

本文主要讲解page.on的事件监听处理，未后续进行接口巡检做准备

page.on事件监听

Events事件

puppeteer提供了下列类型的浏览器事件监听：

- [page.on\('close'\)](#)
- [page.on\('console'\)](#)
- [page.on\('dialog'\)](#)
- [page.on\('domcontentloaded'\)](#)
- [page.on\('error'\)](#)
- [page.on\('frameattached'\)](#)
- [page.on\('framedetached'\)](#)
- [page.on\('framenavigated'\)](#)
- [page.on\('load'\)](#)
- [page.on\('metrics'\)](#)
- [page.on\('pageerror'\)](#)
- [page.on\('request'\)](#)
- [page.on\('requestfailed'\)](#)
- [page.on\('requestfinished'\)](#)
- [page.on\('response'\)](#)
- [page.on\('workercreated'\)](#)
- [page.on\('workerdestroyed'\)](#)

什么是浏览器事件？

浏览器事件是指浏览器的生命周期中发生的一系列操作，例如加载，关闭，console输出信息。puppeteer封装了常见的浏览器事件监听，可以方便的在浏览器的指定事件发生时进行自定义操作。

下面介绍一些最常见的事件监听的应用：

事件	触发
page.on('load')	当页面的 load 事件被触发时触发。
page.on('console')	当页面js代码调用了 console 的某个方法，比如 console.log，或者 console.dir 的时候触发。
page.on('pageerror')	当发生页面js代码没有捕获的异常时触发。
page.on('requestfinished')	当页面的某个/全部请求成功完成时触发。
page.on('response')	当页面的某个请求接收到对应的 response 时触发。
page.on('request')	

1-page.on('load')

场景：监听页面的load事件触发

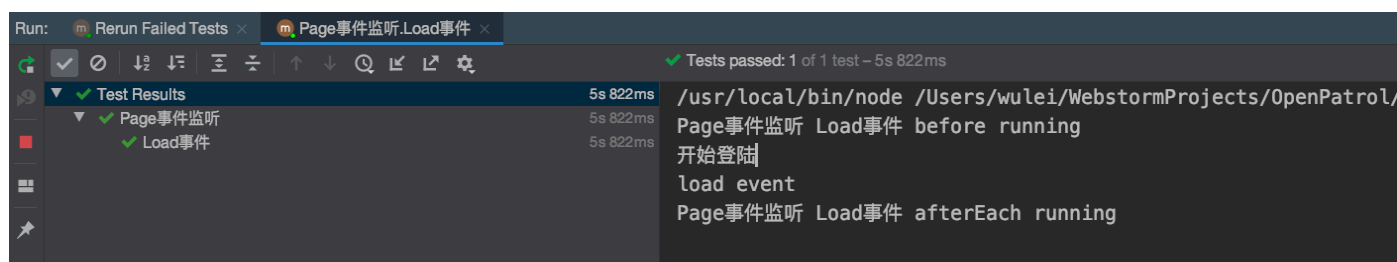
应用：暂无

新建：test/automation/cases/Demo/PageOnTest.js

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

describe('Page事件监听',function () {
  before(Base.before);
  // after(Base.after);
  afterEach(Base.afterEach);
  this.timeout(40000);
  it('Load事件', async function () {
    //load事件
    await page.on('load',async load=>{
      console.log('load event');
    });
    console.log('开始登陆')
    //1.博客园
    await page.goto("https://account.cnblogs.com/signin");
    //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');
  })
})
```

运行后发现先输出开始登陆，等浏览器load后，打印了load日志



2-page.on('console')

场景：监听页面的load事件触发

应用：页面巡检

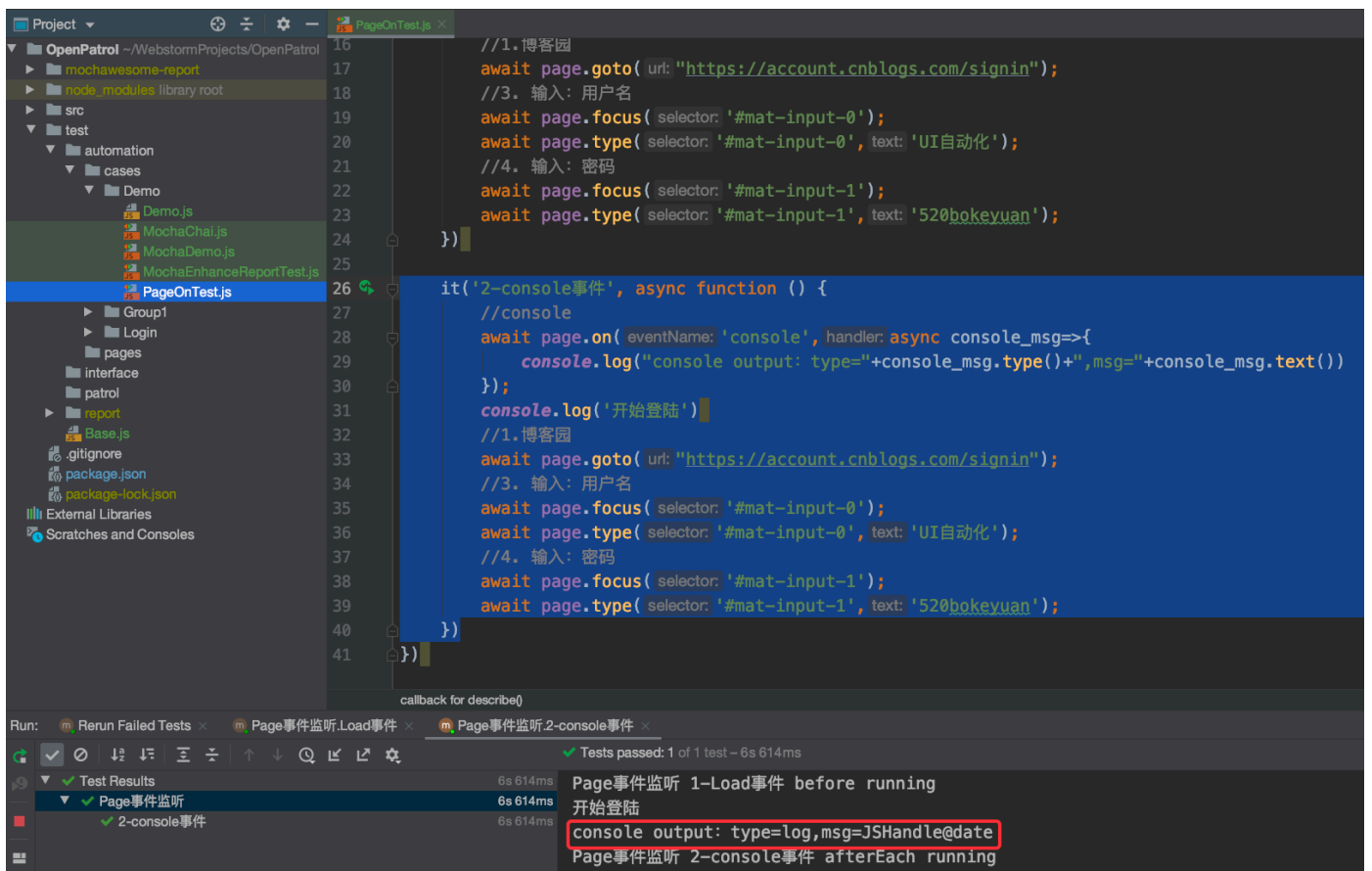
说明：浏览器console很多错误都在这里展示，页面巡检需要重点关注的一部分

新增it：test/automation/cases/Demo/PageOnTest.js

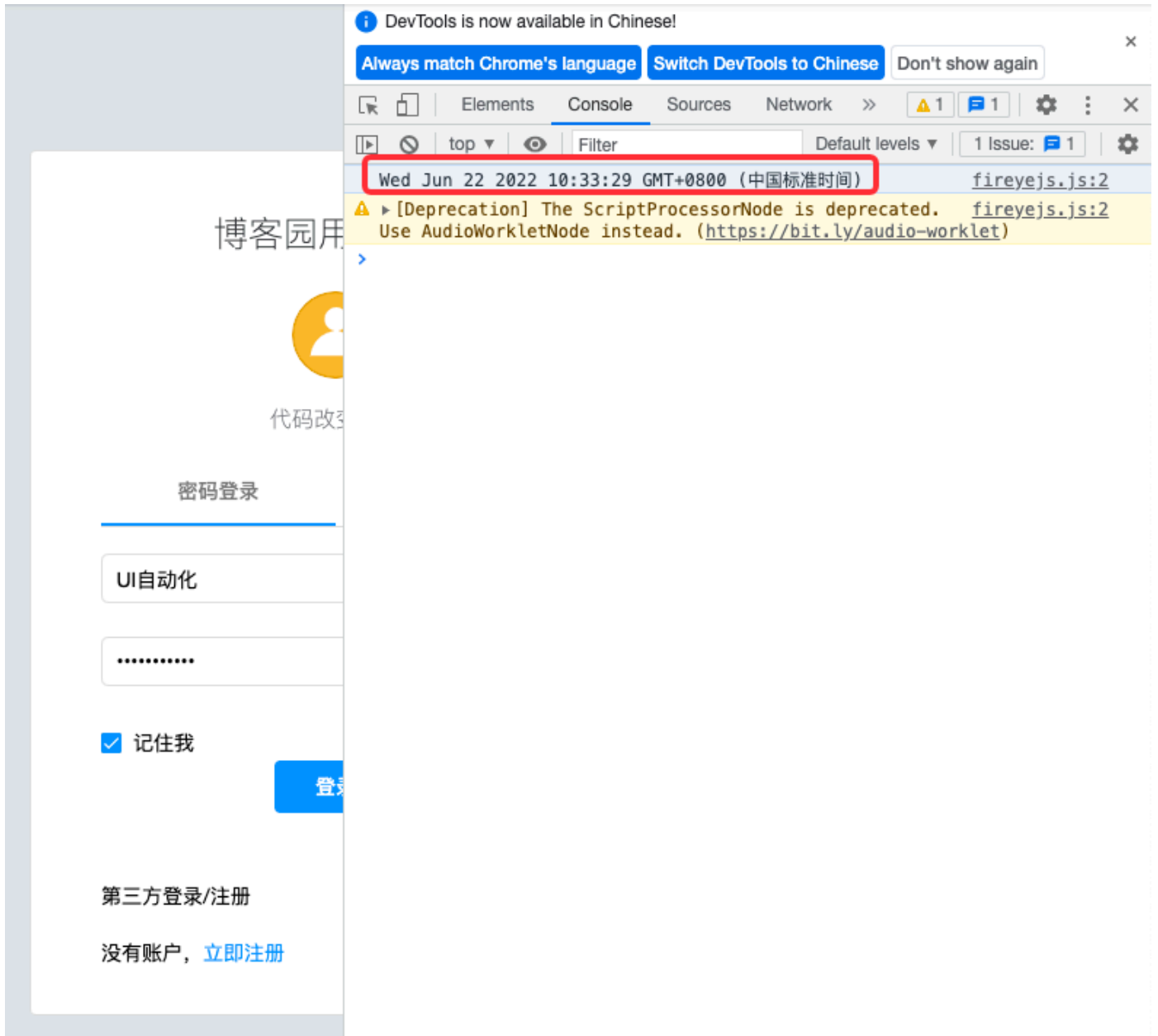
```

it('2-console事件', async function () {
  //console
  await page.on('console', async console_msg=>{
    console.log("console output:
type="+console_msg.type()+" ,msg="+console_msg.text())
  });
  console.log('开始登陆')
  //1. 博客园
  await page.goto("https://account.cnblogs.com/signin");
  //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');
})

```



可以看到console有输出当前时间:



3-page.on('pageerror')

场景：当发生页面js代码没有捕获的异常时触发。巡检page页面异常

应用：页面巡检

说明：**pageerror**页面巡检最重要的一部分，代表页面已经出现了异常

新增it：test/automation/cases/Demo/PageOnTest.js

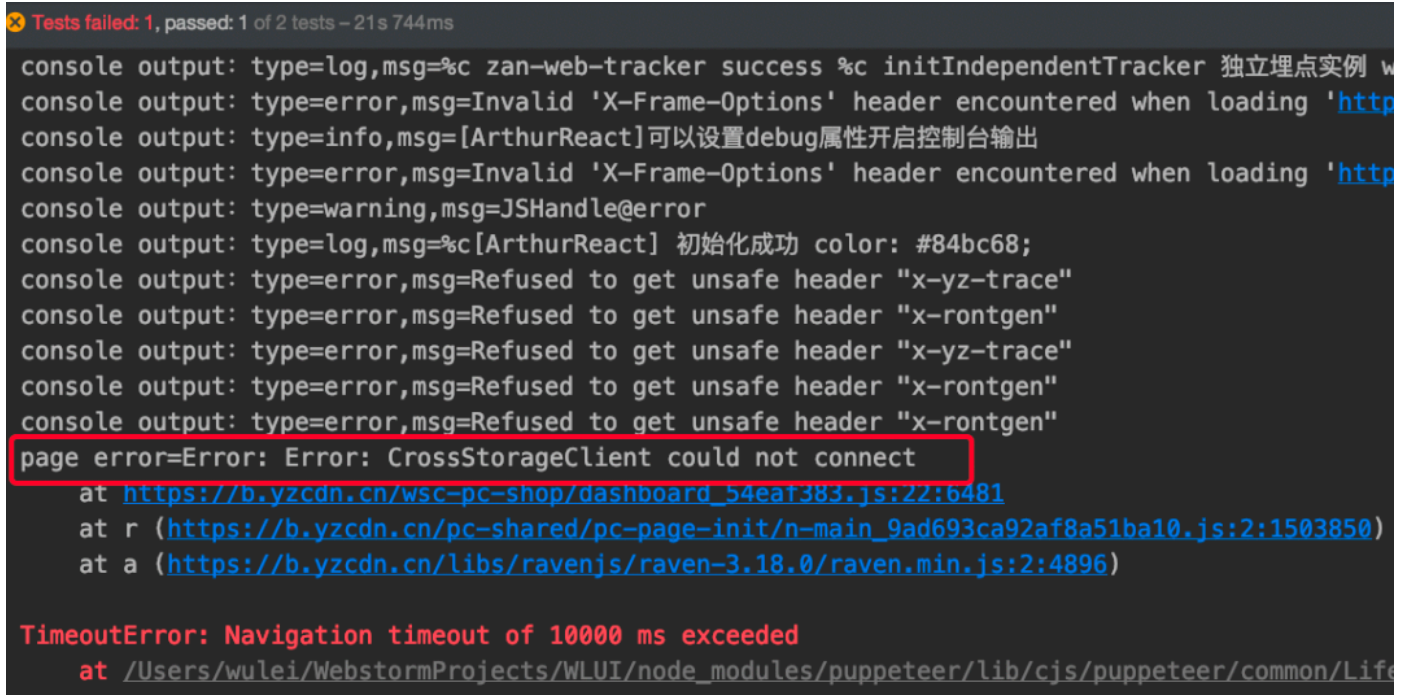
```
it('3-pageerror事件', async function () {  
  //pageerror  
  await page.on('pageerror', async error=>{  
    console.log("page error"+error);  
  });  
  console.log('开始登陆')  
  //1.博客园  
  await page.goto("https://account.cnblogs.com/signin");  
});
```

```

//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');
})

```

由于pageerror是偶发事件，这里找一个截图



```

Tests failed: 1, passed: 1 of 2 tests - 21s 744ms

console output: type=log,msg=%c zan-web-tracker success %c initIndependentTracker 独立埋点实例 w
console output: type=error,msg=Invalid 'X-Frame-Options' header encountered when loading 'http
console output: type=info,msg=[ArthurReact]可以设置debug属性开启控制台输出
console output: type=error,msg=Invalid 'X-Frame-Options' header encountered when loading 'http
console output: type=warning,msg=JSHandle@error
console output: type=log,msg=%c[ArthurReact] 初始化成功 color: #84bc68;
console output: type=error,msg=Refused to get unsafe header "x-yz-trace"
console output: type=error,msg=Refused to get unsafe header "x-rontgen"
console output: type=error,msg=Refused to get unsafe header "x-yz-trace"
console output: type=error,msg=Refused to get unsafe header "x-rontgen"
console output: type=error,msg=Refused to get unsafe header "x-rontgen"
page error=Error: Error: CrossStorageClient could not connect
  at https://b.yzcdn.cn/wsc-pc-shop/dashboard_54eat383.js:22:6481
  at r (https://b.yzcdn.cn/pc-shared/pc-page-init/n-main_9ad693ca92af8a51ba10.js:2:1503850)
  at a (https://b.yzcdn.cn/libs/ravenjs/raven-3.18.0/raven.min.js:2:4896)

TimeoutError: Navigation timeout of 10000 ms exceeded
  at /Users/wulei/WebstormProjects/WLUI/node_modules/puppeteer/lib/cjs/puppeteer/common/Life

```

4-page.on('requestfinished')

场景：当页面发生请求时触发，请求的类型可以是图片，xhr，字体等

应用：页面巡检，接口巡检

说明：**requestfinished**代表全部浏览器开发者工具**Network**里的全部请求

新增it: test/automation/cases/Demo/PageOnTest.js

```

it('4-requestfinished事件', async function () {
  //requestfinished
  await page.on('requestfinished', async request => {
    const res = request.response();
    if (res.status()===200) {
      console.log("请求="+request.url()+"，请求数据="+request.postData()+"，返回状态="+res.status());
      res.text().then(text => {
        console.log("返回体="+text);
      });
    }
  });
  //1.博客园
  await page.goto("https://account.cnblogs.com/signin");

```

requestfinished事件非常的多，一般的用法是进行x h r类型的判断

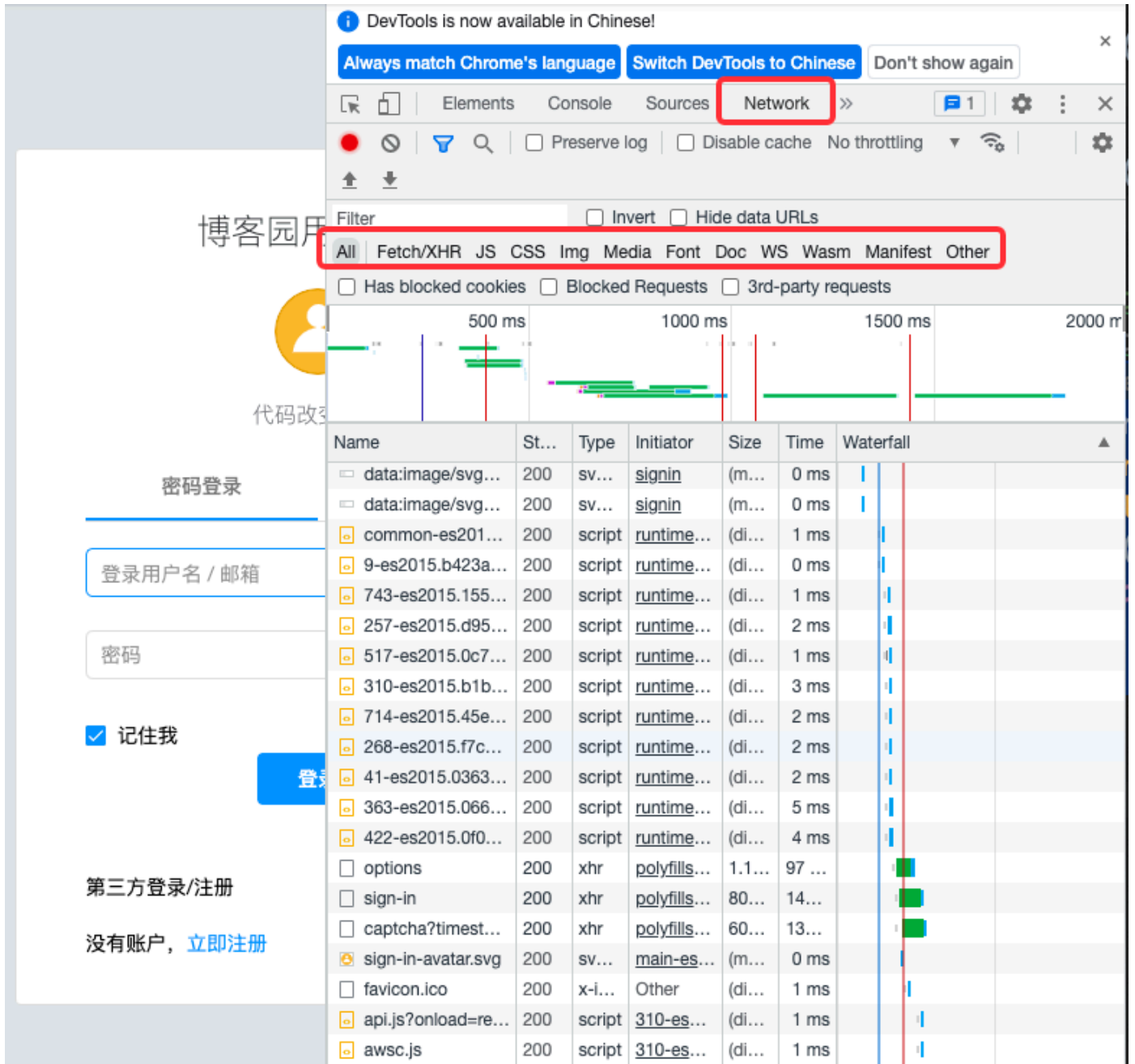
```

await page.on('requestfinished', async request => {
  const res = request.response();
  if (request.resourceType() == "xhr" && res.status()!==200) {
    console.log("接口="+request.url()+"，请求数据="+request.postData()+"，返回状态="+res.status());
    res.text().then(text => {
      console.log("返回体="+text);
    });
  }
});

```

resourceType的类型：

('Document' | 'Stylesheet' | 'Image' | 'Media' | 'Font' | 'Script' | 'TextTrack' | 'XHR' | 'Fetch' | 'EventSource' | 'WebSocket' | 'Manifest' | 'SignedExchange' | 'Ping' | 'CSPViolationReport' | 'Preflight' | 'Other');



5-page.on('response')

场景：当接口有返回时监听，可以获取接口的返回和状态

应用：接口巡检

说明：**response**代表指定请求的响应监听，接口巡检的底层

新增it: test/automation/cases/Demo/PageOnTest.js

```
it('5-response事件', 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)})
    //response
    await page.on('response', async res => {

        if(res.url().indexOf("https://www.cnblogs.com/aggsite/allsitecategories")!==-1
&&res.status() === 200){
            res.text().then(text => {
                console.log("返回体="+text);
            });
        }
    });
    //等待响应返回200
    await page.waitForResponse(response => response.status() === 200);
    await page.waitForTimeout(2000);
    await page.goto("https://home.cnblogs.com/u/2910916");
})

```

可以看到制定接口的返回信息:



6-page.on('request')

场景: 对指定请求进行监听, 可以修改请求或者mock请求返回

应用: 请求拦截, Mock

说明: **request**代表指定请求的操作, 多用于请求拦截或者接口Mock

- 监听的请求的类型resourceType的类型:

('Document' | 'Stylesheet' | 'Image' | 'Media' | 'Font' | 'Script' | 'TextTrack' | 'XHR' | 'Fetch' | 'EventSource' | 'WebSocket' | 'Manifest' | 'SignedExchange' | 'Ping' | 'CSPViolationReport' | 'Preflight' | 'Other');

- await page.setRequestInterception(true) 开启拦截, 必须先开启;

- `await req.respond()` 返回一个自定义响应（Mock返回）；
- `await req.abort()` 继续请求；
- `await req.continue()` 继续请求；
- `await req.respond()`和`await req.continue()` 不能同时调用；
- 使用`await req.respond()` 拦截返回值注意跨域；
- 使用`await req.respond()` 拦截返回值注意返回json或字符串

1-请求拦截

⚠注意：

1-request监听之前必须先开启拦截：`page.setRequestInterception(true)`

2-其他请求一定要放行：`request.continue()`

新增it: test/automation/cases/Demo/PageOnTest.js

```
it('6-request事件-请求拦截', 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)})
    //request: 拦截图片请求
    await page.setRequestInterception(true);//开启请求拦截
    await page.on('request',async request => {
        if(request.resourceType()==='image'){
            return request.abort();//图片请求，直接丢弃
        }else{
            return request.continue();//其他请求，继续
        }
    });
    //等待响应返回200
    await page.waitForResponse(response => response.status() === 200);
```

```

await page.waitForTimeout(2000);
await page.goto("https://home.cnblogs.com/u/2910916");
})

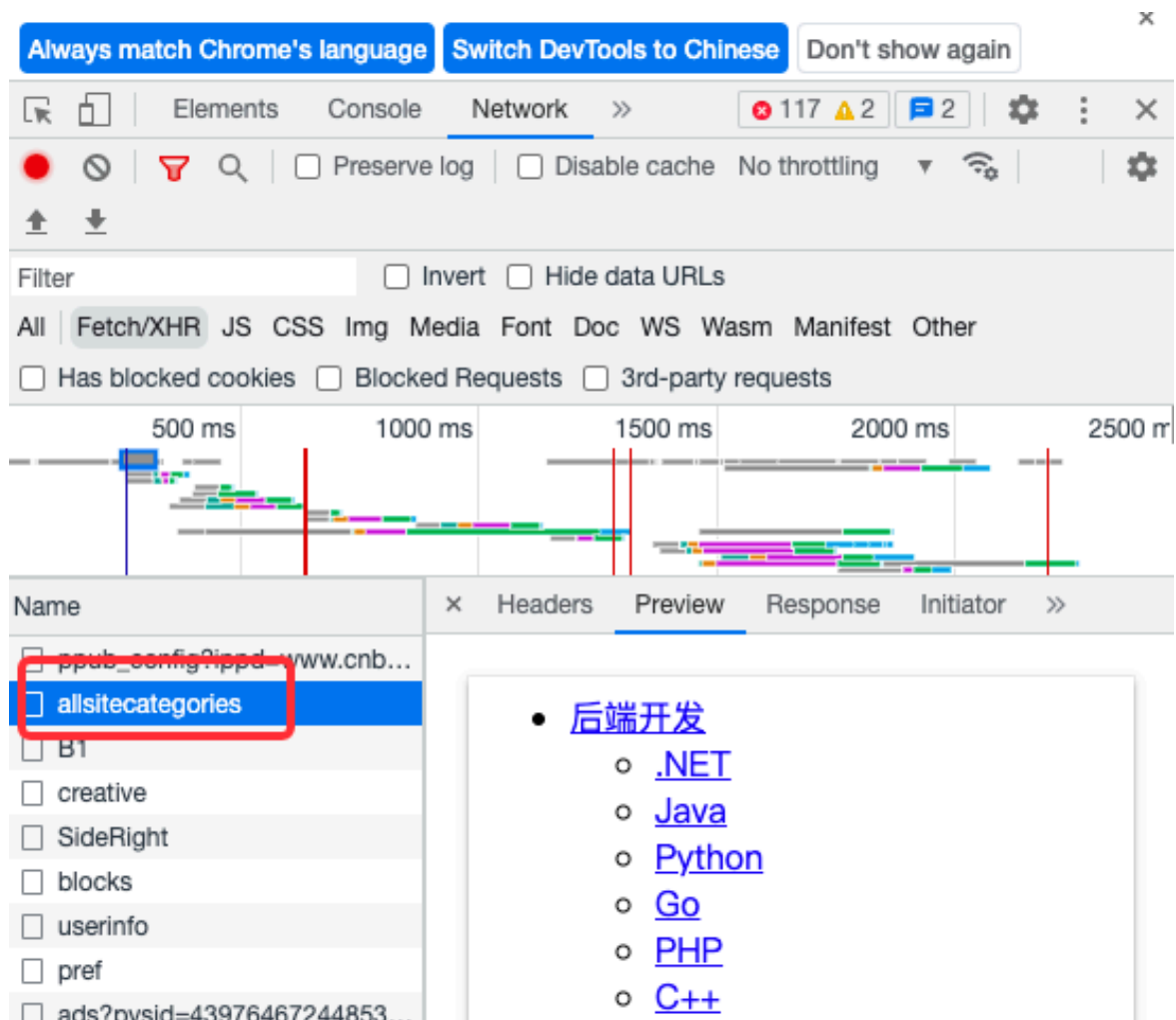
```

可以看到图片请求被拦截：



2-Mock接口返回

在5-page.on('response')中我们知道 (<https://www.cnblogs.com/aggsite/allsitecategories>) 接口返回的是一组列表



- ☐ collect?v=1&_v=j96&a=4910...
- ☐ getconfig?url=https%3A%2...
- ☐ postIds
- ☐ sodar?sv=200&tid=gpt&tv=2...
- ☐ B28010219.338308031;dc_tr...
- ☐ advview?ai=C69tqPbuyYuGR...
- ☐ B28010219.338308031;dc_p...

- [Ruby](#)
- [Swift](#)
- [C语言](#)
- [Erlang](#)
- [Delphi](#)
- [Scala](#)
- [R语言](#)
- [Verilog](#)
- [Dart](#)
- [Rust](#)
- [其他语言](#)

- [软件设计](#)

- [架构设计](#)
- [面向对象](#)
- [设计模式](#)
- [领域驱动设计](#)

- [前端开发](#)

- [Html/Css](#)
- [JavaScript](#)
- [jQuery](#)
- [HTML5](#)
- [Angular](#)
- [React](#)
- [Vue](#)

这里我们可以将这个接口的返回改成自定义信息

```
{"code":0,"msg":""}
```

新增it: test/automation/cases/Demo/PageOnTest.js

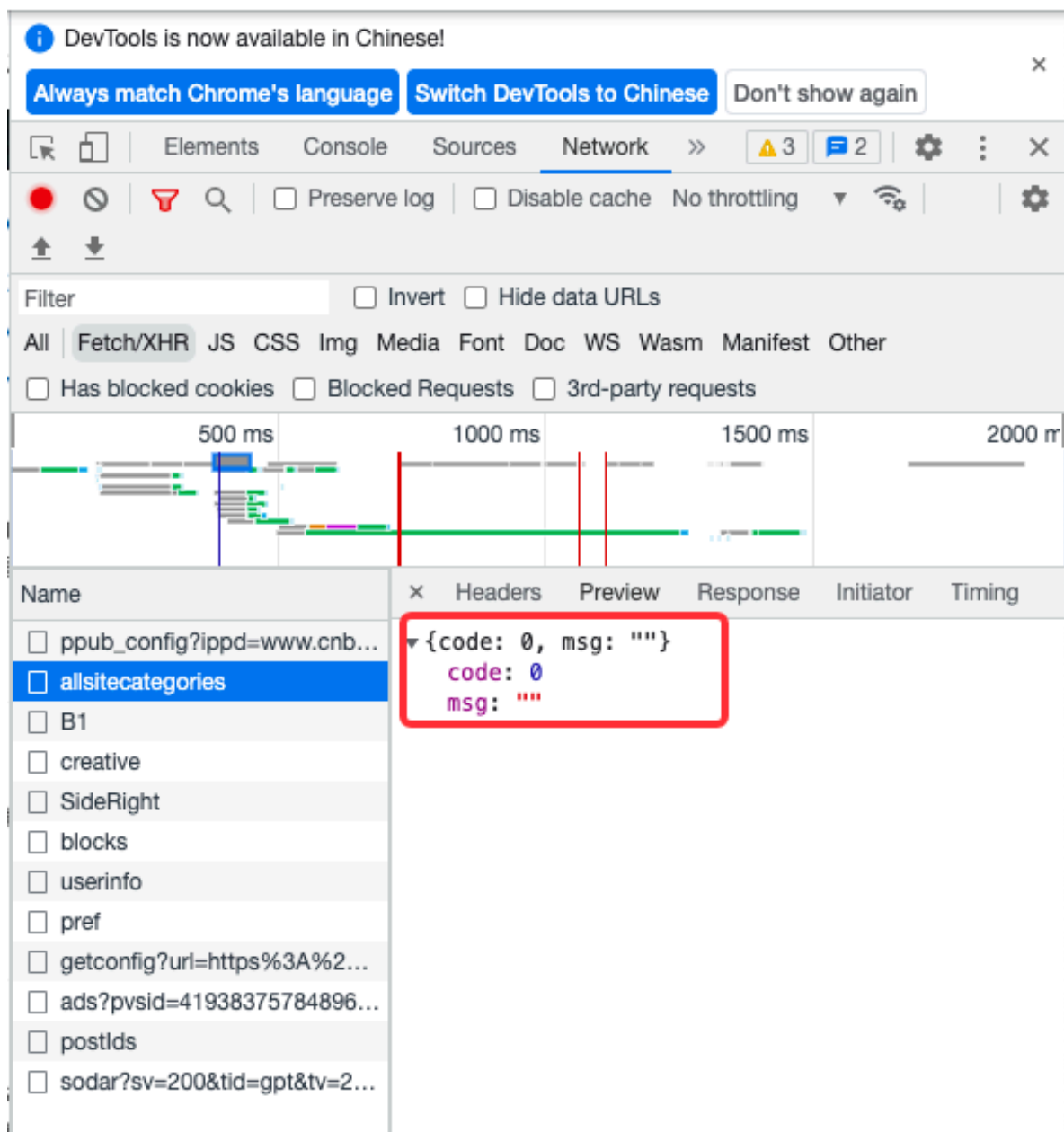
```
it('6-request事件-Mock', async function () {
  //1. 登陆: 博客园
  await page.goto("https://account.cnblogs.com/signin");
  //2. 点击: 密码登陆
  await page.waitForSelector('#mat-tab-label-0-0 > div',{timeout:2*1000}).then(ele=>
{ele.click()}).catch(e=>{console.log(e)})
  //3. 输入: 用户名
  await page.focus('#mat-input-0');
  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)})
//request
await page.setRequestInterception(true);//开启请求拦截
await page.on('request',async request => {
  if(request.url()=== "https://www.cnblogs.com/aggsite/allsitecategories"){
    await request.respond({//自定义返回
      status: 200,
      headers: {
        'Access-Control-Allow-Origin': '*',
      },
      contentType: 'application/json; charset=utf-8',
      body: JSON.stringify({"code":0,"msg":""})
    });
  }else{
    return request.continue();//其他请求, 继续
  }
});
//等待响应返回200
await page.waitForResponse(response => response.status() === 200);
await page.waitForTimeout(2000);
})

```

查看请求:



7-全局监听

对于一些重要的事件监听可以进行全局监听（例如：console，pageerror），因此可以在初始化页面时就进行监听。

修改：src/system/init.js 的createPage方法，加入监听事件：

```
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;
//监听load事件
page.on('load', async load => {
    console.log("load started");
});
//监听console事件:可以按照类型进行过滤
page.on('console', async console_msg => {
    if(console_msg.type()==='error'){
        console.log("console error output msg="+console_msg.text());
    }
});
//监听page error事件
page.on('pageerror', async error_msg => {
    console.log("page error="+error_msg);
});
//监听request finished事件，对事件进行输出
page.on('requestfinished', async request => {
    const res = request.response();
    if (request.resourceType() == "xhr" && res.status()!==200) {
        console.log("接口="+request.url()+"，请求数据="+request.postData()+"，
返回状态="+res.status());
        res.text().then(text => {
            console.log("返回体="+text);
        });
    }
});
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;

```

```

//监听load事件
page.on('load', async load => {
  console.log("load started");
});
//监听console事件:可以按照类型进行过滤
page.on('console', async console_msg => {
  if(console_msg.type()=='error'){
    console.log("console error output msg="+console_msg.text());
  }
});
//监听page error事件
page.on('pageerror', async error_msg => {
  console.log("page error="+error_msg);
});
//监听request finished事件, 对事件进行输出
page.on('requestfinished', async request => {
  const res = request.response();
  if (request.resourceType() == "xhr" && res.status()!==200) {
    console.log("接口="+request.url()+"", 请求数据="+request.postData()+"",
返回状态="+res.status());
    res.text().then(text => {
      console.log("返回体="+text);
    });
  }
});

return page;
}
}

```

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

Project

OpenPatrol ~ /WebstormProjects/OpenPatrol

node_modules

src

test

automation

cases

Demo

Group1

Login

LoginBlogTest.js

LoginTest.js

pages

interface

patrol

report

Base.js

.gitignore

package.json

package-lock.json

External Libraries

Scratches and Consoles

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

```
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");
  })
})
```

Run: LoginBase 博客园.Case1-登陆

Test Results

3s 626ms

2s 337ms

1s 289ms

3s 626ms

1s 289ms

2s 337ms

Tests passed: 2 of 2 tests - 3s 626ms

/usr/local/bin/node /Users/wulei/WebstormProjects/OpenPatrol/node_modules/mocha/bin/_mocha --ui bdd --reporter /Applicat:

LoginBase Case1-打开百度 before running

LoginBase Case1-打开百度 after running

接口=https://www.baidu.com/s?ie=utf-8&mod=11&isbd=1&id=339ACC3FD7526681&ie=utf-8&f=8&rsv_bp=1&rsv_idx=1&tn=baidu&wd=pup

接口=https://www.baidu.com/s?ie=utf-8&mod=11&isbd=1&id=339ACC3FD7526681&ie=utf-8&f=8&rsv_bp=1&rsv_idx=1&tn=baidu&wd=pup

接口=https://www.baidu.com/s?ie=utf-8&mod=11&isbd=1&id=339ACC3FD7526681&ie=utf-8&f=8&rsv_bp=1&rsv_idx=1&tn=baidu&wd=pup

接口=https://www.baidu.com/s?ie=utf-8&mod=11&isbd=1&id=339ACC3FD7526681&ie=utf-8&f=8&rsv_bp=1&rsv_idx=1&tn=baidu&wd=pup

console error output msg=Access to XMLHttpRequest at 'https://wappass.baidu.com/static/captcha/tuxing.html?&logid=8631434

console error output msg=Failed to load resource: net::ERR_FAILED

接口=https://www.baidu.com/s?ie=utf-8&mod=11&isbd=1&id=339ACC3FD7526681&ie=utf-8&f=8&rsv_bp=1&rsv_idx=1&tn=baidu&wd=pup

console error output msg=Access to XMLHttpRequest at 'https://wappass.baidu.com/static/captcha/tuxing.html?&logid=7966191

console error output msg=Failed to load resource: net::ERR_FAILED

LoginBase Case2-搜索 afterEach running

LoginBase Case2-搜索 after running