第 2-6 课：使用 Spring Boot 和 Thymeleaf 演示上传文件

在互联网行业中上传文件是一个高频的使用场景，常用的案例有上传头像、上传身份证信息等。Spring Boot 利用 MultipartFile 的特性来接收和处理上传的文件，MultipartFile 是 Spring 的一个封装的接口，封装了文件上传的相关操作，利用 MultipartFile 可以方便地接收前端文件，将接收到的文件存储到本机或者其他中间件中。

首先通过一个小的示例来了解 Spring Boot 对上传文件的支持，项目前端页面使用 Thymeleaf 来处理。

快速上手

添加依赖包

```
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-web</artifactId>
</dependency>
<dependency>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-starter-thymeleaf</artifactId>
</dependency>
```

引入了 spring-boot-starter-thymeleaf 做页面模板引擎。

配置信息

常用配置内容，单位支持 MB 或者 KB：

```
#支持的最大文件
spring.servlet.multipart.max-file-size=100MB
#文件请求最大限制
spring.servlet.multipart.max-request-size=100MB
```

以上配置主要是通过设置 MultipartFile 的属性来控制上传限制，MultipartFile 是 Spring 上传文件的封装类，包含了文件的二进制流和文件属性等信息，在配置文件中也可对相关属性进行配置。

除过以上配置，常用的配置信息如下：

- spring.servlet.multipart.enabled=true，是否支持 multipart 上传文件
- spring.servlet.multipart.file-size-threshold=0，支持文件写入磁盘
- spring.servlet.multipart.location=，上传文件的临时目录
- spring.servlet.multipart.max-file-size=10Mb，最大支持文件大小

- spring.servlet.multipart.max-request-sizee=10Mb，最大支持请求大小
- spring.servlet.multipart.resolve-lazily=false，是否支持 multipart 上传文件时懒加载

启动类

```
@SpringBootApplication
public class FileUploadWebApplication {

    public static void main(String[] args) throws Exception {
        SpringApplication.run(FileUploadWebApplication.class, args);
    }

    //Tomcat large file upload connection reset
    @Bean
    public TomcatServletWebServerFactory tomcatEmbedded() {
        TomcatServletWebServerFactory tomcat = new TomcatServletWebServerFactory()
;
        tomcat.addConnectorCustomizers((TomcatConnectorCustomizer) connector -> {
            if ((connector.getProtocolHandler() instanceof AbstractHttp11Protocol<
?>)) {
                //-1 means unlimited
                ((AbstractHttp11Protocol<?>) connector.getProtocolHandler()).setMa
xSwallowSize(-1);
            }
        });
        return tomcat;
    }

}
```

TomcatServletWebServerFactory() 方法主要是为了解决上传文件大于 10M 出现连接重置的问题，此异常内容 GlobalException 也捕获不到。

编写前端页面

上传页面：

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<body>
<h1>Spring Boot file upload example</h1>
<form method="POST" action="/upload" enctype="multipart/form-data">
    <input type="file" name="file" /><br/><br/>
    <input type="submit" value="Submit" />
</form>
</body>
</html>
```

非常简单的一个 Post 请求，一个选择框选择文件、一个提交按钮，效果如下：

上传结果展示页面：

```
<!DOCTYPE html>
<html lang="en" xmlns:th="http://www.thymeleaf.org">
<body>
<h1>Spring Boot - Upload Status</h1>
<div th:if="${message}">
    <h2 th:text="${message}"/>
</div>
</body>
</html>
```

效果图如下：

编写上传控制类

访问 localhost:8080 自动跳转到上传页面：

```
@GetMapping("/")
public String index() {
    return "upload";
}
```

上传业务处理：

```
@PostMapping("/upload")
public String singleFileUpload(@RequestParam("file") MultipartFile file,
                               RedirectAttributes redirectAttributes) {
    if (file.isEmpty()) {
        redirectAttributes.addFlashAttribute("message", "Please select a file to u
pload");
        return "redirect:uploadStatus";
    }
    try {
        // Get the file and save it somewhere
        byte[] bytes = file.getBytes();
        // UPLOADED_FOLDER 文件本地存储地址
        Path path = Paths.get(UPLOADED_FOLDER + file.getOriginalFilename());
        Files.write(path, bytes);

        redirectAttributes.addFlashAttribute("message",
                "You successfully uploaded '" + file.getOriginalFilename() + "'");

    } catch (IOException e) {
        e.printStackTrace();
    }
    return "redirect:/uploadStatus";
}
```

上面代码的意思就是，通过 MultipartFile 读取文件信息，如果文件为空跳转到结果页并给出提示；如果不为空读取文件流并写入到指定目录，最后将结果展示到页面。最常用的是最后两个配置内容，限制文件上传大小，上传时超过大小会抛出异常：

Spring Boot - Upload Status

org.apache.tomcat.util.http.fileupload.FileUploadBase$SizeLimitExceededException: the request was rejected because its size (108105436) exceeds the configured maximum (10485760)

当然在真实的项目中我们可以在业务中会首先对文件大小进行判断，再将返回信息展示到页面。

异常处理

这里演示的是 MultipartException 的异常处理，也可以稍微改造监控整个项目的异常问题。

```
@ControllerAdvice
public class GlobalExceptionHandler {

    @ExceptionHandler(MultipartException.class)
    public String handleError1(MultipartException e, RedirectAttributes redirectAt
tributes) {
        redirectAttributes.addFlashAttribute("message", e.getCause().getMessage())
;
        return "redirect:/uploadStatus";
    }
}
```

设置一个 @ControllerAdvice 用来监控 Multipart 上传的文件大小是否受限，当出现此异常时在前端页面给出提示。利用 @ControllerAdvice 可以做很多东西，比如全局的统一异常处理等，感兴趣的读者可以抽空了解一下。

上传多个文件

在项目中经常会有一次性上传多个文件的需求，我们稍作修改即可支持。

前端页面

首先添加可以支持上传多文件的页面，内容如下：

```
<!DOCTYPE html>
<html xmlns:th="http://www.thymeleaf.org">
<body>
<h1>Spring Boot files upload example</h1>
<form method="POST" action="/uploadMore" enctype="multipart/form-data">
    文件1： <input type="file" name="file" /><br/><br/>
    文件2： <input type="file" name="file" /><br/><br/>
    文件3： <input type="file" name="file" /><br/><br/>
    <input type="submit" value="Submit" />
</form>
</body>
</html>
```

后端处理

后端添加页面访问入口：

```
@GetMapping("/more")
public String uploadMore() {
    return "uploadMore";
}
```

在浏览器中输入网址，http://localhost:8080/more，就会进入此页面。

MultipartFile 需要修改为按照数组的方式去接收。

```
@PostMapping("/uploadMore")
public String moreFileUpload(@RequestParam("file") MultipartFile[] files,
                             RedirectAttributes redirectAttributes) {
    if (files.length==0) {
        redirectAttributes.addFlashAttribute("message", "Please select a file to u
pload");
        return "redirect:uploadStatus";
    }
    for(MultipartFile file:files){
        try {
            byte[] bytes = file.getBytes();
            Path path = Paths.get(UPLOADED_FOLDER + file.getOriginalFilename());
            Files.write(path, bytes);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
    redirectAttributes.addFlashAttribute("message", "You successfully uploaded all
");
    return "redirect:/uploadStatus";
}
```

同样是先判断数组是否为空，在循环遍历数组内容将文件写入到指定目录下。在浏览器中输入网址 http://localhost:8080/more，选择三个文件进行测试，当页面出现以下信息时表示上传成功。

```
Spring Boot - Upload Status
You successfully uploaded all
```

总结

经过本课的内容发现 Spring Boot 完美支持文件上传，不论是单个文件上传还是多个文件上传都很简单。在上传文件的过程中也可以通过配置文件来选择关闭或者限制上传文件大小，利用 @ControllerAdvice 可以灵活地对异常进行全局处理。

> 点击这里下载源码。