vue+Minio uploads multi-document progress

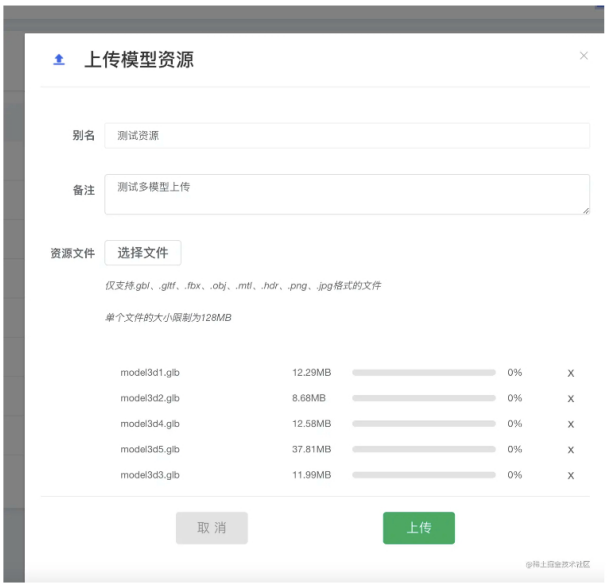
Breeze midnight 2022-03-16 09:48 7091

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 Mino

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

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

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 dispacth and commit thoroughly
16 points · 1 comment

The mobile end rem and vw must be known at the front end
17 points · 4 comments

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` 就是可用于临时下载的URL。 如果没有传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

```js
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: 'Q3AM3UQ867SPQQA43P2F', //后端提供
    secretKey: 'zuf+tfteSlswRu7BJ86wekitnifILbZam1KYY3TG'
});

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

```html
<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>
```

```js

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

```

Then it is to format the file

```js
//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({
```

```
            message: `服务器连接超时`,
            type: 'error'
          })
          return err
        }
        let fileIcon = this.getFileIcon(file)
        let fileUploadProgress = '0%' //文件上传进度
        this.fileInfoList.push({
          file, //文件
          fileIcon, //文件对应的图标 className
          fileUploadProgress, //文件上传进度
          filePutUrl: presignedUrl, //文件上传put方法的url
          fileGetUrl: '', //文件下载的url
        })
      })
    })
    this.fileList = [...this.fileInfoList]
  },
```

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

```js
//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

```js
//上传文件到存储桶
  async handleUplaod(){
    let _this = this
    if(this.fileInfoList.length < 1) {
      this.$message({
        message: `请先选择文件`,
        type: 'warning'
      })
      return
    }
    //先上传文件的基本表单信息，获取表单信息的id
    try{
      const {remark, alias} = _this.uploadFormData
      let res = await uploadModelSourceInfo({remark, serviceName: alias})
      _this.modelSourceInfoId = res.message
    }catch(error){
      if(error) {
        _this.$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.fileList.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)
```

```
        },
        onError: ({error, errorFile}) => {

        }
      }

      let httpInstance = this.createUploadHttp(config) //创建http请求实例
      this.httpQueue.push(httpInstance) //将http请求保存到队列中
    })
  },


//更新对应文件的上传进度
updateFileUploadProgress(uploadingFile, progress) {
this.fileInfoList.forEach((item, index) => {
  if(item.file.name === uploadingFile.name){
    item.fileUploadProgress = (Number(progress)*100).toFixed(2) + '%'
  }
})
},

//更新上传完成文件的下载地址
updateFileDownloadUrl(uploadedFile){
const userBucket = this.userBucket
this.fileInfoList.forEach((item, index) => {
  if(item.file.name === uploadedFile.name){
    this.minioClient.presignedGetObject(userBucket, uploadedFile.name, 24*60*60, (err, presignedUrl)
      if (err) return console.log(err)
      item.fileGetUrl = presignedUrl
    })
  }
})
},
```

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

```
                                                                                                    js Copy code

watch:{
  fileInfoList: {
    handler(val){
      //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
      if(val.length < 1) return
      let allFileHasUpload = val.every((item, index) => {
        return item.fileGetUrl.length > 1
      })
     if(allFileHasUpload) {
       this.allFileHasUpload = allFileHasUpload
       const {modelSourceInfoId} = this
       if(modelSourceInfoId.length < 1) {
         return
       }
       const url = process.env.VUE_APP_BASE_API + "/vector-map/threeDimensionalModelService/invokeMa
       const files = val.map((ite, idx) => {
         return {
            fileName: ite.file.name,
            fileUrl: ite.fileGetUrl
         }
       })
       this.syncAllUploadedFile(url, files, modelSourceInfoId)
     }

    },
    deep: true
  }
},



//同步已上传的文件到后端
syncAllUploadedFile(url, files, modelSourceInfoId){
    let xhr = new XMLHttpRequest()
    xhr.onload = () => {
      if (xhr.status === 200 && xhr.status < 300 || xhr.status === 304) {
        try {
         const res = JSON.parse(xhr.responseText)
         if(res && res.code === 200){
           this.$message({
             message: '上传完成',
             type: 'success'
           })
            this.$emit('close')
            this.fileInfoList = []
            this.fileList = []
            this.httpQueue = []
         }
        } catch(error) {
         this.$message({
           message: '上传失败，请检查服务',
           type: 'error'
         })
        }
      }
    }
    xhr.open("post", url, true)
    xhr.setRequestHeader('Content-Type', 'application/json')
    xhr.setRequestHeader('X-Access-Token', getToken())
    //将前面1.1获取文件信息的id作为头信息传递到后端
    xhr.setRequestHeader('ThreeDimensionalModel-ServiceID', modelSourceInfoId)
    xhr.send(JSON.stringify(files))
},
```

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

```js
//删除文件，并取消正在文件的上传
  deleteFile(fileInfo, index){
    this.httpQueue[index] && this.httpQueue[index].abort()
    this.httpQueue[index] && this.httpQueue.splice(index, 1)
    this.fileInfoList.splice(index, 1)
    this.fileList.splice(index, 1)
    this.removeRemoteFile(fileInfo)
  },

  //清空文件并取消上传队列
  clearFile() {
    this.fileInfoList.forEach((item, index) => {
      this.httpQueue[index] && this.httpQueue[index].abort()
      this.httpQueue[index] && this.httpQueue.splice(index, 1)
      this.removeRemoteFile(item)
    })
    this.fileInfoList = []
    this.httpQueue = []
    this.fileList = []
  },
  //删除远程文件
  removeRemoteFile(fileInfo){
    const userBucket = this.userBucket
    const { fileUploadProgress, file} = fileInfo
    const fileName = file.name
    const complete = fileUploadProgress === '100.00%' ? true : false
    if(complete){
     this. minioClient.removeObject(userBucket, fileName, function(err) {
        if (err) {
          return console.log('Unable to remove object', err)
        }
        console.log('Removed the object')
      })
    }else{
      this.minioClient.removeIncompleteUpload(userBucket, fileName, function(err) {
        if (err) {
          return console.log('Unable to remove incomplete object', err)
        }
        console.log('Incomplete object removed successfully.')
      })
    }
  },
```

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.

```js
import{uploadModelSourceInfo, uploadModelSource, getMinioConfig} from '@/api/map'
import AES from '@/utils/AES.js'
import { getToken } from '@/utils/auth'
import * as myValiDate from "@/utils/formValidate";
```

```js
/**
 * 文件说明
 * @Author: zhuds
 * @Description: 模型资源上传弹窗
    分为3个步骤
    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
    2.然后将文件上传存储桶
    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递给后端，注意，此时的请求头要戴上第1步获取的文件信息id
 * @Date: 2/28/2022, 1:13:20 PM
 * @LastEditDate: 2/28/2022, 1:13:20 PM
 * @LastEditor:
 */
<template>
  <div class="upload-model">
    <el-dialog
          :visible.sync="isVisible"
          @close="close()"
          :show-close ="false"
          :close-on-click-modal="false"
          top="10vh"
      v-if="isVisible"
      :destroy-on-close="true"
    >
          <div slot="title" class="header-title">
                  <div class="icon"></div>
                  <span>上传模型资源</span>
                  <i class="el-icon-close"  @click="close()"></i>
          </div>

      <el-form
        :label-position="labelPosition"
        label-width="80px"
        :model="uploadFormData"
        ref="form"
        :rules="rules"
      >
        <el-form-item label="别名">
          <el-input   size="small" v-model="uploadFormData.alias"></el-input>
        </el-form-item>

        <el-form-item label="备注">
          <el-input type="textarea" v-model="uploadFormData.remark"  size="small"></el-input>
        </el-form-item>

        <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.glb、.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>

      </el-form>

      <div class="file-list" v-show="fileInfoList.length > 0">
        <div class="file-item" v-for="(item, index) in fileInfoList" :key="index">
          <div class="icon"></div>
          <div class="name">{{item.file.name}}</div>
          <div class="size">{{(item.file.size/1024/1024).toFixed(2)}}MB  </div>
          <div class="progress">
            <div class="bar" :stsyle="{width: item.fileUploadProgress}"></div>
          </div>
          <div class="rate">{{item.fileUploadProgress}}</div>


          <div class="delete-btn" @click="deleteFile(item, index)">x</div>
        </div>
      </div>
      <div class="custom-footer">
        <button class="info" @click="close()">取 消</button>
        <button class="success" @click="handleUplaod()">上传</button>
      </div>

    </el-dialog>
  </div>
</template>

<script>

  import{uploadModelSourceInfo, uploadModelSource, getMinioConfig} from '@/api/map'
  import AES from '@/utils/AES.js'
  import { getToken } from '@/utils/auth'
  import * as myValiDate from "@/utils/formValidate";
  let Minio = require('minio')

  export default {
    name: 'UploadModelDialog',
    props: {
      isVisible: {
        type: Boolean,
        default: false
      },
    },
    data(){
      return {
        labelPosition: 'right',
        uploadFormData: {
          alias: '', //服务名称
          remark: '', //备注
        },
        rules: {
          serviceName: [{
           validator: myValiDate.validateServiceName,
           trigger: "blur",
           required: true,
          }],
        },
        acceptFileType:".glb,.gltf,.fbx,.obj,.mtl,.hdr,.png,.jpg, .mp4",
        fileList:[], //待上传的文件列表
        fileInfoList: [], //格式化后的文件信息列表
        userBucket: null,
        httpQueue: [], //上传文件的http队列
        allFileHasUpload: false, //是否完成上传
        modelSourceInfoId: '', //模型资源基本信息的id
      }
    },

    watch:{
      fileInfoList: {
        handler(val){
          //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
          if(val.length < 1) return
          let allFileHasUpload = val.every((item, index) => {
            return item.fileGetUrl.length > 1
          })
         if(allFileHasUpload) {
           this.allFileHasUpload = allFileHasUpload
           const {modelSourceInfoId} = this
           if(modelSourceInfoId.length < 1) {
             return
           }
           const url = process.env.VUE_APP_BASE_API + "/vector-map/threeDimensionalModelService/invo
           const files = val.map((ite, idx) => {
             return {
                   fileName: ite.file.name,
                   fileUrl: ite.fileGetUrl
             }
           })
           this.syncAllUploadedFile(url, files, modelSourceInfoId)
         }

        },
        deep: true
      }
    },

    created() {
      this.initMinioClient()
    },

    beforeDestroy() {
      if(!this.allFileHasUpload) {
        this.clearFile()
      }
    },

    methods:{
       //创建存储桶
      async initMinioClient(){
```

```
      const { code, result, message } =  AES.decryptToJSON(await getMinioConfig({}))
      if(!result || code !== 200) {
         this.$customMessage.error({message: '获取存储桶配置信息出错'})
         return false
      }
      let {accessKey, bucketName, endPoint, secretKey} = result
      //console.log({accessKey, bucketName, endPoint, secretKey})

      let endPointStr = endPoint.split(":")[1]
      let formatPort = Number(endPoint.split(":")[2])
      let formatEndPoint = endPointStr.split('//')[1]
      this.userBucket = bucketName
      this.minioClient = new Minio.Client({
        useSSL: false,
        partSize: '20M',
        port: formatPort,
        endPoint: formatEndPoint,
        accessKey,
        secretKey
      });
      let userBucket = this.userBucket
      //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
      this.minioClient.bucketExists(userBucket, (err)=> {
        if (err && err.code == 'NoSuchBucket') {
          this.minioClient.makeBucket(userBucket, 'us-east-1', (err)=> {
            if (err) {
              return console.log('创建存储桶失败', err)
            }
           // console.log('Bucket created successfully in "us-east-1".')
          })
        }else{
          //console.log('存储桶存在')
        }
      })
    },

    close(flag = false) {
      this.$emit('close', flag)
      //关闭弹窗时，如果文件没有上传完成，则清空文件
      if(!this.allFileHasUpload) {
        this.clearFile()
      }
    },
    selectFile() {
      let inputDOM = this.$refs.uploadInput
      inputDOM.click();
    },

    getFileSize(file){
      let fileSize = ''
      if(file.size / 1024 < 1){
        fileSize = file.size + 'B'
      }else if(file.size / 1024 /1024 < 1){
        fileSize = file.size + 'KB'
      }else if(file.size / 1024 /1024 >=1){
         fileSize = file.size + 'MB'
      }else{
      }

      return fileSize

    },

    //删除文件，并取消正在文件的上传
    deleteFile(fileInfo, index){
      this.httpQueue[index] && this.httpQueue[index].abort()
      this.httpQueue[index] && this.httpQueue.splice(index, 1)
      this.fileInfoList.splice(index, 1)
      this.fileList.splice(index, 1)
      this.removeRemoteFile(fileInfo)
    },

    //清空文件并取消上传队列
    clearFile() {
      this.fileInfoList.forEach((item, index) => {
        this.httpQueue[index] && this.httpQueue[index].abort()
        this.httpQueue[index] && this.httpQueue.splice(index, 1)
        this.removeRemoteFile(item)
      })
      this.fileInfoList = []
      this.httpQueue = []
      this.fileList = []
    },
    //删除远程文件
    removeRemoteFile(fileInfo){
      const userBucket = this.userBucket
      const { fileUploadProgress, file} = fileInfo
      const fileName = file.name
      const complete = fileUploadProgress === '100.00%' ? true : false
      if(complete){
       this. minioClient.removeObject(userBucket, fileName, function(err) {
          if (err) {
            return console.log('Unable to remove object', err)
          }
          console.log('Removed the object')
        })
      }else{
        this.minioClient.removeIncompleteUpload(userBucket, fileName, function(err) {
          if (err) {
            return console.log('Unable to remove incomplete object', err)
          }
          console.log('Incomplete object removed successfully.')
        })
      }
    },
    //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
        }
        //创建文件上传的url并格式化文件信息
        this.minioClient.presignedPutObject(userBucket, file.name, 24 * 60 * 60, (err, presignedUr
          if (err) {
            this.$message({
              message: `服务器连接超时`,
              type: 'error'
            })
            return err
          }
          let fileIcon = this.getFileIcon(file)
          let fileUploadProgress = '0%'
          this.fileInfoList.push({
            file, //文件
            fileIcon, //文件对应的图标 className
            fileUploadProgress, //文件上传进度
            filePutUrl: presignedUrl, //文件上传put方法的url
            fileGetUrl: '', //文件下载的url
          })
        })
      })
      this.fileList = [...this.fileInfoList]
    },

    //1.上传文件到存储桶
    async handleUplaod(){
      let _this = this
      if(this.fileInfoList.length < 1) {
        this.$message({
          message: `请先选择文件`,
          type: 'warning'
        })
        return
      }
      //1.1先上传文件的基本表单信息，获取文件信息的id
      try{
        const {remark, alias} = _this.uploadFormData
        let res = await uploadModelSourceInfo({remark, serviceName: alias})
        _this.modelSourceInfoId = res.message
      }catch(error){
        if(error) {
          _this.$message({
            message: `上传失败，请检查服务`,
            type: 'error'
          })
          return
        }
      }

      //1.2开始将模型资源上传到远程的存储桶
      this.fileList.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)
          },
          onError: ({error, errorFile}) => {

          }
        }

        let httpInstance = this.createUploadHttp(config)
        this.httpQueue.push(httpInstance)
      })
    },

    //1更新对应文件的上传进度
    updateFileUploadProgress(uploadingFile, progress) {
      //console.log({uploadingFile, progress})
      this.fileInfoList.forEach((item, index) => {
        if(item.file.name === uploadingFile.name){
          item.fileUploadProgress = (Number(progress)*100).toFixed(2) + '%'
        }
      })
    },

    //更新上传完成文件的下载地址
    updateFileDownloadUrl(uploadedFile){
      const userBucket = this.userBucket
      this.fileInfoList.forEach((item, index) => {
        if(item.file.name === uploadedFile.name){
          this.minioClient.presignedGetObject(userBucket, uploadedFile.name, 24*60*60, (err, presi
            if (err) return console.log(err)
            item.fileGetUrl = presignedUrl
           // console.log(presignedUrl)
          })
        }
      })
    },

    //同步已上传的文件到后端
    syncAllUploadedFile(url, files, modelSourceInfoId){
      let xhr = new XMLHttpRequest()
      xhr.onload = () => {
        if (xhr.status === 200 && xhr.status < 300 || xhr.status === 304) {
          try {
            const res = JSON.parse(xhr.responseText)
            if(res && res.code === 200){
              this.$message({
                message: '上传完成',
                type: 'success'
              })
              // setTimeout(() => {
                this.$emit('close')
                this.fileInfoList = []
                this.fileList = []
                this.httpQueue = []
```

```
            }
          } catch(error) {
            this.$message({
              message: '上传失败，请检查服务',
              type: 'error'
            })
          }
        }
      }
      xhr.open("post", url, true)
      xhr.setRequestHeader('Content-Type', 'application/json')
      xhr.setRequestHeader('X-Access-Token', getToken())
      //将前面1.1获取文件信息的id作为头信息传递到后端
      xhr.setRequestHeader('ThreeDimensionalModel-ServiceID', modelSourceInfoId)
      xhr.send(JSON.stringify(files))
    },

    //获取文件类型图标class
    getFileIcon(file) {
      const { type } = file
      let icon = ''
      return icon
    },

    //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) => {
        onUploadProgress({
          progressEvent: e,
          uploadingFile: file
        })
      }, false)

      http.onload = () => {
        if (http.status === 200 && http.status < 300 || http.status === 304) {
          try {
            const result = http.responseURL
            onUploaded({ result, uploadedFile: file})
          } catch(error) {
            onError({ error, errorFile: file})
          }
        }
      }
      http.open("PUT", filePutUrl, true);
      Object.keys(customHeader).forEach((key, index) =>{
        http.setRequestHeader(key, customHeader[key])
      })
      http.send(file);
      return http
    }

  }
}
</script>

<style scoped lang="scss">

.header-title {
//  height: 24px;
  border-bottom: 1px solid #EFEFEF;
  padding-bottom: 20px;
  //outline: 1px solid red;
  .icon {
    width: 26px;
    height: 26px;
    background-image: url(../images/icon_upload.png);
    float: left;
    background-size: 100% 100%;
    margin-right: 18px;
    margin-left: 10px;
  }
  span {
    font-size: 23px;
    font-family: Source Han Sans CN;
    font-weight: 500;
    color: #333333;
  }
  i {
    float: right;
    font-size: 16px;
    color: rgba(176, 176, 176, 1);
    cursor: pointer;
    &:hover {
      color: #0069D5 ;
    }
  }
}
.tip {
  font-size: 12px;
  display: block;
}
.file-list {
  box-sizing: border-box;
  padding: 5px;
 // border: 1px solid red;
 // width: 840px;
  margin: 5px auto;
  margin-left: 80px;
  .file-item {
    min-height: 32px;
    display: flex;
    justify-content: flex-start;
    align-items: center;
    font-size: 12px;
    div {
      margin-right: 15px;
      text-align: left;
    }
    .name {
      width: 200px;
    }
    .size {
      width: 60px;
    }
    .progress {
      width: 180px;
      height: 8px;
```

```
      border-radius: 4px;
      background-color: #E2E2E2;
      .bar {
        width: 50%;
        height: 8px;
        border-radius: 4px;
        background-color: #13A763;
      }
    }
    .rate {
      width: 60px;
     // border: 1px solid red;

    }
    .delete-btn {
      cursor: pointer;
      font-size: 16px;
    }
  }
}

.custom-footer {
 // border: 1px solid red;
  display: flex;
  justify-content: space-evenly;
  align-items: center;
  width: 100%;
  height: 80px;
  background-color: #fff;
  //box-shadow: 0px 1px 0px 0px red;
  border-top: 1px solid #efefef;
  z-index: 10;
  button {
    width: 90px;
    height: 40px;
    border-radius: 4px;
    border: 0;
    font-size: 16px;
    font-family: Source Han Sans CN;
    &:focus {
      border: 0;
      outline: 0;
    }

    &.info {
      color: #8f8f8f;
      background: #e2e2e2;
      &:hover{
        background-color: #A6A9AD;
        cursor: pointer;
        color: #fff;
      }
    }
    &.success {
      background: #12a763;
      color: #fff;
      &:hover{
        cursor: pointer;
        background-color: #73C132;
      }
    }
  }
}
</style>
```

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：

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 11月前

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，瓜分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

点赞 回复

1年前

vite+vue3中怎么使用

点赞 10

清风夜半 （作者） 12月前

一样使用啊，基本全是js代码

点赞 回复

回复 **清风夜半** 11月前

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”

点赞 回复

查看更多回复

zyh619 开发人员 @ 武汉 1年前

老哥，有试过vite使用minio吗？minio里面的依赖各种报错

点赞 4

照古今 9月前

兄弟，解决了吗?

点赞 回复

照古今 9月前

你解决了吗，老哥，我也遇到了

点赞 回复

查看更多回复

LetMeTouchTou... 1年前

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

点赞 1

清风夜半 （作者） 1年前

文章中为了讲解我明写了，项目中是通过加密服务获取的

点赞 回复

知北 前端工程师 1年前

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

点赞 1

耗子会飞 1年前

必须要控制的呀，不然大并发请求，直接崩了

点赞 回复

無思童 @ 1年前

看起来挺复杂的

点赞 2

清风夜半 （作者） 1年前

里面代码有需要封装的地方，还能进一步优化

点赞 回复

耗子会飞 1年前

分片上传，前端工作量很大，后端没什么压力

点赞 回复

用户646887783... 1年前

几个JS文件能提供一下参考学习，感谢。

点赞 回复

最好沉默 frontend @ yellow lem... 1年前

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

点赞 2

清风夜半 （作者） 1年前

是的，存储服务是后端搭建的，部署在云服务器

点赞 回复

耗子会飞 1年前

前端不做完整性验证

点赞 回复

迎风破浪 Java开发 @ 新时达 1年前

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 2

清风夜半 (作者) 1年前

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 1年前

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 前炉石美服第一 星际2... 1年前

有兴趣聊聊吗 你可以看下我沸点

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 83 14

Anxyh | 1年前 | Java

Minlo 最佳分片上传PLUS

7753 35 40

泉城老铁 | 2年前 | Spring Boot

springboot +vue 集成minio上传文件

1831 11 8

高蓓蓓 | 12月前 | 前端

Vue文件下载进度条

6487 113 22

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 8162 289

躺平的sx | 3月前 | Java

Java构建minio文件系统实现分片上传

1496 17 2

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 6157 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 383 5

tinyOrange | 2 years ago | Big data

Minio architecture analysis

6647 6 comment

Xiling | Before November | front end · JavaScript

Try S3+minio to upload large files

1857 9 4

Meow | August | front end

TypeScript upload file to MinIO

869 5 comment

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

minio quick entry

5830 twenty four comment

alwaysVe | 3 years ago | Node.js

Nodejs file upload, monitor upload progress

3983 twenty one comment

OrzR3 | 1 year ago | front end · Vuex

Vue project download progress plan

5897 42 4