- 图床服务(picture hosting service，不是 “picture bed” 哦~)，是web服务中常见常用的服务。其功能在于允许用户将图片放在服务器上并提供公开可访问的链接。

  有了上一节(【Node.js】使用Multer的文件上传下载)的铺垫，这一节就来小试牛刀，制作一个简易的仅支持上传和浏览的图床应用了

  # 需求分析

  首先，提供上传图片的接口。在上传图片后，需要返回给用户图片的url以便于用户的访问。

  简单起见，我们约定，所有图片都存放在项目根目录下的`imgs`文件夹中，url中格式为`/img/图片名`前缀的请求即表示获取相应图片

  对于上传文件，设计两个接口：

  1. 上传单张图片，随机生成名称，返回url路径(`/upload/singleImage`)
  2. 同一上传多张图片，最多不超过9张，随机名称，返回成功存储图片信息列表，包含其url路径(`/upload/multiImages`)

对所有的图片要求：

- 大小不能超过20K（$20 \times 1024 = 20480$字节）
- 格式必须为以下几种之一,不区分大小写：
  -- jpg
  -- jpeg
  -- gif
  -- png
  -- webp
  -- bmp
- 转存结束后要删除缓存

浏览

在实现上传功能之前，有必要先将浏览图片的功能配置好，以便于测试使用。

```
var express = require('express')
var app = express()

//路由为/img的请求映射到./imgs文件夹下的静态文件
app.use('/img',express.static('./imgs'))
```

上传

完整代码如下：

```
var express = require('express')
var multer = require('multer')
var fs = require('fs')

var app = express()
var upload = multer({//设置文件缓存地址
    dest:'./temp'
```

```
})

/* 全局常量 */
const HOST = 'localhost'              //服务器地址
const PORT = 8080                     //端口号
const SUFFIXES = {                    //合法后缀
    png:true,
    jpg:true,
    jpeg:true,
    bmp:true,
    webp:true,
    gif:true
}
const MAX_SIZE = 20480                //文件大小上限
const IMAGE_URL = 'img'               //图片url前缀
const IMAGE_DIRECTORY = './imgs'      //本地保存图片路径

app.use('/img',express.static('./imgs'))    //配置静态资源

/* 工具方法 */

//校验后缀
function checkSuffix(suffix){
    return SUFFIXES[suffix]
}

//校验文件大小
function checkSize(size){
    return size <= MAX_SIZE
}

//构造返回结果
function resultMessage(code,message,data = null){
    return {
        code:code,
        message:message,
        data:data
    }
}

//正确返回结果
function ok(){
    return resultMessage(200,'ok')
}

//带数据的正确返回结果
function ok(data){
    return resultMessage(200,'ok',data)
}

//错误返回结果
function err(message){
    return resultMessage(400,message)
}

//指定长度随机小写字母字符串
function randomStr(len){
    let name = ''
```

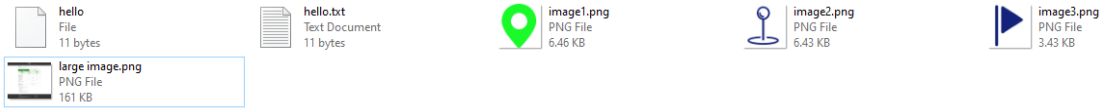
```
    while(len-- > 0){
        name += String.fromCharCode(Math.floor(Math.random() * 26) + 97)
    }
    return name
}

//获取随机文件名（当前时间戳-六位随机字符串）
function generateRandomFileName(){
    let name = ''
    name += new Date().getTime()
    name += '-' + randomStr(5)
    return name
}

//删除文件
function deleteFile(file){
    console.log('删除文件：',file)
    fs.unlinkSync(file)
}

//接受单个文件
app.post('/upload/singleImage',upload.single('file'),(req,res)=>{

    console.log(req.file)

    let file = req.file

    if(file == undefined){
        res.send(err('未检测到文件！'))
        res.end()
        return
    }


    let originalName = file.originalname

    //校验文件名称格式
    if(originalName.split('.').length != 2){
        res.send(err('图片名称格式错误！'))
        res.end()
        return
    }


    let suffix = originalName.split('.')[1]

    //校验文件大小
    if(!checkSize(file.size)){
        res.send(err('图片过大！请确保图片大小在20k以内！'))
        res.end()
        return
    }

    //校验文件后缀
    if(!checkSuffix(suffix)){
        res.send(err('图片格式错误！'))
        res.end()
        return
```

```
    }

    //转存文件
    let tempFile = file.path

    let fileName = generateRandomFileName();
    let fullFileName = `${fileName}.${suffix}`
    let filePath = `${IMAGE_DIRECTORY}/${fullFileName}`

    fs.readFile(tempFile,(err,data)=>{
        if(err){
            res.send(err('图片保存错误! '))
            res.end()
            return
        }

        fs.writeFileSync(filePath,data)
    })

    //构造url并返回
    let url = `http://${HOST}:${PORT}/${IMAGE_URL}/${fullFileName}`
    res.send(ok(url))
    res.end()

    //删除缓存文件
    deleteFile(tempFile)
    return
})

//接收多个文件
app.post('/upload/multiImages',upload.array('files',9),(req,res)=>{
    console.log(req.files)

    res.set({
      'content-type': 'application/json; charset=utf-8'
    })

    let files = req.files

    if(files == undefined){
        res.send(err('未接收到文件! '))
        res.end()
        return
    }

    //返回结果集
    let results = []

    //遍历处理文件
    for(let idx in files){
        let file = files[idx]
        let tempFile = file.path
        result = {
            name:file.originalname,
            url:'',
            err:''
        }
        results[idx] = result
```

```
        //校验文件名称格式
        let originalName = file.originalname
        if(originalName.split('.').length != 2){
            result.err = '图片名称格式错误！'
            deleteFile(tempFile)
            continue
        }

        //校验文件后缀
        let suffix = originalName.split('.')[1]
        if(!checkSuffix(suffix)){
            result.err = '图片类型错误！'
            deleteFile(tempFile)
            continue
        }

        //校验文件大小
        if(!checkSize(file.size)){
            result.err = '图片过大！请确保图片大小在20k以内！'
            deleteFile(tempFile)
            continue
        }

        //转存文件

        let fileName = generateRandomFileName();
        let fullFileName = `${fileName}.${suffix}`
        let filePath = `${IMAGE_DIRECTORY}/${fullFileName}`

        let flag = true
        fs.readFile(tempFile,(err,data)=>{
            if(err){
                result.err = "图片保存错误！"
                flag = false
            }else{
                fs.writeFileSync(filePath,data)
            }


        })


        //构造url并填写信息列表
        let url = `http://${HOST}:${PORT}/${IMAGE_URL}/${fullFileName}`

        if(flag){
            result.url = url
        }

        //删除缓存文件
        deleteFile(tempFile)
    }

    //返回信息列表
    res.send(ok(results))
    res.end()
```

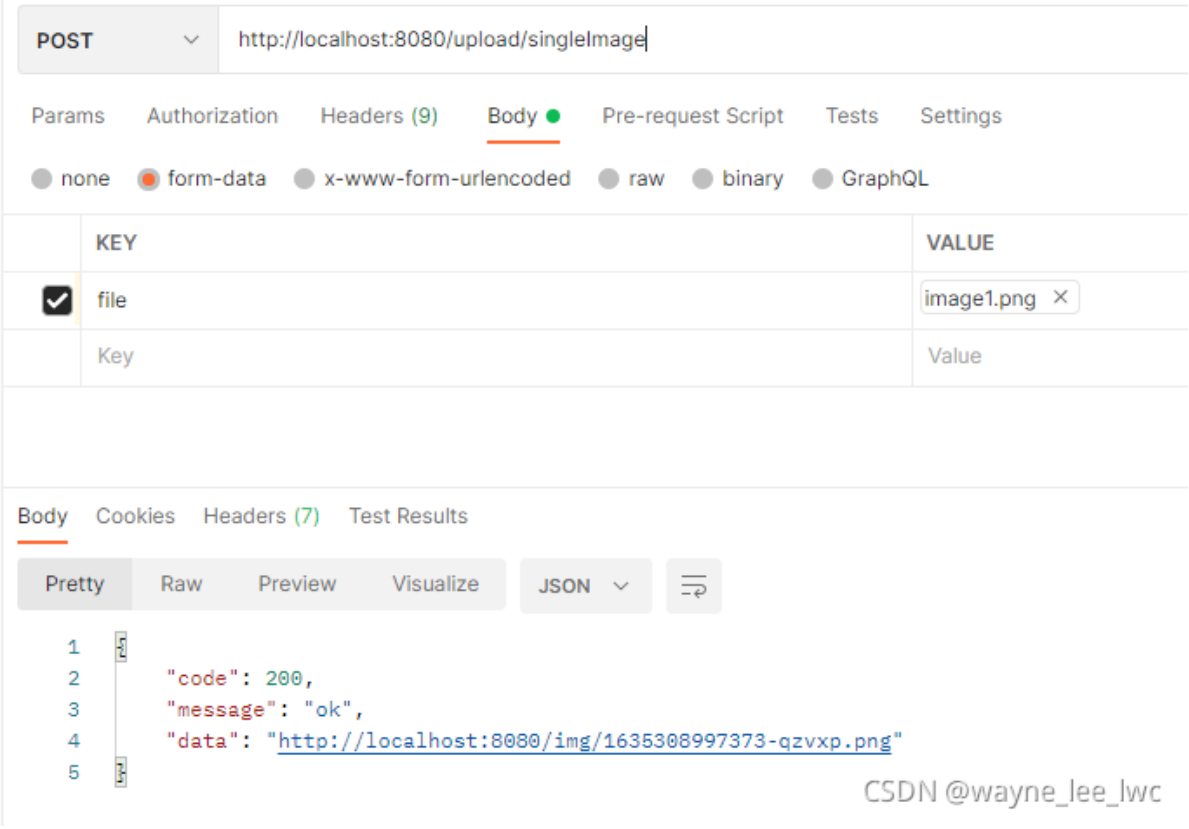
```
    return
})

var server = app.listen(PORT,()=>{
    console.log(`picture hosting service is listening on port ${PORT}`)
})
```

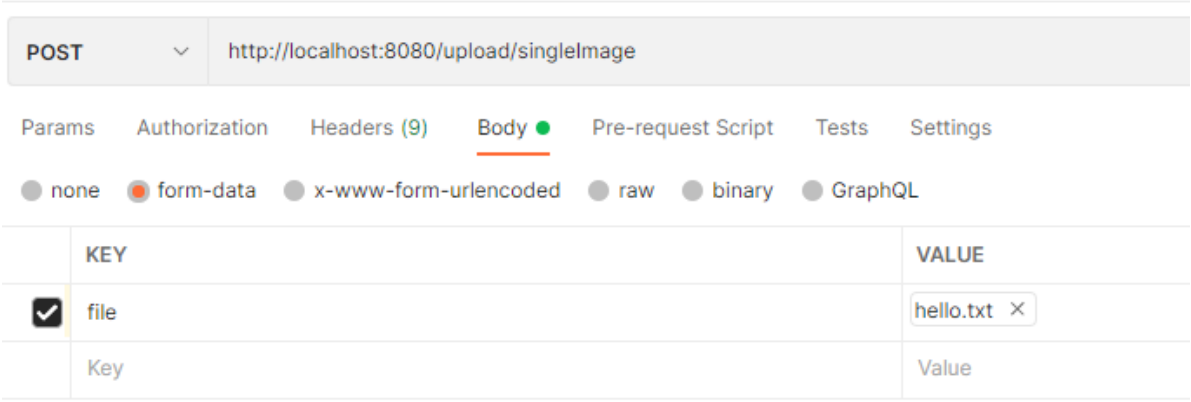
准备一些用于上传测试的图片：

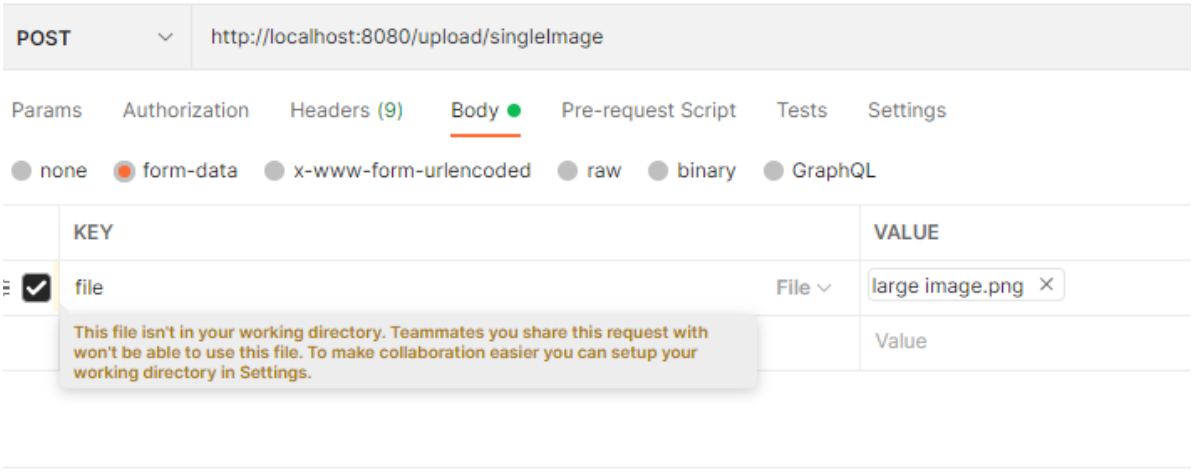


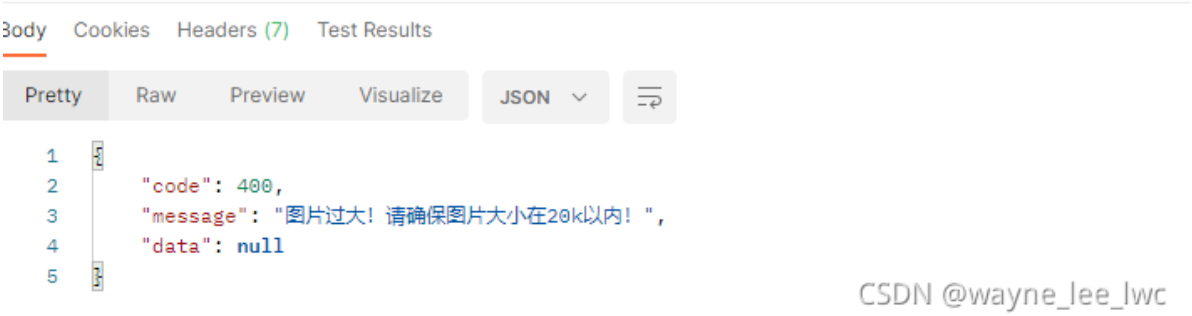
其中有三个正常的，两个格式错误的，还有一个超过大小限制的

先使用postman发送上传单个文件试一试：

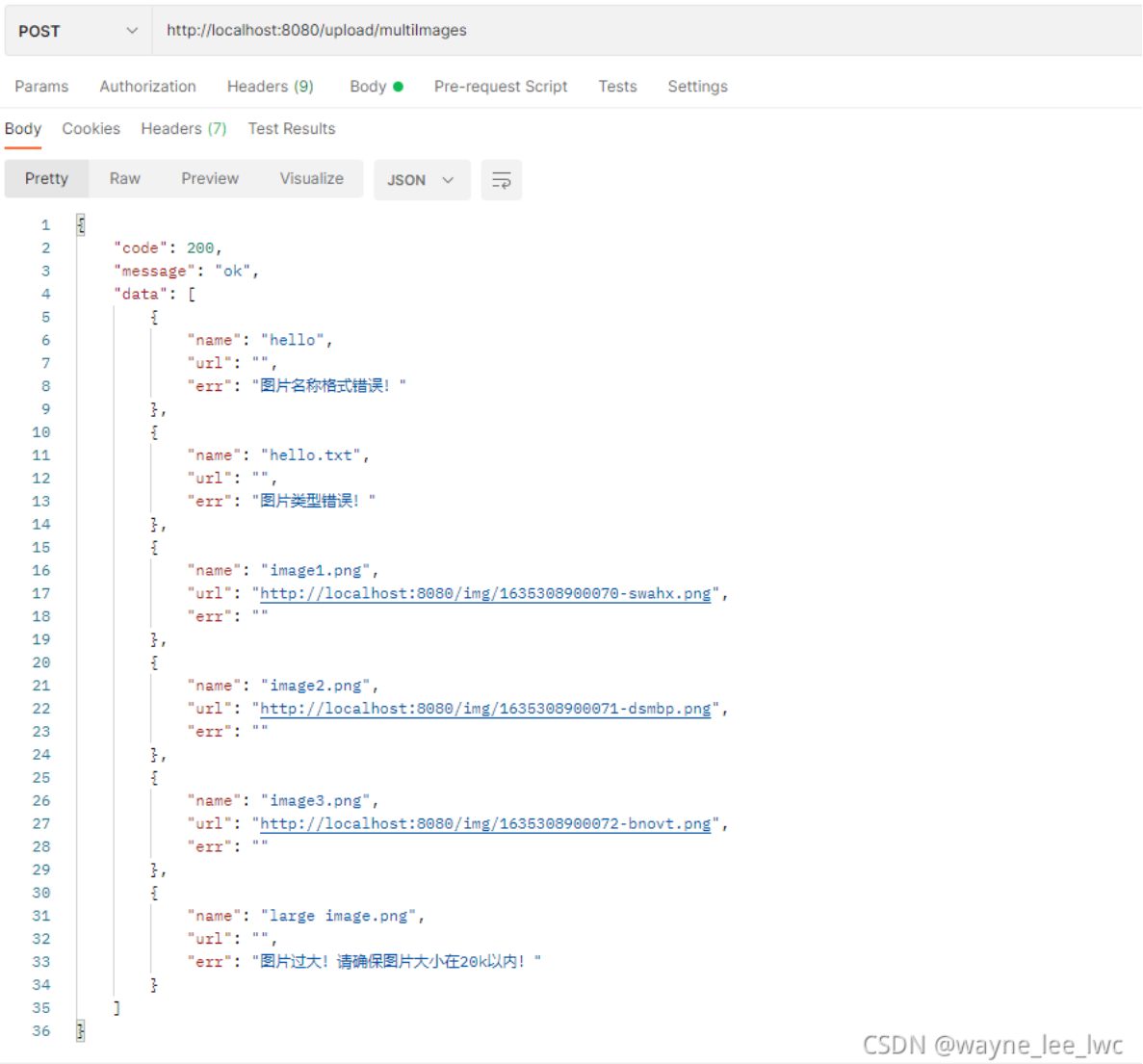








再试试多文件上传接口，将6张图片全部上传：



到此为止，我们简易图床应用就完成了！

本文用到的代码已整理，可供小伙伴们下载：

- github仓库
- csdn资源

参考资料

- Multer 模块 npm 首页
- 《深入浅出Node.js》
- 《Learn NodeJS in 1 Day》
- 《The node craftsman book》
- 《MERN Projects for Beginners》

往期内容

- 【Node.js】下载安装及简单使用
- 【Node.js】Express搭建服务端应用及接收请求参数
- 【Node.js】使用Multer的文件上传下载