Spring mvc:

1. Khai báo file pom dùng server jetty và build in maven:

<build>

<finalName>Lab3\_SpringMVC</finalName>

<sourceDirectory>src/main/java</sourceDirectory>

<testSourceDirectory>src/test/java</testSourceDirectory>

<resources>

<resource>

<directory>src/main/resources</directory>

</resource>

</resources>

<plugins>

<plugin>

<groupId>org.apache.maven.plugins</groupId>

<artifactId>maven-compiler-plugin</artifactId>

<version>3.1</version>

<configuration>

<source>1.8</source>

<target>1.8</target>

<encoding>UTF-8</encoding>

</configuration>

</plugin>

<plugin>

<groupId>org.eclipse.jetty</groupId>

<artifactId>jetty-maven-plugin</artifactId>

<version>9.2.15.v20160210</version>

</plugin>

</plugins>

</build>

1. Build lại project = mvn clean install

Chạy server jetty: mvn jetty:run

1. Tạo dispatcher:

<context:component-scan base-package="edu.java.spring.controller"/>

<bean class="org.springframework.web.servlet.view.InternalResourceViewResolver">

<property name="suffix" value=".jsp"></property>

</bean>

Khai báo file web.xml về dispatcher mẫu:

<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"

xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"

xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee

http://xmlns.jcp.org/xml/ns/javaee/web-app\_3\_1.xsd"

version="3.1">

<display-name>Spring MVC WEB APPlication</display-name>

<servlet>

<servlet-name>hanoi-dispatcher</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup>

</servlet>

<servlet-mapping>

<servlet-name>hanoi-dispatcher</servlet-name>

<url-pattern>/</url-pattern>

</servlet-mapping>

</web-app>

1. Dùng model view chuyển trang:

Đường dẫn sẽ /welcome và trong jsp dùng ${name} gọi biến

@RequestMapping(value="welcome",method = RequestMethod.GET)

public ModelAndView printMessage(){

ModelAndView mv = new ModelAndView();

mv.setViewName("index");// chuyển đến trang index

mv.addObject("name"," Ngoc Trinh");

return mv;

}

Dùng redirect sang trang khác:

@RequestMapping(value="site",method=RequestMethod.GET)

public String redirect(){

return "redirect:http://moom.vn";

}

1. Service lấy về text:

@RequestMapping(value="data",method=RequestMethod.GET,produces=MediaType.TEXT\_PLAIN\_VALUE)

public @ResponseBody String raw(){

return "Xin chao moi nguoi!";

}

1. Khi thêm một trường mới vào tài liệu dispatcher:

ví dụ thêm: xmlns:mvc=<http://www.springframework.org/schema/mvc>

thì trong schemalocation phải có tương ứng:

xsi:schemaLocation="http://www.springframework.org/schema/beans

http://www.springframework.org/schema/beans/spring-beans-4.2.xsd

http://www.springframework.org/schema/context

http://www.springframework.org/schema/context/spring-context-4.2.xsd

http://www.springframework.org/schema/mvc

<http://www.springframework.org/schema/mvc/spring-mvc-4.3.xsd>">

viết phải giống y hệt cả dấu /

để chỉ ra file resource chưa html

<mvc:annotation-driven/>

<mvc:resources mapping="/resources/\*\*" location="/resources/"/>

1. 1 simple về luồng, sử dụng đường dẫn chứa query (query là kiểu đường dẫn ?data=):

@Controller

public class StudentController {

@RequestMapping(value = "student/add",method = RequestMethod.GET)

public ModelAndView add(){

return new ModelAndView("student.form","command",new Student());

}

@RequestMapping(value = "student/save",method = RequestMethod.POST)

public ModelAndView save(@RequestParam(value = "name",required=false) String name,@RequestParam(value="age",required=false) int age){

ModelAndView mv = new ModelAndView();

Student student = new Student(name,age);

mv.setViewName("student.view");

mv.addObject("student",student);

return mv;

}

}

Vào trang student/add sẽ qua controller add , controller add điều hướng qua student.form.jsp, rồi từ student.form ấn submit thì điều hướng qua save controller, save controller điều hướng qua student.view hiển thị.

Trong student.form action sẽ đc chỉ ra là save, ở đây có sử dụng jstl:

<%@ page language="java" contentType="text/html; charset=ISO-8859-1"

pageEncoding="ISO-8859-1"%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<%@taglib uri="http://www.springframework.org/tags/form" prefix="form" %>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>Insert title here</title>

</head>

<body>

<h2>Please input student information</h2>

<form method="POST" action="save">

<table>

<tr>

<td>Name:</td>

<!-- <td><input name="name" type="text"></td>-->

<td><form:input path="command.name"/></td>

</tr>

<tr>

<td>Age:</td>

<!--<td><input name="age" type="text"/></td>-->

<td><form:input path="command.age" type="number"/></td>

</tr>

<tr>

<td colspan="2">

<input type="submit" value="Submit"/>

</td>

</tr>

</table>

</form>

</body>

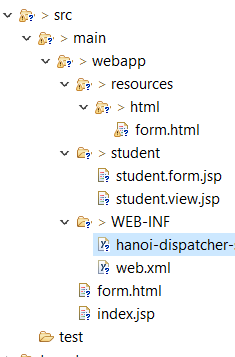
Còn trog student.view:

<body>

Name:${student.name}

<br/>

</body>



Cách đặt file.

1. Dùng BindingResult cho lỗi:

@RequestMapping(value = "student/save",method = RequestMethod.POST)

public ModelAndView save(@Valid @ModelAttribute("command") Student student,BindingResult result){

ModelAndView mv = new ModelAndView("student.form","command",student);

if(result.hasErrors()){

mv.addObject("errors",result);

return mv;

}

return mv;

}

<form:errors path="command.name">

1. Dùng message cho lỗi thẳng vào object:

@NotBlank

@Size(min=2,max=100,message="Can nhap 2->100 ki tu")

private String name;

viết thẳng mess vào obj

1. Dùng file properties riêng . config dispatcher để dùng đc file messages riêng:

<bean class="org.springframework.context.support.ResourceBundleMessageSource" id="messageSource">

<property name="basename" value="messages"></property>

</bean>

Tạo file:



Bên trong:

#{ValidationClass}.{modelObjectName}.{field}

Size.command.name=Canh bao:2->100

Range.command.age=Canh bao:tuoi khong dung!

NotBlank=Khong de trong.

Tương ứng với student:

public class Student {

private int id;

@NotBlank

@Size(min=2,max=100)

private String name;

@Range(min=1,max=150)

private int age;

cách hiển thị trong form như 8) <form:errors path="command.name"></form:errors>

1. AppContextLoaderListener mở đầu ứng dụng:

@Override

public void contextDestroyed(ServletContextEvent event) {

System.out.println("Huy");

}

@Override

public void contextInitialized(ServletContextEvent event) {

System.out.println("Khoi tao");

}

Khai báo trong web.xml:

<listener-class>

edu.java.spring.AppContextLoaderListener

</listener-class>

</listener>

1. @autowire trong spring, nhớ khai báo:

<bean class="org.springframework.beans.factory.annotation.AutowiredAnnotationBeanPostProcessor" />

Chỉ cần autowire là tư động hiểu không cần khai báo bean có tên tương ứng trừ bean của lớp muốn thay thuộc tính

Tức là đoạn code dưới chỉ cần khai báo duy nhất dataSource trong bean:

package edu.java.spring.dao.impl;

@Component

public class StudentDAOImpl implements StudentDAO,DisposableBean {

private DataSource dataSource;

private JdbcTemplate jdbcTemplate;

public DataSource getDataSource() {

jdbcTemplate = new JdbcTemplate(dataSource);

return dataSource;

}

@Autowired

public void setDataSource(DataSource dataSource) {

this.dataSource = dataSource;

jdbcTemplate = new JdbcTemplate(dataSource);

}

Dùng từ khóa @Component để spring xác định thành phần. khi đó trong dispatcher phải khai báo:

<context:component-scan base-package="edu.java.spring.dao.impl"/>

Và trong student controller có autowired studentDAO nhưng trong dispatcher ko cần bean của student dao

@Controller

public class StudentController {

@Autowired

private StudentDAO studentDAO;

1. Đường dẫn có giá trị trong urllink:

@RequestMapping(value = "/student/delete/{id}")

public String delete(@PathVariable(name = "id")int id){

studentDAO.delete(id);

return "redirect:/student/list";

}

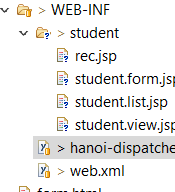
1. Quản lí form việc new ModelAndView thì:

ModelAndView mv = new ModelAndView("student/student.form","command",student);

Với dispatcher:

<property name="prefix" value="/WEB-INF/"></property>

Vậy:

F

1. Dùng hidden form

<form method="POST" action="/student/save" commandName="command">

<form:hidden path="command.id" />

Quan trọng đoạn action căn chỉnh sang đường dẫn khác.

1. Forward sang đg link khác:

ModelAndView mv = new ModelAndView("forward:/student/save","command",student);

return mv;

1. Dùng bộ thư viện Jackson để tương tác service:

<dependency>

<groupId>com.fasterxml.jackson.core</groupId>

<artifactId>jackson-databind</artifactId>

<version>2.9.4</version>

</dependency>

@RequestMapping(value="/student/json/{id}",method = RequestMethod.GET)

public @ResponseBody Student model(@RequestParam(name="id")int id){

return studentDAO.get(id);

}

1. Cách viết 1 service:

@RequestMapping(value="/student/json/{id}",method = RequestMethod.GET)

public @ResponseBody Student model(@PathVariable(name="id")int id){

return studentDAO.get(id);

}

Sau đó dùng js lấy json:

<script>

function view(id) {

var xmlHttp = new XMLHttpRequest();

xmlHttp.open("GET","json/"+id,true);

xmlHttp.onload = function(){

if(this.status != 200) return;

console.log(this.responseText);

var student = JSON.parse(this.responseText);

document.getElementById('content').innerHTML = 'Name: ' + student.name;

var dialog = document.getElementById('viewStudent');

dialog.show();

};

xmlHttp.send();

}

</script>

1. Dùng tiles viết template html:

<dependency>

<groupId>org.apache.tiles</groupId>

<artifactId>tiles-extras</artifactId>

<version>3.0.5</version>

</dependency>

Trong dispatcher config bean để dùng:

<bean id="tilesConfigurer"

class="org.springframework.web.servlet.view.tiles3.TilesConfigurer">

<property name="definitions">

<list>

<value>/tiles/definitions.xml</value>

</list>

</property>

</bean>

Tạo file definitions.xml:

<?xml version="1.0" encoding="UTF-8"?>

<!DOCTYPE tiles-definitions PUBLIC

"-//Apache Software Foundation//DTD Tiles Configuration 3.0//EN"

"http://tiles.apache.org/dtds/tiles-config\_3\_0.dtd">

<tiles-definitions>

<definition name="studentTemplate" template="/student/layout.jsp">

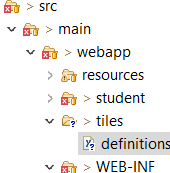
<put-attribute name="menu" value="/student/menu.jsp"/>

<put-attribute name="body" value=""/>

<put-attribute name="footer" value="/student/footer.jsp"/>

</definition>

</tiles-definitions>



Tạo các file jsp tương ứng với menu,body,footer

Sau đó nhét vào file jsp như một template,nhét vào body :

<tiles:insertDefinition name="studentTemplate">

<tiles:putAttribute name="body">

<h2>Listofstudent</h2>

<body>

</tiles:putAttribute>

</tiles:insertDefinition>

1. Up ảnh :

<dependency>

<groupId>commons-fileupload</groupId>

<artifactId>commons-fileupload</artifactId>

<version>1.3.3</version>

</dependency>

Tạo nút up chú ý kiểu enctype là multipart fom data:

<c:if test="${id != null}">

<h1>Upload</h1>

<form method="post" action="/student/avatar/save" enctype="multipart/form-data">

<input type="hidden" name="id" value="${id}"/>

<input type="file" name="file"/>

<input type="submit" value="Upload"/>

</form>

</c:if>

Config trg dispatcher để dùng multipartResolver :

<bean id="multipartResolver" class="org.springframework.web.multipart.commons.CommonsMultipartResolver">

<property name="maxUploadSize" value="100"></property>

</bean>

Trong controller đầu tiên khởi tạo file ảnh, sau đó lấy path rồi truyền dữ liệu vào file ảnh:

@RequestMapping(value="/student/avatar/save",method = RequestMethod.POST)

public String handleFormUpload(@RequestParam("file") MultipartFile file,@RequestParam(value="id")int id,HttpServletRequest request){

if(file.isEmpty()) return "/student.error";

ServletContext ctx = request.getSession().getServletContext();

Path avatarPath = getImagePath(request, id);

try {

Files.write(avatarPath, file.getBytes(),StandardOpenOption.CREATE);

} catch (IOException e) {

e.printStackTrace();

}

return "redirect:/student/list";

}

private Path getImagePath(HttpServletRequest request,int id){

ServletContext ctx = request.getSession().getServletContext();

String diskPath=ctx.getRealPath("/");

File folder = new File(diskPath + File.separator + "avatar" + File.separator);

folder.mkdirs();

return new File(folder,String.valueOf(id)+".jpg").toPath();

}

1. Lấy ảnh:

@RequestMapping(value="/student/avatar/{id}")

public ResponseEntity<byte[]> getImage(@PathVariable(value = "id")Integer id,HttpServletRequest request){

ByteArrayOutputStream byteOutput = new ByteArrayOutputStream();

if(id != null){

Path avatarPath = getImagePath(request, id);

if(Files.exists(avatarPath))

try {

byteOutput.write(Files.readAllBytes(avatarPath));

} catch (IOException e) {

e.printStackTrace();

}

}

HttpHeaders headers = new HttpHeaders();

headers.setContentType(MediaType.IMAGE\_JPEG);

return new ResponseEntity<byte[]>(byteOutput.toByteArray(),headers,HttpStatus.CREATED);

}

ở đây ko dùng thẳng files.readallbyte vào responseEntity mà lại dùng thông qua byteOutput để xử lí exception về io nếu có xảy ra.

1. Xử lí lỗi nhảy qua trang lỗi, config dispatcher và tạo 1 trang lỗi, khi có exception nó sẽ nhảy qua dispatcher này:

<bean class="org.springframework.web.servlet.handler.SimpleMappingExceptionResolver">

<property name="exceptionMappings">

<props>

<prop key="java.lang.Exception">student/student.error</prop>

</props>

</property>

</bean>