

vue+Minio uploads multi-document progress

Breeze midnight 2022-03-16 09:48 ⚡ 7091

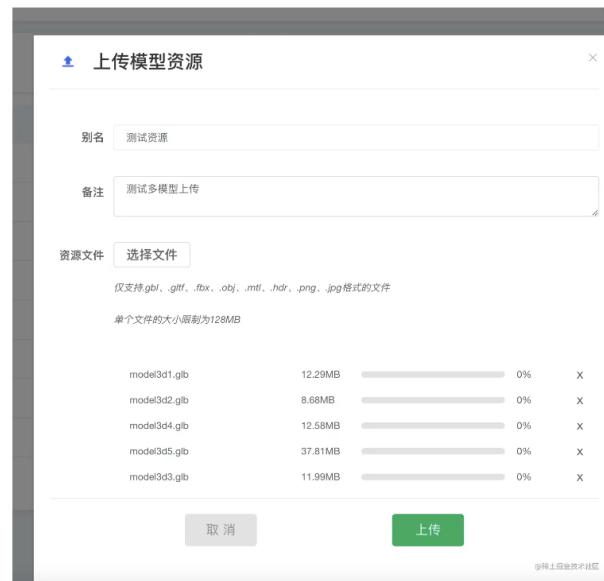


background

I suddenly received a product demand recently, which is a bit special. I will share it with you here. The demand is as follows

1. Submit the form and upload the model resources
2. The model file is a big file, to show the upload progress, and it can be deleted at the same time
3. The model file should be uploaded to the server, and the data of the table should be synchronized to the database
4. At the same time, the model address after simultaneous upload must be uploaded to the database
5. Use Minio for document management at the back end

The design drawings are as follows



At first, I thought it was a simple form upload. I found that it was not a big file upload, but soon I found that not only upload large files, but also link the file information to the form.

Based on this wonderful situation, I discussed with the back brother and decided to use the following plan

Implementation plan

Walk in 2 steps

1. When clicking on the upload, submit the form information to the database first, and then return to a form id at the back end
2. When all files are uploaded, call another service and send the uploaded address and form id to the back end

In this way, the above requirements are completed

Learn about Minio

Here everyone first understand Minio's js [SDK document](#)

There are 2 very important interfaces to use today

One is to generate an address for file upload

```
presignedPutObject(bucketName, objectName, expiry[, callback])
```

生成一个给HTTP PUT请求用的presigned URL。浏览器/移动端的客户端可以用这个URL进行上传，即使其所在的存储桶是私有的。这个presigned URL可以设置一个失效时间，默认值是7天。

参数	类型	描述
bucketName	string	存储桶名称。
objectName	string	对象名称。
expiry	number	失效时间（以秒为单位），默认是7天，不得大于七天。
callback(err, presignedUrl)	function	如果err不是null则代表有错误。presignedUrl用于使用PUT请求进行上传。如果没有传callback的话，则返回一个Promise对象。

示例

```
// expires in a day.
minioClient.presignedPutObject("mybucket", "hello.txt", 24*60*60, function(err, presignedUrl) {
  if (err) return console.log(err);
  console.log(presignedUrl);
})
```



Breeze midnight

Front end R&D engineer

attention

Private letter

Get some praise 1,228

Article read 109,594

Nuggets have supported dark models

It can be configured in "My settings-General Setup"

Click to go

Related articles

[Vue+Echarts enterprise-level large-screen project adaptation plan](#)
693 praise · 246 comments

[The headache of the element-ui component in vue defaults to the modular modification of css](#)
148 points · 30 comments

[Vuex detailed one: thoroughly understand the state, mapState, mapGetters, mapMutations, mapActions](#)
69 points · 10 comments

[Finally told Vuex's dispatch and commit thoroughly](#)
16 points · 1 comment

[The mobile end rem and vw must be known at the front end](#)
17 points · 4 comments



限时领掘金会员

立即前往 >

table of Contents

background

Implementation plan

- Learn about Minio
- Implementation steps
 - 1.Create storage barrels
 - 2.Select file
 - 3.Create upload queues
 - 4.Start uploading
 - 5.After uploading, the synchronize...
 - 6.Delete file

Complete code

Source code sharing

Next

[Vuex detailed one: thoroughly...](#)

One is to get the download address of the file after uploading

```
presignedGetObject(bucketName, objectName, expiry, cb)
生成一个给HTTP GET请求用的presigned URL。浏览器/移动端的客户端可以用这个URL进行下载，即使其所在的存储桶是私有的。这个presigned URL可以设置一个失效时间，默认值是7天。
```

参数

参数	类型	描述
bucketName	string	存储桶名称。
objectName	string	对象名称。
expiry	number	失效时间（以秒为单位），默认是7天，不得大于七天。
callback(err, presignedUrl)	function	如果 err 不是null则代表有错误，presignedUrl 就是指用于临时下载的URL。如果没有设置callback的话，则返回一个 promise 对象。

示例

```
// expires in a day.
minioClient.presignedGetObject("myBucket", "hello.txt", 24 * 60 * 60, function(err, presignedUrl) {
  if (err) return console.log(err);
  console.log(presignedUrl);
})
```

Implementation steps

Here is a request for upload using the original ajax request. As for why, there will be a later

1.Create storage barrels

Create an example of Minio upload

```
var Minio = require("minio")
this.minioClient = new Minio.Client({
  endPoint: '192.168.172.162', //后端提供
  port: 9000, //端口号默认9000
  useSSL: true,
  accessKey: 'Q3AM4CUQ867SPQQA43P2F', //后端提供
  secretKey: 'zurftte8lswRu7BJ86wekitnifIlbZam1KYy3TG'
})
this.userBucket = 'yourBucketName' //这里后端需要提供给你，一个存储桶名字
```

2.Select file

Use the input label to select files here. When clicking on the selection files, call the click method of input to open the local folder

```
<el-form-item label="资源文件">
  <el-button
    style="marginRight:10px;"
    @click="selectFile()"
    size="mini"
    >选择文件</el-button>
  <input
    :accept="acceptFileType"
    multiple="multiple"
    type="file"
    id="uploadInput"
    ref="uploadInput"
    v-show="false"
    @change="getAndFormatFile()"
  >
<i class="tip">仅支持.gbl .gltf .fbx .obj .mtl .hdr .png .jpg格式的文件</i>
<i class="tip">单个文件的大小限制为128MB</i>
</el-form-item>
```

```
selectFile() {
  let inputDOM = this.$refs.uploadInput
  inputDOM.click();
},
```

Then it is to format the file

```
//格式化文件并创建上传队列
getAndFormatFile(){
  let files = this.$refs.uploadInput.files
  const userBucket = this.userBucket
  if(files.length > 6){
    this.$message({
      message: '最大只能上传6个文件',
      type: 'warning'
    })
    return
  }
  files.forEach((file, index) => {
    if ((file.size / 1024 / 1024).toFixed(2) > 128) { //单个文件限制大小为128MB
      this.$message({
        message: '文件大小不能超过128MB',
        type: 'warning'
      })
      return
    }
    //创建文件的put方法的url
    this.minioClient.presignedPutObject(userBucket, file.name, 24 * 60 * 60, (err, presignedUrl) =>
      if (err) {
        this.$message({
```

```

24     message: '服务器连接超时',
25     type: 'error'
26   })
27   return err
28 }
29 let fileIcon = this.getFileIcon(file)
30 let fileUploadProgress = '0%' //文件上传进度
31 this.userInfoList.push({
32   file, //文件
33   fileIcon, //文件对应的图标 className
34   fileUploadProgress, //文件上传进度
35   filePutUrl: presignedUrl, //文件上传put方法的url
36   fileGetUrl: '', //文件下载的url
37 })
38 })
39 })
40 this fileList = [...this.userInfoList]
41 },

```

3.Create upload queues

Here is defined as a method for creating file upload requests, using native XMLHttpRequest , It accepts the following parameters

- **file** :Documents to be uploaded
- **filePutUrl** :Put method address for file upload
- **customHeader** :Self-defined head messages
- **onUploadProgress** :Progress monitoring function uploaded by file
- **onUploaded** :Monitoring function completed by file upload
- **onError** :Wrong monitoring function uploaded from the file

```

//创建上传文件的http
createUploadHttp(config){
  const {file, filePutUrl, customHeader, onUploadProgress, onUploaded, onError} = config
  let fileName = file.name
  let http = new XMLHttpRequest();
  http.upload.addEventListener("progress", (e) => { //监听http的进度，并执行进度监听函数
    onUploadProgress({
      progressEvent: e,
      uploadingFile: file
    })
  }, false)
  http.onload = () => {
    if (http.status === 200 && http.status < 300 || http.status === 304) {
      try {
        //监听http的完成事件，并执行上传完成的监听函数
        const result = http.responseURL
        onUploaded({result, uploadedFile: file})
      } catch(error) {
        //监听错误
        onError({error, errorFile: file})
      }
    }
  }
  http.open("PUT", filePutUrl, true);
  //加入头信息
  Object.keys(customHeader).forEach((key, index) =>{
    http.setRequestHeader(key, customHeader[key])
  })
  http.send(file);
  return http //返回该http实例
}

```

4.Start uploading

```

//上传文件到存储桶
async handleUpload(){
  let _this = this
  if(this.userInfoList.length < 1) {
    this.$message({
      message: '请选择文件',
      type: 'warning'
    })
    return
  }
  //先上传文件的基本表单信息，获取表单信息的id
  try{
    const {remark, alias} = _this.uploadFormdata
    let res = await uploadModelSourceInfo({remark, serviceName: alias})
    _this.modelSourceInfoId = res.message
  }catch(error){
    if(error) {
      _this.$message({
        message: '上传失败，请检查服务',
        type: 'error'
      })
      return
    }
  }
  //开始将模型资源上传到远程的存储桶
  this.userInfoList.forEach((item, index) => {
    const {file, filePutUrl} = item
    let config = {
      file,
      filePutUrl,
      customHeader: {
        "X-FILENAME": encodeURIComponent(file.name),
        "X-Access-Token": getToken()
      },
      onUploadProgress: ((progressEvent, uploadingFile) => {
        let progress = (progressEvent.loaded / progressEvent.total).toFixed(2)
        this.updateFileUploadProgress(uploadingFile, progress)
      }),
      onUploaded: ({result, uploadedFile}) => {
        this.updateFileDownloadUrl(uploadedFile)
      }
    }
  })
}

```

```

42     },
43     onError: ({error, errorFile}) => {
44
45   }
46 }
47
48 let httpInstance = this.createUploadHttp(config) //创建http请求实例
49 this.httpQueue.push(httpInstance) //将http请求保存到队列中
50 })
51 },
52
53
54 //更新对应文件的上传进度
55 updateFileInfoProgress(uploadingFile, progress) {
56 this.userInfoList.forEach((item, index) => {
57 if(item.fileName === uploadingFile.name){
58 item.fileUploadProgress = (Number(progress)*100).toFixed(2) + '%'
59 }
60 })
61 },
62
63 //更新上传完成文件的下载地址
64 updateFileDownloadUrl(uploadedFile){
65 const userBucket = this.userBucket
66 this.userInfoList.forEach((item, index) => {
67 if(item.fileName === uploadedFile.name){
68 this.minioClient.presignedGetObject(userBucket, uploadedFile.name, 24*60*60, (err, presignedUrl)
69 if (err) return console.log(err)
70 item.fileGetUrl = presignedUrl
71 })
72 }
73 })
74 },
75
76

```

5 After uploading, the synchronized file address is given to the back end

Monitor the list of documents in the watch. When all the document progress is 100%, it means that the upload is complete, and then the file information can be synchronized

```

1
2 watch:{
3   fileInfoList: {
4     handler(val){
5       //1.3所有文件都上传到存储桶后，将上传完成后的文件地址、文件名字同步后端
6       if(val.length < 1) return
7       let allFileHasUpload = val.every((item, index) => {
8         return item.fileGetUrl.length > 1
9       })
10      if(allFileHasUpload) {
11        this.allFileHasUpload = allFileHasUpload
12        const (modelSourceInfoId) = this
13        if(modelSourceInfoId.length < 1) {
14          return
15        }
16        const url = process.env.VUE_APP_BASE_API + "/vector-map/threeDimensionalModelService/invokeM
17        const files = val.map((ite, idx) => {
18          return {
19            fileName: ite.fileName,
20            fileUrl: ite.fileGetUrl
21          }
22        })
23        this.syncAllUploadedFile(url, files, modelSourceInfoId)
24      }
25
26    },
27    deep: true
28  },
29 },
30
31
32
33 //同步已上传的文件到后端
34 syncAllUploadedFile(url, files, modelSourceInfoId){
35   let xhr = new XMLHttpRequest()
36   xhr.onload = () => {
37     if (xhr.status === 200 && xhr.status < 300 || xhr.status === 304) {
38       try {
39         const res = JSON.parse(xhr.responseText)
40         if(res && res.code === 200){
41           this.$message({
42             message: '上传完成',
43             type: 'success'
44           })
45           this.$emit('close')
46           this.userInfoList = []
47           this.listFiles = []
48           this.httpQueue = []
49         }
50       } catch(error) {
51         this.$message({
52           message: '上传失败, 请检查服务',
53           type: 'error'
54         })
55       }
56     }
57   }
58   xhr.open("post", url, true)
59   xhr.setRequestHeader('Content-Type', 'application/json')
60   xhr.setRequestHeader('X-Access-Token', getToken())
61   //将前段1.1获取文件信息的id作为头信息传递到后端
62   xhr.setRequestHeader('ThreeDimensionalModel-ServiceID', modelSourceInfoId)
63   xhr.send(JSON.stringify(files))
64 },
65

```

6.Delete file

Pay attention to when deleting files

- Delete local file cache

- Delete the files in the storage barrel
- Stop http requests corresponding to current documents

```

1 //删除文件，并取消正在文件的上传
2   _deleteFile(fileInfo, index){
3     this.httpQueue[index] && this.httpQueue[index].abort()
4     this.httpQueue[index] && this.httpQueue.splice(index, 1)
5     this.userInfoList.splice(index, 1)
6     this fileList.splice(index, 1)
7     this.removeRemoteFile(fileInfo)
8   },
9
10 //清空文件并取消上传队列
11 clearFile() {
12   this.userInfoList.forEach((item, index) => {
13     this.httpQueue[index] && this.httpQueue[index].abort()
14     this.httpQueue[index] && this.httpQueue.splice(index, 1)
15     this.removeRemoteFile(item)
16   })
17   this.userInfoList = []
18   this.httpQueue = []
19   this fileList = []
20 },
21 //删除远程文件
22 removeRemoteFile(fileInfo){
23   const userBucket = this.userBucket
24   const { fileUploadProgress, file } = fileInfo
25   const fileName = file.name
26   const complete = fileUploadProgress === '100.00%' ? true : false
27   if(complete){
28     this.minioClient.removeObject(userBucket, fileName, function(err) {
29       if (err) {
30         return console.log('Unable to remove object', err)
31       }
32       console.log('Removed the object')
33     })
34   }else{
35     this.minioClient.removeIncompleteUpload(userBucket, fileName, function(err) {
36       if (err) {
37         return console.log('Unable to remove incomplete object', err)
38       }
39       console.log('Incomplete object removed successfully.')
40     })
41   }
42 },
43
44

```

Complete code

The complete code here is copied directly from the project, which uses some self-packaged services and methods such as Backend interface, AES decryption, access to Token, form verification, etc.

```

1 import{uploadModelSourceInfo, uploadModelSource, getToken} from '@api/map'
2 import AES from '@utils/AES.js'
3 import { getToken } from '@utils/auth'
4 import * as myValidDate from "@utils/formValidate";

```

```

1 /**
2  * 文件说明
3  * @Author: zhuds
4  * @Description: 模型资源上传弹窗
5  * 分为3个步骤
6  * 1.先将文件的基本表单信息上传给后端，获取文件信息的ID
7  * 2.然后把文件上传存储
8  * 3.等所有文件都上传完成后，再将上传完成后的文件信息传送给后端，注意，此时的请求头要带上第1步获取的文件信息id
9  * @Date: 2/28/2022, 1:13:29 PM
10 * @LastEditDate: 2/28/2022, 1:13:29 PM
11 * @LastEditor:
12 */
13 <template>
14   <div class="upload-model">
15     <el-dialog
16       :visible.sync="isVisible"
17       @close="close()"
18       :show-close="false"
19       :close-on-click-modal="false"
20       top="10vh"
21       v-if="isVisibile"
22       :destroy-on-close="true"
23     >
24       <div slot="title" class="header-title">
25         <div class="icon"></div>
26         <span>上传模型资源</span>
27         <i class="el-icon-close" @click="close()"></i>
28       </div>
29
30       <el-form
31         :label-position="labelPosition"
32         label-width="80px"
33         :model="uploadFormData"
34         ref="form"
35         :rules="rules"
36       >
37         <el-form-item label="别名">
38           <el-input size="small" v-model="uploadFormData.alias"></el-input>
39         </el-form-item>
40
41         <el-form-item label="备注">
42           <el-input type="textarea" v-model="uploadFormData.remark" size="small"></el-input>
43         </el-form-item>
44
45         <el-form-item label="资源文件">
46           <el-button
47             style="marginRight:10px;"
48             @click="selectFileDialog"
49             size="mini">
50             >选择文件</el-button>

```

```

51      <input
52        :accept="acceptFileType"
53        multiple="multiple"
54        type="file"
55        id="uploadInput"
56        ref="uploadInput"
57        v-show="false"
58        @change="getAndFormatFile()"
59      >
60      <i class="tip">仅支持.gbl、.gltf、.fbx、.obj、.mtl、.hdr、.png、.jpg格式的文件</i>
61      <i class="tip">单个文件的大小限制为128MB</i>
62    </el-form-item>
63
64  </el-form>
65
66  <div class="file-list" v-show="fileInfoList.length > 0">
67    <div class="file-item" v-for="(item, index) in fileInfoList" :key="index">
68      <div class="icon"></div>
69      <div class="name">{{item.file.name}}</div>
70      <div class="size">{{(item.file.size/1024).toFixed(2)}}MB </div>
71      <div class="progress">
72        <div class="bar" :style="{width: item.fileUploadProgress}"></div>
73      </div>
74      <div class="rate">{{item.fileUploadProgress}}</div>
75
76      <div class="delete-btn" @click="deleteFile(item, index)">x</div>
77    </div>
78  </div>
79  <div class="custom-footer">
80    <button class="info" @click="close()">取消</button>
81    <button class="success" @click="handleUpload()>上传</button>
82  </div>
83
84
85  </el-dialog>
86 </div>
87 </template>
88
89 <script>
90
91 import {uploadModelSourceInfo, uploadModelSource, getMinioConfig} from '@/api/map'
92 import AES from '@/utils/AES.js'
93 import { getToken } from '@/utils/auth'
94 import * as myValidDate from '@/utils/formValidate';
95 let Minio = require('minio')
96
97 export default {
98   name: 'UploadModelDialog',
99   props: {
100     isVisible: {
101       type: Boolean,
102       default: false
103     },
104   },
105   data(){
106     return {
107       labelPosition: 'right',
108       uploadFormData: [
109         alias: '', //服务名称
110         remark: '', //备注
111       ],
112       rules: [
113         serviceName: [
114           validator: myValidDate.validateServiceName,
115           trigger: 'blur',
116           required: true,
117         ],
118       ],
119       acceptFileType:".glb,.gltf,.fbx,.obj,.mtl,.hdr,.png,.jpg,.mp4",
120       fileList: [], //待上传的文件列表
121       fileInfoList: [], //格式化后的文件信息列表
122       userBucket: null,
123       httpQueue: [], //上传文件的http队列
124       allFileHasUpload: false, //是否完成上传
125       modelSourceInfoId: '', //模型资源基本信息的id
126     }
127   },
128   watch:{
129     fileInfoList: {
130       handler(val){
131         //1.3所有文件都上传到存储桶后，将上传完成后的文件地址、文件名字同步后端
132         if(val.length < 1) return
133         let allFileHasUpload = val.every((item, index) => {
134           return item.fileGetUrl.length > 1
135         })
136         if(allFileHasUpload) {
137           this.allFileHasUpload = allFileHasUpload
138           const {modelSourceInfoId} = this
139           if(modelSourceInfoId.length < 1) {
140             return
141           }
142           const url = process.env.VUE_APP_BASE_API + "/vector-map/threeDimensionalModelService/inv
143           const files = val.map((ite, idx) => {
144             return {
145               fileName: ite.file.name,
146               fileUrl: ite.fileGetUrl
147             }
148           })
149           this.syncAllUploadedFile(url, files, modelSourceInfoId)
150         }
151       }
152     },
153     deep: true
154   }
155 },
156
157 created() {
158   this.initMinioClient()
159 },
160
161 beforeDestroy() {
162   if(!this.allFileHasUpload) {
163     this.clearFile()
164   }
165 }
166
167 methods:{
168   //创建存储桶
169   async initMinioClient(){
170

```

```
171     const { code, result, message } = AES.decryptToJson(await getMinioConfig())
172     if(!result || code !== 200) {
173       this.$customMessage.error({message: '获取存储桶配置信息出错'})
174       return false
175     }
176     let {accessKey, bucketName, endPoint, secretKey} = result
177     //console.log({accessKey, bucketName, endPoint, secretKey})
178
179     let endPointStr = endPoint.split(":")[1]
180     let formatPort = Number(endPoint.split(":")?[2])
181     let formatEndPoint = endPointStr.split("//")?[1]
182     this.userBucket = bucketName
183     this.minioClient = new Minio.Client({
184       useSSL: false,
185       partSize: '20M',
186       port: formatPort,
187       endPoint: FormatEndPoint,
188       accessKey,
189       secretKey
190     });
191     let userBucket = this.userBucket
192     //userBucket只能作为字符串变量传入，不能作为其他变量的属性或者函数返回值，属于Minio的一个规定
193     this.minioClient.bucketExists(userBucket, (err)=> {
194       if (err && err.code == 'NoSuchBucket') {
195         this.minioClient.makeBucket(userBucket, 'us-east-1', (err)=> {
196           if (err) {
197             return console.log('创建存储桶失败', err)
198           }
199           // console.log('Bucket created successfully in "us-east-1".')
200         })
201       }else{
202         //console.log('存储桶存在')
203       }
204     })
205   },
206
207   close(flag = false) {
208     this.$emit('close', flag)
209     //关闭弹窗时，如果文件没有上传完成，则清空文件
210     if(!this.allFileHasUpload) {
211       this.clearFile()
212     }
213   },
214   selectFile() {
215     let inputDOM = this.$refs.uploadInput
216     inputDOM.click();
217   },
218
219   getFileSize(file){
220     let fileSize = ''
221     if(file.size / 1024 < 1){
222       fileSize = file.size + 'B'
223     }else if(file.size / 1024 /1024 < 1){
224       fileSize = file.size + 'KB'
225     }else if(file.size / 1024 /1024 >=1){
226       fileSize = file.size + 'MB'
227     }else{
228     }
229
230     return fileSize
231   },
232
233   //删除文件，并取消正在文件的上传
234   deleteFile(fileInfo, index){
235     this.httpQueue[index] && this.httpQueue[index].abort()
236     this.httpQueue[index] && this.httpQueue.splice(index, 1)
237     this.fileInfoList.splice(index, 1)
238     this fileList.splice(index, 1)
239     this.removeRemoteFile(fileInfo)
240   },
241
242   //清空文件并取消上传队列
243   clearFile() {
244     this.fileInfoList.forEach((item, index) => {
245       this.httpQueue[index] && this.httpQueue[index].abort()
246       this.httpQueue[index] && this.httpQueue.splice(index, 1)
247       this.removeRemoteFile(item)
248     })
249   }
250   this.fileInfoList = []
251   this.httpQueue = []
252   this fileList = []
253 },
254 //删除远程文件
255 removeRemoteFile(fileInfo){
256   const userBucket = this.userBucket
257   const { fileUploadProgress, file } = fileInfo
258   const fileName = file.name
259   const complete = fileUploadProgress === '100.00%' ? true : false
260   if(complete){
261     this.minioClient removeObject(userBucket, fileName, function(err) {
262       if (err) {
263         return console.log('Unable to remove object', err)
264       }
265       console.log('Removed the object')
266     })
267   }else{
268     this.minioClient.removeIncompleteUpload(userBucket, fileName, function(err) {
269       if (err) {
270         return console.log('Unable to remove incomplete object', err)
271       }
272       console.log('Incomplete object removed successfully.')
273     })
274   }
275 }
276 //格式化文件并创建上传队列
277 getAndFormatFile(){
278   let files = this.$refs.uploadInput.files
279   const userBucket = this.userBucket
280   if(files.length > 6) {
281     this.$message({
282       message: '最大只能上传6个文件',
283       type: 'warning'
284     })
285     return
286   }
287   files.forEach((file, index) => {
288     if ((file.size / 1024 / 1024).toFixed(2) > 128) { //单个文件限制大小为128MB
289       this.$message({
290         message: '文件大小不能超过128MB',
291       })
292     }
293   })
294 }
```

```
291         type: 'warning'
292     })
293     return
294 }
//创建文件上传的url并格式化文件信息
295 this.minioClient.presignedPutObject(userBucket, file.name, 24 * 60 * 60, (err, presignedUr
296 if (err) {
297     this.$message({
298         message: '服务器连接超时',
299         type: 'error'
300     })
301     return err
302 }
303 let fileIcon = this.getFileIcon(file)
304 let fileUploadProgress = '0%'
305 this.userInfoList.push({
306     file, //文件
307     fileIcon, //文件对应的图标 className
308     fileUploadProgress, //文件上传进度
309     filePutUrl: presignedUrl, //文件上传put方法的url
310     fileGetUrl: '', //文件下载的url
311     fileGetUrl: ''
312 })
313 })
314 })
315 this.userInfoList = [...this.userInfoList]
316 },
317
318 //1.上传文件到存储桶
319 async handleUpload(){
320     let _this = this
321     if(this.userInfoList.length < 1) {
322         this.$message({
323             message: '请选择文件',
324             type: 'warning'
325         })
326         return
327     }
328     //1.1先上传文件的基本表单信息，获取文件信息的id
329     try{
330         const {remark, alias} = _this.uploadFormData
331         let res = await uploadModelSourceInfo({remark, serviceName: alias})
332         _this.modelSourceInfoId = res.message
333     }catch(error){
334         if(error) {
335             _this.$message({
336                 message: '上传失败, 请检查服务',
337                 type: 'error'
338             })
339             return
340         }
341     }
342
343     //1.2开始将模型资源上传到远程的存储桶
344     this.userInfoList.forEach((item, index) => {
345         const {file, filePutUrl} = item
346         let config = {
347             file,
348             filePutUrl,
349             customHeader: {
350                 "X-Filename": encodeURIComponent(file.name),
351                 "X-Access-Token": getToken()
352             },
353             onUploadProgress: ({progressEvent, uploadingFile}) => {
354                 let progress = (progressEvent.loaded / progressEvent.total).toFixed(2)
355                 this.updateFileUploadProgress(uploadingFile, progress)
356             },
357             onUploaded: ({result, uploadedFile}) => {
358                 this.updateFileDownloadUrl(uploadedFile)
359             },
360             onError: ({error, errorFile}) => {
361                 }
362             }
363         }
364
365         let httpInstance = this.createUploadHttp(config)
366         this.httpQueue.push(httpInstance)
367     }),
368 },
369
370     //1更新对应文件的上传进度
371     updateFileUploadProgress(uploadingFile, progress) {
372         //console.log(uploadingFile, progress)
373         this.userInfoList.forEach((item, index) => {
374             if(item.fileName == uploadingFile.name){
375                 item.fileUploadProgress = (Number(progress)*100).toFixed(2) + '%'
376             }
377         })
378     },
379
380     //更新上传完成文件的下载地址
381     updateFileDownloadUrl(uploadedFile){
382         const userBucket = this.userBucket
383         this.userInfoList.forEach((item, index) => {
384             if(item.fileName == uploadedFile.name){
385                 this.minioClient.presignedGetObject(userBucket, uploadedFile.name, 24*60*60, (err, presi
386                 if (err) return console.log(err)
387                 item.fileGetUrl = presignedUrl
388                 // console.log(presignedUrl)
389             })
390         })
391     },
392 },
393
394     //同步已上传的文件到后端
395     syncAllUploadedFile(url, files, modelSourceInfoId){
396         let xhr = new XMLHttpRequest()
397         xhr.onload = () => {
398             if (xhr.status === 200 && xhr.status < 300 || xhr.status === 304) {
399                 try {
400                     const res = JSON.parse(xhr.responseText)
401                     if(res && res.code === 200){
402                         this.$message({
403                             message: '上传完成',
404                             type: 'success'
405                         })
406                         // setTimeout(() => {
407                         //     this.$emit('close')
408                         //     this.userInfoList = []
409                         //     this.userInfoList = []
410                         //     this.httpQueue = []
411                         //     })
412                     }
413                 }
414             }
415         }
416     }
417 }
```

```
411         // j, 上传)
412     }
413   } catch(error) {
414     this.$message({
415       message: '上传失败, 请检查服务',
416       type: 'error'
417     })
418   }
419 }
420 }
421 xhr.open("post", url, true)
422 xhr.setRequestHeader('Content-Type', 'application/json')
423 xhr.setRequestHeader('X-Access-Token', getToken())
424 //将前面1.1获取文件信息的id作为头信息传递到后端
425 xhr.setRequestHeader('ThreeDimensionalModel-ServiceID', modelSourceInfoId)
426 xhr.send(JSON.stringify(files))
427 },
428
429 //根据文件类型图标class
430 getFileIcon(file) {
431   const { type } = file
432   let icon = ''
433   return icon
434 },
435
436 //创建上传文件的http
437 createUploadHttp(config){
438   const {file, filePutUrl, customHeader, onUploadProgress, onUploaded, onError} = config
439   let fileName = file.name
440   let http = new XMLHttpRequest();
441   http.upload.addEventListener("progress", (e) => {
442     onUploadProgress({
443       progressEvent: e,
444       uploadingFile: file
445     })
446   }, false)
447
448   http.onload = () => {
449     if (http.status === 200 && http.status < 300 || http.status === 304) {
450       try {
451         const result = http.responseURL
452         onUploaded({ result, uploadedFile: file})
453       } catch(error) {
454         onError({ error, errorFile: file})
455       }
456     }
457   }
458   http.open("PUT", filePutUrl, true);
459   Object.keys(customHeader).forEach((key, index) =>{
460     http.setRequestHeader(key, customHeader[key])
461   })
462   http.send(file);
463   return http
464 }
465
466 }
467 }
468 </script>
469
470 <style scoped lang="scss">
471
472 .header-title {
473   // height: 24px;
474   border-bottom: 1px solid #EFEFEF;
475   padding-bottom: 20px;
476   //outline: 1px solid red;
477   .icon {
478     width: 26px;
479     height: 26px;
480     background-image: url(../images/icon_upload.png);
481     float: left;
482     background-size: 100% 100%;
483     margin-right: 10px;
484     margin-left: 10px;
485   }
486   span {
487     font-size: 23px;
488     font-family: Source Han Sans CN;
489     font-weight: 500;
490     color: #333333;
491   }
492   i {
493     float: right;
494     font-size: 16px;
495     color: rgba(176, 176, 176, 1);
496     cursor: pointer;
497     &:hover {
498       color: #006600 ;
499     }
500   }
501 }
502 .tip {
503   font-size: 12px;
504   display: block;
505 }
506 .file-list {
507   box-sizing: border-box;
508   padding: 5px;
509   // border: 1px solid red;
510   // width: 840px;
511   margin: 5px auto;
512   margin-left: 80px;
513   .file-item {
514     min-height: 32px;
515     display: flex;
516     justify-content: flex-start;
517     align-items: center;
518     font-size: 12px;
519     div {
520       margin-right: 15px;
521       text-align: left;
522     }
523     .name {
524       width: 200px;
525     }
526     .size {
527       width: 60px;
528     }
529     .progress {
530       width: 180px;
531       height: 8px;
532     }
533   }
534 }
```

```
532     border-radius: 4px;
533     background-color: #E2E2E2;
534     .bar {
535       width: 50%;
536       height: 8px;
537       border-radius: 4px;
538       background-color: #13A763;
539     }
540   }
541   .rate {
542     width: 60px;
543     // border: 1px solid red;
544   }
545   .delete-btn {
546     cursor: pointer;
547     font-size: 16px;
548   }
549 }
550 }
551 }
552 }
553 .custom-footer {
554 // border: 1px solid red;
555 display: flex;
556 justify-content: space-evenly;
557 align-items: center;
558 width: 100%;
559 height: 80px;
560 background-color: #fff;
561 //box-shadow: 0px 1px 0px 0px red;
562 border-top: 1px solid #eefefef;
563 z-index: 10;
564 button {
565 width: 90px;
566 height: 40px;
567 border-radius: 4px;
568 border: 0;
569 font-size: 16px;
570 font-family: Source Han Sans CN;
571 &:focus {
572 border: 0;
573 outline: 0;
574 }
575
576 &.info {
577 color: #878f8f;
578 background: #e2e2e2;
579 &:hover{
580 background-color: #A6A9AD;
581 cursor: pointer;
582 color: #fff;
583 }
584 }
585 &.success {
586 background: #12a763;
587 color: #fff;
588 &:hover{
589 cursor: pointer;
590 background-color: #73C132;
591 }
592 }
593 }
594 }
595 </style>
596
597
```

Source code sharing

In fact, at the beginning, I wanted to provide a demo. I found that this thing is strongly tied to the product function. Without the tested service address and storage bucket, I cannot make an open case.

So the above code is just to provide you with a way and idea of realization. The specific details in it deal with what I do is more complicated

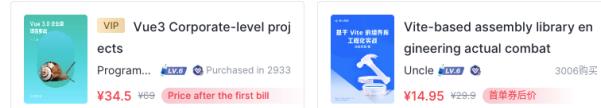
If there is something you don't understand, talk about the comment area

label : [Vue.js](#) [JavaScript](#) [front end](#)

The article is included in the column :



Related brochures



评论



看完啦, [登录](#) 分享一下感受吧~

全部评论 32

● 最新 ◆ 最热



掘金酱

首席客服君 @ 掘金

掘金技术社区签约计划正在招募优质作者，掘金酱邀你报名！即日起到7月31日，报名+投稿，不仅可以抽奖 AirPods, 分瓜2万元奖池，成为签约作者还能写作变现！现金收益、流量扶持、签约认证等你参与~ juejin.cn

悟 点赞 回复

11小时前



vite+vue3中怎么使用

悟 点赞 回复

1年前



清风夜半

(作者)

一样使用啊，基本全是js代码

悟 点赞 回复

12小时前



回复 清风夜半

vite中使用minio,里面的依赖会各种报错...

“一样使用啊，基本全是js代码”

悟 点赞 回复

查看更多回复 ▾



zyh619

开发人员 @ 武汉

老哥，有试过vite使用minio吗？minio里面的依赖各种报错😭

悟 点赞 回复

1年前



熙古今

兄弟，解决了吗？

悟 点赞 回复

9小时前



熙古今

你解决了吗，老哥，我也遇到了

悟 点赞 回复

9小时前

查看更多回复 ▾



LetMeTouchTou...

accessKey,
secretKey 直接暴露在客户端上，合适吗？。。。

悟 点赞 回复

1年前



清风夜半

(作者)

文章中为了讲解我明写了，项目中是通过加密服务获取的

悟 点赞 回复

1年前



知北

前端工程师

要是能控制上传并发数量，可以选择更多文件上传就更好了

悟 点赞 回复

1年前



耗子会飞

必须要控制的呀，不然大并发请求，直接崩了

悟 点赞 回复

1年前



無思童

看起来挺复杂的

悟 点赞 回复

1年前



清风夜半

(作者)

里面代码有需要封装的地方，还能进一步优化

悟 点赞 回复

1年前



耗子会飞

分片上传，前端工作量很大，后端没什么压力

悟 点赞 回复

1年前



用户646887783...

几个JS文件能提供一下参考学习，感谢。

悟 点赞 回复

1年前



最好沉默

frontend @ yellow lem...

存储桶相关的是不是应该后端写接口出来，前端只做上传资源和文件上传完整性验证

悟 点赞 回复

1年前



清风夜半

(作者)

是的，存储服务是后端搭建的，部署在云服务器

悟 点赞 回复

1年前



耗子会飞

前端不做完整性验证

悟 点赞 回复

1年前



迎风破浪

Java开发 @ 新时达

你这个有个很大的问题，不支持分片上传，大文件有问题的

1年前

1 2

清风夜半 (作者)

业务需求没要求大文件分片，后面规定了最大文件为64MB

2 回复

迎风破浪 回复 清风夜半

64m确实没必要

“业务需求没要求大文件分片，后面规定了最大文件为64MB”

点赞 回复



god830

Java Spring Boot

前炉石美服第一 星际2...

有兴趣聊聊吗 你可以看下我湊点

点赞 回复

相关推荐

徐小白 | 1年前 | Java Spring Boot

SpringBoot 整合 Minio 上传文件

6492 点赞 14 回复

Anxyh | 1年前 | Java

Minio 最佳分片上传PLUS

7753 点赞 40 回复

泉城老铁 | 2年前 | Spring Boot

springboot +vue 集成minio上传文件

1831 点赞 8 回复

高倍稳 | 12月前 | 前端

Vue文件下载进度条

6487 点赞 22 回复

随风而逝-风逝 | 3年前 | 面试 Vue.js

30 道 Vue 面试题，内含详细讲解（涵盖入门到精通，自测 Vue 掌握程度）

64.9w 点赞 8162 回复 289

躺平的sx | 3年前 | Java

Java构建minio文件系统实现分片上传

1496 点赞 2 回复

yeyan1996 | 3年前 | JavaScript

字节跳动面试官：请你实现一个大文件上传和断点续传

30.5w 点赞 6157 回复 610

程序猿小卡 | 5年前 | Node.js 前端 GitHub

Node.js: upload files, how service ends obtain file upload progress

1.0w 点赞 383 回复 5

tinyOrange | 2 years ago | Big data

Minio architecture analysis

6647 点赞 6 回复 comment

Xiling | Before November | front end JavaScript

Try S3+minio to upload large files

1857 点赞 9 回复 4

Meow | August | front end

TypeScript upload file to MinIO

869 点赞 5 回复 comment

Love sweet potato po... | 1 year ago | back end Yun Yuansheng

minio quick entry

5830 点赞 twenty four 回复 comment

alwaysVe | 3 years ago | Node.js

Nodejs file upload, monitor upload progress

3983 点赞 twenty one 回复 comment

OrzR3 | 1 year ago | front end Vuex

Vue project download progress plan

5897 点赞 42 回复 4

After logging in to the bug, you can get the following rights immediately :

Free trial courses

Collect useful articles

Check the footprint

Subscribe to high-quality columns

Experience check-in lottery

Raise growth level

Log in immediately

First use ? Click me to register