目录

[概述 2](#_Toc299106170)

[背景 2](#_Toc299106171)

[应用领域与使用对象 2](#_Toc299106172)

[参考资料 2](#_Toc299106173)

[术语与缩写解释 2](#_Toc299106174)

[系统综述 3](#_Toc299106175)

[系统结构 3](#_Toc299106176)

[系统功能 3](#_Toc299106177)

[运行环境 3](#_Toc299106178)

[硬件设备要求 3](#_Toc299106179)

[软件需求 3](#_Toc299106180)

[系统操作说明 3](#_Toc299106181)

[系统安装与初始化 3](#_Toc299106182)

[系统目录结构 4](#_Toc299106183)

[操作步骤 4](#_Toc299106184)

[XML文件详细说明 5](#_Toc299106185)

[webSite 5](#_Toc299106186)

[testCase 6](#_Toc299106187)

[数据的初始化 6](#_Toc299106188)

[Html标签 7](#_Toc299106189)

[Function标签 8](#_Toc299106190)

[Verify标签 9](#_Toc299106191)

[Assert标签 9](#_Toc299106192)

[Choose标签 9](#_Toc299106193)

[config 10](#_Toc299106194)

MYDLINK AUTOMATION

使用说明书

# 概述

## 背景

selenium只提供了一些基本功能的API，不能完全满足对网站系统的自动化测试，同时还需要测试报告的生成，软件就应运而生。

本手册的介绍了软件的方方面面：软件的流程，软件的使用，软件的操作及其一些简单的例子，目的在于更好的了解软件，更好的使用软件。

## 应用领域与使用对象

适用于一些网站系统的自动化测试。

## 参考资料

|  |  |
| --- | --- |
| 网址 | 简介 |
| <http://www.w3school.com.cn/xpath/> | **XPath 是一门在 XML 文档中查找信息的语言。XPath 可用来在 XML 文档中对元素和属性进行遍历。** |
| <http://wenku.baidu.com/view/7fb3512558fb770bf78a55fa.html> | **Selenium介绍** |
|  |  |

## 术语与缩写解释

|  |  |
| --- | --- |
| 缩写&术语 | 解释 |
| selenium | Selenium也是一个用于Web应用程序测试的工具。 |
| API | 是一些预先定义的函数，目的是提供应用程序与开发人员基于某软件或硬件的以访问一组例程的能力，而又无需访问源码，或理解内部工作机制的细节。 |
|  |  |

# 系统综述

## 系统结构

结合系统所具有的功能包括输入、处理和输出提供该软件的总体结构图表。

## 系统功能

* 检查给出的testcase，config和其他的xml文件的完整和正确性，如果不正确给与相应的提示。
* 自动化的跑完整的testcase，不需要人为的干预。
* 系统最后会输出报告。在报告中有错误的的操作步骤会有截图的链接，点击链接可以看到错误操作的截图。

# 运行环境

## 硬件设备要求

* 内存：通常应该确保操作系统占有的内存，软件最大占用内存一般为20M。
* 硬盘：软件的安装需要占用50M的空间，但是随着软件的运行次数的增加，产生的报告文件会随之增加，需要考虑报告文件的占用空间。
* CPU：通常家用计算机的CPU能正常运行，可根据实际CPU的使用率来决定是否增加投入。
* 计算机台数：要具体根据测试的平台要求和浏览器要求来决定使用的数量。

## 软件需求

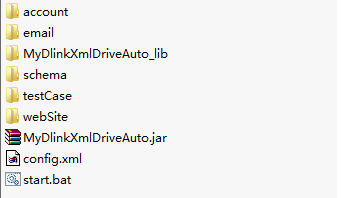
* 操作系统：windows系列操作系统，Linux/Unix操作系统等。
* 所需软件：firefox，ie，googlechrome，java1.5或者以上和XML编辑软件。

# 系统操作说明

## 系统安装与初始化

* 安装文件名字：MyDlinkXmlDriveAuto.rar
* 安装文件大小：约为25M（版本：3.0）
* 安装方法：解压到指定的目录（目录中不应存在非英文类字符和空格字符）

## 系统目录结构



## 操作步骤

* 添加网站页面数据
  + 在系统的目录中会有个文件夹叫“webSite”，里边存放的是需要测试网站页面上元素的xpath，javascript，图片地址，颜色，字体大小，文本内容等等。在自动化系统工具初始化的时候会加载这些内容。为自动化工具提供一些必要的数据。
* 编写测试用例
  + 在系统的目录中会有个文件夹叫“testCase”，这个文件夹里边放的是根据测试用例所编写的xml文件，它是测试用例操作的集合。它会告诉自动化测试工具一步一步的操作。每次测试用例版本发生变化的时候，就需要修改下对应的xml文件来保证自动化测试工具是按照测试用例的步骤来执行的。
* 编辑配置文件
  + 自动化工具运行的时候需要一些参数，运行测试用例主机的IP地址，
  + 测试的网站系统的地址，浏览器的类型，账户库，邮件库，网站数据来源，测试用例来源等等。
* 启动selenium服务器
* 启动自动化系统工具
  + 命令提示符方法
    - 用cd命令打开系统所在的目录。
    - 输入java -Dfile.encoding=UTF-8 -jar MyDlinkXmlDriveAuto.jar config.xml
    - config.xml可以换成其他的配置文件。
    - 回车即能启动自动化工具
  + bat文件方法
    - 记事本编辑以下内容java -Dfile.encoding=UTF-8 -jar MyDlinkXmlDriveAuto.jar config.xml
    - config.xml可以换成其他的配置文件。
    - 另存它为bat格式的文件到系统目录。
    - 双击这个bat文件就能启动自动化工具
* 处理输出的报告
  + 在系统的目录下会找到一个名为“output”的文件夹，每次运行完之后会在里边自动生成2份报告，报告是以时间来命名的。一份是xml格式的，一份是html格式的。在html格式的报告中有详细的测试结果和截图。

# XML文件详细说明

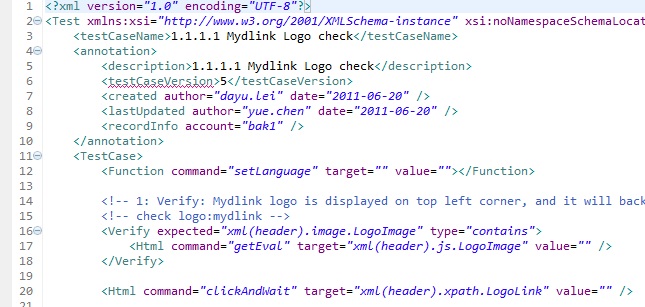
## webSite

下图即是webSite类的xml文件结构



* xpath标签里包含了一个叫“name”的属性，它是用于来寻找xpath的标记，标签里的内容即为真实的xpath。
* 当有相同的name时，取第一个name的对应xpath标签里的值。
* js和image标签与xpath标签类似。
* message标签带有不同语言的子标签，是为了网站系统的多语言设计的。

## testCase



* testCaseName：测试用例的名称
* description：测试用例的描述
* testCaseVersion：测试用例的版本
* created：创建时间&作者
* lastUpdated：最后修改时间和作者
* recordInfo：使用的账号和设备
* testCase：测试用例的具体步骤

### 数据的初始化

在图中第16行的Verify标签中的expected属性的为”*xml(header).image.LogoImage*”，在程序初始化测试用例的时候”*xml(header).image.LogoImage*”会被转化为webSite下header.xml文件中name为*LogoImage*的image标签的值，具体的xml为”<image name=*"LogoImage"*>/images/logo\_fp.png</image>”，即”*xml(header).image.LogoImage*”会被初始化为*”/images/logo\_fp.png*”。

<Verify expected=*"xml(header).image.LogoImage"* type=*"contains"*>

<Html command=*"getEval"* target=*"xml(header).js.LogoImage"* value=*""* />

</Verify>

会被初始化为

<Verify expected=*"/images/logo\_fp.png"* type=*"contains"*>

<Html command=*"getEval"* target=*"window.jQuery('a.logo').css('background-image')"* value=*""* />

</Verify>

*“xml(header).image.LogoImage”*能简写成”*image(header).LogoImage”*。其他也可以类似的简写，如*”xpath(header).LogoImage”*和*”js(header).LogoImage”*。

### 

### Html标签

<Html command=*"getEval"* target=*"xml(header).js.LogoImage"* value=*""* />

在html标签中属性*target*和*value*都是为*command*提供数据。

下表是一些Html标签的使用示例：

|  |  |  |  |
| --- | --- | --- | --- |
| Command的值 | target | value | 注释 |
| *open* | O | X | 地址栏输入网址后跳转 |
| <Html command=*"open"* target=*"xml(url).pagePath.Faq"* value=*""* /> | | | |
| *getValue* | O | X | 从selenium中获得值 |
| <Html command=*"getValue"* target=*"xml(faq).xpath.TitleBarText"* value=*""* /> | | | |
| *type* | O | O | 向输入框中输入 |
| <Html command=*"type"* target=*"xml(entrance).xpath.PasswordInput"* value=*"中"*> | | | |
| *attribute* | O | O | 获得元素的属性值 |
| <Html command=*"attribute"* target=*"xml(entrance).xpath.TwitterImageLink"* value=*"href"* /> | | | |
| *clickAndWait* | O | X | 点击元素后等待页面的加载完成 |
| <Html command=*"clickAndWait"* target=*"xml(faq).xpath.Mydlink"* value=*""* /> | | | |
| *click* | O | X | 点击元素 |
| <Html command=*"click"* target=*"xml(faq).xpath.Mydlink"* value=*""* /> | | | |
| *waitForElementDisplay* | O | O | 等待一个元素的出现 |
| <Html command=*"waitForElementDisplay"* target=*"xml(entrance).xpath.WhatsMydlinkButton"* value=*"60" /*> | | | |
| *sleep* | O | X | 等待一段时间 |
| <Html command=*"sleep"* target=*"2"* value=*""* /> | | | |
| *isEditable* | O | X | 判断是否可编辑（true/false） |
| <Html command=*"isEditable"* target=*"xml(entrance).xpath.EmailInput"* value=*""* /> | | | |
| *isElementPresent* | O | X | 判断是否出现（true/false） |
| <Html command=*"isElementPresent"* target=*"xml(header).xpath.SignOut"* value=*""*></Html> | | | |
| *check* | O | X | 单选框/复选框的勾选 |
| <Html command=*"check"* target=*"xml(login).xpath.rememberMeCheckBox"* value=*"" /*> | | | |
| *uncheck* | O | X | 单选框/复选框的不勾选 |
| <Html command=*"uncheck"* target=*"xml(login).xpath.rememberMeCheckBox"* value=*"" /*> | | | |
| *isVisible* | O | X | 判断是否显示（true/false） |
| <Html command=*"isVisible"* target=*"xml(entrance).xpath.News1"* value=*"href"* /> | | | |
| *getTarget* | O | X | 获得target的值 |
| <Html command=*"getTarget"* target=*"${mailFromPersonal}"* value=*""* /> | | | |
| *isTextPresent* | O | X | 判断字符串是否出现（true/false） |
| <Html command=*"isTextPresent"* target=*"xml(404Error).message.PageError404DescriptionText2"* value=*""* /> | | | |
| *goBack* | X | X | 点击浏览器上的返回键 |
| <Html command=*"goBack"* target=*""* value=*""* /> | | | |
| *waitForPageToLoad* | O | X | 等待页面加载完成 |
| <Html command=*"waitForPageToLoad"* target=*""* value=*""* /> | | | |
| *getSelectedLabel* | O | X | 下拉列边框选中的标签值 |
| <Html command=*"getSelectedLabel"* target=*"xml(myProfile).xpath.CountryValue"* value=*""* /> | | | |
| *openWindow* | O | X | 新窗口打开一个地址 |
| <Html command=*"openWindow"* target=*"xml(footer).xpath.ContactUsLink"* value=*""* /> | | | |
| *addCustomRequestHeader* | O | O | 输入路由的账号密码（IE需开代理模式） |
| <Html command=*"addCustomRequestHeader"* target=*"account(Direct).device.1.userName"* value=*"account(Direct).device.1.passwd"*></Html> | | | |
| *getLocation* | X | X | 获得当前地址栏里的地址 |
| <Html command=*"getLocation"* target=*""* value=*""* /> | | | |
| *runScript* | O | X | 执行一段javascript |
| <Html command=*"runScript"* target=*"xml(support).js.932LButtonClick"* value=*""* /> | | | |
| *mouseDown* | O | X | 触发mousedown的方法 |
| <Html command=*"mouseDown"* target=*"xml(support).xpath.RightButton"* value=*""* /> | | | |
| *mouseUp* | O | X | 触发mouseup的方法 |
| <Html command=*"mouseUp"* target=*"xml(support).xpath.RightButton"* value=*""* /> | | | |

### Function标签

<Function command=*"setLanguage"* target=*""* value=*""*></Function>

下表是一些Function标签的使用示例：

|  |  |  |  |
| --- | --- | --- | --- |
| Command的值 | target | value | 注释 |
| *setLanguage* | X | X | 登出再选择网站系统语言 |
| <Function command=*"setLanguage"* target=*""* value=*""*></Function> | | | |
| *selectDevice* | O | X | 选择设备 |
| <Function command=*"selectDevice"* target=*"account(onLine).device.1.sn"* value=*""*></Function> | | | |
| *getEmailInfo* | O | X | 根据主题和收件人收取邮件，并把邮件内容放入临时变量 |
| <Function command=*"getEmailInfo"* target=*"forgetPassword"* value=*""* /> | | | |
| *waitForLiveView* | O | X | 等待liveview出现 |
| <Function command="waitForLiveView" target="120" value="" /> | | | |
| *getConnectionType* | O | O | 获得连接方式（Direct/Port/Relay） |
| <Function command=*"getConnectionType"* target=*"120"* value=*""* /> | | | |
| *getAllConnectionType* | O | X | 获得该账户下所有设备的连接方式 |
| <Function command=*"getAllConnectionType"* target=*"120"* value=*""* /> | | | |

### Verify标签

<Verify expected=*"xml(header).image.LogoImage"* type=*"contains"*>

<Html command=*"getEval"* target=*"xml(header).js.LogoImage"* value=*""* />

</Verify>

Verify标签其实就是验证下html标签或者function标签的返回值，每个html标签或者function标签都会有返回值。

Verify标签里有个属性叫type，它的值可以是contains，in，equals（默认值），notequals，notcontains，notin等等。

### Assert标签

<Verify expected=*"xml(header).image.LogoImage"* type=*"contains"*>

<Html command=*"getEval"* target=*"xml(header).js.LogoImage"* value=*""* />

</Verify>

Assert标签与Verify标签类似，但是如果这个标签fail了会导致整个case跳过。

### Choose标签

<Choose>

<When>

<Verify expected=*"re\_checkbox"*>

<Html command=*"attribute"* target=*"xml(entrance).xpath.RememberMeCheckBox"* value=*"class"* />

</Verify>

</When>

<Do>

<Html command=*"runScript"* target=*"xml(entrance).js.RememberMeBoxClick"* value=*""*></Html>

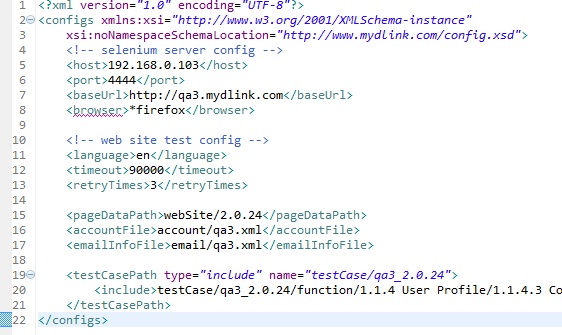
</Do>

<Otherwise></Otherwise>

</Choose>

Choose标签其实就是Verify，Html，Function标签的组合使用，依靠Verify来判断，如果正确就执行Do，反之就执行Otherwise。

## config



* Host：运行测试用例主机的IP地址
* Port：运行测试用例主机的端口号
* baseUrl：测试网站系统的地址
* browser：测试使用的浏览器类型
* language：测试网站的语言项
* timeout：超时时间设置
* retryTimes：等待live view 的重试次数
* pageDataPath：网站页面数据文件夹的路径
* accountFile：测试使用的账户库
* emailInfoFile：测试使用的邮件库
* testCaseParh：测试用例