前置知识（SpringBoot文件上传与下载）

时间：2023年02月11日 19:40:03

一、Multipart?

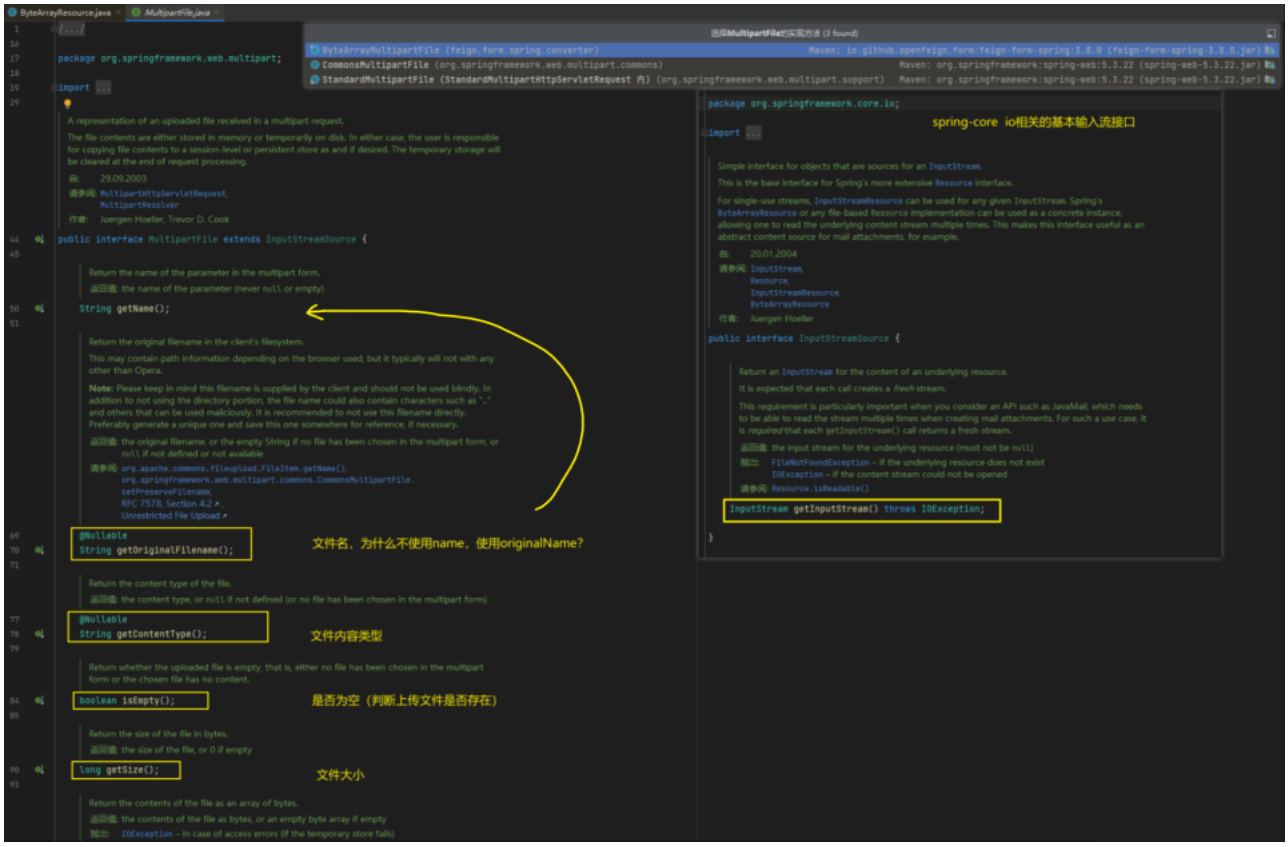
> HTML中有没有一种统一的文件上传方式，例如通过表单的encrypt类型指定。

- enctype 属性规定在将表单数据发送到服务器之前如何对其进行编码。

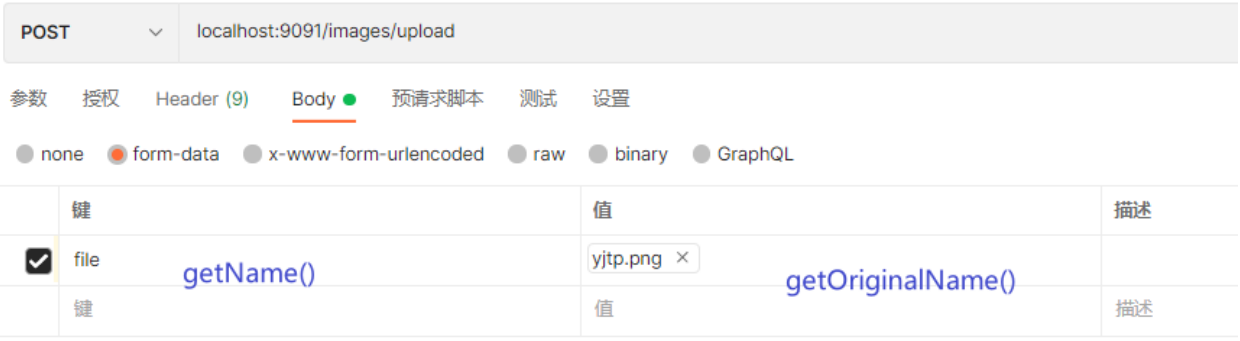
值	描述
application/x-www-form-urlencoded	默认。在发送前对所有字符进行编码（将空格转换为 "+" 符号，特殊字符转换为 ASCII HEX 值）。
multipart/form-data	不对字符编码。当使用有文件上传控件的表单时，该值是必需的。
text/plain	将空格转换为 "+" 符号，但不编码特殊字符。

1. 什么是MultipartFile

- html表单中的一种数据encrypt格式，用于上传文件
- spring中提供的一个接口+多个实现，用于接收并处理前端web表单请求中的文件数据



- *getName*
  - 获取请求文件的属性名



- *getOriginalName*
  - 获取文件名
- *getContentType*
  - image/png
  - ...(MIME类型)
- *getSize*
  - 获取文件大小（字节）
- *isEmpty*
  - 上传文件是否为空 -> getSize==0（*例如一个空的.txt文件*）

2. 多文件使用Multipart接收

> - 不使用注解指定文件所在的key
> - 使用注解指定文件所在的key，此时可以使用@RequestPart或@RequestParam。

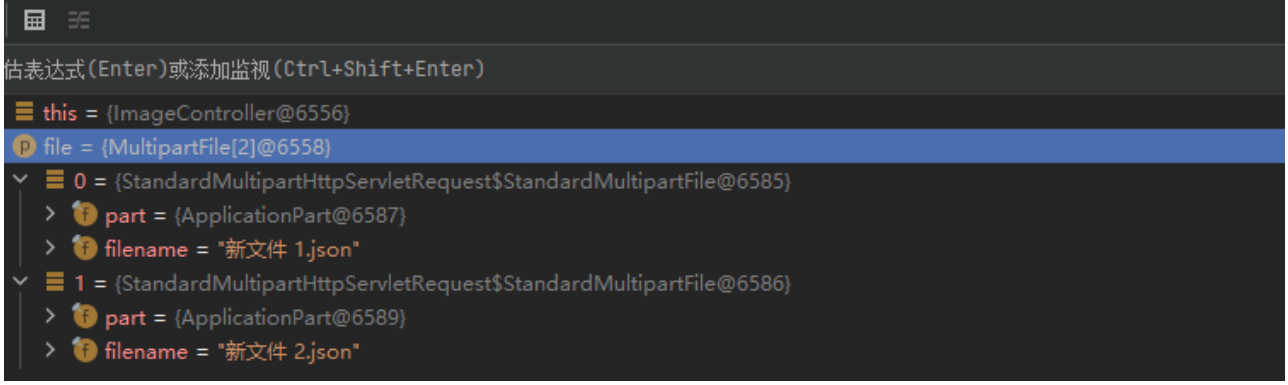
```
/**
 * 根据表单属性解析出Multipart，可以使用@RequestPart、也可以使用@RequestParam
 *
 * @param fileArray 多文件
 */
0 个用法  新 *
@PostMapping("/upload")
public void multiImageImport(@RequestPart("files") MultipartFile[] fileArray) {

    System.out.println(Arrays.toString(fileArray));
}
```

```
0 个用法  新 *
@PostMapping("/upload")
public void multiImageImport(MultipartFile[] file) {   file: MultipartFile[2]@6558

    System.out.println("file = " + file);   file: MultipartFile[2]@6558
}
```

多文件直接使用数组接收

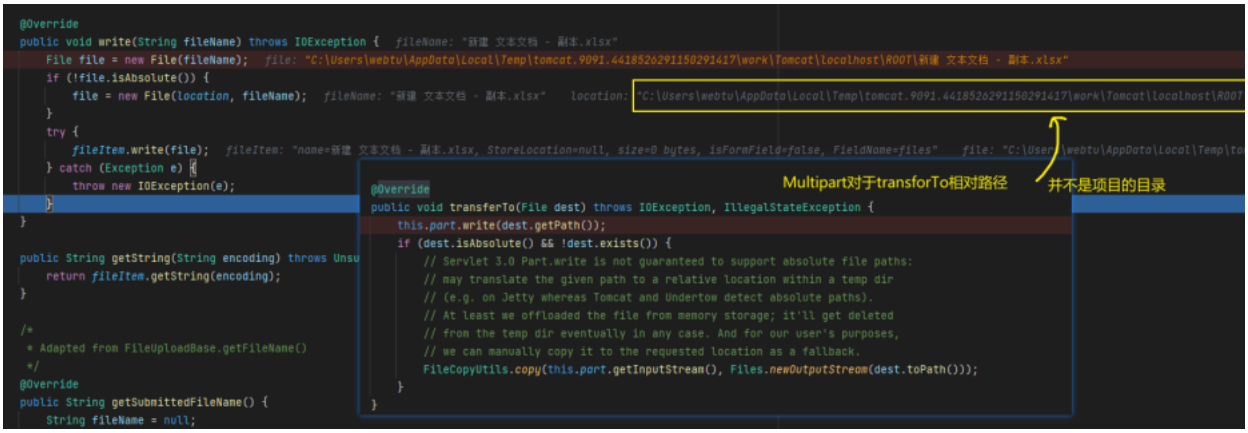


3. Multipart数据写入文件

- *使用MultipartFile提供的transferTo（File），但是这里只支持绝对路径，因为相对路径默认不在项目目录下！*

```
/**
 * 根据表单属性解析出Multipart，可以使用@RequestPart、也可以使用@RequestParam
 *
 * @param fileArray 多文件
 */
0 个用法  新 *
@PostMapping("/upload")
public void multiImageImport(@RequestPart("files") MultipartFile[] fileArray) {

    for (MultipartFile multipartFile : fileArray) {
        String filename = Objects.requireNonNull(multipartFile.getOriginalFilename());
        try (
                OutputStream os = new FileOutputStream(filename);
                InputStream is = multipartFile.getInputStream();
        ) {
            multipartFile.transferTo(new File(multipartFile.getOriginalFilename()));
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

- 

- ***<u>如果想要使用相对路径，请手动获取InputStream，创建OutputStream，手动将数据写入到输出流</u>***。
  - JDK9中对InputStream提供了transferTo接口，可以直接以8KB的缓冲区大小将数据写入到输出流；
  - ```java
    /**
     * 根据表单属性解析出Multipart，可以使用@RequestPart、也可以使用@RequestParam
     *
     * @param fileArray 多文件
     */
    0 个用法  新 *
    @PostMapping("/upload")
    public void multiImageImport(@RequestPart("files") MultipartFile[] fileArray) {

        for (MultipartFile multipartFile : fileArray) {
            String filename = Objects.requireNonNull(multipartFile.getOriginalFilename());
            try (
                    OutputStream os = new FileOutputStream(filename);
                    InputStream is = multipartFile.getInputStream();
            ) {
                // JDK9+
                // is.transferTo(os);

                final int BUFFER_SIZE = 8 * 1024;
                byte[] bytes = new byte[BUFFER_SIZE];
                int read;
                while ((read = is.read(bytes, off: 0, BUFFER_SIZE)) >= 0) {
                    os.write(bytes, off: 0, read);
                }
            } catch (IOException e) {
                throw new RuntimeException(e);
            }
        }
    }
    ```
    ```

4. Multipart文件在哪一个Servlet中进行加载?

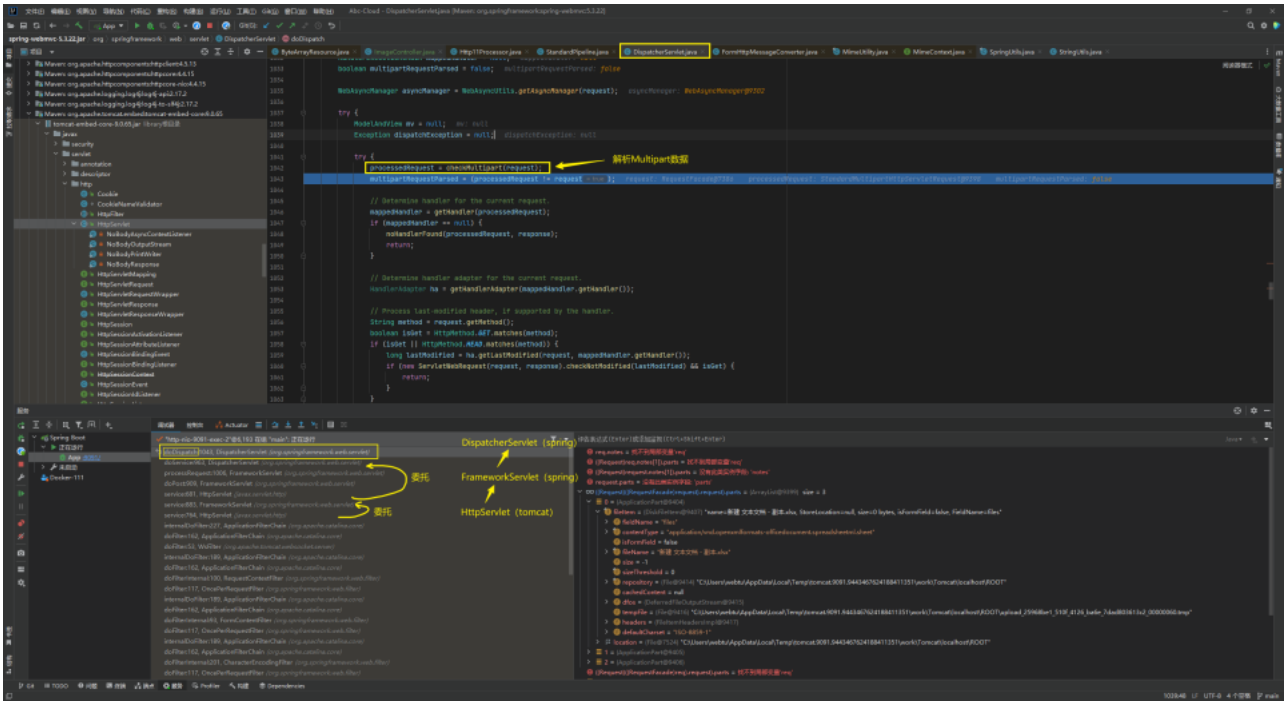
5. MIME解析（Spring-Web）

> *问这个问题之初，是为了找到spring内部有无描述所有MIME类型的类*

- 最后发现文件上传的类型，是在form表单选中文件后自动初始化，没有在后端进行解析MIME格式
  - https://developer.mozilla.org/zh-CN/docs/Web/HTML/Element/Input/file
  - https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Basics_of_HTTP/MIME_types
- spring-web也提供了MIME解析类
  -

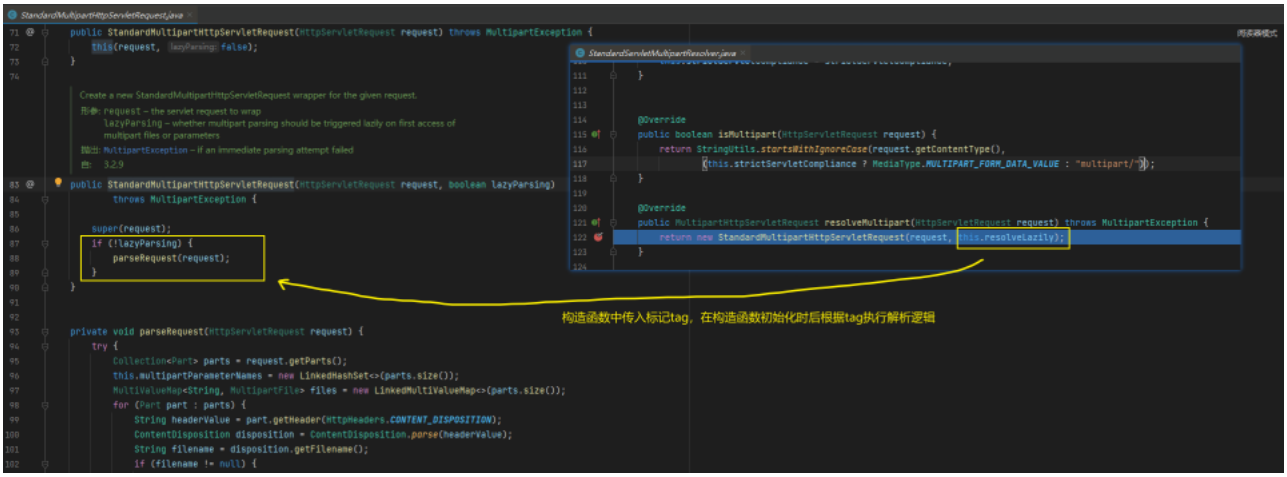
```java
@SpringBootApplication
public class App {

    0 个用法  trivis *
    public static void main(String[] args) {
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.tzt").orElse(MediaType.TEXT_XML));  //
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.txt").orElse(MediaType.TEXT_XML));
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.png").orElse(MediaType.TEXT_XML));
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.gif").orElse(MediaType.TEXT_XML));
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.xlsx").orElse(MediaType.TEXT_XML));
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.pdf").orElse(MediaType.TEXT_XML));
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.ts").orElse(MediaType.TEXT_XML));  // X
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.flv").orElse(MediaType.TEXT_XML));
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.mp4").orElse(MediaType.TEXT_XML));
        SpringApplication.run(App.class, args);
    }
}
```



```
Convert the request into a multipart request, and make multipart resolver available.
If no multipart resolver is set, simply use the existing request.
形参:  request – current HTTP request
返回值: the processed request (multipart wrapper if necessary)
请参阅: MultipartResolver.resolveMultipart
protected HttpServletRequest checkMultipart(HttpServletRequest request) throws MultipartException {
    if (this.multipartResolver != null && this.multipartResolver.isMultipart(request)) {
        if (WebUtils.getNativeRequest(request, MultipartHttpServletRequest.class) != null) {
            if (DispatcherType.REQUEST.equals(request.getDispatcherType())) {
                logger.trace("Request already resolved to MultipartHttpServletRequest, e.g. by MultipartFilter");
            }
        }
        else if (hasMultipartException(request)) {
            logger.debug("Multipart resolution previously failed for current request - " +
                    "skipping re-resolution for undisturbed error rendering");
        }
        else {
            try {
                return this.multipartResolver.resolveMultipart(request);
            }
            catch (MultipartException ex) {
                if (request.getAttribute(WebUtils.ERROR_EXCEPTION_ATTRIBUTE) != null) {
                    logger.debug( message: "Multipart resolution failed for error dispatch", ex);
                    // Keep processing error dispatch with regular request handle below
                }
                else {
                    throw ex;
                }
            }
        }
    }
    // If not returned before: return original request.
    return request;
}
```

multipart处理器



二、MIME解析?

> https://developer.mozilla.org/zh-CN/docs/Web/HTTP/Basics_of_HTTP/MIME_types

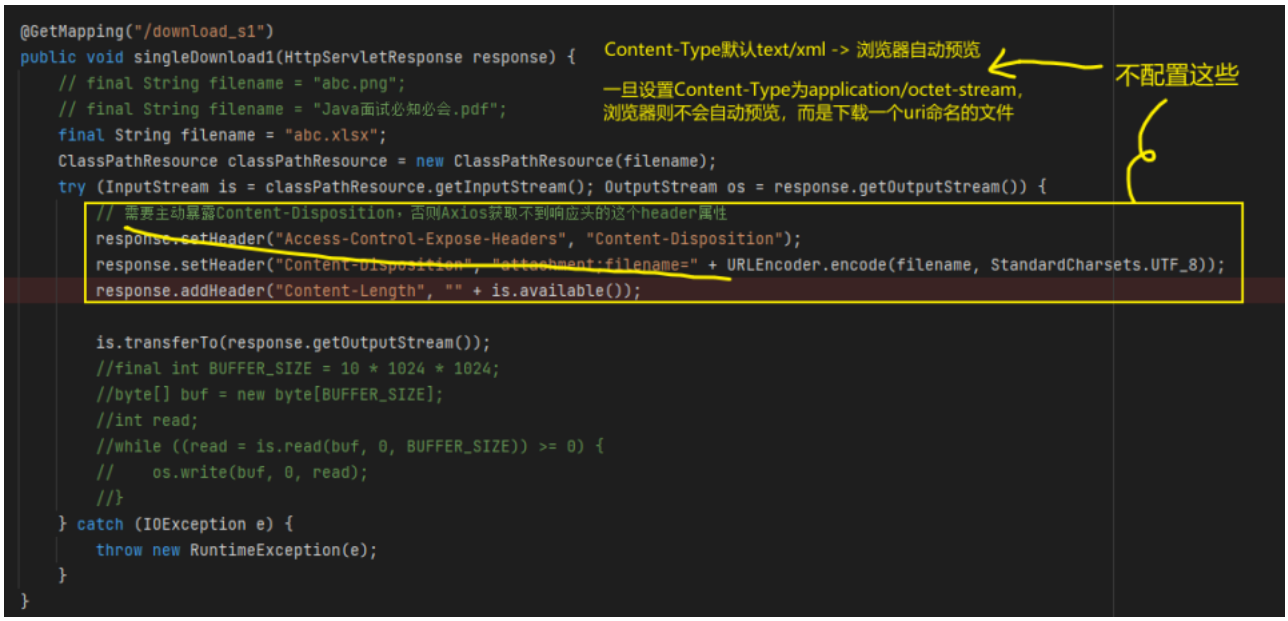
- spring-web也提供了MIME解析类
  -

```java
@SpringBootApplication
public class App {

    0 个用法  trivis *
    public static void main(String[] args) {
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.tzt").orElse(MediaType.TEXT_XML));  // X
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.txt").orElse(MediaType.TEXT_XML));
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.png").orElse(MediaType.TEXT_XML));
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.gif").orElse(MediaType.TEXT_XML));
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.xlsx").orElse(MediaType.TEXT_XML));
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.pdf").orElse(MediaType.TEXT_XML));
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.ts").orElse(MediaType.TEXT_XML));  // X
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.flv").orElse(MediaType.TEXT_XML));
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.mp4").orElse(MediaType.TEXT_XML));
        SpringApplication.run(App.class, args);
    }
}
```

三、application/octet-stream?

> 默认blob类型为text/xml，如果手动设置为application/octet-stream，浏览器则不会自动预览（图片、PDF）
>
> - 而是统一下载为一个URI名称的.file格式文件




```java
@GetMapping("/download_s1")
public void singleDownload1(HttpServletResponse response) {
    // final String filename = "abc.png";
    // final String filename = "Java面试必知必会.pdf";
    final String filename = "abc.xlsx";
    ClassPathResource classPathResource = new ClassPathResource(filename);
    try (InputStream is = classPathResource.getInputStream(); OutputStream os = response.getOutputStream()) {
        // 需要主动暴露Content-Disposition，否则Axios获取不到响应头的这个header属性
        response.setHeader("Access-Control-Expose-Headers", "Content-Disposition");
        response.setHeader("Content-Disposition", "attachment;filename=" + URLEncoder.encode(filename, StandardCharsets.UTF_8));
        response.addHeader("Content-Length", "" + is.available());

        is.transferTo(response.getOutputStream());
        //final int BUFFER_SIZE = 10 * 1024 * 1024;
        //byte[] buf = new byte[BUFFER_SIZE];
        //int read;
        //while ((read = is.read(buf, 0, BUFFER_SIZE)) >= 0) {
        //    os.write(buf, 0, read);
        //}
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

四、Axios无法获取到Content-Disposition?

默认情况下，header只有六种 simple response headers （简单响应首部）可以暴露给外部：

- Cache-Control
- Content-Language
- Content-Type
- Expires
- Last-Modified
- Pragma

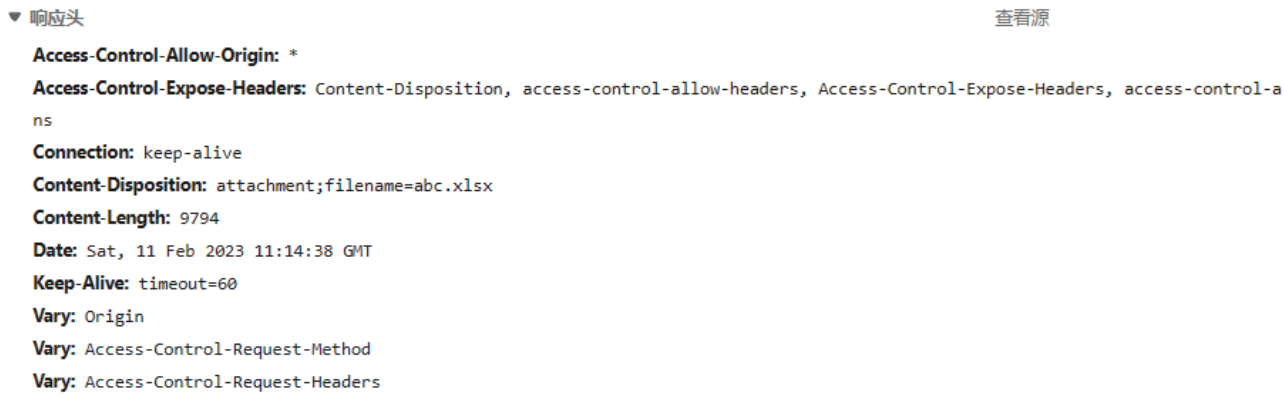
1. 解决方案一（全局控制）

```
import lombok.NonNull;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class CORSConfiguration {

    /**
     * 注意：跨域会导致页面获取不到response-headers!
     *
     * @return WebMvcConfigurer
     */
    @Bean
    public WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurer() {
            @Override
            public void addCorsMappings(@NonNull CorsRegistry registry) {
                registry.addMapping("/images/**")
                        .allowedOrigins("*")
                        .allowedMethods("GET", "POST")
                        .exposedHeaders("Content-Disposition",
                                "access-control-allow-headers",
                                "Access-Control-Expose-Headers",
                                "access-control-allow-methods",
                                "access-control-allow-origin",
                                "access-control-max-age",
                                "X-Frame-Options")
                        .allowCredentials(false).maxAge(3600);
            }
        };
    }

}
```



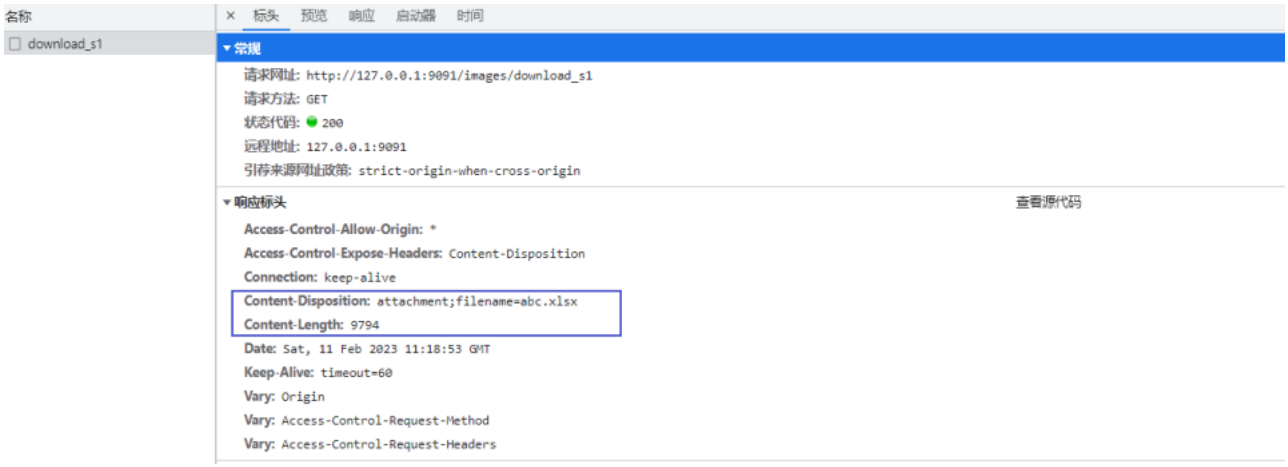
2. 方案二（针对指定接口进行控制）

```
response.setHeader("Access-Control-Expose-Headers", "Content-Disposition");
```

```
    /**
     * 使用Content-Disposition进行图片下载
     *
     * @param response HttpServletResponse
     */
    @GetMapping("/download_s1")
    public void singleDownload1(HttpServletResponse response) {
        // final String filename = "abc.png";
        // final String filename = "Java面试必知必会.pdf";
        final String filename = "abc.xlsx";
        ClassPathResource classPathResource = new ClassPathResource(filename);
        try (InputStream is = classPathResource.getInputStream(); OutputStream os = response.getOutputStream()) {
            response.setHeader("Access-Control-Expose-Headers", "Content-Disposition");
            response.setHeader("Content-Disposition", "attachment;filename=" + URLEncoder.encode(filename,
StandardCharsets.UTF_8));
            response.addHeader("Content-Length", "" + is.available());

            is.transferTo(response.getOutputStream());
            //final int BUFFER_SIZE = 10 * 1024 * 1024;
            //byte[] buf = new byte[BUFFER_SIZE];
            //int read;
            //while ((read = is.read(buf, 0, BUFFER_SIZE)) >= 0) {
            //    os.write(buf, 0, read);
            //}
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
```

```
> res
< ▼ {data: Blob, status: 200, statusText: '', headers: i, config: {…}, …}
    ▶ config: {transitional: {…}, adapter: Array(2), transformRequest: Array(1), transformResponse: Array(1), timeout: 30000, …}
    ▶ data: Blob {size: 9794, type: 'text/xml'}
    ▼ headers: i
        content-disposition: "attachment;filename=abc.xlsx"
        content-length: "9794"
        Symbol(Symbol.toStringTag): (...)
      ▶ [[Prototype]]: Object
    ▶ request: XMLHttpRequest {onreadystatechange: null, readyState: 4, timeout: 30000, withCredentials: false, upload: XMLHttpRequestUpload, …}
      status: 200
      statusText: ""
    ▶ [[Prototype]]: Object
>
```



五、CORS跨域问题?

> 请在后端处理跨域问题CORS
>
> - Cross-Origin-Resource-Sharing 跨域资源共享

```
import lombok.NonNull;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

@Configuration
public class CORSConfiguration {

    @Bean
    public WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurer() {
            @Override
            public void addCorsMappings(@NonNull CorsRegistry registry) {
                registry.addMapping("/images/**")
                        .allowedOrigins("*")
                        .allowedMethods("GET", "POST")
                        .exposedHeaders("Content-Disposition",
                                "access-control-allow-headers",
```

```
                        "Access-Control-Expose-Headers",
                        "access-control-allow-methods",
                        "access-control-allow-origin",
                        "access-control-max-age",
                        "X-Frame-Options")
                .allowCredentials(false).maxAge(3600);
        }
    };
  }

}
```

六、SpringBoot上传文件太大报错?

1. 上传文件大小超出限制


```yaml
spring:
  servlet:
    multipart:
      max-file-size: 100MB
      max-request-size: 300MB
```

```html
<h3>上传单个文件</h3>
<div style="text-align: center">
    <form action="upload" enctype="multipart/form-data" method="post">
        <input type="file" name="file" required>
        <br>
        <input type="submit" value="上传单个文件">
    </form>
</div>
<br><hr style="width: 400px;">
<h3>上传多个文件</h3>
<form action="multiUpload" enctype="multipart/form-data" method="post">
    <input type="file" name="file" required>
    <br>
    <input type="file" name="file" required>
    <br>
    <input type="file" name="file" required
    ><br>
    <input type="submit" value="上传多个文件">
</form>
```

2. POST请求大小超出限制

> 配置Tomcat允许的POST请求最大值

```yaml
server:
  port: 8081
  tomcat:
    threads:
      max: 200
    max-connections: 8192
    max-http-form-post-size: 2MB
    max-swallow-size: 2MB
```

form表单中post请求最大大小

requestBody最大大小

调整为-1表示不限制

提示!
前端如果使用Axios统一发送get/post请求,
请在后端调节max-swallow-size至合适大小
↓
特别是某些业务需要上传序列化后的二进制数据,
此时requestBody中数据很容易会>2MB

（下方配置是默认值）

```yaml
server:
  port: 8081
  tomcat:
    threads:
      max: 200
    max-connections: 8192
    max-http-form-post-size: 2MB
    max-swallow-size: 2MB
  max-http-header-size: 8KB

spring:
  servlet:
    multipart:
      max-file-size: 1MB
      max-request-size: 10MB
```