功能自动化测试方案

**目录**

[1 前言 2](#_Toc433188913)

[1.1 文档目的 2](#_Toc433188914)

[1.2 名词术语 2](#_Toc433188915)

[2 功能自动化测试实施原则 3](#_Toc433188916)

[2.1 实施原则 3](#_Toc433188917)

[2.2 实施功能自动化测试的优缺点 3](#_Toc433188918)

[3 实施范围和目标 5](#_Toc433188919)

[3.1 实施范围 5](#_Toc433188920)

[3.2 实施目标 5](#_Toc433188921)

[3.3 总体实施策略 5](#_Toc433188922)

[4 技术方案实施内容 6](#_Toc433188923)

[4.1 Sahi 的特性和优势： 6](#_Toc433188924)

[4.2 Sahi 的工作原理： 9](#_Toc433188925)

[4.2.1 第一步：录制 10](#_Toc433188926)

[4.2.2 第二步：精炼脚本 10](#_Toc433188927)

[4.2.3 第三步：回放 11](#_Toc433188928)

[4.3 Sahi 的安装部署与配置 12](#_Toc433188929)

[5 实施管理建议 20](#_Toc433188930)

[5.1 实施策略建议 20](#_Toc433188931)

[5.2 人员配置 20](#_Toc433188932)

[5.3 实施计划 21](#_Toc433188933)

[5.4 交付物 21](#_Toc433188934)

# 前言

## 文档目的

功能自动化测试方案是为XXX系统功能测试使用自动化工具，实现以自动化测试为主的目标而编写的技术和实施方案。

文档的主要目的是提供自动化测试的技术方案、实施内容、实施步骤，以及关键的技术实现手段等。本文的预期读者为测试中心相关人员。

## 名词术语

* Sahi：是Tyto Software 旗下的一个基于业务的开源 Web 应用自动化测试工具。Sahi运行为一个代理服务器，并通过注入 JavaScript 来访问 Web 页面中的元素。Sahi支持 HTTPS 并且独立于 Web 站点，简单小巧却功能强大。它相对于 Selenium 等自动化测试工具，在动态 ID 元素查找和隐式页面等待处理等方面具有一定的优势。选择Sahi工具来实现具体 Web 项目的自动化测试是一个很不错的选择。
* 功能测试：功能测试又称正确性测试，它检查软件的功能是否符合规格说明。由于正确性是软件最重要的质量因素，所以其测试也最重要。
* 自动化测试：使用商业提供的自动化测试工具或者自己开发的工具对目标系统进行测试。机器自动执行的测试，替代人完成重复性劳动，但不能完全取代人。自动化测试需要用到测试工具，测试工程师的参与，自动化测试技术可应用于所有的测试阶段
* Web 测试背景：随着 Web 技术和互联网的发展，Web 应用产品越来越丰富，基于 Web 页面测试的需求与日俱增。在当前全球软件都在追求高效、敏捷的开发模式的大背景下，Web 自动化测试成为了新一波技术探讨和研究的热潮。因为传统的手工测试不仅效率低，并且测试质量受限于测试人员的一些情绪和心情。若当一个测试人员带着烦躁情绪来测这些繁杂的大量重复性工作，测试的质量令人担忧。更何况，当这项测试工作涉及到全球化方面的测试时，多语言版本的测试工作导致该测试工作量的成倍增加，这无疑是一项巨大的考验！
* 检查点：用来验证脚本执行结果是否达到预期。可以在录制的过程中建立检查点，也可以在录制完成之后再建立检查点。

# 功能自动化测试实施原则

## 实施原则

功能自动化测试过程中工具不可能完成所有的工作，工具仍然是测试过程中的辅助手段。对于工具主要是解决测试过程中的重复性的工作任务。另外实施自动化的测试，对被测系统也有更高的要求，总结功能自动化测试的实施原则如下：

1. 使用自动化工具测试，要求被测系统开发比较稳定，较少发生功能的变更；
2. 在自动化测试脚本录制前，被测系统的界面相对稳定；
3. 功能测试自动化要求测试数据环境中的测试数据相对充裕，满足多次重复回归测试的要求；
4. 要求被测系统的版本运行比较稳定，较少发生测试中止的情况；
5. 分期分步骤实施，优先选择产品功能比较稳定的系统进行；
6. 完善的、可复用的数据参数、脚本库是一个长期的积累过程。

## 实施功能自动化测试的优缺点

功能的自动化测试与手工测试虽然有很多局限，但是同样有其优势，随着自动化测试技术和工具的发展，对于比较稳定的产品的功能测试中，自动化测试占有越来越重要的地位。使用Sahi可以加快整个测试的过程，在产品的版本发布之后，可以重复使用测试脚本进行测试，具体来说：

自动化测试的优点：

* 提高测试效率，降低测试成本；
* 重复性强的手工劳动独立用自动化实现；
* 快速的回归测试，提高新版本发布的速度和质量；
* 避免人工测试容易犯的错误，如：错误测试，漏测试，多测试等；
* 很容易就实现并发性测试；
* 测试可重用，采用脚本和数据可以很容易实现重用。

自动化测试的缺点：

* 规范的测试管理，测试需求，测试用例；
* 不能创造性发现测试脚本没有设计的缺陷；
* 高质量的测试用例；
* 高素质的自动化测试工程师；
* 对测试环境要求比较严格；
* 测试需求变化可能引起大量的测试用例，自动测试脚本的修改、维护。

# 实施范围和目标

## 实施范围

1. 工具范围：目前考虑Sahi、Excel等工具的使用和集成；持续集成工具暂时先不考虑；
2. 系统范围：定位在测试中心基础测试环境中的系统；
3. 测试阶段的范围：局限在回归测试后期、以及上线后的功能回归测试，目前暂不包括LT、内部测试中的功能测试部分。

## 实施目标

1. 功能自动化测试系统应该能完成集成测试、以及上线后功能的回归测试；
2. 方案目标对有界面和无界面的交易测试都能完成，有界面的交易支持如下方式：
   1. 支持字符终端界面；
   2. 支持B/S的Web界面；
   3. 支持C/S的Windows应用程序界面；
3. 功能自动化测试方案对目前大部分应用系统都可以进行测试；
4. 实现自动化脚本录制、自动化脚本执行、自动化缺陷报告和管理。

## 总体实施策略

1. 首先从目前系统中选择适合自动化测试的项目和系统；
2. 其次确定实施功能自动化测试的阶段和时机；
3. 第三从适合的项目中选择适合自动化测试实施的功能和交易。

具体实施策略参见第6节的实施管理建议。

# 技术方案实施内容

## Sahi的特性和优势：

当提及面向 Web 的自动化测试，相信许多读者会想到或者说使用过 Selenium、Watir等工具，而对于Sahi就可能比较陌生。首先，让我们先来了解下Sahi工具。它是一款印度公司Tyto Software 开发的成熟的开源 Web 自动化测试工具。Sahi简单易用，能良好支持 Ajax 和 Web2.0 技术，同时适用于敏捷和传统的不同测试模式。那么，它与其他非常流行的 Web 自动化测试工具有哪些不同和优势呢？让我们将其与主流自动化测试工具 Selenium 和Watir来进行一番对比，请参考图 1：

**图 1. Sahi与其他工具的对比**



从上图的对比可以看出，Selenium 支持的脚本语言比较丰富，且自带 Selenium IDE 自动录制工具，Watir执行的速度相对其他较快。而Sahi同样具备了自带的录制器，且支持几乎所有浏览器，且对 JS 支持较好，拥有页面等待判断机制，内置 Java 异常报告，支持 Ajax 等优势。

下面，本文将详细介绍一下Sahi的几大优势。

**基于上下文的页面识别机制：**

大多数如 Selenium 等 Web 自动化测试工具或是自动化框架，都采用类似基于 DOM 的定位策略、Xpath定位策略和 id、name、identifier 等页面元素定位策略。

Identifier 定位是最普遍的一种定位方式，当不能识别为其它定位方式后，默认为 identifier 定位。在这种策略下，第一个使用 id 的页面元素将被识别出来，如果没有使用指定 id 的元素，那么将识别第一个名字与指定条件相符的元素。

例如，identifier 识别 username 元素的定位策略：identifier=username

Id 定位是在知道元素具体 id 特征的情况下的一种更精确定位。例如，定位页面元素loginFrom：id=loginFrom

name 定位方式是去识别第一个匹配名称属性的 UI 元素。如果多个元素拥有相同的名称属性，可以使用 value 过滤器来进一步优化您的定位策略。例如，定位页面元素为 username：  
name=username

Xpath定位是在 XML 中定位元素的方法，而 HTML 可以被看作是 XML 的一种实现。XPath扩展了上面 id 和 name 定位方式，提供了绝对路径和相当路径两种查找方式。

绝对路径：html/body/div[1]/div[1]/div[3]/div[1]/form/span/input[1]

相对路径查找：//div[@id='fm']/form/span/input

然而，在实际的情况下，页面元素并非如预期般明确。一些动态页面的 DOM 树常常随着 Web 产品的更新而频繁改变。许多的元素值如 ID、Name 等在代码中并不是必须的，常常会缺省。并且，属性值往往不是唯一对应的，页面中有时会存在相同属性的元素。当缺省 id 值或是Xpath定位失效时，上述这几种查找定位方式往往显得无助和脆弱。

Sahi采用了一种主动查找的机制，它不受限于特定的元素属性。在没有 ID、Name 值的情况下，它可以使用一些如“title，value”等属性，这些都是页面可见的属性，所见即所得。同时，Sahi会通过传入这些可见可识别的属性值，来按照Sahi预设的机制进行查找识别。Sahi允许开发者对每一种元素设置不同属性和特定的查找顺序，包括那些自定义的属性名。所以Sahi相对于其他的 Web 自动化测试工具更灵活更开放。

比如，\_link(“valueName”)用来定位一个定义为“valueName”的 link，这里的valueName并不一定是 value 的属性值，也可以是它的 id、title 等。

前面提到了Sahi主动查找的机制，那么它是如何去查找 DOM 节点下的特定元素的呢？Sahi主要提供了三种基于上下文的元素 API：\_in，\_near 和\_under。

从字面意思上，我们不难理解，\_in 是指在某个 DOM 节点下查找某个元素，这比Xpath的不管是绝对路径或是相对路径查找都来的灵活，不会因为 DOM 树内部结构发生变化而导致路径失效找不到元素的问题。

\_near 是指在某个元素附近查找相应设定规则条件的最近一个元素，这对于一个页面中有多个相同属性值的情况提供了一个很好的解决方式，使查找的范围更精确。

\_under 是指在某个元素下方开始查找，找到符合条件的最近一个元素，一般\_under 都适用在具有相同偏移量的同一列中。下面，我们来看一个例子，加深对Sahi这种基于上下文识别查找机制的理解：

**图 2. 案例网页**



假设，在图 2 显示的 Web 页面的所有 text box 的 name=”q”，那么，Sahi的侦探器通过一些标识来鉴别它们，如(\_textbox("q"), \_textbox("q[1]")和\_textbox("q[2]"))。

如果，我们要定位“Ruby for Rails”那一行的 text box，即\_textbox("q[1]")。传统的元素识别会遇到多个相同属性元素的问题，即使是Xpath的定位方式也会因为在它前面加了一行新的数据而导致Xpath定位失败的情况。

这时Sahi可以通过\_near 这种方式来定位：

\_textbox("q",\_near(\_cell("Ruby for Rails")))

当要定位 check box 时，我们又会发现，“Ruby for Rails”这一行有“Recommend”和“Already own”两个 check box，为了更准确地定位，我们可以结合\_under，例如：\_checkbox(0,\_near(\_cell("Ruby for Rails")),\_under(\_cell("Recommend")))。

如果在整个页面中存在多个这样的表格，我们还可以用\_in 来进一步缩小范围，如：\_checkbox(0,\_near(\_cell("Ruby for Rails")),\_under(\_cell("Recommend")),  
\_in(\_cell("Cost))).

同时值得一提的是，Sahi API 中的 identifier 参数都支持正则表达式，例如，\_div(/name.\*/) 用来识别所有以某种预属性值是 name 开头的 div。

**隐式页面加载响应等待机制：**

现在越来越多的 Web 应用采用 Ajax 的应用技术，来支持网页数据的异步请求响应。当前一般的 Web 自动化测试工具没有一个智能的处理机制，来判断何时可以继续下一个操作。像 Selenium 等自动化测试工具通常会在脚本中人为来设定一个固定的等待时间。但这往往被证实不一定是准确的。实际测试中，人是很难准确判断每一个操作请求需要的合理时间数值。因为，等待时间设置过短，下一步操作在被测应用请求还未返回就执行了，或是由于网络因素使正常的响应时间变长，都可能导致测试过程找不到相应的页面元素，从而导致整个测试用例失败的情况。而如果把时间设置过长，又会造成在一些正常响应过程中的不必要等待的时间浪费，降低了测试效率。

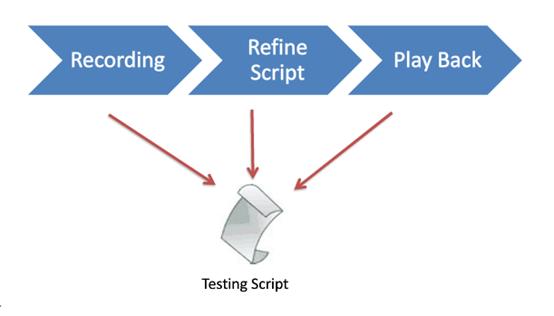
当然，一些测试人员会在自动化测试脚本中加入一些自定义的代码。通过轮询界面上某个指定元素，来判断请求响应是否返回，进而决定继续下一步操作或者是超时。但是，这样的查找过程会导致整个脚本代码变得非常臃肿，加大了开发的成本。更何况，在一个动态的页面找到指定的元素本身就不是一件容易的事。

Sahi内置了智能的页面等待机制，能够自动判断 Ajax 请求是否已经处理完毕，然后继续下一步操作。并且，这一点对于用户是“隐式”的，不需要增加额外的代码。

## Sahi的工作原理：

简单地说，用Sahi实现自动化测试有三步，录制，精炼脚本和回放，如下图：

**图 3. Sahi工作的三个主要过程**

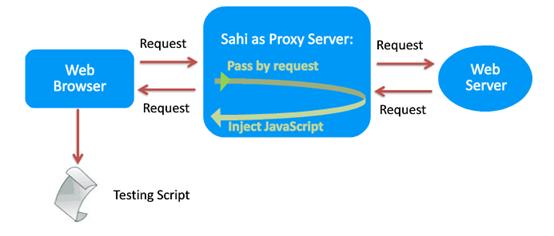


如上图Sahi就是先用其自带的录制工具，把大致的操作过程录制下来，并用Sahi代码记录下整个操作过程。随后，将自动生成的代码进一步的精炼和开发，调用一些外部API 或编写特定代码来实现特定的操作。最后，用Sahi来回放保存好的最终脚本，Sahi就将自动对 Web 应用进行定义好的测试操作。

下面，本文将对这三个过程进行详细说明。

### 第一步：录制

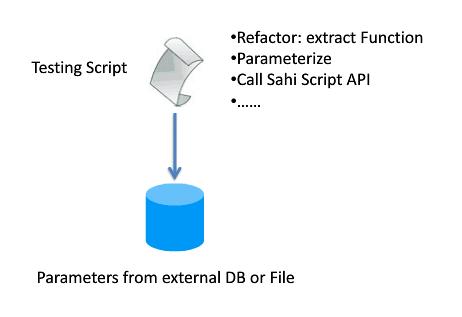
**图 4. Recording 过程的工作原理**



Sahi是通过运行为一个代理服务器，并通过设置浏览器代理为Sahi服务器。这样Sahi的脚本就能够通过 request 请求来注入到 JavaScript 里以访问 Web 页面中的元素。如图，可以很清晰的看到，Sahi就是 Web 浏览器和 Web 服务器之间的一个中间代理。

### 第二步：精炼脚本

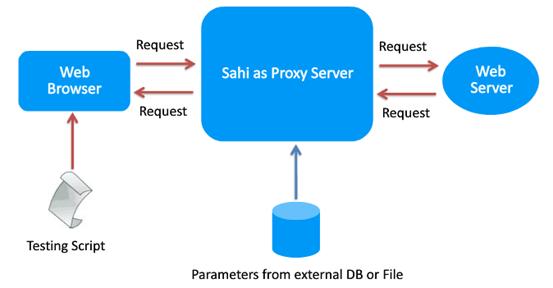
**图 5. Refine Script 过程的工作原理**



录制的脚本都是指定元素并唯一操作的，这时就需要对代码进行重构，抽取出核心的功能块，对其中的元素进行参数化处理，以实现重用。这样的数据可以从外部的 DB 或文件中读取而来。与此同时，也可调用Sahi API 或外部 Java 等 API 实现特定的一些功能。

### 第三步：回放

**图 6. Play back 过程的工作原理**



Sahi运行提炼好的脚本来自动化测试操作，并生成测试报告。

## Sahi的安装部署与配置

Sahi虽然是Tyto公司的产品，但它的下载放在世界上最大的开源软件开发网站SourceForge上，可以通过点击[这里](http://sourceforge.net/projects/sahi/?source=directory)下载。

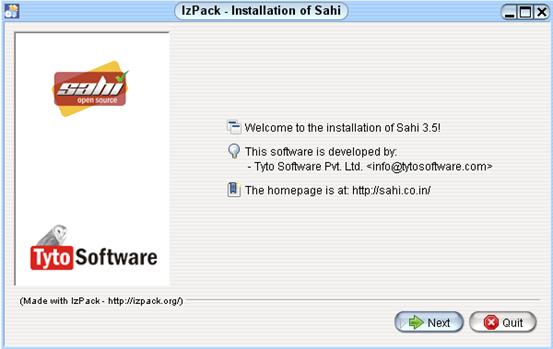
**图 7. Sahi下载**



默认推荐是下载**install\_sahi\_xxx.jar**，这是一个可执行文件，包含了Sahi的安装器和Sahi工具及其源代码。当然您也可以点击上图红框处**“Browse All Files”**来选择历史版本和一些免安装压缩文件。比如，选择只包含Sahi工具的**sahi\_xxx.zip**文件，或者包含了Sahi和源代码的免安装压缩包文件**sahi-src\_xxx.zip**。

一般建议选择推荐的Sahi安装包文件即可，这样可以免去一些设置操作，并可以选择是否安装源代码。双击 jar 文件进行安装，如图：

**图 8. Sahi安装**



安装过程非常简单，待安装完成后双击桌面图标打开Sahi程序。打开程序先会出现一个Sahi Dashboard，它能自动开启Sahi代理服务来启动浏览器，而不需要繁琐的代理服务器设置操作。当然如有需要，您也可以手动修改这些代理设置。

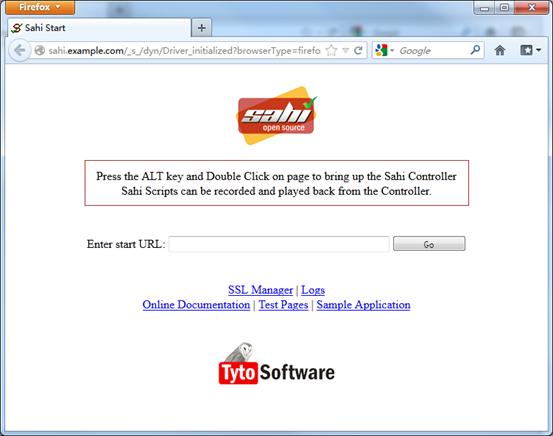
**图 9. Sahi Dashboard 界面**



Sahi会自动去侦探您系统里安装的一些浏览器，并在Sahi Dashboard 上显示出来，如果发现有一些其他的浏览器未被准确侦探出来，您也可以点击下面的“Configure”来进行配置添加进来。

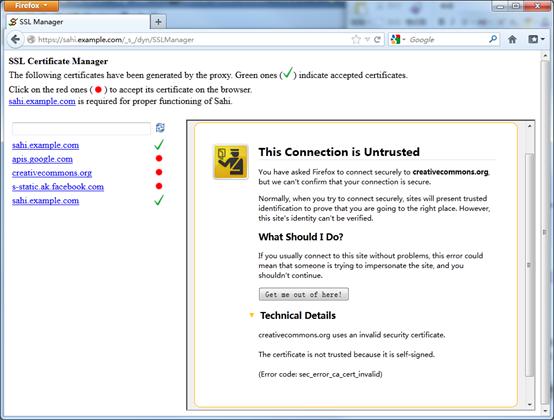
接下来，通过点击Sahi Dashboard 上的浏览器图标按钮来启动相应浏览器。

**图 10. Sahi启动firefox浏览器**



您可以输入起始测试的网页 URL 开始您的测试。如果测试的目标 URL 是 HTTPS 协议的，也可以点击“SSL Manager”来查看和管理 SSL 证书。

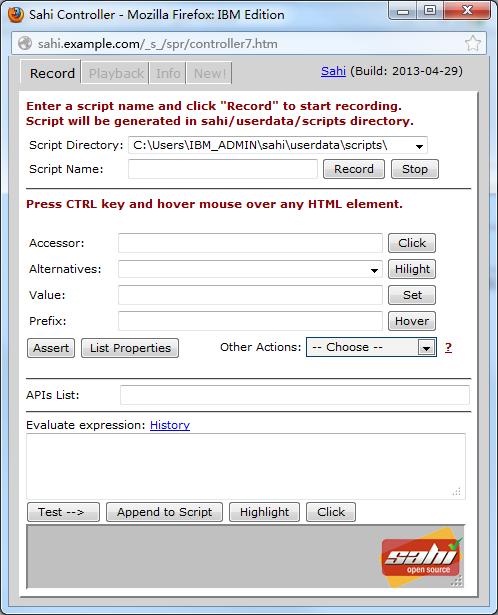
**图 11. Sahi SSL 管理界面**



按住 Alt 键并双击页面，将弹出Sahi控制窗口，如图 12：

这个窗口相当于Sahi的主控台，在这里我们可以来录制和回放Sahi脚本，并编辑和管理脚本信息。

**图 12. Sahi Controller 录制**



在 Record 视图界面，输入一个脚本名称，点击“Record”，这时Sahi录制器便开始工作了。把鼠标移到浏览器上的目标网页上，您的所有操作过程都将被记录下来。您也可以自定义增加一个 Assertion。按住 Ctrl 键，把鼠标移动到目标网页的任意一个 HTML 元素，那么这个Accessor会自动出现在Sahi控制器中。这时，便可以自定制对该元素的操作。常用的操作有“点击”，“高亮”，“赋值等。同时，您可以通过“Append to Script”按钮来加到脚本代码中。录制完成后按“Stop”来结束整个过程。

**图 13. Sahi自动生成脚本精炼**

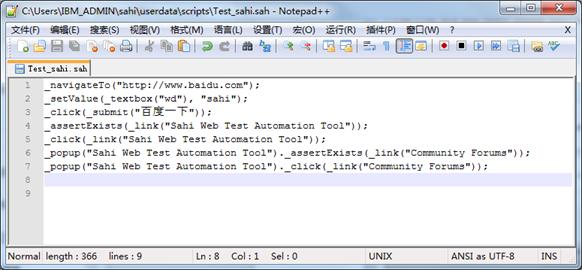


图 13 是一个简单的Sahi自动录制过程得到的Sahi脚本代码。其大致过程为：通过百度搜索“sahi”关键字，校验Sahi官网的 assert 是否存在，点击进入Sahi官网后继续校验assert“Community Forums”，点击进入。通过前一节“Sahi Controller 录制”来完成这个操作过程，那么，您可以在默认目录“C:\Users\IBM\_ADMIN\sahi\userdata\scripts”中找到先前命名为“Test\_sahi”的脚本文件，我们可以将这段代码进行一个精炼和丰富的过程，比如在点击“Community Forums”链接前将它进行高亮操作：

\_popup("Sahi Web Test Automation Tool")\_highlight(\_link("Community Forums"));

或者您想在Sahi脚本代码中调用内置的 Java 类，例如：

functionprintThroughJava(s){

java.lang.System.out.println("Through Java: "+s);}

printThroughJava("Hi there");

“Through Java: Hi there”将在sahi的命令行中输出。

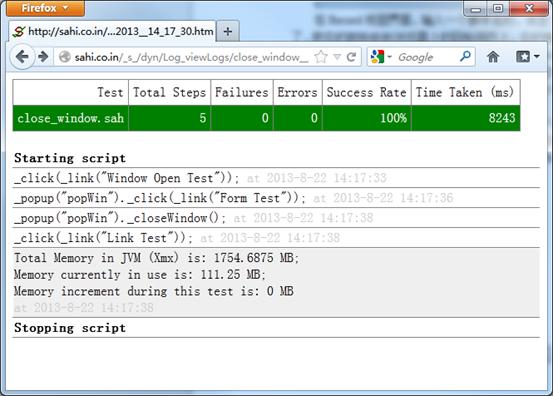
**图 14. Sahi Controller 回放**



回放的时候，只需要在Sahi控制台上切换到“Playback”tab页面，找到脚本存放的路径，下面就有开始、暂停和结束等按钮来进行操作。需要注意的是，开始以前必须给它设置一个“Stat URL”否则无法回放脚本。脚本回放的时候，在“Statements”里可以看到脚本运行的日志，比如操作步骤和一些错误信息等。

通过点击右下角的“View Logs”可以查看详细的Sahi运行日志报告：

**图 15. Sahi日志报告**



由图可见，这样自动录制生成的脚本代码都是Sahi代码，我们可以在实际的 Java 项目中调用这些Sahi代码，以实现重用。其实，我们可以通过打开sahi/config/sahi.properties文件将其中属性设置为controller.mode=java 来实现自动录制脚本的语言为 Java。值得注意的是，改为 Java 语言录制后的Sahi控制器和原来有所不同，它的界面更简洁，功能也更简单一些，没有了自动回放功能。因为，这更多是为了自动生成一些简单的脚本，来提高开发人员的开发效率。

# 实施管理建议

## 实施策略建议

策略有如下建议：

* 分期实施：

实施分为两个阶段，先选择一、两个试点项目实施，试点成熟后再推广到其他的项目中；

* 先易后难：

选择的实施对象尽量是非核心的系统或对目前的其他系统影响不大的项目，这样可以减少实施风险，避免对核心系统或其他系统造成进度等影响；

* 选择稳定的功能：

自动化测试如果对功能不稳定的系统，会造成测试脚本的维护工作量比较大，自动化测试会被异常中止的情况；

* 逐步完善：

自动化测试体系不仅仅包括测试脚本的录制编写和执行，还包括测试数据环境、测试脚本库、测试工具的集成等工作，需要逐步完善整个自动化测试体系。

## 人员配置

在自动化测试实施过程中，需要相关的角色和人员，不同的角色负责相应的职责，具体需要的人员角色如下：

* 测试经理：

制定测试计划；

协调各个小组工作、协调行方资源；

跟踪测试进度；

协商解决测试中的遇到的问题。

* 产品经理：

为自动化测试提供业务指导，数据准备。

* 测试工程师：

搭建测试环境，数据准备；

定义脚本的框架，调试脚本框架；

分析测试结果，优化测试脚本，配合系统优化。

根据框架生成测试脚本，数据参数化，添加检查点，执行测试，记录测试结果。

* 开发人员：

系统优化，测试过程中问题定位。

## 实施计划

| **阶段** | **时间** | **阶段关键任务/里程碑目标** |
| --- | --- | --- |
| 计划准备阶段 |  | 确定项目的目标和范围；  确定项目的实施计划；  确定项目实施的人员需求；  功能自动化测试方案通过评审。 |
| 环境搭建和准备 |  | 搭建自动化测试环境；  熟悉和了解被测系统的功能需求。 |
| 测试需求分析阶段 |  | 分析被测系统需求；  完成《测试需求分析报告》；  分析测试系统的数据需求；  完成测试数据需求，初步完成主要的测试数据。 |
| 测试脚本开发 |  | 完成脚本框架；  完成被测功能的脚本的录制、编写和调试；  完成被测脚本的参数设置和测试数据；  发布测试脚本基线版本。 |
| 测试执行 |  | 测试执行、报告问题。 |
| 测试分析总结 |  | 测试结果；  测试总结。 |

## 交付物

|  |  |
| --- | --- |
| **阶段** | **交付物** |
| 计划准备阶段 | 《功能自动化测试实施方案》（具体方案，针对实施的对象）  《功能自动化测试实施计划》 |
| 环境搭建和准备 | 搭建好的自动化测试环境；  《测试环境搭建操作指南》 |
| 测试需求分析阶段 | 《测试需求分析报告》  《测试数据需求》  准备好的测试数据 |
| 测试脚本开发 | 功能自动化测试脚本文件 |
| 测试执行 | 测试执行记录  测试发现缺陷报告 |
| 测试分析总结 | 测试分析报告 |