**GitHub Repository 링크**

[**https://github.com/rlathwls03/helloSpringDataJpa**](https://github.com/rlathwls03/helloSpringDataJpa)

**주요 소스 및 설명 (함수 설명은 주석으로 첨부)**

|  |  |
| --- | --- |
| WebConfig.java  package kr.ac.hansung.cse.hellospringdatajpa.config;  import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.web.servlet.config.annotation.ViewControllerRegistry; import org.springframework.web.servlet.config.annotation.WebMvcConfigurer; import org.thymeleaf.extras.springsecurity6.dialect.SpringSecurityDialect;  @Configuration public class WebConfig implements WebMvcConfigurer {  @Override  public void addViewControllers(ViewControllerRegistry registry)  {  registry.addViewController("/").setViewName("home");  registry.addViewController("/login").setViewName("login");  registry.addViewController("/home").setViewName("home");  registry.addViewController("/admin/home").setViewName("adminhome");  registry.addViewController("/accessDenied").setViewName("403");  }   @Bean  public SpringSecurityDialect securityDialect() {  return new SpringSecurityDialect();  } } | * 뷰 컨트롤러 등록 * Thymeleaf에서 Spring Security 태그 사용 활성화 |
| WebSecurityConfig.java  package kr.ac.hansung.cse.hellospringdatajpa.config;  import org.springframework.beans.factory.annotation.Autowired; import org.springframework.context.annotation.Bean; import org.springframework.context.annotation.Configuration; import org.springframework.security.authentication.AuthenticationManager; import org.springframework.security.config.annotation.authentication.configuration.AuthenticationConfiguration; import org.springframework.security.config.annotation.web.builders.HttpSecurity; import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity; import org.springframework.security.core.userdetails.UserDetailsService; import org.springframework.security.crypto.bcrypt.BCryptPasswordEncoder; import org.springframework.security.crypto.password.PasswordEncoder; import org.springframework.security.web.SecurityFilterChain;  @Configuration public class WebSecurityConfig {   @Autowired  private UserDetailsService customUserDetailsService;   @Bean  public PasswordEncoder passwordEncoder() {  return new BCryptPasswordEncoder();  }   @Bean  public AuthenticationManager authenticationManager(AuthenticationConfiguration config) throws Exception {  return config.getAuthenticationManager();  }   private static final String[] PUBLIC\_MATCHERS = {  "/webjars/\*\*",  "/css/\*\*",  "/js/\*\*",  "/images/\*\*",  "/about/\*\*",  "/contact/\*\*",  "/error/\*\*",  "/console/\*\*"  };   @Bean  public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  http  // URL 접근 권한 설정(인가)  .authorizeHttpRequests(authz -> authz  .requestMatchers(PUBLIC\_MATCHERS).permitAll()  .requestMatchers("/", "/home", "/signup", "/signup-success").permitAll()  // 로그인 페이지는 모두 접근 허용  .requestMatchers("/login").permitAll()  .requestMatchers("/admin/\*\*").hasRole("ADMIN")  // 상품 목록 조회는 ROLE\_USER 이상 (ROLE\_USER, ROLE\_ADMIN)  .requestMatchers("/products").hasAnyRole("USER", "ADMIN")  // 나머지 요청은 인증된 사용자만 허용  .anyRequest().authenticated()  )  // 로그인 설정  .formLogin(formLogin -> formLogin  .loginPage("/login")  .defaultSuccessUrl("/products", true)  .failureUrl("/login?error")  .usernameParameter("username") // login.html에서 form 필드 name="username"  .passwordParameter("password") // login.html에서 form 필드 name="password"  .permitAll()  )  // 로그아웃 설정  .logout(logout -> logout  .logoutUrl("/logout")  .logoutSuccessUrl("/login?logout")  .permitAll()  )  .exceptionHandling(exceptions -> exceptions  .accessDeniedPage("/accessDenied")  )  .userDetailsService(customUserDetailsService)  // CSRF 비활성화: GET /logout 을 허용하도록 설정  .csrf(csrf -> csrf.disable());  /\* .csrf(csrf -> csrf  .ignoringRequestMatchers("/api/\*\*"));\*/    return http.build();  } } | * Spring Security 필터 체인 구성 * 인증, 인가 규칙 정의 * 로그인 로그아웃 설정 * 사용자 정보 조회 방법 등록 * 비밀번호 암호화 방법 설정 * CSRF 토큰 설정 |
| AdminController.java  package kr.ac.hansung.cse.hellospringdatajpa.controller;  import kr.ac.hansung.cse.hellospringdatajpa.entity.MyUser; import kr.ac.hansung.cse.hellospringdatajpa.repository.UserRepository; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.security.access.prepost.PreAuthorize; import org.springframework.stereotype.Controller; import org.springframework.ui.Model; import org.springframework.web.bind.annotation.\*;  import java.util.List;  @Controller @RequestMapping("/admin") public class AdminController {   @Autowired  private UserRepository userRepository;   // 전체 사용자 목록 조회 (ROLE\_ADMIN만 접근)  @GetMapping("/users")  // Method Security를 사용할 경우 @PreAuthorize("hasRole('ADMIN')") 추가  public String listAllUsers(Model model) {  List<MyUser> allUsers = userRepository.findAll();  model.addAttribute("users", allUsers);  return "admin\_users"; // /templates/admin\_users.html  } } | * 관리자용 기능이 모여있는 컨트롤러 |
| ProductController.java  package kr.ac.hansung.cse.hellospringdatajpa.controller;  import kr.ac.hansung.cse.hellospringdatajpa.entity.Product; import kr.ac.hansung.cse.hellospringdatajpa.service.ProductService; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Controller; import org.springframework.ui.Model; import org.springframework.web.bind.annotation.\*; import jakarta.validation.Valid; import org.springframework.validation.BindingResult;  import java.util.List;  @Controller @RequestMapping("/products")  public class ProductController {   @Autowired  private ProductService service;   // 상품 목록 조회 (ROLE\_USER, ROLE\_ADMIN)  @GetMapping({"", "/"}) // products 또는 products/ 둘 다 매핑  public String viewHomePage(Model model) {   List<Product> listProducts = service.listAll();  model.addAttribute("listProducts", listProducts);   return "index";  }   // 상품 생성 폼 (ROLE\_ADMIN만 템플릿 단에서 노출됨)  @GetMapping("/new")  public String showNewProductPage(Model model) {   Product product = new Product();  model.addAttribute("product", product);   return "new\_product";  }   // 상품 수정 폼 (ROLE\_ADMIN만 템플릿 단에서 노출됨)  @GetMapping("/edit/{id}")  public String showEditProductPage(@PathVariable(name = "id") Long id, Model model) {   Product product = service.get(id);  model.addAttribute("product", product);   return "edit\_product";  }   // @ModelAttribute는 Form data (예: name=Laptop&brand=Samsung&madeIn=Korea&price=1000.00)를 Product 객체  // @RequestBody는 HTTP 요청 본문에 포함된  // JSON 데이터(예: {"name": "Laptop", "brand": "Samsung", "madeIn": "Korea", "price": 1000.00})를 Product 객체에 매핑  @PostMapping("/save")  public String saveProduct(  @Valid @ModelAttribute("product") Product product,  BindingResult bindingResult,  Model model) {  // 1) 유효성 검사 오류가 있으면, 다시 폼으로 돌아가서 에러 메시지 출력  if (bindingResult.hasErrors()) {  // 새로 작성인지 수정인지에 따라 뷰를 구분  if (product.getId() == null) {  // new\_product.html  return "new\_product";  } else {  // edit\_product.html  return "edit\_product";  }  }  // 2) 오류 없으면 저장  service.save(product);  return "redirect:/products";  }   // 상품 삭제 (ROLE\_ADMIN만 템플릿 단에서 노출됨)  @GetMapping("/delete/{id}")  public String deleteProduct(@PathVariable(name = "id") Long id) {   service.delete(id);  return "redirect:/products";  } } | * 상품 관련 CRUD 기능을 담당 * /products/\*\* 경로로 들어오는 요청을 처리 |
| RegistraionController.java  package kr.ac.hansung.cse.hellospringdatajpa.controller;  import kr.ac.hansung.cse.hellospringdatajpa.entity.MyRole; import kr.ac.hansung.cse.hellospringdatajpa.entity.MyUser; import kr.ac.hansung.cse.hellospringdatajpa.service.RegistrationService; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Controller; import org.springframework.ui.Model; import org.springframework.web.bind.annotation.ModelAttribute; import org.springframework.web.bind.annotation.RequestMapping; import org.springframework.web.bind.annotation.RequestMethod;  import java.util.ArrayList; import java.util.List;  @Controller public class RegistrationController {   @Autowired  private RegistrationService registrationService;   // GET /signup: 회원가입 폼 보여주기  @RequestMapping(value = "/signup", method = RequestMethod.GET)  public String signup(Model model) {   MyUser user = new MyUser();  model.addAttribute("user", user);   return "signup";  }   // POST /signup: 실제 회원가입 처리  @RequestMapping(value = "/signup", method = RequestMethod.POST)  public String signupPost(@ModelAttribute("user") MyUser user, Model model) {   if (registrationService.checkEmailExists(user.getEmail())) {  model.addAttribute("emailExists", true);  return "signup";  }  else {  List<MyRole> userRoles = new ArrayList<>();   MyRole role = registrationService.findByRolename("ROLE\_USER");  userRoles.add(role);   // 특정 이메일 주소인 경우 ADMIN 역할 추가  if ("admin@hansung.ac.kr".equals(user.getEmail())) {  MyRole roleAdmin = registrationService.findByRolename("ROLE\_ADMIN");  userRoles.add(roleAdmin);  }   registrationService.createUser(user, userRoles);   return "redirect:/";  }  } } | * 사용자 회원가입 기능 담당 * 로그인하지 않은 상태에서도 /signup으로 접근 가능 |
| MyRole.java  package kr.ac.hansung.cse.hellospringdatajpa.entity;  import jakarta.persistence.\*; import lombok.Getter; import lombok.NoArgsConstructor; import lombok.Setter;  import java.util.List;  @Entity @Table(name="roles") @Getter @Setter @NoArgsConstructor public class MyRole {  @Id  @GeneratedValue(strategy= GenerationType.IDENTITY)  private Integer id;   @Column(nullable=false, unique=true)  private String rolename;   // mappedBy: User.roles 필드가 이 관계를 관리  @ManyToMany(mappedBy="roles")  private List<MyUser> users;   public MyRole(String rolename) {  this.rolename = rolename;  } } |  |
| MyUser.java  package kr.ac.hansung.cse.hellospringdatajpa.entity;  import jakarta.persistence.\*; import lombok.Getter; import lombok.NoArgsConstructor; import lombok.Setter;  import java.util.List;  @Entity @Table(name="users") @Getter @Setter @NoArgsConstructor public class MyUser {  @Id  @GeneratedValue(strategy= GenerationType.IDENTITY)  private Integer id;   @Column(nullable=false)  private String password;   @Column(nullable=false, unique=true)  private String email;   @ManyToMany(cascade=CascadeType.MERGE)  @JoinTable(  name="user\_role",  joinColumns={@JoinColumn(name="USER\_ID", referencedColumnName="ID")},  inverseJoinColumns={@JoinColumn(name="ROLE\_ID", referencedColumnName="ID")})  private List<MyRole> roles; } |  |
| Product.java  package kr.ac.hansung.cse.hellospringdatajpa.entity;  import jakarta.persistence.\*; import jakarta.validation.constraints.\*; import lombok.Getter; import lombok.NoArgsConstructor; import lombok.Setter; import lombok.ToString;  import java.math.BigDecimal;  @Getter @Setter @ToString @NoArgsConstructor @Entity @Table(name = "product") public class Product {  @Id  @GeneratedValue(strategy = GenerationType.IDENTITY)  private Long id;   @NotBlank(message = "Product name은 필수 입력입니다.")  private String name;   @NotBlank(message = "Brand는 필수 입력입니다.")  private String brand;   @NotBlank(message = "MadeIn은 필수 입력입니다.")  private String madeIn;   // 1) BigDecimal 타입으로 변경  // 2) precision=10, scale=2 로 컬럼을 DECIMAL(10,2) 형태로 생성  @NotNull(message = "Price는 필수 입력입니다.")  @Positive(message = "Price는 0보다 커야 합니다.")  @Digits(integer = 10, fraction = 2, message = "Price는 소수점 둘째자리까지 허용합니다.")  @Column(precision = 10, scale = 2)  private BigDecimal price;   public Product(String name, String brand, String madeIn, BigDecimal price) {  this.name = name;  this.brand = brand;  this.madeIn = madeIn;  this.price = price;  } } |  @NotNull : null 값을 허용하지 않음 → 반드시 가격을 입력해야 함   @Positive : 양수(0을 포함하지 않는 양수)만 허용 (0은 허용되지 않음). “0 이하일 경우 에러”   @Digits(integer = 10, fraction = 2) :   * 정수 부분 최대 10자리, 소수점 둘째 자리까지(소수점 아래 최대 2자리) 허용 * message 속성으로 “소수점 둘째자리까지 허용” 메시지를 지정 |
| ProductRepository.java  package kr.ac.hansung.cse.hellospringdatajpa.repository;  import kr.ac.hansung.cse.hellospringdatajpa.entity.Product; import org.springframework.data.domain.Pageable; import org.springframework.data.jpa.repository.JpaRepository; import org.springframework.data.jpa.repository.Query;  import java.util.List;  public interface ProductRepository extends JpaRepository<Product, Long> {  Product findByName(String name);  List<Product> findByNameContaining(String searchKeyword, Pageable paging);   // used to retrieve records from the Product entity  // where the name attribute contains a specific substring  //JPQL(Java Persistence Query Language)을 사용한 사용자 정의 쿼리, %는 0개 이상 문자와 일치  @Query("select p from Product p where p.name like %?1%")  List<Product> searchByName(String name); } |  |
| RoleRepository.java  package kr.ac.hansung.cse.hellospringdatajpa.repository;  import kr.ac.hansung.cse.hellospringdatajpa.entity.MyRole; import org.springframework.data.jpa.repository.JpaRepository;  import java.util.Optional;  public interface RoleRepository extends JpaRepository<MyRole, Integer> {  Optional<MyRole> findByRolename(String rolename); } |  |
| UserRepository.java  package kr.ac.hansung.cse.hellospringdatajpa.repository;  import kr.ac.hansung.cse.hellospringdatajpa.entity.MyUser; import org.springframework.data.jpa.repository.JpaRepository;  import java.util.Optional;  public interface UserRepository extends JpaRepository<MyUser, Integer> {  Optional<MyUser> findByEmail(String email); } |  |
| CustomUserDetailsService.java  package kr.ac.hansung.cse.hellospringdatajpa.service;  import kr.ac.hansung.cse.hellospringdatajpa.entity.MyUser; import kr.ac.hansung.cse.hellospringdatajpa.repository.UserRepository; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.security.core.GrantedAuthority; import org.springframework.security.core.authority.AuthorityUtils; import org.springframework.security.core.userdetails.UserDetails; import org.springframework.security.core.userdetails.UserDetailsService; import org.springframework.security.core.userdetails.UsernameNotFoundException; import org.springframework.stereotype.Service; import org.springframework.transaction.annotation.Transactional;  import java.util.Collection;  // 인증 관리자는 UserDetailsService 를 통해 UserDetails 객체를 획득하고 // 이 UserDetails 객체에서 인증 및 인가에 필요한 정보를 추출하여 사용한다. @Service @Transactional public class CustomUserDetailsService implements UserDetailsService {  @Autowired  private UserRepository userRepository;   @Override  public UserDetails loadUserByUsername(String userName)  throws UsernameNotFoundException {  MyUser myUser = userRepository.findByEmail(userName)  .orElseThrow(() -> new UsernameNotFoundException("Email: " + userName + " not found"));  return new org.springframework.security.core.userdetails.User(myUser.getEmail(),  myUser.getPassword(), getAuthorities(myUser));  }   private static Collection<? extends GrantedAuthority> getAuthorities(MyUser user)  {  String[] userRoles = user.getRoles()  .stream()  .map((role) -> role.getRolename())  .toArray(String[]::new);   Collection<GrantedAuthority> authorities = AuthorityUtils.createAuthorityList(userRoles);  return authorities;  } } | * 이메일을 기준으로 데이터베이스에서 사용자 정보를 조회하여 UserDetails 객체로 변환 * 조회된 사용자 비밀번호와 권한 목록을 Spring Security 인증 프로세스에 제공하여 로그인 처리를 수행 |
| ProductService.java  package kr.ac.hansung.cse.hellospringdatajpa.service;  import kr.ac.hansung.cse.hellospringdatajpa.entity.Product; import kr.ac.hansung.cse.hellospringdatajpa.repository.ProductRepository; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.stereotype.Service; import org.springframework.transaction.annotation.Transactional;  import java.util.List; import java.util.NoSuchElementException;  @Service @Transactional public class ProductService {   @Autowired  private ProductRepository repo;   public Product get(long id) {  return repo.findById(id)  .orElseThrow(() -> new NoSuchElementException("Product not found with id: " + id));  }   public List<Product> listAll() {  return repo.findAll();  }   public void save(Product product) {  repo.save(product);  }   public void delete(long id) {  repo.deleteById(id);  } } | * ProductRepository를 통해 상품 조회, 저장, 삭제 기능을 캡슐화 |
| RegistrationService.java  package kr.ac.hansung.cse.hellospringdatajpa.service;  import kr.ac.hansung.cse.hellospringdatajpa.entity.MyRole; import kr.ac.hansung.cse.hellospringdatajpa.entity.MyUser;  import java.util.List;  public interface RegistrationService {  MyUser createUser(MyUser user, List<MyRole> userRoles);   boolean checkEmailExists(String email);   MyRole findByRolename(String rolename); } | * 사용자 생성, 이메일 중복 확인, 권한 조회 기능을 추상 메서드로 선언 * 구현체가 회원가입 로직 전반을 수행하도록 계약 정의 |
| RegistrationServiceImpl.java  package kr.ac.hansung.cse.hellospringdatajpa.service;  import kr.ac.hansung.cse.hellospringdatajpa.entity.MyRole; import kr.ac.hansung.cse.hellospringdatajpa.entity.MyUser; import kr.ac.hansung.cse.hellospringdatajpa.repository.RoleRepository; import kr.ac.hansung.cse.hellospringdatajpa.repository.UserRepository; import org.slf4j.Logger; import org.slf4j.LoggerFactory; import org.springframework.beans.factory.annotation.Autowired; import org.springframework.security.crypto.password.PasswordEncoder; import org.springframework.stereotype.Service; import org.springframework.transaction.annotation.Transactional;  import java.util.List; import java.util.Optional;  @Service @Transactional public class RegistrationServiceImpl implements RegistrationService {  private final Logger logger = LoggerFactory.getLogger(this.getClass());   @Autowired  private UserRepository userRepository;   @Autowired  private RoleRepository roleRepository;   @Autowired  private PasswordEncoder passwordEncoder;   public MyUser createUser(MyUser user, List<MyRole> userRoles) {  for (MyRole ur : userRoles) {  if (roleRepository.findByRolename(ur.getRolename()).isEmpty()) {  roleRepository.save(ur);  }  }   // generate new Bcrypt hash  String encryptedPassword = passwordEncoder.encode(user.getPassword());  user.setPassword(encryptedPassword);   user.setRoles(userRoles);   // User 저장  MyUser newUser = userRepository.save(user);   return newUser;  }   // 이메일 중복 체크  public boolean checkEmailExists(String email) {  if (userRepository.findByEmail(email).isPresent()) {  return true;  }   return false;  }   public MyRole findByRolename(String rolename) {  Optional<MyRole> role = roleRepository.findByRolename(rolename);  return role.orElseGet(() -> new MyRole(rolename));  }  } | * 전달된 사용자와 권한 목록을 받아, 권한이 DB에 없으면 먼저 저장, 비밀번호르르 BCrypt로 암호화 후 MyUser를 저장 * 이메일 중복 여부 체크, 특정 권한 이름으로 MyRole을 조회, 새로 생성하여 회원가입 시 역할 할당 |
| admin\_user.html  <!DOCTYPE html> <html xmlns="http://www.w3.org/1999/xhtml"  xmlns:th="http://www.thymeleaf.org"  xmlns:sec="https://www.thymeleaf.org/extras/spring-security"> <head th:insert="~{common :: commonHead}">  <title>Admin: User List</title> </head> <body> <div class="container">  <h1 class="mt-5">[Admin] 전체 사용자 목록</h1>   <!-- 상단에 관리자인지 확인하는 메시지 -->  <div class="mb-3">  <span sec:authorize="hasRole('ADMIN')">  관리자 <b sec:authentication="name"></b>님, 아래는 전체 사용자 목록입니다.  <a th:href="@{/logout}" class="btn btn-secondary btn-sm">로그아웃</a>  </span>  </div>   <table class="table table-bordered">  <thead>  <tr>  <th>User ID</th>  <th>Email</th>  <th>Roles</th>  </tr>  </thead>  <tbody>  <tr th:each="user : ${users}">  <td th:text="${user.id}">ID</td>  <td th:text="${user.email}">Email</td>  <td>  <span th:each="role : ${user.roles}"  th:text="${role.rolename} + ' '">Role</span>  </td>  </tr>  </tbody>  </table> </div> <div th:insert="~{common :: commonScript}"></div> </body> </html> |  |
| edit\_product.html  <!DOCTYPE html> <html xmlns="http://www.w3.org/1999/xhtml"  xmlns:th="http://www.thymeleaf.org"  xmlns:sec="https://www.thymeleaf.org/extras/spring-security">  <head th:insert="~{common :: commonHead}">  <title>Edit Product</title> </head>  <body> <div class="container">  <h1 class="mt-5 text-center">Edit Product</h1>  <br />   <div sec:authorize="hasRole('ADMIN')">  <form th:action="@{/products/save}" th:object="${product}" method="post">  <div class="row justify-content-center">  <div class="col-md-6">  <div class="mb-3">  <label for="productId" class="form-label">Product ID:</label>  <input type="text" id="productId" class="form-control" th:field="\*{id}" readonly />  </div>   <div class="mb-3">  <label for="productName" class="form-label">Product Name:</label>  <input type="text" id="productName" class="form-control" th:field="\*{name}" />  <!-- name 필드 검증 에러 -->  <div th:if="${#fields.hasErrors('name')}" class="text-danger mt-1">  <p th:errors="\*{name}"></p>  </div>  </div>   <div class="mb-3">  <label for="brand" class="form-label">Brand:</label>  <input type="text" id="brand" class="form-control" th:field="\*{brand}" />  <!-- brand 필드 검증 에러 -->  <div th:if="${#fields.hasErrors('brand')}" class="text-danger mt-1">  <p th:errors="\*{brand}"></p>  </div>  </div>   <div class="mb-3">  <label for="madeIn" class="form-label">Made In:</label>  <input type="text" id="madeIn" class="form-control" th:field="\*{madeIn}" />  <!-- madeIn 필드 검증 에러 -->  <div th:if="${#fields.hasErrors('madeIn')}" class="text-danger mt-1">  <p th:errors="\*{madeIn}"></p>  </div>  </div>   <!-- Price -->  <div class="mb-3">  <label for="price" class="form-label">Price:</label>  <input type="number"  id="price"  class="form-control"  th:field="\*{price}"  min="0"  step="0.01"  placeholder="가격을 입력하세요"  required />  <!-- price 필드 검증 에러 -->  <div th:if="${#fields.hasErrors('price')}" class="text-danger mt-1">  <p th:errors="\*{price}"></p>  </div>  </div>   <div class="text-center">  <button type="submit" class="btn btn-primary">Save</button>  </div>  </div>  </div>  </form>  </div>   <div sec:authorize="!hasRole('ADMIN')" class="alert alert-danger text-center">  접근 권한이 없습니다.  </div> </div>  <!-- Bootstrap JS 로딩 --> <div th:insert="~{common :: commonScript}"></div> </body> </html> |  |
| index.html  <!DOCTYPE html> <html xmlns="http://www.w3.org/1999/xhtml"  xmlns:th="http://www.thymeleaf.org"  xmlns:sec="https://www.thymeleaf.org/extras/spring-security">  <head th:insert="~{common :: commonHead}">  <title>Product Manager</title> </head>  <body> <div class="container">  <h1 class="mt-5">Product List</h1>   <!-- 로그인 상태/권한별 상단 메뉴 -->  <div class="mb-3">  <span sec:authorize="isAuthenticated()">  환영합니다, <b sec:authentication="name"></b>님!  <a th:href="@{/logout}" class="btn btn-secondary btn-sm">로그아웃</a>  </span>  <span sec:authorize="!isAuthenticated()">  <a th:href="@{/login}" class="btn btn-primary btn-sm">로그인</a>  <a th:href="@{/signup}" class="btn btn-success btn-sm">회원가입</a>  </span>  </div>   <!-- ADMIN만 보이는 “전체 사용자 보기” 버튼 -->  <div class="mb-3" sec:authorize="hasRole('ADMIN')">  <a th:href="@{/admin/users}" class="btn btn-warning">전체 사용자 보기</a>  </div>   <!-- ROLE\_ADMIN인 경우에만 New 버튼 노출 -->  <div sec:authorize="hasRole('ADMIN')">  <a class="btn btn-primary mb-3" th:href="@{/products/new}">Create New Product</a>  </div>   <table class="table table-bordered">  <thead>  <tr>  <th>Product ID</th>  <th>Name</th>  <th>Brand</th>  <th>Made In</th>  <th>Price</th>  <th>Actions</th>  </tr>  </thead>  <tbody>  <tr th:each="product : ${listProducts}">  <td th:text="${product.id}">Product ID</td>  <td th:text="${product.name}">Name</td>  <td th:text="${product.brand}">Brand</td>  <td th:text="${product.madeIn}">Made in</td>  <td th:text="${product.price}">Price</td>  <td>  <!-- ROLE\_ADMIN인 경우에만 Edit/Delete 버튼 보여주기 -->  <a class="btn btn-primary btn-sm"  th:href="@{'/products/edit/' + ${product.id}}"  sec:authorize="hasRole('ADMIN')">Edit</a>  <a class="btn btn-danger btn-sm"  th:href="@{'/products/delete/' + ${product.id}}"  th:onclick="return confirm('정말 삭제하시겠습니까?')"  sec:authorize="hasRole('ADMIN')">Delete</a>  <!-- ROLE\_USER인 경우에는 단순히 조회만 가능하므로 빈 칸 노출하거나,  "권한 없음" 메시지를 뿌릴 수도 있음 -->  <span sec:authorize="hasRole('USER') and !hasRole('ADMIN')">  권한 없음  </span>  </td>  </tr>  </tbody>  </table> </div>  <!-- Bootstrap JS fragment 삽입 --> <div th:insert="~{common :: commonScript}"></div> </body> </html> |  |
| login.html  <!doctype html> <html xmlns="http://www.w3.org/1999/xhtml"  xmlns:th="http://www.thymeleaf.org"> <head>  <meta charset="utf-8">  <meta name="viewport" content="width=device-width, initial-scale=1">   <!-- Bootstrap 5.3.3 CSS -->  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"  rel="stylesheet"  integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"  crossorigin="anonymous">   <!-- Custom signin.css -->  <link th:href="@{/css/signin.css}" rel="stylesheet">   <title>Login Page</title> </head> <body>  <div class="container mt-5">  <div class="row justify-content-center">  <div class="col-md-6">  <h1 class="text-center mb-4">로그인</h1>   <form method="post" th:action="@{/login}">  <div th:if="${param.error}" class="alert alert-danger" role="alert">  Invalid Email and Password.  </div>   <div th:if="${param.logout}" class="alert alert-info" role="alert">  Successfully logged out  </div>   <div class="form-floating mb-2">  <input type="email" id="username" name="username" class="form-control" placeholder="Email" required autofocus>  <label for="username">Email address</label>  </div>   <div class="form-floating mb-3">  <input type="password" id="password" name="password" class="form-control" placeholder="Password" required>  <label for="password">Password</label>  </div>   <div class="d-grid">  <button class="btn btn-primary btn-lg" type="submit">Sign In</button>  </div>  </form>  </div>  </div> </div>  <!-- Bootstrap JS Bundle --> <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"  integrity="sha384-ENjdO4Dr2bkBIFxQpeoAZJy0pOer+AfD5VXWfVZlAUa0xNfQ0aEw+7F7bPjD1L3y"  crossorigin="anonymous"></script> </body> </html> |  |
| new\_product.html  <!DOCTYPE html> <html xmlns="http://www.w3.org/1999/xhtml"  xmlns:th="http://www.thymeleaf.org"  xmlns:sec="https://www.thymeleaf.org/extras/spring-security">  <head th:insert="~{common :: commonHead}">  <title>Create New Product</title> </head>  <body> <div class="container">  <h1 class="mt-5 text-center">Create New Product</h1>  <br />   <!-- ADMIN만 접근 가능한 페이지지만, 혹시 인증된 USER가 직접 URL로 접근했을 때를 대비하여  sec:authorize 어노테이션으로 보안 처리 -->  <div sec:authorize="hasRole('ADMIN')">  <form th:action="@{/products/save}" th:object="${product}" method="post">  <div class="row justify-content-center">  <div class="col-md-6">  <div class="mb-3">  <label for="productName" class="form-label">Product Name:</label>  <input type="text" id="productName" class="form-control" th:field="\*{name}" />  <!-- name 필드 검증 에러 -->  <div th:if="${#fields.hasErrors('name')}" class="text-danger mt-1">  <p th:errors="\*{name}"></p>  </div>  </div>  <div class="mb-3">  <label for="brand" class="form-label">Brand:</label>  <input type="text" id="brand" class="form-control" th:field="\*{brand}" />  <!-- brand 필드 검증 에러 -->  <div th:if="${#fields.hasErrors('brand')}" class="text-danger mt-1">  <p th:errors="\*{brand}"></p>  </div>  </div>  <div class="mb-3">  <label for="madeIn" class="form-label">Made In:</label>  <input type="text" id="madeIn" class="form-control" th:field="\*{madeIn}" />  <!-- madeIn 필드 검증 에러 -->  <div th:if="${#fields.hasErrors('madeIn')}" class="text-danger mt-1">  <p th:errors="\*{madeIn}"></p>  </div>  </div>  <!-- Price -->  <div class="mb-3">  <label for="price" class="form-label">Price:</label>  <input type="number"  id="price"  class="form-control"  th:field="\*{price}"  min="0"  step="0.01"  placeholder="가격을 입력하세요"  required />  <!-- price 필드 검증 에러 -->  <div th:if="${#fields.hasErrors('price')}" class="text-danger mt-1">  <p th:errors="\*{price}"></p>  </div>  </div>  <div class="text-center">  <button type="submit" class="btn btn-primary">Save</button>  </div>  </div>  </div>  </form>  </div>  <div sec:authorize="!hasRole('ADMIN')" class="alert alert-danger text-center">  접근 권한이 없습니다.  </div> </div>  <!-- Bootstrap JS 삽입 --> <div th:insert="~{common :: commonScript}"></div> </body> </html> |  |
| signup.html  <!doctype html> <html xmlns="http://www.w3.org/1999/xhtml"  xmlns:th="http://www.thymeleaf.org"> <head>  <meta charset="utf-8">  <meta name="viewport" content="width=device-width, initial-scale=1">   <!-- Bootstrap 5.3.3 CSS -->  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/css/bootstrap.min.css"  rel="stylesheet"  integrity="sha384-QWTKZyjpPEjISv5WaRU9OFeRpok6YctnYmDr5pNlyT2bRjXh0JMhjY6hW+ALEwIH"  crossorigin="anonymous">   <title>SignUp Page</title> </head> <body>  <div class="container mt-5">  <div class="row justify-content-center">  <div class="col-md-6">  <h1 class="text-center mb-4">회원 가입</h1>   <!-- Form 시작 -->  <form th:action="@{/signup}" th:object="${user}" method="post">   <!-- 이메일 중복 경고 -->  <div th:if="${emailExists}" class="alert alert-danger" role="alert">  Email already exists  </div>   <div class="mb-3">  <label for="email" class="form-label">Your Email:</label>  <input type="email" class="form-control" id="email" th:field="\*{email}" required />  </div>   <div class="mb-3">  <label for="password" class="form-label">Password:</label>  <input type="password" class="form-control" id="password" th:field="\*{password}" required />  </div>   <div class="d-grid">  <button type="submit" class="btn btn-primary btn-lg">Sign Up</button>  </div>  </form>  </div>  </div> </div>  <!-- Bootstrap JS Bundle --> <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.3/dist/js/bootstrap.bundle.min.js"  integrity="sha384-ENjdO4Dr2bkBIFxQpeoAZJy0pOer+AfD5VXWfVZlAUa0xNfQ0aEw+7F7bPjD1L3y"  crossorigin="anonymous"></script> </body> </html> |  |
| application.properties  spring.application.name=helloSpringDataJpa  # context path: http://localhost:8080/helloSpringDataJpa server.servlet.context-path=/helloSpringDataJpa  # === DataSource === spring.datasource.url=jdbc:mysql://localhost:3306/sales?useSSL=false&characterEncoding=UTF-8&serverTimezone=Asia/Seoul spring.datasource.username=root spring.datasource.password=csedbadmin spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  # === SQL 초기화 (개발 프로파일 전용) === # executes initialization scripts(schema.sql, data.sql) every time the application is run spring.sql.init.mode=always # used to configure the encoding of initialization scripts spring.sql.init.encoding= UTF-8  # === JPA === # 운영은 validate, 개발은 create 또는 update spring.jpa.hibernate.ddl-auto=create spring.jpa.show-sql=true spring.jpa.properties.hibernate.format\_sql=false # After the ddl-auto execution, data.sql is executed and the data is applied spring.jpa.defer-datasource-initialization=true  # === Logging === logging.level.kr.ac.hansung=trace |  |
| HelloSpringDataApllication.java  package kr.ac.hansung.cse.hellospringdatajpa;  import org.springframework.boot.SpringApplication; import org.springframework.boot.autoconfigure.SpringBootApplication;  @SpringBootApplication public class HelloSpringDataJpaApplication {   public static void main(String[] args) {  SpringApplication.run(HelloSpringDataJpaApplication.class, args);  }  } |  |
| data.sql  insert into product (name, brand, made\_in, price) values ('Galaxy S6', 'Samsung Corp', 'Korea', 600.0); insert into product (name, brand, made\_in, price) values ('Galaxy S8', 'Samsung Corp', 'Korea', 800.0); insert into product (name, brand, made\_in, price) values ('Galaxy S10', 'Samsung Corp', 'Korea', 1000.0); insert into product (name, brand, made\_in, price) values ('Galaxy S21', 'Samsung Corp', 'Korea', 1000.0);  insert into product (name, brand, made\_in, price) values ('MacBook Air1', 'Apple', 'China', 10000); insert into product (name, brand, made\_in, price) values ('MacBook Air2', 'Apple', 'China', 10000); insert into product (name, brand, made\_in, price) values ('MacBook Air3', 'Apple', 'China', 10000); insert into product (name, brand, made\_in, price) values ('MacBook Air4', 'Apple', 'China', 10000); insert into product (name, brand, made\_in, price) values ('MacBook Air5', 'Apple', 'China', 10000); insert into product (name, brand, made\_in, price) values ('MacBook Pro1', 'Apple', 'China', 15000); insert into product (name, brand, made\_in, price) values ('MacBook Pro2', 'Apple', 'China', 15000);  insert into product (name, brand, made\_in, price) values ('iPad Air', 'Apple', 'China', 500); insert into product (name, brand, made\_in, price) values ('iPad Pro', 'Apple', 'China', 800);  insert into product (name, brand, made\_in, price) values ('소나타', 'Hyundai', 'Japan', 20000); insert into product (name, brand, made\_in, price) values ('그랜저', 'Hyundai', 'Japan', 30000); insert into product (name, brand, made\_in, price) values ('제너시스', 'Hyundai', 'Japan', 50000); insert into product (name, brand, made\_in, price) values ('에쿠스', 'Hyundai', 'Japan', 60000);  insert into product (name, brand, made\_in, price) values ('Accord', 'Honda', 'Japan', 25000); insert into product (name, brand, made\_in, price) values ('sienna', 'Honda', 'Japan', 40000);  insert into product (name, brand, made\_in, price) values ('Camry', 'Toyota', 'Japan', 25000); insert into product (name, brand, made\_in, price) values ('Lexus', 'Toyota', 'Japan', 50000); |  |
|  |  |

**단계별 수행 결과를 보이는 스크린샷**

1. 회원가입(admin)

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

1. 회원가입(sjkim)

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

1. DB

Product

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

roles

텍스트, 스크린샷, 폰트, 번호이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

user\_role

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

users

텍스트, 폰트, 스크린샷이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

1. 로그인/로그아웃 (로그인 성공, 실패 시 사용자 맞춤 메시지 출력)

admin으로 로그인

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

sjkim으로 로그인

텍스트, 소프트웨어, 멀티미디어 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

로그인 실패

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

로그아웃

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

1. 1) admin인 경우 products 화면

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

2) sjkim인 경우 products 화면

텍스트, 스크린샷, 소프트웨어, 번호이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

1. 1) admin인 경우 상품 등록, 수정, 삭제(상품 등록/수정 시 유효성 검사)

텍스트, 소프트웨어, 컴퓨터 아이콘, 운영 체제이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

수정

텍스트, 스크린샷, 소프트웨어, 컴퓨터 아이콘이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

수정 반영

텍스트, 폰트, 번호, 라인이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

삭제

텍스트, 스크린샷, 소프트웨어, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

삭제 반영

텍스트, 폰트, 번호, 라인이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

1. ROLE\_ADMIN 사용자만 접근 가능한 관리자 페이지 구현(전체 사용자 목록 확인)

텍스트, 소프트웨어, 컴퓨터 아이콘, 멀티미디어 소프트웨어이(가) 표시된 사진

AI 생성 콘텐츠는 정확하지 않을 수 있습니다.

**자기평가**

* 1, 2, 3번 모두 구현 완료했습니다.

**과제 수행 느낀 점**

* 이번 과제를 통해 CRUD뿐만 아니라 보안을 이용해 실제 서비스 아키텍처를 경험할 수 있었다. Spring Boot와 Spring Security의 자동 설정 덕분에 설정 파일과 몇 줄의 코드로 인증 및 인가 기능을 적용할 수 있음을 깨닫고 사람들이 왜 Spring Boot가 편하다고 말하는지 알 수 있는 계기가 되었다. 정보보안 시간에 배운 비밀번호 암호화(salt와 hash 사용)를 연결해서 배우니까 더 이해가 잘 되고 보안 쪽이 어렵지 않고 친근하게 다가왔다.