简介

本学期众包平台项目组计划使用 Jenkins+Selenium 的组合分别对众包项目前后端进行持续 CI 和自动化测试。

考虑到前端工程可能使用 OwnCloud 进行管理，因此前端组计划将 Selenium 及前端测试做成一个能够从 Jenkins 流程中拆分的模块进行实现，最终选择了使用 Karma+Mocha 进行前端的单元测试和集成测试，使用 Selenium+Robotframework 进行前端的 E2E 测试。

Karma+Mocha 实施

1. 重新构建 Vue 项目：
    a) 使用 Vue-cli 创建一个新的 Vue 项目
    b) 测试管理工具选择 Karma
    c) js 测试框架选择 Mocha
    d) 测试断言库选择 Chai
    e) 引入 phantomjs-shim
    f) 将测试浏览器配置为 Chrome
    g) npm install 安装原项目依赖及新的测试依赖
2. 配置文件 karma.conf.js：

```
// This is a karma config file. For more details see
//
http://karma-runner.github.io/0.13/config/configuration-file.html
// we are also using it with karma-webpack
//   https://github.com/webpack/karma-webpack

var webpackConfig = require('../../build/webpack.test.conf')

module.exports = function karmaConfig(config) {
  config.set({
    // to run in additional browsers:
    // 1. install corresponding karma launcher
    //    http://karma-runner.github.io/0.13/config/browsers.html
    // 2. add it to the `browsers` array below.
    browsers: ['Chrome'],
    frameworks: ['mocha', 'sinon-chai', 'phantomjs-shim'],
```

```
    reporters: ['spec', 'coverage'],
    files: ['../node_modules/sinon/pkg/sinon.js', // 引入 sinon
      './index.js'],
    preprocessors: {
      './index.js': ['webpack', 'sourcemap']
    },
    webpack: webpackConfig,
    webpackMiddleware: {
      noInfo: false
    },
    coverageReporter: {
      dir: './coverage',
      reporters: [
        { type: 'lcov', subdir: '.' },
        { type: 'text-summary' }
      ]
    }
  })
}
```

3. 使用 index.js 配置测试及测试用例目录，我们将测试范围限制在了众包前端的 /src/views 目录下，因此测试中包含了对组件的单元测试和对单个页面的集成测试。

```
import Vue from 'vue'

Vue.config.productionTip = false

// require all test files (files that ends with .spec.js)
const testsContext = require.context('./specs', true, /\.spec$/)
//const testsContext = require.context('./specs', true, /Login.spec$/)
testsContext.keys().forEach(testsContext)

// require all src files except main.js for coverage.
// you can also change this to match only the subset of files that
// you want coverage for.
const srcContext = require.context('../../src/views', true, /\.vue$/)

srcContext.keys().forEach(srcContext)
```

4. 编写测试用例
5. 使用 npm run test 运行测试用例
6. 按照配置文件在 test/coverage 下找到相应的测试报告
7. 测试用例样例：

```
import Vue from 'vue'
import axios from 'axios'

const MyModuleInjector =
require('!!vue-loader?inject!../../../src/views/ProjectDetail.vue')
const ProjectDetailWithMocks = MyModuleInjector({
    '../axios/api': {
        getQuery(resolve) {
            return 1
        },
        getSession() {
            return "wujie"
        },
        getToken(resolve) {
            return { "tokenid": "wujie", "username": "wujie" }
        },
        participate(url, id, self){
            self.isEnroll = true;
        },
        getProject(url, projectId, self) {
            var data = {
                "status": "200",
                "result": {
                    "requirement": {
                        "id": 1, "requirementName": "微信小游戏开发",
"requirementType": "微信平台开发", "startTime": 1526140800000,
"endTime": 1527696000000, "needManager": 1, "requirementDetail": "开
发一个弹一弹游戏", "file": null, "requirementState": 0, "creatorId": 2,
"projectId": 1
                    },
                    "developerList": [{
                        "username": "test", "email": "test@test.com",
"mobile": "123456789"
                    }]
                }
            }
            self.setValue({data})
            data.result.requirement.requirementState = 1
            self.setValue({data})
            data.result.requirement.requirementState = 2
            self.setValue({data})
            data.result.developerList.push({
                "username": "wujie", "email": "test@test.com", "mobile":
"123456789"
```

```
            })
            self.setValue({data})
            self.participateProject()
            self.checkWorker({
                "username": "test", "email": "test@test.com", "mobile":
"123456789"
            })

        }
    }
})

describe('ProjectDetailWithMocks', () => {
    // 组件实例
    const Constructor = Vue.extend(ProjectDetailWithMocks);
    // 挂载组件
    const vm = new Constructor().$mount();

    it('单个项目详情页面测试', () => {
        vm._watcher.run()
        expect(vm.$data.isLogin).to.equal(true);
        expect(vm.$data.isEnroll).to.equal(true);
    })
})
```

Selenium+Robotframework 实施

目前 Selenium IDE 在 Firefox 60+上正在经历改版过程，缺少一定的功能，比如测试用例的本地导出，推荐使用 Katalon Recorder，能够在 Firefox 55+上提供完整的 Selenium IDE 功能。

Selenium IDE 过程：

1. Firefox 组件商城找到 Katalon Recorder
2. 安装即可

使用 Selenium IDE 可以较快的生成和导出测试脚本。

自动化 E2E 测试使用了 Selenium Webdriver+Robotframework 的组合，原因主要如下：

1. 代码简单，测试脚本比 Selenium Webdriver+JUnit 更简洁
2. Selenium Webdriver 和 Selenium IDE 的测试语句之间存在一定的差异，导致即使将

Selenium IDE 导出的脚本转换为 JUnit 版并不一定能直接运行成功，甚至需要重写。

3. 众包项目并非测试驱动开发，因此页面布局中缺少必要的锚点 id，Selenium Webdriver 缺少一定的解决方案。Robotframework 可以使用自己的断言库对 Selenium Webdriver 库进行补充，简化脚本编写，以及跳过一些坑。

Robotframework 配置过程：

1. Robotframework 基于 Python2.7，安装必要的 Python 环境（Linux 自带）
2. `pip install robotframework` 安装 Robotframework
3. `pip install selenium2library` 安装 selenium 库依赖
4. 编写测试用例
5. 运行 `robot xxx.robot` 运行单个测试套件
6. 在当前目录下找到相应的测试报告、XML 和日志输出
7. 测试套件样例：

```
*** Settings ***
Library    Selenium2Library

*** Variables ***

*** Test Cases ***
Create Requirement
    open browser    http://10.60.38.173/#/    FireFox

    click element    xpath=//a[contains(text(),'登录')]
    click element    xpath=//input[@type='text']
    input text    xpath=//input[@type='text']    test
    input text    xpath=//input[@type='password']    123456
    click element    xpath=//button[@type='button']
    wait until page contains element
//div[@id='app']/div/div[2]/div/div/span/span

    click element
xpath=//div[@id='app']/div/div[2]/div/div[2]/ul/div/li[2]/ul/li[3]
    click element    css=button.el-button.el-button--text
    input text    css=input.el-input__inner    test
    click element
xpath=//div[@id='app']/div/div[2]/div/div[3]/div/div[2]/div/div/div[
2]/form/div[2]/div/div/div/span/span/i
    click element    xpath=//div[4]/div/div/ul/li/span
    click element    xpath=//input[@id='']
```

```
    wait until element is visible    //tr[5]/td/div/span
    click element    //tr[5]/td/div/span
    click element    xpath=(//input[@id=''])[2]
    wait until element is visible
//div[5]/div/div/div[2]/table/tbody/tr[5]/td/div/span
    click element
//div[5]/div/div/div[2]/table/tbody/tr[5]/td/div/span
    click element    css=span.el-switch__core
    click element    css=span.el-switch__button
    click element    css=span.el-switch__button
    click element    css=textarea.el-textarea__inner
    input text    css=textarea.el-textarea__inner    test
    click element    css=button.el-button.el-button--primary

    close browser

Update Requirement
    open browser    http://10.60.38.173/#/    FireFox

    click element    xpath=//a[contains(text(),'登录')]
    click element    xpath=//input[@type='text']
    input text    xpath=//input[@type='text']    test
    input text    xpath=//input[@type='password']    123456
    click element    xpath=//button[@type='button']
    wait until page contains element
xpath=//div[@id='app']/div/div[2]/div/div[2]/ul/div/li[2]/ul/li[3]

    click element
xpath=//div[@id='app']/div/div[2]/div/div[2]/ul/div/li[2]/ul/li[3]

    Execute Javascript
document.getElementsByClassName('el-icon-edit')[0].click()

    click element    css=input.el-input__inner
    input text    css=input.el-input__inner    new_test
    click element
xpath=//div[@id='app']/div/div[2]/div/div[3]/div/div[2]/div/div/div[
2]/form/div[2]/div/div/div/span/span/i
    click element    xpath=//div[4]/div/div/ul/li[2]
    click element    xpath=//input[@id='']
    wait until element is visible    //tr[5]/td/div/span
    click element    //tr[5]/td/div/span
    click element    xpath=(//input[@id=''])[2]
```

```
    wait until element is visible
//div[5]/div/div/div[2]/table/tbody/tr[5]/td/div/span
    click element
//div[5]/div/div/div[2]/table/tbody/tr[5]/td/div/span
    click element    css=span.el-switch__core
    click element    css=textarea.el-textarea__inner
    input text    css=textarea.el-textarea__inner    new test
    click element    css=button.el-button.el-button--primary

    close browser

Delete Requirement
    open browser    http://10.60.38.173/#/    FireFox

    click element    xpath=//a[contains(text(),'登录')]
    click element    xpath=//input[@type='text']
    input text    xpath=//input[@type='text']    test
    input text    xpath=//input[@type='password']    123456
    click element    xpath=//button[@type='button']
    wait until page contains element
xpath=//div[@id='app']/div/div[2]/div/div[2]/ul/div/li[2]/ul/li[3]

    click element
xpath=//div[@id='app']/div/div[2]/div/div[2]/ul/div/li[2]/ul/li[3]
    Execute Javascript
document.getElementsByClassName('el-icon-delete')[0].click()

    close browser
```