TestPlant Documentation

English 简体中文 日本語

重复使用代码

无法避开这个问题，因为在测试中重复使用是无法改变的事实。幸运的是，有很多简单的方法，可以让脚本和处理程序执行重复的任务。

脚本

对于在多个脚本中使用的代码，创建一个可按需要调用的脚本。如果在同一个套件或帮助程序套件中，使用简单的名称（没有空格或特殊字符），可将脚本名称用在Run命令中，后面加所需的参数。

对于毫不相关的套件中的脚本，或者脚本名称中有空格或特殊字符，可采用Run命令后面加脚本路径名。

对于测试套件中未包含的脚本，使用脚本路径名调用Run命令。

示例：是一个脚本内调用脚本

```
CycleWindows                                                    //调用简单脚本名称运行命令。

ConnectionScript "Old SUT", "New SUT"                           //调用简单脚本名称加参数。

Run "/Users/Seiji/Documents/EggPlant Suites/Validation.suite/Scripts
/ConnectionScript"                                              //调用Run命令从一个毫不相关的套件中打开脚本。
```

处理程序

也可编写有点类似于子脚本的"处理程序"，在同一个脚本中重复使用代码。处理程序是主脚本的一部分，随时都可调用，就像调用其它脚本一样。

示例：CaptureTheScreen处理程序

```
to CaptureTheScreen prefix, count                               //启动该处理程序。

put prefix& "ScreenShot" & count into fileName                  //将prefixScreenShotCount添加到fileName。

put "/tmp/" & fileName into tempFile                            //将"/tmp/filename"添加到tempFile。

CaptureScreen tempFile                                          //捕获屏幕截图。

return fileName                                                 //将fileName设置为函数返回值。

end CaptureTheScreen                                            //结束该处理程序。
```

处理程序运行结果

如果脚本或处理程序像上述例子一样，返回一个值，则可调用下一行中的Results函数，访问该值。因此，使用上述例子，你可以调用处理程序，并访问结果，如下所示。

示例：返回处理程序运行结果

```
CaptureTheScreen "MacTest", 6                                   //调用CaptureTheScreen处理程序。

put the result into newFile                                     //将返回的值添加到变量中。
```

类似于命令或函数的处理程序

CaptureTheScreen处理程序是一个通用处理程序，也可将其视为一个函数。使用的最大差别是，一条命令本身是一个完整语句，无论是否返回值均可使用，但是函数是作为表达式的一部分而被调用。

示例：像调用命令和函数一样，调用处理程序

```
CaptureTheScreen "MacTest", 6                                   //像调用命令一样，调用运行处理程序。

put CaptureTheScreen ("MacTest",6)                              //像调用函数一样，调用处理程序，返回处理程序的值。（注意括号）
```

从另一个脚本调用处理程序

有三个方法，可从另一个脚本调用处理程序：

1. 使用's（所有格s）调用脚本名称，后面跟处理程序名称和参数。
2. 调用脚本名称，后面跟一个圆点和脚本名称。

3. 调用处理程序名称，后面跟*of*和脚本名称。

示例：从另一个脚本调用处理程序

```
run Logging's CaptureTheScreen "MacTest", 6                    //使用脚本名称和's（所有格s）调用处理程序。

run Logging.CaptureTheScreen "MacTest", 6                      //采用"脚本.处理程序"的格式调用处理程序。

run CaptureTheScreen of Logging "MacTest", 6                   //采用"处理程序of脚本"的格式调用处理程序。
```

测试跨平台应用程序

本节说明如何依靠现有脚本将测试扩展到多个操作系统。

创建平台和特定语言的套件

最初，看起来似乎能轻易为每个待测平台创建一整套全新的脚本，但是随着应用程序的演进，保持脚本的一致性就变得很困难。

一个长期有效的解决方案是首先创建一个基础脚本，然后为不同操作系统间会出现变化的各功能调用外部脚本。你可以为不同平台的套件编写不同的脚本，然后指示脚本在特定的运行中使用特定的套件。

示例：退出脚本

Windows套件中

```
TypeText ControlKey & AltKey & "x"
```

Mac OS X套件中

```
TypeText CommandKey & "q"
```

本例中，基础脚本可像调用命令一样调用脚本名称*Quit*。至于是运行Windows版本还是运行Mac版本，则取决于指定的套件版本。

设置正确的套件

`InitialSuites`这一全局属性会告诉脚本在查找脚本或图像资源时首先要核查的套件。（此类套件甚至优先于目前脚本运行的套件。）如果设置多个初始套件（initial suite），记住在转到下一个"高级"套件之前搜索各初始套件的帮助程序。

在上例中，应确保通过将`InitialSuites`的值设置为正确套件，基础脚本能够查找到*Quit*脚本。

示例： InitialSuites

```
Set the InitialSuites to ("Windows")                           //将Windows套件设置为首选套件，核查资源。

Quit                                                           //运行首先查找到的Quit脚本。
```

注：同所有全局属性一样，脚本运行过程中可随时更改 `InitialSuites`设置。（*Initial*是指搜索的初始套件，不是指脚本的初始状态。）

使用多语文本图像

如果界面元素仅以文本形式便可识别，则可在脚本中使用生成的文本图像代替捕获图像。这可以为创建多语套件节省大量时间。

例如，假设在一个应用程序中必须点击同一个英语、法语和西班牙语的打开按钮。你可以单独捕获每种语言版本的按钮图像，但是生成一个文本图像，在按需要更改文本内容会更加快捷。

示例： 使用特定语言的文本图像

```
Click (text:"Open", textStyle:"Button")

Click (text:"Ouvrir", textStyle:"Button")

Click (text:"Abierto", textStyle:"Button")
```

优化：翻译脚本

添加翻译脚本可使特定语言套件的使用更高效，如下所示。

示例：翻译脚本

```
params EnglishWord                                             //获取英文文字。

set translation to {Hello:"Bonjour",Open:"Ouvrir",Yes:"Oui"}   //将翻译变量设为属性列表，每一个英文输入对应一个法语值。

return translation's (EnglishWord)                             //返回英文输入对应的法语。
```

现在，可将`Text`值设为"翻译"脚本返回的值在基础脚本中生成文本图像。

示例：调用翻译脚本

```
Set the InitialSuites to ("French")      //确保"翻译"脚本取自翻译套件。

Click (text:Translate ("Hello"))         //运行EnglishWord参数值为"Hello"的"翻译"脚本；将"翻译"脚本返回的值用于属性列表中的文本值。
```

使用翻译脚本时，不要忘记创建一个与基础脚本语言相匹配的版本。虽然不需要真正将你会的语言进行翻译，但是基础脚本需要在遇上`Translate`调用时能运行任务。基础脚本到基础脚本的"翻译"脚本不需要翻译任何内容，只是获取参数并返回相同的参数。

示例：翻译脚本

`params BaseWord`	//获取文字。
`return BaseWord`	//返回相同的文字。

从主脚本运行

套件编辑器的计划面板包含一个很实用的批量管理和运行脚本图形界面，但是需要更灵活运行脚本时，必须编写主脚本。

主脚本最强大的功能是`RunWithNewResults`命令，使用该命令，可利用返回的脚本运行结果生成将来的计划。

`RunWithNewResults`将其它脚本用作参数。参数脚本自身生成结果，并返回给主脚本。这样做的好处是，可以根据之前运行返回的值，设置脚本，在特定条件下运行。利用主脚本可运行任意数量的脚本，以及管理运行结果。

下面的范例脚本说明了主脚本的功能。首先，运行初始测试。如果初始测试失败，以电子邮件向系统管理员发送告警；如果初始测试成功，则继续执行一系列测试，将结果以文本串格式，保存在测试结束时生成的日志文件中。

示例：主脚本

`set TestList to ("Test1", "Test2", "Test3")`	//创建一系列脚本执行任务。
`RunWithNewResults "InitialTest"`	
`put the result into Outcome`	
`if the status of Outcome is not "Success" then`	//如果“InitialTest”的结果不为“Success”（成功），记录运行的日期和时间。
`convert Outcome’s runDate to date and long time`	
`sendMail (to:"administrator@yourcompany.com", from:"JoeTester@yourcompany.com", subject:"Initial Test Failed", body:"Test run at" && rundate of Outcome && "had" && errors of Outcome && "errors")`	//发送电子邮件上报运行的日期和错误。（&& 表示将文本串用空格隔开。）
`else`	//否则
`repeat with each testScript of TestList`	//针对TestList中的每一个脚本。
`RunWithNewResults testScript`	//运行脚本，并将结果添加到Outcome。
`put the result into Outcome`	
`put testScript & ":"&& status of Outcome && Return after currentReport`	//将"脚本：状态"，和返回字符添加到currentReport。
`if the status of Outcome is "Failure" then`	//如果状态为 Failure（失败）
`run "CleanupScript"`	//运行“CleanupScript”。
`end if`	
`end repeat`	//运行完TestList中的最后一个测试后结束。
`Log "Final Results"`	//记录“最终结果”。
`repeat with each line of currentReport`	
`log it`	//记录currentReport中的每一行内容。
`end repeat`	
`end if`	

‹ 图像更新工具 up 收集和使用数据 ›

Printer-friendly version