本文主要介绍Mocha测试框架的使用，进行UI自动化的测试用例编写

Mocha简介

官方文档： https://mochajs.cn/

Mocha（发音"摩卡"）诞生于2011年，最流行的JavaScript测试框架之一，在浏览器和Node环境都可以使用

Mocha 提供的如下基本功能模块：

describe()

context()

it()

before()

beforeEach()

afterEach()

after()

.only()

.skip()

简单理解几个概念：

describe():单测类

describe块称为"测试套件"（test suite），类比TestNG的**@Test类**，表示一组相关的测试。它是一个函数，第一个参数是测试套件的名称（"加法函数的测试"），第二个参数是一个实际执行的函数。

it()：单测方法

it块称为"测试用例"（test case），类比TestNG的**@Test方法**，表示一个单独的测试，是测试的最小单位。它也是一个函数，第一个参数是测试用例的名称（"1 加 1 应该等于 2"），第二个参数是一个实际执行的函数。

钩子函数

Mocha在describe块之中，提供测试用例的四个钩子：before()、after()、beforeEach()和afterEach()。它们会在指定时间执行。类似TestNG的：**@BeforeClass，@AfterClass，@BeforeMethod，@AfterMethod**。

Skip()

跳过执行，类别TestNG的：**@Test(enabled=false)**

Only()

仅运行

Mocha简单例子

1-创建新文件

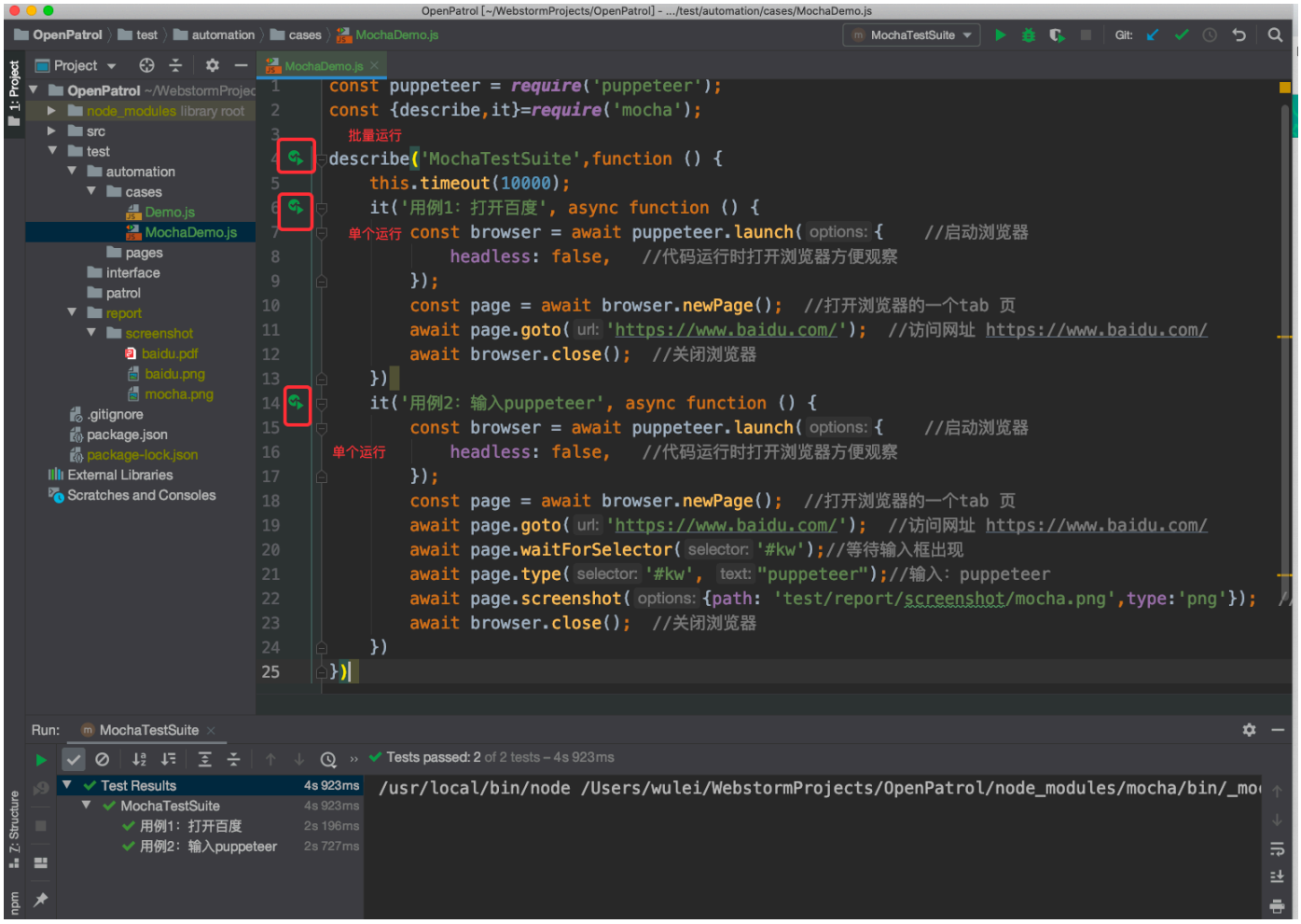
新建：test/automation/cases/MochaDemo.js

代码：

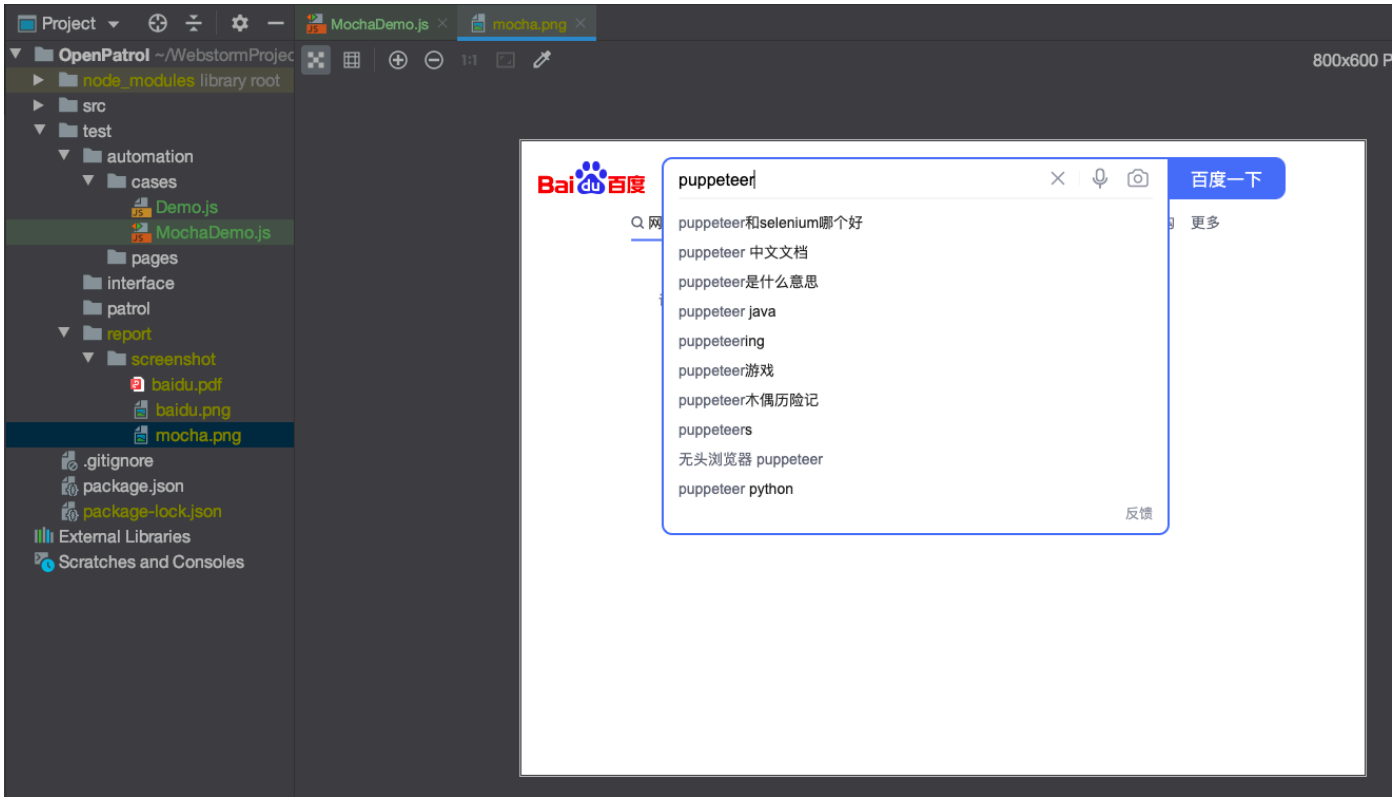
```
const puppeteer = require('puppeteer');
const {describe,it}=require('mocha');

describe('MochaTestSuite',function () {
    this.timeout(10000);
    it('用例1：打开百度', async function () {
        const browser = await puppeteer.launch({     //启动浏览器
            headless: false,   //代码运行时打开浏览器方便观察
        });
        const page = await browser.newPage();  //打开浏览器的一个tab 页
        await page.goto('https://www.baidu.com/');  //访问网址 https://www.baidu.com/
        await browser.close();  //关闭浏览器
    })
    it('用例2：输入puppeteer', async function () {
        const browser = await puppeteer.launch({     //启动浏览器
            headless: false,   //代码运行时打开浏览器方便观察
        });
        const page = await browser.newPage();  //打开浏览器的一个tab 页
        await page.goto('https://www.baidu.com/');  //访问网址 https://www.baidu.com/
        await page.waitForSelector('#kw');//等待输入框出现
        await page.type('#kw', "puppeteer");//输入：puppeteer
        await page.screenshot({path: 'test/report/screenshot/mocha.png',type:'png'});
 //截图
        await browser.close();  //关闭浏览器
    })
})
```

2-运行

可以选择批量或者单个运行，运行后截图保存成功



截图：



Mocha钩子函数

引入钩子函数：

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

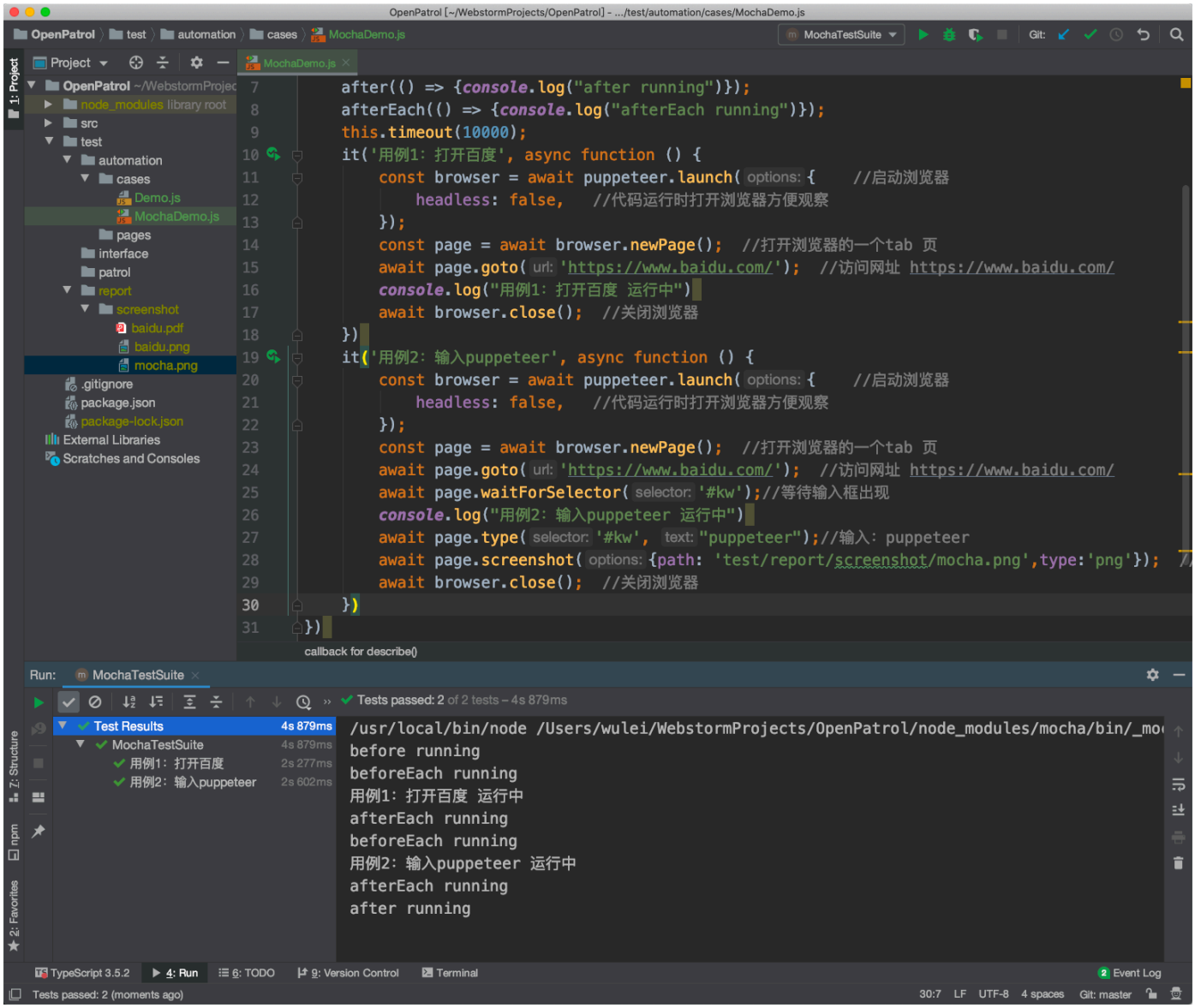
在it运行前，定义钩子函数

```
before(() => {console.log("before running")});
beforeEach(() => {console.log("beforeEach running")});
after(() => {console.log("after running")});
afterEach(() => {console.log("afterEach running")});
```

完整代码：

```
const puppeteer = require('puppeteer');
const {describe,it,before,beforeEach,after,afterEach}=require('mocha');

describe('MochaTestSuite',function () {
    before(() => {console.log("before running")});
    beforeEach(() => {console.log("beforeEach running")});
    after(() => {console.log("after running")});
    afterEach(() => {console.log("afterEach running")});
    this.timeout(10000);
    it('用例1：打开百度', async function () {
        const browser = await puppeteer.launch({    //启动浏览器
            headless: false,   //代码运行时打开浏览器方便观察
        });
        const page = await browser.newPage();  //打开浏览器的一个tab 页
        await page.goto('https://www.baidu.com/');  //访问网址 https://www.baidu.com/
        console.log("用例1：打开百度 运行中")
        await browser.close();  //关闭浏览器
    })
    it('用例2：输入puppeteer', async function () {
        const browser = await puppeteer.launch({    //启动浏览器
            headless: false,   //代码运行时打开浏览器方便观察
        });
        const page = await browser.newPage();  //打开浏览器的一个tab 页
        await page.goto('https://www.baidu.com/');  //访问网址 https://www.baidu.com/
        await page.waitForSelector('#kw');//等待输入框出现
        console.log("用例2：输入puppeteer 运行中")
        await page.type('#kw', "puppeteer");//输入：puppeteer
        await page.screenshot({path: 'test/report/screenshot/mocha.png',type:'png'});
 //截图
        await browser.close();  //关闭浏览器
    })
})
```

重新运行：



可以看到：before/after运行一次，beforeEach/afterEach 在每个IT前后都运行一次

说明：在自动化测试中，钩子函数可以作为环境/数据的初始化和回收

Mocha skip/only

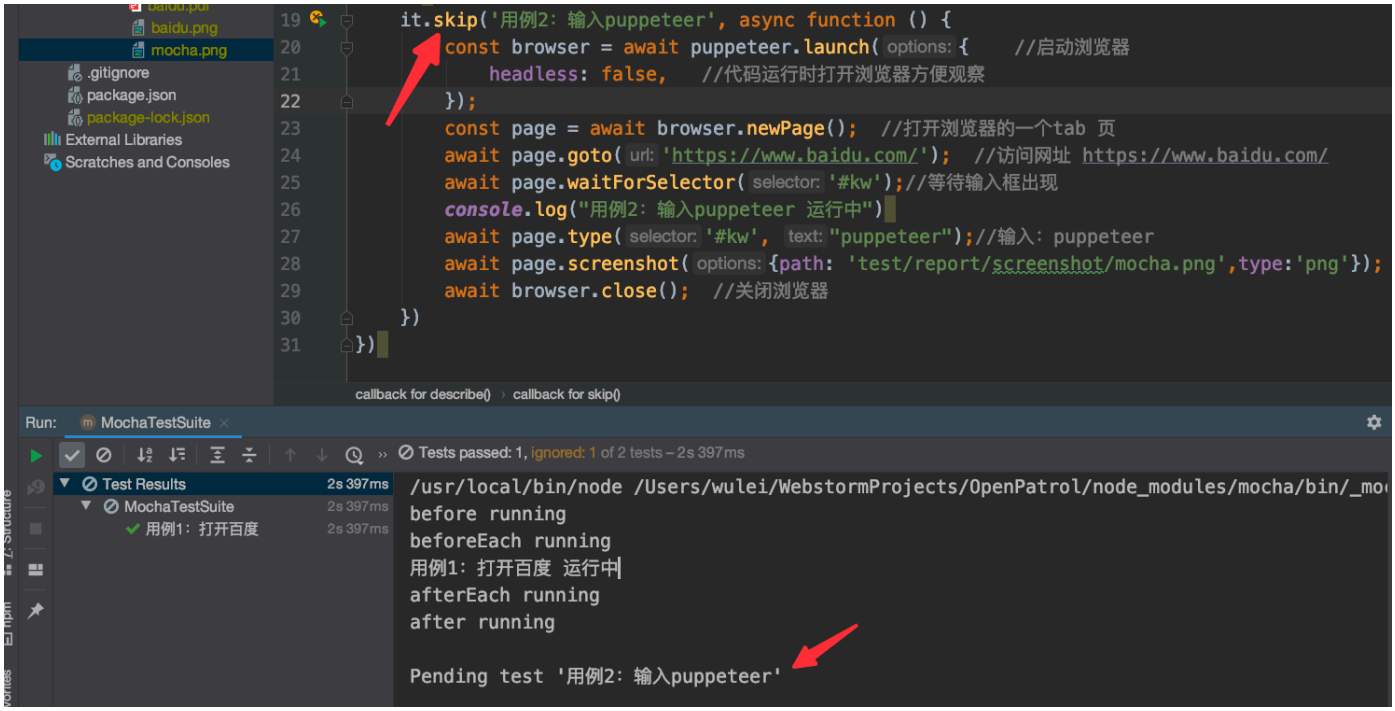
describe 块和 it 块都允许调用 only() 和 skip() 方法。

only() 方法表示在当前的父 describe 块下，只执行该单元的测试。

skip() 方法表示在当前的父 describe 块下，跳过不执行该单元的测试。

当在一个 describe 块下，同时存在 only() 和 skip() 方法，则只执行 .only() 方法。

下面是it.skip()例子，其他类比即可


baidu.png
mocha.png
.gitignore
package.json
package-lock.json
External Libraries
Scratches and Consoles
it.skip('用例2: 输入puppeteer', async function () {
const browser = await puppeteer.launch(options: { //启动浏览器
headless: false, //代码运行时打开浏览器方便观察
});
const page = await browser.newPage(); //打开浏览器的一个tab 页
await page.goto(url: 'https://www.baidu.com/'); //访问网址 https://www.baidu.com/
await page.waitForSelector(selector: '#kw');//等待输入框出现
console.log("用例2: 输入puppeteer 运行中")
await page.type(selector: '#kw', text: "puppeteer");//输入: puppeteer
await page.screenshot(options: {path: 'test/report/screenshot/mocha.png',type:'png'});
await browser.close(); //关闭浏览器
})
})
callback for describe() › callback for skip()
Run: MochaTestSuite
Tests passed: 1, ignored: 1 of 2 tests – 2s 397ms
Test Results 2s 397ms
MochaTestSuite 2s 397ms
用例1: 打开百度 2s 397ms
/usr/local/bin/node /Users/wulei/WebstormProjects/OpenPatrol/node_modules/mocha/bin/_mo
before running
beforeEach running
用例1: 打开百度 运行中
afterEach running
after running
Pending test '用例2: 输入puppeteer'