src/main/java/com/nhnacademy/springboard/spirngmvcboard/config/DataBaseConfig.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/config/DataBaseProperties.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/config/MybatisConfig.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/config/RootConfig.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/config/WebAppInitializer.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/config/WebConfig.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/controller/BoardController.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/controller/ControllerBase.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/controller/LoginController.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/controller/MainController.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/controller/ManagementUserController.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/controller/PostController.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/controller/RegistController.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/domain/Board.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/domain/Domain.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/domain/Gender.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/domain/Post.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/domain/User.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/init/DatabaseInitializer.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/mapper/BoardMapper.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/mapper/PostMapper.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/mapper/UserMapper.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/repository/BoardRepository.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/repository/PostRepository.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/repository/UserRepository.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/service/impl/BoardServiceyImpl.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/service/impl/PostServiceImpl.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/service/impl/UserServiceImpl.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/service/BoardService.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/service/FindPasswordRequest.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/service/LoginRequest.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/service/LoginService.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/service/PostService.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/service/UserService.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/utility/DomainUtil.java
src/main/java/com/nhnacademy/springboard/spirngmvcboard/Base.java
src/main/resources/mapper/maps/BoardMapper.xml src/main/resources/mapper/maps/PostMapper.xml
src/main/resources/mapper/maps/UserMapper.xml src/main/resources/db.properties src/main/webapp
src/main/webapp/resources src/main/webapp/resources/style.css 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

package com.nhnacademy.springboard.spirngmvcboard.config;

import javax.sql.DataSource; import lombok.RequiredArgsConstructor; import lombok.extern.slf4j.Slf4j; import org.apache.commons.dbcp2.BasicDataSource; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.jdbc.datasource.DataSourceTransactionManager; import org.springframework.transaction.PlatformTransactionManager;

@Slf4j @RequiredArgsConstructor @Configuration public class DataBaseConfig { private final DataBaseProperties databaseProperties; @Bean public DataSource dataSource() { BasicDataSource basicDataSource = new BasicDataSource(); basicDataSource.setDriverClassName(databaseProperties.getDriverClassName()); log.info("driverClassName={}", databaseProperties.getDriverClassName()); basicDataSource.setUrl(databaseProperties.getUrl()); log.info("url={}", databaseProperties.getUrl()); basicDataSource.setUsername(databaseProperties.getUsername()); log.info("username={}", databaseProperties.getUsername()); basicDataSource.setPassword(databaseProperties.getPassword()); log.info("password={}", databaseProperties.getPassword()); basicDataSource.setInitialSize(databaseProperties.getInitialSize()); log.info("initialSize={}", databaseProperties.getInitialSize()); basicDataSource.setMaxTotal(databaseProperties.getMaxTotal()); log.info("maxTotal={}", databaseProperties.getMaxTotal()); basicDataSource.setMinIdle(databaseProperties.getMinIdle()); log.info("minIdle={}", databaseProperties.getMinIdle()); basicDataSource.setMaxIdle(databaseProperties.getMaxIdle()); basicDataSource.setTestOnBorrow(databaseProperties.isTestOnBorrow()); if(databaseProperties.isTestOnBorrow()) { basicDataSource.setValidationQuery(basicDataSource.getValidationQuery()); } log.info("testOnBorrow={}", databaseProperties.isTestOnBorrow()); log.info("basicDataSource {}",basicDataSource); return basicDataSource; }

@Bean public PlatformTransactionManager transactionManager(){ return new DataSourceTransactionManager(dataSource()); }

} package com.nhnacademy.springboard.spirngmvcboard.config;

import lombok.Getter; import lombok.NonNull; import org.springframework.beans.factory.annotation.Value; import org.springframework.context.annotation.PropertySource; import org.springframework.stereotype.Component;

@Component @Getter @PropertySource(value="classpath:/db.properties") public class DataBaseProperties { @NonNull @Value("${db.username}") private String username; @NonNull @Value("${db.password}") private String password; @NonNull @Value("${db.driverClassName}") private String driverClassName; @NonNull @Value("${db.url}") private String url; @NonNull @Value("${db.initialSize}") private Integer initialSize; @NonNull @Value("${db.maxTotal}") private Integer maxTotal; @NonNull @Value("${db.maxIdle}") private Integer maxIdle; @NonNull @Value("${db.minIdle}")

private Integer minIdle; @NonNull @Value("${db.maxWaitMillis}") private Integer maxWaitMillis; @NonNull @Value("${db.validationQuery}") private String validationQuery; @NonNull @Value("${db.testOnBorrow}") private String testOnBorrow;

public boolean isTestOnBorrow() { return testOnBorrow.equals("true"); } }

package com.nhnacademy.springboard.spirngmvcboard.config;

import javax.sql.DataSource; import lombok.RequiredArgsConstructor; import org.apache.ibatis.session.SqlSessionFactory; import org.mybatis.spring.SqlSessionFactoryBean; import org.mybatis.spring.annotation.MapperScan; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.core.io.support.PathMatchingResourcePatternResolver;

@RequiredArgsConstructor @Configuration @MapperScan(basePackages = "com.nhnacademy.springboard.spirngmvcboard.mapper.**") public class MybatisConfig { private final DataSource dataSource; @Bean public SqlSessionFactory sqlSessionFactory() throws Exception { PathMatchingResourcePatternResolver resolver = new PathMatchingResourcePatternResolver(); SqlSessionFactoryBean sessionFactory = new SqlSessionFactoryBean(); sessionFactory.setDataSource(dataSource); sessionFactory.setMapperLocations(resolver.getResources("classpath*:/mapper/maps/*.xml")); org.apache.ibatis.session.Configuration configuration = new org.apache.ibatis.session.Configuration(); configuration.setMapUnderscoreToCamelCase(true);

```
//camelcase 설정
//https://mybatis.org/mybatis-3/configuration.html
sessionFactory.setConfiguration(configuration);
return sessionFactory.getObject();
```

}

}

package com.nhnacademy.springboard.spirngmvcboard.config;

import com.fasterxml.jackson.annotation.JsonInclude; import com.fasterxml.jackson.databind.ObjectMapper; import com.fasterxml.jackson.databind.SerializationFeature; import com.fasterxml.jackson.datatype.jsr310.JavaTimeModule; import com.nhnacademy.springboard.spirngmvcboard.Base; import com.nhnacademy.springboard.spirngmvcboard.domain.Board; import com.nhnacademy.springboard.spirngmvcboard.mapper.BoardMapper; import org.mybatis.spring.mapper.MapperScannerConfigurer; import org.springframework.context.MessageSource; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.ComponentScan; import org.springframework.context.annotation.Configuration; import org.springframework.context.support.ResourceBundleMessageSource; import org.springframework.stereotype.Controller;

@Configuration @ComponentScan(basePackageClasses = {Base.class}, excludeFilters = {@ComponentScan.Filter(classes={Controller.class})}) public class RootConfig {

@Bean(name = "objectMapper") public ObjectMapper objectMapper() { ObjectMapper objectMapper = new ObjectMapper(); //pretty print json objectMapper.enable(SerializationFeature.INDENT_OUTPUT); //value -> null 무시 objectMapper.setSerializationInclusion(JsonInclude.Include.NON_NULL); //LocalDate, LocalDateTime support jsr310 objectMapper.registerModule(new JavaTimeModule()); //timestamp 출력을 disable. -> 문자열형태로 출력 objectMapper.disable(SerializationFeature.WRITE_DATES_AS_TIMESTAMPS); return objectMapper; }

@Bean public MessageSource messageSource() { ResourceBundleMessageSource resourceBundleMessageSource = new ResourceBundleMessageSource(); resourceBundleMessageSource.setBasename("message"); resourceBundleMessageSource.setDefaultEncoding("UTF-8"); return resourceBundleMessageSource; }

@Bean public MapperScannerConfigurer mapperScannerConfigurer() { MapperScannerConfigurer configurer = new MapperScannerConfigurer(); configurer.setBasePackage("com.nhnacademy.springboard.spirngmvcboard.mapper"); return configurer; }

}package com.nhnacademy.springboard.spirngmvcboard.config;

import javax.servlet.Filter; import org.springframework.web.context.WebApplicationContext; import org.springframework.web.filter.CharacterEncodingFilter; import org.springframework.web.servlet.DispatcherServlet; import org.springframework.web.servlet.FrameworkServlet; import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;

public class WebAppInitializer extends AbstractAnnotationConfigDispatcherServletInitializer {

@Override protected Class<?>[] getRootConfigClasses() { return new Class[]{RootConfig.class}; }

@Override protected Class<?>[] getServletConfigClasses() { return new Class[]{WebConfig.class}; }

@Override protected String[] getServletMappings() { return new String[]{"/"}; }

@Override protected Filter[] getServletFilters() { CharacterEncodingFilter encodingFilter = new CharacterEncodingFilter(); encodingFilter.setEncoding("UTF-8"); encodingFilter.setForceEncoding(true); return new Filter[]{encodingFilter}; }

@Override protected FrameworkServlet createDispatcherServlet(WebApplicationContext servletAppContext) { DispatcherServlet dispatcherServlet = new DispatcherServlet(); dispatcherServlet.setApplicationContext(servletAppContext); return dispatcherServlet; } } package com.nhnacademy.springboard.spirngmvcboard.config;

import com.fasterxml.jackson.databind.util.Converter; import com.nhnacademy.springboard.spirngmvcboard.init.DatabaseInitializer; import com.nhnacademy.springboard.spirngmvcboard.mapper.BoardMapper; import com.nhnacademy.springboard.spirngmvcboard.mapper.PostMapper; import com.nhnacademy.springboard.spirngmvcboard.mapper.UserMapper; import com.nhnacademy.springboard.spirngmvcboard.service.UserService; import com.nhnacademy.springboard.spirngmvcboard.thymeleaf.CustomTagDialet; import java.time.LocalDate;

```
import java.time.LocalDateTime; import java.time.format.DateTimeFormatter; import
org.springframework.beans.BeansException; import org.springframework.context.ApplicationContext;
import org.springframework.context.ApplicationContextAware; import
org.springframework.context.MessageSource; import org.springframework.context.MessageSourceAware;
import org.springframework.context.annotation.Bean; import
org.springframework.context.annotation.ComponentScan; import
org.springframework.context.annotation.Configuration; import
org.springframework.format.FormatterRegistry; import org.springframework.util.StringUtils; import
org.springframework.web.servlet.config.annotation.EnableWebMvc; import
org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry; import
org.springframework.web.servlet.config.annotation.ViewResolverRegistry; import
org.springframework.web.servlet.config.annotation.WebMvcConfigurer; import
org.springframework.web.servlet.resource.PathResourceResolver; import
org.thymeleaf.spring5.SpringTemplateEngine; import
org.thymeleaf.spring5.templateresolver.SpringResourceTemplateResolver; import
org.thymeleaf.spring5.view.ThymeleafViewResolver; import org.thymeleaf.templatemode.TemplateMode;

@Configuration @EnableWebMvc @ComponentScan(basePackageClasses =
{com.nhnacademy.springboard.spirngmvcboard.controller.ControllerBase.class}) public class WebConfig
implements WebMvcConfigurer, ApplicationContextAware , MessageSourceAware { ApplicationContext
applicationContext ; MessageSource messageSource ; @Override public void
setApplicationContext(ApplicationContext applicationContext) throws BeansException {
this.applicationContext = applicationContext; }

@Override public void setMessageSource(MessageSource messageSource) { this.messageSource =
messageSource; } @Bean public ThymeleafViewResolver thymeleafViewResolver() {
ThymeleafViewResolver thymeleafViewResolver = new ThymeleafViewResolver();
thymeleafViewResolver.setTemplateEngine(springTemplateEngine());
thymeleafViewResolver.setCharacterEncoding("UTF-8"); thymeleafViewResolver.setOrder(1);
thymeleafViewResolver.setViewNames(new String[]{"*"}); return thymeleafViewResolver; } public
SpringTemplateEngine springTemplateEngine() { SpringTemplateEngine templateEngine = new
SpringTemplateEngine(); templateEngine.setTemplateResolver(springResourceTemplateResolver());
templateEngine.setTemplateEngineMessageSource(messageSource); templateEngine.addDialect(new
CustomTagDialet()); return templateEngine;

}

@Bean public DatabaseInitializer databaseInitializer(UserService userService, BoardMapper boardMapper,
PostMapper postMapper, UserMapper userMapper) { return new DatabaseInitializer(userService,
boardMapper, postMapper, userMapper); } @Override public void addFormatters(FormatterRegistry
registry) { registry.addConverter(String.class, LocalDate.class, source -> { if (source == null ||
source.isEmpty()) { return null; } return LocalDate.parse(source, DateTimeFormatter.ofPattern("yyyy-MM-
dd")); }); }

public SpringResourceTemplateResolver springResourceTemplateResolver() {
SpringResourceTemplateResolver templateResolver = new SpringResourceTemplateResolver();
templateResolver.setApplicationContext(applicationContext);
templateResolver.setCharacterEncoding("UTF-8"); templateResolver.setPrefix("/WEB-INF/views/");
```

templateResolver.setSuffix(".html"); templateResolver.setCacheable(false); templateResolver.setTemplateMode(TemplateMode.HTML); return templateResolver; }

@Override public void addResourceHandlers(ResourceHandlerRegistry registry) { registry.addResourceHandler("/resources/**") .addResourceLocations("/resources/") .setCachePeriod(3600) .resourceChain(true) .addResolver(new PathResourceResolver()); }

@Override public void configureViewResolvers(ViewResolverRegistry registry) { registry.viewResolver(thymeleafViewResolver()); } } package com.nhnacademy.springboard.spirngmvcboard.controller;

import com.nhnacademy.springboard.spirngmvcboard.domain.Board; import com.nhnacademy.springboard.spirngmvcboard.domain.Post; import com.nhnacademy.springboard.spirngmvcboard.domain.User; import com.nhnacademy.springboard.spirngmvcboard.service.BoardService; import com.nhnacademy.springboard.spirngmvcboard.service.PostService; import java.nio.file.AccessDeniedException; import java.util.List; import javax.servlet.http.HttpSession; import lombok.extern.slf4j.Slf4j; import org.springframework.stereotype.Controller; import org.springframework.ui.Model; import org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.PostMapping; import org.springframework.web.bind.annotation.RequestMapping;

@Slf4j @Controller @RequestMapping("/board") public class BoardController implements ControllerBase { private final BoardService boardService; private final PostService postService;

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

@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") int id) { log.info("id={}", id); Board board = boardService.findById((long) id); List posts = postService.findByBoardId(board.getId()); 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) { // 수정 User loginUser = (User) session.getAttribute("user"); post.setAuthor(loginUser); post.setBoard(boardService.findById(boardId)); // 수정 postService.save(post); return "redirect:/board/" + boardId; // 수정 }

@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", post); model.addAttribute("boardId", boardId); // 삭제 model.addAttribute("board", boardService.findById(boardId)); // 추가 return "board/write"; } } package com.nhnacademy.springboard.spirngmvcboard.controller;

public interface ControllerBase {

} package com.nhnacademy.springboard.spirngmvcboard.controller;

import com.nhnacademy.springboard.spirngmvcboard.domain.User; import com.nhnacademy.springboard.spirngmvcboard.service.FindPasswordRequest; import com.nhnacademy.springboard.spirngmvcboard.service.LoginRequest; import com.nhnacademy.springboard.spirngmvcboard.service.LoginService; import com.nhnacademy.springboard.spirngmvcboard.service.UserService; import com.nhnacademy.springboard.spirngmvcboard.service.LoginRequest.FindIdRequest; import java.util.Objects; import javax.servlet.http.Cookie; import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpServletResponse; import javax.servlet.http.HttpSession; import lombok.RequiredArgsConstructor; import lombok.extern.slf4j.Slf4j; import org.springframework.stereotype.Controller; import org.springframework.ui.Model; import org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PostMapping; import org.springframework.web.bind.annotation.RequestMapping;

@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.findByUseremail(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.findByUseremail(findPasswordRequest.getEmail()); if (user == null) {

model.addAttribute("errorMessage", "해당 이메일로 가입한 사용자가 존재하지 않습니다."); 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.findByUseremail(findIdRequest.getEmail()); if (user == null) { model.addAttribute("errorMessage", "해당 이메일로 가입한 사용자가 존재하지 않습니다."); return "user/find-id"; } model.addAttribute("message", "해당 이메일로 가입한 사용자의 아이디는 " + user.getUsername() + " 입니다."); return "user/find-id-result"; } } package com.nhnacademy.springboard.spirngmvcboard.controller;

import com.nhnacademy.springboard.spirngmvcboard.domain.User; import javax.servlet.http.HttpServletRequest; import javax.servlet.http.HttpSession; import org.springframework.stereotype.Controller; import org.springframework.ui.Model; import org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.RequestMapping;

@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"; } } package com.nhnacademy.springboard.spirngmvcboard.controller;

import com.nhnacademy.springboard.spirngmvcboard.domain.User; import com.nhnacademy.springboard.spirngmvcboard.service.UserService; import java.util.List; import lombok.extern.slf4j.Slf4j; import org.springframework.stereotype.Controller; import org.springframework.ui.Model; import org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.PostMapping; import org.springframework.web.bind.annotation.RequestMapping;

@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 userList = userService.findAll(); model.addAttribute("userlist", userList); return "managementUser"; }

@GetMapping("/delete/{id}") public String doDeleteUser(@PathVariable("id") Long id, Model model) { userService.delete(id); model.addAttribute("userList", userService.findAll()); return "redirect:/usermanage/list"; }

@GetMapping("/update/{id}") public String validateUpdateUser(@PathVariable("id") Long id, Model model) { User user = userService.findById(id); if (user == null) { model.addAttribute("error", "해당 사용자가 없습니

다."); return "managementUser"; } model.addAttribute("user", user); return "userUpdate"; }

@PostMapping("/update/{id}") public String doUpdate(@ModelAttribute("user") User user, @PathVariable("id") Long id, Model model) { User existingUser = userService.findById(id); if (existingUser == null) { model.addAttribute("error", "해당 사용자가 없습니다."); return "managementUser"; } existingUser.setUsername(user.getUsername()); existingUser.setPassword(user.getPassword()); existingUser.setEmail(user.getEmail()); existingUser.setBirthDate(user.getBirthDate()); userService.update(existingUser); model.addAttribute("userList", userService.findAll()); return "userRegist"; } } package com.nhnacademy.springboard.spirngmvcboard.controller;

import com.nhnacademy.springboard.spirngmvcboard.domain.Board; import com.nhnacademy.springboard.spirngmvcboard.domain.Post; import com.nhnacademy.springboard.spirngmvcboard.domain.User; import com.nhnacademy.springboard.spirngmvcboard.service.BoardService; import com.nhnacademy.springboard.spirngmvcboard.service.PostService; import com.nhnacademy.springboard.spirngmvcboard.service.UserService; import java.nio.file.AccessDeniedException; import javax.servlet.http.HttpSession; import org.apache.ibatis.javassist.NotFoundException; import org.springframework.stereotype.Controller; import org.springframework.ui.Model; import org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PathVariable; import org.springframework.web.bind.annotation.PostMapping; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestParam;

@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.findById(postId);
  User author = userService.findById(post.getAuthor().getId());
  post.setAuthor(author);

  if (post == null) {

    try {
      throw new NotFoundException("Post not found with id " + postId);
    } catch (NotFoundException e) {
```

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

```
  // 로그인한 유저와 게시물 작성자가 같은 경우에만 수정 가능
  if (user != null && post.getAuthor().getId().equals(user.getId())) {
    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.findById(postId);

```
  // 로그인한 유저와 게시물 작성자가 같은 경우에만 수정 가능
  if (loginUser != null &&
  post.getAuthor().getId().equals(loginUser.getId())) {
    post.setTitle(updatedPost.getTitle());
    post.setContent(updatedPost.getContent());
    postService.save(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.findById(postId);

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

}

}

package com.nhnacademy.springboard.spirngmvcboard.controller;

import com.nhnacademy.springboard.spirngmvcboard.domain.User; import com.nhnacademy.springboard.spirngmvcboard.service.UserService; import lombok.extern.slf4j.Slf4j; import org.springframework.format.annotation.DateTimeFormat; import org.springframework.stereotype.Controller; import org.springframework.ui.Model; import org.springframework.web.bind.annotation.GetMapping; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.PostMapping; import org.springframework.web.bind.annotation.RequestMapping; @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.findByUseremail(user.getEmail());

```
  if (existingUser != null) {
    model.addAttribute("error", "이미 존재하는 사용자 이메일입니다. 다른 사용자 이메일을 선
  택해주세요.");
    log.info("User already exists: {}", user.getUsername());
    return "userRegist";
  }

  try {
    userService.save(user);
  } catch (Exception e) {
    model.addAttribute("error", "등록 중 오류가 발생했습니다. 다시 시도해주세요.");
    return "userRegist";
  }

  return "redirect:/";
```

}

@GetMapping("/regist") public String doRegist() { return "userRegist"; } } package com.nhnacademy.springboard.spirngmvcboard.domain;

import lombok.Getter; import lombok.Setter; import lombok.ToString;

@Getter @Setter @ToString public class Board { private Long id; private String name; private String description;

```
  public Board() {
  }


  public Board(String name, String description) {
    this.name = name;
    this.description = description;
  }
```

}

package com.nhnacademy.springboard.spirngmvcboard.domain;

import com.nhnacademy.springboard.spirngmvcboard.utility.DomainUtil; import java.time.LocalDate; import java.time.LocalDateTime; import java.util.Objects; import lombok.Getter; import lombok.Setter; import lombok.ToString; // 도메인 클래스 @Getter @Setter @ToString public class Post { private Long id; private String title; private String content; private LocalDateTime createdAt; private LocalDateTime updatedAt; private Long authorId; private Long boardId; private User author; private Board board; public Post(){} public Post(String title, String content, User author, Board board) { this.title = title; this.content = content; this.createdAt = LocalDateTime.now(); this.updatedAt = LocalDateTime.now(); this.author = author; this.board = board; } public Post(String title, String content, User author, Board board, LocalDateTime

createdAt, LocalDateTime updatedAt) { this.title = title; this.content = content; this.createdAt =createdAt; this.updatedAt = updatedAt; this.author = author; this.board = board; } public void setAuthor(User author) { this.author = author; this.authorId = author.getId(); }

public void setBoard(Board board) { this.board = board; this.boardId = board.getId(); } public Board getBoard() { return board; }

}

//package com.nhnacademy.springboard.spirngmvcboard.domain; // // import com.nhnacademy.springboard.spirngmvcboard.utility.DomainUtil; // import java.time.LocalDate; // import java.time.LocalDateTime; // import java.util.Objects; // import lombok.Getter; // import lombok.Setter; // import lombok.ToString; //// 도메인 클래스 //@Getter //@Setter //@ToString //public class Post { // private Long id; // private String title; // private String content; // private LocalDateTime createdAt; // private LocalDateTime updatedAt; // private Long authorId; // private Long boardId; // // public Post(){} // public Post(String title, String content, Long authorId, Long boardId) { // this.title = title; // this.content = content; // this.createdAt = LocalDateTime.now(); // this.updatedAt = LocalDateTime.now(); // this.authorId = authorId; // this.boardId = boardId; // } // public Post(String title, String content, Long authorId, Long boardId, LocalDateTime createdAt, LocalDateTime updatedAt) { // this.title = title; // this.content = content; // this.createdAt =createdAt; // this.updatedAt = updatedAt; // this.authorId = authorId; // this.boardId = boardId; // } // // //}

package com.nhnacademy.springboard.spirngmvcboard.domain;

import java.time.LocalDate; import java.time.LocalDateTime; import lombok.Getter; import lombok.Setter; import lombok.ToString; import org.springframework.format.annotation.DateTimeFormat;

@ToString @Getter @Setter public class User { private Long id; private String username; private String password; private String email;

@DateTimeFormat(pattern = "yyyy-MM-dd") private LocalDate birthDate; private LocalDateTime createdAt;

public User() { }

public User(String username, String password, String email, LocalDate birthDate) { this.username = username; this.password = password; this.email = email; this.birthDate = birthDate; }

} package com.nhnacademy.springboard.spirngmvcboard.init;

import com.nhnacademy.springboard.spirngmvcboard.domain.Board; import com.nhnacademy.springboard.spirngmvcboard.domain.Post; import com.nhnacademy.springboard.spirngmvcboard.domain.User; import com.nhnacademy.springboard.spirngmvcboard.mapper.BoardMapper; import com.nhnacademy.springboard.spirngmvcboard.mapper.PostMapper; import com.nhnacademy.springboard.spirngmvcboard.mapper.UserMapper; import com.nhnacademy.springboard.spirngmvcboard.service.UserService; import java.time.LocalDate; import java.time.LocalDateTime; import java.util.ArrayList; import java.util.List; import lombok.extern.slf4j.Slf4j; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.context.annotation.Bean; import org.springframework.stereotype.Component;

@Slf4j @Component public class DatabaseInitializer {

private final UserService userService; private final BoardMapper boardMapper; private final PostMapper postMapper; private final UserMapper userMapper;

@Autowired public DatabaseInitializer(UserService userService, BoardMapper boardMapper, PostMapper postMapper, UserMapper userMapper) { this.userService = userService; this.boardMapper = boardMapper; this.postMapper = postMapper; this.userMapper = userMapper; }

@Bean public void initializeDatabase() { Board board1 = new Board("공지사항", "웹 사이트 관련 공지사항"); if (boardMapper.findById(1L) == null) { boardMapper.save(board1); } board1.setId(boardMapper.findByName("공지사항").getId());

```
log.info("board1 id: {}", board1.getId());

Board board2 = new Board("자유게시판", "자유롭게 글을 올리는 게시판입니다.");
if (boardMapper.findById(2L) == null) {
  boardMapper.save(board2);
}
board2.setId(boardMapper.findByName("자유게시판").getId());

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

User user = new User("admin", "1234", "admin@nhnacademy.com",
LocalDate.now());
if (userMapper.findByUseremail(user.getEmail()) == null) {
  userMapper.save(user);
} else {
  user.setId(userMapper.findByUseremail(user.getEmail()).getId());
}

Post post1 = new Post("서비스 이용 안내", "안녕하세요. NHN Academy 웹 사이트를 이용해
주시는 여러분께 감사드립니다.", user, board1);
post1.setId(1L);
post1.setCreatedAt(LocalDateTime.now().minusDays(7));
post1.setUpdatedAt(LocalDateTime.now().minusDays(7));

Post post2 = new Post("나만의 공부법", "여러분은 자신만의 공부법이 있으신가요? 제가 추천하
는 공부법을 소개합니다.", user, board2);
post2.setId(2L);
post2.setCreatedAt(LocalDateTime.now().minusDays(5));
post2.setUpdatedAt(LocalDateTime.now().minusDays(5));

Post post3 = new Post("자유롭게 글을 올려보세요", "이 게시판은 자유롭게 글을 올리는 공간입
니다.", user, board2);
post3.setId(3L);
post3.setCreatedAt(LocalDateTime.now().minusDays(3));
post3.setUpdatedAt(LocalDateTime.now().minusDays(3));

Post post4 = new Post("NHN IT 교육 서비스 이용 후기", "NHN에서 제공하는 IT 교육 서비스
를 이용하셨나요? 후기를 남겨보세요!", user, board2);
post4.setId(4L);
```

```
  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) { // 게시물의 ID를 이용해 이미 존재하는지 확인 Post existingPost = postMapper.findById(post.getId()); if (existingPost == null) { // 게시물이 존재하지 않으면 저장 postMapper.save(post); } } }package com.nhnacademy.springboard.spirngmvcboard.mapper;

import com.nhnacademy.springboard.spirngmvcboard.domain.Board; import org.apache.ibatis.annotations.Mapper; import java.util.List;

@Mapper public interface BoardMapper { void save(Board board);

Board findById(Long id);

Board findByName(String title);

void update(Board board);

void delete(Long id);

List findAll(); } package com.nhnacademy.springboard.spirngmvcboard.mapper;

import com.nhnacademy.springboard.spirngmvcboard.domain.Board; import com.nhnacademy.springboard.spirngmvcboard.domain.Post; import org.apache.ibatis.annotations.Delete; import org.apache.ibatis.annotations.Insert; import org.apache.ibatis.annotations.Mapper; import org.apache.ibatis.annotations.Select; import org.apache.ibatis.annotations.Update; import java.util.List;

@Mapper public interface PostMapper { void save(Post post);

Post findById(Long id);

void update(Post post);

void delete(Long id);

List findAll(); List findByBoard(Board board); List findByBoardId(Long boardId); List findByAuthorId(Long userId);

} package com.nhnacademy.springboard.spirngmvcboard.mapper;

import com.nhnacademy.springboard.spirngmvcboard.domain.User; import org.apache.ibatis.annotations.Delete; import org.apache.ibatis.annotations.Insert; import org.apache.ibatis.annotations.Mapper; import org.apache.ibatis.annotations.Select; import org.apache.ibatis.annotations.Update; import java.util.List; import org.springframework.stereotype.Repository;

@Mapper public interface UserMapper { void save(User user); User findById(Long id);

User findByUsername(String username); User findByUseremail(String email);

void update(User user);

void delete(Long id);

List findAll(); } package com.nhnacademy.springboard.spirngmvcboard.service.impl;

import com.nhnacademy.springboard.spirngmvcboard.domain.Board; import com.nhnacademy.springboard.spirngmvcboard.mapper.BoardMapper; import com.nhnacademy.springboard.spirngmvcboard.service.BoardService; import java.util.List; import lombok.RequiredArgsConstructor; import org.springframework.stereotype.Service;

@Service @RequiredArgsConstructor public class BoardServiceImpl implements BoardService { private final BoardMapper boardMapper; @Override public void save(Board board) { boardMapper.save(board); } @Override public Board findById(Long id) { return boardMapper.findById(id); }

@Override public void update(Board board) { boardMapper.update(board); }

@Override public Board findByName(String title) { return boardMapper.findByName(title); }

@Override public void delete(Long id) { boardMapper.delete(id); }

@Override public List findAll() { return boardMapper.findAll(); }

} package com.nhnacademy.springboard.spirngmvcboard.service.impl;

import com.nhnacademy.springboard.spirngmvcboard.domain.Board; import com.nhnacademy.springboard.spirngmvcboard.domain.Post; import com.nhnacademy.springboard.spirngmvcboard.mapper.PostMapper; import com.nhnacademy.springboard.spirngmvcboard.service.PostService; import java.util.List; import lombok.RequiredArgsConstructor; import org.springframework.stereotype.Service; import org.springframework.transaction.annotation.Transactional;

@RequiredArgsConstructor @Transactional @Service public class PostServiceImpl implements PostService { private final PostMapper postMapper;

@Override public void save(Post post) { postMapper.save(post); }

@Override public Post findById(Long id) { return postMapper.findById(id); }

@Override public List findByAuthorId(Long userId) { return postMapper.findByAuthorId(userId); }

@Override public void update(Post post) { postMapper.update(post); }

@Override public void delete(Long id) { postMapper.delete(id); }

@Override public List findAll() { return postMapper.findAll(); }

@Override public List findByBoard(Board board) { return postMapper.findByBoardId(board.getId()); }

@Override public List findByBoardId(Long boardId) { return postMapper.findByBoardId(boardId); }

} package com.nhnacademy.springboard.spirngmvcboard.service.impl;

import com.nhnacademy.springboard.spirngmvcboard.domain.User; import com.nhnacademy.springboard.spirngmvcboard.mapper.UserMapper; import com.nhnacademy.springboard.spirngmvcboard.repository.UserRepository; import com.nhnacademy.springboard.spirngmvcboard.service.UserService; import java.util.List; import lombok.RequiredArgsConstructor; import org.springframework.stereotype.Service;

@Service @RequiredArgsConstructor public class UserServiceImpl implements UserService { private final UserMapper userMapper; @Override public void save(User user) { userMapper.save(user); }

@Override public User findById(Long id) { return userMapper.findById(id); }

@Override public User findByUsername(String username) { return userMapper.findByUsername(username); }

@Override public User findByUseremail(String email) { return userMapper.findByUseremail(email); }

@Override public void update(User user) { userMapper.update(user); }

@Override public void delete(Long id) { userMapper.delete(id); }

@Override public List findAll() { return userMapper.findAll(); }

} package com.nhnacademy.springboard.spirngmvcboard.service;

import com.nhnacademy.springboard.spirngmvcboard.domain.Board; import java.util.List;

public interface BoardService { void save(Board board); Board findByName(String title);

```
Board findById(Long id);
void update(Board board);
void delete(Long id);
List<Board> findAll();
```

} package com.nhnacademy.springboard.spirngmvcboard.service;

import lombok.Getter; import lombok.Setter;

@Getter @Setter public class FindPasswordRequest { private String email; private String passwword; } package com.nhnacademy.springboard.spirngmvcboard.service;

import javax.validation.constraints.NotBlank; import lombok.Getter; import lombok.Setter;

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

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

```
// 생성자, getter, setter
```

} } package com.nhnacademy.springboard.spirngmvcboard.service;

import com.nhnacademy.springboard.spirngmvcboard.domain.User; import com.nhnacademy.springboard.spirngmvcboard.mapper.UserMapper; import lombok.RequiredArgsConstructor; import lombok.extern.slf4j.Slf4j; import org.springframework.stereotype.Service; @Slf4j @Service @RequiredArgsConstructor public class LoginService { private final UserMapper userMapper;

public boolean isValidate(LoginRequest loginRequest) { User user = userMapper.findByUseremail(loginRequest.getEmail()); boolean result = user != null && user.getPassword().equals(loginRequest.getPwd()); log.info("isValidate : {}", result); return result; } } package com.nhnacademy.springboard.spirngmvcboard.service;

import com.nhnacademy.springboard.spirngmvcboard.domain.Board; import com.nhnacademy.springboard.spirngmvcboard.domain.Post; import com.nhnacademy.springboard.spirngmvcboard.domain.User; import java.util.List;

public interface PostService { void save(Post post);

Post findById(Long id); void update(Post post); void delete(Long id); List findByAuthorId(Long userId); List findAll(); List findByBoard(Board board); List findByBoardId(Long boardId); } package com.nhnacademy.springboard.spirngmvcboard.service;

import com.nhnacademy.springboard.spirngmvcboard.domain.User; import java.util.List;

public interface UserService { void save(User user); List findAll(); User findById(Long id); User findByUsername(String username); User findByUseremail(String email); void update(User user); void delete(Long id); } package com.nhnacademy.springboard.spirngmvcboard.thymeleaf;

import java.util.Collections; import java.util.Set; import org.thymeleaf.context.IExpressionContext; import org.thymeleaf.dialect.AbstractDialect; import org.thymeleaf.dialect.IExpressionObjectDialect; import org.thymeleaf.expression.IExpressionObjectFactory; import org.thymeleaf.spring5.expression.SpringStandardExpressionObjectFactory;

//todo#15 CustomTagDialet 생성 , 커스텀 함수를 사용하기위해서 IDialect 확장한 IExpressionObjectDialect 구현. public class CustomTagDialet extends AbstractDialect implements IExpressionObjectDialect {

```
public CustomTagDialet() {
    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 뒀떤 TagUtils 객체;
             return new TagUtils();
        }

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

} package com.nhnacademy.springboard.spirngmvcboard.thymeleaf;

import com.nhnacademy.springboard.spirngmvcboard.domain.Gender;

//todo#14 thymeleaf에서 사용할 custom 함수, M or F 받아서 남성 or 여성을 리턴하는 함수 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 ""; } } } package com.nhnacademy.springboard.spirngmvcboard.utility;

import java.security.CryptoPrimitive; import java.util.UUID;

public class DomainUtil { public static String createUUID(){ String uuid= UUID.randomUUID().toString(); return uuid; } } package com.nhnacademy.springboard.spirngmvcboard; public interface Base {

}

INSERT INTO board(name, description) VALUES (#{name}, #{description})

```
<update id="update"
parameterType="com.nhnacademy.springboard.spirngmvcboard.domain.Board">
  UPDATE board
  SET name = #{name}, description = #{description}
  WHERE id = #{id}
</update>

<delete id="delete" parameterType="long">
  DELETE FROM board WHERE id = #{id}
</delete>

<select id="findById" parameterType="long"
  resultType="com.nhnacademy.springboard.spirngmvcboard.domain.Board">
  SELECT * FROM board WHERE id = #{id}
</select>

<select id="findAll"
```

```
    resultType="com.nhnacademy.springboard.spirngmvcboard.domain.Board">
    SELECT * FROM board
  </select>
```

SELECT * FROM post WHERE id=#{id} UPDATE post SET title=#{title}, content=#{content}, updated_at=NOW(), author_id=#{author.id}, board_id=#{board.id} WHERE id=#{id} DELETE FROM post WHERE id=#{id} INSERT INTO User SET username = #{username}, password = #{password}, email = #{email}, birth_date = #{birthDate}, created_at = NOW() UPDATE User SET username = #{username}, password = #{password}, email = #{email}, birth_date = #{birthDate} WHERE id = #{id} DELETE FROM User WHERE id = #{id} /* 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; }

게시물 수정

제목 : 제목을 입력하세요.

```
<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>
```

Error message goes here

게시물 상세보기

게시물 ID

게시판 ID

제목

작성자

내용

작성일시

삭제

홈으로

게시물 목록

```
<!-- 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>
```

글쓰기
홈으로

```
<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>
```

Error message goes here