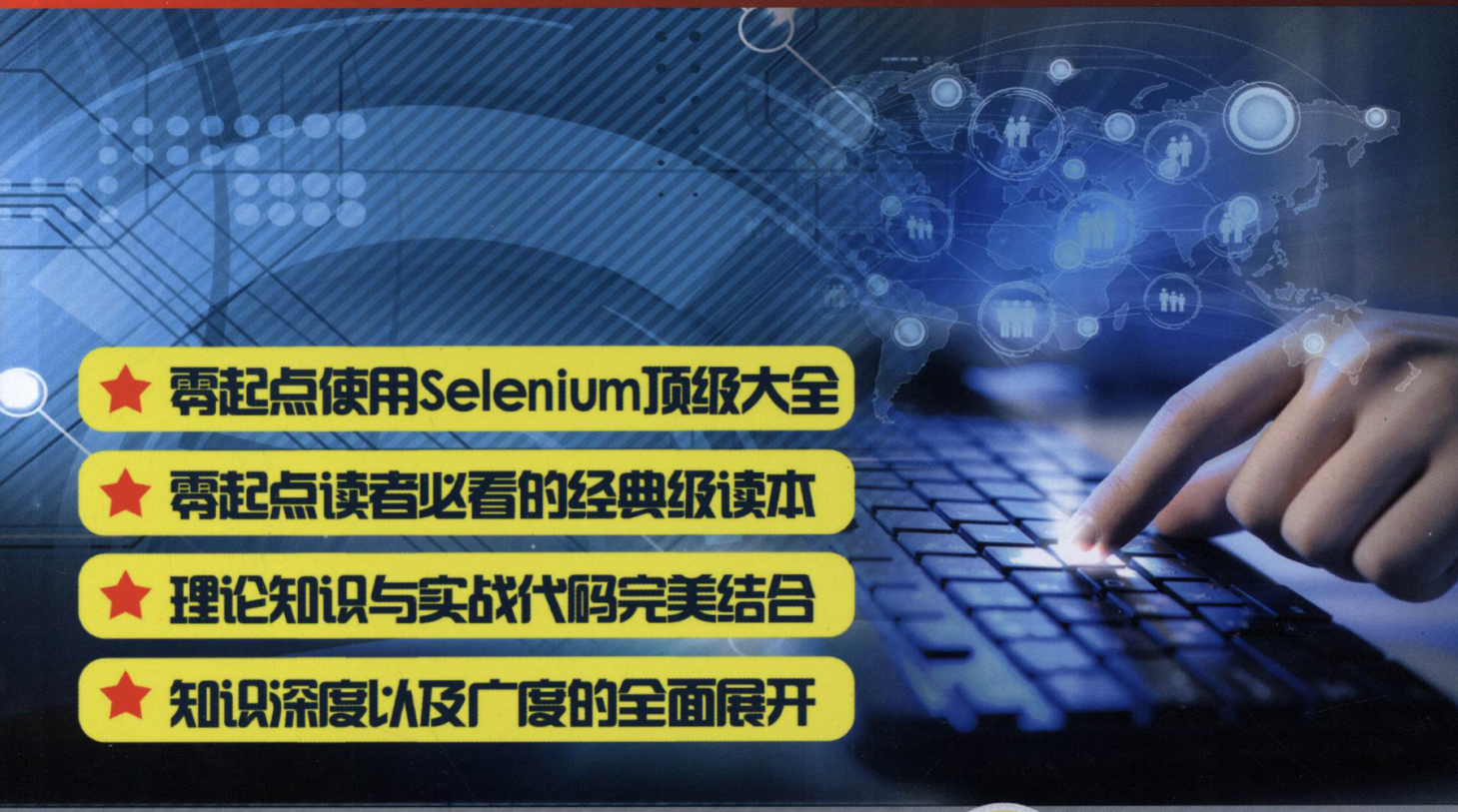
白洛 编著

基于Selenium 2的自动化测试

——从入门到精通



- 零起点使用Selenium顶级大全
- 零起点读者必看的经典级读本
- 理论知识与实战代码完美结合
- 知识深度以及广度的全面展开

机械工业出版社
CHINA MACHINE PRESS

基于 Selenium 2 的自动化测试
——从入门到精通

白　洛　编著

机 械 工 业 出 版 社

本书向开发人员和测试人员展示了如何使用 Selenium 进行 Web 自动化测试。本书从自动化测试的特点娓娓道来，引出了主角 Selenium；介绍了 Selenium IDE 的使用；讲述了获取页面元素和定位页面元素的多种方式；讲解了 WebDriver 与 Selenium RC 的区别、WebDriver 的架构和设计理念；阐述了 WebDriver 的部署、基本使用方法、对 HTML5 特性的支持，以及如何迁移已有的 Selenium RC 代码到 Selenium WebDriver 的解决方案；展示了在嵌入式系统中使用 Selenium 进行自动化测试的方法，涵盖 Android、iOS 和 Raspberry Pi 等；此外，本书还描述了 Selenium Grid 的架构和部署方法；最后介绍了 Selenium 周边的测试工具和套件。无论从深度还是广度上，本书为开发人员和测试人员学习并掌握 Selenium 提供了一定的辅助作用。

本书适合开发人员、测试人员、测试管理人员使用，也适合作为大中专院校相关专业师生的学习用书，以及培训学校的教材。

图书在版编目（CIP）数据

基于 Selenium 2 的自动化测试：从入门到精通/白洛编著. —北京：机械工业出版社，2014.7（2015.8 重印）

ISBN 978-7-111-46783-0

Ⅰ.①基… Ⅱ.①白… Ⅲ.①软件工具-自动检测 Ⅳ.①TP311.56

中国版本图书馆 CIP 数据核字（2014）第 106088 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）
策划编辑：张俊红 责任编辑：闾洪庆 版式设计：赵颖喆
责任校对：张玉琴 封面设计：路恩中 责任印制：李 洋
三河市宏达印刷有限公司印刷
2015 年 8 月第 1 版第 2 次印刷
184mm×260mm · 13.75 印张 · 334 千字
标准书号：ISBN 978-7-111-46783-0
定价：39.80 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务		网络服务	
社服务中心：	（010）88361066	教材网：	http://www.cmpedu.com
销售一部：	（010）68326294	机工官网：	http://www.cmpbook.com
销售二部：	（010）88379649	机工官博：	http://weibo.com/cmp1952
读者购书热线：	（010）88379203	**封面无防伪标均为盗版**	

前　言

多年以前，自动化测试还不够成熟。人们往往把自动化测试当作手工测试的附属品，就像当年把测试当成开发的附属品一样。从 1997 年的开源运动开始，开源软件在过去的十多年间蓬勃发展。大到操作系统，小到一个功能模块，开源运动已经渗透到各个领域并结出丰硕的果实。在软件测试领域，特别是自动化测试领域，开源软件涵盖了方方面面。从单元测试、功能测试到性能测试，从 Web 页面测试到数据库、多媒体、通信等应用领域的测试，都可以发现开源软件的身影。开源软件给自动化测试技术带来巨大变革和进步。

Web 自动化测试从无到有、由浅入深，已经逐步走向成熟。互联网的出现亦改变了许多软件研发的模式。在互联网和软件产业，一切变化都如此迅速，以至于许多最近几年才出版的软件测试方面的书籍内容已跟不上软件更新迭代的速度。许多红极一时的测试工具和测试实践，在当前的环境下效率会大打折扣。作为 Web 测试工具中的佼佼者——Selenium 已经走过了近 10 个春秋，从 Selenium 1 发展到 Selenium 2，可谓“十年磨一剑”。它是如何做到与时俱进并继续引领业界潮流的呢？本书将由浅入深地展示 Selenium 的独特魅力之所在。

而国内的 Web 自动化测试资源，经过这么多年的发展，已然在自动化测试领域遍地开花。无论测试人员的水平，还是测试工具的成熟度，抑或测试资料的完备性，都已达到蓬勃发展的阶段。正所谓“乱花渐欲迷人眼”，如何在众多的 Web 自动化测试工具和资料中选取最适合自己学习、并且能应用到实际项目中的测试套件，已经成为不少自动化测试人员犹豫不决的选择题。

许多刚入行的测试人员希望获取入门级的书籍，以此引导他们入门和上手某测试工具。本书由浅入深，从最基本的 Selenium IDE 的使用进行介绍并逐步深入，进而讲解了 Selenium 2/WebDriver 的架构、功能、使用等方面的相关知识点；除了纵向的知识深入，本书还横向涵盖了一些拓展知识点，包括 Web HTML5、嵌入式系统的自动化测试、软件持续集成、其他 Web 自动化测试工具等。希望本书不仅仅只是一本为初级测试人员准备的入门书籍，还能成为资深测试人员的案头参考资料。

为了便于读者实践，本书还提供了配套的代码包和工具包，基本囊括了本书中的示例代码、配置文件和自动化脚本文件。所有的示例代码文件、配置文件和自动化脚本文件都以章节进行了划分，并且通过第 4 章的内容配置好 WebDriver 和 Eclipse 后，导入相关的 Java 示例代码即可运行。而配套的配置文件和自动化脚本文件可根据对应章节所阐述的方法部署到合适的环境中运行。

特别感谢我的幺爷爷黄忠霖教授，我才有机会结识机械工业出版社电工电子分社张俊红副社长。同时感谢张俊红副社长，没有他的鼓励、支持和指导，我很难有勇气和毅力完成这样一本书。此外，还需要感谢张讯讯和张慧智在文字方面的审校工作。最后，要特别感谢我的妻子黄静嵘以及家人对我的理解和支持。

由于水平和时间的限制，书中难免会存在一些错误，还望见谅，并恳请读者提出宝贵意见和建议。

作者于上海

目　录

第 1 章

初识 Selenium

1.1 简　　介

Selenium 中文释义为“硒”，在化学中是一种非金属元素。可以用作光敏材料、电解锰行业催化剂等，是动物体必需、对植物有益的营养元素。

本书要介绍的 Selenium 是用于 Web 应用程序的自动化测试工具。基于 Selenium 的测试用例会直接运行在浏览器中，就像真正的用户在操作一样。其支持的浏览器范围非常广泛，包括各个平台的主流浏览器。Selenium 的主要功能包括：

1）功能性测试：创建回归测试验证软件功能和用户需求。

2）兼容性测试：测试应用程序在不同的操作系统和不同的浏览器中是否运行正常。

值得注意的是，Selenium 并不适用于进行网站后台性能方面的测试。但是结合其他第三方工具，Selenium 可被用于对网站前端性能进行适当的评估。

Selenium 源于 ThoughtWorks 公司，已经在全球范围内遍地开花。Selenium 能在测试领域成为一棵常青树，与其开源性和开放性密不可分。感谢开源先驱 Richard Stallman，为我们这个世界带来了开源运动这个先进的理念并为之奋斗一生。感谢 Jason Huggins 为我们创造了 Selenium 和现在流行的嵌入式自动化测试套件 Appium。

正如 Selenium 的化学特性一样，它作为测试领域的有益元素，已经为千千万万的测试人员带来了营养丰富的盛宴，滋养着一代又一代的测试同僚们。

1.2 自动化测试

自动化测试是把以人为驱动的测试行为转化为机器执行的一种过程。手工测试通常是在设计测试用例并通过评审之后，由测试人员根据测试用例中描述的规程一步步执行测试，得到实际结果并将其与期望结果进行比较。在此过程中，为了节省人力、时间或者硬件资源，提高测试效率，便引入了自动化测试的概念。通过自动化测试可以极大地提升回归测试、稳定性测试和兼容性测试的工作效率，在保障产品质量和持续构建等方面起到举足轻重的作用。特别是在敏捷开发模式下，自动化测试更是必不可少的步骤。

近年来，测试领域和几年前相比发生许多令人欣喜的变化：

1）敏捷开发模式的盛行掀起自动化测试的一轮热潮，测试和开发合作越来越密切。

2）测试工作的技术性越来越强，以往常见的“基于开源软件提升开发效率”的模式也被广泛应用到测试工作中。

3）组件化、软件产品线开发模式的进一步成熟，开发效率和测试效率随之进一步得到提高。

自动化测试相较于手工测试的优势在于：

1）自动化测试可以完成某些手工测试难以完成的工作，如并发测试、压力测试等。

2）自动化测试可以提高手工测试的工作效率，如执行具有多个重复步骤的测试用例。

3）自动化测试在敏捷开发过程中，可以快速验证代码修改的正确性。

4）自动化测试和手工测试相辅相成，互相促进。

实施自动化测试前需要对软件开发过程进行分析，以观察其是否适合使用自动化测试。通常需要满足以下几个条件：

1）需求变动不频繁。测试脚本的稳定性决定了自动化测试的维护成本。如果软件需求变动过于频繁，测试人员需要根据变动的需求来更新测试用例和相关的测试脚本，而脚本的维护本身就是一个代码开发的过程，需要修改和调试，必要时还需要修改自动化测试的框架。如果耗费的成本高于节省的测试成本，那么自动化测试便是失败的。如果项目中的某些模块相对稳定，而某些模块需求变动性很大，可以针对相对稳定的模块进行自动化测试，而变动较大的仍采用手工测试。

2）项目周期足够长。自动化测试需求的确定、框架的设计、测试脚本的编写和调试都需要相当长的时间来完成，这个过程本身就是一个测试软件的开发过程，需要较长的时间来完成。如果项目的周期比较短，没有足够的时间去支持这样一个过程，那么自动化测试便成为笑谈。

3）自动化测试脚本可重复使用。如果费尽心血开发了一套近乎完美的自动化测试脚本，而脚本的重复使用率很低，致使期间所耗费的成本大于所创造的经济价值，自动化测试便成为了测试人员的练手之作，而并非是真正可产生效益的测试手段。

4）手工测试无法完成的测试工作。某些测试采用手工的方式无法完成，或者需要投入大量时间与人力，此时就可以考虑引入自动化测试，如性能测试、配置测试、兼容性测试、大数据量输入测试等。

自动化测试虽然有如此多的优势，那是不是意味着自动化测试就是“包治百病”的软件“银弹”呢？有些人可能会有如下误区：

1）自动化测试是一种比人工测试更先进、更高级的测试手段。自动化测试既有自身的优点，也有其局限性。例如对于需求不明确，或者界面经常发生变动的产品就不适合使用自动化测试。自动化测试与手工测试的关系应该是相辅相成，互相弥补各自的局限性，相互促进。

2）所有的手工测试都应该被 100% 的自动化。一味片面地追求自动化率，不仅软件的质量得不到提高，而且还会让测试人员疲于奔命，投入和产出的性价比很低。有不少负面测试就只能通过手工测试的方式完成并进行验收。自动化测试不是万能的，需要根据实际情况引入并有的放矢地设定其覆盖率。

3）自动化测试能够发现大量的缺陷，它比手工测试更有效。实际情况是，自动化测试只能发现 30% 以下的软件缺陷，而手工测试反而能发现更广泛且很深层次的问题。自动化测试在回归测试时可以节省很多时间并快速验收，但这并不意味着其发现问题的能力比手工测试更强。单从发现缺陷的角度而言，自动化测试的效率低于手工测试。

4）即使一次性的软件项目也应该采用自动化测试。自动化测试的投入成本，至少要在好几个发布版本之后才能体现其价值。因此对于一次性的软件项目，应该避免采用自动化测试方案。

5）自动化测试只是测试工程师的事情，与开发人员没有关系。在软件开发过程中，首先需要考虑软件本身的可测试性。如果开发人员一开始就不把软件的可测试性考虑进来，会导致开发的软件难以测试，甚至无法实现自动化测试。

6）商业自动化测试工具更靠谱，一定要选用商业自动化测试工具。就自动化测试工具而言，测试团队应该根据自身实际情况来选择自动化测试工具。商业自动化测试工具有技术团队进行支持，遇到问题也许能尽快得到支持。但是如果有特殊的需求，这类软件往往没有自由的可定制功能。而开源自动化测试工具由于源代码都是开放的，如果团队有特殊的定制需求，可以由测试团队自行修改开源自动化测试工具来满足团队需要。

1.3　Web 自动化测试

当前绝大多数企业应用都是基于 Web 的应用系统，人们可以通过 Web 浏览器便捷地访问它们。现在炙手可热的“云计算”大潮也将这一趋势推向了高潮。很多组织和公司采用持续改进的开发模式来应对这种趋势，并采用自动化测试来适应这种高强度的持续开发的模式。相较传统桌面软件的自动化测试，Web 自动化测试指的是 Web 应用系统从用户界面层面进行的自动化测试，通过用户界面测试内部的业务逻辑。Web 自动化测试的自身特点如下：

1）Web 页面上出现的元素可能具有不确定性。

2）不同操作系统上不同 Web 浏览器之间的兼容性。

3）Web 应用的高并发性和容错性。

4）移动设备上的 Web 客户端兼容性、旋转性和各种触摸特性。

Web 自动化测试一直都不是一件容易的事情。尤其是在研发团队广泛采用 UI 框架和敏捷开发来提升交付效率的今天，Web 自动化测试变得愈发困难；未来随着 Web 组件式开发技术和云计算技术的逐渐成熟和落地，又会对 Web 自动化测试提出怎样的挑战？Web 开发过程中 UI 框架的广泛采用极大提高了开发效率和用户体验，也从技术的角度保障了敏捷开发的蓬勃发展。然而 UI 框架自动生成的海量页面源码却让原本就举步维艰的 Web 自动化测试变得雪上加霜：测试脚本的维护成本更大、回放稳定性更差、断言更困难。Web 组件式框架的采用可以让用户非常容易地卸载、替换软件的部分模块。原本是个“整体”的软件被“分解”之后可支持灵活组装。我们又该如何保障按照模块依赖关系组装出的所有软件版本都是可用的呢？云计算又能为 Web 自动化测试带来哪些机遇呢？关于 Web 自动化测试，有如此多的问题摆在我们面前，就让我们勇敢地去面对和迎接这些新的挑战吧。

1.4　Selenium 的前世今生

早在 2004 年，Jason Huggins 还供职于 ThoughtWorks 公司。他当时正在开发公司内部的时间和费用系统，该系统使用了大量的 JavaScript 代码来进行构建。虽然当时 Internet Explorer 还

占有绝对的统治地位，但是 ThoughtWorks 公司内部有不少员工在使用其他类型的浏览器，尤其是 Mozilla 基金会的浏览器。当员工使用不同的浏览器访问这个时间和费用系统时，一旦发现异常，就会提交 Bug 报告。当时的开源测试工具一般只支持占主流地位的 Internet Explorer 浏览器，或者仅仅只是模拟浏览器的行为，如 HttpUnit。购买商业工具授权的成本对于这个小型项目的有限预算而言也不太现实。

正是在这样的情况下，Jason Huggins 和他所在的团队决定自主开发一个基于浏览器并且采用 JavaScript 编程语言的测试工具。Selenium 自动化测试工具自此诞生，并且在 2004 年基于 Apache 2.0 开源协议对外发布，正式命名为 Selenium Core。

Selenium Core 的设计之初并不能绕过浏览器的“同源”规则，因此开发人员不得不将待测试的产品、Selenium Core 和测试脚本均部署到同一台服务器上来完成自动化测试工作。但在实际研发过程中，将上述各项分拆在不同的机器上运行的需求却与日俱增。为了解决这个问题，Jason Huggins 所在的研发团队编写了 HTTP 代理，以让 Selenium 截获所有的 HTTP 请求。使用 HTTP 代理的优势之一就是可以绕过浏览器的“同源”规则，让待测试的产品、Selenium Core 和测试脚本三者分散在不同的机器上。此外，这个代理的设计方式还使得采用多语言编写 Selenium 测试脚本的能力成为可能，因为测试脚本只需要关心将标准的 HTTP 请求发送到指定的 URL 即可，而 Selenium 本身并不需要关心这些 HTTP 请求是由什么程序语言编写而成。正是由于这种离散的分布方式使得测试脚本可以远程控制浏览器并执行测试用例，Selenium Remote Control 也因此得名，也可简称为 Selenium RC。Selenium RC 主要包括两个部分：一个是 Selenium RC Server；另一个是提供各种编程语言绑定的客户端驱动。

时间一晃就到了 2007 年年初，ThoughtWorks 公司发布了酝酿已久的 WebDriver 雏形。WebDriver 的设计理念是将端到端的测试与底层具体的测试工具隔离开，并采用了设计模式中常见的适配器（Adapter）模式来达到目标。

Selenium RC 与 WebDriver 最为显著的差异体现在 API 的组织上。Selenium RC 采用了基于字典方式的 API，而 WebDriver 的 API 则更加地面向对象。此外，Selenium Core（Selenium RC 的核心）基本上是 JavaScript 应用，主要运行在浏览器的安全沙箱中。而 WebDriver 则是原生绑定到浏览器中并绕开了浏览器的安全模型，其代价就是框架本身的开发投入显著增加，每个浏览器的 WebDriver 都需要单独实现。

常言道，合久必分，分久必合。历史的车轮运转到公元 2009 年 8 月，Selenium RC 和 WebDriver 项目宣布合并，也就是后来大家熟知的 Selenium WebDriver。合并以后，WebDriver 也支持了多语言绑定，克服了最初只支持 Java 绑定的缺陷。同时 Selenium WebDriver 除了支持 PC 上传统的浏览器 Chrome、Firefox、Internet Explorer，还支持嵌入式设备上基于 Webkit 内核的浏览器，包括 Android、iOS 上的浏览器等。

1.5 Selenium 1

Selenium 1 即我们所熟知的 Selenium RC。其已经为众多的测试机构和部门服役了多年并贡献卓越。Selenium RC 的典型使用方式如下：

1）测试人员基于客户端驱动所提供的 API 来编写测试用例脚本。

2）测试程序打开浏览器，此时 Selenium RC Server 绑定 Selenium Core 并自动将它嵌入

到浏览器中。Selenium Core 实际上是一系列 JavaScript 函数，它们使用浏览器内置的 JavaScript 翻译器来翻译和执行 Selense Command。

3）客户端驱动持续执行测试用例脚本并发送特定的命令到 Selenium RC Server。这些特定的命令即 Selense Command。

4）Selenium RC Server 解释 Selense Command，并触发 Selenium Core 执行对应的 JavaScript 代码来完成相应操作。

5）浏览器上所有的请求和响应都通过 Selenium RC 的 HTTP 代理与实际的 Web 应用服务器进行交互，并且 Selenium RC 一旦收到响应就将页面传递给浏览器。但它会篡改源，使得页面看上去好像来自于与 Selenium Core 同源的服务器（这样 Selenium Core 就遵循了同源规则）。

6）浏览器接收到 Web 页面后，便在框架或者窗口中展示页面。

但毕竟事物都有更新换代的阶段，Selenium 1 也不例外。正所谓老树发新芽，Selenium 2 就是从 Selenium 1 这棵历史老树上发出的新芽。截至目前，Selenium 1 还继续被维护并提供一些 Selenium 2 尚不支持的功能，如某些编程语言的支持和某些浏览器的支持。因此，为了避免大量的基于 Selenium 1 的测试架构和代码被遗弃，Selenium 团队提供了一种折中的方式来让基于 Selenium 1 的测试架构和代码能继续发挥余热。

某些使用 Selenium 多年的公司和机构，希望能够继续维护基于 Selenium RC 的测试集合，并继续添加新的测试代码；但是还有一部分使用 Selenium 多年的公司和机构，他们希望能将基于 Selenium RC 的测试代码迁移到 WebDriver 上来，让那些老旧但成熟的代码和测试集合重新焕发新生。本书在 5.5 节展示了如何从 Selenium RC 迁移已有代码到 WebDriver 的解决方案。

1.6　Selenium 2

Selenium 2 的主要新特性就是将 WebDriver API 集成进 Selenium RC，从而解决 Selenium 1 所面临的一系列局限性问题。WebDriver 的创建者 Simon Stewart 曾在 WebDriver 和 Selenium 社区中回答了合并的原因：

“WebDriver 和 Selenium 为什么会合并？究其根本，是 WebDriver 和 Selenium 可以互相弥补对方的缺点。而且，Selenium 开源项目的资助者们也希望两者可以合并。我认为这种合并方式就是用户可以获取的最好的组合架构方式。”

WebDriver 与 Selenium RC 合并的结晶就是 Selenium 2。其 API 的设计非常精巧，既易于理解又易于拓展。Selenium 2 不与任何的测试框架绑定，这样便于与其他测试工具进行集成，如 JUnit 或 TestNG 等。

WebDriver 的实现和具体的浏览器相关，包括 HtmlUnit Driver、Firefox Driver、Chrome Driver、Internet Explorer Driver 等。

1）HtmlUnit Driver：HtmlUnit Driver 是目前运行速度最快和最轻量级的 WebDriver 实现。正如其名字所体现的，HtmlUnit Driver 基于 HtmlUnit，优点是纯 Java 实现，所以容易跨平台使用。

2）Firefox Driver：Firefox Driver 是最容易配置和使用的 WebDriver，因为所有的准备工作都伴随 Java 语言绑定的客户端被打包在一起。只要下载 WebDriver Java Client Driver 就能

够使用 Firefox WebDriver。

3）Chrome Driver：Chrome Driver 是针对 Google Chrome 浏览器开发的 WebDriver，因此其跨平台性也是非常的优异。

4）Internet Explorer Driver：Internet Explorer Driver 只能运行在 Windows 操作系统上，相较于 Firefox Driver 和 Chrome Driver，其运行速度略显缓慢。

Selenium 2 相较于 Selenium 1 还有一个重要变化，用户可以通过 WebDriver 来测试手机应用，无论在模拟器上还是真实设备上。这是在 Selenium 1 时代无法实现的功能。

1.7 Selenium 3

在本书撰写之际，Selenium 3 也已经由 Selenium 团队提上议程，并正在紧锣密鼓地开发中。据悉，Selenium 3 会移除原有的 Selenium Core 的实现部分，并且 Selenium RC 的 API 也将被去掉。其他一些变化包括但不限于以下内容：

1）所有的 Driver 在消息响应中统一使用状态字符串，而不是状态码，这样更容易辨识。

2）为基于 Html 格式的测试集合提供一个新的运行器，特别是针对 Selenium IDE 所导出的 Html 测试集合。其原理是基于 WebDriver-backed 的 RC 实现。

3）将 Selenium 3 的分支和包含 Selenium RC 的代码分支分开进行编译打包。

4）在 WebDriver. quit（）方法之后如果还使用 WebDriver，则会报错：IllegalStateException。

5）对 WebDriver. quit（）方法的多次调用也是允许的。

6）重构 WebDriver 的构造函数。

7）架构迁移到 Netty 或者 Webbit 服务器上。

1.8 Selenium IDE

Selenium IDE 的优点如下：

1）录制功能快捷方便，上手快。

2）代码转换功能易用，容易生成其他编程语言的测试用例代码。

3）支持跨域。

4）不依赖 Java 运行时环境。

Selenium IDE 的缺点如下：

1）录制回复方式的稳定性和可靠性有限。

2）只支持 Mozilla Firefox。

3）只支持 Selense Command 语言，虽然可以导出成其他编程语言的测试用例。

4）对于复杂的页面逻辑其处理能力有限。

1.9 Selenium Grid

Selenium Grid 运用多个机器同时并列运行，目的在于加快测试用例运行的速度，从而减

少测试运行的总时间。对于大型测试套件和需要处理海量数据验证的测试套件，Selenium Grid 毫无疑问可以节约大量时间。Selenium Grid 的另一个优势在于可以通过节省测试时间而更快地将测试结果返还给开发人员。越来越多的软件开发团队运用敏捷开发方式，他们希望在最短时间内获得测试人员的测试结果而不是在漫漫长夜中等待着测试通过。

Selenium Grid 还可被用于在多种运行环境中进行测试，即并行测试多种浏览器。当测试套件运行起来时，Selenium Grid 会接收到每个测试用例及其对应浏览器的组合信息，并分配每个测试用例去测试其对应浏览器。

除此之外，对于相同类型和版本的浏览器来建立测试矩阵也是可行的。Selenium Grid 的使用相当灵活，以上所列举的并行测试浏览器的例子也可结合起来使用，用于测试每种类型和版本的浏览器的多个实例。

Selenium Grid 包含一个 Hub 和至少一个 Node。Hub 会接收到即将被执行的测试用例及其相关信息，即测试用例将在哪种浏览器和操作系统上运行。Hub 将记录每个“注册过”的 Node 的配置信息，并能通过这些信息自动选择可用的且符合浏览器与平台搭配要求的 Node。Node 被选中后，测试用例所调用的 Selenium 命令就会被发送至 Hub，Hub 再将这些命令发送到指定给该测试用例的 Node。随即 Node 开始启动浏览器，并执行这些 Selenium 命令对指定的 Web 程序或 Native 程序进行测试。

1.10　Selenium 与嵌入式

随着 Android、iOS、Raspberry Pi、Firefox OS、Ubuntu Phone OS、Sailfish OS 这一系列的嵌入式系统平台和操作系统的崛起，人们的手持设备和各式各样的嵌入式设备正变得越来越多样化。同一个网站，如果需要在这么多种不同平台的设备上都能够通过 Web 方式正常访问，其兼容性测试极具挑战性。而 Selenium 正在成为这场百家争鸣没有硝烟的战争中极具竞争力的测试工具和手段。

嵌入式设备中，尤其是需要在手机设备上测试 Web 应用程序，一直以来都是采用手工测试的方式。而 Selenium WebDriver 赋予了我们采用自动化方式来测试嵌入式设备上基于浏览器应用的能力，可谓一代神器。WebDriver 使得测试人员能够方便地编写自动化测试代码以确保自己的网站除了在 PC 上，还能在手机上通过浏览器正常访问。

WebDriver 为手机设备提供了专门针对触摸屏的 API，因此可以让测试程序模拟人通过手指操作触摸屏的真实动作，如单击、触划、滚动、长按等有别于普通 PC 上用鼠标操作的特殊行为。此外，还可以让测试程序对屏幕上的内容进行旋转，以及与 HTML5 的特性进行交互，如本地存储、会话存储和应用程序缓存等。

本书会基于 Android、iOS 和 Raspberry Pi 这几个平台为例来讲解 Selenium 在嵌入式平台上的使用。

1.11　Selenium 与云计算

迎着云计算普及的春风，并配合 Selenium Grid，基于 Selenium 来完成云端的测试已经变得可行。除了可以测试传统桌面系统上的 Web 应用，还可以将嵌入式设备放到云计算平台

中来完成自动化测试工作。其优势在于资源的合理调配，并且能够尽可能多地解决测试多种不同操作系统类型或者不同尺寸屏幕的兼容性问题。特别是现在热门的 Android 平台和 iOS 平台，它们都有相应的模拟器程序，使得在云中进行测试变得可行。

基于 Selenium 且比较流行的云测试平台当属 Sauce Labs，它是一个提供自动化功能测试的云测试服务公司。而其创始者兼首席技术官就是 Selenium 的创始人 Jason Huggins，可见 Sauce Labs 的技术实力非同一般。虽然 Selenium 在 Web 自动化测试方面有得天独厚的优势——兼容性测试，可以涵盖多个操作系统上的不同版本的多种浏览器，但要完成如此庞大规模的兼容性测试，就需要保留与兼容性矩阵所含格子一样多的虚拟机。这对于小团队而言，维护如此庞大规模的虚拟机并不现实。Sauce Labs 正是基于这样的需求应运而生。对于初创团队而言，他们可以在云端去完成兼容性矩阵的测试而不需要自己购买大量的硬件并自行维护成千上万台测试虚拟机。

为了应对嵌入式系统的大量普及，尤其是 Android 和 iOS，Jason Huggins 又再一次开发了针对嵌入式系统且基于 WebDriver 的自动化测试框架 Appium，并且开源托管在 GitHub 上。本书也会针对 Appium 的使用展示 WebDriver 的魅力所在。Appium 除了可以支持 Web 应用，还可以支持原生的 app 程序和 Hybrid（即 Web 和原生 app 的混搭模式）app 的测试。这也是 Sauce Labs 针对嵌入式设备所提供的云端测试解决方案。

1.12 小　　结

本章从自动化测试的特点、Web 自动化的独有特性娓娓道来，引出了主角 Selenium。从 Selenium 的前身介绍到 Selenium 的现状，让大家对 Selenium 有一个初步的认识。虽然其发展历史算不上惊心动魄，但也经历了跌宕起伏的过程。作为测试领域的一朵奇葩，Selenium 已然功成名就，并将继续引领时代潮流。