02_SpringBoot实现图片下载

时间：2023年02月11日 13:47:33

方式一：将图片数据base64编码为字符串，以JSON的形式返回

方式二：直接将二进制数据流写入response

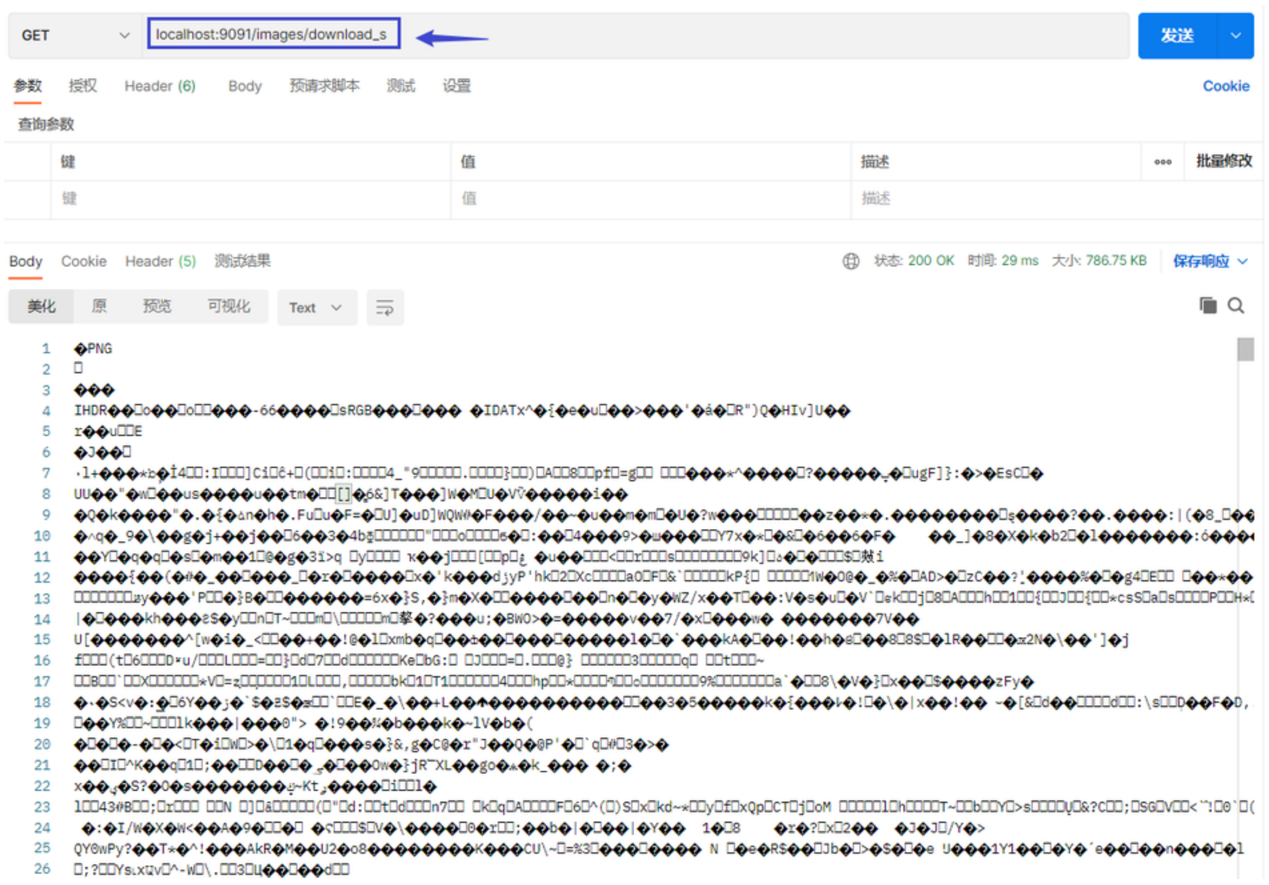
注意：请在后端处理跨域问题CORS

> 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 {

    /**
     * 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")
                        .allowCredentials(false).maxAge(3600);
            }
        };
    }

}
```

方式一：将图片数据base64编码为字符串，以JSON的形式返回

```
    @GetMapping("/download_s")
    public String singleDownload() {
        ClassPathResource classPathResource = new ClassPathResource("abc.png");
        try (InputStream inputStream = classPathResource.getInputStream()) {
            return new String(inputStream.readAllBytes());
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
```



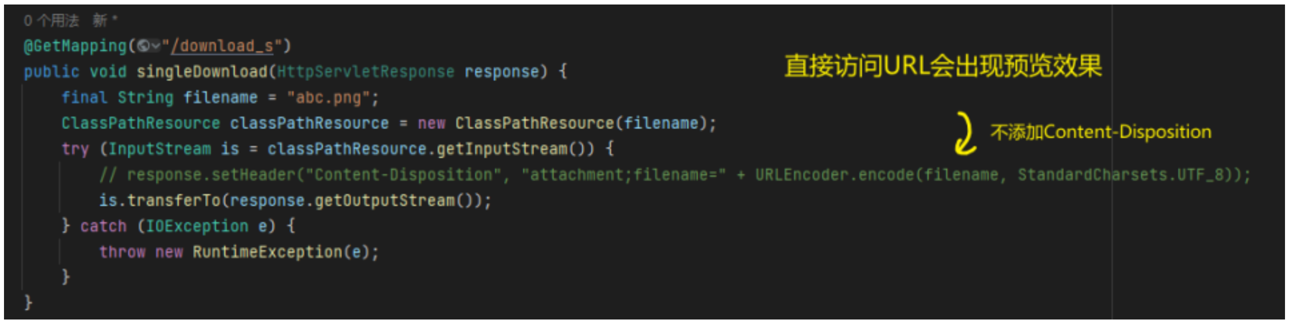
键	值	描述	批量修改
键	值	描述	

```
�PNG
□
���
IHDR��□o��□o□□���-66�����□sRGB���□��� �IDATx^�{�e�u□��>���'�á�□R")Q�HIv]U��
r��u□□E
�J��□
·l+���*b�İ4□□:I□□□]Ci□ĉ+□(□□i□:□□□□4_"9□□□□□.□□□□}□□)□A□□8□□pf□=g□□ □□□���*^����□?�����̧�□ugF]}:�>�EsC□�
UU��"�w□��us����u��tm�□□[]�̧6&]T���]W�M□U�VV̈������i��
�Q�k����"�.�{�ₐn�h�.Fu□u�F=�□U]�uD]WQW⩍�F���/��~�u��m�m□�U�?w���□□□□□��z��*�.���������□ş����?��.����:|(�8_□��
�^q�_9�\��g�j+��j��□6��3�4bᵹ□□□□□□□"□□□o□□□□6�□:��□4����9>�w����□□Y7x�*□�&□�6���6�F�    ��_]�8�X�k�b2□�l��������:ó����
��Y□�q�q□�s□�m��1□@�g�3ï>q □y□□□□ ĸ��j□□□[□□p□¿ �u��□□□<□□r□□□s□□□□□□□□9k]□₀�□�□□□$□燚i
����{��(�⋕�_��□���_□�r�□����□x�'k���d;yP'hk□2□Xc□□□□a0□F□&`□□□□□kP{□ □□□□□1W�0@�_�%�□AD>�□zC��?̣����%�□�g4□E□□ □��*��
□□□□□□□ᵶy���'P□□�}B�□□�������=6x�}S,�}m�X�□□�������□��□n�□�y�WZ/x��T□��:V�s�u□�V`□ɞk□□j□8□A□□□h□□1□□{□□J□□{□□*csS□a□s□□□□P□□H*□
|�□����kh����ᵋS�y□□n□T~□□□m□\□□□□□m□擊�?���u;�BW0>�=�����v��7/�x□����w� �������7V��
U[�������^[w�i�_<□□��+��!@�l□xmb�q□����□����□�����l�□�`���kA�□��!��h�ƨ□���8□8$□�lR��□□�ᵯ2N�\��']�j
f□□□(t□6□□□D*u/□□□L□□□=□□}□d□7□□d□□□□□□Ke□bG:□ □J□□□=□.□□□@} □□□□□□3□□□□□q□ □□t□□□~
□□B□□`□□X□□□□□□□*V□=z□□□□□□1□L□□□,□□□□□bk□1□T1□□□□□□□4□□□hp□□*□□□□ⁿ□□ₒ□□□□□□□□9%□□□□□□□□a`�□□8\�V�}□x��□$����zFy�
�·�S<v�:�□6Y��;�`$�ƨ$�ᵯ□□`□□E�_�\��+L��↑�����������□□��3�5������k�{���⊦�!□�\�|x��!�� ~�[&□d��□□□□d□□:\s□□D��F�D,
□��Y%□□~□□□lk���|���0"> �!9��½�b���k�~lV�b�(
�□�□�-��<□T�i□W□>�\□1�q□���s�}&,gC@�r"J��Q�@P'�□`q□#□3�>�
��□I□^K��q□1□;��□□D��□ ̧�□��0w�}jR¯XL��go�ₐ�k_��� �;�
x��̧�S?�0�s�������ₑ-Kt,����□i□□l�
1□□43#B□□;□r□□□ □□N □]□ₐ□□□□□(□"□d:□□t□d□□□n7□□ □k□q□A□□□□F□6□^(□)S□x□kd~*□□y□f□xQp□CT□j□oM □□□□□1□h□□□□T~□□b□□Y□>s□□□□y□&?C□□;□SG□V□□<"!□0`□(
 �:�I/W�X�W<��A�9□□�□ �ς□□□$□V�\����□0�r□□;��b�|�□��|�Y�� 1�□8  �r�?□x□2�� �J�J□/Y�>
QY0wPy?��T*�^!���AkR�M��U2�o8��������K���CU\~□=%3□���□���� N □�e�R$��□Jb�□>�$�□�e U���1Y1��□�Y�`e��□��n���□�l
□;?□□Ys.x̄v□^-W□\.□□3□u��□��d□□
```

方式二：直接将二进制数据流写入response

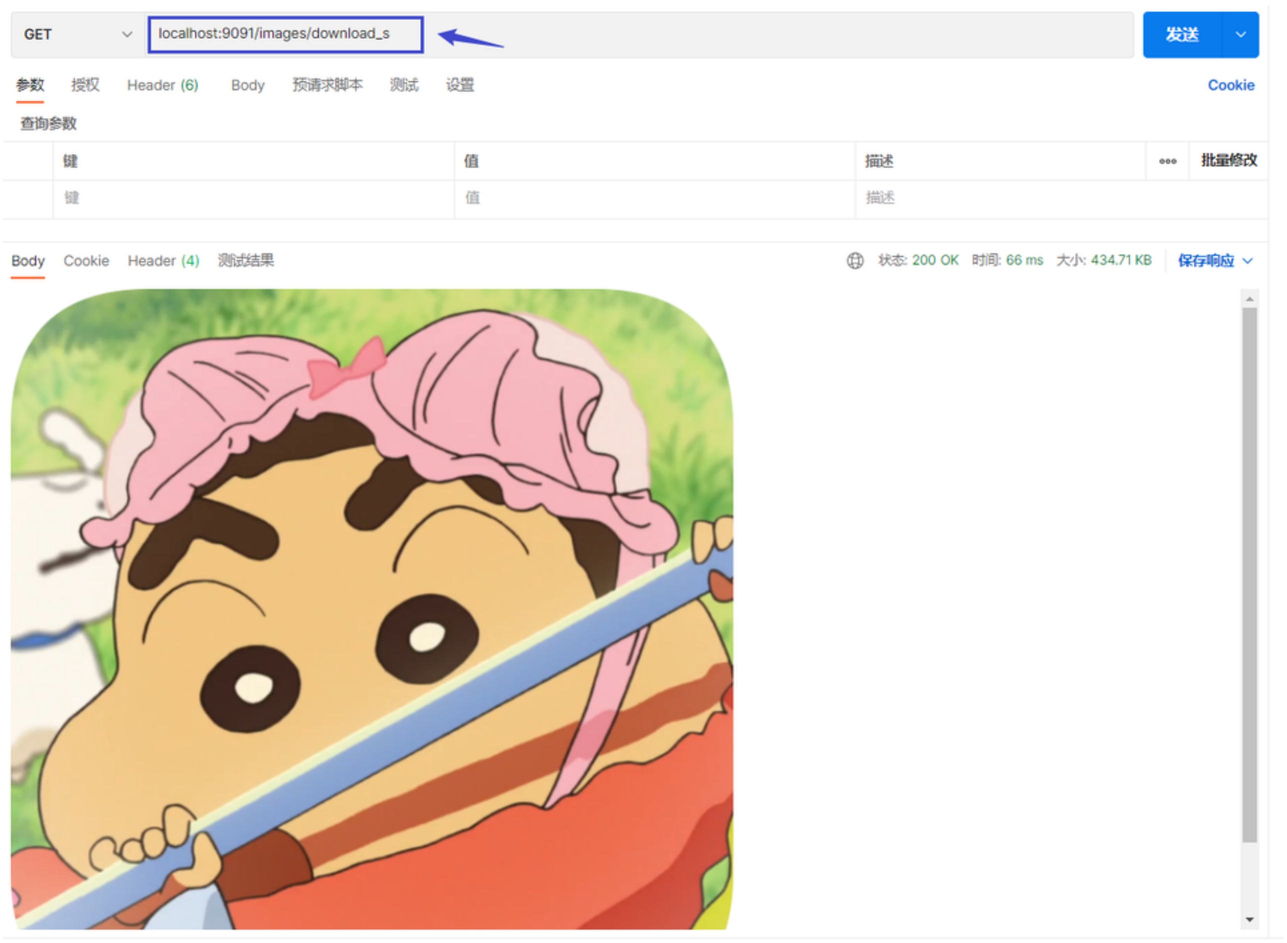
- 这种方式代码简洁，但是文件名、文件类型在响应中都不能指定（适合预览的需求）

> 此时，在浏览器中访问，将会是预览的效果；（为什么？默认输出流的ContentType=text/xml）
>
> - response.setContentType(MediaType.TEXT_XML_VALUE);
>   - 如果是图片、pdf会自动预览（Office系列文档可不行哦！）
> - response.setContentType(MediaType.APPLICATION_OCTET_STREAM_VALUE);
>   - download_s1 (1)
>   打开文件
>
>   download_s1
>   打开文件
> - 如果想要下载，则需要在前段页面中进行额外处理；
>   - 前端处理：预览
>   - 前端处理：下载



```
0 个用法  新 *
@GetMapping("/download_s")
public void singleDownload(HttpServletResponse response) {
    final String filename = "abc.png";
    ClassPathResource classPathResource = new ClassPathResource(filename);
    try (InputStream is = classPathResource.getInputStream()) {
        // response.setHeader("Content-Disposition", "attachment;filename=" + URLEncoder.encode(filename, StandardCharsets.UTF_8));
        is.transferTo(response.getOutputStream());
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```

```
@GetMapping("/download_s")
public String singleDownload(HttpServletResponse response) {
    ClassPathResource classPathResource = new ClassPathResource("abc.png");
    try {
        classPathResource.getInputStream().transferTo(response.getOutputStream());
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
    return "OK";
}
```



localhost:9091/images/download_s

状态: 200 OK 时间: 66 ms 大小: 434.71 KB

1. 前端处理（下载）

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>文件预览与下载</title>
    <script src=”./axios.min.js”></script>
    <script>
        function downloadFile() {
            let axios1 = axios.create({
                baseURL: ’http://127.0.0.1:9091’,
                timeout: 30000,
                // 默认是json类型数据
                headers: {
                    // ’Content-Type’: ’application/x-www-form-urlencoded; charset=utf-8’
                    ’Content-Type’: ’application/json; charset=utf-8’
                }
            });
            // 从后台获取数据
            axios1.get(’/images/download_s’, {
                responseType: ’blob’
            }).then(res => {
                const url = window.URL.createObjectURL(new Blob([res.data]))
                const link = document.createElement(’a’)
                link.href = url
                link.setAttribute(’download’, ’下载.png’) // 下载文件的名称及文件类型后缀
                document.body.appendChild(link)
                link.click()
                window.URL.revokeObjectURL(url) // 释放掉blob对象
            });
        }
    </script>
</head>
<body>

<button onclick=”downloadFile()”>下载文件</button>
</body>
</html>
```

2. 前端处理（预览）

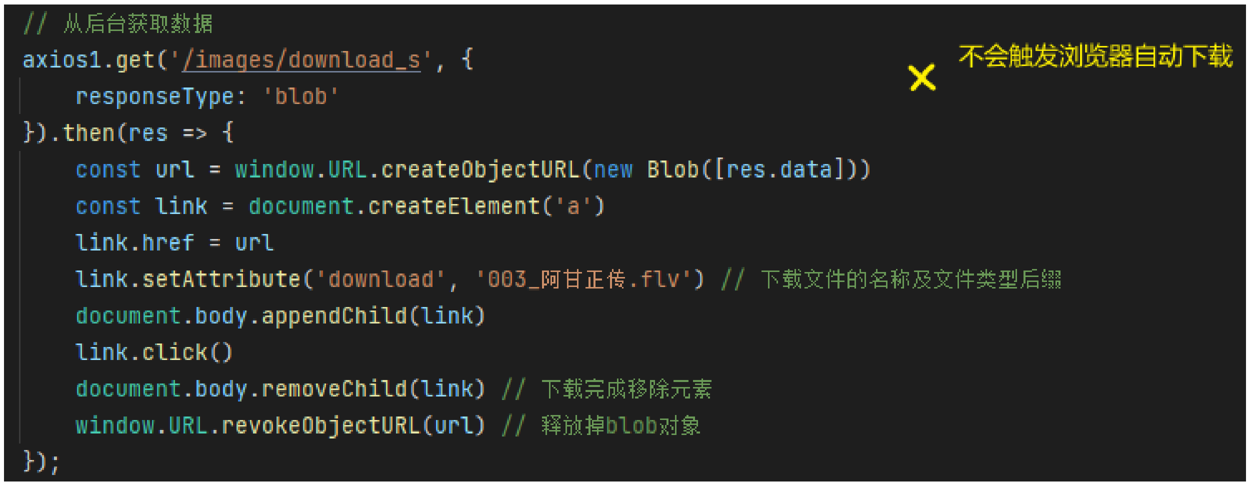
- 1–>直接访问URL
- 2–>后台拦截blob常见URL对象，打开(
    - 此时，需要在创建URL对象的同时指定文件类型（MIME类型）

```
<!DOCTYPE html>
<html lang=”en”>
<head>
    <meta charset=”UTF-8”>
    <title>文件预览与下载</title>
    <script src=”./axios.min.js”></script>
    <script>
        function previewFile() {
            let axios1 = axios.create({
                baseURL: ’http://127.0.0.1:9091’,
                timeout: 30000,
                // 默认是json类型数据
                headers: {
                    // ’Content-Type’: ’application/x-www-form-urlencoded; charset=utf-8’
                    ’Content-Type’: ’application/json; charset=utf-8’
                }
            });
            // 从后台获取数据
            axios1.get('/images/download_s', {
                responseType: 'blob'
            }).then(res => {
                const url = window.URL.createObjectURL(new Blob([res.data], {type: "image/png"}))
                window.open(url);
                window.URL.revokeObjectURL(url) // 释放掉blob对象
            });
        }
    </script>
</head>
<body>

<button onclick="previewFile()">预览文件</button>
</body>
</html>
```

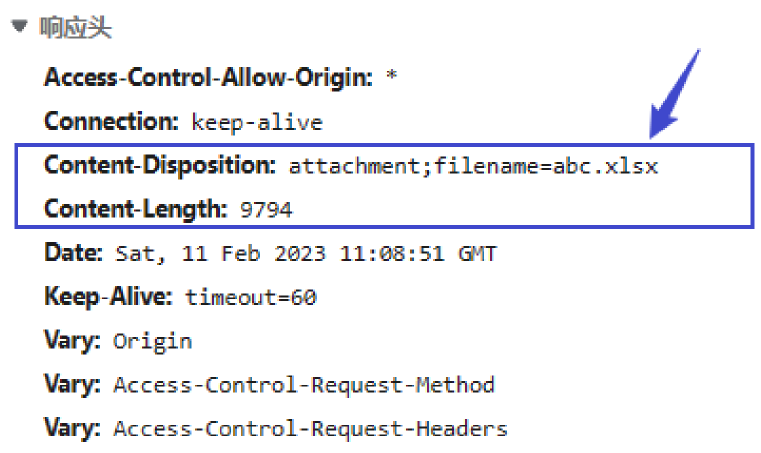
方式三：自定义文件写入输出流的相关属性

> 这里又可以细分为两种方式
>
> ***提示：HttpHeaders.CONTENT_DISPOSITION，spring-web中提供了HTTP请求属性枚举。***

1. ***方式一（使用Content-Disposition）***

- 文档
    - Content-Disposition - HTTP | MDN
    - Do I need Content-Type: application/octet-stream for file download?
- 好处：URL一旦被点击/访问，就会进行文件下载（调出浏览器的下载功能，这里会实时显示进度）
    - 

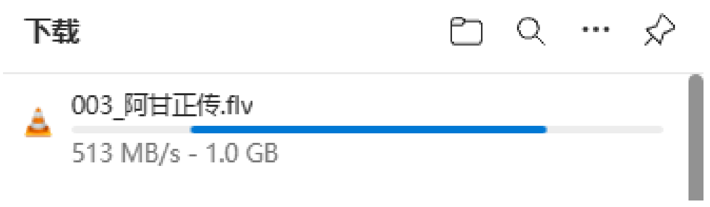
```
// 从后台获取数据
axios1.get('/images/download_s', {
    responseType: 'blob'
}).then(res => {
    const url = window.URL.createObjectURL(new Blob([res.data]))
    const link = document.createElement('a')
    link.href = url
    link.setAttribute('download', '003_阿甘正传.flv') // 下载文件的名称及文件类型后缀
    document.body.appendChild(link)
    link.click()
    document.body.removeChild(link) // 下载完成移除元素
    window.URL.revokeObjectURL(url) // 释放掉blob对象
});
```



▼ 响应头

```
Access-Control-Allow-Origin: *
Connection: keep-alive
Content-Disposition: attachment;filename=abc.xlsx
Content-Length: 9794
Date: Sat, 11 Feb 2023 11:08:51 GMT
Keep-Alive: timeout=60
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
```

- 坏处：文件名需要手动从Content–Disposition中提取，没办法精确的指定文件MIME类型

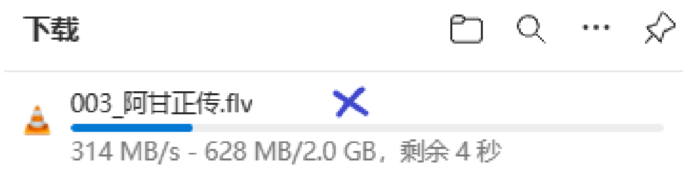
  

  下载

  003_阿甘正传.flv

  513 MB/s - 1.0 GB

  - 为什么没有显示文件具体大小呢？而是在循环滚动?
    - 响应头中添加Content–Length即可
      - *InputStream#available返回大小为int类型（最大2G）*

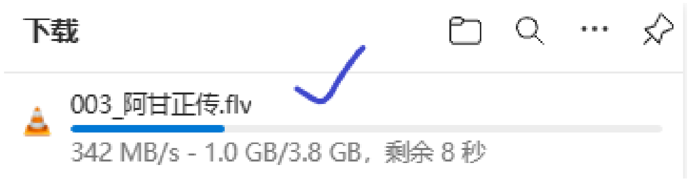
        

        下载

        003_阿甘正传.flv

        314 MB/s - 628 MB/2.0 GB，剩余 4 秒

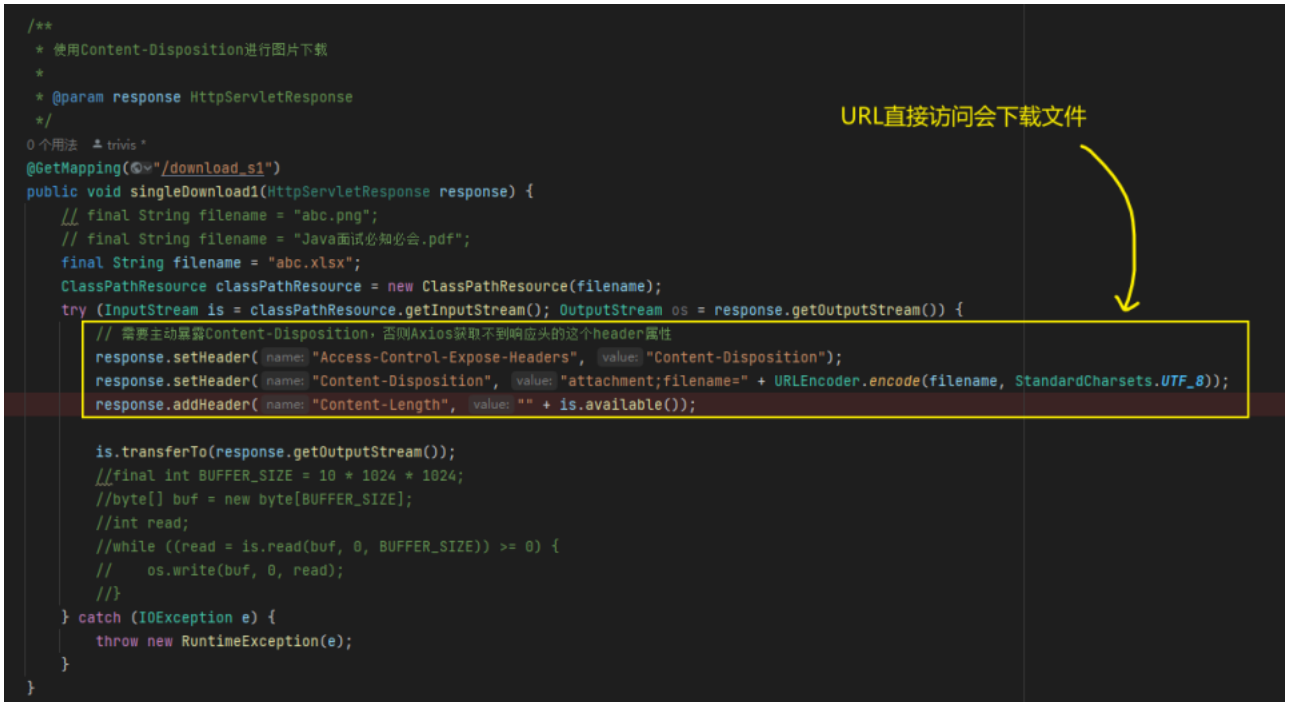
      - *classPathResource.getFile().length() 返回的类型为long（如果文件过大，请使用该属性获取真实文件大小）*

        下载

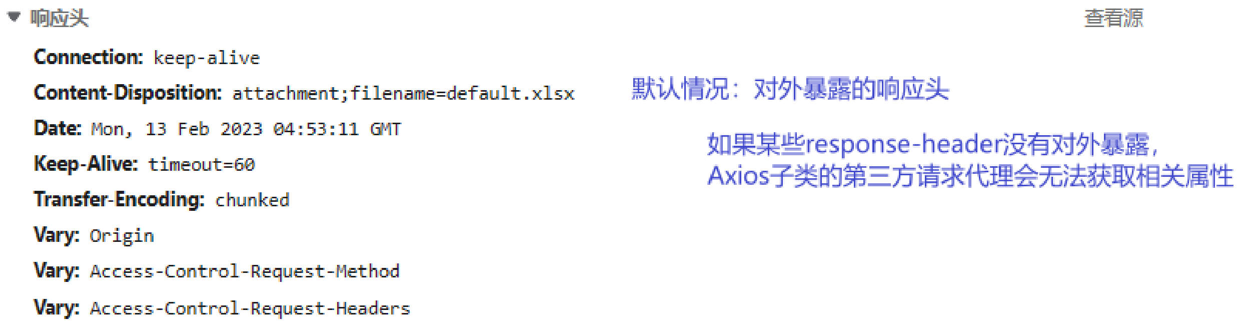
        003_阿甘正传.flv

        342 MB/s - 1.0 GB/3.8 GB，剩余 8 秒



```
/**
 * 使用Content-Disposition进行图片下载
 *
 * @param response HttpServletResponse
 */
0 个用法  trivis *
@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( name: "Access-Control-Expose-Headers",  value: "Content-Disposition");
        response.setHeader( name: "Content-Disposition",  value: "attachment;filename=" + URLEncoder.encode(filename, StandardCharsets.UTF_8));
        response.addHeader( name: "Content-Length",  value: "" + 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);
    }
}
```

URL直接访问会下载文件



▼ 响应头　　　　　　　　　　　　　　　　　　　　　　查看源

```
Connection: keep-alive
Content-Disposition: attachment;filename=default.xlsx
Date: Mon, 13 Feb 2023 04:53:11 GMT
Keep-Alive: timeout=60
Transfer-Encoding: chunked
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
```

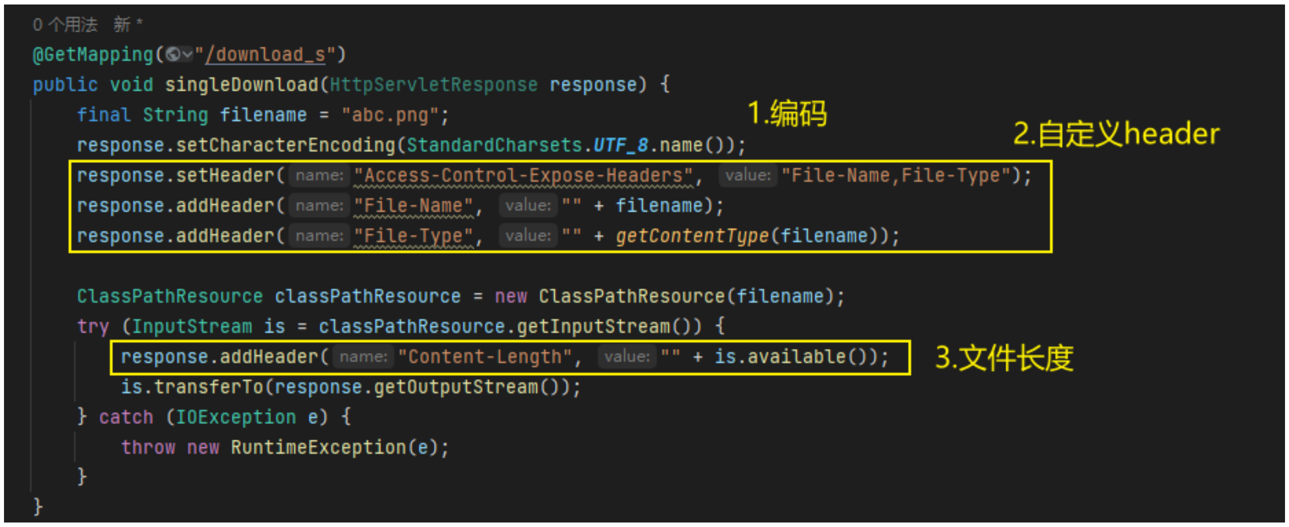
默认情况：对外暴露的响应头

如果某些response-header没有对外暴露，
Axios子类的第三方请求代理会无法获取相关属性

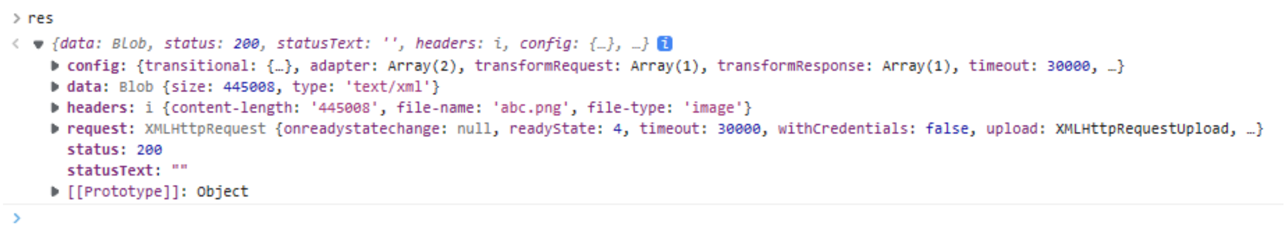
```
/**
 * 使用Content-Disposition进行图片下载
 *
 * @param response HttpServletResponse
 */
@GetMapping("/download_s1")
public void singleDownload1(HttpServletResponse response) {
    // final String filename = "abc.png";
    // final String filename = "Java面试必知必会.pdf";
    final String filename = "abc.xlsx";
    ClassPathResource classPathResource = new ClassPathResource(filename);
    try (InputStream is = classPathResource.getInputStream(); OutputStream os = response.getOutputStream()) {
        // 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);
        }
    }
```

2. ***方式二（自定义header）***



```
0 个用法  新 *
@GetMapping("/download_s")
public void singleDownload(HttpServletResponse response) {
    final String filename = "abc.png";
    response.setCharacterEncoding(StandardCharsets.UTF_8.name());
    response.setHeader( name: "Access-Control-Expose-Headers",  value: "File-Name,File-Type");
    response.addHeader( name: "File-Name",  value: "" + filename);
    response.addHeader( name: "File-Type",  value: "" + getContentType(filename));

    ClassPathResource classPathResource = new ClassPathResource(filename);
    try (InputStream is = classPathResource.getInputStream()) {
        response.addHeader( name: "Content-Length",  value: "" + is.available());
        is.transferTo(response.getOutputStream());
    } catch (IOException e) {
        throw new RuntimeException(e);
    }
}
```



```
> res
< ▾ {data: Blob, status: 200, statusText: '', headers: i, config: {…}, …} i
   ▸ config: {transitional: {…}, adapter: Array(2), transformRequest: Array(1), transformResponse: Array(1), timeout: 30000, …}
   ▸ data: Blob {size: 445008, type: 'text/xml'}
   ▸ headers: i {content-length: '445008', file-name: 'abc.png', file-type: 'image'}
   ▸ request: XMLHttpRequest {onreadystatechange: null, readyState: 4, timeout: 30000, withCredentials: false, upload: XMLHttpRequestUpload, …}
     status: 200
     statusText: ""
   ▸ [[Prototype]]: Object
>
```

```
    @GetMapping("/download_s")
    public void singleDownload(HttpServletResponse response) {
        final String filename = "abc.png";
        response.setCharacterEncoding(StandardCharsets.UTF_8.name());
        response.setHeader("Access-Control-Expose-Headers", "File-Name,File-Type");
        response.addHeader("File-Name", "" + filename);
        response.addHeader("File-Type", "" + getContentType(filename));

        ClassPathResource classPathResource = new ClassPathResource(filename);
        try (InputStream is = classPathResource.getInputStream()) {
            response.addHeader("Content-Length", "" + is.available());
            is.transferTo(response.getOutputStream());
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
```

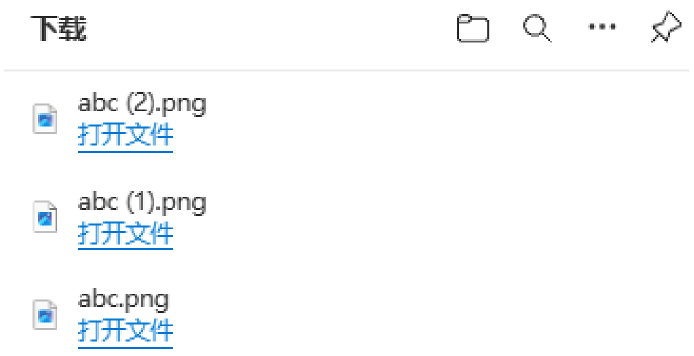
1. 前端预览

- 需要了解到文件具体MIME类型才能够正常预览（推荐使用自定义header，否则MIME类型需要前端手动指定）

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>文件预览与下载</title>
    <script src="./axios.min.js"></script>
    <script>
        function previewFile() {
            let axios1 = axios.create({
                baseURL: 'http://127.0.0.1:9091',
                timeout: 30000,
                // 默认是json类型数据
                headers: {
                    // 'Content-Type': 'application/x-www-form-urlencoded; charset=utf-8'
                    'Content-Type': 'application/json; charset=utf-8'
                }
            });
            // 从后台获取数据
            axios1.get('/images/download_s', {
                responseType: 'blob'
            }).then(res => {
                const url = window.URL.createObjectURL(new Blob([res.data], {type: "image/png"}))
                window.open(url);
                window.URL.revokeObjectURL(url) // 释放掉blob对象
            });
        }
    </script>
</head>
<body>

<button onclick="previewFile()">预览文件</button>
</body>
</html>
```

2. 前端下载

- 需要了解到文件具体MIME类型、文件名才能够正常预览（推荐使用自定义header，否则MIME类型和名称都需要前端手动指定，这是不合适的）


下载

abc (2).png
打开文件

abc (1).png
打开文件

abc.png
打开文件

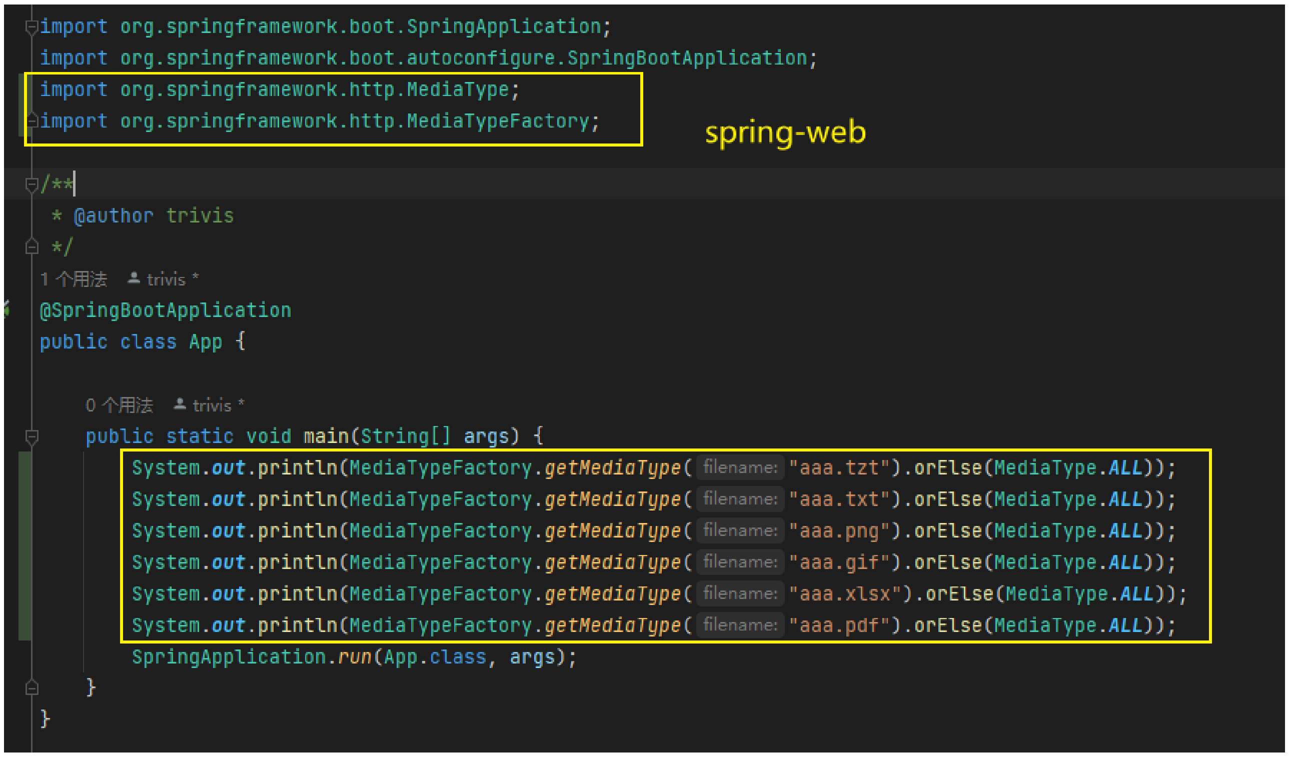
① 方式一：解析Content-DIsposition

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>文件预览与下载</title>
    <script src="./axios.min.js"></script>
    <script>
        function downloadFile() {
            let axios1 = axios.create({
                baseURL: ’http://127.0.0.1:9091’,
                timeout: 30000,
                // 默认是json类型数据
                headers: {
                    // ’Content-Type’: ’application/x-www-form-urlencoded; charset=utf-8’
                    ’Content-Type’: ’application/json; charset=utf-8’
                }
            });
            // 从后台获取数据
            axios1.get(’/images/download_s1’, {
                responseType: ’blob’
            }).then(res => {
                const filename = res.headers[’content-disposition’].split(”filename=”)[1];
                const url = window.URL.createObjectURL(new Blob([res.data]))
                const link = document.createElement(’a’)
                link.href = url
                link.setAttribute(’download’, filename ?? ”anonymous.file”) // 下载文件的名称及文件类型后缀
                document.body.appendChild(link)
                link.click()
                window.URL.revokeObjectURL(url) // 释放掉blob对象
            });
        }
    </script>
</head>
<body>

<button onclick=”downloadFile()”>下载文件</button>
</body>
</html>
```

② 方式二：解析自定义header

```
<!DOCTYPE html>
<html lang=”en”>
<head>
    <meta charset=”UTF-8”>
    <title>文件预览与下载</title>
    <script src=”./axios.min.js”></script>
    <script>
        function downloadFile() {
            let axios1 = axios.create({
                baseURL: ’http://127.0.0.1:9091’,
                timeout: 30000,
                // 默认是json类型数据
                headers: {
                    // ’Content-Type’: ’application/x-www-form-urlencoded; charset=utf-8’
                    ’Content-Type’: ’application/json; charset=utf-8’
                }
            });
            // 从后台获取数据
            axios1.get(’/images/download_s’, {
                responseType: ’blob’
            }).then(res => {
                let url = window.URL.createObjectURL(new Blob([res.data], {’type’: res.headers[’file-type’]}));
                let a = document.createElement(’a’);
                a.href = url;
                a.download = res.headers['file-name'];
                a.click();
                document.body.removeChild(a);
                window.URL.revokeObjectURL(url);
            });
        }
    </script>
</head>
<body>

<button onclick="downloadFile()">下载文件</button>
</body>
</html>
```

附录1：MIME类型（解析）



```java
import org.springframework.boot.SpringApplication;
import org.springframework.boot.autoconfigure.SpringBootApplication;
import org.springframework.http.MediaType;
import org.springframework.http.MediaTypeFactory;

/**
 * @author trivis
 */
@SpringBootApplication
public class App {

    public static void main(String[] args) {
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.tzt").orElse(MediaType.ALL));
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.txt").orElse(MediaType.ALL));
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.png").orElse(MediaType.ALL));
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.gif").orElse(MediaType.ALL));
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.xlsx").orElse(MediaType.ALL));
        System.out.println(MediaTypeFactory.getMediaType( filename: "aaa.pdf").orElse(MediaType.ALL));
        SpringApplication.run(App.class, args);
    }
}
```

附录2：application/octet-stream

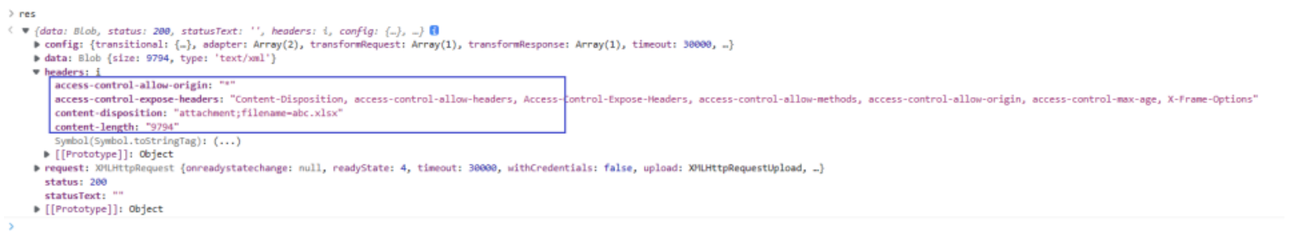
> 默认blob类型为text/xml，如果手动设置为application/octet-stream，浏览器则不会自动预览（图片、PDF）
>
> - 而是统一下载为一个URI名称的.file格式文件

附录3：Axios无法读取响应头headers的Content-Disposition

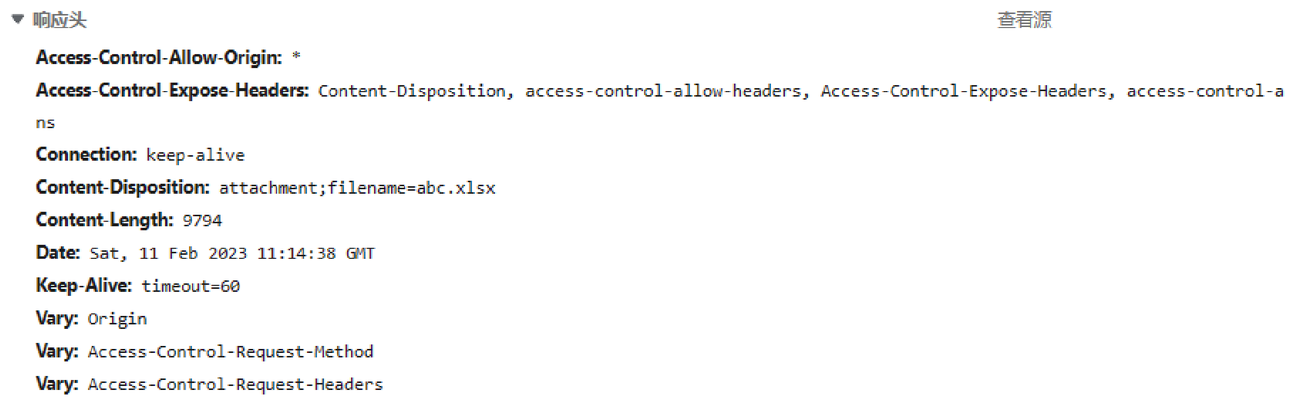
> 默认情况下，响应头的中headers并不会全部都暴露给外部（浏览器或其他网络请求程序）
>
> - Content-Length
>   -
> - Content-Type
>   -
> - Content-Disposition
>   -

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);
            }
        };
    }

}
```



```
> 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
      access-control-allow-origin: "*"
      access-control-expose-headers: "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"
      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
>
```

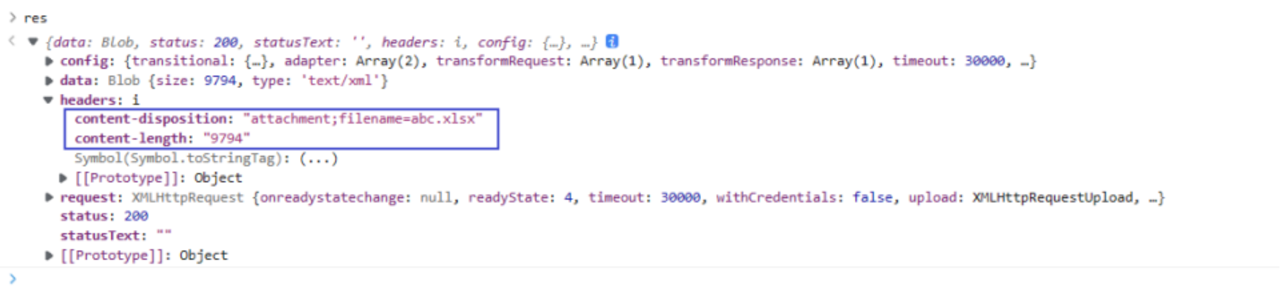


```
▼ 响应头                                                                       查看源
  Access-Control-Allow-Origin: *
  Access-Control-Expose-Headers: Content-Disposition, access-control-allow-headers, Access-Control-Expose-Headers, access-control-a
  ns
  Connection: keep-alive
  Content-Disposition: attachment;filename=abc.xlsx
  Content-Length: 9794
  Date: Sat, 11 Feb 2023 11:14:38 GMT
  Keep-Alive: timeout=60
  Vary: Origin
  Vary: Access-Control-Request-Method
  Vary: Access-Control-Request-Headers
```

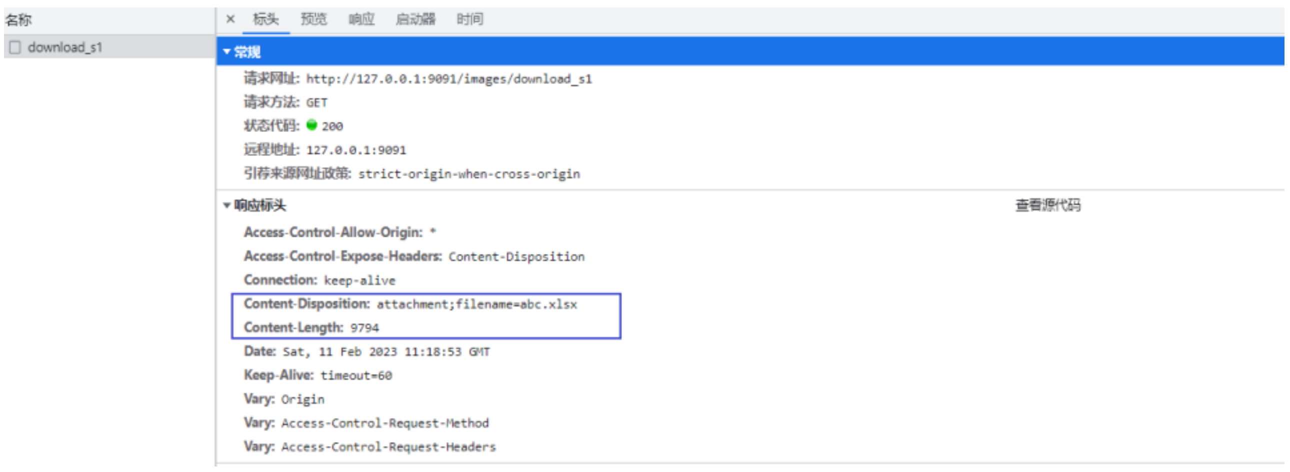
2. 方案二（针对指定接口进行控制）

```
// 单个
response.setHeader(”Access–Control–Expose–Headers”, ”Content–Disposition”);

// 多个
response.setHeader(”Access–Control–Expose–Headers”, ”aaa,bbb,ccc,ddd”);
```

```
    /**
     * 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);
        }
    }
```



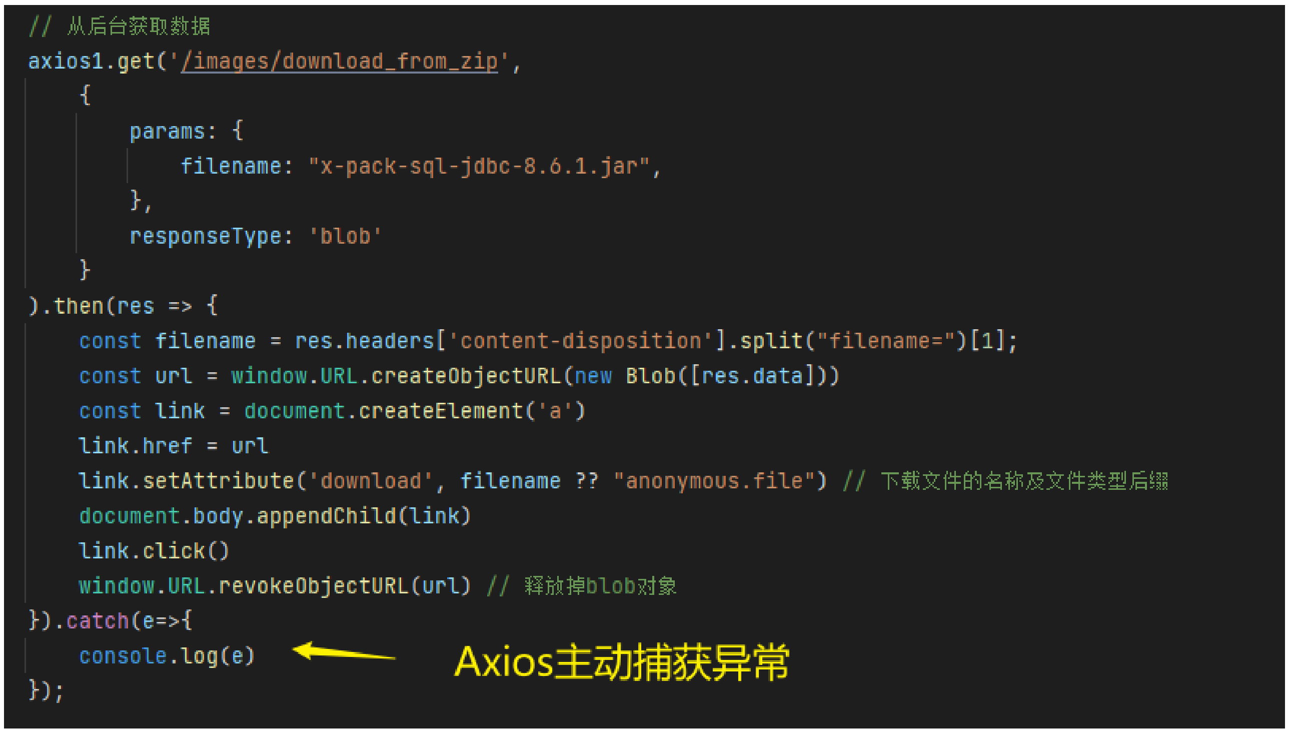
```
> res
< ▼ {data: Blob, status: 200, statusText: '', headers: i, config: {…}, …} ℹ
  ▶ config: {transitional: {…}, adapter: Array(2), transformRequest: Array(1), transformResponse: Array(1), timeout: 30000, …}
  ▶ data: Blob {size: 9794, type: 'text/xml'}
  ▼ headers: i
      content-disposition: "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
>
```



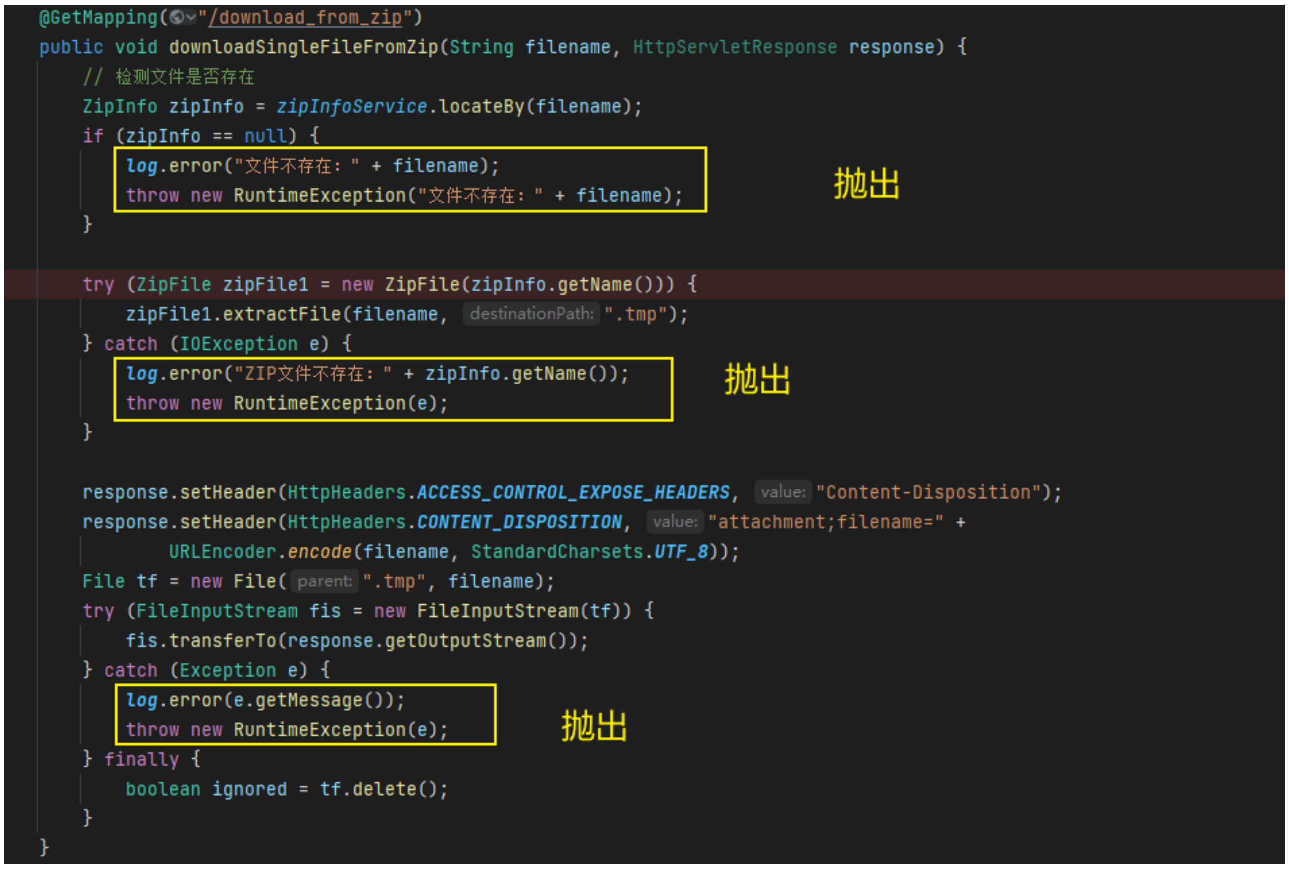
```
名称             × 标头 预览 响应 启动器 时间
download_s1
▼ 常规
    请求网址: http://127.0.0.1:9091/images/download_s1
    请求方法: GET
    状态代码: ● 200
    远程地址: 127.0.0.1:9091
    引荐来源网址政策: strict-origin-when-cross-origin
▼ 响应标头                                          查看源代码
    Access-Control-Allow-Origin: *
    Access-Control-Expose-Headers: Content-Disposition
    Connection: keep-alive
    Content-Disposition: attachment;filename=abc.xlsx
    Content-Length: 9794
    Date: Sat, 11 Feb 2023 11:18:53 GMT
    Keep-Alive: timeout=60
    Vary: Origin
    Vary: Access-Control-Request-Method
    Vary: Access-Control-Request-Headers
```

附录4：下载文件接口的异常抛出?

1. 直接throw异常即可，前端即可获取500
2. 如果正常触发下载，前端立刻得到200响应



```
// 从后台获取数据
axios1.get('/images/download_from_zip',
    {
        params: {
            filename: "x-pack-sql-jdbc-8.6.1.jar",
        },
        responseType: 'blob'
    }
).then(res => {
    const filename = res.headers['content-disposition'].split("filename=")[1];
    const url = window.URL.createObjectURL(new Blob([res.data]))
    const link = document.createElement('a')
    link.href = url
    link.setAttribute('download', filename ?? "anonymous.file") // 下载文件的名称及文件类型后缀
    document.body.appendChild(link)
    link.click()
    window.URL.revokeObjectURL(url) // 释放掉blob对象
}).catch(e=>{
    console.log(e)
});
```



```
@GetMapping("/download_from_zip")
public void downloadSingleFileFromZip(String filename, HttpServletResponse response) {
    // 检测文件是否存在
    ZipInfo zipInfo = zipInfoService.locateBy(filename);
    if (zipInfo == null) {
        log.error("文件不存在: " + filename);
        throw new RuntimeException("文件不存在: " + filename);
    }

    try (ZipFile zipFile1 = new ZipFile(zipInfo.getName())) {
        zipFile1.extractFile(filename, destinationPath: ".tmp");
    } catch (IOException e) {
        log.error("ZIP文件不存在: " + zipInfo.getName());
        throw new RuntimeException(e);
    }

    response.setHeader(HttpHeaders.ACCESS_CONTROL_EXPOSE_HEADERS, value: "Content-Disposition");
    response.setHeader(HttpHeaders.CONTENT_DISPOSITION, value: "attachment;filename=" +
            URLEncoder.encode(filename, StandardCharsets.UTF_8));
    File tf = new File( parent: ".tmp", filename);
    try (FileInputStream fis = new FileInputStream(tf)) {
        fis.transferTo(response.getOutputStream());
    } catch (Exception e) {
        log.error(e.getMessage());
        throw new RuntimeException(e);
    } finally {
        boolean ignored = tf.delete();
    }
}
```