03_SpringBoot多张图片上传与下载

时间：2023年2月12日13:21:17

一、多图片上传

> 直接使用MultipartFile[]

```
/**
 * 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
 */
@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();
        ) {
            is.transferTo(os);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
}
```

二、多图片下载

> ① 发送多个单图片下载请求
>
> ② 放在压缩包内返回（推荐）
>
> - 不压缩
> - 压缩　***注意响应中的文件大小！***
> - 加密压缩

虽然JDK中也存在ZipInputStream+ZipOutputStream，但是功能有限，推荐使用第三方封装包：

官方文档：https://github.com/srikanth-lingala/zip4j

```
<!-- zip4j -->
<dependency>
    <groupId>net.lingala.zip4j</groupId>
    <artifactId>zip4j</artifactId>
    <version>2.11.4</version>
</dependency>
```

1. 封装工具类

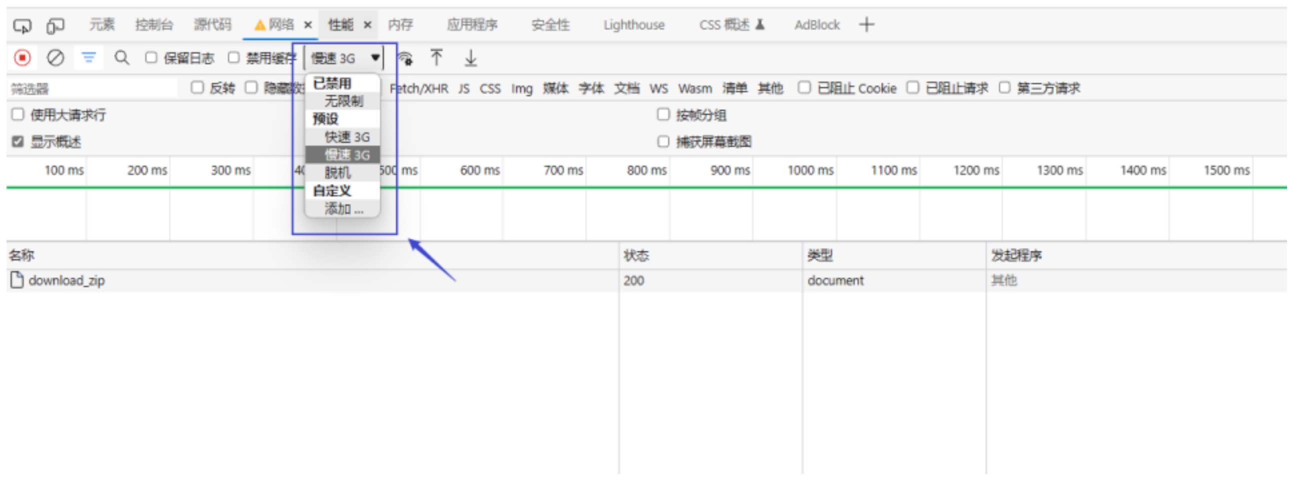
```
import net.lingala.zip4j.io.outputstream.ZipOutputStream;
import net.lingala.zip4j.model.ZipParameters;
import net.lingala.zip4j.model.enums.AesKeyStrength;
import net.lingala.zip4j.model.enums.CompressionMethod;
import net.lingala.zip4j.model.enums.EncryptionMethod;
import org.springframework.web.multipart.MultipartFile;

import java.io.*;
import java.util.List;

/**
 * Zip压缩工具类
 *
 * @since 1.9
 * @author trivis
 */
public class ZipUtils {
    private ZipUtils() {
    }

    private ZipUtils(ZipParameters zp, char[] p) {
        zipParameters = zp;
        password = p;
    }

    private ZipParameters zipParameters;
    private char[] password;

    public static ZipUtils init() {
        return initStore();
    }

    public static ZipUtils init(String password){
        return initFull(CompressionMethod.STORE, true,
                EncryptionMethod.ZIP_STANDARD, null, password);
    }

    public static ZipUtils initStore() {
        return initFull(CompressionMethod.STORE, false,
                null, null, null);
    }

    public static ZipUtils initDeflate() {
        return initFull(CompressionMethod.DEFLATE, false,
                null, null, null);
    }

    public static ZipUtils initFull(CompressionMethod compressionMethod, boolean encrypt,
                                    EncryptionMethod encryptionMethod, AesKeyStrength aesKeyStrength, String password) {
        if (encrypt && password == null) {
            throw new IllegalArgumentException("encrypt is true, but you not set password!");
        }
        return new ZipUtils(buildZipParameters(compressionMethod, encrypt, encryptionMethod, aesKeyStrength),
                password != null ? password.toCharArray() : null);
    }
```

```
public void to(OutputStream outputStream, List<File> fileList) throws IOException {
    try (ZipOutputStream zos = initializeZipOutputStream(outputStream, zipParameters.isEncryptFiles(), password)) {
        for (File fileToAdd : fileList) {

            // Entry size has to be set if you want to add entries of STORE compression method (no compression)
            // This is not required for deflate compression
            if (zipParameters.getCompressionMethod() == CompressionMethod.STORE) {
                zipParameters.setEntrySize(fileToAdd.length());
            }

            zipParameters.setFileNameInZip(fileToAdd.getName());
            zos.putNextEntry(zipParameters);

            try (InputStream inputStream = new FileInputStream(fileToAdd)) {
                inputStream.transferTo(zos);
            }
            zos.closeEntry();
        }
    }
}
public void to(OutputStream outputStream, MultipartFile[] files) throws IOException {
    try (ZipOutputStream zos = initializeZipOutputStream(outputStream, zipParameters.isEncryptFiles(), password)) {
        for (MultipartFile file : files) {

            // Entry size has to be set if you want to add entries of STORE compression method (no compression)
            // This is not required for deflate compression
            if (zipParameters.getCompressionMethod() == CompressionMethod.STORE) {
                zipParameters.setEntrySize(file.getSize());
            }

            zipParameters.setFileNameInZip(file.getOriginalFilename());
            zos.putNextEntry(zipParameters);

            try (InputStream inputStream = file.getInputStream()) {
                inputStream.transferTo(zos);
            }
            zos.closeEntry();
        }
    }
}
public long to(File outputZipFile, List<File> fileList) throws IOException {
    try (FileOutputStream fos = new FileOutputStream(outputZipFile)) {
        to(fos, fileList);
    }
    return outputZipFile.length();
}
public long to(File outputZipFile, MultipartFile[] files) throws IOException {
    try (FileOutputStream fos = new FileOutputStream(outputZipFile)) {
        to(fos, files);
    }
    return outputZipFile.length();
}


/**
 * 使用OutputStream初始化ZipOutputStream
 *
```

```
     * @param os
     * @param encrypt
     * @param password
     * @return
     * @throws IOException
     */
    private static ZipOutputStream initializeZipOutputStream(OutputStream os, boolean encrypt, char[] password)
            throws IOException {
        if (encrypt) {
            return new ZipOutputStream(os, password);
        }
        return new ZipOutputStream(os);
    }

    /**
     * 构建zip参数
     * 1. AesKeyStrength在EncryptionMethod.AES时生效，EncryptionMethod为其他（ZIP_STANDARD）时，该参数设置为null即可
     *
     * @param compressionMethod
     * @param encrypt
     * @param encryptionMethod
     * @param aesKeyStrength
     * @return
     */
    private static ZipParameters buildZipParameters(CompressionMethod compressionMethod, boolean encrypt,
                                                    EncryptionMethod encryptionMethod, AesKeyStrength aesKeyStrength) {
        compressionMethod = compressionMethod != null ? compressionMethod : CompressionMethod.STORE;
        // encrypt=true时，EncryptionMethod.NONE将自动替换为EncryptionMethod.ZIP_STANDARD
        if (encryptionMethod == null || EncryptionMethod.NONE.equals(encryptionMethod)) {
            if (encrypt) {
                encryptionMethod = EncryptionMethod.ZIP_STANDARD;
            } else {
                encryptionMethod = EncryptionMethod.NONE;
            }
        }
        aesKeyStrength = aesKeyStrength != null ? aesKeyStrength : AesKeyStrength.KEY_STRENGTH_128;
        ZipParameters zipParameters = new ZipParameters();
        zipParameters.setCompressionMethod(compressionMethod);
        zipParameters.setEncryptFiles(encrypt);
        zipParameters.setEncryptionMethod(encryptionMethod);
        zipParameters.setAesKeyStrength(aesKeyStrength);
        return zipParameters;
    }
}
```

2. 接口测试

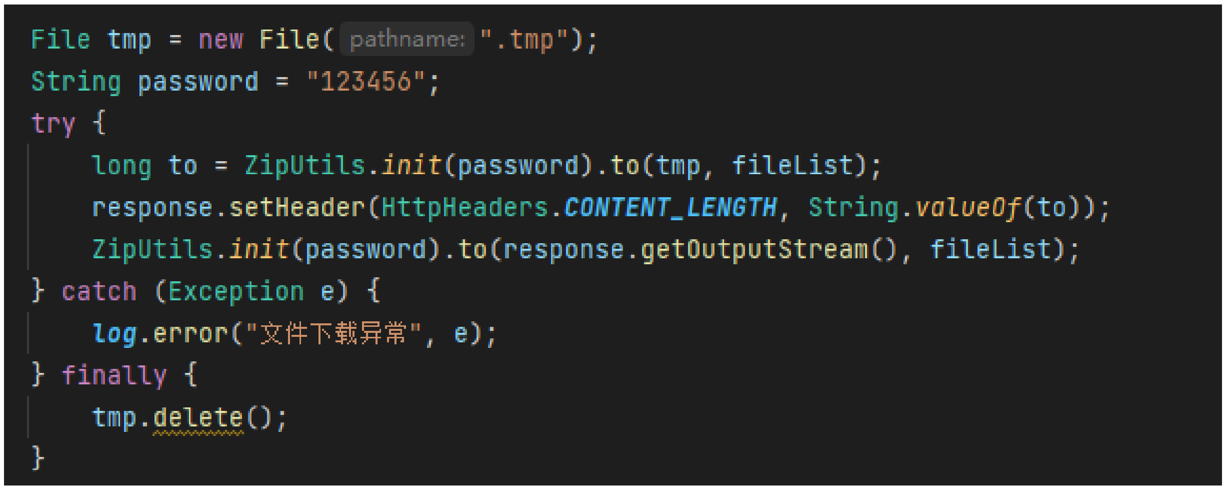
> 调试时，可以设置浏览器下载速度


元素 控制台 源代码 网络 性能 内存 应用程序 安全性 Lighthouse CSS 概述 AdBlock
保留日志 禁用缓存 慢速 3G
已禁用
无限制
预设
快速 3G
慢速 3G
脱机
自定义
添加 ...
筛选器 反转 隐藏数据 Fetch/XHR JS CSS Img 媒体 字体 文档 WS Wasm 清单 其他 已阻止 Cookie 已阻止请求 第三方请求
使用大请求行 按帧分组
显示概述 捕获屏幕截图
100 ms 200 ms 300 ms 500 ms 600 ms 700 ms 800 ms 900 ms 1000 ms 1100 ms 1200 ms 1300 ms 1400 ms 1500 ms
名称 状态 类型 发起程序
download_zip 200 document 其他

•

```
/**
 * 批量导出图片到zip
 *
 * @param response HttpServletResponse
 */
@GetMapping(”/download_zip”)
public void zipDownload(HttpServletResponse response) {
    final String zipFilename = ”files.zip”;
    List<String> filenames = new ArrayList<>();
    List<File> fileList = new ArrayList<>();
    filenames.add(”abc.png”);
    filenames.add(”Java面试必知必会.pdf”);
    filenames.add(”abc.xlsx”);
    filenames.add(”abc.xlsx”);
    filenames.add(”abc.xlsx”);
    for (String filename : filenames) {
        try {
            File file = new ClassPathResource(filename).getFile();
            fileList.add(file);
        } catch (IOException e) {
            // todo log
            throw new RuntimeException(e);
        }
    }
    response.setHeader(HttpHeaders.ACCESS_CONTROL_EXPOSE_HEADERS, ”Content-Disposition”);
    response.setHeader(HttpHeaders.CONTENT_DISPOSITION, ”attachment;filename=” +
            URLEncoder.encode(zipFilename, StandardCharsets.UTF_8));

    String password = ”123456”;
    try {
        // ZipUtils.init().to(response.getOutputStream(), fileList);
        ZipUtils.init(password).to(response.getOutputStream(), fileList);
    } catch (Exception e) {
        // todo log
        log.error(”文件下载异常”, e);
        // 重置reset，使得return返回数据生效（忽略response中下载相关配置）
        // 正常不会这么做，必须确保下载正常，下载接口不提供返回值。如果异常，需要手动排查。
        // response.reset();
        // return ”error”;
    }
}
```

3. 设置Content-Length

> 直接将数据写入OutputStream，会使得前端无法准确获取Content-Length，以致于无法确定具体下载进度。
>
> - 解决方案一：写入本地缓存文件，读取缓存文件大小，传输该缓存文件，删除缓存文件。（***大文件时，注意写入本地文件的时间***）
>   - 
>
> ```java
> File tmp = new File( pathname: ".tmp");
> String password = "123456";
> try {
>     long to = ZipUtils.init(password).to(tmp, fileList);
>     response.setHeader(HttpHeaders.CONTENT_LENGTH, String.valueOf(to));
>     ZipUtils.init(password).to(response.getOutputStream(), fileList);
> } catch (Exception e) {
>     log.error("文件下载异常", e);
> } finally {
>     tmp.delete();
> }
> ```
>
> - 解决方案二：***不设置Content-Length***，对于ZIP压缩文件的下载；
>   - 如果直接下载ZIP，是可以指定Content-Length的；
>   - 如果要在逻辑中实时压缩一堆文件，对于是否要设置Content-Length则需要慎重考虑，当然也可以设置（缓存zip在机器中，获取大小，再次传输，时间+++）。

三、解析ZIP包中的数据

1. 流操作

> 获取MultipartFile，解析
>
> 1. *获取ZIP文件输入流，包装为ZipInputStream*
> 2. *读取至OutputStream*

```
/**
 * 解析ZIP（压缩包内不包含目录）
 * 1. 获取ZIP文件输入流，包装为ZipInputStream
 * 2. 读取至OutputStream
 *
 * @param zipFile zip格式压缩文件
 */
@PostMapping(”/upload_zip”)
public String imageZipImport(@RequestPart(”file”) MultipartFile zipFile) {
    String dirNamm = zipFile.getOriginalFilename();
    dirNamm = dirNamm == null ? ”_files” : dirNamm.substring(0, dirNamm.indexOf(”.”));
    File dirFile = new File(dirNamm);
    boolean ignored = dirFile.mkdir();
```

```
        final String password = "123456";
        LocalFileHeader localFileHeader;
        try (InputStream inputStream = zipFile.getInputStream();
             ZipInputStream zipInputStream = new ZipInputStream(inputStream, password.toCharArray())) {
            while ((localFileHeader = zipInputStream.getNextEntry()) != null) {
                File extractedFile = new File(dirFile, localFileHeader.getFileName());
                try (OutputStream outputStream = new FileOutputStream(extractedFile)) {
                    zipInputStream.transferTo(outputStream);
                }
            }
        } catch (IOException e) {
            log.error(e.getMessage());
            return "error";
        }

        return "ok";
    }
```

2. 解析到文件

> https://github.com/srikanth-lingala/zip4j

Extracting all files from a zip

```
new ZipFile("filename.zip").extractAll("/destination_directory");
```

Extracting all files from a password protected zip

```
new ZipFile("filename.zip", "password".toCharArray()).extractAll("/destination_directory");
```

Extracting a single file from zip

```
new ZipFile("filename.zip").extractFile("fileNameInZip.txt", "/destination_directory");
```

Extracting a folder from zip (since v2.6.0)

```
new ZipFile("filename.zip").extractFile("folderNameInZip/", "/destination_directory");
```

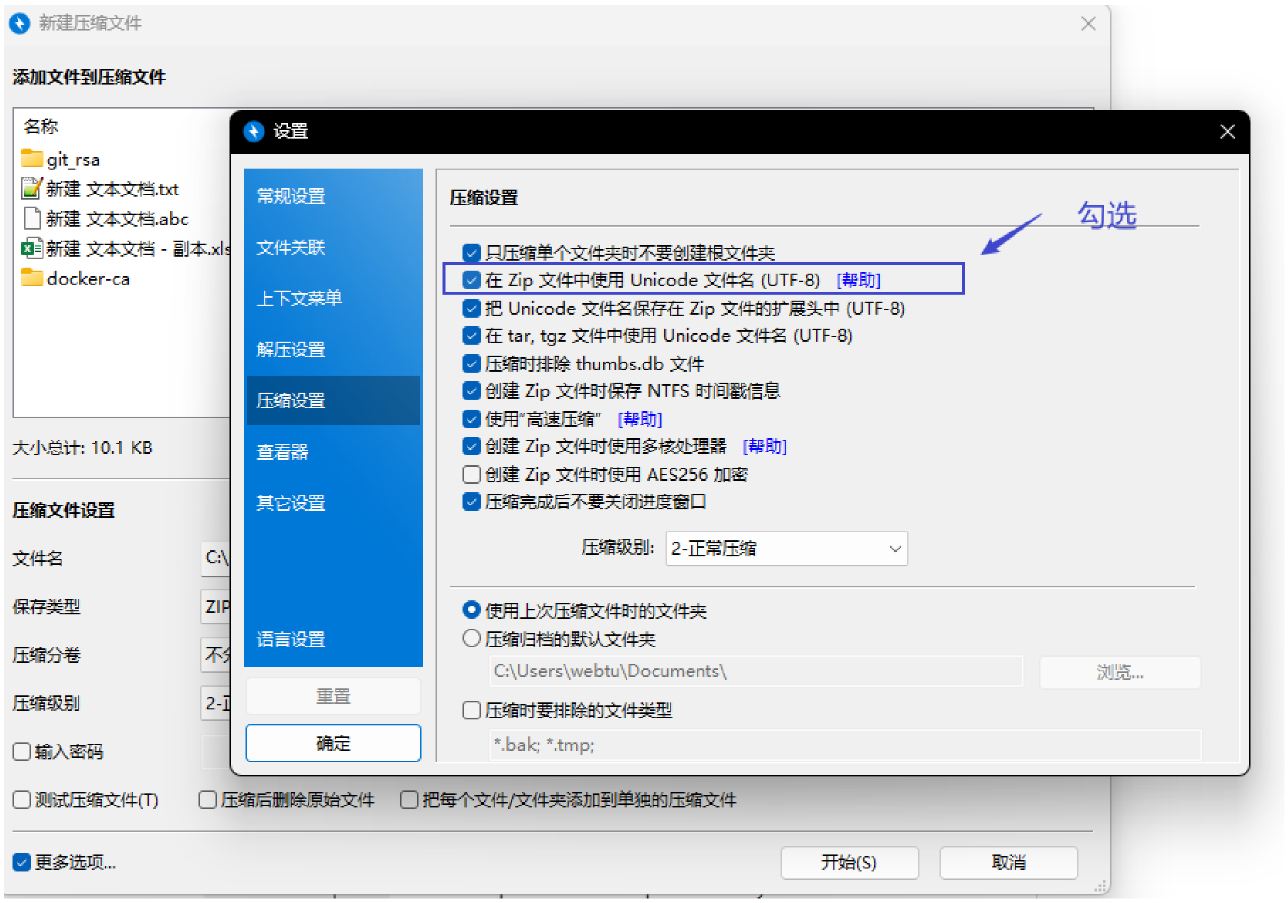
Extracting a single file from zip which is password protected

```
new ZipFile("filename.zip", "password".toCharArray()).extractFile("fileNameInZip.txt", "/destination_director
```

```
    /**
     * 解析ZIP
     * 1. 提取所有文件到指定目录
     *
     * @param zipFile zip格式压缩文件
     */
    @PostMapping(”/upload_zip”)
    public String imageZipImport(@RequestPart(”file”) MultipartFile zipFile) {
        String uuid = UUID.randomUUID().toString().replaceAll(”–”, ””);
        try(FileOutputStream fos = new FileOutputStream(”.” + uuid)) {
            zipFile.getInputStream().transferTo(fos);
        }catch (IOException e) {
            log.error(e.getMessage());
            return ”false”;
        }

        final String p = ”123456”;
        try(ZipFile zipFile1 = new ZipFile(”.” + uuid, p.toCharArray())){
            // /destination_directory –> C盘
            // destination_directory  –> 当前目录
            zipFile1.extractAll(”destination_directory”);
        } catch (IOException e) {
            log.error(e.getMessage());
            return ”false”;
        } finally {
            boolean ignored = new File(”.” + uuid).delete();
        }

        return ”ok”;
    }
```

3. 压缩包内文件名中文乱码?

> 如果使用BandiZip压缩，请确保压缩包压缩配置正确。



新建压缩文件

添加文件到压缩文件

名称
git_rsa
新建 文本文档.txt
新建 文本文档.abc
新建 文本文档 - 副本.xls
docker-ca

大小总计: 10.1 KB

压缩文件设置

文件名 C:\

保存类型 ZIP

压缩分卷 不分

压缩级别 2-正

☐ 输入密码

☐ 测试压缩文件(T) ☐ 压缩后删除原始文件 ☐ 把每个文件/文件夹添加到单独的压缩文件

☑ 更多选项...

设置

压缩设置

勾选

- ☑ 只压缩单个文件夹时不要创建根文件夹
- ☑ 在 Zip 文件中使用 Unicode 文件名 (UTF-8) [帮助]
- ☑ 把 Unicode 文件名保存在 Zip 文件的扩展头中 (UTF-8)
- ☑ 在 tar, tgz 文件中使用 Unicode 文件名 (UTF-8)
- ☑ 压缩时排除 thumbs.db 文件
- ☑ 创建 Zip 文件时保存 NTFS 时间戳信息
- ☑ 使用"高速压缩" [帮助]
- ☑ 创建 Zip 文件时使用多核处理器 [帮助]
- ☐ 创建 Zip 文件时使用 AES256 加密
- ☑ 压缩完成后不要关闭进度窗口

压缩级别: 2-正常压缩

- ● 使用上次压缩文件时的文件夹
- ○ 压缩归档的默认文件夹

C:\Users\webtu\Documents\

☐ 压缩时要排除的文件类型

*.bak; *.tmp;