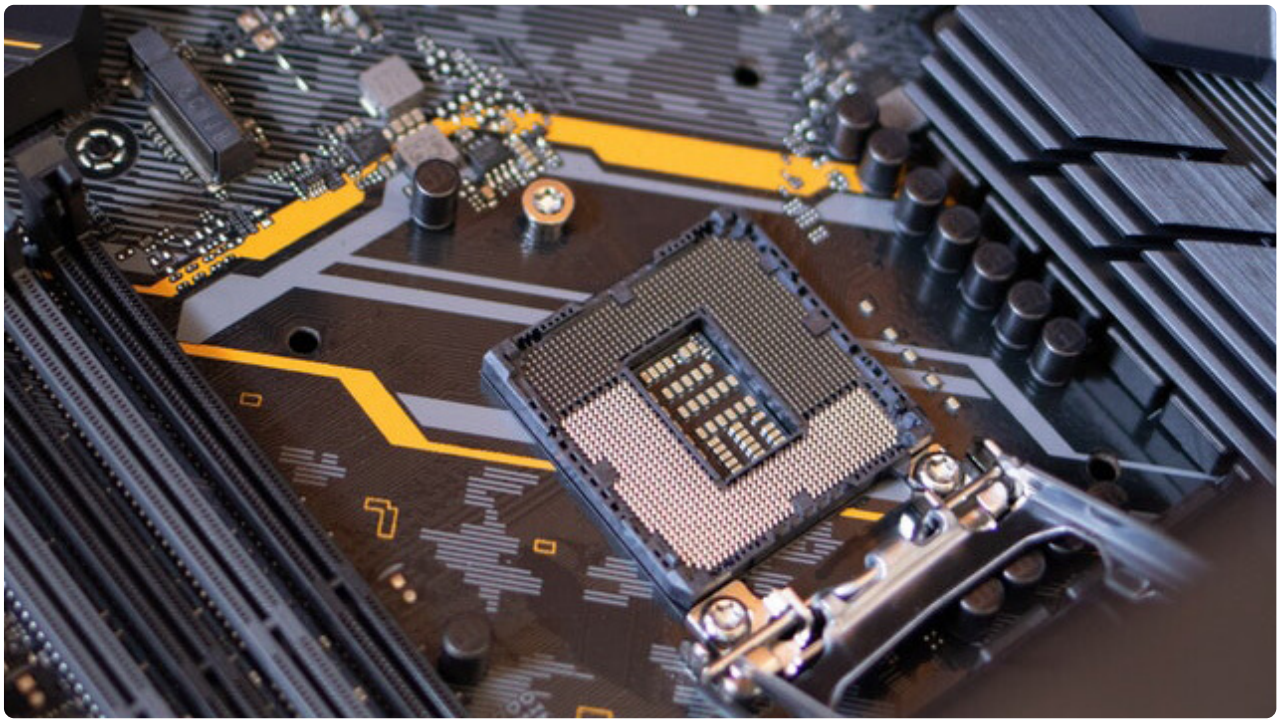
12 upload 图片上传接口

更新时间：2019-09-04 13:58:35



> 最聪明的人是最不愿浪费时间的人。
>
> ——但丁

图片上传逻辑的实际流程就是找到一个服务器，将你上传的图片储存下来并提供图片的url和尺寸，大型的web应用一般有自己独立的图片服务，并且图片服务是支持CDN功能。而一般自己开发的小型项目，基于成本考虑，也可以将图片上传到自己的服务器上，但是缺点就是无法享受到图片服务的增值功能（CDN，裁剪，滤镜等等），下面我分别就两种方案来讲解。

使用阿里云图片服务

阿里云OSS图片处理服务（Image Processing，简称 IMG），是阿里云OSS对外提供的海量、安全、低成本、高可靠的图片处理服务。您可以将原始图片上传保存在 OSS 上，通过简单的 RESTful 接口，在任何时间、任何地点、任何互联网设备上对图片进行处理。图片处理服务提供图片处理接口，图片上传请使用 OSS 上传接口。基于IMG，您可以搭建出跟图片相关的服务。

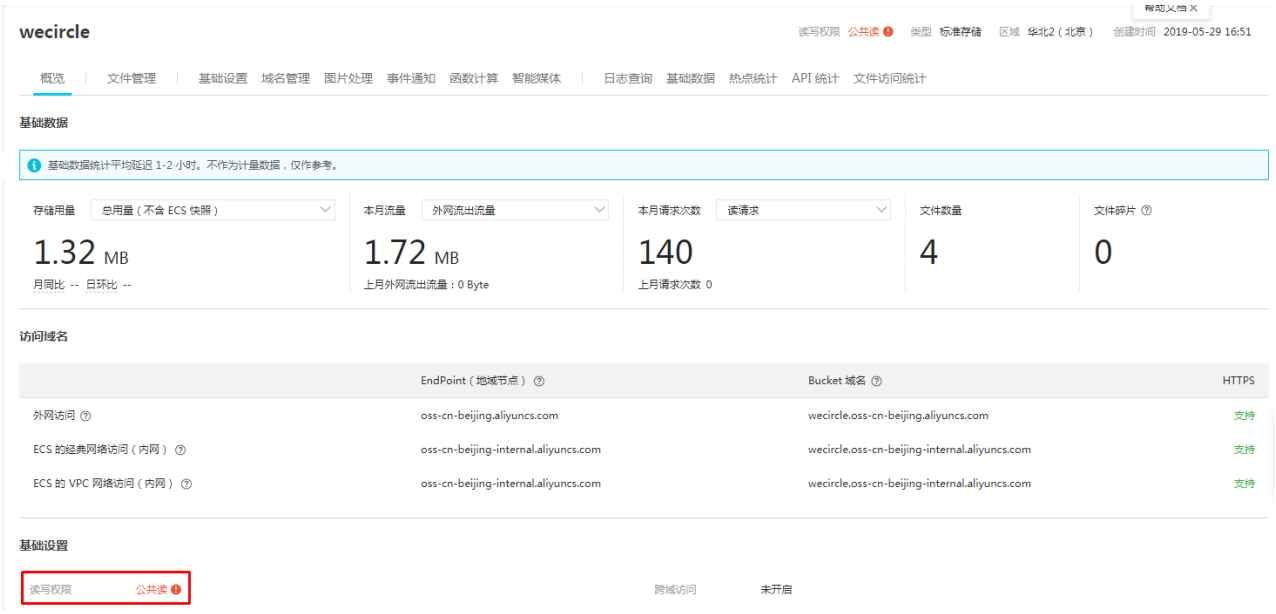
在使用阿里云图片服务之前我们首先要开通阿里云的图片服务。

开通阿里云图片服务：

前提条件在之前短信服务就已经讲解过了这里就不在说了，首先开通OSS服务：

我们在OSS主界面入口新建一个Bucket:

Bucket也叫做存储空间，包含了图片服务，流媒体服务，大数据服务等等，我们用到的是图片服务，进入Bucket主界面：



这里需要注意将权限设置成`公共读`，这样我们就可以在代码里请求图片的url得到图片。

使用**multer**中间件将图片存储在本地

采用multer是Express官方推出的，用于Node.js的`multipart/form-data`请求数据处理的中间件，multer在解析完请求体后，会向Request对象中添加一个body对象和一个file或files对象（上传多个文件时使用files对象 ）,同时可以设置storage来配置存储的路径，对于使用Express的项目来说，可以很方便的完成文件上传存储功能。

安装`multer`:
在后端项目的根目录执行下面命令：

```
npm install multer --save
```

设置存储:

使用第一种方案和第二种方案首先都需要先将上传的图片存储在本地。

在 routes 文件夹下的 post.js里添加上图片存储相关的逻辑，主要是利用 multer 来配置一下存储路径和存储文件名，如下代码所示:

```
var multer  = require('multer');
var storage = multer.diskStorage({
  destination: function (req, file, cb) {
    cb(null, './public/upload/');    // 保存的路径，备注：需要自己创建
  },
  filename: function (req, file, cb) {
    var extname = path.extname(file.originalname);//获取文件扩展名
    // 将保存文件名设置为 字段名 + 时间戳+文件扩展名，比如 logo-1478521468943.jpg
    cb(null, file.fieldname + '-' + Date.now() + extname);
  }
});

var upload = multer({ storage: storage});//multer实例
```

调用阿里云**Node.js SDK**上传图片

第一种方案，采用阿里云的图片服务即图片不是存在我们的服务器上，而是存在远端的阿里云服务上：

安装 ali-oss：

在后端项目的根目录执行下面命令：

```
npm install ali-oss --save
```

调用上传接口：

在 routes 文件夹下的 post.js 中新建一个 uploadimgaliyun 路由，下面这段代码主要创建一个client实例，这个是阿里OSS的上传SDK要求的，供后续使用。

```
const OSS = require('ali-oss');
const client = new OSS({
  region: 'oss-cn-beijing',//bucket所在的区域
  accessKeyId: 'LTAIEGH3Ov5cRwBW',//accessKeyId
  accessKeySecret: 'xxx',//accessKeySecret，请各位使用自己的accessKeySecret
  bucket: 'wecircle'
});
```

- 这里的region可以在Bucket主界面找到。
- accessKeyId和accessKeySecret在之前的短信服务已经获取到，这里直接使用就可以。
- bucket是在新建时填写的bucket名称 。

在 routes 文件夹下的 post.js 中创建了一个post方法的路由，路径是/uploadimgaliyun，当浏览器请求 http://xx.xx.xx/uploadimgaliyun 就会进入这个方法，第二个参数采用 upload.single('image') 接收，表示使用我们上面创建的 multer 的实例。

```
/*
* 阿里云图片上传
*/
router.post('/uploadimgaliyun',upload.single('image'), async function(req, res, next) {
  const OSS = require('ali-oss');
  const client = new OSS({
    region: 'oss-cn-beijing',//bucket所在的区域
    accessKeyId: 'LTAIEGH3Ov5cRwBW',//accessKeyId
    accessKeySecret: '5WmdSUraFE1fDjlPYdLhxYDMwIsGbM',//accessKeySecret，请各位使用自己的accessKeySecret
    bucket: 'wecircle'
  });
  //从请求中得到文件数据
  var file = req.file;
  //得到图片尺寸
  var dimensions = sizeOf(file.path);
  //调用阿里云接口上传
  let result = await client.put(file.filename, file.path);//接收2个参数，文件名和文件路径地址
  console.log(result);
  //返回数据
  res.json({
    code:0,
    data:{
      url: result.url,
      size: dimensions
    }
  });
  //删除临时图片文件
  fs.unlinkSync(file.path)
});
```

这里的流程是：

1. 前端页面通过uploader组件将图片上传，进入这个方法。
2. 利用 multer 将图片存储在本地服务器上，并生成文件名。
3. 调用阿里云SDK的`client.put()`方法将图片上传到阿里云服务上。
4. 返回给前端的url是一个远端的域名的图片url地址。
5. 将本地存储的图片删除，本地不存图片，只作为一个临时图片上传，传完就删除。

通过阿里云图片服务得到的url是一个外网地址，同时支持CDN服务，我们可以通过参数来使用不同的图片尺寸：

图片缩放参数

操作名称： `resize`

- 指定宽高缩放

名称	描述	取值范围
m	指定缩略的模式： ◦ lfit：等比缩放，限制在指定w与h的矩形内的最大图片。 ◦ mfit：等比缩放，延伸出指定w与h的矩形框外的最小图片。 ◦ fill：固定宽高，将延伸出指定w与h的矩形框外的最小图片进行居中裁剪。 ◦ pad：固定宽高，缩略填充。 ◦ fixed：固定宽高，强制缩略。	lfit、mfit、fill、pad、fixed，默认为 lfit。
w	指定目标缩略图的宽度。	1-4096
h	指定目标缩略图的高度。	1-4096
l	指定目标缩略图的最长边。	1-4096
s	指定目标缩略图的最短边。	1-4096
limit	指定当目标缩略图大于原图时是否处理。值是 1 表示不处理；值是 0 表示处理。	0/1，默认是 1
color	当缩放模式选择为 pad（缩略填充）时，可以选择填充的颜色(默认是白色)参数的填写方式：采用 16 进制颜色码表示，如 00FF00（绿色）。	[000000-FFFFFF]

`https://wecircle.oss-cn-beijing.aliyuncs.com/image-1559179895732.png?x-oss-process=image/resize,l_500` 可以得到最长边为500px的缩放的图片。

更多的配置，可以参考这里。

开发本地上传upload接口

第二种方案就是代码存储在本地服务器上，成功之后返回一个本地的url地址，并且不会删除。

在 routes 文件夹下的 post.js 中创建了一个post方法的路由，路径是/uploadimg，当浏览器请求`http://xx.xx.xx/uploadimg`就会进入这个方法，第二个参数采用`upload.single('image')`接收，表示使用我们上面创建的 multer 的实例。

```
/*
* 本地图片上传
*/
router.post('/uploadimg', upload.single('image'),function(req, res, next) {
  //从请求中得到文件数据
  var file = req.file;
  //得到图片尺寸
  var dimensions = sizeOf(file.path);
  console.log('文件类型：%s', file.mimetype);
  console.log('原始文件名：%s', file.originalname);
  console.log('文件大小：%s', file.size);
  console.log('文件保存路径：%s', file.path);

  res.json({
    code:0,
    data:{
      url: config.uploadPath+file.filename,//本地的图片url地址
      size: dimensions
    }
  });
});
```

这里的流程是：

1. 前端页面通过uploader组件将图片上传，进入这个方法。
2. 利用 multer 将图片存储在本地服务器上，并生成文件名和url地址。
3. 这时我们返回给前端的url是一个本地同域名的图片url地址。

小结

本章主要讲解了开发图片上传接口，同时介绍了两种上传的方案，一种将图片存在本地服务器，一种将图片上传到阿里云图片服务。
相关技术点：

1. 什么是第三方图片服务以及阿里云图片服务OSS介绍。
2. Express结合使用multer中间件将图片存储在本地方法。

本章节完整源代码地址：Github

}