04_SpringBoot上传文件压缩存储

时间：2023年02月13日 12:56:38

需求：

- 前端上传3个文件，后端压缩上传的文件（存储在本地）
- 为了下载时可以快速定位该文件，需要存储每个文件的名称+大小信息（元信息/索引信息）

一、引入第三方zip工具包

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

```
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;
    }
}
```

二、编写接口

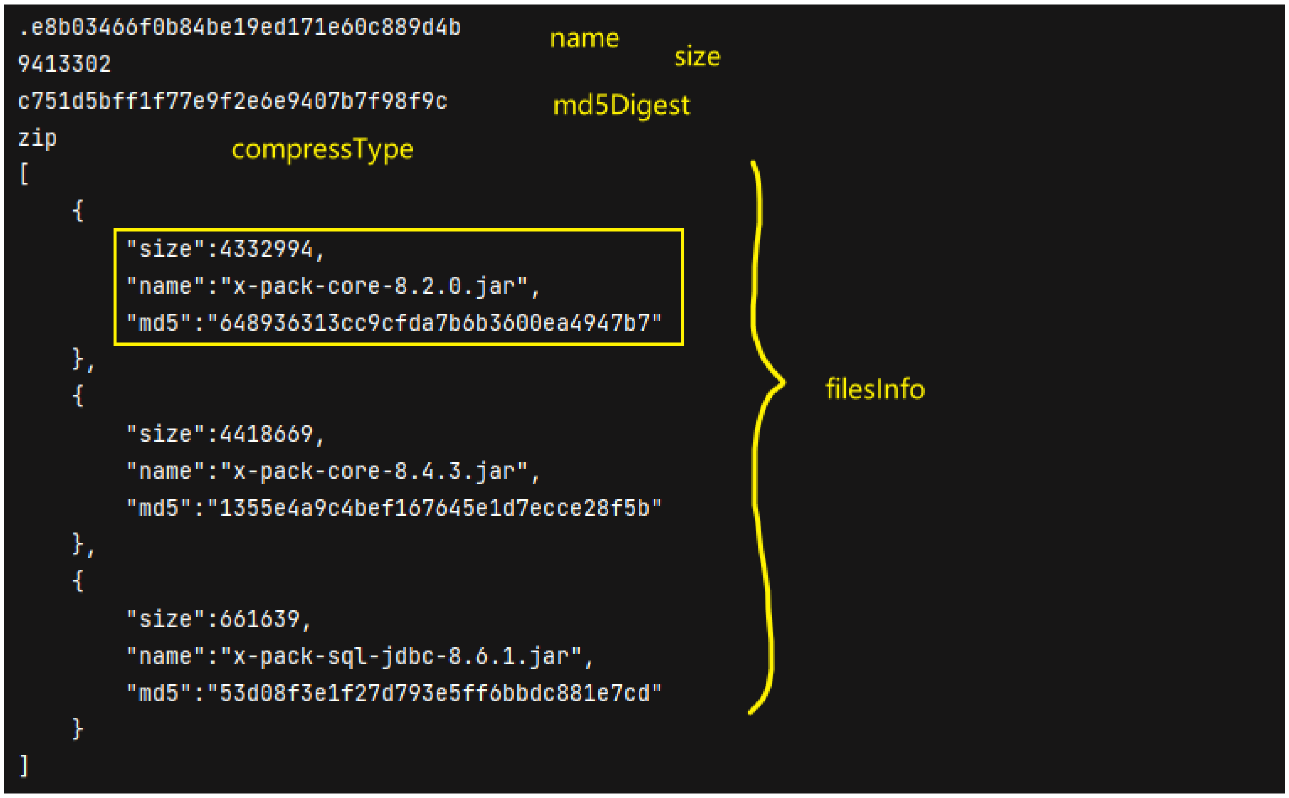
```
    /**
     * 将上传的文件封装为zip，存储在本地
     * 1. MultipartFile可以获取输入文件流，如何将流写入zip输出流
     *
     * @param zipFiles MultipartFile Array
     * @return ”ok”/”error”
     */
    @PostMapping(”/upload”)
    public String imageImportZip(@RequestPart(”files”) MultipartFile[] zipFiles) {
        File file = new File(”default.zip”);
        try {
            ZipUtils.init().to(file, zipFiles);
        } catch (IOException e) {
            log.error(e.getMessage());
            return ”error”;
        }

        return ”ok”;
    }
```

三、元数据如何存储?

元数据十分重要，请确保元数据存储服务做了充足的容灾备份。

> 存储是为了更快的定位到文件所在的压缩文件：
>
> - 通过文件的某个属性，定位到文件所在的压缩包，解压指定文件返回

1. 规定数据存储格式



```
.e8b03466f0b84be19ed171e60c889d4b
9413302
c751d5bff1f77e9f2e6e9407b7f98f9c
zip
[
    {
        "size":4332994,
        "name":"x-pack-core-8.2.0.jar",
        "md5":"648936313cc9cfda7b6b3600ea4947b7"
    },
    {
        "size":4418669,
        "name":"x-pack-core-8.4.3.jar",
        "md5":"1355e4a9c4bef167645e1d7ecce28f5b"
    },
    {
        "size":661639,
        "name":"x-pack-sql-jdbc-8.6.1.jar",
        "md5":"53d08f3e1f27d793e5ff6bbdc881e7cd"
    }
]
```

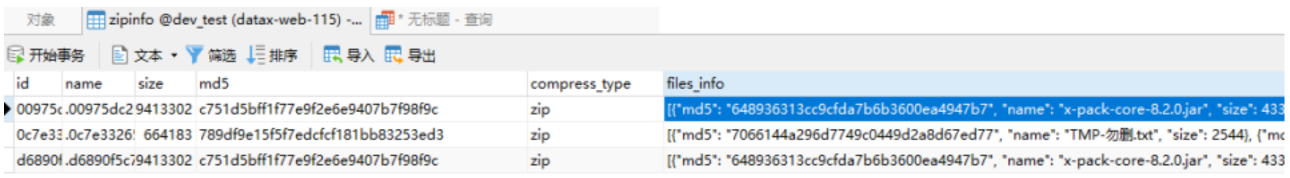
```
/**
 * 1. 将上传的文件封装为zip，存储在本地
 * 2. 将数据源元信息写入MySQL
 *
 * @param zipFiles MultipartFile Array
 * @return ”ok”/”error”
 */
@PostMapping(”/upload”)
public String imageImportZip(@RequestPart(”files”) MultipartFile[] zipFiles) {
    final String compressType = ”zip”;
    long size = 0;
    String md5Digest = ””;
    StringBuilder md5DigestTemp = new StringBuilder();
    String uuid = UUID.randomUUID().toString().replaceAll(”–”, ””);
    String name = ”.” + uuid;
    // 获取文件MD5
    List<Map<String, Object>> filesInfo = new ArrayList<>();
    for (MultipartFile zipFile : zipFiles) {
        try (InputStream is = zipFile.getInputStream()) {
            long si = zipFile.getSize();
            String md5i = DigestUtils.md5DigestAsHex(is);
            size += si;
            md5DigestTemp.append(md5i);
            Map<String, Object> t = new HashMap<>();
            t.put(”name”, zipFile.getOriginalFilename());
            t.put(”size”, si);
            t.put(”md5”, md5i);
            filesInfo.add(t);
        } catch (IOException e) {
            throw new RuntimeException(e);
        }
    }
```

```
        md5Digest = DigestUtils.md5DigestAsHex(md5DigestTemp.toString().getBytes(StandardCharsets.UTF_8));

        // name、size、md5Digest、compressType、filesInfo
        System.out.println(name);
        System.out.println(size);
        System.out.println(md5Digest);
        System.out.println(compressType);
        System.out.println(JSONObject.toJSONString(filesInfo, JSONWriter.Feature.PrettyFormat));

        ZipInfo zipInfo = new ZipInfo(uuid, name, size, md5Digest, compressType, JSONObject.toJSONString(filesInfo));
        if(zipInfoService.save(zipInfo)) {
            File file = new File(name);
            try {
                ZipUtils.init().to(file, zipFiles);
            } catch (IOException e) {
                log.error(e.getMessage());
                return "error";
            }
        }else{
            log.error("insert data error!");
            return "error";
        }

        return "ok";
    }
```

2. 数据存储在哪里?

> MySQL 5.7.8+ 已经支持JSON数据类型
>
> - https://dev.mysql.com/doc/refman/5.7/en/json.html
>   -
>   As of MySQL 5.7.8, MySQL supports a native JSON data type defined by RFC 7159 that enables efficient access to data in JSON (JavaScript Object Notation) doc
>
> MySQL-JSON如何使用?
>
> - https://dev.mysql.com/doc/refman/5.7/en/json-search-functions.html
> - https://www.yiibai.com/mysql/json.html

（数据库中数据存储格式）



id	name	size	md5	compress_type	files_info
00975c	.00975dc2	9413302	c751d5bff1f77e9f2e6e9407b7f98f9c	zip	[{"md5": "648936313cc9cfda7b6b3600ea4947b7", "name": "x-pack-core-8.2.0.jar", "size": 433
0c7e33	.0c7e3326	664183	789df9e15f5f7edcfcf181bb83253ed3	zip	[{"md5": "7066144a296d7749c0449d2a8d67ed77", "name": "TMP-勿删.txt", "size": 2544}, {"md
d6890f	.d6890f5c7	9413302	c751d5bff1f77e9f2e6e9407b7f98f9c	zip	[{"md5": "648936313cc9cfda7b6b3600ea4947b7", "name": "x-pack-core-8.2.0.jar", "size": 433

3. 如何查询MySQL中的JSON数据?

- 函数
- JSONPath

```
// 利用函数，精准匹配
SELECT * FROM zipinfo WHERE JSON_CONTAINS(files_info,JSON_OBJECT('name', "x-pack-sql-jdbc-8.6.1.jar"))
ORDER BY size ASC LIMIT 100;

// 利用JSONPath，模糊匹配/精准匹配
SELECT * FROM zipinfo WHERE INSTR(files_info->>'$[*].name', 'x-pack')>0 ORDER BY size ASC LIMIT 100;
```

4. 编写相关接口

> 1. GET请求，传入文件名，精准匹配找到文件所在的压缩包；
> 2. 从压缩包中解压指定的文件，返回。
>
> 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", "/destinatio_director
```

```
    /**
     * 关于下载文件时的异常?
     * 1. 直接throw异常即可，前端即可获取500
     * 2. 如果触发下载，则正常返回200
     *
     * @param filename 文件名
     * @param response HttpServletResponse
     */
    @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, ”.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, ”Content-Disposition”);
        response.setHeader(HttpHeaders.CONTENT_DISPOSITION, ”attachment;filename=” +
                URLEncoder.encode(filename, StandardCharsets.UTF_8));
        File tf = new File(”.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();
        }
    }
```

```
<?xml version=”1.0” encoding=”UTF-8”?>
<!DOCTYPE mapper PUBLIC ”-//mybatis.org//DTD Mapper 3.0//EN” ”http://mybatis.org/dtd/mybatis-3-mapper.dtd”
>
<mapper namespace=”com.abc.business.images.mapper.ZipInfoMapper”>

    <select id=”_locateBy” resultType=”com.abc.business.images.domain.entity.zip.ZipInfo”>
        SELECT * FROM zipinfo WHERE INSTR(files_info->>’$[*].name’, #{filename})>0 ORDER BY size ASC LIMIT 1;
    </select>

</mapper>
```

```
http://localhost:9091/images/download_from_zip?filename=x-pack-sql-jdbc-8.6.1.jar
```

今天			
x-pack-sql-jdbc-8.6.1 (1).jar	2023/2/13 16:42	Executable Jar File	647 KB
x-pack-sql-jdbc-8.6.1.jar	2023/2/13 16:39	Executable Jar File	647 KB

（使用Axios下载，在Axios中使用catch捕获下载异常）

```
<!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_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)
            });
        }
    </script>
</head>
<body>

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