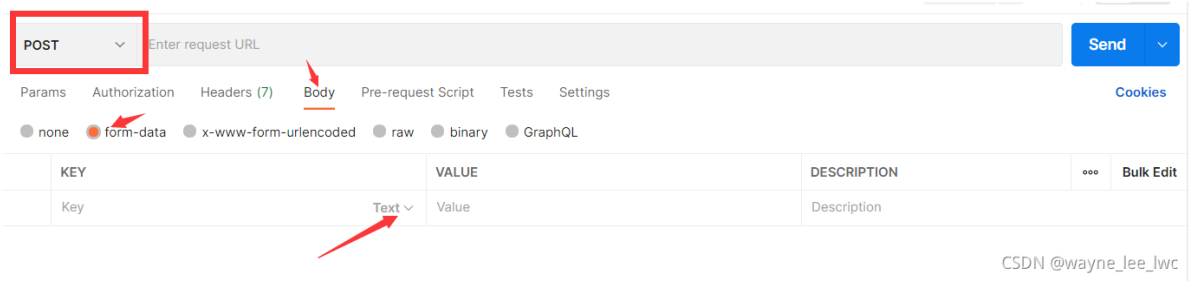
上回说到，使用Express框架搭建起了简易的后端，并可以使用它接受各种形式的参数。

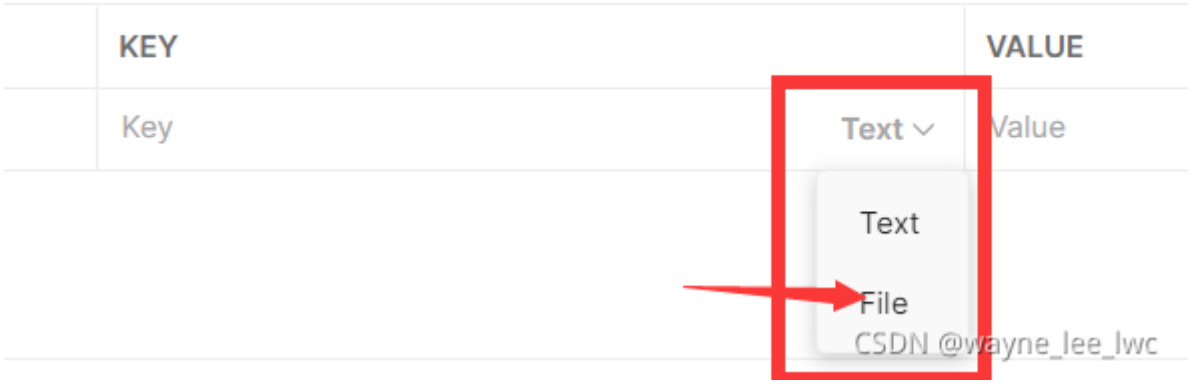
这一节，我们来解决一个web开发中非常基础但也是非常重要的需求——文件上传与下载问题

文件上传

上传一个文件，使用的http的请求方法是post，请求头中内容类型（Content-type）为`multipart/form-data`。现在为了测试后台接口，我们使用postman进行post请求发送：



- 更改请求方法
- 选中Body
- 选择form-data格式
- 下方参数处可以修改参数类型为文件



Multer中间件

以http请求发送出去的文件，会以二进制流的格式通过网络到达服务器，但显然直接处理它会很不方便，尤其是在一个请求中包含多个文件以及其他参数时。

实际上，我们更希望可以将上传的文件在进入请求处理前被接受并缓存并将有关他的信息封装在一个对象之中便于后续的操作。

那么今天的主角——`Multer`中间件就可以帮助我们很好的完成这些需求。它可以将multipart/form-data类型请求中的图片接收并进行过滤，封装，便于在业务处理时使用。

（更详细介绍可以查看Multer的NPM官网文档，下文内容大多参考自该文档）

首先，我们来安装Multer并在代码中引用他：

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

multer配置

在使用之前，需要生成一个multer的实例并对其进行配置：

```
var upload = multer({
    dest:'./temp'
})
```

`upload`是根据配置产生的multer对象，在之后的请求处理中使用它。配置是以对象的形式传入的，最常用的就是这个dest，表示文件缓存的磁盘地址。配置后，接受的文件会直接写在磁盘中充当临时文件。如果没有配置这个内容，则回将文件以buffer的形式保存在内存中。

除了dest，还有其他可配置选项，不过通常应用中，我们只需配置dest：

Key	Description
`dest` or `storage`	Where to store the files
`fileFilter`	Function to control which files are accepted
`limits`	Limits of the uploaded data
`preservePath`	Keep the full path of files instead of just the base name

完整代码：

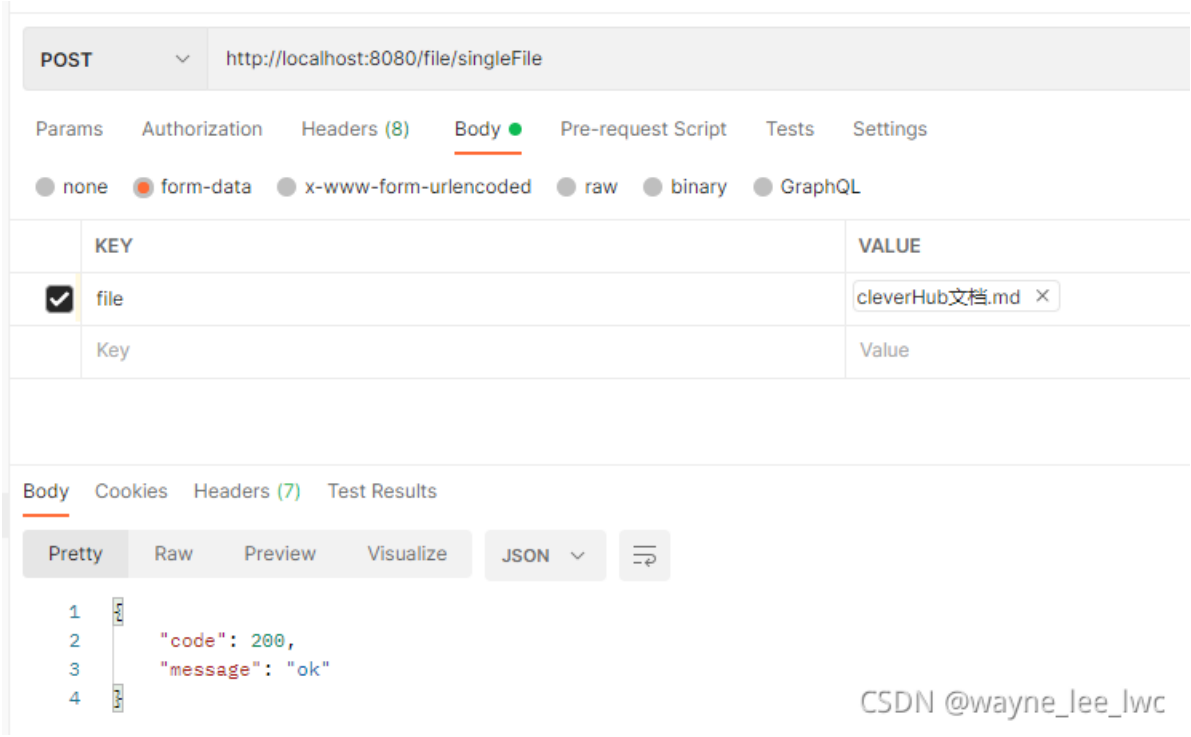
```
var multer = require('multer')

var upload = multer({
    dest:'./temp'
})
```

单文件

传递单文件时，需要在请求处理中添加`upload.single(fileName)`中间件，例如：

```
app.post('/file/singleFile',upload.single('file'),(req,res)=>{
    console.log('Received a request with file:');
    console.log(req.file);
    res.send({
        code:200,
        message:'ok'
    });
})
```

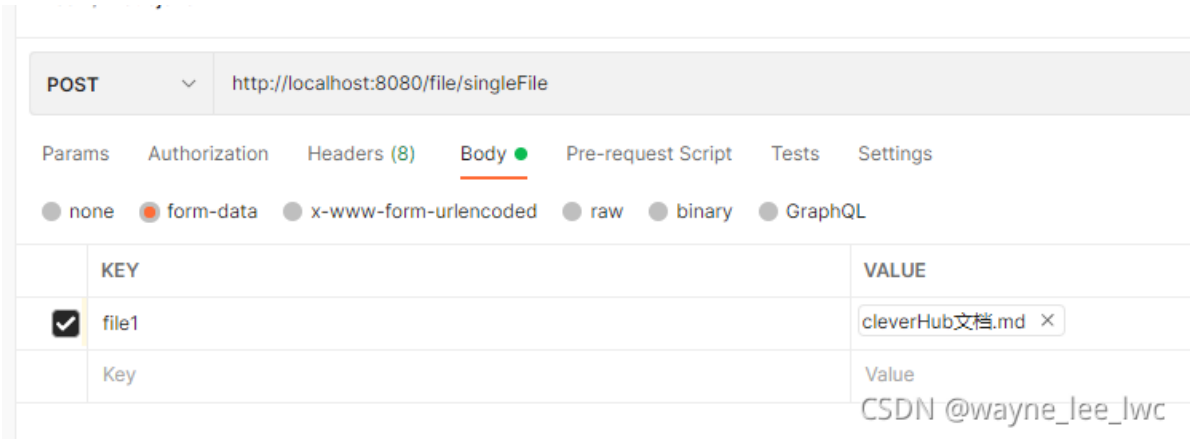
此时，文件就会被处理好封装在req的file中，我们使用postman发送请求进行验证：



可以看到后端成功接收文件并缓存在指定目录并封装：

```
Received a request with file:
{
  fieldname: 'file',
  originalname: 'cleverHub文档.md',
  encoding: '7bit',
  mimetype: 'text/markdown',
  destination: './temp',
  filename: '87a9fadd36e91cdda5d32abc59d504ce',
  path: 'temp\\87a9fadd36e91cdda5d32abc59d504ce',
  size: 6460
}
```

如果发送的文件名称与接收名称相悖，multer就会报错：




```
MulterError: Unexpected field
    at wrappedFileFilter (D:\myData\博客代码\New folder\node_modules\multer\index.js:40:19)
    at Busboy.<anonymous> (D:\myData\博客代码\New folder\node_modules\multer\lib\make-middleware.js:114:7)
    at Busboy.emit (events.js:400:28)
    at Busboy.emit (D:\myData\博客代码\New folder\node_modules\busboy\lib\main.js:38:33)
    at PartStream.<anonymous> (D:\myData\博客代码\New folder\node_modules\busboy\lib\types\multipart.js:213:13)
    at PartStream.emit (events.js:400:28)
    at HeaderParser.<anonymous> (D:\myData\博客代码\New folder\node_modules\dicer\lib\Dicer.js:51:16)
    at HeaderParser.emit (events.js:400:28)
    at HeaderParser._finish (D:\myData\博客代码\New folder\node_modules\dicer\lib\HeaderParser.js:68:8)
    at SBMH.<anonymous> (D:\myData\博客代码\New folder\node_modules\dicer\lib\HeaderParser.js:40:12)
```

多文件

多文件上传也是我们常见的需求。有时，需要用一个参数传递多个文件，又是，需要多个参数传递多个文件。

这两种情况，multer都有相应的接受方式。

array形式

array形式可以接受一个参数多个文件，只需要调用upload的array方法即可，它的参数格式为：

```
upload.array(文件参数名，最大文件数量)
```

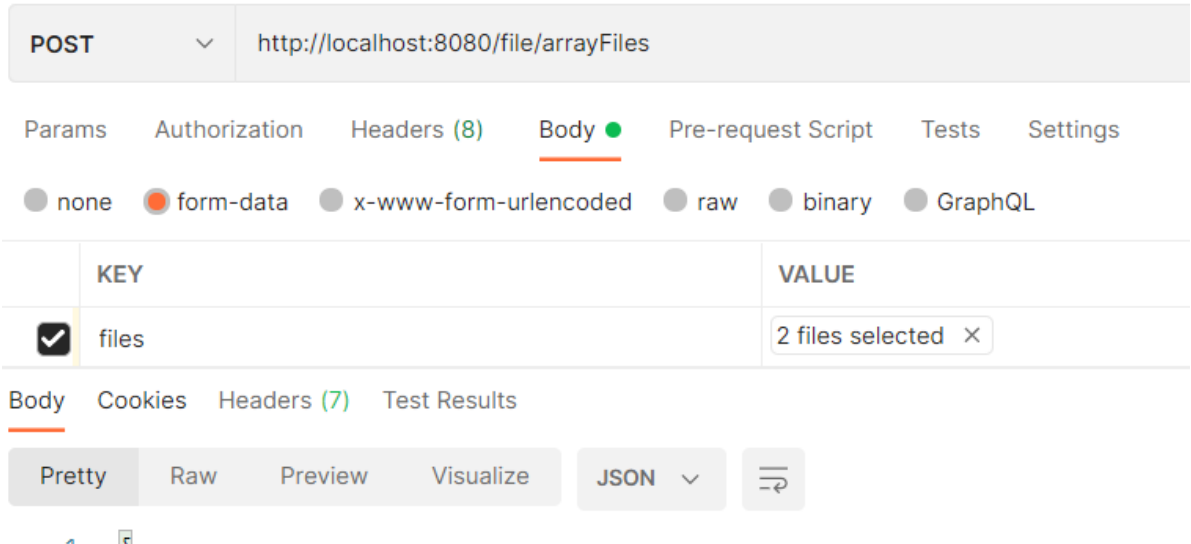
与single不同的时，array将解析出来的文件列表封装在`req.files`中而非`req.file`中

使用示例：

```
app.post('/file/arrayFiles',upload.array('files',2),(req,res)=>{
    console.log('Received a request with files:');
    console.log(req.files);
    res.send({
        code:200,
        message:'ok'
    })
})
```

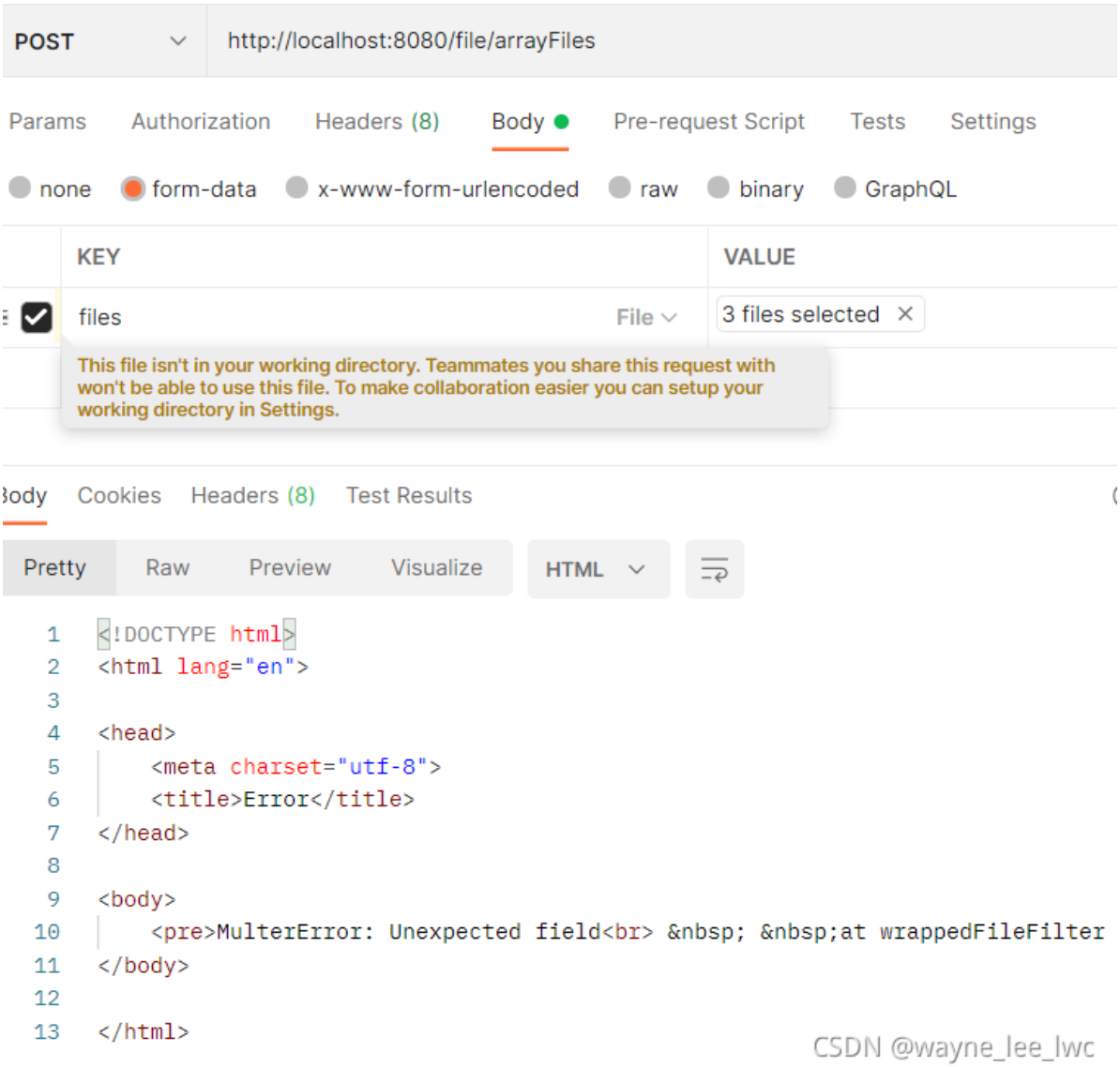
上面的示例表示改接口可以接受不超过两个文件的files参数。

使用postman发送两个文件给files：




```
Received a request with files:
[
  {
    fieldname: 'files',
    originalname: 'blog_dev_log.md',
    encoding: '7bit',
    mimetype: 'text/markdown',
    destination: './temp',
    filename: '9db2982e39ec2edffeabc4ff52f42ad6',
    path: 'temp\\9db2982e39ec2edffeabc4ff52f42ad6',
    size: 1609
  },
  {
    fieldname: 'files',
    originalname: 'cleverHub.md',
    encoding: '7bit',
    mimetype: 'text/markdown',
    destination: './temp',
    filename: '514dd630584a204ffa4b843745d81fb3',
    path: 'temp\\514dd630584a204ffa4b843745d81fb3',
    size: 6460
  }
]
```

如果发送的文件数量超过了最大文件数量，则multer会报错：



feilds形式

feilds方法相比array方法更加灵活，它允许接受多个参数的多个文件，且可以对每个参数的最大文件数量进行限制，使用格式如下：

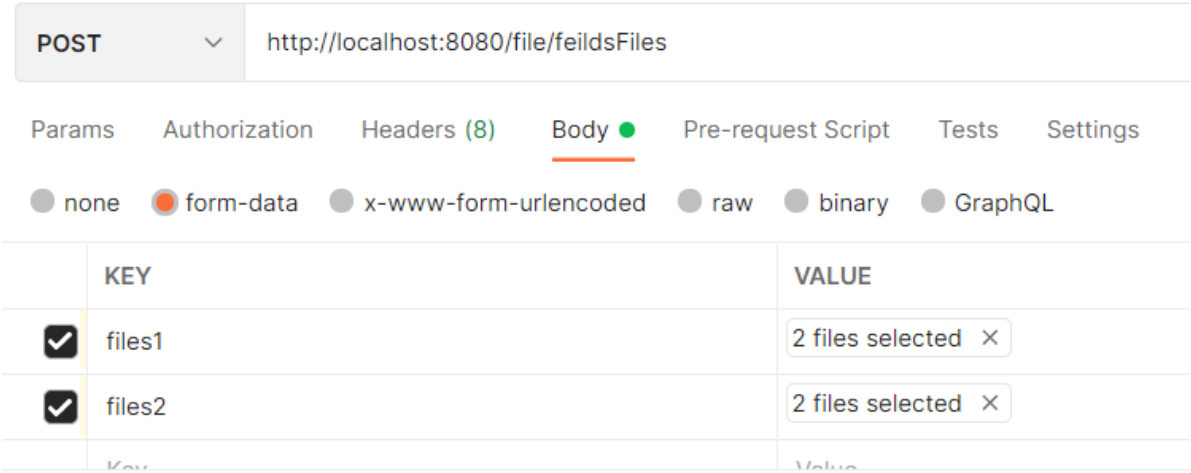
```
upload.feilds([{name:文件名称1,maxCount:最大文件1数量},{name:文件名称2,maxCount:最大文件2数量}...])
```

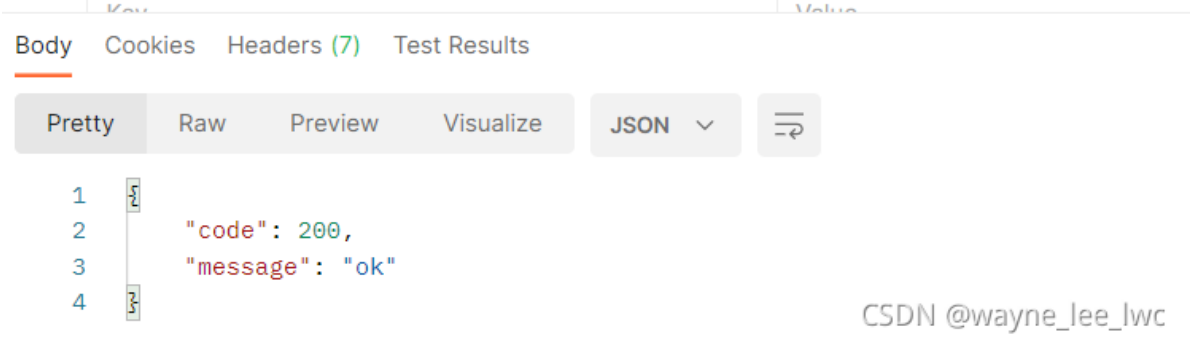
和array方法一样，feilds方法也会将接收到的文件解析在`req.files`中，不过不同的是，解析在其中的文件并不是数组的形式，以map的形式将每个参数对应文件列表的形式封装成对象

使用示例：

```
app.post('/file/feildsFiles',upload.fields([{name:'files1',maxCount:2},
{name:'files2',maxCount:2}]),(req,res)=>{
    console.log('Received a request with files')
    console.log(req.files);
    res.send({
        code:200,
        message:'ok'
    })
})
```

使用postman给files1和files2分别发送两个文件：






```
Received a request with files
[Object: null prototype] {
  files1: [
    {
      fieldname: 'files1',
      originalname: 'idictionary.md',
      encoding: '7bit',
      mimetype: 'text/markdown',
      destination: './temp',
      filename: '86c57ef3711059384863de563a343d8f',
      path: 'temp\\86c57ef3711059384863de563a343d8f',
      size: 2341
    },
    {
      fieldname: 'files1',
      originalname: 'cleverHub.md',
      encoding: '7bit',
      mimetype: 'text/markdown',
      destination: './temp',
      filename: '1397ff3cb14d0226f4c71666febe3aec',
      path: 'temp\\1397ff3cb14d0226f4c71666febe3aec',
      size: 6460
    }
  ],
  files2: [
    {
      fieldname: 'files2',
      originalname: 'blog_dev_log.md',
      encoding: '7bit',
      mimetype: 'text/markdown',
      destination: './temp',
      filename: '2cc12dd0c024659cc93828d2e614ce59',
      path: 'temp\\2cc12dd0c024659cc93828d2e614ce59',
      size: 1609
    },
    {
```

```
      fieldname: 'files2',
      originalname: 'Conputer English.md',
      encoding: '7bit',
      mimetype: 'text/markdown',
      destination: './temp',
      filename: '8c265eba1cf8b1de32dcfb4a8043b70e',
      path: 'temp\\8c265eba1cf8b1de32dcfb4a8043b70e',
      size: 5588
    }
  ]
}
```


文件下载

文件下载功能，其实就是对静态资源请求的的url路径进行解析并映射到其对应的文件上再返回的过程。

这个功能在express框架中被封装在中间件`express.static`中，其使用方法为：

```
app.use(url路径,express.static(静态文件目录))
```

这条语句可以将前面的url路径和后面的静态资源目录建立映射，之后的请求在有url路径前缀时会自动的映射到对应目录下的静态资源
使用示例，将`/image`路径映射到项目根目录下的`images`文件夹：

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

app.use('/image',express.static('./images'))
```

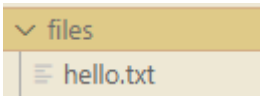
对于`express.static`的使用没有次数的限制，我们可以注册更多的静态资源映射：

```
app.use('/file',express.static('./files'))
app.use('/audio',express.static('./audios'))
app.use('/vedio',express.static('./vedios'))
app.use('/image',express.static('./images'))
```

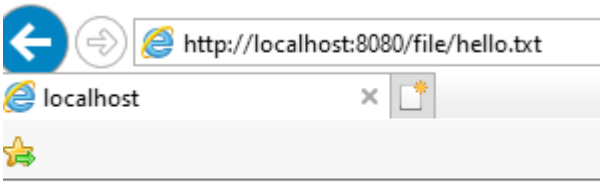
完整代码：

```


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

app.use('/file',express.static('./files'))
app.use('/audio',express.static('./audios'))
app.use('/vedio',express.static('./vedios'))
app.use('/image',express.static('./images'))

var server = app.listen(8080,()=>{
    console.log('Server is listening on port 8080')
})
```

在file下放入一个文件：



执行程序，在浏览器中访问：`http://localhost:8080/file/hello.txt`




```
hello world
```

本文用到的代码已整理好，可供小伙伴们下载：

- github仓库
- csdn资源

参考资料

- Multer 模块 npm 首页
- 《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