

GitHub Repository 링크

<https://github.com/rlathwls03/helloSpringDataJpa>

주요 소스 및 설명 (함수 설명은 주석으로 첨부)

<p>WebConfig.java</p> <pre>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(); } }</pre>	<ul style="list-style-type: none">- 뷰 컨트롤러 등록- Thymeleaf에서 Spring Security 태그 사용 활성화
<p>WebSecurityConfig.java</p> <pre>package kr.ac.hansung.cse.hellospringdatajpa.config; import</pre>	<ul style="list-style-type: none">- Spring Security 필터 체인 구성- 인증, 인가 규

```

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/**"
    };
};

```

칙 정의

- 로그인 로그아웃 설정
- 사용자 정보 조회 방법 등록
- 비밀번호 암호화 방법 설정
- CSRF 토큰 설정

```

@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").permi
tAll()
            .requestMatchers("/admin/**").ha
sRole("ADMIN")
            // 상품 목록 조회는 ROLE_USER 이상
(ROLE_USER, ROLE_ADMIN)
            .requestMatchers("/products").ha
sAnyRole("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.html 에서 form 필드 name="username"
            .passwordParameter("password")
// login.html 에서 form 필드 name="password"
            .permitAll()
        )
        // 로그아웃 설정
        .logout(logout -> logout
            .logoutUrl("/logout")
            .logoutSuccessUrl("/login?logout
")
            .permitAll()
        )
        .exceptionHandling(exceptions ->
exceptions
            .accessDeniedPage("/accessDenied
")
        )
        .userDetailsService(customUserDetailsSe
rvice)
        // CSRF 비활성화: GET /logout 을
허용하도록 설정
        .csrf(csrf -> csrf.disable());
        /* .csrf(csrf -> csrf
            .ignoringRequestMatchers("/api/*
*")); */

    return http.build();
}

```

<pre>} }</pre>	
<p>AdminController.java</p> <pre>package kr.ac.hansung.cse.hellospringdatajpa.controller; import kr.ac.hansung.cse.hellospringdatajpa.entity.MyUser; import kr.ac.hansung.cse.hellospringdatajpa.repository.User Repository; import org.springframework.beans.factory.annotation.Autowir ed; import org.springframework.security.access.prepost.PreAutho rize; 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_ADMIN 만 접근) @GetMapping("/users") // Method Security 를 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 } }</pre>	<ul style="list-style-type: none"> - 관리자용 기능 이 모여있는 컨트롤러
<p>ProductController.java</p> <pre>package kr.ac.hansung.cse.hellospringdatajpa.controller; import kr.ac.hansung.cse.hellospringdatajpa.entity.Product; import kr.ac.hansung.cse.hellospringdatajpa.service.Product Service; import org.springframework.beans.factory.annotation.Autowir ed; import org.springframework.stereotype.Controller;</pre>	<ul style="list-style-type: none"> - 상품 관련 CRUD 기능을 담당 - /products/** 경로로 들어오 는 요청을 처 리

```

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_ADMIN 만 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_ADMIN 만 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";
    }

    // @ModelAttribute 는 Form data (예:
name=Laptop&brand=Samsung&madeIn=Korea&price=1000.00
)를 Product 객체
    // @RequestBody 는 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_ADMIN 만 템플릿 단에서 노출됨)
    @GetMapping("/delete/{id}")
    public String deleteProduct(@PathVariable(name =
"id") Long id) {

        service.delete(id);
        return "redirect:/products";
    }
}

```

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.Registr
ationService;
import
org.springframework.beans.factory.annotation.Autowir
ed;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import
org.springframework.web.bind.annotation.ModelAttribu
te;
import
org.springframework.web.bind.annotation.RequestMappi
ng;
import
org.springframework.web.bind.annotation.RequestMetho
d;

```

- 사용자 회원가입 기능 담당
- 로그인하지 않은 상태에서도 /signup으로 접근 가능

```

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:/";
        }
    }
}

```

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 name 은 필수
입력입니다.")
    private String name;

    @NotBlank(message = "Brand 는 필수 입력입니다.")
    private String brand;

    @NotBlank(message = "MadeIn 은 필수 입력입니다.")
    private String madeIn;

    // 1) BigDecimal 타입으로 변경
    // 2) precision=10, scale=2 로 컬럼을
DECIMAL(10,2) 형태로 생성
    @NotNull(message = "Price 는 필수 입력입니다.")
    @Positive(message = "Price 는 0 보다 커야 합니다.")
    @Digits(integer = 10, fraction = 2, message =
"Price 는 소수점 둘째자리까지 허용합니다.")
    @Column(precision = 10, scale = 2)
    private BigDecimal price;
}

```

- @NotNull : null 값을 허용하지 않음
→ 반드시 가격을 입력해야 함
- @Positive : 양수(0 을 포함하지 않는 양수)만 허용 (0 은 허용되지 않음). “0 이하일 경우 에러”
- @Digits(integer = 10, fraction = 2) :
 - 정수 부분 최대 10 자리, 소수점 둘째 자리까지(소수점 아래 최대 2 자리) 허용
 - message 속성으로 “소수점 둘째자리까지 허용” 메시지를 지정

```

        public Product(String name, String brand, String
madeIn, BigDecimal price) {
            this.name = name;
            this.brand = brand;
            this.madeIn = madeIn;
            this.price = price;
        }
    }
}

```

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 0 개 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> {

```

<pre>Optional<MyRole> findByRolename(String rolename); }</pre>	
<p>UserRepository.java</p> <pre>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); }</pre>	
<p>CustomUserDetailsService.java</p> <pre>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</pre>	<ul style="list-style-type: none"> - 이메일을 기준으로 데이터베이스에서 사용자 정보를 조회하여 UserDetails 객체로 변환 - 조회된 사용자 비밀번호와 권한 목록을 Spring Security 인증 프로세스에 제공하여 로그인 처리를 수행

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(m
yUser.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;
    }
}
```

ProductService.java

```
package
kr.ac.hansung.cse.hellospringdatajpa.service;

import
kr.ac.hansung.cse.hellospringdatajpa.entity.Product;
import
kr.ac.hansung.cse.hellospringdatajpa.repository.Prod
uctRepository;
import
org.springframework.beans.factory.annotation.Autowir
ed;
import org.springframework.stereotype.Service;
import
org.springframework.transaction.annotation.Transacti
onal;

import java.util.List;
import java.util.NoSuchElementException;
```

- ProductReposit
ory를 통해 상
품 조회, 저장,
삭제 기능을
캡슐화

```

@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);
    }
}

```

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);
}

```

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

```

- 사용자 생성, 이메일 중복 확인, 권한 조회 기능을 추상 메서드로 선언
- 구현체가 회원가입 로직 전반을 수행하도록 계약 정의

- 전달된 사용자와 권한 목록을 받아, 권한이 DB에 없으면 먼저 저장, 비밀번호를 BCrypt로 암호

```

kr.ac.hansung.cse.hellospringdatajpa.repository.Role
Repository;
import
kr.ac.hansung.cse.hellospringdatajpa.repository.User
Repository;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import
org.springframework.beans.factory.annotation.Autowir
ed;
import
org.springframework.security.crypto.password.Passwor
dEncoder;
import org.springframework.stereotype.Service;
import
org.springframework.transaction.annotation.Transacti
onal;

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()).isE
mpty()) {
                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;
    }

```

화 후 MyUser
를 저장

- 이메일 중복
여부 체크, 특
정 권한 이름
으로 MyRole을
조회, 새로 생
성하여 회원가입
시 역할 할
당

```
// 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));
}
}
```

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>

  <!-- ADMIN 만 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_ADMIN 인 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_ADMIN 인 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?') ">Delete</a>
            <!-- ROLE_USER 인 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/d
ist/css/bootstrap.min.css"
rel="stylesheet"
integrity="sha384-
QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMh
jY6hW+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/di
st/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 />

  <!-- ADMIN 만 접근 가능한 페이지지만, 혹시 인증된
USER 가 직접 URL 로 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.jsdelivrivr.net/npm/bootstrap@5.3.3/d
ist/css/bootstrap.min.css"
    rel="stylesheet"

```

```

        integrity="sha384-
QWTKZyjpPEjISv5WaRU90FeRpok6YctnYmDr5pNlyT2bRjXh0JMh
jY6hW+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/di
st/js/bootstrap.bundle.min.js"
  integrity="sha384-
ENjdO4Dr2bkBIFxQpeoAZJy0pOer+Afd5VXWfVZ1AUa0xNfQ0aEw
+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

```

HelloSpringDataAppllication.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);

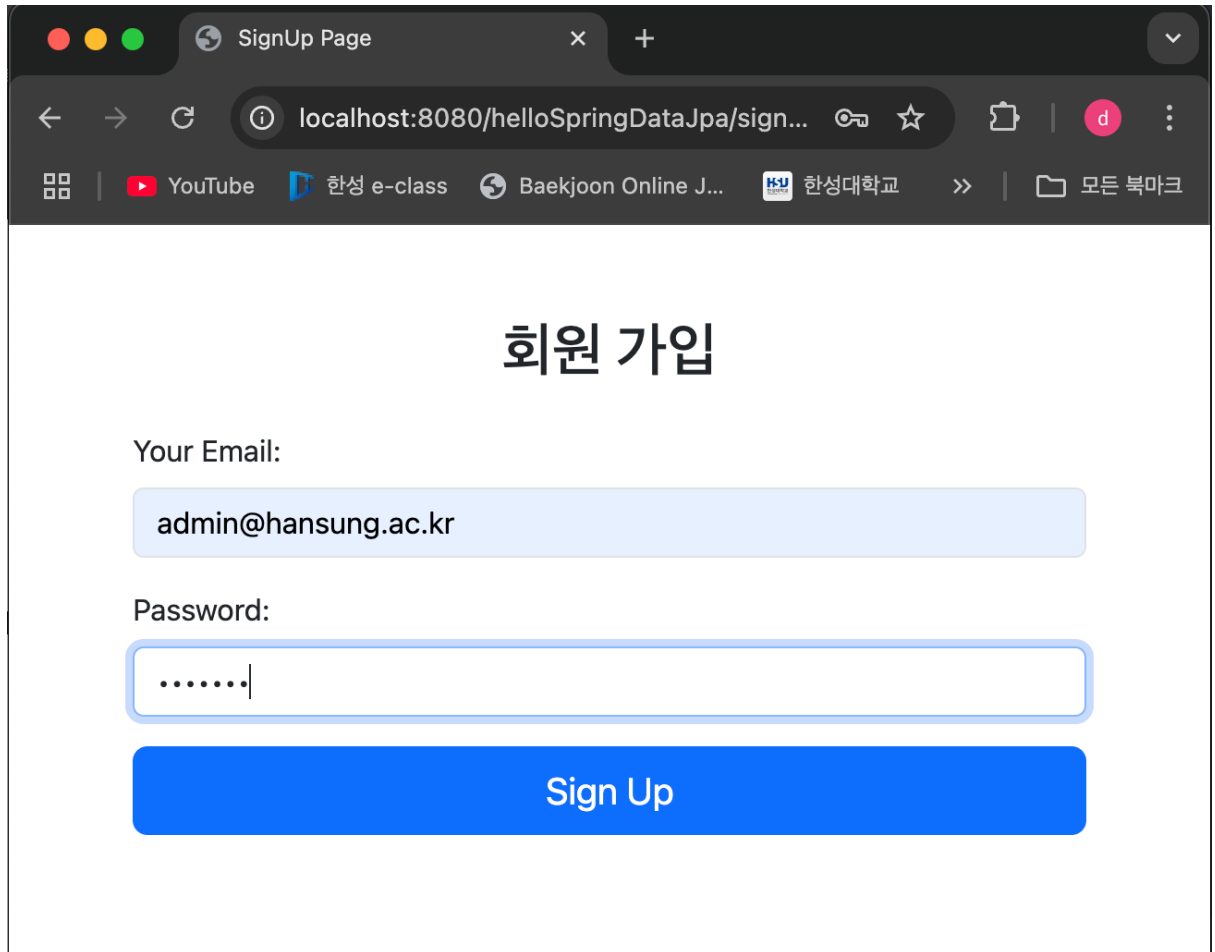
insert into product (name, brand, made_in, price)
values ('Accord', 'Honda', 'Japan', 25000);
insert into product (name, brand, made_in, price)
values ('sienna', 'Honda', 'Japan', 40000);

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)



회원 가입

Your Email:

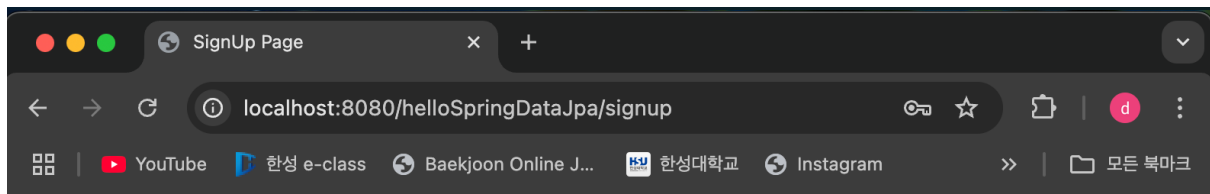
admin@hansung.ac.kr

Password:

.....

Sign Up

2. 회원가입(sjkim)



회원 가입

Your Email:

sjkim@hansung.ac.kr

Password:

.....

Sign Up

3. DB

Product

product					
WHERE					
ORDER BY					
	price	id	brand	made_in	name
1	600.00	1	Samsung Corp	Korea	Galaxy S6
2	800.00	2	Samsung Corp	Korea	Galaxy S8
3	1000.00	3	Samsung Corp	Korea	Galaxy S10
4	1000.00	4	Samsung Corp	Korea	Galaxy S21
5	10000.00	5	Apple	China	MacBook Air1
6	10000.00	6	Apple	China	MacBook Air2
7	10000.00	7	Apple	China	MacBook Air3
8	10000.00	8	Apple	China	MacBook Air4
9	10000.00	9	Apple	China	MacBook Air5
10	15000.00	10	Apple	China	MacBook Pro1
11	15000.00	11	Apple	China	MacBook Pro2
12	500.00	12	Apple	China	iPad Air
13	800.00	13	Apple	China	iPad Pro
14	20000.00	14	Hyundai	Japan	소나타
15	30000.00	15	Hyundai	Japan	그랜저
16	50000.00	16	Hyundai	Japan	제너시스
17	60000.00	17	Hyundai	Japan	에쿠스
18	25000.00	18	Honda	Japan	Accord
19	40000.00	19	Honda	Japan	sienna
20	25000.00	20	Toyota	Japan	Camry
21	50000.00	21	Toyota	Japan	Lexus

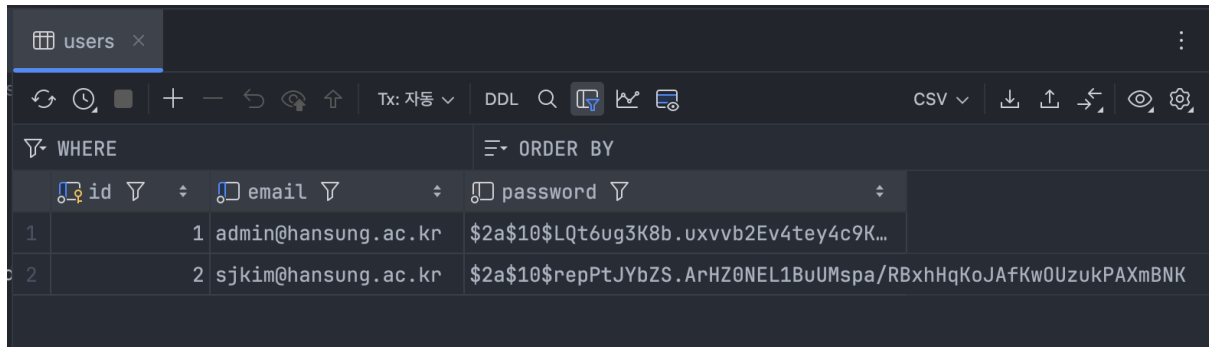
roles

roles	
Tx: 자동 ▼	
DDL	
WHERE	
ORDER BY	
id	rolename
1	ROLE_ADMIN
2	ROLE_USER

user_role

user_role	
Tx: 자동 ▼	
DDL	
WHERE	
ORDER BY	
role_id	user_id
1	1
2	1
3	2

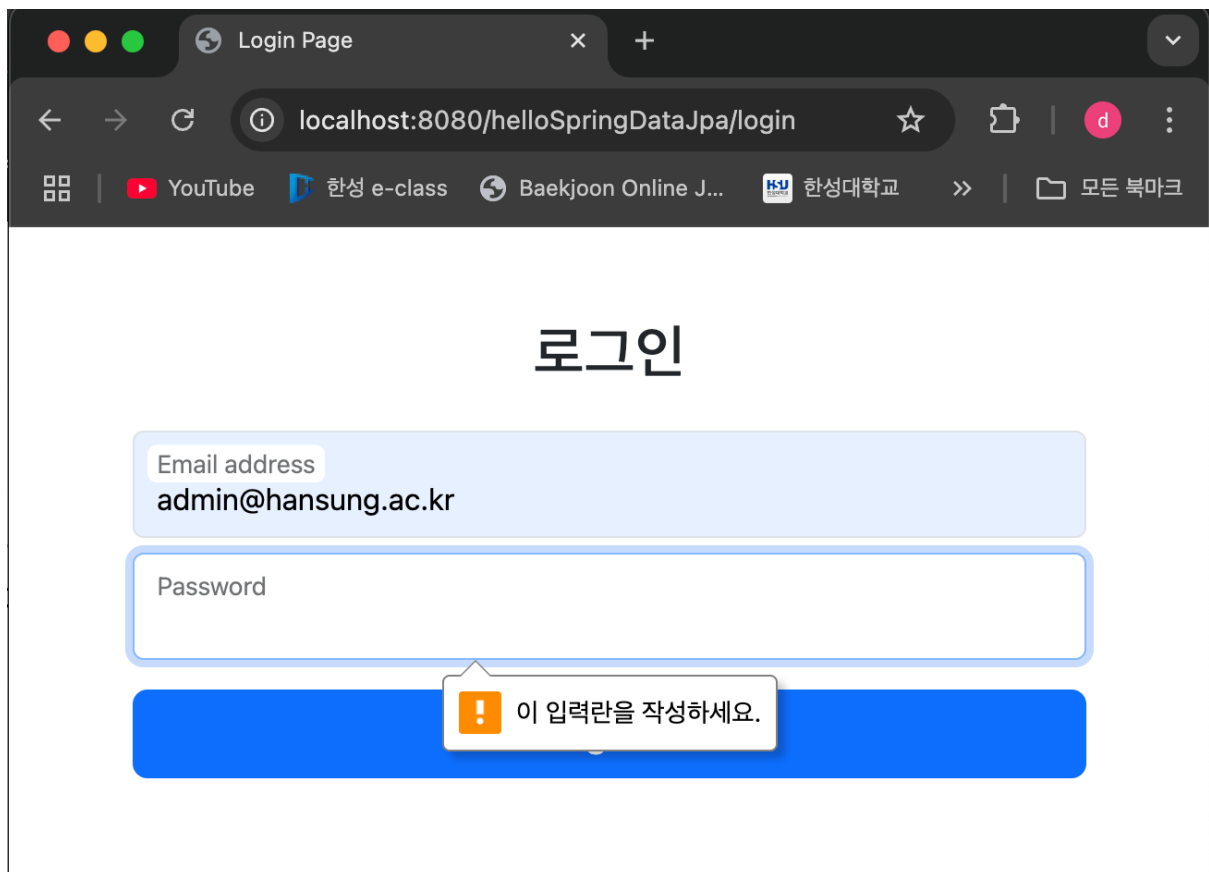
users



	id	email	password
1	1	admin@hansung.ac.kr	\$2a\$10\$LQt6ug3K8b.uxvzb2Ev4tey4c9K...
2	2	sjkim@hansung.ac.kr	\$2a\$10\$repPtJYbZS.ArHZ0NEL1BuUMspa/RBxhHqKoJAfKw0UzukPAXmBNK

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

admin으로 로그인



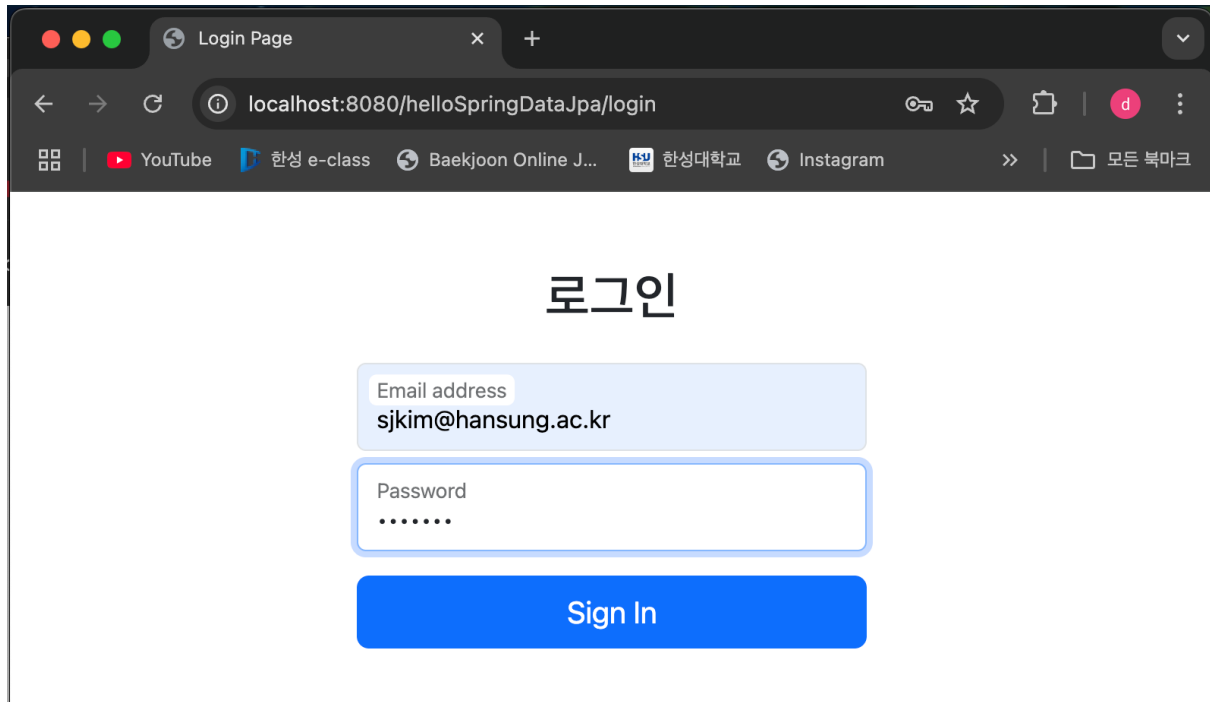
로그인

Email address
admin@hansung.ac.kr

Password

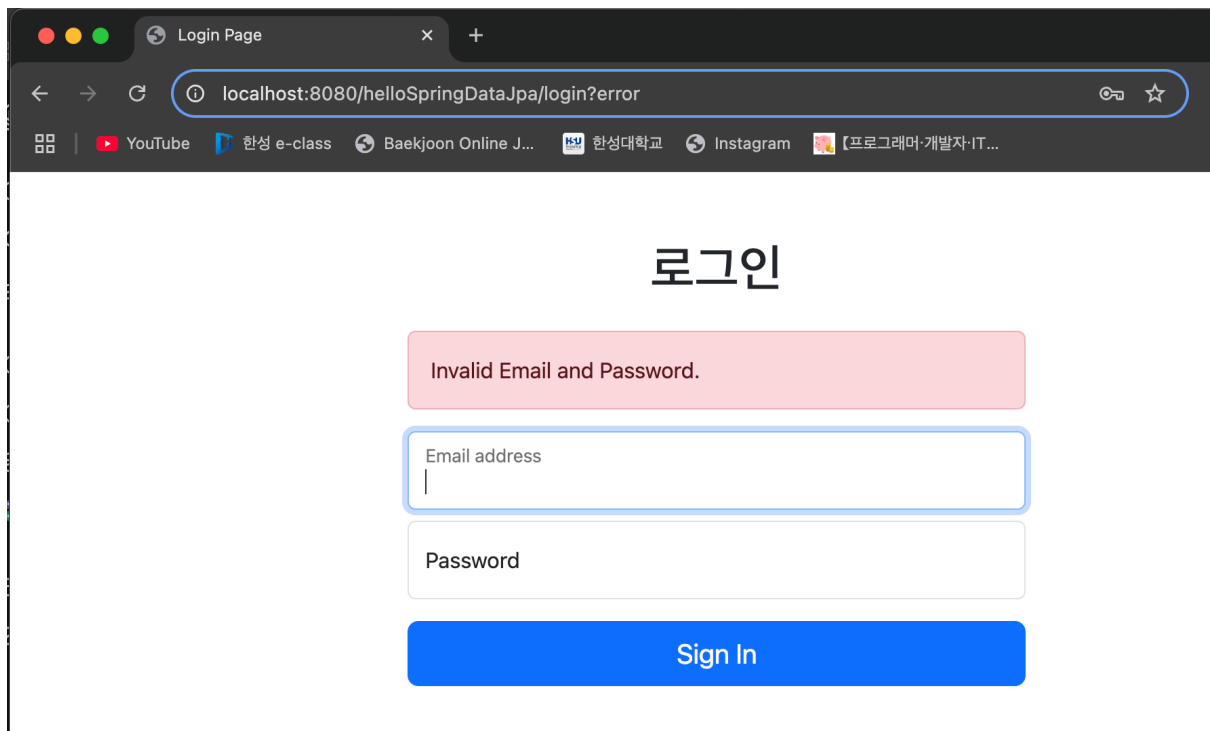
이 입력란을 작성하세요.

sjkim으로 로그인



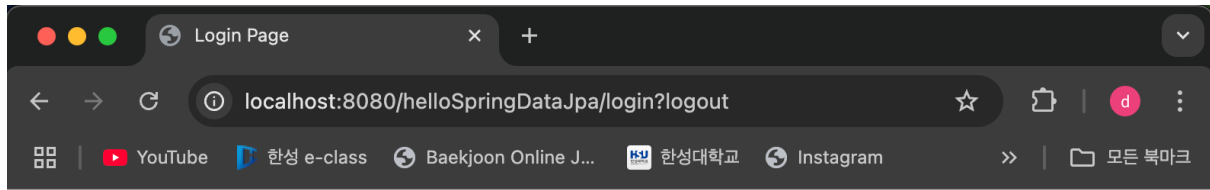
The screenshot shows a web browser window with the title "Login Page". The address bar displays "localhost:8080/helloSpringDataJpa/login". The browser's bookmark bar includes links to YouTube, 한성 e-class, Baekjoon Online J..., 한성대학교, and Instagram. The main content area features the heading "로그인" (Login) in large black text. Below the heading are two input fields: "Email address" with the value "sjkim@hansung.ac.kr" and "Password" with masked characters ".....". A blue "Sign In" button is positioned below the password field.

로그인 실패



The screenshot shows the same web browser window, but the address bar now includes "?error" at the end of the URL: "localhost:8080/helloSpringDataJpa/login?error". The main content area still has the heading "로그인". A red error message box appears above the input fields, displaying the text "Invalid Email and Password.". The "Email address" and "Password" input fields are now empty. The blue "Sign In" button remains at the bottom.

로그아웃

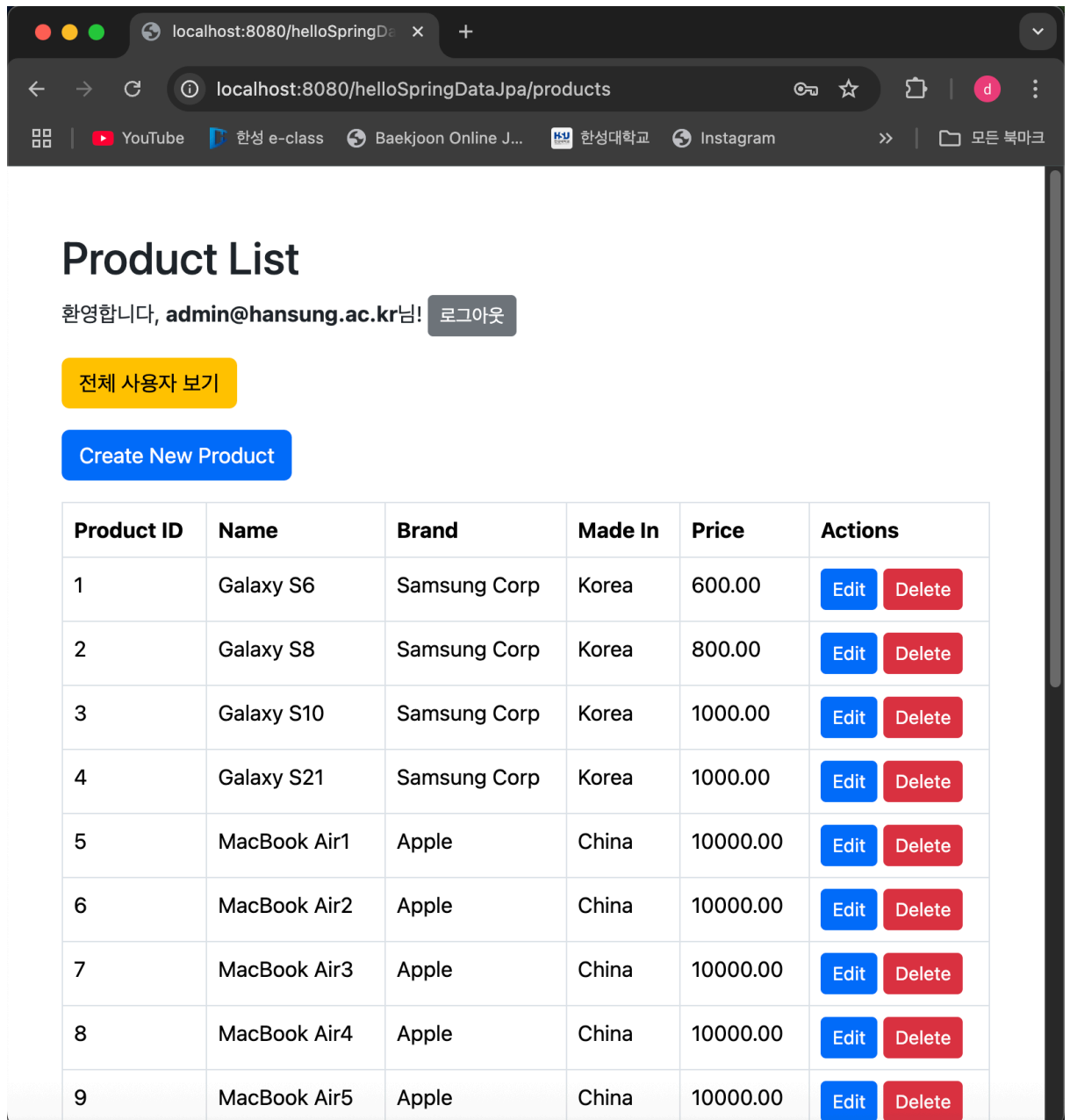


로그인

Successfully logged out

Sign In

5. 1) admin인 경우 products 화면



localhost:8080/helloSpringDataJpa/products

Product List

환영합니다, admin@hansung.ac.kr님! [로그아웃](#)

[전체 사용자 보기](#)

[Create New Product](#)

Product ID	Name	Brand	Made In	Price	Actions
1	Galaxy S6	Samsung Corp	Korea	600.00	Edit Delete
2	Galaxy S8	Samsung Corp	Korea	800.00	Edit Delete
3	Galaxy S10	Samsung Corp	Korea	1000.00	Edit Delete
4	Galaxy S21	Samsung Corp	Korea	1000.00	Edit Delete
5	MacBook Air1	Apple	China	10000.00	Edit Delete
6	MacBook Air2	Apple	China	10000.00	Edit Delete
7	MacBook Air3	Apple	China	10000.00	Edit Delete
8	MacBook Air4	Apple	China	10000.00	Edit Delete
9	MacBook Air5	Apple	China	10000.00	Edit Delete

2) sjkim인 경우 products 화면

localhost:8080/helloSpringDataJpa/products

Product List

환영합니다, sjkim@hansung.ac.kr님! [로그아웃](#)

Product ID	Name	Brand	Made In	Price	Actions
1	Galaxy S6	Samsung Corp	Korea	600.00	권한 없음
2	Galaxy S8	Samsung Corp	Korea	800.00	권한 없음
3	Galaxy S10	Samsung Corp	Korea	1000.00	권한 없음
4	Galaxy S21	Samsung Corp	Korea	1000.00	권한 없음
5	MacBook Air1	Apple	China	10000.00	권한 없음
6	MacBook Air2	Apple	China	10000.00	권한 없음
7	MacBook Air3	Apple	China	10000.00	권한 없음
8	MacBook Air4	Apple	China	10000.00	권한 없음
9	MacBook Air5	Apple	China	10000.00	권한 없음
10	MacBook Pro1	Apple	China	15000.00	권한 없음
11	MacBook Pro2	Apple	China	15000.00	권한 없음
12	iPad Air	Apple	China	500.00	권한 없음
13	iPad Pro	Apple	China	800.00	권한 없음

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

localhost:8080/helloSpringDc x +

localhost:8080/helloSpringDataJpa/products/save

☆

d

YouTube 한성 e-class Baekjoon Online J... 한성대학교 Instagram >> 모든 북마크

Create New Product

Product Name:

Product name은 필수 입력입니다.

Brand:

Brand는 필수 입력입니다.

Made In:

MadeIn은 필수 입력입니다.

Price:

0

Price는 0보다 커야 합니다.

Save

localhost:8080/helloSpringDe x +

localhost:8080/helloSpringDataJpa/products/save ☆

YouTube 한성 e-class Baekjoon Online J... 한성대학교 Instagram >> 모든 북마크

Create New Product

Product Name:

MSI 노트북

Product name은 필수 입력입니다.

Brand:

MSI

Brand는 필수 입력입니다.

Made In:

대한민국

MadeIn은 필수 입력입니다.

Price:

500

Price는 0보다 커야 합니다.

Save

localhost:8080/helloSpringDataJpa/products							
8	MacBook Air4	Apple	China	10000.00	Edit	Delete	
9	MacBook Air5	Apple	China	10000.00	Edit	Delete	
10	MacBook Pro1	Apple	China	15000.00	Edit	Delete	
11	MacBook Pro2	Apple	China	15000.00	Edit	Delete	
12	iPad Air	Apple	China	500.00	Edit	Delete	
13	iPad Pro	Apple	China	800.00	Edit	Delete	
14	소나타	Hyundai	Japan	20000.00	Edit	Delete	
15	그랜저	Hyundai	Japan	30000.00	Edit	Delete	
16	제너시스	Hyundai	Japan	50000.00	Edit	Delete	
17	에쿠스	Hyundai	Japan	60000.00	Edit	Delete	
18	Accord	Honda	Japan	25000.00	Edit	Delete	
19	sienna	Honda	Japan	40000.00	Edit	Delete	
20	Camry	Toyota	Japan	25000.00	Edit	Delete	
21	Lexus	Toyota	Japan	50000.00	Edit	Delete	
22	MSI 노트북	MSI	대한민국	500.00	Edit	Delete	

수정

Edit Product

Product ID:

Product Name:

Brand:

Made In:

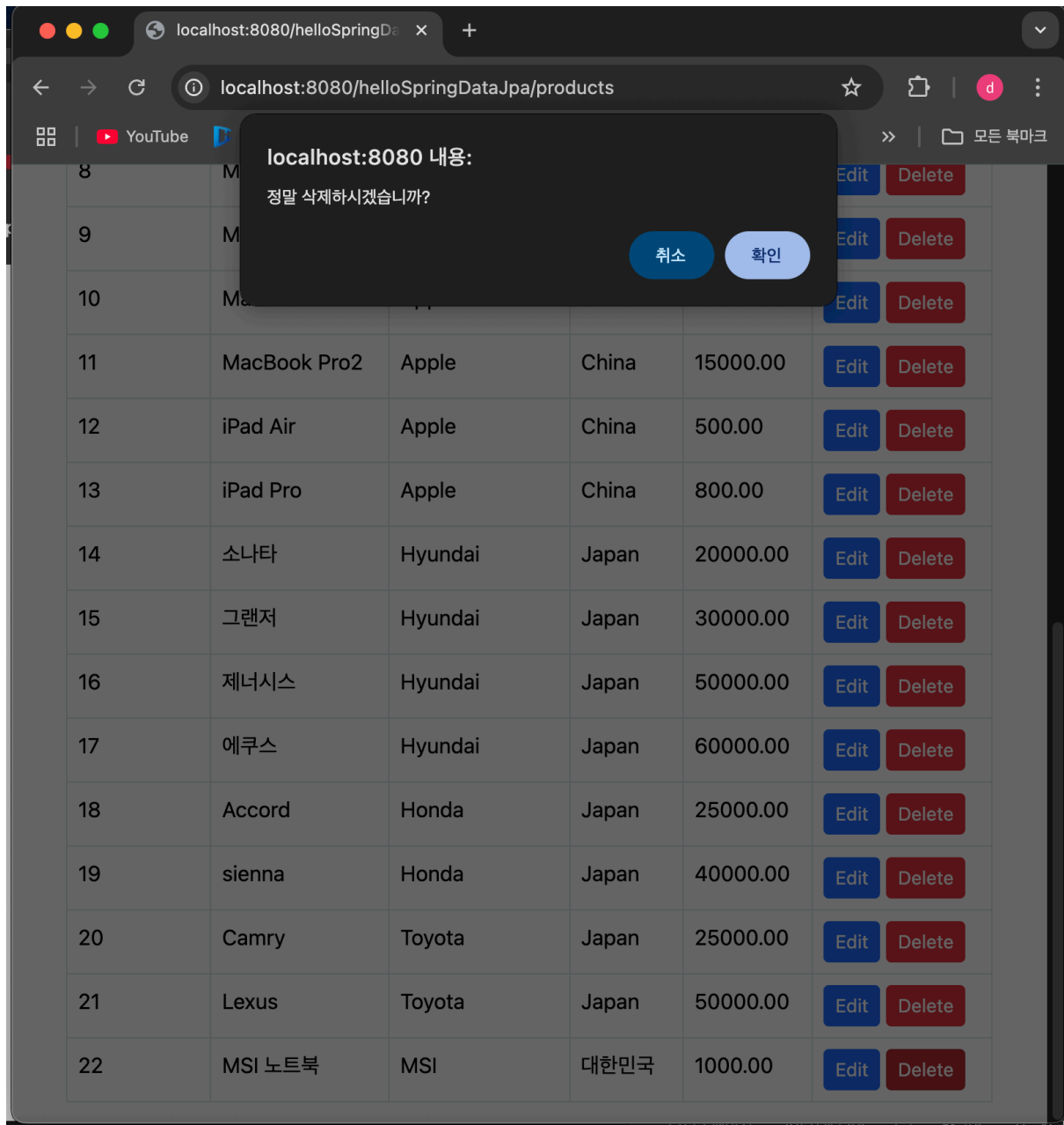
Price:

Save

수정 반영

21	Lexus	Toyota	Japan	50000.00	Edit	Delete
22	MSI 노트북	MSI	대한민국	1000.00	Edit	Delete

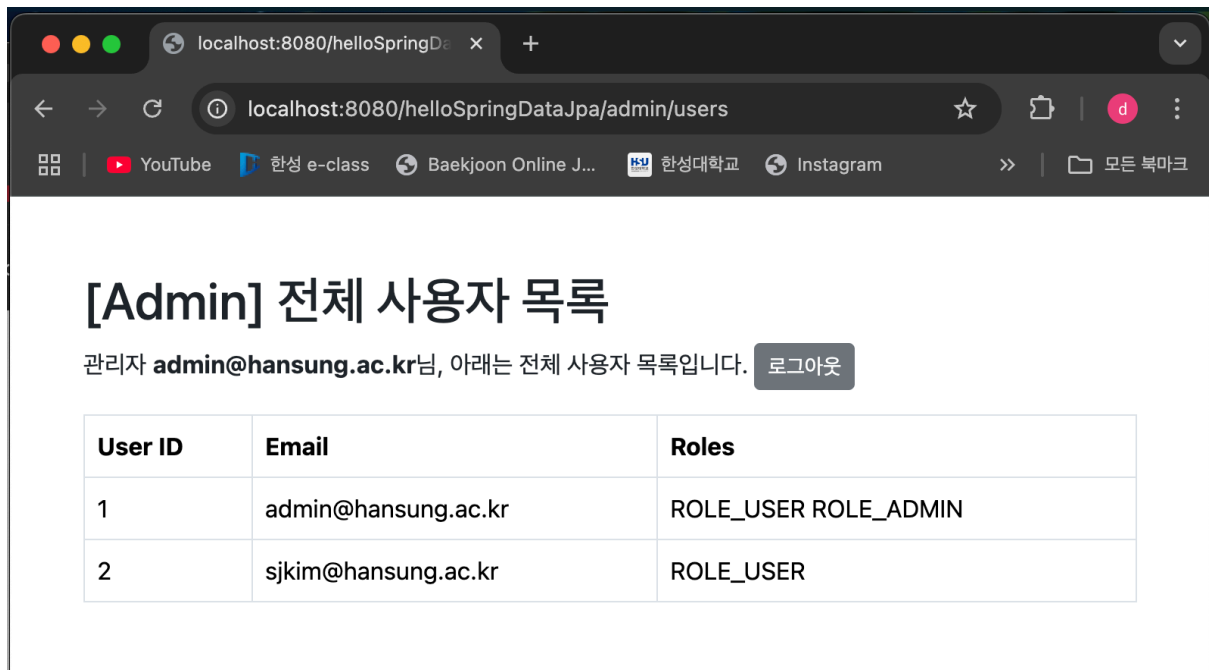
삭제



삭제 반영

20	Camry	Toyota	Japan	25000.00	<button>Edit</button> <button>Delete</button>
21	Lexus	Toyota	Japan	50000.00	<button>Edit</button> <button>Delete</button>

7. ROLE_ADMIN 사용자만 접근 가능한 관리자 페이지 구현(전체 사용자 목록 확인)



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.