

Selenium 中文文档

作者: qi_ling2006 <http://qi-ling2006.iteye.com>

Translate:Planisnothing
by:Jarvi

目 录

1. 正文

1.1 Selenium用户指南 - 第一章 Selenium 2.0 文档修订注解4

1.2 Selenium用户指南 - 第二章 入门 5

1.3 Selenium用户指南 - 第三章 Selenium IDE[1]11

1.4 Selenium用户指南 - 第三章 Selenium IDE[2]14

1.5 Selenium用户指南 - 第三章 Selenium IDE[3]18

1.6 Selenium用户指南 - 第三章 Selenium IDE[4]25

1.7 Selenium用户指南 - 第三章 Selenium IDE[5]31

1.8 Selenium用户指南 - 第三章 Selenium IDE[6]41

1.9 Selenium用户指南 - 第四章 Selenium 2.0和WebDriver[1]47

1.10 Selenium用户指南 - 第四章 Selenium 2.0和WebDriver[2]53

1.11 Selenium用户指南 - 第四章 Selenium 2.0和WebDriver[3]58

1.12 Selenium用户指南 - 第四章 Selenium 2.0和WebDriver[4]65

1.13 Selenium用户指南 - 第五章 WebDriver: 高级用法68

1.14 Selenium用户指南 - 第六章 Selenium 1 (Selenium RC)[1]73

1.15 Selenium用户指南 - 第六章 Selenium 1 (Selenium RC)[2]82

1.16 Selenium用户指南 - 第六章 Selenium 1 (Selenium RC)[3]88

1.17 Selenium用户指南 - 第六章 Selenium 1 (Selenium RC)[4]94

1.18 Selenium用户指南 - 第七章 测试设计的考虑[1]102

1.19 Selenium用户指南 - 第七章 测试设计的考虑[2]108

1.20 Selenium用户指南 - 第八章 Selenium-Grid118

1.21 Selenium用户指南 - 第九章 用户扩展119

1.1 Selenium用户指南 - 第一章 Selenium 2.0 文档修订注解

发表时间: 2012-05-18

Selenium 2.0 文档修订注解

你好，欢迎！文档团队欢迎你，并要感谢你对Selenium的兴趣。

我们目前正在为Selenium 2.0 发布更新本文档。这意味着我们正在编写和编辑新资料和修订旧的资料。在阅读时，你可能遇到打字错误或其他较小的错误。如果如此，请保持对我们的耐心。不是隐藏信息直到最终完成，我们频繁地检查和修订新的资料。尽管如此，我们首先检查我们的素材和我们对我们提交的信息的精确性和可用性有足够的自信。然而，如果你发现任何错误，特别是我们的代码示例错误，请务必让我们知道。你可以创建一个带有"Docs Error"主题的新问题 (<http://code.google.com/p/selenium/issues/entry>)。

我们已经在这文档上工作的非常努力。并且，正如刚刚说到的，我们将在新的版本在一次努力工作。为何？因为我们绝对相信Selenium是Web应用程序测试的最佳工具。我们觉得它的可扩展性和灵活性，以及与浏览器的高度集成，通过提供可得到的专门的工具，是无与伦比的。我们非常兴奋地推荐Selenium，并且希望去拓展它的用户社区。简而言之，我们真的想要充分地展示Selenium。

我们相信你会同样地兴奋，一旦你理解Selenium如何达成测试自动化。它与其他自动化工具是相当的不同。无论你是否你是一个新手，或者已经使用过一段时间，我们相信本文档对传播相关的知识真的有帮助。我们的目标是让测试自动化的新手可以使用本文档作为进身之阶。然而，同时我们也介绍了大量的有经验的软件工程师感兴趣的高级，测试设计的主题。我们编写的"Sel-Docs"，对各种能力的测试工程师，快速地成为高效编写自己的Selenium测试的工程师都是有幫助的。有经验的用户和初学者同样地可以从我们的Selenium用户指南中获益。

非常感谢你的阅读

-- Selenium文档团队

1.2 Selenium用户指南 - 第二章 入门

发表时间: 2012-05-18

入门

Web应用程序测试自动化

今天的软件应用, 许多或许是大多数, 是运行在一个浏览器中的基于Web的应用程序. 这些应用程序的测试效果在不同的公司和组织间变化很大. 在一个高度互动和响应的软件时代, 许多组织在开发过程中都会使用某种形式的敏捷方法, 测试自动化正在成为软件项目的必要条件. 测试自动化是问题的答案. 测试自动化意味着使用一个软件工具, 对被测试的应用程序运行可重复的测试. 为回归测试提供响应能力。

测试自动化有许多优点. 大多数是和测试的可重复性, 可执行测试的速度有关. 有大量的、可得到的, 有助于测试自动化开发的商用和开源工具. Selenium可能是最广泛使用的解决方案. 本文档将帮助新手和有经验的用户, 学习有效的技术, 为Web应用程序构造测试自动化.

用户指南介绍Selenium, 讲授它的功能, 和提供常用的、由Selenium社区积累的最佳实践。提供了许多示例。同样, 提供有关Selenium内部结构的技术信息, 和Selenium的推荐使用方法。

测试自动化对改善一个软件团队的测试过程的长期的效能有特别的益处。测试自动化支持：

- - 频繁地回归测试
 - 提供开发者快速地回馈
 - 几乎没有限制测试案例的迭代执行
 - 支持敏捷和极限开发方法
 - 测试案例的文档化
 - 自定义缺陷报告
 - 查找手工测试遗漏的缺陷

自动化还是不自动化?

自动化总是有益的么? 应何时决定自动化测试案例?

自动化测试并不总是有益的。有时手工测试可能是更适当地。例如, 如果应用程序的用户接口将在不久的将来作出显著地改变, 则任何自动化测试可能需要重新编写。同样, 有时仅仅是没有足够的时间来构造自动化测试。从短期来看, 手

工测试可能是更有效的。如果一个应用程序有非常紧的时间期限，没有现成的、可得到的测试自动化，而且测试必须在给定的时间范围内完成，显然手工测试是最佳的解决方案。

Selenium介绍

Selenium是一套不同的软件工具，每个拥有一个不同的方法来支持测试自动化。大多数的Selenium QA工程师聚焦于一个或两个，很好地满足他们的项目需求的工具，然而，学习所有的工具将给予你许多地、解决不同的测试自动化问题的选项。整个工具集，带有一套丰富的测试特性，特别适合于所有类型的Web应用程序测试的需要。这些操作是高度灵活的，为定位UI元素，比较预期的测试结果和实际的应用程序行为，提供许多选项。Selenium的一个关键特征是，对一个测试可以在多个浏览器平台执行的支持。

Selenium项目简史

Selenium首次诞生在2004年，当Jason Huggins在ThoughtWorks公司测试一个内部的应用程序的时候。作为一个聪明的家伙，他认识到可以更好的使用他的时间比对做出的变更进行手工地测试。他开发了一个可以驱动与页面交互的Javascript库，允许他自动地在多个浏览器上自动地重复运行测试。那个Javascript库最终成为Selenium的内核，成为所有的Selenium RC（远程控制）和Selenium IDE的基础。Selenium RC是开拓性的，因为没有其他的产品允许你从一个你选择的语言去控制浏览器。

尽管Selenium是一个令人吃惊的工具，他也不是没有缺陷。因为他的基于自动化引擎的Javascript，和浏览器应用到Javascript上的安全限制，个别事情成为不可能完成的任务。让事情变得“更糟糕的”是，Web应用程序随着时间的过去变得越来越强大，使用新浏览器提供的各种各样特定的功能，让这种限制变得越来越令人不快。

在2006年，Google的一个大胆的工程师Simon Stewart启动了一个称之为WebDriver的项目。Google已经是Selenium的一个重量级用户，但测试者不得不工作在有限制的产品上。Simon想要一个测试工具。可以使用浏览器和操作系统的“本地”方法与浏览器直接对话，因此可以避免沙盒Javascript环境的约束。WebDriver项目开始致力于解决Selenium的痛点。

到2008年，北京的奥林匹克运动会标志着中国作为一个全球力量的到来，在美国庞大的抵押贷款违约触发了自大萧条以来最糟糕的国际衰退，黑暗骑士被人们看见了两次，还要遭受不合时宜的失去希斯·莱杰（希斯·安德鲁·莱杰，Heath Andrew Ledger，1979年4月4日 - 2008年1月22日，澳大利亚男演员，曾以《黑暗骑士》一片获得第81届奥斯卡金像奖）的不良影响。但那一年最重大的故事是Selenium和WebDriver的融合。Selenium有着庞大的社区和商业支持，但WebDriver无疑地是这个工具的未来。两个工具的结合为所有的用户提供了一个公共的特征集，并且带来了某些最耀眼地、在同一个屋檐下的测试自动化思想。或许为何WebDriver和Selenium进行融合的最佳说明是Simon Stewart在2009年8月6日给WebDriver和Selenium社区的一封邮件中的陈述，

为何两个项目合并？部分是因为WebDreiver标定了某些在Selenium中的不足之处（例如能够去旁路JS沙盒，并且我们有一个华丽的API），部分是因为Selenium标定了某些WebDriver的不足之处（诸如对广泛的浏览器的支持），部分是因为主要的Selenium贡献者和我觉得这是提供用户最佳的可能框架的最佳方式。

Selenium工具套件

Selenium是由多个软件工具组成的。每一个有一个特定的角色

Selenium2 (亦称Selenium WebDriver)

Selenium 2是该项目的未来方向，和对Selenium工具包的最新的增加物。这崭新的自动化工具提供了各种各样的令人敬畏的特征，包括一个更有聚合力和面向对象的API，以及一个对旧的实现的限制的解决方法。

正如你在Selenium项目简史中读到的，Selenium和WebDriver的开发者一致同意两个工具各有优势，并且合并这两个项目将产生一个更健壮的自动化工具。

Selenium 2.0是该项努力的产物。它支持WebDriver API及其潜在的技术，以及Selenium 1的技术在WebDriver的底层，为了最大化移植你的测试的灵活性。此外，为了向后兼容Selenium 2仍然运行Selenium 1的Selenium RC接口。

Selenium 1 (亦称Selenium RC或远程控制)

正如你在Selenium项目简史中读到的，长期以来Selenium RC是最重要的Seleniumx项目，在WebDriver/Selenium合并产生Selenium 2，这最新的和更加强有力的工具之前。

Selenium 1 仍然被支持（大部分是在维护模式），并且提供了某些在Selenium 2可能暂时没有提供的特征，包括对几个语言（Java，Javascript，Ruby，PHP，Python，Perl和C#）的支持，以及对几乎每种浏览器的支持。

Selenium 集成开发环境

Selenium IDE（集成开发环境）是一个用于构造测试脚本的原型工具。它是一个Firefox插件，并且提供了一个易于使用的开发自动化测试的接口。Selenium IDE有一个录制功能，可以记录用户执行的动作，然后可以导出它们作可重用的脚本，以许多种编程语言中的一种，稍后可以被执行。

注释

虽然Selenium IDE有一个“保存（Save）”功能，那个允许用户去以基于表格的形式保存这些测试，为了稍后的导入和执行，但它不是设计用于运行你的测试，也不是涉及用于构造所有你将需要的自动化测试。特别是Selenium IDE不为测试脚本提供迭代或条件语句。在编写的时候没有计划去增加这些功能。这理由部分是技术上的，部分是基于Selenium的开发者鼓励在测试自动化中总是需要一定数量的编程工作的最佳实践。Selenium IDE仅仅打算做为一个快速原型工具。Selenium的开发者很认真地推荐，健壮的测试自动化应该使用Selenium 2或Selenium 1，用许多支持的编程语言中的一种。

Selenium栅格

Selenium栅格允许Selenium RC解决方案扩展为大规模的测试集，和为那些必须运行在多个环境下的测试集。

Selenium栅格允许你并行地运行你的测试，也就是说，不同的测试可以在相同的时间运行在不同的远程机器上。这有

两个好处。首先，如果你有一个大规模的测试集，或一个运行缓慢的测试集，你可以增加它的性能，通过使用Selenium栅格去划分你的测试集，运行不同的测试在同一时间在不同的机器上。同样，如果你必须运行你的测试集在多个环境，你可以有不同的远程机器的支持和运行你的测试在同一时间在不同的远程机器上。在任何一种情形下，Selenium都将充分利用并行处理，极大地改善运行你的测试所花费的时间。

Selenium工具选择

许多人从Selenium IDE开始，如果你还没有编程或脚本语言的经验，你可以使用Selenium IDE去熟悉Selenium命令。使用IDE你可以快速地创建简单的测试，有时甚至在几秒内。

然而，我们不推荐你使用Selenium IDE做所有的自动化测试。为了有效地使用Selenium，你需要构造和运行你的测试，使用Selenium 2或Selenium 1，连同一种支持的编程语言。选择哪种编程语言取决于你。

在编写本文档的时候,Selenium的开发者正计划使Selenium-WebDriver成为Selenium的未来方向。Selenium 1被提供为了向后兼容。我们将在相应的章节讨论两者的优缺点。

我们推荐Selenium的完全新手应通读所有的章节。然而对那些正在使用Selenium并且正在从头开始构造一个新的测试的开发者，你或许想要选用Selenium 2，因为这是Selenium将在未来继续支持的部分。

支持的浏览器平台

在Selenium 2.0, 支持的浏览器将依赖于你使用的是Selenium-WebDriver还是Selenium-RC而变化。

Selenium-WebDriver

Selenium-WebDriver支持下面的浏览器，以及这些浏览其兼容的操作系统。

- > Google Chrome 12.0.712.0+
- > Internet Explorer 6, 7, 8, 9 - 32 and 64-bit
- > Firefox 3.0, 3.5, 3.6, 4.0, 5.0, 6, 7
- > Opera 11.5+
- > HtmlUnit 2.9
- > Android – 2.3+ 为移动电话和平板电脑(设备或模拟器)
- > iOS 3+ 移动电话(设备或模拟器) 以及3.2+ 平板电脑(设备或模拟器)

注释：在编写文档时，Android2.3有一个模拟器bug，会妨碍驱动器在设备模拟器上正常的工作。然而，在平板电脑模拟器和真实的设备上工作良好的。

Selenium 1.0和Selenium-RC

这是旧的Selenium 1.0的支持平台。它应该仍然适用于Selenium 2.0的Selenium-RC。

Browser	Selenium IDE	Selenium 1 (RC)	Operating Systems
Firefox 3.x	录制和回放测试	启动浏览器运测试	Windows, Linux, Mac
Firefox 3	录制和回放测试	启动浏览器运测试	Windows, Linux, Mac
Firefox 2	录制和回放测试	启动浏览器运测试	Windows, Linux, Mac
IE 8	测试执行只能通过Selenium RC*	启动浏览器运测试	Windows
IE 7	测试执行只能通过Selenium RC*	启动浏览器运测试	Windows
IE 6	测试执行只能通过Selenium RC*	启动浏览器运测试	Windows
Safari 4	测试执行只能通过Selenium RC	启动浏览器运测试	Windows, Mac
Safari 3	测试执行只能通过Selenium RC	启动浏览器运测试	Windows, Mac
Safari 2	测试执行只能通过Selenium RC	启动浏览器运测试	Windows, Mac
Opera 10	测试执行只能通过Selenium RC	启动浏览器运测试	Windows, Linux, Mac
Opera 9	测试执行只能通过Selenium RC	启动浏览器运测试	Windows, Linux, Mac
Opera 8	测试执行只能通过Selenium RC	启动浏览器运测试	Windows, Linux, Mac
Google Chrome	测试执行只能通过Selenium RC	启动浏览器运测试	Windows, Linux, Mac
Others	测试执行只能通过Selenium RC	可能部分支持**	如适用

*通过Selenium IDE在Firfox上开发的测试，可用通过一个简单的Selenium RC命令行在任何其它的浏览器上执行。

**Selenium RC服务器可以启动任何执行，但依赖于浏览器的安全设置，可能有技术上的限制将限制某些功能。

适应性和扩展性

你将发现Selenium是高度灵活的。有许多方法你可以增加功能到Selenium测试脚本和Selenium框架来自定义你的测试自动化。这或许是Selenium与其它自动化工具比较而言最伟大的优势。这些自定义方法描述在贯穿本文档的各个地方。此外，因为Selenium是开源的，源代码总是可以被下载和修改。

本文档包含内容

用户指南的以新用户和那些已经使用过Selenium但仍在寻找附加的知识的目标。我们介绍Selenium给新用户，并且我们不假定先前有Selenium的经验。然而，我们假定用户至少有一个队测试自动化的基本理解。对那些有经验的用户，本指南可以作为一个参考资料。对有经验的用户，我们推荐浏览章节和子标题（选择阅读）。我们提供了有关Selenium架构的信息，常见用法的示例，和一个有关测试设计技术的章节。

本参考在剩下的章节里提供了：

Selenium IDE介绍了Selenium IDE以及描述了如何使用Selenium集成开发环境构造测试脚本。如果你在编程方面没有多少经验，但仍然希望学习测试自动化，这是你应该开始的地方，你会发现你可以使用Selenium IDE创建一些自动化的测试。同样如果你是一个在编程方面有经验的用户，这一章可能仍然对你来说是有趣，因为你可以使用Selenium IDE来开发你测试的快速原型。这一章也演示了如何“导出”你的测试脚本到一种支持的编程语言，为了增加更多的Selenium IDE不支持的高级能力。Selenium 2 解释了如何使用Selenium 2 开发一个自动化测试程序。Selenium 1 解释如何使用Selenium RC API开发一个自动化的测试程序。提供了许多示例使用编程语言和脚本语言。同样，Selenium RC的安装和设置也被包含在这一章。描述了Selenium RC支持的各种各样的模式或配置，以及他们的优缺点和限制。提供了一个构架图有助于演示这些关键点。对新的Sel-R用户感觉困难的经常发生的公共问题的解决方案也在本章描述，例如，处理安全证书，https请求，弹出和打开一个新的窗口。测试设计考虑这一章提供了Selenium-WebDriver和Selenium RC使用的编程技术。我们也演示了一些在用户论坛上经常提及的一些技术，诸如如何设计setup和teardown函数，如何实现数据驱动的测试（可以在多次测试时改变数据的测试），以及其他的公共的测试自动化编程方法。Selenium栅格这一章仍然没有被开发出来。用户扩展描述了Selenium可以被修改，扩展和自定义的方法。

文档团队 - 作者们的过去和现在

按字母的顺序，下面的人员对本用户指南的创作，它的发布结构或两者做出了重要的贡献。我们非常感激他们所有的人。

Dave Hunt

Luke Inman-Semeran

Mary Ann May-Pumphrey

Noah Sussman

Paul Grandjean

Peter Newhook

Santiago Suarez-Ordonez

Simon Stewart

Tarun Kumar

致谢

特别感谢Patrick Lightbody。作为一个SeleniumHQ站点的管理员，对Selenium RC的主要贡献者，他的支持是无价的，对编写本用户指南的第一个发布。Patrick帮助我们理解我们的读者。他为了我们每一个人发布本文档到seleniumhq.org提供了所需的一切。同样要感谢Andras Hatvani 有关在发布解决方案上的建议，感谢Amit Kumar参与我们的讨论并协助审核本文档。

当然，我们必须对Selenium的开发者致谢。他们真的设计了一个令人惊奇的工具。没有最初的设计者的想象力，和现在的开发者的不断地努力，我们就不会有这样一个很棒的工具需要推荐给你。

1.3 Selenium用户指南 - 第三章 Selenium IDE[1]

发表时间: 2012-02-12

入门

Selenium-IDE (集成开发环境) 是一个使用于开发Selenium测试案例的工具。它是一个易于使用的Firefox插件，并且是通常是开发测试案例最有效的方式。它也包含一个上下文菜单，允许你首先从浏览器目前显示的页面中选择一个UI元素，然后按照这选择的UI元素的上下文，从带有预定义的参数的Selenium命令列表中选择一个命令。它不仅仅是一个节省时间的工具，也是一个学习Selenium脚本语法的极佳的方法。

本章都是有关Selenium IDE以及如何有效使用它的内容。

安装IDE

首先使用Firefox从SeleniumHQ[下载](http://seleniumhq.org/download/)页下载IDE。

Firefox会保护你免于从不熟悉的位置安装插件，如此你将需要点击‘允许 (Allow)’去继续这个安装，正如显示在下面的屏幕快照。

但从Firefox下载时，你将看到下面的窗口。

IDE特征

菜单条

文件（File）菜单有创建测试案例和测试集（测试案例的集合）的选项。使用这些菜单项你可以增加一个新的测试案例，打开一个测试案例，保存一个测试案例，以选择的语言导入一个测试案例。你也可以打开最近的测试案例。所有这些选项对测试集也是可用的。

编辑（Edit）菜单允许复制，粘贴，删除，取消和选择全部操作，在你的测试案例中用于编辑命令。选项（Options）菜单允许改变设置。你可以设置某些命令的超时值，增加用户定义的用户扩展到基本的Selenium命令集，指定保存你的测试案例使用的格式（语言）。帮助（Help）菜单是标准的Firefox帮助菜单；仅仅包含一个属于Selenium IDE的菜单项UI-Element-Documentation

工具条

工具条包含用于控制测试案例执行的按钮，包括一个用于调试你的测试案例的单步（step）测试按钮。最右边的那个带有红点的按钮是录制（record）按钮。

运行（Run）：运行目前选择的测试。当只有一个测试案例被装载时，该按钮和运行所有按钮有相同的效果。

测试运行器模式（TestRunner Mode）：允许你在一个Selenium内核测试运行器装载的浏览器中运行测试。测试运行器现在通常不会使用，很可能将被废弃。该按钮主要为了向后兼容测试运行器的目的。大多数的用户可能不需要这个按钮。

如果你在命令（Command）域开始输入，将显示一个基于你输入的字符填充的下拉列表；你可以从下拉列表中选择你想要的命令。

日志（Log）/参考（Reference）/UI元素（UI-Element）/分组（Rollup）窗格

底部的窗格提供四个不同的功能 - 日志 (Log) , 参考 (Reference) , UI元素 (UI-Element) , 分组 (Rollup) 。

日志 (Log)

当你运行测试案例时，错误消息和显示进度的信息性消息被自动显示在这个窗格，即使你没有首先选择日志 (Log) 选项页。这些消息对测试案例的调试通常是有用的。注意 “Clear” 按钮用于清除日志。同样注意 “Info” 按钮是一个下拉列表，允许选择需要日志的不同级别的信息。

尽管参考 (Reference) 选项页作为一个快速的参考是非常有价值的，但仍然需要经常地查阅[Selenium参考文档](chrome://selenium-ide/content/selenium-core/scripts/ui-doc.html)。

UI元素和分组

关于这两个选项页 (涉及高级功能) 的详细信息可以在帮助 (Help) 菜单的UI-Element文档中找到。

© Copyright 2008-2012, Selenium Project. Last updated on Feb 02, 2012.

1.4 Selenium用户指南 - 第三章 Selenium IDE[2]

发表时间: 2012-02-12

构造测试案例

有三个主要的用于开发测试案例的方法。经常地，一个测试的开发者会需要所有这三个技术。

录制

许多刚上手的用户从他们与一个Web站点的交互中录制一个测试案例起步。当Selenium被首次打开时，录制（Record）按钮默认为开启。如果你不希望Selenium IDE开始自动的录制，你可以通过选项（Options）菜单的选项（Options...）菜单项，打开选项对话框，和取消选择“Start recording immediately on open”关闭此它

在录制期间，Selenium IDE会基于你的动作自动插入命令到你的测试案例。典型的，这将包括：

点击一个链接 - click 或 clickAndWait命令

键入值 - type命令

从下拉列表框选择一个选项 - select命令

点击复选框或单选按钮 - click命令

这里是一些需要注意的“陷阱”：

type命令可能需要在Web页的某个其它的地方点击才会录制。

点击一个链接通常录制一个click命令。你通常需要改变它到clickAndWait命令，以便确保你的测试案例暂停，直到新的页面被完全地装载。否则，你的测试案例会继续运行命令在页面已经装载所有它的UI元素前。这会引引起一个未预期的测试案例失败。

用上下文菜单增加验证和断言

你的测试案例也需要检查Web页的属性。这就需要assert和verify命令。我们不会再这里描述这些命令的具体细节；那将在Selenium命令 - “Selenese” 章节中描述。在此我们仅仅描述如何去增加它们到你的测试案例。

在使用Selenium IDE进行录制过程中，在显示你的 测试应用程序的浏览器中，右点页面的任何地方。你会看到一个显示verify和/或assert命令的上下文菜单。

首次使用Selenium，可能仅仅是列出一个Selenium命令，当你使用IDE时，你将发现附加的命令会增加到这个上下文菜单。Selenium IDE会试图预测在目前的Web页面上，你选择的UI元素将需要的命令及其参数。

让我们来看看这是如何工作的。打开一个你选择的Web页，并且在页面上选择一个文本块。一个段落或一个标题就可以。现在，右点选择的文本。上下文菜单应该给予你一个verifyTextPresent命令和支持的参数应该是这文本本身。

同样，注意这显示所有可得到的命令（ Show All Available Commands ）菜单选项。这将显示更多的命令，以及支持的参数，为测试你目前选择的UI元素。

试试更多的UI元素。试着右点一个图像，或一个用户控件，像一个按钮或复选框。你可能需要使用显示所有可得到的命令（ Show All Available Commands ）去看看更多的选项而不是verifyTextPresent。一旦你选择这些其它的选项，比较常用的一些命令将会显示在主上下文菜单上。例如，选择一个图像选择verifyElementPresent将稍后引起那个命令在主上下文菜单时是可得到的，在你下次选择和右点一个图像时。

再次说明，这些命令将在Selenium命令章节中详细解释。不过现在，请使用IDE去录制和选择命令进入你的测试案例并运行它。你可以了解许多Selenium命令，仅仅通过用IDE。

编辑

插入命令

Table视图

在你的测试案例中选择一个你希望插入命令的插入点。在测试案例窗格中，左点你希望插入一个新命令的行，然后，右点和选择插入新命令（ Insert New Command ）；IDE将增加一个空白行在你选择的行的前面。现在可以使用命令（ Command ）编辑文本域去输入新的命令和它的参数。

源代码视图

在你的测试案例中选择一个你希望插入命令的插入点。在测试案例窗格，在你希望插入新命令的命令之间左点，然后键入要求的HTML标记去创建一个三列的行，包括命令，第一个参数（如果命令需要一个），和第二个参数（再次说明，如果需要）。确信保存你的测试在切换回Table视图前。

插入注释

增加注释可以让你的测试案例更具有可读性。当测试被运行时，将忽略这些注释。

注释也可以用来在你的测试中增加垂直空白（一个或多个空白行）；只是创建一个空白的注释。一个空的命令会在执行期间引起一个错误，但空注释不会。

Table视图

在测试案例中选择你希望插入注释的行。右点并选择插入新注释（Insert New Comment）。现在可以使用命令（Command）域键入注释。你的注释将显示作紫色的文本。

Source视图

在测试案例中选择你希望插入注释的点。增加一个HTML风格的注释，例如，`<!-- 这儿是你的注释 -->`。

编辑命令或注释

Table视图

仅仅选择需要改变的行，然后使用命令（Command），目标（Target）和值（Value）域进行编辑。

Source视图

因为源代码视图提供了WYSUWYG（所见即所得）的编辑器，你只需要直接修改你希望修改的命令，参数或注释。

打开和保存一个测试案例

如同大多数程序，在文件（File）菜单下有保存和打开命令。然而，Selenium区分测试案例和测试集。要保存你的Selenium IDE测试为以后使用，你可以保存各个测试案例，或保存测试集。如果你的测试集中的测试案例还没有保存，将会先于保存测试集前，提示你保存测试案例。

当打开一个现存的测试案例或测试集时，Selenium IDE会显示它的Selenium命令在测试案例窗格。

运行测试案例

IDE为运行你的测试案例提供了许多选项。你可以一次运行一个测试案例，停止和启动它，一次运行一行，运行一个你目前正在开发的单一的命令，并且你可以批运行一个完整的测试集。在IDE，测试案例的运行非常地灵活。

运行一个测试案例

点击运行（Run）按钮去运行当前显示的测试案例。

运行一个测试集

点击运行所有（Run All）按钮运行目前装载的测试集中的所有测试案例。

停止和启动

暂停（Pause）按钮可以停止一个正在运行的测试案例。按钮的图标会改变成重新开始（Resume）按钮。继续点击将重新开始运行。

在中途停止

你可以在测试案例中设置一个断点，引起测试案例停止在一个特定的命令上。这对调试你的测试案例是有用的。设置一个断点，请选择一个命令，右点，然后从上下文菜单选择切换断点（Toggle Breakpoint）

从中途启动

你可以告诉IDE，在你的测试案例的中间，从一个特定的命令开始运行。这也被用于调试。要设置一个启动点，选择一个命令，右点，然后从上下文菜单选择设置/清楚启动点（Set/Clear Start Point）

运行任何单一的命令

双击任何单一的命令去运行它自己。这是有用的，当编写一个单一的命令。它允许你立即测试一个你正在构造的命令，但你不确信是否它是正确的。你可以双击运行看看是否它可以正确地运行。这也是可得到的从上下文菜单。

使用基URL运行测试案例在不同的域（Domain）

基URL（Base URL）域在Selenium IDE窗口的顶部，它对允许测试案例被运行在跨越不同的域（Domain）非常有用。假定一个命名为http://news.portal.com 的站点有一个内部的beta站点命名为http://beta.news.portal.com。任何为这些站点的测试案例，在open语句中都应该指定一个相对的URL作为参数，而不是绝对URL（诸如使用http：或https：协议开始的URL）。Selenium IDE将创建一个绝对URL，通过添加open命令的参数到基URL的末尾。例如，下面的测试案例会运行在http://news.portal.com/about.html:

_images/chapt3_img21_BaseURL_prod.png

相同的测试案例，带有修改的基URL设置会运行在http://beta.news.portal.com/about.html:

_images/chapt3_img22_BaseURL_beta.png

© Copyright 2008-2012, Selenium Project. Last updated on Feb 02, 2012.

1.5 Selenium用户指南 - 第三章 Selenium IDE[3]

发表时间: 2012-02-12

Selenium 命令 - “Selenese”

Selenium命令，常被称为Selenese，是一套运行在你的测试中的命令。这些命令的一个序列是一个测试脚本。在此，我们将详细解释这些命令，并提供你许多选择，你可以在使用Selenium测试你的应用程序中使用。

Selenium提供了一套丰富的命令集，为了可以完全地，以你可以想象的任何方式测试你的Web应用程序。命令集常常被称为Selenese。这些命令在本质上创建了一种测试语言。

在Selenese，人们可以测试现存的UI元素，基于HML标记，可以测试指定的内容，测试中断的链接，输入域，选择列表选项，提交窗体，以及其他内容之间的表格数据。此外Selenium命令支持测试窗口的大小，鼠标的位置，警告，Ajax功能，弹出窗口，事件处理，以及许多其他的Web应用程序特征。命令参考列出了所有可得到的命令。

一个命令就是告诉Selenium应作是么。Selenium命令可以分成三类：动作（Action），存取器（Accessor）和断言（Assertion）。

动作（Action）通常是操纵应用程序状态的命令。它们完成类似“点击链接”和“选择选项”的工作。如果一个动作（Action）失败，或者有一个错误，当前测试的执行将终止。

许多动作（Action）可以被调用带有“AndWait”后缀，例如“clickAndWait”。这个后缀告诉Selenium该动作将引起浏览器做出一个对服务器的调用，Selenium将等待一个新的页面被装载。

存取器（Accessor）检查应用程序的状态，并存储结果在变量中，诸如“storeTitle”。它们也被使用于自动生成断言。

断言（Assertion）类似于存取器，但它们验证应用程序的状态符合预期。示例包括“确信页面的标题是X”和“验证复选框被选中”。

所有的Selenium断言（Assertion）可以以三个方式进行使用：“assert”，“verify”和“waitFor”。例如你可以“assertText”，“verifyText”和“waitForText”。当一个“assert”失败时，测试被退出。当一个“verify”失败时，测试将继续执行，日志这个失败。这允许一个单一的“assert”来确保应用程序是在正确的页面上，跟着是一堆“verify”断言，进行窗体域值及标签等等的测试。

“waitFor”命令等待某些条件成真（可能对Ajax应用程序的测试是有用的）。它们将立即成功，如果条件已经为真。然而，它们将失败和停止测试，如果条件没有在目前的超时设置内成真（见后面的setTimeout动作）。

脚本语法

Selenium命令是简单的，它们由命令和两个参数组成，例如：

```
verifyText //div//a[2] Login
```

参数并不总是需要的；这依赖于命令。在某些情况下，需要两个参数，一个参数，或者根本就不带参数。这里是几个示例：

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

命令参考为每一个命令描述了参数的需求情况。

参数是多种多样的，然而它们典型的是：

识别页面中UI元素的定位器

验证或断言预期的页面内容的文本模式

用于在一个输入域键入文本或在一个选项列表选择一个选项的文本模式或Selenium变量

定位器，文本模式，Selenium变量，以及命令本身在Selenium命令章节有相当详细的描述。

从Selenium IDE运行的Selenium脚本以一个HTML文本文件的格式存储在一个文件中。它是由带有三列的一个HTML表格组成的。第一列标示Selenium命令，第二列是目标（Target），最后一列包含一个值（Value）。第二和第三列依赖于选择的Selenium命令可能不需要值，但它们必须被呈现。每一个表行代表一个新的Selenium命令。这是一个测试的示例，完成打开一个页。断言页的标题，然后验证页面上的某些内容：

```
open      /download/  
assertTitle Downloads  
verifyText Downloads
```

在浏览器中绘制作一个表，看起来如下：

```
open /download/  
assertTitle Downloads  
verifyText //h2 Downloads
```

Selenese的HTML语法可以用于编写和运行测试，而无需编程语言的知识。只要具备基本的Selenese和Selenium IDE的知识，你就可以快速地创作和运行测试案例。

测试集

一个测试集是测试的集合。时常一个人会运行在一个测试集中的所有测试做一个连续的批作业。

当使用Selenium IDE时，测试集可以被定义使用一个单一的HTML文件。语法同样是简单的。一个HTML表格定义一个测试列表，每一行定义每一个测试的地文件系统路径。一个示例就可以说明一切：

```
<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>
```

一个类似于这个的文件就可以允许从Selenium IDE一次运行所有的测试。一个接着一个。

测试集也可以使用Selenium RC来维护。这可以通过编程来达到，同样可以以多种方式被完成。普通的单元测试可以被使用于维护一个测试集，如果一个人正在使用Selenium RC用Java。替代地名如果C#是选择的语言，也可以使用Nunit。如果使用一个解释性语言，像Python，使用Selenium RC可能需要在建立一个测试集时进行一些简单的编程工作。因为使用Selenium RC的全部理由就是可以充分利用编程的逻辑于你的测试，这通常不应该是一个问题。

常用的Selenium命令

总结一下我们的Selenium简介，我们向你展示了几个典型的Selenium命令。这些可能就是最常用的构造测试的命令。

`open`

使用URL打开一个页面。

`click/clickAndWait`

执行一个点击操作，可选的等待一个新的页面被装载。

`verifyTitle/assertTitle`

验证一个预期的页面标题。

`verifyTextPresent`

验证在页面某个地方的预期文本。

`verifyElementPresent`

验证由HTML标记定义的预期UI元素在页面上是否呈现。

`verifyText`

验证预期文本和它响应的HTML标记在页面上呈现。

`verifyTable`

验证一个表的预期内容。

`waitForPageToLoad`

暂停执行直到一个预期的新页面被装载。当`clickAndWait`被使用时将自动调用。

`waitForElementPresent`

暂停执行直到一个预期的由HTML标记定义的元素出现在页面上。

验证页面元素

验证一个页面上的UI元素可能是你的自动化测试最常用的功能。Selenese允许以多种方式检测UI元素。理解这些不同的方法是重要的，因为这写方法定义了你的实际测试内容。

例如，你将测试...

一个元素在页面的某个地方出现了么？

特定的文本在页面的某个地方出现了么？

特定的文本是在页面上的某个特定的位置么？

例如，如果你正在测试一个文本标题，这文本和它的在页面顶部的位置可能是与你的测试相关的。然而，如果你正在测试在一个主页面上一个图像的存在性，而且Web的设计者频繁地改变这个特定的图像文件，以及它在页面上的位置，那么你仅仅希望去测试一个图像（非特定的图像文件）在页面的某个地方。

断言或验证？

在“assert”和“verify”之间的选择取决于失败的管理和便利性。当检查页面上的第一个段落是否正确几乎是不重要，即使你失败了，但浏览器正在显示一个正确的页面的时候。如果你没有在正确的页面，你可能希望推出你的测试案例，以便立即调查原因和快速地修复问题。另一方面，你可能希望检查一个页面的许多属性，而在首次失败是不退出测试，因为这将允许你查看在页面上的所有失败，并采取适当的行动。实际上一个“assert”将使测试失败并退出当前的测试案例，然而“verify”将使测试失败并继续运行测试案例。

这个特征的最佳使用方法是逻辑地分组你的测试命令，在每个分组的开始带有一个“assert”命令，跟着一个或多个“verify”测试命令。示例如下：

```
Command Target Value
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
```

上面的示例，首先打开一个页，然后通过比较页面的标题与预期的值“assert”正确的页面被装载。只要这个通过，后面的命令将运行并“verify”文本被呈现在预期的位置。测试案例然后“assert”第一个表的第二行的第一列包含预期的值，并且仅当这个通过了，再继续“verify”那一行的剩余的单元格。

verifyTextPresent

该命令用于验证特定的文本存在页面的某个地方。带有一个单一的参数 - 需要验证的文本模式。例如：

Command Target Value

`verifyTextPresent Marketing Analysis`

这将引发Selenium搜索，验证，文本串“Marketing Analysis”出现在正在验证的页面的某个地方。使用`verifyTextPresent`，当你仅仅对文本本身出现在页面上感兴趣时。不要使用它，当你需要测试文本出现在页面的哪里时。

使用这个命令，当你测试一个特定的UI元素必须出现，而不是它的内容时。这个验证部检查文本，仅只HTML标记。一个常见的使用检查一个图像的呈现。

Command Target Value

`verifyElementPresent //div/p/img`

这个命令验证一个图像，使用一个HTML标记，紧跟在<div>和<p>标记的后面。第一个（也是唯一的一个）参数是定位器，告诉Selenese命令如何找到这个元素。定位器被解释在下一节。

`verifyElementPresent` 可以被使用于在页面中检查任何HTML标记的存在性。你可以检查链接，段落，层等的存在性。这儿是几个示例：

Command Target Value

`verifyElementPresent //div/p`

`verifyElementPresent //div/a`

`verifyElementPresent id=Login`

`verifyElementPresent link=Go to Marketing Research`

`verifyElementPresent //a[2]`

`verifyElementPresent //head/title`

这些示例演示了一个UI元素可以被测试的各种各样的方式。再次说明，定位器被解释在下一节。

verifyText

使用`verifyText`，当文本以及它的UI元素必须被测试时。`verifyText`必须使用一个定位器。如果你选择XPath或DOM定位器，你可以验证特定的文本出现在页面上，相对于其他UI组件的特定的位置。

Command Target Value

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

1.6 Selenium用户指南 - 第三章 Selenium IDE[4]

发表时间: 2012-02-16

定位元素

对大多数Selenium命令，一个目标（Target）是必须的。目标标识在一个Web应用程序的内容中的一个元素，并且由一个定位策略跟着一个位置组成，以locatorType（定位类型）=location（位置）的形式出现。在许多情况下，定位类型可以忽略。各种各样的定位类型解释如下，每一个都带有示例。

按identifier（按标识符）定位

这可能是最常用的定位元素的方法，它是包罗万象的缺省，当没有可识别的定位类型被使用时。应用这个策略，带有id属性匹配位置值的第一个元素将被使用。如果没有元素包含一个匹配的id属性，带有name属性匹配位置值的第一个元素将被使用

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>
```

下面的定位器策略将从以上的HTML代码段返回由行号标示的元素

identifier=loginForm (3)

identifier=password (5)

identifier=continue (6)

continue (6)

因为标识符类型定位器是默认的，在上面的前三个示例中的identifier=不是必须的。

按id定位

这种类型的定位器是比标识符（ identifier ）有更多限制的定位器，但也更明确。当你知道元素的id属性时使用它。

```
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" />
8 </form>
9 </body>
10 </html>
```

id=loginForm (3)

按name定位

name定位器类型将定位第一个带有一个匹配的name属性的元素。如果多个元素有相同的name属性值，则你可以使用过滤器进一步改善你的定位策略。默认的过滤器类型是值（匹配值属性）

```
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" />
8 </form>
9 </body>
10</html>
```

name=username (4)

name=continue value=Clear (7)

name=continue Clear (7)

name=continue type=button (7)

Note

注释

不像某些XPath和DOM定位器类型，上面三个定位器类型允许Selenium，不依赖于UI元素在页面上的位置来测试UI元素。因此即时页面的结构发生变化，测试仍然可以通过。你可能或可能不希望测试页面的结构是否发生变化。在Web页面的设计者频繁地改变页面，但页面的功能必须进行回归测试的情况下，通过id和name属性进行测试，或通过任何HTML属性进行测试，将变得非常重要。

按XPath来定位

XPath是XML文档中用于定位节点的语言。因为HTML可能是一个XML的实现（XHTML），Selenium用户可以利用这强有力的语言去定位他们的Web应用程序中的元素。XPath扩展了最简单的按id和name属性定位的方法（同样支持），释放了各种各样的可能性，例如定位页面上的第三个复选框。

使用XPath的主要的理由之一是你可能没有你希望定位的元素的适当的id或name属性。你可以使用XPath，要么以绝对方式（不建议）定位元素，要么以相对于一个有id或name属性的元素进行相对地定位。XPath定位器也可以按非id和name属性的属性定为元素。

绝对定位的XPath包含从根（html）开始的所有元素的定位，作为一个结果，即使应用程序进行最不重要的调整也可能导致定位的失败。通过查找一个带有id或name属性的邻近元素（理想地是一个父元素），你可以基于相对的关系定位你的目标元素。这种关系不太可能改变，这种定位方式会让你的测试更健壮。

因为只有XPath定位器以“//”开始，当指定一个XPath定位时，无需包含xpath=标签。

```
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" />
8 </form>
9 </body>
10</html>
```

xpath=/html/body/form[1] (3) - Absolute path (would break if the HTML was changed only slightly)

//form[1] (3) - First form element in the HTML

xpath=//form[@id='loginForm'] (3) - The form element with attribute named 'id' and the value 'loginForm'

xpath=//form[input/@name='username'] (4) - First form element with an input child element with attribute named 'name' and the value 'username'

//input[@name='username'] (4) - First input element with attribute named 'name' and the value 'username'

//form[@id='loginForm']/input[1] (4) - First input child element of the form element with attribute named 'id' and the value 'loginForm'

//input[@name='continue'][@type='button'] (7) - Input with attribute named 'name' and the value 'continue' and attribute named 'type' and the value 'button'

//form[@id='loginForm']/input[4] (7) - Fourth input child element of the form element with attribute named 'id' and value 'loginForm'

这些事例包含了某些基础知识，要学习更多的内容，推荐访问下面的资源：

W3Schools XPath Tutorial

W3C XPath Recommendation

还有几个非常有用的Firefox插件，有助于发现一个元素的XPath：

XPath Checker - suggests XPath and can be used to test XPath results.

Firebug - XPath suggestions are just one of the many powerful features of this very useful add-on.

XPath Checker - 建议XPath并可以用于测试XPath的结果

Firebug - XPath建议仅仅是这非常有用的插件的许多强有力特征中的一个。

Locating Hyperlinks by Link Text

按链接文本定位超级链接（link定位器）

这是一个通过使用超级链接的文本，定位Web页面中的超级链接的简单方法。如果两个超级链接带有相同的文本，则时用第一个匹配。

1<html>

2 <body>

3 <p>Are you sure you want to do this?</p>

4 Continue

5 Cancel

6</body>

7<html>

link=Continue (4)

link=Cancel (5)

Locating by DOM

按DOM定位

文档对象模型代表一个HTML文档，可以使用Javascript访问。这个定位策略使用Javascript对页面上的元素进行计算，这样可以层次的.标记法简化元素的定位。

因为只有dom定位器以“document”开始，当指定一个DOM定位器时，dom=标签不是必需的。

```
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" />
8 </form>
9 </body>
10 </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)
```

你可以使用Selenium本身已经其他的站点以及延伸站点去研究你的Web应用程序的DOM。在W3Schools上有很好的可供参考的资料。

按CSS定位

CSS（级联样式表）是用于描述HTML和XML文档应如何绘制的语言。CSS使用选择器绑定样式属性到文档中的元素。

```
1 <html>
2 <body>
```

```
3 <form id="loginForm">
4 <input class="required" name="username" type="text" />
5 <input class="required passfield" name="password" type="password" />
6 <input name="continue" type="submit" value="Login" />
7 <input name="continue" type="button" value="Clear" />
8 </form>
9 </body>
10 <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"] (4)

css=#loginForm input:nth-child(2) (5)

有关CSS选择器的更多信息，最好的地方是W3C发布站点。在那里你可以找到其他的参考资料。

注释

绝大多数有经验的Selenium用户推荐CSS作为他们的选择的定位策略，因为它比XPath定位速度要快很多，并且可以在一个固有的HTML文档中找到最复杂的对象。

隐含定位器

在下面的情况下，你可以选择忽略定位器类型：

没有显式地定义定位器策略的定位器会默认使用identifier定位器策略。见按identifier定位。

以 “//” 开始的定位器将使用XPath定位器策略。见按XPath定位。

以 “document” 开始的定位器将使用DOM定位器策略。见按DOM定位。

© Copyright 2008-2012, Selenium Project. Last updated on Feb 02, 2012.

1.7 Selenium用户指南 - 第三章 Selenium IDE[5]

发表时间: 2012-02-16

匹配文本模式

如同定位器，模式是Selenese命令经常需要的参数类型。需要模式的命令例子，如verifyTextPresent，verifyTitle，verifyAlert，assertConfirmation，verifyText和verifyPrompt。正如上面所提及的，link定位器就是利用一个模式。模式允许你，通过特殊字符的使用，预期的文本来描述，而非精确地指定文本。

有三种类型的模式：通配符（globbing），正则表达式，以及精确（exact）

通配符模式

大部分人熟悉通配符，因为它被使用在DOS的文件扩展名或Unix/Linux的命令行，如ls *.c。在这个示例中，通配符用于显示所有存在于当前目录下的以.c扩展名结束的文件。通配符是相当有限的。只有两个特殊的字符被Selenium的实现所支持。

* 解释为“匹配任何字符”，例如，无，单个字符或多个字符

[]（字元类别）解释为“匹配任何在方括号中发现的任何一个单一的字符”。一个连字符可以用于指定一个字符范围（在ASCII字符集中连续的）的简写方式。几个示例让字元类别的功能变得清楚：

[aeiou] 匹配任何的小写元音字符

[0-9] 匹配任何数字

[a-zA-Z0-9] 匹配任何字母数字字符

在大多数其他应用的上下文中，通配符包含第三个特殊的字符？。然而，Selenium通配符模式只支持*和字元类别。

要为Selenese命令指定一个通配符模式参数，你可以给这个模式带有glob:前缀标签。然而，因为通配符模式是默认的，可以忽略这个标签仅仅指定模式。

下面是两个使用通配符模式的命令示例。在页面上实际被测试的链接的文本是“Film/Television Department”；通过使用模式而不是精确的文本，click命令将工作，即使链接的文本变成了“Film & Television Department”或“Film and Television Department”。通配符模式的*将匹配“任何或无”字符在单词“Film”和“Television”之间。

CommandTarget Value

click link=glob:Film*Television Department

verifyTitleglob:*Film*Television*

由点击超级链接到达的页面的实际标题是“De Anza Film And Television Department - Menu”。通过使用模式匹配而非精确的文本，只要单词“Film”和“Television”出现（以这个顺序）在页面标题任何地方，verifyTitle将通过。例如，页面的所有者可能缩写页面标题为“Film & Television Department”，这个测试仍然将通过。在链接以及其他简单测试（诸如verifyTitle）中使用模式，可以极大地减少此类测试的维护工作量。

正则表达式模式

正则表达式模式是三种Selenese支持的模式类型中最强大的模式。正则表达式也被大多数高级的编程语言支持，许多文本编辑器，许多工具，包括Linux/Unix命令行实用工具grep，sed，以及awk。在Selenese，正则表达式模式允许用户执行许多那些多其他的方法来说非常困难的任务。例如，假定你的测试需要确保一个特定的表的单元格只能包含数字。regexp: [0-9]+是一个匹配任何长度的数字的简单模式。

Selenese通配符模式仅仅支持*和[]（字元类别）特征，而Selenese正则表达式模式则提供了与在Javascript中同样广泛的特殊字符。下面是这些特殊字符的一个子集：

PATTERN MATCH

. 任何单个字符

[] 字符类别：任何出现的括弧中的单个字符

* 数量：0或多个前面的字符（或组）

+ 数量：1或多个前面的字符（或组）

? 数量：0或1个前面的字符（或组）

{1,5} 数量：1到5个前面的字符（或组）

| 可选：在左边或右边的字符/组

() 分组：常用于可选和/或数量

在Selenese中，正则表达式模式需要带有regexp: 或regexpi: 前缀。前者是大小写敏感的，后者是不敏感的。

几个示例将有助于澄清正则表达式可以如何被使用在Selenese命令中。第一个使用可能是最常用的正则表达式 - “.”。这两个字符序列可以被解释作“0次或多次出现的任何字符”或更简单地，“有或没有”。这等价于通配符模式的*（一个单一的星号）。

CommandTarget Value

clicklink=regexp:Film.*Television Department

verifyTitleregexp:. *Film.*Television.*

上面的事例在功能上与较早的、使用通配符的测试示例相同。唯一的差别是前缀（ regexp: 替代了 glob: ）和这“有或无”模式（ . * 替代了单一的* ）。

下面更复杂的示例测试Yahoo! Weather（天气）页面的Anchorage, Alaska（阿拉斯加的安克雷奇）包含的有关日出的信息。

Command Target Value

open http://weather.yahoo.com/forecast/USAK0012.html

verifyTextPresent regexp:Sunrise: *[0-9]{1,2}:[0-9]{2} [ap]m

让我们逐步地检视一下上面的正则表达式：

Sunrise: * 字符串Sunrise:跟着0或多个空格

[0-9]{1,2} 1个或2个数字（每天的小时）

:字符：（没有特殊字符涉及）

[0-9]{2} 2个数字（分钟）跟着一个空格

[ap]m “a”或“p”跟着“m”

精确匹配

Selenium的精确模式具有边际用途。它根本不使用特殊的字符。如此，如果你需要查找一个实际的*号（对通配符和正则表达式模式是特殊字符），精确模式是完成它的一种方法。例如，如果你希望从下拉列表中选择一个带标签“Real *”的选项，下面的代码可能工作也可能不工作。星号在glob:Real *模式中，会匹配任何字符或无字符。这样，如果有一个更靠前的带有标签“Real Numbers”的选择项将被选择，而不是“Real *”选项。

```
select //selectglob:Real *
```

为了确保“Real *”项被选择，exact:前缀可以使用于创建一个精确匹配模式，如下所示：

```
select//selectexact:Real *
```

当然，可以通过在正则表达式中转义*达成相同的效果：

```
select //select regexp:Real \*
```

对大多数的测试者不太可能需要查找一个*号或一对方括号（通配符中的字元类别）。因此，通配符模式和正则表达式模式对我们大多数人来说是足够的。

“AndWait” 命令

一个命令和它的AndWait替代之间的区别是，规则的命令（例如click）将执行一个动作，然后尽可能快地继续到下一个命令，然而AndWait替代（例如clickAndWait）会告诉Selenium在动作完成后，等待页面被装载。

AndWait替代通常被使用在，当一个动作引起浏览器导航到另外一个页面或重新装载当前的页面的时候。

需要注意的是，如果你使用一个不会触发导航/刷新的动作的AndWait命令时，你的测试会失败。这会发生，因为Selenium会到达AndWait的超时而没有看到任何导航或刷新做出，从而引起Selenium引发一个超时异常。

在AJAX应用程序中的waitFor命令

在AJAX驱动的Web应用程序中，数据从服务器端获取，但不会刷新页面。使用AndWait命令将不会工作，因为页面没有实际的刷新。暂停测试的执行一段时间也不是一个好的方法，因为Web元素可能依赖于系统的响应能力，装载或其他不可控制的因素，而出现在规定的期限的前或后，从而导致测试的失败。最好的方法是在一个动态的期限内等待需要的元素，然后一旦这个元素被发现就继续执行。

这可以完成使用一个waitFor命令，如waitForElementPresent或waitForVisible，动态地等待，每间隔一秒检查要求的条件，一旦条件满足则继续脚本中下一条命令的执行。

计算和流控制序列

当运行一个脚本时，它会简单地按顺序运行，一个命令跟着一个。

Selenese本身不支持条件语句（如if-else）或迭代（如for，while）。许多有用的测试可以无需流控制而执行。然而，对于一个动态内容的功能性测试，可能涉及到多页，编程的逻辑时常是需要的。

当流控制需要时，有三个选项：

运行脚本使用Selenium-RC和一个客户端库，诸如Java或PHP，利用编程语言的流控制功能。

在测试脚本中使用storeEval命令运行一小段Javascript代码段

安装goto_sel_ide.js扩展

大多数的测试者，使用Selenium RC API，导出测试脚本到一个编程语言文件（见Selenium IDE章节）。然而，一些组织，只要可能，更愿意直接运行从Selenium IDE他们的测试脚本（例如，当他们有许多低级别的开

发人员为他们运行测试的时候，或者编程技能不足的时候）。如果这是你的情况，考虑使用Javascript代码段或goto_sel_ide.js扩展。

存储命令和Selenium变量

你可以，在脚本的开始，使用Selenium变量去存储常量。同样，当与一个数据驱动的测试设计结合时（在后面部分讨论），Selenium变量可以用于存储，从命令行、其他程序或文件，传递给你的测试程序的值。

普通的store命令是许多存储命令的基础，可以使用于存储简单的常量值在Selenium变量中。它带有两个参数，被存储的文本值以及一个Selenium变量。当选择一个你的变量的名称时，使用标准的、仅使用字母数字字符的变量命名约定。

CommandTargetValue

storepaul@mysite.orguserName

稍后在你的脚本中，你会想要使用这个存储的变量值。去存取这个变量值，将变量装入大括弧（{}）中，并且前置一个美元符号，像下面这个。

CommandTargetValue

verifyText//div/p\${userName}

经常使用变量于存储输入域的输入。

CommandTargetValue

typeid=login\${userName}

Selenium变量可以被使用于第一个和第二个参数，Selenium会先于命令执行的任何其他操作，解释变量。Selenium变量也可以用于定位器表达式中。

每一个verify和assert命令都存在一个对等的store命令。这里是几个经常使用的存储命令。

storeElementPresent

对应于verifyElementPresent。仅仅存储一个布尔值“true”或“false”，依赖于是否UI元素被发现。

storeText

对应于verifyText。使用一个定位器识别特定的页面文本。文本如果发现就存储在变量中。它可以用于从页面中抽取文本。

storeEval

该命令带有一个脚本作为第一个参数。在Selenese中嵌入Javascript在下一节讨论。这个命令允许测试存储运行脚本的结果在一个变量中。

Javascript和Selenese参数

Javascript可以应用于两种类型的Selenese参数：脚本和非脚本（通常是表达式）。在大多数的情况下，你会想要，在被使用作一个Selenese参数的Javascript代码段中，存取和/或操纵一个测试案例的变量。所有的在你的测试案例中创建的变量，都存储在一个Javascript关联数组中。关联数组具有字符串索引，而非顺序的数字索引。包含你的测试案例的关联数组被命名为storedVars。当你希望在Javascript代码段中存取或操作变量的时候，你必须引用它做storedVars['你的变量名']。

Javascript的脚本参数用法

有几个Selenese命令可以指定脚本参数，包括assertEval，verifyEval，storeEval，和等待waitForEval。这些参数无需特殊的语法。Selenium IDE的用户只需简单地将一个Javascript的代码段放入到适当的域中，通常是目标（Target）域（因为一个脚本参数通常是第一个或唯一的一个参数）。

下面的示例演示一个Javascript代码段是如何应用于执行一个简单的数值计算：

Command Target Value

```
store 10 hits
storeXPathCount //blockquote blockquotes
storeEval storedVars[ 'hits' ]-storedVars[ 'blockquotes' ] paragraphs
```

下面的示例演示一个Javascript代码段可以包含对函数的调用，在这个示例中，是对String对象的toUpperCase和toLowerCase方法的调用。

Command Target Value

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

Javascript的非脚本参数用法

Javascript也可以用于生成参数的值，甚至参数没有被指定为具有脚本类型。然而，在这种情况下，需要使用特殊的语法 - Javascript代码片断必须被包围在封闭的大括弧（{}）中，并且带有javascript的标签前缀，如javascript {这儿是你的代码}。下面是一个示例，type命令的第二个参数置是由Javascript代码使用特殊的语法生成的。

Command	Target	Value
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，允许你打印文本到你的测试输出。这是有用的，对于提供你信息性进度说明，当你的测试正在运行时，显示在控制台。这些说明也可以为你的测试结果报告提供上下文，可以被使用于你的测试发现一个问题时，查找缺陷存在在页面的哪里。最后，echo语句可以使用于打印Selenium变量的内容。

Command	Target	Value
echo	Testing page footer now.	
echo	q	Username is \${userName}

警告，弹出和多窗口

假定你正在测试一个页面，看起来像这样。

```
1<!DOCTYPE HTML>
2<html>
3<head>
4 <script type="text/javascript">
5 function output(resultText){
6 document.getElementById('output').childNodes[0].nodeValue=resultText;
7 }
8
9 function show_confirm(){
10 var confirmation=confirm("Chose an option.");
11 if (confirmation==true){
12 output("Confirmed.");
13 }
14 else{
15 output("Rejected!");
16 }
17 }
18
19 function show_alert(){
20 alert("I'm blocking!");
21 output("Alert is gone.");
22 }
23 function show_prompt(){
```

```
24 var response = prompt("What's the best web QA tool?", "Selenium");
25 output(response);
26 }
27 function open_window(windowName){
28 window.open("newWindow.html", windowName);
29 }
30 </script>
31</head>
32<body>
33
34 <input type="button" id="btnConfirm" onclick="show_confirm()" value="Show confirm box" />
35 <input type="button" id="btnAlert" onclick="show_alert()" value="Show alert" />
36 <input type="button" id="btnPrompt" onclick="show_prompt()" value="Show prompt" />
37 <a href="newWindow.html" id="lnkNewWindow" target="_blank">New Window Link</a>
38 <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alert/confirm对话框，以及移动焦点到新打开的弹出窗口。幸运地是，Selenium可以替代javascript弹出窗口。

但在我们开始分别讨论alert/confirm/prompt的细节前，理解它们之间的共性是有益的。警告，确认和提示对话框都有下面的变种

Command Description

assertFoo(pattern) 抛出错误，如果模式不匹配弹出的文本

assertFooPresent 抛出错误如果弹出是不可得到的

assertFooNotPresent 抛出错误如果任何弹出呈现

storeFoo(variable) 存储弹出的文本到变量

storeFooPresent(variable) 存储弹出文本在变量，然后返回true或false

当在Selenium运行时，Javascript弹出窗口不会出现。这是因为函数调用在运行时实际上被Selenium自己的Javascript重写了。不管怎样，不能因为你不能看到弹出窗口，就意味着你不必处理它。要处理一个弹出窗口，你必须调用assertFoo(pattern)函数。如果你在断言一个弹出窗口的出现时失败，你的下一个命令将会被阻塞，你会得到一个错误，类似于这下面[error] Error: There was an unexpected Confirmation! [Chose an option]

警告

让我们从警告开始，因为这是最简单的需要处理的弹出。要开始，首先在浏览器中打开上面的那个样品页面，然后点击“Show alert”按钮。你会注意到在你关闭了这个alert对话框后，文本“Alert is gone.”被显示在页面上。现在运行这相同的步骤，并用Selenium IDE录制下来，然后验证文本是否在你关闭警告后被增加了。你的测试看起来像这个：

Command Target Value

open /

click btnAlert

assertAlert I' m blocking!

verifyTextPresent Alert is gone.

你可能觉得“那时奇怪的，我从没有断言那个警告。”。但这是Selenium IDE处理的和为你关闭了这个警告。如果你移出那个步骤并回放这个测试，你会得到下面的错误：[error] Error: There was an unexpected Alert! [I'm blocking!]. 你必须包含一个警告的断言去认可它的出现。

如果你只是想要去断言一个出现的警告，但既不知道也不在乎它包含啥文本，你可以使用assertAlertPresent。这将返回一个true或false，在为false时，将停止这个测试。

确认

确认的行为和警告的完全一致，使用assertConfirmation和assertConfirmationPresent提供如同警告的对应操作相同的特征。然而，默认地当一个确认弹出时，Selenium将选择OK。试着录制点击在相同页面的“Show confirm box”按钮，但在弹出得确认对话框中点击“Cancel”按钮，然后断言这输出的文本，你的测试看起来像这个：

Command Target Value

open /

click btnConfirm

chooseCancelOnNextConfirmation

assertConfirmation Choose an option.

verifyTextPresent Rejected

chooseCancelOnNextConfirmation 函数告诉Selenium，所有后面的确认对话框应该返回false。这可以被重置，通过调用chooseOkOnNextConfirmation。

你可能注意到，你不能回放这个测试，因为Selenium抱怨有一个没有处理的确认。Selenium IDE录制事件的顺序，引起click和chooseCancelOnNextConfirmation的顺序错误（如果你仔细考虑会认为这是合理的，在你打开确认对话框前，Selenium不知道你会取消）。只需切换两个命令的顺序，你的测试将运行良好。

© Copyright 2008-2012, Selenium Project. Last updated on Feb 02, 2012.

1.8 Selenium用户指南 - 第三章 Selenium IDE[6]

发表时间: 2012-02-17

调试

调试意味着在测试案例中查找和修复错误。这是测试案例开发的常规组成部分。

We won't teach debugging here as most new users to Selenium will already have some basic experience with debugging. If this is new to you, we recommend you ask one of the developers in your organization.

我们在这里不会讲授调试，因为大多数Selenium的新用户应该已经有调试方面的基本经验。如果对你来说是全新的，我们推荐向你组织的其他开发人员求教。

断点和起点

Selenium IDE支持断点的设置，和从测试案例内部的任何点启动和停止运行中的测试案例的能力。也就是说，可以运行到测试案例中间的指定命令，在该点检查测试案例的行为。要做到这点，在需要检查的命令的前一个命令上设置一个断点。

要设置一个断点，选择一个命令，右点，然后从上下文菜单选择切换断点（Toggle Breakpoint）。然后点击运行（Run）按钮运行测试案例，从开始到断点。

有时候运行一个测试案例，从中间的某个地方到测试案例的结束，或开始点后面的某个断点是有用的。例如，假定你的测试案例首先登录到Web站点，然后执行一系列的测试，而你正在尝试调试这些测试中的一个。不管怎样，你仅仅需要登录一次，但你需要不断地重复运行你的测试，当你正在开发它们。你可以登录一次，然后运行你的测试案例，从你的测试登录部分后面的一个开始点。那将防止你不得不每次重新运行你的测试时需要手动地注销。

要设置一个起点，选择一个命令，右点，然后从上下文菜单选择设置/清除启动点（Set/Clear Start Point）。然后点击运行（Run）按钮执行测试从起点开始。

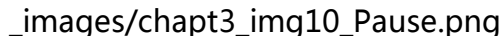
单步通过一个测试案例

要执行一次执行一个命令地执行（单步“Step through”）测试，遵循下面的步骤：

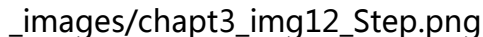
从工具栏，用运行（Run）按钮启动测试案例。

_images/chapt3_img09_Run.png

用暂停（Pause）按钮立即暂停执行测试案例。



重复地选择单步执行（Step）按钮。



查找（Find）按钮

查找（Find）按钮用于看一看，在浏览器中当前显示的页面上的那个UI元素，被使用在目前选择的Selenium命令。这是有用的，当为一个命令的第一个参数构造定位器的时候（参见Selenium命令一章的定位器节）。它可以与任何一个标识Web页面上的一个UI元素的命令一起使用，例如，click，clickAndWait，type，以及某个assert和verify命令，包括其他的。

从表（Table）视图，选择任何带有定位器参数的命令。点击查找（Find）按钮。现在观看Web页面：应该有一个亮绿色的矩形包围着由定位器参数指定的元素。

页面源代码调试

时常，当调试一个测试案例时，你只需要看看页面的源代码（你正在试图测试的Web页面的HTML）去确定问题。Firefox让这个变得很容易。只需要右点Web页面，选择“View-）Page Source”。HTML将打开在一个分离的窗口。使用它的搜索功能（Edit-）Find）进行关键字搜索，找到你正试图测试的UI元素的HTML。

替代地，仅仅选择你希望察看源代码的部分Web页面。然后右点Web页面，选择查看选择源代码（View Selection Source）。在这种情况下，分离的HTML窗口将只包含少量的源代码，你选择的部分会高亮显示。

定位器助手

无论何时Selenium IDE录制一个定位器类型的参数，它将存储附加的信息，那个允许用户去查看其它可能的定位器类型的参数，可用于替代的。这个特征是非常有用的，对于学习更多的有关定位器的知识，也是时常需要的，可以帮助人们去构造一个不同于录制的类型的定位器类型。

定位器助手被提供在Selenium IDE窗口上，作为一个在目标（Target）域的右手端可访问的下拉列表（只有当目标域包含一个录制的定位器类型参数时才显示）。下面是一个快照，显示一个命令的下拉列表内容。注意下拉列表的第一列提供候选的定位器，第二列指示每一个候选者的类型。



编写一个测试集

一个测试集是一个显示在IDE最左边的窗格的测试案例的集合。测试集窗格可以手动地打开或关闭，通过选择这个窗格的右边的半腰的小点。

测试集窗格会被自动打开，当一个存在的测试集被打开，或者当用户从文件（File）菜单选择新建测试案例（New Test Case）。在这种情况下，新的测试案例将立即出现在前一个测试案例的下面。

Selenium IDE仍然不支持装载一个已经存在的测试案例进入一个测试集。希望通过增加一个已经存在的测试案例来创建或修改一个测试集的用户，必须手工编辑一个测试集文件。

一个测试集文件是一个HTML文件，包含一个单列的表。在<tbody>节的每行的每个单元格，包含一个到测试案例的链接。下面的一个示例是一个包含四个测试案例的测试集。

```
<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>
```

注释

测试案例文件应该不必与调用它们的测试集文件在一个地方。在Mac OS和Linux系统，正是如此。然而在编写本文档时，一个bug会阻止Windows用户将测试案例文件放置在调用它们的测试集放置的其他地方。

用户扩展

用户扩展是允许一个人创建他或她自己的自定义或特征去增加附加功能的Javascript文件。时常以自定义命令的形式，尽管扩展不限于附加自定义命令。

有大量的用户创建的有用的[扩展](#)。

重要：此节已经过时 - 我们将很快修订

或许所有Selenium IDE扩展中最流行的扩展是，以while循环和基本条件语句的形式提供流控制的扩展。该扩展是[goto_sel_ide.js](#)。有关如何使用该扩展提供的功能的示例，参见扩展的作者创建的页面。

要安装这个扩展，可在Selenium IDE的选项菜单中，打开选项对话框，在“General”选项页的Selenium Core extensions域添加扩展文件在本地计算机的路径名称。

[_images/chapt3_img32_Extensions_install.png](#)

在选择OK按钮后，你必须关闭和重新打开Selenium IDE，以便IDE读取扩展文件。任何对扩展做出的改变也需要你关闭和重新打开Selenium IDE。

有关编写你自己的扩展的信息，可以在Selenium参考文档的底部找到。

格式化

格式化（Format）在选项（Options）菜单下，允许你选择一种语言用于保存和显示测试案例。默认是HTML。

如果你将使用Selenium RC运行测试案例，格式化功能用于转换你的测试案例到一种编程语言。选择一种你用来开发你的测试程序的语言，诸如，Java，PHP。然后简单地使用导出测试案例（File => Export Test Case As）存储测试案例。你的测试案例会被转换为你选择的语言的一系列函数。基本上，Selenium IDE会为你生成支持你的测试的程序代码。

另外，要注意如果生成的代码不适合你的需要，你可以通过编辑一个定义生成过程的配置文件修改它。每一种支持的语言都有可编辑的配置。这个在选项对话框的Format标签下（Options => Options...）。

注释

在编写本文当时，Selenium的开发者还没有提供对这个功能的支持。不管怎样作者已经可以以受限的方式修改C#格式化，并且可以很好地工作。

在不同的浏览器执行Selenium IDE测试

尽管Selenium IDE仅可在Firefox运行测试，然而用Selenium IDE开发的测试可以运行在其他的浏览器中，通过使用一个简单的命令行接口调用Selenium RC服务器。该主题在Selenium RC一章的运行Selenese测试节讨论。

故障排除

下面是一列描述使用Selenium IDE产生问题的常见来源的图像/说明：

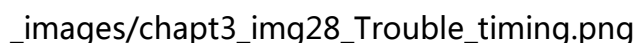
使用本格式，表（Table）视图不可得到

这个消息可能偶尔会显示在表（Table）选项页，当Selenium IDE启动时。变通方法是关闭和重新打开Selenium IDE。要获取更多的信息参见[issue 1008](#)。如果你不能可靠地重现它，请提供详细信息以便我们可以修复。

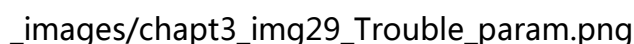
错误装载测试案例：命令没有发现

你使用File => Open试图打开一个测试集文件，使用File => Open Test Suit代替。

一个增强的要求已经发出，以便改进这个错误消息。参见[issue 1010](#)。

_images/chapt3_img28_Trouble_timing.png

这种类型的错误可能表示一个与时间相关的问题，例如在你的命令中由定位器指定的元素没有被完全装载，当命令执行的时候。试着在命令前放置一个pause 5000，以确定问题是否真的与时间相关。如果是这样，考虑使用一个适当的waitFor*或*AndWait命令在这个失败的命令前。

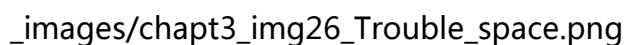
_images/chapt3_img29_Trouble_param.png

当你试图使用一个变量替代时失败，如上面的open命令的情形，这表示你还没有实际地创建这个你试图存取值的变量。这有时候是由于放置这个变量在值（Value）域，当它应该在目标（Target）域时出现，反之亦然。在上面的示例中，store命令的两个参数错误地以相反地顺序放置。对任何Selenese命令，第一个要求的参数必须在目标（Target）域，第二个要求的参数（如果存在）必须在值（Value）域。

```
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]
```

在你的测试集中的一个测试案例不能被找到。确信这个测试案例位于测试集指示的位置。另外，确信你实际的测试案例文件具有.html扩展名，包括在文件名称中，以及引用它们的测试集文件中。

一个增强的要求已经发出，以便改进这个错误消息。参见[issue 1011](#)。

_images/chapt3_img26_Trouble_space.png

Selenium IDE是空格敏感的！在命令前后的一个额外空格会引起命令不可识别。

该缺陷已经被提升。参见[issue 1012](#)。

_images/chapt3_img27_Trouble_extension.png

你的扩展文件的内容还没有被Selenium IDE读取。确信你指定了适当的到扩展文件的路径名，在选项对话框的Selenium Core extensions域。另外，Selenium IDE必须重新启动，在对扩展文件或Selenium Core extensions域的内容作出任何改变后。

_images/chapt3_img30_Trouble_collapsed.png

这种类型的错误看起来好像是Selenium IDE在不该产生错误的地方生成了一个失败。实际上，Selenium IDE是正确的，因为在这个测试案例中实际的值不匹配指定的值。问题是日志文件错误消息会压缩一系列的两个或多个空格成一个空格，这是容易令人混淆的。在这个示例中，verifyTitle的参数在单词 “Selenium” 和 “web” 之间有两个空格，而页面的实际标题只有一个空格。因此，Selenium IDE正确地产生了一个错误，但是一个误导性质的错误。

该缺陷已经被提升。参见[issue 1013](#)

© Copyright 2008-2012, Selenium Project. Last updated on Feb 02, 2012.

1.9 Selenium用户指南 - 第四章 Selenium 2.0和WebDriver[1]

发表时间: 2012-02-17

Selenium 2.0 和 WebDriver

注释：我们目前正工作在文档化这些章节。我们相信此处的信息是准确的，然而，要知道我们仍然在这一章上工作着。会提供附加的信息，我们会让这一章变得更加可靠。

Selenium 2.0 特征

Selenium 2.0 有许多令人兴奋的新特征以及对Selenium 1 的改进。这些引入的新特征发布在官方的Selenium Blog的发布声明中。

最主要的新特征是与WebDriver API的集成。该功能标定了大量的限制，并提供了一个可选的，和更简单的编程接口。目标是开发一个面向对象的，为更多数量的浏览器提供附加的支持的，以及对现代的高级Web应用程序测试问题提供改善支持的API。

注释：我们将增加一个Selenium 2.0 新特征描述 - 暂时我们推荐读者阅读发布声明。

Selenium服务器 - 何时使用它

你可能，或可能不，需要Selenium服务器，这取决于你打算如何使用Selenium。如果你打算完全使用WebDriver API，你不一定需要Selenium服务器。Selenium-WebDriver使用每个浏览器对自动化的本地支持直接对浏览器发出调用。如何作出这些调用，依赖于你正在使用的浏览器。

为何你需要Selenium服务器的几个理由是 -

你正在使用Selenium栅格，在许多机器/虚拟机上分布你的测试

你希望连接到一个带有特定版本的浏览器的远程机器，在你目前的机器上没有哪个版本。

你不使用Java绑定，而更喜欢使用HtmlUnit驱动器。

建立一个Selenium-WebDriver项目

安装Selenium用于建立一个开发项目，以便你可以使用Selenium编写程序。如何做依赖于你的编程语言和你的开发环境。

Java

使用Maven是最容易的建立一个Selenium 2.0 Java项目的方式。使用一个Maven pom.xml(项目配置)文件，Maven会下载Java绑定（Selenium 2.0 Java客户端库）和所有它的依赖项，并将为你创建这个项目。一旦你完成了这个步骤，你可以导入Maven项目到你偏爱的IDE，IntelliJ IDEA或Eclipse。

首先，创建一个文件夹去包含你的所有Selenium项目文件。然后你需要一个pom.xml文件，为了使用Maven。这可以用一个文本编辑器创建。我们不会讲授pom.xml文件的细节，或者如何使用Maven，因为已经有与之相关的卓越的参考资料。你的pom.xml文件看起来像这样。创建这个文件在你为项目创建的文件夹中。

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/
maven-4.0.0.xsd">
<modelVersion>4.0.0</modelVersion>
<groupId>MySel20Proj</groupId>
<artifactId>MySel20Proj</artifactId>
<version>1.0</version>
<dependencies>
<dependency>
<groupId>org.seleniumhq.selenium</groupId>
<artifactId>selenium-java</artifactId>
<version>2.16.1</version>
</dependency>
</dependencies>
</project>
```

确信你指定了最新的版本。在编写本文档时，上面列出的版本是最新的，然而在Selenium 2.0这个发布后，有频繁的发布。检查当前发布的[Maven下载页](#)，相应地编辑上面的依赖项。

现在，从一个命令行，CD到项目目录并运行Maven如下：

```
mvn clean install
```

这会下载Selenium和所有它的依赖项，并增加它们到项目中。

最后，导入这个项目到你偏爱的开发环境中。对不熟悉的用户，我们有提供一个显示这个过程的附录。

[导入一个Maven项目到IntelliJ IDEA。](#) [导入一个Maven项目到Eclipse。](#)

C#

自Selenium 2.2.0起，C#绑定被发布做带有其他依赖动态链接库的一套签名的动态链接库（dll）。在2.2.0前，所有的Selenium动态链接库是不签名的。要包含Selenium在你的项目中，只需从<https://code.google.com/p/selenium/downloads/list>下载最新的Selenium .NET的zip文件。如果你使用Windows Vista或以上的版本，你应先于解压缩而解决这个zip文件：右点zip文件，点击“属性（Properties）”，点击“解决（Unblock）”然后点击“确定（OK）”。

解压缩zip文件的内容，在Visual Studio（或你选择的IDE）中增加对每个解压缩的文件的引用到你的项目中

官方NuGet包：`RC WebDriver`
`WebDriverBackedSelenium Support`

Python

如果你正在使用Python进行测试自动化，那么你或许已经熟悉Python的开发。从命令行运行下面的命令去增加Selenium到你的Python环境。

使用pip安装Selenium

需要安装pip，同样pip有一个对setuptools的依赖。

讲授Python开发本身已经超出了本文档的范畴，不管怎样，有许多Python相关的资源，你组织中的开发者可能可以帮助你加快速度。

Ruby

如果你正在使用Ruby进行测试自动化，那么你或许已经熟悉Ruby的开发。从命令行运行下面的命令去增加Selenium到你的Ruby环境。

```
gem install selenium-webdriver
```

讲授Ruby开发本身已经超出了本文档的范畴，不管怎样，有许多Ruby相关的资源，你组织中的开发者可能可以帮助你加快速度。

Perl

Perl绑定由第三方提供，请参考如何安装/起步的任何第三方的文档。在编写本文档时，有一个已知的Perl绑定。

PHP

PHP绑定由第三方提供，请参考如何安装/起步的任何第三方的文档。在编写本文档时，有三个已知的绑定：By Chibimagic By Lukasz Kolczynski and By the Facebook.

从Selenium 1.0迁移

对那些已经有使用Selenium 1.1编写的测试集的用户，我们提供了一些如何从现存的代码迁移到Selenium 2.0的技巧。Simon Stewart，领先的Selenium 2.0的开发者，写过一篇有关从Selenium 1.0迁移的文章。我们已经包含在附录中。

迁移从Selenium RC到Selenium WebDriver

Selenium-WebDriver起步

WebDriver是一个自动化Web应用程序测试的工具，尤其是验证如预期工作。WebDriver的目标是提供一个友好的API，易于开发和理解，比Selenium RC (1.0) API更容易使用，这将帮助你让测试变得更容易理解和维护。它不与任何特定的测试框架关联，以便可以同样好地使用在单元测试或普通的老的“main”方法中。这一节介绍WebDriver API，并帮助你开始熟悉它。从建立一个WebDriver项目开始，如果你还没有。这个已经在前一节，建立一个Selenium-WebDriver项目，中描述。

一旦你的项目建立，你可以看到WebDriver如同任何常规的类库：它是自包含的，通常不需要记住要开始任何附加的处理，或使用前运行任何安装器，不同于带有Selenium RC的代理服务器。

注释：要使用 Chrome，Opera，Android和iPhone驱动器需要附加的步骤。

你现在已经准备好可以编写代码。一个容易的方法是从这个示例开始，在Google搜索术语“Cheese”，然后输出结果页面的标题到控制台。

```
package org.openqa.selenium.example;

import org.openqa.selenium.By;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.WebElement;
import org.openqa.selenium.firefox.FirefoxDriver;
import org.openqa.selenium.support.ui.ExpectedCondition;
import org.openqa.selenium.support.ui.WebDriverWait;

public class Selenium2Example {
    public static void main(String[] args) {
        // 创建一个Firefox驱动器的新实例
        // 注意后面的代码依赖于借口而不是实现
        WebDriver driver = new FirefoxDriver();

        // 访问Google
```

```
driver.get("http://www.google.com");
// 可选的，可以像这样完成相同的事情
// driver.navigate().to("http://www.google.com");

// 按名称找到输入元素
WebElement element = driver.findElement(By.name("q"));

// 键入要搜索的文本
element.sendKeys("Cheese!");

// 提交窗体。WebDriver会找到元素所属的窗体
element.submit();

// 检查页面标题
System.out.println("Page title is: " + driver.getTitle());

// Google搜索用Javascript动态绘制
// 等待页面被装载，10秒后超时
(new WebDriverWait(driver, 10)).until(new ExpectedCondition<Boolean>() {
    public Boolean apply(WebDriver d) {
        return d.getTitle().toLowerCase().startsWith("cheese!");
    }
});

// 应看到："cheese! - Google Search"
System.out.println("Page title is: " + driver.getTitle());

// 关闭浏览器
driver.quit();
}
}
```

在接下来的部分，我们要学习更多的关于如何使用WebDriver，诸如在浏览器的历史中前后导航，如何测试使用frame和window的Web站点。我们也会提供更多的全面的讨论和示例。

[1.10 Selenium用户指南 - 第四章 Selenium 2.0和WebDriver\[2\]](#)

发表时间: 2012-02-17

WebDriver驱动器入门

WebDriver是编写测试使用的关键的接口名称，有几个实现。包括：

HtmlUnit驱动器

这个是目前最快和最轻量级的WebDriver实现。正如名称所指示的，这是基于HtmlUnit的。HtmlUnit是一个基于Java的、无GUI的Webbrowser实现。对任何语言绑定（除了Java），要求Selenium服务器使用这个驱动器。

用法

```
WebDriver driver = new HtmlUnitDriver();
```

优点

- WebDriver的最快实现
- 纯Java解决方案，因此是平台无关的
- 支持Javascript

缺点

- 模拟其他浏览器的Javascript行为（参加下面）

在HtmlUnit驱动器中的Javascript

流行的浏览器中没有使用，由HtmlUnit（Rhino）使用的Javascript引擎。如果你使用HtmlUnit测试Javascript，结果可能和其他浏览器产生的结果差别很大。

当我们说“Javascript”时，我们实际上意味着“Javascript和DOM”。虽然DOM是由W3C定义的，但每个浏览器在DOM的实现和Javascript如何与之交互上，有它自己的怪异和差别。HtmlUnit有一个令人难忘的DOM的完整实现，以及对使用Javascript的良好支持。但和其他的浏览器没有差别的是：它也有自己的怪异和差别，与W3C的标准以及其他主要浏览器的DOM实现，而不管它模拟其他浏览器的能力。

使用WebDriver，我们必须作出一个选择；是使能HtmlUnit的Javascript能力，而冒着团队遇到问题的风险，还是保持Javascript无效，尽管知道有越来越多的浏览器依赖于Javascript？我们采用保守的方法，当我们使用HtmlUnit时，默认不支持。随WebDriver和HtmlUnit的每个发布，我们重新评估这个决定：在某个时刻，我们希望使能Javascript在HtmlUnit。

使能Javascript

如果你不能等待，使能Javascript支持是很容易的。

```
HtmlUnitDriver driver = new HtmlUnitDriver(true);
```

这将引起HtmlUnit驱动器默认地模拟Firefox 3.6的Javascript处理。

Firefox驱动器

使用一个Firefox插件控制Firefox浏览器。从在机器上安装中剥离使用的Firefox Profile，仅包含Selenium WebDriver.xpi(插件)。几个设置也默认地被改变（参见源代码看看做了哪些改变）。Firefox驱动器有能力运行在Windows，Mac，Linux，并已经测试在版本3.0，3.6，5，6，7和8上测试。

用法

```
WebDriver driver = new FirefoxDriver();
```

优点

- 运行在一个真正的浏览器并支持Javascript
- 比Internet Explorer驱动器更快

缺点

- 比HtmlUnit驱动器慢

修改Firefox Profile

假定你希望修改用户代理串（如上所示），你必须修改包含成打的有用的扩展的Firefox Profile。有两种方法可以得到这个Profile。假定那个Profile已经使用Firefox的Profile管理员创建（firefox -ProfileManager）。

```
ProfileIni allProfiles = new ProfilesIni();  
FirefoxProfile profile = allProfiles.getProfile("WebDriver");  
profile.setPreferences("foo.bar", 23);  
WebDriver driver = new FirefoxDriver(profile);
```

替代地，如果Profile还没有在Firefox注册：

```
File profileDir = new File("path/to/top/level/of/profile");
FirefoxProfile profile = new FirefoxProfile(profileDir);
profile.addAdditionalPreferences(extraPrefs);
WebDriver driver = new FirefoxDriver(profile);
```

当我们开发Firefox驱动器的功能的时候，我们暴露了使用它们的能力。例如，直到感觉Linux上的Firefox的本地事件已经稳定，默认无效它们。要使能它们：

```
FirefoxProfile profile = new FirefoxProfile();
profile.setEnableNativeEvents(true);
WebDriver driver = new FirefoxDriver(profile);
```

信息

参见wiki页的Firefox部分，以获得最新的信息。

Internet Explorer驱动器

这个驱动器使用一个动态链接库控制，因此只有在Windows OS可以使用。每一个Selenium的发布，它的核心功能已经在版本6，7，8在XP，以及版本9在Windows 7上经过测试

用法

```
WebDriver driver = new InternetExplorerDriver();
```

优点

- 运行在一个真实的浏览器，支持Javascript连同你的终端用户可见的所有怪异模式。

缺点

- 显然Internet Explorer驱动器仅仅工作在Windows操作系统!
- 比较而言速度慢（尽管仍然相当快）
- 在大多数版本XPath没有本地的支持。自动注入Sizzle比其他浏览器满很多，而且在同一个浏览器进行CSS选择器的比较时较慢。
- 在版本6和7CSS不被本地支持。替代注入Sizzle。
- 在IE8和9，CSS选择器是本地的，但这些浏览器不完全支持CSS3。

信息

参见wiki页的Internet Explorer部分，以获得最新的信息。请特别注意要求的配置（ Required Configuration ）部分。

Chrome驱动器

Chrome驱动器由Chromium项目自身维护和支持。WebDriver与Chrome一起工作，通过chromedriver二进制（可以在Chromium项目的下载页找到）。你需要有chromedriver和安装的chrome浏览器的一个版本。chromedriver需要防止在你的系统路径的某个地方，以便WebDriver可以自动地发现。Chrome浏览器本身由chromedriver在缺省的安装路径发现。两者可以由环境变量重写。请参考wiki以获得更多信息。

用法

```
WebDriver driver = new ChromeDriver();
```

优点

- 运行在真实的浏览器，并支持Javascript
- 因为Chrome是基于Webkit的浏览器，Chrome驱动器可能允许你验证工作在Safari的你的站点。注意因为Chrome使用它自己的V8 Javascript引擎，而不是Safari的Nitro引擎，Javascript的执行可能有差别。

缺点

- 比HtmlUnit驱动器慢

信息

参见我们的wiki为了最新的信息。更多的信息可以被发现在下载页。

开始运行使用Chrome驱动器。

下载Chrome驱动器可执行文件，然后遵循在wiki页的其他指令。

Opera驱动器

参见Opera驱动器的wiki文章，在Selenium wiki以获得关于使用Opera驱动器的信息。

iPhone驱动器

参见iPhone驱动器的wiki文章，在Selenium wiki以获得关于使用Mac iOS驱动器的信息。

Android驱动器

参见Android驱动器的wiki文章，在Selenium wiki以获得关于使用Android驱动器的信息。

© Copyright 2008-2012, Selenium Project. Last updated on Feb 02, 2012.

[1.11 Selenium用户指南 - 第四章 Selenium 2.0和WebDriver\[3\]](#)

发表时间: 2012-02-18

命令和操作

存取一个页面

可能你想使用WebDriver做的第一件事情是导航到一个页面。要做到这点的普通方法是通过调用 “get” 。

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

取决于几个因素，包括OS/浏览器的组合，WebDriver可能或可能不会等待页面被装载。在某些情况下，WebDriver可能返回控制，先于页面完成，或甚至启动和装载。为了确保健壮性，你需要使用显式或隐含地等待，以等待元素存在页面上。

定位UI元素（Web元素）

用WebDriver定位元素可以使用WebDriver实例本身，或者在一个WebElement上。每一种语言绑定都暴露了一个“查找单个元素”和“查找多个元素”方法。第一个返回一个WebElement对象，否则将抛出一个异常。后一个返回一个WebElement列表，它可能返回一个空列表，如果没有DOM元素匹配这个查询。

“查找”方法带有一个定位器或称之为“By”的查询对象。“By”策略列表如下：

按id查找

这是定位一个元素最有效和首选的方法。UI开发者制造的常见陷阱是在页面上没有唯一的id或自动生成的id，两者都应该避免。在一个html元素上的class属性，比自动生成的id属性更适当。

下面是如何找到一个看起来像这样的元素的示例：

```
<div id="coolestWidgetEvah">...</div>
```

```
WebElement element = driver.findElement(By.id("coolestWidgetEvah"));
```

按类（class）名称查找

此处的“Class”意指DOM元素的class属性。时常在实际使用中，有许多DOM元素带有相同的class，因此查找多个元素比查找第一个元素成为更实际的选项。

如何查找一个看起来像这样的元素的示例：

```
<div class="cheese"> <span>Cheddar</span> </div> <div  
class="cheese"> <span>Gouda</span> </div>  
List<WebElement> cheeses = driver.findElements(By.className("cheese"));
```

按标记 (tag) 名称查找

元素的DOM标记名称

如何查找一个看起来像这样的元素的示例：

```
<iframe src="..."> </iframe>  
WebElement frame = driver.findElement(By.tagName("iframe"));
```

按名称 (name) 查找

用匹配的name属性找到输入元素

如何查找一个看起来像这样的元素的示例：

```
<input name="cheese" type="text"/>  
WebElement cheese = driver.findElement(By.name("cheese"));
```

按链接文本查找

用可见的文本查找超链接元素

如何查找一个看起来像这样的元素的示例：

```
<a href="http://www.google.com/search?q=cheese">cheese</a>  
WebElement cheese = driver.findElement(By.linkText("cheese"));
```

按部分链接文本查找

用部分可见的文本查找超链接元素

如何查找一个看起来像这样的元素的示例：

```
<a href="http://www.google.com/search?q=cheese">search for cheese</a>  
WebElement cheese = driver.findElement(By.partialLinkText("cheese"));
```

按css样式查找

如同名称所隐含的，这是一个按css的定位器策略。缺省地使用本地浏览器支持，请参考W3C的css选择器，以获得一系列常用可得到的css选择器列表。如果浏览器没有对css选择器本地的支持，则使用Sizzle。IE6，7和Firefox 3.0目前使用Sizzle作为css查询引擎。

小心，不是所有浏览器都是一样的，某些css可能工作在一个版本，但在另一个版本不工作。

下面是查找cheese的示例

```
<div id="food"><span class="dairy">milk</span><span class="dairy aged">cheese</span></div>
WebElement cheese = driver.findElement(By.cssSelector("#food span.dairy.aged"));
```

按XPath查找

在高级别，只要可能，WebDriver使用浏览器内置的XPath能力。在那些没有内置XPath支持的浏览器，我们提供了我们自己的实现。这可能回导致某些未预期得行为，除非你知道各种各样的XPath引擎之间的差异。

Driver	Tag and Attribute Name	Attribute Values	Native XPath Support
HtmlUnit Driver	Lower-cased	As they appear in the HTML	Yes
Internet Explorer Driver	Lower-cased	As they appear in the HTML	No
Firefox Driver	Case insensitive	As they appear in the HTML	Yes

这有一点抽象，看看下面的HTML片段：

```
<input type="text" name="example" />
<INPUT type="text" name="other" />
List<WebElement> inputs = driver.findElements(By.xpath("//input"));
```

下面是找到的匹配的数量

XPath expression	HtmlUnit Driver	Firefox Driver	Internet Explorer Driver
//input	1 ("example")	2	2
//INPUT	0	2	0

有时HTML元素的属性不需要显式的声明，它们有缺省的已知值。例如，“input” 标记不需要“type” 属性，因为缺省为“text”。经验法则是使用WebDriver的XPath时，不应预期能够匹配这些隐含的属性。

使用Javascript

你可以执行任何Javascript，以找到一个元素，只要你找到一个DOM元素，它将自动转换为WebElement对象。

简单的示例在一个装载了jQuery的页面上：

```
WebElement element = (WebElement) ((JavascriptExecutor)driver).executeScript("return  
$('.cheese')[0]");
```

查找页面上的每个标签所指的input元素：

```
List<WebElement> labels = driver.findElements(By.tagName("label"));  
List<WebElement> inputs = (List<WebElement>) ((JavascriptExecutor)driver).executeScript(  
"var labels = arguments[0], inputs = []; for (var i=0; i < labels.length; i++){" +  
"inputs.push(document.getElementById(labels[i].getAttribute('for'))); } return inputs;", labels);
```

用户输入 - 填充窗体

我们已经看到如何键入文本到一个textarea或text域，但其他的元素哪？你可以“切换”复选框的状态，你可以使用“click”去设置类似于选项（OPTION）标记的选择。处理SELECT标记也不是太糟糕：

```
WebElement select = driver.findElement(By.tagName("select"));  
List<WebElement> allOptions = select.findElements(By.tagName("option"));  
for (WebElement option : allOptions) {  
System.out.println(String.format("Value is: %s", option.getAttribute("value")));  
option.click();  
}
```

这将找到页面上的第一个“SELECT”元素，并依次循环通过每一个选项（OPTION），打印出它们的值，然后依次选择每一个。正如你注意到的，这不是处理SELECT元素最有效的方式。WebDriver提供许多类，包括一个称之为“Select”的，提供了实用的方法与它们交互。

```
Select select = new Select(driver.findElement(By.tagName("select")));  
select.deselectAll();  
select.selectByVisibleText("Edam");
```

这将取消页面上的第一个SELECT元素选择所有的选项，然后选择带有显示文本“Edam”的选项。

一旦，你完成了窗体的填充，你可能希望提交它。完成这个的一个方法是查找“submit”按钮然后“click”它。

```
driver.findElement(By.id("submit")).click();
```

替代地，WebDriver在每个元素上有替代的方法“submit”。如果你在一个窗体内的元素上调用它，WebDriver会走查DOM直到找到这个包围的窗体，然后在其上调用提交。如果一个元素没有在窗体内，则抛出NoSuchElementException。

```
element.submit();
```

在window和frame之间移动

有些Web应用程序有许多frame和多个window。WebDriver支持使用 “switch” 方法在命名的window之间移动：

```
driver.switchTo().window("windowName");
```

所有对驱动器的调用，现在将被解释作定向到特殊的window。但你如何获得window的名称？可以看看打开它的Javascript或链接：

```
<a href="somewhere.html" target="windowName">Click here to open a new window</a>
```

替代地，你可以传递一个 “window句柄” 给 “switchTo().window()” 方法，知道这个，就可以像这样遍历每一个打开的窗口：

```
for (String handle : driver.getWindowHandles()) {  
    driver.switchTo().window(handle);  
}
```

你也可以从一个frame转到到另一个（或到iframe）：

```
driver.switchTo().frame("frameName");
```

可以访问子frame，通过使用.(dot)分隔路径，你也可以指定frame按它的索引。那就是：

```
driver.switchTo().frame("frameName.0.child");
```

这将会到名称为 “frameName” 的frame的第一个子frame的名称为 “child” 的frame。从top（根窗口）开始评估所有的frame。

弹出对话框

从Selenium 2.0 beta 1 开始，有内置的对弹出对话框的处理支持。在你触发一个动作打开一个弹出对话框后，你可以访问这个对话框用下面的方式：

```
Alert alert = driver.switchTo().alert();
```

这将返回目前打开的alert对象。用这个对象你现在可以接受，关闭，读取它的内容，甚至键入一个提示框的内容。这个接口在alert，confirm和prompt对话框上工作的同样好。参考JavaDoc以获取更多的信息。

导航：历史和位置

在较早，我们涉及了导航到一个页面使用“get”命令（`driver.get("http://www.example.com")`）。正如你看到的，WebDriver有大量的较小的，聚焦于任务的接口，而导航是一个有用的任务。因为装载一个页面是一个如此基本的要求，完成这个任务的方法在主WebDriver接口，但它仅仅是一个下面方法的同义词：

```
driver.navigate().to("http://www.example.com");
```

重申：“`navigate().to()`”和“`get()`”做完全相同的事情。只是一个比另一个更容易的键入。

“`navigate`”接口也暴露在你的浏览器历史中前后移动的能力：

```
driver.navigate().forward();
```

```
driver.navigate().back();
```

请注意这个功能完全依赖于潜在的浏览器。如果你使用这个功能在不同的浏览器，出现一些未预期的事情是完全可能的。

Cookies

在我们移动到下个步骤前，你可能对理解如何使用cookies感兴趣。首先，你必须在cookie有效的域。如果你试图预置cookies，先于你开始与一个站点的交互，而且你的主页很大，需要花费一些时间装载。

// 跳转到正确的域

```
driver.get("http://www.example.com");
```

// 现在设置cookie，这个对整个域是有效的。

```
Cookie cookie = new Cookie("key", "value");
```

```
driver.manage().addCookie(cookie);
```

// 现在输出当前的URL,所有可得到得cookie

```
Set<Cookie> allCookies = driver.manage().getCookies();
```

```
for (Cookie loadedCookie : allCookies) {
```

```
System.out.println(String.format("%s -> %s", loadedCookie.getName(), loadedCookie.getValue()));
```

```
}
```

可以用三种方法删除cookie

// 按名称

```
driver.manage().deleteCookieNamed("CookieName");
```

```
// 按Cookie
```

```
driver.manage().deleteCookie(cookie);
```

```
// 或所有
```

```
driver.manage().deleteAllCookies();
```

改变用户代理

使用Firefox驱动器是容易的：

```
FirefoxProfile profile = new FirefoxProfile();
```

```
profile.addAdditionalPreference("general.useragent.override", "some UA string");
```

```
WebDriver driver = new FirefoxDriver(profile);
```

拖放

这是一个使用Actions类执行拖放操作的示例。需要使能内置事件。

```
WebElement element = driver.findElement(By.name("source"));
```

```
WebElement target = driver.findElement(By.name("target"));
```

```
(new Actions(driver)).dragAndDrop(element, target).perform();
```

© Copyright 2008-2012, Selenium Project. Last updated on Feb 02, 2012.

[1.12 Selenium用户指南 - 第四章 Selenium 2.0和WebDriver\[4\]](#)

发表时间: 2012-02-18

驱动器特性和权衡

支持WebDriver的Selenium-RC

Java版的WebDriver提供了一个Selenium RC API实现。这意味着在使用Selenium RC API时，使用潜在的WebDriver技术。这主要是为了提供向后的兼容性。它允许那些有现存的、使用Selenium RC API的测试集的用户，可以在外表下使用WebDriver。提供用于使迁移到Selenium-WebDriver的道路变得轻松。同样，这允许人们并行地，在一个相同的测试中使用两个API。

Selenium-WebDriver像这样使用：

// 你可以使用任何WebDriver实现。在这里Firefox被用于作为一个示例。

```
WebDriver driver = new FirefoxDriver();
```

// 一个 “base url” ， Selenium用于解析相对URL。

```
String baseUrl = "http://www.google.com";
```

// 创建Selenium实现

```
Selenium selenium = new WebDriverBackedSelenium(driver, baseUrl);
```

// 用Selenium执行动作

```
selenium.open("http://www.google.com");
```

```
selenium.type("name=q", "cheese");
```

```
selenium.click("name=btnG");
```

// 返回潜在的WebDriver实现。这将引用相同的WebDriver实例，如同上面的 “driver” 变量

```
WebDriver driverInstance = ((WebDriverBackedSelenium) selenium).getWrappedDriver();
```

// 最后，关闭浏览器。调用WebDriverBackedSelenium 的stop方法

// 替代调用driver.quit()。否则JVM会继续在浏览器已经关闭后运行

```
selenium.stop();
```

优点

- 允许WebDriver和Selenium API并行地运行。
- 提供一个简单的机制用于管理从Selenium RC API到WebDriver的迁移。
- 不需要独立的Selenium RC服务器被运行。

缺点

- 没有实现每个方法。
- 更多的高级Selenium用法（使用“browserbot”或其他来自Selenium内核的、内置的Javascript方法）可能不工作。
- 某些方法可能比较慢，由于潜在的实现上的差别

支持WebDriver

WebDriver不支持Selenium RC支持的所有浏览器，如此当使用WebDriver API时，为了提供那种支持，你可以充分利用SeleneseCommandExecutor。

下面的代码以这种方式支持Safari（确信取消弹出阻塞）

```
DesiredCapabilities capabilities = new DesiredCapabilities();
capabilities.setBrowserName("safari");
CommandExecutor executor = new SeleneseCommandExecutor(new URL("http://localhost:4444/"),
new URL("http://www.google.com/"), capabilities);
WebDriver driver = new RemoteWebDriver(executor, capabilities);
```

用这种方法目前有某些较多的限制，尤其是findElement不能如预期般工作。同样，因为我们正在使用Selenium内核驱动浏览器，你会受到Javascript沙盒的限制。

Selenium WebDriver wiki

你可以在WebDriver的wiki找到进一步的WebDriver资源。

下一步

这一章只是高级别的WebDriver和某些关键特征的浏览。一旦熟悉Selenium-WebDriver API，你就会想要学习如何去构造可维护和可扩展的测试集，并在AUT的功能频繁地变化时降低脆弱性。大多数的Selenium专家现在推荐的方法是使用页面对象设计模式（Page Object Design Pattern），以及可能的页面工厂（Page Factory）设计你的测试代码。Selenium-WebDriver提供对这个的支持，通过提供一个PageFactory类在Java

和C#中。这个，以及其他的高级主题，被介绍在下一章。此外，为了获得这种技术的高级别描述，你可能希望看看测试设计考虑一章。这些章节介绍了通过使测试代码更模块化来编写更可维护的测试。

© Copyright 2008-2012, Selenium Project. Last updated on Feb 02, 2012.

1.13 Selenium用户指南 - 第五章 WebDriver: 高级用法

发表时间: 2012-02-19

WebDriver : 高级用法

显式和隐含等待

等待是让自动化任务的执行，先于继续到下一个步骤，推移一定数量的时间。

显式等待

显式等待是，先于代码的继续执行，而定义的等待某个条件发生的代码。最糟糕的情况是Thread.sleep()，设置条件为一个需要等待的精确时间段。有一些提供的便利方法，可以帮助你编写代码仅仅等待需要的时间。

WebDriverWait与ExpectedCondition的结合是一种可以完成这个目标的方式。

Java

```
WebDriver driver = new FirefoxDriver();
driver.get("http://somedomain/url_that_delays_loading");
WebElement myDynamicElement = (new WebDriverWait(driver, 10))
.until(new ExpectedCondition<WebElement>(){
@Override
public WebElement apply(WebDriver d) {
return d.findElement(By.id("myDynamicElement"));
}});
```

C#

```
IWebDriver driver = new FirefoxDriver();
driver.Url = "http://somedomain/url_that_delays_loading";
WebDriverWait wait = new WebDriverWait(driver, TimeSpan.FromSeconds(10));
IWebElement myDynamicElement = wait.Until<IWebElement>((d) =>
{
return d.FindElement(By.Id("someDynamicElement"));
});
```

这将等待最多10秒先于抛出一个TimeoutException，或者如果找到这个元素，则将在0 - 10秒内返回。

WebDriverWait默认每隔500毫秒调用ExpectedCondition直到它成功返回。

此示例与第一个隐含等待示例在功能上等价。

隐含等待

一个隐含等待告诉WebDriver去轮询DOM一段时间，当试图查找不是立即可得到的一个或多个元素时。默认的设置是0。一旦设置，隐含等待将具有WebDriver对象实例的生存期。

Java

```
WebDriver driver = new FirefoxDriver();
driver.manage().timeouts().implicitlyWait(10, TimeUnit.SECONDS);
driver.get("http://somedomain/url_that_delays_loading");
WebElement myDynamicElement = driver.findElement(By.id("myDynamicElement"));
```

C#

```
WebDriver driver = new FirefoxDriver();
driver.Manage().Timeouts().ImplicitlyWait(TimeSpan.FromSeconds(10));
driver.Url = "http://somedomain/url_that_delays_loading";
IWebElement myDynamicElement = driver.FindElement(By.Id("someDynamicElement"));
```

远程WebDriver

捕获一个截图

Java

```
import java.io.File;
import java.net.URL;

import org.openqa.selenium.OutputType;
import org.openqa.selenium.TakesScreenshot;
import org.openqa.selenium.WebDriver;
import org.openqa.selenium.remoteAugmenter;
import org.openqa.selenium.remote.DesiredCapabilities;
import org.openqa.selenium.remote.RemoteWebDriver;

public class Testing {

    public void myTest() throws Exception {
        WebDriver driver = new RemoteWebDriver(
            new URL("http://localhost:4444/wd/hub"), DesiredCapabilities.firefox());

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

```
// RemoteWebDriver does not implement the TakesScreenshot class
// if the driver does have the Capabilities to take a screenshot
// then Augmenter will add the TakesScreenshot methods to the instance
WebDriver augmentedDriver = new Augmenter().augment(driver);
File screenshot = ((TakesScreenshot)augmentedDriver).getScreenshotAs(OutputType.FILE);
}
}
```

C#

```
// Add this class to your code and use this instead of RemoteWebDriver
// You will then be able to cast it to ITakesScreenshot and call GetScreenshot
```

```
public class ScreenShotRemoteWebDriver : RemoteWebDriver, ITakesScreenshot
{
    public ScreenShotRemoteWebDriver(Uri RemoteAdress, ICapabilities capabilities)
    : base(RemoteAdress, capabilities)
    {
    }
}
```

```
/// <summary>
/// Gets a <see cref="Screenshot"/> object representing the image of the page on the screen.
/// </summary>
/// <returns>A <see cref="Screenshot"/> object containing the image.</returns>
public Screenshot GetScreenshot()
{
    // Get the screenshot as base64.
    Response screenshotResponse = this.Execute(DriverCommand.Screenshot, null);
    string base64 = screenshotResponse.Value.ToString();

    // ... and convert it.
    return new Screenshot(base64);
}
}
```

```
// And then the usage would be:
```

```
ScreenShotRemoteWebDriver webDriver = new ScreenShotRemoteWebDriver(new
Uri("http://127.0.0.1:4444/wd/hub"), DesiredCapabilities.Firefox());
// ... do stuff with webDriver
Screenshot ss = ((ITakesScreenshot)webDriver).GetScreenshot();
string screenshot = ss.AsBase64EncodedString;
byte[] screenshotAsByteArray = ss.AsByteArray;
ss.SaveAsFile(activeDir + TestSuiteName + "/" + FileNanme + imageFormat, ImageFormat.Jpeg);
```

高级用户交互

Todo

浏览器启动管理

Todo

包含的主题：

- 恢复cookie
- 改变Firefox配置文件
- 运行带插件的浏览器

使用代理

Internet Explorer

最容易和推荐的方式是在即将运行测试的机器上手工设置代理。如果那是不可能的或你希望测试运行带有一个不同的配置或代理，则你可以使用Capabilities 对象，应用下面的技术。这将临时地改变系统的代理设置，并且当完成时改变回最初的状态。

```
String PROXY = "localhost:8080";
org.openqa.selenium.Proxy proxy = new org.openqa.selenium.Proxy();
proxy.setHttpProxy(PROXY)
.setFtpProxy(PROXY)
.setSslProxy(PROXY);
DesiredCapabilities cap = new DesiredCapabilities();
cap.setPreference(CapabilityType.PROXY, proxy);

WebDriver driver = new InternetExplorerDriver(cap);
```

Chrome

基本上与internet explorer是相同的。在windows , Chrome使用机器上与IE相同的配置。在Mac , Chrome使用系统首选项->网络设置 (System Preference -> Network settings) 。在Linux (Ubuntu) , Chrome使用系统->首选项->网络代理首选项 (System > Preferences > Network Proxy Preferences) , 可选地在 " /etc/environment " 设置http_proxy。本文档编写时, 还不知道如何编程地设置代理。

Firefox

Firefox维护代理配置在一个概要文件中。你可以预置代理在概要文件中, 然后使用这个Firefox概要, 或者你可以在行进中设置在概要中, 正如下面的示例所显示的。

```
String PROXY = "localhost:8080";
```

```
org.openqa.selenium.Proxy proxy = new org.openqa.selenium.Proxy();
```

```
proxy.setHttpProxy(PROXY)
```

```
.setFtpProxy(PROXY)
```

```
.setSslProxy(PROXY);
```

```
DesiredCapabilities cap = new DesiredCapabilities();
```

```
cap.setPreference(CapabilityType.PROXY, proxy);
```

```
WebDriver driver = new FirefoxDriver(cap);
```

Opera

Todo

HTML5

Todo

并行运行你的测试

Todo

© Copyright 2008-2012, Selenium Project. Last updated on Feb 02, 2012.

[1.14 Selenium用户指南 - 第六章 Selenium 1 \(Selenium RC\)\[1\]](#)

发表时间: 2012-02-19

引言

如同你在Selenium项目简史中读到的，Selenium RC长期以来是主Selenium项目，在WebDriver/Selenium合并产生Selenium 2.0，这个最新和更强大的工具以前。

Selenium 1 仍然被积极地支持（主要地在维护模式）并提供某些在Selenium 2.0暂时不可得到的某些特征，包括对几个语言（Java，Javascript，Ruby，PHP，Python，Perl和C#）以及对几乎所有的浏览器的支持。

Selenium RC是如何工作的

首先，我们描述Selenium RC的组件如何运转，以及每一个在运行你的测试脚本中的角色。

RC组件

Selenium RC组件是：

- 启动和杀死浏览器的Selenium服务器，解释和运行由测试程序传递的Selenese命令，和作为一个HTTP代理，解释和验证在浏览器和AUT之间传递的HTTP消息。
- 客户端库提供每种编程语言和Selenium RC服务器之间的接口。

此处是一个简单的架构图...

此图显示客户端库与服务器通讯，传递每个用于执行的Selenese命令。然后服务器使用Selenium 核心JavaScript命令传递Selenium命令到浏览器。浏览器使用它的JavaScript解释器，执行Selenium命令。这将运行你在你的测试脚本中指定的Selenese动作或验证。

Selenium服务器

Selenium服务器接受来自你的测试程序的Selenium命令，解释它们，并将运行这些测试的结果报告回你的程序。

RC服务器打包Selenium内核，并且自动注入到浏览器。这将在你的测试程序打开浏览器时发生（使用一个客户端库API函数）。Selenium内核是一个JavaScript程序，实际上是使用浏览器内置的JavaScript解释器解释和执行Selenium命令的一套JavaScript函数。

服务器使用简单的HTTP GET/POST请求接受来自你的程序的Selenese命令。这意味着你可以使用任何一种可以发送HTTP请求的编程语言，在浏览器上去自动化Selenium测试。

客户端库

客户端库提供允许你从你自己设计的程序运行Selenium命令的编程支持。每一个支持的语言，有一个不同的客户端库。一个Selenium客户端库提供一个编程接口（API），也就是，一套函数，可以从你自己的程序运行Selenium命令。在每个接口，有一个编程支持每个Selenese命令的编程函数。

客户端库取得一个Selenese命令，并传递它到Selenium服务器，用于处理对要测试的应用程序（Application Under Test - AUT）的一个指定的动作或测试。客户端库也接受命令的结果并传递回你的程序。你的程序可以接受这个结果并存储它到一个程序变量和报告做一个成功或失败，或可能采取校正措施，如果它是一个未预期的错误。

如此创建一个测试程序，你仅需编写一个使用客户端库API的，运行一套Selenium命令的程序，并且，可选地，如果你已经有一个在Selenium IDE创建的Selenese测试脚本，你可以生成Selenium RC代码。Selenium IDE可以转换（使用导出菜单项）它的Selenium命令到一个客户端驱动程序的API函数调用。详见Selenium IDE一章，有关从Selenium IDE导出RC代码的部分。

安装

安装对Selenium来说，有点用词不当。Selenium有一组可得到的库，用于你选择的编程语言。你可以从下载页面下载它们。

一旦你选择了一个工作语言，你仅仅需要：

- 安装Selenium RC服务器。
- 用语言特定的客户端驱动程序建立一个编程项目。

安装Selenium服务器

Selenium RC服务器只是一个Java的jar文件（selenium-server-standalone-`<版本号>`.jar），不需要任何特定的安装。只需下载这个zip文件，然后抽取服务器到期望的目录就足够。

运行Selenium服务器

在开始任何测试前，你必须启动服务器。到Selenium RC服务器安装的目录，然后从命令行控制台运行下面的命令。

```
java -jar selenium-server-standalone-<version-number>.jar
```

可以简化，通过创建一个批或Shell可执行程序（.bat在Windows，.sh在Linux）包含上面的命令。然后在桌面准备一个指向可执行文件的快捷方式，并且简单地双击这个图标启动服务器。

为了服务器的运行，你需要安装Java和正确地配置Path环境变量。你可以在控制台通过运行下面的命令，检查是否你有正确安装的Java。

```
java -version
```

如果你得到一个版本号，你已经准备好开始使用Selenium RC。

使用Java客户端驱动程序

- 从SeleniumHQ下载页面，下载Selenium java客户端驱动程序zip。
- 抽取selenium-java-<版本号>.jar文件。
- 打开你期望的Java IDE（Eclipse，NetBeans，Intellij，Netweaver，等等）。
- 创建一个java项目。
- 增加selenium-java-<版本号>.jar文件的引用到你的项目。
- 将selenium-java-<版本号>.jar文件 增加到你的项目的类路径
- 从Selenium IDE导出一个脚本到一个Java文件，并包含在你的Java项目中，或直接使用Selenium Java客户端API编写你的Selenium测试。API在本章的稍后介绍。你可以要么使用JUnit，或者TestNG来运行你的测试，或者你可以编写你自己的简单main（）程序。这些概念在本章的后面有解释。
- 从控制台运行Selenium服务器。
- 执行你的测试从Java IDE或从命令行。

有关Java 测试项目配置的详细信息，参见附录节在Eclipse配置Selenium RC和在Intellij配置Selenium RC。

使用Python客户端驱动程序

- 通过PIP安装Selenium，在SeleniumHQ下载页有使用说明的链接。
- 要么用Python编写你的Selenium测试，或从Selenium IDE导出 脚本到一个Python文件。
- 从控制台运行Selenium服务器。
- 从控制台执行你的测试，或你的Python IDE

有关Python客户端驱动器配置的详细信息，参见附录的Python客户端驱动程序配置。

使用.NET客户端驱动程序

- 从Selenium下载页面，下载Selenium RC。
- 抽取到文件夹。
- 下载和安装NUnit（注释：你可以使用NUnit做你的测试引擎。如果你还不熟悉NUnit，你也可以编写一个简单的main()函数去运行你的测试；不管怎样NUnit是一个非常有用的测试引擎。）
- 打开你期望的.NET IDE（Visual Studio，SharpDevelop，MonoDevelop）
- 创建一个类库（.dll）
- 增加对下面的DLL的引用：nmock.dll，nunit.core.dll，nunit.framework.dll，ThoughtWorks.Selenium.Core.dll，ThoughtWorks.Selenium.IntegrationTests.dll和ThoughtWorks.Selenium.UnitTests.dll
- 使用一个.NET语言（C#或VB.NET）编写Selenium测试，或从Selenium IDE导出脚本到一个C#文件，然后复制代码到你刚刚创建的类文件。
- 编写自己的简单main()程序或你可以包含NUnit在你的项目中运行你的测试。这些概念在本章稍后说明。
- 从控制台运行Selenium服务器
- 运行你的测试从IDE，从你的NUnit GUI或从命令行。

在Visual Studio中配置.NET客户端驱动程序的特定详细信息，参见附录的.NET客户端驱动程序配置

使用Ruby客户端驱动程序

- 如果你还没有RubyGems，从RubyForge安装它。
- 运行gem安装selenium-client。
- 在测试脚本的顶部增加需要的"selenium/client"
- 使用任何Ruby测试工具编写测试脚本（例如Test::Unit，Mini::Test或RSpec）
- 从控制台运行Selenium RC服务器
- 以你运行任何其它的Ruby脚本相同的方式执行你的测试

有关Ruby客户端程序的详细信息，参见Selenium-Client文档

从Selenese到一个程序

为了使用Selenium RC，这主要的任务是转换你的Selenese到一个编程语言。在本节，我们提供几个不同的语言特定的示例。

样品测试脚本

让我们从一个Selenese测试脚本示例开始。想象一下用Selenium IDE记录下面的测试。

```
open /
type q
selenium rc
clickAndWait btnG
assertTextPresent Results * for selenium rc
```

注释：这个示例使用Google搜索引擎页<http://www.google.com>

作为一个编程代码的Selenese

这是导出到每一个支持的编程语言的测试脚本（通过Selenium IDE）。如果你至少有面向对象编程语言的基本知识，你会理解Selenium如何运行Selenese命令，通过阅读下面的这些示例中的一个。要以一个特定的语言查看示例，选择这些按钮中的一个。

```
/** Add JUnit framework to your classpath if not already there
 * for this example to work
 */
package com.example.tests;

import com.thoughtworks.selenium.*;
import java.util.regex.Pattern;

public class NewTest extends SeleneseTestCase {
    public void setUp() throws Exception {
        setUp("http://www.google.com/", "*firefox");
    }

    public void testNew() throws Exception {
        selenium.open("/");
        selenium.type("q", "selenium rc");
        selenium.click("btnG");
```

```
selenium.waitForPageToLoad("30000");
assertTrue(selenium.isTextPresent("Results * for selenium rc"));
}
}
```

在下一节，我们会解释如何去构造一个测试程序，使用生成的代码。

编写你的测试

现在我们将演示如何编写你自己的测试，使用示例以支持的每一种编程语言。基本上有两个任务。

- 从Selenium IDE生成你的脚本到一个编程语言，可选地修改结果
- 编写一个非常简单的主程序执行生成的代码。

可选地，你可以采用一个测试引擎平台，像Java的JUnit或TestNG，或.NET的NUnit，如果你使用这些语言中的一种。

在这里，我们展示语言特定的示例。语言特定的API常常是彼此不同的，因此你可以为每一个找到各自的解释。

- Java
- C#
- Python
- Ruby
- Perl, PHP

Java

对Java，人们要么使用JUnit或TestNG作为测试引擎。某些开发环境，像Eclipse有对这些插件的直接支持。这使它更容易使用。讲授JUnit或TestNG超出了本文档的范畴，相关材料可以在线获得，也有可得到的出版物。如果你在一个使用Java的机构里，你的开发人员已经有这些测试框架中的一个的使用经验。

你可能想要重命名测试类，将“NewTest”改成你自己选择的某个名称。同样你需要改变在下面的语句中的浏览器打开参数。

```
selenium = new DefaultSelenium("localhost", 4444, "*iehta", "http://www.google.com/");
```

Selenium IDE生成的代码看起来像这个。示例有手工增加的注释，用于附加的澄清。

```
package com.example.tests;

// 指定测试包

import com.thoughtworks.selenium.*;

// 这是驱动程序的导入。你将使用这个实例化一个浏览器，并让它做你要求的。

import java.util.regex.Pattern;

// Selenium IDE增加一个Pattern模块。因为有时需要使用它进行正则表达式验证。

// 如果你使用它，你可以移除

public class NewTest extends SeleneseTestCase {

// 我们创建我们的Selenium测试案例

public void setUp() throws Exception {
    setUp("http://www.google.com/", "*firefox");

// 实例化和启动一个浏览器

}

public void testNew() throws Exception {
    selenium.open("/");
    selenium.type("q", "selenium rc");
    selenium.click("btnG");
    selenium.waitForPageToLoad("30000");
    assertTrue(selenium.isTextPresent("Results * for selenium rc"));

// 这些是测试步骤

}

}

C#
```

.NET客户端驱动程序与Microsoft.NET一起工作。可以与任何.NET测试框架，如NUnit或Visual Studio 2005 Team System。

Selenium IDE假定你使用NUnit作为你的测试框架。在下面生成的测试代码中，你可以看到。它包括用于NUnit的using语句，以及响应的NUnit属性，标识测试类中每一个成员函数的角色。

你可能想要重命名测试类，将“NewTest”改成你自己选择的某个名称。同样你需要改变在下面的语句中的浏览器打开参数。

```
selenium = new DefaultSelenium("localhost", 4444, "*iehta", "http://www.google.com/");
```

生成的代码看起来类似于这个。

你可以允许NUnit去管理你的测试的执行。或替代地，你可以编写简单的main()程序，实例化测试对象和依次运行三个方法中的每一个SetupTest()，TheNewTest()，和TeardownTest()。

Python

Pyunit是用于Python的测试框架，要学习Pyunit，参考它的[官方文档](#)。

基本的测试结构是：

Ruby

Selenium IDE生成适当的Ruby代码，但需要老的Selenium gem。因为Selenium的官方Ruby驱动程序是Selenium-Client gem，而不是老的Selenium gem。实际上，Selenium gem已经停止继续开发。

因此，建议更新任何IDE生成的Ruby脚本如下：

- 1、在行1，改变必须的“selenium”到必须的“selenium/client”
- 2、在行11，改变Selenium::SeleniumDriver.new到Selenium::Client::Driver.new

你可能也想要改变类名称到比“Untitled”包含更多意义的某个名称，并改变测试方法的名称“test_untitled”。

这里是一个，上面描述的，由Selenium IDE生成的Ruby代码修改而创建的简单示例。

Perl, PHP

文档团队的成员还没有用Perl或PHP语言使用过Selenium RC。如果你正用这两种语言中的一种使用Selenium RC，请联系文档团队（参见做出贡献一章）。我们很愿意包含来自你的某些示例或经验，以支持Perl和PHP用户。

© Copyright 2008-2012, Selenium Project. Last updated on Feb 02, 2012.

[1.15 Selenium用户指南 - 第六章 Selenium 1 \(Selenium RC\)\[2\]](#)

发表时间: 2012-02-19

了解API

Selenium RC API使用命名约定，假定你理解Selenese，大量的接口是自解释的。在此，我们解释最关键以及可能不太明显的内容。

启动浏览器

```
setUp("http://www.google.com/", "*firefox");
```

这些示例中的每一个打开浏览器，并分配浏览器“浏览器实例”给一个程序变量以代表这个浏览器。这个程序变量被使用于调用来自于浏览器的方法。

创建浏览器实例需要的参数是：

host指定服务器位于的计算机的IP地址。通常，这是与客户端运行的相同的机器，在这种情况下，可以是localhost。在某些客户驱动程序中，这是一个可选参数。port指定服务器侦听等待客户连接的TCP/IP socket端口。在某些客户驱动程序中这也是可选的。browser是你希望运行测试的浏览器。这时一个必须的参数。url测试中的应用程序的基url。这是所有的客户端库所必须的，这是用于启动browser-proxy-AUT通讯的完整信息。

注意，某些客户端库要求，要启动浏览器必须显式地调用它的start()方法。

运行命令

一旦你有一个初始化的浏览器，并分配给一个变量（通常命名为“selenium”），你可以从浏览器变量，通过调用不同的方法让它运行Selenese命令。

```
selenium.type( "field-id" , " string to type" )
```

在后台，浏览器将实际执行一个键入操作，基本上与一个用户在浏览器中键入一样，但使用定位器和在调用中指定的字符串。

报告结果

Selenium RC没有它自己的、用于报告的机制，他允许你使用选择的编程语言按需构造一个报告。这很好，但如果你仅仅想要那些已经为你准备好的东西哪？时常一个现存的类库或测试框架可以满足你的需要，比你自己开发测试报告代码更快。

测试框架报告工具

对许多编程语言来说，测试框架是可得到的。这些，以及它们的主要功能，那个提供的一个灵活的测试引擎用来执行你的测试代码的，包括用于报告结果的库代码。例如，Java有两个常用的测试框架，JUnit和TestNG，.NET也有它自己的NUnit。

我们不会在此讲解框架本身；那个超出了本用户指南的范畴。我们仅仅介绍框架的特征，那些与Selenium有关的以及你应用的某些技术。在互联网上，可以获得有关这些测试框架的很好的数据以及相关的信息。

测试报告库

用于使用你选择的编程语言报告测试结果的，特别创建的第三方库也是可得到的。这些时常支持各种各样的格式，包括HTML或PDF。

最好的方法是什么？

大部分测试框架的新手从使用框架内置的报告功能开始。从那里大多数人可以查看任何可得到的库，因为这比开发自己的要花更少的时间。当你开始使用Selenium时，毫无疑问你会为报告过程开始投入你自己的“打印语句”。那会逐步地引导你开发你自己的报告，或并行地使用一个库或测试框架。不管怎样，在一个初期的，但很短的学习曲线后，你会自然地开发适合你情况的效果最好的东西。

测试报告示例

为了演示，我们会引导你使用某些特定的、Selenium支持的某些语言的工具。在此列出的是由指南的作者常用的，以及得到广泛使用的工具。

测试报告为Java

- 如果Selenium测试案例的开发使用JUnit，则JUnit报告可以被使用于生成测试报告。详细说明参考JUnit Report
- 如果Selenium测试案例使用TestNG开发，则生成测试报告没有额外的工作需要做。TestNG框架生成一个HTML报告，列出测试结果的详细信息。更多的信息参见TestNG Report。
- ReportNG是TestNG框架的一个HTML报告插件。用于作为一个默认TestNG的HTML报告的替代。ReportNG提供一个简单、代码着色的测试结果视图。更多的信息参见ReportNG。
- 同样，如果需要一个好的概要报告可试用一下TestNG-xslt。一个TestNG-xslt看起来像这样。

更多的信息参见TestNG-xslt。

日志Selenese命令

- Logging Selenium可以用于生成在你的测试中所有Selenese命令以及每个执行成功或失败的报告，Logging Selenium扩展Java客户端驱动程序以增加Selenese的日志能力。请参考Logging Selenium。

测试报告为Python

- 当使用Python客户端驱动程序时，可以使用HTML TestRunner生成测试报告。参见HTML TestRunner。

测试报告为Ruby

- 使用Ruby，如果RSpec被使用于编写Selenium测试案例，那么它的HTML报告可以用于生成测试报告。参考RSpec Report以获得更多信息。

注释

如果你对一个语言无关的独立日志将做什么感兴趣，可以看看Selenium Server Logging。

增加某些调味瓶到你的测试

现在我们将接触到使用Selenium RC的全部理由，增加编程逻辑到你的测试。它与任何程序是相同的。程序流使用条件语句和迭代进行控制。此外，你可以报告进度信息使用I/O。在本节，我们将展示某些编程语言构造如何与Selenium结合解决公共的测试问题的示例。

你会发现，当你从简单的页面元素存在性的测试，到涉及多Web页的动态功能和变化的数据的测试转换时，你会需要编程逻辑来验证预期的结果。基本上，Selenium IDE不支持迭代和标准的条件语句。你可以使用嵌入在Selenese参数的javascript完成某些条件语句，然而迭代是不可能的，而且大多数的条件语句使用一个编程语言更容易完成。此外，你可能需要异常处理用于错误恢复。为这些以及其他的理由，我们编写了本节去演示公共编程技术的使用，以便给于你在自动化测试中一个极大的“验证能力”。

在本节的示例是使用C#和Java编写的，然而这代码是简单的，可以容易地改写到其他支持语言。如果你有有关面向对象编程语言的基础知识，你应该没有困难理解本节。

迭代

迭代是人们需要在他们的测试中完成的最普通的事情之一。例如，你可能希望执行一个搜索多次。或者，或许为了验证你的测试结果，你需要处理从一个数据库返回的“结果集”。

使用我们较早使用的相同的Google搜索示例，让我们检查Selenium搜索结果。这个测试可以使用Selenese：

```
open /
type q selenium rc
clickAndWait btnG
assertTextPresent Results * for selenium rc
type q selenium ide
clickAndWait btnG
assertTextPresent Results * for selenium ide
type q selenium grid
clickAndWait btnG
assertTextPresent Results * for selenium grid
```

这个代码已经被重复运行这相同的步骤3次。相同代码的多个拷贝不是一个好的编程实践，因为需要做更多的工作去维护。通过使用编程语言，我们可以迭代搜索结果，提供一个更灵活和可维护的解决方案。

用C#:

```
// 一个字符串的集合
```

```
String[] arr = {"ide", "rc", "grid"};
```

```
// 循环执行数组中的每隔字符串
```

```
foreach (String s in arr) {
    sel.open("/");
    sel.type("q", "selenium " + s);
    sel.click("btnG");
    sel.waitForPageToLoad("30000");
    assertTrue("Expected text: " + s + " is missing on page.", sel.isTextPresent("Results * for selenium " + s));
}
```

条件语句

为了演示在测试中使用条件语句，我们从一个示例开始。在运行Selenium测试时，一个公共的问题会出现，当在页面中某个预期的元素是不可得到时。例如，当运行下面的代码行时：

```
selenium.type("q", "selenium " + s);
```

如果元素“q”不在页面上，那么一个异常会抛出。

com.thoughtworks.selenium.SeleniumException: ERROR: Element q not found

这可能造成你的测试退出。对某些测试，那是你想要的。但当你的测试脚本有许多其他后面的测试的时候，这经常不是你想要。

一个更好的方法是首先验证元素是否真正地呈现，然后在没有的情况下采用替代的操作。让我们以Java为例。

// 如果元素是可得到的，则执行键入操作

```
if(selenium.isElementPresent("q")) {  
    selenium.type("q", "Selenium rc");  
} else {  
    System.out.printf("Element: " +q+ " is not available on page.")  
}
```

这个方法的好处是可以继续测试的执行，即使某些UI元素是不可得到的。

从你的测试执行JavaScript

执行JavaScript非常方便。Selenium API的getEval方法可以使用于从Selenium RC执行JavaScript。

考虑一个包含一个没有静态标识符的复选框的应用程序。在这种情况下，你可以从Selenium RC执行JavaScript获得所有复选框的id，然后检查它们。

```
public static String[] getAllCheckboxIds () {  
    String script = "var inputId = new Array();"; // Create array in java script.  
    script += "var cnt = 0;"; // Counter for check box ids.  
    script += "var inputFields = new Array();"; // Create array in java script.  
    script += "inputFields = window.document.getElementsByTagName('input');"; // Collect input  
    elements.  
    script += "for(var i=0; i<inputFields.length; i++) {"; // Loop through the collected elements.  
    script += "if(inputFields[i].id !=null " +  
    "&& inputFields[i].id !='undefined' " +  
    "&& inputFields[i].getAttribute('type') == 'checkbox') {"; // If input field is of type check box and input  
    id is not null.  
    script += "inputId[cnt]=inputFields[i].id ;" + // Save check box id to inputId array.  
    "cnt++;"; + // increment the counter.  
    "}" + // end of if.  
    "};"; // end of for.
```

```
script += "inputId.toString();" ;// Convert array in to string.  
String[] checkboxIds = selenium.getEval(script).split(","); // Split the string.  
return checkboxIds;  
}
```

计算页面中图像的数量：

```
selenium.getEval("window.document.images.length;");
```

记住在DOM表达式中使用windows对象，因为默认引用window，而不是测试窗口。

© Copyright 2008-2012, Selenium Project. Last updated on Feb 02, 2012.

[1.16 Selenium用户指南 - 第六章 Selenium 1 \(Selenium RC\)\[3\]](#)

发表时间: 2012-02-19

服务器选项

当服务器启动时，命令行选项可以使用于改变默认的服务器行为。

回想一下，服务器是通过运行下面的命令行启动的。

```
$ java -jar selenium-server-standalone-<version-number>.jar
```

要查看选项的列表，运行服务器带有-h选项。

```
$ java -jar selenium-server-standalone-<version-number>.jar -h
```

你会看到所有服务器可以使用选项的列表，以及每个选项的简短描述。提供的描述并不总是足够的，因此我们提供了某些比较重要的选项的注释。

代理配置

如果你的AUT在一个需要授权的HTTP代理的后面，那么你应该使用下面的命令配置http.proxyHost，http.proxyPort，http.proxyUser和http.proxyPassword。

```
$ java -jar selenium-server-standalone-<version-number>.jar -Dhttp.proxyHost=proxy.com -  
Dhttp.proxyPort=8080 -Dhttp.proxyUser=username -Dhttp.proxyPassword=password
```

多窗口模式

如果你正在使用Selenium 1.0，你可以跳过这一节，因为多窗口模式是默认的行为。然而，在1.0版本之前，Selenium默认地运行测试中的应用程序在一个子帧模式，正如显示在这儿的。

某些应用程序不能正确地运行在子帧模式，需要被装载到顶级帧的窗口。多窗口模式选项允许AUT运行在各自的窗口，而不是默认的帧模式，这样它就可以拥有需要的顶级帧。

对旧的Selenium版本，你必须使用下面的选项显式地指定多窗口模式。

-multiwindow

自Selenium RC 1.0起，如果你想要在一个单一的帧运行你的测试（例如使用早期Selenium版本的标准模式），你可以向Selenium 服务器要求这个使用下面的选项。

-singlewindow

指定Firefox配置文件

Firefox不会同时运行两个实例，除非你为每个实例指定各自的配置文件。Selenium 1.0和后面的版本，自动运行在一个各自的配置文件中，因此如果你正在使用Selenium 1.0，你可以跳过这一节。然而，如果你正在使用就的Selenium版本，或者如果你需要使用一个特殊的配置文件用于测试（诸如增加一个https证书或需要安装某些插件），你将需要显式地指定配置文件。

首先创建一个不同的Firefox配置文件，遵循下面的步骤。打开Windows的开始菜单，选择“运行...”，然后输入下面的命令：

```
firefox.exe -profilemanager  
firefox.exe -P
```

使用这个对话框创建新的配置文件。那么当你运行Selenium服务器的时候，告诉它使用新的Firefox配置文件，使用服务器命令行选项firefoxProfileTemplate，和使用文件名称以及目录路径指定配置文件。

-firefoxProfileTemplate "path to the profile"

警告

确信放置你的配置文件在一个新的不同于默认的文件夹中。Firefox配置文件管理员工具会删除在文件夹中的所有文件，如果你删除一个配置文件，而不管是否它们是配置文件或不是。

有关Firefox配置文件的更多信息，可以在Mozilla的知识库中找到。

使用-htmlSuite在服务器中直接运行Selenese

你可以在Selenium服务器直接运行Selenese html文件，通过传递html文件到服务器的命令行。例如：

```
java -jar selenium-server-standalone-<version-number>.jar -htmlSuite "*firefox"  
"http://www.google.com" "c:\absolute\path\to\my\HTMLSuite.html"  
"c:\absolute\path\to\my\results.html"
```

这将自动启动你的HTML测试集，运行所有的测试并保存一个带有结果的良好HTML报告。

注释

当适用这个选项时，服务器将启动测试并等待指定的秒数去完成测试；如果测试没有在指定的时间内完成，命令将退出，并带有一个非零的退出代码和不会生成结果文件。

命令行很长，因此当输入时要小心。注意这要求你传递一个HTML Selenese测试集，而不是一个单一的测试。同样需要知道-htmlSuite选项与-interactive是不兼容的，你不能同时运行两者。

Selenium服务器日志

服务器端日志

当启动Selenium服务器时，可以使用-log选项去记录Selenium服务器报告的有价值的信息到一个文本文件。

```
java -jar selenium-server-standalone-<version-number>.jar -log selenium.log
```

日志文件比标准的控制台文件更冗长（它包括调试级别的日志消息）。日志文件包括日志器名称，以及日志这个消息的线程的ID编号。例如：

```
20:44:25 DEBUG [12] org.openqa.selenium.server.SeleniumDriverResourceHandler -  
Browser 465828/:top frame1 posted START NEW
```

消息格式是

TIMESTAMP(HH:mm:ss) LEVEL [THREAD] LOGGER - MESSAGE

消息可能是多行的

浏览器端日志

在浏览器端的JavaScript（Selenium内核）也日志重要的消息；在许多情况下，这些可能对终端用户来说比正常的Selenium服务器日志更有用的。要存取浏览器日志，传递-browserSideLog参数给Selenium服务器。

```
java -jar selenium-server-standalone-<version-number>.jar -browserSideLog
```

要日志浏览器端的日志到一个文件，-browserSideLog必须与-log参数组合使用。

给指定的浏览器指定路径

你可以为Selenium RC指定浏览器的路径。这是有用的，如果你有相同浏览器的不同版本，并且希望使用特定的一个。同样，这被使用于允许你的测试运行在Selenium RC不直接知道的浏览器。当指定运行模式，使用带有浏览器可执行文件的完全路径的*custom限定符：

*custom <到浏览器的路径>

Selenium RC构架

注释

此主题试图解释Selenium RC后面的技术实现。对Selenium用户来说，了解这个不是必要的，但可能对理解某些在未来使用中遇到问题的理解是有帮助的。

详细地理解Selenium RC服务器如何工作，为什么使用代理注入和提升特权模式，必须首先理解同源策略。

同源策略

Selenium面对的主要限制是同源策略。这个安全限制适用于市面上的每个浏览器，它的目标是确保一个站点的内容永远不能被另外一个站点的脚本访问。同源策略强制任何在浏览器中装载的代码只能操作在Web站点的域。它不能执行在另外一个站点的函数。例如，浏览器装载JavaScript代码，当它装载www.mysite.com时，它不能对www.mysite2.com—even，如果它是你的另外一个站点，运行装载的代码。如果这是可能的，放置在任何你打开的站点的脚本就能够读取有关你的银行帐号的信息，如果你有一个打开在其他选项页的帐号页面。这被称为XSS（跨站点脚本）。

在此策略下，Selenium内核（和它的使所有魔法发生的JavaScript命令）必须放置在测试中的应用程序相同的源。

从历史上来看，Selenium内核有这个问题的限制，因为它使用JavaScript实现。然而，Selenium RC不受同源策略的限制。使用Selenium服务器作代理可以避免这个问题。基本上，它告诉浏览器，浏览器是工作在服务器提供的单一“伪”站点。

注释

你可以在有关同源策略和XSS的Wikipedia 页，找到关于这个主题的附加信息。

策略注入

Selenium用于避免同源策略的第一个方法是策略注入。在策略注入模式，Selenium服务器扮演做一个，在浏览器和测试中的应用程序之间的，配置客户HTTP代理。它然后在一个虚构的URL下伪装AUT。

[1] 代理是在两个部分之间传递皮球的第三人。它承担交付AUT到浏览器的“Web服务器”的作用。是一个代理给予Selenium服务器可以对AUT的真实URL进行“撒谎”的能力。

[2] 浏览器被启动，带有一个设置为localhost：4444作为HTTP代理的配置文件，这就是为什么任何浏览器做出的HTTP请求将通过Selenium服务器，和响应也通过它，而不是来自真实的服务器

此处是一个架构图。

当测试集以你喜欢的语言启动时，下面的过程会发生：

- 1、客户驱动器建立一个与Selenium服务器的连接。

- 2、Selenium RC服务器启动一个浏览器（或重用一个老的），带有一个URL，那个注入Selenium内核JavaScript进入装载浏览器的Web页面。
 - 3、客户驱动程序传递一个Selenese命令到服务器。
 - 4、服务器解释这个命令，然后触发相应的JavaScript执行，去执行在浏览器中的命令。
 - 5、Selenium-Core指示浏览器按第一个指令行动，典型地是打开一个AUT页面。
 - 6、浏览器接受这打开的请求，并从Selenium RC服务器请求Web站点的内容（被设置做浏览器使用的HTTP代理）。
 - 7、Selenium RC服务器与Web服务器通讯请求页面，一旦接受，它将发送这个页面到浏览器，并掩饰来源让浏览器以为页面来自同一个服务器（这允许Selenium-Core遵循同源策略）。
- 8、浏览器接受Web页，并绘制在为其保留的frame/window中。

提升权限

这个方法的流程非常类似于代理注入（Proxy Injection），主要的差异是浏览器以一个特殊的称之为提升权限的模式启动，那允许Web站点执行一般不许可的事情（如XSS，或填充文件上传输入域，以及对Selenium非常有用的任何事情）。通过使用这些浏览器模式，Selenium Core能够直接打开AUT和读取/与它的内容交互，而无需通过Selenium RC服务器传递整个AUT。

此处是一个架构图。

当测试集以你喜欢的语言启动时，下面的过程会发生：

- 1、客户驱动器建立一个与Selenium服务器的连接。
- 2、Selenium RC服务器启动一个浏览器（或重用旧的一个），带有一个装载Selenium-Core进入页面的URL。
- 3、Selenium-Core得到第一个来自客户驱动器的指令（通过另外一个对Selenium RC服务器做出的HTTP请求）。
- 4、Selenium-Core指示浏览器按第一个指令行动，典型地是打开一个AUT页面。
- 5、浏览器接受打开请求，并向Web服务器请求此页面。一旦浏览器接受此Web页面，就绘制在为其保留的frame/window中。

[1.17 Selenium用户指南 - 第六章 Selenium 1 \(Selenium RC\)\[4\]](#)

发表时间: 2012-02-19

处理HTTPS和弹出窗口安全

许多应用程序从HTTP切换到HTTPS，当它们需要发送加密的信息诸如密码或信用卡信息。这对今天的Web应用程序来说的是共同的。Selenium RC支持这个。

为确保HTTPS站点的真实，浏览器需要一个安全证书。否则，当浏览器使用HTTPS存取AUT时，它将假定应用程序是不可信任的。当如此时，浏览器将显示一个安全弹出窗口，这些弹出窗口不能使用Selenium RC关闭。

当在Selenium RC测试中处理HTTPS时，必须使用支持的运行模式，并为你处理安全证书。你需要在你的测试程序初始化Selenium时，指定该运行模式。

在Selenium RC 1.0 beta 2以及以后的版本为此运行模式使用*firefox或*iexplore。在早期的版本，包括Selenium RC 1.0 beta 1，使用*chrome或*iehta。使用这些运行模式，你可以不安装任何特定的安全证书；Selenium RC将为你处理这个。

在版本1.0，推荐运行模式*firefox或*iexplore。然而，有附加的*iexploreproxy和*firefoxproxy运行模式。这些仅仅是为了提供向后的兼容性，除非遗留的测试程序需要，不应该使用。它们有安全证书处理，以及多窗口运行，如果你的应用程序打开附加的浏览器窗口，的限制。

在早期的Selenium RC版本，*chrome或*iehta是支持HTTPS和处理安全弹出窗口的运行模式。它们被认为是试验模式，尽管它们相当稳定和许多人使用它们。如果你正在使用Selenium 1.0，你不需要也不应该使用这些旧的运行模式。

安全证书的解释

正常情况下，你的浏览器将信任你正在测试的，安装了你拥有的安装证书的应用程序。你可以在浏览器的选项或Internet属性（如果你不知道你的AUT的安全证书，可询问你的系统管理员）中检查它。当Selenium装载你的浏览器时，它会注入代码到浏览器和服务器的截获的消息。浏览器现在认为不信任的软件将试图伪装你的应用程序。它通过使用弹出消息警告你。

要回避这个，Selenium RC（再次重申，使用支持的运行模式）会临时地安装它自己的安全证书，到你的客户机器一个浏览器可以存取的位置。这将欺骗浏览器认为它访问的是一个不同于你的AUT的站点，而有效地抑制警告弹出窗口。

另外一个由早期的Selenium版本使用的方法是安装随你的Selenium安装提供的Cybervillians安全证书。然而，大部分的用户不再需要做这个；如果你正运行Selenium RC在代理注入模式，你可能需要显式地安装这个安全证书。

支持附加的浏览器和浏览器配置

Selenium API支持运行在除了Internet Explorer和Mozilla Firefox的多种浏览器。参见SeleniumHG.org的Web站点，查看支持的浏览器类型。此外，当浏览器不直接被支持时，你仍然可以运行Selenium测试，在你选择的浏览器，通过让你的应用程序启动浏览器使用“*custom”模式（也就是说，替代*firefox或*iexplore）。要使用这个，你在API的调用中，传递浏览器可执行文件的路径。这也可以使用交互模式在服务器中完成。

```
cmd=getNewBrowserSession&1=*custom c:\Program Files\Mozilla  
Firefox\MyBrowser.exe&2=http://www.google.com
```

使用不同的浏览器配置运行测试

正常地，Selenium RC自动配置浏览器，但如果你使用“*custom”模式启动浏览器，你可以强制Selenium RC启动浏览器使用不同的配置，而不是使用自动配置。

例如，你可以像这样使用一个自定义配置启动Firefox：

```
cmd=getNewBrowserSession&1=*custom c:\Program Files\Mozilla  
Firefox\firefox.exe&2=http://www.google.com
```

注意当以这种方式启动浏览器时，你必须手动配置浏览器使用Selenium服务器做为一个代理。正常地，这仅仅意味着打开你的浏览器首选项，并指定“localhost:4444”作为一个HTTP代理，但此操作方法可能对不同的浏览器，有根本地区别。详细信息可查阅你的浏览器文档。

Mozilla浏览器可能在如何启动和停止上改变。可能需要设置MOZ_NO_REMOTE环境变量使得Mozilla浏览器行为有更多的可预测性。Unix浏览器应该避免使用shell脚本启动浏览器；通常直接使用二进制可执行文件（如firefox-bin）会更好。

解决公共的问题

在开始使用Selenium RC时，有几个常常遇到的潜在的问题。我们将在此介绍它们以及它们的解决方案。

不能连接到服务器

当你的测试程序不能连接到Selenium服务器时，Selenium在你的测试程序中抛出一个异常。它会显示这个消息或一个类似的一个：

"Unable to connect to remote server (Inner Exception Message:
No connection could be made because the target machine actively
refused it)"

(using .NET and XP Service Pack 2)

如果你看见类似于这样的一个消息，确信你已经启动了Selenium服务器。如果已经启动，那么在Selenium客户端库和Selenium服务器之间存在一个连接问题。

当开始使用Selenium RC时，大多数人从在相同的机器上运行测试程序（带有Selenium客户端库）和Selenium服务器开始。要完成这个，使用“localhost”作为你的连接参数。我们推荐以这种方式开始，因为它减少使用初期潜在的网络问题的影响。假定你的操作系统有典型的网络和TCP/IP设置，你应该没有多少困难。事实上，许多人选择以这种方式运行测试。

不管怎样，如果你真的想要运行Selenium服务器在远程机器，应该确保在两个机器间有有效的TCP/IP连接。

如果你难以连接，可以使用常用的网络连接工具，如ping，telnet，ifconfig (Unix) /ipconfig (Windows) 等等，确保你有有效的网络连接。如果不熟悉这些，你的系统管理员可以帮助你。

不能装载浏览器

好的，不是一个友好的错误消息，很抱歉，但如果Selenium服务器不能装载浏览器你很可能看到这个错误

(500) Internal Server Error

这可能是由于

- Firefox (Selenium 1.0) 不能启动，因为浏览器已经打开和你没有指定一个不同的配置文件。参见在服务器选项的Firefox配置文件一节。
- 你正在使用的运行模式不匹配在你的机器上的任何浏览器。检查传递给Selenium的参数，当你的程序打开浏览器时。
- 你显式地指定了浏览器的路径（使用“*custom” - 见上文），但路径是不正确的。检查和确保路径是正确的。同样查看Selenium用户组确信你的浏览器和“*custom”参数没有已知的问题。

Selenium不能找到AUT

如果你的测试程序成功地启动浏览器，但浏览器不显示你正在测试的Web站点，最大的可能原因是你的测试程序没有使用正确的URL。

这可能容易发生。当你使用Selenium-IDE导出你的脚本，它插入一个虚拟的URL。你必须为你测试的应用程序手动改变这个URL到正确的一个。

当准备一个配置文件时，Firefox拒绝关闭

这时常发生在当你运行Selenium RC测试程序在Firefox时，但你已经有一个Firefox浏览器会话在运行，而你在启动Selenium服务器时没有指定一个不同的配置文件。这个来自测试程序的错误看起来像这样：

Error: java.lang.RuntimeException: Firefox refused shutdown while preparing a profile

此处是来自服务器的完整错误消息：

```
16:20:03.919 INFO - Preparing Firefox profile...
```

```
16:20:27.822 WARN - GET /selenium-server/driver/?cmd=getNewBrowserSession&1=*firefox&2=http%3a%2f%2fsage-webapp1.qa.idc.com HTTP/1.1
```

```
java.lang.RuntimeException: Firefox refused shutdown while preparing a profile
at org.openqa.selenium.server.browserlaunchers.FirefoxCustomProfileLauncher.waitForFullProfileToBeCreated(FirefoxCustomProfileLauncher.java:277)
...
```

```
Caused by: org.openqa.selenium.server.browserlaunchers.FirefoxCustomProfileLauncher$FileLockRemainedException: Lock file still present! C:\DOCUME~1\jsvec\LOCALS~1\Temp\customProfileDir203138\parent.lock
```

要解决这个，参见指定一个不同的Firefox配置文件一节。

版本问题

确信你的Selenium版本支持你的浏览器版本。例如，Selenium RC 0.92不支持Firefox3。有时候你可能是幸运的（我就是）。但不要忘记检查你正在使用的Selenium支持哪个浏览器的版本。如果有疑问，使用最新的，带有你的浏览器版本最广泛支持的Selenium发布版本。

Error message: “(Unsupported major.minor version 49.0)” while starting server

这个错误说你没有使用正确地Java版本。Selenium服务器需要Java 1.5或更高。

加倍仔细地检查你的Java版本，从命令行运行这个。

```
java -version
```

你应该看到一个显示Java版本的消息。

```
java version "1.5.0_07"
```

```
Java(TM) 2 Runtime Environment, Standard Edition (build 1.5.0_07-b03)
```

```
Java HotSpot(TM) Client VM (build 1.5.0_07-b03, mixed mode)
```

如果你看到一个较低版本号，你可能需要更新JRE，或者你可能仅仅需要增加它到你的PATH环境变量。

404 error when running the getNewBrowserSession command

如果当你试图打开一个页面在 “<http://www.google.com/selenium-server/>” 而你得到一个404错误，则一定是因为Selenium服务器没有正确地配置做一个代理。“Selenium服务器”目录没有在google.com上；只有当代理被适当地配置时才会存在。代理配置高度依赖于带有*firefox，*iexplore，*opera，或*custom的浏览器是如何启动的。

- *iexplore：如果浏览器启动是使用*iexplore，你可能有一个Internet Explorer代理设置问题。Selenium服务器试图在Internet选项控制面板配置全局代理设置。你必须确信当Selenium服务器启动浏览器时，它们被正确地配置。试着看看你的Internet选项控制面板。点击“连接（Connections）”选项页，然后点击“局域网设置（LAN Settings）”。如果你需要使用一个代理去存取你想要测试的应用程序，你需要启动Selenium服务器用“-Dhttp.proxyHost”；参见代理配置为了更多的详细信息。

- *custom：当使用*custom时，你必须正确地配置代理（手动），否则你会得到一个404错误。加倍检查你的正确配置的代理设置。为了检查是否正确地配置了代理，可以试试故意地错误配置浏览器。试试配置浏览器使用错误的代理服务器宿主或错误的端口。如果你已经成功地错误配置浏览器的代理设置，那么浏览器将不能够连接到Internet，这是一个确信一个人正在正确地调整相关设置的方式。

- 对其他的浏览器(*firefox，*opera)，我们自动为你硬编码代理，如此此功能没有已知的问题。如果你遇到404错误，并且已经仔细地遵从用户指南，可以张贴你的结果给Selenium用户组，从用户社区寻求某些帮助。

许可拒绝错误

该错误最常见的原因是你的会话正试图，通过跨域的边界（诸如存取一个页面从<http://domain1>，然后存取一个页面从<http://domain2>）或切换协议（从<http://domainX>移动到<https://domainX>）违反同源策略。

这个错误也可能发生，当JavaScript试图去查找UI对象，那个仍然是不可得到的（先于这个页面已经完成装载），或者不再是可得到的（在页面已经开始卸载）。这最常遇到是在带有几个部分的AJAX页面，或者装载和/或重新装载独立的较大页面的子帧。

错误可能是间歇的。经常不可能用调试器再生这种问题，因为当调试器的负荷增加到系统后，源于竞争条件的问题原因是不可再生的。许可问题被包括在本教程中。请仔细地阅读有关同源策略，代理注入的章节。

处理浏览器弹出窗口

有几种在Selenium测试中得到的“弹出窗口”。你可能不能运行Selenium命令关闭这些弹出窗口，如果它们是由浏览器激发而不是你的AUT。你可能需要知道如何管理它们。每一种类型的弹出窗口有区别地标定。

- HTTP基本验证对话框：这些对话框用于提示登录到站点的用户/密码。要登录到需要HTTP基本验证的站点，在URL中使用用户/口令，正如在RFC1738中的描述，像这样：

```
open( "http://myusername:myuserpassword@myexample.com/blah/blah/blah ").
```

- SSL证书警告：Selenium RC自动试图去伪装SSL证书，当服务器使能做一个代理时；参见HTTPS一节为了更多的信息。如果你的浏览器被正确地配置，你应该永远也不会看到SSL证书警告，但你可能需要配置浏览器信任我们危险的“CyberVillains” SSL证书授权。再次重申，有关如何完成请参考HTTPS一节。

- 模态的JavaScript的alert/confirmation/prompt对话框：Selenium试图隐藏这些对话框（通过替代window.alert，window.confirm和window.prompt），因此它们不会停止你的页面的执行。如果你看到一个alert弹出窗口，可能是因为它是在页面装载过程中激发的，那个通常对我们来说太早，以至于不能保护这个页面。Selenium包含用于断言和验证alert和confirmation弹出窗口的命令。参见在第四章中的有关这些主题的章节。

在Linux，为什么我的Firefox浏览器会话没有关闭？

在Unix/Linux，你必须直接调用“firefox-bin”，如此确信可执行文件在指定的路径上。如果通过一个shell脚本执行Firefox，当杀死浏览器Selenium RC时，也将杀死shell脚本，留下浏览器继续运行。你可以直接指定firefox-bin的路径，像这样。

```
cmd=getNewBrowserSession&1=*firefox /usr/local/firefox/firefox-bin&2=http://www.google.com
```

Firefox *chrome使用自定义的配置文件不工作

检查Firefox配置文件文件夹 -> prefs.js -> user_pref("browser.startup.page", 0); 注释此行像这样：
"//user_pref("browser.startup.page", 0);" 然后重试。

当正在装载父页面时，装载一个自定义的弹出窗口可以么（例如，先于父页的JavaScript window.onload() 函数运行）？

不可以。Selenium在装载页面时，依赖于解释器去决定窗口名称。这些解释器在捕捉新窗口时工作最佳，如果这些窗口被装载在onload()函数之后。Selenium不可能识别在onload()函数之前装载的窗口。

验证命令的问题

如果你从Selenium-IDE导出你的测试，你可能发现你从测试得到空的验证字符串（依赖于你使用的编程语言）。

注释：本节还没有开发。

Safari和MultiWindow模式

注释：本节还没有开发。

Firefox在Linux

在Unix/Linux，1.0以前的Selenium版本需要直接调用“firefox-bin”，如果你正在使用一个以前的版本，确信真实的可执行文件在指定的路径上。

在大多数的Linux发布中，实际的firefox-bin位于：

```
/usr/lib/firefox-x.x.x/
```

其中x.x.x是程序的版本号。如此要增加路径到用户搜索路径，你必须增加下面的到你的.bashrc文件：

```
export PATH="$PATH:/usr/lib/firefox-x.x.x/"
```

如果需要，你可以直接指定firefox-bin的路径到你的测试中，像这样：

```
"*firefox /usr/lib/firefox-x.x.x/firefox-bin"
```

IE和样式属性

如果你正运行测试在Internet Explorer，你不能定位元素使用样式属性。例如：

```
//td[@style="background-color:yellow"]
```

在Firefox，Opera或Safari，这会工作完美的，但在IE不行。IE按大写字母解释在@style中的关键字。因此，即使源代码是小写，你也应该使用：

```
//td[@style="BACKGROUND-COLOR:yellow"]
```

如果你的测试打算工作在多个浏览器中，这是一个问题，但你可以容易地编码你的测试去检测这种情况，并试图替代这个仅工作在IE的定位器。

Error encountered - “Cannot convert object to primitive value” with shut down of *googlechrome browser

要避免这个错误，你必须启动浏览器带有一个无效同源策略检查的选项。

```
selenium.start("commandLineFlags=--disable-web-security");
```

在那里我可以询问在这里没有回答的问题的？

试一下我们的用户组。

© Copyright 2008-2012, Selenium Project. Last updated on Feb 02, 2012.

1.18 Selenium用户指南 - 第七章 测试设计的考虑[1]

发表时间: 2012-02-19

测试设计入门

我们在这一章中提供的信息，对测试自动化的新手和有经验的QA专业人士都是有帮助的。此处我们描述最公共的自动化测试类型。我们也描述常用的、在测试自动化中的“设计模式”，用于改善你的自动化测试集的可维护性和可扩展性。富有经验的读者将觉得这些内容是有趣的，如果还没有使用这些技术。

测试的类型

你应测试你的应用程序的那个部分？这依赖于你的项目的各个方面：用户的期望，项目允许的时间，项目经理设置的优先级等等。一旦项目的边界被定义，你，作为测试者，将必定可以做出测试内容的决定。

在此，我们创建了几个术语，用于分类你可能在你的Web应用程序上执行的测试的类型。这些术语绝不是标准，尽管我们在此提出的概念典型地用于Web应用程序的测试。

静态内容测试

最简单的测试类型，内容测试，是一个简单的对静态的、没有变化的、UI元素的存在性测试。例如：

- 每一页是否有预期的页面标题？这可以被用于验证你的，在追随一个连接后发现一个预期的页面的测试。
- 应用程序的主页包含一个预期的图像在页面的顶端么？
- Web站点的每一页包含一个带有到公司联系人页面的链接，隐私权策略，和商标信息的页脚区域么？
- 每个页面开始于带有<h1>标记的标题文本么？并且，每一页有正确的文本在那个标题中么？

你可能或可能不需要内容测试。如果你页面的内容不太可能被影响，那么手工地测试页面内容可能是更有效率的。如果，例如，你的应用程序涉及的文件被移动到不同的位置，内容测试可能是有价值的。

链接测试

Web站点的频繁的错误来源是中断的链接，或缺失链接后面的页面。测试涉及点击每个链接并验证预期的页面。如果静态链接不是频繁地改变则手工测试可能是有效的。然而，如果你的Web设计者频繁地变更链接，或文件偶尔被重新定位，链接测试应该被自动化。

功能测试

这些是在你的应用程序中的特定功能的测试，需要某些类型的用户输入，并且返回某些类型的结果。时常一个功能测试会涉及多个页面，每个带有基于窗体的输入页面，包含输入域的集合，提交和取消操作，以及一个或多个响应页面。用户输入可以通过文本输入域，复选框，下拉列表框或任何浏览器支持的输入。

功能测试时常是你将自动化的最复杂的测试，但通常也是最重要的。典型的测试可能是登录，注册到站点，用户帐号操作，帐号设置改变，复杂的数据存取操作，以及其他。功能测试典型地反映用户的使用场景，用于限定功能，设计或你的应用程序。

动态元素测试

时常一个Web页面元素有一个唯一的标识符，用于唯一地在页面中定位那个元素。通常这被实现使用html标记的“id”或“name”属性。这些名称可能是静态的，诸如不变的字符串常量。它们也可以是动态生成的值，将随每个页面的示例而变化。例如，某些Web服务器可能在一个页面实例中命名一个显示的文档为doc3861，在另一个不同的页面实例中为doc6248，取决于用户存取啥文档。一个验证那个文档存在性的测试脚本，可能没有一个一致的标识符用于定位那个文档。时常，带有可变的标识符的动态元素是，在基于用户动作的某个类型的结果页面上。这必定依赖于Web应用程序的功能。

这是一个示例：

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

这显示一个HTML复选框。它的id (addForm:_ID74:_ID75:0:_ID79:0:checkBox) 是一个动态生成的值。下次打开相同的页面，它可能是一个不同的值。

Ajax测试

Ajax是支持动态改变用户接口元素的，可以动态改变元素而无需浏览器重新装载页面的技术，诸如动画，RSS订阅，以及实时的数据更新等等。有无数的方法，Ajax可以被用来更新页面上的元素。但，在Ajax驱动的应用程序中，被认为是最容易的方式是，数据可以从应用程序服务器存取，然后显示在页面上，而无需重新装载整个页面。只有页面的一部分，或严格地说元素自身被重新装载。

验证结果

断言和验证

何时应该使用断言 (assert) 命令，何时应该使用验证 (verify) 命令？这取决于你。差别在于当检查失败的时候你想要什么发生。你想要你的测试终止，或继续并简单地记录失败的检查么？

这是需要权衡的。如果你使用一个断言，测试将停止在那个点，不运行任何随后的检查。有时，或许是经常，那就是你想要的。如果测试失败，你会立即知道测试没有通过。测试引擎，诸如TestNG和JUnit有常用的开发

环境（第五章）的插件，可以方便地标记这些测试做失败的测试。优点：你可以立即看到检查是否通过。缺点：当一个检查失败时，其他的检查将永远也不会执行，这样你就无法得到有关它们状态的信息。

反之，验证命令不会终止测试。如果你的测试仅仅使用验证命令，不论检查是否找到缺陷你可以确保（假如没有非预期的异常出现）测试会运行完成。缺点：你必须做更多的检查测试结果的工作。也就是说，你不会得到回馈从TestNG或JUnit。你将需要检查控制台打印输出或日志输出的结果。并且你每次运行测试，你都需要花费时间浏览整个输出。如果你正在运行数百个测试，每一个有自己的日志，这将是一个耗时的的工作，则即时反馈的断言可能是更合适的。由于即时的反馈，断言比验证更常用。

权衡：assertTextPresent, assertElementPresent, assertText

你现在应该熟悉这些命令，和使用它们的技术。如果不熟悉，请首先参考第三章。当构造你的测试时，你将需要决定：

- 我仅仅检查页面上的文本存在么？（verify/assertTextPresent）
- 我仅仅检查页面上的HTML元素存在么？也就是说，文本，图像，或其他内容不做检查，仅仅和HTML标记相关。（verify/assertElementPresent）
- 我必须测试两者，元素和它的内容么？（verify/assertText）

没有直接的答案。它依赖于你的测试的需求。当然，取决于你测试的应用程序的需求。如果不能肯定，使用assertText，因为这是最严格的检查类型。你可以稍后改变它，但至少你不会错过任何潜在的失败。

Verify/assertText是最特定的测试类型。它会失败，如果HTML元素（标记）或文本不是你的测试所预期的。或许你的Web设计者正在频繁地改变页面，你不希望你的测试在每次他们做出改变时测试失败，因为这些改变是预期的。然而，假定你仍然需要检查页面上的某些东西，比如一个段落，或标题文本，或一个图像。在这种情况下，你可以使用verify/assertElementPresent。这将确保特定类型的元素存在（如果使用XPath还可以确保元素相对于页面上的其它元素存在）。但你不关心内容是啥。你仅仅关注一个特定的元素，比如说，一个图像在特定的位置。

要对做出这些类型的决定有感觉需要一些时间和一点点经验。它们是容易的概念，容易在你的测试中做出改变。

定位策略

选择一个定位策略

在页面中选择对象有许多种方法。但啥是这些定位类型的选择依据？回顾一下我们可以定位一个对象使用：

- 元素的id

- 元素的name属性
- 一个XPath语句
- 一个链接文本
- 对象文档模型 (DOM)

从测试的性能方面来说，假定页面源代码中的id或name属性具有良好的命名，使用id或name属性定位器是最有效的，而且是使你的测试代码更可读的。XPath语句需要花费更长的时间进行处理，因为浏览器必须运行它的XPath处理器。在Internet Explorer 7中，XPath据知是特别慢。通过链接文本定位时常是便利和性能良好的。尽管这个技术是特定于链接。同样，如果链接文本可能频繁改变，使用<a>元素定位可能是更好的选择。

尽管有时，你必须使用XPath定位器。如果页面源代码没有id或name属性，你可能不得不选择使用XPath定位器。（DOM定位器不再常用，因为XPath可以做它可以做的任何事情甚至更多。DOM定位器是可得到的，仅仅为了支持遗留的测试。）

使用XPath有一个使用id或name属性定位没有的优点。使用XPath（和DOM）你可以定位一个对象与页面上另外一个对象的关系。例如，如果一个链接必须出现在一个<div>节的第二个段落里，你可以使用XPath来指定。使用id和name定位器，你只可以指定它们在页面上出现，也就是说，在页面的某个地方。如果你必须测试一个图像，显示在公司的logo出现在页面的顶端，在一个页头，XPath可能是更好的选择。

定位动态元素

正如在较早的，有关测试类型一节所描述的，一个动态元素是一个页面元素，它的标识符随每个页面的实例而变化。例如，

```
<a class="button" id="adminHomeForm" onclick="return oamSubmitForm('adminHomeForm',  
'adminHomeForm:_ID38');" href="#">View Archived Allocation Events</a>
```

这个HTML锚标记定义一个带有“adminHomeForm” id属性的按钮。与大多数HTML标记比较，这是一个相当复杂的锚标记，但它仍然是一个静态标记。这个HTML在每次浏览器装载这个页面的时候都是相同的。它的id保持不变，在所有这个页面的实例中。也就是说，当这页被显示的时候，这个UI元素总是有这个标识符。如此，对你的点击这个按钮的测试脚本，仅仅需要使用下面的selenium命令。

```
click adminHomeForm
```

或者，在Selenium 1.0中

```
selenium.click("adminHomeForm");
```

你的应用程序，不管怎样，可能动态生成HTML，在那里对Web页的不同实例标识符可能是变化的。例如，对一个动态页面，HTML可能看起来像这样。

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

这定义一个复选框。它的id和name属性（addForm:_ID74:_ID75:0:_ID79:0:checkBox）是动态生成的值。在这种情况下，使用一个标准的定位器可能看起来像下面的。

```
click addForm:_ID74:_ID75:0:_ID79:0:checkBox
```

或者，在Selenium RC

```
selenium.click("addForm:_ID74:_ID75:0:_ID79:0:checkBox");
```

对给定的动态生成的标识符，这种方法可能不工作。下次页面装载，标识符可能是一个不同于使用在Selenium命令中的值，并因此将不会被找到。点击操作将失败带有一个“element not found（元素没有找到）”错误。

要纠正这一点，一个简单的解决方案是仅仅使用XPath定位器，而不是试图去使用id定位器。如此，对这个复选框你可以简单地使用：

```
click //input
```

或者，如果它不是第一个在页面上的input元素（它很可能不是），可以试一下一个更详细的XPath语句：

```
click //input[3]
```

或

```
click //div/p[2]/input[3]
```

如果不管如何，你都需要使用id去定位这个元素，则需要一个不同的解决方案。你可以从Web站点捕捉这个id，先于使用它在一个Selenium命令。它可像这样做。

```
String[] checkboxids = selenium.getAllFields(); // Collect all input IDs on page.
for(String checkboxid:checkboxids) {
    if(checkboxid.contains("addForm")) {
        selenium.click(expectedText);
    }
}
```

这个方法是可行的，如果只有一个复选框的id包含“expectedText”文本。

定位Ajax元素

正如提供在上面的测试类型节，一个带有Ajax实现的页面元素是一个可以被动态刷新的元素，而不必刷新整个页面。定位和验证一个Ajax元素的最佳方式是使用Selenium 2.0 WebDriver API。这是特别设计用来进行Ajax元素的测试的，在测试Ajax元素方面，Selenium 1.0 有一些限制。

在Selenium 2.0，你使用waitFor () 方法，等待一个页面元素变成可得到的。这参数是一个WebDriver实现的定位器By对象。这个被详细地解释在WebDriver一章。

要使用Selenium 1.0 (Selenium RC) 完成这个，涉及更多的代码，但也不困难。方法是去检查这个元素，如果它是不可得到的，则等待一个预定的时间段，然后再一次检查它。用一个带有预定义的超时，如果元素仍然没有发现就终止这个循环的，循环执行这个。

让我们考虑一个带有一个链接 (link=ajaxLink) 的页面，在点击一个页面上的按钮时 (没有刷新页面)。这可以使用一个for循环，用Selenium来处理。

```
// 循环初始
```

```
for (int second = 0;; second++) {
```

```
// 如果循环到达60秒，则中断这个循环
```

```
if (second >= 60) break;
```

```
// 搜索元素"link=ajaxLink"，如果可得到则中断循环
```

```
try { if (selenium.isElementPresent("link=ajaxLink")) break; } catch (Exception e) {}
```

```
// 暂停1秒
```

```
Thread.sleep(1000);
```

```
}
```

这肯定不是惟一的解决方案。Ajax在用户论坛是一个公共的主题，我们推荐搜索一下以前的讨论，看看其他人已经做了那些事情。

© Copyright 2008-2012, Selenium Project. Last updated on Feb 02, 2012.

1.19 Selenium用户指南 - 第七章 测试设计的考虑[2]

发表时间: 2012-02-21

封装Selenium调用

正如任何编程，你会想要使用实用函数去处理，遍及你的测试的重复代码。一种防止重复代码的方式是，使用你自己设计的函数或类方法，封装频繁使用的调用。例如，许多测试在一个页面上，会频繁地点击一个页面元素，并等待页面装载。

```
selenium.click(elementLocator);  
selenium.waitForPageToLoad(waitPeriod);
```

代替重复这段代码，你可以编写一个封装器方法执行这两个函数。

```
/**  
 * Clicks and Waits for page to load.  
 *  
 * param elementLocator  
 * param waitPeriod  
 */  
public void clickAndWait(String elementLocator, String waitPeriod) {  
    selenium.click(elementLocator);  
    selenium.waitForPageToLoad(waitPeriod);  
}
```

元素呈现的“安全操作”

另外一个常用的封装Selenium方法是，在执行某些操作前，检查元素是否呈现。这有时被称为“安全操作”。例如，下面的方法可能被使用于实现一个安全操作，

```
/**  
 * Selenium-RC -- Clicks on element only if it is available on page.  
 *  
 * param elementLocator  
 */  
public void safeClick(String elementLocator) {  
    if(selenium.isElementPresent(elementLocator)) {  
        selenium.click(elementLocator);  
    }  
}
```

```
} else {  
    // Using the TestNG API for logging  
    Reporter.log("Element: " +elementLocator+ ", is not available on page - "  
    +selenium.getLocation());  
}  
}
```

此示例使用Selenium 1 API , 但Selenium 2 也支持。

```
/**  
 * Selenium-WebDriver -- Clicks on element only if it is available on page.  
 *  
 * param elementLocator  
 */  
public void safeClick(String elementLocator) {  
    WebElement webElement = getDriver().findElement(By.XXXX(elementLocator));  
    if(webElement != null) {  
        selenium.click(elementLocator);  
    } else {  
        // Using the TestNG API for logging  
        Reporter.log("Element: " +elementLocator+ ", is not available on page - "  
        + getDriver().getUrl());  
    }  
}
```

在第二个示例中"XXXX"仅仅是一个占位符, 代表可以被调用的多个方法中的一个。

使用安全方法取决于测试开发者的判断。因此, 如果测试执行需要继续, 甚至在页面缺失元素的情况下, 则可以使用安全方法, 然而应该记录一个关于缺失元素的消息到日志。这基本上实现了一个带有报告的验证机制, 而非一个退出的断言。但如果元素为了能够执行进一步的操作, 必须在页面上是可得到的(例在一个门户站点的登录按钮), 那么安全方法技术不应该被使用。

UI映射

一个UI映射是一个存储测试集的所有定位器到一个地方, 为了当在一个AUT的UI元素的标识符或路径改变时, 方便修改的机制。测试脚本然后使用这个UI映射用来定位要测试的元素。基本上, 一个UI映射是一个测试脚本对象的仓库, 对应于被测试应用程序的UI元素。

什么使得UI映射有用？它的主要目的是使得测试脚本管理更容易。当一个定位器需要被编辑时，有一个中心的位置用于容易地查找对象，而不是必须从头至尾搜索测试脚本代码。同样，它允许在一个单一的位置改变标识符，而不是不得不在多个地方做出改变，在一个测试脚本中，或者在多个测试脚本中处理。

简言之，一个UI映射有两个重大的优点：

- 为UI对象使用一个中心位置，代替让他们分散在脚本中。这使得脚本维护更有效。
- 含义模糊的HTML标识符和名称可以给予更可读的，改善测试脚本的可读性的名称。

考虑下面的，难以理解的示例（Java）：

```
public void testNew() throws Exception {
    selenium.open("http://www.test.com");
    selenium.type("loginForm:tbUsername", "xxxxxxx");
    selenium.click("loginForm:btnLogin");
    selenium.click("adminHomeForm:_activitynew");
    selenium.waitForPageToLoad("30000");
    selenium.click("addEditEventForm:_IDcancel");
    selenium.waitForPageToLoad("30000");
    selenium.click("adminHomeForm:_activityold");
    selenium.waitForPageToLoad("30000");
}
```

对不熟悉AUT的页面源代码的任何人来说，这脚本可能是难以阅读的。甚至是应用程序的正式用户理解这样的脚本在做什么也有困难。一个更好的脚本可以是：

```
public void testNew() throws Exception {
    selenium.open("http://www.test.com");
    selenium.type(admin.username, "xxxxxxx");
    selenium.click(admin.loginbutton);
    selenium.click(admin.events.createnewevent);
    selenium.waitForPageToLoad("30000");
    selenium.click(admin.events.cancel);
    selenium.waitForPageToLoad("30000");
    selenium.click(admin.events.viewoldevents);
    selenium.waitForPageToLoad("30000");
}
```

现在，使用某些注释和空白以及UI映射标识符做出一个非常可读的脚本。

```
public void testNew() throws Exception {

// 打开应用程序URL

selenium.open("http://www.test.com");

// 提供管理用户名称

selenium.type(admin.username, "xxxxxxx");

// 点击一个登录 ( Login ) 按钮

selenium.click(admin.loginbutton);

// 点击创建新事件 ( Create New Event ) 按钮

selenium.click(admin.events.createnewevent);
selenium.waitForPageToLoad("30000");

// 点击取消 ( Cancel ) 按钮

selenium.click(admin.events.cancel);
selenium.waitForPageToLoad("30000");

// 点击查看旧事件 ( View Old Events ) 按钮

selenium.click(admin.events.viewoldevents);
selenium.waitForPageToLoad("30000");
}
```

一个UI映射可以被实现，使用各种各样的方法。一个人可以创建一个类或结构，那个仅仅存储公共的字符串变量，每一个存储一个定位器。替代地，可以使用一个存储键值对的文件。在Java一个包含键/值对的属性文件可能是最佳的方法。

考虑一个属性文件prop.properties，为来自上一个示例的UI元素分配具有亲和力的标识符别名。

```
admin.username = loginForm:tbUsername
admin.loginbutton = loginForm:btnLogin
admin.events.createnewevent = adminHomeForm:_activitynew
admin.events.cancel = addEditEventForm:_IDcancel
admin.events.viewoldevents = adminHomeForm:_activityold
```

定位器会仍然引用HTML对象，但你已经在测试脚本和UI元素之间引入了一个抽象层。值读取自属性文件，并且使用在测试类，以实现UI映射。参考下面的链接，以获取关于Java属性文件的更多信息。

页面对象设计模式

页面对象是一个，用于增强测试的可维护性和减少代码重复，在测试自动化领域流行的设计模式。一个页面对象是一个面向对象的类，作为你的AUT页面的接口。当测试需要与页面的UI交互时，使用页面对象类的方法。好处是，如果页面UI改变，测试本身不需要改变，仅仅在页面对象中的代码需要改变。结果是，支持新UI的所有改变都位于同一个地方。

页面对象设计模式提供了下面的优点。

- 1、有一个测试代码和页面特定代码的清晰分离，出入定位器（或者定位器的恶使用，如果你使用UI映射）和布局。
- 2、有一个单一的页面提供的服务和操作的仓库，而不是让这些服务散布在测试中。

在两种情况下，这允许任何由于UI改变而需要的修改可以在一个地方做出。有关这项技术的有用信息可以在无数的blog中找到，因为这个测试设计模式正得到广泛地使用。我们鼓励希望了解更多的读者，在英特网上搜索有关这个主题的blog。很多人写过这个设计模式，并且可以提供超越本用户指南的有用技巧。为让你开始，我们使用一个简单的示例演示一个页面对象。

First, consider an example, typical of test automation, that does not use a page object.

首先，考虑一个示例，典型的测试自动化，没有使用页面对象。

```
/**
 * Tests login feature
 */
public class Login {

    public void testLogin() {
        selenium.type("inputBox", "testUser");
        selenium.type("password", "my supersecret password");
        selenium.click("sign-in");
        selenium.waitForPageToLoad("PageWaitPeriod");
        Assert.assertTrue(selenium.isElementPresent("compose button"),
            Login was unsuccessful");
    }
}
```


使用这个方法有两个问题。

1、在测试方法和AUT的定位器（在这个示例中是id）之间没有分离；两者纠缠在一个方法中。如果AUT的UI改变它的标识符，布局，或者一个登录如何输入和继续，这测试本身必须改变。

2、id定位器散布在多个测试中，那些必须使用这个个登录页的所有测试中。

应用页面对象技术，这个示例可以重写，像下面的这个登录页的页面对象示例。

```
/**  
  
 * 页面对象封装登录页  
  
 public class SignInPage {  
  
 private Selenium selenium;  
  
 public SignInPage(Selenium selenium) {  
 this.selenium = selenium;  
 if(!selenium.getTitle().equals("Sign in page")) {  
 throw new IllegalStateException("This is not sign in page, current page is: "  
 +selenium.getLocation());  
 }  
 }  
  
 /**  
 * Login as valid user  
 *  
 * @param userName  
 * @param password  
 * @return HomePage object  
 */  
 public HomePage loginValidUser(String userName, String password) {  
 selenium.type("usernamefield", userName);  
 selenium.type("passwordfield", password);  
 selenium.click("sign-in");  
 selenium.waitForPageToLoad("waitPeriod");  
  
 return new HomePage(selenium);  
 }
```

```
}  
}
```

一个Home页面的页面对象可能像这个。

```
/**  
 * Page Object encapsulates the Home Page  
 */  
public class HomePage {  
  
    private Selenium selenium;  
  
    public HomePage(Selenium selenium) {  
        if (!selenium.getTitle().equals("Home Page of logged in user")) {  
            throw new IllegalStateException("This is not Home Page of logged in user, current page" +  
                +selenium.getLocation());  
        }  
    }  
  
    public HomePage manageProfile() {  
        // Page encapsulation to manage profile functionality  
        return new HomePage(selenium);  
    }  
  
    /*More methods offering the services represented by Home Page  
    of Logged User. These methods in turn might return more Page Objects  
    for example click on Compose mail button could return ComposeMail class object*/  
}
```

现在登录测试可以使用这两个页面对象如下：

```
/**  
 * Tests login feature  
 */  
public class TestLogin {  
  
    public void testLogin() {  
        SignInPage signInPage = new SignInPage(selenium);
```

```
HomePage homePage = signInPage.loginValidUser("userName", "password");
Assert.assertTrue(selenium.isElementPresent("compose button"),
Login was unsuccessful");
}
}
```

在页面对象可以如何设计方面，有大量的灵活性，但有几个基本的规则，用于达成你的测试代码想要的可维护性。页面对象本身应该永远不要做验证和断言。这是你的测试部分，并且应该总是在测试代码中，永远不要在一个页面对象中。页面对象将包含页面的呈现，和页面通过方法提供的服务，但在页面对象中没有代码与即将被测试的内容相关。

有一个，唯一的，验证，可以和应该被写在页面对象里，那就是去验证那个页面，和可能的在页面上的关键元素，应被正确地装载。这个验证应该在页面对象实例化时完成。在上面的示例中，SignInPage和HomePage构造函数检查预期的页面是可得到的，而且已经准备好接受来自测试的请求。

一个页面对象不一定需要代表一个完整的页面。页面对象设计模式可以使用于代表在一个页面上的组件。如果在AUT的一个页面上有多个组件，每一个组件有一个分离的页面对象，这可以改善它的可维护性。

其他的设计模式也可以用于测试。有些人使用页面工厂实例化他们的页面对象。讨论这些设计模式超出了本用户指南的范畴。这里，我们仅仅希望引入这些概念，让读者意识到某些可以做的事情。正如较早提及的，很多人发布过有关这个主题的blog，我们鼓励读者查找有关这些主题的blog。

数据驱动测试

数据驱动的测试指的是使用带有变化的数据的相同的测试（或多个测试）多次。这些数据集常常来自外部文件，诸如.csv文件，文本文件，或可能装载自一个数据库。数据驱动的测试是一个常用的数据测试自动化技术，使用于验证面对许多变化的输入的应用程序。当测试是为变化的数据设计时，输入数据可以扩展，本质地创建附加的测试，而无需改变测试代码。

In Python:

这Python脚本打开一个文本文件。该文件在每一行包含一个不同的搜索串。代码然后保存这个到一个字符串数组，并迭代通过这个数组，执行一个搜索和断言每个搜索串。

只是一个非常基本的示例，但这思想是去运行一个带有变化的数据的测试，可以被容易地完成，使用编程或脚本语言。为了更多的示例，参见Selenium RC wiki，从电子表格读取数据或使用TestNG的数据提供者能力的示例。此外，这是一个在测试自动化专业人士中众所周知的主题，包括那些不使用Selenium的，如此在Internet上可以找到许多有关这个“数据驱动测试”主题的blog

数据库验证

另外一个共同的测试类型是比较在UI中的数据与实际存储在AUT数据库中的数据。因为你也可以从一个编程语言做数据库查询，假定你有数据库支持函数，你可以使用它们去存取数据，然后使用这个数据去验证AUT显示的内容是正确的。

考虑这个取自数据库的注册邮件地址的示例，然后稍后与UI进行比较。一个建立一个DB连接，然后从DB存取数据的示例可能看起来像这样。

In Java:

```
// 装载Microsoft SQL Server JDBC driver。

Class.forName("com.microsoft.sqlserver.jdbc.SQLServerDriver");

// 准备连接url

String url = "jdbc:sqlserver://192.168.1.180:1433;DatabaseName=TEST_DB";

// 得到数据库连接

public static Connection con =
DriverManager.getConnection(url, "username", "password");

// 创建可以使用于编写DDL和DML SQL语句的语句对象

public static Statement stmt = con.createStatement();

// 通过Statement.executeQuery方法发出SQL SELECT语句到数据库

// 返回一个包含请求信息，以及带有数据行的ResultSet对象

ResultSet result = stmt.executeQuery
("select top 1 email_address from user_register_table");

// 从 "result" 对象存取 "email_address" 值

String emailaddress = result.getString("email_address");

// 使用emailAddress值登录到应用程序

selenium.type("userID", emailaddress);
selenium.type("password", secretPassword);
selenium.click("loginButton");
selenium.waitForPageToLoad(timeOut);
```

```
Assert.assertTrue(selenium.isTextPresent("Welcome back" +emailaddress), "Unable to log in for user"
+emailaddress)
```

这个简单的，从一个数据库存取数据的Java示例。

© Copyright 2008-2012, Selenium Project. Last updated on Feb 02, 2012.

1.20 Selenium用户指南 - 第八章 Selenium-Grid

发表时间: 2012-02-21

请参考[Selenium Grid Web站点](#)

此节还未开发。如果有在Selenium Grid 方面有经验的社区的成员，并愿意作出贡献，请联系文档团队。我们期盼着你的贡献。

© Copyright 2008-2012, Selenium Project. Last updated on Feb 02, 2012.

1.21 Selenium用户指南 - 第九章 用户扩展

发表时间: 2012-02-21

用户扩展

注释：本节接近完成，但还没有被审核和编辑。

引言

通过增加你自己的动作，断言，和定位器策略扩展Selenium可能是相当简单的。增加Javascript方法到Selenium对象原型和PageBot对象原型。在启动时，Selenium会自动查找在这些原型中的方法，使用名称模式来识别那个是动作，断言和定位器。下面的示例给出了Selenium可以如何用Javascript扩展的指导。

动作

在Selenium原型中，以“do”开始的所有方法被添加做动作。对每个动作foo，也有一个注册的动作fooAndWait。一个动作可以最多带有两个参数，在测试中被传递做第二和第三列值。例如：增加一个“typeRepeated”动作到Selenium，键入文本两次到一个文本框。

```
Selenium.prototype.doTypeRepeated = function(locator, text) {
```

```
// 所有的定位器策略由"findElement"自动处理
```

```
var element = this.page().findElement(locator);
```

```
// 创建要键入的文本
```

```
var valueToType = text + text;
```

```
//替代元素文本用新文本
```

```
this.page().replaceText(element, valueToType);
```

```
};
```

存取器/断言

在Selenium原型中，所有的getFoo和isFoo方法被增加做存取器方法（storeFoo），为每一个存取器都有注册的一个assertFoo，verifyFoo和waitForFoo。一个断言方法带有最多两个参数，在测试中被传递做第二和第三列值。你也可以定义你自己断言做简单的“assert”方法，也会自动生成“verify”和“waitFor”命令。例

如：增加一个valueRepeated断言，用来确保元素只是由提供的文本经重复组成。在测试中，可得到的2个命令是assertValueRepeated 和 verifyValueRepeated。

```
Selenium.prototype.assertValueRepeated = function(locator, text) {
```

```
// 所有的定位器策略由"findElement"自动处理
```

```
var element = this.page().findElement(locator);
```

```
// 创建要验证的文本
```

```
var expectedValue = text + text;
```

```
//得到实际的元素
```

```
var actualValue = element.value;
```

```
// 确信实际的值匹配预期的
```

```
Assert.matches(expectedValue, actualValue);
```

```
};
```

原型生成附加的命令

所有的在Selenium原型的getFoo和isFoo方法自动地导致可得到的storeFoo，assertFoo，assertNotFoo，verifyFoo，verifyNotFoo，waitForFoo，waitForNotFoo命令。例如，如果你增加一个getTextLength()方法，下面的命令将自动可得到：storeTextLength，assertTextLength，assertNotTextLength，verifyTextLength，verifyNotTextLength，waitForTextLength，和waitForNotTextLength命令。

```
Selenium.prototype.getTextLength = function(locator, text) {
```

```
return this.getText(locator).length;
```

```
};
```

同样要注意上面描述的assertValueRepeated也可以实现做isValueRepeated，带有额外地好处，同样自动可以得到assertNotValueRepeated，storeValueRepeated，waitForValueRepeated 和 waitForNotValueRepeated。

定位器策略

所有的在PageBot原型的locateElementByFoo被增加做定位器策略。一个定位器策略带有2个参数，第一个是定位串（减去前缀），和第二个是被搜索的文档。例如：增加一个“valuerepeated=”定位器，查找第一个元素的value属性等于提供的值的重复。


```
// "inDocument"是一个你正在搜索的文档

PageBot.prototype.locateElementByValueRepeated = function(text, inDocument) {

// 创建要搜索的文本

var expectedValue = text + text;

// 循环通过所有的元素，查找一个有value === 我们的预期值的

var allElements = inDocument.getElementsByTagName("*");

for (var i = 0; i < allElements.length; i++) {

var testElement = allElements[i];

if (testElement.value && testElement.value === expectedValue) {

return testElement;

}

}

return null;

};
```

在Selenium IDE中只用用户扩展

在Selenium IDE中用户扩展非常容易使用

- 1、创建你的用户扩展，并存储作user-extensions.js。尽管这个名称在技术上不是必须的，保持一致性是一个良好的实践。
- 2、打开Firefox，然后打开Selenium IDE
- 3、在工具栏上，点击Options
- 4、在Selenium核心扩展（Selenium Core Extensions）点击浏览（Browse），并查找user-extensions.js文件，然后点击OK。
- 5、你的用户扩展仍然不会被装载，你必须关闭和重新启动Selenium IDE。
- 6、在你的空白测试中，创建一个新命令，你的用户扩展现在是一个在命令下拉列表中的选项。

在Selenium RC中，使用用户扩展

如果你Google “Selenium RC user-extension” 十次，你会找到十个不同的使用这个特征的方法，下面，是官方的Selenium建议的方法。

示例

C#

1、放置你的用户扩展在与你的Selenium服务器相同的目录下。

2、如果你正在使用由Selenium IDE生成的客户端代码，你将需要做出几个小的修改。首先，你会需要在类的作用域创建一个HttpCommandProcessor对象（在SetupTest方法的外面，紧接着私有的private StringBuilder verificationErrors下面）

3、接着，实例化HttpCommandProcessor对象，使用下面的测试步骤。

```
HttpCommandProcessor proc;
```

```
proc = new HttpCommandProcessor("localhost", 4444, "*iexplore", "http://google.ca/");
```

4、使用你创建的HttpCommandProcessor实例化DefaultSelenium 对象

```
selenium = new DefaultSelenium(proc);
```

5、在你的测试代码中，用HttpCommandProcessor的方法DoCommand()，执行你的用户扩展。这个方法带两个参数：一个字符串标识你想要使用的用户扩展方法，以及字符串数组传递做参数。注意你的函数的第一个字母是小写字母，而不用管在你的用户扩展中的大小写。Selenium自动地处理去保证公共Javascript的命名惯例。因为Javascript是大小写敏感的，你的测试将失败，如果你用大写字母开始这个命令。inputParams是你希望传递给Javascript用户扩展的参数数组。在上述情况中，数组只有一个字符串，因为我们的扩展只有一个参数，而更长的数组将映射每个索引到相应的参数。牢记为Selenium IDE设计的用户扩展只能带有两个参数。

```
string[] inputParams = {"Hello World"};
```

```
proc.DoCommand("alertWrapper", inputParams);
```

6、使用-userExtensions 参数并传递你的user-extensions.js文件启动测试服务器。

```
java -jar selenium-server.jar -userExtensions user-extensions.js
```

© Copyright 2008-2012, Selenium Project. Last updated on Feb 02, 2012.