自动化测试新手上路

## 一、初识庐山真面目

自动化测试离不开工具，我们首推测试界声名显赫的“QuickTest Professional 10.0”，简称QTP。至于推荐他的理由，大家可以到网上一搜便知。

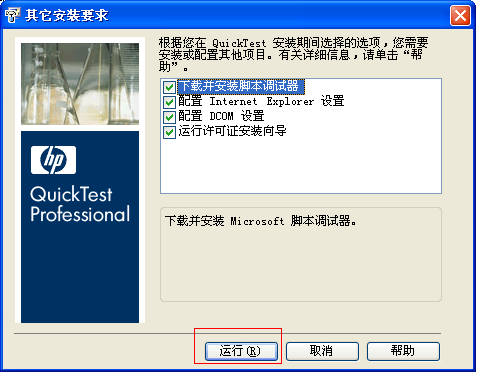
### 1、安装QTP

首先第一个环节便是介绍：如何安装这个让人爱不释手的工具—QTP。

1. 找到工具：途径有两个，一个是拿着U盘找安装过的同事copy；另一个是到测试部共享的工具服务器上下载，地址：\\10.1.145.152\测试部\Tools，文件名为：QTP10 安装文件.rar。推荐，第一个途径，能够又快又准确的找到你想得到的东西，并且在联络感情之余可以随便请教安装与试用之道，当然这一定要看当事人是否有这个时间。如果人家没有这个时间也别急，请耐心看完此文档。
2. 检查安装环境：
   1. C盘空间：因为我们推荐将QTP安装在C盘，所以请检查C盘是否有足够的空间，安装完的QTP大概640M左右。
   2. 是否有病毒：建议杀一下病毒，之前有同事因为病毒安装不成功的案例。
   3. 暂时关掉某些杀毒软件：例如已知的杀毒软件“诺顿”、“卡巴斯基”。原因是这两个杀毒软件会将我们的特别文件当做病毒给杀掉，无法顺利安装试用QTP。
   4. 检查IE版本：理论上QTP是支持IE6、7、8的，但IE8需要另下补丁，IE7也遇到一些不可理喻的问题，所以在安装QTP之前烦请将IE7或8卸载，降至IE6。
3. 开始安装。

得到安装文件后，解压，点击“”，一路下一步，注意：开始有一个步骤需要安装QTP需要的组件，一个是“.net framework 2.0”和一个关于C++的东西，不能跳过，必须安装。

直到见到如“图一”的页面，停下来确认是否能够上外网。最好是能够联网安装，因为这一步有一个“下载并安装脚本调试器”，由于不明确这个调试器是个什么东西，也就没找到相关的安装包。不安装此调试器的直接后果是，无法调试你编的QTP脚本，很麻烦。

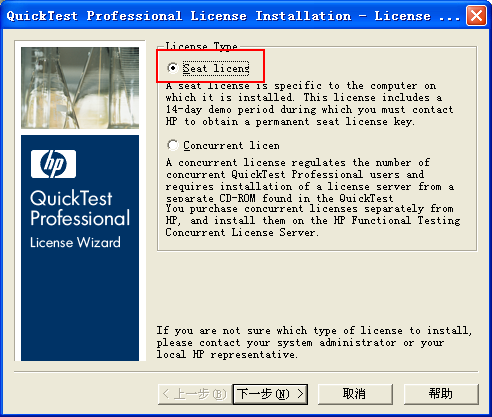


**图一：其他安装需求**

如果实在不能联网安装，则将“下载并安装脚本调试器”前面的复选框取消勾选即可以继续安装，待能够上外网的时候再说（重新打开此页面的位置在：开始🡪程序🡪 QuickTest Professional🡪Tools🡪Additional Installation Requirments），其他三项必须保留选中，然后点击【运行】按钮。

本来到这一步应该是最后一步了，但为了能够正常使用QTP，我们还要这样做。

当点击“运行”的下一步，会弹出“运行许可证安装向导”，选择第一个选项点击【下一步】，如图二所示：



**图二：运行许可证安装向导第一步**

再点一次下一步，见到图三：



**图三：输入许可证密钥**

再在安装压缩包中找到mgn-mqt82.exe文件，将其copy到路径：C:\Program Files\Common Files\Mercury Interactive下，在这个目录下一般都会存在一个名叫“License Manager”的文件夹，如果没有请手工创建。然后执行mgn-mqt82.exe文件，他会在“License Manager”文件夹中生成名为“lservrc”的文件，将其打开。例如：X8AWUP6RQ763KBKC7OS5CEYACKV8P5MSABJT4QSNJ7X8NYZNKZ58CXPJHDQSTJO73Y6QVXR5BR# "QuickTestPro" version "6.0", no expiration date, exclusive

6O54XNSPNDI8RUIZWNAFRJTY4KJIWHM6KXCHWFNLUFE4H6ZMR2GCUCCV7DL8XGJIK6E2LM# "FT-Unified" version "1.0", no expiration date, exclusive

我们取文档中第一个出现“#”的位置，copy“#”之前的字符串，例如：“X8AWUP6RQ763KBKC7OS5CEYACKV8P5MSABJT4QSNJ7X8NYZNKZ58CXPJHDQSTJO73Y6QVXR5BR”，到图三中粘贴到输入框内，然后不用管任何提示，下一步到安装完成。

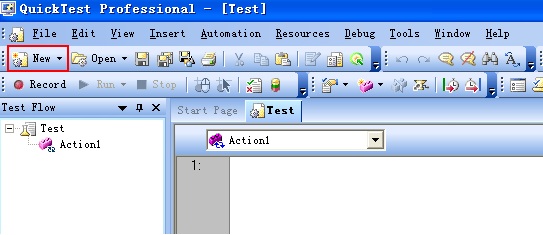
到这一步，你的QTP已经安装成功，赶紧去试用一下吧。（根据以往经验，个别机器需要重启计算机，大多数都不用。）

### 2、你的“HelloWorld”

费劲周折的安装好工具是不是有点喜悦呢？别急，更喜悦的还在后面，下面请跟我共同小用一下被HP称作“宝刀屠龙”的QTP吧。（编者注：“神剑倚天”即是业内传颂的LoadRunner，当然本故事纯属编者梦呓，请勿当真）

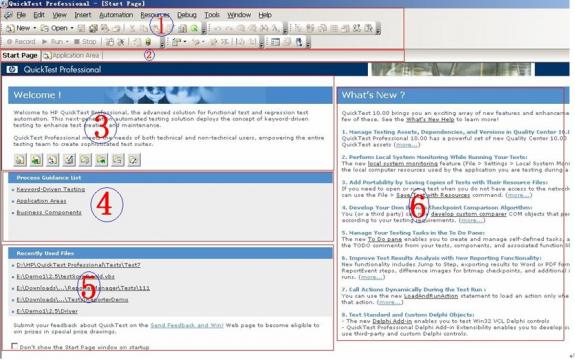
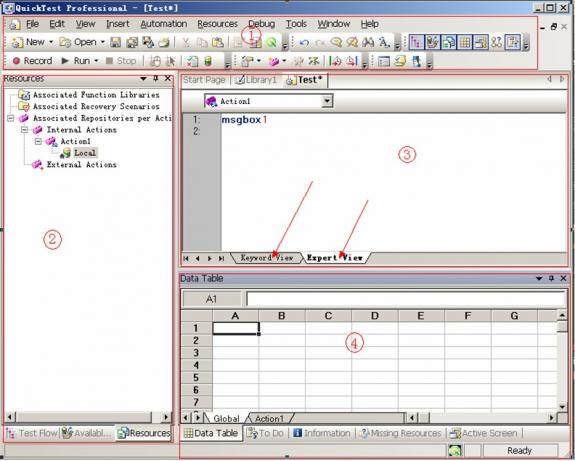
首先，开启我们的“HelloWorld”之旅吧。

第一步：创建脚本文件，点击“New”按钮，如图所示。



第二步：保存脚本文件，菜单项：File🡪save。提示：请将QTP默认存放脚本的路径改成其他盘符，以免系统崩溃造成不必要的损失。

第三步：QTP操作界面简介：

1、Add in Manager 插件管理界面   
程序启动后程序会停留在插件选择的界面    
  
该页面由用户选择需要加载的插件。   
这是一个QTP插件管理器，每次启动前需要选择对应的插件才能进行测试。   
插件的含义与作用：   
Add-in 的选择是为了能够成功识别对应Add-in的测试对象控件，也就是说是和被测控件有关，而跟什么什么语言写的是没有关系的。   
2、QTP开始页   
选择好插件后点击OK按钮就会出现以下界面    
  
  
（1）用1框选中的区域是我们常见的菜单栏和工具栏，也是我们最常用的地方下一课时我将对qtp中比较重要的一些菜单进行讲解。   
（2）用2框选的区域是我们目前所打开和创建的一些qtp文件，由选项卡形式呈现出来   
在我们再做一个项目时，常常是由多个文件组成，这时就在2区域来回切换比较方便。   
（3）3区域是由几个打开和新建qtp文件的快捷菜单按钮。前4个为新建，后4个为打开。   
创建有些qtp文件时需要与qc连接，在以后的课程会具体提到。   
（4）4区域是创建几种qtp类型文件的向导。不做重点。   
（5）该区域用来显示最近打开的qtp文件   
（6）该区域显示qtp10.0一些新的东西的说明。可以稍微了解。   
3、test项目界面   
菜单File-new-Test，新建Test类型qtp项目文件，或者使用快捷键Ctrl+N，或者使用开始页的快捷菜单新建，或者使用菜单栏中的new按钮新建（供大家选择）   
新建后我们会看到如下的界面    
    
  
  
（1）1区，主要是菜单栏和工具栏，下面简单介绍下工具栏中的按钮作用。   
（2）2区，该区类似于vs中的解决方案资源管理器，用来显示项目的组成目录、可用的关键字以及一些测试资源，通过该界面一目了然。   
（3）3区，该区属于我们的工作区之一，该区域有两种视图模式，分别是keyword view（关键字视图）和Expert View（专家视图）。关键字视图主要显示每个步骤的操作对象操作方法和值可以很直观的看到，专家视图主要是把关键字视图中的所有内容用代码形式体现出来。 我们通常用到的是专家视图。  
（4）4区，该区由图可以很直观的看到，分为数据表、to do、信息栏、缺少的资源、快照。   
数据表主要用来存放数据用的，可以在测试时调用数据表中我们所设置的数据，达到参数化的目的。Todo暂不做了解，该工具也是qtp中比较诱人的一块，以后可能会提到它。信息栏，我们在检查脚本语法编译情况时，若有错误会在该区域中显示。missing Resources区，用来显示我们当前打开的qtp项目所缺少的资源。Active Screen快照区，qtp录制时的一些快照，录制结束后，可以在快照中进行检查点的插入等操作，不用在录制状态进行插入，录制时所抓取的快照，受tools-options菜单中的active Screen配置有关，具体可以到该菜单下去了解。    
  
①②③主要是新建开打保存菜单按钮   
④剪贴复制粘贴   
⑤第一个按钮没试过（qtp启用编辑状态）第二个按钮会常用到对应File—setting菜单，用来设这运行策略、资源、环境、参数等配置。后面有用到的地方会具体讲解   
⑥撤销注释，撤销注释，查找，替换   
⑦各种窗口显示按钮，各位将每个按钮都点击一片，在观察窗口有什么变化就知道了。也是一个比较常用的菜单。   
⑧开始录制脚本、运行脚本、停止脚本录制。经常会用   
⑨模拟录制和低级录制，在有些场合，对象识别不了，我们可以使用模拟录制和低级录制方法来解决，但是这种录制方法受软件所处的坐标等因素影响，不建议使用。不灵活   
⑩脚本运行结果和对象库，对象库是用来存放录制时所操作的对象，也是qtp程序中最主要的东西之一。   
第十一区对应的是插入输入值、检查点，插入或调用新操作，切割Action，步骤生成器   
输出值和检查点主要是对某对象的值输出和检查某对象的值，插入或调用新Action以及切割Action在后面课程会具体介绍暂用不上，步骤生成器是用来生成脚本的，例如qtp自身带的一些对象和方法，就可以用步骤生成器输入参数生成调用该方法的一段脚本，不知道所调用qtp自带的方法怎么使用可以点击该界面的一个问号按钮，就可以找到对象的帮助文档。   
第十二区是用来插入事务的，与LR中的事务一样，qtp中的事务也主要是为LR调用qtp时所用。这里不多介绍   
最后一区对应的是 选项，检查编译，对象间谍插件。选项对应Tools---options菜单后面的课程会讲解几个重要的菜单会详细讲解它。检查编译是对当前的脚本进行错误检查，看编译是否通过，是否有语法错误，若有错误会在information区显示出来。对象间谍工具比较重要，也是经常会用到的地方，主要用来查看被测对象的属性等信息

第四步：开始录制、回放脚本。

1. 点击【Record】按钮。默认弹出设置窗口，对录制和回放进行设置，如下图：



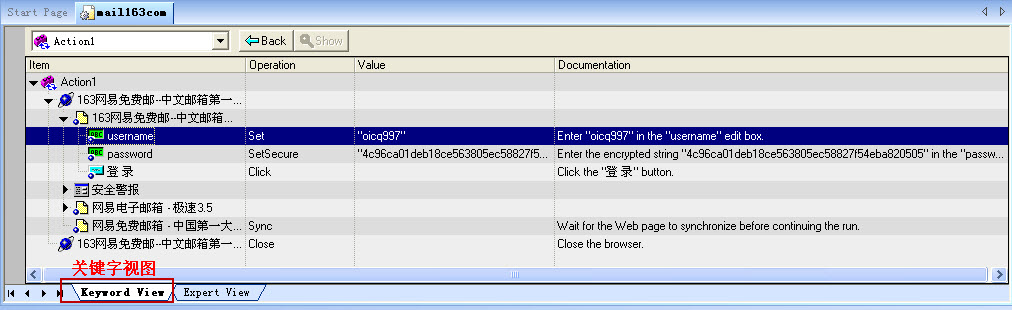
设置项解释：

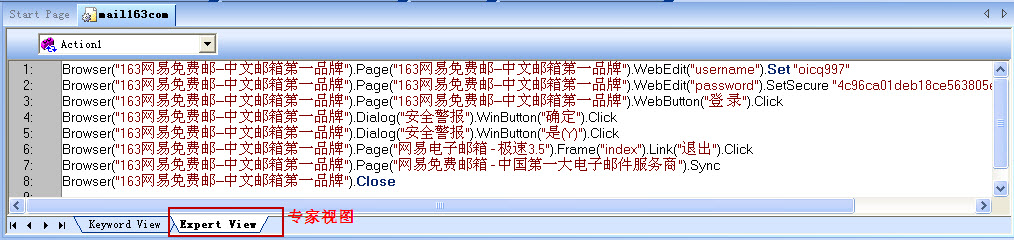
* 录制并运行已经打开的浏览器，打开浏览器的操作需要手工介入。
* 打开浏览器的操作由QTP来执行，如图所示，我们只需设置URL和浏览器类型即可，例如：http://mail.163.com，浏览器类型使用默认的IE。

1. 开始录制，你只需按照你预想的操作步骤完成一次操作，QTP便会记录下操作轨迹，同时生成脚本代码。
2. 回放代码，点击【Run】按钮即可回放，QTP会按照之前录制的步骤重新播放一次操作步骤。

**录制完脚本后，你应该继续了解：**

1. 学会查看两种方式的脚本视图（关键字图和专家视图），我们经常用到的是专家视图，也称代码视图。





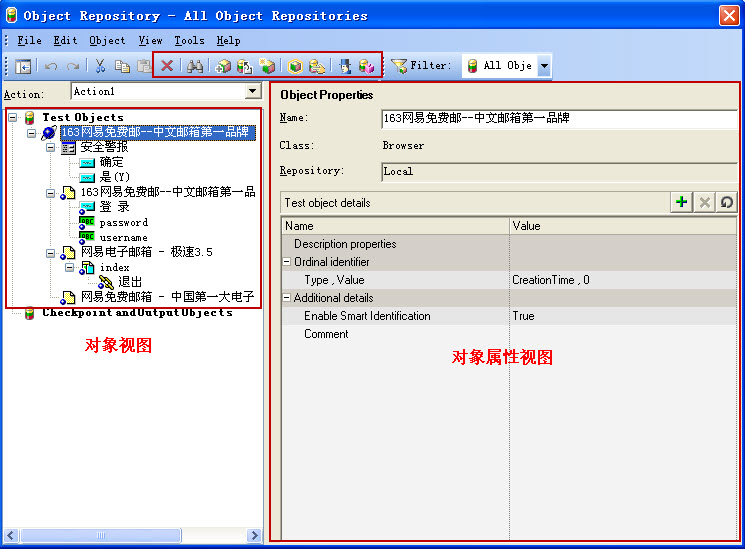
1. 学会查看脚本内容



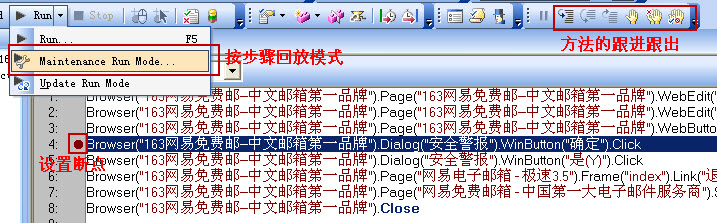
* **脚本生成方式：**实际上，脚本的生成是按照树形结构的方式安排的，这个情况可以与对象库对应观察。这种模式的好处就在于，无论有多少个测试对象，无论测试对象的位置多复杂，都可以很快的通过这种唯一路径迅速找到，便于定位。也为QTP的高级应用描述性编程做好了铺垫。
* **脚本过程解释：**
  1. 在输入框输入用户名“oicq997”。
  2. 在输入框输入密码“\*\*\*\*\*\*\*”。
  3. 点击【登录】按钮。
  4. 弹出的安全警报对话框。
  5. 第二个弹出的安全警报对话框。
  6. 登出操作
  7. Sync的意思是等待页面刷新完成。
  8. 关掉浏览器。

1. 学会查看对象库

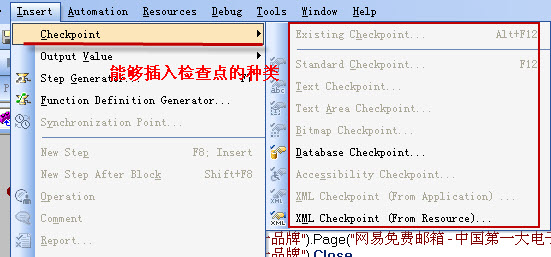
对象库是QTP非常关键的技术，他能够将已知的大部分网页空间做成标准的对象类，通过页面不同对象的属性来区分实际对象。QTP的对象库也是按照树形结果排列的，这与脚本的结构是一致的。



1. 学会DEBUG，与其他编程语言基本一致。



1. 学会插入检查点，检查点是QTP用来设置我们测试过程中需要验证的某一步骤。QTP支持如下几种检查点的类型。

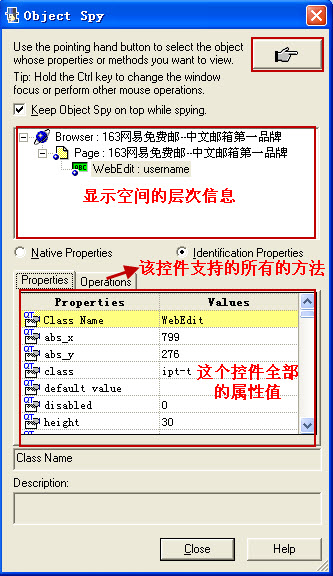


1. 学会使用Object Sby，非常重要的一个功能，在编写脚本的时候会经常用到。

使用方法：点击红框中的手指按钮，QTP会切到浏览器页面，用小手点击哪个控件，哪个空间的属性信息就会被读取出来。

NativeProperties：显示在执行测试对象的时候所获取的测试对象的属性和方法。

IdentificationProperties：显示在录制或编写测试步骤时所获取的测试对象的属性和方法。



至此，QTP的基本功能已经交代完毕，要想了解更多，请SVN下载更多的资料继续学习。SVN路径：svn://10.1.145.249/Autotest/DCSM\_Autotest/02 New Tester Readme/自动化测试-新手包/03 资料查阅/QTP基础，推荐文档：《QTP8 Tutorial\_cn.pdf》

## 二、初出茅庐，小试牛刀

至此QTP的基础知识已经普及的差不多了，从现在开始进入实战状态，请各位打起精神来。实战部分将分为两个部分介绍：自动化测试的框架介绍和QTP的高级应用-描述性编程。

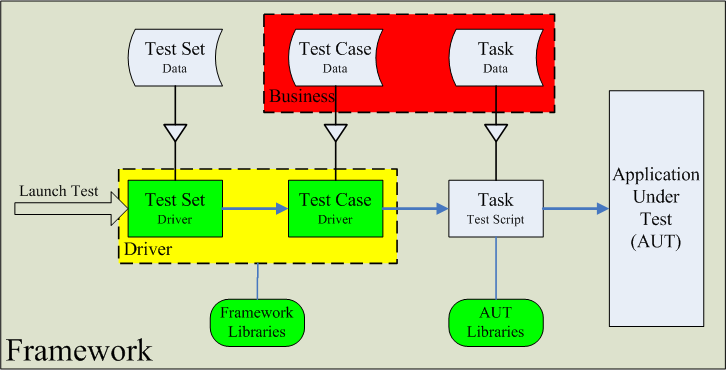
### 1．框架介绍

在实际的工作中，光有一件称手的武器是不够的，我们往往还需要准备一本能够驾驭这件武器的秘籍，才能将武器发挥到极致。也就是说，如何能够使QTP应用到实战当中去为我们发挥它的作用呢，这就需要有一个比较合适的框架来规范我们的测试行为，使自动化测试有章可循、有法可依。

#### 1.1解决方案概述

* 用VBS的Function代替QTP脚本中的Action。
  + 不使用Action复用，而使用Function的加载和调用。直接减少QTP脚本的数量。
* 使用单一的QTP脚本入口。
  + 这样整个工程中，就只有一个QTP脚本，其他的都是VBS文件，并且没有了过多的Excel文件。确保冗余文件达到最少。
* 数据文件统一维护。
  + 将所有脚本需要用到的测试数据统一放到一个或多个Excel文件中，方便了维护，同时也减少了Excel文件的数量。
* 利用描述性编程代替手工录制
  + 可编程性有利于脚本的拓展和管理，可以处理相对复杂的检查点。
  + 可以舍弃占用空间大，维护不方便的对象库。

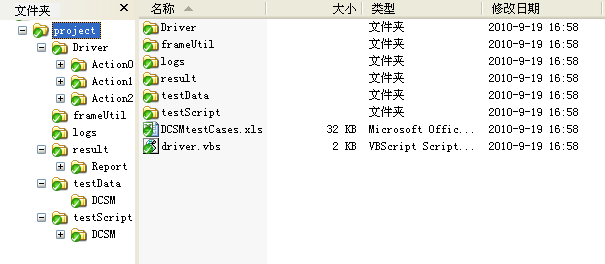
#### 1.2框架图示



#### 1.3体系结构

1. **Test Set驱动脚本**
   * 初始化
   * 解析Test Set的数据文件
   * 调用TestCase驱动脚本，向它传递测试用例的文件名
   * 如果有必要，调用框架所必需的库函数
2. **TestCase驱动脚本**
   * 解析TestCase的数据文件
   * 根据TestCase的解析结果，加载测试脚本，加载测试数据
   * 根据TestCase的内容，调用Task的测试脚本，并且把测试数据的集合传递给测试脚本
   * 如果有必要，调用框架所必需的库函数
3. **Task测试脚本**
   * 执行指定的任务（如输入数据，点击按钮等）
   * 生成日志，以及测试报告
   * 如果需要则调用用户自定义库函数
4. **库函数**
   * 一般的和具体应用程序的函数可以被调用，以执行指定的任务。

#### 1.4框架的存储结构

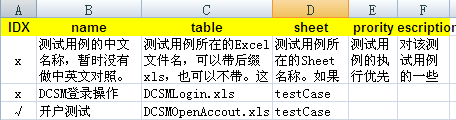


解释：

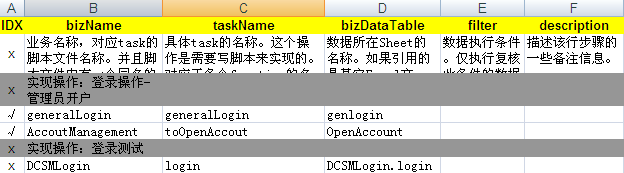
* Project文件夹，整个工程的最高一级目录，名称可以修改。
* Driver目录，这个是QuickTest的脚本文件，整个框架的入口。这个脚本包含了testSet和testcase的驱动脚本。
* frameUtil目录，存放用来支持框架的一些函数库。
* logs目录，存放脚本运行日志
* testData目录，存放测试用例以及测试数据的Excel文件
* testScript目录，存放task脚本，全部存储为vbs文件。
* driver.vbs文件，使用了QTP的automation object model，也是整个框架的入口。可以直接执行该vbs脚本，因此可以做成Windows的自动任务，在指定时间点执行。

注意：testData和testScript目录下的内容，是真正需要开发的。

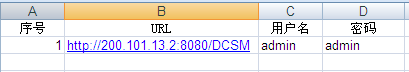
#### 1.5数据的组织



**Test Set**

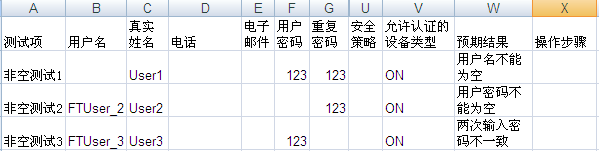


**Test Case**





**Test Data**



Sheet名为：OpenAccount

解释：

**TEST SET：**

* Test Set数据文件是用来管理测试用例文件的，在这里，扮演了TD的管理测试用例和脚本的角色。很显然，这个数据文件没有TD的功能强大，不能体现测试用例对需求的覆盖，没有测试用例之间的树形结构…但是作为一个轻量级的测试框架，只要能够记录、管理50~100个测试用例文件，就暂时够用了。
* **IDX**：index，“√”表示该行数据有效，“x”表示该行数据无效。主要是考虑到Excel直接面对人，用0，1来标识有效无效，很不直观。
* **name**：测试用例的名字，用中文标示即可，这个字段只是让人看起来比较直观，并不会被用到。
* **table**：这个字段非常重要，它指向测试用例所在的Excel的文件名。可以带后缀，也可以不带后缀。不需要指定文件所在的路径。
* **sheet**：表示测试用例在Excel文件的Sheet名称。如果不填写的话，默认为第一个Sheet。
* 考虑到有些测试用例极其复杂，仅仅使用Excel形式的测试用例是远远不能实现其复杂的逻辑，也可以把测试用例写成vbs文件，直接执行该vbs脚本。目前暂未实现。

**TEST CASE:**

* TestCase数据文件中记录的就是测试用例，只要不是太过于复杂的测试流程，都可以直接写在Excel文件中，当然，需要符合一定的规范，很显然，这离自然语言还是有一定的距离 。这种形式比较适合step-by-step的测试用例，并且粒度比较粗，减少了测试用例的步骤数。
* **IDX**：index，“√”表示该行数据有效，“x”表示该行数据无效。
* **bizName**：被测试的业务的名称，比如说“银行收款”这个业务。事实上，这个名称需要与存储改业务的task脚本的名称保持一致，而且需要与task脚本中定义的class的名称也保持一致。在没有做名称的中英文对照字典之前，bizName最好使用英文名，所以最好使用“collection”，而不是“银行收款”。
* **taskName**：task的名称，原子业务的名称，比如新增、修改、删除等。这个名称与task脚本中定义的各个Function的名称应该保持一致。同样，暂时也最好不要使用中文。
* **bizDataTable**：测试数据所在的Excel的Sheet名称。比如说进行登陆操作时，需要输入用户名和密码等数据，这些数据单独放在某个Excel文件中的某个Sheet中，把这个Sheet的名称写到bizDataTable这个字段中即可，框架会自动去加载这个Sheet中的所有数据。
* **filter**：测试数据的过滤条件。可能准备的测试数据比较多，但是在当前这一个step中，只需要执行某一条或几条数据，就可以使用filter这个条件。比如说，登陆时，想用错误的用户名和密码来登陆，那么可以这样写：用户类型=baduser。当然“用户类型”是测试数据所在Excel表格中定义过了的字段。

**TEST DATA：**

* Test data数据文件，用来存放测试数据。没有区分输入数据、验证数据、以及输出数据，总之，只要是测试过程中，需要用到的数据，都一古脑的做成一行，放到这个文件中。一是觉得这样在设计测试数据时，比较方便和直观，二是也没有更好的设计思想。
* 对业务的验证，按照设想，是通过日志文件和测试报告来体现的。所以把验证数据与输入数据放到一起，亦不会有太大的不妥。

**数据文件总结：**

* 基于把测试设计和脚本开发分开的思路，设计了这三个Excel表格。测试设计时，主要是设计testcase和test data这两个Excel表格。毕竟，这才是自动化测试最核心的价值所在。脚本写得再完美，没有好的测试设计、测试数据，对测试本身并不会有太大的帮助。因此，希望尽可能把这些Excel表格做得更易用。所谓的框架，就是把这3个Excel表格串起来的东西。

#### 1.6驱动脚本的逻辑

**Test Set 驱动脚本伪码:**

1. 将test set数据所在的整个Sheet加载到QuickTest的Runtime Table中。
2. 检查test set数据文件中一共有多少行记录需要执行。
3. 读取一行数据。
4. 如果该行的IDX的值为“√”，那么调用执行TestCase驱动脚本，并且把测试用例所在的Excel文件名和Sheet名作为参数传给TestCase脚本。
5. 转向第3步，直到所有数据被执行完毕。
6. 删除加载到Runtime Table中的所有数据。

**TestCase 驱动脚本伪码:**

1. 将TestCase所在的Sheet加载到QuickTest的RunTime Table中。
2. 检查TestCase数据文件中一共有多少行记录需要执行。
3. 遍历整个Sheet，加载所有需要被用到的Task脚本：逐行读取有效数据（IDX=“√”），如果bizName的值所指向的Task脚本没有被加载，那么加载之，直到所有数据被执行完毕。
4. 遍历整个Sheet，加载所有需要被用到的Test data数据：逐行读取有效数据（IDX=“√”），如果bizDataTable的值所指向的Sheet（该Sheet的名称应该在本测试用例中是唯一的）没有被加载到RunTime Table中，那么加载之，直到所有数据被执行完毕。
5. 获取一行的测试步骤的数据。
6. 如果该行的IDX的值为“√”，那么解析filter信息，形成条件语句。
7. 逐行读取test data所在的Sheet，如果该行数据是满足条件语句的，那么调用Task脚本，执行对应的操作。
8. 转向第5步，直到所有数据被执行完毕。
9. 删除加载到Runtime Table中的所有Sheet。

**Task 脚本说明:**

脚本结构很简单，与其他市面上常用的代码结构非常类似。

首先要定义一个类，类名要与脚本名称一致。目的主要是为了在调用某个Function时，如开户（toOpenAccout），会去找指定Class的toOpenAccout方法，而不会出现找错的情况。一般情况下，一个功能模块只对应一个类。例如，用户管理的增删改查功能模块，脚本名称建议起作AccoutManagement.vbs,那么class的名称也要对应的起作AccoutManagement。

其次，在类中加入方法，类中方法的数量与名称要根据业务的实际需要，视情况而定。例如，用户管理的增删改查，我们可以在类中加入四个方法，即：toOpenAccout(Sheet\_Name)，toDelAccout(Sheet\_Name)，toChangeAccount(Sheet\_Name)，toFindAccount(Sheet\_Name)。其中，方法的参数是固定的“Sheet\_Name”，这个参数是用来接收被我们参数化的测试数据，所以为了配合框架的功能Task方法的参数有且只有一个参数，不支持多个参数。各个Function脚本，需要全部使用描述性编程来实现，不能采用录制的方式生成。

最后，能够公用的方法尽量提取出来放到公共类中，以便减少重复代码，增加代码的利用率，节约开发和管理成本。切记，这一点非常重要。例如，登录和注销操作即可以当做公共方法，公共方法的放置地点一般在框架文件结构..\testScript\DCSM\ generalFunction\文件夹下。

**脚本模板：**

‘用户管理模块

Class AccoutManagement

‘用户管理—开户的Task

Function toOpenAccout(Sheet\_Name)

Msgbox(DataTable(“用户名”,Sheet\_Name))

……

End Function

‘用户管理—删除用户的Task

Function toDelAccout(Sheet\_Name)

Msgbox(DataTable(“用户名”,Sheet\_Name))

……

End Function

End Class

#### 1.7框架情况总结

* **优势：** 
  + 脚本文件少，占用的空间也少，方便管理和维护。
  + 可以使用版本控制工具对单个的数据文件或者vbs文件进行版本控制。
  + 测试设计和脚本开发解耦。
  + 测试用例和数据的展现更加人性化。
* **缺点：** 
  + 异常的捕获考虑得很少。
  + 日志只是输出到文件，不利于查看。如果把日志输出到数据库，就可以生成相应的测试执行报告。
  + 测试用例的管理太过于简陋。
  + 无法定制测试报告。

### 2．QTP的高级应用—描述性编程

在这个文档中，我只做简单的概述，相关资料请查阅：

svn://10.1.145.249/Autotest/DCSM\_Autotest/02 New Tester Readme/自动化测试-新手包/03 资料查阅/描述性编程相关

描述性编程的意义：说白了就是丢掉QTP在录制过程中存储的对象库，采用一种描述的方式来表达浏览器上的控件。

描述编程的实质：就是利用测试对象（浏览器控件，例如WebButton等）本身的属性来定位测试过程中的控件，例如登录按钮其中有个属性叫“name”，值为“login”，QTP提供一种方式来描述这个按钮。

描述性编程的具体格式和做法，请参阅资料文档，这里就不做赘述了。推荐文档：《QTP描述性编程介绍.pdf》

## 三、木已成舟，只欠东风

实际上，以上的文档应该足以应付我们的自动化测试工具了，但总觉得差了点东西。什么东西呢？对，那就是经验和技巧了。在这一章节，我将主要针对在实际工作中能够应用到的相关经验和技巧做以说明。让大家少走弯路，尽快上手。

#### 理论知识都懂了，在实际工作中将如何开始自动化测试之旅呢？

答：

1. 前提： QTP已经成功安装。
2. 准备测试框架。将自动化测试框架从SVN中check下来。地址是：svn://10.1.145.249/Autotest/DCSM\_Autotest/DCNAutoTestFrame。建议将框架的目录，也就是project目录直接放到某个盘符的根目录下，这样有助于框架寻找定位数据表和Task脚本。对照以上的框架介绍，熟悉框架内容。
3. 此时，如果测试用例已经设计完毕则直接拿着测试用例进入编码工作。如果没有，则先设计测试用例（测试用例的设计技巧稍后奉上），在进行脚本开发，我们不推荐没有用例直接开发脚本，这样很可能造成文不对题而进行不必要的返工。
4. 设计测试用例是需要评审的，无论是同一设计或者是单独设计都需要至少将测试计划中定义的三个角色（测试（框架）负责人、测试用例设计人员、自动化测试脚本开发人员）召集一起进行评审，如果有必要最好请当时手工测试该模块的测试人员一起评审，审议通过够方可使用。评审的衡量标准主要是：测试数据的完整性，测试检查点设计的合理性，测试用例覆盖率的合理性，测试步骤的是否清晰等。
5. 测试脚本需要进行三个步骤才能够提交至SVN服务器。首先，不带测试数据（不在框架中执行）的情况下调试通过，这个步骤是保证测试过程各个控件对象识别的正确性；其次，在本机带上测试数据（将脚本加入框架中执行）调试通过，这个步骤的要点是保证用例中提到检查点的覆盖程度和测试数据与脚本的匹配程度。最后，提交给负责人进行最终审核后，提交至SVN服务器，这一步重点保证脚本格式是否正确、注释是否完备、是否有重大检查点遗漏等。
6. 最终脚本的执行统一由测试（框架）负责人来负责。
7. 测试报告的搜集和发布也统一由测试（框架）负责人来负责。

#### 2．设计自动化测试用例有何技巧？

答：

最好是找到该模块的测试人员，要一份之前通过评审的测试用例。我们可以以此为基准设计我们的测试用例。实际上，两种测试用例的内容是差不多的，只是侧重点和表达方式不同罢了。这样的好处在于，能够最大程度的重用或借鉴之前的测试用例，节省出来精力和时间用来设计测试数据，自动化测试用例的重点在于能够建立有效、完备的数据驱动，道理就在这。

在设计测试数据时，我们不要求把每个输入框的各种情况都要覆盖，但要求在这个测试用例中至少要将出现的各种输入异常考虑全面。

最好用某种方式将测试数据规划清楚。例如“管理员开户”的案例中用到的等价类划分法。这样的好处在于可以更加清晰，全面的设计测试数据，提高测试有效性。

#### 3．脚本开发的三字箴言。

脚本的开发大体分为四步：找对象；编逻辑；写报告；入框架。

第一步：找对象。千万不要误会，我们这里的对象应该理解为测试对象，也就是web控件。根据本人的经验，首先要录一遍测试步骤，这样对象基本加入到QTP的测试库中，以便随时查看控件属性，查找并定位控件。要熟练使用Object Spy，这个工具能够更完整的将控件的属性显示出来，而且还能列出该控件支持的所有方法。其次，将录制时生成的脚本用描述性编程的方式修改或重写一下，然后将对象库中的对象删除，脚本依然能够运行，说明找对象成功。先录脚本的另一个好处是在编程时不用再考虑测试步骤的问题，因为步骤已经录好，只需专心找控件属性即可。还有一个提示是，录完脚本后可以将对应的测试步骤当做注释copy到脚本文件中。

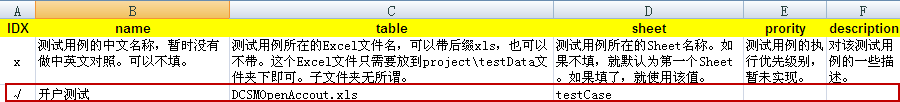
第二步：编逻辑。这里面出现最多的就是“IF else EndIf”了，这里可能会出现很多个IF嵌套，建议这种嵌套最多嵌三层，如果写多了，过一段时间后即便是开发脚本的人也不一定能够很清晰的读懂这里面的逻辑了。还有就是写足够的注释，至少要让接受这个脚本的人能读懂。

第三步：写报告。QTP自己集成了一个报告生成器，我们可以在代码中插入不同类型的报告描述，例如Reporter.ReportEvent micPass, "测试用例A-执行成功",“登录信息判断正确。”，将成功的标志插入到QTP报告中，并写上标题和描述。

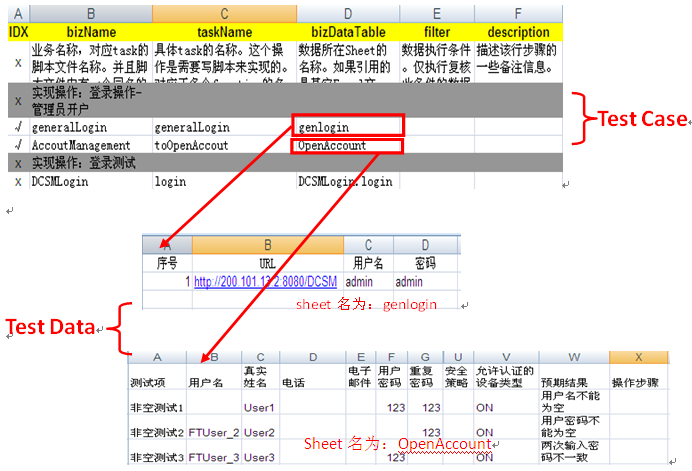
第四步：入框架。脚本调试通过后一定要放到框架中跑几次，并认真的观察执行过程，在这个过程你能够发现自己脚本还有那些缺陷，包括逻辑缺陷和操作步骤缺陷等，完善脚本，提高脚本的健壮性。

#### 4．step-by-step，手把手的教你如何完成一个自动化测试场景。

1. 安装QTP。
2. 将SVN上的相关文件下载到本地。svn://10.1.145.249/Autotest/DCSM\_Autotest
3. 打开“DCNAutoTestFrame”文件夹，里面放着的就是自动化测试框架。浏览一遍框架存储结构，查阅本文档的第二章的1.4，了解各文件夹的作用。
4. 开始设计测试用例，用例的设计涉及到两个Excel文档。
5. 打开“DCNAutoTestFrame”文件夹下的DCSMtestCases.xls（视业务系统的不同而不同）文档，将待测试的用例信息放入表格中，填写时请认真阅读表格上方的提示信息。填写格式如下：



1. 在文件夹“testData”下找到相应的应用系统，创建一个Excel文件，名称可根据实际业务功能命名，但要与上一个文档的sheet字段相同。例如，名称起做DCSMOpenAccount.xls，与上一个文档中的table字段相同。 这个文档共有两个部分的内容，一部分是用来关联数据和Task脚本文件的，放置在文档的第一个sheet，sheet的名称起做“testCase”与上一个文档中的sheet字段相同。另一部分是用来存储自动化测试数据的，通常放在testCase制表项的后面，每一个sheet中的数据对应一个功能用例，数据书写规则：第一行放置列标题，列表不可重名，列标题将在脚本中引用数据时当做参数使用，从第二行开始放置测试数据。两部分内容的对应关系如下图：



1. 开始开发脚本，具体方法请参照本章节的第3小节“脚本开发的三字箴言”，待脚本调试通过后放入框架根目录的“testScript”下。
2. 支持所有的测试准备工作已经完毕，我们开始加载框架，运行脚本。

首先，打开QTP，Open框架根目录下的driver.vbs文件，注意这个脚本文件不准随意改动，否则测试脚本将无法正常执行。

然后，点击Run按钮，弹出对话框选择ResultReport的存储路径，我们将ResultReport存储在框架下面的“result”文件夹下，开始执行脚本。第一次运行脚本的时候最好要仔细观察脚本的运行过程，排查脚本的BUG。

最后，脚本执行完毕后将会自动弹出报告。

1. 通过测试报告分析脚本的运行情况。

## 四、大结局

虽然文档已经近尾声，但我们要清醒的认识到，自动化测试尤其是测试框架，不是一蹴而就的工作，他是一个持续改进，坚持不懈的过程。能够遇见的是，这个过程很可能是一个痛并快乐着的过程。衷心希望以上的内容不会成为废话一堆，最好能够直击你们的痛处，有任何建议请及时跟我联系，并亲切的指出其中的不足，我会虚心改正的。