# SoapUI 使用说明

### zhangrong317@gmail.com

## 1 SoapUI 介绍

由于 Web 服务是被程序调用的, 一般不会提供界面让最终用户或测试人员直接使用,在 SoapUI 等工具出现之前,测试人员不得不自己编写程序来测试它, 这就要求测试人员花费很大的精力了解底层的接口,调用关系和详细的协议,导致他们不能把注意力集中到测试中。

SoapUI 的出现极大的改变了这一局面。作为一个开源的工具, SoapUI 强大的功能、易用的界面,吸引了很多用户。用户可以在 SoapUI 中通过简单的操作完成复杂的测试,不需要了解底层的细节,极大的减轻了工作量。SoapUI 支持多样的测试,例如功能测试,性能测试,回归测试等。到目前为止 SoapUI 的下载量已经超过了 100 万次,成为了 Web 服务测试标准和领先的 Web 服务测试 工具。它不仅仅可以测试基于 SOAP 的 Web 服务,也可以测试 REST 风格的 Web 服务, 后者也是本文介绍的重点。

SoapUI 基于 Java 开发,支持多个平台,安装非常简单。读者可以到 SoapUI 的 官方网站下载一个安装包 (本文使用的是 Window 版本 3.0.1),直接安装即可。在该安装包中,包括了一个 SoapUI 所需要的 JRE1.6 版本。安装完毕以后,读者需要设置 JAVA\_HOME 变量指向到相应的 JRE 目录,同时修改 PATH 变量,将 JRE1.6 的 bin 目录添加进去。

# 2 SoapUI 使用过程

### 2.1 创建/导入工程

- 安装并运行 SoapUI 之后,你就可以创建第一个 SoapUI 工程了。程序第 一次打开时,左侧导航面板上,自动有一个空的 Projects 工程。
- 右击左侧导航面板中的工作空间节点 "Projects", 选择 "New SoapUI Project"。

III P Project	ate	So Reques						
⊕ - € ci ⊕ - € 1	New soapVI Project <u>I</u> mport Project	Ctrl-M Ctrl-I						
±•∎ ₹	Impo <u>r</u> t Remote Project	Impo <u>r</u> t Remote Project						
	Save All Projects	Ctrl+Alt-S						
	Open All Closed Projects							
	Close All Open Projects							
	Rename	F2						
	New Workspace							
	Switch Workspace							
	Online Help	F1						

图表 2-1

■ 页面弹出 "New SoapUI Project" TAB 页,填入 Project Name, Initial WSDL/WADL 可填入 URL 地址或直接导入 WSDL 文件,导入文件后, 如下图所示:

🌔 New soapUI Pi	coject 🛛
<b>New soapUI Proje</b> Creates a new soap	ct . VI Project in this workspace
Project Name:	test1_file
Initial WSDL/WADL:	C:\Documents and Settings\Owne Browse
Create Requests:	Create sample requests for all operations?
Create TestSuite:	✔ Creates a TestSuite for the imported WSDL or WADL
Create MockService:	Creates a Web Service Simulation of the imported WSDL
Add REST Service:	Opens dialog to create REST Service
Relative Paths:	Stores all file paths in project relatively to project file (requires save)
	OK Cancel

图表 2-2

默认选上:

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

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

点击 OK 按钮后,页面弹出保存工程的提示,以 project 名称+"-soapui -project.xml"的形式进行命名,因此上述工程在保存时页面给出默认命 名为 test1\_file-soapui-project.xml,直接点击保存即可。

■ 保存成功后,页面继续弹出"Generate TestSuite" TAB 页:

👌 Generate Tes	stSuite	
<b>Generate TestSu</b> Generates TestSui	<b>ite</b> te with TestCase(s) for all Operations in this Interface	ζζζ
TestSuite:	<create> -</create>	
Style:	One TestCase for each Operation Single TestCase with one Request for each Operation	
Request Content:	<ul> <li>Use existing Requests in Interface</li> <li>Create new empty requests</li> </ul>	
Operations:	test10	<b>^</b>
	✓ test3	
	l test4	-
	Select all Unselect all	_
Generate LoadTest:	Generates a default LoadTest for each created TestCase	
	OK	Cancel

图表 2-3

选择:

Single TestCase with one Request for each Operation (说明:为每个接口的 请求都创建一个测试用例)

Create new empty requests(说明:创建一个空的请求)

Operations 中选择要测试的 WS 接口方法,如果一个 WS 有多个方法, Operations 中会列出所有方法,只须选择要测试的方法即可,上图,去 掉了 test10、test2 等接口的测试。

最后勾选上 Generates a default LoadTest for each created TestCase (说明: 为每个创建好的测试用例生成一个默认的负载测试)

选择完毕后,点击 OK 按钮, 进入测试用例命名页面,命名完毕后,确 定。



图表 2-4

在测试用例编写完毕后,可使用 ctrl+s 键,保存当前的工程。

■ 如果要导入其他人的工程,可通过选择"Import Project",找到 test- soapui -project.xml,选中后即可导入工程。

### 2.2 创建测试用例

■ 上面操作已经增加了 test1 的 Web 服务,接下来可以执行请求了。在上面增加接口的时候,已经根据 WSDL 的 Schema 定义为每一个操作创建了默认请求。

🖶 🖳 test1_file							
🖨 I. RequestServiceSoapBinding							
🖶 😂 notify							
🖶 🗳 Query							
🖶 🗳 QueryJDC							
🖶 👶 QueryJSY							
🕀 😂 QueryWZ							
🕀 😂 test1							
🖶 🗳 test10							
🖶 🤹 test2							
🖶 🤹 test3							
🖶 🤹 test4							
🕀 🛱 test6							
🖶 🧔 test7							
🖶 🧔 test8							
🖮 🗘 test9							
🖻 🗄 test1_file_TestSuite							
🖶 🗹 test1 TestCase							
💷 🗹 test4 TestCase							
💷 🗹 test6 TestCase							
🖅 🗹 test7 TestCase							
💷 🗹 test8 TestCase							
🗄 🗹 🗹 test9 TestCase							

图表 2-5

在 RequestServiceSoapBinding 节点下展开了 WS 服务中所有的方法,而 我们的测试包 test1\_file\_TestSuite 中根据"创建、导入工程"的第4步, 而仅创建了我们要测试的方法的测试用例。

■ 现在将以测试 test1 方法为例,来介绍用例的创建过程。按照下图所示, 打下测试包下的"test1 TestCase",在展开的"Test Steps"下选择"test1", 双击打开。



图表 2-6

双击"test1"后,在 SoapUI 的右侧会出现请求编辑器:



图表 2-7

请求编辑器分为三部分:

- 顶部的工具栏,包含一组请求相关的动作、操作
- 左边是请求区域
- 右边是响应区域

SoapUI 默认生成的请求中,"?"表示需要被替换的内容。根据需要,可以替换或者删除掉这些值。本接口需要一个名为 id 的入参,可在请求区域找到如下内容: <id xsi:type="soapenc:string"

xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">?</id>

"id"即为参数名,找到上面的"?",替换为 abcd 任意字符串。

通过按下工具栏最左边的按钮(绿色箭头)来发送本次请求,请求会在 后台执行,响应内容会出现在编辑器的右边,test1方法没有任何逻辑, 任意的入参均不会影响到输出结果,出参为一个一维数组,第一个值为 123,第二个值为456。 根据上述返回的结果报文后,可看到接口已被正确的调用,为在测试中 不用人为地进行接口功能是否正确的判断,因此加入断言 Assertions, 可由程序直接对返回结果进行判断。点击下图左上角的增加断言按钮:

	test1							
٠	+- 30 AP	0		۵		http://192.168.0.193:8080/QQFWstudy/servi	ces,	/Re
Raw XML	<soapen <soape <ws: <ss <id <th>vienv:H env:Bi test1 is xsi xsi:ty ttest1 env:E w:Env</th><th>elope ody&gt; soap type=" ody&gt; lody&gt; /elop</th><th>e xm r/&gt; e: e:</th><th>Instx: qtArr</th><th>si="http://www.w3.org/2001/XMLSchema-instand odingStyle="http://schemas.xmlsoap.org/soap/enc- of_soapenc_string" soapenc:arrayType="soap oct Assertion Select assertion to add WS-Security Status 确定 取消</th><th></th><th>Raw XML</th></id </ss </ws: </soape </soapen 	vienv:H env:Bi test1 is xsi xsi:ty ttest1 env:E w:Env	elope ody> soap type=" ody> lody> /elop	e xm r/> e: e:	Instx: qtArr	si="http://www.w3.org/2001/XMLSchema-instand odingStyle="http://schemas.xmlsoap.org/soap/enc- of_soapenc_string" soapenc:arrayType="soap oct Assertion Select assertion to add WS-Security Status 确定 取消		Raw XML

图表 2-8

会弹出"Select Assertion"对话框,通过下拉框选择"Contains"的断言,确定后弹出如下对话框,在 Content 中填入内容,此处是表示返回的结果报文里应该包含的字段,根据我们 test1 接口的返回值,填写如下,点击"OK",插入断言完毕,程序会在运行用例时,自动帮我们校验返回的结果报文是否包含"123"内容。

👌 Contains Asse	rtion	
<b>Contains Assertic</b> Specify options	n	X
Content:	123	
Ignore Case: Regular Expression:	Ignore case in comparison Use token as Regular Expression	
0		OK Cancel

图表 2-9

说明:

"Test Steps"中可创建多个测试用例,组成一个测试用例集,在运行该 test steps 时,会根据用例的顺序从上到下将用例进行一次测试,将上一 用例的输出作为下一用例的输入再组织相应的用例,此处待进一步研究。

### 2.3 创建负载测试

性能测试一般使用 loadrunner,或者自己写的调用客户端进行测试。 loadrunner 是全面的性能测试工具,对一般开发人员来说太重,并且需要 license。 自己写调用的客户端则测试的统计数据也需要写程序处理,比较麻烦。这里推荐 使用 SoapUI, SOAPUI 可以直接根据 WSDL 生成 SOAP 数据包,手工填入参数 后可以直接进行性能测试。

在创建完测试用例后,本工程的负载脚本也由在最初创建好工程时,已 经默认创建完毕,在此可直接打开使用,如下,可直接点开 Load Tests 节点,节点下包含名称为"LoadTest1"的负载脚本,双击打开。



图表 2-10

 双击打开后,页面如下显示,设置过程参考如下,场景为100用户并发, 持续运行10分钟,没有思考时间。相应的SoapUI可设置Threads=100, Test Delay=0, Limit=600,后面的下拉框选择Seconds,表示600秒。设 置完毕后,点击左上方的绿色箭头,程序开始进行负载测试。

🕲 LoadTest 1										막다	×
▶ 🗙 🖂 🖂 00 诸 💥 🕖						Limi	t: 60	Seconds	•		
Threads: 100 🜩 Strategy Simple	🔻 Test Delay	0 Random	0.5								
Test Step		max avg	last	ent		tps	bytes	bps	err	rat	<b>I</b>
test1	0	0	0	0	0	0	0	0	0	0	
TestCase:	0	0	0	0	0	0	0	0	0	0	
iX *		Show Types:	- All -					•	Show Steps:	- All -	•
time 🔺	ty	/pe		step				message			R.
10:38:22:953	Message					LoadTest	t started at	. Sat Jan 30 10	:38:22 CST 20	10	-
2010-01-30 10:39:41:187	Step Status		test1			TestStep	p [test1] re	sult status is	FAILED; [jav	a. net	
2010-01-30 10:47:24:453	Step Status		test1			TestStep	p [test1] re	sult status is	FAILED; [jav	a. net	
2010-01-30 10:48:35:953	Message					LoadTest	t ended at S	at Jan 30 10:4	8:35 CST 2010		
											-
0 entries											
LoadTest Log LoadTest Assertions Setup Script TearDown Script											

图表 2-11

负载测试过程中,右上方会有进度条显示测试的进度情况,SoapUI提供
 了2个图表和一个简要列表的形式列出了测试过程中相关数据的监控,

第9页共16页

如下图,下图为简要列表形式提供的数据:

🙆 LoadTest 1											f X
▶ 🗙 🖻 🕫 % 🔞 Limit: 600 🖨 Seconds 💌 100%											
Threads: 100 🔹 Strategy Simple	💌 Test D	elay 01	Random 0.5								
Test Step	min	max	avg	last	cnt	tps	bytes	bps	err	rat	Ę
testl	6	48378	291.45	21031	205856	339.04	156450560	257677	0		0 🔺
TestCase:	6	48378	291.45	21031	205856	339.04	156450560	257677	0		0

图表 2-12

点击上方红色方框框住的按钮,会弹出下方的监控图表,图中只有曲线, 没有任何数据说明,只能看到变化的情况,由于无相应的刻度,而无法 直观地看出数据大小:



图表 2-13

SoapUI 还提供了另一个图表,此图表与上与图表类似,不过仅能显示线

程数与另一统计内容的曲线变化情况,另一统计内容可通过下图红色方 框里的 "select statistic"进行选择,如下:

🙆 LoadTest 1		
🕨 🗙 🖂 🖂 👐 🍵 💥 🔞		
Threads: 100 Strategy Simple - Test Delay 0 Random 0.5		
Statistics History for [LoadTest 1]	d <sup>™</sup> Ø	×
Select Statistic: TPS 🗨 Resolution: data 💌	Ť8	0
		-
ThreadCount Total		
IN CONSTRUCT FORM           IN E           Show Types:		F

图表 2-14

# 3 与 LoadRunner 的比较

使用 LoadRunner 提供的 Webservice 协议进行相同接口的测试。

■ 不加校验的脚本(脚本名称: LR\_1)如下:

```
//@oolong 2/2/2010
Action()
{
    lr_start_transaction("here_start");
    web_service_call( "StepName=test1_101",
```

```
"SOAPMethod=RequestJaxRPCService.RequestJaxRPC.test1",
     "ResponseParam=response",
     "WSDL=C:/Documents and Settings/Owner/
                                                   桌 面
/RequestService.wsdl",
     "UseWSDLCopy=1",
     "Snapshot=t1264818214.inf",
     BEGIN_ARGUMENTS,
     "xml:sss=<sss><string></string></sss>",
     "id=aff",
     END_ARGUMENTS,
     BEGIN_RESULT,
     END_RESULT,
     LAST);
  lr_end_transaction("here_start", LR_AUTO);
  return 0;
}
```

■ 加了校验的脚本(脚本名称: LR\_2)如下,下面的脚本提供了对返回 结果的一个校验,类似 SoapUI 里提供的断言:

```
Action()
{
    char com[] = "123";
    lr_start_transaction("here_start");
    web_service_call( "StepName=test1_101",
    "SOAPMethod=RequestJaxRPCService.RequestJaxRPC.test1",
    "ResponseParam=response",
    "WSDL=C:/Documents and Settings/Owner/ 桌 面
/RequestService.wsdl",
    "UseWSDLCopy=1",
```

```
"Snapshot=t1264818214.inf",
     BEGIN_ARGUMENTS,
      "xml:sss=<sss><string></string></sss>",
     "id=aff",
     END_ARGUMENTS,
     BEGIN_RESULT,
      "test1Return[1]=Param_result",
     END_RESULT,
     LAST);
  if(strcmp(lr_eval_string("{Param_result}"),com)==0)
  {
     lr_end_transaction("here_start", LR_AUTO);
     lr_vuser_status_message("成功");
  }
  else
  {
     lr_end_transaction("here_start", LR_FAIL);
     lr_error_message(lr_eval_string("{Param_result}"));
  }
  return 0;
}
```

- 场景与 SoapUI 的场景一致: 100 用户并发,持续运行 10 分钟,没有思考时间。对 LR\_2 脚本进行性能测试后,发现响应时间比使用 SoapUI 进行测试的响应时间来的大,因此把校验过程注释掉,使用 LR\_1,又进行了一次负载测试。
- 从LR 可以得到的结果图表较多,以下列出几个示意图:

TPS 图如下:

#### Analysis Summary

Period: 30-01-2010 10:57:46 - 30-01-2010 11:08:04

 Scenario Name:
 D:\Program Files\Mercury\LoadRunner\scenario\test.lrs

 Results in Session:
 C:\Documents and Settings\Owner\Local Settings\Temp\res.lrr

 Duration:
 10 minutes and 18 seconds.

#### Statistics Summary

Maximum Running Vusers:	100	
Total Throughput (bytes):	109,483,911	
Average Throughput (bytes/second):	176,872	
Total Hits:	120,642	
Average Hits per Second:	194.898	View HTTP Responses Summary

#### Transaction Summary

Transactions: Total Passed: 241,484 Total Failed: 4 Total Stopped: 0 Average Response Time

Std. Transaction Name Minimum 90 Percent Average Maximum Pass Fail Stop Deviation Action Transaction 0.006 0.491 48.003 1.383 2.983 120,642 2 0 here start 0 0.006 0.491 48.003 1.383 2.983 120,642 2 vuser end Transaction 0 0 0 0 0 100 0 0 0 vuser init Transaction 0 0.047 0.582 0.095 0.122 100 0







平均事务响应时间如下:



图表 3-3

可以看到由 LR 得到的结果,图表丰富,数据完整,提供了更好、更直 观的说明作用。

■ 性能测试结果数据比较

脚本名称	平均响应时间	总事务数	TPS
SoapUI 脚本	291.45MS	205856	339.04
LR_1	0.491S	120646	194.898
LR_2	0.606S	96636	159.464

由上表及上面的分析得出以下结论:

- SoapUI 是专门针对 ws 接口的测试工具,在实现对相同接口测试时,SoapUI 表现出来的性能更优越。
- SoapUI 在发送请求时,是直接以组装好的 soap 报文进行发送, 而 LR 是使用 web\_service\_call 方法,从方法传入相应的参数, 再由 LR 组装为 soap 报文后,再发往接口进行调用,因此 LR 在 组装报文时,会有相应时间的耗费。LR 脚本中创建的事务,就 包含了这段组装报文的时间,因此响应时间会比 SoapUI 的响应 时间更大。LR 与 SoapUI 的差别应该还有更多,在此我尚未研究 的更深入。
- 对于 LR,在测试中若增加对返回结果的校验,也会耗费一定的时间,从上面的数据可以看出,时间差大约 0.12s 左右,这也与

第 15 页 共 16 页

校验中使用的方法有关系,如果方法高效的话,这个时间差也将 更少。

● SoapUI 提供的结果数据的分析不如 LR 那么详细与全面,但对于 接口级的测试已足够,且速度更优。

目前 WS 接口有多种语言可以实现,除了 JAVA、C++,当前还有遇到 WCF, 生成的 WSDL 文件无法直接读到接口的入参与出参,此种接口生成的 WSDL, LoadRunner 读取时直接失败,暂找不到解决方法。而使用 SoapUI,本人已测试 过,可支持 java、c++,且 wcf 这种形式的接口也可支持。