文档密级:公司内部 A

RedwoodHQ 使用文档（用例编写及测试执行）

陈晨

RedwoodHQ 使用文档（用例编写及测试执行）.....1
1. 背景.....5
2. 安装.....6
2.1 服务端.....6
2.2 客户端（测试机器）.....6
2.3 一些问题.....7
3. 主界面功能介绍.....8
3.1 选择项目 Choose Project: Sample 8
3.2 设置 Settings 8
3.2.1 用户管理 Users 8
3.2.2 项目管理 Projects 9
3.2.3 邮件配置管理 Email 11
3.3 掌握 Action Actions 11
3.4 用例编写 Test Cases 12
3.4.1 左侧 Test Cases 页面 Test Cases 12
3.4.2 左侧 Test Case Tree 页面 Test Case Tree 12
3.4.3 左侧用例搜索及过滤页面 8 ALL 12
3.4.4 编写用例.....12
3.4.4.1 用例名称 Name: 13
3.4.4.2 用例描述 Description: 13
3.4.4.3 用例状态 Status: To be Automated 13
3.4.4.4 用例标签 Tags: 14
3.4.4.5 用例类型 Test Case Type: 14
3.4.4.6 用例历史 History 14
3.4.4.7 用例清理操作 After State 14
3.4.4.8 用例主体——ActionCollection Action Collection 17

3.4.4.9 用例主体——SelectScript Select Script: 18

3.4.4.10 数据驱动 Test Case Data 18

3.5 测试执行 Execution 20

3.5.1 测试机器管理 Machines 20

3.5.1.1 测试机器 IP Host Name/IP 20

3.5.1.2 测试机器客户端端口 Port 20

3.5.1.3 测试机器 VNC 端口 VNC Port 20

3.5.1.4 测试机器客户端最大线程数 Max Threads 20

3.5.1.5 测试机器标签 Tags 20

3.5.1.6 测试机器角色 Roles 20

3.5.1.7 测试机器描述 Description 21

3.5.1.8 测试机器状态 State 21

3.5.1.9 编辑机器配置 21

3.5.1.10 删除机器 21

3.5.1.11 机器环境变量配置 21

3.5.2 全局变量管理 Variables 22

3.5.1 是否为执行级别的全局变量 Execution Var 22

3.5.2 全局变量的标签 Tags 22

3.5.3 全局变量名 Name 23

3.5.4 全局变量默认值 Value 23

3.5.5 允许的取值 Possible Values 23

3.5.6 编辑全局变量 23

3.5.7 删除全局变量 23

3.5.3 测试集管理 Test Sets 23

3.5.4 测试执行管理 Executions 24

3.5.4.1 测试执行搜索 Search: 24

3.5.4.2 显示被锁定的测试执行 Show Locked: 25

3.5.4.3 删除测试执行 25

3.5.4.4 对比测试执行 25

3.5.4.5 测试执行名称 Name Name: 25

3.5.4.6 测试集 Test Set Test Set: 25

3.5.4.7 测试执行运行状态 Status 25

3.5.4.8 测试执行运行次数统计 Totals 25

3.5.4.9 测试执行标签 Tags Tags: 26

3.5.4.10 测试执行上次运行时间 Last Run 26

3.5.4.11 测试执行运行时间 Run Time 26

3.5.4.12 锁定测试执行 Lock State: Lock 26

3.5.4.13 编辑测试执行 26

3.5.4.14 删除测试执行 26

3.5.4.15 邮件提醒 Email Notification 26

3.5.4.16 常用设置 Settings 27

3.5.4.17 全局变量设置 Set Variables 27

3.5.4.18 测试执行总体情况 Execution Totals 27

3.5.4.19 测试执行总体情况图 Execution Chart 27

3.5.4.20 测试执行机器选择 Select Machines 28

3.5.4.21 测试执行用例选择 Test Cases 28

1. 背景

自动化测试从出现到发展到今天，一共经历了三个阶段：

1. 脚本录制/回放阶段。该阶段主要是以工具录制并记录操作的过程和数据形成脚本，通过回放来重复人工操作的过程。该方法的优点很明显：脚本生成简单，无需手动编写或者仅需少量修改即可投入使用；每一个用例都可以对应一个脚本，将脚本按照用例层次管理起来即可，非常方便。但是，该方案存在一个非常致命的缺点：界面或则操作的变化带来的维护成本太高。假如说界面自动化测试中，一个元素的查找关键（比如 ID 或者控件 Handle 什么的）发生了变化，那么与之相关的所有脚本都要进行修改，如果脚本量非常大，代价太大；另外，如果脚本不得不进行修改才能够使用，就得了解该工具的脚本编写方法，因此，对编写技术也是有一定的要求的。
2. 数据驱动 (data driven) 阶段。方法是：从数据文件读取输入数据，通过变量的参数化，将测试数据传入测试脚本。比如大家最了解的将测试数据样本按行存到 csv 或者 excel 当中，测试工具或者脚本读取这些文件，将数据分别传到待测产品中进行测试。该方法的优点：数据与脚本分离，可重用性提高，一个脚本或者工具就能搞定多组用例。缺点也很明显：仍然没能彻底解决变化带来的维护成本；脚本编写技术掌握需求仍然较高。
3. 关键字驱动（keyword driven）：按照关键字/Action 进行分解，形成一系列的关键字对象用于测试。包含被操作对象、操作、参数和返回值四大类。优点：数据、脚本实现及对象、关键字三者全部分离，因此受待测对象变化影响最小（只需更新对应脚本即可，可能就几个，如果采用对象库维护，可能只需要改一个配置）。但它也有自己的缺点：依照上层越简单，底层越复杂的原则，底层脚本实现难度高，因此对底层开发人员要求非常高，难度超过了前两个阶段。

本文的说明对象 RedwoodHQ 是基于关键字驱动阶段的测试工具，它与比较著名的测试框架

RobotFramework 比较接近，且各有各的优点和缺点。在这里就不做横向比较了。

因为 RedwoodHQ 系统较为复杂，小功能点很多，本文档会尽可能的将经常会用的功能点介绍出来。遗漏也在所难免，欢迎大家指出，也欢迎大家花点时间自行探索一下该系统。

2. 安装

2.1服务端

目前电信平台组的服务端已经部署好了，地址是 http://192.168.59.51:3000. 所以电信平台组的各位同事可以跳过此步骤，只需部署客户端就行了。

从官方网站 http://redwoodhq.com 下载服务端安装包，直接安装即可。安装过程中可以选择是否安装 VNC（用来远程查看测试机器，服务端一般都选择装），同时也可以调整服务端的访问端口等等选项，请按需配置。

如需要在安装后调整配置，可以修改安装目录下的 properties.conf.

安装完成后，通过 http://服务端 IP:服务端端口（默认是 3000）来访问。默认管理员用户名密码：admin/admin

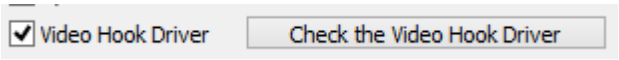
安装完成后，任务管理器里会出现两个进程: node.exe 和 mongod.exe. 如安装了客户端则会有两个 node.exe.

2.2客户端（测试机器）

因为客户端安装包比较大，建议在服务端安装好以后，从服务端左上角的这个链接 Download Agent 下载，比从官网下载快（因为是服务端自带的）。

安装时同样可以选择是否安装 VNC. 但由于安装的 UltraVNC 是绿色的，没有带显示器驱动，所以在 Windows 下的表现不好，远控时很卡，如需要通过 RedwoodHQ 远程查看 Windows

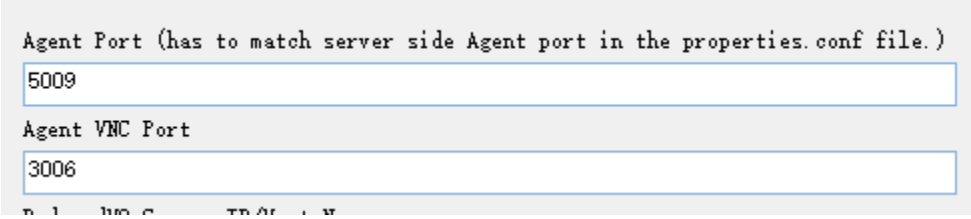
测试机器的话，请到 UltraVNC 的官网下载一个显示器驱动安装到客户端上，能大大提高性能（地址：http://www.uvnc.com/downloads/mirror-driver.html, XP 以上请选择 Win7 驱动）。安装后重启系统，重启后在系统托盘处右键 VNC 托盘图标，选择 Properties, 勾上此项：



即可。

安装时也可以配置 IP 地址和端口等等一些参数，注意：

这两项需要和服务端的 IP 地址和端口号保持一致



这两项需要和服务端配置的端口一致。

安装完成后，任务管理器里会出现 node.exe 进程，同时在服务端的机器管理页面中会发现此机器的信息。

2.3一些问题

1. Windows 服务端据说比 Linux 服务端表现稳定且性能好一些，所以建议在 Windows 下部署服务端。
2. 官方仅提供了 Ubuntu 版本的服务端和客户端，是否能在其他 Linux 发行版本下运行尚未测试。
3. Windows 下安装时如果使用了远程桌面，在断开或者最小化远程桌面的时候安装进程疑似会暂停，所以请保持远程桌面窗口，待安装完毕后再行断开或最小化（安装过程时间挺长）
4. 服务端或者客户端的 Windows 版本在多人登录这台机器时，会出现多个 node.exe 进程，应该是该工具将启动项放在了 HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\Run

里有关，因为貌似不影响使用，可以暂时不管。

5. 在第 4 步出现的情况下，会出现使用工具提供的"Stop RedwoodHQ" 和"Stop Agent RedwoodHQ"没用的情况（点击后发现 node.exe 或者 mongod.exe 进程都还在）。此时可以采取杀对应进程的方式解决。

3. 主界面功能介绍

登录服务端网站后默认进入 Execution 主界面。主界面的主要功能大致划分为：

3.1 选择项目 Choose Project: Sample

在第一次使用 RedwoodHQ 的时候，仅有一个自带的 Sample 项目。后续使用的时候，需要对应产品创建不同的项目。

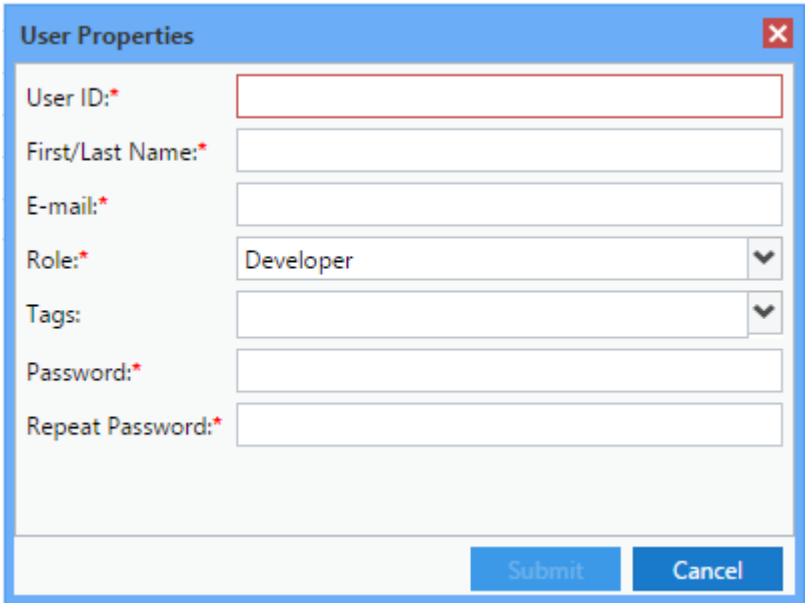
3.2设置 Settings

系统设置页面，通常由管理员管理。主要分为以下三个设置内容：

3.2.1 用户管理 Users

可以在此处管理使用 RedwoodHQ 的用户（添加、配置、删除等）。**目前该系统权限部分做的不够好，会发现各个用户都有创建用户和调整系统配置的权限。为了防止出现问题，目前建议该操作由 admin 账户进行。**官方也提供了严格的权限管理补丁，目前尚未尝试：https://github.com/dmolchanenko/RedwoodHQ/pull/8

使用 Add User 可以添加用户。弹出的页面如下所示：

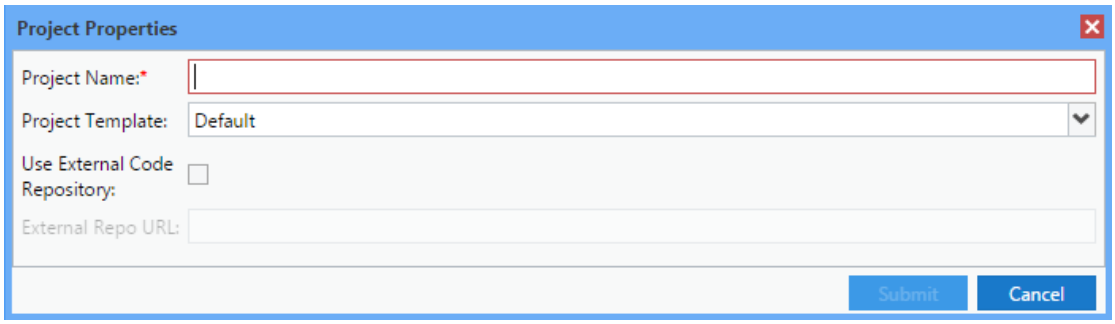


注意：

1. Role 分为 Developer（底层开发人员，对整个系统都有访问和操作权限）以及 Test Designer（用例设计和执行人员。Script 部分仅有查看代码和 Git pull 更新代码的权限）。创建用户时建议严格按照角色权限进行选择。
2. Tags 用来创建备注，一般在搜索或者分类的时候用到。创建 tag 的方法是在 Tags 框里输入 tag 名称后回车，会自动创建一个 tag. 已经创建的 tag 在下拉框里可以选择。

3.2.2 项目管理 Projects

可以在此处管理项目（添加、配置、删除等）。使用 Add Project 可以添加项目。弹出的页面如下所示：



其中：

1. 可以选择 Project Template 下拉框里的一个模板，来创建一个项目（默认安装只有 Java Based Selenium 一个类型）。例如如果想创建使用 Java 或者 Groovy 代码的网页类测试项

目，即可以在下拉框里选择 Java Based Selenium，这样创建好的项目里已经有现成的基于 Selenium 的各种 Action 了，可以直接拿来使用。

如果想创建自定义的模板，请参考服务端安装目录\project_templates\java_project_selenium 目录里的结构，新建一个目录例如 **java_project_appium**，把代码等等按照此结构放进去，随后修改服务端的三处 js 文件即可，分别是：

1) cli\createProject.js，追加红框里的内容。

```
var argv = require('optimist')
    .usage('Usage: $0 --name [name] --template [Default, Java Based Selenium, Java Based Appium]')
    .demand(['name','template'])
    .argv;
```

2) public\view\ProjectEdit.js，追加红框里的内容。

```
xtype:'combo',
fieldLabel: 'Project Template',
store: ["Default","Java Based Selenium", "Java Based Appium"],
name: 'template',
value:"Default",
forceSelection: true,
editable: false,
allowBlank: false,
listeners: {
```

3) routes\scripts.js，添加红框内的如下内容，也就是多添加几个判断。注意如果是其它语言，需要另起一个 language 判断，这里不再细说：

```
exports.CreateNewProject = function(projectName,language,template,callback){
    var templatePath = "";
    if(language == "Java/Groovy"){

        if (template == "Java Based Selenium") {
            template = "java_project_selenium";
        }
        else if (template == "Java Based Appium") {
            template = "java_project_appium";
        }
        else{
            template = "java_project";
        }
        //templatePath = path.resolve(__dirname,"../project_templates/"+"multi_test");
        templatePath = path.resolve(__dirname,"../project_templates/"+template);
    }
```

4) 完成后重启服务端即可生效。

2. RedwoodHQ 默认是使用内置的 git 源来进行版本控制的，如果想使用外部的 git 源，可以将 Use External Code Repository: ☑ 勾选，在 External Repo URL: 处填写外部 git 源的地址。

3.2.3 邮件配置管理 Email

此处是配置发送邮件的一些参数，就不详细介绍了。其中 Server Host:* 配置成服务端的 IP 地址就行了。

3.3掌握 Action Actions

对于用例编写及执行人员，尤其是初次使用此系统的人员，首先要做的事情就是掌握目前已有的 Action 的含义和用法，具体步骤是：

1. 先理解 Action 的含义（实际上就是把测试过程中的操作方法抽取整理出来，形成了一系列共用的方法，加以实现，从而形成了 Action，使用 Action 就类似于调用函数。），知道每个用例实际上就是 Action 的组合。

2. 打开 Actions 界面，查看左侧 Actions 列表里的所有 Action. 通过查看每个 Action 的描述 Description: 和所需参数 Add Parameter | Name | Parameter Type | Possible Values ，了解每个 Action 所做的事情，所需要传递的参数等等各种细节。在期间，如果对其中的 Action 理解有困难的话，可以求助于 Action 编写人员（一般是底层开发）；如果对 Action 的描述、参数设置等等各种细节不满意，或者发现缺少自己需要的 Action 的话，可以直接找编写人员，要求其添加 Action，或者修改 Action 的描述或者设计等等。这样下来，对于每个 Action 都有一个全面的了解的话，写用例自然就不在话下了。

3. 注意：不要自行编辑、删除 Action，就算当前有权限。尤其是删除 Action，会对已有的用例造成影响，导致用例无效。

3.4用例编写 Test Cases

3.4.1 左侧 Test Cases 页面 Test Cases

展示的是当前项目中所有的用例

3.4.2 左侧 Test Case Tree 页面 Test Case Tree

是按照每一个用例的 tag 来分类展示的，查看更方便更清晰，因此建议每个用例都加上自己的 tag, 后面会详细介绍方法。

3.4.3 左侧用例搜索及过滤页面 8 ALL

位于 Test Cases 下面。

上面的文本框是用来搜索用例的，目前暂只能按照用例名称来进行检索，不支持用例描述检索。**所以建议大家在给用例起名的时候，把每个用例的测试对象和特征写在用例名称里，后期用例量大的情况下方便搜索，便于管理。例如 “BVT_001_登录测试_登录名正确”。**

下面的那个下拉框是根据用例状态来筛选用例的。用例编写的过程中，可以指定用例当前状态，比如“已经自动化（Automated）”、“待自动化（To be automated）”、“待维护（Need Maintenance, 表示用例已自动化但是有问题待修改）”。RedwoodHQ 在执行时只会跑 Automated 状态的用例。

3.4.4 编写用例

点击 New Test Case，会展现用例设计界面。

点击 ，可以复制当前的测试用例。

点击 ，可以查看当前用例的实现代码。

3.4.4.1 用例名称 Name:

填写用例名称。建议大家可以把名字起得清楚一点、唯一一点，便于后期查找和维护，例如“BVT_001_登录测试_登录名正确”。

需要大家注意的一点是，**RedwoodHQ 在执行用例的时候，是根据用例名称的先后顺序（就是字符串的大小比较）来执行的**，比如：

- "BVT_001_登录错误"会比"BVT_002_登录正确"先执行（因为前面都一样但是 001 小于 002）
- "BVT"开头的会比"Function"开头的先执行（B 小于 C）
- 汉字也是按照拼音顺序来的，比如“测试”开头的要先于“稳定”开头的用例执行。

因此如果对用例执行顺序有要求的话，可以将用例命名按照需要的顺序排列就可以了。

官方也计划增加自行指定用例执行顺序的功能，不过尚未完成。

3.4.4.2 用例描述 Description:

填写用例描述。建议大家将之前用例的内容全部贴进来，将自动化用例和用例文档对照起来，方便管理。

3.4.4.3 用例状态 Status: To be Automated

表示当前用例完成的状态。分为“已经自动化（Automated）”、“待自动化（To be automated）”、“待维护（Need Maintenance，表示用例已自动化但是有问题待修改）”三种。

RedwoodHQ 在执行时只会跑 Automated 状态的用例。一般这里的状态都填写 "Automated".

3.4.4.4 用例标签 Tags:

描述用例特征的标签，一般建议填描述该用例的分类，例如“功能”、“稳定性”等等。填写方法是输入文字后回车就可以了。

填写后，可以在左边的 Test Case Tree 看到按照 tag 文件夹分类的用例，避免了用例数量太多导致左边展示太多，不好查看的问题，查看更方便更清晰。**所以建议大家每个用例都加上标签。**

可以填入多个 tag 标签，但是会导致左边的 Test Case Tree 展示混乱。而且目前也暂时不支持 tag 分层展示，因此不建议填多个 tag.

3.4.4.5 用例类型 Test Case Type:

该自动化用的实现类型。分为 Action Collection、Junit、TestNG、Pytest、Script 五种。

一般用例类型都是 Action Colletion. 除非底层开发人员明确告知此用例是由 Java 单元测试工具完成的，则根据实际使用框架选择 Junit 或者 TestNG; Python 单元测试工具完成的则选择 Pytest. 一般来说很少有需要直接使用脚本 Script 来完成的用例。

3.4.4.6 用例历史 History

此处记录了用例的历史版本，类似于 SVN 的版本管理。基本上每次保存用例后系统都会记录一次历史。点击 View Test Case 可以查看当时用例的内容。点击 可以将当前用例回滚至这个版本。

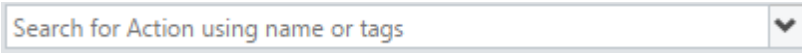
3.4.4.7 用例清理操作 After State

在多个用例的实际运行当中，有很多场景下面，上一个用例执行的一些操作会影响到下一个用例的正常执行。例如上一个用例进行了登录操作，而下一个用例又需要执行登录操作，如果上一个用例没有在结束前注销的话，会导致下一个用例执行失败。这里的 After State 就是为了在每个用例执行完毕后清理环境来准备的。

After State 里的 Action 不论是在用例执行成功还是失败，都会执行一遍。

After State 的展现方式与下面的用例步骤 Action Collection 是一致的，这里就放在一起说说

每个功能的作用。

1) 

这里填写清理环境所使用的 Action. 可以直接填写，系统会根据字母自动展现匹配的 Action（但是暂不支持使用 Action 名称的任意部分来展现匹配），也可以使用右边的下拉按钮来选择一个 Action.

2)

填写好 Action 后使用此按钮来添加 Action. Action 会追加到最后。

3)

在当前选择的 Action 之后、下一个 Action 之前插入一个 Action.

提示：当前选中的 Action 是以浅灰色背景显示的，未被选中的 Action 是以深灰色背景显示的。

4)

复制当前选中的 Action.

5)

将之前复制的用例粘贴到最后。提示：此系统支持用例间的 Action 的复制粘贴。

6)

将所有的 Action 的细节收缩起来。

7)

将所有的 Action 的细节展开。

8)

控制 Action 出错后的行为。分为三种：

Record Error Stop Test Case: 记录当前的出错信息（可以在随后的运行报告里看到），并停止该用例的执行。

Record Error Continue Test Case: 记录当前的出错信息，但是该用例的执行继续。

Ignore Error Continue Test Case: 忽略当前的出错信息继续执行用例（出错信息将不会被记录）。

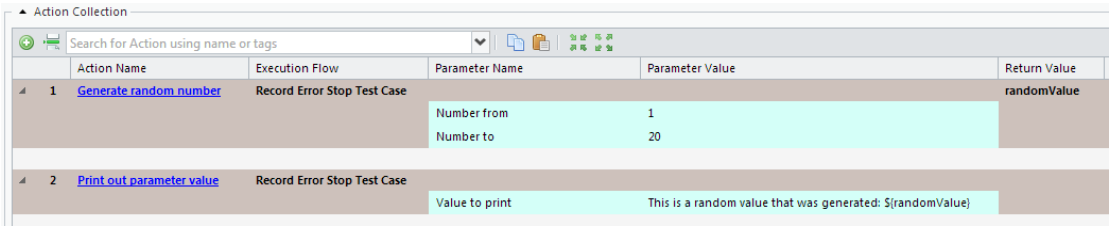
请按需选择。

9) Parameter Name | Parameter Value

Action 所需要的参数名称和参数值。请根据 Action 里的描述，对应填写参数值。**注意：如果想传空值，将参数值单元格清空即可，请不要使用 RedwoodHQ 提供的<NULL>.**

10) Return Value

接收该 Action 返回值的参数。比如如果想把一个 Action 的返回值作为参数传递给下面的 Action，这里可以用一个**变量名**来接收，然后在下一个 Action 里当做参数**${变量名}**传进去即可。例如下面的例子：

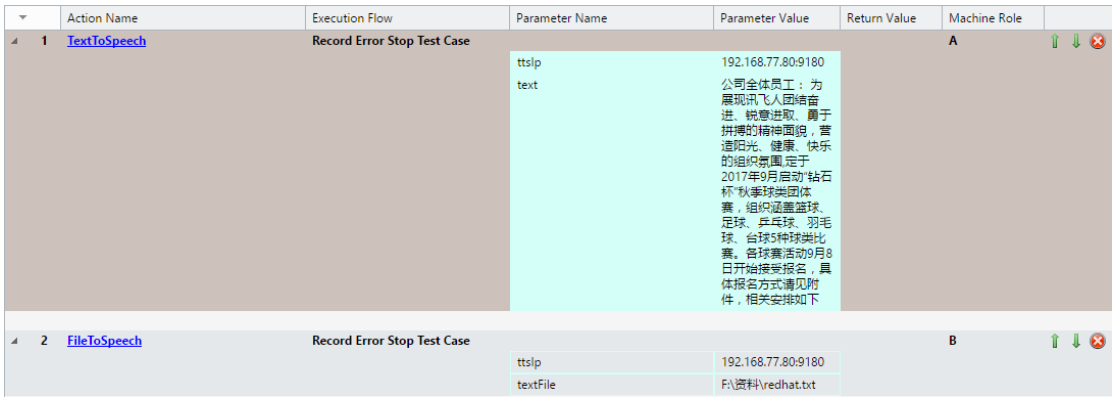


"Generate random number"生成 1 到 20 的一个随机数，返回值保存在 **randomValue** 里。随后**"Print out parameter value"**会打印生成的随机数，注意参数值里的**${randomValue}.**

11) Machine Role

机器在测试中的角色。在一些测试当中，单个用例需要在不同的机器上执行不同的操作，例如，在 A 机器上启动一个服务，启动 OK 后再在 B 机器上运行一个工具。可以通过 Machine Role 这个功能达到这个目的，具体方法是：

- 在测试执行 Execution 页面的机器管理 Machines 中找到自己需要用的机器，比如机器 A 和机器 B. 点击两个机器后面的 或者双击机器那一行对其进行编辑，在 Roles 一列为两个机器定义一个角色名（比如就是 A 和 B，当然还是建议定一个容易理解的名字，例如就是机器的 IP 地址去掉点），填写完成后回车并点击 Update 提交。
- 随后，在用例里，为需要指定机器运行的 Action 都指定上 Machine Role，例如：



这里的 "TextToSpeech" Action 就会在机器 A 上运行，而 "FileToSpeech"会在机器 B 上运行，也就达到了分开运行的目的。

- 不要忘了在测试执行 Executions 的时候，把角色名为 A 和 B 的这两台机器都勾选上：

Select Machines

Search:

	Host Name/IP	Threads	Max Threads	Port	Roles
	10.5.19.46	1	10	5009	Default
	10.5.19.45	1	10	5009	Default
	10.5.19.47	1	10	5009	Default
☑	10.5.20.37	1	10	5009	A
	10.5.21.9	1	1	5009	Default
	10.5.220.254	1	1	5009	Default
	10.5.224.171	1	1	5009	Default
	192.168.50.69	1	10	5009	Default
☑	192.168.59.51	1	10	5009	B

- 注意，需要此功能的时候不要将机器的 Machine Role 取成一样的名字，这样只会使得用例在其中的随机一台机器上运行，而不是并发执行。
- 可以为一台机器取多个角色名。
- 需要两台以上的机器的话，就多取几个 Role 的名字。
- 建议 Role 名字就取成该机器的 IP 地址去掉点，这样非常容易区分。

12)

将当前选中的 Action 上移一个位置。

13)

将当前选中的 Action 下移一个位置。

14)

删除当前选中的 Action.

3.4.4.8 用例主体——ActionCollection

Action Collection

如果在用例类型中选择的是 "ActionCollection"的话，这里就是添加用例 Action 集合的地方。整体的布局和功能与 AfterState 一致，可以参考用例清理操作里的说明，这里就不加阐述了。

3.4.4.9 用例主体——SelectScript Select Script:

如果在用例类型中选择的是 "Junit"、"TestNG"、"Script"或者"Pytest"的任意一种的话，这里就是添加用例运行的主方法。点击按钮，在弹出的树状图中找到对应的脚本及方法名后确定， Language: 处选择正确的对应语言即可。

3.4.4.10 数据驱动 Test Case Data

一些用例可能会对同一个场景传入不同的数据，检查功能是否正确。还是可以以登录作为例子，比如用户名为空、用户名超长、用户名含有特殊字符、密码为空、密码超长、密码没有特殊字符、密码有特殊字符……等等。如果用正常的用例设计思路的话，每个场景都是一个用例，这样会有很多个用例。其实对于这种操作完全一样的用例，可以这样做：比如上面的登录验证，就设计一个用例，把用户名、密码分别定义为一个参数，把不同的测试场景数据放在 csv 文件里，每个场景一行，然后用例每次读取一行进行测试，这样就达到了只设计一个用例测试所有场景的目的。这就是数据驱动 (Data Driven)测试。

本系统通过 Test Case Data 这个功能同样可以做到数据驱动测试。以登录作为例子，假如登录测试需要测的是用户名、密码，方法是：

1) 通过点击 Add Column，分别加入名为 Username、Password 的列。实际测试中根据自己要验证多少个点来实际添加多少列，添加后的效果如下：

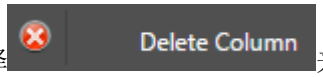
Add Column Add Row

	UserName	Password

2) 通过点击 Add Row，来为每列的不同参数赋值。比如按照上面我们设计的登录场景，赋值后的效果如下：

Add Column Add Row

	UserName	Password
1		123456
2	111111111111111111111111111111111111111111	123456
3	!@#$%^&*(((_	123456
4	admin	
5	admin	111111111111111111111111111111111111111111111111111
6	admin	123456
7	admin	!@#$%^&*()_+

可以右键每列的变量名，选择 

来删除此列；可以通过点击每行后面的 来删除每行。

3) 编写用例。上面的用户名 UserName、密码 Password 在用例的 Action 里面，可以用 ${TCData.UserName}、${TCData.Password}来表示（${TCData.列名}。在参数下拉框里也能看到设置好的列名）。如图所示：

	Action Name	Execution Flow	Parameter Name	Parameter Value
1	Open Browser	Record Error Stop Test Case		
			URL	www.amazon.com
			Browser Type	${Browser}
2	Set Username	Record Error Stop Test Case		
			username	${TCData.UserName}
3	Set Password	Record Error Stop Test Case		
			password	${TCData.Password}
4	Login	Record Error Stop Test Case		
5	Logout	Record Error Stop Test Case		

注意图中红框里的两处，用户名和密码就分别以${TCData.UserName}、${TCData.Password}来表示了。

另外注意登录测试后需要使用 logout 注销，以防下一次登录时，页面已经是已登录状态。

4) 注意在执行前，要在 Execution 页面的 Test Cases 下面，点击一下 按钮，系统会根据场景数量，自动生成对应数量的用例。例如上面设计的场景有 7 组数据，这里就分别生成了 7 个用例：

Test Cases

Search:

Name	Tags	Status
test tts_1	ttsdemo	Not Run
test tts_2	ttsdemo	Not Run
test tts_3	ttsdemo	Not Run
test tts_4	ttsdemo	Not Run
test tts_5	ttsdemo	Not Run
test tts_6	ttsdemo	Not Run
test tts_7	ttsdemo	Not Run

随后全部勾上，执行即可。

3.5测试执行 Execution

3.5.1 测试机器管理 Machines

此处管理已注册到 RedwoodHQ 服务端的测试机器的配置。一般安装客户端以后，机器会自动注册到服务端，当然如果有需要也可以通过 Add Machine 来手动添加测试机器。

可以通过双击测试机器一行或者点击后面的修改机器配置。

3.5.1.1 测试机器 IP Host Name/IP

如果测试机器在安装客户端的时候安装了 VNC 的话，可以点击机器 IP 远程访问。

3.5.1.2 测试机器客户端端口 Port

测试机器上的 RedwoodHQ 客户端所使用的端口。

3.5.1.3 测试机器 VNC 端口 VNC Port

测试机器上的 VNC 所使用的端口。

3.5.1.4 测试机器客户端最大线程数 Max Threads

如果需要在一个客户端上执行多个测试任务，需要调大这里的数值。

3.5.1.5 测试机器标签 Tags

可以为每台测试机器添加一个标签，便于区分。

3.5.1.6 测试机器角色 Roles

参考“用例清理操作”一节对于 Machine Role 的描述。这里推荐将 Role 就设置为机器的 IP

地址去掉点。

3.5.1.7 测试机器描述 Description

可以在此处对每台测试机器添加一个说明。

3.5.1.8 测试机器状态 State

这里一般显示的是机器上有没有在运行任务，运行了多少个任务等。

3.5.1.9 编辑机器配置

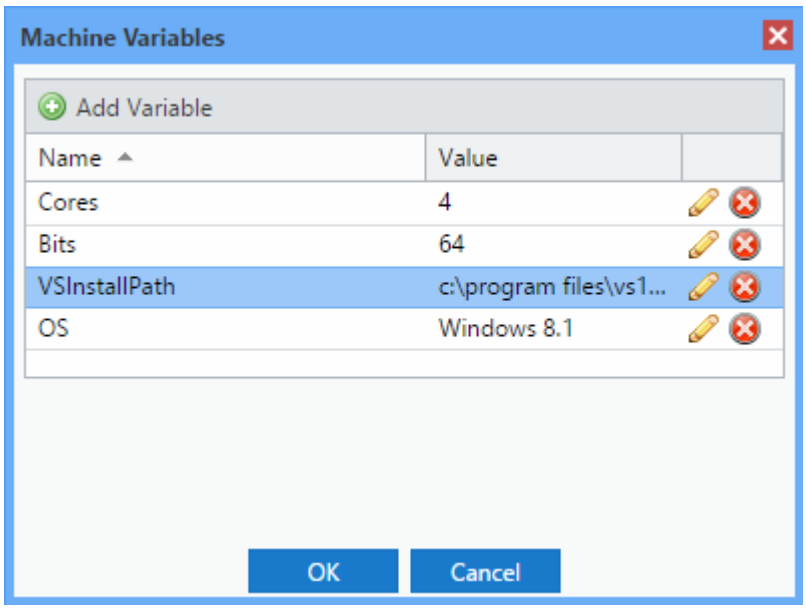
可以点击它来对已有的机器配置进行编辑。

3.5.1.10 删除机器

可以点击它来删除对应的机器。

3.5.1.11 机器环境变量配置

在实际测试当中有可能需要获取机器的一些信息，例如操作系统版本、CPU 数量、操作系统位数等等，可以点击此按钮，在弹出的窗口中设置一些环境变量保存这些信息。例如：



随后在用例的 Action 里面，可以通过**${Machine.角色名(Role).变量名}**来访问。比如我要获得 Role 为 1052037 这台机器的核数 Cores，就可以用**${Machine.1052037.Cores}**来访问。

3.5.2 全局变量管理 Variables

此处可以定义全局变量。全局变量可以被用例 Action 和代码使用。Action 里使用全局变量的格式是**${全局变量名}**，而代码使用的格式是 **redwood.launcher.Launcher.variables.全局变量名**。

通过点击 Add Variable 添加全局变量定义， Search: 搜索全局变量。

3.5.1 是否为执行级别的全局变量 Execution Var

如果这里的框被钩上的话，每个 Execution 里可以单独对其赋值，也就是说每个 Execution 可以使用不同的值。
如果没有被勾上的话，所有的 Execution 就只能用同一个值，也就是在这里配置的默认值，见下面的“全局变量默认值”一节。

3.5.2 全局变量的标签 Tags

可以为每个全局变量添加一个标签，便于区分。

3.5.3 全局变量名 Name

全局变量名称。

3.5.4 全局变量默认值 Value

此全局变量默认值。如果 Execution Var 选项被钩上的话，每个 Execution 里可以单独配置值，否则全部 Execution 都只用这里配置的默认值。

3.5.5 允许的取值 Possible Values

此处定义该全局变量允许的取值，即值可以从此处定义的值任取其一。定义的方法是输入值以后回车。

3.5.6 编辑全局变量

可以点击它来对已有全局变量进行编辑。

3.5.7 删除全局变量

可以点击它来删除对应的全局变量。

3.5.3 测试集管理 Test Sets

此处管理用例的测试集合。比如，功能测试集就是所有功能用例的集合等等。

一个 Execution 对应一个测试集，所以按自己的执行需要来定义测试集。

使用 New Test Set 来添加一个测试集，在 Test Set Name: 上填上测试集的名称。下面的用例列表里勾选上需要添加的用例，保存即可。

3.5.4 测试执行管理 Executions

测试执行管理页面。新建测试执行、执行和结果展示都在这里。

通过 New Execution 添加新的测试执行。

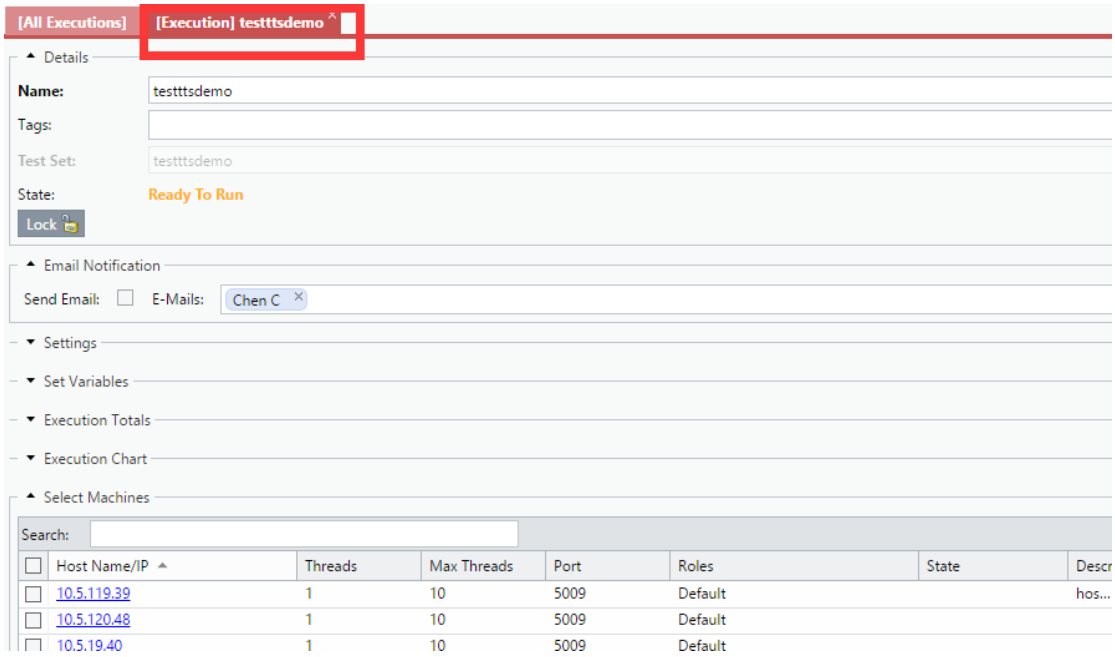
通过保存测试执行。

通过导出当前测试执行上次的执行结果为 PDF 格式。

通过运行当前的测试执行。

通过停止当前的测试执行。

注意：如果对上一次执行的用例进行了修改，在下一次执行前需要关闭之前打开的执行窗口，也就是这个窗口：



然后再打开并进行执行操作，否则有可能出现执行的还是之前的用例内容的问题。

3.5.4.1 测试执行搜索 Search:

搜索对应的测试执行。

3.5.4.2 显示被锁定的测试执行 Show Locked:

如果此项被选中，那些被标记为锁定状态的测试执行会被显示。锁定功能的解释请参见下面的“锁定测试执行”一节。

3.5.4.3 删除测试执行

删除被勾选上的测试执行。

3.5.4.4 对比测试执行

假如两个测试执行跑的是一样的用例（例如两个冒烟测试），需要对比这两次执行结果的区别的话，可以选中这两次执行，随后点击此按钮，系统将会展示两者的运行数据，包括运行成功/失败/未运行的数量，和每个用例的详细运行结果。

3.5.4.5 测试执行名称 Name Name:

测试执行命名。

3.5.4.6 测试集 Test Set Test Set:

该测试执行所包含的测试集。在新建测试执行界面，可以选择一个测试集，作为此测试执行执行的用例集合。

3.5.4.7 测试执行运行状态 Status

包括 Running（正在执行）和 Ready to Run（等待执行）。

3.5.4.8 测试执行运行次数统计 Totals

由三个数字表示，分别表示运行成功、运行失败、运行中断（例如运行中点了停止按钮）。

3.5.4.9 测试执行标签 Tags Tags:

测试执行对应的标签。

3.5.4.10 测试执行上次运行时间 Last Run

上一次执行该测试执行的时间。

3.5.4.11 测试执行运行时间 Run Time

上一次该测试执行运行花费的时间。

3.5.4.12 锁定测试执行 Lock State: Lock

如果锁定了一个测试执行，该测试执行不能被运行。

3.5.4.13 编辑测试执行

可以点击它来对测试执行进行编辑。

3.5.4.14 删除测试执行

可以点击它来删除对应的测试执行。

3.5.4.15 邮件提醒 Email Notification

设置测试执行完成后是否通过邮件发送测试结果。

3.5.4.16 常用设置 Settings

设置测试执行的一些常用参数。具体是：

1) Retry Count: 0

设置某个用例 Action 失败后的重试次数。默认为 0，即不重试。

2) Ignore Status:

勾选后，测试用例中标记为“待自动化（To be automated）”、“待维护（Need Maintenance）”状态的用例依然被执行。

3) No Screen Shots:

勾选后，整个测试执行过程中不会进行截图操作（失败的 Action 也不会）。

4) UI Snapshots:

勾选后，每个 Action 都会进行一次截图操作。

5) No After State:

勾选后，所有用例中的 AfterState 操作都不会被执行。

3.5.4.17 全局变量设置 Set Variables

之前在全局变量管理里添加的全局变量会在这里显示，可为其单独赋值。
如果全局变量管理里对应全局变量的 Execution Var 没有勾选的话，这里不会显示它，使用其默认值。

3.5.4.18 测试执行总体情况 Execution Totals

展示测试执行总体情况的数据。

3.5.4.19 测试执行总体情况图 Execution Chart

展示测试执行总体情况的饼状图。

3.5.4.20 测试执行机器选择 Select Machines

在此处选择执行测试需要的机器，根据测试需要可以进行单选或者多选。如果测试用例的 Action 里指定了多个 Role 的机器，在这里需要把这些 Role 对应的机器选上。

注意：在多选的情况下，如果被选择的机器的 Role 一样，不代表同一个用例会同时运行在这些机器上，而是会逐个运行。

机器列表里的 Machine Base State 可以编辑指定一个 Action，在所有用例执行前会在此台机器上执行所选择的 Action，成功后才进行用例执行，失败则停止执行用例。此功能可以用在准备机器环境上，例如清理机器或者安装工具等。

机器列表里的 Base State Result 展示 Machine Base State 执行的状态或结果。例如 Running、Passed 或者 Failed.

3.5.4.21 测试执行用例选择 Test Cases

展示了测试执行包含的所有测试用例，勾选需要执行的用例运行即可。

测试用例一行展示了测试用例的执行状态、所用时间、执行结果、错误细节等等。其中在 Note 一列可以为每个用例添加个说明，比如这个用例是为了验证哪个缺陷，或者为啥不执行该用例等等。其余比较容易理解，这里就不详细介绍了。

：**当待执行的用例里包含数据驱动的内容时，执行前请一定记得点击此按钮**，系统将会根据数据内容将含数据驱动的用例扩展成多个用例。

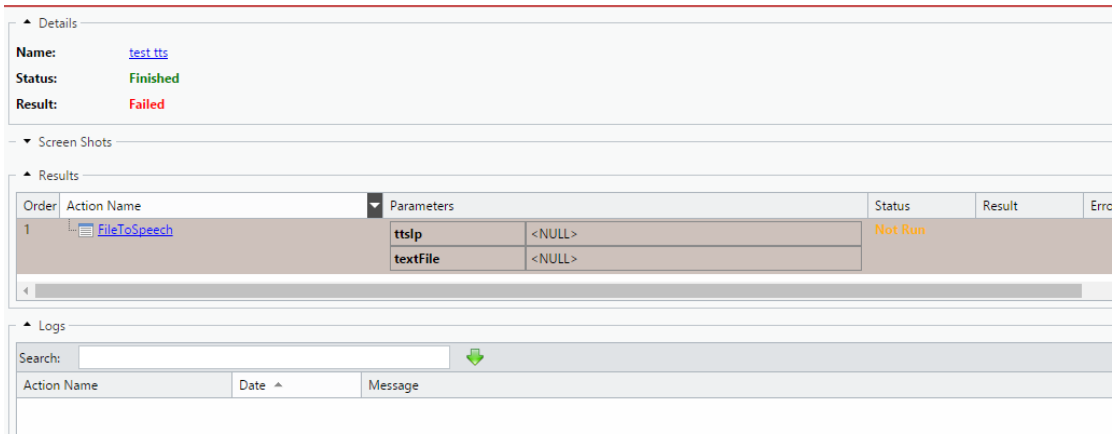
Debug: ☐：勾选此选项后，会多出两个内容：

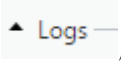
Start Action End Action

在这里可以通过添加用例开始 Action 的序号和结束 Action 的序号，控制用例执行的 Action 范围，从而达到调试的目的。例如，Start Action 填 2，End Action 填 4 后执行，该用例只会运行 Action2、3 和 4.

Search Notes: ：搜索带有指定内容的 Note 所属的用例。

执行过程中可以点击测试用例名称，来实时观察用例的执行过程、执行输出、执行结果等等。点击后展现的界面如下图：



其中  展示的是底层代码屏幕输出的内容。其余部分意义容易理解，也不详细介绍了。