本文主要介绍mochawesome的使用，mocha+mochawesome用于生成完整的单元测试报告

1-mochawesome简介

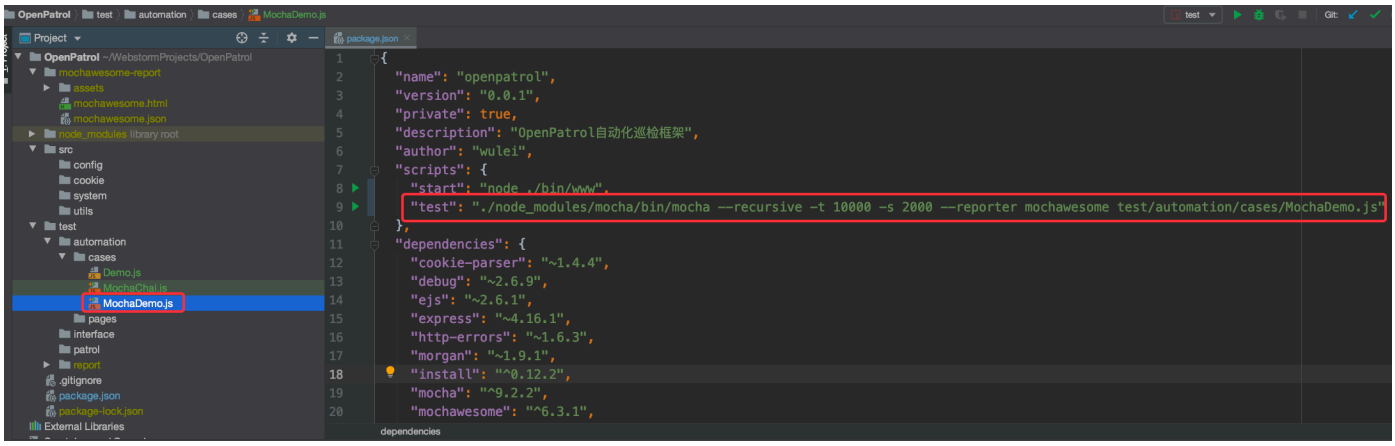
mochawesome是一个生成自定义Javascript测试的报告库，通常都是配合mocha使用。也可以和 mochawesome-report-generator 配合生成增强的 HTML/CSS报告。

2-基本使用流程

1-package.json

package.json scripts部分新增test设置

```
"test": "./node_modules/mocha/bin/mocha --recursive -t 10000 -s 2000 --reporter mochawesome test/automation/cases/MochaDemo.js"
```



参数解释：

--recursive：递归子目录
默认情况下，mocha查找glob ./test/.js，因此您可能希望将测试放在test/文件夹中。如果要包含子目录，请使用--recursive，因为./test/*.js只匹配第一级中的文件，test并且./test/**/.js只匹配第二级中的文件test。

-t, --timeout
指定测试用例超时，默认为2秒。要覆盖，您可以以毫秒为单位传递超时，或者使用s后缀ex：--timeout 2s或--timeout 2000等效的值传递。

-s, --slow
指定“slow test”测试阈值，默认为75毫秒。Mocha使用它来突出显示花费太长时间的测试用例。

文件目录

支持单文件/通配符：test/automation/Mocha*.js 代表Mocha开头的文件

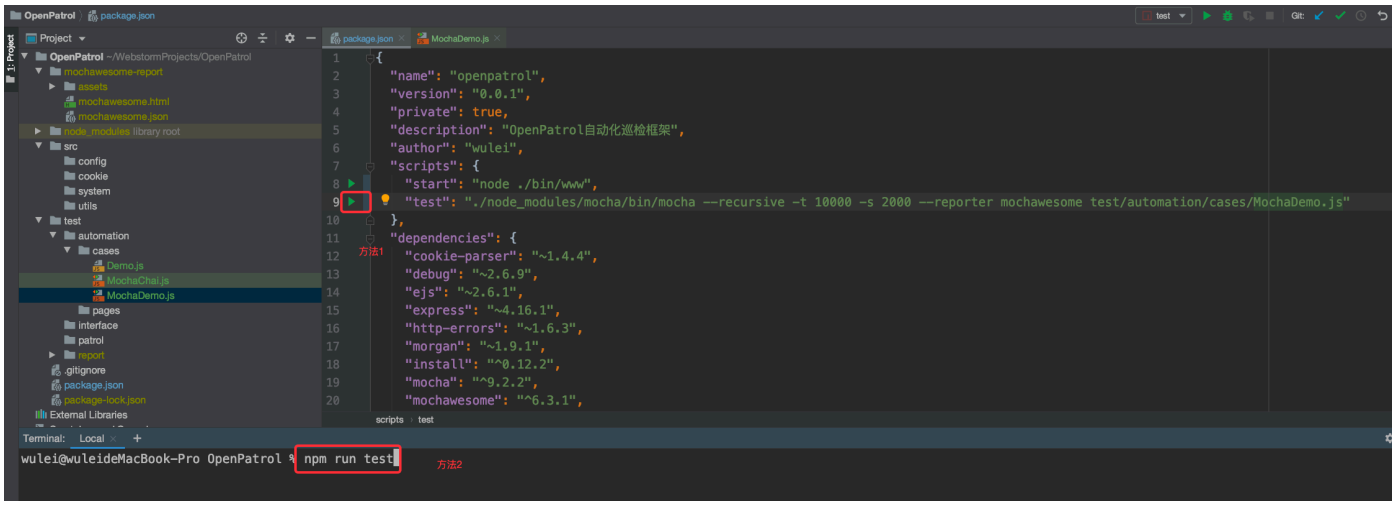
文件目录：test/automation/ 目录下所有的文件

2-运行

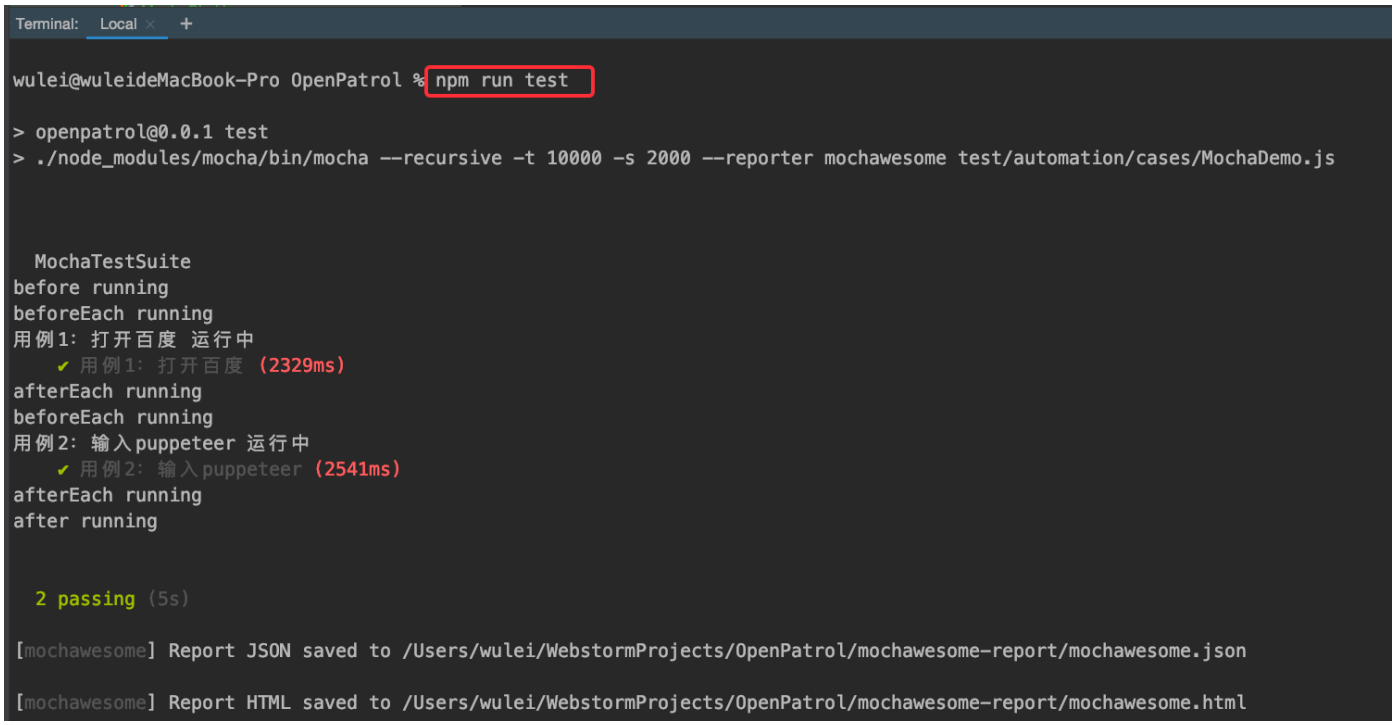
运行测试有两种方式：

1-npm run test :命令行运行

2-点击运行按钮

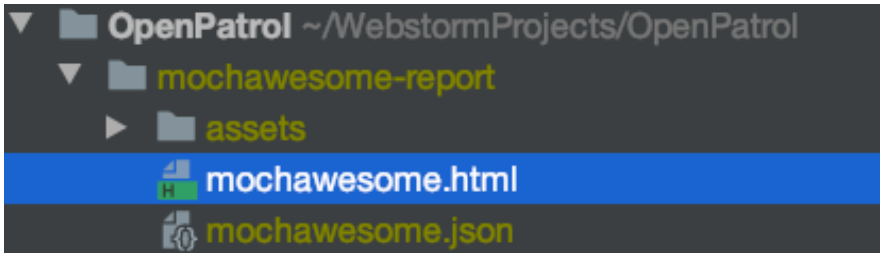


通过：npm run test 运行后

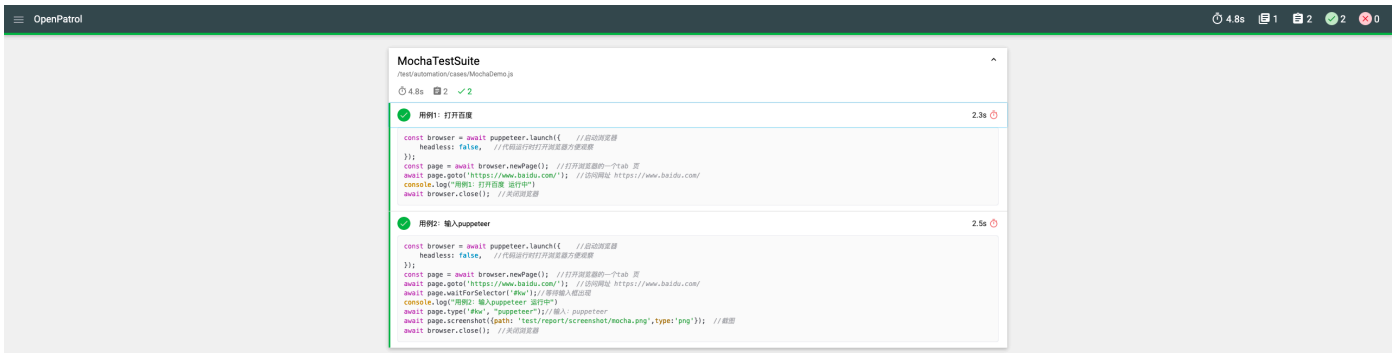


3-查看报告

默认报告路径：mochawesome-report/



报告内容：



3-增强测试报告

默认的报告内容展示的信息比较简单，只有代码部分的展示，但是如果出错时，我们希望看到请求/响应的信息。mochawesome提供了addContext方法可以给测试报告添加额外的自定义内容。

1-addContext()源码

源码：mochawesome/src/addContext.js

const addContext = require('mochawesome/addContext');
addContext有两个参数，第一个参数传递mocha的this实例，第二个参数是自定义信息。

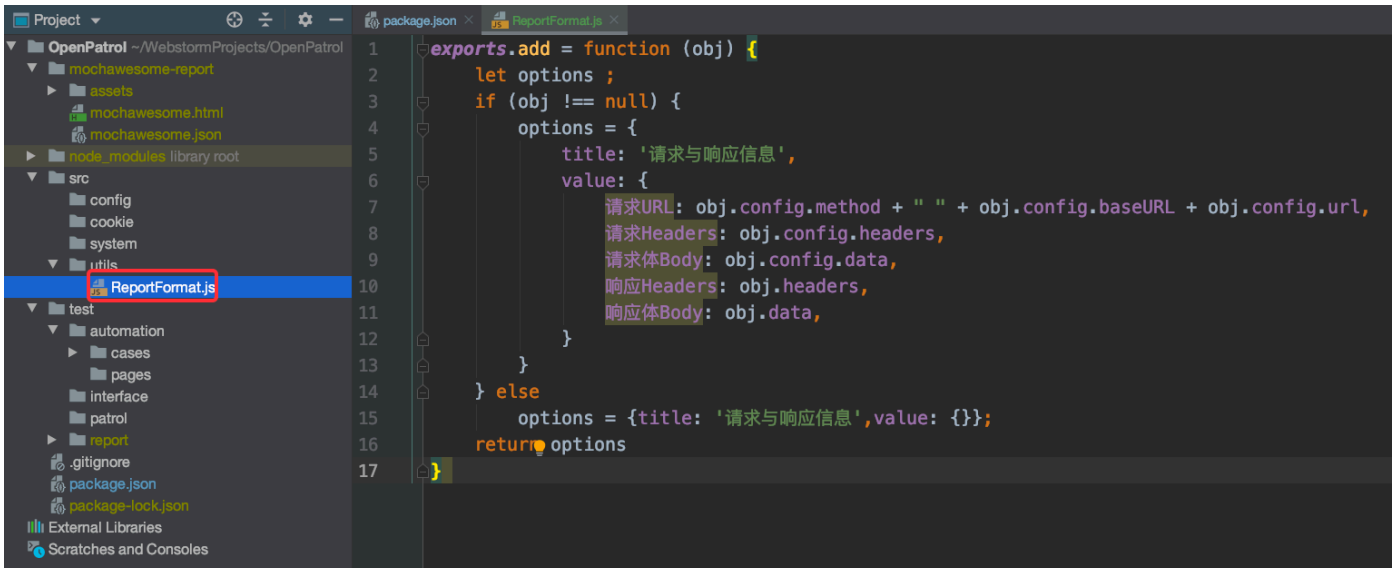
```
 *   addContext(this, {
 *     title: 'Expected number of something'
 *     value: 42
 *   });
 *
 *   assert('something');
 * });
 *
 */

const addContext = function (...args) {
  // Check args to see if we should bother continuing
  if (args.length !== 2 || !isObject(args[0])) {
    log(ERRORS.INVALID_ARGS, level: 'error');
    return;
  }

  const ctx = args[1];

  // Ensure that context meets the requirements
  if (!_isValidContext(ctx)) {
    log(ERRORS.INVALID_CONTEXT(ctx), level: 'error');
    return;
  }
```

源码里面的使用介绍

2-自定义方法

新建：src/utils/ReportFormat.js ，代码如下

```
exports.add = function (obj) {
    let options ;
    if (obj !== null) {
        options = {
            title: '请求与响应信息',
            value: {
                请求URL: obj.config.method + " " + obj.config.baseURL + obj.config.url,
                请求Headers: obj.config.headers,
                请求体Body: obj.config.data,
                响应Headers: obj.headers,
                响应体Body: obj.data,
            }
        }
    } else
        options = {title: '请求与响应信息',value: {}};
    return options
}
```

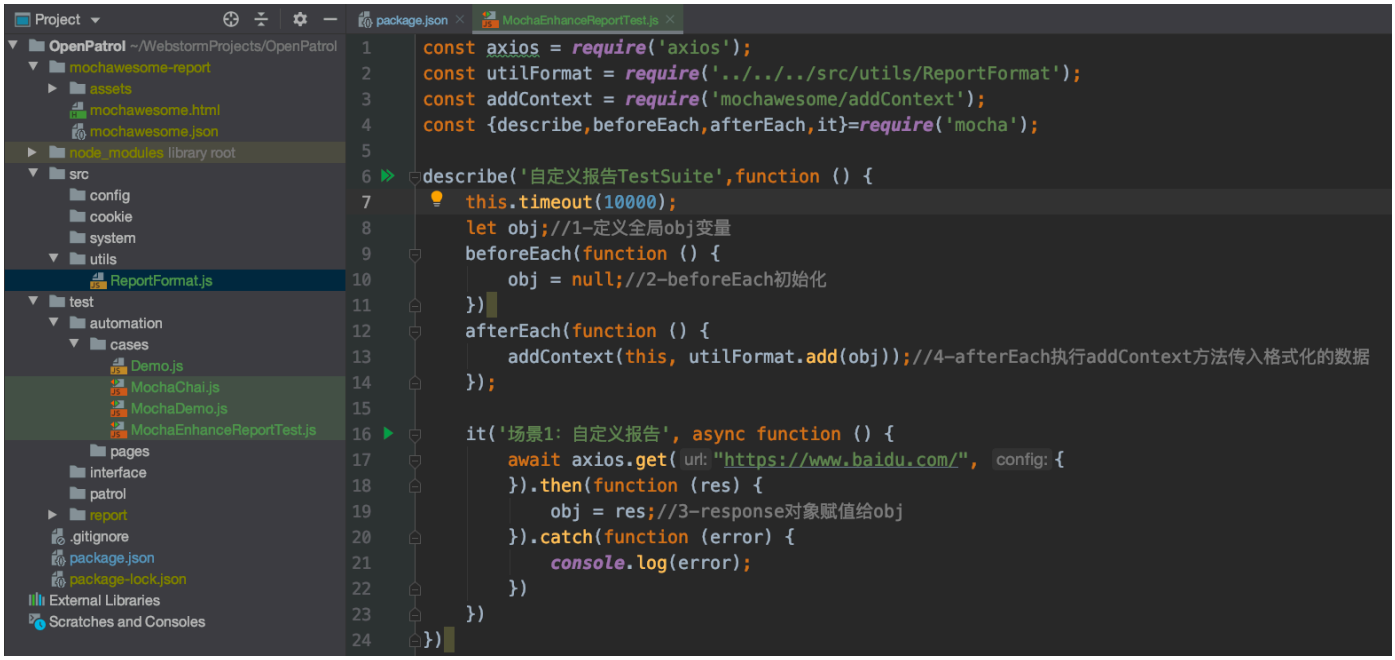


3-新建用例

新建：test/automation/cases/MochaEnhanceReportTest.js

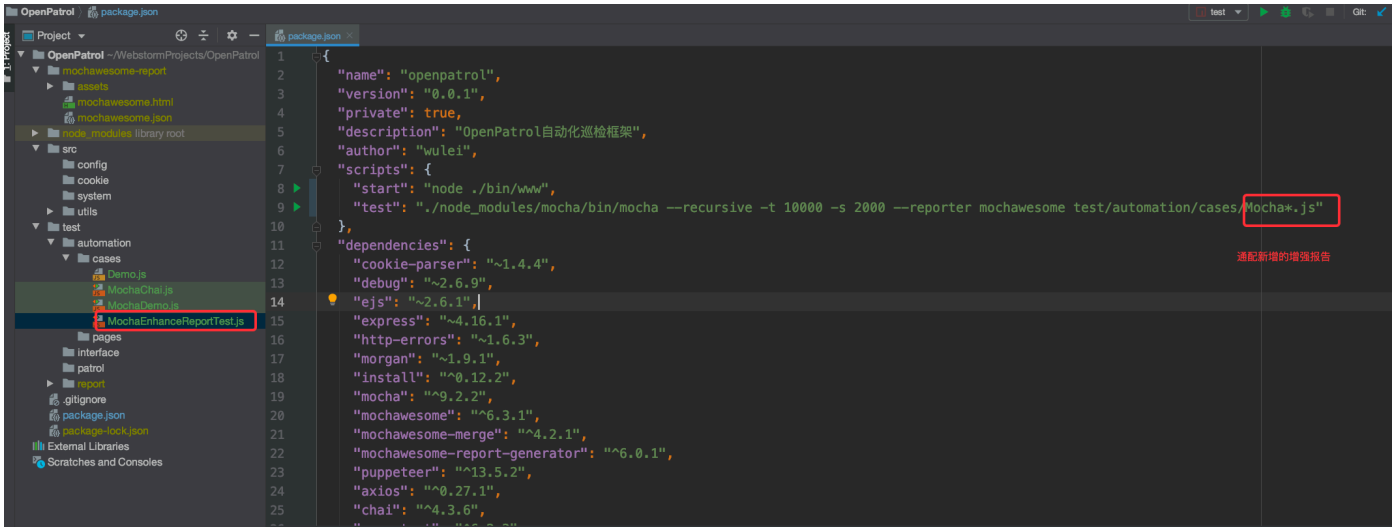
```
const axios = require('axios');
const utilFormat = require('../../../src/utils/ReportFormat');
const addContext = require('mochawesome/addContext');
const {describe,beforeEach,afterEach,it}=require('mocha');

describe('自定义报告TestSuite',function () {
    this.timeout(10000);
```

```
    let obj;//1-定义全局obj变量
    beforeEach(function () {
        obj = null;//2-beforeEach初始化
    })
    afterEach(function () {
        addContext(this, utilFormat.add(obj));//4-afterEach执行addContext方法传入格式化的数
据
    });

    it('场景1：自定义报告', async function () {
        await axios.get("https://www.baidu.com/", {
        }).then(function (res) {
            obj = res;//3-response对象赋值给obj
        }).catch(function (error) {
            console.log(error);
        })
    })
})
```

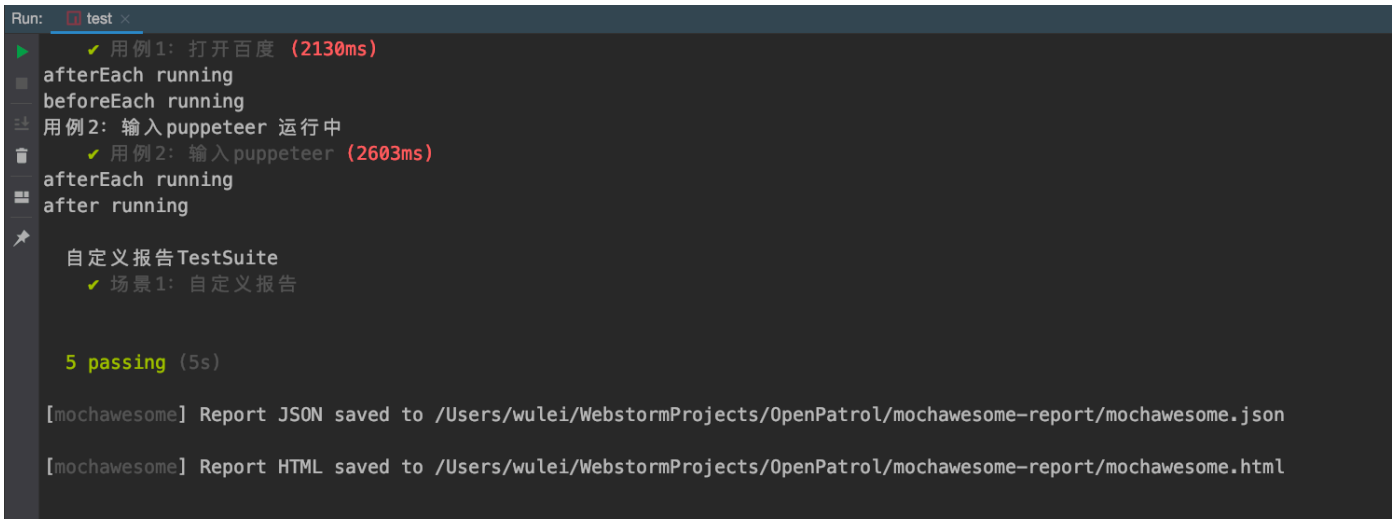


4-运行脚本

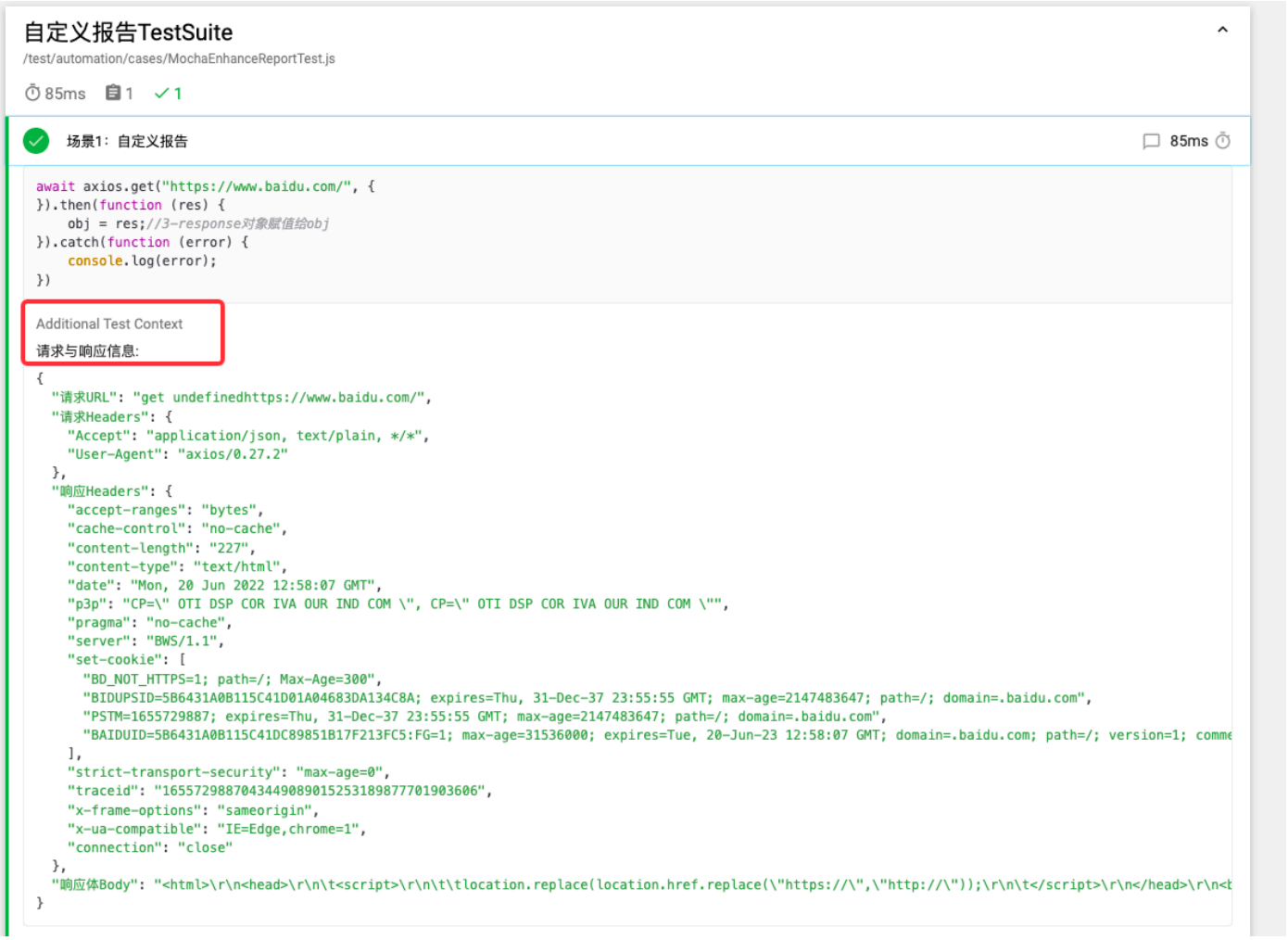
修改package.json，scripts部分，匹配到新增的用例


OpenPatrol
package.json
Project
OpenPatrol ~/WebstormProjects/OpenPatrol
mochawesome-report
assets
mochawesome.html
mochawesome.json
node_modules library root
src
config
cookie
system
utils
test
automation
cases
Demo.js
MochaChai.js
MochaDemo.js
MochaEnhanceReportTest.js
pages
interface
patrol
report
.gitignore
package.json
package-lock.json
External Libraries
Scratches and Consoles
{
"name": "openpatrol",
"version": "0.0.1",
"private": true,
"description": "OpenPatrol自动化巡检框架",
"author": "wulei",
"scripts": {
"start": "node ./bin/www",
"test": "./node_modules/mocha/bin/mocha --recursive -t 10000 -s 2000 --reporter mochawesome test/automation/cases/Mocha*.js"
},
"dependencies": {
"cookie-parser": "~1.4.4",
"debug": "~2.6.9",
"ejs": "~2.6.1",
"express": "~4.16.1",
"http-errors": "~1.6.3",
"morgan": "~1.9.1",
"install": "^0.12.2",
"mocha": "^9.2.2",
"mochawesome": "^6.3.1",
"mochawesome-merge": "^4.2.1",
"mochawesome-report-generator": "^6.0.1",
"puppeteer": "^13.5.2",
"axios": "^0.27.1",
"chai": "^4.3.6",
通配新增的增强报告

5-运行后查看


Run:
test
✔ 用例1: 打开百度 (2130ms)
afterEach running
beforeEach running
用例2: 输入puppeteer 运行中
✔ 用例2: 输入puppeteer (2603ms)
afterEach running
after running
自定义报告TestSuite
✔ 场景1: 自定义报告
5 passing (5s)
[mochawesome] Report JSON saved to /Users/wulei/WebstormProjects/OpenPatrol/mochawesome-report/mochawesome.json
[mochawesome] Report HTML saved to /Users/wulei/WebstormProjects/OpenPatrol/mochawesome-report/mochawesome.html

报告：



4-自定义报告信息

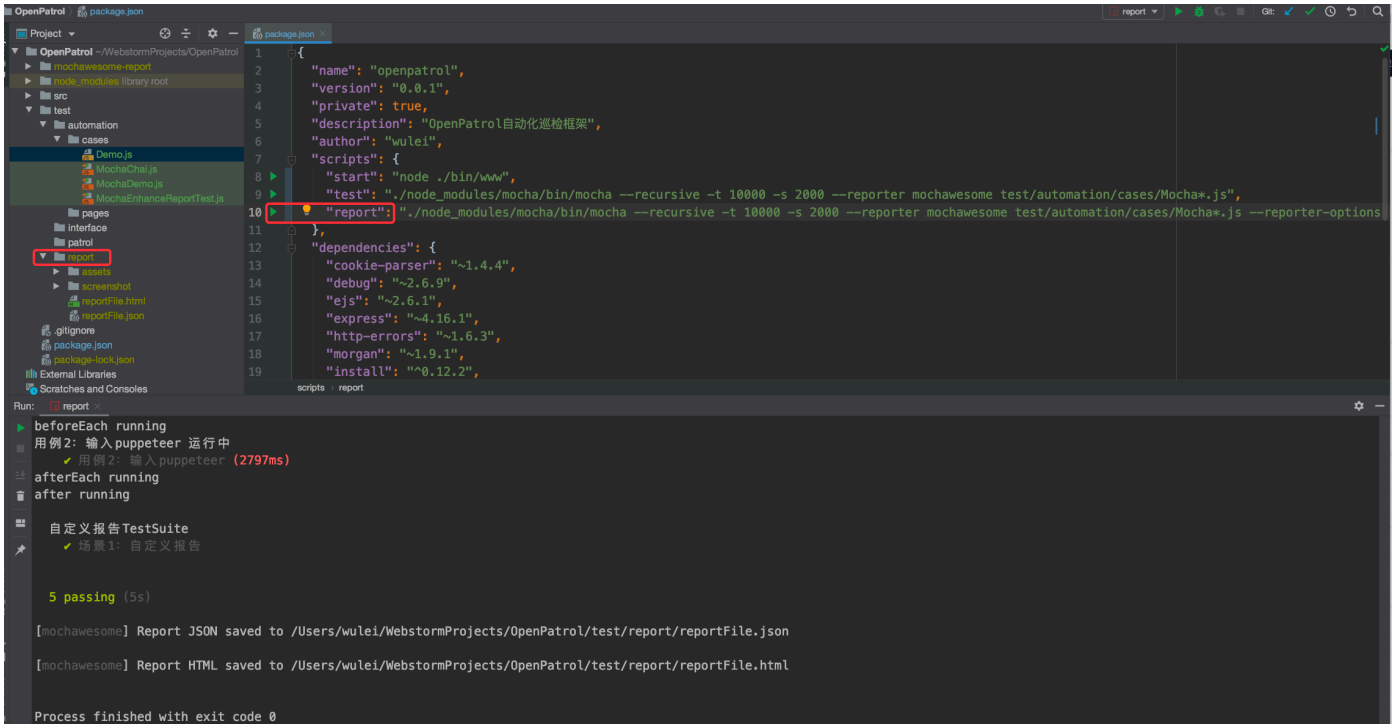
mochawesome可以通过addContext进行附加信息的输出外，还可以通过[mochawesome-report-generator](mochawesome-report-generator) 进行更多的自定义报告。

1-package.json新增

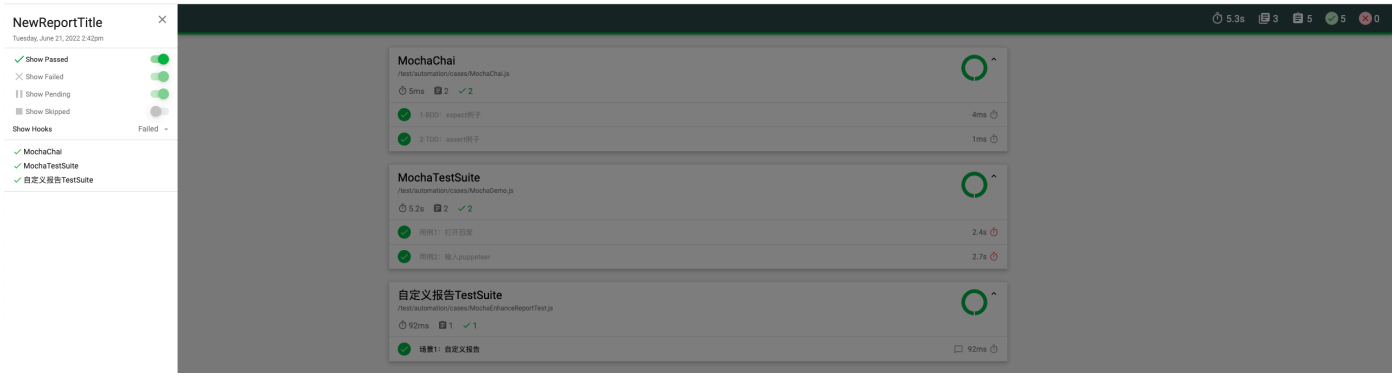
```
"report": "./node_modules/mocha/bin/mocha --recursive -t 10000 -s 2000 --reporter mochawesome test/automation/cases/Mocha*.js --reporter-options reportDir=test/report,reportFilename=reportFile,reportPageTitle=NewPageTitle,reportTitle=NewReportTitle,code=false,charts=true"
```

2-运行report命令

可以发现报告目录指定到test/report下



查看 test/report/reportFile.html



3-参数解释

[完整参数解释]

可以对比上面的命令配置查看具体报告的差别：

Flag	Type	Default	Description
-f, --reportFilename	string	mochawesome	测试报告文件名称(html,json)
-o, --reportDir	string	mochawesome-report	测试报告指定存放目录
-t, --reportTitle	string	mochawesome	测试报告左上角标题
-p, --reportPageTitle	string	mochawesome-report	测试报告浏览器标题
-i, --inline	boolean	false	内联报告
--cdn	boolean	false	CDN加载
--assetsDir	string	/mochawesome-report/assets	assets目录
--charts	boolean	false	是否展示圆饼图统计成功与失败
--code	boolean	true	是否展示源码，默认展示
--autoOpen	boolean	false	是否自动打开
--overwrite	boolean	true	是否重写
--timestamp, --ts	string		时间戳格式，默认：Tuesday, June 21, 2022 2:42pm
--showPassed	boolean	true	是否展示：通过的报告
--showFailed	boolean	true	是否展示：失败的报告
--showPending	boolean	true	是否展示：挂起的报告
--showSkipped	boolean	false	是否展示：跳过的报告
--showHooks	string	failed	展示hooks函数信息
--saveJson	boolean	false	是否保存为json文件
--saveHtml	boolean	true	是否保存为html文件，默认是
--dev	boolean	false	是否开启 dev mode

5-分组

由于package.json命令可以指定运行的目录，所以通常的情况是：

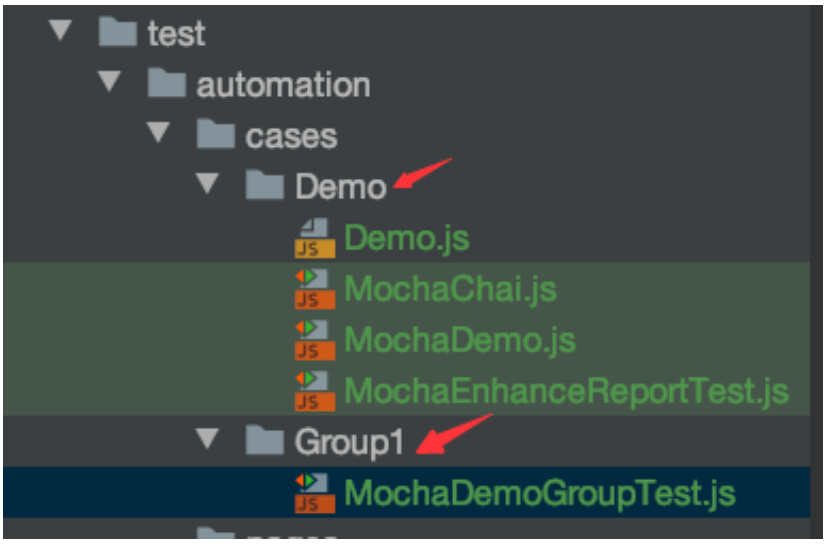
->>测试用例目录分组

->>测试报告分组

->>设置不同的命令

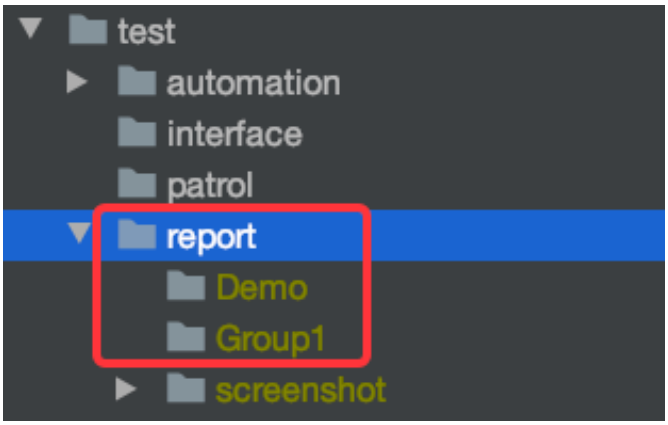
1-将cases目录分组

cases目录下：Demo分组，Group1分组(复制一个用例进去即可)



2-测试报告分组

test/report 创建对应的分组:Demo,Group1



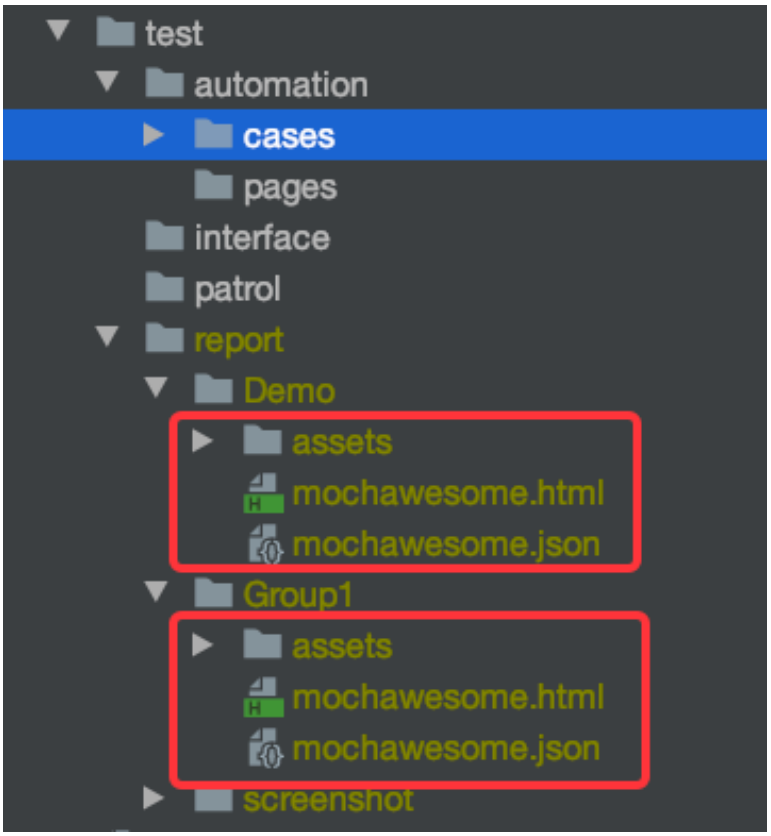
3-分组命令

package.json新增

```
"demo_report": "./node_modules/mocha/bin/mocha --recursive -t 10000 -s 2000 --reporter
mochawesome test/automation/cases/Demo/Mocha*.js --reporter-options
reportDir=test/report/Demo",
"g1_report": "./node_modules/mocha/bin/mocha --recursive -t 10000 -s 2000 --reporter
mochawesome test/automation/cases/Group1/ --reporter-options
reportDir=test/report/Group1"
```

4-查看

分别运行上面新增的命令，查看报告

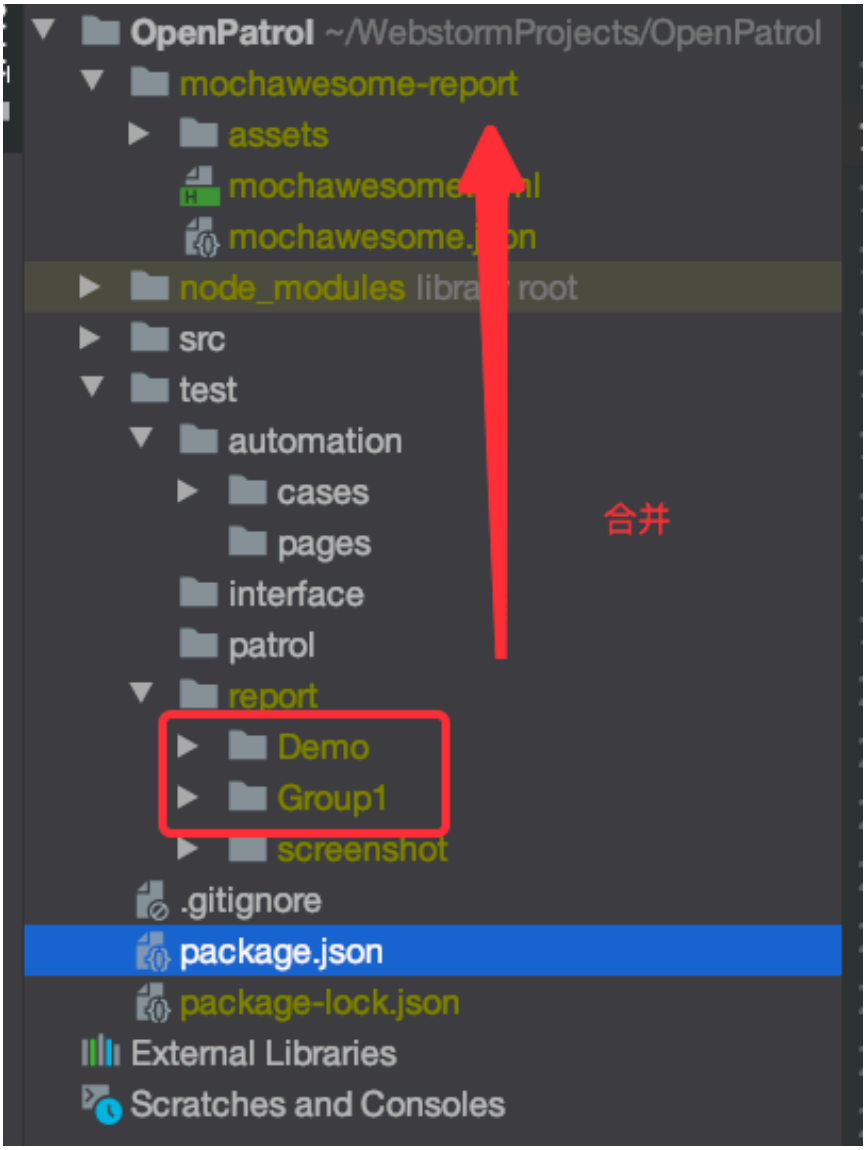


6-合并

按照5的方法分组运行产生报告后，如果有些合并需求（例如：查看某几个分组的合并报告），mochawesome-merge 可以将多个分组的报告进行合并后展示。（说明：这些依赖创建工程时都已经引入）

1-合并报告

将Demo，Group1的报告合并，生成到mochawesome-report（说明：暂不支持指定目录）



2-新增命令

package.json新增合并命令:

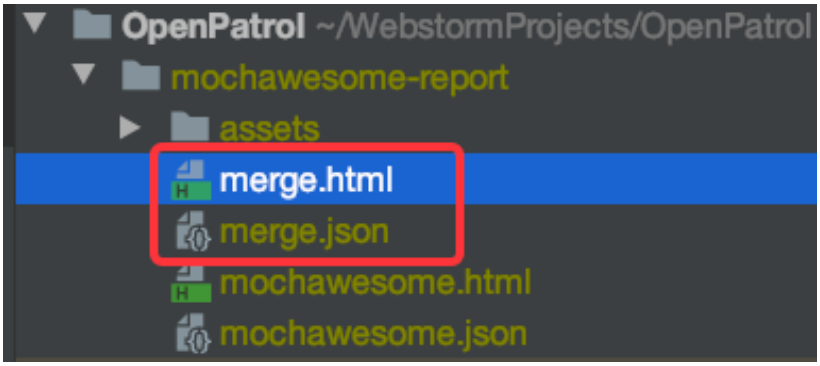
```
"mocha_merge": "mochawesome-merge ./test/report/**/*.json > ./mochawesome-report/merge.json",
"merge_report": "npm run mocha_merge && marge ./mochawesome-report/merge.json "
```

注意此时的路径设置!

3-运行merge_report

```
Run:  merge_report
/usr/local/bin/node /usr/local/lib/node_modules/npm/bin/npm-cli.js run merge_report --scripts-prepend-node-path=auto

> openpatrol@0.0.1 merge_report
> npm run mocha_merge && marge ./mochawesome-report/merge.json


> openpatrol@0.0.1 mocha_merge
> mochawesome-merge ./test/report/**/*.json > ./mochawesome-report/merge.json


✓ Reports saved:
/Users/wulei/WebstormProjects/OpenPatrol/mochawesome-report/merge.html

Process finished with exit code 0
```

4-查看报告



查看发现2个报告合并到一起展示：