soapUI 学习文档

不是前言的前言

记得一个搞开发的同事突然跑来叫能不能做个 WebService 性能测试，当时我就凌乱了，不淡定啊，因为我是做测试的，以前连 WebService 是什么不知道，毕竟咱没开发背景，等等，在这里先鄙视一下自己。后来就去求助群里的朋友，他们介绍了 soapUI 可以做 WebService 性能测试。于是，就下载下来捣鼓了一翻，只是看着官方文档简单的配置了一下。

WebService 是什么东东，记得也是慢慢才理解的；这几天有空，所以再想学习一下 soapUI，51testing 上有很多关于 soapUI 的文章，大多都比较零散，再一下比较浅显，所以以看官方文档来学。后来，发现了“流口水的小猪”写得 soapUI 的总结，非常好。很兴奋地加了此人好友，此人比较低调，在此就不过多介绍了，嘻嘻！所以，看完这个系列的文章的之后，就有冲动想把它整理成一个文档，让大家更好的学习与交流。

不是声明的声明：

做这个系列文章的人不容易，整理这个文档的人（就是我啦~!）也不容易，所以，我也要来声明一下。

不得用将本文档用于任何商业用途，你要是拿这个换钱了，俺就用法律告你。

本文档共 11 节内容，除第 5 节由本人添加，其它全部由“流口水的小猪”原创。由编辑原因，文档所带截图不清晰，请访问原文博客阅读。

原文取于“流口水的小猪”163 博客：http://luyongxin88.blog.163.com/

虫师（博客园）

2011 年 8 月

第一节、WebService 基础

WebService

它是一种构建应用程序的普遍模型,可以在任何支持网络通信的操作系统中实施运行;

它是一种新的 web 应用程序分支，是自包含、自描述、模块 化的应用，可以发布、定位、通过 web 调用。

Web Service 是一个应用组件,它逻辑性的为其他应用程序提供数据与服务.各应用程序通过网络协议和规定的一些标准数据格式（Http，XML，Soap)来访问 Web Service,通过 Web Service 内部执行得到所需结果.

Web Service 可以执行从简单的请求到复杂商务处理的任何功能。一旦部署以后，其他 Web Service 应用程序可以发现并调用它部署的服务。

在构建和使用 Web Service 时,主要用到以下几个关键的技术和规则:

1.XML:描述数据的标准方法.

2.SOAP:表示信息交换的协议.

3.WSDL:Web 服务描述语言.

4.UDDI 通用描述、发现与集成，它是一种独立于平台的，基于 XML 语言的用于在互联网上描述商务的协议。

http://www.webxml.com.cn/zh_cn/index.aspx 这个网站中有不少免费的 WebService 可用

----------我个人觉得下面的这个理解起来更容易些，甚至我都觉得他有点像 API，只是放到 web 中了而已----------

1，什么是 Web Service ？

Web Service 就是一个网络组件（一个可以通过网络访问的程序）。

它有一个或多个端口（Port），这些端口用于接收客户端的请求，并返回响应

请求和响应的 都是一种基于 XML 的消息。 不过这种消息遵循特定的格式（SOAP ）。

2，怎样调用 Web Service?

可能这样说不太准确，应该是“怎样调用 Web Service 中定义的操作 ”

每个 Web Service 都有一个描述文件（WSDL ），

它描述 一个 Web Service 的如下方面：

（1）服务的端口（接收 SOAP 消息的端口）

（2）服务提供的操作

（3）操作的输入输出格式的定义（通过 XMLSchema 定义输入输出格式）

有了 Web Service 的描述文件（WSDL ），我们就知道怎样调用这个 Web Service 中定义的操作了。

（1）通过服务提供的操作找到你想调用的操作

（2）找到这个操作的输入格式的定义（XMLSchema ），按照这种输入格式构造一个 SOAP 消息

（3）将这个 SOAP 消息发送到服务的指定端口

（4）准备接收一个从 Web Service 服务器返回的 SOAP 响应吧 ！

3, Web Service 服务器

一个 Web Service 服务器，本质上和一个 Web 服务器是相同的。

它主要做下面这些事：

--> 监听网络端口（监听服务端口）

--> 接收客户端请求（接收 SOAP 请求）

--> 解析客户端请求（解析 SOAP 消息，将 SOAP 消息转换为数据对象）

--> 调用业务逻辑 （调用 Web Service 实现类的特定操作，参数是由 SOAP 消息转换而来的数据对象）

--> 生成响应 （将返回值转换为 SOAP 消息）

--> 返回响应 （返回 SOAP 响应）

==========

XML

什么是 XML?

- XML 指可扩展标记语言（EXtensible Markup Language）
- XML 是一种标记语言，很类似 HTML
- XML 的设计宗旨是传输数据，而非显示数据
- XML 标签没有被预定义。您需要自行定义标签。

- XML 被设计为具有自我描述性。
- XML 是 W3C 的推荐标准

XML 与 HTML 的主要差异

XML 不是 HTML 的替代。

XML 和 HTML 为不同的目的而设计：

XML 被设计为传输和存储数据，其焦点是数据的内容。

HTML 被设计用来显示数据，其焦点是数据的外观。

HTML 旨在显示信息，而 XML 旨在传输信息。

没有任何行为的 XML。XML 是不作为的。也许这有点难以理解，但是 XML 不会做任何事情。XML 被设计用来结构化、存储以及传输信息。

下面是 John 写给 George 的便签，存储为 XML：

```
<note>

<to>George</to>

<from>John</from>

<heading>Reminder</heading>

<body>Don't forget the meeting!</body>

</note>
```

这个标签有标题以及留言。它也包含了发送者和接受者的信息。但是，这个 XML 文档仍然没有做任何事情。它仅仅是包装在 XML 标签中的纯粹的信息。我们需要编写软件或者程序，才能传送、接收和显示出这个文档。

XML 仅仅是纯文本

XML 没什么特别的。它仅仅是纯文本而已。有能力处理纯文本的软件都可以处理 XML。

不过，能够读懂 XML 的应用程序可以有针对性地处理 XML 的标签。标签的功能性意义依赖于应用程序的特性。

通过 XML 您可以发明自己的标签

上例中的标签没有在任何 XML 标准中定义过（比如 <to> 和 <from>）。这些标签是由文档的创作者发明的。这是因为 XML 没有预定义的标签。在 HTML 中使用的标签（以及 HTML 的结构）是预定义的。HTML 文档只使用在 HTML 标准中定义过的标签（比如 <p> 、<h1> 等等）。XML 允许创作者定义自己的标签和自己的文档结构。

XML 不是对 HTML 的替代

XML 是对 HTML 的补充。XML 不是对 HTML 的替代，理解这一点很重要。在大多数 web 应用程序中，XML 用于传输数据，而 HTML 用于格式化并显示数据。

XML 应用于 web 开发的许多方面，常用于简化数据的存储和共享。

XML 把数据从 HTML 分离

如果你需要在 HTML 文档中显示动态数据，那么每当数据改变时将花费大量的时间来编辑 HTML。通过 XML，数据能够存储在独立的 XML 文件中。这样你就可以专注于使用 HTML 进行布局和显示，并确保修改底层数据不再需要对 HTML 进行任何的改变。通过使用几行 JavaScript，你就可以读取一个外部 XML 文件，然后更新 HTML 中的数据内容。

XML 简化数据共享

在真实的世界中，计算机系统和数据使用不兼容的格式来存储数据。XML 数据以纯文本格式进行存储，因此提供了一种独立于软件和硬件的数据存储方法。这让创建不同应用程序可以共享的数据变得更加容易。

XML 简化数据传输

通过 XML，可以在不兼容的系统之间轻松地交换数据。对开发人员来说，其中一项最费时的挑战一直是在因特网上的不兼容系统之间交换数据。由于可以通过各种不兼容的应用程序来读取数据，以 XML 交换数据降低了这种复杂性。

XML 简化平台的变更

升级到新的系统（硬件或软件平台），总是非常费时的。必须转换大量的数据，不兼容的数据经常会丢失。XML 数据以文本格式存储。这使得 XML 在不损失数据的情况下，更容易扩展或升级到新的操作系统、新应用程序或新的浏览器。

XML 使您的数据更有用

由于 XML 独立于硬件、软件以及应用程序，XML 使您的数据更可用，也更有用。不同的应用程序都能够访问您的数据，不仅仅在 HTML 页中，也可以从 XML 数据源中进行访问。通过 XML，您的数据可供各种阅读设备使用（手持的计算机、语音设备、新闻阅读器等），还可以供盲人或其他残障人士使用。

XML 用于创建新的 Internet 语言

很多新的 Internet 语言是通过 XML 创建的：

其中的例子包括：

- XHTML - 最新的 HTML 版本
- WSDL - 用于描述可用的 web service
- WAP 和 WML - 用于手持设备的标记语言
- RSS - 用于 RSS feed 的语言
- RDF 和 OWL - 用于描述资源和本体
- SMIL - 用于描述针针对 web 的多媒体

假如开发人员都是理性的

假如他们都是理性的，就让未来的应用程序使用 XML 来交换数据吧。

未来也许会出现某种字处理软件、电子表格程序以及数据库，它们可以使用纯 XML

==========

WSDL

什么是 WSDL?

- WSDL 指网络服务描述语言

- WSDL 使用 XML 编写
- WSDL 是一种 XML 文档
- WSDL 用于描述网络服务
- WSDL 也可用于定位网络服务
- WSDL 还不是 W3C 标准

WSDL 可描述网络服务（Web Services）

WSDL 指网络服务描述语言 (Web Services Description Language)。

WSDL 是一种使用 XML 编写的文档。这种文档可描述某个 Web service。它可规定服务的位置，以及此服务提供的操作（或方法）。

WSDL 文档仅仅是一个简单的 XML 文档。

它包含一系列描述某个 web service 的定义。

WSDL 文档结构

WSDL 文档是利用这些主要的元素来描述某个 web service 的：

元素	定义
<portType>	web service 执行的操作
<message>	web service 使用的消息
<types>	web service 使用的数据类型
<binding>	web service 使用的通信协议

一个 WSDL 文档的主要结构是类似这样的：

```
<definitions>
<types>
  definition of types.....
</types>

<message>
  definition of a message....
</message>

<portType>
  definition of a port.......
</portType>

<binding>
  definition of a binding....
</binding>
```

```
</definitions>
```

PS：实际上以上的结构，可以用浏览器打开一个具体的 webservice 来看，比如以下查询手机归属地的 wsdl 格式的 webservice

http://fy.webxml.com.cn/webservices/EnglishChinese.asmx?wsdl

WSDL 文档可包含其它的元素，比如 extension 元素，以及一个 service 元素，此元素可把若干个 web services 的定义组合在一个单一的 WSDL 文档中。

WSDL 端口

<portType> 元素是最重要的 WSDL 元素。 它可描述一个 web service、可被执行的操作，以及相关的消息。 可以把 <portType> 元素比作传统编程语言中的一个函数库（或一个模块、或一个类）。

WSDL 消息

<message> 元素定义一个操作的数据元素。 每个消息均由一个或多个部件组成。可以把这些部件比作传统编程语言中一个函数调用的参数。

WSDL types

<types> 元素定义 web service 使用的数据类型。 为了最大程度的平台中立性，WSDL 使用 XML Schema 语法来定义数据类型。

WSDL Bindings

<binding> 元素为每个端口定义消息格式和协议细节。

WSDL 实例

这是某个 WSDL 文档的简化的片段：

```
 <message name="getTermRequest">
   <part name="term" type="xs:string"/>
</message>

<message name="getTermResponse">
   <part name="value" type="xs:string"/>
</message>
```

```
<portType name="glossaryTerms">
  <operation name="getTerm">
        <input message="getTermRequest"/>
        <output message="getTermResponse"/>
  </operation>
</portType>
```

在这个例子中，<portType> 元素把 "glossaryTerms" 定义为某个端口的名称，把 "getTerm" 定义为某个操作的名称。

操作 "getTerm" 拥有一个名为 "getTermRequest" 的输入消息，以及一个名为 "getTermResponse" 的输出消息。

<message> 元素可定义每个消息的部件，以及相关联的数据类型。

对比传统的编程，glossaryTerms 是一个函数库，而 "getTerm" 是带有输入参数 "getTermRequest" 和返回参数 getTermResponse 的一个函数。

第二节、通过 axis2 创建 WebService 实例。

通过这一节的学习，更真切的感觉到了什么是 webservice，我们怎样自己写一个简单的 webservice，这样，对我们后面利用 soapUI 测试 webservice 有帮助。如果你都不知道你测试的东西是什么，就直接测试，那似乎有点盲目。

- 开发 webservice 工作准备
- 开发简单的 webservice
- 利用 tomcat+axis2 发布 webservice
- 调用（测试）webservice

==========开发 WebService 的环境准备==========

使用 axis2

---

axis2 是新一代的 web service 开发工具，它会让你的 web service 开发变得轻松，快捷。下面让我们以一个实际的例子来体验一下。

首先，工欲善其事，必先利其器。就让我们先做好一些必备的准备工作吧。

1.下载 axis2 的 2 进制的包和 war,现在的最新版本是 1.4.1 发布时间是 2008-8-25

地址分别是：http://apache.mirror.phpchina.com/ws/axis2/1_4_1/axis2-1.4.1-bin.zip

http://apache.mirror.phpchina.com/ws/axis2/1_4_1/axis2-1.4.1-war.zip

2.把下载后的 war 放入 tomcat 的 webapps 目录里，然后启动 tomcat，这样 war 包就会自动解压为目录 axis2

在浏览器中输入http://localhost:7890/axis2/ ,如果一切正常你会看到下面的画面 （这个东西，我以前就弄过，所以基本很快就搞定了）

3. 就开始准备一下 axis2 的 eclispe 的插件了。axis2 的 eclispe 插件分为 2 个，一个是帮助我们生成 aar 文件的，另一个是帮我们用 wsdl 文件生成 stub 代码的。

下载地址是

http://www.apache.org/dyn/mirrors/mirrors.cgi/ws/axis2/tools/1_4_1/axis2-eclipse-service-archiver-wizard.zip

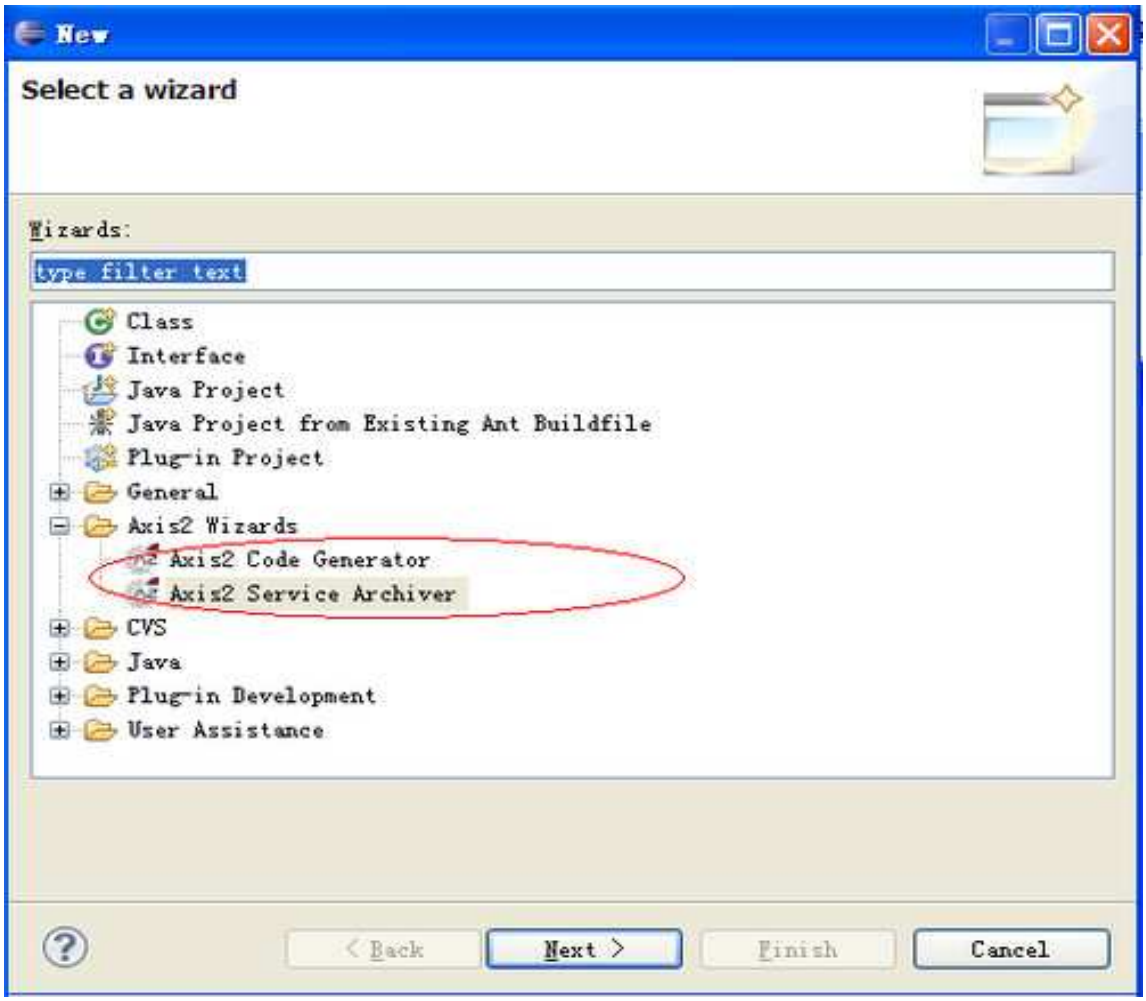
http://www.apache.org/dyn/mirrors/mirrors.cgi/ws/axis2/tools/1_4_1/axis2-eclipse

-codegen-wizard.zip

下 载完 2 个压缩文件后，可以直接把解压后的文件拷贝到 plugins 目录中，也可以在 links 目录中写文件路径的方式来安装插件，安装完插件后，打开 eclipse，在 package explorer 中点击右键--->选择 new---->other 如果安装正确你会看到

（这个地方只有有点要说明：就是加压后的文件不能直接放，注意到解压后目录是 \axis2-eclipse-service-archiver-wizard\Axis2_Service_Archiver_1.3.0,

我们要放的是 Axis2_Service_Archiver_1.3.0 这个文件夹，而不是上一层文件夹，刚开始我直接解决，拖过去的，原来拖的是 axis2-eclipse-service-archiver-wizard，重启 eclipse 怎么也找不到插件，找了很久的原因才发现）

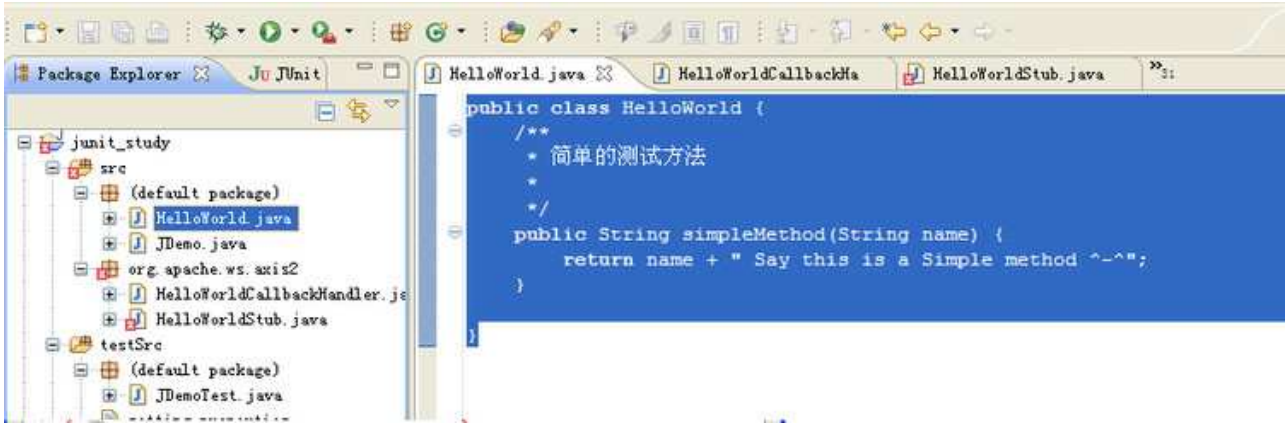


这样我们的准备工作就好了哦。

==========开发一个简单的 WebService==========

开发、发布自己的 webservice

1. 首先自己写一个 java 类



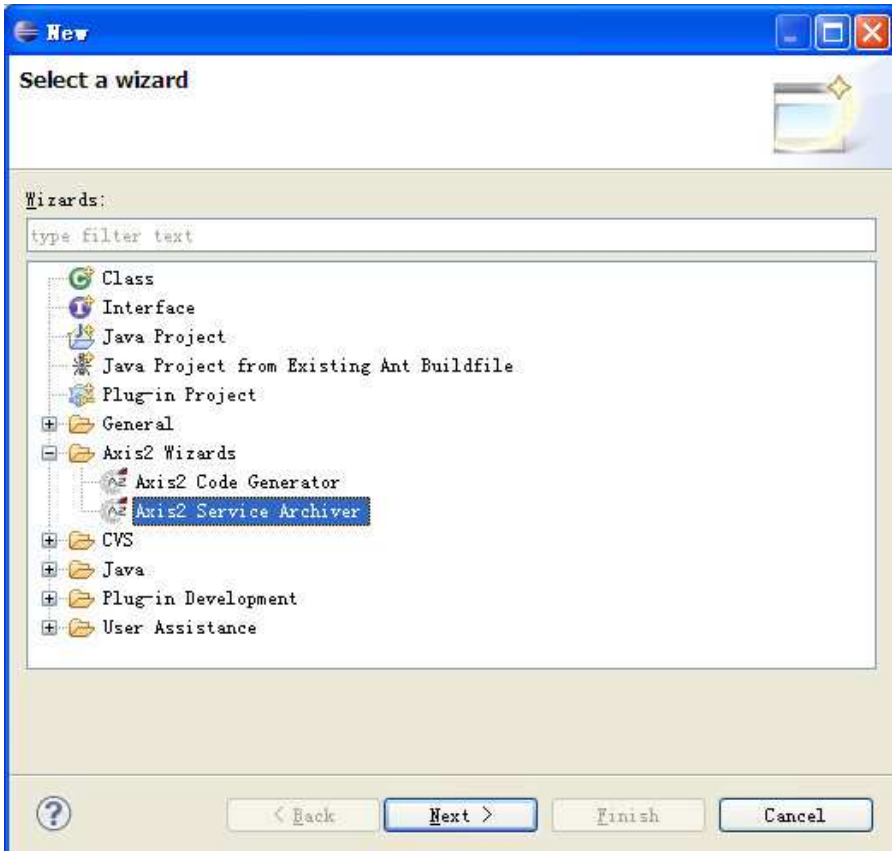
代码如下：

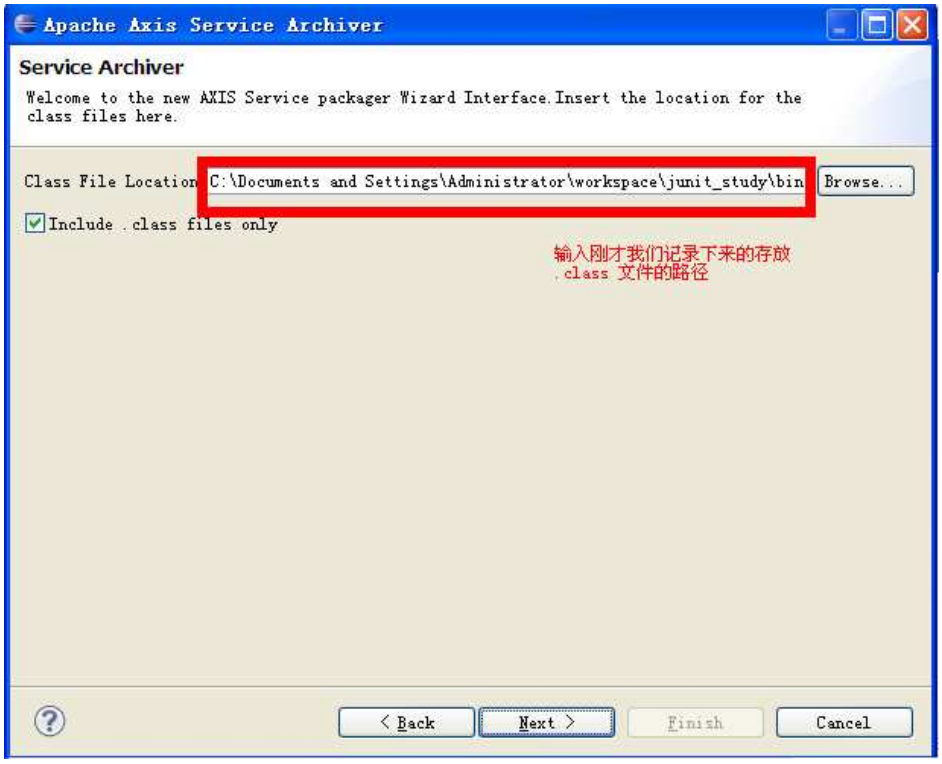
```
public class HelloWorld {
/**
 * 简单的测试方法
 *
 */
public String simpleMethod(String name) {
 return name + " Say this is a Simple method ^-^";
}
}
```

这里特别注意，刚开始，编辑好后要保存，eclipse 会自动的编辑成.class 文件，需要把存放.class 文件的目录记住，我这里是

C:\Documents and Settings\Administrator\workspace\junit_study\ 这个后面发布的时候用的到。（原文中没有详细说明这个）

在 eclispe 的 package Explorer 中点击右键，在菜单中选择新建--->other...----->Axis2 Service Archiver



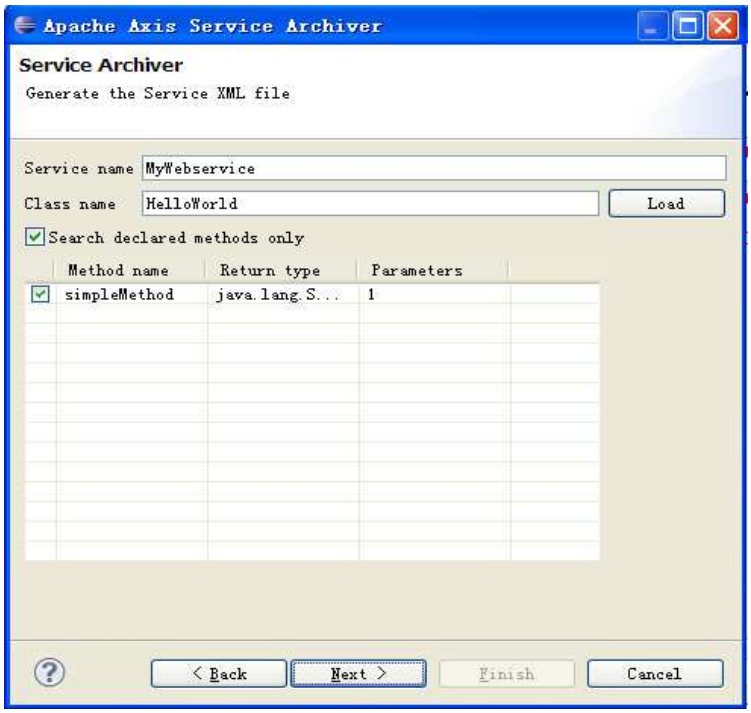


点击 next 之后进入了选择 wsdl 文件，这里我们选择 skip wsdl

点击 next 之后，进入的是选择 jar 文件的页面，这里我们没有外部的 jar，所以点击 next 直接跳过这个页面。

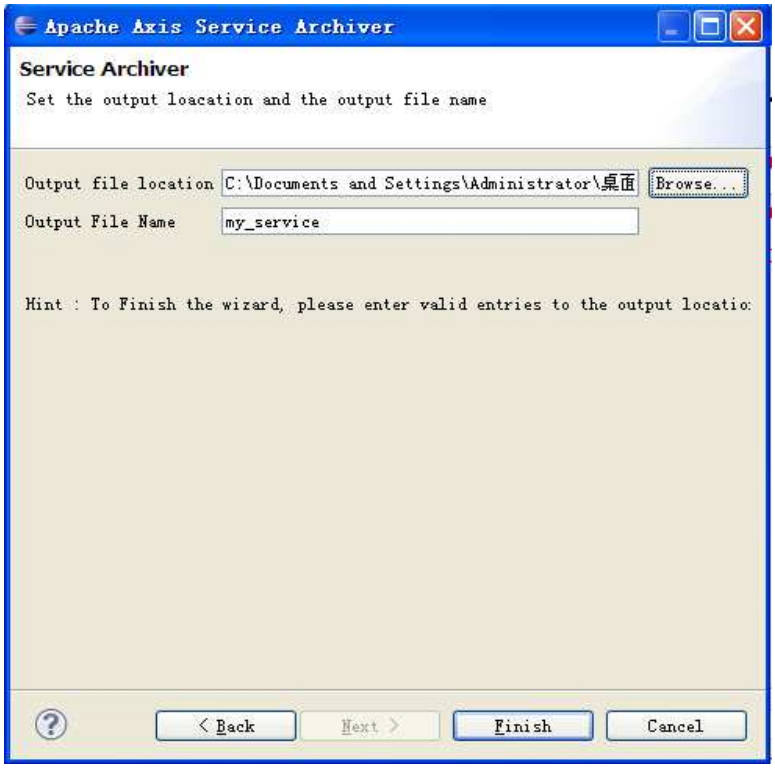
点击 next 之后，进入的是选择 xml 页面，这里我们选择的是自动生成 xml，也就是勾选 Generate the service xml automatically 这一项

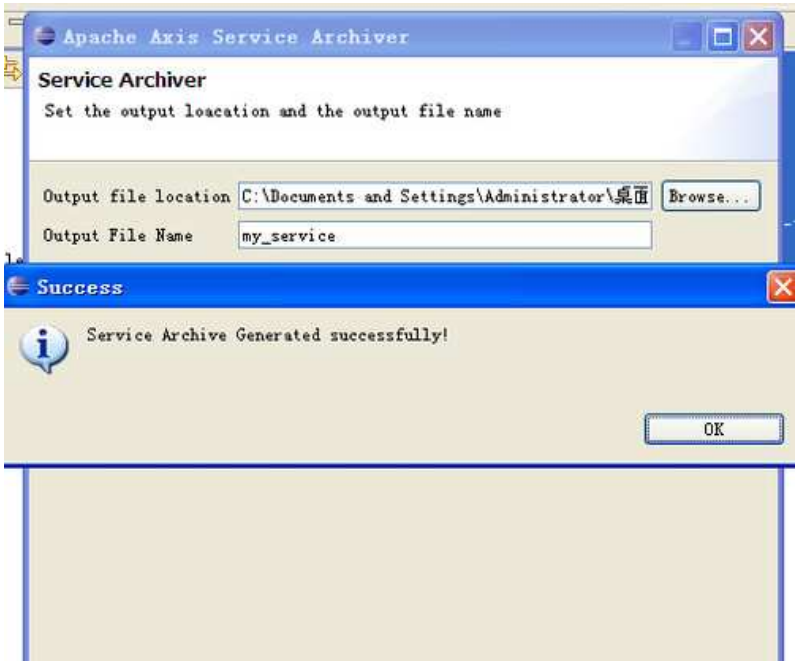
5.　点击 next 之后，进入的是生成 xml 文件的页面，在 service name 里填写这个服务所起的名字，这里我起名为 MyWebserver,然后在 class name 中
填写要发布的类，这里一定要写全路径，写好后就可以点击 load 按钮，
如果一切 ok 的话，你会看到如下画面



点击 next 后，进入的是输出 artiver 文件的页面，先要在 output File location 中选择要输出的路径，

在 output File Name 中输入 artiver 文件的名称。我起的名字是 my_service （这里输入的 filename 只是一个文件名而已，无其他什么意义）





==========发布 WebService==========

这样，在我的桌面上就生成了一个 my_service.aar 文件，将其放入到 \Tomcat6.0\webapps\axis2\WEB-INF\services 中，

打开 http://localhost:7890/axis2/services/listServices

Available services

Version

Service EPR : http://localhost:7890/axis2/services/Version

Service Description : Version

Service Status : Active
Available Operations

- getVersion

HelloWorld

Service EPR : http://localhost:7890/axis2/services/HelloWorld

Service Description : HelloWorld

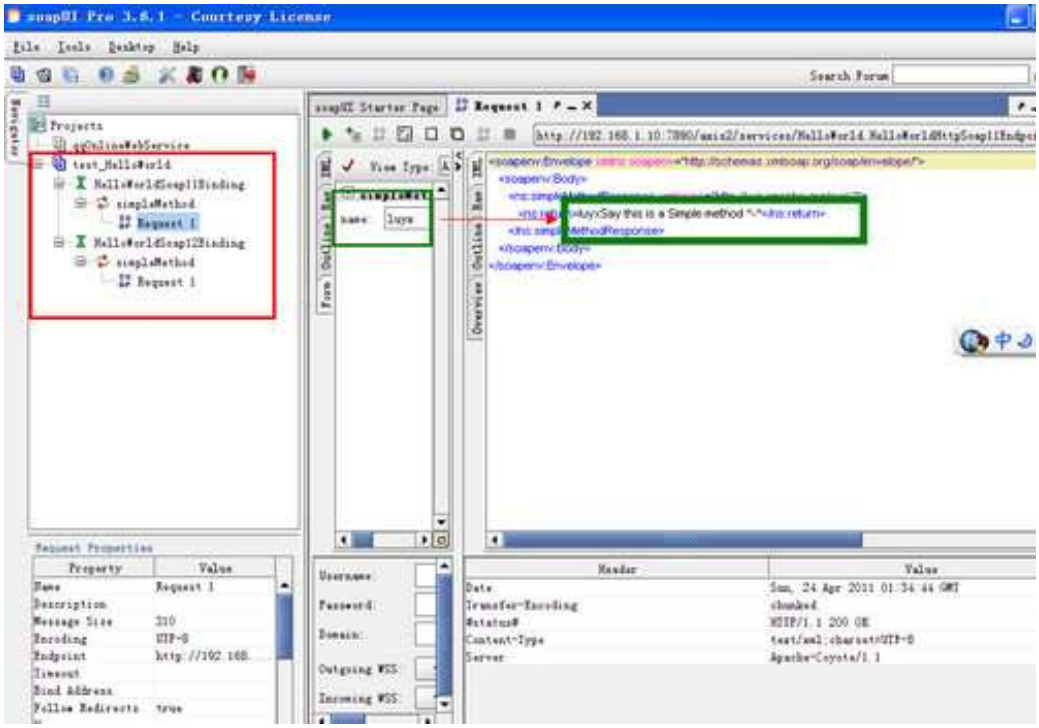
Service Status : Active
Available Operations

- simpleMethod

点击上面的 HelloWorld 连接，就打开了如下的页面

原来如此，这下我就知道 soapUI 可以上场了。

==========测试 WebService==========



PS：原文中的三其实就是说的用 java 来调用我们刚才发布的 webservice，我只是用了 soapUI 来调用了而已。

第三节、SoapUI 基础知识

已经学习总结了两次的 soapui 的学习，但还没真正进入主题的学习，都是在打围。现在要正式来看看 soapUI 了。

==========有关 SoapUI==========

首先，有关 soapUI 的官方网站似乎有两个

http://www.soapui.org/ & http://www.eviware.com/

我经常都被搞糊涂，两个上面都可以下载 soapui，不过只有前者有 soapui 的教程。

据我所知，soapui 有 windows 和 linux 版本。我学习的主要是 windows 版本。且虽然能够在网站上申请 trial license，但可能还是不及 soapui pro 的功能全。并且很显然，还有 loadUI 就是用来测试 webservice 的性能的吧。

有关 soapUI / soapUI pro /loadUI 的比较，请参考 http://www.soapui.org/ 网站的 About soapUI？Product Comparison 。这样有具体的比较，但我们如果不是专业的测试 webservice，或者说我们是客串测试 webservice，soapui 够用了吧。

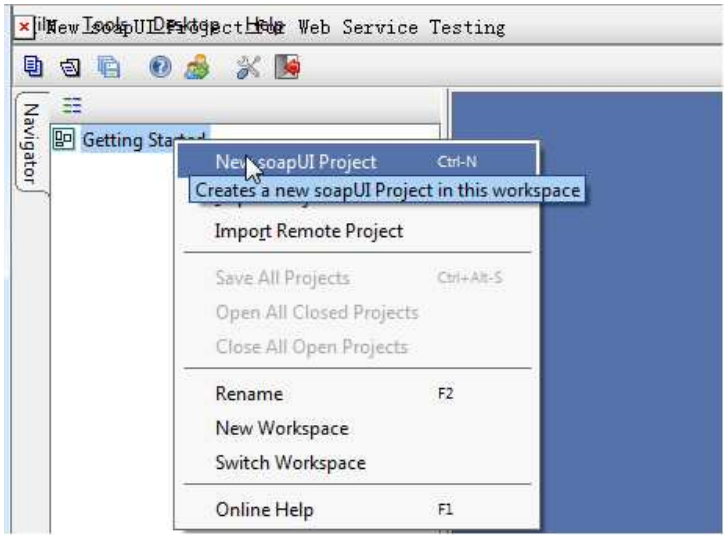
==========有关 SoapUI 的安装省略，按提示的图形界面安装即可==========

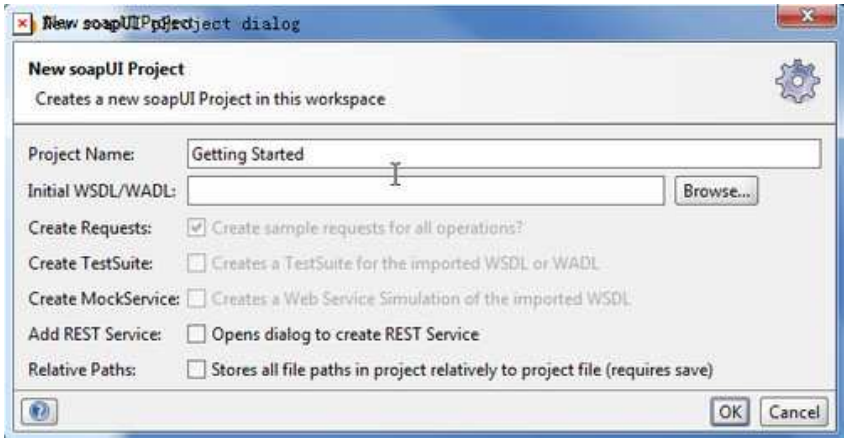
==========创建 project，添加 WSDL==========

project 的 soapUI 测试 webservice 的测试基础，在创建 project 之后你才能干其他事情：功能测试，性能测试等等工作。

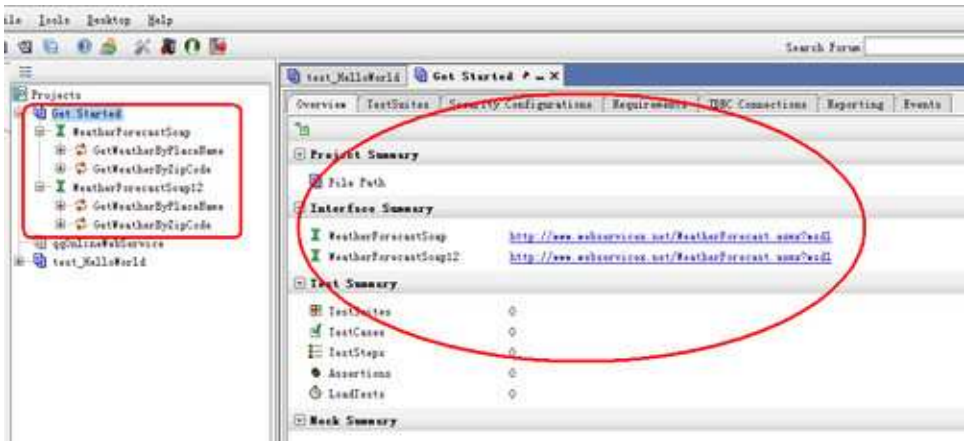
创建 project 和 WSDL 比较简单，右键--》【New soapUI Project】--> 输入 project name/WSDL--》【ok】

直接贴图好了：

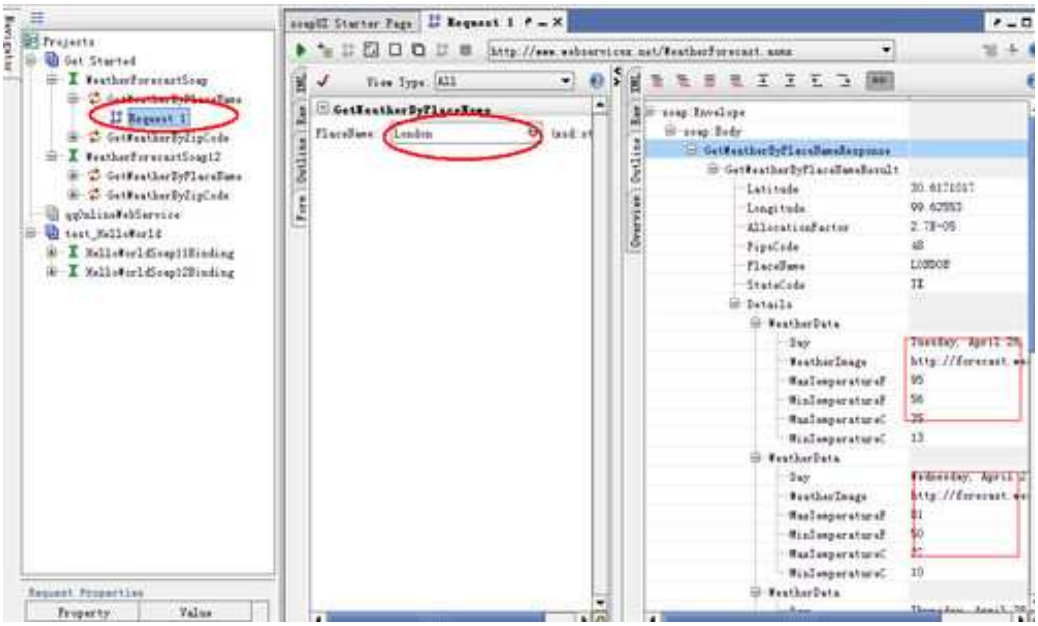




当然这里的 WSDL 也可以暂时不输入，只写一个 project name，后面在创建好了 project 右键添加也行。



现在可以直接运行刚才的导入的 webservice 的 request 了



第四节、功能测试基础

当创建好 project 并且导入了 WSDL 后，我们就可以开始创建测试用例以便开展测试了。SoapUI 有多种方法添加测试用例，且可以通过 Groovy 或者 javaScript 脚本来增强脚本的功能。

下面我们要做的事是：

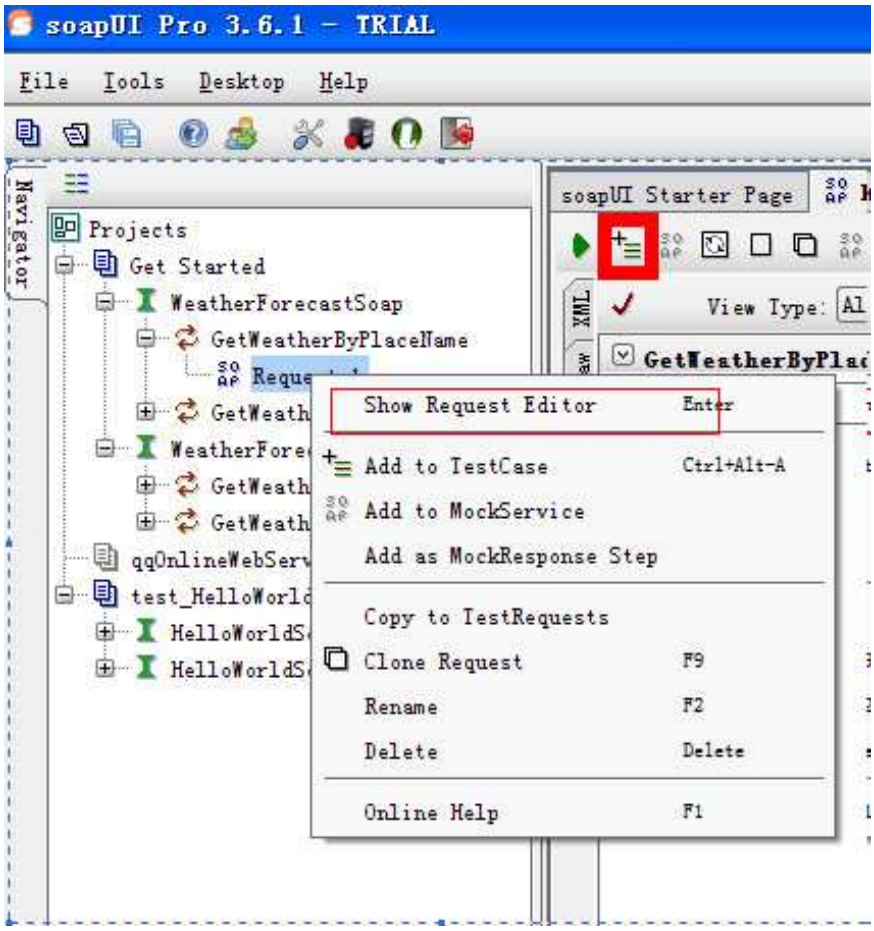
1 为 WSDL 创建一个 test case

2. 为 test case 加上一点断言 assertion

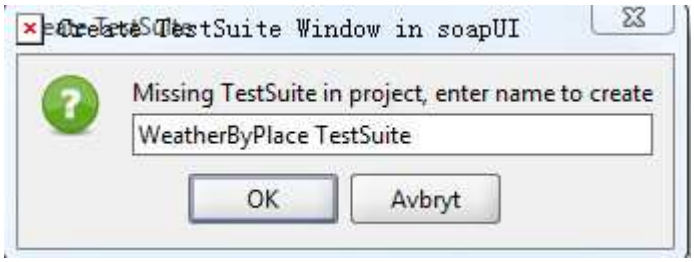
3. 运用 test case

为 WSDL 创建 test case

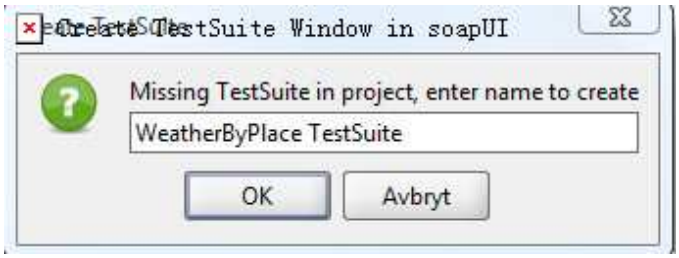
1.1 展开 Webservice，右键选择“Add to request ”或者在 request 页面选择添加 testcase



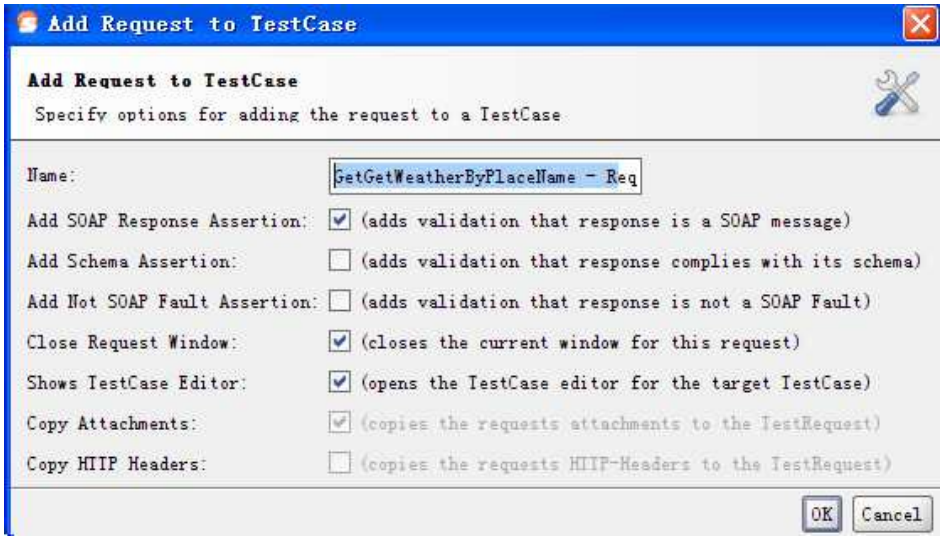
1.2. 在后续弹出的创建 TestSuit 对话框中，输入 TestSuit 的名称



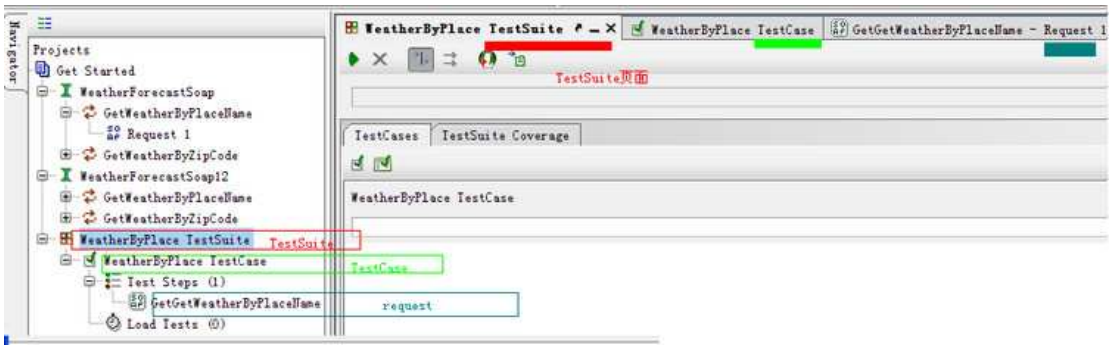
1.3. 确定后要求输入 TestCase 对话框



1.4. 接着为这个测试用例添加 request，这才是测试用例的核心



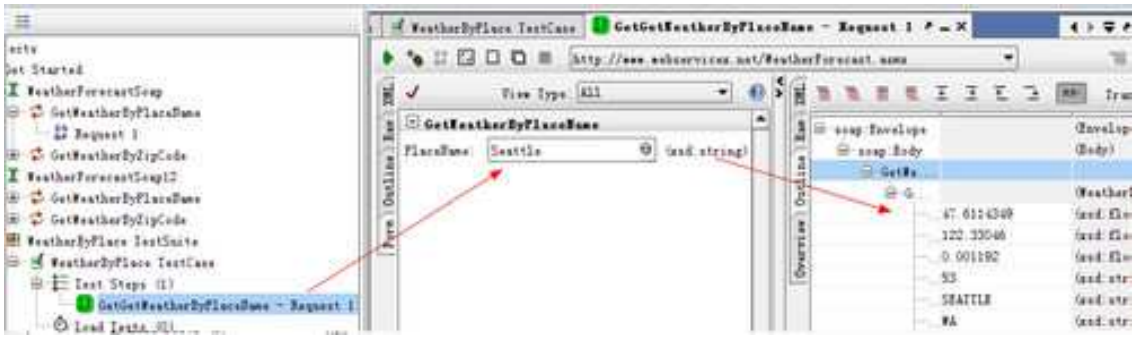
1.5.确定后就能看到如下的界面



这时我们可以看到 TestSuite --》 TestCase --》Request 的组织结构

另外就是点击不同的结构目录，右框中会根据点击的不同显示页面有所不同。

到了这一步，我们就可以打开 TestCase 中的 Request，输入参数，进行测试了。

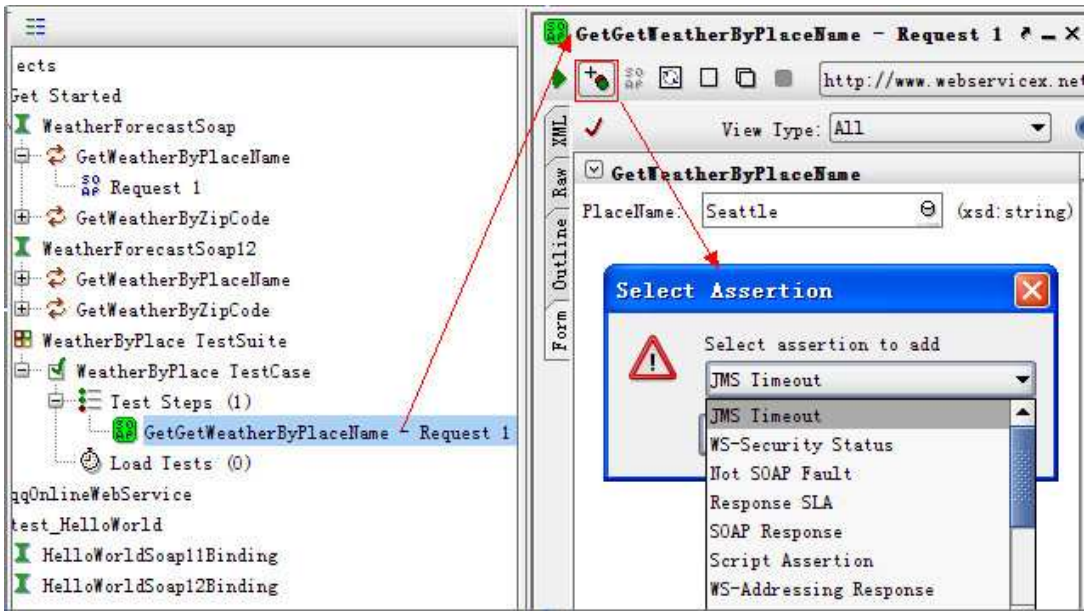


到了上面，显然我们的测试用例不太可能这么简单，需要添加一些检查点（就是 soapUI 的 assertion）。下面我们就添加 assertion

==========

添加 Assertion

2.1 双击 Request，打开 request 页面，点击“Adds an assertion”



2.2 在弹出的选择 Assertion 中选择断言的类型

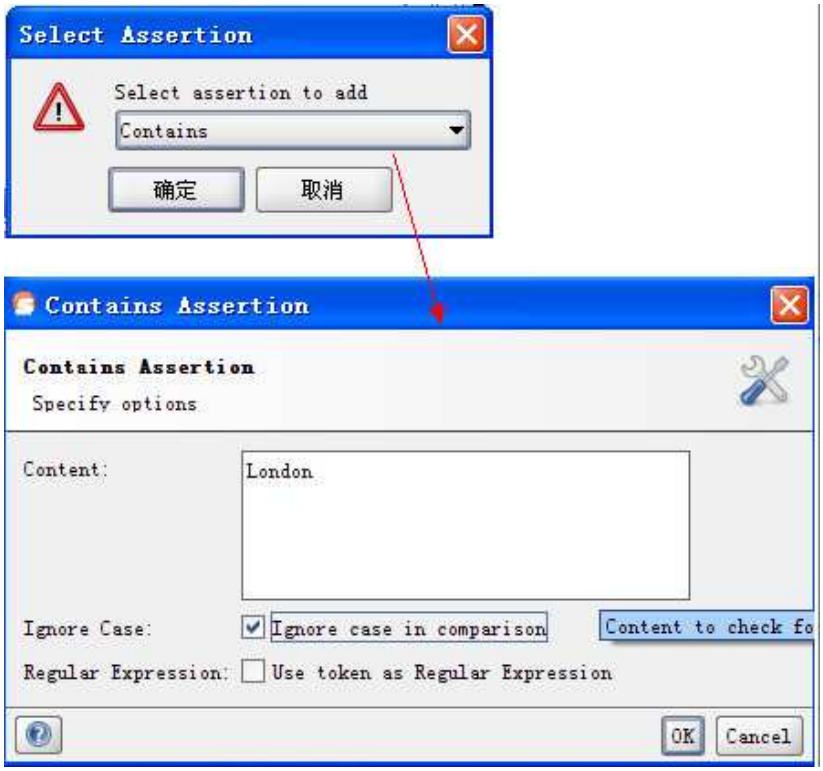
我们从这里能看到 soapUI 的 assertion 类型有

类型	说明
Schema Compliance	
Contains	检查是否存在特定字符串
Not Contains	检查是否不存在特定字符串
SOAP Fault	
Not SOAP Fault	
SOAP Response	
Response SLA	
XPath Match	Xpath 表达式的结果是否是期望值
XQuery Match	
Script Assertion	自己写脚本判断
WS-Security Status	
WS-Addressing Response	
WS-Addressing Request	

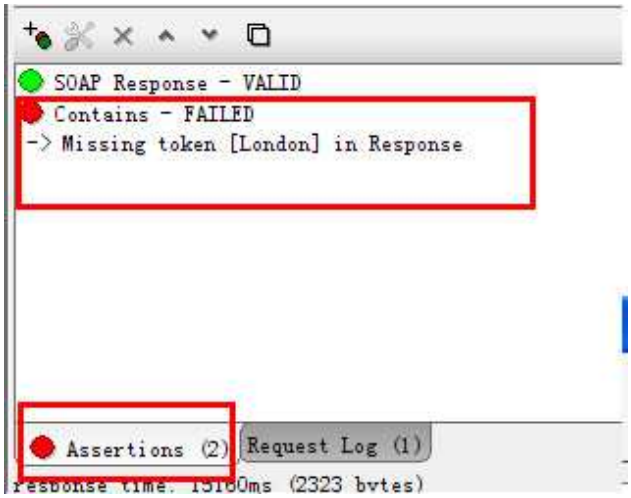
我们添加一个 Contain 类型的断言（或者叫检查点）

2.3 在接着的 Contains 断言的具体属性中，输入要检查的内容，

例如我们后面要查询 London 的天气，这里我们输入 London，并勾选上忽略大小写



确定后，你会发现你 assertion 是 Failed



这里要说明的是

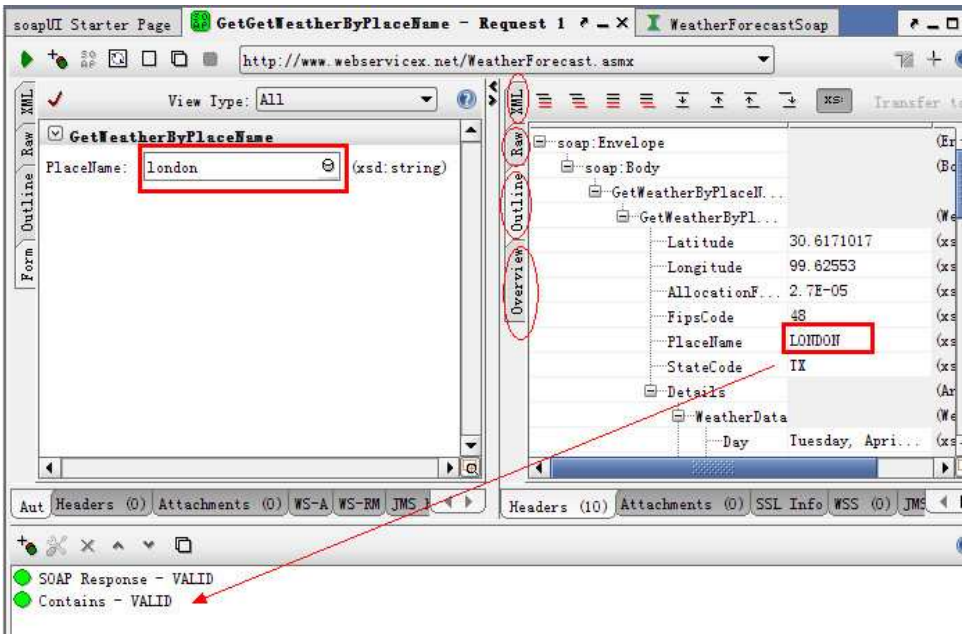
1. 失败是正常的，因为我们之前运行过这个用例，在添加 assertion 后它会立即在上一次执行的结果中执行一次 assertion

2. 我们只添加了一个 assertion，为什么有两个呢。因为 SOAP Response 是测试用例默认自带的 assetion

==========

执行用例

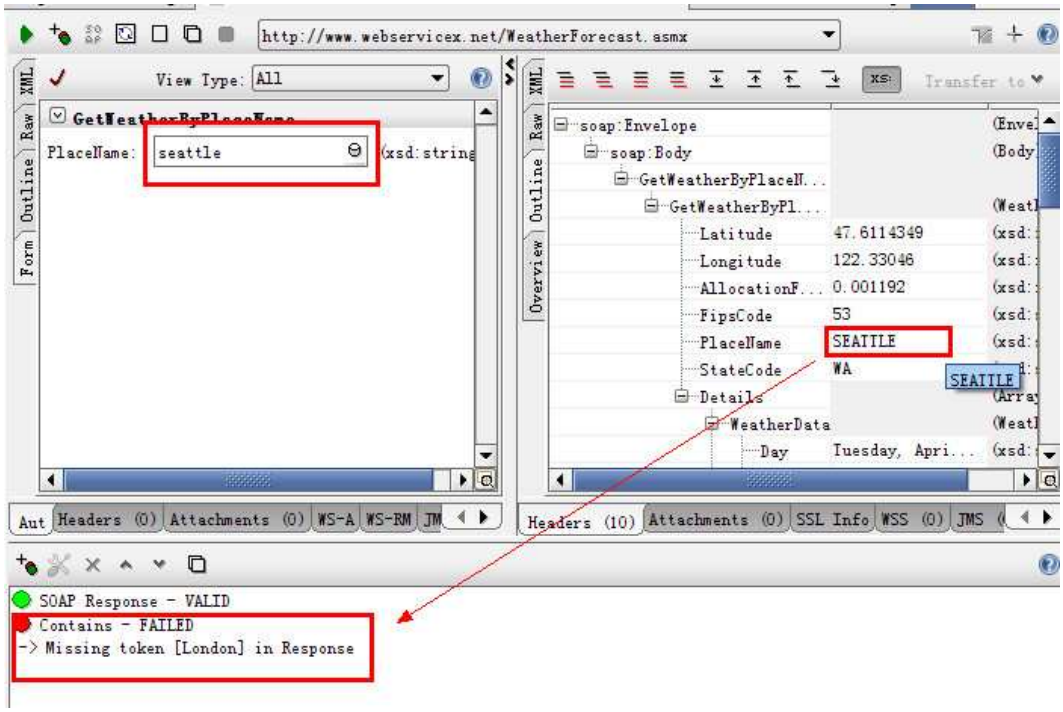
3.1. 点击执行，我们可以看到测试用例执行是成功的



还要注意，测试的结果，可以以多种方式显示：XML / Raw / Outline / Overview

其实我们再仔细看 request 也是有几种显示形式的 xml/ Raw /Outline /Form ，只是一般我们用 form 形式的而已

3.2. 那么我们修改一下查询条件为 seattle，测试用例还能成功吗？



PS:

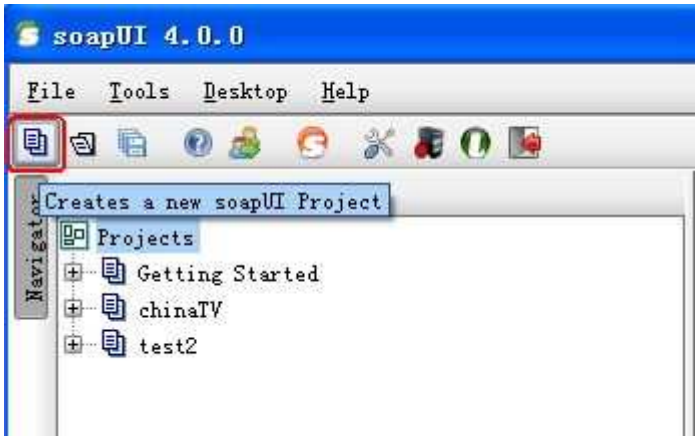
需 要说明的是，我们在测试网上提供的免费的 webservice 的时候，有时候 webservice 暂停了服务，这时我们测试是永远也不能成功的，那么我们 如何判断 webservice 的服务已经暂停了呢，我们只需要在浏览器中打开 webservice 的 WSDL 地址，如果能显示 xml 文件，就说明服务是好 的。当然这是一般的判断条件，并不绝对。

第五节、创建一个性能测试

本节我所加的，创建一个简单的性能测试计划。由于接触 soapUI 不深，所以只是罗列了操作步骤，以及最后性能的的一些设置。不过，相信根据下面的操作，不熟悉 WebService 和 soapUI 的同学，一样可以完成简单的 WebService 性能测试。

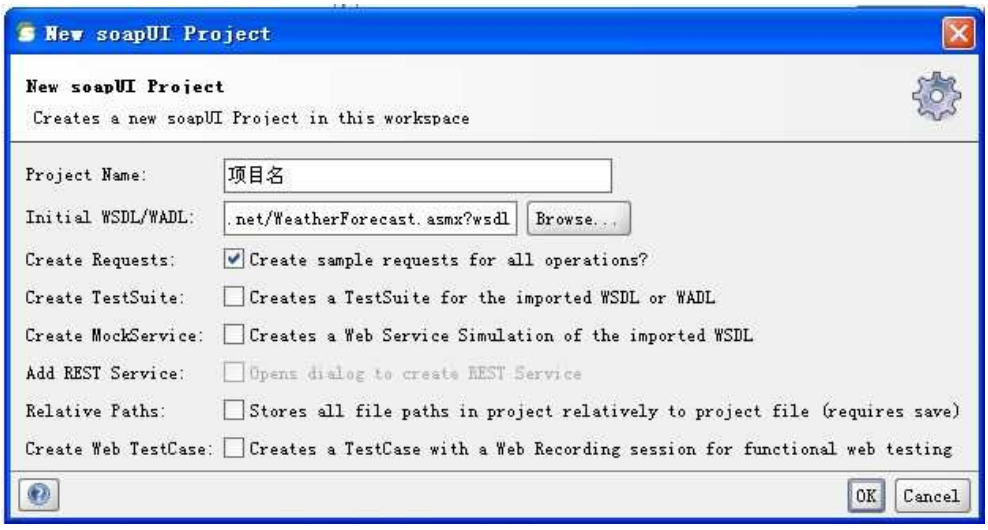
第一步：

新建一个项目：点击新建按钮就行了。



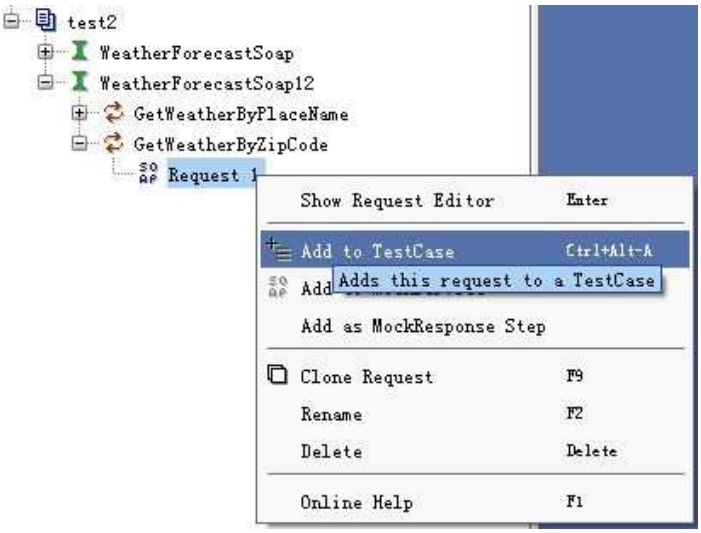
在打开的窗口中填写你项目名，顺便也把你要测试 WebService 地址也一块填写的吧！当然，也这里也可以空都，留在之后添加。

http://www.webservicex.net/WeatherForecast.asmx?wsdl



第二步：

创建一个测试计划，右键点击 Request---Add to TestCase

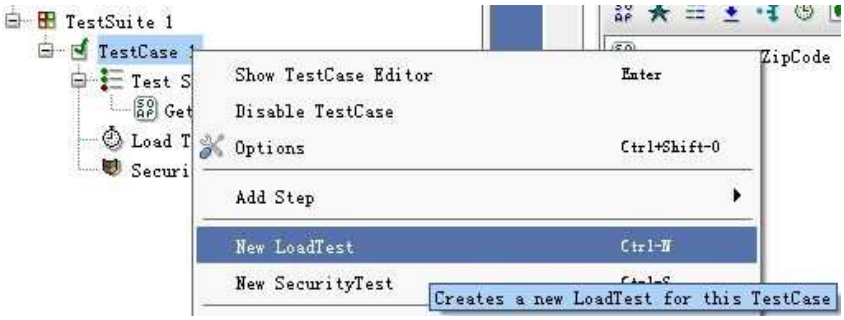


然后会弹出对话框提示，点　确定----确定-----OK 计划就创建成功了。

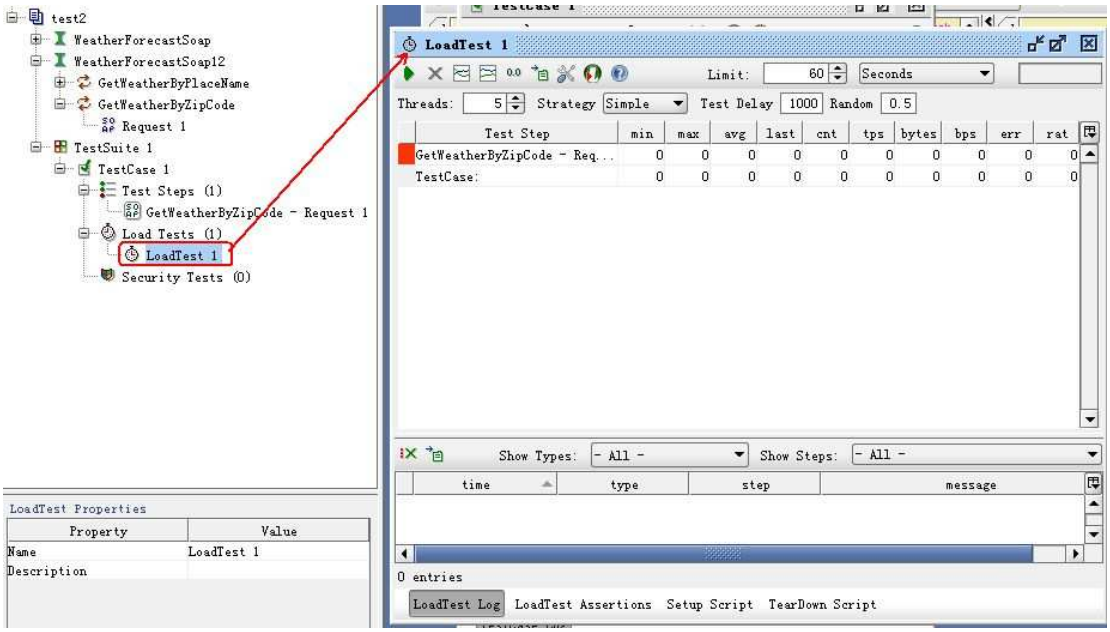
其实，GetWeatherByZipCode--Request 1　是功能测试页面，因为咱要做的是性能，这里不在细讲，有兴趣可以看一下。

第三步：

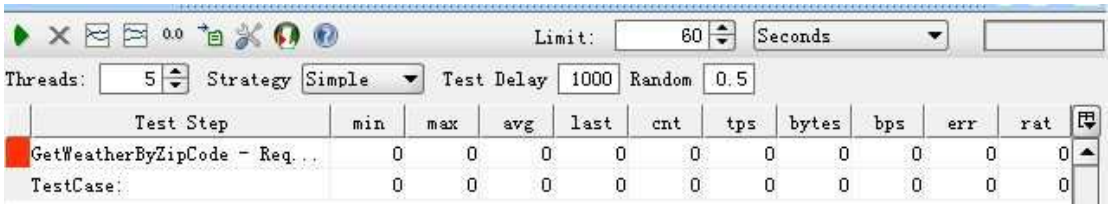
创建一个性能测试，右键点击 TestCase---New LoadTest



填写测试名，弹出性能测试窗口。



窗口介绍：

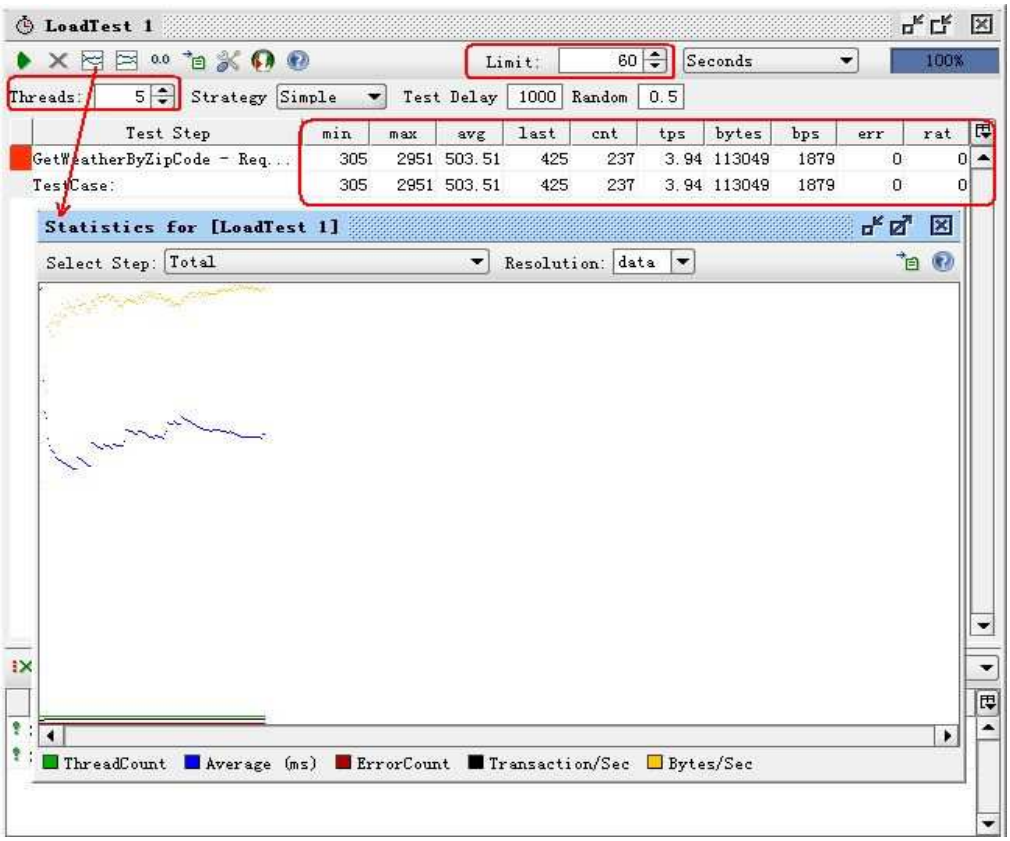


运行：点击左上角的绿色按钮。

Threads ：设置虚拟用户数。

两个折线的按钮：打开是图形结果

Limit:运行时间（s 秒）



第六节、SoapUI 自带的 WebService 实例

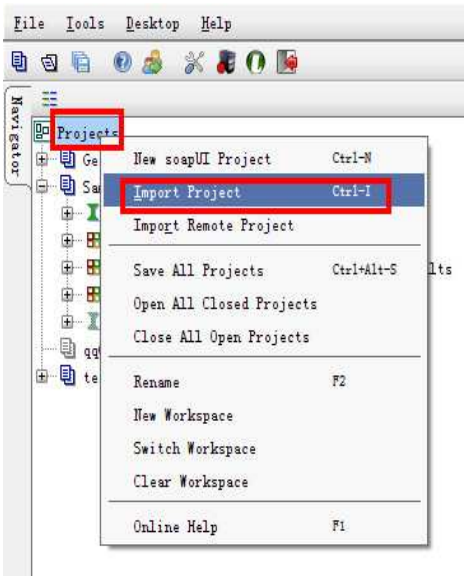
soapUI 教程很体贴，为我们做了一个 webservice 的实例，并且利用 soapUI 的 webservice 模拟功能（MockService）为我们学习 soapUI 提供了方便。下面就总结一下这个自带的 webservice 以及其他内容。 所有学习来自于 http://www.soapui.org/Getting-Started/web-service-sample-project.html 只是其他对相应的 request 没有做太多的说明，我进行学习进行一些补充说明。

=======soapui 自带的 webservice 实例======

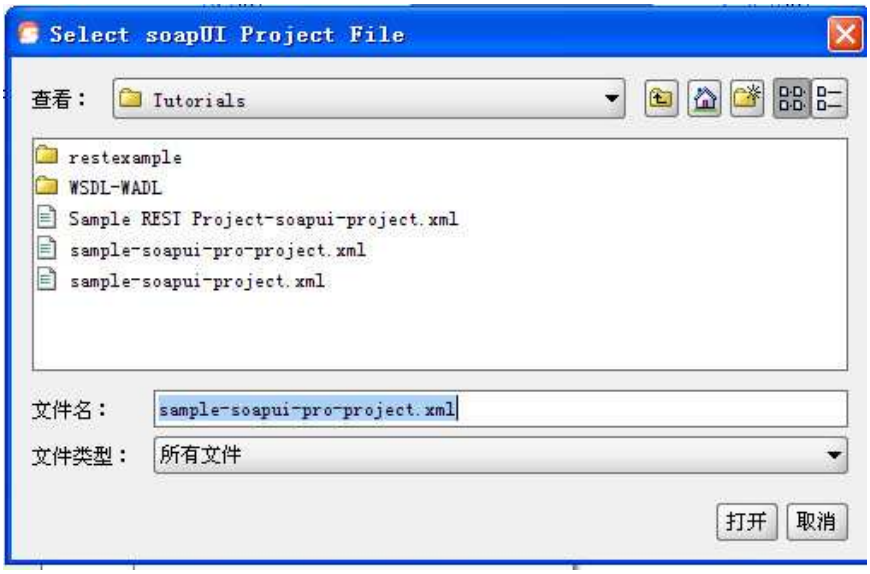
我们用的东东是 ..\soapUI-Pro-3.6.1\Tutorials 文件夹下的 sample-soapui-pro-project.xml

---

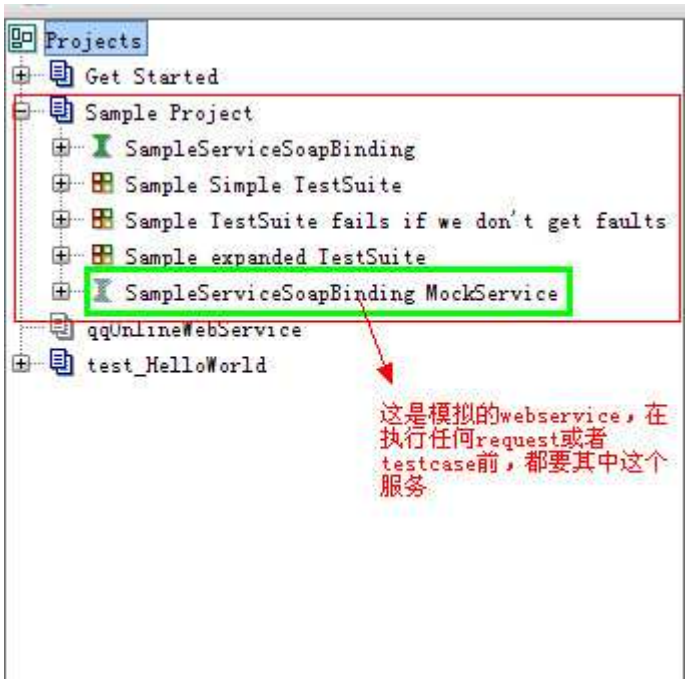
1. 打开 project （import preject）。在 Project 上右键选择“Import Project”



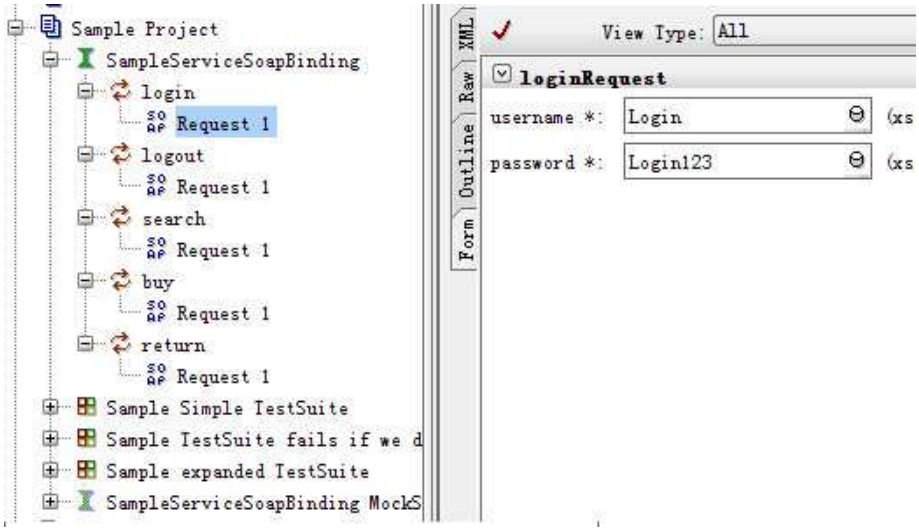
2. 再弹出的选择框中浏览选择 sample-soapui-pro-project.xml



3. 完成后的显示结果



4. 展开 SoapBinding



从这里我们可以看到这个服务有 5 个 Request，login/logou/search/buy/return，在右方可以看到有 request 的 form 格式，也就是这个 request 需要的参数。分别点击后能看到不同 request 的不同参数。

请求	参数	返回/说明
login	Username Password	Sessionid
logout	Sessionid	Sessionid 销毁
search	Sessionid Searchstring	Searchstring 的范围: Item 1 / Item 2 /Item 3/Item 4 Item 5 / all
buy	Sessionid buystring	Buystring 和 searchstring 一样

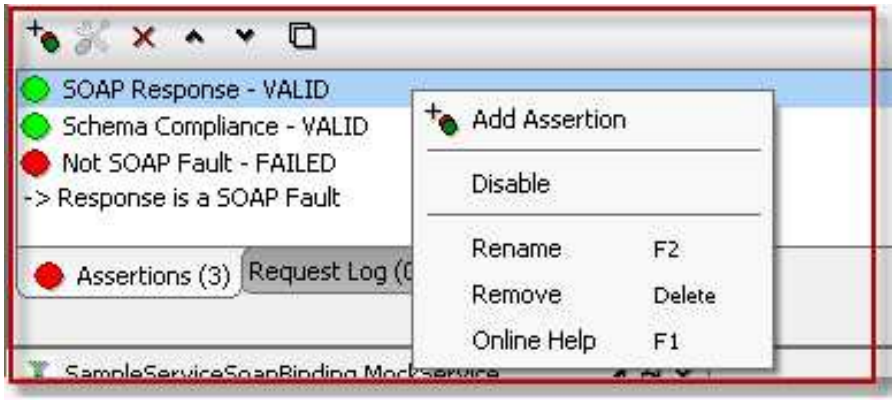
上面没有介绍 return 请求，是我没看懂这个 return 是干什么的，自带的测试用例中也没有怎么用这个 return。

在学习中利用这个服务时，可以分别点击以上的请求自己试试看。在执行前，一定要运行 MockService

第七节、SoapUI 的 assertion

通过第四节的学习，我们知道了 soapUI 提供的多种类型的 assertion，那么我们来看看常用的 assertion

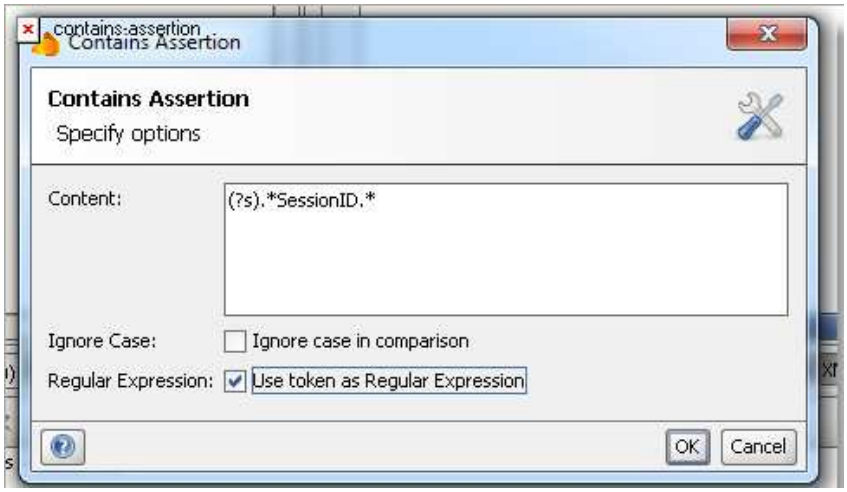
首先看看 assertion 的管理



我们常用的

- Contains - checks for the existence of a specified string (see below)
- Not Contains - checks for the non-existence of a specified string (see below)
- Reponse SLA - check the response time against a specified value (see below)
- XPath Match - compares the result of an XPath expression to an expected value （估计用的不多）
- XQuery match - compares the result on an XQuery expression to an expected value (估计用的不多)
- Script - runs an arbitrary script that can be used to validate the received message as desired

其中的 contains/Not Contains 基本差不多，添加的界面如下



其中一是要注意可以忽略大小写

二是可以用正则表达式，这个东东还是可以去实践实践的，学习地址在

http://www.jdocs.com/javase/7.b12/java/util/regex/Pattern.html

第八节、SoapUI 的 MockService 功能

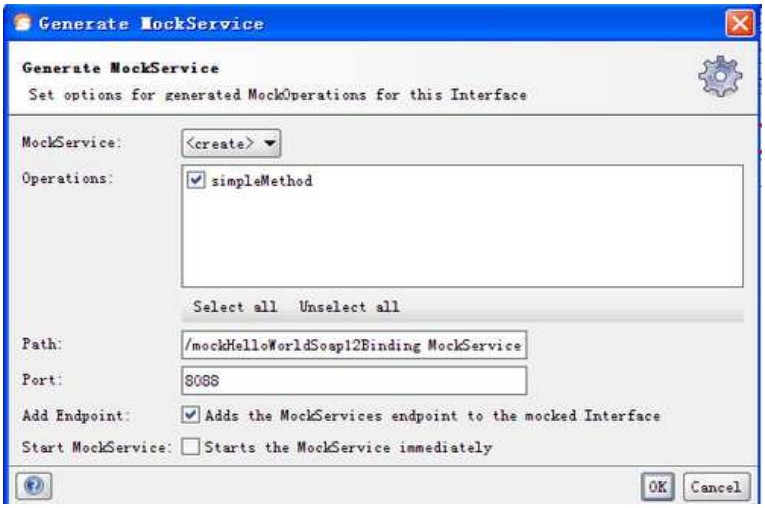
soapUI 的 MockService 功能

事 实上，我到现在也没搞明白 MockService 到底是在模拟什么，根据我看的，应该是模拟的服务的 response，并且这个服务的 WSDL 文件已经创 建好了的。不明白如果 WSDL 都已经好了，那不是是 webservice 好了吗？先不管了，学习如何创建再说吧，以后再体会。

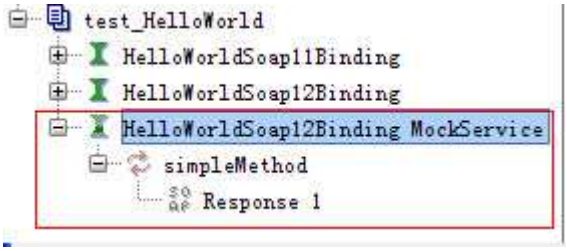
还是以 soapUI 学习（2）中创建的 webservice 为例来创建 MockService 吧

1. 创建 project，使用 wsdl：http://localhost:7890/axis2/services/HelloWorld?wsdl
2. 创建 MockService

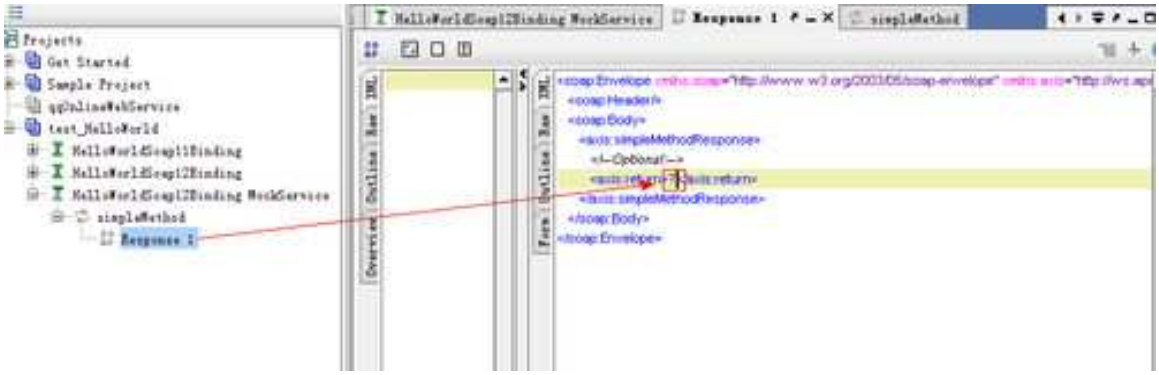
选择一个 soap 右键选择 Generate MockService



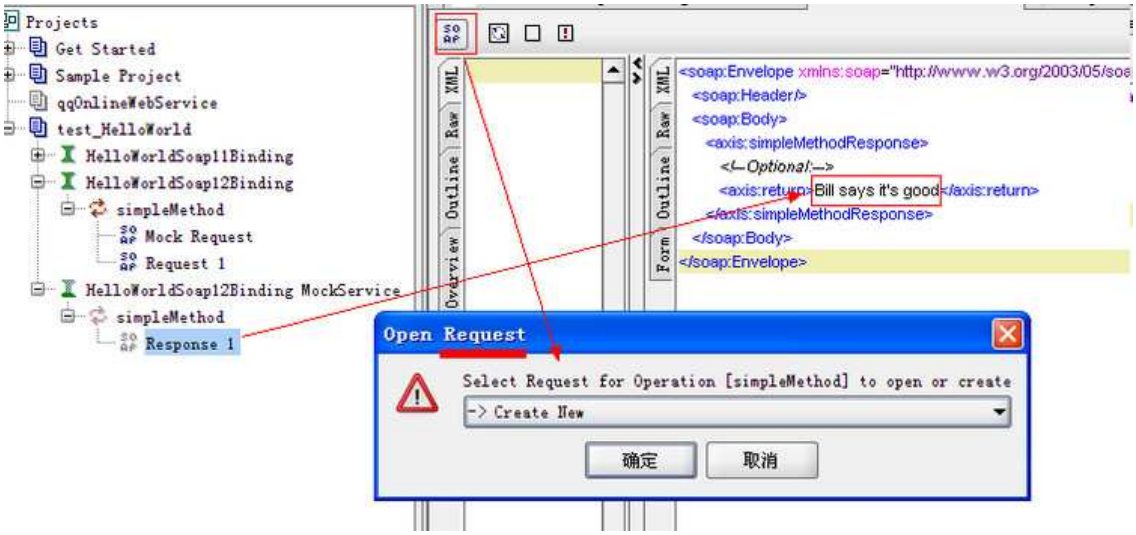
创建好后就如下



3. 编写 MockService 的 Response，也就是修改 Response （其实说白了，做模拟就是模拟返回值，输入的参数我们当然是知道的）

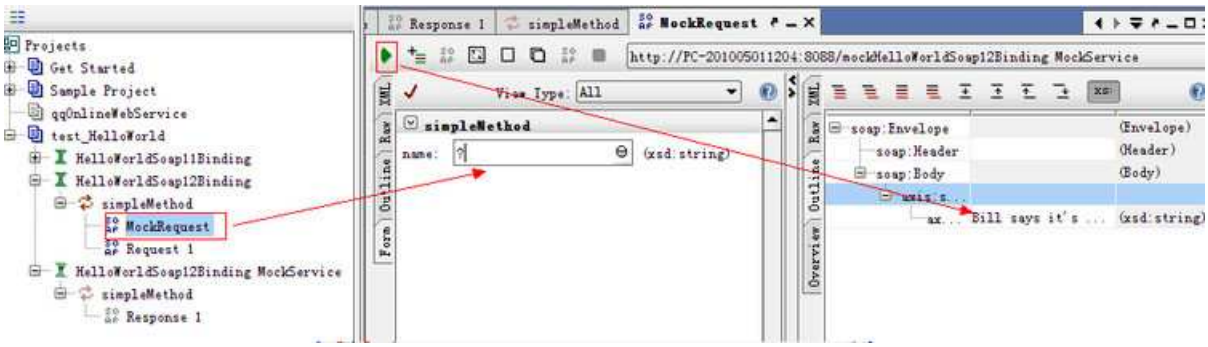


将？改为一个模拟的返回值，例如我们这里改为“Bill says it is good”



4. 创建一个 Request 名称为：MockRequest ，

并且关闭 tomcat（为了看出是 MockService 在起作用，而不是原来真实的 webservice 起作用），启动 MockService 后执行 MockRequest



因为这里没有用到 name 这个参数做任何的事情，所以无论在 name 参数中输入什么，都返回“Bill says it's good”

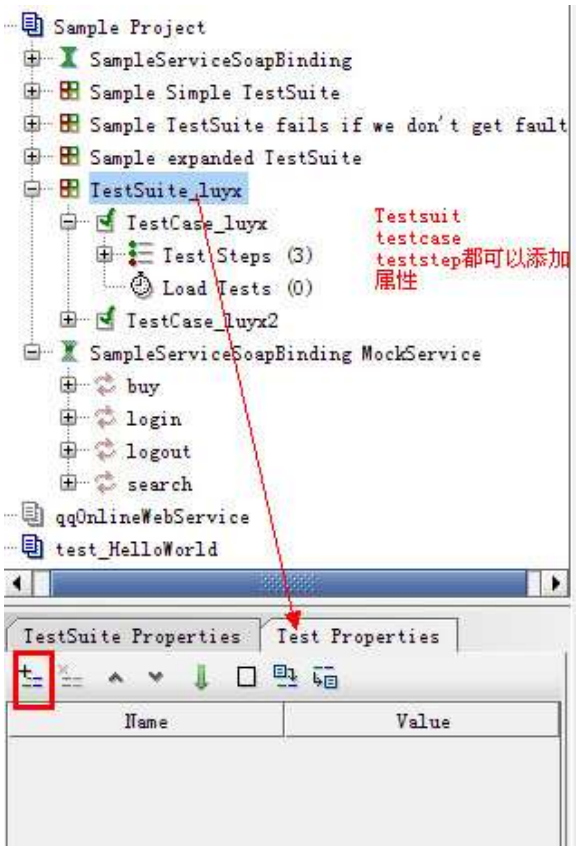
有关 MockService 的功能，官方网站上讲的很多，这里只是了解一下而已，我想除非是真正大量测试 Webservice，否则对 soapUI 基础功能有所了解就能应付一般的 webservice 测试需要了吧。

第九节、Transferring Property Values 传递属性值

- soapUI 的属性概念
- Transfer property ---直接传递到具体操作
- Transer Property ---通过变量存放

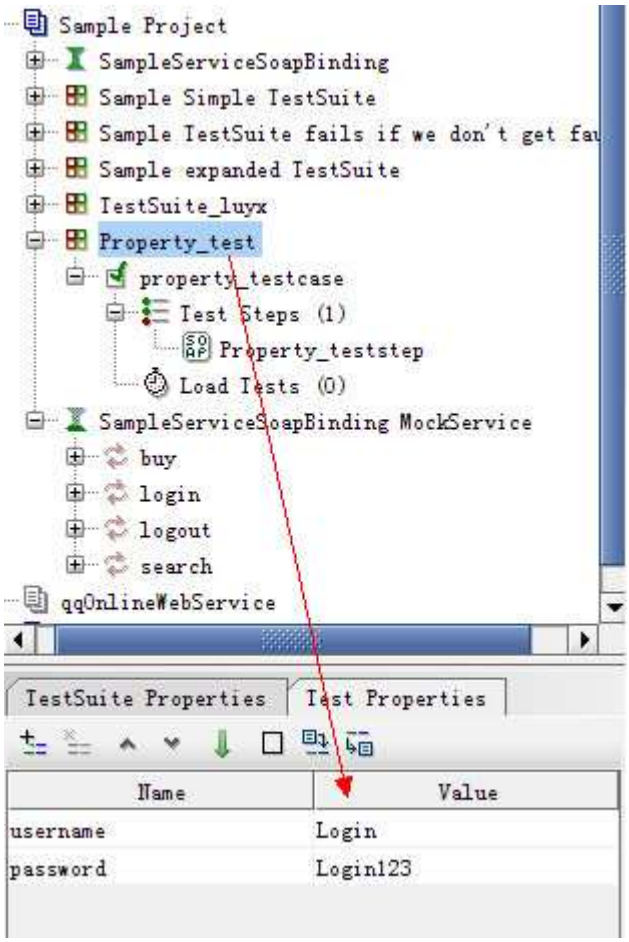
==========soapUI 的属性概念==========

经过学习，我们知道，soapUI 的每个层次都可以创建属性 Property

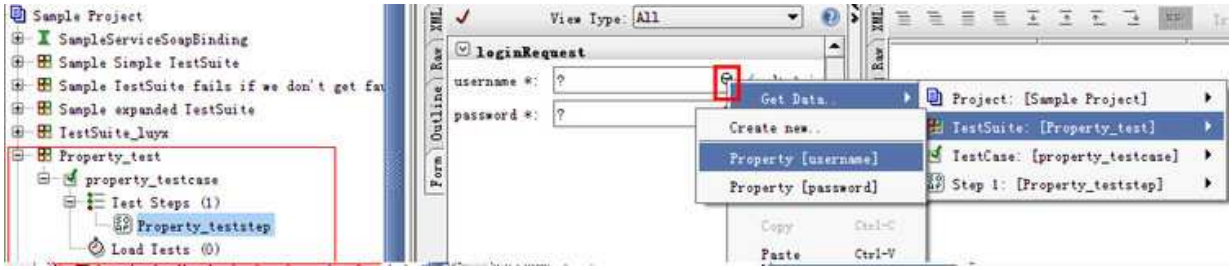


这些属性，可以用来传递参数，例如可以在 testsuite 中创建两个属性：username/password，用来进行登录

1. 首先我们创建一个测试用例，添加登录 request
2. 在 testsuite 层次上创建两个属性 username=Login / password=Login123

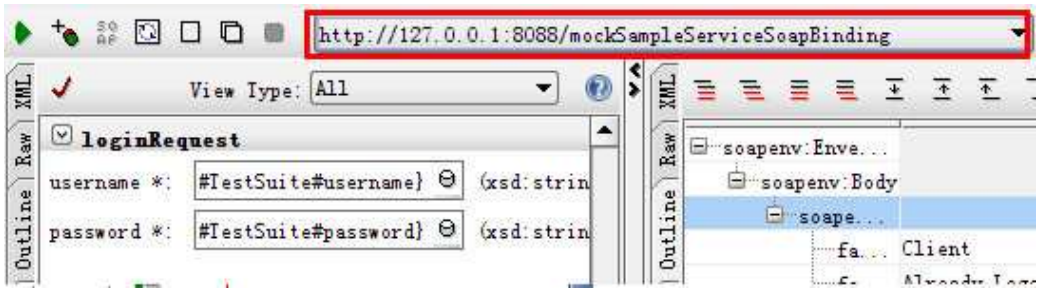


3.打开 request，在 form 形式下点击输入框后面的“Get Data”按钮选择创建的属性参数。分别设置 username/password



4. 接着会弹出初始值对话框，注意这里输入的值要和创建属性时的值一致，不然输入的值就把原来属性值覆盖了

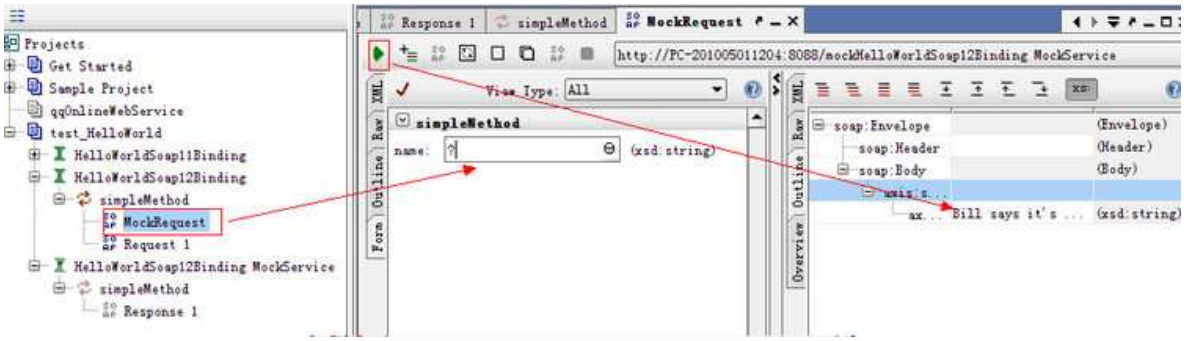
另外，注意我这里是使用的 soapUI 自带的 Sample Project ，并且要使用 MockService 的 Endpoint：



5.设置好后就可以运行啦。

按上面的操作应该可以运行成功的。如果登录失败，提示“Already Logged In”，那么只要把 MockService 重启一下就可以了。

其实有关 soapUI 的 Property 的应用，在自带的 sample project 中应用的比较充分，值得学习

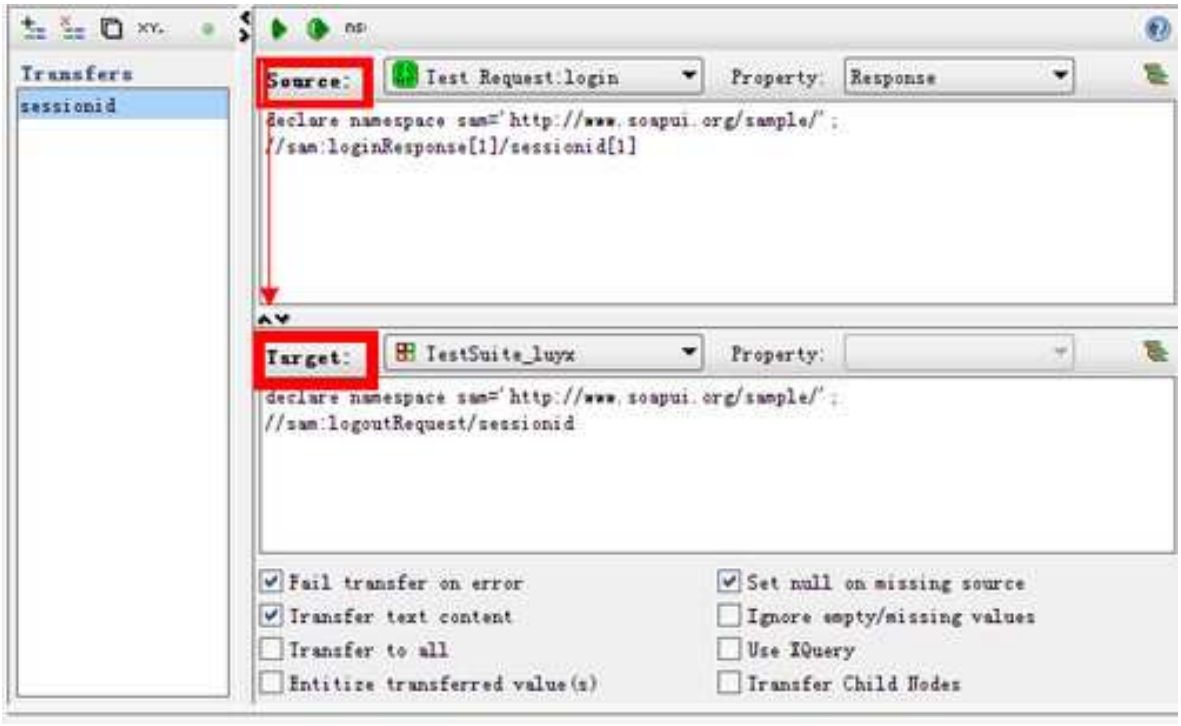


==========

上面还没说到正题呢，我们要说的是 Property transfer。

我们要进行 login---》logout，其中 login 产生了 sessionid，并且使用这个 sessionid 进行 logout。

首先看看 Property transfer 的主要页面，就是从 Source 传向 Target。



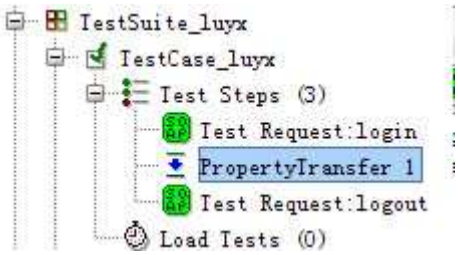
根据 soure 到 target 的两种方式，我们进行两种说明。两种方式分别是：

一种是直接将 soure 获得的值传到下一个具体的步骤；

另一种是将 soure 中获得的值存在一个变量中，以后直接引用，这个的好处是可以使用多次。

==========将 soure 获得的值传到下一个具体的步骤==========

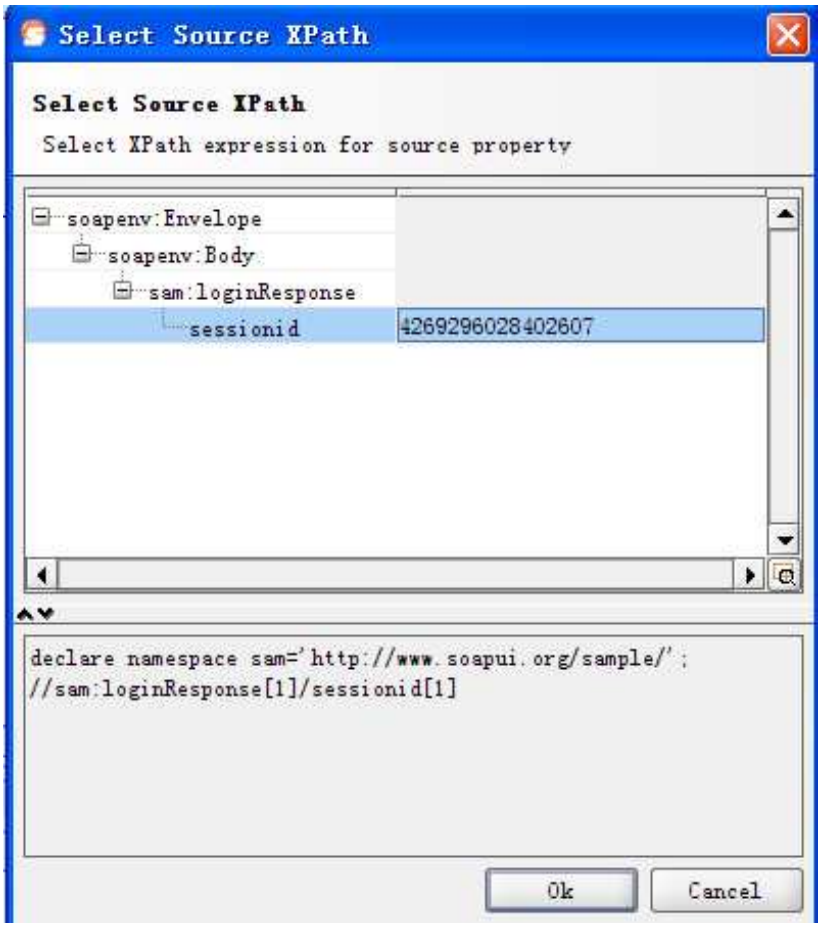
1. 首先创建测试用例，添加 3 个步骤：login --》PropertyTransfer --》logout



2. 双击 PropertyTransfer，打开编辑页面

添加 PropertyTansfer，名称：SessionID

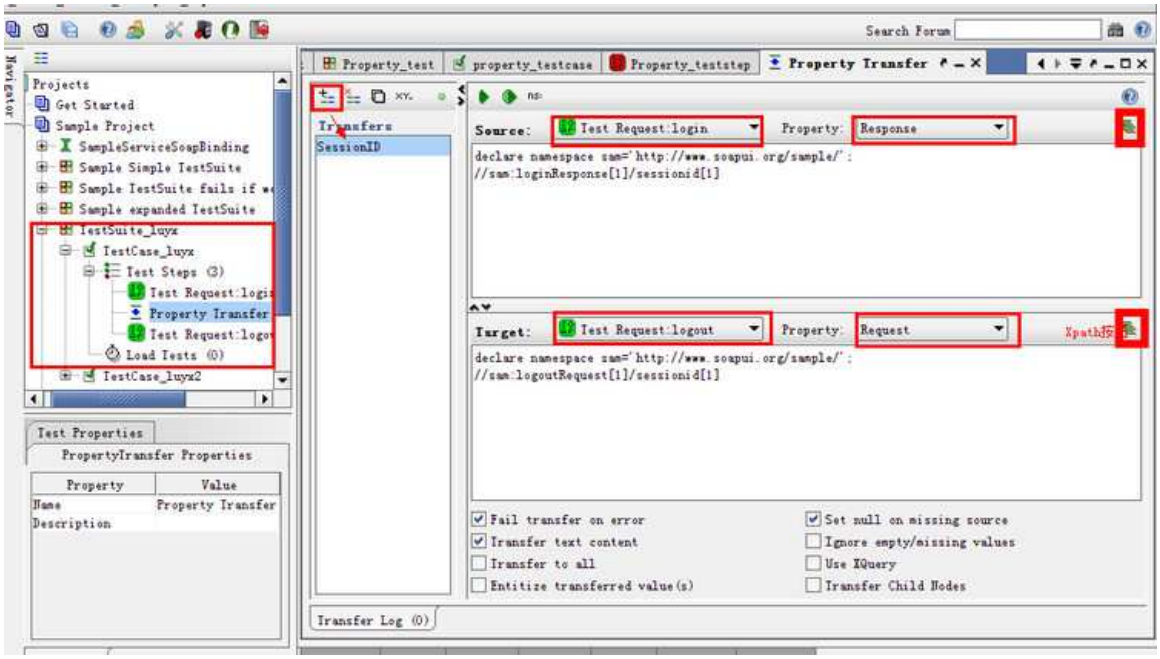
Source 中选择：Test Request:login　　Property:Response 点击 Xpath 图标设置 Xpath



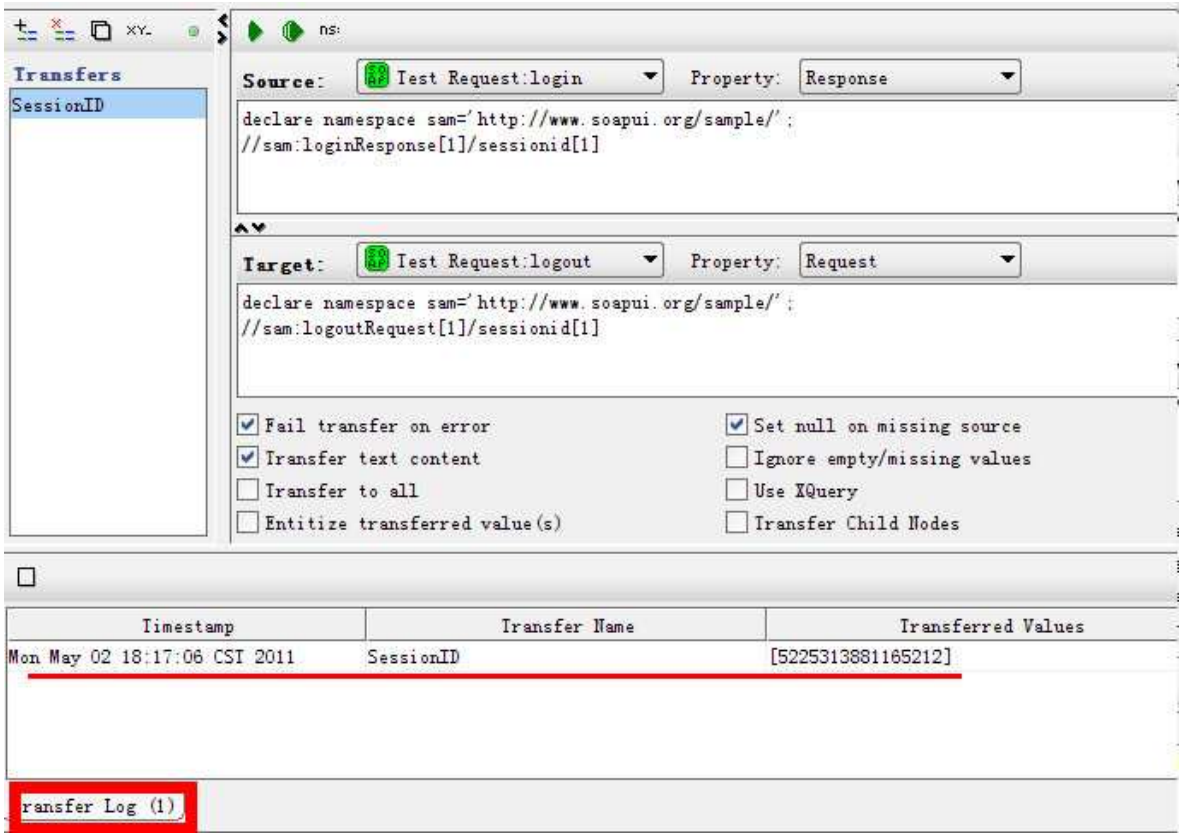
（在 sessionid 的数字上点击一下，就会出现下面的 Xpath）

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



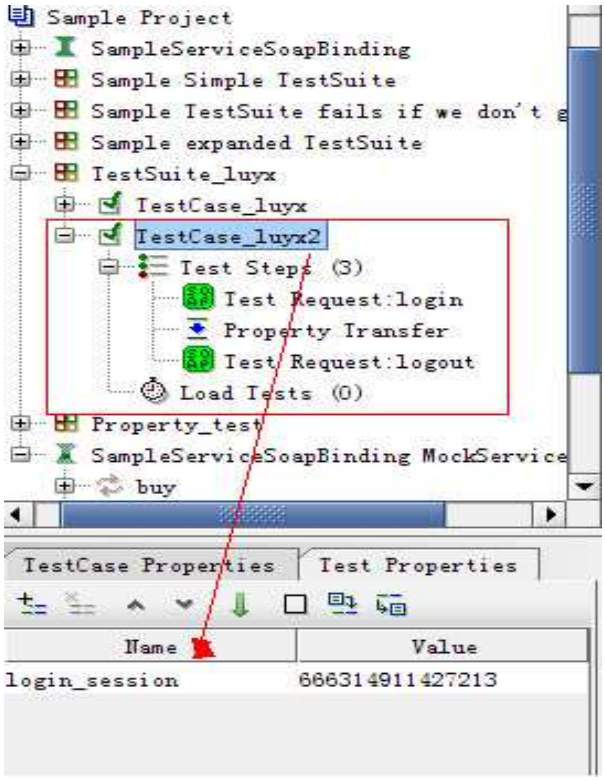
3. 试着运行一下这个测试用例，运行后切换到 PropertyTransfer 页面



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。

==========将 soure 中获得的值存在一个变量中==========

1. 同样的，我们创建一个测试用例，login --》PropertyTransfer --》logout

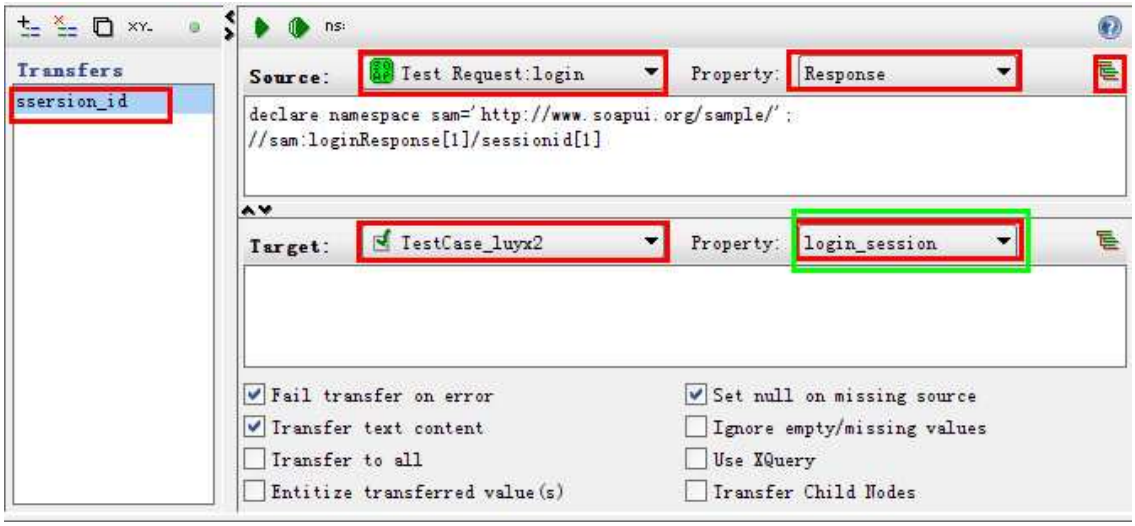


2. 不同的是,我们在 Testcase 的层次上新建一个属性 login_session,值随便输入一个即可。这个 login_session 就用来做变量存储从 login response 中获得的 session

3. 打开 PorpertyTransfer 的编辑页面

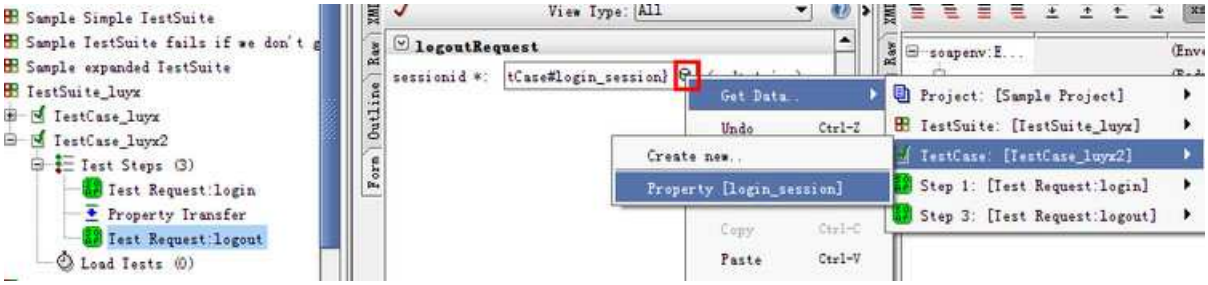
新建 session_id ， source: Test Request:login Property:Response, 选择 Xpth，选中 Response 中的 sessionid

target： TestCase_luyx2 Property:login_session

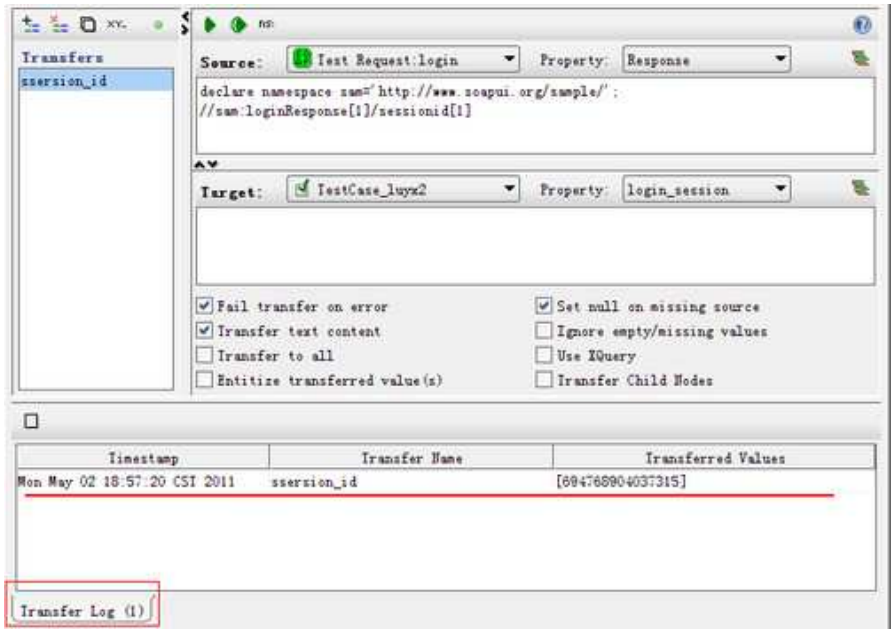


4. 接着，打开 logout，为 logout 的 session 选择参数，选择后的结果是

“${#TestCase#login_session}”

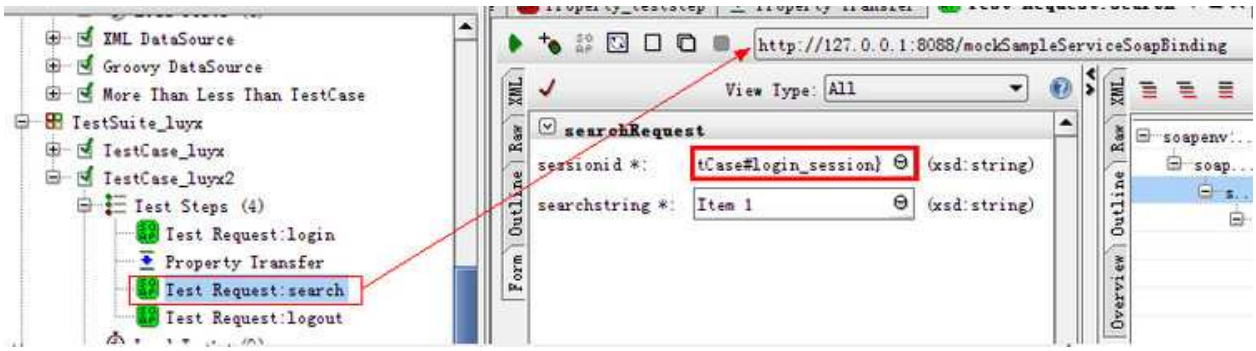


5. 一下 OK，运行，切换到 PropertyTransfer，同样有 log 显示



这就完成了将 soure 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 searchestring 一个固定的值，运行也是可以的哦



第十节、soapUI 的 DataSource

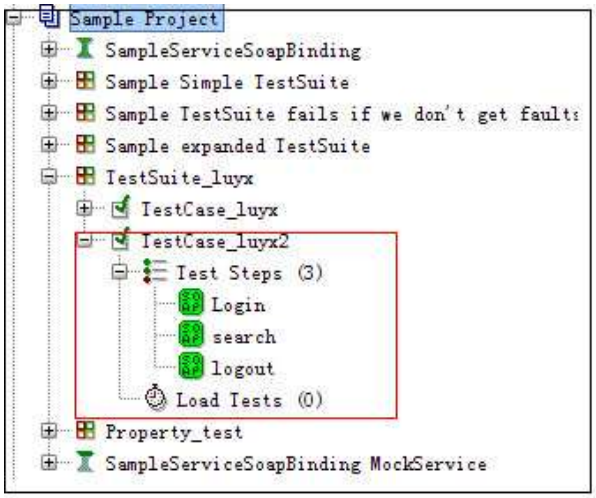
以 soapUI 自带的 sample project 为例，学习利用 DataSource 进行 login --》search（循环）--》logout

DataSource 的类型有：Data Connection/ Grid / File/ XML /groovy /excel /directory/JDBC

为了让 DataSource 能循环起来，还要和 DataSource Loop 结合，下面分布学习 Grid / File /excel /Data Connection

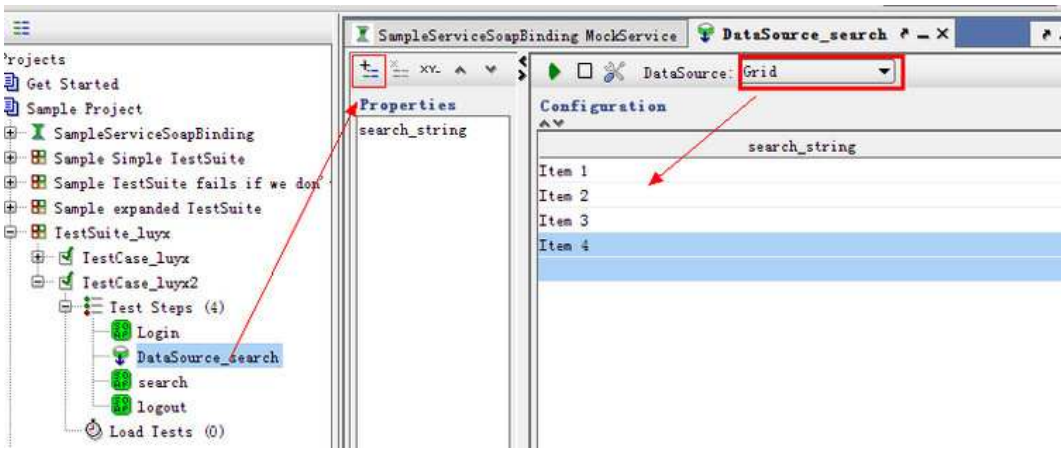
1. 创建 testcase，添加基础的 step：login/ search/ logout

并且在 search 中对



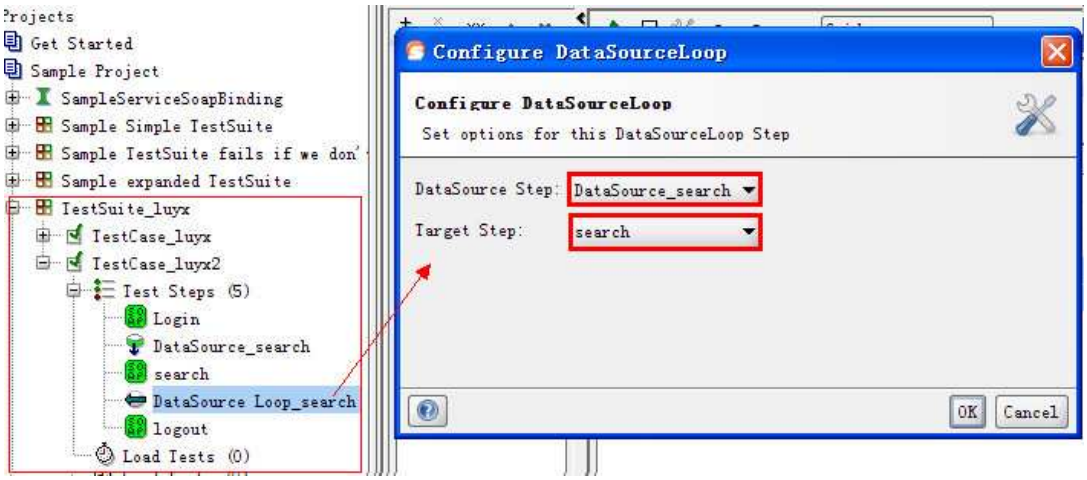
2. 新增 step：DataSource

并且新增一个 Property:search_string, 选择类型：Grid 新增参数：Item 1 / Item 2 / Item 3 / Item 4



在新参数后，可以运行 DataSource 中的绿色运行按钮，可以测试一下 datasource 是否可用（下面有 data log 查看）

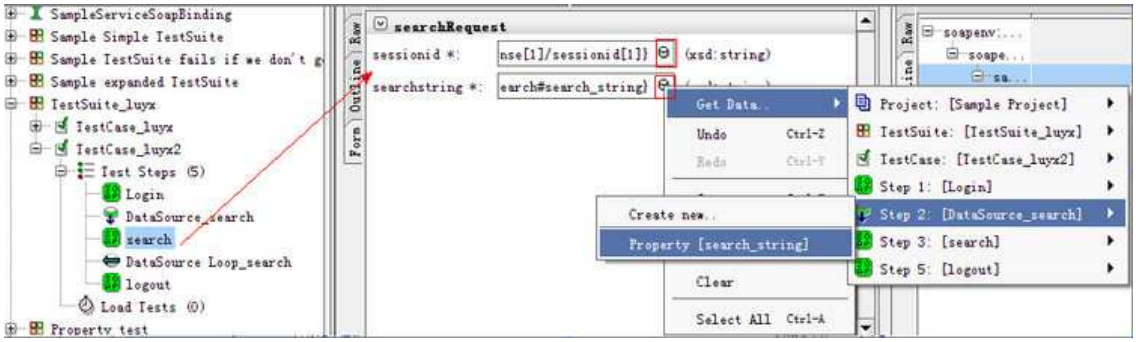
3. 新增 step:DataSource Loop，在 DataSourceLoop 配置对话框中选择 datasource step，选择 target step



4. 在 search request 中配置相关 session id / searchstring

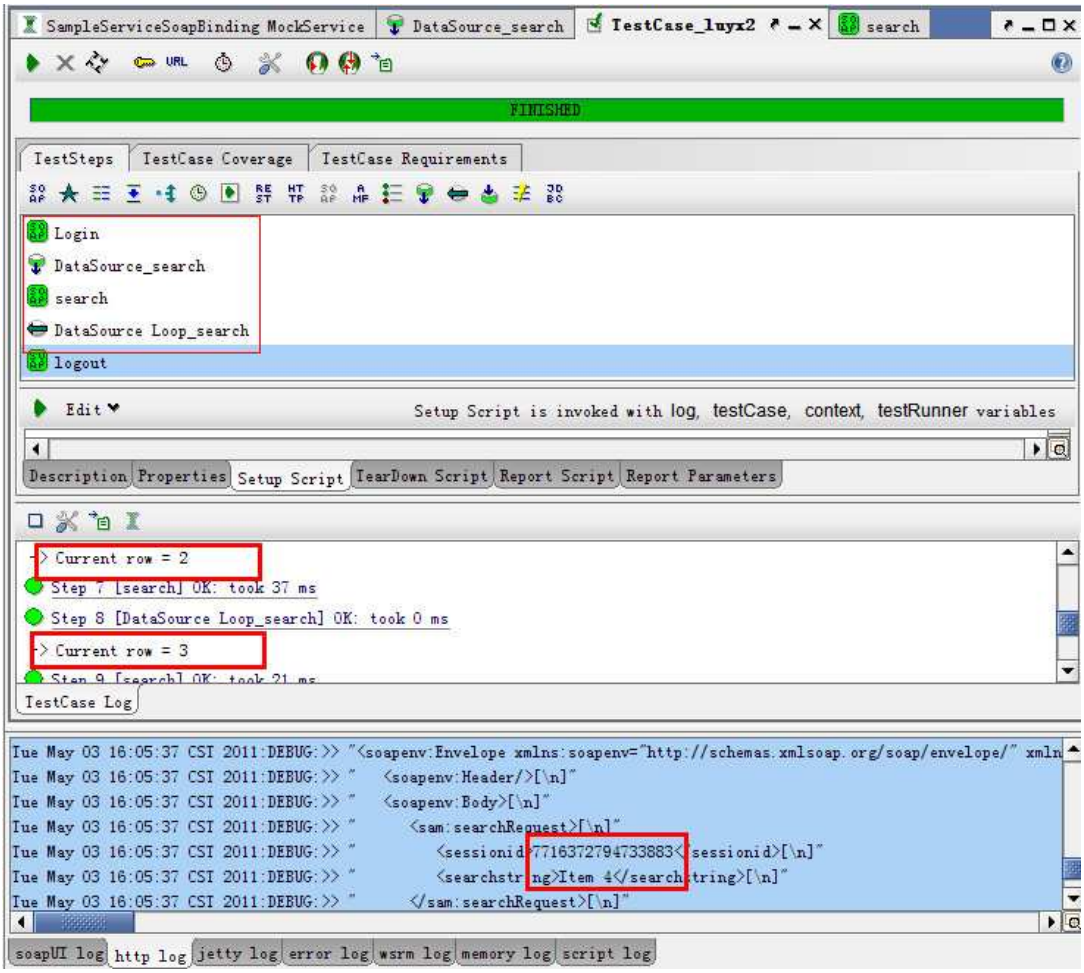
session id 的参数设置，就是选择 login response 产生的 session （之前的学习中有介绍过）

searchstring 的参数设置类似，这里我们选择在 DataSource 中添加的 search_string



5. 以上设置好后，就能运行测试用例了，

通过 TestCase log /Http Log 我们能看出测试用例确实利用了我们添加的参数在循环测试



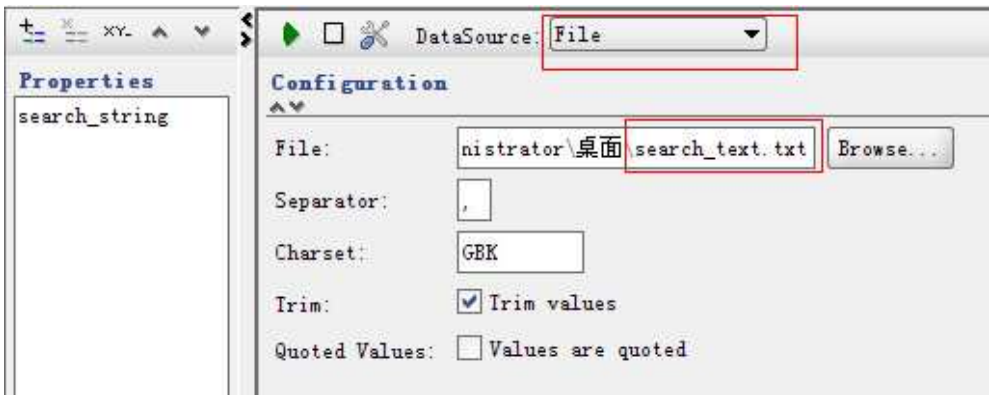
上面学习了 DataSource 中最简单的 Grid 类型，下面接着学习 File /Excel / Dataconnection

其实我们只有简单改变一下步骤 2 中 DataSource 的配置，根据类型的不同进行相应的设置就可以了

File 类型

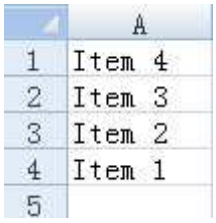
准备格式如下的文件（参数必须一行一行写，我尝试用，隔开不起作用）

在配置项中选择相应的文件，运行测试一下，一般没什么问题



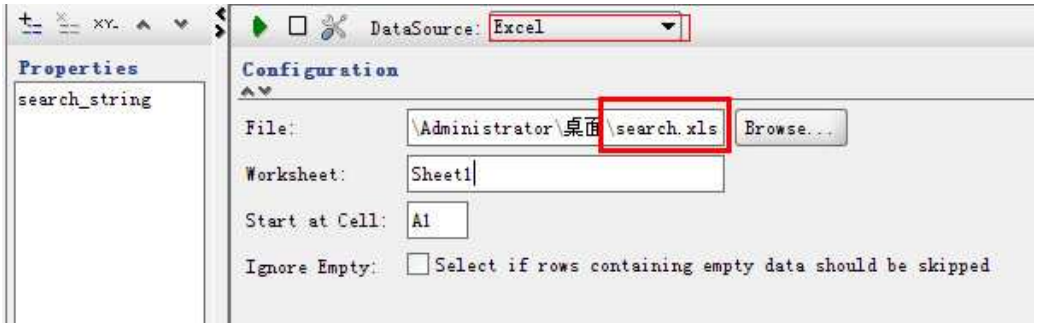
Excel 类型

准备 excel 文件，注意保存为 2003 格式，2007 格式似乎不能读取



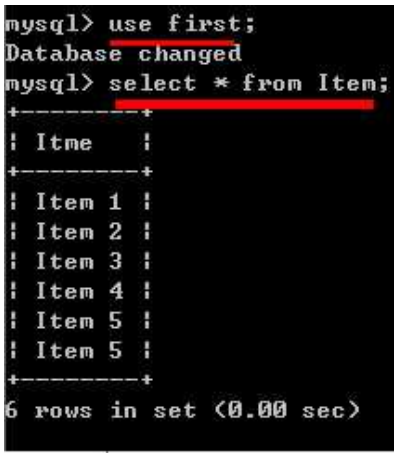
	A
1	Item 4
2	Item 3
3	Item 2
4	Item 1
5	

配置一下



DataConnection 类型 （我现在的是 mysql 连接）

首先在 mysql 数据库中准备数据

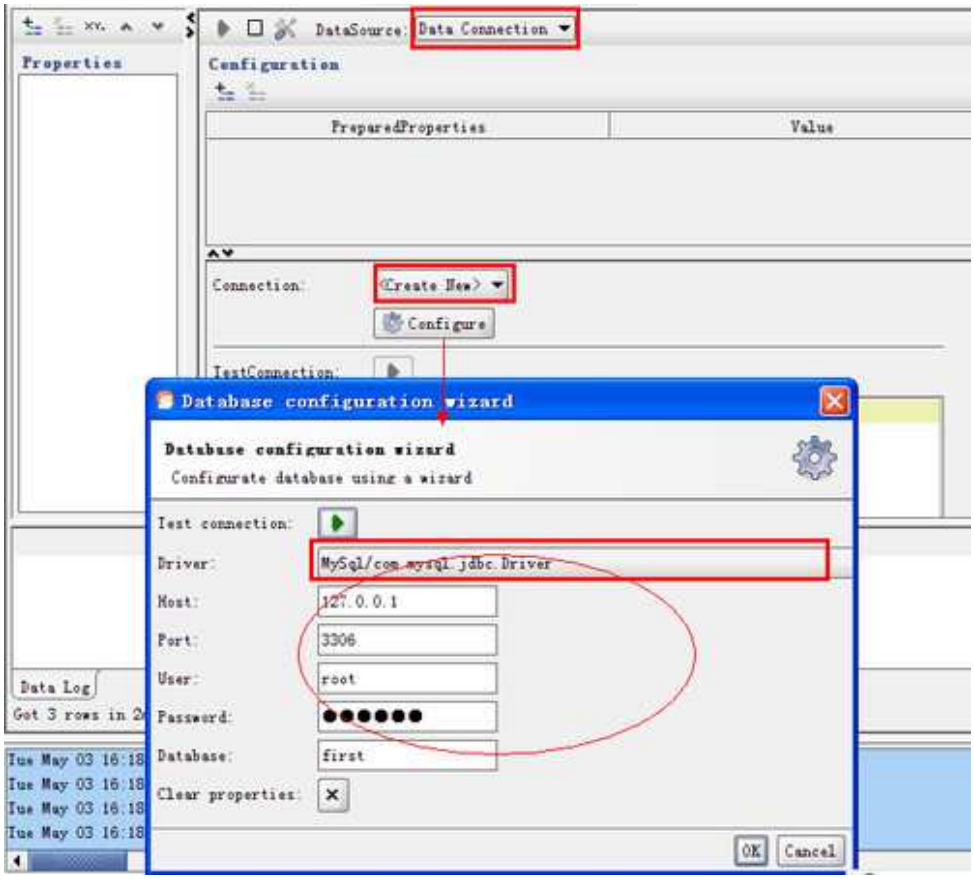


```
mysql> use first;
Database changed
mysql> select * from Item;
+---------+
| Itme    |
+---------+
| Item 1  |
| Item 2  |
| Item 3  |
| Item 4  |
| Item 5  |
| Item 5  |
+---------+
6 rows in set (0.00 sec)
```

配置 data connection 正确填写各项，并利用 Test connection 测试连接是否成功

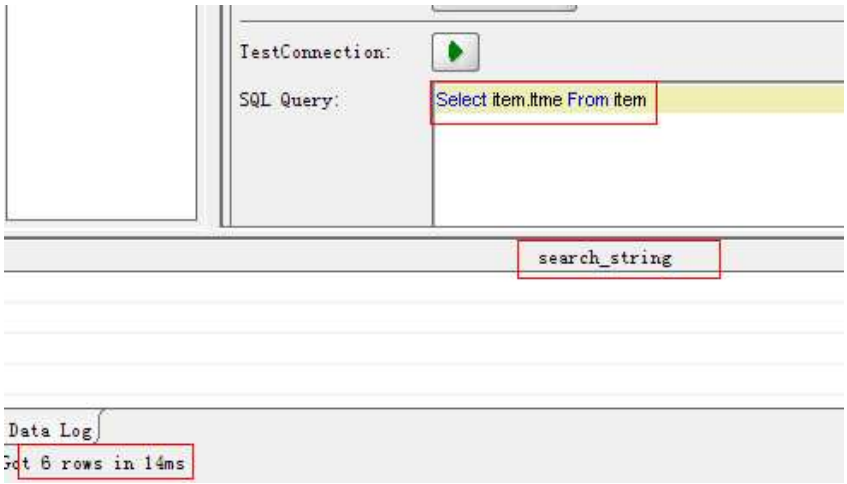
注意的是，在配置之前，检查 soapUI 安装目录下的 \jre\lib\ext 中是否有 mysql jdbc 的 driver，如果没有必须下载放进行，否则不能连接成功

我从 CSDN 中下载的 mysql-connector-java.jar



配置好连接后，有相应的 sql 查询语句配置页面，也可以自己写，我这些的是：select * from item;

执行一下，看看能不能查询出结果 （data log 中显示出来了，但具体的值没显示出来，不知道为何，但确实可用）



第十一节、不是总结的总结

有关 soapUI 学习的总结，我想就写到这里了吧。虽然我只学习了 soapUI 的皮毛而已。

还有很多没有学习，例如

- Datasink 测试数据的存储
- Report 测试结果
- 自动测试 与 junit/maven 联合
- soapUI eclipse 的插件
- 还有能发挥 soapUI 更强大功能的 groovy 脚本
- 等等

因为没有直接测试过一个 webservice 的项目，也不知道具体这些学习在实践中怎么运用，我想有了之前学习的有关 soapUI 的基础和了解，如果遇到 webservice 的测试，再来学习研究应该还是可以很快上手的吧。

有关 soapUI 的学习，自己觉得值得提倡的是将起源头作为学习的开始，在学习前了解 xml ，webservie，然后再来学习 soapUI，这样的学习过程是值得以后使用的。

同时也能感觉到，很多知识都是相关的，比如在 soapUI 中就看到了 xpath，junit，maven 等等。知识不会嫌多啊！

==========

从 3 月中旬开始做学习总结现在，越总结越越得有必要，并且有时不我待的感觉。如不再不总结，可能都忘记很多的东西。

总结的时候，也会去书店看看书，书店的书一般来说要比网络上看的有感觉，有深度，容易扩散思维，但很快又能收回来了，记录下来，回来网络上查查。写出的应 该都是我们这个行业的高手，他们关注的我也应该去关注。有时候很为自己惭愧，他们在几年前看的东西，我现在看了他们的书才知道，多向他们学习，站在巨人的 肩膀上，看的更远。

也能感觉出有时总结有点形式化，觉得是在写博客，把写博客看得比总结更重要了。另外就是总结没有连续起来，一会总结 mysql，一会总结其他的了。另外， 图片比较多，我就怕像我们看其他人的博客一下，写的时候图片能显示，但过了一段时间图片就看不到了。这也是我没有换 163 博客的原因，我相信 163 作为大 的门户网站，图片应该在过很久也能显示出来。

流口水的小猪（163 博客）