

基于 SoapUI 的接口教程

日期	修订版本	修改内容	作者/变更
2013 年 11 月 12 日	V1.0	初稿（接口的功能、自动化、性能的测试实现	高学文 QQ:1253530678
2013 年 12 月 26 日	V1.1	修改部分文字 bug,完善自动化部分的断言功能	高学文
2014 年 5 月 16 日	V1.2	修改部分文字 bug	高学文
2015 年 3 月 14 日	V1.3	增加天气预报中 theUserID 获取方法，补充传参常见错误	高学文

注：初稿中可能还存在一些文字方面和少部分功能上标注的 BUG，欢迎有经验的同时交流分享。教程分为五个部分：基本概念、接口功能测试、接口自动化测试、接口性能测试、小结等。

目录

一 关于 SoapUI 基本概念	1
1.1 SoapUI 简介	1
1.2 WSDL 介绍	1
1.2.1 WSDL 定义.....	1
1.3 下载与安装.....	1
1.3.1 下载.....	1
1.3.2 安装前准备	2
1.3.3 安装 SoapUI	2
二、使用 soapUI 测试 Webservice 接口	5
2.1 创建一个项目	5
2.2、SoapUI 项目结构简述	7
2.3 开始一个测试：	8
2.3.1WeatherWebServiceSoap 服务的集合.....	8
2.3.2WeatherWebServiceSoap TestSuite 测试用例的集合.....	9
2.4 Properties 变量的使用	10
2.5 手工添加断言	11
2.6 Transferring Property Values 传递属性值.....	14
2.7 变量的存储.....	17
三、实现基于 Excel 表的自动化测试。	19
四、性能测试	24

一 关于 SoapUI 基本概念

1.1 SoapUI 简介

实际项目中，大多 Web 服务是通过接口调用实现，通常不会提供界面让最终用户或测试人员直接使用，因此给测试工作带来了麻烦，测试人员不得不自己编写代码来测试它。SoapUI 是一款强大且易用性强的接口功能与性能测试工具。测试人员可以通过 SoapUI 操作完成基于 SOAP 的 Web 服务和 REST 风格的 Web 服务。Webservice 是我们比较常见的接口之一，本文重点围绕 Webservice 接口开展案例介绍。

1.2 WSDL 介绍

WSDL (Web Services Description Language, Web 服务描述语言) 是一种 XML Application，他将 Web 服务描述定义为一组服务访问点，客户端可以通过这些服务访问点对包含面向文档信息或面向过程调用的服务进行访问(类似远程过程调用)。WSDL 首先对访问的操作和访问时使用的请求/响应消息进行抽象描述，然后将其绑定到具体的传输协议和消息格式上以最终定义具体部署的服务访问点。相关的具体部署的服务访问点通过组合就成为抽象的 Web 服务。本文将详细讲解 WSDL 文档的结构，并分析每个元素的作用。

1.2.1 WSDL 定义

WSDL 是一个用于精确描述 Web 服务的文档，WSDL 文档是一个遵循 WSDL XML 模式的 XML 文档。WSDL 文档将 Web 服务定义为服务访问点或端口的集合。在 WSDL 中，由于服务访问点和消息的抽象定义已从具体的服务部署或数据格式绑定中分离出来，因此可以对抽象定义进行再次使用：消息，指对交换数据的抽象描述；而端口类型，指操作的抽象集合。用于特定端口类型的具体协议和数据格式规范构成了可以再次使用的绑定。将 Web 访问地址与可再次使用的绑定相关联，可以定义一个端口，而端口的集合则定义为服务。

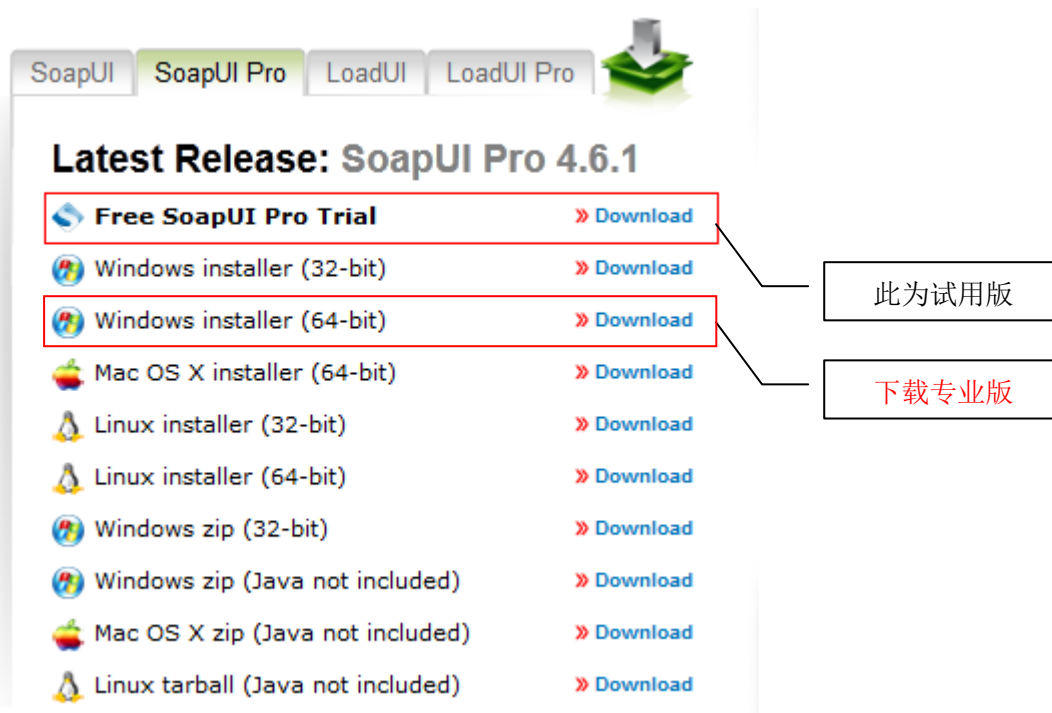
一个 WSDL 文档通常包含 7 个重要的元素，即 types、import、message、portType、operation、binding、service 元素。这些元素嵌套在 definitions 元素中，definitions 是 WSDL 文档的根元素。

1.3 下载与安装

1.3.1 下载

我们可以打开 SoapUI 的官网地址：<http://www.soapui.org/>，官网首页提供了多种平台下的安装包下载，主要分为两种不同的版本：

版本	介绍
SoapUI Pro	收费版，拥有强大功能本文主要围绕该版本做功能介绍。
Free SoapUI Pro Trial	SoapUI Pro 的试用版，下载后拥有 14 天的免费试用期。
SoapUI	开源版本，功能相对较少，可以通过网上下载源代码，可以通过 groovy 二次开发。



1.3.2 安装前准备

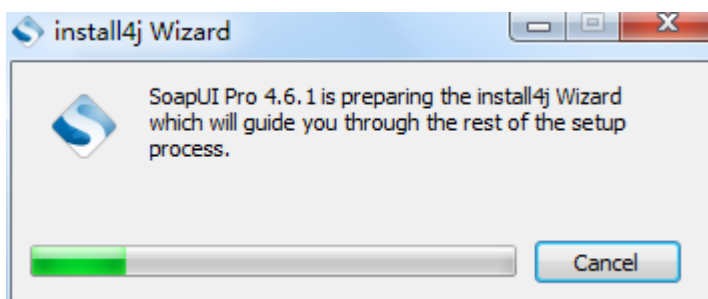
由于 SoapUI 基于 JAVA 开发，所以安装前需要本机拥有 JAVA 环境，JDK6.0 下载地址：

<http://www.oracle.com/technetwork/java/javasebusiness/downloads/java-archive-downloads-javase6-419409.html>

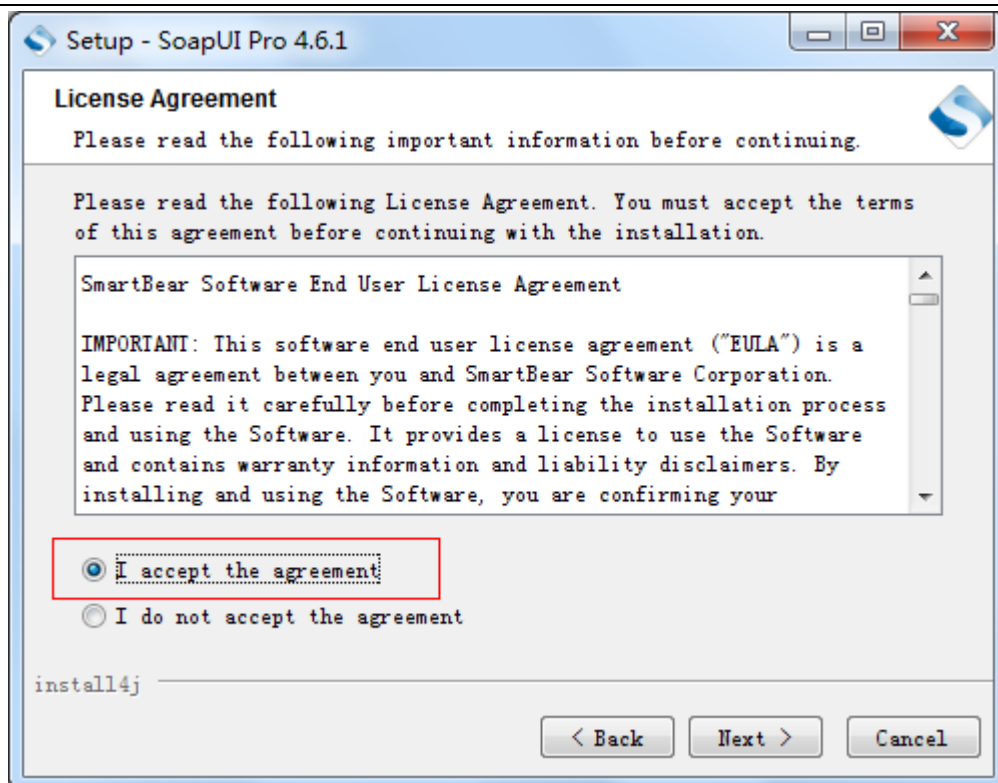
JDK 具体安装过程，请详见附件一：《JDK 安装教程》

1.3.3 安装 SoapUI

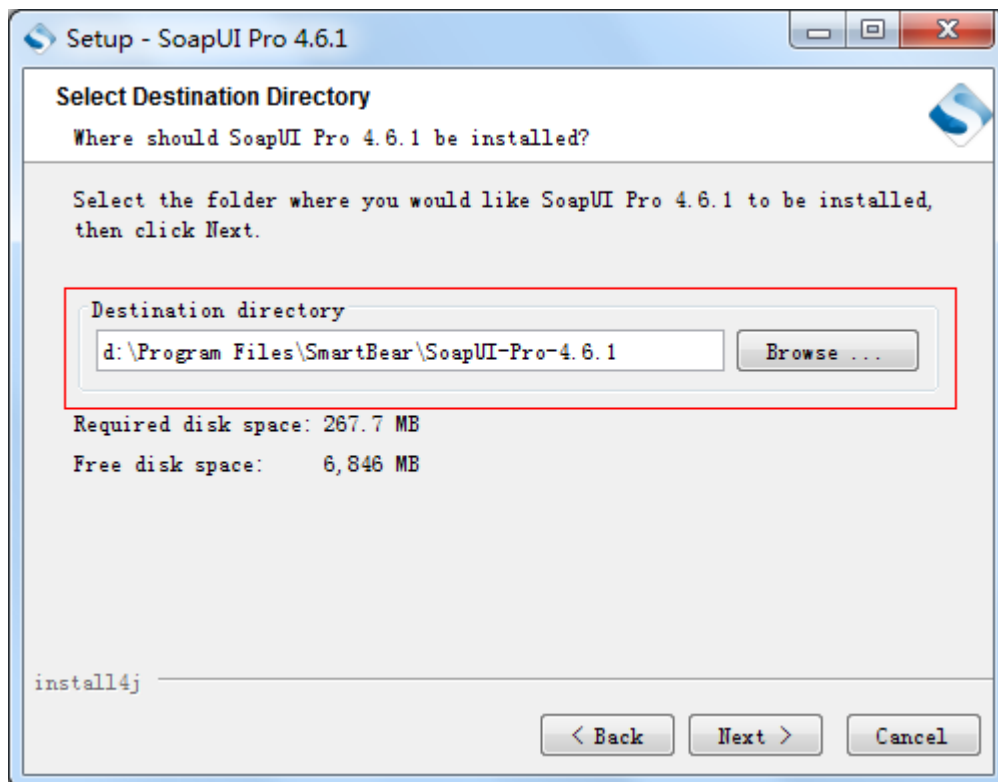
1、双击 SoapUI-Pro-x64-4.6.1（笔者使用 for Win7 64bit 版本）



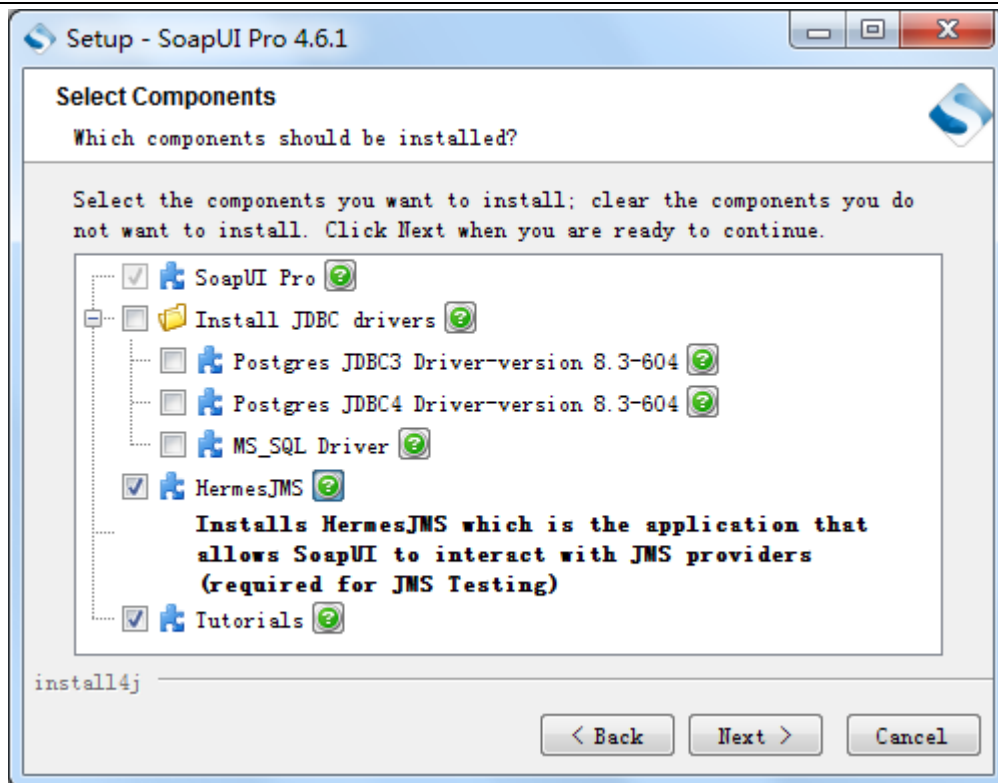
2、选择“接受”并进行下一步



3、选择安装目录，建议安装在非系统目录下，笔者安装在 D 盘下



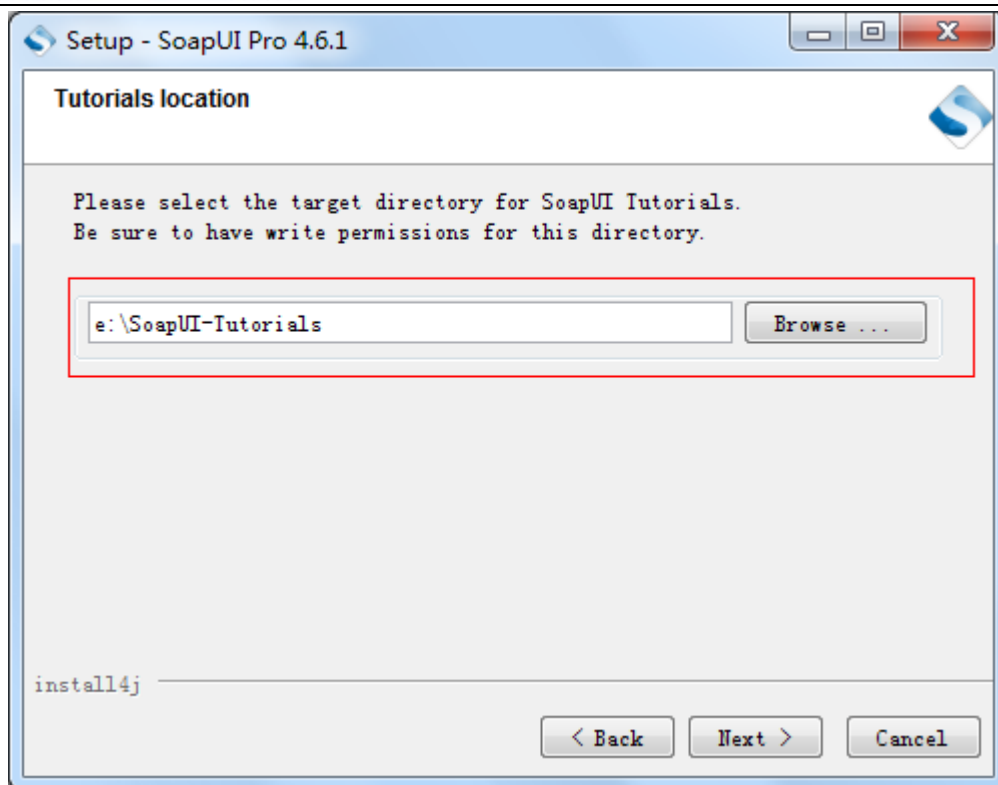
4、在此可选 JDBC 驱动，在此按默认选择即可，下一步



5、选择是否安装 LoadUI（高级性能测试工具）,在此勾选 Yes



6、选择 Tutorials location 安装目录，下一步，直至提示 finished 完成（切记该目录地址，下面的部分案例会使用到）



二、使用 soapUI 测试 Webservice 接口

2.1 创建一个项目

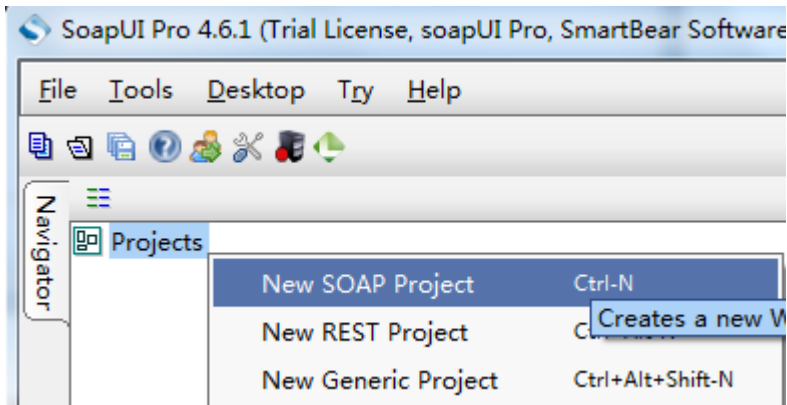
为了方便与朋友们分享该工具的使用方法，笔者采用中国气象局提供的 WSDL，WSDL 通常是一个地址或是一个文件，可以跟开发人员要相关接口的 WSDL 地址或文件，也可以通过 Loadrunner 等工具自动获取 WSDL。作为测试的我们，在服务器端给定我们访问地址后，我们如何生成自己的客户端呢？

给定的地址一般分为以下两种：

第一，`http://.../WapInterface`；我们可以通过在其后加 `.wsdl`（`http://.../WapInterface?wsdl`）获取 wsdl 内容，通过另存为 `.wsdl` 即可获取 wsdl 文件。

第二，`http://...//GovOnline.asmx`；我们使用同样的方法 `http://...//GovOnline.asmx?wsdl` 即可方便的得到 wsdl 文件。

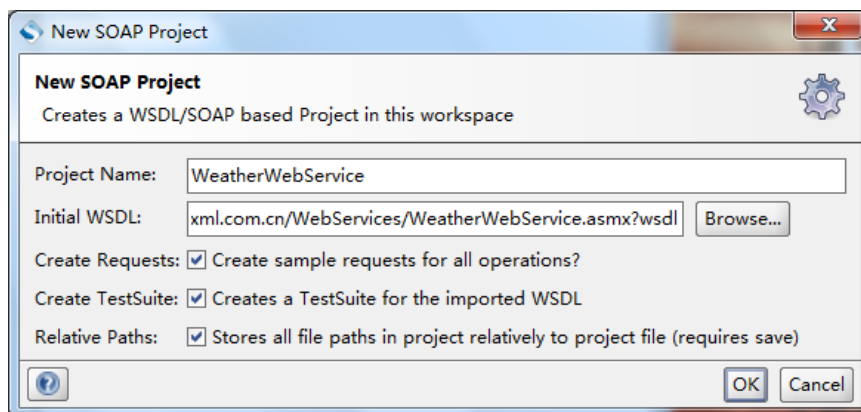
【第一步】：在 Projects 上右键，选择 “New SOAP Project”



【第二步】：在 Initial WSDL 处粘贴中国气象台 Webservice 地址：

<http://www.webxml.com.cn/WebServices/WeatherWebService.asmx?wsdl>

（截至本文再次修改时，原接口的 `getWeatherbyCityName` 方法已经无法使用，但是 `getWeatherbyCityNamePro` 方法仍可使用，两种方法都是可以获取天气信息，请读者在实际操作中对对应替换为该方法。`getWeatherbyCityNamePro` 需要在 webxml.com.cn 注册一个用户，登录进入后，获取“WEB 服务 用户 ID”在“我的 WEB 服务”中点击试用，其中“用户 ID”就是 `getWeatherbyCityNamePro` 方法中的 `theUserID` 应该填写的值）



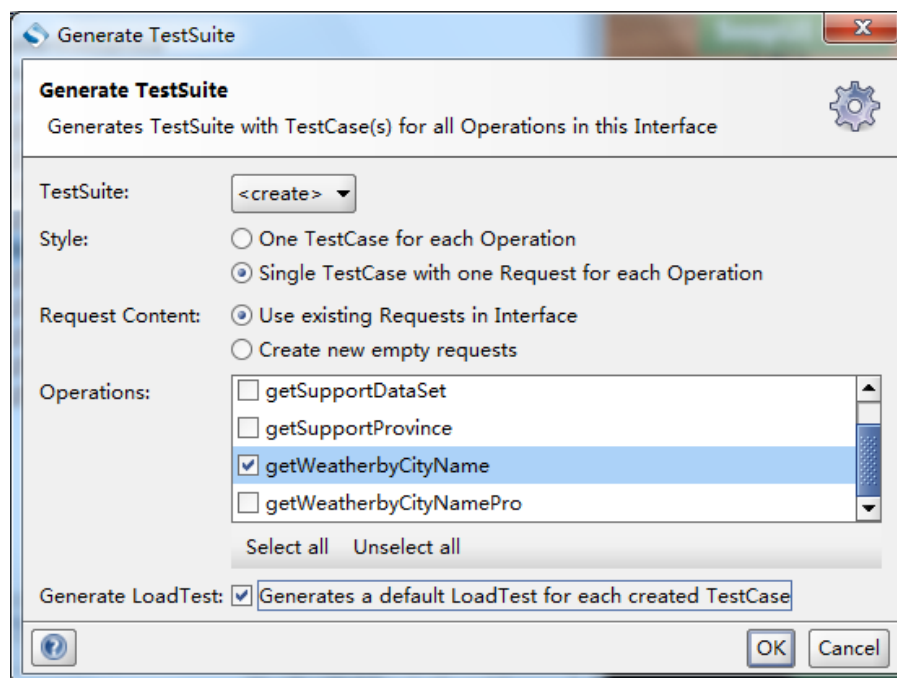
说明：

Create sample requests for all operations?: 为每个接口创建一个请求的例子

Creates a TestSuite for the imported WSDL or WADL: 为 WSDL 或 WADL 创建一个测试包

Stores all file paths in project relatively to project file: 保存项目中的相对项目文件

【第三步】选择创建方式及要测试的接口，以完成项目的创建。



说明：

One TestCase for each Operation: 每个接口创建一个用例

Single TestCase with one Request for each Operation: 创建一个用例包含每个接口对应的请求

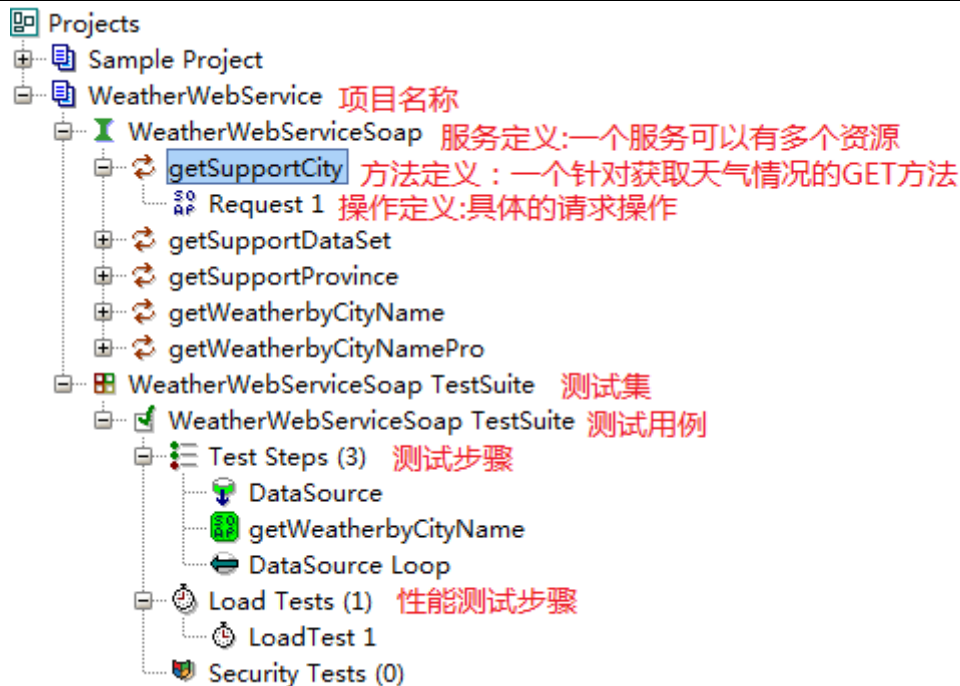
Use existing Requests in Interface: 使用已有的请求

Create new empty requests: 创建一个空的请求

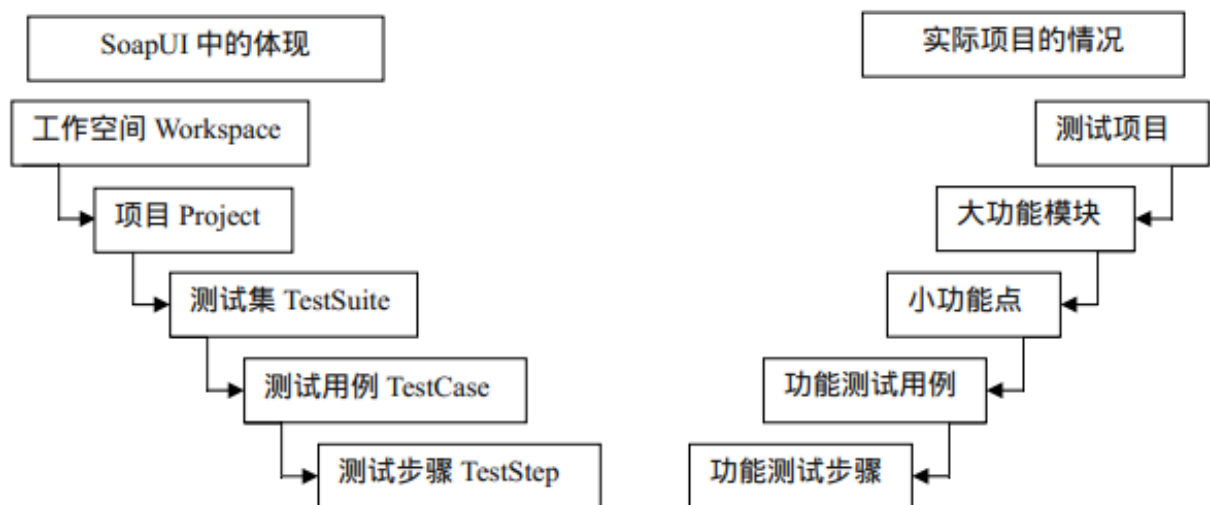
Generates a default LoadTest for each created TestCase: 每个用例生成一个负载测试

2.2、SoapUI 项目结构简述

完成之后会在左边的树形结构中生成 2 部分：服务定义和测试集。



读者会发现: 它所提供的测试用例的管理与我们测试项目所需要的层级的映射关系是相当贴近的, 对应关系图如下

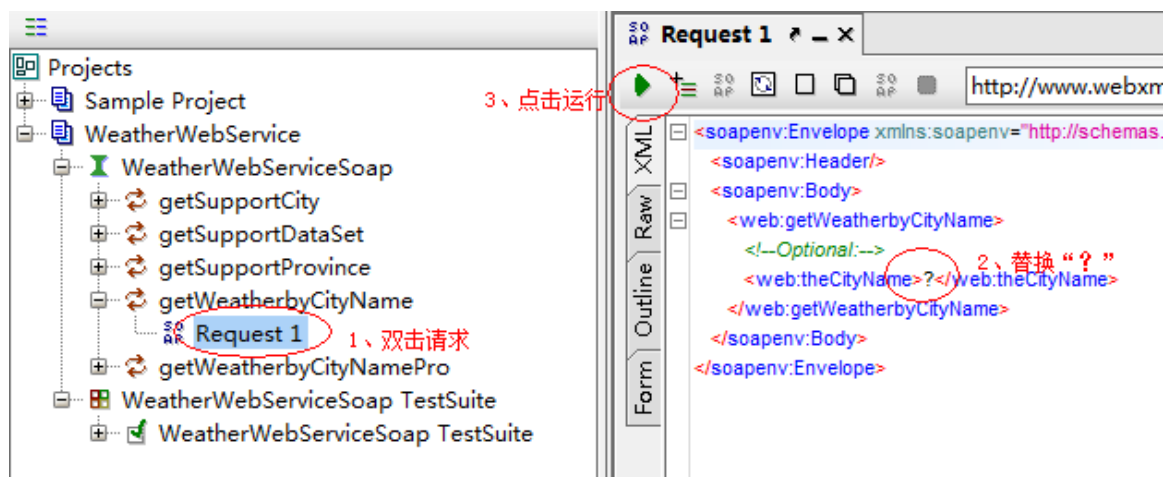


2.3 开始一个测试:

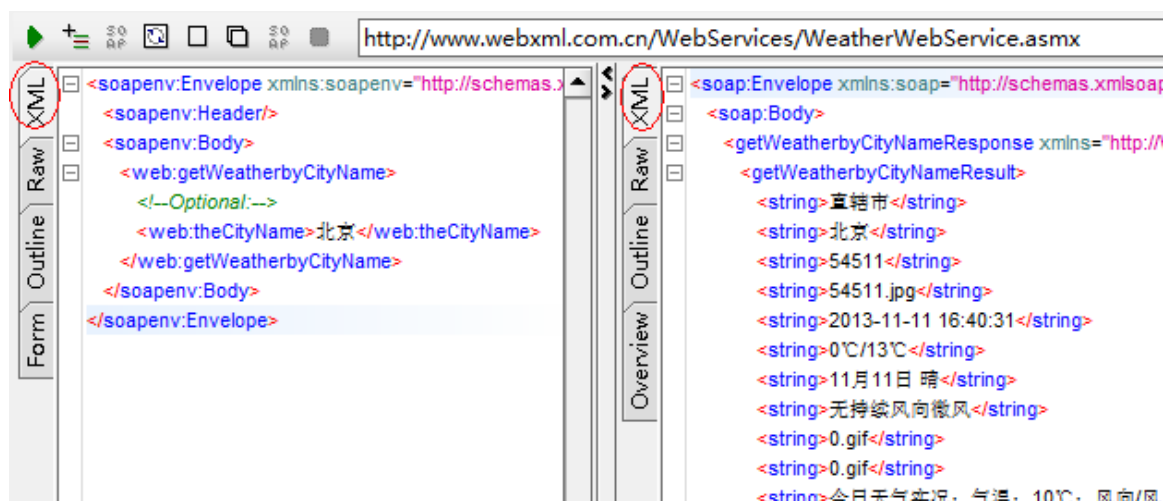
2.3.1 WeatherWebServiceSoap 服务的集合

创建项目的时候我们选择了 Create sample requests for all operations, 所以每个接口都会自动创建一个请求, 双击它就可以打开编辑面板, 左边是请求内容, 右边是响应内容。

把每个节点的“?”替换成需要的内容, 点击绿色的箭头发送就可以了。右边的内容就是服务器返回的结果, 同时可以看到系统后台有相同的日志显示。



运行后返回的结果：显示北京地区的天气情况



(当然，如果你是正式版，在 Form 中可以直接填写你要查询的字符。)

2.3.2 WeatherWebServiceSoap TestSuite 测试用例的集合

TestSuite 是测试用例的集合，且里面每个测试用例包含测试步骤和负载测试。负载测试可以测试响应时间并统计测试结果。这里暂不讨论。

在创建时已经自动给每个接口生成了一个发送请求的测试步骤，如上图一样，初始的节点内容是“？”，要替换掉。



【WeatherWebServiceSoap 服务集合】与【WeatherWebServiceSoap TestSuite 测试用例的集】的区别：“服务集合”更适合手工临时验证接口，而“测试用例集”即可以临时完成手工测试，而且还能形成固定用例，实现参数化，做接口自动化、性能测试工作。

2.4 Properties 变量的使用

除了手工在请求中修改“？”之外，我们还可以通过“Properties”来“参数化”处理。这样就更方便我们的测试工作。具体使用方法如下：

【步骤一】建立 Properties：



细心的读者会发现，在选择“Add Step”后，弹出的内容非常多，在此做个说明：

Test Request：发送一个 soap 请求

Groovy Script：用 Groovy 脚本定义的步骤。Groovy 是一种脚本语言，语法跟 java 类似。

Properties：定义变量/属性

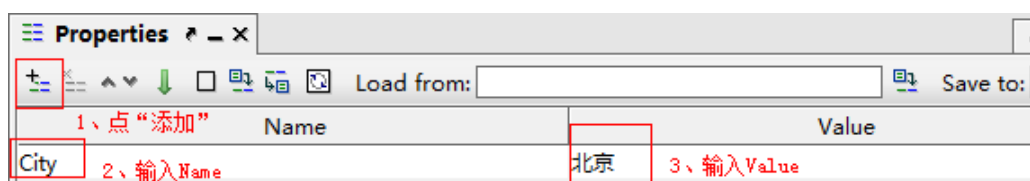
Property Transfer：传值。可以把指定的属性的值传给另一个属性，也可以给请求中节点赋值。

Conditional Goto：跳转，符合一定条件就跳到第 N 步

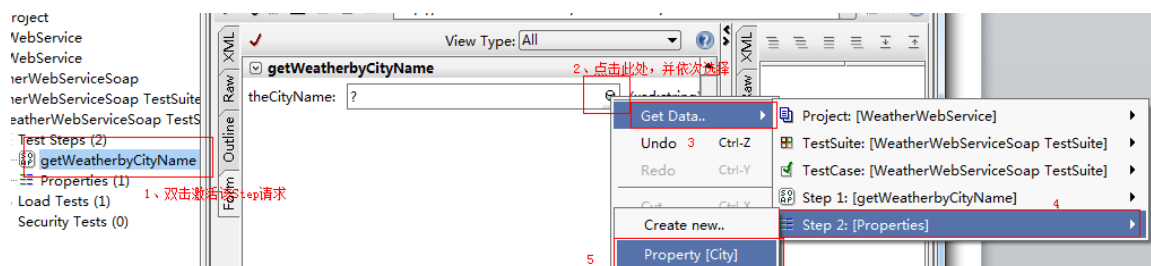
Delay：延迟，可以调整用例执行时间，模拟人工思考时间。

Run TestCase：在用例中执行另一个用例。

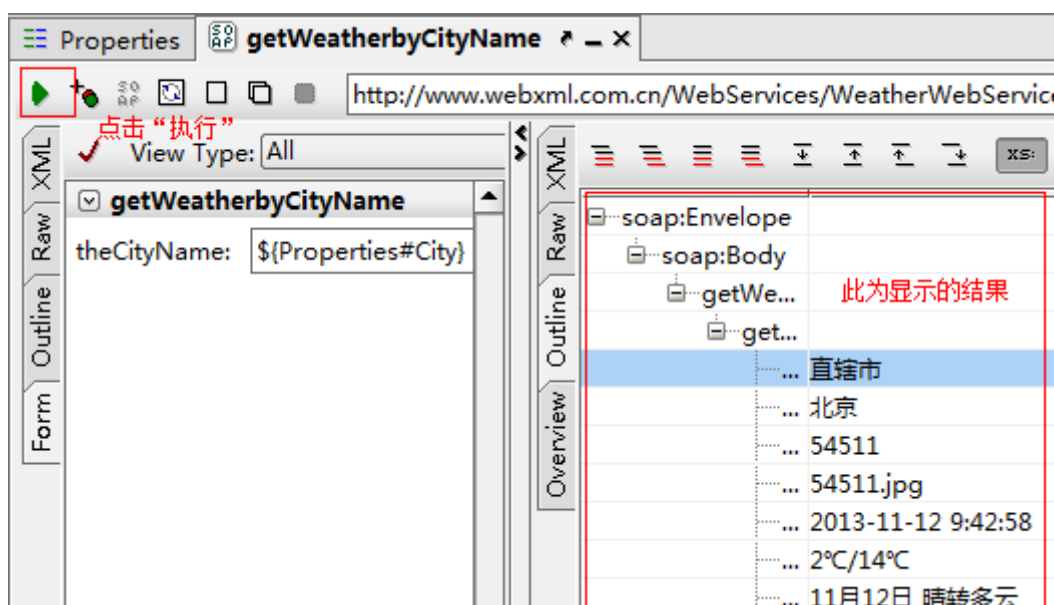
【步骤二】：输入“属性”名称并确定，输入“参数化”的名字和取值：



【步骤三】在 `getWeatherbyCityName` 中引用该参数化属性：



【步骤四】：系统提示是否用该参数化属性替代“？”，点“确定后”，就可以执行了。

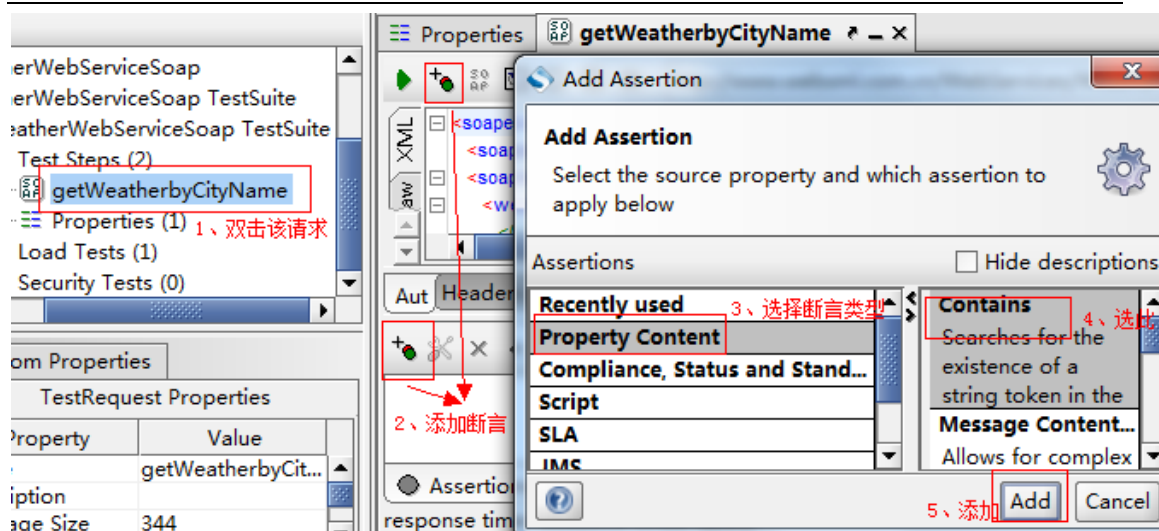


2.5 手工添加断言

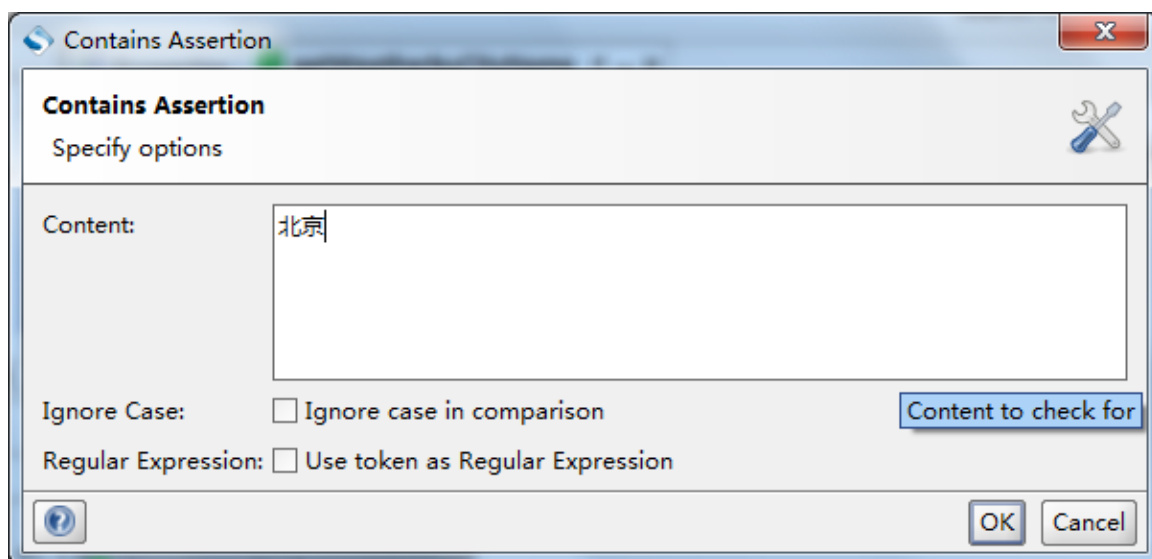
一个用例在书写时，往往包含“预期结果”，那在 SoapUI 中怎么来实现呢？在此，我们需要添加“断言”，也就是检查点。一旦用例执行后，系统可以根据检查点（预期结果）来判断该条用例是否 PASS。

在此，我们延续上一个请求，给 `getWeatherbyCityName` 这步添加“断言”：

【步骤一】：

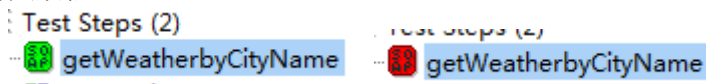


【步骤二】：添加要检查的值



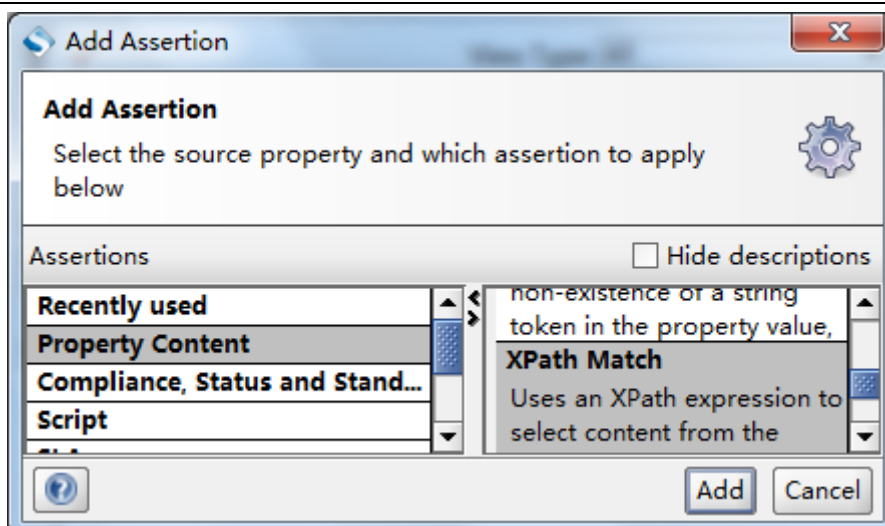
【步骤三】：执行  getWeatherbyCityName 校验结果。

如果执行通过，“getWeatherbyCityName”会变成绿色，否则为红色。错误时可以在 log 查看失败信息。

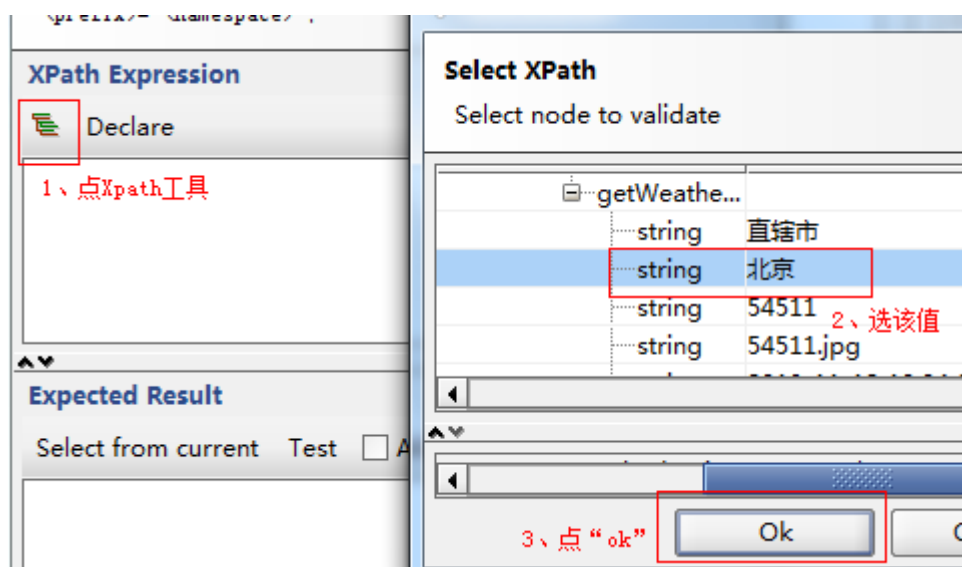


Contains 比较简单，只要指定包含的文本内容即可。如果系统返回的 XML 中，包含多个要检查的字符，我们要检查某一个特定属性的值该怎么办呢？下面介绍一下 Xpath Match 的用法：

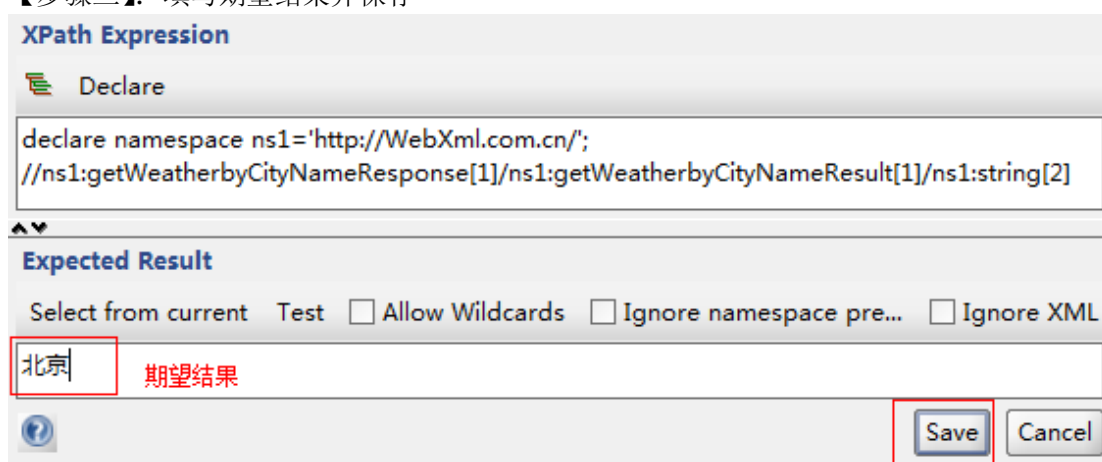
【步骤一】：选择 Xpath 方法校验：



【步骤二】:



【步骤三】: 填写期望结果并保存



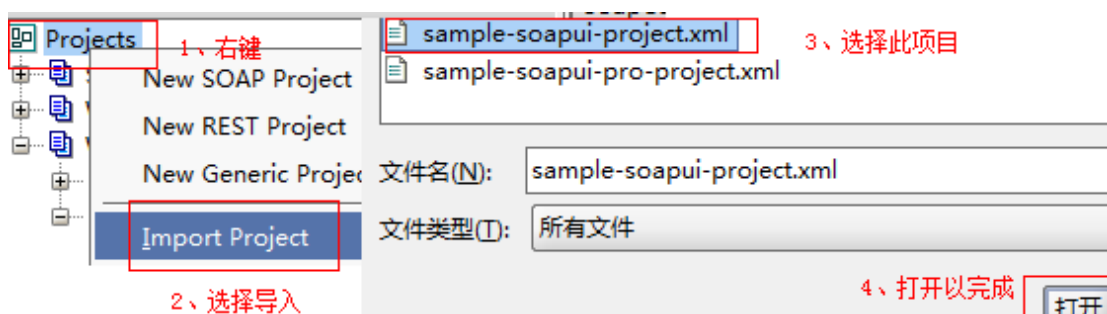
简要说明: 上图 declare 部分为自动生成, 意思是获取上一部中第二个 String 中的实际值, Expected Result 这个节点是返回结果编码。下面的“北京” 是这个节点的期望值。如果返回“北京”表示查询成功。

【步骤四】：执行并观察结果。

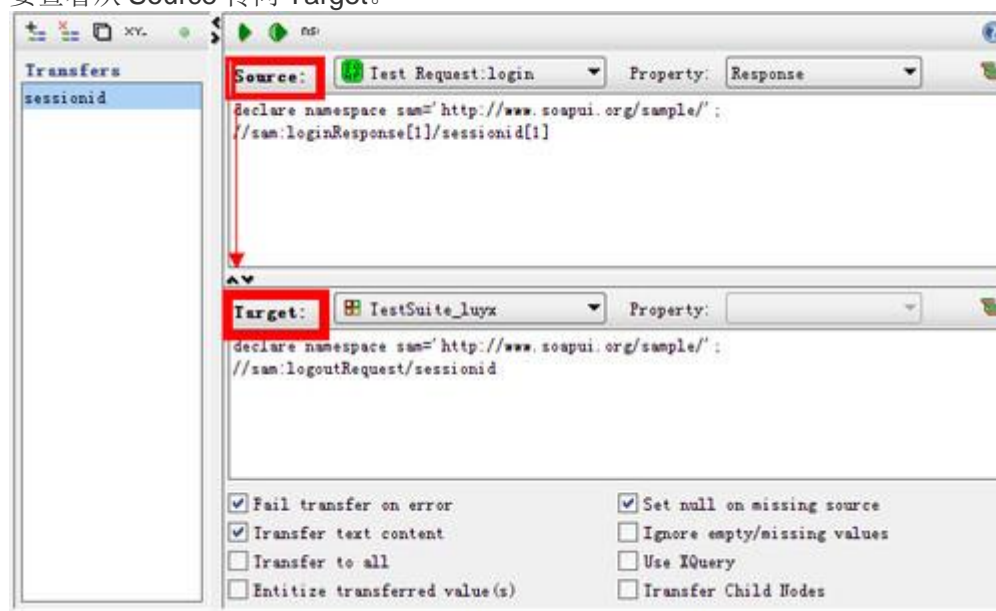
通过上面的学习，我们已经基本掌握了 SoapUI 测试接口的基本方法，对于更复杂的方法，比如某一个新请求需要用到前面一个请求返回的结果时，我们怎么获取呢？比如下一个请求需要登录获得的 SessionID，还记得添加 Properties 时，在“Add Step”中有一个“Property Transfer”（传值）功能吗？我们主要通过该功能实现。线面举一个简单登录的例子：

2.6 Transferring Property Values 传递属性值

软件安装时候，我们设置的 Tutorials location 的目录吗？笔者是在：E:\SoapUI-Tutorials，下面的项目以 SoapUI 自带的“sample-soapui-project”为例，介绍该用法。使用方法：新建项目一样，我们这次选择“sample-soapui-project”：



我们要进行 login -> logout 的操作，其中 login 产生了 sessionid，并且使用这个 sessionid 进行 logout。首先我们熟悉一下“Add Step”选择“Property transfer”后的主要页面，主要查看从 Source 传向 Target。

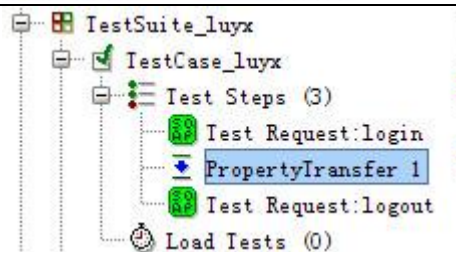


根据 source 到 target 的两种方式，我们进行两种说明。两种方式分别是：

- 1、直接将 source 获得的值传到下一个具体的步骤；
- 2、将 source 中获得的值存在一个变量中，以后直接引用，以便重复多次使用。

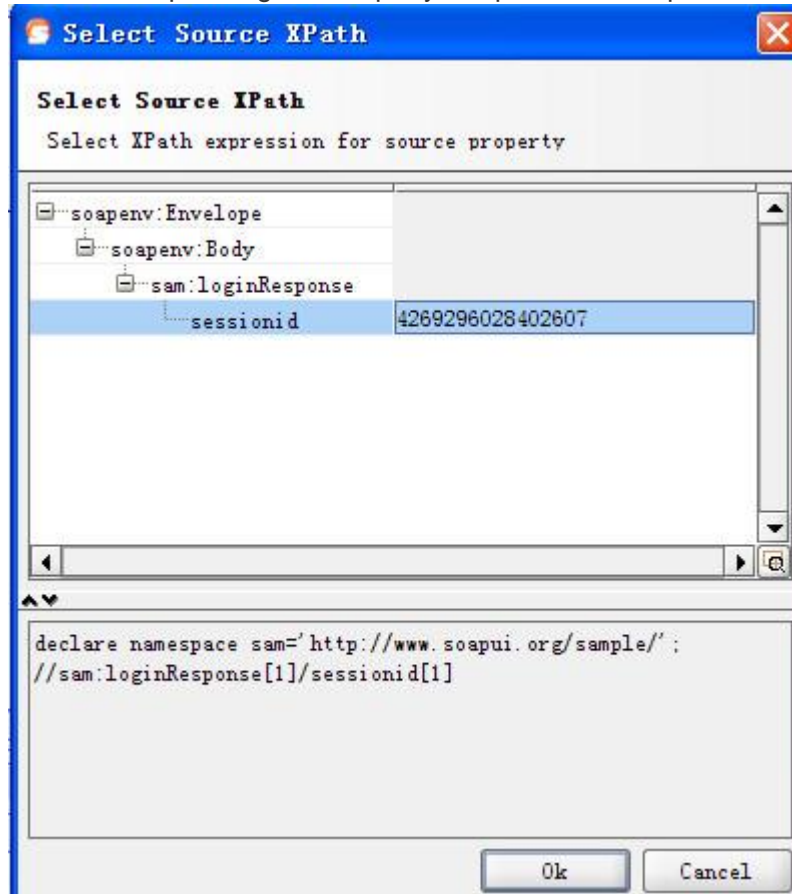
具体操作步骤如下：

【步骤一】：创建测试用例，添加 3 个步骤：login -> Property Transfer -> logout



【步骤二】：选择要获取的值：

双击 PropertyTransfer，打开编辑页面,添加 Property Tansfer，名称：SessionID
Source 中选择：Test Request:login Property:Response 点击 Xpath 图标设置 Xpath

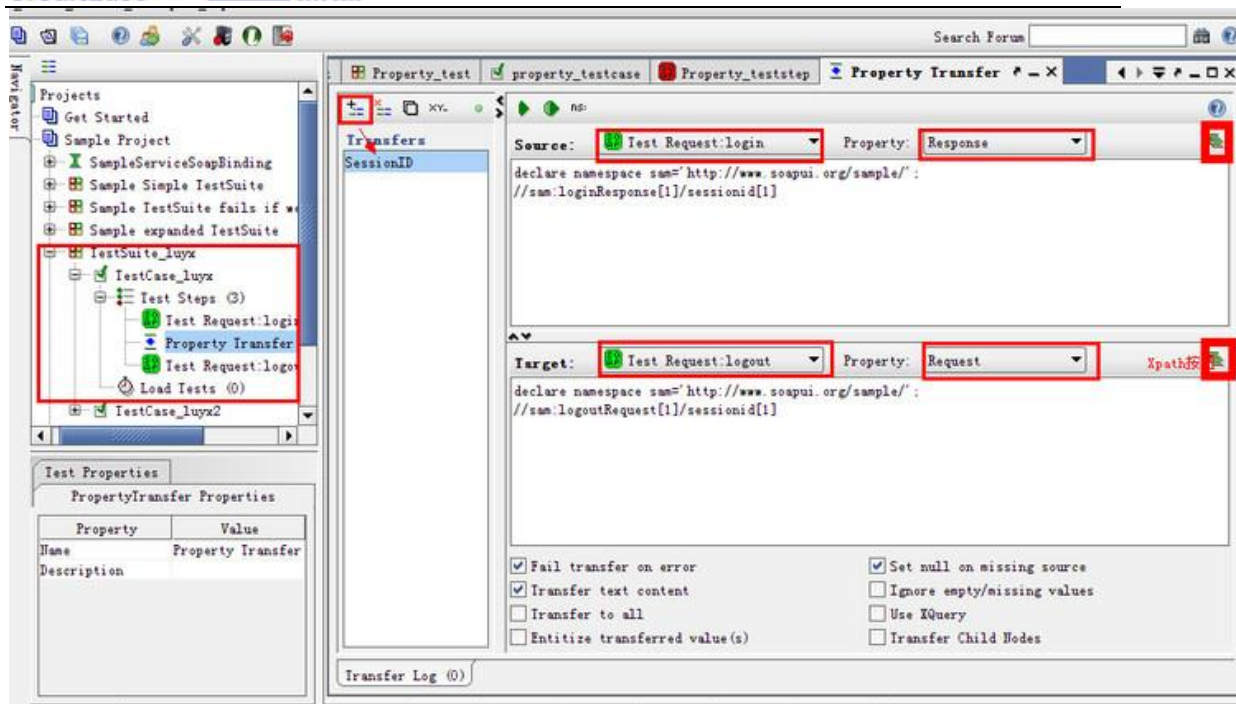


（在 sessionid 的数字上点击一下，就会出现下面的 Xpath）

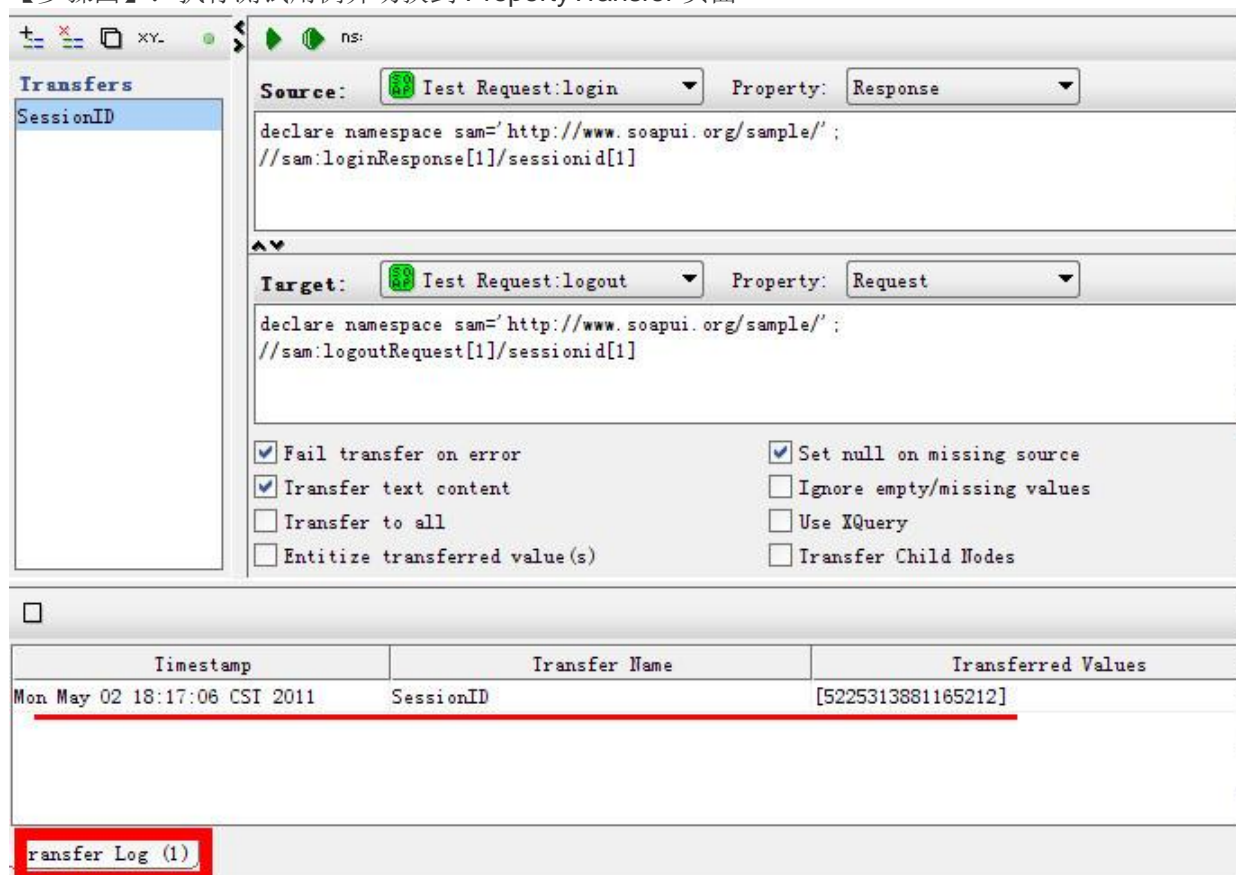
【步骤三】：设置 Target

Target 中选择：Test Request:logout Property:Request 点击 Xpath 图标设置 Xpath
设置好后的页面情况

此处应该注意的是 logout 中 session 处的值是什么，如果在设置完要替换的值后，logout 该处的值发生变化，PropertyTransfer 会失败，因为这里替换的是原来 logout 里面的值，新的值 PropertyTransfer 不认识。



【步骤四】：执行测试用例并切换到 PropertyTransfer 页面



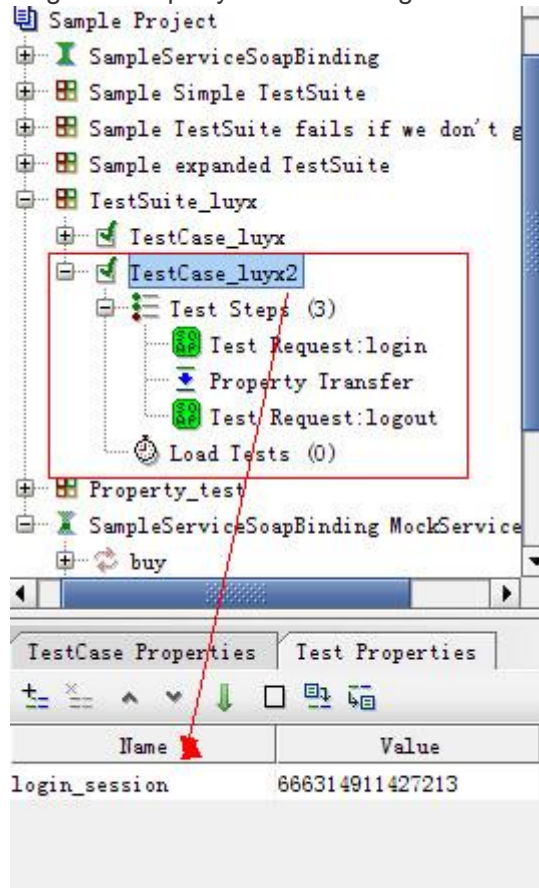
能看到 transfer log，从这里我们能看到传输的值对不对，如果不是 login 产生的 sessionid，那就说明不对，就要检查原因了。

2.7 变量的存储

在上面，我们已经学会如何在 Properties 中设置参数化属性，其实还有另外一种方式来存储变量，下面仍以 2.4 种的内容作为案例：

【步骤一】：创建用例

1、创建测试用例，login -> PropertyTransfer -> logout

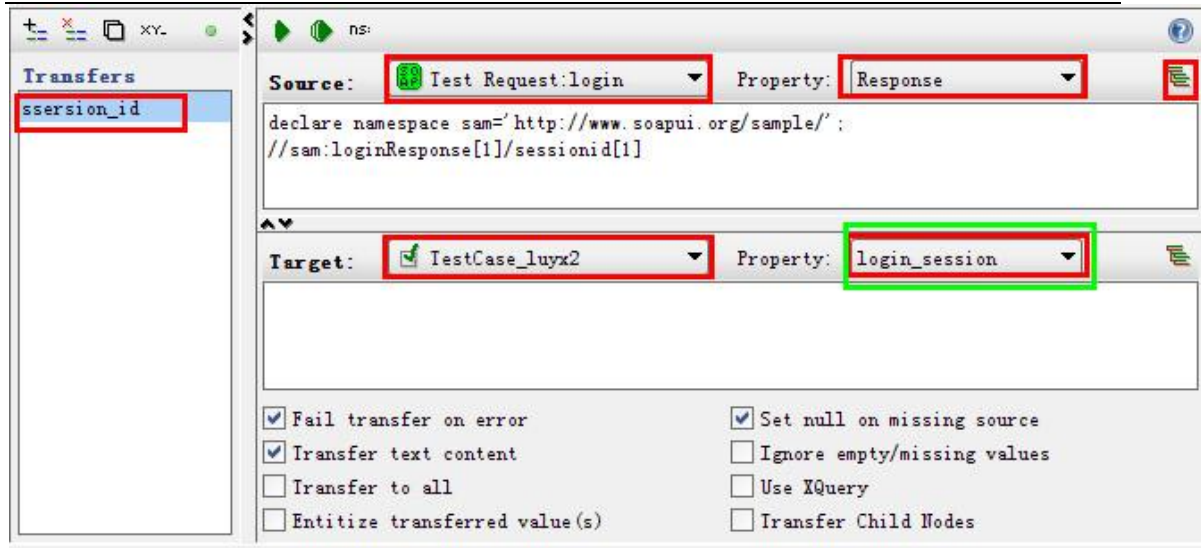


2. 不同的是，我们在 Testcase 的层次上新建一个属性 login_session,值随便输入一个即可。这个 login_session 就用来做变量存储从 login response 中获得的 session

【步骤二】：编辑 PorpertyTransfer

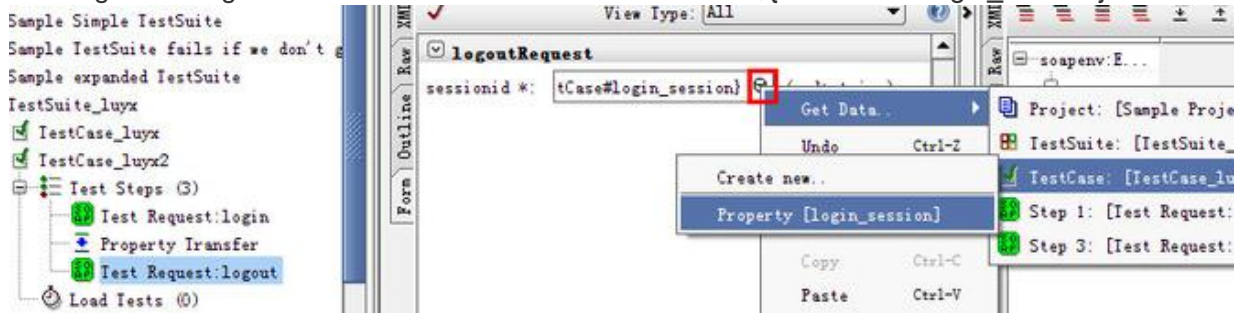
新建 session_id ， source: Test Request:login Property:Response, 选择 Xpth, 选中 Response 中的 sessionid

target: TestCase_luxx2 Property:login_session



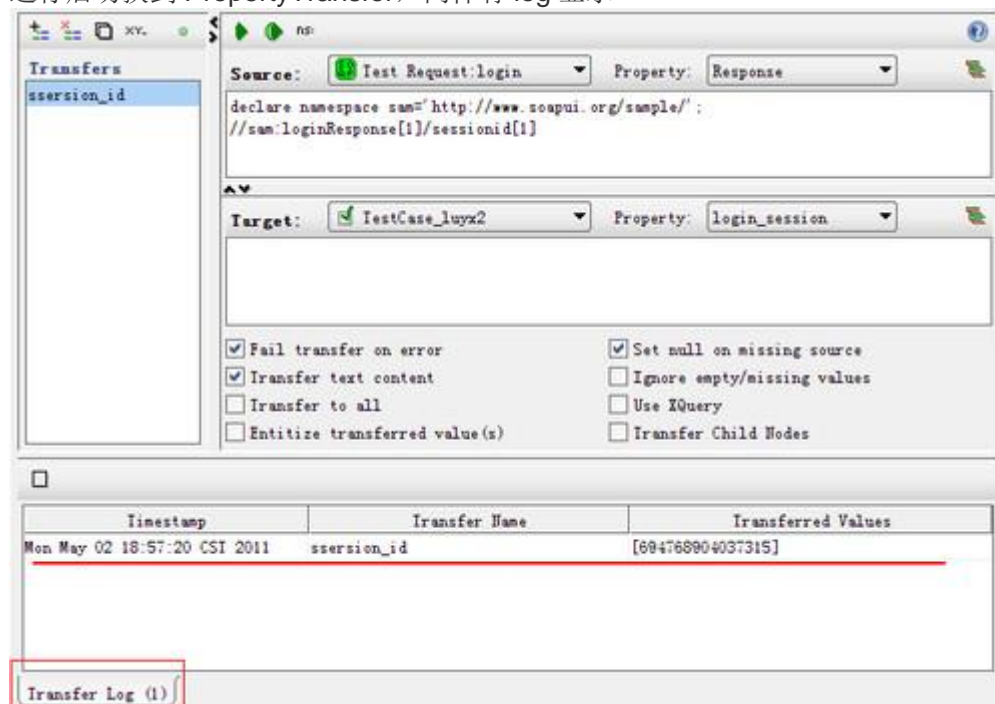
【步骤三】：选择参数化属性

打开 logout, 为 logout 的 session 选择参数, 选择后的结果是“\${#TestCase#login_session}”



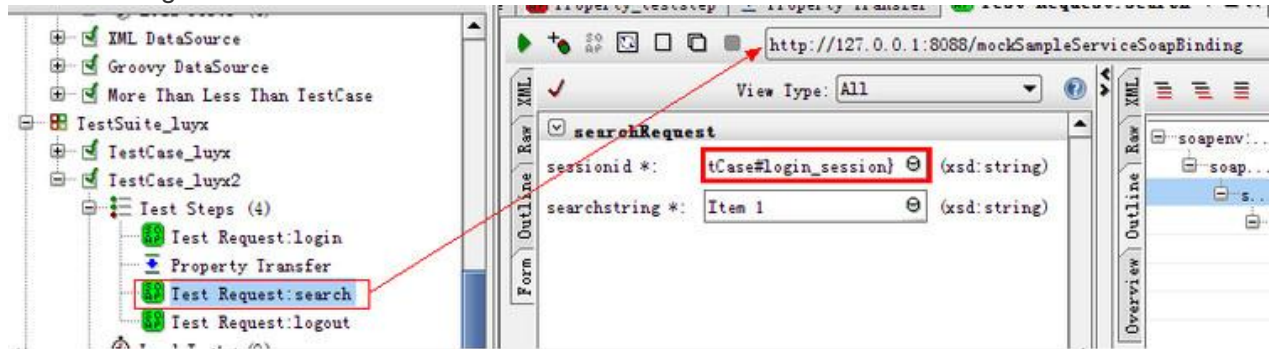
【步骤四】：执行并校验结果

运行后切换到 PropertyTransfer, 同样有 log 显示



这就完成了将 `source` 中获得的值存在一个变量中，在后面多次使用的实验了，虽然我们这样只用了一次而已，但也能看到用多次只是在相应要使用的地方参数化就行了。

我们快速地在原来的测试用例上添加一个步骤 `search`，参数化 `sessionid`，给 `searchstring` 一个固定的值，同样可以运行。



三、实现基于 Excel 表的自动化测试。

日常测试工作中，对于接口校验往往需要填写多组字符来验证，在 SoapUI 里如何实现 Loadrunner 中类似参数化的批量数据呢？在此我们需要借助外部 Excel 数据，目前 SoapboxUI 支持 XLS 格式的 Excel 文档，暂不支持 XLSX 格式的文档，在此请注意。通过此功能，我们可以实现让 SoapUI 自动从系统中调取对应批量数据，循环执行用例的自动化测试目的。

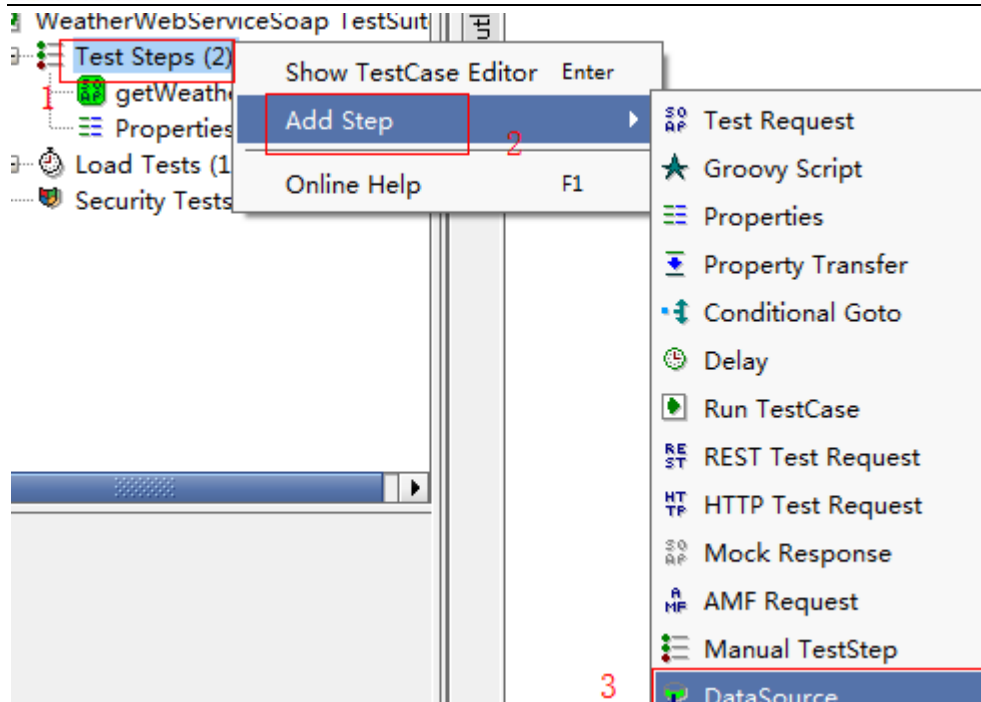
回到气象局天气查询的案例，因为查询条件只有一个，即：城市名称，因此我们需要提前准备一个 Excel 形式的文档：

【步骤一】：准备 xls 格式的文档

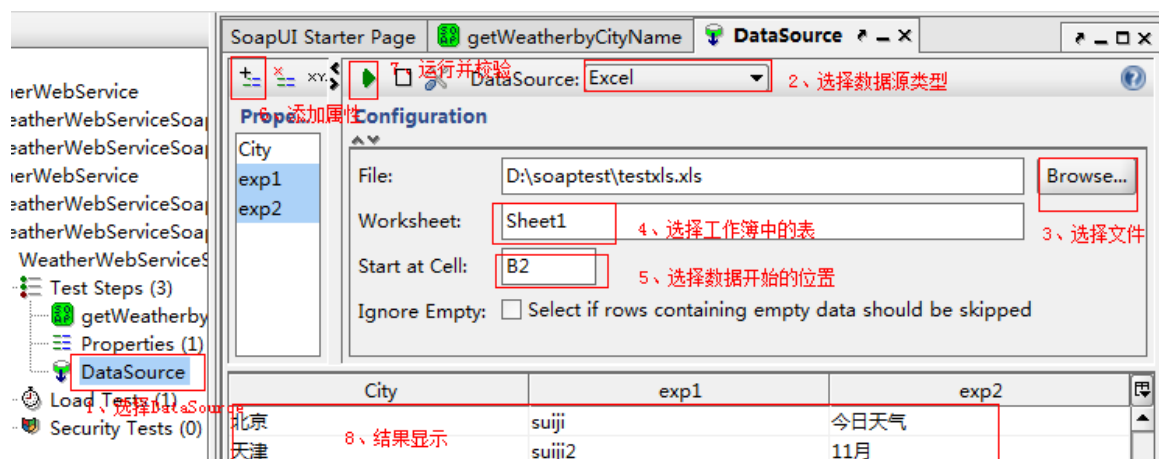
根据接口传参的数量，在 excel 中创建测试数据，参数名称不必一定要和接口一致，但是为了理解方便，最还要意义对应。下图为两条测试数据，其中 A 列为测试情景，方便测试人员看懂。

A	B	C	D
getWeatherbyCityName	city	test	exp
	北京	suiji	今日天气
	天津	suiji2	11月

【步骤二】：使用 DataSource 引用准备好的数据

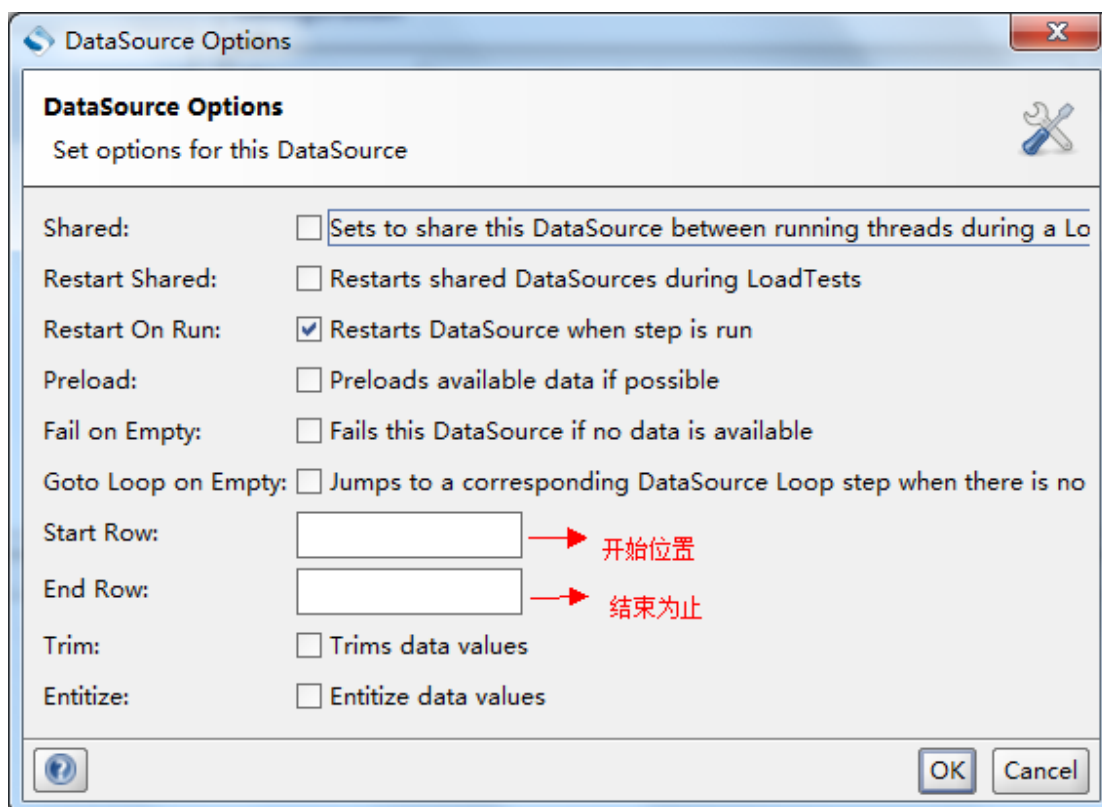


【步骤三】：选择本地 Excel 并进行设计



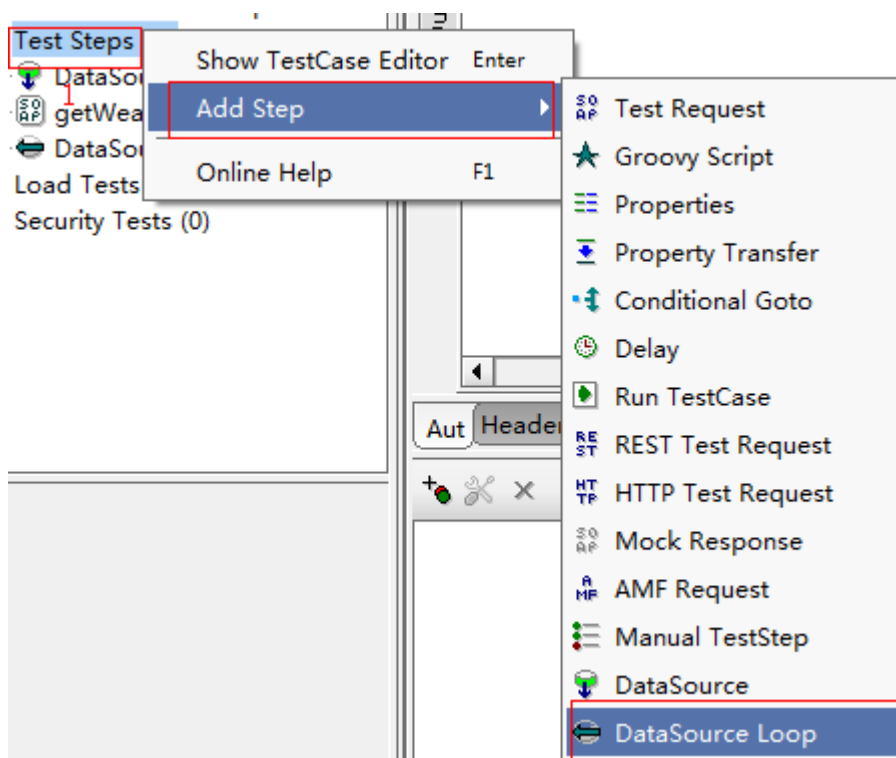
上图中，exp1 字段是为了说明属性名与 excel 表中只存在位置对应关系，而不存在必须名称一致的情况。exp2 为实际期望结果。其中第六步“添加属性”的实现内容为：properties 从本地 excel 读取数据，后续的其它方法调用 properties 就可以取到 excel 中的数据。另外，我们也可以实现对数据分段读取的控制，从而实现通过修改 excel 表就能修改用例的目的。

数据控制：

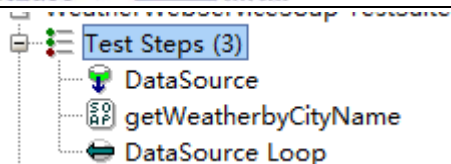


这样就可以取出 excel 中的测试数据了，但是要循环执行，还需要添加 datasource loop。

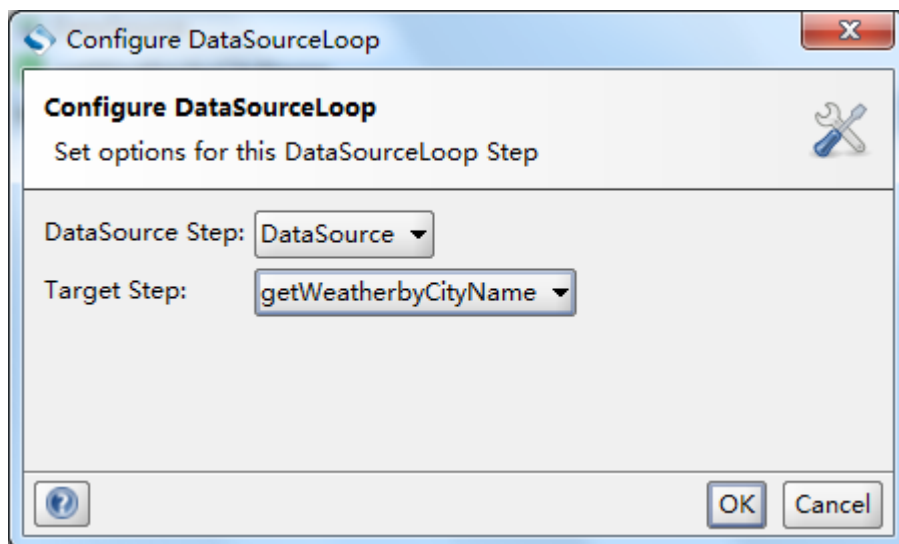
【步骤四】：添加 DataSource Loop



添加完成后的效果：（注意顺序）

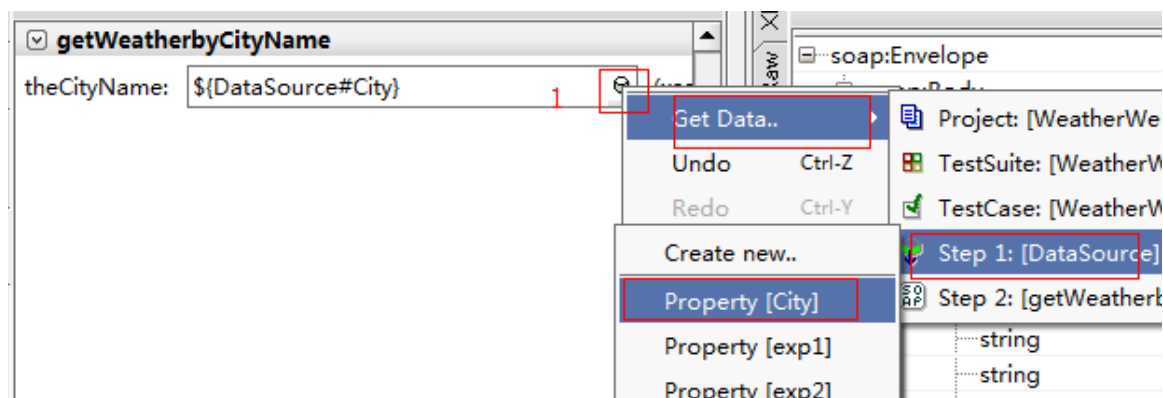


双击配置 DataSourceLoop，选择 Target Step 为 getWeatherbyCityName



（此处需要注意的是数据要先准备，然后执行测试，最后循环，所以顺序应该依次是 DataSource、TestStep、DataSource Loop。）

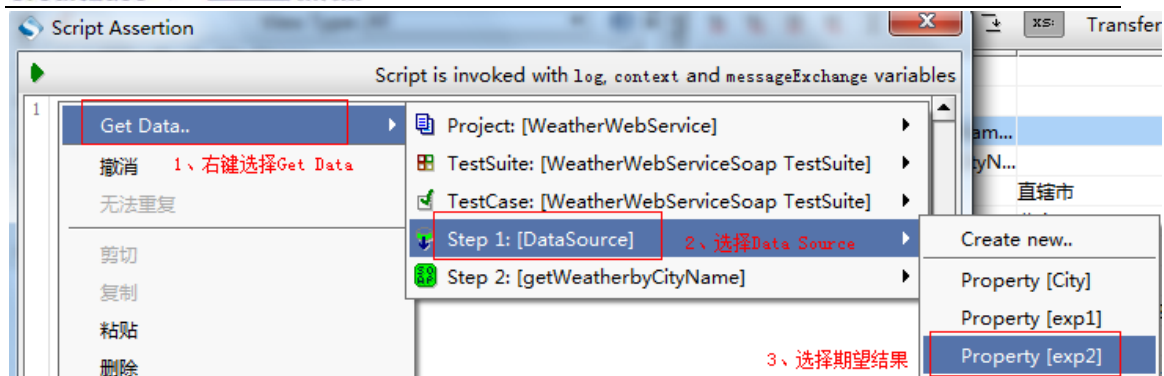
【步骤五】：设置测试用例数据的引用



测试的目的是判断实际结果与期望结果，这里要添加断言，普通 content 断言比较简单，这里不做介绍，这里介绍的是添加 script 断言

【步骤六】：从 Excel 中获取断言内容：

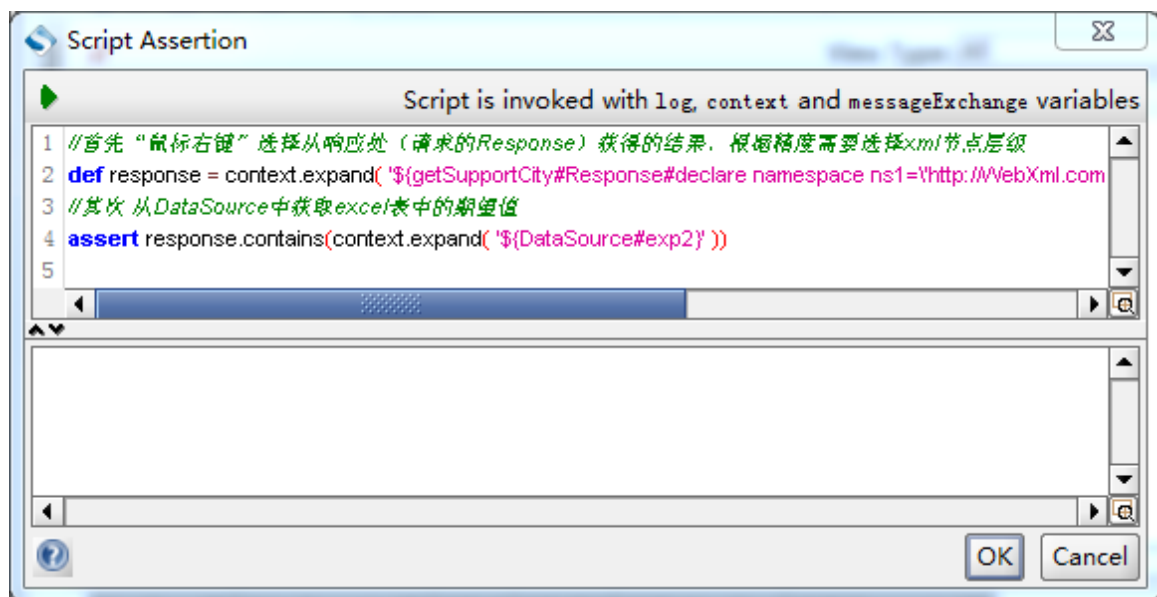
断言的添加方法同 2.4 中所示，只不过预期结果的期望值来源不同：



添加后并手工修改代码的结果：

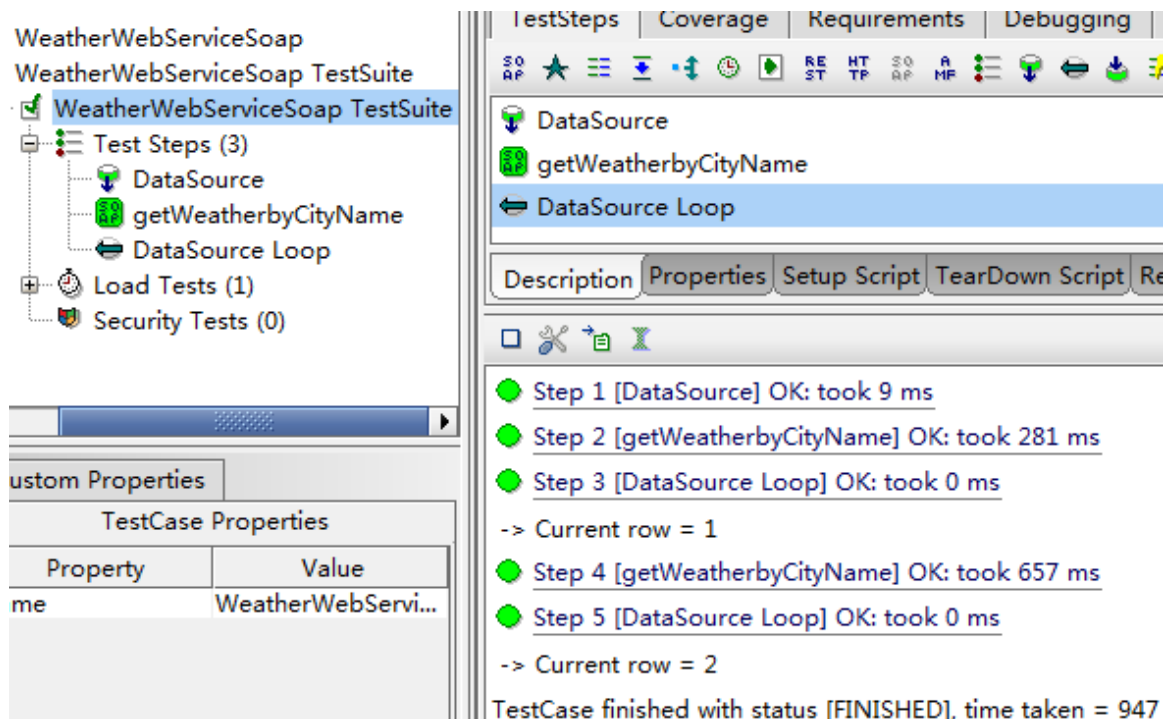
注意：该部分的判断采用字符串比较方法，语法如下：

```
def mess = 'Hello World' //定义一个 mess 变量，判断 mess 中是否包含 ' orl'
mess.contains('orl')    // Result: true
```



【步骤七】：执行测试用例（注意要选中测试用例，而不是测试步骤）

结果如下：



WeatherWebServiceSoap
WeatherWebServiceSoap TestSuite
WeatherWebServiceSoap TestSuite
Test Steps (3)
DataSource
getWeatherbyCityName
DataSource Loop
Load Tests (1)
Security Tests (0)

Custom Properties
TestCase Properties

Property	Value
me	WeatherWebServi...

TestSteps Coverage Requirements Debugging

DataSource
getWeatherbyCityName
DataSource Loop

Description Properties Setup Script TearDown Script Re

Step 1 [DataSource] OK: took 9 ms
Step 2 [getWeatherbyCityName] OK: took 281 ms
Step 3 [DataSource Loop] OK: took 0 ms
-> Current row = 1
Step 4 [getWeatherbyCityName] OK: took 657 ms
Step 5 [DataSource Loop] OK: took 0 ms
-> Current row = 2
TestCase finished with status [FINISHED], time taken = 947

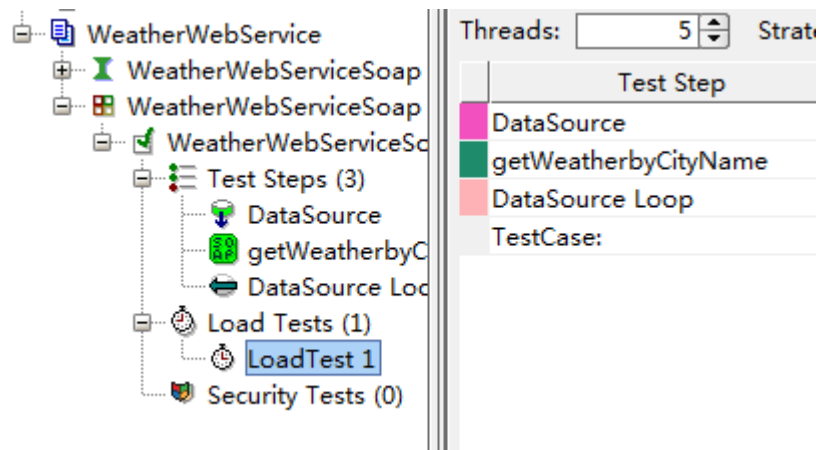
至此，自动化的初步测试工作就已经完成了。

四、性能测试

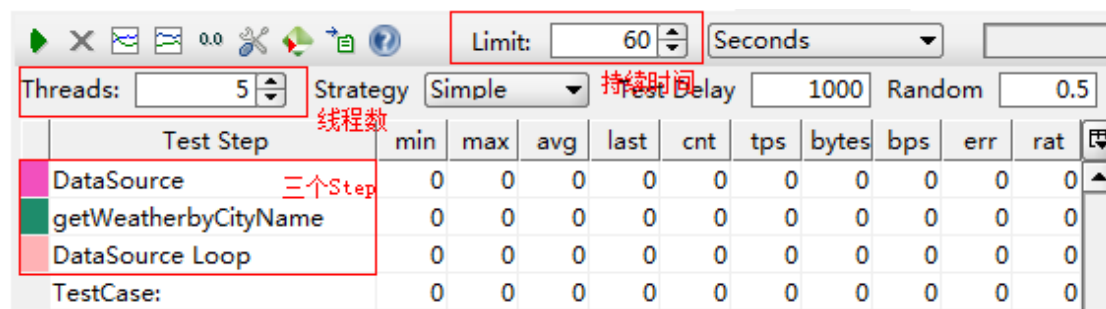
以往我们通过 Loadrunner 或是 jmeter 测试 Webservice，对于没有性能测试经验的同学可能会比较困难，SoapUI 把接口的性能测试工作的步骤变得非常简单。性能测试在 soapUI 中称为 Load Test，针对一个 soapUI 的 TestCase，可以建立一个或多个 LoadTest，这些 LoadTest 会自动的把 TestCase 中的所有步骤都添加到其中，在运行的时候，soapUI 会自动的使用多个线程来运行这些 TestStep，同时也会监控它们的运行时间，例如最短时间，最长时间，平均时间等等。这样用户能够很直观的看到 Webservice 的响应时间，从而对性能进行调优。

【步骤一】：激活 LoadTest

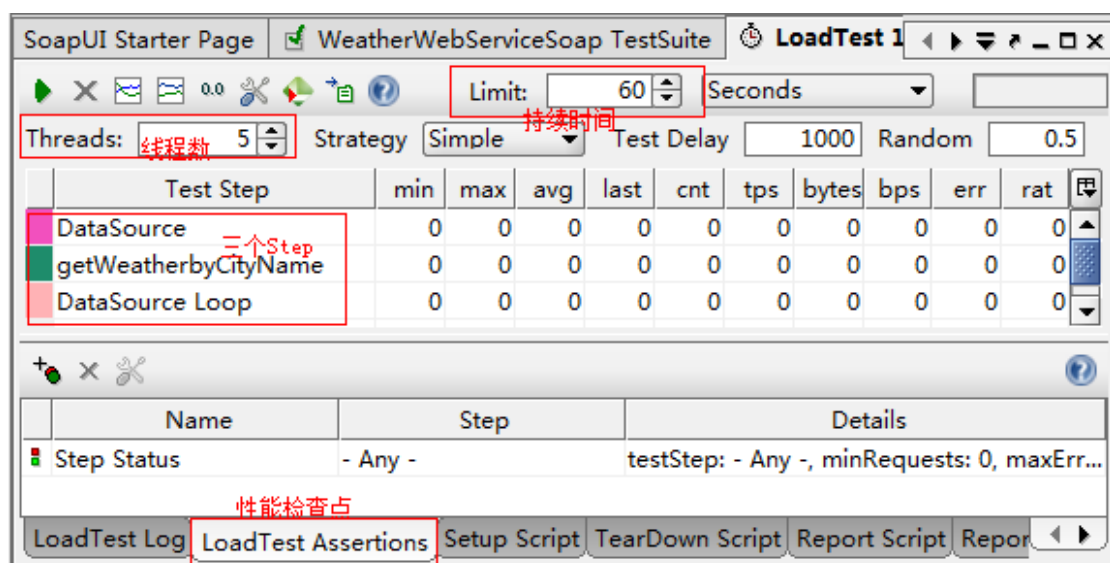
建立 LoadTest 非常简单，只需要在“Load Tests”上点击右键，选择“New LoadTest”，然后输入名称即可，下面是关于 getWeatherbyCityName 这一接口的性能测试，可以看到有三个 TestStep：



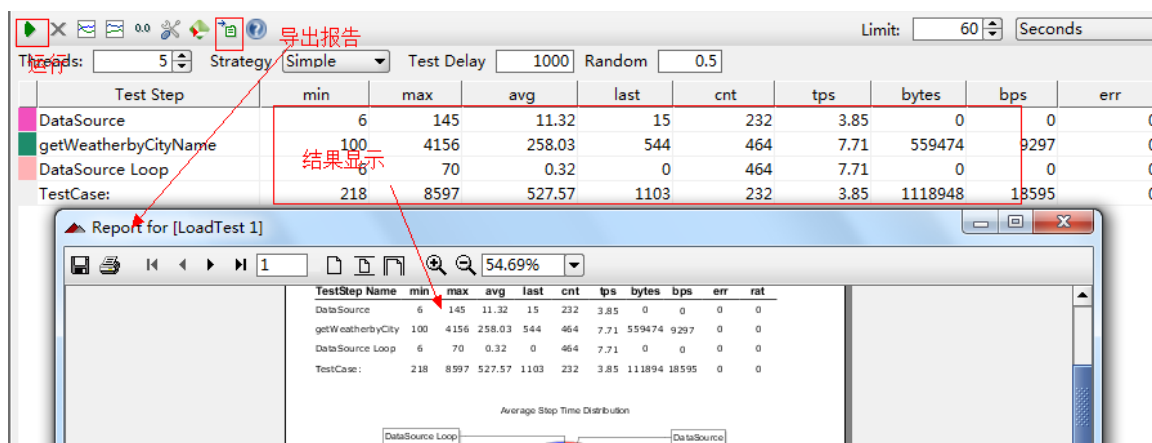
【步骤二】：设置 Threads（线程数）、Limit（限制时长）



性能测试还支持断言，用户可以对一个 TestStep 或 TestCase 设置运行时间要求，例如平均时间大于 2 秒就认为失败，点击下图中的“LoadTest Assertions”就可以设置。当然根据需要，用户也可以编写脚本来做一些准备工作，或者清除工作。参见下图的“Setup Script”和“TearDown Script”。



【步骤三】：收集测试结果，导出报告



3、 小结：

截至到此，SoapUI 的基本功能介绍完毕，当然此版本使用教程只是为了满足我公司内部的测试人员测试使用，其他更高级的功能，比如集成 SVN、程序的二次开发、集成构建工具等没有介绍。在后续版本中，视实际情况再次修改添加。