计算机系统应用 ISSN 1003-3254, CODEN CSAOBN
Computer Systems & Applications,2018,27(11):51−56 [doi: 10.15888/j.cnki.csa.006601]

E-mail: csa@iscas.ac.cn
http://www.c-s-a.org.cn
Tel: +86-10-62661041

基于 Selenium 的 Web 自动化测试解释器①

沈大框[1], 黄永锋[1], 罗保国[2]

[1](东华大学 计算机科学与技术学院, 上海 201620)
[2](中国科学院 软件研究所, 北京 100190)
通讯作者: 黄永锋, E-mail: 741239226@qq.com

摘　要: Selenium 自动化测试工具在测试领域已经被广泛应用. 使用时它需要依赖其它编译器或解释器才能执行测试. 一方面让测试人员需要熟练的编程技术才能使用, 另一方面让测试开发难度过高. 为了降低自动化测试门槛, 针对 Web 自动化测试设计了一种基于 Selenium 的解释器. 首先深入剖析 Selenium 的测试原理, 依据它测试接口的使用状况对主要的测试类构建依赖关系, 然后根据 Web 程序的测试需求构建解释器的各个模块. 参照编程人员的编码习惯和参考 Python、JQuery 等语法对每个模块的语法详细设计. 最后设计一个具体的测试用例对解释器进行评价. 实验结果表明, 该解释器克服了使用 Selenium 门槛高的缺点, 让编写的测试脚本更加简短整洁, 测试性能也有明显的提高.

关键词: 解释器; Selenium; Web; 自动化测试

引用格式: 沈大框,黄永锋,罗保国.基于 Selenium 的 Web 自动化测试解释器.计算机系统应用,2018,27(11):51–56. http://www.c-s-a.org.cn/1003-3254/6601.html

Web Automatic Test Inerpreter Based on Selenium

SHEN Da-Kuang[1], HUANG Yong-Feng[1], LUO Bao-Guo[2]

[1](Computer Science and Technology Academy, Donghua University, Shanghai 201620, China)
[2](Institute of Software, Chinese Academy of Sciences, Beijing 100190, China)

Abstract: Selenium automated testing tools have been widely used in the field of testing. In the scenario, they need to rely on other compilers or interpreters to perform tests. On the one hand, testers need sophisticated programming skills to use the tools, and on the other hand, test development is too difficult. In order to reduce the automation test threshold, an interpreter based on Selenium is designed for Web automated testing. Firstly, it analyzes the testing principle of Selenium, and builds dependencies on the main test classes based on the usage of the test interface, and then constructs the modules of the interpreter according to the testing requirements of the Web program. For detailed design of each module's syntax, we referred to the programmer's coding conventions, Python, JQuery, and other syntax. Finally, a specific test case is designed to evaluate the interpreter. The experimental results show that the interpreter overcomes the disadvantages of using Selenium's high threshold, makes the written test script more concise and cleaner, and the test performance has also been significantly improved.

Key words: interpreter; Selenium; Web; automated testing

21 世纪以来, 随着通信技术的飞速发展, 互联网行业市场份额所占比重越来越高. 传统落后而且效率低下的手工软件测试已经不能满足市场的需求, Web 程序自动化测试技术越来越受人们的重视. 这几年, 各种

① 收稿时间: 2018-03-19; 修改时间: 2018-04-11; 采用时间: 2018-04-20; csa 在线出版时间: 2018-10-24

Web 自动化测试工具被相继推广发行, 已经得到广泛的应用. 但这些测试工具主要研究浏览器中 Javascript 的执行机制和它们对不同浏览器的兼容性.以此在原来的基础上丰富库函数, 从而增加测试的功能和提高测试工具的稳定性. 虽然这些测试工具功能全面, 能够让繁琐的任务自动化, 节省人力资源, 降低测试成本. 但是近年的研究只是对测试工具的功能积木式的累加, 并没有实质的创新与突破. 大多数测试工具依然需要借助传统的编译器才能执行测试. 这需要使用者具备一定的编程能力，而且测试脚本开发成本较高. 所以当下 Web 软件自动化工具的发展依然让测试具有很高的门槛[1–3].

目前, 较为常见的 Web 自动化测试工具集主要包括 Ranorex、QTP、WinRunner、Selenium、Watir、Sahi 等. 前三者是商用的 Web 程序自动化测试工具, 功能强大但价格昂贵. 后三者是开源的 Web 程序自动化测试工具.虽然在使用上比不上商用所能提供便利的可视化效果和舒适的使用方式, 但是已经能够满足测试人员的使用需求.而且 Web 应用不同于传统的 Window 应用, Web 产品具有产品数量大、产品变化快、对硬件环境和网络敏感等特点.所以人们会去优先选用像 Selenium 这种兼容性较强的测试工具[4].

事实上, Selenium 测试工具是历史发展最为悠久、测试功能最为全面、脚本支持最多的测试框架. Selenium 可以部署在 Windows、Linux 和 macOS 平台上. 它支持脚本的录制与回放功能, 通过 Selenium IDE 来录制测试脚本, 并提供 UI 编辑中间的任何步骤. 录制保存后可以后可以使用 Selenium RC 或者 Selenium WebDriver 来回放录制内容. 录制的脚本被记录在 Selenese 中, 这是一种针对测试 Web 应用程序的 Selenium 命令集. 但是它的命令集不具备编程语言的特性, 只能处理简单的测试单元. 实际应用中, 人们很少使用 Selenese 来编写测试用例, 而是使用 Java, Python, Ruby 等[5].

所以本文选择以 Selenium 为切入点, 从研究 Selenium 的测试机制出发, 设计一个专用的 Web 自动化解释器. 该解释器能实现测试脚本与测试数据的分离、自动生成测试报告和支持半自动化测试，从而提高 Web 自动化测试的柔性和实用性. 通过分析 Selenium WebDriver API 中的测试功能来构建测试的类的依赖关系, 把 Selenium 测试功能划分成为一个层次分明和分工明确的结构体. 然后根据 Web 应用程序自动化测试的流程和 Selenium 的测试功能的划分对该解释器构建模块. 从测试的需求出发, 使能初步设计出一个以 Selenium 为后端, 更加符合人们使用习惯的脚本解释器. 最终期望测试人员使用该解释器能够降低自动化测试的门槛和提高测试开发的效率.

1 解释器的模块构建

1.1 Selenium 的工作原理

Selenium 升级到 2.0 以后, 它重点发展了 Selenium WebDriver. 由之前的必须先启动 Selenium Server 使其与各个浏览器进行通信交互发展到直接调用浏览器本身的 WebDrive 驱动进行与浏览器的会话, 绕过 JavaScript 沙箱, 提高了 Selenium 的测试效率和对各个浏览器的兼容性. 图 1 是 Selenium 的测试原理. 新一代的 Selenium 做 Web 软件自动化测试, 本质上是调用 Selenium API 关于 WebDriver 的 API. WebDriver 是嵌入在浏览器中的驱动, 能够直接调用浏览器的原生 JavaScript 来控制浏览器, 相当于模拟人类对浏览器的操作. 如点击某个位置、给某个输入框输入键值等. 本章从 Selenium WebDriver 入手, 着重分析 WebDriver API 接口的特点, 先对 Selenium 测试进行模块拆解, 然后对解释器进行模块建立[6].

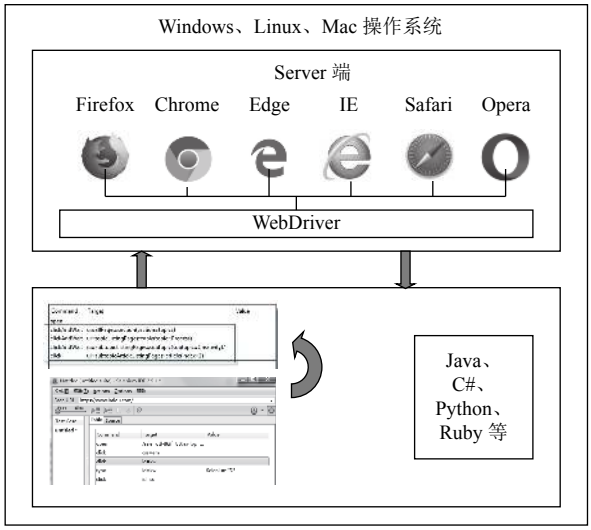

图 1 Selenium 的测试原理

1.2 构建解释器模块

根据 SeleniumAPI 文档, Selenium 对浏览器的控制操作主要放在了 RemoteWebDriver 、RemoteWebElement、RemoteWindow 和 Actions 这四个类中. 前三

个类分别表示浏览器、控件、和窗口层次的概念. 浏览器位于最高层次, RemoteWebDriver 类中的方法能够对浏览器和窗口的特性进行控制, 也可以对控件进行查找并创建控件对象. RemoteWebElement 类里包含控件的属性值和控件的动作, 它自己也提供对控件的查找与创建. RemoteWindow 类中包含窗口的属性值和窗口的动作, 但是它必须通过浏览器来创建并控制窗口的动作. 图 2 是三个测试类之间的依赖关系. Actions 类对上述三个类测试功能的扩展, 一方面它提供了模拟鼠标和键盘的一些动作以及组合键使用. 另一方面 Actions 类既可以完成控件单一的动作, 也可以把多个动作合成一个具有测试顺序的组合动作. 综上所述, 根据这几个类提供的测试方法对其进行分类, 可以把 Selenium 的测试分为 5 个模块, 分别为: 获取控件属性值, 获取窗口属性值、控件定位、执行控件动作、执行窗口动作, 这也是对应解释器的初始模块.

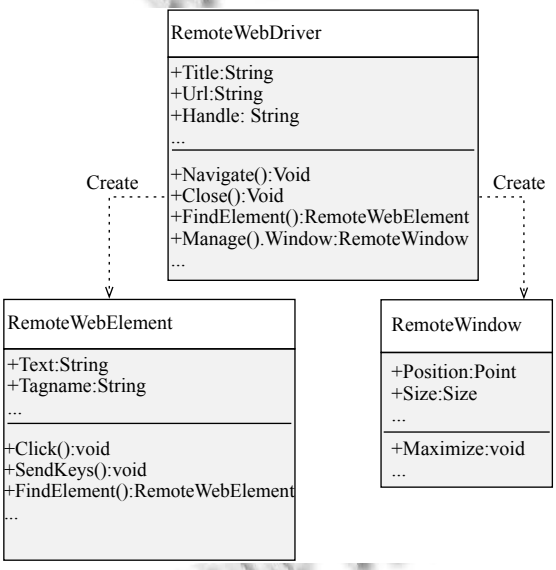

图 2 测试类之间的依赖关系图

在 Web 自动化测试中, 除了对浏览器的窗口与页面进行模拟人类的操作. 还需要去检查页面的属性值是否和预设的结果一样, 以及触发某一动作后, 检查一下该动作是否触发成功. 所以除了根据 Selenium 提供的 API 把解释器构建成上述 5 个模块以外,还需要根据实际的测试需求增加数据的验证. 上述 5 个模块的测试都需要进行数据验证, 它们共同指向数据验证模块. 如图 3 是解释器的模块结构图. SeleniumAPI 中没有对数据验证提供相关的接口, 需要测试开发人员使用它所依赖的程序语言人工编程实现. 这也就给测试带来很多额外的开发成本. 从测试需求出发, 在解释器设计阶段给解释器增加数据验证与检查模块.这样能增加解释器的可扩展性, 降低测试脚本、测试数据和测试报告之间的耦合性.

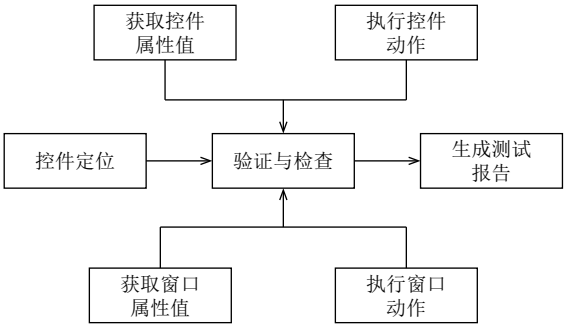

图 3 解释器的模块结构图

2 解释器语法的研究

Selenium 使用 Java、Python、Ruby 等编程语言编写的测试脚本, 一方面开发难度较高, 另一方面可读性较差. 本文研究的基于 Selenium 的解释器是专门针对测试的, 避免了像 Selenese 那样仅提供测试指令集, 无法进行具有控制逻辑的测试. 解释器的基础的语法参照 Python 和 Octave 等解释器型的语法规则, 删除了与测试无关的部分, 同时也根据测试的特点进行改进和增添新的测试使用方式. 该解释器仍包含一般编程语言具有的基本模块如变量与常量、表达式、算术运算符、关系运算符、逻辑运算符、条件语句、循环语句、宏定义、内部函数与外部函数、输入与输出等. 这使得解释器能完成具有复杂逻辑的测试需求, 同时简化整个编译器的语法, 降低使用难度.

2.1 窗口属性值和控件元素属性值获取模块

在测试的过程中, 经常有获取浏览器窗口与控件元素属性值的需求. 如获取这些属性值来验证它们的正确性或者暂时保存下来给后续的测试步骤使用. 窗口属性值主要包括窗口的 URL、窗口的标题、窗口的 Handle、窗口的 Body 内容等. 控件元素的属性主要包括 XY(横纵) 坐标、元素的标签名、元素的属性值、和元素的 css 属性值等. 如果用 Java 或 Python 的 Selenium 来获取上述属性值, 则需先创建窗口或控件的对象, 然后调用对象里的方法来实现, 这需要测试人

员有面向对象的编程思想，并且开发步骤繁琐. 参考 Javascript 的框架 JQuery 获取元素对象的方法, Selenium 解释器可以使用$(string_name) 来获取窗口的属性值,其中括号里的变量为窗口属性的名称. 用$(*var)、$(.class)、$(:attr) 等获取控件元素的属性值, 其中 var 表示控件元素特定简写值如 X、Y 分别表示坐标; class 表示控件元素的 class 属性值; attr 表示控件元素的 Attribute 属性值.

2.2 控件定位模块

Selenium 在获取当前页面的控件元素时, 通过筛选方法来得到控件对象. 使用时需要调用 FindElement (string type, string value) 函数, 它传入了两个参数: 定位方式和定位的属性值. 解释器依照上述使用方式, 把定位方式定义到脚本的语法层次. 根据 Web 程序测试的特性, Selenium 提供了 8 种控件查找方式, 分别是 Xpath、LinkText、Id、Tagname、ClassName、Name、PartialLinkText、CssSelector. 为了提高控件抓取的效率, 和抓取控件的多样性、以及符合编程人员的使用习惯, 解释器参考 JQuery 和 W3C 对网页的定义的标准, 用@/xpath、@&LinkText、@#id、@<tagname、@.ClassName、@^name、@~PartialLinkText、@*CssSelector 来表示控件的定位.

2.3 窗口与控件的动作执行模块

执行动作包含两类动作: 一是当前控件的动作, 二是浏览器的窗口的动作. 控件的动作包括点击、文本框传入键值、移动鼠标到指定位置等. 浏览器的窗口动作包括打开某一版本的浏览器、切换窗口、导航窗口到指定链接、处理弹框等. 执行动作时与函数的调用一样, 把动作执行看做内部函数的调用. 直接使用函数名称并传入相应的参数来实现, 比如 OpenWindow (string url) 表示当前浏览器打开一个指定 URL 的新窗口.

2.4 验证与检查模块

Web 软件自动化测试要求在测试某个功能后, 需要检查该功能的正确性. 即验证当前页面或者经过某次执行动作后页面的一些属性值与给定的参数值是否相等. 验证检查需求有很多, 如验证窗口的 URL、控件元素的标签名、某个 Select 控件是否被选中等. 使用 Java 或 Python 的 Selenium 做 Web 自动化测试验证, 需要根据根据测试报告的格式定义数据结构, 然后手工编程实现判断验证内容的正确性, 最后把验证结果写入测试报告对象中，它们均不能自动生成测试报告. 在解释器层面先引入测试步骤的概念, 并在它内部预先定义好通用的测试报告格式, 这些测试报告可以用外部配置文件去修改, 以便兼容其他平台或公司内部测试报告格式. 使用验证表达式!($(url) == url) 能够直接完成上述验证测试的需求, 并把验证结果写入到测试报告[7].

2.5 半自动化测试

解释器为了适应市场的需求, 率先提出“半自动化测试”的思路, 在适合自动化执行的地方嵌入机器代码, 在不适合于自动化的地方仍旧由人工进行. 脚本中嵌入/1: message, 解释器遇到这行代码会弹出提示信息, 提示信息即脚本中的 message. 测试人员可以先手工测试一段难以实现自动化的步骤, 并填写是手工测试的完成情况, 然后点击弹框的确定按钮,解释器会继续执行后续的测试. 这种设计方案中, 提供人工操作与机器代码执行可随时切换的功能, 从而让测试团队在安排自动化测试的任务时更加灵活.

2.6 解释器执行过程

解释器 (英语: Interpreter), 是一种计算机程序, 能够把高级编程语言逐行解释运行. 在本文设计的解释器中, 它先调用词法分析器取得 Token, 每遇到一个终止符就调用语法分析器生成语法树. 然后调用已经封装好的基于 C#的 Selenium 特征库, 执行该语句. 浏览器 Web 页面会接收到解释器发来的命令, 处理相应的动作. 总之解释器的目标是努力向前按步执行下去, 直到脚本程序执行结束[8–10]. 图 4 是自动化测试解释器流程图.

3 解释器的测试与评价

3.1 解释器的测试

为了证明该解释器的功能性、测试脚本的简易性和可阅读性. 专门设计一个登陆验证的测试用例, 然后使用本文设计的解释器对测试用例进行脚本开发和测试执行. 登陆验证通过对登录名和密码的输入框分别键值为空、键值错误信息、键值正确信息, 一共 9 种情况, 然后点击登录，验证登录结果与预先设定的结果是否一致. 这 9 组数据全部测试通过, 登录验证功能才能判断为通过[11]. 登录验证的测试脚本和测试报告如下:

```
Import E:/login.csv as $var//导入数据源
Chrome(56) //启动 chrome 56 版本
```

```
ImplicitlyWait(60) //设置全局等待时间为 60 s
Navigate(http://localhost:8088/biz/login/toLogin.do)
//导航到指定链接
while ($var) //开始循环
  @#admin_code //按照 id=admin_code 查找控件
  SetElementText($user)//输入框先清空在传值
  @#password
  SetElementText($pw)
  @#login_action
  clickElement() //鼠标左击控件
  @#message
  !($(*N)==$text) //验证登录并记录测试报告
endwhile //结束循环
```

Checkpoint	Fial Test Cycle	Fail Cycle	Fial Rate
空的用户名，空的密码	1	0	0%
空的用户名，错误的密码	1	0	0%
空的用户名，正确的密码	1	0	0%
错误的用户名，空的密码	1	0	0%
错误的用户名，错误的密码	1	0	0%
错误的用户名，正确的密码	1	0	0%
正确的用户名，空的密码	1	0	0%
正确的用户名，错误的密码	1	1	100%
正确的用户名，正确的密码	1	0	0%

从上述测试脚本看出, 几乎每行测试脚本都能看到测试的影子, 脚本具有极强的可读性. 十几行代码就实现一个完整的登录验证测试, 并能自动生成测试报告. 说明该解释器语法的设计很适合进行 Web 自动化测试, 测试开发很容易实现.

3.2 解释器的评价

为了能综合评价该解释器的实用性, 使用本文设计的解释器与 Selenium 的 Java、Python 和 Ruby 版对上节登录验证的测试用例在 9 组登录数据上进行了脚本开发, 并执行测试. 脚本开发过程中对开发难度做和测试数据与测试脚本分离难度做了评估. 测试过程中, 采集了测试用例的运行时间. 最后综合从开发难度、是否生成测试报告、是否支持半自动化测试、测试数据与测试脚本分离难度和运行时间 5 个维度综合评价解释器. 其中运行时间能有效评估解释器的执行效率. 表 1 是 Selenium 测试的比较.

由此可见, 本文设计的解释器更加容易实现测试开发和测试数据和测试脚本的分离. 并且解释器增加了自动生成测试报告和支持半自动化测试的功能. 一定程度上增强了该解释器的实用性. 由表中的运行时间数据, 该解释器在同一个测试用例上的运行时间少于 Selenium 的其他语言,得出该解释器的执行效率也有所提升. 总体来说本文设计的解释器在 Web 自动化测试方面具有更高的实用性和优越性.

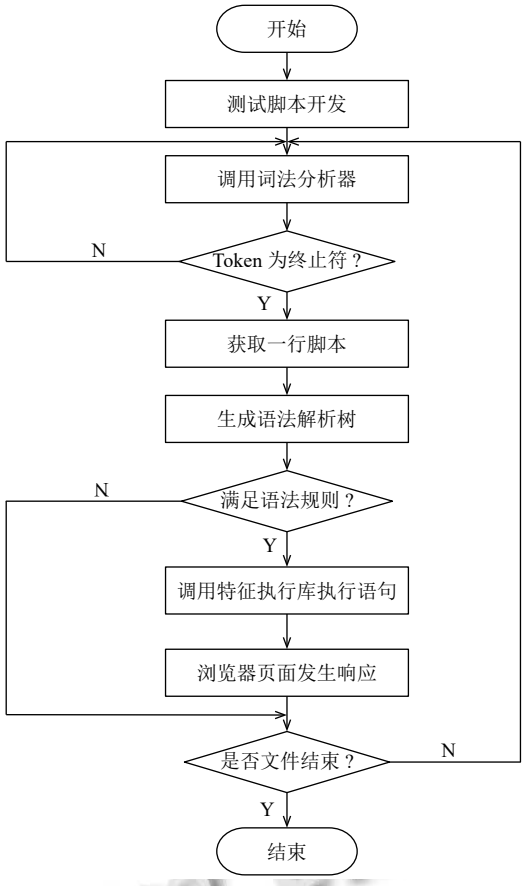

图 4 自动化测试解释器流程图

表 1 Selenium 测试的比较

程序语言	开发难度	生成测试报告	半自动化测试	分离难度	运行时间 (s)
Self	简单	是	支持	简单	139.83
Java	困难	否	不支持	困难	147.81
Python	一般	否	不支持	一般	171.62
Ruby	一般	否	不支持	困难	176.30

4 总结与展望

本文从解释器的研究出发优化了 Selenium 自动化测试, 初步设计了一个专门用于 Web 自动化测试的解释器. 该解释器的语法从测试角度出发, 使得测试人员更加容易学习和使用. 从实验结果可以看出基于 Selenium 的解释器可以降低测试人员的使用门槛, 提升测试的效率, 具有更高的使用价值. 但当前解释器对 Web 测试的语法还不够精炼和完善, 解释器的稳定性

和执行效率还有很大的提升空间. 未来将会在这两个方向去做更深入的研究.

参考文献

1 马春燕, 朱怡安, 陆伟. Web 服务自动化测试技术. 计算机科学, 2012, 39(2): 162–169. [doi: 10.3969/j.issn.1002-137X.2012.02.038]

2 Satheesh A, Singh M. Comparative study of open source automated Web testing tools: Selenium and Sahi. Indian Journal of Science & Technology, 2017, 10(13): 1–9.

3 黄华林. 使用 Selenium 进行 Web 应用自动化测试的研究. 电脑开发与应用, 2012, 25(4): 54–56. [doi: 10.3969/j.issn.1003-5850.2012.04.019]

4 沈佳宇, 喻擎苍. 基于 Watir 框架的 Web 自动化测试. 电脑编程技巧与维护, 2012, (14): 95–96, 158. [doi: 10.3969/j.issn.1006-4052.2012.14.039]

5 Qi XF, Wang ZY, Mao JQ, *et al*. Automated testing of web applications using combinatorial strategies. Journal of Computer Science and Technology, 2017, 32(1): 199–210. [doi: 10.1007/s11390-017-1699-x]

6 石敏. 面向 Web 网页的自动化测试技术研究[硕士学位论文]. 上海: 东华大学, 2014.

7 张竞帆. 基于 Selenium 的一种 Web 自动化测试系统的设计与实现[硕士学位论文]. 北京: 北京交通大学, 2017.

8 Barany G. Python interpreter performance deconstructed. Proceedings of the Workshop on Dynamic Languages and Applications. Edinburgh, United Kingdom. 2014. 1–9.

9 Cao L, Dong KJ, Yuan BW. Design and implementation of a test automation platform for Web applications based on selenium. e-Science Technology & Application, 2014, 5(6): 44–52.

10 张军林, 阳富民, 胡贯荣. JavaScript 语言解释器的设计与实现. 计算机工程与应用, 2003, 39(30): 124–125. [doi: 10.3321/j.issn:1002-8331.2003.30.039]

11 尹帮治. 一种新的网站用户登录验证方案. 微型电脑应用, 2008, 24(10): 13–15. [doi: 10.3969/j.issn.1007-757X.2008.10.005]