

上海泽众软件科技有限公司

泽众 **AutoRunner Process** 自动化测试软件

帮助文档

本文件属上海泽众软件科技有限公司所有，  
未经书面许可，不得以任何形式复印或传播。

## 目录

1 产品介绍.....	1
1.1 公司简介.....	1
1.2 产品简介.....	1
1.2.1 自动测试简介.....	1
1.2.2 AutoRunner 功能测试工具简介.....	3
1.3 系统要求.....	4
2 新手入门.....	4
2.1 产品安装.....	4
2.2 用户界面.....	5
2.2.1 菜单栏.....	5
2.2.2 工具栏.....	9
2.2.3 工作区.....	11
2.3 项目与脚本操作.....	15
2.3.1 项目操作.....	15
2.3.2 脚本操作.....	18
2.4 录制脚本.....	21
2.4.1 Windows 程序脚本录制.....	21
2.4.2 Java 程序脚本录制.....	24
2.4.3 IE 脚本录制.....	25
2.4.4 Flex 程序脚本录制.....	27
2.4.5 Silverlight 程序脚本录制.....	28
2.4.6 谷歌浏览器脚本录制.....	29
2.4.7 Edge 浏览器脚本录制.....	29
2.4.8 火狐浏览器脚本录制.....	30
2.5 回放脚本.....	31
2.6 高级功能.....	38
2.6.1 参数表.....	38
2.6.2 对象比较.....	44
2.6.3 回放设置.....	45
3 高手进阶.....	47
3.1 参数表编辑.....	47
3.2 参数传递.....	48
3.3 函数调用.....	51
3.4 记录击键.....	54
3.5 记录时间间隔.....	54
3.6 扫描 JDK.....	55
3.7 手工添加对象.....	55
3.8 修改权重.....	56
3.9 添加校验点.....	58
3.9.1 校验属性.....	58
3.9.2 校验数据库.....	59
3.9.3 校验消息框.....	60
3.9.4 校验矩形文本.....	61

3.9.5 校验文件文本.....	62
3.9.6 校验 Excel 文件.....	63
3.9.7 校验正则表达式.....	64
3.10 脚本串联.....	65
3.11 脚本参数化.....	66
4 AutoRunner Process.....	68
4.1 产品介绍.....	68
4.2 产品安装.....	68
4.3 用户界面.....	69
4.3.1 菜单栏.....	69
4.3.2 工具栏.....	71
4.3.3 工作区.....	71
4.4 操作流程.....	74
4.4.1 新建 AutoRunner Process 项目.....	74
4.4.2 新建 AutoRunner Process 流程.....	76
4.4.3 导入脚本.....	77
4.4.4 执行流程脚本.....	78
5 脚本命令.....	81
5.1 activeTitle.....	81
5.2 addClassPath.....	81
5.3 beginTime.....	82
5.4 callScript.....	82
5.5 captureScreen.....	82
5.6 checkDatabase.....	82
5.7 checkExcelCell.....	83
5.8 checkExcelColumn.....	83
5.9 checkExcelRow.....	84
5.10 checkExcelWhole.....	84
5.11 checkFileText.....	85
5.12 checkImage.....	85
5.13 checkMessageBox.....	85
5.14 checkProperty.....	86
5.15 checkRectText.....	86
5.16 checkRegex.....	86
5.17 clickControl.....	87
5.18 clickScreen.....	87
5.19 closeLowLog.....	87
5.20 collapse.....	88
5.21 controlExist.....	88
5.22 dragControl.....	88
5.23 dragScreen.....	89
5.24 endTime.....	89
5.25 expand.....	90
5.26 getCaptureScreenWhenError.....	90

---

5.27	getDatabase	91
5.28	getExcelCell	91
5.29	getExcelColumn	91
5.30	getExcelRow	92
5.31	getExcelWhole	92
5.32	getFileText	92
5.33	getFrom	93
5.34	getMessageBox	93
5.35	getParameterDataList	93
5.36	getProperty	94
5.37	getRectText	94
5.38	getStopWhenError	94
5.39	getSynchronizationTime	95
5.40	getTimeSpan	95
5.41	getY	95
5.42	ieControl	96
5.43	ieWindow	97
5.44	inputDown	97
5.45	inputKey	97
5.46	inputString	98
5.47	inputUp	98
5.48	loadObjectElement	98
5.49	menu	99
5.50	menuExist	99
5.51	modifyDatabase	99
5.52	mouseDown	100
5.53	mouseMove	100
5.54	mouseUp	101
5.55	ObjectElement	101
5.56	openLowLog	102
5.57	pressKey	102
5.58	pressString	103
5.59	putExcelWhole	103
5.60	putInto	104
5.61	putParameterDataList	105
5.62	regexTitle	106
5.63	select	106
5.64	selectState	107
5.65	setCaptureScreenWhenError	108
5.66	setState	108
5.67	setStopWhenError	109
5.68	setSynchronizationTime	109
5.69	setTimeSpan	109
5.70	setValue	109

5.71 setX.....	110
5.72 setY.....	110
5.73 sleep.....	110
5.74 source.....	110
5.75 startApplication.....	111
5.76 stopRunning.....	111
5.77 table.....	111
5.78 window.....	112
5.79 windowExist.....	112
6 常见问题.....	113
6.1 Flex 程序录制不出脚本.....	113
6.2 IE 对象回放不通过.....	113
6.3 IE 脚本录制时某次操作没有被记录.....	113
6.4 IE 录制不出脚本.....	114
6.5 setValue 命令无效.....	116
6.6 Silverlight 程序录制不出脚本.....	116
6.7 Vista、Win7、Win2008 中注意事项.....	117
6.8 安装出错.....	117
6.9 表格控件的录制与回放.....	117
6.10 不能识别对象.....	118
6.11 回放不停止或回放时间过长.....	118
6.12 回放对象不在对象库中.....	118
6.13 回放时提示找不到对象.....	118
6.14 键盘键码表.....	119
6.15 录制 Qt 程序.....	120
6.16 密码框获取不到值.....	121
6.17 内嵌网页不能录制脚本.....	122
6.18 校验矩形区域文本命令结果有误.....	122
6.19 循环参数表未执行.....	122
6.20 AR 录好脚本之后点击执行没有反应。.....	122
6.21 AR 如何进行两个对象内属性数据的比较。.....	123
6.22 计算机如何录制 10 以上的数据如何参数化。.....	124
6.23 AR 针对 excel 表格下拉列表的抓取失败的原因。.....	124
6.24 AR 安装失败。.....	125
6.25 AR 使用时报-24 的错误.....	125
6.26 AR 使用时报出-13 的错误。.....	126
6.27 AR 使用时报出-30 的错误。.....	126
6.28 AR 的 lic 报出-1 的错。.....	126
6.29 AR 无法正常安装，报出未关闭某些进程。.....	126
6.30 使用 AR 时抓取不到对象，提示找不到对象。.....	126
6.31 AR 的 LICENCE 报-7 错误。.....	127
6.32 AR 录制出现空白。.....	127
6.33 使用云版的 AR，报出-26 的错误。.....	130
6.34 AR 在使用过程中报出-22 的错误。.....	130

---

6.35 AR4.0 录制谷歌浏览器。.....	130
6.36 AR 中 try+catch 的用法。.....	131
6.37 AR 无法录 制下拉框。.....	132
6.38 AR 无法录制密码。.....	133
6.39 AR 修改参数化。.....	133
6.40 AR 支持什么浏览器。.....	133
6.41 自动记录脚本文件后，尚需要较多的手动编辑来完成。.....	133
6.42 AR 找不到对象。.....	134
7 服务支持.....	134

# 1 产品介绍

## 1.1 公司简介

上海泽众软件科技有限公司成立于 2003 年 10 月，公司的宗旨是通过向软件开发企业提供最优质的测试工具软件。测试解决方案和测试咨询服务，帮助国内用户提高软件的品质。我们一直致力于软件测试工具领域的探索。于 2004 年推出了终端自动测试引擎（Terminal AutoRunner，简称 TAR），在 2005 年，推出了第一个测试管理工具 TestCenter，TestCenter 与 TAR 一起构成了泽众软件自动化测试解决方案。在 2006 年推出了自动功能测试工具：AutoRunner，在 2012 年推出了自动性能测试工具：PerformanceRunner。AutoRunner、PerformanceRunner 与 TestCenter 构成了泽众软件的另外一个完整的自动化测试解决方案。经过不懈的努力，PerformanceRunner 自动测试引擎、TestCenter 得到了用户的认可和应用，表现出强大的生命力。PerformanceRunner 测试工具、TestCenter 测试管理工具已经成为测试工具软件市场的重要选择。

## 1.2 产品简介

### 1.2.1 自动测试简介

自动测试过程就是通过模拟人工操作，完成对被测试系统的输入，并且对输出进行检验的过程。自动测试是由软件代替人工操作，对被测试系统的发出指令，模拟操作，完成自动测试过程。

#### ● 测试脚本

自动测试，就是使用一个程序来测试另一个程序（被测试的应用系统）功能的正确性。如果用来测试的程序本身非常复杂，也需要被测试，或者编写困难，那么自动测试就失去了意义。因此，用来测试另外一个程序的程序往往是非常简单的，我们把这个程序称为“测试脚本”。测试脚本通常在测试工具的 IDE 里执行，并且获得 IDE 的支持。

#### ● 自动记录

当我们编写测试脚本的时候，往往发现编写脚本本身是很困难的：了解脚本

的语法、了解测试过程、把测试过程转换为测试脚本语句。自动记录，就是通过记录一个操作过程来自动获得测试脚本的过程。通过自动记录，我们就能够得到一个完善的脚本，通过修改这个脚本，我们得到更通用的测试脚本。

### ● 同步点

在执行测试脚本的时候，测试脚本语句的操作对象是 GUI 的对象。测试脚本通过这个对象的属性（如：名称、位置、winclass、disable 等）来确定哪个对象是我们需要操作的对象。这个查找对象的过程如果失败，意味着：第一，应用系统的响应比较慢，需要等待一段时间再进行一次定位；第二，该对象不存在。这个查找、定位对象的过程，我们称为同步点。AutoRunner 的同步点都是隐含方式的：在操作对象的时候进行自动同步，如果同步失败会停止执行后续脚本或继续下一条脚本命令（由 setStopWhenError 命令控制），可以用 setSynchronizationTime 命令设置同步时间。

### ● 检查点

测试的目的是检查数据是否正确。在测试的过程中，我们需要检查某次请求的响应数据是否符合预期。这个检查的位置和条件，我们称为检查点。在 PerformanceRunner 中，使用 check（“objectname”，“property”，“期望值”）来作为检查点的脚本语句，它检查对象 objectname 的属性 property 是否和期望值一致。可以使用检查点来检验响应数据的各个部分，如 header 字段的各项属性，body 字段的内容。

### ● 循环参数表与数据驱动

测试脚本是针对一个测试过程的。一个测试过程往往需要众多的数据来测试。通过自动录制得到的脚本，所有的输入数据都是常数，是固定的。如果需要使用一个测试脚本测试多组数据，就需要对脚本进行参数化，把固定的常数修改为来自数据源变量。这个过程我们称为参数化。采用了参数化的脚本，我们称为数据驱动的模式。使用 PerformanceRunner 完成自动测试：通过录制的方式自动生成测试脚本，不需要用户通过编写测试脚本来创建；通过检查点向导来创建检查点，只面向业务，不需要手工修改脚本；自动的参数化和数据驱动支持，一步到位的创建数据驱动脚本；



## 1.2.2 AutoRunner 功能测试工具简介

AutoRunner 是自动化的功能测试工具，功能测试的目标是根据 GUI 的界面或者报表来检查软件的实际功能是否和需求定义的功能相一致。

AutoRunner 自动测试工具适用于功能测试、回归测试、系统测试、疲劳测试、组合测试、每日构建测试等，可以提高测试效率，降低测试人工成本，帮助用户找被测对象的缺陷，特别是对于一些通过手工测试很难发现的缺陷。

### AutoRunner 可以进行

- Windows 类型对象测试，一般为用 C++/Delphi/VB/C#等技术开发的桌面程序。
- IE 网页对象测试，一般性的网站，比如大的门户类网站。
- Java 对象测试，一般为用 AWT/Swing/SWT 等技术开发的桌面程序。
- Flex 对象测试，一般为用 Adobe 公司的 FlashBuilder 开发工具开发的 Flex 网页程序。
- Silverlight 对象测试，一般为用微软公司的 Visual Studio 开发工具开发的 Silverlight 网页程序。
- WPF 对象测试，一般为用微软公司的 Visual Studio 开发工具开发的 WPF 桌面程序。
- QT 对象测试，一般为基于诺基亚 QT 库开发的桌面程序。

### AutoRunner 特点

- 使用 BeanShell 语言作为脚本语言，使脚本更少，更易于理解。BeanShell 语法自身也兼容 Java 语法。
- 采用关键字提醒、关键字高亮的技术，提高脚本编写的效率。
- 提供了强大的脚本编辑功能。
- 支持同步点。
- 支持各种需求的校验。包括对对象属性、数据库、文件文本、Excel 表格、正则表达式、消息框文本、矩形区域文本等的校验。
- 支持参数化，同时支持数据驱动的参数化。
- 支持测试过程的错误提示功能。
- 允许用户在某个时刻从被测试系统中获取对象各种的信息，例如：一个对

话框上的按钮的名字等属性信息。

- 通过设置对象的识别权重，可以在各种情况下有效识别对象。
- AutoRunner4.3.5 新增了许多命令函数，有利于测试人员进行各种功能测试，熟练掌握这些命令函数，能够让测试人员编写出更简练、更高效的测试脚本。

## 1.3 系统要求

在安装本软件之前请确认系统配置符合以下条件：

- 操作系统要求：Windows 7/10；
- IE 浏览器要求：IE9、IE11；
- 内存要求：不少于 128M；
- 磁盘空间要求：不少于 150M 剩余磁盘空间；

## 2 新手入门

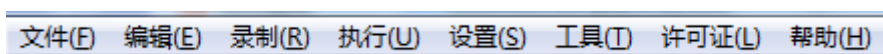
### 2.1 产品安装

- 进入本公司网站 [www.spasvo.com](http://www.spasvo.com) 注册账户。
- 注册之后登录即可下载 AutoRunner。
- 双击安装图示，按照提示安装完成，在安装的过程中由于本软件要录制网页脚本，因而加入了必需的网页插件，在安装插件时有些杀毒软件会出现拦截提示，这是正常现象，放行即可安装，如果禁止则不能正常录制网页脚本。
- AutoRunner 的试用期为 15 天，如果想长期使用需要申请 license，申请 license 的过程：打开软件，点击菜单【许可证】->【申请许可证】，按工具提示进行申请。
- 将生成的 req 文件上传到公司网站上，我们会通过用户注册时的邮箱将 license 发送至您邮箱。
- 收到 license 文件后在软件中点击菜单【许可证】->【导入许可证】即可。

## 2.2 用户界面

(Integrated Development Environment 简称 IDE) 软件是用于程序开发环境的应用程序，一般包括代码编辑器、编译器、调试器和图形用户界面工具，也就是集成了代码编写功能、分析功能、编译功能、Debug 功能等一体化的开发软件套。所有具备这一特性的软件或者软件套(组)都可以叫做 IDE。如微软的 Visual Studio 系列，Borland 的 C++ Builder、Delphi 系列等。

### 2.2.1 菜单栏



AutoRunner4.3.5 中的菜单栏如上图所示，主菜单包含文件、编辑、录制、执行、设置、工具、许可证、帮助等菜单项，下面对每一项做一个简介。

#### 1、文件菜单

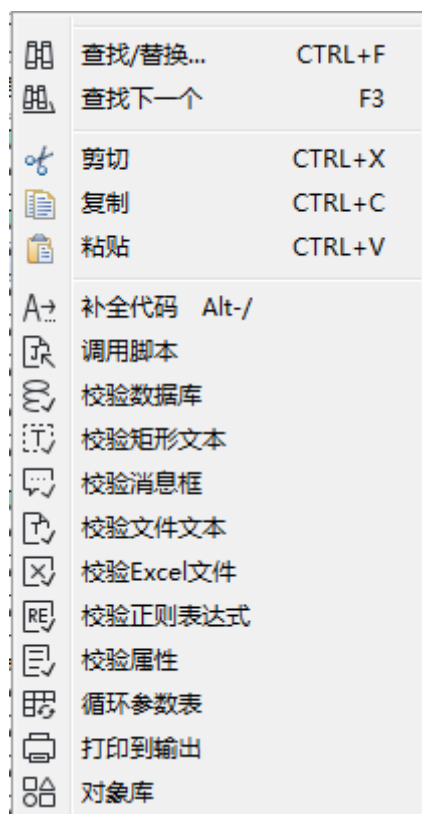


如上所示，所有对脚本的管理操作都可以在文件菜单下完成，包括对脚本的新建，导入，保存，另存为，关闭，改变工作空间，最近打开，退出等等。

- 新建：可选择新建项目或者新建脚本；
- 导入：可选择导入项目或者导入脚本（导入的项目/脚本存在工作空间但不在系统列表内显示时）；
- 保存：保存脚本内容；
- 另存为：将选择的脚本另存为；
- 关闭：关闭当前选中的脚本；

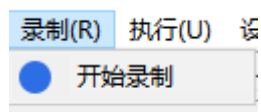
- 关闭全部：关闭所有打开的脚本文件；
- 改变工作空间：修改 AR 的工作空间路径；
- 最近打开：显示最近打开过的项目脚本名称；

## 2、编辑菜单



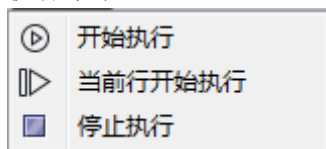
如上所示，所有对脚本的编辑操作都可以在编辑菜单下完成，包括对脚本的查找，替换，剪切，复制，粘贴，循环参数表，对象库，以及对各种需求的校验。

## 3、录制菜单



如上图所示，录制菜单比较简单，只有一个“开始录制”菜单项，用来启动脚本录制功能。

## 4、执行菜单

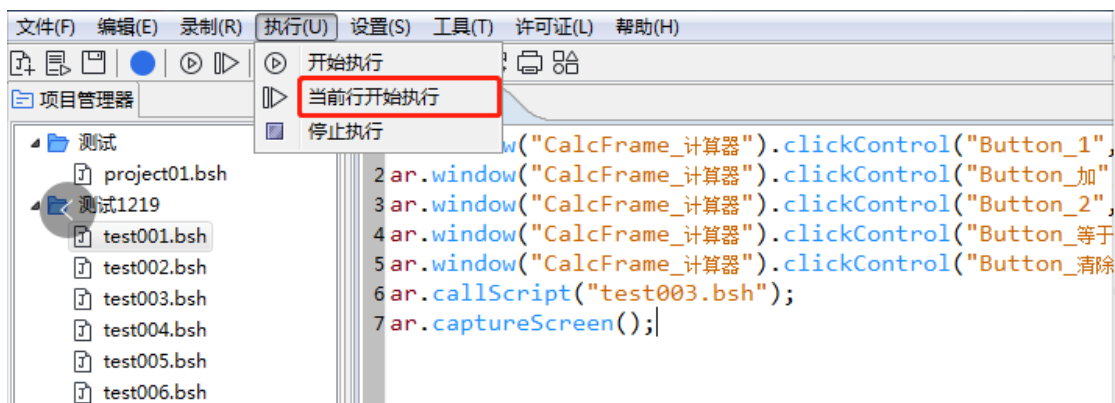


如上图所示，执行菜单包含三个菜单项，开始执行菜单启动回放脚本功能，在回放过程中如果时间比较长或是遇到问题需要提前关闭回放。当前行开始执行菜单指定的当前行启动回放脚本功能，同样，在回放过程中如果时间比较长或是

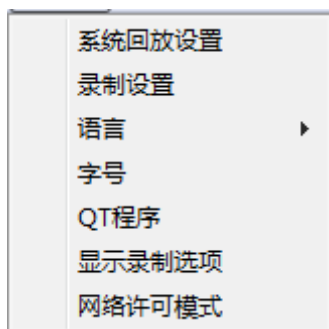
遇到问题需要提前关闭回放。可以点击停止执行以提前终止。

## 5、当前行开始执行

可以通过执行菜单的当前行开始执行功能，实现从脚本编辑器光标所在行开始进行回放操作。如下：

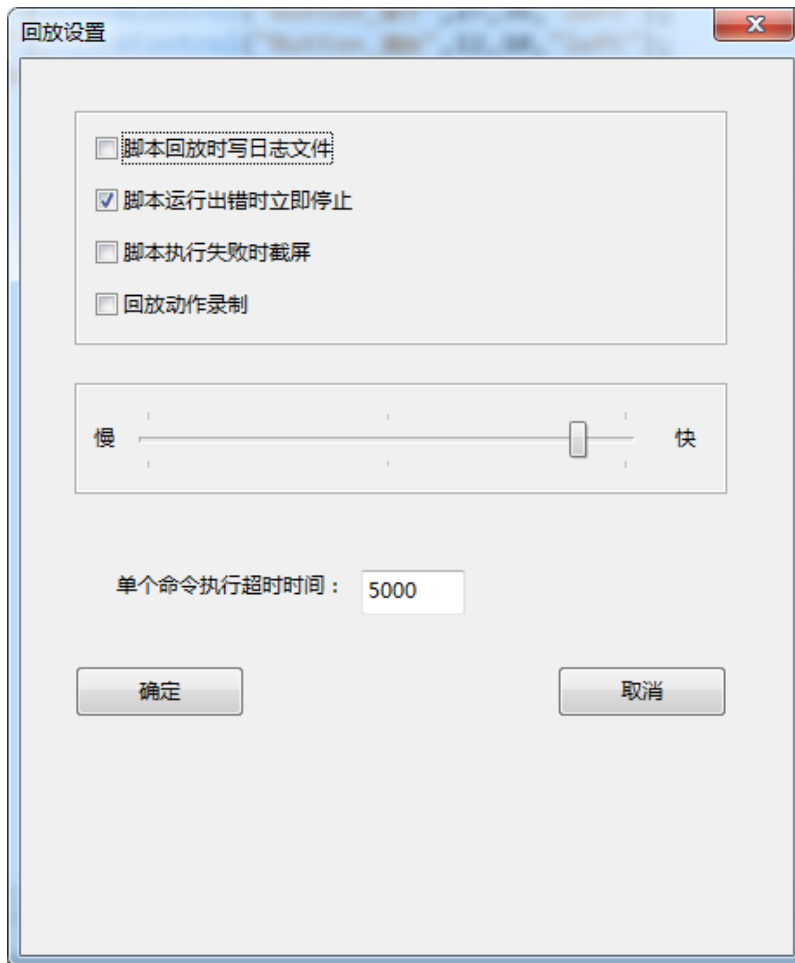


## 6、设置菜单



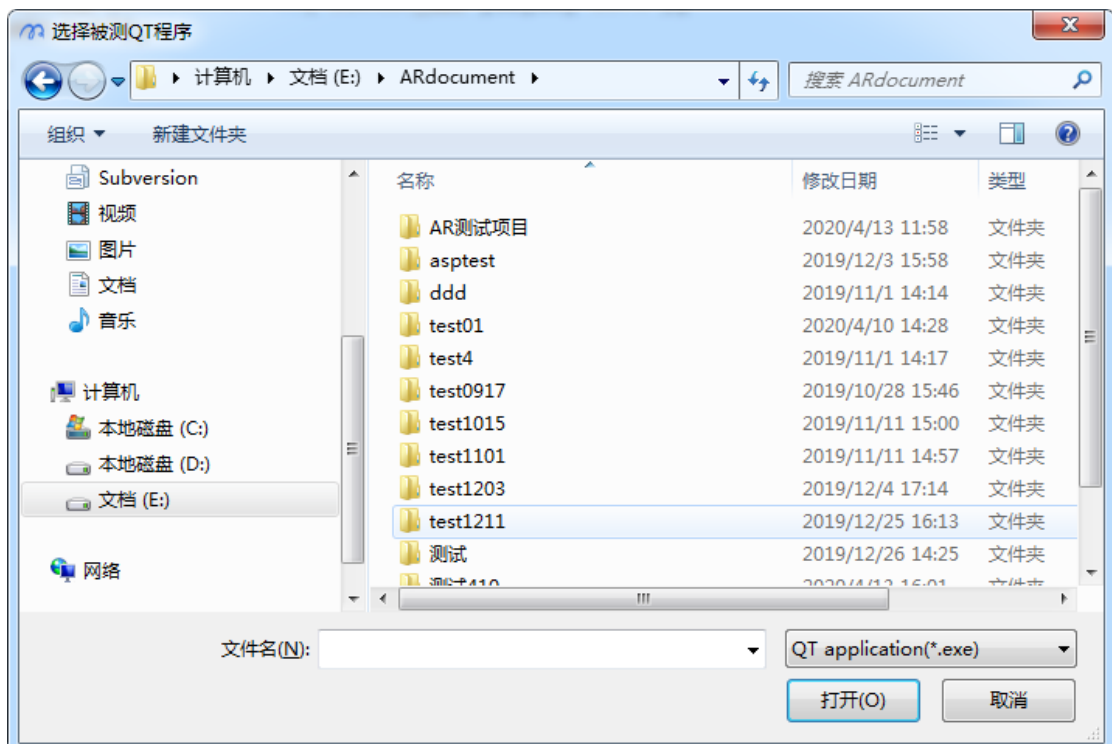
如上图所示，设置菜单中包含了系统回放设置、录制设置、语言设置、字号、QT 程序、显示录制选项及网络许可模式等的设置，软件现在支持简体中文和英文和台湾繁体设置，在程序初次启动时，会根据系统默认语言选择初始语言类型和字号，当用户手动选择语言类别和字号时，软件再次启动才会生效。本软件自带 JRE1.6，如果用户电脑上在安装了 AutoRunner 后又安装了其他的 JDK 版本，则可以点击此按钮，将 AutoRunner 所需要的相应文件写入 JDK 中，省去了用户重装 AutoRunner 的麻烦。

关于系统设置选项：



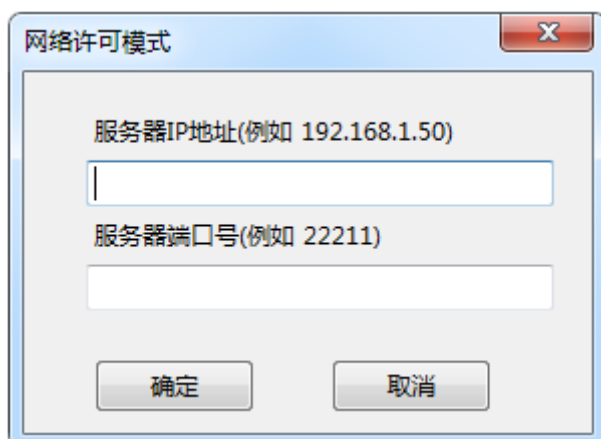
回放设置，可设置某些属性，如：设置播放速度，单个命令执行超时时间等。

关于 QT 程序设置选项：



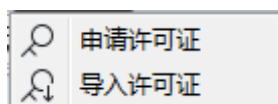
则可设置选择需要被测的 QT 程序。

关于网络许可模式选项设置：



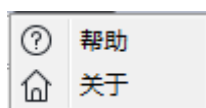
可通过对服务器 IP 地址和服务器端口号的设置来设置网络许可模式。

## 7、许可证菜单



如上图所示，许可证菜单可以方便用户在软件中完成许可证的生成和导入工作，由于免费的许可证试用期为 15 天，当试用期过后再次使用软件后会提示 LIC 过期的提示，此时可以点击“申请许可证”菜单项，根据需要产生 req 文件，获得我公司为您配置的 LIC 文件后，可以点击“导入许可证”菜单项将其导入即可。

## 8、帮助菜单

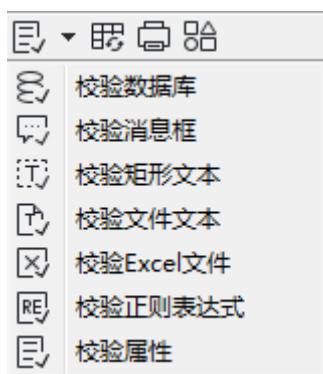


如上图所示，帮助菜单为您提供软件使用帮助和公司产品介绍。

## 2.2.2 工具栏



● 如上图所示，工具栏共有 14 个按钮，其中按钮 11 校验属性下包含以下模块



下面简单介绍各个按钮其功能。

按钮 1: 新建脚本, 和【文件】→【新建】→【脚本】 菜单功能一样;

按钮 2: 运行项目录制程序, 和【文件】→【运行录制程序】 菜单功能一样;

按钮 3: 保存改动脚本 (快捷键 Ctrl+S), 和【文件】→【保存】 菜单功能一样;

按钮 4: 录制脚本, 和【录制】→【开始录制】 菜单功能一样;

按钮 5: 回放脚本, 和【执行】→【开始执行】 菜单功能一样;

按钮 6: 从指定行开始回放脚本, 和【执行】→【当前行开始执行】 菜单功能一样;

按钮 7: 脚本编辑时用以查找替换 (快捷键 Ctrl+F), 和【编辑】→【查找/替换】 菜单功能一样;

按钮 8: 脚本编辑时用以查找下一匹配点 (快捷键 F3), 和【编辑】→【查找下一个】 菜单功能一样;

按钮 9: 补全代码, 和【编辑】→【补全代码】 菜单功能一样;

按钮 10: 调用脚本, 和【编辑】→【调用脚本】 菜单功能一样;

按钮 11: 校验属性, 和【编辑】→【校验属性】 菜单功能一样, 按钮 11 下包含的各个模块功能如下:

校验数据库, 和【编辑】→【校验数据库】 菜单功能一样;

校验消息框文本, 和【编辑】→【校验消息框】 菜单功能一样;

校验矩形文本, 和【编辑】→【校验矩形文本】 菜单功能一样;

校验文件文本, 和【编辑】→【校验文件】 菜单功能一样;

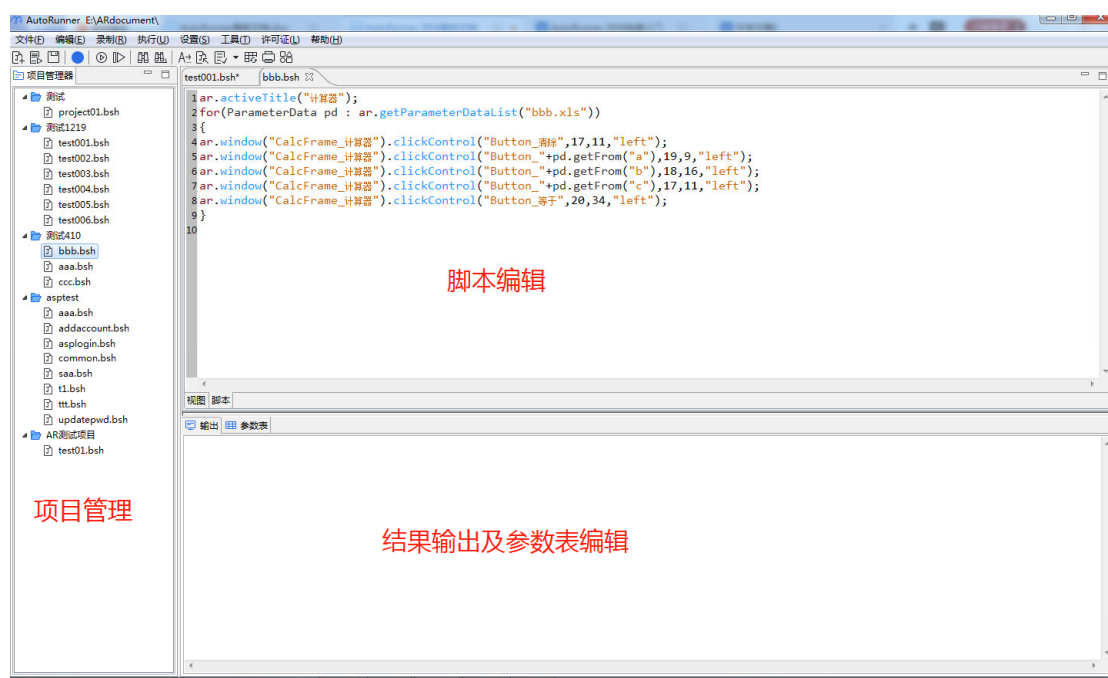
校验 Excel 文件, 和【编辑】→【校验 Excel 文件】 菜单功能一样;



校验正则表达式，和【编辑】→【校验正则表达式】菜单功能一样；  
 按钮 12：循环参数表，和【编辑】→【循环参数表】菜单功能一样；  
 按钮 13：打印到输出，和【编辑】→【打印到输出】菜单功能一样；  
 按钮 14：脚本对象库信息，和【编辑】→【对象库】菜单功能一样；

### 2.2.3 工作区

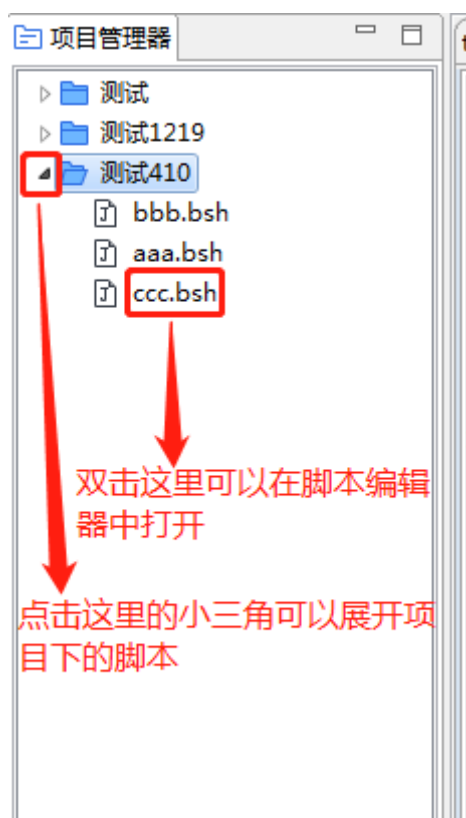
● 项目管理区：创建项目，创建测试脚本，进行项目浏览，切换对象浏览，在 AutoRunner4.3.5 中位于垂直拆分条的左边；脚本编辑区：对测试脚本编辑，在 AutoRunner4.3.5 中位于水平拆分条的上部；结果输出及参数表编辑：测试脚本标准输出，查看测试信息，编辑参数表，在 AutoRunner4.3.5 中位于水平拆分条的下部。



#### ● 【项目管理器】

项目管理器用来显示当前 IDE 中所有的项目，并且显示项目中的脚本。项目管理器中的项目及脚本组织成一个树状结构，每一个项目名称是一个文件夹，其下的脚本都位于此文件夹下。对于每一个节点，如果是项目名称，点击项目名称旁边的小三角可以展开；如果是脚本，则双击可以把这个脚本在编辑器中打开。

树支持鼠标右键菜单，支持删除、增加等操作。

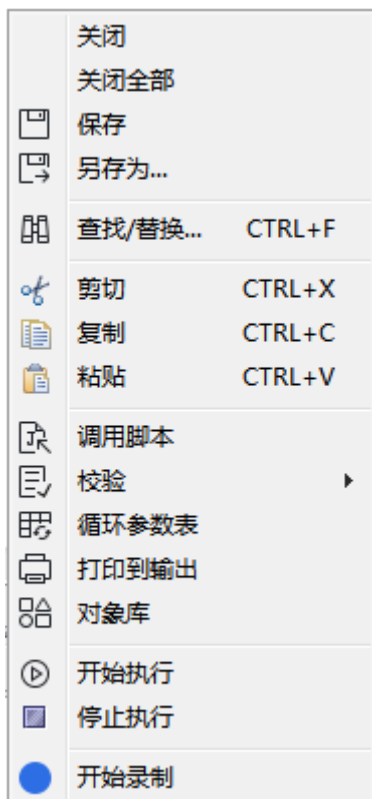


### ● 【脚本编辑器】

双击项目管理器中要编辑的脚本可打开脚本编辑器。编辑器可实现关键字着色，支持多行注释（/\*\*/）及单行注释符（//），支持脚本命令自动补全（快捷键 Alt+/）。如果脚本已被编辑过但还没有保存，在脚本表单中相应的脚本名称后会有一个星号提示符，提示你保存脚本，点击工具栏的保存按钮或是快捷键 Ctrl+S 保存后星号消失。

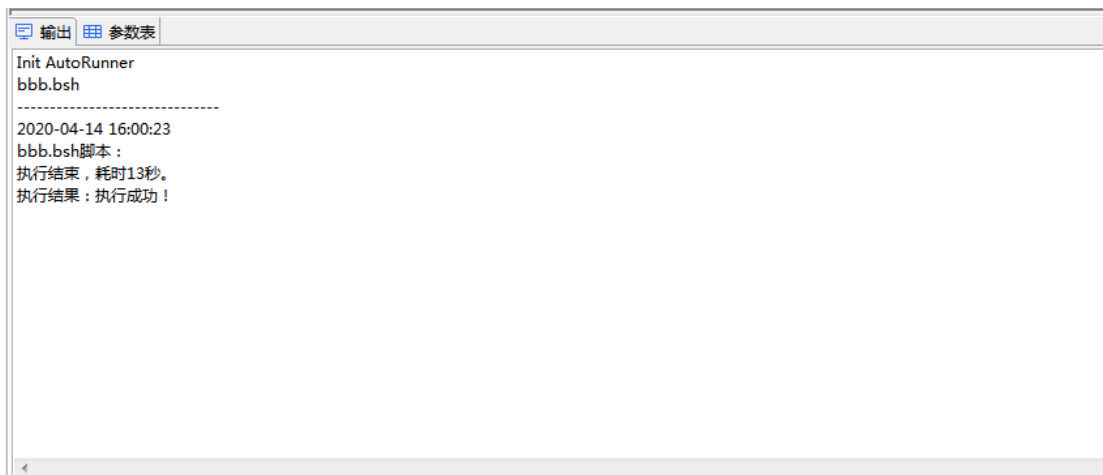


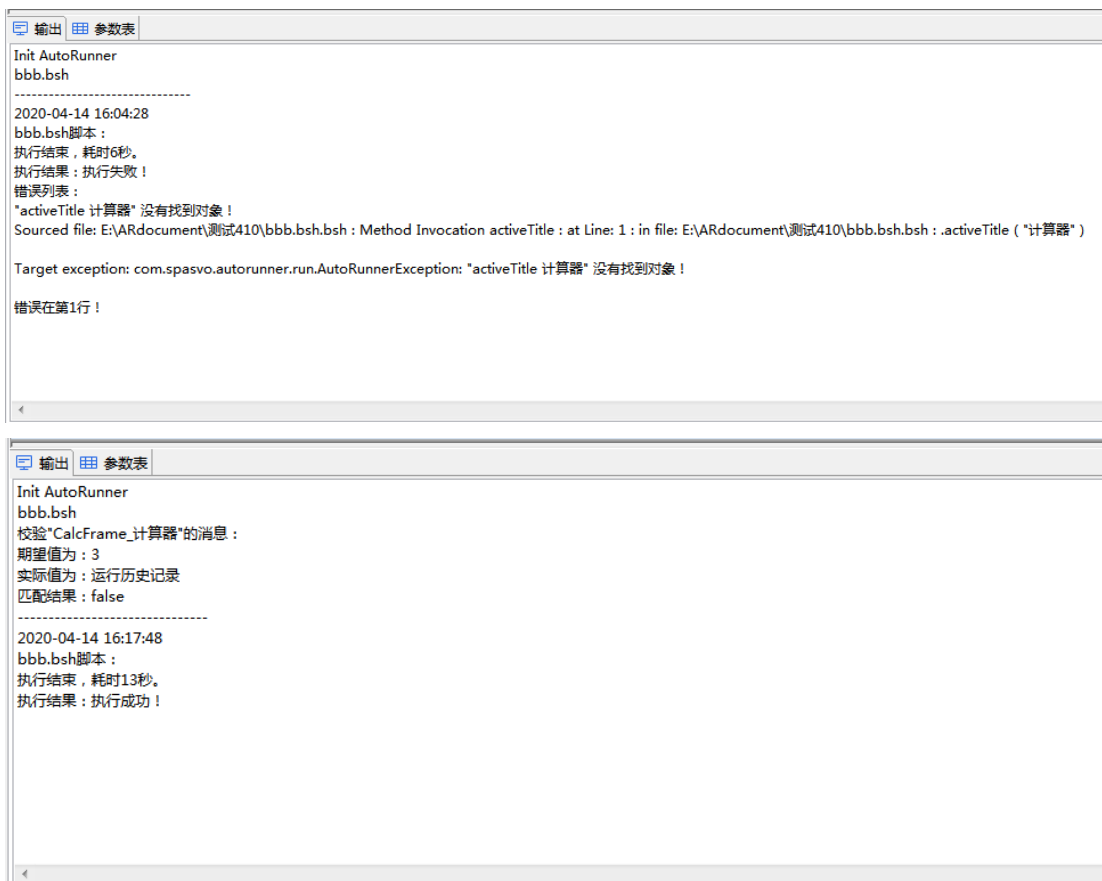
在编辑器中右击鼠标可以弹出如下快捷菜单，包含了一些常用的快捷操作菜单项：



### ● 【运行时刻的错误和输出】

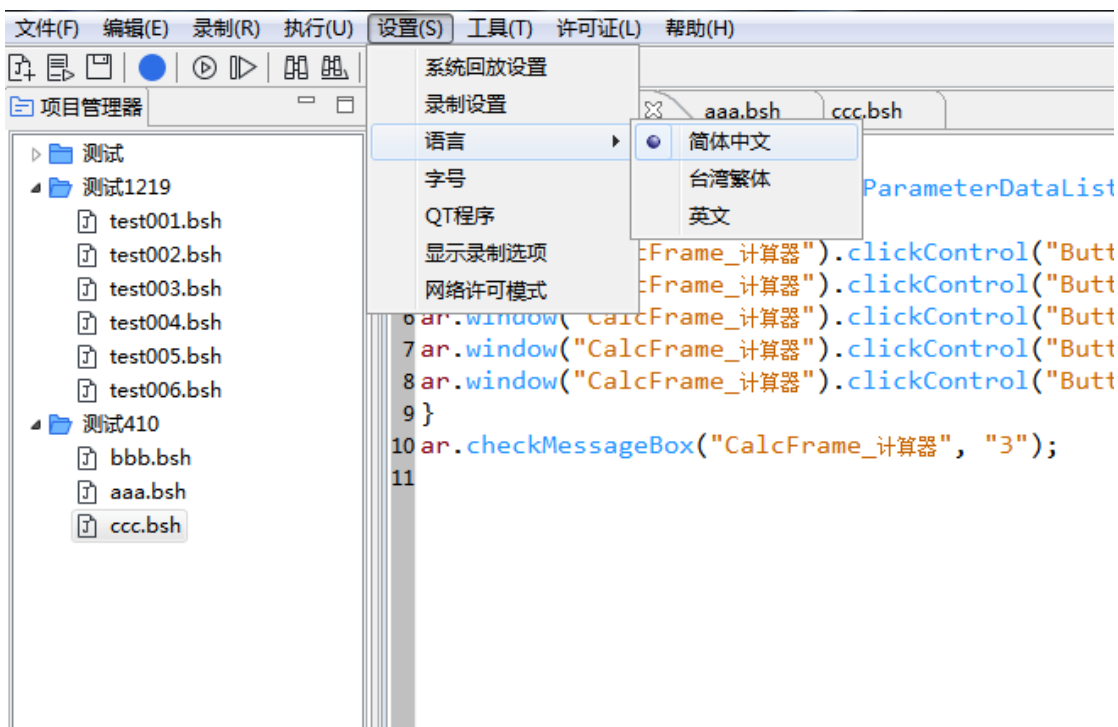
在脚本执行时候显示错误和输出，可以显示脚本中所要求打印输出的语句，可以显示校验结果。若运行报错，则会显示脚本中的报错语句，方便用户找出不能正确执行原因。





● 语言支持

本软件支持：简体中文、台湾繁体、英文。



根据需要设置语言之后，重启本软件生效。

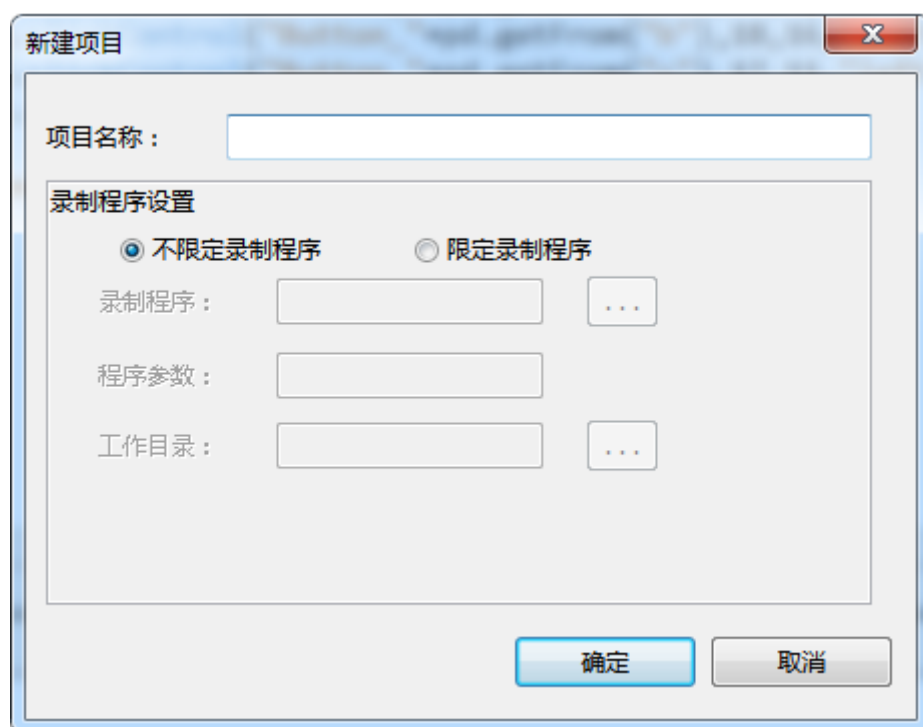
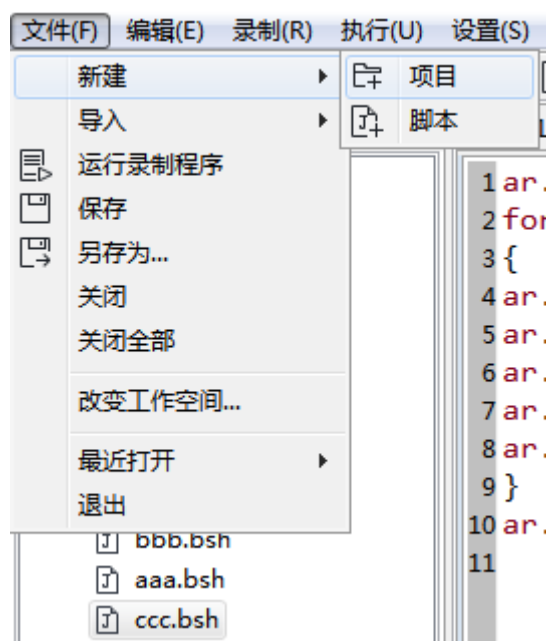
## 2.3 项目与脚本操作

### 2.3.1 项目操作

- 新建项目、导入项目

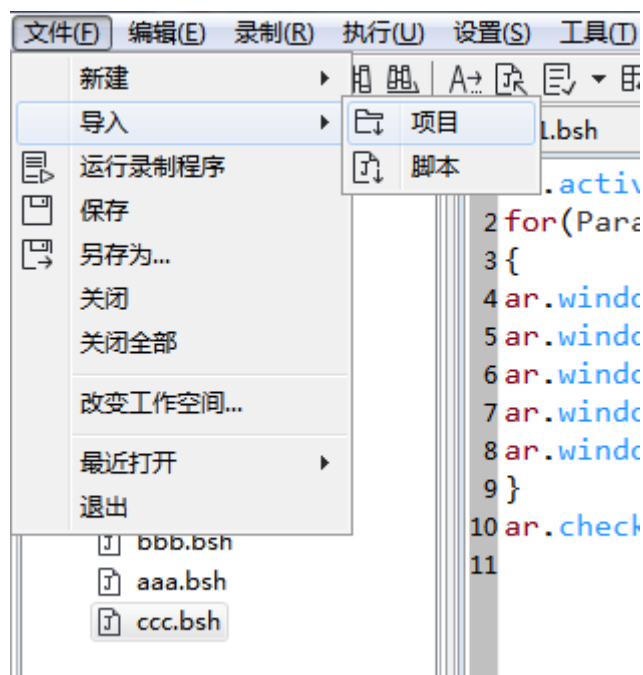
方式一：

新建项目，弹出新建项目对话框

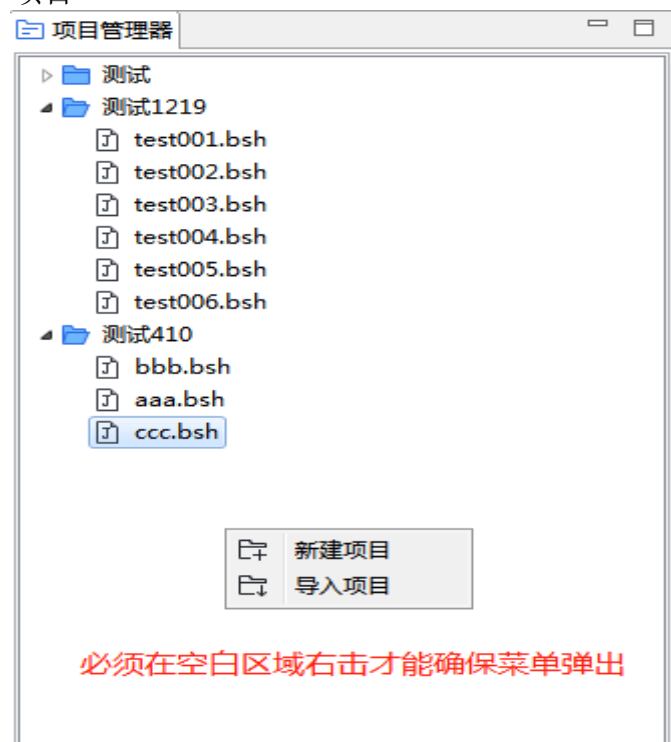


如果项目限定了录制程序，为使限定有效，必须在录制前通过本功能运行录制程序，如果未执行本功能，录制被视为不限定录制程序。注意：限定了录制程序，用户只能对该程序录制，即使该程序运行了另一程序 B，B 程序也是不可录制的。

### 导入项目

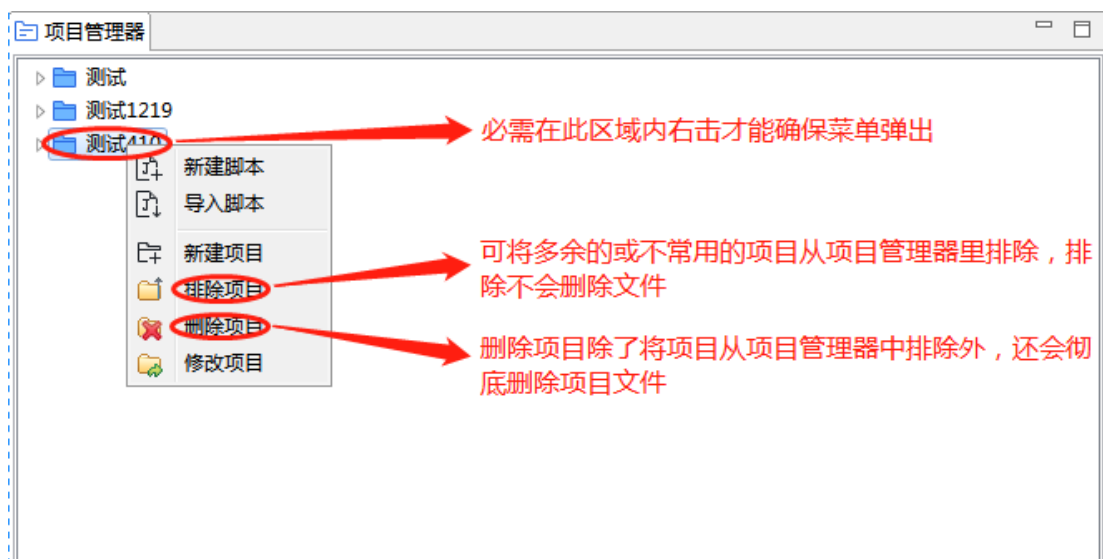


方式二：在项目管理器空白区域点击鼠标右键，在弹出的菜单中选中新建/导入项目



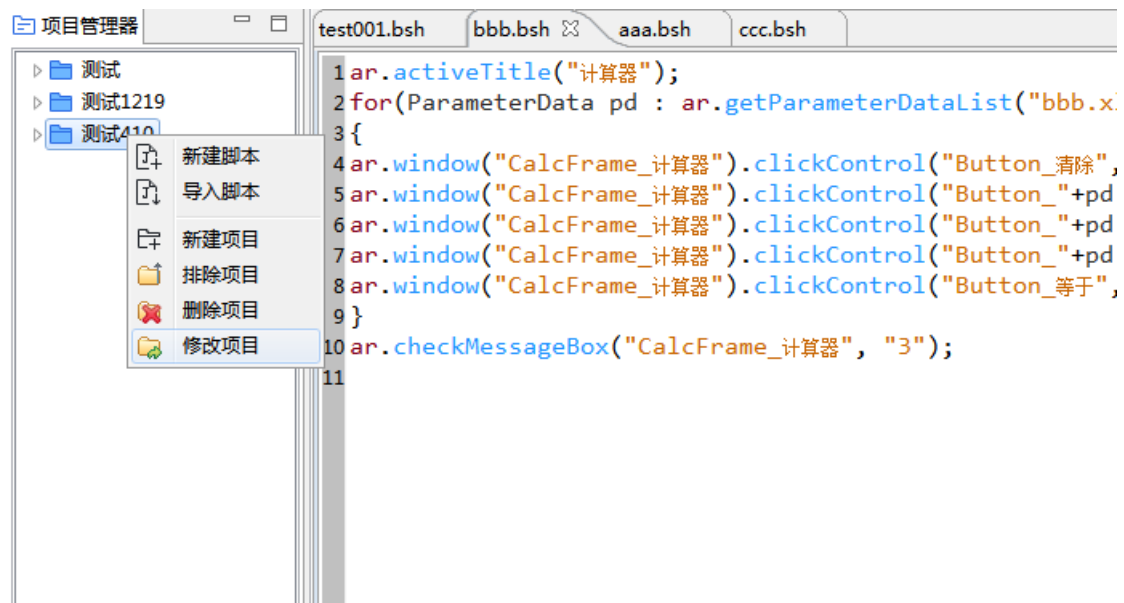
### ● 排除项目、删除项目、修改项目

方法：选中一个项目，鼠标放在项目上点击右键，弹出菜单

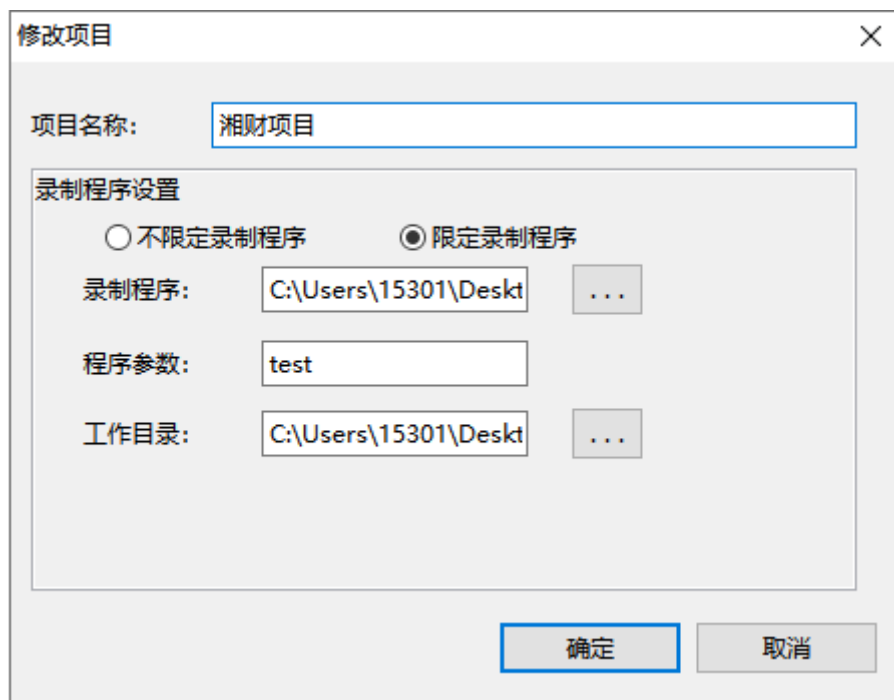


### ● 修改项目

选定你要修改的项目名称，右键菜单修改：



点击修改，可对项目名称重新设置。

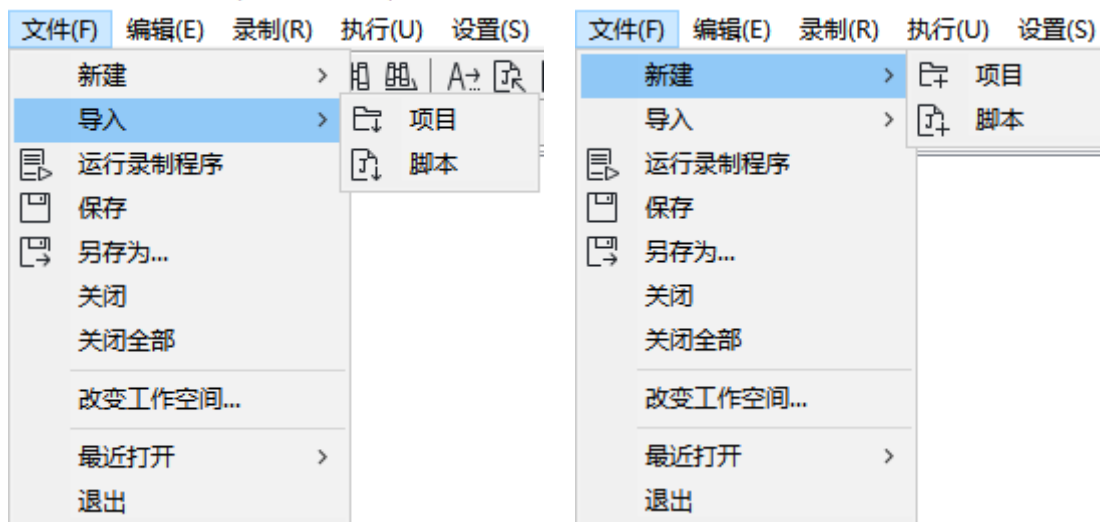


最大的优点是，可以为没有指定录制程序的项目给予指定录制程序，或者给予指定好了录制程序的项目，对录制程序进行重新指定，和不指定操作。

### 2.3.2 脚本操作

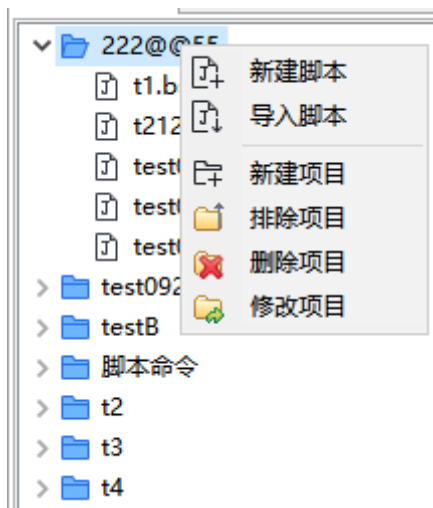
#### ● 新建脚本、导入脚本

方式一：

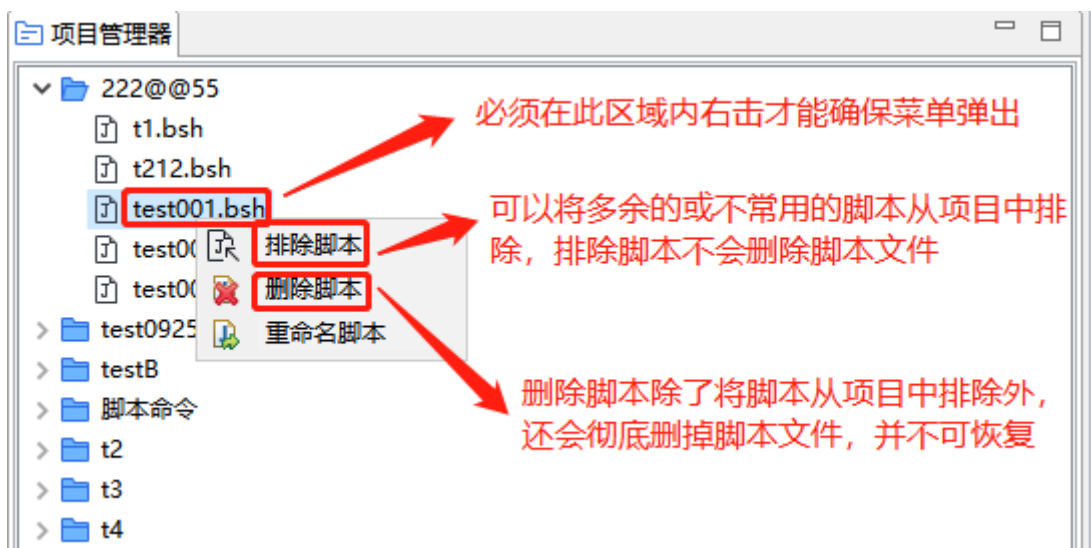


方式二：



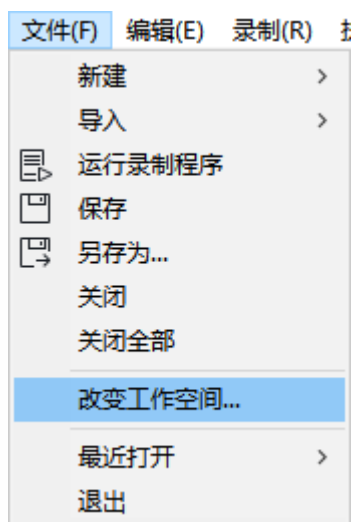


● 排除脚本、删除脚本、重命名脚本

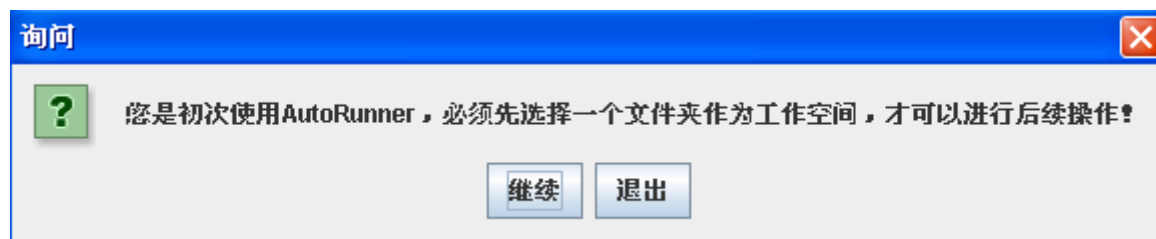


● 改变工作空间

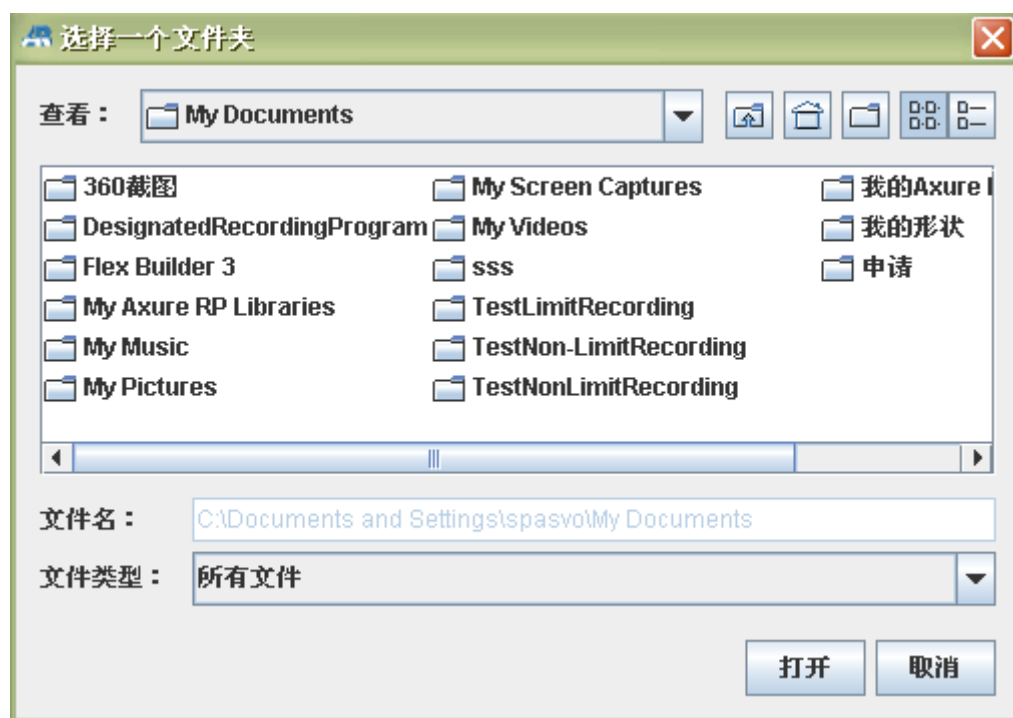
在项目管理器中的所有项目及脚本都在同一个工作空间，如果想打开不在此工作空间的其他项目或脚本，可以通过切换工作空间来实现。操作方式：**【文件】** → **【改变工作空间…】**，见下图。



注意：在切换工作空间时，会清空当前项目管理器中的所有项目，然后才会加载新的工作空间的项目和脚本，不同工作空间中的脚本不会出现在同一项目管理器中；在第一次使用 AutoRunner 时会提示设定一个工作目录，如下图，默认是“我的文档”。



当点击改变工作空间时，弹出如下文件夹选择框：



选择文件夹即可。

选好工作空间之后，如果以后要更改工作空间，必须重新启动 AR，才能生效。

## 2.4 录制脚本

### 2.4.1 Windows 程序脚本录制

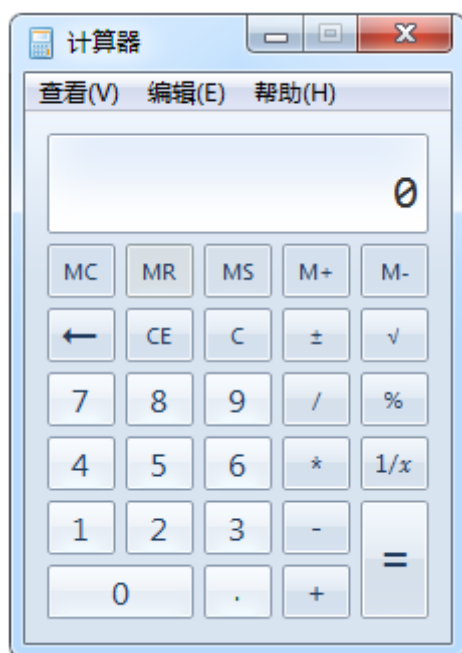
以录制 Windows 中自带的计算器为例，详细的介绍一下录制 Windows 程序脚本的过程。

#### ● 创建脚本

根据前面的【工程与脚本操作】一节所述方法创建一个名为 test1.bsh 的脚本（脚本名可任取），双击脚本打开。

#### ● 录制脚本

先打开要录制的计算器程序（【开始】→【运行】输入 calc 回车即可），如下图所示



点击菜单【录制】→【开始录制】。

录制功能新增：

1、暂停当前录制：



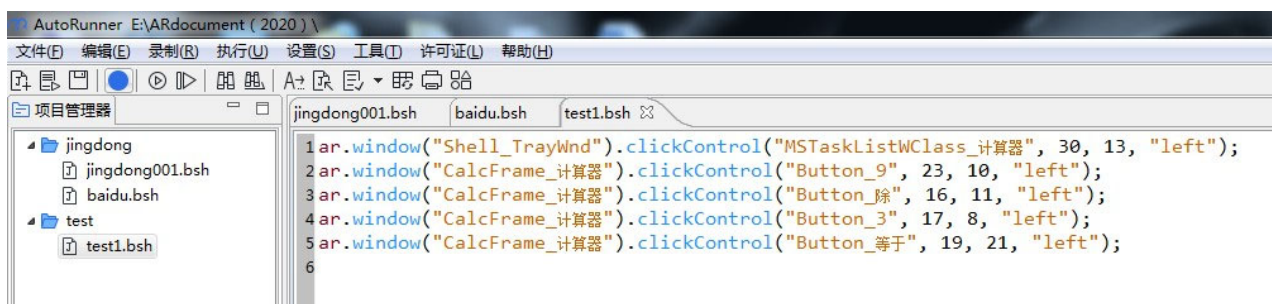
2、继续当前录制:



3、录制完毕，停止录制:

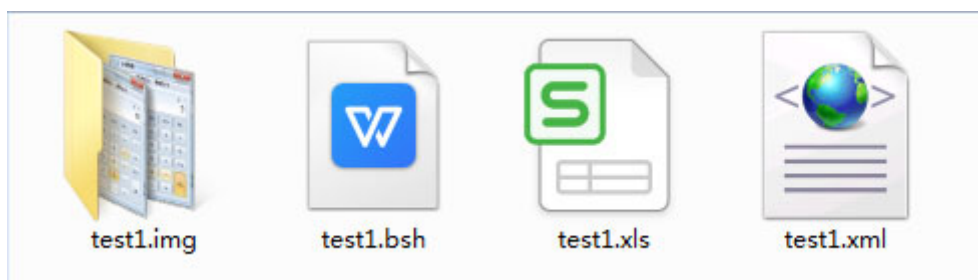


例如，录制一个计算器  $9/3=3$  的录制脚本为,如下图:



### ● 生成文件

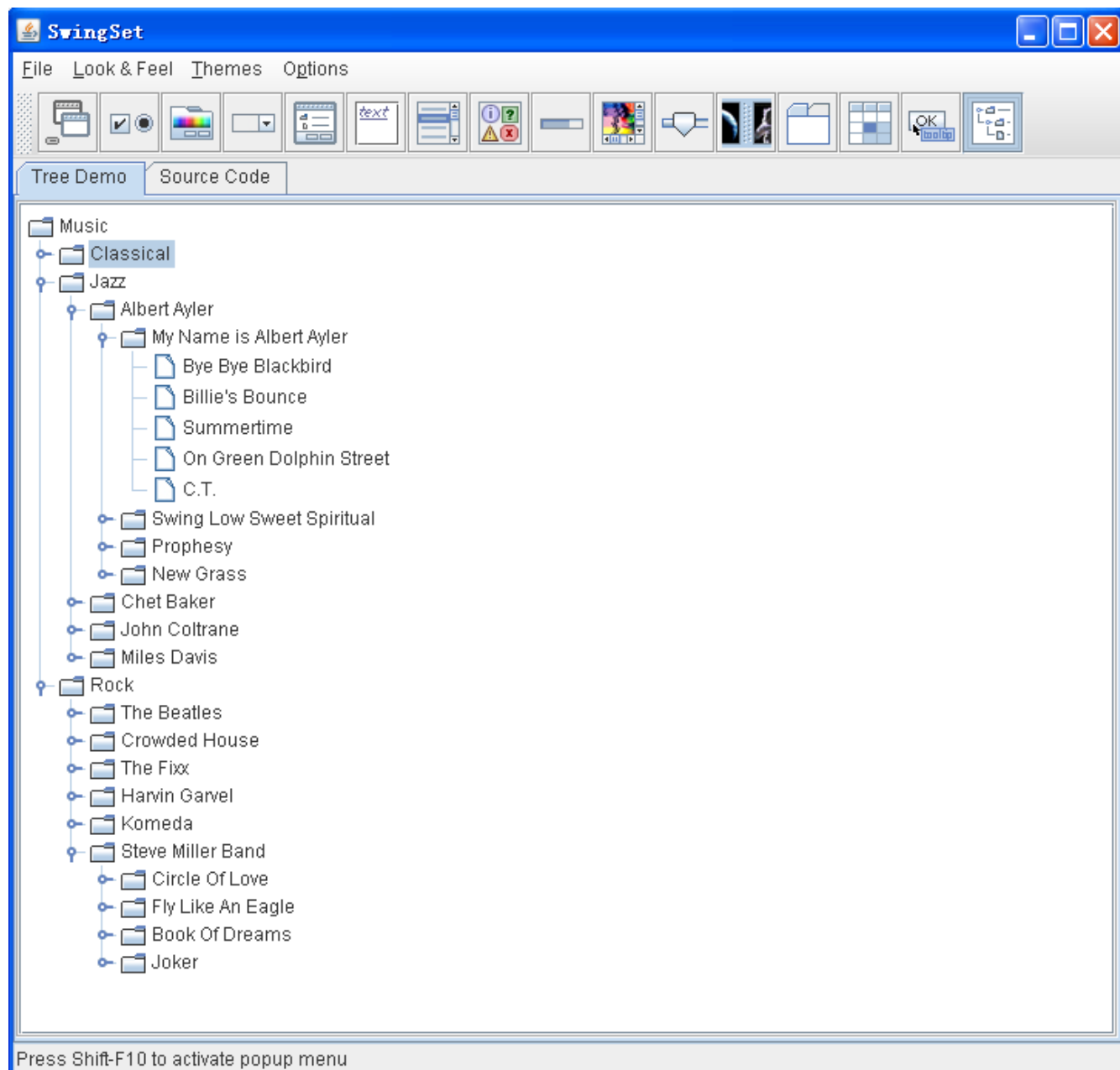
在录制好脚本后，在项目目录下会存在如下几个文件



第一个为脚本文件，保存了脚本编辑器中的脚本；第二个为参数表文件，是一个 excel 表格，所有的参数化数据都将被保存到这里，当然在我们没用到参数化时，此文件中无数据。第三个为对象库文件，是一个 xml 格式，前面我们看到的对象库信息会被保存到这里，对象库可以进行编辑，编辑后也会被保存下来。上面的三个文件都可以在软件中修改，不建议在软件外编辑。

## 2.4.2 Java 程序脚本录制

如果你的电脑装了 sun 公司的 JDK，里面会自带一些 Java 例子程序，我们就以自带的一个 SwingSet2.jar 为例，介绍一下 Java 程序的录制，此程序在你的 JDK 目录下的“demo\jfc”文件夹里。按照录制 Windows 程序脚本的步骤，先建好一个 Java.bsh 脚本，双击 SwingSet2.jar，选择工具栏的最后一项，里面是一个树控件，如下图所示。



随便点击某个树节点，或是展开收起某个树节点，会有如下类似的动作脚本被记录。

```

1 ar.window("SunAwtFrame_SwingSet").select("tree", "Music\rClassical\rBeethoven\rconcertos\rNo. 1 - C");
2 ar.window("SunAwtFrame_SwingSet").collapse("tree", "Music\rRock\rSteve Miller Band\rJoker");
3 ar.window("SunAwtFrame_SwingSet").expand("tree", "Music\rClassical\rMozart");
4

```

第一句中有一个 select 动作，表示我刚点击了这棵树里面的 No. 1-C 节点，节点的层次关系在第二个参数表示出来，层与层之间以\r 分隔，在回放时如果此节点的上一级直至根节点，它们中的某些层或全部层处于收起状态，那么首先会展开收起的层然后再点击一下节点。第二句是一个收起树中节点的动作，第二个参数表示出了要收起的节点，层与层之间以\r 分隔。第三句是一个展开树中节点的动作，第二个参数表示出了要展开的节点，层与层之间以\r 分隔。

Java 对象库元素属性参数如下图所示。



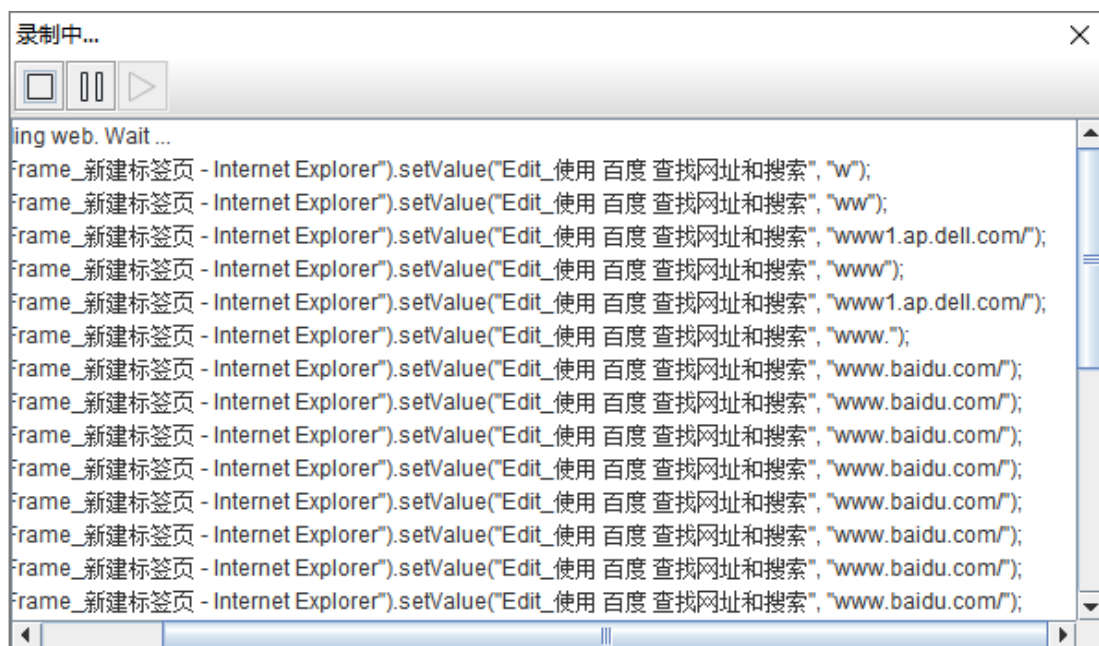
名称	值	权重
category	JAVACONTROL	100
id	0,1,0,0,1,0,0,0,0,0,	100
name		100
role	tree	100
state	enabled,focusable,visible,showing,opaque	0
value	MusicClassicalBeethovenconcertosNo. 1 - C	0
description		0
location	268,303	0
position	9,129	0
width	693	0
height	668	0

如果发现不能正常录制 Java 脚本，请看“常见问题”一节。

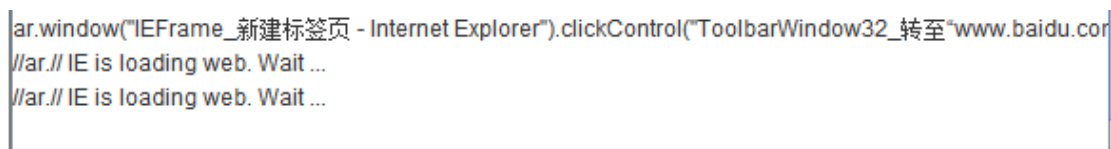
### 2.4.3 IE 脚本录制

以录制百度网页 (<http://www.baidu.com/>) 为例，详细的介绍一下录制 IE 网页脚本的过程。新建一个 IETest1.bsh 脚本，点击录制按钮，在搜索框中我们输入网址 (www.baidu.com)，记录下来的信息将会是如下这个样子。





当点击网页跳转按钮后还会记录下下面几句：



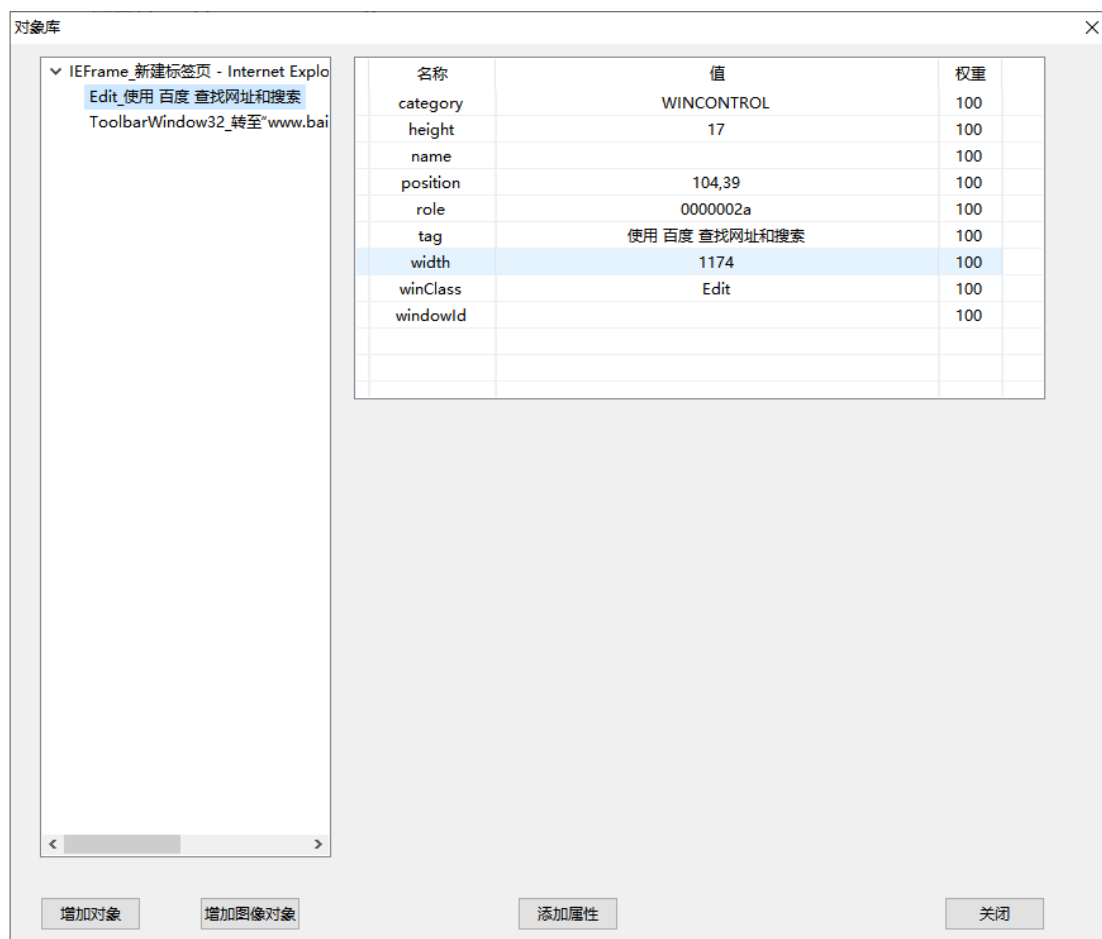
停止录制时我们看到的脚本会是这个样子的：



录制下来的信息很多，但脚本却只有三句话。这是因为在录制的时候，信息框中显示的是我们键盘和鼠标的每一步动作，键盘每按下一个键都会被当成一条动作信息输出，这样录制者就知道此时软件正在监视录制动作。当录制结束后，如果录制脚本中的某一些动作能够合并，软件会将其合并为一句输出，比如这里的 setValue 动作，在脚本中只出现了一条。在录制的过程中，如果网页被切换到一个新窗口或是一个新的网页，比如上面点击搜索按钮后页面跳转，还会打印出两行注释信息（绿色），当要录制新网页上的内容时，应当等待网页加载完成的信息打出后再操作。

IE 对象库元素属性参数如下图所示。





如果发现不能正常录制 IE 脚本，请看“常见问题”一节。

#### 2.4.4 Flex 程序脚本录制

运行 Flex 程序必须安装 Adobe 公司的 Flash 插件，AutoRunner 当前版本支持常用的标准 Flex 控件录制。控件属性和 Windows 控件属性一样，下面就是一段 Flex 程序脚本。

```

1 ar.window("IEFrame_C:\\Control.html - Microsoft Internet Explorer").setValue("ComboBox", "A9");
2 ar.window("IEFrame_C:\\Control.html - Microsoft Internet Explorer").clickControl("PushButton_按钮", 45, 13,
3 ar.window("IEFrame_C:\\Control.html - Microsoft Internet Explorer").setState("RadioButton_单选框 2 of 2", "
4 ar.menu("IEFrame_C:\\Control.html - Microsoft Internet Explorer").clickControl("MenuItem_集团1", 28, 10, "1
5

```

Flex 对象库元素属性参数如下图所示。

名称	值	权重
category	FLEXCONTROL	100
defaultAction	Open	0
description		0
exStyle	00000800	0
height	21	100
help		0
keyboardShort...		0
location	673,419	0
name	集团1	100
position	338,131	100
role	0000000c	100
state	40300000	0
style	56000000	0
value		0
width	56	100
winClass	MenuItem	100

如果发现不能正常录制 Flex 脚本，请看“常见问题”一节。

#### 2.4.5 Silverlight 程序脚本录制

运行 Silverlight 程序必须安装微软的 Silverlight 插件，AutoRunner 当前版本支持常用的标准 Silverlight 控件录制。控件属性和 Windows 控件属性一样，下面就是一段 Silverlight 程序脚本。

```

1 ar.window("IEFrame_SilverlightTest - Microsoft Internet Explorer").clickControl("Button_Button", 4
2 ar.window("IEFrame_SilverlightTest - Microsoft Internet Explorer").setState("CheckBox_CheckBox", "
3 ar.window("IEFrame_SilverlightTest - Microsoft Internet Explorer").select("ComboBox", "3");
4 ar.window("IEFrame_SilverlightTest - Microsoft Internet Explorer").select("ListBox", "List Item 2"

```

Silverlight 对象库元素属性参数如下图所示。

名称	值	权重
category	SILVERLIGHTCONTROL	100
defaultAction		0
description		0
exStyle	00000800	0
height	100	100
help	listBox1	0
keyboardShort...		0
location	42,715	0
name		100
position	40,800	100
role	00000021	100
state	00000000	0
style	56000000	0
value		0
width	120	100
winClass	ListBox	100

如果发现不能正常录制 Silverlight 脚本，请看“常见问题”一节。

### 2.4.6 谷歌浏览器脚本录制

在菜单栏/设置/录制设置中选择谷歌浏览器时，点击录制会自动弹出谷歌浏览器，并且只能录制谷歌浏览器。



### 2.4.7 Edge 浏览器脚本录制

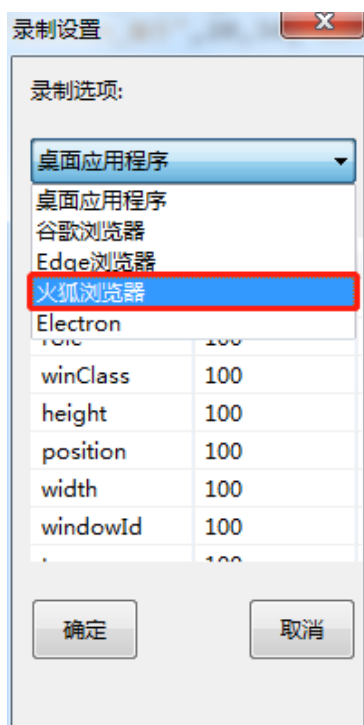
在菜单栏/设置/录制设置中选择Edge浏览器时，点击录制会自动弹出Edge

浏览器，并且只能录制 Edge 浏览器。



### 2.4.8 火狐浏览器脚本录制

在菜单栏/设置/录制设置中选择火狐浏览器时，点击录制会自动弹出火狐浏览器，并且只能录制火狐浏览器。



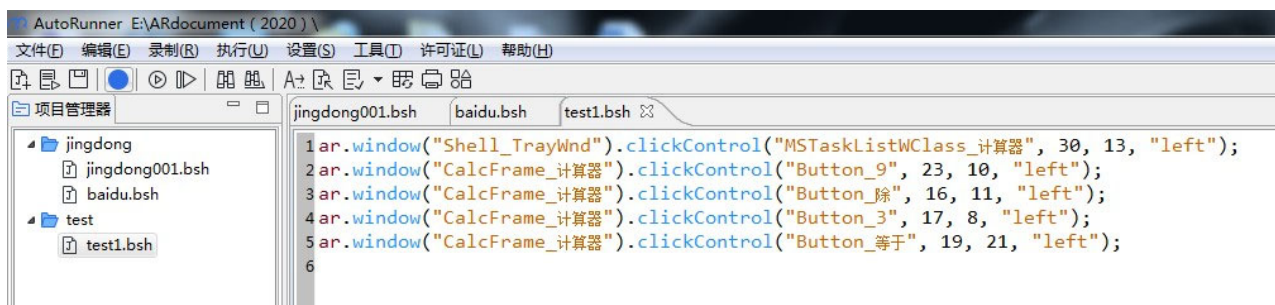
## 2.5 回放脚本


回放脚本的过程,实质是对先前的录入动作的一次重复操作,只是这个过程是根据录入的脚本自动完成的。对于回放来说,不管是回放 Windows 程序脚本还是 Java 程序脚本还是 IE 程序脚本都基本相同。下面就以先前录入的计算器脚本

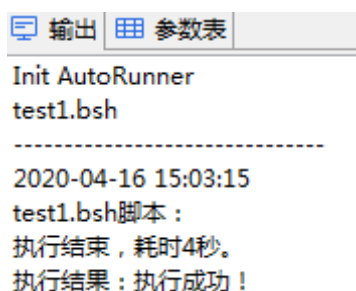
为例，介绍一下回放操作及注意事项。

## ● 回放

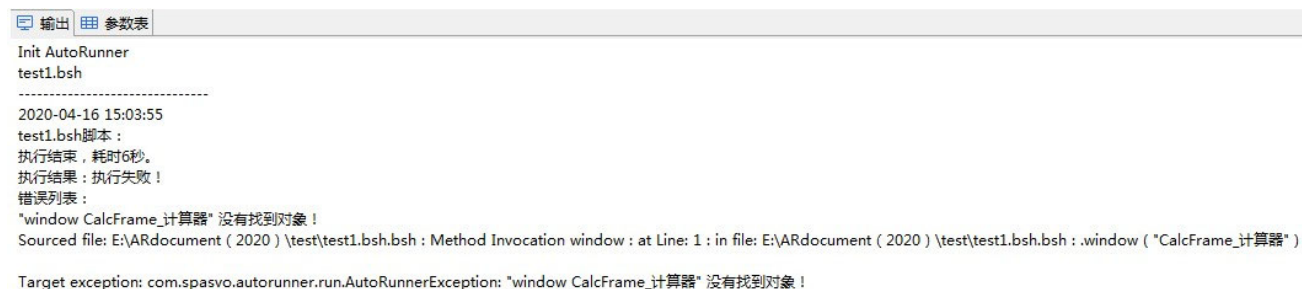
脚本代码如下：



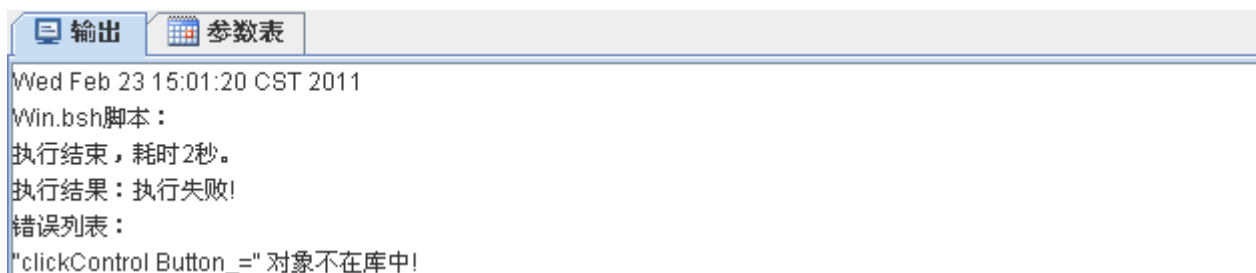
点击菜单【执行】→【开始执行】或者点击工具栏的回放按钮，此时软件进入回放阶段，界面会被隐藏，回放的结果会在输出窗口中显示，如回放成功会有如下信息输出。



如果回放之前将计算器窗口关闭，回放后会有如下信息输出，提示执行 window 动作时，计算器窗口对象没有找到。



如果回放之前在对对象库中将等号的属性信息删除，回放后会有如下信息输出，提示回放 clickControl 动作时，等号对象在对象库中没有发现。

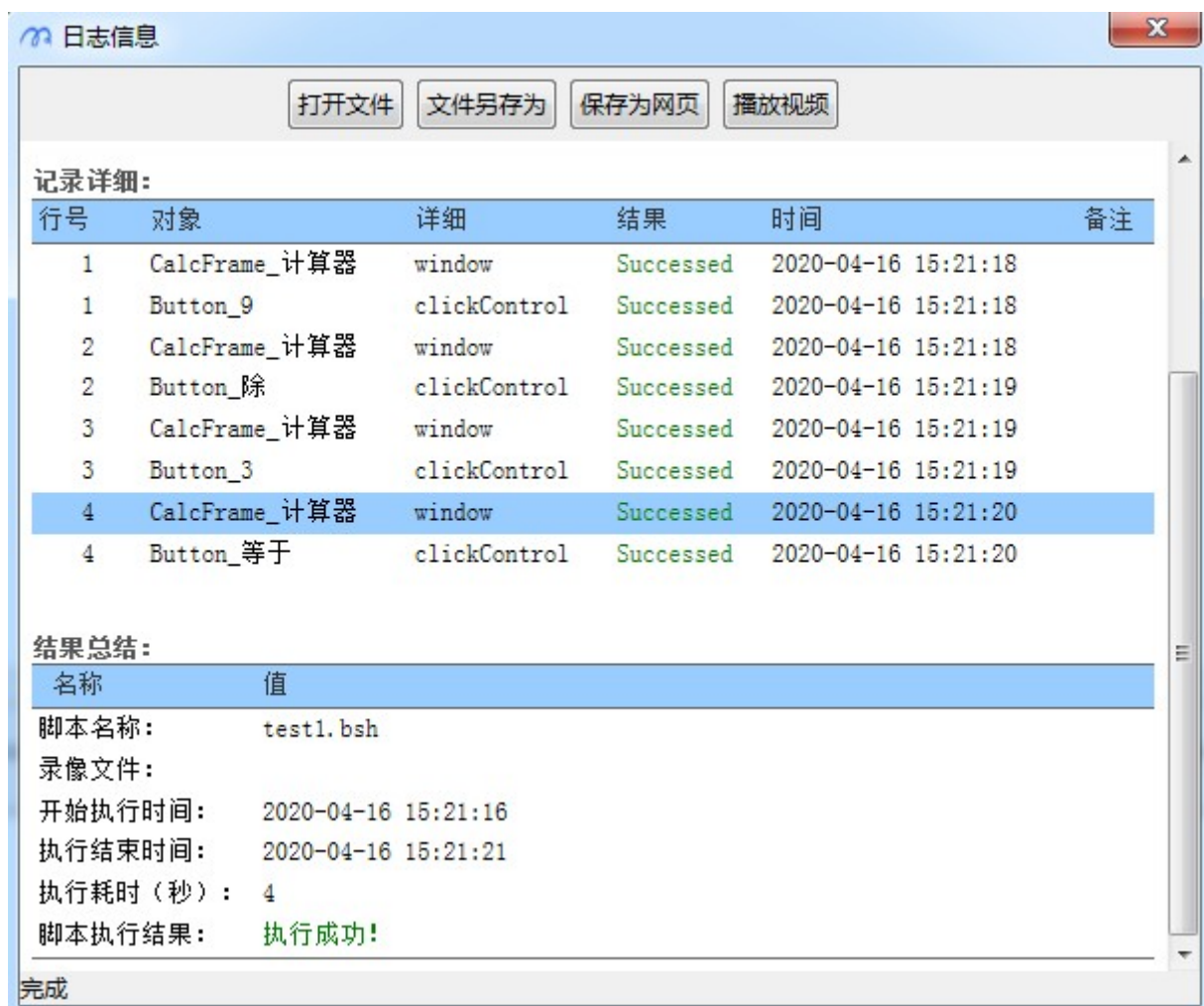




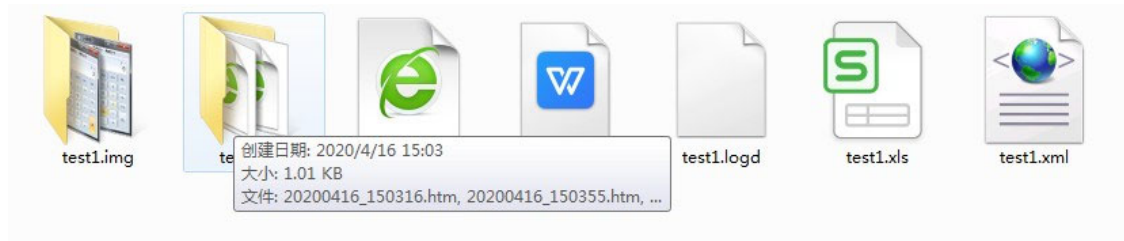
## ● 回放日志查看

当执行完脚本后，系统保存执行结果到工作目录，此时系统自动弹出执行结果查看窗口，在该窗口用户可更友好地查看执行的结果信息。日志中需要体现检查点信息，含检查点名。执行结果查看界面以独立窗口形式展示，窗口以 HTML 的形式用列表显示对象的执行结果。

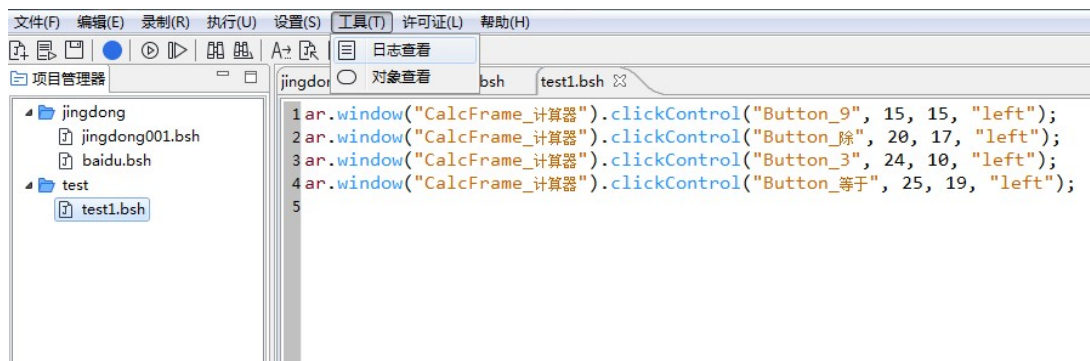
例如：执行 TestLimitRecording 项目下的 TestOne.bsh 脚本：



执行完毕后，在 AutoRunner 的工作空间，保存了相对应的和脚本名字相同的文件名的\*.logd 文件。例如：

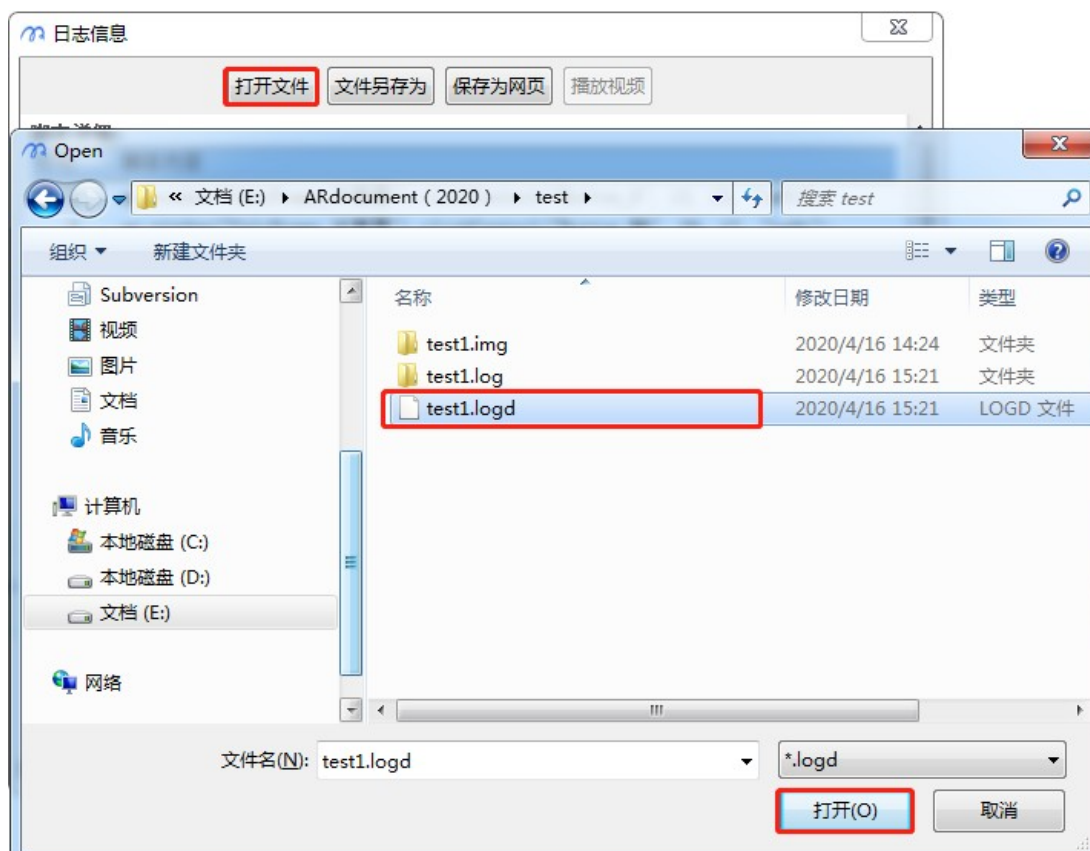


通过菜单工具，来进行日志查看操作。



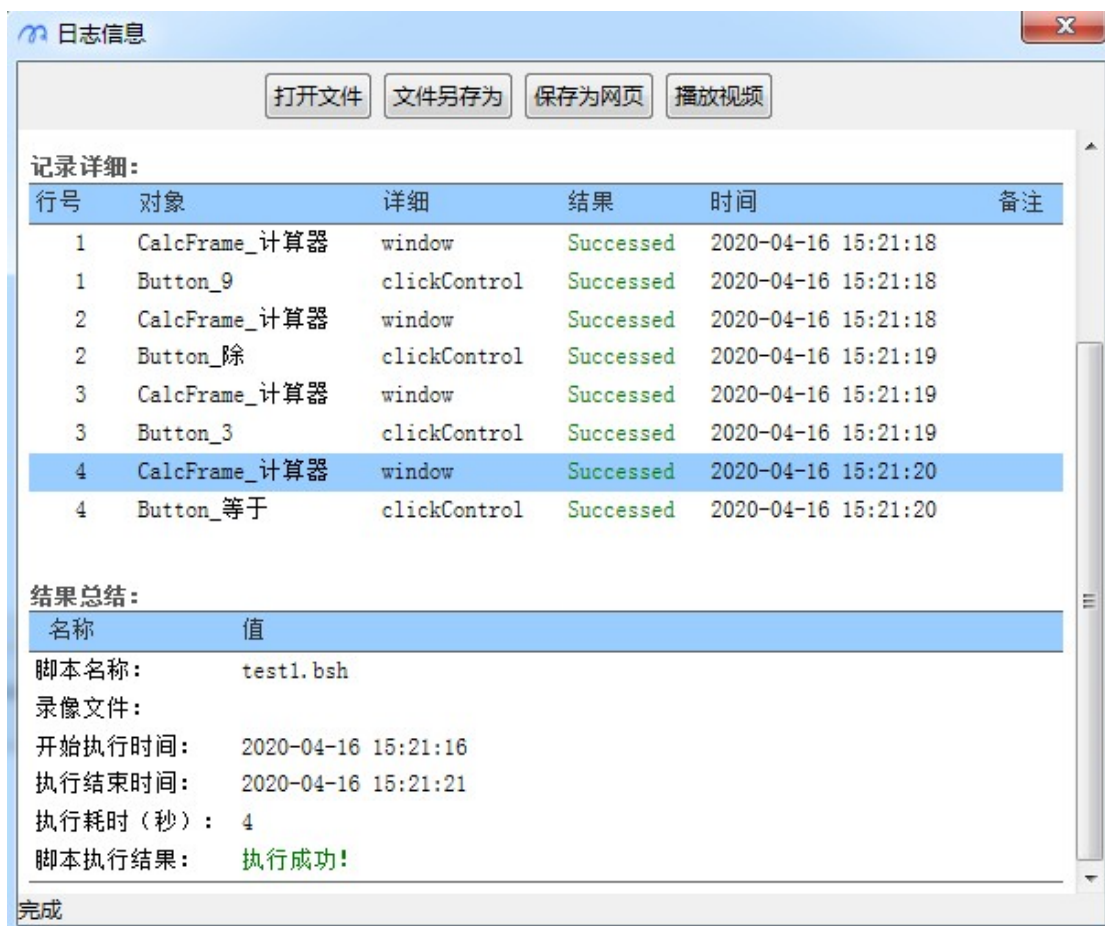
弹出如下窗口：

只能操作打开按钮，其余按钮不可操作。以下是打开 TestOne.logd 文件。



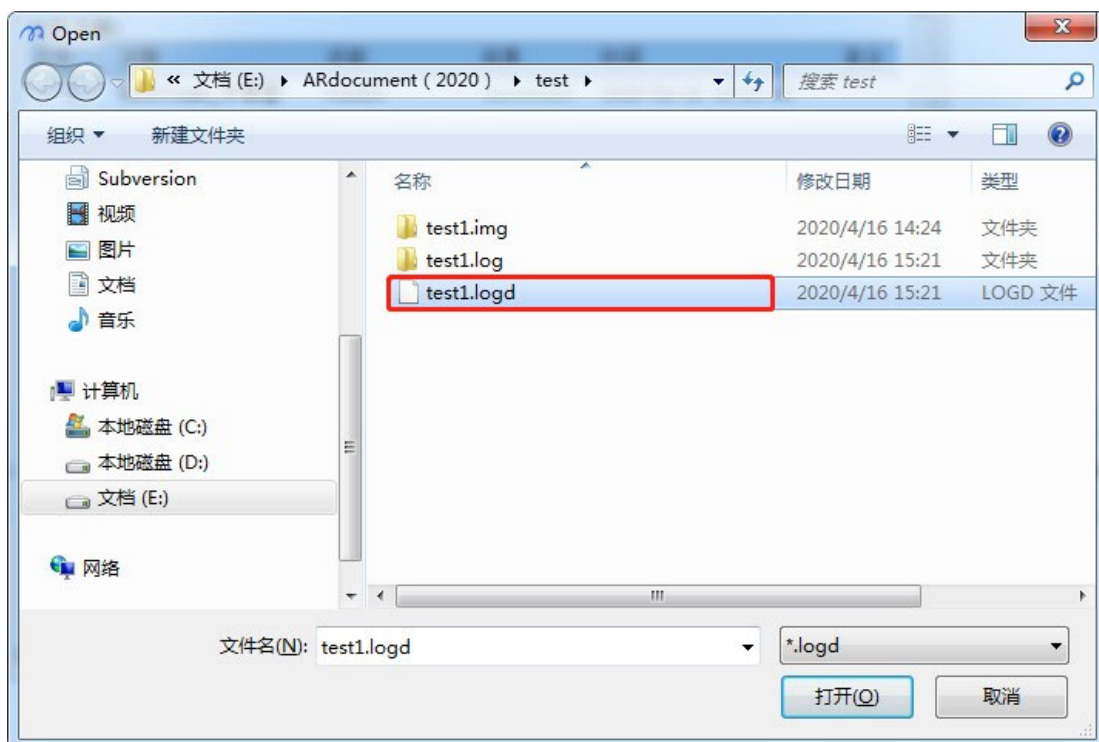
点击打开：可展示日志信息：





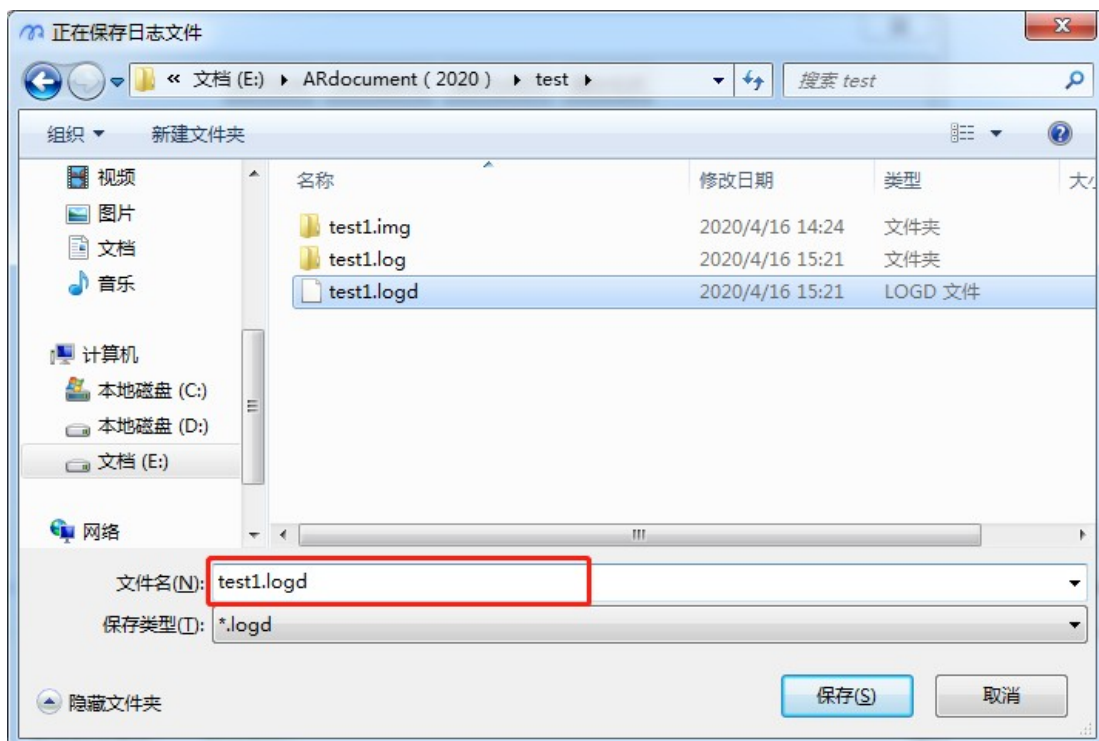
脚本详细，记录详细及脚本最终的执行结果。整个脚本的执行结果：包括以下项：脚本名，运行开始时间，运行结束时间，运行结果点击记录详细的某一行脚本可同步指定到脚本详细的具体行。此时，可将当前的日志信息，另存为一个\*.logd 文件、保存为一个\*.html 网页。

操作按钮：打开文件按钮



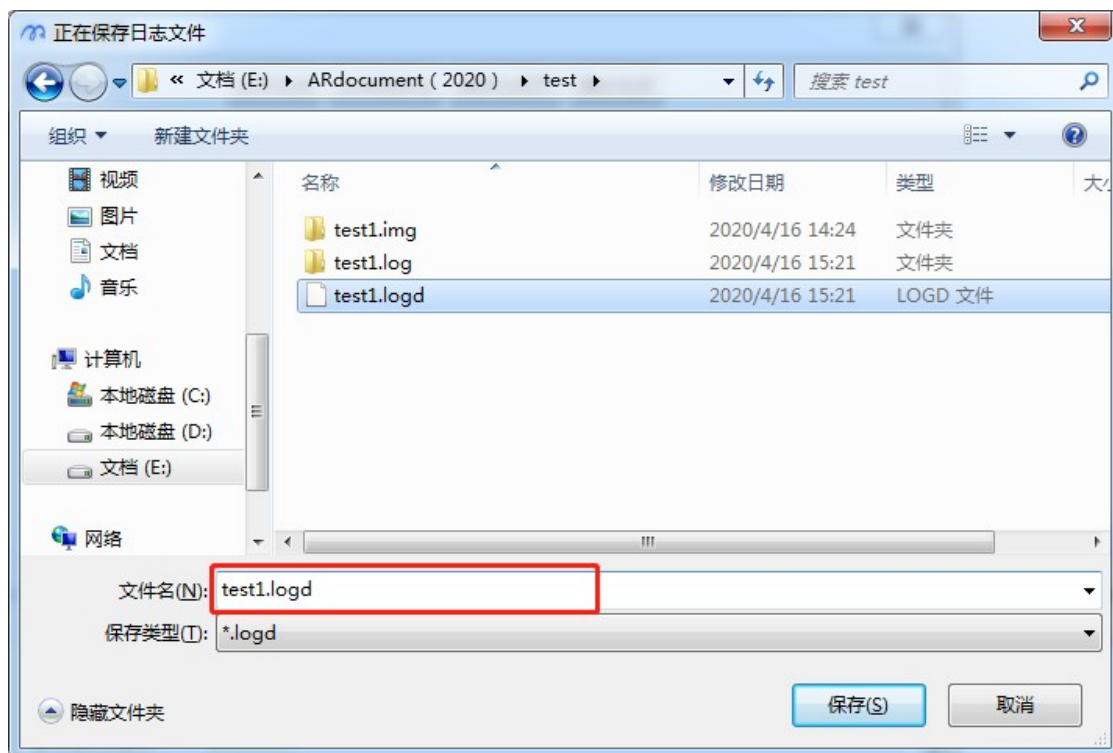
可打开一个\*.logd 文件(文件格式为: 规范定义的一个 xml 文件), 以 html 文件作为展示在当前的窗口。

操作按钮: 保存文件按钮



将当前以 html 网页形式展现的日志信息, 保存为以特定的 xml 格式的 .logd 文件。保存到本地可通过打开文件来加载显示日志信息。

操作按钮：保存网页按钮



将当前的日志信息，保存为本地的网页文件。（\*.html）

操作按钮：播放视频按钮

当点击播放按钮时，可以播放当前脚本执行的录屏视频, 例如，播放 TheLimitRecording 下的 TestOne 录制视频，如下：



● 注意事项

1、回放时必须要将待回放的脚本打开，否则回放菜单和工具栏的回放按钮将不可用。

2、在编辑器中打开了多个脚本时，应确保待回放的脚本是当前编辑的脚本。

3、回放时一定要将各项状态置于录制时的初始状态，否则回放可能会失败。

4、在回放的过程中不要再操作鼠标和键盘，直至回放完成，否则回放可能会失败。

5、如果回放的脚本被手工修改过一定要及时保存，否则修改过或是被注释掉的动作在回放时会得不到体现。

6、对于回放 Java 程序脚本，如果控件中带有滚动条（比如树控件、列表控件等），在控件中的元素不可见的情况下对元素执行的操作可能会失败。

7、在回放时可能录制下的脚本并不能完全满足要求，比如脚本命令参数化、跨脚本回放，校验脚本、修改对象库权重等等，此时就得手动修改脚本，这些内容都属于高级应用范围，具体操作请看“高手进阶”一节。


## 2.6 高级功能

### 2.6.1 参数表

#### ● 【数据参数表】

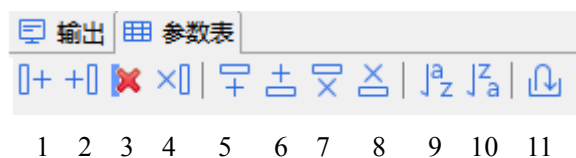
参数表用在参数化脚本过程中，可以用参数表工具栏对参数表进行各种编辑，编辑后的数据以 xls 文件形式被保存，每一列是一次循环。

```
1 for(ParameterData pd : ar.getParameterDataListFromFile("Subtract.xls").subList(0, 3))
2 {
3     ar.window("SciCalc_计算器").clickControl("Button_C");
4     ar.window("SciCalc_计算器").clickControl("Button_" + pd.getFrom("btnAFirst"));
5     ar.window("SciCalc_计算器").clickControl("Button_" + pd.getFrom("btnASecond"));
6     ar.window("SciCalc_计算器").clickControl("Button_-");
7     ar.window("SciCalc_计算器").clickControl("Button_" + pd.getFrom("btnBFirst"));
8     ar.window("SciCalc_计算器").clickControl("Button_" + pd.getFrom("btnBSecond"));
9     ar.window("SciCalc_计算器").clickControl("Button_=");
10 }
11
```



*	名称	#0	#1	#2
1	btnAFirst	7	9	4
2	btnASecond	1	0	3
3	btnBFirst	4	6	7
4	btnBSecond	5	0	7

### 【参数表工具栏】



参数表工具栏用来编辑参数表，如上图所示，每一个按钮功能如下：

- 1、在表头插入一列数据；
- 2、在表尾插入一列数据；
- 3、删除表头的一列数据；
- 4、删除表尾的一列数据；
- 5、在表头插入一行数据；
- 6、在表尾插入一行数据；
- 7、删除表头的一行数据；
- 8、删除表尾的一行数据；
- 9、将所有的行按名称的升序排列
- 10、将所有的行按名称的降序排列
- 11、从脚本对应的 xls 表中重新载入数据（参数表数据都被保存在与脚本同级同名的 xls 表中）；

### 对象库

#### ● 【对象库】

对象库中保存了所有对象的详细信息，不同对象的信息都不完全相同，在回放时就是根据这些属性来准确找到对应窗口中的控件，使之响应相应的动作。对于 Windows 程序来说某一个对象的属性包含如下内容。

名称	值	权重
category	WINCONTROL	100
winClass	SciCalc	100
name	计算器	100
value		0
role	00000009	100
state	00140000	0
defaultAction		0
description		0
help		0
keyboardShort...		0
style	14ca0044	0
exStyle	00050900	0
location	154,203	0
position	0,0	100
width	260	100
height	245	100

Java 程序对象的属性包含如下内容。

名称	值	权重
category	JAVACONTROL	100
id	0,1,0,0,1,0,0,0,0,0,0,	100
name		100
role	tree	100
state	enabled,focusable,visible,showing,opaque	0
value	MusicRock	0
description		0
location	285,292	0
position	9,129	0
width	708	0
height	537	0

网页对象的属性包含如下内容。

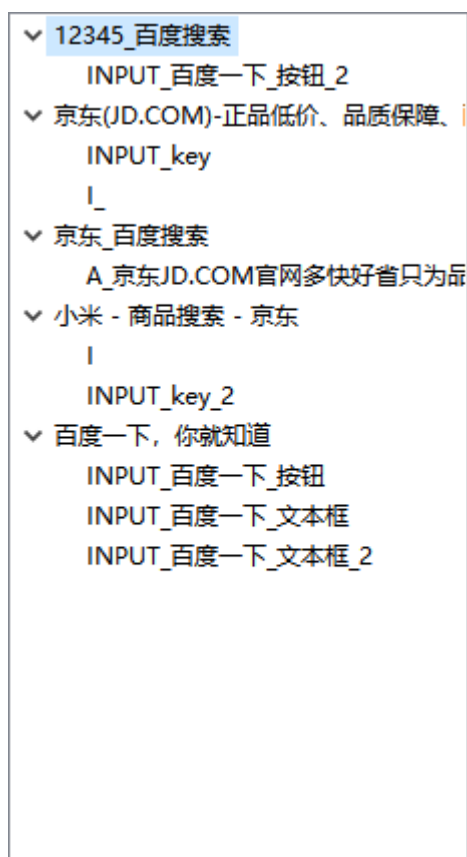
名称	值	权重
category	IECONTROL	100
id	1,1,3,2,0,0,0,1,2,0,	0
tagName	INPUT	100
className	lst	0
innerId		100
name	q	100
type	text	100
offsetLeft	0	100
offsetTop	0	100
offsetWidth	512	100
offsetHeight	32	100
description	q	100
value		0
urlLink		0
disabled	false	0
readOnly	false	0
innerText		0
outerText		0

Flex 程序对象的属性包含如下内容。



名称	值	权重
category	FLEXCONTROL	100
defaultAction	Press	0
description		0
exStyle	00000800	0
height	23	100
help		0
keyboardShort...		0
location	686,232	0
name	按钮	100
position	306,23	100
role	0000002b	100
state	00100004	0
style	56000000	0
value		0
width	72	100
winClass	PushButton	100

这些属性的值和权重都可以手工编辑，方法是双击每一个要编辑的窗格即可；在不处于编辑状态时可以右击鼠标，在弹出菜单中选择复制即可复制出所在窗格的文本。面板的左边是一颗属性树，如下图所示：



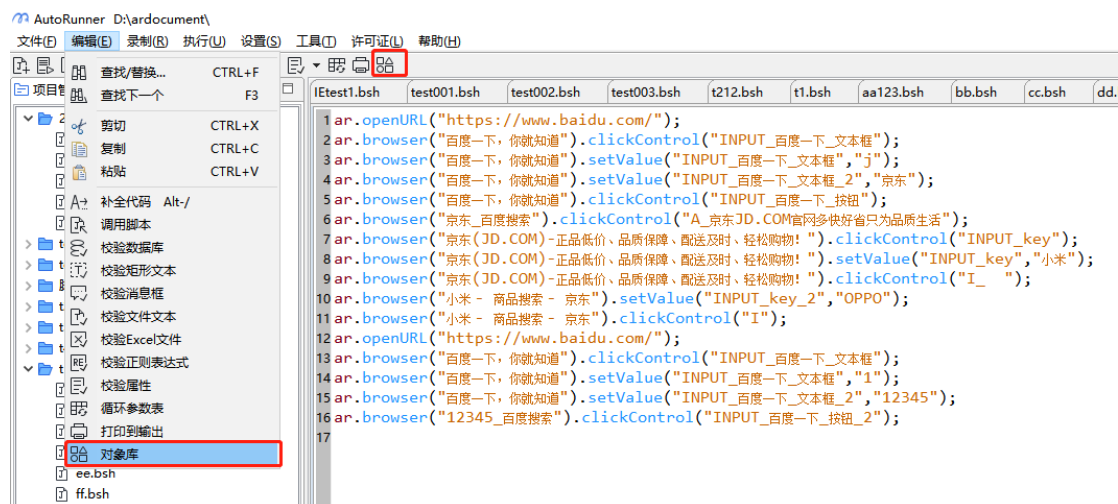
每个可折叠的点都是一个窗口，节点名即是窗口对象名（不是窗口名）；其下的子节点都是窗口中的控件，节点名即是控件对象名（不是控件名）。每点击一下树中的节点，右边的属性栏中就会显示出节点的各项属性，并可以进行编辑。

在节点上点击鼠标右键，可以弹出菜单，对节点名进行删除和复制操作。面板的下方是两个按钮，分别是“增加对象”和“关闭”。增加对象用于手工添加对象，具体看高级应用，关闭按钮关闭对象库，并保存所做的修改。

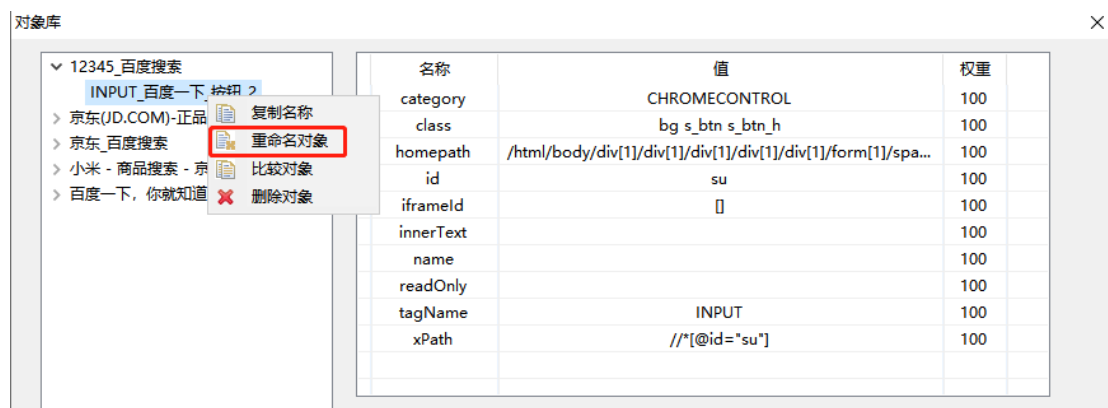
## ● 【对象重命名】

### 对象重命名

首先，打开对象库。

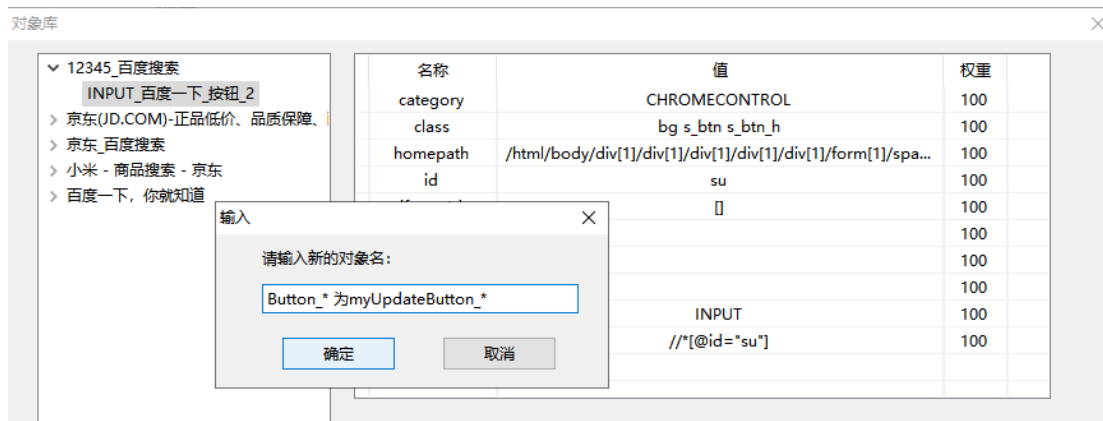


需要选中对象进行重命名：

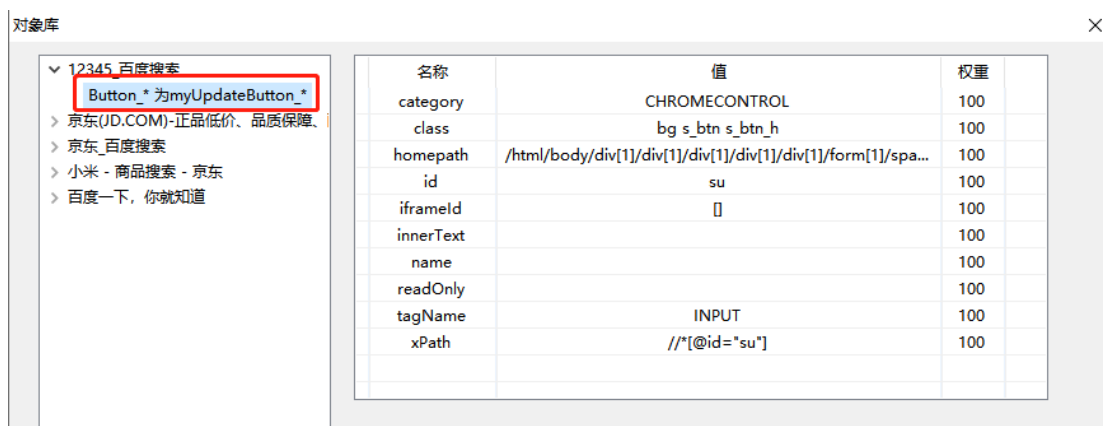


例如命名，Button\_\* 为 myUpdateButton\_\*





结果如下:



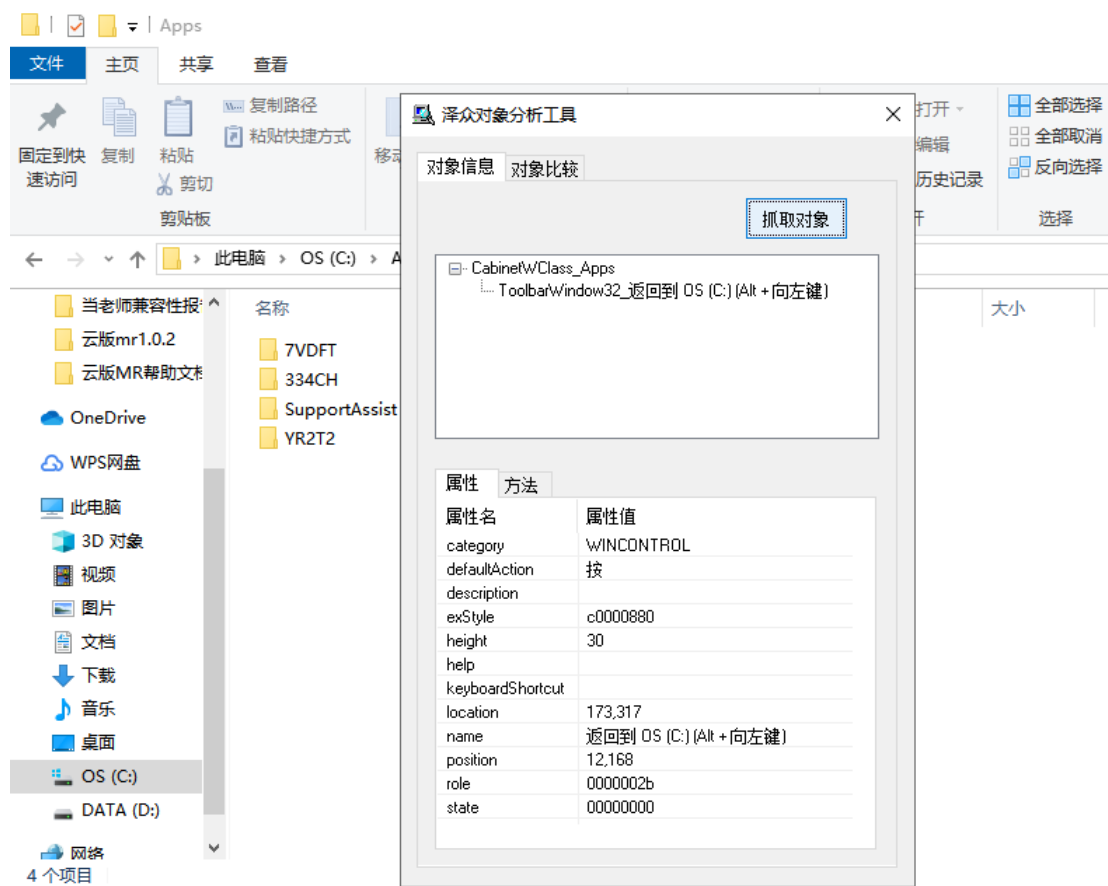
关闭对象库窗口后，修改生效。

### ● 【对象信息】

通过菜单【工具】→【对象查看】→对象信息



通过在【抓取对象】按钮上按下鼠标左键，拖动到某接口对象上，释放左键，这时该对象信息会显示在对象信息窗口中。例如，选择我的计算机的后退按钮对象如图：



通过本功能用户抓取一个对象，包括对象的属性名称和属性值，以及这个对象所用的方法。

## 2.6.2 对象比较

【工具】→【对象查看】来比较对象

通过本功能，用户可以通过选择 2 个对象，来显示这 2 个对象的属性，并比较这 2 个对象的属性差异，有差异部分的属性在第 2 个对象相应属性中用红色字体标出

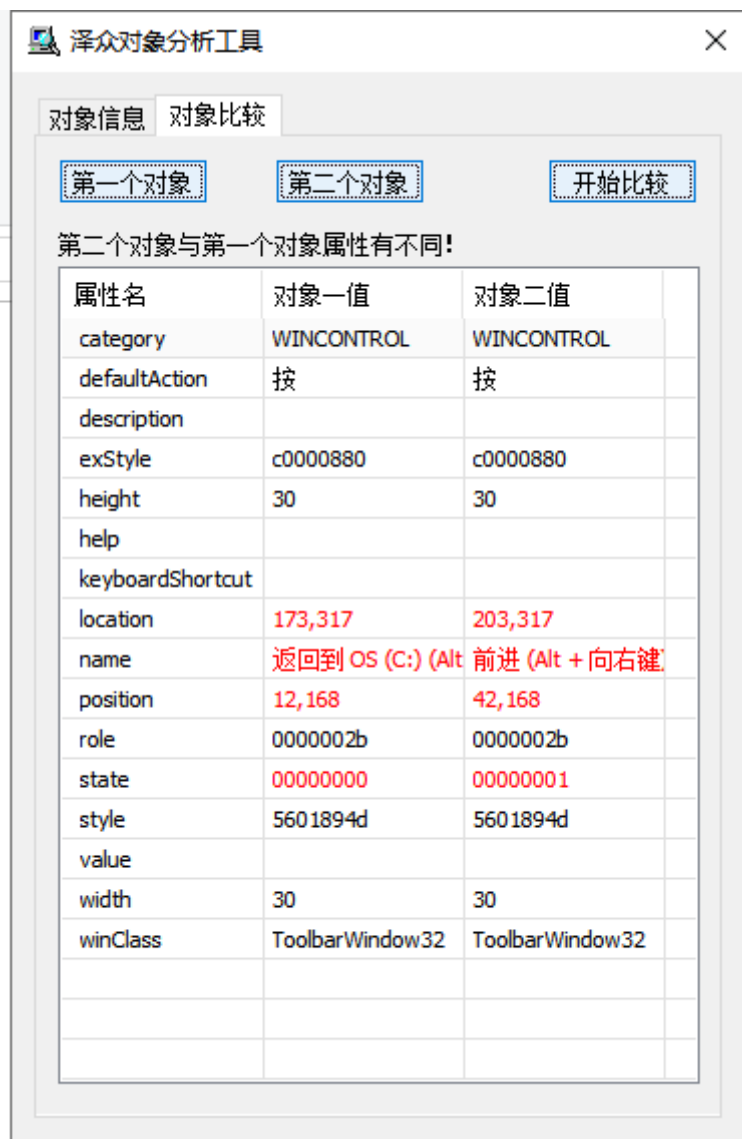
A、图中表格列出用户所选对象属性：第一列名称“属性名”，表示属性名称；第二列名称“对象 1 属性值”，表示选中第一个对象的属性值；第三列名称“对象 2 属性值”，表示选中第二个对象的属性值；

B、点击“选择第一个对象”按钮不放，然后鼠标移动到需要比较的对象上释放按键，系统将该对象属性值显示在表格第二列。

C、点击“选择第二个对象”按钮不放，然后鼠标移动到需要比较的对象上

释放按键，系统把该对象属性值显示在表格第三列。

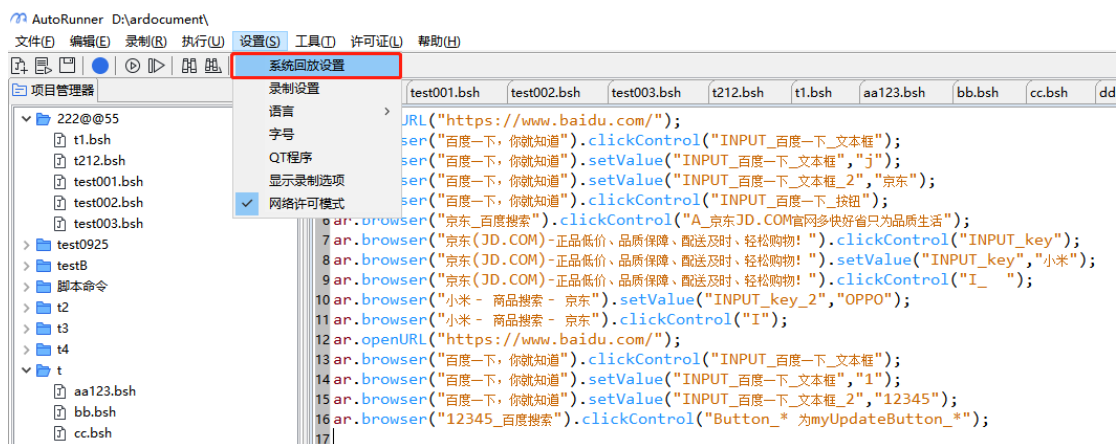
D、点击“开始比较”按钮，系统对这 2 个对象的属性值进行一一对比，如有某个属性值不同，则这 2 个对象对应的值用红色字体显示（即在表格第二、三列）。



比较两个对象之间的属性值的不同。

### 2.6.3 回放设置

本功能主要用来定义一些影响系统回放的设置信息。这些设置信息对系统中所有脚本回放都有影响，要变更具体某个脚本的这些设置信息时，可在脚本中通过命令修改来实现，通过点击 **【设置】** → **【系统回放设置】**

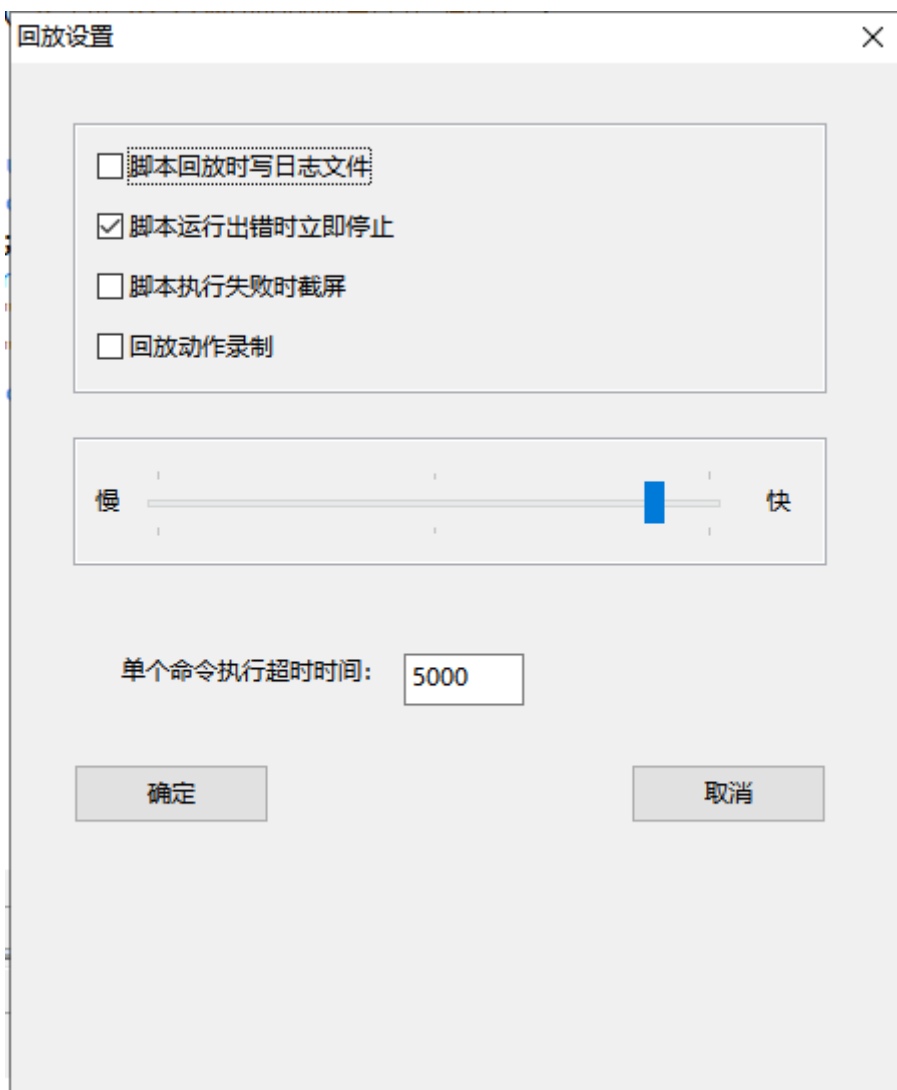


1、在上图“选项”中：

勾选“脚本回放时写日志文件”表示开放底层 log 功能，否则关闭底层 log 功能。勾选“脚本运行出错时立即停止”表示如果运行出错则立即停止，否则继续执行，勾选“脚本执行失败时截屏”表示脚本执行失败时系统截屏，否则不截屏。

2、在“播放速度”中，用户通过此拖动条来调整命令之间的执行间隔实现调节脚本回放的速度。

3、在“播放超时”中，用户可设置单个命令执行的超时时间（默认 5000ms），在命令运行一次失败时，会等待一段时间（由播放速度决定）后再次执行动作，如果执行动作的总时间超过此设置的时间，就会放弃命令的执行，继续下一条命令。



### 3 高手进阶

#### 3.1 参数表编辑

在循环参数表之前首先要进行参数表编辑，参数表如下图所示：

输出		参数表					
	名称	#0	#1	#2	#3	#4	
1	btnA	0	5	1	9	6	
2	btnB	1	4	3	9	7	
3	result	1	9	4	18	13	

第一栏是工具栏，主要是对参数表的行列进行增加删除操作、排序操作等等。最后一个图标是从关联的 xls 文件中重新载入数据，当编辑了参数表中的某

个单元格时左上角会有一个 \* 标记，提示保存修改，点击菜单栏中的【文件】→【保存】或按下 Ctrl+S 快捷键可以保存修改，并且星号消失。

参数表“名称”这一栏可以是有意义的任何名字，它用于 getFrom、putInto 等命令函数的第一个参数。

在每一个单元格中双击可以对单元格进行编辑，在每一个单元格中鼠标右击可以复制单元格内容。

## 3.2 参数传递

在进行脚本串联调用时，可能某些数据要在不同脚本之间共享，也就是参数传递。参数传递命令为 getFrom、putInto，这两条命令都是 ParameterData 类成员函数。

### ● 一般性的参数传递

下面是一段计算器的脚本。前面五句做了一个加法操作，第七句是得到计算器的计算结果，第八句是用 putInto 命令将结果保存到名为 result 的变量用，变量名可以任取，其中 parameterData 是一个 ParameterData 类型的全局变量。第九句是调用 Notepad.bsh 记事本脚本。

```

1 ar.window("SciCalc_计算器").clickControl("Button_C", 39, 16, "left");
2 ar.window("SciCalc_计算器").clickControl("Button_1", 16, 14, "left");
3 ar.window("SciCalc_计算器").clickControl("Button_+", 15, 15, "left");
4 ar.window("SciCalc_计算器").clickControl("Button_2", 25, 14, "left");
5 ar.window("SciCalc_计算器").clickControl("Button_=", 16, 14, "left");
6 //ar.window("SciCalc_计算器").//record element:Edit
7 String str = ar.window("SciCalc_计算器").getProperty("Edit", "value");
8 ar.parameterData.putInto("result", str);
9 ar.callScript("Notepad.bsh");

```

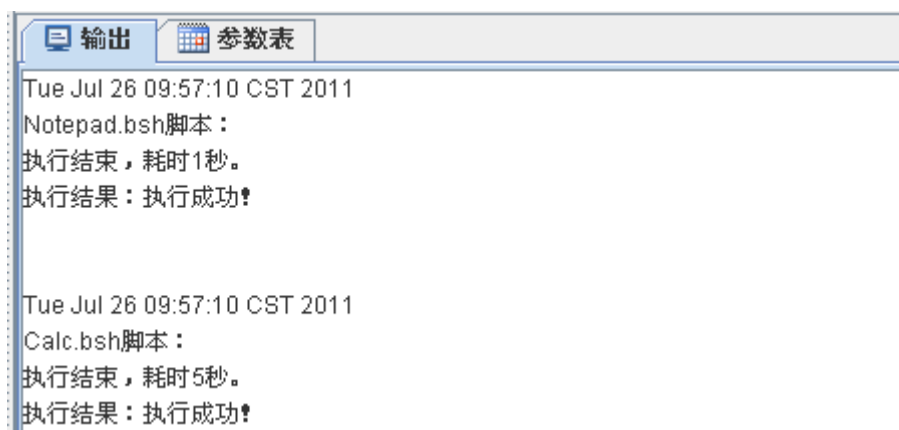
记事本的脚本比较简单，第一句是从 parameterData 中把 result 的结果取出，这里调用的是 getFrom 命令。第三句是将结果输出到记事本中。

```

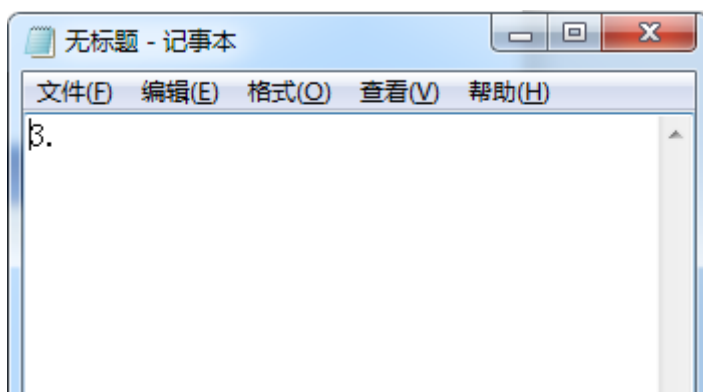
Notepad.bsh
1 String str = ar.parameterData.getFrom("result");
2 //ar.window("Notepad_新建 文本文档.txt - 记事本").//record element:Edit
3 ar.window("Notepad_新建 文本文档.txt - 记事本").setValue("Edit", str);

```

下面是执行计算器脚本的运行结果：



记事本中打印的内容如下：



### ● 参数化脚本中的参数传递

下面是一段计算器的参数化脚本。

```

1
2
3 for (ParameterData pd : ar.getParameterDataList("E:\\Workspace\\runScript\\Add.xls").subList(0, 3))
4 {
5     ar.window("SciCalc_计算器").clickControl("Button_C");
6     ar.window("SciCalc_计算器").clickControl("Button_" + pd.getFrom("btnAFirst"));
7     ar.window("SciCalc_计算器").clickControl("Button_" + pd.getFrom("btnASecond"));
8     ar.window("SciCalc_计算器").clickControl("Button_" + pd.getFrom("btnAThird"));
9     ar.window("SciCalc_计算器").clickControl("Button_+");
10    ar.window("SciCalc_计算器").clickControl("Button_" + pd.getFrom("btnBFirst"));
11    ar.window("SciCalc_计算器").clickControl("Button_" + pd.getFrom("btnBSecond"));
12    ar.window("SciCalc_计算器").clickControl("Button_" + pd.getFrom("btnBThird"));
13    ar.window("SciCalc_计算器").clickControl("Button_=");
14    ar.parameterData = pd;
15    String str = ar.window("SciCalc_计算器").getProperty("Edit", "value");
16    pd.putInto("addResult", str);
17    ar.callScript("Notepad.bsh");
18 }

```

	名称	#0	#1	#2
1	btnAFirst	1	7	3
2	btnASecond	6	3	4
3	btnAThird	3	9	0
4	btnBFirst	7	5	6
5	btnBSecond	5	8	1
6	btnBThird	4	7	2

循环参数表执行加法操作，第 14 句，这一句把临时变量 pd 保存在了 ar.parameterData 参数中，因为在 Notepad 中还会用到 pd 参数。第 15 句获得计算结果；第 16 句将结果保存到一个名为 addResult 的变量中，变量名可以是有意义的其他名字。第 17 句调用 Notepad.bsh 保存计算结果。

下面是 Notepad.bsh 脚本代码（两个脚本必需在同一项目中，文本文件名称为 Output.txt）

```

1 String strNotepad = ar.window("Notepad_Output.txt - 记事本").getProperty("Edit", "value");
2 String strAddResult = ar.parameterData.getFrom("addResult");
3 strNotepad +=
4 ar.parameterData.getFrom("btnAFirst") +
5 ar.parameterData.getFrom("btnASecond") +
6 ar.parameterData.getFrom("btnAThird") + " + " +
7 ar.parameterData.getFrom("btnBFirst") +
8 ar.parameterData.getFrom("btnBSecond") +
9 ar.parameterData.getFrom("btnBThird") + " = ";
10 ar.window("Notepad_Output.txt - 记事本").setValue("Edit", strNotepad + strAddResult + "\r\n");

```

这段脚本先获取计算器和记事本的值，第三句是将表达式的值拼接起来，重新设置记事本的值。注意：这里用 ar.parameterData.getFrom("??") 直接获取 xls 表中的数据，因为在 Calc 脚本中 ar.parameterData 参数已经被赋值了。运行 Calc 脚本结果如下：



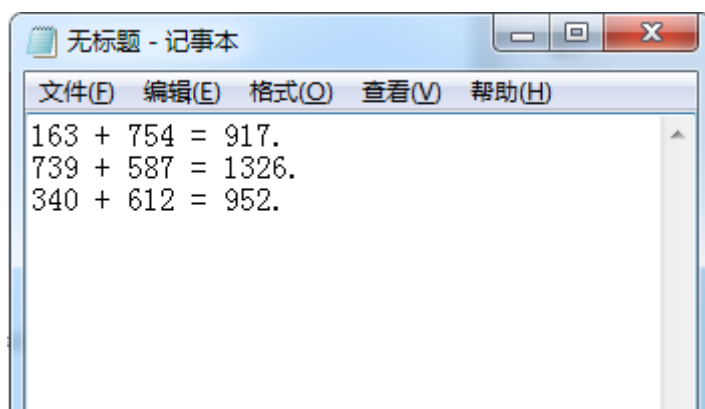
```
Thu Mar 03 13:10:34 CST 2011
Notepad.bsh脚本：
执行结束，耗时1秒。
执行结果：执行成功!

Thu Mar 03 13:10:42 CST 2011
Notepad.bsh脚本：
执行结束，耗时1秒。
执行结果：执行成功!

Thu Mar 03 13:10:50 CST 2011
Notepad.bsh脚本：
执行结束，耗时1秒。
执行结果：执行成功!

Thu Mar 03 13:10:50 CST 2011
Calc.bsh脚本：
执行结束，耗时23秒。
执行结果：执行成功!
```

记事本中打印的内容如下：



注：

每个脚本中用到的对象，都必须在此脚本的对象库中存在。

### 3.3 函数调用

一、在一个脚本中调用函数方法如下图所示：

```
1 int fun_add(int i, int j)
2  { //这个是普通函数
3      return i + j;
4  }
5 public class TestIn
6  { //这个是类中的函数
7      public int fun_sub(int i, int j)
8      {
9          return i - j;
10     }
11 }
12 //这里可以调用
13 System.out.println("add=" + fun_add(1,2));
14 TestIn ti = new TestIn();
15 System.out.println("sub=" + ti.fun_sub(1,2));
```

脚本

输出 参数表

```
add=3
sub=-1
Tue Aug 30 11:04:08 CST 2011
a.bsh脚本:
执行结束,耗时1秒。
执行结果:执行成功!
```

二、在两个脚本中调用函数的方法如下图所示（示例中 B 脚本中的自定义函数被 C 脚本调用）：

```
b.bsh
1  int fun_mul(int i, int j)
2  {/这个普通函数
3      return i * j;
4  }
5
6  public class TestOther
7  {/这个类中的函数
8      public int fun_div(int i, int j)
9      {
10         return i / j;
11     }
12 }
```

```
c.bsh
1  //注意这里要写绝对路径
2  source("E:\\Workspace\\test1\\b.bsh");
3
4  System.out.println("mul=" + fun_mul(10,2));
5  TestOther to = new TestOther();
6  System.out.println("div=" + to.fun_div(10,2));
```

脚本

输出 参数表

```
mul=20
div=5
Tue Aug 30 11:24:40 CST 2011
c.bsh脚本:
执行结束,耗时1秒。
执行结果:执行成功!
```

三、调用 jar 包中函数的方法如下图所示:



```
d.bsh
1 //注意这里要写绝对路径
2 addClassPath("E:\\Workspace\\test1\\test.jar");
3
4 com.test.Test t = new com.test.Test();
5 t.fun();
```

脚本

输出 参数表

hello ooo  
Tue Aug 30 13:22:12 CST 2011  
d.bsh脚本：  
执行结束，耗时1秒。  
执行结果：执行成功!

### 3.4 记录击键

#### ● 记录击键

如果事先在设置菜单中勾选了【显示录制选项】，在点击录制按钮后弹出的对话框中会有【记录击键】选项，选中时录制会记录键盘所有的击键动作，在没有选中的情况下只会记录下几个比较特殊的键，比如回车键、Alt+Tab 复合键等，击键命令为 `inputKey`。在一般的录制中这个功能并不是必需的，它的作用主要是用于需要记录键盘操作的情况。因为此功能并不常用，在默认情况下没有选中，其录制和回放操作和普通的录制、回放没有区别。在手工添加 `inputKey` 命令时，输入参数请参考键盘键码表。

### 3.5 记录时间间隔

如果事先在设置菜单中勾选了【显示录制选项】，在点击录制按钮后弹出的对话框中会有【记录时间间隔】选项，选中时录制会记录每次动作到上一次动作的时间间隔，也就是会出现 `sleep` 命令，它的时间单位是毫秒。在一般的录制中这个功能并不是必需的，它的作用主要是用于需要精确记录整个录制过程的情况，回放时要求保证与录制时在时间上同步，比如在某些对时间很敏感的操作中，在某些网页的录制中（网页需要加载时间），或是在某些必需要加入延时操作的

特殊场合。因为此功能并不常用，在默认情况下没有选中，其录制和回放操作和普通的录制、回放没有区别。

### 3.6 扫描 JDK

可能有些软件的使用者电脑上装了不少一个版本的 JDK，或者软件先前使用时电脑中装的是一个低版本，而后又装了一个更高级的版本，此时再次使用软件可能出现不能录制 Java 程序脚本的情况(当然，首先得确定以前是可以录制的)，解决的方法是点击软件的菜单【设置】→【扫描 JDK/JRE】即可，在扫描之前应先关闭所有网页和 Java 应用程序，当提示扫描成功时就可正常录制 Java 程序脚本了。

### 3.7 手工添加对象

在以下情况下可能需要手工添加对象到对象库中：

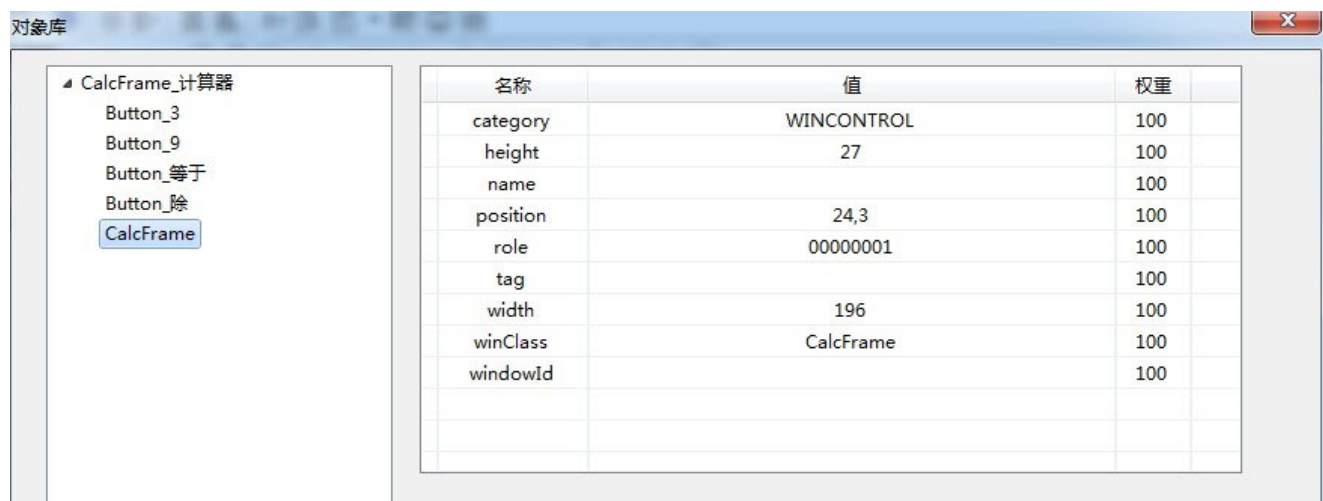
- A、回放时出现某个对象在对象库中没有找到的错误；
- B、由于错误修改对象属性导致回放失败；
- C、某些对象在录制时不方便录制或是录制失败；
- D、不想重录已有脚本，只是想增加某个或某几个新对象到对象库。

手工添加对象的两种方法：

录制过程中手工添加。以计算器为例，在录制中如果我们想把计算器的结果输出框对象（Edit 控件）也录制下来方便我们后来设置校验点。方法是：将鼠标移到结果输出框上面，并同时按下键盘上的 Ctrl 和 Alt 键并保持一秒左右，在右下角的信息输出框中会有如下一句信息显示。

```
//ar.window("SciCalc_计算器").record element:Edit
```

当出现此信息时，表示对象已被添加到了对象库中，在对象库中的信息如下：



B、直接在对象库中添加。同样是录制计算器的结果输出框对象。方法是：先打开对象库，在面板的左下角会有一个“增加对象”的按钮，点击之后软件会再次进入录制状态，将鼠标移到结果输出框上再同时按下键盘上的 Ctrl 和 Alt 键并保持 1 秒左右即可添加成功。

注：

A、在操作时，应先将鼠标移动到要添加的对象上再同时按下键盘上的 Ctrl 和 Alt 键，这样可避免不必要的对象被记录在对象库中；

B、对象被记录所耗时间不会超过一秒，当信息输出框中显示出类似第一幅图的添加信息时表示已添加成功，所以不需要长时间按下 Ctrl 和 Alt 键；

C、有些对象只能手工记录，比如上面提到的计算器的结果输出框对象，因为它不接受输入，实际等同于一个静态的文本控件（Static），而静态控件在录制时是不会被记录的。

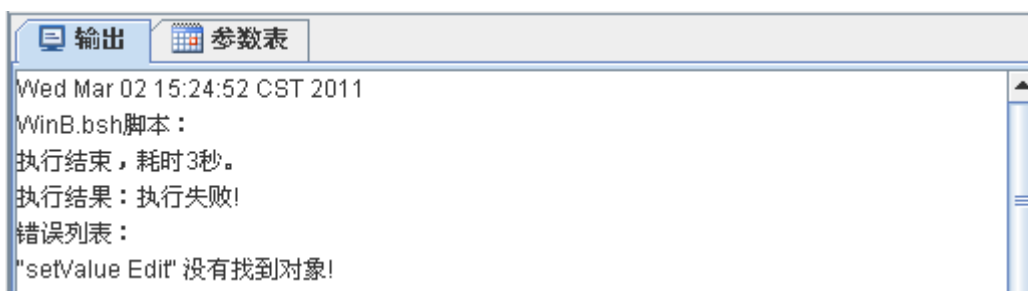
### 3.8 修改权重

每一个对象都有很多的属性，每个属性后都附带了一个权重，对于一些很重要的属性默认的权重是 100，而对于不是太重要的属性默认权重是 0。当回放某个对象动作时首先会在窗口中找到这一对象，之后再与之进行各项属性比较，如果属性权重大于 0，就会比较属性值，如果相等则继续下一个属性比较，如果不相等也不会马上认为这不是要找的对象（模糊识别），而是会将权重值累加，当不相等属性值的累加权重大于 100 时才认为对象不匹配。所以，如果一个对象的某项属性值变动了，而它的权重又是 100 的话，就可以通过降低权重使回

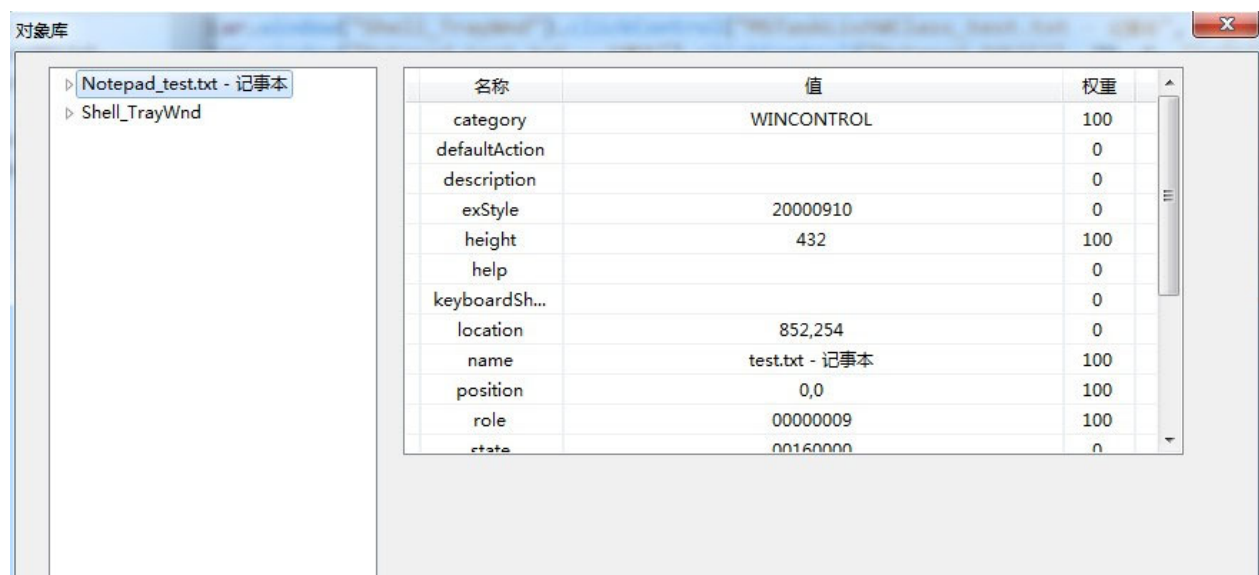
放得以通过。下面以录制记事本程序为例，介绍一下权重的修改。新建一个名为 test.txt 的文本文档，双击打开，录制一段在记事本中输入文本的动作（比如输入文本 spasvo），脚本如下：

```
1 ar.window("Notepad_test.txt - 记事本").setValue("Edit", "spasvo");
```

在回放之前先，改变记事本窗口的大小，比如拉宽窗体，回放时会有如下错误：



因为此时 Edit 对象的宽度已经改变，而宽度的权重是 100，所以回放失败。要使得回放通过，可以修改宽度属性的权重。打开对象库，在右边的树中找到名为 Edit 的节点并单击打开属性面板，将其中的 width 属性权重设为 50（小于 100 的数值均可），关闭对象库后回放就可以成功。



在实际使用当中，也可以根据情况修改其他参数的权重，如果修改的权重项越多，回放时查找到的目标对象的准确度会越低，所以一般的回放不建议手动修改，只有在极其特殊的情况下，比如对象的某项属性会经常性动态变动而它的权重又是 100 时才适合使用；当然如果觉得某项属性很重要，而在对象库中的权重又是 0，你也可以手动将它的权重设为 100，这样在回放时此项属性也会进行匹配判断，以提高准确度。通常情况下不建议使用者手工修改权重，一是修改权重

后可能会导致查找的对象有误，另一个原因是软件在回放时内部做了一些智能判断，如果第一次查找对象失败，会自动的将某一些权重设为零，再重新查找。

### 3.9 添加校验点

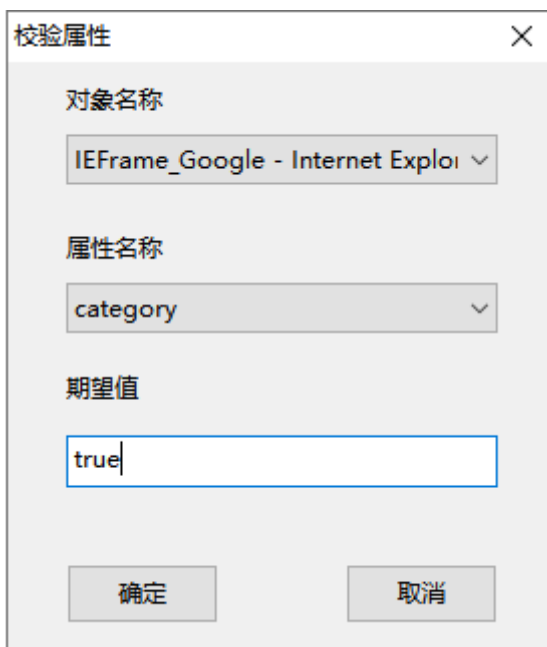
当前版本支持对象属性校验、数据库校验、矩形区域文本校验、消息框文本校验、文件文本校验、Excel 文件校验、正则表达式校验等，插入校验命令有如下三种方式：

- 1、通过【编辑】菜单添加；
- 2、在脚本编辑器中右击添加；
- 3、通过工具栏添加。

所有校验命令的返回值都是一个布尔值，如果实际值和期望值相等匹配，返回结果为 true，否则返回 false。

#### 3.9.1 校验属性

校验属性的脚本命令为 [checkProperty](#) 命令，校验属性对话框如下图所示



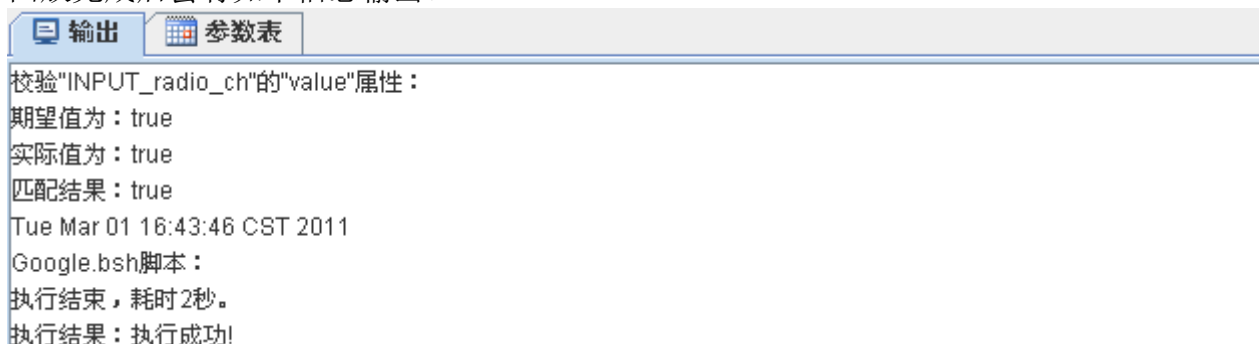
第一栏中列举出了对象库中所有的对象名称，在这里选择需要校验的对象；  
第二栏中列举出了此对象支持的所有属性，在这里选择需要校验的属性；  
第三栏中填入期望值，这个值根据你选择的对象和属性的不同而不同，可以参考对象库中相应属性的属性值（期望值都是字符串）。



点击确定后，在脚本编辑器中光标位置处会自动添加如下一行脚本。

```
ar.window("IEFrame_Google - Internet Explorer").checkProperty("INPUT_radio_ch", "category", "true");
```

回放完成后会有如下信息输出。



### 3.9.2 校验数据库

校验数据库的脚本命令为 [checkDatabase](#)(See 1.4.2)命令，校验数据库对话框如下图所示：



The dialog box titled '校验数据库' (Check Database) contains the following fields:

- 类型** (Type): A dropdown menu with 'MySQL' selected.
- 地址(例如 192.168.1.50:12345/mydb)** (Address): A text box containing '192.168.0.104:3306/testcenter'.
- 用户名** (Username): A text box containing 'root'.
- 密码** (Password): A text box containing 'root'.
- SQL语句** (SQL Statement): A text box containing 'select count(\*) from user'.
- 期望值** (Expected Value): A text box containing '4'.

At the bottom, there are two buttons: '确定' (OK) and '取消' (Cancel).

第一栏中列举出了当前支持的数据库类型，在这里选择校验的数据库类型；

第二栏输入数据库地址、端口号、数据库名称，格式如上图所示；

第三栏输入数据库登录用户名；

第四栏输入数据库登录密码；

第五栏输入数据库查询语句；

第六栏中填入期望值。

点击确定后，在脚本编辑器中光标位置处会自动添加如下一行脚本。

```
ar.checkDatabase("MySQL", "192.168.0.104:3306/testcenter", "root", "root", "select count(*) from user", "4");
```

回放完成后会有如下信息输出。



### 3.9.3 校验消息框

校验消息框的脚本命令为 [checkMessageBox](#) 命令，校验消息框对话框如下图所示：



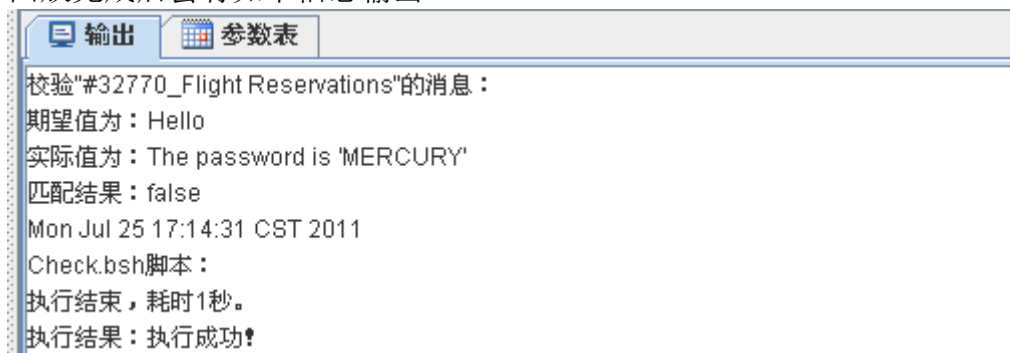
第一栏中列举出了对象库中所有的窗口对象名称，在这里选择需要校验的消息窗口对象；

第二栏中填入期望值，这个值一般为消息框中的消息文本。

点击确定后，在脚本编辑器中光标位置处会自动添加如下一行脚本。

```
ar.checkMessageBox("#32770_Flight Reservations", "Hello");
```

回放完成后会有如下信息输出。



### 3.9.4 校验矩形文本

校验矩形文本的脚本命令为 [checkRectText](#) 命令，校验矩形文本对话框如下图所示：



第一栏中列举出了对象库中所有的对象名称，在这里选择需要校验的对象；

第二栏中填入矩形左上角坐标（相对于对象左上角的坐标）；

第三栏中填入矩形右下角坐标（相对于对象左上角的坐标）；

第四栏中填入矩形区域文本的期望值；

点击确定后，在脚本编辑器中光标位置处会自动添加如下一行脚本。

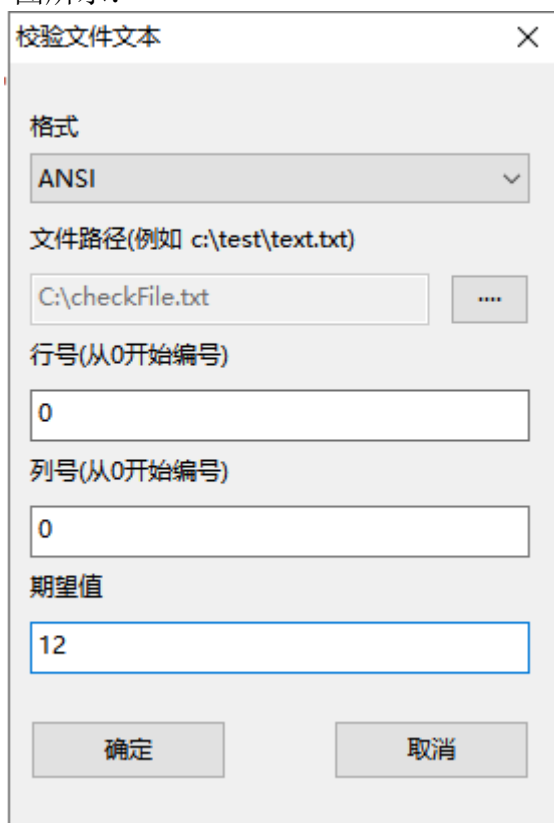
```
ar.window("#32770_记事本").checkRectText("button_ok", 0, 0, 60, 23, "ok");
```

回放完成后会有如下信息输出。



### 3.9.5 校验文件文本

校验文件文本的脚本命令为 [checkFileText](#)(See 1.4.7)命令，校验文件文本对话框如下图所示：



第一栏中列举出了常用文件的编码格式，在这里选择文件的编码格式；

第二栏中填入文件的全路径；

第三栏中填入待校验文本在文件中的行号；

第四栏中填入待校验文本在文件中的列号；

第五栏中填入期望值；

点击确定后，在脚本编辑器中光标位置处会自动添加如下一行脚本。

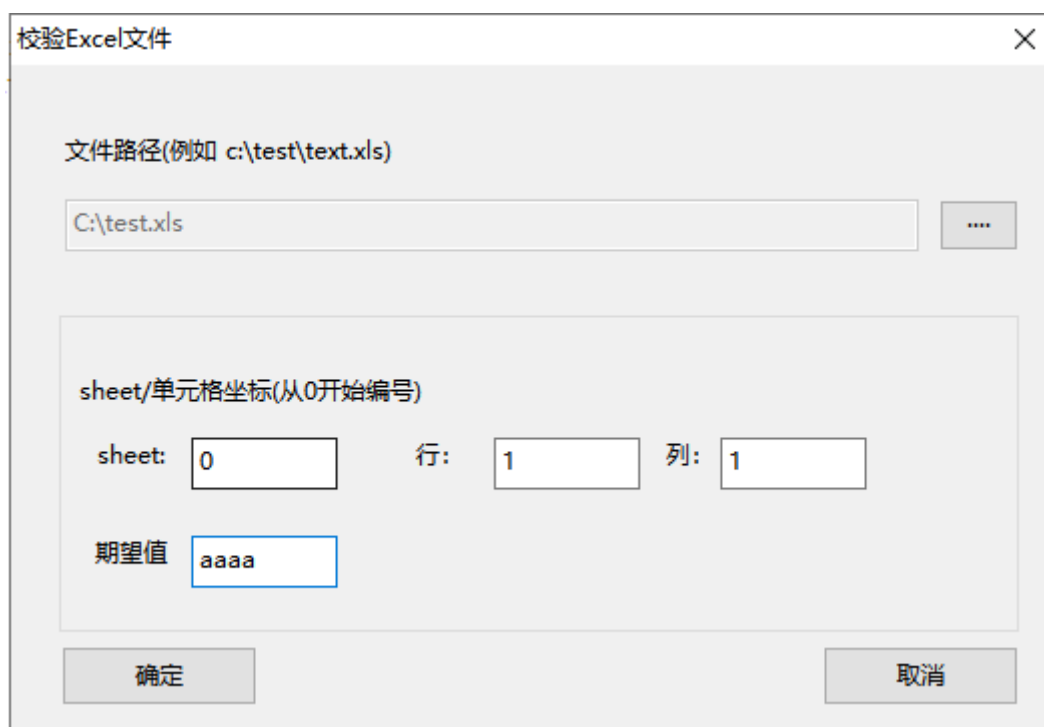
```
ar.checkFileText("ANSI", "C:\\checkFile.txt", 0, 0, "12");
```

回放完成后会有如下信息输出。



### 3.9.6 校验 Excel 文件

校验 Excel 文件的脚本命令为 [checkExcelCell](#)(See 1.4.3)命令，校验 Excel 文件对话框如下图所示：



第一栏填入 Excel 文件全路径；

第二栏中填入单元格坐标;

第三栏中填入期望值;

点击确定后, 在脚本编辑器中光标位置处会自动添加如下一行脚本。

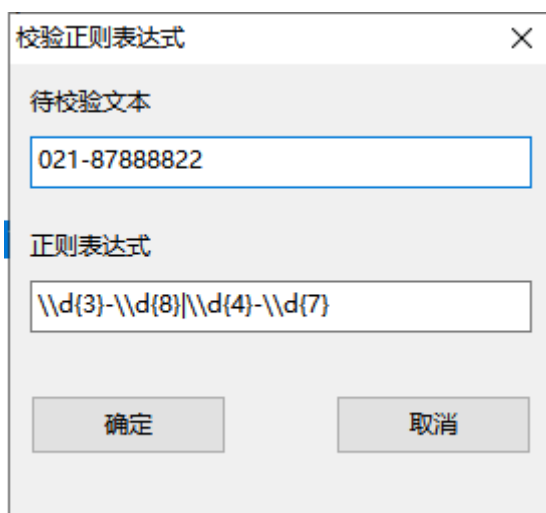
```
ar.checkExcelCell("C:\\test.xls",0, 1, 1, "aaaa");
```

回放完成后会有如下信息输出。



### 3.9.7 校验正则表达式

校验正则表达式的脚本命令为 [checkRegex](#) 命令, 校验正则表达式对话框如下图所示:



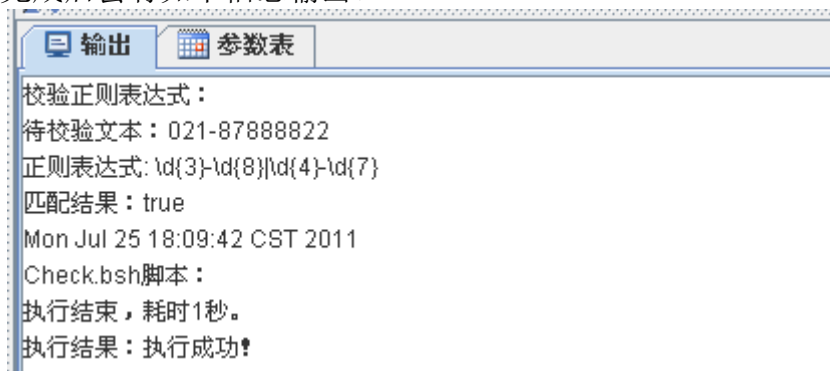
第一栏中填入待校验文本;

第二栏中填入正则表达式;

点击确定后, 在脚本编辑器中光标位置处会自动添加如下一行脚本。

```
ar.checkRegex("021-87888822", "\\d{3}-\\d{8}|\\d{4}-\\d{7}");
```

回放完成后会有如下信息输出。



### 3.10 脚本串联

脚本的串联支持不同脚本文件之间的相互调用。callScript 命令为脚本串联命令，下面就以计算器为例介绍一下串联脚本的使用：

编写或录制两个脚本（名称分别为 WinA.bsh、WinB.bsh）。

下面是 WinA.bsh 的录制内容，这段脚本是简单点击计算器按钮 1、2、3。

```

1 ar.window("SciCalc_计算器").clickControl("Button_1", 19, 12, "left");
2 ar.window("SciCalc_计算器").clickControl("Button_2", 11, 9, "left");
3 ar.window("SciCalc_计算器").clickControl("Button_3", 21, 10, "left");
  
```

下面是 WinB.bsh 的录制内容，这段脚本是简单点击计算器按钮 4、5、6。

```

1 ar.window("SciCalc_计算器").clickControl("Button_4", 26, 12, "left");
2 ar.window("SciCalc_计算器").clickControl("Button_5", 27, 14, "left");
3 ar.window("SciCalc_计算器").clickControl("Button_6", 19, 12, "left");
  
```

如果想在运行 WinA.bsh 脚本时调用 WinB.bsh 脚本，则在 WinA.bsh 里这样调用，当执行第四句时，就会跳转到 WinB 脚本去执行 WinB 脚本的动作

```

1 ar.window("SciCalc_计算器").clickControl("Button_1", 19, 11, "left");
2 ar.window("SciCalc_计算器").clickControl("Button_2", 17, 11, "left");
3 ar.window("SciCalc_计算器").clickControl("Button_3", 22, 11, "left");
4 ar.callScript("WinB.bsh");
  
```

注：


A、脚本串联只能在同一项目下的脚本之间，不支持跨项目串联脚本。

- B、两个脚本可以是对同一个窗口进行，也可以操作不同的窗口。
- C、脚本之间不能互调，比如在 a 脚本中调用了 b，那么在 b 脚本中可以再调用 c 脚本，而绝不能调用 a 脚本，否则会使回放进入死循环，所以在脚本串联当中不能形成调用环。
- D、如两个脚本是对同一个窗口进行操作，就要注意调用脚本命令的放置位置，保证调用时各对象的属性和对象库中的对象属性一致。


### 3.11 脚本参数化

一条普通脚本只能执行某个特定的动作，将脚本参数化后则可以执行不同的功能。脚本参数化之前，必需要编辑好参数表，具体编辑方法请查看参数表编辑一节。

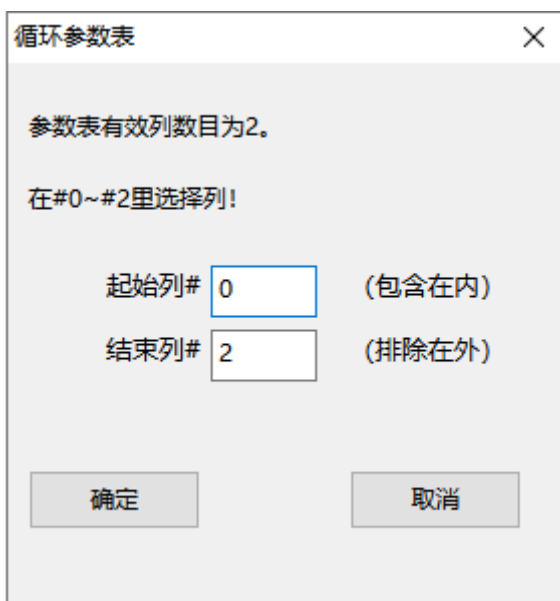
下面是一段简单的参数表数据。



*	名称	#0	#1
1	btnA	1	2
2	btnB	7	9

接下来是增加参数化脚本循环体。点击菜单【编辑】→【循环参数表】或直接点击工具栏的循环参数表按钮或在脚本编辑器中右击，会弹出如下信息面板。





他会从参数表中提取信息，注意小括号中的文字，起始列是包含在内的，而结束列被排除在外。点击确定后脚本编辑器中会添加如下循环参数表代码，这是一个循环结构的空壳，它和参数表关联，参数表中的数据都是保存在和脚本同名的 xls 表中的。

```

1  for(ParameterData pd : ar.getParameterDataList("Param.xls").subList(0, 2)) {
2
3  }

```

下面是一段简单的计算器加法操作脚本

```

1  ar.window("SciCalc_计算器").clickControl("Button_C", 39, 16, "left");
2  ar.window("SciCalc_计算器").clickControl("Button_1", 16, 14, "left");
3  ar.window("SciCalc_计算器").clickControl("Button_+", 15, 15, "left");
4  ar.window("SciCalc_计算器").clickControl("Button_2", 25, 14, "left");
5  ar.window("SciCalc_计算器").clickControl("Button_=", 16, 14, "left");

```

下面是将脚本的第二句和第四句进行参数化，使它执行上面参数表中的 1+7、2+9 的加法操作，参数化脚本如下：

```

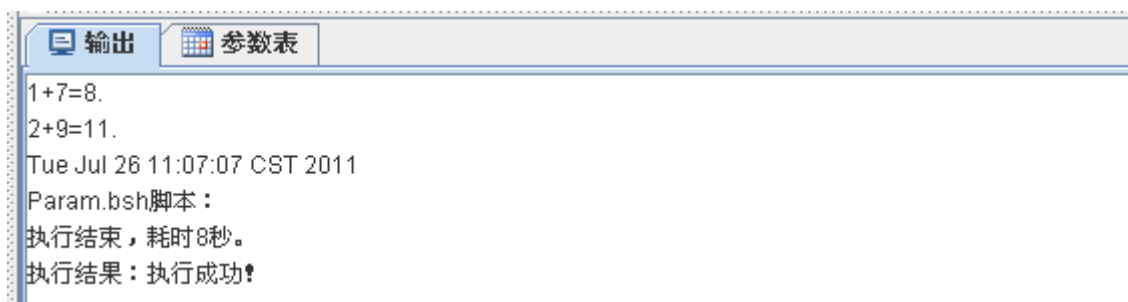
1  for(ParameterData pd : ar.getParameterDataList("Param.xls").subList(0, 2)) {
2      ar.window("SciCalc_计算器").clickControl("Button_C", 39, 16, "left");
3      ar.window("SciCalc_计算器").clickControl("Button_" + pd.getFrom("btnA"), 16, 14, "left");
4      ar.window("SciCalc_计算器").clickControl("Button_+", 15, 15, "left");
5      ar.window("SciCalc_计算器").clickControl("Button_" + pd.getFrom("btnB"), 25, 14, "left");
6      ar.window("SciCalc_计算器").clickControl("Button_=", 16, 14, "left");
7      System.out.println(pd.getFrom("btnA") + "+" + pd.getFrom("btnB") + "=" + ar.window("SciCalc_计算器
8  }

```

第二句实际上是将 Button\_1 拆分为 Button\_和 pd.getFrom("btnA") 的组合，在每次循环中 pd 的数据都不同，第一个调用 pd.getFrom("btnA") 获得的值为 1，第

二次获取到的值为 2，所以第一次循环点击的是按钮 1，而第二次点击的是按钮 2。第四句脚本的参数化功能类似。最后一句是输出计算结果。

结果输出如下



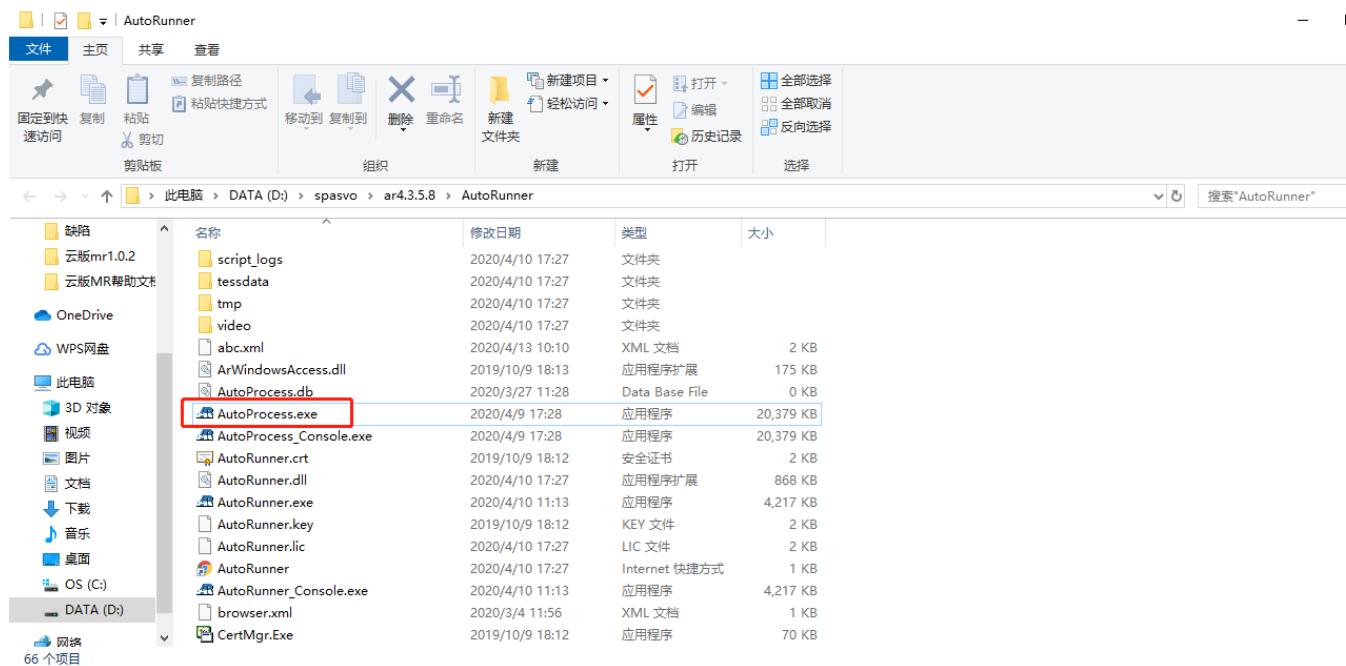
## 4 AutoRunner Process

### 4.1 产品介绍

● RPA (Robotic Process Automation) 机器人流程自动化，是一种能够模拟人类来执行重复性任务的软件。它所依赖的技术旨在统筹安排、执行并提升业务 workflow。业务用户只需负责通过图形方式显示的计算机操作界面对 RPA 软件进行编程和动态设定即可。RPA 机器人流程自动化，是一种能够模拟人类来执行重复性任务的软件。它所依赖的技术旨在统筹安排、执行并提升业务 workflow。业务用户只需负责通过图形方式显示的计算机操作界面对 RPA 软件进行编程和动态设定即可。

### 4.2 产品安装

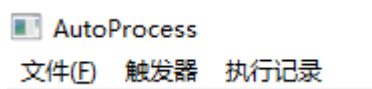
● 不需要独立安装，只需要在安装完 AutoRunner 的基础上，在 AutoRunner 的安装目录下打开 AutoProcess.exe 程序文件即可，如图所示。



## 4.3 用户界面

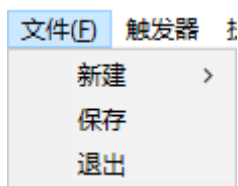
(Integrated Development Environment 简称 IDE) 软件是用于程序开发环境的应用程序，一般包括代码编辑器、编译器、调试器和图形用户界面工具，也就是集成了代码编写功能、分析功能、编译功能、Debug 功能等一体化的开发软件套。所有具备这一特性的软件或者软件套(组)都可以叫做 IDE。如微软的 Visual Studio 系列，Borland 的 C++ Builder、Delphi 系列等。

### 4.3.1 菜单栏



AutoProcess 中的菜单栏如上图所示，主菜单包含文件、触发器和执行记录等菜单项，下面对每一个菜单项做一个简单的介绍。

#### 1、文件菜单

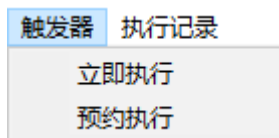


如上所示，所有对流程的管理操作都可以在文件菜单下完成，包括对流程的

新建，保存和退出等等。

- 新建：可选择新建项目或者新建流程；
- 保存：保存所创建好的项目或流程；
- 退出：退出 AutoProcess 程序；

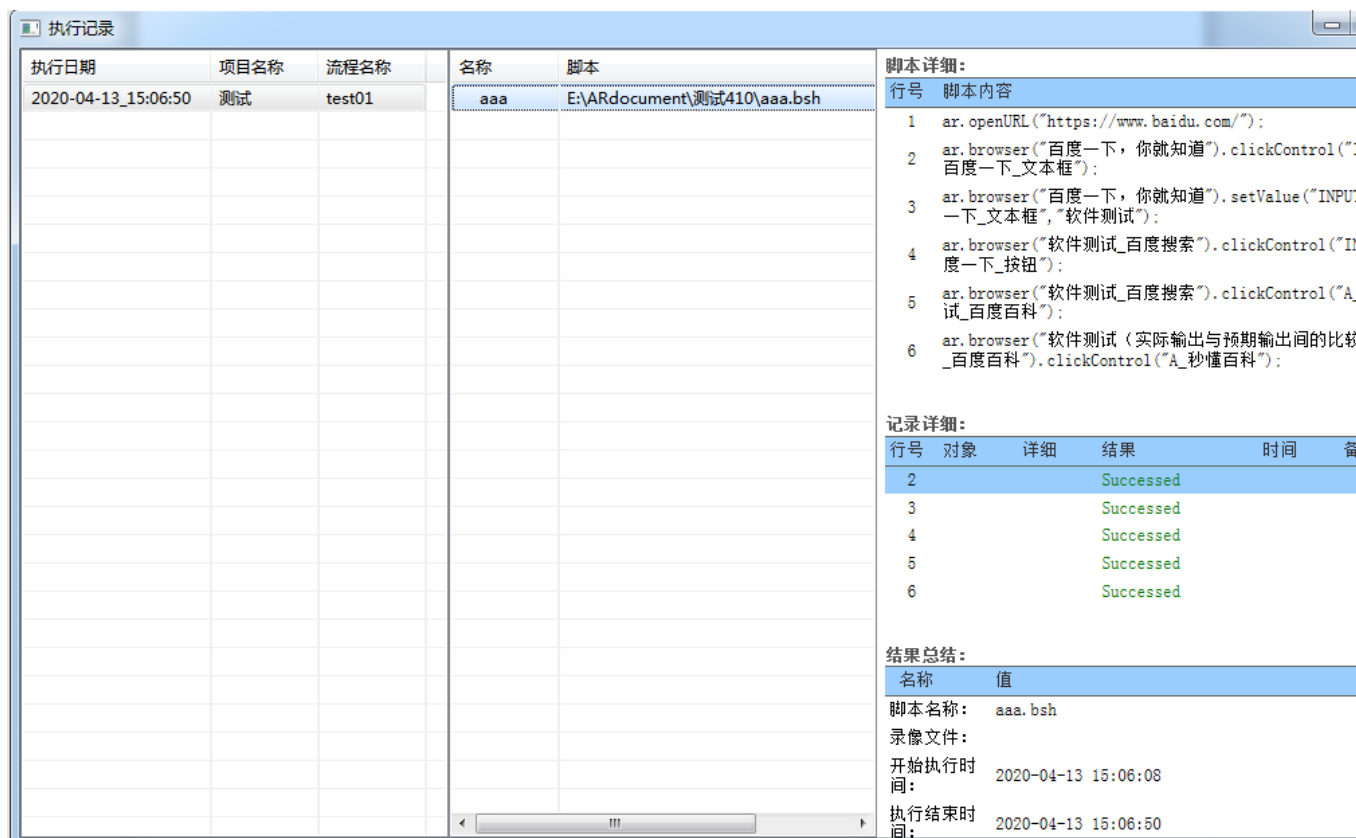
## 2、触发器菜单



如上图所示，触发器控制流程脚本的执行时间，包括立即执行和预约执行两个操作。

- 立即执行：从选中流程脚本点击立即执行的那一刻起立即执行流程脚本；
- 预约执行：在弹出的时间选择框中选择预约时间，到指定的预约时间时才会执行流程脚本；

## 3、执行记录菜单



如上所示，点击执行记录菜单弹出执行记录页面，执行记录包括流程、脚本和执行详细等内容。

- 流程：包括执行日期、项目名称和流程名称；
- 脚本：包括脚本名称，脚本和状态；
- 执行详细：包括脚本详细、记录详细和结果总结；

### 4.3.2 工具栏

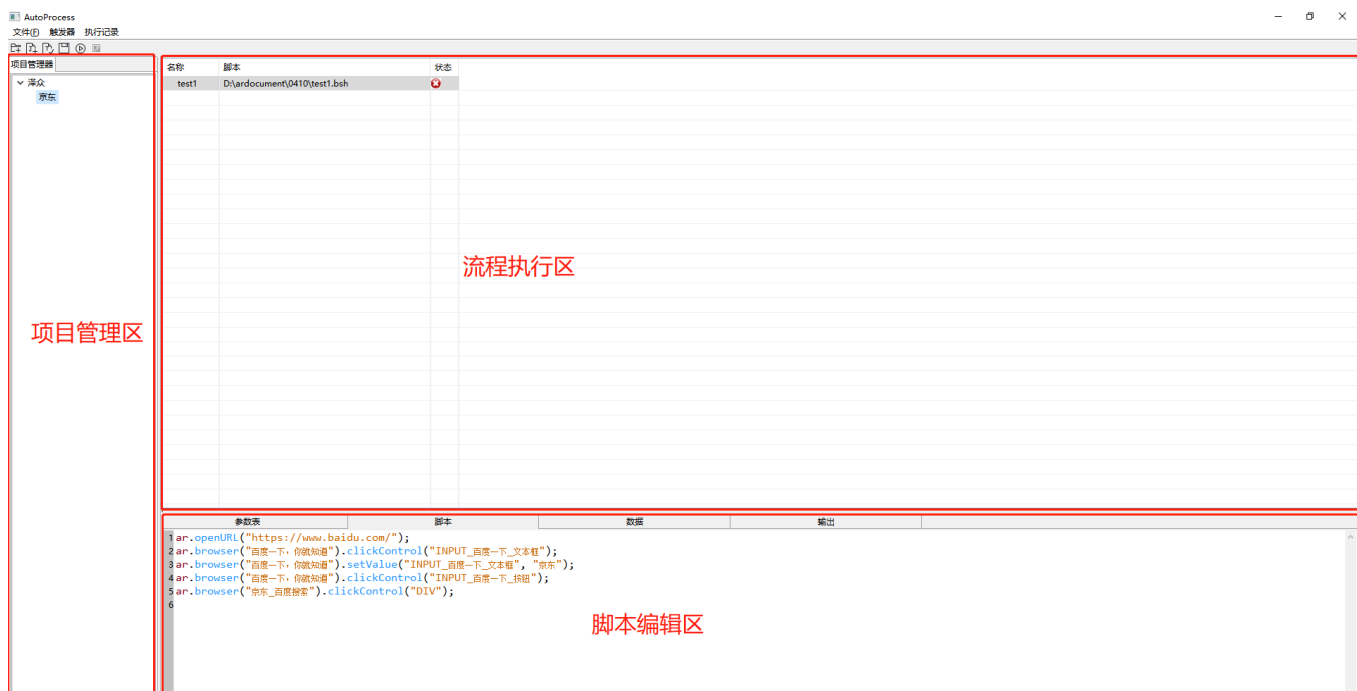


1 2 3 4 5 6

- 如上图所示，工具栏共有 6 个按钮，下面简单介绍各个按钮其功能。  
按钮 1：新建项目，和【文件】→【新建】→【项目】菜单功能一样；  
按钮 2：新建流程，和【文件】→【新建】→【流程】菜单功能一样；  
按钮 3：导入脚本，从保存脚本的目录将脚本导入至 AutoProcess；  
按钮 4：保存（快捷键 Ctrl+S），和【文件】→【保存】菜单功能一样；  
按钮 5：立即执行，和【触发器】→【立即执行】菜单功能一样；  
按钮 6：停止执行，中断流程脚本的执行过程；

### 4.3.3 工作区

- 项目管理区：创建项目，创建流程，进行项目浏览，切换对象浏览，在 AutoProcess 中位于垂直拆分条的左边；流程执行区：导入的脚本在此块区域组成执行流程，每个脚本有执行成功或者失败两种状态，在脚本的状态一栏会有两种状态的标志，在 AutoProcess 中位于水平拆分条的上部；脚本编辑区：包括参数表、脚本、数据和输出四个模块，可以对脚本进行编辑，在 AutoProcess 中位于水平拆分条的下部。



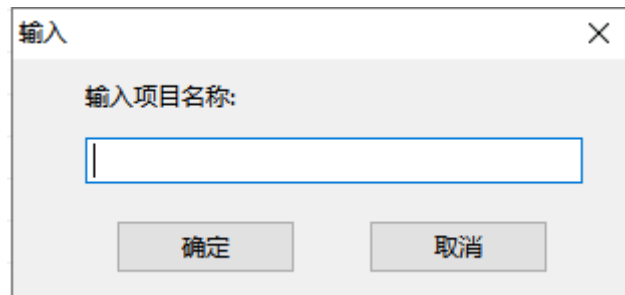
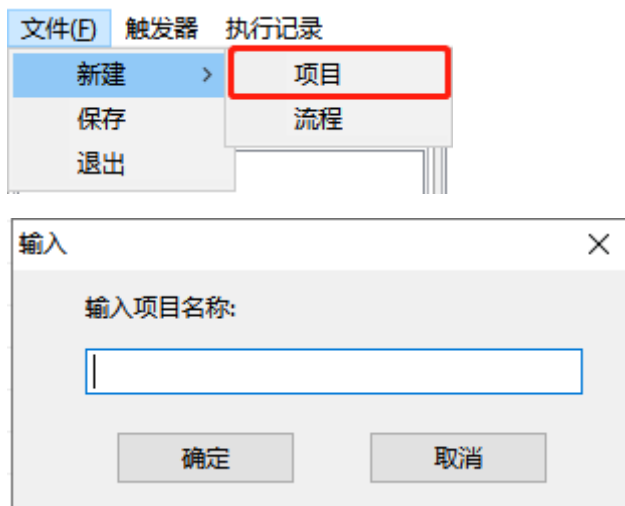
### ● 【项目管理器】

项目管理器用来显示当前 IDE 中所有的项目，并且显示项目中的流程。项目管理器中的项目及流程组织成一个树状结构，每一个项目名称是一个文件夹，其下的流程都位于此文件夹下。对于每一个节点，如果是项目名称，点击项目名称旁边的小三角可以展开；如果是流程，则单击选中可以把这个流程在流程执行器中打开。树支持鼠标右键菜单，支持新增、修改和删除等操作。







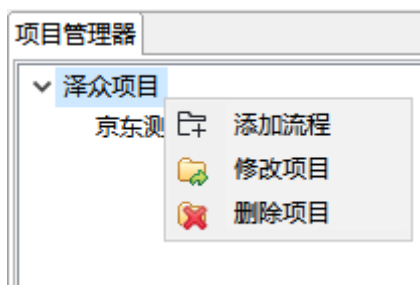


方式二：在项目管理器空白区域点击鼠标右键，在弹出的菜单中选中新建项目。



### ● 修改项目、删除项目

方法：选中一个项目，鼠标放在项目上点击右键，弹出菜单

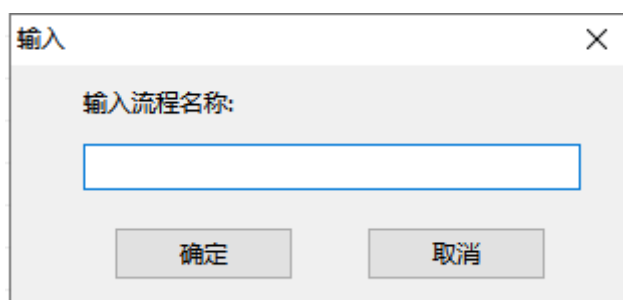
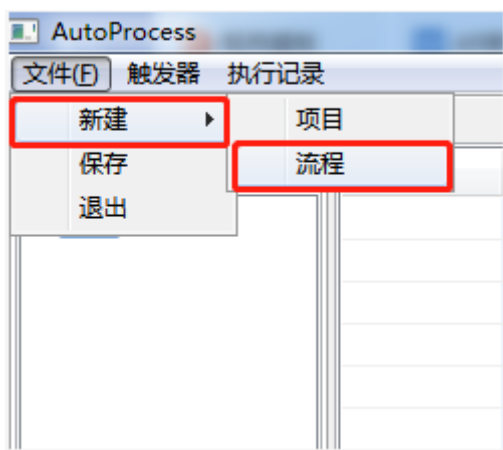


#### 4.4.2 新建 AutoRunner Process 流程

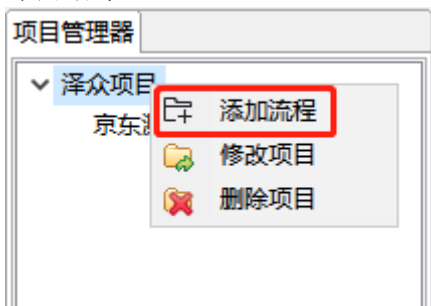
##### ● 新建流程

方式一：

如下图，在文件菜单栏下，点击【新建】-【流程】，弹出新建流程对话框，输入流程名称点击确定即可新建一个流程，注意此操作的前提是已存在项目，且需要选中该项目，才能完成新建流程，否则流程按钮置灰无法新建。



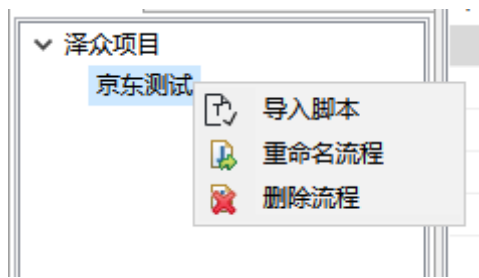
方式二：在项目管理器中选择一个项目点击鼠标右键，在弹出的菜单中选中添加流程。



##### ● 重命名流程、删除流程

方法：选中一个流程，鼠标放在流程上点击右键，弹出菜单，在弹出的菜单

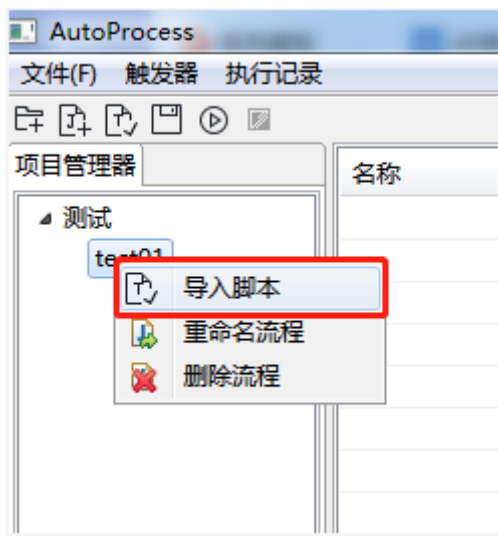
中选择重命名流程、删除流程。

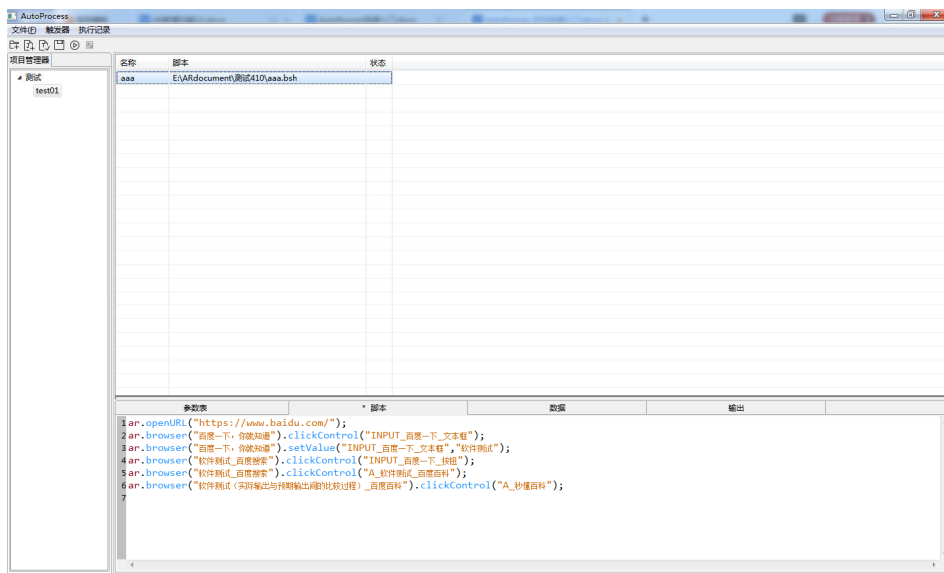
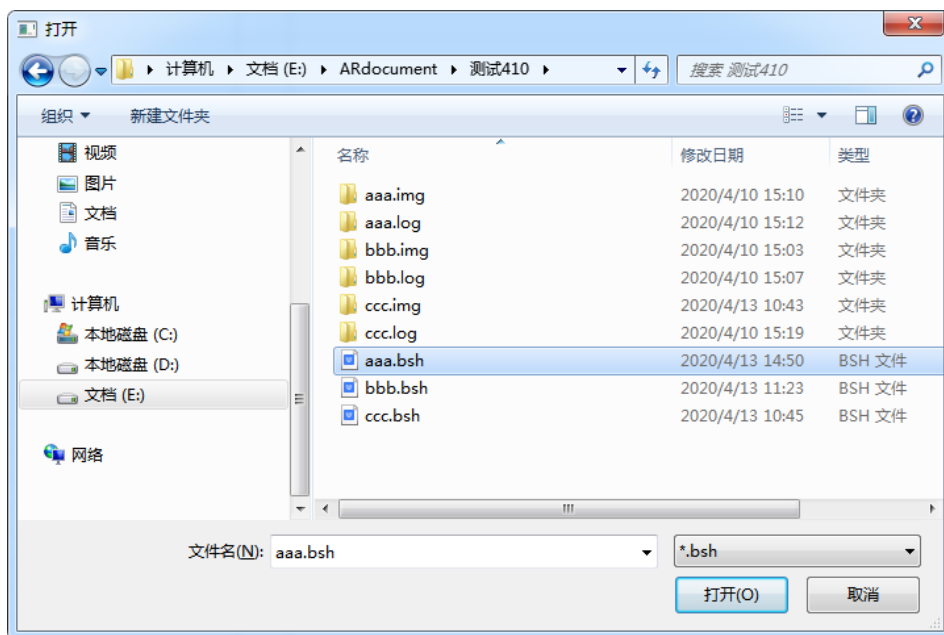


#### 4.4.3 导入脚本

##### ● 导入脚本

方法：选中一个流程，鼠标放在流程上点击右键，在弹出菜单中选中【导入脚本】，在文件选择框中选择已录制完成要导入的脚本文件，点击打开，脚本成功导入。



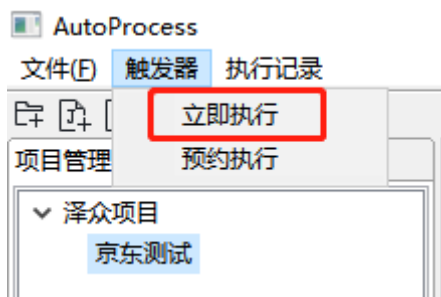


#### 4.4.4 执行流程脚本

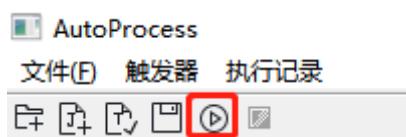
##### ● 立即执行流程脚本

方式一：

如下图，在触发器菜单栏下，点击【触发器】-【立即执行】，即可立即开始执行流程。

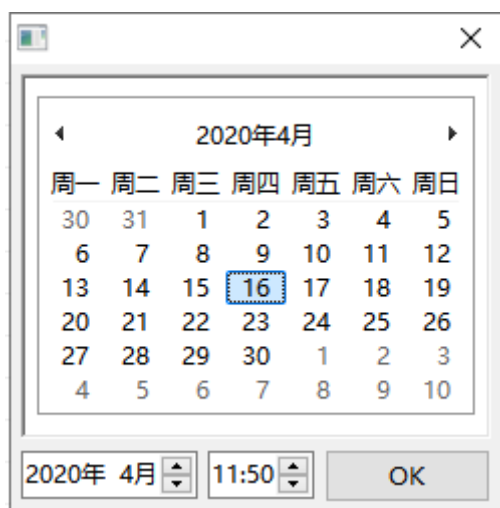
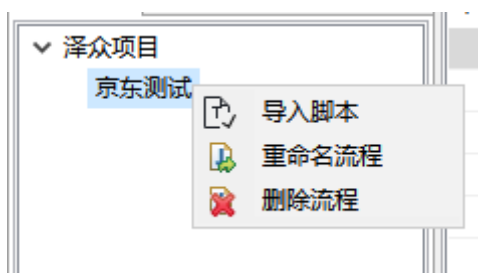


方式二：在工具栏中，点击下图中的图标，即可立即开始执行流程。



### ● 预约执行流程脚本

如下图，在触发器菜单栏下，点击【触发器】-【预约执行】，弹出时间框选择想要预约的时间，到预约时间才会执行流程。

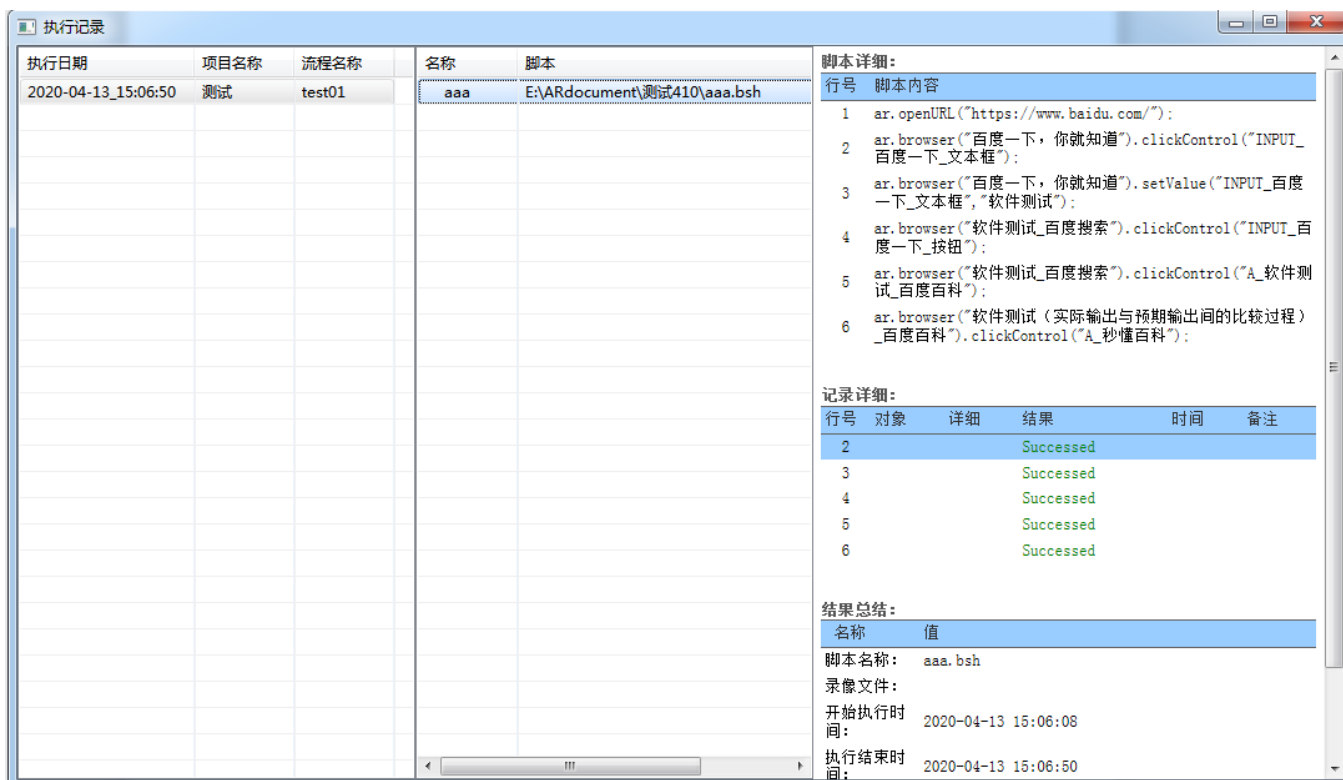


### ● 执行步骤

1. 点击开始执行按钮，开始执行脚本

2. AutoProcess 界面隐藏，系统回放脚本

3. 回放结束，点击导航栏【执行记录】，生成脚本执行记录，截图如下：



4. 如果回放成功，脚本一栏的执行状态会显示为绿色执行通过的状态，截图如下：

名称	脚本	状态
aaa	E:\ARdocument\测试410\aaa.bsh	

5. 如果回放失败，备注栏会备注相关脚本错误信息且脚本一栏的执行状态会



命令举例: `addClassPath("C:\\test\\test.jar");` // [代码示例](#)

相关命令: [source](#)

### 5.3 beginTime

命令含义: 时间统计命令, 开始记录时间统计对象。

命令参数: 一个参数, 时间统计对象名。

命令返回: true, 表示执行成功; false, 表示失败, 统计时间对象名重复。

命令产生: 手工添加。

命令举例: `ar.beginTime("Object1");`

相关命令: [endTime](#)

### 5.4 callScript

命令含义: 脚本串联调用命令。

命令参数: 一个参数, 要求输入调用脚本全名称。

命令产生: 在脚本之间调用时出现此命令, 手工添加或点击编辑菜单中的【调用脚本】来添加。

命令举例: `ar.callScript("Win.bsh");`

### 5.5 captureScreen

命令含义: 抓取屏幕。

命令参数: 无

命令产生: 手工添加, jpg 图片保存在脚本目录下, 以脚本名加日期的形式保存。

命令举例: `ar.captureScreen();`

### 5.6 checkDatabase

命令含义: 校验数据库。

命令参数: 六个参数, 第一个参数输入数据库类型, 第二个参数输入数据库地址, 第三个参数输入数据库访问用户名, 第四个参数输入数据库访问密码, 第五个参数输入数据库查询语句, 第六个参数输入校验期望值。

命令产生: 手工添加。



命令返回：校验成功返回 true，否则返回 false

命令举例：

例 1，校验单个字符串返回值

```
ar.checkDatabase("SQL Server", "192.168.1.50:12345/mydb", "spasvo",  
"123", "SELECT name FROM students WHERE id=45", "Zhang san");
```

例 2，校验数组形式的返回值

```
String [] strExpect = {"Zhang san", "男", "18"};  
ar.checkDatabase("SQL Server", "192.168.1.50:12345/mydb", "spasvo",  
"123", "SELECT name, sex, age FROM students WHERE id=45", strExpect);
```

例 3，校验二维表形式的返回值

```
String [][] strExpect = {"Zhang san", "男", "18"}, {"Li si", "男",  
"20"}};  
ar.checkDatabase("SQL Server", "192.168.1.50:12345/mydb", "spasvo",  
"123", "SELECT name, sex, age FROM students WHERE id > 45 AND id <  
50", strExpect);
```

相关命令：[getDatabase](#)、[modifyDatabase](#)

## 5.7 checkExcelCell

命令含义：校验 Excel 单元格文本。

命令参数：四个参数，第一个参数输入 Excel 文件全路径，第二个参数输入欲检验的 sheet 页（从 0 开始），第三个参数输入单元格行号（从 0 开始），第四个参数输入单元格列号（从 0 开始），第五个参数输入期望值。

命令产生：手工添加或向导添加。

命令返回：校验成功返回 true，否则返回 false

命令举例：`ar.checkExcelCell("C:\\\\Test.xls", 0, 0, 0, "Spasvo");`

相关命令：[getExcelCell](#)

## 5.8 checkExcelColumn

命令含义：校验 Excel 某列文本。

命令参数：三个参数，第一个参数输入 Excel 文件全路径，第二个参数输入单元格列号(从 0 开始)，第三个参数输入期望值(字符串一维数组)。

命令产生：手工添加或向导添加。

命令返回：校验成功返回 true，否则返回 false

命令举例：

```
String [] strColumn = {"18", "18", "19"};  
ar.checkExcelColumn("C:\\Test.xls", 2, strColumn);
```

相关命令：[getExcelColumn](#)

## 5.9 checkExcelRow

命令含义：校验 Excel 某行文本。

命令参数：三个参数，第一个参数输入 Excel 文件全路径，第二个参数输入欲检验的 sheet 页（从 0 开始），第三个参数输入单元格行号(从 0 开始)，第四个参数输入期望值(字符串一维数组)。

命令产生：手工添加或向导添加。

命令返回：校验成功返回 true，否则返回 false

命令举例：

```
String [] strRow = {"Name", "Sex", "Age"};  
ar.checkExcelRow("C:\\Test.xls", 0, 0, strRow);
```

相关命令：[getExcelRow](#)

## 5.10 checkExcelWhole

命令含义：校验 Excel 文本。

命令参数：两个参数，第一个参数输入 Excel 文件全路径，第二个参数输入期望值(字符串二维数组)。

命令产生：手工添加或向导添加。

命令返回：校验成功返回 true，否则返回 false

命令举例：

```
String [][] strCheck = {"Zhang san", "男", "18"}, {"Li si", "男",
```

```
"20"}}};  
ar.checkExcelWhole("C:\\Test.xls", strCheck);  
相关命令: getExcelWhole
```

## 5.11 checkFileText

命令含义: 校验文件文本。

命令参数: 五个参数, 第一个参数输入文本格式, 第二个参数输入文件全路径, 第三个参数输入行号, 第四个参数输入列号, 第五个参数输入期望值。

命令产生: 手工添加。

命令返回: 校验成功返回 true, 否则返回 false

命令举例: `ar.checkFileText("ANSI", "c:\\test\\test.txt", 1, 1, "a");`

相关命令: [getFileText](#)

## 5.12 checkImage

命令含义: 校验图片, 对比两个图片的相似度。

命令参数: 六个参数, 第一个参数输入屏幕 X 坐标, 第二个参数输入屏幕 Y 坐标, 第三个参数输入图片宽度, 第四个参数输入图片高度, 第五个参数输入图片的保存地址(全路径), 第六个参数输入相似度(建议填 80-95 直接, 成功率会比较高)。

命令产生: 手工添加。

命令返回: 校验成功返回 true, 否则返回 false

命令举例: `ar.checkImage(70, 80, 174, 166, "C:\\Users\\Administrator\\Desktop\\test1.bmp", 80);`

说明: 图片获取可使用 AR 自带的图片对象截取工具, 使用该工具可获取图片的屏幕坐标及高度宽度等信息。

## 5.13 checkMessageBox

命令含义: 校验消息框文本。

命令参数: 两个参数, 第一个参数输入消息框对象名, 第二个参数输入期望值。

命令产生：手工添加。

命令返回：校验成功返回 true，否则返回 false

命令举例：`ar.checkMessageBox("#32770_About", "T1 Application v1.0");`

相关命令：[getMessageBox](#)

## 5.14 checkProperty

命令含义：校验对象属性。

命令参数：三个参数，第一个参数输入对象名或对象的 ObjectElement，第二个参数输入待检验属性名，第三个参数输入待检验属性的期望值。

命令产生：手工添加。

命令返回：校验成功返回 true，否则返回 false

命令举例：`ar.window("SciCalc_计算器").checkProperty("Edit", "value", "10");`

相关命令：[getProperty](#)

## 5.15 checkRectText

命令含义：校验矩形文本。

命令参数：六个参数，第一个参数输入对象名或对象的 ObjectElement，接下来四个参数输入待校验文本的左、顶、右、底坐标，第六个参数输入期望值。

命令产生：手工添加。

命令返回：校验成功返回 true，否则返回 false

命令举例：`ar.window("SciCalc_计算器").checkRectText("SciCalc", 0, 0, 100, 100, "编辑");`

相关命令：[getRectText](#)

## 5.16 checkRegex

命令含义：校验正则表达式。

命令参数：两个参数，第一个参数输入校验文本，第二个参数输入正则表达式。

命令产生：手工添加。

命令返回：校验成功返回 true，否则返回 false

命令举例：`ar.checkRegex("021-87888822",  
"\\d{3}-\\d{8}|\\d{4}-\\d{7}")`;

## 5.17 clickControl

命令含义：点击窗口中的某一对象。

命令参数：四个参数，第一个参数输入对象名或对象的 ObjectElement，第二、三两个参数输入点击的位置（相对对象本身的坐标，有默认值 5、5），第四个参数输入点击方式（有默认值"left"）。

命令产生：点击对象（比如按钮）或菜单时出现此命令。

命令举例：`ar.window("SciCalc_计算器").clickControl("Button_1", 20,  
13, "left")`;

`ar.window("SciCalc_计算器").clickControl("Button_1", 20, 13)`;

`ar.window("SciCalc_计算器").clickControl("Button_1")`;

相关命令：[dragControl](#)

## 5.18 clickScreen

命令含义：点击屏幕。

命令参数：三个参数，第一个参数输入点击位置的 X 坐标，第二个参数输入点击位置的 Y 坐标，第三个参数输入点击方式（有默认值"left"）。

命令产生：手动添加。

命令举例：`ar.clickScreen(10, 20, "left")`;

`ar.clickScreen(10, 20)`;

相关命令：[dragScreen](#)

## 5.19 closeLowLog

命令含义：关闭底层 log 功能。

命令参数：无。

命令产生：手动添加。

命令举例：`ar.closeLowLog()`;

相关命令：[openLowLog](#)

## 5.20 collapse

命令含义：收起某一个节点。

命令参数：五个参数，第一个参数输入对象名或对象的 `ObjectElement`，第二个参数输入收起节点的全路径（层与层之间用 `\r` 分隔），第三、四两个参数输入点击的位置（相对对象本身的坐标，有默认值 5、5），第五个参数输入点击方式（有默认值“left”）。

命令产生：在树对象中收起节点时出现此命令。

命令举例：`ar.window("CabinetWClass_ 我的电脑").collapse("SysTreeView32", "桌面\r我的电脑\rSystem(C:)", 20, 13, "left");`

`ar.window("CabinetWClass_我的电脑").collapse("SysTreeView32", "桌面\r我的电脑\rSystem(C:)", 20, 13);`

`ar.window("CabinetWClass_我的电脑").collapse("SysTreeView32", "桌面\r我的电脑\rSystem(C:)");`

相关命令：[expand](#)

## 5.21 controlExist

命令含义：判断控件是否存在。

命令参数：一个参数，输入对象库中存在的待检查控件的对象名或控件的 `ObjectElement`

命令产生：手工添加。

命令返回：存在返回 `true`，否则返回 `false`

命令举例：`ar.window("SciCalc_计算器").controlExist("Button_1");`

相关命令：[menuExist](#)、[windowExist](#)

## 5.22 dragControl

命令含义：在对象上拖拉。

命令参数：六个参数，第一个参数输入对象名或对象的 ObjectElement，第二、三个参数输入鼠标按下的位置（相对对象本身的坐标），第四、五个参数输入鼠标弹起的位置（相对对象本身的坐标），第六个参数输入拖拉方式（有默认值“left”）。

命令产生：在鼠标拖拉时出现此命令。

命令举例：`ar.window("SciCalc_计算器").dragControl("SciCalc_计算器", 51, 170, 50, 196, "left");`

`ar.window("SciCalc_计算器").dragControl("SciCalc_计算器", 51, 170, 50, 196);`

相关命令：[clickControl](#)

## 5.23 dragScreen

命令含义：在屏幕上拖拉。

命令参数：五个参数，第一、二个参数输入鼠标按下的位置（相对屏幕本身的坐标），第三、四个参数输入鼠标弹起的位置（相对屏幕本身的坐标），第五个参数输入拖拉方式（有默认值“left”）。

命令产生：手动添加。

命令举例：`ar.dragScreen(51, 170, 50, 196, "left");`

`ar.dragScreen(51, 170, 50, 196);`

相关命令：[clickScreen](#)

## 5.24 endTime

命令含义：时间统计命令，记录该统计时间对象的从 beginTime 调用后到当前的时间差，并打印统计信息到输出界面。

命令参数：一个参数，时间统计对象名。

命令返回：指定时间统计对象的统计时间，单位毫秒；没有该对象则返回 0。

命令产生：手工添加。

命令举例：

`ar.beginTime("Object1"); //1、开始统计 Object1`

.....

`//2、其他脚本语句`

```
ar.endTime("Object1"); //3、返回 Object1 从 1 到 3 的执行时间
..... //4、其他脚本语句
ar.endTime("Object1"); //5、返回 Object1 从 1 到 5 的执行时间
```

相关命令: [beginTime](#)

## 5.25 expand

命令含义: 展开某一个节点。

命令参数: 五个参数, 第一个参数输入对象名或对象的 ObjectElement, 第二个参数输入展开节点的全路径(层与层之间用\r 分隔), 第三、四两个参数输入点击的位置(相对对象本身的坐标, 有默认值 5、5), 第五个参数输入点击方式(有默认值"left")。

命令产生: 在树对象中展开节点时出现此命令。

命令举例: `ar.window("CabinetWClass_ 我的电脑").expand("SysTreeView32", "桌面\r 我的电脑\r System(C:)", 20, 13, "left");`

```
ar.window("CabinetWClass_我的电脑").expand("SysTreeView32", "桌面\r 我的电脑\r System(C:)", 20, 13);
```

```
ar.window("CabinetWClass_我的电脑").expand("SysTreeView32", "桌面\r 我的电脑\r System(C:)");
```

相关命令: [collapse](#)

## 5.26 getCaptureScreenWhenError

命令含义: 获取某句脚本执行失败时是否截屏的状态标记。

命令参数: 无

命令产生: 手工添加。

命令返回: 脚本执行失败时会触发截屏返回 true, 否则返回 false

命令举例: `boolean b = ar.getCaptureScreenWhenError();`

相关命令: [setCaptureScreenWhenError](#)



## 5.27 getDatabase

命令含义：查询数据库。

命令参数：五个参数，第一个参数输入数据库类型，第二个参数输入数据库地址，第三个参数输入数据库访问用户名，第四个参数输入数据库访问密码，第五个参数输入数据库查询语句。

命令产生：手工添加。

命令返回：返回查询结果。

命令举例：

```
String [][] str = ar.getDatabase("SQL Server",  
"192.168.1.50:12345/mydb", "spasvo", "123", "SELECT name, sex, age  
FROM students WHERE id > 45 AND id < 50");
```

相关命令：[checkDatabase](#)、[modifyDatabase](#)

## 5.28 getExcelCell

命令含义：获取 Excel 单元格文本。

命令参数：三个参数，第一个参数输入 Excel 文件全路径，第二个参数输入单元格行号(从 0 开始)，第三个参数输入单元格列号(从 0 开始)。

命令产生：手工添加或向导添加。

命令返回：以字符串形式返回指定单元格文本。

命令举例：`String str = ar.getExcelCell("C:\\Test.xls", 0, 0);`

相关命令：[checkExcelCell](#)

## 5.29 getExcelColumn

命令含义：获取 Excel 某列文本。

命令参数：两个参数，第一个参数输入 Excel 文件全路径，第二个参数输入单元格列号(从 0 开始)。

命令产生：手工添加或向导添加。

命令返回：以一维字符串数组形式返回指定列文本。

命令举例：`String [] str = ar.getExcelColumn("C:\\Test.xls", 2);`

相关命令: [checkExcelColumn](#)

### 5.30 getExcelRow

命令含义: 获取 Excel 某行文本。

命令参数: 两个参数, 第一个参数输入 Excel 文件全路径, 第二个参数输入单元格行号(从 0 开始)。

命令产生: 手工添加或向导添加。

命令返回: 以一维字符串数组形式返回指定行文本。

命令举例: `String [] str = ar.getExcelRow("C:\\Test.xls", 0);`

相关命令: [checkExcelRow](#)

### 5.31 getExcelWhole

命令含义: 获取 Excel 文本。

命令参数: 一个参数, 第一个参数输入 Excel 文件全路径。

命令产生: 手工添加或向导添加。

命令返回: 以二维字符串数组形式返回指定的 Excel 文件文本。

命令举例: `String [][] str = ar.getExcelWhole("C:\\Test.xls");`

相关命令: [checkExcelWhole](#)

### 5.32 getFileText

命令含义: 获取文件文本。

命令参数: 五个参数, 第一个参数输入文本格式, 第二个参数输入文件全路径, 第三个参数输入行号, 第四个参数输入列号, 第五个参数输入待获取的文本长度。

命令产生: 手工添加。

命令返回: 以 String 类型返回文件文本

命令举例: `String str = ar.getFileText("ANSI", "c:\\test\\test.txt", 1, 1, 1);`

相关命令: [checkFileText](#)

### 5.33 getFrom

命令含义: ParameterData 类中的成员函数, 获取指定参数的值。

命令参数: 一个参数, 输入参数名称。

命令返回: 指定参数的值, 以字符串返回。

命令产生: 手工添加。

命令举例: `ar.parameterData.getFrom("addResult");`

命令说明: 此命令用于跨脚本数据传递, 在一个脚本中通过 `putInto` 命令将数据保存到变量名为 `addResult` 的变量中 (变量名可以任取), 如下面例子所示, 在另一个脚本中, 调用 `getFrom` 函数, 如“命令举例”所示, 就可以重新获取到保存在 `addResult` 变量中的值。

相关命令: [putInto](#)

### 5.34 getMessageBox

命令含义: 获取消息框文本。

命令参数: 一个参数, 输入消息框对象名。

命令产生: 手工添加。

命令返回: 以 `String` 类型返回消息框文本

命令举例: `String str = ar.getMessageBox("#32770_About");`

相关命令: [checkMessageBox](#)

### 5.35 getParameterDataList

命令含义: 从 `xls` 表中获取数据列表。

命令参数: 一个参数, 输入当前项目中的 `xls` 文件名或不在当前项目中的全路径名。

命令返回: `LinkedList` 类型的数据链表。

命令产生: 手工添加。

命令举例:

```
1 List list = ar.getParameterDataList("C:\\Users\\Administrator\\Desktop\\Book1.xls");
2 for (int i = 0; i < list.size(); i++) {
3     System.out.println(a.get(i).getFrom("key"));
4     System.out.println(a.get(i).getFrom("key1"));
5 }
```

说明：key 为 Excel 首列字符串

相关命令：[putParameterDataList](#)

## 5.36 getProperty

命令含义：获取对象属性。

命令参数：两个参数，第一个参数输入对象名或对象的 ObjectElement，第二个参数输入属性名。

命令产生：手工添加。

命令返回：以 String 类型返回属性文本

命令举例：`String str = ar.window("SciCalc_ 计算器").getProperty("Edit", "value");`

相关命令：[checkProperty](#)

## 5.37 getRectText

命令含义：获取矩形内的文本。

命令参数：五个参数，第一个参数输入对象名或对象的 ObjectElement，后四个参数依次输入矩形的左上角、右下角坐标（相对于对象的坐标）。

命令产生：手工添加。

命令返回：以 String 类型返回矩形内的文本

命令举例：`String str = ar.window("SciCalc_ 计算器").getRectText("Button_1", 0, 0, 50, 20);`

相关命令：[checkRectText](#)

## 5.38 getStopWhenError

命令含义：获取如果运行出错是否立即停止。

命令参数：无。

命令返回：布尔值（true 或 false）。

命令产生：手工添加。

命令举例：`boolean bStop = ar.getStopWhenError();`

相关命令：[setStopWhenError](#)

### 5.39 getSynchronizationTime

命令含义：获取同步时间，单位毫秒，默认 3000。

命令参数：无。

命令返回：整型数值。

命令产生：手工添加。

命令举例：`int iSyncTime = ar.getSynchronizationTime();`

相关命令：[setSynchronizationTime](#)

### 5.40 getTimeSpan

命令含义：获取两条运行命令之间的时间间隔，单位毫秒，默认 500。

命令参数：无。

命令返回：整型数值。

命令产生：手工添加。

命令举例：`int iSpanTime = ar.getTimeSpan();`

相关命令：[setTimeSpan](#)

### 5.41 getY

命令含义：得到鼠标的默认的相对垂直偏移量。

命令参数：无。

命令产生：手工添加。

命令返回：整数类型的坐标值（默认时返回值为 5）。

命令举例：`int iPosY = ar.getY();`

相关命令：[setY](#)

## 5.42 ieControl

情况一：

命令含义：根据条件字符串搜索 IE 控件。

命令参数：两个参数，第一个参数输入 tagName 属性的值，第二个参数输入条件字符串。

命令产生：手工添加。

命令举例：`ar.window("IEFrame_Google - Microsoft Internet Explorer").ieControl("INPUT", "className=lst");`

情况二：

命令含义：在第一种情况的基础上在它的附近去查找对象，相对位置关系由最后两个参数来指定

命令参数：四个参数，第一个参数输入 tagName 属性的值，第二个参数输入条件字符串，第三第四个参数输入相对偏移量。

命令产生：手工添加。

命令举例：`ar.window("IEFrame_Google - Microsoft Internet Explorer").ieControl("INPUT", "className=lst", 50, 30).clickControl();`

例子解释：当执行到 ieControl 命令时，AutoRunner 会在网页中查找 tagName 等于 INPUT 并且 className 等于 lst 的元素，找到之后再以这个元素的左上角为参考点，将 x 坐标偏移 50 个像素，y 坐标偏移 30 个像素，获取此位置下的元素（设元素为一个按钮），接下来执行的操作都是对这个按钮的操作，这里是一个 clickControl 命令，鼠标就点击了按钮。

相关命令：[ieWindow](#)

### 第二个参数说明

- 一、条件字符串对空格是敏感的；
- 二、条件字符串格式为：属性名称=属性值，其中属性名称只能是以下几个值中的一个（最常用的是 innerText name description）：

className、description、disabled、innerId、innerText、name、outerText、readOnly、type、urlLink

### 命令说明

ieControl 命令之后还有一级命令，格式为  
ar.window(...).ieControl(...).XXX(...);

此处 XXX 类似于 ar.window(...).xxx(...);XXX 比 xxx 要少第一个参数。

## 5.43 ieWindow

命令含义：激活窗口，并且窗口中必需包含指定控件。

命令参数：三个参数，第一个参数输入窗口对象名或窗口的 ObjectElement，第二个参数输入指定控件 tagName 属性的值，第三个参数输入指定控件的条件字符串。

命令产生：手工添加。

命令举例：ar.ieWindow("IEFrame\_Google - Microsoft Internet Explorer", "INPUT", "className=lst");

相关命令：[ieControl](#)

## 5.44 inputDown

命令含义：模拟键盘输入，只模拟键按下，没有模拟键弹起。

命令参数：一个参数，输入待按键的键名称。

命令产生：手工添加

命令举例：ar.inputDown("Enter");

ar.inputDown("F1");

相关命令：[inputUp](#)

## 5.45 inputKey

命令含义：模拟键盘输入，按键盘上的某个键（包含按下弹起两个动作）。

命令参数：一个参数，输入待按键的键名称。

命令产生：在录制时如果没有选中“记录击键”，则只对键盘上一些特殊按

键才会有这个命令出现，比如按下回车键或是 Alt+Tab 键等；如果选中“记录击键”，则每次进行键盘操作都会有这个命令出现。

命令举例：`ar.inputKey("Enter");`

`ar.inputKey("Alt+Ctrl+Del");`

`ar.inputKey("Ctrl+S");`

`ar.inputKey("F1");`

相关命令：[pressKey](#)

## 5.46 inputString

命令含义：模拟键盘输入一串字符。

命令参数：一个参数，输入字符串。

命令产生：手动添加。

命令举例：`ar.inputString("spasvo");`

相关命令：[pressString](#)

## 5.47 inputUp

命令含义：模拟键盘输入，只模拟键弹起，没有模拟键按下。

命令参数：一个参数，输入待按键的键名称。

命令产生：手工添加

命令举例：`ar.inputUp("Enter");`

`ar.inputUp("F1");`

相关命令：[inputDown](#)

## 5.48 loadObjectElement

命令含义：加载对象库中对象，创建对象的 ObjectElement。

命令参数：两个参数，第一个参数输入窗口对象名，第二个参数输入窗口中的控件对象名(如果只想获取窗口，此参数可填空)。

命令产生：手工添加。

命令返回：返回对象的 ObjectElement

命令举例：`ObjectElement oe = ar.loadObjectElement("SciCalc_计算器",`



```
null);  
        ObjectElement oe = ar.loadObjectElement("SciCalc_计算器",  
        "");  
        ObjectElement oe = ar.loadObjectElement("SciCalc_计算器",  
        "Button_1");
```

## 5.49 menu

命令含义：激活弹出菜单。

命令参数：一个参数，输入待激活的弹出菜单对象名或菜单的 ObjectElement。

命令产生：在选择某项弹出菜单时出现此命令。

命令举例：ar.menu("#32768\_查看(V)");

相关命令：[window](#)

## 5.50 menuExist

命令含义：判断菜单是否存在。

命令参数：一个参数，输入对象库中存在的待检查菜单的对象名或菜单的 ObjectElement

命令产生：手工添加。

命令返回：存在返回 true，否则返回 false

命令举例：ar.menuExist("#32768\_查看(V)");

相关命令：[controlExist](#)、[windowExist](#)

## 5.51 modifyDatabase

命令含义：修改数据库。

命令参数：五个参数，第一个参数输入数据库类型，第二个参数输入数据库地址，第三个参数输入数据库访问用户名，第四个参数输入数据库访问密码，第五个参数输入数据库语句。

命令产生：手工添加。

命令举例：ar.modifyDatabase("SQL Server", "192.168.1.50:1433/mydb",

”sa”, ”123456”, ”UPDATA students SET mark=70 WHERE id=45”);

相关命令: [checkDatabase](#)、[getDatabase](#)

## 5.52 mouseDown

命令含义: 按下鼠标

命令参数: 一个参数, 输入鼠标按下方式 (有默认值”left”)。

命令产生: 手工添加。

命令举例: `ar.mouseDown(”left”);`

`ar.mouseDown();`

相关命令: [mouseUp](#)

## 5.53 mouseMove

命令含义: 将鼠标移动到指定的对象。

情况一: 对于非容器类控件

命令参数: 三个参数, 第一个参数输入对象名或对象的 ObjectElement, 第二、三两个参数输入点击的位置 (相对对象本身的坐标, 有默认值 5、5)。

命令举例: `ar.window(”SciCalc_计算器”).mouseMove(”Button_1”, 20, 13);`

`ar.window(”SciCalc_计算器”).mouseMove(”Button_1”);`

情况二: 对于容器类控件

命令参数: 四个参数, 第一个参数输入对象名或对象的 ObjectElement, 第二个参数: 对于列表视图对象、列表对象、组合框对象, 输入的是选中的索引名称; 对于树对象, 输入的是选中的节点全路径(层与层之间用\r分隔), 第三、四两个参数输入点击的位置 (相对索引或节点本身的坐标, 有默认值 5、5)。

命令举例: `ar.window(”CabinetWClass_我的电脑”).mouseMove(”SysListView32”, ”textTmp.doc”, 57, 11);`

`ar.window(”CabinetWClass_我的电脑”).mouseMove(”SysListView32”,`

```
"textTmp.doc");  
ar.window("CabinetWClass_我的电脑").mouseMove("SysTreeView32", "桌面\r 我的电脑\rSystem(C:)", 29, 7);  
ar.window("CabinetWClass_我的电脑").mouseMove("SysTreeView32", "桌面\r 我的电脑\rSystem(C:)");
```

命令产生：手工添加。

## 5.54 mouseUp

命令含义：弹起鼠标

命令参数：一个参数，输入鼠标弹起方式（有默认值"left"）。

命令产生：手工添加。

命令举例：ar.mouseUp("left");

```
ar.mouseUp();
```

相关命令：[mouseDown](#)

## 5.55 ObjectElement

类名，创建一个 ObjectElement 对象可以调用 [loadObjectElement](#) 函数。

此类的相关命令函数如下（oe 是一个 ObjectElement 类对象）：

1、getPropertyValue

命令含义：获取属性值。

命令参数：一个参数，输入属性名称。

命令产生：手动添加。

命令返回：以字符串形式返回属性值。

命令举例：String strName = oe.getPropertyValue("name");

相关命令：setPropertyValue

2、setPropertyValue

命令含义：设置属性值。

命令参数：两个参数，第一个参数输入属性名称，第二个参数输入属性

值。

命令产生：手动添加。

命令举例：oe.setPropertyValue("name", "sss");

相关命令：getPropertyValue

### 3、getPropertyWeight

命令含义：获取属性权重。

命令参数：一个参数，输入属性名称。

命令产生：手动添加。

命令返回：以整形形式返回属性权重。

命令举例：oe.getPropertyWeight("name");

相关命令：setPropertyWeight

### 4、setPropertyWeight

命令含义：设置属性权重。

命令参数：两个参数，第一个参数输入属性名称，第二个参数输入属性权重。

命令产生：手动添加。

命令举例：oe.setPropertyWeight("name", 100);

相关命令：getPropertyWeight

## 5.56 openLowLog

命令含义：发放底层 log 功能，调用此函数后，在脚本回放时将保存各种回放信息。

命令参数：无。

命令产生：手动添加。

命令举例：ar.openLowLog();

相关命令：[closeLowLog](#)

## 5.57 pressKey

命令含义：向对象击键。

命令参数：五个参数，第一个参数输入对象名或对象的 ObjectElement，第二个参数输入按压的键，第三、四两个参数输入点击的位置（相对对象本身的坐标，有默认值 5、5），第五个参数输入点击方式（有默认值“left”）。

命令产生：手动添加。

命令举例：`ar.window("CabinetWClass_我的电脑").pressKey("Edit_地址", "Enter", 10, 10, "left");`

`ar.window("CabinetWClass_我的电脑").pressKey("Edit_地址", "Enter", 10, 10);`

`ar.window("CabinetWClass_我的电脑").pressKey("Edit_地址", "Enter");`

相关命令：[inputKey](#)

## 5.58 pressString

命令含义：向对象输入字符串。

命令参数：五个参数，第一个参数输入对象名或对象的 ObjectElement，第二个参数输入字符串，第三、四两个参数输入点击的位置（相对对象本身的坐标，有默认值 5、5），第五个参数输入点击方式（有默认值“left”）。

命令产生：手动添加。

命令举例：`ar.window("CabinetWClass_我的电脑").pressString("Edit_地址", "C:\\", 10, 10, "left");`

`ar.window("CabinetWClass_我的电脑").pressString("Edit_地址", "C:\\", 10, 10);`

`ar.window("CabinetWClass_我的电脑").pressString("Edit_地址", "C:\\");`

相关命令：[inputString](#)

## 5.59 putExcelWhole

命令含义：将内容保存到 Excel 表格内。

命令参数：两个参数，第一个参数输入二维数组，第二个参数输入 Excel 的全路径。

命令产生：手工添加或向导添加。

命令返回：操作成功返回 true，操作失败返回 false。

命令举例：String[][] a = {"第一行第一列", "第一行第二列"}, {"第二行第一列", "第二行第二列"};

```
ar.putExcelWhole(a  
"C:\\Users\\Administrator\\Desktop\\test.xls");
```

相关命令：[getExcelWhole](#)

## 5.60 putInto

命令含义：ParameterData 类中的成员函数，设置指定参数名的值。

命令参数：两个参数，第一个参数输入参数名称，第二个参数输入参数值。

命令产生：手工添加。

命令举例：参见下面例子。

相关命令：[getFrom](#)

命令说明：此命令用于跨脚本数据传递。

```

1  for(ParameterData pd : ar.getParameterDataList("E:\\Workspace\\q\\A.xls").subList(0, 5)
2  {
3      ar.window("SciCalc_计算器").clickControl("Button_C");
4      ar.window("SciCalc_计算器").clickControl("Button_" + pd.getFrom("btnA"));
5      ar.window("SciCalc_计算器").clickControl("Button_+");
6      ar.window("SciCalc_计算器").clickControl("Button_" + pd.getFrom("btnB"));
7      ar.window("SciCalc_计算器").clickControl("Button_=");
8      ar.window("SciCalc_计算器").checkProperty("Edit", "value", pd.getFrom("result"));
9      ar.parameterData = pd;
10     String str = ar.window("SciCalc_计算器").getProperty("Edit", "value");
11     pd.putInto("addResult", str);
12 }
13

```

脚本

输出 参数表

	名称	#0	#1	#2	#3	#4
0	btnA	1	2	5	4	9
1	btnB	7	9	6	8	8
2	result	8.	11.	11.	12.	17.

其中 pd 为变量名，可以更改：

## 5.61 putParameterDataList

命令含义：将数据列表写入 xls 表中。

命令参数：两个参数，第一个参数输入 LinkedList 类型的数据，第二个参数输入当前项目中的 xls 文件名或不在当前项目中的全路径名。

命令产生：手工添加。

命令返回：操作成功返回 true，否则返回 false

命令举例：

```
1 import java.util.Map;
2
3 LinkedList list = new LinkedList();
4 for (int i = 0 ; i < 10; i++) {
5     HashMap map = new HashMap();
6     ParameterData data = new ParameterData(map);
7     data.putInto("K1", "V1");
8     data.putInto("K2", "V2");
9     data.putInto("K3", "V3");
10    data.putInto("K4", "V4");
11    data.putInto("K5", "V5");
12    data.putInto("K6", "V6");
13    data.putInto("K7", "V7");
14    list.add(data);
15 }
16 ar.putParameterDataList(list, "C:\\Users\\Administrator\\Desktop\\Test.xls");
17
18
19
20
```

说明：V1,V2 等为需要输入的字符串

相关命令：[getParameterDataList](#)

## 5.62 regexTitle

命令含义：通过正则表达式匹配来激活窗口。

命令参数：一个参数，输入匹配窗口的正则表达式

命令产生：手工添加。

命令举例：`ar.regexTitle(".* - Microsoft Internet Explorer");`

相关命令：[activeTitle](#)

## 5.63 select

命令含义：鼠标选中了某个对象，一般这样的对象在被选中后状态会改变。

命令参数：五个参数，第一个参数输入对象名或对象的 ObjectElement，第

二个参数：对于列表视图对象、列表对象、组合框对象，输入的是选中的索引值；对于树对象，输入的是选中的节点全路径（层与层之间用\r分隔），第三、四两个参数输入点击的位置（相对对象本身的坐标，有默认值 5、5），第五个参数输入点击方式（有默认值"left"）。

命令产生：在列表视图对象、列表对象、组合框对象、树对象中选择索引值



时出现此命令。

```
命令举例： ar.window("CabinetWClass_doc").select("SysListView32",
"textTmp.doc", 57, 11, "left");
ar.window("CabinetWClass_doc").select("SysListView32",
"textTmp.doc", 57, 11);
ar.window("CabinetWClass_doc").select("SysListView32",
"textTmp.doc");
ar.window("CabinetWClass_我的电脑").select("SysTreeView32", "桌面
\r 我的电脑\rSystem(C:)", 29, 7, "left");
ar.window("CabinetWClass_我的电脑").select("SysTreeView32", "桌面
\r 我的电脑\rSystem(C:)", 29, 7);
ar.window("CabinetWClass_我的电脑").select("SysTreeView32", "桌面
\r 我的电脑\rSystem(C:)");
```

相关命令：[selectState](#)

## 5.64 selectState

命令含义：设置带复选框的树控件或列表控件的节点状态。

命令参数：六个参数，第一个参数输入带复选框的树控件或列表控件的对象名或窗口的 ObjectElement，第二个参数输入节点名或节点路径名，第三个参数输入节点状态，第四个参数输入相对于节点左上角的 x 方向偏移量（默认值 5），第五个参数输入相对于节点左上角的 y 方向偏移量（默认值 5），第六个参数输入鼠标点击方式（默认值为 left）

命令产生：手工添加。

命 令 举 例 :

```
ar.window("#32770_winCheckBox").selectState("SysListView32",
"item0", "true");
ar.window("#32770_winCheckBox").selectState("SysListView32",
"item0", "true", 7, 11);
ar.window("#32770_winCheckBox").selectState("SysListView32",
"item0", "true", 7, 11, "left");
```

```
ar.window("#32770_winCheckBox").selectState("SysTreeView32", "
item0\ritem1\ritem2", "true");
ar.window("#32770_winCheckBox").selectState("SysTreeView32", "
item0\ritem1\ritem2", "true", 7, 11);
ar.window("#32770_winCheckBox").selectState("SysTreeView32", "
item0\ritem1\ritem2", "true", 7, 11, "left");
```

相关命令：[setState](#)、[select](#)

## 5.65 setCaptureScreenWhenError

命令含义：设置脚本执行失败时是否截屏的状态标记。

命令参数：一个参数，输入布尔类型的值，如果为 true 则当脚本执行失败时会截屏，保存错误现场，如果为 false 则不截屏。

命令产生：手工添加。

命令举例：`ar.setCaptureScreenWhenError(true);`

相关命令：[getCaptureScreenWhenError](#)

## 5.66 setState

命令含义：设置对象选中与否的状态。

命令参数：五个参数，第一个参数输入对象名或对象的 ObjectElement，第二个参数输入状态（true 或 false），第三、四两个参数输入点击的位置（相对对象本身的坐标，有默认值 5、5），第五个参数输入点击方式（有默认值“left”）。

命令产生：在点击单选框对象时出现此命令。

命令举例：`ar.window("IEFrame_Google - Microsoft Internet Explorer").setState("INPUT_radio_ch", "true", 10, 10, "left");`

```
ar.window("IEFrame_Google - Microsoft Internet Explorer").setState("INPUT_radio_ch", "true", 10, 10);
```

```
ar.window("IEFrame_Google - Microsoft Internet Explorer").setState("INPUT_radio_ch", "true");
```

相关命令: [setValue](#)、[selectState](#)

## 5.67 setStopWhenError

命令含义: 设置如果运行出错是否立即停止。

命令参数: 一个参数, 布尔类型, true 表示出错停止, false 表示继续下面动作。

命令产生: 手工添加。

命令举例: `ar.setStopWhenError(true);`

相关命令: [getStopWhenError](#)

## 5.68 setSynchronizationTime

命令含义: 设置同步时间, 在命令运行一次失败时, 会等待一段时间 (setTimeSpan 所设置, 默认 500 毫秒) 后再次执行动作, 如果执行动作的总时间超过此命令设置的时间 (默认 3000 毫秒), 就会放弃命令的执行, 继续下一条命令。

命令参数: 一个参数, 输入同步时间 (单位毫秒)。

命令产生: 手工添加。

命令举例: `ar.setSynchronizationTime(3000);`

相关命令: [getSynchronizationTime](#)

## 5.69 setTimeSpan

命令含义: 设置两条运行命令之间的时间间隔。

命令参数: 一个参数, 输入间隔时间 (单位毫秒)。

命令产生: 手工添加。

命令举例: `ar.setTimeSpan(1000);`

相关命令: [getTimeSpan](#)

## 5.70 setValue

命令含义: 设置可输入文本的对象的值。

命令参数: 两个参数, 第一个参数输入对象名或对象的 ObjectElement,

第二个参数输入设置的文本。

命令产生：在向接受文本输入的对象中输入文本时出现此命令。

命令举例：`ar.window("IEFrame_Google - Microsoft Internet Explorer").setValue("INPUT_text_q", "http://www.baidu.com");`

相关命令：[setState](#)

## 5.71 setX

命令含义：设置鼠标的默认的相对水平偏移量。

命令参数：一个参数，输入设置的 X 坐标。

命令产生：手工添加。

命令举例：`ar.setX(10);`

相关命令：[getX](#)

## 5.72 setY

命令含义：设置鼠标的默认的相对垂直偏移量。

命令参数：一个参数，输入设置的 Y 坐标。

命令产生：手工添加。

命令举例：`ar.setY(10);`

相关命令：[getY](#)

## 5.73 sleep

命令含义：停顿命令。

命令参数：一个参数，输入停顿时间，单位是毫秒。

命令产生：在录制时选勾选上“记录时间间隔”时，各命令之间都会出现此命令, 如果没有勾选只能手工添加。

命令举例：`ar.sleep(3000);`

## 5.74 source

命令含义：导入外部脚本文件，一般在跨脚本函数调用的情况下用到。

命令参数：一个参数，输入外部脚本文件全路径。

命令产生：手工添加。

命令返回：无

命令举例：`source("C:\\test\\test.bsh");` // [代码示例](#)

相关命令：[addClassPath](#)

## 5.75 startApplication

命令含义：执行某个.exe 程序。

命令参数：两个参数，第一个参数输入程序路径，第二个参数输入附加参数（若无附加参数可设为空）。

命令产生：手工添加。

```
命令          举          例          :  
ar.startApplication("C:\\WINDOWS\\system32\\regsvr32.exe",  
"\"C:\\Program Files\\Spasvo\\AutoRunner\\SpasvoIe.dll\"");  
ar.startApplication("regedit", null);
```

## 5.76 stopRunning

命令含义：停止回放。

命令参数：无。

命令产生：手工添加。

命令举例：`ar.stopRunning();`

## 5.77 table

命令含义：将表格控件的指定行列滚动到可见区。

命令参数：三个参数，第一个参数输入表格对象名或表格的 ObjectElement，第二个参数输入单元格的行号，第三个参数输入单元格的列号。

命令产生：手工添加。

```
命令举例：ar.window("WindowsForms_Form1").table("Table", 0, 0);  
ar.window("WindowsForms_Form1").table("Table", 0,  
0).clickControl();
```

## 命令说明

1、table 命令之后还有一级命令，格式为  
ar.window(...).table(...).XXX(...);

2、此处 XXX 类似于 ar.window(...).xxx(...);XXX 比 xxx 要少第一个参数。

3、单元格的行号和列号都从 0 开始;

4、此命令后可以再接 clickControl、setValue、select、setState 等命令，分别对应普通单元格、可编辑单元格、带 ComboBox 控件的单元格和带 CheckBox 控件的单元格。

## 5.78 window

命令含义：将目标窗口激活并设为前台窗口。

命令参数：两个参数，第一个参数输入待激活的窗口对象名或窗口的 ObjectElement，第二个参数输入约束条件(选填，当不填时采用默认约束)。

命令产生：绝大部分脚本命令执行具体动作时都有这一句，录制时产生。

命令举例：  
ar.window("SciCalc\_计算器");  
ar.window("SciCalc\_计算器", "strict");

相关命令：[menu](#)

**注：约束条件有以下四种：**

default：默认的约束条件，此条件忽略窗口的大小。

strict：严格的约束条件，此条件会匹配对象库中权重 100 的所有对象属性。

ignoreName：忽略窗口名字约束条件，此条件忽略窗口名字。

ignoreClass：忽略窗口类名约束条件，此条件忽略窗口类名。

## 5.79 windowExist

命令含义：判断窗口是否存在。

命令参数：一个参数，输入对象库中存在的待检查窗口的对象名或窗口的 ObjectElement

命令产生：手工添加。

命令返回：存在返回 true，否则返回 false

命令举例：ar.windowExist("SciCalc\_计算器");

相关命令：[menuExist](#)、[controlExist](#)

## 6 常见问题

### 6.1 Flex 程序录制不出脚本

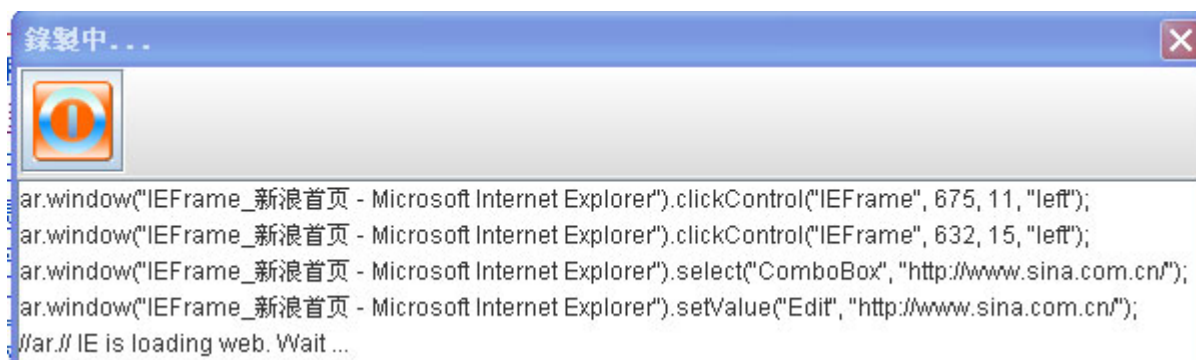
- A、请确认您申请的 lic 文件时是支持 Flex 程序的录制，如果不支持的话，是录制不了的。
- B、请确认录制的控件是常用的 Flex 控件，对于数据表格、日期控件等不常用控件尚不支持自动化测试。
- C、请确认待测试的元素是控件而不是图片，很多看上去是标准控件的元素可能其实是图片，对于这样的非常规界面 AutoRunner 不能录制出脚本。通过手工添加对象，查看对象的 winClass 属性是否为 Graphic，可以得知对象是否为图形。

### 6.2 IE 对象回放不通过

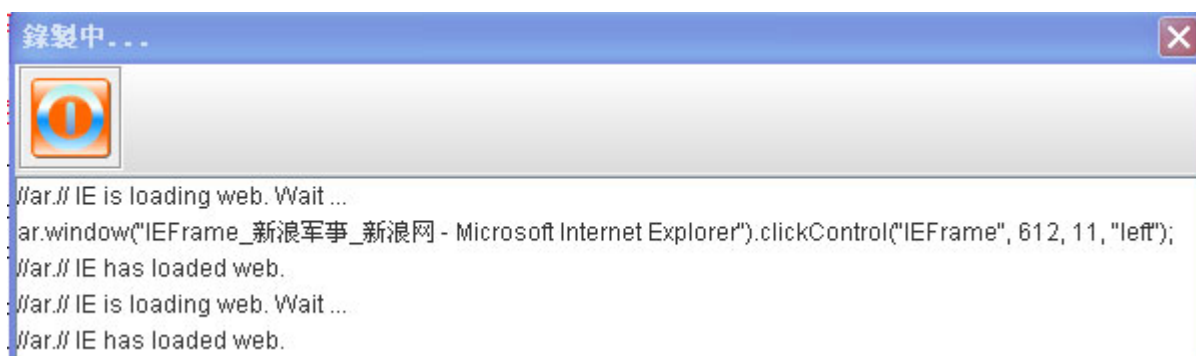
- A、对于 Win7、Vista、Windows2008 等操作系统，如果有打开浏览器的操作应该调用 startApplication 命令而不应该用双击桌面浏览器图标的方式打开，以便能获取最高权限。
- B、在回放时请确定网页已恢复到录制时状态，并且已经加载完成；
- C、在执行动作后有新网页打开时，可以在后加一个 sleep 命令设置一个等待时间，以便等待网页被完全加载。在回放时，可以适当删减一些不必要的命令等。

### 6.3 IE 脚本录制时某次操作没有被记录

在录制网页脚本时，如果发现某一次点击动作没有被记录通常是由于此网页还没有被加载完整，有如下图记录面板中最后一行提示，在网页没加载完之前所有在网页中的操作都不会被记录。



当网页加载完成时，有如下图记录面板中最后一行提示。



因此，在录制网页脚本时应等待页面加载完整后才开始录制，如果网页有很多帧，在录制面板中就会多次提示页面加载完整，此时还应查看网页状态栏中最左边是否有加载完成的提示，只有同时显示加载完整后才可录制。

## 6.4 IE 录制不出脚本

A、目前软件只支持 IE 浏览器的录制，如果你使用的浏览器不是 IE，网页录制不出脚本。

B、请确认在软件安装时，杀毒软件弹出的插件拦截消息被放行，如果选择禁止的话，网页录制不出脚本。

C、请确认您申请的 lic 文件时是否选中了支持 IE 的录制，如果没有选的话，网页录制不出脚本。

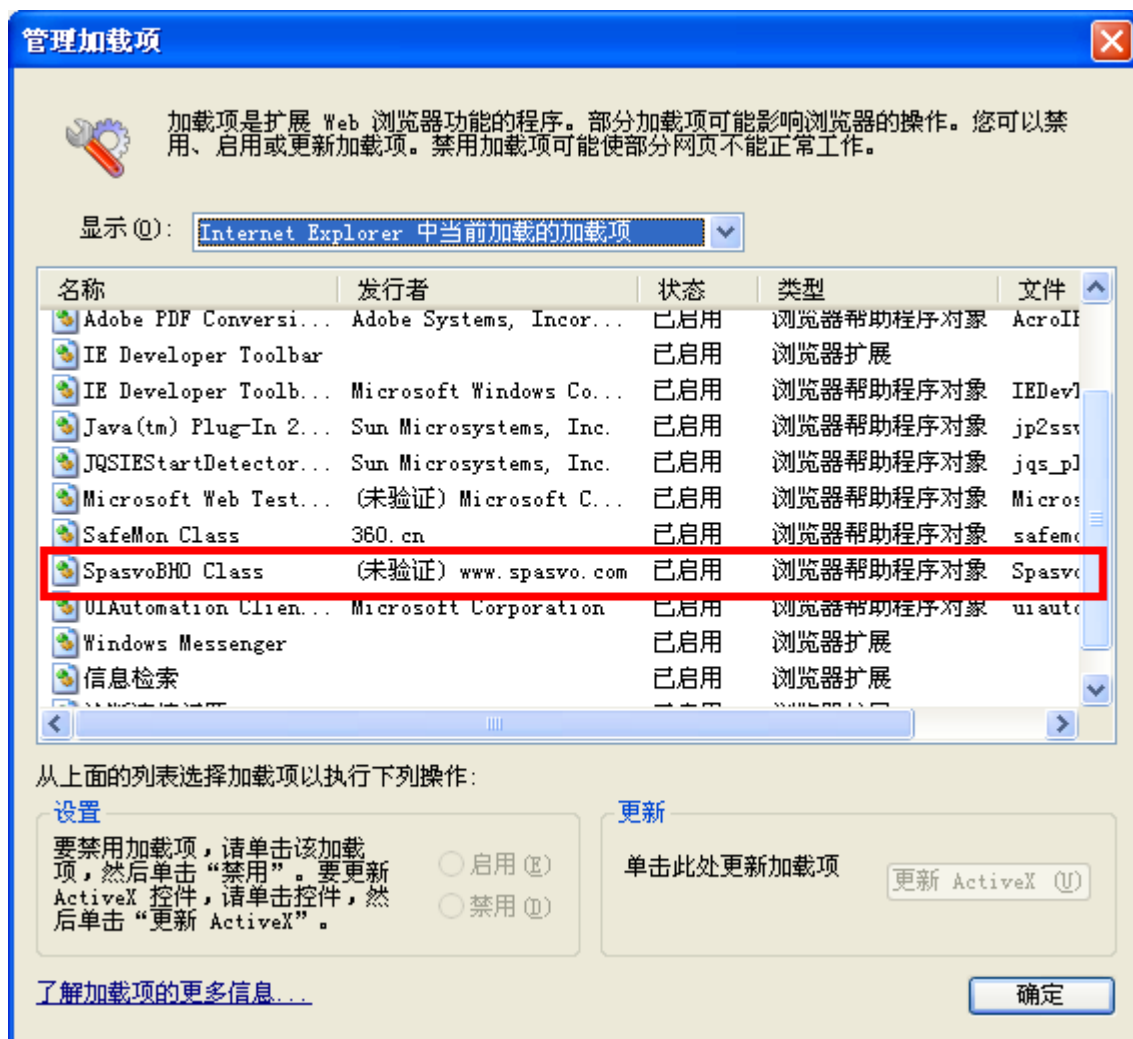
D、随意打开一个网页，点击【工具】-->【管理加载项】，尝试找到下图中的项，如果没有找到那么请点击【开始】菜单，选择【运行】，然后输入

```
regsvr32 "C:\Program Files\Spasvo\AutoRunner\SpasvoIe.dll"
```

其中双引号中的动态库路径请以电脑中的实际安装路径为准，在这期间



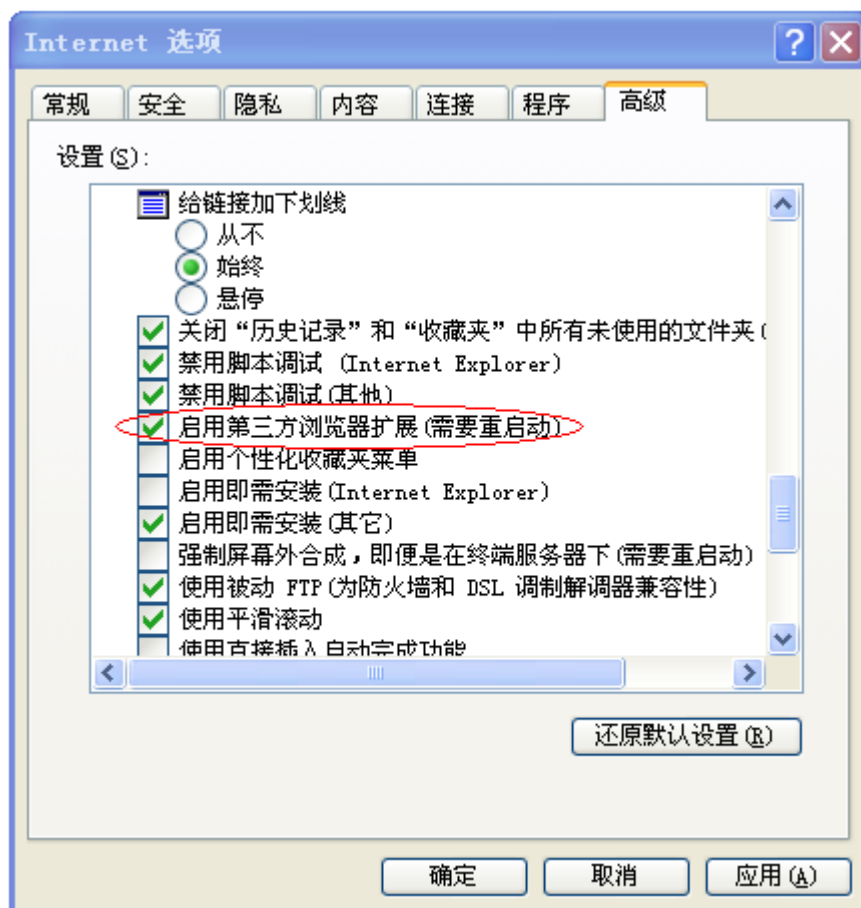
如遇杀毒软件的拦截提示请放行，当弹出注册成功的消息提示时，请重开浏览器再次尝试录制网页



如果找到上图中的项，请查看它的状态是否处于“已启用”状态，如果没有，请启用它，之后请重开浏览器再次尝试录制网页。

E、如果操作系统是 Win7 、Vista 或者 Windows2008，则浏览器应该以右键管理员身份运行。

F、如果是 Windows2003 或是 Windows2008 服务器操作系统，由于安全级别比较高，在录制不出脚本时请打开浏览器【工具】菜单，选择【Internet 选项】，如下图所示。将图中带圈的选项勾选，再重启电脑即可录制网页脚本。



## 6.5 setValue 命令无效

setValue 命令直接调用对象的内置函数将值发送到控件，比如 Edit 控件中设置文本、滑块控件中设置滑块的位置，在对这些控件做设值操作时正常情况下有可能要触发一系列的后续动作。由于控件设计特殊等原因，它只响应键盘输入的设置操作，而对通过内置函数的设值操作不触发后续动作，因而导致接下来的脚本回放出现组件没有发现的错误。对于这种情况，解决的方法是用 pressString 命令替换 setValue 命令。pressString 对每一个字符的输入都是模拟键盘输入完成的，这样后续动作就可以正常触发，脚本正常回放。

## 6.6 Silverlight 程序录制不出脚本

A、请确认您申请的 lic 文件时是支持 Silverlight 程序的录制，如果不支持的话，是录制不了的。

B、请确认录制的控件是常用的 Silverlight 控件，对于数据表格、日期

控件等不常用控件或是复杂的第三方控件商提供的控件尚不支持自动化测试。

C、请确认待测试的元素是控件而不是图片，很多看上去是标准控件的元素可能其实是图片，对于这样的非常规界面 AutoRunner 不能录制出脚本。通过手工添加对象，查看对象的 winClass 属性是否为 Image，可以得知对象是否为图形。

## 6.7 Vista、Win7、Win2008 中注意事项

如果软件运行在 Vista、Win7、Win2008 等操作系统上，

录制：在录制之前，应用程序应以右键管理员身份运行；

回放：在回放之前，在脚本中添加 startApplication 命令来启动应用程序；

## 6.8 安装出错

此问题多发生在 Win7 操作系统上。由于软件安装时要进行各种读写操作、电脑硬件信息获取、插件的注册等，每一项操作都需要程序有最高权限，而在 Win7 下默认是没有最高权限的，所以安装往往会出现问题。解决方法是：右击安装包，在弹出菜单中选择以【管理员权限运行】即可。

## 6.9 表格控件的录制与回放

- 1、目前支持的标准表格控件类型为 WindowsForms 表格，Java 表格，QT 表格，Silverlight 表格，WPF 表格等，不支持 Flex 表格及自定义表格的录制与回放；
- 2、表格的录制采用手工添加对象的方式（鼠标移动到标题头上再同时按下 Alt+Ctrl）将其添入对象库中，不支持自动录制操作。对表格的自动化测试脚本需手工编写，相关的命令主要为 table，具体用法请参考“脚本命令”中的 table 一节。
- 3、对于表格控件的单元格操作只支持 clickControl、select、setValue、setState 等操作。clickControl 命令是点击单元格；select 是选择带 ComboBox 的单元格中 ComboBox 中的某个索引；setValue 是设置可编辑单元格中的文本；setState 是设置带 CheckBox 的单元格中

CheckBox 控件的状态。

## 6.10 不能识别对象

在录制脚本时遇到这个问题，可以通过手工添加对象的方式添加没有识别的对象，在默认情况下，对于静态控件，无效控件，图片控件，自绘控件，不可编辑的文本控件，一般滚动条控件等对象在录制时都不会被记录下来，如果要对这些控件做操作，可以手工添加进对象库。

## 6.11 回放不停止或回放时间过长

回放不能停止，很大一部分是发生在脚本的串联调用过程中，由于串联调用的脚本形成了一个调用环（A 脚本调用 B 脚本，而在 B 脚本中又调用了 A 脚本），导致回放进入死循环，解决的办法可以通过热键强行终止执行（Alt+Ctrl+Shift+S）。对于回放时间过长，有可能是执行的脚本过多导致，也有可能是执行某一句脚本失败，而软件进行了多次尝试，解决的办法是，添加 `ar.setStopWhenError(true)` 命令，使脚本执行失败时立刻停止，还可以调用 `setSynchronizationTime` 命令将时间设短一点。

## 6.12 回放对象不在对象库中

所有的回放对象在对象库中都要存在才能回放成功，如果没有则会报这个错误，此时可以手工添加对象到对象库，方法是：打开对象库；点击左下角的“增加对象”按钮；将鼠标移动到待添加的对象上方；同时按下键盘上的 Ctrl 和 Alt 键并维持 1 秒左右，在右下角的信息提示框中，会显示录制下来的对象，且对象已自动添加到了对象库中。

## 6.13 回放时提示找不到对象

A、回放时，应该保证被测程序当前状态和录制时的最初状态一致，如果不一致，则可能出现对象未找到的错误。

B、在一台电脑上录制脚本，在另一台电脑上回放。这种情况下，应该保证两天电脑的操作系统类型一样，电脑屏幕分辨率一致，如果不一致，则

可能出现对象未找到的错误。

C、如果是在一台电脑上出现此问题，解决的方法是：新建一个脚本，手工添加刚回放不了的对象，将此时的属性与先前的属性做一个比较，着重比较权重为 100 的属性，查看哪些属性不一致。由于当前的版本在回放时采用智能查找对象的方式，会有针对性的将某些权重置为零，之后再次进行查找，但对象的 position 属性不会被置为零，所以如果这个属性不同的话，可以手工调整 position 属性的值，则可以回放通过。

## 6.14 键盘键码表

此键码表用于 inputKey 、pressKey 的参数。

### 1、特殊键

退出键：“Esc”

制表键：“Tab”

大写锁定键：“Caps Lock”

左上档键：“Shift”

右上档键：“Right Shift”

控制键：“Ctrl”

换挡键：“Alt”

左 Windows 键：“Left Windows”

右 Windows 键：“Right Windows”

Application 键：“Application”

回格键：“Backspace”

回车键：“Enter”

抓屏、系统请求键：“Sys Req”

滚动锁定键：“Scroll Lock”

插入键：“Insert”

删除键：“Delete”

起始键：“Home”

结束键：“End”

上页键：“Page Up”

下页键：“Page Down”

上箭头键：“Up”

下箭头键：“Down”

左箭头键：“Left”

右箭头键：“Right”

数码锁定键：“Num Lock”

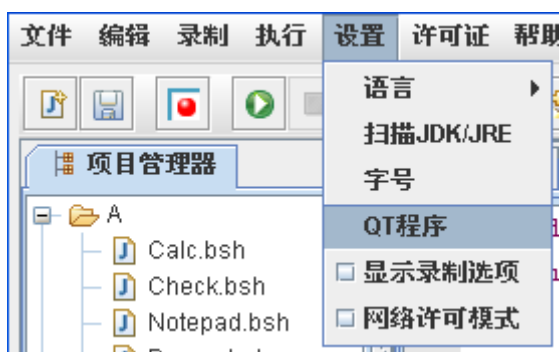
## 2、F 功能键

F1 到 F12 键：“F1” 到 “F12”

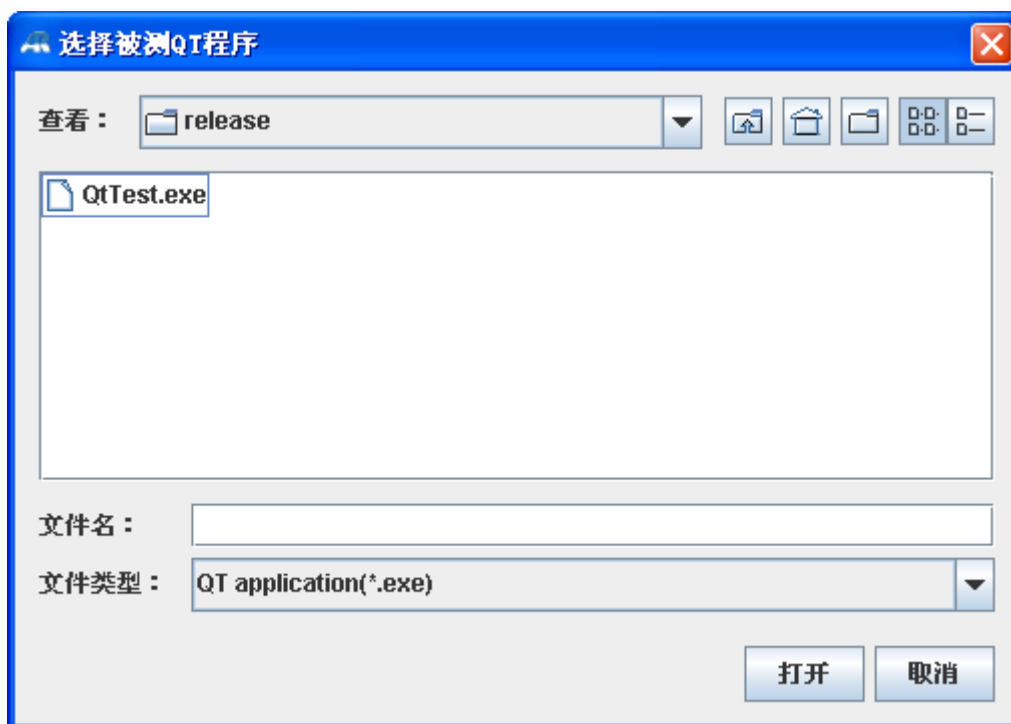
## 6.15 录制 Qt 程序

在录制 Qt 程序之前请确认您的 Lic 支持 Qt 对象的录制，之后再按如下步骤进行：

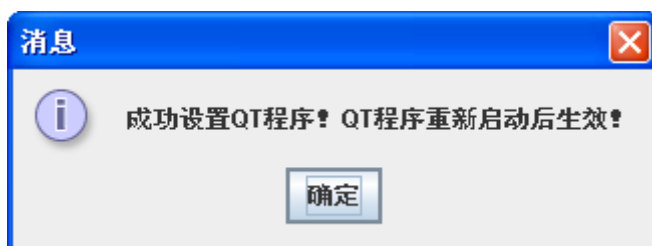
1、在【设置】菜单中选择【QT 程序】菜单项，如下图所示；



2、在弹出的对话框中选择一个待录制的 Qt 程序，如下图所示；



3、打开应用程序后会弹出如下配置完成的提示框，再次打开此程序时就可以录制 Qt 程序脚本了。



4、正常录制 Qt 应用程序时脚本如下图所示。

```

1 ar.window("QWidget_MainWindow").setState("RadioButton_Radio1", "true");
2 ar.window("QWidget_MainWindow").setState("CheckBox_CheckBox", "false");
3 ar.window("QWidget_MainWindow").clickControl("PushButton_PushButton", 33, 13, "left");
4 ar.window("QWidget_MainWindow").setValue("Slider", "10");
5 ar.window("QWidget_MainWindow").select("ComboBox", "item5");
6 ar.window("QWidget_MainWindow").setValue("Text", "123");
7 ar.window("QWidget_MainWindow").select("Tree", "item3\ritem3.3", 25, 6, "left");
8 ar.window("QWidget_MainWindow").select("List", "item3", 20, 6, "left");

```

## 6.16 密码框获取不到值

对于密码形式的编辑控件，由于密码受到保护 setValue 往往没有值，这

是正常现象，在回放时可以手工输入密码。在某些情况，密码控件对象在录制时可能也没有记录下来，此时可以通过手工添加组件的方式将其加入对象库。

## 6.17 内嵌网页不能录制脚本

对于一些内嵌网页的桌面程序，由于当前版本尚不支持内嵌网页的脚本录制，所以在网页录制时会出现不能自动识别对象和录制脚本的现象。如果要对这类网页进行自动化测试，可以先通过手工添加对象的方式 (Alt+Ctrl) 把待测对象加入对象库，之后手工编写测试脚本进行测试，对于大多数的网页链接和按钮使用的脚本命令通常是 `clickControl`，对于可编辑的文本控件使用的脚本命令通常是 `setValue` 等，其他脚本的添加请详细阅读相关脚本命令。

## 6.18 校验矩形区域文本命令结果有误

A、对于微软自带的程序，比如计算器、记事本等，微软做了特殊处理，`checkRectText` 函数 获取不到指定区域的值，建议用非微软的程序做测试，比如 QTP 航班订票例子。

B、确认所要校验的矩形区域是否有文本存在，矩形区域的左上角坐标和右下角坐标都是相对于对象左上角坐标的偏移。

## 6.19 循环参数表未执行

- A、检查参数表数据设计的是否正确；
- B、对参数表进行的任何修改是否都已保存；
- C、循环参数表的循环体脚本是否编写得当；
- D、检查参数化后，回放时的实际对象在对象库中是否存在。
- E、循环参数表是很高级的脚本流程设计，在编写之前请认真查看“高手进阶”的相关内容和例子。

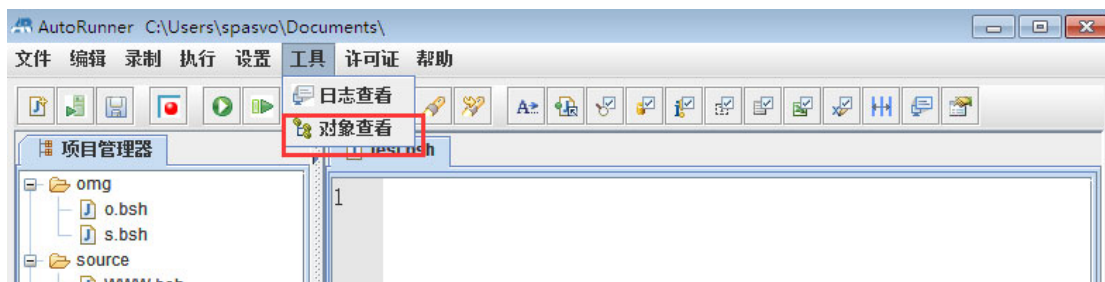
## 6.20 AR 录好脚本之后点击执行没有反应。

解答：AR 没能正确安装，建议卸载之后重装

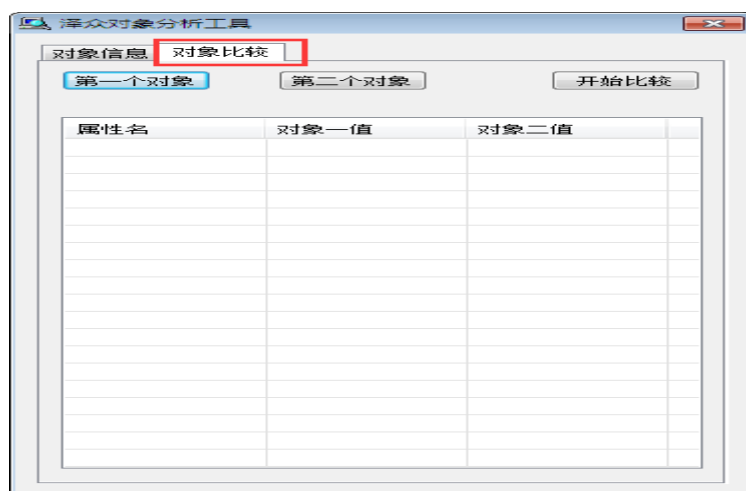


## 6.21 AR 如何进行两个对象内属性数据的比较。

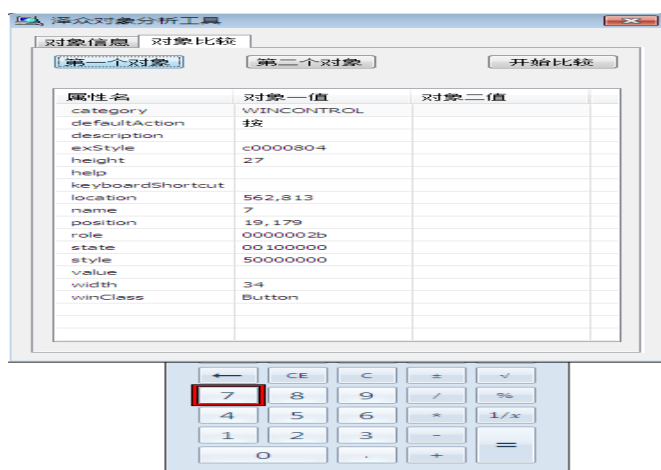
解答：第一步：打开 AR 的工具选项，选择对象查看；



第二步：点击对象比较：



第三步：点击第一个对象，鼠标左键按住不放拖拽到想要比较的对象：



第四步：点击第二个对象，操作和上一步一样；

最后：点击开始比较就好

## 6.22 计算机如何录制 10 以上的数据如何参数化。

解答：详细过程如图所示：

```

1  for (ParameterData pd : ar.getParameterDataList("test.xls")/*.subList(0, 2)*/)
2  {
3      //ar.parameterData = pd;//ar.parameterData可用于脚本之间传递参数
4  ar.window("CalcFrame_计算器").clickControl("Button_"+pd.getFrom("x"), 15, 11, "left");
5  ar.window("CalcFrame_计算器").clickControl("Button_"+pd.getFrom("x1"), 21, 13, "left");
6  ar.window("CalcFrame_计算器").clickControl("Button_"+pd.getFrom("q"), 20, 13, "left");
7  ar.window("CalcFrame_计算器").clickControl("Button_"+pd.getFrom("y"), 19, 17, "left");
8  ar.window("CalcFrame_计算器").clickControl("Button_"+pd.getFrom("y1"), 18, 10, "left");
9  ar.window("CalcFrame_计算器").clickControl("Button_等于", 19, 37, "left");
10 ar.window("CalcFrame_计算器").checkProperty("Static_结果", "value", pd.getFrom("z"));
11
12 ar.window("CalcFrame_计算器").clickControl("Button_清除", 21, 11, "left");
13
14 }
15
16

```

	名称	#0	#1
0	x	0	1
1	x1	3	3
2	q	加	加
3	y	1	1
4	y1	2	5
5	z	15	28

校验"Static\_结果"的"value"属性:  
期望值为: 15  
实际值为: 15  
匹配结果: true  
校验"Static\_结果"的"value"属性:  
期望值为: 28  
实际值为: 28  
匹配结果: true

2014-10-28 10:42:32  
test.bsh脚本:  
执行结束,耗时15秒。  
执行结果:执行成功!

## 6.23 AR 针对 excel 表格下拉列表的抓取失败的原因。

解答：是 windows 标准控件不能录制的情况，需要与我们销售人员联系，根

据软件来定制开发。

## 6.24 AR 安装失败。

解答：1、JDK 未能正确安装，可以打开 cmd 输入 java -version，javac 查看 JDK 是否正确安装，如果提示不是内部命令，要进行环境变量的配置；2、一定要使用超级管理员账户来操作。

下面是正确的显示：

```
C:\Users\Administrator>java -version
java version "1.6.0_45"
Java(TM) SE Runtime Environment (build 1.6.0_45-b06)
Java HotSpot(TM) Client UM (build 20.45-b01, mixed mode, sharing)

C:\Users\Administrator>
```

```
C:\Users\Administrator>javac
用法: javac <选项> <源文件>
其中, 可能的选项包括:
-g 生成所有调试信息
-g:none 不生成任何调试信息
-g:<lines,vars,source> 只生成某些调试信息
-nowarn 不生成任何警告
-verbose 输出有关编译器正在执行的操作的消息
-deprecation 输出使用已过时的 API 的源位置
-classpath <路径> 指定查找用户类文件和注释处理程序的位置
-cp <路径> 指定查找用户类文件和注释处理程序的位置
-sourcepath <路径> 指定查找输入源文件的位置
-bootclasspath <路径> 覆盖引导类文件的位置
-extdirs <目录> 覆盖安装的扩展目录的位置
-endorseddirs <目录> 覆盖签名的标准路径的位置
-proc:<none,only> 控制是否执行注释处理和/或编译。
-processor <class1>[,<class2>,<class3>...] 要运行的注释处理程序的名称; 绕过默认
的搜索进程
-processorpath <路径> 指定查找注释处理程序的位置
-d <目录> 指定存放生成的类文件的位置
-s <目录> 指定存放生成的源文件的位置
-implicit:<none,class> 指定是否为隐式引用文件生成类文件
-encoding <编码> 指定源文件使用的字符编码
-source <版本> 提供与指定版本的源兼容性
-target <版本> 生成特定 UM 版本的类文件
-version 版本信息
-help 输出标准选项的提要
-Akey[=value] 传递给注释处理程序的选项
-X 输出非标准选项的提要
-J<标志> 直接将 <标志> 传递给运行时系统

C:\Users\Administrator>
```

## 6.25 AR 使用时报-24 的错误

解答：AR 连不上服务器，首先看一下能不能联网，再查看服务器的 licence

server 有没有启动，如果已经启动了的话，查看一下服务器的 licence 有没有过期，然后再查看 AR 设置里面的网络许可模式有没有正确配置。

## 6.26 AR 使用时报出-13 的错误。

解答：使用的 AR 账号过期，和工作人员联系延长使用时间。

## 6.27 AR 使用时报出-30 的错误。

解答：使用的是网络版的 AR，账号过期，和工作人员联系延长使用时间。

## 6.28 AR 的 lic 报出-1 的错。

解答：因为自己的 lic 是从别的位置申请的，为无效文件，lic 文件必须是自己电脑导出的。

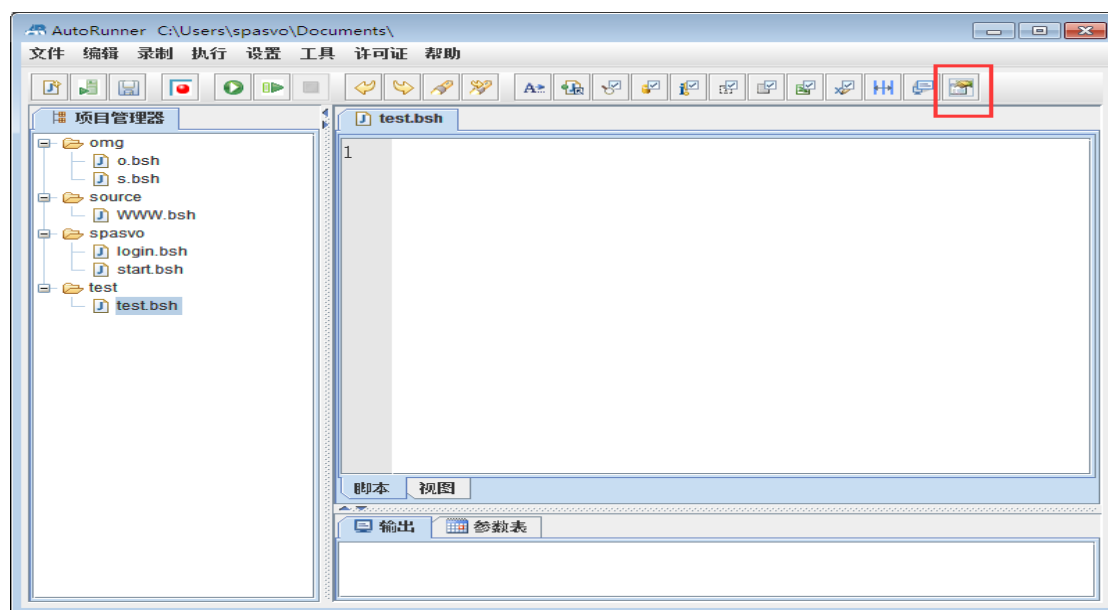
## 6.29 AR 无法正常安装，报出未关闭某些进程。

解答：到进程里面关掉某些 Java 进程就好了。

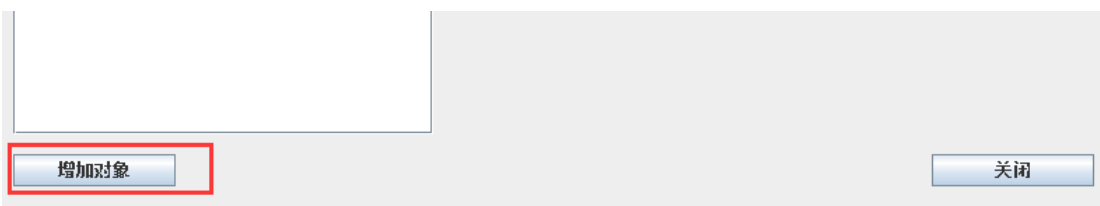
## 6.30 使用 AR 时抓取不到对象，提示找不到对象。

解答：先确认对象是否可以抓取，然后到对象库里面可以使用强制抓取获得对象。

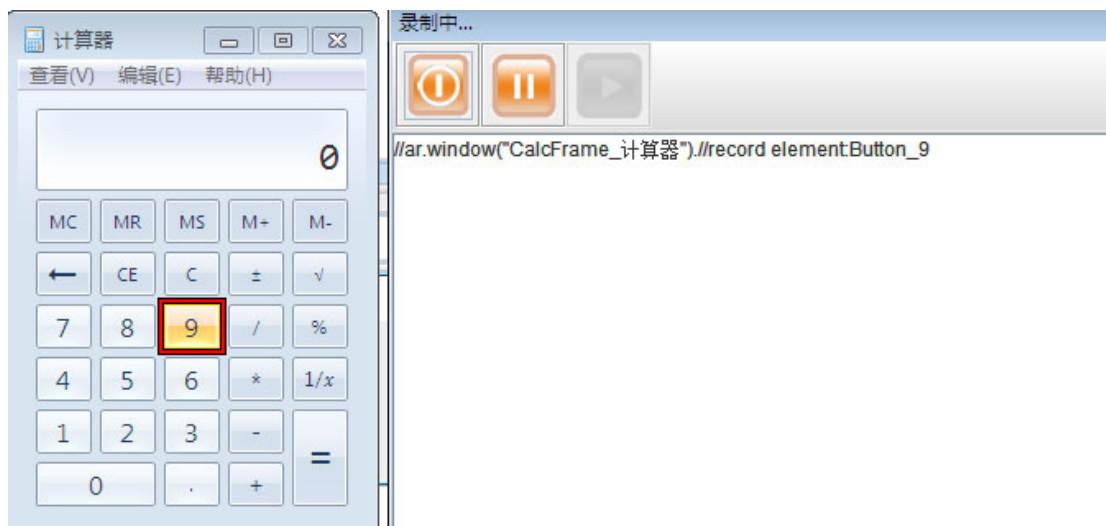
第一步：点击 AR 的对象库



第二步：点击增加对象



第三步：按住 Ctrl+Alt 强制抓取对象：



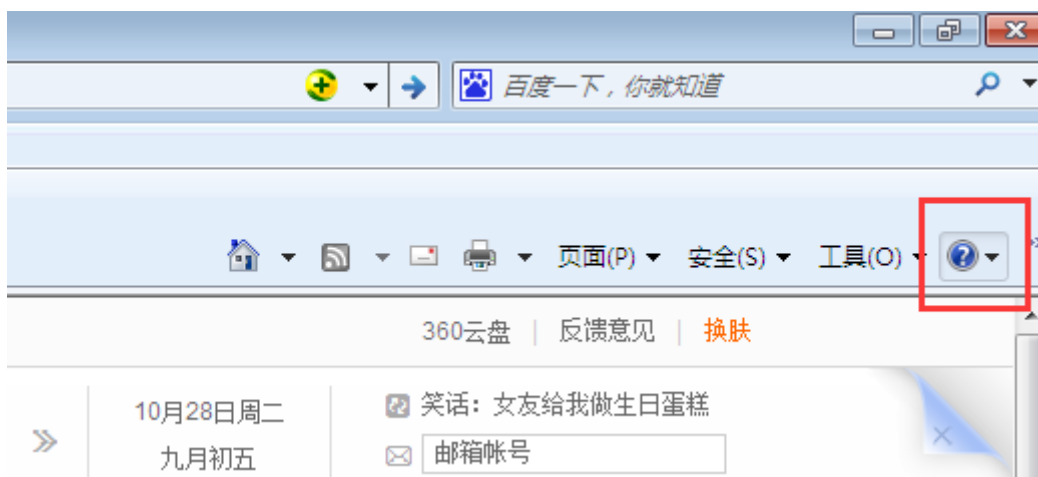
### 6.31 AR 的 LICENCE 报-7 错误。

解答：licence 产品信息不匹配,可能是 licence 是别的工具申请的 licence 导入了 AR

### 6.32 AR 录制出现空白。

解答：可能是 IE 浏览器版本的问题,建议使用 IE6~IE10 的版本。如何查看 IE 浏览器版本:

第一步：打开 IE 浏览器点击帮助图标;



第二步：点击关于 Internet explorer



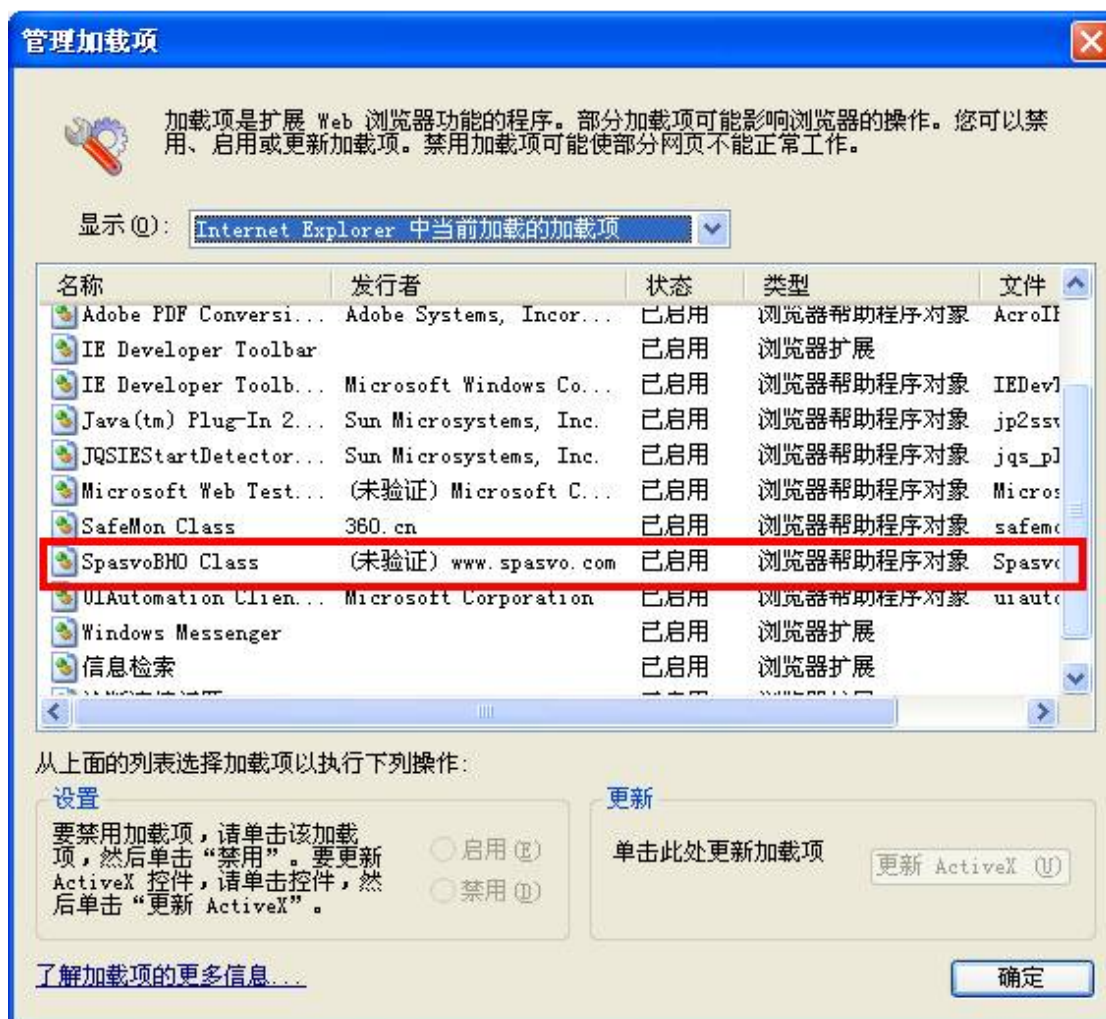
或者由于以下原因：

A、请确认在软件安装时，杀毒软件弹出的插件拦截消息被放行，如果选择禁止的话，网页录制不出脚本。

B、请确认您申请的 lic 文件时是否选中了支持 IE 的录制，如果没有选的话，网页录制不出脚本。

C、随意打开一个网页，点击【工具】-->【管理加载项】，尝试找到下图中的项，如果没有找到那么请点击【开始】菜单，选择【运行】，然后输入  
`regsvr32"C:\Program Files\Spasvo\AutoRunner\SpasvoIe.dll"`

其中双引号中的动态库路径请以电脑中的实际安装路径为准，在这期间如遇杀毒软件的拦截提示请放行，当弹出注册成功的消息提示时，请重开浏览器再次尝试录制网页

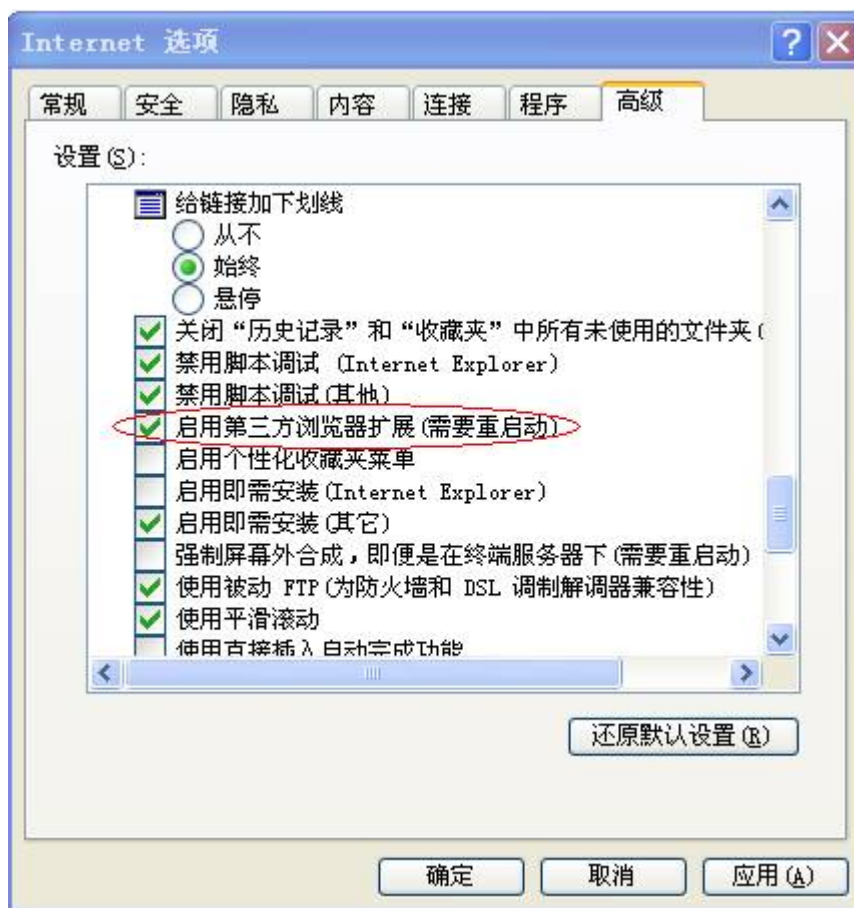


如果找到上图中的项, 请查看它的状态是否处于“已启用”状态, 如果没有, 请启用它, 之后请重开浏览器再次尝试录制网页。

D、如果操作系统是 Win7 、 Vista 或者 Windows2008, 则浏览器应该以右键管理员身份运行。

E、如果是 Windows2003 或是 Windows2008 服务器操作系统, 由于安全级别比较高, 在录制不出脚本时请打开浏览器【工具】菜单, 选择【Internet 选项】, 如下图所示。将图中带圈的选项勾选, 再重启电脑即可录制网页脚本。





### 6.33 使用云版的 AR，报出-26 的错误。

解答：首先确认这个错误是链接不到服务器，然后自己申请确认排除服务器的原因，之后就是网络和账号过期的原因。

### 6.34 AR 在使用过程中报出-22 的错误。

解答：可能是网络不稳定造成的错误，建议检查一下自己的网络环境。

### 6.35 AR4.0 录制谷歌浏览器。

解答：AR4.1 版本录制谷歌浏览器，需要在谷歌浏览器的扩展程序中添加 AR 安装目录下的 browser 文件夹下的 chrome\_ext.crx 文件。另外录制谷歌浏览器录取不到输入操作，即 setValue 函数。如果想回放输入操作需要在录制完相应对象后手动输入 setValue 的函数。



## 6.36 AR 中 try+catch 的用法。

解答：try 就像一个网，把 try {} 里面的代码所抛出的异常都网住，然后把异常交给 catch {} 里面的代码去处理。最后执行 finally 之中的代码。无论 try 中代码有没有异常，也无论 catch 是否将异常捕获到，finally 中的代码都一定会被执行。

虽然 Java 执行时期系统所提供的预设处理器对除错很有用，你通常想要自己处理例外。这样做有两个优点：第一，它让你修正错误。第二，它可以避免程式自动终止。每当错误发生时，如果你的程式就停止而且列印出堆叠追踪，大多数的使用者都会感到很困惑。很幸运，你很容易就能避免这种情形。

要防备并且处理执行时期错误，只要将你要监视的程式码放在 try 区块里即可在 try 区块之后紧接著在 catch 子句里指定你希望捕捉的例外型态  
错误捕捉例子：

```
try
{
    code; //将自己的代码放在其中;
} catch(e) //如果上面的代码有错误，这里就捕获
{
    alert(e.number); //获得错误信息
}
```

例如：

```
Import java.io.*; //调用 io 包
public class SimpleCharInOut
{
    public static void main(String args[])
    {
        char ch=' '; //定义个字符 ch 初始为 ' '
        System.out.println("Enter a character please"); //在屏幕上输出 Enter
        a character please
        try{//你要监视的程式码放在 try 区块里即可在 try 区块之后紧接著在 catch
```

子句里指定你希望捕捉的例外型态

```
ch=(char)System.in.read();//将从键盘输入的字符赋给 ch
}
catch(IOException e)//如果上面的代码有错误，这里就捕获
{ };//错误后不进行操作
System.out.println("You're entered character: "+ch);//在屏幕上输出
You're entered character:
//和 ch 的值
}
}
```

在例如 try{

```
int i = 1/0;
}catch(Exception e){
}
```

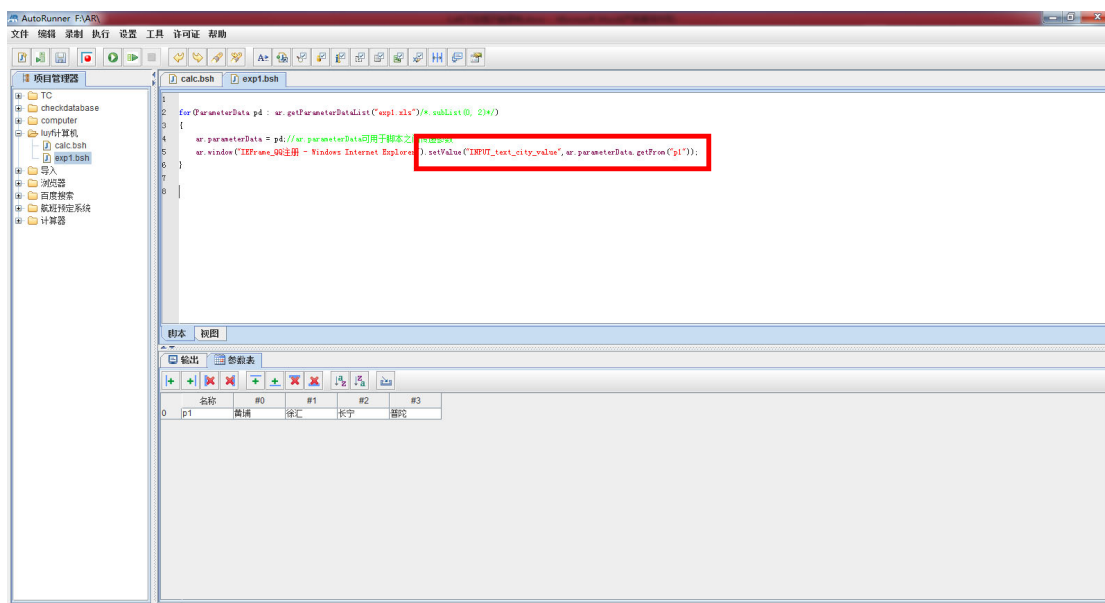
一个计算的话，如果除数为 0，则会报错，如果没有 try 的话，程序直接崩溃用 try 的话，则可以让程序运行下去，并且输出为什么出错！

用 try 的话，配合 log4j 使用会对程序的日后维护帮助很大。

### 6.37 AR 无法录制下拉框。

解答：使用 setvalue 或者使用 select 函数解决

如图



### 6.38 AR 无法录制密码。

解答：setValue 用 pressString 代替就可以了

如果回放的时候无法回放密码时，主要看被测系统，一般系统如果加密的话现在这里无实现的。

### 6.39 AR 修改参数化。

解答：只能通过程序提供的界面（过于紧凑，不便输入）添加参数，而不能直接修改保存参数的外部 EXCEL 文件，操作不方便

脚本文件总共有 3 个文件：

.bsh（脚本源文件）、.xml（对象库文件）、.xls（数据参数表文件）。

其中录制完毕即有.bsh 文件和.xml 文件。

方法 1：在参数表中添加参数数据保存后即可生成.xls 文件。

方法 2：可以直接在外部编辑.xls 表（文件名与脚本名相同），再在软件中导入。

### 6.40 AR 支持什么浏览器。

解答：支持 IE6、7、8，AR4.1 支持谷歌浏览器。

### 6.41 自动记录脚本文件后，尚需要较多的手动编辑来完成。

解答：自动化测试录制不是目的。目的在于回放。为了回放的精准、正确，有些

时候确实需要对脚本进行增、删、改，自动化测试工具要求测试人员有一定的编程基础及排错能力，特别是对于高级的测试脚本，都是要求测试人员手工编写，工具是不能替代的。当然，我们现在的版本需要修改的已经不是很多了，除非是额外加的功能。

## 6.42 AR 找不到对象。

解答：修改权重为 0

## 7 服务支持

AutoRunner 产品通过在线 MSN、电话、电子邮件为您提供支持与服务，也可访问我们的网站 <http://www.spasvo.com/>；为保证服务质量，确保有效地解决用户的问题，保障用户的项目实施进度，技术支持仅向授权用户和授权试用用户提供。请您在联系泽众技术支持时，告知您的单位名称和服务代码。

技术支持：

MSN(技术支持): [spasvo\\_support@hotmail.com](mailto:spasvo_support@hotmail.com)

电话： 021-31357887

传真： 021-31357887-8017

产品服务：

有关培训、产品购买及试用授权方法的问题，请与销售代表联系，或联系泽众咨询热线：

电话： 021-31357887

传真： 021-31357887-8017

电子邮件: [sales@spasvo.com](mailto:sales@spasvo.com)