@fs/attachment 说明文档

2021/4/12

简介

`@fs/attachment` 提供文件的 `上传/下载/删除` 服务，支持云存储和本地存储。

SVN：http://10.8.30.22/FS-SSMC/trunk/3.00.10/code/web/fs-npm/server/fs-attachment

存储形式

- 云存储：基于七牛云服务实现的文件服务
- 本地存储：本地磁盘

服务形式

- 原生 API 服务：应用侧直接引用 npm 包，调用 API 编程接口
- REST API 服务：使用 `fs-web-server-scaffold` 脚手架，该服务以中间件形式提供

使用

安装：`npm install @fs/attachment`

Native API

Native API 即之前 `fs-attachment` 七牛云存储的 API 使用方式。

```
const Attachment = require('@fs/attachment');
const opts = {
    qiniu: { // 可选
        accessKey: 'your access key',
        secretKey: 'your secret key',
        bucket: 'bucket',
        domain: 'bucket domain'
    },
    local: { // 不指定七牛云配置，默认使用本地存储服务
        origin: 'your local storage server origin', // 必填
        rootPath: 'static',
        childPath: 'upload'
    },
    uploadPath: 'your path',
    maxSize: 10485760 // 10M
};
const attachment = new Attachment(opts);
let rslt = attachment.upload(file);
// let rslt = attachment.upload(file, {uploadPath: 'report'}); // 文件上传路径自定义为“report”，这里的 uploadPath 优先级高于 opts.uploadPath
attachment.download(rslt.key);
attachment.remove(rslt.key);
```

REST API

接口定义

```
let prefix = opts.routePrefix ? `/${opts.routePrefix}` : '';

/**
 * @api {POST} /attachments/:p 上传文件.
 * @apiVersion 1.0.0
 * @apiGroup Attachment
 */
router.post(`${prefix}/attachments/:p`, attachment.upload(mw));

/**
```

```
 * @api {GET} /attachments 下载文件.
 * @apiVersion 1.0.0
 * @apiGroup Attachment
 */
router.get(`${prefix}/attachments`, attachment.download(mw));

/**
 * @api {DELETE} /attachments 删除文件.
 * @apiVersion 1.0.0
 * @apiGroup Attachment
 */
router.del(`${prefix}/attachments`, attachment.deletee(mw));

/**
 * @api {POST} /attachments 批量删除文件.
 * @apiVersion 1.0.0
 * @apiGroup Attachment
 */
router.post(`${prefix}/attachments`, attachment.bulkDelete(mw));
```

项目应用

我司 Web 前后端服务均基于 `fs-web-server-scaffold` 前端脚手架，基于该脚手架，`@fs/attachment` 以中间件形式提供 REST 服务，配置示例如下：

```
/**
 * Created by Julin on 2021/04/02.
 */
'use strict';

const path = require('path');
const os = require('os');

const dev = process.env.NODE_ENV == 'development';

const ANXINCLOUD_FC_CLOUD = process.env.ANXINCLOUD_FC_CLOUD;
const ANXINCLOUD_FC_LOCAL_SVR_ORIGIN =
process.env.ANXINCLOUD_FC_LOCAL_SVR_ORIGIN;
const ANXINCLOUD_FC_LOCAL_ROOT_PATH = process.env.ANXINCLOUD_FC_LOCAL_ROOT_PATH;

const PORT = 4000;

const product = {
    port: PORT,
    staticDirs: [path.join(__dirname, ANXINCLOUD_FC_LOCAL_ROOT_PATH ||
'static')],
    mws: [{
        entry: require('@fs/attachment').entry, // @fs/attachment 中间件配置
        opts: Object.assign({
            routePrefix: '_file-server',
            uploadPath: 'other',
            maxSize: 10485760 // 10M
        }, ANXINCLOUD_FC_CLOUD == 'true' ? {
            qiniu: {
                domain: 'http://p7q1f8t1p.bkt.clouddn.com',
```

```
                bucket: 'anxinyun-test',
                accessKey: "5XrM4wEB9YU6RQwT64sPzzE6cYFKZgssdP5Kj3uu",
                secretKey: "w6j2ixR_i-aelc6I7S3HotKIX-ukMzcKmDfH6-M5"
            }
        } : {
            local: {
                origin: ANXINCLOUD_FC_LOCAL_SVR_ORIGIN || `localhost:${PORT}`,
                rootPath: 'static',
                childPath: 'upload'
            }
        })
    }],
    dc: {
        url: 'postgres://FashionAdmin:123456@10.8.30.39:5432/axy',
        opts: {
            pool: {
                max: 20,
                min: 10,
                idle: 10000
            },
            define: {
                freezeTableName: true, // 固定表名
                timestamps: false // 不含列 "createAt"/"updateAt"/"DeleteAt"
            },
            timezone: '+08:00',
            logging: false
        },
        models: []
    },
    logger: {
        level: 'info',
        json: false,
        filename: path.join(__dirname, 'log', 'runtime.test.log'),
        colorize: false,
        maxsize: 1024 * 1024 * 5,
        rotationFormat: false,
        zippedArchive: true,
        maxFiles: 10,
        prettyPrint: true,
        label: '',
        timestamp: true,
        eol: os.EOL,
        tailable: true,
        depth: null,
        showLevel: true,
        maxRetries: 1
    }
};

const development = {
    port: product.port,
    mws: product.mws,
    dc: product.dc,
    logger: product.logger
};

if (dev) {
    // logger
```

```
    development.logger.filename = path.join(__dirname, 'log',
'development.test.log');
    development.logger.level = 'debug';
    development.dc.opts.logging = console.log;
}

module.exports = dev ? development : product;
```

Web 侧文件上传/下载/删除

安心云 Web 侧 console 和 project 均涉及文件访问服务：

- SVN：http://10.8.30.22/FS-Anxinyun/trunk/codes/web/console
- SVN：http://10.8.30.22/FS-Anxinyun/trunk/codes/web/project

下面以 console 为例，针对`@fs/attachment`，给出两种使用方案：webapi 侧提供文件接口，web 侧提供文件接口。

web-api 文件 REST API

Web API 服务地址：`localhost:4000`

- 文件上传：`POST localhost:4000/attachments/:p`
- 文件下载：`GET localhost:4000/attachments`
- 文件删除：`DELETE localhost:4000/attachments`
- 批量删除文件：`POST localhost:4000/attachments`

web-api 引入 @fs/attachment 中间件

SVN：http://10.8.30.22/FS-Anxinyun/trunk/codes/web-api

- staticDirs：静态资源路径
- mws：`@fs/attachment` 中间件引入

```
// config.js

const ANXINCLOUD_QINIU_DOMAIN_QNDMN_RESOURCE =
process.env.ANXINCLOUD_QINIU_DOMAIN_QNDMN_RESOURCE || flags.qndmn;
const ANXINCLOUD_QINIU_BUCKET_RESOURCE =
process.env.ANXINCLOUD_QINIU_BUCKET_RESOURCE || flags.qnbkt;
const ANXINCLOUD_QINIU_AK = process.env.ANXINCLOUD_QINIU_AK || flags.qnak;
const ANXINCLOUD_QINIU_SK = process.env.ANXINCLOUD_QINIU_SK || flags.qnsk;
const ANXINCLOUD_FC_UPLOAD_PATH = process.env.ANXINCLOUD_FC_UPLOAD_PATH ||
flags.fcUploadPath;
const ANXINCLOUD_FC_LOCAL_SVR_ORIGIN =
process.env.ANXINCLOUD_FC_LOCAL_SVR_ORIGIN || flags.fcSvrOrigin;
const ANXINCLOUD_FC_LOCAL_ROOT_PATH = process.env.ANXINCLOUD_FC_LOCAL_ROOT_PATH
|| flags.fcLocalRootPath;
```

```
const ANXINCLOUD_FC_LOCAL_CHILD_PATH =
process.env.ANXINCLOUD_FC_LOCAL_CHILD_PATH || flags.fcLocalChildPath;

const ANXINCLOUD_FC_CLOUD = !!(ANXINCLOUD_QINIU_AK && ANXINCLOUD_QINIU_SK &&
ANXINCLOUD_QINIU_BUCKET_RESOURCE && ANXINCLOUD_QINIU_DOMAIN_QNDMN_RESOURCE);

const product = {
    port: flags.port || 8080,
    staticDirs: [path.join(__dirname, ANXINCLOUD_FC_LOCAL_ROOT_PATH ||
'static')],
    mws: [{
        entry: require('@fs/attachment').entry,
        opts: Object.assign({
            uploadPath: ANXINCLOUD_FC_UPLOAD_PATH,
            maxSize: 10485760 // 10M
        }, ANXINCLOUD_FC_CLOUD ? {
            qiniu: {
                domain: ANXINCLOUD_QINIU_DOMAIN_QNDMN_RESOURCE,
                bucket: ANXINCLOUD_QINIU_BUCKET_RESOURCE,
                accessKey: ANXINCLOUD_QINIU_AK,
                secretKey: ANXINCLOUD_QINIU_SK
            }
        } : {
            local: {
                origin: ANXINCLOUD_FC_LOCAL_SVR_ORIGIN,
                rootPath: ANXINCLOUD_FC_LOCAL_ROOT_PATH,
                childPath: ANXINCLOUD_FC_LOCAL_CHILD_PATH
            }
        })
    }]
}
```

console 使用 web-api 接口

通常，前端页面可以直接使用 webapi 侧的文件接口。

如果需要**预处理**或**后处理**，可以通过前端 `routes` 发送 webapi 文件接口请求、进行相关处理。

`routes` SVN：http://10.8.30.22/FS-Anxinyun/trunk/codes/web/console/routes/attachment/index.js

```
const request = require('superagent');
const parse = require('async-busboy');
const path = require('path');

const UploadPath = {
    project: [".txt", ".dwg", ".doc", ".docx", ".xls", ".xlsx", ".pdf", ".png",
".jpg"],
    report: [".doc", ".docx", ".xls", ".xlsx", ".pdf"],
    data: [".txt", ".xls", ".xlsx"],
    image: [".png", ".jpg", "svg"],
    three: [".js"],
    video: [".mp4"]
}

module.exports = {
```

```
    entry: function (app, router, opts) {

        const WEBAPI_ORIGIN = opts.apiUrl;

        let download = async function (ctx, next) {
            const { fetchUrl } = opts.qiniu;
            if (ctx.path && ctx.path.includes(fetchUrl)) {
                try {
                    const { filename } = ctx.request.query;
                    const fkey = decodeURI(ctx.path.slice(fetchUrl.length +
1)).replace(/\.json$/, '.js');
                    const token = ctx.query.token;
                    if (!token) throw 'upload error: token is null.';
                    ctx.status = 200;
                    ctx.body = request.get(`${WEBAPI_ORIGIN}/attachments?
token=${token}`).query({ src: fkey, filename });
                } catch (err) {
                    ctx.fs.logger.error(err);
                    ctx.status = 404;
                    ctx.body = { error: 'file not found.' }
                }
            } else {
                await next();
            }
        }

        let upload = async function (ctx, next) {
            try {
                const { files } = await parse(ctx.req);
                const file = files[0];
                const p = ctx.query.type || "image";
                if (!UploadPath[p]) throw '附件存放的文件夹名称无效.';
                const token = ctx.query.token; // token =
(JSON.parse(sessionStorage.getItem('user')) || {}).token;
                if (!token) throw 'token is null.';
                const proxyReq =
request.post(`${WEBAPI_ORIGIN}/attachments/${p}?token=${token}`);
                proxyReq.attach('file', file, file.filename);
                const res = await proxyReq;
                ctx.status = res.status;
                ctx.body = { filename: res.body.uploaded };
            } catch (e) {
                let errMsg = `upload error: ${e}`;
                ctx.status = 500;
                ctx.fs.logger.error(errMsg);
                ctx.body = { err: errMsg };
            }
        }

        let remove = async function (ctx, next) {
            try {
                const fkeys = ctx.request.body;
                const token = ctx.query.token;
                if (!token) throw 'upload error: token is null.';
                const proxyReq = request.post(`${WEBAPI_ORIGIN}/attachments?
token=${token}`).send(fkeys);
                const res = await proxyReq;
                ctx.status = res.status;
```

```
                ctx.body = res.body.keys;
            } catch (err) {
                ctx.status = 500;
                ctx.fs.logger.error(err);
                ctx.body = { err: 'upload cleanup error.' };
            }
        }

        router.use(download);
        router.post('/_upload/new', upload);
        router.post('/_upload/cleanup', remove);
    }
};
```

console 文件 REST API

Web 服务地址：`localhost:5000`

- 文件上传：`POST localhost:5000/attachments/:p`
- 文件下载：`GET localhost:5000/attachments`
- 文件删除：`DELETE localhost:5000/attachments`
- 批量删除文件：`POST localhost:5000/attachments`

console 引入 @fs/attachment 中间件

同前面 "web-api 引入 @fs/attachment 中间件" 。

routePrefix 解决方案

利用 `routePrefix` 解决 api 与页面路由定义**可能**的冲突、路由可读性。

- 页面路由：
  - `文档管理` Page Router：`localhost:5000/pan`
  - `用户日志` Page Router：`localhost:5000/log`
- 文件下载 REST API：
  - 不使用 `routePrefix`：`localhost:5000/attachments`
  - 使用 `_file-server` 路由前缀：`localhost:5000/_file-server/attachments`

```
const product = {
    mws: [{
        entry: require('@fs/attachment').entry,
        opts: Object.assign({
            routePrefix: '_file-server', // 利用 `routePrefix` 解决 api 与页面路由定义可能的冲突、路由可读性
            uploadPath: 'other',
            maxSize: 10485760 // 10M
        }, ANXINCLOUD_FC_CLOUD == 'true' ? {
            qiniu: {
                domain: 'http://p7q1f8t1p.bkt.clouddn.com',
```

```
                bucket: 'anxinyun-test',
                accessKey: "5XrM4wEB9YU6RQwT64sPzzE6cYFKZgssdP5Kj3uu",
                secretKey: "w6j2ixR_i-aelc6I7S3HotKIX-ukMzcKmDfH6-M5"
            }
        } : {
            local: {
                origin: ANXINCLOUD_FC_LOCAL_SVR_ORIGIN || `localhost:${PORT}`,
                rootPath: 'static',
                childPath: 'upload'
            }
        })
    }],
}
```

前端拆包待处理

console 配置优化

@fs/attachment 支持云存储和本地存储，因此，安心云 console 配置字段名 `qiniu` 待优化。

SVN：http://10.8.30.22/FS-Anxinyun/trunk/codes/web/console/config.js

```
{
    entry: require('./routes').entry,
    opts: {
        qiniu: { // @TODO: qiniu -> fc
            fetchUrl: '/_file-server',
            domain: ANXINCLOUD_QINIU_DOMAIN_QNDMN_RESOURCE // @TODO: 域名使用
ANXINCLOUD_QINIU_DOMAIN_QNDMN_RESOURCE 不恰当
        },
    }
}
```

前端路由优化

SVN：http://10.8.30.22/FS-Anxinyun/trunk/codes/web/console/routes/attachment/index.js

```
const getResourceRoot = async function (ctx) {
    const { domain } = opts.qiniu; // @TODO: opts.qiniu -> opts.fc

    ctx.status = 200;
    ctx.body = { root: domain };
}

router.get('/_resource/root', getResourceRoot);
```

resourceRoot 优化

问题：resourceRoot 引入大量代码

console sections 中大量 `resourceRoot` 属性值的使用：

SVN：http://10.8.30.22/FS-Anxinyun/trunk/codes/web/console/client/src/sections

```
// client\src\sections\project-monitor\containers\cdContainer\cedian.js
<img style={{ width: 200, cursor: 'pointer' }} src=
{`${this.props.resourceRoot}/${text}`} />
```

SVN：http://10.8.30.22/FS-Anxinyun/trunk/codes/web/console/client/src/layout/actions/global.js

```
export const INIT_RESOURCE_ROOT = 'INIT_RESOURCE_ROOT';
export function initResourceRoot() {
    let resourceRoot = localStorage.getItem('resource-root');
    if (resourceRoot) {
        return {
            type: INIT_RESOURCE_ROOT,
            payload: {
                resourceRoot
            }
        }
    } else {
        return dispatch => {
            RouteRequest.get('/_resource/root').then(res => {
                localStorage.setItem('resource-root', res.root);

                dispatch({
                    type: INIT_RESOURCE_ROOT,
                    payload: {
                        resourceRoot: res.root
                    }
                })
            });
        }
    }
}
```

解决：资源持久化

- 文件上传 REST API

  HTTP Request：

  ```
  POST /attachments/:p
  ```

  Response 200：

  ```
  {
      uploaded: ctx.query.uploadedBy == 'url' ? furl : fkey, // 默认返回 key 值
      key: fkey,
      url: furl
  };
  ```

  其中，`uploaded` 属性：根据 `uploadedBy` 查询参数给出资源的 `key` 或 `url` 结果。

- 持久化数据
  - 持久化 `key` 值：`upload/data/${filename}`
  - 持久化 `url` 值：`https://files.anxinyun.cn/upload/data/${filename}`
- PG Table `t_netdisk_file` 示例

id	file_link	...
1	upload/data/吴淞桥*日报表*2018年11月12日_d2af9cf7-ea87-4d5f-a74a-1ebbaf37e307.xls	
2	https://files.anxinyun.cn/upload/data/吴淞桥*日报表*2018年11月12日_d2af9cf7-ea87-4d5f-a74a-1ebbaf37e307.xls	

关于 token

- webapi 文件接口服务：`token` 是 webapi 鉴权的令牌，若前端调用 webapi 接口，则需要提供 `token` 查询参数 (cookie 存储 `token`)。
- web 文件接口服务：Web侧以 `@fs/attachment` 中间件提供 REST API 服务，可以不限定 `token` 查询参数 (不安全，不推荐)。