TIB 自动化测试工作室
http://www.cnblogs.com/testware/

QTP 测试框架

-------By saffron

目录

测试框架产生的原因

1、 脚本文件过大：

X 主要是两方面原因导致，一是对象库的文件，默认生成得每个空的对象库文件为 192K，这样一个空的 QTP 脚本文件就至少需要 192K*2=384K 的空间（Action0 和 Action1），如果分割的 Action 多的话，占用的空间就更多。二是 Excel 的文件，同样由于分割 Action，每个 Action 需要使用一个独立的 Sheet，包括脚本中调用的 Action，这个在复杂的脚本中，表现得更加明显。

2、 文件数量过多：

X 一个最简单的 QTP 脚本，共有 4 个文件夹，15 个文件，当分割 Action 较多时，文件数与 Action 的个数呈正比上升。

另外，如果使用 Action 复用的方式的话，会在维护、转移、版本控件等方面存在巨大的困难。

3、 不利于查看脚本：

X 使用 Action 的方式来保存脚本，用户在查看相关脚本时，不得不需要打开 QTP，然后再把相关 Action 导入进去，这样将不利于脚本的查看。

4、 不便于脚本批量运行：

X 虽然 QTP 自带一个批量运行工具（Test Bath Runner）,通过 Bath→Add 的方法我们可以批量加载所要运行的 Action。但是如果你想要重新调整 Action 的执行顺序的话，那你接下来就有的你忙了。

一、解决方案：

1、 使用 Excel 中来代替 QTP 脚本中的 Action

√ 将脚本保存到 Excel 中，确保一个 Task 一张表，然后通过字符串拼接的方式来读取 Excel 中的脚本，同时脚本需要读取的数据，也和 Task 一起存放，这样可以直接减少 QTP 文件的数量。

2、 使用 Excel 来管理测试对象

√ 脚本所用到的对象统一放在一个 Excel 中，并将相同模块的对象放在一个 sheet 中。在运行相关模块的脚本前，先将某个模块的对象库提前加载进去，并且使用 Excel 来管理对象库也更加直观。

3、 使用 Excel 来代替 Test Bath Runner

√ 将测试计划保存到 Excel 中，可以很方便的更改测试的执行顺序，且每个人测试人员都拥有一张 TestPlan，可以更加清楚的了解测试人员编辑脚本的情况。

4、 使用 saffron 来封装脚本

X 以 B/S 结构的脚本为例：

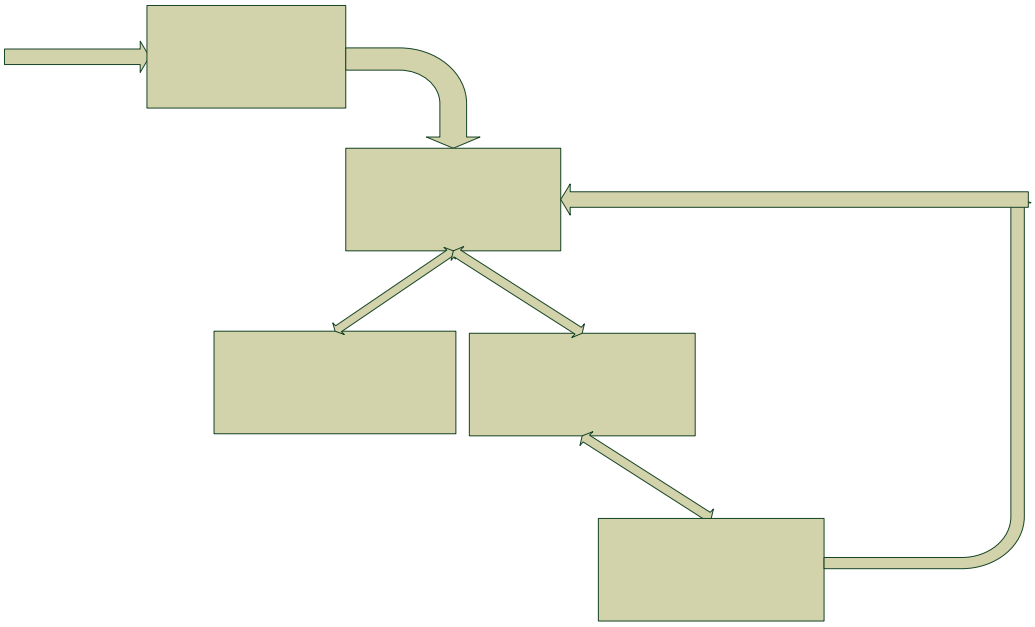
```
Browser(……).Page(……).WebEdit(……).set “text”
```

√ 使用 saffron 之后的脚本：

```
Obj_Set "^WebEdit","$username","$user"
```

大家可以看到，使用 saffron 后，我们的脚本看起来会更加的简洁，同时由于它是被封装过方法，我们可以在该方法中将测试记录也包括进去，这样就可以实现为每一个步骤产生一个记录。

二、框架运行图



数据的组织形式如下：

	A	B	C	D	E	F	G	H
1	IDX	TestCase_Nubmer	TestCase_Name	Script_File		Filter_Exp	Object_Repository	
2	x	测试用例编号	测试用例名称	脚本文件地址	调用脚本的方法名称	数据筛选条件	对象库文件地址	脚本所引用的sheet
3	x	脚本编写日期：2010-5-29						
4	√	TestCase001	登录系统	登录系统.xlsx	do_login	ID=1	ObjRepository.xlsx	login
5	x	TestCase002	退出系统	退出系统.xlsx	do_logout		ObjRepository.xlsx	logout

	A	B	C
1	IDX	Test_Case_Script	Test_Case_Data
2	x	测试脚本	测试数据
3	√	Public Function do_login() BrowseTo "$url" Obj_Set "^WebEdit","$username","$user" Obj_Set "^WebEdit","$password","$pwd" Obj_Click "^WebButton","$login"	TestData

	A	B	C	D	E
1	IDX	Control_Type	Key	Value	Object_Descrtiption
2	x	页面中控件的类型（不参与实际脚本运行，可省略）	将本列中所有的值写入Dictionary的Key中	将本列中所有的值写入Dictionary的Item中，支持多属性操作	页面中相关控件的描述（不参与实际脚本运行，可省略）
		页面中的对象			
		dit	username	name:=username	
		dit	password	name:=password	
		utton	login	name:=登录	
			register	name:=立即注册	
8	√	WebTable	table	name:=短信息	

	A	B	C	D
1	ID	url	user	pwd
2	1	http://www.foundersoft.com/bbs/login.aspx	在路上	19840222
3	2	http://www.foundersoft.com/bbs/login.aspx	zhangyong.sz	053203

说明：

IDX：当 IDX 为“ √ ”时，Driver 文件就会读取这一行，同时加载对象库、根据筛选条件来读取测试数据并运行脚本；如时 IDX 为：“x”时，表示不运行该行数据。

TestCase_Number：测试用例号

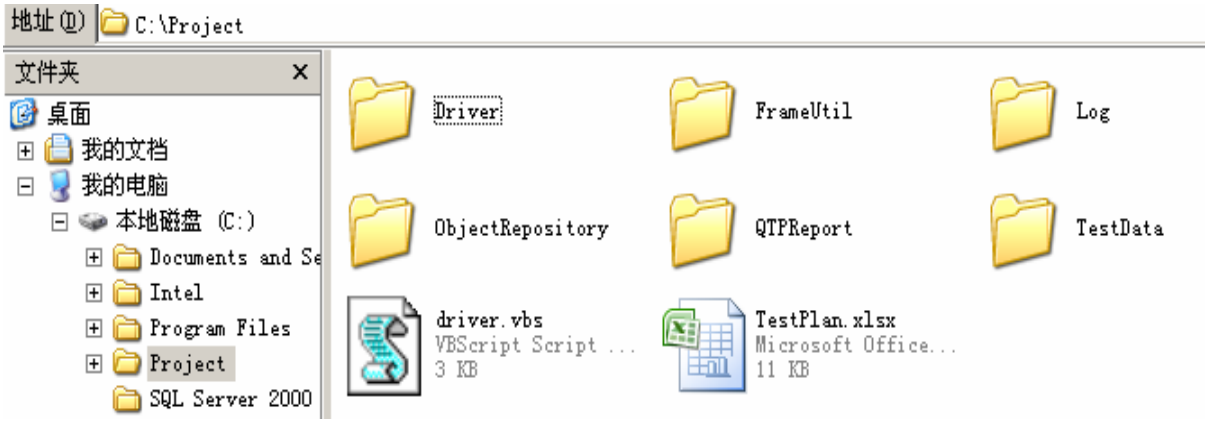
TestCae_Name：测试用例名称

Script_File：对应到测试脚本及测试数据的存放地址，并指示驱动这个脚本的方法名称

Filter_Exp：数据的筛选条件，这是只有运行 ID＝ 1 的数据

Object_Repository: 对应到测试对象库的地址，并指示存放该对象的 sheet 名

三、项目结构图



项目结构图说明：

Driver: 整个框架的入口，功能和 Test Bath Runner 类似，用于批量运行脚本。

FrameUtil: 存放用来支持框架的一些函数库，其中包括：Dictionary.vbs、Excel.vbs、diverUtil.vbs（驱动 Driver 的公共方法）、saffron.vbs（封装测试脚本的方法）

Log: 存放脚本运行后的日志

ObjectRepository: 对象库，用于存放脚本所用到的测试对象，功能和 QTP 中的对象库的概念一致。

QTPReport: QTP 的报表文件

TestData: 由于存放测试脚本及测试数据，我们将脚本及脚本所用到的数据放在一起，这样我们到维护脚本的同时，也可以维护到相关测试数据。

driver.vbs: 使用了 QTP 的 automation object model，也是整个框架的入口。可以直接执行该 vbs 脚本，因此可以做成 Windows 的自动任务，在指定时间点执行。

TestPlan: 测试计划，用于记录脚本的信息，根据相关的筛选条件来有选择的运行脚本，同时它也记载了相关脚本所用到的对象库文件地址及相关的测试数据地址，它使得将下面的 ObjectRepository 及 TestDat 关键起来。并且我们规定，每个测试人员都将拥有一张自己的 TestPlan，这样就可以很方便的来管理批量运行的测试记录。

四、脚本编写规范

我们的脚本是基于 saffron 来改写的，所以我们在正式编写脚本时，先看一下我们所要使用到的方法：

1．BrowseTo (url)

说明：

用于打开浏览器并导航到指定的 url，如果系统中已打开多个浏览器，该方法会将之前打开的浏览器都关闭后，再打开浏览器并导航到指定的 url。

示例：

BrowseTo "$url"

参数中的$符号，代表方法中所用到参数，是存储在不同地方的数据，这里指 TestData 中相关的数据。

当然，我们也可以将参数直接写进去，如：BrowseTo “http://www.baidu.com”

2．Obj_Click (objType, objProperty)

说明：

调用相关对象的 Click 方法（目前有以下几个对象支持 Click 方法：Link、WebButton、Image）

示例：

Obj_Click "^WebButton","$login"

参数中的^符号，代表的是测试对象；$代表方法中所用到参数，是存储在不同地方的数据，这里指 ObjectRepository 中的数据。

当然，我们也可以将参数直接写进去，如：Obj_Click "^WebButton","name:=login" name:=login 是用来描述 WebButton 对象的属性值。

另外，如果是运行 Link 或 Image 的话，我们脚本就得写成下面的样子：(下面其它方法也和这里的类似处理)

Obj_Click "^Link","$login"

Obj_Click "^ Image ","$login"

3．Obj_Exist (objType,objProperty)

说明：

调用相关对象的 Exist 方法（支持所有的对象）

示例：

Obj_Exist ("^Link","$register")

参数中的^符号，代表的是测试对象；$代表方法中所用到参数，是存储在不同地方的数据，这里指 ObjectRepository 中的数据。我们也可以将参数直接写进去，这里不做演示。

4．Obj_GetPropertyValue (objType,objIdent,objProperty)

说明：

调用相关对象的 GetROProperty 方法（支持所有的对象）

示例：

Obj_GetPropertyValue("^WebEdit","^value","$username")

这里有两个^符号，第一个参数中的^符号，代表的是测试对象；第二个参数中的^符号，代表的是该测试对象中的相关属性值（就是 WebEdit 的属性值）；$代表方法中所用到参数，是存储在不同地方的数据，这里指 ObjectRepository 中的数据。

整句话连起来的意思是：获取属性名为$username 的 WebEdit 对象中的 value 的属性值

5．Obj_GetValueByIndex(objType,objPropetry,intRow,intColumn)

说明：

获取 WebList 和 WebTable 中相关索引中的值。

示例：

Obj_GetValueByIndex "^WebTable","$table","1","2"

参数中的^符号，代表的是测试对象；$代表方法中所用到参数，是存储在不同地方的数据，这里指 ObjectRepository 中的数据；1 代表获取第一行；2 代表获取第 2 列。

整句话连起来的意思是：获取属性名为$table 的 WebTable 对象的第 1 行第 2 列中的值。

如果是获取 WebList 中的值的话，我们将最后一个参数设置为 0，如：

Obj_GetValueByIndex "^WebList","$list","1","0"

6．Obj_Select (objType,objProperty, text)

说明：

调用 listbox 、WebRadioGroup 中的 select 方法。

示例：

Obj_Select "^WebList","$list","$value"

参数中的^符号，代表的是测试对象；第一个$,代表 WebList 的属性值，这里指 ObjectRepository 中的数据；第二个$,代表 WebList 选择的项。

整句话连起来的意思是：获取属性名为$ list 的 WebList 对象中的名为$ value 的项。当然，我们可以使用索引，如：Obj_Select "^WebList","$list","1"

整句话连起来的意思是：获取属性名为$ list 的 WebList 对象中索引为 1 的项

7．Obj_Set (objType,ObjProperty, text)

说明：

调用 WebEdit、WebCheckBox、WebFile 中的 Set 方法

示例：

Obj_Set "^WebEdit","$username","$user"

参数中的^符号，代表的是测试对象；第一个$,代表 WebEdit 的属性值，这里指 ObjectRepository 中的数据；第二个$,代表 WebEdit 中输入的值，这里指 TestData 中的数据。

整句话连起来的意思是：给属性名为$ username 的 WebEdit 对象设置一个$user 的值

五、脚本编写示例

下面是一个登录脚本的示例代码，其中的 do_login 应与 TestPlan 中的方法名一致：

```
Public Function do_login()
    BrowseTo "$url"

    If Obj_Exist("^WebEdit","$username") Then
        Obj_Set "^WebEdit","$username","$user"
        msgbox Obj_GetPropertyValue("^WebEdit","^value","$username")
    End If

    If Obj_Exist("^WebEdit","$password") Then
        Obj_Set "^WebEdit","$password","$user"
        msgbox Obj_GetPropertyValue("^WebEdit","^value","$password")
    End If

   If Obj_Exist("^WebButton","$login") Then
       Obj_Click("^WebButton","$login")
   End If
End Function
```