

Web性能测试的专业工具书，
软件测试工程师的良师益友。

零成本实现 Web性能测试

——基于Apache JMeter

温素剑 编著



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

内 容 简 介

本书是一本关于 Web 性能测试的实战书籍，读者朋友们在认真阅读完本书后，相信能够将所学知识应用到生产实践中。本书首先介绍基础的性能测试理论，接着详细介绍如何使用 JMeter 完成各种类型的性能测试。实战章节中作者以测试某大型保险公司电话销售系统为例，手把手教会读者如何用 JMeter 来完成一个实际的性能测试任务。

本书内容丰富、知识点讲解透彻，适合软件测试工程师、测试经理、高等院校相关专业的学生参考学习，同时也可作为相关培训班的教材。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有，侵权必究。

图书在版编目（CIP）数据

零成本实现 Web 性能测试：基于 Apache JMeter / 温素剑编著. —北京：电子工业出版社，2012.2
ISBN 978-7-121-15526-0

I. ①零… II. ①温… III. ①计算机网络—程序设计 IV. ①TP393

中国版本图书馆 CIP 数据核字（2011）第 265546 号

策划编辑：张月萍

责任编辑：贾 莉

特约编辑：赵树刚

印 刷：北京东光印刷厂

装 订：三河市皇庄路通装订厂

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱 邮编：100036

开 本：787×980 1/16 印张：22.25 字数：479 千字

印 次：2012 年 2 月第 1 次印刷

印 数：3500 册 定价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

印 次：2012 年 1 月第 2 次印刷

印 数：5001~6000 册 定价：65.00 元

前言

作者曾经有幸在国内最大的电信设备供应商工作过一段时间，其间听一些资深老员工讲过一个故事。这个故事可以被当做笑话来听，不过笑笑之后却总也忘不掉。话说 20 世纪 90 年代初的某一天，国内第一台自主研发的大型固话交换机，终于千呼万唤地“闪亮”登场了。于是乎，这家公司马上向用户大力推销这款设备，但是用户提出了一个很实际的问题，彻底难住了这家公司。问题很简单，就是需要一份性能测试报告来证明这台设备真能支持宣称的话务容量。那时候还没有成熟的电信领域性能测试工具，该怎么办呢？幸好有聪明的领导想出了一个中国式的解决办法。

某天下午，全公司的员工都放下了手头的工作，每人怀抱一部老式电话机（还要靠转盘来拨号），等领导倒数“三、二、一”后集体打电话。据说当时人数不够，达不到用户要求的通话量，甚至出现了一个人操作两部电话机的情况。作者没能一睹当时的盛况，一直深感遗憾。

幸好科学技术发展到今天，已经有了多款成熟的性能测试工具，否则测试人员一定会发疯。试想当前的电信交换机话务容量早已翻了不知多少倍，如果还要靠人海战术去测试，即使全公司的员工双手双脚去操作电话机，也肯定是忙不过来的。测试人员应该为测试技术的飞速发展而感到欢心鼓舞。那么现在是否就可以高枕无忧了？答案是否定的。当前测试人员面临的问题不再是有没有性能测试工具，而是有没有**合适**的性能测试工具。

怎么界定“合适”一词？我想至少可以包含如下几个方面：

- 技术先进，功能强大。
- 支持多种测试类型（协议）。
- 易学易用。

- 拥有良好的可扩展性。
- 拥有良好的可移植性（跨平台）。
- 合理的价格。

当前性能测试工具有很多，但同时满足了以上数个条件的却很少。在 Web 性能测试领域，目前有两种工具被广泛使用，其一是 LoadRunner，另一个就是 JMeter。不过 LoadRunner 并不是一款“合适”的工具，在此作者并不否认 LoadRunner 是一款优秀的性能测试工具，但它唯一的缺点就是过于昂贵（关于 LoadRunner Liscence 及其支持服务的具体价格，感兴趣的朋友可以向 HP 公司了解，作者相信其价格会将中国 90% 以上的软件公司挡在门外）。JMeter 具备了 LoadRunner 95% 以上的功能，但其价格却无限接近于零，可谓性价比极高。当然相对于商业工具 LoadRunner，JMeter 也有其自身的缺点。它最大的缺点就是没有专业的售后支持队伍，不过想想商业工具贵得令人咂舌的维护支持费用，也就能够释怀了。

写作背景

作者目前在一家大型保险公司 IT 测试部门工作，带领一个测试团队负责测试公司的电话销售系统。这个系统非常庞大，由多个子系统构成，同时它又与很多公司内部/外部系统（如银联、银行的交易系统）发生交互，目前公司有数万员工依赖它来完成每日的销售任务。如此复杂的一个系统，偏偏又拥有数量众多的用户，读者朋友可以试想一下，只要此系统稍有异常，业务部门的投诉绝对会让 IT 部门“吃不了兜着走”。

面对频繁的版本发布，严格的系统性能测试是不可或缺的。测试部门也花大价钱购买了商业工具 LoadRunner，但是在实际工作中，作者发现测试人员还是受到颇多限制的。其一，测试资源存在瓶颈，公司购买的 Liscence 是有限的，无法完全满足测试人员的需求，经常出现人等机器的情况，甚至影响到了软件版本的及时发布；其二，公司购买的并发数许可也是有限的，如果需要模拟更大的系统压力，公司还得再掏钱；其三，公司购买的协议类型是有限的，如果需要使用其他未购置的协议类型，公司依然需要再掏钱（测试人员很难用临时的测试需求去说服领导花上一大笔经费）。

在尝试说服领导增加预算失败后，只能转而寻求其他解决办法，那就是开源性能测试工具 **JMeter**。经过大规模的试用后，发现 **JMeter** 完全能够满足测试人员的需求。

“云计算”绝对是当前最热的 IT 词汇，甚至沾上一点“云”概念的股票都会一飞冲天。“云”听起来很虚幻，其实就是瘦客户端加网格计算。今后客户端不再会有大量的计算任务，计算和存储都被放在云上。在作者看来，今后的客户端应该就是一个浏览器，用户的所有操作都是通过浏览器来实现的。Google 刚发布的操作系统 **Chrome OS**，就是基于这一理念设计的。**B/S** 和 **C/S** 架构的软件系统，应该会慢慢演进到 **Browser/Cloud**（浏览器/云）模式。如此看来，在“云计算”时代，**Web** 性能测试依然很重要，而且会越来越重要。因此，作者萌生了写作一本关于 **Web** 性能测试的书籍。

本书内容

本书不是一本讲述深奥测试理论的教科书，而是一本实战类的书籍。作者想要达到的目标就是——读者朋友们在认真读完本书后，马上就能在生产实践中用上所学内容。本书首先介绍基础的性能测试理论，接着详细介绍如何使用 **JMeter** 来完成各种类型的性能测试，而最重要的是性能测试实战章节。实战章节中作者以测试某大型保险公司电话销售系统为例，手把手教会读者如何用 **JMeter** 来完成一个实际的性能测试任务。第 1 章介绍性能测试理论；第 2~12 章详细介绍 **JMeter** 工具在各种场景下的使用方法，以及如何分析性能测试结果；第 13 章是性能测试实战。

目标读者

本书的目标读者是初级或者资深软件测试工程师，以及有意降低性能测试成本的测试经理。本书也适合应届本科毕业生，帮助他们熟练掌握性能测试的方法和技巧，是求职就业一块不错的敲门砖。本书着重介绍如何使用 **JMeter** 开源性能测试工具来构建 **Web** 性能测试体系。

感谢

首先要感谢我的家人，正是有他们默默的支持，我才能静下心来写作；其次还要感谢参与本书编写的部门同事刘兴翬、何邱、邓智、谷明、李喆、李坤、袁春梅、唐明娟、李颖、岑海菊、陈建红、路菁、李超、曾泗维；最后还要感谢电子工业出版社张月萍编辑的热情帮助。

温素剑

2011 年 12 月 16 日

目录

第 1 章 性能测试基础	1
1.1 初识性能测试	1
1.1.1 性能测试的概念	1
1.1.2 性能测试的目的	2
1.1.3 性能测试的常见分类	2
1.1.4 性能测试的常见指标	3
1.1.5 性能测试的基本流程	4
1.2 开源 Web 性能测试	8
1.2.1 Web 性能测试的重要性	8
1.2.2 开源 Web 性能测试介绍	8
1.2.3 开源性能测试的优势	9
1.3 本章小结	10
第 2 章 JMeter 基础知识	11
2.1 JMeter 简介	11
2.1.1 JMeter 主要特点	12
2.1.2 JMeter 常用术语	13
2.1.3 JMeter 测试结果字段的意义	13

2.2	JMeter 工作原理	14
2.3	JMeter 的安装与目录结构	15
2.3.1	JMeter 安装配置要求	15
2.3.2	JMeter 目录结构	15
2.4	如何运行 JMeter	18
2.5	配置 JMeter	25
2.6	JMeter 与 LoadRunner 优缺点对比	25
2.7	本章小结	26
第 3 章	Web 性能测试脚本录制与开发	27
3.1	JMeter GUI 基本操作	27
3.2	JMeter 常用测试元件	30
3.3	JMeter 脚本开发基础	37
3.3.1	JMeter 执行顺序规则	37
3.3.2	作用域规则	38
3.3.3	JMeter 属性和变量	40
3.3.4	使用变量参数化测试	40
3.4	创建 Web 测试计划	41
3.5	录制 Web 测试脚本	47
3.5.1	使用代理录制 Web 性能测试脚本	47
3.5.2	使用 Badboy 录制 Web 性能测试脚本	52
3.6	创建高级 Web 测试计划	57
3.7	本章小结	58
第 4 章	数据库性能测试脚本开发	59
4.1	创建数据库测试计划	59
4.2	九步轻松搞定 Oracle 数据库性能测试	62
4.3	本章小结	68

第 5 章	FTP 性能测试脚本开发	69
5.1	FTP 是什么	69
5.2	创建 FTP 测试计划	74
5.3	本章小结	78
第 6 章	LDAP 性能测试脚本开发	79
6.1	LDAP 是什么	79
6.2	创建 LDAP 测试计划	90
6.3	LDAP 常见操作指南	95
6.4	创建扩展 LDAP 测试计划	97
6.5	本章小结	107
第 7 章	Web Service 性能测试脚本开发	108
7.1	Web Service 是什么	108
7.2	创建 Web Service 测试计划	112
7.3	本章小结	115
第 8 章	JMS 性能测试脚本开发	116
8.1	JMS 是什么	116
8.2	创建 JMS 点对点测试计划	121
8.3	创建 JMS Topic 测试计划	124
8.4	本章小结	128
第 9 章	服务器监控测试脚本开发	129
9.1	创建监控测试计划	129
9.2	本章小结	133
第 10 章	详解 JMeter 测试原件	134
10.1	详解 JMeter 监听器	134

10.2	详解 JMeter 逻辑控制器	144
10.3	详解 JMeter 配置元件	161
10.4	详解 JMeter 定时器	181
10.5	详解 JMeter 前置处理器	187
10.6	详解 JMeter 后置处理器	196
10.7	详解 JMeter 采样器	205
10.8	详解 JMeter 其他测试元件	248
10.9	本章小结	261
第 11 章 JMeter 进阶知识		262
11.1	详解 JMeter 函数和变量	262
11.2	详解 JMeter 正则表达式	282
11.3	详解 JMeter 远程测试	285
11.4	详解 JMeter 最佳实践经验	291
11.5	一些小技巧	297
11.6	本章小结	299
第 12 章 性能测试结果分析		300
12.1	如何分析性能测试结果	300
12.2	如何借助监听器发现性能缺陷	303
12.2.1	监听器——性能测试分析的基石	303
12.2.2	巧用监听器——识别性能缺陷	322
12.3	借助 Ant 实现批量测试和报表生成	330
12.4	本章小结	331
第 13 章 JMeter 性能测试实战——电话销售系统		332
13.1	测试背景和测试目标	332
13.2	分析确定性能测试指标	332
13.3	录制创建性能测试脚本	334

13.4	运行性能测试脚本.....	337
13.5	分析性能测试结果.....	340
13.6	上报性能测试缺陷.....	342
13.7	本章小结.....	343

Web性能测试脚本录制与开发

3.1 JMeter GUI 基本操作

测试计划描述了 JMeter 运行时将会执行的一系列步骤。一个完整的测试计划会包含一个或多个线程组、逻辑控制器、采样器、监听器、定时器、断言和配置元件。

1. 添加/移除测试元件

如果想要为测试计划添加测试元件，先选中测试树上的某个元件，接着用鼠标右键单击，在弹出的快捷菜单中选择“添加”命令，然后在其级联菜单中选择一个新元件，如图 3-1 所示。另外也可以通过选择“打开”、“合并”命令，从外部文件中加载和添加测试元件。

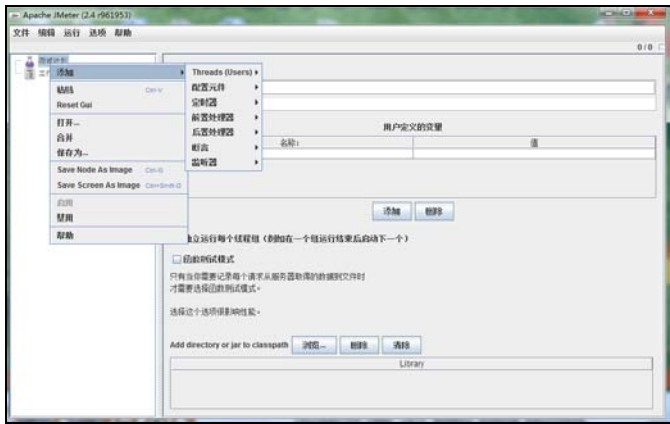


图 3-1 JMeter 添加测试元件

如果要移除某个测试元件，先选中该测试元件，接着用鼠标右键单击，并在弹出的快捷菜单中选择“删除”命令。

2. 加载和保存测试元件

如果想要从文件中加载测试元件，首先选中想要添加测试元件的地方，并用鼠标右键单击，在弹出的快捷菜单中选择“合并”命令，接着在弹出的对话框中选择外部文件（保存有待添加的测试元件），如此 JMeter 就会将测试元件添加到测试树中。

如果要保存测试树中的某个测试元件，可以选中该测试元件后用鼠标右键单击，在弹出的快捷菜单中选择“保存为”命令。JMeter 会保存选中的测试元件，以及其下的子测试元件。通过这种方法，可以保存测试树的某个片段或者独立的测试元件，以供后续使用。



小贴士

挂在工作台下的测试元件不会与测试计划一起保存，但可以通过上述方式单独保存。

3. 配置测试树中的测试元件

可以在 JMeter 图形用户界面的右侧，找到测试树中任何一个测试元件所对应的控制面板。这些控制面板可以帮助用户设定某个测试元件的行为。用户可以设定的内容，由测试元件所属类型所决定。



小贴士

用户可以在测试树中拖动和释放测试元件，以便调整测试元件的先后顺序。

4. 保存测试计划

尽管这不是必须的，但是笔者强烈建议在运行测试计划前，先将它保存到某个文件中。要保存测试计划，选择“文件”菜单下的“保存测试计划”或者“保存测试计划为”命令。



小贴士

JMeter 允许用户全部或者部分保存测试计划，方法可参见保存测试元件。

5. 运行测试计划

要运行测试计划，选择“运行”→“启动”命令，或按“Ctrl+R”组合键。当 JMeter 处于运行状态时，在图形用户界面的右上角有一个绿色小盒子，就在菜单栏的下方。检查 JMeter 是否处于运行状态的另一个办法是，如果“运行”菜单下的“启动”选项被置灰，而“停止”选项可用，那么 JMeter 就处于运行状态。

绿色小盒子右边的数字，代表激活状态线程数/总线程数，如图 3-2 所示。需要注意的是，这里的数字只包含了本地的 JMeter 线程，并不包括任何运行在远端的 JMeter 线程（JMeter 处于客户端/服务器模式下）。

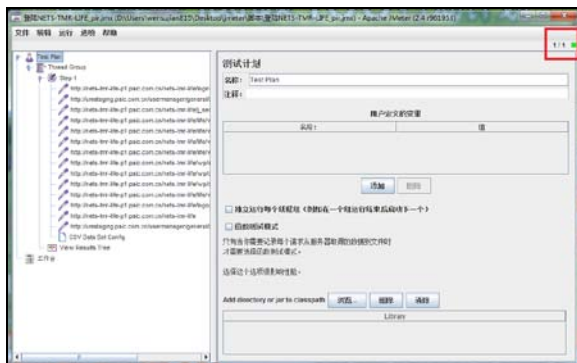


图 3-2 JMeter 处于运行状态

6. 终止测试

在菜单中有两种命令可以用于终止测试。

- **停止 (Ctrl+.)**: 如果可能的话，立刻停止所有线程。JMeter 2.3.2 及其以后版本的很多采样器都支持中断，这就意味着处于激活状态的线程可以被更快地终止。停止命令会在默认超时时间内，检查所有线程是否都已正确终止。如果有线程在超时时间内没有终止，那么就会弹出一条提示信息。停止命令可以被多次执行，不过一旦它失败后，就必须退出 JMeter，以便恢复干净的执行环境。
- **关闭 (Ctrl+,)**: 要求线程在当前工作完成后停止，这项操作不会中断任何采样器的工作。关闭对话框会一直处于激活状态，直到所有线程都已经停止。

JMeter 2.3.2 及其以后版本，允许用户在关闭持续太长时间后，发起停止操作。首先关掉“关闭对话框”，接着选择“运行”→“停止”命令，或者直接使用组合键 Ctrl +.。当 JMeter 运行在非 GUI 模式下时，没有菜单栏，JMeter 也不会响应组合键，例如，Ctrl +.。因此在 2.3.2 版本以后，JMeter 会监听特殊端口（默认为 4445，参考 JMeter 属性 jmeterengine.nongui.port）上的命令。目前支持的命令包括：

- **Shutdown**: 关闭。
- **StopTestNow**: 停止。

这两条命令可以分别使用 bin 目录下的 shutdown[.cmd|.sh]或者 stoptest[.cmd|.sh]脚本来完成。

7. 错误报告

JMeter 会将告警或者错误记录到 jmeter.log 文件中，甚至还包括一些测试自己的信息。偶尔会有一些 JMeter 无法跟踪/记录的错误信息，这些信息会出现在命令控制台中。如果测试运行起来后，不像测试人员期望的那样，那么应该首先检查日志文件中是否有错误信息（如调用函数时出现语法错误）。

采样器错误（如 HTTP 404：文件未找到）通常并不记录到日志文件中，而是作为采样结果的属性值加以保存。采样结果的状态可以在各种监听器中查看。

3.2 JMeter 常用测试元件

JMeter 测试计划有一个被称为“函数测试模式”的选项，当这一选项被选中后，就会促使 JMeter 记录下每一次采样从服务器获取的数据。如果测试人员在测试监听器中配置了保存测试数据的文件，那么这些数据就会被记录到该文件中。这项功能很有用，特别是测试人员可能需要简单运行一下测试脚本，以便验证 JMeter 的配置是否正确，以及服务器返回的结果是否符合预期。不过如此一来，保存测试数据的文件会迅速变得庞大起来，JMeter 的性能也会受到影响。因此当测试人员使用 JMeter 进行压力测试时，应该关闭这一选项（默认情况下它是关闭的）。如果不记录测试数据到文件中，那么这一选项选中与不选中就没有区别。另外，测试人员可以使用监听器上的“Configure”按钮，来配置哪些测试数据应该被保存，如图 3-3 所示。

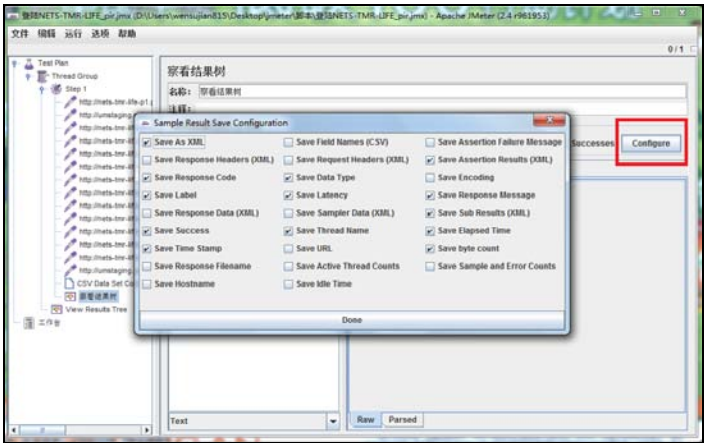


图 3-3 设定 JMeter 如何记录测试数据

1. 线程组

线程组是任何测试计划的起点，所有的逻辑控制器和采样器都必须放在线程组之下。其他的测试元件（如监听器）可以被直接放在测试计划之下，这些测试元件对所有线程组都生效。线程组就像它的名字所描述的那样，被用来管理执行性能测试所需的 JMeter 线程。用户通过线程组的控制面板可以：

- 设置线程数量。
- 设置线程启动周期。
- 设置执行测试脚本的循环次数。

每一个 JMeter 线程都会完整地执行测试计划，而且它们之间是完全独立运行的。这种多线程机制被用来模拟服务器应用的并发连接。参数 Ramp-Up Period 告诉 JMeter 达到最大线程数需要多长时间。假定共有 10 个线程，Ramp-Up Period 为 100 秒，那么 JMeter 就会在 100 秒内启动所有 10 个线程，并让它们运转起来。每一个测试线程都会在上一个线程启动 10 秒之后才开始运行。假定共有 30 个线程，Ramp-Up Period 为 120 秒，那么线程启动的间隔就为 4 秒。

Ramp-Up 参数不能设定得太短，否则在测试的初始阶段会给予服务器过大的压力。Ramp-Up 参数也不能设定得太长，否则就会发生第一个线程已经执行完毕，而最后一个线程还没有启动的情况（除非测试人员期望这种特殊情况发生）。

如何找到一个合适的 Ramp-Up 参数值？作者建议初始值可以设定为 Ramp-Up=总线程数，后续再根据实际情况适当增减。

默认情况下，JMeter 线程组被设定成只执行一遍，用户可以根据实际需要设定参数“循环次数”，如图 3-4 所示。

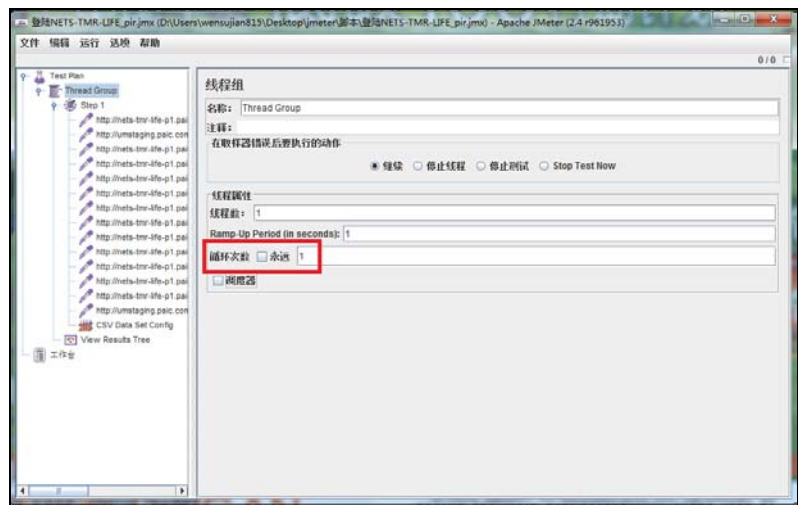


图 3-4 设定参数“循环次数”

JMeter 在 1.9 版本引入了调度器，用户可以选中“调度器”选项，以便展开额外的调度器控制面板，如图 3-5 所示。在调度器控制面板中，可以设定测试运行的“启动时间”和“结束时间”。测试启动后会一直等待，直到用户设定的启动时间。测试运行期间，JMeter 会在每一次循环结束后，检查是否已经达到结束时间。如果已经达到了结束时间，JMeter 就会终止测试运行，否则 JMeter 会继续下一个测试循环。

另外，用户还可以设定“持续时间”和“启动延迟”两项参数。需要注意的是，“启动延迟”会使“启动时间”无效，而“持续时间”会使“结束时间”无效。

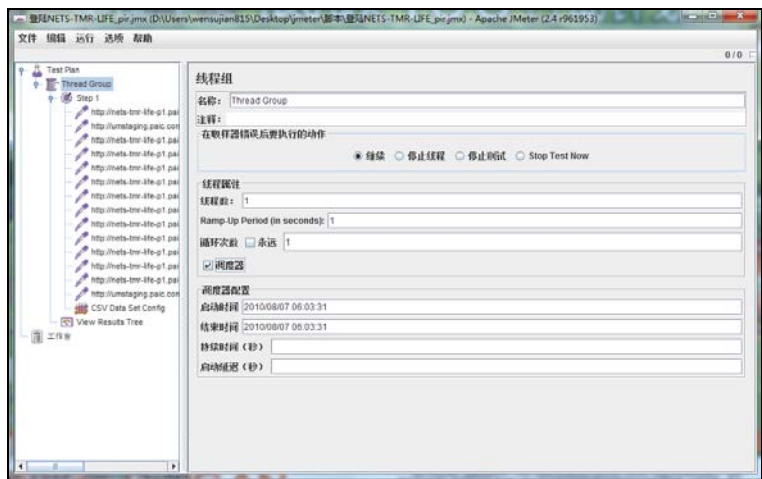


图 3-5 展开额外的调度器控制面板

2. 控制器

JMeter 有两种类型的控制器：采样器和逻辑控制器，二者结合起来驱动了测试进程。采样器被 JMeter 用来向服务器发送请求。例如，当测试人员想往服务器发送一个 HTTP 请求时，就加入一个 HTTP 请求采样器。测试人员还可以通过为采样器添加配置元件来定制化请求。

用户可以使用逻辑控制器来控制 JMeter 的测试逻辑，比如何时发送请求。举一个例子：测试人员可以插入交替控制器来轮流发送多个请求。

1) 采样器

采样器告诉 JMeter 发送一个请求到指定服务器，并等待服务器的请求。采样器会按照其在测试树中的顺序去执行，还可以用逻辑控制器来改变采样器运行的重复次数。

JMeter 采样器包含：

- FTP Request
- HTTP Request
- JDBC Request
- Java object request
- LDAP Request
- SOAP/XML-RPC Request
- WebService (SOAP) Request

每一种采样器都有多种参数可供设置。测试人员还可以通过在测试计划中加入一个或者多个配置元件，来进一步定制化采样器。

如果测试人员打算向同一个服务器发送同一类请求，可以考虑使用默认配置元件。每一类采样器都有一个或多个对应的默认配置元件。一定记住应为测试计划添加一个监听器，以便查看和存储（存储到磁盘）请求的结果。

如果测试人员想检查服务器响应的内容，可以为对应采样器添加断言。例如，当对 Web 应用做压力测试时，服务器虽然成功返回了"HTTP Response"代码，但是页面上可能会有错误，或者丢失了部分页面片段。针对这种情况，测试人员可以添加断言来检查特定的 HTML 标签，或者常见的错误信息等。JMeter 允许在断言中使用正则表达式。

2) 逻辑控制器

逻辑控制器可以帮助用户控制 JMeter 的测试逻辑，特别是何时发送请求。逻辑控制器可以

改变其子测试元素的请求执行顺序。

为了进一步弄明白逻辑控制器的作用，考虑如下测试树：

- Test Plan
- Thread Group
- Once Only Controller
- Login Request (an HTTP Request)
- Load Search Page (HTTP Sampler)
- Interleave Controller
- Search "A" (HTTP Sampler)
- Search "B" (HTTP Sampler)
- HTTP default request (Configuration Element)
- HTTP default request (Configuration Element)
- Cookie Manager (Configuration Element)

首先让我们看登录请求（Login Request），在整个测试中它只会运行一次，在剩下的测试循环中它会被忽略，原因就在于仅一次控制器（Once Only Controller）。

在登录之后，接着是载入搜索页面的采样器（想象有这样一个 Web 应用系统，在用户登录后进入搜索页面）。它仅仅是一个普通采样器，没有父逻辑控制器。在加载搜索页面后，我们想做一个搜索。事实上我们想做两种完全不同的搜索，而且在不同搜索之间还要重新加载搜索页面。如此一来就需要 4 个 HTTP 请求测试元件（加载搜索页面，搜索“A”；加载搜索页面，搜索“B”）。这里有一个更简单的办法，那就是使用交替逻辑控制器，它会在每一次测试循环中仅执行一个子请求。它还会记住子请求的顺序，而不是随机执行。交替执行两个子请求没有太大意义，但是它可以轻松扩展到 8 个或者 20 个，甚至更多子请求。

注意交替逻辑控制器下的 HTTP 请求默认值（HTTP Default Request）。考虑这样一种情况，搜索“A”和搜索“B”有同样的路径信息（HTTP 请求明细包含域、端口、方法、协议、路径、参数以及其他可选的信息）。这就意味着它们都是搜索请求，而且背后有同样的搜索引擎（servlet 或者 cgi-script）。与其在每一个 HTTP 采样器路径域中设置同样的信息，不如将它们抽取出来放到一个配置元件中。当交替控制器处理请求搜索“A”和搜索“B”时，它会为请求的空白信息栏填充 HTTP 请求默认值中保存的值。因此将请求的路径域留为空白，并将该信息放到配置元件中。在这个例子中对配置元件的好处没有得到充分展现，但是它演示了配置元件的特性。

测试树中的下一个测试元件是另一个 HTTP 请求默认值 (HTTP Default Request)，这一次它对整个线程组都生效。线程组有一个内在的逻辑控制器，它像上面描述的那样使用配置元件，它会为每一个请求填充默认值。在 Web 测试中，它可以用来存储域字段，如此一来所有 HTTP 采样器都可以不填域字段。测试人员需要做的就是将信息放入 HTTP 请求默认值中，再把它添加到线程组之下。这么做的好处在于测试人员可以测试不同站点的应用，而仅仅需要改变测试计划中的一个地方。否则，测试人员就需要手动修改每一个采样器。

最后一个测试元件就是 HTTP Cookie 管理器。所有的 Web 测试都应该添加 Cookie 管理器，否则 JMeter 就会忽略 Cookie。通过把 HTTP Cookie 管理器添加到线程组层级，就能确保所有 HTTP 请求使用相同的 Cookie。

需要注意的是，逻辑控制器可以被组合起来使用，以便达到各种测试目的。

3. 监听器

监听器提供了对 JMeter 在测试期间收集到的信息的访问方法。“图形结果”监听器会将系统响应时长绘制在一张图片之中。“查看结果树”监听器会展示采样器请求和响应的细节，还能以 HTML 和 XML 格式展示系统响应的基础部分。其他监听器通过总结或者聚合方式展示信息。

另外，监听器可以将测试数据导入到文件之中，以供后续分析。所有监听器都会提供一个输入域，以便于用户指定存储测试数据的文件。监听器还会提供一个配置按钮，用来配置存储测试数据的哪些字段，以及选用的存储格式 (CSV 或者 XML)。读者朋友需要注意的是，所有监听器都保存同样的数据，唯一的区别是它们如何展示数据。

监听器可以在测试的任何地方添加，包括直接放在测试计划之下。它们仅收集测试树中相同或者更低级别测试元件的数据。

4. 定时器

默认情况下，JMeter 线程在发送请求之间没有间歇。建议为线程组添加某种定时器，以便设定请求之间应该间隔多长时间。如果测试人员不设定这种延迟，JMeter 可能会在短时间内产生大量访问请求，导致服务器被大量请求所淹没。

定时器会让作用域内的每一个采样器都在执行前等待一个固定时长。如果测试人员为线程组添加了多个定时器，那么 JMeter 会将这些定时器的时长叠加起来，共同影响作用域范围内的采样器。定时器可以作为采样器或者逻辑控制器的子项，目的是只影响作用域内的采样器。

要在测试计划中的某个位置添加暂停，测试人员可以使用“Test Action”采样器。

5. 断言

用户可以使用断言来检查从服务器获得的响应内容。通过断言可以测试服务器返回的响应与测试人员的期望是否相符。

例如，测试人员可以断言某个查询的响应中包含特定的文字信息。测试人员可以使用 Perl 格式的正则表达式来描述响应中应该包含的文字，或者它应该与整个响应相符。

测试人员可以为任何采样器添加断言。例如，测试人员可以为 HTTP 请求添加断言，用于检查文本 “</HTML>”。接下来 JMeter 就会检查该文本是否出现在 HTTP 响应中。如果 JMeter 不能找到该文本，那么它就会将请求标记为失败。

需要注意的是，断言会影响作用域内的所有采样器。如果要让断言只影响某个采样器，需要将断言作为该采样器的子项。

如果要查看断言结果，可以为线程组添加“断言结果”监听器。失败的断言，也会在“查看结果树”和“用表格查看结果”两种监听器中显示。另外，在“Summary Report”和“聚合报告”中还会以错误百分率的形式统计。

6. 配置元件

配置元件与采样器紧密关联。虽然配置元件并不发送请求（除了 HTTP 代理服务器例外），但它可以添加或者修改请求。

配置元件仅对其所在的测试树分支有效。例如，假设测试人员在一个简单逻辑控制器中放置了一个 HTTP Cookie 管理器，那么该 HTTP Cookie 管理器只对放置在简单逻辑控制器内的其他逻辑控制器生效。如图 3-6 所示，该 Cookie 管理器对“Web Page 1”和“Web Page 2”有效，而对“Web Page 3”无效。

另外，相比父分支的配置元件，子分支内部的配置元件优先级更高。例如，我们定义了两个 HTTP 请求默认值：“Web Defaults 1”和“Web Defaults 2”。因为将“Web Defaults 1”放在循环控制器的内部，所以只对“Web Page 2”生效。而其他的 HTTP 请求使用“Web Defaults 2”，原因在于“Web Defaults 2”被置于线程组之下（所有分支的父分支）。

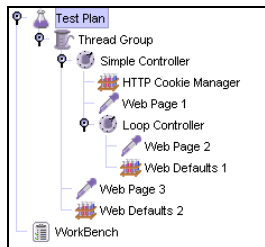


图 3-6 配置元件的作用域



小贴士

这里有个例外，配置元件“用户定义的参数”会在测试的初始阶段执行（无论它处于测试树的哪个位置）。为了便于理解，建议将它放在线程组的开始部分。

7. 前置处理器

前置处理器会在采样器发出请求之前做一些特殊操作。如果前置处理器附着在某个采样器之下，那么它只会在该采样器运行之前执行。前置处理器通常用于在采样器发出请求前修改采样器的某些设置，或者更新某些变量的值（这些变量不在服务器响应中获取值）。请参考本书关于作用域的介绍，以便了解更多关于前置处理器使用的细节。

8. 后置处理器

后置处理器会在采样器发出请求之后做一些特殊操作。如果后置处理器附着在某个采样器之下，那么它只会在该采样器运行之后执行。后置处理器通常被用来处理服务器的响应数据，特别是服务器响应中提取数据。请参考本书关于作用域的介绍，以便了解更多关于后置处理器使用的细节。

3.3 JMeter 脚本开发基础

3.3.1 JMeter 执行顺序规则

JMeter 执行顺序规则如下：

- 配置元件
- 前置处理器

- 定时器
- 采样器
- 后置处理器（除非服务器响应为空）
- 断言（除非服务器响应为空）
- 监听器（除非服务器响应为空）



小贴士

只有当作用域内存在采样器时，定时器、断言、前置/后置处理器才会被执行。逻辑控制器和采样器按照在测试树中出现的顺序执行。其他测试元件会依据自身的作用域范围来执行，另外还与测试元件所属的类型有关（归属于同一类型的测试元件，会按照它们在测试树中出现的顺序来执行）。

例如，在如下测试计划中：

- Controller
 - Post-Processor 1
 - Sampler 1
 - Sampler 2
 - Timer 1
 - Assertion 1
 - Pre-Processor 1
 - Timer 2
 - Post-Processor 2

执行顺序为：

```
Pre-Processor 1
Timer 1
Timer 2
Sampler 1
Post-Processor 1
Post-Processor 2
Assertion 1
```

```
Pre-Processor 1
Timer 1
Timer 2
Sampler 2
Post-Processor 1
Post-Processor 2
Assertion 1
```

3.3.2 作用域规则

JMeter 测试树中既包含遵循分层规则的测试元件，又包含遵循顺序规则的测试元件。有些测试元件在测试树中严格遵循分层规则（监听器、配置元件、后置处理器、前置处理器、断言、定时器），而另外一些测试元件遵循原始的顺序规则（逻辑控制器、采样器）。在测试人员创建测试计划的同时，实际上就创建了一个采样器请求的顺序列表（描述了测试步骤的执行顺序）。用户经常使用逻辑控制器来管理这些采样器请求，不过即使如此，JMeter 执行顺序依然是唯一确定的。考虑如下测试树，如图 3-7 所示。

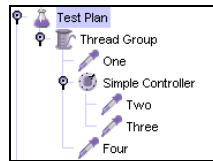


图 3-7 顺序规则举例

采样器的执行顺序应该是：One、Two、Three、Four。

有一些逻辑控制器会影响其子测试元件的执行顺序，例如循环控制器。关于这些逻辑控制器的详细使用方法，请参考 JMeter 工具的帮助文档。

其他测试元件遵循分层规则。例如，断言在测试树中就遵循分层规则。如果断言的父测试元件是一个采样器，那么它就仅对该采样器生效。如果断言的父测试元件是一个逻辑控制器，那么它对该逻辑控制器下的所有子采样器都生效。考虑如下测试树，如图 3-8 所示。

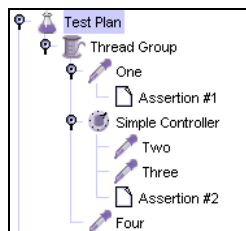


图 3-8 分层规则举例 1

Assertion #1 只对请求 One 生效，而 Assertion #2 对请求 Two 和 Three 生效。下面来看另外一个例子，如图 3-9 所示，这次还会用到定时器。

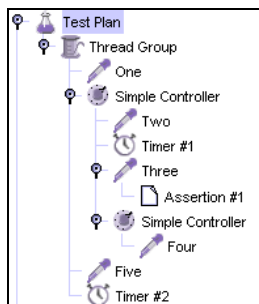


图 3-9 分层规则举例 2

在该例中，采样器的命名表明了它们的执行顺序。Timer #1 对请求 Two、Three、Four 生效（注意，遵循分层规则的测试元件不会受顺序规则约束），Assertion #1 仅仅对请求 Three 生效，Timer #2 会对所有请求生效。

希望通过上面两个例子能让读者明白配置元件（遵循分层规则）的作用域。

配置元件（HTTP 信息头管理器、Cookie 管理器和 HTTP 授权管理器）与默认配置元件（Configuration Default Element）的处理方式不同。默认配置元件包含的设置会被合并成一系列变量值（采样器可以访问），而配置元件的设置不会被合并。对一个采样器而言，如果在相同的作用域范围内有多个配置元件，那么只有一个配置元件会被应用，而且目前没有办法指定哪个配置元件会被应用。

3.3.3 JMeter 属性和变量

JMeter 属性统一定义在 `jmeter.properties` 文件中。JMeter 属性在测试脚本的任何地方都是可见的（全局），通常被用来定义一些 JMeter 使用的默认值。例如，属性 `remote_hosts` 定义了 JMeter 在远程模式下运行的服务器地址。属性可以在测试计划中引用（参见本书介绍 JMeter 函数的章节，读取属性函数 `_P`），但是不能作为特定线程的变量值。

JMeter 变量对于测试线程而言是局部变量。这就意味着 JMeter 变量在不同测试线程中，既可以是完全相同的，也可以是不同的。

如果有某个线程更新了变量，那么仅仅是更新了变量在该线程中复制的值。例如，“正则表

达式提取器”（后置处理器）会依据它所在线程的采样结果来更新变量值，该变量值可以供相同的线程后续使用。关于如何引用变量和函数，请参见本书介绍 JMeter 函数与变量的章节。

注意，通过测试计划和“用户定义的变量”（配置元件）两种方式定义的变量，在 JMeter 启动时对这个测试计划都是可见的。如果同一个变量在多个“用户定义的变量”（配置元件）中被定义，那么只有最后一个定义会生效。一旦某个线程启动后，那么整个变量集合的初始值就会被复制到该线程中。其他测试元件，例如“用户变量”（前置处理器）或者“正则表达式提取器”（后置处理器）可以被用来重新定义变量（或者创建新变量），这些重定义仅仅影响当前线程。

可以通过 `_setProperty` 函数来定义 JMeter 属性。JMeter 属性对于整个测试计划都是可见的（全局），因此可以用于在线程间传递信息（这种情况并不多见）。



小
贴
士

属性和变量都是大小写敏感的。

3.3.4 使用变量参数化测试

变量并不一定要一直发生变化——如果变量定义之后一直不用，那么它的值就会保持不变。因此测试人员可以用变量来代替某些在测试计划中经常出现的表达式，或者某些在单次测试运行过程中不发生变化，但在多次测试运行之间会发生变化的事物，例如，主机名或者线程数量。

在考虑如何构建测试计划时，需要注意哪些在测试运行期间是恒定不变的（常量），而哪些在不同线程之间可能会发生变化（变量）。对于常量应该有单独的命名规则，例如加前缀 `C_` 或者 `K_`，或者使用大写，以便区别于变量。另外，还需要考虑哪些对于测试线程而言是独享的，例如计数器或者通过“正则表达式提取器”（后置处理器）提取的变量。测试人员可以对这些变量也采用不同的命名策略。

例如，测试人员可以在测试计划中如此命名常量：

```
HOST          www.example.com
THREADS       10
LOOPS         20
```

可以在测试计划中使用 `${HOST}`、`${THREADS}` 来引用测试变量。如果测试人员接下来想改变主机名，只需修改对应变量的值即可。这种方法适用于并发量较小的情况，对于大并发的压力测试最好使用 JMeter 属性。例如：

```
HOST          ${__P(host,www.example.com)}
THREADS       ${__P(threads,10)}
LOOPS         ${__P(loops,20)}
```

可以通过命令行来改变 JMeter 属性的值，例如：

```
jmeter ... -Jhost=www3.example.org -Jloops=13
```

3.4 创建 Web 测试计划

在这一节中，将会介绍如何创建一个简单的测试计划，用于测试 Web 站点。我们会模拟 5 个并发用户，对 Jakarta Web 站点的两个页面进行访问。另外，每个并发用户都会运行测试两次。因此测试计划产生的总请求数目为 $(5 \text{ 并发用户}) \times (2 \text{ 请求}) \times (\text{重复 } 2 \text{ 次}) = 20 \text{ HTTP 请求}$ 。要构建该测试计划，测试人员需要用到如下测试元件：线程组 (Thread Group)、HTTP 请求 (HTTP Request)、HTTP 请求默认值 (HTTP Request Defaults) 和图形结果 (Graph Results)。

1. 添加并发用户

创建 JMeter 测试计划的第一步就是添加线程组测试元件。线程组会告诉 JMeter 需要模拟的并发用户数，以及并发用户发送请求的频率和数目。

要添加线程组，首先选中测试计划，接着单击鼠标右键，在弹出的快捷菜单中选择“Add”→“Threads(Users)”→“ThreadGroup”命令。测试人员现在就应该能够在测试计划下看到线程组。如果没有看到，单击测试计划以便展开测试计划树。

接下来，测试人员需要修改线程组的默认设置。在测试树中选中线程组后，测试人员应该能够在 JMeter 窗口的右半部分看到线程组的控制面板，如图 3-10 所示。

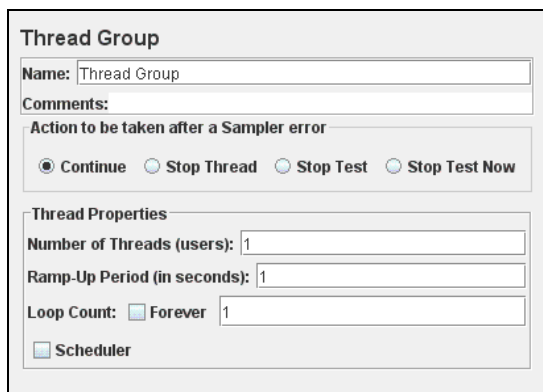


图 3-10 线程组的控制面板

首先为线程组起一个有意义的名字，在名称域中输入 Jakarta Users，接着设置线程数为 5，如图 3-11 所示。保持 Ramp-Up Period 的值不变（为 1 秒），这一设置会告诉 JMeter 启动并发用

户的时间间隔。例如，如果测试人员将 Ramp-Up Period 设置为 5 秒，那么 JMeter 会在 5 秒内将所有并发用户启动起来。因此假设我们有 5 个并发用户和 5 秒的 Ramp-Up Period，那么启动并发用户的间隔为 1 秒（5 并发用户 / 5 秒 = 1 用户每秒）。如果测试人员将该值设为 0，那么 JMeter 会立刻启动所有的并发用户。

最后在循环次数（Loop Count）中输入 2，这一属性会告诉 JMeter 重复测试多少次。如果测试人员设置的循环次数为 1，那么 JMeter 只会运行测试一遍。要让 JMeter 不断运行测试计划，请选中“(Forever) 永远”复选框。

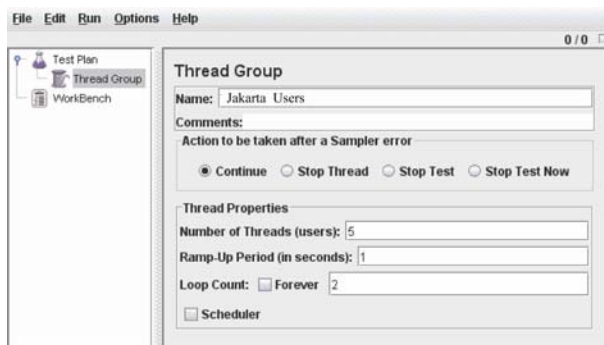


图 3-11 线程组 Jakarta Users



小贴士

对于大多数软件而言，测试人员在控制面板中做了某种修改后，必须手动确认一下。但是 JMeter 不同，它会自动接受用户在控制面板中做出的修改。假如测试人员改变了某个测试元件的名称，那么在测试人员离开控制面板后，JMeter 会使用新名称来更新测试树。

2. 添加默认 HTTP 请求属性

现在已经定义了并发用户数，下一步需定义并发用户需要进行的操作了。在这里，测试人员将学会如何设定 HTTP 请求的默认值。在后面测试人员将学会如何添加 HTTP 请求（测试元件），并使用此处设定的默认值。

首先从选中 Jakarta Users（线程组）测试元件开始。单击鼠标右键，在弹出的快捷菜单中选

择 “Add” → “Config Element” → “HTTP Request Defaults” 命令。接着选中这个新测试元件，查看它的控制面板，如图 3-12 所示。

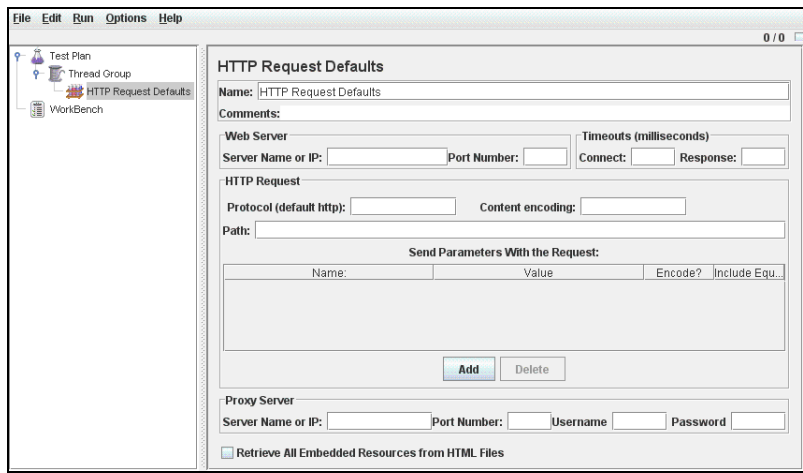


图 3-12 HTTP 请求默认值（HTTP Request Defaults）

像大多数 JMeter 测试元件一样，HTTP 请求默认值（HTTP Request Defaults）也有对应的控制面板。此处不修改名称域，保留原来的值。

让我们跳到下一个设置域——Server Name or IP。对于当前正在构建的这个测试计划，所有的请求都要发往 jakarta.apache.org，因此测试人员需要将 jakarta.apache.org 放到该设置域中。这个域是唯一需要设定的默认值，因此其他域就保留原来的值即可，如图 3-13 所示。

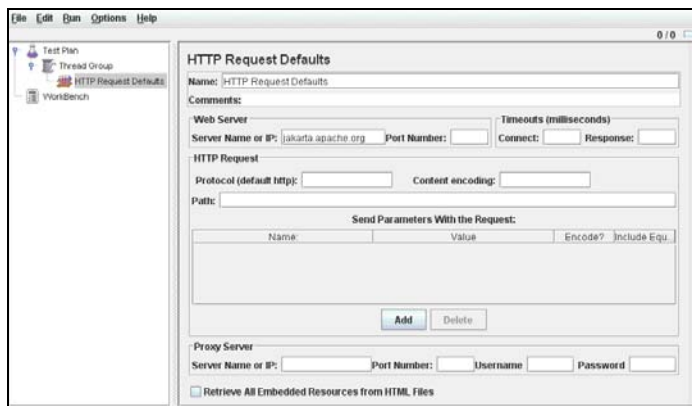


图 3-13 设置 Server Name or IP 域



小贴士

HTTP 请求默认值 (HTTP Request Defaults) 不会让 JMeter 去发送 HTTP 请求, 它只是定义了 HTTP 请求 (测试元件) 使用到的默认值。

3. 添加对 Cookie 的支持

通常所有 Web 测试都要支持 Cookie, 除非测试人员的应用系统很特别, 不使用 Cookie。要添加对 Cookie 的支持, 只需要为测试计划中的每一个线程组添加一个 HTTP Cookie 管理器 (HTTP Cookie Manager)。这样一来, 每一个测试线程都会拥有独立的 Cookie, 但是这些 Cookie 会在 HTTP 请求对象间共享。

要添加 HTTP Cookie 管理器 (HTTP Cookie Manager), 只需简单地选中线程组, 接着选择 “Add” → “Config Element” → “HTTP Cookie Manager” 命令 (既可以通过编辑菜单, 也可以通过右键弹出菜单)。

4. 添加 HTTP 请求

在测试计划中, 需要发送两个 HTTP 请求。第一个请求针对 Jakarta 主页 (<http://jakarta.apache.org/>), 而第二个请求针对项目指导页面 (<http://jakarta.apache.org/site/guidelines.html>)。



小贴士

JMeter 会按照它们在测试树中出现的顺序发送请求。

首先为线程组 (Jakarta Users) 添加一个 HTTP 请求 (Add→Sampler→HTTP Request)。接着在测试树中选中该 HTTP 请求, 并编辑其属性, 如图 3-14 所示。

(1) 将名称 (Name) 改为 “Home Page”。

(2) 将路径 (Path) 设置为 “/”。注意此处并不需要设定 Server Name, 原因在于测试人员已经在 HTTP 请求默认值 (HTTP Request Defaults) 中设定了默认值。

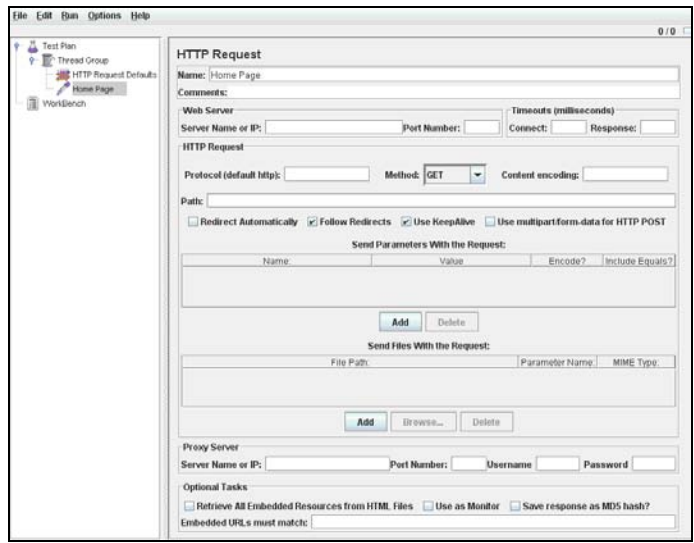


图 3-14 添加第一个 HTTP 请求（主页 <http://jakarta.apache.org/>）

接下来，添加第二个 HTTP 请求，并编辑其属性，如图 3-15 所示。

- (1) 将名称（Name）改为“Project Guidelines”。
- (2) 将路径（Path）设置为“/site/guidelines.html”。

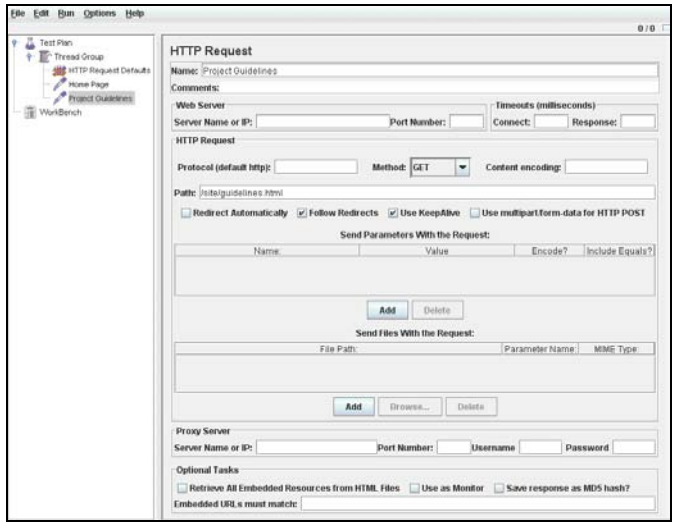


图 3-15 添加第二个 HTTP 请求（主页 <http://jakarta.apache.org/>）

5. 添加监听器用于查看/存储测试结果

测试人员为测试计划添加的最后一个测试元件就是监听器，如图 3-16 所示。该测试元件负责将所有 HTTP 请求的结果存储在一个文件中，并以可视化的模型加以展示。

选中线程组（Jakarta Users），并添加一个图形结果（Graph Results）监听器（Add → Listene → Graph Results）。接下来，测试人员需要指明保存测试结果的目录和文件名。测试人员既可以在 filename 输入域中填写，也可以通过单击“Browse”按钮来选择一个文件。

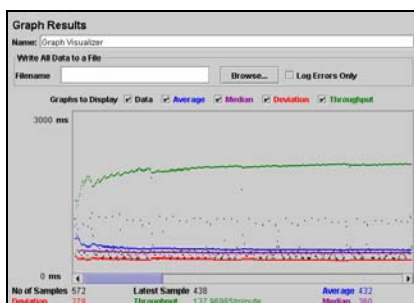


图 3-16 图形结果监听器

6. 登录 Web 站点

在上面描述的测试计划中不涉及登录，但是有些 Web 站点要求在执行特定操作前必须先登录。在 Web 浏览器中，登录界面通常就是一个表单（Form），其中有用户名和密码输入域，以及提交表单（Form）会用到的按钮。该按钮会产生一个 POST 请求，并将表单中的元素作为参数。

要使用 JMeter 完成登录，测试人员需要添加一个 HTTP 请求，并将方法设为 POST，如图 3-17 所示。测试人员需要知道表单使用的输入域名称和目标页面。所有这些信息都可以通过查看登录页面的代码来获取（如果这一点很难做

图 3-17 模拟 HTTP 登录请求

到，测试人员可以使用 JMeter 代理录制（JMeter Proxy Recorder）来实现）。将目标页面设置为提交按钮所在的页面。另外还需单击“Add”按钮两次，增加用户名和密码。有些时候登录表中还包含一些隐藏信息，它们也需要在这里添加。

3.5 录制 Web 测试脚本

3.5.1 使用代理录制 Web 性能测试脚本

本节主要介绍如何使用 JMeter 代理录制 Web 性能测试脚本。对于 JMeter 初学者而言，创建测试计划的一个简单办法就是使用 JMeter 代理。代理所要完成的工作就是录制发往服务器的请求。JMeter 代理目前不支持录制 HTTPS 协议，原因在于 HTTPS 是安全协议，代理无法破译其通信内容，并录制请求参数或者 Cookie。幸好存在多种解决该问题的办法，其中最简单的一种就是使用 Badboy（<http://www.badboy.com.au/>）工具。

1. 使用 JMeter 代理的基本步骤

- （1）启动 JMeter，在 Windows 中使用 jmeter.bat，在 UNIX 中使用 jmeter.sh。
- （2）选中测试树中的测试计划（Test Plan）。
- （3）用鼠标右键单击测试计划（Test Plan），添加一个新的线程组：Add→Thread Group，如图 3-18 所示。

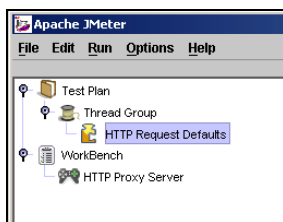


图 3-18 使用 JMeter 代理

- （4）选中线程组（Thread Group）。
- （5）单击鼠标右键，在弹出的快捷菜单中选择“Add”→“Config Element”→“HTTP Request Defaults”命令。
- （6）Protocol：输入“http”。
- （7）Server Name or IP：输入“jakarta.apache.org”。

(8) Path: 保留为空。

(9) Port Number: 输入“80”,如图 3-19 所示。

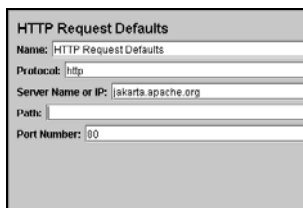


图 3-19 配置 HTTP 请求默认值 (HTTP Request Defaults)

(10) 选中工作台。

(11) 用鼠标右键单击工作台并添加 HTTP 代理 (Add→Non-test Elements→HTTPProxy Server)。

(12) Port 域: 输入“8088”,如图 3-20 所示。这一步骤指明了代理使用的端口号。

(13) Target Controller: 从下拉列表中选择“Test Plan > Thread Group”。这一步骤指明了代理录制的脚本会挂在测试树的哪个分支下。

(14) 单击“Patterns to Include”中的“Add”按钮,这会产生一个空白输入域。

(15) 输入“.**.html”。

(16) 单击“Patterns to Exclude”中的“Add”按钮,这会产生一个空白输入域。

(17) 输入“.**.gif”。

(18) 单击底部的“Start”按钮。

(19) 启动 Internet Explorer, 但是不关闭 JMeter。



小贴士

用户必须保证包含 (Include) 和排除 (Exclude) 样式的设定是正确的。下面是一些常用的

图片和页面类型的样式 :

```
.* - all
.*\*.png - png images
.*\*.gif - gif images
.*\*.jpg - jpeg images
.*\*.php
.*\*.jsp
```

```
.*\.html  
.*\.htm  
.*\.js
```

这里有些小技巧，在开始录制脚本前最好将浏览器的主页设为空白页。通过这种方法，可以减少 JMeter 在会话期间录制到不想要的页面访问的次数。针对不同站点录制脚本时，需要相应调整过滤样式。

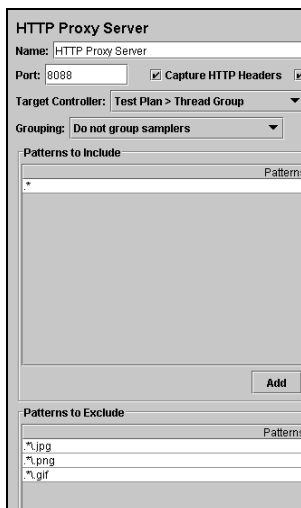


图 3-20 HTTP 代理服务器（HTTP Proxy Server）

- (20) 在 IE 中选择工具栏，选择“Tools”→“Internet Options”命令。
- (21) 选择“Connection”选项卡。
- (22) 单击“Lan Settings”按钮（应该在选项卡底部）。
- (23) 选中“Use a Proxy Server for Your Lan”选项，地址和端口号输入域应该变得可以修改了。
- (24) Address：输入“Localhost”或者是机器的 IP 地址。
- (25) Port：输入“8088”。
- (26) 单击“OK”按钮。
- (27) 再次单击“OK”按钮，这时测试人员将返回到浏览器主界面。
- (28) 在 IE 浏览器顶部的地址栏中，输入“<http://jakarta.apache.org/jmeter/index.html>”，接着按回车键。

(29) 随便单击 JMeter 页面上的几个链接。

(30) 关闭 Internet Explorer，将视线转回 JMeter 窗口上。

2. 重新检视测试计划

展开线程组后，测试人员应该能发现多个采样器，如图 3-21 所示。这个时候，测试计划就应该能够被保存了。如果前面忘记了添加默认 HTTP 请求设置，那么现在测试人员不得不手工删除采样器的服务器名（Servername）和端口（Port）。在当前例子中，没有默认的请求参数。如果所有页面都需要某个特定请求参数，那么测试人员需要在 HTTP 请求默认值中添加一行，以便保存该参数。

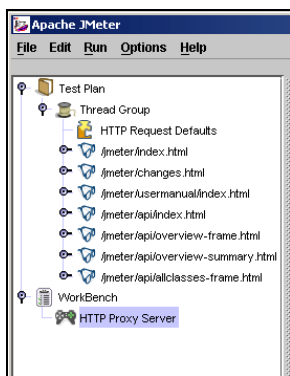


图 3-21 通过 JMeter 代理录制的脚本

(1) 选中线程组（Thread Group）。

(2) 单击鼠标右键，在弹出的快捷菜单中选择“Add”→“Listener”→“Aggregate Report”命令，添加一个聚合报告（Aggregate Report），如图 3-22 所示。聚合报告能够展现一些基本的统计信息。

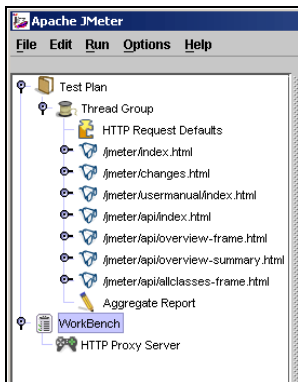


图 3-22 添加聚合报告

- (3) 选中线程组 (Thread Group)。
- (4) Number of Threads: 输入“5”，如图 3-23 所示。
- (5) Ramp-Up Period: 保持不变。
- (6) Loop Count: 输入“100”。

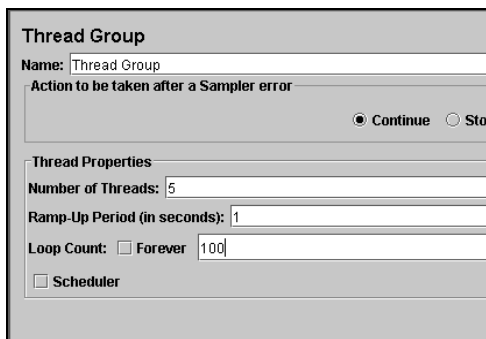


图 3-23 配置线程组

3. 运行测试

此刻，我们已经做好准备运行第一个 JMeter 测试脚本，看看会发生什么。首先保存测试计划。当测试人员准备运行测试时，有两种方式：

- Run→ Start（运行→启动）。
- 按“Ctrl + R”组合键。

在测试人员启动测试前，先选中聚合报告 (Aggregate Report)。在测试运行期间，统计信

息会不断变化直到测试结束。在测试结束后，聚合报告应该如图 3-24 所示。

Aggregate Report

Name: Aggregate Report

Write All Data to a File

Filename

Browse...

☐ Log Errors On

URL	# Samples	Average	Median	90% Line	Min
jmeterindex...	500	192.644	161	211	90
jmeterchan...	500	2186.54	2022	3135	670
jmeteruser...	500	1388.156	1272	2043	340
jmeterapiin...	500	225.85	180	250	100
jmeterapiu...	500	1197.198	1071	1803	261
jmeterapiu...	500	4115.756	3975	5598	1001
jmeterapiu...	500	1376.828	1262	2093	400
TOTAL	3500	1526.13885...	1192	3615	90

图 3-24 聚合报告

测试运行期间，JMeter 窗口的右上角应该有一个绿色小盒子，如图 3-25 所示。当测试结束后，小盒子应该会变灰。

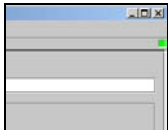


图 3-25 JMeter 运行提示

3.5.2 使用 Badboy 录制 Web 性能测试脚本

本节主要介绍如何使用 Badboy 录制 Web 性能测试脚本。由于测试工具 Badboy 支持对 HTTPS 协议的录制，因此可以作为 JMeter 代理录制的有益补充。用户可以从 <http://www.badboy.com.au/> 下载 Badboy 的安装文件，安装过程很简单，一路单击“下一步 (Next)”按钮即可。下面以“登录网络 U 盘”为例，介绍如何使用 Badboy 录制 Web 性能测试脚本。

1. 使用 Badboy 录制用户操作

(1) 启动 Badboy。首次启动 Badboy 时，录制按钮默认处于选中状态，如图 3-26 所示中的红色小圆点。

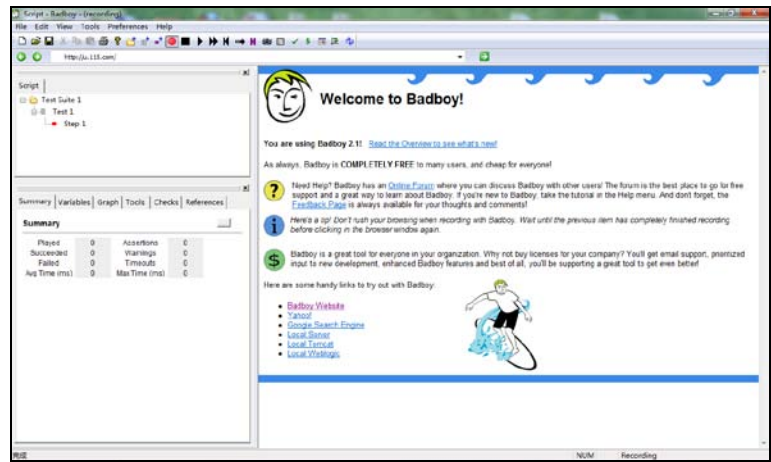


图 3-26 启动 Badboy

(2) 在 Badboy 工具地址栏中输入测试网址，然后按回车键。Badboy 工具会使用内置的浏览器访问对应的网址，如图 3-27 所示。



图 3-27 Badboy 访问测试网址

(3) 在 Badboy 工具打开的页面中完成登录所需的各项操作，接下来可以在左上角的脚本框中看到录制产生的测试脚本，如图 3-28 所示。



图 3-28 使用 Badboy 完成脚本录制

2. 导出 Badboy 测试脚本

(1) 在 Badboy 中完成脚本录制后，可以将测试脚本导出成 JMX 格式，以便后续供 JMeter 使用，如图 3-29 和图 3-30 所示。

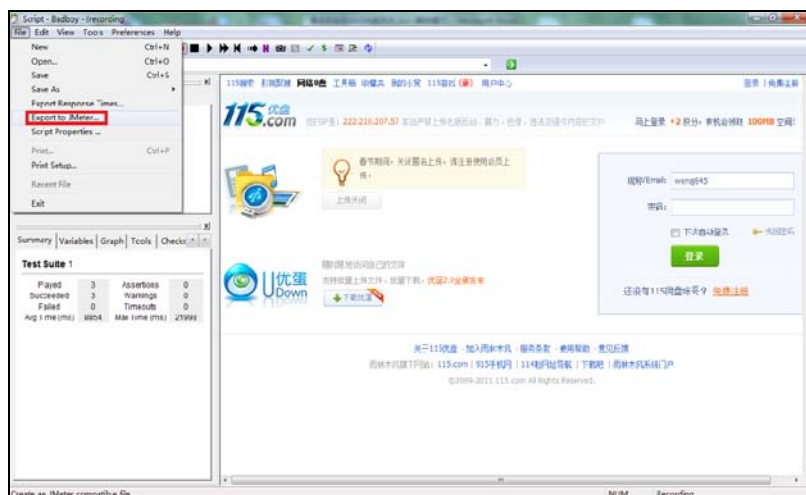


图 3-29 导出 Badboy 测试脚本 (1)

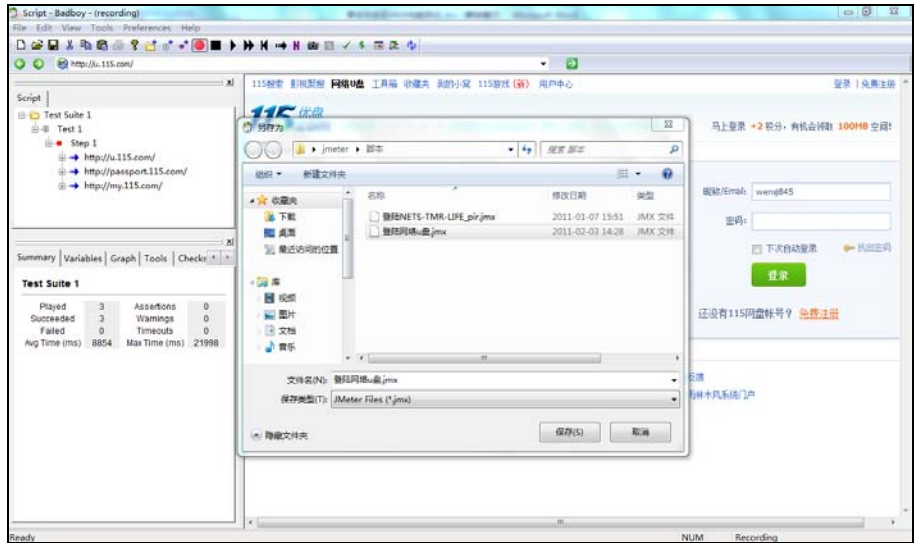


图 3-30 导出 Badboy 测试脚本（2）

（2）使用 JMeter 打开通过 Badboy 导出生成的测试脚本“登录网络 U 盘.jmx”，如图 3-31 所示。

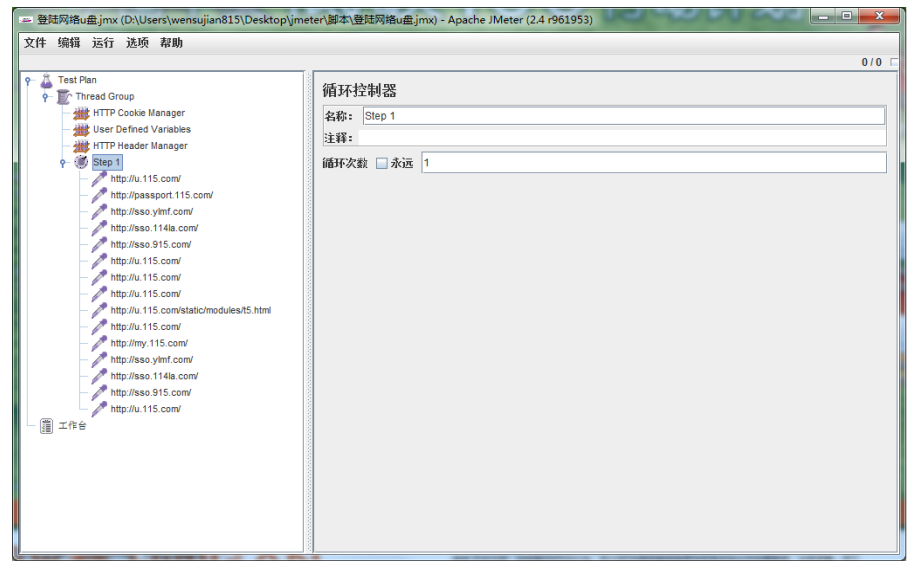


图 3-31 使用 JMeter 打开 Badboy 生成的测试脚本

（3）为测试脚本添加监听器，查看结果树和图形结果，如图 3-32 所示。

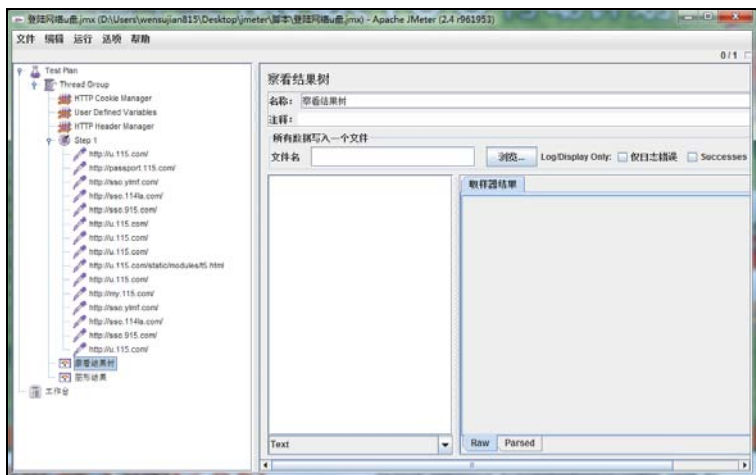


图 3-32 为测试脚本添加监听器

3. 运行测试

用户可以通过以下两种方式运行测试：

- Run→Start（运行→启动）。
- 按“Ctrl+R”组合键。

现在让我们观察监听器（查看结果树），监视测试的运行情况，如图 3-33 所示。

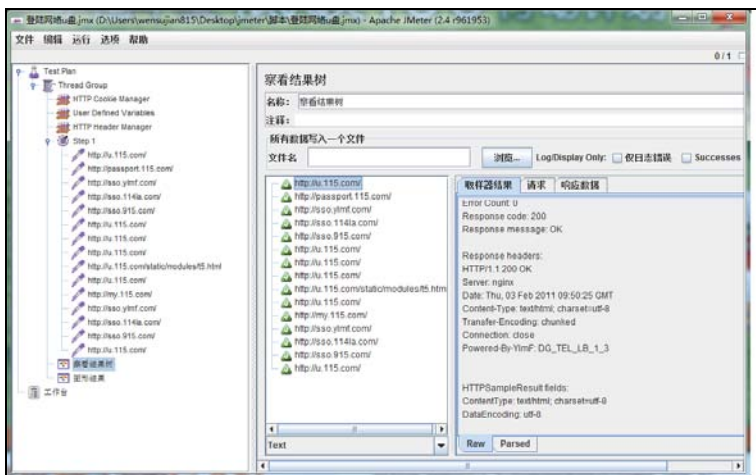


图 3-33 查看结果树

从图 3-33 中我们可以看到测试运行正常，所有 HTTP 请求都得到了服务器的正确响应，性能测试脚本录制成功。如图 3-34 所示，从中可以看到测试的图形结果。

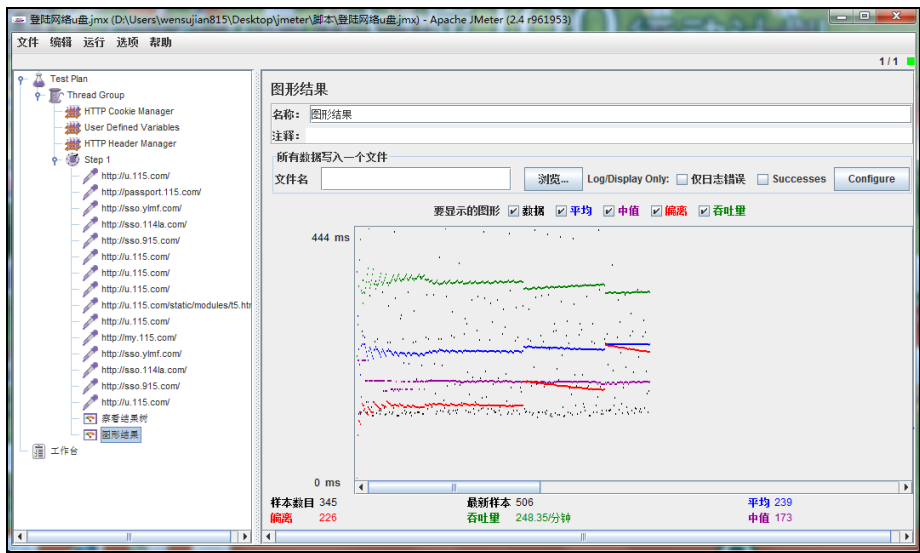


图 3-34 图形结果

3.6 创建高级 Web 测试计划

在这一节中，将介绍如何创建高级 Web 测试计划，以便测试 Web 站点。

1. 使用 URL 回写来处理用户会话

如果测试人员的 Web 应用系统使用 URL 回写而非 Cookie 来保存会话信息，那么测试人员需要做一些额外的工作来测试 Web 站点。

为了正确回应 URL 回写，JMeter 需要解析从服务器收到的 HTML，并得到唯一的会话 ID。测试人员需要使用合适的 HTTP URL 回写修改器来完成这一点。测试人员只需简单地将会话 ID 参数的名称放入修改器中，修改器就会找到会话 ID，并将其放入每个请求之中。如果请求之中已经有了会话 ID，那么它就会被替换掉。如果选中了“Cache Session ID?”选项，那么最近一个被找到的会话 ID 将会保存下来。当前一个 HTTP 采样不包含会话 ID 时，就会使用到保存下来的会话 ID 值。

URL 回写例子：

如图 3-35 所示，其中显示了一个使用 URL 回写的测试计划。请注意，URL 回写修改器(HTTP

URL Re-writing Modifier) 被添加在简单控制器 (Simple Controller) 之下, 这就意味着它只影响简单控制器下的请求。

如图 3-36 所示, 可以看到 URL 回写修改器的 GUI, 其中只有一个输入域供用户指明会话 ID 参数的名称。这里还有一个选项, 用于指明会话 ID 是路径的一部分 (使用 “;” 划分), 而不是作为请求的参数。

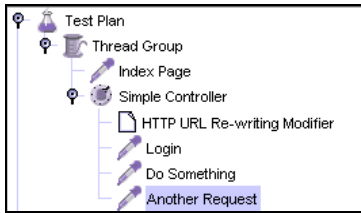


图 3-35 URL 回写例子的测试树

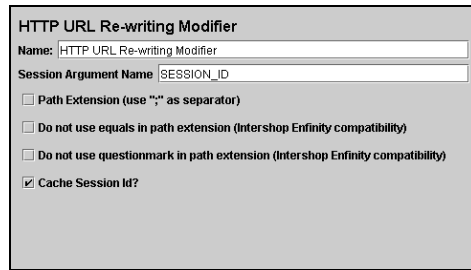


图 3-36 HTTP URL 回写修改器

2. 使用 HTTP 信息头管理器

使用 HTTP 信息头管理器, 可以帮助测试人员设定 JMeter 发送的 HTTP 请求头部所包含的信息。HTTP 信息头中包含有 “User-Agent”、“Pragma”、“Referer” 等属性。

HTTP 信息头管理器 (HTTP Header Manager) 就像 HTTP Cookie 管理器 (HTTP Cookie Manager), 应该尽可能地放在线程组一级。除非因为某些原因, 测试人员希望不同的 HTTP 请求使用不同的 HTTP 信息头。

3.7 本章小结

本章首先对 JMeter 图形用户界面的基本操作做了一个概要介绍, 为后续开发 JMeter 性能测试脚本打下了基础; 接着介绍了 JMeter 的各种常用测试元件, 以及 JMeter 测试脚本的各项规则, 并介绍了属性和变量; 之后介绍了如何录制和开发 Web 性能测试脚本; 最后介绍了开发 Web 性能测试脚本所需的一些进阶知识。

第 11 章

JMeter进阶知识

11.1 详解 JMeter 函数和变量

JMeter 函数可以被认为是某种特殊的变量，它们可以被采样器或者其他测试元件所引用。函数调用的语法如下：

```
${__functionName(var1,var2,var3)}
```

其中，`__functionName` 匹配被调用的函数名称。用圆括号包含函数的形参，例如 `${__time(YMD)}`，不同函数要求的参数也不同。有些 JMeter 函数不要求参数，则可以不使用圆括号，例如 `${__threadNum}`。

如果一个函数的参数中包含逗号，那么必须对逗号进行转义（使用“\”），否则 JMeter 会把逗号当成参数分隔符。例如：

```
${__time(EEE\, d MMM yyyy)}
```

变量引用的语法如下：

```
${VARIABLE}
```

如果测试计划中引用了未定义的变量或者函数，那么 JMeter 并不会报告/记录错误信息，引用返回的值就是引用自身。例如，假设字符串 UNDEF 没有被定义为变量，那么 `${UNDEF}` 返回的值就是 `${UNDEF}`。变量、函数（包括属性）都是大小写敏感的。JMeter 2.3.1 及其后续版本会剔除参数名中的空格，例如，`${__Random(1.63, LOTTERY)}` 中的“LOTTERY”会被“LOTTERY”所代替。



小贴士

属性不同于变量。变量对线程而言是局部的，所有线程都可以访问属性，就使用 `__P` 或者

__property 函数。

如表 11-1 所示为 JMeter 内置函数的列表（按类型划分）。

表 11-1 JMeter 内置函数列表

函数类型	函数名称	注释
Information	threadNum	get thread number
Information	machineName	get the local machine name
Information	time	return current time in various formats
Information	log	log (or display) a message (and return the value)
Information	logn	log (or display) a message (empty return value)
Input	StringFromFile	read a line from a file
Input	FileToString	read an entire file
Input	CSVRead	read from CSV delimited file
Input	XPath	Use an XPath expression to read from a file
Calculation	counter	generate an incrementing number
Calculation	intSum	add int numbers
Calculation	longSum	add long numbers
Calculation	Random	generate a random number
Scripting	BeanShell	run a BeanShell script
Scripting	javaScript	process JavaScript (Mozilla Rhino)
Scripting	jexl	evaluate a Commons Jexl expression
Properties	property	read a property
Properties	P	read a property (shorthand method)
Properties	setProperty	set a JMeter property
Variables	split	Split a string into variables
Variables	V	evaluate a variable name
Variables	eval	evaluate a variable expression
Variables	evalVar	evaluate an expression stored in a variable
String	regexFunction	parse previous response using a regular expression
String	char	generate Unicode char values from a list of numbers
String	unescape	Process strings containing Java escapes (e.g. \n & \t)
String	unescapeHtml	Decode HTML-encoded strings
String	escapeHtml	Encode strings using HTML encoding

1. 使用函数可以做什么

目前有两种类型的函数：用户定义的静态值（或者变量）和 JMeter 内置函数。

当需要编译测试树或者提交运行时，用户可以使用自定义变量来代替常用的静态值。这种替换只在测试的开始阶段执行一次。一个典型的应用就是使用自定义变量来替换所有 HTTP 请求的 DOMAIN 域，例如，做出轻微改动，就可以让同一个测试脚本适配多个服务器。

需要注意，目前变量不支持嵌套；例如 `${Var${N}}` 不能正常工作。但是在 JMeter 2.2 及其以后版本中，可以借助函数 `__V(variable)` 来达成嵌套变量的目的（如 `_${__V(Var${N})}`）。在早期的 JMeter 版本中可以使用 `__BeanShell(vars.get("Var${N}"))`。

这种类型的替换也可以不用函数来实现，但是就不像使用函数时那么直观和方便。用户可以创建默认配置测试元件，它们会填充采样器中的空白设置。

使用 JMeter 内置函数，用户可以基于前面的服务器响应数据、函数所在线程、当前时间或者其他资源来动态地计算变量值。这些变量的值会在整个测试期间针对每个请求动态更新。



小贴士

函数可以在多个线程间共用。在测试计划中每次函数调用，都是采用独立的函数实例。

2. 函数和变量可以被用在哪里

函数和变量理论上可以被用在任何测试元件的任何输入域之中（除了测试计划之外，见下面的内容）。有些输入域不支持随机数组，因为它们只接受数字，这样一来就不支持函数。当然，大多数输入域支持函数。

将函数用于测试计划（Test Plan）的设置时，会受到一些限制。此种情况下，JMeter 线程的变量在函数被处理时还没有被设定，因此变量作为参数传递时没有初始化，函数引用当然不会生效。如此一来，`split()`、`regex()` 及变量赋值函数就都不能正常工作。函数 `threadNum()` 同样不能正常工作，该函数在测试计划层没有意义。在测试计划中，函数 `intSum`、`longSum`、`machineName`、`BeanShell`、`javaScript`、`jexl`、`random`、`time`、`property functions`、`log functions` 应该能正常工作。

配置元件是通过一个独立线程处理的。因此函数（如 `__threadNum`）不能在这些测试元件（如用户定义的变量）之中正常工作。另外还需要注意，在用户定义的变量（UDV）中定义的变量，在 UDV 被处理前是不能使用的。



小贴士

当在 SQL 代码中引用变量/函数时，需要为文本字符串加上必要的引号。例如，使用：

```
SELECT item from table where name='${VAR}'
```

而非:

```
SELECT item from table where name=${VAR}
```

(除非 VAR 自身就包含引号)。

3. 怎样引用函数和变量

在测试元件中引用某个变量，可以通过使用 “\${” 和 “}” 将变量名括起来实现。

函数使用相同的办法加以引用，但是依据惯例，函数名以 “__” 开头，以区别于变量名。部分函数会携带参数，参数放在圆括号中，以逗号加以分隔。如果函数没有参数，那么可以省略圆括号。

如果参数值中包含逗号，必须对其加以转义。如果测试人员需要在参数值中包含一个逗号，可以这样转义：“\,”。这主要影响脚本函数，例如 JavaScript、BeanShell、Jexl 有必要对脚本方法调用中的所有逗号加以转义。例如：

```
${__BeanShell(vars.put("name\\","value"))}
```

另外，测试人员还有一种选择，即将脚本定义为一个变量，例如，在测试计划中定义：

```
SCRIPT vars.put("name","value")
```

脚本可以如下般引用：

```
${__BeanShell(${SCRIPT})}
```

这里没有必要对 SCRIPT 变量的内容进行转义，因为函数的调用先于变量被其值所替换。该方法适合于 BSF 或者 BeanShell 采样器，这两种采样器可以用于测试 JavaScript、Jexl 和 BeanShell 脚本。

函数可以引用变量及其他函数，例如 \${__XPath(\$__P(xpath.file),\${XPATH})}，使用 “xpath.file” 作为文件名，变量 XPATH 的内容作为搜索表达式。

JMeter 提供了一个工具，用来帮助测试人员使用各种内置函数实现函数调用。使用该工具，只需复制-粘贴。工具不会为测试人员自动转义值，因为函数可以作为其他函数的参数，测试人员应该只对文本进行转义。



小贴士

如果一个字符串既包含反斜线 (“\”)，又包含函数或者变量引用，那么出现在“\$”、“,”或者“\”之前的反斜线会被移除。这个操作对于嵌套函数 (被嵌套的函数表达式包含逗号或者

\$() 是有必要的。如果字符串中不包含函数和变量引用，那么出现在“\$”、“,”或者“\”之前的

反斜线就不会被移除。

用户可以使用__logn() 函数来报告变量或者函数的值。__logn() 函数可以在测试计划中的任何地方被引用，前提条件是被报告的值已经被定义。另外，Java 请求采样器可以被用来产生一个包含变量引用的采样；输出结果会在合适的监听器中展示。JMeter 2.3 及其以后版本中包含一个 Debug Sampler，可以使用它来展示变量的值（如在查看结果树中展示）。



小
贴
士

如果测试人员定义了一个用户定义静态变量，且该变量名与 JMeter 内置函数名相同，那么测试人员的静态变量就会覆盖同名内置函数。

4. 函数助手对话框

测试人员可以在 JMeter 的选项菜单中找到函数助手对话框（“Function Helper”对话框），如图 11-1 所示。

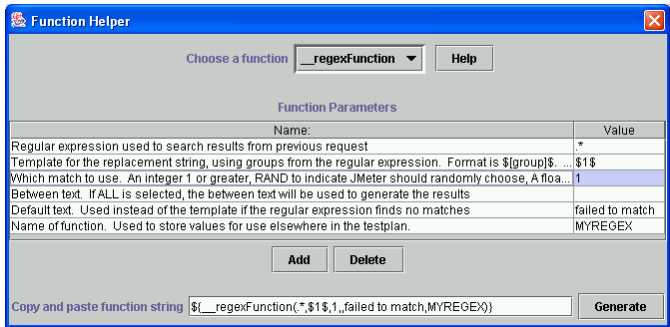


图 11-1 函数助手（Function Helper）对话框

使用函数助手，测试人员可以从下拉列表选择一个函数，并为其参数设定值。在图 11-1 中，表格的左边一列是函数参数的简要描述，右边一列是供用户填充参数的值。不同函数要求的参数也不同。

当测试人员完成以上设置后，请单击“Generate”按钮，函数助手会为测试人员生成函数调用所需的字符串，测试人员所要做的只是将它复制-粘贴到测试计划中去。

5. 常用 JMeter 函数

1) __regexFunction

正则表达式函数可以使用正则表达式（用户提供的）来解析前面的服务器响应（或者是某个变量值）。函数会返回一个有模板的字符串，其中携带有可变的值。

`__regexFunction` 还可以被用来保存值，以便供后续使用。在函数的第 6 个参数中，测试人员可以指定一个引用名。在函数执行以后，测试人员可以使用用户定义值的语法来获取同样的值。例如，如果测试人员输入“`refName`”作为第 6 个参数，那么测试人员可以使用：

- `${refName}` 来引用第 2 个参数（Template for the replacement string）的计算结果，这依赖于函数的解析结果。
- `${refName_g0}` 来引用函数解析后发现的所有匹配结果。
- `${refName_g1}` 来引用函数解析后发现的第一个匹配组合。
- `${refName_g#}` 来引用函数解析后发现的第 n 个匹配组合。
- `${refName_matchNr}` 来引用函数总共发现的匹配组合数目。

参数如表 11-2 所示。

表 11-2 参数描述

函数参数	描述	是否必需
第 1 个参数	第 1 个参数是用于解析服务器响应数据的正则表达式。它会找到所有匹配项。如果测试人员希望将表达式中的某部分应用在模板字符串中，一定记得为其加上圆括号。例如， <code></code> 。这样就会将链接的值存放到第一个匹配组合中（这里只有一个匹配组合）。又如， <code><input type="hidden" name="(.)" value="(.)"></code> 。在这个例子中，链接的 <code>name</code> 作为第一个匹配组合，链接的 <code>value</code> 会作为第二个匹配组合。这些组合可以用在测试人员的模板字符串中	是
第 2 个参数	这是一个模板字符串，函数会动态填写字符串的部分内容。要在字符串中引用正则表达式捕获的匹配组合，请使用语法： <code>\$(group_number)\$</code> 。例如 <code>\$1\$</code> 或者 <code>\$2\$</code> 。测试人员的模板可以是任何字符串	是
第 3 个参数	<p>第 3 个参数告诉 JMeter 使用第几次匹配。测试人员的正则表达式可能会找到多个匹配项。对此，测试人员有 4 种选择：</p> <ul style="list-style-type: none"> ■ 整数，直接告诉 JMeter 使用第几个匹配项。“1”对应第一个匹配，“2”对应第二个匹配，以此类推 ■ <code>RAND</code>，告诉 JMeter 随机选择一个匹配项 ■ <code>ALL</code>，告诉 JMeter 使用所有匹配项，为每一个匹配项创建一个模板字符串，并将它们连接在一起 ■ 浮点值 0 到 1 之间，根据公式（找到的总匹配数目*指定浮点值）计算使用第几个匹配项，计算值向最近的整数取整 	否，默认为 1

第 4 个参数	如果在上一个参数中选择了“ALL”，那么这第 4 个参数会被插入到重复的模板值之间	否
第 5 个参数	如果没有找到匹配项返回的默认值	否
第 6 个参数	重用函数解析值的引用名，参见上面内容	否
第 7 个参数	输入变量名称。如果指定了这一参数，那么该变量的值就会作为函数的输入，而不再使用前面的采样结果作为搜索对象	否

2) __counter

每次调用计数器函数都会产生一个新值，从 1 开始每次加 1。计数器既可以被配置成针对每个虚拟用户是独立的，也可以被配置成所有虚拟用户公用的。如果每个虚拟用户的计数器是独立增长的，那么通常被用于记录测试计划运行了多少遍。全局计数器通常被用于记录发送了多少次请求。

计数器使用一个整数值来记录，允许的最大值为 2,147,483,647。

目前计数器函数实例是独立实现的（JMeter 2.1.1 及其以前版本，使用一个固定的线程变量来跟踪每个用户的计数器，因此多个计数器函数会操作同一个值）。全局计数器（FALSE）每个计数器实例都是独立维护的。

参数如表 11-3 所示。

表 11-3 参数描述

函数参数	描述	是否必需
第 1 个参数	True，如果测试人员希望每个虚拟用户的计数器保持独立，与其他用户的计数器相区别。false，全局计数器	是
第 2 个参数	重用计数器函数创建值的引用名。测试人员可以这样引用计数器的值：\${refName}。这样一来，测试人员就可以创建一个计数器后，在多个地方引用它的值（JMeter 2.1.1 及其以前版本，这个参数是必需的）	否

3) __threadNum

函数__threadNum 只是简单地返回当前线程的编号。线程编号不依赖于线程组，这就意味着从函数的角度来看，某个线程组的线程#1 和另一个线程组的线程#1 是没有区别的。另外，该函数没有参数。



小贴士

这一函数不能用在任何配置元件中（如用户定义的变量），原因在于配置元件是由一个独立线程运行的。另外在测试计划（Test Plan）中使用也是没有意义的。

4) __intSum

函数__intSum 可以被用来计算两个或者更多整数值的合。

参数如表 11-4 所示。

表 11-4 参数描述

函数参数	描述	是否必需
第 1 个参数	第 1 个整数值	是
第 2 个参数	第 2 个整数值	是
第 n 个参数	第 n 个整数值	否
最后一个参数	重用函数计算值的引用名。如果用户指定了这一参数，那么引用名中必须包含一个非数字字母，否则它会被当成另一个整数值，而被函数用于计算	否



小贴士

JMeter 2.3.1 及其以前版本，要求必须有引用名参数。后续 JMeter 版本中，引用名是可选的参数，但是引用名不能是整数值。

5) __longSum

函数__ longSum 可以被用来计算两个或者更多长整型值的合。

参数如表 11-5 所示。

表 11-5 参数描述

函数参数	描述	是否必需
第 1 个参数	第 1 个长整型值	是
第 2 个参数	第 2 个长整型值	是
第 n 个参数	第 n 个长整型值	否
最后一个参数	重用函数计算值的引用名。如果用户指定了这一参数，那么引用名中必须包含一个非数字字母，否则它会被当成另一个长整型值，而被函数用于计算	否

6) __StringFromFile

函数__StringFromFile 可以被用来从文本文件中读取字符串。这对于需要大量可变数据的测试很有用。例如，当测试一个银行应用系统时，测试人员可能需要 100 条甚至 1000 条账户信息。

使用配置元件 CSV Data Set Config ，也能达到相同的目的，而且方法更简单。但是该配置元件目前不支持多输入文件。

每次调用函数，都会从文件中读取下一行。当到达文件末尾时，函数又会从文件开始处重新读取，直到最大循环次数。如果在一个测试脚本中对该函数有多次引用，那么每一次引用都会独立打开文件，即使文件名是相同的（如果函数读取的值，在脚本其他地方也有使用，那么就需要为每一次函数调用指定不同的变量名）。

如果在打开或者读取文件时发生错误，那么函数就会返回字符串 “**ERR**”。

参数如表 11-6 所示。

表 11-6 参数描述

函数参数	描述	是否必需
文件名	文件名（可以使用相对于 JMeter 启动目录的相对路径）。如果要在文件名中使用可选的序列号，那么文件名必须适合转成十进制格式。参考下面的例子	是
变量名	一个引用名（refName）的目的是复用这一函数创建的值。可以使用语法 \${refName}来引用函数创建的值。默认值为 “StringFromFile_”	否
初始序列号	初始序列号（如果省略这一参数，终止序列号会作为一个循环计数器）	否
终止序列号	终止序列号（如果省略这一参数，序列号会一直增加下去，不会受到限制）	否

当打开或者重新打开文件时，文件名参数将会被解析。

每次执行函数时，引用名参数（如果支持）将会被解析。

使用序列号：当使用可选的序列号时，文件名需要使用格式字符串 java.text.DecimalFormat。当前的序列号会作为唯一的参数。如果不指明可选的初始序列号，就使用文件名作为起始值。一些有用的格式序列如下：

- #：插入数字，不从零开始，不包含空格。
- 000：插入数字，包含 3 个数字组合，不从零开始。

例如：

```
pin#'.dat -> pin1.dat, ... pin9.dat, pin10.dat, ... pin9999.dat
pin000'.dat -> pin001.dat ... pin099.dat ... pin999.dat ... pin9999.dat
pin'.dat# -> pin.dat1, ... pin.dat9 ... pin.dat999
```

如果不希望某个格式字符被翻译，测试人员需要为它加上单引号。注意 “.” 是格式字符，必须被单引号所包含。

如果省略了初始序列号，而终止序列号参数将会作为循环计数器，文件将会被使用指定的次数。例如：

- \${_StringFromFile(PIN#'.DAT,,1,2)}：读取 PIN1.DAT, PIN2.DAT。
- \${_StringFromFile(PIN.DAT,,,2)}：读取 PIN.DAT 两次。

7) __machineName

函数__machineName 返回本机的主机名。

参数如表 11-7 所示。

表 11-7 参数描述

函数参数	描述	是否必需
变量名	重用函数计算值的引用名	否

8) `__javaScript`

函数 `__javaScript` 可以用来执行 JavaScript 代码片段（非 Java），并返回结果值。JMeter 的 `__javaScript` 函数会调用标准的 JavaScript 解释器。JavaScript 会作为脚本语言使用，因此测试人员可以做相应的计算。

在脚本中可以访问如下一些变量。

- **Log**: 该函数的日志记录器。
- **Ctx**: `JmeterContext` 对象。
- **Vars**: `JmeterVariables` 对象。
- **threadName**: 字符串包含当前线程名称（在 2.3.2 版本中它被误写为“theadName”）。
- **sampler**: 当前采样器对象（如果存在）。
- **sampleResult**: 前面的采样结果对象（如果存在）。
- **props**: JMeter 属性对象。

Rhinoscript 允许通过它的包对象来访问静态方法。例如，用户可以使用如下方法访问 `JMeterContextService` 静态方法：

`Packages.org.apache.jmeter.threads.JMeterContextService.getTotalThreads()`



小贴士

JMeter 不是一款浏览器，它不会执行从页面下载的 JavaScript。

参数如表 11-8 所示。

表 11-8 参数描述

函数参数	描述	是否必需
JavaScript 代码片段	待执行的 JavaScript 代码片段。例如： <ul style="list-style-type: none">■ <code>new Date()</code>: 返回当前日期和时间■ <code>Math.floor(Math.random()*\${maxRandom}+1)</code>：在 0 和 变量 <code>maxRandom</code> 之间的随机数■ <code>\${minRandom}+Math.floor(Math.random()*\${maxRandom}-\${minRandom}+1)</code>: 在变量 <code>minRandom</code> 和 <code>maxRandom</code> 之间的随机数■ <code>"\${VAR}"=="abcd"</code>	是
变量名	重用函数计算值的引用名	否



小贴士

请记得为文本字符串添加必要的引号。另外，如果表达式中有逗号，请确保对其转义。例如，`${__javaScript('${sp}'.slice(7,99999))}`，对 7 之后的逗号进行了转义。

9) __Random

函数__Random 会返回指定最大值和最小值之间的随机数。

参数如表 11-9 所示。

表 11-9 参数描述

函数参数	描述	是否必需
最小值	最小数值	是
最大值	最大数值	是
变量名	重用函数计算值的引用名	否

10) __CSVRead

函数__CSVRead 会从 CSV 文件读取一个字符串（请注意与 StringFromFile 的区别）。



小贴士

JMeter 1.9.1 以前的版本仅支持从单个文件中读取，JMeter 1.9.1 及其以后版本支持从多个文件中读取。

在大多数情况下，新配置元件 CSV Data Set 更好用一些。

当对某个文件进行第一次读取时，文件将被打开并读取到一个内部数组中。如果在读取过程中找到了空行，函数就认为到达文件末尾了，即允许拖尾注释（这一特性是 JMeter 1.9.1 版本引入的）。

后续所有对同一个文件名的引用，都使用相同的内部数组。另外，文件名大小写对函数调用很重要，哪怕操作系统不区分大小写，`CSVRead(abc.txt,0)`和 `CSVRead(aBc.txt,0)`会引用不同的内部数组。

使用*ALIAS 特性可以多次打开同一个文件，另外还能缩减文件名称。

每一个线程都有独立的内部指针指向文件数组中的当前行。当某个线程第一次引用文件时，函数会为线程在数组中分配下一个空闲行。如此一来，任何一个线程访问的文件行，都与其他线程不同（除非线程数大于数组包含的行数）。



小贴士

默认情况下，函数会在遇到的每一个逗号处断行。如果测试人员希望在输入的列中使用逗号，那么测试人员需要换一个分隔符（通过设置属性 `csvread.delimiter` 来实现），且该符号没有在 CSV 文件的任何列中出现。

参数如表 11-10 所示。

表 11-10 参数描述

函数参数	描述	是否必需
文件名	设置从哪个文件读取（或者*ALIAS）	是
列数	从文件的哪一列读取。0=第一列， 1= 第二列，依此类推。“next”为走到文件的下一行。*ALIAS 为打开一个文件，并给它分配一个别名	是

例如，测试人员可以用如下参数来设置某些变量：

```
COL1a ${__CSVRead(random.txt,0)}
COL2a ${__CSVRead(random.txt,1)}${__CSVRead(random.txt,next)}
COL1b ${__CSVRead(random.txt,0)}
COL2b ${__CSVRead(random.txt,1)}${__CSVRead(random.txt,next)}
```

上面的例子会从一行中读取两列，接着从下一行中读取两列。如果所有变量都在同一个前置处理器中（用户参数上定义的），那么行都是顺序读取的。否则，不同线程可能会读取不同的行。



小贴士

这一函数并不适合于读取很大的文件，因为整个文件都会被存储到内存之中。对于较大的文件，请使用配置元件 CSV Data Set 或者 StringFromFile 。

11) __property

函数 `__property` 会返回一个 JMeter 属性的值。如果函数找不到属性值，而又没有提供默认值，则它会返回属性的名称。

例如，

- `${__property(user.dir)}`：返回属性 `user.dir` 的值。
- `${__property(user.dir,UDIR)}`：返回属性 `user.dir` 的值，并保存在变量 `UDIR` 中。
- `${__property(abcd,ABCD,atod)}`：返回属性 `abcd` 的值（如果属性没有定义，返回“atod”），并保存在变量 `ABCD` 中。

- `${__property(abcd,,atod)}`：返回属性 `abcd` 的值（如果属性没有定义，返回“`atod`”），但是并不保存函数的返回值。

参数如表 11-11 所示。

表 11-11 参数描述

函数参数	描述	是否必需
属性名	获取属性值、所需的属性名	是
变量名	重用函数计算值的引用名	否
默认值	属性未定义时的默认值	否

12) `_P`

函数 `_P` 是一个简化版的属性函数，目的是使用在命令行中定义的属性。不同于函数 `_property`，本函数没有提供选项用于设置保存属性值的变量。另外，如果没有设置默认值，默认值自动设为 1。之所以选择 1，原因在于它对于很多常见测试变量都是一个合理值，例如，循环次数、线程数、启动线程耗时间等。

例如：定义属性值：

```
jmeter -Jgroup1.threads=7 -Jhostname=www.realhost.edu
```

获取值如下。

- `${__P(group1.threads)}`：返回属性 `group1.threads` 的值。
- `${__P(group1.loops)}`：返回属性 `group1.loops` 的值。
- `${__P(hostname,www.dummy.org)}`：返回属性 `hostname` 的值，如果没有定义该属性则返回值 `www.dummy.org`。

在上面的例子中，第一个函数调用返回 7，第二个函数调用返回 1，而最后一个函数调用返回 `www.dummy.org`（除非这些属性在其他地方有定义）。

参数如表 11-12 所示。

表 11-12 参数描述

函数参数	描述	是否必需
属性名	获取属性值、所需的属性名	是
默认值	属性未定义时的默认值。如果省略此参数，默认值自动设为 1	否

13) `__log`

函数 `__log` 会记录一条日志，并返回函数的输入字符串。

参数如表 11-13 所示。

表 11-13 参数描述

函数参数	描述	是否必需
待记录字符串	一个字符串	是
日志级别	OUT、ERR、DEBUG、INFO（默认）、WARN 或者 ERROR	否
可抛弃的文本	如果非空，会创建一个可抛弃的文本传递给记录器	否
注释	如果存在，注释会在字符串中展示，用于标识日志记录了什么	否

OUT 和 ERR 的日志级别，将会分别导致输出记录到 System.out 和 System.err 中。在这种情况下，输出总是会被打印（它不依赖于当前的日志设置）。

例如，

- `${__log(Message)}`：写入日志文件，形如 “...thread Name : Message”。
- `${__log(Message,OUT)}`：写到控制台窗口。
- `${__log(${VAR},,,VAR=)}`：写入日志文件，形如 “...thread Name VAR=value”。

14) __logn

函数__logn 会记录一条日志，并返回空字符串。

参数如表 11-14 所示。

表 11-14 参数描述

函数参数	描述	是否必需
待记录字符串	一个字符串	是
日志级别	OUT, ERR, DEBUG, INFO（默认）, WARN 或者 ERROR	否
可抛弃的文本	如果非空，会创建一个可抛弃的文本传递给记录器	否

OUT 和 ERR 的日志级别，将会分别导致输出记录到 System.out 和 System.err 中。在这种情况下，输出总是会被打印（它不依赖于当前的日志设置）。

例如，`${__logn(VAR1=${VAR1},OUT)}`：将变量值写到控制台窗口中。

15) __BeanShell

函数__BeanShell 会执行传递给它的脚本，并返回结果。

关于 BeanShell 的详细资料，请参考 BeanShell 的 Web 站点：<http://www.beanshell.org/>。

需要注意，测试脚本中每一个独立出现的函数调用，都会使用不同的解释器，但是后续对函数调用的援引会使用相同的解释器。这就意味着变量会持续存在，并跨越函数调用。

单个函数实例可以从多个线程调用。另外，该函数的 `execute()`方法是同步的。

如果定义了属性 “beanshell.function.init”，那么它会作为一个源文件传递给解释器。这样就

可以定义一些通用方法和变量。在 bin 目录中有一个初始化文件的例子：`BeanShellFunction.bshrc`。

如下变量在脚本执行前就已经设置了。

- `log`: 函数 `BeanShell(*)` 的记录器。
- `ctx`: 目前的 JMeter Context 变量。
- `vars`: 目前的 JMeter 变量。
- `props`: JMeter 属性对象。
- `threadName`: 线程名（字符串）。
- `sampler`: 当前采样器（如果存在）。
- `sampleResult`: 当前采样器（如果存在）。

“*”意味着该变量在 JMeter 使用初始化文件之前就已经设置了。其他变量在不同调用之间可能会发生变化。

参数如表 11-15 所示。

表 11-15 参数描述

函数参数	描述	是否必需
BeanShell 脚本	一个 BeanShell 脚本（不是文件名）	是
变量名	重用函数计算值的引用名	否

例如，

- `${__BeanShell(123*456)}`: 回 56088。
- `${__BeanShell(source("function.bsh"))}`: 行在 `function.bsh` 中的脚本。



小贴士

请记得为文本字符串及代表文本字符串的 JMeter 变量添加必要的引号。

16) __plit

函数 `__split` 会通过分隔符来拆分传递给它的字符串，并返回原始的字符串。如果分隔符紧挨在一起，那么函数就会以变量值的形式返回“?”。拆分出来的字符串，以变量 `${VAR_1}`、`{VAR_2}`... 以此类推的形式加以返回。JMeter 2.1.2 及其以后版本，拖尾的分隔符会被认为缺少一个变量，会返回“?”。另外，为了更好地配合 `ForEach` 控制器，现在 `__split` 会删除第一个不用的变量（由前一次分隔符所设置）。

例如，在测试计划中定义变量 `VAR="a|c|"`：

```
${__split(${VAR},VAR),|}
```

该函数调用会返回 VAR 变量的值，例如"a|c|"，并设定 VAR_n=4（3，JMeter 2.1.1 及其以前版本）、VAR_1=a、VAR_2=?、VAR_3=c、VAR_4=?（null，JMeter 2.1.1 及其以前版本）、VAR_5=null（JMeter 2.1.2 及其以后版本）变量的值。

参数如表 11-16 所示。

表 11-16 参数描述

函数参数	描述	是否必需
待拆分字符串	一个待拆分字符串，例如 "a b c"	是
变量名	重用函数计算值的引用名	否
分隔符	分隔符，例如 " "。如果省略了此参数，函数会使用逗号做分隔符。需要注意的是，假如测试人员要多此一举，明确指定使用逗号，需要对逗号转义，如 "\",	否

17) __XPath

函数__XPath 读取 XML 文件，并在文件中寻找与指定 XPath 相匹配的地方。每调用函数一次，就会返回下一个匹配项。到达文件末尾后，会从头开始。如果没有匹配的节点，那么函数会返回空字符串，另外，还会向 JMeter 日志文件写一条警告信息。



小贴士

整个节点列表都会被保存在内存之中。

例如：

```
${__XPath(/path/to/build.xml, //target/@name)}
```

这会找到 build.xml 文件中的所有目标节点，并返回下一个 name 属性的内容。

参数如表 11-17 所示。

表 11-17 参数描述

函数参数	描述	是否必需
XML 文件名	一个待解析的 XML 文件名	是
XPath	一个 XPath 表达式，用于在 XML 文件中寻找目标节点	是

18) __setProperty

函数__setProperty 用于设置 JMeter 属性的值。函数的默认返回值是空字符串，因此该函数可以被用在任何地方，只要对函数本身调用是正确的。

通过将函数可选的第 3 个参数设置为“true”，函数就会返回属性的原始值。

属性对于 JMeter 是全局的，因此可以被用来在线程和线程组之间通信。

参数如表 11-18 所示。

表 11-18 参数描述

函数参数	描述	是否必需
属性名	待设置属性名	是
属性值	属性的值	是
True/False	是否返回属性原始值	否

19) __time

函数__time 可以通过多种格式返回当前时间。

参数如表 11-19 所示。

表 11-19 参数描述

函数参数	描述	是否必需
格式	设置时间所采用的格式	否
变量名	待设置变量名	否

如果省略了格式字符串，那么函数会以毫秒的形式返回当前时间。其他情况下，当前时间会被转成简单日期格式。包含如下形式：

- YMD = yyyyMMdd。
- HMS = HHmmss。
- YMDHMS = yyyyMMdd-HHmmss。
- USER1 = JMeter 属性 time.USER1。
- USER2 = JMeter 属性 time.USER2。

用户可以通过修改 JMeter 属性来改变默认格式，例如，time.YMD=yyMMdd。

20) _jexl

函数_jexl 可以用于执行通用 JEXL 表达式，并返回执行结果。感兴趣的读者可以从下面这两个网页链接获取更多关于 JEXL 的信息。

- <http://commons.apache.org/jexl/reference/syntax.html>。
- http://commons.apache.org/jexl/reference/examples.html#Example_Expressions。

参数如表 11-20 所示。

表 11-20 参数描述

函数参数	描述	是否必需
表达式	待执行的表达式。例如, $6*(5+2)$	是
变量名	待设置变量名	否

如下变量可以通过脚本进行访问。

- log: 函数记录器。
- ctx: JMeterContext 对象。
- vars: JMeterVariables 对象。
- props: JMeter 属性对象。
- threadName: 字符串包含当前线程名称 (在 2.3.2 版本中它被误写为“theadName”)。
- sampler: 当前的采样器对象 (如果存在)。
- sampleResult: 前面的采样结果对象 (如果存在)。
- OUT - System.out, 例如 OUT.println("message")。

JEXL 可以基于它们来创建类, 或者调用方法, 例如:

```
Systemclass=log.class.forName("java.lang.System");
now=Systemclass.currentTimeMillis();
```

需要注意的是, Web 站点上的 JEXL 文档错误地建议使用“div”做整数除法。事实上“div”和“/”都执行普通除法。



小贴士

JMeter 2.3.2 以后的版本允许在表达式中包含多个声明。JMeter 2.3.2 及其以前的版本只处理第一个声明 (如果存在多个声明, 就会记录一条警告日志)。

21) __V

函数__V 可以用于执行变量名表达式, 并返回执行结果。它可以被用于执行嵌套函数引用 (目前 JMeter 不支持)。

例如, 如果存在变量 A1、A2 和 N=1, 则:

- \${A1}: 能正常工作。
- \${A\${N}}: 无法正常工作 (嵌套变量引用)。
- \$__V(A\${N}): 可以正常工作。A\${N}变为 A1, 函数 __V 返回变量值 A1。

参数如表 11-21 所示。

表 11-21 参数描述

函数参数	描述	是否必需
变量名表达式	待执行变量名表达式	是

22) __evalVar

函数__evalVar 可以用来执行保存在变量中的表达式，并返回执行结果。

如此一来，用户可以从文件中读取一行字符串，并处理字符串中引用的变量。例如，假设变量“query”中包含有“select \${column} from \${table}”，而“column”和“table”中分别包含有“name”和“customers”，那么\${__evalVar(query)}将会执行“select name from customers”。

参数如表 11-22 所示。

表 11-22 参数描述

函数参数	描述	是否必需
变量名	待执行变量名	是

23) __eval

函数__eval 可以用来执行一个字符串表达式，并返回执行结果。

如此一来，用户就可以对字符串（存储在变量中）中的变量和函数引用做出修改。例如，给定变量 name=Smith、column=age、table=birthdays、SQL=select \${column} from \${table} where name='\${name}'，那么通过\${__eval(\${SQL})}，就能执行“select age from birthdays where name='Smith'”。这样一来，就可以与 CSV 数据集相互配合，例如，将 SQL 语句和值都定义在数据文件中。

参数如表 11-23 所示。

表 11-23 参数描述

函数参数	描述	是否必需
变量名	待执行变量	是

24) __char

函数__char 会将一串数字翻译成 Unicode 字符，另外还请参考下面__unescape()函数。

参数如表 11-24 所示。

表 11-24 参数描述

函数参数	描述	是否必需
Unicode 字符编码（十进制数或者十六进制数）	待转换的 Unicode 字符编码，可以是十进制数或者十六进制数	是

例如：

```
${__char(0xC,0xA)} = CRLF
${__char(165)} = ĩ¿ (yen)
```

25) __unescape

函数__unescape 用于反转义 Java-escaped 字符串，另外还请参考上面的__char 函数。

参数如表 11-25 所示。

表 11-25 参数描述

函数参数	描述	是否必需
待反转义字符串	待反转义字符串	是

例如：

```
${__unescape(\r\n)} = CRLF
${__unescape(1\t2)} = 1[tab]2
```

26) __unescapeHtml

函数__unescapeHtml 用于反转义一个包含 HTML 实体的字符串，将其变为包含实际 Unicode 字符的字符串。支持 HTML 4.0 实体。

例如，字符串 “<Français>” 变为 “<Français>”。

如果函数不认识某个实体，就会将实体保留下来，并一字不差地插入结果字符串中。例如，“>&zzzz;x” 会变为 “>&zzzz;x”。

参数如表 11-26 所示。

表 11-26 参数描述

函数参数	描述	是否必需
待反转义字符串	待反转义字符串	是

27) __escapeHtml

函数__escapeHtml 用于转义字符串中的字符（使用 HTML 实体）。支持 HTML 4.0 实体。

例如，“bread” & “butter”变为“"bread" & "butter"”。

参数如表 11-27 所示。

表 11-27 参数描述

函数参数	描述	是否必需
待转义字符串	待转义字符串	是

28) __FileToString

函数__FileToString 可以被用来读取整个文件。每次对该函数的调用，都会读取整个文件。如果在打开或者读取文件时发生错误，那么函数就会返回字符串 “**ERR**”。

参数如表 11-28 所示。

表 11-28 参数描述

函数参数	描述	是否必需
文件名	包含路径的文件名（路径可以是相对于 JMeter 启动目录的相对路径）	是
文件编码方式（如果不采用平台默认的编码方式）	读取文件需要用到的文件编码方式。如果没有指明就使用平台默认的编码方式	否
变量名	引用名（refName）用于重用函数创建的值	否

每次函数执行时都会解析文件名、编码方式、引用名参数。

6. 预定义变量

大多数变量都是通过函数调用和测试元件（如用户定义变量）来设置的；在这种情况下用户拥有对变量名的完整控制权。但是有些变量是 JMeter 内置的。例如，

- **Cookiename:** 包含 Cookie 值。
- **JMeterThread.last_sample_ok:** 最近的采样是否可以（true/false）。
- **START** 变量（参见后续内容）。

7. 预定义变量属性

JMeter 属性集是在 JMeter 启动时通过系统属性初始化的；其他补充 JMeter 属性来自于 jmeter.properties、user.properties 或者命令行。

JMeter 还另外定义了一些内置属性。下面是具体列表。从方便的角度考虑，属性 START 的值会被复制到同名变量中去。

- **START.MS:** 以毫秒为单位的 JMeter 启动时间。

- **START.YMD**: JMeter 启动日期格式 yyyyMMdd。
- **START.HMS**: JMeter 启动时间格式 HHmmss。
- **TESTSTART.MS**: 以毫秒为单位的测试启动时间。

请注意: **START** 变量/属性表征的是 JMeter 启动时间, 而非测试的启动时间。它们主要用于文件名之中。

11.2 详解 JMeter 正则表达式

1. 概览

JMeter 中包含范本匹配软件 Apache Jakarta ORO。在 Jakarta 网站上有一些关于它的文档, 例如 a summary of the pattern matching characters :

<http://jakarta.apache.org/oro/api/org/apache/oro/text/regex/package-summary.html>。

另外, 还有关于该软件老版本的文档 OROMatcher User's guide, 也许会有一些帮助。URL 地址: <http://www.savarese.org/oro/docs/OROMatcher/index.html>。

JMeter 的范本匹配与 Perl 语言的范本匹配类似。一个安装完整的 Perl 会包含很多关于正则表达式的文档 (搜寻 perlrequick、perlretut、perlre、perlref)。

我们有必要分清楚包含 (Contains) 和匹配 (Matches) 的差异, 它们用于响应断言测试元件:

- **包含 (Contains)** 意味着正则表达式至少部分匹配目标, 例如, 'alphabet' 包含 'ph.b.', 因为正则表达式匹配其子字符串 'phabe'。
- **匹配 (Matches)** 意味着正则表达式完全匹配目标。例如, 'alphabet' 匹配 'al.*t'。

在这一情况下, 它等同于使用 ^ 和 \$ 封装正则表达式, 即 '^al.*t\$'。但是事情并不总是这样。例如, 正则表达式 'alp|.lp.*' 包含于 'alphabet', 但并不匹配 'alphabet'。

原因在于当范本匹配器在 'alphabet' 中找到序列 'alp' 后, 就会停止尝试其他组合, 而且 'alp' 不同于 'alphabet', 它不包含 'habet'。



小贴士

不同于 Perl, 没必要将正则表达式用 // 封装。

2. 实例

1) 提取单个字符串

假设测试人员期望匹配 Web 页面的如下部分：name="file" value="readme.txt">并提取 readme.txt。

一个符合要求的正则表达式：

```
name="file" value="(.*+?)">
```

上面用到的特殊字符包括如下几个。

- (和)：封装了待返回的匹配字符串。
- .：匹配任何字符。
- +：一次或多次。
- ?：不要太贪婪，在找到第一个匹配项后停止。



小贴士

如果没有?，在找到第一个">"后，会继续寻找，直到最后一个">"，这么做很可能不是测试人员期望的。

尽管上面的表达式可以达到目的，但是使用如下表达式更有效率：name="file" value="([^\"]+)">，其中["-]意味着匹配任何东西（除了"）。在这种情况下，匹配引擎在找到第一个右侧"后，就会停止搜索。而上面例子中的匹配引擎会去寻找">"。

2) 提取多个字符串

假设测试人员期望匹配 Web 页面的如下部分：name="file" value="readme.txt">，并提取 file.name 和 readme.txt。

一个符合要求的正则表达式：

```
name="([^\"]+)" value="([^\"]+)"
```

这会创建两个组合，并可用于 JMeter 正则表达式模板，形如\$1\$ 和\$2\$。

JMeter 正则表达式提取器会将组合的值放在指定变量中，如图 11-2 所示。

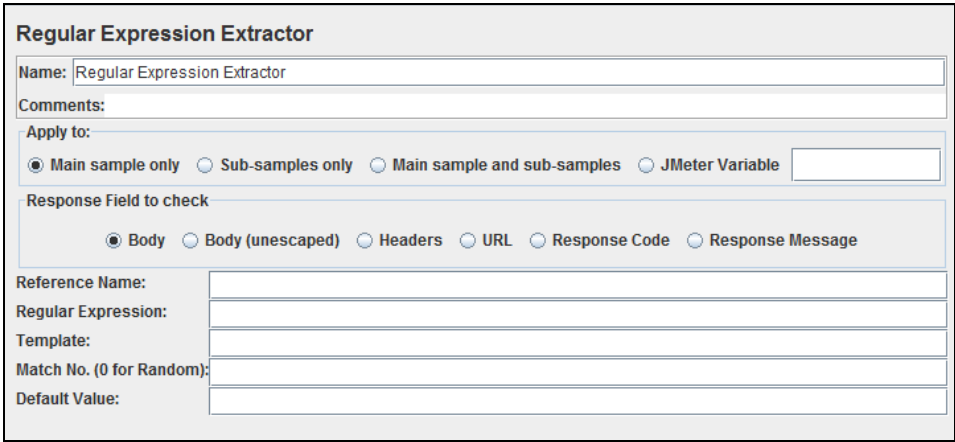


图 11-2 JMeter 正则表达式提取器

例如，

- 引用名称：MYREF。
- 正则表达式：name="(.+?)" value="(.+?)"。
- 模板：\$1\$2\$。



小贴士

不要用//封装正则表达式。

如下变量的值将会被设定。

- MYREF: file.namereadme.txt。
- MYREF_g0: name="file.name" value="readme.txt"。
- MYREF_g1: file.name。
- MYREF_g2: readme.txt。

这些变量后续可以在 JMeter 测试计划中引用，形如\${MYREF}、\${MYREF_g1}等。

3. 关键字

正则表达式使用特定字符作为关键字，这些字符对正则表达式引擎有特殊意义。在字符串中使用这些字符必须进行转义（使用反斜杠“\”），目的是将它们当成原始字符，而非正则表达式的关键字。下面是关键字和它们的含义。

- ()：组合。

- []: 字符集合。
- {}: 重复。
- +?: 重复。
- .: 任意匹配字符。
- \: 转义字符。
- | -: 选择符。
- ^\$: 字符串或行的起始和结尾。

注意，ORO 不支持 \Q 和 \E 关键字。

4. 修改器 (Modifier)

理论上修改器可以被放置在正则表达式的任何地方，并被放置的位置开始向后生效。(ORO 存在一个 BUG，修改器不能放在正则表达式的末尾。尽管修改器在这里不生效)。

单行 (?s) 和多行 (?m) 修改器通常都被放在正则表达式的开头。

忽略 (?i) 修改器可以被用来仅仅影响正则表达式的某一部分，例如：

```
Match ExAct case or (?i)ArBiTrARY(?-i) case
```

由于单行和多行修改器的设置不同，范本匹配的表现也略有不同。请注意，单行和多行操作符之间没有任何关联；它们可以被单独指定。

1) 单行模式

单行模式只影响关键字符“.”。默认情况下，“.”可以匹配任何字符（除了换行）。在单行模式下，“.”还匹配换行。

2) 多行模式

多行模式只影响关键字符“^”和“\$”。默认情况下，“^”和“\$”仅仅匹配字符串的开始和结尾。而在多行模式下“^”和“\$”匹配每一行的开始和结尾。

11.3 详解 JMeter 远程测试

如果运行 JMeter 客户端的机器性能不能满足测试需要，那么测试人员可以通过单个 JMeter GUI 客户端来控制多个远程 JMeter 服务器，以便对服务器进行压力测试，模拟足够多的并发用

户。通过远程运行 JMeter，测试人员可以跨越多台低端计算机复制测试，这样就可以模拟一个比较大的服务器压力。一个 JMeter GUI 客户端实例，理论上可以控制任意多的远程 JMeter 实例，并通过它们收集测试数据，如图 11-3 所示。这样一来，就有了如下特性：

- 保存测试采样数据到本地机器。
- 通过单台机器管理多个 JMeter 执行引擎。
- 没有必要将测试计划复制到每一台机器，JMeter GUI 客户端会将它发往每一台 JMeter 服务器。



小贴士

每一台 JMeter 远程服务器都执行相同的测试计划。JMeter 不会在执行机间做负载均衡，每一台服务器都会完整地运行测试计划。

在 1.4GHz~3GHz 的 CPU、1GB 内存的 JMeter 客户端上，可以处理线程 100~300。但是 Web Service 例外。XML 处理是 CPU 运算密集的，会迅速消耗掉所有的 CPU。一般来说，以 XML 技术为核心的应用系统，其性能将是普通 Web 应用的 10%~25%。另外，如果所有负载由一台机器产生，网卡和交换机端口都可能产生瓶颈，所以一个 JMeter 客户端线程数不应超过 100。

采用 JMeter 远程模式并不会比独立运行相同数目的非 GUI 测试更耗费资源。但是，如果使用大量的 JMeter 远程服务器，可能会导致客户端过载，或者网络连接发生拥塞。

请注意，假如测试人员将 JMeter 执行引擎安装在应用服务器（测试目标）上，那么这显然会加重应用服务器的负担，测试结果也将变得不可信。作者推荐的方式是将 JMeter 远程服务器放在应用服务器（测试目标）所在的同一个网段内。这样做既可以减少 JMeter 收集测试结果对网络产生的冲击，又可以避免对应用服务器（测试目标）性能产生影响。

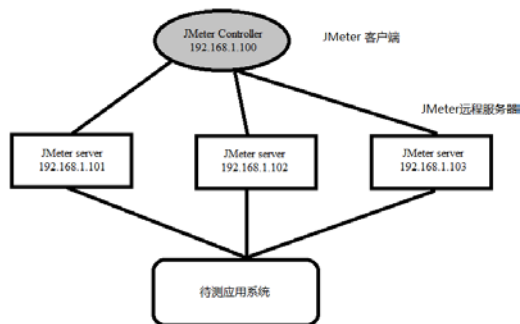


图 11-3 JMeter 远程测试原理图

下面是启动 JMeter 远程测试的基本步骤：

步骤 1：配置节点

确保所有节点（JMeter 客户端和 JMeter 远程服务器）运行相同版本的 JMeter。尽可能在所有操作系统上使用相同的 Java 版本。

如果测试用到了外部数据文件，那么请注意这些文件不会被 JMeter 客户端分发，因此测试人员需要确保每台执行机上都保存了这些数据文件（其所在目录也必须正确）。如果有必要，用户可以为每台执行机设置不同的属性变量，即在 JMeter 远程服务器上编辑 `user.properties` 或者 `system.properties` 文件。这些属性将会在 JMeter 远程服务器启动时被识别，并有可能被应用到测试计划之中，从而影响测试执行（例如，与其他远程服务器发生交互）。另外，不同的 JMeter 远程服务器可能会使用不同内容的数据文件（例如，每台服务器必须使用不同的 ID，就以此来划分数据文件）。

步骤 2：启动远程服务器

要启动 JMeter 远程节点，请在执行机上运行 `JMETER_HOME/bin/jmeter-server`（UNIX）或者 `JMETER_HOME/bin/jmeter-server.bat`（Windows）脚本。

请注意，每个远程节点上只能运行一个 JMeter 远程服务器脚本，除非采用不同的 RMI 端口。从 JMeter 2.3.1 开始，JMeter 远程服务器会自己启动 RMI 注册；用户没有必要单独启动 RMI 注册。假设测试人员一定要单独启动 RMI 注册，可以在远程节点上定义 JMeter 属性 `server.rmi.create=false`。

默认情况下，JMeter 远程服务器的 RMI 使用动态端口号。这样就会为防火墙配置带来麻烦，因此 JMeter 2.3.2 及其以后的版本，会检查 JMeter 属性 `server.rmi.localport`。如果该值非零，JMeter 远程服务器就会用它来作为本地端口号。

步骤 3：将 JMeter 远程服务器的 IP 地址添加到客户端属性文件中

编辑 JMeter 控制机的属性文件。在 `/bin/jmeter.properties` 文件中找到属性“`remote_hosts`”，使用 JMeter 远程服务器的 IP 地址作为其属性值。可以添加多个服务器的 IP 地址，以逗号作为分隔。

请注意测试人员还可以使用 `-R` 命令行选项来指明将会使用的远程服务器。这与使用 `-r` 和 `-Jremote_hosts={服务器列表}` 的效果相同。例如 `jmeter -Rhost1,127.0.0.1,host2`。

如果测试人员定义 JMeter 属性 `server.exitaftertest=true`，那么远程服务器在运行完单个测试

后就会退出。-Z 标志也有同样的效果，参见后面的内容。

步骤 4 (a)：通过 GUI 客户端启动 JMeter 测试

现在轮到启动 JMeter GUI 客户端了。在 MS-Windows 环境下运行 “bin/jmeter.bat” 脚本，在 UNIX 环境下运行 “bin/jmeter” 脚本。测试人员会发现在运行（Run）菜单下，包含两个子菜单 “Remote Start” 和 “Remote Stop”，如图 11-4 所示。这两个子菜单中包含测试人员在属性文件中设置的 JMeter 远程服务器 IP 地址。此刻，请使用远程启动和停止来代替普通的 JMeter 启动和停止。

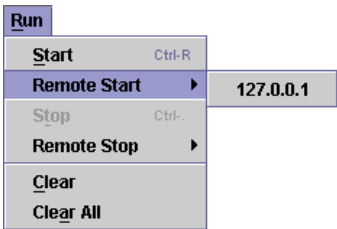


图 11-4 运行菜单

步骤 4 (b)：通过非 GUI 客户端启动 JMeter 测试

还有另外一种方法，测试人员可以通过非 GUI 客户端来启动远程服务器。命令如下：

```
jmeter -n -t script.jmx -r
```

或者：

```
jmeter -n -t script.jmx -R server1,server2...
```

其他标志可能也有用。

- -Gproperty=value: 在所有 JMeter 远程服务器中定义一个属性（可以多次出现）
- -Z: 在测试结束后退出远程服务器。

第一个例子会启动定义在 JMeter 属性 remote_hosts 中的远程服务器；而第二个例子会先定义远程服务器列表，接着启动它们。命令行客户端会在所有服务器停止后退出。

1. 手动配置 JMeter 远程测试

在某些情况下，jmeter-server 脚本不能正常工作（如果测试人员使用的操作系统不在 JMeter 开发者预期的范围内）。下面介绍如何启动 JMeter 远程服务器（对应上面的步骤 2），其中包含了更多人工操作。

步骤 2 (a): 启动 RMI 注册

从 JMeter 2.3.1 开始, JMeter 远程服务器会自己启动 RMI 注册, 因此这里的内容不适用于普通情况。如果要采用历史版本的操作方法, 首先在 JMeter 远程服务器上定义 JMeter 属性 `server.rmi.create=false`, 并遵循如下指南。

JMeter 使用 Remote Method Invocation (RMI) 作为远程通信机制。因此, 测试人员需要用到 JDK "bin" 目录中的 RMI 注册程序 (名为 "rRmiregistry")。在运行 Rmiregistry 之前, 请确保如下 jar 存在于测试人员的系统 classpath 中:

- JMeter_HOME/lib/ext/ApacheJMeter_core.jar。
- JMeter_HOME/lib/jorphan.jar。
- JMeter_HOME/lib/logkit-1.2.jar。

注册程序需要访问特定 JMeter 类。运行 Rmiregistry 无须参数。默认情况下应用程序会监听端口 1099。

步骤 2 (b): 启动 JMeter 远程服务器

一旦 RMI 注册程序运行起来, 就启动 JMeter 远程服务器。JMeter 启动脚本需携带 "-s" 选项。

步骤 3 和步骤 4 同上面的介绍。

2. 一些小技巧

JMeter/RMI 要求建立一个从客户端到远程服务器的连接。这就会用到测试人员所选择的端口号, 默认值是 1099。JMeter/RMI 还要求建立一个反向连接, 目的是从远程服务器向客户端返回测试采样结果。这就会用到一个更高数字的端口号。如果在 JMeter 客户端与 JMeter 远程服务器之间存在任何防火墙或者网络过滤器, 那么测试人员就需要确保它们已经被正确配置, 并允许相关连接通信。如果有必要, 请使用监听软件来观察通信的过程。

如果 JMeter 运行在 Suse Linux 上, 下面这些技巧对测试人员可能会有帮助。默认的安装可能会启动防火墙。在这种情况下, 远程测试将无法正常工作。如果测试人员发现连接被拒绝后, 可以通过下面的选项打开 debugging。从 JMeter 2.3.1 版本开始, RMI 注册由 JMeter 远程服务器启动; 不过相关选项依然可以通过 JMeter 命令行传递。例如, "`jmeter -s -Dsun.rmi.loader.logLevel=verbose`" (省略了 -J 前缀)。另外这些属性还可以被定义在

system.properties 文件中。

解决的方法是从 etc/hosts 中删除对 127.0.0.1 和 127.0.0.2 的回送 (Loopback)。当 127.0.0.2 的回送无效时, jmeter-server 将无法连接到 Rmiregistry。

替换:

```
`dirname $0`/jmeter -s "$@"
```

为:

```
HOST="-Djava.rmi.server.hostname=[computer_name][computer_domain]
-Djava.security.policy=`dirname $0`/[policy_file]"
`dirname $0`/jmeter $HOST -s "$@"
```

同时创建一个规则 (Qolicy) 文件, 添加[computer_name][computer_domain]行到/etc/hosts。

3. 如何使用不同端口号

默认情况下, JMeter 使用标准 RMI 端口号 1099 (这是可以改变的)。要想成功改变使用的端口号, 需满足如下条件:

- 在远程服务器, 启动 Rmiregistry 使用新端口号。
- 在远程服务器, 启动 JMeter 并预先定义 server_port 属性。
- 在客户端, 更新 remote_hosts 属性, 在其中包含 remote host:port 设置。

从 JMeter 2.1.1 版本开始, jmeter-server 脚本支持改变端口号。例如, 假设测试人员希望使用端口号 1664 (可能因为 1099 端口已经被其他应用程序占用了)。

Windows 系统 (DOS 窗口中):

```
C:\JMETER> SET SERVER_PORT=1664
C:\JMETER> JMETER-SERVER [other options]
```

UNIX 系统:

```
$ SERVER_PORT=1664 jmeter-server [other options]
[N.B. use upper case for the environment variable]
```

在这两种情况下, 脚本都会在指定端口上启动 Rmiregistry, 接着以远程服务器模式启动 JMeter, 并已经定义了 “server_port” 属性。

选定的端口号将会被记录到远程服务器的 jmeter.log 文件中 (Rmiregistry 不会创建一个日志文件)。

4. 使用采样批次

测试计划中的监听器会把它们的结果返回到 JMeter 客户端,而 JMeter 客户端默认情况下会将这些结果写入到指定文件中,采样结果会在产生后立即发回 JMeter 客户端。这样就会对网络 and JMeter 客户端产生很大的压力。用户可以通过设置一些属性,来改变默认操作。

模式 (Mode) (采样结果发送模式) 默认是 Standard。

- **Standard:** 在采样结果产生后立即发送。
- **Hold:** 将采样结果保存在一个数组中,直到测试结束。这可能会占用远程服务器的大量内存。
- **Batch:** 当计数器或者时间超过阈值之后,发送保存的采样结果。
- **Statistical:** 当计数器或者时间超过阈值之后,以概要的形式发送采样结果;采样结果以线程组 (Thread Group) 名称和采样标签 (Sample Label) 进行概要统计。积累的数据域包括: elapsed time、latency、bytes、sample count、error count,其他数据域将会被丢弃。
- **Stripped:** 将成功采样的响应数据移除。
- **StrippedBatch:** 将成功采样的响应数据移除,并批次发送。
- **Custom implementation:** 将模式参数设置为测试人员的客户化采样发送器的类名。该类必须实现接口 `SampleSender`,并且类的构造函数只有一个 `RemoteSampleListener` 型的参数。

如下属性会影响 Batch 和 Statistical 模式。

- **num_sample_threshold:** 一个批次中的采样数目 (默认为 100)。
- **time_threshold:** 等待的毫秒数 (默认为 60 秒)。

11.4 详解 JMeter 最佳实践经验

1. 限制线程数目

机器硬件性能将会限制用户可以运行的 JMeter 有效线程数目。另外这还依赖于待测服务器的性能 (快速服务器会加重 JMeter 的负担,因为它响应请求的速度更快)。运行的 JMeter 线程越多,统计得到的时间信息就越不准确。JMeter 负担越重,每一个线程等待 CPU 的时

间就越长，得到的时间统计信息也就越发膨胀。如果测试人员需要完成大并发量的负载测试，那么请考虑在多台机器上运行多个非 GUI JMeter 实例。

2. 灵活使用用户变量

有一些测试计划会针对不同的虚拟用户/线程，使用不同的变量值。例如，测试人员所希望测试的业务流程，可能会要求每个虚拟用户独立地完成系统登录操作。这可以通过 JMeter 的特性轻松实现。

例如：

- 创建一个文本文件，其中包含用户名和密码，通过逗号进行分隔。将它放到测试计划所在的同一个目录之中。
- 为测试计划添加一个 CSV DataSet 配置元件，如图 11-5 所示。命名变量 USER 和 PASS。
- 在合适的采样器中，用\${USER}代替登录名，用\${PASS}代替密码。

CSV DataSet 配置元件会为每一个虚拟用户从文件中读取一行。

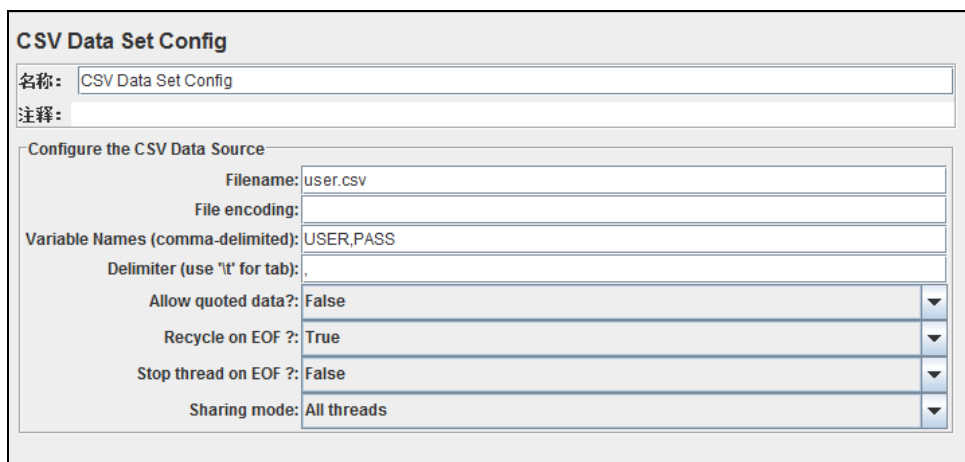


图 11-5 CSV DataSet 配置元件

3. 减少 JMeter 对资源的要求

减少资源占用的一些建议。

- 使用非 GUI 模式：jmeter -n -t test.jmx -l test.jtl。
- 尽可能地少使用监听器；如果像上面一样使用了-l 标志，那么就可以删除所有监听器或

者使它们失效。

- 避免使用一大堆相同的采样器，而应该在一个循环中使用相同的采样器，并使用变量（CSV Data Set）来修改这些采样器。或者可能会用到 Access Log Sampler（在这里不能使用 Include Controller，因为它会将指定文件中的所有测试元件直接添加到测试计划中来）。
- 不要使用函数测试模式（Functional Mode）。
- 以 CSV 格式输出测试结果，尽量不要使用 XML 格式。
- 仅保存测试人员需要的数据。
- 尽量少使用断言。

如果测试人员的测试需要大量数据（特别是数据需要随机化），请提前准备好测试数据，放到数据文件中，以 CSV Dataset 方式读取。这样就能避免在测试运行阶段浪费资源。

4. 参数化测试

通常使用不同的设置反复运行相同的测试会很有用。例如，改变线程或者循环的数目，或者改变主机名。

办法之一就是在测试计划中定义一系列变量，并在测试元件中使用这些变量。例如，用户可以定义变量 `LOOPS=10`，并在线程组中引用 `${LOOPS}`。如果要循环运行 20 次，只需要在测试计划中改变变量 `LOOPS` 的值。

如果要以非 GUI 模式运行大量测试，使用变量参数化就比较麻烦。一种解决办法就是用属性来代替变量，例如 `LOOPS=${__P(loops,10)}`。这样就会使用属性“loops”的值，如果属性 loops 不存在，就使用默认值 10。属性“loops”接下来可以在 JMeter 命令行中定义：`jmeter ... -Jloops=12 ...`。如果有很多属性需要一起改变，那么解决办法就是使用一组属性文件。用户可以使用命令行 `-q` 选项，以便将合适的属性文件传递给 JMeter。

5. BeanShell 服务器

BeanShell 解释器有一个非常有用的特性，它表现得一台服务器，可以通过 Telnet 或者 HTTP 访问。



小贴士

这里没有安全机制。任何人只要能连接上对应端口，就能执行任何 BeanShell 命令。这些

命令可以提供对 JMeter 应用程序和主机不受限制的访问。不要启动 BeanShell 服务器，除

非已经对端口访问做了保护，例如，通过防火墙。

如果测试人员确实希望使用 BeanShell 服务器，请在 `jmeter.properties` 文件中定义如下属性：

```
beanshell.server.port=9000
beanshell.server.file=../extras/startup.bsh
```

在上面的例子中，BeanShell 服务器将会被启动，并监听端口 9000 和 9001。端口 9000 将会用于 HTTP 访问。端口 9001 将会用于 Telnet 访问。`startup.bsh` 文件将被 BeanShell 服务器处理，它可以用于定义各种函数及初始化变量。文件 `startup` 中定义了设置/打印 JMeter 及系统属性的各种方法。测试人员可以在 JMeter 控制台中看到如下内容：

```
Startup script running
Startup script completed
Httpd started on port: 9000
Sessiond started on port: 9001
```

这里有一个实际例子，假设测试人员有一个以非 GUI 模式长期运行的 JMeter 测试，并且测试人员希望能在测试运行期间不定时改变吞吐率。测试计划中包含一个恒定的吞吐率定时器，它是以属性的形式定义的，形如 `$_P(throughput)`。如下 BeanShell 命令可以被用于改变测试：

```
printprop("throughput");
curr=Integer.decode(args[0]); // Start value
inc=Integer.decode(args[1]); // Increment
end=Integer.decode(args[2]); // Final value
secs=Integer.decode(args[3]); // Wait between changes
while(curr <= end){
    setprop("throughput",curr.toString()); // Needs to be a string here
    Thread.sleep(secs*1000);
    curr += inc;
}
printprop("throughput");
```

该脚本可以被存储到一个文件中（如 `throughput.bsh`），接着使用 `bshclient.jar` 将其传递给 BeanShell 服务器。例如：

```
java -jar ../lib/bshclient.jar localhost 9000 throughput.bsh 70 5 100 60
```

6. BeanShell 脚本编程

1) 回顾

每一个 BeanShell 测试元件都有独立的解释器备份（针对每个线程）。如果测试元件被重复

调用，例如，被放在循环之中，那么在多次调用间解释器将被保留，除非选中了“Reset bsh.Interpreter before each call”复选框，如图 11-6 所示。

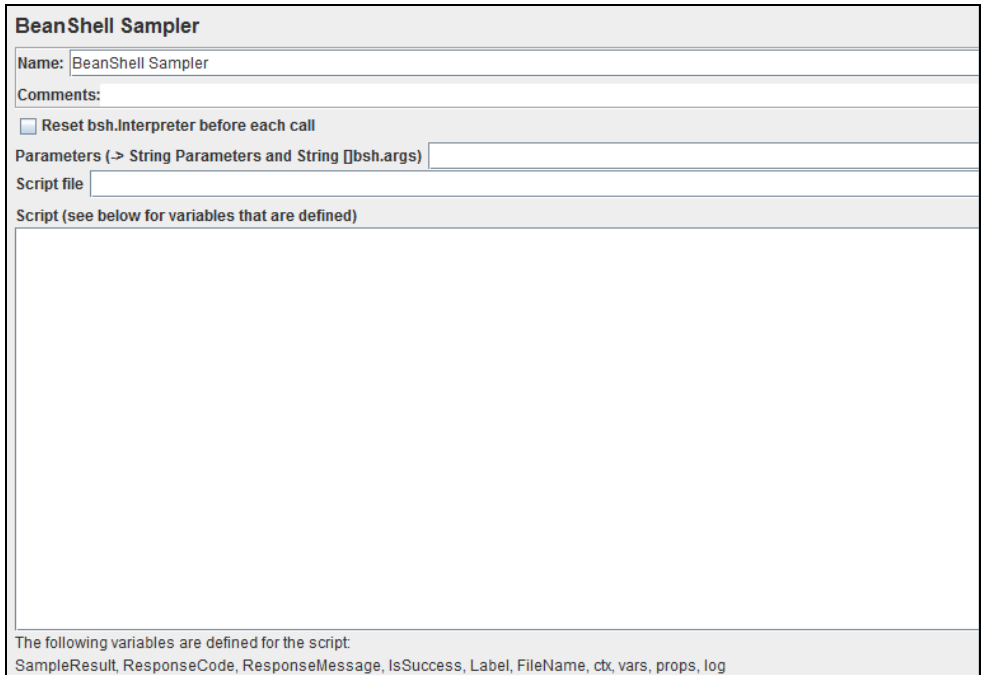


图 11-6 BeanShell 采样器

一些长期运行的测试可能导致解释器使用很多内存；如果遇到这种情况，请尝试使用 Reset 选项。

测试人员可以使用命令行解释器，以便在 JMeter 之外测试 BeanShell 脚本：

```
$ java -cp bsh-xxx.jar[:other jars as needed] bsh.Interpreter file.bsh [parameters]
```

或者：

```
$ java -cp bsh-xxx.jar bsh.Interpreter
bsh% source("file.bsh");
bsh% exit(); // or use EOF key (e.g. ^Z or ^D)
```

2) 共享变量

变量可以被定义在 startup（初始化）脚本中。它们将会被一直保留下来，跨越测试元件的多次调用，除非使用到了 reset 选项。

脚本同样可以访问 JMeter 变量，只需使用“vars”变量的 get()和 put()方法，例如，

`vars.get("HOST"); vars.put("MSG","Successful");`。`get()`和`put()`方法只支持字符串类型的变量，不过这里还有`getObject()`和`putObject()`方法可以被用于任何对象。**JMeter** 变量对线程而言是局部的，但是可以被所有测试元件使用（不仅是 **BeanShell**）。

如果测试人员需要在线程间共享变量，那么可以使用 **JMeter** 属性：

```
import org.apache.jmeter.util.JmeterUtils;
String value=JmeterUtils.getPropDefault("name","");
JmeterUtils.setProperty("name","value");
```

JMeter bin 目录中的**.bshrc** 样例文件中包含有 `getprop()`和`setprop()`定义的范例。

另一种共享变量的可行方法是使用“**bsh.shared**”的共享 namespace。例如：

```
if (bsh.shared.myObj == void){
// not yet defined, so create it:
myObj=new AnyObject();
}
bsh.shared.myObj.process();
```

与其在测试元件中创建共享对象，不如在 **startup** 文件（由 **JMeter** 属性“**beanshell.init.file**”定义）中创建。该文件只会被处理一次。

7. 使用 BeanShell、JavaScript 或者 Jexl 开发脚本函数

在 **JMeter** 中很难以函数形式编写和测试脚本。不过 **JMeter** 提供了 **BSF** 和 **BeanShell** 采样器作为代替。

创建一个简单的测试计划，其中包含 **BSF** 采样器（**BSF Sampler**）和查看结果树（**Tree View Listener**）。在采样器的脚本面板中编码，接着运行测试计划以便对脚本进行测试。如果发生任何错误，都会在查看结果树中展现。另外运行脚本的结果会以响应形式展现。

一旦脚本能够正常工作，就可以将它以变量形式保存在测试计划中。接下来可以使用脚本变量来实现函数调用。例如，假设有一个 **BeanShell** 脚本存储在变量 **RANDOM_NAME** 中，函数调用形如`${__BeanShell(${RANDOM_NAME})}`。没有必要在脚本中转义逗号，因为函数调用解析发生在替换函数值之前。

8. 使用代理服务器

用户使用代理服务器，最重要的一件事情就是过滤测试人员不想要的信息。例如，有时候无须记录对图像的请求（**JMeter** 可以被设置为下载页面上的所有图片）。这些多余的采样可能会扰乱测试人员的测试计划。在大多数情况下，页面文件都有一个通用的扩展名，

如.jsp、.asp、.php、.html等。对于这些页面文件，测试人员可以在字段“Include Pattern”中输入“.*\jsp”（扩展名），以便将它们加入关注范围。

同样地，测试人员可以在“Exclude Pattern”中输入“.*\gif”将其排除在外。测试人员可以使用相同的方法来排除 StyleSheets、JavaScript等页面文件。请测试这些设置，以便验证 JMeter 录制的操作就是测试人员所期望的。

代理服务器会期望能够找到一个线程组，在该线程组下应该有一个录制控制器，录制的 HTTP 请求就放在录制控制器的下面。这样就可以将所有采样器封装到某个控制器之下，该控制器可以被重命名用于描述测试案例。

现在，请浏览一下测试案例的步骤。如果测试人员没有预先定义测试案例，那么请使用 JMeter 去记录测试人员的操作，以便定义测试案例。一旦测试人员完成了一系列测试步骤的录制工作，请将整个测试案例存储到某个文件中。然后刷新，开始录制新的测试案例。通过这种方式，测试人员可以快速录制大量测试案例“粗稿”。

代理服务器的另外一个重要特性，就是测试人员可以从录制得到的采样中抽象出一些特定的通用元素。通过在测试计划（作为测试元件理解）中定义一些用户自定义变量或者使用配置元件（用户定义的变量（User Defined Variables）），测试人员就可以让 JMeter 在录制得到的采样中自动替换这些值。假设，测试人员测试某个应用服务器 xxx.yyy.com，测试人员可以定义一个变量 server，且值为 xxx.yyy.com，然后在测试人员录制得到的采样中在任何地方发现该值，都会被 JMeter 替换为“\${server}”。



小贴士

请注意这里的匹配是大小写敏感的。

如果 JMeter 不能录制任何操作，请确保浏览器使用了正确的代理设置。某些浏览器会忽略代理 localhost 或者 127.0.0.1；遇到这类问题，请使用本地主机名或者 IP 地址来代替。

如果发生错误“unknown_ca”，可能是由于测试人员尝试录制 HTTPS，而浏览器又不接受 JMeter 代理服务器证书所导致的。

11.5 一些小技巧

1. 在线程间传递变量

JMeter 变量作用域局限于所属线程。这样设计是经过深思熟虑的，目的是让测试线程能够独立运转。有时候用户可能需要在不同线程间（可能属于同一个线程组，也可能不属于同一个

线程组）传递变量。

其中一种方法就是使用属性。属性为所有 JMeter 线程所共享，因此当某个线程设置一个属性后，其他线程就可以读取更新后的值。

如果存在大量数据需要在线程间传递，那么可以考虑使用文件。例如，测试人员可以在一个线程中使用监听器，保存响应到文件（Save Responses to a file）或者 BeanShell PostProcessor。而在另外一个线程中使用 HTTP 采样器的“file:”协议来读取文件，接着使用一个后置处理器或者 BeanShell 测试元件提取信息。

如果在测试启动前测试人员就能获得测试数据，那么最好将数据保存到文件中，使用 CSV Dataset 读取。

2. 启动 Debug 日志记录

大多数测试元件都支持 Debug 日志记录。如果通过 GUI 运行测试计划，那么在选中测试元件后，可以通过“帮助”菜单 enable 或者 disable。在“帮助”菜单 中有一个选项“What’s this node?”，通过它可以查看 GUI 和测试元件的类名，如图 11-7 所示。通过它们，测试人员可以决定修改哪一项 JMeter 属性，以便修改日志级别。

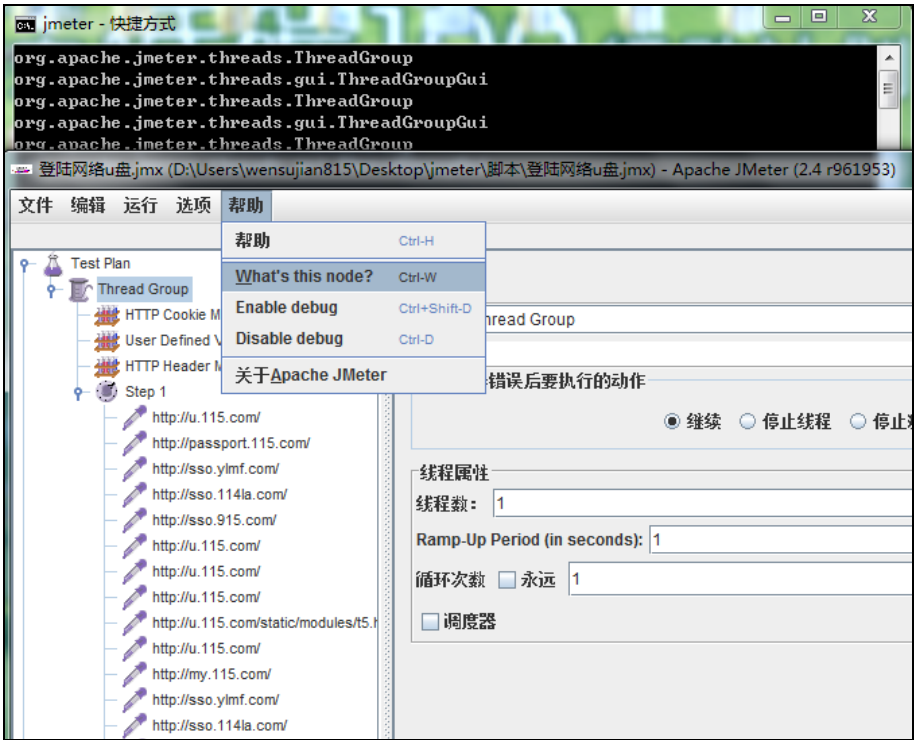


图 11-7 BeanShell 采样器

下面是文件 `jmeter.properties` 中关于日志级别的属性：

```
#Logging levels for the logging categories in JMeter. Correct values are
FATAL_ERROR, ERROR, WARN, INFO, and DEBUG

# To set the log level for a package or individual class, use:
# log_level.[package_name].[classname]=[PRIORITY_LEVEL]
# But omit "org.apache" from the package name. The classname is optional.
Further examples below.

Log_level.jmeter=INFO
log_level.jmeter.junit=DEBUG
#log_level.jmeter.control=DEBUG
#log_level.jmeter.testbeans=DEBUG
#log_level.jmeter.engine=DEBUG
#log_level.jmeter.threads=DEBUG
#log_level.jmeter.gui=WARN
#log_level.jmeter.testelement=DEBUG
#log_level.jmeter.util=WARN
#log_level.jmeter.util.classfinder=WARN
#log_level.jmeter.test=DEBUG
#log_level.jmeter.protocol.http=DEBUG
# For CookieManager, AuthManager etc:
#log_level.jmeter.protocol.http.control=DEBUG
#log_level.jmeter.protocol.ftp=WARN
#log_level.jmeter.protocol.jdbc=DEBUG
#log_level.jmeter.protocol.java=WARN
#log_level.jmeter.testelements.property=DEBUG
log_level.jorphan=INFO
```

11.6 本章小结

本章内容比较宽泛，总结起来都是一些 JMeter 高阶知识。其中“详解 JMeter 函数和变量”一节，请重点关注各种 JMeter 内置函数；“详解 JMeter 正则表达式”一节，对入门读者很重要；“详解 JMeter 远程测试”一节在实际工作中会经常用到，非常重要；“解密 JMeter 最佳实践经验”和“一些小技巧”都是一些实践经验，供读者借鉴。

零成本实现Web性能测试

——基于Apache JMeter

目前LoadRunner是使用最广泛的Web性能测试工具，但其昂贵的价格将大多数中小软件企业挡在了门外，使用盗版软件不仅不道德，还会面临法律风险。以JMeter为代表的开源性能测试工具，不仅完全免费，而且足以满足我们绝大多数性能测试需求。免费而又好用的东西，自然值得我们推崇，故本书旨在介绍如何使用开源性能测试工具JMeter来构建你的Web性能测试体系，为中小软件企业节省成本。

本书特点：

- 着重介绍如何使用开源性能测试工具JMeter，来构建Web性能测试体系；
- 以某大型保险集团公司的实际性能测试为范例，提供了完整的Web性能测试解决方案；
- 从实战出发，向读者演示包含性能测试需求分析、性能测试案例设计、性能测试环境搭建、性能测试执行、性能测试结果分析的完整性能测试流程；
- 提供使用JMeter经常要用到的一些资料，帮助读者轻松掌握Web性能测试的方法。



策划编辑：张月萍
责任编辑：贾 莉
封面设计：李 玲

本书贴有激光防伪标志，凡没有防伪标志者，属盗版图书。

上架建议：软件测试

ISBN 978-7-121-15526-0



9 787121 155260 >

定价：59.00元