本文主要介绍在UI自动化测试与前端接口测试组合代码的编写

为什么要组合?

UI自动化测试的问题：只进行流程和页面的校验，缺乏打开页面时的后端接口校验

前端接口测试的问题：只进行了接口级别的响应校验，缺乏与页面元素展示的对比

因此两者在结合时可以互相弥补彼此的劣势，并且可以让测试同学熟悉完整的前后端交互细节

下面通过一个具体的例子来进行分析

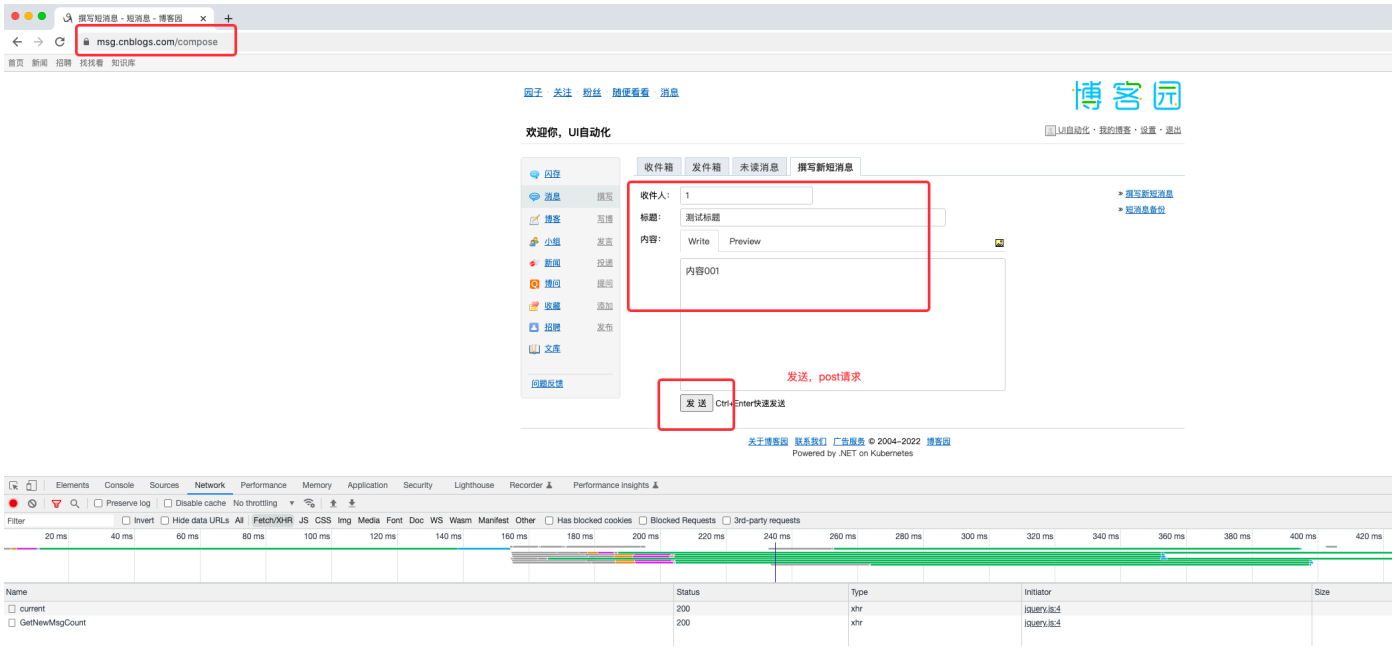
场景：撰写新短消息

1-场景流程分析

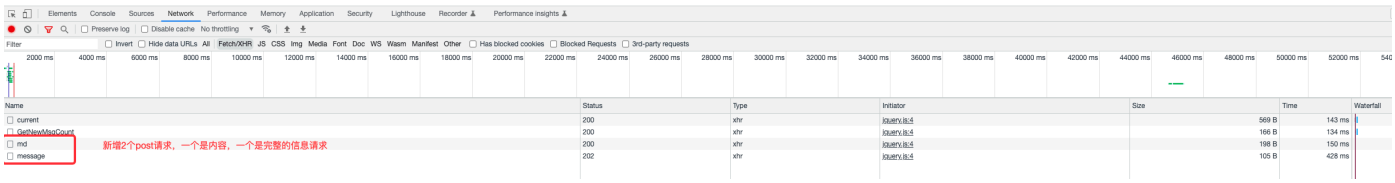
1-登陆：略

2-进入【短消息页面】（后端发送2个get请求），点击【撰写新短消息】按钮

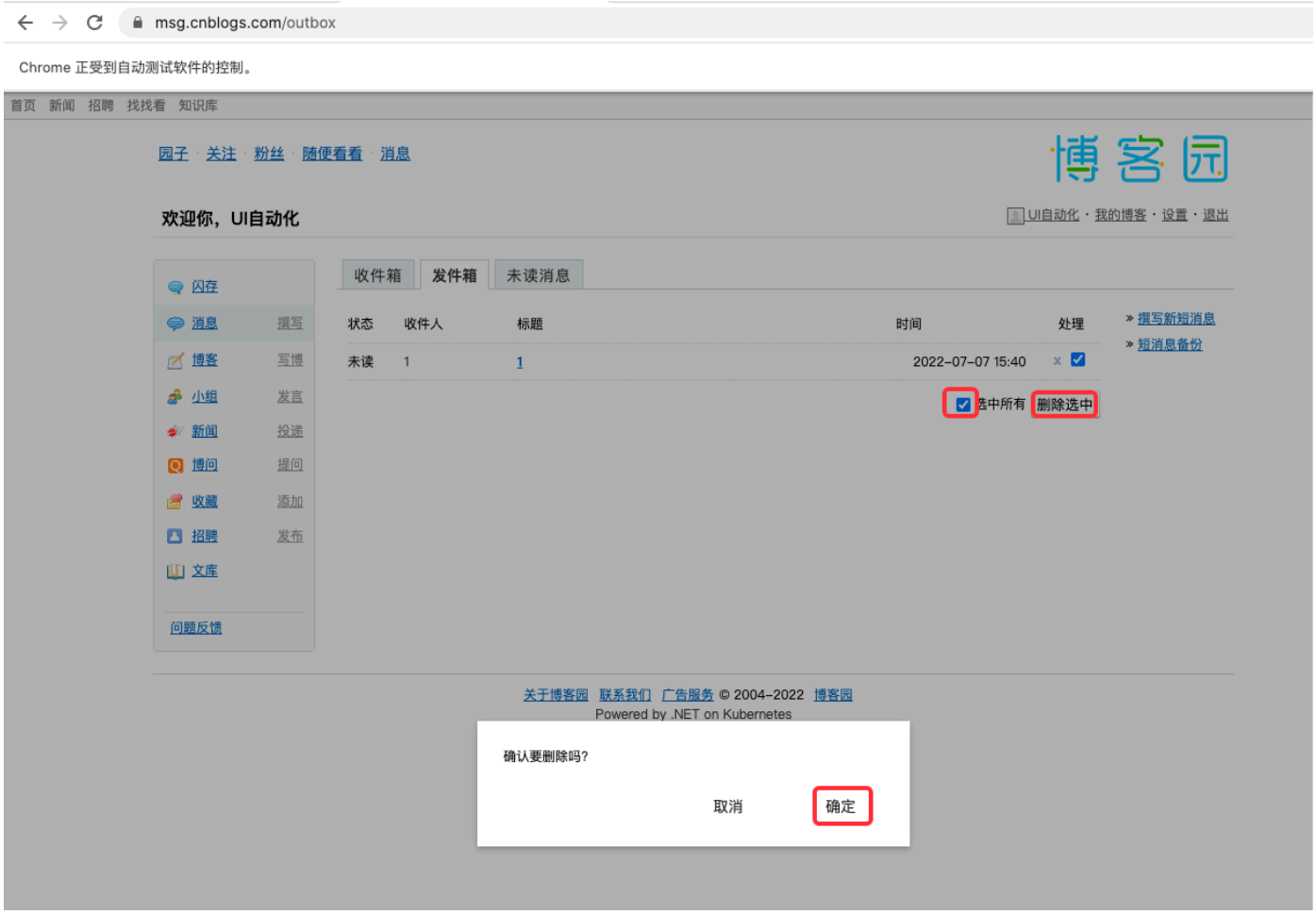
3-在【撰写页面】，输入短消息信息，点击【发送】按钮



4-发送2个post请求，文字提示发送成功，出现收信箱按钮



5-【发件箱页面】,查看发送的消息，选择所有，确认删除，提示删除成功


msg.cnblogs.com/outbox
Chrome 正受到自动测试软件的控制。
首页 新闻 招聘 找找看 知识库
园子 · 关注 · 粉丝 · 随便看看 · 消息
博客园
欢迎你，UI自动化
UI自动化 · 我的博客 · 设置 · 退出
闪存
消息 撰写
博客 写博
小组 发言
新闻 投递
博问 提问
收藏 添加
招聘 发布
文库
问题反馈
收件箱 发件箱 未读消息
状态 收件人 标题 时间 处理
» 撰写新短消息
» 短消息备份
未读 1 1 2022-07-07 15:40
选中所有 删除选中
关于博客园 联系我们 广告服务 © 2004-2022 博客园
Powered by .NET on Kubernetes
确认要删除吗?
取消 确定


园子 · 关注 · 粉丝 · 随便看看 · 消息
博客园
欢迎你，UI自动化
UI自动化 · 我的博客 · 设置 · 退出
闪存
消息 撰写
博客 写博
小组 发言
新闻 投递
博问 提问
收藏 添加
招聘 发布
文库
问题反馈
收件箱 发件箱 未读消息
状态 收件人 标题 时间 处理
» 撰写新短消息
» 短消息备份
删除成功
选中所有 删除成功
关于博客园 联系我们 广告服务 © 2004-2022 博客园
Powered by .NET on Kubernetes

2-场景抽象

上述过程涉及3个页面，每个页面含有若干请求，并且每个页面都有请求

收件箱页面

URL：https://msg.cnblogs.com/inbox

按钮：撰写短消息按钮

请求：

1. get https://msg.cnblogs.com/api/user/current
2. get https://msg.cnblogs.com/api/message/GetNewMsgCount

撰写短消息页面

URL：https://msg.cnblogs.com/compose

按钮：

- 输入框：收件人
- 输入框：标题
- 输入框：内容
- 按钮：发送
- 文字：发送成功
- 按钮：返回收信箱

请求：

1. Post https://msg.cnblogs.com/api/common/md
2. Post https://msg.cnblogs.com/api/message

发件箱页面

URL：https://msg.cnblogs.com/outbox

按钮：

- 单选框：选择所有
- 按钮：删除选中
- 按钮：确认删除
- 文字：删除成功

请求：

1. Post https://msg.cnblogs.com/api/message/batchdelete?ids=3711531

注意此时ID是会变动的

3-PO模式页面

在：test/automation/pages/Blog

新增2个页面配置：InboxPage.js，ComposePage.js

InboxPage

```
const Element = require('../Element');
const newPuppeteer = require('../../../../src/utils/NewPuppeteer');
const InboxPage = {
    //url
    'inbox_url': "https://msg.cnblogs.com/inbox",
    //元素封装
    'button_compose': Element('#right_sidebar > div > ul > li:nth-child(1) >
a[href=\'/compose\']', '撰写短消息'),
    //api
    'api_current': "https://msg.cnblogs.com/api/user/current",
    'api_getMgsCount': "https://msg.cnblogs.com/api/message/GetNewMsgCount",

    //常见方法
    /**
     * 跳转撰写短消息
     */
    gotoComposeMsg: async function () {
        //1. 点击撰写短消息
        await newPuppeteer.click(this.button_compose);

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

};
module.exports=InboxPage;
```

ComposePage

```
const Element = require('../Element');
const newPuppeteer = require('../../../../src/utils/NewPuppeteer');
const ComposePage = {
    //url
    'compose_url': "https://msg.cnblogs.com/compose",

    //元素封装
    'input_receiver': Element('#txtIncept', '收件人'),
    'input_title': Element('#txtTitle', '标题'),
    'input_content': Element('#txtContent', '内容'),
    'button_send': Element('#btnSend', '收件人'),
```

```
    'text_success': Element('#span_msg > div > span', '发送成功'),
    'button_inbox': Element('#span_msg > div > a[href=\'/inbox\']', '返回收信箱'),
    //api
    'api_md': "https://msg.cnblogs.com/api/common/md",
    'api_getMgsCount': "https://msg.cnblogs.com/api/message",

    //常见方法
    /**
     * 撰写短消息
     */
    composeMsg: async function (receiver,title) {
        //输入信息并跳转
        await newPuppeteer.focusAndType(this.input_receiver,receiver);
        await newPuppeteer.focusAndType(this.input_title,title);
        await newPuppeteer.focusAndType(this.input_content,'content'+Date.now());
        await newPuppeteer.click(this.button_send);

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

};
module.exports=ComposePage;
```

OutboxPage

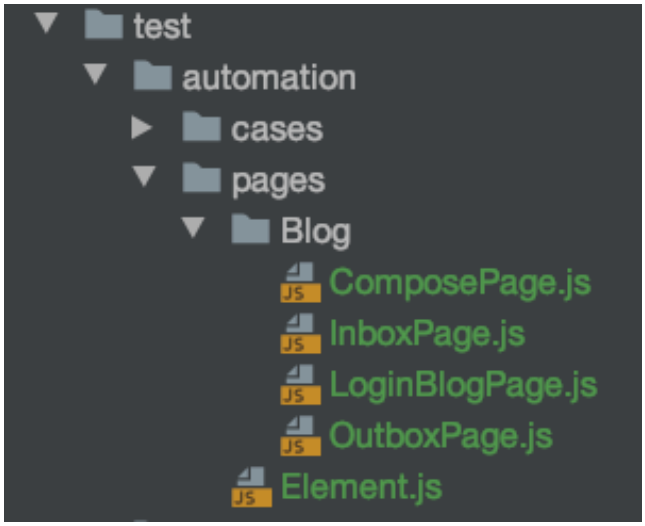
```
const Element = require('../Element');
const newPuppeteer = require('../../../../src/utils/NewPuppeteer');
const OutboxPage = {
    //url
    'outbox_url': "https://msg.cnblogs.com/outbox",
    //元素封装
    'checkbox_all': Element('#chkSelAll', '选择所有'),
    'button_delete': Element('#btnBatDel', '删除选中'),
    'dialog_box': Element('body > div.alertify > div > div', '确定要删除吗? '),
    'dialog_ok': Element('body > div.alertify > div > div > nav > button.ok', '确定'),
    'text_delete': Element('#opt_tips', '删除成功'),
    //api
    'api_batchDelete': " https://msg.cnblogs.com/api/message/batchdelete?ids=3711531",

    //常见方法
    /**
     * 删除所有消息
     */
    deleteAllMsg: async function () {
        //1. 点击选中全部
        await newPuppeteer.click(this.checkbox_all);
        //2. 点击确认删除
```

```
        await newPuppeteer.click(this.button_delete);
        //3. 如果弹出框，点击确认删除
        if(await newPuppeteer.isExist(this.dialog_box)){
            await newPuppeteer.click(this.dialog_ok);
        }
    }

};
module.exports=OutboxPage;
```

全部页面：



4-UI与接口测试

新增： test/automation/cases/Message/ComposeMsgTest.js

```
const {describe,it,before,after,afterEach}=require('mocha');
const Base=require('../../../Base');
const loginCookie = require('../../../../src/cookie/HandleCookie');
const {assert}=require('chai');
const inboxPage = require('../../pages/Blog/InboxPage');
const composePage = require('../../pages/Blog/ComposePage');
const outboxPage = require('../../pages/Blog/OutboxPage');
const newPuppeteer = require('../../../../src/utils/NewPuppeteer');
const Api=require('../../../../src/utils/Api')

describe('博客园-短消息',function () {
    before(Base.before);
    // after(Base.after);
    afterEach(Base.afterEach);
    this.timeout(1000000);
    it('Case1-消息发送与删除', async function () {
        //页面1：收件箱
        await loginCookie.loginBlogCheck(inboxPage.inbox_url,null,null);
        //收件箱页面API测试
```

```
        await Api.get(inboxPage.api_current)
            .then((response) => {
                console.log('resp='+JSON.stringify(response));
                assert.equal(response.body.alias, "2910916");
                assert.equal(response.body.displayName, "UI自动化");
                assert.equal(response.status, 200);
            });

        await Api.get(inboxPage.api_getMgsCount)
            .then((response) => {
                console.log('resp='+JSON.stringify(response));
                assert.equal(response.body, 0);
                assert.equal(response.status, 200);
            });

        //页面2：撰写短消息
        await inboxPage.gotoComposeMsg();
        //撰写短消息
        await composePage.composeMsg('2',"title2");
        //判断文字存在：发送成功
        await newPuppeteer.isExist(composePage.text_success);
        //判断元素：返回收信箱存在
        await newPuppeteer.isExist(composePage.button_inbox);
        //页面3：发件箱
        await page.goto(outboxPage.outbox_url);
        //删除所有消息
        await outboxPage.deleteAllMsg();
        //判断删除成功出现
        await newPuppeteer.isExist(outboxPage.text_delete);
    })
})
```

报告