本文主要介绍通过钩子函数进行测试环境的初始化/回收操作，并且进行测试框架的全局配置操作

1-钩子函数

Mocha在describe块之中提供的4个钩子函数：before()、after()、beforeEach()和afterEach()，它们会在指定时间执行，类似TestNG的注解

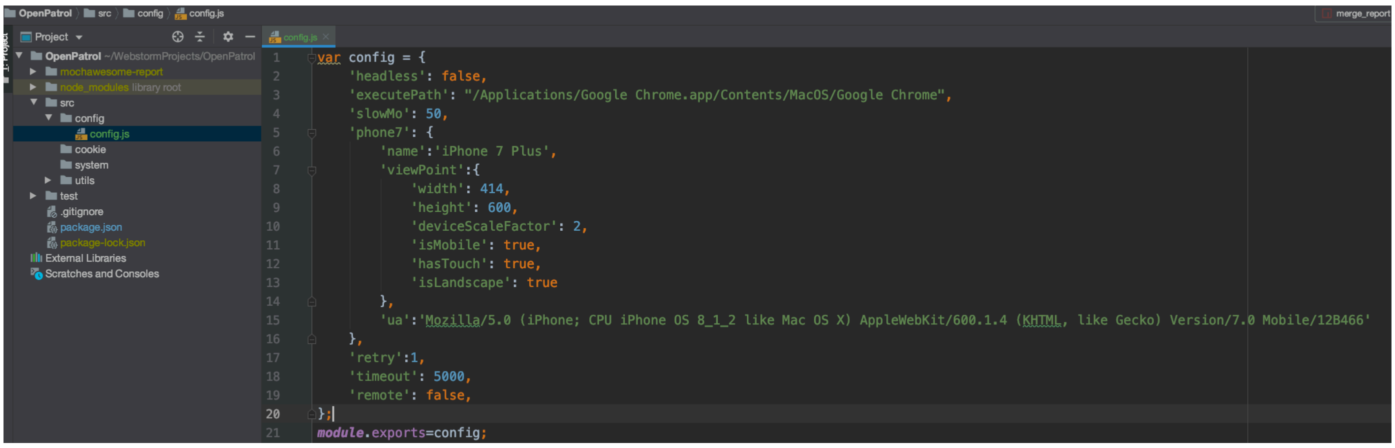
```
describe('hooks', function() {
    before(function() {
      // 在本区块的所有测试用例之前执行
    });
    after(function() {
      // 在本区块的所有测试用例之后执行
    });
    beforeEach(function() {
      // 在本区块的每个测试用例之前执行
    });
    afterEach(function() {
      // 在本区块的每个测试用例之后执行
    });
});
```

在before/after阶段可以方便的进行测试环境的初始化和回收

2-全局配置文件

创建：src/config/config.js

```
var config = {
    'headless': false,
    'executePath': "/Applications/Google Chrome.app/Contents/MacOS/Google Chrome",
    'slowMo': 50,
    'phone7': {
        'name':'iPhone 7 Plus',
        'viewPoint':{
            'width': 414,
            'height': 600,
            'deviceScaleFactor': 2,
            'isMobile': true,
            'hasTouch': true,
            'isLandscape': true
        },
        'ua':'Mozilla/5.0 (iPhone; CPU iPhone OS 8_1_2 like Mac OS X)
AppleWebKit/600.1.4 (KHTML, like Gecko) Version/7.0 Mobile/12B466'
    },
    'retry':1,
    'timeout': 5000,
    'remote': false,
};
module.exports=config;
```



```
var config = {
    'headless': false,
    'executePath': "/Applications/Google Chrome.app/Contents/MacOS/Google Chrome",
    'slowMo': 50,
    'phone7': {
        'name':'iPhone 7 Plus',
        'viewPoint':{
            'width': 414,
            'height': 600,
            'deviceScaleFactor': 2,
            'isMobile': true,
            'hasTouch': true,
            'isLandscape': true
        },
        'ua':'Mozilla/5.0 (iPhone; CPU iPhone OS 8_1_2 like Mac OS X) AppleWebKit/600.1.4 (KHTML, like Gecko) Version/7.0 Mobile/12B466'
    },
    'retry':1,
    'timeout': 5000,
    'remote': false,
};
module.exports=config;
```

3-全局初始化

创建：src/system/init.js

```
const puppeteer = require('puppeteer');
const defaultConfig = require('../../src/config/config')

module.exports = {
    setConfig: function () {//全局配置
        if (global.config != null || global.config != undefined) {
            return;
        }
        global.config = {
            ...defaultConfig,
        }
    },
    createBrowser: async function() {//生成浏览器
        // 加载全局配置
        this.setConfig();
        let args = ['--disable-dev-shm-usage','--disable-setuid-sandbox','--material-
hybrid', '--no-sandbox']

        if (global.config.remote == false) {//默认情况下
            const browser = await puppeteer.launch({
                headless: global.config.headless, // 默认为无界面模式
                slowMo: global.config.slowMo, // 调试用，比较慢的模式
                isMobile: true,
                hasTouch: true,
                ignoreHTTPSErrors:true,
                args,
                executablePath: global.config.executePath

            });
            return browser;
        }else {
            const browser = await puppeteer.connect({
                defaultViewport: null,
                headless: true, // 默认为无界面模式
                slowMo: global.config.slowMo, // 调试用，比较慢的模式
                isMobile: true,
                hasTouch: true,
                ignoreHTTPSErrors:true,
                args,
                executablePath: global.config.executePath,
            });
            return browser;
        }
    },
    createPage: async function(browser) {//创建页面
        if (global.config.remote == false) {//默认情况，本地
            const page = await browser.newPage();
            await page.setViewport({
                'width': 1280,
                'height': 800
            });
            page.setDefaultNavigationTimeout(global.config.timeout);
            let targets = await browser.targets();
            const targetPages = await targets.filter(target => target.type() ===
'page');
            //如果打开了多个页面，那么就要切换到最后一个页面
            if (targetPages.length > 2) {
                await targetPages[targetPages.length - 1].page();
            }
            global.page = page;
            return page;
        } else {
            const pages = await browser.pages();
            const page = pages[0];
            page.setViewport({
                'width': 1280,
                'height': 800
            });
            page.setDefaultNavigationTimeout(global.config.timeout);
            let targets = await browser.targets();
            const targetPages = await targets.filter(target => target.type() ===
'page');
            //如果打开了多个页面，那么就要切换到最后一个页面
            if (targetPages.length > 2) {
                await targetPages[targetPages.length - 1].page();
            }
            global.page = page;
            return page;
        }
    }
}
```

说明：

》global是mocha.js定义的全局变量，一些公共的变量，可以放到global下

》global.config作为框架后续的全局配置

》setConfig方法：初始化全局配置，默认使用src/config/config.js的配置

》createBrowser方法：通过全局配置，创建全局的浏览器

》createPage方法：创建全局的page，兼容了一些异常情况

4-创建Base

创建：test/Base.js

```
const assert = require('assert');
const init = require('../src/system/init');

class Base{
    static async before() {//before初始化页面
        try {
            console.log(this.currentTest.fullTitle()+" before running");
            this.currentTest.retries(3);//全局设置失败重试次数=3
            if (global.browser != null || global.browser != undefined) {
                await global.browser.close();
            }
            global.browser = await init.createBrowser();
            await init.createPage(browser);
        } catch (e) {
            console.log(this.currentTest.fullTitle()+" before running failed,msg="+e);
            assert.ok(false);
        }

    }
    static async after() {///after关闭页面
        try {
            console.log(this.currentTest.fullTitle()+" after running");
            await global.browser.close();
        } catch (e) {
            console.log(this.currentTest.fullTitle()+" after running failed:浏览器关闭异
常,msg="+e);
            assert.ok(false);
        }
    }

    static async afterEach() {//失败截图
        try {
            console.log(this.currentTest.fullTitle()+" afterEach running");
            if (this.currentTest.state == 'failed') {
                console.log("开始截图")
                let currentTime = new Date().getTime();
                await page.screenshot({
                    path:
'./test/report/screenshot/'+this.currentTest.title+currentTime+'.png',
                    type: 'png',
                    fullPage: true
                });
            }
        } catch (e) {
            console.log("截图失败,msg="+e);
        }
    }
}

module.exports=Base;
```

自定义before，after，afterEach方法

before方法

功能：设置全部失败重拾次数；创建全局的global.browser/global.page；

after方法

功能：关闭global.browser

afterEach方法

功能：失败的方法截图并且保存到指定的目录

this属性获取

在方法里面使用到：this.currentTest，完整的thist属性可以通过如下方法：

for (var key in this){

 console.log("test="+key);

}

输出：

this.=_runnable

this.=test

this.=currentTest

this.=runnable

this.=timeout

this.=slow

this.=skip

this.=retries

this.currentTest属性获取

在方法里面使用到：this.currentTest.state，完整的this.currentTest属性可以通过如下方法：

```
for (var key in this.currentTest ){
  console.log("test.currentTest="+key);
}
```

输出：

```
this.currentTest=type
this.currentTest=title
this.currentTest=fn
this.currentTest=body
this.currentTest=async
this.currentTest=sync
this.currentTest=_timeout
this.currentTest=_slow
this.currentTest=_retries
this.currentTest=timedOut
this.currentTest=_currentRetry
this.currentTest=pending
this.currentTest=file
this.currentTest=parent
this.currentTest=ctx
this.currentTest=intellij_test_node
this.currentTest=_events
this.currentTest=_eventsCount
this.currentTest=callback
this.currentTest=timer
```

this.currentTest=duration

this.currentTest=state

this.currentTest=speed

this.currentTest=reset

this.currentTest=retriedTest

this.currentTest=markOnly

this.currentTest=clone

this.currentTest=serialize

this.currentTest=timeout

this.currentTest=slow

this.currentTest=skip

this.currentTest=isPending

this.currentTest=isFailed

this.currentTest=isPassed

this.currentTest=retries

this.currentTest=currentRetry

this.currentTest=fullTitle

this.currentTest=titlePath

this.currentTest=clearTimeout

this.currentTest=resetTimeout

this.currentTest=globals

this.currentTest=run

this.currentTest=_timeoutError

this.currentTest=_maxListeners

this.currentTest=setMaxListeners

this.currentTest=getMaxListeners

this.currentTest=emit

this.currentTest=addListener

this.currentTest=on

this.currentTest=prependListener

this.currentTest=once

this.currentTest=prependOnceListener

this.currentTest=removeListener

this.currentTest=off

this.currentTest=removeAllListeners

this.currentTest=listeners

this.currentTest=rawListeners

this.currentTest=listenerCount

this.currentTest=eventNames

5-测试Base

创建完毕Base后，后续需要进行UI相关的自动化时，可以继承Base的方法：

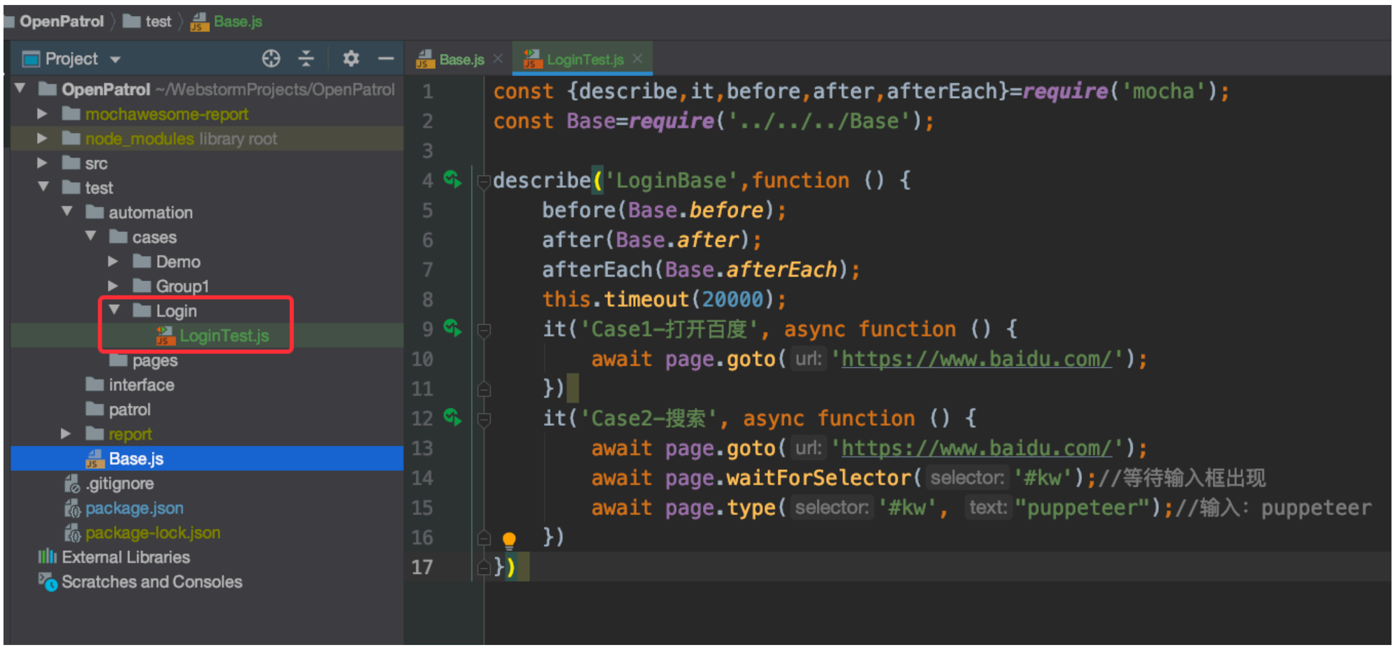
1-创建测试代码

创建：test/automation/cases/Login/LoginTest.js

```
const {describe,it,before,after,afterEach}=require('mocha');
const Base=require('../../../Base');

describe('LoginBase',function () {
    before(Base.before);
    after(Base.after);
    afterEach(Base.afterEach);
    this.timeout(20000);
    it('Case1-打开百度', async function () {
        await page.goto('https://www.baidu.com/');
    })
    it('Case2-搜索', async function () {
        await page.goto('https://www.baidu.com/');
        await page.waitForSelector('#kw');//等待输入框出现
        await page.type('#kw', "puppeteer");//输入：puppeteer
    })
})
```

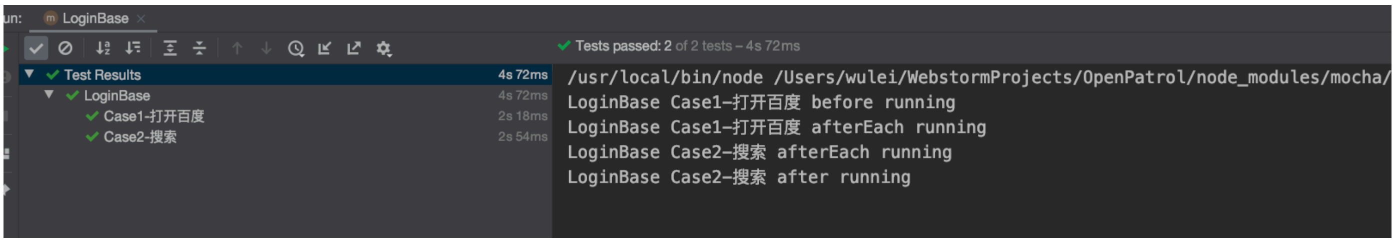
可以看到在钩子函数中，调用了我们自定义的初始化相关的方法，可以完成全局的page的生成，后续的测试it可以直接使用page



```
const {describe,it,before,after,afterEach}=require('mocha');
const Base=require('../../../Base');

describe('LoginBase',function () {
    before(Base.before);
    after(Base.after);
    afterEach(Base.afterEach);
    this.timeout(20000);
    it('Case1-打开百度', async function () {
        await page.goto( url: 'https://www.baidu.com/');
    })
    it('Case2-搜索', async function () {
        await page.goto( url: 'https://www.baidu.com/');
        await page.waitForSelector( selector: '#kw');//等待输入框出现
        await page.type( selector: '#kw', text: "puppeteer");//输入: puppeteer
    })
})
```

2-运行



```
Tests passed: 2 of 2 tests – 4s 72ms

Test Results                4s 72ms
  LoginBase                 4s 72ms
    Case1-打开百度          2s 18ms
    Case2-搜索              2s 54ms
```

```
/usr/local/bin/node /Users/wulei/WebstormProjects/OpenPatrol/node_modules/mocha/
LoginBase Case1-打开百度 before running
LoginBase Case1-打开百度 afterEach running
LoginBase Case2-搜索 afterEach running
LoginBase Case2-搜索 after running
```

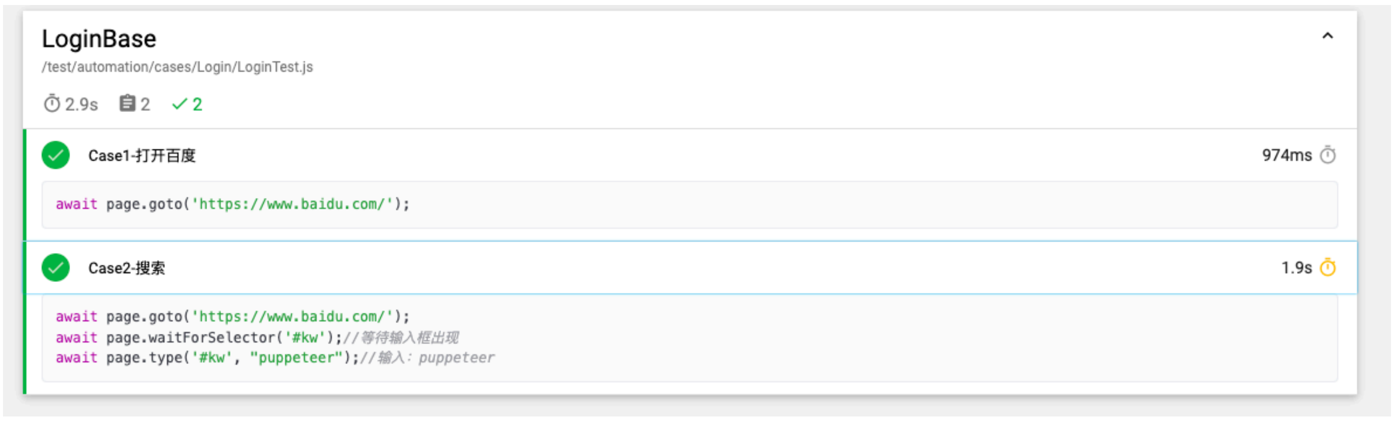
3-创建命令

修改package.json命令

```
"report": "./node_modules/mocha/bin/mocha --recursive -t 10000 -s 2000 --reporter
mochawesome test/automation/cases/Login/",
```

运行：npm run report

查看报告



LoginBase
/test/automation/cases/Login/LoginTest.js
2.9s 2 ✓ 2

✓ Case1-打开百度 974ms

```
await page.goto('https://www.baidu.com/');
```

✓ Case2-搜索 1.9s

```
await page.goto('https://www.baidu.com/');
await page.waitForSelector('#kw');//等待输入框出现
await page.type('#kw', "puppeteer");//输入: puppeteer
```