2014.03

自动化测试参考

原来如此

—愿你在测试的道路上胸怀千里缜密如丝

2014.03

第一篇　软件自动化测试

第一章　自动化测试必备常识

一直以来，自动化测试是目前软件行业中比较流行和推崇的测试方式。随着自动化的快速普及，越来越多的人开始重视它，使用它。那么到底什么是自动化测试呢？如何实施自动化测试？以及需要遵从哪些规范？这些都是进行自动化测试前必须宁清的问题。先来看一个社区讨论吧，相信这些问题也曾经困扰着大家。从中我们也许可以找到坚持自动化测试道路的理由。

1.1 自动化测试必须逾越的障碍

职场中人，或多或少都经历过痛苦的职业定位选择。那些把软件测试作为职业发展的人，在定位之初一定会有着各种的扪心自问 - 为什么要选择这个方向？会不会有前途？这个上手容易吗？…

特别是初入职场的同学，相信会有这样的痛苦。从网络讨论中，我找出了一些有代表性的问题，如果你还在为这些问题而郁闷，不妨来看看。一起来梳理好这些问题，为今后的发展扫清思想上的包袱。如果你没有，请跳过这一节。

问题点：

自动化测试的发展前景怎么样？相比于开发，测试的技术含量是否偏低？测试人员提升自身竞争力的速度是否没开发快？

回答 l –

就测试和研发这两个行业的人员的平均水平来说，毋庸置疑，测试的技术含量是弱于研发的。但就行业前景来说，个人以为，测试的前景要好于研发。

理由：

- 相比于开发来说，测试行业还是个新兴的行业，成熟度也比较低，所以也就蕴含着更多的机会，如果你够努力和专注的话，成功的可能性要更大一些。
- 对一个产品来说，一个好的测试的影响力要大于一个研发工程师的影响力，

为职业分工的不同，测试覆盖的业务范围要大于开发，而且和研发一样，测试要对最终的产品“质量”承担责任，所以，可能产生的影响也会更大。

- 测试的技术含量，绝不仅仅局限于测试自动化，还包括产品评测、质量标准、质量流程、业务能力、项目管理、风险管理等。

我个人就见过很多优秀的测试工程师，他们在项目/产品中发挥的作用，远远比那些同级别的研发工程师要大[1]。

回答 2 –

一个优秀的自动测试工具需要的是对整个架构的熟悉和对整体测试的考虑，真要做好也不简单。可以这么说，自动化测试目前情况是“能用”就凑合，“好用”很难。相对而言，因为目前普遍水平还是较低的，因为只要努力一些，表现好一点应该可以很顺利脱颖而出，但要长久保持自己的竞争力依然需要更进一步进行系统的学习，而且因为这个更注重整体架构的学习，进度方面慢一点没关系。

回答 3 –

想成为一个有竞争力的测试工程师，需要比开发还懂得代码的技巧，更要比普通的测试懂得如果快捷有效的测试。自动化测试也是一种方式，一般重视测试的公司，产品变化和速度不大的使用自动化的机率大。相反如果是在一个敏捷开发不同产品的公司干测试，你想用自动化都难，需要设计和写脚本，更新和维护，相对来说没有一般的手工来得快和方便，不同的公司产品不同的岗位需求，竞争也各不相同。

我的观点是 –

从目前来看，自动化测试就像是目前 IT 行业中的大数据和云。好热，看上去很美。但真正要看出他的价值，还真不容易。通常也就是用几个工具，写几行代码而已，很少有项目从上到下都在遵从自动化的规范。也有些人热衷于从 ROI 中来找答案，我可以告诉你，如果你没有具体的衡量标准，实施细则以及项目从上到下的

配合。那 ROI 的结果只是拿来忽悠客户的幌子，什么节省 effort 呀，什么减少测试人员的压力啊。神马的神马。都不过是为了虚张声势而已。呵呵，悲观了不是？

但如果你将测试作为长期职业来经营。那么了解和熟悉自动化测试是一个必不可少的过程。我虽然不赞成将自动化往高大上的层面靠，但缺少了对它的了解。无论你在测试行业做到什么职位都将是一种遗憾，这种遗憾也必将会给你带来惨痛的损失。所以，既然选择了这行，不防用心来了解下它。选择测试没有错，走不会自动化测试的职业发展之路就是大错特错。

那么测试是什么呢？我们这个回答或许能代表相当大一部分的观点：

- Testing is an extremely creative and intellectually challenging task.
- It is the process of executing a program with the intent of finding errors.
- A good test case is one that has a high probability of detecting an as-yet undiscovered error.
- A successful test case is one that detects an as-yet undiscovered error.
- Your ideas for enhancements are just as important in this test effort as finding errors.

1.2 什么时候需要自动化测试

这个问题几乎每个学习自动化的人都会遇到，答案也是各不相同，但问题本质上离不开这几个因素：

1.> Test Times – if you often required to do regression test
2.> Test Environment – if the test environment is stable
3.> Reusable Repeat Operation – if your projects have the large of reusable repeat operation
4.> Project release frequency – if your project is a long term project and if the release is frequency
5.> Effort Measure – if the effort is measurable/calculate between manual and automation

基本上如果一个项目满足这些必要条件，那么自动化测试的优势就会比较明显，这也是衡量我们是否导入自动化测试的一个前置判断。另外，通过上面的分析，我们也很容易得出自动化测试的优缺点 –

The merit of using Automation Test

1.> It's more convenient for regression test

2.> It's more quickly and efficiently to run the complex cases

3.> It can take advantage of the resources well and make the people pay more attention to the jobs that more needs manual focus on

4.> Reduce the mistakes of human caused

5.> More accurate to simulate the manual test

6.> Save effort

The demerit of Automation

1.> Automation 不能完全代替 Manual

2.> Manual test 必定会比自动化测试发现更多的 bug

3.> 自动化测试对环境和相关前置条件的依赖比较大

4.> 自动化测试没有 Manual 测试的想象空间大

5.> 不要奢望自动化测试能帮助你发现所有的问题

1.3 自动化测试流程

什么是自动化的测试流程？这也是每个学习自动化测试的人员必须面对和回答的问题， 做任何事情都必须遵循一定的准则。 自动化测试流程的作用就在如此。它是指导自动化测试的总体原则和自动化工作展开的参考依据。那么什么是自动化测试流程呢？简言之，如下图（图 1-1 自动化测试示意图）

图 1-1 自动化测试示意图

上图只是一个形象的示意图，可解析为更为具体和详尽的流程图 –

---------- Automation Work Flow and Role Responsibility----------

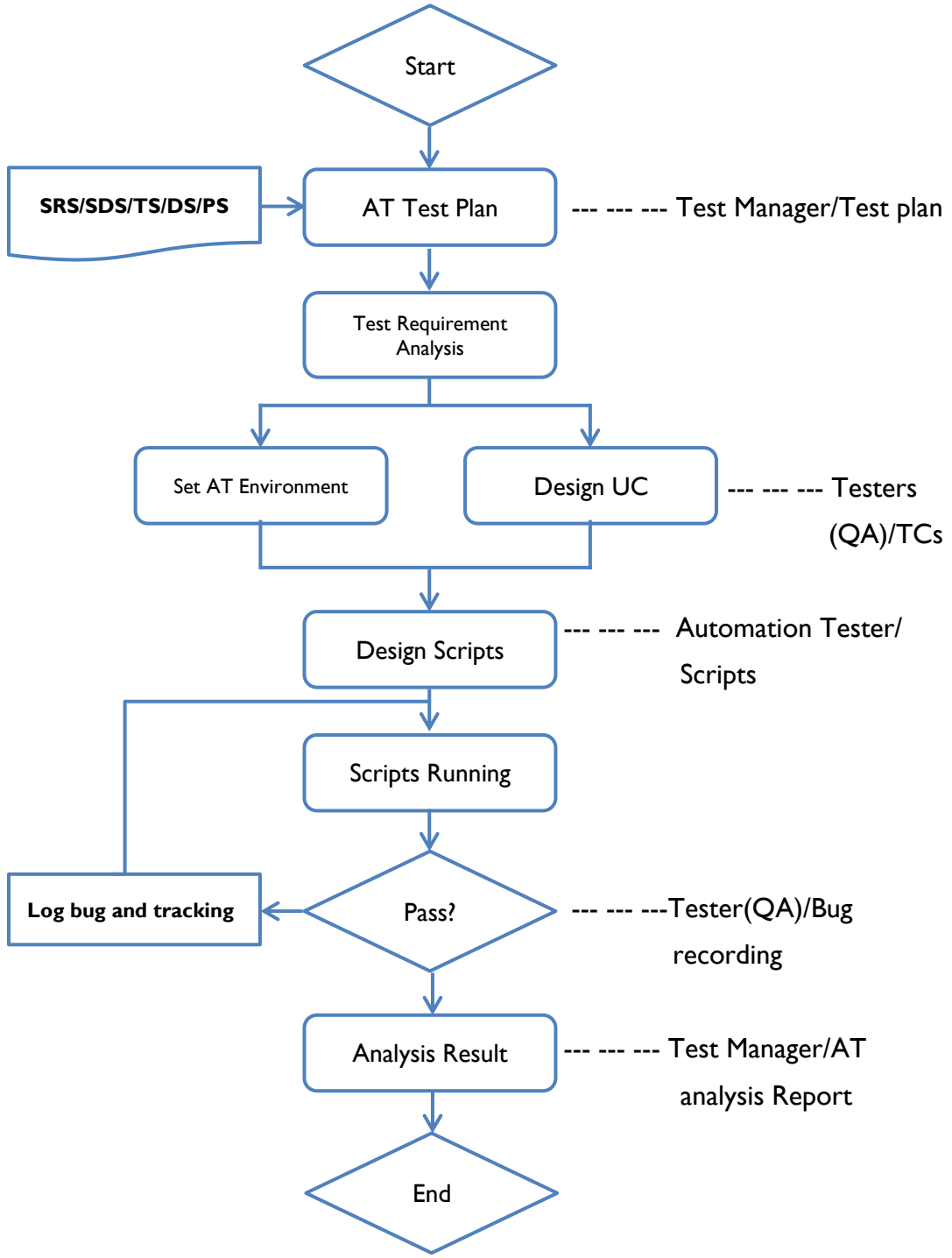

图 1-2 自动化测试流程

1.4 脚本设计方法的选择

上面叙述中简要地说明了自动化过程中几个重要的概念和不得不面对的问题，理清了这些基本问题后，接下来我们要做的是如何设计自动化脚本了，自动化测试工具有很多，如用于性能测试的 Loadrunner， Robot，Silk Performer 等；用于功能测试的如 QTP，Winrunner，Robot 和 Silk Test 等。其中尤以 Loadrunner 和 QTP 最为流行，这里我将以这两个工具为基础平台来展开后续的描述。

说到自动化测试，可能有人已经想到了录制回放，没错，初学最常用的方法就是这个。“录制回放”的方法是最直观最简单的，但也很脆弱，它的开发成本虽然较低，但维护成本很高；那些使用先进的关键字驱动测试的方法，虽然开发成本偏高，然后维护成本则偏低。测试经理需要在这些方法中作出明智的选择，以便更好地平衡开发难度和总体成本。

从脚本设计方式来分，编写脚本的方法大致可分为以下几种：

1.> 线性的
2.> 结构化的
3.> 共享的
4.> 数据驱动的
5.> 关键字驱动的

1.4.1 线性脚本编写方法

线性脚本编写方法是使用简单的录制回放的方法，测试工程师使用这种方法来自动化本地测试系统的流程或某些系统测试用例。它可能包含某些多余的、有时候并不需要的函数脚本。

优缺点：

1. 是一种非结构化的编程方式
2. 测试用例由脚本定义
3. 非常低的开发成本
4. 测试人员所需要的编程方面的技巧几乎可以忽略

5. 不需要前期的脚本计划和设计
6. 测试数据在脚本中是硬编码的
7. 脚本会很脆弱，因此维护成本会很高
8. 没有公用的脚本，因此可能造成重复劳动

1.4.2 结构化脚本编写方法

结构化脚本编写方法在脚本中使用结构化控制。结构控制让测试员可以控制测试脚本或测试用例的流程。在脚本中，典型的结构控制是使用 “if-else”， “switch”，“for”，“while” 等条件状态语句来帮助实现判定、实现某些循环任务、调用其它覆盖普遍的功能函数。

优缺点：

1. 是结构化的脚本编写方法
2. 测试用例在脚本中定义
3. 编程的成本要比线性脚本编写方法略为高一点
4. 需要测试员的调整编码技巧
5. 需要某种程度上的预先规划计划
6. 测试数据也是在脚本中被硬编码的
7. 因为相对稳定一点，所以需要相对少的脚本维护，维护成本比线性脚本编写方法要低一些
8. 除了编程知识外，还需要一些脚本语言的知识

1.4.3 共享脚本编写方法

共享脚本编写方法是把代表应用程序行为的脚本在其它脚本之间共享。意味着把被测应用程序的公共的、普遍的功能点的测试脚本独立出来，其它脚本对其进行调用。这使得某些脚本按照普遍功能划分来标准化、组件化。这种脚本甚至也可以使用在被测系统之外的其它软件应用系统。

优缺点：

1. 脚本是结构化的

2. 测试用例在脚本中定义
3. 开发成本相对于结构化脚本编写方法来说要降低一些，因为减少了很多重复的劳动
4. 需要测试员有调整代码的编程技巧
5. 由于脚本需要模块化，所以需要更多的计划和设计
6. 测试数据也是硬编码的
7. 脚本维护成本要比线性脚本编写方法的低

1.4.4 数据驱动脚本编写方法

这种方法把数据从脚本分离出去，存储在外部的文件中。这样脚本就只是包含编程代码了。这在测试运行时要改变数据的情况下是很有效率的。这样脚本在测试数据改变时也不需要修改代码。有时候，测试的期待结果值也可以跟测试输入数据一起存储在数据文件中。

优缺点：

1. 脚本是以结构化的方式编程的
2. 测试用例由测试数据或脚本定义
3. 由于脚本参数化和编程成本，这种方法的开发成本跟共享脚本编写方法比较起来要相对高一些
4. 需要测试员需要有较高的代码调整方面的编程技巧
5. 需要更多的计划和设计
6. 数据独立存储在数据表或外部文件
7. 脚本维护成本较低
8. 推荐在需要测试正反数据或者不同版本数据的时候使用

1.4.5 关键字驱动脚本编写方法

这种方法把检查点和执行操作的控制都维护在外部数据文件。因此测试数据和测试的操作序列控制都是在外部文件中设计好的，除了常规的脚本外，还需要额外的库来翻译数据。是数据驱动测试方法的扩展。

优缺点：

1. 综合了数据驱动脚本编写方法、共享脚本编写方法、结构化脚本编写方法
2. 测试用例由数据定义
3. 开发成本高，因为需要更多的计划，设计和开发方面的投入
4. 要求测试人员有很强的编程能力
5. 最初的计划和设计、管理成本会比较高
6. 数据在外部文件存储
7. 维护成本比较低
8. 需要额外的框架或库，因此测试员需要更多的编程技巧

比较下来：

1. 关于开发的成本
   随着脚本编写方法从线性到关键字驱动的改变，开发的成本不断地增加。

2. 关于维护的成本
   随着脚本编写方法从线性到关键字驱动的改变，自动化测试的维护的成本在不断的降低。

3. 关于编程技能要求
   随着脚本编写方法从线性到关键字驱动的改变，对一个测试员的编程熟练程度的要求在增加。

4. 关于设计和管理的需要
   随着脚本编写方法从线性到关键字驱动的改变，设计和管理自动化测试项目的要求在增加。

综上所述，脚本的开发方式有很多，具体采取哪一种方式，需要结合项目本身特点和个人的偏好。总之从节约 effort 的角度出发，选取最合适自己的方法。

1.5 自动化脚本框架（Framework）

自动化脚本框架没有一个统一的标准，在我看来，既然是框架那就是要做到尽可能少的人工干预。完整的测试报告生成和良好的移植性。有些人喜欢把什么东西

都朝框架里塞，为框架添加很多功能，这样虽然可以减少代码的开发，但也增加了维护框架的额外成本。

在个人多年的实践中，测试框架本身没有优劣之分，一个良好的测试框架，其主要目的还是为了提高产品的测试效率，减少测试周期和维护脚本的成本，为产品的发布提供可靠的质量保证和赢得时间。从这个角度上说，我个人比较赞成的做法是：测试工程师在一开始的时候使用自己的一套框架，等大家都熟悉起来以后，再把这套框架进行升级和补充（遵行减少人为干预和完整报告生成的原则），然后再移植到统一的服务器上去。而不是在项目一开始就设计一个高大全的框架，这种画饼充饥的做法常常会适得其反，耽误项目的测试进度，呵呵。。。不是吗？

或者在项目开始的时候找一些有经验的人一起讨论，确定大致的步骤，大家照此步骤各自展开，然后定期更新和完善。最终找到合适的框架模式。下面是我常用的测试框架，比较简单但便于脚本管理和执行，也方便移植。在后续的介绍中，我将以此框架为基础来组织我的脚本和用例。

1. **AT Framework Structure –**
   - Script folder
   - Input File
   - Output File
   - Report
   - Object Repository
   - Function Library
   - ScreenShot
   - ReadMe

下图为我的一个真实脚本的组织框架

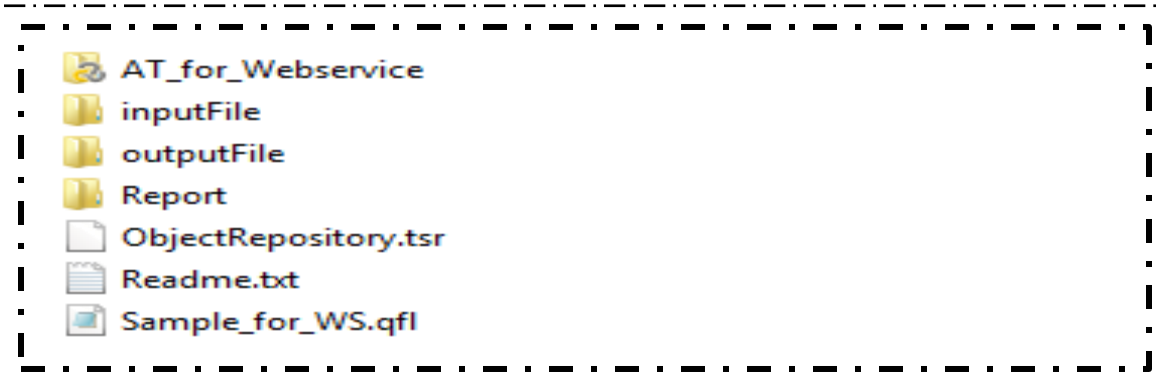

图 I-I 简单的脚本框架

从目录分类来看，这个框架能很好的解释结构化脚本的设计思想。如果你所在的项目组没有成熟的，普遍的脚本框架，可以考虑使用这样的结构来组织你的脚本。到此位置我已介绍完了本章的内容。在这一章里，我们讨论了几个现实问题，帮助那些选择做软件测试的同学理顺了思路。也讲述了实施自动化测试的几种方法，还提到了自动化的测试框架，别小看这些基础知识。它是成就你职业蓝图的重要一环。

第二章　常用测试用例的自动化实现

在这一章里，我将重点介绍一些常用测试用例的自动化实现。为什么要做这样的安排，因为我觉得对于自动化人员来说，大多数时候，一些有用的案例参考比起长篇大论的理论讲述，更容易让人接受，也最省测试人员的时间，对脚本的编写效率当然效果也是最直接的。哪些用例是最有代表性的呢？这里我不放从如下几个方面来展开，比喻基于 Web Services 的自动化实现，基于。。。。。

2.1 案例介绍

2.1.1 基本 Web Services 的测试用例自动化实现

在这个案例中我将讲述如何实现基于 Web Services 的用例转换。这是因为 Web Service 不仅是日常测试中常见的一类 Case。而且还因为这类 Case 广泛地运用了 Web 的通用技术，如 HTML，XML，SOAP，WSDL 等。这些都是软件行业耳目能详的名词。熟悉和掌握这些技术。对软件测试非常重要。先来看下这类 Case 的测试步骤吧。

1）测试步骤 （Steps Statement）–

1. Get the original test info *WSDL* from requirement
2. Launch this WSDL with initialization
3. Send request by SOAP tools
4. Get the response from SOAP
5. Check the return back status

上面是测试 Web Services 常用的测试步骤，根据这些步骤，如何展开脚本的编写呢？参考第一章介绍的自动化测试框架结构，和脚本编写方法，我们采取如下的计划来设计这个脚本。首先是 Framework 的选择，结构如下。

2）测试框架（Framework） –

- Script Name
- Input file

- Output file
- Screenshot
- Function Library
- Report folder
- Read me

其次是测试数据的准备，首先需要获得 Web Services 的 WSDL 参数，它告知从何处获得 Web Services 的文件- （XML）。然后就是初始化参数，确定哪些字段需要赋值，它是 Send Request 的前置条件。有了这些我们才可以进行 Web Services 的操作。这里我准备了一个 Web Services 文件 - getWebServicesFile.xml，格式如下。

3）测试数据（Data Preparation）–

```
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:web="http://webservice.businessRelationship.mdm.it.hp.com/">
  <soapenv:Header />
      - <soapenv:Body>
       -<web:createUpdateCompetitor>
         - <!-- Optional:  -->
          <actionCode>?</actionCode>
         - <!-- Zero or more repetitions: -->
         - <businessAreaGroups>
            - <!-- Optional: -->
          <businessAreaGroupId>?</businessAreaGroupId>
            - <!-- Optional: -->
          <businessAreaGroupName>?</businessAreaGroupName>
         </businessAreaGroups>
         - <!-- Optional: -->
        <businessRelationshipId>?</businessRelationshipId>
         - <!-- Optional: -->
        <organizationId>?</organizationId>
```

```
      - <!-- Optional: -->
      <panHPIndicator>?</panHPIndicator>
      - <!-- Zero or more repetitions: -->
     -<worldReginGroups>
      - <!-- Optional: -->
      <worldRegionId>?</worldRegionId>
      - <!-- Optional: -->
      <worldRegionName>?</worldRegionName>
     </worldReginGroups>
      - <!-- Optional: -->
     <worldwideIndicator>?</worldwideIndicator>
  </web:createUpdateCompetitor>
  </soapenv:Body>
  </soapenv:Envelope>
```

4）脚本实现（Script Implement）-

准备好了数据（即上面提到的 Web Service 文件），接下来就是脚本的编写啦，下面的函数介绍将帮助实现自动化脚本转换的整个过程。我将逐一介绍每个函数的功能和用法，以及编写过程中哪些需要注意的地方。

函数 **Get_InitialXMLPath(scriptName)** 此方法主要用于获取测试路径。脚本中的输出文件（主要指 Output file 和 Report folder 中的文件）将以此路径为基础来设置各自的绝对路径。

```
Function Get_InitialXMLPath(scriptName)
    Dim initialXMLPath, getScriptPath
    getScriptPath = Environment.Value("TestDir")
    initialXMLPath = Split(getScriptPath, scriptName)(0)
    Get_InitialXMLPath = initialXMLPath
End Function
```

点评：上面函数用到了两个常用的方法 Split 和 Environment。他们常常用于分割字符串以提取变量值的操作上。另外，需要注意一点的是，给函数名赋值，这个相信大家用的多。在 VB 中，一般说来，函数的定义有两种 Function 和 Subroutine。我们可以给函数名赋值，但不能给子函数（Sub）和过程事件（Event）做赋值操作。

函数 **InitializeXml_And_SaveXml_To_InputFile(initialXmlPath, initialXmlName, initialExcelFilePath, arrayVar, publicPath)** 主要用于初始化 Web Services 文件，此文件将是 Web Services Send 操作的 input 参数。

```
Function InitializeXml_And_SaveXml_To_InputFile(initialXmlPath, initialXmlName,
initialExcelFilePath, arrayVar, publicPath)
    Dim xmlDoc, getParameter
    Set xmlDoc = XMLUtil.CreateXML  ' load initial XML file for initialization
    xmlDoc.LoadFile initialXmlPath
    DataTable.ImportSheet initialExcelFilePath, 1, 2  'import initial Xls file for elements
     initialization
    DataTable.SetCurrentRow(arrayVar(Ubound(arrayVar)))    ' Set current Row to the
     specified row that is the last one of ARRAY
    For i = 0 To Ubound(arrayVar) - 1      ' initialized initial xml file
        getParameter = DataTable.Value(i+1, 2)
        If getParameter <> "" Then
            xmlDoc.GetRootElement.ChildElementsByPath("//" & arrayVar(i)).Item(1).&_
            SetValue(getParameter)
            else
            xmlDoc.GetRootElement.ChildElementsByPath("//" & arrayVar(i)).Item(1).Clear
        End If
    Next
    xmlDoc.SaveFile publicPath & "inputfile\" & Split(initialXmlName, ".XML")(0) &_
```

```
    "_initialized.xml"     'save initilizated xml file to output folder
    InitializeXml_And_SaveXml_To_InputFile = publicPath & "inputfile\" & Split(initialXml
    Name, ".XML")(0) & "_initialized.xml"
    Set xmlDoc = Nothing
End Function
```

点评：此函数引用了一个 XMLUti 对象，并调用了此对象中一个重要的方法 - 'ChildElementsByPath' 该方法可以提取 XML 文件中各个指定的元素值，这是非常重要的应用，它是实现 XML 文件自动化处理的关键所在。另外，对 XML 各个元素的初始化，这里还调用了 DataTable 对象，通过 DataTable 对 excel 的操作，可以将事先录入 excel 文件的初始值自动地赋给相应的 XML 元素。实现初始化元素的自动化操作。完成赋值后，将此文件保存至 Framework 中的 input file 路径下，供后续 Web Service 的 Send request 使用。

函数 **SendRequest_And_GetResponse(webServiceURL, contentType, soapAction, initializedXmlFile, publicPath, initialXmlName)** 将实现 WebServices 的两大功能 Send Request 和 Get Response。

```
Function SendRequest_And_GetResponse(webServiceURL, contentType, soapAction,
initializedXmlFile, publicPath, initialXmlName)
    Dim objWinHttp, objFSO, getRequestXmlString, getResponse
    Set objWinHttp = CreateObject("WinHttp.WinHttpRequest.5.1")
    objWinHttp.Open "POST", webServiceURL, False ' Open Http Connection
    objWinHttp.SetRequestHeader "Content-Type", contentType ' set the request header
    objWinHttp.SetRequestHeader "SOAPAction", soapAction ' set the request header


    Set objFSO = CreateObject("Scripting.FileSystemObject")
    getRequestXmlString = objFSO.OpenTextFile(initializedXmlFile, 1, False).ReadAll ' get
    initialized XML String for request
    objWinHttp.Send getRequestXmlString ' send request with XLM string NOT
```

```
    FileName
    getResponse = objWinHttp.ResponseText ' WinHttp doesn't has the save function. So i
    will create XML object for saving response file

    Set xmlDoc = XMLUtil.CreateXML ' create xml object for saving resposne xml as a file
    xmlDoc.Load getResponse
    xmlDoc.SaveFile publicPath & "outputFile\" & Split(initialXmlName, ".XML")(0) & _
    "_Response.xml"
    SendRequest_And_GetResponse = publicPath & "outputFile\" &_
    Split(initialXmlName  ".XML")(0) & "_Response.xml" ' Function Name will return the
    response file
    Set xmlDoc = Nothing
    Set objFSO = Nothing
    Set objWinHttp = Nothing
End Function
```

点评： 此函数的精髓在于引用了 “WinHttp.WinHttpRequest.5.1” 对象，此对象是 MSXML 的底层对象，调用此对象的 setRequestHeader 和 send 方法，可以模拟 Http 协议操作，给其他 Http 服务器发送请求。这正好给模拟 Web Services 操作提供了可能。有关 WinHttp.WinHttpRequest.5.1 对象的其他方法和属性，可以参考 MSDN 说明。另外此函数还调用了 FSO 对象，此对象在自动化测试中应用比较频繁，它提供了对各类文件的读写操作。值得注意的是 SoapAction 和 ContentType 两个参数值的获取，这可以方便地通过 SoapUI 来提取（SoapAction = WebURL+ WebPortName），参数 webServiceURL 通常是发送请求的链接地址，通常是 WSDL 协议地址，具体是不是还得借助 SoapUI 这类工具来帮助分析并提取。

接下来要做的是自定义一个测试 Report, 方便 review 自动化测试结果，为什么要做这个？因为 Web Services 的测试，通常是没有 GUI 界面的操作。单纯的靠 QTP/UFT 自动生成的 Report，可读性不强，他只有一个最终结果是 Pass 或是 Fail 不能直观的反映每一步的执行状况。自定义一个 Report 可增强测试的可读性。

函数 **CheckReportFolder_And_DefineReportHeader(publicPath)** 将实现 Report 的定义和初始化

```
Function CheckReportFolder_And_DefineReportHeader(publicPath)
    Dim reportColumn, x
    reportColumn = Array("", "Script_Name", "Send_Request", "Get_Response", "Return
     _Message", "Final_Test_Status")

    ' Create report file in the specified path
    Set objFSO = CreateObject("Scripting.FileSystemObject")
    Set objExcel = CreateObject("Excel.Application")

    ' Define the Report file and initialize report columns
    If Not objFSO.FileExists(publicPath & "Report\Report.xlsx") Then
        objExcel.Workbooks.Add
        objExcel.ActiveWorkbook.SaveAs publicPath & "Report\Report.xlsx"
        objExcel.Workbooks.Open publicPath & "Report\Report.xlsx"

        For x = 1 To Ubound(reportColumn)
            objExcel.Sheets.Item(1).Cells(1,x) = reportColumn(x)
        Next
        objExcel.ActiveWorkbook.Save
    End If

    'Release customized objects
    Set objFSO = Nothing
    objExcel.Quit
    Set objExcel = Nothing
End Function
```

点评：此函数主要调用了 FSO 的一些基本方法。并结合 Excel 对象帮助初始化自定义的 Report. 在这个自定义的 Report 里，将简要的列出“脚本的名称”、“send request”、“get Response ”、“返回的信息” 和“最终结果状态”等五个字段。方便用户阅读测试执行的过程。有兴趣的童鞋，可以试着用 QTP/UFT 自带的 DataTable 对象来定义 Report 格式。效果应该是一样的。

函数 **ValidateResponseFile_And_GetReport(publicPath, scriptName, sendRequestPath, getResponsePath)** 对自定义的 Report 进行赋值并判断最终状态结果

```
Function ValidateResponseFile_And_GetReport(publicPath, scriptName,
sendRequestPath, getResponsePath)
    Dim objFSO, objExcel, xmlDoc, rowPoint, getResultMessage, y
    'Customize Report form
    Set objFSO = CreateObject("Scripting.FileSystemObject")
    Set objExcel = CreateObject("Excel.Application")
    Set xmlDoc = XMLUtil.CreateXML

    objExcel.Workbooks.Open publicPath & "Report\Report.xlsx"
    rowPoint = objExcel.Sheets.Item(1).UsedRange.Rows.Coun
    xmlDoc.LoadFile getResponsePath
    getResultMessage = xmlDoc.GetRootElement.ChildElementsByPath("//description").Ite
m(1).Value ' return the element's value

 For y = 1 To 5
      Select Case y
         Case 1
         objExcel.Sheets.Item(1).Cells(rowPoint + 1, y) = scriptName
         Case 2
         objExcel.Sheets.Item(1).Cells(rowPoint + 1, y) = sendRequestPath
```

```
            Case 3
            objExcel.Sheets.Item(1).Cells(rowPoint + 1, y) = getResponsePath
            Case 4
            objExcel.Sheets.Item(1).Cells(rowPoint + 1, y) = getResultMessage
            Case 5
            objExcel.Sheets.Item(1).Cells(rowPoint + 1, y) = "PASS"
        End Select
    Next
    objExcel.ActiveWorkbook.Save
    objExcel.Quit
    Set objExcel = Nothing
    Set objFSO = Nothing
End Function
```

点评： 在定义好 Report 的格式后，接下来要做的就是给各个字段赋值了。对于上面定义的格式。Send request 和 get response 其实只要从 Framework 中的 input file 和 output 拿到它的 Path 即可，返回的信息也比较简单。通过 Xpath 方法也可以从 response 中提取。参考上面提到的“ChildElementsByPath” 方法。需要主要的是最后一个字段的赋值。他需要判断，并根据判断结果给出最终值。 一起来看看下面函数是如何实现的。

最后一个函数就是帮助美化 Report, 使其更加美观。主要使用 Excel 对象的基本方法和属性。这里不多解释了，有兴趣大家可以多试试看。这也是为什么没有使用 Datatable 对象的原因。Excel 对象包含了更多的属性和方法，可以实现如字体及颜色的定义。

函数 **FormatReport(reportPath)** 将测试 Report 进一步进行格式定义。如，Failed – Red font、Passed – Greenfont、Warning or Others – Yellow font

```
Function  FormatReport(reportPath)
    Dim excelApp, getAvailableRow, inRow
```

```
    Set excelApp = CreateObject("Excel.Application")
    excelApp.Workbooks.Open reportPath
    excelApp.Sheets.Item(1).Range("A1:E1").Interior.ColorIndex = 32 'define color for the
     specified range
    'excelApp.Sheets.Item(1).Range("A1:E1").Font.Color = vbBlack
    excelApp.Sheets.Item(1).Range("A1:E1").Font.Size = 12
    excelApp.Sheets.Item(1).Range("A1:E1").Font.Bold = True ' any value here for the bold
     is okay

    getAvailableRow = excelApp.Sheets.Item(1).UsedRange.Rows.Count
    'excelApp.Sheets.Item(1).Range("E2:E"&getAvailableRow - 1 &"""").Interior.ColorIndex = 46

    For inRow = 2 To getAvailableRow
       If excelApp.Sheets.Item(1).Cells(inRow, 5).value = "PASS" Then 'Get the cell value
          excelApp.Sheets.Item(1).Cells(inRow, 5).Interior.ColorIndex = 10
       ElseIf excelApp.Sheets.Item(1).Cells(inRow, 5).value = "FAIL" Then
          excelApp.Sheets.Item(1).Cells(inRow, 5).Interior.ColorIndex = 3
       Else
          excelApp.Sheets.Item(1).Cells(inRow, 5).Interior.ColorIndex = 6
       End If
    Next
    excelApp.ActiveWorkbook.Save
    excelApp.Quit
    Set excelApp = nothing
End Function
```

至此，有关 Web Services 的 Case 自动化已介绍完毕。看起来很多，其实重要的就是几个对象的方法运用而已（如 WinHttp.WinHttpRequest.5.1, Xpath, FSO 和 Excel）。 上面的函数介绍仅供实际编写脚本时的参考，不能完全照搬。那些代码只在指定环境下是可行的。

2.1.2 基于 WebSite 的测试用例自动化实现

在这一案例中，我将着重介绍有关 Web Site 的一些常用转换操作。说起 Web Site 大家都很太熟悉，几乎每个做测试的人都接触过，它是软件测试中最常用的一类 Cases。Web Site 包含的操作十分丰富，常见的有登陆，检索，界面检查，后台访问与确认等。下面我将找一些实例，尽可能地将这些操作涵盖进来。

跟上面的脚本框架一样，这里我同样采用（图 1-1）的 Framework, 先来看一个实例，在这个实例里，我们需要对一个 Web Site 上的连接进行操作。具体步骤如下：

2.1.2.1 基于 WebEdit，WebButton 和 Linkage 的操作

Step1 : Close Opened IE and Login WebSite with Account

Step2 : Navigate the specified linkage

Step3 : Download the file from the specied URL

先来看看 Step 1 如何登陆 Website，通常的做法就是关闭现有的 IE 窗口，并打开指定的 URL 窗口。这一步在 Web 测试中经常遇到。这里我梳理了几种常用的方法，希望对大家有所帮助。

方法 **Close_Current_Windows** 关闭打开的 IE 窗口。即关闭 'Windows Task Manager' 里所有的 IE 进程

```
Function Close_Current_Windows
    Set wScriptShell = CreateObject("WScript.Shell")
    wScriptShell.Exec("TaskKill.exe /FI ""imagename eq iexplor*"" /F") ' 语句 1 ： Close IE windows which already openned
    SystemUtil.Run "taskKill", "/FI ""imagename eq iexplo*""", "", "", 3 ' 语句 2 ： Close IE by taskKill executable file
    SystemUtil.CloseProcessByName "Iexplore.exe" ' 语句 3 ： Close IE by programe
```

```
    Name
    Set wScriptShell = Nothing
End Function
```

点评： 上面三条语句都能达到同样的效果， 关闭 IE 窗口的目的，是为了减少后续 IE 操作中不必要的干扰。可执行文件 TaskKill 的用法和参数可参阅 TaskKill /? 帮助。方法 'SystemUtl. Run' 中的参数可理解为：

File：执行文件的名称，可以是文件名或可执行的 Application 名称

Params：文件执行时所带的参数。 默认为空

dir：默认的被执行文件的路径

op：文件被执行的操作。默认是 Open

mode：单文件被执行时，如何显示文件。3 - 代表最大化窗口。

关闭所有打开的 IE 窗口后，接下来要做的就是打开指定的 URL 窗口。实现脚本如下：

```
Set oPenURL = CreateObject("InternetExplorer.application")
   oPenURL.Visible = True
   oPenURL.Navigate2 uRL

   Set wScriptShell = Nothing
Set oPenURL = Nothing
```

或者

```
SystemUtil.Run "IExplore.exe", URL, "", "", 3
```

这两种写法都可以，怎么样？简单吧？开启新的页面后，紧接着就是登录了。登录框通常是 Web Edit 对象，下面的函数主要是针对 Web Edit 所做的操作，请参考。

方法 **LoginURL(uRL, userName, passWord)** 登录指定的 URL，并验证登录结果

```
Function LoginURL(uRL, userName, passWord)
    Browser("title:=.*").page("title:=.*").WebEdit("name:=userName", "html_
     tag:=INPUT").Set userName
    Browser("title:=.*").page("title:=.*").WebEdit("name:=password", "html_
    tag:=INPUT").Set passWord
    Wait(5)
    Browser("title:=.*").page("title:=.*").WebButton("name:=Login", "html_
    tag:=BUTTON").Click

    If Browser("title:=.*").page("title:=.*").Link("html tag:=A", "name:=Log_
    Out").Exist Then
        Reporter.ReportEvent micPass, "Login the URL: " & uRL, "it has been Login_
    Successfully"
        Else
        Reporter.ReportEvent micFail, "Login the URL: " & uRL, "it hasn't been logged in_
    yet"
    End If
End Function
```

点评：这个函数主要是针对 Web Edit 对象的操作，Set 是其常用的用法。另外，值得注意的是，描述性语言编程的用法，大家都知道，描述性语言编程有两种写法，这里只是其中一种，另一种，我将在后续的脚本中介绍。

下面的函数将 Active 一个连接，并 download 一个文件。

方法 **Check_And_Save_TemplateFile(tempFileName, tempFileName1, newTemplateFile)** 从指定的 linkage download 一个文件，并保存文件

```
Function Check_And_Save_TemplateFile(tempFileName, tempFileName1,
newTemplateFile)
```

```
Set temFSO = CreateObject("Scripting.FileSystemObject")
if temFSO.FileExists(rootPath & "Global_TD\" & tempFileName) Then
   temFSO.DeleteFile(rootPath & "Global_TD\" & tempFileName)
ElseIf temFSO.FileExists(rootPath & "Global_TD\" & tempFileName1)  Then
   temFSO.DeleteFile(rootPath & "Global_TD\" & tempFileName1)
End If

Browser("Address Doctor Batch Service").Page("Address Doctor Batch"&_
"Service").WebButton("Download Template").Click
Wait(5)

if Browser("Address Doctor Batch Service").WinButton("To help protect your"&_
   "security").Exist(0) Then
   Browser("Address Doctor Batch Service").WinButton("To help protect your"&_
   "security").Click
   Set msShell = CreateObject("Wscript.shell")
   msShell.SendKeys "{Down}"
   msShell.SendKeys "{Enter}"
   Browser("Address Doctor Batch Service").Dialog("Windows Internet"&_
   "Explorer").WinButton("Cancel").Click
   Browser("Address Doctor Batch Service").Page("Address Doctor Batch"&_
   "Service").WebButton("Download Template").Click
End If

Do
   Browser("Address Doctor Batch Service").Dialog("File"&_
   "Download").WinButton("Save").Click
Loop until Not Browser("Address Doctor Batch Service").Dialog("File"&_
"Download").WinButton("Save").Exist

Dialog("download.do from").Dialog("Save As").WinEdit("File"&_
```

```
    "name:").Set newTemplateFile
    Dialog("download.do from").Dialog("Save As").WinButton("Save").Click
End Function
```

点评： 此函数的最引人注意的是对象 'Wscript.shel' 的使用，当只能键盘操作时，可以调用此对象来模拟用户的行为。相似的用法，下面还会使用到，请留意。

上面介绍的所有方法，也只是参考而已，不具有通用性，但每个方法都有其独特的对象引用，这些对象的常用属性和方法，才是用户需要重点理会的。

2.1.2.2 基于 WebEdit，WebButton 和 Linkage 的操作

Step 1: Loading external test data and function library
Step 2: Close Opened IE and Open New IE with Navigation the specified linkage
Step 3: Login WebSite with Account
Step 4: Call Functions to do Create/Update/Inactive/Reactivate
Step 5: LogOut and Closed IE.

本案例也是基于 Web 自动化测试，所以在测试的步骤上相差不是很远，当然，测试步骤的差异也离不开测试框架的设计。测试框架就决定了测试的步骤。
Okay，首先我们跟着测试的步骤走，我们一起来看一下 Step one。

```
Dim DataPath
DataPath =RootPath &"\TestData\Global_Data.xml"
Environment.LoadFromFile(DataPath)

DataPath =RootPath &"\TestData\TestData.xls"
DataTable.ImportSheet(DataPath, "NationalIdentifiType",2)
```

点评： 这个方法是灵活引用外部文件，把文档的相对路径引用到脚本，用一个总的路径变量控制，如果脚本路径改变，不用改变每个脚本，只要把全局变量 RootPath 改一下，所有的脚本就可以正常使用。另外，本方法应用的 UFT 的两个主要的对象的方法，Environment 和 DataTable。

Step 2 and 3：Web 测试通用的起始点，传入指定页面的参数以及登录用户和密码 去登录到指定系统，前面有做过详细的介绍，这里就不赘述。把应用的代码贴与此处，仅供参考。

方法 **Login(strURL,strEmail,strPassword)** 传入指定的参数登录，此登录方法内部调用了一个 ***OpenBrowser(strURL)*** 的方法 去打开指定的登录页面

```
Function Login(strURL,strEmail,strPassword)
   Call OpenBrowser(strURL)

   wait 6
   If Window("Windows").Dialog("Security Warning").Exist(0) then
      Window("Windows").Dialog("Security Warning").WinButton("Yes").Click
   End if

   wait (15)
   Do While true
      If  Browser("HP Employee Portal").Page("HP Employee”&_
         “Portal").WebEdit("Email").Exist(0) Then
         Browser("HP Employee Portal").Page("HP Employee”&_
         “Portal").WebEdit("Email").Set strEmail
         Browser("HP Employee Portal").Page("HP Employee”&_
         “Portal").WebEdit("PassWord").Set strPassword
         Browser("HP Employee Portal").Page("HP Employee_
         “Portal").Image("BtnLogin").Click
         Exit Do
      End If
   Loop
   wait (15)
```

```
    If Browser("CMU").Page("Home").Link("Log Off").Exist(0) Then
        Call WriterReporter(1, "Login to CMU", "Login CMU portal successfully.")
    Else
        Call WriterReporter(0, "Login to CMU", "Login CMU portal Failed.")
        'ExitAction
    End If
End Function
```

Step 4：这里的测试步骤跟具体测试项目的不同而不同，具体的实现就不赘述，把几个有用的方法跟大家分享一下。

方法 **CheckWarnMsg(strMsg)** 是检查操作后是否弹出指定的提示信息。

```
Function CheckWarnMsg(strMsg)
    Dim obj
    Set obj = Description.Create
    obj("micclass").value = "WebElement"
    obj("html id").value = "messageBar"

    Dim oRegExp
    Set oRegExp = New RegExp
    oRegExp.Global=True
    oRegExp.IgnoreCase =True
    oRegExp.Pattern = ".*" & strMsg & ".*"
    wait(3)
    Set oEle = Browser("title:=.*").Page("title:=.*").ChildObjects(obj)
    If oEle.count >0 Then
        strValue = oEle(0).GetRoProperty("innertext")
        If oRegExp.Test(strValue) Then
```

```
            CheckWarnMsg = True
        Else
            CheckWarnMsg = False
        End If
    Else
        CheckWarnMsg = False
    End If
    Set oEle = Nothing
    Set obj = Nothing
End Function
```

点评： 此方法是本项目的一个通用方法，只要参数改变一下，就可以直接调用。这里有两大亮点，第一： 引用的描述性编程，可以使用所有的对象， 通用性比较强。第二： 应用另一大功能点 - 正则表达式。正则表达式在每种编程语言中都比较常用，在这里也不例外，当然也增加了本方法的灵活性。

方法 **DisplayStyle(strId)** 这里是用来检查操指定对象是显示还是隐藏

```
Function DisplayStyle(strId)
  Dim oDesc
   set oDesc  = Description.Create
   oDesc("micclass").Value = "WebElement"
   oDesc("html id").Value = strId "'clsContent"
   Reporter.Filter = rfDisableAll

   DisplayStyle = "none"
   Set oElem = Browser("title:=.*").Page("title:=.*").ChildObjects(oDesc)
   If  oElem.count >0Then
      set ostyle = oElem(0).Object.currentstyle
```

```
    DisplayStyle = ostyle.display
  End If
  Set oDesc = Nothing
  Set oElem = Nothing
  Reporter.Filter = rfEnableAll
End Function
```

点评：本方法的一个亮点就是引用对象的 Object 属性，通用这个属性引入了 CSS 功能。由于 CSS 是对象的样式的主要限制方法，Object 的引用从而扩充 UFT 的对对象的样式的检查。

Step 5： 也是 Web 测试通用的方法，前面也有详解，就不多说，贴出代码供参考。

方法 **LogOut ()** 这个方法主要实现测试系统，然后关掉 IE

```
Function LogOut()
  If Browser("CMU").Page("Home").Link("Log Off").Exist(0) Then
    Browser("CMU").Page("Home").Link("Log Off").Click
    Call WriterReporter(1, "Log out CMU", "Log out MDCP successfully.")
  Else
    Call WriterReporter(0, "Log out CMU", "No found log off link.")
  End If
  SystemUtil.CloseProcessByName "iexplore.exe"
End Function
```

此案例的设计，通过项目两年的使用，主要有两个优点，第一，容易上手。由于设计的比较简单，抽出了通用的方法，对应新手来说，很容易上手。 第二，通用性比较强。 对应相似类型的项目都可以使用。

通过上面的两个案例介绍，我们对基于数据驱动和关键字驱动的脚本编写有了一个初步直观的认识，函数是实现测试过程的具体载体。如何划分函数以及如何编写函数不是这本书要讲的内容。但有几个概念还是不得不提一下。细心的你也许已

经发现了，上面给出的函数的参数，到底该如何赋值呢？ 一般说来，对变量的赋值有三种方式，具体用哪一种，需要根据项目要求和个人偏好来决定。

1.） 直接变量赋值法 （这种比较简单，对于数据驱动的脚本来说不太便于维护，一旦参数变换，需要进入程序更改变量的值）

2.）表单赋值法（DataTable）- 这种方法很常见，调用 DataTable 对象中的 GetSheet 和 Value 方法即可。

3.） 环境变量赋值法（Environment）- 这也是常用的一种赋值方法，我们可以在 QTP 的环境变量设置中自定义用户变量。并保存为 XML 格式，完成后就可以调用 Environment 的 Value 方法对变量进行赋值

另外，不得不提的一个概念是描述性编程（DP），在上面的脚本中已经给出了它的使用步骤和技巧，这里还是要单独拎出来讲一讲，因为这个概念太重要了，它在自动化脚本编写过程中使用频率超高。下面我就从实战的角度来阐述下这个概念。

2.1.2.3 描述性编程 - Description Programme

描述性编程即用对象的属性来表述对象本身。它有两种表现写法，一种是直接表达法，另一种是 Description 对象表达法。下面是两种写法的具体表现 ， 还是以上面介绍的 abc 对象为例 –

1） 描述性编程- Description 对象 –

```
Set abc = Description.Create()
    abc("innertext").value = objName
    abc("micclass").value = "WebElement"
    abc("html tag").value = "LABEL"
    Browser("title:=.*").Page("title:=.*").ChildObjects(abc)(1)'Get the second object. the
    first object is '0'
    Browser("title:=.*").Page("title:=.*").ChildObjects(abc)(indexNumber).highlight
    Browser("title:=.*").Page("title:=.*").ChildObjects(abc)(indexNumber).Click
Set abc = Nothing
```

这里申请了一个 abc 变量，用它来表述 WebElement 的对象。它使用了 WebElement 的属性'innertext'，'micClass'和'html tag'。这些属性通常可用 Spy 获取。另外有些人可能会说每次用 Spy 识别对象的属性都是 oaky 的，挑取这些属性用 description 对象编程后就是不能准确的识别对象。什么原因?

要么你选取的属性还不够充分，要么是挑选的属性值并不是唯一的。最好的办法是用 Count (如 abc.count)检验下，看是否能唯一识别对象。如果不是，你还得在表述的对象后面用下标明示，如语句 –

```
Browser("title:=.*").Page("title:=.*").ChildObjects(abc)(1)
```

2) 描述性编程- 直接表达法 –

看下面这条语句，就是用直接属性表达法来描述性编程的。

```
Browser("HP SiteMinder Login").Page("CMU - User Test_2").WebTable("Class
_Name:=WebTable", "html tag:=TABLE", "index:=1" ).GetCellData(i, 2)
```

两种写法都表示同一个对象。用户可根据自己的喜好选择。当表述的对象不唯一需要用下标（index Value）来标示时，请注意两种下标的写法。

另外，在这两种写法中，细心的你也许发现第一种用到了 ChileItem 方法，第二种则直接用对象名 WebTable 表述。

另外， 什么时候需要使用描述性编程呢?

当一个对象无法唯一确定时，或者说对象的属性每次都在改变（通常反映在 Index 值上，这也是为什么上面的描述性编程都会加上 Index 值来定位对象）这时，用描述性编程来定义对象是很好的选择。

当然，你也可以试试把每次变动的属性给去掉（通常在 mandatory 属性栏中）如果对象还能识别的话，你也可以直接用对象库中的对象而不需要描述性编程。

上面这个例子详细介绍了如何登陆一个 WebSite 和查找相应的对象，以及验证对象内容这一常见流程。通过这一案例，我们初步了解了一些常见对象的脚本书写规则（对象库中的对象），重点在描述性对象编程上（不在对象库中的对象）。下面我还会陆续介绍一些 WebSite 的用例和常见写法。帮助加深这类 Cases 的常用属性和方法的调用。

2.1.2.4 CheckPoint 的设计

作为测试结果的判断，加入 Checkpoint 是我们常用的方法。那么如何有效的写 Checkpoint 呢？有两种方法可以借鉴。一种是通过 QTP/UFT 的可视化视图（通常在 “Active Screen” 选项卡中）添加 Checkpoint；另一种则是直接加入 Reporter 语句来自定义检查点。前一种方法主要用于静态页面对象的检查。后一种这多用于动态页面的对象检查。那么如何写 Reporter 语句呢？

通常一个 Reporter 语句包含在一个条件判断语句中的。比喻我们要检验一个 WebElement 对象是否显示正确。一种做法是我们截取这个对象的文本字符串，然后对这个字符串进行判断，由此判断的结果来定义我们的 Reporter。如下面代码所示：这个 WebElement 对象的 innertext 属性包含字符串“Customer ID 50051250 is Created successfully”，此时我们可以截取字符 created successfully 作为验证的依据：

```
returnString = Browser("title:=.*").Page("title:=.*").WebElement("html _
id:=datatable_info").GetROProperty("innertext")

Trim(Split(returnSting, "is")(1)) = "Created successfully" Then
   Reporter.ReportEvent micPass, "Validate the create process", "'Created successfully' _
   was involved into return message, so the create is successfully"
   Else
   Reporter.ReportEvent micFail, "Validate the create process", "We can't find the word_
   of 'Created successfully' in the whole return string, so the create is failure"
End If
```

除此之外，我们还有另一种写法来表述这个 Reporter，那就是正则表达式，这也是在实际脚本编写中用得比较多的一种方法。两种方法原理差不多都是用来比较在返回的字符串中是否有我们期望的关键字。不同的是使用正则表达式可以更松散的进行比较和匹配。那么该怎么做呢，请看下面的代码：

```
Dim regExpression, matchs

Set regExpression = New RegExp ' Define a regular expression
regExpression.Global = True ' Set global applicability
regExpression.IgnoreCase = True ' Set case insensitivity
regExpression.Pattern = "Created successfully"
Set matchs = regExperssion.Test(oString) ' Search the pattern in the oString
If matchs  Then
   Reporter.ReportEvent micPass, "Validate the word 'Created successfully'", "It was_
   found in the whole return string, So, it's passed"
   Else
   Reporter.ReportEvent micFail, "Validate the word 'Created successfully'", "It's wasn't_
   “found in the whold return string, so it's failure"
End If

Set regExpression  = Nothing
```

点评： 这个案例中主要介绍了 Checkpoint 的用法。Reporter 是实现 Checkpoint 的常用对象。通过对页面对象的属性判断并结合 Reporter 对象方法是实现 Checkpoint 的常见的做法。在编写 Checkpoint 脚本过程中，借助正则表达式来实现 Reporter 的正反逻辑表达是一种普遍的做法，请务必熟悉。

另外， With 函数也能用在这里帮助实现 Checkpoint 的设计。思路就是把要验证的页面或者对象属性放入 With structure 中，对其进行批量执行。这样一来我们就可以对 with constructure 中的对象或页面进行批量验证了。

```
With Browser("MDM Data Loader").Page("MDM Data Loader").Frame("Frame") 'put
frame object into with constructure
    tRowND = .WebTable("SucDetails").RowCount
    tColumnND = .WebTable("SucDetails").ColumnCount (1)

    For cRowND = 1 to tRowND -1
    cellData_SNr = .WebTable("SucDetails").GetCellData(cRowND+1, 1)
    cellData_SttCd = .WebTable("SucDetails").GetCellData(cRowND+1, 2)
    'return = Instr(1, cellData, UploadFileNM)
    If cellData_SttCd = "S" Then
       Reporter.ReportEvent micPass, "Validate the Status Code of Sequence Number
       " & cellData_SNr, "The Status Code of Sequence Number " &cellData_SNr &"_
       Is " & cellData_SttCd & ". Passed."
    Else
       Reporter.ReportEvent micFail, "Validate the Status Code of Sequence Number_
        " &cellData_SNr,"The Status Code of Sequence Number " &cellData_SNr &" is_
        not S. Failed."
       Exit Function
    End If
    Next
  End With
```

~~2.1.2.5 基于文件传输的 Test Case 自动化实现~~

~~在实际测试中，有一类 Case 也是经常遇到的，想想会是什么样的 Case 呢？是不是有一类 Case 经常需要上传或下载文件，将初始化好的数据以文件的形式上传，然后批量执行相关的操作？对了，就是这类的 Case，自动化实现其实没什么难度，这里我就摘录一个我们曾经的用例，目的是为了让大家更好地了解这类 case 的自动化实现。实例 Case 的步骤为：~~

~~Step1: Close Current IE Window~~

Step2: Open a new linkage URL

Step3: Check Specified folder and clear the existing file

Step4: Download and save template file

Step5: Load an initialized file and upload

一起来看看如何用代码实现上面的操作：

首先是是关闭已打开的 IE 窗口，这个大家都知道该怎么写，上面的案例中也给出了好几种写法，具体用哪一个，看个人喜好吧。这里我选择用 WshShell 对象。主要是因为这个对象不常用，但他能实现 OS 与 User 之间的简单通讯，还是能给我们脚本的编写带来很多方便。熟悉和掌握这个方法很有必要。

```
Set wScriptShell = CreateObject("WScript.Shell")
wScriptShell.Exec("TaskKill.exe /FI ""imagename eq iexplor*"" /F") ' Close IE windows which already openned
Wait(5)
'
Set oPenURL = CreateObject("InternetExplorer.application")
oPenURL.Visible = True
oPenURL.Navigate2 uRL
'
Set wScriptShell = Nothing
Set oPenURL = Nothing
```

再来就是 URL 的登陆了，这个太常见了，几乎所有的页面测试都能用到，这里我就不详细介绍方法了，但有一点需要注意，就是变量的参数化，有些人用的是 DataTable，有的人喜欢用 XML，有的干脆就赋值语句。效果上并没有什么优劣，用 XML 的时候，需要先初始化好 XML file。（可以在 QTP/UFT 的 Setting→Environment→Use Define 中操作）这个在上面的案例中有提到。本例代码实现如下：

```
Browser("title:=.*").page("title:=.*").WebEdit("name:=userName", "html _
tag:=INPUT").Set userName
Browser("title:=.*").page("title:=.*").WebEdit("name:=password", "html_
tag:=INPUT").Set passWord
Wait(5)

Browser("title:=.*").page("title:=.*").WebButton("name:=Login", "html_
tag:=BUTTON").Click

If Browser("title:=.*").page("title:=.*").Link("html tag:=A", "name:=Log_
Out").Exist Then
    Reporter.ReportEvent micPass, "Login the URL: " & uRL, "it has been Login_
    Successfully"
Else
    Reporter.ReportEvent micFail, "Login the URL: " & uRL, "it hasn't been logged in_
    yet"
End If
```

（说明：这里我用了描述性对象编程 DPS，目的是为了加深对这一写法的熟悉。其实直接用对象库的对象来得更快）

接下来就是检查指定的 Folder 了，看看里面是否存在需要 Download 的文件，如果有就删除，然后就是去页面 Download 新的 template。就是这么个过程，简单吧？代码写起来也很容易，需要注意的是，有些页面会弹出一些安全对话之类的东东，有些是弹窗，有的是安全提示。怎么模拟手动操作来点选这些提示呢？借鉴下下面的写法吧：

```
Set temFSO = CreateObject("Scripting.FileSystemObject")
if temFSO.FileExists(rootPath & "Global_TD\" & tempFileName) Then
    temFSO.DeleteFile(rootPath & "Global_TD\" & tempFileName)
ElseIf temFSO.FileExists(rootPath & "Global_TD\" & tempFileName1) Then
    temFSO.DeleteFile(rootPath & "Global_TD\" & tempFileName1)
```

```
    End If
    
    Browser("Address Doctor Batch Service").Page("Address Doctor Batch
    Service").WebButton("Download Template").Click
    Wait(5)
    
    if Browser("Address Doctor Batch Service").WinButton("To help protect your
    security").Exist(0) Then
        Browser("Address Doctor Batch Service").WinButton("To help protect your
        security").Click
        Set msShell = CreateObject("Wscript.shell")
        msShell.SendKeys "{Down}"
        msShell.SendKeys "{Enter}"
        Browser("Address Doctor Batch Service").Dialog("Windows Internet
        Explorer").WinButton("Cancel").Click
        Browser("Address Doctor Batch Service").Page("Address Doctor Batch
        Service").WebButton("Download Template").Click
    End If
    
    Do
        Browser("Address Doctor Batch Service").Dialog("File
        Download").WinButton("Save").Click
    Loop until Not Browser("Address Doctor Batch Service").Dialog("File
    Download").WinButton("Save").Exist
    
Dialog("download.do from").Dialog("Save As").WinEdit("File
name:").Set newTemplateFile
Dialog("download.do from").Dialog("Save As").WinButton("Save").Click
```

~~好了，这类 Cases 的介绍就到这里了，后面的步骤大同小异，对可以用对象库实现的步骤就用对象库里的对象来实现，不能提取对象的，用 DSP 去捕捉或者调用 WshShell 的一些对象去模拟，简单吧 ☺。~~

2.1.3 基于数据库连接和操作的 Test Case 自动化实现

在实际测试中，还有一类常见的 Cases 经常需要测试，那就是有关数据库类的 Cases。是不是早就想问这个问题了？下面我们来介绍下有关数据库类型的 Cases 的常用方法和步骤。相信看过这些方法后，你不再畏惧 DB 类的 Cases 了。把这类 Cases 的自动化学好，将有助于你提高 Case 的自动化转化率，因为有关 DB 的 Case 在日常测试中比比皆是。搞定这一类 Case 的自动化，你的自动化测试将更有价值。一起来看看吧。

在介绍 DB 类 cases 的自动化之前不得不提两个对象。想象一下，通常我们手工做数据库类的 Cases 是怎么操作的。是不是可以大致分为两步：第一步登陆连接数据库，并执行 SQL 语句找到相应的记录；第二步就是验证搜索的记录，看是否匹配期望值。是的，没错，自动化 DB Case 也是要实现这两步。怎么做呢？废话少说，直接上代码吧，我会把主要的对象以及方法 highlight 出来，并以备注的形式给予说明，省得我一个一个函数去介绍。

```
Set objADODB = CreateObject("ADODB.Connection")
Set objRs = CreateObject("ADODB.Recordset")
```

2.1.3.1 用描述性的方式建立 Excel 数据库连接 –

```
objADODB.Provider = "Microsoft.Jet.OLEDB.4.0" 'Microsoft provide OLE DB Provider
objADODB.Properties ("Extended Properties").Value = "Excel 8.0;HDR=Yes;IMEX=1" '
objADODB.Open "C:\123.xls"
msgbox objADODB.State ' 1 means successful, 0 means failure
```

2.1.3.2 用 Connection.Open 的方法直接打开 Excel 数据库

```
objCon.Open = "Provider = Microsoft.ACE.OLEDB.12.0; Persist Security Info=True;"& _
"Data source = C:\DBSheet.xlsx; Extended Properties = 'Excel 12.0; HDR=yes; IMEX=2'"
msgbox objCon.State
```

2.1.3.3 用 Connection.ConnectionString 方法直接打开 Excel 数据库

```
objCon.ConnectionString = "Provider = Microsoft.ACE.OLEDB.12.0; Persist Security" &_
" Info = True; Data source = C:\DBSheet.xlsx; Extended Properties = 'Excel 12.0; " &_
HDR=yes; IMEX=2'"
objCon.Open
msgbox objCon.State
```

2.1.3.4 用 ISAM 驱动连接 Excel 数据库

```
conString = "DSN=Excel Files; DBQ=C:\DBSheet.xlsx" '此种方法需要先安装 ISAM
（Indexed Sequential Access Method）驱动，它是 ODBC 的 Jet，不然没法连接。
objCon.Open conString
msgbox objCon.State
```

注意： ADODB.Connection 对象主要用于数据库的连接；而 DODB.Recordset 主要用于存储返回的记录；另外需要介绍的是 ExtendProperties 的几个参数设置：HDR=Yes，这代表第一行是标题，不做为数据使用，如果用 HDR=NO，则表示第一行不是标题，做为数据来使用。系统默认的是 YES 参数 Excel 8.0 对于 Excel 97 以上版本都用 Excel 8.0

IMEX (IMport EXport mode)设置

IMEX 有三种模式：

0 is Export mode

1 is Import mode

2 is Linked mode (full update capabilities)

我这里特别要说明的就是 IMEX 参数了，因为不同的模式代表著不同的读写行为：

当 IMEX=0 时为“汇出模式”，这个模式开启的 Excel 档案只能用来做“写入”用途。

当 IMEX=1 时为“汇入模式”，这个模式开启的 Excel 档案只能用来做“读取”用途。

当 IMEX=2 时为“连結模式”，这个模式开启的 Excel 档案可同时支援“读取”与“写入”用途。

再来看下面的语句，打开指定的记录并将结果返回给 Recordset 集

```
strQuery = "Select * from [Sheet1$] where FirstName = 'Zhang'"
objRs.Open strQuery, objADODB, 1, 3 'I means here open forward only, 3 means here lock pessimistic (Note: number accept here only)

counter = objRs.RecordCount
msgbox counter

' Read out the first record of querying result.
msgbox objRs.Fields(0): msgbox objRs.Fields(1): msgbox objRs.Fields(2):
objRs.MoveNext

' Read out the second record of querying result.
msgbox objRs.Fields(0): msgbox objRs.Fields(1): msgbox objRs.Fields(2)

For i = 1 To counter
   If objRs.BOF or objRs.EOF Then
      msgbox "The record is out of the real scope"
      Exit for
      else
      msgbox objRs.Fields.Item(0).Value
      msgbox objRs.Fields.Item(1).Value
      msgbox objRs.Fields.Item(2).Value
      objRs.Fields.Item(2).Value = "Changed" ' 只有在 IMEX 设为 1 或者 2 时才可以写操作
      msgbox objRs.Fields.Item(2) 'Should show the 'Changed' when set the IMEX = "1" or "2" only
      objRs.MoveNext
```

```
    End If
Next
```

通常使用 ADODB 对象就是为了连接数据库和读取数据。至于后续对读取记录的比较和输出，可以用比较语句、条件语句实现。这里不一 一列举了。

补充说明，上面的例子是将 Excel 表单作为数据库对象来进行操作的，对 Excel 的连接是通过 Windows 引擎来实现的，也就是 ADODB.Connection 的属性设置。注意两点，在以后的使用中可以少走很多弯路。 引擎"Microsoft.Jet.OLEDB.4.0" 和 "Microsoft.ACE.OLEDB.12.0" 分别对应 Excel 的不同版本或 Windows 系统版本。在使用 ACE 引擎时（Win7 已经不带 Jet 引擎了，XP 里包含了 Jet 引擎），不但可以对 excel 所有的版本进行操作，而且还可以对后缀为.xlsx 的 excel 文件进行操作；这是两种引擎最主要的区别。当然两种引擎都支持在 Excel 文件打开的模式下进行操作。

在操作 ADODB 对象时，最常出现的异常是，系统往往报 ODBC Jet、ACE 安装错误，这主要是由于 Access DataBaseEngine 版本所引起的。可以上 Microsoft 的官网下载更低的版本调试。（如，AccessDataEngine_x64 或者 AccessDataEngine.exe 即可）

另外，既然将 Excel 当数据库对象来操作，那么在 Excel 中也可以使用 SQL 语。句来进行查找。语法为："SELECT [列名 1], [列名 12] FROM [Sheet1$]". 即 Excel 工作表名后加上"$"，在"[ ]"内加入列名

也许你要问，QTP/UFT 不是有 DataTable 吗？对单元格和行、列的操作完全可以使用 DataTable 对象来完成。是的，没错。但对于大量的数据单元格操作，DataTable 对象并不可取。其实现的复杂度和执行速度远没有使用数据库对象来得简单。下面的案例，可以帮助说明：

如在下列 Excel 文件中（表 2-1 学生成绩清单），需要找出所有科目为 “语文” 的同学名单，怎么做呢？分几步来实现？套用上面的案例讲解，我们可以分这么几步来实现：

第一步： 连接数据库（将 Excel 文件当作数据库处理，即参数中的 Data Resource）

```
Set objADODB = CreateObject("ADODB.Connection")
Set objRs = CreateObject("ADODB.Recordset")
Set fso = CreateObject("Scripting.FileSystemObject")

'objADODB.Provider = "Microsoft.Jet.OLEDB.4.0" 'Microsoft provide OLE DB Provider
objADODB.Provider = "Microsoft.ACE.OLEDB.12.0"
objADODB.Properties("Extended Properties").Value = "Excel 12.0; HDR=yes; IMEX=2"
objADODB.Open "C:\alan.xlsx"
Print "Check the OLEDB connection Status " & objADODB.State ' 1 means successful, 0 means failure
objADODB.Close
```

注意：下面的方法同样可以实现 Excel 数据库的连接

```
objADODB.Open "Provider = Microsoft.ACE.OLEDB.12.0; Persist Security Info = True;Data source = C:\alan.xlsx; Extended Properties = 'Excel 12.0; HDR=yes; IMEX=2'"
Print "Check the OLEDB connection Status " & objADODB.State ' 1 means successful, 0 means failure
```

Column_XingMing	Column_Subjects	Column_Grade	Column_Sexual	Column_Class
Yuan, Hong-bo	语文	98	Male	Class_One
Wang, li-li	数学	96	Female	Class_Two
Yuan, Jun-tao	物理	99	Male	Class_Two
Liu, Yan-Ping	地理	89	Female	Class_One
Ma, Yan	化学	70	Male	Class_One
Wang, Yong-Feng	几何	88	Female	Class_Two
Wang, Yong-Di	语文	90	Male	Class_Two
Wang, Dan-Dan	数学	92	Female	Class_Two
Wang, yi	物理	63	Male	Class_One
Pan, Wei	地理	84	Female	Class_Two
Pan, Jie	化学	75	Male	Class_One
Liu, Fang	几何	82	Female	Class_One
Yuan, Ying	语文	88	Male	Class_Two
Zhang, Zhi-Xia	数学	80	Female	Class_Two
Li, Peng	物理	85	Male	Class_One
LI, Ke-Ai	地理	89	Female	Class_One
Li, Ling	化学	83	Male	Class_Two
Meng, Dai	几何	82	Female	Class_Two
Xiao, He	数学	82	Male	Class_Two
Chen, Xi	物理	88	Female	Class_One
Xu, Xiao-li	地理	85	Male	Class_Two
Xu, Yan	化学	86	Female	Class_One
Yang, Mei	几何	92	Male	Class_One
Yang, Sun	数学	75	Female	Class_Two
Pan, Xiao-Xiao	物理	84	Male	Class_Two
Chen, Li	语文	99	Female	Class_One

表 2-1 学生成绩清单

第二步： 检索符合条件的记录

```
sql = "Select * from [Sheet1$] where column_subjects = '语文'"
recordCount = objRs.RecordCount
Print "数据集中记录数是： " & recordCount

fieldCount = objRs.Fields.Count
Print "数据集中数据记录的字段数是： " & fieldCount
```

```
Do
   For each b in objRs.Fields
   print b
   Next
   print "*****************************************************************************"
   objRs.MoveNext
Loop While Not objRs.EOF
```

注意：上面语句已经将搜索的记录显示在 QTP/UFT 的 Output 视窗里，便于观察。要想使用这些记录还得单独拎出来。

第三步：提取搜索的记录

```
'Create another Excel to store the return record
If Not fso.FileExists("C:\SaveReturnExcel.xlsx") Then
   Set excelApp = CreateObject("excel.application")
   excelApp.Workbooks.Add.SaveAs "C:\SaveReturnExcel.xlsx"
   excelApp.Workbooks.Open "C:\SaveReturnExcel.xlsx"
Else
   Set excelApp = CreateObject("excel.application")
   excelApp.Workbooks.Open "C:\SaveReturnExcel.xlsx"
End If


If excelApp.Sheets.Count > 0 Then
   sheetFlag = False
   For i = 1 To excelApp.Sheets.Count
      If excelApp.Sheets.Item(i).Name = "Search_Result" Then
         sheetFlag = True
      End If
   Next
```

```
End If


If sheetFlag = True Then
    excelApp.Sheets.Item("Search_Result").Activate
    Else
    excelApp.Sheets.Item("Search_Result").Activate
End If

For j = 1 To recordCount
    For i = 0 To fieldCount - 1
        excelApp.Worksheets.Item("Search_Result").Cells(j,i+1) = objRs.Fields.Item(i).Value
    Next
    objRs.MoveNext
Next

excelApp.ActiveWorkbook.Save

excelApp.Quit
Set excelApp = Nothing

Set objRs = Nothing
Set objADODB = Nothing
Set fso = Nothing
```

执行后，脚本将在 C 盘根目录创建一个名为 “SaveReturnExcel.xlsx” 的文件，并且将搜索的记录写入 “Search_Result” 的表单里。结果如下 (表 2-2 搜索成绩清单)：

an, Hong-bo	语文	98	Male	Class_One
Wang, Yong-Di	语文	90	Male	Class_Two
Yuan, Ying	语文	88	Male	Class_Two
Chen, Li	语文	99	Female	Class_One
Zhou, Mei-Mei	语文	85	Male	Class_One
Chen, Xi-Mei	语文	82	Female	Class_Two

表 2-2 搜索成绩清单

怎么样？使用上面的方法来操作数据量大的表单记录是不是很方便？这种方法在操作数据量大的表达时特别有效。

上面提取搜索的记录，主要是把 RecordSet 对象中的记录逐一填充在 Excel 文件指定的字段中。除此之外，我们还有一种简便的方法来处理类似情况。即将搜索到的记录直接 Copy 在 Excel 某一表单（Sheet）里。如何实现呢？请看下面代码 –

```
Set objSheet = CreateObject("Excel.Application")
objSheet.Workbooks.Open "C:\DBSheet2.xlsx"
objSheet.Sheets.Add.Name = "Result"
objSheet.Sheets.Item(1).Activate

objSheet.Workbooks.Open "C:\DBSheet3.xlsx"
objSheet.Sheets.Add.Name = "My"

objSheet.Sheets.Item(1).Range("A2").CopyFromRecordset objCon.Execute(strSQL)
'objSheet.Sheets.Item(1).Cells(2, 4).CopyFromRecordset objCon.Execute(strSQL)

objSheet.DisplayAlerts = False
objSheet.Workbooks.Item(1).Save
'objSheet.Workbooks.Item(2).Save
objSheet.Quit

objCon.Close
Set objCon = Nothing
set objSheet = Nothing
```

上面代码就是讲搜索到的记录，直接 copy 在 excel 文件（DBSheet3.xlsx）的表单（Sheet1）中，并且以 A2 开始的单元格里，使用的是 CopyFromRecordset 方法。

思考一下

如不用这种方式，使用 DataTable 的对象属性来实现上述的自动化操作，该怎么做?

接下来再看一个例子，就是如何自动化操作 Oracle 数据库中的表单。像这样的用例在日常测试中也是大量的存在。这里我依然用一个实例来帮助说明如何自动化这类型的 Cases。请看下面代码。为便于理解，我在代码中加了必要的注解。请一并关注。

```
Dim objCn, objRs, strSQL, statusFlag_1, statusFlag_2, Iterator
statusFlag_1 = False
statusFlag_2 = False

Set objCn=Createobject("ADODB.Connection")
Set objRs =Createobject("ADODB.Recordset")

objCn.ConnectionString = "Provider=OraOLEDB.Oracle; User Id="+ DBUser + &_
";Password="+ DBPassword +"; Data Source="+DBName
objCn.Open

print objCn.State

strSQL = "Select * from" &_
"PROC_LBOPC_A_DETAIL_STATUS,PROC_LBOPC_A_DETAIL" &_
"where fk_proc_a_detail_id=proc_detail_id and" &_
"invoice_id='"& billingTransactionNo &"'"

objRs.Open strSQL, objCn,1, 3 '1 means here open forward only;  3 means here lock pessimistic (Note: number accept here only)
print objRs.RecordCount
```

```
If objRs.RecordCount > 0 Then
    For Iterator = 1 To objRs.RecordCount
    '查看执行 SQL 语句后返回的记录数量

        If objRs.Fields.Item("Status").Value = "PROC" Then '注意获取字段值的写法
            statusFlag_1 = True
        ElseIf objRs.Fields.Item("Status").Value = "EDIT" Then
            statusFlag_2 = True
        End If
        objRs.MoveNext
    Next
End If

If statusFlag_1 = True and statusFlag_2 = True Then
    reporter.ReportEvent micPass, "Step " & stepNo & " : Check DB on DBName“ &_
    “to see the status ", &_  '注意断字符的运用
    " Two records with 'PROC' and 'EDIT' status was found there"
    Else
    reporter.ReportEvent micFail, "Step " & stepNo & " : Check DB on DBName”&_
    “to see the status", "there is NO records with 'PROC' and 'EDIT' there"
End If

objRs.Close
objCn.Close

Set objCn = Nothing
Set objRs = Nothing
```

点评： 这个案例在测试中普遍遇到，实现案例的自动化也不仅仅局限于上面的方法。参考上面的用例，可以使我们快速地掌握有关 Excel 大量数据应用和数据

库表单处理用例的自动化实现。这里需要理解的对象是 ADODB.Connection 和 ADODB.Recordset，同时数据库连接字符串也是一个容易出错的地方。

2.1.4 基于 Outlook 邮件的测试用例自动化实现

如何自动化操作 Outlook 也是大家比较关心的话题，有些 Application 有邮件发送功能，对这部分功能进行测试，我们常常会收到一封邮件，判断邮件的接受与否往往是断定测试成败的一个参考。同时我们也希望把测试的结果以邮件的形式自动发送给客户。这些涉及到对邮件应用程序接口的操作，下面我大概总结下这些常见功能的用法。

跟其他应用程序一样，Outlook 也也为用户保留了一个统一的编程接口 MAPI，使用这个接口提供的方法和属性，我们可以轻松的操作 Outlook. 先来介绍了如何自动发送邮件功能吧。

2.1.4.1 运用 QTP/UFT 如何自动发送 Outlook 邮件

这个功能相信很多做测试的人都期待实现。特别是当测试完成后自动发一封邮件给相关人员，是不是感觉特别专业？好了，废话少说，直接给代码实现。下面的函数将发送一封邮件给相关人员，并加载一附件。

```
Function SendA_Mail_to_SpecifiedPeople(maiTo, mailCC, mailBCC, subject, mailbody, attchedFile)
    Set mailObj = CreateObject("Outlook.application")
    set mailMsg = mailobj.CreateItem(olMailItem) '创建一封新邮件。
    'Set mailSpc = mailobj.GetNamespace("MAPI")

    mailMsg.to = maiTo
    mailMsg.CC = mailCC
    mailMSG.BCC = mailBCC
    mailMsg.Subject = subject
    mailMsg.body = mailbody
```

```
    mailMsg.Importance = olImportanceHigh
    mailMsg.attachments.add attchedFile
    mailMsg.send

    mailMsg.Quit
    Set mailObj = Nothing
    Set mailMsg = Nothing
End Function
```

是不是很简单？还不快试试。这里需要强调一点的是 CreateIte()方法，它可以创建 Outlook 支持的各种邮件类型。CreateItem(olMailItem)方法最常用。它封装了新建邮件的基本功能。

2.1.4.2 检测 Outlook Top 文件夹

这个功能也很实用，很多人在自己的邮箱里设置了不同时期的个人邮件夹，这里给出了如何识别并进入指定的邮箱文件夹。

```
Set myItem = CreateObject("Outlook.application")
maxFile = myItem.GetNamespace("MAPI").Folders.Count

For i = 1 To maxFile
    Set checkMailBoxFile = myItem.GetNamespace("MAPI").Folders(i)
    Print checkMailBoxFile
Next
```

程序将打印出你的个人邮箱文件夹来。确定要进入那个邮箱，你也可以直接将参数 i 换成个人邮箱文件夹的名称。如想进入 “2012.10-2013.10” 的邮箱，就将 i 替换成这个邮箱名即可。

2.1.4.3 检索默认邮箱的文件夹

这个实例将检索默认邮箱的各个子文件夹

```
Set myItem = CreateObject("Outlook.application")
For i = 1 To 15
   set mailBox = myItem.GetNamespace("MAPI").GetDefaultFolder(i)
   msgbox mailBox
   Print i & " : "& mailBox
Next
```

运行结果为：

3 : 已删除邮件
4 : 发件箱
5 : 已发送邮件
6 : 收件箱
7 : 收件箱
8 : 收件箱
9 : 日历
10 : 联系人
11 : 日记
12 : 便笺
13 : 任务
14 : Reminders

有了这个清单，我们很容易进入相关的文件夹。比喻想检索收件箱，我们只需将变量 i 设置为 6，7，8 即可。每个人的默认邮箱设置可能不同，这个清单值也有可能会不一样。有兴趣的话可以自己测试下。

接下来就是查看收件箱里的邮件了，下面我将给大家介绍几个常用的方法。有了这些 方法，我们可以查看是否有新的邮件进来，是否属于测试过程中产生的邮件。我想这也是大家最感兴趣的功能点了。一起来看看下面的实现。

2.1.4.4 检索默认邮箱的 Inbox 文件夹

先来看看如何进入默认邮箱的收件夹（Inbox）和如何检索 Inbox 里的各个子文件夹。请看下面的代码。

```
Set myItem = CreateObject("Outlook.application")
countFolders = myItem.GetNamespace("MAPI").GetDefaultFolder(6).Folders.Count

For i = 1 To countFolders
   Set getEachFolder = myItem.GetNamespace("MAPI").GetDefaultFolder(6).Folders._
   Item(i)
   Print i & "：  " & getEachFolder
Next
```

这是对我的邮箱运行后的测试结果，请参考。

1： Automation Community
2： Boss' Mail
3： China_SalesIT_DevTeam
4： Conversation History
5： CQ Team
6： Customer's Mail
7： Deleted Items
8： EnglishCorner_Datum
9： Important Notices
10： Inida_Chandrababu
11： Inida_Vijay
12： ITI_ChianTeam
13： ITI_Customer
14： MDA_Team
15： PDIM_NA
16： Press_Team
17： Private Mail
18： SalesIT-MDMIT

19： ATPT

基于收件夹里的这些子文件夹，也许你要问，我想知道 ATPT 文件夹里的邮件个数和未读邮件个数。那么该怎么做呢？ 请看下面的代码。

2.1.4.5 检索收件箱里指定文件夹的邮件

下面的代码实现了查看 ATPT 文件夹的邮件数，未读邮件数，指定发件人的邮件数以及符合邮件主题的未读邮件数。这些功能正好可以用来检查我们是否能收到应用程序向我们发送的邮件。 是不是很激动?

```
Set myItem = CreateObject("Outlook.application")
Set checkInboxFolder = myItem.GetNamespace("MAPI").GetDefaultFolder(6).Folders.Item("ATPT")

mailCount = checkInboxFolder.Items.Count
unreadMailCount1 = checkInboxFolder.UnReadItemCount '此句等同于下面的语句
unreadMailCount2 = checkInboxFolder.Items.Restrict("Not [Unread] = 'True'").Count
'此句等同于上面的语句

Print "TotalMailCount :" & mailCount & vblf &_
"unreadMailCount1 : " & (unreadMailCount1) & vblf &_
"unreadMailCount2 : " & unreadMailCount2

unreadMailCount3 = checkInboxFolder.Items.Restrict("[TO] = 'Yuan, Hong-Bo (Alan, HPIT-DS-TCoE)' And Not [Unread] = 'True'").Count
print "unreadMailCount3 : " & unreadMailCount3

unreadMailCount4 = checkInboxFolder.Items.Restrict("[CC] = 'Yuan, Hong-bo' And Not [Unread] = 'True'").Count
print "unreadMailCount4 : " & unreadMailCount4
```

```
unreadMailCount5 = checkInboxFolder.Items.Restrict("[BCC] = 'Ning Zhu' And Not [Unread] = 'True'").Count
print "unreadMailCount5 : " & unreadMailCount5

unreadMailCount6 = checkInboxFolder.Items.Restrict("[FROM] = 'Ning Zhu' And Not [Unread] = 'True'").Count
print "unreadMailCount6 : " & unreadMailCount6
```

执行后程序打印结果如下：

TotalMailCount :11

unreadMailCount1 : 3

unreadMailCount2 : 3

unreadMailCount3 : 3

unreadMailCount4 : 0

unreadMailCount5 : 0

unreadMailCount7 : 3

2.1.4.6 检索收件箱指定文件夹里指定的邮件

其实上面的方法应该足以帮助我们应对那些有关 Outlook 的测试用例了。当然我们也可以深入邮件内容去检查，看是否有满足需求的邮件存在。常用的方法是还是调用 olMailItem 的方法去实现。

例如，这里 ATPT 文件夹里有一份 title 为这样的邮件，我们需要检验是否存在这样的一份邮件。怎么写脚本呢？

Mail Title ： Announcement: our project 'ABC' will release 2.09 from 9:30am to 1:00pm on 6/30/2014 (UTC Time)

```
Set myItem = CreateObject("Outlook.application")
Set checkInboxFolder = myItem.GetNamespace("MAPI").GetDefaultFolder(6).Folders.Item("ATPT")
checkInboxFolder.Display

Set specifiedMails = checkInboxFolder.Items.Restrict("[subject] = 'Announcement: our project 'ABC' will release 2.09 from 9:30am to 1:00pm on 6/30/2014 (UTC Time)'")
Msgbox specifiedMails.Count
'检索有多少封 title 为指定内容的邮件


Set specifiedMails = checkInboxFolder.Items.Restrict("[subject] = 'Announcement: our project 'ABC' will release 2.09 from 9:30am to 1:00pm on 6/30/2014 (UTC Time)' And Not [Unread] = 'True'")
Msgbox specifiedMails.Count
'检索有多少封 title 为指定内容的邮件并且状态是未读的

Set specifiedMails = Nothing
Set checkInboxFolder = Nothing
Set myItem = Nothing
```

点评： 上面的几个案例主要围绕有关 Outlook 用例自动化的解说，一般来说，对任何外部引用程式的自动化都需要获得该应用程式的开放式用户接口，outlook 提供了三个重要的 GUI 接口（Microsoft.Office.Interop.Outlook.MAPIFolder/ Microsoft.Office.Interop.Outlook.MailItem/ Microsoft.Office.Interop.Outlook._NameSpace），对这些接口方法的调用是实现自动化的测试的根本。这一届重点需要掌握的是 CreateItem(olMailItem)，Folder，GetDefaultFolder 以及 Item 等方法的使用。

2.1.5 如何自定义测试报告

回顾一下案例一，也许你已经觉察到了，在案例一中，我们的 report 就是自定义的。是不是？我们定义了一个 Excel 文档，把 Send Request 和 Get Response 的地址给拿出来。方便 review 测试的结果。当时并没有提这个话题，主要因为案例一重点还是讲解如何自动化基于 WS（Web Services）的测试用例。

今天我们重点的讲一讲如何自定义测试报告，这点很重要，测试过程怎么样其实没多少人关注，测试结果是否漂亮往往确定老板或者客户对你个人的看法。呵呵。。。 扯远了，经验之谈。总而言之，能交出一份与众不同的测试报告，确实能给自己加分。

自定义测试报告有很多种，常见的有基于网页形式的 Html 文档，或者是基于表单形式的 Excel 文档，后一种已经介绍过了（还是不知道？那就请回到案例一中，把最后一个函数琢磨琢磨）。这里再来说一说如何设计基于网页形式的 Html 文档报告。

跟建房子一样，首先要有清晰地图纸，这里我们需要先设计出 Html 文档的样式（就是你希望报告最终以什么样的格式呈现出来），这是我设计的报告格式，很简单（稍微复习下 HTML 文档就能做出来）

a) Part One : Set the header of your definite report

```
<Div align="center">
<!DOCTYPE html>
<html lang="zh-cn">
<head>
<meta http-equiv="charset" content="iso-8859-1">
<style type="text/css">
th {color:purple}
td {color:black}
</style>
</head>
```

```
<body>
<div align="Center">
<table border="0" width="90%" bgcolor="Yellow" cellpadding="10">
<h3 style="color:blue" align="left"><font face="Futura Bk">Customized Automaton
 Report_001</font></h3>
<tr>
<th style="color:blue" align="left"><font size="2" face="Futura
 Bk">Executor:</font></th>
<td style="color:red; align:left"><font size="2" face="Futura
 Bk"><i>Alan</i></font></td>
<th style="color:blue" align="left"><font size="2" face="Futura Bk">Execute
 Date:</font></th>
<td style="color:red" align="left"><font size="2" face="Futura Bk"><i>2014-06-
 06</i></font></td>
<th style="color:blue" align="left"><font size="2" face="Futura Bk">Script
 Name:</font></th>
<td style="color:red" align="left"><font size="2" face="Futura
 Bk"><i>ScriptForCase012014-06-06</i></font></td>
<tr>
<tr>
<th style="color:blue" align="left"><font size="2" face="Futura
 Bk">CaseNo:</font></th>
<td style="color:red; align:left"><font size="2" face="Futura Bk"><i>001</i></font></td>
<th style="color:blue" align="left"><font size="2" face="Futura Bk">Test
 Environment:</font></th>
<td style="color:red; align:left"><font size="2" face="Futura
 Bk"><i>Win7+IE8+UFT12.0</i></font></td>
<th style="color:blue" align="left"><font size="2" face="Futura
 Bk">Consumption:</font></th>
<td style="color:red; align:left"><font size="2" face="Futura Bk"><i>5
 Seconds</i></font></td>
```

```
<tr>
</table>
<hr width="90%">
</div>
<table width="90%" border="0" cellspacing="10">
<colgroup span="2" align="left"></colgroup>
<colgroup align="center"></colgroup>
<tr>
<th><font size="3" face="Aharoni"><u>Checkpoint</u></font></th>
<th><font size="3" face="Aharoni"><u>Content Statement</u></font></th>
<th><font size="3" face="Aharoni"><u>Result</u></font></th>
</tr>
```

b) Part Two : Set your checkpoint tracking form

```
<tr bgcolor=white>
<td><li><font size="2" facce="arial">Validate the login process and see if it's smooth as
 TC</font></li></td>
<td><font size="2" facce="arial">Login successfully</font></td>
<td><font size="2" facce="arial" color=green><b>PASSED</b></font></td>
</tr>
<tr bgcolor=#87CEEB>
<td><li><font size="2" facce="arial">Validate the login process and see if it's smooth as
 TC - Validation for Case 001, adding additional message 'th'</font></li></td>
<td><font size="2" facce="arial">Login successfully and adding a plus message, like this ,
 like that , ll lj ljsdfa like all the thingks</font></td>
<td><font size="2" facce="arial" color=red><b>FAILED</b></font></td>
</tr>
<tr bgcolor=white>
<td><li><font size="2" facce="arial">Validate the login process and see if it's smooth as
 TC</font></li></td>
<td><font size="2" facce="arial">Login successfully</font></td>
```

```
<td><font size="2" facce="arial" color=green><b>PASSED</b></font></td>
</tr>
```

c) Part Three : Set the definite report ending

```
</table>
</body>
</html>
<br>
<hr width="90%">
</div>
```

d) 代码实现

根据上面的 HTML Report 文档，我们需要用函数的形式把每一部分都写入 HTML 文件，最后打开这个文件，就是我们想要的 Report。先来看看第一部分是如何实现的，请看下面代码，将第一部分内容写入 Html 文件。

```
Function addCheckpointHeaderToReport(caseNo, colorVariable)
    Dim abc, aFile  '函数中的变量定义不可与函数自身的形参重复，

    Set abc = CreateObject("Scripting.FileSystemObject")
    Set aFile = abc.CreateTextFile("C:\addReportContent.html", True)
    'Set aFile = nothing

    Set aFile = abc.CreateTextFile("C:\reportHeader.html", True) ‘ True 代表创建的文件是否可以覆盖已存在的同名文件。
    aFile.WriteLine("<!DOCTYPE html>")
    'aFile.WriteLine("<html lang=""zh-cn"">")
    aFile.WriteLine("<head>")

'    <meta charset="gbk" />
```

```
aFile.WriteLine("<meta http-equiv=""charset"" content=""iso-8859-1"">")
aFile.WriteLine("<style type=""text/css"">")
aFile.WriteLine("th {color:purple}")
aFile.WriteLine("td {color:black}")
aFile.WriteLine("</style>")
aFile.WriteLine("</head>")

""ReportI - for enviroment report
aFile.WriteLine("<body>")
aFile.WriteLine("<div align=""Center"">")
'aFile.WriteLine("<hr width=""90%"">")
aFile.WriteLine("<table border=""0"" width=""90%"" bgcolor=""Yellow""
cellpadding=""5"">")

aFile.WriteLine("<h3 style=""color:blue"" align=""left""><font face=""Futura
Bk"">Customized Automaton Report_"& caseNo & "</font></h3> ")
aFile.WriteLine("<tr>")
aFile.WriteLine("<th style=""color:blue"" align=""left""><font size=""2"" face=""Futura
 Bk"">Executor:</font></th>")
aFile.WriteLine("<td style=""color:red; align:left""><font size=""2"" face=""Futura
Bk""><i>Alan</i></font></td>")
aFile.WriteLine("<th style=""color:blue"" align=""left""><font size=""2"" face=""Futura
Bk"">Execute Date:</font></th>")
aFile.WriteLine("<td style=""color:red"" align=""left""><font size=""2"" face=""Futura
Bk""><i>2014-06-06</i></font></td>")
aFile.WriteLine("<th style=""color:blue"" align=""left""><font size=""2"" face=""Futura
Bk"">Script Name:</font></th>")
aFile.WriteLine("<td style=""color:red"" align=""left""><font size=""2"" face=""Futura
Bk""><i>Scripts for Case001</i></font></td>")
aFile.WriteLine("<tr>")
```

```
    aFile.WriteLine("<tr>")
    aFile.WriteLine("<th style=""color:blue"" align=""left""><font size=""2"" face=""Futura
    Bk"">CaseNo:</font></th>")
    aFile.WriteLine("<td style=""color:red; align:left""><font size=""2"" face=""Futura
    Bk""><i>001</i></font></td>")
    aFile.WriteLine("<th style=""color:blue"" align=""left""><font size=""2"" face=""Futura
    Bk"">Test Environment:</font></th>")
    aFile.WriteLine("<td style=""color:red; align:left""><font size=""2"" face=""Futura
    Bk""><i>Win7+IE8+UFT12.0</i></font></td>")
    aFile.WriteLine("<th style=""color:blue"" align=""left""><font size=""2"" face=""Futura
    Bk"">Consumption:</font></th>")
    aFile.WriteLine("<td style=""color:red; align:left""><font size=""2"" face=""Futura
    Bk""><i>5 Seconds</i></font></td>")
    aFile.WriteLine("<tr>")

    aFile.WriteLine("</table>")
'   aFile.WriteLine("<hr width=""90%"">")
    aFile.WriteLine("</div>")     '定义 div 标签，对此标签的操作将影响到此标签中的
    table
    aFile.WriteLine("<br>")
    'aFile.WriteLine("<hr align=""left"" width=""80%"">")

    '''Report II - for Checkpoints list
    aFile.WriteLine("<table width=""90%"" border=""0"" cellpadding=""2""
    cellspacing=""5"" bgcolor=""green"">")
    aFile.WriteLine("<colgroup span=""2"" align=""left""></colgroup>")
    aFile.WriteLine("<colgroup align=""center""></colgroup>")

    aFile.WriteLine("<tr bgcolor=""Green"">")
    aFile.WriteLine("<th><font size=""3"" facce=""Aharoni""
```

```
    color=""yellow""><u>Checkpoint</u></font></th>")
    aFile.WriteLine("<th><font size=""3"" facce=""Aharoni""
    color=""yellow""><u>Content Statement</u></font></th>")
    aFile.WriteLine("<th><font size=""3"" facce=""Aharoni""
    color=""yellow""><u>Result</u></font></th>")
    aFile.WriteLine("</tr>")

    Set abc = Nothing
    Set aFile = nothing
End Function
```

点评： 先创建一个 FSO 对象，然后由这个对象创建一个 Html header 文件“ReportHeader.html”，再将第一部分 xml 文件格式写入这个 header 文件。这里主要调用的是 FSO 的 Writeline 方法。（此方法在 FSO.CreateStream 对象里）。接着再来看第二部分函数的实现。下面的函数就是把每个 checkpoint 值用 Html 表述出来。

```
Function addCheckpointContentToReport(checkpointName, checkpointContent, result)
    Dim abc, aFile, colorVariable, resultColor

    Set abc = CreateObject("Scripting.FileSystemObject")
    If abc.FileExists("C:\addReportContent.html") Then
        Set aFile = abc.OpenTextFile("C:\addReportContent.html", 8)'8 是 iomdoe 的参数
        值， 8 代表 for Appending，即 Open a file and write to the end of the file.
    Else
        Set aFile = abc.CreateTextFile("C:\addReportContent.html", True)
    End If

    '加入颜色判断
    If result = "FAILED" Then
        colorVariable = "grey"
        resultColor = "red"
    ElseIf result = "PASSED" Then
```

```
        colorVariable = "white"
        resultColor = "green"
    End If

    aFile.WriteLine("<tr bgcolor="& colorVariable &">")
    aFile.WriteLine("<td><li><font size=""2""
    face=""arial"">" & checkpointName & "</font></li></td>")
    aFile.WriteLine("<td><font size=""2""
    face=""arial"">"& checkpointContent & "</font></td>")

    If result = "FAILED" Then
        aFile.WriteLine("<td><font size='2' face='arial' color=" & resultColor & "><a
href='http://www.sina.com.cn/'><b>"& result &"</a></b></font></td>") 'Note : 属性值
应该始终被包括在引号内。双引号是最常用的，不过使用单引号也没有问题。在某
些个别的情况下，比如属性值本身就含有双引号，那么您必须使用单引号

      Else
        aFile.WriteLine("<td><font size=""2"" face=""arial""
        color=" & resultColor & "><b>"& result &"</b></font></td>")
    End If

    aFile.WriteLine("</tr>")

    Set abc = Nothing
    Set aFile = nothing
End Function
```

点评：这部分函数使用的方法和第一部分完全一样，只是参数化了部分变量，为的是动态写入 Checkpoint 的相关变量值。

接下来要做的是什么呢？想想看？我们已经完成了 Report 的 Header 和 Content 部分。结合上面完整的 HTML Report 看，我们还缺少第三部分的代码没有

用函数表述出来，并且还缺少一个函数将这三部分的内容拼合成一个完整的 HTML 代码。下面的函数就是要完成这样的工作，一起来看看它是如何实现的。

```
Function getFinalReport(createFinalReportFile, getHeaderFile, getContentFile)
    Dim abc, aFile, temp, getHeder, getContent

    Set abc = CreateObject("Scripting.FileSystemObject")

'   If abc.FileExists(createFinalReportFile) Then
'       set aFile = abc.OpenTextFile(createFinalReportFile, 8)
'   Else
'       Set aFile = abc.CreateTextFile(createFinalReportFile, True)
'   End If

    Set aFile = abc.CreateTextFile(createFinalReportFile, True)

    Set temp = abc.OpenTextFile(getHeaderFile, 1)
    getHeader = temp.ReadAll
    Set temp = nothing

    Set temp = abc.OpenTextFile(getContentFile, 1)
    getContent = temp.ReadAll
    Set temp = nothing
    'msgbox getContent
    aFile.WriteLine("<Div align=""center"">")
    aFile.WriteLine getHeader
    aFile.WriteLine getContent
    aFile.WriteLine("</table>")
    aFile.WriteLine("</body>")
    aFile.WriteLine("</html>")
    aFile.WriteLine("<hr width=""90%"">")
```

```
    aFile.WriteLine("<br>")
    Set abc = nothing
    Set aFile = nothing
End Function
```

上面的三个函数共同实现了个性化的 report 制定，看起来好像挺恐怖的，其实还是很好理解的，关键是看用户需要制定什么样的 report，这个 report 的 Html 代码是怎么样的(需要点 Html 的基本知识，也可以借助专业工具如 FrontPage 实现)，然后拆分 Html 代码，调用 QTP/UFT 的相关方法（FSO）实现动态组装。上面三个函数正式基于这一原理展开的。

2.1.6 如何自动化外部参数设置

这个本想放在后面再说，因为它不是一个完整的 Case，和上面的案例比较起来，它只是一个辅助设置。主要用来增强脚本的可移植性和健壮性。但这部分内容又是和上面的案例介绍是紧密相连的。它们是一个完整脚本不可或缺的部分。所以还是有必要在这里介绍下。

了解这一部分的内容，可以帮助我们更好地理解 QTP/UFT 脚本的结构逻辑。有些人写完了脚本，调试也没问题，可就是在将脚本移送给老板时出错。是不是很郁闷？这还算好的，怕就怕你正得意的将脚本分享给客户，好让他们给你一个肯定的赞，结果却被告知，不能 running。这水是不是泼的够凉？呵呵，别着急，其实也没什么，不就是几个设置吗？一起来研究下。

2.1.6.1 Open Specified Script

```
Set qtpEnvApp = CreateObject("QuickTest.application") ' 首先创建 UFT 对象，然后调用对象方法属性对其进行设置

qtpEnvApp.launch ' 开启 UFT Application
qtpEnvApp.visible = True ' 使 UFT 界面可见
qtpEnvApp.Open "C:\Users\yuahongb\Desktop\FY14-One\forUFTpractice\" &_
```

```
"Outlook_Functions_Scripts", False, True ' 打开指定的脚本
```

2.1.6.2 Set exectuion behavior *(Setting points from : Tools →Option)*

```
qtpEnvApp.Options.Run.RunMode = "Normal" 'or Fast
qtpEnvApp.Options.Run.ViewResult = True ' the result will popsup when run session end
qtpEnvApp.Options.Run.ImageCaptureForTestResults = "OnError" ' Always / OnError
OnWarning / Never
```

2.1.6.3 Set Run setting *(Setting points from : File →Settings →Run)*

```
Set qtpEnvSet = qtpEnvApp.Test.Settings
With qtpEnvSet
    .Run.IterationMode = "oneIteration" 'rngAll ' rngIterations
    .Run.DisableSmartIdentification = True
    .Run.OnError = "NextStep" 'Dialog/ NextIteration/ Stop/ NextStep
End With

' 另一种执行方式设置，脚本将运行 DataTable 的第一行和第二行
With qtpEnvSet
    .Run.IterationMode = "rngIterations"
    .Run.StartIteration = 1
    .Run.EndIteration = 2
    .Run.OnError = "NextStep"
End With
```

2.1.6.4 Set Associate Libarary and Object

```
qtpEnvApp.Open "your script name", False, False ' Open a test
Set associatedLibraries = qtpEnvApp.Test.Settings.Resources.Libraries ' Get the libraries
collection object

' Add Utilities.vbs if it's not in the collection
```

```
If associatedLibraries.Find("C:\Utilities.vbs") = -1 Then ' If the library cannot be found in the collection
    associatedLibraries.Add "C:\Utilities.vbs", 1 ' Add the library to the collection
End If

'Set Object repository
Set qtRepositories = qtpEnvApp.Test.Actions("Action1").ObjectRepositories ' Get the object repositories collection object of the "Action1" action

' Add MainApp.tsr if it's not already in the collection
If qtRepositories.Find("C:\MainApp.tsr") = -1 Then ' If the repository cannot be found in the collection
    qtRepositories.Add "C:\MainApp.tsr", 1 ' Add the repository to the collection
End If

' 5. Set the DataTable
Set qtTestResources = qtpEnvApp.Test.Settings.Resources

' Specify an external Data Table file
qtTestResources.DataTablePath = "C:\Resources\Default.xls"
```

2.1.6.5 Format Report and Lauch the Test

```
Set outputTestResult = qtpEnvApp.Options.Run.AutoExportReportConfig
With outputTestResult
    .AutoExportResults = True
    .DataTableReport = False
    .ExportLocation = "C:\QTPProjectName\TestReport" '存储脚本的路径，不是文件名
    .LogTrackingReport = False
    .ScreenRecorderReport = False
```

```
    .StepDetailsReport = True
    .SystemMonitorReport = False
    .StepDetailsReportFormat = "Detailed" 'or short/User-Defined XSL
    .StepDetailsReportType = "HTML" 'or Doc/Pdf
End With

qtpEnvapp.Test.Run ' 执行打开的测试
'或者将执行结果指定到某一存储位置,设置路径为：Run --> run
Set testResult = CreateObject("QuickTest.RunResultsOptions")
testResult.ResultsLocation "C:\Tests\Test1\Res1"
qtpEnvApp.Test.Run testResult

qtpEnvApp.Quit

Set qtpEnvApp = Nothing
Set qtpEnvSet = Nothing
Set qtRepositories = Nothing
Set qtLibararies = Nothing
Set testResult = Nothing
Set outputTestResult = Nothing
```

点评：上面的脚本从注释中可以知道他们的作用，基本上就是代替手工设置 Run-time Setting 和 Option 选项。另外脚本的执行和测试报告的输出也可以用自动化脚本实现。

需要注意的是，利用 QuickTest.Application 接口设置 QTP 的执行环境，有些参数是不能在当前窗口的脚本中进行设置的（例如，在当前打开的 QTP 脚本中，是不能利用 QuickTest.Application 接口进行 FunctionLibarary 的加载操作的，原因是，在脚本执行时，函数库会被自动锁定。不允许进行本脚本的函数库操作。）好在，QTP/UFT 提供了另外两种加载函数的方法，ExecutionFile 和 LoadFunctionLibarary。

如此一来，我们还是可以通过自动化语句将所有的执行环境进行事前设定。一种比较普遍的做法是，将当前脚本的函数库，对象库，测试数据，以及执行参数和最终报告输出细节进行函数封装。并置于单个脚本的头部，这样，当我们执行某一个具体脚本时，不用担心这些外部环境的设置了。请看下面的函数代码，它封装了当前脚本的常用环境变量 –

```
Function SetExecutionEnvironment(objFileName, dataTableName, runIteration,
fromGlobalRow, toGlobalRow, viewResult, autoExportResult, reportType,
reportFolderName)

    Dim scriptPath, actionName, commonPath, qtpObj, fso
    Dim dateArray, getCurrentDate, reportPath

  ' Test Setting -> Resource
  Set qtpObj = CreateObject("QuickTest.Application")
  Set fso = CreateObject("Scripting.FileSystemObject")

  scriptPath = Environment.Value("TestDir")
  actionName = Environment.Value("ActionName")
  scriptName = environment.Value("TestName")
  commonPath = Left(scriptPath, InstrRev(scriptPath, scriptName) - 1)

  ' Check and define report path -
  dateArray = Split(Date(), "/")
  getCurrentDate = dateArray(2)&dateArray(0)&dateArray(1)

  If fso.FolderExists(commonPath & reportFolderName) Then
     If fso.FolderExists(commonPath & reportFolderName & "\" & getCurrentDate) Th
        en
        reportPath = commonPath & reportFolderName & "\" & getCurrentDate
        Else
        fso.CreateFolder commonPath & reportFolderName & "\" & getCurrentDate
```

```
            reportPath = commonPath & reportFolderName & "\" & getCurrentDate
        End If
    Else
        fso.CreateFolder commonPath & "\" & reportFolderName
        fso.CreateFolder commonPath & reportFolderName & "\" & getCurrentDate
        reportPath = commonPath & reportFolderName & "\" & getCurrentDate
    End If
    ''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''''

    qtpObj.Test.Actions(actionName).ObjectRepositories.RemoveAll
    qtpObj.Test.Actions(actionName).ObjectRepositories.Add
commonPath & objFileName

    ' Test Setting -> Resource
    'qtpObj.Test.DataTable.Import commonPath & dataTableName ' import a sheet to a design
time data table (也就是用指定的一个文件来初始化 data table 关联的文件。sheet 数以
关联的文件 sheet 数为准) 另外，当用这种方法加载 test Data 时，请记住 DataTable 的
初始设置为 Default Location
    qtpObj.Test.Settings.Resources.DataTablePath = commonPath & dataTableName ' 决
定运行次数（Iteration）是由指定 Global 表的有效行数来决定的。所以，当应用外部
data table 来初始化 QTP 的关联 test table 时，请务必包含 Global 表单。这样下面的运
行设置就可以随心制定了

    ' Test Setting -> Run
    If runIteration = "One" Then
        qtpObj.Test.Settings.Run.IterationMode = "oneIteration" ' another Values - 'rngAll',
'rngIterations/StartIteration/EndIteration'
    ElseIf runIteration = "All" Then
        qtpObj.Test.Settings.Run.IterationMode = "rngAll" ' another Values - 'rngAll',
'rngIterations/StartIteration/EndIteration'
    Else
        qtpObj.Test.Settings.Run.IterationMode = "rngIterations"
```

```
        qtpObj.Test.Settings.Run.StartIteration = fromGlobalRow
        qtpObj.Test.Settings.Run.EndIteration = toGlobalRow
    End If

    qtpObj.Test.Settings.Run.DisableSmartIdentification = True

    ' Tools -> Option -> Run Session
    If viewResult = "Yes" Then
        qtpObj.Options.Run.ViewResults = True
    Else
        qtpObj.Options.Run.ViewResults = False
    End If

    If autoExportResult = "Yes" Then
        qtpObj.Options.Run.AutoExportReportConfig.AutoExportResults = True

        qtpObj.Options.Run.AutoExportReportConfig.StepDetailsReport = True
        qtpObj.Options.Run.AutoExportReportConfig.StepDetailsReportType = reportTyp
e 'another values - 'PDF', 'DOC'
        qtpObj.Options.Run.AutoExportReportConfig.StepDetailsReportFormat = "Short" '
another Values - 'Detailed', 'UserDefined'
        qtpObj.Options.Run.AutoExportReportConfig.ExportLocation = reportPath
        qtpObj.Options.Run.AutoExportReportConfig.ExportForFailedRunsOnly = False

        qtpObj.Options.Run.AutoExportReportConfig.ScreenRecorderReport = False
        qtpObj.Options.Run.AutoExportReportConfig.DataTableReport = False
        qtpObj.Options.Run.AutoExportReportConfig.LogTrackingReport = False
        qtpObj.Options.Run.AutoExportReportConfig.SystemMonitorReport = False
    ElseIf autoExportResult = "No" Then
        qtpObj.Options.Run.AutoExportReportConfig.AutoExportResults = False
    End If
```

```
    qtpObj.Options.Run.RunMode = "Fast" ' another Value - 'Normal'

    qtpObj.Test.Save
'   qtpObj.Quit
    Set qtpObj = Nothing
    Set fso = Nothing
End Function
```

在调用这个函数前，还需要加载 ExecutionFile 或 LoadFunctionLibarary 语句来激活指定的函数库。如下面代码所示 –

```
scriptPath = Environment.Value("TestDir")
scriptName = environment.Value("TestName")
commonPath = Left(scriptPath, InstrRev(scriptPath, scriptName) - 1)

LoadFunctionLibrary commonPath & "practiceScript_001_library.qfl" '用这种方法来加载函数库，可以实现由脚本来自动化执行环境,用这种方式加载数据库时，TestData 的加载最好用 qtpObj.Test.Settings.Resources.DataTablePath 而不要用 qtpObj.Test.DataTable.Import
'ExecuteFile commonPath & "practiceScript_001_library_ansi.qfl" '用这种方式加载函数库时，可以使用上述两种方式加载 TestData

'Note -
' 本文通过简单的例子和大家一起分析了两种动态调用函数库方式的区别，其实两种方式没有谁好谁坏，只有适合和不适合，如果你的测试设计过程中需要使用到类，那么推荐你使用 ExecuteFile 方法，但是如果你使用的是纯函数库并没有类，那么 LoadFunctionLibrary 的调试功能能让更快的定位错误。
```

小结：利用 QTP/UFT 提供的接口和语句，可以实现执行环境的自动化设定。当脚本中调用多个 Action 时，每个 action 的设置可以参考上述步骤和方法，但执行的循环次数是以当前设置的 Runtime 次数为准（即脚本运行前的 runtime 设置为准）。

尽管这个设置在脚本 running 过程中是可以改变的。另外，不建议每个 action 调用各自不同的测试数据。也就是说当多个 Action 串联一起执行时，建议把测试数据集中在一个 test data 文件中。

第三章 常用方法及场景参考

通过第二章的学习，使我们对常见测试用例的自动化实现有了一个相对完整的认识。只要多加练习，不断深化和巩固对这些方法和思路的认识，必将有助于你的自动化水平的提高。另外，案例中所涉及的编程技巧也有一定的普遍性，理解和熟悉他们对个人技能的提升也大有裨益。当然，这些案例都是在特定的项目环境下挑选出来的。不能完全照搬。它的价值在于，给我们指出了一条有效的途径去实现各类常见用例的自动化转换。接下来，我将试着整理一些常用的测试方法，这不仅有利于提高编程效率，更重要的是能在短时间内将我们的自动化水平提升到一个新的高度。怎么样，心动了吧?

3.1 常用方法介绍

3.1.1 关于 MST 文件的设置

相信大家都有这样的困惑，就是在你编写脚本的时候，每次增加一个 Action，都需要手动的加入一些注释去做一些说明，比喻这个 Action 的目的是什么，对应的用例步骤是哪一个，还有创作者，创建日期什么的等等信息。下面是我常用的注解，使用注释是一个好的习惯，但每次要手动去做，不免有些繁琐，这里为大家推荐一个简便的方法，使用 Mecury Script Templat 可以帮助我们省去不少的麻烦。一起看看如何做。

```
'============================================================
'1.Script_Name: Script_001
'2.Business_Statement: XXXX
'3.Project_Name: Project_XXX
'4.Mapping_Cases:Case_01
'5.Author:XXX
'6.Created_Date:YYYY_MM_DD
'============================================================
```

其实很简单，Mecury 早就想到这点了，只需要设置一个配置文件，将上述内容实现填入这个文件，并命名为 “ActionTemplate.MST” 即可。定义好后将该文件放入 QTP/UFT 安装路径下的 /Dat 文件夹下。

3.1.2 关于随机数的产生

在实际测试中，我们常常需要随机生成一个数来参数化变量，有时还希望能将时间和日期附上。或者需要随机生成一个字符串等等。这些情况经常会遇到，下面的函数能帮助我们实现这些要求。

3.1.2.1 随机生成一个 1 ~ n 之间的自然数

```
Function GetRandomNumber(byVal n, byRef rndNumber)
   Randomize
   rndNumber = Int(n * Rnd) + 1
   GetRandomNumber = rndNumber
End Function
```

点评：主要是 Rnd 函数和 Randomize 函数的应用，Randomize 函数为 Rnd 函数提供了一个初始的随机参考值。通常以系统时间为基准产生的。如果不事先引用 Randomize 函数，那么 Rnd 函数每次调用都会生成相同的随机数。当然如果要生成一个从 0 到 n~1 的自然数，就直接是 Int(n * Rnd)

3.1.2.2 随机生成一个 指定长度的字符串（由大小写字母随机组成）

```
Function GetRandomString(strLength)
  For i=1 to strLength
     Randomize
     y = Int(2* Rnd) ' only select from the upper case and the lowercase letter, so only cases
     can be choosed here

     Select Case y
     Case 0
       vChar = int(26*Rnd) +  65 ' the Upper Case's ASCII is from 65 ~ 90   // 26 * Rnd will
        generate a random number from 0 to 25
     Case 1
       vChar = int(26*Rnd)  + 97 ' the Lowercase letter 's ASCII is from 97 ~ 122
```

```
    End Select

   GetRandomString = GetRandomString & Chr(vChar)
  Next
End Function
```

3.1.2.3 随机生成一个 指定长度的字符串（由数字，字母和下划线组成的）

```
 Function GetRandomString(strLength)
  For i=1 to strLength - 1
     Randomize
    y = Int(3* Rnd) 'there are three cases can be select here with upper case, lowercase letter
    and digitals, so only cases can be choosed here

    Select Case y
    Case 0
      vChar = int(26*Rnd) +  65
    Case 1
      vChar = int(26*Rnd)  + 97
    Case 2
      vDigital = int(10*Rnd)  ' Generate random number between 0 ~ 9
    End Select

    If y = 2 Then
       GetRandomString1 = GetRandomString1 & vDigital
       Else
       GetRandomString2 = GetRandomString2 & Chr(vChar)
    End If

   GetRandomString = GetRandomString2 & "_" & GetRandomString1
```

```
  Next
End Function
```

点评： 上面两个函数大同小异，主要还是 Randomize 和 Rnd 函数的使用，另外，根据 ASCII 表，我们可以随机产生一个范围在大写字母表内的整数值（65～90）和范围在小写字母表内的整数值（97～122），然后再将其类型强制转化为字符（Chr）。上面的方法，可以拆开来写，比如想生成一个制定字母个数和数字个数的字符串，可以将字母部分和数字部分分开来写。然后再将其组合即可。

3.1.3 关于日期和时间

在实际测试中，经常要使用当前的日期和时间作为输入参数，这里大概介绍下如何提取当前日期和时间的函数实现。

```
Function GetTime_Date(byVal dateDelimeter, timeDelimeter)
   Dim hh, min, ss, yy, mn, dd, getTime, getDate

   'Get current time
   hh = Hour(Now)
   min = Minute(Now)
   ss = Second(Now)
   If hh<10 Then
      hh = "0"& hh
   ElseIf min<10 Then
      min = "0"& min
   ElseIf ss <10 Then
      ss = "0" & ss
   End If

   getTime = hh & timeDelimeter & min & timeDelimeter & ss
```

```
    'Get current Date
    yy = year(now)
    mm = month(now)
    dd = day(now)

    GetDate = yy & dateDelimeter & mm & dateDelimeter & dd

    GetTime_Date = GetDate & " " & getTime
End Function
```

点评：这个函数关键是在获取当前日期和时间后做了一些配置而已。获取当期日期和时间的函数是 Now（），获取当期时间的函数是 Time（）；除此之外，像一些常用的时间日期函数也需要留意，如 Hour（），Minute（），Second（），Day（），Month（），Year（），Weekday（）等等。另外，DateDiff 函数也是用的比较广泛的，此函数主要用于获取时间间隔。在实际使用中，也应该多加留意。

3.1.4 SendKeys 方法的使用

这个方法在实际使用也比较常见。多见于通过键盘发送字符串给活动的窗口或程序。语法为： SendKeys *String*

这里参数 String 为指定发送的击键。可以是一个或者多个字符。如果指定单个键盘的字符，如 A，B，C 等。可以用键盘字符本身加双引号表示。如 “A”，“B”，“C” 后者组合键 “ABC” 。对于一些对 SendKeys 函数有特殊意义的符号，如“+”，“^”，“%” 等需要加大括号括起来。如，“{+}”，“{^}”，“{%}”。参考 Help 文件，那里列出了所有特殊含义符号的 String 代码。参考下面的函数，一起来理解下如何使用 SendKeys 来编写脚本。

```
Set WshShell = CreateObject("WScript.Shell")
WshShell.Run "Calc"
Wait 5
```

```
Wshshell.SendKeys "1"
Wshshell.SendKeys "{+}"
Wshshell.SendKeys "2"
WshShell.SendKeys "{=}"
Wshshell.SendKeys "%{f4}" ' Close the Claclutor App
```

点评： 这个方法主要是利用 Wscript.Shell 组件里的 SendKeys 方法，帮助实现模拟键盘的操作。通常用于那些缺少 Interface 的 Application page。

3.1.5 变量定义部分

在 VB 中，变量的定义分为两种形式，一个是显示定义，一个是隐式定义。显示定义变量时，需要在一开始声明“**Option Explicit** ”关键字。即表明变量的使用是先申明，后使用。下面代码，清晰表达了显示和隐式的区别。

```
Option Explicit   ' Force explicit variable declaration.
Dim MyVar   ' Declare variable.
MyInt = 10   ' Undeclared variable generates error.
MyVar = 10   ' Declared variable does not generate error.
```

想要编译通过，需要将变量 MyInt 先声明(即 Dim MyInt)或者去掉 Option Explicit 关键字.

说到这里，我想再提一下函数和子程序的相关知识，尽管我在上面的介绍中提及过它们的用法和区别，但并没有涉及到如何调用的问题。这里再作一点补充。我们知道函数 Function 和子程序 Sub（Subroutine）其实都是封装了一定语句的完整功能代码。它们的用法基本相同，不同的是 Function 可以返回值，而 Sub 则不能有返回值。在调用它们的时候，通常有如下几种方式：一是直接使用 Call 关键字。此时的函数参数必须用括号括起来。二是直接写函数名和子程序名称，此时的函数的参数不需要括号，三是用类封装起来。直接类名.函数名即可。具体哪一种，看个人偏好了。

3.1.6 有关字符串整理的常用方法

LTrim/RTrim/Trim

这个比较简单，但却是很常用，主要是去除字符串前后的空格，这对于获取地址或者有效字符比较有用。

Ltrim – 剔除字符串左边的空格

Rtrim – 剔除字符串右边的空格

Trim – 剔除字符串左右两边的空格

3.1.7 设置脚本环境的几个常用函数

1.) LoadAndRunAction

LoadAndRunAction(TestPath, ActionName, [Option]Iteration, [Option]Parameters)

这里要注意两个参数的定义。TestPath – 就是你的 QTP/UFT 脚本路径；Iteration 的赋值有 oneIteration/(0) or alliteration/(1)

这个函数将调用你脚本库中的脚本来执行。如果被执行的脚本还关联了 FuntionLibrary，ObjectRepository， DataTable 或者外部 XML 文件等等，你得先把这些测试数据在当前脚本中关联起来，不然肯定是跑不过的（道理很简单，没有满足执行条件）。

2.) LoadFunctionLibrary(path)

顾名思义这个函数就是装入函数库，如果你不在 QTP/UFT 的 setting 中设置关联函数库的话，可以使用这个函数进行动态库函数加载。加载后运行的脚本可以正常调用库中的函数，类，子函数等。

3.) RepositoriesCollection

此对象中包含多个方法，可对对象库进行增，删，移除等操作，跟上面函数功能一样，如果你没有在 Resources 中关联对象库，可以调用此对象的相关操作进行动态加载对象库，加载后，可以正常使用库中的对象。

如：RepositoriesCollection.Add "../test.tsr"

4.) ExecutionFile(Path)

此方法常常用于将函数库中的类读入当前环境中。怎么理解呢，如果你的脚本中包含了类函数，由于 QTP 中不能直接对类函数进行引用，即您在脚本编写模式下无法直接调用类中的函数。怎么办呢？可以使用这个方法，把类函数关联进来。当前必须确保类函数是 ANSI 格式。否则会报字符格式不匹配错误，这应该算是 QTP 的一个 bug 吧。

写到这里，很自然想到了无人定时脚本执行的问题。实际上这个话题在上一章也提到过，这里再来梳理下吧，并且给出一个实例参考。

想要实现无人执行脚本，无非就是要让脚本能够自动加载，包括脚本关联的 Library，Object，DataTable 等等。将他们编写成批处理命令。再通过 Window 或者专用的定时执行工具加载这个批处理，实现无人定时执行脚本。想想是不是很激动？不要小看这些微小的改变，它会让你看上去立马高大上起来。因为自动化测试上手并不难，而且技术思维很容易同质化，也就是说你做的东西，很容易被复制。要想看上去不一样，就得在测试思路和最终输出结果上下点功夫。总之，一句话，要玩点花样，使自己看上去不那么菜。呵呵，言归正传，还是看看如何实现无人监控测试吧

3.1.8 如何实现脚本的定时自动执行

在上一章节，我们就提出过这个问题，就是在 QTP/UFT 环境设置那一环节。在那一环节里，我们介绍过如何设置 QTP 的执行环境。就是创建 QTP 组件，然后通过引用其组件中的各个对象，实现对执行环境的设置，也可以实现执行某个指定的脚本的操作。当然这一切还是在 QTP 的环境下进行的。如果真的只能在 QTP 环境下做这些的话，那就没有必要引用 QTP 组建进行各种执行和环境的设置了。因为那样只会使事情复杂化，还不如手工设置来的方便。那么有没有可能不在 QTP 的环境下，也能自动的执行指定的脚本呢？当然是可以的。请看下面的步骤。

1.) 先调用 QTP 组件，编写测试环境和指定要执行的脚本。这一步可以在 QTP 的环境下快速完成。
2.) 将第一步定义好的执行操作保存为 VBS 文件
3.) 执行 VBS 文件

重点在第三步上，执行 VBS 文件有两种方式，第一种是直接双击保存的那个 VBS 文件即可（在 Window 的环境下，它会关联 Wscript.exe 全程是 'Microsoft Windows Script Host' 它是 Window 提供的脚本关联自动执行程序），但这种方法要想实现指定时段执行脚本比较困难，因为 VBS 不能直接被 window 执行。还得人工干预。第二种方式是借助 Wscript 命令将 VBS 文件封装成 bat 文件。这样一来 Window 就可以设置何时执行 bat 文件了。那么如何封装 VBS 使其成为 Bat 文件呢？先看一段代码 –

```
On Error Resume Next
Sub ATSub
   Echo off & Cls
   Echo Adding Batching Code here following VBS rules & pause
   Start wscript \\E:vbs "…&%x…"
   Exit Sub
End Sub
MsgBox "Test and see if the VBS can be executed here"
```

把这段代码保存为 bat 文件。如果这段代码能执行，就说明 Wscript 脚本自动执行程序伪装成功。可以顺利执行 VBS 文件内容。那么接来下就好办了。借用 WTS（Window Task Sheduler）就可以实现指定时段的脚本自动化执行了。我这里给出一个我的实例。因为牵涉到具体的脚本，我只给出实现过程就好。脚本那一部分无需加入。

1.> VBS 文件

```
On Error Resume Next
'loadandRunaction "C:\Backup_PreviousYears\FY14-One\forUFTpractice\Customize_ExcelReport_another2", "Action1", OneIteration ' 装载并执行脚本
Set qtApp = CreateObject("QuickTest.Application")
qtApp.Launch 'Launch UFT
qtApp.Visible = True ' Make UFT visible
```

```
qtApp.Options.Run.ViewResults = True

qtApp.Open "C:\Backup_PreviousYears\FY14-
One\forUFTpractice\Customize_ExcelReport_another2", False ' 装载脚本

qtApp.Test.Run "", 10, ""    '执行脚本
```

上面的 VBS 为我们定义了要执行的脚本和怎么去执行这个脚本。接下来我们还要定义何时有谁来执行这个脚本，请看这个文件。

2.> Bat 文件

```
On Error Resume Next
start "Automated Execution Script" /Max /High wscript //E:vbs //logo "c:\temp.vbs"
Or
Cmd.exe /c wscript //e:vbs //logo "c:\temp.vbs"
```

这个文件只有这么简单的一行代码，却将 VBS 文件伪装成可执行文件 Bat。也就是说现在启动 Bat 就可以执行 VBS 脚本了。这要归功于 MS-DOS 中的 Start/Cmd 命令，它允许第三方 Command 的执行。

3.> Windows Task Scheduler

到了这一步就差不多大功告成了，你只需要借助 WTS 来设置你的 Bat 文件启动时间和执行行为即可。

怎么样，简单吧？这里要注意两个问题，一个是因为 VBS 文件不能直接被 WTS 执行，所以我们必须要进行 Bat 封装。另外，由于 Bat 是基于 MS-DOS 命令行来的，因此要保证 bat 文件的正确性，就不能将 VBS 代码直接写入其中，否则也是没办法执行的。所以要进行 VBS 的 Bat 伪装。也就是上面 Bat 文件的编写。只有一行代码，却是解决问题的关键。自动化测试有时候就是这么回事。看起来很神奇的表象其实就只有那么几步。搞清楚了，也就没什么神秘可言了。

3.1.9 类的使用

我们都知道 QTP 的默认脚本语言是 VB，VB 也是面向对象的语言架构，既然面向对象，那么一个重要的概念就是类和对象了，对象在实际脚本编写中，我相信每个人都有概念也知道会使用。下面重要介绍下类以及类的使用等相关技巧。

```
Class ClassName()

End Class
```

类的定义很简单，由 Class 关键字引出。如上面定义了一个类名为 ClassName 的类，并且不带参数。

类是面向对象编程的概念。 当一个类要向外部公开一个属性（变量）时有两种做法：

1.）可直截了当地用 Public 声明这个变量或方法，这样外部过程就可轻而易举地读写这个变量。

2.）用 Private 声明变量，然后用 Public Property Get.../Property Let...过程向外部公开这个属性(只读/只写)。在 Property 过程里，可加入数据合法性验证代码，任何读写这个属性的外部过程都必须通过 Property 过程的合法性验证，这就是所谓的封装，这样的类更加强健。

事实上，即便我们 Public 声明属性，VB 在后台总是用第二种方法处理这个属性。当这个变量是一个对象变量（Object）时，用 Property Set...过程。

文字说明太抽象了，来段代码吧 –

```
'Property Set
Class File
   Private m_fso
   Public Property Set fso(para_fso)
      m_fso = pata_fso
```

```
    End Property
End Class

Set fso = CreateObject("scripting.filesystemobject")
Set objFile = New File
'必须加上 Set，否则报错
Set objfile.fso = fso

'Property Let
Class File1
    Private m_fso
    Public Property Let fso(para_fso)
        m_fso = pata_fso
    End Property
End Class

Dim fso
Set fso = CreateObject("scripting.filesystemobject")
Set objFile = New File1
'不能加上 Set，否则报错
objfile.fso = fso
```

上面两个例子，阐述了 Set 和 Let 的用法，一个用来给对对象赋值，一个用来给变量赋值

另一种解释为： **Property Get** 是获取属性值，**Property Let** 是给属性赋值在 VB 6.0 中有两种类型的属性，即用向类模块中添加 **Public** 变量的方法定义的一般属性和用向类模块中添加 **Property GET/LET/SET** 过程的方法定义的属性过程， 一般使用属性过程。

属性过程分只读（**Property GET**）、只写（**Property LET/SET**）、读写（**Property GET** 和 **LET/SET**）属性过程三种， **SET** 用于定义对象类型属性过程，而 **LET** 用于定义非对象类型属性过程。

GET/LET/SET 的含义是：当读取/使用（即 Access 存取）该属性值时，执行 Property Get 中的代码； 当写入/修改（即 Assign 指派）该属性值时，执行 Property Let 或 Set 中的代码。因为在存取或指派属性值时能执行一段代码，我们就可以在属性过程中增加许多代码（如检验代码），这正是属性过程的优点。Get 可以有返回值，Set 和 Let 则不会 有返回值，它们必须带变量，而且它们的变量只是帮助初始化属性值。

上面说的都是定义类的属性，类中除了可以有属性外，还可以有方法，也就是说可以把 Function 和 Sub 方法直接放入类中，供后续用户调用。

Me 也是类中一个比较常用的概念，他代表类本身，但在类定义时还需要应用类中属性或方法时，可以用 Me.**Property**/Me.method

3.1.9.1 类的调用

类的调用又是如何进行的呢？在 QTP、UFT 中，你会发现直接在 Resource 中加载含有 Class 的 VBS 文件是无法使 QTP、UFT 识别类的，也不能直接使用类中的方法和属性。怎么办呢？

有四种解决办法可供参考，请大家务必熟悉 –

1.）直接在 Action 脚本编写区中定义类，这样就可以直接调用类的方法和属性了。

2.）在 Resource 中关联 VBS 文件，但需要在 VBS 文件中事先实例化这个类，下面两种实例化方式都可以。

```
Dim Class1 : Set Class1 = New SearchInExcel
```

或者

```
Function Class1
```

```
    Set Class1 = new SearchInExcel
End Function
```

如此一来，就可以直接在脚本编辑区中通过 Class1 来调用类的方法与属性了。也可以在脚本中再重新一个实例变量，如 ***Set Class2 = Class1*** 当然，这完全是多此一举。

3.）用 Executefile 导入含类的 VBS 文件，然后在脚本编辑区的任何地方都可以实例化类。不必把实例化语句写在 VBS 文件里。但要注意，vbs/qfl 文件的类型必须是 ANSI 的。否写会报 Invalid Character 错误。

4.）用 LoadFunctionLibrary 导入 VBS 文件，此方法和 2 中描述的原理一样，需要在 VBS 中事先实例化类。不可将实例化语句写在脚本编辑区。因为使用 loadFunctionLibrary 函数加载的 VBS 文件是不识别类的。但却可识别对象变量和函数返回值。这也是此方法和 Executefile 的一个明显区别。此方法不能方便地调出类中所包含的属性和方法。

思考：什么时候需要用到类呢？使用类有什么好处？

这里我简要阐述下，类的作用和使用时机。尽管前面说了不少关于类的规则和使用方式。但具体在什么情况下，为什么要使用？这个问题恐怕还是困扰着不少读者。个人认为，类的使用可以大大简化我们的代码重复量。使脚本的结构更加的严谨，给脚本的维护带来很多方便。为什么这么说呢？

首先，使用类可以使得我们的常用方法易于调用，把方法封装进类后，我们只需像记住类名，即可调用全部方法。

其次，将对象用描述性语言定义后封装成类的属性，也可以使我们省去对象维护的麻烦和关联。在代码编写中也无需重复对象的层级结构，直接调用累的相关属性即可。

再次，将函数和对象封装成类后，我们对脚本的管理就变成了对单纯的类函数的管理了。由函数驱动的脚本，可以抽象出更高的表达。也就是说脚本可以进行多次

的封装，直到方便最终用户使用。关于类的具体使用，我还会在下面的介绍中逐步加入新的内容。以帮助读者加深印象。

3.1.10 Instr/InstrRev/Array/Dim/ReDim/Len/Left/Right

这些函数也是平时编写脚本时使用频率较高的函数，了解和熟悉它们的用法，对提高脚本的编写效率非常有帮助，这里我就不从语法上一一讲解了，直接给出几个实例，更有助于理解和消化。

1.>Instr(start, string1, string2, *compare*)

```
a = Instr(5, "CanyouCanacanasaCanercancanaCan", "can", 1)
'a = 7
a = Instr(5, "CanyouCanacanasaCanercancanaCan", "can", 0)
'a = 11
```

此函数用于查找 String2 在 String1 中出现的位置，compare 为 “0” 意味着查找要区分大小写（即 “二进制” 比较），为 “1” 则是不区分大小写（即 “文本” 比较）。Start 就是开始查找的起始位置。后面的字符位置是顺延的，并不是从开始查找的起始位置重新计数的。而且 star 的最大值不能超出 string1 的长度。不然搜索结果为 0.

2.>InstrRev(string1, string2, *start, compare*)

```
b = InStrRev("CanyouCanacanasaCanercancanaCanabc", "can", 3, 0)
'b = 0
b = InStrRev("CanyouCanacanasaCanercancanaCanabc", "can", 9, 0)
'b = 0
b = InStrRev("CanyouCanacanasaCanercancanaCanabc", "can", 13, 0)
'b = 11
b = InStrRev("CanyouCanacanasaCanercancanaCanabc", "can", 3, 1)
'b = 1
```

此函数也是查找一个字符串在另一个字符串中出现的位置，只不过是从指定的字符长度里查找最后一个出现的位置而已（这里 star 参数就是限定搜索的字符串长度）。它与 Instr 函数功能的区别是，一个从给定的字符串长度里查到第一次出现给

定字符串的位置；另一个是从给定的字符串长度里查到最后一个出现给定字符串的位置。其他参数说明都一样。

这两个函数通常用在什么地方呢？想想看，我们什么时候需要用到一个字符串在另一个字符串中出现的位置？在 QTP 中使用该函数最多的时候就是查找一个路径中某个文件或文件夹的位置，这样我们就可以得到基于这个文件或文件夹的上级完整路径了。这个应该不难理解，后面我会再给一个实例佐证的。

3.>Array/Dim/Redim/LBound/UBound

这几个函数太常见了（准确的说应该是关键字），相信大家也经常使用，但不见得都知道它的正确用法，通常在定义数组变量的时候我们会使用到这些个关键字。大致说一下，Array 定义一个数组变量，Dim 定义一个任何类型的变量，Redim 重定义某一个数组变量。 如：

```
Dim a, b(), c
a = Array(1, 2, 3, 4)
Set c = CreateObject("Excel.Application")

ReDim b(2)
b(2) = a(2)

ReDim Preserve b(3)
b(3) = a(3)
```

此时 b(2) 的值还是 3，也就是 b(2)的值仍将得以保留。

Redim 主要用于数组变量的重定义，就是可以随时改变数组下标的大小，满足程序要求。使用时要注意的是，用 Array 定义的数组是直接赋给不带带下标的变量，被赋值的变量就是一个数组变量。这点尤其要注意。下标只用于数组的取值运算。LBound 和 UBound 函数常用来度量一个数组的空间大小。可以作为 For 循环的起至参数。

4.> **Len/Left/Right**

```
a = "Abcd1234"
```

```
a1 = Len(a)
'a1 = 8

b = left(a, 6)
'b = "Abcd"

b = right(a, 4)
'b = "1234"
```

这一小节所介绍的函数使用频率较高，通常都是搭配起来使用的。多数用于获取路径，以及检查点的设计上。下面的例子集中说明这些函数是如何搭配使用的。也是 QTP 的常见设计思路。

请看这个实例 –
取出 Scripts 文件夹的根目录路径：

```
Dim a1, b1, c1
a1 = "C:\Projectabc\AutomationFolder\Scripts"
b1 = "Scripts"

c1 = Left(a1, InstrRev(a1, b1, -1, 1) - 1)
print c1
' C:\Projectabc\AutomationFolder\

a1 = Array("value1", "value2", "value3", "value4", "value5", "value6")
For i = LBound(a1) To UBound(a1)
   print a1(i)
Next
```

3.1.11 转义字符

什么是转义字符？为什么要把转义字符单独提出来讲？如果你用过描述性方法定义对象的话，一定不会陌生。有些对象的属性和值常常含有 ‘, “, [,], <, > 等特殊字符。如果直接将它们作为一般字符或字符串处理。常常会出现找不到对象，或对象不能识别等问题。

为什么呢？就是因为程序不能将这些特殊字符正确的识别出来。所以我们要借助转义字符来表达了。\ 是一个转义字符。跟在它后面的第一个字符将变得没有意义或特殊意义。使用转义字符主要是为了输入一些不方便直接用键盘输入的字符。其实任何字符都可以转义获得。那么对于有些属性中的特殊字符怎么来转义表达呢？比喻，有一个 SAPGuiButton 的对象，他的描述性表达是：

.SAPGuiButton(“name:=btn\[0\]”, “guicomponettype:=40”)； 这里如果不用转义字符而直接写成 .SAPGuiButton(“name:=btn[0]”, “guicomponettype:=40”)行不行呢？可以看下调试的结果：

第一种写法，可以正确识别对象，通过保存截图（CaptureBitmap）可以简单的判断是否是正确的识别了对象；第二种写法就报错了，提示不能找到 SAPGuiBotton。

3.1.12 环境变量

环境变量在自动化测试脚本中的使用也是很普遍的现象，其本质也就是变量的定义和使用。只不过使用环境变量后，可以更简洁更有效的查看变量表。从而方便用户的使用。

通常定义环境变量有两种方式，一种是内部定义（Internal）一种是外部定义（External）。在 QTP/UFT 的 Environment 设置窗口中创建所需的变量或者通过直接变量赋值的方法定义的变量都是内部环境变量。如下面创建的变量均为内部环境变量。

Environment（）= “”

3.1.13 输入输出函数

Inputbox 和 Print 函数是 VB 的常用输入输出函数。它们通常用于给变量赋值和输出变量的内容。用法很简单，但使用频率却很高。必须掌握。另外，VB 中信息的显示函数 Msgbox 也是随处可见的。请一并熟悉。

3.2 场景参考

3.2.1 做自己的 Excel 测试报告

这个问题在上面的章节中有个类似的介绍，不过那只是针对特定的场景应用，这里我介绍一个通用的 Excel Report template 供大家参考。先来阐述下这个 Report 的生成思路 –

Step1 : Prepare your test Data → **Step2** : import your test data into UFT → **Step3** : Execute Scripts and fillful test Data → **Step4** : Recall my 'excelReportClass' to generate final excel report

注意，在 Step1 中的 testData 必须包含这么三个 table （Input，Output，CheckPoint）因为后面的 ExcelClass 就是基于这三个 table 来定义的格式的。

另外，在脚本执行中，必须要有数据来更新着三个 table，也就是说 Input 中的数据被脚本读取后，脚本要能自动 fillful output 和 Checkpoint 表单。如何填充 output 和 checkpoint 表单需要根据具体的 case，在编写脚本的过程中实现的。这也是为什么我们最红的 Excel report 是不受具体 Case 限制的原因所在。

除此之外，在这三个表单中，有些字段是必须固定好的（当然，你也可以改变这些字段的名称，个人觉得没必要），请看下面的 Sample：

表一 Input Table Define :

Flag	Customer_Name	Customer_ID	Customer_Addr	City	PostCode	……

在 Input 表中，Flag 字段名是固定的，他的取值是 Y/N，当为 Y 是，该行记录被执行，为 N 时，该行记录被忽略掉。其他字段，用户均可自定义名称和个数。

表二 Output Table Define：

Customer_ID	Create_Date	Creator	Region	Experiod	……

在 Output 表中，没有任何预先定义的字段，用户可以根据实际需要自定义字段名称和个数。

表三 Checkpoint Table：

CheckItem_1	CheckItem_2	CheckItem_3	CheckItem_4	……	FinalResult

在 Checkpoint 表中，我们也无需预定义任何字段，用户可以自定义字段名称和个数，但需要在最后一个字段给予 Passed or Failed 的确认，因此，最后一个字段通常定义为 FianlResult

在定义好 DataTable 后，接下来就是脚本的实现了。这是跟具体 case 相关的过程，测试人员需要根据具体的 case 编写脚本，并完成 DataTable 中 Output，Checkpoint 表单的更新。Input 通常在脚本开始时，事先设定好的。完成脚本后，就可以调用这个 ExcelClass 来完成自定义 report 了。下面我将类的详细实现 Post 在下面，调用的时候，我们只需按顺序完成从方法 1 到方法 9 的调用。

```
Class CustomizeExecIReport
    'Declare Property - arguent
    Private  reportPath

    ' Declare Property - methods/Function/Sub
    Public Function CreateExcelReportForSpecifiedCase_1(reportPath)
        Dim fso, excelObj, sheetCount

        KillSpecifiedProcess("EXCEL*")
        Set fso = CreateObject("Scripting.FileSystemObject")
        Set excelObj = CreateObject("Excel.Application")

        If Not fso.FileExists(reportPath) Then
            excelObj.Workbooks.Add.Saveas reportPath
            sheetCount = excelObj.Worksheets.Count
```

```
        If sheetCount > 1 Then
            Do
                excelObj.Worksheets(sheetCount).Delete
                sheetCount = sheetCount - 1
            Loop While sheetCount > 1
        End If
        excelObj.Worksheets.Item(sheetCount).Name = "Execution_Times_01"
        excelObj.Sheets.Item(sheetCount).Activate
    Else
        excelObj.Workbooks.Open reportPath
        sheetCount = excelObj.Worksheets.Count
        excelObj.Sheets.Add.Name = "Execution_Times_0" & sheetCount+1
        excelObj.Sheets.Item(1).Activate
    End If

    excelObj.Worksheets.Item(1).Cells.Font.Name = "Gill Sans MT"
    excelObj.Worksheets.Item(1).Cells.Font.FontStyle = "Normal"
    excelObj.Worksheets.Item(1).Cells.Font.Size = 12
    excelObj.DisplayAlerts = False
    'excelObj.Workbooks.Item(1).Save
    excelObj.Workbooks.Item(1).Save

    excelObj.Quit
    Set fso = Nothing
    Set excelObj = Nothing
End Function

Public Function DefineReportTitle_2(reportPath, businessSummary)
    Dim excelObj, selection
    Set excelObj = CreateObject("Excel.Application")
    excelObj.Workbooks.Open reportPath
```

```
    Set selection = excelObj.Worksheets.Item(1)
    With selection
        .Cells(1, 1).Font.Name = "Gill Sans MT"
        .Cells(1, 1).Font.FontStyle = "Bold"
        .Cells(1, 1).Font.Size = 16
        .Cells(1, 1).Font.Color = vbBlack
        .Cells(1, 1).Value = "Summary Introduce for this Case or Script - "

        .Cells(3, 1).Font.ColorIndex = 23
        .Cells(3, 1).Font.Size = 13
        .Cells(3, 1).Font.FontStyle = "Italic"
        .Cells(3, 1).Value = businessSummary
    End With

    excelObj.Worksheets.Item(1).Rows(2).RowHeight = 9

    excelObj.DisplayAlerts = False
    excelObj.Workbooks.Item(1).Save

    excelObj.Quit
    Set selection = Nothing
    Set excelObj = Nothing
End Function

Public Sub DefineReportBaseInformationSection_3(reportPath)
    Dim sheetCount, selection
    Set excelObj = CreateObject("Excel.Application")

    excelObj.Workbooks.Open reportPath
    usedRow = excelObj.Worksheets.Item(1).UsedRange.Rows.Count
```

```
    Set selection = excelObj.Sheets.Item(1)
    With selection
        .Cells(usedRow+2, 1).Font.Name = "Gill Sans MT"
        .Cells(usedRow+2, 1).Font.FontStyle = "Bold"
        .Cells(usedRow+2, 1).Font.Size = 15
        '.Cells(usedRow+2, 1).Font.Color = vbBlue
    End With

    selection.Cells(usedRow+2, 1).Value = "Test Report Part 1 - Basic Information"
    'selection.Range("A1:K1").Merge
    excelObj.Worksheets.Item(1).Range("A" & usedRow+2 & ":" & "B" & usedRow &_
    +2).Borders(4).LineStyle = 6
    excelObj.Worksheets.Item(1).Rows(usedRow+3).RowHeight = 9

    excelObj.DisplayAlerts = False
    excelObj.Workbooks.Item(1).Save

    excelObj.Worksheets.Item(1).Range("A" & usedRow+2 & ":" & "B" & usedRow &_
    +2).Borders(4).LineStyle = 6 '9 means double full line, 6 means dashed line

    excelObj.Quit
    Set selection = Nothing
    Set excelObj = Nothing
End Sub

Public Function IntilializeBaseInformation_4(byRef reportPath)
    Dim usedRow, selection

    Set excelObj = CreateObject("Excel.Application")
    excelObj.Workbooks.Open reportPath
```

```
usedRow = excelObj.Worksheets.Item(1).UsedRange.Rows.Count
IntilializeBaseInformation_4 = usedRow

Set selection = excelObj.Worksheets.Item(1)
With selection
   '.Range("A" & row+4 & ":A"& row+9).Borders.Color = vbBlue
   .Range("A" & usedRow+1 & ":B"& usedRow+6).Borders.LineStyle = 1 ' o means
    no border/1 means add border
   .Range("A" & usedRow+1 & ":B"& usedRow+6).Borders(7).Weight = 4
   .Range("A" & usedRow+1 & ":B"& usedRow+6).Borders(8).Weight = 4
   .Range("A" & usedRow+1 & ":B"& usedRow+6).Borders(9).Weight = 4
   .Range("A" & usedRow+1 & ":B"& usedRow+6).Borders(10).Weight = 4

   .Range("A" & usedRow+1, "A" & usedRow+6).Font.FontStyle = "Normal"
   .Range("B" & usedRow+1, "B" & usedRow+6).Font.FontStyle = "Italic"
   .Range("A" & usedRow+1 & ":B"& usedRow+6).HorizontalAlignment = 2 ' 2 :
   left '.Range("A:X").HorizontalAlignment = 2 ' 2 : left    from column A to X will align by
   left ' refine this again at the checkpoint funtion
   .Range("A" & usedRow+1 & ":A"& usedRow+6).Interior.ColorIndex = 15


   'Define Basic Information Type
   .Cells(usedRow + 1, 1).Value = "Test Case/Script Name : "
   .Cells(usedRow + 2, 1).Value = "Test Data : "
   .Cells(usedRow + 3, 1).Value = "Test Start Time : "
   .Cells(usedRow + 4, 1).Value = "Test End Time : "
   .Cells(usedRow + 5, 1).Value = "Duration : "
   .Cells(usedRow + 6, 1).Value = "Test Executor : "


'          ' initializate the Basic Information
```

```
'           .Cells(usedRow + 1, 2).Value = caseName
'           .Cells(usedRow + 2, 2).Value = testDate
'           .Cells(usedRow + 3, 2).Value = startTime
'           .Cells(usedRow + 4, 2).Value = endTime
'           .Cells(usedRow + 5, 2).Value = during
'           .Cells(usedRow + 6, 2).Value = executor
        End With

        excelObj.DisplayAlerts = False
        excelObj.Workbooks.Item(1).Save

        excelObj.Quit
        Set selection = Nothing
        Set excelObj = Nothing
    End Function

    Public Function  DefineInputSection_5(reportPath)
        Dim excelObj, usedRow, sheetCount, counter, getItem, inputColumnsCount,
selection, inputRowCount

        Set excelObj = CreateObject("Excel.Application")
        excelObj.Workbooks.Open reportPath
        sheetCount = excelObj.Worksheets.Count
        usedRow = excelObj.Worksheets.Item(1).UsedRange.Rows.Count
        counter = 0

        inputColumnsCount = DataTable.GetSheet("Input").GetParameterCount
        getItem = GetAlphabetFromDictionary(inputColumnsCount)

        Set selection = excelObj.Worksheets.Item(1).Range("A" & usedRow+2) 'NOte:
```

```
Range usage
With selection
    .Font.Size = 15
    .Font.FontStyle = "Bold"
End With
Set selection = Nothing

excelObj.Sheets.Item(1).Cells(usedRow+2, 1).Value = "Test Report Part II  - " &_
Input Parameters "
excelObj.Sheets.Item(1).Range("A" & usedRow+2,
getItem & usedRow+2).Borders(4).LineStyle = 6
excelObj.Worksheets.Item(1).Rows(usedRow+3).RowHeight = 9


excelObj.Sheets.Item(1).Range("A" & usedRow+4, &_
getItem & usedRow+4).Interior.Color = vbGreen
'excelObj.Sheets.Item(1).Range("A" & usedRow+4 & ":" & getItem &
usedRow+4).Interior.Color = vbRed

For i = 1 To inputColumnsCount
    excelObj.Sheets.Item(1).Cells(usedRow+4, &_
    i).Value = Datatable.GetSheet("Input").GetParameter(i).Name
Next

inputRowCount = DataTable.GetSheet("Input").GetRowCount
For i = 1 To inputRowCount
    DataTable.SetCurrentRow(i)
    If DataTable.Value("Flag", "Input") = "Y" Then
        counter = counter + 1
        For j = 1 To inputColumnsCount
            excelObj.Sheets.Item(1).Cells(usedRow+4+counter, &_
```

```
            j).Value = DataTable.Value(j, "Input")
        Next
    End If
Next

excelObj.Sheets.Item(1).Range("A" & usedRow+4, &_
getItem & usedRow+4+counter).Borders.LineStyle = 1

excelObj.Sheets.Item(1).Range("A" & usedRow+4, &_
getItem & usedRow+4+counter).Borders(7).Weight = 4
excelObj.Sheets.Item(1).Range("A" & usedRow+4, &_
getItem & usedRow+4+counter).Borders(8).Weight = 4
excelObj.Sheets.Item(1).Range("A" & usedRow+4, &_
getItem & usedRow+4+counter).Borders(9).Weight = 4
excelObj.Sheets.Item(1).Range("A" & usedRow+4, &_
getItem & usedRow+4+counter).Borders(10).Weight = 4

excelObj.Sheets.Item(1).Range("A" & usedRow+2, &_
getItem & usedRow+2).Borders(4).LineStyle = 6

excelObj.DisplayAlerts = False
excelObj.Workbooks.Item(1).Save
excelObj.Quit
Set excelObj = Nothing
End Function

Public Function  DefineOutputSection_6(reportPath)
Dim sheetCount, usedRow, outputColumnsCount, conter, getItem, selection,
Dim excelObj

Set excelObj = CreateObject("Excel.Application")
```

```
excelObj.Workbooks.Open reportPath

sheetCount = excelObj.Worksheets.Count
usedRow = excelObj.Worksheets.Item(1).UsedRange.Rows.Count
outputColumnsCount = DataTable.GetSheet("Output").GetParameterCount
getItem = GetAlphabetFromDictionary(outputColumnsCount)
counter = 0

Set selection = excelObj.Worksheets.Item(1).Range("A" & usedRow+2) 'please use
Range not Cells.
With selection
   .Font.FontStyle = "Bold"
   .Font.Size = 15
End With
Set selection = Nothing

excelObj.Worksheets.Item(1).Cells(usedRow+2, 1).Value = "Test Report " &_
"Part III - Output during the execution"
excelObj.Sheets.Item(1).Range("A" & usedRow+2, &_
getItem & usedRow+2).Borders(4).LineStyle = 6
excelObj.Worksheets.Item(1).Rows(usedRow+3).RowHeight = 9

excelObj.Worksheets.Item(1).Range("A" & usedRow+4, &_
getItem & usedRow+4).Interior.ColorIndex = 28

For i = 1 To outputColumnsCount
   excelObj.Worksheets.Item(1).Cells(usedRow+4, &_
   i).Value = DataTable.GetSheet("Output").GetParameter(i).Name
Next
```

```
outputRowCount = DataTable.GetSheet("Output").GetRowCount
For i = 1 To outputRowCount
   DataTable.SetCurrentRow(i)

   For j = 1 To outputColumnsCount
      If DataTable.Value(j, "Output") <> "" Then
         counter = counter + 1
         For q = 1 To outputColumnsCount
            excelObj.Sheets.Item(1).Cells(usedRow+4+counter, &_
            q).Value = DataTable.Value(q, "Output")
         Next
         Exit For
      End If
   Next
Next

'add double solid line
excelObj.Sheets.Item(1).Range("A" & usedRow+4, &_
getItem & usedRow+4+counter).Borders.LineStyle = 1
excelObj.Sheets.Item(1).Range("A" & usedRow+4, &_
getItem & usedRow+4+counter).Borders(7).Weight = 4
excelObj.Sheets.Item(1).Range("A" & usedRow+4, &_
getItem & usedRow+4+counter).Borders(8).Weight = 4
excelObj.Sheets.Item(1).Range("A" & usedRow+4, &_
getItem & usedRow+4+counter).Borders(9).Weight = 4
excelObj.Sheets.Item(1).Range("A" & usedRow+4,&_
getItem & usedRow+4+counter).Borders(10).Weight = 4

excelObj.Sheets.Item(1).Range("A" & usedRow+2, &_
getItem & usedRow+2).Borders(4).LineStyle = 6
```

```
    excelObj.DisplayAlerts = False
    excelObj.Workbooks.Item(1).Save
    excelObj.Quit
    Set excelObj = Nothing
End Function

Public Function  DefineCheckpointSection_7(reportPath)
    Dim sheetCount, usedRow, outputColumnsCount, conter, getItem, selection,
    Dim excelObj, usedColumn

    Set excelObj = CreateObject("Excel.Application")
    excelObj.Workbooks.Open reportPath

    sheetCount = excelObj.Worksheets.Count
    usedRow = excelObj.Worksheets.Item(1).UsedRange.Rows.Count
    outputColumnsCount = DataTable.GetSheet("Checkpoint").GetParameterCount
    getItem = GetAlphabetFromDictionary(outputColumnsCount)
    counter = 0

    Set selection = excelObj.Worksheets.Item(1).Range("A" & usedRow+2)
    With selection
        .Font.FontStyle = "Bold"
        .Font.Size = 15
    End With
    Set selection = Nothing

    excelObj.Worksheets.Item(1).Cells(usedRow+2, 1).Value = "Test Report " &_
    "Part IV - Get Checkpoints"
    excelObj.Sheets.Item(1).Range("A" & usedRow+2, &_
    getItem & usedRow+2).Borders(4).LineStyle = 6
    excelObj.Worksheets.Item(1).Rows(usedRow+3).RowHeight = 9
```

```
excelObj.Worksheets.Item(1).Range("A" & usedRow+4, &_
getItem & usedRow+4).Interior.ColorIndex = 24

For i = 1 To outputColumnsCount
    excelObj.Worksheets.Item(1).Cells(usedRow+4, &_
    i).Value = DataTable.GetSheet("Checkpoint").GetParameter(i).Name
Next


outputRowCount = DataTable.GetSheet("Checkpoint").GetRowCount
For i = 1 To outputRowCount
    DataTable.SetCurrentRow(i)

    For j = 1 To outputColumnsCount
        If DataTable.Value(j, "Checkpoint") <> "" Then
            counter = counter + 1
            For q = 1 To outputColumnsCount
                excelObj.Sheets.Item(1).Cells(usedRow+4+counter, &_
                q).Value = DataTable.Value(q, "Checkpoint")
            Next
            Exit For
        End If
    Next
Next

excelObj.Sheets.Item(1).Range("A" & usedRow+4, &_
getItem & usedRow+4+counter).Borders.LineStyle = 1
excelObj.Sheets.Item(1).Range("A" & usedRow+4, &_
getItem & usedRow+4+counter).Borders(7).Weight = 4
excelObj.Sheets.Item(1).Range("A" & usedRow+4, &_
```

```
getItem & usedRow+4+counter).Borders(8).Weight = 4
excelObj.Sheets.Item(1).Range("A" & usedRow+4, &_
getItem & usedRow+4+counter).Borders(9).Weight = 4
excelObj.Sheets.Item(1).Range("A" & usedRow+4, &_
getItem & usedRow+4+counter).Borders(10).Weight = 4

excelObj.Sheets.Item(1).Range("A" & usedRow+2, &_
getItem & usedRow+2).Borders(4).LineStyle = 6

For i = 1 To outputRowCount
    If excelObj.Sheets.Item(1).Cells(usedRow+4+i, &_
    outputColumnsCount).Value = "FAILED" or excelObj.Sheets.Item(1). &_
    Cells(usedRow+4+i, outputColumnsCount).Value = "Failed" or excelObj. &_
    Sheets.Item(1).Cells(usedRow+4+i, outputColumnsCount).Value = "failed" &_
Then
    excelObj.Sheets.Item(1).Cells(usedRow+4+i, &_
    outputColumnsCount).Interior.Color = vbRed
        ElseIf excelObj.Sheets.Item(1).Cells(usedRow+4+i, &_
        outputColumnsCount).Value = "PASSED" or excelObj.Sheets.Item(1). &_
        Cells(usedRow+4+i, outputColumnsCount).Value = "Passed" or excelObj. &_
        Sheets.Item(1).Cells(usedRow+4+i, outputColumnsCount). &_
        Value = "passed" Then
            excelObj.Sheets.Item(1).Cells(usedRow+4+i, &_
            outputColumnsCount).Interior.Color = vbGreen
    End If
Next

excelObj.DisplayAlerts = False
excelObj.Workbooks.Item(1).Save
excelObj.Quit
Set excelObj = Nothing
```

```
End Function


Public Function DefineTheEndOfReport_8(reportPath)
    Dim excelObj, getItem, usedRow, usedColumn, selection
    Set excelObj = CreateObject("Excel.Application")

    excelObj.Workbooks.Open reportPath
    usedColumn = excelObj.Worksheets.Item(1).UsedRange.Columns.Count
    usedRow = excelObj.Worksheets.Item(1).UsedRange.Rows.Count

    getItem = GetAlphabetFromDictionary(usedColumn)


    excelObj.Sheets.Item(1).Range("A1" & ":" & getItem & 1).Borders(4).Weight = 4
    excelObj.Worksheets.Item(1).Range("A" & ":" & getItem).HorizontalAlignment = 2

    Set selection = excelObj.Worksheets.Item(1)
    With selection
        .Cells(usedRow+2, 1).Font.Size = 16
        .Cells(usedRow+2, 1).Font.FontStyle = "Bold"
        .Range("A" & usedRow+3, "A" & usedRow+6).Font.Color = vbBlue
        .Range("A" & usedRow+2 & ":" & getItem & usedRow+2).Borders(4).LineStyle =
1

        .Cells(usedRow+2, 1).Value = "Additional Remarked - "
        .Rows(usedRow+3).RowHeight = 9

        .Cells(usedRow+4, 1).Value = "Result Statement - See the 'Result column at the
         last part, if the checkpoint shown PASSED, then its backgroud would be fulfilled
```

```
            with Green, or it remarked with Red that means the checkpoint is Failed"
            .Cells(usedRow+5, 1).Value = "To judge this test or execution Passed only when
            all the checkpoints were shown PASSED/Green."
            .Cells(usedRow+6, 1).Value = "Oppositely, As long as anyone checkpoint fulfilled
            with Red, we can consider this test/execution 'FAILED', You need to go back and
            check your scripts to find out what caused the checkpoint fail."
        End With

'       excelObj.Cells.EntireColumn.AutoFit 'adjust columns width automatically
'       'excelObj.Columns("A:A").ColumnWidth = 30
'       'excelObj.Columns("B:B").ColumnWidth = 25
'       excelObj.Sheets.Item(1).Columns("A:A").Insert
'       excelObj.Sheets.Item(1).Columns("A:A").ColumnWidth = 5

        excelObj.ActiveWindow.DisplayGridlines = False
        excelObj.ActiveWindow.DisplayHeadings = False
        excelObj.ActiveWindow.DisplayFormulas = False
        excelObj.ActiveWindow.DisplayWhitespace = False
        excelObj.ActiveWindow.DisplayRuler = False

        excelObj.Sheets.Item(1).Rows(1).RowHeight = 40
        excelObj.DisplayAlerts = False
        excelObj.Workbooks.Item(1).Save

        excelObj.Quit
        Set selection = Nothing
        Set excelObj = Nothing
        'KillSpecifiedProcess("EXCEL*")
    End Function

    Function AssignBasicInformation_9(byRef reportPath, byVal usedRow, caseName,
```

```
testDate, startTime, endTime, duration, executor)
    Dim selection

    Set excelObj = CreateObject("Excel.Application")
    excelObj.Workbooks.Open reportPath

    Set selection = excelObj.Worksheets.Item(1)

    With selection
        .Cells(usedRow + 1, 2).Value = caseName
        .Cells(usedRow + 2, 2).Value = testDate
        .Cells(usedRow + 3, 2).Value = startTime
        .Cells(usedRow + 4, 2).Value = endTime
        .Cells(usedRow + 5, 2).Value = duration
        .Cells(usedRow + 6, 2).Value = executor
    End With

    excelObj.Cells.EntireColumn.AutoFit 'adjust columns width automatically
    excelObj.Columns("A:A").ColumnWidth = 30
    'excelObj.Columns("B:B").ColumnWidth = 25
    excelObj.Sheets.Item(1).Columns("A:A").Insert
    excelObj.Sheets.Item(1).Columns("A:A").ColumnWidth = 5

    excelObj.DisplayAlerts = False
    excelObj.Workbooks.Item(1).Save

    excelObj.Quit
    Set selection = Nothing
    Set excelObj = Nothing
    KillSpecifiedProcess("EXCEL*")
End Function
```

```
Private Function GetAlphabetFromDictionary(key)
    Dim d
    Set d = CreateObject("Scripting.Dictionary")
    d.Add 1, "A" : d.Add 2, "B" : d.Add 3, "C" : d.Add 4, "D" : d.Add 5, "E" :
    d.Add 6, "F" : d.Add 7, "G" : d.Add 8, "H" : d.Add 9, "I" : _
    d.Add 10, "J" : d.Add 11, "K" : d.Add 12, "L" : d.Add 13, "M" : d.Add 14, "N" : _
    d.Add 15, "O" : d.Add 16, "P" : d.Add 17, "Q" : d.Add 18, "R" : d.Add 19, "S" :
    d.Add 20, "T" : _
    d.Add 21, "U" : d.Add 22, "V" : d.Add 23, "W" : d.Add 24, "X" : d.Add 25, "Y" :
    d.Add 26, "Z"

    GetAlphabetFromDictionary = d.Item(key)
    Set d = Nothing
End Function

Property Get LastTime(startTime, endTime)
    Dim h, n, s

    h = DateDiff("h", startTime, endTime)
    n = DateDiff("n", startTime, endTime)
    If n > 60 Then
        n = n mod 60
    End If
    s = DateDiff("s", startTime, endTime)
    If s > 60 Then
        s = s mod 60
    End If

    LastTime = h &" Hours, " & n & " Minutes, " & s & " Seconds."
End Property
```

```
    Property Get StartTime
        StartTime = now()
    End Property

    Property Get EndTime
        EndTime = now()
    End Property

    Private Sub KillSpecifiedProcess(killProcess)
        Set wsShell = CreateObject("Wscript.Shell")
        wsShell.Run "TaskKill /F /FI " & """imagename eq " & killProcess & """", 0, True
    End Sub
End Class
```

这个类一共包含 9 个函数，三个属性，基本上是围绕 ExcelApplication 接口来展开的，这个接口中提供了丰富的 Excel 操作，这也是我们为什么能自动生成 ExcelReport 的基础。上面的函数和属性实现代码，仅供参考，有兴趣的同学，可以 copy 下来，细细研究下。里面有很多方法还是比较实用的。实际应用中，你只需要从函数 1 到函数 9 依次调用即可。

3.2.2 从指定的表单中搜索符合要求的字段

在这一部分，我将通过几个实例并结合上述介绍的知识点来详细解说下，如何利用 QTP/UFT 平台实现具体的应用场景。首先来看场景一，在这个场景里，我们要实现 Excel 数据的挑选。也就是在大量的数据表中挑选出符合要求的单元数据来。类似这样的任务是不是在日常工作中经常遇到？搞懂了下面的脚本，也就彻底解决了类似问题。

Scenario: *❶ Search a specified Excel sheet by defined character (from a excel file) → ❷ pick up search result and output it to the specified sheet (another excel file) → ❸ setting the output*

form

```
Class SearchInExcel ' Don't attach the parameter when defining a calss
  Private Function GetAlphabetFromDictionary(key)
    Dim d
    Set d = CreateObject("Scripting.Dictionary")
    d.Add 1, "A" : d.Add 2, "B" : d.Add 3, "C" : d.Add 4, "D" : d.Add 5, "E" :
    d.Add 6, "F" : d.Add 7, "G" : d.Add 8, "H" : d.Add 9, "I" : d.Add 46, "AT" :
    d.Add 47, "AU" : d.Add 48, "AV" : d.Add 49, "AW" : d.Add 50, "Ax" :
    d.Add 51, "AY" : d.Add 52, "AZ" :_
    d.Add 10, "J" : d.Add 11, "K" : d.Add 12, "L" : d.Add 13, "M" :
    d.Add 14, "N" :  d.Add 53, "BA" :  d.Add 54, "BB" :  d.Add 55, "BC" :
    d.Add 56, "BD" :  d.Add 57, "BE" :  d.Add 58, "BF" :  d.Add 59, "BG" :
    d.Add 60, "BH" :  d.Add 61, "BI" :  d.Add 62, "BJ" :_
    d.Add 15, "O" : d.Add 16, "P" : d.Add 17, "Q" : d.Add 18, "R" : d.Add 19, "S" :
    d.Add 20, "T" :  d.Add 63, "BK" : d.Add 64, "BL" : d.Add 65, "BM" : d.Add 66, "BN" :
    d.Add 67, "BO" : d.Add 68, "BP" : d.Add 69, "BQ" : d.Add 70, "BR" : d.Add 71, "BS" :
    d.Add 72, "BT":_
    d.Add 21, "U" : d.Add 22, "V" : d.Add 23, "W" : d.Add 24, "X" : d.Add 25, "Y" :
    d.Add 26, "Z" :  d.Add 27, "AA" : d.Add 28, "AB" : d.Add 29, "AC" : d.Add 30, "AD" :
    d.Add 31, "AE" :  d.Add 73, "BU" : d.Add 74, "BV": d.Add 75, "BW": d.Add 76, "BX":
    d.Add 77, "BY": d.Add 78, "BZ":_
    d.Add 32, "AF" : d.Add 33, "AG" : d.Add 34, "AH" : d.Add 35, "AI" : d.Add 36, "AJ" :
    d.Add 37, "AK" : d.Add 38, "AL" : d.Add 39, "AM" : d.Add 40, "AN" :
    d.Add 41, "AO" :
    d.Add 42, "AP" : d.Add 43, "AQ" : d.Add 44, "AR" : d.Add 45, "AS"
    GetAlphabetFromDictionary = d.Item(key)
    Set d = Nothing
  End Function

  Function KillProcess_Excel(processImage)
```

```
    Set wsShell = CreateObject("Wscript.Shell")
    'wsShell.Run "TaskKill /F /FI " & """imagename eq EXCE*""", 0, True
    wsShell.Run "TaskKill /F /FI ""imagename eq """ & processImage, 0, True
    Set wsShell = Nothing
End Function

Function SearchSpecifiedExcelAndGenerateNewSheet(searchedExcelFile,
        searchedSheet, searchedFromColumn, searchFromRow, searchedByChar,
        maxRow, saveFileName)
    Set objExcel = CreateObject("Excel.Application")
    Set fso = CreateObject("Scripting.FileSystemObject")

    If fso.FileExists(saveFileName) Then
       saveFileOption = "outputfileExisting"
       Else
       saveFileOption = "outputfileNotExisting"
    End If

    ' open first excel - searched file
    objExcel.Workbooks.Open searchedExcelFile
    totalColmns = objExcel.Worksheets.Item(searchedSheet).UsedRange.Columns.Cou
    nt
    totalRows = objExcel.Worksheets.Item(searchedSheet).UsedRange.Rows.Count

    Select Case saveFileOption
       Case "outputfileExisting"
       'open second excel file - test result file
       objExcel.Workbooks.Open saveFileName
       times = objExcel.Workbooks.Item(2).Sheets.Count

       objExcel.Workbooks.Item(2).Sheets.Add.Name = "SearchBy_" & searchedByCha
```

```
 r & "_RunTimes_" & times + 1
objExcel.Workbooks.Item(2).Save
objExcel.Workbooks.Item(2).Close

objExcel.Workbooks.Open saveFileName
Set workSheet1 = objExcel.Workbooks.Item(1).Worksheets.Item(searchedShee
 t)
Set workSheet2 = objExcel.Workbooks.Item(2).Worksheets.Item(1)
msgbox objExcel.Workbooks.Item(1).Sheets.Count
msgbox objExcel.Workbooks.Item(2).Sheets.Count

counterCol = 0
counterRow = 0
For i = searchedFromColumn To totalColmns 'i is pointor of Columns while j is the
pointer of rows
   For j = searchFromRow To totalRows
      judgeMessage = workSheet1.Cells(j, i)

'       msgbox judgeMessage
      Set regEx = New RegExp
      regEx.IgnoreCase = True
      regEx.Global = True

      regEx.Pattern = searchedByChar
      yesNoFlag = regEx.Test(judgeMessage)

'       msgbox yesNoFlag
      If yesNoFlag Then
         counterRow = counterRow + 1
         key = Int(counterCol /maxRow) + 1
         'combineTCName = workSheet1.Cells(j, 2) & "_" & workSheet1.Cells(1, i) &
```

```
            "_" & workSheet1.Cells(j, 1)
            getCells = workSheet1.Cells(j, i)
            workSheet2.Range(GetAlphabetFromDictionary(key) & counterRow).val
            ue = getCells
            counterCol = counterCol + 1
            'counterCol = counterCol + 1 ' Add this sentence, you can output by one
             interval column
            'counterRow = counterRow + 1' Add this sentence, you can output by one
            interval row

            If counterRow = maxRow Then
               workSheet2.Range(GetAlphabetFromDictionary(key) & 1 & ":" & Get
               Alphabet
               FromDictionary(key) & counterRow).Interior.ColorIndex
               = randomNumber(0, 56)
               counterRow = 0
            End If
         End If
      Next
   Next


   Case "outputfileNotExisting"
      objExcel.DisplayAlerts = False
      'objExcel.Workbooks.Add.Save
      objExcel.Workbooks.Add

      Set workSheet1 = objExcel.Workbooks.Item(1).Worksheets.Item(searchedSh
       eet)
      Set workSheet2 = objExcel.Workbooks.Item(2).Worksheets.Item(1)
```

```
counterCol = 0
counterRow = 0
For i = searchedFromColumn To totalColmns 'i is pointor of Columns while j is
the pointer of rows
    For j = searchFromRow To totalRows
        judgeMessage = workSheet1.Cells(j, i)
'       msgbox judgeMessage
        Set regEx = New RegExp
        regEx.IgnoreCase = True
        regEx.Global = True

        regEx.Pattern = searchedByChar
        yesNoFlag = regEx.Test(judgeMessage)
'       msgbox yesNoFlag
        If yesNoFlag Then
            counterRow = counterRow + 1
            key = Int(counterCol /maxRow) + 1
            'combineTCName = workSheet1.Cells(j, 2) & "_" & workSheet1.Cells(1, i)
            & "_" & workSheet1.Cells(j, 1)
            getCells = workSheet1.Cells(j, i)
            workSheet2.Range(GetAlphabetFromDictionary(key) & counterRow).
            value =
            getCells
            counterCol = counterCol + 1
            'counterCol = counterCol + 1 ' Add this sentence, you can output by one
             interval column
            'counterRow = counterRow + 1' Add this sentence, you can output by one
            interval row

            If counterRow = maxRow Then
                workSheet2.Range(GetAlphabetFromDictionary(key) & 1 & ":" & G
```

```
                    etAlphabetFromDictionary(key) & counterRow).Interior.
                    ColorIndex = randomNumber(0, 56)
                    counterRow = 0
                End If
            End If
        Next
    Next

    workSheet2.UsedRange.Font.Name = "Gill Sans MT"
    workSheet2.UsedRange.Font.Size = 12
    workSheet2.UsedRange.Font.Color = vbBlack
    workSheet2.UsedRange.VerticalAlignment = 2
    workSheet2.UsedRange.HorizontalAlignment = 1
    'Alignment, 1=auto | Alignment, 2=left | Alignment, 3=centre |Alignment, 4=right | |
     Alignment, 1=top | Alignment, 2=middle | Alignment, 3=bottom |
    workSheet2.UsedRange.EntireRow.AutoFit
    workSheet2.UsedRange.EntireColumn.AutoFit
    'workSheet2.UsedRange.AutoFit
End Select

workSheet2.UsedRange.Font.Name = "Gill Sans MT"
workSheet2.UsedRange.Font.Size = 12
workSheet2.UsedRange.Font.Color = vbBlack
workSheet2.UsedRange.VerticalAlignment = 2
workSheet2.UsedRange.HorizontalAlignment = 1
'Alignment, 1=auto | Alignment, 2=left | Alignment, 3=centre |Alignment, 4=right | |
Alignment, 1=top | Alignment, 2=middle | Alignment, 3=bottom |
workSheet2.UsedRange.EntireRow.AutoFit
workSheet2.UsedRange.EntireColumn.AutoFit
'workSheet2.UsedRange.AutoFit
```

```
        objExcel.DisplayAlerts = False
        If saveFileOption = "outputfileExisting" Then
            objExcel.Workbooks.Item(2).Save
            ElseIf saveFileOption = "outputfileNotExisting" Then
                workSheet2.SaveAs saveFileName
        End If

        objExcel.Workbooks.Close
        objExcel.Quit
        Set objExcel = Nothing
        Set fso = Nothing
    End Function
End Class
```

在这个场景中，只需调用上述类库中的两个方法即可完成表单的搜索。本质是利用 VB 的正则表达式和调用 Excel 的接口对象完成。这些方法在前面的章节中都有详细介绍，这里是从实战的角度加以应用的。结合前面介绍的内容，读者很容易融会贯通。

3.2.3 从指定的表单中搜索符合要求的字段

前面介绍的场景是模糊搜索的一个典型应用。适用于包含字符串的搜索要求。对那些精确到某一具体值的搜索不是很有效。那么如何自动化指定精度值的表单搜索呢？在上面的章节中我们有提到过 ADODB 对象的用法，借助此对象，我们可以用 SQL 语句来进行有条件的搜索。没错，SQL 语句是强大的搜索应用器。利用 SQL 提供的搜索功能，我们可以精确地挑选出需要的数据。一起来看看在脚本上是如何实现的。

Scenario：*search specified Excel file → design SQL → pick up the records → save these records to a specified Excel file.*

```
Function searchExcelbySQL(searchedExcel, sql, saveSearchResultToFile)
    Set objADO = CreateObject("ADODB.Connection")
    Set objRS = CreateObject("ADODB.Recordset")
```

```
    Set objExcel = CreateObject("Excel.Application")
    'Print "Print the ADODB Connect status, it should be '0' this time >>> " &  objADO.State

    ' Connect the specified Excel file
    objADO.Provider = "Microsoft.Jet.OLEDB.4.0"
    objADO.Properties("Extended Properties").Value = "Excel 8.0; HDR=Yes; IMX=2"
    objADO.Open searchedExcel
    'Print "Print the ADODB Connect status, it should be '1' this time >>> " &  objADO.State

    objRS.Open sql, objADO, 1, 3

    objExcel.Workbooks.Add
    For Iterator = 0 To objRS.Fields.Count - 1
        objExcel.Sheets.Item(1).Cells(1,
Iterator + 1) = objRS.Fields.Item(Iterator).Name '  will print the column name not the record
value
    Next

    rowCount = objExcel.Sheets.Item(1).UsedRange.Rows.Count
    objExcel.Sheets.Item(1).Cells(rowCount + 2, 1).CopyFromRecordset
objADO.Execute(sql)

'    For i = 1 To objRS.RecordCount
'        rowCount = objExcel.Sheets.Item(1).UsedRange.Rows.Count
'        For j = 0 To objRS.Fields.Count - 1
'            objExcel.Sheets.Item(1).Cells(rowCount + 1, j + 1) = objRS.Fields.Item(j).Value ' or use
objRS.Fields.Item(j)
'        Next
'        objRS.MoveNext
'    Next
'
```

```
' 参数 ConnectionString ususlly composed by Provider，Propertiese and Data Course
' ConnectionString = Provier + Properties + DataSource

    objExcel.Workbooks.Item(1).SaveAs saveSearchResultToFile
    objExcel.Quit

    Set objADO = Nothing
    Set objRS = Nothing
    Set objExcel = Nothing
End Function
```

在这个场景中，只需正确地写好 SQL 语句，就能快速的检索出需要的记录。这些记录将另存为一个单独的 Excel 文件里。这里的方法和前面章节 '2.1.3.4 用 ISAM 驱动连接 Excel 数据库' 介绍的内容相似，只不过封装的更为完整，更具适用价值。

3.2.4 全面掌握基于 XML 的自动化编程

这一部分内容在 2.1.1 中有过提及，但当时的介绍主要是基于 WebService 测试用例的自动化脚本所展开的。所涉及的内容不全面。在这一章节里，我将详细介绍下基于 XML 文档的自动化实现，以及它的常用方法和策略。通过这一节的学习，我们应该能够掌握任何基于 XML 文档的自动化应用。

在 QTP/UFT 中，我们需要先创建一个 XML 对象，然后利用其对象封装的属性和方法来操作具体的 XML 文档，下面这些方法和属性是自动化 XML 过程中经常使用的，请熟悉他们。

```
Set doc = XMLUtil.CreateXML()
doc.LoadFile "C:\AlanStudio_FY15\5. UFT_Scripts\BookStore.xml"
nodeCount = doc.GetRootElement.ChildElements.Count ' Get the sub node q'ty
Set root = doc.GetRootElement
Set allChildren = root.ChildElements

' 1. output tag name 和 tag value
```

```
For i = 1 To nodeCount
    print ">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>"
    print root.ChildElements.Item(i).ElementName & "::" & root.ChildElements.Item(i).
    Value & vbLf
    For j = 1 To allChildren.Item(i).ChildElements.Count
        print allChildren.Item(i).ChildElements.Item(j).ElementName & " <--->
        " & allChildren.Item(i).ChildElements.Item(j).Value & vblf
    Next
    print ">>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>" & vbcrlf & vbcr
Next
```

再来看一些例子 –

```
Set doc = XMLUtil.CreateXML()
doc.LoadFile "C:\AlanStudio_FY15\5. UFT_Scripts\BookStore.xml"
nodeCount = doc.GetRootElement.ChildElements.Count ' Get the sub node q'ty
Set root = doc.GetRootElement
Set allChildren = root.ChildElements

' 1. 获取根元素及其值 -
root.ElementName ' will return 'BookStore'
root.Value ' will return 'bookstore Value is here'

' 2. 获取子元素及其值
doc.ChildElementsByPath("//book").Item(1).ElementName
doc.ChildElementsByPath("//book").Item(1).Value

allChildren.Item(1).ElementName
allChildren.Item(1).Value

' 3. 获取元素架构通过给定的元素名称
allChildren.ItemByName("book") ' return the first element constructure
```

```
allChildren.ItemByName("book").NextSibling ' return the next element constructure based on current element
allChildren.AllItemsByName("book").Item(4) ' return the fourth element constructure
doc.ToString ' get the whole xml content as a string

' 4. 获取元素结构并重新生成新的 XML 文件
Set fso = CreateObject("Scripting.FileSystemObject")
set temp = fso.CreateTextFile("C:\temxml.xml")
temp.WriteLine allChildren.AllItemsByName("book").Item(4)
```

上面这些例子中，对象方法 'ChildElementsByPath' 是很有用的。他可以帮助我们选择需要的元素、标签的内容。这在 WebService 的自动化脚本中已有体现。

另外，方法 'ItemByName' 和 'AllItemsByName("book").Item(4)' 也很重要，在我们需要知道某个标签的内容是，它可以快速地提取所需的标签格式。省得通过方法 'ElementName' 和 'Value' 去一 一读取了。最后，我们可以选择性的将提取的元素重新生成新的 XML 文件。这对操作内容丰富，结构复杂的 XML 文档特别有用。

XML 结构严谨，标签没有预定义，用户可随意定义，但必须成对出现，并遵行一定的逻辑顺序规则。丰富的 XML 自定义标签可以表达人们日常生活中的各种信息，因此我们常说 XML 是传递信息和实现信息的语言。他注重的是信息的传递，HTML、XHTML 则侧重于信息的界面展示。初学者可能对标识符的使用有诸多的不习惯，但这并不影响我们驾驭脚本的能力，个人认为，掌握了上面这些 XML 方法，就可以随心所欲的操作 XML 文件了。自动化 XML 脚本也不过如此。

3.2.5 基于输出 Table 选择性的生成测试报告

根据已知的 Excel 输出，自定义 ExcelReport。这是 "3.2.1 做自己的 Excel 测试报告" 的另一具体应用，主要用于这样的场景 –

1.> 有结果输出 Table，但没有专门的 report 报告

2.> 需要提供单独的测试报告

3.> 脚本中没有单独定义测试报告

针对于这类需求，我们可以参考下面的脚本来快速生成独立的测试报告。

```
Function CustomizeReportBySpecifiedSheet(referenceFilePath, referenceSheet,
    part1Title, part2Flag, part2Title_1, part2Title_2, part3Title, reportName)
    Dim objExcl, usedColumns, columnSequence

    Set wsShell = CreateObject("Wscript.Shell")
    wsShell.Run "TaskKill /F /FI " & """imagename eq EXCE*""", 0, True
    Set wsShell = Nothing

    Set objExcl = CreateObject("Excel.Application")
    objExcl.Workbooks.Open referenceFilePath

    totalRecords = objExcl.Workbooks.Item(1).Sheets.Item(referenceSheet).UsedRange.R
    ows.Count
    For Iterator = 2 To totalRecords ' 因第一行是 Title 栏位，数据是从第二行开始的
        objExcl.DisplayAlerts = False
        objExcl.Workbooks.Add.Save

        Set workSheet1 = objExcl.Workbooks.Item(1).Sheets.Item(referenceSheet)
        Set workSheet2 = objExcl.Workbooks.Item(2).Sheets.Item(1)

        ' Get the used columns No.
        usedColumns = workSheet1.UsedRange.Columns.Count
        print "Used Cols No# from Sheet1 - " & usedColumns

        ' Initialize report columns name from the sample columns.
        For columnSequence = 1 To usedColumns
            workSheet2.Cells(1, columnSequence).Font.Size = 12 'the benefit of this writing can
            help us to get ride off the dictionary methods
            workSheet2.Cells(1, columnSequence) = workSheet1.Cells(1, columnSequence) '
```

```
    此行从 Test Data template copy 栏位
    'workSheet2.Cells(2, columnSequence + 1) = workSheet1.Cells(Iterator,
    columnSequence + 1) ' 注意超出范围的处理' because the usedColumns beginning
    from the sequence 0
    workSheet2.Cells(2, columnSequence) = workSheet1.Cells(Iterator,
    columnSequence) ' 注意超出范围的处理' because the usedColumns beginning from
    the sequence 0
    workSheet2.Cells(1, columnSequence).Interior.ColorIndex = 15
Next
workSheet2.UsedRange.Borders.LineStyle = 1 '给指定的范围加边框。

If part2Flag = "Y" Then '
    ' set IA2 checkpoint column
    workSheet2.Cells(4, 1) = part2Title_1 '注意此处和 40 行处的写法
    workSheet2.Range("G4") = "Result" '注意此处和 49 行处的写法

    workSheet2.Range("A4:F4").Merge
    workSheet2.Range("A4:G4").Font.Size = 12
    workSheet2.Range("A4:G4").Font.Bold = True
    workSheet2.Range("A4:G4").Font.Italic = True
    'workSheet2.Range("A3:J3").Interior.Color = vbBlue 'Background Color
    workSheet2.Range("A4:G4").Font.Color = vbBlack

    ' set the IA2 checkpoint content
    workSheet2.Cells(5, 1).Value = part2Title_2 & workSheet1.Cells(Iterator, 3).Valu
    e & """ ' Cells 参数只能用行、列来表示。此处行 2 数可参数化
    workSheet2.Range("A5:F5").Merge '一般情况下，只有属性才能赋值，而函数
    则不能赋值，除非是函数有返回值
    workSheet2.Range("A5:F5").Font.FontStyle = "Gill Sans MT"
    workSheet2.Range("A5:F5").Font.Size = 11
    workSheet2.Range("A5:F5").Borders.LineStyle = 1 ""
```

```
        ' set IA2 result
        workSheet2.Range("G5:K5").Merge
        workSheet2.Range("G5").Value = "Vendor Number :
        " & workSheet1.Cells(Iterator, 2) ' 2 可参数化
        workSheet2.Range("G5:J5").Borders.LineStyle = 1
    End If
    workSheet2.Range("A8:J8").Merge
    workSheet2.Range("A8").Value = part3Title

    workSheet2.Range("A8").Font.Size = 12
    workSheet2.Range("A8").Font.Bold = True
    workSheet2.Range("A8").Font.Italic = True
    'workSheet2.Range("A8").Interior.Color = vbBlue 'Background Color
    workSheet2.Range("A8").Font.Color = vbBlack

    'Define headers title of LH side
    workSheet2.Range("A9").Value = "Check_Items"
    workSheet2.Range("B9:F9").Merge
    workSheet2.Range("B9:F9").Value = "Expectation_Result" '注意此句和第 48 句的
    写法。range 参数尽管不一样，但实际结果是一样的
    workSheet2.Range("G9:I9").Merge
    workSheet2.Range("G9:I9").Value = "Actual_Result"
    workSheet2.Range("J9").Value = "Final_Matched_Result" ' 给 Range 指定的单元格
    赋值可用 range.value 或者直接赋值

    workSheet2.Range("A9:J9").Font.Size = 12
    workSheet2.Range("A9:J9").Font.FontStyle = "Gill Sans MT"
    workSheet2.Range("A9:J9").Interior.ColorIndex = 15

    ' fillup the LH check content
    For columnSequence = 2 To usedColumns '第一列的序号不拿来做比较，因此
```

```
    从第二列开始比较有数值的列。
    If workSheet1.Cells(Iterator, columnSequence) <> "" Then
        cellName = workSheet2.UsedRange.Rows.Count
        workSheet2.Range("A" & cellName + 1) = workSheet1.Cells(1,
        columnSequence)

        workSheet2.Range("B" & cellName + 1 & ":F" & cellName + 1).Merge
        workSheet2.Range("B" & cellName + 1) = workSheet1.Cells(Iterator,
        columnSequence)

        workSheet2.Range("G" & cellName + 1 & ":I" & cellName + 1).Merge
        workSheet2.Range("G" & cellName + 1) = workSheet1.Cells(Iterator,
        columnSequence)


        If workSheet2.Range("B" & cellName + 1).Value = workSheet2.Range("G" & c
           ellName + 1) Then
           workSheet2.Range("J" & cellName + 1) = "Matched"
           If workSheet2.Range("J" & cellName + 1) = "Matched" Then
              workSheet2.Range("J" & cellName + 1).Font.Color = vbYellow
           End If
        End If
        workSheet2.Range("A" & cellName + 1, "J" & cellName + 1).Font.Size = 11 'no
        tice the range property writing, it's different with before.
        'workSheet2.Range("J" & cellName + 1) = "Matched"
    End If
Next

workSheet2.UsedRange.HorizontalAlignment = 2 '向左对齐
workSheet2.Range("J10:J" & cellName + 1).Interior.ColorIndex = 17
workSheet2.Range("J10:J" & cellName + 1).HorizontalAlignment = 3 ' 居中对
```

```
齐
workSheet2.Range("A9:J" & cellName + 1).Borders.LineStyle = 1 ""

workSheet2.UsedRange.Font.Name = "Gill Sans MT"
 workSheet2.Cells.EntireColumn.AutoFit

objExcl.DisplayAlerts = False
objExcl.ActiveWindow.DisplayGridlines = False
objExcl.ActiveWindow.DisplayHeadings = False
objExcl.ActiveWindow.DisplayFormulas = False
objExcl.ActiveWindow.DisplayWhitespace = False
objExcl.ActiveWindow.DisplayRuler = False


workSheet2.Rows(1).Insert
workSheet2.Rows(1).RowHeight = 26 ' Set the row width
workSheet2.Rows(1).Font.Name = "Gill Sans MT"
workSheet2.Cells(1, 1) = part1Title
workSheet2.Cells(1, 1).Font.Size = 12
workSheet2.Cells(1, 1).Font.Bold = True
workSheet2.Cells(1, 1).Font.Italic = True
workSheet2.Cells(1, 1).Font.Color = vbBlack


colKey = GetAlphabetFromDictionary(usedColumns)
workSheet2.Rows(1).Insert
workSheet2.Rows(1).RowHeight = 38
workSheet2.Range("A1:" & colKey & "1").Borders(4).LineStyle = 3 ' =6 means the
bold dash line; 9 means double solid line; 4/5/ means single dash line; 2/3 means round
dot line; 1 means the sigle solid line;
workSheet2.Range("A1:" & colKey & "1").Borders(4).Weight = 3 ' 1-左 2-右 3-顶 4-
```

```
    底 5-斜( \ ) 6-斜( / )  Again, weight = 4 ' 设置线条的宽度
    workSheet2.Range("A1:" & colKey & "1").Font.Name = "Aharoni"
    workSheet2.Cells(1, 1).Value = "Test Report for Record # 1"
    workSheet2.Cells(1, 1).Font.Size = 22
    workSheet2.Cells(1, 1).Font.Bold = True
    workSheet2.Cells(1, 1).Font.Color = vbBlue


    workSheet2.Columns(1).Insert
    rowCount = workSheet2.UsedRange.Rows.Count
    workSheet2.Range("A1:A" & rowCount + 3).Borders(2).LineStyle = 1
    workSheet2.Range("A1:A" & rowCount + 3).Borders(2).Weight = 4 ' 设置线条的
    宽度 weight may set to 3, 4 or 1

    'objExcl.Workbooks.Item(2).SaveAs "C:\Report_Temp\VDM001_SRS_Accounting Vendor_"
    & "Record" & Iterator & "_" & workSheet1.Cells(Iterator, 2).Value
    objExcl.Workbooks.Item(2).SaveAs "C:\Report_Temp\" & RandomNumber(1, 2999
    ) & "_" & reportName ' **********生成的 report 路径。**********
    objExcl.Workbooks(2).Close
Next

objExcl.DisplayAlerts = False
objExcl.Workbooks(1).Close
objExcl.Quit

Set workSheet1 = Nothing
Set workSheet2 = Nothing
Set objExcl = Nothing

'excelObj.Sheets.Item(1).Rows(1).RowHeight = 40
'excelObj.Sheets.Item(1).Columns("A:A").Insert
```

```
End Function

Private Function GetAlphabetFromDictionary(key)
    Dim d
    Set d = CreateObject("Scripting.Dictionary")
    d.Add 1, "A" : d.Add 2, "B" : d.Add 3, "C" : d.Add 4, "D" : d.Add 5, "E" : _
    d.Add 6, "F" : d.Add 7, "G" : d.Add 8, "H" : d.Add 9, "I" : d.Add 46, "AT" :_
    d.Add 47, "AU" : d.Add 48, "AV" : d.Add 49, "AW" : d.Add 50, "Ax" : _
    d.Add 51, "AY" : d.Add 52, "AZ" :_
    d.Add 10, "J" : d.Add 11, "K" : d.Add 12, "L" : d.Add 13, "M" : _
    d.Add 14, "N" :  d.Add 53, "BA" :  d.Add 54, "BB" :  d.Add 55, "BC" :  _
    d.Add 56, "BD" :  d.Add 57, "BE" :  d.Add 58, "BF" :  d.Add 59, "BG" :  _
    d.Add 60, "BH" :  d.Add 61, "BI" :  d.Add 62, "BJ" :_
    d.Add 15, "O" : d.Add 16, "P" : d.Add 17, "Q" : d.Add 18, "R" : d.Add 19, "S" : _
    d.Add 20, "T" :  d.Add 63, "BK" : d.Add 64, "BL" : d.Add 65, "BM" : d.Add 66, "BN" :
_
    d.Add 67, "BO" : d.Add 68, "BP" : d.Add 69, "BQ" : d.Add 70, "BR" : d.Add 71, "BS" :
_
    d.Add 72, "BT":_
    d.Add 21, "U" : d.Add 22, "V" : d.Add 23, "W" : d.Add 24, "X" : d.Add 25, "Y" : _
    d.Add 26, "Z" :  d.Add 27, "AA" : d.Add 28, "AB" : d.Add 29, "AC" : d.Add 30, "AD" :
_
    d.Add 31, "AE" :  d.Add 73, "BU" : d.Add 74, "BV": d.Add 75, "BW": d.Add 76, "BX":
_
    d.Add 77, "BY": d.Add 78, "BZ":_
    d.Add 32, "AF" : d.Add 33, "AG" : d.Add 34, "AH" : d.Add 35, "AI" : d.Add 36, "AJ" :
_
    d.Add 37, "AK" : d.Add 38, "AL" : d.Add 39, "AM" : d.Add 40, "AN" : _
    d.Add 41, "AO" : _
    d.Add 42, "AP" : d.Add 43, "AQ" : d.Add 44, "AR" : d.Add 45, "AS"
    GetAlphabetFromDictionary = d.Item(key)
```

```
    Set d = Nothing
End Function
```

点评：这个场景常用于测试报告的再处理。它是基于已生成的 Excel 文件，来重构测试报告的，前提是这个 Excel 文件中包含了重构测试报告的必要信息，如输入数据，输出数据，最终结果等。

3.2.5 基于文本文件的字符串处理

在文本文件中处理字符串操作也是一类常见的脚本处理场景。关于字符串的处理，前面我也介绍了一些相关的方法，但那些方法的操作前提是在已知搜索字符串的情景下进行的。那么如何处理事先并不知情的字符串呢？这就要涉及到文本文件的操作了，通常在这种情形下，我们需要先处理文本文件，然后基于文件的内容做进一步的字符串处理。我给大家准备了三个场景，很简单，但确实在某些现实需求中经常碰到。

3.2.5.1 获取文本文件中某一指定行的内容

```
Function getTextForASpecifiedRow_FromATextFile(row, filePath)
    Dim fso, flag, Iterator

    flag = 1
    Set fso = CreateObject("Scripting.FileSystemObject")

    If fso.FileExists(filePath) Then
        Set theFile = fso.OpenTextFile(filePath, 1, False)

        For Iterator = 1 To row - 1
            If Not theFile.AtEndOfLine Then
                theFile.SkipLine
'               theFile.ReadLine
            End If
        Next
        print theFile.Line
```

```
    Else
        flag = 0
    End If

    '注意： 文本文件不能有空行，不然脚本会默认空行为结尾行。读取文本行数据; Skipline 和 Readline 方法会使得文本指针（其实 fso 属性和方法没有指针概念，但可以这么理解）下移一行

    If flag = 1 Then
        If Not theFile.AtEndOfLine Then
            getTextForASpecifiedRow_FromATextFile = theFile.ReadLine
            Else
            getTextForASpecifiedRow_FromATextFile = "The Text is over the line!"
        End If
    Else
        getTextForASpecifiedRow_FromATextFile = "the file is not existing!!!"
    End If

    Set theFile = Nothing
    Set fso = Nothing
End Function
```

3.2.5.2 验证文本文件中是否存在某一指定的字符串

```
Function ValidateStringExistingWithinTextFile(filePath, string_Searchedfor)
    Dim fso, myFile, searchedText

    Set fso = CreateObject("Scripting.FileSystemObject")
    If fso.FileExists(filePath) Then
        Set myFile = fso.OpenTextFile(filePath, 1, False) '1 means for reading, 2 means for writing, 8 means for appending; False means the file won't be creaetd if the file doesn't exist
```

```
    searchedText = myFile.ReadAll
    ValidateStringExistingWithinTextFile = searchedText

    position = Instr(1, searchedText, string_Searchedfor, 1) ' the last 1 means the
     searching is not case sensitive. but the '0' means on the reverse

    If position > 0 Then
      Reporter.Filter = rfEnableAll  ' or Reporter.Filter = rfDisableAll ' 打开和关闭向
      QTP 报告输出日志的操作。有些步骤不希望出现在 Report 中，可以使用这个
      开关语句。当然这是指 QTP 自生成的报告。
      reporter.ReportEvent micPass, "Searching a specified string within a text
       file", "Yes, It was found at this file!"
      'print reporter.ReportPath
    Else
      'Reporter.Filter = rfDisableAll
      reporter.ReportEvent micFail, "Searching a specified string within a text
       file", "Unfortunately, It wasn't found at this file!"
    End If
  Else
    Reporter.ReportEvent micFail, "Searching a specified string within a text file", "Sorry,
    There isn't this file at the path. Please check if you get the right name and path!"
  End If

  Set myFile = Nothing
  Set fso = Nothing
End Function
```

3.2.5.3 搜索文本文件并找出包含指定字符串的所有记录行

```
Function SearchStringWithATextfile_GetSearchReturn(filePath, searchedforString)
  Dim fso, buttomRow, myfile, count, position, rawText
```

```
    Set fso = CreateObject("Scripting.FileSystemObject")
    If fso.FileExists(filePath) Then
        Set myfile = fso.OpenTextFile(filePath, 1, False)

        Do
            myfile.SkipLine
        Loop While Not myfile.AtEndOfLine
        buttomRow = myfile.Line
        Set myfile = Nothing

        Set myfile = fso.OpenTextFile(filePath, 1, False)
        count = 0
        For i = 1 To buttomRow
            If Not myfile.AtEndOfLine Then
                rawText = myfile.ReadLine
                print rawText
                position = Instr(1, rawText, searchedforString, 1)

                If position > 0 Then
                    reporter.ReportEvent micPass, "Searching the specified string within a text
                    file", "Yes, It was found at the row # " & i & " , And this row context is
                    '" & rawText & "'"
                    count = count + 1
'               ElseIf count = 0 Then
'                   Reporter.ReportEvent micFail, "Searching the specified string within being
                    searched string", "Unfortunately, it isn't found at this text file!"
                End If
            End If
        Next
```

```
    If count = 0 Then
        Reporter.ReportEvent micFail, "Searching the specified string within a text
        file", "Unfortunately, it isn't found at this text file!"
    End If
  Else
    Reporter.ReportEvent micFail, "Searching the specified string within a text
     file", "Ooh, my Gad! there isn't this file at the path, please check the file name and
      the path again!" & vbCR & _
    "And ensure you provided the right file here!"
  End If

  Set myfile = Nothing
  Set fso = Nothing
End Function
```

3.2.5.4 搜索文本字符流并替换搜索的字符串

```
Function SearchStringWithinTextAndInsteadWithSpecifiedString(filePath, findString,
searchedforString, SaveasfilePath)
  Dim fso, buttomRow, myfile, count, position, rawText

  Set fso = CreateObject("Scripting.FileSystemObject")
  If fso.FileExists(filePath) Then
    Set myfile = fso.OpenTextFile(filePath, 1, False)

    Do
      myfile.SkipLine
    Loop While Not myfile.AtEndOfLine
    buttomRow = myfile.Line
    Set myfile = Nothing

    Set myfile = fso.OpenTextFile(filePath, 1, False)
```

```
        getText = myfile.ReadAll
        replacedText = Replace(getText, findString, replaceString)
        Set myfile = Nothing


        Set myfile = fso.OpenTextFile(SaveasfilePath, 2, True) '此处 2 for writing; True for
        creating if there isn't a specified file existing
        myfile.Write replacedText

        SearchStringWithinTextAndInsteadWithSpecifiedString = replacedText

'       myfile.WriteBlankLines(3)
'       myfile.WriteLine replacedText
        myfile.Close
    Else
        Reporter.ReportEvent micFail, "Searching the specified string within a text
        file", "Ooh, my Gad! there isn't this file at the path, please check the file name and
        the path again!" & vbCR & _
        "And ensure you provided the right file here!"
    End If
    Set myfile = Nothing
    Set fso = Nothing
End Function
```

点评：上述五种场景涵盖了常见的文本字符串处理案例，实际使用中要体会的是注释行的用法，文本处理的函数不多，常见的由 OpenTextFile，Read， Write 等等。

3.1.7 VBScript 错误处理机制

第四章 QTP/UFT 常见问答

第四章 UFT/QTP 常用设置

XXX

2.1 Options 选项

这个选项位于菜单栏的 Tools→ Options

在这个 Table 里有很多功能可以设置。常用的设置有这些 Tools → Options → General

2.1.1 Starts Options：这里可以设置 “Display Add-in”；“Display Start Page on Startup”

2.1.2 Run Sessions：这里可以设置 “View results when run session ends”

另外一个常用的设置是在 Tools → Options → GUI Testing

2.1.3 Folders：这里可以设置 Relative path 和 Absolute Path

第三章 Performance Testing

Performance 测试也是自动化测试的一个重点领域，了解和掌握这一部分内容，也是完善自动化测试技能的一个补充。Performance 测试包含的内容很多，这可以从其支持的协议看出。但日常测试中往往集中在 HTML，WebService 等协议，这里我将以实际的测试用例来详细介绍如何利用 LR（Load Runner）来实现这类 Cases 的性能测试。先来看几个 Cases。

3.1 典型案例

关于 Web Service 的案例，通常我们遇到最多的 Cases 类似于下面的步骤，这应该是 Web Services 的标准测试步骤了，也是 PT 中测试 WebService 最常用的步骤。先来了解下，后面再来看怎么脚本实现。

Case One –

Step1： Get WSDL from URL、File、UDDI （http://mdm-itg.houston.hp.com:8180/customer-manager/services/CompetitorService?wsdl）

Step2：Initialize Parameters with Customer IDs

Step3：Execute Search operation to get the response

Case two –

关于 HTML 的案例，这部分 Cases 包含的情况很多，主要是因为基于 Html 的操作很普遍，比喻页面登录，查找，比较，搜索等等，都是常见的测试，这里我也提炼了一些基本的步骤，不一定全面，但却是可以代表很大一部分基于 Html 操作的 PT 方案。

Step1：Get the URL for portal

Step2：Login with account

Step3：Initialize with Pramenters

Step4：Execute Searching and get the response

上面两中情况的性能测试在软件测试中出现的比率很高。熟练理解和编写这类脚本可以让我们轻松对付 performance 测试中 80%以上的 Cases. （呵呵。。。别拍砖，个人就是这么认为的）

这部分内容要怎么介绍呢？采取类似于 AT 的，一个案例一个案例的说比较乏味，也不太容易理解。因为基本上就是脚本代码重复 Copy 和 Review。不具有参考性。还是围绕这两类 Cases 步骤，以及与其相关的知识点逐一展开吧。

3.2 Common Function Statement

String and Parameter Function –

Lr_eval_string : Returns the string argument after evaluating embedded parameters.

char *lr_eval_string(const char *instring);

The lr_eval_string function returns the input string after evaluating any embedded parameters. If the string argument contains only a parameter, the function returns the current value of the parameter.

Embedded parameters must be in brackets.

也就是说此函数返回任何被输入参数的被计算值。

Data Type conversion Function

Atoi : Converts a string to an integer value

int atoi(const char *string);

atoi reads the initial portion of the string only, by stopping at the first non-numerical character.

此函数是字符串转换函数，它只读取字符串内第一个数字的值，并将其转换成整形数值。如“71 dollars and 81 cents”函数将输出 71. 如果初始字符串不是以数字开头，则函数返回 0. 如“a value from 71 dollors and 81 cents” 函数将输出 0.

Message Function

lr_output_message : Sends a Vuser message to the log file and output window with location information.

int lr_output_message(const char *format, exp_1, exp_2,...exp_n);

这里需要注意的是 format 字符串的表述，他又很多种格式，常用的有 int(d, i) 和 char（s）。如(“pleae ouput the value of this paramerter %d, %i“, parameter), 将输出 十进制整型数值；但如果写成(“pleae ouput the value of this paramerter %s”, parameter) 将输出字符串知道碰到 `\0'转型字符或者用户给定的精确字符位置。

Header Function

web_add_header : Adds a customized header to the next HTTP request.

int web_add_header(const char *Header, const char *Content);

这个函数很好理解，Header 和 Content 是 Html 里常见的标签。即：

Header： The name of the user–defined header.Examples: "Accept", "Accept–Encoding"

Content：The value (data) associated with the user–defined header.

Action Function

Web_submit_data : Performs an "unconditional" or "contextless" form submission.

int web_submit_data(const char *StepName, const char *Action, <List of Attributes>, ITEMDATA, <List of data>, [EXTRARES, <List of Resource Attributes>,] LAST);

StepName 就是一常量字符串，可自定义

Action 通常就是 URL 值

List of Attributes 通常包含属性-

- Method – POST or Get
- TargetFrame =blank or 其他值
- RecContentType = “Text/Html” or 其他值
- Referer
- Snapshot
- Mode
- ITEMDATA,

- "Name=grpType", "Value=radRoundtrip", ENDITEM,

各个属性的值可参考下面函数的相同属性值说明。

Web_custom_request : Allows you to create a custom HTTP request with any method supported by HTTP.

int Web_custom_request(const char *RequestName, <List of Attributes>, [EXTRARES, <List of Resource Attributes>,] LAST);

❖ Statement for this Function :

RequestName, List of Attribute 是必须的参数，尤其是 attributes 参数，它有如下选项：

URL - the URL (Uniform Resource Locator) of the Web page to load. The URL may be any of the following protocols: HTTP, HTTPS, or FTP.

Method – "POST" or "GET"

TargetFrame –

_BLANK: Opens a new window.

_PARENT: Replaces the parent of the last (changed) frame.

_SELF: Replaces the last (changed) frame.

_TOP: Replaces the whole page.

Body

Web_url : Loads a specific URL （Get request only）

int Web_url(const char *StepName, const char *url, <List of Attributes>, [EXTRARES, <List of Resource Attributes>,] LAST);

❖ Statement for this Function :

常用的 List of Attributes 有这些选项 –

TargetFrame - The name of the frame containing the current link or resource. Click TargetFrame for more information.

RecContentType - The content–type of RESPONSE header during recording, for example, *text/html*, *application/x–javascript*. It is read to determine whether the target URL is a recordable resource. Click RecContentType for more information.

Referer - The URL of the referring Web page. The page that referred to the current page. If the location was explicitly expressed, this attribute is omitted.

Resource - A value indicating whether the URL is a resource:

0 – the URL is not a resource

1 – the URL is a resource

Snapshot - The file name of the snapshot file (*inf* extension), used for correlation.

Mode - The Recording Level: HTML or HTTP.

WebService Function

Web_Service_Call : Calls a web Service

int Web_service_call(const char *StepName, [URL,] *ExpectedResponse,* <List of specifications>, [BEGIN_ARGUMENTS, *Arguments*, END_ARGUMENTS,] [*Send Attachments*,][BEGIN_RESULT, *Results*, END_RESULT,] [*Receive Attachments*,] LAST);

注意：有些函数的使用需要写在请求函数的前面。这是为什么呢? 通常这一类函数含有 reg 关键字。我们也称这类函数叫注册型函数。它告诉 VuGen 下一个请求返回时需要处理的。所以该函数必须写在该请求函数的前面。否则就会提示无法获得关联结果的错误。

3.3 需具备的 C 基础

因为 LR 的脚本是基于 C 设计的，所以对 C 的基本了解是很有必要的，大家知道有关 C 的书籍太多了，这里并不是要专门来介绍 C 的方方面面，我将试着将一些常用的 C 知识点罗列出来，供大家写脚本时参考，提供理论支持。

3.3.1 LR 中常用 C 语法归纳

- 每一个 C 程序都必须有 main（）函数
- 函数体内由大括弧{ }括起来

- “；”是 C 语句的结束标志。每个语句或变量说明（定义）的最后必须以分号“;”结尾，它是语句终止符。

- 符号和字母，都要用半角格式的
- Printf()输出函数
- \n 转义字母,代表换行
- /* */注释符
- %d 用来代替“，”后面的变量，并指定该变量以什么类型输出
- %d 按整型输出
- %f 按实型输出
- %c 以字符形式输出，只输出一个字符

- 一个 C 语言程序可以由多个函数组成，但必须包含且只能包含一个主函数 main（）函数是组成 C 语言程序的基本单位。

- 一个函数由函数头和函数体两部分组成。函数头一般包括函数返回类型，函数名称和包含在一对圆括号“（）”中的函数参数以及参数类型等。一个函数也可以没有参数。C 中除了主函数为 main（）外，其他标准库函数和用户自定义函数可以自行命名。但要符合 C 语言的标识符命名规则。

- 函数体：即函数首部下面一对花括号“{}”内的部分。在函数体中一般包括说明部分和执行部分。说明部分使实体名与被说明的实体相联系。如变量名和被吊函数的声明等。有的简单程序可以没有这一部分。

- 一个 C 程序中函数出现的次序可以是任意的，但总是从 main 函数开始执行。
- 前面带有“#”的语句，如#include，#define 等都是编译预处理命令。
- C 语言程序书写比较自由，一行可有多条语句，一条语句也可写成多行（但不能讲一个单词分开写）。为了提高程序的可读性，往往是一行只写一个语句。并以缩进法来体现语句的层次。

- 函数头定义了函数名称，函数类型，函数的形式参数及其类型。如： int max（int x, int y）

- 指针是以*号来区分的，指针就是地址；指针变量就是存储变量的地址。

- 函数体，即是函数所要实现的功能定义。就是花括号内的部分“{ }“。函数体一般包括：
  ①变量说明。 如 int a, b, c;
  ②执行部分。由若干条语句组成，每个语句都以分号作为结束符（；）

- 字符常量与字符串常量的区别：字符常量用单引号括起来的部分，而字符串指被双引号括起来的部分。注意下面语句的区别：
  Char c,b;
  C=9; c='a'
  String e;
  e=9; c="abcdef";

- C 中最最重要的是指针，掌握指针就掌握了 C 的精华。
- 简单地说，指针就是地址。
- 变量的地址就是该变量的指针。
- 存放地址的变量称为指针变量。
- 用“*”代表“指向”，如*p2 代表他所指向的变量（假设为 i）的同一内存单元。
- 下面两个语句等价：
  I = 3; （直接访问）
  *p2 = 3; (间接访问)

- 又如指针变量定义：
  Int *p1; (*表示该变量为指针变量，但变量名为 P1)

- 指针变量的引用：
  ①用地址运算符 &

P1=&I;

②用指针运算符 * (实行间接访问)

*p1 = 100; K=*p1;

Notice : 指针变量只能存放地址（指针）

P1=100; (不允许)

- 函数的返回值：
  函数的返回值是指流程从被调函数返回到主函数。有的函数有返回值，有的没有。通常该返回值是通过 return 语句来实现的。

- Return 语句有两个作用：一个是宣告一次函数调用的结束；二个是带回函数的返回值，送到调用表达式中去。

- 函数的调用有语句调用。如：Printf（“welcome to our team %s”, myteam）；
  表达式调用。如：m=2*max(a,b); z=sqrt(x)+sqrt(y); 还有参数调用。如：
  printf(“check the max value is : %d”, max(a,b));

- 编译预处理
  编译预处理是在编译之前对程序的一些与加工。预处理命令是有 ANSI C 统一规定的，他不是 C 语言的组成部分，不能直接对他们进行编译。必须在编译之前先对这些特殊命令进行预处理，并将处理结果和源程序一起再进行通常的编译，最终得到目标程序。

- C 语言提供的预处理主要有三种：宏定义，文件包含，条件编译。为了与 C 语言的语句区别开来，这些命令以符号“#”开头，且后面不加分号。
  如：不带参数的宏定义： #define PI 3.1415926
  如：带参数的宏定义：#define S(a,b) a*b
  Area=S(4,5);

  如：文件包含定义：#include “文件名”或者是#include<文件名>

这是我们在 C 中常遇到的预处理形式。如：#include<stdio.h>或
#include“stdio.h”

如：条件编译形式有-
#ifdef debug
Printf(“x=%d, y=%d, z=%d\n”, x,y,z);
#endif
条件编译就是告诉编译程序有些语句只有在满足一定的条件下才参与编译的。

3.4 常用协议之 – HTTP

这个协议大家并不陌生，多少也知道点，但要说全很难。确实如此，光介绍 Http 的书籍就不计其数。（直到现在，我都在后悔为什么选择学这个行业，太多东西需要学习了，哈。。。当然说回来，哪一行不要学习呢，除非你不想上进）那么我们是不是非得完全了解了它才可以使用呢？显然不是的。有些使用其实不需要我们深入太多，就 LR 测试而言，我们需要必备 Http 哪些知识点呢？其实不需要太多。梳理下原理，记住几个典型就可以上手了。

3.4.1 HTTP 协议主要特征

1. 支持客户/服务器模式
2. 简单快速的访问和相应机制
3. 灵活的传输数据类型
4. 无连接。即每次连接只处理一个请求，服务器处理完客户端请求收到应答后就断开连接
5. 无状态。就是说协议对于事务处理没有记忆能力。

3.4.2 有关 HTTP 的几个重要概念

URL： Universal Resource Location. 格式为：http://host [“:”port][abs_path]
它是一种特殊的 URI（Universal Resource Identifier）。它包含了要查找某个资源的足够的信息。

Http 请求： 它由三部分组成。请求行、消息报头、请求正文
请求行以一个方法符号开头，以空格分开，后面跟着的 URI 和协议的版本。格式如下：Method Request-URI HTTP-Version CRLF 这里需要解释下的是 CRLF，它是表示回车和换行的作用。不可单独使用。只能出现在请求行里。

请求方法有很多种，常见的有：
GET 请求获取 Request-URI 所标识的资源
POST 在 Request-URI 所标识的资源后附加新的数据
（这两个方法最常见，我会在后面单独来聊一聊）
HEAD 请求获取由 Request-URI 所标识的资源的响应消息报头
PUT 请求服务器存储一个资源，并用 Request-URI 作为其标识符
DELETE 请求服务器删除 Request-URI 所标识的资源
TRACE 请求服务器回送收到的请求信息，主要用于测试或诊断
CONNECT 保留以备将来使用
OPTION 请求查询服务器的性能，或者查询与资源相关的选项和需求

还记得我上面提到的 Fildder 工具吗？用它可以很显式的查看 HTTP 协议的各请求表述。我将在稍后截图详细解说。下面来看看消息报头的介绍。
消息报头格式为：**名称: 值** （是不是很简单？）

常见的请求报头有：
Accept： image/gif 表明客户端希望接受 GIF 图像格式的资源；
Accept： text/html 表明客户端希望接受 Html 文本
Accept-Encoding： gzip.deflate 表明客户端希望的编码方式
Accept-Language： zh-cn 指定客户端的接受语言
Host： http://google.com.cn/ 这个是必需的。告知访问的服务器地址

Http 响应：在接受和解释请求消息后，服务器返回一个 HTTP 响应消息。这个响应消息也是由三部分组成的，状态行、消息报头、响应正文

状态行格式：HTTP-Version status-Code Reason-Phrase CRLF

其中，Http-Version 代表服务器 HTTP 协议的版本；Status-Code 表示服务器发回的响应状态代码；Reason-Phrase 表示状态代码的文本描述。这里这里着重讲讲状态代码的含义。状态代码用三位数字来表示，第一个数字代表了相应的类别。通常有五种可能的值。（这点务必了解，对我们看 Web 返回信息或日志之类的非常有用，有经验的工程师从状态代码就能很快的判别是什么类型的错误。是不是貌似很牛？其实你也可以的）。

那么是哪五种可能的值呢?

1XX： 指示信息 -- 表示信息请求已被接受，继续处理
2XX： 成功 -- 表示请求已被成功接受，理解和受理
3XX： 重定向 – 要完成请求必须进行更进一步的操作
4XX： 客户端错误 – 请求有语法错误或请求无法实现
5XX： 服务器端错误 – 服务器未能实现合法的请求

响应消息报头可参考上面的请求报头描述。响应报头允许服务器传递不能放在状态行中的附加响应信息，以及关于服务器的信息和对 Request-URI 所标识的资源进行下一步访问的信息。

下面是用 Fillder 查看 Google 页面的 Http 协议方法的截图。这对我们理解上面的协议介绍很有帮助，也很直观的给出了 Http 协议的框架形式。

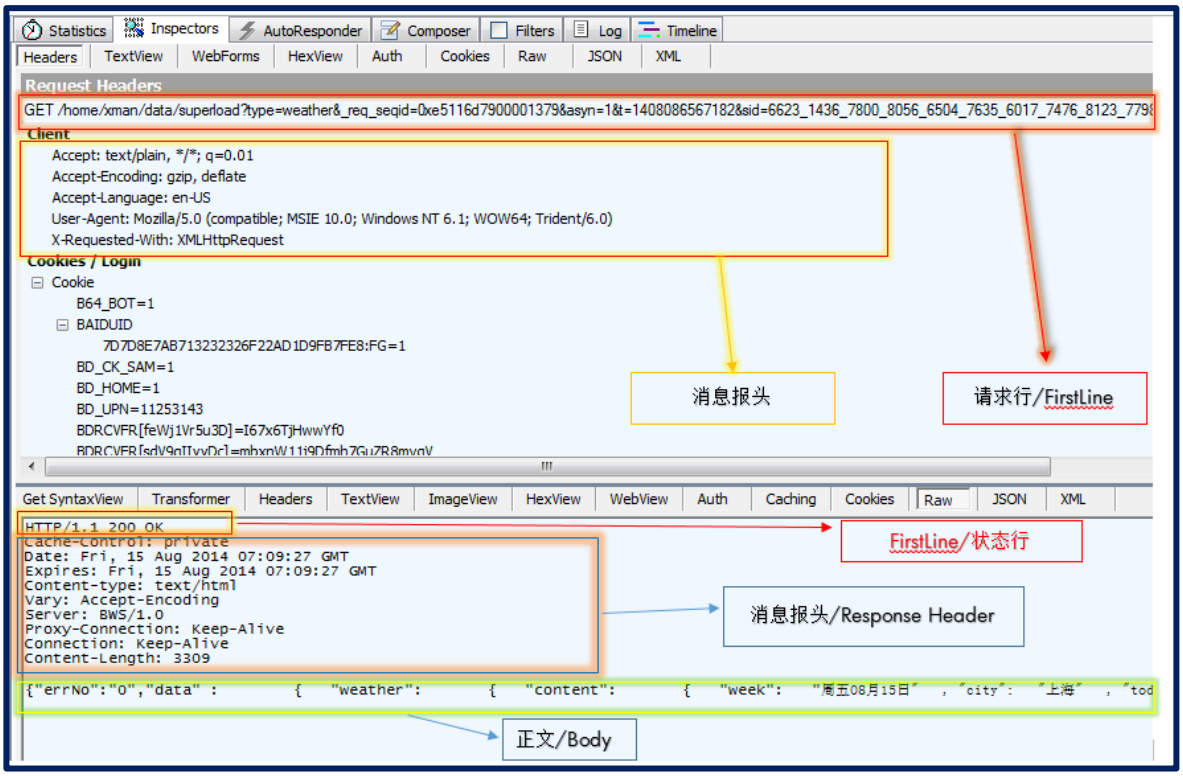

图 4-1 Http 协议框架图

另外，两个最常用的 Http 协议请求（Get、POST）也应该要有一个基本的认识，了解和掌握这两个协议的基本特征，对我们的脚本设计大有裨益。 这不仅是因为两种协议在各个网站中的广泛应用，还因为它在 LR 脚本函数中出现的频率奇高。也难怪，毕竟人家是互联网的基础协议嘛。

如上面所述，下面我们单独聊聊 GET 和 POST 方法，以 LR 为驱动，我们到底需要知道 GET 和 POST 的哪些知识呢？个人认为不需要太深入，但明白下面的内容将帮助我们更好的理解 GET 和 POST 方法的使用。

- GET 和 POST 是 http 协议中客户端最常见的请求方式，除此之外还有另外的方式，如 PUT，HEAD 等，但很少使用。
- GET 和 POST 的区别主要在于客户端传递参数的时候。
- GET 把请求参数跟在 URL 后面，用问号隔开，如：
  /Web30/yy?login=yyyyyyy&password=123

- POST 把请求参数放到 http 请求协议的内容里。
- Get 请求 URL 会显示在浏览器地址栏里，所以不如 POST 安全。
- 不同的浏览器会对 URL 的长度进行限制，所以 GET 不能传太多的参数。
- GET 请求只能传文本给服务器，POST 可以传广本和二进制数据，如上传文件。
- 地址栏输入 URL，超链接，表单中设置 method='get'，都是常见的 GET 方式请求。
- 只有表单设置 method='post'，是 Post 请求。

3.5 常用协议 – WS/SOAP,WSDL, UDDI

WebService 是通过一系列标准和协议来确保程序之间的动态连接。在 WebService 中最常用的协议是 Soap， WSDL 和 UDDI 了解和熟悉这些基本协议，对后面我们编写 WS 的脚本非常有帮助，一起来认识下。

SOAP： Simple Object Access Protocol 的缩写，SOAP 是简单的对象访问协议，它规定了 Web Service 之间是怎样传递消息的。一般来说它包含这些方面的内容：

1. 传递信息的格式为 XML。这就使得 WebService 能够在任何平台上，用任何语言进行实现。
2. 远程对象方法调用的格式。规定了怎样表示被调用对象以及调用的方法名称和参数类型等。
3. 参数类型和 XML 格式之间的映射。这是因为，被调用的方法有时候需要传递一个复杂的参数，那么怎么样用 XML 来表示一个对象的参数呢？这就需要 SOAP 来定义范围。
4. 异常处理以及其他信息等。

WSDL：Web Services Description Language 的缩写。顾名思义，WSDL 就是 Web Service 的定义语言。当你实现了某种服务的时候（即实现了一种 WebService 功能）为了让别的程序调用，你是不是得告诉大家你的服务接口，监听端口号，传递参数的个数，类型以及返回结果等等？ 这样别的应用程序才能调用你的服务。WSDL 就是干这个事的。它规定了有关 Web Services 描述的标准。

UDDI：Universal Description Discovery and Integration 的缩写。简单地说，就是用于集中存放和查找 WSDL 描述文件，起着目录服务器的作用。

一个 Web Services 的工作流大致可以理解成：

- 实现一个 Web Services， 使其能够接受和响应 SOAP 消息，很多工具可以帮助实现这一块。
- 撰写一个 WSDL 文件用于描述此 Web Services. 同样现在有很多工具可以自动生成 WSDL 文件。
- 将此 WSDL 发布到 UDDI 上
- 其他应用程序（客户端）从 UDDI 上搜索到你的 WSDL。
- 根据你的 WSDL，客户端可以编写程序，调用你的 Web Services。

3.6 脚本设计思路

Performance 测试不同于 AT，通常我们主张用 Recording 的办法来录制脚本，这样做不仅可以节省时间，而且对脚本中的各种 Action（如 .CSS , .Do，.Json 或者 Jave Script， Asp 等等控件）能够很好的识别。对于脚本的维护也是显而易见的。请看下面的脚本实现。

For Case one –

1.）录制 WebService 脚本

```
web_service_call( "StepName=customSearchLookupService_101",
"SOAPMethod=CustomerLookupEnWebServiceService|CustomerLookupEnWebSe
 rvicePort|customSearchLookupService",
"ResponseParam=response",
"Service=CustomerLookupEnWebServiceService",
"ExpectedResponse=SoapResult",
"Snapshot=t1395726342.inf",
BEGIN_ARGUMENTS,
```

```
    "xml:customSearchLookupService1="
      "<customSearchLookupService1>"
        "<customerId>{CustomerId}</customerId>"
      "</customSearchLookupService1>",
    END_ARGUMENTS,
    BEGIN_RESULT,
    END_RESULT,
    LAST);
```

注意 Web_Service_Call 用法,从面试的语法可知，Web_Service_Call 通常由

ExpectedResponse

specifications

StepName

参数构成，其他的都是 optional. 用户可以根据实际情况选择使用。每个参数的定义可以从 Help 文档中查询，（F1: 查询函数用法）

2.）设置 Checkpoint

```
getValuefromResponse = lr_xml_find("XML={response}",
"Query=/Envelope[1]/Body[1]/customSearchLookupServiceResponse[1]/customerSearchLookup
ResultGroup[1]/externalProfileGroup[1]/externalProfileLocatorBusinessLocatorNumber[1]/text()[1]",
 "Value=0709898899",
 LAST);
```

3.）判断 Checkpoint 的条件

```
if (getValuefromResponse == 1){
    lr_end_transaction("RequestAndResponse", LR_PASS);
}
else{
    lr_end_transaction("RequestAndResponse", LR_FAIL);
```

```
}
return 0;
```

录制和设置 Chekcpioint 部分可以直接使用 LR 的界面完成，第三部需要手工编写 Checkpoint 条件。根据函数的语法（参见上面 3.2 部分内容）其实也不难完成。

Note：再次体会 LR C 语法的基本特点，每条语句都以分号结束。小括号内的每一参数赋值均已逗号结束，并且每条语句必须以双引号引用起来。引用变量值使用大括号{ }。另外，如果双引号里的内容又是被上引号引用的，那么就需要用反斜杠 “/”进行转换。（这在 LR 的 help 文件里有详细说明）

For Case Two –

这部分脚本完全是手工编写的，目的就是如何利用上面介绍的函数来编写 LR 脚本。（如果可以录制的话，还是建议录制的办法，这里介绍的手工编写代码，仅供那些不能录制脚本的情况作为参考）

1.）Login URL

使用”Action Function – Web_Submit_Data“函数发送 URL 请求

```
web_submit_data("login.pl",
    "Action=https://it-services-itg.external.hp.com/auth/login.pl",
    "Method=POST",
    "RecContentType=text/html",
    "Referer=",
    "Snapshot=t5.inf",
    "Mode=HTML",
    "EncodeAtSign=YES",
    ITEMDATA,
    "Name=action", "Value=logon", ENDITEM,
    "Name=deviceos", "Value=2.0.0", ENDITEM,
    "Name=devicetype", "Value=TouchPad", ENDITEM,
    "Name=deviceNDUID", "Value=123", ENDITEM,
```

```
        "Name=user", "Value=your email address", ENDITEM,
        "Name=password", "Value=your nt account", ENDITEM,
        "Name=osType", "Value=WebOS", ENDITEM,
        LAST);
```

注意相关参数的设置，同样可以参考 Help 文档设置 Web_Submit_Data 函数。接下来就是 Search 操作了，由下面的函数实现。

2.） Search with the specified initialization

```
web_reg_find("Search=Body",
        "SaveCount=findcount",
        "Text=SDGT",
        LAST);

 status = web_url("rplAnywhere-web",
        "URL=https://it-services-itg.external.hp.com/onebox/rplAnywhere-
         web_1_0_0/services/rpl?name=BIN%20LADEN&countryCode=US",
        "Resource=0",
        "RecContentType=text/html",
        "Referer=",
        "Snapshot=t6.inf",
        "Mode=HTML",
        LAST);
```

对照 Web_URL 语法，这里对三个必要参数赋值- Step Name, URL 和 Attribute list. 除了这个函数外，还可以用其他函数如 web_submit_data, web_URL 代替。他们都属于 Action Function. 同样可达到发送 URL 请求的目的。

接下来就是 Checkpoint 检查了，代码如下

3.) Checkpoint Setting and Validation

```
    lr_output_message("request status :%d", status);
    lr_output_message("find count :%d", atoi(lr_eval_string("{findcount}")));

    if (atoi(lr_eval_string("{findcount}")) > 0){//check the times of finding 'SDGT' string
            lr_output_message("found the value at the return page");
            lr_end_transaction("AdhocSearching", LR_PASS);
        }
        else{
            lr_output_message("found fail");
            lr_end_transaction("AdhocSearching", LR_FAIL);
            }

    return 0;
```

此处的函数的理解可参考上面的解析。这里不再重复了。到此为止，两种案例的代码实现介绍完毕，是不是还是感觉云里雾里？很正常，这主要是因为 LR 的 C 实现函数参数太多，而且使用起来也不是很有感觉。

比喻说调用 Web_Custom_request 函数发送 URL 请求。那么函数 Web_Custom_request 到底该参数化哪些参数？每个参数的值该如何获取？还是没有一个直观的认识。这也是你尽管看了上面那些函数解析后后任然不知所措的根本所在。这是个难点。但我们有办法破解 ☺

方法一： 如何识别哪些方法需要被 Action Function 提取

这是手工开发 LR 代码的必要步骤，你首先要知道哪些页面方法需要被 Action Function 提取，找出这些方法，然后调用 Action Function 函数予以实现。难就难在找出这些方法上。看好下面的介绍，解决上面问题的 solution 就要出来了 –

进入相关页面，进入代码视图，进入 Network 视图，反复操作页面，看哪些方法的 response 是关联下一个步骤的。把这些方法提出来，一 一调用 Action Function 函数予以实现。 好了这就是要领 。(这里主要是针对 record 困难的 Chrome 环境。 按 F12 进入)

不知道？还是不知道？那你打开一个 GUI 界面（IE 或者 Chrome 都可以）进入代码视图模式，反复进行页面操作（如 login），然后在 Network 视图里看看哪些方法被识别出来了，再看看这些方法的 response，有哪些跟下一步骤的界面相关。提出来。这就是 Action Function 的函数要去实现的步骤。就这么简单。实在不会的话，就按照上面的介绍反复操练吧，知道熟练找出页面方法为止。

方法二： 如何识别哪些方法需要被 Action Function 提取

请使用第三方工具来帮助提取。这里推荐使用 **Fiddler** 工具，这个工具可以很方便的看出页面操作对象以及方法属性。用它可以提高我们的代码开发效率，有兴趣的童鞋不妨试试。个人感觉还是不错的。

使用它可以帮我们快速找出页面相关元素，以及相应方法，找到这些方法和元素后，跟方法 一 一样，我们需要将这些方法和元素以函数的形式表述出来。

这就是 LR C 脚本开发应该遵行的规则。当然如果能 record 的话，还是建议大家 record。毕竟系统自动抓出的方法和属性要比手工去添加的来得快和准。

3.7 有用的脚本参考

再来看看下面这些脚本，一起来回顾下上面介绍的方法，看看这些方法及函数是如何应用在这些实例中的。

```
Action()
{

//The Purpose of this case is to search a string of 'AlanYuan' from Google.com.cn

  int status;
   lr_start_transaction("send");
```

```
web_reg_find("Search=Body",//这里说明在 Body 的范围内查找
            "SaveCount=return_Count",//这里表示把返回值的个数放在变量 ret_Count
              里
             "Text=mobile",//这里表示查找的内容是“mobile”
             LAST);

status=web_url("Google_Search",
   "URL=Http://www.google.com.cn/s?wd=alan_yuan" // 调用 Http 的 GeT 请求
    "TargetFrame=Main",
    "Resource=0",
    "RecContentType=text/html",
    "Mode=http",
    LAST );

lr_output_message("Request Status:%d",status);
lr_output_message("查找到的返回值个:%d",
atoi(lr_eval_string("{return_Count}")));


if (atoi(lr_eval_string("{return_Count}")) > 0){//这里判断检查到的个数
    lr_output_message("Rec successful.");
     lr_end_transaction("send", LR_PASS);
 }
 else{
     lr_error_message("Rec failed");
     lr_end_transaction("send", LR_FAIL);
 }

lr_error_message("adfasf");
```

```
  return 0;
}
```

再来看一个关于利用 Http POST 请求的实例

```
Action()
{
    int status;
     lr_start_transaction("send");

    web_reg_find("Search=Body",//这里说明在 Body 的范围内查找
                 "SaveCount=ret_Count",//这里表示把返回值的个数放在变量 ret_Count
                  里
                 "Text=t",//这里表示查找的内容是“t”
                 LAST);

    status=web_submit_data("trs",
                "Action=http://192.168.77.185:9002/recognizeText",//地址
                "Method=POST",//POST 请求
                "RecContentType=text/html",
                "Mode=HTML",
                ITEMDATA,
                "Name=usercontent","Value=gprs",ENDITEM,//这一行表示传入一个参数
                 usercontent，值为 gprs
                "Name=Accept","Value=text/plain",ENDITEM,
                LAST);

     lr_output_message("Request Status:%d",status);

    if (atoi(lr_eval_string("{ret_Count}")) > 0){//这里判断检查到的个数
        lr_output_message("Rec successful.");
```

```
        lr_end_transaction("send", LR_PASS);
    }
    else{
        lr_error_message("Rec failed");
        lr_end_transaction("send", LR_FAIL);
    }

    return 0;
}
```

上面两个例子中使用的函数前面都有详细的使用说明，请结合这两个实例再细细体会，他们是如何调用和赋值的。弄懂了这两个案例，LR 大部分的 Cases 可以拿来套用了。

小结： 从上面的案例脚本中，我们应该有这样的意识，LR 脚本基本上是靠函数的调用来完成手工模拟实现的。那些在录制过程中生成的脚本，如 Css， JSP, Asp, Jpeg, Png 等等都可以去掉，保留那些在 Response 里有期望界面的函数即可。

3.8 Vuser Generator Setting

在这个应用中，我认为最关键的设置有两个，一个是“Run_time_Setting” 另一个就是 View 了。Run_time_Setting 帮助用户设置了脚本在执行时的各个关键参数；而 View 则帮助用户完成了脚本设计过程中各种不同的可视化界面操作。为什么这样讲，主要是因为，我们在设计 LR 脚本的时候，常常会借助于 LR View 的各种视图来帮助查看执行情况和设置 CheckPoint。怎么着操作的呢，还是先上图吧。请看下图 3-2 Vuser Common Setting， 这样更直观些。

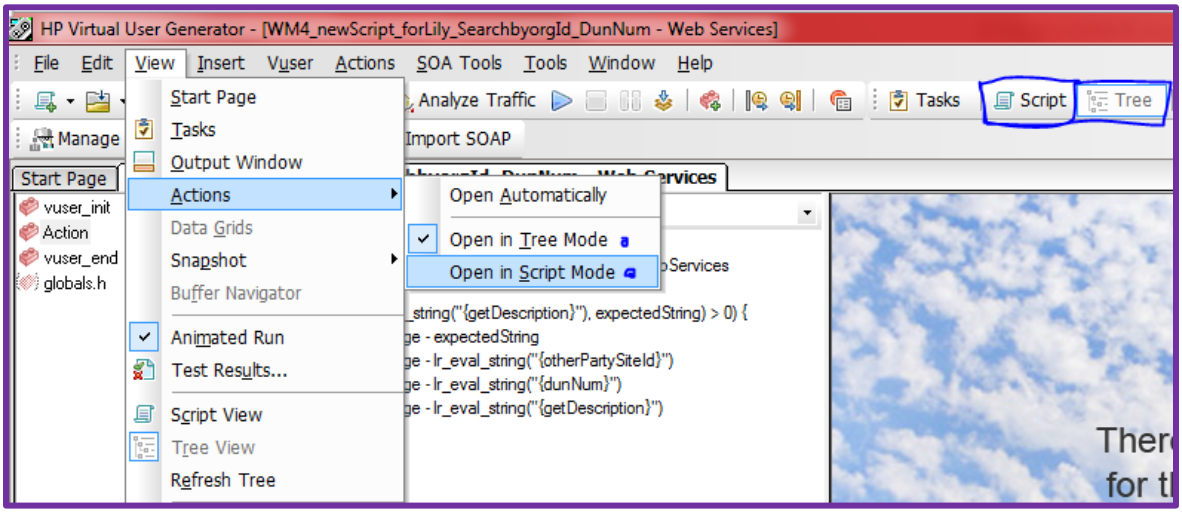

图 3-2 Vuser Common Setting

上图是我们在 Vuser Generator 中常用的视图，它有两种模式 Script 和 Tree。通常脚本代码在 Script 视图里更直观。而设置参数和 Checkpoint 时则用 Tree 视图更加方便。

Tree 视图里有三个 Table，分别是 Step Properties，Checkpoint 和 Sanpshot. 顾名思义，每个 Table 都有其专用的属性，用户可以在不同的视图里，进行不同的操作和设置。如：Step Properties 里设置各步骤的属性或参数化变量，在 Checkpoint 里设置 Checkpoint 点，在 Snapshot 视图里查看执行的状况。

另一个常用设置是 Run_time_Setting， 这个视图里可以设置执行脚本的各种约束，主要有 Run Logic， Log 和 Miscllenеous. 这些设置有空多看看，做到心中有谱就好。没必要一 一 挑选出来讲解。

3.9 Controller Brief Introduction

3.9.1 Scenario Design

在这个应用中，主要是 Scenario 的设置，也就是定义我们的压力测试方案。Loading 测试就是按照这个设置标准进行的。总的来说它分为三个步骤：Scenario Groups Define, Scenario Schedule 和 Global Schedule.

在 Groups 里，我们将要进行压力测试的脚本导入进来， 并选择执行这些脚本的服务端（Generators）；然后在 Schedule 里设置加压模式。有两个地方需要注意：一个是 Schedule by 选项，一个是 Run mode 选项。

Schedule by 如果选择 Scenario 模式，则所有的脚本（位于 Groups）将作为一个整体来加压。也就是说脚本不能串联执行。只能合并并发执行。如果选择 Group 模式，这脚本可以分开来设置执行的顺序。另外， Global Schedule 里的 Action 是可以添加和删除的，根据实际需要，用户可以自定义 Schedule 的 action 步骤。

3.9.2 Run

在这个 Table 里，主要是执行和监控各个指标参数的。将需要监控的指标/参数从 Availablelist 中拖拽到右方窗口即可，双击也可以添加的。其他的，我觉得没什么，多观察测试过程中各个参数的变化。进一步了解各个参数的监控目的才是这部分的重点。

3.10 Analysis on Report

分析测试结果是一项特别需要内力的技术活。他需要综合软硬件知识和数据库配置方案等各项技术。这对测试人员来说无疑是个挑战。不期望一开始大家都能完全透析各项测试指标。但最基本的测试项目反馈，应该有所了解。下面一起来看看常用的分析过程。

3.10.1 查看测试异常和原因分析

在 LoadRunner Controller 里，可以监控各种测试指标。当有错误发生时，可即时查阅错误信息。Controller 会对对各种错误类型给出预定义。测试者可以查询相关错误信息，来判明错误根源。请看下图 3-2 LR 错误处理。

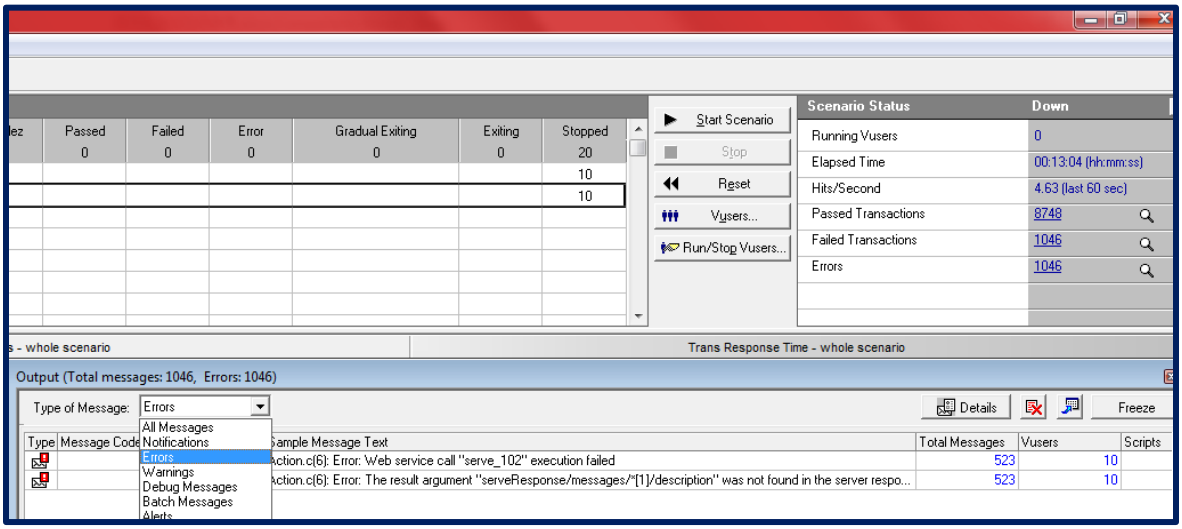

图 3-2 LR 错误处理

在 Output 视窗里，可以继续查看 Error Message 的 details 信息。Loader Runner 将错误信息进行了不同侧面的封装。用户可以通过各种不同的 Filter 来获取所需的错误描述（如：filter by ‘Total Messages’; filter by ‘Vusers’; filter by ‘Scripts’; filter by Generators 等等）。是不是很强大?

3.10.2 不同的输出视图

这个其实就是在 Analysis 应用中使用不同的 Filter，使得测试结果呈现出不同的侧面。如何使用 Filter，有哪些常用的操作呢？一般测试结果后，测试结果就保存在

用户定义的路径下，并以 Lrr 为后缀名。结果分析需要在 LR 的 Analysis 应用中实现。点击 Analysis 打开测试结果。根据不同的需求，做出不同的结果展现。

通常在 Summary Report 里给出了测试结果的统计。Graphs 里面是各个监控参数/指标的示意图。可以分别进入观察。在下面的属性栏里（Properties）列出了各个参数/指标的操作因素。根据这些可操作因素，用户可以整理出不同侧面的测试结果。常用的 Properties 有：Filter， Granularity，Group by 等。根据这些可操作因素，可以让测试结果呈现出不同的表达侧面。例如，在 Controller 里设置了多个脚本的 Scenario，那么用户可以在 Group By 里选择不同的脚本就可以单独拎出某个脚本的压力测试结果。如下图 3-3 '测试结果分析' 所示。

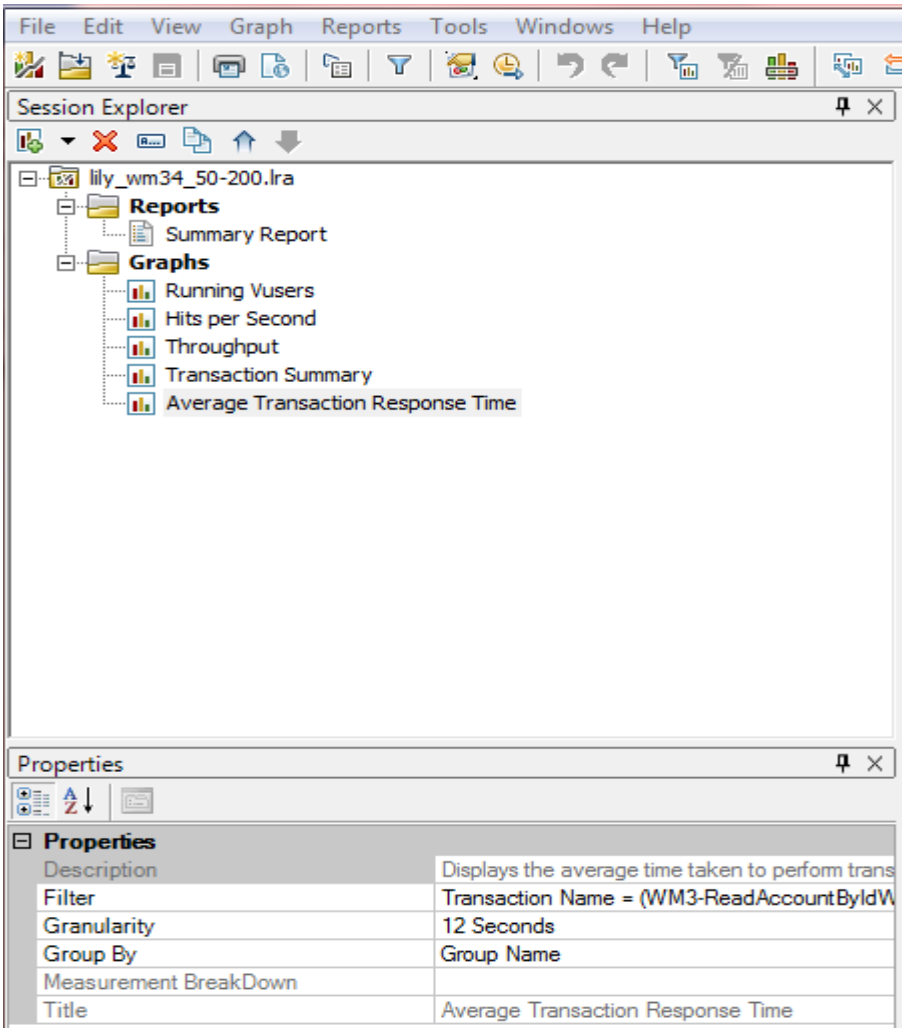

图 3-3 测试结果分析

Granularity 可以调节参数图形曲线的监控颗粒度大小。
Filter 可以设置不同的筛选条件，找出需要的 Transaction 指标

另外，在 Report Templates 选项中，我们还可以定制测试 Report 的呈现格式，比喻字体呀，需要显示的 Item 呀，颜色等等的。都可以轻松的设置。具体请参考截图。

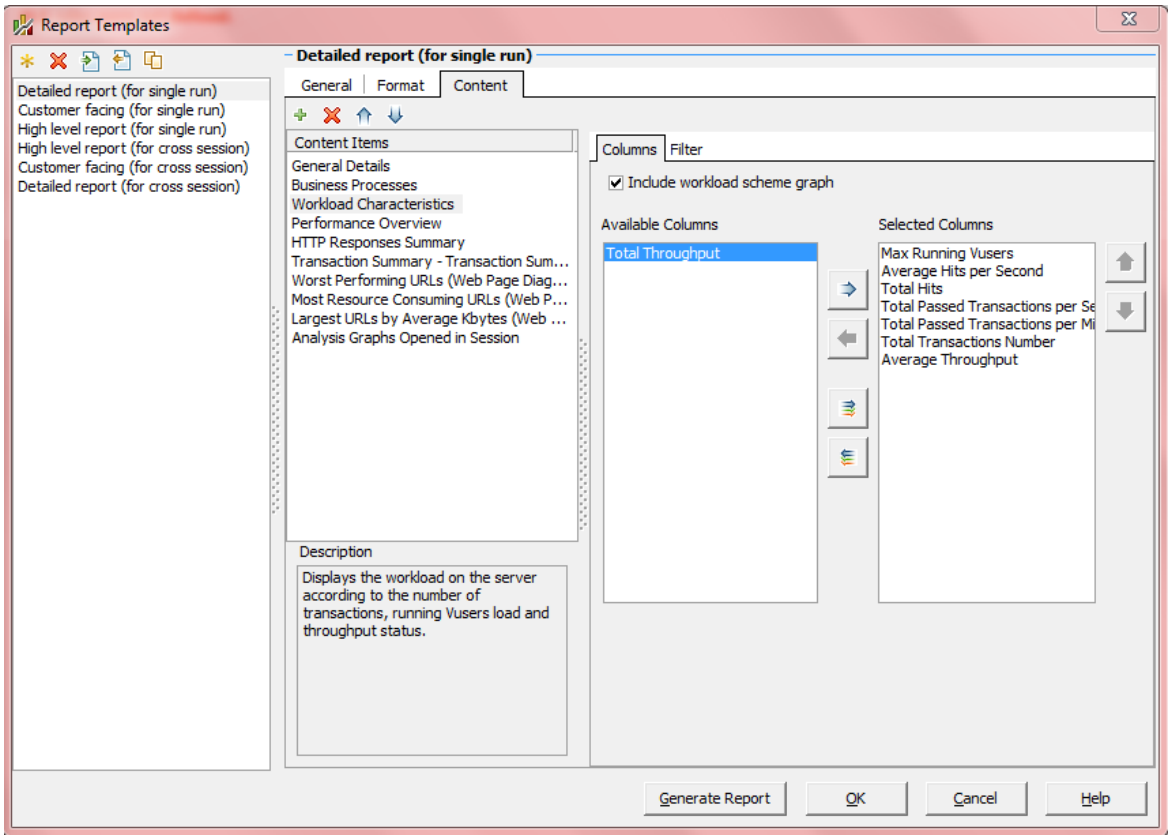

图 4-4 客户化测试报告

好了，差不多大概描述了一下 LR 的三大应用程序，Vuser Generator，Controller 和 Analysis 的基本功能和常用设置。任何一个压力测试都是由这三个应用是协同工作来完成的。用户在 Vuser Generator 里编写 PT 脚本，并调试。在 Controller 里设置 loading 条件，并按设置的条件执行导入脚本的压力测试。最后在 Analysis 里分析测试结果。整理出客户需要的测试报告。

3.11 LR 测试 WebServices 的三种方法介绍

3.11.1 Web_Service_Call()

LR 测试 WS 类 Cases 是比较常用的 PT 测试， LR 里提供了多种方法来发送 XML 文件，这也使得我们有多种途径实现 LR 对 WS 的操作。先来看看第一种方法 Web_Service_Call()

这个方法最常用，步骤为： 在 Vuser Generator 里选择 WebServices Protocol，然后 import WS（在 import URL 里填入 WSDL URL 即可），接着就是添加新的 WS 并初始化 input 和 output，加入 checkpoint 等。（具体实现可参考上面的 CaseOne）

3.11.2 Soap_Request()

此方法是先保存 WS 为 XML 文件，然后利用 LR 的 'Import Soap' button 来导入这个 XML 文件，并给出 WSDL URL 参数。这里主要是要弄明白 Soap_Request 这个函数的用法，请看下面代码：

```
Action()
{
char *expectedResult = "CUSTOMER";

lr_start_transaction("SalesAccount");
        soap_request("StepName=SOAP Request",
        "URL=http://mdm-itg.houston.hp.com:8180/customer-、
        \manager/services/SalesAccountReadWebService?wsdl",
        "SOAPEnvelope="
        "<soapenv:Envelope
        xmlns:soapenv=\"http://schemas.xmlsoap.org/soap/envelope/\"
        \xmlns:web=\"http://webservice.salesAccount.mdm.it.hp.com/\">"
        "<soapenv:Header></soapenv:Header>"
        "<soapenv:Body>"
        "<web:SalesAccount>"
        "<BusinessRelationshipId>50001313O</BusinessRelationshipId>"
        "<BusinessRelationshipNaturalKey>"
```

```
"<BusinessRelationshipAppliesToCode></BusinessRelationshipAppliesToCode>"
"<OrganizationID></OrganizationID>"
"<PanHPFlag></PanHPFlag>"
"<BusinessAreaGroupId></BusinessAreaGroupId>"
"<BusinessRelationshipUnitName></BusinessRelationshipUnitName>"
"<BusinessRelationshipTypeCode></BusinessRelationshipTypeCode>"
"</BusinessRelationshipNaturalKey>"
"</web:SalesAccount>"
"</soapenv:Body>"
"</soapenv:Envelope>",
"Snapshot=t1407919024.inf",
"ResponseParam=response",
LAST);

if(strstr(lr_eval_string("{response}"), expectedResult) > 0){
lr_end_transaction("SalesAccount", LR_PASS);
}else{
lr_end_transaction("SalesAccount", LR_FAIL);
}
return 0;
}
```

ResponseParam= response 说明将返回的内容记录在 response 参数里。依据这个返回参数值，我们可以写后面的判断语句了。其他跟 Web_Service_call 方法类似。

3.12 补充案例

上面介绍了 LR 的常用函数功能，一般脚本开发步骤和常见的写法以及测试结果的整理等等。应该说涵盖了大部分常见的 LR 测试情景。用户可以参考这些用例不断的练习和实践。达到加深巩固之目的。这里我再介绍下另一些有用的案例，也希望能对初学者有所帮助，所有的案例介绍只是起到抛砖引玉的作用，在实际应用中不可完全照搬。

跟用 QTP/UFT 做 Regression 测试一样，有一类 Case 经常要访问数据库，然后对数据库进行操作，在 LR 中，我们也经常碰到类似的情况，那么如何对此类 Case 做 Performance 测试呢？其实步骤差不多，区别就在于如何访问数据库和如何将对数据库的操作转化为脚本的执行。对不对？这里我设计了一个案例，差不多能 Cover 到这种情况。一起来看看是如何编写脚本的。

Addition 1：LR 如何操作数据库

此案例执行过程，大致可分为如下几步：

1.> 根据条件，搜索数据库表 1，得到测试数据。

2.> 根据条件，搜索数据库表 2，看是否存在表 1 的记录，如果存在则删除。

3.> 执行 WS，并用步骤 1 中的数据初始化。

4.> 度量 WS 的 Response time

脚本实现如下 –

a.) 使用 lr_db_connect()实现数据库连接

```
lr_db_connect("StepName=Connect",

"ConnectionString=Provider=OraOLEDB.Oracle; Data Source=HPMDMI; ”
“Persist Security Info=True; User ID=mdm_extend; Password=password_1234",

"ConnectionName=db1",
"ConnectionType=OLEDB",
LAST );
```

需要注意的是，不同数据库 Provider 和 ConnectionType 会不一样。另外，如果一行写不下时可以另起一行写，两行均需双引号括起来，在第二行后加逗点。

b.) 查询数据库表 I 里的特定记录， 使用函数 lr_db_executeSQLStatment()

```
getRecordQty = lr_db_executeSQLStatement("StepName="
"Query_MDM_EXTEN.ALIGNMENT_OUTPUT_TBL",

"ConnectionName=db1",

"SQLStatement=select * from ALIGNMENT_OUTPUT_TBL where"
"ALIGNMENT_RESULT_MESSAGE = 'Matched Site' and"
"ALIGNED_COUNTRY_CODE = 'CA' and rownum < 10",

"DatasetName=myDataset",
LAST);

lr_output_message("How many records did we get from this search: %d",
getRecordQty);
```

注意，个语句的换行写法，每一条独立语句以逗点结束，整个语句完成以分号结束。每条独立语句要分行写时，上面一行需要引号，无须逗点，下面一行才需逗点分隔。

c.) 从查询结果中得到需要的字段值（取第一行值）， 需要调用函数 lr_db_getvalue()

```
lr_db_getvalue("StepName=GetValue",
 "DatasetName=myDataset",
"Column=RET_SRC_FIELDS_FLAG",
"Row=1",
"Outparam=srcCompanyName",
LAST);
```

lr_output_message("Output the Field Value from the return records, the SRC_COMPANY_NAME = %s", lr_eval_string("{srcCompanyName}"));

d.) 同样找出 'SRC_ADDR_LINE_1' 的值

lr_db_getvalue("StepName=GetValue",
"DatasetName=myDataset",
"Column=SRC_ADDR_LINE_1",
"Row=1",
"Outparam=srcAddline1",
LAST);

lr_output_message("Output the Field Value from the return records, the SRC_ADDR_LINE_1 = %s", lr_eval_string("{srcAddline1}"));

e.) 搜索表 2，看是否存在表 1 里的特定记录

NumRows = lr_db_executeSQLStatement("StepName="
"Search_CMX_ORS.C_algnmnt_LRN ",
"ConnectionName=db1",

"SQLStatement=Select * From CMX_ORS.C_algnmnt_LRN where SRC_ORG_NM"
"='{srcCompanyName}'and SRC_ADDR_LN_1 ='{srcAddline1}'",

"DatasetName=myDataset",
LAST);

lr_output_message("Return the search result from CMX_ORS.C_Algnmnt_LRN %d rows.", NumRows);

f.) 根据需求，删除表 2 里的特定记录

NumRows =
lr_db_executeSQLStatement("StepName=Search_CMX_ORS.C_algnmnt_LRN ",

```
"ConnectionName=db1",
"SQLStatement=Delete From CMX_ORS.C_algnmnt_LRN where SRC_ORG_NM =”
‘‘{srcCompanyName}' and SRC_ADDR_LN_1='{srcAddline1}'",

"DatasetName=myDataset",
LAST);

lr_output_message("Delete the return record from ALignment sheet'
CMX_ORS.C_Algnmnt_LRN  %d rows:'", NumRows);
```

完成上面的步骤后，我们已经清楚了表 2 中的特定记录，接下来要做的就是执行 WS，并用表 1 中的数据初始化。到了这一步就可以参考上面介绍的 WS 脚本编写步骤去做，代码如下。

```
web_service_call( "StepName=getFlexableWorkFlowResult_101",

"SOAPMethod=AlignmentServiceForWsdlRestService|AlignmentServiceForWsdlRestPort
|getFlexableWorkFlowResult",
        "ResponseParam=response",
        "Service=AlignmentServiceForWsdlRestService",
        "URL=http://d9t0676g.houston.hp.com:8180/alignment/rest/call",
        "ExpectedResponse=SoapResult",
        "Snapshot=t1406602171.inf",
        BEGIN_ARGUMENTS,
        "xml:getFlexableWorkFlowResult1="
            "<getFlexableWorkFlowResult1>"
                "<regAppId>TEST</regAppId>"
                "<retSrcFieldsFlag>{retsrcfieldsFlag}</retSrcFieldsFlag>"
                "<srcAddrId>{srcAddId}</srcAddrId>"
                "<srcAddrLine1>{srcAddline1}</srcAddrLine1>"
                "<srcAddrLine2>{srcAddrLine2}</srcAddrLine2>"
                "<srcAddrLine3>{srcAddrLine3}</srcAddrLine3>"
```

```
                    "<srcCityNm>{srcCityNm}</srcCityNm>"
                    "<srcCountryCode>{srcCtryCD}</srcCountryCode>"

    "<srcOrganizationNm>{srcCompanyName}</srcOrganizationNm>"
                        "<srcPostalCode>{srcPostCD}</srcPostalCode>"

    "<srcStateProvNm>{srcSTAProNM}</srcStateProvNm>"
                        "<srcTxnKey1>Test_PT</srcTxnKey1>"
                    "</getFlexableWorkFlowResult1>",
                END_ARGUMENTS,
                BEGIN_RESULT,
                END_RESULT,
                BEGIN_CHECKPOINTS,
                    CHECKPOINT,
    "XPATH=getFlexableWorkFlowResultResponse[1]/reasonCode[1]",
    "Value=MATCH-SITE",
                END_CHECKPOINTS,

LAST);
```

到此为止，我们完成了这个案例的全部脚本。通过上面介绍的几个函数（从 a 到 f）我们几乎可以测试 Oracle 数据库下的任何 Performance Cases. 索不同的是 SQL 语句和表单的差异而已。