本文主要介绍如何进行页面超链接的巡检

evaluate系列方法

Puppeteer 的 Page 对象提供了一系列 evaluate 方法，你可以通过他们来执行一些自定义的 js 代码，主要提供了下面三个 API

(1). page.evaluate(pageFunction, ...args)

返回一个可序列化的普通对象，pageFunction 表示要在页面执行的函数， args 表示传入给 pageFunction 的参数， 下面的 pageFunction 和 args 表示同样的意思。

举例：

```
const result = await page.evaluate(x => {
  return Promise.resolve(8 * x);
}, 7); //  (译者注:  7 可以是你自己代码里任意方式得到的值)
console.log(result); // 输出 "56"
```

例子2：参数传递

```
##单参数:
await page.evaluate( example => { /* ... */ }, example );
##两参数:
await page.evaluate( ( example_1, example_2 ) => { /* ... */ }, example_1, example_2 );
##多参数:
let name = 'jack';
let age  = 33;
let location = 'Berlin/Germany';
await page.evaluate(({name, age, location}) => {
    console.log(name);
    console.log(age);
    console.log(location);
},{name, age, location});
```

(2). Page.evaluateHandle(pageFunction, ...args)

在 Page 上下文执行一个 pageFunction, 返回 JSHandle 实体.

例子：获取页面的动态HTML 代码

```
const bodyHandle = await page.evaluateHandle(() => document.body);
const resultHandle = await page.evaluateHandle(body => body.innerHTML, bodyHandle);
console.log(await resultHandle.jsonValue());
await resultHandle.dispose();
```

(3). page.evaluateOnNewDocument(pageFunction, ...args)

在文档页面载入前调用 pageFunction, 如果页面中有 iframe 或者 frame, 则函数调用 的上下文环境将变成子页面的，即iframe 或者 frame, 由于是在页面加载前调用，这个函数一般是用来初始化 javascript 环境的，比如重置或者 初始化一些全局变量。

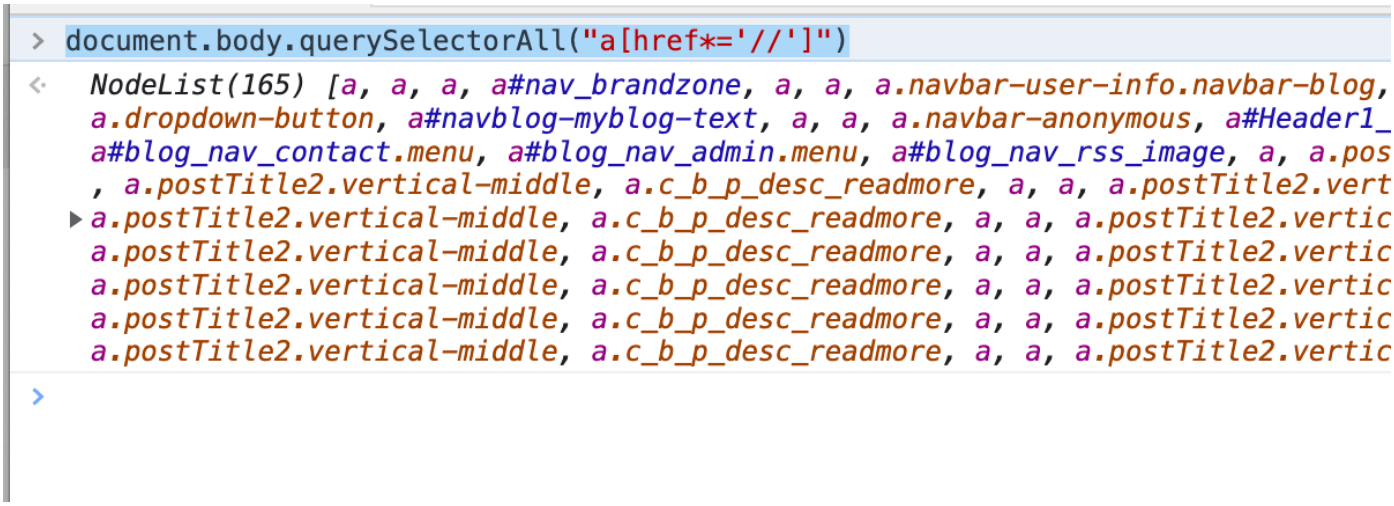
例子：下面是在页面加载前重写 `navigator.languages` 属性的例子

```
// preload.js

// 重写 `languages` 属性，使其用一个新的get方法
Object.defineProperty(navigator, "languages", {
  get: function() {
    return ["en-US", "en", "bn"];
  }
});

// 假设 preload.js 和当前的代码在同一个目录
const preloadFile = fs.readFileSync('./preload.js', 'utf8');
await page.evaluateOnNewDocument(preloadFile);
```

子链接巡检

1-获取全部子链接

对于普通的页面链接，可以通过document.body.querySelectorAll("a[href*='//']") 获取全部列表。



在puppeteer中可以通过evaluate执行JS方法获取：

```
//页面-普通超链接
const urls = await page.evaluate(
    () => Array.from(document.body.querySelectorAll('a[href*=\'//\']'), ({ href }) =>
href)
);
```

2-链接去重

由于一个页面的超链接有很多是重复的，因此巡检之前可以对超链接进行数组排重（思路：通过set）

新增方法 distinct：src/utils/Patrol.js

```
module.exports = {
    //接口响应巡检
    patrolResp:async function(response, apis) {
        const url_list = apis instanceof Array ? apis : [apis];
        if (url_list.indexOf(response.url()) !== -1 && response.status() === 200) {
            response.text().then(text => {
                console.log(response.request().resourceType() + '=' + response.url() +
",返回值=" + text);
            });
        }
    },
    //数组去重
    distinct:function (arr) {
        const newArr = [...new Set(arr)];
        return newArr;
    }
}
```

3-巡检超链接

遍历超链接，进行访问

```
        // 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);
            });
        }
```

4-巡检方法

巡检方法：获取全部链接数组，数组排重，遍历巡检

新增方法 patrolNormalHref：src/utils/Patrol.js

```
module.exports = {
    //接口响应巡检
```

```
    patrolResp:async function(response, apis) {
        const url_list = apis instanceof Array ? apis : [apis];
        if (url_list.indexOf(response.url()) !== -1 && response.status() !== 200) {
            response.text().then(text => {
                console.log(response.request().resourceType() + '=' + response.url() +
",返回值=" + text);
            });
        }
    },
    //数组去重
    distinct:function (arr) {
        const newArr = [...new Set(arr)];
        return newArr;
    },
    //页面普通链接巡检
    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);
            });
        }
    }
}
```

5-测试

修改并运行：test/automation/cases/Login/LoginBlogTest.js

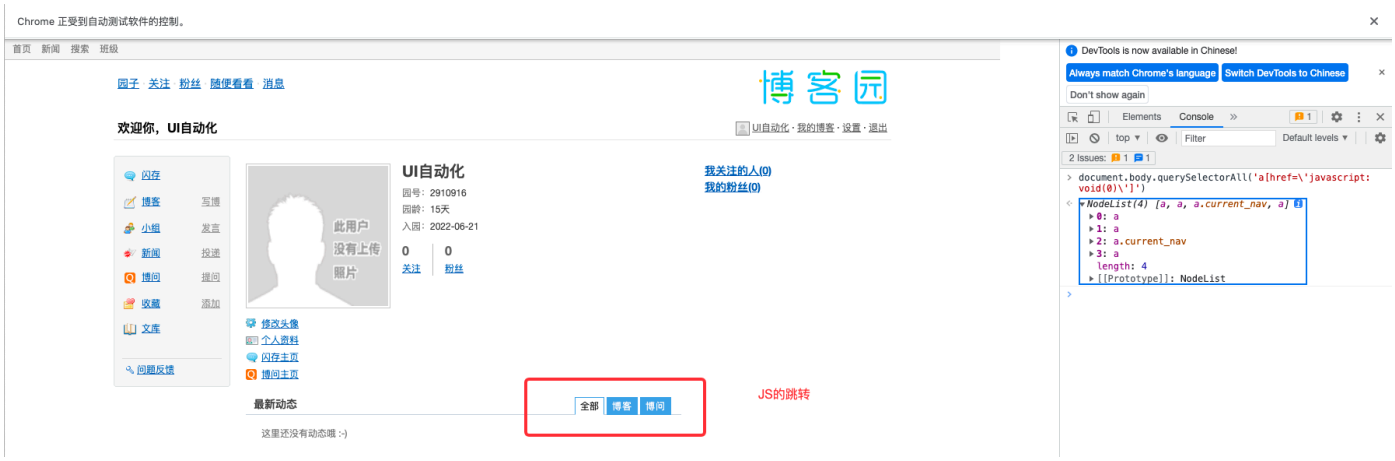
```
const {describe,it,before,after,afterEach}=require('mocha');
const Base=require('../../../Base');
const {assert}=require('chai');
const patrol = require('../../../../src/utils/Patrol');
```

```
describe('博客园',function () {
    before(Base.before);
    // after(Base.after);
    afterEach(Base.afterEach);
    this.timeout(1000000);
    it('Case1-登陆', async function () {
        //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','UI自动化');
        //4. 输入：密码
        await page.focus('#mat-input-1');
        await page.type('#mat-input-1','520bokeyuan');
        //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)})
        //接口巡检
        const api1="https://a1.cnblogs.com/group/B1";
        const api2="https://www.cnblogs.com/ajax/wechatshare/getconfig?
url=https%3A%2F%2Fw.cnblogs.com%2F";
        const list=[api1,api2];
        await page.on('response', async response => {
            await patrol.patrolResp(response,list);
        });
        //等待响应返回200
        await page.waitForResponse(response => response.status() === 200);
        await page.waitForTimeout(2000);
        //7. 进入：我的
        await page.goto("https://home.cnblogs.com/u/2910916");
        await page.waitForTimeout(2000);
        //巡检
        await patrol.patrolNormalHref(page,"https://home.cnblogs.com/u/2910916");

        //8. 用户头像，并截图
        // const avatar=await page.$('#user_profile_block > table > tbody > tr >
td:nth-child(1) > div > img');
        // await avatar.screenshot({
        //     path: './test/report/screenshot/avatar.png',
        //     type: 'png'
        // })
```

```
        //9. 验证园号=2910916
        // const name =await page.$eval('#user_profile > li:nth-child(2) > span:nth-child(2)',el => el.textContent);
        // assert.equal(name,'2910916');
    })
})
```

特殊子链接

在页面上会有一些特殊的JS跳转的按钮，例如document.body.querySelectorAll('a[href='javascript:void(0)']')，其他类型的链接类似，这些元素点击后有的会跳转到新页面，有的是在当前页面跳转。



针对这些特殊的链接，点击后，如果跳转新页面（判断当前页面的URL是不是当前页面），点击后需要返回

```
//特殊子链接
        const special_urls = await page.$$('a[href=\'javascript:void(0)\']');

        for (let element of special_urls) {
            try {
                await element.tap();
                await page.waitForTimeout(5000);
                //如果点击后，页面跳转了，需要返回
                if(url!==page.url()){
                    console.log('特殊子链接跳转到新页面'+page.url());
                    await page.goBack({ waitUntil: "networkidle0" }); // 返回到原先的检测页面
                }
            } catch (e) {
                console.log(e);
                break;
            }
        }
```

patrolSpecialHref方法兼容特殊子链接：

```
//页面特殊链接巡检
    patrolSpecialHref:async function(page,url) {
        //指定访问页面
        await page.goto(url);
        //特殊子链接
        const special_urls = await page.$$('a[href=\'javascript:void(0)\']');
        console.log('子链接统计:'+url);
        console.log('特殊子链接:'+special_urls.length);
        for (let element of special_urls) {
            try {
                await element.tap();
                await page.waitForTimeout(5000);
                //如果点击后，页面跳转了，需要返回
                if(url!==page.url()){
                    console.log('特殊子链接跳转到新页面'+page.url());
                    await page.goBack({ waitUntil: "networkidle0" }); // 返回到原先的检测
页面
                }
            } catch (e) {
                console.log(e);
                break;
            }
        }
    }
```

完整：src/utils/Patrol.js

```
module.exports = {
    //接口响应巡检
    patrolResp:async function(response, apis) {
        const url_list = apis instanceof Array ? apis : [apis];
        if (url_list.indexOf(response.url()) !== -1 && response.status() !== 200) {
            response.text().then(text => {
                console.log(response.request().resourceType() + '=' + response.url() +
",返回值=" + text);
            });
        }
    },
    //数组去重
    distinct:function (arr) {
        const newArr = [...new Set(arr)];
        return newArr;
    },
    //页面普通链接巡检
    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);
            });
        }
    },
    //页面特殊链接巡检
    patrolSpecialHref:async function(page,url) {
        //指定访问页面
        await page.goto(url);
        //特殊子链接
        const special_urls = await page.$$('a[href=\'javascript:void(0)\']');
        console.log('子链接统计:'+url);
        console.log('特殊子链接:'+special_urls.length);
        for (let element of special_urls) {
            try {
                await element.tap();
                await page.waitForTimeout(5000);
                //如果点击后，页面跳转了，需要返回
                if(url!==page.url()){
                    console.log('特殊子链接跳转到新页面'+page.url());
                    await page.goBack({ waitUntil: "networkidle0" }); // 返回到原先的检测
页面
                }
            } catch (e) {
                console.log(e);
                break;
            }
        }
    }
}
```

修改：test/automation/cases/Login/LoginBlogTest.js

```
const {describe,it,before,after,afterEach}=require('mocha');
const Base=require('../../../Base');
const {assert}=require('chai');
const patrol = require('../../../../src/utils/Patrol');

describe('博客园',function () {
    before(Base.before);
    // after(Base.after);
    afterEach(Base.afterEach);
    this.timeout(1000000);
    it('Case1-登陆', async function () {
        //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','UI自动化');
        //4. 输入：密码
        await page.focus('#mat-input-1');
        await page.type('#mat-input-1','520bokeyuan');
        //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)})
        //接口巡检
        const api1="https://a1.cnblogs.com/group/B1";
        const api2="https://www.cnblogs.com/ajax/wechatshare/getconfig?
url=https%3A%2F%2Fw.cnblogs.com%2F";
        const list=[api1,api2];
        await page.on('response', async response => {
            await patrol.patrolResp(response,list);
        });
        //等待响应返回200
        await page.waitForResponse(response => response.status() === 200);
        await page.waitForTimeout(2000);
        //7. 进入：我的
        await page.goto("https://home.cnblogs.com/u/2910916");
        await page.waitForTimeout(2000);
        //巡检
        await patrol.patrolNormalHref(page,"https://home.cnblogs.com/u/2910916");
        await patrol.patrolSpecialHref(page,"https://home.cnblogs.com/u/2910916");

        //8. 用户头像，并截图
```

```
        // const avatar=await page.$('#user_profile_block > table > tbody > tr > td:nth-child(1) > div > img');
        // await avatar.screenshot({
        //     path: './test/report/screenshot/avatar.png',
        //     type: 'png'
        // })
        //9. 验证园号=2910916
        // const name =await page.$eval('#user_profile > li:nth-child(2) > span:nth-child(2)',el => el.textContent);
        // assert.equal(name,'2910916');
    })
})
```

说明：此时的巡检代码看起来还比较复杂，后续引入cookie登陆后，将优化巡检代码