# 起源：

<http://www.testclass.net/selenium_python/selenium-history/>

Jason Huggins在2004年发起了Selenium项目，所有被测试的浏览器都支持Javascript。

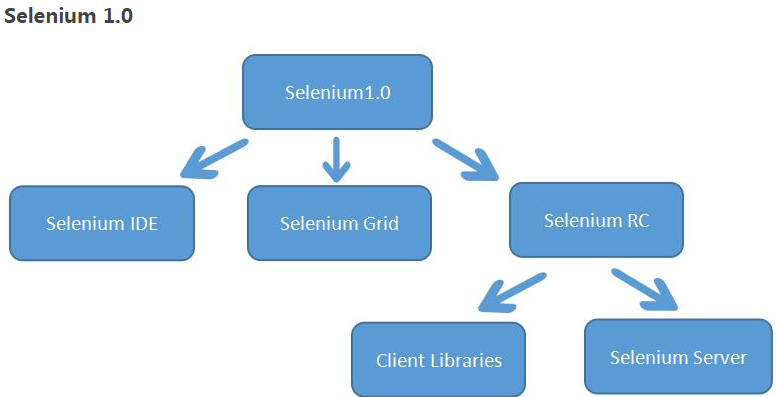
用Javascript编写一种测试工具来验证浏览器页面的行为；

这个JavaScript类库就是Selenium core，同时也是selenium RC、Selenium IDE的核心组件，Selenium由此诞生。

selenium RC、Selenium IDE的核心组件是selenium core ，core是验证浏览器页面行为的JS类库

# Selenium 1.0

Selenium 1.0 = Selenium IDE + Selenium Grid + Selenium RC



## Selenium IDE

### 简介：

**（辅助工具，基于Firefox的插件，可以提供录制、回放的功能）**

最为简单，对象是被参考

快速帮助初学者进入熟悉selenium

支持Java ，C ，python ，ruby

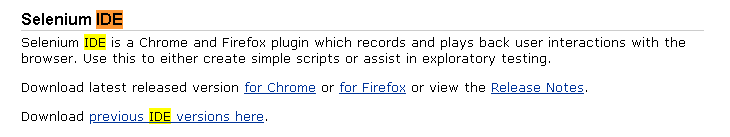
生成支持RC 和web driver 的API

录制，修改参数，生成脚本，导出熟悉的语言，回放

### 安装IDE：

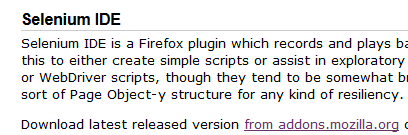
<https://www.seleniumhq.org/download/#selenium_ide>

**2018年4月份有了更新，有chrome版本的了（20180510：不知道以前有没有），Firefox也有了更加新的版本。如下 ：**



废弃：

2、在1中如下页面跳转：<https://addons.mozilla.org/en-US/firefox/addon/selenium-ide/>

，

3、Firefox版本高，selenium版本低。

下载旧的Firefox，如37。<http://ftp.mozilla.org/pub/firefox/releases/>

4、重复步骤2，安装重启，在工具菜单下显示；或者在工具栏中有图标；

### 使用IDE：

target +conmmand+value

#### 录制时，某个控件无法识别：

方法1.用键盘的TAB切换到焦点

方法2. 如当内嵌了<iframe>标签，相当于内嵌了另一个页面。非标准的控件。

[先点代码，切换为<textarea>，然后识别控件。19分15秒。](http://study.163.com/course/courseLearn.htm?courseId=1003096013#/learn/video?lessonId=1003523032&courseId=1003096013)

#### 登录后，报错，element link =XXXX not foun

时间差，需要加入等待。

加一条：waitXXX ，如conmmand: waitForElementPresent， target:link=注销

或者直接等待要验证（assert）的元素

#### 自动化操作-》自动化测试，加入判断

[差不多7分钟开始](http://study.163.com/course/courseLearn.htm?courseId=1003096013#/learn/video?lessonId=1003526032&courseId=1003096013)

加入command： assertXXX， 如conmmand: assertElementPresent，登录成功的特征

#### assertText，行号改变了，导致报错

assertText

### IDE无法解决的问题：

1、需要if判断的时候

2、需要循环的时候

3、从外部数据源获取时（xml ，数据库，excel）

4、灵活的断言（包含，以XX开始/结束等等），只有死的断言方式

5、进行灵活的结构设计（如需要框架管理几千条用例 ）

6、需要灵活、健壮、维护性高。

还有很多无法解决的问题。

## 二、Selenium RC，

Remote Control =远程控制

远程代理，代理服务器

服务器S《------RCServer-------》浏览器B

**基于JavaScript方式来远程输入，来操作html元素。**

最开始的框架，后来慢慢被webdriver取代。

很多时候RC和webdriver混用。

### 安装、运行：

RC server（seleniumhq.org ，**selenium-server-standalone-XXX.JAR**） JDK，eclipse，

运行RC Server（跨平台），

java –jar selenium-server-standalone-XXX.JAR port 4445

在eclipse里调试测试脚本。

新建java项目，src下新建包，拖入导出的java脚本

引入jar包，Junit库。（项目-属性-java build path- libraries ， 1、add jars 2. add library）

[导出的代码优化：](http://study.163.com/course/courseLearn.htm?courseId=1003096013#/learn/video?lessonId=1003525037&courseId=1003096013)

# Selenium 2.0

Selenium 2.0 = Selenium 1.0 + WebDriver

## 三、WebDriver

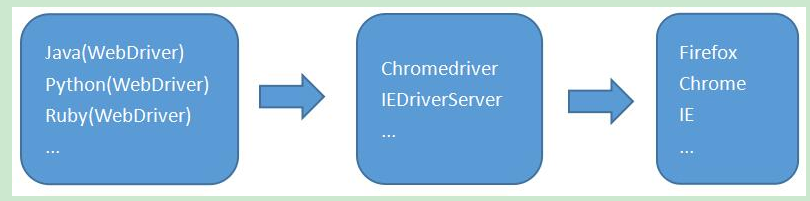
WebDriver是通过原生浏览器支持或者浏览器扩展来直接控制浏览器。

WebDriver针对各个浏览器而开发，取代了嵌入到被测Web应用中的JavaScript。

与浏览器紧密集成，因此支持创建更高级的测试，避免了JavaScript安全模型导致的限制。

除了来自浏览器厂商的支持之外，WebDriver还利用操作系统级的调用，模拟用户输入。

需要强调的是，在Selenium 2.0中主推的是WebDriver，可以将其看作Selenium RC的替代品。



<http://www.testclass.net/selenium_python/install-selenium/>

如下

### 安装Python

安装时勾选add to path

(D:\Program Files\Python\Python37\Scripts\;D:\Program Files\Python\Python37\;)

### 安装selenium

**pip install selenium**

#### pip下载的内容：

Windows是python目录下Lib\site-packages\;

Linux是/usr/local/lib/python/dist-packages/

### 下载浏览器驱动并添加路径到环境变量

<https://sites.google.com/a/chromium.org/chromedriver/home>

（如chromedriver.exe）

### 测试例子1

打开一款Python编辑器，默认Python自带的IDLE也行。创建 baidu.py文件，输入以下内容：

from selenium import webdriver

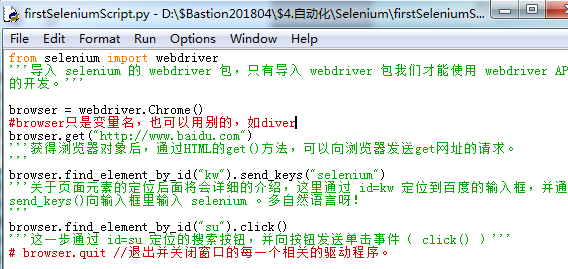
driver = webdriver.Chrome()

driver.get('https://www.baidu.com')

print(driver.title)

driver.quit()

### 测试例子2：



### 报错：

1.Chrome版本

2.没有chromedriver

3. python 的 chromedriver' executable needs to be in PATH.

<https://blog.csdn.net/dreaming_terry/article/details/78323495>

4.

# Selenium 3.0

Selenium 3.0 = Selenium 2.0 - Selenium RC（Remote Control）

直到2016年7月，Selenium3.0悄悄发布第一个beta版。

2、Selenium3.0只支持Java8版本以上。

3、Selenium3.0中的Firefox浏览器驱动独立了，以前装完selenium2就可以驱动Firefox浏览器了，现在和Chrome一样，必须下载和设置浏览器驱动。

4、MAC OS 集成Safari的浏览器驱动。默认在/usr/bin/safaridriver 目录下。

5、只支持IE 9.0版本以上。

# 原理: webdriver 原理：

1. WebDriver 启动目标浏览器，并绑定到指定端口。该启动的浏览器实例，做为 web driver 的 remote server。

2. Client 端通过 CommandExcuter 发送 HTTPRequest 给 remote server 的侦听端口（通信协议： the webriver wire protocol）

3. Remote server 需要依赖原生的浏览器组件（如：IEDriverServer.exe、chromedriver.exe），来转化转化浏览器的 native 调用。

被打开的浏览器成了一个http server.这个server的默认端口是7055,

此后它开始响应client端(我们的Java代码)的各种对浏览器操作行为的请求.

**同一个浏览器驱动即可以处理Java语言的脚本，也可以处理Python语言的脚本呢？**

这就要提到WebDriver基于的协议：JSON Wire protocol。

JSON Wire protocol是在http协议基础上，对http请求及响应的body部分的数据的进一步规范。