

基于 Java EE 的企业应用系统设计

Spring MVC

王晓东

wangxiaodong@ouc.edu.cn

中国海洋大学

June 14, 2017



References

1. Spring MVC: A Tutorial (Second Edition) (ISBN 9781771970310)



大纲

Spring 文件上传

示例：Apache Commons FileUpload 上传文件

Spring 文件下载

示例：隐藏资源

示例：防止交叉引用

Spring 用户登录

监听器



接下来…

Spring 文件上传

示例：Apache Commons FileUpload 上传文件

Spring 文件下载

示例：隐藏资源

示例：防止交叉引用

Spring 用户登录

监听器



文件上传

❖ Spring MVC 中处理文件上传有两种方法

1. 使用 Apache Commons FileUpload 元件；
2. 利用 Servlet 3.0 及其更高版本的内置支持。



文件上传表单

```
<form action="action" enctype="multipart/form-data" method="post">  
  Select a file <input type="file" name="fieldName"/>  
  <input type="submit" value="Upload"/>  
</form>
```

enctype 说明

enctype 属性规定在发送到服务器之前应该如何对表单数据进行编码。默认地，表单数据会编码为“application/x-www-form-urlencoded”。即是在发送到服务器之前，所有字符都会进行编码（空格转换为“+”加号，特殊符号转换为 ASCII HEX 值）。

为了上传文件，必须将 HTML FORM 的 enctype 属性值设为 multipart/form-data。



多文件上传表单

在 HTML 5 之前，如果想要上传多个文件，必须使用多个文件 input 元素。

HTML 5 中，通过在 input 元素中引入 multiple 属性，使得多个文件的上传变得更加简单。

编写以下任意一行代码，便可生成一个按钮供选择多个文件

```
<input type="file" name="fieldName" multiple/>  
<input type="file" name="fieldName" multiple="multiple"/>  
<input type="file" name="fieldName" multiple=""/>
```



MultipartFile 接口

上传到 Spring MVC 应用程序中的文件会被包在一个

`org.springframework.web.multipart.MultipartFile` 对象中。唯一的任务就是用类型为 `MultipartFile` 的属性编写一个 `domain` 类。

❖ MultipartFile 接口的方法

- ▶ `byte[] getBytes()` 以字节数组的形式返回文件的内容。
- ▶ `String getContentType()` 返回文件的内容类型。
- ▶ `InputStream getInputStream()` 返回一个 `InputStream`，从中读取文件的内容。
- ▶ `String getName()` 以多部分的形式返回参数的名称。
- ▶ `String getOriginalFilename()` 返回客户端本地驱动器中的初始文件名。
- ▶ `long getSize()` 以字节为单位，返回文件的大小。
- ▶ `boolean isEmpty()` 表示被上传的文件是否为空。
- ▶ `void transferTo(File destination)` 将上传的文件保存到目标目录下。



用 Commons FileUpload 上传文件

对版本低于 Servlet 3.0 的容器，需要 Apache Commons FileUpload 元件和 Apache Commons IO 元件完成文件的上传。

下载地址

- ▶ <http://commons.apache.org/proper/commons-fileupload/>
- ▶ <http://commons.apache.org/proper/commons-io/>

此外，还需要在 Spring MVC 配置文件中定义 multipartResolver。

```
<bean id="multipartResolver"
class="org.springframework.web.multipart.commons.CommonsMultipartResolver">
<property name="maxUploadSize" value="2000000" />
</bean>
```



示例：Apache Commons FileUpload 上传文件

接下来...

Spring 文件上传

示例：Apache Commons FileUpload 上传文件

Spring 文件下载

示例：隐藏资源

示例：防止交叉引用

Spring 用户登录

监听器



示例: Apache Commons FileUpload 上传文件

Domain 类

Product.java

```
public class Product implements Serializable {
    private static final long serialVersionUID = 74458L;

    // JSR303 Bean Validation
    @NotNull
    @Size(min=1, max=10)
    private String name;

    private String description;
    private Float price;
    private List<MultipartFile> images;

    public List<MultipartFile> getImages() {
        return images;
    }

    public void setImages(List<MultipartFile> images) {
        this.images = images;
    }

    // other setter and getter methods.
}
```



示例: Apache Commons FileUpload 上传文件

ProductController 类

ProductController.java

```
@Controller
public class ProductController {
    ... // input-product

    @RequestMapping(value = "/save-product")
    public String saveProduct(HttpServletRequest servletRequest, BindingResult bindingResult,
        @ModelAttribute Product product, Model model) {
        List<MultipartFile> files = product.getImages();
        List<String> fileNames = new ArrayList<String>();
        if (null != files && files.size() > 0) {
            for (MultipartFile multipartFile : files) {
                String fileName = multipartFile.getOriginalFilename();
                fileNames.add(fileName);
                File imageFile = new File(servletRequest.getServletContext()
                    .getRealPath("/image"), fileName);
                try {
                    multipartFile.transferTo(imageFile);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        }
        model.addAttribute("product", product);
        return "ProductDetails";
    }
}
```



示例：Apache Commons FileUpload 上传文件

配置文件

❖ 注意在配置文件中添加 multipartResolver bean

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

注：利用 multipartResolverbean 的 maxUploadSize 属性可以设置能够接受的最大上传文件容量。



示例: [Apache Commons FileUpload](#) 上传文件

JSP 页面

ProductForm.jsp

```
<form:form commandName="product" action="product_save" method="post"
  enctype="multipart/form-data">
  ...
  <label for="name">Product Name: </label>
  <form:input id="name" path="name" cssErrorClass="error" />
  <form:errors path="name" cssClass="error" />
  ...

  <label for="image">Product Image: </label>
  <input type="file" name="images[0]" />

  <p id="buttons">
    <input id="reset" type="reset" tabindex="4">
    <input id="submit" type="submit" tabindex="5" value="Add Product">
    ...
  </form:form>
```



示例: Apache Commons FileUpload 上传文件

JSP 页面

ProductDetails.jsp

```
<h5>Details:</h5>
Product Name: \${product.name}<br/>
Description: \${product.description}<br/>
Price: \${product.price}

<p>Following files are uploaded successfully.</p>

<c:forEach items="\${product.images}" var="image">
  <li>\${image.originalFilename}
    
    \${image.originalFilename}"/>
  </li>
</c:forEach>
```



完整代码参考

springmvc-fileupload-01



示例：Apache Commons FileUpload 上传文件

用 Servlet3 及其更高版本上传文件



请自行搜索学习。



接下来…

Spring 文件上传

示例：Apache Commons FileUpload 上传文件

Spring 文件下载

示例：隐藏资源

示例：防止交叉引用

Spring 用户登录

监听器



文件下载

为了将像文件资源发送到浏览器，需要在控制器中完成以下工作：

1. 对请求处理方法使用 void 返回类型，并在方法中添加 HttpServletResponse 参数。
2. 将响应的内容类型设为文件的内容类型。Content-Type 标题在某个实体的 body 中定义数据的类型，并包含媒体类型和子类型标识符。如果不清楚内容类型，并且希望浏览器始终显示 Save As（另存为）对话框，则将它设为 APPLICATION/OCTET-STREAM。
3. 添加一个名为 Content-Disposition 的 HTTP 响应标题，并赋值 attachment; filename=fileName，这里的 fileName 是默认文件名，应该出现在 File Download（文件下载）对话框中，它通常与文件同名。



文件下载

❖ 将文件发送到浏览器的代码

```
FileInputStream fis = new FileInputStream(file);  
BufferedInputStream bis = new BufferedInputStream(fis);  
byte[] bytes = new byte[bis.available()];  
response.setContentType(contentType);  
OutputStream os = response.getOutputStream();  
bis.read(bytes);  
os.write(bytes);
```



示例：隐藏资源

接下来...

Spring 文件上传

示例：Apache Commons FileUpload 上传文件

Spring 文件下载

示例：隐藏资源

示例：防止交叉引用

Spring 用户登录

监听器



示例：隐藏资源

❖ 基本功能

- ▶ ResourceController 类处理用户登录，并将一个 secret.pdf 文件发送给浏览器。
- ▶ secret.pdf 文件放在 WEB-INF/data 目录下，无法直接访问。只有得到授权的用户，才能访问该文件。
- ▶ 如果用户没有登录，应用程序就会跳转到登录页面。



ResourceController

```
@Controller
public class ResourceController {
    @RequestMapping(value = "/login")
    public String login(@ModelAttribute Login login, HttpSession session, Model model) {
        model.addAttribute("login", new Login());
        if ("kevin".equals(login.getUserName()) && "secret".equals(login.getPassword())) {
            session.setAttribute("loggedIn", Boolean.TRUE);
            return "Main";
        } else {
            return "LoginForm";
        }
    }
}
```



示例：隐藏资源

ResourceController +

```
@RequestMapping(value = "/resource-download")
public String downloadResource(HttpSession session,
    HttpServletRequest request, HttpServletResponse response) {
    if (session == null || session.getAttribute("loggedIn") == null) {
        return "LoginForm";
    }

    String dataDirectory = request.getServletContext().getRealPath("/WEB-INF/data");
    File file = new File(dataDirectory, "secret.pdf");
    if (file.exists()) {
        response.setContentType("application/pdf");
        response.addHeader("Content-Disposition", "attachment; filename=secret.pdf");
        byte[] buffer = new byte[1024];
        FileInputStream fis = null;
        BufferedInputStream bis = null;
```



示例：隐藏资源

ResourceController ++

```
try {
    fis = new FileInputStream(file);
    bis = new BufferedInputStream(fis);
    OutputStream os = response.getOutputStream();
    int i = bis.read(buffer);
    while (i != -1) {
        os.write(buffer, 0, i);
        i = bis.read(buffer);
    } catch (IOException e) {
        // do something, probably forward to an Error page.
    } finally {
        if (bis != null) {
            try {
                bis.close();
            } catch (IOException e) {
                if (fis != null) {
                    try {
                        fis.close();
                    } catch (IOException e) {
                        return null;
                    }
                }
            }
        }
    }
}
```



示例：隐藏资源

LoginForm.jsp

```
<form:form commandName="login" action="login" method="post">
  <fieldset>
    <legend>Login</legend>
    <p>
      <label for="userName">User Name:</label>
      <form:input id="userName" path="userName" cssErrorClass="error" />
    </p>
    <p>
      <label for="password">Password: </label>
      <form:password id="password" path="password" cssErrorClass="error" />
    </p>
    <p id="buttons">
      <input id="reset" type="reset" tabindex="4">
      <input id="submit" type="submit" tabindex="5" value="Login">
    </p>
  </fieldset>
</form:form>
```



示例：隐藏资源

Main.jsp

```
<%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c" %>
<!DOCTYPE HTML>
<html>
<head>
<title>Download Page</title>
<style type="text/css">@import url("<c:urlvalue="/css/main.css"/>"); </style>
</head>
<body>
  <div id="global">
    <h4>Please click the link below. </h4>
    <p>
      <a href="resource-download">Download</a>
    </p>
  </div>
</body>
</html>
```



示例: 防止交叉引用

接下来...

Spring 文件上传

示例: Apache Commons FileUpload 上传文件

Spring 文件下载

示例: 隐藏资源

示例: 防止交叉引用

Spring 用户登录

监听器



示例: 防止交叉引用

👉 目的：避免其他站点通过交叉引用“窃取”你的网站资产

通过编程控制，使得只有当 referer 标题¹中包含你的域名时才发出资源，可以在一定程度上防止交叉引用。

示例实现 ImageController 类，使得**仅当 referer 标题不为 null 时，才将图片发送给浏览器**。这样可以防止仅在浏览器中输入网址就能下载图片的情况发生。

¹根据 HTTP 协议，在 HTTP 头中有一个字段为 Referer，它记录了该 HTTP 请求的来源地址。



示例: 防止交叉引用

ImageController

```
@Controller
public class ImageController {
    @RequestMapping(value="/get-image/{id}" , method = RequestMethod.GET)
    public void getImage(@PathVariable String id, HttpServletRequest request,
        HttpServletResponse response, @RequestHeader String referer) {
        if (referer != null) {
            // do something, send the file to the user.
        }
    }
}
```



接下来...

Spring 文件上传

示例：Apache Commons FileUpload 上传文件

Spring 文件下载

示例：隐藏资源

示例：防止交叉引用

Spring 用户登录

监听器



Spring MVC 拦截器

Spring MVC 的拦截器类似于 Servlet 开发中的过滤器 Filter，用于对处理器进行预处理和后处理。本质也是 AOP（面向切面编程），符合横切关注点的所有功能都可以放入拦截器实现。

❖ 常见应用场景

日志记录 记录请求信息的日志，以便进行信息监控、信息统计、计算 PV 等。

权限检查 如登录检测，进入处理器检测是否登录，如果没有直接返回到登录页面。

性能监控 通过拦截器在进入处理器之前记录开始时间，在处理完后记录结束时间，从而得到该请求的处理时间，以监控请求处理行为。

通用行为 只要是多个请求处理器都需要的即可使用拦截器实现。如，读取 cookie 得到用户信息并将用户对象放入请求，从而方便后续流程使用。



拦截器接口

org.springframework.web.servlet.HandlerInterceptor

```
package org.springframework.web.servlet;

public interface HandlerInterceptor {
    boolean preHandle(
        HttpServletRequest request, HttpServletResponse response,
        Object handler)
        throws Exception;

    void postHandle(
        HttpServletRequest request, HttpServletResponse response,
        Object handler, ModelAndView modelAndView)
        throws Exception;

    void afterCompletion(
        HttpServletRequest request, HttpServletResponse response,
        Object handler, Exception ex)
        throws Exception;
}
```



拦截器接口方法说明

preHandle 预处理回调方法。实现处理器的预处理（如登录检查），第三个参数为响应处理器（如 **Controller** 实现）。返回值 **true** 表示继续流程（如调用下一个拦截器或处理器）；**false** 表示流程中断（如登录检查失败），不会继续调用其他的拦截器或处理器，此时我们需要通过 **response** 来产生响应。

postHandle 后处理回调方法。

afterCompletion 整个请求处理完毕回调方法。



拦截器接口方法说明

preHandle 预处理回调方法。

postHandle 后处理回调方法。实现处理器的后处理（但在渲染视图之前），此时我们可以通过 **modelAndView**（模型和视图对象）对模型数据进行处理或对视图进行处理，**modelAndView** 也可能为 **null**。

afterCompletion 整个请求处理完毕回调方法。



拦截器接口方法说明

`preHandle` 预处理回调方法。

`postHandle` 后处理回调方法。

`afterCompletion` 整个请求处理完毕回调方法。在视图渲染完毕时回调，如性能监控中我们可以在此记录结束时间并输出消耗时间，还可以进行一些资源清理，类似于 `try-catch-finally` 中的 `finally`，但仅调用处理器执行链中 `preHandle` 返回 `true` 的拦截器的 `afterCompletion`。



拦截器适配器

有时候我们可能只需要实现三个回调方法中的某一个，如果实现 `HandlerInterceptor` 接口的话，三个方法必须实现，不管你需要不需要。

Spring MVC 提供了一个 `HandlerInterceptorAdapter` 适配器，允许我们只实现需要的回调方法。

```
public abstract class HandlerInterceptorAdapter implements HandlerInterceptor {  
    // 省略代码，此处所以三个回调方法都是空实现，preHandle返回true。  
}
```



示例：用户登录验证



示例参考

springmvc-interceptor-01



接下来...

Spring 文件上传

示例：Apache Commons FileUpload 上传文件

Spring 文件下载

示例：隐藏资源

示例：防止交叉引用

Spring 用户登录

监听器



监听器配置

在 Spring MVC 中配置监听器与原生 Servlet Listener 监听器配置相同。



参考

springmvc-intro-01



THE END

wangxiaodong@ouc.edu.cn

