

本文主要介绍如何通过cookie处理进行快速登陆

cookie登陆于巡检优化

Puppeteer的Page提供了cookie的处理方法

方法	说明
page.setCookie()	设置指定cookie
page.cookies()	返回全部cookie信息
page.deleteCookie()	删除指定cookie

通过cookie访问页面时可以直接访问。

1-cookie登陆流程

- 》 登录时， cookie校验
- 如果有正确的cookie信息-直接登陆；
- 如果无正确的cookie信息：
- 》 登陆后浏览器带有完整的cookie信息， 设置为： global.cookies
- 》 写入cookie信息到文件

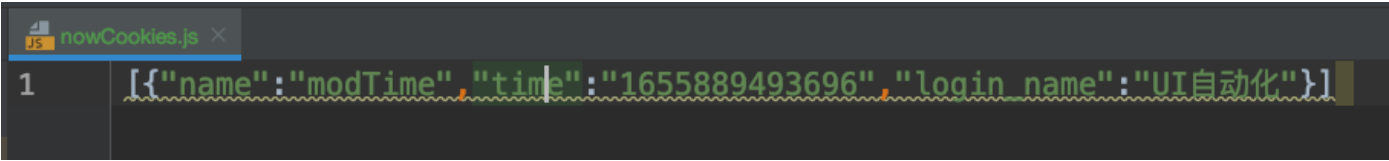
2-cookie文件

修改： src/config/config.js 新增cookies的文件目录设置

```
'login_name': 'UI自动化',
'login_pwd': '520',
'cookies': './src/cookie/nowCookies.js',
```

新建文件： src/cookie/nowCookies.js

```
[{"name": "modTime", "time": "1655889493696", "login_name": "UI自动化"}]
```



3-Base cookie处理

修改： test/Base.js

before方法： 每个用例开始前， 如果有未关闭的浏览器， 先将cookie设置为undefined， 再关闭

after方法：每个用例结束后，先将cookie设置为undefined，再关闭

```
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) {
      global.cookies = undefined;///开始前如果有未关闭的浏览器，先cookie置为空
      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( value: false);
  }
}

static async after() {///after关闭页面
  try {
    console.log(this.currentTest.fullTitle()+" after running");
    global.cookies = undefined;///最后cookie置为空
    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( value: false);
  }
}
```

完整代码：

```
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) {
        global.cookies = undefined;///开始前如果有未关闭的浏览器，先cookie置为空
        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");
        global.cookies = undefined;//最后cookie置为空
        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/screenShot/'+this.currentTest.title+currentTime+'.png',
                type: 'png',
                fullPage: true
            });
        }
    } catch (e) {
        console.log("截图失败,msg="+e);
    }
}

module.exports=Base;

```

4-cookie写入

⚠️ 写入cookie到文件时，我们多写一段自定义内容用于后续校验

⚠️ fs库提供了文件操作的方法

```

/**
 * 写入cookie: 浏览器cookies+自定义的cookie信息一起存在文件里持久化
 * 自定义cookie信息主要是记录了时间
 */
writeCookie: async function (login_name,cookieFile) {
    if (cookieFile == null) {
        return console.error(cookieFile);
    }
    //读取浏览器cookies信息
    let cookie = await page.cookies();
    //自定义cookie检查内容
    let check = {name: "modTime"};

```

```

check.time = new Date().getTime().toString();
check.login_name = login_name;
check.cookie_str = cookie.map(item=>`${item.name}=${item.value}`).join(';');//写
入格式化后的cookie
//将自定义的cookie信息加入到浏览器读取的内容中
cookie.push(check);
const cookieJson = JSON.stringify(cookie);
//浏览器+自定义的cookieJson写入到cookieFile（文件）中
fs.writeFile(cookieFile, cookieJson, function (err) {
    if (err) {
        return;
    }
    console.log("异步cookies文件写入成功");
    fs.readFile(cookieFile, function (err) {
        if (err) {
            return console.error(err);
        }
        console.log("异步读取cookies文件成功 ");
    });
});
}

```

⚠️: `check.cookie_str = cookie.map(item=>`${item.name}=${item.value}`).join(';');//写入格式化后的cookie`
 这一步的操作是`page.cookies()`获取的cookie数组，转化为浏览器形式的cookie字符串

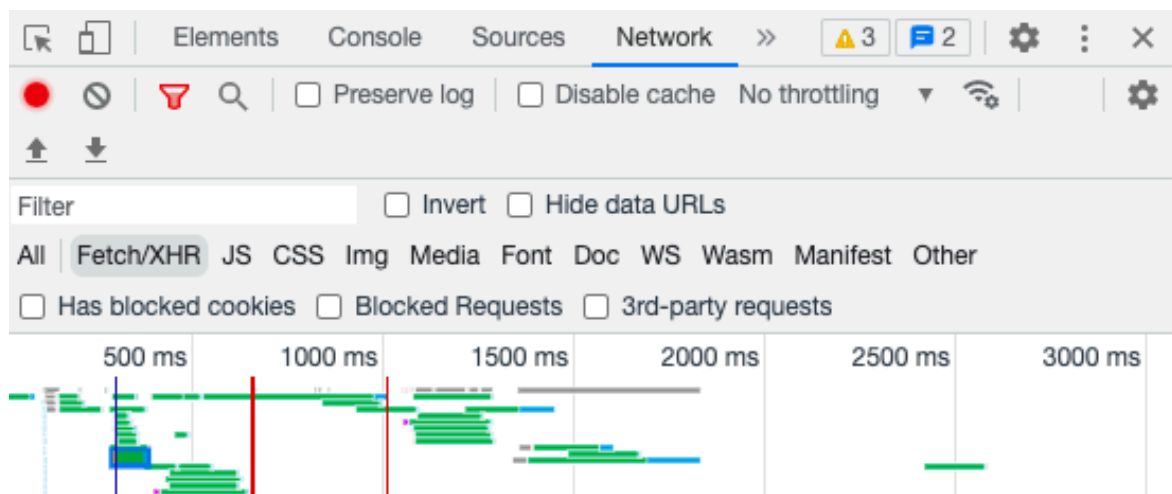
`page.cookies()` 例子:

```

[{"name": "_gat_gtag_UA_476124_1", "value": "1", "domain": ".cnblogs.com", "path": "/", "expires": 1656406444, "size": 22, "httpOnly": false, "secure": false, "session": false, "sameParty": false, "sourceScheme": "Secure", "sourcePort": 443},
{"name": "_gid", "value": "GA1.2.1776849710.1656406384", "domain": ".cnblogs.com", "path": "/", "expires": 1656492784, "size": 31, "httpOnly": false, "secure": false, "session": false, "sameParty": false, "sourceScheme": "Secure", "sourcePort": 443}]

```

浏览器cookie例子:



Name	× Headers Preview Response Initiator >>
<input type="checkbox"/> ppub_config?ipd=www.cnb...	vary: Origin
<input type="checkbox"/> allsitecategories	x-content-type-options: nosniff
<input type="checkbox"/> B1	x-frame-options: SameOrigin
<input type="checkbox"/> creative	
<input type="checkbox"/> SideRight	▼Request Headers View sou
<input type="checkbox"/> blocks	Accept: application/json, text/javascript, */
<input checked="" type="checkbox"/> userinfo	*; q=0.01
<input type="checkbox"/> pref	Accept-Encoding: gzip, deflate, br
<input type="checkbox"/> ads?pvsid=15953208146134...	Accept-Language: zh-CN,zh;q=0.9
<input type="checkbox"/> collect?v=1&_v=j96&a=7008...	Connection: keep-alive
<input type="checkbox"/> postIds	Cookie: _gid=GA1.2.331396283.1656412739; _ga=G
<input type="checkbox"/> getconfig?url=https%3A%2...	A1.2.1340719590.1656412731; Hm_lvt_866c9be12d
<input type="checkbox"/> sodar?sv=200&tid=gpt&tv=2...	4a814454792b1fd0fed295=1656412739; _ga_3Q0DVS
	GN10=GS1.1.1656412731.1.0.1656412738.53; .CNB
	logsCookie=5C6C22D2EB6DA6EFA632FCA31E3E4EE4E8
	8F8A0DDC528790AC8D5E659EFC617178338FF5901E030
	37FF7AD5E837E3267EB3EB8457C05FA27F3E455D2B2B5
	D32117BD80D0955913B47B0DD60523674B66793C74D0;
	.Cnblogs.AspNetCore.Cookies=CfDJ8E0BBtWq0dNFo
	DS-ZHPSe52xeqNi7XRiQpHMZDi7vKwGPOAopCOAhIt-Q6
	g0IZt1UWE67XU2rJ_6Fbk2FbYtKTqaKwarT_AlcvhPY1V
	y0ewowEhI2TmXS2rLr7qKMYbTrLfqn9gY6nANhd6GzK9y
	-KH7uGp7qattE1nQdjTnBxKniSmlTtuD1WkZTYKhL2yOK
	Q7J0ZuKK-0wTRg6CCYICZhmG1W0hLJPXz9c9Z04l6V46R
	3QgY93bEuYy0SVWAAPjepyvMm4rJIVKftFZ_C-JYonyn9
	bGGC_I1qkvWfJB6LtfVx0UkHeG6Ej_2yPx0hIiPen1qA2
	2e_Y1ECxFs6LIchM9ypRONPgryKdHnw7wBaUoEWBa75kx
	4TNniPtFVMYrYUH9ILQoRTaKhM4ZmEgB866wNzkip2owt
	QVlwh0vD_Px1lpXp2Lox18lYBuxp0_exba2Mn1ozPbjgI
	v4bT7RwTaNFHZ_6e8iSYKWBsDTZyGyKV1eijlm1gQlq1N
	JbSp4-Efg-oz44-VN4Qu0tWk1aR3V0E_n1l_tN8vGPfo7
	Inn6fHvpHNTpEPiMfJN-W4teQ; affinity=165641278
	4.74.100.989171 c1e5fb103ec944b8b4c27a6b37078
	3a0; __gads=ID=efee5014cf80b018-226eaa65dbd40
	03e:T=1656412784:S=ALNI_MbH4QjIL16XfWL_s0lwDp
	7AKUi3Jg; __gpi=UID=0000070d19646a51:T=165641
	2784:RT=1656412784:S=ALNI_MbuaPfk_q0v4Hsjaw8-
	Aosg83DWYA; Hm_lpvt_866c9be12d4a814454792b1fd
	0fed295=1656413155
	Host: account.cnblogs.com

5-cookie读取

cookie读取是从文件读取cookie内容并且设置为page的cookie，并且设置全局cookies信息

```

/**
 * @param cookieFile 需要输入的cookies存储路径, config文件中有默认
 * @returns {Promise.<void>}
 */
readCookie: async function (cookieFile) {
  var data = fs.readFileSync(cookieFile);
  console.log("同步读取cookies文件成功");
  var cookies = JSON.parse(data);
  const check = cookies.pop();//读取cookie文件, POP自定义的信息进行校验
  global.cookieStr=check.cookie_str;//设置全局的参数
  await page.setCookie(...cookies);
  global.cookies = await page.cookies();
}

```

6-cookie校验

⚠ cookie校验的流程是读取cookie文件的内容, 校验我们自定义的cookie内容查看是否过期

⚠ cookie校验通过后直接跳转到指定URL

```

/**
 * 检查cookie信息是否符合要求
 * 如果存在正确的cookie信息, 直接登陆到指定URL
 */
checkCookie: async function (login_name, cookieFile, gotoUrl) {
  const TodayTimeStamp = (new Date(new Date().setHours(0, 0, 0, 0)) / 1000 *
1000).toString();//当天0点的时间戳
  try {
    var data = fs.readFileSync(cookieFile);
  } catch (error) {
    console.log(error);
    return false;
  }
  const cookies = JSON.parse(data);
  const check = cookies.pop();//读取cookie文件, POP自定义的信息进行校验
  if (check.time > TodayTimeStamp) {
    if (check.login_name == login_name) {
      await this.readCookie(cookieFile);
      console.log('cookie校验通过, 直接登陆并跳转到指定URL='+gotoUrl);
      await page.goto(gotoUrl);
      return true;
    }
  }
}

```

7-cookie登陆

⚠ cookie登陆完整流程

```

/**
 * 登陆流程，有cookie直接跳转；无cookie，登陆后写入cookie信息后跳转
 */
loginBlogCheck: async function (url, loginName, loginPwd, cookie_File) {
    let login_name = (loginName == undefined ? global.config.login_name :
loginName);
    let login_pwd = (loginPwd == undefined ? global.config.login_pwd : loginPwd);
    let cookieFile = (cookie_File === undefined ? global.config.cookies :
cookie_File);
    //1-有正确cookie，直接跳转targetURL
    const goodCookies = await this.checkCookie(login_name, cookieFile, url);
    //2-无正确cookie，先登陆，写入cookie信息，后跳转targetURL
    if (!goodCookies) {
        console.log("cookies校验错误，开始登陆并写入cookie");
        if (global.cookies === undefined) {
            //1. 登陆：博客园
            await page.goto("https://account.cnblogs.com/signin");
            //2. 点击：密码登陆
            await page.waitForSelector('#mat-tab-label-0-0 > div',
{timeout:2*1000}).then(ele=>{ele.click()}).catch(e=>{console.log(e)})
            //3. 输入：用户名
            await page.focus('#mat-input-0');
            await page.type('#mat-input-0', login_name);
            //4. 输入：密码
            await page.focus('#mat-input-1');
            await page.type('#mat-input-1', login_pwd);
            //5. 勾选：记住我（可选）
            // await page.click('#mat-checkbox-1 > label > span.mat-checkbox-inner-
container');
            //6. 点击：登陆
            await page.waitForSelector('body > app-root > app-sign-in-layout > div
> div > app-sign-in > app-content-container > div > div > div > form > div > button',
{timeout:2*1000}).then(ele=>{ele.click()}).catch(e=>{console.log(e)})
            //等待响应返回200
            await page.waitForResponse(response => response.status() === 200);
            await page.waitForTimeout(2000);

            //登陆后获取浏览器cookie
            global.cookies = await page.cookies();
            //写入cookie
            await this.writeCookie(login_name, cookieFile);
        }
        // 如果第一次进来的时候有cookie
        let newPageCookies = await page.cookies();
        // 设置global.cookieStr参数

global.cookieStr=newPageCookies.map(item=>`${item.name}=${item.value}`).join(';')
        if (newPageCookies.length === 0) {
            // 这里需要加上"...", 是因为cookies是一个多值的数组

```



```

        console.log("没有cookie, 加进来")
        await page.setCookie(...global.cookies);
    }
    // 选择店铺并跳转到目标页面
    console.log('指定URL', url);
    await page.goto(url,{timeout:20000})
}
}

```

8-cookie处理

创建：src/cookie/HandleCookie.js

```

const fs = require('fs');

const HandleCookie = {
    /**
     * 登陆流程, 有cookie直接跳转; 无cookie, 登陆后写入cookie信息后跳转
     */
    loginBlogCheck: async function (url,loginName,loginPwd,cookie_File) {
        let login_name = (loginName == undefined ? global.config.login_name :
loginName);
        let login_pwd = (loginPwd == undefined ? global.config.login_pwd : loginPwd);
        let cookieFile = (cookie_File === undefined ? global.config.cookies :
cookie_File);
        //1-有正确cookie, 直接跳转targetURL
        const goodCookies = await this.checkCookie(login_name, cookieFile, url);
        //2-无正确cookie, 先登陆, 写入cookie信息, 后跳转targetURL
        if (!goodCookies) {
            console.log("cookies校验错误, 开始登陆并写入cookie");
            if (global.cookies === undefined) {
                //1. 登陆: 博客园
                await page.goto("https://account.cnblogs.com/signin");
                //2. 点击: 密码登陆
                await page.waitForSelector('#mat-tab-label-0-0 > div',
{timeout:2*1000}).then(ele=>{ele.click()}).catch(e=>{console.log(e)})
                //3. 输入: 用户名
                await page.focus('#mat-input-0');
                await page.type('#mat-input-0',login_name);
                //4. 输入: 密码
                await page.focus('#mat-input-1');
                await page.type('#mat-input-1',login_pwd);
                //5. 勾选: 记住我 (可选)
                // await page.click('#mat-checkbox-1 > label > span.mat-checkbox-inner-
container');
                //6. 点击: 登陆
                await page.waitForSelector('body > app-root > app-sign-in-layout > div
> div > app-sign-in > app-content-container > div > div > div > form > div > button',
{timeout:2*1000}).then(ele=>{ele.click()}).catch(e=>{console.log(e)})
            }
        }
    }
}

```



```

        //等待响应返回200
        await page.waitForResponse(response => response.status() === 200);
        await page.waitForTimeout(2000);

        //登陆后获取浏览器cookie
        global.cookies = await page.cookies();
        //写入cookie
        await this.writeCookie(login_name, cookieFile);
    }
    // 如果第一次进来的时候有cookie
    let newPageCookies = await page.cookies();
    // 设置global.cookieStr参数
    global.cookieStr=newPageCookies.map(item=>`${item.name}=${item.value}`).join(';')
    if (newPageCookies.length === 0) {
        // 这里需要加上"...", 是因为cookies是一个多值的数组
        console.log("没有cookie, 加进来")
        await page.setCookie(...global.cookies);
    }
    // 选择店铺并跳转到目标页面
    console.log('指定URL', url);
    await page.goto(url, {timeout:20000})
}
},
/**
 * 检查cookie信息是否符合要求
 * 如果存在正确的cookie信息, 直接登陆到指定URL
 */
checkCookie: async function (login_name, cookieFile, gotoUrl) {
    const TodayTimeStamp = (new Date(new Date().setHours(0, 0, 0, 0)) / 1000 *
1000).toString();//当天0点的时间戳
    try {
        var data = fs.readFileSync(cookieFile);
    } catch (error) {
        console.log(error);
        return false;
    }
    const cookies = JSON.parse(data);
    const check = cookies.pop();//读取cookie文件, POP自定义的信息进行校验
    if (check.time > TodayTimeStamp) {
        if (check.login_name === login_name) {
            await this.readCookie(cookieFile);
            console.log('cookie校验通过, 直接登陆并跳转到指定URL='+gotoUrl)
            await page.goto(gotoUrl);
            return true;
        }
    }
}
},
/**

```

```

* 读取符合要从文件中读取cookies塞到浏览器中
* @param cookieFile 需要输入的cookies存储路径, config文件中有默认
* @returns {Promise.<void>}
*/
readCookie: async function (cookieFile) {
  var data = fs.readFileSync(cookieFile);
  console.log("同步读取cookies文件成功");
  var cookies = JSON.parse(data);
  const check = cookies.pop();//读取cookie文件, POP自定义的信息进行校验
  global.cookieStr=check.cookie_str;//设置全局的参数
  await page.setCookie(...cookies);
  global.cookies = await page.cookies();
},

```

```

/**

```

```

* 写入cookie: 浏览器cookies+自定义的cookie信息一起存在文件里持久化
* 自定义cookie信息主要是记录了时间
*/

```

```

writeCookie: async function (login_name,cookieFile) {
  if (cookieFile == null) {
    return console.error(cookieFile);
  }
  //读取浏览器cookies信息
  let cookie = await page.cookies();
  //自定义cookie检查内容
  let check = {name: "modTime"};
  check.time = new Date().getTime().toString();
  check.login_name = login_name;
  check.cookie_str =cookie.map(item=>`${item.name}=${item.value}`).join(';');//写

```

入格式化后的cookie

```

  //将自定义的cookie信息加入到浏览器读取的内容中
  cookie.push(check);
  const cookieJson = JSON.stringify(cookie);
  //浏览器+自定义的cookieJson写入cookieFile (文件) 中
  fs.writeFile(cookieFile, cookieJson, function (err) {
    if (err) {
      return;
    }
    console.log("异步cookies文件写入成功");
    fs.readFile(cookieFile, function (err) {
      if (err) {
        return console.error(err);
      }
      console.log("异步读取cookies文件成功 ");
    });
  });
},

```

```

},

```

```
};

module.exports=HandleCookie;
```

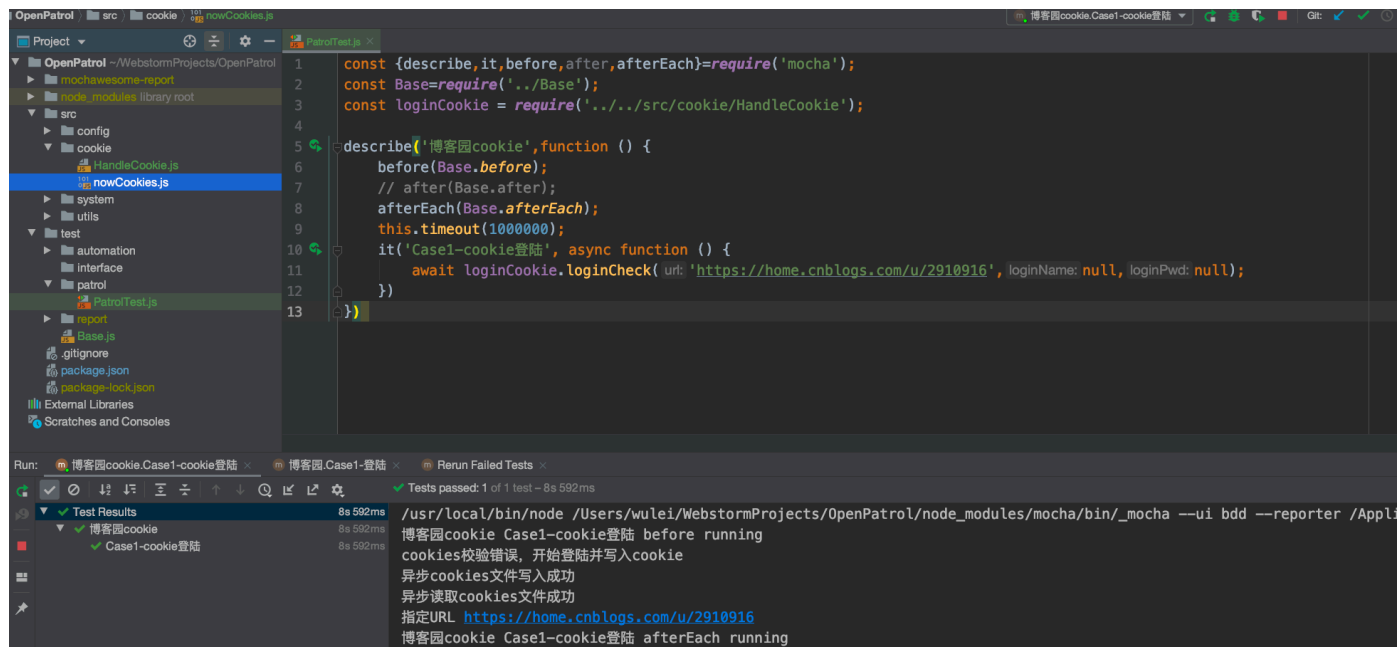
9-cookie登陆测试

新建：test/patrol/PatrolTest.js

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

describe('博客园cookie',function () {
  before(Base.before);
  // after(Base.after);
  afterEach(Base.afterEach);
  this.timeout(1000000);
  it('Case1-cookie登陆', async function () {
    await
loginCookie.loginBlogCheck('https://home.cnblogs.com/u/2910916',null,null);
  })
})
```

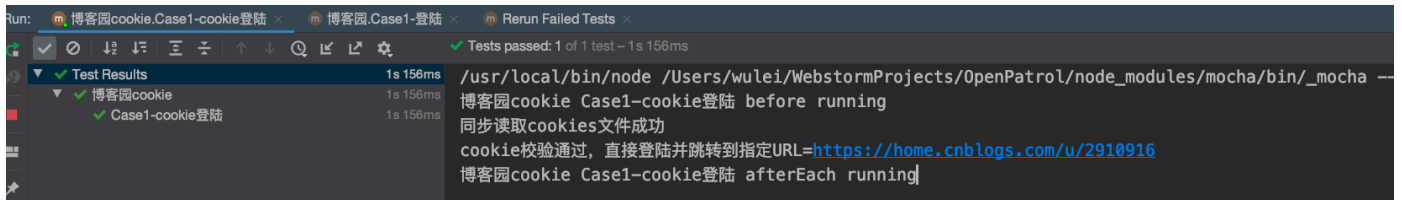
运行：



cookie文件写入成功：



再次运行：发现直接登陆到指定页面



巡检优化

修改：test/patrol/PatrolTest.js

巡检代码优化

```
const {describe,it,before,after,afterEach}=require('mocha');
const Base=require('../Base');
const loginCookie = require('../src/cookie/HandleCookie');
const patrol = require('../src/utlis/Patrol');

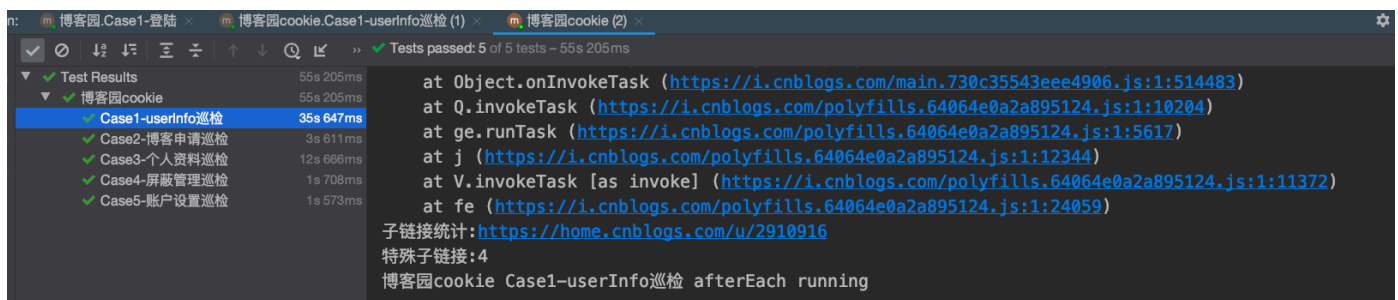
describe('博客园cookie',function () {
  before(Base.before);
  // after(Base.after);
  afterEach(Base.afterEach);
  this.timeout(1000000);
  it('Case1-userInfo巡检', async function () {
    const userInfo="https://home.cnblogs.com/u/2910916";
    await loginCookie.loginBlogCheck(userInfo,null,null);
    //巡检
    await patrol.patrolNormalHref(page,userInfo);
    await patrol.patrolSpecialHref(page,userInfo);
  })
  it('Case2-博客申请巡检', async function () {
    const myBlog="https://account.cnblogs.com/blog-apply";
    await loginCookie.loginBlogCheck(myBlog,null,null);
    //巡检
    await patrol.patrolNormalHref(page,myBlog);
    await patrol.patrolSpecialHref(page,myBlog);
  })
  it('Case3-个人资料巡检', async function () {
    const profile="https://home.cnblogs.com/set/profile/";
```

```

    await loginCookie.loginBlogCheck(profile,null,null);
    //巡检
    await patrol.patrolNormalHref(page,profile);
    await patrol.patrolSpecialHref(page,profile);
  })
it('Case4-屏蔽管理巡检', async function () {
  const blocks="https://account.cnblogs.com/settings/blocks";
  await loginCookie.loginBlogCheck(blocks,null,null);
  //巡检
  await patrol.patrolNormalHref(page,blocks);
  await patrol.patrolSpecialHref(page,blocks);
})
it('Case5-账户设置巡检', async function () {
  const account="https://account.cnblogs.com/settings/account";
  await loginCookie.loginBlogCheck(account,null,null);
  //巡检
  await patrol.patrolNormalHref(page,account);
  await patrol.patrolSpecialHref(page,account);
})
})

```

运行后:



链接巡检合并

代码: src/utlis/Patrol.js

为了减少代码的重复性,可以在patrolNormalHref方法后面,调用patrolSpecialHref方法,这样可以巡检全部链接

```

patrolNormalHref:async function(page,url) {
  //指定访问页面
  await page.goto(url);
  //页面-普通超链接
  const urls = await page.evaluate(
    () => Array.from(document.body.querySelectorAll('a[href*="//\']'), ({ href })
=> href)
  );
  //排重
  const distinct_urls=this.distinct(urls);
  console.log('子链接统计:'+url);
}

```

```
console.log('普通子链接:'+distinct_urls.length);
// 巡检页面超链接
for (const url in distinct_urls){
    await page.goto(distinct_urls[url],{timeout:10000,waitUntil:
'domcontentloaded'}).catch(async error => {
        console.log('巡检异常URL='+distinct_urls[url]);
        console.log('异常信息='+error);
    });
    page.on('pageerror', pageerror=> {
        console.log('page_error='+pageerror);
    });
}
//巡检特殊子链接
await patrolSpecialHref(page,url);
}
```