目　录

致谢

当前文档 《Selenium IDE 官方文档翻译》 由 进击的皇虫 使用 书栈(BookStack.CN) 进行构建，生成于 2018-05-28。

书栈(BookStack.CN) 仅提供文档编写、整理、归类等功能，以及对文档内容的生成和导出工具。

文档内容由网友们编写和整理，书栈(BookStack.CN) 难以确认文档内容知识点是否错漏。如果您在阅读文档获取知识的时候，发现文档内容有不恰当的地方，请向我们反馈，让我们共同携手，将知识准确、高效且有效地传递给每一个人。

同时，如果您在日常生活、工作和学习中遇到有价值有营养的知识文档，欢迎分享到 书栈(BookStack.CN) ，为知识的传承献上您的一份力量！

如果当前文档生成时间太久，请到 书栈(BookStack.CN) 获取最新的文档，以跟上知识更新换代的步伐。

文档地址：http://www.bookstack.cn/books/selenium-ide-doc

书栈官网：http://www.bookstack.cn

书栈开源：https://github.com/TruthHun

分享，让知识传承更久远！ 感谢知识的创造者，感谢知识的分享者，也感谢每一位阅读到此处的读者，因为我们都将成为知识的传承者。

前言

简介

Selenium IDE 是一款 web UI 自动化测试工具，是基于录制和回放操作的火狐浏览器的插件。本教程是官方资料的翻译。

如果不喜欢阅读文字教程，可以尝试学习 Selenium IDE web 自动化测试视频课程，并完成相应的通关任务。

来源(书栈小编注)

> *https://github.com/wangding/selenium-ide-doc*

简介

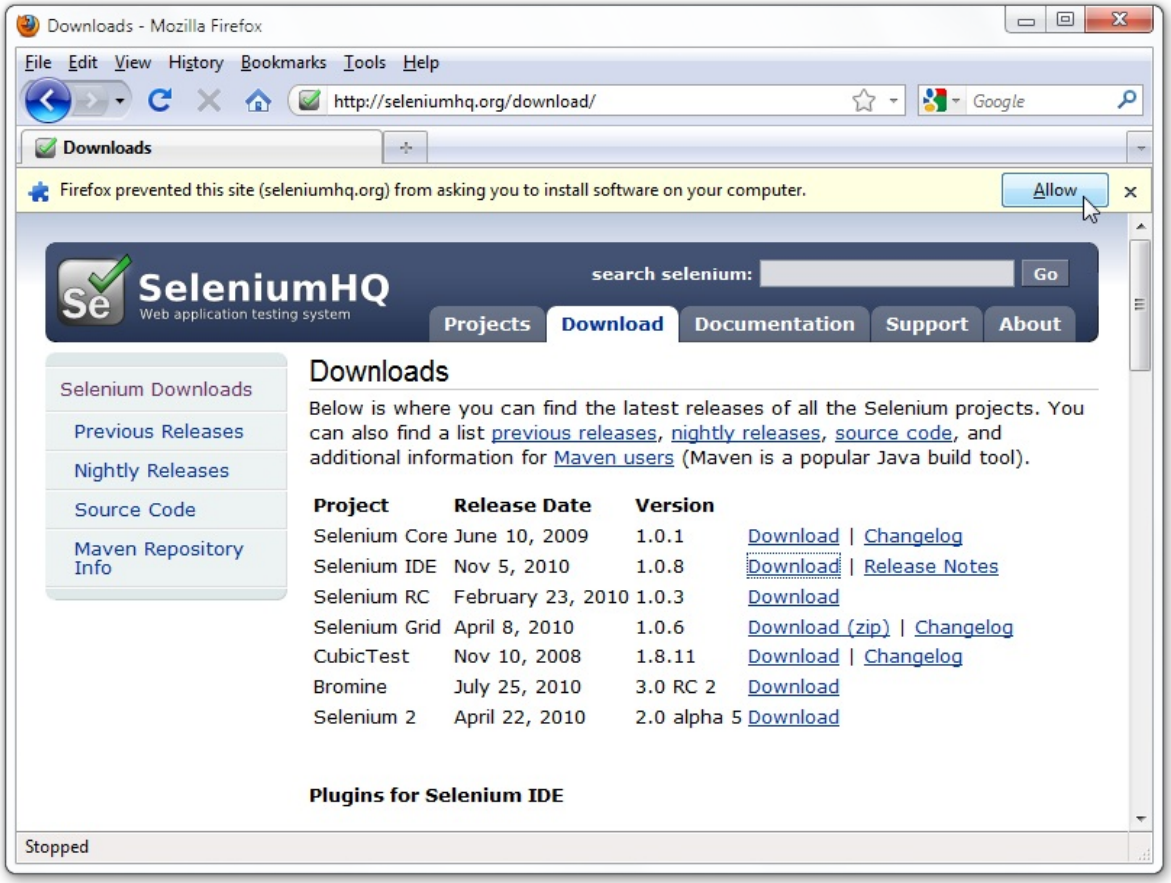
Selenium IDE（Integrated Development Environment，集成开发环境）是用来开发 Selenium 测试用例的工具。它是一个易于使用的火狐浏览器插件，通常是开发测试用例最有效的方法。它还包含一个上下文菜单，允许你首先从浏览器当前显示的页面中选择一个 UI 元素，然后从 Selenium 的命令列表中选择命令，依据网页中选择的 UI 元素的上下文使用预定义的参数。这不仅节省时间，而且是学习 Selenium 脚本语法的一个好方法。

这个章节专门介绍 Selenium IDE ，以及如何高效使用这个工具。

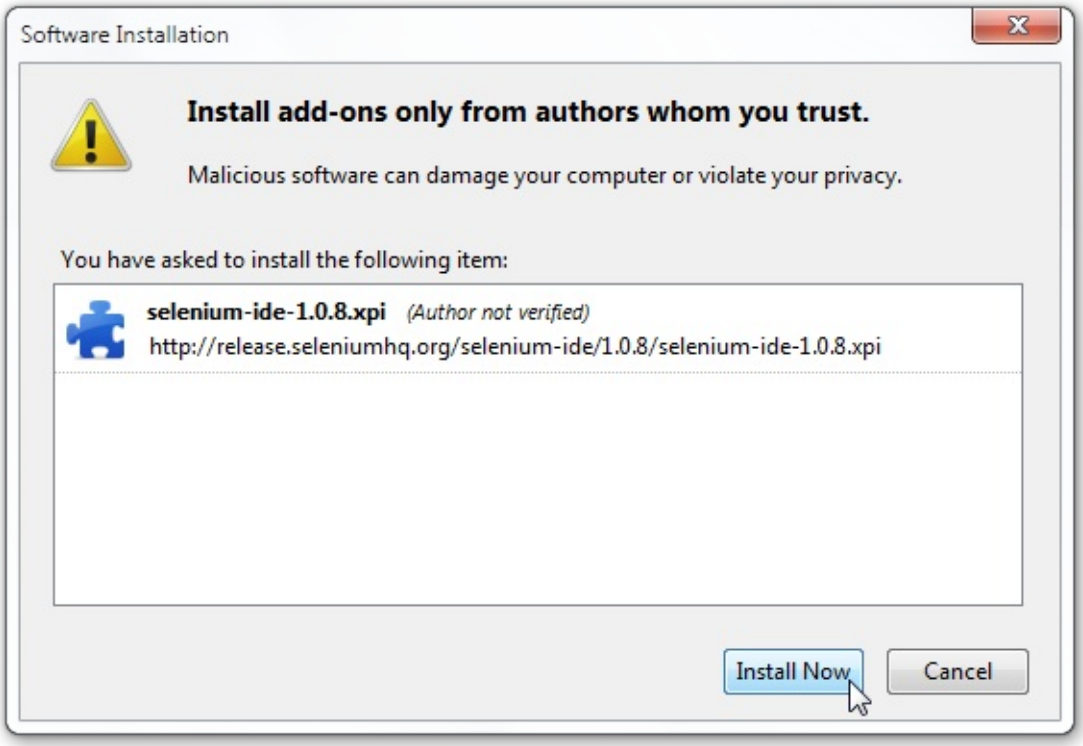
安装 Selenium IDE

首先，用火狐浏览器从 SeleniumHQ 下载页面下载 IDE。

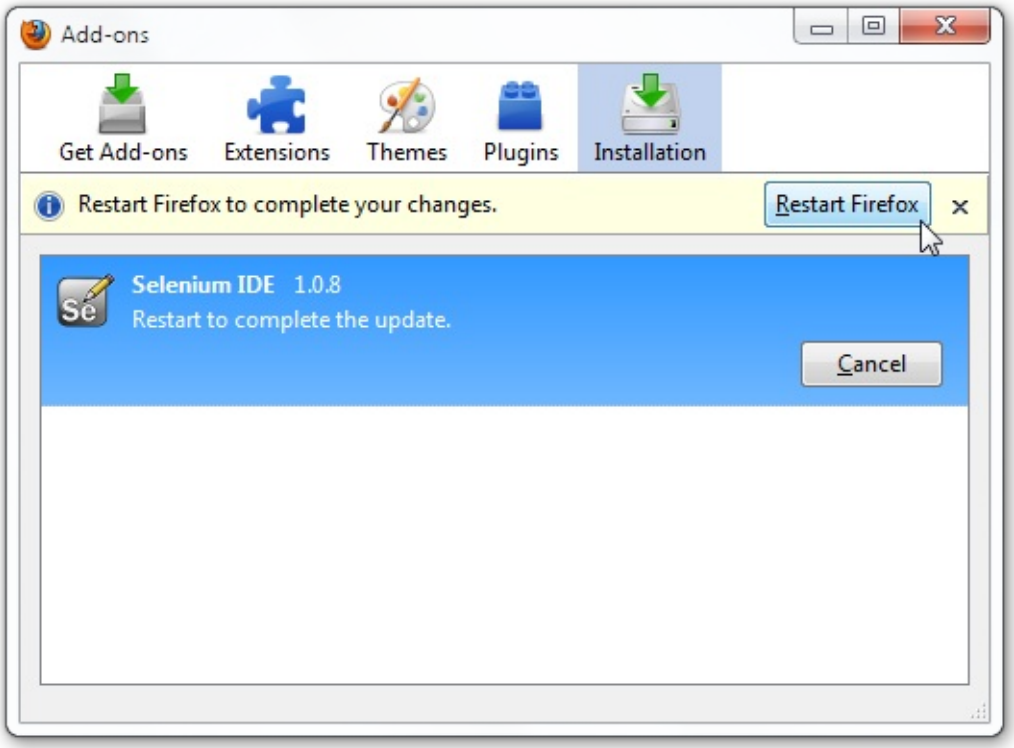
出于保护的目的，火狐浏览器会阻止你从不熟悉的地点安装插件，你需要点击“Allow”来继续安装，如下图所示。



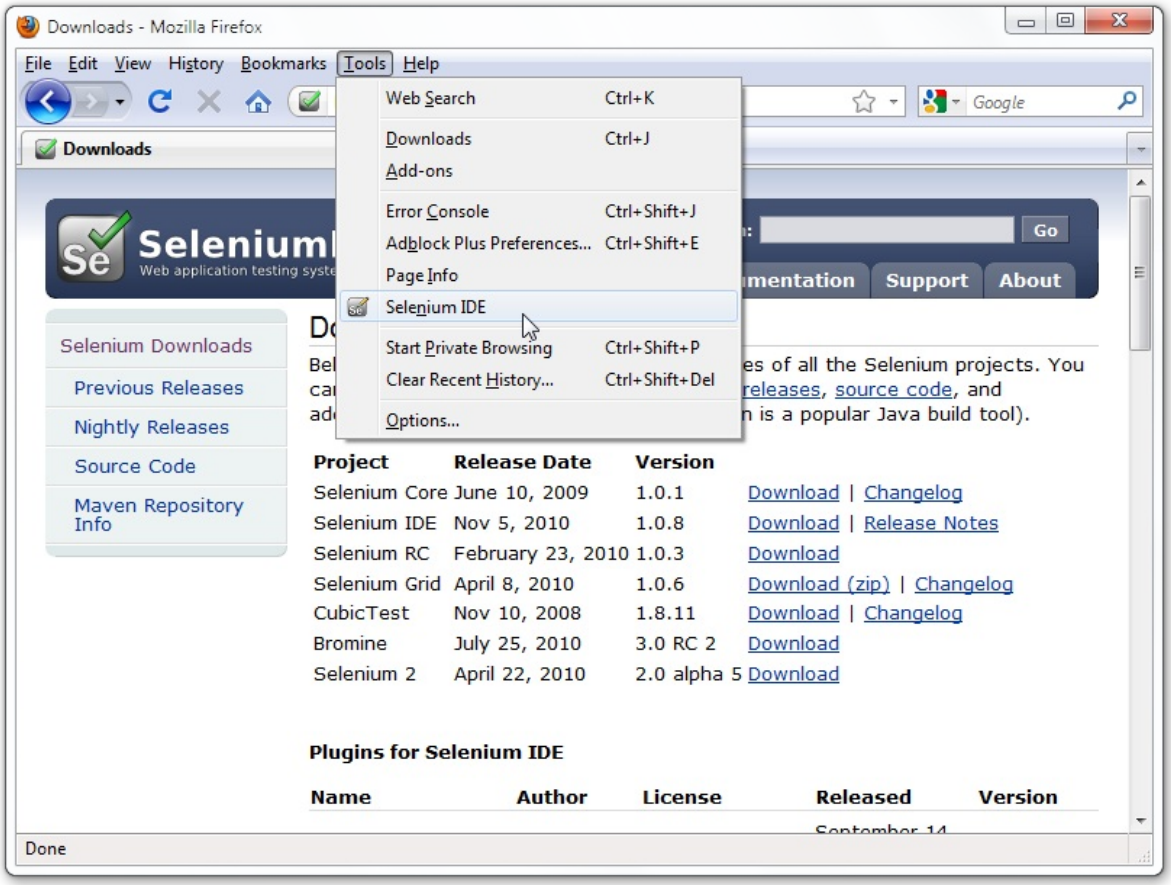
当火狐浏览器下载插件的时候，会出现下面的窗口。



点击“Install Now”。火狐浏览器会弹出组件管理窗口，首先显示一个进度条，然后下载结束，如下图所示。

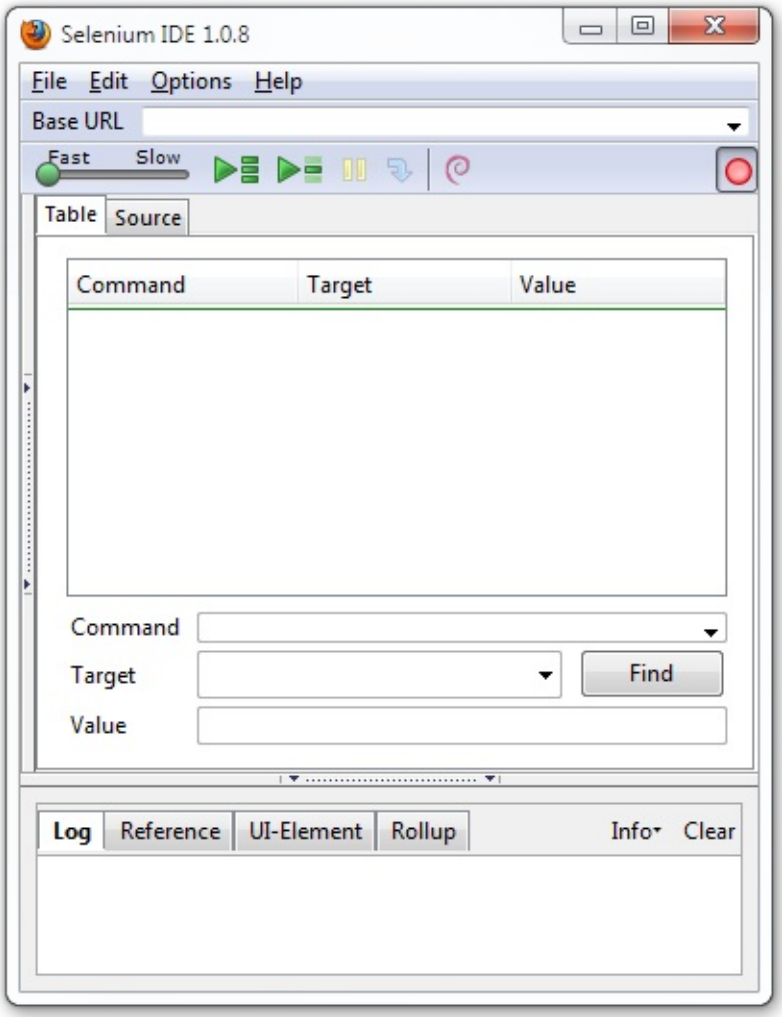


重启火狐浏览器，然后你会在浏览器的工具菜单下面找到刚安装好的 Selenium IDE 。


Downloads - Mozilla Firefox
File Edit View History Bookmarks Tools Help
Web Search Ctrl+K
Downloads Ctrl+J
Add-ons
Error Console Ctrl+Shift+J
Adblock Plus Preferences... Ctrl+Shift+E
Page Info
Selenium IDE
Start Private Browsing Ctrl+Shift+P
Clear Recent History... Ctrl+Shift+Del
Options...
Downloads
Selenium
Web application testing syste
Go
mentation
Support
About
Selenium Downloads
Previous Releases
Nightly Releases
Source Code
Maven Repository Info
releases, source code, and
is a popular Java build tool).
Project Release Date Version
Selenium Core June 10, 2009 1.0.1 Download | Changelog
Selenium IDE Nov 5, 2010 1.0.8 Download | Release Notes
Selenium RC February 23, 2010 1.0.3 Download
Selenium Grid April 8, 2010 1.0.6 Download (zip) | Changelog
CubicTest Nov 10, 2008 1.8.11 Download | Changelog
Bromine July 25, 2010 3.0 RC 2 Download
Selenium 2 April 22, 2010 2.0 alpha 5 Download
Plugins for Selenium IDE
Name Author License Released Version
Done

启动 Selenium IDE

启动 Selenium IDE 非常容易，只要点击火狐浏览器的工具菜单，在工具菜单或者工具菜单下的 web 开发者子菜单下找到 Selenium IDE 菜单项，点击即可。Selenium IDE 启动之后，会显示一个空白脚本编辑的窗口，窗口上有菜单栏，可以创建新的测试用例。



IDE 的功能

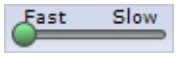
菜单栏

文件菜单用来管理测试用例和测试套件（一组测试案例）。通过文件菜单可以添加一个新的测试用例，打开测试用例，保存测试用例，将测试用例导出成某种编程语言的代码文件。打开最近的测试用例。所有这些操作也适用于测试套件。

编辑菜单允许对测试用例中被选择的命令进行复制、粘贴、删除、取消，等操作。Options 菜单允许改变设置。你可以设置某些命令的超时值，添加用户自定义扩展来扩展 Selenium 的基本命令，指定保存测试用例所适用的格式（即：编程语言）。帮助菜单是火狐浏览器的标准菜单。

工具栏

工具栏上的按钮用来控制测试用例的执行，包括：单步执行的功能用来调试测试用例。最右边的一个红点按钮，是录制测试脚本的。



速度控制：控制测试用例运行的速度。

全部运行：运行整个测试套件，当一个测试套件中加载了多个测试用例时。

运行：运行当前选中的测试。当只加载了一个测试用例时，这个按钮和全部运行按钮效果相同。

暂停/继续：允许停止和重新启动运行测试用例。

单步执行：允许你单步执行测试用例，即，一次运行测试用例中的一个命令。用于调试的测试用例。

测试运行模式：这种高级特性允许将一系列的Selenium命令组合成一个单一动作来重复执行。Rollup rules的详细文档在帮助菜单的UI-Element中查看。

应用回滚规则：这个高级功能允许一组 Selenium 命令作为一个动作来执行。关于回滚规则的详细文档，请参考帮助菜单的 UI-元素 文档。

录制：录制用户在浏览器中的操作。

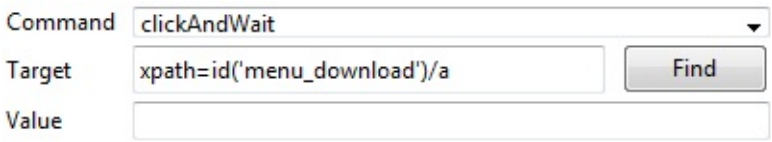
测试用例面板

自动化测试脚本显示在测试用例面板中。它有两个选项卡，一个是表格选项卡，以容易阅读的表格形式来显示命令和参数。

Command	Target	Value
open	/	
waitForPageToLoad		
clickAndWait	xpath=id('menu_download')/a	
assertTitle	Downloads	
verifyText	xpath=id('mainContent')/h2	Downloads

另一个是源选项卡，显示存储测试用例文件的原生格式。默认情况下，测试用例的原生文件格式是 HTML，当然可以改变成其他编程语言，如 Java 或 C#，或像 Python 这样的脚本语言。有关详细信息，请参阅 Options 菜单。源视图允许在原代码状态对测试用例进行编辑，包括复制、剪切和粘贴操作。

Command（命令），Target（目标）和 Value（值），三个字段用来显示当前选中命令及其参数。你可以很方便的修改当前选中命令的这三个参数。窗口下方的参考面板中会显示选中命令的完整参考信息，一般来说命令的第一个参数通常都是目标字段。如果在参考面板中显示的命令有第二个参数，则它通常都是值字段。



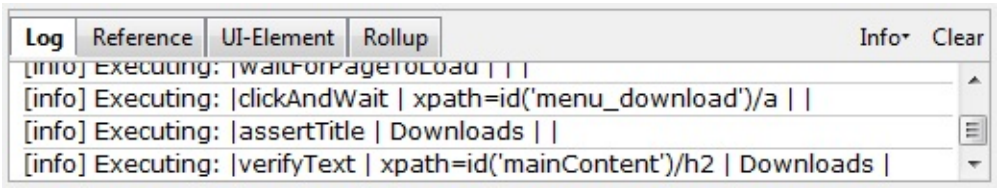
如果你开始在命令字段键入命令，根据你键入的字符，会出现一个待选的命令下拉列表，您可以从中选择您想要命令。

日志/参考/UI元素/回滚 窗格

底部窗格有四个不同的选项卡，分别是：Log（日志），Reference（参考），UI-Element（UI元素）和 Rollup（回滚）。到底显示哪个窗格，取决于被点选的选项卡是哪个。

日志窗格

运行测试用例时，即使你不首先选择日志选项卡，错误消息和正常消息随着测试案例的执行信息会自动显示在此窗格。这些信息通常用于调试测试用例。注意 Clear 按钮用来清除日志。还有 Info 按钮下拉列表允许你选择不同级别的日志信息。



参考窗格

当我们在表格视图下，输入或修改 Selenese 命令或参数时，默认会切换到参考选项卡。在表格视图中，参考面板将显示当前命令的文档。无论在表视图还是在源视图中，当我们输入或修改命令时，确保命令参数中的目标和值字段与参考面板中提示的参数列表信息相匹配，这是至关重要的。参数的数量必须匹配，参数的顺序必须匹配，而且参数的类型必须匹配。如果上述三个方面有任何不匹配之处，命令将不能正确执行。

UI-元素和回滚窗格

这两个窗格的详细信息（这涉及到高级功能）请在 Selenium IDE 的帮助菜单的 UI-元素 文档中查看。

静态内容测试

你应该测试应用程序的哪些部分？这取决于项目的具体情况，包括：用户期望，项目周期，项目经理对项目优先级的设定，等。一旦定义好了项目边界，测试工程师肯定会对测试工作做出多方面的决策。

我们��在下文提到 web 应用测试的几种类型。这些术语绝不是标准，但是这些概念在 web 应用测试中确实是非常典型的。

静态内容测试

内容测试，这是最简单的测试类型，只需要测试一个静态的，不变的，UI元素出现在页面上。例如：

- 每个网页有预期的页面标题吗？这可以用于验证点击链接后的预期页面。
- web 应用主页面的顶部是否包含一个图片？
- 网站的每个页面是否包含一个页脚区，该区域包含公司的联系页面，法律政策和商标信息页面的链接？
- 每个页面开头的标题文字是否使用 h1 标记？并且，每个页面是否有正确的标题文本？

测试内容对于你来说有可能需要，也可能没必要。如果你的页面内容很少改动或者不太重要，那么手工测试页面内容就足够了。假如 web 应用包含的文件被移动到不同的位置，内容测试可能就会很有价值。

链接测试

网站经常的错误来源是链接失效或链接的页面丢失。这种测试工作包括点击每个链接并验证预期页面。如果是静态链接并且很少改变，手工测试可能就足够了。但是如果网页设计师经常改变链接，或者偶尔发生文件迁移，链接测试应该实现自动化。

功能测试

测试应用程序的特定功能，需要某种类型的用户输入，并返回某种类型的结果。通常功能测试会涉及多个页面，有基于表单的输入页面，包含多个输入字段的集合，以及提交和取消操作，有一个或多个

响应页面。用户可以通过文本字段、复选框、下拉列表、或任何其他浏览器支持的控件进行输入。

功能测试通常是自动化测试中最复杂的，但通常是最重要的。典型的测试可以是登录操作，账户注册，用户账户操作，修改帐户设置，复杂的跨域数据检索，等等。功能测试通常对应于特定的用户场景，而用户场景是对应用程序功能和特性的描述。

测试动态元素

通常一个 web 页面元素都有一个唯一的标识符用于在页面中定位该元素。通常是通过 html 标记的 ��id� ﹐属性或 name 属性来实现的。这些名字可以是静态的，即，不变的字符串常量。它们也可以是每个页面动态生成的值。例如，一些 web 服务器可能在一个页面实例上为显示的文档命名为 doc3861 ，用户检索的另一个页面实例为该文档命名为 ��doc6148a。一个验证文档存在的测试脚本可能没有一个统一的标识符用于定位文档。通常，结果页面上不同标识符指定的动态元素往往基于用户的动作。这当然取决于 web 应用的功能。

这里有一个例子:

```
1. <input type="checkbox" value="true" id="addForm:_ID74:_ID75:0:_ID79:0:
2.     checkBox"/>
```

Ajax 测试

Ajax 是一种技术，来支持动态变化的用户界面元素，界面元素可以动态更改而无需浏览器重新加载页面，例如: 动画、RSS 种子和实时数据更新。有无数方法可以使用 Ajax 更新 web 页面上的元素。但是，最简单的方法是在 ajax 开发的应用程序中，从应用程序服务器检索数据，然后显示在页面上，而不用重新加载整个页面。只有一部分的页面或者仅仅是元素本身被重新加载。

测试设计介绍 | 目录 | 验证测试结果

制作测试案例

开发测试案例有三个主要的方法：录制、利用上下文菜单添加验证和断言以及编辑和修改测试案例。通常自动化测试工程师应该同时掌握这三种方法。

录制

很多新手都是从录制一个与网站交互的测试用例，开始学习自动化测试的。当第一次打开 Selenium IDE 时，录制按钮默认是启动的。如果你不想要 Selenium IDE 自动开始录制，在系统设置中可以关掉。具体操作如下：点击 Options 菜单 > Options… 菜单项，打开系统设置对话框，取消勾选 “Start recording immediately on open.”

录制期间，Selenium IDE 会根据你的操作自动在测试用例中插入命令。通常，这将包括：

- 点击一个链接，click 或 clickAndWait 命令
- 输入值，type 命令
- 从一个下拉列表中选择选项，select 命令
- 点击复选框或单选按钮，click 命令

这里有一些陷阱需要注意：

- type 命令需要点击网页的某些区域的才能录制。
- 链接跳转通常录制成 click 命令。你会经常需要把 click 命令改为 clickAndWait，来确保您的测试用例会暂停，直到新页面加载完成。否则，您的测试用例将在页面加载新的 UI 元素之前继续运行命令。这将会导致意想不到的测试用例运行失败。

通过上下文菜单添加断言和验证

测试用例通常需要检查网页的属性，这需要用到断言和验证命令。我们不会描述这些命令的细节，在 Selenium 的命令 Selenese 章节中有详细介绍。这里我们会简单地描述如何将它们添加到测试用例中。

在 Selenium IDE 录制时，在浏览器显示被测应用程序的页面上右键单击页面上的任何地方。您将

看到一个上下文菜单显示验证 和/或 断言命令。

第一次使用 Selenium 时，在右键快捷菜单中 Selenium 能只列出几个命令。当你使用 IDE 一段时间后，您会发现额外的命令很快就会被添加到这个菜单上。Selenium IDE 将试图预测当前网页上您需要选择的 UI 元素会用到的命令以及参数。

让我们看看这是如何工作的。打开一个网页，选择页面上的一块文本。这个文本可以是一个段落或标题，他都会正常工作。现在，右击选中的文本，上下文菜单中应该给你一个 verifyTextPresent 命令和建议参数应该文本本身。

同时，注意到所有可用命令的菜单选项。这表明更多的命令，以及建议的参数，可以用来测试你当前选择的 UI 元素。

尝试更多的 UI 元素。右击图片，或用户控件按钮或复选框。您可能需要使用显示所有可用的命令来查看 verifyTextPresent 以外的选项。一旦你选择其他选项，更常用的将出现在一级上下文菜单。例如，为一个图片选择 verifyElementPresent 后，会导致该命令在你下一次选择一个图片并单击鼠标右键时，该命令出现在上下文菜单中。

再一次，这些命令将在 Selenium 命令章节中有详细解释。现在，请随意使用IDE来录制测试用例，从网页上右键选择命令插入到测试用例中，然后运行它。通过不断体验IDE，你可以学到很多关于 Selenium 的命令。

编辑

插入命令

表格视图

首先，选择要插入命令的位置。在测试用例窗格中，左键单击一条命令，新插入的命令将会出现在这条命令前面。右键单击该命令并在上下文菜单中选择插入命令，IDE 将在选定的命令前面添加一个空白行。然后，在命令字段编辑域文本框中输入新命令及其参数。

源视图

首先，选择要插入命令的位置。在测试用例窗格中，在要插入的两命令之间点击鼠标左键，输入 HTML 标记，创建一个包含三列的表格行。三列分别对应命令（第一列），第一个参数（第二列），和第二个参数（第三列）。例如：

```
1. <tr>
2.     <td>Command</td>
3.     <td>target (locator)</td>
```

```
    <td>Value</td>
</tr>
```

插入注释

注释可以添加到测试用例中，来增加测试案例的可读性。运行测试用例时，这些注释会被忽略。

通过添加空白的注释，注释还可以用来在测试案例中增加垂直方向的空白（一个或多个空行）。一个空的命令在执行时会报错，而空白注释却不会。

表格视图

在测试用例中选择要插入注释的行。右键单击并选择插入注释。在命令字段中，输入注释。紫色的注释文本会出现在测试案例中。

源视图

在测试用例中选择要插入注释的行。添加一个 HTML 样式的注释，例如：`<!— 这里是注释 —>` 。

编辑命令或注释

表格视图

选择要修改或编辑的行，然后编辑命令、目标和值这三个字段。

源视图

源视图提供了相当于一个 WYSIWYG （所见即所得）的编辑器，只需修改相应行的命令、参数或注释即可。

打开和保存测试案例

与很多程序类似，Selenium IDE 的文件菜单也有保存和打开命令。然而，Selenium 对测试用例和测试套件是分别对待的。把 Selenium IDE 制作的测试案例保存下来，可以方便日后重复使用，当然，你可以选择保存成单个的测试案例或者保存成测试套件。如果测试套件中的测试用例还没有保存，Selenium IDE 会提示在保存测试套件前先保存测试案例。

当你打开一个已存在的测试案例或套件，Selenium IDE 会在测试用例面板中显示测试案例中的命令。

运行测试案例

Selenium IDE 有很多不同的方式来运行测试案例。你可以一次完整的运行整个测试案例，或者暂停运行并继续运行，或者一次执行一行，或者执行正在开发的这个命令，或者通过测试套件批量运行若干测试案例。在 Selenium IDE 中运行测试案例非常灵活和方便。

运行一个测试案例
单击运行按钮，可以运行当前显示的测试用例。

运行一个测试套件
单击运行全部按钮，可以运行当前加载的测试套件中的所有测试案例。

暂停和开始
暂停按钮可以用来中止运行中的测试案例。随后，这个按钮的图标会变成继续执行。点击此按钮，测试案例将继续执行。

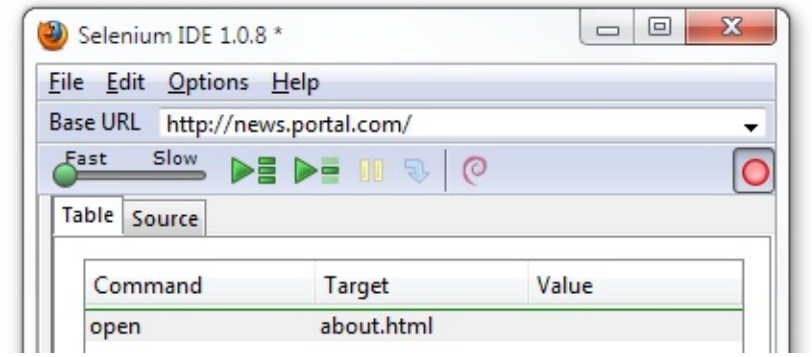
运行中暂停
可以在测试案例中设置一个断点，测试案例会在断点命令处暂停执行。这个功能在调试时非常有用。设置断点的方法是，在需要中断的命令上，单击鼠标右键，从上下文菜单中选择 Toggle breakpoint。

从中间开始执行
可以让 Selenium IDE 从测试案例中间的一个特定命令处开始执行。这个功能也主要用于调试。设置起点的方法是，在需要中断的命令上，单击鼠标右键，从上下文菜单中选择 Set/Clear Start Point。

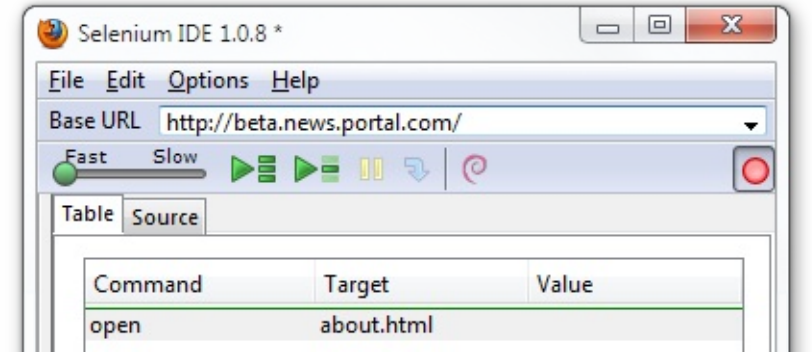
运行一个命令
双击任何一个命令来单独执行它。在编写命令时，这个功能很有用。它允许你立即测试编写的命令，尤其是当你不确定它是否正确时。您可以双击它，看它是否运行正确，也可以从上下文菜单中操作。

使用基址运行测试案例

Selenium IDE 窗口顶部的 Base URL 字段对于测试用例运行在不同域时非常有用。假设一个名为 http://news.portal.com 的网站有一个名为 http://beta.news.portal.com 的内部测试网站。针对这些网站的任何测试用例，都是以 open 命令开始的，open 命令指定一个相对 URL 路径作为参数，而不是绝对 URL 路径（绝对路径是从一个协议，比如 http: 或 https: 开始的）。Selenium IDE 通过将 open 命令的参数追加到 Base URL 后面，来创建一个绝对的 URL 路径。例如，下面的测试用例将会运行在 http://news.portal.com/about.html:



同样的测试案例修改一下 Base URL 设置就可以运行在 http://beta.news.portal.com/about.html:



Selenium 命令集：Selenese

Selenium 命令通常被称为 selenese，是用来运行测试的一个命令的集合。这些命令的序列是一个测试脚本。这里将详细解释这些命令，展示使用 Selenium 测试 web 应用程序时的很多选择。

Selenium 提供了一组丰富的命令，以几乎任何你可以想象的方式，全方位测试你的 web 应用。这些命令通常被称为 selenese 命令集，这些命令基本上构造了一个测试语言。

例如，利用 selenese，你可以依据 UI 元素的 HTML 标记来测试该 UI 元素在页面上是否存在，测试页面上的具体内容，测试失效链接，输入字段，下拉列表选择，待提交的表单，表格数据，等等。此外，Selenium 命令还能测试窗口大小，鼠标位置、alerts 信息、Ajax 功能，弹出窗口，事件处理，以及许多其他 web 应用程序的特性。命令参考中列出了所有可用的命令。

命令会告诉 Selenium 做什么。Selenium 命令分三大类：Action 动作，Accessors 访问器和 Assertions 判断。

- Action 动作类命令，一般用来操作应用程序的状态。他们做的事情，类似：点击这个链接、选择那个选项。如果动作失败，或者有错误，当前测试会停止执行。

  许多动作会增加一个 AndWait 后缀，如：clickAndWait。这个后缀告诉 Selenium 动作将会导致浏览器发送请求到服务器，而 Selenium 应该等待一个新页面被完全加载。

- Accessors 访问器类命令，用来检查应用程序的状态，并将结果存储在变量中。如：storeTitle。他们也用于自动生成断言。

- Assertions 判断类命令，很像访问器类命令，但他们验证应用程序的状态是否符合预期。例如：包括 “make sure the page title is X” 和 “verify that this checkbox is checked”。

所有Selenium 判断类命令又分三种使用方式：assert 断言类，verify 验证类，和 waitFor 等待类。例如，你可以 assertText，verifyText 和 waitForText。当一个 assert 失败时，测试会中止。当一个 verify 失败时，测试将继续执行，记录下失败信息。可以用一个 assert 命令确保应用程序在正确的页面上，紧随其后的是一组 verify 判断来测试表单字段值，标签，等等。

waitFor 命令等待一些条件发生（在测试 Ajax 应用程序时，非常有用）。如果条件已经是正确的，他们马上会成功。如果条件在超时设置的范围内不成立，他们将失败并停止测试。

脚本语法

Selenium 的命令很简单，他们包括命令和两个参数。例如：

verifyText	//div//a[2]	Login

参数并不总是必需的，这取决于命令。在某些情况下，两个参数都是必需的，有些情况只需要一个参数，还有些情况可能不需要任何参数。下面有几个例子：

命令	目标	值
goBackAndWait		
verifyTextPresent	Welcome to My Home Page	
type	id=phone	(555) 666-7066
type	id=address1	${myVariableAddress}

命令参考中有每个命令参数的描述。

命令的参数虽各不相同，但是他们通常是：

- locatoer 定位器用来识别页面中的UI元素。
- text pattern 文本模式用来验证或断言页面内容。
- text pattern 文本模式或 Selenium 变量用来在文本输入域中输入文本或在选项列表中选择一个选项。

定位器、文本模式，Selenium 变量和命令本身在 Selenium 命令中有非常详细地描述。

Selenium IDE 中运行的脚本，会存储在一个 HTML 格式的文本文件中。他被包含在一个三列的 HTML 表格中。第一列是 Selenium 命令，第二列是目标，最后一列是值。第二列和第三列可能不需要，这取决于所选的 Selenium 命令，但他们应该存在。每个表格行代表一个 Selenium 命令。下面是一个脚本的例子，在这个脚本中会打开一个页面，然后断言页面标题，最后验证页面上的一些内容：

```
<table>
    <tr><td>open</td><td>/download/</td><td></td></tr>
    <tr><td>assertTitle</td><td></td><td>Downloads</td></tr>
    <tr><td>verifyText</td><td>//h2</td><td>Downloads</td></tr>
</table>
```

这个脚本文件在浏览器中呈现为表格的样子如下：

open	/download/	
assertTitle	Downloads	
verifyText	//h2	Downloads

Selenese HTML 语法可以用来编写和运行测试，而无需编程语言的知识。有了 selenese 的基本知识和 Selenium IDE 你可以快速制作和运行测试案例。

测试套件

测试套件是测试的集合。通常会在一个测试套件将所有的测试作为一个连续的批处理作业运行。

Selenium IDE 可以用一个简单的 HTML 文件来定义测试套件。语法也很简单。一个 HTML 表格定义了一个测试列表，其中的每一行定义了每个测试文件的路径。举个例子：

```
<html>
<head>
<title>Test Suite Function Tests - Priority 1</title>
</head>
<body>
<table>
  <tr><td><b>Suite Of Tests</b></td></tr>
  <tr><td><a href="./Login.html">Login</a></td></tr>
  <tr><td><a href="./SearchValues.html">Test Searching for Values</a></td></tr>
  <tr><td><a href="./SaveValues.html">Test Save</a></td></tr>
</table>
</body>
</html>
```

类似上面这个文件这将允许 Selenium IDE 逐一运行测试。

测试套件还可以用 Selenium RC 来维护。通过编程，可以有多种实现方式。如果使用 Selenium RC 和 Java，通常使用 Junit 来维护一个测试套件。此外，如果使用 C# 语言，Nunit 可以用来维护测试套件。如果使用解释型语言像 Python，Selenium RC 需要添加一些简单的编程，来建立一个测试套件。因为 Selenium RC 可以使用程序代码的编程逻辑，所以实现测试套件并不困难。

脚本语法 | 目录 | Selenium 常用命令

Selenium 常用命令

作为对 Selenium 的总结性介绍，我们将向您介绍几个常用的 Selenium 命令。这些是构建测试中最常用的命令。

open
使用 URL 打开一个页面.

click/clickAndWait
执行点击操作或者等待新页面加载完成后执行点击操作。

verifyTitle/assertTitle
检查页面的标题是否为预期的内容。

verifyTextPresent
检查页面上某处的文字是否为预期的内容。

verifyElementPresent
验证用 HTML 标签定义的，预期的 UI 元素，在页面上是否存在。

verifyText
验证预期的文本和其相应的 HTML 标记是否出现在页面上。

verifyTable
验证表格是否包含预期的内容。

waitForPageToLoad
暂停执行，直到预期的新页面成功加载。当使用 clickAndWait 命令时，此命令自动被调用。

waitForElementPresent
在一个 HTML 标签定义的 UI 元素，在页面上出现前，暂停执行脚本，直到该UI 元素出现为止。

验证页面元素

验证页面元素

验证 UI 元素是否出现在网页上，是最常见的自动化测试任务之一。Selenese 有多种手段来检查 UI 元素。重要的是，你要了解这些不同的方法，因为这些方法定义了你的测试行为。

例如，你会测试……

1. 元素是否出现在页面上？
2. 特定的文本是否出现在页面上？
3. 特定的文本是否出现页面的指定位置？

例如，如果您正在测试一个文本标题，文本和它的位置在页面的顶部，对于你的测试可能很重要。然而，如果您正在测试一个主页，这个主页面上有一个图片，网页设计师经常改变页面上的图片文件以及图片出现的位置，这时你只需要测试图片（而不是特定的图像文件）在页面上是否出现就可以了。

断言和验证

- verifyTextPresent
- verifyElementPresent
- verifyText

在 “assert”（断言）和 “verify”（验证）之间选择的关键点，取决于对测试失败如何管理以及管理的便利性。如果在检查浏览器显示的是否是预期页面时，测试已经失败，那么在此基础上再检查页面的第一个段落是否正确几乎没有意义了。如果打开的页面不对，你可能想立刻中止测试用例，检查原因并修复问题。另一种情况，你可能希望检查页面上的许多属性，在测试用例碰到第一个失败后并不终止执行，这将允许你检查所有页面上的失败，然后采取适当的行动。“assert”测试失败会中止当前的测试用例，而“verify”测试失败，会继续运行测试用例。

使用这个功能的最佳实践是将测试命令进行逻辑上的分组，在每组命令开始前有一个“assert”，后面跟着一个或多个“verify”测试命令。举个例子：

命令	目标	值
open	/download/	
assertTitle	Downloads	
verifyText	//h2	Downloads
assertTable	1.2.1	Selenium IDE
verifyTable	1.2.2	June 3,2008
verifyTable	1.2.3	1.0 beta 2

上面的例子首先打开一个页面，然后比较当前页面的标题与期望值是否一致，来 asserts（断言）当前加载的页面是正确的。只有这个断言测试通过后，才执行以下的命令并“verify”（验证）文本出现在预期的位置。接下来的测试用例继续“asserts”（断言）第一个表的第二行第一列包含预期的值，只有当这个断言测试通过，才继续验证这一行剩余的单元格。

verifyTextPresent

命令 verifyTextPresent 用于验证页面上是否存在指定的文本。此命令需要一个文本模式参数，用来做验证。例如：

命令	目标	值
verifyTextPresent	Marketing Analysis	

这将导致 Selenium 去搜索，并验证文本字符串 “Marketing Analysis” 出现在被测页面的某个地方。当你只考虑页面上是否出现特定的文本时，请使用 verifyTextPresent 命令。当您还需

要测试页面上文本出现的位置时，不要用这个命令。

verifyElementPresent

用这个命令来测试特定的 UI 元素，而不是其内容，出现在页面上。这个验证不检查文本而是检查 HTML 标记。一个常见的用法是检查一个图片的存在。

命令	目标	值
verifyElementPresent	//div/p/img	

这个命令验证一个图片，由 img HTML 标签指定的，是否在页面上存在，而且它在一个 div 标签和一个 p 标签下面。第一个（也是惟一一个）参数是一个定位器告诉 Selenese 命令如何找到这个元素。定位器会下一节中解释。

verifyElementPresent 可以被用来检查页面中存在的任何HTML标签。您可以检查链接、段落、块等是否存在。下面是几个例子。

命令	目标	值
verifyElementPresent	//div/p	
verifyElementPresent	//div/a	
verifyElementPresent	id=Login	
verifyElementPresent	link=Go to Marketing Research	
verifyElementPresent	//a[2]	
verifyElementPresent	//head/title	

这些例子说明了UI元素可能被测试的各种方式。定位器是在下一节中解释。

verifyText

当文本和它的UI元素都必须被测试时使用 verifyText 命令。verifyText 必须使用定位器。如果你选择 XPath 或 DOM 定位器，您可以验证在页面上特定的文本相对于其他 UI 组件出现在页面上的特定的位置。

命令	目标	值
verifyText	//table/tr/td/div/p	This is my text and it occurs right after the div inside the table.

验证页面元素 | 目录 | 定位元素

定位元素

对于许多 Selenium 命令，目标是必需的。这个目标用来标示 web 应用程序内容中的指定元素，他是用 locatorType = location 格式所描述。在许多情况下定位器类型可以省略。下面逐一解释各种定位器类型的例子。

标示符定位

这是最常用的定位元素的方法，当没有指定定位器类型时，默认使用这种定位器。这个策略会检索 id 属性值与 location 匹配的第一个元素。如果没有元素的 id 属性与之匹配，那么会检索 name 属性与 location 匹配的第一个元素。

例如，你的页面源代码可能有id和name属性如下：

```
1. <html>
2.  <body>
3.   <form id="loginForm">
4.    <input name="username" type="text" />
5.    <input name="password" type="password" />
6.    <input name="continue" type="submit" value="Login" />
7.   </form>
8.  </body>
9. <html>
```

下面的定位策略将返回上面 HTML片段 中相匹配元素的行号：

- identifier=loginForm (3)
- identifier=password (5)
- identifier=continue (6)
- continue (6)

因为 identifier 类型的定位是默认的，前三个例子的 identifier = 不是必需的。

ID 定位

这种类型的定位器比 identifier 定位器类型限制更多，但也更加明确。当你知道一个元素的 id 属性时使用这个定位器。

```
<html>
 <body>
  <form id="loginForm">
   <input name="username" type="text" />
   <input name="password" type="password" />
   <input name="continue" type="submit" value="Login" />
   <input name="continue" type="button" value="Clear" />
  </form>
 </body>
<html>
```

- id=loginForm (3)

Name 定位

Name 定位器类型将定位与 Name 属性相匹配的第一个元素。如果有多个名称属性相同的元素，那么您可以使用过滤器来进一步完善你的位置策略。默认的过滤器类型是值（匹配 value 属性）。

```
<html>
 <body>
  <form id="loginForm">
   <input name="username" type="text" />
   <input name="password" type="password" />
   <input name="continue" type="submit" value="Login" />
   <input name="continue" type="button" value="Clear" />
  </form>
</body>
<html>
```

- name=username (4)
- name=continue value=Clear (7)
- name=continue Clear (7)
- name=continue type=button (7)

注意

不像某些类型的定位器，如：XPath 和 DOM。上面三种类型的定位器允许 Selenium 测试 UI 元素不依赖与在页面上的位置。如果页面结构或者组织改变了，测试仍将通过。你可能会或可能不会对

页面结构发生变化的页面进行测试。如果网页设计师经常改变页面，但其功能必须回归测试，通过 id 和 name 属性，或通过任何 HTML 属性进行测试就变得非常重要了。

XPath 定位

XPath 是用于定位 XML 文档中节点的语言。由于 HTML 可以实现为 XML（XHTML），Selenium 的用户可以利用这个强大的语言来定位他们 web 应用程序中的元素。XPath 超出(以及支持)简单的 id 或 name 属性定位，并开辟了各种新的可能性，如定位页面上第三方的复选框。

使用 XPath 的主要原因之一是对于一个你希望找到的元素，你没有合适的 id 或 name 属性来定位。您可以用 XPath 绝对定位元素（不建议），或相对于一个有 id 或 name 属性的元素来定位。XPath 定位器也可以通过使用 id 和 name 属性之外的其他属性来定位元素。

绝对的 XPath 包含从根(html)所有元素的位置，因此只要轻微的调整应用程序页面，XPath 就会失效。通过寻找附近的 id 或 name 属性的元素（理想情况下是一个父元素）可以基于这种位置关系来定位你的目标元素。这种位置关系通常不太可能改变，可以使你的测试更加健壮。

因为只有 XPath 定位器从 // 开始，所以在指定 XPath 定位器时没有必要包含 xpath = label。

```
<html>
  <body>
   <form id="loginForm">
    <input name="username" type="text" />
    <input name="password" type="password" />
    <input name="continue" type="submit" value="Login" />
    <input name="continue" type="button" value="Clear" />
   </form>
 </body>
<html>
```

- xpath=/html/body/form[1] (3) - 绝对路径（对 HTML 轻微的修改会令此路径失效）
- //form[1] (3) - HTML中的第一个form 元素
- xpath = //form[@id = 'loginForm'] (3) - 拥有 id 属性值为 loginForm 的 element 元素
- xpath=//form[input/@name='username'] (3) - 拥有一个 input 子元素，该元素的 name 属性值为 username 的第一个表单元素
- //input[@name='username'] (4) - name 属性值为 username 的第一个 input 元素
- //form[@id='loginForm']/input[1] (4) - id 为 loginForm 的表单元素的第一个 input 子元素
- //input[@name='continue'][@type='button'] (7) - name 属性值为 continue 以及 type 属性值为 button 的 input 元素
- //form[@id='loginForm']/input[4] (7) - id 属性值为 loginForm 的表单的第四个

input 子元素

上面的例子都很基础，如果想更深入的学习，请参考下面的学习资料：

- W3Schools XPath Tutorial
- W3C XPath Recommendation

下面是一些非常有用的火狐浏览器的插件，能辅助定位 XPath 中的元素：

- XPath Checker - 生成 XPath 并检查 XPath 的有效性
- Firebug - 非常强大而有用的插件， XPath 只是其中很小的功能

Link Text 定位超链接

用链接文字来定位网页面上的超链接元素是一个很简单的方法。如果两个链接有相同的文本，那么第一个匹配的元素被定位。

```
1. <html>
2.  <body>
3.   <p>Are you sure you want to do this?</p>
4.   <a href="continue.html">Continue</a>
5.   <a href="cancel.html">Cancel</a>
6. </body>
7. <html>
```

- link=Continue (4)
- link=Cancel (5)

DOM 定位

文档对象模型用来表示一个 HTML 文档，可以使用 JavaScript 操作和访问。这个位置策略使用 JavaScript 来评估一个元素是否页面上，他通过使用分层的点号标记法，使得元素定位得到简化。

因为只有 dom 定位器由 document 开始，所以当指定一个 DOM 定位器时没有必要包括 dom = label 。

```
1.  <html>
2.   <body>
3.    <form id="loginForm">
4.     <input name="username" type="text" />
5.     <input name="password" type="password" />
6.     <input name="continue" type="submit" value="Login" />
7.     <input name="continue" type="button" value="Clear" />
```

```
    </form>
  </body>
  <html>
```

- dom=document.getElementById(‘loginForm’) (3)
- dom=document.forms[‘loginForm’] (3)
- dom=document.forms[0] (3)
- document.forms[0].username (4)
- document.forms[0].elements[‘username’] (4)
- document.forms[0].elements[0] (4)
- document.forms[0].elements[3] (7)

您可以使用 Selenium 以及其他网站或者浏览器扩展来探索你的 web 应用程序的 DOM。W3Schools 是一个好的参考。

CSS 定位

CSS（Cascading Style Sheets，层叠样式表）是用于描述 HTML 和 XML 文档样式的语言。CSS 使用选择器来绑定文档中元素的样式属性。这些选择器可以使用 Selenium 作为另一个定位策略。

```
<html>
 <body>
  <form id="loginForm">
   <input class="required" name="username" type="text" />
   <input class="required passfield" name="password" type="password" />
   <input name="continue" type="submit" value="Login" />
   <input name="continue" type="button" value="Clear" />
  </form>
</body>
<html>
```

- css=form#loginForm (3)
- css=input[name=”username”] (4)
- css=input.required[type=”text”] (4)
- css=input.passfield (5)
- css=#loginForm input[type=”button”] (7)
- css=#loginForm input:nth-child(2) (5)

关于 CSS 选择器的更多信息，请参考 W3C publication。你会在那里找到额外的资料。

注意

经验丰富的 Selenium 用户推荐 CSS 作为他们的定位策略，因为 CSS 选择器速度大大快于 XPath 并且可以在一个 HTML 文档中找到最复杂的对象。

隐式定位器

以下三种情况你可以省略定位器类型：

- 定位器没有明确指定定位策略的默认使用identifier 定位策略。
- 定位器开始与 // 将使用XPath定位策略。
- 定位器开始于 document 将使用DOM定位策略。

匹配文本模式

类似定位器参数，文本模式是另一种常用的 Selenium 命令参数。需要使用文本模式的命令，例如: verifyTextPresent, verifyTitle, verifyAlert, assertConfirmation, verifyText, verifyPrompt。上面已经提到，LinkText 定位器可使用文本模式。文本模式使用特殊字符来模糊匹配预期的文本，而不必准确的描述该文本。

有三种类型的模式: Globbing 模式，正则表达式和 exact。

Globbing 模式

很人都熟悉 globbing ，因为在 DOS 或 Unix / Linux 命令行下经常用 globbing 模式来匹配文件名，如: ls *.c。这个例子中，globbing 模式用于匹配当前目录下所有的 .c 扩展名的文件。Globbing 的功能很有限。Selenium 只支持和实现两个特殊字符:

- * 匹配任何事情，例如: 什么也没有，一个字符，或许多字符。
- [ ]（字符类）匹配方括号中的任何单个字符。一个连字符指定一定范围的字符（通常是连续的 ASCII 字符集）。下面几个例子将演示字符类的用法:
    - [aeiou] 匹配任何一个小写的元音字母
    - [0 - 9] 匹配任何数字
    - [a-zA-Z0-9] 匹配任何字母和数字符号

在很多其他系统中，globbing 还包括第三个特殊字符问号 “?”。然而，Selenium globbing 模式只支持星号和字符类。

在一个 Selenese 命令中，你可以用 glob: 标记来声明，指定使用 globbing 文本模式参数。然而，由于 globbing 文本模式是默认的，所以你也可以省略 glob: 标签而只保留文本模式本身。

下面例子中，有两个命令使用 globbing 文本模式。在页面上实际的链接文本是 “Film/Television Department”，使用了文本模式而不是准确的文本，click 命令将继续工作，即使链接文本改为“Film & Television Department” 或 “Film and Television Department”。glob 文本模式的星号会匹配 “Film” 单词和 “Television”单词中间的任何符号或者没有符号。

命令	目标	值
click	link=glob:Film*Television Department	

verifyTitle	glob:*Film*Television*	

点击链接所打开页面的实际标题是 “De Anza Film And Television Department - Menu”。通过使用文本模式而不是准确的文本，只要页面标题中的任何地方前后出现 “Film” 和 “Television” 这两个单词，则 verifyTitle 将验证通过。例如，如果网页作者缩短标题为 “Film & Television Department”，测试会通过。在链接文本或者普通文本中使用文本模式将会大大减小测试案例的维护成本。

正则表达式文本模式

正则表达式文本模式在 Selenese 支持的三种文本模式中是功能最强大的。很多高级编程语言、很多文本编辑器以及很多工具中都支持正则表达式。包括 Linux / Unix 命令行实用工具 grep、sed 和 awk。在 Selenese 中，正则表达式文本模式允许用户执行许多非常困难的任务。例如，假设测试需要确保一个特定表单元中只包含一个数字。regexp: [0 - 9]+ 这个简单的文本模式，它将匹配任何长度的十进制数。

而 Selenese globbing 文本模式只支持通配符*和[ ]（字符类）功能，Selenese 正则表达式文本模式提供存在于 JavaScript 中的相同广泛的特殊字符。下面是一个特殊字符的子集：

模式	匹配
.	任何单个字符
[ ]	字符类：出现在方括号中的任意单个字符
*	数量：0 个或多个前面的字符（或者组）
+	数量：1 个或多个前面的字符（或者组）
?	数量：0 个或 1 个前面的字符（或者组）
{1,5}	数量：1 至 5 个 前面的字符（或者组）
|	交替：竖道左边的字符/组或者右边的字符/组
( )	组：经常用在 and/or 替换的量

正则表达式文本模式在 Selenese 命令参数中需要用 regexp: 或 regexpi: 前缀。前者是大小写敏感的，后者是不区分大小写的。

下面几个例子将帮助阐明如何使用正则表达式模式与 Selenese 命令。第一个例子是正则表达式最常用的方式 .* （点星）。这两个字符序列可以表示 0 或出现任何字符或者更简单，任何字符或没有字符。它相当于 globbing 模式中的一个 * （星号）。

命令	目标	值
click	link=regexp:Film.*Television Department	
verifyTitle	regexp:.*Film.*Television.*	

上面的例子与前面使用 globbing 字符模式例子的功能相同。唯一的区别是前缀（regexp: 代替了 glob:）任何字符或没有字符模式（. 代替了 ）。

下面例子稍微复杂一些，测试 Anchorage Alaska 地区的 Yahoo! 天气页面中的日出时间：

命令	目标	值
open	http://weather.yahoo.com/forecast/USAK0012.html	
verifyTextPresent	regexp:Sunrise: *[0-9]{1,2}:[0-9]{2} [ap]m	

下面将逐一解释上面的正则表达式：

Sunrise: *	Sunrise: 字符串跟着 0 个或多个空格
[0-9]{1,2}	1 或 2 位数字（表示小时）
:	字符 ： 小时和分钟中间的分割字符
[0-9]{1,2}	2 位数字（表示分钟）后面跟一个空格
[ap]m	a 或者 p 后面跟着 m 字符，表示 ap 或者 pm

Exact 文本模式

Selenium 的 exact 文本模式只有非常少的使用场合。这种文本模式没有什么特殊字符，所以顾名思义称为准确模式。所以，当你需要匹配一个包含星号的文本时，globbing 和正则表达式都是用星号作为特殊字符，这时 exact 文本模式就派上用场了。例如，你想在下拉列表中选择 “Real ” 选项，下面的代码可能有效或者无效。*glob:Real* 文本模式将匹配 Real 开头的任何字符串。这样，如果这个选项前面有一个选项文本为 “Real Numbers”，这个选项将匹配成功，而不是 “Real *”。

select	//select	glob:Real *

为了确保 “Real *” 选项被选择，exact: 前缀的 exact 文本模式如下所示：

select	//select	exact:Real *

实际上正则表达式文本模式通过转义星号可以实现同样的效果：

select	//select	regexp:Real *

测试人员平时很少需要匹配带星号或方括号（globbing 模式的字符类）的文本。因此，globbing 文本模式和正则表达式文本模式能解决绝大多数的字符匹配的问题。

AndWait 命令

命令（如常见的 click ）与 AndWait 后缀命令的区别是，普通命令会执行动作，并以最快的速度继续执行下面的命令，而 AndWait 后缀命令（如 clickAndWait ）告诉 Selenium 在动作完成后等待页面加载完成。

AndWait 后缀命令经常用在浏览器导航到另一个页面或重新加载当前页面时。

请注意，如果您在某个动作上使用一个 AndWait 命令，他不触发导航或者刷新，您的测试将会失败。这是因为 Selenium 达到了 AndWait 超时上限，却没有看到任何导航或刷新，导致 Selenium 抛出超时异常。

匹配文本模式 | 目录 | WaitFor 命令

AJAX 应用中的 WaitFor 命令

在 AJAX 驱动的 web 应用程序中，应用程序从服务器检索数据，而无需刷新页面。andWait 命令在页面没有真正刷新前不会工作。让当前运行的测试暂停一段时间也不是一个好方法，因为被测的 web 元素可能比预定的暂停时间出现的略晚或稍早，这主要取决于系统的响应能力，负载大小或其他不可控因素，这些都会导致测试失败。最好的方法就是在一个动态的时长内等待所需的元素，一旦找到界面元素，继续执行测试。

waitFor 命令实现了上述功能，waitForElementPresent 或 waitForVisible，这些命令会动态的等待，每隔一秒会检查所需的条件，一旦条件满足则继续执行脚本中的下一个命令。

AndWait 命令 | 目录 | 顺序执行和流程控制

顺序执行和流程控制

脚本是一个命令一个命令顺序执行的。

Selenese 本身不支持条件语句（if - else, 等）或循环迭代（for, while, 等）。没有流程控制也可以制作很多有用的测试案例。然而，对于动态内容的功能测试，可能涉及到多个页面，编程逻辑通常是必要的。

如果必须使用流程控制，有以下三种解决方案：

1. 使用 Selenium RC 运行脚本结合客户端库，如：Java 或 PHP 库，利用编程语言的流程控制功能。
2. 用 storeEval 命令从脚本中运行一个小的 JavaScript 代码片段。
3. 安装 goto_sel_ide.js 扩展。

很多测试人员将测试脚本导出成一个编程语言的代码文件，结合 Selenium RC API 一起使用（参见Selenium IDE 章）。然而，有些组织更倾向于尽可能地用 Selenium IDE 运行测试脚本（例如，当他们有初级测试工程师，或当编程技能缺乏）。如果是这种情况，请考虑 JavaScript 代码片段或 goto_sel_ide.js 扩展。

AJAX 应用中的 WaitFor 命令 | 目录 | 访问器命令和变量参数

访问器命令和变量参数

在脚本的开头，可以使用 Selenium 变量来存储常数。另外，在数据驱动测试中（将在后面一节中讨论），Selenium 变量可用于存储从命令行，从另一个程序，或从一个文件中传递的数据。

store 命令是所有存储命令中最基础的命令，它仅仅能把一个常量存储在一个 Selenium 变量中。它包括两个参数，存储的文本值和 Selenium 变量。使用标准的变量命名约定来为变量起名，即变量名中只能包含字母和数字。

命令	目标	值
store	paul@mysite.org	userName

在后面的脚本中，你可以使用变量中的数据。要访问变量的值，用花括号({ })括住变量名，并前置一个美元符号。

命令	目标	值
verifyText	//div/p	${userName}

变量的常见用法是存储 input 字段中的输入数据。

命令	目标	值
type	id = login	${userName}

Selenium 变量可以用在第一或第二个参数，并且相比较其他操作符会被命令优先解释执行。Selenium 变量也可以用在一个定位器表达式中。

每个验证和断言命令都有等价的 store 命令。下面是一些常用的 store 命令。

storeElementPresent

对应于 verifyElementPresent 命令。它存储一个布尔值 true 或 false ，取决于 UI 元素是否出现。

storeText

StoreText 对应于 verifyText 命令。它使用一个定位器来识别页面文本。如果找到文本，则存储在变量中。StoreText 可以用来提取被测页面中的文本信息。

storeEval

该命令需要一个脚本作为第一个参数。在 Selenese 中使用 JavaScript 参数，将在下一节中介绍。StoreEval 允许将 JavaScript 脚本运行的结果存储在一个变量中。

顺序执行和流程控制 | 目录 | JavaScript 和 Selenese 变量参数

JavaScript 和 Selenese 变量参数

JavaScript 可以用于两种类型的 Selenese 参数：脚本参数和非脚本参数（通常是表达式参数）。很多情况下，你可能需要访问或操作 Selenese 参数中的JavaScript 代码片段中的变量。测试用例中的所有变量被存储在 JavaScript 关联数组中。关联数组用字符串来索引而不是用连续的数字索引。存放测试案例变量的关联数组的变量名是 storedVars。当你想在 JavaScript 代码片段中访问或操作变量时，你需要通过 storedVars['yourVariableName'] 来访问。

使用 JavaScript 脚本参数

有些 Selenese 命令会指定 script 脚本参数，包括 assertEval, verifyEval, storeEval, waitForEval。这些参数没有需要特殊的语法要求。Selenium IDE 用户只需放置一个 JavaScript 代码片段的到适当的字段，通常是 Target 目标字段（因为脚本参数通常是第一或唯一的参数）。

下面的例子说明了如何使用 JavaScript 代码片段来执行一个简单的数值计算：

命令	目标	值
store	Edith Wharton	name
storeEval	storedVars['name'].toUpperCase()	uc
storeEval	storedVars['name'].toLowerCase()	lc

下一个示例演示了如何在 JavaScript 代码片段中调用方法，示例代码中，JavaScript 字符串对象调用了 toUpperCase 方法和 toLowerCase 方法。

使用非脚本参数

JavaScript 脚本还可以用于生成参数的值，即使没有指定参数是 script 脚本类型。在这种情况下，通过使用特殊的语法，即整个参数值由 javascript 做前缀，在花括号中放置代码片段，例如：javascript { 这里是你的代码 }。下面的例子中在 type 命令的第二个 value 值参数中通过 JavaScript 代码使用这个特殊语法产生参数值：

命令	目标	值
store	league of nations	searchString
type	q	javascript{storedVars['searchString'].toUpperCase()}

echo - Selenese 打印命令

Selenese 有一个简单的命令可以在测试时打印输出文本。当测试在运行时，在控制台中打印显示测试中的信息，这个功能往往是非常有用的。这些信息可以用来提供测试上下文的结果报告，当你在测试中发现问题时，这个结果报告经常用来报告测试页面时发现的缺陷。最后，还可以使用 echo 语句打印 Selenium 变量的内容。

警告、弹窗和多个窗口

假设有一个测试页面，其代码如下所示。

```
<!DOCTYPE HTML>
<html>
<head>
  <script type="text/javascript">
    function output(resultText){
      document.getElementById('output').childNodes[0].nodeValue=resultText;
    }

    function show_confirm(){
      var confirmation=confirm("Chose an option.");
      if (confirmation==true){
        output("Confirmed.");
      }
      else{
        output("Rejected!");
      }
    }

    function show_alert(){
      alert("I'm blocking!");
      output("Alert is gone.");
    }
    function show_prompt(){
      var response = prompt("What's the best web QA tool?","Selenium");
      output(response);
    }
    function open_window(windowName){
      window.open("newWindow.html",windowName);
    }
    </script>
</head>
<body>

  <input type="button" id="btnConfirm" onclick="show_confirm()" value="Show confirm box" />
  <input type="button" id="btnAlert" onclick="show_alert()" value="Show alert" />
  <input type="button" id="btnPrompt" onclick="show_prompt()" value="Show prompt" />
  <a href="newWindow.html" id="lnkNewWindow" target="_blank">New Window Link</a>
  <input type="button" id="btnNewNamelessWindow" onclick="open_window()" value="Open Nameless
```

```
    Window" />
39.   <input type="button" id="btnNewNamedWindow" onclick="open_window('Mike')" value="Open Named
    Window" />
40.
41.   <br />
42.   <span id="output">
43.   </span>
44. </body>
45. </html>
```

用户必须响应警告/确认弹窗，并且将焦点移到新开的弹窗上。幸运的是，Selenium 可以处理 JavaScript 弹窗。

在我们开始介绍警告/确认/提示的细节之前，有必要先了解一些共性的基础知识。警告、确认和提示都有下面的一些命令模式：

命令	描述
assertFoo(*pattern*)	如果模式不匹配弹窗中的文本，抛出错误
assertFooPresent	如果没有弹窗，抛出错误
assertFooNotPresent	如果存在任何弹窗，抛出错误
storeFoo(*variable*)	把弹窗中的文本存储在一个变量中
storeFooPresent(*variable*)	把弹窗中的文本存储在一个变量中并返回真或假

当 Selenium 运行时，JavaScript 弹窗将不会出现。这是因为 JavaScript 的函数调用，在运行时被 Selenium 自己的 JavaScript 代码覆盖。然而，没看到弹窗，并不意味着不用处理它。通过调用 AssertFoo(pattern) 来处理弹窗。如果弹窗显示的断言失败了，测试会暂停并提示类似下面的错误信息：
[error] Error: There was an unexpected Confirmation! [Chose an option.]

Alerts 警告弹窗

首先介绍警告弹窗，因为这是要处理的最简单的弹窗。首先，在浏览器中打开上面的 HTML 示例文件，单击“显示警告”弹窗按钮。请注意，当你关闭警告弹窗后，页面上会显示“Alert is gone.”。现在，在 Selenium IDE 的录制模式下，重复刚才的操作，注意文本验证命令在关闭弹窗后会自动加到了脚本中。录制好的测试案例将会是下面的样子：

命令	目标	值
open	/	
click	btnAlert	
assertAlert	I’m blocking!	
verifyTextPresent	Alert is gone.	

你可能会奇怪，自己并没有添加警告弹窗的断言，这些都是 Selenium IDE 为我们做的处理，Selenium IDE 同时也为我们关闭了警告弹窗。如果你去掉断言命令，然后回放脚本，你会收到以下错误：
[error] Error: There was an unexpected Alert! [I'm blocking!]

你必须在脚本中包含一个断言命令来承认弹窗的存在。

如果你只是想断言警告弹窗出现了，并不关心它包含的文本是什么，这时可以使用 assertAlertPresent。他将返回真或假，当返回假时，停止测试。

Confirmations 确认弹窗

确认弹窗的行为跟警告弹窗类似，assertConfirmation 和 assertConfirmationPresent 提供与警告弹窗命令类似的功能。默认情况下，弹出确认弹窗后，Selenium 会选择“OK”。试着录制下列操作，在示例页面上点击“显示确认弹窗”的按钮，然后在确认弹窗框中选择 “Cancel” 按钮，然后断言输出的文本。录制好的测试案例看上去可能像下面这样：

命令	目标	值
open	/	
click	btnConfirm	
chooseCancelOnNextConfirmation		
assertConfirmation	Choose an option.	
verifyTextPresent	Rejected	

chooseCancelOnNextConfirmation 函数告诉 Selenium 所有后续的确认应返回 false。它可以通过调用 chooseOkOnNextConfirmation 来重置。

请注意，此测试案例不能正常回放，届时 Selenium 会提示有一个未处理的确认。这是因为 Selenium IDE 录制事件的顺序错误导致的 Click 和 chooseCancelOnNextConfirmation 两个命令的顺序错误。（仔细想想，Selenium 并不知道在你打开一个确认弹窗之前会取消操作）只要交换前后这两个命令的位置，测试案例就会正常工作。

Prompts 提示弹窗

提示弹窗的行为跟警告弹窗类似，assertPrompt 和 assertPromptPresent 提供与警告弹窗命令类似的功能。默认情况下，弹出的提示信息后，Selenium 将等待你输入数据。试着录制下列操作，在示例页面上点击“显示提示弹窗”的按钮，然后在提示框中输入 Selenium 。录制好的测试案例看上去可能像下面这样：

命令	目标	值

open	/	
answerOnNextPrompt	Selenium!	
click	id=btnPrompt	
assertPrompt	What’s the best web QA tool?	
verifyTextPresent	Selenium!	

如果你在提示弹窗上点选取消按钮，你可能会注意到，answerOnNextPrompt 只会显示一个空白的目标。在提示弹窗上 Selenium 对待取消和一个空白的输入是一样的。

调试脚本

调试是发现错误和修复错误，是测试案例开发过程中的常见动作。

这里不打算介绍如何调试，很多 Selenium 的新手也已经具备基本的调试经验。如果调试对于你是全新的领域，建议你向公司里的开发人员请教。

断点和起点

Selenium IDE 通过设置断点可以让运行中的测试脚本在断点处暂停执行或者继续执行。通过断点，你可以让自动化脚本运行在特定的命令处暂停，可以观察该脚本在这个点上的行为。要做到这一点，需要在待观察的命令前一条命令处设置断点。

设置一个断点，需要在表格视图选择一条命令，单击鼠标右键，从上下文菜单中选择“Toggle Breakpoint”。然后点击运行按钮，运行脚本，脚本将会从第一个测试案例脚本的第一行一直运行到断点处。

有些时候需要从测试案例脚本的中间某行开始运行，一直运行到最后一行。或者从测试案例脚本的中间某行开始运行，一直运行到断点处。这个时候设置起始点就很有用处了。举一个常见的例子，假设你的测试案例需要先登录网站，然后执行一系列测试。现在你需要对登录后的测试案例进行调试，通过在登录测试脚本之后设置起始点，这样你调试脚本的时候，就不用每次都从登录开始执行测试案例，因为，你在登录之前势必要先退出登录，这就省去每次退出登录了。

设置一个起始点，先选中一个命令，在该命令上点击鼠标右键，在上下文菜单中选择“Set/Clear Start Point”。然后点击运行按钮，测试案例从起始点开始运行了。

单步执行

通过下面的步骤和方法，可以一次执行测试案例的一个命令，也就是所谓的单步执行：

1. 通过工具栏的启动按钮，启动测试脚本的执行；

2. 通过工具栏的暂停按钮，快速暂停测试脚本的执行；

3. 重复的点击单步执行按钮，点击一次，执行一个测试脚本中的命令。

Find按钮

“Find” 按钮用来查看当前选中的 Selenium 命令所操作的 web 页面上的 UI 元素。对于一个命令，如果第一个参数是定位器，这个工具用来验证定位器是否正确很有用。这个按钮可以用在所有需要操作 UI 界面元素的命令上，例如: click, clickAndWait, 以及相应的断言和验证命令。

从表格视图上，选择带有定位器参数的一个命令，点击 “Find” 按钮，现在观察页面。在定位器所标识的页面元素周围有一个绿色的方框包围。

页面源代码

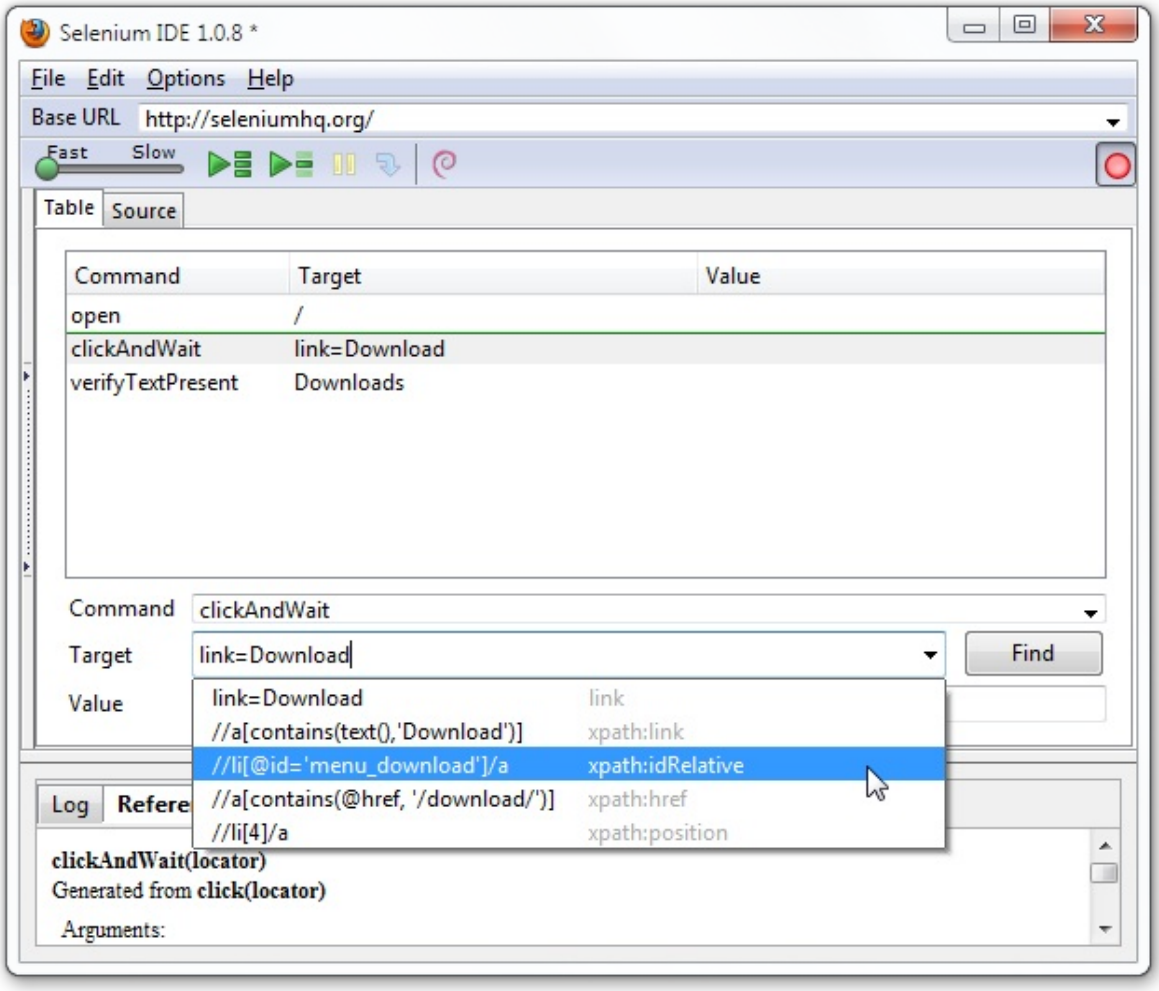
在调试测试脚本时，经常需要查看被测网页的源代码来定位问题。火狐浏览器查看网页源代码非常容易。只要在被测网页上点击鼠标右键选择“查看网页源代码”，网页源代码会显示在另外一个窗口中，使用查找功能可以找到 HTML 代码中的指定 UI 元素的信息。

另一种方式，选择网页上感兴趣的部分元素，点击鼠标右键选择“查看选中部分源代码”，火狐浏览器打开一个新窗口，只有一部分网页 HTML 代码显示出来，其中选中部分 UI 界面元素的网页 HTML 代码被高亮显示出来了。

辅助定位

当 Selenium IDE 录制一个有定位类型参数的命令时，他会存储一些额外的定位信息，允许用户去替换选择可能的定位参数。此功能有利于我们学习掌握更多关于定位器的知识，并且经常可以用来辅助修改录制好的测试脚本。

这个定位器辅助工具在 Selenium IDE 窗口上命令字段下面的 Target（目标）字段的右边，是一个下拉列表。只有当 Target（目标）字段录制的时候是定位器时，这个辅助工具才会生效。



警告、弹窗和多个窗口 | 目录 | 编写测试套件

编写测试套件

测试套件是测试用例的集合，测试用例在 Selenium IDE 的左侧窗格中进行管理。通过点击一排带小点的分隔条，可以手动的打开和关闭左侧面板。

当用户通过文件菜单打开一个现有的测试套件或新建一个测试用例时，测试套件面板会自动打开。在后一种情况下，新建的测试用例将立即出现在以前的测试用例下面。

通过“文件”->“添加测试用例”菜单项，Selenium IDE 也支持加载已有的测试用例。可以将现有测试用例添加到一个新的测试套件中。

一个测试套件文件是一个 HTML 文件，其中包含一个单列的表格。 中每行中的单元格中包含了一个测试用例文件的链接。下面是一个包含了四个测试用例的测试套件的例子：

```
<html>
<head>
    <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
    <title>Sample Selenium Test Suite</title>
</head>
<body>
    <table cellpadding="1" cellspacing="1" border="1">
        <thead>
            <tr><td>Test Cases for De Anza A-Z Directory Links</td></tr>
        </thead>
    <tbody>
        <tr><td><a href="./a.html">A Links</a></td></tr>
        <tr><td><a href="./b.html">B Links</a></td></tr>
        <tr><td><a href="./c.html">C Links</a></td></tr>
        <tr><td><a href="./d.html">D Links</a></td></tr>
    </tbody>
    </table>
</body>
</html>
```

注意

测试套件与测试案例文件之间的路径关系，不要随便改变。如果路径关系改变了，一定要记着更新文件中的链接地址。

调试脚本 | 目录 | 用户自定义扩展

用户自定义扩展

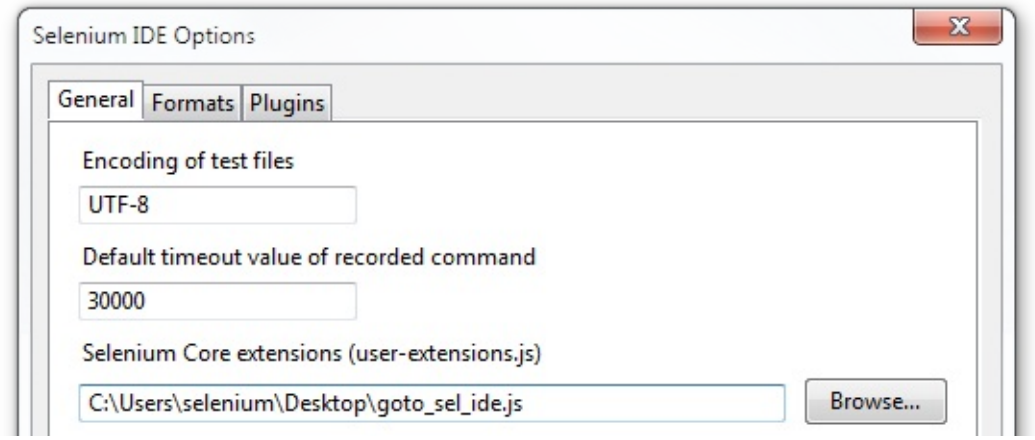
用户自定义扩展是用户自己创建的 JavaScript 文件，对 Selenium IDE 现有的功能进行定制以及功能扩展。通常这种定制和扩展是以自定义命令的形式来体现的，当然也不仅限于命令。

这里可以找到很多有用的扩展。

注意：这部分信息已经过时，我们将很快修改。

也许最受欢迎的 Selenium IDE 扩展就是流程控制，流程控制扩展将提供 while 循环和条件判断。这个扩展是 goto_sel_ide.js。如何使用这个扩展所提供的功能，看看作者创建的页面。

安装这个扩展的步骤为，在 Selenium-IDE 的 Options 菜单 => Options 子菜单项 => General 选项卡下，把 goto_sel_ide.js 在计算机上的路径信息填到 Selenium Core extensions 字段中，即可。



点选 OK 按钮后，需要关闭并重启 Selenium IDE，以保证扩展文件被载入。对扩展文件的任何修改都需要关闭并重新打开 Selenium-IDE。

关于如何编写用户自定义扩展，请查看 Selenium 参考文档。

经验证有时单步调试对于 Selenium IDE 和用户自定义扩展是很有用的。基于 XUL/Chrome 唯一能够进行调试的扩展是 Venkman，这个扩展直到 Firefox 32 版本才支持。经验证在 Firefox 32 和 Selenium IDE 2.9.0 中已经支持单步调试。

格式化

在 Options 菜单下，Format 子菜单，可以选择用来保存和显示测试案例所使用的编程语言。默认格式是 HTML。

如果使用 Selenium RC 来运行测试用例，这个功能用于将测试用例转化为一种编程语言。首先，选择一种编程语言，例如 Java、PHP，该编程语言将结合 Selenium RC 来开发测试程序。然后，使用“文件”=>“Export Test Case As”来保存测试案例。测试用例将根据你选择的语言转化为一系列的函数。Selenium IDE 可以生成基础的程序代码来支持后续的测试。

同时，注意，如果生成的代码不符合你的需要，你可以通过通过修改配置文件来改变它，配置文件中定义了程序代码的生成过程。每个受支持的语言都有可编辑的配置设置。在 Options => Options => Formats 选项卡下。

注意

新版本的 Selenium IDE 已经不支持这个菜单项了。但是导出代码的功能还是保留的。

用户自定义扩展 | 目录 | 跨浏览器运行 Selenium IDE 脚本

跨浏览器运行 Selenium IDE 脚本

虽然 Selenium IDE 只能运行在火狐浏览器上，但是 Selenium IDE 开发的测试案例却可以运行在其他浏览器上。使用一个简单的命令行接口来调用 Selenium RC 服务器。这个话题在Selenium RC 运行 Selenese 测试部分章节有介绍。-htmlSuite 命令行选项是跟这个特定功能相关的。

格式化 | 目录 | 排错

排错

下面成对的图片/解释列表描述了 Selenium IDE 中经常会出现的问题及原因:

Table view is not available with this format.
这种格式的表格视图不可用。

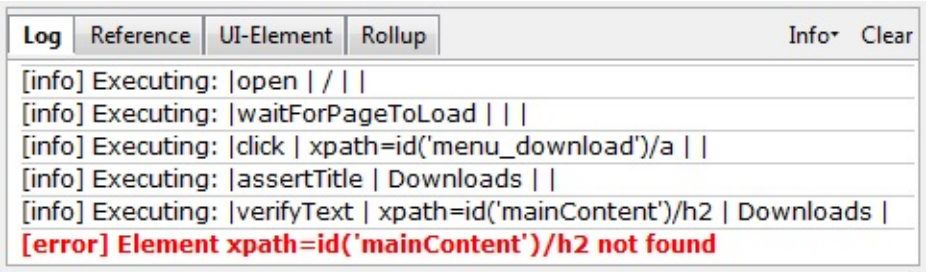
Selenium IDE 启动时这个消息偶尔会显示在表格视图中。Selenium IDE 的工作区将会关闭并重启。要想了解更多信息，请见 issue 1008。如果你能复现这个问题，请提供具体细节以便我们修复此问题。

---

error loading test case: no command found
测试用例加载错误:命令没有找到

你一定是用“文件 => 打开”试图打开一个测试套件文件。请使用“文件 => 打开测试套件”菜单进行操作。

一个改进请求已提交了，希望改善这个错误消息。请见 issue 1010。



这种类型的错误可能表明一个时间问题，例如，命令中定位器指定的元素在命令执行的时候没有完全加载。尝试在命令前面插入一个 **pause 5000** 的命令来确定是否确实是与时间相关的问题。如果是这样，在失败的命令之前使用一个适当的 **waitFor*** 或 ***AndWait** 命令。

类似上面的例子，你在 **open** 命令中，试图使用变量而导致失败，提示你在使用变量中的值之前并没有创建这个变量。这是由于把变量的值字段，应该放在目标字段。在上面的示例中，**store** 命令的两

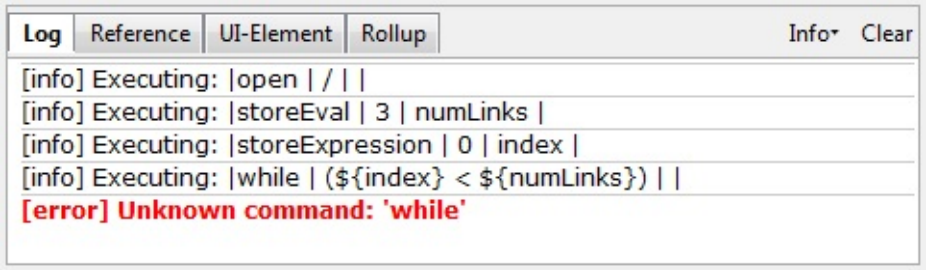
有创建这个变量。这是由于把变量的值字段，应该放在目标字段。在上面的示例中，**store** 命令的两个参数顺序放反了。对于任何 Selenese 命令，第一个参数必须在目标字段，第二个参数（如果有的话）必须在值字段。

---

error loading test case: [Exception… "Component returned failure code: 0x80520012 (NS_ERROR_FILE_NOT_FOUND) [nsIFileInputStream.init]" nresult: "0x80520012 (NS_ERROR_FILE_NOT_FOUND)" location: "JS frame :: chrome://selenium-ide/content/file-utils.js :: anonymous :: line 48" data: no]

测试套件中的某个测试用例无法找到。确保测试用例确实位于测试套件中所指定的位置。同时，确保实际的测试用例文件扩展名是 .html ，并且在测试套件文件中指定的也是 .html 扩展名。

一个改进请求已提交了，希望改善这个错误消息。请见 issue 1011。



Selenium IDE 没有载入相应的 js 扩展。确保你在 **Options => Options => General** 中的 **Selenium Core extensions** 字段填写了正确的 js 扩展文件的名称和地址。并且，任何时候修改了 **Selenium Core extensions** 字段的值之后，Selenium IDE 都必须重新启动。