前端的测试工具使用selenium-webdriver，该工具为多种语言提供了相应的API，可以和多种语言绑定，我们使用的是JavaScript绑定，使用WebDriverJS编写的selenium测试用例可以直接使用node命令运行

官方文档上的WebDriverJS API使用简介

<http://docs.seleniumhq.org/docs/03_webdriver.jsp#selenium-webdriver-api-commands-and-operations>

以及官方网站上的完整WebDriverJS API文档

<http://seleniumhq.github.io/selenium/docs/api/javascript/index.html>

（暂时没有找到比较好的WebDriverJS中文API文档）

（网上比较多的是selenium-webdriver的java绑定相关的教程，而且selenium有一个selenium IDE工具，作为FireFox的插件，可以实现脚本的自动化录制，在FireFox上的操作会被自动录制成脚本，并且可以导出为不同的语言格式，其中就包括java语言格式的脚本，可以直接在eclipse上运行，不过我还没有实践过。目前还不支持导出为js语言。如果使用java绑定的话，倒是可以使用selenium IDE这个工具的自动录制脚本功能）

关于如何使用Node.js安装selenium-webdriver并且运行一个使用selenium-webdriver编写的简单的case，可以参考这篇博客<http://www.cnblogs.com/fnng/p/5854875.html>

英语好的同学可以直接参看这篇文章：

<https://simpleprogrammer.com/2014/02/03/selenium-with-node-js/>

使用步骤：

1、安装Node.js

2、下载浏览器驱动文件（上面的博客提供了不同的浏览器驱动文件的下载链接），把这些驱动文件放到一个目录下（比如C:\path），然后在环境变量中配置该路径

3、新建一个目录作为当前工程目录（比如C:\demo），在该目录下创建一个test文件夹，用来存放测试文件，在test文件夹下新建一个js文件（如test.js），粘贴如下的代码

var webdriver = require('selenium-webdriver');

var driver = new webdriver.Builder().

withCapabilities(webdriver.Capabilities.chrome()).

build();

driver.get('http://www.google.com');

driver.findElement(webdriver.By.name('q')).sendKeys('simple programmer');

driver.findElement(webdriver.By.name('btnG')).click();

driver.quit();

这是一个使用selenium-webdriver编写的简单的测试脚本，包含打开谷歌首页、输入simple programmer并搜索以及退出这几个简单的动作

4、安装selenium-webdriver。打开cmd，切换到该工程目录下，执行如下语句

npm install selenium-webdriver

该语句将selenium-webdriver包下载到当前工程目录下的node\_modules文件夹

5、安装好selenium-webdriver之后，就可以运行上面新建test.js测试脚本了，在cmd中输入

node test\test.js

该测试脚本就会被运行了

以上是selenium-webdriver在Node.js上的使用，下面介绍在这一基础上，如何结合Mocha框架编写并运行测试用例，具体仍然可以参考上面的那篇英文文章：

<https://simpleprogrammer.com/2014/02/03/selenium-with-node-js/>

（关于Mocha的使用和基本语法，可以参考这篇博客，Mocha可以用来做包括前端和后台的测试

<http://blog.csdn.net/yuhason0422/article/details/42673687>）

下面简单介绍一下在以上基础上，如何结合Mocha和selenium-webdriver做自动化测试（以下仅是工具使用简介，具体的Mocha语法请参看上面的文章）

假设已经在当前工程目录下安装了selnium-webdriver

1. 首先在全局安装Mocha

npm install -g mocha

1. 将上面的test.js文件中的内容替换为如下内容

var assert = require('assert'),

test = require('selenium-webdriver/testing'),

webdriver = require('selenium-webdriver');

test.describe('Google Search', function() {

test.it('should work', function() {

var driver = new webdriver.Builder().

withCapabilities(webdriver.Capabilities.chrome()).

build();

driver.get('http://www.google.com');

var searchBox = driver.findElement(webdriver.By.name('q'));

searchBox.sendKeys('simple programmer');

searchBox.getAttribute('value').then(function(value) {

assert.equal(value, 'simple programmer');

});

driver.quit();

});

});

这是用Mocha框架编写的前端测试脚本，功能和上面的脚本相同

注意该前端测试脚本和一般的用Mocha框架编写的测试脚本的区别，即上面代码中用阴影标示出的部分

1. 使用mocha命令执行test.js

mocha test\test.js

以上是如何使用selenium-webdriver工具和Mocha框架编写并且执行前端自动化测试脚本的介绍，接下来介绍在这一基础上如何结合Grunt框架来做自动化测试

Grunt的使用入门可以参看Grunt的官方中文文档<http://www.gruntjs.net/getting-started>

Grunt的基础使用这里就不介绍了，下面介绍一下使用grunt-mocha-test插件在Grunt上执行Mocha测试脚本的步骤（关于grunt-mocha-test插件，具体可以参见官方文档<https://www.npmjs.com/package/grunt-mocha-test>）：

1. 进入工程目录，执行如下命令，生成package.json项目配置文件

npm init

输入该命令之后，会提示输入一些相关的项目信息，为了简单起见，可以一路按回车，之后就可以看到当前工程目录下多出了一个package.json文件

1. 安装selenium-webdriver

npm install grunt --save-dev

参数--save-dev表示安装selenium-webdriver的同时在package.json文件的devDependensies属性下添加该依赖项

1. 安装Grunt

npm install grunt --save-dev

1. 安装grunt-mocha-test插件

npm install grunt-mocha-test --save-dev

1. 在当前工程目录下安装Mocha（注意不是全局安装）

npm install mocha --save-dev

1. 在当前目录下生成一个Gruntfile.js文件，该文件定义Grunt的任务配置，在文件中粘贴如下代码

module.exports = function(grunt){

grunt.initConfig({

mochaTest: {

test: {

options: {

},

src: ['test/\*.js']

}

}

});

grunt.loadNpmTasks('grunt-mocha-test');

grunt.registerTask('tahitianTest', 'mochaTest');

};

简单说明一下上面的代码：

module.exports = function(grunt){}函数是Grunt框架通用的wrapper函数，该wrapper函数中还包括三个通用函数grunt.initConfig()、grunt.loadNpmTasks()和grunt.registerTask()

grunt.initConfig()是任务配置函数，在其中配置一些需要执行的任务（task），mochaTest就是一个grunt-mocha-test插件定义的任务。在任务中可以包含多个目标（target），如上面的test，目标的名字是自定义的。目标内可以设置一些配置项（options，关于grunt-mocha-test插件的options说明可以参看该插件的官方文档https://www.npmjs.com/package/grunt-mocha-test），除了目标级的options，也可以设置任务级的options，上面的例子中没有设置任何的options。src属性指定需要执行的mocha测试脚本文件，上面的例子中使用了通配符，表示执行当前目录下的test文件夹中的所有js文件

grunt.loadNpmTasks()函数加载本地的插件

grunt.registerTask()是自定义任务函数，该函数的第一个参数是自定义的任务名，自定义的任务的内容可以是已经存在的一个或者多个任务，也可以是自定义的函数，上面的例子中创建了一个自定义的任务tahitianTest，任务内容是已经存在的mochaTest任务（tahitianTest相当于是mochaTest任务的别名），设置自定义任务是可选的

保存Gruntfile.js文件

1. 回到cmd，输入以下指令执行Grunt任务

grunt tahitianTest

该命令会自动执行当前目录下的test文件夹中的所有mocha测试脚本文件

（直接使用Mocha命令mocha test\\*.js也可以实现相同功能）

以上就是我这几天所学习的测试框架的基本内容。我目前也只限于知道如何把一个case给跑起来，一些具体的配置我还不清楚，关于为什么要使用这些框架也不是很明白。（尤其是grunt，就我目前所知，唯一的好处就是可以批量执行一些测试任务，但是这个功能通过mocha命令也可以实现。可能grunt-mocha-test插件的options配置属性里会有一些有用的东西，这个还得到时候再仔细看看）