link: null
title: 珠峰架构师成长计划
description: gulpfile.js
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=90 sentences=306, words=1833

1.实现gulp

1.1 安装依赖

```
npm install fs-extra undertaker vinyl-fs --save
```

1.2 gulpfile.js

gulpfile.js

```
const { series, parallel } = require('gulp');
const defaultTask = (done) => {
    console.log('defaultTask');
    done();
}

const oneTask = (done) => {
    setTimeout(() => {
        console.log('oneTask');
        done();
    }, 1000);
}
const twoTask = (done) => {
    setTimeout(() => {
        console.log('twoTask');
        done();
    }, 1000);
}
const threeTask = (done) => {
    setTimeout(() => {
        console.log('threeTask');
        done();
    }, 1000);
}

const seriesTask = series(oneTask, twoTask, threeTask);
const parallelTask = parallel(oneTask, twoTask, threeTask);
exports.default = defaultTask;
exports.series = seriesTask;
exports.parallel = parallelTask;
```

1.3 package.json

```
{
  "main": "lib/index.js",
  "bin": {
    "gulp4": "./bin/gulp4.js"
  }
}
```

1.4 bin\gulp4.js

bin\gulp4.js

```

const path = require('path');
const gulpInst = require('../lib');
const logEvents = require('./logEvents');
const registerExports = require('./register-exports');
logEvents(gulpInst);
const taskName = process.argv[2];
const toRun = taskName || 'default';
const configPath = path.join(process.cwd(), 'gulpfile.js');
console.log(`Using gulpfile ${configPath}`);
const exported = require(configPath);
registerExports(gulpInst, exported);
gulpInst.parallel(toRun)(() => { console.log('done') });
```

1.5 bin\logEvents.js

bin\logEvents.js

```
function logEvents(gulpInst) {
    gulpInst.on('start', function (evt) {
        console.log(`Starting ${evt.name} ...`);
    });
    gulpInst.on('stop', function (evt) {
        console.log(`Finished ${evt.name} after ${evt.duration[0]} ms`);
    });
}
module.exports = logEvents;
```

1.6 bin\register-exports.js

bin\register-exports.js

```
function registerExports(gulpInst, tasks) {
    let taskNames = Object.keys(tasks);
    taskNames.forEach(taskName => {
        gulpInst.task(taskName, tasks[taskName]);
    });
}
module.exports = registerExports;
```

1.7 lib\index.js

lib\index.js

```
const util = require('util');
const Undertaker = require('undertaker');
function Gulp() {
    Undertaker.call(this);
    this.task = this.task.bind(this);
    this.series = this.series.bind(this);
    this.parallel = this.parallel.bind(this);
}
util.inherits(Gulp, Undertaker);
const inst = new Gulp();
module.exports = inst;
```

2.实现undertaker

2.1 lib\index.js

lib\index.js

```
const util = require('util');
+const Undertaker = require('./undertaker');
function Gulp() {
    Undertaker.call(this);
    this.task = this.task.bind(this);
    this.series = this.series.bind(this);
    this.parallel = this.parallel.bind(this);
}
util.inherits(Gulp, Undertaker);
const inst = new Gulp();
module.exports = inst;
```

2.2 undertaker.js

lib\undertaker.js

```
let { inherits } = require('util');
let EventEmitter = require('events')
function Undertaker() {
    EventEmitter.call(this);
    this._tasks = {};
}
inherits(Undertaker, EventEmitter);
function task(name, fn) {
    this._tasks[name] = fn;
}
function series() {
    let args = Array.from(arguments);
    let fn = buildSeries(args);
    return fn.bind(this);
}
function parallel() {
    let args = Array.from(arguments);
    let fn = buildParallel(args);
    return fn.bind(this);;
}
function run(taskName, done) {
    let fn = this._tasks[taskName];
    fn(done);
}
Undertaker.prototype.task = task;
Undertaker.prototype.series = series;
Undertaker.prototype.parallel = parallel;
Undertaker.prototype.run = run;
function buildSeries(values) {
    function series(done) {
        let length = values.length;
        let idx = 0;
        let results = [];
        const next = (idx) => {
            let value = values[idx];
            if (typeof value !== 'function') {
                value = this._tasks[value];
            }
            let startHr = process.hrtime();
            this.emit('start', { name: value.name });
            value((err, result) => {
                this.emit('stop', { name: value.name, duration: process.hrtime(startHr) });
                results[idx] = result;
                if (++idx >= length) {
                    done(err, results);
                } else {
                    next(idx);
                }
            });
        }
        next(idx);
    }
    return series;
}
function buildParallel(values) {
    function parallel(done) {
        let length = values.length;
        let count = length;
        let results = [];
        const next = (idx) => {
            let value = values[idx];
            if (typeof value !== 'function') {
                value = this._tasks[value];
            }
            let startHr = process.hrtime();
            this.emit('start', { name: value.name });
            value((err, result) => {
                this.emit('stop', { name: value.name, duration: process.hrtime(startHr) });
                results[idx] = result;
                if (--count === 0) {
                    done(err, results);
                }
            });
        }
        for (idx = 0; idx < length; idx++) {
            next(idx);
        }
    }
    return parallel;
}
module.exports = Undertaker;
```

2.使用流操作

2.1 gulpfile.js

```
const { src, dest } = require('gulp');
const defaultTask = () => {
    return src('src/scripts/**/*.js').pipe(dest('dist'));
}

exports.default = defaultTask;
```

```
gulp
```

2.实现流操作

2.1 lib\index.js

```
const util = require('util');
const Undertaker = require('undertaker');
+const vfs = require('vinyl-fs');
function Gulp() {
    Undertaker.call(this);
    this.task = this.task.bind(this);
    this.series = this.series.bind(this);
    this.parallel = this.parallel.bind(this);

+   this.src = this.src.bind(this);
+   this.dest = this.dest.bind(this);
}
util.inherits(Gulp, Undertaker);
+Gulp.prototype.src = vfs.src;
+Gulp.prototype.dest = vfs.dest;
const inst = new Gulp();
module.exports = inst;
```

3.vinyl-fs基础

3.1 glob

- gulp内部使用了node-glob模块来实现其文件匹配功能

匹配符 说明 星 匹配文件路径中的0个或多个字符，但不会匹配路径分隔符 星星 匹配路径中的0个或多个目录及其子目录 ? 匹配文件路径中的一个字符(不会匹配路径分隔符) [...] 匹配方括号中出现的字符中的任意一个，当方括号中第一个字符为^或!时，则表示不匹配方括号中出现的其他字符中的任意一个 !(pattern1/pattern2/pattern3) 匹配任何与括号中给定的任一模式都不匹配的 ?(pattern1/pattern2/pattern3) 匹配括号中给定的任一模式0次或1次 +(pattern1/patter2n/pattern3) 匹配括号中给定的任一模式至少1次 (pattern1/pattern2/pattern3) 匹配括号中给定的任一模式0次或多次 @(pattern1/pattern2/pattern3) 匹配括号中给定的任一模式1次

glob-stream.js

```
const { Readable } = require('stream');
let { inherits } = require('util');
let { Glob } = require('glob');
var globParent = require('glob-parent');
var toAbsoluteGlob = require('to-absolute-glob');
function GlobStream(glob, opt = {}) {
    opt.cwd = opt.cwd || process.cwd();
    Readable.call(this, { objectMode: true });
    let absoluteGlob = toAbsoluteGlob(glob, opt);
    let basePath = globParent(absoluteGlob);
    let globber = new Glob(absoluteGlob, opt);
    this._globber = globber;
    globber.on('match', (filepath) => {
        let obj = {
            cwd: opt.cwd,
            base: basePath,
            path: filepath
        };
        this.push(obj);
    });
    globber.once('end', () => {
        this.push(null);
    });
}
inherits(GlobStream, Readable);

GlobStream.prototype._read = function () {
    this._globber.resume();
};
module.exports = GlobStream;
```

```
let GlobStream = require('./glob-stream');
const glob = 'src/scripts/**/*.js';
let gs = new GlobStream(glob);
gs.on('data', (data) => {
    console.log(data);
});
```

3.2 vinyl

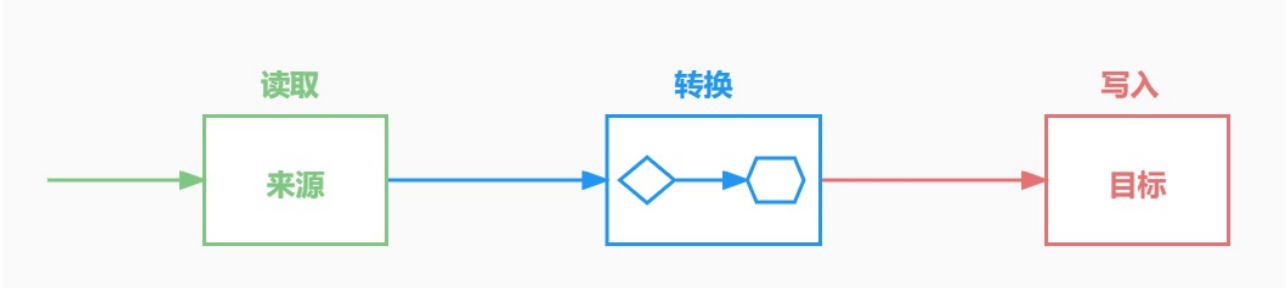
- gulp.src中这个流里的内容不是原始的文件流,而是一个虚拟文件对象流,这个虚拟文件对象中存储着原始文件的路径、文件名和内容等信息 vinyl

```
var File = require('vinyl');
var indexFile = new File({
    cwd: "/",
    base: "/test/",
    path: "/test/index.js",
    contents: new Buffer("zhufeng")
});

console.log(File.isVinyl(indexFile));
console.log(indexFile.isBuffer());
console.log(indexFile.isStream());
```

3.3 wrap-vinyl.js

- through2内部仅是封装了Transform的构造函数



wrap-vinyl.js

```
var File = require('vinyl');
var through = require('through2');
function wrapVinyl() {
    function wrapFile(globFile, encoding, callback) {
        var file = new File(globFile);
        callback(null, file);
    }
    return through.obj(wrapFile);
}
module.exports = wrapVinyl;
```

```
let GlobStream = require('./glob-stream');
let wrapVinyl = require('./wrap-vinyl');
const glob = 'src/scripts/**/*.js';
let gs = new GlobStream(glob);
gs.pipe(wrapVinyl()).on('data', (data) => {
    console.log(data);
});
```

4.实现vinyl-fs

4.1 vinyl-fs\index.js

lib\vinyl-fs\index.js

```
const src = require('./lib/src');
const dest = require('./lib/dest');
module.exports = {
    src,
    dest
};
```

4.2 src\index.js

lib\vinyl-fs\lib\src\index.js

```
var gs = require('./glob-stream');
var readContents = require('./read-contents');
var wrapVinyl = require('./wrap-vinyl');
function src(glob) {
    let gsStream = gs(glob);
    let vinylStream = gsStream.pipe(wrapVinyl());
    let contentsStream = vinylStream.pipe(readContents());
    return contentsStream;
}
module.exports = src;
```

4.3 glob-stream.js

lib\vinyl-fs\lib\src\glob-stream.js

```
let { Readable } = require('readable-stream');
let { inherits } = require('util');
let { Glob } = require('glob');
var globParent = require('glob-parent');
var toAbsoluteGlob = require('to-absolute-glob');
inherits(GlobStream, Readable);
GlobStream.prototype._read = function () {
    this._globber.resume();
};
function globStream(glob, opt = {}) {
    opt.cwd = opt.cwd || process.cwd();
    return new GlobStream(glob, opt);
}
function GlobStream(glob, opt) {
    Readable.call(this, { objectMode: true });
    let absoluteGlob = toAbsoluteGlob(glob, opt);
    let basePath = globParent(absoluteGlob);
    let globber = new Glob(absoluteGlob, opt);
    this._globber = globber;
    globber.on('match', (filepath) => {
        let obj = {
            cwd: opt.cwd,
            base: basePath,
            path: filepath
        };
        this.push(obj)
    });
    globber.once('end', () => {
        this.push(null);
    });
}
module.exports = globStream;
```

4.4 read-contents.js

lib\vinyl-fs\lib\src\read-contents.js

```
let fs = require('fs');
let through = require('through2');
function readContents() {
    function readFile(file, encoding, callback) {
        fs.readFile(file.path, encoding, (err, data) => {
            file.contents = Buffer.from(data);
            callback(null, file);
        });
    }
    return through.obj(readFile);
}
module.exports = readContents;
```

4.5 wrap-vinyl.js

lib\vinyl-fs\lib\src\wrap-vinyl.js

```
var File = require('vinyl');
var through = require('through2');
function wrapVinyl() {
    function wrapFile(globFile, encoding, callback) {
        var file = new File(globFile);
        callback(null, file);
    }
    return through.obj(wrapFile);
}
module.exports = wrapVinyl;
```

4.6 dest\index.js

lib\vinyl-fs\lib\dest\index.js

```
const writeContents = require('./write-contents');
function dest(outFolder) {
    return writeContents(outFolder);
}
module.exports = dest;
```

4.7 write-contents.js

lib\vinyl-fs\lib\dest\write-contents.js

```
const fs = require('fs-extra');
const path = require('path');
var through = require('through2');
function writeContents(outFolder) {
    function writeFile(file, encoding, callback) {
        var basePath = path.resolve(file.cwd, outFolder);
        var writePath = path.resolve(basePath, file.relative);
        file.path = writePath;
        fs.ensureDir(path.dirname(writePath), (err) => {
            fs.writeFile(file.path, file.contents, encoding, callback);
        });
    }
    return through.obj(writeFile);
}
module.exports = writeContents;
```

5.实现插件

5.1 gulpfile.js

gulpfile.js

```
const { src, dest } = require('gulp4');
var gulpPrefixer = require('./gulp-prefixer');
var gulpBabel = require('./gulp-babel');
const defaultTask = () => {
    return src('src/scripts/**/*.js')
        .pipe(gulpPrefixer('/**prepended**/\n'))
        .pipe(gulpBabel({ presets: ["@babel/preset-env"] }))
        .pipe(dest('dist'));
}
exports.default = defaultTask;
```

5.2 gulp-prefixer.js

gulp-prefixer.js

```
var through = require('through2');
function gulpPrefixer(prefixText) {
    prefixText = Buffer.from(prefixText);
    var stream = through.obj(function (file, enc, next) {
        if (file.isBuffer()) {
            file.contents = Buffer.concat([prefixText, file.contents]);
        }
        this.push(file);
        next();
    });
    return stream;
};
module.exports = gulpPrefixer;
```

5.3 gulp-babel.js

gulp-babel.js

```
var through = require('through2');
const babel = require('@babel/core');
function gulpBabel(options) {
    var stream = through.obj(function (file, enc, next) {
        const { code } = babel.transformSync(file.contents, options);
        file.contents = Buffer.from(code);
        this.push(file);
        next();
    });
    return stream;
};
module.exports = gulpBabel;
```