[Thực hành] Validating Form Input

**Mục tiêu**

 Luyện tập validate các trường dữ liệu trong form.

**Mô tả**

Trong phần này chúng ta sẽ phát triển chức năng cho phép validate thông tin của người dùng.

Chức năng này cho phép nhập vào thông tin người dùng và hiển thị các thông báo tương ứng với giá trị của các trường dữ liệu.

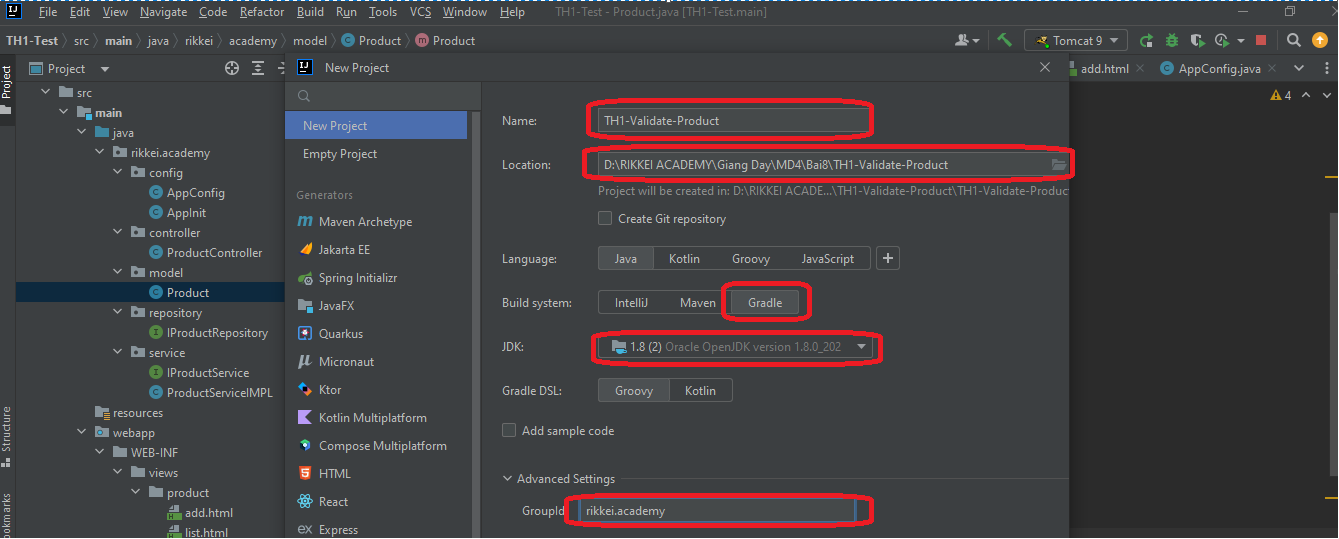
Yêu cầu đối với dữ liệu người dùng bao gồm:

* Tên bắt buộc, có độ dài tối thiểu là 2 ký tự, tối đa là 30 ký tự
* Tuổi bắt buộc, có giá trị lớn hơn 18

Để hoàn thành bài thực hành, học viên cần:

* Đưa mã nguồn lên GitHub

**Hướng dẫn**



* + Tạo package: rikkei.academy.config => Tạo 2 class: **AppInit** và **AppConfig** để config đường dẫn và database với Spring MVC
  + **AppConfig**:

package rikkei.academy.config;  
  
import org.springframework.beans.BeansException;  
import org.springframework.beans.factory.annotation.Qualifier;  
import org.springframework.context.ApplicationContext;  
import org.springframework.context.ApplicationContextAware;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.ComponentScan;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.data.jpa.repository.config.EnableJpaRepositories;  
import org.springframework.data.web.config.EnableSpringDataWebSupport;  
import org.springframework.format.FormatterRegistry;  
import org.springframework.jdbc.datasource.DriverManagerDataSource;  
import org.springframework.orm.jpa.JpaTransactionManager;  
import org.springframework.orm.jpa.JpaVendorAdapter;  
import org.springframework.orm.jpa.LocalContainerEntityManagerFactoryBean;  
import org.springframework.orm.jpa.vendor.HibernateJpaVendorAdapter;  
import org.springframework.transaction.PlatformTransactionManager;  
import org.springframework.transaction.annotation.EnableTransactionManagement;  
import org.springframework.web.servlet.config.annotation.EnableWebMvc;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;  
import org.thymeleaf.spring5.SpringTemplateEngine;  
import org.thymeleaf.spring5.templateresolver.SpringResourceTemplateResolver;  
import org.thymeleaf.spring5.view.ThymeleafViewResolver;  
import org.thymeleaf.templatemode.TemplateMode;  
import rikkei.academy.service.IProductService;  
import rikkei.academy.service.ProductServiceIMPL;  
  
import javax.persistence.EntityManager;  
import javax.persistence.EntityManagerFactory;  
import javax.sql.DataSource;  
import java.util.Properties;  
  
@Configuration  
@EnableWebMvc  
@EnableTransactionManagement  
@EnableSpringDataWebSupport  
@ComponentScan("rikkei.academy.controller")  
@EnableJpaRepositories("rikkei.academy.repository")  
public class AppConfig implements WebMvcConfigurer, ApplicationContextAware {  
 private ApplicationContext applicationContext;  
  
 @Override  
 public void setApplicationContext(ApplicationContext applicationContext) throws BeansException {  
 this.applicationContext = applicationContext;  
 }  
  
 //Cấu hình Thymleaf  
 @Bean  
 public SpringResourceTemplateResolver templateResolver() {  
 SpringResourceTemplateResolver templateResolver = new SpringResourceTemplateResolver();  
 templateResolver.setApplicationContext(applicationContext);  
 templateResolver.setPrefix("/WEB-INF/views");  
 templateResolver.setSuffix(".html");  
 templateResolver.setTemplateMode(TemplateMode.*HTML*);  
 templateResolver.setCharacterEncoding("UTF-8");  
 return templateResolver;  
 }  
  
 @Bean  
 public SpringTemplateEngine templateEngine() {  
 SpringTemplateEngine templateEngine = new SpringTemplateEngine();  
 templateEngine.setTemplateResolver(templateResolver());  
 return templateEngine;  
 }  
  
 @Bean  
 public ThymeleafViewResolver viewResolver() {  
 ThymeleafViewResolver viewResolver = new ThymeleafViewResolver();  
 viewResolver.setTemplateEngine(templateEngine());  
 viewResolver.setCharacterEncoding("UTF-8");  
 viewResolver.setContentType("UTF-8");  
 return viewResolver;  
 }  
  
 //Cấu hình JPA  
 @Bean  
 @Qualifier(value = "entityManager")  
 public EntityManager entityManager(EntityManagerFactory entityManagerFactory) {  
 return entityManagerFactory.createEntityManager();  
 }  
  
 @Bean  
 public LocalContainerEntityManagerFactoryBean entityManagerFactory() {  
 LocalContainerEntityManagerFactoryBean em = new LocalContainerEntityManagerFactoryBean();  
 em.setDataSource(dataSource());  
 em.setPackagesToScan("rikkei.academy.model");  
  
 JpaVendorAdapter vendorAdapter = new HibernateJpaVendorAdapter();  
 em.setJpaVendorAdapter(vendorAdapter);  
 em.setJpaProperties(additionalProperties());  
 return em;  
 }  
 @Bean  
 public DataSource dataSource() {  
 DriverManagerDataSource dataSource = new DriverManagerDataSource();  
 dataSource.setDriverClassName("com.mysql.cj.jdbc.Driver");  
 dataSource.setUrl("jdbc:mysql://localhost:3306/bai8\_validate");  
 dataSource.setUsername("root");  
 dataSource.setPassword("Minhtri29092014");  
 return dataSource;  
 }  
  
 @Bean  
 public PlatformTransactionManager transactionManager(EntityManagerFactory emf) {  
 JpaTransactionManager transactionManager = new JpaTransactionManager();  
 transactionManager.setEntityManagerFactory(emf);  
 return transactionManager;  
 }  
  
 public Properties additionalProperties() {  
 Properties properties = new Properties();  
 properties.setProperty("hibernate.hbm2ddl.auto", "update");  
 properties.setProperty("hibernate.dialect", "org.hibernate.dialect.MySQL5Dialect");  
 return properties;  
 }  
 }

* + **AppInit**:

package rikkei.academy.config;  
  
import org.springframework.web.filter.CharacterEncodingFilter;  
import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;  
  
import javax.servlet.Filter;  
  
public class AppInit extends AbstractAnnotationConfigDispatcherServletInitializer {  
 @Override  
 protected Class<?>[] getRootConfigClasses() {  
 return new Class[]{AppConfig.class};  
 }  
  
 @Override  
 protected Class<?>[] getServletConfigClasses() {  
 return new Class[0];  
 }  
  
 @Override  
 protected String[] getServletMappings() {  
 return new String[]{"/"};  
 }  
 @Override  
 protected Filter[] getServletFilters() {  
 CharacterEncodingFilter filter = new CharacterEncodingFilter();  
 filter.setForceEncoding(true);  
 filter.setEncoding("UTF-8");  
 return new Filter[]{filter};  
 }  
}

* + import các thư viện vào **build.gradle** như sau:

compileOnly('javax.servlet:javax.servlet-api:4.0.1')  
 implementation group: 'org.springframework', name: 'spring-core', version: '5.3.2'  
 implementation group: 'org.springframework', name: 'spring-context', version: '5.3.2'  
 implementation group: 'org.springframework', name: 'spring-beans', version: '5.3.2'  
 implementation group: 'org.springframework', name: 'spring-web', version: '5.3.2'  
 implementation group: 'org.springframework', name: 'spring-webmvc', version: '5.3.2'  
 implementation group: 'org.thymeleaf', name: 'thymeleaf-spring5', version: '3.0.11.RELEASE'  
 implementation group: 'nz.net.ultraq.thymeleaf', name: 'thymeleaf-layout-dialect', version: '2.5.2'  
 implementation group: 'org.hibernate', name: 'hibernate-core', version: '5.3.0.Final'  
 implementation group: 'org.hibernate', name: 'hibernate-entitymanager', version: '5.3.0.Final'  
 implementation group: 'org.springframework', name: 'spring-orm', version: '5.3.2'  
 implementation group: 'mysql', name: 'mysql-connector-java', version: '8.0.22'  
 implementation group: 'org.springframework.data', name: 'spring-data-jpa', version: '2.4.2'  
 //Thư vện validate  
// https://mvnrepository.com/artifact/javax.validation/validation-api  
 implementation group: 'javax.validation', name: 'validation-api', version: '2.0.1.Final'  
 implementation group: 'org.hibernate', name: 'hibernate-validator', version: '6.0.10.Final'

* Các bạn để ý các thư viện liên quan đến validate
  + Tạo package: **model** => Tạo class: **Product** trong package này

package rikkei.academy.model;  
  
import javax.persistence.\*;  
import javax.validation.constraints.\*;  
  
@Entity  
@Table(name = "products")  
public class Product {  
 @Id  
 @GeneratedValue(strategy = GenerationType.*AUTO*)  
 private Long id;  
 @NotEmpty  
 @Size(min = 2, max = 10)  
 private String name;  
 @Min(18)  
 private int price;  
  
 public Product(Long id,  
 @NotEmpty  
 @Size(min = 2, max = 10)  
 String name,  
 @Min(18)  
 int age) {  
 this.id = id;  
 this.name = name;  
 this.price = age;  
 }  
  
 public Product() {  
 }  
  
 public Long getId() {  
 return id;  
 }  
  
 public void setId(Long id) {  
 this.id = id;  
 }  
  
 public String getName() {  
 return name;  
 }  
  
 public void setName(String name) {  
 this.name = name;  
 }  
  
 public int getPrice() {  
 return price;  
 }  
  
 public void setPrice(int price) {  
 this.price = price;  
 }  
}

**@NotEmpty: không để trống**

**@Size(min = 2, max = 30): độ dài từ 2 đến 30**

**@Min(18): Gía trị điền vào phải lớn hơn 18**

* + Tạo package: **repository** trong package: **rikkei.academy** => Tạo class: **IRepository:**

package rikkei.academy.repository;  
  
import org.springframework.data.domain.Page;  
import org.springframework.data.domain.Pageable;  
import org.springframework.data.jpa.repository.Query;  
import org.springframework.data.repository.PagingAndSortingRepository;  
import org.springframework.data.repository.query.Param;  
import org.springframework.stereotype.Repository;  
import rikkei.academy.model.Product;  
@Repository  
public interface IProductRepository extends PagingAndSortingRepository<Product, Long> {  
 Page<Product> findAllByNameContaining(String name, Pageable pageable);  
 @Query("SELECT p FROM Product AS p WHERE p.name LIKE CONCAT('%',:name,'%')")  
 Page<Product> findByNameProduct(@Param("name") String name, Pageable pageable);  
}

@Query (Đoạn này ta thực hiện một câu lệnh query xuống MYSQL tương đương với hàm findAllByNameContaining ở trên.

* + Tạo package: **service** => Tạo interface: **IProductService**

package rikkei.academy.service;  
  
import org.springframework.data.domain.Page;  
import org.springframework.data.domain.Pageable;  
import org.springframework.data.repository.query.Param;  
import rikkei.academy.model.Product;  
  
public interface IProductService {  
 Page<Product> findAll(Pageable pageable);  
 void save(Product product);  
 Page<Product> findAllByNameContaining(String name, Pageable pageable);  
 Page<Product> findByNameProduct(@Param("name") String name, Pageable pageable);  
}

* + Tạo class: **ProductServiceIMPL** triển khai interface trên:

package rikkei.academy.service;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.data.domain.Page;  
import org.springframework.data.domain.Pageable;  
import org.springframework.stereotype.Service;  
import rikkei.academy.model.Product;  
import rikkei.academy.repository.IProductRepository;  
  
@Service  
public class ProductServiceIMPL implements IProductService{  
 @Autowired  
 IProductRepository productRepository;  
 @Override  
 public Page<Product> findAll(Pageable pageable) {  
 return productRepository.findAll(pageable);  
 }  
  
 @Override  
 public void save(Product product) {  
 productRepository.save(product);  
 }  
  
 @Override  
 public Page<Product> findAllByNameContaining(String name, Pageable pageable) {  
 return productRepository.findAllByNameContaining(name,pageable);  
 }  
  
 @Override  
 public Page<Product> findByNameProduct(String name, Pageable pageable) {  
 return productRepository.findByNameProduct(name,pageable);  
 }  
}

* + Thêm đoạn code sau vào: **AppConfig**:

@Bean  
IProductService productService(){  
 return new ProductServiceIMPL();  
}

* + Tạo package: controller => **ProductController**

package rikkei.academy.controller;  
  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.data.domain.Page;  
import org.springframework.data.domain.Pageable;  
import org.springframework.data.web.PageableDefault;  
import org.springframework.stereotype.Controller;  
import org.springframework.validation.BindingResult;  
  
import org.springframework.validation.annotation.Validated;  
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.RequestParam;  
import org.springframework.web.servlet.ModelAndView;  
import rikkei.academy.model.Product;  
import rikkei.academy.service.IProductService;  
  
  
  
  
@Controller  
public class ProductController {  
 @Autowired  
 IProductService productService;  
 @GetMapping(value = {"/", "/product"})  
 public ModelAndView listProduct(@PageableDefault(sort = "name",size = 3) Pageable pageable){  
 Page<Product> productPage = productService.findAll(pageable);  
 ModelAndView modelAndView = new ModelAndView("/product/list");  
 modelAndView.addObject("pageProduct",productPage);  
 return modelAndView;  
 }  
 @GetMapping("/create/product")  
 public ModelAndView showFormCreateProduct(){  
 ModelAndView modelAndView = new ModelAndView("/product/add");  
 modelAndView.addObject("createProductForm",new Product());  
 return modelAndView;  
 }  
 @PostMapping("/create/product")  
 public String createProduct(@Validated @ModelAttribute("createProductForm") Product product, BindingResult bindingResult){  
 if(bindingResult.hasFieldErrors()){  
 return "/product/add";  
 }  
 productService.save(product);  
 return "redirect:/product";  
 }  
 @GetMapping("/search")  
 public ModelAndView searchProduct(@RequestParam("search") String search, Pageable pageable){  
 Page<Product> productPage;  
 if(!search.trim().equals("")){  
 productPage = productService.findByNameProduct(search,pageable);  
 } else {  
 productPage = productService.findAll(pageable);  
 }  
 ModelAndView modelAndView = new ModelAndView("/product/list");  
 modelAndView.addObject("pageProduct",productPage);  
 return modelAndView;  
 }  
}

**@Validated**

**BindingResult bindingResult giữ kết quả xác nhận và ràng buộc Product có xảy ra lỗi không.**

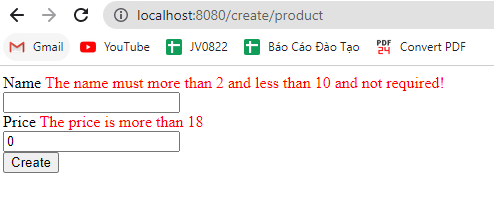
* + **Tạo tầng views: webapp/WEB-INF/product**
  + Tạo file**: add.html**

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Add Product</title>  
</head>  
<body>  
<form action="/create/product" method="post" th:object="${createProductForm}">  
 <label>Name</label>  
 <span style="color: red" th:if="${#fields.hasErrors('name')}" th:error="\*{name}">The name must more than 2 and less than 10 and not required!</span>  
 <input type="text" name="name" th:field="\*{name}"><br>  
 <label>Price</label>  
 <span style="color: red" th:if="${#fields.hasErrors('price')}" th:error="\*{price}">The price is more than 18</span>  
 <input type="text" name="age" th:field="\*{price}"><br>  
 <button type="submit">Create</button>  
</form>  
</body>  
</html>

* + Tạo file**: list.html:**

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<html lang="en">  
<head>  
 <meta charset="UTF-8">  
 <title>Add Product</title>  
</head>  
<body>  
<form action="/create/product" method="post" th:object="${createProductForm}">  
 <label>Name</label>  
 <span style="color: red" th:if="${#fields.hasErrors('name')}" th:error="\*{name}">The name must more than 2 and less than 10 and not required!</span>  
 <input type="text" name="name" th:field="\*{name}"><br>  
 <label>Price</label>  
 <span style="color: red" th:if="${#fields.hasErrors('price')}" th:error="\*{price}">The price is more than 18</span>  
 <input type="text" name="age" th:field="\*{price}"><br>  
 <button type="submit">Create</button>  
</form>  
</body>  
</html>

* + Add Tomcat và nhập liệu các trường hợp sai quan sát kết quả validate:



https://github.com/nguyendongminhtri/MD4-Bai8-TH1-Validate-Product.git

### **Hướng dẫn nộp bài:**

Up code lên github.

Paste link github vào phần nộp bài.