

Parasoft SOAtest 入门手册

最后更新时间: June 12, 2014

目录

第 1 章：SOATEST 介绍.....	4
核心能力.....	4
用户界面.....	6
术语表.....	12
中英文翻译对照表.....	12
第 2 章：基础 WEB 服务的单元测试.....	13
第 1 课：自动创建测试套件.....	14
第 2 课：理解 WSDL 测试.....	19
第 3 课：理解自动生成的测试.....	21
第 4 课：查看 HTTP 通信消息.....	22
第 5 课：理解测试成功和失败.....	25
第 6 课：添加更多测试，删除不需要的测试.....	26
第 7 课：嵌套测试套件.....	27
练习 1：创建 SOATEST 项目用于培训练习.....	30
第 3 章：自动 WEB 服务验证和管理测试数据.....	31
第 1 课：通过在整个响应上自动地创建回归控制来配置 WEB SERVICE 验证.....	31
第 2 课：在验证 WEB SERVICES 响应时忽略 XML 消息中动态或不适当的值.....	36
第 3 课：使用 XML 断言器验证 WEB 服务响应消息中的指定元素.....	38
第 4 课：使用数据源来驱动测试和场景.....	41
练习 1：BOOKSTORE 测试套件创建.....	45
练习 2：CUSTOMER #1 测试套件创建.....	45
练习 3：数据源和多个回归.....	45
第 4 章：WEB 服务场景测试.....	46
第 1 课：使用 XML TRANSFORMER 工具来对 XML 消息的一部分进行回归控制.....	46
第 2 课：使用 XML DATA BANK 工具将值从一个测试传递到另一个测试.....	50
第 3 课：测试套件逻辑——管理测试的依赖关系.....	55
第 4 课：使用自定义脚本扩展 SOATEST 的功能.....	56
练习 1：创建 CUSTOMER #2 包含 XML DATA BANK.....	59
练习 2：创建 CUSTOMER #3 包含 XML DATA BANK.....	59
练习 3：使用 CALCULATOR 服务的另外一个场景.....	59
第 5 章：WEB 服务测试相关内容.....	61
第 1 课：验证 SOAP MESSAGES 是否遵从 WSDL 使用的 SCHEMAS.....	61
第 2 课：测试 SOAP MESSAGE 对 WS-I BASIC PROFILE 的符合性.....	61
第 3 课：从 SOATEST 界面生成 HTML 报告.....	63
第 6 章：端到端测试.....	64
准备工作：设置 DEMO 应用 (PARABANK).....	64
第 1 课：测试数据库.....	65
练习 1：拓展数据库测试.....	67
第 2 课：测试 RESTFUL 服务.....	68
第 3 课：验证 WEB 页面.....	70
练习 2：拓展 RESTFUL 测试.....	70

练习 3: 创建一个端到端测试场景.....	70
第 7 章: 压力测试.....	71
典型工作流.....	71
第 1 课: 创建一个 WEB 应用功能性测试.....	72
第 2 课: 创建一个服务功能性测试.....	74
第 3 课: 创建并执行一个压力测试 (为 WEB/SERVICE 功能性测试).....	75
第 4 课: 自定义压力测试配置文件和场景.....	79
第 5 课: 创建自定义压力测试组件.....	81
第 6 课: 查看报告.....	89
第 7 课: 保存项目为 .LT 文件.....	89

第 1 章：SOAtest 介绍

Parasoft SOAtest 是一个完整生命周期的质量平台，为安全性、可靠性、符合标准的业务流程提供了一个持续的质量管理过程。它提供一个企业级的解决方案：

- **质量管控**：持续测量每个服务如何符合通常的来自自己的组织和合作伙伴定义的动态期望。
- **环境管理**：在现今的异构环境下降低测试的复杂性——分布式组件或供应商特定技术的有限可见/控制。
- **端到端的测试**：持续验证所有复杂事务的关键部分，包括可能延伸到的 web 接口，后端服务，ESB，数据库和任何中间事物。
- **过程可见和控制**：建立一个可持续的工作流程，帮助整个团队在完整生命周期内有效的开发，共享，和管理质量资产演变。

核心能力

质量管控	
设计和开发政策强制	确保在跨分布式系统中从应用程序代码语句到业务流程的互操作性，安全性和一致性。自动和持续地实行业标准和自定义的政策。
缺陷预防	指导开发人员避免常见的安全性和互操作性问题，降低下游测试和调试的时间和成本。
基于注册的政策管理	自动测试注册的服务并验证其是否遵守注册中心所定义的政策。最终结果报告给注册中心（并实时更新）能够在整个生命周期中为服务的质量和一致性提供持续地可见性。
环境管理	
应用程序行为虚拟化	自动地模拟服务的行为，然后在跨多个环境中部署他们——简化协作开发和测试活动。可以从功能测试或实际运行环境的数据中模拟服务。
平台认识	促进任何基础设施组件的互操作性和可见性——降低用于定义复杂测试和了解它们是如何在异构系统之间传递的学习曲线。提供对主流行业平台即时可用的支持，包括： <ul style="list-style-type: none"> • AmberPoint • HP • IBM • Microsoft • Oracle/BEA • Progress Sonic • Software AG/webMethods • TIBCO
端到端测试：	
SOA 相关的测试构建	先进的自动化测试和 SOA-Aware 使得可扩展测试的构建变得快速。自动从工件生成测试，诸如 WSDL, WADL, UDDI, WSIL, XML Schema, BPEL, HTTP 流量, 和关键行业平台 (参见第 2 页中的列表)。
端到端的场景验证	促进快速、增量开发的测试套件，验证端到端的操作。这可能跨 ESB 的消息传递层，Web 界面，数据库，和 EJB。这确保底层实现的可靠性。
高级 web 应用测试	引导团队开发健壮的、无噪声的回归测试和丰富的高度动态的基于浏览器的应用程序。

多层业务流程验证	验证业务流程持续满足异构系统跨多层次的期望。当修改对关键流程产生影响的时候发出警报—提供一个安全网络以降低更改的风险并实现对业务需求快速和敏捷的响应。
ESB事件监控与验证	可视化和跟踪测试所触发的内部流程事件，促进直接从测试环境中对问题的快速诊断。还持续验证是否关键事件随着系统的发展继续满足功能预期。
压力/性能测试	验证应用程序在高负荷下的性能和功能。现有的端到端的功能测试用于压力测试，使得全面测试和持续性能监控变得容易。
安全性测试	预防安全漏洞通过渗透测试和执行复杂的身份验证、加密和访问控制的测试场景。
过程可见和控制：	
流线型协同 workflow	实现SOA质量是一个协同成就，涉及许多机构内的不同参与者，从服务和Web界面开发人员、到QA、到业务分析师。通过建立一个可持续的工作流程自动生成，分配，分发质量任务给合适的团队成员。该解决方案使得整个团队更有效率。

用户界面

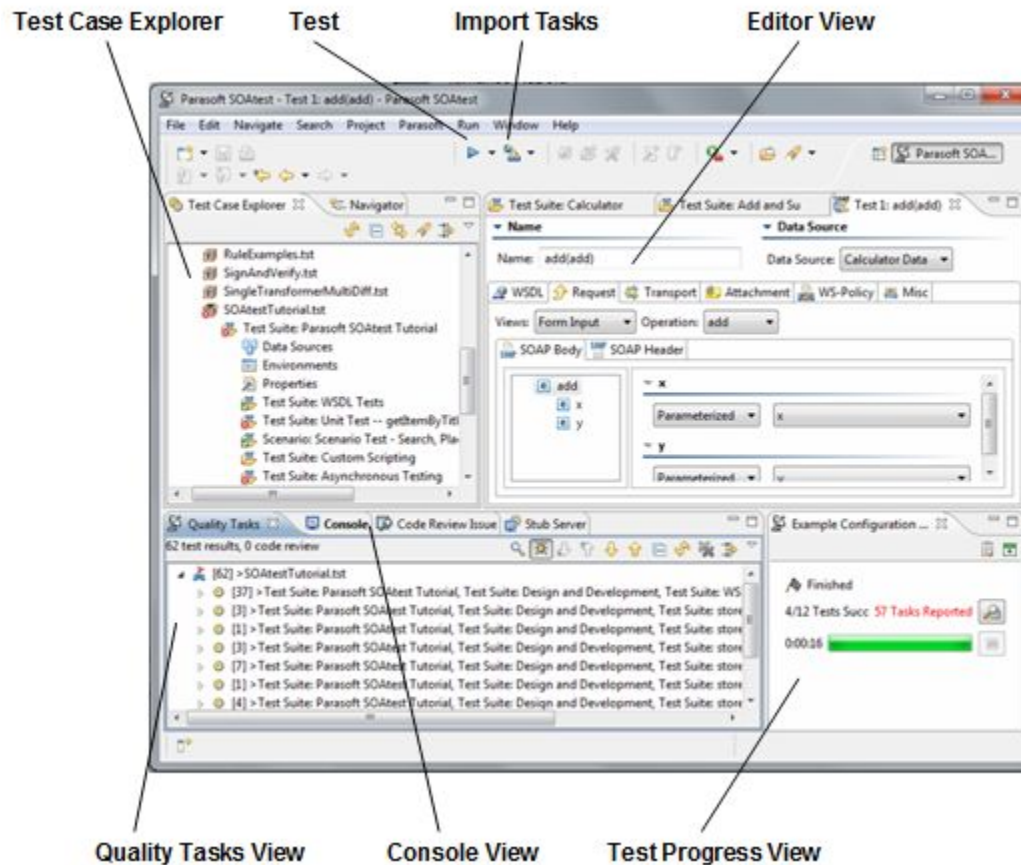
SOAtest 透视图

SOAtest透视图已内建到Eclipse工作台，提供一组设计的功能来帮助你配置，运行，审查测试。

你可以通过以下任意一种方式打开透视图：

- 在快捷工具栏中单击**SOAtest透视图**按钮(在工作台的右上角)。
- 在快捷工具栏中单击**打开透视图**按钮，选择**其它**，然后在打开的透视图选择对话框中选择**SOAtest**。
- 选择菜单**Window>打开透视图>其它**，然后在打开的透视图选择对话框中选择**SOAtest**。

SOAtest透视图提供特定的视图、工具栏按钮和菜单项，用于配置、执行和审查测试。



视图

SOAtest功能依赖以下视图：

版权 - Parasoft 公司 2014

- **测试用例浏览器：**测试用例浏览器显示可用的项目和测试用例。测试用例浏览器能同时打开多个项目。每个项目又可以同时打开多个测试用例。
- **质量任务视图：**SOAtest 在质量任务视图中列出测试结果。这个视图默认是打开的。如果它不可见，选择 **Parasoft> Show View> SOAtest** 打开它。
- **控制台视图：**控制台显示所有执行测试有关的信息，包括多少个测试被执行，多少个测试失败，多少个测试跳过。
- **测试过程视图：**这个视图是 SOAtest 报告测试执行过程和状态的地方。
- **编辑器视图：**编辑器视图是工作台最大的面板。是 SOAtest 显示工具/测试配置面板或源代码的地方，取决于选择什么测试用例或导航节点。例如，如果你在测试用例浏览器中双击一个 SOAP Client 工具节点，会在编辑器中打开一个 SOAP Client 工具配置面板。

工具栏按钮

SOAtest 添加了如下按钮到工具栏：

- **测试：**测试按钮允许你快速执行任何可用的测试配置。如果简单的单击测试按钮，SOAtest 将基于偏爱的测试配置执行测试。如果你使用测试按钮内右边的下拉菜单，你可以使用任何有效的测试配置进行测试。
- **导入我的推荐任务：**导入我的推荐任务按钮允许你从 Parasoft Team Server 导入所选类别的可用的结果。这使你可以通过 GUI 来审查和分析来自命令行测试的结果。如果你简单的单击我的推荐任务按钮，SOAtest 将导入所有你的测试任务的一个子集
 - 1、你负责(基于 SOAtest 的任务分配)
 - 2、用 SOAtest 审查和定位(基于你的团队配置 SOAtest 每天每个小组成员报告的最大任务数量)。如果你使用导入我的推荐任务按钮右侧的下拉菜单，你可以选择你想要导入结果的类型。

Parasoft 菜单命令

Parasoft 菜单提供如下命令：

- **测试执行[最喜爱的配置]：**启动一个当前设定的收藏的测试配置开始测试。
- **测试历史：**启动一个选择的测试配置开始测试。这里仅列出最近执行的测试配置。
- **测试执行：**启动一个选择的测试配置开始测试。这里列出所有可用的测试配置。
- **测试配置：**打开测试配置对话框，你可以在这里查看，修改，创建测试配置。
- **启动RuleWizard：**打开RuleWizard，一个图形化自动创建静态分析自定义规则的工具。
- **浏览> 团队服务器：**打开团队服务器浏览器对话框，在这里可以访问，配置，更新测试配置，规则，规则映射文件，报告。
- **浏览> 团队服务器报告：**打开团队服务器上可用的HTML报告文件。
- **浏览> 报告中心报告：**打开报告中心报告，来源于SOAtest测试和其他源。
- **导入：**导入所择种类的团队服务器上可用的结果。
- **查看视图：**打开可用视图。
- **首选项：**打开首选项对话框。
- **支持：**提供多种方式联系技术支持团队。
- **帮助：**在在线帮助系统中打开用户手册。
- **取消|激活许可：**取消/激活一个SOAtest LicenseServer许可。

Scanning 透视图

Scanning透视图设计用于促进对静态分析扫描资源的审查和重新测试。

打开Scanning透视图：

- 选择**Window> Open Perspective> Other> Scanning**。

这个透视图类似于S0Atest透视图，但有两个附加的功能：

- 测试单一URL或文件的Quick Test工具栏按钮。这个按钮能被加到任意透视图，通过选择**Window> Customize Perspective> Commands**并单击挨着S0Atest Scanning的复选框。
- Scanned Resources视图。这个视图能被加到任意透视图，通过选择**Window> Show View> Parasoft> Scanned Resources Listing**。

Load Test 透视图

Load Test透视图设计用于帮助你准备web功能测试进行压力测试。

打开Load Test透视图：

- 选择**Window> Open Perspective> Other> Load Test**。

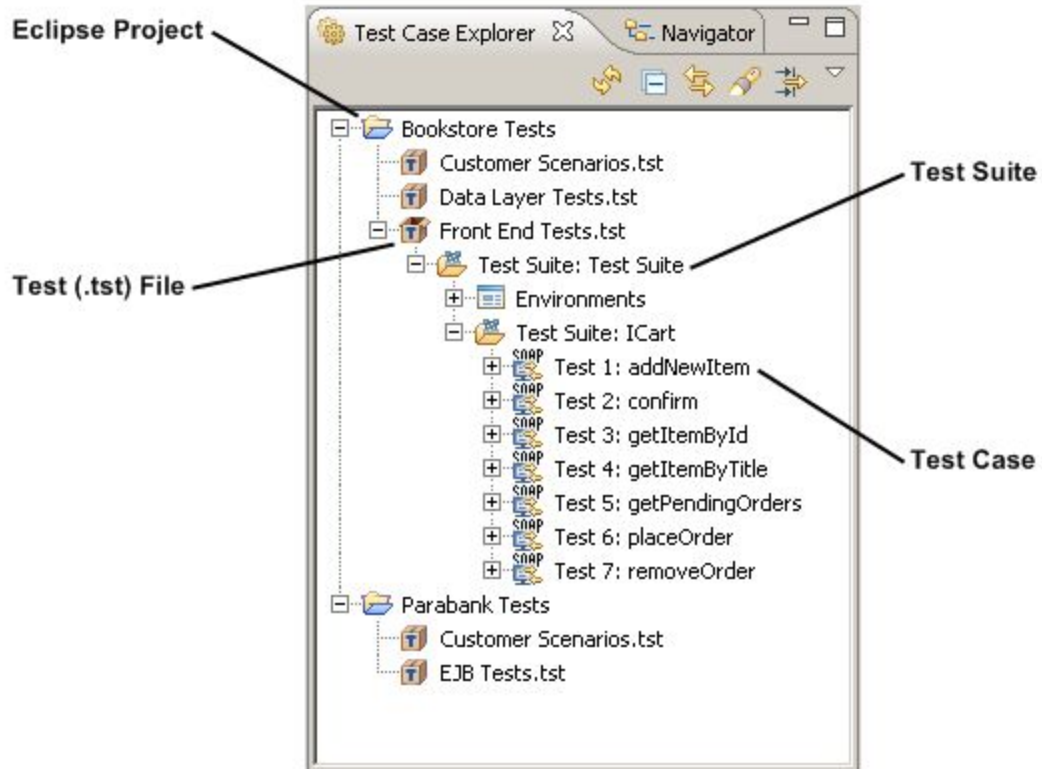
这个透视图类似于S0Atest透视图，但有如下附加的功能：

- 两个工具栏按钮(进行压力测试配置和进行压力测试验证)，允许你运行自动测试配置和验证。
- Load Test Explorer，列出了可用的web功能测试。注意，任何跟压力测试无关的web功能测试组件将不被显示在这个视图中——例如，基于浏览器的验证和data banks。
- Load Test Explorer右击菜单运行自动测试配置和验证（同工具栏按钮）。
- 详细测试配置面板，通过在Load Test Explorer中双击测试访问。

工程和测试文件

S0Atest基于Eclipse开发环境构建。Eclipse使用“projects”作为它的基础组织单元。S0Atest使用测试文件（以.tst作为文件扩展名），作为它的基础组织单元。一个Eclipse项目能包含多个.tst文件。它们能包含你想要分析的源文件和任何其它对你的环境有用的资源。

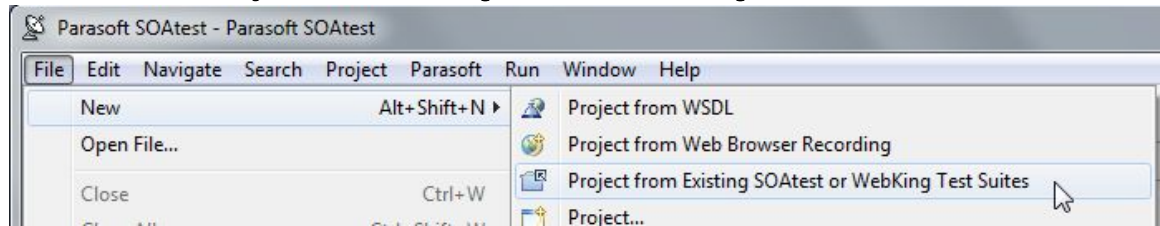
每个S0Atest测试文件(.tst)能包含多个测试套件/场景，工具，输入，和桩。你可决定如何组织它们。保持文件大小不要过大来提高可维护性，我们建议为每个不同的测试需求使用一个测试文件。



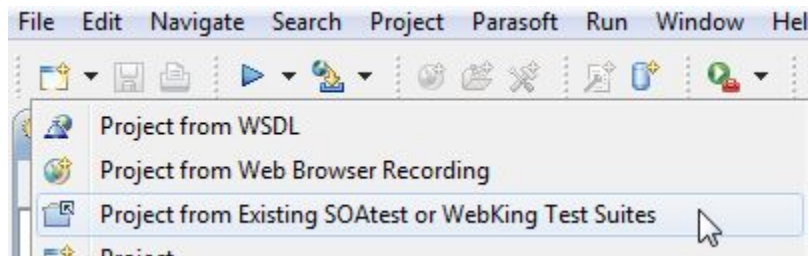
创建新工程

多个 .tst 文件可以组织在一个工程内。首先我们基于一个存在的测试套件创建一个新工程。

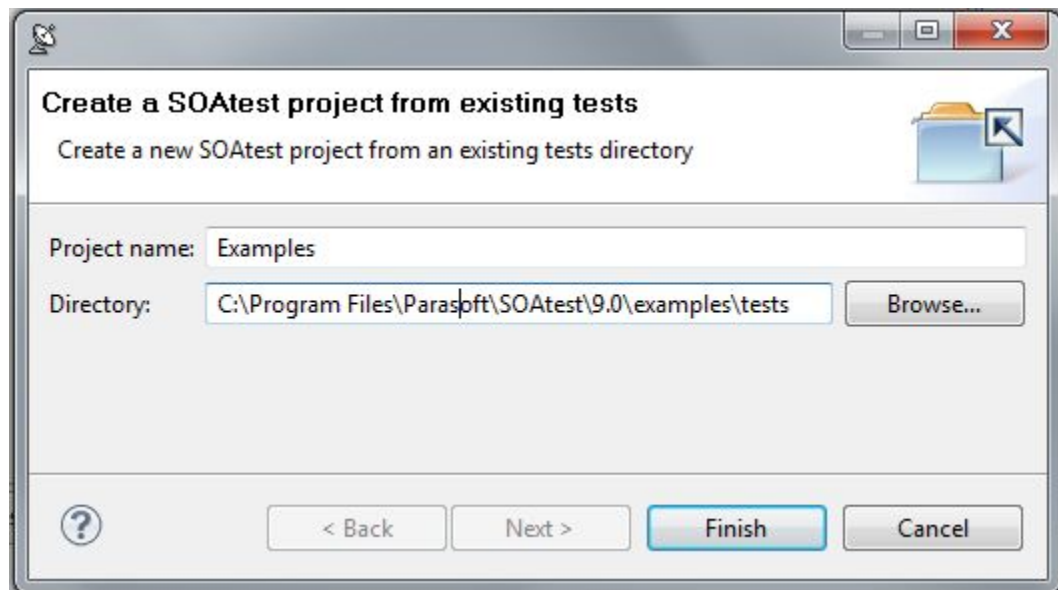
1. 选择 **File > New > Project from Existing SOAtest or WebKing Test Suites**。



或者，从**新建**工具栏按钮（左上角）选择下拉菜单命令。



2. 在 **Project Name** 字段中输入 Examples。
3. 指定项目的测试套件路径，通过单击 Browse，然后导航到 “[SOAtest_installation_directory]/examples/tests”。
4. 单击 **Finish**。



示例项目将被添加到 Test Case Explorer。它将包含多个测试文件 (.tst)。

你也可以通过新建项目向导选择不同的命令创建新项目 (例如 Project from WSDL, Project from Web Browser Recording)。选择 **File > New > Other** 并转到 **SOAtest** 文件夹，来查看所有可用的新建项目选项。

打开和关闭测试（.tst）文件

默认，.tst文件是关闭的。所有打开的.tst文件会被加载到内存中。

有两种方式打开一个.tst文件：

- 双击.tst文件对应的Test Case Explorer节点。
- 右击.tst文件对应的Test Case Explorer节点，然后从弹出的快捷菜单选择**Open Test (.tst) File**。

关闭的.tst文件显示“closed box”图标：



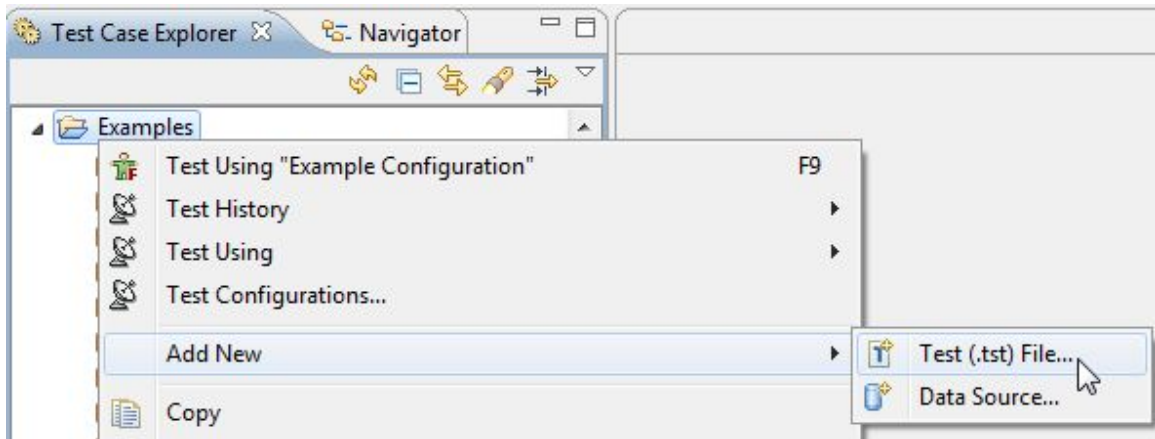
打开的.tst文件显示“open box”图标：



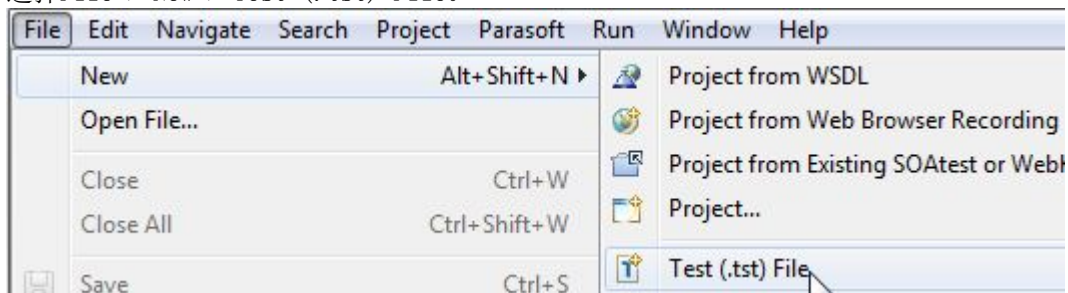
创建新的测试（.tst）文件

我们推荐你为每一个独立的测试需要创建一个测试文件（.tst file）。有两种方式来创建一个新的测试（.tst）文件。

1. 右击项目节点，然后从弹出的快捷菜单选择Add New> Test（.tst）File。



2. 选择File > New > Test（.tst）File。



向导将会引导你完成测试用例的创建过程，然后会添加一个包含生成的测试用例的.tst文件。

当你完成下面的章节，你将学会如何创建不同类型的测试套件。

术语表

中英文翻译对照表

Web Service: Web服务

Traffic: 通信信息, 通信消息, 通信报文

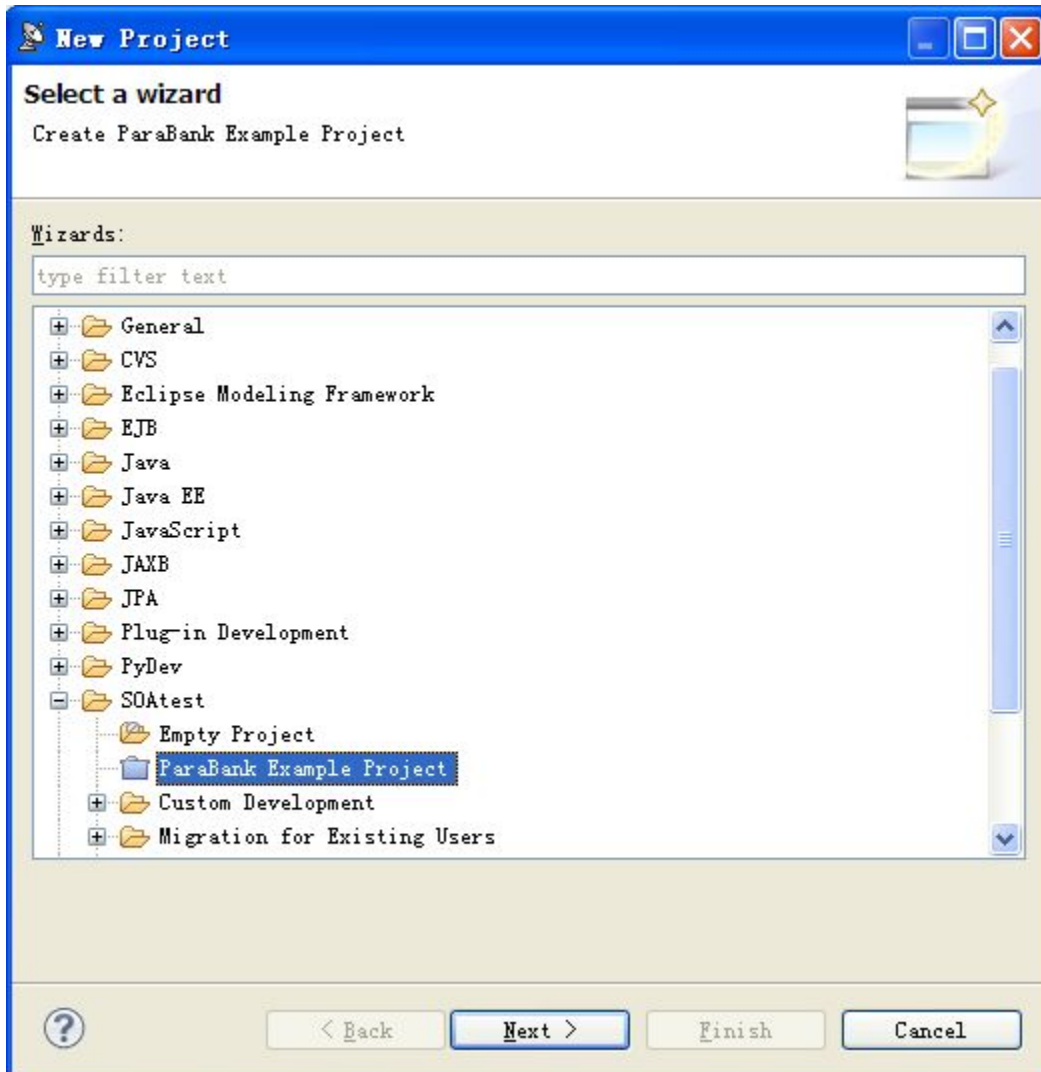
第 2 章：基础 Web 服务的单元测试

在这章中你将学会使用SOAtest进行基础的Web Service测试和创建一个完整的基于WSDL的功能测试套件。我们还将向您介绍SOAP客户端的GUI，HTTP流量和软件的整体轮廓。

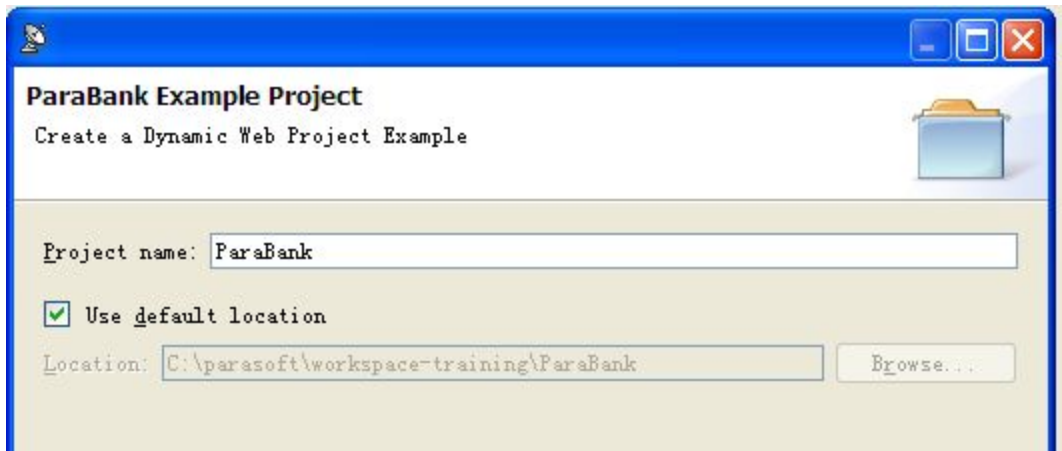
环境准备

创建ParaBank

打开SOAtest, 选择File>New>Project, 选择SOAtest>ParaBank Example Project



点击Next，选择项目名称，然后单击Finish。



然后我们就能看到ParaBank项目载入的进度框，以及Server的创建和Tomcat的启动。

创建完成后，我们会在浏览器看到如下的ParaBank运行起来的界面：



这说明我们的环境已准备完成，ParaBank项目已成功创建出来。

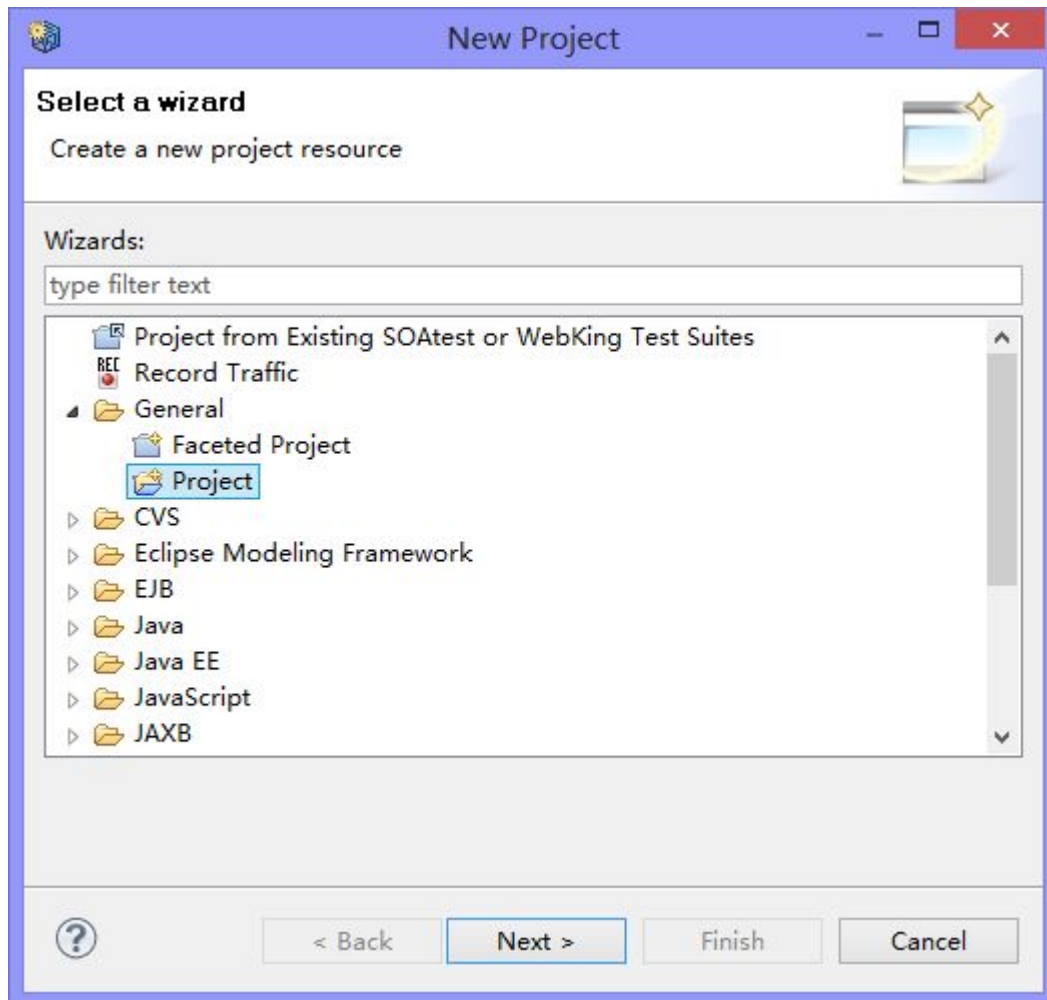
Web Service “http://localhost:18080/parabank/services/store-01?wsdl”

第 1 课：自动创建测试套件

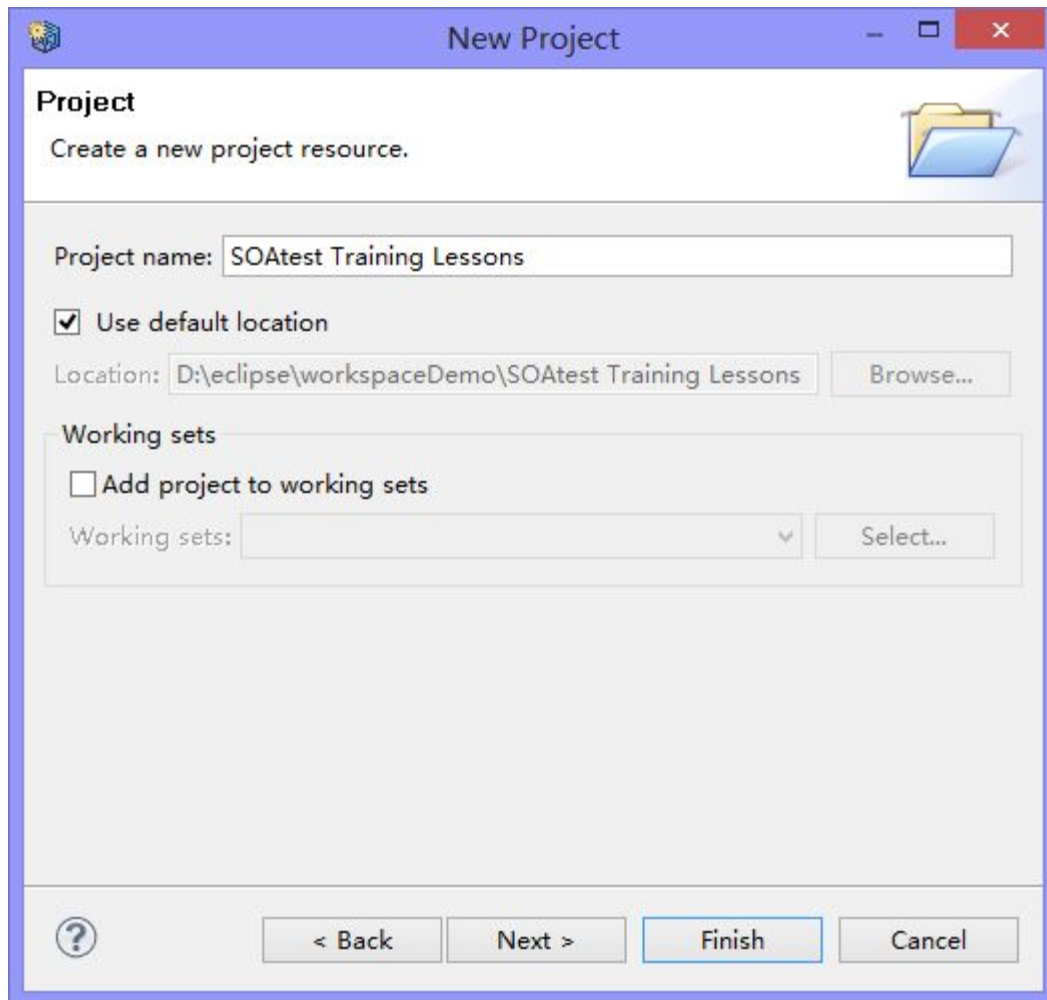
SOAtest能通过一份WSDL文档自动创建测试套件。如果WSDL中有多个业务操作，SOAtest将为每个操作创建一个测试用例，并提供一种方式来记录通信报文和运行测试。这些测试用例提供基础的功能和回归测试。

从WSDL自动生成测试用例套件:

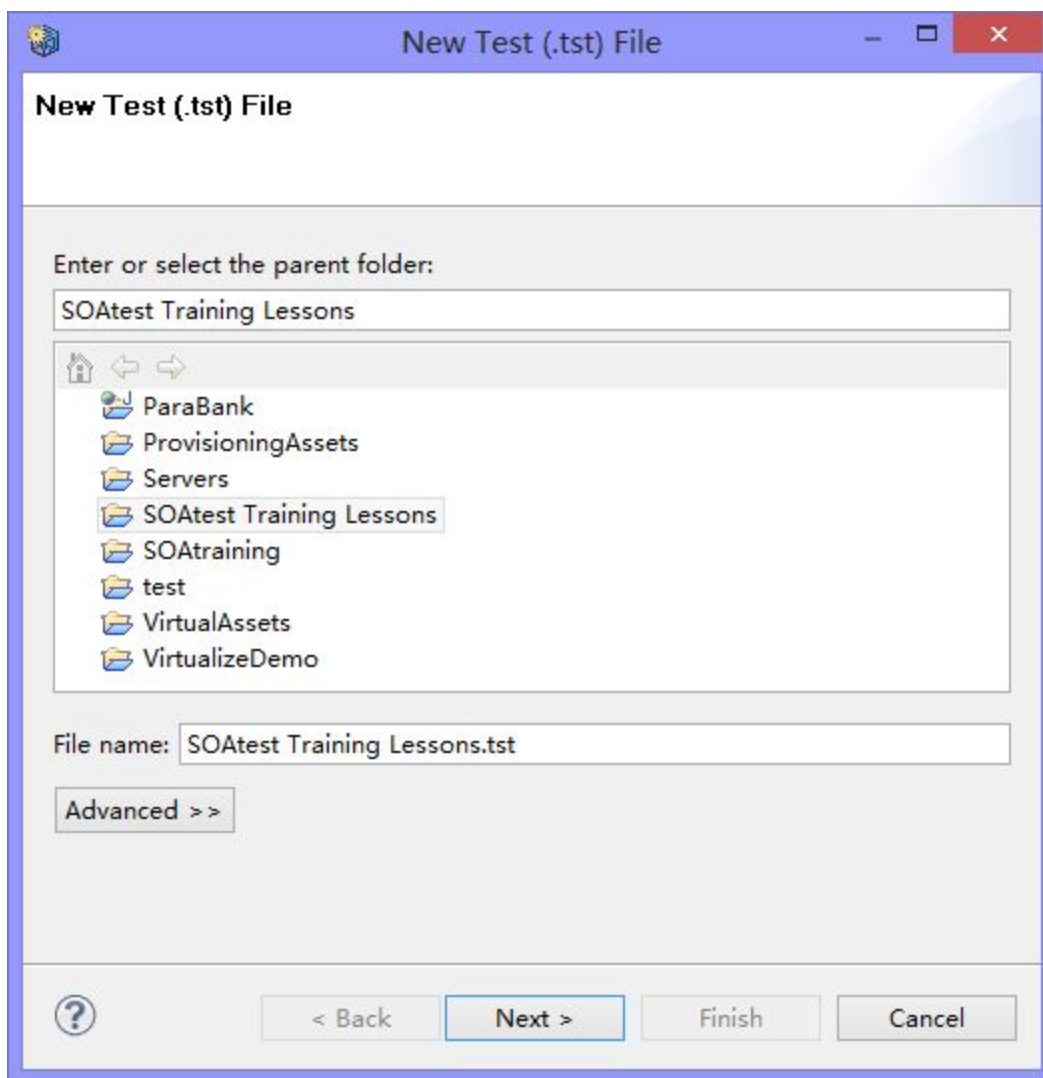
1. 从**新建**工具栏按钮 (左上角) 打开下拉菜单, 然后选择**Project...**。
2. 选择**General>Project**, 点击**下一步**。



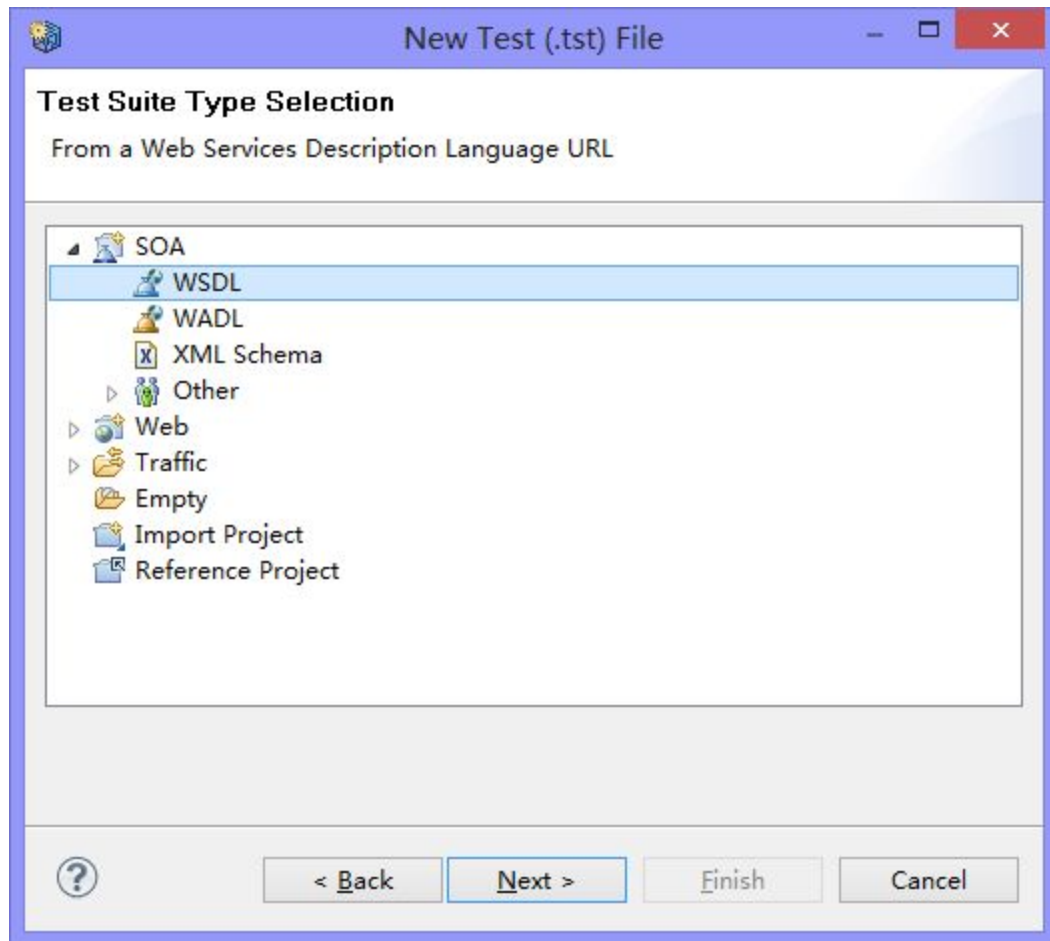
3. 在**Project name**字段中输入**SOAtest Training Lessons**。然后单击按钮**Finish**。



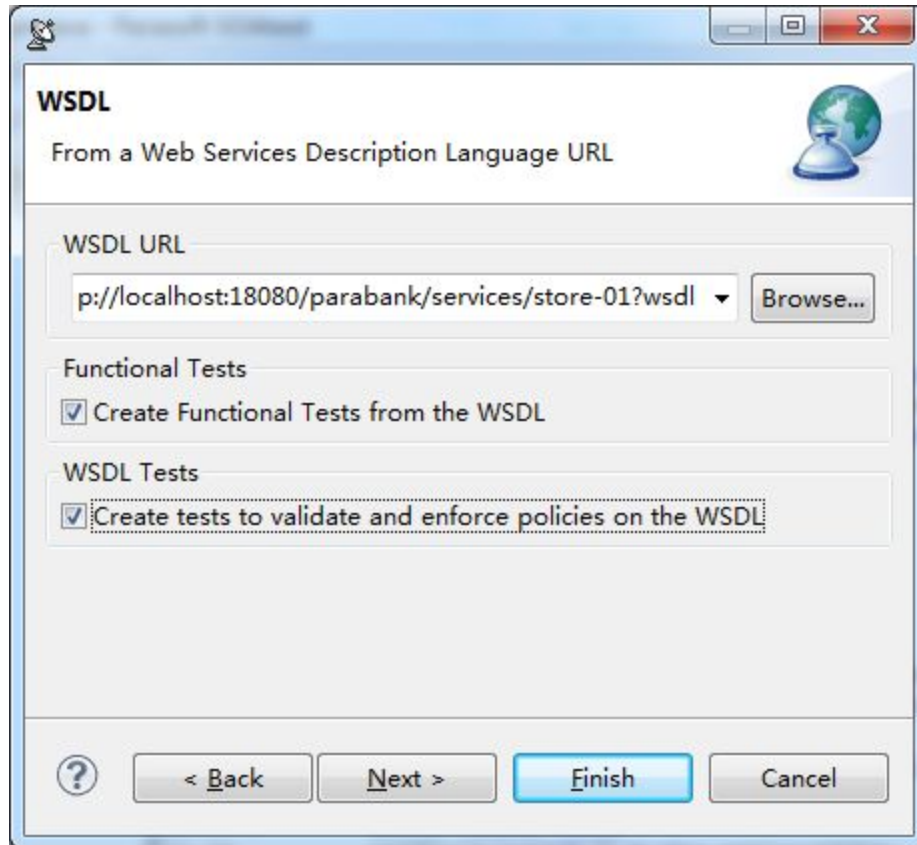
4. 从新建工具栏按钮（左上角）打开下拉菜单，然后选择Test(.tst)File。
5. 选择项目SOAtest Training Lessons，在文件名框中输入SOAtest Training Lessons.tst，点击下一步。



6. 选择SOA>WSDL，点击下一步。



7. 在WSDL URL字段中输入<http://localhost:18080/parabank/services/store-01?wsdl>。



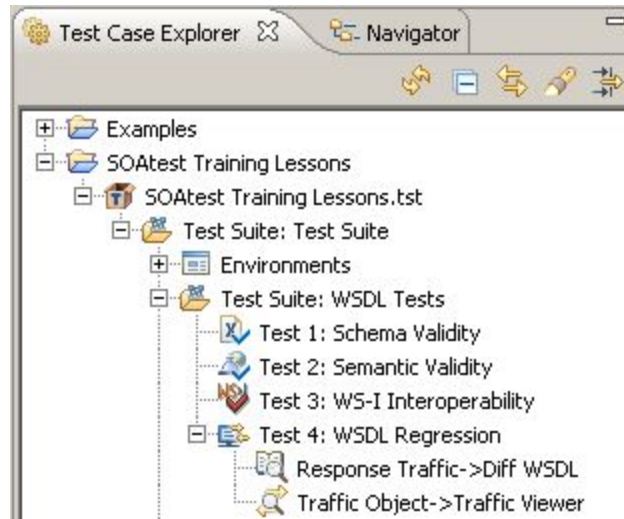
8. 勾选Create Functional Tests from the WSDL多选框，同时勾选Create tests to validate and enforce policies on the WSDL多选框。
9. 单击Finish。

第2课：理解 WSDL 测试

因为你选择了Create tests to validate and enforce policies on the WSDL多选框，在一个独立的测试套件WSDL Tests中，4个WSDL测试将被自动创建。

WSDL验证是测试web services的第一步。虽然各种工具可以自动生成WSDL，但是不能保证他们一定是正确的。当WSDL被手动改变，WSDL验证变得更加重要。SOAtest能自动地生成全面的WSDL测试套件，来确保你的WSDL符合schema并验证测试传递的XML。另外，它还执行互操作性检查来验证你的web service是否具备与其它符合WS-I服务的互操作性。

1. 看到WSDL验证测试套件，打开Test Case Explorer视图，展开WSDL Tests树。



SOAtest从WSDL URL自动创建如下WSDL测试:

Test 1: Schema Validity: 针对来自W3C的WSDL scheme, 对WSDL执行XML验证。

Test 2: Semantic Validity: 检查WSDL的正确性, 通过像一个实际的服务消费者一样分析和消费它, 但更严格地遵守标准。

Test 3: WS-I Interoperability: 针对WS-I Basic Profile 1.1, 验证WSDL。

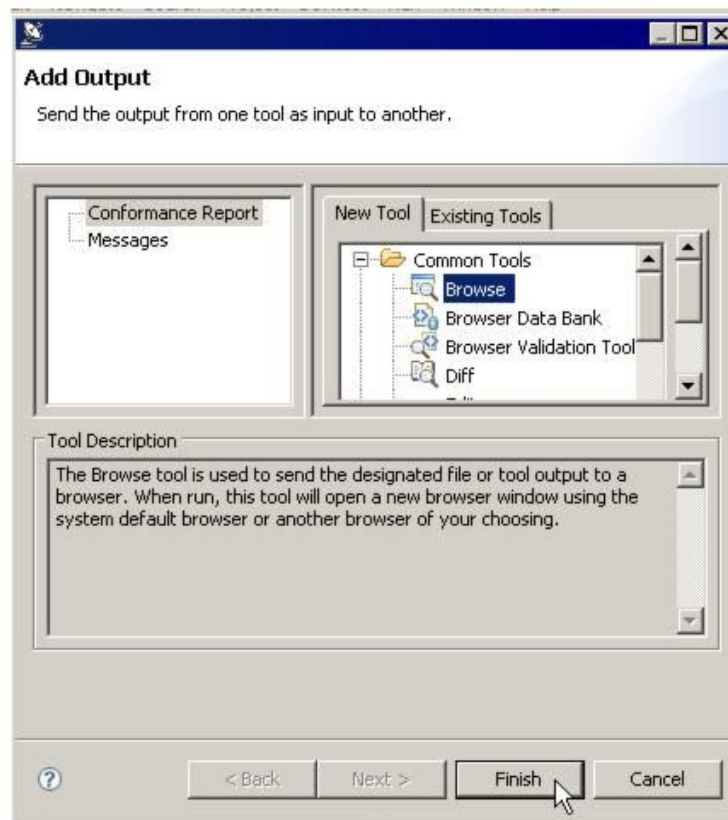
Test 4: WSDL Regression: 为WSDL创建回归测试, 以便WSDL改变时能被检测到。

2. 选择节点**Test 3: WS-I Interoperability Check**并单击工具栏按钮**Add test or output...**。

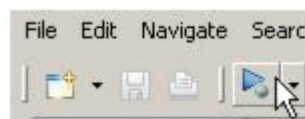


这将打开向导**Add Output**, 将显示可用的工具列表。另外, 选择的工具将在Tool Description字段显示描述信息。

3. 在Add Output向导中, 从左边面板选择Conformance Report, 从Show下拉菜单选择All, 从右边面板选择Browse, 然后单击Finish按钮。这将在你运行测试时发送WS-I符合报告到你的web浏览器。



4. 在Test Case Explorer中选择**Test Suite: WSDL Tests**节点并单击**Test**工具栏按钮。



如果有错误发生，他们将被显示在质量任务视图中——位于SOAtest界面的下方。你可以双击错误在界面右侧面板查看附加的信息。你还可以在打开的web浏览器中检查一致性报告。

第3课：理解自动生成的测试

在第1课，SOAtest已经为你的WSDL中定义的每一个业务操作创建了一个测试。这些测试可以移到单独的测试套件。通过每个测试用例创建一个测试套件，你可以组织你的测试环境来最大化可读性和可重用性。

使用SOAP Client工具添加测试用例。它是大多数测试用例的基础。通过在Test Case Explorer中双击相应的节点，你可以查看任何测试用例的细节。如果你双击SOAP Client测试节点，将在配置面板打开这个测试用例的细节。SOAtest决定WSDL输入参数的类型和结构。你可以按如下方式修改测试参数：

改变这个测试元素...	SOAP Client用户控制界面...
测试名称	名称
被测服务的WSDL URI	WSDL标签页


改变这个测试元素...	SOAP Client用户控制界面...
测试使用的消息协议	Misc标签页
SOAP消息的MIME类型	Misc标签页
会话超时	Misc标签页
SOAP消息发送和接收的URL	Transport标签页
测试说明	Misc标签页
传输协议	Transport标签页
SOAP动作 (服务如何处理请求)	Transport标签页
SOAP信封 (SOAP body和/或SOAP header) 类型	Request标签页 Literal XML 选项指定SOAP信封使用XML。 使用 Form XML 选项来配置XML消息，通过一个XML requests and responses的树视图。 使用 Scripted XML 选项从脚本来动态生成XML。 使用 Form Input 选项可以从界面文本框输入参数。
SOAP信封 (SOAP body和/或SOAP header) 细节	用户图形界面下方的控件（属于SOAP信封）。可用的控件会根据信封的类型变化。

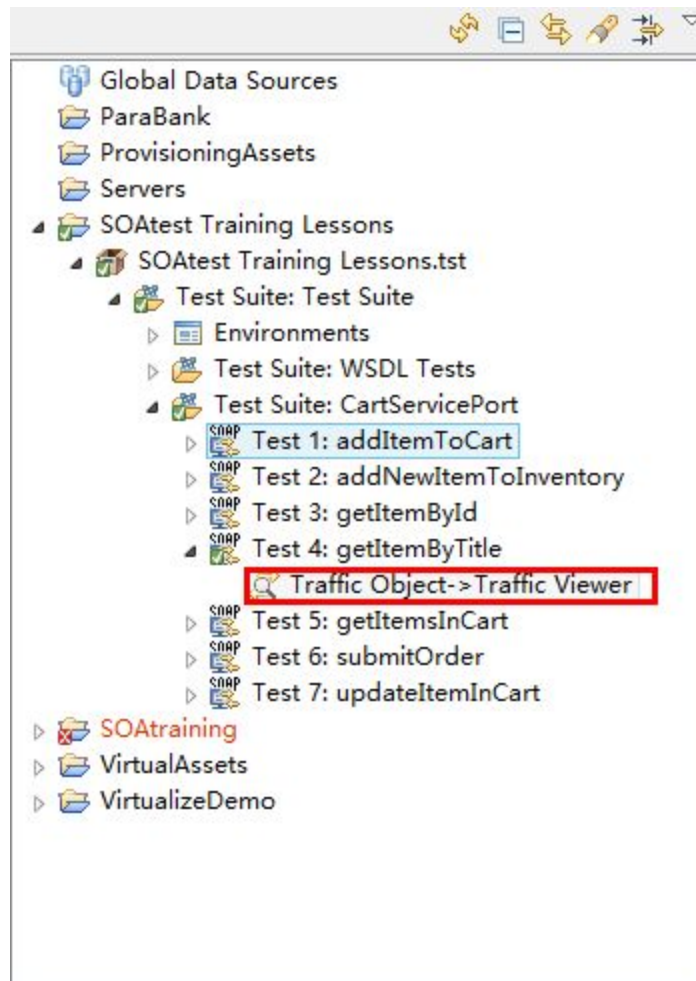
你知道吗？

如果你指定 WSDL，SOAP Client 将根据 WSDL 自动更新字段，诸如 **Router** 和 **Messaging Convention**。你可以手动编辑这些字段，如果 SOAP Client 的 **Constrain to WSDL** 没有勾选。

第 4 课：查看 HTTP 通信消息

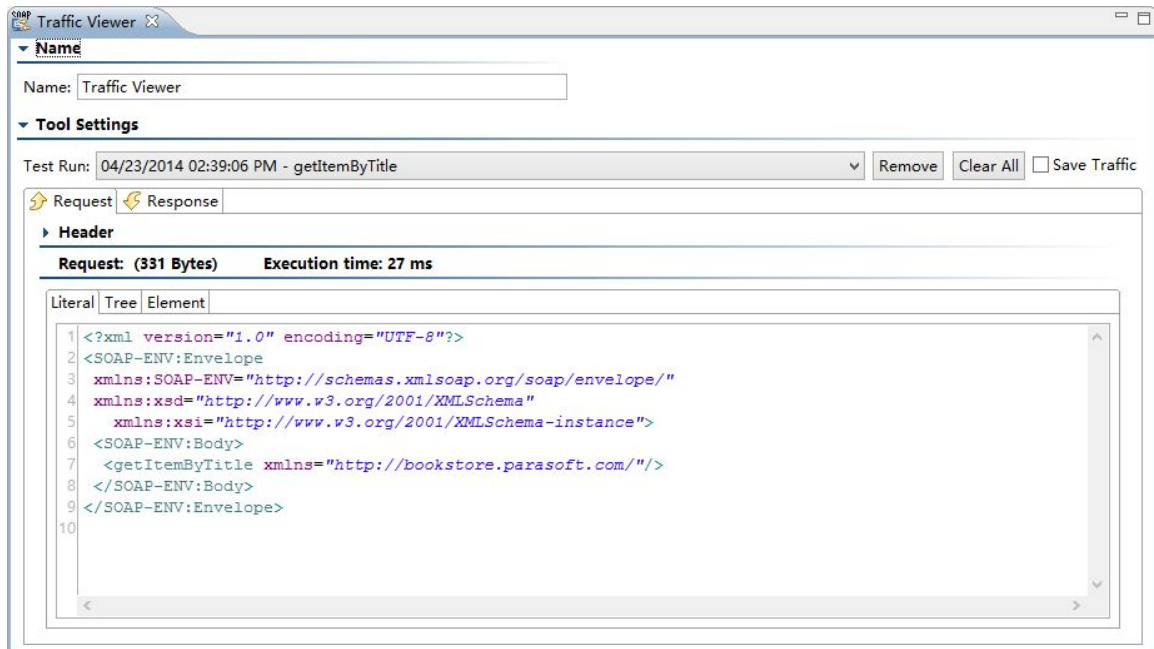
你可以记录并查看HTTP traffic (SOAP请求和SOAP响应)，通过单击**Traffic Object> Traffic Viewer**节点。你每次执行测试套件，SOAtest会为测试套件中的每一个测试自动记录客户端到服务器端的HTTP traffic。Traffic Viewer工具帮助你管理和可视化每一个测试用例的SOAP请求和SOAP响应。Traffic Viewer能记录多个对应每次测试执行的HTTP traffic实例。

1. 在Test Case Explorer中选择**Test Suite: CarServicePort>Test 4:getItemByTitle**。
2. 单击工具栏按钮**Test**。 
3. 双击Traffic Viewer打开。



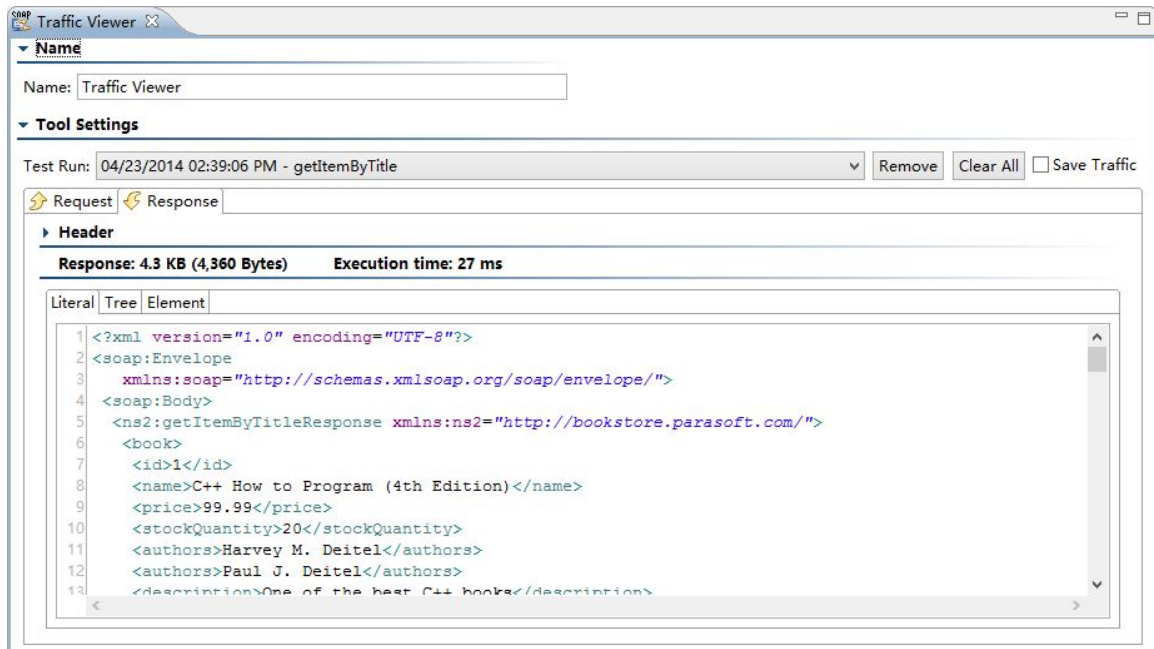
你可以为Traffic Viewer工具配置如下选项：

- **名称：**指定Traffic Viewer工具的名称。
- **测试执行：**显示选择的测试用例的时间戳。
- **移除：**单击可以删除从Test Run下拉菜单中选择的Traffic实例。
- **清除所有：**单击可以删除所有Test Run下拉菜单中的Traffic实例。
- **数据源行：**显示HTTP traffic使用的数据源行值。选择相应的行来显示相关的SOAP请求和SOAP响应。（数据源行菜单仅在相应的测试用例有使用数据源时被显示）。
- **请求：**显示请求的HTTP Header和Body，对应到从Test Run下拉菜单选择的测试，同样对应到选择的数据源行（如果可用）。



你可以通过 3 个不同的标签查看请求Body。每一个标签页提供不同的图形界面展示XML消息。3 个标签页如下：

- **文本：**默认，请求body显示在文本标签页。Literal标签页允许你直接编辑请求Body的文本。如果你发现你必须滚动左右滚动条以便查看HTTP traffic，你可以在Literal视图单击并按下CTRL + B美化XML。按下CTRL + B后，将显示良好的XML格式，缓解必须从左到右滚动。
- **树：**标签页允许你通过一个树视图配置请求Body。
- **元素：**显示了XML元素名称和值对，没有杂乱的名称空间和属性。允许您快速验证一个元素的存在，并确认它有正确的值。
- **响应：**显示响应的HTTP Header和Body，对应到从Test Run下拉菜单选择的测试，同样对应到选择的数据源行（如果可用）。



你可以通过 3 个不同的标签查看响应Body。每一个标签页提供不同的图形界面展示XML消息。3 个标签页如下：

- **文本：**默认，响应body显示在文本标签页。Literal标签页允许你直接编辑响应Body的文本。如果你发现你必须滚动左右滚动条以便查看HTTP traffic，你可以在Literal视图单击并按下CTRL + B美化XML。按下CTRL + B后，将显示良好的XML格式，缓解必须从左到右滚动。
- **树：**Tree标签页允许你通过一个树视图查看响应Body。
- **元素：**显示了XML元素名称和值对，没有杂乱的名称空间和属性。允许您快速验证一个元素的存在，并确认它有正确的值。

你知道吗？

您可以切换请求和响应视图，通过单击请求或响应标签页。

第 5 课：理解测试成功和失败

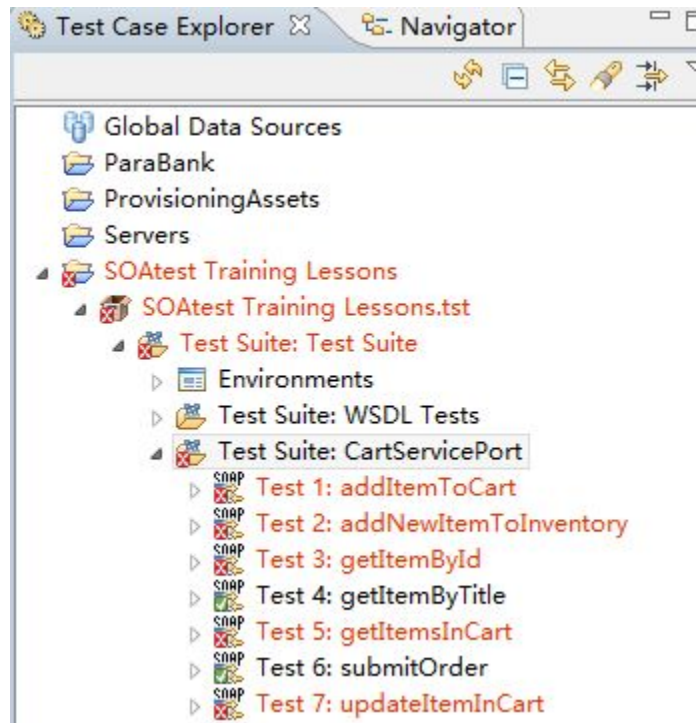
SOAtest如何标示测试成功：

1. 在Test Case Explorer中选择Test Suite: CarServicePort。
2. 单击工具栏按钮Test。



3. 审查Test Progress视图中报告的结果(标示Example Configuration——执行的测试配置的名称)。同时注意在Test Case Explorer中紧挨着每个测试的标记。绿色的标记表示测试成功。如果测试失败，将显示红色的X图标。

到目前为止，示例在 Test Case Explorer 中显示了 2 个绿色的图标表示每个测试的成功，5 个红色的图标表示每个测试的失败。测试套件也被标记了红色的图标表示其下有失败的测试。



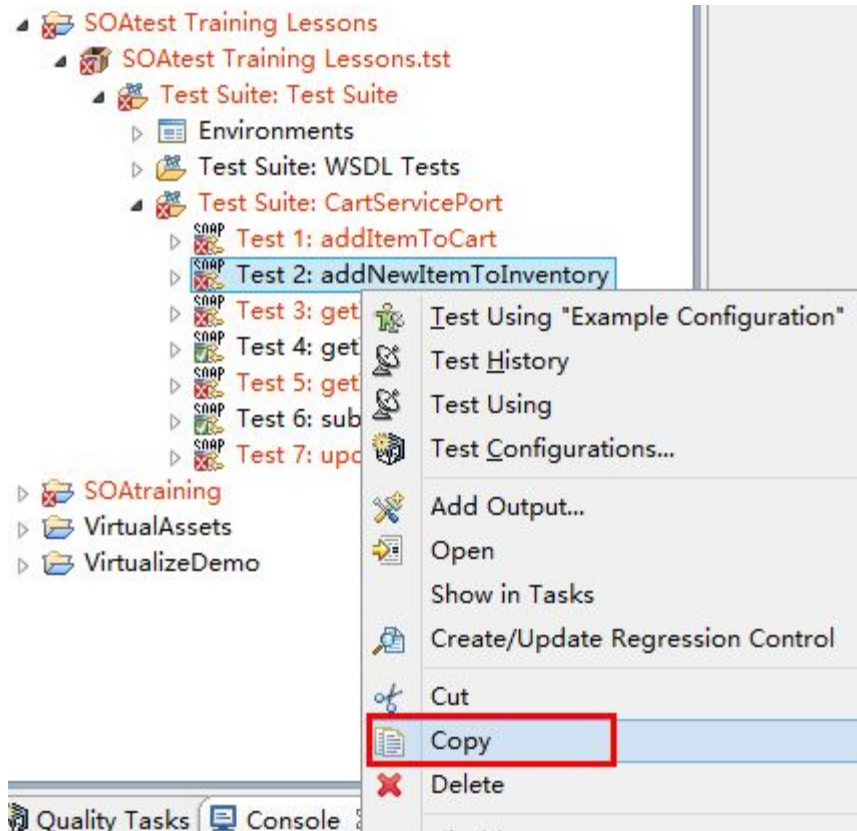
SOAtest运行一个测试套件或一个独立的测试用例后，任何错误都将显示在质量任务视图。如果你双击这些错误，SOAtest将显示SOAP失败的消息和堆栈追踪。或者，你也可以查看测试的HTTP traffic来查看实际的XML响应。

第 6 课：添加更多测试，删除不需要的测试

最快的添加测试到测试用例的方式是复制一个相似的自动生成的测试，然后修改所需的测试参数。这将是非常有用的，例如，如果你想要一个测试来自动生成参数，以便你可以测试服务如何处理各种可能的输入，但是你还想测试服务如何处理一组特定的输入。

如何添加一个新的测试到当前测试套件：

1. 复制一个你想测试的业务操作的测试用例（自动生成），通过右击这个测试的Test Case Explorer 节点，然后从快捷菜单选择Copy。



2. 粘贴测试到测试套件，通过右击你想在其下新建测试的Test Case Explorer节点，然后从快捷菜单选择Paste。如果测试用例包含输出，则输入会被一起复制到新的测试用例。
3. 选择新建的测试的Test Case Explorer节点，然后修改所需的测试参数。
4. (可选)以正常的方式添加一个输出到测试用例（右击测试并选择Add Output）。你也可以使用一个SOAP Client作为另一个的输出。

删除新添加的测试，右击你新添加的测试用例的Test Case Explorer节点，然后从快捷菜单中选择Delete。

你知道吗？

你可以一次删除多个测试，通过按下 CTRL 键用鼠标选择，然后右击从快捷菜单中选择 Delete。

第 7 课：嵌套测试套件

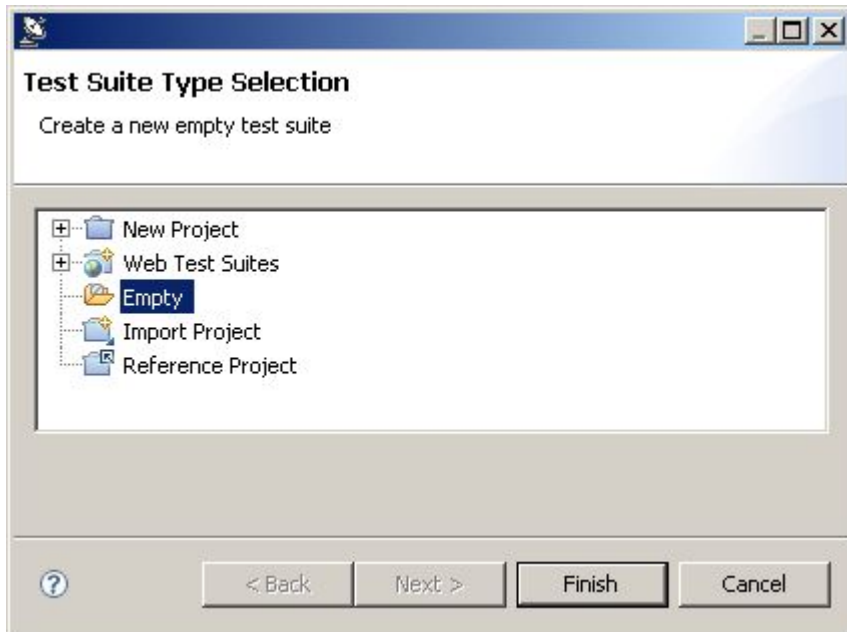
你可以进一步组织你的测试套件，通过群组测试用例。这将特别有用的，如果你的项目是由大量的测试分类组成，而且你想分离和区分你的测试用例。

嵌套测试套件：

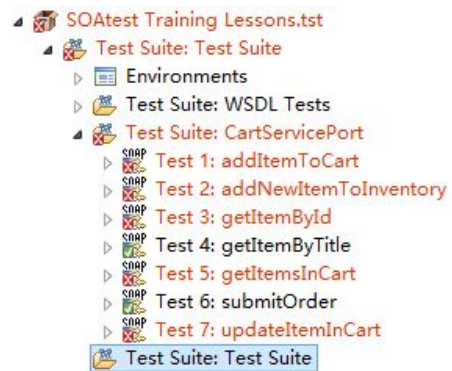
1. 在Test Case Explorer中选择Test Suite: Test Suite根节点，然后单击Add test suite...工具栏按钮。



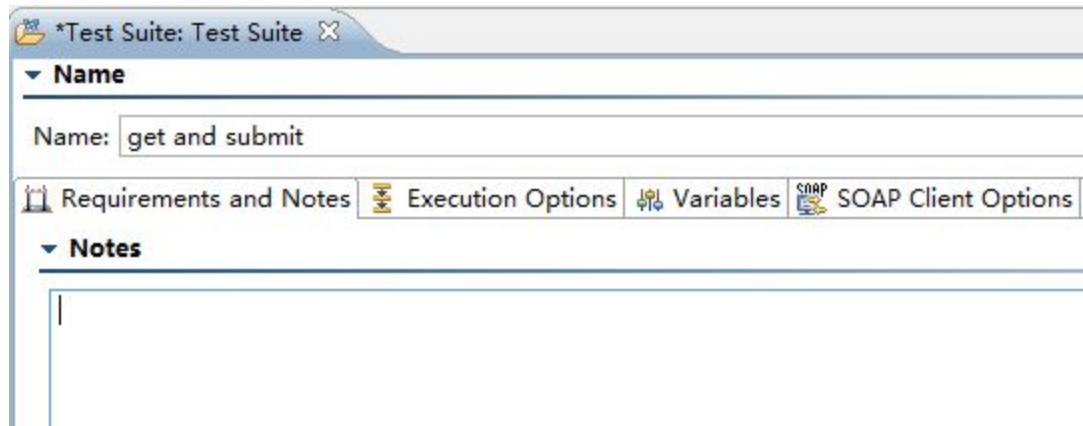
2. 在Add Test Suite向导中，选择Empty并单击Finish。



一个空的测试套件被创建：



3. 在测试套件配置面板(在界面右侧)，Name字段中输入“get and submit”。



4. 单击工具栏按钮**Save**。
5. 复制Test 4: getItemByTitle到get and submit测试套件。
6. 复制Test 3: submitOrder到get and submit测试套件。
7. 在Test Case Explorer标签页右击Test Suite: Add and Subtract节点并从快捷菜单选择Expand All。你拖拽的测试现在被添加到嵌套的测试套件中。
8. 执行整个测试套件，通过选择主测试套件节点并单击Test按钮。或者执行一个单一的测试套件通过选择嵌套的测试套件节点并单击Test按钮。

你也可以保存嵌套的测试套件到一个独立的文件，通过右击期望的嵌套测试套件节点并从快捷菜单选择Export。

你知道吗？

你可以从 HTTP traffic 导入测试套件。SOAtest 将从 HTTP traffic 解析有效的 SOAP 请求并创建 SOAP Client。

练习 1: 创建 *SOAtest* 项目用于培训练习

既然已经熟悉了<http://localhost:18080/parabank/services/store-01?wsdl>，试着做做如下内容：

1. 使用同一个WSDL自动创建一个新项目。
2. 项目命名为SOAtest Training Exercises。
3. 执行测试套件。
4. 创建一个空的测试套件，命名为submit and get。
5. 将执行成功的测试用例复制到submit and get。
6. 重新排列测试套件使得顺序为submitOrder, getItemByTitle
7. 执行测试套件。

第 3 章：自动 Web 服务验证和管理测试数据

在这章中我们将介绍功能测试的概念。功能测试确保服务对于给定的请求能做出适当的响应。然而，由于Web服务的复杂性，这个任务一点都不简单。对于大多数Web服务，它不可能预测客户端将发送哪种类型的请求。列出所有可能的请求是不可行的，因为可能的输入空间是无限的。因此，验证服务是否可以处理各种请求类型和参数非常重要。要做到这一点，你可以使用数据源功能参数化SOAtest用于单元测试的值。当服务需要测试存储在电子表格或数据库中的值，数据源功能特别有用。

功能测试的第二个方面是创造回归测试来确保今天能正常运行的内容将来还能正常运行。回归测试在任何更改引起错误时通过提醒来帮助维持你的网络服务的完整性。它是通过检查过去通过功能性测试的服务是否继续通过功能性测试来做到的。

这章中我们将使用<http://localhost:18080/parabank/services/store-01?wsdl>，它是一间在线书店的应用程序接口。这项网络服务由以下操作组成：

- `getItemById(int)` - 以给定的条目id返回书籍。当前有效值是 1 到 9。
- `getItemByTitle(String)` - 匹配你的标题搜索条件返回书籍对象列表。这个操作每调用 5 次，它返回的条目价格将增加 1.00 美元。示例关键字：Linux, Java, C++, program。留空将获取数据库中的所有书籍。
- `addItemToCart(int, int)` - 输入条目id和数量，返回一个包括一个书籍对象、数量和一个唯一订单号的订单对象。
- `getItemsInCart(int)` - 返回存在指定购物车中的订单列表。
- `addNewItemToInventory(Book)` - 使你能够将新书加到数据库中（虚拟的）。随便添加任何你想要的东西；并不会真的将他们添加到持久数据库。在 20 分钟内没有访问这些新的条目，就会从数据库中删除。

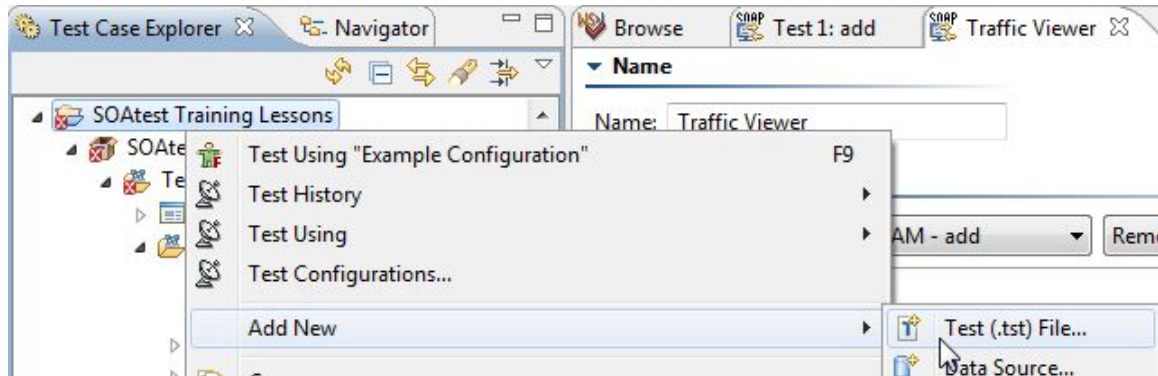
第 1 课：通过在整个响应上自动地创建回归控制来配置 Web Service 验证

回归控制让你通过一个简单有的方式来确定是否Web service的功能自最后一次测试执行后已经改变。SOAtest使用回归控制来表示测试的预期数据或“基线”。SOAtest能自动创建回归控制。当需要这样做时，SOAtest用当前选择的测试用例的输出来创建Diff控制。

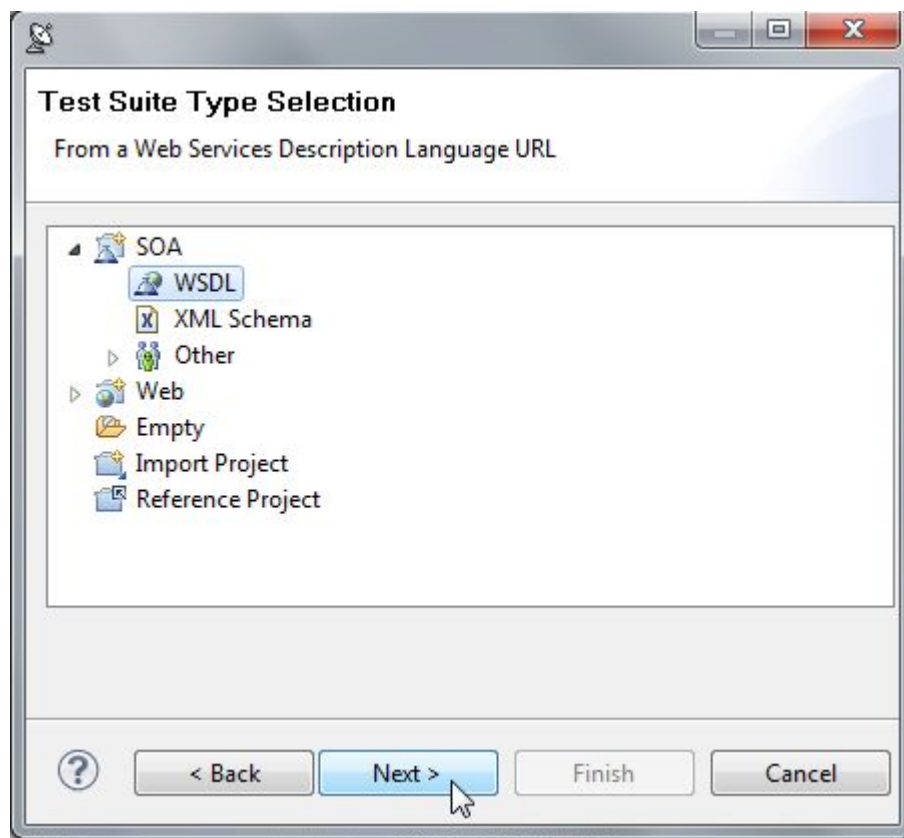
如果你为自动生成的测试创建回归控制，SOAtest将对每个测试用例使用相同的输入值，并检查随后的测试执行是否产生相同的测试输出。你可以为整个测试套件创建回归控制 或是选择的测试用例。

创建一个 .tst 文件，在其中添加回归控制：

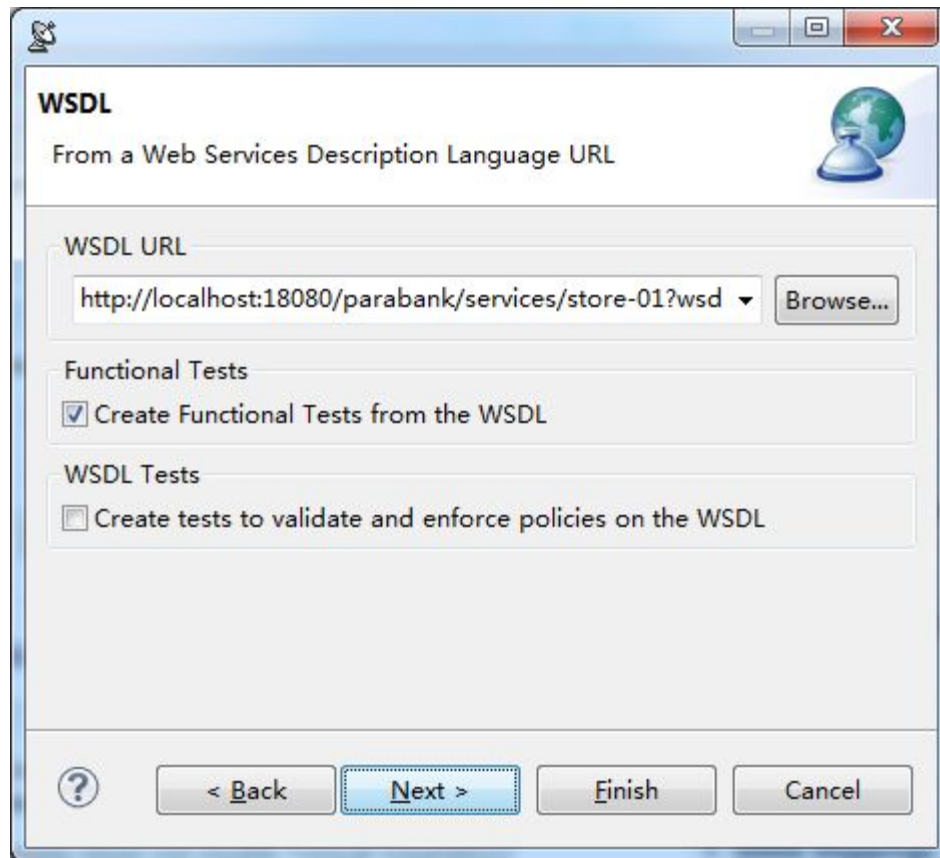
1. 右击之前章节的 SOAtest Training Lessons 工程，然后从快捷菜单选择 Add New > Test (.tst) File。



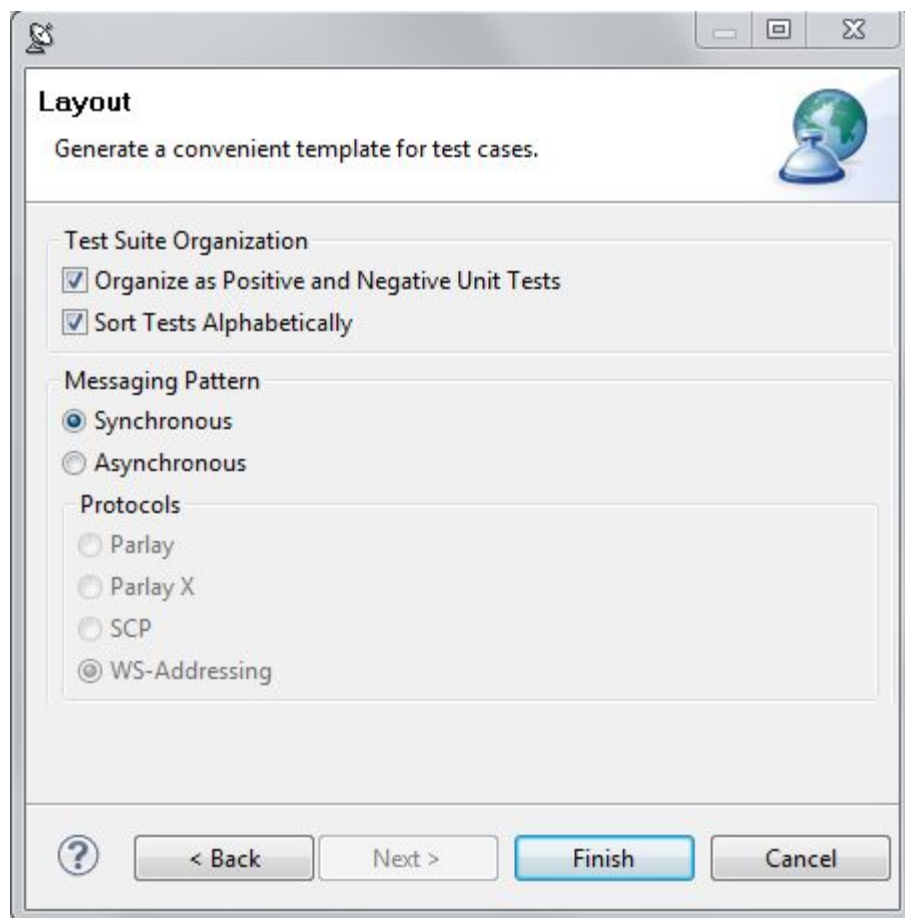
2. 为文件输入一个名称 (例如，Intermediate Chapter)，然后单击 Next。
3. 选择 SOA > WSDL，然后单击 Next。



4. 在 WSDL URL 字段中输入 `http://localhost:18080/parabank/services/store-01?wsdl`。
5. 如果没有勾选 Create Functional Tests from the WSDL 多选框，勾选它。

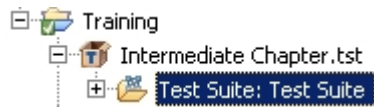


6. 单击Next按钮 4 次，出现Layout对话框。
7. 勾选Organize as Positive and Negative Unit Tests 多选框。



8. 单击按钮**Finish**。新创建的测试套件显示在测试用例浏览器中。

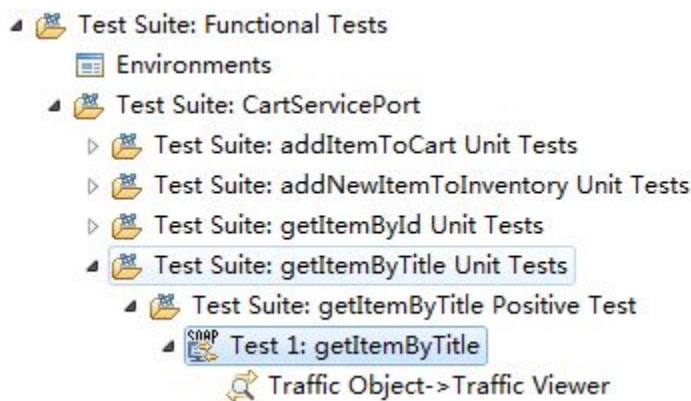
9. 双击新建的Test Suite: Test Suite节点。



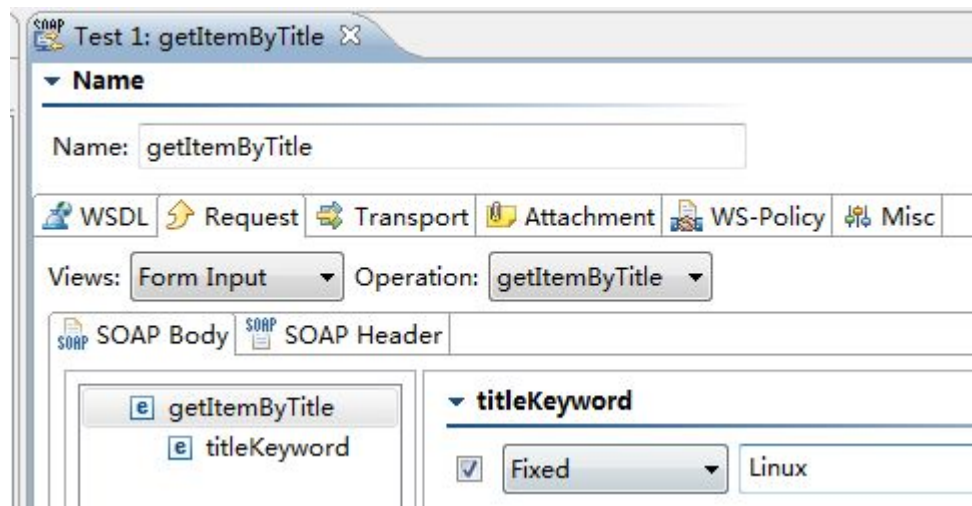
10. 在测试套件配置面板(在界面右侧), Name字段中输入“Functional Tests”。
11. 单击工具栏按钮Save。

浏览新建的项目并利用SOAtest自动创建回归控制:

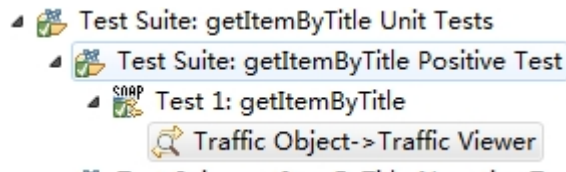
1. 在Functional Tests测试套件中右击Test Suite:CartServicePort节点, 然后选择Expand All。这时, 将显示7个子测试套件, 其它中的每个子测试套件中会显示一个测试用例。7个测试套件覆盖了WSDL的每一个业务操作。7个测试套件针对7业务操作分别包含正面和负面测试套件。它们是两种重要的测试情况, 发送预期或非预期的数据到服务。
2. 右击Test Suite:CartServicePort 节点并选择Collapse Children, 然后右击Test Suite: getItemByTitle Unit Tests 节点并选择Expand Children。
3. 双击Test Suite: getItemByTitle Positive Test> Test 1: getItemByTitle 节点。



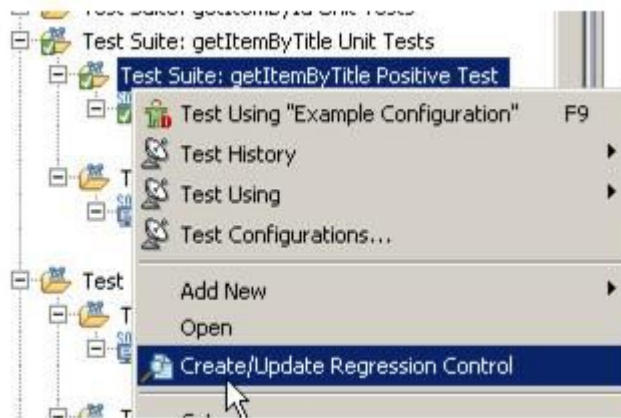
4. 在测试配置面板打开Request 标签页, 在titleKeyword字段输入Linux, 然后单击Save。这指明我们将使用关键字Linux查询书籍。



5. 选择Test 1: getItemByTitle 节点并单击Test工具栏按钮。这将使用参数Linux调用getItemByTitle业务操作。
6. 展开Test 1: getItemByTitle节点并双击Traffic Object> Traffic Viewer 节点。HTTP Traffic面板将打开并显示从执行运行记录的traffic。



7. 右击Test Suite: getItemByTitle Positive Test 节点并从快捷菜单选择Create/Update Regression Control，然后在打开的响应验证向导中选择Create/Update Regression Control，单击Finish。



SOAtest自动执行测试基于从服务收到的值创建回归控制。

我们现在有一个功能测试，它用一个单一的输入值测试Web service的getItemByTitle业务操作。可以用相同的方式为WSDL中定义的其他业务操作创建功能测试。

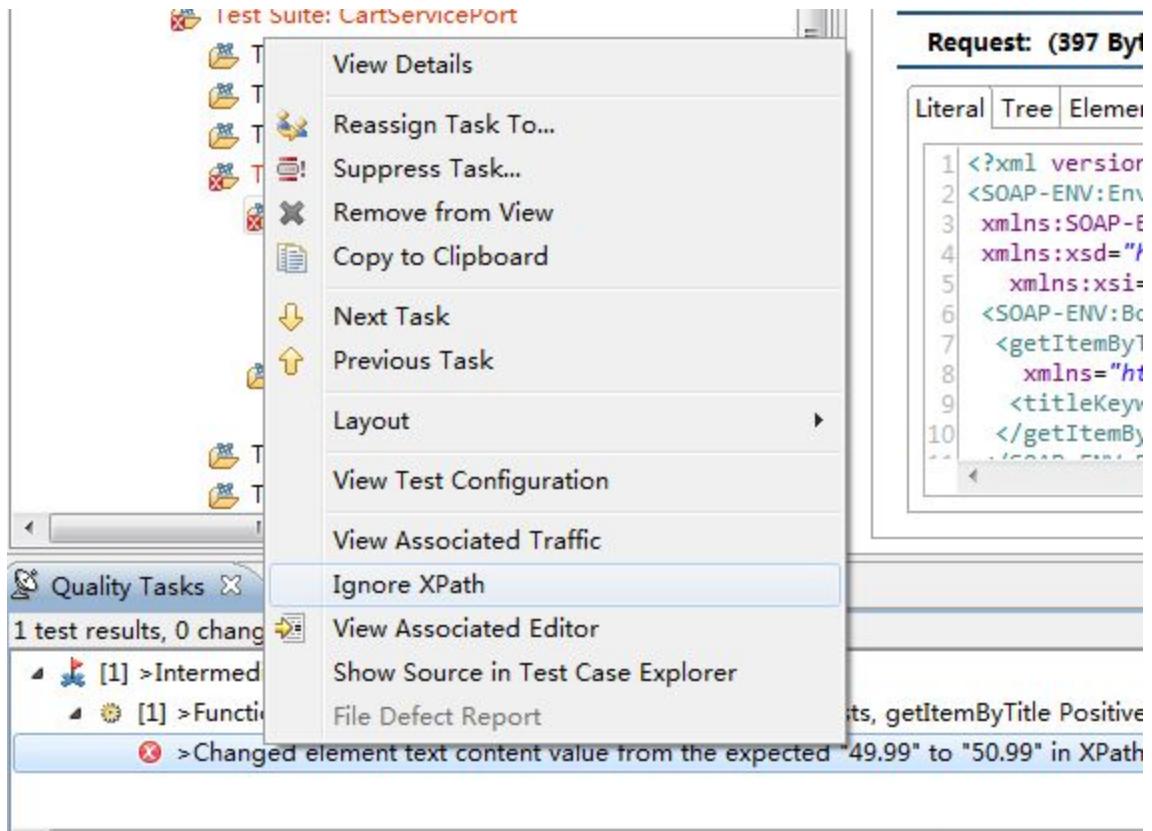
你知道吗？

你可以为任何工具的任何输出创建回归控制。通过为每个输入添加一个新的Diff工具并选择Update Regression Controls，你可以为多重输出创建多重回归控制。

第2课：在验证 Web Services 响应时忽略 XML 消息中动态或不适当的值

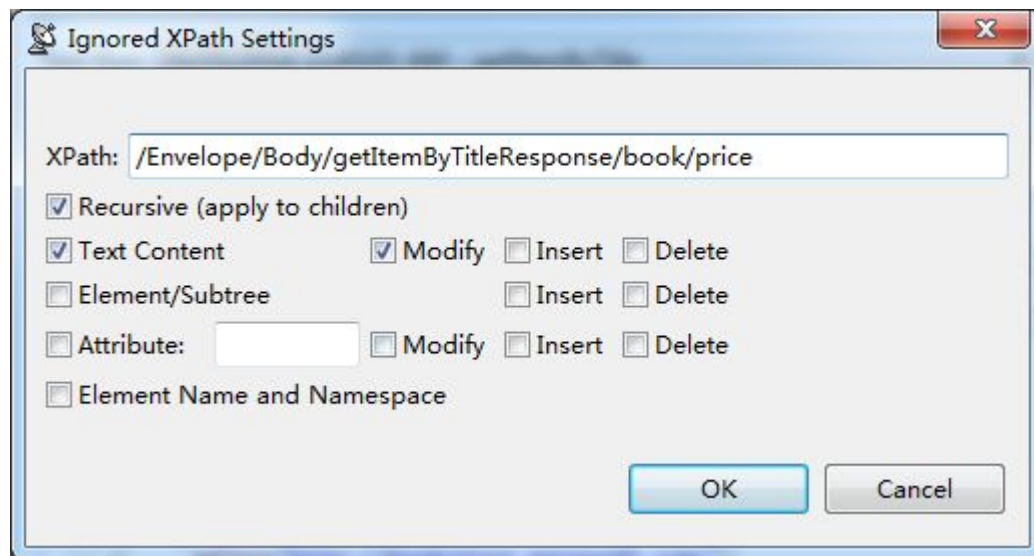
当创建回归控制时，时间戳或会话变量会导致它失败，忽略这些动态值对我们创建回归控制非常有帮助。在bookstore示例中，“price”元素是一个动态值，测试每执行 5 次书的价格将上升 1 元。在这个示例中，我们将配置回归控制忽略这个元素。

1. 执行Test 1: getItemByTitle数次。
2. 注意数次测试执行后，回归控制失败且一个任务报告在质量任务面板。这是因为price元素改变了。在这个测试用例中我们想忽略price元素。
3. 在质量任务视图右击错误信息并从快捷菜单选择Ignore XPath。



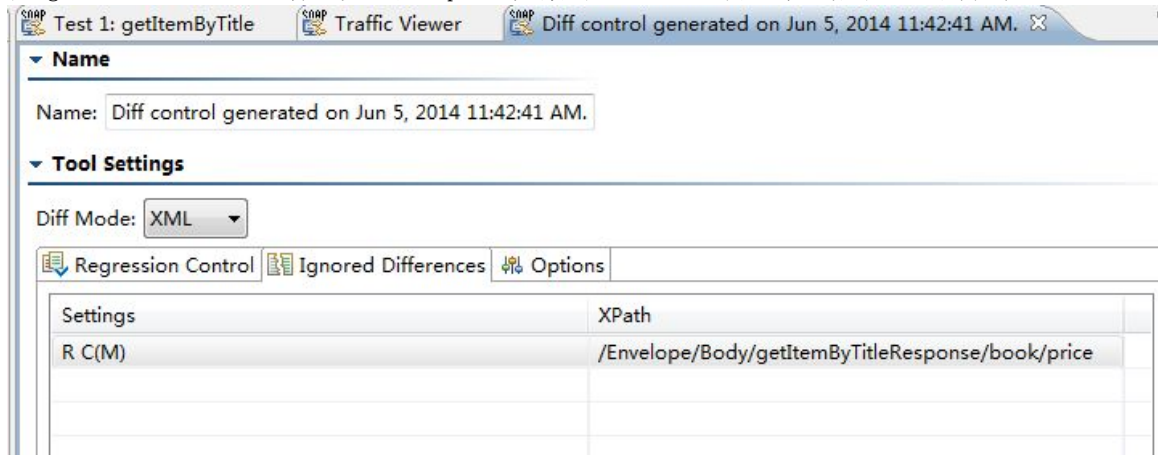
将出现 Ignored XPath Settings 对话框。price 元素的 xpath

“/Envelope/Body/getItemByTitleResponse/Result/i/price” 被自动填充到 XPath 字段。



4. 确保 Recursive, Text Content 和 Modify 多选框都有勾选并单击 OK。这将指示回归控制递归的忽略任何 price 元素中文本内容的修改。
5. 在 Test 1: getItemByTitle 中双击 Response SOAP Envelope > Diff control 节点。
6. 在测试配置面板中选择 Ignored Differences 标签页。

7. 在Ignored Differences标签页，注意price元素的XPath已经添加到忽略的XPath列表中。



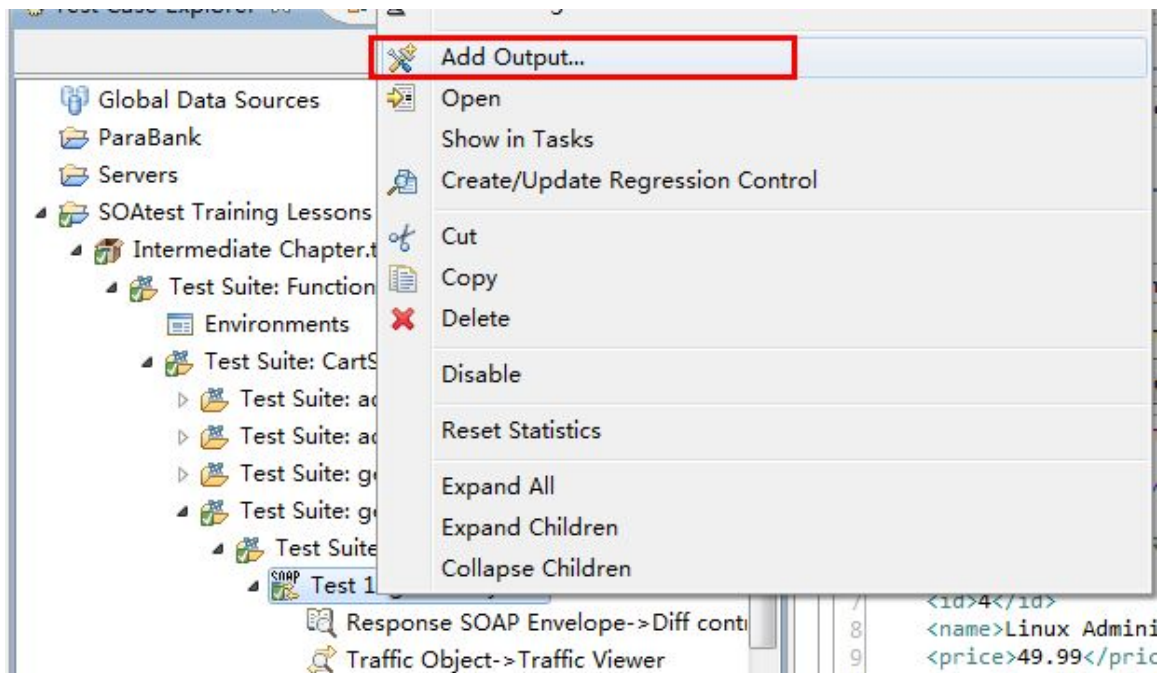
XPath指定的所有price元素现在将被忽略。

8. 再次执行测试Test 1: getItemByTitle，它会成功。

第3课：使用XML断言器验证Web服务响应消息中的指定元素

你可以使用XML断言器来强制XML消息的数据正确性。它通常连接到一个SOAP Client或Messaging Client来验证服务返回的数据。XML断言器支持复杂的消息验证需求，不需要编写脚本就允许你简单地创建和维护XML消息的断言。使用XML断言器完成如下步骤：

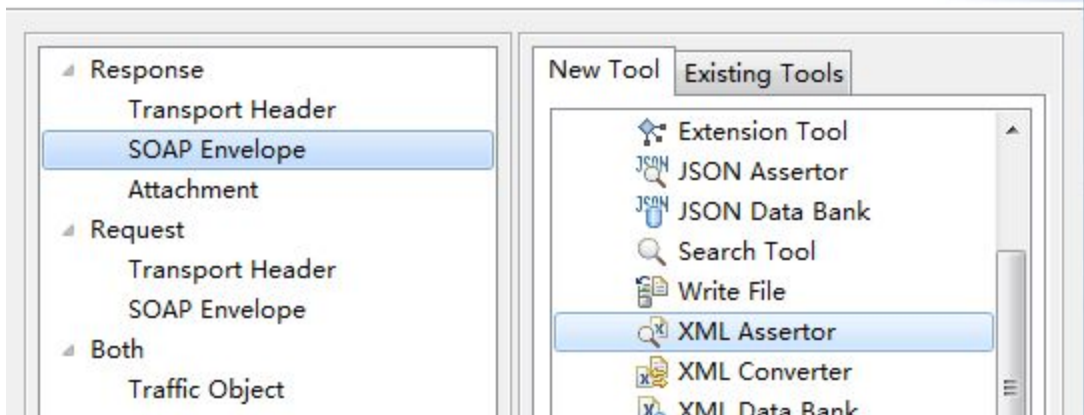
1. 右击Test 1: getItemByTitle节点（源自之前的课程）并从快捷菜单选择Add Output。



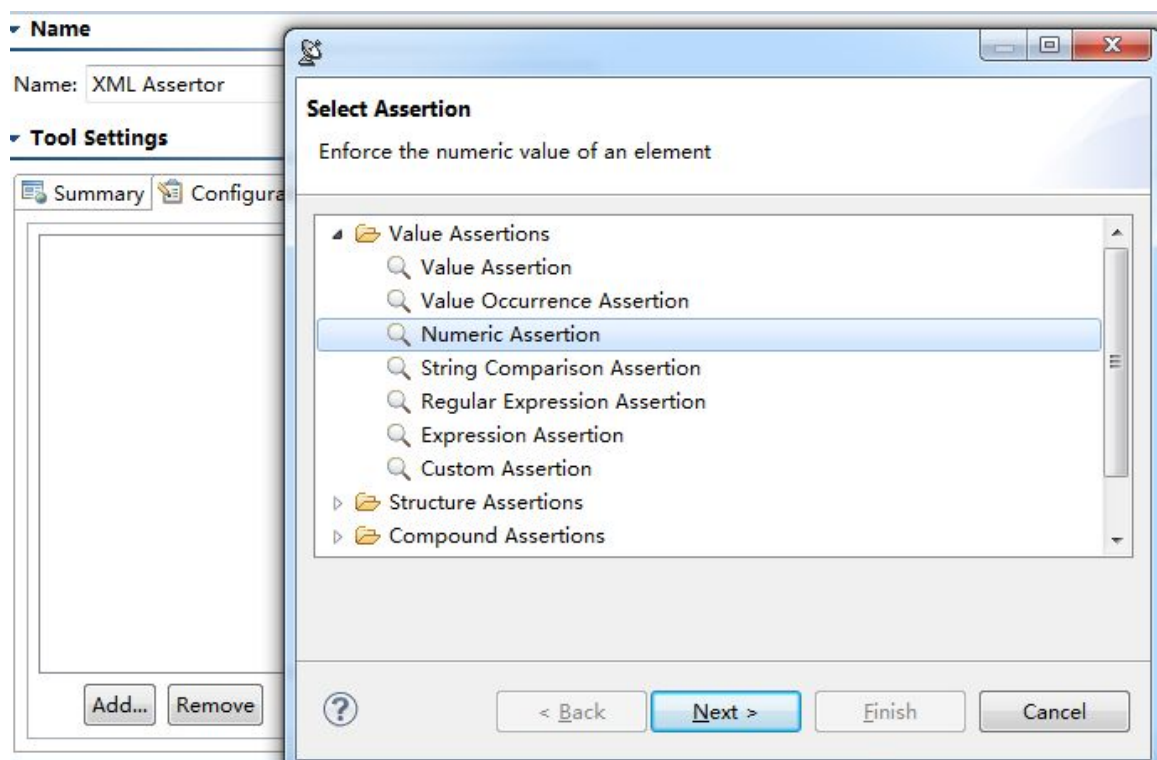
2. 在添加输出向导中，在左边面板选择Response> SOAP Envelope，在右边面板选择XML Asserter，单击Finish按钮。

Add Output

Send the output from one tool as input to another.

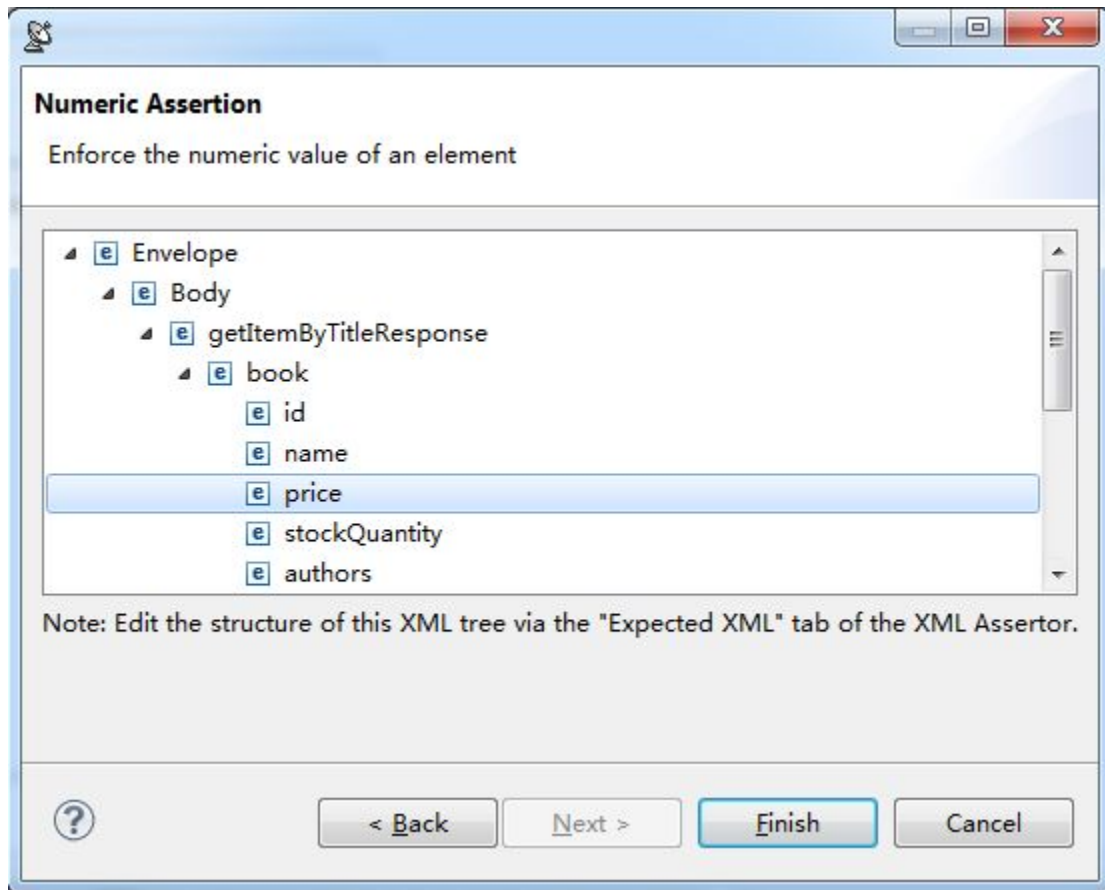


3. 在XML Asseror面板，打开Configuration 标签页并单击Add。
4. 在选择断言向导中，展开Value Assertions，选择Numeric Assertion，然后单击Next。



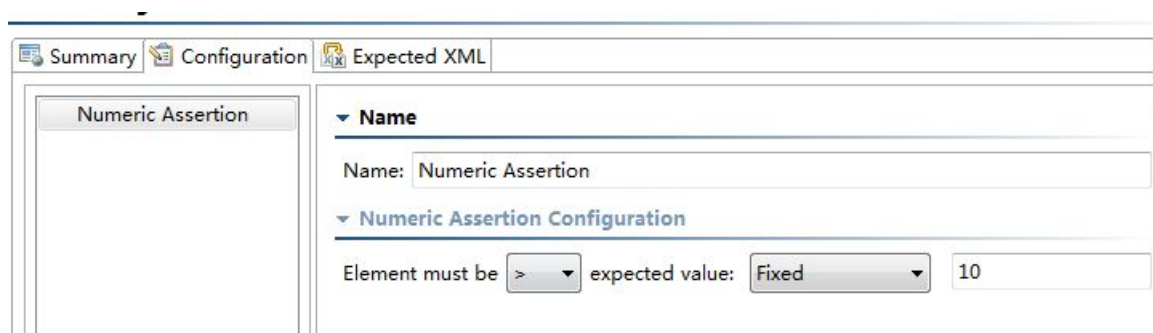
Numeric Assertion 对话框将显示XML消息的树状视图，你可选择一个要验证的值。

5. 从Numeric Assertion对话框选择price元素并单击Finish按钮。



XML断言器的配置标签页现在填入了一个数字断言。

6. 在XML断言器的Configuration 标签页。从Element must be下拉菜单select “>”，并在Expected Value字段输入10。



7. 单击Save工具栏按钮来保存XML断言器配置的更改。
8. 单击工具栏按钮Test。测试成功。

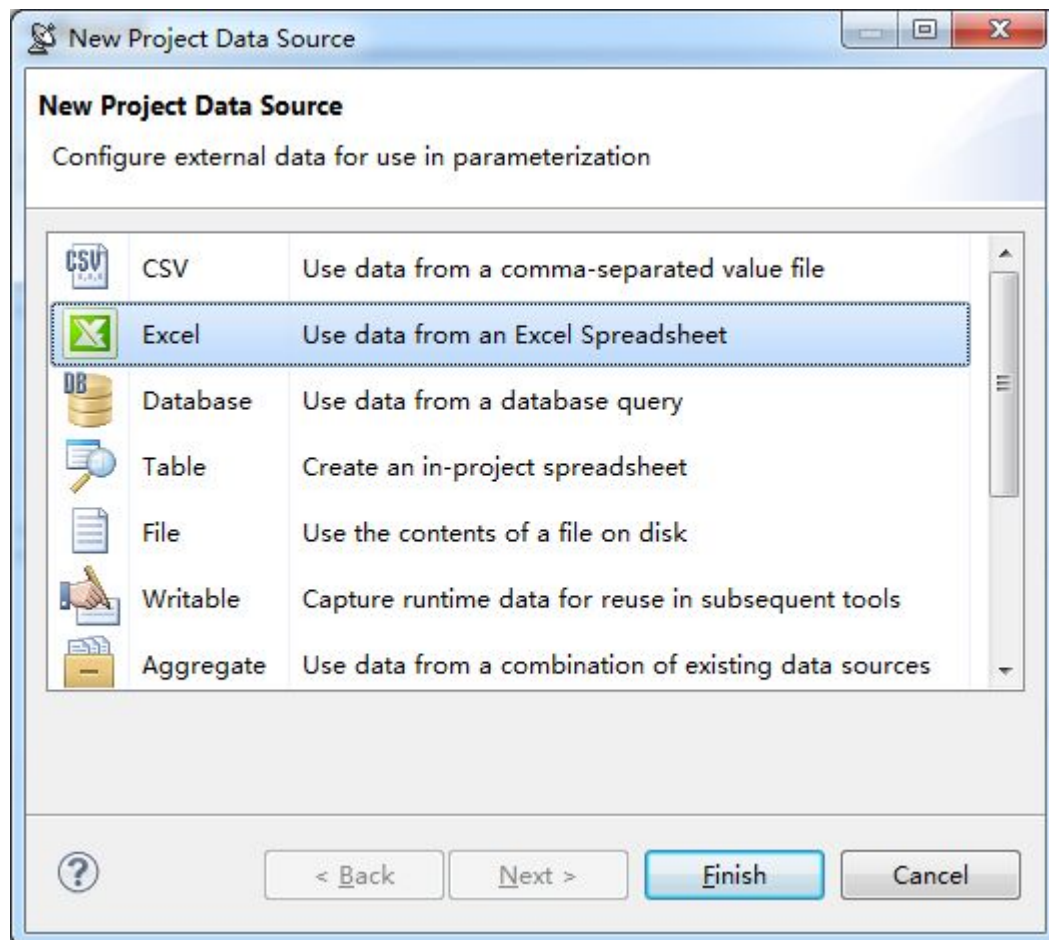
你可以添加其它断言应用到消息上(诸如一个数字断言来验证price元素)， 通过在XML断言器Configuration标签页单击Add按钮。

第 4 课：使用数据源来驱动测试和场景

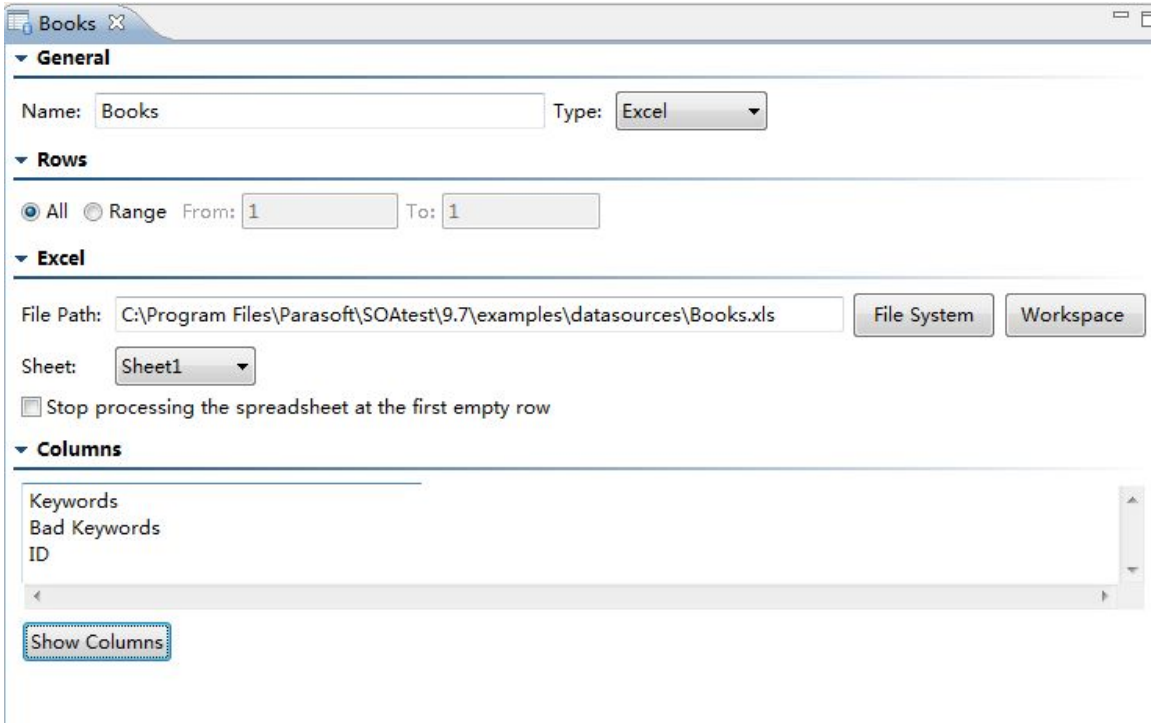
功能测试确保服务对于给定的请求能做出适当的响应。服务是否能处理各种请求类型和参数非常重要，不仅仅是单一的值。针对这一目标，复制相同的测试和场景于是能执行不同的输入数据，但扩展不好且难以维护。

在这章中，我们将使用S0Atest的数据源功能。这允许我们在一个单一的测试用例中测试多个输入值。数据源功能特别有用，当服务需要测试存储在电子表格或数据库中的值。

1. 右击根测试套件节点Test Suite: Functional Tests 并从快捷菜单选择Add New> Data Source...
2. 从New Data Source向导中选择Excel并单击Finish按钮。



3. 在数据源配置面板，完成如下步骤：
 - 在Name字段中输入Books。
 - 单击File System按钮定位并选择Books.xls文件（在S0Atest安装目录下的examples/datasources目录下）。
 - 单击Open。
 - 单击Show Columns按钮来显示Excel电子表格列名。
 - 单击工具栏按钮Save。



Books

General

Name: Books Type: Excel

Rows

☒ All ☐ Range From: 1 To: 1

Excel

File Path: C:\Program Files\Parasoft\SOAtest\9.7\examples\datasources\Books.xls File System Workspace

Sheet: Sheet1

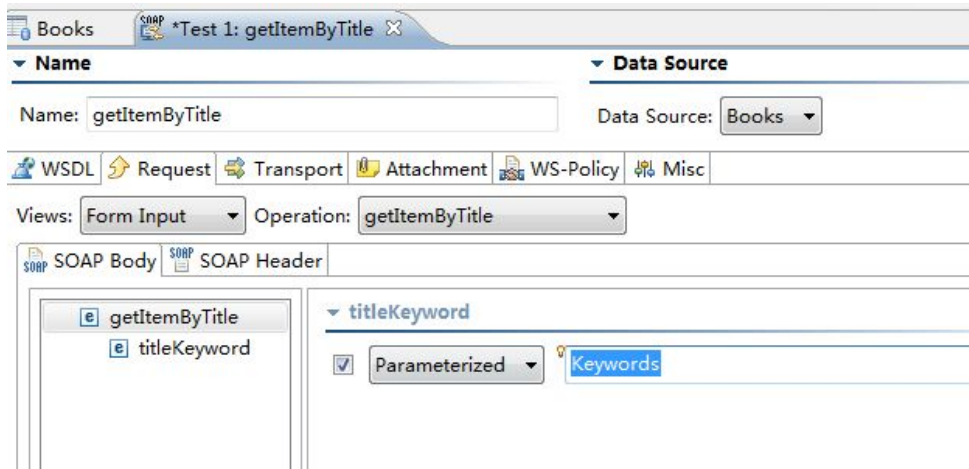
☐ Stop processing the spreadsheet at the first empty row

Columns

Keywords
Bad Keywords
ID

Show Columns

4. 返回并双击getItemByTitle 测试节点（源自之前课程）。Books应该已经显示在Data Source下拉菜单中，它现在出现在测试配置面板。
5. 为测试配置面板底部的titleKeyword下拉菜单，选择Parameterized和Keywords，然后单击Save工具栏按钮。



Books ***Test 1: getItemByTitle**

Name Name: getItemByTitle **Data Source** Data Source: Books

WSDL Request Transport Attachment WS-Policy Misc

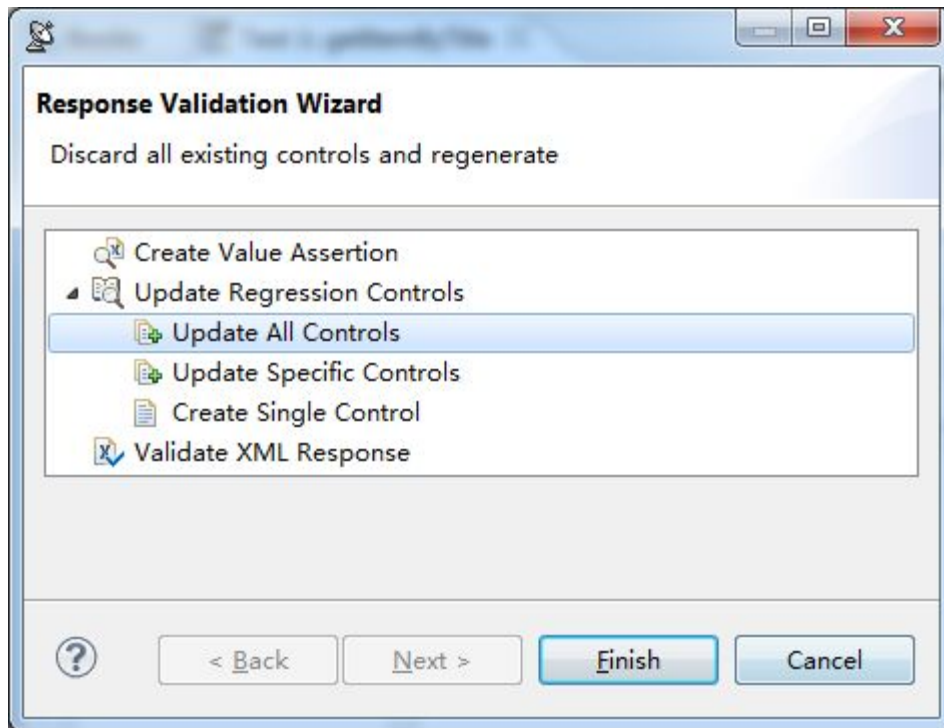
Views: Form Input Operation: getItemByTitle

SOAP SOAP Body SOAP Header

getItemByTitle
titleKeyword

☒ Parameterized Keywords

6. 单击Test工具栏按钮并注意质量任务面板中的错误信息。针对Keywords列中每一行都将执行一次测试，但由于之前的创建的XML断言器和回归控制而失败。现在我们需要更新回归控制。
7. 右击Response SOAP Envelope> XML Asserter节点并从快捷菜单选择Delete。
8. 右击getItemByTitle节点并选择Create/Update Regression Control。
9. 在Response Validation向导中，展开Update Regression Controls节点，选择Update All Controls，并单击Finish按钮。



- 如果我们期望数据源的每一行都返回相同的响应，Create Single Control是适当的选择。测试过程中，数据源中每行返回的响应都将与这个控制进行比较。
- 选择Update All Controls是因为我们期望数据源中每行返回不同的响应。现在将为数据源中每行创建不同的控制。测试过程中，数据源每行返回的响应都将与其对应的控制进行比较。

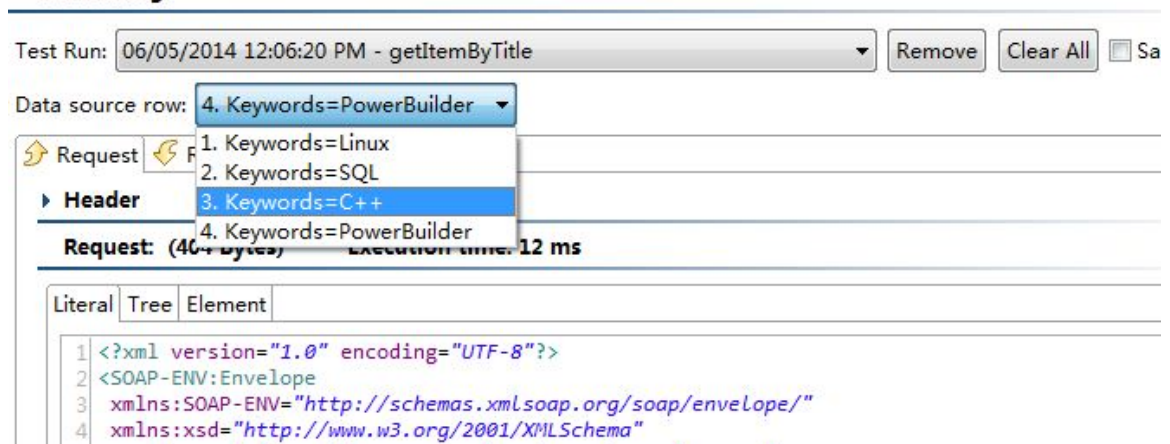
10. 选择Response SOAP Envelope> Diff control节点。Diff控制显示在界面的右边。

注意，Data source row下拉列表包含多个为数据源中每行创建的控制。你可以在这里改变控制，如果执行不同的控制实例。更多配置Diff工具的信息，参见SOAtest用户手册。

11. 选择getItemByTitle节点并单击工具栏按钮Test。

12. 双击getItemByTitle节点下的Traffic Object> Traffic Viewer节点并注意测试执行了4次，Keyword列中每个keyword值执行了一次。

▼ Tool Settings



你知道吗？

你可以指定数据源的行的范围来执行测试。在测试创建时，你可很方便的设置行从 1 到 1 来简化测试的创建。

如果从一个数据源中创造回归控制失败，错误将会报告在质量任务视图中，包含对于特定的测试使用的数据源中的特定行号。

你知道吗？

你可以双击主测试套件节点打配置面板，并改变多个回归控制映射策略。

练习 1: Bookstore 测试套件创建

尝试如下步骤:

1. 在SOAtest Training Exercise .tst文件的根节点Test Suite:Test Suite下, 针对 <http://localhost:18080/parabank/services/store-01?wsdl> 创建一个新的测试套件。
2. 命名测试套件为Bookstore Unit Tests。

练习 2: Customer #1 测试套件创建

接下来, 完成如下步骤:

1. 从之前的练习复制测试套件Test Suite: CartServicePort并粘贴到根节点Test Suite: Test Suite。
2. 重命名新的测试套件为Customer #1
3. 从 Customer #1 中删除以下的测试: `addNewItemToInventory`, `submitOrder`, `updateItemInCart`。

练习 3: 数据源和多个回归

最后, 完成如下步骤:

1. 添加一个内建表数据源到测试套件Customer #1。
2. 添加一列ID#并填入值 1, 2, 3, 4, 5, 6。
3. 添加一列Keywords并填入值C++, Linux, PowerBuilder, SQL。
4. 用数据源的列参数化每个测试用例的getItemByID和getItemByTitle。
5. 为两个测试都创建多个归控制。
6. 忽略价格的XPath。
7. (提高) 为每个XML响应添加一个XML验证器 (详细请参见第 7 章第 1 课)。

第 4 章：Web 服务场景测试

在前面的章节中，我们学习了如何单元测试一个Web 服务操作。这个基本上就足够对应用服务的测试了。在大多数情况下，为了测试Web服务的应用程序逻辑，还需要进一步的测试。例如，让我们看看getItemsInCart() 操作。从操作导致的响应将依赖于你所做的操作顺序。这样，getItemsInCart() 需要伴随着其他操作，例如submitOrder() 一起测试整个应用程序的逻辑。

在本章中，你将进入到 SOAtest 的高级功能，这将有助于您的应用程序逻辑测试。你将学习如何使用XML来提取通过XML Transformer 可用于在链接工具中的Xpath/Element值。或者在测试套件中可用作其他工具的输入。你也将获得关于如何在SOAtes中编写脚本的知识。

第1 课：使用XML Transformer 工具来对XML 消息的一部分进行回归控制

通常情况下，一个SOAP消息中包含了大量的XML 有效载荷。但是，你可能只关心该有效载荷的一小部分，并希望使用SOAP响应或者请求的某部分的元素来创建回归控制。一次性地忽略所有这些元素可以变得非常繁琐，而且对太多的参数使用XML Assertor来执行字符串比较也可能是很乏味。在这种情况下，使用 XML Transformer工具将是更有效地。

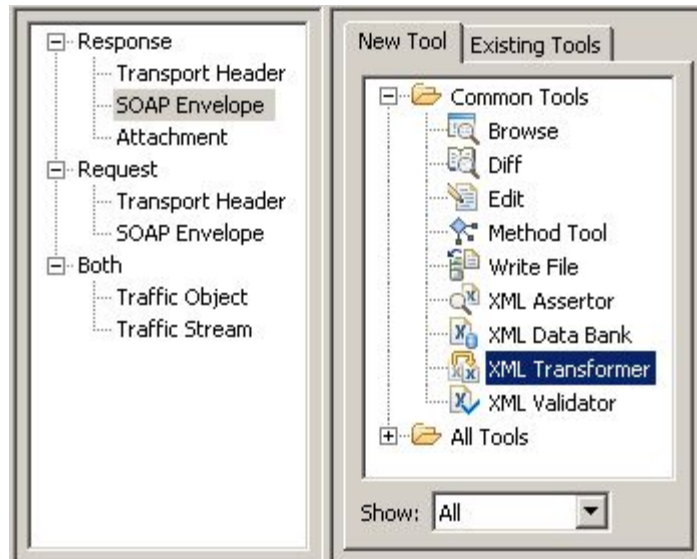
XML Transformer 为你提供了 XSLT 一样的功能来转换任何 XML。如果你想创建一个只使用SOAP响应或者请求的几个元素来创建一个回归控制，这样是非常有用的。

创建一个应用XML Transformer的.tst文件：

1. 右击之前章节的SOAtest Training Lessons工程，然后从快捷菜单选择New Test (.tst) File。
2. 为文件输入一个名称(例如，Advanced Chapter)，然后单击Next。
3. 选择New Project > WSDL，然后单击Next。
4. 从WSDL URL字段中选择http://localhost:18080/parabank/services/store-01?wsdl。这个应该在下拉菜单中出现，因为它在之前的课程中输入过。
5. 如果没有勾选Create Functional Tests from the WSDL多选框，勾选它。
6. 单击按钮Finish。新创建的测试套件显示在测试用例浏览器中。
7. 双击Test 4: getItemByTitle并置空keyword字段。

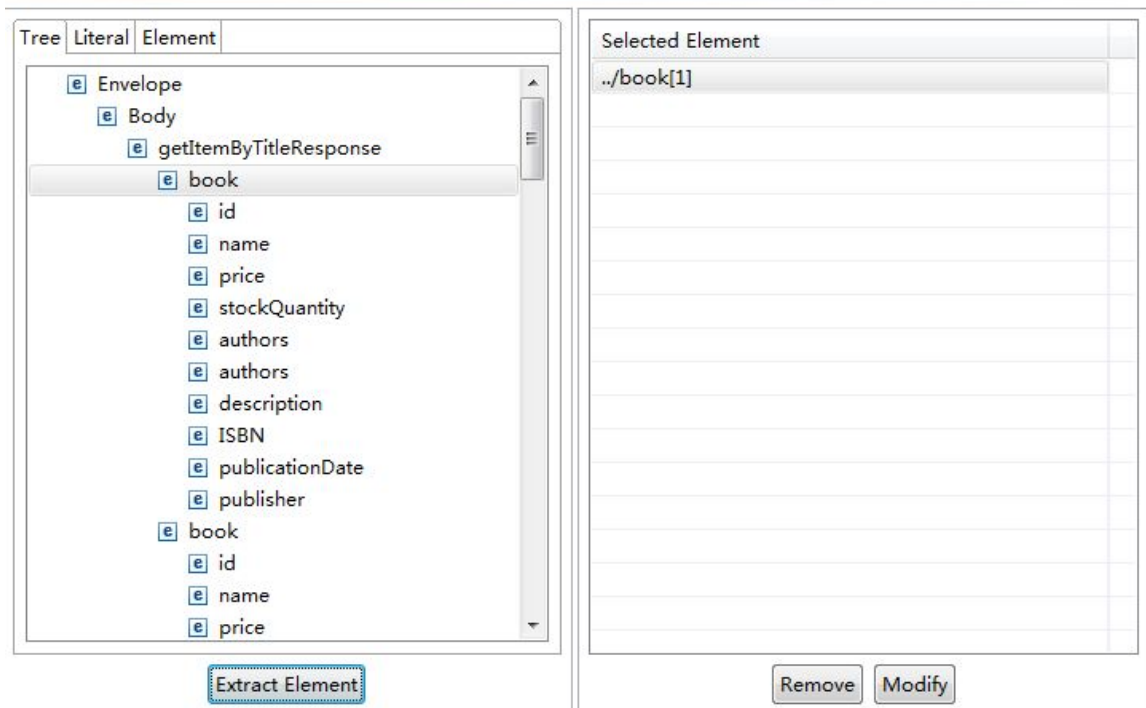
配置XML Transformer，完成如下步骤：

1. 右击Test 4: getItemByTitle测试节点并选择Add Output。



2. 在Add Output向导中，从左边面板选择Response> SOAP Envelope，从右边面板选择XML Transformer，并单击Finish。一个节点Response SOAP Envelope> XML Transformer 被添加至节点getItemByTitle下。
3. 注意，Response SOAP Envelope> XML Transformer 配置面板被打开并显示XML响应的树状视图，视图包含多个book元素，每个book元素的名称为i。
4. 从XML查看树选择第 1 个book元素，单击Extract Element按钮。Element被添加到Selected Element列表。添加到此处的Element将指定回归控制的元素。

Tool Settings

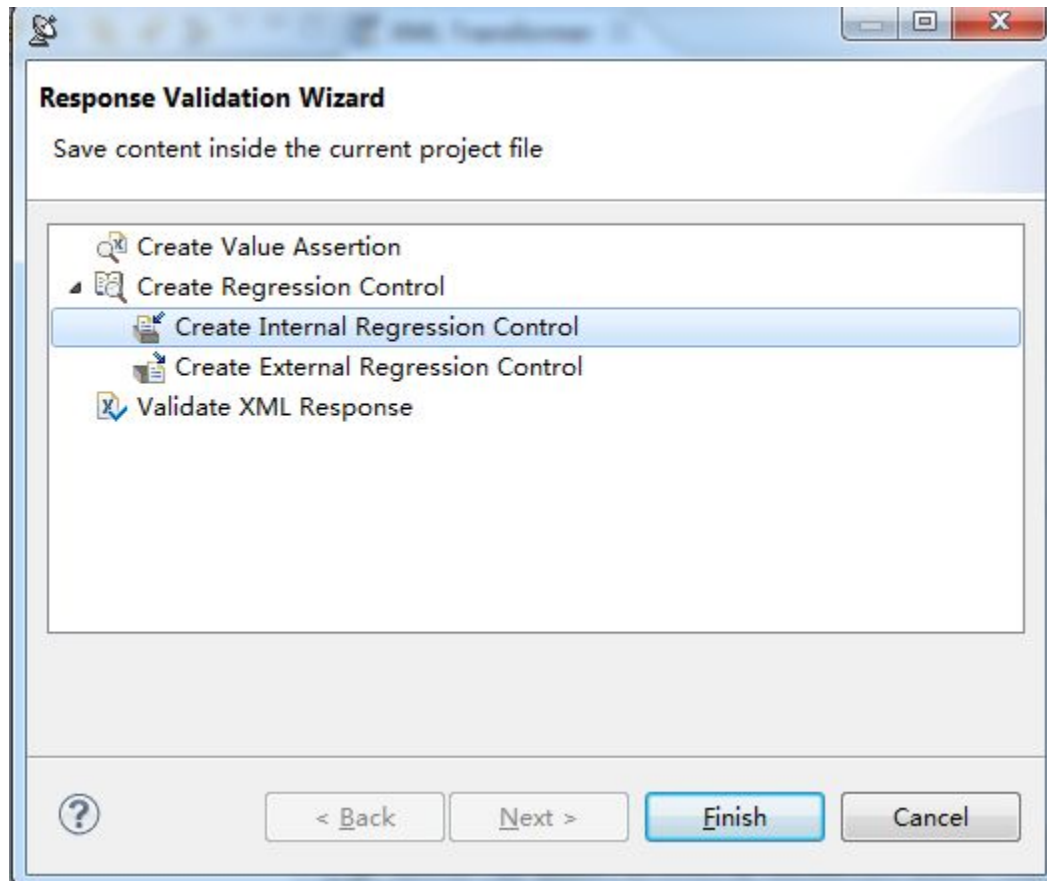


5. 展开Options部分并勾选Canonicalize XML output。

▼ Options

- ☐ Save expected XML
- ☒ Canonicalize XML output
- ☐ Allow alteration
- ☐ Extract empty elements as:
- ☐ Extract missing elements as:
- ☐ Wrap Output in XML

6. 单击Save工具栏按钮来保存XML Transformer的配置。
7. 右击测试节点Test 4: getItemByTitle, 然后选择Create/Update Regression Control。在对话框中选择选项Create Internal Regression Control。



在打开的对话框中，然后单击Finish。Transformed XML> Diff control 节点出现在Response SOAP Envelope> XML Transformer 节点下方。如果双击这个节点，你会看到你刚添加的第1本书（第1个book元素）出现在新的 Diff control中。

Diff control generated on Jun 5, 2014 1:35:01 PM. X

Name

Name: Diff control generated on Jun 5, 2014 1:35:01 PM.

Tool Settings

Diff Mode: XML

Regression Control Ignored Differences Options

Mode: ☒ Literal XML ☐ Form XML

Regression control source: Editor

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <book>
3   <id>1</id>
4   <name>C++ How to Program (4th Edition)</name>
5   <price>102.99</price>
6   <stockQuantity>20</stockQuantity>
7   <authors>Harvey M. Deitel</authors>
8   <authors>Paul J. Deitel</authors>
9   <description>One of the best C++ books</description>
10  <ISBN>0130384747</ISBN>
11  <publicationDate>2002-08-12T00:00:00+08:00</publicationDate>
12  <publisher>Prentice Hall</publisher>
13 </book>
14
```

现在XML中第 1 本书的任何变化都会导致测试失败。任何其它书籍的改变不会导致测试失败。注意，你可能想要如之前课程中所做一样，在这本书中忽略价格的值。

第 2 课：使用 XML Data Bank 工具将值从一个测试传递到另一个测试

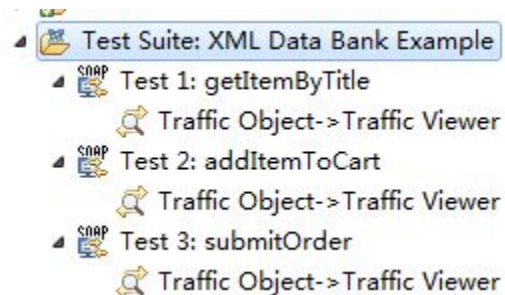
XML Data Bank 工具允许你将测试套件中某个测试用例的参数提供并应用于该测试套件中的另一个测试用例。换句话说，你可以提取 Test 1 的 SOAP 响应参数做为 Test 2 的 SOAP 请求参数。XML Data Bank 工具可以链接到任何其它输出 XML 的 SOAtest 工具。它能够提取任何来自 XML 的信息并用于之后的测试。

例如，你可以配置一个测试套件来测试 bookstore 的 Web service 交易。这个套件的 Test 1 能够使用用户 ID 登录，然后 SOAP 响应返回一个会话 ID。这个套件的 Test 2 被配置为使用 Test 1 的会话 ID 来执行交易。你可以随意配置测试套件中的测试用例使用 SOAP 响应参数做为 SOAP 请求参数。

在我们配置测试套件使用 XML Data Bank 工具之前，我们必须确定测试套件至少有 2 个测试用例可用。这一课，我们使用之前课程的测试套件：

在开始前，按如下步骤配置测试套件：

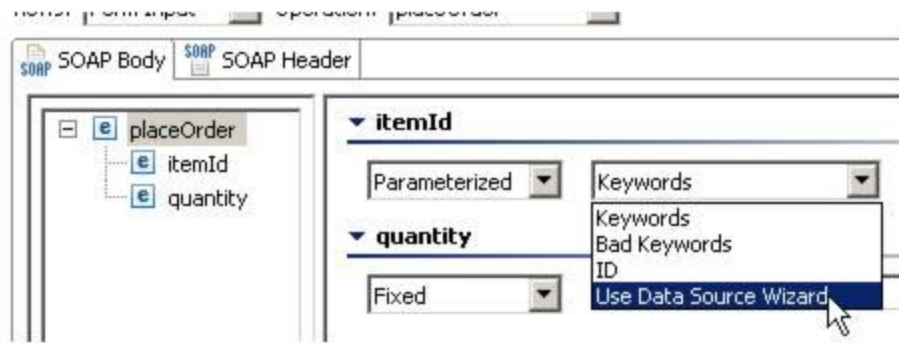
1. 选择 Test Suite: CartServicePort 测试套件节点，然后右击并从快捷菜单选择 Copy。
2. 右击根节点 Test Suite: Test Suite 并快捷菜单选择 Paste。这将复制测试套件 CartServicePort。
3. 双击新建的节点 Test Suite: CartServicePort 2 并重命名为 XML Data Bank Example。单击工具栏按钮 Save。
4. 在 XML Data Bank Example 下面配置 3 个测试按照如下的顺序：**getItemByTitle**, **addItemToCart** and **submitOrder**. 右击其它测试用例选择 Delete。



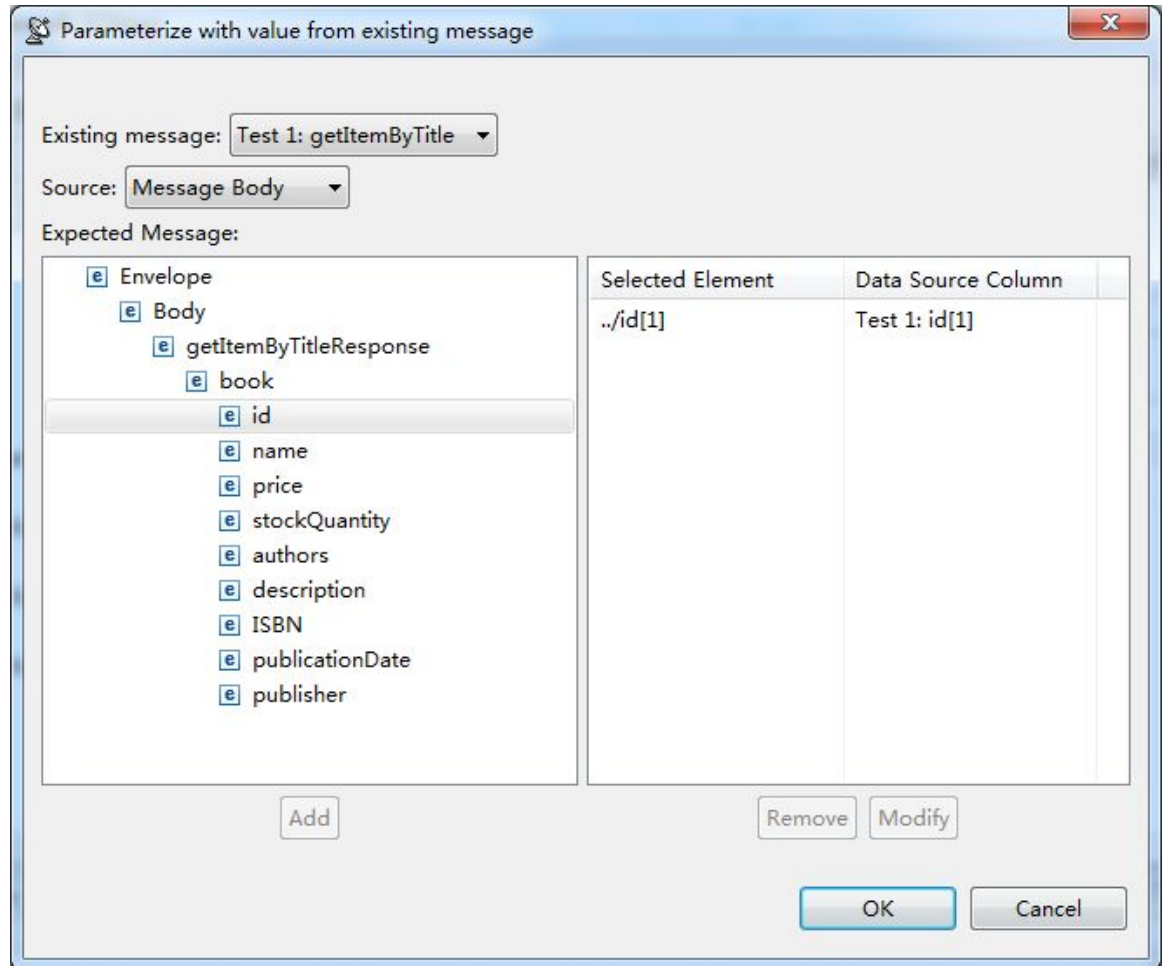
5. 确保第 3 个测试用例下除了 Traffic Viewer 外没有其它工具。展开节点 Test 1: getItemByTitle，如果 XML Transformer 工具有从之前的课程中复制过来，删除 XML Transformer 工具。在这个练中不需要这个工具，我们会在之后重新配置回归控制。
6. 右击根测试套件节点 Test Suite: Test Suite 并从快捷菜单选择 Add New> Data Source。

配置 XML Data Bank，按如下步骤：

1. 将 Test 1: getItemByTitle 请求 keyword 字段指定为 Linux。
2. 双击节点 Test 2: addItemToCart。
3. 从 itemId 元素下拉菜单中选择 Parameterized 和 Use Data Source Wizard。如果你的测试没有可见数据源，可能不会出现其它项，这没有关系。

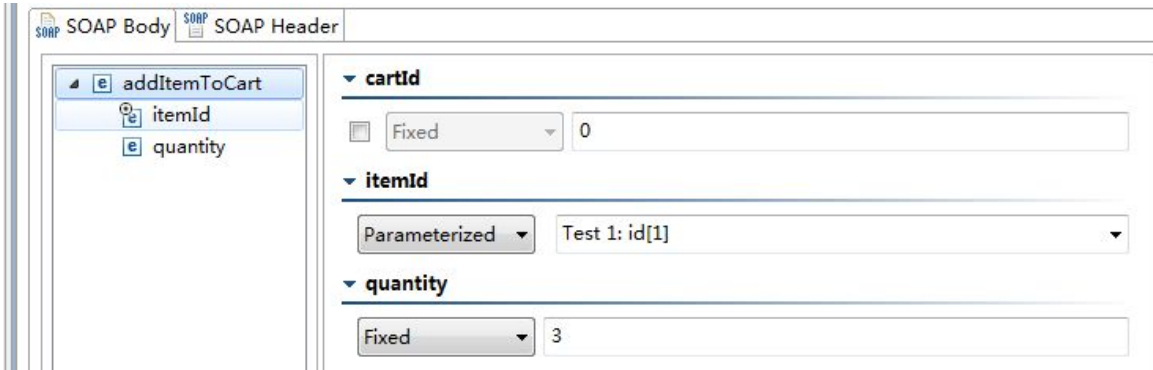


4. 按如下步骤完成Parameterize with value from existing test response 对话框，当测试执行用时，从Test 1 存储的itemId元素的值将自动做为itemId元素的值插入：
- 从对话框顶部Test栏选择Test 1: getItemByTitle。
 - 从预期的XML树中选择id元素并单击Add 按钮。Id元素显示在Selected XPath列中并对应显示与所选测试关联的列Data Source column name。
 - 单击OK 按钮，然后单击Save工具栏按钮。



Test 1: id现在做为itemId的一个参数值显示在界面右侧。同时你会发现Response SOAP Envelope> XML Data Bank 节点出现在测试套件中的getItemByTitle 节点下方。

5. 在测试配置面板中，为quantity元素输入一个固定的值 3，然后单击Save工具栏按钮。



SOAP Body SOAP Header

- addItemToCart
 - itemId
 - quantity
 - Fixed 3

cartId

Fixed 0

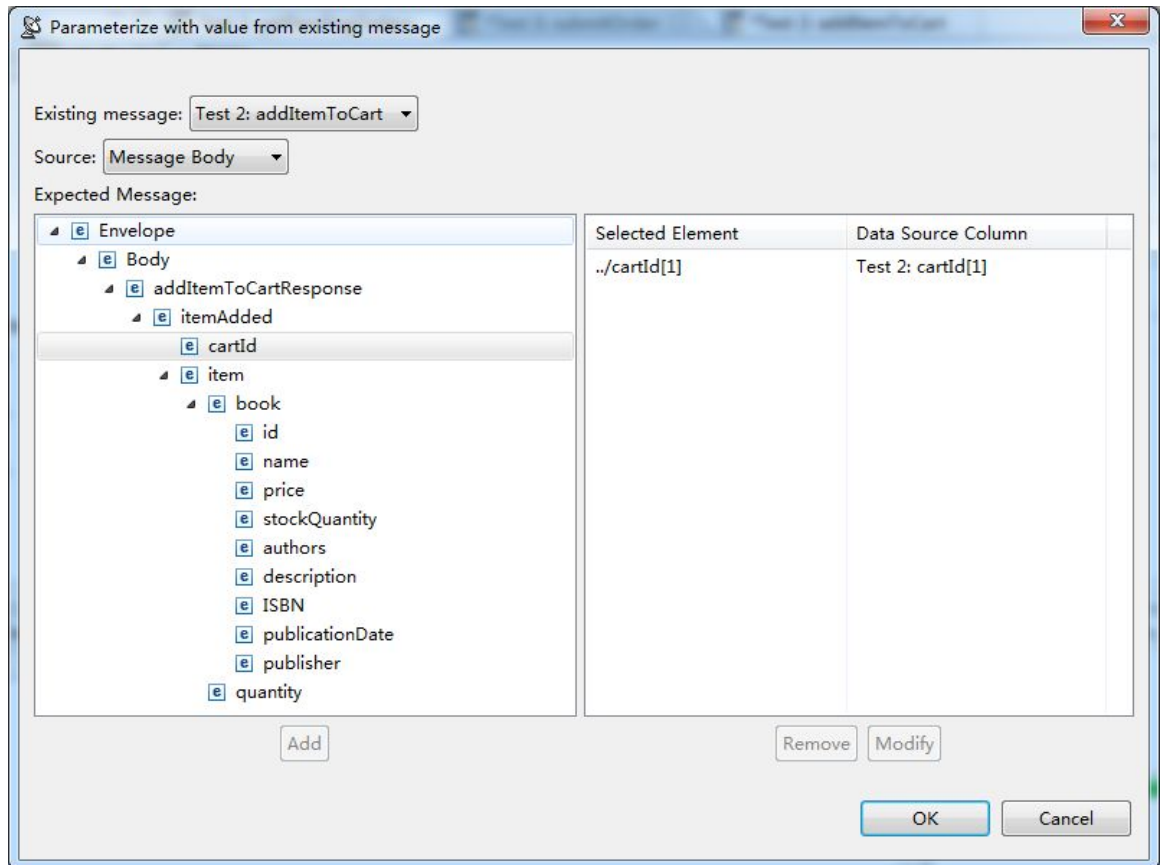
itemId

Parameterized Test 1: id[1]

quantity

Fixed 3

6. 双击测试节点Test 3: submitOrder。
7. 从carId元素下拉菜单中选择Parameterized 和 Use Data Source Wizard。
8. 按如下步骤完成Parameterize with value from existing test response 对话框，当测试执行用时，从addItemToCart存储的carId元素的值将自动做为carId元素的值插入：
- 从对话框顶部Test栏选择Test 2: addItemToCart。
 - 从预期的XML树中选择carId元素并单击Add按钮。carId元素显示在Selected Element列 中并对应显示与所选测试关联的列Data Source column name。
 - 单击OK 按钮，然后单击Save工具栏按钮。



Parameterize with value from existing message

Existing message: Test 2: addItemToCart

Source: Message Body

Expected Message:

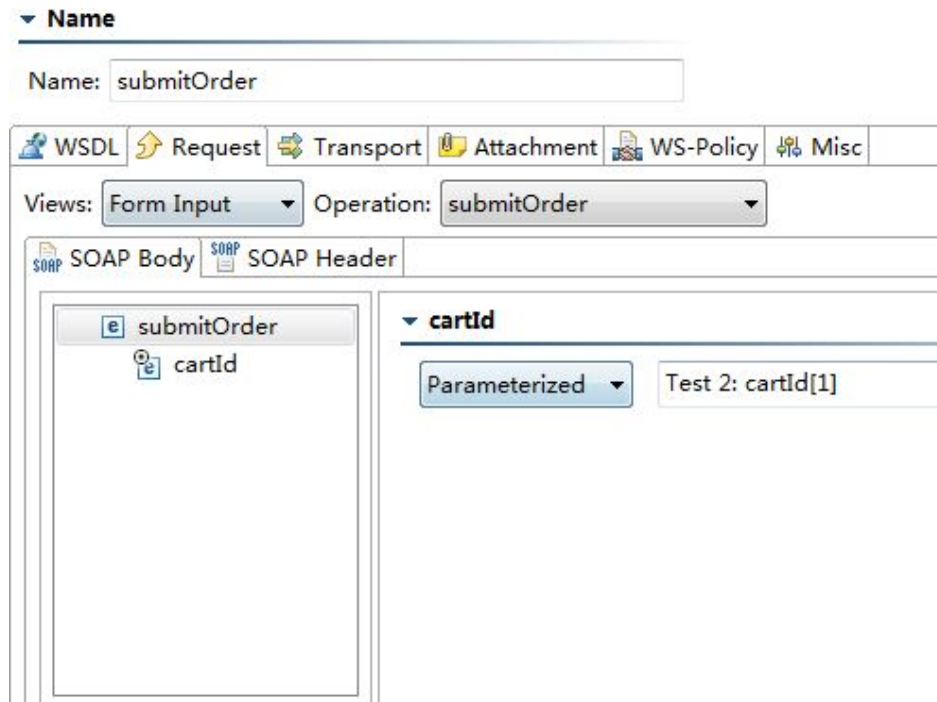
- Envelope
 - Body
 - addItemToCartResponse
 - itemAdded
 - cartId
 - item
 - book
 - id
 - name
 - price
 - stockQuantity
 - authors
 - description
 - ISBN
 - publicationDate
 - publisher
 - quantity

Selected Element	Data Source Column
./cartId[1]	Test 2: cartId[1]

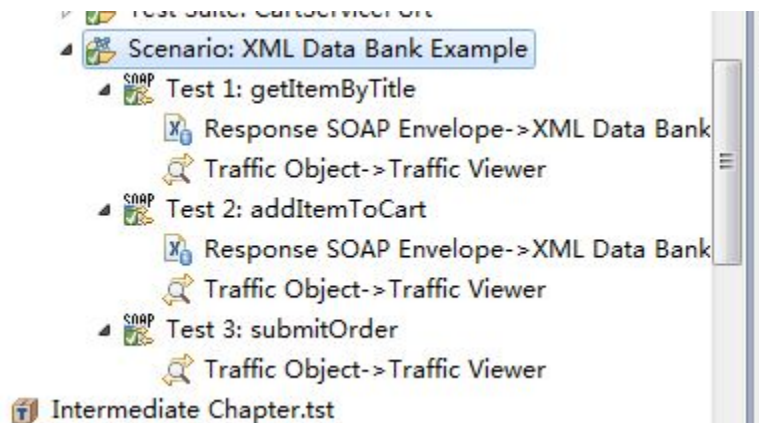
Add Remove Modify

OK Cancel

注意，carId[1]现在做为carId的一个参数值显示在测试配置面板。同时你会发现 Response SOAP Envelope > XML Data Bank 节点出现在测试套件Scenario: XML Data Bank Example中的placeOrder节点下方。



- 选择Scenario: XML Data Bank Example节点并单击Test工具栏按钮。当测试执行用时，从 addItemToCart存储的carId元素的值将自动做为submitOrder的carId元素的值插入。



- 浏览traffic，通过展开Scenario: XML Data Bank Example 并双击每一个测试用例的 Traffic Object > Traffic Viewer 节点。

注意，从Test 1 返回的书的itemId被用做Test 2 的输入。同时Test 2 的carId被用做Test 3 的输入。

现在让我们用数据源参数化测试场景，使得它可以迭代多本书而不是一本。

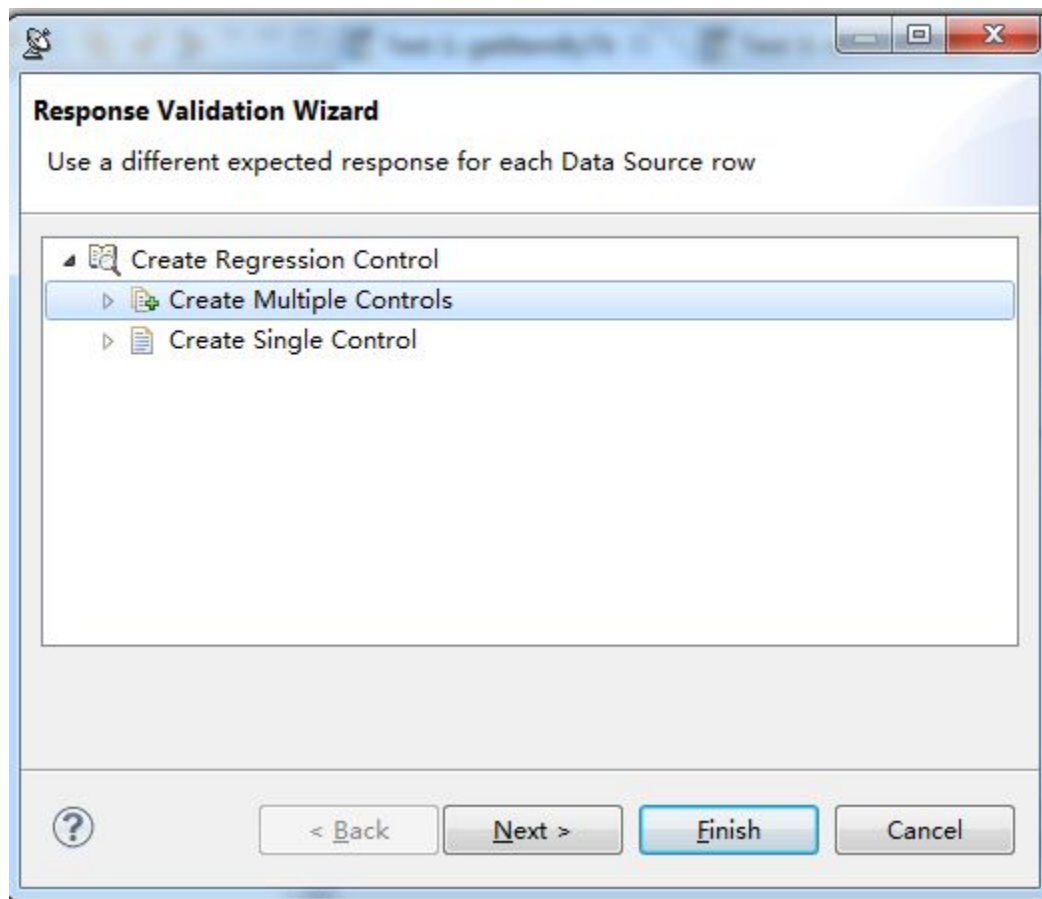
- 从 **New Project Data Source** 想到中选择 **Excel** 并且点击 **Finish** 按钮。
- 在数据源配置面板，完成如下步骤：
 - 在**Name**字段中输入Books。

- 单击File System按钮定位并选择Books.xls文件（在S0Atest安装目录下的examples/datasources目录下）。
 - 单击Open。
 - 保存
 - 单击Show Columns按钮来显示Excel电子表格列名。
3. 返回并双击节点**Test 1: getItemByTitle**。确保已经从Data Source下拉菜单中选择了Books，它现在出现在测试配置面板。
 4. 为测试配置面板底部的titleKeyword下拉菜单，选择Parameterized和Keywords，然后单击Save工具栏按钮。

当你执行这个场景（通过选择**Scenario: XML Data Bank Example** 然后单击测试按钮或按下F9键），这个场景将遍历数据源中所有可用的书。打开 traffic 并查看 Book id值是如何从响应到请求动态传递的。

现在让我们为场景定义成功的关键点让只有在我们期望的响应返回时测试才通过：

1. 右击节点Scenario: XML Data Bank Example并选择Create/Update Regression Control。
2. 在Response Validation向导中，展开Create Regression Controls节点，选择Create Multiple Controls，并单击Finish按钮。测试被执行且回归控制被添加到每个SOAP Client测试的响应下。



3. 选择Scenario: XML Data Bank Example节点并单击Test工具栏按钮。注意现在所有的测试都失败了。

4. 检查在质量任务视图中出现的错误消息。回归失败是因为在响应消息内部出现的动态内容（价格和订单号）。在下面的步骤中，我们将忽略这种动态数据类型的元素。
5. 在质量任务视图中，右键点击每个测试套件节点下面报告的第一个错误并且从快捷菜单中选择 **Ignore XPath**。在显示对话框中的 **Ignore XPath Settings** 中，点击 **OK** 按钮。你应该忽略三个 XPath 在这个步骤中，每个测试用例一个。
6. 选择 **Scenario: XML Data Bank Example** 节点并且点击测试工具栏按钮。所有的测试应该都成功通过。

现在你已经创建了一个完整的功能性场景测试来测试一个可能的业务处理，这个业务处理是在正常使用书店服务过程中可能会发生的。这个场景不依赖于硬编码的 book ID 或者订单号，而是通过运行一个测试步骤得到这些动态值并且使用他们。这在处理这种动态值是特别有价值的，例如会话令牌，认证令牌，事务 ID 等。同样在很难获得一个具有稳定数据并且测试数据经常改变的测试环境时，这种方式也是非常有用的。在这一课中我们创建的测试场景可以用来测试这种业务，不需要考虑在数据库中哪些书籍可用。

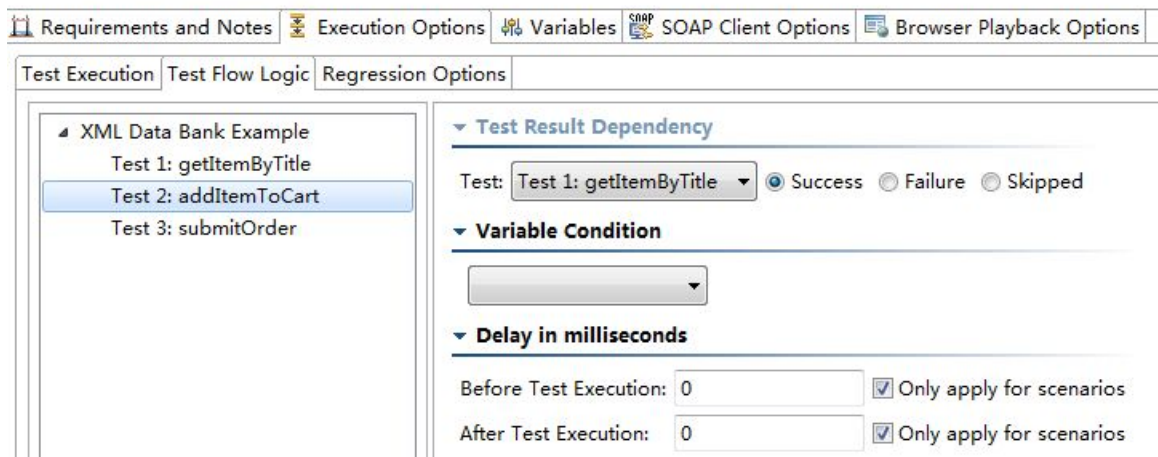
第 3 课：测试套件逻辑——管理测试的依赖关系

通常情况下，你会想要指定你的 SOA 客户端测试用例的执行流程，你可能希望从一个测试传递动态结果到后续的测试，例如只当 Test 1 成功时才运行 Test 2。例如，你可能希望只在 **getItemByTitle** 成功时才创建一个测试场景来运行 **addItemToCart**，并且在当 **addItemToCart** 成功时才运行 **submitOrder**。这有助于减少如果在场景中的一个关键步骤失败时出现太多失败的噪音 - 在我们的例子中就没有必要在检索数据项失败时下订单或者取消它们。此外，它也有助于保持你的系统具有一致状态。例如，如果 **addItemToCart** 在不正确的书籍上执行，那么你可能需要在你的测试环境中返回并且恢复数据库的状态，然后再次一致地运行测试场景。

您可以轻松地在一个测试套件中使用测试套件逻辑创建一个高效的工作流程：

1. 在测试用例浏览器中双击我们在上一课中创建的 **Scenario: XML Data Bank Example** 节点。测试套件选项显示在配置面板的右边。
2. 打开 **Execution Options > Test Flow Logic** 选项卡。
3. 选择 **Test 2: addItemToCart**，切换 **Test Result Dependency** 下拉菜单为 **Test 1: getItemByTitle**。这样就使 Test 2 只在当 Test 1 通过时才执行。
4. 选择 **Test 3: submitOrder**，切换 **Test Result Dependency** 下拉菜单为 **Test 2: addItemToCart**。这样就使 Test 3 只在当 Test 2 通过时才执行。

Name: XML Data Bank Example



- **成功**: 根据在 **Test** 下拉菜单中选择的测试用例执行成功时, 选择是否后续的测试用例应该被运行。如果在 **Test** 下拉菜单中选择的测试用例没有成功, 后续的测试用例将不会执行。
- **失败**: 根据在 **Test** 下拉菜单中选择的测试用例执行失败时, 选择是否后续的测试用例应该被运行。如果在 **Test** 下拉菜单中选择的测试用例没有失败, 后续的测试用例将不会执行。
- **跳过**: 根据在 **Test** 下拉菜单中选择的测试用例执行跳过时, 选择是否后续的测试用例应该被运行。如果在 **Test** 下拉菜单中选择的测试用例没有跳过, 后续的测试用例将不会执行。

第 4 课: 使用自定义脚本扩展 SOAtest 的功能

可能存在一种情况就是你一个测试需求要求你添加自定义的功能或者逻辑到你的测试用例中。SOAtest 允许你轻松地扩展你的测试环境。

使用SOAtest的XML Asserter, 你可以集成使用Jython (支持 Java 的 Python), Java, 或 JavaScript自定义脚本到 SOAtest 中。这意味着几乎任何测试情况都可以轻松地处理。即使这种情况对于SOAtest当前的工具集并不是直接支持。

在这个例子中, 您将使用在以前的例子中的书店服务来创建一个场景测试。在这种场景中你将根据它的标题查找一本书, 然后验证书的价格是一个偶数。

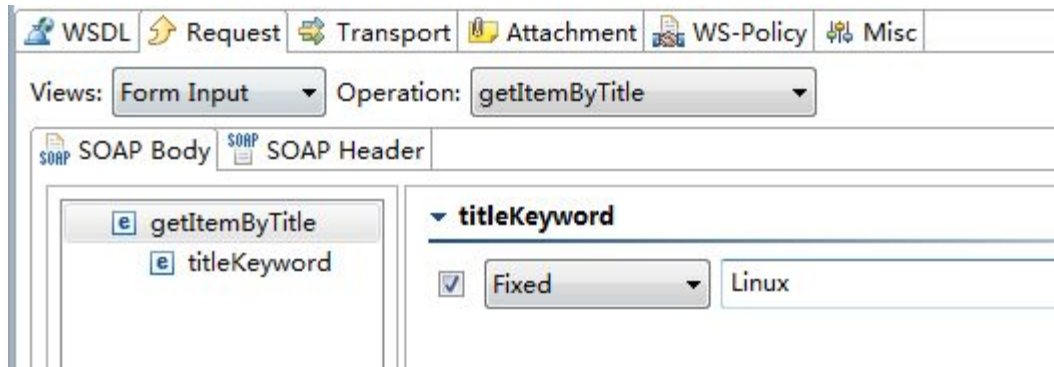
1. 选择 Test Suite: Test Suite 根节点并点击 **Add test suite** 工具栏按钮。



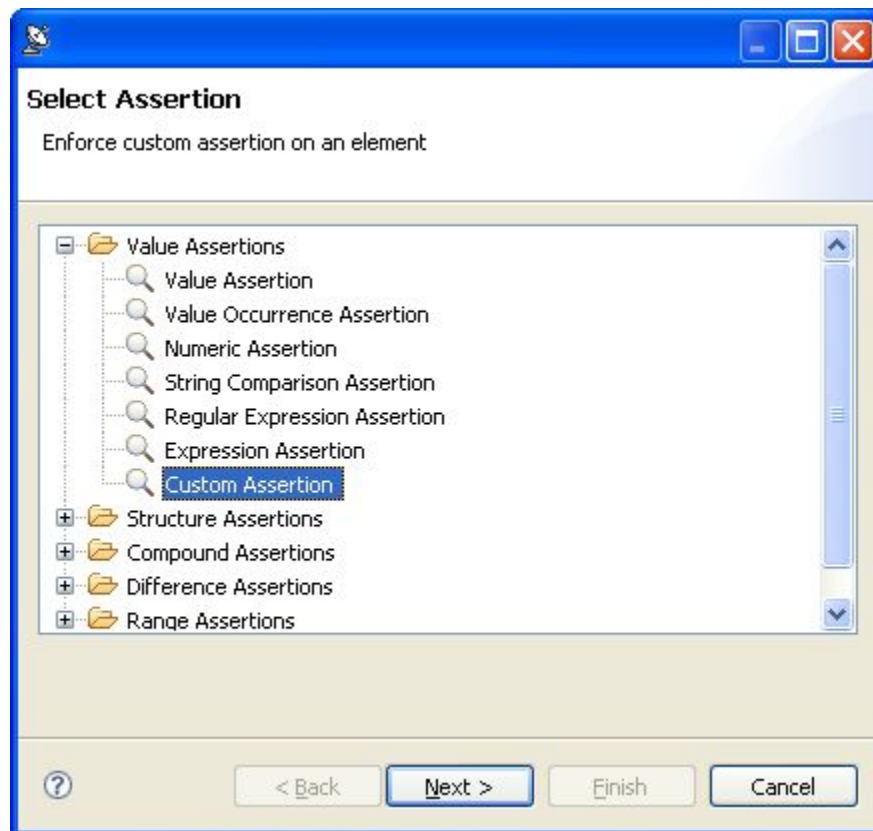
2. 在 Add Test Suite 向导中, 点击 **Empty**, 然后点击 **Finish**
3. 关于新建的 **Test Suite: Test Suite** 配置面板显示出来。在测试配置面板中的 **Name** 字段里输入 Custom Scripting, 然后点击 **Save**.
4. 选择 **Test Suite: Custom Scripting** 节点并点击 **Add test or output** 按钮。



5. 在 Add Test 向导中, 在右边选择 **SOAP Client**, 然后点击 **Finish**. 一个 SOAP 客户端工具就添加到测试套件中。
6. 对于新建的 **Test 1: SOAP Client** 测试的配置面板显示出来。在测试配置面板中的 **Name** 字段里输入 Validate Book ID Value 值。
7. 选择测试配置面板的 **WSDL** 标签, 在 **WSDL URL** 字段中输入 `http://localhost:18080/parabank/services/store-01?wsdl`。
8. 打开 **Request** 标签, 然后从 **Operation** 下拉框中选择 `getItemByTitle`。
9. 在 **titleKeyword** 元素输入框中输入 `Linux` 作为 **Fixed** 值, 然后点击 **Save** 工具栏按钮。



10. 右键单击 **Test 1: Validate Book ID Value** 节点然后选择 **Add Output**.
11. 在 **Add Output** 向导中, 在左边选择 **Response> SOAP Envelope**, 然后在右边选择 **XML Asserter**, 然后单击 **Finish**. 这表示告诉 SOAtest 链接一个 XML Asserter 到 SOAP 客户端的 XML 响应输出。
12. 在 XML Asserter 测试配置面板中打开 **Configuration** 标签, 然后单击 **Add** 按钮。
13. 在选择的断言对话框中, 展开 **Value Assertion** 节点, 选择 **Custom Assertion**, 然后单击 **Next** 按钮。

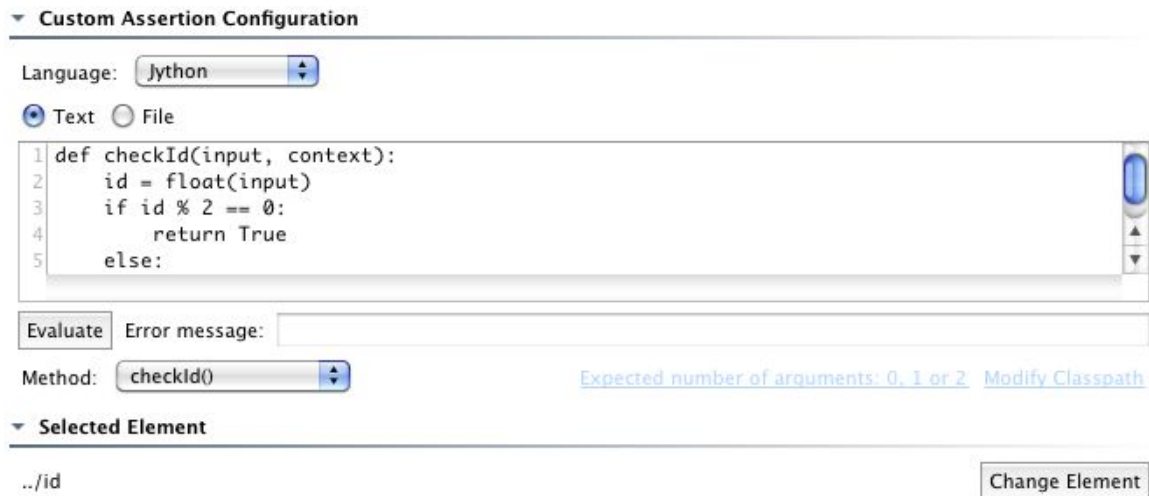


14. 这个 **Custom Assertion** 对话框显示一个XML消息的树形视图, 从这里你可以选择一个单独的值来执行。
15. 在 XML 树形视图中选择 **id** 元素并且单击 **Finish** 按钮。测试配置标签将使用一个方法断言来填充。

16. 在测试配置标签的 **Text** 字段中，输入下面的脚本以确保价格值为偶数：

```
def checkId(input, context):
    id = float(input)
    if id % 2 == 0:
        return True
    else:
        return False
```

请注意这个例子是使用 Python 程序语言写的并且缩进对于脚本能够正常运行非常有用。S0Atest 将协助你使用正确的缩进格式。



17. 从 **Language** 下拉菜单中选择 **Jython**。

如果你喜欢使用 JavaScript 作为脚本语言，你可以使用下面等价的脚本：

```
function checkId(input, context) {
    id = input
    if (id % 2 == 0) {
        return true
    } else {
        return false
    }
}
```

18. 从 **Method** 下拉菜单中选择 **checkId()**。

19. 单击工具栏按钮 **Save**。

20. 选择并运行 **Test 1: Validate Book ID Value** 节点并点击 **Test** 按钮。注意这个测试通过或失败取决于服务是否返回奇数还是偶数。试试其它搜索关键字(例如C++) 来获取不同的 id 值。

你知道吗？

对于 Jython，你可以在你的脚本中使用 Java 对象。你也可以通过从主菜单中选择 **Parasoft> Help> Parasoft S0Atest Extensibility API** 访问 S0Atest 扩展性 API。

练习 1: 创建 Customer #2 包含 XML Data Bank

1. 在你的 SOAtest Training Exercises.tst 文件的根测试套件中创建一个新的测试套件并且命名为 Customer #2.
2. 按照下面的方式自定义这个测试套件:
 - **Test 1: addItemToCart:** 订购 id = 1 的 5 份订单。
 - **Test 2: getItemsInCart:** 获得由 Test 1 订购的书籍列表。
 - **Test 3: submitOrder:** 这会清空待办订单的列表; 在这个操作之后调用 **getItemsInCart** 将导致一个空的响应。
 - 添加一个包含值 1, 2, 3, 4, 5, 6 的表格数据源, 并使用它来参数化 Test 1。
 - 创建回归控制。这个回归控制应该验证当执行 **getItemsInCart** 响应时出现 6 本书, 并且在 Test 3 响应中为 “true”。

提示: 不需要创建一个新的测试套件, 你可以复制和粘贴 Test Suite: Customer #1 然后重命名为 Customer #2 这样就完成了一半工作。

提示: 而不是创建一个新的测试套件, 你可以复制和粘贴测试套件: 客户 #1, 并将其重命名为客户 #2, 你有一半

练习 2: 创建 Customer #3 包含 XML Data Bank

1. 在你的现有项目中创建一个空的测试套件, 然后命名为 Customer #3.
2. 按照下面的方式自定义这个测试套件:
 - **Test 1: getItemByTitle:** 使用这个操作来获取 C++ 书籍的 ID, 使用 XML Data Bank 工具, 你将能够提取 ID 并在 Test 2 中使用它来下一个订单。
 - **Test 2: addItemToCart:** 使用前面测试 getItemByTitle() 的响应获取的 ID 来为 C++ 书籍下一个订单。
 - **Test 3: getItemsInCart:** 注意 C++ 书籍已经订购了, 所以它应该出现在这个测试的响应中。
 - **Test 4: submitOrder:** 这个操作将清空待办订单的列表。
 - 添加一个数据源包含 C++, Productivity, Defect, SQL, 并参数化 Test 1 使用这些值。
 - 配置 **Test 2: addItemToCart** 以便订购的书籍跟在 **Test 1: getItemByTitle** 中的一样。
3. 现在寻找一个方法来自动验证在 Test 2 响应中的订单号值匹配在 Test 3 响应中的订单号值, 而不是通过表面上检查这些值。

提示: 你可能需要使用一个 XML Assertor 或者一个 XML Transformer 和 Diff 工具来实现这个目的。

练习 3: 使用 Calculator 服务的另外一个场景

首先, 创建一个测试场景在 calculator Web 服务上执行下面的步骤:

1. Test 1: Add, 使用 1 和 11 作为输入参数, 然后提取结果为 **sum1**
2. Test 2: Add, 使用 2 和 22 作为输入参数, 然后提取结果为 **sum2**

3. Test 3: Multiply, use **sum1** and **sum2** as input parameters

其次， 使用一个数据源来参数化前面两个测试。在你的数据源中创建一个列包含来自于 **Test 3: Multiply** 期望的响应值，并且配置这个测试让它比较响应值和数据源列。

第 5 章： Web 服务测试相关内容

第 1 课：验证 SOAP Messages 是否遵从 WSDL 使用的 Schemas

SOAtest 拥有针对Schema来验证请求与应答消息的能力。

要创建这样在WSDL中针对schema验证SOAP消息的 .tst 文件，你需要：

1. 右键之前章节中创建的SOAtest训练课程项目，选择 **新建> 测试(.tst) 文件**
2. 给文件命名（比如，额外的材料），然后点击 **下一步**。
3. 选择 **SOA> WSDL**，点击 **下一步**。
4. 从**WSDL URL框**中选择 `http://localhost:18080/parabank/services/store-01?wsdl`
5. 如果没有勾选**Create Functional Tests from the WSDL**多选框，勾选它。
6. 单击按钮**Finish**。新创建的测试套件显示在测试用例浏览器中。

要在WSDL中针对schema进行SOAP消息验证：

1. 右击 **Test 1: add** 节点，并选择**添加输出**。**添加输出**向导将会显示。
2. 在**添加输出** 向导中，在左侧面板选择 **应答> SOAP 封装**，在右侧面板选择 **XML 验证器**，点击 **Finish.**，**SOAP封装应答> XML 验证器** 节点会显示其下的 **Test 1** 节点。
3. 在XML验证器配置面板中，确保勾选框 **针对引用的WSDL中的schema验证**（**Validate against schema referenced in WSDL**）已经被勾选了。
4. 选中 **Test 1** 节点，点击 **测试（Test）** 工具栏按钮。

第 2 课：测试 SOAP Message 对 WS-I Basic Profile 的符合性

WS-I 解析器工具不仅解析WSDLs，同时也对SOAP消息进行解析，并验证他们对WS-I断言的一致性。

要测试SOAP消息对WS-I BP 1.0 的一致性，完成以下内容：

1. 在测试用例浏览器视图中右键 **Test 1: 添加** 节点，从快捷菜单中选择 **添加输出**。**添加输出** 向导将会出现。
2. 在**添加输出**向导中，从左侧面板中选中 **全部（Both）> Traffic 对象**，从右侧面板中选择 **WS-I**，点击**完成**，一个 **Traffic 对象> WS-I** 节点会显示在 **Test 1** 节点下。
3. 在 **WS-I** 配置面板中，确保**Calculator** 已经从其父节点下拉菜单中被选择，并且在名称下拉框中选择为 **ICalculator**。
4. 选择 **Test 1** 节点，点击 **Test** 工具栏按钮。HTTP Traffic 和 WSDL 会被通过 WS-I 工具针对其BP 1.0 的一致性做出检查。

该测试运行完成之后，会有一个错误。假定你不想要修复该问题，相反的，你想要创建一个回归控制来抓取之后的断言失败将其作为网页服务的开发过程。由于测试工具是和SOAtest紧密结合的，你可以轻易的使用一致性报告来创建回归测试。

要从WS-I 工具中创建一个回归控制，完成以下步骤：

1. 右键 **Traffic 对象> WS-I Tool** 节点，并选择 **添加输出**。**添加输出**向导将会显示。

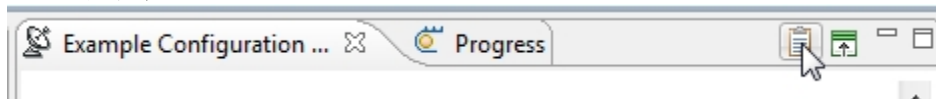
2. 在**添加输出**向导中，在左侧面板中选择**消息**，在右侧面板中选择 **差分 (Diff)** 并点击 **完成**。**消息 > 差分 (Diff)** 节点显示在 **Traffic 对象 > WS-I** 节点之下。
3. 右击 **Test 1** 节点，并在快捷菜单中选择 **创建/更新 回归控制**。
4. 选择 **创建回归控制**，点击 **完成**
5. 双击 **消息 > 差分** 节点，并注意到 断言失败是被设置在右侧界面面板中的 **Text** 栏中。
6. 选中 **Test 1** 节点，并点击 **Test** 工具栏按钮。测试成功了，因为在WS-I 断言结果中没有回归控制。

第3课：从 SOAtest 界面生成 HTML 报告

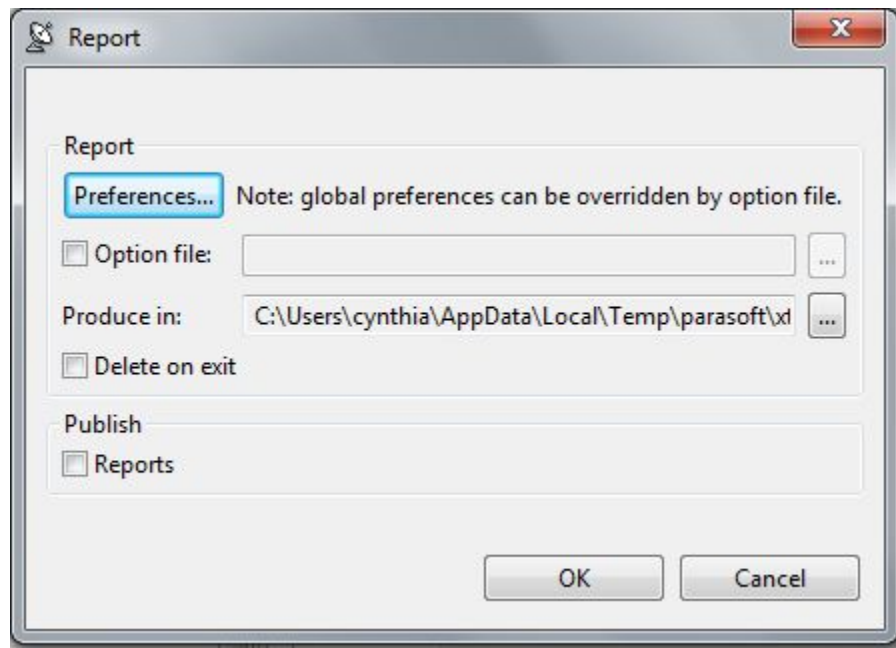
对用户来说可以对团队成员分享他们的测试结果是非常重要的。SOAtest可以自动对团队成员发布测试结果并分配任务。另外的，你也可以从ad-hoc来共享测试执行情况。

如何开始：

1. 在测试用例浏览器视图中，选择在之前课程中创建的 **SearchOrderConfirm** 场景。
2. 运行该场景。
3. 点击**生成报告**工具按钮。

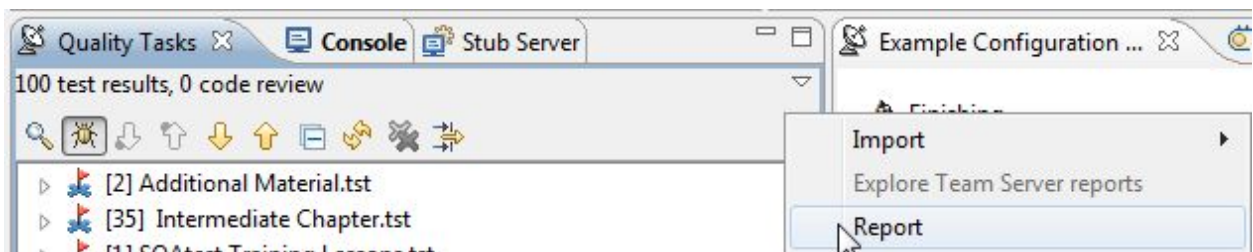


报告对话框会打开并询问是否定义报告首选项，指定报告保存的位置，指定报告是否要发布到Team Server上。



4. 点击 [...] 按钮，选择一个你想保存报告的位置。
5. 点击 **OK** 来生成报告。HTML报告将会在你的SOAtest中被生成。
6. 来到你所指定生成报告的位置。你会看到HTML格式，XML格式的报告以及一些生成的图片。

注意：你也可以再质量任务视图中通过点击小三角中的下拉窗口，点击**Report**按钮完成报告的生成。



第 6 章：端到端测试

S0Atest 的各种功能可以被结合在一起对一个应用执行端到端测试。在该类测试中，所有应用组件都会被覆盖：用户通过浏览器访问的网页前端界面；用于与其他应用进行数据交互的后端服务；用于通讯数据存储的数据库；和其他组件，包括中间件，第三方应用程序等。

仅针对于前台测试的典型方法是，使用屏幕扫描工具来录制用户在页面上执行的不同动作，或者应用的一个部分。然而，系统中的集成错误是很难通过这种手段侦测的，这样就迫使测试应用需要用到端到端测试的技术。将所有部件联合测试是更具有健壮性的测试方法，当然它可以通过 S0Atest 来自动化的完成。

在一下的示例中，我们将会使用名为 ParaBank 的 Demo 应用来展示如何在一个测试场景中测试不同部件和不同技术层面。Parabank 是一个针对客户的商业银行网站。它包括 Web UI、SOAP、RESTful 服务和一个 HSLDB 数据库，并采用不同的技术。

准备工作：设置 Demo 应用 (Parabank)

1. 根据第二章--环境准备内容搭建 ParaBank 服务



注意图中左上角的图案。在练习中，点击该图标会来到“Administration”页面，在此可以配置应用的数据状态。

在练习中的任何时候如果你需要重置 Parabank 保存的账户和客户数据，在数据库标题下点击 **Clean** 按钮。

Administration

Database



完成准备工作，创建你的账户：

1. 导航到 <http://localhost:18080/parabank>
2. 在左侧栏中点击 **Register**。
3. 填写个人信息进行注册，然后点击最底下的 **REGISTER** 按钮。

第 1 课：测试数据库

Parasoft SOAtest 可以用来通过发送SQL语句来和数据库进行交互并验证返回的结果集。这个功能通常用来帮助用户准备适当的条件优先级来测试（建立测试），恢复到之前的条件（卸载测试），并作为端到端测试场景的一部分来验证数据。在本课中，我们会使用HSQLDB数据库包括在ParaBank Demo 应用中来执行SOAtest的数据库验证功能。

我们将以全局配置的模式来添加数据库配置集。全局数据库配置允许你使用多种工具来一次性输入信息。同时，如果你之后需要更新信息，你可以在其中仅修改一处并自动化修改应用的所有适当的工具和设置。

在开始之前，SOAtest需要知道在哪里发现ParaBank HSQLDB数据库的JDBC驱动（在hsqldb.jar文件中），要这么做：

1. 进入 **Parasoft > Preferences...**，选择 **Parasoft > JDBC Drivers**。
2. 在该首选项窗口中，点击 **New** 并定位到 **hsqldb.jar** 文件。文件可以在如下位置找到：

```
[SOAtest 9.6 install
directory]\eclipse\plugins\com.parasoft.xtest.libs.web_
9.0.0.20100729\root\tomcat\webapps\parabank\WEB-INF\lib
```

3. 点击 **Apply**，**OK**。

要开始测试：

1. 在测试用例浏览器中选择 **ParaBankQATest.tst > Test Suite: Test Suite** 节点然后点击添加测试套件工具按钮。



2. 在添加测试套件向导中，选择 **Empty**，然后点击 **Finish** 按钮。
3. 双击新创建的测试套件，重命名为 **DB Example**，然后保存关闭该测试套件编辑器。

4. 在测试用例浏览器视图中，选择 **ParaBankQATest.tst > Test Suite: DB Example** 节点，

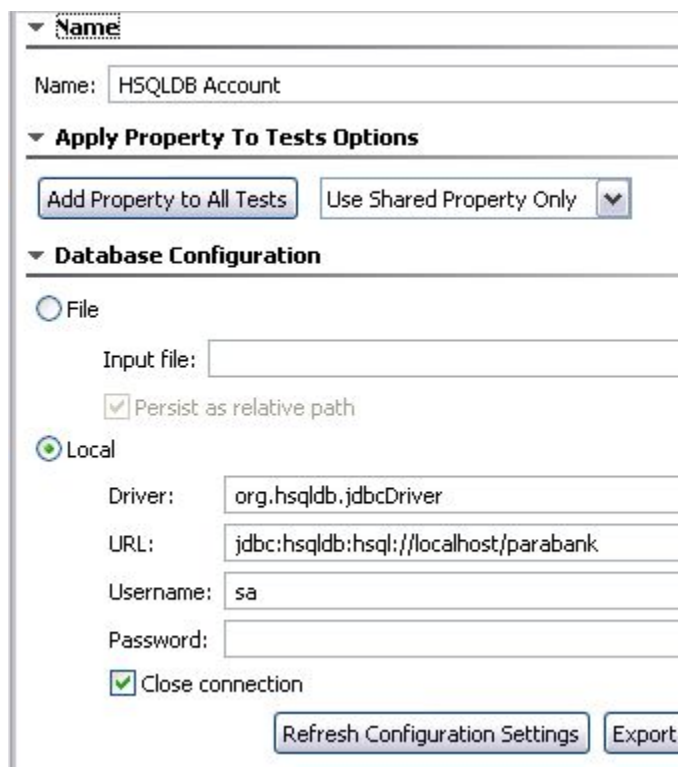
点击，点击 **Add property** 工具按钮。



5. 选择 **Global Property > Database Account**，然后点击 **Finish**。

6. 以如下方式完成 **Database Properties** 配置：

- 在 **Name** 栏中输入 **HSQldb** 账户。
- 在 **Driver** 栏中输入 **org.hsqldb.jdbcDriver**
- 在 **URL** 栏中输入 **jdbc:hsqldb:hsq://localhost/parabank**
- 在 **Username** 栏中输入 **sa**。
- 在 **Password** 栏中留白。



7. 保存并关闭编辑器。
8. 在测试用例浏览器中选择 **ParaBankQATest.tst > Test Suite: DB Example** 节点，点击 **Add test or output** 工具按钮。



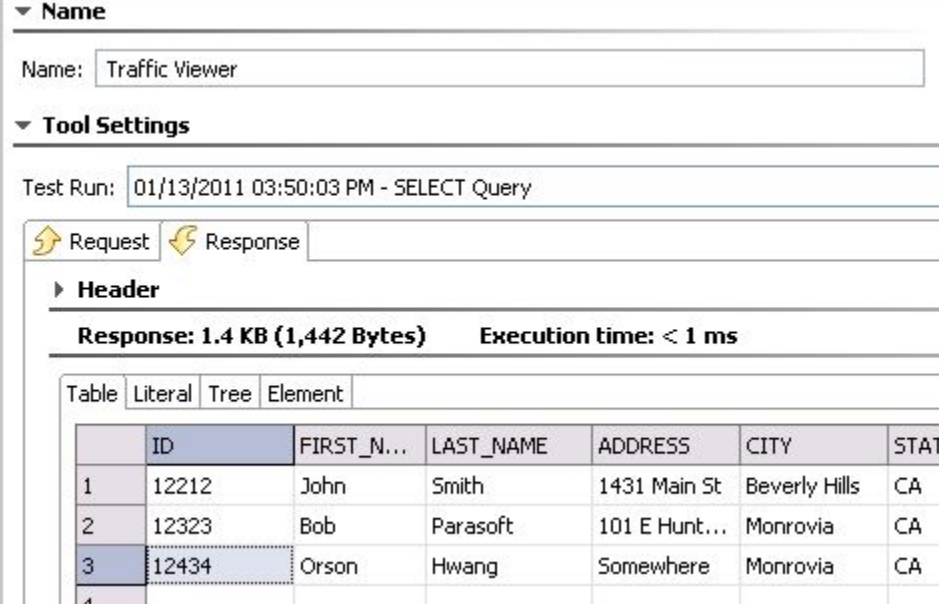
9. 在左侧面板中选择 **Standard Test**，在右侧选择 **DB Tool**，然后点击 **Finish** 按钮。新的 **DB Tool** 会被添加它的编辑器会被打开。

10. 完成 DB Tool的配置方法如下：

- a. 在Name栏中输入 SELECT 查询。
- b. 在 Connection 面板中，选择Use Shared Property 按钮。
- c. 在Shared Property 下拉菜单中，确保HSQLDB账户已经被选择。
- d. 打开 *SQL 查询* 面板添加查询语句：SELECT * from Customer
- e. 保存工具编辑器。

你现在可以使用该工具来直接查询Parabank的HSQLDB数据库的账户表。这允许你从前台测试中来证明后端数据库的正确性。

- f. 运行该工具。查看交易并注意你账户的ID值。



The screenshot shows the DB Tool configuration window. Under the 'Name' section, the name is 'Traffic Viewer'. Under 'Tool Settings', the 'Test Run' is '01/13/2011 03:50:03 PM - SELECT Query'. The 'Request' and 'Response' tabs are visible, with the 'Response' tab selected. The response is displayed as a table with the following data:

Response: 1.4 KB (1,442 Bytes) Execution time: < 1 ms						
Table	Literal	Tree	Element			
	ID	FIRST_N...	LAST_NAME	ADDRESS	CITY	STAT
1	12212	John	Smith	1431 Main St	Beverly Hills	CA
2	12323	Bob	Parasoft	101 E Hunt...	Monrovia	CA
3	12434	Orson	Hwang	Somewhere	Monrovia	CA
4						

练习 1: 拓展数据库测试

修改在之前课程中创建的资产来包含如下内容：

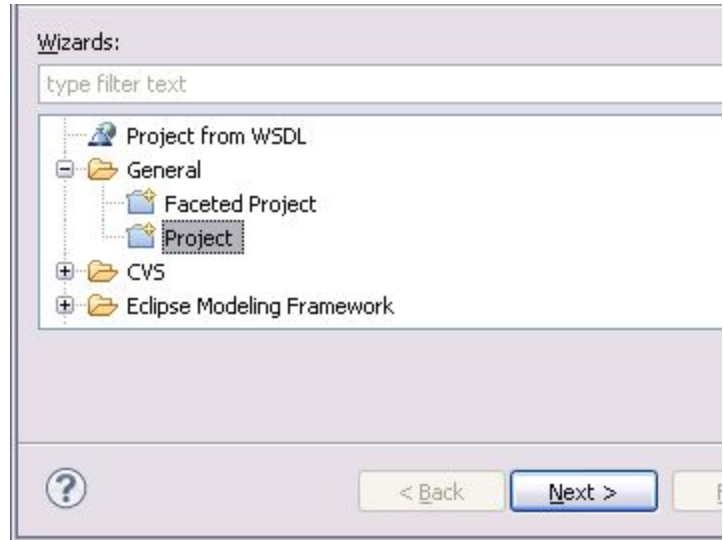
1. 对于你自己的账户（在“准备工作”中创建的），在Customer数据库表中创建一个XML验证器来检查你的客户ID号的**FIRST_NAME**值是否等于你之前创建的账户。
 - 提示：当XML工具（比如 XML 断言器）被链接到DB Tool，其结果会自动转换为XML使链接工具可以使用。
2. 从之前课程 10f 中查询并保存你的唯一ID号，然后将其传入到 **getCustomer** SOAP 请求中
 - 提示：使用XML Data Bank 和在如下WSDL位置中定义的**getCustomer** 操作：
<http://localhost:18080/parabank/services/ParaBank?wsdl>

第2课：测试 RESTful 服务

Parasoft SOAtest 可以通过REST客户端工具完成测试 RESTful 服务。在下一个示例中，你会使用 ParaBank应用中的REST API来验证账户拥有者信息，而不通过网页用户界面。

如何开始：

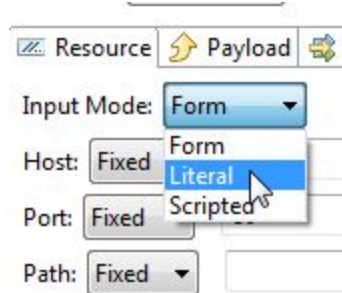
1. 通过 **File > New > Project...**，选择 **General > Project**，**Next**的方式来创建一个新的项目目录
2. 在**Project Name:** 栏旁边，输入名称 **ParaBank Tests**，点击 **Finish**.
3. 右键项目文件夹，创建新的 .tst 文件，选择 **Add New > Test (.tst) File**.
 - a. 将其命名为 **ParabankQATest** 点击 **Next**.
 - b. 选择 **Empty** 在 **Test Suite Type Selection**之下
4. 在测试用例浏览器中选择 **ParabankQATest.tst > Test Suite: Test Suite:** **Test Suite**节点，点击 **Add test suite** 工具栏按钮。



5. 在Add Test Suite 向导中，选择**Empty**，点击 **Finish**.
6. 双击新创建的测试套件，修改名称为 **REST Example**，保存并关闭测试用例编辑器
7. 在测试用例浏览器中选择 **ParaBankQATest.tst > Test Suite: Test Suite > Test Suite: REST Example** 节点，点击**Add test or output** 工具栏按钮。



8. 选择左边的**Standard Test**，在选择右边的**REST Client**，点击 **Finish** 按钮。新的REST客户端就会被添加，其编辑器也会被打开。
9. 重命名工具 **REST Call Acct Info**.
10. 修改 **Input Mode** 为 **Literal**.



Resource Payload

Input Mode: **Form**

Host: Fixed **Form**

Port: Fixed **Literal**

Path: Fixed **Scripted**

大多数 RESTful 服务是GET基础的服务，有一个URL和查询组成；他们通常以一个XML或者JSON进行应答。

11. 为 RESTful 服务提供请求，通过输入 URL

`http://localhost:8080/parabank/services/bank/customers/12212`

这构成了一个对 Parabank的 RESTful 服务的调用来获得指定用户ID为 12212 的个人数据。



▼ Name

Name: REST Client

▼ Header

Resource: `http://localhost:8080/parabank/services/bank/customers/12212`

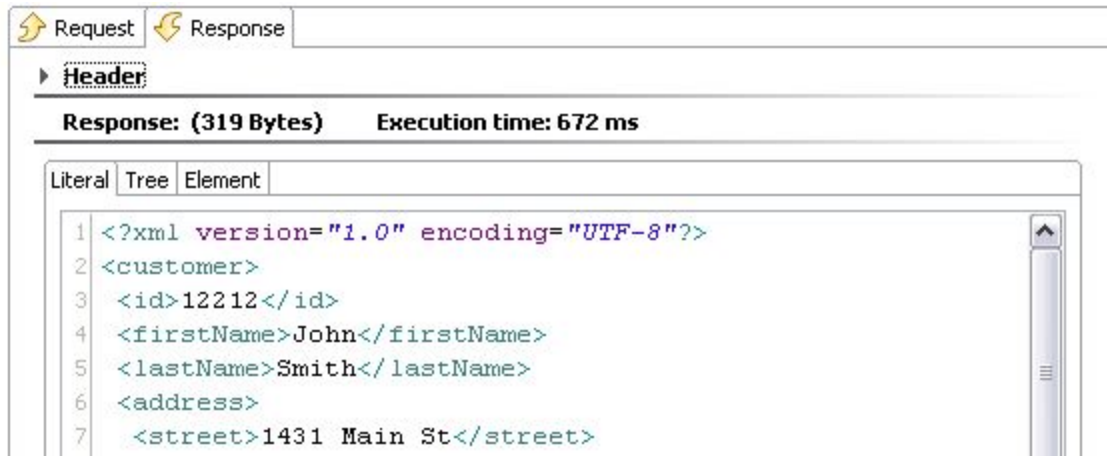
Method: GET

Resource Payload HTTP Options Success Criteria

Input Mode: **Literal**

Fixed `http://localhost:8080/parabank/services/bank/customers/12212`

12. 修改 Input Mode 为 Form 。注意 S0Atest 已经将URL分离为各自的部分
13. 保存REST 客户端编辑器
14. 选择 REST 客户端，并点击 **Run Tests**。在链式的 **Traffic Viewer** 工具中， 验证你所得到的应答是否包含 客户名称， 地址， 电话和社保账号。



第 3 课: 验证 Web 页面

练习 2: 拓展 RESTful 测试

修改之前课程中创建的资产，包含如下内容：

1. 创建一个验证指定检查 `firstName = John` 并且 `lastName = Smith`.
2. 数据驱动 REST 请求使其不仅包含 12212，同时包含在之前准备工作步骤中创建的你自己的客户账号。
 - 提示：使用测试步骤 “测试数据库” 的 10f 来验证你的客户 ID
 - 提示：你将同时会需要从之前的数据库测试练习中更新验证器。

练习 3: 创建一个端到端测试场景

1. 清除 Parabank 数据库，通过点击在 Administration 页面上的 **CLEAN** 按钮（查阅本章节开始处的准备工作最后一步）。
2. 创建一个场景在网页界面中来执行并验证以下的用户用例执行（在其他级别上的验证）：
 - a. 以用户 John Smith 登陆（登陆名 = john，密码 = demo）。
 - b. 确保账户余额等于 \$5022.93。
 - c. 打开一个新账户。
 - d. 在账户概况页面中确保新账户的余额为 \$100.00.
 - i. 同时通过 Web 服务层面确保该账户 (REST 和 SOAP)。
 - ii. 最后，确保在数据库中的账户情况
 - e. 请求一个 \$1000 贷款，保证金 \$200，按照 step 2b，从老账户取款相当于保证金的数量（提示：使用 “fromAccountId” 元素来从保证金中指定账户 #）。
 - f. 存款 \$500 到新账户，如之前的 step 2e（此处使用网络服务层面）
 - i. 从前台层面验证。
 - ii. 从网络服务层面验证 (REST 和 SOAP)。
 - iii. 从数据库验证。

- g. 登出.
- h. 从 Administration 页面清除数据。

第 7 章：压力测试

一旦在SOAtest中创建了功能性测试，则下一个步骤是用Parasoft Load Test的压力测试. 压力测试模拟大量使用的条件，揭露仅在这些条件中出现的漏洞. 你可以压力测试Web，SOA和结合的端对端测试（通过web服务器，JMS，网页接口，数据库等扩展到消息层之外的测试场景）。

另外，Parasoft Load Test包括执行Parasoft压力测试组件API的组件的任意组价的压力测试组织框架；例如，它可以容许JUnits的性能测试和并发测试，或执行Parasoft组件API的带有轻量级基于Socket组件的压力测试. 这使压力测试专用化，并为组织结构执行性能验证过程中面对的各种独特的复杂性所改变。

如果你在之前版本的SOAtest 和WebKing中有配置的压力测试，则它们可以被导入并完全支持。

Parasoft Load Test使你可以完全控制各方面的压力测试，包括以下领域：

- **使用多台机器(联网)：**你可以在你的网络上使用多台机器，生成大量的负载，与单个机器生成的相比数量更多. 单击压力测试窗口中的 *MACHINES* 文件夹，开发的GUI. 对每一个机器，你可以选择高通量模式。高通量模式通过禁用某个响应处理操作使用相同的硬件生成更高的负载强度。
- **用户配置文件：**创建用户配置文件可以直接相关压力测试到功能性测试. 这意味着一旦创建了功能性测试，就不需要进一步的步骤在负载下运行它. 在压力测试窗口中双击*PROFILES* 文件，查看被创建的每个配置文件。
- **自定义场景：**压力测试提供四种默认压力测试场景（Bell, Buffer Test, Linear Increase, Steady Load）或允许创建自定义场景. 可以通过创建这些场景模拟在正常使用中可能发生的真实生活场景. 单击 *SCENARIOS* 文件夹，查看为本示例提供的场景。
- **监视器：**可以添加监视器到负责测试以监控压力测试过程中各种系统资源. 右键单击 *MONITORS* 文件夹，查看可以添加的监视器. Load Test支持SNMP，Windows Perfmon和JMX 监视器。

虽然培训指南中没有展示，压力测试还可以执行以下内容：

- 模仿多个 web 服务使用模式
通过在网络中使用其它计算机生成大量的负载
- 从监视器搜集数据 (SNMP, Windows, JMS, rstat, Oracle 以及其他)
- 对比多个压力测试报告，并以图表形式表示出来，使压力测试运行之间的性能变化形象化
- 生成执行时间和请求/响应大小的直方图

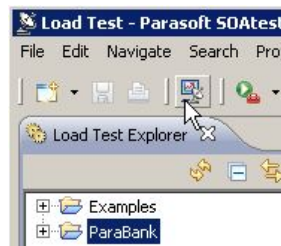
典型 workflow

关于包含 web 功能性测试的测试场景

压力测试任意包含web功能测试的SOAtest功能性测试套件的典型 workflow：

- 1 使用浏览器记录web场景.
- 2 用期望的验证和提取自定义测试场景
- 3 切换到SOAtest中的压力测试透视图.

- 4 在压力测试资源管理器里选择合适的测试套件，单击 **VALIDATE FOR LOAD TESTING** 工具按钮。



- 如果成功执行，则可能不需要进一步的场景配置/调整，且当前你的场景已经为Parasoft Load Test的压力测试准备好。
- 如果执行失败，你可以使用SOAtest配置测试套件，为压力测试做好准备。例如，在某个浏览器场景内用户动作之间传递的各种URL参数值可能需要配置。更多详情参见SOAtest用户指南的压力测试章节。

- 5 启动 Parasoft压力测试，并为在SOAtest中创建的 .tst 文件创建新的压力测试场景。

相同的工作流也可以应用到端对端测试SOAtest测试场景(通过Web服务，JMS，web接口，数据库等在信息层以外扩展的测试场景)。Parasoft压力测试将会将并发性从驱动SOAtest功能性测试套件中任何测试中移除。

关于所有其它 SOAtest 测试场景

压力测试任意不包括Web功能性测试的SOAtest功能性测试套件的典型工作流如下：

- 1 在SOAtest中定义并自定义该场景。
- 2 启动 Parasoft压力测试，为你在SOAtest中创建的 .tst 文件创建一个新的压力测试场景。

第 1 课：创建一个网络应用功能性测试

在这些压力测试课堂中，我们将使用Parasoft所有的名为“Parabank, ”的站点，该站点为模拟银行web应用程序。

记录一个新的 Web 场景

如果要记录我们压力测试中需要使用的Web功能性测试：

- 1 在SOAtest中创建一个空的项目，然后选择 **FILE> NEW> 新建.TST文件>**
- 2 输入**PARABANK** 作为项目名称, 然后单击 **NEXT**.
- 2 选择 **Record new functional test**, 然后单击 **Next**.
- 3 按以下步骤完成Record Web Functional Tests向导页面：
 - a 在**TEST SUITE NAME** 字段中输入 **PARABANK FUNCTIONAL TEST**.
 - b 在**START RECORDING FROM** 字段中输入**HTTP://LOCALHOST:8080/PARABANK/INDEX.HTM**
 - c 单击**FINISH** 按钮. 测试将会启动，打开一个浏览器窗口。

从起始位置记录
从一个特定的地址开始记录一个新的功能性测试

测试套件名称
ParaBank Functional Test

开始录制于
http://localhost:8080/parabank/index.htm

测试类型
☒ 生成功能性测试
☐ 自动生成回应桩
☐ 生成非同步请求测试

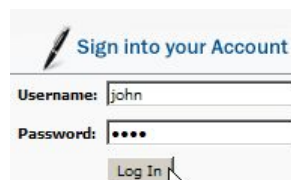
录制使用
☒ Internet Explorer
☐ Firefox
☐ Chrome

Internet Explorer 路径
 C:\Program Files\Internet Explorer\IEXPLORE.EXE 浏览(B)...
 选择的 Internet Explorer 版本: 8.0.6001.18702

可维护性的测试报告
☒ 生成可维护性的测试报告

4 在打开的浏览器窗口中, 执行以下动作:

- a 输入 *JOHN* 为用户名, 输入 *DEMO* 为密码, 然后单击 *LOG IN*.



Sign into your Account

Username: john

Password:

Log In

- b 单击你在账户列表看到的第一个账户连接.

Accounts Overview			Sign Off
Account	Balance*	Available Amount	
12456	\$-4556.55	\$0.00	

- c 在页面的右上方的单击 *SIGN OFF* 链接.



d 关闭浏览器. 这将结束记录会话

为压力测试验证场景

如果要验证测试是否为压力测试准备好, 则:

- 1 选择 **WINDOW> OPEN PERSPECTIVE> OTHER> LOAD TEST** 切换SOAtest到压力测试透视图。左边会打开一个压力测试资源管理器. 这与Test Case Explorer类似, 但是双击每个浏览器测试案例会打开一个特殊的编辑器, 用于准备压力测试的测试案例步骤. 该编辑器显示了各种浏览器加载网页做出的URL请求, 以及这些请求中任意参数化的值。
- 2 选择根**PARABANK** 测试案例节点, 并单击工具栏中的 **VALIDATE FOR LOAD TESTING** 按钮. 例如, 该测试会成功, 表明功能性web场景已经为压力测试准备好。



现在你可以像第 3 课中描述的那样配置并执行压力测试。

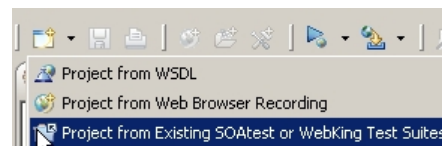
第 2 课: 创建一个服务功能性测试

为了示范压力测试, 我们将使用SOAtest个别指导中的测试套件. 如果你的SOAtest工作空间没有指导课程, 或者不可用, 则按以下步骤新建:

- 1 在SOAtest中选择 **FILE> NEW> PROJECT FROM EXISTING SOATEST OR WEBKING TEST SUITES**.

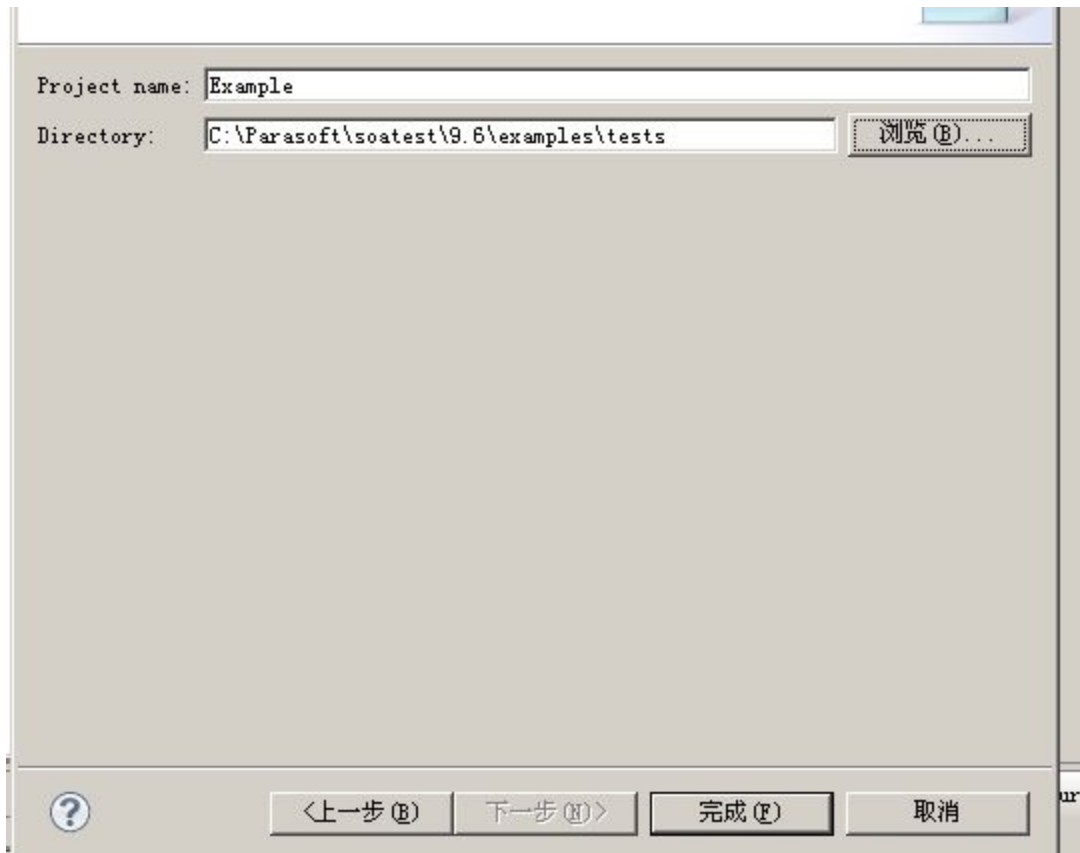


- 或者, 从**新建**工具栏按钮 (左上角) 选择下拉菜单命令。



- 2 在 **PROJECT NAME** 字段中输入 **EXAMPLE**.
- 3 单击 **BROWSE** 导航到 **[SOATEST_INSTALLATION_DIRECTORY]/EXAMPLES/TESTS** 指定项目测试套件的位置

4 单击**FINISH**。示例项目将被添加到Test Case Explorer。它将会包含多个测试 (.tst files)。



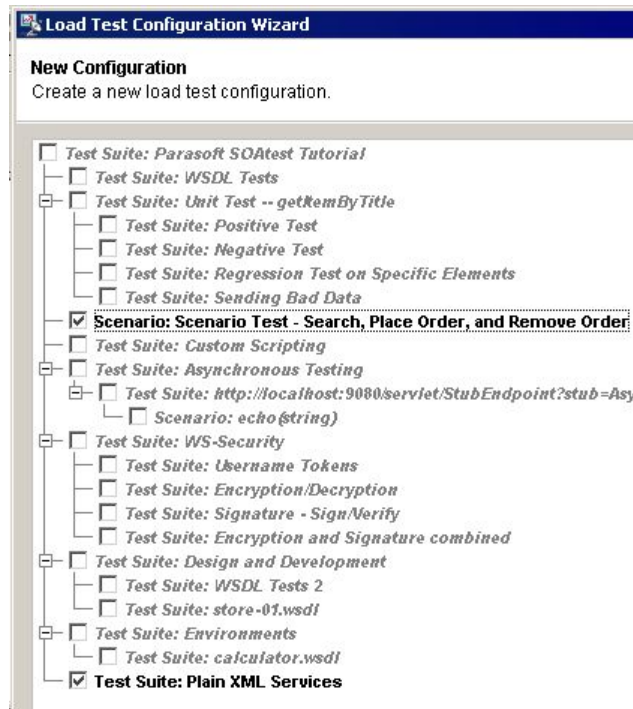
现在你可以如第 3 课中描述的那样配置并执行压力测试。

第 3 课:创建并执行一个压力测试 (为 Web/Service 功能性测试)

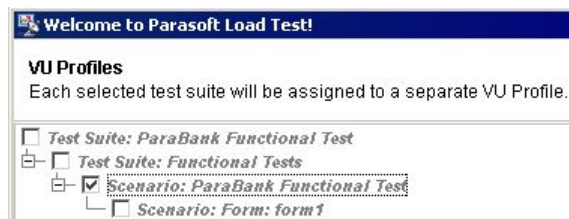
如果要为上个课程中提到的Web或者Service测试定义并运行一个压力测试，则：

- 1 打开Parasoft Load Test:
 - **WINDOWS:** 选择 *START> PROGRAMS> PARASOFT> LOAD TEST*.
 - **LINUX OR SOLARIS:** 改变当前目录为 *LOADTEST* 目录，然后在命令提示符中输入以下命令：
./LOADTEST
- 2 在欢迎向导界面中选择 *NEW PROJECT*，然后单击**NEXT**.
- 3 选择**SOATEST**，然后单击**NEXT**.
- 4 做下列几件事之一：
 - **SERVICES:** 浏览到**SOATESTTUTORIAL.TST**，单击 **NEXT**。如果你重新开始创建，则会在你的工作空间中。否则， *[PARASOFT LOAD TEST INSTALL DIR]/EXAMPLES/TESTS/SOATESTTUTORIAL.TST* 中打开样本文件
 - **WEB:** 浏览到 *PARABANK FUNCTIONAL TEST.TST* (在SOAtest工作空间中)，然后单击**NEXT**.
- 5 在树中选择以下，并单击**NEXT**:

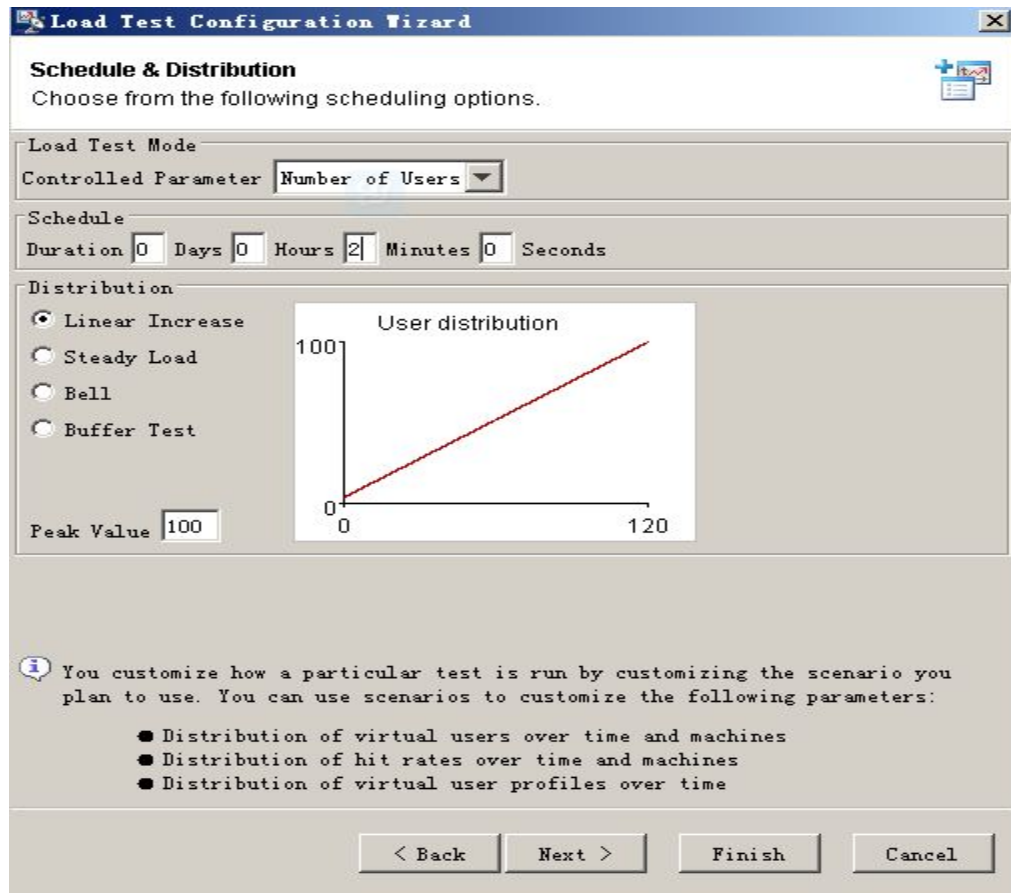
- **SERVICES:** “Scenario: Scenario Test - Search, Place Order, and remove Order” , “Test Suite: Plain XML Services”



- **WEB:** Scenario: ParaBank Functional Test.

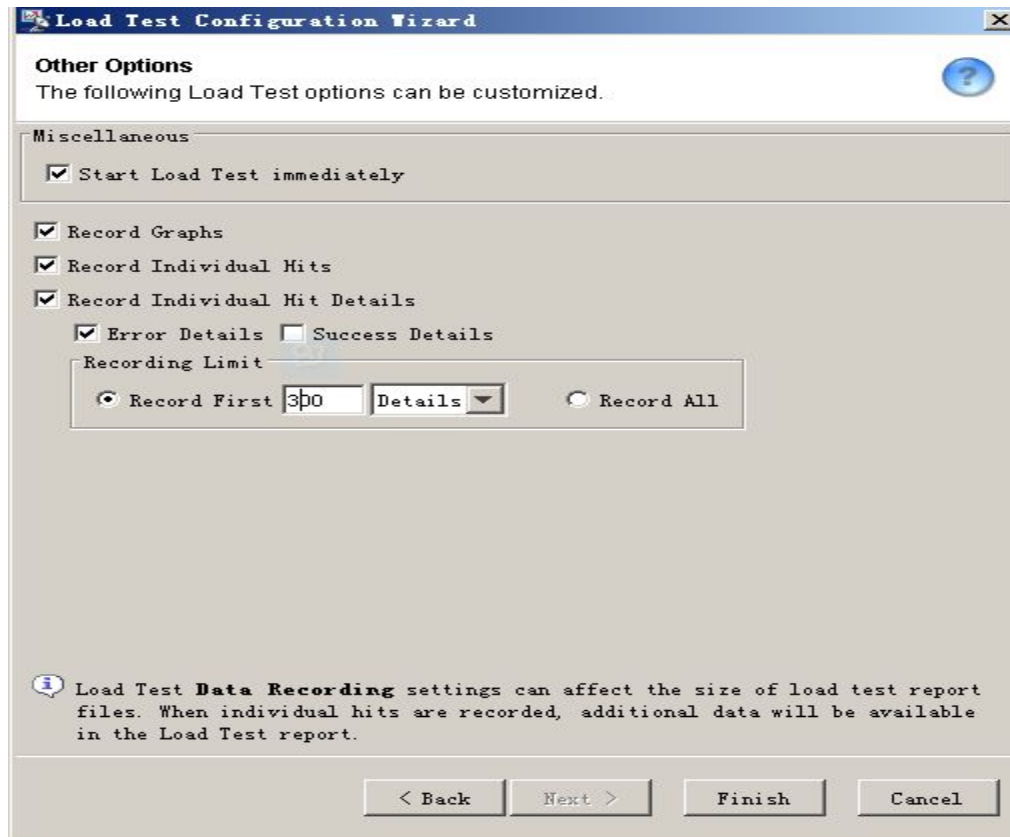


- 7 在Schedule & Distribution 面板中, 输入 2 分钟作为持续时间, 选择 *LINEAR INCREASE* 作为分布规律, 单击 *NEXT*, 直到出现 *OTHER OPTIONS* 面板.

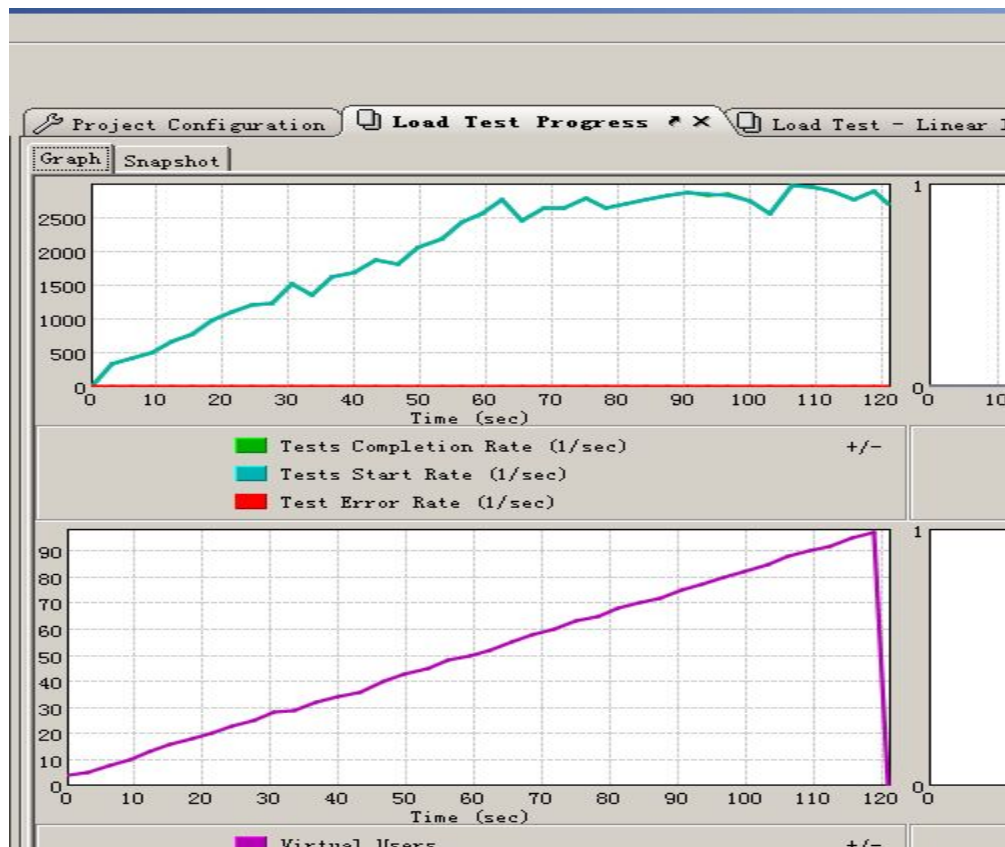


- 为了选择Linear Increase选项, 你必须有合适的Load Testing许可证. 如果你许可证上没有没有可用的Virtual Users, 你只能生成Steady Load场景.

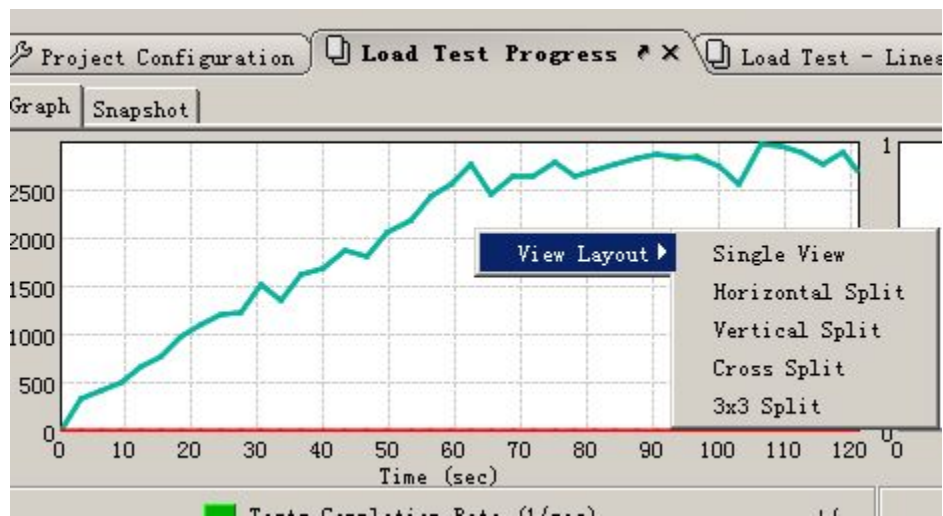
8 在其它选项面板中激活 *START LOAD TEST IMMEDIATELY* 和 *RECORD INDIVIDUAL HITS*, 然后单击 *FINISH*



Load Test将会启动指定的压力测试. 左边的GUI面板会出现一个新的Load Tests 标签, Load Test 进度面板会出现Graph标签.



你可以通过在Graphs标签中右键单击选择其中一个可用的布局选项来改变表格的数量 and 布局。



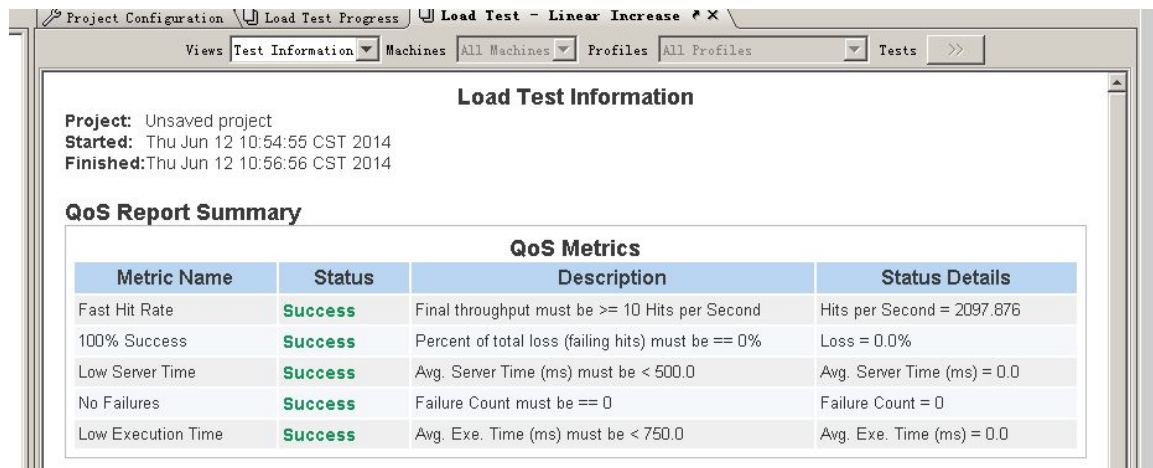
压力测试可以在任意时间被中止,但是我们将让它在监控和操纵测试正在进行的细节的同时运行完整的 2 分钟。

查看Load Test Progress标签,注意 **Graph** 标签显示以下内容:

- 按照本课程的步骤 7 中选择的**Linear Increase**场景, Virtual Users曲线稳定上升,呈直线型方式
- 如果**TESTS COMPLETED** 和 **TESTS STARTED** 曲线彼此想匹配,则表明测试正迅速被服务(如,响应被迅速接收到).如果这些曲线之间有很大的区别,则测试执行的时间会更长。

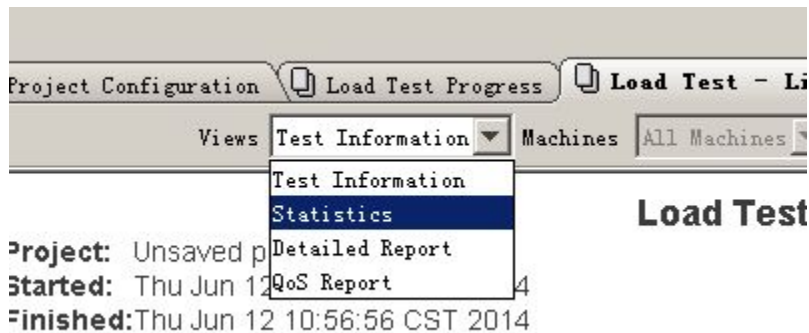
同时注意 **SNAPSHOT** 选项卡显示了当前活跃的虚拟用户和它们调用的操作. 在测试执行期间, 此标签内的信息每 3 秒更新一次.

压力测试完成后, Results 面板将会显示 Test Information 总结, 包含项目的名称, 压力测试开始和结束的时间, 选择的场景以及任何机器和配置文件.



你可以选择查看压力测试的不同统计报告. 完成以下步骤以查看详细的压力测试统计:

1. 从 Results 面板中的 Views 菜单选择 Statistics.



查看统计报告时, **OUTPUT TYPES** 菜单可用. **OUTPUT TYPES** 菜单决定显示输出报告的类型. 该菜单中的两种类型的报告都可用, 每个报告显示不同列的信息. 报告类型之间的区别有:

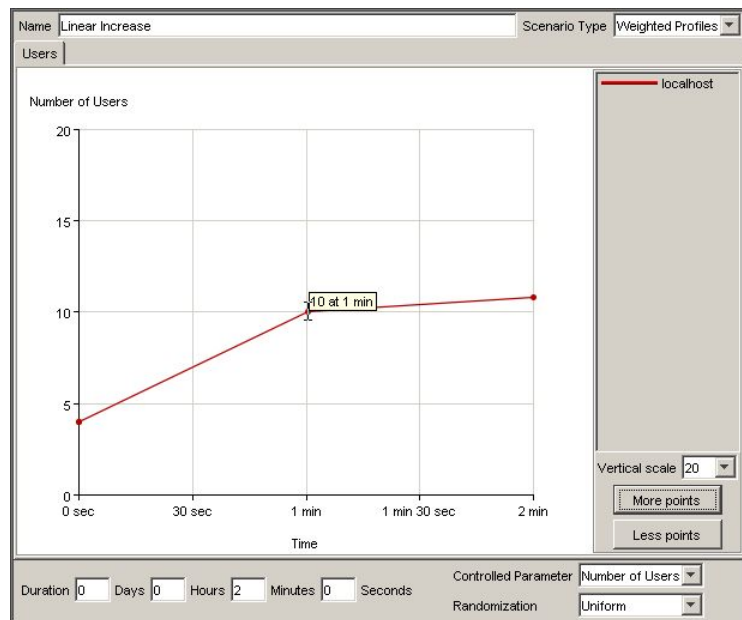
- **GENERIC REPORTS:** 包含测试套件名称, 测试指数, 测试名称, 最小/最大/平均时间 (ms), 运行次数和失败次数.
- **NETWORK CLIENT REPORT:** 包含所有通用报告中的信息, 以及最小/最大/平均 Ping (ms), 最小/最大/平均请求大小 (bytes), 最小/最大/平均响应大小 (bytes) 最小/最大/平均总大小 (bytes).

第 4 课: 自定义压力测试配置文件和场景

你可以通过自定义你计划使用的配置文件和场景自定义特殊的压力测试运行方式. 你可以决定压力测试持续的时间长度, 虚拟用户的分布, 随着时间推移的点击率和机器, 以及随着时间推移用户配置文件的分布.

完成以下步骤以自定义压力测试:

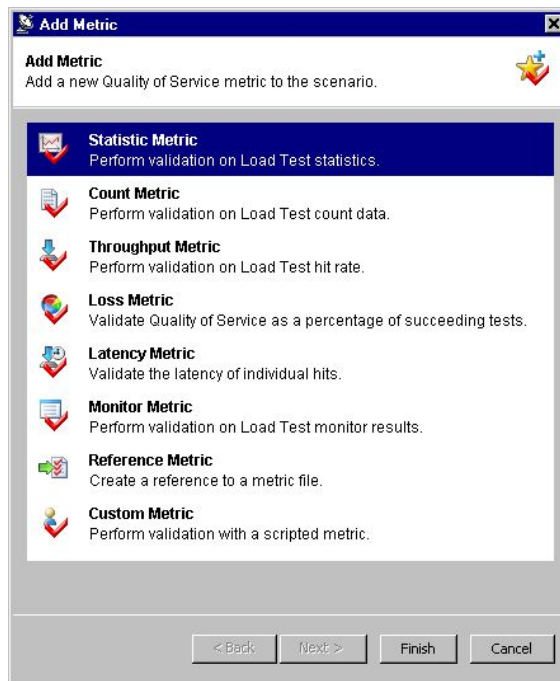
- 1 双击 **LOAD TESTS** 标签中的**PROFILES** 文件夹, 并选择其中一个可用的测试套件节点. 配置面板显示在右边.
- 2 在面板的底部, 将延时 **VALUE** 改为 **3** 秒. 这将会模仿用户在决定订购书籍之前犹豫的行为.
- 3 在**SCENARIOS** 节点下方选择**LINEAR INCREASE**. Linear Increase 场景控制Results面板中的显示, 且User图表显示本地服务器曲线.
- 4 拖放本地服务器曲线的端点到 2 分钟 10 个用户的坐标点.
- 5 从 **VERTICAL SCALE** 下拉菜单中选择 **20**.
- 6 单击**MORE POINTS** 按钮. 本地服务器行中心将会出现一个点.
- 7 单击并拖动新的点到 1 分钟 10 个用户的坐标点处.



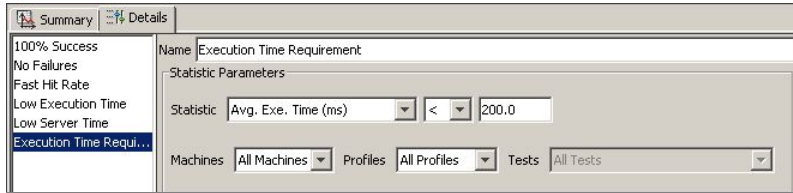
- 8 扩展 **LINEAR INCREASE** 节点, 并选择下方的 **QOS** 节点. 显示**SUMMARY** 和 **DETAILS** 标签.
- 9 选择**DETAILS** 标签并单击**NEW** 按钮.



ADD METRIC 向导显示.



- 1 选择 *STATISTIC METRIC* 并单击 *FINISH*.
- 1 在打开的度量配置面板中, 在 *NAME* 字段中输入 *EXECUTION TIME REQUIREMENT*.
- 1 在右边的GUI *STATISTIC*下拉菜单中, 选择 *AVG. EXE. TIME (MS)* 和小于号 (<), 并在文本字段中输入 200.



如果测量的执行时间大于 200 毫秒，则会导致压力测试结果为“失败”。

1 在 **SCENARIOS** 分支下方选择 **LINEAR INCREASE** 节点并单击 **LOAD TEST** 工具栏按钮。Load Test 将会启动自定义的压力测试，**GRAPH** 标签出现在 GUI 面板的右边。

1 等待（2 分钟）压力测试完成。当运行压力测试时，你可以通过选择合适的复选框查看 **GRAPH** 标签里的各种参数。

第 5 课：创建自定义压力测试组件

Parasoft Load Test 功能可以通过创建自定义测试组件扩展。Parasoft Load Test 充当压力测试组件或压力测试 beans 的容器。一个压力测试组件是一个定义虚拟用户在压力测试过程中行为的功能性核心。压力测试容器为组件提供以下服务：

- 创建，调用，处理虚拟用户（和压力测试组件）使其适合压力测试场景。
- 从本地和远程机器收集组件报告，并聚集它们，成为一个合并的压力测试报告。
- 在压力测试开始前序列化组件到远程机器上。
- 为在压力测试项目文件中保存组件配置提供方法。
- 通过 GUI 为组件配置提供方法。

一个压力测试组件可以通过扩展位于 `com.parasoft.simulator.api` 程序包中的压力测试组件 Java API 的 `SimRunnable` 类被创建。`SimRunnable` 代表模拟器可运行，模拟器是压力测试容器的名称，模拟多个虚拟用户行为。

建立组件示例

组件示例位于你的 Load Test 安装的 **EXAMPLES\DEV\COMPONENTS** 文件夹中。`simple-runnable-source.jar` 和 `socket-runnable-source.jar` 档案包含源，配置和为本指南中描述的组件建立文件。你将需要解压这些档案到 Load Test 组件项目目录（必须手动创建）。

创建一个 Load Test 组件项目目录

如果要在 Load Test 安装之外创建一个 Load Test 组件项目目录，则：

1. 复制 `simple-runnable-source.jar` 到组件项目目录，并解压组件源档案。
2. 解压 `socket-runnable-source.jar` 到 `SocketRunnable` 目录（必须手动创建）。每个源目录包含 `component-build.xml` ANT 构建脚本。
3. 打开该文件，确保 `load.test.install.root` 属性指向 SOAtest/Load Test 安装目录。
4. 改变 `load.test.jar.path` 属性为指向 `loadtest.jar` 文件（在 `eclipse\plugins\com.parasoft.xtest.libs.web_[version_number]\root of the SOAtest/Load Test` 安装目录中。

创建 SimpleRunnable Load Test 组件

如果要在 Eclipse 中创建一个 SimpleRunnable 压力测试组件项目，则：

1. 打开新项目向导, 选择 **Java Project**, 然后单击**Next**.
2. 命名该项目为 `SimpleRunnable`.
3. 选择 **Create project from existing source**, 选择 `SimpleRunnable`目录, 然后单击**Next**.
4. 在**Source** 标签中, 确保默认输出文件夹设置为`SimpleRunnable/target/ classes`.
5. 打开 **Libraries**标签, 单击 **Add External Jar** 并添加 `parasoft.jar` (在 `S0Atest/Load Test`安装目录的 `eclipse\plugins\com.parasoft.xtest.libs.web_[version]\root`目录中).
6. 在Eclipse中, 右键单击 `component-build.xml` 并选择 **Run As> Ant Build** 创建组件 `.jar` 文件.

注意:

- 该组件应该用Java 1.5 JRE创建—用JRE 1.5.0_18 更好.
- 每个案例项目中`component-build.xml` ANT 脚本的默认目标创建了一个组件`jar`档案文件, 可以部署到`Load Test`作为自定义或内置组件. 如果你使用Eclipse编译组件项目, 从`component-build.xml` 脚本的`jar`目标中移除编译目标依赖性.

创建一个 SocketRunnable 压力测试组件

`SocketRunnable`组件首先扩展 `SimpleRunnable` component. `Compile SimpleRunnable`将 `main.jar` 从 `SimpleRunnable`项目的根目录转移到`SocketRunnable`项目的`lib`目录中.

如果在Eclipse中新建一个`SocketRunnable`压力测试组件项目, 则:

1. 打开新项目向导, 选择**Java Project**, 单击 **Next**.
2. 选择 **Create project from existing source**, 选择`SimpleRunnable`目录, 单击 **Next**.
3. 在**Source** 标签中, 确保默认输出文件夹设置为`SimpleRunnable/target/ classes`.
4. 打开 **Libraries** 标签, 单击 **Add External Jar** 并添加 `parasoft.jar` (在 `S0Atest/Load Test` 安装目录的 `eclipse\plugins\com.parasoft.xtest.libs.web_[version]\root`目录中).
5. 在 **Libraries** 标签中, 确保 `lib/jtidy-r820.jar` 和`lib/main.jar` 在项目类路径上.
6. 在Eclipse中, 右键单击`component-build.xml` 并选择 **Run As> Ant Build** 创建组件 `.jar` 文件.

加载/部署示例组件到压力测试

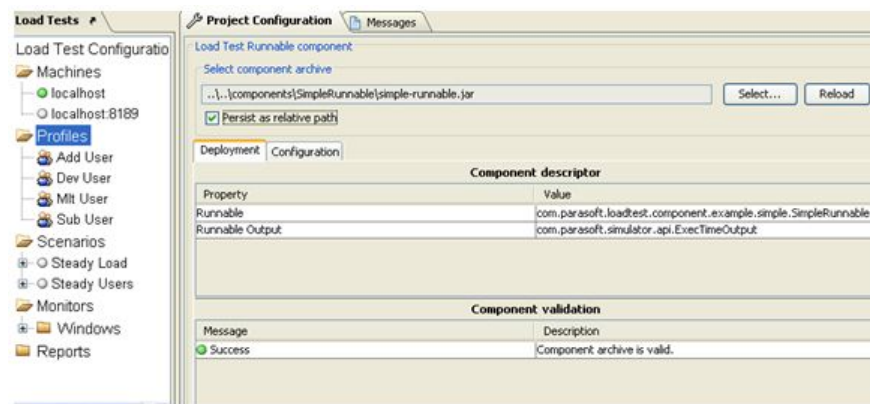
一个组件可以被加载到`Load Test`作为外部的档案文件或作为内置组件被部署.

从外部档案文件加载一个组件

如果要从外部档案文件加载一个组件, 则:

- 1 在`Load Test` GUI中, 选择`Load Test Configuration`树的**PROFILES** 节点.
- 2 在 `Project Configuration` 视图中, 单击 **SELECT**.
- 3 在`Select a Component Archive`会话框中, 选择**LOCAL OPTION**, 单击 **NEXT**, 然后选择你的压力测试组件档案文件.

- 4 单击 **FINISH**。将会出现一个组件部署视图，如下。



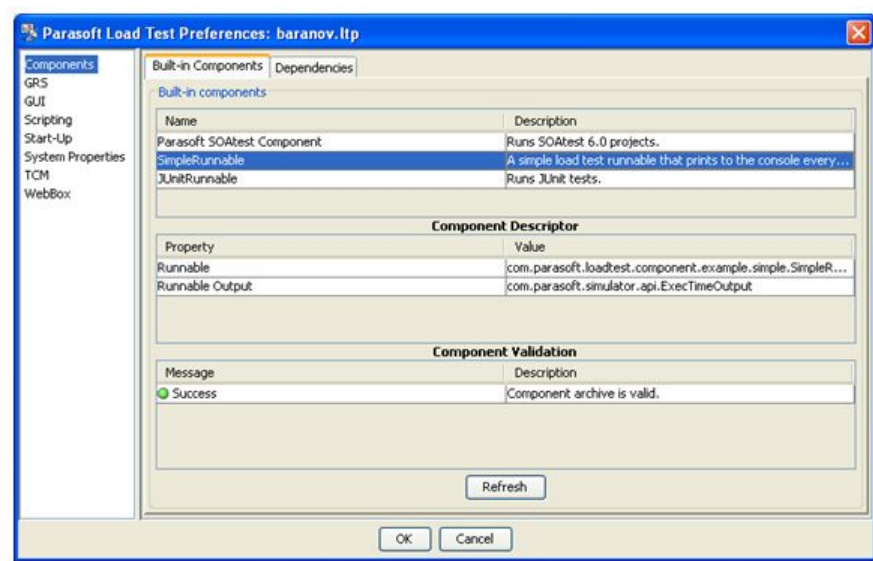
如果组件验证失败，错误信息将会在Deployment视图的下方的表格中显示出来。

将组件档案文件作为内置组件部署

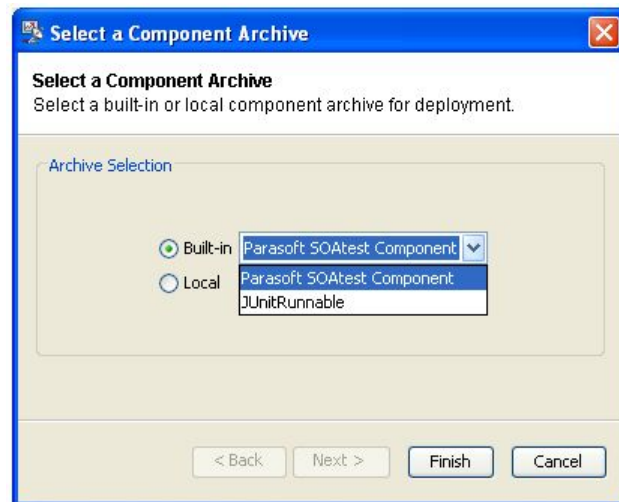
如果要将组件档案文件作为内置组件部署，则：

1. 将组件.jar档案文件放到SOAtest/Load Test 安装的
`ECLIPSE\PLUGINS\COM.PARASOFT.XTEST.LIBS.WEB_[VERSION] \ROOT\LT-COMPONENTS\BUILT-IN` 文件夹中。
2. 重启Load Test.
3. 从**FILE** 菜单中, 选择 **CUSTOMIZE PREFERENCES**. 你将会在内置组件部署视图中看见组件部署详情，如下。

3



如果内置组件被成功部署，则它在Built-in下拉列表的Select Component Archive对话框中可用，如下：



使用压力测试组件 API

通过选择Load Test应用程序菜单栏的**HELP**菜单选择**API > COMPONENT API**子菜单，Load Test组件API的JavaDoc文档可用。

分析 SimpleRunnable 组件示例

本章节检测SimpleRunnable Load Test组件。

SimpleRunnable是一个模拟可配置的模拟时间测试执行和产生一个可配置频率的错误的基本组件。这是为了让你熟悉压力测试组件API和组件档案创建和部署的。压力测试组件的主要方法是运行(**PrivateContext**上下文)。模拟用户行为的组件行为应该在这个方法内被执行。组件运行方法被一个特定的配置文件的**Virtual User**对象调用。虚拟用户被压力测试应用程序根据压力测试场景调用。

其余的 SimpleRunnable方法执行容器要求的各种服务。下面，我们将描述最重要的组件方法。关于余下方法的详情，参见SimpleRunnable类和SimRunnable的 JavaDoc的源代码。

以下是组件最重要方法的描述：

- `SimRunnableOutput run(PrivateContext context)` - 组件的核心功能方法。各种容易-指定变量通过PrivateContext作为变元传递到该方法可供使用。参见API 文档中的查看可用语境变量列表。
- `String canRun()` - 此方法被容器调用以确保组件被正确配置且可以被调用。检查所有组件成员变量是否被正确初始化。如果没有，则返回一个错误描述。如果组件准备完毕，则返回空值。返回非空值错误描述将会防止Load Test启动，错误字符串将会显示。
- `boolean done()` - 该方法被容器调用该类的run()方法之后调用，以决定组件是否完成它的活动。一个组件可以模拟多个步骤的用户行为，每个步骤之后紧跟着虚拟用户判断时间（就像压力测试配置文件中配置的一样）。在这种情况下，run()方法将被调用多次，直到done()返回为真。返回为假将会导致结构框架保持该组件类的实例从属的虚拟用户，并再次调用run()方法。返回为真则使该组件类的实例从属的虚拟用户从虚拟用户池中被移除。

重要 - 释放虚拟用户

你最后需要从done()方法返回真,以便使虚拟用户从池中移除.如果压力测试场景需要,则一旦虚拟用户被从池中移除,会有相同配置文件的新的虚拟用户替代.释放虚拟用户很重要,因为它使结构框架遵循预定虚拟用户的起伏或者每秒的点击率.

- SimRunnable prototype() - 这是一个工厂模式.每次虚拟用户被容器创建时,它都会被分配一个新的组件类的实例.当一个组件被部署到容器内,该容器创建该组件类的实例,充当创建在压力测试过程中分配给虚拟用户的组件类对象的原型.框架调用组件类的prototype()方法获取组件原型的副本,使组件决定如何得到该副本.当执行该方法时,确保所有必要的组件成员变量被复制到新建的组件实例中.你可以通过值或者引用复制组件类成员:
- 当通过引用复制组件类成员时,注意这些类成员应该被充分保护,防止并行存取或修改.如果出现这种问题,可以考虑通过值复制(深层复制).
- 当通过值复制组件类成员时(深层复制)时,记住,数百个,可能数千个虚拟用户(和该类的实例可以在)Load Test应用程序运行压力测试的时候在其中共存.应用程序内虚拟用户的数量取决于Load Test模式和压力测试场景中预定的值.一个压力测试组件有太多的内存占用,会导致应用程序内存溢出.在这种情况下,考虑通过引用复制成员变量.
- 无效 onDeploy(DeploymentContext context) - 该方法在组件档案被部署到容器时被容器调用.不同的部署-指定变量通过DeploymentContext以变元的形式传递到该方法可用.该方法的SimpleRunnable'执行展示了如何访问组件部署包含的配置(及其它)文件.
- 无效的writeExternal()和无效 readExternal() - 转移组件状态到远程机器的方法.

注意SimpleRunnable类的writeExternal()和readExternal()方法只有框架实现,不能编写或读取任何数据.这些方法被用于转移组件状态到远程Load Test机器. SimpleRunnable组件不需要这种状态转移,因为包含在组件档案的组件外部配置文件被用于本地和远程机器上的组件配置.我们将再不同组件示例中查看writeExternal()和readExternal()的实现.

接下来,我们看一下SimpleRunnable的外部配置文件. SimpleRunnable组件的源文件夹包含两个组件配置文件: LongExecTime.cfg 和ZeroExecTime.cfg.其中之一被包含在组件档案中(参见component-build.xml ANT脚本).

当组件档案文件被部署到容器中时,该档案被解压,档案内容通过DeploymentContext的DEPLOYMENT_BASE属性对组件可用. SimpleRunnable类的onDeploy方法的实现展示.cfg组件文件是如何被定为并解析以初始化SimpleRunnableConfiguration的实例的.如果一个压力测试配置包含远程机器,组件档案则在压力测试开始之前每个组件被部署之后被转移到每个远程机器.

部署进程完成后, onDeploy方法被调用. onDeploy方法从包含在组件档案的.cfg文件中初始化组件.仅在组件档案没有在远程安装中出现,或它在主机和远程机器上的总和检查不相同,它会被序列化到远程机器上.这确保了组件档案转移到远程机器的行为仅在需要的时候发生.

组建档案描述符

lt-jar.xml 组件档案描述符必须被包干在每个组件档案的根目录中.该描述符必须包含来自SimRunnable的主要组件类的类名称和来自SimRunnableOutput的组件输出的类名称t.参见组件档案描述符文件格式的示例包含组件的 lt-jar.xml文件.

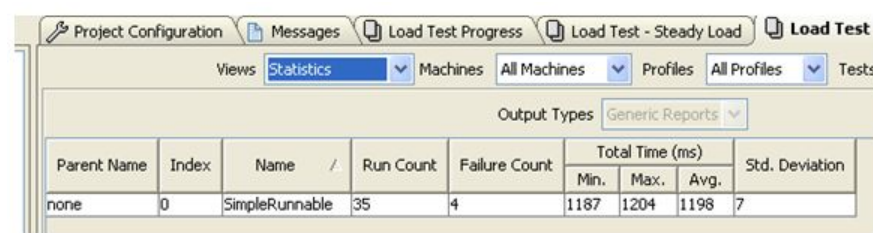
用 SimpleRunnable 组件运行压力测试

一旦你将SimpleRunnable组件部署到Load Test并熟悉SimpleRunnable源代码和组件档案结构, 你可以对组件档案配置进行修改, 重新运行压力测试, 并查看修改后组件配置的效果. 例如, 你可以按照以后步骤, 以改变该组件模拟的测试执行时间:

- 1 修改.cfg 文件中的execTimeMs参数.
- 2 运行组件档案ANT构建脚本.
- 3 在Load Test组件视图中, 单击**RELOAD** 按键.



- 4 重新运行压力测试. 注意压力测试报告的统计部分的测试执行时间遵循配置的执行时间持续时间.



Parent Name	Index	Name	Run Count	Failure Count	Total Time (ms)			Std. Deviation
					Min.	Max.	Avg.	
none	0	SimpleRunnable	35	4	1187	1204	1198	7

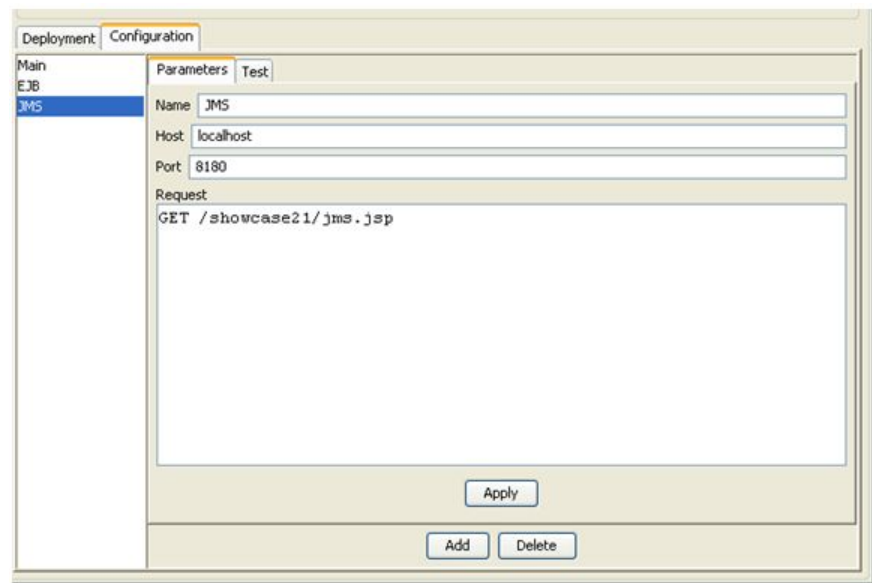
执行统计反应了组件配置文件的execTimeMs 参数改变为 1200.

分析 SocketRunnable 组件示例

本章节检测SocketRunnable组件. SocketRunnable类扩展SimpleRunnable 且示范以下:

- 在应用程序日志中记录内部错误.
- 创建组件配置GUI面板.
- 创建允许概要文件特定的组件配置压力测试配置文件GUI面板.
- 在压力测试项目中保存组件配置.
- 转移组件配置到远程机器上.
- 使用 jar库扩展组件功能.
- 局部化组件资源.

SocketRunnable组件使用SocketTest类实例建立一个到指定主机和端口的连接, 发送自定义请求并使用Java Sockets接收响应. 从属于该组件的SocketTest实例在SocketRunnable组件视图中是可配置的.



在应用程序日志上记录内部错误

你可能立刻需要该日志，所以从SocketRunnable构造函数中的引擎获取。

```
ENGINE ENGINE = (ENGINE)CONTEXT.GET(PRIVATECONTEXT.ENGINE);  
COMPONENTLOG LOG = ENGINE.GETCOMPONENTLOG();
```

创建组件配置 GUI 面板

一个组件可以通过执行SimRunnableConfigViewProvider接口提供它自定义配置视图到容器. 该容器会在以上所示的Load Test Runnable组件面板的配置标签显示该视图. 该容器调用SimRunnableConfigViewProvider接口的getConfigView()方法已获得组件的配置视图. 在本例中，SocketRunnableView类执行组件配置视图. SocketRunnableView类使用位于in com.parasoft.loadtest.component.example.socket.test.view 程序包的SocketTest*View类作为单个SocketTest实例配置.

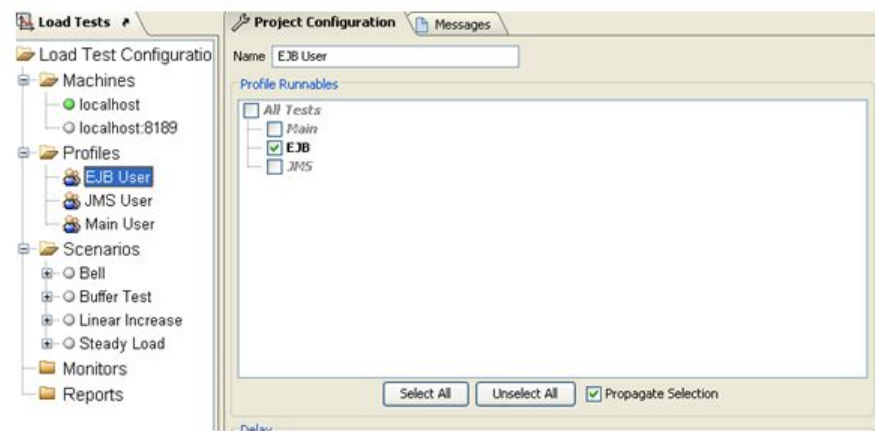
为概要指定组件配置创建压力测试概述 GUI 面板

压力测试组件用于执行压力测试概要指定行为执行. 执行 ProfileRunnable或 SubProfileRunnable接口的类充当压力测试组件的概要文件-特殊子类. 这些类执行组件的概要相关行为.

SocketRunnable示例中有两个执行ProfileRunnable接口的类: SocketProfileRunnable和SocketTestWrapper. SocketProfileRunnable充当根配置可执行文件;它是SocketTestWrappers的容器. SocketRunnable类保留了SocketProfileRunnable的实例, 所有配置的SocketTestWrappers都保存在此. SocketRunnable还保留了一个SocketTestWrappers的矢量; 这个矢量代表着指定压力测试配置文件的一组 ProfileRunnables. 压力测试配置中的每个配置文件保留这组件类的复件。这保证了配置指定组件配置不冲突，也不会与彼此覆写。

SocketRunnable的populateSubProfileRunnables()方法基于从概要指定PrivateContext中获取的ProfileRunnableMap填充selectedProfileRunnables变量. ProfileRunnableMap包含压力测试概要指定的ProfileRunnable选集.

为了让你选择压力测试概要指定的ProfileRunnables, SocketRunnable组件如下执行ProfileRunnableViewProvider接口.



SocketRunnable使用API的ProfileRunnableTreeView.

在压力测试项目保存组件配置

压力测试组件可以通过执行SaveablePropertiesProvider接口保存它的配置到压力测试项目. 组件配置由SimRunnableProperties类表现, SimRunnableProperties类保存了组件属性名称/项值对.

组件属性必须以字符串值的形式保存, 不依赖于组件类. 这确保了一个项目可以被加上Load Test应用程序的意思, 即使组件没有被部署. 同时, 这使项目可以保存不同组件类型配置. 该容器使用SimRunnableProperties类的标记字段返回合适的属性实例到组件. 该标签值应该被组件开发人员设置. SoatestRunnable使用“SOCKET_RUNNABLE_SAVEABLE_PROPS_V.1”字符串作为它可救得属性标签.

如果保存你的组件配置需要比一组名称/值属性对更复杂的数据结构, 你可以使用XML作为提替代的选择. 序列化你的组件配置, 使其为XM表达, 并保存XML字符串到SimRunnableProperties实例的单个名称/值对. 关于如何使用XML保存并还原组件配置, 请参见SocketRunnableXMLSerializer.

转移一个组件配置到远程机器

一个可以从压力测试项目文件中读取或可以被你或你的团队修改的压力测试配置必须被转移到远程压力测试机器. 组件状态转移必须在组件类执行的ParasoftExternalizable接口的readExternal/writeExternal方法内执行. 因为Since the configurable part of the SocketRunnable组件的可配置部分由类型SocketProfileRunnable的rootProfileRunnable成员变量加以表示, 它不得被转移到远程机器上. SocketProfileRunnable类序列化到远程机器在该类的readFrom/writeOn和readExternal/writeExternal方法的实现中被演示. 或者, 到远程机器的序列化可以再次使用保存组件配置到压力测试项目的代码. 关于SocketRunnable例子, SocketRunnableXMLSerializer类功能性可以被再次使用以序列化/反序列化组件配置到远程机器上, 作为一个XML字符串.

使用Jar库扩展组件功能性

你的组件代码可以使用你选择的jar Java库. SocketRunnable示例代码使用Jtidy库分析它可以接受以响应网页请求的HTML代码 (参见SocketTest类的checkTidy方法). 组件依赖jar文件必须包含在组件jar档案lib文件夹内 (参见component-build.xml ant脚本创建的socket.jar组件档案).

定位组件资源

你可以使用LocalizationUtil API类定位组件字符串. The class of the SocketRunnable示例的Localization类加载合适的资源束并包裹该LocalizationUtil方法以便于访问.

第 6 课: 查看报告

一旦压力测试被完成, 收集的数据将会被分析, 以便查看应用程序/服务在加载下是如何执行的. 压力测试使你能配置并生成压力测试报告.

在这一课中, 你将会学到如何查看详细的报告, 并且如何生成HTML报告.

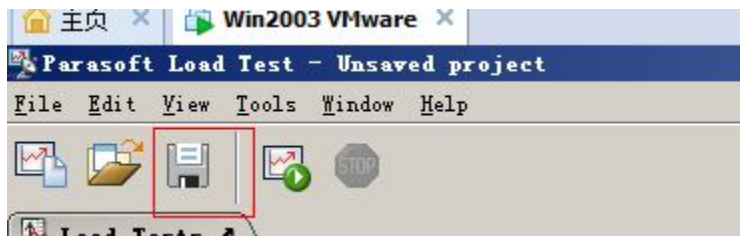
完成以下步骤以查看详细的报告:

1. 从Results 面板的Views 下拉菜单中选择Detailed Report. Graph标签显示了各种参数. 在图表内, 你可以执行以下内容:
 - 如果要查看图表和视图中不同的参数, 选择位于面板地步的期望的参数复选框.
 - 如果要查看对数尺度上的多个参数, 选择LOG SCALE 复选框. 对数尺度使你看清相同图表上多个曲线的形状 (即使显示的值与另一个值相差很远).
 - 如果要从整个图表中查看错误, 在图表的任意处右键单击, 从快捷菜单中选择SHOW ERRORS.
 - 如果要查看详细报告的 HTML 版本, 在图表的任意处右键单击, 从快捷菜单中选择VIEW REPORT. (如果要在查看之前配置报告, 选择VIEW REPORT CONFIGURATION.)

如你所见, 你可以从DETAILED REPORT的GRAPH标签中收集压力测试的工作详情. 通过HISTOGRAM 和 TABLE 标签还可以获得附加的信息. 更多关于这些标签和压力测试详细报告的详情, 单见“Reviewing and Customizing Load Test Results”, 第 82 页.

第 7 课: 保存项目为 .lt 文件

- 1 保存该压力测试项目: 单击SAVE PROJECT 工具栏按钮.



- 2 在打开的对话框中的FILE NAME 字段为项目输入一个名称, 并单击SAVE 按钮. 压力测试保存项目并添加.LT 扩展名到文件名称.