Table of Contents

Web自动化测试

目标

1. 了解什么是自动化
2. 理解什么是自动化测试
3. 为什么要使用自动化测试

---

1. 什么是自动化？

概念：由机器设备代替人为自动完成指定目标的过程

1.1 优点：

1. 减少人工劳动力
2. 工作效率提高
3. 产品规格统一标准
4. 规模化(批量生产)

2. 什么是自动化测试

概念：让程序代替人为去验证程序功能的过程

2.1 为什么要进行自动化测试？

1. 解决-回归测试
2. 解决-压力测试
3. 解决-兼容性测试
4. 提高测试效率,保证产品质量

回归测试：项目在发新版本之后对项目之前的功能进行验证；
压力测试：可以理解多用户同时去操作软件，统计软件服务器处理多用户请求的能力
兼容性测试：不同浏览器（IE、Firefox、Chrome）等等

2.2 自动化测试相关知识

自动化测试在什么阶段开始？

功能测试完毕(手工测试)

手工测试：就是由人去一个一个输入用例，然后观察结果；

自动化测试所属分类

1. 黑盒测试(功能测试)
2. 灰盒测试(接口测试)
3. 白盒测试(单元测试)

提示：Web自动化测试属于黑盒测试(功能测试)

优点

1. 较少的时间内运行更多的测试用例；
2. 自动化脚本可重复运行；
3. 减少人为的错误；
4. 测试数据存储

缺点

1. 不能取代手工测试；
2. 手工测试比自动化测试发现的缺陷更多；
3. 测试人员技能要求；

误区：

1). 自动化测试完全替代手工测试
2). 自动化测试一定比手工测试厉害
3). 自动化可以发掘更多的BUG

思考

在软件测试领域中，自动化测试有哪些分类呢？

3. 自动化测试分类

1. Web-(UI)自动化测试(本阶段学习)
2. 接口-自动化测试

3. 移动(app)-自动化测试
4. 单元测试-自动化测试

4. Web自动化测试课程安排

序号	阶段	知识点
1	第一阶段 自动化入门	1. 认识自动化及自动化测试 2. 自动化测试工具(框架)选择 3. SeleniumIDE插件的使用 4. Firebug插件工具及使用
2	第二阶段 **Web**自动化工具**(WebDriver)**基础篇	1. 元素基础定位 2. Xpath、CSS元素定位方式 3. 元素操作 4. 浏览器的操作方法
3	第三阶段 **Web**自动化工具**(WebDriver)**中级篇	1. 鼠标操作 2. 键盘操作 3. 元素等待 4. HTML特殊元素处理 5. 窗口截图
4	第四阶段 **Web**自动化**(**高级篇**)**	1. 自动化测试模型 2. UnitTest框架 3. UnitTest断言 4. 生成HTML测试报告 5. 测试报告自动发送邮件

第一章

目标

1. 什么是Web自动化测试
2. Web自动化测试工具

Web自动化测试

目标

1. 了解什么是Web自动化测试
2. 了解Web自动化测试常用工具

1. 什么是Web自动化测试？

概念：让程序代替人为自动验证Web项目功能的过程

2. 什么Web项目适合做自动化测试？

1.需求变动不频繁
2.项目周期长
3.项目需要回归测试

3. 如何进行Web自动化测试？(主流测试-工具)

1. QTP（收费）
   QTP是商业的功能测试工具，收费，支持web，桌面自动化测试。
2. Selenium（开源）【本阶段学习】
   Selenium是开源的web自动测试工具，免费，主要做功能测试。
3. Jmeter（开源、Web、接口、性能）
   Jmeter是由Apache公司使用Java平台开发的一款测试工具，支持（Web、接口测试、性能测试）
   提示：Web测试在通信层(无UI界面)
5. Loadrunner（收费、Web、性能）
   Loadrunner是商业性能测试工具，收费，功能强大，适合做复杂场景的性能测试
6. Robot framework
   Robot Framework是一个基于Python可扩展地(关键字驱动)的测试自动化框架；

3.1 主流工具-汇结：

Web自动化测试：selenium、robot framework
App端自动化测试：Appium、Monkeyrunner、UIautomation

PC客户端（win32）自动化测试：QTP
接口自动化测试：Jmeter、Postman、httpUnit、RESTClient
云测平台：Testin Testbird
性能测试：Jmeter、LoadRunner

4. 什么是Selenium？

概念： Selenium是一个用于Web应用程序测试的工具；中文的意思（硒）

4.1 Selenium特点

1. 开源软件：源代码开放可以根据需要来增加工具的某些功能
2. 跨平台：linux 、windows 、mac
3. 核心功能：就是可以在多个浏览器上进行自动化测试
4. 多语言：Java、Python、C#、JavaScript、Ruby等
5. 成熟稳定：目前已经被google ，百度， 腾讯等公司广泛使用
6. 功能强大：能够实现类似商业工具的大部分功能，因为开源性，可实现定制化功能

4.1 Selenium家族(发展史)【了解】

重点：
1. SeleniumIDE
2. Selenium2.0(WebDriver)

Selenium IDE安装与运行

目标

1. 使用Selenium IDE录制脚本
2. 使用Selenium IDE录制的脚本转换成Python语言

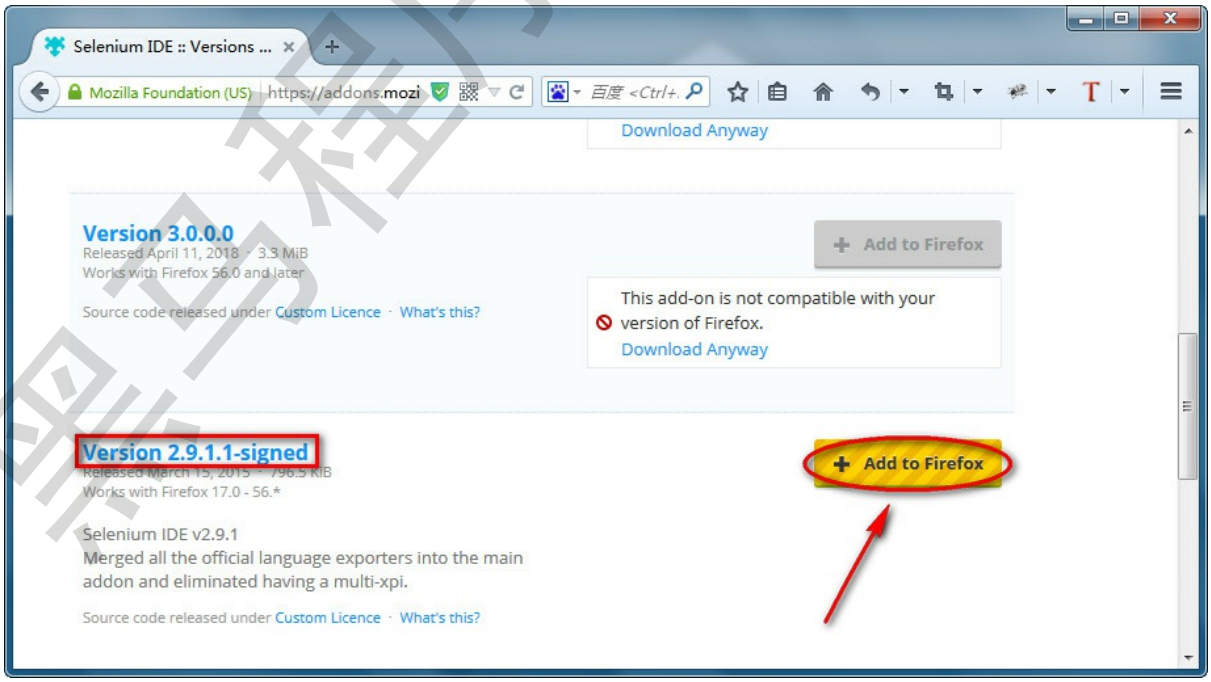
1. Selenium IDE 是什么？

Selenium IDE：是一个Firefox插件，用于记录和播放用户与浏览器的交互。（录制Web操作脚本）

1.1 为什么要学习Selenium IDE？

1. 使用Selenium IDE录制脚本，体验自动化脚本魅力
2. 使用Selenium IDE录制的脚本转换为代码语言
   (在后期我们自己设计脚本时，如果不知道用什么方式定位元素，可使用此方法参考)

1.2 安装方式



1. 官网安装

Version: 2.9.1.1

通过官网安装插件：https://addons.mozilla.org/en-GB/firefox/addon/selenium-ide/versions/

2. 附加组件管理器

1). 火狐浏览器 V24-V35

2). 附加组件管理器-->搜索selenium IDE

提示：

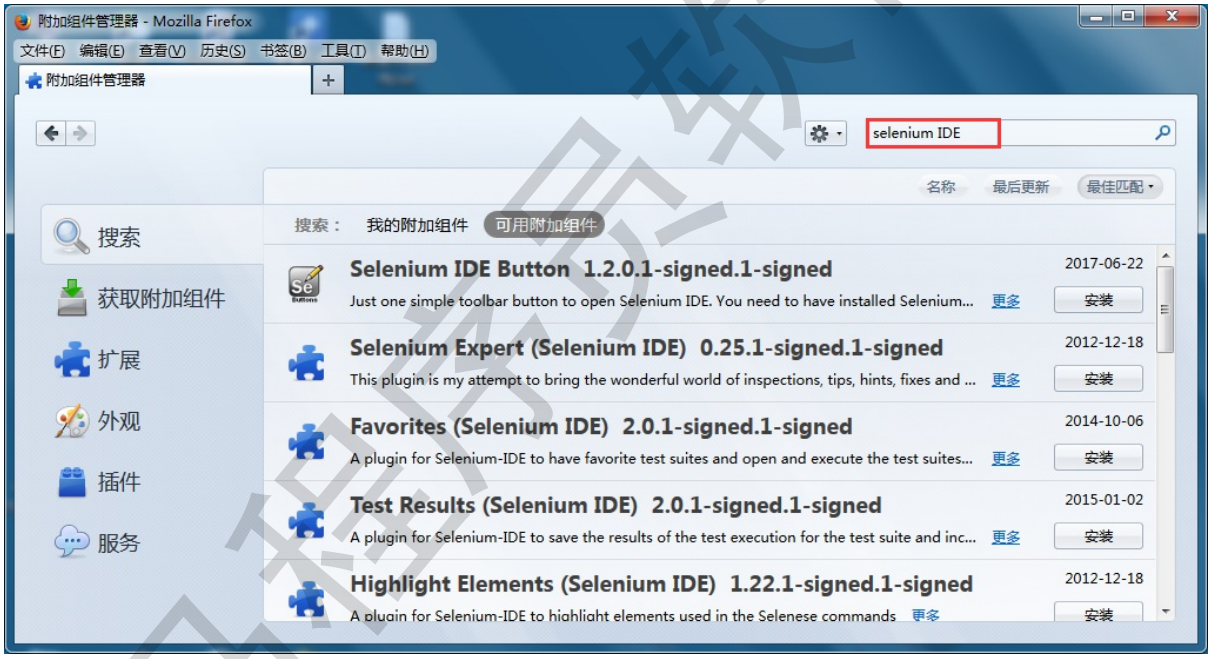
1. IDE前面有个空格
2. 附加组件管理器启动方式-
    1) 工具菜单->附加组件
    2) Ctrl+Shift+A

3. 离线安装

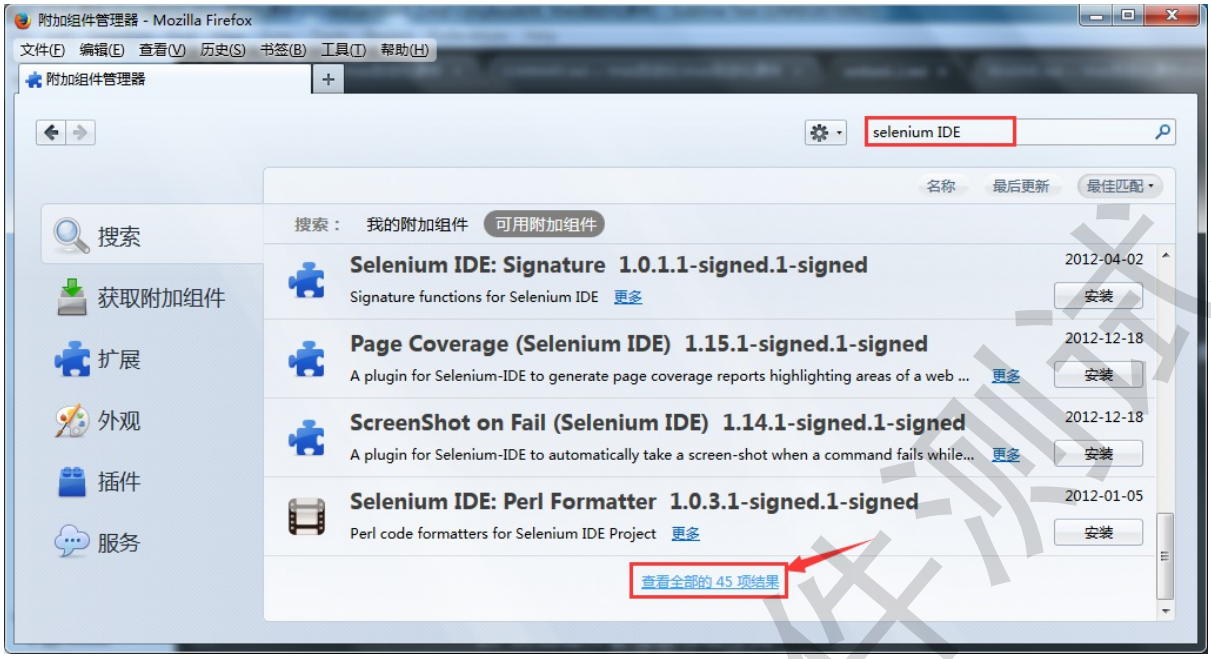
下载：https://github.com/SeleniumHQ/selenium-ide/releases

安装：下载好selenium_ide-2.9.1-fx.xpi直接拖入浏览器安装

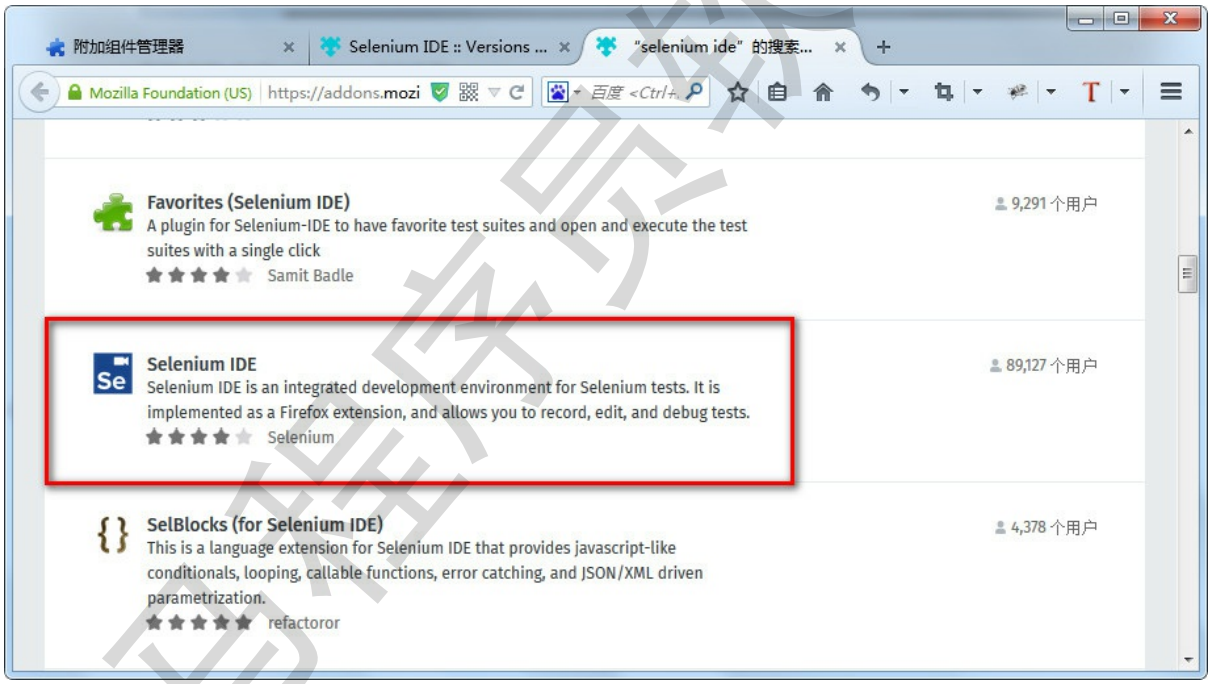
在线安装 搜索**selenium IDE**示意图



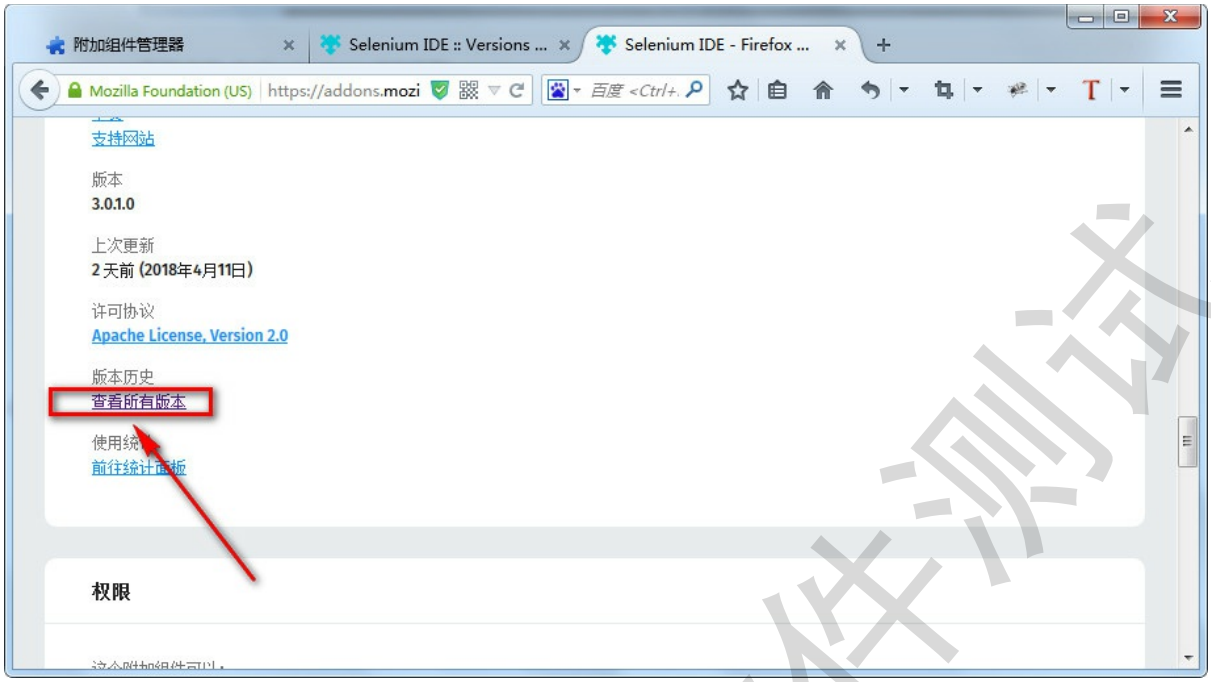
在线安装 点击查看全部**XX**项结果



在线安装 选择**selenium IDE**安装



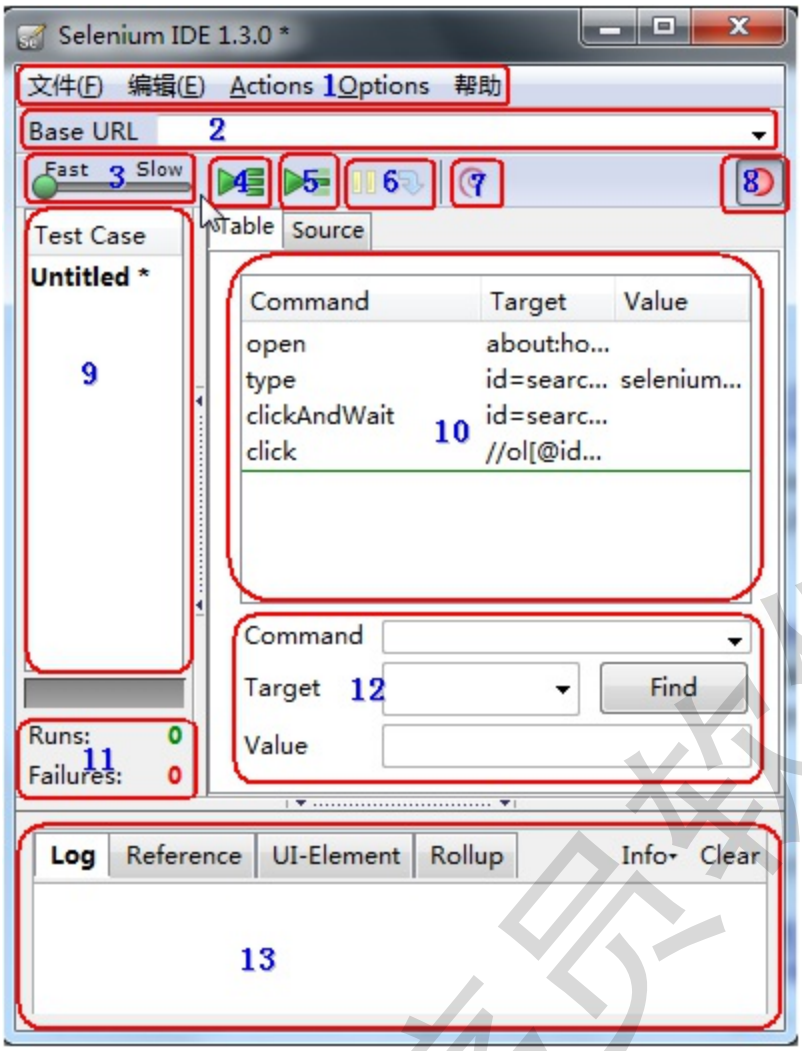
滚动条下拉选择 查看所有版本



火狐浏览器 V35.0 选择 selenium IDE 2.9.1.1

1.3 Selenium IDE运行

1. Ctrl+Alt+S
2. 工具栏—>Selenium IDE



1. 文件：创建、打开和保存测试案例和测试案例集。编辑：复制、粘贴、删除、撤销和选择测试案例中的所有命

令。Options ：用于设置seleniunm IDE。

2. 用来填写被测网站的地址。
3. 速度控制：控制案例的运行速度。
4. 运行所有：运行一个测试案例集中的所有案例。
5. 运行：运行当前选定的测试案例。
6. 暂停/恢复：暂停和恢复测试案例执行。
7. 单步：可以运行一个案例中的一行命令。
8. 录制：点击之后，开始记录你对浏览器的操作。
9. 案例集列表。
10. 测试脚本：table标签：用表格形式展现命令及参数。source标签：用原始方式展现，默认是HTML语言格式，

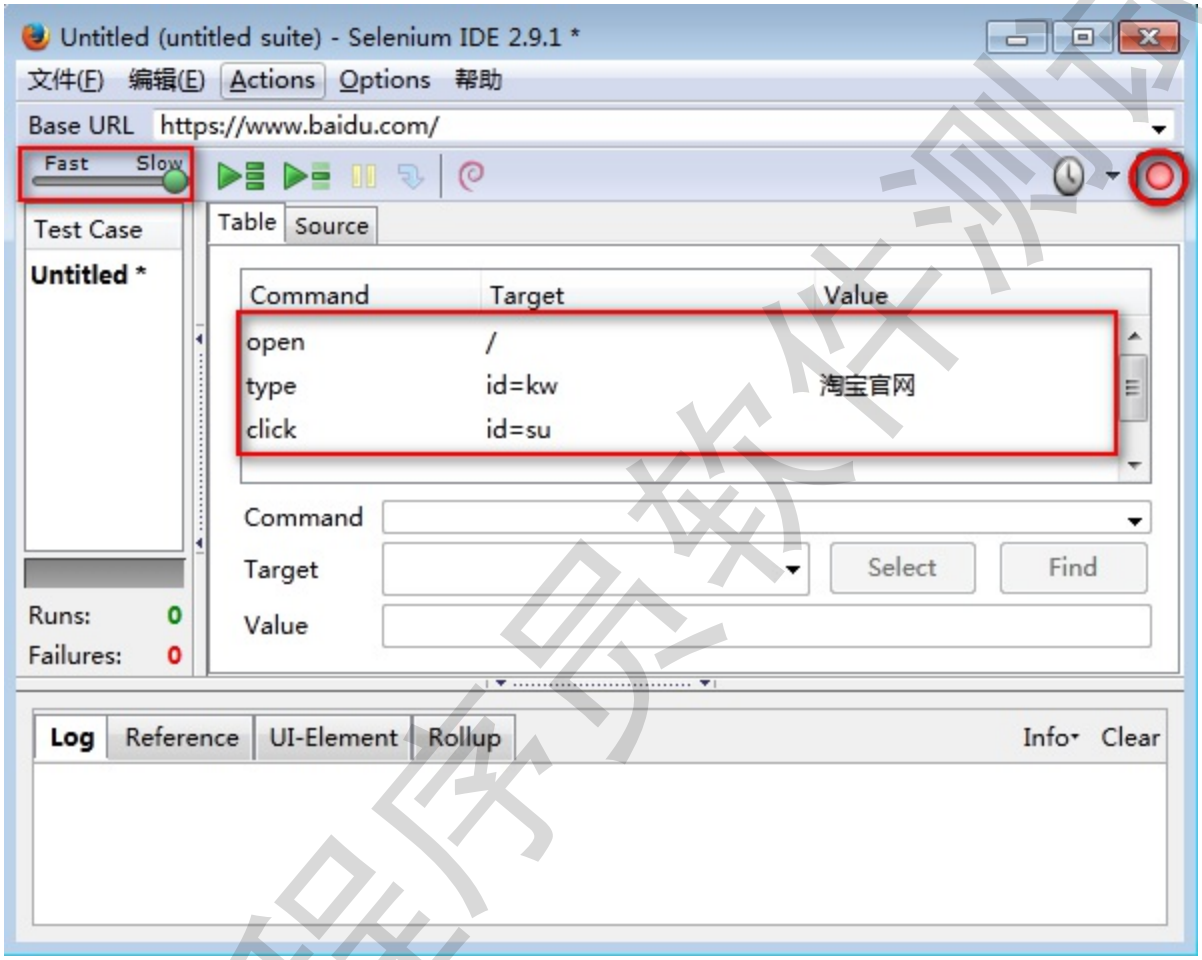
也可以用其他语言展示。

11. 查看脚本运行通过/失败的个数。
12. 当选中前命令对应参数。
13. 日志/参考/UI元素/Rollup

练习1

需求：使用Selenium IDE插件录制->打开百度搜索引擎，搜索框输入[淘宝官网]，点击[百度一下]按钮

练习1 脚本



重点分析：

1. 录制：录制时红色录制按钮一定要打开->按下状态
2. 回放：由于网络延迟原因-建议选择最低
3. 浏览器：回放时浏览器要保持打开状态（否则点击回放，脚本无响应）

重点说明：

1. id=kw：为百度搜索文本框id属性和值
2. id=su：为百度一下按钮id属性和值

思考？

如何快速查找一个元素标签的属性和值？

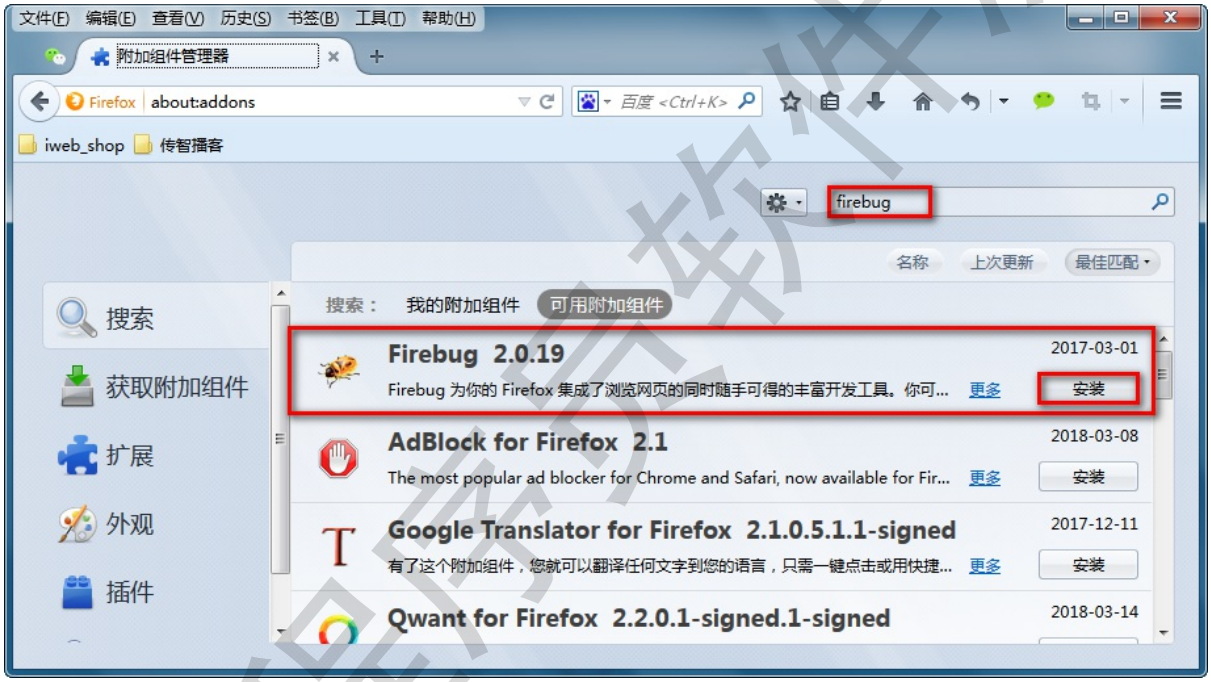
2. 定位调试插件

1. FireBug【重要】

FireBug插件是火狐浏览器一款插件，能够调试所有网站语言，同时也可以快速定位HTML页面中的元素；

作用：定位元素(获取元素定位和查看元素属性)；

2.1 Firebug 插件安装



在线安装：

1). 火狐浏览器 V35

2). 附加组件管理器-->搜索FireBug

练习2

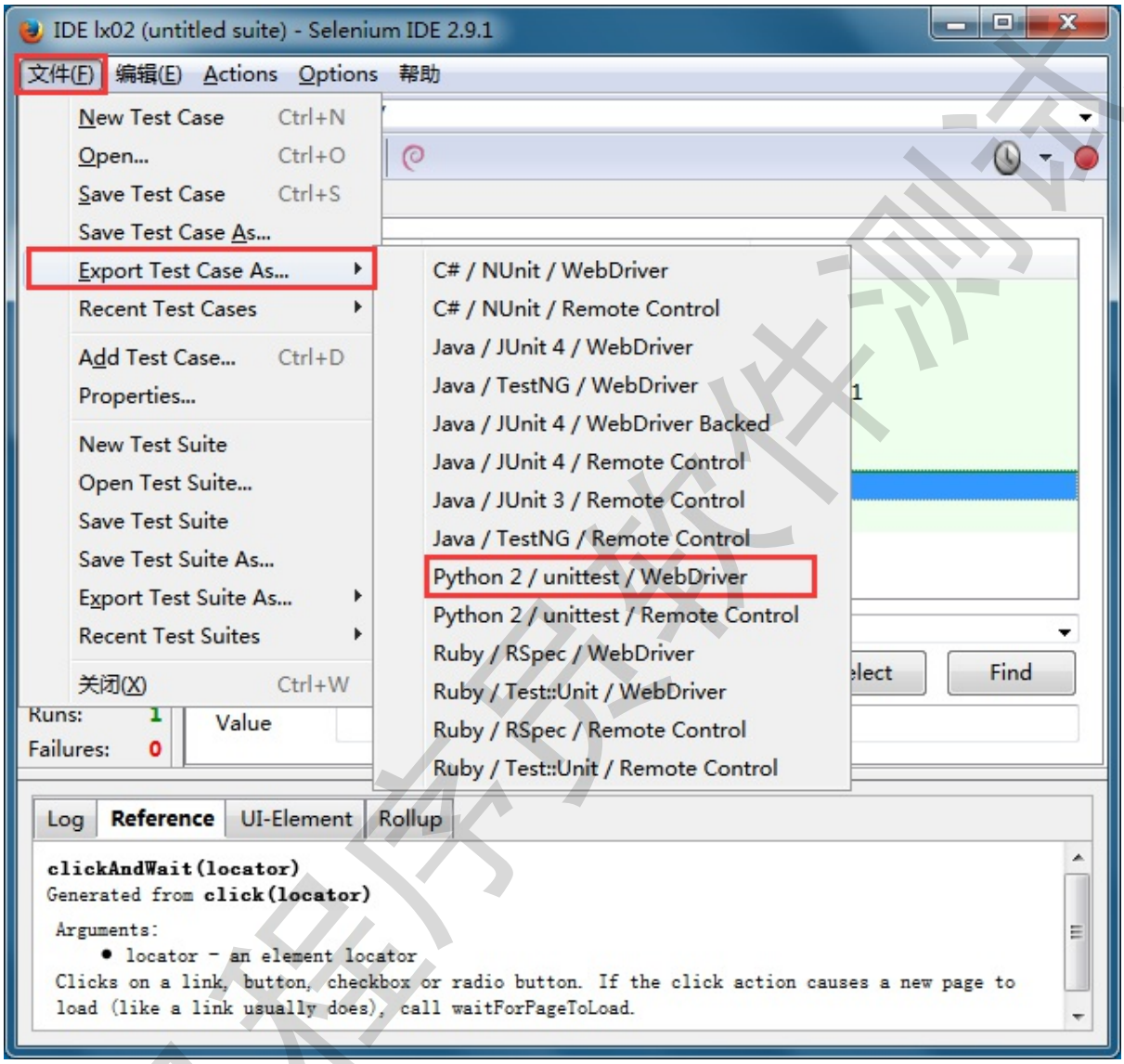
实现天涯论坛自动登录个人账号

练习2 测试数据

1. 地址：http://bbs.tianya.cn/
2. user:testcn1001

```
3. pwd:test2XXX
```

将练习2脚本 转换成Python语言




```
文件菜单->Export Test Cast As..->python2/unittest/WebDriver
```

提示：

```
录制脚本时候是录制鼠标和键盘的所有在浏览器操作，那么脚本会出现多余的步骤，有时候我们需要手动填写脚本
或修改脚本,所有我们有必要对Selenium IDE脚本编辑与操作有所了解;
```

3. Selenium IDE脚本编辑与操作 【了解】

目的：手动修改或编写脚本（采用录制方式很容易记录出多余的操作）

3.1 编辑一行命令

在Table标签下选中某一行命令，命令由command、Target、value三部分组成。可以对这三部分内容那进行编辑。

3.2 插入命令

在某一条命令上右击，选择“insert new command”命令，就可以插入一个空白，然后对空白行进程编辑

3.3 插入注释

鼠标右击选择“insert new comment”命令插入注解空白行，本行内容不被执行，可以帮助我们更好的理解脚本，
插入的内容以紫色字体显示。

3.4 移动命令

有时我们需要移动某行命令的顺序，我们只需要左击鼠标拖动到相应的位置即可。

3.5 删除命令

选择单个或多个命令，然后点击鼠标右键选择“Delete”

3.6 命令执行

选定要执行的命令点击单个执行按钮即可，注意：有一些命令必须依赖于前面命令的运行结果才能成功执行，否则会导致执行失败。

提示：

我们对脚本的编辑和操作已有所了解，那么我们练习1和练习2录制出来的脚本中那些命令又都是有什么作用？

4. Selenium IDE常用命令【了解】

在这里我们只对几个常用的命令做个介绍

4.1 open(url)命令

作用：打开指定的URL，URL可以为相对或是绝对URL；

Target：要打开的URL；value值为空
    1). 当Target为空，将打开Base URL中填写的页面；
    2). 当Target不为空且值为相对路径，将打开Base URL + Target页面。如，假设Base URL为http://www.zhi97.com，而Target为/about.aspx，则执行open命令时，将打开http://www.zhi97.com/about.aspx
    3). 当Target以http://开头时，将忽略Base URL，直接打开Target的网址；

4.2 pause(waitTime)

作用：暂停脚本运行
waitTime：等待时间，单位为ms；//Target=1000

4.3 goBack()

作用：模拟单击浏览器的后退按钮；
提示：由于没有参数，所以Target和Value可不填；

4.4 refresh()

作用：刷新当前页；
提示：由于没有参数，所以Target和Value可不填；

4.5 click(locator)

作用：单击一个链接、按钮、复选框或单选按钮；
提示：如果该单击事件导致新的页面加载，命令将会加上后缀“AndWait”，即“clickAnd Wait”，或“waitForPageToLoad”命令；

4.6 type(locator,value)

作用：向指定输入域中输入指定值；也可为下拉框、复选框和单选框按钮赋值.
Target：元素的定位表达式；
Value：要输入的值；

4.7 close()

作用：模拟用户单击窗口上的关闭按钮；
提示：由于没有参数，所以Target和Value可不填；

---

5. 总结

1. 为什么学习Selenium IDE插件工具
2. Selenium安装、启动方式
3. FireBug作用
4. Selenium IDE常用命令

第2章-WebDriver(Selenium2.0)

目标

1. 掌握WebDriver元素定位方法

回顾 Selenium家族

对于我们只需要关注以下两点：
1. SeleniumIDE(已学完)
2. Selenium2.0(WebDriver)

提示：
    1). Selenium2.0=Selenium1.0+WebDriver
    2). Selenium1.0 和 WebDriver原属于两个不同的东西，由于某种原因已合并
    3). Selenium2.0以后我们简称WebDriver

WebDriver概述

目标

1. 了解WebDriver概述
2. WebDriver环境搭建

1. 什么是WebDriver?

1. Webdriver (Selenium2）是一种用于Web应用程序的自动测试工具；
2. 它提供了一套友好的API；
3. Webdriver完全就是一套类库，不依赖于任何测试框架，除了必要的浏览器驱动；

说明：

API：应用编程接口说明（WebDriver类库内封装非常多的方法，要使用这些方法，就需要友好的调用命名规则）

思考

WebDriverAPI都支持哪些浏览器？

1.1 WebDriverAPI 支持的浏览器

1. Firefox （FirefoxDriver）【推荐-本阶段学习使用】
2. IE（InternetExplorerDriver）
3. Opera（OperaDriver）
4. Chrome （ChromeDriver）
5. safari（SafariDriver）
6. HtmlUnit (HtmlUnit Driver)

提示：

Firefox、Chrome：对元素定位和操作有良好的支持，同时对JavaScript支持也非常好。
IE:只能在windows平台运行，所有浏览器中运行速度最慢
HtmlUnit：无GUI(界面)运行，运行速度最快；

推荐原因：

1. Selenium IDE

```
    2. FireBug
    3. 对WebDriver API支持良好
```

思考：

```
我们知道API为应用编程时使用，那么它支持哪些编程语言来使用呢？
```

1.2 WebDriverAPI 支持的开发语言

官网文档：https://docs.seleniumhq.org/docs/03_webdriver.jsp

```
1. Java
2. Python
3. PHP
4. JavaScript
5. Perl
6. Ruby
7. C#
```

2. 为什么要学习WebDriver?

```
1. 自动化测试概念
2. WebDriver-定位元素
3. WebDriver-操作元素
```

3. 环境搭建

3.1 为什么要环境搭建？

```
1. 盖房子
2. MP3
3. 开发语言
```

3.2 基于Python环境搭建

```
1. Windows系统（在这里我们以Windows7为案例）
2. Python 3.5（以上版本）
3. 安装selenium包
4. 浏览器
5. 安装PyCharm
```

说明：Python3 和PyCharm咱们上阶段课已使用，在这里不在重复；

4. selenium安装

说明：在安装selenium时，前提是Python3.5以上版本安装完毕且能正常运行

4.1 selenium 安装、卸载、查看命令

```
安装：pip install selenium==2.48.0
      1). pip:通用的 Python 包管理工具。提供了对 Python 包的查找、下载、安装、卸载的功能。
      2). install: 安装命令
      3). selenium==2.48.0: 指定安装selenium2.48.0版本（如果不指定版本默认为最新版本）
卸载：pip uninstall selenium
查看：pip show selenium
```

4.2 火狐浏览器【推荐】

```
1. FireFox 48以上版本
        Selenium 3.X +FireFox驱动—geckodriver
2. Firefox 48 以下版本
        Selenium2.X 内置驱动
```

4.3 IE浏览器(了解)

```
1. IE 9以上版本
        Selenium3.X +IE驱动
2. IE 9以下版本
        Selenium 2.X +IE驱动
```

4.4 谷歌浏览器

```
selenium2.x/3.x +Chrome驱动
```

chromedriver版本	支持的Chrome版本
v2.24	v52-54
v2.23	v51-53
v2.22	v49-52
v2.21	v46-50
v2.20	v43-48
v2.19	v43-47
v2.18	v43-46
v2.17	v42-43
v2.13	v42-45
v2.15	v40-43
v2.14	v39-42
v2.13	v38-41
v2.12	v36-40
v2.11	v36-40
v2.13	v38-41
v2.12	v36-40
v2.11	v36-40
v2.10	v33-36
v2.9	v31-34
v2.8	v30-33
v2.7	v30-33
v2.6	v29-32
v2.5	v29-32
v2.4	v29-32

4.5 浏览器-总结

各个驱动下载地址： http://www.seleniumhq.org/download/

1. 浏览器的版本和驱动版本要一致！
   （如果是32bit浏览器而Driver是64bit则会导致脚本运行失败！）
2. 浏览器驱动下载好后需要添加Path环境便令中，或者直接放到Python安装目录，因为Python以添加到Path中
3. 推荐使用火狐浏览器(24、35)版

---

5. 总结

1. WebDriver是什么？
2. 为什么要搭建环境？
3. selenium 安装、卸载、查看命令
4. 为什么推荐火狐浏览器和火狐48版本以下

WebDriver-元素定位

目标

1. 了解元素各种定位方法
2. 掌握id、name、class_name、tag_name、link_text、partial_link_text定位的使用

1. 为什么要学习元素定位方式？

1. 让程序操作指定元素，就必须先找到此元素；
2. 程序不像人类用眼睛直接定位到元素；
3. WebDriver提供了八种定位元素方式

2. WebDriver 元素定位方式

1. id
2. name
3. class_name
4. tag_name
5. link_text
6. partial_link_text
7. Xpath
8. Css

定位方式分类-汇总：

1). id、name、class_name：为元素属性定位
2). tag_name：为元素标签名称
3). link_text、partial_link_text：为超链接定位(a标签)
4). Xpath：为元素路径定位
5). Css：为CSS选择器定位

案例-1 注册页面

1. 为了更好的学习这八种方式和网络的关系，我们在案例-1注册页面上来练习自动化脚本设计，提高学习效率和

脚本执行速率
2. 语言使用Python
3. 开发工具使用Pycharm
4. selenium使用2.48.0

2.1 id定位

说明：HTML规定id属性在整个HTML文档中必须是唯一的，id定位就是通过元素的id属性来定位元素；
前提：元素有id属性

实现案例-1需求：
1). 打开注册A.html页面，使用id定位，自动填写(账号A：admin、密码A:123456)
2). 填写完毕后，3秒钟关闭浏览器窗口

id定位方法

```
find_element_by_id()
```

id定位实现 步骤分析

1. 导入selenium包 --> from selenium import webdriver
2. 导入time包 --> from time import sleep
3. 实例化火狐浏览器 --> driver=webdriver.Firefox()
4. 打开注册A.html --> driver.get(url)
5. 调用id定位方法 --> driver.find_element_by_id("")
6. 使用send_keys()方法发送数据 --> .send_keys("admin")
7. 暂停3秒 --> sleep(3)
8. 关闭浏览器 --> quit()

说明：为了接下来更好的而学习体验，我们先暂时使用下，send_keys()和quit()方法，在2.4节元素操作讲解；

id定位 案例-1代码：

```
from selenium import webdriver
from time import sleep
driver=webdriver.Firefox()
url='E:\\测试\\课件\\Web自动化\\Web自动化课件\\02img\\注册A.html'
driver.get(url)
user=driver.find_element_by_id("userA")
user.send_keys("admin")
pwd=driver.find_element_by_id("passwordA")
pwd.send_keys("123456")
sleep(3)
```

```
driver.quit()
```

id定位-总结

```
1. 导包
2. url中\\转义
3. id定位方法
3. 发送内容方法
4. 暂停方法
5. 关闭浏览器
```

2.2 name定位

```
说明：HTML规定name属性来指定元素名称，因此它的作用更像人名，name的属性值在当前文档中可以不是唯一的
      ，name定位就是根据元素name属性来定位
前提：元素有name属性

实现案例-1需求：
    1). 打开注册A.html页面，使用name定位，自动填写(账号A：admin、密码A:123456)
    2). 填写完毕后，3秒钟关闭浏览器窗口
```

name定位方法

```
find_element_by_name()
```

name定位实现 步骤分析

```
1. 参考id定位
```

2.3 class_name定位

```
说明：HTML规定了class来指定元素的类名，用法和name、id类似；
前提：元素有class属性

实现案例-1需求：
    通过class_name定位电话号码A，并发送18611111111
```

class_name定位方法

```
find_element_by_class_name()
```

class_name定位实现 步骤分析

```
1. 参考id定位
```

2.4 tag_name定位

```
说明：HTML本质就是由不同的tag(标签)组成，而每个tag都是指同一类，所以tag定位效率低，
      一般不建议使用；tag_name定位就是通过标签名来定位；
实现案例-1需求：
    1). 打开注册A.html页面，使用tag_name定位，自动填写(账号A：admin)
    2). 填写完毕后，3秒钟关闭浏览器窗口
```

tag_name定位方法

```
1. find_element_by_tag_name()
        返回：符合条件的第一个标签
2. 如何获取第二个元素？稍后(2.7节)讲解
```

tag_name定位实现 步骤分析

```
1. 参考id定位
```

2.5 link_text定位

```
说明：link_text定位与前面4个定位有所不同，它专门用来定位超链接文本（<a>标签</a>）。

实现案例-1需求：
    1). 打开注册A.html页面，使用link_text定位(访问 新浪 网站)超链接
    2). 3秒钟关闭浏览器窗口
```

link_text定位方法

```
1. 方法：find_element_by_link_text()
2. 说明：需要传入a标签全部文本(访问 新浪 网站)
```

link_text 步骤分析

```
1. 参考id定位
```

2. 点击 --> click()

2.6 partial_link_text定位

说明：partial_link_text定位是对link_text定位的补充，partial_like_text为模糊匹配：link_text全部匹配

实现案例-1需求：

1). 打开注册A.html页面，使用partial_link_text定位(访问 新浪 网站)超链接
2). 3秒钟关闭浏览器窗口

partial_link_text定位方法

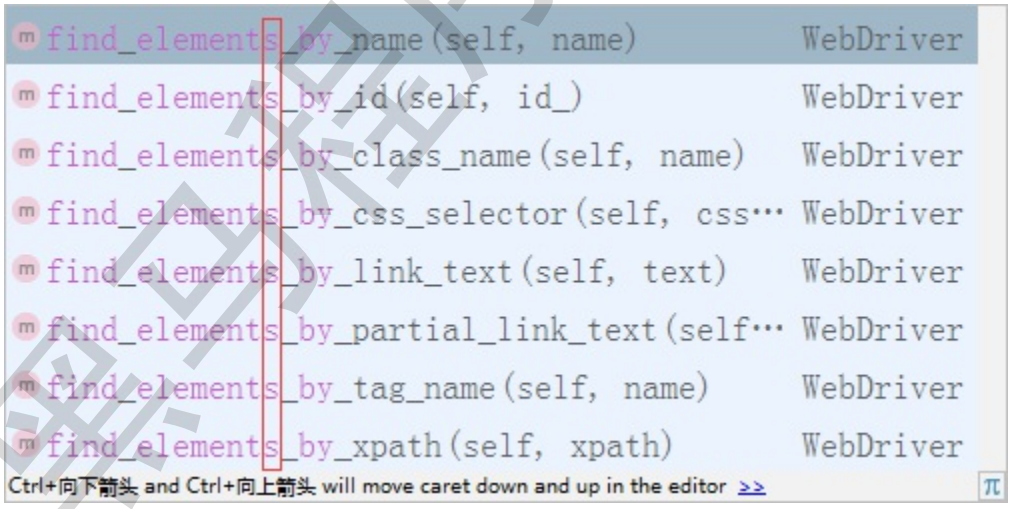
1. 方法：find_element_by_partial_link_text()
2. 说明：需要传入a标签局部文本-能表达唯一性(访问 新浪 网站)

partial_link_text 步骤分析

1. 参考link_text定位

提示

在我们学习使用以上方法的时候，发现有个共同的相似方法：



2.7 find_element[s]_by_XXX()

作用：

1). 查找定位所有符合条件的元素

```
    2). 返回的定位元素格式为数组(列表)格式;
说明:
    1). 列表数据格式的读取需要指定下标(下标从0开始)
```

操作**(2.4 tag_name)**

```
说明：使用tag_name获取第二个元素(密码框)

代码:
    ...
    driver.find_elements_by_tag_name("input")[1].send_keys("123456")
    ...
```

---

3. 2.1-2.6定位 总结

```
1. id、name、class_name
2. tag_name
3. link_text、partial_link_text
4. find_elements_by_XXX()
```

思考？

```
1. 在实际项目中标签没有id、name、class属性改如何定位？
2. id、name、class属性值为动态获取，随着刷新或加载而变化，改如何定位？
```

Xpath、CSS定位

目标

1. 熟悉Xpath定位策略
2. 熟悉CSS定位策略

为什么要学习Xpath、CSS定位？

1. 在实际项目中标签没有id、name、class属性
2. id、name、class属性值为动态获取，随着刷新或加载而变化

1. 什么是Xpath？

1. XPath即为XML Path 的简称，它是一种用来确定XML文档中某部分位置的语言。
2. HTML可以看做是XML的一种实现，所以Selenium用户可以使用这种强大的语言在Web应用中定位元素。

XML：一种标记语言，用于数据的存储和传递。 后缀.xml结尾

提示：Xpath为强大的语言，那是因为它有非常灵活定位策略；

思考

Xpath有那些策略呢？

2. Xpath定位策略(方式)

1. 路径-定位
   1). 绝对路径
   2). 相对路径
2. 利用元素属性-定位
3. 层级与属性结合-定位
4. 属性与逻辑结合-定位

Xpath定位 方法

```
driver.find_element_by_xpath()
```

2.1 路径(绝对路径、相对路径)

```
绝对路径：从最外层元素到指定元素之间所有经过元素层级路径 ；如:/html/body/div/p[2]
            提示：
                1)．绝对路径以/开始
                2)．使用Firebug可以快速生成，元素XPath绝对路径

相对路径：从第一个符合条件元素开始(一般配合属性来区分)；如：//input[@id='userA']
            提示：
                1)．相对路径以//开始
                2)．使用Friebug扩展插件FirePaht可快速生成，元素相对路径

提示：为了方便练习Xpath，可以在FireBug内安装扩展插件-FireFinder插件；
      1)．火狐浏览器-->组件管理器-->搜索FireFinder
```

使用Xpath实现 案例-1

```
需求：
    1)．使用绝对路径和相对路径分别实现，账号A：admin;密码A：123456；自动化脚本设计
```

2.2 利用元素属性

```
说明：快速定位元素，利用元素唯一属性；
示例：//*[@id='userA']
```

2.3 层级与属性结合

```
说明：要找的元素没有属性，但是它的父级有；
示例：//*[@id='p1']/input
```

2.4 属性与逻辑结合

```
说明：解决元素之间个相同属性重名问题
示例：//*[@id='telA' and @class='telA']
```

2.5 Xpath-延伸

```
//*[text()="xxx"]                                  文本内容是xxx的元素

//*[starts-with(@attribute,'xxx')]                    属性以xxx开头的元素

//*[contains(@attribute,'Sxxx')]                    属性中含有xxx的元素
```

2.6 Xpath-总结

```
1. 如何通过Friebug快速生成绝对路径
2. 如果通过Friebug快速生成相对路径
3. Xpath策略有那些
```

---

3. CSS定位

3.1 什么是CSS?

```
1. CSS（Cascading Style Sheets）是一种语言，它用来描述HTML和XML的元素显示样式；
   (css语言书写两个格式：
                            1. 写在HTML语言中<style type="text/css">...
                            2. 写在单独文件中 后缀.css
       )
2. 而在CSS语言中有CSS选择器(不同的策略选择元素)，在Selenium中也可以使用这种选择器；
提示：
    1. 在selenium中极力推荐CSS定位，因为它比XPath定位速度要快
    2. css选择器语法非常强大，在这里我们只学习在测试中常用的几个
```

CSS定位 方法

```
driver.find_element_by_css_selector()
```

3.2 CSS定位常用策略 (方式)

```
1. id选择器
2. class选择器
3. 元素选择器
4. 属性选择器
5. 层级选择器
```

id选择器

```
说明：根据元素id属性来选择
格式：#id 如：#userA <选择id属性值为userA的所有元素>
```

使用CSS实现 案例-2

```
需求：
    1). 使用CSSid定位实现，账号A：admin;密码A：123456；自动化脚本设计
```

class选择器

```
说明：根据元素class属性来选择
格式：.class 如：.telA <选择class属性值为telA的所有元素>
```

元素选择器

```
说明：根据元素的标签名选择
格式：element 如：input <选择所有input元素>
```

属性选择器

```
说明：根据元素的属性名和值来选择
格式：[attribute=value] 如：[type="password"] <选择所有type属性值为password的值>
```

层级选择器

```
说明：根据元素的父子关系来选择
格式：element>element 如：p>input <返回所有p元素下所有的input元素>
提示：> 可以用空格代替 如：p input 或者 p [type='password']
```

3.3 CSS延伸

```
1. input[type^='p'] 说明：type属性以p字母开头的元素
2. input[type$='d'] 说明：type属性以d字母结束的元素
3. input[type*='w'] 说明：type属性包含w字母的元素
```

3.4 CSS总结

选择器	例子	描述

#id	#userA	id选择器，选择id="userA"的所有元素
.class	.telA	class选择器，选择class="telA"的所有元素
element	input	选择所有input元素
[attribute=value]	[type="password"]	选择type="password"的所有元素
element>element	p>input	选择所有父元素为p元素的input元素

4. XPath与CSS类似功能对比

定位方式	XPath	CSS
元素名	//input	input
id	//input[@id='userA']	#userA
class	//*[@class='telA']	.telA
属性	1. //※[text()="xxx"] 2. //※[starts-with(@attribute,'xxx')] 3. //※[contains(@attribute,'xxx')]	1. input[type^='p'] 2. input[type$='d'] 3. input[type*='w']

说明：由于显示排版原因以上所有(※)号代替(*)

5. 八种元素定位总结：

```
1. id
2. name
3. class_name
4. tag_name
5. link_text
6. partial_link_text
7. Xpath
8. Css
```

说明：

1). 元素定位我们就学到这里了

2). WebDriver除了提供以上定位API方法(driver.find_element_by_xxx())外，还提供了另外一套写法；

3). 调用find_element()方法，通过By来声明定位的方法，并且传入对应的方法和参数（了解-熟悉即可）

6. 定位(另一种写法)-延伸【了解】

说明：第二种方法使用By类的封装的方法,所以需要导入By类包

6.1 导入By类

导包：from selenium.webdriver.common.by import By

6.2 By类的方法

方法：find_element(By.ID,"userA")
　　备注：需要两个参数，第一个参数为定位的类型由By提供，第二个参数为定位的具体方式
示例：

```
1. driver.find_element(By.CSS_SELECTOR,'#emailA').send_keys("123@126.com")
2. driver.find_element(By.XPATH,'//*[@id="emailA"]').send_keys('234@qq.com')
3. driver.find_element(By.ID,"userA").send_keys("admin")
4. driver.find_element(By.NAME,"passwordA").send_keys("123456")
5. driver.find_element(By.CLASS_NAME,"telA").send_keys("18611111111")
6. driver.find_element(By.TAG_NAME,'input').send_keys("123")
7. driver.find_element(By.LINK_TEXT,'访问 新浪 网站').click()
8. driver.find_element(By.PARTIAL_LINK_TEXT,'访问').click()
```

6.3 find_element_by_xxx()和find_element() 区别

说明：通过查看find_element_by_id底层实现方法，发现底层也是调用的By类方法进行的封装：

```
def find_element_by_id(self, id_):
"""Finds an element by id.

:Args:
 - id\_ - The id of the element to be found.

:Usage:
   driver.find_element_by_id('foo')
"""
return self.find_element(by=By.ID, value=id_)
```

总结：虽然方法一样，但WebDriver推荐 find_element_by_xxx()这种方法