01_SpringBoot实现图片上传

时间：2023年02月10日 14:40:06

方案一：将图片使用base64编码为字符串

> 请求中携带：
>
> - filename
> - base64编码的图片数据

```java
@RestController
@RequestMapping("/images")
public class ImageController {

    0 个用法  新 *
    @PostMapping("/upload")
    public void multiImageImport(@RequestBody ImageUploadDTO imageUploadInfo) {

        try (BufferedOutputStream bo = new BufferedOutputStream(new FileOutputStream( name: "backup-" + imageUploadInfo
                .getFilename()))) {
            bo.write(Base64Utils.decodeFromString(imageUploadInfo.getImage()));
            bo.flush();
        } catch (IOException e) {
            // todo log
            throw new RuntimeException(e);
        }

    }
}
```

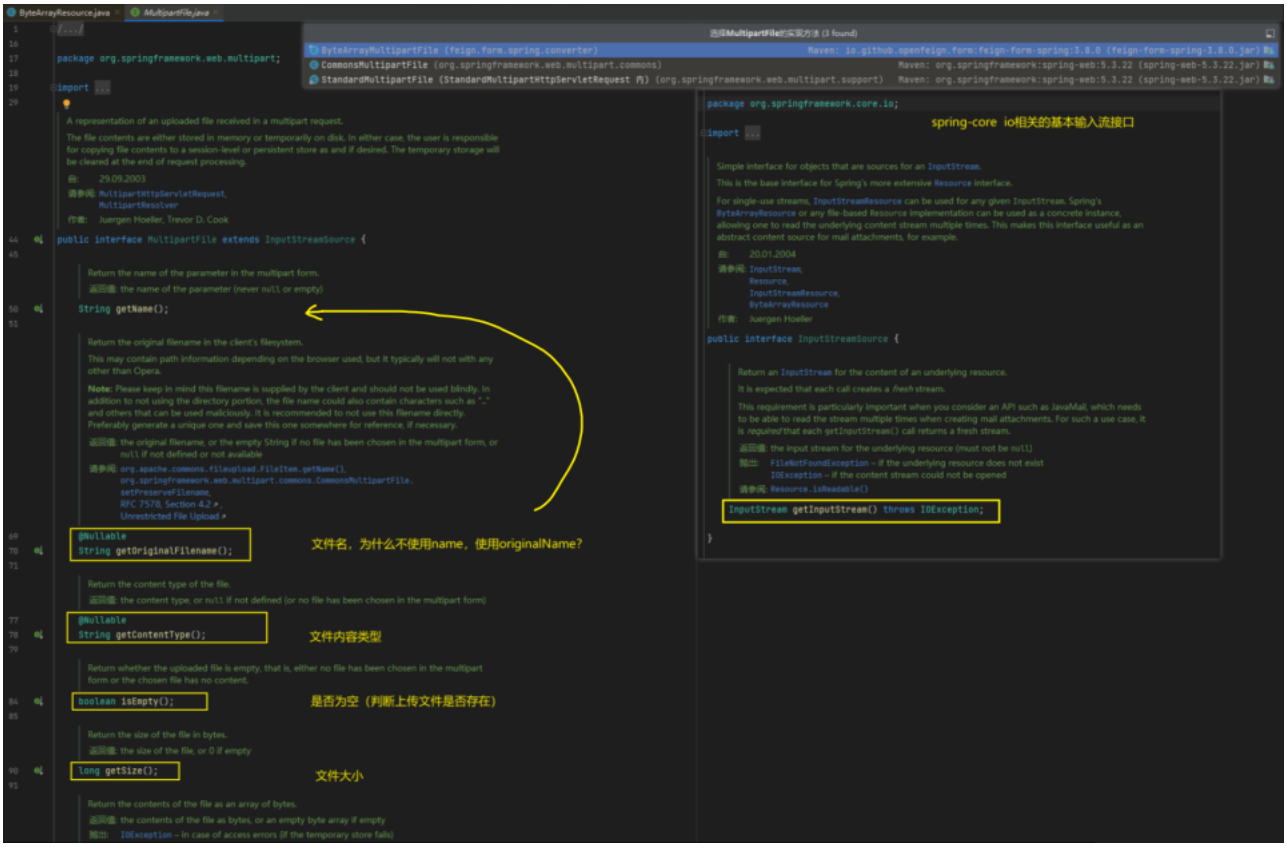
```java
@Getter
@Setter
@Accessors(chain = true)
@NoArgsConstructor
@AllArgsConstructor
public class ImageUploadDTO {

    0 个用法
    private String filename;

    // base64编码的单份图片
    0 个用法
    private String image;

}
```

方案二：使用MultipartFile

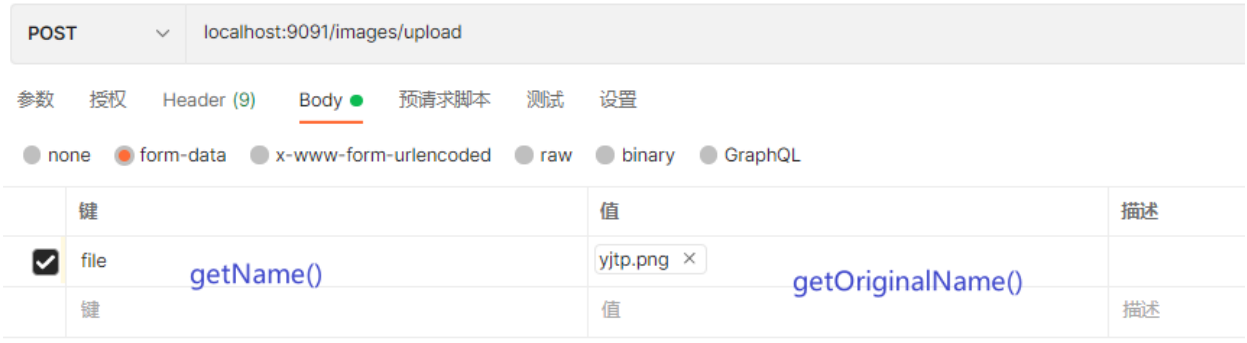
> HTML中有没有一种统一的文件上传方式，例如通过表单的encrypt类型指定。
>
> - enctype 属性规定在将表单数据发送到服务器之前如何对其进行编码。
>
> | 值 | 描述 |
> | --- | --- |
> | application/x-www-form-urlencoded | 默认。在发送前对所有字符进行编码（将空格转换为 "+" 符号，特殊字符转换为 ASCII HEX 值）。 |
> | multipart/form-data | 不对字符编码。当使用有文件上传控件的表单时，该值是必需的。 |
> | text/plain | 将空格转换为 "+" 符号，但不编码特殊字符。 |

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

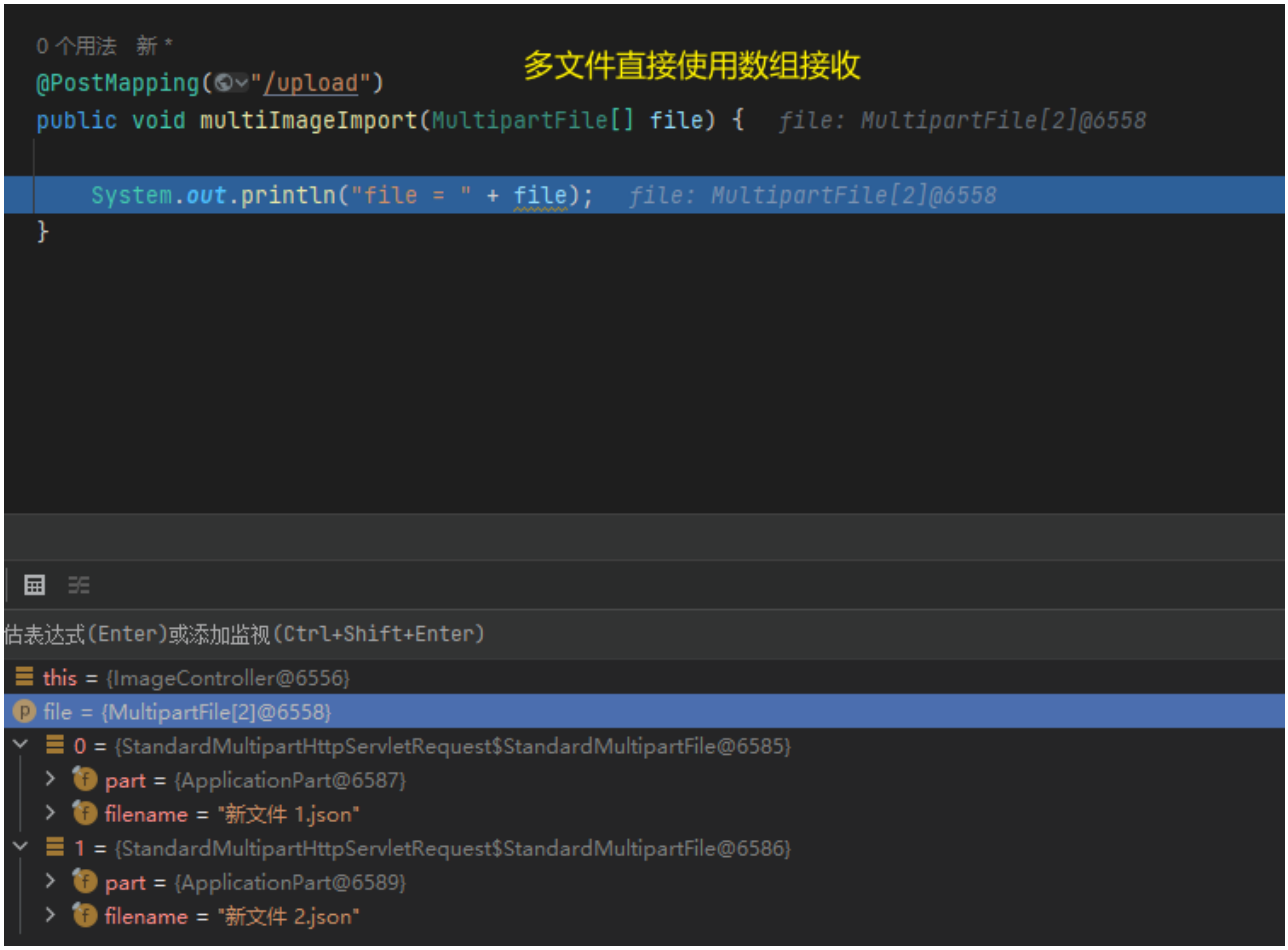


  -
- *getOriginalName*
  - 获取文件名
- *getContentType*
  - image/png
  - ...(MIME类型)
- *getSize*
  - 获取文件大小（字节）
- *isEmpty*
  - 上传文件是否为空 -> getSize==0（*例如一个空的.txt文件*）

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。

```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) {

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

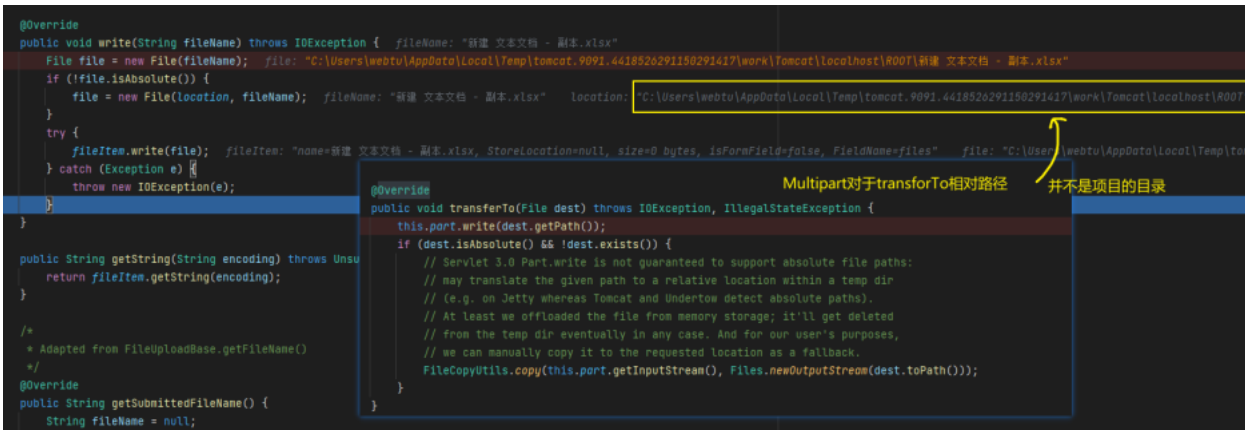


3. Multipart数据写入文件

- *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！*

-

```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();
        ) {
            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);
        }
    }
}
}
```

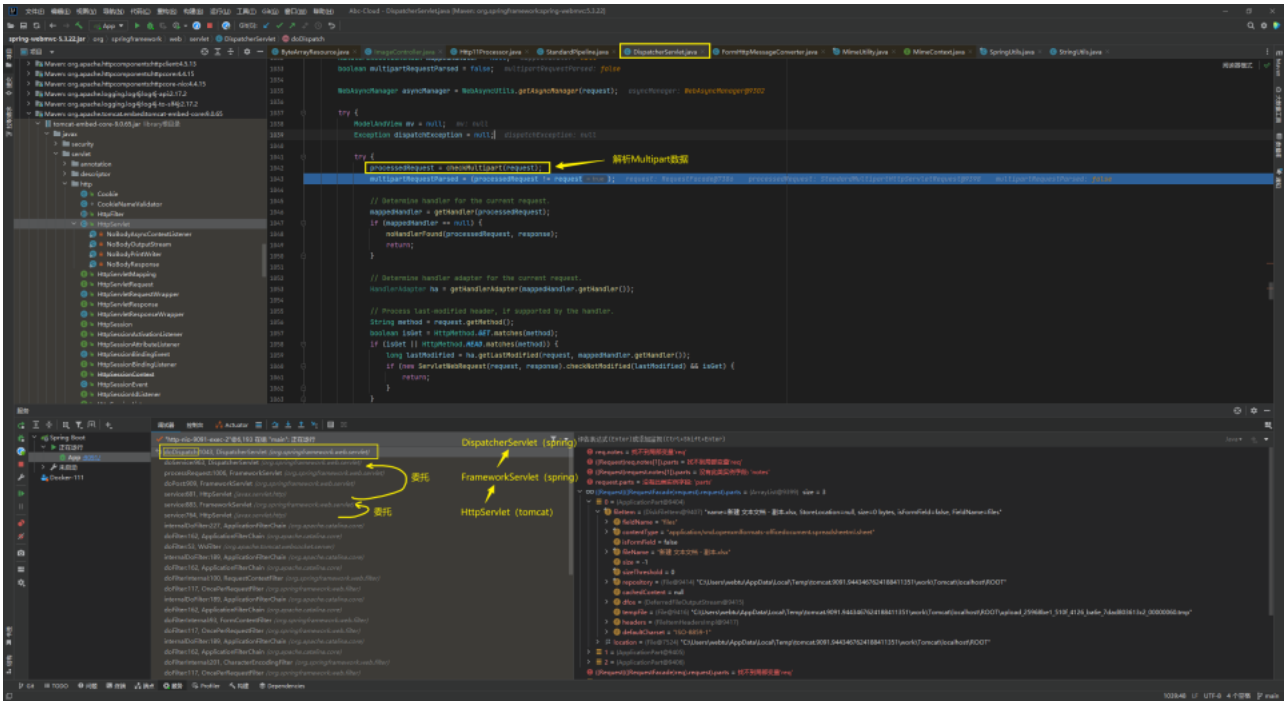
附录1：Multipart文件在哪一个Servlet中进行加载?

附录2：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

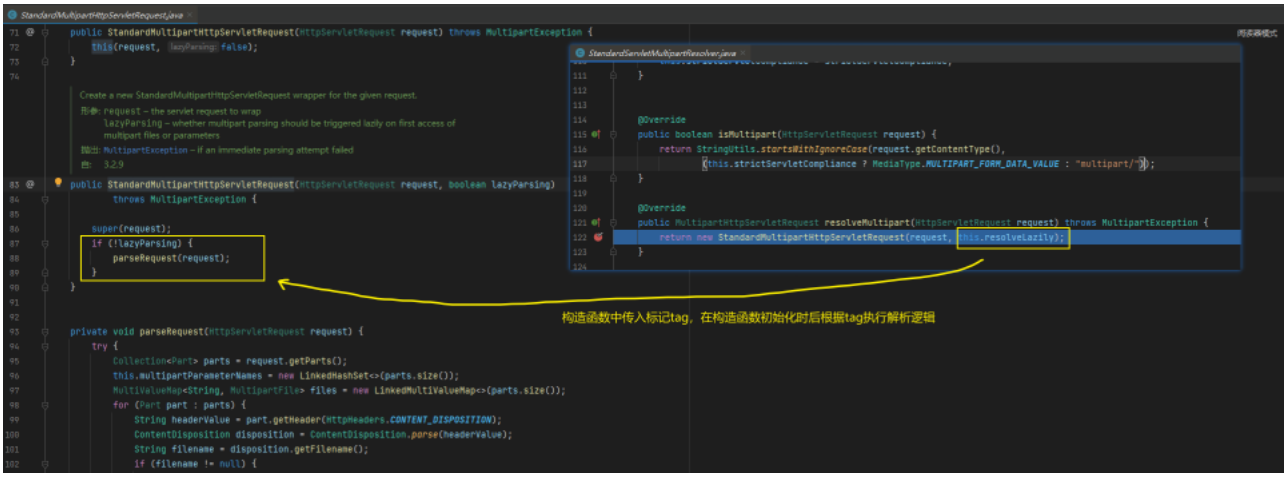
```java
@SpringBootApplication
public class App {

    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
```

```java
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



附录3：CORS后端跨域配置

```
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
     */
    @Bean
    public WebMvcConfigurer corsConfigurer() {
        return new WebMvcConfigurer() {
            @Override
            public void addCorsMappings(@NonNull CorsRegistry registry) {
                registry.addMapping("/test/**")
                        .allowedOrigins("http://localhost:8880", "http://localhost:8881")
                        .allowedMethods("GET", "POST")
                        .allowCredentials(false).maxAge(3600);
            }
        };
    }

}
```