第3天: SpringMVC

一、目标

1. 实现SpringMVC文件上传
2. 掌握SpringMVC统一异常处理
3. 了解SpringMVC的拦截器

二、文件上传

2.1 传统方式

2.1.1 页面要求

1. 需要设置method为post
2. 需要设置enctype为multipart/form-data
3. 需要设置文件

- index.jsp

```
<h3>文件上传</h3>
<form action="/upload.do"
      method="post" <%-- 1. 需要设置method为post提交 --%>
      enctype="multipart/form-data" <%-- 2. 需要设置enctype为multipart/form-data --%>>
    <input name="name" type="text" value="Jason" />
    <input name="file" type="file" /> <%-- 3. 需要设置文件 --%>
    <input type="submit"/>
</form>
```

2.1.2 后端代码

添加依赖: pom.xml

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

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <parent>
    <artifactId>mvc2</artifactId>
    <groupId>com.itheima</groupId>
    <version>1.0-SNAPSHOT</version>
  </parent>
  <modelVersion>4.0.0</modelVersion>
  <artifactId>mvc-day02-controller</artifactId>
  <packaging>war</packaging>
  <dependencies>
    <dependency>
```

```
      <version>5.1.7.RELEASE</version>
    </dependency>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>servlet-api</artifactId>
      <version>2.5</version>
    </dependency>
    <!-- Json转换由jackson完成 -->
    <dependency>
      <groupId>com.fasterxml.jackson.core</groupId>
      <artifactId>jackson-databind</artifactId>
      <version>2.9.9</version>
    </dependency>
    <!-- 文件上传的组件支持 -->
    <dependency>
      <groupId>commons-fileupload</groupId>
      <artifactId>commons-fileupload</artifactId>
      <version>1.4</version>
    </dependency>
  </dependencies>
</project>
```

处理请求: UploadController

```
package com.itheima.controller;

import org.apache.commons.fileupload.FileItem;
import org.apache.commons.fileupload.disk.DiskFileItemFactory;
import org.apache.commons.fileupload.servlet.ServletFileUpload;
import org.springframework.stereotype.Controller;
import org.springframework.web.bind.annotation.RequestMapping;

import javax.servlet.http.HttpServletRequest;
import java.io.File;
import java.util.List;
import java.util.UUID;

/**
 * 文件上传案例代码.
 *
 * @author : Jason.lee
 * @version : 1.0
 */
@Controller
public class UploadController {

    /**
     * 传统文件上传
     *  需要使用commons-fileupload组件的原生API解析请求并处理文件存储
     */
    @RequestMapping("upload")
    public String hello(HttpServletRequest request) throws Exception {
        // 1. 指定上传目录
        String path =
request.getSession().getServletContext().getRealPath("/upload");
```

```
            file.mkdir();
        }
        // 2. 创建组件工具
        DiskFileItemFactory factory = new DiskFileItemFactory();
        ServletFileUpload upload = new ServletFileUpload(factory);
        // 3. 解析请求数据
        List<FileItem> items = upload.parseRequest(request);
        for (FileItem item : items) {
            // 判断是否是普通表单字段
            //
            if (item.isFormField()) {
                System.out.println("普通字段: " + item.getString("UTF-8"));
            } else {
                // 保存文件
                String id = UUID.randomUUID().toString(); // 保证文件名不重复
                item.write(new File(path + "/" + id + item.getName()));
            }
        }
        return "success";
    }
}
```

2.2 SpringMVC

2.2.1 页面

- index.jsp

```
<h3>文件上传</h3>
<form action="/upload.do"
      method="post" <%-- 1. 需要设置method为post提交 --%>
      enctype="multipart/form-data" <%-- 2. 需要设置enctype为multipart/form-data --%>
      >
    <input name="name" type="text" value="Jason" />
    <%-- 文件名name需要严格与接收参数名称一致 --%>
    <input name="img" type="file" /> <%-- 3. 需要设置文件 --%>
    <input type="submit"/>
</form>
```

2.2.2 配置

- pom.xml

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

<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
http://maven.apache.org/xsd/maven-4.0.0.xsd">
  <parent>
    <artifactId>mvc2</artifactId>
    <groupId>com.itheima</groupId>
    <version>1.0-SNAPSHOT</version>
```

```
  <packaging>war</packaging>
  <dependencies>
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-webmvc</artifactId>
      <version>5.1.7.RELEASE</version>
    </dependency>
    <dependency>
      <groupId>javax.servlet</groupId>
      <artifactId>servlet-api</artifactId>
      <version>2.5</version>
    </dependency>
    <!-- Json转换由jackson完成 -->
    <dependency>
      <groupId>com.fasterxml.jackson.core</groupId>
      <artifactId>jackson-databind</artifactId>
      <version>2.9.9</version>
    </dependency>
    <!-- 文件上传的组件支持 -->
    <dependency>
      <groupId>commons-fileupload</groupId>
      <artifactId>commons-fileupload</artifactId>
      <version>1.4</version>
    </dependency>
  </dependencies>
</project>
```

- springMVC.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvn="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="http://www.springframework.org/schema/beans
       http://www.springframework.org/schema/beans/spring-beans.xsd
http://www.springframework.org/schema/context
https://www.springframework.org/schema/context/spring-context.xsd
http://www.springframework.org/schema/mvc
https://www.springframework.org/schema/mvc/spring-mvc.xsd">

    <!-- 1. 开启注解扫描 -->
    <context:component-scan base-package="com.itheima.controller"/>
    <!-- 2. 开启注解驱动 (配置了处理器映射器, 处理器适配器等) -->
    <mvn:annotation-driven/>
    <!-- 3. 配置视图解析器 -->
    <bean
class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/pages/"/>
        <property name="suffix" value=".jsp"/>
    </bean>
    <!-- 4. 配置文件解析器-->
    <!-- id: 必须固定为 multipartResolver -->
    <bean id="multipartResolver"
```

```
    </bean>
</beans>
```

2.2.3 代码

- com.itheima.controller.UploadController

```
/**
 * SpringMVC文件上传
 *  基于commons-fileupload组件处理的请求
 */
@RequestMapping("upload")
public String hello(HttpServletRequest request, String name, MultipartFile
img) throws Exception {
    // 1. 指定上传目录
    String path =
request.getSession().getServletContext().getRealPath("/upload");
    File file = new File(path);
    if (!file.exists()) {
        file.mkdir();
    }
    // 2. 处理上传数据
    System.out.println(name);
    String id = UUID.randomUUID().toString(); // 保证文件名不重复
    String filename = img.getOriginalFilename(); // 获取文件名的API不是getName
    // 保存文件的API是transferTo (转移到指定位置)
    img.transferTo(new File(path + "/" + id + filename));
    return "success";
}
```

三、统一异常处理

3.1 概念

- 在项目开发过程中, 全局的捕捉异常进行统一的处理

3.1.1 背景

> 在三层架构中, 如果Dao持久层有异常可以抛出到调用方(Service), Service有异常可以抛出到Controller, 但是Controller有异常, 不建议抛出到客户端(用户), 因为用户看不懂异常, 无法进行处理会导致用户体验非常差。

3.1.2 处理

方案一:

- 在Controller层使用try catch捕捉异常

```
@Controller
public class UserController {
```

```
    UserService userService;

    @RequestMapping("list")
    public String list(){
        try{
            userService.findAll();
        }catch(Exception e){
            // 异常处理
            return "error";
        }
        return "success";
    }
}
```

方案二:

- 在servlet过滤器(Filter)调用代码上捕捉异常

```
public class ExceptionFilter implements Filter {
    public void doFilter(ServletRequest request, ServletResponse response,
FilterChain chain){
        try{
            chain.doFilter(request, response);
        }catch(Exception e){
            // 异常处理
        }
    }
}
```

方案三:

- 使用SpringMVC提供的异常处理器支持

3.2 案例演示

- com.itheima.ex.UserController

```
package com.itheima.ex;

import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.RequestMapping;

/**
 * 模拟查询所有用户异常.
 *
 * @author : Jason.lee
 * @version : 1.0
 */
@Controller
public class UserController {


```

```
        model.addAttribute("msg", "用户列表");
        // 希望抛出异常进行友好提示: 返回error.jsp页面
        int i = 1/0;
        return "success";
    }
}
```

- com.itheima.ex.CustomExceptionHandler

```
package com.itheima.ex;

import org.springframework.stereotype.Component;
import org.springframework.web.bind.annotation.ExceptionHandler;
import org.springframework.web.servlet.HandlerExceptionResolver;
import org.springframework.web.servlet.ModelAndView;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 * 自定义异常处理器.
 *
 * @author : Jason.lee
 * @version : 1.0
 */
@Component
public class CustomExceptionHandler implements HandlerExceptionResolver {


    /**
     * 发现异常将调用该方法处理
     * @param httpServletRequest 请求
     * @param httpServletResponse 响应
     * @param o 处理的方法
     * @param e 异常
     * @return 返回视图
     */
    @Override
    public ModelAndView resolveException(HttpServletRequest
httpServletRequest,
                                         HttpServletResponse
httpServletResponse,
                                         Object o, Exception e) {
        ModelAndView mv = new ModelAndView();
        mv.setViewName("error");
        mv.addObject("msg", e.getMessage());
        return mv;
    }
}
```

四、SpringMVC拦截器

4.1.1 什么是拦截器

> springmvc框架中的拦截器，相当于web阶段学习的过滤器（filter），可以实现前置增强和后置增强功能。在springmvc框架中，拦截器可以对处理器方法执行预处理（前置增强），和执行后处理（后置增强）。

4.1.2 拦截器的作用

- Spring MVC 的处理器拦截器类似于 Servlet 开发中的过滤器 Filter
- 用于对处理器进行**预处理** 和 **后处理。**
- 用户可以自己定义一些拦截器来实现特定的功能。

4.1.3 拦截器与过滤器

组件名称	组件来源	作用范围
过滤器(Filter)	WEB项目均可使用 (源于Servlet规范)	可以在web.xml中配置/*过滤所有资源
拦截器(Interceptor)	SpringMVC项目中 (SpringMVC特有的组件)	只拦截Controller的方法对应的资源

4.2 案例

4.2.1 编写拦截器

- com.itheima.interceptor.CustomHandlerInterceptor

```
1  package com.itheima.interceptor;
2
3  import org.springframework.web.servlet.HandlerInterceptor;
4  import org.springframework.web.servlet.ModelAndView;
5
6  import javax.servlet.http.HttpServletRequest;
7  import javax.servlet.http.HttpServletResponse;
8
9  /**
10  * 自定义拦截器.
11  *
12  * @author : Jason.lee
13  * @version : 1.0
14  */
15 public class CustomHandlerInterceptor implements HandlerInterceptor {
16
17     /**
18      * Controller方法调用钱执行的方法 【前置通知】
19      * @return 是否执行处理器方法
20      */
21     @Override
22     public boolean preHandle(HttpServletRequest request,
   HttpServletResponse response, Object handler) throws Exception {
23         System.out.println("1. 先执行preHandle方法..");
24         return false;
```

```
     * Controller方法调用成功后执行的方法 【后置通知】
     *  需要preHandle方法返回true
     *  并且Controller方法没有异常才执行
     */
    @Override
    public void postHandle(HttpServletRequest request, HttpServletResponse response, Object handler, ModelAndView modelAndView) throws Exception {
        System.out.println("3. 后执行postHandle方法..");
    }

    /**
     * Controller方法执行完成后执行的方法 【最终通知】
     *  需要preHandle方法返回true
     *  不论Controller方法是否异常都会执行
     */
    @Override
    public void afterCompletion(HttpServletRequest request, HttpServletResponse response, Object handler, Exception ex) throws Exception {
        System.out.println("4. 最终执行afterCompletion方法..");
    }
}
```

4.2.2 配置拦截器

- springMVC.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xmlns:context="http://www.springframework.org/schema/context"
       xmlns:mvc="http://www.springframework.org/schema/mvc"
       xsi:schemaLocation="http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context.xsd http://www.springframework.org/schema/mvc https://www.springframework.org/schema/mvc/spring-mvc.xsd">


    <context:component-scan base-package="com.itheima.ex"/>

    <mvc:annotation-driven/>

    <bean id="viewResolver" class="org.springframework.web.servlet.view.InternalResourceViewResolver">
        <property name="prefix" value="/WEB-INF/pages/"/>
        <property name="suffix" value=".jsp"/>
    </bean>

    <!--
        <mvc:interceptors>: 配置拦截器链(多个)
            <mvc:interceptor>: 定义拦截器
                <mvc:mapping path="/**"/>: 拦截的资源路径-@RequestMapping
                <bean: 拦截器定义
```

```
25      <mvc:interceptors>
26          <mvc:interceptor>
27              <!-- path: 拦截的资源路径 -->
28              <mvc:mapping path="/**"/>
29              <!-- class: 拦截器类路径 -->
30              <bean
    class="com.itheima.interceptor.CustomHandlerInterceptor"/>
31          </mvc:interceptor>
32      </mvc:interceptors>
33  </beans>
```

4.2.3 执行流程