthor_UserId(userId);
    return allPosts;
}

@Override
public List<Post> findByBoardId(Long boardId) {
    List<Post> allPosts = postRepository.findByBoard_BoardId(boardId);
    return allPosts;
}

public Optional<Post> findById(Long postId) {
    return postRepository.findById(postId);
}

@Override
public void deletePostById(Long postId) {
    postRepository.deleteById(postId);
}
}
```

# UserServiceImpl.java

---

```
@Slf4j
@Service
@RequiredArgsConstructor
public class UserServiceImpl implements UserService {

    private final UserRepository userRepository;

    @Override
    public User createUser(User user) {
        return userRepository.save(user);
    }

    @Override
    public User getUserById(Long userId) {
        log.info("userId : {}", userId);
        return userRepository.findById(userId).orElse(null);
    }

    @Override
    public User getUserByEmail(String email) {
        List<User> allUsers = userRepository.findAll();
        if (allUsers.isEmpty()) {
            return null;
        } else {
            Optional<User> userByEmail = allUsers.stream()
                .filter(user -> user.getEmail().equals(email))
                .findFirst();
            return userByEmail.orElse(null);
        }
    }

    @Override
    public User modifyUser(User user) {
        return userRepository.save(user);
    }

    @Override
    public List<User> findAll() {
        return userRepository.findAll();
    }

    @Override
    public void deleteUserById(Long userId) {
        userRepository.deleteById(userId);
    }
}
```

```
@Getter
@Setter
public class FindPasswordRequest {
    private String email;
    private String password;
}
```

```
@Getter
@Setter
public class LoginRequest {
    @NotBlank(message="이메일을 입력하세요")
    private String email;
    @NotBlank(message="비밀번호를 입력하세요")
    private String pwd;

    @Getter
    @Setter
    public static class FindIdRequest {
        private String email;

        // 생성자, getter, setter
    }
}
```

```
@Slf4j
@Service
@RequiredArgsConstructor
public class LoginService {
    private final UserMapper userMapper;

    public boolean isValid(LoginRequest loginRequest) {
        User user = userMapper.findByUseremail(loginRequest.getEmail());
        boolean result = user != null &&
        user.getPassword().equals(loginRequest.getPwd());
        log.info("isValid : {}", result);
        return result;
    }
}
```



- com/nhnacademy/springboard/spirngmvcboard/thymeleaf/CustomTagDialect.java
- com/nhnacademy/springboard/spirngmvcboard/thymeleaf/TagUtils.java

```
//todo#15 CustomTagDialect 생성 , 커스텀 함수를 사용하기위해서 IDialect 확장
IExpressionObjectDialect 구현.
public class CustomTagDialect extends AbstractDialect implements
IExpressionObjectDialect {

    public CustomTagDialect() {
        super("nhnacademy");
    }

    @Override
    public IExpressionObjectFactory getExpressionObjectFactory() {
        return new SpringStandardExpressionObjectFactory(){
            @Override
            public Set<String> getAllExpressionObjectNames() {
                return Collections.singleton("nhnacademy");
            }

            @Override
            public Object buildObject(IExpressionContext context, String
expressionObjectName) {
                super.buildObject(context, expressionObjectName);
                //todo#15 미리 만들어 뒀던 TagUtils 객체;
                return new TagUtils();
            }

            @Override
            public boolean isCacheable(String expressionObjectName) {
                return true;
            }
        };
    }
}
```

```
public class TagUtils {
    public String gender(Gender gender){
        if(gender.name().equals("M")){
            return "남성";
        } else if (gender.name().equals("F")) {
            return "여성";
        }else {
            return "";
        }
    }
}
```

```

public interface BoardRepository extends JpaRepository<Board, Long> {
    Optional<Board> findByBoardName(String boardName);
}

@Repository
public interface PostRepository extends JpaRepository<Post, Long> {

    List<Post> findByAuthor_UserId(Long userId);

    List<Post> findByBoard_BoardId(Long boardId);

    Optional<Post> findByPostId(Long postId);
}

public interface RepositoryBase {
}

public interface UserRepository extends JpaRepository<User, Long> {
}

```

- com/nhnacademy/springboard/spirngmvcboard/repository/BoardRepository.java
- com/nhnacademy/springboard/spirngmvcboard/repository/PostRepository.java
- com/nhnacademy/springboard/spirngmvcboard/repository/RepositoryBase.java
- com/nhnacademy/springboard/spirngmvcboard/repository/UserRepository.java

```

public interface BoardRepository extends JpaRepository<Board, Long> {
    Optional<Board> findByBoardName(String boardName);
}

```

```

@Repository
public interface PostRepository extends JpaRepository<Post, Long> {

    List<Post> findByAuthor_UserId(Long userId);

    List<Post> findByBoard_BoardId(Long boardId);

    Optional<Post> findByPostId(Long postId);
}

```

```
public interface RepositoryBase {  
  
}
```

```
public interface UserRepository extends JpaRepository<User, Long> {  
  
}
```

- com/nhnacademy/springboard/spirngmvcboard/entity/Board.java
- com/nhnacademy/springboard/spirngmvcboard/entity/Post.java
- com/nhnacademy/springboard/spirngmvcboard/entity/User.java

```
@Entity  
@Table(name = "Boards")  
@Getter  
@Setter  
@NoArgsConstructor  
public class Board {  
    @Id  
    @Column(name = "board_id")  
    @GeneratedValue(strategy= GenerationType.IDENTITY)  
    private Long boardId;  
    @Column(name = "board_name")  
    private String boardName;  
    private String description;  
  
}
```

```
@Entity  
@Table(name = "Posts")  
@Getter  
@Setter  
@EqualsAndHashCode  
@ToString  
@NoArgsConstructor  
public class Post {  
    @Id  
    @Column(name = "post_id")  
    @GeneratedValue(strategy = GenerationType.IDENTITY)  
    private Long postId;  
    @ManyToOne(fetch = FetchType.EAGER)  
    @JoinColumn(name = "author_id")  
    private User author;
```

```

@ManyToOne(fetch = FetchType.EAGER)
@JoinColumn(name = "board_id")
private Board board;

private String title;
private String content;
@Column(name = "created_at")
private LocalDateTime createdAt = LocalDateTime.now();
@Column(name = "updated_at")
private LocalDateTime updatedAt = LocalDateTime.now();

}

```

```

@ToString
@Getter
@Setter
@Entity
@Table(name = "Users")
@NoArgsConstructor
public class User {
    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    @Column(name = "user_id")
    private Long userId;
    private String email;
    @Column(name = "username")
    private String username;
    private String password;
    @Column(name = "birth_date")
    private LocalDate birthDate;
    @Column(name = "created_at")
    private LocalDateTime createdAt = LocalDateTime.now();
}

```

## html

---

- src/main/webapp/WEB-INF
- src/main/webapp/WEB-INF/views
- src/main/webapp/WEB-INF/views/board
- src/main/webapp/WEB-INF/views/board/edit.html
- src/main/webapp/WEB-INF/views/board/post.html
- src/main/webapp/WEB-INF/views/board/posts.html

- src/main/webapp/WEB-INF/views/board/write.html
- src/main/webapp/WEB-INF/views/category
- src/main/webapp/WEB-INF/views/category/category.html
- src/main/webapp/WEB-INF/views/login
- src/main/webapp/WEB-INF/views/login/loginForm.html
- src/main/webapp/WEB-INF/views/user
- src/main/webapp/WEB-INF/views/user/find-id.html
- src/main/webapp/WEB-INF/views/user/find-id-result.html
- src/main/webapp/WEB-INF/views/user/find-password.html
- src/main/webapp/WEB-INF/views/user/find-password-result.html
- src/main/webapp/WEB-INF/views/loginInfo.html
- src/main/webapp/WEB-INF/views/main.html
- src/main/webapp/WEB-INF/views/managementUser.html
- src/main/webapp/WEB-INF/views/userRegist.html
- src/main/webapp/WEB-INF/views/userUpdate.html

```

/*공통 스타일*/
body {
    font-family: Arial, sans-serif;
    background-color: #f8f9fa;
    margin: 0;
}

.container {
    background-color: #ffffff;
    padding: 30px;
    border-radius: 5px;
    box-shadow: 0px 0px 5px 0px rgba(0, 0, 0, 0.1);
    max-width: 800px;
    margin: 50px auto;
}

.link, .auth-links a, .user-info a, .user-table a {
    text-decoration: none;
    color: #3498db;
    font-size: 14px;
}

.link:hover, .auth-links a:hover, .user-info a:hover, .user-table a:hover
{
    text-decoration: underline;
}

.auth-links {
    display: flex;
    justify-content: center;
    gap: 20px;
}

/*로그인 정보 스타일*/

```

```
.login-info {
  display: flex;
  justify-content: space-between;
  align-items: center;
  flex-wrap: wrap;
  margin-bottom: 20px;
}

.login-info ul {
  list-style: none;
  padding: 0;
  margin: 0;
  display: flex;
  gap: 10px;
}

.login-info li {
  display: inline;
}

/*메인 페이지 스타일*/
.main-title {
  text-align: center;
  margin-bottom: 30px;
}

.welcome-message {
  font-weight: bold;
  margin-bottom: 10px;
}

.user-id {
  font-weight: bold;
}

.link.user-list, .link.logout {
  display: inline-block;
  margin-right: 10px;
}

/*유저 리스트 페이지 스타일*/
.user-row:hover {
  background-color: #f8f9fa;
}

.link.update, .link.delete {
  font-weight: bold;
}

.table {
  margin-bottom: 30px;
}

.user-info {
  text-align: center;
```

```
}

.user-info a {
  margin-left: 15px;
}

/*로그인 및 회원가입 페이지 스타일*/
.form {
  width: 300px;
  margin: 50px auto;
}

.form input {
  display: block;
  width: 100%;
  padding: 10px;
  margin-bottom: 10px;
  border-radius: 5px;
  border: 1px solid #ccc;
}

.form label {
  font-size: 14px;
  font-weight: bold;
  margin-bottom: 5px;
  display: block;
}

.form button {
  width: 100%;
  display: block;
  margin-top: 10px;
}

/*유저 리스트 페이지 스타일*/
.user-list-title {
  text-align: center;
  margin-bottom: 30px;
}

.user-table {
  width: 100%;
  margin-bottom: 30px;
  border-collapse: collapse;
}

.user-table th, .user-table td {
  padding: 10px;
  text-align: left;
  border: 1px solid #ccc;
}

.user-table th {
  background-color: #3498db;
```

```
    color: #fff;
}

/*추가 및 수정된 스타일 코드*/
# wrapper {
    display: flex;
    flex-direction: column;
    align-items: center;
    justify-content: center;
    min-height: 100vh;
}

.home-link {
    position: absolute;
    top: 20px;
    left: 20px;
    font-size: 16px;
    font-weight: bold;
    background-color: #3498db;
    color: #fff;
    padding: 10px 20px;
    border-radius: 5px;
    text-decoration: none;
}

.link.home:hover {
    background-color: #2980b9;
}

.form {
    background-color: #fff;
    padding: 30px;
    border-radius: 5px;
    box-shadow: 0px 0px 5px 0px rgba(0, 0, 0, 0.1);
}

.form .input-group {
    margin-bottom: 15px;
}

.form .input-group label {
    display: block;
    font-size: 14px;
    font-weight: bold;
    margin-bottom: 5px;
}

.form .input-group input {
    display: block;
    width: 100%;
    padding: 10px;
    border-radius: 5px;
    border: 1px solid #ccc;
}
```



```
.form button {
  background-color: #3498db;
  color: #fff;
  padding: 10px;
  border-radius: 5px;
  font-size: 16px;
  font-weight: bold;
  border: none;
  cursor: pointer;
  transition: background-color 0.3s;
}

.form button:hover {
  background-color: #2980b9;
}

.form button:active {
  transform: translateY(2px);
}

/*게시판 목록 스타일*/
.board-list {
  display: flex;
  flex-direction: column;
  gap: 10px;
}

.board-list a {
  font-size: 16px;
  font-weight: bold;
  color: #3498db;
}

.board-list a:hover {
  text-decoration: underline;
}
```

## post.html

---

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>게시물 상세보기</title>
</head>
<body>
<h1>게시물 상세보기</h1>
<table>
  <tr>
    <th>게시물 ID</th>
```

```

        <td th:text="${post.id}"></td>
    </tr>
</tr>
    <th>게시판 ID</th>
    <td th:text="${post.board.name}"></td>
</tr>
</tr>
    <th>제목</th>
    <td th:text="${post.title}"></td>
</tr>
</tr>
    <th>작성자</th>
    <td th:text="${post.author.username}"></td>
</tr>
</tr>
    <th>내용</th>
    <td th:text="${post.content}"></td>
</tr>
</tr>
    <th>작성일시</th>
    <td th:text="${post.createdAt}"></td>
</tr>
</table>
<a class="link" th:href="@{/board/{boardId}(boardId=${board.id})}"
th:text="돌아가기"></a>

<!-- 현재 로그인한 유저가 작성한 글인 경우에만 수정/삭제 버튼을 보여줌 -->
<div th:if="${session.user != null and session.user.id ==
post.author.id}">
    <a th:href="@{/board/{boardId}/post/{postId}/edit(boardId=${board.id},
postId=${post.id})}">
        <button>수정</button>
    </a>

    <form th:action="@{/board/{boardId}/deletePost(postId=${post.id},
boardId=${board.id})}" method="POST">
        <button type="submit">삭제</button>
    </form>
</div>

</body>
</html>

```

## posts.html

```

<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head>

```

```

<meta charset="UTF-8">
<link rel="stylesheet" th:href="@{/resources/style.css}" type="text/css"
/>
<title>게시물 목록</title>
</head>
<body>
<div id="wrapper">
  <a href="/" class="home-link">홈</a>
  <div class="container">
    <h1 class="main-title">게시물 목록</h1>

    <!-- 로그인 유저 정보 -->
    <div th:replace="loginInfo :: userinfo"></div>

    <table class="post-table">
      <thead>
        <tr>
          <th>글번호</th>
          <th>제목</th>
          <th>작성자</th>
          <th>작성일</th>
        </tr>
      </thead>
      <tbody>
        <tr th:each="post : ${posts}">
          <td th:text="${post.id}"></td>
          <td>
            <a class="post-link" th:if="${post.board != null}"
th:href="@{/board/{boardId}/post/{postId}"
(boardId=${post.board.id},postId=${post.id})" th:text="${post.title}">
</a>
            </td>
          <td th:text="${post.author.username}"></td>
          <td th:text="${#temporals.format(post.createdAt, 'yyyy-MM-dd
HH:mm:ss')}"></td>
        </tr>
      </tbody>
    </table>
  </div>
  <div th:if="${session.user != null}">
    <a th:href="@{/board/{boardId}/write(boardId=${board.id})}" class="btn
write-btn">글쓰기</a>
  </div>

</div>
</body>
</html>

```

## write.html

```

<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>글쓰기 폼</title>
  <link rel="stylesheet" th:href="@{/resources/style.css}" type="text/css"
/>
</head>
<body>
<div id="wrapper">
  <a href="/" class="home-link">홈으로</a>
  <div th:with="boardId=${board.id}">

    <form th:method="POST"
th:action="@{/board/{boardId}/write(boardId=${boardId})}"
th:object="${post}" class="post-form">
      <div class="input-group">
        <label for="input-title" class="input-label">제목 :</label>
        <input name="title" type="text" placeholder="제목을 입력하세요."
required id="input-title">
      </div>
      <div class="input-group">
        <label for="input-content" class="input-label">내용 :</label>
        <textarea name="content" rows="10" placeholder="내용을 입력하세요."
required id="input-content"></textarea>
      </div>
      <input name="authorId" type="hidden" th:value="${session.user.id}">
      <input name="board.id" type="hidden" th:value="${board.id}">
      <button type="submit" id="submit-btn">글쓰기</button>
    </form>

  </div>
  <!-- 에러 메시지 출력 부분 -->
  <div th:if="${errorMessage}" class="alert alert-danger mt-3"
role="alert">
    <span th:text="${errorMessage}">Error message goes here</span>
  </div>

</div>
</body>
</html>

```

## edit.html

```

<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>게시물 수정</title>

```

```

<link rel="stylesheet" th:href="@{/resources/style.css}" type="text/css"
/>
</head>
<body>
<div id="wrapper">
  <h1>게시물 수정</h1>
  <form
th:action="@{/board/{boardId}/post/{postId}/edit(boardId=${post.board.id},
postId=${post.id})}" method="post">
    <div class="input-group">
      <label for="input-title" class="input-label">제목 :</label>
      <input name="title" type="text" th:value="${post.title}"
placeholder="제목을 입력하세요." required id="input-title">
    </div>

    <div class="input-group">
      <label for="input-content" class="input-label">내용 :</label>
      <textarea name="content" rows="10" placeholder="내용을 입력하세요."
required id="input-content" th:text="${post.content}"></textarea>
    </div>

    <button type="submit" id="submit-btn">수정</button>
  </form>

  <!-- 에러 메시지 출력 부분 -->
  <div th:if="${errorMessage}" class="alert alert-danger mt-3"
role="alert">
    <span th:text="${errorMessage}">Error message goes here</span>
  </div>
</div>
</body>
</html>

```

## category.html

```

<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <link rel="stylesheet" th:href="@{/resources/style.css}" type="text/css"
/>
  <title>Title</title>
</head>
<body>
<div id="wrapper">
  <a href="/" class="home-link">홈으로</a>
  <div class="container" th:fragment="category">
    <h1 class="main-title">게시판</h1>
    <div class="board-list" th:each="board: ${boards}" >
      <a class="link" th:href="@{/board/{boardId}(boardId=${board.id})}"

```

```

th:text="${board.name}"></a>
    </div>
</div>
</div>
</body>
</html>

```

## loginForm.html

---

```

<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" th:href="@{/resources/style.css}" type="text/css"
/>
    <title>Title</title>
</head>
<body>
<div id="wrapper">
    <a href="/" class="home-link">홈으로</a>
    <div class="container" th:fragment="category">
        <h1 class="main-title">게시판</h1>
        <div class="board-list" th:each="board: ${boards}" >
            <a class="link" th:href="@{/board/{boardId}(boardId=${board.id})}"
th:text="${board.name}"></a>
        </div>
    </div>
</div>
</body>
</html>

```

## find-id.html

---

```

<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <link rel="stylesheet" th:href="@{/resources/style.css}" type="text/css"
/>
    <title>Title</title>
</head>
<body>
<div id="wrapper">
    <a href="/" class="home-link">홈으로</a>

```

```

<div class="container" th:fragment="category">
  <h1 class="main-title">게시판</h1>
  <div class="board-list" th:each="board: ${boards}" >
    <a class="link" th:href="@{/board/{boardId}(boardId=${board.id})}"
th:text="${board.name}"></a>
  </div>
</div>
</div>
</body>
</html>

```

## find-id-result.html

```

<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>아이디 찾기 결과</title>
  <link rel="stylesheet" th:href="@{/resources/style.css}" type="text/css"
/>
</head>
<body>
<div id="wrapper">
  <a href="/" class="home-link">홈</a>

  <div class="message">
    <span th:text="${message}"></span>
  </div>

  <!-- 에러 메시지 출력 부분 -->
  <div th:if="${errorMessage}" class="alert alert-danger mt-3"
role="alert">
    <span th:text="${errorMessage}">Error message goes here</span>
  </div>
</div>
</body>
</html>

```

## find-password.html

```

<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">

```

```

<title>비밀번호 찾기</title>
<link rel="stylesheet" th:href="@{/resources/style.css}" type="text/css"
/>
</head>
<body>
<div id="wrapper">
  <a href="/" class="home-link">홈</a>

  <form th:method="POST" th:action="@{/find-password}"
th:object="${FindPasswordRequest}" class="find-form">
    <div class="input-group">
      <label for="input-email" class="input-label">이메일 :</label>
      <input name="email" type="email" placeholder="example@gmail.com"
required id="input-email">
    </div>

    <button type="submit" id="submit-btn">비밀번호 찾기</button>
  </form>

  <!-- 에러 메시지 출력 부분 -->
  <div th:if="${errorMessage}" class="alert alert-danger mt-3"
role="alert">
    <span th:text="${errorMessage}">Error message goes here</span>
  </div>

</div>
</body>
</html>

```

## find-password-result

```

<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head>
  <meta charset="UTF-8">
  <title>비밀번호 찾기 결과</title>
  <link rel="stylesheet" th:href="@{/resources/style.css}" type="text/css"
/>
</head>
<body>
<div id="wrapper">
  <a href="/" class="home-link">홈</a>

  <div th:if="${message}" class="alert alert-success mt-3" role="alert">
    <span th:text="${message}">Success message goes here</span>
  </div>

  <div th:if="${errorMessage}" class="alert alert-danger mt-3"
role="alert">
    <span th:text="${errorMessage}">Error message goes here</span>
  </div>

```



```

    </div>

</div>
</body>
</html>

```

## login-info.html

```

<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<body>
<div th:fragment="userinfo">
    <div class="login-info">
        <ul>
            <li><span th:text="${#messages.msg('info.id')}" /> : <span
th:text="${user?.id}" ></span></li>
            <li><span th:text="${#messages.msg('info.name')}" /> : <span
th:text="${user?.username}" ></span></li>
            <li>
                <button type="button" onclick="location.href=location.pathname +
'?locale=ko' ">한국어</button>
            </li>
            <li>
                <button type="button" onclick="location.href=location.pathname +
'?locale=en' ">영어</button>
            </li>
        </ul>
    </div>
</div>
</body>
</html>

```

## main.html

```

<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Login</title>
    <link rel="stylesheet" th:href="@{/resources/style.css}" type="text/css"
/>
</head>
<body>
<div id="wrapper">
    <div class="container">
        <h1 id="main-title">MAIN</h1>

```

```

<th:block th:insert="~{loginInfo :: userinfo}" ></th:block>

<div th:if="${user != null}" class="user-info">
    <p class="welcome-message">환영합니다. <span th:text="${user.username}"
class="user-id"></span></p>
    <a href="/usermanage/list" class="user-list-link">유저리스트</a>
    <a href="/board" class="board-link">게시판</a>
    <a href="/logout" class="logout-link">로그아웃</a>
</div>

<div th:if="${user == null}" class="auth-links">
    <a href="/login" class="login-link">로그인</a>
    <a href="/regist" class="register-link">회원가입</a>
</div>

<script>
    var user = [[${user != null} ? ${user} : 'null']];
    console.log(user);
</script>

</div>
</div>
</body>
</html>

```

## managementUser.html

```

<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>User List</title>
    <link rel="stylesheet" th:href="@{/resources/style.css}" type="text/css"
/>
</head>
<body>
<th:block th:insert="~{loginInfo :: userinfo}" ></th:block>

<a href="/" class="home-link">홈</a>

<div class="container">
    <h1 class="mt-5 mb-5 user-list-title">User List</h1>
    <table class="table table-bordered table-striped user-list-table">
        <thead>
            <tr>
                <th>ID</th>
                <th>Name</th>
                <th>Email</th>
                <th>Birth Date</th>
                <th>Created At</th>
            </tr>

```

```

        <th>Update</th>
        <th>Delete</th>
    </tr>
</thead>
<tbody>
<tr th:each="user : ${userlist}" class="user-row">
    <td th:text="${user.id}" class="user-id"></td>
    <td th:text="${user.username}" class="user-name"></td>
    <td th:text="${user.email}" class="user-email"></td>
    <td th:text="${user.birthDate}" class="user-birthDate"></td>
    <td th:text="${user.createdAt}" class="user-createdAt"></td>
    <td>
        <a th:href="@{/usermanage/update/{id}(id=${user.id})}"
class="update-link">Update</a>
    </td>
    <td>
        <a th:href="@{/usermanage/delete/{id}(id=${user.id})}"
onclick="return confirm('정말로 삭제하시겠습니까?')" class="delete-
link">Delete</a>
    </td>
</tr>
</tbody>
</table>
</div>
<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js"></script>
<script
src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.3/dist/umd/popper.min
.js"></script>
<script
src="https://maxcdn.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js">
</script>
</body>
</html>

```

## userRegist.html

```

<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>REGIST</title>
    <link rel="stylesheet" th:href="@{/resources/style.css}" type="text/css"
/>
</head>
<body>
<div id="wrapper">
    <th:block th:insert="~{loginInfo :: userinfo}" ></th:block>

    <div class="container">

```

```

    <form th:method="POST" th:action="@{/regist/}" class="registration-
form">
    <a href="/" class="home-link">홈</a>

    <p class="name-label">이름 :</p>
    <input name="username" type="text" placeholder="이름" required
class="name-input">

    <p class="email-label">이메일 :</p>
    <input name="email" type="text" placeholder="example@example.com"
required class="email-input">
    <div th:if="${error}" class="error-message">
        <span th:text="${error}"></span>
    </div>
    <p class="pwd-label">비밀번호 :</p>
    <input name="password" type="password" placeholder="1234" required
class="pwd-input">

    <p class="birth-date-label">생년월일 :</p>
    <input name="birthDate" type="date" required class="birth-date-
input">

    <button type="submit" class="submit-button">회원 가입</button>
</form>
</div>
</div>
</body>
</html>

```

## userUpdate.html

```

<!DOCTYPE html>
<html lang="ko" xmlns:th="http://www.thymeleaf.org">
<head>
    <meta charset="UTF-8">
    <title>Title</title>
    <link rel="stylesheet" th:href="@{/resources/style.css}" type="text/css"
/>
</head>
<body>
<div id="wrapper">
    <th:block th:insert="~{loginInfo :: userinfo}" ></th:block>

    <div class="container">
        <a href="/" class="home-link">홈</a>

        <h1 class="mt-5 mb-5 update-title">User List</h1>
        <table class="table table-bordered table-striped user-table">
            <thead>

```

```
<tr>
  <th>ID</th>
  <th>Password</th>
  <th>Name</th>
</tr>
</thead>
<tbody>
<tr>
  <td th:text="${user.id}" class="user-id"></td>
  <td th:text="${user.pwd}" class="user-pwd"></td>
  <td th:text="${user.name}" class="user-name"></td>
</tr>
</tbody>
</table>

<form th:method="post" th:action="@{/usermanage/update/${user.id}}"
class="update-form">
  <p class="name-label">이름 :</p>
  <input name="name" type="text" placeholder="이름" required
class="name-input">
  <input name="id" style="display: none" th:value="${user.id}"
class="hidden-id"/>

  <p class="pwd-label">비밀번호 :</p>
  <input name="pwd" type="password" placeholder="1234" required
class="pwd-input">
  <button type="submit" class="submit-button">수정</button>
</form>
</div>
</div>
</body>
</html>
```