[Thực hành] Upload file

**Mục tiêu**

Luyện tập upload file trong ứng dụng Spring MVC. Luyện tập config Spring MVC bằng Class JAVA

**Mô tả**

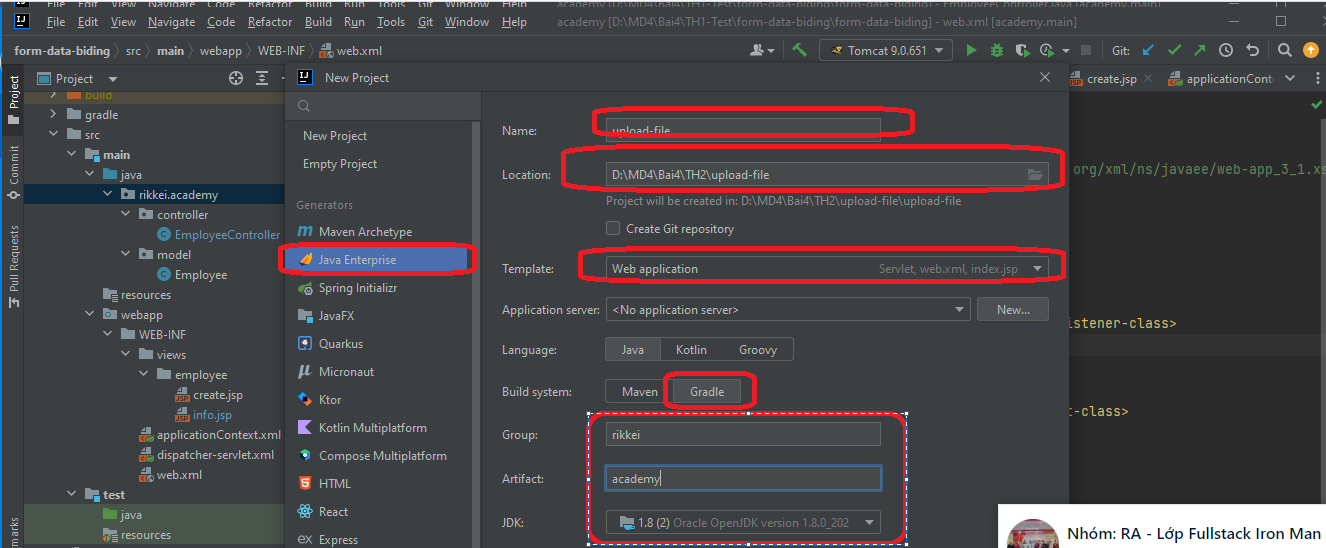
Trong phần này, chúng ta sẽ phát triển một ứng dụng gallery ảnh đơn giản.

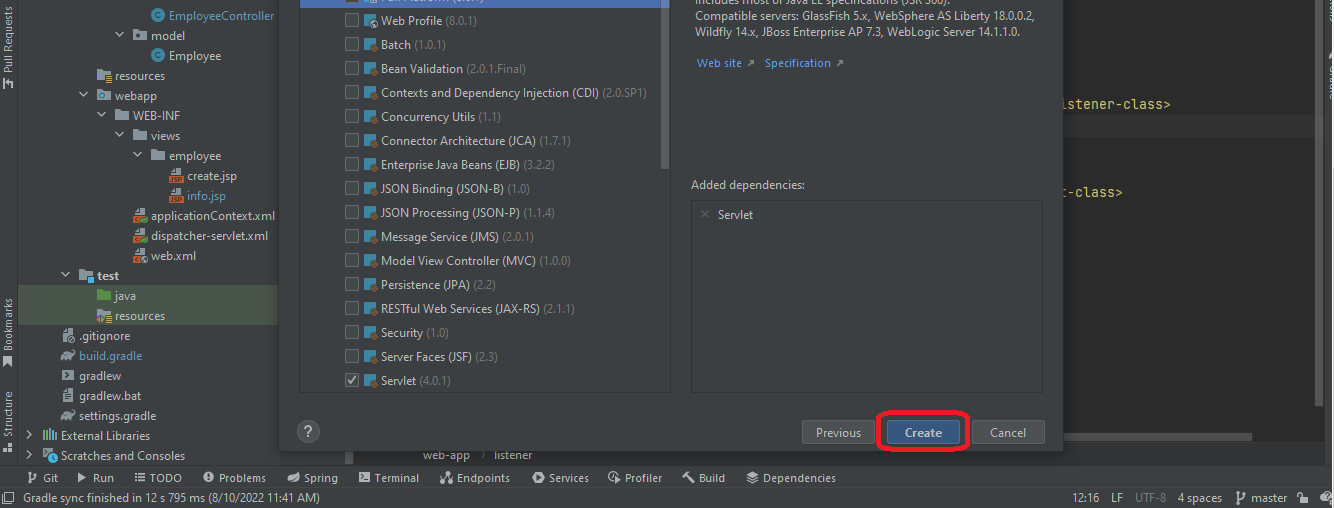
Ứng dụng có các chức năng chính:

* Upload file ảnh
* Hiển thị danh sách các ảnh đã upload
* Xem các ảnh đã upload

Hướng dẫn

* Tạo project sử dụng gradle: =>File => New=>Project => Làm đầy đủ chi như hình ảnh dưới:

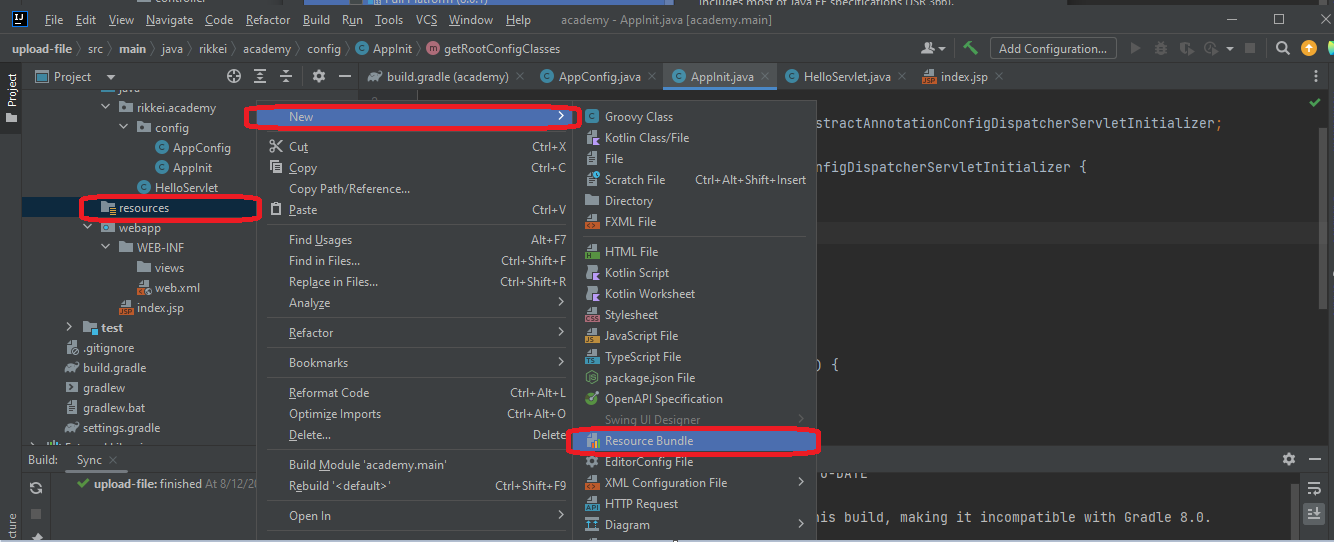




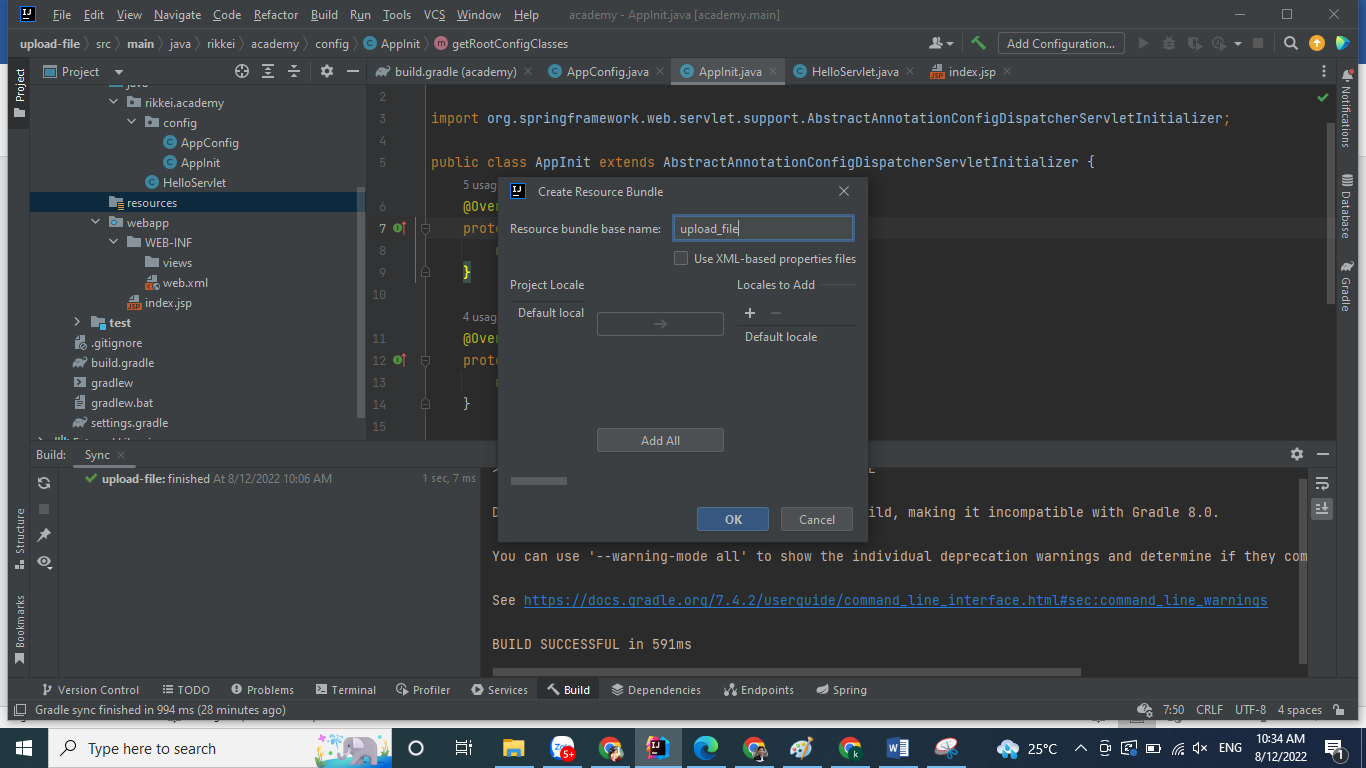
* Thêm các thư viện sau vào trong file build.gradle
* implementation group: 'org.springframework', name: 'spring-core', version: '5.3.2'  
  implementation group: 'org.springframework', name: 'spring-context', version: '5.3.2'  
  implementation group: 'org.springframework', name: 'spring-beans', version: '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: 'commons-fileupload', name: 'commons-fileupload', version: '1.4'
* Tạo packeage: config trong package: rikkei.academy => Tạo Class: AppConfig và thêm code vào như sau:
* package rikkei.academy.config;  
    
  import org.springframework.beans.BeansException;  
  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.web.servlet.config.annotation.EnableWebMvc;  
  import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;  
  import org.thymeleaf.spring5.SpringTemplateEngine;  
  import org.thymeleaf.spring5.templateresolver.SpringResourceTemplateResolver;  
  import org.thymeleaf.spring5.view.ThymeleafViewResolver;  
  import org.thymeleaf.templatemode.TemplateMode;  
    
  @Configuration  
  @EnableWebMvc  
  @ComponentScan("com.codegym.controller")  
  public class AppConfig extends WebMvcConfigurerAdapter implements 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");  
   return viewResolver;  
   }  
    
  }
* TạoClass**: AppInit** trong packe config và thêm code như sau:
* package rikkei.academy.config;  
    
  import org.springframework.web.servlet.support.AbstractAnnotationConfigDispatcherServletInitializer;  
    
  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[]{"/"};  
   }  
  }

**3.3 Tạo file upload\_file.properties trong thư mục resource như sau:**

Chuột phải vào thư mục **resource** chọn **New** => **Resource Bundle**



và đặt tên là **upload\_file => Chọn OK.**



Cấu hình file **upload\_file.properties**:

file-upload = D:\\image\\

Trong đó file-upload dùng để khai báo đường dẫn lưu trữ file trên máy tính của bạn sau khi được upload. (Trong máy tính của bạn tại ổ D tạo 1 folder có tên image)

* Thêm các config mới vào trong class **AppConfig** để cấu hình upload file

@Value("${file-upload}")  
private String fileUpload;  
  
@Override  
public void addResourceHandlers(ResourceHandlerRegistry registry) {  
 registry.addResourceHandler("/image/\*\*")  
 .addResourceLocations("file:" + fileUpload);  
  
}  
  
@Bean(name = "multipartResolver")  
public CommonsMultipartResolver getResolver() throws IOException {  
 CommonsMultipartResolver resolver = new CommonsMultipartResolver();  
 resolver.setMaxUploadSizePerFile(52428800);  
 return resolver;  
}

* Thêm annotation **@PropertySource** ở trên cùng của class**: AppConfig** để config đường dẫn tới file**: upload\_file.properties**

@PropertySource("classpath:upload\_file.properties")

Class **AppConfig** saukhi được cấu hình đầy đủ

package rikkei.academy.config;  
  
import org.springframework.beans.BeansException;  
import org.springframework.beans.factory.annotation.Value;  
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.context.annotation.PropertySource;  
import org.springframework.web.multipart.commons.CommonsMultipartResolver;  
import org.springframework.web.servlet.config.annotation.EnableWebMvc;  
import org.springframework.web.servlet.config.annotation.ResourceHandlerRegistry;  
import org.springframework.web.servlet.config.annotation.WebMvcConfigurerAdapter;  
import org.thymeleaf.spring5.SpringTemplateEngine;  
import org.thymeleaf.spring5.templateresolver.SpringResourceTemplateResolver;  
import org.thymeleaf.spring5.view.ThymeleafViewResolver;  
import org.thymeleaf.templatemode.TemplateMode;  
  
import java.io.IOException;  
  
@Configuration  
@EnableWebMvc  
@ComponentScan("rikkei.academy.controller")  
@PropertySource("classpath:upload\_file.properties")  
public class AppConfig extends WebMvcConfigurerAdapter implements 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");  
 return viewResolver;  
 }  
 //Cấu hình upload file  
 @Value("${file-upload}")  
 private String fileUpload;  
  
 @Override  
 public void addResourceHandlers(ResourceHandlerRegistry registry) {  
 registry.addResourceHandler("/image/\*\*")  
 .addResourceLocations("file:" + fileUpload);  
  
 }  
  
 @Bean(name = "multipartResolver")  
 public CommonsMultipartResolver getResolver() throws IOException {  
 CommonsMultipartResolver resolver = new CommonsMultipartResolver();  
 resolver.setMaxUploadSizePerFile(52428800);  
 return resolver;  
 }  
  
  
}

* Tạo package: model => Tạo Class: **Product** và thêm code vào như sau:
* package rikkei.academy.model;  
    
  public class Product {  
   private int id;  
    
   private String name;  
    
   private String description;  
    
   private String image;  
    
   public Product() {  
   }  
    
   public Product(int id, String name, String description, String image) {  
   this.id = id;  
   this.name = name;  
   this.description = description;  
   this.image = image;  
   }  
    
   public int getId() {  
   return id;  
   }  
    
   public void setId(int id) {  
   this.id = id;  
   }  
    
   public String getName() {  
   return name;  
   }  
    
   public void setName(String name) {  
   this.name = name;  
   }  
    
   public String getDescription() {  
   return description;  
   }  
    
   public void setDescription(String description) {  
   this.description = description;  
   }  
    
   public String getImage() {  
   return image;  
   }  
    
   public void setImage(String image) {  
   this.image = image;  
   }  
    
  }

Trong lớp **Product**, các bạn chú ý phải có các ***getter/setter*** để có thể truy cập các thuộc tính trong View.

* Tạo lớp **ProductForm** trong package: model
* package rikkei.academy.model;  
    
  import org.springframework.web.multipart.MultipartFile;  
    
  public class ProductForm {  
   private int id;  
    
   private String name;  
    
   private String description;  
    
   private MultipartFile image;  
    
   public ProductForm() {  
   }  
    
   public ProductForm(int id, String name, String description, MultipartFile image) {  
   this.id = id;  
   this.name = name;  
   this.description = description;  
   this.image = image;  
   }  
    
   public int getId() {  
   return id;  
   }  
    
   public void setId(int id) {  
   this.id = id;  
   }  
    
   public String getName() {  
   return name;  
   }  
    
   public void setName(String name) {  
   this.name = name;  
   }  
    
   public String getDescription() {  
   return description;  
   }  
    
   public void setDescription(String description) {  
   this.description = description;  
   }  
    
   public MultipartFile getImage() {  
   return image;  
   }  
    
   public void setImage(MultipartFile image) {  
   this.image = image;  
   }  
    
  }

Trong lớp **ProductForm** vẫn có những thuộc tính giống như lớp **Product** nhưng kiểu dữ liệu của thuộc tính image sẽ đổi thành ***MultipartFile*** để lưu trữ File được upload.

* Xây dựng interface **IProductService** trong package: **service**

package rikkei.academy.service;  
  
import rikkei.academy.model.Product;  
  
import java.util.List;  
  
public interface IProductService {  
 List<Product> findAll();  
  
 void save(Product product);  
  
 Product findById(int id);  
  
 void update(int id, Product product);  
  
 void remove(int id);  
  
}

* Ta sẽ đi xây dựng lớp **ProductService** thi hành các phương thức đã được khai báo nguyên mẫu trong interface **IProductService**:

package rikkei.academy.service;  
  
import rikkei.academy.model.Product;  
  
import java.util.ArrayList;  
import java.util.List;  
  
public class ProductService implements IProductService{  
 private List<Product> products = new ArrayList<>();  
  
 @Override  
 public List<Product> findAll() {  
 return products;  
 }  
  
 @Override  
 public void save(Product product) {  
 products.add(product);  
 }  
  
 @Override  
 public Product findById(int id) {  
 return products.get(id);  
 }  
  
 @Override  
 public void update(int id, Product product) {  
 for (Product p : products) {  
 if (p.getId() == id) {  
 p = product;  
 break;  
 }  
 }  
 }  
  
 @Override  
 public void remove(int id) {  
 for (int i = 0; i < products.size(); i++) {  
 if (products.get(i).getId() == id) {  
 products.remove(i);  
 break;  
 }  
 }  
 }  
  
}

* Xây dựng trang layout

File **layout.html** cũng chứa các khai báo về Style Sheet cho tất cả các trang con.

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head th:fragment="head">  
 <meta charset="utf-8" />  
 <meta http-equiv="X-UA-Compatible" content="IE=edge" />  
 <meta name="viewport" content="width=device-width, initial-scale=1" />  
 <title>Product Management</title>  
 <style type="text/css">  
 h3{background-color: blue; color: white; padding: 10px 0 10px 20px;}  
 footer{margin: 10px 0;}  
  
 #products {  
 font-family: "Trebuchet MS", Arial, Helvetica, sans-serif;  
 border-collapse: collapse;  
 width: 100%;  
 }  
 #products td, #products th {  
 border: 1px solid #ddd;  
 padding: 8px;  
 }  
 #products tr:nth-child(even){background-color: #f2f2f2;}  
 #products tr:hover {background-color: #ddd;}  
 #products th {  
 padding-top: 12px;  
 padding-bottom: 12px;  
 text-align: left;  
 background-color: #4CAF50;  
 color: white;  
 }  
 input[type=text],input[type=email]{  
 width: 100%;  
 padding: 12px 0px;  
 margin: 8px 0;  
 display: inline-block;  
 border: 1px solid #ccc;  
 border-radius: 4px;  
 box-sizing: border-box;  
 }  
 input[type=submit] {  
 width: 300px;  
 background-color: #4CAF50;  
 color: white;  
 padding: 14px 20px;  
 margin: 8px 0;  
 border: none;  
 border-radius: 4px;  
 cursor: pointer;  
 text-align: center;  
 }  
 input[type=submit]:hover {  
 background-color: #45a049;  
 }  
 #form {  
 width: 500px;  
 border-radius: 5px;  
 background-color: #f2f2f2;  
 padding: 20px;  
 }  
 .notify span{font-size: 15px; color: blue;}  
 </style>  
</head>  
<body>  
<footer th:fragment="footer">  
 &copy; 2022 Rikkei Academy  
</footer>  
</body>  
</html>

**=> Controller:** Xây dựng Class: **ProductController**

* package rikkei.academy.controller;  
    
  import org.springframework.stereotype.Controller;  
  import org.springframework.ui.Model;  
  import org.springframework.web.bind.annotation.GetMapping;  
  import org.springframework.web.bind.annotation.RequestMapping;  
  import rikkei.academy.model.Product;  
  import rikkei.academy.service.IProductService;  
  import rikkei.academy.service.ProductService;  
    
  import java.util.List;  
    
  @Controller  
  public class ProductController {  
   private final IProductService productService = new ProductService();  
    
   @GetMapping(value = {"/","/product"})  
   public String index(Model model) {  
   List<Product> products = productService.findAll();  
   model.addAttribute("products", products);  
   return "/index";  
   }  
    
  }
* Giao diện Createfile: **index.html** trong thư mục views
* <!DOCTYPE html>  
  <html xmlns:th="http://www.thymeleaf.org">  
  <head th:replace="/layout :: head">  
  </head>  
  <body>  
  <h3>Product Management</h3>  
  <p>  
   <a th:href="@{/product/create}">  
   Add new product  
   </a>  
  </p>  
  <table id = "products">  
   <thead>  
   <tr>  
   <th>Name</th>  
   <th>Description</th>  
   <th>Image</th>  
   </tr>  
   </thead>  
   <tbody>  
   <tr th:each="product : ${products}">  
   <td th:text="${product.name}"></td>  
   <td th:text="${product.description}"></td>  
   <td><img width="100" height="100" th:src="@{'/image/' + ${product.image}}" alt=""></td>  
   </tr>  
   </tbody>  
  </table>  
  <footer th:replace="/layout :: footer"></footer>  
  </body>  
  </html>

**=> Cấu hình Tomcat. Xóa: index.jsp, HelloServlet**

**=> Chạy thử ứng dụng.**

**Bước 10:**Upload file

Hiển thị form thêm mới sản phẩm qua đoạn code sau trong file: **index.html**

<p>  
 <a th:href="@{/product/create}">  
 Add new product  
 </a>  
</p>

Trong class **ProductController** chúng ta viết một phương thức create như sau:

@GetMapping("/create")  
public ModelAndView showCreateForm() {  
 ModelAndView modelAndView = new ModelAndView("/create");  
 modelAndView.addObject("productForm", new ProductForm());  
 return modelAndView;  
}

Lưu ý ở đây chúng ta sẽ add một Object là ProductForm có chứa thuộc tính image có kiểu dữ liệu là MultipartFile để xử lý việc lưu trữ file được chọn.

Tạo file: **create.html** trong thư mục **views** lưu ý ở thẻ form chúng ta thêm thuộc tính: enctype="multipart/form-data"

<!DOCTYPE html>  
<html xmlns:th="http://www.thymeleaf.org">  
<head th:replace="/layout :: head"></head>  
<body>  
<h3>Add Product</h3>  
<p>  
 <a th:href="@{/product}">  
 Back to product list  
 </a>  
</p>  
<div id="form">  
 <form th:action="@{/product/save}" th:object="${productForm}"  
 method="POST" novalidate="novalidate" enctype="multipart/form-data">  
 <input type="hidden" th:field="\*{id}"/>  
 <div>  
 <label>Name</label>  
 <input type="text" th:field="\*{name}"/>  
 </div>  
 <div>  
 <label>Description</label>  
 <input type="text" th:field="\*{description}"/>  
 </div>  
 <div>  
 <label>Image</label>  
 <input type="file" th:field="\*{image}"/>  
 </div>  
 <input type="submit" value="Save"/>  
 </form>  
</div>  
<footer th:replace="/layout :: footer"></footer>  
</body>  
</html>

Thêm phương thức POST save vào ProductController để xử lý request được gửi lên

@Value("${file-upload}")  
private String fileUpload;  
@PostMapping("/save")  
public ModelAndView saveProduct(@ModelAttribute ProductForm productForm) {  
 MultipartFile multipartFile = productForm.getImage();  
 String fileName = multipartFile.getOriginalFilename();  
 try {  
 FileCopyUtils.*copy*(productForm.getImage().getBytes(), new File(fileUpload + fileName));  
 } catch (IOException ex) {  
 ex.printStackTrace();  
 }  
 Product product = new Product(productForm.getId(), productForm.getName(),  
 productForm.getDescription(), fileName);  
 productService.save(product);  
 ModelAndView modelAndView = new ModelAndView("/create");  
 modelAndView.addObject("productForm", productForm);  
 modelAndView.addObject("message", "Created new product successfully !");  
 return modelAndView;  
}

Trong đó đoạn code này để lấy ra đường dẫn file:

@Value("${file-upload}")  
private String fileUpload;

**Bước 11:**Chạy chương trình

Các bạn chạy chương trình rồi quan sát kết quả.

Lúc này dự án của chúng ta sẽ chạy trên cổng 8080 và truy cập [http://localhost:8080](http://localhost:8080/contact)/product

Mã nguồn tham khảo:

https://github.com/nguyendongminhtri/MD4-Bai4-TH3-Upload-File.git

Hướng dẫn nộp bài:

Up code lên github.

Paste link github vào phần nộp bài.