Broadview
www.broadview.com.cn

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

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

博文视点·IT出版旗舰品牌
技术凝聚实力·专业创新出版
Broadview
www.broadview.com.cn

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。

上架建议：软件测试

ISBN 978-7-121-15526-0

9 787121 155260 >

定价：59.00元

策划编辑：张月萍
责任编辑：贾　莉
封面设计：李　玲

零成本实现Web性能测试

——基于Apache JMeter

澄鉴剑 编著

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

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。

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。

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翠、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 初识性能测试

1.1.1 性能测试的概念

关于“性能测试是什么”这个问题，真可谓众说纷纭，我们不妨先博览众家之言，以期获得一个全面的认识。

性能测试是通过自动化的测试工具模拟多种正常峰值及异常负载条件来对系统的各项性能指标进行测试。负载测试和压力测试都属于性能测试，两者可以结合进行。通过负载测试，确定在各种工作负载下系统的性能，目标是当负载逐渐增加时，测试系统各项性能指标的变化情况。压力测试是通过确定一个系统的瓶颈或者不能接受的性能点，来获得系统能提供的最大服务级别的测试。如图 1-1 所示为一个典型的性能测试曲线。

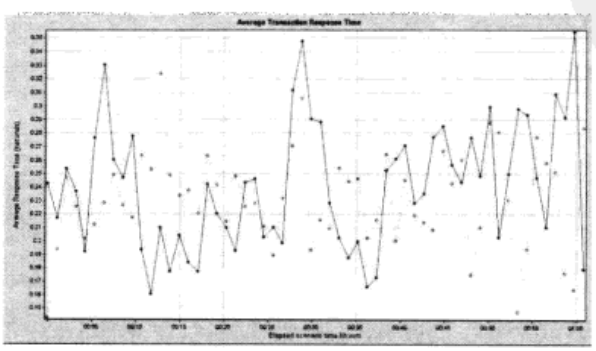


图 1-1 典型的性能测试曲线

以上是百度词条对性能测试的定义，下面再来看看维基百科对于性能测试的定义：

在计算机领域，软件性能测试被用来判断计算机、网络、软件程序或者驱动程序的速度和效率。这一过程会在同一实验环境下进行大量测试，以便于衡量系统功能的响应时长或者 MIPS（每秒执行指令数目）等指标。其他系统特性，如可靠性、可量测性、互用性等，也可以用性能测试来衡量。性能测试通常与压力测试一起进行。

读者朋友应该可以发现以上两种定义的细微差别，百度词条认为性能测试由负载测试和压力测试构成，而维基百科定义的性能测试几乎等同于负载测试。作者更认同百度词条描述的性能测试定义，如果读者朋友有不同的看法，也没有什么关系，毕竟本书不是一本深奥的测试理论书籍，而是一本测试实战类书籍。

1.1.2 性能测试的目的

性能测试的目的是验证软件系统是否能够达到用户提出的性能指标，同时发现软件系统中存在的性能瓶颈，以优化软件，最后起到优化系统的目的。性能测试包括以下几个方面：

- 评估系统的能力：测试中得到的负荷和响应时长数据可以被用于验证所计划的模型的能力，并帮助做出决策。
- 识别体系中的弱点：受控的负荷可以被增加到一个极端的水平并突破它，从而修复体系的瓶颈或薄弱的地方。
- 系统调优：重复运行测试，验证调整系统的活动是否得到了预期的结果，从而改进性能。
- 检测软件中的问题：长时间的测试执行可导致程序发生由于内存泄露引起的失败，揭示程序中隐含的问题或冲突。
- 验证稳定性（Resilience）、可靠性（Reliability）：在一个生产负荷下执行测试一定的时间是评估系统稳定性和可靠性是否满足要求的唯一方法。

1.1.3 性能测试的常见分类

性能测试包括负载测试、强度测试和容量测试等。

- **负载测试（Load Testing）**：负载测试是指通过测试系统在资源超负荷情况下的表现，来发现设计上的错误或验证系统的负载能力。在这种测试中，将使测试对象承担不同的工作量，以评测和评估测试对象在不同工作量条件下的性能行为，以及持续正常运行的能力。负载测试的目标是确定并确保系统在超出最大预期工作量的情况下

仍能正常运行。此外，负载测试还要评估性能特征，如响应时长、事务处理速率和其他与时间相关的性能指标。

- **压力测试（Stress Testing）**：在软件工程中，压力测试是对系统不断施加压力的测试，是通过确定一个系统的瓶颈或者不能接收的性能点，来获得系统能提供的最大服务级别的测试。例如测试一个 Web 站点在大量的负荷下，何时系统的响应会退化或失败。
- **容量测试（Volume Testing）**：容量测试确定系统可处理同时在线的最大用户数。

1.1.4　性能测试的常见指标

在实际工作中我们经常会对两种架构的软件进行测试：B/S 和 C/S，它们关注的指标有哪些区别？

对于 B/S 架构的软件，一般会关注如下 Web 服务器性能指标。

- Avg Rps：平均每秒钟的响应次数＝总请求次数/秒数。
- Avg time to last byte per terstion （mstes）：平均每秒业务脚本的迭代次数。
- Successful Rounds：成功的请求。
- Failed Rounds：失败的请求。
- Successful Hits：成功的点击次数。
- Failed Hits：失败的点击次数。
- Hits Per Second：每秒点击次数。
- Successful Hits Per Second：每秒成功的点击次数。
- Failed Hits Per Second：每秒失败的点击次数。
- Attempted Connections：尝试连接数。
- Throughput：吞吐率。

对于 C/S 架构的程序，由于软件后台通常为数据库，所以我们更注重数据库的测试指标。

- User Connections：用户连接数，也就是数据库的连接数量。
- Number of Deadlocks：数据库死锁。
- Butter Cache Hit：数据库 Cache 的命中情况。

注意，在实际性能测试过程中，需要观察的性能指标并不限于以上提到的这些，需要根据实际情况做出选择和权衡，有些指标如 CPU 占用率、内存占用率、数据库连接池等，也有非常重要的参考意义。

1.1.5 性能测试的基本流程

1. 明确性能测试需求

明确性能测试需求是性能测试的第一步，有了好的开始，事情就成功了一半。性能测试需求应该明确测试涉及的功能点。表 1-1 所示是一个性能测试需求的简单例子。

表 1-1 性能测试示例

测试需求名称	描述
性能测试-简单查询	【参与者】：无 【概述】：测试简单查询性能 【前置条件】：无 【后置条件】：无 【业务数据】：无 【不可测试原因】：无 【流程规则】：无 【业务规则】： 涉及 UC： UC-XX- XX -B-021 快速查询功能 UC- XX - XX -B-024 下一任务功能 UC- XX - XX -B-038 系统操作历史显示 UC- XX - XX -B-039 客户接触历史显示 UC- XX - XX -B-043 内呼比对查询 UC- XX - XX -B-049 投保单查询 UC- XX -B-009 活动列表(有效) UC- XX -B-010 活动列表(其他状态) 性能指标参见测试方案 【页面规则】：无 【特殊规则】：无 【接口规则】：无 【检查内容】： 简单查询性能是否满足要求

2. 制定性能测试方案

性能测试方案应该详尽地描述如何进行性能测试，其中应该至少包括：测试背景、测试目的、测试范围、测试进入条件、测试退出条件、测试指标要求、测试策略、测试时机、测试风险和测试资源。

其中测试范围、测试进入条件、测试退出条件、测试策略、测试风险、测试资源尤为重要。下面是一个性能测试方案的简单例子：

1）测试进入条件

（1）不遗留 L1 的缺陷。

（2）性能测试数据准备完毕。

（3）系统功能测试已结束。

2）测试退出条件

（1）各场景执行时间达到测试场景要求。

（2）系统出现大量错误，暂停执行性能测试。

3）测试指标要求

关键功能的响应时长和用户数如表 1-2 所示。

表 1-2　关键功能的响应时长和用户数

关键功能	平均使用次数	平均用户数	高峰段用户数	平均响应时长	可接受最长响应时长	使用时间段
×××功能		20 000	23 000	<4 s	≤7 s	8:00~21:00
×××功能		20 000	23 000	<3 s	≤6 s	8:00~21:00
×××功能		20 000	23 000	<3 s	≤6 s	8:00~21:00
×××功能		20 000	23 000	<4 s	≤7 s	8:00~21:00
×××功能		20 000	23 000	<2 s	≤4 s	8:00~21:00
×××功能		20 000	23 000	<6 s	≤15 s	8:00~21:00
×××功能		20 000	23 000	<3 s	≤6 s	8:00~21:00

4）测试策略

测试环境指标折算：

测试环境平均并发数 =(最大在线人数×10%)/n

式中，n 是生产环境和测试环境服务器配置折算比，例如 n = 公倍数((生产 Web 服务器数/

测试 Web 服务器数),(生产 APP 服务器数/测试 APP 服务器数))×(生产服务器内存/测试服务器内存)，一般算下来 $n=4$。

存量数据：

按照日均×××任务量来推算存量数据。为了确保系统高可用率及迅速响应，系统会对数据做定期归档，如表 1-3 所示。

表 1-3 存量数据范例

存量数据类型	归档时间
××信息	××××秒
××信息	××××秒
××结果	××××秒
××历史	××××秒
××信息	××××秒

组合场景测试：

（1）用下一任务功能获取优先级最高的任务。

（2）编辑客户信息。

（3）保费计算。

（4）投保单录入。

（5）记录电销结果。

这 5 项内容应该是坐席操作最多的功能，比例①:②:③:④:⑤=3:2:2:1:3。

测试通过标准：

（1）平均响应时长满足测试指标要求。

（2）90%响应时长满足测试指标要求。

（3）2 小时压力测试中脚本没有报错。

测试场景总的指导原则：

场景设置：每 1 秒增加 5 个用户，以 x 个并发为基础，逐渐递增至目标并发数，直到服务器吞吐率不再增加，响应时长随用户增加比例稳定同比增加。达到并发量之后再持续运行脚本 x 小时左右。场景中的并发用户匀速启动和停止。

后台监控：

后台监控工具的选择如表 1-4 所示。

表 1-4　后台监控范例

是否需要	监控对象	监控指标	使用工具
是	Web、APP 主机	CPU、MEMORY、I/O	openview
是	数据库服务器	CPU、MEMORY、I/O	openview
是	数据库	TOP　SQL 等	APM
是	中间件	等待队列	CONSOLE

3．编写性能测试案例

测试人员应该以性能测试需求和性能测试方案作为输入，以便产出性能测试案例。如表 1-5 所示为一个性能测试案例的简单例子。

表 1-5 性能测试案例范例

测试案例名称	测试步骤	描述	预期结果
性能测试-登录（身份验证）	步骤 1	计算性能测试并发用户数，方法参见《×××项目_性能测试方案》	确定性能测试并发用户数
	步骤 2	准备性能测试脚本	性能测试脚本准备完成
	步骤 3	为性能测试准备存量数据，其中： ××信息 ××××条数据 ××信息 ××××条数据××结果 ××××条数据 ××历史 ×××× 条数据 ××信息 ×××× 条数据	准备存量数据完成
	步骤 4	执行脚本，验证系统是否满足相关性能测试指标 平均响应时长 <x 秒 90%响应时长≤x 秒	系统满足相关性能测试指标
	步骤 5	执行 x 小时压力测试	①系统满足相关性能测试指标 ②x 小时压力测试中脚本没有报错

4．执行性能测试案例

性能测试案例只是一个指导性的测试文件，具体如何完成性能测试工作，要依赖于测试人员的技能和具体的性能测试工具。在本书后面的章节，会围绕这一问题做深入探讨。

5. 分析性能测试结果

毫无疑问，分析性能测试结果非常重要，而且很有难度。可以这么说，会执行性能测试案例的人是“徒弟”，能够准确全面分析性能测试结果的人是“师父”。如何分析性能测试结果，也是本书后面章节需要深入探讨的一个问题。

6. 生成性能测试报告

如何生成一份准确严谨的性能测试报告，是一项技巧性的工作。读者朋友只要遵循一定原则，并掌握一些文字技巧就不难办到。一份性能测试报告，至少应该包含如下内容。

- 测试基本信息：包含测试目的、报告目标读者、术语定义、参考资料。
- 测试环境描述：包含服务器软/硬件环境、网络环境、测试工具、测试人员。
- 性能测试案例执行分析：需要详细描述每个测试案例的执行情况，以及对应的测试结果分析。
- 测试结果综合分析及建议：对本次性能测试做综合分析，并给出测试结论和改进建议。
- 测试经验总结。

1.2 开源 Web 性能测试

1.2.1 Web 性能测试的重要性

当前绝大多数企业应用系统都是基于 Web 的应用系统，人们可以通过 Internet 浏览器便捷地访问它们。在可以预见的将来，“云计算”会进一步推动这种趋势。最近几年电子商务的发展日新月异，B2C、C2C 规模不断扩大，对 Web 应用系统的性能要求也越来越高。如果 Web 应用系统响应缓慢，甚至发生系统崩溃，由此带来的损失，绝对让企业无法容忍。作为软件质量保证的最后一关，测试人员责无旁贷，必须采取切实有效的方法来验证 Web 应用系统的性能是否达到组织的要求。

1.2.2 开源 Web 性能测试介绍

在前一节中，已经介绍了 Web 性能测试的重要性。那么，如此重要的事情，当然需要高度重视，在预算充足的情况下，选择商业测试工具并无不妥。假如测试部门预算紧张又该如何做？

坐以待毙当然不行，幸好我们还有另一个选择，那就是使用开源性能测试工具。

是否应该选择开源性能测试工具？要想回答这个问题，首先要明确所在组织的测试策略，其次还要考虑组织的现实情况。回答以下几个问题，将有助于做出正确的判断：

- 测试部门是否拥有非常充足的预算来购置商业性能测试工具？
- 测试人员是否拥有基本的编程经验？
- 测试人员是否拥有良好的英文阅读能力？
- 测试部门领导是否信任免费的开源测试工具？

如果所在的测试部门预算较为紧张，测试人员拥有基本的编程经验，而且他们具备良好的英文阅读能力，测试部门领导又对开源性能测试工具不抱有排斥心理，那么恭喜你，你所在的测试部门非常适合使用开源性能测试工具。

1.2.3　开源性能测试的优势

1. 更少的 IT 投入

目前主流商业性能测试工具价格不菲，例如 HP 公司的 LoadRunner（LoadRunner8.0：Controller 55 000 美元，Monitor 75 000 美元），支持的最大的虚拟用户数目：100（123000 美元）、250（165 000 美元）、500（206 000 美元）、1 000（309 000 美元）。使用开源性能测试工具可以大幅节约 IT 投入，因为开源工具的获取成本为零。

2. 更灵活的定制工具

商业性能测试工具通常拥有良好的文档支持，却普遍失之于灵活。测试人员很难定制商业性能测试工具，无法为它添加个性化的特性。与此相反，开源性能测试工具能够很好地支持个性化定制，例如为工具添加个性化的测试方法。

3. 更快、更便捷的服务支持

商业性能测试工具拥有专业的支持团队，但这种服务普遍昂贵而又响应不及时。采用开源性能测试工具，测试人员可以方便地从各种专业讨论组和技术论坛中获取支持，甚至可以和工具开发团队进行远程交流。

4. 更灵活的测试框架，避免绑定 IT 服务供应商

当发现商业性能测试工具不能满足组织需要时，昂贵的价格将阻止测试人员及时做出调整。一旦更换不合适的工具，前期的 IT 投资就将毫无意义。采用开源性能测试工具，测试部门能够及时地调整测试框架，而不用担心投资损失。

1.3 本章小结

本章首先从性能测试的基本概念入手，简要介绍了性能测试的定义、目的、分类、关注指标和基本流程，接着介绍了 Web 性能测试的重要性，以及开源 Web 性能测试的优势。如果决定采用开源性能测试工具来构建组织的性能测试体系，那么请系好“安全带”，我们即将开始体验开源性能测试的奇妙旅程。

第 2 章

JMeter基础知识

2.1 JMeter 简介

Apache JMeter 是 100%的 Java 桌面应用程序，用于对软件做压力测试。它最初被设计用于 Web 应用测试，但后来扩展到其他测试领域。另外，JMeter 能够对应用程序做功能/回归测试，通过创建带有断言的脚本来验证被测程序返回了期望的结果。为了保证最大限度的灵活性，JMeter 允许使用正则表达式创建断言。

Apache JMeter 可以用于对静态和动态资源（文件、Servlet、Perl 脚本、Java 对象、数据库和查询、FTP 服务器等）的性能进行测试。它可以用于对服务器、网络或对象模拟繁重的负载来测试它们的强度或分析不同压力类型下的整体性能。测试人员可以使用它做性能图形分析或者测试服务器/脚本/对象在大并发负载下的表现。

1．JMeter 的历史

Apache Software Foundation 的 Stefano Mazzocchi 是 JMeter 的最初开发人员。起初是为了测试 Apache JServ 的性能（一个后来被 Apache Tomcat 项目替代的项目）。后续的开发者重新设计了 JMeter，增强了 GUI 并添加了对功能测试的支持。

2．JMeter 远景

随着开发人员利用插件架构的优势，JMeter 的能力迅速扩展。将来开发的主要目标是使得 JMeter 尽可能地变成一个有用的衰退测试工具，而无损 JMeter 的压力测试能力。

2.1.1 JMeter 主要特点

JMeter 的主要特点包括如下。

（1）支持对多种服务类型进行测试，包括：

- Web - HTTP, HTTPS
- SOAP
- Database via JDBC
- LDAP
- JMS
- Mail - POP3(S) and IMAP(S)

（2）支持通过录制/回放方式获取测试脚本。

（3）具备高可移植性，是 100%纯 Java 程序。

（4）采用多线程框架，允许通过多个线程并发取样及通过独立的线程组对不同的功能同时取样。

（5）精心设计的 GUI 支持高速用户操作和精确计时。

（6）支持缓存和离线的方式分析/回放测试结果。

（7）具备高扩展性，包括：

- 插拔式的取样器支持无限制扩充测试能力。
- 提供各种负载统计表和可插拔的计时器。
- 数据分析和可视化插件提供了很好的扩展性及个性化。
- 支持通过预置函数为测试提供动态输入数据，以及通过预置函数对测试数据进行操作。
- 支持用脚本构造采样器（1.9.2 及以上版本支持 BeanShell）。

尽管从 Web 服务或者其他远程服务的角度，JMeter 看上去很像是一款浏览器，但实际上它并不是浏览器，因为它仅支持浏览器的部分操作。例如，JMeter 不会执行 HTML 页面中包含的 JavaScript，而且 JMeter 对于 HTML 页面的处理方式也与浏览器不同（JMeter 有可能会以 HTML 方式读取服务器响应，但其所耗时间不会被计算到任何一次采样中）。

2.1.2　JMeter 常用术语

- 采样器（Samplers）：采样器是 JMeter 测试脚本的基础单元，用户可以用它来向服务器发出一个特定的请求，采样器会在超时前等待服务器的响应。
- 逻辑控制器（Logic Controllers）：用户通过逻辑控制器来控制 JMeter 测试脚本的执行顺序，以便测试能够按照用户期望的顺序和逻辑执行。
- 监听器（Listeners）：监听器被用来收集测试结果信息，并以用户指定的方式加以展示。
- 配置元件（Configuration Elements）：配置元件被用来设置一些 JMeter 测试脚本公用的信息。
- 断言（Assertions）：断言被用来验证服务器实际返回的信息与用户期望的情况是否相符。
- 定时器（Timers）：定时器被用来保存 JMeter 测试脚本与时间相关的一些信息，例如思考时间（Think Time）。
- 前置处理器（Pre-Processors）：在前置处理器的作用范围内，任何采样器被执行前，都要先执行前置处理器。
- 后置处理器（Post-Processors）：在后置处理器的作用范围内，任何采样器被执行后，都要执行对应的后置处理器。
- 测试计划（Test Plan）：测试计划是 JMeter 测试脚本的根节点，关于整个测试脚本的一些基础设置，可以在测试计划中设定，例如用户定义变量。
- 线程组（Thread Group）：线程组定义了一个虚拟用户池，其中每一个虚拟用户都使用同样的测试脚本。
- 工作台（WorkBench）：工作台被用来保存暂时不使用的测试元素，当测试人员保存测试计划时，工作台中的内容不会被一起保存。

2.1.3　JMeter 测试结果字段的意义

JMeter 测试结果字段的意义如下。

- Label：定义 HTTP 请求名称。
- Samples：表示这次测试中一共发出了多少个请求。

- Average：平均响应时长——默认情况下是单个 Request 的平均响应时长，当使用了 Transaction Controller 时，也可以以 Transaction 为单位显示平均响应时长。
- Median：中位数，也就是 50%用户的响应时长。
- 90% Line：90%用户的响应时长。
- Min：访问页面的最小响应时长。
- Max：访问页面的最大响应时长。
- Error%：错误请求的数量/请求的总数。
- Throughput：默认情况下表示每秒完成的请求数（Request per Second），当使用了 Transaction Controller 时，也可以表示类似 LoadRunner 的 Transaction per Second 数。
- KB/Sec：每秒从服务器端接收到的数据量。

在这里提到 JMeter 的测试结果字段，可能会让读者朋友们感到困惑，作者的目的仅仅是让读者朋友对使用 JMeter 做 Web 性能测试可以获取的测试结果有个初步印象。

2.2 JMeter 工作原理

让我们以 Web 性能测试为例，介绍 JMeter 的工作原理，如图 2-1 所示。JMeter 可以作为 Web 服务器与浏览器之间的代理网关，以便捕获浏览器的请求和 Web 服务器的响应，如此就可以很容易地生成性能测试脚本。有了性能测试脚本，JMeter 就可以通过线程组来模拟真实用户对 Web 服务器的访问压力。这与 LoadRunner 的工作原理基本一致。

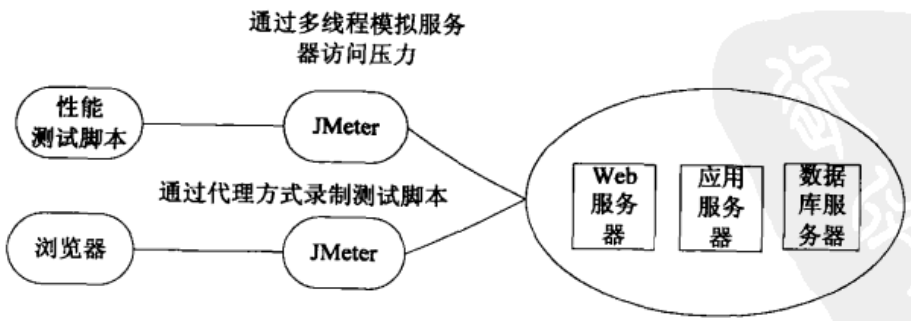


图 2-1 JMeter 工作原理

2.3　JMeter 的安装与目录结构

2.3.1　JMeter 安装配置要求

1．Java 版本

JMeter 要求 JVM 1.5 或更高版本。

2．操作系统

JMeter 是一个 100%的 Java 应用程序，它在任何支持完整 Java 实现的系统上都应该能正常运行。JMeter 在如下操作系统中被测试过：

- UNIX（Solaris，Linux 等）
- Windows（98，NT，XP 等）
- OpenVMS Alpha 7.3+

2.3.2　JMeter 目录结构

JMeter 的安装非常简单，只需从官网（http://jakarta.apache.org/site/downloads/downloads_JMeter.cgi）下载并解压之后即可使用。写作本书时 JMeter 的最新版本为 2.4，下载并解压后的 JMeter 包含以下几个目录：bin、docs、extras、lib 及 printable_docs 等，如图 2-2 所示。

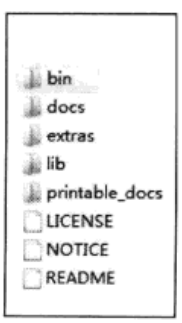


图 2-2　JMeter 目录结构

1．bin 目录

在 Windows 操作系统下，运行 JMeter.bat 就可以看到 JMeter 的 Swing GUI 客户端，如图 2-3

所示。JMeter 图形用户界面简洁而实用，用户可以通过它来完成各种测试所需的配置。

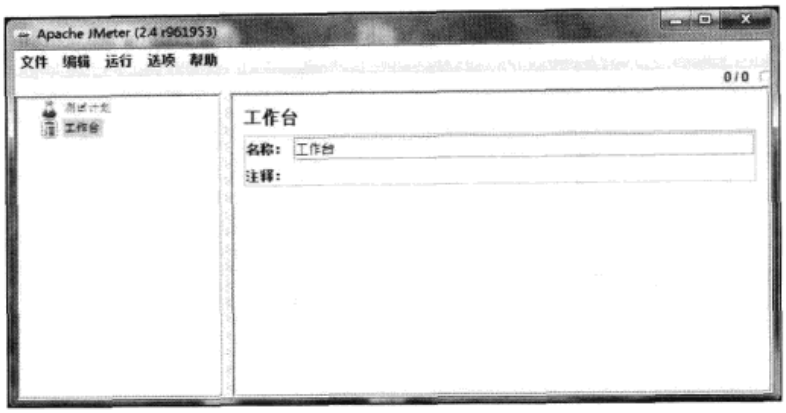


图 2-3 JMeter 图形用户界面

需要注意的是，JMeter 用户可以根据运行 JMeter 的计算机配置，来适当调整 JMeter.bat 中的 JVM 调优设置，如下所示：

```
set HEAP=-Xms512m -Xmx512m
set NEW=-XX:NewSize=128m -XX:MaxNewSize=128m
set SURVIVOR=-XX:SurvivorRatio=8 -XX:TargetSurvivorRatio=50%
set TENURING=-XX:MaxTenuringThreshold=2
set EVACUATION=-XX:MaxLiveObjectEvacuationRatio=20%
set RMIGC=-Dsun.rmi.dgc.client.gcInterval=600000
-Dsun.rmi.dgc.server.gcInterval=600000
set PERM=-XX:PermSize=64m -XX:MaxPermSize=64m
rem set DEBUG=-verbose:gc -XX:+PrintTenuringDistribution
```

根据经验，堆值(HEAP)最多设置为物理内存的一半，默认设置为 512MB。如果堆值(HEAP)超过物理内存的一半，JMeter 运行速度会变慢，甚至会出现“内存溢出”的错误。如果 JMeter 出现运行异常，测试人员可以在 JMeter.log 日志文件中看到相关信息。

另外，如果用户想要获取更详细的日志，可以修改 bin 目录下 JMeter.properties 文件中的一个属性 log_level. JMeter。默认为 INFO，查错时可以改为 DEBUG。

```
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
```

JMeter 的属性设置也可以在图形用户界面中展示，选择“添加”→“非测试元件”→“Property Display”，如图 2-4 所示。

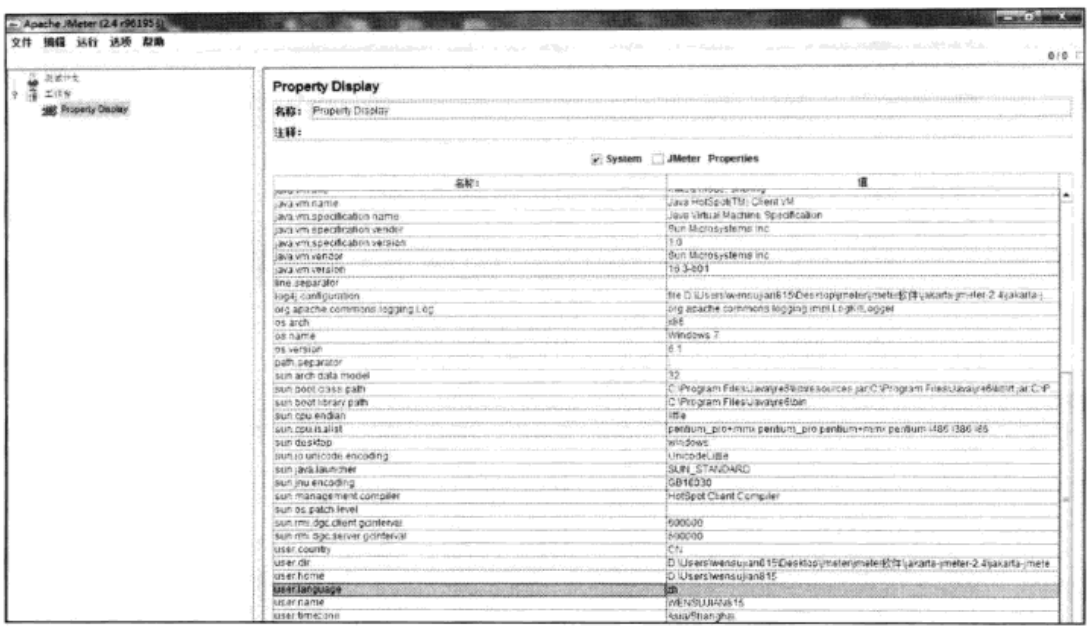


图 2-4　JMeter Property Display

从图 2-4 中可以看到 user.language 属性，默认值 zh 代表中文。作者个人更喜欢设置为 en，即采用英文的菜单和元件名，因为翻译过来的中文名称并不准确，有可能误导用户。

2．docs 和 printable_docs 目录

docs 目录下的文件是 JMeter 的 Java Docs，而 printable_docs 的 usermanual 子目录下的内容是 JMeter 的用户手册文档，其中 component_reference.html 是最常用到的核心元件帮助文档。用户也可以在 JMeter 图形用户界面的帮助菜单中，查看测试元件的帮助信息，如图 2-5 所示。printable_docs 的 demos 子目录下有一些常用的 JMeter 脚本案例，可以作为参考。

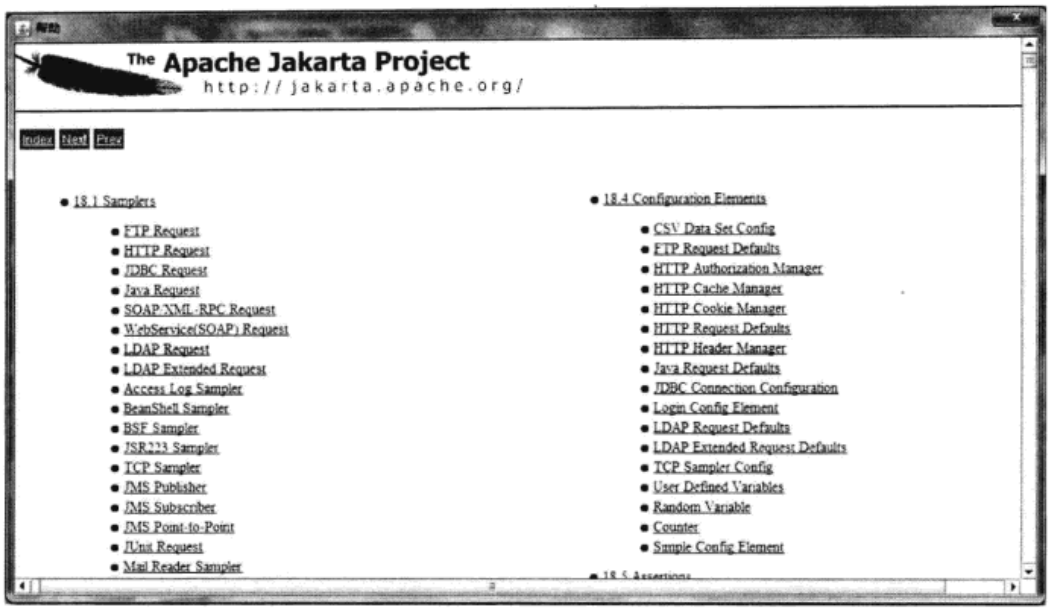


图 2-5　JMeter 帮助文档

3．extras 目录

extras 目录下的文件提供了对构建工具 Ant 的支持，可以使用 Ant 来实现测试自动化，例如批量脚本执行，产生 HTML 格式的报表。测试运行时，可以把测试数据记录下来，JMeter 会自动生成一个.jtl 文件。将该文件放到 extras 目录下，运行“ant –Dtest=文件名 report”，就可以生成测试统计报表。

4．lib 目录

lib 目录下包含两个子目录，其中 ext 子目录存放有 JMeter 的核心 jar 包，另一个 junit 子目录存放 JUnit 测试脚本。用户扩展所依赖的包，应该直接放在 lib 目录下，而非 lib/ext 下。

2.4　如何运行 JMeter

要启动 JMeter，在 Windows 下运行 jmeter.bat，或者在 UNIX 下运行文件 jmeter，这两个文件都可以在 bin 目录下找到。在一个很短的等待之后，JMeter 的图形用户界面就会出现。在 bin 目录中，还有其他几个测试人员可能会用到的脚本。

Windows 脚本文件（.cmd 要求 Windows 2000 及其后续版本）。

- jmeter.bat：运行 JMeter（默认 GUI 模式）。

- jmeter-n.cmd：加载一个 JMX 文件，并在非 GUI 模式下运行。
- jmeter-n-r.cmd：加载一个 JMX 文件，并在远程非 GUI 模式下运行。
- jmeter-t.cmd：加载一个 JMX 文件，并在 GUI 模式下运行。
- jmeter-server.bat：以服务器模式启动 JMeter。
- mirror-server.cmd：在非 GUI 模式下启动 JMeter 镜像服务器。
- shutdown.cmd：关闭一个非 GUI 实例（优雅的）。
- stoptest.cmd：停止一个非 GUI 实例（中断式的）。

小贴士　关键字 LAST 可以与 jmeter-n.cmd、jmeter-t.cmd 和 jmeter-n-r.cmd 一起使用，这就意味着最近一个测试计划是交互式地运行。

通过环境变量 JVM_ARGS，我们可以修改在 jmeter.bat 中定义的 JVM 设置。例如：

```
set JVM_ARGS="-Xms1024m -Xmx1024m -Dpropname=propvalue"
jmeter -t test.jmx ...
```

UNIX 脚本文件，应该能在绝大多数 Linux/UNIX 系统下运行。

- jmeter：运行 JMeter（默认 GUI 模式）。定义了一些 JVM 设置，但并不是对所有 JVM 都生效。
- jmeter-server：以服务器模式启动 JMeter（通过合适的参数来调用 jmeter 脚本）。
- jmeter.sh：没有指定 JVM 选项的非常基础的 jmeter 脚本。
- mirror-server.sh：在非 GUI 模式下启动 JMeter 镜像服务器。
- shutdown.sh：关闭一个非 GUI 实例（优雅的）。
- stoptest.sh：停止一个非 GUI 实例（中断式的）。

如果当前使用的 JVM 不支持某些 JVM 选项，那么测试人员就有必要修改 jmeter shell 脚本。另外用户可以使用 JVM_ARGS 环境变量来修改或者添加 JVM 选项，例如：

```
JVM_ARGS="-Xms1024m -Xmx1024m" jmeter -t test.jmx [etc.]
```

就会覆盖脚本中的堆设置。

1. JMeter 的 Classpath

JMeter 会通过以下目录中的 jar 文件加载类。

- JMETER_HOME/lib：公用包。

- JMETER_HOME/lib/ext：包含 JMeter 元件和补丁。

如果测试人员开发了新的 JMeter 元件，那么需要将对应 jar 包放到 JMeter 的 lib/ext 目录下。JMeter 会自动加载这些 jar 包中的 JMeter 元件。

公用支持的 jar 包，应该被放在 lib 目录下。如果测试人员不想将 JMeter 扩展包放在 lib/ext 目录下，那么就需要在 jmeter.properties 文件中定义属性 search_paths。不要在 lib/ext 目录中放置公用包，它应该只供 JMeter 元件使用。

其他 jar 包（JDBC、JMS 实现包，以及任何其他支持包）都应该被放置在 lib 目录，而非 lib/ext 目录下。

小贴士 JMeter 只会寻找.jar 文件，而非.zip。

测试人员还可以加载放在$JAVA_HOME/jre/lib/ext 目录中的公用包，或者在 jmeter.properties 文件中定义 user.classpath 属性。

需要注意的是，设置环境变量 CLASSPATH 对 JMeter 不起作用。这是因为 JMeter 是以“java-jar”方式启动，而该 Java 命令会默默地忽略 CLASSPATH 变量，而且使用–jar 可以携带 -classpath/-cp 选项（这对所有 Java 程序都是一样的，并非只针对 JMeter）。

2. 使用代理服务器

如果测试对象隐藏在防火墙/代理服务器之后，那么测试人员需要向 JMeter 提供防火墙/代理服务器的主机名和端口号。用户可以通过命令行来运行 jmeter.bat/jmeter 文件，并携带如下参数，以达到提供主机名和端口号的目的。

- -H [代理服务器主机名或者 IP 地址]。
- -P [代理服务器端口]。
- -N [不使用代理的主机]（例如 *.apache.org|localhost）。
- -u [代理验证的用户名-如果要求的话]。
- -a [代理验证的密码-如果要求的话]。

例如：jmeter -H my.proxy.server -P 8000 -u username -a password -N localhost。

小贴士 JMeter 有内置的 HTTP 代理服务器，它可以被用来录制 HTTP 或者 HTTPS 协议的浏览器会话。请不要将它与上面描述的内容混淆起来，两者的功能完全不同。

3．非 GUI 模式（命令行模式）

对于非交互式测试，测试人员可以选择不使用 GUI 来运行 JMeter。要达到这一目的，可以使用如下命令行选项：

- -n 指明 JMeter 以非 GUI 模式运行。
- -t [JMX 文件（其中包含测试计划）的名称]。
- -l [JTL 文件（存放测试采样数据）的名称]。
- -j [JMeter 日志文件的名称]。
- -r 在 JMeter 属性“remote_hosts”中定义的服务器上远程运行测试脚本。
- -R [远程服务器列表] 在指定的远程服务器上运行测试。

这些 JMeter 脚本同样允许测试人员指明可选的防火墙/代理服务器信息：

- -H [代理服务器主机名或者 IP 地址]。
- -P [代理服务器端口号]。

4．服务器模式

JMeter 支持分布式测试，即在远程节点上以服务器模式运行 JMeter，并通过 JMeter 图形用户界面（GUI）来控制这些服务器。测试人员也可以通过非 GUI 模式去运行远程测试。要想启动这些服务器，测试人员可以在服务器节点上运 jmeter-server/jmeter-server.bat。

这些 JMeter 脚本同样允许测试人员指明可选的防火墙/代理服务器信息：

- -H [代理服务器主机名或者 IP 地址]。
- -P [代理服务器端口号]。

例如：jmeter-server -H my.proxy.server -P 8000。

如果用户希望这些远程服务器在单次测试运行后退出，那么请定义如下 JMeter 属性 server.exitaftertest=true。

要想通过非 GUI 模式在客户端运行测试，使用如下命令：

```
jmeter -n -t testplan.jmx -r [-Gprop=val] [-Gglobal.properties] [-Z]
```

- -G 被用来定义要在服务器中设置的 JMeter 属性。
- -Z 意味着在测试结束后退出服务器。

- -Rserver1,server2-可以用来代替-r，该选项会指定一系列远程服务器，但并不会重新定义 remote_hosts 这一属性。

5. 通过命令行重置 JMeter 属性

Java 系统属性、JMeter 属性、日志属性都可以通过命令行直接重置（无须修改 jmeter.properties 文件）。要达到这一目的，可以使用如下命令行选项。

- -D[prop_name]=[value]：定义一个 Java 系统属性值。
- -J[prop name]=[value]：定义一个本地 JMeter 属性。
- -G[prop name]=[value]：定义一个 JMeter 属性，并发往所有远程服务器。
- -G[propertyfile]：定义一个文件，其中包含有 JMeter 属性，并将该文件发往所有远程服务器。
- -L[category]=[priority]：重置一个日志设置，对特定类型的日志设定优先级。

例子：

```
jmeter -Duser.dir=/home/mstover/jmeter_stuff \
-Jremote_hosts=127.0.0.1 -Ljmeter.engine=DEBUG

jmeter -LDEBUG
```

小贴士 JMeter 在启动初始阶段处理命令行设置的属性，但此时日志系统已经初始化完毕，因此，如果用户使用-J 标志去更新 log_level 或者 log_file 属性，就不会有任何作用。

6. 日志和错误信息

JMeter 不会在发生运行错误时弹出对话框，因为这样做会影响到测试运行。另外，JMeter 还会忽略函数或者变量引用的任何错误。

如果测试期间发生了一个错误，那么 JMeter 会将一条错误信息写到日志文件中去。JMeter 使用的日志文件名称被定义在 jmeter.properties 文件中（或者在命令行中使用-j 选项指定，参见本书后续内容）。默认情况下是 jmeter.log 文件，用户可以在 bin 目录中找到它。

JMeter 2.2 版本之后添加了一个新的命令行选项：-j jmeterlogfile。在初始属性文件被读取之后，JMeter 就会处理该命令选项，优先于其他任何属性的处理。如此一来，用户就可以不使用默认的 jmeter.log 日志文件。用户可以使用测试计划名来命名日志文件，例如，测试计划 Test27.jmx 对应日志文件 Test27.log。

日志文件中不仅记录录制脚本时产生的错误，还记录测试运行时的错误。例如：

```
10/17/2003 12:19:20 PM INFO  - jmeter.JMeter: Version 1.9.20031002
10/17/2003 12:19:45 PM INFO  - jmeter.gui.action.Load: Loading file:
c:\mytestfiles\BSH.jmx
10/17/2003 12:19:52 PM INFO  - jmeter.engine.StandardJMeterEngine: Running
the test!
10/17/2003 12:19:52 PM INFO  - jmeter.engine.StandardJMeterEngine: Starting
1 threads for group BSH. Ramp up = 1.
10/17/2003 12:19:52 PM INFO  - jmeter.engine.StandardJMeterEngine: Continue
on error
10/17/2003 12:19:52 PM INFO  - jmeter.threads.JMeterThread: Thread BSH1-1
started
10/17/2003 12:19:52 PM INFO  - jmeter.threads.JMeterThread: Thread BSH1-1
is done
10/17/2003 12:19:52 PM INFO  - jmeter.engine.StandardJMeterEngine: Test has
ended
```

日志文件对定位错误原因很有帮助（JMeter 不会弹出错误对话框而中断测试）。

7. 全部命令行选项列表

通过命令行运行“jmeter -?”，可以查询命令行选项列表，如下所示：

```
-h, --help
       print usage information and exit
 -v, --version
       print the version information and exit
-p, --propfile {argument}
       the jmeter property file to use
-q, --addprop {argument}
       additional property file(s)
-t, --testfile {argument}
       the jmeter test(.jmx) file to run
-j, --jmeterlogfile {argument}
       the jmeter log file
-l, --logfile {argument}
       the file to log samples to
-n, --nongui
```

```
        run JMeter in nongui mode
-s, --server
        run the JMeter server
-H, --proxyHost (argument)
        Set a proxy server for JMeter to use
-P, --proxyPort (argument)
        Set proxy server port for JMeter to use
-u, --username (argument)
        Set username for proxy server that JMeter is to use
-a, --password (argument)
        Set password for proxy server that JMeter is to use
-J, --jmeterproperty (argument)=(value)
        Define additional JMeter properties
-G, --globalproperty (argument)[=(value)]
        Define Global properties (sent to servers)
        e.g. -Gport=123
         or -Gglobal.properties
-D, --systemproperty (argument)=(value)
        Define additional System properties
-S, --systemPropertyFile (filename)
        a property file to be added as System properties
-L, --loglevel (argument)=(value)              Define loglevel:
[category=]level
        e.g. jorphan=INFO or jmeter.util=DEBUG
-r, --runremote (non-GUI only)
        Start remote servers (as defined by the jmeter property
        remote_hosts)
-R, --remotestart  server1,... (non-GUI only)
        Start these remote servers (overrides remote_hosts)
-d, --homedir (argument)
        the jmeter home directory to use
-X, --remoteexit
        Exit the remote servers at end of test (non-GUI)
```

小贴士 JMeter 日志文件名可以使用简单日期格式，即在日志文件名后附带当前日期。日期格式要用单引号括起来，例如：'jmeter_'yyyyMMddHHmmss'.log'。

关键字 LAST 可以与标志-t、-j 或者 -l 一起使用，这就意味着最近一个测试计划是交互式地运行。

2.5　配置 JMeter

如果想要改变与 JMeter 运行相关的属性，那么测试人员可以修改 bin 目录下的 jmeter.properties 文件，或者根据 jmeter.properties 文件创建用户自己的属性文件，并在命令行中指定属性文件名。

在 JMeter 2.2 版本以后，引入了用户属性文件 **user.properties** 和系统属性文件 **system.properties**。这两个文件如果存在于 bin 目录中将被 JMeter 自动加载。

JMeter 用户经常用到的属性如表 2-1 所示。

表 2-1　JMeter 常用属性

属性	描述	是否必须
ssl.provider	如果不想使用 JMeter 内置的 Java 实现（SSL），测试人员可以指明自己的 SSL 实现类	否
xml.parser	用户可以指定 XML 解析器。默认值是：org.apache.xerces.parsers.SAXParser	否
remote_hosts	用逗号分隔的 JMeter 远程主机（或者主机:端口）列表。如果 JMeter 运行于分布式环境，那么需要指明运行有 JMeter 远程服务器的机器列表。这能帮助测试人员通过 GUI 来控制这些远程机器	否
not_in_menu	指明测试人员不想在 JMeter 菜单中看到的测试元件列表。 以后会有越来越多的测试元件加入 JMeter，测试人员可能希望定制 JMeter，以便它只显示测试人员经常使用到的测试元件。 测试人员可以在此处罗列测试元件的类名或者类标签（出现在 JMeter 用户界面中的字符串），那么它们就不会再出现在菜单中	否
search_paths	JMeter 搜索扩展类的路径列表（通过;分隔），例如扩展采样器。这是用于添加没有放在 lib/ext 目录之中的其他 jar	否
user.classpath	JMeter 搜索公用类的路径列表。这是用于添加没有放在 lib 目录之中的其他 jar	否
user.properties	包含有补充 JMeter 属性的文件名。该属性文件将在初始属性文件之后加载，但是会先于-q 和-J 选项处理	否
system.properties	包含有补充系统属性的文件名。该属性文件会先于-S 和-D 选项处理	否

2.6　JMeter 与 LoadRunner 优缺点对比

JMeter 与 LoadRunner 都是非常优秀的性能测试软件，它们孰优孰劣众说纷纭。本书作者认为，相对于 JMeter 而言，商业性能测试软件 LoadRunner 支持的测试协议更广，图形分析能力

更强，易用性也要强一些。但是这些并不是质的差距，LoadRunner 的基本功能，JMeter 都具备，只是没有那么完善而已。由于 LoadRunner 价格非常昂贵，因此这些差距是完全可以接受的。特别是对于中小软件企业而言，JMeter 的性价比无疑拥有致命的吸引力。读者朋友通过表 2-1 可以了解到 JMeter 与 LoadRunner 各自的优缺点。

表 2-2　JMeter 与 LoadRunner 对比

对比项	JMeter	LoadRunner
安装	简单，下载解压即可	复杂，LoadRunner 安装包大于 1GB，在一台主频 3.0、内存 1GB 的 PC 上安装，安装时间大于 1 小时
录制/回放模式	支持	支持
测试协议	偏少，用户可自行扩展	较多，用户不可自行扩展
分布式大规模压力测试	支持	支持
IP 欺骗功能	不支持	支持
图形报表	支持（较弱）	支持（很强）
测试逻辑控制	支持	支持
监控服务器资源（CPU、内存等）	支持	支持
功能测试	支持	不支持

2.7　本章小结

本章首先对 JMeter 做了一个概要介绍，主要介绍了 JMeter 的特点和一些技术术语，为后面的学习扫清障碍；接着介绍了 JMeter 的工作原理，使读者进一步了解 JMeter 是如何进行性能测试的；之后介绍了 JMeter 的安装和目录结构；最后介绍了 JMeter 和 LoadRunner 各自的优缺点。

第 3 章

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。

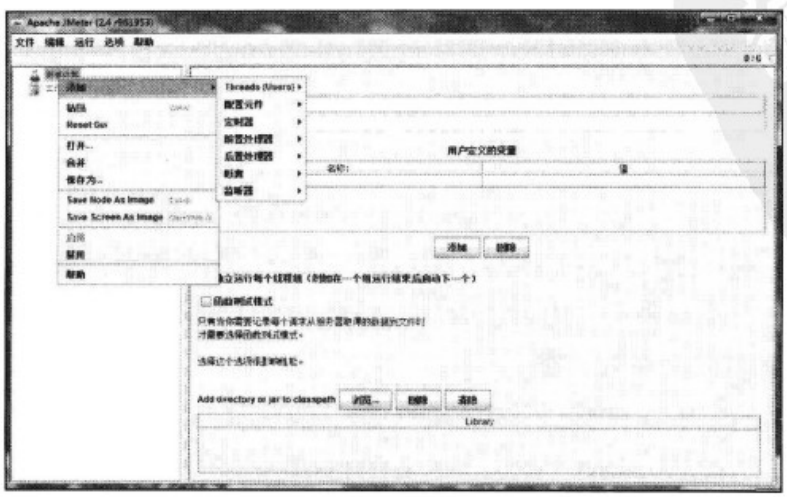


图 3-1 JMeter 添加测试元件

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。

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。

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。

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）。

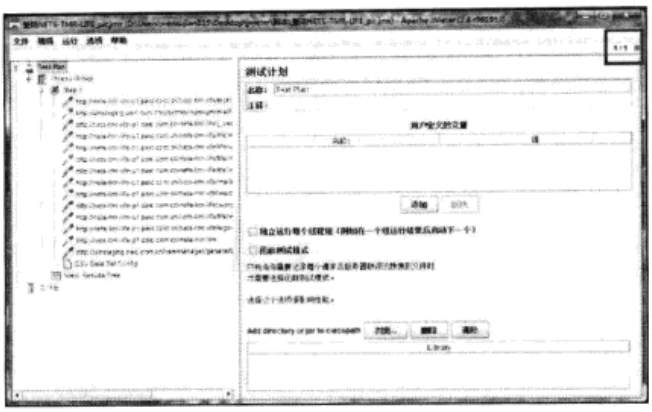


图 3-2　JMeter 处于运行状态

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。

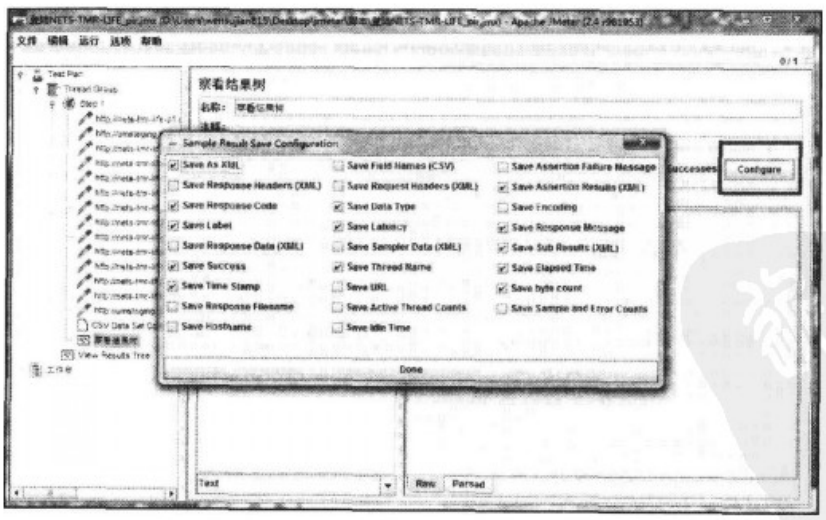


图 3-3 设定 JMeter 如何记录测试数据

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。

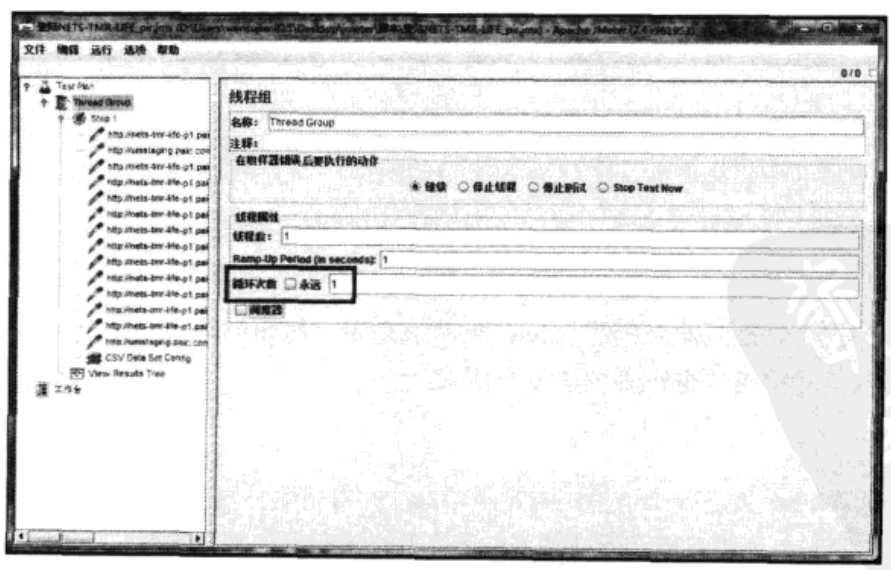


图 3-4　设定参数“循环次数”

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。

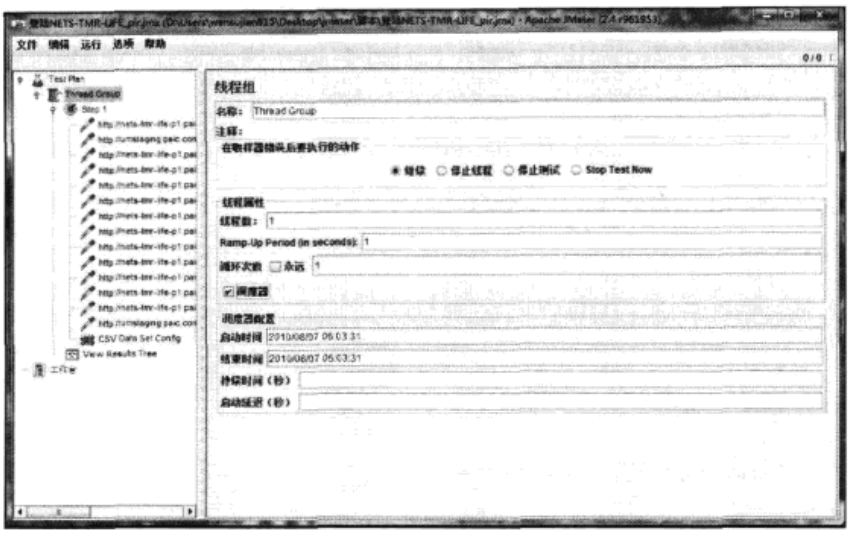


图 3-5　展开额外的调度器控制面板

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件的请求执行顺序。

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）。

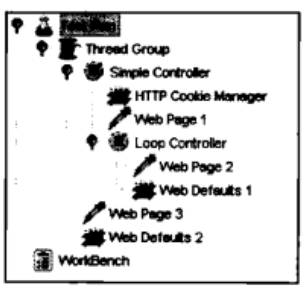


图 3-6 配置元件的作用域

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）

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，不过即便如此，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。

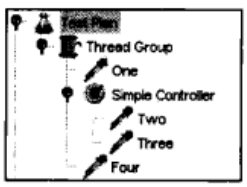


图 3-7 顺序规则举例

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。

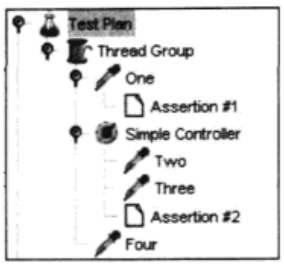


图 3-8　分层规则举例 1

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。

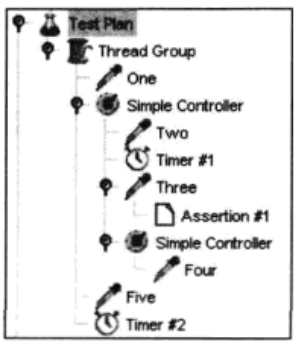


图 3-9　分层规则举例 2

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）。

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 并发用户）×（2 请求）×（重复 2 次）= 20 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。

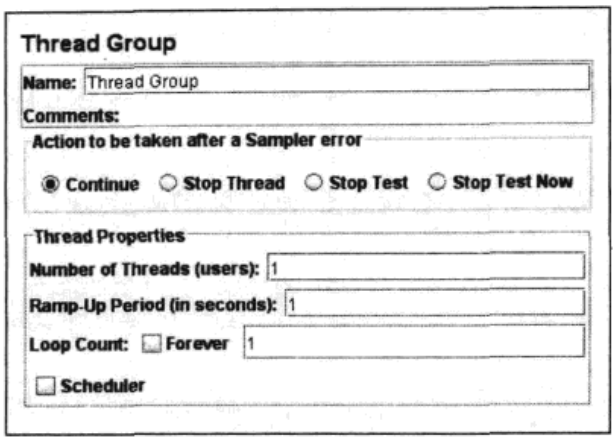


图 3-10　线程组的控制面板

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。

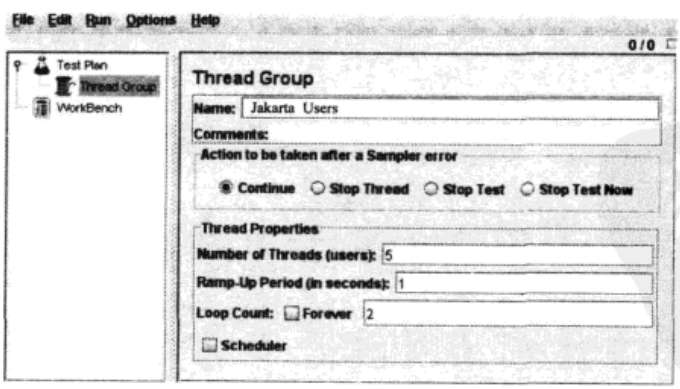


图 3-11　线程组 Jakarta Users

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。

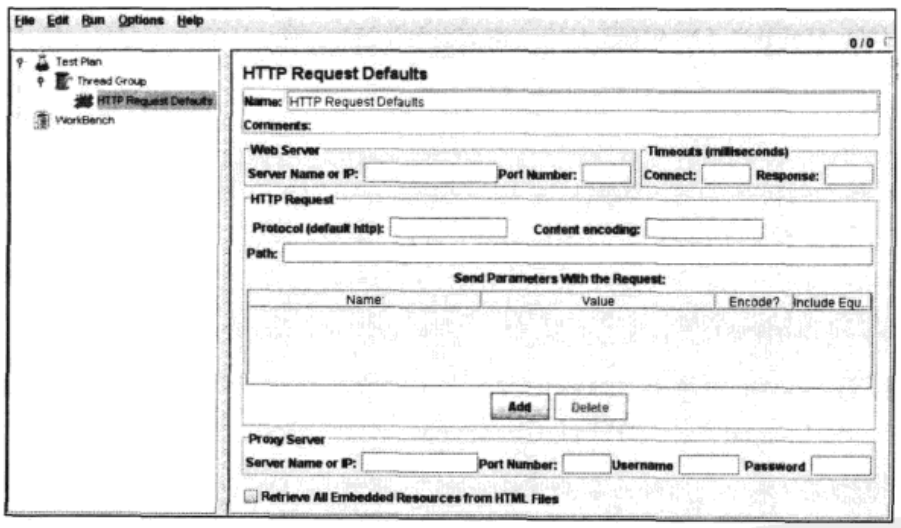


图 3-12　HTTP 请求默认值（HTTP Request Defaults）

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。

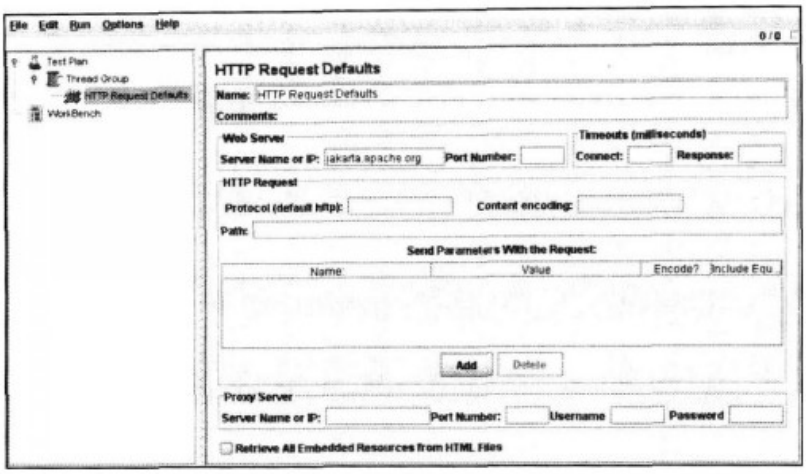


图 3-13　设置 Server Name or IP 域

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）。

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。

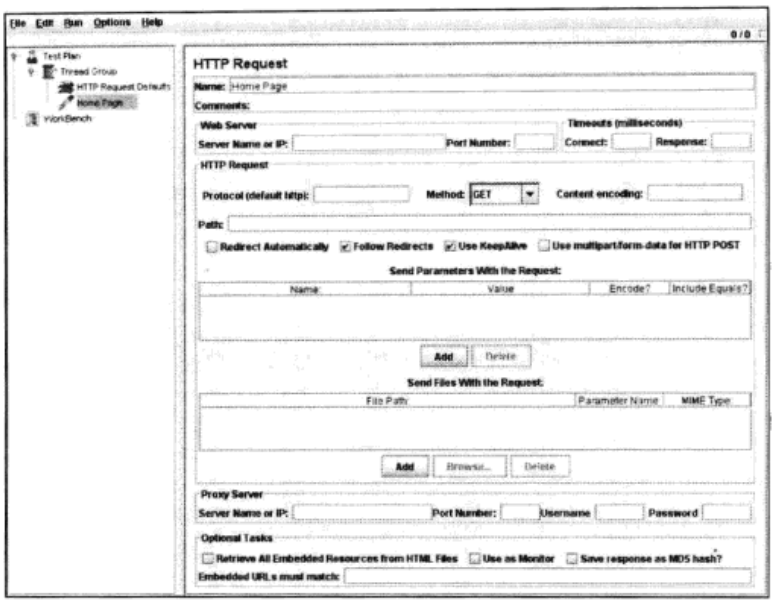


图 3-14　添加第一个 HTTP 请求（主页 http://jakarta.apache.org/）

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”。

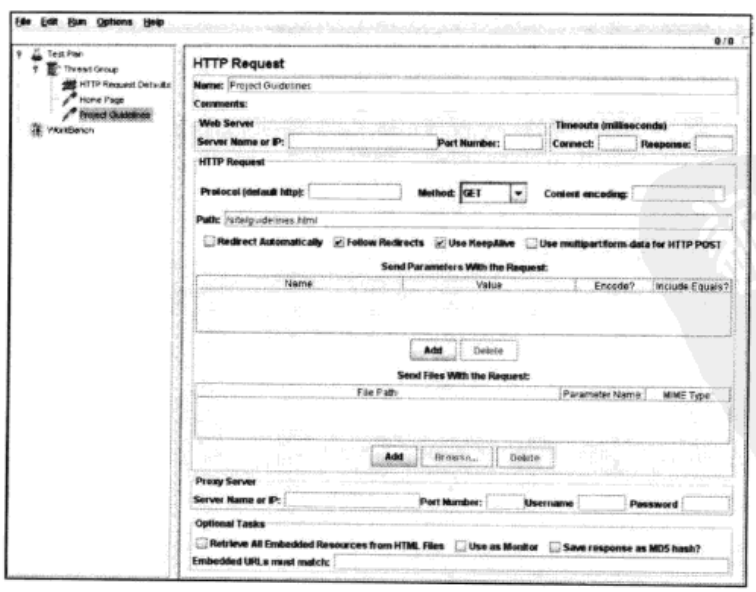


图 3-15　添加第二个 HTTP 请求（主页 http://jakarta.apache.org/）

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。

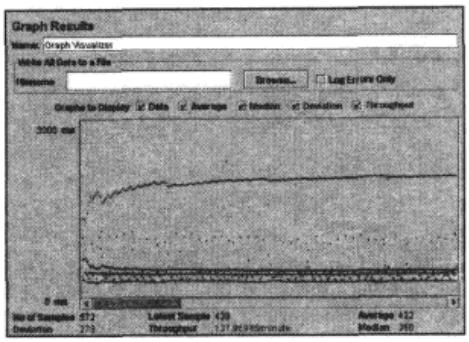


图 3-16　图形结果监听器

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

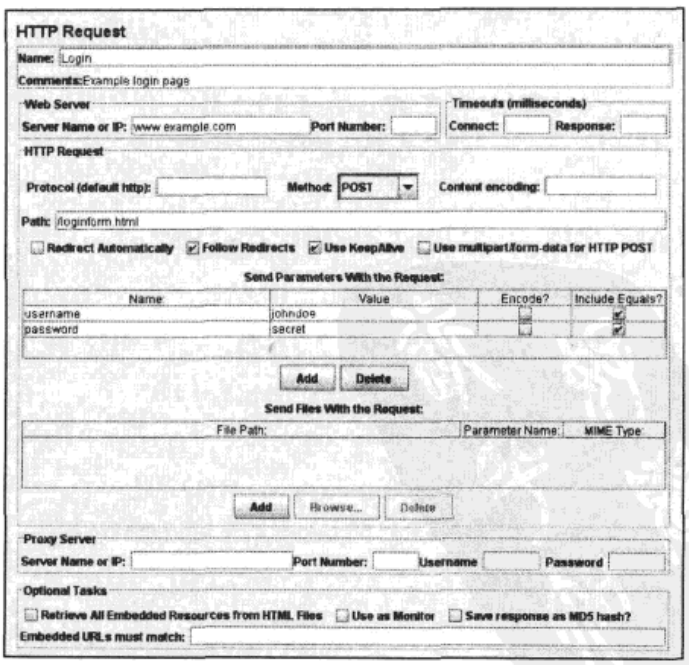


图 3-17　模拟 HTTP 登录请求

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。有些时候登录表单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。

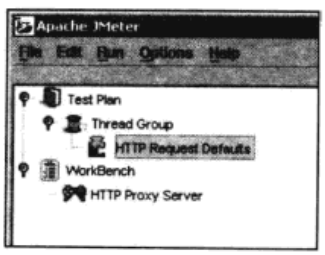


图 3-18　使用 JMeter 代理

（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。

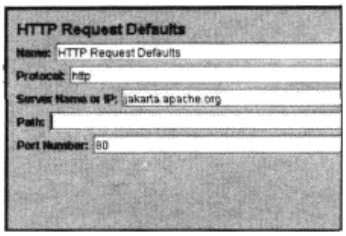


图 3-19　配置 HTTP 请求默认值（HTTP Request Defaults）

（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。

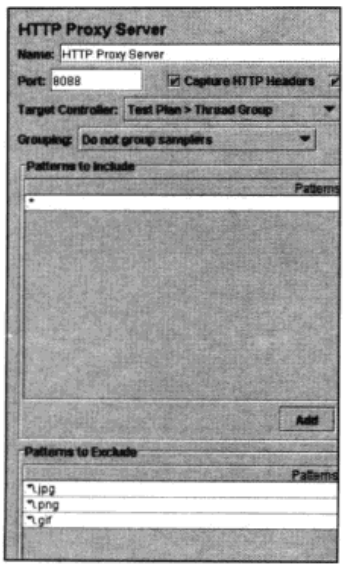


图 3-20　HTTP 代理服务器（HTTP Proxy Server）

（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。

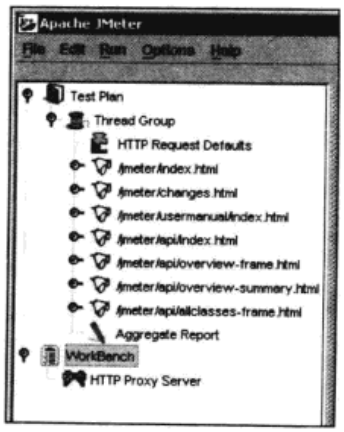


图 3-22　添加聚合报告

（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”。

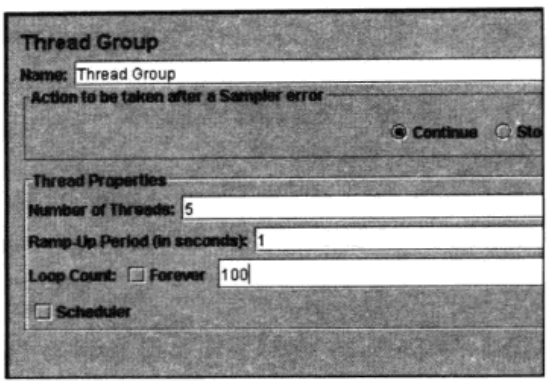


图 3-23　配置线程组

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“Crtl＋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。

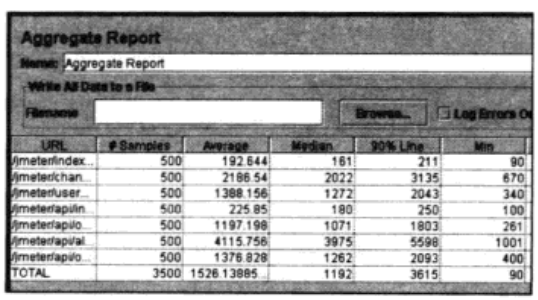


图 3-24　聚合报告

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。

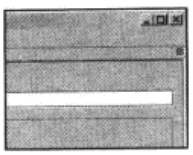


图 3-25　JMeter 运行提示

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。

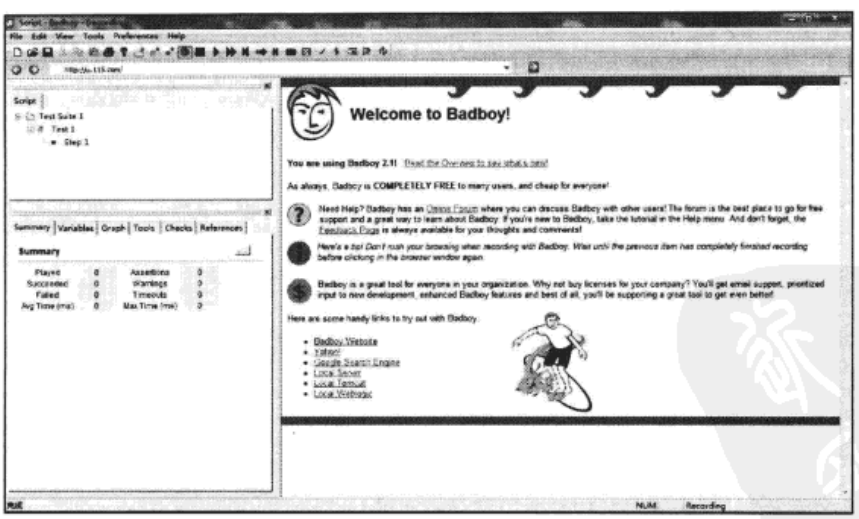


图 3-26　启动 Badboy

（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。

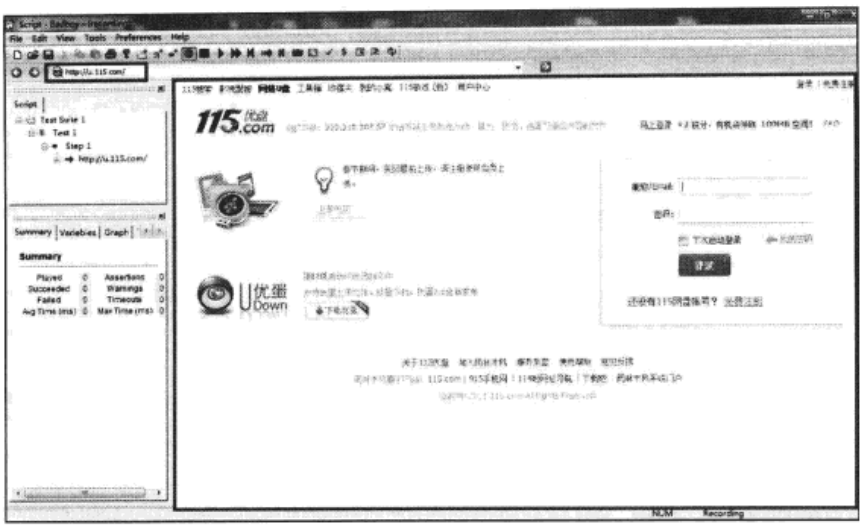


图 3-27　Badboy 访问测试网址

（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。

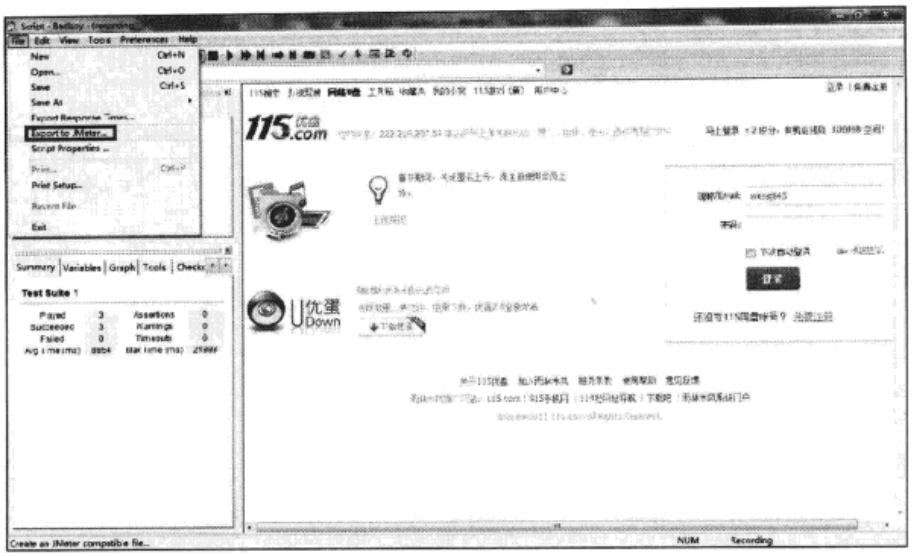


图 3-29　导出 Badboy 测试脚本（1）

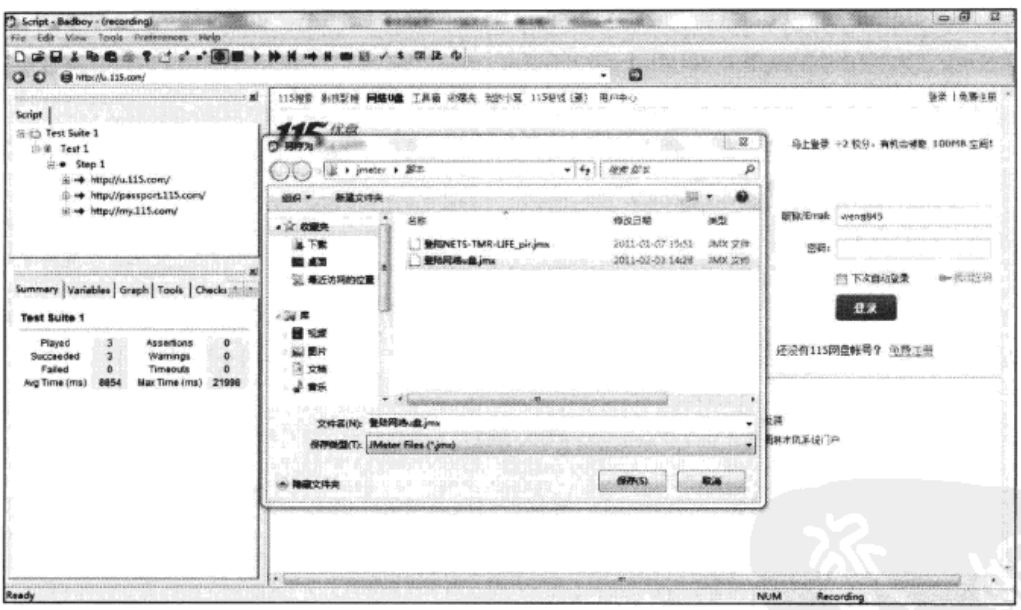


图 3-30　导出 Badboy 测试脚本（2）

（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。

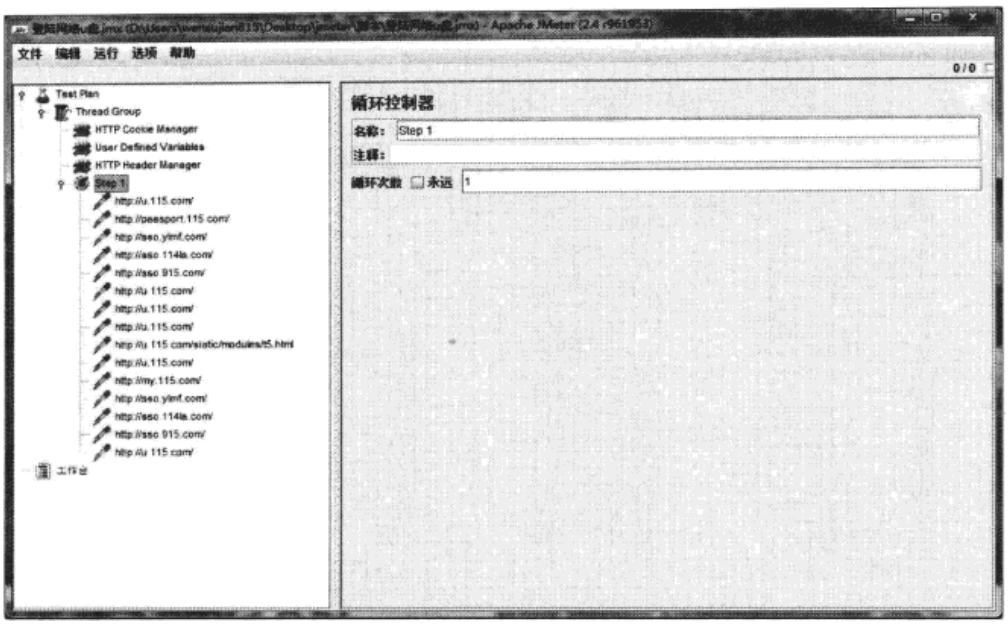


图 3-31　使用 JMeter 打开 Badboy 生成的测试脚本

（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。

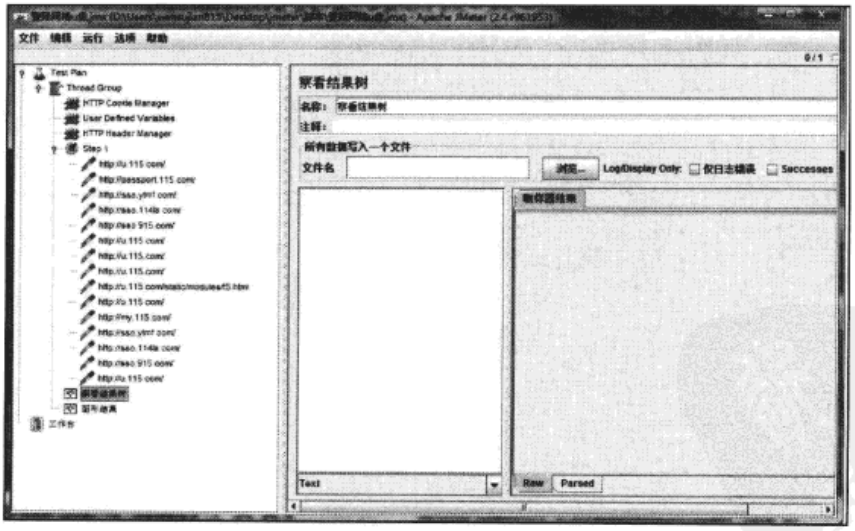


图 3-32　为测试脚本添加监听器

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“Crtl+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。

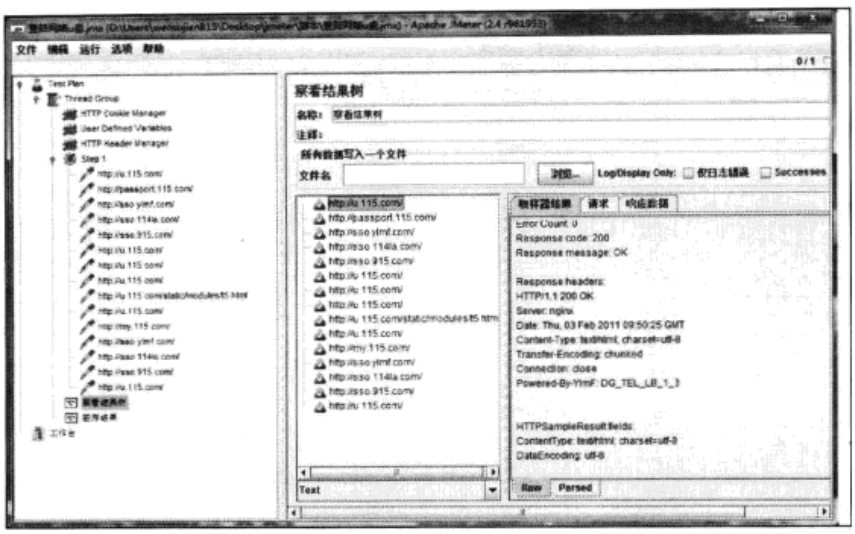


图 3-33 查看结果树

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。

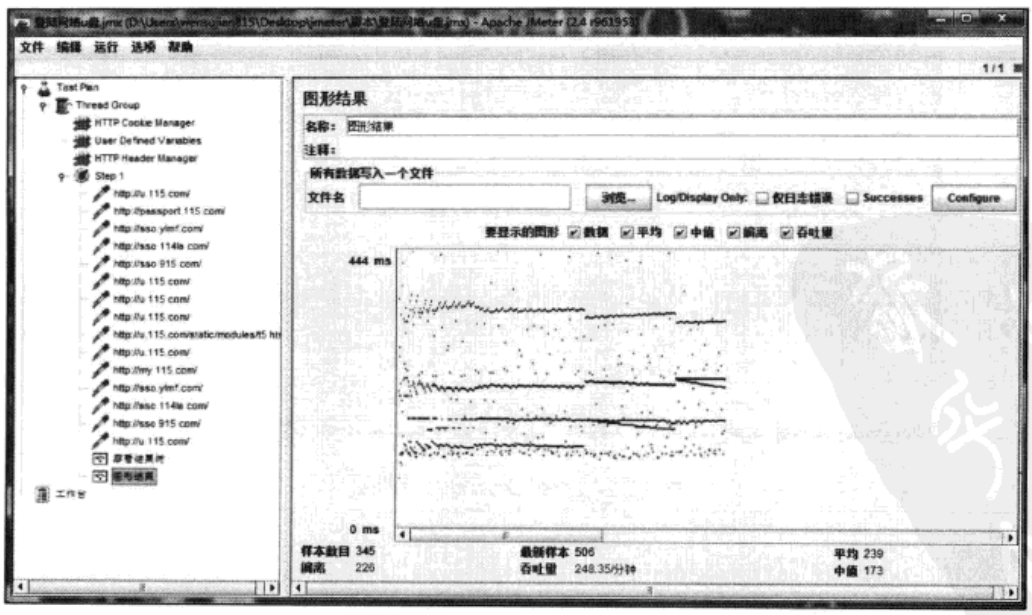


图 3-34 图形结果

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。

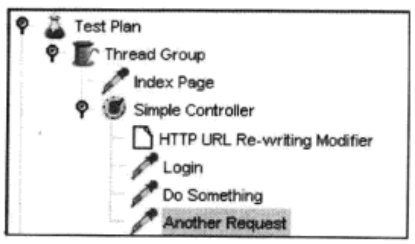


图 3-35　URL 回写例子的测试树

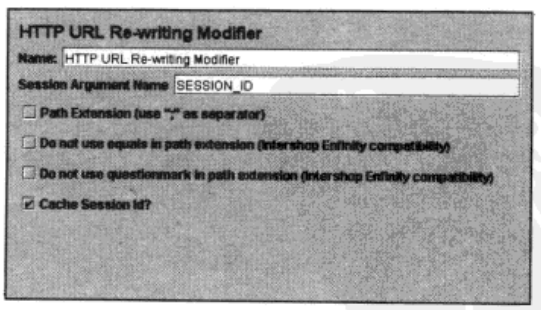


图 3-36　HTTP URL 回写修改器

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 4 章

数据库性能测试脚本开发

4.1 创建数据库测试计划

本节主要介绍如何创建一个简单的测试计划用于测试数据库服务器。在本节中测试人员会创建 10 个并发用户，而每个并发用户会发送两个 SQL 请求到数据库服务器。另外，每个用户都会运行测试 3 遍。因此，总共发送的测试数目是（10 并发用户）×（2 请求）×（重复 3 遍）= 60 JDBC 请求。要构建这一测试计划，测试人员需要使用到的测试元件包括：线程组（Thread Group）、JDBC 请求（JDBC Request）、图形结果（Graph Results）。

1. 添加并发用户

首先要做的是添加线程组（Thread Group），并修改其默认配置，如图 4-1 所示。

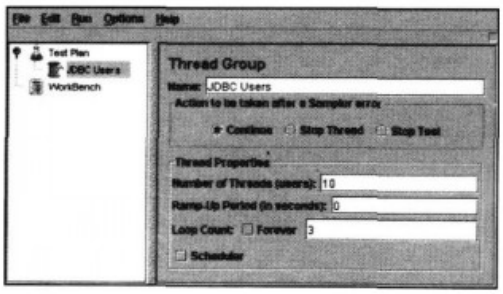


图 4-1 JDBC Users 线程组

2. 添加 JDBC 请求

现在我们已经定义了并发用户，接下来要做的是定义 JDBC 请求需要完成的操作。

先选中 JDBC Users(线程组),然后单击鼠标右键,在弹出的快捷菜单中选择“Add”→“Config Element” → “JDBC Connection Configuration” 命令。接着选中刚添加的新测试元件,观察它的控制面板,如图 4-2 所示。

设置好下面这些输入域(这里假设我们使用的是本地 MySQL 数据库)。

- Variable Name Bound to Pool:该值在整个测试计划中应该是唯一的,以便 JDBC 采样器区分不同的连接配置。
- Database URL:jdbc:mysql://localhost:3306/mydb,数据库连接串。
- JDBC Driver class:com.mysql.jdbc.Driver,数据库驱动程序。
- Username:guest,数据库用户名。
- Password:password for guest,对应的数据库用户密码。

其他输入域可以保持默认值不变。

JMeter 会使用控制面板中设定的默认配置来创建一个数据库连接池,这一连接池可以被 JDBC 请求所引用(使用“Variable Name”输入域)。测试人员可以在测试计划中加入多个 JDBC 配置,但是它们必须有不同的名字。另外,多个 JDBC 请求可以引用同一个连接池。

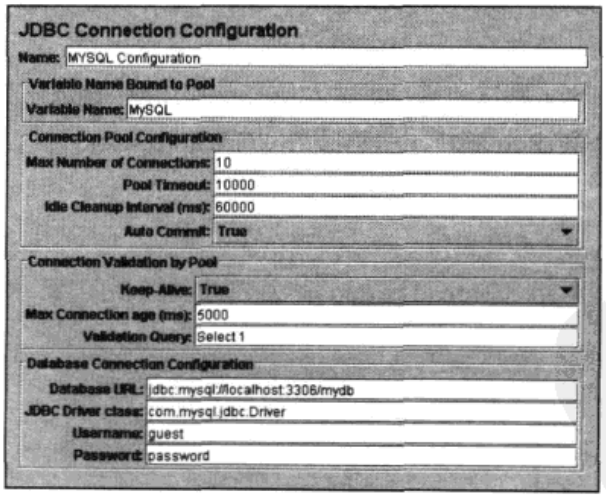


图 4-2 JDBC Connection Configuration(JDBC 连接默认配置)

再次选中 JDBC Users(线程组),然后单击鼠标右键,在弹出的快捷菜单中选择“Add”→“Sampler”→“JDBC Request”命令。接着选中新添加的测试元件(JDBC Request),并查看它的控制面板,如图 4-3 所示。

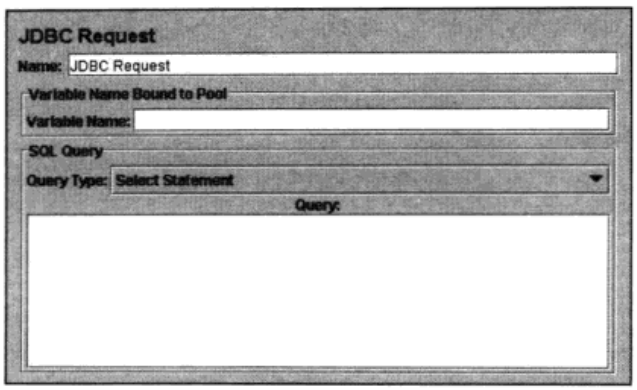


图 4-3　JDBC 请求（JDBC Request）

在我们的测试计划中，会发送两个 JDBC 请求。第一个是“Eastman Kodak Stock”，第二个是“Pfizer Stock”（当然在实际工作中测试人员必须做出一些修改以适应特定的数据库），如图 4-4 和图 4-5 所示。

从编辑如下属性开始。

（1）将名称改为“Kodak”。

（2）输入连接池的名称：MySQL（与 JDBC 配置元件的名称相同）。

（3）输入 SQL 查询语句。

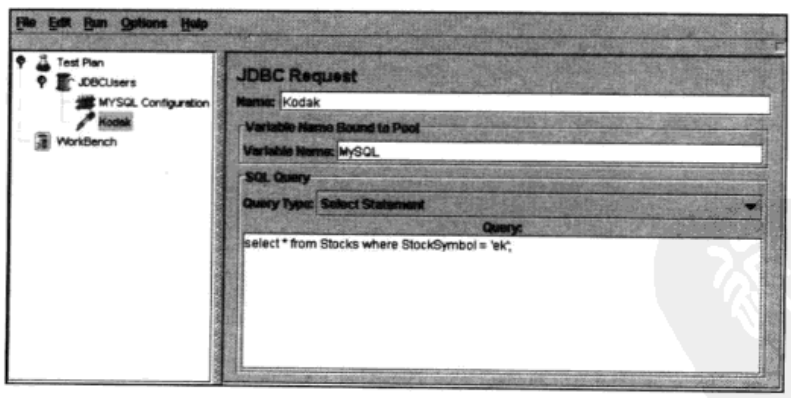


图 4-4　JDBC 请求（Kodak）

下一步，添加第二个 JDBC 请求并编辑如下属性。

（1）将名称改为“Pfizer”。

（2）输入 SQL 查询语句。

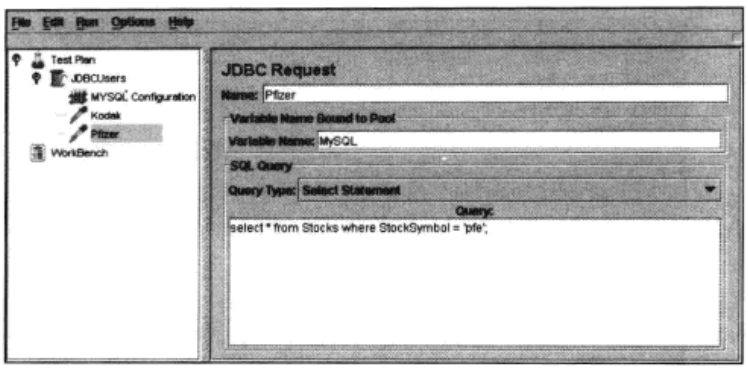


图 4-5 JDBC 请求（Pfizer）

3. 添加监听器用于查看/存储测试结果

测试人员需要做的最后一步，就是为测试计划添加一个监听器。该测试元件负责将所有 JDBC 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。

选中测试元件 JDBC Users，并添加一个监听器——图形结果（Graph Results），如图 4-6 所示。

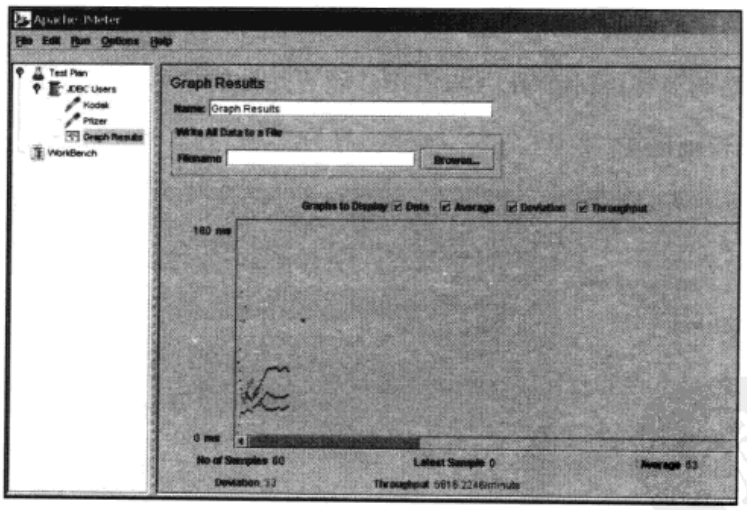


图 4-6 监听器——图形结果（Graph Results）

4.2 九步轻松搞定 Oracle 数据库性能测试

本节主要介绍如何使用 JMeter 完成针对 Oracle 数据库的性能测试。

（1）复制 Oracle 的 JDBC 驱动 jar 包文件（ojdbc14.jar）到 JMeter 的 lib 目录下。ojdbc14.jar 文件一般位于 Oracle 的安装目录下的 jdbc\lib 目录中。

（2）进入 bin 目录运行 jmeter.bat 启动 JMeter，如图 4-7 所示。

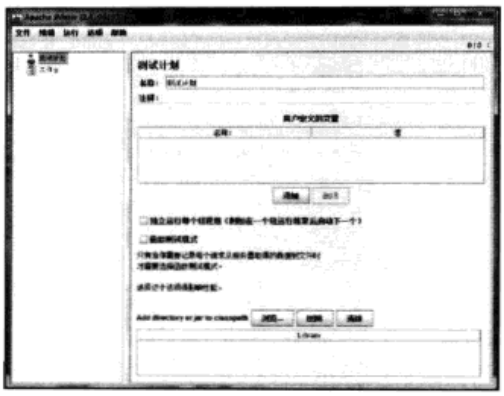


图 4-7　JMeter 图形用户界面

（3）在测试计划下新增一个线程组，如图 4-8 所示。

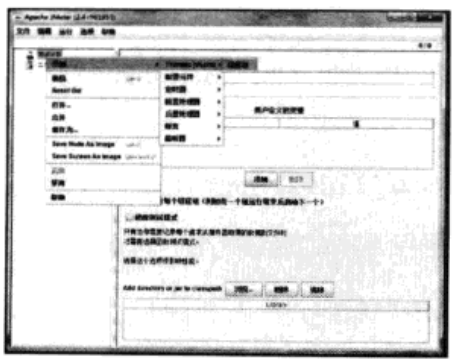


图 4-8　新增线程组

本例中测试线程组信息如图 4-9 所示，线程数 5 个，循环执行 10 次，即总共会有 50 次请求。

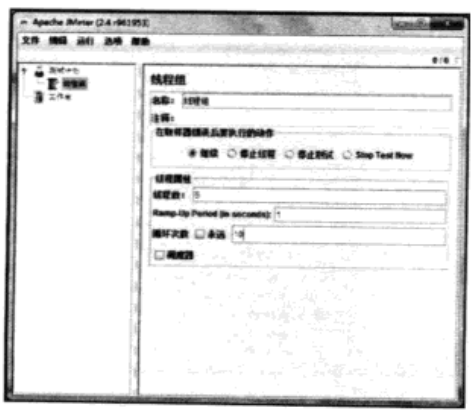


图 4-9　配置测试线程组

（4）再在线程组下新增一个 JDBC 连接池配置，如图 4-10 所示。

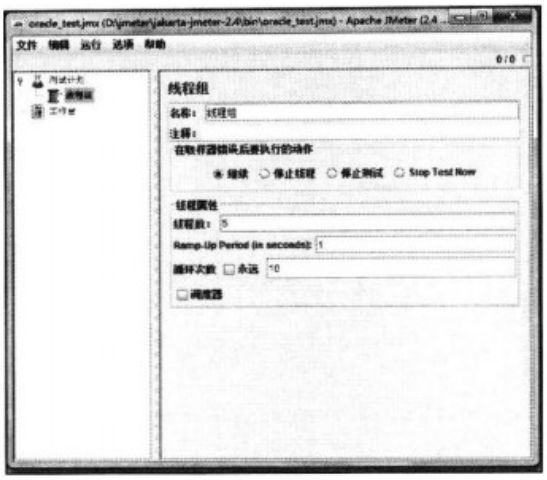


图 4-10　新增 JDBC 连接池配置

配置信息如图 4-11 所示，主要信息如下。

Database URL：数据库地址，格式为 jdbc:oracle:thin:@[IP 地址]:[端口号]:[实例名]。如下是本次的例子。

- jdbc:oracle:thin:@192.168.0.126:1521:ydgl。
- JDBC Driver class：数据库 JDBC 驱动类名，oracle.jdbc.driver.OracleDriver。
- Username：数据库连接用户名。
- Password：数据库连接密码。

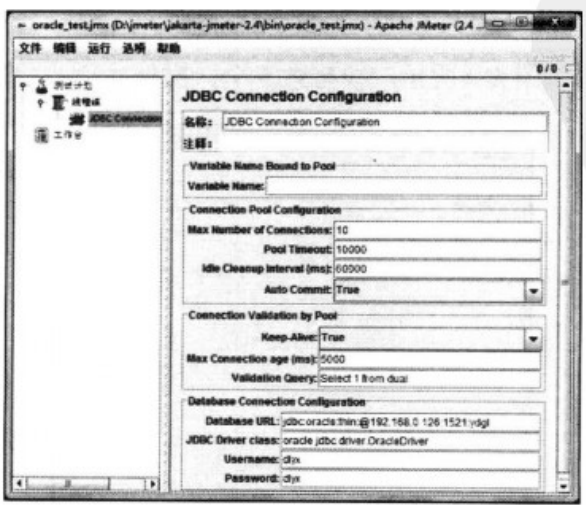


图 4-11　配置 JDBC 连接池

（5）接着新增一个 JDBC 请求，如图 4-12 所示。

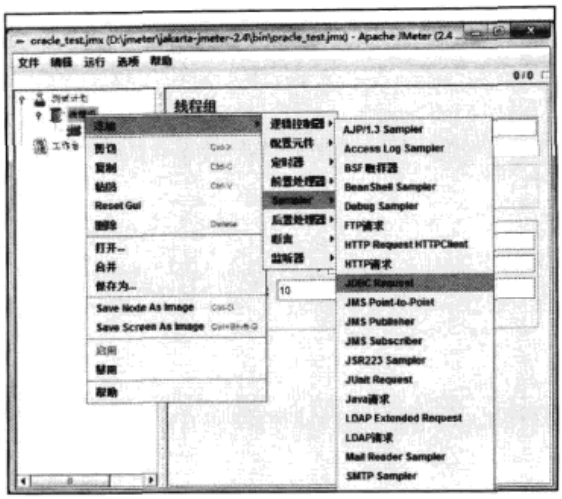


图 4-12　新增 JDBC 请求

本例中的测试 SQL 是 select count(*) from dba_tables，如图 4-13 所示。实际工作中测试人员当然需要更换 SQL。

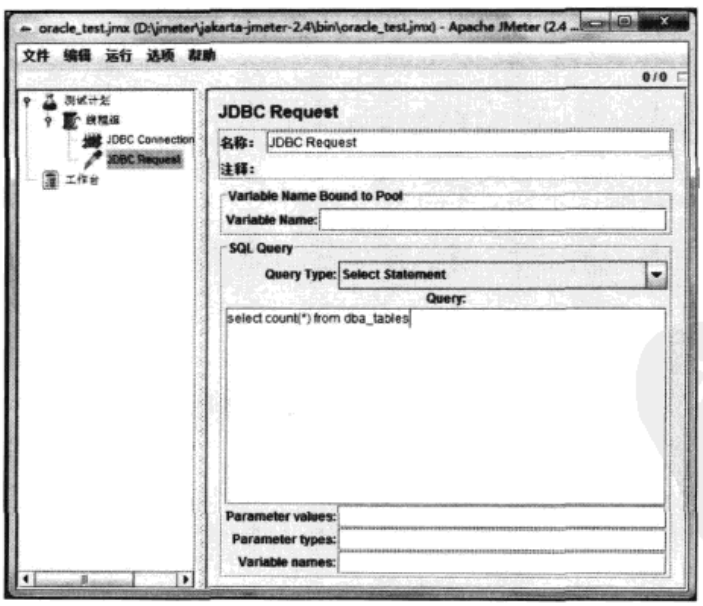


图 4-13　配置 JDBC 请求

（6）添加一个聚合报告，用来展示测试结果，如图 4-14 所示。

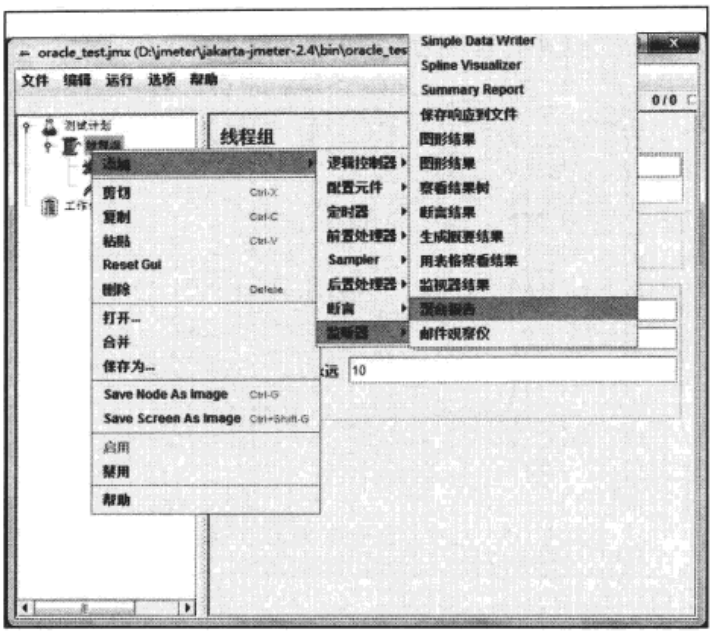


图 4-14　添加聚合报告

（7）保存测试计划，如图 4-15 所示。在运行测试前保存测试计划是一个好习惯。

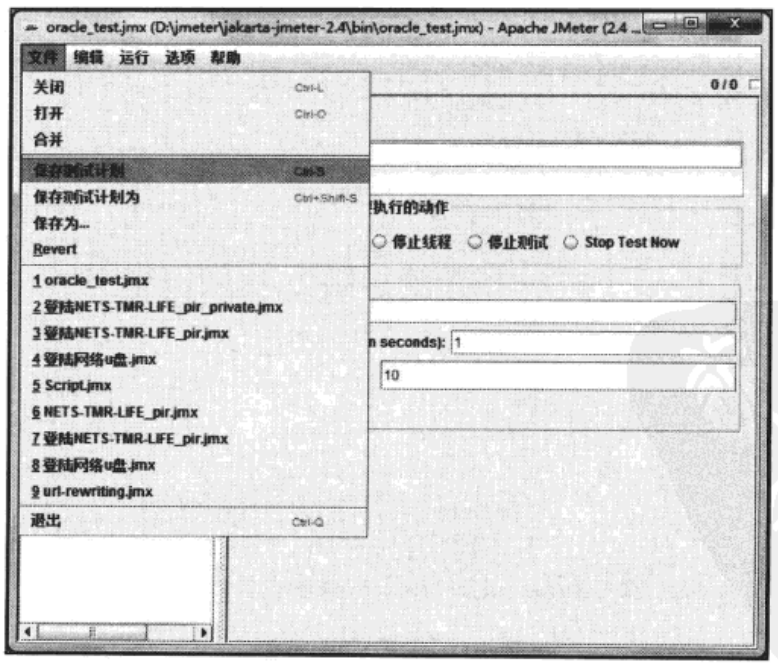


图 4-15　保存测试计划

（8）从“运行”菜单启动测试，如图 4-16 所示。

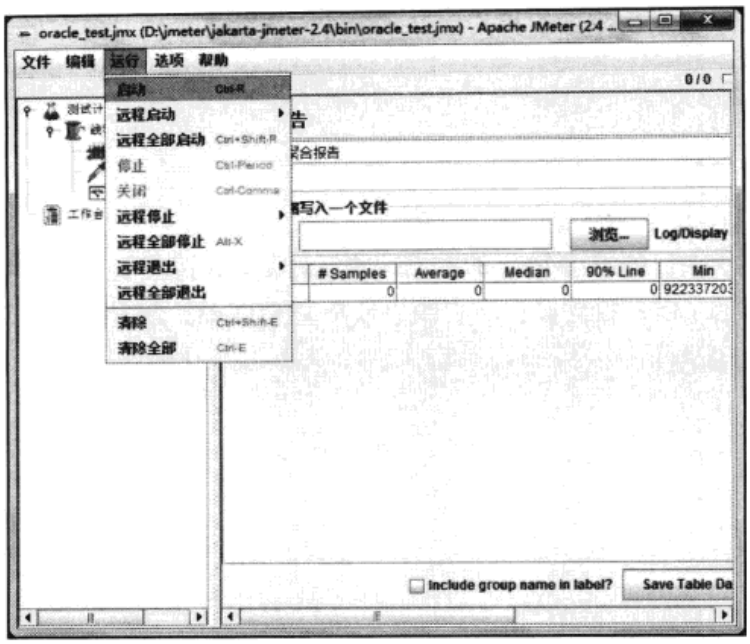


图 4-16　启动测试

（9）通过聚合报告查看测试结果，如图 4-17 所示。

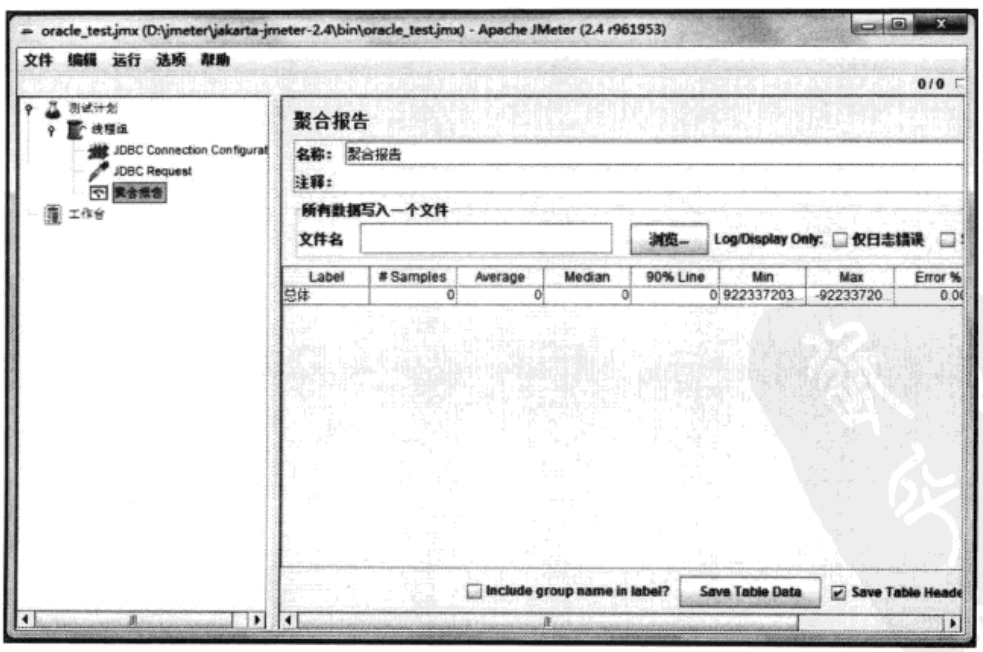


图 4-17　聚合报告

聚合报告中显示了本次测试的相关信息，如访问次数、平均时间、最大时间、最小时间、出错率等。

4.3 本章小结

本章首先介绍了如何创建 JMeter 的数据库测试计划，让读者朋友对如何开发数据库性能测试脚本有一个总体的了解；接着以生产实践中最常见的 Oracle 数据库为例，介绍完整的数据库性能测试流程。

第 5 章

FTP性能测试脚本开发

5.1 FTP 是什么

FTP 是 File Transfer Protocol（文件传输协议）的英文简称，而中文简称为“文传协议”，用于在 Internet 上控制文件的双向传输。同时，它也是一个应用程序（Application）。用户可以通过它把自己的 PC 与世界各地所有运行 FTP 协议的服务器相连，访问服务器上的大量程序和信息。FTP 的主要作用就是让用户连接一个远程计算机(这些计算机上运行着 FTP 服务器程序)，查看远程计算机上有哪些文件，然后把文件从远程计算机上复制到本地计算机，或把本地计算机的文件传送到远程计算机上去。

1. 工作原理

以下载文件为例，当测试人员启动 FTP 从远程计算机复制文件时，实际上启动了两个程序：一个是本地机计算上的 FTP 客户程序，它向 FTP 服务器提出复制文件的请求；另一个是启动在远程计算机的上的 FTP 服务器程序，它响应测试人员的请求，把测试人员指定的文件传送到测试人员的计算机中。FTP 采用“客户机/服务器”方式，用户端要在自己的本地计算机上安装 FTP 客户程序。FTP 客户程序有字符界面和图形界面两种。字符界面的 FTP 的命令复杂、繁多，而图形界面的 FTP 客户程序操作上要简洁方便得多。

2. 传输协议

简单地说，支持 FTP 协议的服务器就是 FTP 服务器，下面介绍一下什么是 FTP 协议（文件传输协议）。

一般来说，用互联网的首要目的就是实现信息共享，文件传输是信息共享非常重要的一个内容。早期在 Internet 上实现文件传输并不是一件容易的事。我们知道 Internet 是一个非常复杂的计算机环境，有 PC，有工作站，有 MAC，有大型机，而连接在 Internet 上的计算机有成千上万台，并且这些计算机可能运行不同的操作系统，有运行 UNIX 的服务器，也有运行 DOS、Windows 的 PC 和运行 Mac OS 的苹果机等，为解决各种操作系统之间的文件交流问题，需要建立一个统一的文件传输协议，这就是所谓的 FTP。基于不同的操作系统有不同的 FTP 应用程序，而所有这些应用程序都遵守同一种协议，这样用户就可以把自己的文件传送给别人，或者从其他的用户环境中获得文件。

3. 服务器系统

与大多数 Internet 服务一样，FTP 也是一个客户机/服务器系统。用户通过一个支持 FTP 协议的客户机程序，连接到在远程主机上的 FTP 服务器程序。用户通过客户机程序向服务器程序发出命令，服务器程序执行用户所发出的命令，并将执行的结果返回到客户机。比如说，用户发出一条命令，要求服务器向用户传送某一个文件的一份备份，服务器会响应这条命令，将指定文件送至用户的机器上。客户机程序代表用户接收到这个文件，将其存放在用户目录中。

在 FTP 的使用中，用户经常遇到两个概念：“下载（Download）”和“上传（Upload）”。“下载”文件就是从远程主机复制文件至自己的计算机上；“上传”文件就是将文件从自己的计算机中复制至远程主机上。用 Internet 语言来说，用户可通过客户机程序向（从）远程主机上传（下载）文件。

使用 FTP 时必须首先登录，在远程主机上获得相应的权限以后，方可下载或上传文件。也就是说，要想同哪一台计算机传送文件，就必须具有哪一台计算机的适当授权。换言之，除非有用户 ID 和口令，否则无法传送文件。这种情况违背了 Internet 的开放性，Internet 上的 FTP 主机何止千万，不可能要求每个用户在每一台主机上都拥有账号。匿名 FTP 就是为解决这个问题而产生的。

匿名 FTP 是这样一种机制，即用户可通过它连接到远程主机上，并下载文件，而无须成为其注册用户。系统管理员建立了一个特殊的用户 ID，名为 anonymous，Internet 上的任何人在任何地方都可使用该用户 ID。

通过 FTP 程序连接匿名 FTP 主机的方式同连接普通 FTP 主机的方式差不多，只是在要求提供用户标识 ID 时，必须输入“anonymous”，该用户 ID 的口令可以是任意的字符串。习惯上，用自己的 E-mail 地址作为口令，使系统维护程序能够记录下来谁在存取这些文件。

值得注意的是，匿名 FTP 不适用于所有 Internet 主机，它只适用于那些提供了这项服务的主机。

当远程主机提供匿名 FTP 服务时，会指定某些目录向公众开放，允许匿名存取，系统中的其余目录则处于隐匿状态。作为一种安全措施，大多数匿名 FTP 主机都允许用户下载文件，而不允许用户上传文件，也就是说，用户可将匿名 FTP 主机上的所有文件复制到自己的机器上，但不能将自己机器上的任何一个文件复制到匿名 FTP 主机上。即使有些匿名 FTP 主机确实允许用户上传文件，用户也只能将文件上传至某一指定目录中。随后，系统管理员会去检查这些文件，将这些文件移至另一个公共下载目录中，供其他用户下载。利用这种方式，远程主机的用户就得到了保护，因为可避免有人上传有问题的文件，如带病毒的文件。

作为一个 Internet 用户，可通过 FTP 在任意两台 Internet 主机之间复制文件。实际上，多数人只有一个 Internet 账户，FTP 主要用于下载公共文件，如共享软件、各公司技术支持文件等。Internet 上有成千上万台匿名 FTP 主机，这些主机上存放着数不清的文件，供用户免费复制。实际上，几乎所有类型的信息、所有类型的计算机程序都可以在 Internet 上找到。这是 Internet 吸引我们的重要原因之一。

4．用户分类

1）Real 账户

这类用户在 FTP 服务上拥有账号。当这类用户登录 FTP 服务器时，其默认的主目录就是以其账号命名的目录。但也可以变更到其他目录中去，如系统的主目录等。

2）Guest 用户

在 FTP 服务器中，我们往往会为不同的部门或者某个特定的用户设置一个账户。这个账户有个特点，就是只能够访问自己的主目录。服务器通过这种方式来保障 FTP 服务器上其他文件的安全。这类账户，在 vsftpd 软件中就叫做 Guest 用户。Guest 用户只能够访问其主目录下的目录，而不得访问主目录以外的文件。

3）Anonymous（匿名）用户

这也是我们通常所说的匿名访问。这类用户在 FTP 服务器中没有指定账户，但仍然可以匿名访问某些公开的资源。

在组建 FTP 服务器的时候，我们就需要根据用户的类型，对用户进行归类。默认情况下，Vsftpd 服务器会把建立的所有账户都归属为 Real 用户。但是，这往往不符合企业安全的需要。因为这类用户不仅可以访问自己的主目录，而且还可以访问其他用户的目录，这就给其他用户所在的空间带来一定的安全隐患。所以，企业要根据实际情况，修改用户所在的类别。

5. 启动方式

需要进行远程文件传输的计算机必须安装和运行 FTP 客户程序。在 Windows 操作系统的安装过程中，通常都安装了 TCP/IP 协议软件，其中就包含了 FTP 客户程序。但是该程序是字符界面而不是图形界面的，就需要以命令提示符的方式进行操作，很不方便。

启动 FTP 客户程序的另一途径是使用 IE 浏览器，用户只需要在 IE 地址栏中输入如下格式的 URL 地址即可：ftp://[用户名:口令@]ftp 服务器域名:[端口号] 。

在 CMD 命令行下也可以用上述方法进行连接，通过 put 命令和 get 命令达到上传和下载的目的，通过 is 命令列出目录。除了上述方法外，还可以在 cmd 下输入“ftp”按回车键，然后输入 open IP 来建立一个连接，此方法还适用于在 Linux 下连接 FTP 服务器。

通过 IE 浏览器启动 FTP 的方法尽管可以使用，但是速度较慢，还会因将密码暴露在 IE 浏览器中而导致不安全，因此，用户一般都需要安装并运行专门的 FTP 客户程序。

（1）在本地计算机上登录到国际互联网。

（2）搜索有文件共享的主机或者个人计算机（一般在专门的 FTP 服务器网站上公布，上面有进入该主机或个人计算机的名称、口令和路径）。

（3）当与远程主机或者对方的个人计算机建立连接后，用对方提供的用户名和口令登录到该主机或对方的个人计算机。

（4）登录远程主机或对方的个人计算机成功后，就可以上传测试人员想跟别人分享的或者下载别人授权共享的信息（这里的信息是指既能放到电脑中又能在显示屏上看到的信息）。

（5）完成工作后关闭 FTP 下载软件，切断连接。

为了实现文件传输，用户还要运行专门的文件传输程序，如网际快车等，另外还有很多专门的 FTP 传输软件，如 FlashFXP 就为其中杰出的软件。有兴趣的朋友还可以试试其他的软件，如 LeapFTP，总之一句话，各有各的特色。

6. 传输细节

TCP/IP 协议中，FTP 标准命令 TCP 端口号为 21，Port 方式数据端口为 20。FTP 的任务是从一台计算机将文件传送到另一台计算机，不受操作系统的限制。

7. 传输模式

FTP 的传输有两种方式：ASCII、二进制。

8. ASCII 传输方式

假定用户正在复制的文件包含简单的 ASCII 码文本，如果在远程计算机上运行的不是 UNIX，当文件传输时，FTP 通常会自动地调整文件的内容，以便把文件解释成另一台计算机存储的文本文件格式。

但是常常有这样的情况：用户正在传输的文件包含的不是文本文件，它们可能是程序、数据库、文字处理文件或者压缩文件。在复制任何非文本文件之前，用 binary 命令告诉 FTP 逐字复制。

9. 二进制传输模式

在二进制传输中，保存文件的位序，以便原始和备份的文件是逐位一一对应的，即使目的计算机上包含位序列的文件是没意义的。例如，macintosh 以二进制方式传送可执行文件到 Windows 系统，在对方系统上，此文件不能执行。

如在 ASCII 方式下传输二进制文件，即使不需要也仍会转译，这会损坏数据（ASCII 方式一般假设每一字符的第一有效位无意义，因为 ASCII 字符组合不使用它。如果传输二进制文件，所有的位都是重要的）。

10. 工作方式

FTP 支持两种模式：Standard （PORT，主动方式）、Passive （PASV，被动方式）。

11. PORT 模式

FTP 客户端首先和服务器的 TCP 21 端口建立连接，用来发送命令，客户端需要接收数据的时候在这个通道上发送 PORT 命令。 PORT 命令包含了客户端用什么端口接收数据。在传送数据的时候，服务器端通过自己的 TCP 20 端口连接至客户端的指定端口发送数据。 FTP Server 必须和客户端建立一个新的连接来传送数据。

12. Passive 模式

建立控制通道和 Standard 模式类似，但建立连接后发送 Pasv 命令。服务器收到 Pasv 命令后，打开一个临时端口（端口号大于 1023、小于 65 535），并且向客户端发送在这个端口上传送数据的请求，客户端连接 FTP 服务器此端口，然后 FTP 服务器通过这个端口传送数据。

很多防火墙在设置的时候是不允许接受外部发起的连接的，所以许多位于防火墙后或内网的 FTP 服务器不支持 PASV 模式，因为客户端无法穿过防火墙打开 FTP 服务器的高端端口；而

许多内网的客户端不能用 PORT 模式登录 FTP 服务器，因为从服务器的 TCP 20 端口无法和内部网络的客户端建立一个新的连接，从而造成了无法工作。

5.2 创建 FTP 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 FTP 站点。这里将会创建 4 个并发用户，用于访问 O'Reilly FTP 站点的两个文件。另外，测试人员需要告诉并发用户执行两遍测试计划。因此，总的请求数目是（4 并发用户） ×（2 请求）×（重复 2 遍） =16 FTP 请求。构建该测试计划，测试人员会用到如下测试组件：线程组（Thread Group）、FTP 请求（FTP Request）、FTP Request Defaults 和 Spline Visualizer。

1. 添加并发用户

首先要做的是添加线程组（Thread Group），并修改其默认配置，如图 5-1 所示。

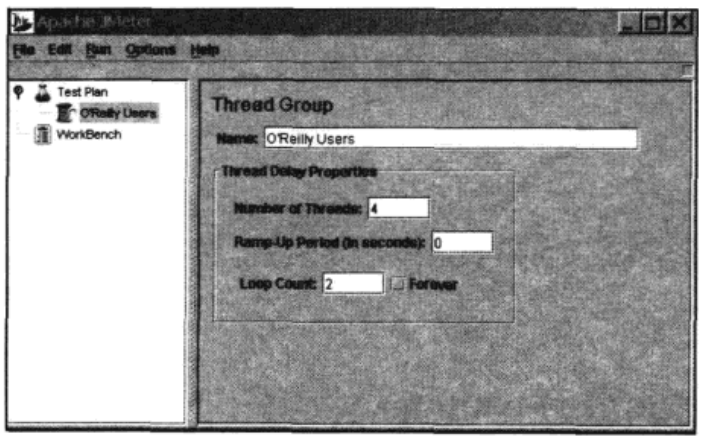


图 5-1 添加线程组 O'Reilly Users

2. 添加默认 FTP 请求属性

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

首先从选中 O'Reilly 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”→“FTP 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 5-2 所示。

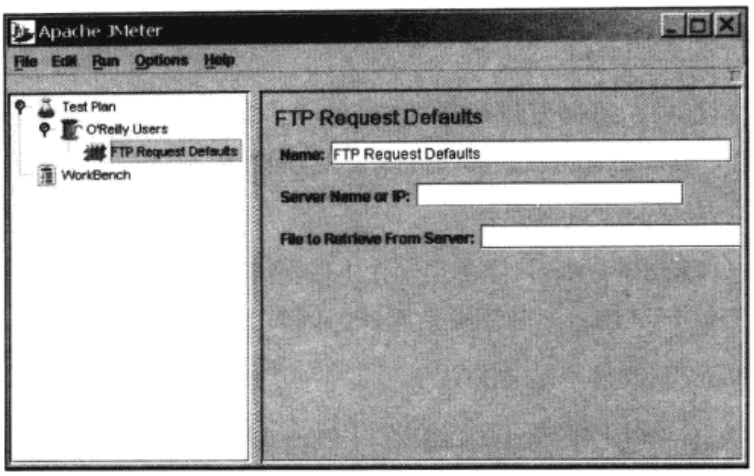


图 5-2　FTP 请求默认值（FTP Request Defaults）（1）

像大多数 JMeter 测试元件一样，FTP 请求默认值（FTP Request Defaults）也有一个名字输入域可供用户修改。在这一例子，对此不做修改，保留默认值。

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 FTP 请求都要发往同一个 FTP 服务器（ftp.oro.com），因此测试人员需要将 ftp.oro.com 放到该设置域中。这个域是我们唯一需要设定的默认值，因此其他域就保留原来的值，如图 5-3 所示。

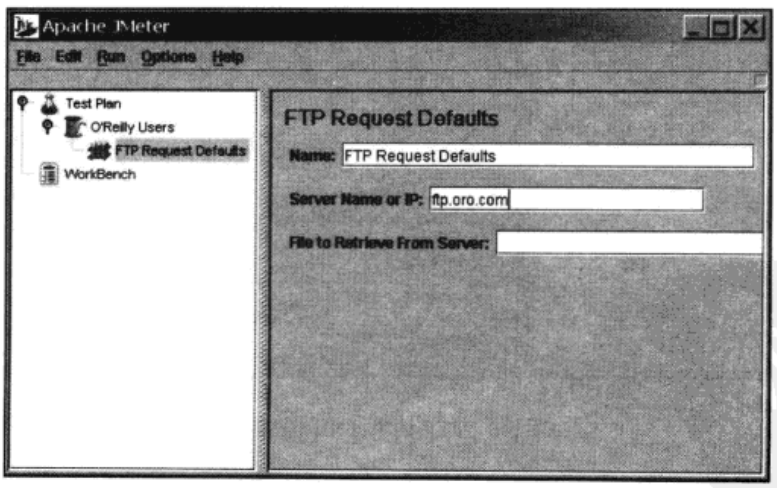


图 5-3　FTP 请求默认值（FTP Request Defaults）（2）

FTP 请求默认值（FTP Request Defaults）不会让 JMeter 发送 FTP 请求，它只是定义 FTP 请求（测试元件）使用的默认值。

3. 添加 FTP 请求

在我们的测试计划中，需要发送两个 FTP 请求。第一个请求针对 O'Reilly mSQL Java README 文件（ftp://ftp.oro.com/pub/msql/java/README），而第二个请求针对 tutorial 文件（ftp://ftp.oro.com/pub/msql/java/tutorial.txt）。

首先为线程组（O'Reilly Users）添加第一个 FTP 请求（Add→Sampler→FTP Request）。接着在测试树中，选中该 FTP 请求，并编辑其属性，如图 5-4 所示。

（1）将名称（Name）改为“README”。

（2）将接收文件在服务器上的路径（File to Retrieve From Server）域设置为“pub/msql/java/README”。

（3）将用户名（Username）设置为“anonymous”。

（4）将密码（Password）设置为“anonymous”。

小贴士

此处测试人员不用设置 Server Name or IP 输入域，因为测试人员在 FTP 请求默认值（FTP Request Defaults）中已经设置过了。

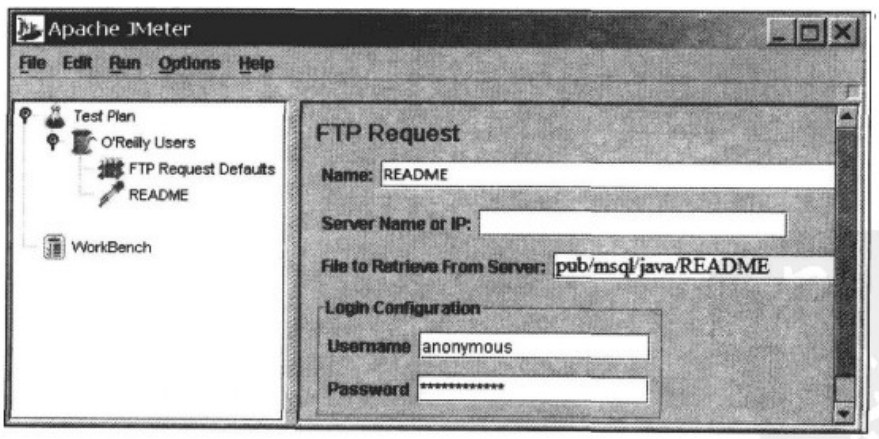


图 5-4 FTP 请求（README）

接着，添加第二个 FTP 请求，并编辑如下属性，如图 5-5 所示。

（1）将名称（Name）改为“tutorial”。

（2）将接收文件在服务器上的路径（File to Retrieve From Server）域设置为“pub/msql/java/tutorial.txt”。

（3）将用户名（Username）设置为“anonymous”。

（4）将密码（Password）设置为“anonymous”。

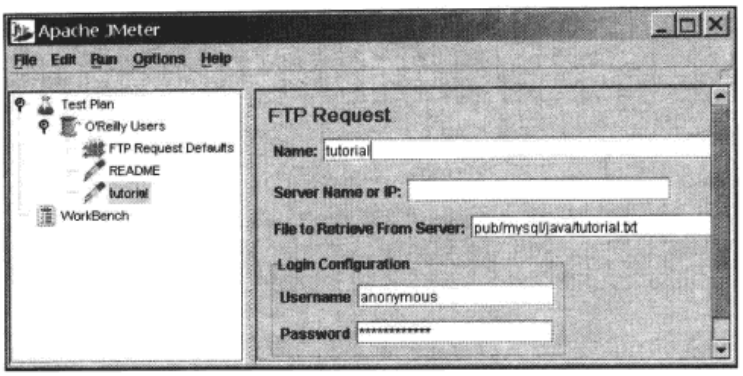


图 5-5　FTP 请求（tutorial）

4．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。该测试元件负责将所有 FTP 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（O'Reilly Users），并添加一个 Spline Visualizer 监听器（Add→Listener→Spline Visualizer），如图 5-6 所示。

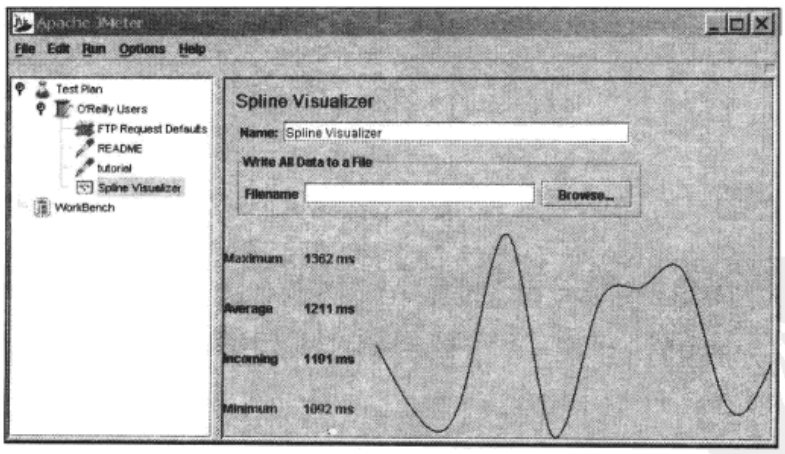


图 5-6　Spline Visualizer 监听器

在最新版本的 JMeter（2.4）中，FTP 请求默认值和 FTP 请求两种测试元件的控制面板有变化，添加了更多的选项。但是 FTP 测试计划的创建流程和测试的基本方法并没有太大变化。在这里作者仅提供最新的控制面板截图，供读者朋友们了解，如图 5-7 和图 5-8 所示。

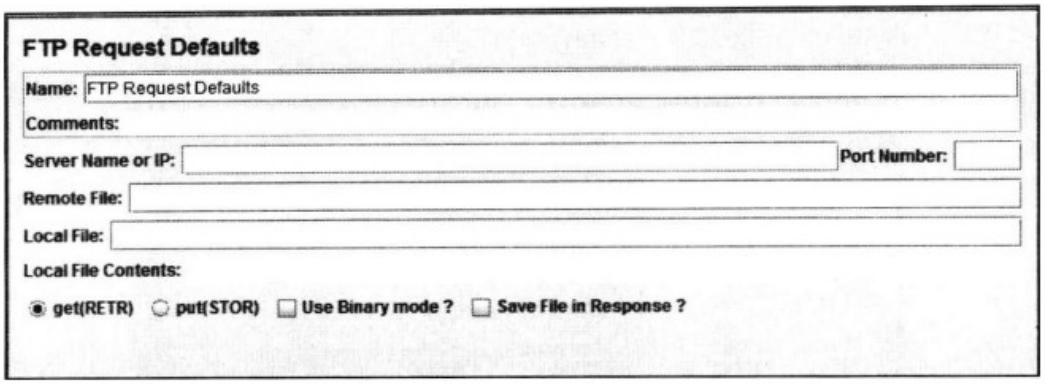


图 5-7 FTP 请求默认值（JMeter 2.4）

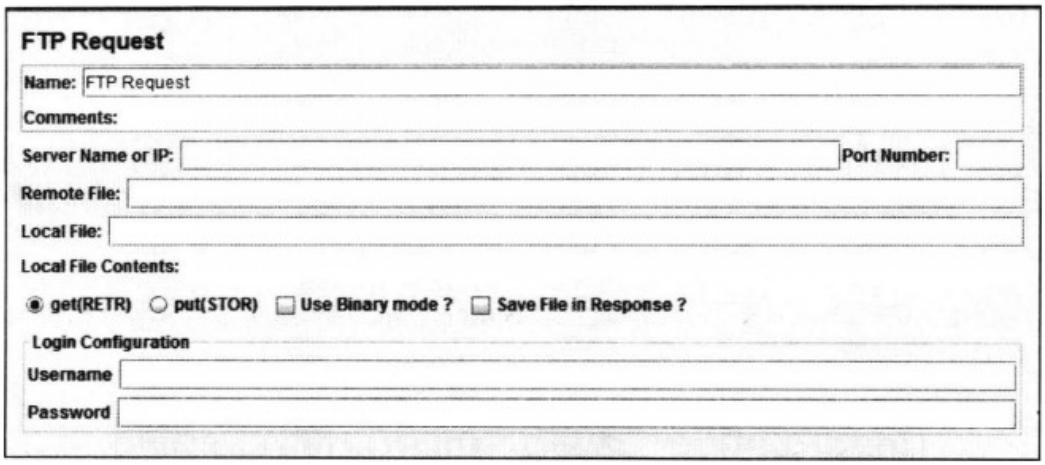


图 5-8 FTP 请求（JMeter 2.4）

5.3 本章小结

本章首先介绍了 FTP 的基本概念，接着介绍了如何创建 JMeter 的 FTP 测试计划。通过阅读本章，读者朋友们可以轻松掌握对 FTP 站点进行性能测试的方法。

第 6 章

LDAP性能测试脚本开发

6.1 LDAP 是什么

本节将主要介绍 LDAP 的基本概念、适用范围及 LDAP 相对于关系数据库的优势和劣势。另外还将介绍一些 LDAP 的简单实例，帮助读者建立对 LDAP 的感性认识，为后续掌握 LDAP 性能测试的方法打下基础。

LDAP 是轻量目录访问协议，英文全称是 Lightweight Directory Access Protocol，一般都简称为 LDAP。它是基于 X.500 标准的，但更简单并且可以根据需要定制。与 X.500 不同，LDAP 支持 TCP/IP 协议，这对访问 Internet 来说是必需的。LDAP 的核心规范在 RFC 中都有定义，所有与 LDAP 相关的 RFC 都可以在 LDAPman RFC 网页中找到。一个简单的 LDAP 组件配置案例，如图 6-1 所示。

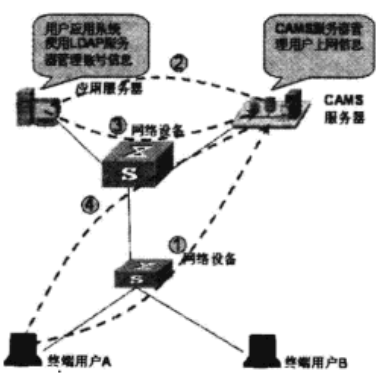


图 6-1 LDAP 组件配置案例

1. LDAP 简介

简单说来，LDAP 是一个用于得到关于人或者资源的集中、静态数据的快速方式。

LDAP 是一个用来发布目录信息到许多不同资源的协议。通常它作为一个集中的地址本使用，不过根据组织者的需要，可以做得更加强大。

2. LDAP 的实质

LDAP 其实是一个电话簿，类似于我们所使用的诸如 NIS（Network Information Service）、DNS（Domain Name Service）等的网络目录，也类似于测试人员在花园中所看到的树木。

不少 LDAP 开发人员喜欢把 LDAP 与关系数据库相比，认为是另一种存储方式，然后在读性能上进行比较。实际上，这种对比的基础是错误的。LDAP 和关系数据库是两种不同层次的概念，后者是存储方式（同一层次如网格数据库、对象数据库），前者是存储模式和访问协议。LDAP 是一个比关系数据库抽象层次更高的存储概念，与关系数据库的查询语言 SQL 属同一级别。LDAP 最基本的形式是一个连接数据库的标准方式。该数据库为读查询做了优化，因此它可以很快地得到查询结果，不过在其他方面，例如更新，就慢得多了。

3. 特殊的数据库

从另一个意义上，LDAP 是实现了指定的数据结构的存储，它是一种特殊的数据库。但是 LDAP 和一般的数据库不同，明白这一点是很重要的。LDAP 对查询进行了优化，与写性能相比 LDAP 的读性能要优秀很多。

就像 Sybase、Oracle、Informix 或 Microsoft 的数据库管理系统（DBMS）是用于处理查询和更新关系型数据库那样，LDAP 服务器也是用来处理查询和更新 LDAP 目录的。换句话来说，LDAP 目录也是一种类型的数据库，但不是关系型数据库。要特别注意的是，LDAP 通常作为一个分级数据库使用，而不是一个关系数据库，因此它的结构用树来表示比用表格好。正因为这样，就不能用 SQL 语句了。

现在 LDAP 技术发展得很快，在企业范围内实现 LDAP 可以让运行在几乎所有计算机平台上的所有应用程序从 LDAP 目录中获取信息。LDAP 目录中可以存储各种类型的数据：电子邮件地址、邮件路由信息、人力资源数据、公用密匙、联系人列表，等等。通过把 LDAP 目录作为系统集成中的一个重要环节，可以简化员工在企业内部查询信息的步骤，甚至连主要的数据源都可以放在任何地方。

4．LDAP 目录的优势

如果需要开发一种提供公共信息查询的系统，一般的设计方法可能是采用基于 Web 的数据库设计方式，即前端使用浏览器而后端使用 Web 服务器加上关系数据库。后端在 Windows 的典型实现可能是 Windows NT + IIS + Access 数据库或者是 SQL Server，IIS 和数据库之间通过 ASP 技术使用 ODBC 进行连接，达到通过填写表单查询数据的功能。

后端在 Linux 系统的典型实现可能是 Linux+ Apache + PostgreSQL，Apache 和数据库之间通过 PHP3 提供的函数进行连接。使用上述方法的缺点是后端关系数据库的引入导致系统整体的性能降低和系统的管理比较烦琐，因为需要不断地进行数据类型的验证和事务完整性的确认；并且前端用户对数据的控制不够灵活，用户权限的设置一般只能是设置在表一级而不是设置在记录一级。

目录服务的推出主要是解决上述数据库中存在的问题。目录与关系数据库相似，是指具有描述性的基于属性的记录集合，但它的数据类型主要是字符型，为了检索的需要添加了 BIN（二进制数据）、CIS（忽略大小写）、CES（大小写敏感）、TEL（电话型）等语法（Syntax），而不是关系数据库提供的整数、浮点数、日期、货币等类型。同样也不提供像关系数据库中普遍包含的大量的函数，它主要面向数据的查询服务（查询和修改操作比一般是大于 10:1），不提供事务的回滚（Rollback）机制。它的数据修改使用简单的锁定机制实现 All-or-Nothing，它的目标是快速响应和大容量查询并且提供多目录服务器的信息复制功能。

现在来介绍一下 LDAP 目录到底有些什么优势。现在 LDAP 的流行是很多因素共同作用的结果。可能 LDAP 最大的优势是：可以在任何计算机平台上，用很容易获得的而且数目不断增加的 LDAP 的客户端程序访问 LDAP 目录，而且也很容易定制应用程序为它加上 LDAP 的支持。

5．LDAP 的协议

LDAP 协议是跨平台和标准协议，因此应用程序就不用为 LDAP 目录放在什么样的服务器上操心了。实际上，LDAP 得到了业界的广泛认可，因为它是 Internet 的标准。厂商都很愿意在产品中加入对 LDAP 的支持，因为他们根本不用考虑另一端（客户端或服务端）是怎么样的。LDAP 服务器可以是任何一个开放源代码或商用的 LDAP 目录服务器（或者还可能是具有 LDAP 界面的关系型数据库），因为可以用同样的协议、客户端连接软件包和查询命令与 LDAP 服务器进行交互。与 LDAP 不同的是，如果软件厂商想在软件产品中集成对 DBMS 的支持，那么通常都要对每一个数据库服务器单独定制。不像很多商用的关系型数据库，测试人员不必为 LDAP 的每一个客户端连接或许可协议付费。大多数的 LDAP 服务器安装起来很简单，也容易维护和优化。

6. LDAP 的服务器

LDAP 服务器可以用“推”或“拉”的方法复制部分或全部数据，例如，可以把数据“推”到远程的办公室，以增加数据的安全性。复制技术是内置在 LDAP 服务器中的，而且很容易配置。如果要在 DBMS 中使用相同的复制功能，数据库厂商就会要求测试人员支付额外的费用，而且也很难管理。

7. LDAP 的使用权限

LDAP 允许测试人员根据需要使用 ACI（一般都称为 ACL 或者访问控制列表）控制对数据读和写的权限。例如，设备管理员可以有权改变员工的工作地点和办公室号码，但是不允许改变记录中其他的域。ACI 可以根据谁访问数据、访问什么数据、数据存在什么地方及其他对数据进行访问控制。因为这些都是由 LDAP 目录服务器完成的，所以不用担心在客户端的应用程序上是否要进行安全检查。

LDAP（Lightweight Directory Acess Protocol）是目录服务在 TCP/IP 上的实现（RFC 1777 V2 版和 RFC 2251 V3 版）。它是对 X.500 的目录协议的移植，但是简化了实现方法，所以称为轻量级的目录服务。在 LDAP 中目录是按照树形结构组织，目录由条目（Entry）组成，条目相当于关系数据库中表的记录；条目是具有区别名 DN（Distinguished Name）的属性（Attribute）集合，DN 相当于关系数据库表中的关键字（Primary Key）；属性由类型（Type）和多个值（Values）组成，相当于关系数据库中的域（Field）。由域名和数据类型组成，只是为了方便检索的需要，LDAP 中的 Type 可以有多个 Value，而不是关系数据库中为降低数据的冗余性要求实现的各个域必须是不相关的。LDAP 中条目一般按照地理位置和组织关系进行组织，非常直观。LDAP 把数据存放在文件中，为提高效率可以使用基于索引的文件数据库，而不是关系数据库。LDAP 协议集还规定了 DN 的命名方法、存取控制方法、搜索格式、复制方法、URL 格式、开发接口等。

LDAP 对于存储这样的信息最为有用，也就是说数据需要从不同的地点读取，但是不需要经常更新。

例如，以下这些信息存储在 LDAP 目录中是十分有效的：

- 公司员工的电话号码簿和组织结构图。
- 客户的联系信息。
- 计算机管理需要的信息，包括 NIS 映射、E-mail 假名等。
- 软件包的配置信息。

- 公用证书和安全密匙。
- 什么时候该用 LDAP 存储数据。

大多数的 LDAP 服务器都为读密集型的操作进行专门的优化。因此，当从 LDAP 服务器中读取数据的时候会比从专门为 OLTP 优化的关系型数据库中读取数据快一个数量级。也正是因为专门为读的性能进行优化，大多数的 LDAP 目录服务器并不适合存储需要经常改变的数据。例如，用 LDAP 服务器来存储电话号码是一个很好的选择，但是它不能作为电子商务站点的数据库服务器。

如果下面每一个问题的答案都是“是”，那么把数据存在 LDAP 中就是一个好主意。

- 需要在任何平台上都能读取数据吗？
- 每一个单独的记录项是不是每一天都只有很少的改变？
- 可以把数据存在平面数据库（Flat Database）而不是关系型数据库中吗？

最后一个问题可能会唬住一些人，其实用平面数据库去存储一些关系型的数据也是很一般的。例如，一条公司员工的记录就可以包含经理的登录名，用 LDAP 来存储这类信息是很方便的。一个简单的判断方法：如果可以把数据存在一张张的卡片中，就可以很容易地把它存在 LDAP 目录中。

8. 安全和访问控制

LDAP 提供很复杂的不同层次的访问控制或者 ACI。因这些访问可以在服务器端控制，这比用客户端的软件保证数据的安全可靠多了。

用 LDAP 的 ACI，可以完成：

- 给予用户改变他们自己的电话号码和家庭地址的权限，但是限制他们对其他数据（如职务名称、经理的登录名等）只有“只读”权限。
- 给予“HR-admins”组中的所有人权限以改变下面这些用户的信息：经理、工作名称、员工号、部门名称和部门号，但是对其他域没有写权限。
- 禁止任何人查询 LDAP 服务器上的用户口令，但是可以允许用户改变他或她自己的口令。
- 给予经理访问他们上级的家庭电话的“只读”权限，但是禁止其他人有这个权限。
- 给予“host-admins”组中的任何人创建、删除和编辑所有保存在 LDAP 服务器中的与计算机主机有关的信息。

- 通过 Web，允许“foobar-sales”组中的成员有选择地给予或禁止他们自己读取一部分客户联系数据的“读”权限。这将允许他们把客户联系信息下载到本地的笔记本电脑或个人数字助理（PDA）上（如果销售人员的软件都支持 LDAP，这将非常有用）。
- 通过 Web，允许组的所有者删除或添加他们拥有的组的成员。例如，可以允许销售经理给予或禁止销售人员改变 Web 页的权限。也可以允许邮件假名（Mail Aliase）的所有者不经过 IT 技术人员就直接从邮件假名中删除或添加用户。“公用”的邮件列表应该允许用户从邮件假名中添加或删除自己（但是只能是自己）。也可以对 IP 地址或主机名加以限制。例如，某些域只允许用户 IP 地址以 192.168.200.*开头的有读的权限，或者用户反向查找 DNS 得到的主机名必须为*.foobar.com。

9. LDAP 目录树的结构

LDAP 目录以树状的层次结构来存储数据。如果测试人员对自顶向下的 DNS 树或 UNIX 文件的目录树比较熟悉，也就很容易掌握 LDAP 目录树这个概念了。就像 DNS 的主机名那样，LDAP 目录记录的标识名（Distinguished Name，DN）用来读取单个记录，以及回溯到树的顶部。后面会做详细的介绍。

为什么要用层次结构来组织数据呢？原因是多方面的，下面是可能遇到的一些情况：

- 如果测试人员想把所有的美国客户的联系信息都“推”位于西雅图办公室（负责营销）的 LDAP 服务器上，但是测试人员不想把公司的资产管理信息“推”到那里。
- 测试人员可能想根据目录树的结构给予不同的员工组不同的权限。在下面的例子里，资产管理组对“asset-mgmt”部分有完全的访问权限，但是不能访问其他地方。
- 把 LDAP 存储和复制功能结合起来，可以定制目录树的结构以降低对 WAN 带宽的要求。位于西雅图的营销办公室需要每分钟更新美国销售状况的信息，但是欧洲的销售情况就只要每小时更新一次就行了。

10. 刨根问底：基准 DN

LDAP 目录树的最顶部就是根，也就是所谓的“基准 DN”。基准 DN 通常使用下面列出的 3 种格式之一。假定我在名为 FooBar 的电子商务公司工作，这家公司在 Internet 上的名字是 foobar。

```
o="FooBar, Inc.", c=US （以 X.500 格式表示的基准 DN）
```

在这个例子中，o=FooBar, Inc.表示组织名，在这里就是公司名的同义词。c=US 表示公司的总部在美国。以前，一般都用这种方式来表示基准 DN。但是事物总是在不断变化的，现在

所有的公司都已经（或计划）上 Internet。随着 Internet 的全球化，在基准 DN 中使用国家代码很容易让人产生混淆。现在，X.500 格式发展成下面列出的两种格式：

- o=foobar . com （用公司的 Internet 地址表示的基准 DN）。这种格式很直观，用公司的域名作为基准 DN。这也是现在最常用的格式。
- dc=foobar, dc=com （用 DNS 域名的不同部分组成的基准 DN）。

就像上面那一种格式，这种格式也是以 DNS 域名为基础的，但是上面那种格式不改变域名（也就更易读），而这种格式把域名：foobar.com 分成两部分 dc=foobar, dc=com。在理论上，这种格式可能会更灵活一点，但是对于最终用户来说也更难记忆一点。考虑一下 foobar.com 这个例子。当 foobar.com 和 gizmo.com 合并之后，可以简单地把“dc=com”当做基准 DN。把新的记录放到已经存在的 dc=gizmo,dc=com 目录下，这样就简化了很多工作（当然，如果 foobar.com 和 wocket 点 edu 合并，这个方法就不能用了）。如果 LDAP 服务器是新安装的，建议测试人员使用这种格式。注意，如果测试人员打算使用活动目录（Actrive Directory），Microsoft 已经限制测试人员必须使用这种格式。

11．在目录树中怎么组织数据

在 UNIX 文件系统中，最顶层是根目录（root），在根目录的下面有很多的文件和目录。像上面介绍的那样，LDAP 目录也是用同样的方法组织起来的。

在根目录下，要把数据从逻辑上区分开。因为历史上（X.500）的原因，大多数 LDAP 目录用 OU 从逻辑上把数据分开来。OU 表示“Organization Unit”，在 X.500 协议中用来表示公司内部的机构，如销售部、财务部等。现在 LDAP 还保留 ou=这样的命名规则，但是扩展了分类的范围，可以分类为：ou=people、ou=groups、ou=devices，等等。更低一级的 OU 有时用来做更细的归类。例如，LDAP 目录树（不包括单独的记录）可能会是这样的：

```
dc=foobar, dc=com
    ou=customers
    ou=asia
    ou=europe
    ou=usa
    ou=employees
    ou=rooms
    ou=groups
    ou=assets-mgmt
    ou=nisgroups
ou=recipes
```

12. 单独的 LDAP 记录

DN 是 LDAP 记录项的名字。在 LDAP 目录中的所有记录项都有一个唯一的“Distinguished Name”，也就是 DN。每一个 LDAP 记录项的 DN 是由两个部分组成的：相对 DN（RDN）和记录在 LDAP 目录中的位置。

RDN 是 DN 中与目录树的结构无关的部分。在 LDAP 目录中存储的记录项都要有一个名字，这个名字通常存在 cn（Common Name）这个属性中。因为几乎所有的东西都有一个名字，在 LDAP 中存储的对象都用它们的 cn 值作为 RDN 的基础。如果我把燕麦粥食谱存为一个记录，我就会用 cn=Oatmeal Deluxe 作为记录项的 RDN。

- 我的 LDAP 目录的基准 DN 是 dc=foobar,dc=com。
- 我把自己的食谱作为 LDAP 的记录项存在 ou=recipes。
- 我的 LDAP 记录项的 RDN 设为 cn=Oatmeal Deluxe。

上面这些构成了燕麦粥食谱的 LDAP 记录的完整 DN。记住，DN 的读法和 DNS 主机名类似。下面就是完整的 DN：

```
cn=Oatmeal Deluxe,ou=recipes,dc=foobar,dc=com
```

下面举一个实际的例子来说明 DN。

现在为公司的员工设置一个 DN，可以用基于 cn 或 uid（User ID）作为典型的用户账号。例如，Foobar 的员工 Fran Smith（登录名：fsmith）的 DN 可以为以下两种格式：

1）基于登录名

```
uid=fsmith,ou=employees,dc=foobar,dc=com
```

LDAP（以及 X.500）用 uid 表示“User ID”，不要把它和 UNIX 的 uid 号混淆了。大多数公司都会给每一个员工唯一的登录名，因此用这个办法可以很好地保存员工的信息。测试人员不用担心以后还会有一个叫 Fran Smith 的人加入公司，如果 Fran 改变了她的名字（结婚？离婚？或宗教原因？），也用不着改变 LDAP 记录项的 DN。

2）基于姓名

```
cn=Fran Smith,ou=employees,dc=foobar,dc=com
```

可以看到这种格式使用了 Common Name（CN）。可以把 Common Name 当成一个人的全名。这种格式有一个很明显的缺点，就是如果名字改变了，LDAP 的记录就要从一个 DN 转移到另一个 DN。但是，我们应该尽可能地避免改变一个记录项的 DN。

13. 定制目录的对象类型

测试人员可以用 LDAP 存储各种类型的数据对象，只要这些对象可以用属性来表示。下面是可以在 LDAP 中存储的一些信息。

- 员工信息：员工的姓名、登录名、口令、员工号、他的经理的登录名、邮件服务器，等等。
- 物品跟踪信息：计算机名、IP 地址、标签、型号、所在位置，等等。
- 客户联系列表：客户的公司名，主要联系人的电话、传真和电子邮件，等等。
- 会议厅信息：会议厅的名字、位置、可以坐多少人、电话号码、是否有投影机。
- 食谱信息：菜的名字、配料、烹调方法及准备方法。

因为 LDAP 目录可以定制成存储任何文本或二进制数据，到底存什么要由测试人员自己决定。LDAP 目录用对象类型（Object Classes）的概念来定义运行哪一类的对象使用什么属性。在几乎所有的 LDAP 服务器中，测试人员都要根据自己的需要扩展基本的 LDAP 目录的功能，创建新的对象类型或者扩展现存的对象类型。

LDAP 目录以一系列“属性对”的形式来存储记录项，每一个记录项包括属性类型和属性值（这与关系型数据库用行和列来存取数据有根本的不同）。下面是笔者存在 LDAP 目录中的一部分食谱记录：

```
dn: cn=Oatmeal Deluxe, ou=recipes, dc=foobar, dc=com
cn: Instant Oatmeal Deluxe
recipeCuisine: breakfast
recipeIngredient: 1 packet instant oatmeal
recipeIngredient: 1 cup water
recipeIngredient: 1 pinch salt
recipeIngredient: 1 tsp brown sugar
recipeIngredient: 1/4 apple, any type
```

请注意上面每一种配料都作为属性 recipeIngredient 值。LDAP 目录被设计成像上面那样为一个属性保存多个值，而不是在每一个属性的后面用逗号把一系列值分开。

因为用这样的方式存储数据，所以数据库就有很大的灵活性，不必为加入一些新的数据就重新创建表和索引。更重要的是，LDAP 目录不必花费内存或硬盘空间处理“空”域，也就是说，实际上不使用可选择的域也不会花费测试人员任何资源。

14. 示例

作为例子的一个单独的数据项，让我们看看下面这个例子。我们用 Foobar, Inc.的员工 Fran

Smith 的 LDAP 记录。这个记录项的格式是 LDIF，用来导入和导出 LDAP 目录的记录项。

```
dn: uid=fsmith, ou=employees, dc=foobar, dc=com
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
objectclass: foobarPerson
uid: fsmith
givenname: Fran
sn: Smith
cn: Fran Smith
cn: Frances Smith
telephonenumber: 510-555-1234
roomnumber: 122G
o: Foobar, Inc.
mailRoutingAddress: fsmith@foobar 点 com
mailhost: mail.foobar 点 com
userpassword: {crypt}3x1231v76T89N
uidnumber: 1234
gidnumber: 1200
homedirectory: /home/fsmith
loginshell: /usr/local/bin/bash
```

属性的值在保存的时候是保留大小写的，但是在默认情况下搜索的时候是不区分大小写的。某些特殊的属性（如 password）在搜索的时候需要区分大小写。

让我们一点一点地分析上面的记录项。

```
dn: uid=fsmith, ou=employees, dc=foobar, dc=com
```

这是 Fran 的 LDAP 记录项的完整 DN，包括在目录树中的完整路径。LDAP（和 X.500）使用 uid（User ID），不要把它和 UNIX 的 uid 号混淆了。

```
objectclass: person
objectclass: organizationalPerson
objectclass: inetOrgPerson
objectclass: foobarPerson
```

可以为任何一个对象根据需要分配多个对象类型。person 对象类型要求 cn（common name）和 sn（surname）这两个域不能为空。persion 对象类型允许有其他的可选域，包括 givenname、telephonenumber 等。organizational Person 给 person 加入更多的可选域，inetOrgPerson 又加入更多的可选域（包括电子邮件信息）。最后，foobarPerson 是为 Foobar 定制的对象类型，加入了很多定制的属性。

```
uid: fsmith
givenname: Fran
sn: Smith
cn: Fran Smith
cn: Frances Smith
telephonenumber: 510-555-1234
roomnumber: 122G
o: Foobar, Inc.
```

以前说过了，uid 表示 User ID。当看到 uid 的时候，就想一想“login”。

请注意，CN 有多个值。就像上面介绍的，LDAP 允许某些属性有多个值。为什么允许有多个值呢？假定测试人员在用公司的 LDAP 服务器查找 Fran 的电话号码，测试人员可能只知道她的名字叫 Fran，但是对人力资源处的人来说她的正式名字叫做 Frances。因为保存了她的两个名字，所以用任何一个名字检索都可以找到 Fran 的电话号码、电子邮件和办公房间号等。

```
mailRoutingAddress: fsmith@foobar 点 com
mailhost: mail.foobar 点 com
```

就像现在大多数的公司都可以上网，Foobar 用 Sendmail 发送邮件和处理外部邮件路由信息。Foobar 把所有用户的邮件信息都存在 LDAP 中。最新版本的 Sendmail 支持这项功能。

```
Userpassword: {crypt}3x1231v76T89N
uidnumber: 1234
gidnumber: 1200
gecos: Frances Smith
homedirectory: /home/fsmith
loginshell: /usr/local/bin/bash
```

注意，Foobar 的系统管理员把所有用户的口令映射信息也都存在 LDAP 中。FoobarPerson 类型的对象具有这种能力。再注意一下，用户口令是用 UNIX 的口令加密格式存储的。UNIX 的 uid 在这里为 uidnumber。提醒测试人员一下，关于如何在 LDAP 中保存 NIS 信息，有完整的一份 RFC，在以后的文章中会谈一谈 NIS 的集成。

15. LDAP 复制

LDAP 服务器可以使用基于“推”或者“拉”的技术，用简单或基于安全证书的安全验证，复制一部分或者所有的数据。

例如，Foobar 有一个“公用的”LDAP 服务器，地址为 ldap.foobar.com，端口为 389。Netscape Communicator 的电子邮件查询功能、UNIX 的“ph”命令要用到这个服务器，用户也可以在任何地方查询这个服务器上的员工和客户联系信息。公司的主 LDAP 服务器运行在相同的计算机

上，不过端口号是 1389。

测试人员可能既不想让员工查询资产管理或食谱的信息，又不想让信息技术人员看到整个公司的 LDAP 目录。为了解决这个问题，Foobar 有选择地把子目录树从主 LDAP 服务器复制到“公用”LDAP 服务器上，不复制需要隐藏的信息。为了保持数据始终是最新的，主目录服务器被设置成即时“推”同步。这种方法主要是为了方便，而不是安全，因为如果有权限的用户想查询所有的数据，可以用另一个 LDAP 端口。

假定 Foobar 通过从奥克兰到欧洲的低带宽数据的连接用 LDAP 管理客户联系信息，可以建立从 ldap.foobar.com:1389 到 munich-ldap.foobar.com:389 的数据复制，像下面这样：

```
periodic pull: ou=asia,ou=customers,o=sendmail.com
periodic pull: ou=us,ou=customers,o=sendmail.com
immediate push: ou=europe,ou=customers,o=sendmail.com
```

“拉”连接每 15 分钟同步一次，在上面假定的情况下足够了。“推”连接保证任何欧洲的联系信息发生了变化就立即被“推”到 Munich。

用上面的复制模式，用户为了访问数据需要连接到哪一台服务器呢？在 Munich 的用户可以简单地连接到本地服务器。如果他们改变了数据，本地的 LDAP 服务器就会把这些变化传到主 LDAP 服务器。然后，主 LDAP 服务器把这些变化“推”回本地的“公用”LDAP 服务器以保持数据的同步。这对本地的用户有很大的好处，因为所有的查询（大多数是读）都在本地的服务器上进行，速度非常快。当需要改变信息的时候，最终用户不需要重新配置客户端的软件，因为 LDAP 目录服务器为他们完成了所有的数据交换工作。

6.2 创建 LDAP 测试计划

在这一节中，作者将会介绍如何创建一个简单的测试计划，用于测试 LDAP 服务器。我们将会创建 4 个并发用户，并向 LDAP 服务器发送 4 个请求。另外，测试人员需要告诉并发用户执行 4 遍测试计划。因此，总的请求数目是（4 并发用户）×（4 请求）×（重复 4 遍）= 64 LDAP 请求。构建该测试计划，测试人员会用到如下测试组件：线程组（Thread Group ）、LDAP 请求（LDAP Request）、LDAP 请求默认值（LDAP Request Defaults） 和用表格查看结果（View Results in Table）。

下面的例子中，作者假设 LDAP 服务器安装在测试人员的本地计算机上。

1. 添加并发用户

首先要做的还是添加线程组（Thread Group），并修改其默认配置，如图 6-2 所示。

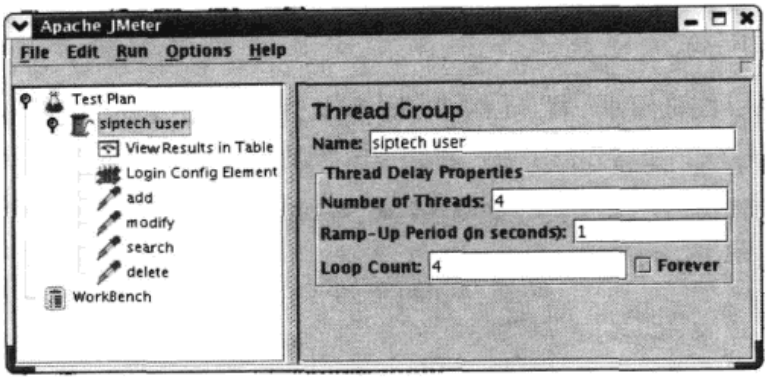


图 6-2　线程组（Thread Group）——siptech users

2．添加登录配置元件

首先从选中 siptech 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”→“Login Config Element”命令。接着选中这个新测试元件，查看它的控制面板，如图 6-3 所示。

像大多数 JMeter 测试元件一样，登录配置元件（Login Config Element）也有一个名字输入域可供用户修改。在这一例子中，对此不做修改，保留默认值。

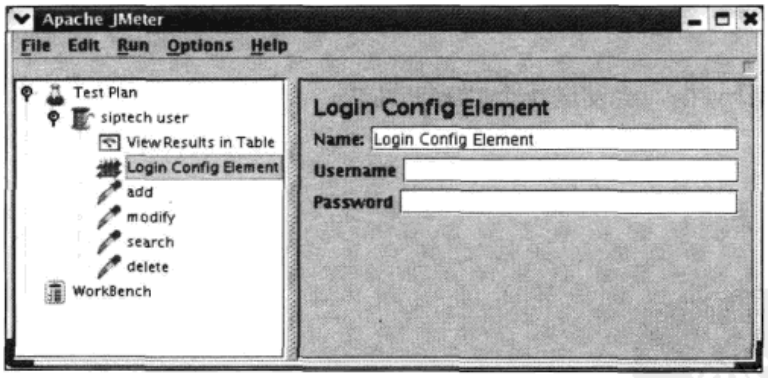


图 6-3　登录配置元件（Login Config Element）

将 Username 域设置为测试人员所测试服务器的用户名，并将 password 域设置为测试人员所测试服务器的密码。如此一来，这些值将作为 LDAP 请求的默认参数。

3．添加 LDAP 请求默认值

从选中 siptech user（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”→“LDAP 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 6-4 所示。

像大多数 JMeter 测试元件一样，LDAP 请求默认值（LDAP Request Defaults）也有一个名字输入域可供用户修改。在这一例子中，对此不做修改，保留默认值。

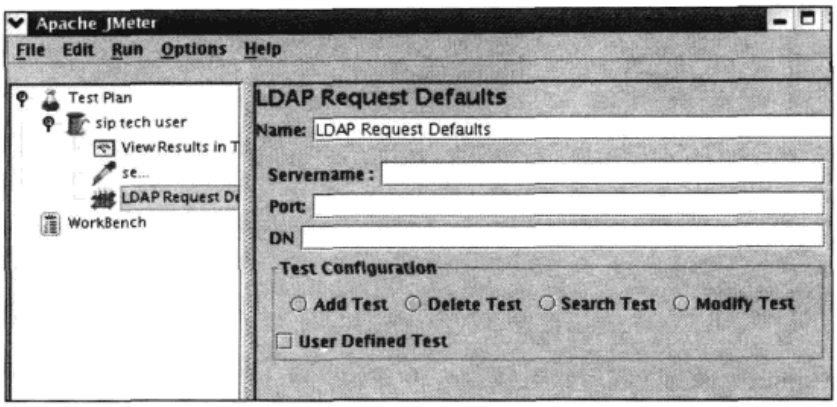


图 6-4　LDAP 请求默认值（LDAP Request Defaults）

将 DN 域设置为测试人员所测试服务器的 Root Dn，并将 Servername 域设置为“localhost”，Port 域设置为 389。如此一来，这些值将作为 LDAP 请求的默认参数。

4．添加 LDAP 请求

在我们的测试计划中，会发送 4 个 LDAP 请求：

- Add 测试。
- Modify 测试。
- Delete 测试。
- Search 测试。

JMeter 会按照请求在测试树中出现的顺序来发送它们。首先为线程组（siptech user）添加一个 LDAP 请求（Add→Sampler→LDAP Request）。接着在测试树中，选中该 LDAP 请求，并编辑其属性，如图 6-5 所示。

（1）将请求名称改为“add”。

（2）选中“Add Test”单选按钮。

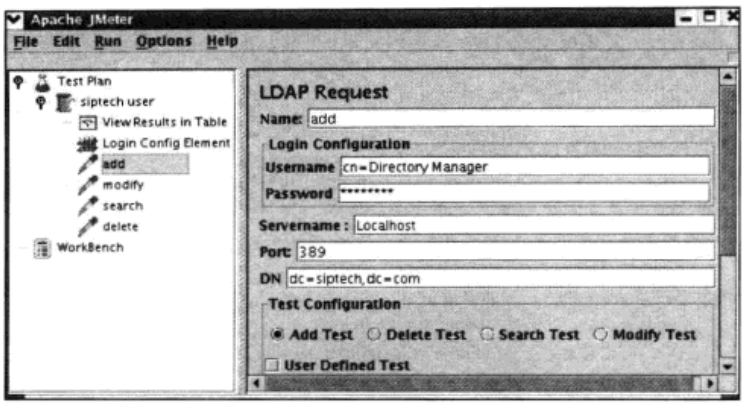


图 6-5　LDAP 请求（LDAP Request）——add

测试人员可以不设置 Server name、Port、Username、Password 和 DN 域的值，因为测试人员已经在登录配置元件（Login Config Element）和 LDAP 请求默认值（LDAP Request Defaults）中设定过这些值了。

接下来，添加更多 LDAP 请求，并编辑其属性。

（1）将请求名称改为“modify”。

（2）选中“Modify Test”单选按钮，如图 6-6 所示。

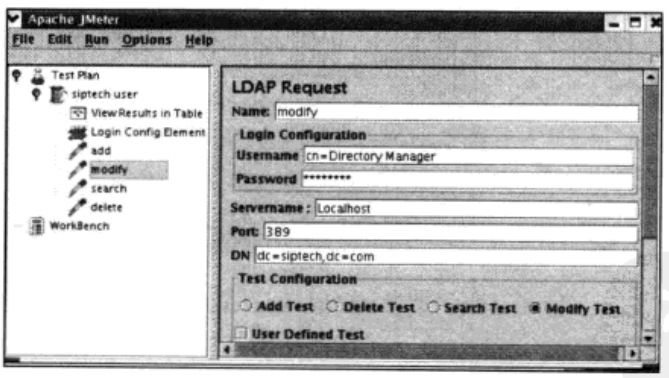


图 6-6　LDAP 请求（LDAP Request）——modify

（1）将请求名称改为“delete”。

（2）选中“Delete Test”单选按钮，如图 6-7 所示。

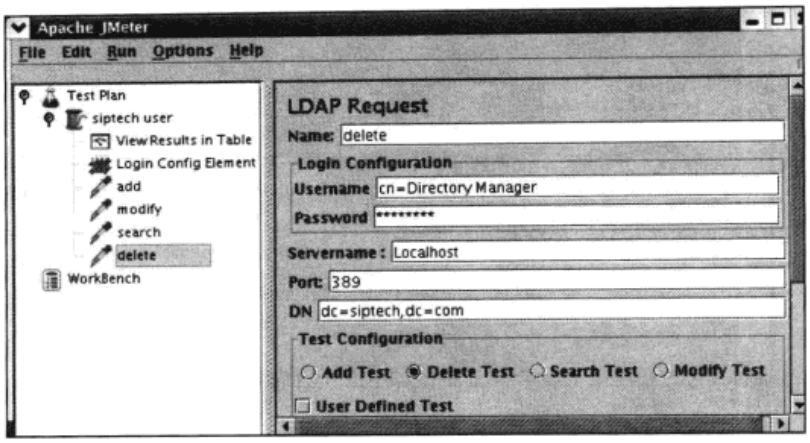


图 6-7　LDAP 请求（LDAP Request）——delete

（1）将请求名称改为“search”。

（2）选中“Search Test”单选按钮，如图 6-8 所示。

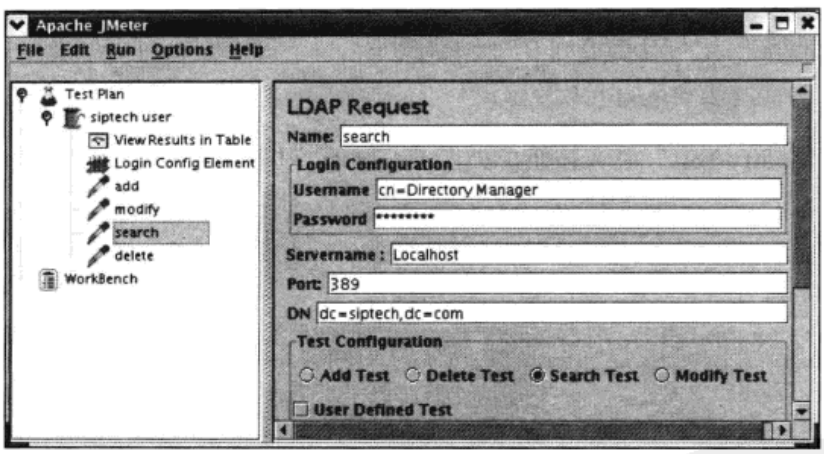


图 6-8　LDAP 请求（LDAP Request）——search

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。该测试元件负责将所有 LDAP 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（siptech users），并添加一个用表格查看结果（View Results in Table）（Add→Listener→View Results in Table），如图 6-9 所示。

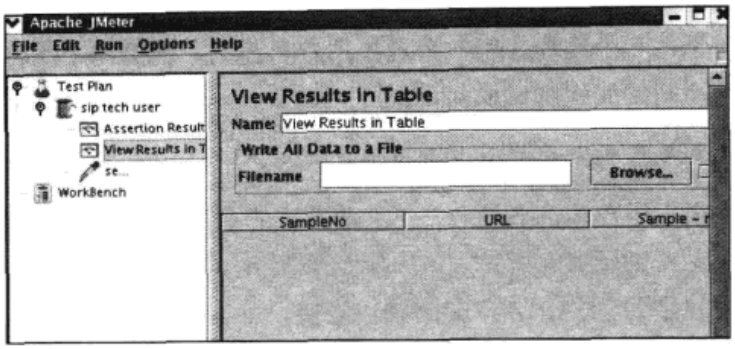


图 6-9　监听器——用表格查看结果（View Results in Table）

6.3　LDAP 常见操作指南

扩展 LDAP 采样器被设计用来支持非常复杂的测试目标，它会尽可能地支持各项 LDAP 操作。在本节中将会介绍共有多少 LDAP 操作，以及它们的作用。针对每一种操作，会简要地介绍它们是如何实现的。

LDAP 服务器可以被视为某种分级的数据库，它们会将对象（条目）存入树中。树的最高部分称为树的根结点。

例如，如果树以 dc=com 作为开始，那么根节点就是 dc=com。

下一级可以存在于根节点之下，例如 dc=siemens。那么对象的全名（DN），就是“dc=siemens,dc=com”。当然，还可以添加下一级，在 dc=siemens,dc=com 下添加“cn=admin”。这样一来，该对象的 DN 就是“cn=admin,dc=siemens,dc=com”。

相对 DN 是 DN 的最后部分，例如 cn=admin。

对象的特性由 objectClasses 决定，objectClasses 可以被视为属性的结合。对象的类型由“structural objectClass”决定，例如 person、organizationalUnit 或者 country。属性中包含有对象的数据，例如 mailadress、name、streetadress 等。每一个属性可以包含 0、1 或者更多值。

1．绑定操作（Bind）

任何与 LDAP 服务器之间的通信，都必须以绑定请求作为开始。LDAP 是一种依赖于状态的协议。如果没有打开与 LDAP 服务器之间的会话，那么其他后续请求都无法被处理。由于 Java 库的一些古怪特性，所以总共有两种不同的绑定操作实现。

1）线程绑定（Thread Bind）

这种绑定意味着打开一个与 LDAP 服务器之间的会话。任何测试计划都应该使用这种操作

作为会话的起点。对于每一个线程（每一个虚拟用户）而言，需要与 LDAP 服务器建立不同的连接，因此需要执行不同的线程绑定操作。

2）单一绑定/解除绑定（Single Bind/Unbind）

这种绑定主要被用于用户验证确认。一个正确设计的 LDAP 客户端，通常会需要验证用户，因此会使用 DN 和密码来执行绑定操作。单一绑定/解除绑定就是为这一目标而实现的。它会通过执行绑定操作，来创建到 LDAP 客户端的独立连接，并在后续关闭该连接（通过单一解除绑定操作）。

2. 解除绑定操作（Unbind）

要关闭一个与 LDAP 服务器之间的连接，需要执行一个解除绑定操作。由于单一绑定/解除绑定已经执行过一次解除绑定操作了，因此只需要再执行一次线程解除绑定操作。这一线程解除绑定操作只是关闭连接，并清除连接所占用的资源。

3. 比较操作（Compare）

比较操作需要 LDAP 对象的完整 DN，以及一个属性和该属性的值。它会简单地检查这个 LDAP 对象是否包含有指定的属性，且该属性的值是否符合预期。一个典型的用法是，校验指定用户组内的特定用户的成员。

4. 搜索操作（Search）

搜索操作就是使用给定的过滤器寻找符合条件的对象。例如，所有满足“employeeType=inactive”的人或者所有 userID= user1 的人。

5. 添加操作（Add）

该操作会简单地在 LDAP 目录中添加一个对象。当然，属性组合和 DN 都应该是正确的。

6. 修改操作（Modify）

该操作会修改指定对象的一个或者多个属性。该操作需要知道待修改对象的 DN，和需要修改的属性及其值。

修改操作分为 3 种。

- 添加：添加一个属性值。
- 替换：使用一个新值来代替旧的属性值。

■ 删除：从一个属性中删除某个值，或者删除一个属性的所有值。

7. 删除操作（Delete）

这一操作从 LDAP 服务器中删除某个对象，它需要待删除对象的 DN。

8. 修改 DN（modDN）

这一操作会修改某个对象的 DN（它会移除该对象）。

该操作可能带来两种效果。第一种，仅仅重命名一条数据，接着指明一个新的 RDN（相对 DN，就是 DN 的最后部分）。

例如，测试人员可以将“cn=admin,dc=siemens,dc=com”重命名为“cn=administrator,dc=Siemens,dc=com”。

第二种，通过指定一个新的上级，来重命名一个完整的分支。

例如，测试人员可以将完整的分支“ou=retired,ou=people,dc=siemens,dc=com”调整为“ou=retired people,dc=siemens,dc=com”。只需指定一个新的 RDN“ou=retired people”，并指定一个新的上级“dc=siemens,dc=com”。

6.4　创建扩展 LDAP 测试计划

扩展 LDAP 采样器拥有更多的可选项，这就意味着测试人员需要花费更多时间来创建一个更加准确的测试计划。测试人员可以对扩展 LDAP 采样器做出精确调整，以便满足不同的测试需求。

在这一节中，将会介绍如何使用扩展 LDAP 采样器来创建扩展 LDAP 测试计划。我们将会创建 1 个并发用户，并向 LDAP 服务器发送 8 个请求。另外，测试人员需要告诉并发用户执行 1 遍测试计划。因此，总的请求数目是（1 并发用户）×（8 请求）×（重复 1 遍）= 8 LDAP 请求。构建该测试计划，测试人员会用到如下测试组件：线程组（Thread Group）、LDAP 扩展请求（LDAP Ext Request）、LDAP 扩展请求默认值（LDAP Extended Request Defaults）和查看结果树（View Results Tree）。

在下面的例子中，假设 LDAP 服务器安装在测试人员的本地计算机上。

针对缺少使用 LDAP 经验的 JMeter 用户，作者专门准备了一个简单的指南（上一节），其

中简要介绍了构建一个复杂测试计划可能会用到的多种 LDAP 操作。

在 LDAP 的 DN（Distinghuished Name）中使用特殊字符时（如在 DN 中使用+号）需要格外小心，这种情况下测试人员需要对特殊字符进行转义，即在特殊字符前加上反斜杠“\”。例如：

- cn=dolf+smits 转义成 cn=dolf\+smits。
- cn=dolf \ smits 转义成 cn=dolf \\ smits。
- cn=c:\log.txt 转义成 cn=c:\\\\log.txt。

1. 添加并发用户

首先要做的还是添加线程组（Thread Group），并修改其默认配置，如图 6-10 所示。

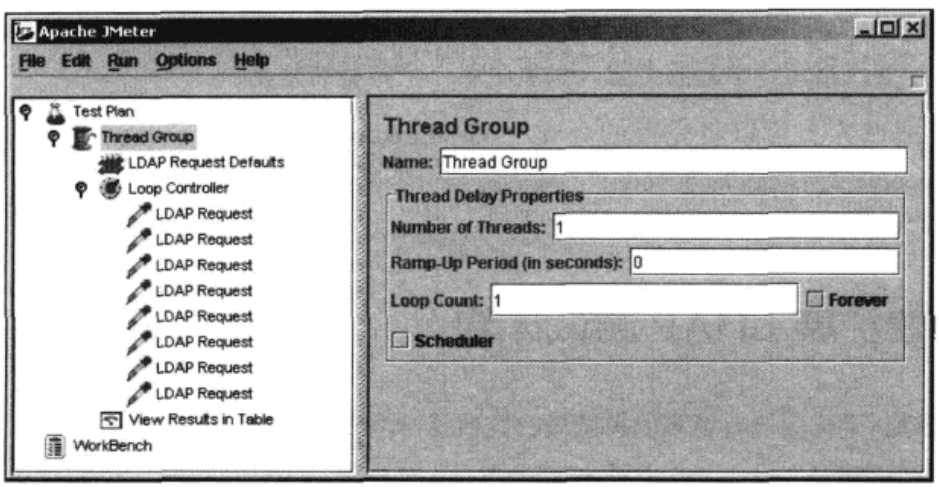


图 6-10　线程组（Thread Group）

2. 添加 LDAP 扩展请求默认值

从选中线程组开始。单击鼠标右键，在弹出的快捷菜单中选择“Add”→“Config Element”→“LDAP Extended 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 6-11 所示。

像大多数 JMeter 测试元件一样，LDAP 扩展请求默认值（LDAP Extended Request Defaults）也有一个名字输入域可供用户修改。在这一例子中，对此不做修改，保留默认值。

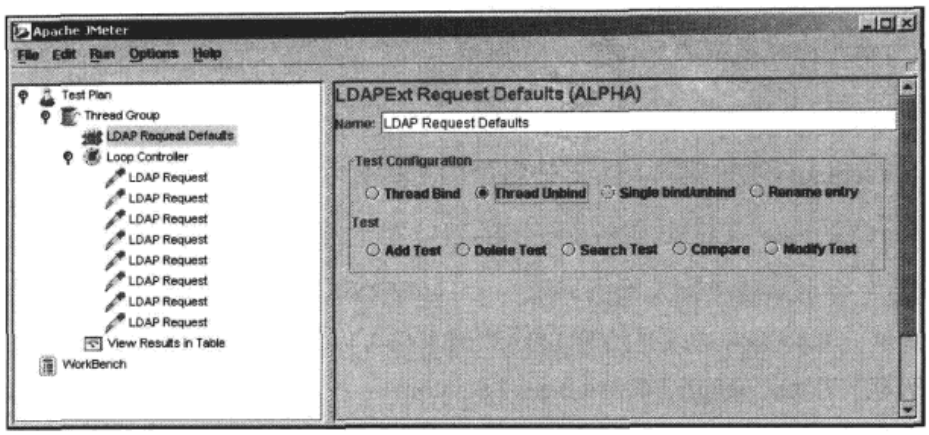


图 6-11　LDAP 扩展请求默认值（LDAP Extended Request Defaults）

对于不同 LDAP 操作而言，都有不同的默认值可供填写。对任何 LDAP 操作都是一样，在填写了某个默认值后，该默认值会被用于后续的 LDAP 扩展请求。对每一个实际请求而言，测试人员都可以选择不使用默认值，只需在 LDAP 扩展请求采样器中填写测试人员想要的值即可。如果没有为 JMeter 测试脚本提供必要的参数，那么测试可能以不可预料的方式失败。

3．添加 LDAP 请求

在我们的测试计划中，将会使用到 8 个 LDAP 请求。

- Thread bind：线程绑定。
- Search Test：搜索测试。
- Compare Test：比较测试。
- Single bind/unbind Test：单一绑定/解除绑定测试。
- Add Test：添加测试。
- Modify Test：修改测试。
- Delete Test：删除测试。
- Rename entry (moddn)：重命名数据条目。
- Thread unbind：线程解除绑定。

JMeter 会按照请求在测试树中出现的顺序来发送它们。要添加一个 LDAP 请求，总是如下：

为线程组添加 LDAP 扩展请求（Add→Sampler→LDAP Ext Request）。接着选中这个新测试元件，查看它的控制面板。

1）添加一个线程绑定请求，如图 6-12 所示

（1）选中“Thread Bind”单选按钮。

（2）在 Servername 域中输入 LDAP 服务器的主机名。

（3）在 Port 域中输入 LDAP 服务器使用的端口号（389）。

（4）（可选）在 DN 域中输入基础 DN，这一基础 DN 将会作为搜索、添加、删除等操作的起点。请注意这一基础 DN 必须是所有 LDAP 请求都可以使用的最高共享层。例如，如果所有用到的信息都存放在 ou=people, dc=siemens, dc=com 之下，那么测试人员就可以使用该值作为基础 DN。这就意味着测试人员不能对分支 ou=users,dc=siemens,dc=com 之下的任何内容做操作。假设测试人员需要对以上两个分支都做改动，那么测试人员需要使用它们的共同上级（dc=siemens,dc=com）作为基础 DN。

（5）（可选）在 Username 域中输入测试人员所使用的用户的 DN，用于校验。如果此域为空，那么就会建立一个匿名绑定。

（6）（可选）在 Password 域中输入测试人员所使用的用户的密码，用于校验。如果此域为空，也会建立一个匿名绑定。

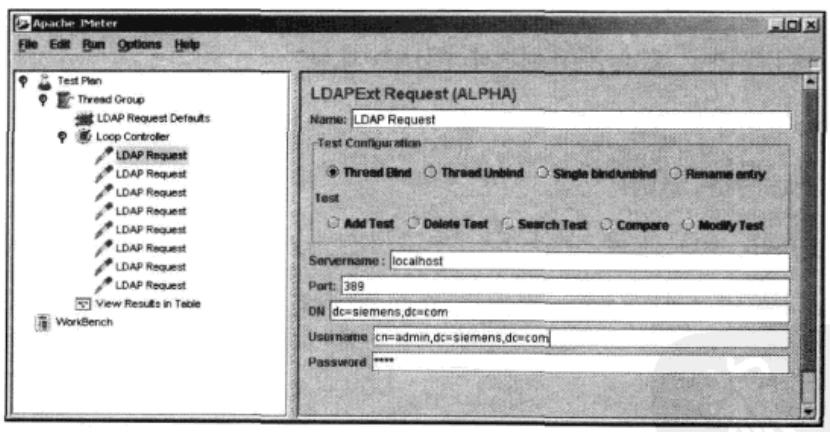


图 6-12　线程绑定（Thread Bind）

2）添加一个搜索请求，如图 6-13 所示

（1）选中“Search Test”单选按钮。

（2）（可选）在 Search base 域中输入搜索的根节点，应该是相对于基础 DN（在线程绑定请求中设定）的 RDN。如果该域为空，将使用基础 DN 作为搜索的根节点。

（3）填写 Search Filter 域，作为搜索的判定标准，例如 cn=john doe。

（4）（可选）在 Scope 域中输入搜索范围，它有 3 个可选项：

- 本层，输入值 0，只在搜索根节点中搜索，只检查根节点的属性或者根节点是否存在。
- 仅一层，输入值 1，仅搜索根节点的下一层。
- 搜索整个分支，输入值 2，搜索根节点之下的所有对象。

（5）（可选）Size limit，指明最大的返回条目数。

（6）（可选）Time limit，指明服务器执行搜索操作的超时时长，单位为毫秒。这并不是 JMeter 等待的最大时长。当一台高性能服务器，通过一条窄带通信线路返回一个很大的结果集时，那么测试人员就不得不等待很长一段时间，直到搜索请求响应到达。但是很显然这个参数并不包含线路的延迟。

（7）（可选）Attributes，搜索响应中测试人员所关注的属性。这可以被用来限制响应的大小，特别是当查询对象包含非常大的属性时（如 JPEG 图像）。有 3 种填写方式：

- 保留为空（默认为空），返回所有属性。
- 放入一个空值（""），由于指定了一个并不存在的属性，所以实际搜索结果中并不返回任何属性。
- 放入属性名，以分号间隔，那么搜索结果中只会返回指定的属性值。

（8）（可选）Return object，可能的值包括“true”和“false”。选择 true，意味着会返回所有 Java 对象属性，也就是将这些 Java 对象属性添加到上面提到的请求属性之中。选择 false，意味着不会返回 Java 对象属性。

（9）（可选）Dereference aliases，可能的值包括“true”和“false”。选择 true，意味着会遵循参考。选择 false，意味着不会。

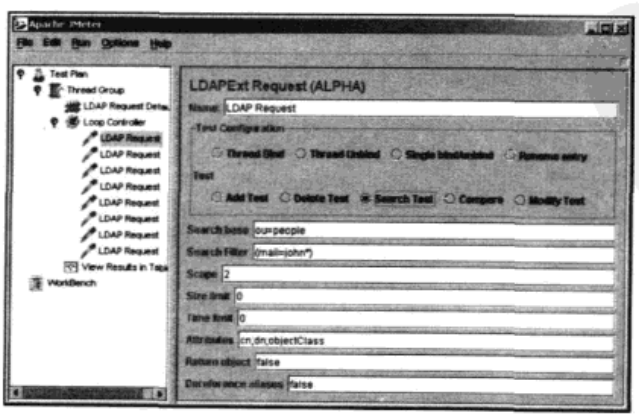


图 6-13　搜索请求（Search Request）

3）添加一个比较请求，如图 6-14 所示

（1）选中“Compare”单选按钮。

（2）输入比较对象的 DN（相对于基础 DN 的 RDN），例如，“cn=john doe,ou=people”。

（3）输入比较过滤器，格式必须符合“attribute=value”，例如：mail=John.doe@siemens.com。

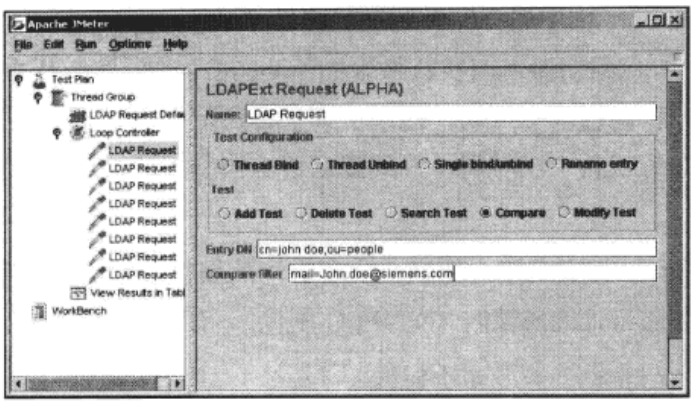


图 6-14　比较请求（Compare Request）

4）添加一个单一绑定/解除绑定，如图 6-15 所示

（1）选中“Single bind/unbind”单选按钮。

（2）输入测试人员用来校验的用户的 DN（全名），例如：cn=john doe,ou=people,dc=siemens,dc=com。如果本域设置为空，那么就会建立一个匿名绑定。

（3）输入测试人员用来校验的用户的密码，如果此域为空，也会建立一个匿名绑定。

（4）请注意，单一绑定/解除绑定实际上是两个独立的操作，但是不容易拆开。

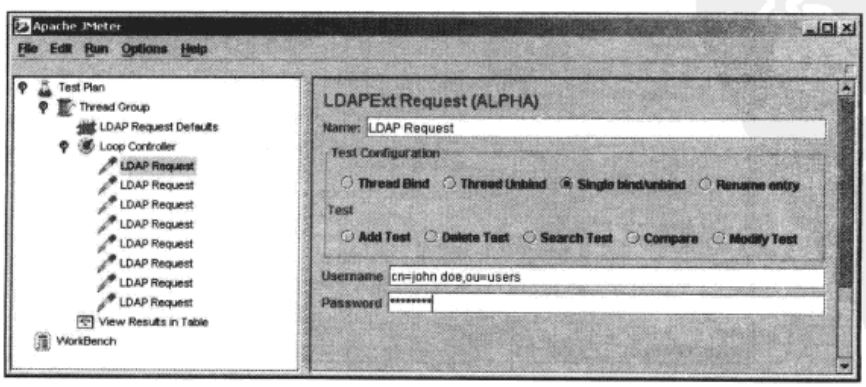


图 6-15　单一绑定/解除绑定（Single Bind/Unbind）

5）添加一个添加请求，如图 6-16 所示

（1）选中“Add Test”选项。

（2）输入添加对象的 DN，相对于基础 DN 的 RDN。

（3）在“Add Test”表格中增加一行，输入属性及其值。如果要为一个属性赋予多个值，测试人员只需多次添加属性，并设置不同的值就行了。所有必需的属性和值都要在这里设定。

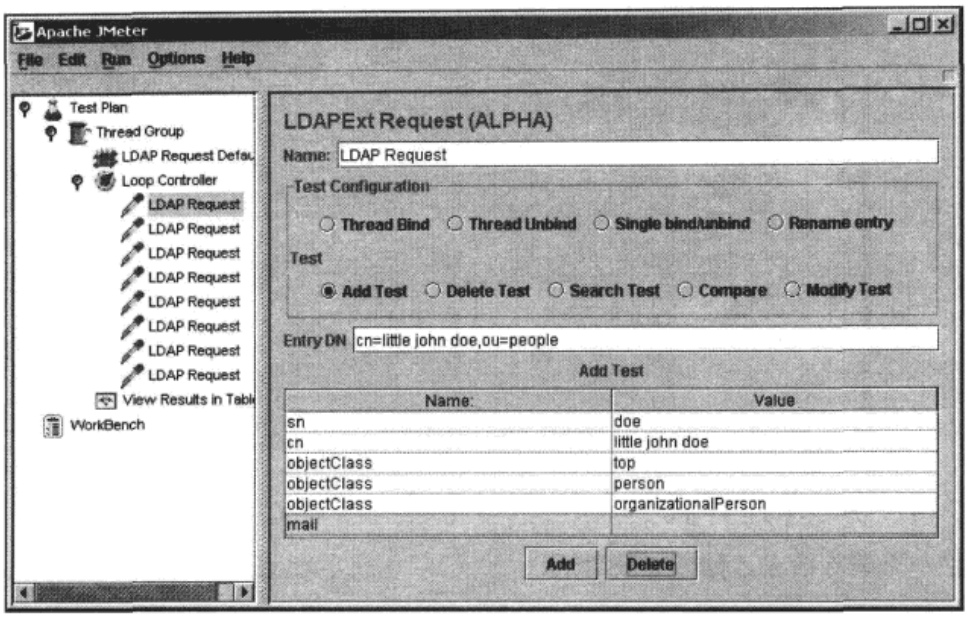


图 6-16　添加请求（Add Request）

6）添加一个修改请求，如图 6-17 所示

（1）选中“Modify Test”单选按钮。

（2）输入修改对象的 DN，相对于基础 DN 的 RDN。

（3）在“Modify Test”表格中增加一行。

（4）测试人员需要输入想要修改的属性、（可选）属性值、操作码。操作码的含义：

- add，有属性值（这种情况下不是可选的），会被添加到对应属性之下。如果属性不存在，会创建属性并添加值。如果属性存在且为多值属性，则为属性添加新值。如果属性存在且为单值属性，那么测试就会失败。
- replace，使用新值去替换属性原有的值。如果属性不存在，会创建属性并添加值。如果属性存在，则旧值会被移除，使用新值代替。

- delete，如果没有给定值，那么所有的属性值都会被移除。如果给定了值，那么只有指定的值会被移除。如果给定的属性值不存在，那么测试就会失败。

（5）（可选）在“Modify Test”表格中添加更多修改内容。

只有当所有指定的修改内容都被成功修改后，整个修改测试才能通过。只要有一项修改失败，那么所有的修改都不会被执行，待修改的数据条目会保持不变。

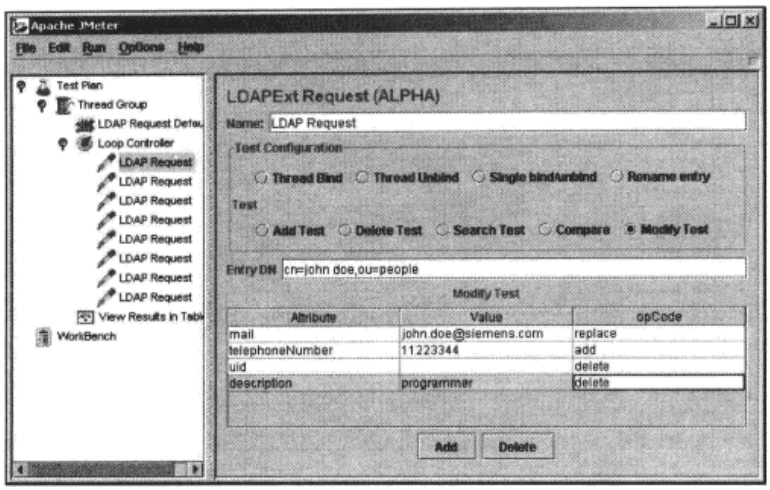


图 6-17　修改请求（Modify Request）

7）添加一个删除请求，如图 6-18 所示

（1）选中“Delete Test”单选按钮。

（2）在 Delete 输入域中输入待删除数据的 DN，相对于基础 DN 的 RDN。例如，假设测试人员想要移除“cn=john doe,ou=people,dc=siemens,dc=com”，那么测试人员可以将基础 DN 设置为“dc=siemens,dc=com”，并在 Delete 域中输入“cn=john doe,ou=people”。

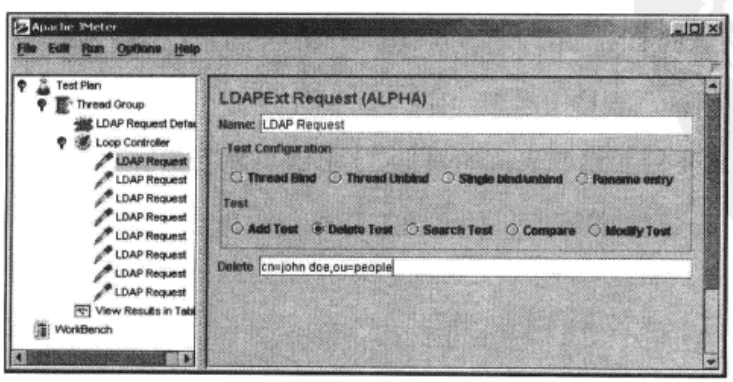


图 6-18　删除请求（Delete Request）

8）添加一个重命名请求，如图 6-19 所示

（1）选中“Rename entry”选项。

（2）在 Old entry name 输入域中填写数据条目的名称，相对于基础 DN。例如，假设测试人员想要移除“cn=john doe,ou=people,dc=siemens,dc=com”，那么测试人员可以将基础 DN 设置为“dc=siemens,dc=com”，并在 Old entry name 域中输入“cn=john doe,ou=people”。

（3）在 New distinghuised name 输入域中输入数据条目的新名称（RDN），相对于基础 DN。如果测试人员只改变 RDN，那么就会简单地重命名该条目。如果测试人员要移动一条数据，例如从“cn=john doe,ou=people”变为“cn=john doe,ou=users”，就只会移动该条目。测试人员也可以移动整个分支（如果测试人员的 LDAP 服务器支持），例如将“ou=people,ou=retired,”变为“ou=oldusers,ou=users,”，这样就会移动整个分支，即将所有退休人员放到测试树的新位置。

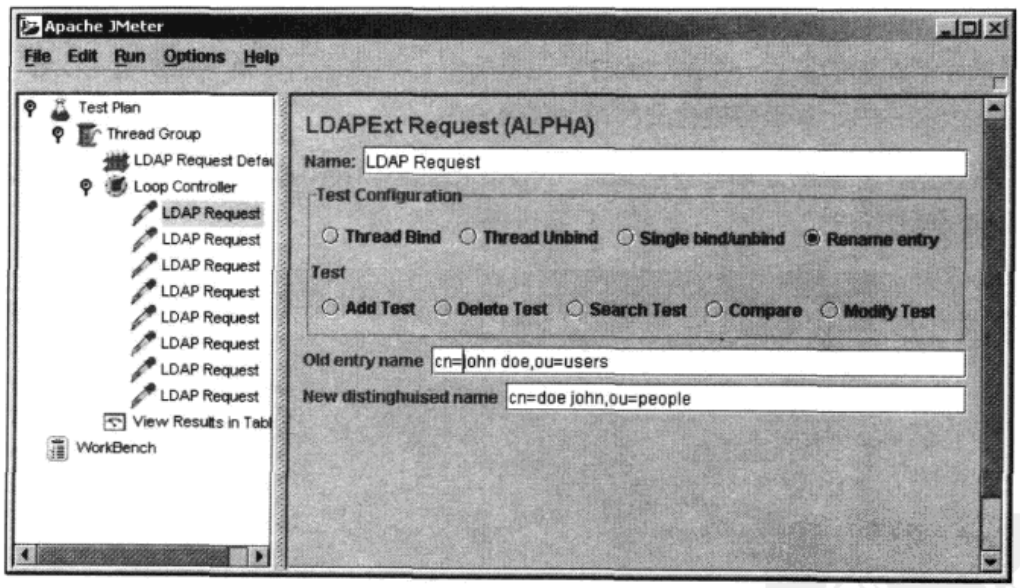


图 6-19　重命名请求（Moddn）

9）添加一个解除绑定请求，如图 6-19 所示。

选中“Thread Unbind”单选按钮即可，因为仅仅是关闭连接。需要知道的信息，系统都已经知道了。

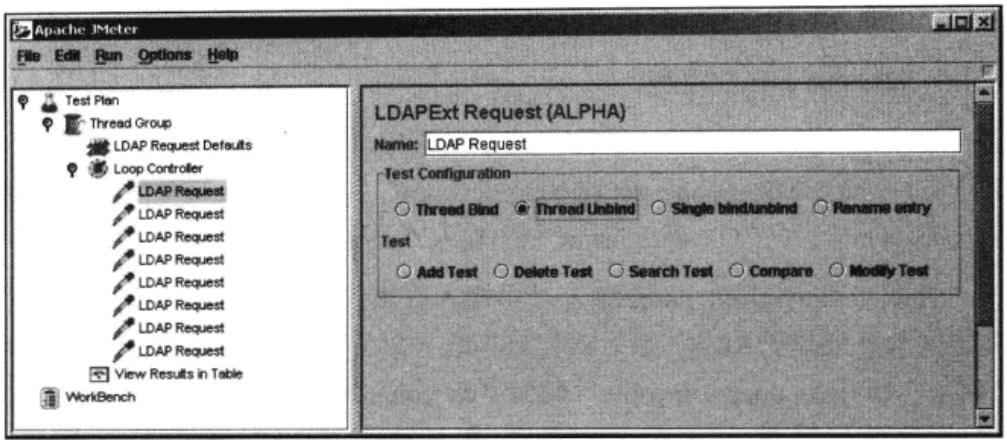


图 6-20　解除绑定请求（Thread Unbind）

4．添加监听器用于查看/存储测试结果

为测试计划添加的最后一个测试元件就是监听器。该测试元件负责将所有 LDAP 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，并添加一个查看结果树（View Results Tree）（Add→Listener→View Results Tree），如图 6-21 所示。

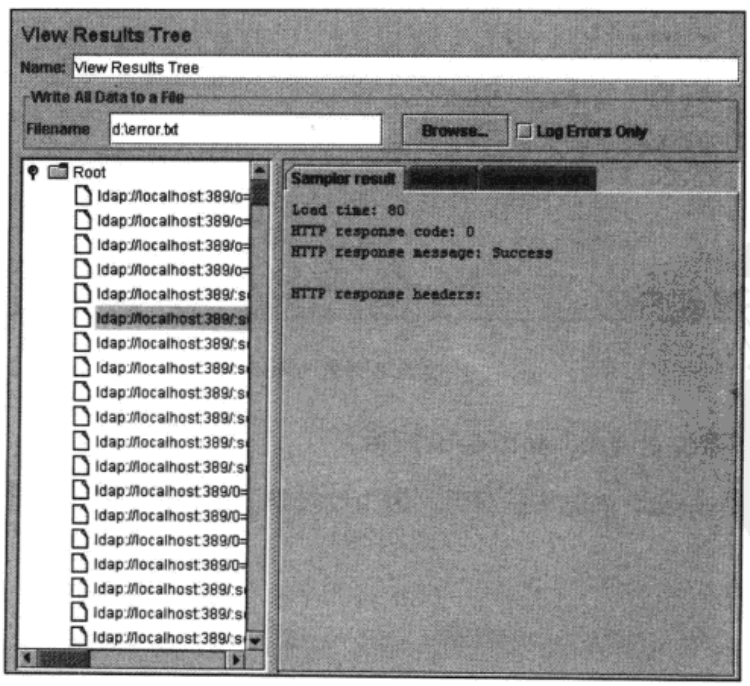


图 6-21　查看结果树（View Results Tree）

在这一监听器中，有 3 个选项卡可供查看，分别为 Sampler Result、Request 和 Response data。

- Sampler result 选项卡中只包含 response time、returncode 和 return message。
- Request 选项卡中简单展示了请求的内容，另外关联信息在这里不会展示。
- 在 Response data 选项卡中包含有发送请求的全部细节，以及接收响应的全部细节。详细信息以自定义 XML 格式加以展示。

6.5　本章小结

本章首先对 LDAP 的基本概念做了一个简要介绍，为后续开发 LDAP 性能测试脚本打下了基础；接着介绍了如何创建 LDAP 测试计划；之后介绍了 LDAP 的常见操作；最后介绍了如何使用扩展 LDAP 采样器创建 LDAP 测试计划。

第 7 章

Web Service性能测试脚本开发

7.1 Web Service 是什么

Web Service 是由企业发布的完成其特定商务需求的在线应用服务，其他公司或应用软件能够通过 Internet 来访问并使用这项在线服务。它是一种构建应用程序的普遍模型，可以在任何支持网络通信的操作系统中实施运行；它是一种新的 Web 应用程序分支，是自包含、自描述、模块化的应用，可以发布、定位、通过 Web 调用。Web Service 是一个应用组件，它逻辑性地为其他应用程序提供数据与服务。各应用程序通过网络协议和规定的一些标准数据格式（Http、XML、Soap）来访问 Web Service，通过 Web Service 内部执行得到所需结果。Web Service 可以执行从简单请求到复杂商务处理的任何功能。一旦部署以后，其他 Web Service 应用程序可以发现并调用它部署的服务。如图 7-1 所示是一个 Web Service 的典型例子。

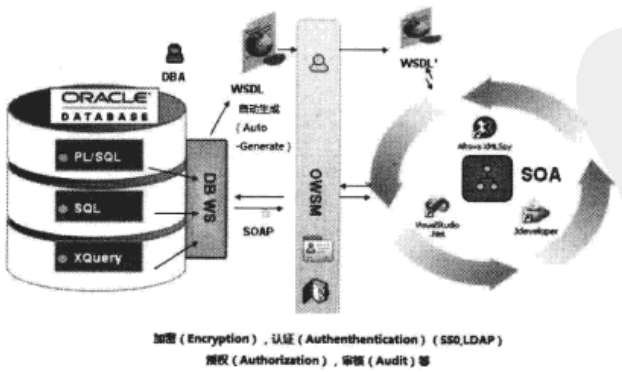


图 7-1 一个 Web Service 的典型例子

1．技术和规则

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

- XML：描述数据的标准方法。
- SOAP：表示信息交换的协议。
- WSDL：Web 服务描述语言。
- UDDI（Universal Description, Discovery and Integration）：通用描述、发现与集成，它是一种独立于平台的、基于 XML 语言的用于在互联网上描述商务的协议。

2．目标

实际上，Web Service 的主要目标是跨平台的可互操作性。为了达到这一目标，Web Service 完全基于 XML（可扩展标记语言）、XSD（XMLSchema）等独立于平台、独立于软件供应商的标准，是创建可互操作的、分布式应用程序的新平台。由此可以看出，在以下 4 种情况下，使用 Web Service 会带来极大的好处。

长项一：跨防火墙的通信

如果应用程序有成千上万的用户，而且分布在世界各地，那么客户端和服务器之间的通信将是一个棘手的问题，因为客户端和服务器之间通常会有防火墙或者代理服务器。在这种情况下，使用 DCOM 就不是那么简单，通常也不便于把客户端程序发布到数量如此庞大的每一个用户手中。传统的做法是，选用浏览器作为客户端，写下一大堆 ASP 页面，把应用程序的中间层暴露给最终用户。这样做的结果是开发难度大，程序很难维护。

如果中间层组件换成 Web Service 的话，就可以从用户界面直接调用中间层组件，从而省掉建立 ASP 页面的这一步。要调用 Web Service，可以直接使用 Microsoft SOAP Toolkit 或.NET 这样的 SOAP 客户端，也可以使用自己开发的 SOAP 客户端，然后把它和应用程序连接起来。这样不仅缩短了开发周期，还减少了代码复杂度，并能够增强应用程序的可维护性。同时，应用程序也不再需要在每次调用中间层组件时，都跳转到相应的“结果页”。

从经验来看，在一个用户界面和中间层有较多交互的应用程序中，使用 Web Service 这种结构，可以节省花在用户界面编程上 20%的开发时间。另外，这样一个由 Web Service 组成的中间层，完全可以在应用程序集成或其他场合下重用。最后，通过 Web Service 把应用程序的逻辑和数据“暴露”出来，还可以让其他平台上的客户重用这些应用程序。

长项二：应用程序集成

企业级的应用程序开发者都知道，企业里经常都要把用不同语言写成的、在不同平台上运行的各种程序集成起来，而这种集成将花费很大的开发力量。应用程序经常需要从运行在 IBM 主机上的程序中获取数据，或者把数据发送到主机或 UNIX 应用程序中去。即使在同一个平台上，不同软件厂商生产的各种软件也常常需要集成起来。通过 Web Service，应用程序可以用标准的方法把功能和数据“暴露”出来，供其他应用程序使用。

例如，有一个订单登录程序，用于登录从客户来的新订单，包括客户信息、发货地址、数量、价格和付款方式等内容；还有一个订单执行程序，用于实际货物发送的管理。这两个程序来自不同软件厂商。一份新订单进来之后，订单登录程序需要通知订单执行程序发送货物。通过在订单执行程序上面增加一层 Web Service，订单执行程序可以把“AddOrder”函数“暴露”出来。这样，每当有新订单到来时，订单登录程序就可以调用这个函数来发送货物了。

长项三：B2B 的集成

用 Web Service 集成应用程序，可以使公司内部的商务处理更加自动化。但当交易跨越供应商和客户、突破公司的界限时会怎么样呢？跨公司的商务交易集成通常叫做 B2B 集成。

Web Service 是 B2B 集成成功的关键。通过 Web Service，公司可以把关键的商务应用“暴露”给指定的供应商和客户。例如，把电子下单系统和电子发票系统“暴露”出来，客户就可以以电子的方式发送订单，供应商则可以以电子的方式发送原料采购发票。当然，这并不是一个新的概念，EDI（电子文档交换）早就是这样了。但是，Web Service 的实现要比 EDI 简单得多。而且 Web Service 运行在 Internet 上，在世界任何地方都可轻易实现，其运行成本就相对较低。不过，Web Service 并不像 EDI 那样是文档交换或 B2B 集成的完整解决方案，Web Service 只是 B2B 集成的一个关键部分，还需要许多其他的部分才能实现集成。

用 Web Service 来实现 B2B 集成的最大好处在于可以轻易实现互操作性。只要把商务逻辑“暴露”出来，成为 Web Service，就可以让任何指定的合作伙伴调用这些商务逻辑，而不管他们的系统在什么平台上运行、使用什么开发语言。这样就大大减少了花在 B2B 集成上的时间和成本，让许多原本无法承受 EDI 的中小企业也能实现 B2B 集成。

长项四：软件和数据重用

软件重用是一个很大的主题，重用的形式很多，重用的程度有大有小。最基本的形式是源代码模块或者类一级的重用，另一种形式是二进制形式的组件重用。

当前，像表格控件或用户界面控件这样的可重用软件组件，在市场上都占有很大的份额。但这类软件的重用有一个很大的限制，就是重用仅限于代码，数据不能重用。原因在于，发布

组件甚至源代码都比较容易，但要发布数据就没那么容易了，除非是不会经常变化的静态数据。

Web Service 在允许重用代码的同时，可以重用代码背后的数据。使用 Web Service，再也不必像以前那样，要先从第三方购买、安装软件组件，再从应用程序中调用这些组件，只需要直接调用远端的 Web Service 就可以了。例如，要在应用程序中确认用户输入的地址，只需把这个地址直接发送给相应的 Web Service，这个 Web Service 就会帮测试人员查阅街道地址、城市、省区和邮政编码等信息，确认这个地址是否在相应的邮政编码区域。Web Service 的提供商可以按时间或使用次数来对这项服务进行收费。这样的服务要通过组件重用来实现是不可能的，那样的话测试人员必须下载并安装好包含街道地址、城市、省区和邮政编码等信息的数据库，而且这个数据库还是不能实时更新的。

另一种软件重用的情况是，把好几个应用程序的功能集成起来。例如，要建立一个局域网上的门户站点应用，让用户既可以查询联邦快递包裹，查看股市行情，又可以管理自己的日程安排，还可以在线购买电影票。现在 Web 上有很多应用程序供应商都在其应用中实现了这些功能，一旦他们把这些功能都通过 Web Service“暴露”出来，就可以非常容易地把所有这些功能都集成到测试人员的门户站点中，为用户提供一个统一的、友好的界面。

将来，许多应用程序都会利用 Web Service，把当前基于组件的应用程序结构扩展为组件/Web Service 的混合结构，既可以在应用程序中使用第三方的 Web Service 提供的功能，也可以把自己的应用程序功能通过 Web Service 提供给别人。两种情况下，都可以重用代码和代码背后的数据。

从以上论述中可以看出，Web Service 在通过 Web 进行互操作或远程调用时是最有用的。不过，也有一些情况，Web Service 根本不能带来任何好处。

短处一：单机应用程序

目前，企业和个人还使用着很多桌面应用程序，其中一些只需要与本机上的其他程序通信。在这种情况下，最好就不要用 Web Service，只要用本地的 API 就可以了。COM 非常适合于在这种情况下工作，因为它既小又快。运行在同一台服务器上的服务器软件也是这样，最好直接用 COM 或其他本地的 API 来进行应用程序间的调用。当然 Web Service 也能用在这些场合，但那样不仅消耗太大，而且不会带来任何好处。

短处二：局域网的同构应用程序

在许多应用中，所有的程序都是用 VB 或 VC 开发的，都在 Windows 平台下使用 COM，都运行在同一个局域网上。例如，有两个服务器应用程序需要相互通信，或者有一个 Win32 或

WinForm 的客户程序要连接局域网上另一个服务器的程序。在这些程序中，使用 DCOM 会比 SOAP/HTTP 有效得多。与此相类似，如果一个.NET 程序要连接到局域网上的另一个.NET 程序，应该使用.NETremoting。有趣的是，在.NETremoting 中也可以指定使用 SOAP/HTTP 来进行 Web Service 调用。不过最好还是直接通过 TCP 进行 RPC 调用，那样会有效得多。

总之，只要从应用程序结构的角度看，有别的方法比 Web Service 更有效、更可行，那就不要用 Web Service。

7.2 创建 Web Service 测试计划

在这一节中，作者将会介绍如何创建一个测试计划来测试 Web Service。我们将会创建 5 个并发用户，并向 1 个页面发送请求。另外，测试人员需要告诉并发用户执行 2 遍测试计划。因此，总的请求数目是（5 并发用户）×（1 请求）×（重复 2 遍） =10 HTTP 请求。构建该测试计划，测试人员会用到如下测试组件：线程组（Thread Group ）、Web Service(SOAP) Request 和图形结果（Graph Results）。

另外需要关注的是 Web Service 采样器。目前的实现需要用到 Apache SOAP 驱动，该驱动会用到 SUN 公司的 activation.jar 和 mail.jar。由于许可（license）限制，JMeter 不能将这些 jar 文件放到 bin 目录下。如果采样器报告从 Web Service 得到了一个错误信息，那么需要仔细检查 SOAP 消息，确保消息格式是正确的。需要特别注意的是，xmlns 属性必须与 WSDL 的描述保持一致。如果 XML 的名称不一致，Web Service 通常会返回一个错误。

1. 添加并发用户

首先要做的还是添加线程组（Thread Group），并修改其默认配置，如图 7-2 所示。

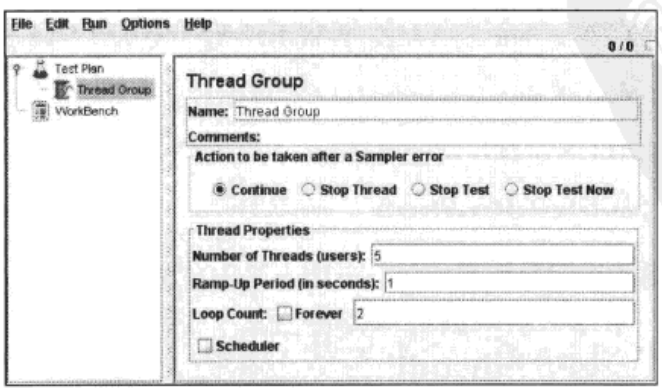


图 7-2 线程组（Thread Group）

2. 添加 Web Service 请求

在我们的测试计划中会使用到.NET Web Service。JMeter 用户只会用到 Web Service 采样器，因此本书不会涉及如何编写一个 Web Service。如果测试人员对编写 Web Service 感兴趣，那就 Google 或者百度一下 Web Service，测试人员会找到很多关于如何使用 Java 和.NET 实现 Web Service 的资料，还会发现通过 Java 和.NET 实现 Web Service 的方法存在很大差异。关于这两种实现方式存在哪些差异，超出了本书范围，请参考其他书籍。

从为线程组添加 WebService(SOAP) Request 采样器开始（Add → Sampler → Web Service(SOAP) Request）。接着选中该采样器，并编辑其属性，如图 7-3 所示。

（1）将 Name 域改为“WebService(SOAP) Request”。

（2）输入 WSDL 的 URL，并单击“Load WSDL”按钮。

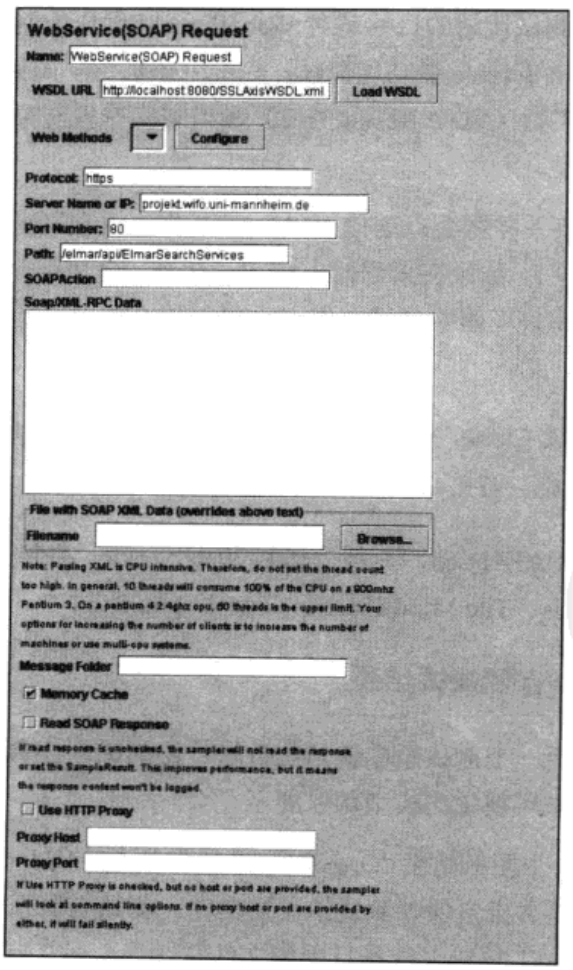


图 7-3　WebService(SOAP) Request 采样器

如果 WSDL 文件被正确加载，那么“Web Methods”下拉列表就会被弹出。如果下拉列表仍然为空，这就意味着获取 WSDL 发生了错误。可以通过浏览器来访问 WSDL 对应的 XML，以便测试它的有效性。例如，假设测试人员要测试 IIS Web Service，那么其 URL 应该形如：http://localhost/myWebService/Service.asmx? WSDL。

接下来，选定 Web MethodS，并单击配置按钮（Configure）。如此一来，采样器会自行填充与 URL 和 SOAPAction 相关的输入域。假设 WSDL 是正确的，那么 SOAPAction 也应该是正确的。

最后一步是将 SOAP 消息粘贴在“SOAP/XML-RPC Data”文本输入域之中。另外，测试人员还可以将 SOAP 消息保存在一个文件之中，通过“Browse…”按钮从本地加载。还有第三种方式，即使用消息文件夹（Message Folder）。采样器会自动从指定文件夹中加载文件，并将其中的文字信息用于填充 SOAP 消息。

如果测试人员不希望 JMeter 读取 SOAP Web Service 的响应，那么就不要选中“Read SOAP Responses”复选项。如果测试计划的目标是对 Web Service 做压力测试，则不选中该选项。如果测试计划的目标是对 Web Service 做功能测试，则选中该选项。如果“Read SOAP Responses”没有被选中，那么查看结果树（View Result Tree）和用表格查看结果（View Results in Table）中就无法看到相关统计结果。

另外需要注意的是，该采样器会自动使用 JMeter 从命令行获取的代理主机名和端口号（前提是采样器中没有设置代理）。如果在采样器中指定了代理，就使用此处的设定。如果测试要求必须使用代理，而命令行和采样器中都没有指定代理，那么该采样器就会失败。这种失败可能会为用户带来很大困扰。

假如测试对象是 Cassini Web Service，那么很不幸它并不可靠。它不能正确地关闭连接，这就会导致 JMeter 挂死或者无法得到响应内容。

现在只有.NET 使用 SOAPAction，其他大多数 Web Service 都会使用空的 SOAPAction，例如 JWSDP、Weblogic、Axis、The Mind Electric Glue 和 gSoap。

3．添加监听器用于查看/存储测试结果

为测试计划添加的最后一个测试原件就是监听器。该测试原件负责将所有 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，并添加一个图形结果（Graph Results）监听器（Add→Listener→Graph Results），如图 7-4 所示。接下来，需要指定保存测试结果的文件。测试人员既可以在 Filename 输入域中填充，也可以通过“Browse”按钮来选择目录和文件名。

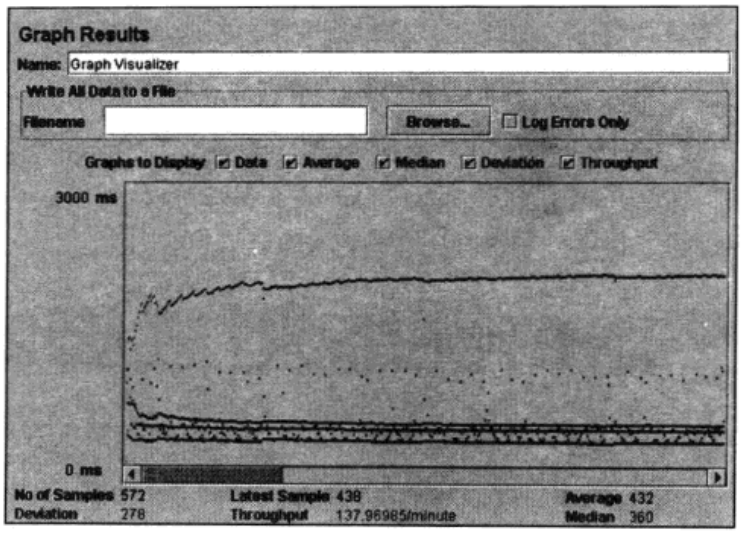


图 7-4　图形结果（Graph Results）

7.3　本章小结

本章首先对 Web Service 的基本概念做了一个介绍，为后续开发 Web Service 性能测试脚本打下了基础；接着介绍了如何创建 Web Service 测试计划。

第 8 章

JMS性能测试脚本开发

8.1 JMS 是什么

JMS 即 Java 消息服务（Java Message Service），应用程序接口是一个 Java 平台中关于面向消息中间件（MOM）的 API，用于在两个应用程序之间或分布式系统中发送消息，进行异步通信。Java 消息服务是一个与具体平台无关的 API，绝大多数 MOM 提供商都对 JMS 提供支持。

1. 定义

JMS 是 Java 平台上有关面向消息中间件的技术规范，它便于消息系统中的 Java 应用程序进行消息交换，并且通过提供标准的产生、发送、接收消息的接口简化企业应用的开发，翻译为 Java 消息服务。如图 8-1 所示为 JMS 的标志。

图 8-1 JMS 标志

2. 简介

JMS 是一种与厂商无关的 API，用来访问消息收发系统。它类似于 JDBC（Java Database Connectivity），这里，JDBC 是可以用来访问许多不同关系数据库的 API，而 JMS 则提供同样与厂商无关的访问方法，以访问消息收发服务。许多厂商目前都支持 JMS，包括 IBM 的 MQSeries、BEA 的 Weblogic JMS Service 和 Progress 的 SonicMQ，这只是几个例子。JMS 使用户能够通过消息收发服务（有时称为消息中介程序或路由器）从一个 JMS 客户机向另一个 JML 客户机发送消息。消息是 JMS 中的一种类型对象，由报头和消息主体两部分组成。报头由路由信息及有关该消息的元数据组成。消息主体则携带着应用程序的数据或有效负载。根据有效负载的类型来划分，可以将消息分为几种类型，它们分别携带简单文本（TextMessage）、可序列化的对象（ObjectMessage）、属性集合（MapMessage）、字节流（BytesMessage）、原始值流（StreamMessage），还有无有效负载的消息（Message）。

3. 历史

Java 消息服务是一个在 Java 标准化组织（JCP）内开发的标准（代号 JSR 914）。2001 年 6 月 25 日，Java 消息服务发布 JMS 1.0.2b。2002 年 3 月 18 日，Java 消息服务发布 1.1，统一了消息域。

4. 体系架构

JMS 由以下元素组成，而生产/消费消息的步骤如图 8-2 所示。

- JMS 提供者，连接面向消息中间件的 JMS 接口的一个实现。提供者可以是 Java 平台的 JMS 实现，也可以是非 Java 平台的面向消息中间件的适配器。
- JMS 客户，生产或消费消息的基于 Java 的应用程序或对象。
- JMS 生产者，创建并发送消息的 JMS 客户。
- JMS 消费者，接收消息的 JMS 客户。
- JMS 消息，包括可以在 JMS 客户之间传递的数据的对象。
- JMS 队列，一个容纳那些被发送的等待阅读的消息的区域。队列暗示，这些消息将按照顺序发送。一旦一个消息被阅读，该消息将被从队列中移走。
- JMS 主题，一种支持发送消息给多个订阅者的机制。

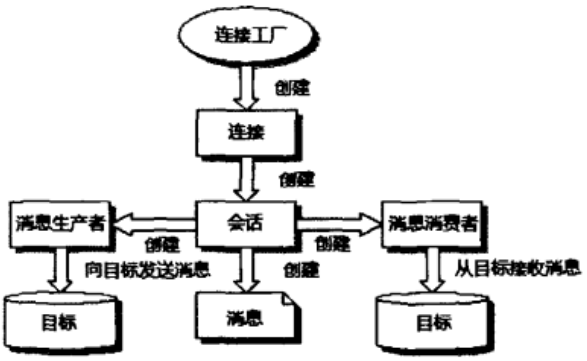


图 8-2 JMS 生产/消费消息的步骤

5. JMS 模型

Java 消息服务应用程序结构支持两种模型：

- 点对点或队列模型。
- 发布者/订阅者模型。

在点对点或队列模型下，一个生产者向一个特定的队列发布消息，一个消费者从该队列中读取消息。这里，生产者知道消费者的队列，并直接将消息发送到消费者的队列。这种模式被概括为：只有一个消费者将获得消息。生产者不需要在接收者消费该消息期间处于运行状态，接收者也同样不需要在消息发送时处于运行状态。每一个成功处理的消息都由接收者签收。

发布者/订阅者模型支持向一个特定的消息主题发布消息。0 或多个订阅者可能对接收来自特定消息主题的消息感兴趣。在这种模型下，发布者和订阅者彼此不知道对方。这种模式好比是匿名公告板。这种模式被概括为：多个消费者可以获得消息。在发布者和订阅者之间存在时间依赖性。发布者需要建立一个订阅（Subscription），以便客户能够购订阅。订阅者必须保持持续的活动状态以接收消息，除非订阅者建立了持久的订阅。在该情况下，在订阅者未连接时发布的消息将在订阅者重新连接时重新发布。

使用 Java 语言，JMS 提供了将应用与提供数据的传输层相分离的方式。同一组 Java 类可以通过 JNDI 中关于提供者的信息连接不同的 JMS 提供者。这一组类首先使用一个连接工厂以连接到队列或主题，然后发送或发布消息。在接收端，客户接收或订阅这些消息。

6. 传递消息方式

JMS 现在有两种传递消息的方式，标记为 NON_PERSISTENT 的消息最多投递一次，而标记为 PERSISTENT 的消息将使用暂存后再转送的机理投递。如果一个 JMS 服务离线，那么持

久性消息不会丢失，但是得等到这个服务恢复联机时才会被传递。所以默认的消息传递方式是非持久性的。即使使用非持久性消息可能降低内务和需要的存储器，并且这种传递方式只有当测试人员不需要接收所有的消息时才使用。

虽然 JMS 规范并不需要 JMS 供应商实现消息的优先级路线，但是它需要递送加快的消息优先于普通级别的消息。JMS 定义了从 0 到 9 的优先级路线级别，0 是最低的优先级，而 9 则是最高的。更特殊的是，0～4 是正常优先级的变化幅度，而 5～9 是加快的优先级的变化幅度。举例来说：

```
topicPublisher.publish (message, DeliveryMode.PERSISTENT, 8, 10000);
//Pub-Sub
```

或：

```
queueSender.send(message,DeliveryMode.PERSISTENT, 8, 10000);
```

P2P 这个代码片断有两种消息模型，映射递送方式是持久的，优先级为加快型，生存周期是 10000（以毫秒度量）。如果生存周期设置为零，这则消息将永远不会过期。当消息需要时间限制否则将使其无效时，设置生存周期是有用的。

JMS 定义了 5 种不同的消息正文格式，以及调用的消息类型，允许测试人员发送并接收一些不同形式的数据，提供现有消息格式的一些级别的兼容性。

- StreamMessage：Java 原始值的数据流。
- MapMessage：一套名称-值对。
- TextMessage：一个字符串对象。
- ObjectMessage：一个序列化的 Java 对象。
- BytesMessage：一个未解释字节的数据流。

7．JMS 应用程序接口

1）ConnectionFactory 接口（连接工厂）

用户用来创建到 JMS 提供者的连接的对象。JMS 客户通过可移植的接口访问连接，这样当下层的实现改变时，代码不需要进行修改。管理员在 JNDI 名字空间中配置连接工厂，这样，JMS 客户才能够查找到它们。根据消息类型的不同，用户将使用队列连接工厂，或者主题连接工厂。

2）Connection 接口（连接）

连接代表了应用程序和消息服务器之间的通信链路。在获得了连接工厂后，就可以创建一个与 JMS 提供者的连接。根据不同的连接类型，连接允许用户创建会话，以发送和接收队列和

主题到目标。

3）Destination 接口（目标）

目标是一个包装了消息目标标识符的对象，消息目标是指消息发布和接收的地点，或者是队列，或者是主题。JMS 管理员创建这些对象，然后用户通过 JNDI 发现它们。和连接工厂一样，管理员可以创建两种类型的目标，即点对点模型的队列，以及发布者/订阅者模型的主题。

4）MessageConsumer 接口（消息消费者）

由会话创建的对象，用于接收发送到目标的消息。消费者可以同步（阻塞模式）或异步（非阻塞模式）接收队列和主题类型的消息。

5）MessageProducer 接口（消息生产者）

由会话创建的对象，用于发送消息到目标。用户可以创建某个目标的发送者，也可以创建一个通用的发送者，在发送消息时指定目标。

6）Message 接口（消息）

消息是在消费者和生产者之间传送的对象，也就是说从一个应用程序传送到另一个应用程序。一个消息有 3 个主要部分。

- 消息头（必须）：包含用于识别和为消息寻找路由的操作设置。
- 一组消息属性（可选）：包含额外的属性，支持其他提供者和用户的兼容。可以创建定制的字段和过滤器（消息选择器）。
- 一个消息体（可选）：允许用户创建 5 种类型的消息（文本消息、映射消息、字节消息、流消息和对象消息）。

消息接口非常灵活，并提供了许多方式来定制消息的内容。

7）Session 接口（会话）

会话表示一个单线程的上下文，用于发送和接收消息。由于会话是单线程的，所以消息是连续的，就是说消息是按照发送的顺序一个一个接收的。会话的好处是它支持事务。如果用户选择了事务支持，会话上下文将保存一组消息，直到事务被提交才发送这些消息。在提交事务之前，用户可以使用回滚操作取消这些消息。一个会话允许用户创建消息生产者来发送消息，创建消息消费者来接收消息。

8. JMS 提供者实现

要使用 Java 消息服务，测试人员必须要有一个 JMS 提供者管理会话和队列。现在既有开

源的提供者也有专有的提供者。

- 开源的提供者包括：Apache ActiveMQ、JBoss 社区所研发的 HornetQ、Joram、Coridan 的 MantaRay、The OpenJMS Group 的 OpenJMS。
- 专有的提供者包括：BEA 的 BEA WebLogic Server JMS、TIBCO Software 的 EMS、GigaSpaces Technologies 的 GigaSpaces、Softwired 2006 的 iBus、IONA Technologies 的 IONA JMS、SeeBeyond 的 IQManager（2005 年 8 月被 Sun Microsystems 并购）、webMethods 的 JMS+ -、my-channels 的 Nirvana、Sonic Software 的 SonicMQ、SwiftMQ 的 SwiftMQ、IBM 的 WebSphere MQ。

8.2　创建 JMS 点对点测试计划

首先需要确保必要的 jar 文件存放于 JMeter 的 lib 目录中。如果不存在，请关闭 JMeter，并将这些 jar 文件复制到 lib 目录中，接着重启 JMeter。目前 JMeter 包含有来自于 Apache Geronimo 的 JMS API，因此测试人员只需要从 JMS 提供商获取实现 JMS 客户端的 jar 文件即可。更多细节请查询对应的 JMS 指导文档。下面以 ActiveMQ 的 JMS 为例。

在本书写作时，最新 ActiveMQ 版本是 5.3.2。测试人员需要将 jar 文件 activemq-all-5.3.2.jar 添加到 classpath 中，例如，将它放到 lib 目录中。

在本节中，作者将会介绍如何创建一个测试计划来测试 JMS 点对点消息解决方案。我们将会创建 5 个并发用户，循环向一个请求队列发送 4 条消息。另外，还会使用一个固定的响应队列来监视响应消息。构建该测试计划，测试人员会用到如下测试组件：线程组（Thread Group）、JMS Point-to-Point 和图形结果（Graph Results）。

1．添加并发用户

首先要做的还是添加线程组（Thread Group），并修改其默认配置，如图 8-3 所示。

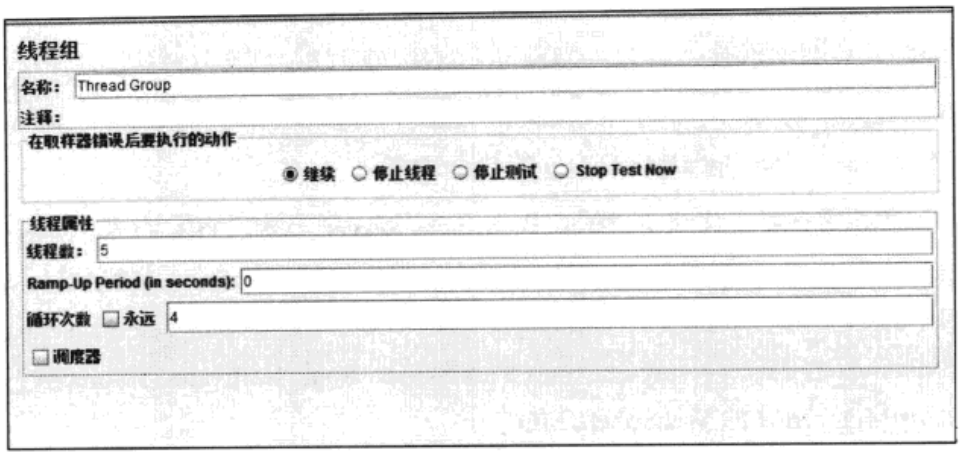


图 8-3　线程组（Thread Group）

2. 添加 JMS Point-to-Point 请求

从添加一个 JMS Point-to-Point 采样器（如图 8-4 所示）开始（Add→Sampler→JMS Point-to-Point），接着在测试树中选中该采样器，并编辑其属性。如表 8-1 所示为一个设置该采样器的例子，该例子被证明在 ActiveMQ 3.0 下能够正常工作。

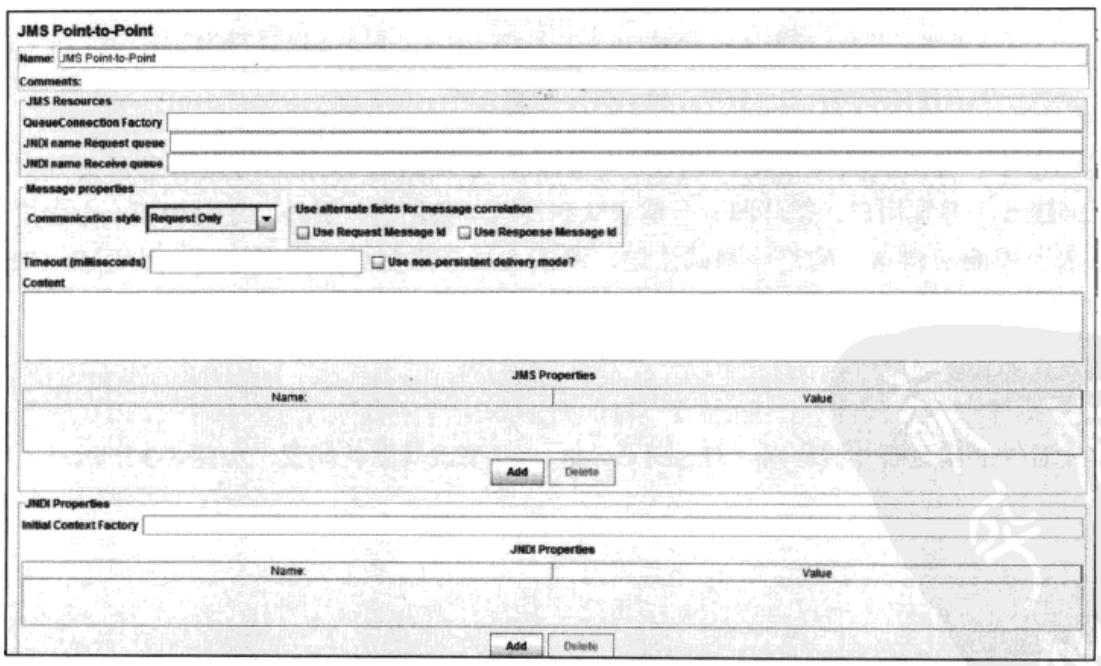


图 8-4　JMS Point-to-Point 采样器

表 8-1　设置 JMS Point-to-Point 采样器的例子

名称	值	描述
JMS Resources		
QueueuConnectionFactory	ConnectionFactory	这是 Connection Factory 在 Active Me 中的默认 JNDI 条目
JNDI Name Request Queue	Q.REQ	等同于定义在 JNDI 属性中的 JNDI 名
JNDI Name Reply Queue	Q.RPL	等同于定义在 JNDI 属性中的 JNDI 名
Message Properties		
Communication Style	Request Response	这意味着测试人员需要至少一个服务器对请求做出响应
Content	test	仅仅是 JMS 消息的内容
JMS Properties		对 Active Me 而言并不需要这一项
JNDI Properties		
InitialContextFactory	org.activemq.jndi.ActiveMQInitialContextFactory	ActiveMQ 的标准 InitialContextFactory
Properties		
providerURL	tcp://localhost:61616	这里定义了 Active Me 消息系统的 URL
queue.Q.REQ	example.A	这里为请求队列定义了一个 JNDI 名 Q.REQ，并指向队列 example.A
queue.Q.RPL	example.B	这里为响应队列定义了一个 JNDI 名 Q.RPL，并指向队列 example. B

3. 添加监听器用于查看/存储测试结果

为测试计划添加的最后一个测试原件就是监听器。该测试原件负责将所有 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，并添加一个图形结果（Graph Results）监听器（Add→Listener→Graph Results），如图 8-5 所示。接下来，需要指定保存测试结果的文件。测试人员既可以在 Filename 输入域中填充，也可以通过“Browse…”按钮来选择目录和文件名。

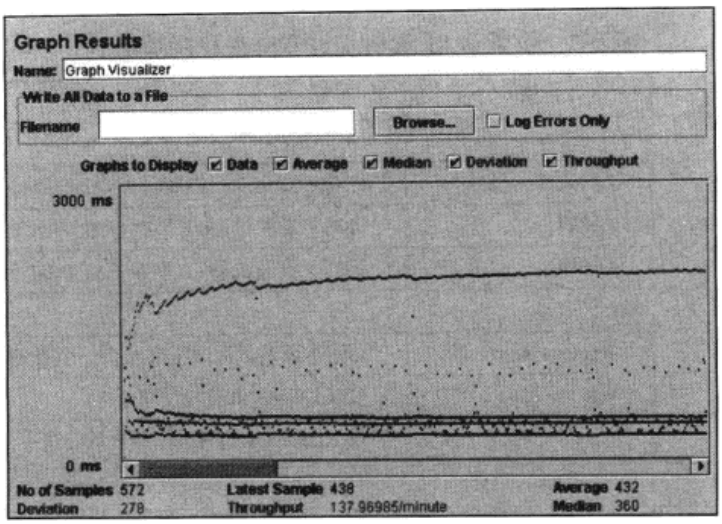


图 8-5 图形结果（Graph Results）

8.3 创建 JMS Topic 测试计划

在这一节中，作者将会介绍如何创建一个测试计划来测试 JMS 提供者。我们将会创建 5 个消费者（Subscriber）和 1 个生产者（Publisher），还会创建 2 个线程组，并将每个线程组的循环次数设置为 10。另外，测试人员需要告诉并发用户执行 2 遍测试计划。因此，总的消息数目是（6 线程）×（1 消息）×（重复 10 遍） = 60 消息。构建该测试计划，测试人员会用到如下测试组件：线程组（Thread Group）、JMS Publisher、JMS Subscriber 和图形结果（Graph Results）。

JMeter 有两种 JMS 采样器，一种使用 JMS 主题（Topic），另外一种使用队列。主题消息通常采用发布/消费模式。主题消息常被用于有一个生产者发布消息，而有多个消费者读取消息的场景之下。队列消息常被用于发送方期待响应的场景之下。在 HTTP 协议中，一个用户简单地发送一条消息，并获得一个响应。而消息系统则不同，它既可以工作在同步模式下，也可以工作在异步模式下。JMS 采样器需要用到对应 JMS 实现的 jar 文件，例如，测试人员可以从 Apache ActiveMQ 获取。

1. 添加并发用户

首先要做的还是添加线程组（Thread Group），并修改其默认配置，如图 8-6 所示。

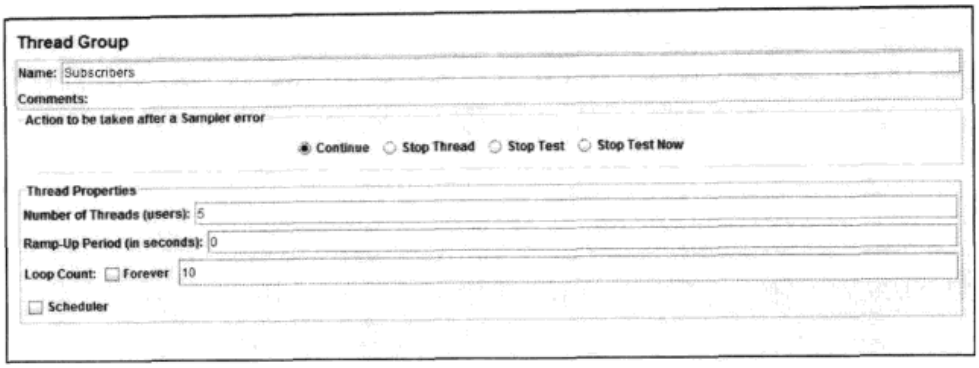


图 8-6　线程组（Thread Group）——Subscribers

接下来还要添加第二个线程组，如图 8-7 所示。

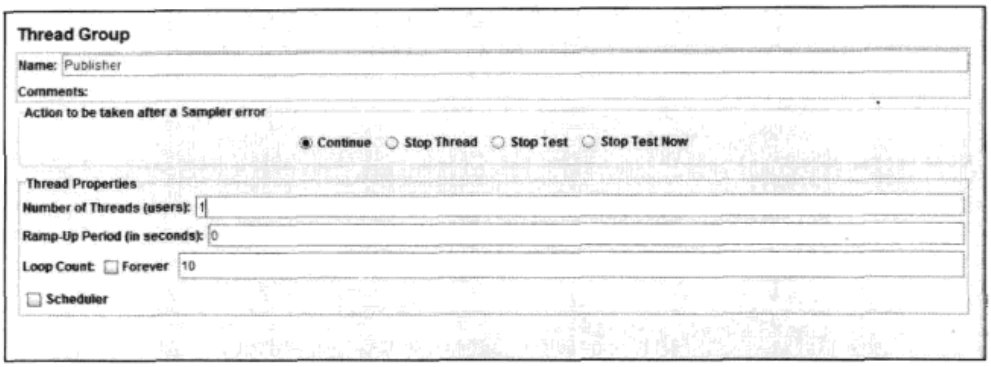


图 8-7　线程组（Thread Group）——Publisher

2. 添加 JMS 生产者和消费者

确保必要的 jar 文件存放于 JMeter 的 lib 目录中。如果不存在，请关闭 JMeter，并将这些 jar 文件复制到 lib 目录中，接着重启 JMeter。

从添加一个 JMS Subscriber 采样器开始（Add→Sampler→JMS Subscriber），接着在测试树中选中该采样器，并编辑其属性，如图 8-8 所示。

（1）在名称域（Name）中输入“JMS Subscriber”。

（2）如果 JMS 提供者使用 jndi.properties 文件，那么就请选中对应选项。

（3）输入 JNDI Initial Context Factory 类的名称。

（4）输入 JMS 提供者的 URL，就是 JNDI 服务器的 URL（如果存在的话）。

（5）输入 Connection Factory 的名称。请参考 JMS 提供者的相关文档，以便寻求更多信息。

（6）输入消息主题（Topic）的名称。

（7）如果 JMS 提供者要求校验，请选中“Required”单选按钮，并输入用户名和密码。例如，Orion JMS 要求校验，而 ActiveMQ 和 MQSeries 不要求校验。

（8）在“Number of samples to aggregate”域中输入 10。由于性能方面的原因，采样器会聚集消息，原因就在于短小的消息到达速度非常快。如果采样器不将消息聚合起来，那么 JMeter 就无法保证正常工作。

（9）如果测试人员想要读取响应，就选中“Read Response”复选框。

（10）这里有两种消费者的客户端实现，用户可以分别加以尝试。

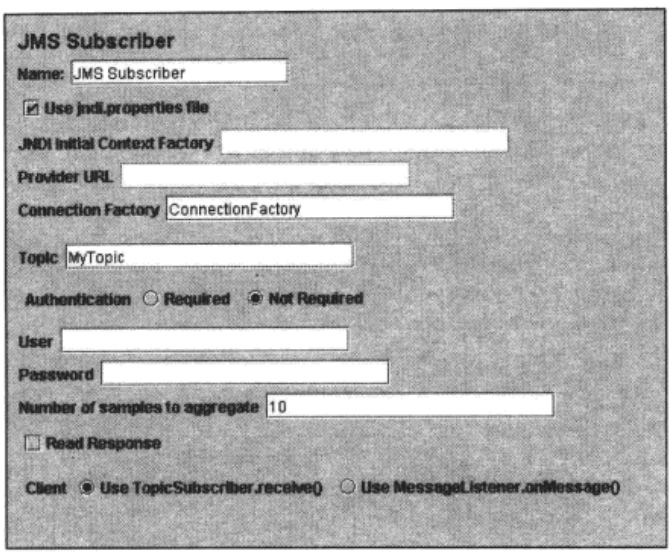


图 8-8　JMS 消费者（JMS Subscriber）采样器

接着添加一个 JMS Publisher 采样器（Add→Sampler→JMS Publisher），然后在测试树中选中该采样器，并编辑其属性，如图 8-9 所示。

（1）在名称域（Name）中输入“JMS Publisher”。

（2）如果 JMS 提供者使用 jndi.properties 文件，那么就请选中对应选项。

（3）输入 JMS 提供者的 URL，就是 JNDI 服务器的 URL（如果存在的话）。

（4）输入 Connection Factory 的名称。请参考 JMS 提供者的相关文档，以便寻求更多信息。

（5）输入消息主题（Topic）的名称。

（6）如果 JMS 提供者要求校验，请选中“Required”单选按钮，并输入用户名和密码。例如，Orion JMS 要求校验，而 ActiveMQ 和 MQSeries 不要求校验。

（7）在“Number of samples to aggregate”域中输入 10。由于性能方面的原因，采样器会聚集消息，原因就在于短小的消息到达速度非常快。如果采样器不将消息聚合起来，那么 JMeter 就无法保证正常工作。

（8）选择合适的配置以利于消息发布，可选方式包括：From file、Random File、Textarea。如果测试人员希望采样器随机地选择消息内容，请将所有可选消息放在一个目录之下，并通过“Browse…”按钮来选中该目录。

（9）选择消息类型（Text Message、Object Message），如果消息使用 Object 格式，一定要确保产生的消息是正确的。

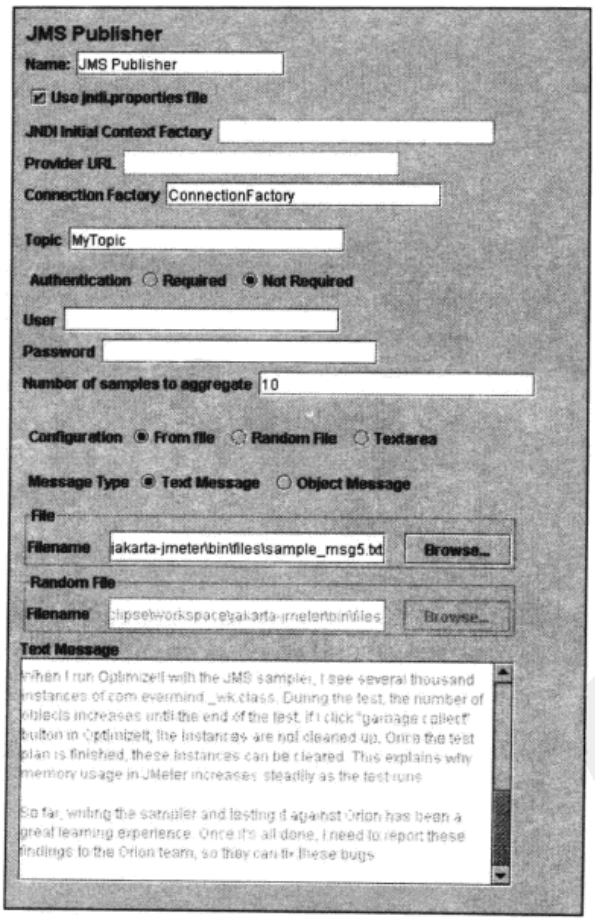


图 8-9　JMS 生产者（JMS Publisher ）采样器

3．添加监听器用于查看/存储测试结果

为测试计划添加的最后一个测试原件就是监听器。该测试原件负责将所有 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，并添加一个图形结果（Graph Results）监听器（Add→Listener→Graph Results），如图 8-10 所示。接下来，需要指定保存测试结果的文件。测试人员既可以在 Filename 输入域中填充，也可以通过“Browse…”按钮来选择目录和文件名。

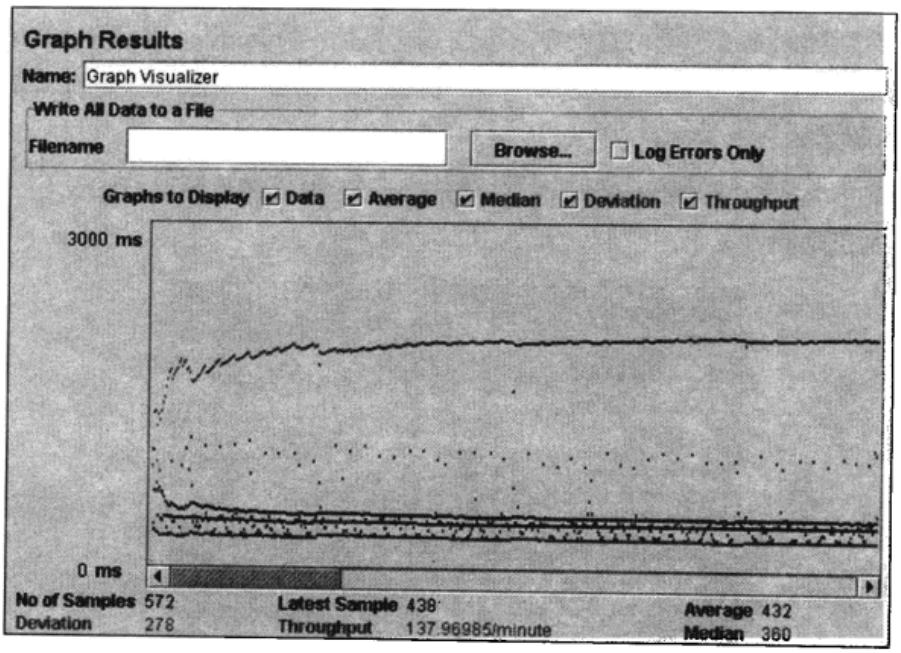


图 8-10 图形结果（Graph Results）

8.4 本章小结

本章首先对 JMS 的基本概念做了一个介绍，为后续开发 JMS 性能测试脚本打下了基础；接着介绍了如何创建 JMS 点对点测试计划及 JMS Topic 测试计划。

第 9 章

服务器监控测试脚本开发

9.1 创建监控测试计划

在本节中，作者将会介绍如何创建一个测试计划来监控 Web 服务器。服务器监控功能对于压力测试和系统管理都非常有用。在压力测试工作中，服务器监控功能可以提供关于服务器性能的额外信息，这能帮助 JMeter 用户直观地看到服务器性能与系统响应时长之间的关系。作为一个系统管理工具，服务器监控功能可以通过一个控制台监控多个服务器。JMeter 服务器监控功能是专门针对 Tomcat 5 的 status servlet 设计的。从理论上来说，任何支持 JMX（Java Management Extension）的 servlet 容器都能部署该 servlet，以提供同样的监控信息。

服务器监控功能也可以与其他 servlet 或者 EJB 容器配合，Tomcat 的 status servlet 可以配合其他容器监控服务器存储统计信息，而无须做任何修改。要获取服务器线程的统计信息，测试人员需要改变 MbeanServer lookup，以便获得正确的 Mbeans。

1. 添加并发用户

首先要做的还是添加线程组（Thread Group），并修改其默认配置，如图 9-1 所示。

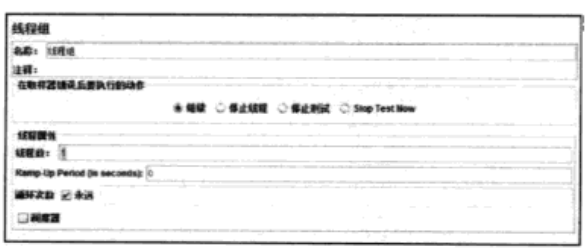


图 9-1 线程组（Thread Group）

需要注意的是：

（1）线程数应该始终设置为 1，如果为单个服务器创建多个监控线程，是非常坏的一件事，这会为服务器带来很大压力。

（2）应该将循环次数设置为“永远”，以便产生足够多的采样，来动态监控服务器的工作状态。

2. 添加 HTTP 授权管理器

为线程组添加 HTTP 授权管理器（HTTP Authorization Manager）（Add→Config Element→HTTP Authorization Manager）。输入 Web 服务器的用户名和密码。另外需要注意的是，服务器监控功能只能针对 Tomcat 5 5.0.19 及其以上版本。关于如何配置 Tomcat，请参考 Tomcat 5 的官方文档。

（1）将 Base URL 保留为空。

（2）输入用户名。

（3）输入密码。

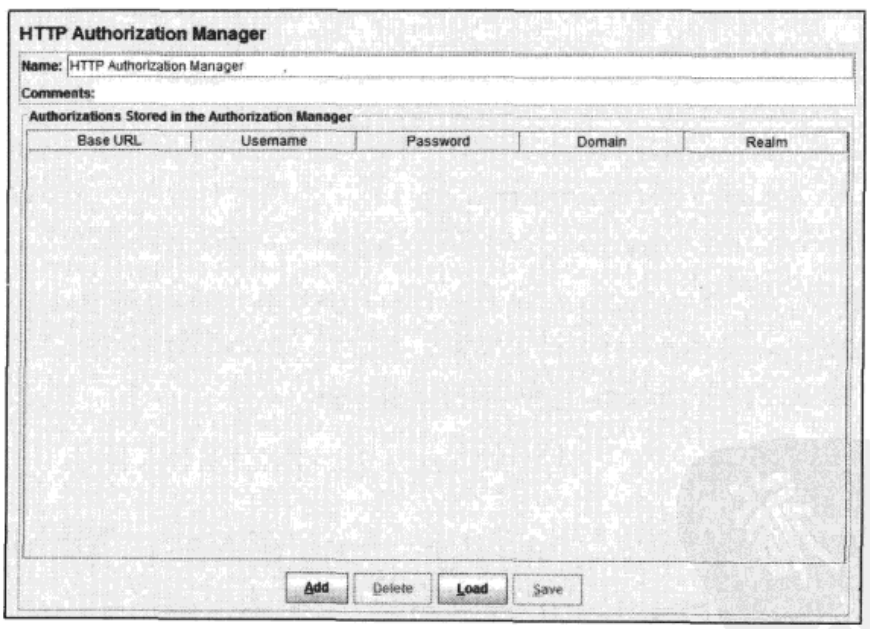


图 9-2　HTTP 授权管理器（HTTP Authorization Manager）

3. 添加 HTTP 请求

为线程组添加一个 HTTP 请求（HTTP Request）（Add→Sampler→HTTP Request），接着在测试树中选中该采样器，并编辑其属性，如图 9-3 所示。

（1）将名称域（Name）改为“Server Status”。

（2）输入 IP 地址或者主机名。

（3）输入端口号。

（4）如果测试人员使用 Tomcat，请将 Path 域设置为“/manager/status”。

（5）添加一个请求参数名为“XML”（大写），其值为“true”（小写）。

（6）选中采样器底部的“Use as Monitor”复选框。

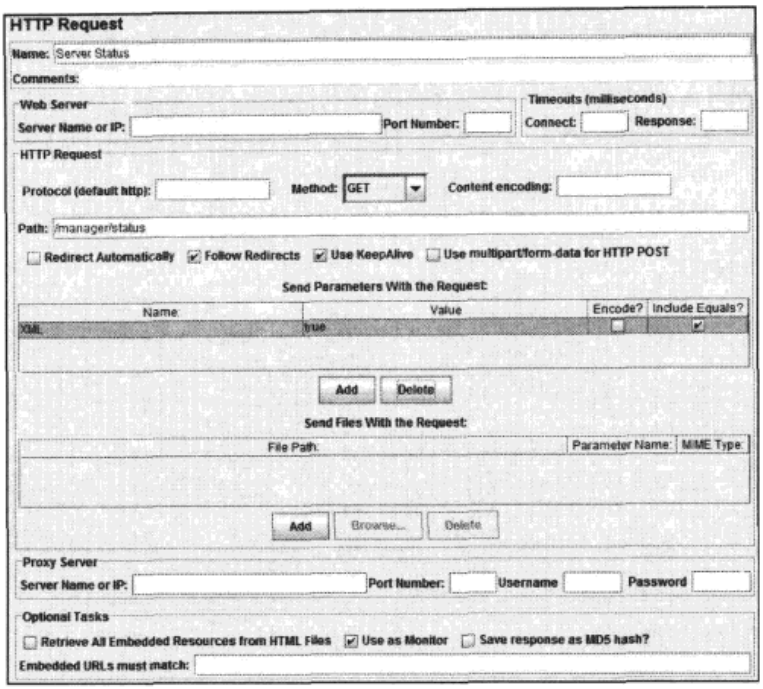


图 9-3　HTTP 请求（HTTP Request）

4．添加固定时长定时器

为线程组添加一个定时器（Add→Timer→Constant Timer），如图 9-4 所示。在“Thread Delay”域中输入 5000。通常，采样间隔小于 5s，会加大服务器的负载。请根据实际情况，调整采样间隔时间。

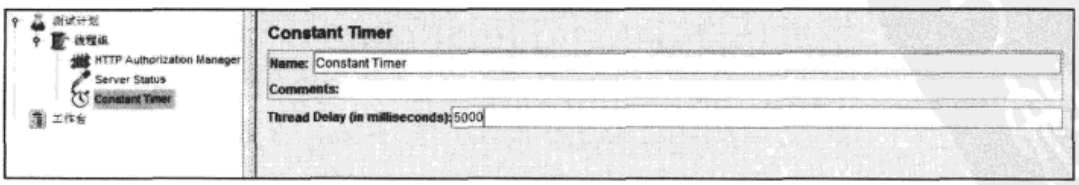


图 9-4　固定时长定时器（Constant Timer）

5. 添加一个监听器来存储测试结果

如果测试人员想保存来自于服务器的原始结果，请添加一个简单的数据监听器。选中线程组，并添加一个 Simple Data Writer 监听器（Add→Listener→Simple Data Writer），如图 9-5 所示。接下来，需要指明输出文件的目录和文件名。测试人员既可以在 Filename 输入域中填充，也可以通过“Browse”按钮来选择目录和文件名。

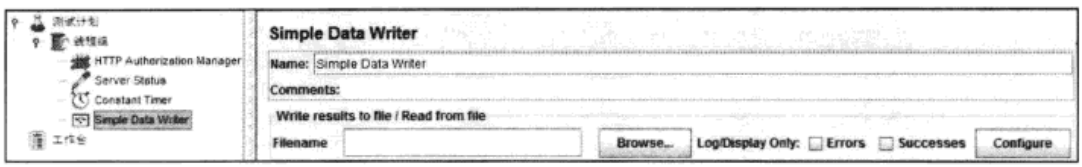


图 9-5 Simple Data Writer 监听器

6. 添加监视器结果

选中测试计划，并添加一个监视器结果（Monitor Results）（Add→Listener→Monitor Results）。

默认情况下，监视器结果会使用来自于采样结果的第一个连接器的结果。用户可以使用“Connection prefix”域来选择一个不同的连接器。如果指定了连接器，那么 JMeter 会寻找第一个匹配 prefix 的连接器。如果没有找到匹配的连接器，就会使用第一个连接器。

监视器结果有两个选项卡：第一个是“Health”选项卡，它会展示监视器结果收到的最近一个采样的状态数据，如图 9-6 所示；第二个是“Performance”选项卡，它会动态地展示服务器性能的变化，如图 9-7 所示。

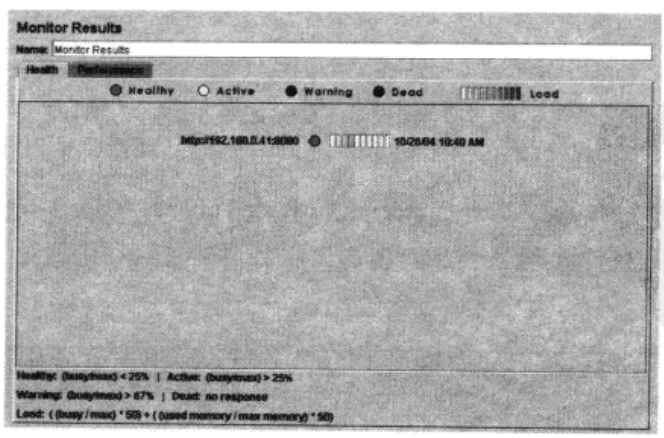


图 9-6 监视器结果（Monitor Results）——“Health”选项卡

这里介绍一下负载是如何计算的。通常，一台服务器在内存耗尽或者线程数达到最大值时会

发生宕机。对于 Tomcat 5 而言，一旦线程数达到最大值，新到的请求就会被放置在一个队列之中，直到有线程被释放为止。目前负载的计算，线程和内存的使用率各占 50%的比重。由于不同容器对线程的依赖程度不同，因此 50/50 的比重是一个保守的计算方法。如果某个容器有很高效的线程管理方式，那么在线程数达到最大值时，测试人员可能看不到服务器性能下降，但是观察内存使用率就一定能看到冲击。

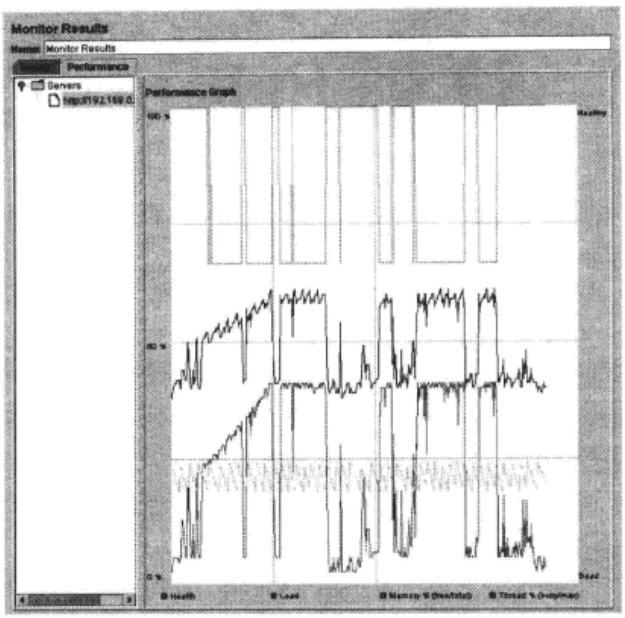


图 9-7　监视器结果（Monitor Results）——“Performance”选项卡

在“Performance”选项卡中有多条变化的曲线。空闲内存曲线可以表明当前分配内存块中有多少空闲内存。Tomcat 5 会返回最大内存值，但不会在图中体现。在一个良好的系统环境下，服务器应该永远也不会达到最大内存值。

请注意在统计图的两侧都有标注。左侧的是百分比，右侧是 Dead/Healthy。如果内存曲线忽上忽下，表明内存分配存在问题。在这种情况下，测试人员可以使用 Borland OptimizeIt 或者 JProbe 对系统内存分配进行优化。测试人员期望的应该是负载很均衡，包括内存和线程。任何古怪的曲线通常都意味着系统性能差，或者存在某种类型的 bug。

9.2　本章小结

本章主要介绍了如何使用 JMeter 来监控 Web 服务器性能，包括如何创建监控测试计划。本章内容虽然不多，但却很重要。在实际工作中，读者朋友遇到的 Web 服务器可能不采用 Tomcat 5，但是监控服务器的基本方法和思想是一致的，需要根据实际需求，灵活地选择服务器监控工具。

第 10 章

详解JMeter测试元件

各类测试元件是创建 JMeter 测试脚本的基石，了解它们的使用方法/适用范围，无疑能帮助测试人员更好地使用 JMeter 来完成各项性能测试工作。本章详解了各类 JMeter 测试元件，包括监听器、逻辑控制器、配置元件、定时器、前置处理器、后置处理器、采样器等，这几乎囊括了所有 JMeter 测试元件。其中部分测试元件比较生僻，不会被经常使用，读者朋友可以选择跳过这部分内容，仅关注常用的测试元件。

10.1 详解 JMeter 监听器

监听器（Listeners）是一种展示采样结果的测试元件。采样结果可以通过树、表格、图片加以展示，或者简单地写入某个结果文件之中。假如用户要查看采样结果中的服务器响应，可以在测试计划中加入察看结果树“View Results Tree”或者用表格查看结果“View Results in Table”这两种监听器。如果用户要通过绘图方式查看响应时长，则可以使用图形结果（Graph Results）、Spline Results、Distribution Graph 这 3 种监听器。要获得最详尽的介绍，请参考 JMeter 帮助文档。

不同的监听器通过不同的方式展示服务器响应信息，但是它们都将同样的原始数据记录到某个输出文件中（假如其中某个监听器指定了一个输出文件）。

“Configure”按钮可以被用来配置哪些数据域会被写入到结果文件之中，以及结果文件的格式 CSV 或者 XML。与 XML 文件相比，CSV 文件占用的磁盘空间要小得多，因此当采样次数很多时，请使用 CSV 格式。

如果测试人员只关心测试计划中的部分采样器，可以将监听器作为采样器的子测试元件。另外，测试人员还可以使用简单控制器将一系列采样器组合起来，并为简单控制器添加一个监听器。

多个采样器可以使用相同的结果文件，前提是它们有相同的配置。

1. 监听器默认配置

监听器默认保存哪些数据域，可以在 jmeter.properties（或者 user.properties）文件中通过属性定义。这些属性会作为监听器配置对话框的默认设置，还会影响命令行-l 标识指定的日志文件（通常针对非 GUI 模式）。

要改变默认格式，请找到 jmeter.properties 文件中的如下行：

```
jmeter.save.saveservice.output_format=
```

如何保存采样信息是可以配置的。要获取最全面的信息，请选择“xml”格式，并在测试计划中选中“Functional Test Mode”复选框。如果复选框项没有被选中，那么默认保存的数据中会包含时间戳、数据类型、线程名、标签、响应时长、消息、编码，以及成功标志。如果选中该复选框，所有信息，包含全部的响应数据都会被记录下来。

下面的例子中，展示了如何设置保存数据的格式，其中使用（|）来分隔。另外还展示了该保存格式对应的输出结果。

```
timeStamp|time|label|responseCode|threadName|dataType|success|failureMes
sage
02/06/03 08:21:42|1187|Home|200|Thread Group-1|text|true|
02/06/03 08:21:42|47|Login|200|Thread Group-1|text|false|Test Failed:
expected to contain: password etc.
```

其他需要在 jmeter.properties 中设置的属性如下所示。在这个例子中使用的输出格式是 CSV，通常在这种格式下使用逗号来分隔数据。但是这里设置的默认分隔符号是“|”，所以这个例子中的 CSV 标记有点特殊。

```
jmeter.save.saveservice.output_format=csv
jmeter.save.saveservice.assertion_results_failure_message=true
jmeter.save.saveservice.default_delimiter=|
```

所有影响结果文件的属性如下：

```
#---------------------------------------------------------------------------
----
# Results file configuration
#---------------------------------------------------------------------------
----

# This section helps determine how result data will be saved.
# The commented out values are the defaults.
```

```
# legitimate values: xml, csv, db.  Only xml and csv are currently supported.
#jmeter.save.saveservice.output_format=xml


# true when field should be saved; false otherwise

# assertion_results_failure_message only affects CSV output
#jmeter.save.saveservice.assertion_results_failure_message=false
#
#jmeter.save.saveservice.data_type=true
#jmeter.save.saveservice.label=true
#jmeter.save.saveservice.response_code=true
# response_data is not currently supported for CSV output
#jmeter.save.saveservice.response_data=false
# Save ResponseData for failed samples
#jmeter.save.saveservice.response_data.on_error=false
#jmeter.save.saveservice.response_message=true
#jmeter.save.saveservice.successful=true
#jmeter.save.saveservice.thread_name=true
#jmeter.save.saveservice.time=true
#jmeter.save.saveservice.subresults=true
#jmeter.save.saveservice.assertions=true
#jmeter.save.saveservice.latency=true
#jmeter.save.saveservice.samplerData=false
#jmeter.save.saveservice.responseHeaders=false
#jmeter.save.saveservice.requestHeaders=false
#jmeter.save.saveservice.encoding=false
#jmeter.save.saveservice.bytes=true
#jmeter.save.saveservice.url=false
#jmeter.save.saveservice.filename=false
#jmeter.save.saveservice.hostname=false
#jmeter.save.saveservice.thread_counts=false
#jmeter.save.saveservice.sample_count=false
#jmeter.save.saveservice.idle_time=false

# Timestamp format
# legitimate values: none, ms, or a format suitable for SimpleDateFormat
#jmeter.save.saveservice.timestamp_format=ms
```

```
#jmeter.save.saveservice.timestamp_format=MM/dd/yy HH:mm:ss

# Put the start time stamp in logs instead of the end
sampleresult.timestamp.start=true


# legitimate values: none, first, all
#jmeter.save.saveservice.assertion_results=none

# For use with Comma-separated value (CSV) files or other formats
# where the fields' values are separated by specified delimiters.
# Default:
#jmeter.save.saveservice.default_delimiter=,
# For TAB, since JMeter 2.3 one can use:
#jmeter.save.saveservice.default_delimiter=\t

#jmeter.save.saveservice.print_field_names=false

# Optional list of JMeter variable names whose values are to be saved in the
result data files.
# Use commas to separate the names. For example:
#sample_variables=SESSION_ID,REFERENCE
# N.B. The current implementation saves the values in XML as attributes,
# so the names must be valid XML names.
# Versions of JMeter after 2.3.2 send the variable to all servers
# to ensure that the correct data is available at the client.
# Optional xml processing instruction for line 2 of the file:
#jmeter.save.saveservice.xml_pi=<?xml-stylesheet type="text/xsl"
href="sample.xsl"?>
```

如果将 timestamp_format 设置为非 ms 格式，那么可能导致 JMeter 在后续读取结果文件时无法识别该值。

I）sample_variables 属性

JMeter 2.3.1 及其后续版本允许用户使用属性 sample_variables 来定义一些补充 JMeter 变量，这些变量的值将会和采样结果一起被保存到 JTL 文件中。这些变量会作为补充列写入到 CSV 文件中，或者作为补充属性写入到 XML 文件中。请参考上面的列表。

2）配置采样结果的保存格式

监听器可以使用配置对话框（如图 10-1 所示）来设置将哪些数据保存到结果文件中（JTL）。以 CSV 结尾的选项只影响 CSV 格式的结果文件；以 XML 结尾的选项只影响 XML 格式的结果文件。目前 CSV 格式的结果文件，不能记录包含换行符的数据。

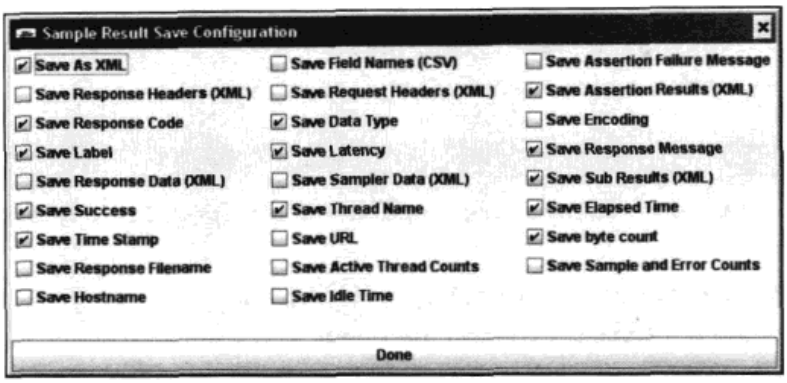


图 10-1　监听器配置对话框

需要注意，cookies、method 和查询语句会作为“Sampler Data”的一部分加以保存。

2. 非 GUI 模式运行测试

当 JMeter 以非 GUI 模式运行时，可以使用-l 标签为测试创建一个监听器（位于测试树的最顶层）。该监听器是测试计划新增的监听器，并不影响测试计划中原有的监听器。可以通过 jmeter.properties 文件中定义的属性来对该监听器进行配置，请参见前面的内容。

下面的例子演示了如何通过命令行来设置结果文件和日志文件：

```
jmeter -n -t testplan.jmx -l testplan_01.jtl -j testplan_01.log
jmeter -n -t testplan.jmx -l testplan_02.jtl -j testplan_02.log
```

需要注意，JMeter 日志消息默认会写到 jmeter.log 文件之中，每次测试运行都会重新创建该文件，因此假如测试人员要保存每次运行的日志文件，就需要用到-j 选项。-j 选项是在 JMeter 2.3 版本之后加入的。

JMeter 2.3.1 及其以后版本，支持在日志文件名中使用变量。如果文件名中包含成对的单引号，那么文件名就会以 Simple Date Format 格式来处理，而且文件名会携带上当前时间。例如，log_file='jmeter_'yyyyMMddHHmmss'.tmp'，它能为每一个日志文件生成唯一的文件名。

3. 监听器资源占用

如果监听的采样器数目很多，那么监听器就会占用大量内存。目前大多数监听器会保存每一次采样的数据备份，例如：

- Simple Data Writer。
- BeanShell/BSF Listener。
- Mailer Visualizer。
- Monitor Results。
- Summary Report。

而下面这些采样器不再保存所有单次采样的副本，因为采样周期相同的采样会被聚合起来。如此一来，监听器需要的内存将会减少，特别是在大多数采样只耗费一秒或者两秒的情况下。

- Aggregate Report。
- Aggregate Graph。
- Distribution Graph。

要减少监听器占用的内存，请选择 Simple Data Writer，并使用 CSV 格式的结果文件。

4. CSV 记录格式

CSV 记录格式依赖于配置监听器过程中选择的数据域，只有指定的数据会被记录到结果文件之中。列的顺序在结果文件中是固定的，如下所示：

- `timeStamp - in milliseconds since 1/1/1970`
- `elapsed - in milliseconds`
- `label - sampler label`
- `responseCode - e.g. 200, 404`
- `responseMessage - e.g. OK`
- `threadName`
- `dataType - e.g. text`
- `success - true or false`
- `failureMessage - if any`

- bytes - number of bytes in the sample
- grpThreads - number of active threads in this thread group
- allThreads - total number of active threads in all groups
- URL
- Filename - if Save Response to File was used
- latency - time to first response
- encoding
- SampleCount - number of samples (1, unless multiple samples are aggregated)
- ErrorCount - number of errors (0 or 1, unless multiple samples are aggregated)
- Hostname where the sample was generated
- IdleTime - number of milliseconds of 'Idle' time (normally 0)
- Variables, if specified

5. XML 记录格式

XML（2.1）的格式如下：

```
<?xml version="1.0" encoding="UTF-8"?>
<testResults version="1.2">

-- HTTP Sample, with nested samples
<httpSample t="1392" lt="351" ts="1144371014619" s="true"
    lb="HTTP Request" rc="200" rm="OK"
    tn="Listen 1-1" dt="text" de="iso-8859-1" by="12407">
 <httpSample t="170" lt="170" ts="1144371015471" s="true"
      lb="http://www.apache.org/style/style.css" rc="200" rm="OK"
      tn="Listen 1-1" dt="text" de="ISO-8859-1" by="1002">
   <responseHeader class="java.lang.String">HTTP/1.1 200 OK
Date: Fri, 07 Apr 2006 00:50:14 GMT
...
Content-Type: text/css
```

```
</responseHeader>
    <requestHeader class="java.lang.String">MyHeader:
    MyValue</requestHeader>
    <responseData class="java.lang.String">body, td, th {
    font-size: 95%;
    font-family: Arial, Geneva, Helvetica, sans-serif;
    color: black;
    background-color: white;
}
...
</responseData>
    <cookies class="java.lang.String"></cookies>
    <method class="java.lang.String">GET</method>
    <queryString class="java.lang.String"></queryString>
    <url>http://www.apache.org/style/style.css</url>
  </httpSample>
  <httpSample t="200" lt="180" ts="1144371015641" s="true"
     lb="http://www.apache.org/images/asf_logo_wide.gif"
     rc="200" rm="OK" tn="Listen 1-1" dt="bin" de="ISO-8859-1" by="5866">
    <responseHeader class="java.lang.String">HTTP/1.1 200 OK
Date: Fri, 07 Apr 2006 00:50:14 GMT
...
Content-Type: image/gif
</responseHeader>
    <requestHeader class="java.lang.String">MyHeader:
    MyValue</requestHeader>
    <responseData
    class="java.lang.String">http://www.apache.org/asf.gif</responseData>
      <responseFile class="java.lang.String">Mixed1.html</responseFile>
    <cookies class="java.lang.String"></cookies>
    <method class="java.lang.String">GET</method>
    <queryString class="java.lang.String"></queryString>
    <url>http://www.apache.org/asf.gif</url>
  </httpSample>
  <responseHeader class="java.lang.String">HTTP/1.1 200 OK
Date: Fri, 07 Apr 2006 00:50:13 GMT
...
Content-Type: text/html; charset=ISO-8859-1
</responseHeader>
```

```
  <requestHeader class="java.lang.String">MyHeader:
  MyValue</requestHeader>
  <responseData class="java.lang.String">
...
&lt;html&gt;
 &lt;head&gt;
...
 &lt;/head&gt;
 &lt;body&gt;
...
 &lt;/body&gt;
&lt;/html&gt;
</responseData>
  <cookies class="java.lang.String"></cookies>
  <method class="java.lang.String">GET</method>
  <queryString class="java.lang.String"></queryString>
  <url>http://www.apache.org/</url>
</httpSample>

-- nonHTTPP Sample

<sample t="0" lt="0" ts="1144372616082" s="true" lb="Example Sampler"
    rc="200" rm="OK" tn="Listen 1-1" dt="text" de="ISO-8859-1" by="10">
  <responseHeader class="java.lang.String"></responseHeader>
  <requestHeader class="java.lang.String"></requestHeader>
  <responseData class="java.lang.String">Listen 1-1</responseData>
  <responseFile class="java.lang.String">Mixed2.unknown</responseFile>
  <samplerData class="java.lang.String">ssssss</samplerData>
</sample>

</testResults>
```

请注意 XML 记录中，采样节点的名称可以是“sample”或者“httpSample”。

对于 JTL 文件而言，XML（2.1）和 XML（2.2）没有区别。

6. 采样属性

采样属性的含义如表 10-1 所示。

表 10-1　采样属性

属性	含义
by	Bytes
de	Data encoding
dt	Data type
ec	Error count (0 or 1, unless multiple samples are aggregated)
hn	Hostname where the sample was generated
it	Idle Time = time not spent sampling (milliseconds) (generally 0)
lb	Label
lt	Latency = time to initial response (milliseconds) - not all samplers support this
na	Number of active threads for all thread groups
ng	Number of active threads in this group
rc	Response Code (e.g. 200)
rm	Response Message (e.g. OK)
s	Success flag (true/false)
sc	Sample count (1, unless multiple samples are aggregated)
t	Elapsed time (milliseconds)
tn	Thread Name
ts	timeStamp (milliseconds since midnight Jan 1, 1970 UTC)
varname	Value of the named variable (versions of JMeter after 2.3.1)

JMeter 2.2 和 2.1.1 版本，会将 Response Code 保存为“rs”，而读取结果文件时会寻找“rc”。这当然会存在问题，因此在后续版本对此做出了修正，Response Code 统一保存为“rc”，读取时兼容“rc”和“rs”。

7．保存响应数据

在必要时，服务器的响应数据可以保存至 XML 记录文件中。当然，这可能导致结果文件变得非常大，因此需要对文本内容进行编码，不过仍然是正确的 XML 格式。另外，响应中的图片不会被保存下来。

还有一种解决办法，那就是使用后置处理器 Save_Responses_to_a_file。它会为作用域范围内的每一个采样器产生一个结果文件，文件名同采样器的名称。文件中的数据，在必要时可以被 JMeter 回读。

8．加载（读取）响应数据

要查看某个历史结果文件，测试人员可以通过“Browse...”按钮来选择一个结果文件。如果有必要的话，可以创建一个空测试计划，只在其中包含一个合适的监听器。

测试结果数据可以从 XML 或者 CSV 格式的结果文件中读取。当从 CSV 结果文件中读取数据时，头部（如果存在）会被用来判断文件中存在哪些数据域。要正确解读一个缺少头部的 CSV 文件，需要正确设置 JMeter 属性。

JMeter 2.3.2 及其以前版本，在从结果文件加载数据前会先清除监听器中已有的数据。在 JMeter 2.3.2 以后的版本中，不再清除数据，而是将结果文件中的数据与监听器中已有的数据进行合并。

9. 保存监听器 GUI 的数据

JMeter 可以将任何监听器保存为一个 PNG 文件。用户可以在测试树中选中某个监听器后，选择“Edit”→“Save As Image”命令，如图 10-2 所示。接着将会出现一个文件对话框，在对话框中输入期望的文件名后，保存监听器。

以表格方式输出结果的监听器，还支持复制和粘贴。在表格中选中想要的单元格后，使用操作系统的复制快捷键（通常为“Ctrl+C”），如此数据就会被复制到剪贴板中，接下来测试人员就可以将数据粘贴到其他应用程序中去（如 Excel 或者记事本）。

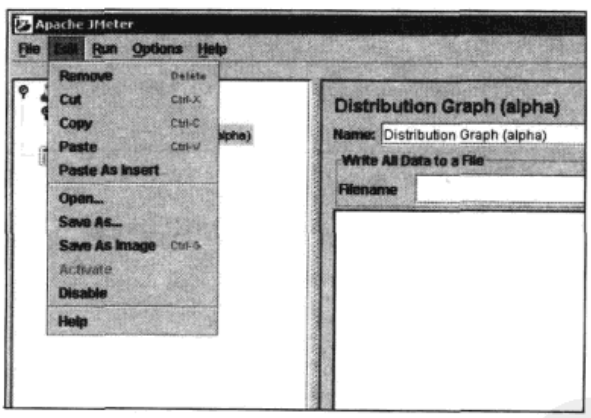


图 10-2 保存监听器 GUI 的数据

10.2 详解 JMeter 逻辑控制器

JMeter 使用逻辑控制器来决定采样器的处理顺序。

1. 简单控制器（Simple Controller）

测试人员可以使用简单逻辑控制器来组合采样器及其他逻辑控制器。不同于其他逻辑控制器，简单逻辑控制器不会提供除存储设备之外的其他功能，如图 10-3 所示。

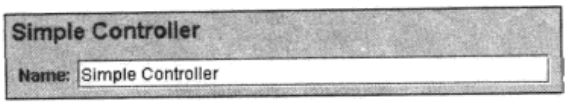


图 10-3　简单逻辑控制器

Name 属性为该元件的描述性名称，用于在测试树中标识该元件，该属性不是必需的。

简单控制器范例如图 10-4 所示。

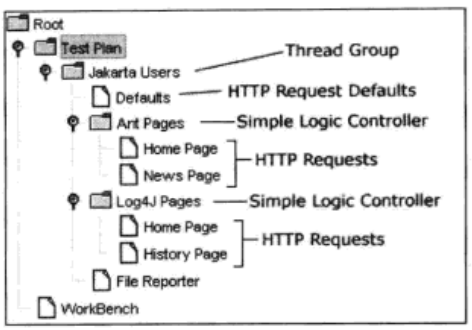


图 10-4　简单逻辑控制器使用范例

以图 10-4 所示的测试计划为例。在这个例子中，我们创建了一个测试计划，其中会发送两个 Ant HTTP 请求及两个 Log4J HTTP 请求。我们将 Ant 和 Log4J 的 HTTP 请求组合起来，放置到简单逻辑控制器之中。请记住，简单控制器不会影响 JMeter 对逻辑控制器（放在简单控制器之下）的处理。因此，在这个例子中，JMeter 会以如下顺序发送请求：Ant Home Page、Ant News Page、Log4J Home Page、Log4J History Page。请注意，File Reporter 被用于在当前目录中保存测试结果，存储文件名为“simple-test.dat”。本例的测试计划，可以从如下地址下载：http://jakarta.apache.org/jmeter/demos/SimpleTestPlan.jmx。

2. 循环控制器（Loop Controller）

如果将采样器或者逻辑控制器放到循环控制器之下，那么 JMeter 会将它们循环执行数次，循环次数在循环控制器中设定，如图 10-5 所示。例如，假如测试人员将一个 HTTP 采样器放入循环控制器之下，循环变量值为 2，并将线程组（Thread Group）的循环次数设置为 3，JMeter 总共会发送 2×3 = 6 HTTP 请求。

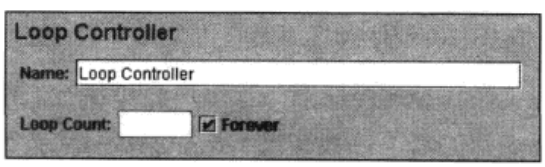


图 10-5　循环控制器（Loop Controller）

参数如表 10-2 所示。

表 10-2 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Loop Count	循环次数，一次测试执行过程中，循环控制器下属子测试元件被执行的次数 特别注意：集成在线程组中的循环控制器，它的表现会有所不同。除非选中“Forever”复选框，否则在达到设定循环次数后，它就会终止测试	是，除非选中了“Forever”选项

循环控制器范例如下：

在这一例子中，我们创建了一个测试计划，发送一个特定 HTTP 请求仅一次，而发送其他 HTTP 请求 5 次，如图 10-6 所示。

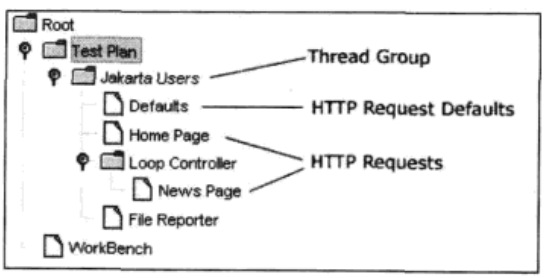


图 10-6 循环控制器（Loop Controller）使用范例

将线程组设置为只有一个单线程，而循环次数也为 1。在这个例子中，不使用线程组控制循环，而是使用循环控制器。可以看到，我们为线程组添加了一个 HTTP 请求，为循环控制器也添加了另外一个 HTTP 请求。我们将循环控制器的循环变量配置为 5。

JMeter 会以如下顺序发送请求：Home Page、News Page、News Page、News Page、News Page 和 News Page。本例的测试计划，可以从如下地址下载：http://jakarta.apache.org/jmeter/demos/LoopTestPlan.jmx。

3. 仅一次控制器（Once Only Controller）

仅一次控制器会告诉 JMeter 只执行其下的控制器一遍，接下来测试计划中的循环执行会跳过该控制器下的所有请求，如图 10-7 所示。

仅一次控制器会在任何父循环控制器的第一次循环之中执行。因此，如果将仅一次控制器放在一个循环控制器（循环 5 次）之下，那么仅一次控制器只会在循环控制器的第一次循环时执行（举例，每 5 次）。请注意，将仅一次控制器放在线程组之下，该控制器的表现也一样（每

次测试只执行一遍），但是现在使用仅一次控制器有了更大的灵活性。

对于要求登录的测试，可以考虑将登录请求放到仅一次控制器中，因为每一个并发线程都只需登录一次来建立会话。

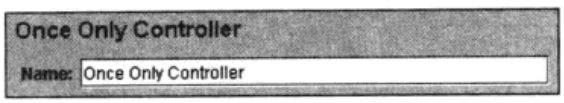


图 10-7　仅一次控制器（Once Only Controller）

仅一次控制器范例如下：

在这一例子中，我们会创建一个测试计划，其中有两个线程会发送 HTTP 请求，如图 10-8 所示。每个线程都会发送一个请求到 Home 页面，紧跟着发送 3 个请求到 Bug 页面。尽管我们将线程组配置为循环 3 次，但是每一个 JMeter 线程只会发送一个请求到 Home 页面，原因是该请求存在于仅一次控制器中。

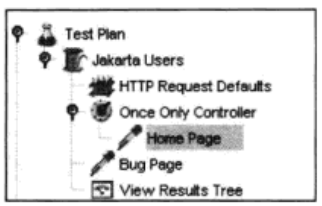


图 10-8　仅一次控制器（Once Only Controller）使用范例

每一个 JMeter 线程都会以如下顺序发送请求：Home Page、Bug Page、Bug Page、Bug Page。请注意，File Reporter 被用于在当前目录中保存测试结果，存储文件名为“simple-test.dat”。本例的测试计划，可以从如下地址下载：http://jakarta.apache.org/jmeter/demos/OnceOnlyTestPlan.jmx。

目前如果将仅一次控制器（Once Only Controller）放置在除线程组和循环控制器之外的其他测试元件之下，那么其行为没有定义，任何事情都有可能发生。

4．交替控制器（Interleave Controller）

如果将采样器或者逻辑控制器放到交替控制器之下，那么每一次循环 JMeter 都会交替执行该控制器下的测试元件，如图 10-9 所示。

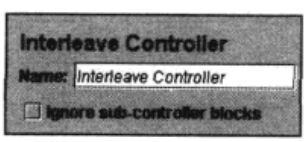


图 10-9　交替控制器（Interleave Controller）

参数如表 10-3 所示。

表 10-3 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Ignore sub-controller blocks	如果选中该复选框，则交替控制器会将其下的子控制器当做单个请求测试元件对待，并且每次测试控件只允许发送一个请求	否

1）简单交替范例

在这一例子中，我们会创建一个线程组，其中有两个线程，循环变量为 5，每个线程总共 10 个请求，如图 10-10 所示。JMeter 发送 HTTP 请求的顺序如表 10-4 所示。

图 10-10 交替控制器（Interleave Controller）范例 1

表 10-4 JMeter 发送 HTTP 请求的顺序

循环次数	每个 JMeter 线程发送这些 HTTP 请求
1	News Page
1	Log Page
2	FAQ Page
2	Log Page
3	Gump Page
3	Log Page
1	News Page
1	Log Page
2	FAQ Page
4	因为交替控制器中已经没有未执行过一次的请求了，JMeter 会从头开始发送第一个 HTTP 请求，那就是 News 页面
4	Log Page
5	FAQ Page
5	Log Page

本例的测试计划，可以从如下地址下载：

http://jakarta.apache.org/jmeter/demos/InterleaveTestPlan.jmx。

2）复杂交替范例

在这一例子中，我们会创建一个线程组，其中只有一个线程，循环变量为 8，如图 10-11 所示。请注意测试计划中有个外层交替控制器，其内嵌套有两个交替控制器。

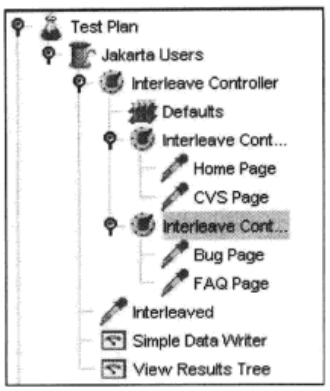


图 10-11　交替控制器（Interleave Controller）范例 2

外层交替控制器轮流执行内层的两个交替控制器，而内层交替控制器会轮流执行其下的每一个 HTTP 请求。每一个 JMeter 线程都以如下顺序发送请求：Home Page、Interleaved、Bug Page、Interleaved、CVS Page、Interleaved、FAQ Page 和 Interleaved。

如果主交替控制器内部的两个交替控制器被简单控制器所替代（如图 10-12 所示），那么执行顺序就是：Home Page、CVS Page、Interleaved、Bug Page、FAQ Page、Interleaved。不过，如果主交替控制器上选中“ignore sub-controller blocks”，那么执行顺序就是：Home Page、Interleaved、Bug Page、Interleaved、CVS Page、Interleaved、FAQ Page 和 Interleaved。

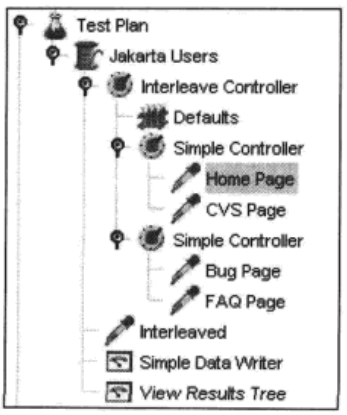


图 10-12　交替控制器（Interleave Controller）范例 3

本例的测试计划，可以从如下地址下载：

http://jakarta.apache.org/jmeter/demos/InterleaveTestPlan2.jmx。

5. 随机控制器（Random Controller）

随机控制器（Random Controller）的表现类似于交替控制器，唯一不同的就是其下的子测试元件不会按顺序轮流执行，而是每次循环随机挑选一个执行，如图 10-13 所示。

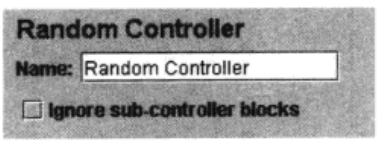


图 10-13　随机控制器（Random Controller）

参数如表 10-5 所示。

表 10-5　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否

6. 随机顺序控制器（Random Order Controller）

随机顺序控制器（Random Order Controller）很像一个简单控制器，其每一个子测试元件都至多执行一次，但是执行顺序却是随机的，如图 10-14 所示。

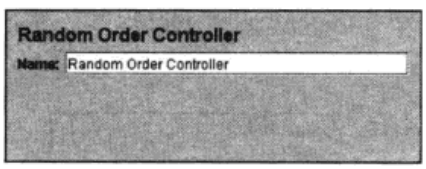


图 10-14　随机顺序控制器（Random Order Controller）

参数如表 10-6 所示。

表 10-6　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否

7. 吞吐量控制器（Throughput Controller）

该控制器的命名有问题，实际上它并不控制吞吐量。如果想人为调节吞吐量，请参考 Constant

Throughput Timer。吞吐量控制器允许用户设定其被执行的频率，如图 10-15 所示。这里有两种模式：Percent Execution 和 Total Executions。Percent Execution 模式下，该控制器会执行固定比例的循环次数（以测试计划总的循环次数为基准）。Total Executions 模式下，该控制器会在执行数次（设定值）后停止执行。同仅一次控制器（Once Only Controller）一样，当父循环控制器开始新一轮循环式，控制器的设定值会被重置。

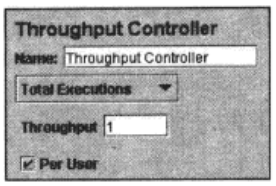


图 10-15　吞吐量控制器（Throughput Controller）

参数如表 10-7 所示。

表 10-7　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Execution Style	决定该元件是否处于 Percent Execution 模式或者 Total Executions 模式	是
Throughput	一个数字。对于 Percent Execution 模式，0～100 表明该控件的执行比率。“50”就意味着该控件在整个测试计划中将执行 50%的循环。对于 Total Executions 模式，该数字指明吞吐量控制器执行的总次数	是
Per User	如果选中该复选框，吞吐量控制器就会以虚拟用户为基础来计算它是否应该执行。如果没有选中该复选框，就是基于所有用户来计算的。例如，如果使用 Total Executions 模式，并且不选中“Per User”复选框，那么设定值就是吞吐量控制器的总执行次数，如果选中“Per User”复选框，那么总的执行次数就是虚拟用户数乘以 Throughput 设定值	是

8．运行时长控制器（Runtime Controller）

运行时长控制器（Runtime Controller）可以控制其下子测试元件允许运行的时长，如图 10-16 所示。

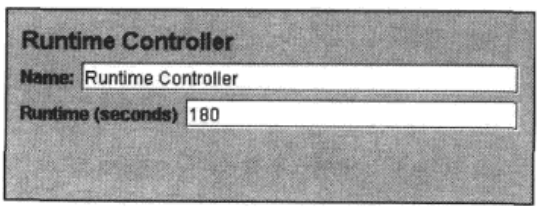


图 10-16　运行时长控制器（Runtime Controller）

参数如表 10-8 所示。

表 10-8　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	是
Runtime (seconds)	要求的运行时长（以秒为单位）	是

9. 如果控制器（If Controller）

用户通过如果控制器可以控制其下的子测试元件是否执行，如图 10-17 所示。

在 JMeter 2.3RC3 之前，判断条件是针对如果控制器之下的每一个可运行测试元件单独评估的，这会导致某些不可预期的 JMeter 运行表现，因此 2.3RC3 版本改为仅在初始入口处评估一次。不过最初的 JMeter 设计也有一定道理，所以 2.3RC4 之后的版本提供了一个额外选项，以便用户可以选择原来的方式。

JMeter 2.3.2 以后的版本，允许将脚本作为变量表达式处理，而不要求使用 JavaScript。在 JavaScript 判断条件中，可以使用函数和变量，最终它们会被用于判断“true”或者“false”；而现在可以不通过 JavaScript 来实现。例如，先前用户可以使用如下条件：${__jexl(${VAR} == 23)}，而这可以被判断为 true/false，该结果会被传递给 JavaScript，而 JavaScript 会返回 true/false。如果选中“Interpret Condition as Variable Expression?”复选框，那么对应的表达式将会被计算，并与“true”相比较，而不需要使用 JavaScript。另外，变量表达式可以返回任何值，而 JavaScript 判断条件必须返回 true/false 或者记录一个错误。

如果判断条件以 JavaScript 方式解释执行，那么脚本就无法访问任何变量。如果测试人员需要访问这些变量，那么请选中“Interpret Condition as Variable Expression?”复选框，并使用__javaScript()函数调用。接下来测试人员就能在脚本中使用“vars”、“log”、“ctx”等对象了。

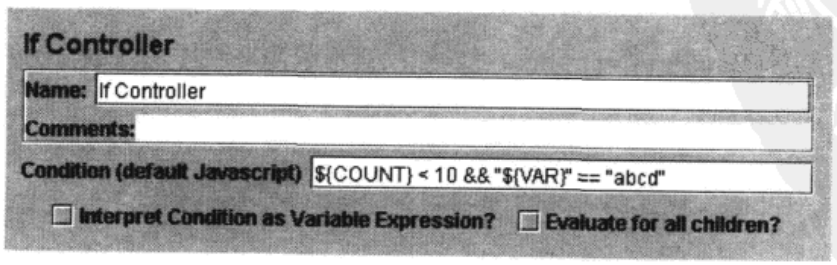


图 10-17　如果控制器（If Controller）

参数如表 10-9 所示。

表 10-9　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Condition (default Javascript)	默认情况下，判断条件以 JavaScript 代码的形式解释执行，返回“true”或者“false”。但这一方式可以被覆盖，见下面	是
Interpret Condition as Variable Expression?	如果选中了这一复选框，那么判断条件必须是一个表达式，其求值为“true”（忽略大小写）。例如，${FOUND} 或者${__jexl(${VAR} > 100)}。不同于 JavaScript 形式，判断条件只检查其值是否匹配“true”（忽略大小写）	是
Evaluate for all children?	判断条件是否针对所有子测试元件，如果不选中该复选框，则仅在初始入口处评估一次	是

范例（JavaScript）：

- `${COUNT} < 10`
- `"${VAR}" == "abcd"`
- `${JMeterThread.last_sample_ok}`（检查最后一个采样是否成功）

如果在解释执行代码时发生错误，那么条件就被认定为 false，并在 jmeter.log 记录一条消息。

范例（变量表达式）：

- `${__jexl(${COUNT} < 10)}`
- `${RESULT}`

10. While Controller

While Controller 会一直运行自己的子测试元件，直到条件变为“false”，如图 10-18 所示。

可能的条件值。

- Blank：当循环中有采样失败了后退出循环。
- LAST：当循环中有采样失败了后退出循环。如果循环前最近的采样失败了，不进入循环。
- Otherwise：当条件等同于字符串“false”时，退出（或者不进入）循环。

该条件可以是任何变量或者函数，且最终可以求值为字符串“false”。允许根据需要使用 JavaScript、BeanShell、属性或者变量。

举个例子：

- ${VAR} -当 VAR 被其他测试元件设置为 false
- ${__javaScript(${C}==10)}
- ${__javaScript("${VAR2}"=="abcd")}
- ${_P(property)} - 当属性在其他地方被设置为"false"

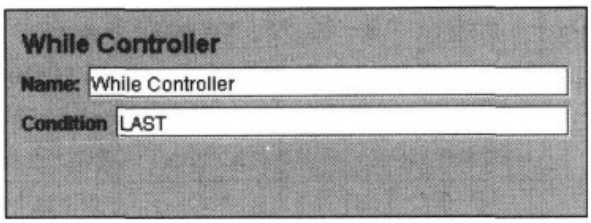


图 10-18 While Controller

参数如表 10-10 所示。

表 10-10 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	是
Condition	Blank、LAST，或者 variable/function	是

11. Switch Controller

Switch Controller 的表现类似于交替控制器（Interleave Controller），每一次循环都执行其下的子测试元件，不过并不是顺序执行，而是根据 Switch Value 选择执行，如图 10-19 所示。

JMeter 2.3.1 以后的版本，Switch Value 还可以是名称。

如果 Switch Value 超过了范围，它就会运行零号测试元件（数字形式下的默认执行）。如果该值为空字符串，同样会运行零号测试元件。

如果该值是非数字的（并且非空），那么 Switch Controller 就会寻找名称相同的测试元件（区别大小写）。如果没有名称匹配，那么就选择名称为“default”的测试元件（不区分大小写）。如果没有名为“default”的测试元件，那么就不会选择任何测试元件，也不会运行任何测试元件。

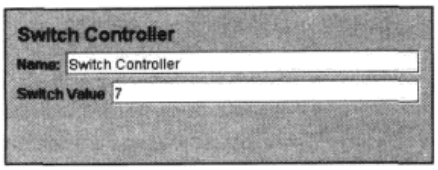


图 10-19　Switch Controller

参数如表 10-11 所示。

表 10-11　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	是
Switch Value	选择执行的子测试元件对应数字（或名称）。子测试元件从 0 开始编号	是

12. ForEach 控制器（ForEach Controller）

ForEach 控制器（ForEach Controller）会循环遍历一系列相关变量，如图 10-20 所示。当测试人员为一个 ForEach Controller 添加采样器（或者控制器）后，每个采样器（或者控制器）都会被执行一次或者多次，每次循环变量都会有一个新值。输入应该包含多个变量，每一个变量都由下画线和数字扩展而来。每一个变量都必须有值。例如，输入变量名为 inputVar，则如下变量应该已经被定义了：

- `inputVar_1 = wendy`
- `inputVar_2 = charles`
- `inputVar_3 = peter`
- `inputVar_4 = john`

分隔符“_”目前是可选的。

当给定的返回值为“returnVar”时，ForEach Controller 之下的采样器和控制器将会连续执行 4 次，上面的那些输入变量将会被用于采样器。

ForEach Controller 非常适合与前置处理器正则表达式一起使用。前置处理器正则表达式可以从前一个请求的结果数据中提取必要的输入变量。通过省略“_”分隔符，ForEach Controller 使用输入变量 refName_g 可以循环遍历所有组合。另外，使用 refName_${C}_g 格式的输入变量还可以循环遍历所有匹配的所有组合，其中 C 是一个计数器变量。

如果 inputVar_1 是空，ForEach Controller 不会返回任何采样。当正则表达式没有返回任何匹配项时，就是这种情况。

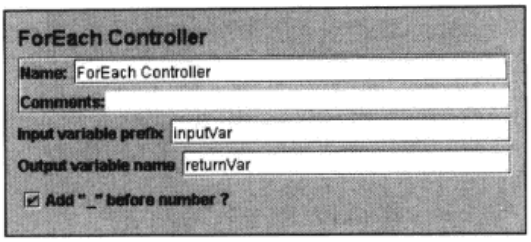


图 10-20　ForEach Controller

参数如表 10-12 所示。

表 10-12 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Input variable prefix	作为输入的变量名前缀	是
Output variable name	变量名，该变量可以在循环中替代采样的部分内容	是
Add "_" before munber	如果不选中，"_" 就会被省略	是

1）ForEach 范例 1

在这一例子中，我们创建一个测试计划，其中只发送一个特定 HTTP 请求一次，接下来针对页面上可以找到的每一个链接发送另外一个 HTTP 请求，如图 10-21 所示。

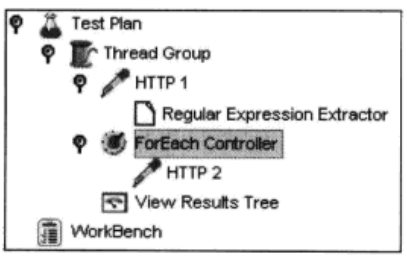


图 10-21　ForEach Controller 范例 1

我们将线程组配置为仅有一个线程，而循环变量的值为 1。测试人员可以看到我们为线程组添加了一个 HTTP 请求，为 ForEach Controller 也添加了一个 HTTP 请求。在第一个 HTTP 请求之后，添加了一个正则表达式提取器，它会从返回页面中提取所有的 html 连接，并将它们放到 inputVar 变量中。

在 ForEach 循环中有一个 HTTP 采样，它用于访问从第一个 HTTP 请求返回页面中提取的所有连接。本例的测试计划，可以从如下地址下载：http://jakarta.apache.org/jmeter/demos/forEachTestPlan.jmx。

2）ForEach 范例 2

在这个例子中，有两个正则表达式和 ForEach Controller，如图 10-22 所示。第一个正则表达式有匹配项，但是第二个正则表达式没有匹配项，因此没有通过第二个 ForEach Controller 运行任何采样器。

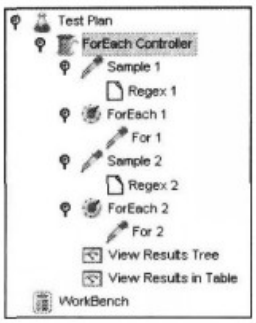


图 10-22　ForEach Controller 范例 2

线程组只有一个线程，循环变量设置为 2。

Sample 1 使用 JavaTest 采样器，返回字符串“a b c d”。

正则提取器使用表达式(\w)\s，匹配一个字母紧跟一个空格，并返回字母（而非空格）。任何匹配项都会加上前缀字符串“inputVar”。

ForEach Controller 提取所有前缀为“inputVar_”的变量，并执行自己的采样，将值传递到变量“returnVar”中。在这种情况下，它会依次将变量值设为“a”、“b”和 “c”。

For 1 采样器是另一个 Java 采样器，其使用返回变量“returnVar”作为采样标签的一部分和采样数据。

Sample 2、Regex 2 和 For 2 几乎是相同的，除了正则表达式改成了“(\w)\sx”，而没有匹配项，因此 For 2 采样器不会被运行。

本例的测试计划，可以从如下地址下载：

http://jakarta.apache.org/jmeter/demos/ForEachTest2.jmx。

13. 模块控制器（Module Controller）

模块控制器（Module Controller）提供了一个机制，即在当前测试计划中动态地替换测试计划片段，如图 10-23 所示。

一个测试计划片段包含一个控制器，且所有测试元件（采样器等）包含在其中。该片段可以被放在任何线程组中，或者放在工作台上。如果片段落在线程组中，那么可以通过让片段中

的控制器失效，来阻止整个片段的运行（除了通过模块控制器来运行）。或者测试人员可以将这些片段放在一个虚构的线程组中，并让整个线程组失效。

这里可以有多个片段，每个片段下都有不同类别的采样器。用户可以通过模块控制器在多个测试案例之间切换，要完成切换只需简单地在模块控制器的下拉列表中选择合适的控件，这样就能方便快捷地交替运行多个测试计划。

片段名可以用来代替空间名，以及控件的所有父测试元件名。例如：

```
Test Plan / Protocol: JDBC / Control / Interleave Controller (Module1)
```

模块控制器使用的任何片段都必须有一个唯一的名称，当测试计划重新加载时，该名称被用于寻找目标控制器。因此，控制器最好不要使用默认名称（就像上面例子所展示的那样），如果不这样做，那么为测试计划添加新测试元件时可能会出现重复的情况。

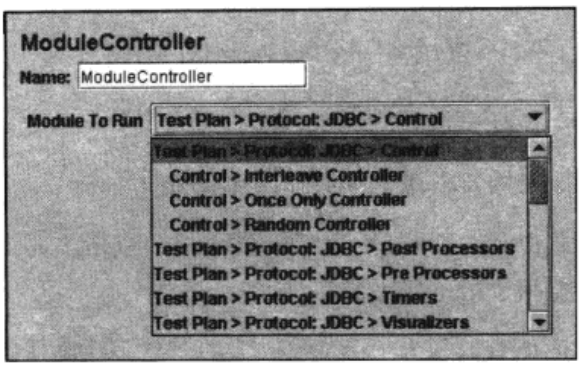


图 10-23　模块控制器（Module Controller）

小贴士　模块控制器（Module Controller）不应该在远程测试或者非 GUI 模式下与工作台组件一起使用，原因在于工作台测试元件不是测试计划 JMX 文件的一部分。任何这类测试都会失败。

参数如表 10-13 所示。

表 10-13　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Module To Run	模块控制器提供一个加载到 GUI 中的控制器列表。选择一个测试人员希望动态替换的	是

14. Include Controller

Include Controller 被设计用于使用外部 JMX 文件，如图 10-24 所示。要想使用它，将采样器添加到简单控制器之下，接着将简单控制器保存为 JMX 文件。该文件接下来可以被用到测试

计划中。被包含的测试计划中不能含有线程组，它只能包含简单控制器，以及简单控制器下的采样器、其他控制器等。

如果测试用到 Cookie Manager 或者 User Defined Variables，那么它们应该放在测试计划的顶层，而不应该放在被包含文件之中，否则就不能保证它们正常工作。

这一测试元件在 Filename 域中不支持变量/函数。但是，如果定义了属性 includecontroller.prefix，其内容将作为路径名的前缀。

如果在指定位置（prefix+filename）没有找到文件，那么控制器就会尝试打开 Filename，相对于 JMX 所在目录（JMeter 2.3.4 以后的版本）。

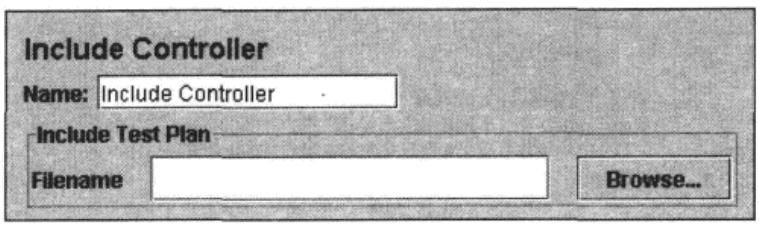


图 10-24　Include Controller

参数如表 10-14 所示。

表 10-14　参数描述

属性	描述	是否必需
Filename	待包含文件	是

15. 事务控制器（Transaction Controller）

事务控制器（Transaction Controller）会产生一个额外的采样，用于衡量执行嵌套测试元件所耗费的全部时间，如图 10-25 所示。请注意这一时间包含了控制器范围内的所有处理，不仅仅是采样。

对于 JMeter 2.3 以后的版本，这里有两种操作模式：

- 附加采样在嵌套采样之后添加。
- 附加采样作为嵌套采样的父采样来添加。

产生的采样时间涵盖了嵌套采样的所有时间，以及任何定时器时长等。依赖于 JMeter 时钟的实现，该采样时间可能会比独立采样加上定时器的总时间要稍长一些。时钟可能在控制器记录起始时间之后，但在第一个采样开始之前记录。结尾也是这样。

只有当所有子采样都成功时，该采样才被认为是成功的。

在父模式下，这些独立采样在查看结果树（Tree View Listener）中仍然可以看到，但是不会作为独立条目出现在其他监听器中。同样，这些子采样也不会出现在 CSV 日志文件中，但是它们可以被保存为 XML 文件。

在父模式下，可以为事务控制器添加断言。不过默认情况下，它们会影响独立采样及整体的事务采样。为了限制采样的范围，可以使用一个简单控制器来包含这些采样，并为简单控制器添加采样。父模式控制器目前不支持任何类型的事务控制器。

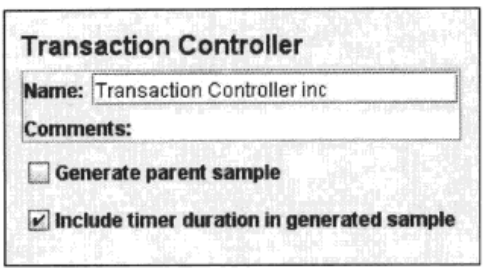


图 10-25　事务控制器（Transaction Controller）

参数如表 10-15 所示。

表 10-15　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件，并用来命名事务	是
Generate parent sample	如果选中了，就会产生一个采样，作为其他采样的父采样，否则产生的采样是一个独立采样	是
Include timer duration in generated sample	是否包含定时器，在产生的采样中前置和后置处理延迟。默认是选中，目的是为了与以前 JMeter 版本的操作相兼容	是

16. 录制控制器（Recording Controller）

录制控制器（Recording Controller）是一个存放位置，指明了代理服务器录制的采样应该放在哪里，如图 10-26 所示。在测试运行期间它没有任何用处，等同于简单控制器。但是在使用 HTTP 代理服务器（HTTP Proxy Server）录制期间，所有录制的采样默认情况下都会被保存到录制控制器之下。

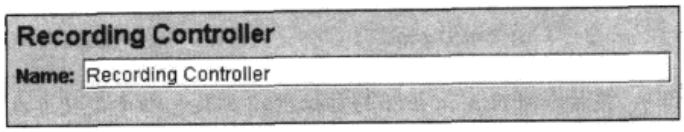


图 10-26　录制控制器（Recording Controller）

参数如表 10-16 所示。

表 10-16　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件。	否

10.3　详解 JMeter 配置元件

Jmeter 配置元件可以用于初始化默认值和变量，以便后续采样器使用。请注意这些配置元件将在其作用域的初始阶段处理，例如，在同一个作用域的任何采样器前。

1．CSV Data Set Config

CSV Data Set Config 被用来从文件中读取数据行，并将它们拆分后存储到变量中，如图 10-27 所示。它比__CSVRead()和_StringFromFile()函数更易于使用。它适合处理众多变量，另外对于使用“随机”和独立值来测试也很有帮助。动态产生唯一/随机值会大量占用 CPU 和内存，因此应该仅在特殊测试中创建数据。如果有必要，来自于文件中的“随机”数据可以与动态参数联系起来，以便为每一次运行创建值集合。例如，使用串联，这比每次运行时生成所有测试数据要“便宜”得多。

JMeter 2.3.1 以后的版本，允许变量被引用，这样就允许变量值包含分隔符。而以前版本则必须选择一个在任何变量值中都没有使用过的分隔符。

JMeter 2.3.4 以后的版本，支持 CSV 文件（其中第一行定义了列名）。要启动这一功能，“Variable Names”域需保留为空，并提供正确的分隔符。

默认情况下，该文件仅打开一次，而每一个线程会使用文件中不同的数据行。至于数据行传递给线程的顺序，依赖于它们执行的顺序（每一次循环都可能发生变化）。数据行在每次测试循环的开始阶段读取。文件名和模式在第一次循环时解析。

请参考下面关于 Sharing mode 的描述，以便掌握附加选项（JMeter 2.3.2+）。如果测试人员希望每个线程拥有自己独立的值集合，那么测试人员就需要创建一系列数据文件，为每个线程准备一个数据文件。例如 test1.csv、test2.csv、...testn.csv。使用文件名 test${__threadNum}.csv，

并将“Sharing mode”设置为“Current thread”。

CSV Dataset 变量在每次测试循环的初始阶段定义。由于定义发生在配置处理完成后，所以它们不能用于一些配置元件（如 JDBC Config），以便在配置时处理它们的内容。不过，这些变量在 HTTP Auth Manager 中能够正常工作，因为 username 是在测试运行时处理的。

作为一个特例，字符串“\t”（除去引号）在 Delimiter 域中被当做 Tab。

当达到文件结尾时，并且 Recycle 选项被置为 True，就会从文件第一行重新开始读取。

如果 Recycle 选项是 False，而 Stop thread 是 False，那么当到达文件结尾时所有变量都将被置为<EOF>。可以通过设置 JMeter 属性 csvdataset.eofstring 来改变该值。

如果 Recycle 选项为 False，而 Stop thread 是 True，那么到达文件结尾后，将导致线程被终止。

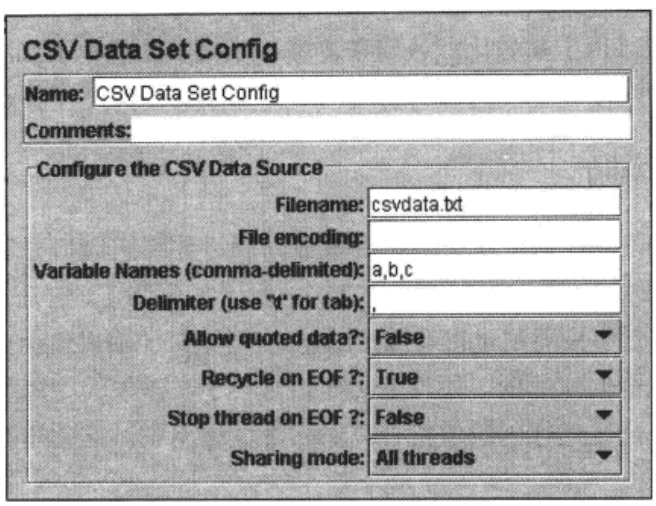


图 10-27　CSV Data Set Config

参数如表 10-17 所示。

表 10-17　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Filename	待读取文件名，解析相对文件名，以当前测试计划的路径为基准。另外还支持绝对路径名，不过请注意它们不应该用于远程模式，除非远程服务器有同样的目录结构。如果同样的物理文件用两种不同方式引用（如 csvdata.txt 和./csvdata.txt），那么它们就会作为不同文件处理。即使操作系统不区分大小写，csvData.TXT 同样会被独立打开	是
File encoding	用于读取文件的编码方式，如果不使用平台默认方式	否

续表

属性	描述	是否必需
Variable Names	变量名列表（逗号分隔）。JMeter 2.3.4 以后的版本支持 CSV 标题行：如果变量名域为空，那么文件的第一行将被读取，并被解释为列名的列表。这些变量名必须使用分隔符加以区分。它们可以使用双引号加以引用	是
Delimiter	分隔符被用来划分文件中的记录。如果数据行中的数据较少，无法更新所有变量，那么它们就保留原有值（如果存在）	是
Allow quoted data?	CSV 文件是否允许值被引用	是
Recycle on EOF?	达到文件结尾后，是否应该从文件开始处重新读取（默认是 True）	是
Stop thread on EOF?	如果 Recycle 设置为 False，达到文件结尾后，线程是否应该终止（默认是 False）	是
Sharing mode	■ All threads：（默认情况）文件在所有线程间共享 ■ Current thread group：每个文件会针对每一个线程组（其中有 CSV Data Set Config）打开一次 ■ Current thread：每个文件会针对每个线程单独打开 ■ Identifier：所有线程共享相同的标识，共享相同的文件。例如，如果有 4 个线程组，测试人员可以使用一个通用 ID，以便在两个或者多个线程组间共享文件。另外，测试人员还可以使用线程号，以便在不同线程组中有相同编号的线程间共享文件	是

2. FTP Request Defaults

FTP Request Defaults 被用于设置 FTP 请求的默认值，如图 10-28 所示。其中 Server Name or IP 选项用于设置服务器名称或者 IP 地址，Remote File 和 Local File 分别用于指定服务器和本地文件所在的路径。

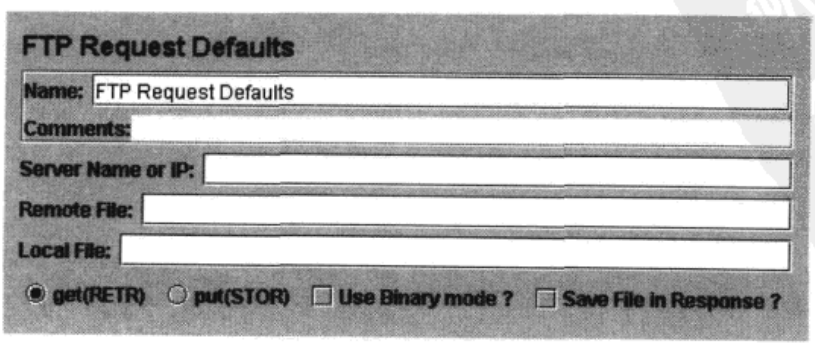


图 10-28　FTP Request Defaults

3. HTTP 授权管理器（HTTP Authorization Manager）

如果在一个采样器的作用域范围内有多个授权管理器，那么目前没有办法确定 JMeter 使用哪一个授权管理器。

授权管理器可以帮助测试人员指定针对 Web 页面（使用服务器校验限制）的一个或者多个登录，如图 10-29 所示。当使用浏览器访问一个受限页面时，测试人员就可以看到这类校验，浏览器会展示一个登录对话框。当遇到这类页面时，JMeter 就会传输登录信息。

校验头不会在查看结果树（Tree View Listener）中展示。

JMeter 2.2 以后的版本，如果设置没有定义，HTTP 客户端采样器默认使用 pre-emptive 校验。要禁止这一功能，就像下面这样设置值，在这种情况下校验仅对一个 challenge 做出响应。

```
jmeter.properties:
httpclient.parameters.file=httpclient.parameters

httpclient.parameters:
http.authentication.preemptive$Boolean=false
```

请注意：上面的设置只影响 HTTPClient 采样器（SOAP 采样器，也使用 HTTPClient）。

当通过 URL 寻找匹配项时，JMeter 会轮流检查每一个条目，并在找到第一个匹配项后停止搜索。因此大多数指定 URL 都是第一次出现在链表中，紧跟着不明确的 URL，重复的 URL 将被忽略。如果想在不同线程中使用不同的 Username/Password，测试人员可以使用变量。变量可以通过 CSV Data Set Config 来设置。

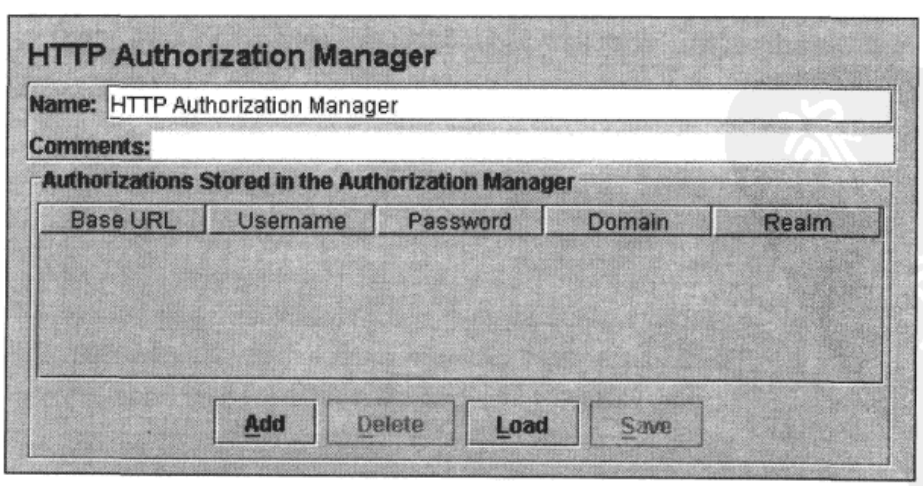


图 10-29　HTTP 授权管理器（HTTP Authorization Manager）

参数如表 10-18 所示。

表 10-18　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Base URL	一个部分或者完整的 URL，用于匹配一个或者多个 HTTP 请求 URL。例如，假如测试人员指定了一个 Base URL （http://jakarta.apache.org/restricted/），对应用户名“jmeter”，密码“jmeter”。如果测试人员发送一个 HTTP 请求到 URL（http://jakarta.apache.org/restricted/ant/myPage.html），授权管理器就会发送用户名为“jmeter”的登录信息	是
Username	校验用的用户名	是
Password	该用户的密码	是
Domain	针对 NTLM 使用的域	否
Realm	针对 NTLM 使用的 realm	否

Realm 仅仅影响 HttpClient 采样器。在 JMeter 2.2 中，Domain 和 Realm 没有独立的列，而是作为用户名的一部分，采用如下格式：[domain\]username[@realm]。这是一个历史特性，现在已经被移除了。

控件有如下几种：

- Add Button：在校验表格中添加一行条目。
- Delete Button：删除当前选中的表格条目。
- Load Button：加载一个前面保存的校验表格，并添加条目到已经存在的校验表格条目中。
- Save Button：将当前校验表格条目保存到文件之中。

当测试人员保存测试计划时，JMeter 会自动保存所有校验表格条目，包含任何密码，没有加密。

校验范例：

在这个例子中，我们在本地服务器上创建了一个测试计划，共发送 3 个 HTTP 请求，其中两个要求登录，而另外一个是开放的，如图 10-30 所示。在我们的服务器上，有一个受限目录名为“secret”，其中有“index.html”和“index2.html”。我们创建了一个登录 ID 名为“kevin”，对应密码为“spot”。因此，在授权管理器中，我们为受限目录创建一行条目，其中有 Username 和 Password，如图 10-31 所示。两个 HTTP 请求名为“SecretPage1”和“SecretPage2”，分别访问“/secret/index.html”和“/secret/index2.html”。另外一个 HTTP 请求名为“NoSecretPage”，会

访问“/index.html”。

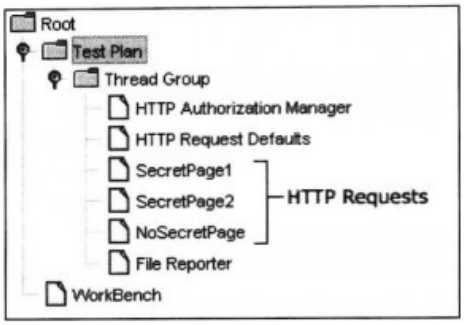


图 10-30　校验范例

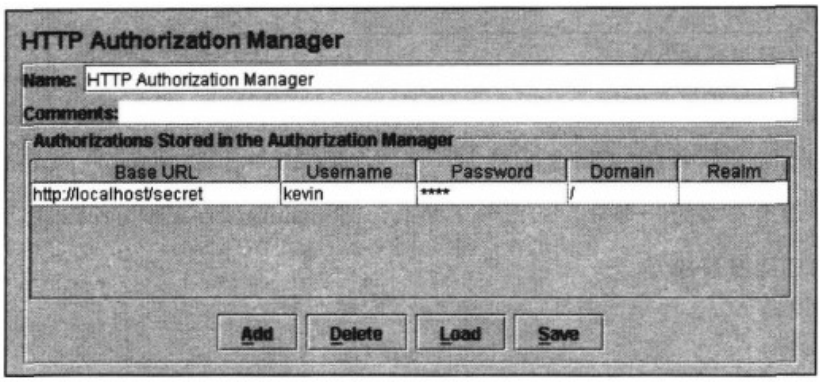


图 10-31　HTTP 授权管理器

当测试人员运行测试计划时，JMeter 会在校验表格中寻找它正在请求的 URL。如果 Base URL 与该 URL 相匹配，那么 JMeter 会将对应信息与该请求一起传递。

本例的测试计划，可以从如下地址下载：

http://jakarta.apache.org/jmeter/demos/AuthManagerTestPlan.jmx。

读者朋友可以从上面的地址下载范例测试计划，但是该测试计划是针对本地服务器构建的，因此无法正常运行它。不过，测试人员可以参考它来设计自己的测试计划。

4. HTTP Cache Manager

这是一个新测试元件，可能会发生改变。

HTTP Cache Manager 被用来为其作用域范围内的 HTTP 请求提供缓存功能，如图 10-32 所示。

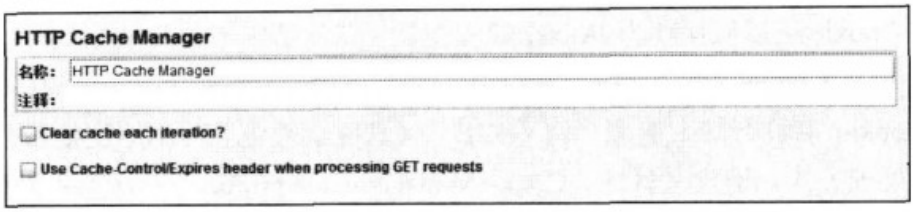


图 10-32　HTTP Cache Manager

如果某个采样成功了（如响应码为 2××），那么最近更新和 Etag（and Expired if relevant）的值将针对 URL 保存下来。在执行下一次采样前，采样器会检查缓存中是否存在某个条目，如果存在，那么将为请求设置 If-Last-Modified 和 If-None-Match 条件头。

另外，如果选中了“Use Cache-Control/Expires headerWhen Processing GET Veguests”复选框，那么 Cache-Control/Expires 值会根据当前时间来选择。如果请求是一个 GET 请求，而时间戳指向未来，那么采样器就会立即返回，而无须从远程服务器请求 URL。这样做是为了模拟浏览器的操作。请注意 Cache-Control 头必须是“public”的，并且只有“max-age”终结选项会被处理。

如果请求文档自从其被缓存以来没有发生改变，那么响应包体就会为空。这同样还包括超期时间指向未来的情况。这可能会导致断言存在问题。

5. HTTP Cookie 管理器（HTTP Cookie Manager）

如果在某个采样器的作用范围内有多个 Cookie 管理器，那么目前没有办法指定哪个管理器将被使用。另外，存在于某个 Cookie 管理器中的 Cookie，不能被其他 Cookie 管理器所访问，因此使用多个 Cookie 管理器需要小心。

Cookie 管理器（如图 10-33 所示）有两个主要功能：

其一，它像 Web 浏览器一样存储和发送 Cookie。如果测试人员有一个 HTTP 请求和响应其中包含有 Cookie，Cookie 管理器会自动存储 Cookie，那么接下来针对特定 Web 站点的所有请求中都会使用该 Cookie。每个 JMeter 线程都有自己的“Cookie 存储域”，因此，如果测试人员正在测试的 Web 站点使用 Cookie 来存储会话信息，那么每个 JMeter 线程都会有自己的会话。请注意这些Cookie不会在Cookie管理器中展示，不过它们可以在查看结果树（View Results Tree）中看到。

JMeter 2.3.2 及其以前的版本不会检查收到的 Cookie 对于 URL 而言是否正确。这就意味着跨域的 Cookie 将被保存下来，并可能在后续被使用。这一点已经在后续的版本中加以修正。要想恢复早前的处理方式，请定义 JMeter 属性“CookieManager.check.cookies=false”。

接收到的 Cookie 可以被保存为 JMeter 线程变量（默认情况下 JMeter 2.3.2 及其以后的版本不再这么做）。要将 Cookie 保存为变量，请定义属性“CookieManager.save.cookies=true”。另外，在被存储前 Cookie 名称会加上前缀“COOKIE_”（这样就避免了与本地变量意外发生重复），要恢复早前的处理方式，请定义属性“CookieManager.name.prefix=”（一个或多个空格）。如果启动了该功能，那么名称为 TEST 的 Cookie，可以通过${COOKIE_TEST}加以引用。

其二，测试人员可以手动为 Cookie 管理器添加一个 Cookie。不过，如果测试人员这么做了，那么该 Cookie 会被所有 JMeter 线程所共享。

从 2.0.3 版本以后，null 值的 Cookie 默认情况下会被忽略。这一点可以通过设置 JMeter 属性来改变：CookieManager.delete_null_cookies=false。请注意这同样会影响到自定义 Cookie，当 Cookie 值更新后任何这类 Cookie 都会被从显示中移除。另外需要注意 Cookie 名必须是唯一的，如果定义了第二个同名 Cookie，那么它就会替换第一个。

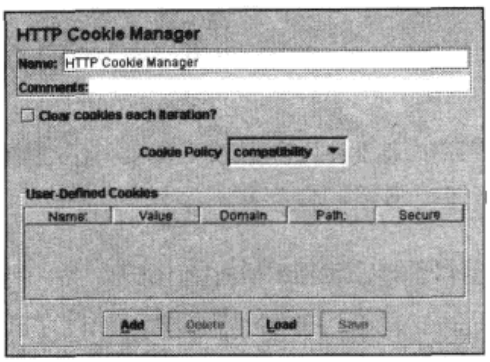


图 10-33　HTTP Cookie 管理器（HTTP Cookie Manager）

参数如表 10-19 所示。

表 10-19　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Clear cookies each iteration?	如果选中了该复选框，那么在每次主线程组循环执行后，所有服务器定义的 Cookie 将被清除。JMeter 2.3 以后的版本，任何定义在 GUI 中的 Cookie 都不会被清除	是
Cookie Policy	Cookie 规则将被用于管理 Cookie。“compatibility”是默认选项，它应该能满足大多数情况。请参考 http://jakarta.apache.org/httpcomponents/httpclient-3.x/cookies.html 和 http://jakarta.apache.org/httpcomponents/httpclient-3.x/apidocs/org/apache/commons/httpclient/cookie/CookiePolicy.html（请注意“ignoreCookies”等同于省略掉 Cookie 管理器）	是

续表

属性	描述	是否必需
User-Defined Cookies	这给了用户一个机会，即在测试运行期间为所有线程使用硬编码的 Cookies。“domain”是服务器的主机名（没有 http://），端口号目前将被忽略	否（不鼓励这么做，除非测试人员知道自己在做什么）
Add Button	为 Cookie 表格添加一行条目	N/A
Delete Button	删除当前选中的表格条目	N/A
Load Button	加载一个前面保存的 Cookie 表格，并将这些条目添加到已经存在的 Cookie 表格条目中	N/A
Save Button	将当前 Cookie 表格保存到文件中去（不保存任何从 HTTP 响应中提取的 Cookie）	N/A

6. HTTP 请求默认值（HTTP Request Defaults）

用户通过该测试元件，可以设置 HTTP 请求使用的默认值，如图 10-34 所示。例如，如果测试人员创建了一个测试计划，其中有 25 个 HTTP 请求，所有请求都发往相同的服务器，如此测试人员可以添加单个 HTTP 请求默认值，并填充“Server Name or IP”域。那么，当测试人员添加 25 个 HTTP 请求采样器时，可以将“Server Name or IP”域保留为空。这些 HTTP 请求采样器将会从 HTTP 请求默认值中继承该域的值。

JMeter 2.2 及其以前版本，端口 80 将被特殊对待，如果采样器使用 HTTPS 协议，它将被忽略。JMeter 2.3 及其以后版本会平等地对待所有端口值；如果在采样器中没有指明端口号，那么就会使用 HTTP 请求默认值中的端口号（如果有提供的话）。

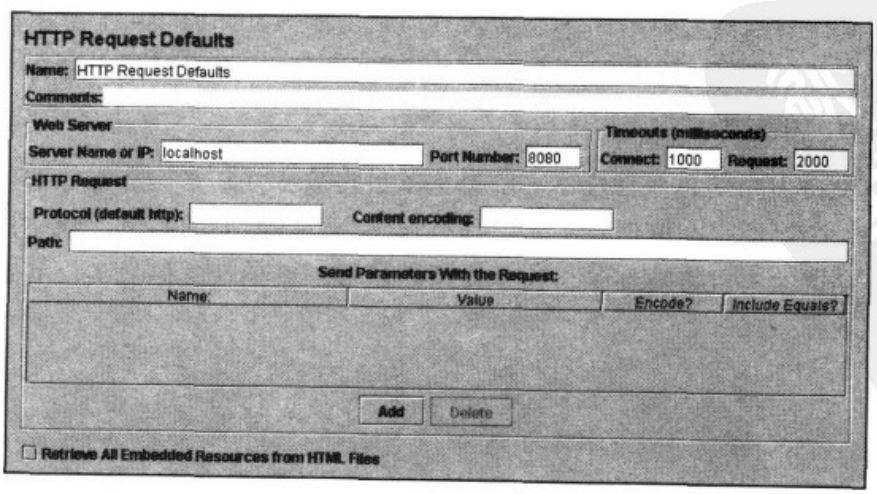


图 10-34　HTTP 请求默认值（HTTP Request Defaults）

参数如表 10-20 所示。

表 10-20 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Server	Web 服务器的域名或者 IP 地址，例如 www.example.com（不包括前缀 http://）	否
Port	Web 服务器监听的端口	否
Connect Timeout	连接超时时长。等待连接打开的毫秒数。当使用默认 Java HTTP 实现时，要求 Java 1.5 或以后版本	否
Response Timeout	响应超时时长。等待响应的毫秒数。当使用默认 Java HTTP 实现时，要求 Java 1.5 或以后版本	否
Protocol	HTTP 或者 HTTPS	是
Method	HTTP GET 或者 HTTP POST	否
Path	指向资源的路径（例如，/servlets/myServlet）。如果该资源要求查询字符串参数，将它们添加到下面的“Send Parameters With the Request”片段中。请注意这里的路径是全路径，而非 HTTP 请求界面指定路径的前缀	否
Send Parameters With the Request	查询字符串会通过测试人员提供的参数列表来产生。每个参数都有名称和值。查询字符串会以正确的方式产生，这依赖于测试人员选择的“Method”（例如，如果测试人员选择 GET，查询字符串会附在 URL 之后；如果选择 POST，那么它会单独发送）。另外，如果测试人员使用 multipart 表单来发送文件，那么查询字符串会依据 multipart 表单规范来创建	否

7. HTTP 信息头管理器（HTTP Header Manager）

用户通过信息头管理器可以添加或者重载 HTTP 请求头，如图 10-35 所示。一直到 JMeter 2.3.2，每一个采样器仅支持一个信息头管理器。如果在同一作用域范围内存在多个信息头管理器，那么只有最后一个会被使用。

JMeter 目前支持多个信息头管理器。信息头条目将被合并起来构成采样器列表。如果一个待合并条目匹配一个已经存在的信息头名，那么它就会替代前面的条目，除非条目值是空，在这种情况下已经存在的条目将被移除。这允许用户设置一系列默认信息头，并对特定采样器加以调整。

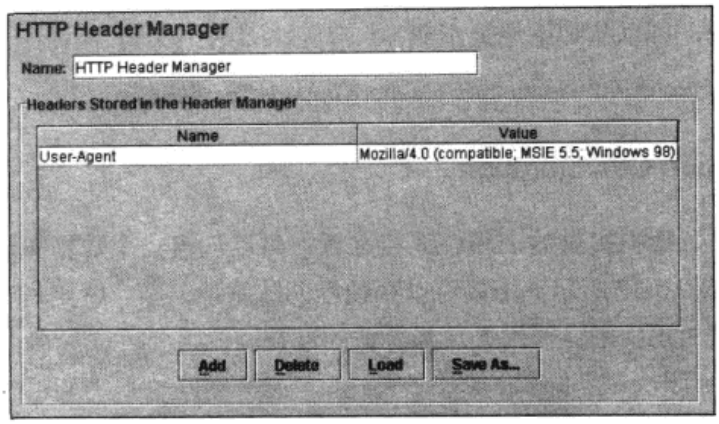


图 10-35　HTTP 信息头管理器（HTTP Header Manager）

参数如表 10-21 所示。

表 10-21　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Name (Header)	请求头的名称。经常用到的两个通用请求头“User-Agent”和“Referer”	否（不过，测试人员至少应该有一个）
Value	请求头的值	否（不过，测试人员至少应该有一个）
Add Button	为信息头表格添加一条目	N/A
Delete Button	删除当前选中的表格条目	N/A
Load Button	加载一个前面保存的信息头表格，并向已经存在的信息头条目添加这些条目	N/A
Save As Button	将当前信息头表格保存到文件中	N/A

信息头管理器范例如下：

在这一例子中，我们创建了一个测试计划，它告诉 JMeter 重载默认“User-Agent”请求头，并使用一个特定的 Internet Explorer 代理字符串来代替，如图 10-36 所示。

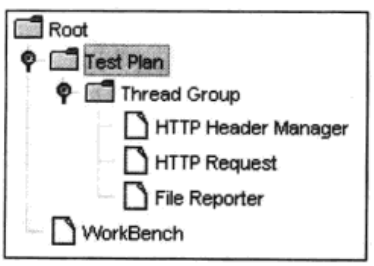


图 10-36　信息头管理器范例

本例的测试计划，可以从如下地址下载：

http://jakarta.apache.org/jmeter/demos/HeaderManagerTestPlan.jmx。

8. JDBC Connection Configuration

使用该测试元件，根据提供的 JDBC 连接设置，可以创建一个数据库连接（被 JDBC 请求采样器所使用），如图 10-37 所示。连接可以在线程间随意地共享，否则每个线程都得有自己的连接。连接配置名被 JDBC 采样器用来选择合适的连接。

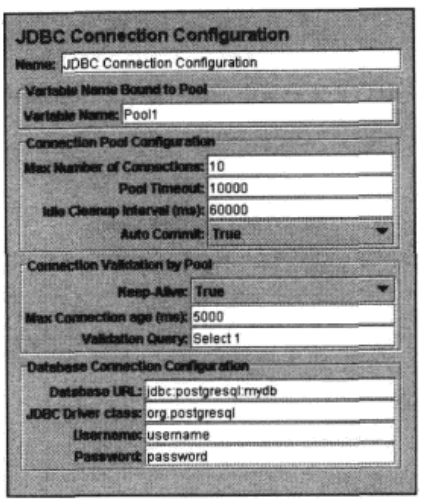


图 10-37　JDBC Connection Configuration

参数如表 10-22 所示。

表 10-22　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Variable Name	连接关联的变量名。可以同时使用多个连接，每一个连接关联不同的变量，这样 JDBC 采样器就能选择合适的连接。每个变量名都必须是不同的。如果两个配置元件使用相同的名称，那么只有一个会被保存下来。JMeter 2.3 以后的版本如果检测到有重复的名称，就会记录一条消息	是
Max Number of Connections	连接池中允许的最大连接数。在大多数情况下，将设置为零（0）。这意味着每个线程都会有自己的连接池，其中只有单个连接。例如，连接不会在线程间共享。如果测试人员真的希望共享连接池，那么就将该参数设置为线程数目，这样可以保证某个线程不会等待其他线程	是
Pool Timeout	如果在尝试重新取得连接的过程中，超过了超时周期，连接池就会抛出一个错误	是

续表

属性	描述	是否必需
Idle Cleanup Interval (ms)	空闲清除间隔时长，单位毫秒	是
Auto Commit	开启或者关闭自动提交功能	是
Keep-Alive	是否保持活动	是
Max Connectionage (ms)	最大连接周期，单位毫秒	是
Validation Query	一个简单的查询用来确定数据库是否能响应	是
Database URL	数据库的 JDBC 连接字符串	是
JDBC Driver class	驱动类的全名（必须在 JMeter 的 classpath 中，最简单的办法是将.jar 文件复制到 JMeter 的 lib 目录中）	是
Username	连接使用的数据库用户	否
Password	连接使用的密码	否

不同的数据库和 JDBC 驱动要求不同的 JDBC 设置。数据库 URL 和 JDBC 驱动类由 JDBC 实现提供者定义。

如果 JMeter 报告没有合适的驱动，那么可能有以下两种原因：

- 没有找到驱动类。在这种情况下，会有一条日志消息，例如：DataSourceElement: Could not load driver: {classname} java.lang.ClassNotFoundException: {classname}。
- 找到了驱动类，但是该类不支持连接字符串。这可能是由于连接字符串中存在语法错误，或者是由于使用了错误的类名。

如果数据库服务器没有运行或者无法访问，那么 JMeter 将会报告 java.net.ConnectException。

一些可能的设置如表 10-23 所示，更多信息请参考 JDBC 驱动文档。

表 10-23　一些可能的设置

数据库	驱动类	数据库 URL
MySQL	com.mysql.jdbc.Driver	jdbc:mysql://host[:port]/dbname
PostgreSQL	org.postgresql.Driver	jdbc:postgresql:{dbname}
Oracle	oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:@//host:port/service 或者 jdbc:oracle:thin:@(description=(address=(host={mc-name})(protocol=tcp)(port={port-no}))(connect_data=(sid={sid})))
Ingres	(2006)	ingres.jdbc.IngresDriver
SQL Server (MS JDBC driver)	com.microsoft.sqlserver.jdbc.SQLServerDriver	jdbc:sqlserver://host:port;DatabaseName=dbname
Apache Derby	org.apache.derby.jdbc.ClientDriver	jdbc:derby://server[:port]/databaseName[;URLAttributes=value[;...]]

9. Java 请求默认值（Java Request Defaults）

通过 Java 请求默认值组件，测试人员可以设置 Java 测试的默认值，如图 10-38 所示。

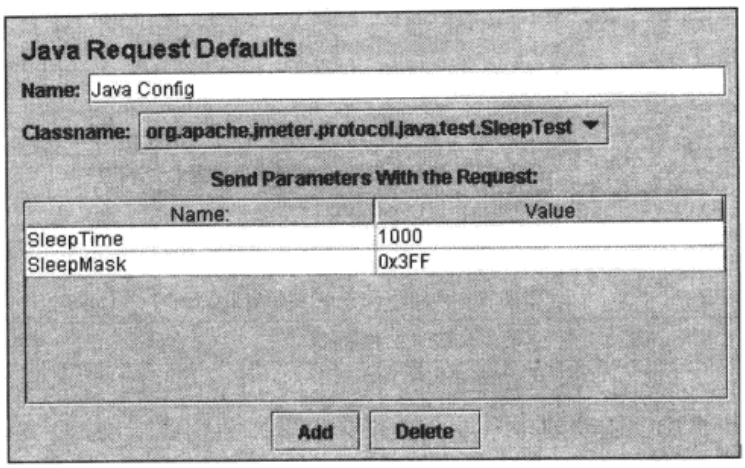


图 10-38　Java 请求默认值（Java Request Defaults）

10. 登录配置元件（Login Config Element）

测试人员可以使用登录配置元件来为采样器（将用户名和密码作为其初始设置的一部分）添加或重载用户名和密码，如图 10-39 所示。

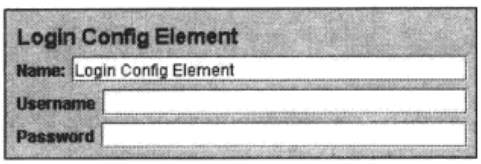


图 10-39　登录配置元件（Login Config Element）

参数如表 10-24 所示。

表 10-24　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Username	使用的默认用户名	否
Password	使用的默认密码	否

11. LDAP 请求默认值（LDAP Request Defaults）

通过 LDAP 请求默认值组件，测试人员可以设置 LDAP 测试的默认值，如图 10-40 所示。

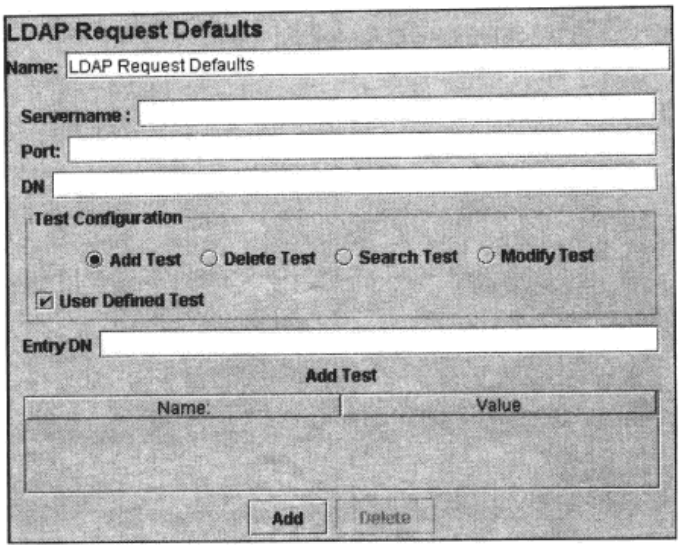


图 10-40　LDAP 请求默认值（LDAP Request Defaults）

12. LDAPExt Request Defaults

通过 LDAP 扩展请求默认值组件，测试人员可以设置扩展 LDAP 测试的默认值，如图 10-41 所示。

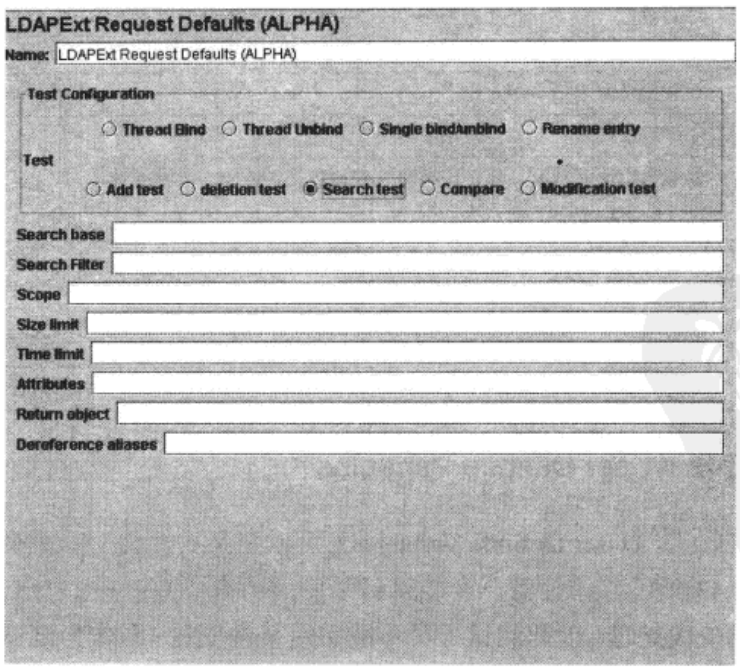


图 10-41　LDAPExt Request Defaults

13. TCP 采样器配置（TCP Sampler Config）

TCP 采样器配置（TCP Sampler Config）提供 TCP 采样器的默认数据，如图 10-42 所示。

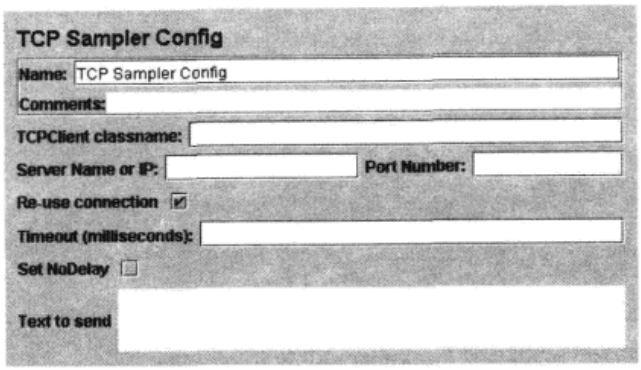


图 10-42　TCP 采样器配置（TCP Sampler Config）

参数如表 10-25 所示。

表 10-25　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
TCPClient classname	TCP 客户端类名。默认指向属性 tcp.handler，如果没有的话，使用 TCPClientImpl	否
Server Name or IP	TCP 服务器名或者 IP	否
Port Number	使用的端口号	否
Re-use connection	如果选中该复选框，那么连接将一直被打开。否则，数据被读取后连接就会被关闭	是
Timeout (milliseconds)	响应超时时间	否
Set Nodelay	属性 NoDelay 是否应该被设置	否
Text to send	待发送文本	否

14. 用户定义的变量（User Definde Variables）

通过用户定义的变量（User Definde Variables）可以定义初始化一系列变量，就像在测试计划中一样，如图 10-43 所示。请注意一个测试计划中的所有 UDV（用户定义的变量），无论它们在哪里，都在初始阶段处理。因此测试人员不能引用某些变量（即作为测试运行一部分定义），例如在一个后置处理器中就不能引用。

UDV（用户定义的变量）不应该与每次调用产生不同结果的函数一起使用。只有第一次函

数调用的结果会被保存到变量之中。不过 UDV 可以与这类函数一起使用（如__P()），例如：

```
HOST      ${__P(host,localhost)}
```

它定义变量“HOST”的值为 JMeter 属性“host”的值，如果没有定义的话，默认值为“localhost”。

要在测试运行阶段定义变量，参考前置处理器 User Parameters。UDV 依据它们在测试计划中出现的先后顺序而被执行，从头到尾。

为了便于理解，作者建议只将 UDV 放在线程组的开头（或者直接放在测试计划之下）。

一旦测试计划和所有 UDV 都被处理后，变量集的结果就会被复制到每一个线程，作为线程初始的变量集。

如果某个动态测试元件，例如 User Parameters（前置处理器）或者 Regular Expression Extractor（正则表达式提取器）定义了一个与 UDV 变量同名的变量，那么它就会替换初始值，并且线程中所有其他测试元件都会看见更新后的变量值。

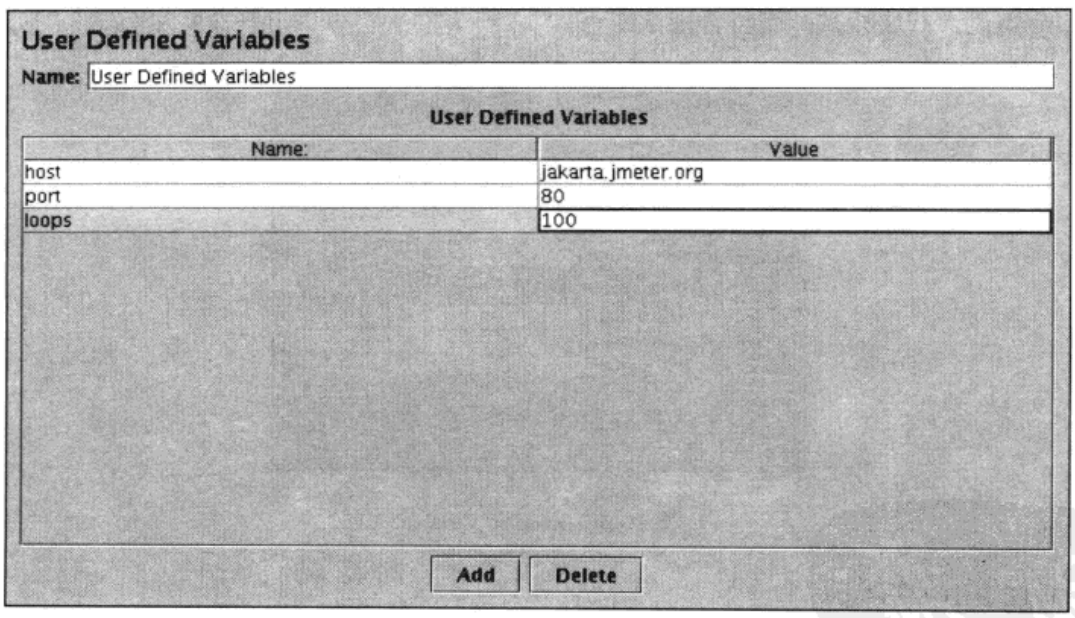


图 10-43　用户定义的变量（User Definde Variables）

小贴士

如果测试人员有多个线程组，应确保对不同值使用不同名称，因为 UDV 会在线程组间共享。另外，UDV 定义的变量在该测试元件被处理前无法引用，因此测试人员无法在 UDV 中引用同一个 UDV 中定义的其他变量。测试人员可以引用在前面 UDV 中定义的变量，或者在测试计划中定义的变量。

参数如表 10-26 所示。

表 10-26 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
User Defined Variables	变量名/值对。后续“Name”下的字符串被${...}包括后，构成对变量的引用。整个${...}组合接下来会被“Value”列下的字符串所替换	否

15. Random Variable

Random Variable 配置元件被用来产生随机数字字符串，接下来将其存放到变量之中，如图 10-44 所示。这比搭配使用 User Defined Variables 和__Random()函数要简单得多。

输出变量由随机数字生成器构建，而结果数字使用格式字符串来格式化。该数字使用公式 Minimum+Random.nextInt(Maximum－Minimum+1)来计算。Random.nextInt()要求为正整数。这就意味着 Maximum-Minimum 其范围必须小于 2 147 483 647，Minimum 和 Maximum 值可以是任何 long 型值，只要相减范围符合规定即可。

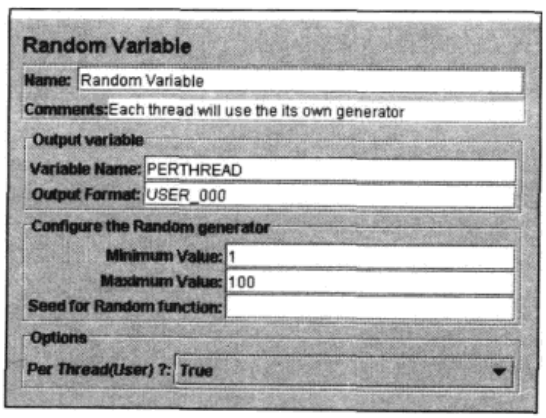


图 10-44 Random Variable

参数如表 10-27 所示。

表 10-27 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Variable Name	变量名，变量用于保存随机字符串	是
Output Format	使用的 java.text.DecimalFormat 格式字符串。例如“000”会产生至少 3 个数字的随机数，或者“USER_000”产生的输出格式为 USER_nnn。如果不指明，就使用 Long.toString()来产生数字	是
Minimum Value	产生随机数的最小值（整型）	否

续表

属性	描述	是否必需
Maximum Value	产生随机数的最大值（整型）	否
Seed for Random function	随机数产生器的种子。默认为当前时间（以毫秒为单位）	是
Per Thread(User)?	如果为 False，则随机数产生器在线程组的所有线程间共享。如果为 True，则每个线程都有自己的随机数产生器	否

16. 计数器（Counter）

该测试元件允许用户创建一个计数器，它可以在线程组中的任何一个地方被引用，如图 10-45 所示。该计数器配置元件允许用户配置一个起点、一个最大值，以及递增量。计数器会从初始值循环到最大值，接着又从初始值开始，如此持续下去直到测试结束。

从版本 2.1.2 开始，计数器使用整型来保存值，因此计数器值的范围为-2 ^ 63～2～63 ^ 1。

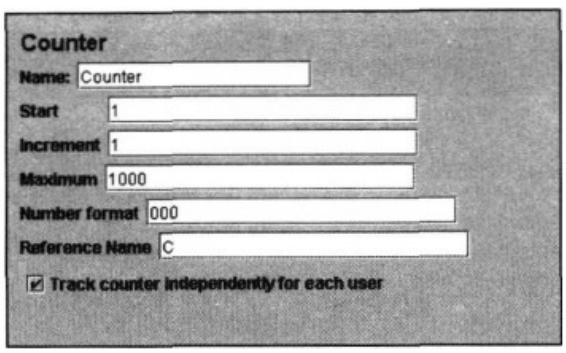


图 10-45　计数器（Counter）

参数如表 10-28 所示。

表 10-28　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Start	计数器的初始值。在第一次循环时，计数器等于该数值	是
Increment	每次循环后，计数器递增的数值	是
Maximum	如果计数器超越最大值，那么它就被重新设置为初始值。版本 2.2 以后，默认值是 Long.MAX_VALUE（以前它为 0）	否
Number format	可选的格式，例如 000 将会格式化 001、002 等。这将传递为十进制格式，因此可以使用任何正确的格式。如果在解释格式时存在问题，那么它就会被忽略（默认格式使用 Long.toString()产生）	否

续表

属性	描述	是否必需
Reference Name	通过它，测试人员可以在其他测试元件中引用计数器的值。语法：${reference_name}	是
Track counter independently for each user	是否是全局计数器，或者是否每个用户都有自己的计数器。如果没有选中该复选框，计数器就是全局的（例如，在第一次循环中，用户#1 获得值“1”，而用户#2 获得值“2”）。如果选中该复选框，则每个用户都有独立的计数器	否

17. 简单配置元件（Simple Config Element）

通过简单配置元件，用户可以在采样器中添加或者重载任意值，如图 10-46 所示。测试人员可以选择值的名称和值本身。尽管某些富有挑战精神的用户可以找到一些该测试元件的用处，但该测试元件最初是为 JMeter 开发人员设计的基础 GUI，以便于他们设计新的 JMeter 测试元件。

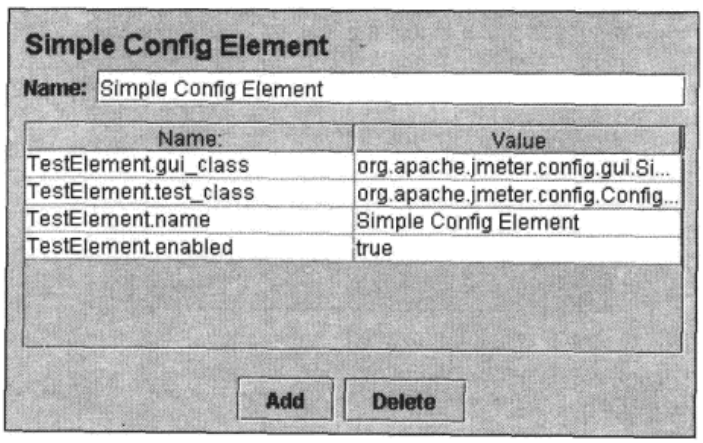


图 10-46 简单配置元件（Simple Config Element）

参数如表 10-29 所示。

表 10-29 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	是
Parameter Name	每个参数的名称。这些值是 JMeter 工作时的内部参数，通常没有文档详细介绍。只有熟悉 JMeter 源代码的人，才会知道这些值	是
Parameter Value	将应用于该参数的值	是

10.4　详解 JMeter 定时器

请注意，对 Jmeter 定时器的处理先于同一作用域范围内的采样器；如果在同一作用域范围内有多个定时器，那么所有定时器都会在每一个采样器前处理。

定时器的处理仅与采样器关联。如果定时器所处的作用域范围内没有采样器，那么定时器就不会被处理。

要将定时器单独应用于某个采样器，只需将定时器作为采样器的子测试元件。定时器将在采样器执行前应用。要让定时器在某个采样器之后生效，可以将定时器添加到下一个采样器之下，或者将定时器添加为 Test Action 采样器的子测试元件。

1. 固定定时器（Constant Timer）

如果测试人员希望每个线程在请求之间间隔固定时长，就使用此定时器，如图 10-47 所示。

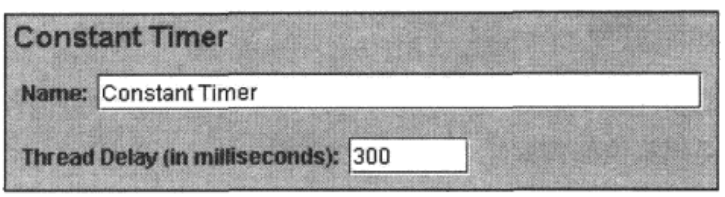


图 10-47　固定定时器（Constant Timer）

参数如表 10-30 所示。

表 10-30　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Thread Delay	等待的毫秒数	是

2. 高斯随机定时器（Gaussian Random Timer）

该定时器会暂停每个线程请求一个随机时长，而大多数时间间隔接近于一个固定值，如图 10-48 所示。总延时是高斯分布值（0.0 到标准差 1.0）乘以偏差值（用户指定）加上偏移值（Offset）的和。

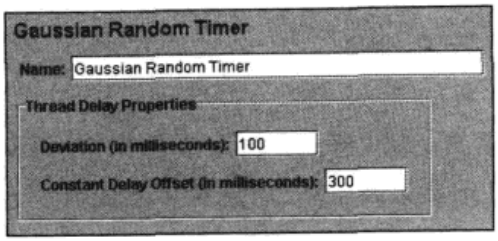


图 10-48　固定定时器（Constant Timer）

参数如表 10-31 所示。

表 10-31　参数说明

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Deviation	以毫秒为单位的偏差值	是
Constant Delay Offset	除随机延迟之外的固定等待毫秒数	是

3．Uniform Random Timer

该定时器会暂停每个线程请求一个随机时长，每个时间间隔都有同样的出现几率，如图 10-49 所示。总延时是随机值加偏移值（Offset）之和。

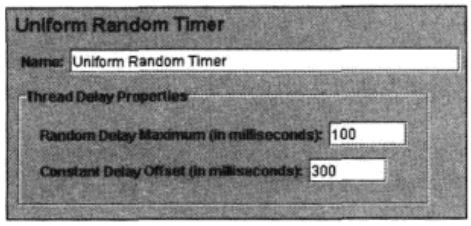


图 10-49　Uniform Random Timer

参数如表 10-32 所示。

表 10-32　参数说明

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Random Delay Maximum	以毫秒为单位的最大随机时长	是
Constant Delay Offset	除随机延迟之外的固定等待毫秒数	是

4．Constant Throughput Timer

该定时器引入了可变暂停时长，通过计算来保证总吞吐量（表征每分钟采样数的术语）尽

可能接近指定的值，如图 10-50 所示。如果测试服务器无法处理这么多请求，或者存在有其他定时器、耗费处理时间的测试元件拖了后腿，那么该定时器也无法保证吞吐量不下降。

尽管该定时器被称为 Constant Throughput timer，但是吞吐量值并不一定是恒定的。它可以由一个变量或者函数调用来定义，这样其值就可以在测试运行期间动态改变。该值可以通过如下方式改变：

- 使用一个计数器变量。
- 使用 JavaScript 或者 BeanShell 函数来提供一个变值。
- 使用远程 BeanShell 服务器来改变某个 JMeter 属性。

请注意，吞吐量值在测试运行期间不应该太频繁地改变，需要一定时间新值才能生效。

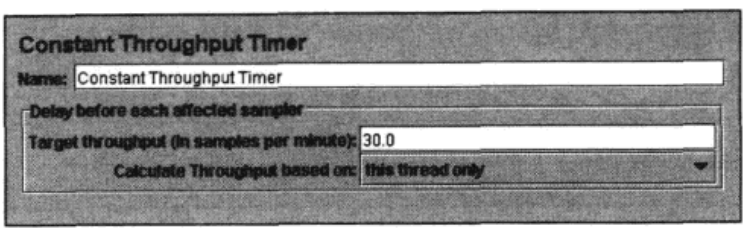


图 10-50　Constant Throughput Timer

参数如表 10-33 所示。

表 10-33　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Target throughput	我们期望定时器产生的吞吐量	是
Calculate Throughput based on	■ this thread only：每个线程都会尝试维持目标吞吐量。总吞吐量与活动线程数的数量成正比 ■ all active threads in current thread group：目标吞吐量在线程组所有活动线程间划分。每个线程都会基于上一次的运行时间，根据需要而进行延迟。在这种情况下，每个其他线程组都需要一个同样设置的 Constant Throughput Timer ■ all active threads in current thread group (shared)：与上面一样，但是每个线程的延迟都是基于线程组中任何一个线程上一次运行的时间 ■ all active threads (shared)：与上面一样，每个线程的延迟都是基于任何一个线程上一次运行的时间	是

5. Synchronizing Timer

Synchronizing Timer 的目的就是阻塞线程，直到 X 个线程已经被阻塞，接下来它们将一起被释放，如图 10-51 所示。如此一来，Synchronizing Timer 就可以在测试计划的多个点上创建大量瞬间压力。该定时器的作用类似于 Loadrunner 的集合点功能。

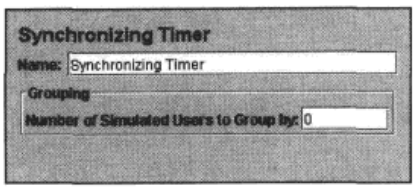


图 10-51　Synchronizing Timer

参数如表 10-34 所示。

表 10-34　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Number of Simulated Users to Group by	一次释放的线程数	是

6. BeanShell Timer

BeanShell Timer 可以被用来产生延迟，如图 10-52 所示。

关于如何使用 BeanShell 的全部细节，请参考 BeanShell Web 站点：http://www.beanshell.org/。

该测试元件支持 ThreadListener 和 TestListener 方法。这些应该在初始化文件中定义。参考 BeanShellListeners.bshrc 文件以便查看更多的定义范例。

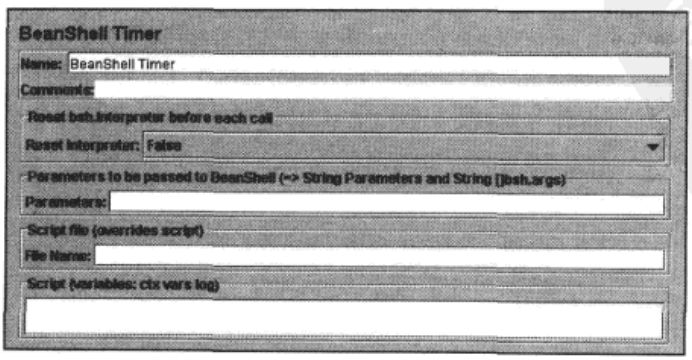


图 10-52　BeanShell Timer

参数如表 10-35 所示。

表 10-35　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Reset bsh.Interpreter before each call	如果设置了该选项，那么就会为每个采样器重新创建解释器。这一点对长期运行的脚本很有必要。更多细节请参考详解 JMeter 最佳实践经验一节	是
Parameters	传递给 BeanShell 脚本的参数。参数存储在如下变量之中： ■ Parameters：包含有参数的字符串，作为单个变量存在 ■ bsh.args：包含有参数的字符串数组，以空格作为间隔	否
Script file	一个文件，其中包含有待运行的 BeanShell 脚本。返回值将作为等待的毫秒数	否
Script	BeanShell 脚本。返回值将作为等待的毫秒数	是（除非提供了脚本文件）

唤醒脚本之前，在 BeanShell 解释器中初始化了一些变量。

- log - (Logger)：可以被用来记录日志文件。
- ctx - (JMeterContext)：可以通过它来访问 context。
- vars - (JMeterVariables)：提供读取/写入访问变量的方法：vars.get(key)；vars.put(key,val)；vars.putObject(“OBJ1”,new Object());。
- props - JMeter 属性，例如

props.get(“START.HMS”)；props.put(“PROP1”,“1234”);。

关于上述变量可以使用的方法，更多细节请参考对应 Javadoc。

如果定义了属性 beanshell.listener.init，那么它可以用于加载初始化文件。在该文件中可以定义 BeanShell 脚本将会用到的方法。

7. BSF Timer

使用 BSF 脚本语言，BSF Timer 可以被用来产生延迟，如图 10-53 所示。

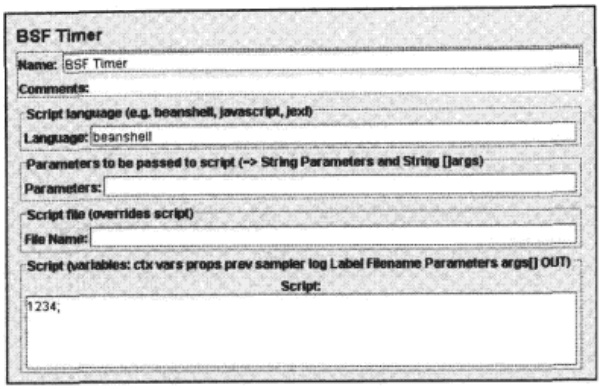


图 10-53　BSF Timer

参数如表 10-36 所示。

表 10-36　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Language	使用的脚本语言	是
Parameters	传递给脚本的参数。这些参数被存储于如下变量之中： ■ Parameters：字符串，以单个变量的形式包含参数 ■ bsh.args：字符串数组包含有参数，以空格加以分隔	否
Script file	文件，包含有待运行的脚本。返回值将转换为 long 型整数，并被用做等待的毫秒数	否
Script	待运行脚本。返回值将作为等待的毫秒数	是（除非提供了脚本文件）

在唤醒脚本之前，已经初始化了部分变量。

- log - (Logger)：可以用来记录日志文件。
- ctx - (JMeterContext)：可以通过它来访问 context。
- vars - (JMeterVariables)：可以通过它来读取/访问 JMeter 变量：vars.get(key); vars.put(key,val); vars.putObject("OBJ1",new Object()); vars.getObject("OBJ2");
- props：JMeter 属性，例如：props.get("START.HMS"); props.put("PROP1","1234");
- prev：前一次的采样结果（如果存在）。
- sampler-(Sampler)：可以通过它来访问最近的采样器。
- Label：字符串标签。
- Filename：脚本文件名（如果存在）。

- OUT – System.out，例如：

```
OUT.println("message")
```

关于上述变量可以使用的方法，更多细节请参考对应 Javadoc。

8．JSR223 Timer

使用 JSR223 脚本语言，JSR223 Timer 可以被用来产生延迟，如图 10-54 所示。更多细节请参考 BSF Timer。

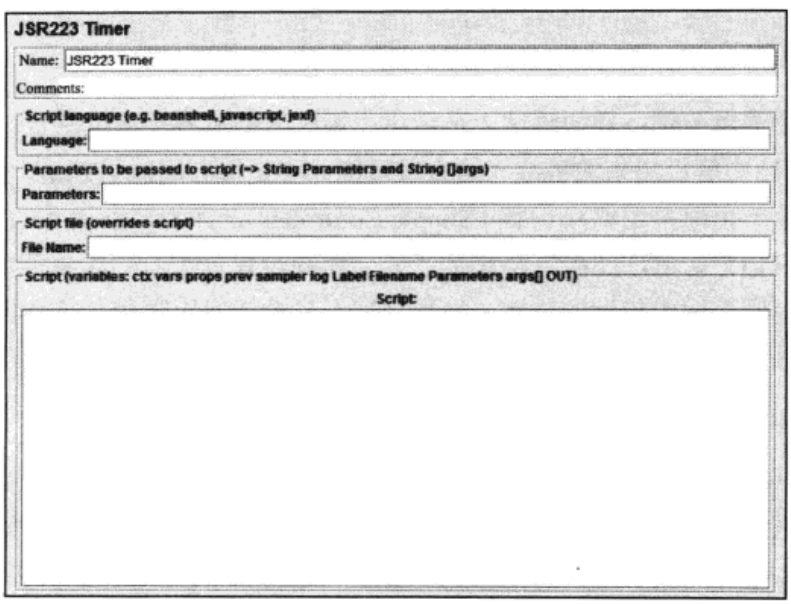


图 10-54　JSR223 Timer

10.5　详解 JMeter 前置处理器

前置处理器被用来修改其作用域范围内的采样器。

1．HTML 链接解析器（HTML Link Parser）

该修改器解析从服务器得到的 HTML 响应，并从中提取链接和表单，如图 10-55 所示。经过该修改器的一个 URL 测试采样将会被检查，以便查看其是否与从前面响应中提取的任何链接或者表单相符。HTML 链接解析器会使用来自于匹配链接或者表单的合适值，来替换 URL 测试采样中的值。它使用 Perl 型的正则表达式来寻找匹配项。

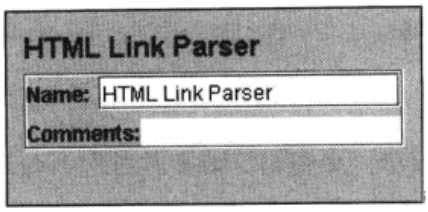


图 10-55　HTML 链接解析器（HTML Link Parser）

小贴士　寻找匹配项会用到协议（Protocol）、主机（Host）、路径（Path）和参数名（Parameter Names），目标采样器中不能含有任何不在响应链接中的参数。

遍历范例如下：

考虑这样一个简单范例：假如测试人员希望“辐射”遍历整个测试站点，一个个点击从服务器返回的 HTML 中解析出来的链接（这么做可能并不是很有用，但确是一个很好的范例）。测试人员需要创建一个简单逻辑控制器（Simple Controller），并为其添加“HTML Link Parser”。接着，创建一个 HTTP 请求，并将域设置为“.*”，路径设置也类似。这会导致测试人员的测试采样匹配任何在返回页面中找到的链接。如果测试人员希望遍历限制在某个特定域中，那么将域值改变为测试人员期望的值。如此一来，就只有指向该域的链接会被点击。

投票范例（如图 10-56 所示）：

给定一个 Web 投票系统，测试人员可能会面对一个投票页面，其中有多个单选按钮形式的投票选项供用户选择。假定这些投票选项是动态变化（根据用户生成）的。如果测试人员希望使用 JMeter 去测试投票，那么既可以创建测试采样（硬编码选择固定值），也可以使用“HTML Link Parser”来解析表单，并在测试人员的 URL 测试采样中插入一个随机投票选项。要做到这一点，请遵照上面的例子，除了配置 Web 测试元件的 URL 选项外，必须确保使用“POST”方法，在域、路径和任何辅助表单参数中填入硬编码值。接着，对于实际的单选按钮参数，填入名称（假定其名为“poll_choice”）和参数值“.*”。当 HTML 链接解析器检查该 URL 测试采样时，会发现它匹配投票表单（并且它不会匹配任何其他表单，假定测试人员已经明确了该 URL 测试采样的其他所有信息），并且 HTML 链接解析器会使用投票表单的匹配参数来替换测试人员的表单参数。因为正则表达式“.*”可以匹配任何东西，所以 HTML 链接解析器可以选择一系列单选按钮。它会随机进行选择，并替换 URL 测试采样中的值。每次运行测试，就会选择一个新的随机值。

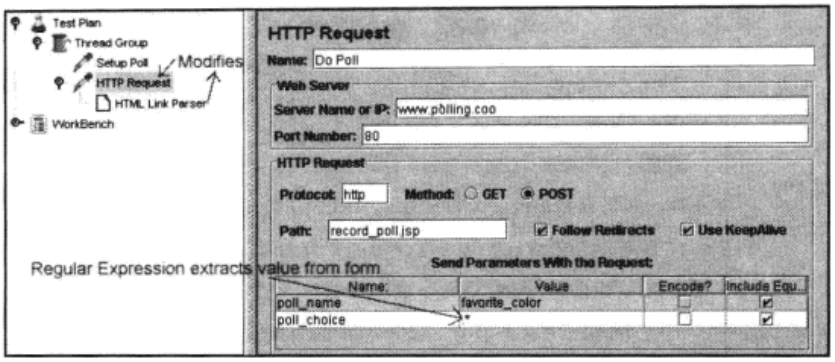


图 10-56　投票范例

小贴士　测试人员必须创建一个测试采样，并将其放在会返回 HTML 页面（其中有链接和表单）且关联动态测试采样的请求之前。

2. HTTP URL 重写修饰符（HTTP URL Re-writing Modifier）

该修改器功能类似于 HTML Link Parser，它比 HTML Link Parser 更易于使用，而且效率更高，如图 10-57 所示。对于使用 URL 重写来保存会话 ID 用于替代 Cookies 的 Web 应用系统，该测试元件可以被放在线程组级别，非常像 HTTP Cookie Manager。只需简单给予 HTTP URL 重写修饰符会话 ID 参数的名称，接着它就会在页面上找到会话 ID，并将其添加到线程组的每一个请求中。

另外，该修改器可以只影响选定的请求。聪明的用户甚至会使用修改器来获取某些值（HTML Link Parser 不能获取）。

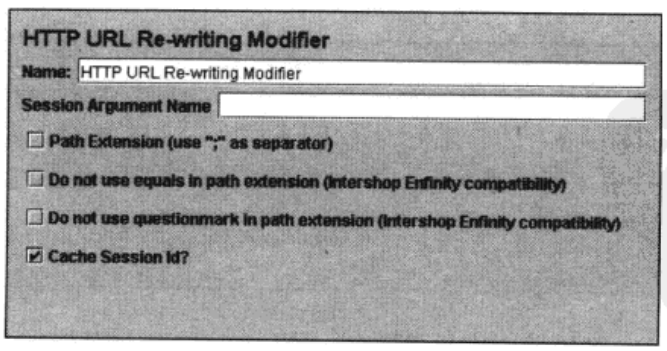


图 10-57　HTTP URL 重写修饰符（HTTP URL Re-writing Modifier）

参数如表 10-37 所示。

表 10-37 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Session Argument Name	从前一个响应抓取的参数名。修改器会从页面中找到该参数，并提取参数值，无论它处于 HREF 或者 FORM 中	是
Path Extension	一些 Web 应用通过附加一个分号加上会话 ID 参数来重写 URL。如果事实如此就选中这一复选框	否
Do not use equals in path extension	一些 Web 应用重写 URL，并不在参数名和值之间添加“=”，例如 Intershop Enfinity	否
Do not use questionmark in path extension	阻止查询字符串，以便在 path extension 中终结，例如 Intershop Enfinity	否
Cache Session Id?	是否保存会话 ID 的值，以便后续会话 ID 不存在时使用	是

3. HTML 参数掩码（HTML Parameter Mask）

笔者不推荐使用该测试元件，可以使用计数器来代替。

HTML 参数掩码（HTML Parameter Mask）被用来产生 HTML 参数的唯一值，如图 10-58 所示。通过指明参数名、前缀和后缀的值，以及计数器参数，该修改器会以“name=prefixcountersuffix”格式生成值。对于被 HTML 参数掩码所修改的任何 HTTP 请求而言，其内同名的任何参数都将会被替换，或者将合适的参数添加到请求参数列表中。

HTTP 请求中参数的值必须是“*”，以使 HTML 参数掩码（HTML Parameter Mask）替换该值。

例如，登录脚本的用户名可以被修改为发送一系列值，例如：user_1、user_2、user_3、user_4 等。

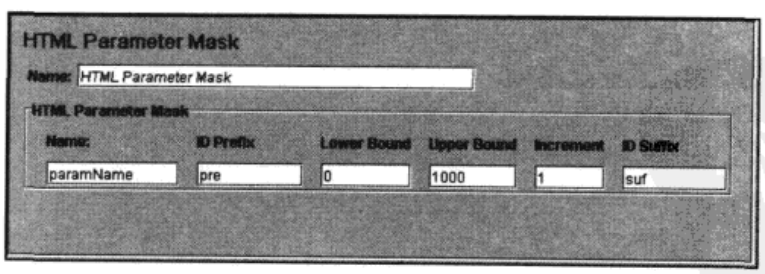


图 10-58 HTML 参数掩码（HTML Parameter Mask）

参数如表 10-38 所示。

表 10-38　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Name (second appearing)	待修改或者待添加到 HTTP 请求中的参数名	是
ID Prefix	一个字符串值，作为产生值的前缀	否
Lower Bound	一个数值，作为计数器的初始值	是
Upper Bound	一个数值，作为计数器的结束值，到达这一数值时，计数器会从 Lower Bound 重新开始循环	是
Increment	每一次循环，计数器增加的值	是
ID Suffix	一个字符串值，作为产生值的后缀	否

4. HTTP 用户参数修饰符（HTTP User Parameter Modifier）

笔者不推荐使用该测试元件，可以使用用户参数（User Parameters）来代替。另外请参考测试元件 CSV Data Set Config，它更适合于使用大量参数的情况。

用户参数修饰符使用一个 XML 文件来获取 HTTP arguments 的值，如图 10-59 所示。任何被该测试元件修改的 HTTP 请求，都将被检查是否存在指定的 arguments。如果存在，这些 arguments 的值将被 XML 文件中的值所替代。XML 文件中可以有多个集合值。该修改器会循环遍历这些值，如此每个请求都会得到一个不同的集合值，直到最后一个集合值，又从头开始。

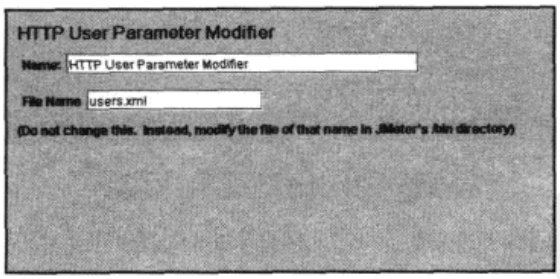


图 10-59　HTTP 用户参数修饰符（HTTP User Parameter Modifier）

参数如表 10-39 所示。

表 10-39　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
File Name	XML 文件名，该文件应该放置在 JMeter 的 bin 目录中，其中包含有集合值	是

5. 用户参数（User Parameters）

JMeter 用户通过用户参数（User Parameters）可以为独立线程的用户变量指定值，如图 10-60 所示。

用户变量同样可以在测试计划中指明，但是并不针对独立线程。在用户参数（User Parameters）的面板中可以为任何用户变量指定一系列值。对于每一个线程而言，变量会被依次赋予一系列值。如果线程数比变量值更多，那么变量值就会被重用。例如，这可以为每个线程指定一个不同的用户 ID。用户变量可以在任何 JMeter 测试元件的任何域中引用。

如果要指定变量，需要单击面板底部的“Add Variable”按钮，并在“Name:”列中输入变量名。如果要在一系列值中添加一个新值，则单击“Add User”按钮，并在新加入列中输入期望的值。

变量值可以在同一个线程组的任何测试元件中访问，语法：${variable}。

另外请参考测试元件 CSV Data Set Config，它更适合于使用大量参数的情况。

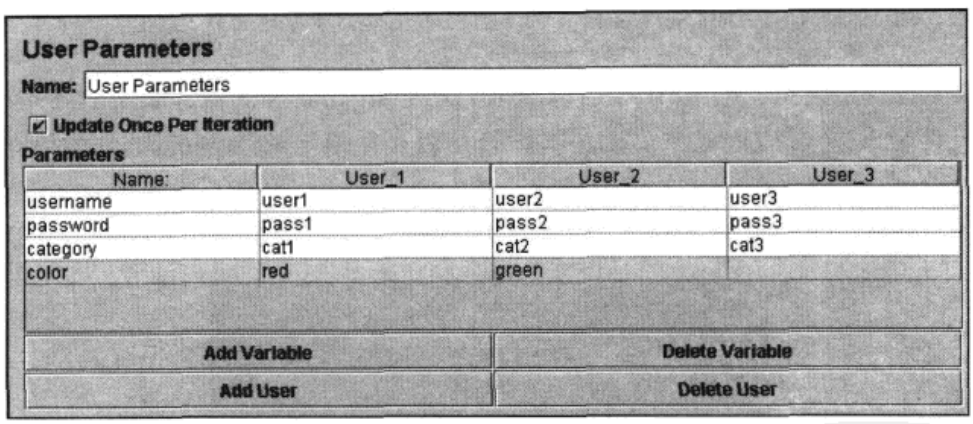


图 10-60　用户参数（User Parameters）

参数如表 10-40 所示。

表 10-40　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Update Once Per Iteration	一个标志位，用于标识用户参数（User Parameters）每一次循环是否只更新其变量一次。如果测试人员将函数集成进用户参数（User Parameters）中，那么测试人员就需要高超的控制能力，以便掌控变量值更新的频率。选中该复选框，可以确保每一次执行流通过用户参数的父控件时，都会更新变量值。不选中该复选框，用户参数会为其作用域范围内的每一个采样请求更新参数	是

6. BeanShell PreProcessor

通过 BeanShell PreProcessor 可以在发生采样之前执行任何代码，如图 10-61 所示。

关于 BeanShell 的全面介绍，请访问 BeanShell 的 Web 站点 http://www.beanshell.org/。

该测试元件支持 ThreadListener 和 TestListener 两种方法。这应该在初始化文件中定义。请查看 BeanShellListeners.bshrc 文件中的定义范例。

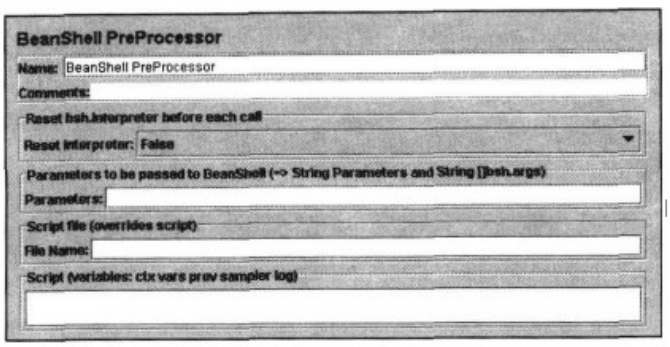


图 10-61　BeanShell PreProcessor

参数如表 10-41 所示。

表 10-41　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Reset bsh.Interpreter before each call	如果设置了该选项，那么就会为每个采样器重新创建解释器。这一点对长期运行的脚本很有必要。更多细节请参考详解 JMeter 最佳实践经验一节	是
Parameters	传递给 BeanShell 脚本的参数。参数存储在如下变量之中： ■ Parameters：包含有参数的字符串，作为单个变量存在 ■ bsh.args：包含有参数的字符串数组，以空格作为间隔	否
Script file	一个文件，其中包含有待运行的 BeanShell 脚本	否
Script	BeanShell 脚本。返回值将被忽略	是（除非提供了脚本文件）

唤醒脚本之前，在 BeanShell 解释器中初始化了一些变量。

- log - (Logger)：可以被用来记录日志文件。
- ctx - (JMeterContext)：可以通过它来访问 context。
- vars - (JMeterVariables)：提供读取/写入访问变量的方法：

vars.get(key); vars.put(key,val); vars.putObject("OBJ1",new Object());

- props – Jmeter 属性，例如：

```
props.get("START.HMS");
props.put" ("PROP1","1234");
```

- prev – (SampleResult)：可以被用来访问前一次的采样结果（如果存在）。
- sampler – (Sampler)：可以被用来访问当前采样。

关于上述变量可以使用的方法，更多细节请参考对应 Javadoc。

如果定义了属性 beanshell.listener.init，那么它可以用于加载初始化文件。在该文件中可以定义 BeanShell 脚本将会用到的方法。

7. BSF PreProcessor

通过 BSF PreProcessor 可以在发生采样之前执行 BSF 脚本代码，如图 10-62 所示。

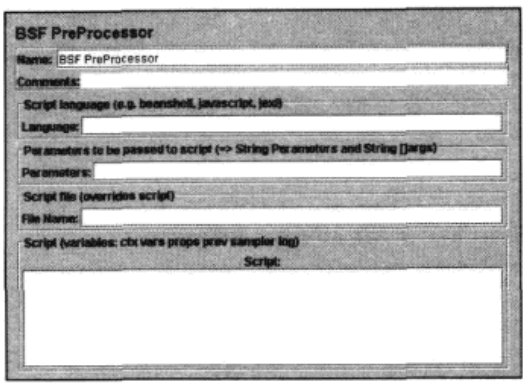


图 10-62　BSF PreProcessor

参数如表 10-42 所示。

表 10-42　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Language	指明 BSF 使用的语言	是
Parameters	传递给脚本的参数。这些参数被存储于如下变量之中： ■ Parameters：字符串，以单个变量的形式包含参数 ■ bsh.args：字符串数组包含有参数，以空格加以分隔	否
Script file	文件，包含有待运行的脚本	否
Script	待运行的脚本	是（除非提供了脚本文件）

BSF PreProcessor 将会使用 BSFEngine.exec()方法对脚本（或者文件）进行处理，脚本执行不会有返回值。

如下 BSF 变量将被初始化，以便后续被脚本使用。

- log - (Logger)：可以用来记录日志文件。
- Label：字符串标签。
- Filename：脚本文件名（如果存在）。
- Parameters：参数（字符串形式）。
- args[]：参数，字符串数组形式（以空格分隔）。
- ctx - (JMeterContext)：可以通过它来访问 context。
- vars - (JMeterVariables)：可以通过它来读取/访问 JMeter 变量。例如：

```
vars.get(key); vars.put(key,val);
vars.putObject("OBJ1",new Object());
vars.getObject("OBJ2");
```

- props - JMeter 属性，例如：

```
props.get("START.HMS");
props.put("PROP1","1234");
```

- prev - (SampleResult)：可以通过它来访问前面的采样结果（如果存在）。
- sampler - (Sampler)：可以通过它来访问当前采样器。
- OUT - System.out：例如：

```
OUT.println("message")
```

关于上述变量可以使用的方法，更多细节请参考对应 Javadoc。

8. JSR223 PreProcessor

通过 JSR223 PreProcessor，可以在发生采样前执行 JSR223 脚本代码，如图 10-63 所示。更多细节，请参考 BSF PreProcessor。

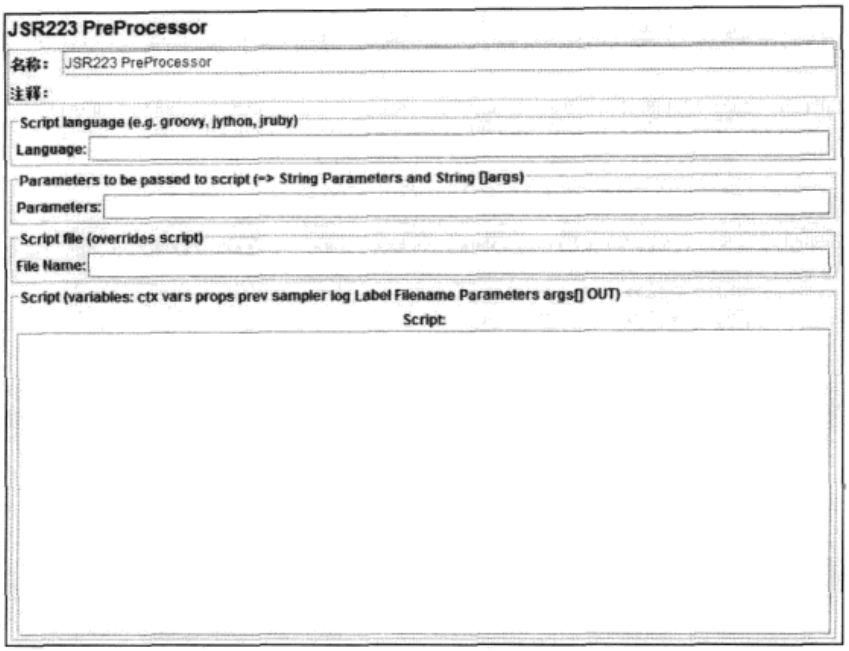


图 10-63 JSR223 PreProcessor

10.6 详解 JMeter 后置处理器

就像名字的含义一样，后置处理器在采样器之后生效。请注意它会影响其作用域范围内的所有采样器，因此如果要让后置处理器仅仅影响一个特定的采样器，那么就将其作为采样器的子测试元件。

注意，除非另外说明，后置处理器不会影响低层采样（子采样）（仅影响父采样）。在 BSF 和 BeanShell 后置处理器的例子中，脚本可以使用 prev.getSubResults()方法（它会返回包含采样结果的数组，如果没有低层采样的话，数组为空）来获取低层采样。

后置处理器在断言之前运行，因此它们无法访问任何断言结果及任何反映断言结果的采样状态。如果测试人员需要访问断言结果，可以使用监听器。另外还需要注意，当所有断言都被运行后，变量 JMeterThread.last_sample_ok 将被设置为“true”或者“false”。

1. 正则表达式提取器（Regular Expression Extractor）

正则表达式提取器允许用户使用 Perl 型正则表达式从一个服务器响应中提取值，如图 10-64 所示。作为一个后置处理器，该测试元件会在其作用域范围内的每一个采样请求之后执行，应用正则表达式，提取要求的值，产生模板字符串，将结果保存到指定变量名中。

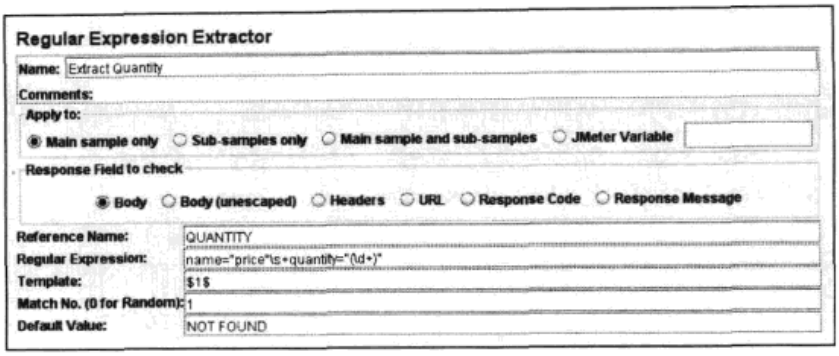


图 10-64　正则表达式提取器（Regular Expression Extractor）

参数如表 10-43 所示。

表 10-43　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Apply to:	这针对可以产生子采样的采样器，例如，HTTP 采样器集成有其他资源。通过事物控制器产生的 Mail Reader 或者采样 ■ Main sample only：仅应用于主采样 ■ Sub-samples only：仅应用于子采样 ■ Main sample and sub-samples：既应用于主采样，又应用于子采样 ■ JMeter Variable：断言将被应用于指定变量的内容。 匹配将轮流应用于所有符合条件的采样。例如，如果这里有 1 个主采样和 3 个子采样，每个采样都含有单个匹配项（总共 4 个匹配项）。设置 match number = 3，Sub-samples only，提取器将会匹配 3 个子采样。设置 match number = 3，Main sample and sub-samples，提取器将会匹配 2 个子采样（第一个匹配项在主采样）。设置 match number = 0 或者负数，所有符合条件的采样都将被处理。设置 match number > 0，则一旦找到足够多的匹配项，就会停止继续匹配	是
Response Field to check	可以检查如下响应域： ■ Body：响应的主题部分，例如 Web 页面的内容 （排除头部） ■ Body (unescaped)：响应的主题部分，所有 Html escape 码将被替换。请注意处理 Html escapes 时不考虑内容，因此可能会发生一些不正确的替换 ■ Headers：对于非 HTTP 采样可能不存在 ■ URL ■ Response Code，例如 200 ■ Response Message，例如 OK	是

续表

属性	描述	是否必需
Reference Name	JMeter 变量名，对应变量用于存储结果。另外请注意每个组合都以[refname]_g#形式保存，其中[refname]是测试人员在 reference name 中输入的字符串，而#是组合编号，group 0 是整个匹配，group 1 是从第一个圆括号集开始的匹配项，等等	是
Regular Expression	用于解析响应数据的正则表达式。其中必须包含至少一个圆括号集“()”，用于捕获字符串的一部分，除非使用组合0。不要将表达式用//封装，除非测试人员同样期望匹配这些字符	是
Template	模板被用来创建一个字符串，会使用找到的匹配项。这是一个灵活的字符串，其中的特殊元素会引用正则表达式中的组合。引用组合的语法是：'1'引用组合 1，'2'引用组合 2，等等。0引用整个表达式的匹配	是
Match No.	指明使用哪一次匹配。 ■ 使用值 0 ，指明 JMeter 应该随机选择一个匹配项 ■ 一个正整数 N 意味着选择第 N 次匹配 ■ 负数被用于与 ForEach 控制器协同工作（参考下面的内容）	是
Default Value	如果正则表达式没有匹配项，那么引用变量将被设置为默认值。这一点对于调试测试很有用。如果没有提供默认值，则很难区别是正则表达式没有匹配项，或者是正则表达式提取器没有被处理，或者可能使用了错误的变量 另外，如果测试人员有多个测试元件设置了相同的变量，测试人员可能希望在表达式没有匹配项时，保持变量不发生变化。在这种情况下，一旦调试完成后就移除默认值	否，不过推荐

如果匹配编号被设置为一个非负整数，并且有一个匹配发生，那么变量将像下面这样设置。

- refName：模板的值。
- refName_gn，其中 n=0,1,2：匹配的组合。
- refName_g：正则表达式的匹配数（排除 0）。

如果没有匹配发生，那么引用变量将被设置为默认值（除非不存在）。另外，下面的变量将被移除：

- refName_g0。
- refName_g1。
- refName_g。

如果匹配编号被设置为负数，那么采样数据所有可能的匹配项都将被处理。变量将像下面这样设置。

- refName_matchNr：找到的匹配数目，可以为 0。
- refName_n，其中 n = 1,2,3 等：通过模板产生的字符串。
- refName_n_gm，其中 m=0,1,2：第 n 次匹配的各项组合。
- refName：总是设置为默认值。
- refName_gn：没有设置。

请注意在这种情况下 refName 变量总是设置为默认值，相关的组合变量不会被设置。

2．XPath Extractor

该测试元件允许用户使用 XPath 查询语言从结构化响应（XML 或者(X)HTML）中提取值，如图 10-65 所示。

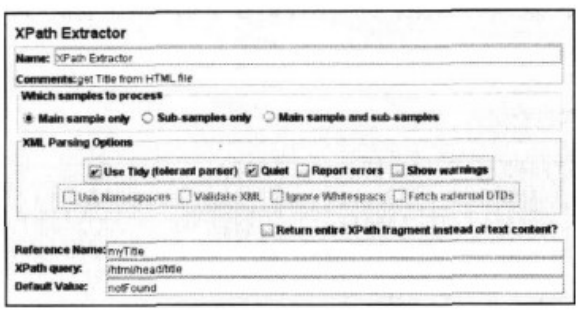


图 10-65　XPath Extractor

参数如表 10-44 所示。

表 10-44　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Apply to:	这是针对可以产生子采样的采样器，例如，HTTP 采样器集成有其他资源。通过事物控制器产生的 Mail Reader 或者采样 ■ Main sample only：仅应用于主采样。 ■ Sub-samples only：仅应用于子采样。 ■ Main sample and sub-samples：既应用于主采样，又应用于子采样 XPath 匹配将轮流应用于所有符合条件的采样，并且所有匹配结果都将被返回	是

续表

属性	描述	是否必需
Use Tidy (tolerant parser)	如果选中了该复选框，则使用 Tidy 将 HTML 响应解析为 XHTML 对于 HTML 响应“Use Tidy”复选框应该被选中。使用 Tidy 这些响应将被转化为正确的 XHTML（XML compatible HTML） 对于 XHTML 或者 XML 响应，“Use Tidy”不应该被选中，例如，RSS	是
Quiet	设置 Tidy Quiet 标志	如果选中了 Tidy 时必需
Report errors	如果 Tidy 发生了错误，那么就因此设置断言	如果选中了 Tidy 时必需
Show warnings	设置 Tidy showWarnings 选项	如果选中了 Tidy 时必需
Use Namespaces	如果选中了该复选框，那么 XML 解析器就会使用 namespace 分析。请注意目前只有在根测试元件定义的 namespace 才会被识别。以后的 JMeter 版本，可能会支持用户定义附加 workspace 名称。期间，//mynamespace:tagname 会被 //*[local-name()='tagname' and namespace-uri()='uri-for-namespace']所替代，其中“uri-for-namespace”是"mynamespace" namespace 的 uri（如果选中了 Tidy，则该复选框不可用）	如果没有选中 Tidy 时必需
Validate XML	通过 schema 来检查文档	如果没有选中 Tidy 时必需
Ignore Whitespace	忽略空格	如果没有选中 Tidy 时必需
Fetch external DTDs	如果选中该复选框，额外的 DTDs 将被取出	如果没有选中 Tidy 时必需
Return entire XPath fragment instead of text content?	如果选中该复选框，将返回 XPath 片段，而不是文本内容。例如，//title 会返回"<title>Apache JMeter</title>"，而不是返回"Apache JMeter"。在这种情况下，//title/text()会返回"Apache JMeter"	是
Reference Name	JMeter 变量名，变量中存储有提取结果	是
XPath query	使用 XPath 语言查询的页面元素。可以返回多个匹配项	是
Default Value	没有找到匹配项时返回的默认值。如果对应节点没有值且没有选中“Return entire XPath fragment instead of text content?”复选框，则也会返回默认值	否

为了便于在 ForEach 控件之中使用，如下变量在测试元件返回时将被设置。

- refName：设置为第一个（或者唯一的）匹配项；如果没有匹配项，则设置为默认值。
- refName_matchNr：设置为匹配项的数目（可能为 0）；

- refName_n：n=1,2,3 等，设置为第一个、第二个、第三个匹配项等。

请注意：最后一个 refName_n 变量将被设置为 null。例如，假设这里有两个匹配项，那么 refName_3 将被设置为 null；如果这里没有匹配项，则 refName_1 将被设置为 null。

关于 XPath 的更多细节，请参考：http://www.topxml.com/xsl/xpathref.asp 和 http://www.w3.org/TR/xpath/。

下面是一些简单的例子。

- /html/head/title：从 HTML 响应中提取 title 元素。
- /book/page[2]：从 book 提取第二个页面。
- /book/page：从 book 提取所有页面。
- //form[@name='countryForm']//select[@name='country']/option[text()='Czech Republic'])/@value：从表单（name 属性为'countryForm'）中的 select 元素（name 属性为'country'）中的 option 元素（匹配 text 'Czech Republic'）中提取 value 属性。

小贴士　当“Use Tidy”被选中后，XML 结果文档可能与原始 HTML 响应有轻微差异：

- 所有页面元素和属性的名称都转换为小写。
- Tidy 尝试纠正不正确的嵌套元件。例如，原始（不正确的）ul/font/li 变为正确的 ul/li/font。

更多关于 Tidy 的信息请参考：http://jtidy.sourceforge.net/。

3. Result Status Action Handler

通过该测试元件，用户可以在相关采样器失败时停止线程或整个测试，如图 10-66 所示。

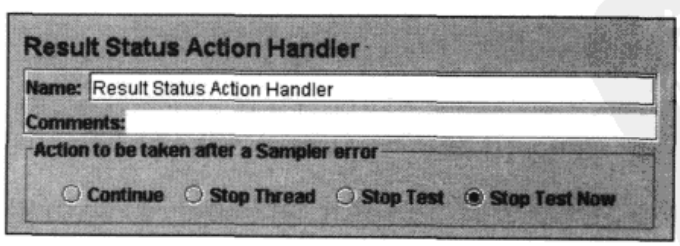


图 10-66　Result Status Action Handler

参数如表 10-45 所示。

表 10-45 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Action to be taken after a Sampler error	明确采样器发生错误后，会执行什么操作，既包括采样自身失败，也包括断言失败。可能的选择包括： ■ Continue：忽略错误继续测试 ■ Stop Thread：退出当前线程 ■ Stop Test：在任何当前采样结束后，整个测试将会停止 ■ Stop Test Now：整个测试会突然停止。如果可能的话，任何当前采样都会被中断	否

4. BeanShell PostProcessor

通过 BeanShell PostProcessor 可以在发生采样之后执行任何代码，如图 10-67 所示。

对于 JMeter 2.2 以后的版本，BeanShell PostProcessor 不再忽略响应数据长度为 0 的采样。

关于 BeanShell 的全面介绍，请访问 BeanShell 的 Web 站点 http://www.beanshell.org/。

该测试元件支持 ThreadListener 和 TestListener 两种方法。这应该在初始化文件中定义。请查看 BeanShellListeners.bshrc 文件中的定义范例。

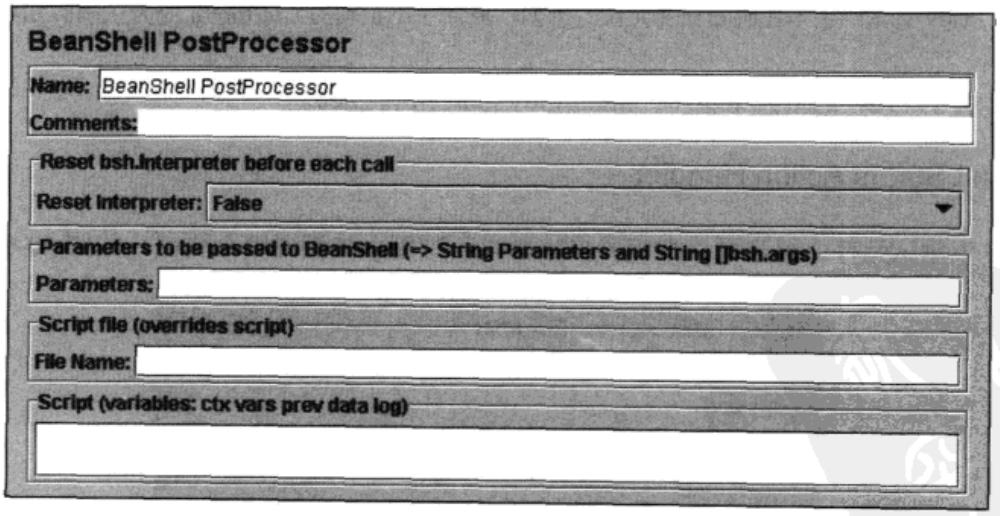


图 10-67 BeanShell PostProcessor

参数如表 10-46 所示。

表 10-46　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Reset bsh.Interpreter before each call	如果设置了该选项，那么就会为每个采样器重新创建解释器。这一点对长期运行的脚本很有必要。更多细节请参考详解 JMeter 最佳实践经验一节	是
Parameters	传递给 BeanShell 脚本的参数。参数存储在如下变量之中： ■ Parameters：包含有参数的字符串，作为单个变量存在 ■ bsh.args：包含有参数的字符串数组，以空格作为间隔	否
Script file	一个文件，其中包含有待运行的 BeanShell 脚本	否
Script	BeanShell 脚本。返回值将被忽略	是（除非提供了脚本文件）

初始化了如下变量，以供脚本使用。

- log - (Logger)：可以被用来记录日志文件。
- ctx - (JMeterContext)：可以通过它来访问 context。
- vars - (JMeterVariables)：提供读取/写入访问变量的方法。例如：

```
vars.get(key); vars.put(key,val); vars.putObject("OBJ1",new Object());
```

- props - JMeter 属性，例如：

```
props.get("START.HMS");
props.put("PROP1","1234");
```

- prev - (SampleResult)：可以被用来访问前一次的采样结果（如果存在）。
- data – (byte [])：可以被用来访问当前采样。

关于上述变量可以使用的方法，更多细节请参考对应 Javadoc。

如果定义了属性 beanshell.listener.init，那么它可以用于加载初始化文件。在该文件中可以定义 BeanShell 脚本将会用到的方法。

5. BSF PostProcessor

通过 BSF PostProcessor 可以在发生采样之后执行 BSF 脚本代码，如图 10-68 所示。

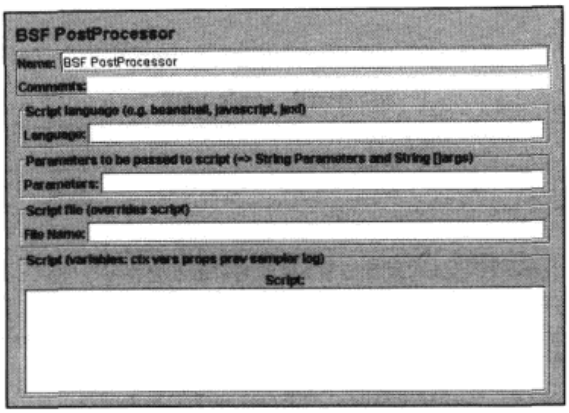


图 10-68　BSF PostProcessor

参数如表 10-47 所示。

表 10-47　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Language	指明 BSF 使用的语言	是
Parameters	传递给脚本的参数。这些参数被存储于如下变量之中： ■ Parameters：字符串，以单个变量的形式包含参数 ■ bsh.args：字符串数组包含有参数，以空格加以分隔	否
Script file	文件，包含有待运行的脚本	否
Script	待运行的脚本	是（除非提供了脚本文件）

BSF PostProcessor 将会使用 BSFEngine.exec()方法对脚本（或者文件）进行处理，脚本执行不会有返回值。

如下 BSF 变量将被初始化，以便后续被脚本使用。

- log - (Logger)：可以用来记录日志文件。
- Label：字符串标签。
- Filename：脚本文件名（如果存在）。
- Parameters：参数（字符串形式）。
- args[]：参数，字符串数组形式（以空格分隔）。
- ctx - (JMeterContext)：可以通过它来访问 context。
- vars - (JMeterVariables)：可以通过它来读取/访问 JMeter 变量。例如：

```
vars.get(key); vars.put(key,val); vars.putObject("OBJ1",new Object());
```

```
vars.getObject("OBJ2");
```

- props - JMeter 属性，例如：

```
props.get("START.HMS"); props.put("PROP1","1234");
```

- prev - (SampleResult)：可以通过它来访问前面的采样结果（如果存在）。
- sampler - (Sampler)：可以通过它来访问当前采样器。
- OUT - System.out，例如：

```
OUT.println("message") 。
```

关于上述变量可以使用的方法，更多细节请参考对应 Javadoc。

6. JSR223 PostProcessor

通过 JSR223 PostProcessor，可以在发生采样后执行 JSR223 脚本代码，如图 10-69 所示。更多细节，请参考 BSF PostProcessor 。

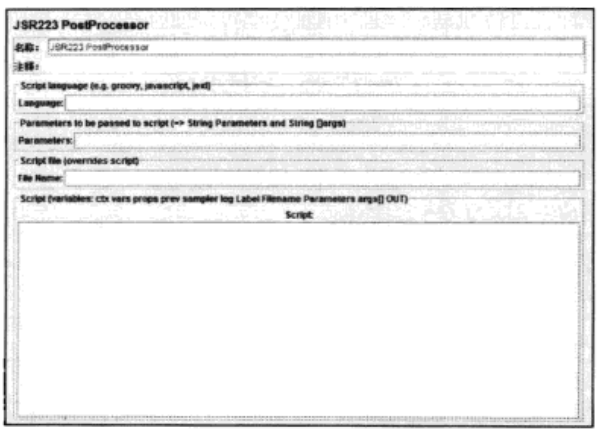


图 10-69　JSR223 PostProcessor

10.7　详解 JMeter 采样器

JMeter 的实际工作依赖于采样器完成。每个采样器（除了 Test Action 之外）都会产生一个或者多个采样结果。采样结果有各种属性（成功/失败、消耗时长、数据尺寸等），并可在各种监听器中查看。

1. FTP 请求（FTP Request）

通过该测试元件(如图 10-70 所示)，JMeter 可以发送一个 FTP "retrieve file"或者"upload file"

请求到某个 FTP 服务器。如果测试人员要发送多个请求到同一个 FTP 服务器，可以考虑使用 FTP Request Defaults 配置元件，如此测试人员就不用在每一个采样器中输入相同的信息。当下载文件时，可以将其存储在磁盘上（本地文件）或者放在响应数据中，或者两者都有。

延迟（Latency）被设置为登录所消耗的时长（JMeter 2.3.1 以后的版本）。

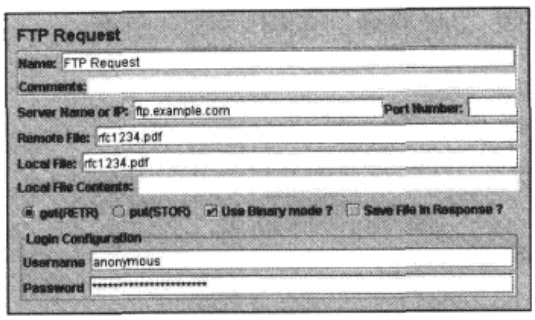


图 10-70　FTP 请求（FTP Request）

参数如表 10-48 所示。

表 10-48　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Server Name or IP	域名或者 FTP 服务器的 IP 地址	是
Port Number	使用的端口号。如果端口号> 0，那么将使用指定的端口，否则 JMeter 会使用默认的 FTP 端口	否
Remote File	获取的文件或者上载目标文件名	是
Local File	上载的文件，或者下载的目标（默认为远程文件名）	是，如果是上传(*)
Local File Contents	提供上传的内容，会覆盖属性 Local File	是，如果是上传(*)
get(RETR) / put(STOR)	是否接收或者上传一个文件	是
Use Binary mode ?	是否使用 Binary 模式（默认 Ascii）	是
Save File in Response ?	是否在响应数据中存储接收文件的内容。如果模式是 Ascii，那么其内容就可以在查看结果树（Tree View Listener）中看到	是，如果是上传(*)
Username	FTP 账户的用户名	通常情况下需要
Password	FTP 账户的密码。可以在测试计划中看到	通常情况下需要

2. HTTP 请求（HTTP Request）

通过该测试元件，JMeter 可以发送一个 HTTP/HTTPS 请求到 Web 服务器，如图 10-71

所示。另外，该测试元件还能解析 HTML 文件以便寻找图片及其他内嵌的资源，并发送 HTTP 请求来获取它们。如下类型的内嵌资源将被接收：

- Images
- Applets
- Stylesheets
- external scripts
- frames
- background images (body, table, TD, TR)
- background sound

默认的解析器是 htmlparser。这一点可以通过修改属性“htmlparser.classname”来修改，更多细节请查看 jmeter.properties 文件。

如果测试人员要向同一个 Web 服务器发送多个请求，可以考虑使用 HTTP Request Defaults 配置元件，如此测试人员就不用在每一个采样器中输入相同的信息。

另外，如果不想一个个手动添加 HTTP 请求，测试人员可以使用 JMeter 的 HTTP Proxy Server 来创建它们。这样做可以节约性能测试脚本开发时间，特别是测试人员需要创建很多 HTTP 请求，而 HTTP 请求携带很多参数时。

总共有如下 3 种版本的采样器。

- HTTP Request：使用默认的 Java HTTP 实现。
- HTTP Request HTTPClient：使用 Apache 通用 HTTPClient。
- AJP/1.3 Sampler：使用 Tomcat mod_jk 协议（允许使用 AJP 模式来测试 Tomcat，无须 Apache httpd），AJP 采样器不支持多文件上载；只有第一个文件才会被使用。

默认（Java）实现有一些限制：

- 无法控制连接如何重用。当连接被 JMeter 释放后，那么无法确定它是否被同一个线程所重用。
- API 最适合于单线程，各种设置（如代理）是通过系统属性定义的，因此会影响所有连接。
- 经由代理处理 HTTPS 存在一个 bug（对连接的处理不正确）。参考 JMeter Java bug 6226610 和 620835。

请注意：FILE 协议仅供测试使用。无论使用哪个 HTTP 采样器，它都由相同代码来处理。

如果请求要求服务器或者代理登录校验（如浏览器创建一个弹出对话框），那么测试人员需要添加一个 HTTP Authorization Manager 配置元件。对于普通登录（如用户在某个表单中输入登录信息），测试人员需要先搞清楚表单提交按钮会做什么，接着创建一个操作方式正确（通常为 POST）且携带表单定义参数的 HTTP 请求。如果页面使用 HTTP，那么测试人员可以使用 JMeter 代理来捕获登录序列。

JMeter 2.2 以前的版本，所有线程和采样器共用一个 SSL 内容。这样做不能真实模拟多个用户的压力。现在每个线程使用独立的 SSL 内容。如果要恢复早前的模式，可以设置 JMeter 属性：

```
https.sessioncontext.shared=true
```

JMeter 默认使用 SSL 协议层级 TLS。如果服务器要用另外的层级，例如 SSLv3，可以改变 JMeter 属性，例如：

```
https.default.protocol=SSLv3
```

JMeter 允许用户改变属性 https.socket.protocols，来生效其他协议。

如果请求用到了 Cookies，那么测试人员还需要 HTTP Cookie Manager。测试人员既可以将这些测试元件添加到线程组，也可以将它们添加到 HTTP 请求。如果测试人员有多个 HTTP 请求需要校验或者 Cookies，那么就将这些测试元件添加到线程组。如此，所有 HTTP 请求测试元件就会共享相同的校验管理器和 Cookie 管理器。

如果请求使用“URL 回写”技术来维护会话，那么请参考本书 3.6 节的内容。

图 10-71　HTTP 请求（HTTP Request）

参数如表 10-49 所示。

表 10-49　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Server Name or IP	Web 服务器的域名或者 IP 地址，例如 www.example.com（不包含前缀 http://）	是，除非 HTTP Request Defaults 有提供
Port Number	Web 服务器监听的端口号，默认为 80	否
Connect Timeout	连接超时时长，等待某个连接打开的毫秒数	否
Response Timeout	响应超时时长，等待某个响应的毫秒数	否
Server (Proxy)	代理服务器的域名或者 IP 地址（不包含前缀 http://）	否
Port Number	代理服务器监听的端口号	否
Username	代理服务器的用户名（可选）	否
Password	代理服务器的密码（可选）	否
Protocol	HTTP、HTTPS 或者 FILE。默认为 HTTP	否
Method	GET、POST、HEAD、TRACE、OPTIONS、PUT、DELETE	是
Content encoding	使用的内容编码（针对 POST 和 FILE）	是
Redirect Automatically	将潜在的 HTTP 协议处理器设置为自动跟随跳转，如此 JMeter 就不能看到它们，当然也不会作为采样出现。它应该仅仅被用于 GET 和 HEAD 请求。HTTPClient 采样器会拒绝将其用于 POST 或者 PUT。另外请注意下面关于 Cookie 和 Header 处理的信息	是
Follow Redirects	只有当“Redirect Automatically”不生效时，该复选框才起作用。如果选中了该复选框，则 JMeter 采样器会检查响应是否是跳转，如果是跳转就跟随它。初始跳转和接下来的响应会作为附加采样。父采样的 URL 和数据域从最终采样（非跳转）获取，不过父字节计数及消耗时间会包含所有采样。延迟取至初始响应（JMeter 2.3.4 以后版本（以前为 0））。请注意 HTTPClient 采样器可能会记录如下信息："Redirect requested but followRedirects is disabled"，这一点可以忽略 2.3.4 以后的版本，JMeter 会瓦解'/../segment'格式的路径，无论是在绝对还是相对跳转 URL。例如，http://host/one/../two => http://host/two。如果有必要的话可以禁止这一操作，通过设置 JMeter 属性 httpsampler.redirect.removeslashdotdot=false	是
Use KeepAlive	JMeter 设置连接：keep-alive header。它在默认 HTTP 实现中无法正常工作，原因在于连接重用不受用户控制。它能与 Jakarta HTTPClient 实现一起工作	是
Use multipart/form-data for HTTP POST	使用一个 multipart/form-data 或者 application/x-www-form-urlencoded POST 请求	是

续表

属性	描述	是否必需
Path	指向访问资源的路径（例如，/servlets/myServlet）。如果资源要求查询字符串参数，将它们添加到下面的“Send Parameters With the Request”片段中。作为特殊情况，如果路径以 http://或者 https://开始，那么就作为完整 URL 使用。在这种情况下，Server、Port 和 Protocol 参数都将被忽略；对于 GET 和 DELETE 方式参数将被忽略	是
Send Parameters With the Request	查询字符串会依据测试人员提供的参数列表来产生。每个参数都有名称和值、编码参数的选项、包含或者排除等号的选项（有些应用系统在值为空字符串的时候，不接受等号）。查询字符串会以正确的方式生成，这依赖于测试人员所选择的“Method”（如如果测试人员选择 GET 或者 DELETE，查询字符串将被附加到 URL 后面；如果选择 POST 或者 PUT，那么查询字符串会单独发送）。另外，如果测试人员使用 multipart form 发送一个文件，那么查询字符串会使用 multipart form 规范来创建。请参考下面关于参数处理的更多信息 另外，测试人员可以对每一个参数指定其是否应该被 URL 编码。如果测试人员不清楚这是什么含义，那么最好选中它。如果测试人员的参数值中包含字符&，或者空格，或者问号，那么通常都要求编码	否
File Path	发送文件的名称。如果保留为空，JMeter 不发送文件；如果填充了该项，则 JMeter 自动以 multipart form 形式发送请求。 如果是 POST 或者 PUT 请求，并发送单个文件且“name”属性被省略了（下面的 Parameter name:），那么文件作为请求的整个包体发送，例如没有添加封装，这样就允许发送任意包体。这一功能是 2.2 以后版本针对 POST 请求设计的，另外 2.3 以后版本也用于 PUT 请求。请参考下面关于参数处理的更多信息	否
Parameter Name	Web 请求参数“name”的值	否
MIME Type	MIME 类型（例如，text/plain）。如果是 POST 或者 PUT 请求且“name”属性被忽略了，或者请求包体仅由参数值构建，那么该域的值将作为请求头 content-type 的值	否
Retrieve All Embedded Resources from HTML Files	告诉 JMeter 解析 HTML 文件，并针对文件中引用的所有 images、Java applets、JavaScript files、CSSs 等发送 HTTP/HTTPS 请求。更多信息请参考下面的内容	否
Use as Monitor	配合 Monitor Results 监听器	是
Save response as MD5 hash?	如果选中了该复选框，那么服务器响应不会存储在采样结果中。JMeter 会计算响应数据的 32 字符 MD5 hash 值，代替服务器响应存储在采样结果中。这一选项针对大数据量测试	是

续表

Embedded URLs must match	如果存在，则该值必须是正则表达式，用于匹配任何可以找到的内嵌 URL。如果测试人员只想下载来自于 http://example.com/ 的内嵌资源，使用表达式 http://example\.com/.*	否
Source IP address	（仅针对 HTTP 请求 HTTPClient）覆盖这一请求的默认本地 IP 地址。JMeter 主机必须有多个 IP 地址（如 IP aliases 或者 network interfaces）。如果定义了属性 httpclient.localaddress，那么将被用于所有 HTTPClient 请求	否

当使用自动跳转时，只有针对初始 URL 发送 Cookies。从 Web 站点跳转到本地服务器，可能导致无法预料的情况发生。例如，www.example.com 跳转 www.example.co.uk。在这种情况下服务器可能会返回两个 URL 的 Cookies，但是 JMeter 只能看到最后一个主机的 Cookies，例如 www.example.co.uk。如果测试计划中的下一请求使用 www.example.com，而非 www.example.co.uk，那么使用的 Cookies 就不正确。同样，只有针对初始请求才会发送 Header，跳转是不会发送的。只有手工创建测试计划才会遇到这种问题，通过录制创建的测试计划会从跳转后的 URL 重新开始。

参数处理如下：

对于 POST 和 PUT 方式，如果没有文件发送，并且 parameter(s)的 name(s)被省略了，那么包体就由参数值串联而成，这样就允许发送任意形式的包体。JMeter 2.3 以后版本，如果设定了编码标志（Encode?），则参数值将被编码。另外，通过上面介绍的 MIME Type，测试人员可以控制发送请求头的 content-type。

Method 处理如下：

除了 PUT Method 不支持 multipart 请求之外，POST 和 PUT 两种 Method 工作方式相类似。PUT Method 的包体必须为下面的一种方式：

- 将包体定义为一个文件。
- 将包体定义为没有名称（name）的参数值集合。

如果测试人员在采样器或者 HTTP defaults 配置元件中定义了有名称（name）的参数，则什么也不会发送。GET 和 DELETE 两种 Method 工作方式相类似。

JMeter 2.1.1 及其以前版本，只有当响应的 content-type 为“text/html”时，才会仔细检查嵌入的资源。其他 content-type 则不会被认定为 HTML。JMeter 2.1.2 引入了一个新属性

HTTPResponse.parsers（解析器 ID 的列表），如 htmlParser 和 wmlParser。对于每一个找到的 ID，JMeter 进一步检查两个属性。

- id.types：content type 列表。
- id.className：用来提取嵌入资源的解析器。

更多关于这些属性的设置请参考 jmeter.properties 文件。如果没有定义属性 HTTPResponse.parser property，JMeter 会使用原来的方式，即仅检查 text/html 类响应。

模拟慢速连接（HttpClient only）：

```
# Define characters per second > 0 to emulate slow connections
#httpclient.socket.http.cps=0
#httpclient.socket.https.cps=0
```

3. JDBC 请求（JDBC Request）

通过该采样器（如图 10-72 所示），测试人员可以发送一个 JDBC 请求（一条 SQL 查询）到某个服务器。不过在使用该采样器之前，测试人员必须首先初始化 JDBC Connection Configuration 配置元件。

如果提供了变量名列表，则这些变量将由 Select 操作返回数据行的对应列（如果提供了变量名）的值来初始化，并且返回数据行的总数也会保存下来。例如，假设 Select 操作返回 2 行 3 列数据，而变量列表是 A,,C，那么如下变量将被初始化：

```
A_#=2 (number of rows)
A_1=column 1, row 1
A_2=column 1, row 2
C_#=2 (number of rows)
C_1=column 3, row 1
C_2=column 3, row 2
```

如果 Select 操作返回 0 行数据，那么 A_#和 C_#变量应该被设置为 0，其他变量都不应该被设置。

如果有必要，旧值将被清除，例如，如果前一个 Select 操作返回 6 行数据，而后一个 Select 操作仅返回 3 行数据，那么 4、5、6 行对应的变量将被删除。

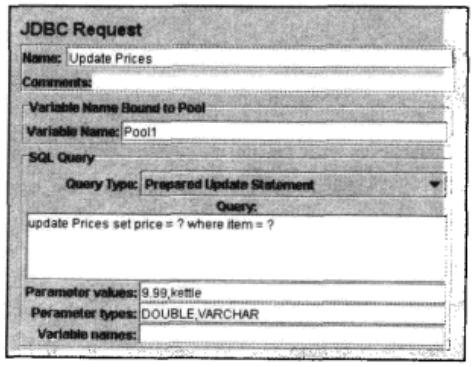


图 10-72　JDBC 请求（JDBC Request）

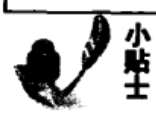
参数如表 10-50 所示。

表 10-50　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Variable Name	连接池绑定的 JMeter 变量的名称。该项参数必须与 JDBC Connection Configuration 配置元件中的“Variable Name”域相同	是
Query Type	根据操作类型来设置该参数： ■ Select Statement ■ Update Statement：这可以用于插入数据 ■ Callable Statement ■ Prepared Select Statement ■ Prepared Update Statement：这可以用于插入数据 ■ Commit ■ Rollback ■ Autocommit(false) ■ Autocommit(true) ■ Edit：这可以是变量引用，用于求上面某一项的值	是
SQL Query	SQL 查询语句。不要输入最后的分号。通常没有必要使用{和}来封装执行语句；不过有时候数据库使用非标准的语法，可能会这么做（当使用{}封装语句时，JDBC 驱动在必要的时候会自动转换语句） ■ select * from t_customers where id=23 ■ CALL SYSCS_UTIL.SYSCS_EXPORT_TABLE (null,?, ?, null, null, null) ■ Parameter values: tablename,filename ■ Parameter types: VARCHAR,VARCHAR 第二个例子假定测试人员使用 Apache Derby	是

续表

属性	描述	是否必需
Parameter values	以逗号分隔的参数值列表。使用]NULL[来指明一个 NULL 参数（如果有必要的话，可以通过定义属性“jdbcsampler.nullmarker”来改变 null 字符串）。如果任何参数值中包含逗号或者双引号，则参数列表必须用双引号来封装，而且任何内嵌的双引号都必须双倍书写，例如： "Dbl-Quote: "" and Comma: ," SQL 语句中有多少个占位符，这里就必须有多少个参数值	是，如果 Prepared 或者 callable 语句带有参数
Parameter types	逗号分隔的 SQL 参数列表（如 INTEGER、DATE、VARCHAR、DOUBLE）。这些类型在 java.sql.Types 类中定义，可以参考：http://download.oracle.com/javase/1.5.0/docs/api/java/sql/Types.html（请注意：JMeter 会使用 JVM 动态定义的类型，因此当测试人员使用不同的 JVM 时，请参考正确的说明文档）。如果可调用的语句有 INOUT 或者 OUT 参数，那么就必须使用前缀来加以标识，例如，使用"INOUT INTEGER"来代替"INTEGER"。如果没有特别指明，"IN"是默认有的，例如"DATE"等同于"IN DATE" 如果参数类型没有在 java.sql.Types 中找到，那么 JMeter 2.3.2 以后的版本还支持对应的整数值，例如，INTEGER = 4，测试人员就可以使用"INOUT 4" SQL 语句中有多少个占位符，这里就必须有多少个参数值	是，如果 Prepared 或者 callable 语句带有参数
Variable names	逗号分隔的变量名列表，变量用于存放 Select 操作返回的查询结果	否



JMeter 2.3.2 以后的版本使用 UTF-8 作为字符集编码，以前的版本使用操作平台默认字符集编码。

4. Java 请求（Java Request）

通过该采样器，测试人员可以控制一个 Java 类，该类实现了接口 org.apache.jmeter.protocol.java.sampler.JavaSamplerClient。通过自己编写该接口的实现，测试人员可以使用 JMeter 来实现多线程、输入参数控制和数据收集。

下拉菜单中提供了 JMeter 在其 Classpath 路径下能够找到的所有此类实现的列表。对应参数在下面的表格中展示，就像实现所定义的那样。这里提供了两个简单的例子（JavaTest 和 SleepTest）。

JavaTest 范例采样器对于检查测试计划很有用，因为它允许测试人员填充大多数数据域的

值，如图 10-73 所示。这些值就可以被 Assertions 等测试元件所使用。数据域中允许使用变量，因此变量的值可以直观地看到。

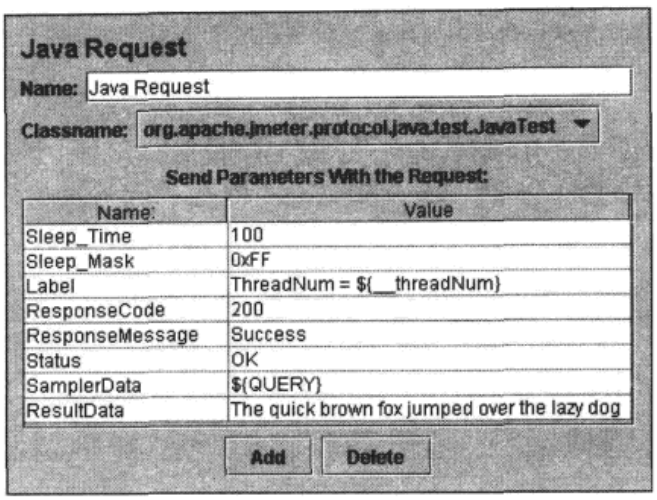


图 10-73　Java 请求（JavaRequest）

目前 Add/Delete 按钮不起任何作用。

参数如表 10-51 所示。

表 10-51　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Classname	JavaSamplerClient 接口的特定实现，用于被采样	是
Send Parameters With the Request	参数列表，参数会被传递给被采样类。所有参数都以字符串形式传递	否

Sleep Time 以如下方式计算:

Sleep_Time 单位毫秒; Sleep_Mask 被用于为 sleep_time 加上一个“随机”值: totalSleepTime = SleepTime + (System.currentTimeMillis() % SleepMask)

5. SOAP/XML-RPC 请求（SOAP/XML-RPC Request）

通过该采样器（如图 10-74 所示），测试人员可以发送一个 SOAP 请求到某个 Web Service。另外它还能基于 HTTP 发送 XML-RPC。它会创建一个 HTTP POST 请求，并将指定的 XML 作为 POST 内容。要改变默认的"Content-type" --"text/xml"，可以使用一个信息头管理器。请注意该采样器会使用所有来自信息头管理器的信息头。如果采样器中指定了 SOAPTction，则会覆盖

信息头管理器中的任何 SOAPAction。SOAP 采样器与 Web Service 采样器最大的区别在于，SOAP 使用原始的 POST 请求，并不要求遵循 SOAP 1.1。

对于 JMeter 2.2 以后的版本，该采样器不再默认使用截取编码（Chunked Encoding）。对于屏幕输入，它目前总是使用数据的大小。文件输入则使用 Java 确定的文件长度。在某些操作系统上，采样器并不适合于所有文件。在这种情况下，可以为其添加一个子信息头管理器，并将 Content-Length 设置为文件的实际长度，或者将 Content-Length 设置为-1（针对 Chunked Encoding）。

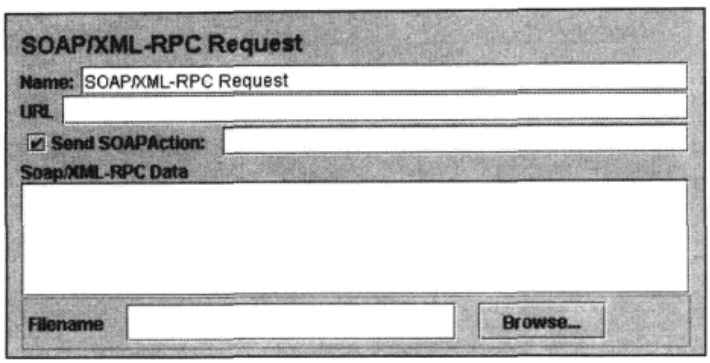


图 10-74　SOAP/XML-RPC 请求（SOAP/XML-RPC Request）

参数如表 10-52 所示。

表 10-52　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
URL	发送 SOAP 请求的目标 URL	是
Send SOAP Action	是否发送一个 SOAPTction 头（覆盖信息头管理器中的设置）	否
Soap/XML-RPC Data	Soap XML 消息，或者 XML-RPC 指令	否
Filename	该元件的描述性名称，用于在测试树中标识该元件	否

6．WebService(SOAP)请求（WebService(SOAP) Request）

该采样器针对运行.NET 1.0 和.NET 1.1 的 IIS Web Service 进行过测试。它针对 SUN JWSDP、IBM Web Services、Axis 和 gSoap toolkit（C/C++）也进行过测试。该采样器使用 Apache SOAP 驱动来序列化（serialize）消息，并使用正确的 SOAPAction 来设置消息头，如图 10-75 所示。目前该采样器不支持自动 WSDL 处理，原因在于 Apache SOAP 当前不提供对其的支持。IBM 和 Sun 都提供 WSDL 驱动。对于 POST 数据这里有 3 个选项，即 text area、external file 或者

directory。如果测试人员希望采样器随机地选择一条消息，就是用 directory 方式。其他情况下，使用 text area 或者一个 file（文件）。如果设置了 file 或者 path，那么采样器就不会使用 text area 中的消息。如果测试人员要用不同的编码方式来测试某个 SOAP Service，就使用 file 或者 path。如果测试人员直接将消息粘贴到文本域（text area）中，那么就无法保持原有的编码方式，或者导致某些错误发生。可以把消息以正确的编码方式存放到文件中，而采样器将以 java.io.FileInputStream 方式读取。

该采样器需要两个 jar 文件 mail.jar 和 activation.jar。这是因为 Apache SOAP 要求使用这些库文件。由于 mail.jar 和 activation.jar 由 Sun 公司发布，因此测试人员不得不单独下载它们。

请注意，该采样器会自己使用测试人员通过命令行传递给 JMeter 的代理主机和代理端口，前提是采样器界面的对应域为空。如果采样器的代理主机和代理端口两个文本域中有值，则它会使用用户在界面中指定的值。这也许并不符合测试人员的期望。

在默认情况下，Web Service 采样器会设置 SOAPHTTPConnection.setMaintainSession (true)。如果测试人员需要维持会话，可以添加一个空的信息头管理器。采样器使用信息头管理器来存储 SOAPHTTPConnection 对象，原因在于 Apache Soap 版本没有提供一个简便的方法来获取和设置 Cookies。

请注意，如果测试人员正在使用 CSVDataSet，就不要选中“Memory Cache”复选框。如果选中了“Memory Cache”复选框，则不会循环到下一个值。这就意味着所有请求都会使用第一个值。

请确保使用<soap:Envelope 而非<Envelope 。例如：

```
<?xml version="1.0" encoding="utf-8"?>
<soap:Envelope
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/">
<soap:Body>
<foo xmlns="http://clients-xlmns"/>
</soap:Body>
</soap:Envelope>
```

目前使用的 SOAP 库不支持 SOAP 1.2，仅支持 SOAP 1.1。另外，该库也不支持访问 HTTP 响应代码（如 200）或者消息（如 OK）。为了规避这一问题，JMeter 2.3.2 以后的版本会检查返回消息的长度。如果长度为 0，则请求被标记为失败（failed）。

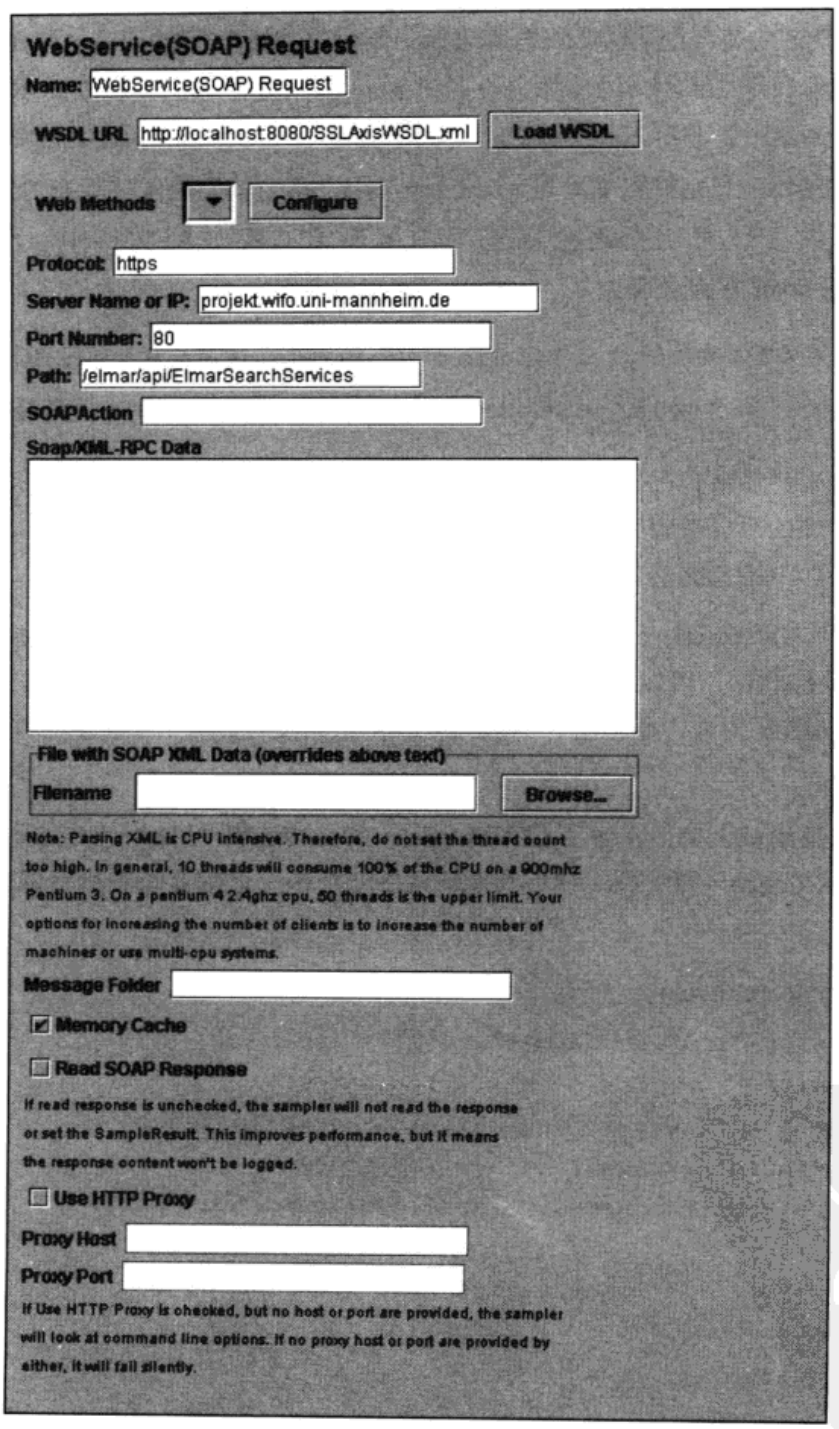


图 10-75　WebService(SOAP)请求（WebService(SOAP) Request）

参数如表 10-53 所示。

表 10-53　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
WSDL URL	WSDL URL 携带对服务的描述。JMeter 2.3.1 以后的版本支持文件：本地 WSDL 协议文件	否
Web Methods	当单击“Load WSDL”按钮时，会从 WSDL 中弹出。选择某个 Method，并单击“Configure”按钮，以便弹出 Protocol、Server、Port、Path 和 SOAPAction 域的值	否
Protocol	HTTP 或者 HTTPS 都是可接受的协议	是
Server Name or IP	主机名或者 IP 地址	是
Port Number	端口号	是
Path	Web Service 的路径	是
SOAPAction	在 Web Service 描述或者 WSDL 中定义的 SOAPAction	是
Soap/XML-RPC Data	Soap XML 消息	是
Soap file	包含 soap 消息的文件	否
Message Folder	包含 soap 文件的文件夹	否
Memory cache	当使用外部文件时，设置该选项将导致文件被处理一次，并存储结果。如果这里有很多不同的大文件，那么可能会占用很多内存	是
Use HTTP Proxy	如果选择使用 HTTP 代理，则选中该复选框	否
Proxy Host	代理主机名	否
Proxy Port	代理端口号	否

7．LDAP 请求（LDAP Request）

通过该采样器，测试人员可以发送不同的 LDAP 请求（Add、Modify、Delete and Search）到某个 LDAP 服务器，如图 10-76 所示。

如果测试人员要发送多个请求到相同的 LDAP 服务器，可以考虑使用 LDAP Request Defaults 配置元件，这样测试人员就不用在每个 LDAP 请求中输入相同的信息。

另外还可以使用 Login Config Element 来完成登录。

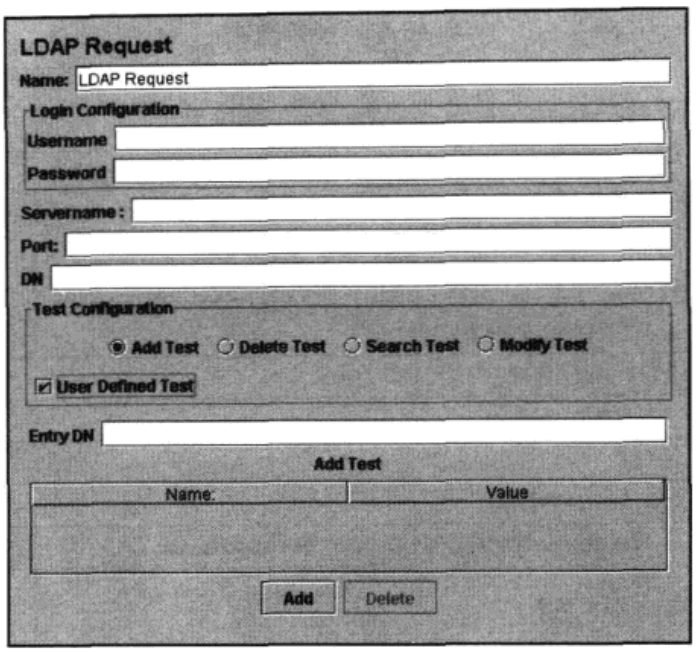


图 10-76　LDAP 请求（LDAP Request）

这里有两种创建测试案例用于测试 LDAP 服务器的方法。

- 采样器内置测试案例（Inbuilt Test）。
- 用户自定义测试案例（User Defined Test）。

这里有 4 种测试 LDAP 的场景，如下：

1）Add Test

（1）采样器内置测试案例（Inbuilt Test）。这会在 LDAP 服务器中添加一个预定义的条目，并计算执行时间。测试结束后，添加的条目会从 LDAP 服务器中删除。

（2）用户自定义测试案例（User Defined Test）。这会在 LDAP 服务器中添加一个用户自定义的条目。用户必须在表格中填入所有属性。条目的内容会从表格中采集。另外还会计算执行时间。测试结束后，添加的条目会从 LDAP 服务器中删除。

2）Modify Test

（1）采样器内置测试案例（Inbuilt Test）。这会在 LDAP 服务器中先创建一个预定义的条目，接着对其进行修改，并计算执行时间。测试结束后，创建的条目会从 LDAP 服务器中删除。

（2）用户自定义测试案例（User Defined Test）。这会在 LDAP 服务器中修改某个条目。用户必须在表格中填入所有属性。条目的内容会从表格中采集。另外还会计算执行时间。测试结

束后，对应条目不会从 LDAP 服务器中删除。

3）Search Test

（1）采样器内置测试案例（Inbuilt Test）。这会先创建一个预定义的条目，接着查找对应的属性，并计算查询的执行时间。测试结束后，创建的条目会从 LDAP 服务器中删除。

（2）用户自定义测试案例（User Defined Test）。这会基于搜索基础（由用户定义）来搜索用户定义的条目（搜索过滤器）。条目必须在 LDAP 服务器中存在。另外还会计算执行时间。

4）Delete Test

（1）采样器内置测试案例（Inbuilt Test）。这会先创建一个预定义条目，接着从 LDAP 服务器中删除该条目，并计算删除的执行时间。

（2）用户自定义测试案例（User Defined Test）。这会删除用户定义的条目，对应条目必须在 LDAP 服务器中存在。另外还会计算执行时间。

参数如表 10-54 所示。

表 10-54　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Server Name or IP	LDAP 服务器的域名或者 IP 地址。JMeter 会假定 LDAP 服务器监听默认端口（389）	是
Port	默认端口（389）	是
DN	与服务器通信的 DN	是
Username	LDAP 服务器的用户名	通常需要
Password	LDAP 服务器的密码	通常需要
Entry DN	要创建或者修改的内容对应的名称；可以不为空。例如：如果测试人员想添加 cn=apache,ou=test，那么需要在表格中添加 name=cn, value=apache	是
Delete	待删除内容的名称；可以不为空	是
Search base	待搜索内容或者对象的名称	是
Search filter	搜索使用的过滤器表达式；可以不为空	是
Add Test	添加到指定内容对象中的名称和值对	是
Modify Test	在指定内容对象中添加或者修改的名称和值对	是

8. LDAP 扩展请求（LDAPEx Request）

通过该采样器可以发送 8 个不同的 LDAP 请求到某个 LDAP 服务器，如图 10-77 所示。它

是 LDAP 采样器的扩展版本，尽管它更难配置，但是却更贴近于真实的 LDAP 会话。

如果测试人员要发送多个请求到相同的 LDAP 服务器，可以考虑使用 LDAP Extended Request Defaults 配置元件，这样测试人员就不用在每个 LDAP 请求中输入相同的信息。

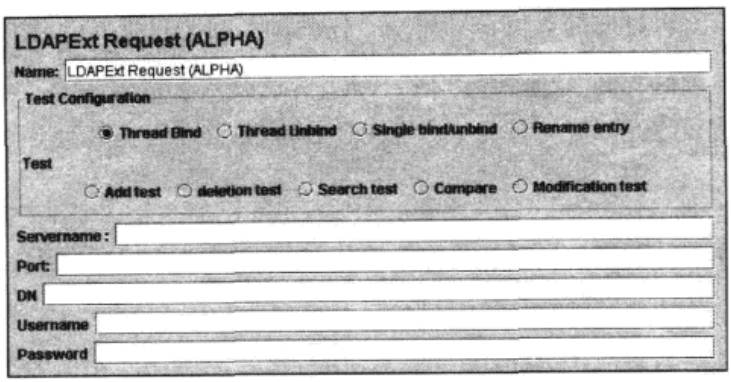


图 10-77　LDAP 扩展请求（LDAPEx Request）

这里共定义了 9 种测试选项，如下：

1）Thread Bind

任何 LDAP 请求都是 LDAP 会话的一部分，因此最先做的事就是启动一个到 LDAP 服务器的会话。启动会话会用到 Thread Bind，等同于 LDAP"bind"操作。用户被要求提供 Username (Distinguished name)和 Password，被用于初始化会话。如果没有指定 Password 或者指定了错误的 Password，则会启动一个匿名的会话。请注意，省略 Password 不会导致测试失败，但是错误的 Password 就会导致测试失败。

参数如表 10-55 所示。

表 10-55　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Servername	LDAP 服务器的名称（或者 IP 地址）	是
Port	LDAP 服务器监听的端口号。如果省略了该参数，JMeter 就假设 LDAP 服务器监听默认端口（389）	否
DN	任何后续操作都会用到的基础对象的 Distinguished name（DN）。它可以被用做所有操作的起点。测试人员不能在高于 DN 的级别上启动任何操作	否
Username	测试人员想绑定的用户的完整 Distinguished Name（DN）	否
Password	用户对应的密码。如果省略了它，就会导致匿名绑定。如果密码不正确，采样器就会返回一个错误，并回到匿名绑定	否

2）Thread Unbind

这是用于结束会话的一个简单操作，它等同于 LDAP"unbind"操作。

参数如表 10-56 所示。

表 10-56　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否

3）Single bind/unbind

这是 LDAP"bind"和"unbind"操作的组合，它可以被用于针对任何用户的校验请求/密码检查。它会打开一个新的会话，仅仅检查 user/password 组合的正确性，接着再次结束会话。

参数如表 10-57 所示。

表 10-57　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Username	测试人员想要绑定的用户的完整 Distinguished Name	是
Password	用户对应的密码。如果省略了它，就会导致匿名绑定。如果密码不正确，采样器就会返回一个错误	否

4）Rename entry

这对应于 LDAP"moddn"操作，它可以被用于重命名某个条目，还可以将某个条目及其下子树整体移到 LDAP 树中的某个不同地方。

参数如表 10-58 所示。

表 10-58　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Old Entry Name	测试人员希望重命名或者移动的对象的当前 Distinguished Name（DN），相对于 Thread Bind 操作中指定的 DN	是
New Distinguished Name	测试人员希望重命名或者移动的对象的新 Distinguished Name（DN），相对于 Thread Bind 操作中指定的 DN	否

5）Add test

这对应于 LDAP"add"操作，它可以被用来添加任何类型的对象到 LDAP 服务器。

参数如表 10-59 所示。

表 10-59　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Entry DN	测试人员期望添加对象的 Distinguished Name（DN），相对于 Thread Bind 操作中指定的 DN	是
Add test	测试人员用于对象的属性及属性值列表。如果要添加多值属性，测试人员需要多次添加同一个属性及其关联值到列表中	是

6）delete test

这对应于 LDAP"delete"操作，它可用于从 LDAP 树中删除一个对象。

参数如表 10-60 所示。

表 10-60　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Delete	测试人员想要删除对象的 Distinguished Name（DN），相对于 Thread Bind 操作中指定的 DN	是

7）Search test

这对应于 LDAP"search"操作，并被用于定义搜索。

参数如表 10-61 所示。

表 10-61　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Search base	测试人员希望在其中搜索的子树的 Distinguished Name（DN），相对于 Thread Bind 操作中指定的 DN	否
Search filter	搜索过滤器，必须使用 LDAP 语法来指定	是
Scope	0 对应基本对象，1 对应第一层，2 对应子树搜索（默认为 0）	否
Size limit	指定测试人员期望从服务器接收到的结果的最大数目（默认为 0，意味着没有限制）。当采样器遇到结果最大数目后就会失败，错误码为 4	否
Time limit	指定服务器在执行搜索时，允许耗费的最大 CPU 时长（单位为毫秒）。请注意，这里并不涉及响应时长（responsetime）（默认为 0，意味着没有限制）	否
Attributes	指定测试人员希望在结果中包含的属性，以分号加以分隔。如果为空域，则返回所有属性	否
Return object	对象应该被返回（true）还是不返回（false）。默认为 false	否
Dereference aliases	如果为 true，则废弃别名；如果为 false，则不会跟随它们（默认为 false）	否

8）Modification test

这对应于 LDAP"modify"操作，它可以被用于修改某个对象。它能添加、删除或者替换某个属性的值。

参数如表 10-62 所示。

表 10-62　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Entry name	测试人员希望修改的对象的 Distinguished Name（DN），相对于 Thread Bind 操作中指定的 DN	是
Modification test	属性—值—操作码组合。操作码可以是任何正确的 LDAP 操作码（add、delete/remove 或者 replace）。如果测试人员不为 delete 操作指定一个值，那么指定属性的所有值都将被删除。如果测试人员为 delete 操作指定一个值，那么只有指定的值会被删除。如果值不存在，则采样器会导致测试失败	是

9）Compare

这对应于 LDAP"compare"操作，它可以将指定属性的值与某些已知值进行比较。在实际工作中，它通常用于检查某个人是否是某些组的成员。在这种情况下，测试人员可以将用户的 DN 作为已知值，与 groupOfNames 类型对象的“member”属性值进行比较。如果比较操作失败了，则测试就会失败，错误码为 49。

参数如表 10-63 所示。

表 10-63　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Entry DN	测试人员想比较属性值的对象的当前 Distinguished Name（DN），相对于 Thread bind 操作中指定的 DN	是
Compare filter	格式如“attribute=value”	是

9．访问日志采样器（Access Log Sampler）（Alpha Code）

访问日志采样器（Access Log Sampler）被设计用来读取访问日志（如图 10-78 所示），并产生 HTTP 请求。测试人员可能不太了解什么是访问日志，其实访问日志就是 Web 服务器维护的一种日志记录，其中记录了 Web 服务器接收到的每一个请求，包括所有图片和 HTML 文件。当前实现已经完整，但是有些特性还没有生效。这里有一个为访问日志解析器设计的过滤器，但目前还不能与前置处理器做关联。以后的 JMeter 版本，应该可以实现这一功能。

Tomcat 使用通用格式的访问日志。这就意味着对于任何使用通用日志格式的 Web 服务器，

都可以适配访问日志采样器（Access Log Sampler）。使用通用日志格式的服务器包括 Tomcat、Resin、Weblogic 和 SunOne。通用日志格式形如：

```
127.0.0.1 - - [21/Oct/2003:05:37:21 -0500] "GET /index.jsp?
%2Findex.jsp= HTTP/1.1" 200 8343
```

解析器的当前实现，仅着眼于引号内的内容,其他内容都将被忽略掉。例如，响应代码将被解析器完全忽略掉。未来的设计也许会过滤掉响应代码不为 200 的条目。测试人员可以很容易地扩展该采样器，只需实现两个必要的接口：

```
org.apache.jmeter.protocol.http.util.accesslog.LogParser
org.apache.jmeter.protocol.http.util.accesslog.Generator
```

当前访问日志采样器（Access Log Sampler）的实现使用 Generator 来创建一个新的 HTTP 采样器。服务器名（Servername）、端口号（Port Number）和获取的图片由访问日志采样器来设置。接下来，解析器以整数 1 来调用，告诉它解析一行条目。在此之后，调用 HTTPSampler.sample()来发送请求。

```
samp = (HTTPSampler) GENERATOR.generateRequest();
samp.setDomain(this.getDomain());
samp.setPort(this.getPort());
samp.setImageParser(this.isImageParser());
PARSER.parse(1);
res = samp.sample();
res.setSampleLabel(samp.toString());
```

日志解析器要求的方法包括：setGenerator(Generator)和 parse(int)。实现 Generator 接口的类应该提供所有方法的具体实现。关于实现任何一种接口的例子，可以参考 StandardGenerator 和 TCLogParser。

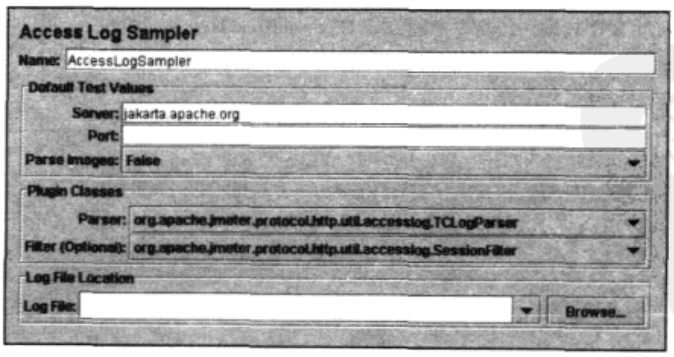


图 10-78　访问日志采样器（Access Log Sampler）

参数如表 10-64 所示。

表 10-64　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Server	Web 服务器的域名或者 IP 地址	是
Port	Web 服务器监听的端口	否（默认为 80）
Log parser class	日志解析类负责解析日志文件	是（默认提供）
Filter	过滤器类被用来挑选出特定日志行	否
Location of log file	访问日志文件的位置	是

TCLogParser 独立地为每个线程处理访问日志。SharedTCLogParser 和 OrderPreservingLogParser 共享对文件的访问，例如，每个线程获取文件的下一条目。

SessionFilter 的作用是跨线程处理 Cookies。它并不过滤任何条目，但会修改 Cookie 管理器，因此指定 IP 的 Cookies 在某一时间会被单个线程所处理。如果有两个线程尝试处理来自相同客户端 IP 地址的采样，那么其中一个采样会等待另一个采样完成。

LogFilter 的作用是通过文件名（Filename）和正则表达式（Regex）来过滤访问日志条目，以及替换文件扩展名。不过目前不能通过 GUI 来配置它，因此无法实际使用。

10. BeanShell 采样器（BeanShell Sampler）

通过该采样器（如图 10-79 所示），测试人员可以使用 BeanShell 脚本语言来编写一个采样器。

关于如何使用 BeanShell 的全部细节，请参考 BeanShell Web 站点：http://www.beanshell.org/。

该测试元件支持 ThreadListener 和 TestListener 方法。这些应该在初始化文件中定义。参考 BeanShellListeners.bshrc 文件以便查看更多定义范例。

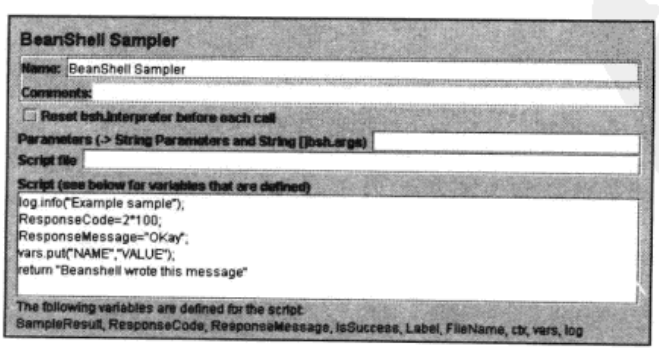


图 10-79　BeanShell 采样器（BeanShell Sampler）

参数如表 10-65 所示。

表 10-65 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Reset bsh.Interpreter before each call	如果选中了该复选框，那么就会为每个采样器重新创建解释器。这一点对长期运行的脚本很有必要。更多细节请参考详解 JMeter 最佳实践经验一节	是
Parameters	传递给 BeanShell 脚本的参数。这是针对 Script file 设计的；至于在 GUI 中定义的脚本，测试人员可以在其中使用变量或者函数引用。参数存储在如下变量之中： ■ Parameters：包含有参数的字符串，作为单个变量存在 ■ bsh.args：包含有参数的字符串数组，以空格作为间隔	否
Script file	一个文件，其中包含有待运行的 BeanShell 脚本。返回值将作为等待的毫秒数	否
Script	BeanShell 脚本。返回值将作为等待的毫秒数	是（除非提供了脚本文件）

每个采样器实例都有自己的 BeanShell 解释器，而这些采样器只能由单个线程调用。

如果定义了属性“beanshell.sampler.init”，那么它将作为一个源文件名传递给解释器。这可以被用来定义通用方法和变量。在 bin 目录中还有一个采样初始化文件：BeanShellSampler.bshrc。

如果提供了脚本文件，那么就使用脚本文件，否则使用 GUI 上录入的脚本。

唤醒脚本之前，在 BeanShell 解释器中初始化了一些变量：

参数域的内容被放置在变量“Parameters”中。字符串同样被分隔为不同的标记，使用单个空格作为分隔符，而结果列表被存放在字符串数组 bsh.args 中。

- log - (Logger)：可以被用来记录日志文件。
- Label：采样器标签。
- Filename：文件名，如果存在。
- Parameters：来自于 Parameters 域的文本。
- bsh.args：各项参数，像上面那样描述划分。
- SampleResult：指向当前采样结果的指针。
- ResponseCode = 200。
- ResponseMessage ="OK"。

- IsSuccess = true。
- ctx - (JMeterContext)：可以通过它来访问 context。
- vars - (JMeterVariables)：提供读取/写入访问变量的方法。例如：

```
vars.get(key); vars.put(key,val); vars.putObject("OBJ1",new Object());
```

- props - JMeter 属性，例如：

```
props.get("START.HMS"); props.put("PROP1","1234");
```

当脚本执行完毕后，控制权又返回到采样器，并将如下脚本变量的内容复制到采样结果的对应变量中：

- ResponseCode，例如 200。
- ResponseMessage，例如"OK"。
- IsSuccess，例如 true/false。

采样结果的响应数据通过脚本的返回值来设置。从版本 2.1.2 开始，如果脚本返回为空，那么可以使用方法 SampleResult.setResponseData(data) 来直接设置响应数据，该数据可以是字符串或者字节数组。数据类型默认为“text”，不过可以通过 SampleResult.setDataType(SampleResult.BINARY)方式将数据类型设置为二进制（Binary）。通过 SampleResult 变量，脚本可以全面访问采样结果（SampleResult）中的所有数据域和方法。例如，脚本可以访问 setStopThread(boolean)和 setStopTest(boolean) 方法。这里有一个简单的脚本例子：

```
if (bsh.args[0].equalsIgnoreCase("StopThread")) {
    log.info("Stop Thread detected!");
    SampleResult.setStopThread(true);
}
return "Data from sample with Label "+Label;
//or, since version 2.1.2
SampleResult.setResponseData("My data");
return null;
```

另外一个例子，确保 jmeter.properties 文件中定义有属性：

```
beanshell.sampler.init=BeanShellSampler.bshrc
```

那么如下脚本将展示响应数据域中所有变量的值：

```
return getVariables();
```

关于各种类（JMeterVariables、SampleResult 等）的可用方法的细节，请参考 Javadoc 或者源代码。请小心使用这些方法，错误使用任何方法可能导致很难找到错误的原因。

11. BSF 采样器（BSF Sampler）

通过该采样器，测试人员可以使用 BSF 脚本语言来编写一个采样器，如图 10-80 所示。

请参考 Apache Bean Scripting Framework 站点，以便获取更多关于支持语言的细节（http://jakarta.apache.org/bsf/index.html）。测试人员可能会需要下载一些关于该语言的 jar 包；它们应该被放置在 JMeter 的 lib 目录中。

默认情况下，JMeter 支持 JavaScript、jexl (JMeter version 2.3.2 and later)、xslt 语言。

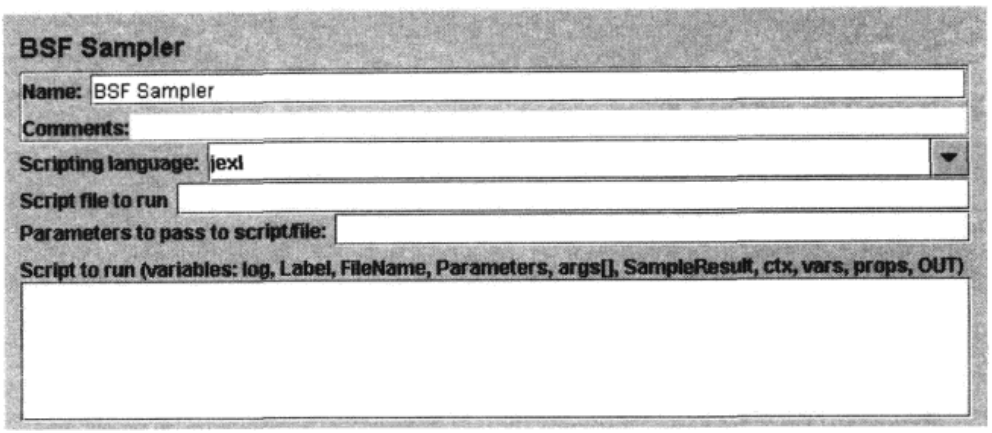


图 10-80 BSF 采样器（BSF Sampler）

参数如表 10-66 所示。

表 10-66 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Language	使用的 BSF 脚本语言名称。请注意，并不是所有下拉列表中的语言都默认支持。默认支持 jexl、JavaScript、xslt。用户在 JMeter lib 目录中安装合适的 jar 包后，其他语言才可用	是
Script file	文件名，文件中包含有待运行的脚本	否
Parameters	传递给脚本文件或者脚本的参数列表	否
Script	待运行脚本	是（除非提供了脚本文件）

唤醒脚本之前，初始化了一些变量。请注意这些是 BSF 变量，例如，它们可以在脚本中直接使用。

- log - (Logger)：可以用来记录日志文件。
- Label：采样器标签。

- Filename：文件名（如果存在）。
- Parameters：来自于 Parameters 域的文本。
- Args：各项参数，像上面那样描述划分。
- SampleResult：指向当前采样结果的指针。
- ctx - (JMeterContext)：可以通过它来访问 context。
- vars - (JMeterVariables)：可以通过它来读取/访问 JMeter 变量：

```
vars.get(key); vars.put(key,val); vars.putObject("OBJ1",new Object());
vars.getObject("OBJ2");
```

- props - JMeter 属性，例如：

```
props.get("START.HMS"); props.put("PROP1","1234");
```

- OUT - System.out，例如：

```
OUT.println("message") 。
```

采样结果的响应数据通过脚本的返回值来设置。从版本 2.1.2 开始，如果脚本返回空，那么可以使用方法 SampleResult.setResponseData(data) 来直接设置响应数据，该数据可以是字符串或者字节数组。数据类型默认为“text”，不过可以通过 SampleResult.setDataType(SampleResult.BINARY)方式将数据类型设置为二进制（Binary）。

通过 SampleResult 变量，脚本可以全面访问采样结果（SampleResult）中的所有数据域和方法。例如，脚本可以访问 setStopThread(boolean)和 setStopTest(boolean) 方法。

与 Beanshell 采样器不同的是，BSF 采样器并不通过脚本变量来设置 ResponseCode、ResponseMessage 和 sample status。当前修改这些值的唯一办法，就是通过 SampleResult 的方法：

- `SampleResult.setSuccessful(true/false)`
- `SampleResult.setResponseCode("code")`
- `SampleResult.setResponseMessage("message")`

12. JSR223 采样器（JSR223 Sampler）

使用 JSR223 脚本语言，JSR223 Sampler 可以被用来产生采样，如图 10-81 所示。更多细节请参考 BSF Sampler。

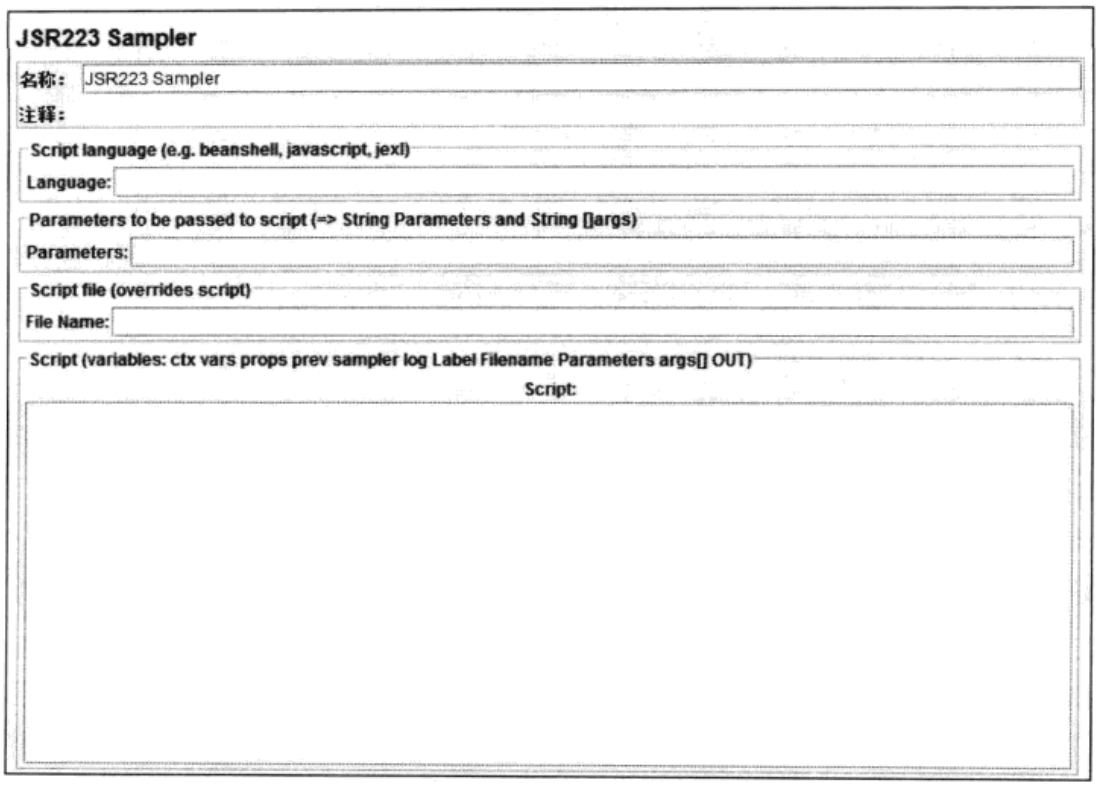


图 10-81　JSR223 采样器（JSR223 Sampler）

13. TCP 采样器（TCP Sampler）

TCP 采样器会打开一个到指定服务器的 TCP/IP 连接，如图 10-82 所示。它接着会发送文本，并等待一个响应。如果选中了“Re-use connection”复选框，则连接将在同一个线程的不同采样器间共享，并使用完全一致的主机名字符串和端口号。不同的主机/端口组合使用不同的连接，不同线程也一样。

如果检测到错误发生，或者没有选中“Re-use connection” 复选框，套接字（Socket）将被关闭。下一次采样时另一个套接字（Socket）将被重新打开。

如下属性可以被用来控制采样器的操作。

- tcp.status.prefix：状态码之前的文本。
- tcp.status.suffix：状态码之后的文本。
- tcp.status.properties：属性文件名，用于将状态码转换为消息。
- tcp.handler：TCP 处理类的名称（默认为 TCPClientImpl），只有当采样器 GUI 中没有指定时才生效。

处理连接的类由 GUI 定义，如果没有定义则使用属性 tcp.handler 的设置。如果还是没有找到，则在 org.apache.jmeter.protocol.tcp.sampler 包中搜索。

用户还可以提供自己的实现，但是实现类必须由 org.apache.jmeter.protocol.tcp.sampler.TCPClient 派生。当前提供的实现类包括 TCPClientImpl、BinaryTCPClientImpl、LengthPrefixedBinaryTCPClientImpl。

这些实现的具体表现如下：

1）TCPClientImpl

该实现非常基础。当读取响应时，它会一直读取到 end of line byte（如果有在属性 tcp.eolByte 中定义），否则它会一直读取到输入流结尾。

2）BinaryTCPClientImpl

该实现会将 GUI 输入（必须为十六进制编码字符串）转换为二进制，并在读取响应时做相反的操作。当读取响应时，它会一直读取到消息的最后一个字节（如果有在属性 tcp.BinaryTCPClient.eomByte 中定义），否则它会一直读取到输入流结尾。

3）LengthPrefixedBinaryTCPClientImpl

该实现会扩展 BinaryTCPClientImpl，即在二进制消息数据前加一个二进制长度字节前缀。长度前缀默认为 2 字节。测试人员可以通过设置属性 tcp.binarylength.prefix.length 来改变前缀的字节数。

（1）Timeout 处理，如果设置了超时时长，那么超时后读取将被终止。如果测试人员使用了 eolByte/eomByte，那么请确保超时时长足够长，否则读取会被提前终止。

（2）Response 处理，如果定义了 tcp.status.prefix，那么将在响应消息中搜索其后的文本，直到遇到后缀为止。如果找到了任何这类文本，那么它将被用来设置响应码。响应消息将从属性文件（如果有提供）中提取。例如，如果前缀="["而后缀="]"，那么如下响应：[J28] XI123,23,GBP,CR 的响应码为 J28。当前响应码处于 400～499 和 500～599 范围内，被认为是失败；所有其他情况被认为是成功。

登录用户名/密码不被 JMeter 提供的 TCP 实现所使用。

在测试运行结尾处套接字（Socket）将被关闭。

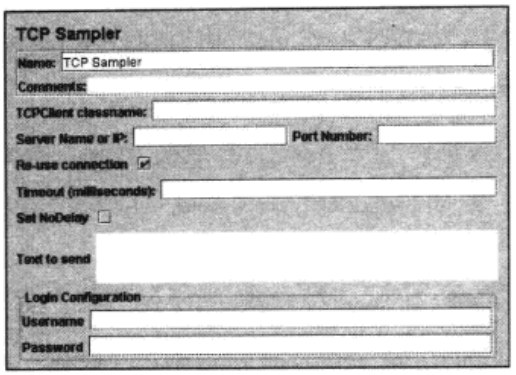


图 10-82　TCP 采样器（TCP Sampler）

参数如表 10-67 所示。

表 10-67　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
TCPClient classname	TCPClient 类的名称，默认采用属性 tcp.handler 的值，如果没有该属性则为 TCPClientImpl	否
Server Name or IP	TCP 服务器的名称或者 IP 地址	是
Port Number	使用的端口号	是
Re-use connection	如果选中该复选框，则连接将一直保持打开状态。否则数据读取后，连接将被关闭	是
Timeout（milliseconds）	响应超时时长	否
Set NoDelay	参考 java.net.Socket.setTcpNoDelay()。如果选中该复选框，则 Nagle 算法会失效，否则将使用 Nagle 算法	是
Text to send	待发送文本	是
Username	登录用户名（不为默认实现所使用）	否
Password	密码（不为默认实现所使用）	否

14. JMS Publisher

该采样器目前是 BETA 代码（正在改进过程中）。

JMS Publisher 会发送消息到指定目标（topic/queue），如图 10-83 所示。对于不熟悉 JMS 的读者，简单说明一下，JMS 就是 J2EE 的消息规范。市面上有多种 JMS 服务器可供选择，其中还包括很多开源选择。

JMeter 并不包含任何 JMS 实现 jar 包；测试人员必须从 JMS 提供方获取（下载）它们，并放置在 JMeter 的 lib 目录中。

图 10-83　JMS Publisher

参数如表 10-68 所示。

表 10-68　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
use Jndi.properties file	使用 jndi.properties。请注意该文件必须位于 classpath 之上，例如，通过更新 JMeter 属性 user.classpath 来实现。如果没有选中该复选框，JMeter 就使用"JNDI Initial Context Factory"和"Provider URL"域来创建连接	是
Initial Context Factory	Context Factory 的名称	否
Provider URL	JMS 提供者的 URL	是，除非使用 jndi.properties
Destination	消息目标（Topic 或者 Queue 名称）	是
Use Authentication	设置 JMS 提供者是否要求校验	是
User	用户名	否
Password	密码	否
Number of samples to aggregate	聚集的采样数目	是
Message source	设置从哪里获取消息	是
Message Type	Text、Map 或者 Object 消息	是

对于 Map 消息类型，JMeter 以文本行的形式读取源数据。每一行必须有 3 个数据域，以逗号分隔。数据域包括条目名、对象类名，如“String”（假定 java.lang 包没有指定）、对象字符串值。

对于每一行条目，JMeter 添加一个指定名称的对象。值由创建一个类的实例来得到，如果有必要，则使用 valueOf(String)方法来转换值。例如：

```
name,String,Example
size,Integer,1234
```

这是一个非常简单的实现，它并不打算支持所有可能的对象类型。

Object 消息类型目前还没有实现。

表 10-69 中有一些值可能对配置 JMS 有用。

表 10-69　对配置 JMS 有用的一些值

Apache ActiveMQ	值	注释
Context Factory	org.apache.activemq.jndi.ActiveMQInitialContextFactory	
Provider URL	vm://localhost	
Provider URL	vm:(broker:(vm://localhost)?persistent=false)	Persistence 不生效
Queue Reference	dynamicQueues/QUEUENAME	动态定义 QUEUENAME 到 JNDI
Topic Reference	dynamicTopics/TOPICNAME	动态定义 TOPICNAME 到 JNDI

15. JMS Subscriber

该采样器目前是 BETA 代码（正在改进过程中）。

JMS Subscriber 会订阅指定对象（Topic 或者 Queue）内的消息，如图 10-84 所示。对于不熟悉 JMS 的读者，简单说明一下，JMS 就是 J2EE 的消息规范。市面上有多种 JMS 服务器可供选择，其中还包括很多开源选择。

JMeter 并不包含任何 JMS 实现 jar 包；测试人员必须从 JMS 提供方获取（下载）它们，并放置在 JMeter 的 lib 目录中。

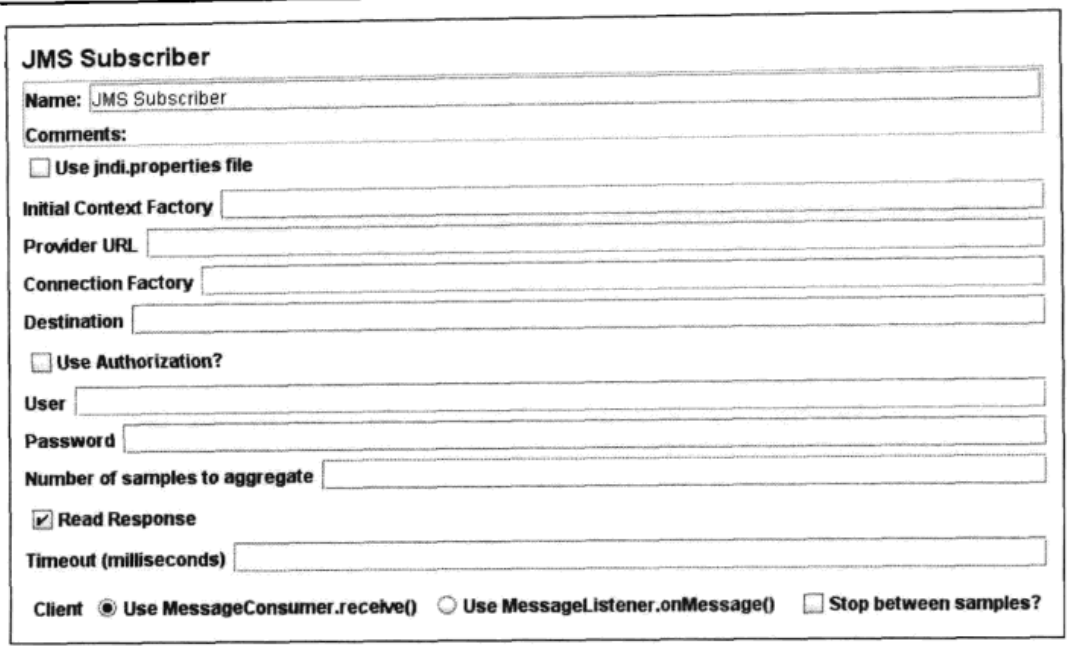


图 10-84　JMS Subscriber

参数如表 10-70 所示。

表 10-70　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Use jndi.properties file	使用 jndi.properties。请注意该文件必须位于 classpath 之上，例如，通过更新 JMeter 的 user.classpath 属性来实现。如果没有选中该复选框，JMeter 就使用"JNDI Initial Context Factory"和"Provider URL"域来创建连接	是
Initial Context Factory	Context Factory 的名称	否
Provider URL	Jms 提供者的 URL	否
Destination	消息目标（Topic 或者 Queue 名称）	是
Use Authentication	设置 JMS 提供者是否要求校验	是
User	用户名	否
Password	密码	否
Number of samples to aggregate	聚集的采样数目	是
Read Response	设置采样器是否读取采样。如果不读取采样，则仅返回响应长度	是
Timeout	指明超时时长（单位为毫秒）。0=没有。这是整体的聚集超时时长，并不针对某个采样	是

续表

属性	描述	是否必需
Client	使用哪一种客户端实现。两种实现都会创建连接，以便读取消息。不过它们使用不同的策略，如下面描述一般。 ■ MessageConsumer.receive()：为每一个请求的消息调用 receive()。保持采样器之间的连接，但是不读取消息除非采样器被激活。这最适合于队列（Queue）订阅。 ■ MessageListener.onMessage()：建立一个监听器，用于存储所有来自于某个队列的消息。 监听器会在采样器完成后保持激活状态。这最适合于话题（Topic）订阅	是
Stop between samples?	如果选中该复选框，那么 JMeter 会在每次采样结束时调用 Connection.stop()，并在每次采样前调用 start()。这在某些情况下很有用，比如有多个采样/线程连接同一个队列。如果没有选中该复选框，则 JMeter 在线程开始时调用 Connection.start()，并一直不调用 stop()，除非遇到线程结束	是

注意，JMeter 2.3.4 及其以前版本对于 MessageConsumer.receive()客户端使用不同的策略。先前的策略会启动一个后台线程，用于轮询消息。该线程在采样器完成后继续存在，因此其效果等同于 MessageListener.onMessage()策略。

16. JMS Point-to-Point

该采样器目前是 BETA 代码（正在改进过程中）。

该采样器通过点对点连接（队列）来发送和可选地接收 JMS 消息，如图 10-85 所示。它不同于发布/消费消息，并且通常用于处理事务。

当创建队列连接时，JMeter 2.3.2 以后的版本使用属性 java.naming.security.[principal|credentials]（如果存在）。如果不需要该操作，设置 JMeter 属性 JMSSampler.useSecurity.properties=false。

JMeter 并不包含任何 JMS 实现 jar 包；测试人员必须从 JMS 提供方获取（下载）它们，并放置在 JMeter 的 lib 目录中。

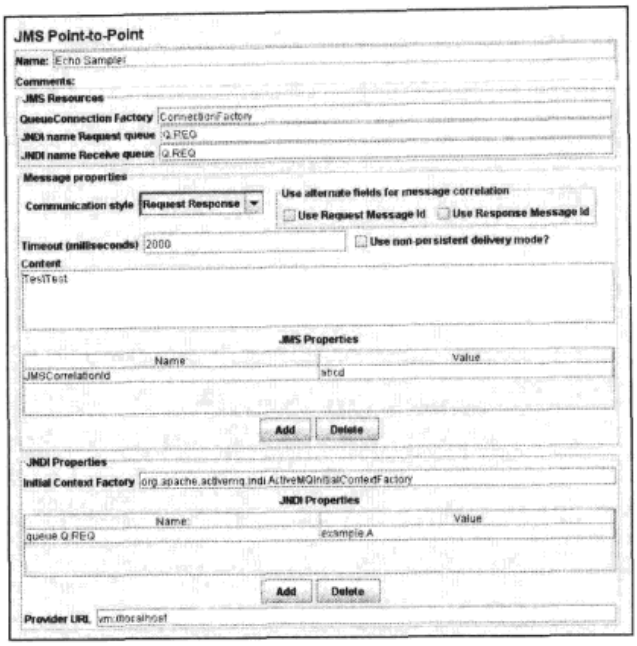


图 10-85　JMS Point-to-Point

参数如表 10-71 所示。

表 10-71　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
QueueConnection Factory	用于连接消息系统的队列连接工厂（Queue Connection Factory）的 JNDI 名	是
JNDI name Request queue	这是消息发送目标队列的 JNDI 名	是
JNDI name Reply queue	接收队列的 JNDI 名。如果这里提供了值，并且通信样式是请求响应，则这一队列会被监听，针对发送请求的响应	否
Communication style	通信样式可以是仅请求（发送请求然后忘记）或者请求响应。“仅请求”意味着只发送消息而不监听响应。因此，它可以被用于为某个系统施加压力。请求响应意味着既发送请求，又监听其收到的响应。具体行为依赖于“JNDI name Reply queue”的值。如果“JNDI name Reply queue”有值，则对应的队列被用来监听响应结果。通过请求的消息 ID 和响应的关联 ID，来匹配请求与响应。如果“JNDI name Reply queue”为空，那么请求方和服务器间的通信会用到临时队列。这与固定响应队列有很大差异。如果使用临时队列，则不同的线程将被阻塞，直到收到响应消息为止	是

续表

属性	描述	是否必需
Use alternate fields for message correlation	这些选项用于确定如何将响应消息与原始请求匹配起来。 ■ Use Request Message Id：如果选中该选项，就使用请求 JMSMessageID，否则使用请求 JMSCorrelationID。在后一种情况下，请求中必须指明 correlation id ■ Use Response Message Id：如果选中该选项，就使用响应 JMSMessageID，否则使用响应 JMSCorrelationID 这里有以下两种经常用到的 JMS 关联样式： ■ JMS 关联 ID 样式，例如，通过关联 ID，匹配请求和响应=> 两个选项都不选中，并提供关联 ID ■ JMS 消息 ID 样式，例如，将请求消息 ID 和响应关联 ID 进行匹配 => 仅选中“Use Request Message Id”复选框。 在两种情况下，都是 JMS 应用负责根据需要添加关联 ID。请注意：如果将同一个队列既用于发送消息，又用于接收消息，则响应消息与请求消息完全一致。在这种情况下，要么提供关联 ID 且两个选项都不选择；或者两个选项都选中，使用消息 ID 来做关联。这可用于测定原始的 JMS 吞吐量	是
属性	描述	是否必需
Timeout	等待响应消息的超时时长（单位为毫秒）。如果在指定时间内没有收到响应，那么特定测试案例就会失败，超时之后收到的特定响应消息将被抛弃	是
Use non-persistent delivery mode?	是否设置 DeliveryMode.NON_PERSISTENT	是
Content	消息内容	否
JMS Properties	JMS Properties 是针对潜在的消息系统而明确的属性。例如，对于 WebSphere 5.1 Web Service，测试人员需要设置 JMS 属性 targetService，以便通过 JMS 来测试 Web Service	否
Initial Context Factory	Initial Context Factory 被用于搜索 JMS 资源	否
JNDI Properties	JNDI Properties 是针对潜在 JNDI 实现的特定	否
Provider URL	Jms 提供者的 URL	否

17. JUnit Request

当前该采样器的实现支持标准 JUnit 协定及其扩展，如图 10-86 所示。它同样包含扩展形如 oneTimeSetUp 和 oneTimeTearDown。该采样器类似于 Java 采样器（JavaSampler），但有一些差异。

（1）该采样器不会使用 JMeter 的 test 接口，而是去扫描 jar 文件，以便寻找由 JUnit TestCase 类派生的类。这包括类和子类。

（2）Junit test jar 文件应该被放置在 jmeter/lib/junit 目录，而不是/lib 目录。对于 JMeter 2.3.1 以后的版本，测试人员还可以使用“user.classpath”属性来指定到哪里寻找 TestCase 类。

（3）不同于 Java 采样器（JavaSampler），JUnit 采样器不使用 name/value 对来配置。采样器假定 setUp 和 tearDown 会正确配置测试。

（4）采样器仅计算 test 方法的耗时，并不包含 setUp 和 tearDown。

（5）每次调用 test 方法，JMeter 会将结果传递给监听器。

（6）支持 oneTimeSetUp 和 oneTimeTearDown，将它们作为方法。由于 JMeter 是多线程的，我们不能像专家一样调用 oneTimeSetUp/oneTimeTearDown。

（7）该采样器将 unexpected exceptions 作为错误上报。标准 JUnit 测试运行器和 JMeter 的实现有显著差异。JMeter 不会为每次测试创建一个 Test 类的新实例，相反它为每一个采样器创建一个实例，并重复使用它。

采样器的当前实现会首先尝试使用 string 类型的构建函数来创建一个实例。如果测试类没有定义 string 类型的构建函数，那么采样器会寻找空构建函数。例如：

空构建函数：

```
public class myTestCase {
public myTestCase() {}
}
```

string 类型的构建函数：

```
public class myTestCase {
public myTestCase(String text) {
super(text);
}
}
```

默认情况下，JMeter 会提供一些成功/失败编码和消息的默认值。用户应该定义一系列独特的成功/失败编码，并在所有测试中使用它们（需保持一致）。

通用指南如下：

如果测试人员使用 setUp 和 tearDown，那么请确保将它们声明为 public。如果测试人员不这么做，那么测试可能无法正确运行。

这里有一些编写 JUnit 测试的通用指南，以便 JUnit 测试能与 JMeter 一起工作。由于 JMeter 以多线程方式运行，因此留意一些事情是很有必要的。

（1）自己编写 setUp 和 tearDown 方法，保证其能适应多线程。这通常意味着避免使用静态（Static）成员。

（2）将测试方法作为可分开的操作单元，而且不要使用很长的操作序列。用户应该将测试方法作为可分散的操作，以便后续将测试方法组合起来作为新的测试计划。

（3）避免测试方法彼此依赖。由于 JMeter 可以任意组合测试方法，因此实际运行表现不同于默认的 JUnit 表现。

（4）如果测试方法是可配置的，请谨慎对待如何存储属性。作者建议从 jar 文件读取属性。

（5）每个采样器都会创建一个测试类的实例。因此需要自己编写测试类，将初始化放在 oneTimeSetUp 和 oneTimeTearDown 中。

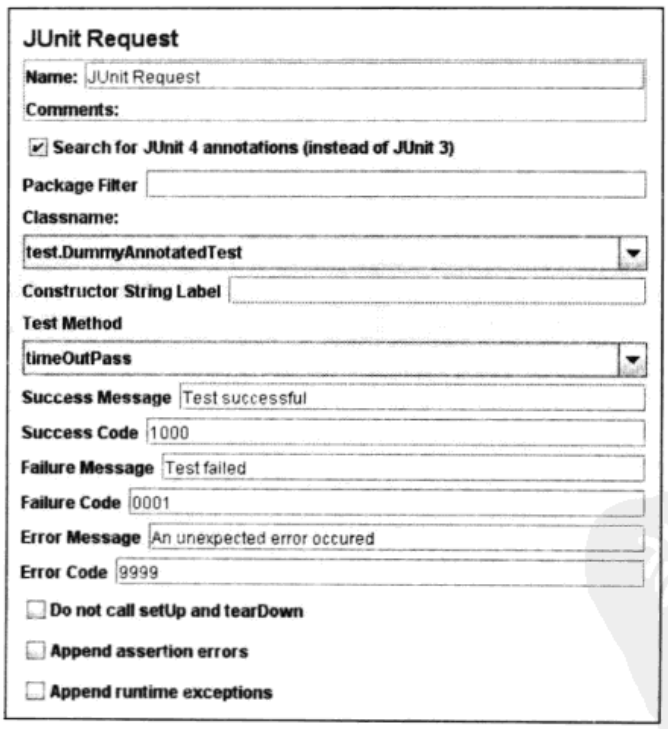


图 10-86 JUnit Request

参数如表 10-72 所示。

表 10-72　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Search for JUnit 4 annotations	选中该复选框后，就搜索 JUnit 4 测试（测试注释）	是
Package Filter	展示以逗号分隔的包列表。例如，org.apache.jmeter,junit.framework	否
Classname	JUnit 测试类的全名	是
Constructor String Label	传递给 string 类型的构建函数的字符串。如果这里提供了某个字符串，则采样器会使用 string 类型的构建函数来替代空构建函数	否
Test Method	待测试方法	是
Success Message	一个描述性的消息，用于指明成功的含义	否
Success Code	一个唯一编码，用于指明测试是成功的	否
Failure Message	一个描述性的消息，用于指明失败的含义	否
Failure Code	一个唯一编码，用于指明测试是失败的	否
Error Message	对错误的描述	否
Error Code	错误编码，并不要求唯一	否
Do not call setUp and tearDown	设置采样器，令其不调用 setUp 和 tearDown。默认情况下，setUp 和 tearDown 应该被调用。不调用这些方法，可能影响到测试，导致结果不准确。该选项应该仅在调用 oneTimeSetUp 和 oneTimeTearDown 时使用。如果选择的方法是 oneTimeSetUp 和 oneTimeTearDown，则该选项应该被选中	是
Append assertion errors	是否附加断言错误到响应消息中	是
Append runtime exceptions	是否将运行异常添加到响应消息中。只有当没有选中“Append assertion errors”复选框时，它才能生效	是

如下 JUnit4 注释将被识别：

- @Test：被用于发现测试方法和类。支持“expected”和“timeout”属性。
- @Before：被当做 JUnit3 中的 setUp()。
- @After：被当做 JUnit3 中的 tearDown()。
- @BeforeClass, @AfterClass：被当做测试方法，因此它们可以根据需要独立运行。

请注意：JMeter 当前直接运行测试方法，而不是将其交予 JUnit。这就允许在采样期间执行 setUp/tearDown 方法。

18. Mail Reader Sampler

Mail Reader Sampler 可以使用 POP3(S)或者 IMAP(S)协议来读取(和可选地删除)邮件消息，如图 10-87 所示。

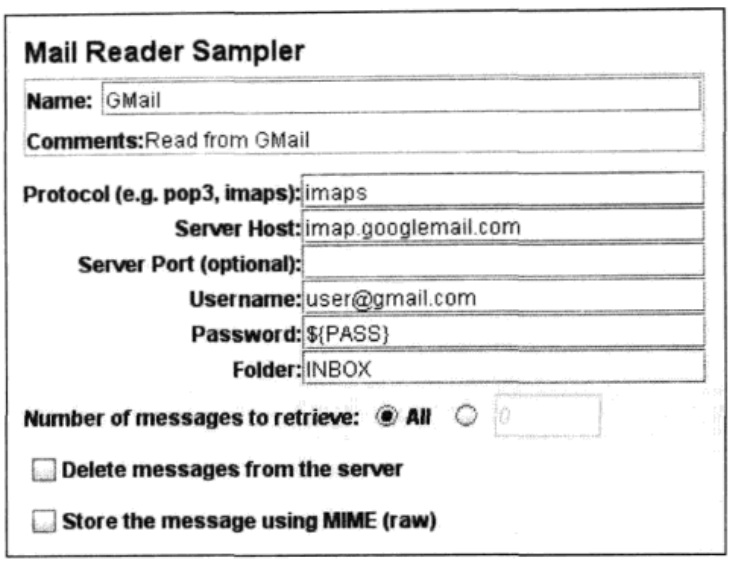


图 10-87 Mail Reader Sampler

参数如表 10-73 所示。

表 10-73 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Protocol（Server Type）	邮件服务提供者使用的协议，如 pop3、pop3s、imap、imaps。或者使用其他字符串来表征服务器协议。例如，使用 file 来表征只读邮件文件服务提供者。POP3 和 IMAP 服务提供者的实际名称为 pop3 和 imap	是
Server Host	服务器的主机名和 IP 地址。另外关于 file 协议请参考下面的内容	是
Server Port	连接服务器需要用到的端口号（可选）	否
Username	用户登录名	否
Password	用户登录密码（注：这一参数未加密地放在测试计划中）	否
Folder	使用的 IMAP(S)文件夹。另外关于 file 协议请参考下面的内容	是，如果使用 IMAP(S)

续表

属性	描述	是否必需
Number of messages to retrieve	该置该参数，用于接收全部或者某些消息	是
Delete messages from the server	如果选中该选项，消息接收到后将被删除	是
Store the message using MIME	是否将接收到的消息以 MIME 方式存储起来。如果选中了该复选框，则整个原始消息都将存储到响应数据中；头部（Headers）将不被保存下来，因为它们可以从数据中获取。如果不选中该复选框，该消息头将被保存下来作为响应头。少数头部（Date、To、From、Subject）将被保存到包体中	是

消息会被保存下来作为主采样的子采样。JMeter 2.3.4 以后的版本，多部分消息（Multipart Message）的各个部分将被作为消息的子采样。

对“file”协议的特殊处理：

file JavaMail 提供者可以被用来从文件中读取原始数据。Server 域用于指明 folder 的父文件夹的路径。单个消息文件应该被保存下来，名称形如 n.msg，其中 *n* 是消息编号。另外还有一种可选情况，server 域可以是某个文件名，文件中含有单个消息。该采样器当前的实现非常基础，主要目的是为了调试。

19. Test Action

Test Action 采样器用在条件控制器之中。该测试元件不会产生采样，而是暂停或者停止选中的目标，如图 10-88 所示。

该采样器同样可以与事务控制器一起工作，因为它可以引入暂停，而无须产生采样。如果实现可变延迟，就将暂停时间设为 0，并添加一个定时器作为子测试元件。

“Stop”操作会在当前处理的采样结束后，停止线程或者测试。“Stop Now”操作会停止测试，而无须等待采样结束；它会中断任何处于激活状态的采样。如果在 5 秒时间限制内，某些线程没有成功停止，那么一条提示消息将会在 GUI 模式下展现。测试人员可以尝试使用 Stop 命令来停止线程，如果还是没能成功停止，就只能退出 JMeter。非 GUI 模式下，如果在 5 秒时间限制内，某些线程无法成功停止，JMeter 就会自行退出（该时间限制可以通过修改 JMeter 属性——jmeterengine.threadstop.wait 来改变）。

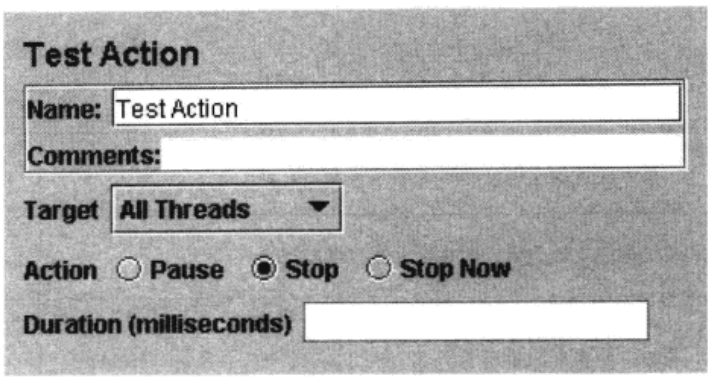


图 10-88 Mail Reader Sampler

参数如表 10-74 所示。

表 10-74 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Target	Current Thread / All Threads（忽略 Pause）	是
Action	Pause / Stop / Stop Now	是
Duration	暂停多久（单位为毫秒）	是，如果选中暂停

20. SMTP Sampler

SMTP Sampler 可以使用 SMTP/SMTPS 协议来发送邮件消息，如图 10-89 所示。它可以为连接设置安全协议（SSL 和 TLS），用户校验也是一样的。如果使用了安全协议，将会发生服务器证书核查。

这里有两种处理此类验证的核查：

- Trust all certificates，这会忽略证书链核查。
- Use a local truststore，如果选中这一选项，则使用本地 truststore 文件验证证书链。

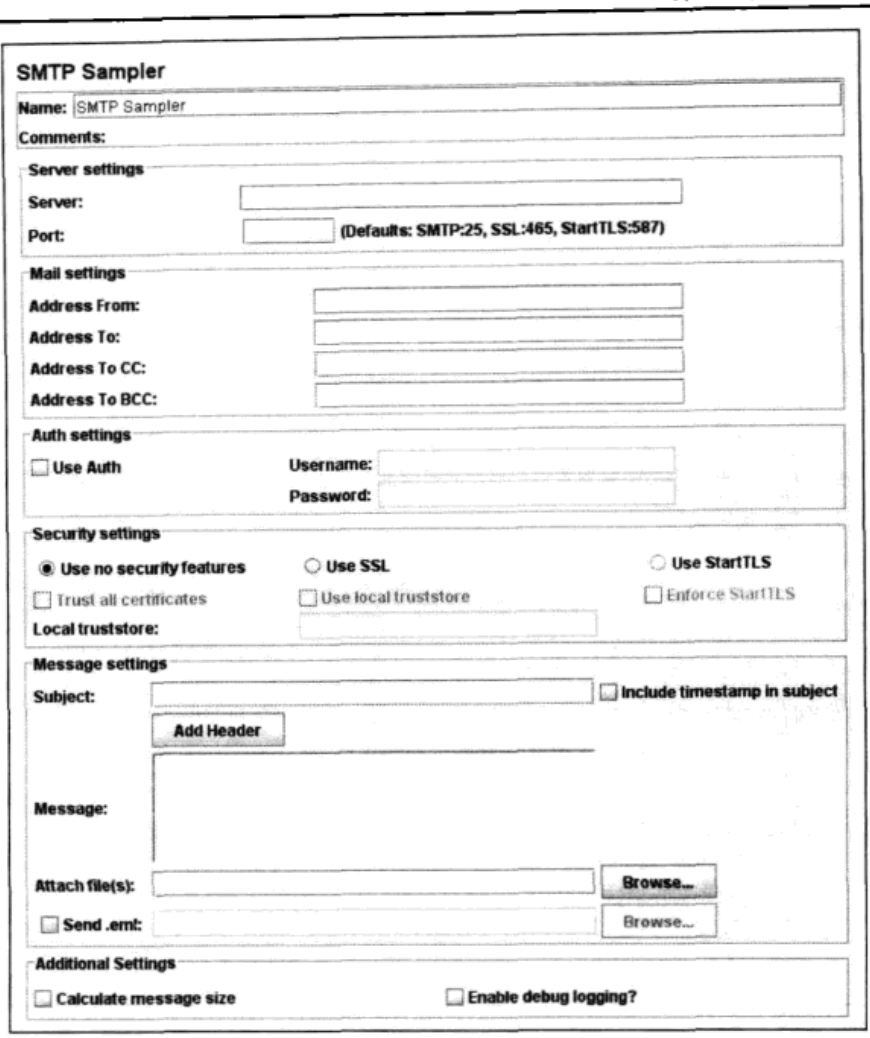


图 10-89　SMTP Sampler

参数如表 10-75 所示。

表 10-75　参数描述

属性	描述	是否必需
Server	服务器的主机名或者 IP 地址	是
Port	连接服务器用到的端口号。默认为：SMTP=25，SSL=465，StartTLS=587	否
Address From	显示在 E-mail 中的源地址	是
Address To	目标 E-mail 地址	是，除非指定了 CC 或者 BCC
Address To BCC	密件抄送目标 E-mail 地址	否

续表

属性	描述	是否必需
Address To CC	抄送目标地址	否
Use Auth	如果 SMTP 服务器要求用户校验，就选中该选项	否
Username	用户登录名	否
Password	用户密码（注：该参数未经加密存放在测试计划中）	否
Use no security features	指明到 SMTP 服务器的连接，不使用任何安全协议	否
Use SSL	指明到 SMTP 服务器的连接，必须使用 SSL 协议	否
Use StartTLS	指明到 SMTP 服务器的连接，应该尝试启用 TLS 协议	否
Enforce StartTLS	如果服务器没有启用 TLS 协议，则连接应该被终止	否
Trust all certificates	选中该复选框后，则接受独立于 CA 的所有证书	否
Use local truststore	选中该复选框后，则只接受本地信任的证书	否
Local truststore	文件路径，文件中含有信任的证书。相对路径以当前目录为基础	否
Subject	E-mail 消息主题	否
Include timestamp in subject	在主题行中包含 System.currentTimemilis()	否
Add Header	可以使用该按钮来定义额外的头部	否
Message	消息主题	否
Attach files	可以被放在消息中的文件	否
Send .eml	如果选中了该复选框，.eml 文件将被发送，用来代替主题中的条目、消息和附带的文件	否
Calculate message size	计算消息的尺寸，并存放在采样结果中	否
Enable debug logging?	如果选中了该复选框，则“mail.debug”属性将被设为“true”	否

10.8 详解 JMeter 其他测试元件

1. 测试计划（Test Plan）

用户可以在测试计划（如图 10-90 所示）中完成对测试的整体设定。

在测试计划中可以定义静态变量及其值，以便在整个测试期间重复使用，如服务器名。例如，变量 SERVER 可以被定义为 www.example.com，在测试脚本的其余部分可以使用${SERVER}来引用它。这能减少今后的脚本维护工作。

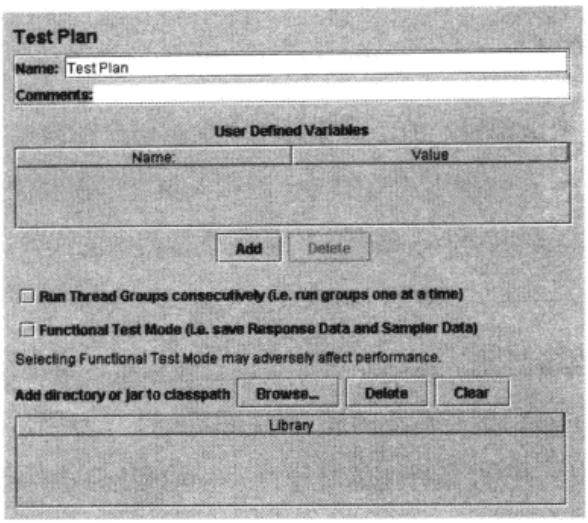


图 10-90 测试计划（Test Plan）

如果相同的变量名，在一个或者多个 User Defined Variables 配置元件中重用，该变量值以测试计划中的最后一次定义为准（从头到尾读取）。这些变量应该应用在多次运行之间会发生变化，而同一次运行内不会发生变化的情况下。

请注意：测试计划中不能引用自己定义的变量。如果测试人员需要构建其他变量而非测试计划变量，则可以使用 User Defined Variables 测试元件。

选择功能测试，就会告诉 JMeter 保存额外的采样信息（即响应数据和采样数据）到所有结果文件中。这会增加测试运行所需的资源，并且可能会增加 JMeter 的性能负担。如果只是某个特定采样需要更多数据，那么就为其添加一个子监听器，并根据需要配置关心的数据域（功能测试选项，目前不影响 CSV 结果文件，因为 CSV 结果文件不能存储这些额外的信息）。

另外，这里还有一个选项，可以告诉 JMeter 以串行方式而非并行方式，运行线程组（Thread Group）。

测试计划提供了一种简单方法为指定测试计划添加 classpath 设置。该项特性是补充性质的，意味着测试人员可以添加 jar 文件或者目录，但是移除一个条目则需重启 JMeter。请注意，不能通过这种方式来添加 JMeter GUI 补丁，因为对它们的处理早于对测试计划的处理。不过这对一些实用的 jar 文件还是有效，例如 JDBC 驱动。

JMeter 属性也提供了一个条目，用来加载额外的 classpath。在 jmeter.properties 文件中，测试人员可以编辑“user.classpath”来引入额外的库。

2. 线程组（Thread Group）

在线程组中定义了一个虚拟用户池，其中每一个虚拟用户都会针对待测服务器执行特定的测试案例，如图 10-91 所示。在线程组的 GUI 中，测试人员可以控制虚拟用户的数量（线程数），启动时间（即花费多长时间来启动所有线程），循环执行测试的次数，以及可选地启动/停止测试时间。

当使用调度器（Scheduler）时，JMeter 会一直运行线程组，除非测试达到设定的循环次数，或者超过持续时长（Duration）/运行截止时间点（End-time）（只需达到单个条件即可）。请注意线程组是否达成退出条件，仅在采样间隔时检查；当达成出口条件后，对应的线程就会退出。JMeter 不会中断正在等待服务器响应的采样器，所以等待结束的时长不固定。

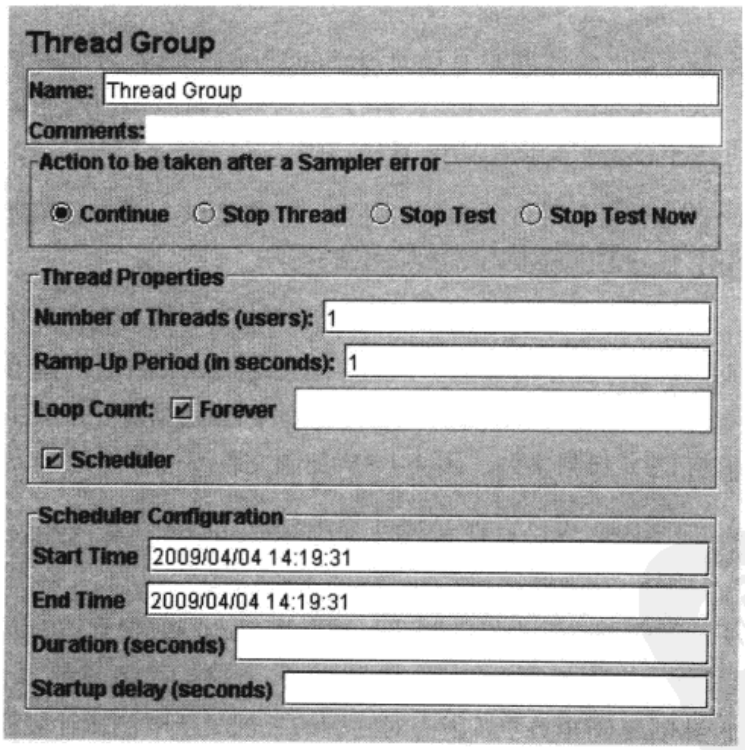


图 10-91　线程组（Thread Group）

参数如表 10-76 所示。

表 10-76　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Action to be taken after a Sampler error	用于决定采样器错误发生后做什么，包括采样器自己失败和断言失败。可选操作包括以下几项。 ■ Continue：忽略错误继续测试 ■ Stop Thread：退出当前线程 ■ Stop Test：当前采样结束后，整个测试将会停止 ■ Stop Test Now：整个测试将会突然停止。如果可以，则任何采样都将被中断	否
Number of Threads	虚拟用户数	是
Ramp-up Period	设置 JMeter 需要多长时间来启动所有线程。如果有 10 个线程，而启动时间为 100 秒，则线程启动间隔时间为 10 秒，并在 100 秒内启动这 10 个线程	是
Loop Count	执行测试的次数。另外选中“forever”会导致测试不断运行，直到手工停止	是，除非选中了 forever
Start Time	如果选中了“Scheduler”复选框，测试人员可以选择一个绝对启动时间。当测试人员启动测试后，JMeter 会等待一段时间，直到特定的启动时间到达后，才开始测试。请注意：Startup Delay 域会覆盖这里的设置，参考下面的内容	否
End Time	如果选中了“Scheduler”复选框，测试人员可以选择一个绝对停止时间。当测试人员启动测试后，JMeter 会等待一段时间，直到特定的启动时间到达后，才开始测试，并在指定的时间结束测试。请注意：Duration 域会覆盖这里的设置，参考下面的内容	否
Duration（seconds）	如果选中了“Scheduler”复选框，测试人员可以选择一个相对停止时间。JMeter 会使用这一参数来计算结束时间，并忽略 End Time 参数	否
Startup delay（seconds）	如果选中了“Scheduler”复选框，测试人员可以选择一个相对启动时间。JMeter 会使用这一参数来计算启动时间，并忽略 Start Time 参数	否

3. 工作台（WorkBench）

工作台提供了一个存放暂时不用的测试元件的地方，这是为了方便复制/粘贴，或者其他目的，如图 10-92 所示。当测试人员保存测试计划时，工作台下的内容不会随其一起保存。工作台可以单独保存，即在工作台上单击鼠标右键，在弹出的快捷菜单中选择“保存”命令。

以下特定测试元件只能放在工作台（WorkBench）中：

- HTTP Proxy Server

- HTTP Mirror Server
- Property Display

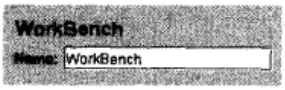


图 10-92 工作台（WorkBench）

4. SSL 管理器（SSL Manager）

通过 SSL 管理器可以选择一个客户端证书，测试人员就可以使用 Public Key Infrastructure (PKI)来测试应用系统。只有当测试人员没有初始化相应的系统属性时，才会用到 SSL 管理器。

1）选择一个客户端证书

测试人员可以使用 Java Key Store（JKS）来格式化 key store，或者使用一个 Public Key Certificate Standard #12（PKCS12）作为测试人员的客户端证书。JSSE 库有一个特性，即 key 必须有至少 6 个字符的密码。

要选择客户端证书，可以通过选择 Options→SSL Manager 命令来实现。默认情况下，测试人员可以看到一个文件选择对话框，用于寻找 PKCS12 文件。测试人员的 PKCS12 文件扩展名必须为“.p12”，以便 SSL 管理器将其识别为 PKCS12 文件。任何其他文件将被当做普通 JKS key store。如果 JSSE 已经被正确安装，则接下来测试人员将被提示输入密码。输入密码的文本框，并不会隐藏测试人员输入的密码，因此请确保此刻没有人在旁边偷窥密码。当前实现假定 key store 的密码，同样是测试人员想要验证的原始 key 的密码。

另外，测试人员还可以通过设定系统属性来选择客户端证书（参考 system.properties 文件）。

下一次运行测试，SSL 管理器会检查测试人员的 key store 是否至少包含一个 key。如果其中只有一个 key，则 SSL 管理器就会直接选择它。如果其中有多个 key，SSL 管理器当前选择第一个 key。当前没有办法选择 keystore 中的其他条目，因此期望的 key 必须处在第一位。

2）需要留意的事情

如果不使用 JDK 附带的 CA 证书，测试人员必须保证测试人员的 Certificate Authority（CA）证书已被正确安装。其中一种安装方法是，将 CA 证书导入一个 JKS 文件，并将 JKS 文件命名为“jssecacerts”。将文件放到 JRE 的 lib/security 文件夹中。对该文件的读取，会先于相同目录中的“cacerts”文件。请注意只要“jssecacerts”文件存在，安装在“cacerts”中的证书就不会被使用。这可能会为测试人员带来麻烦。如果不介意将测试人员的 CA 证书导入“cacerts”文件中，那么测试人员可以使用所有已经安装的 CA 证书来完成验证。

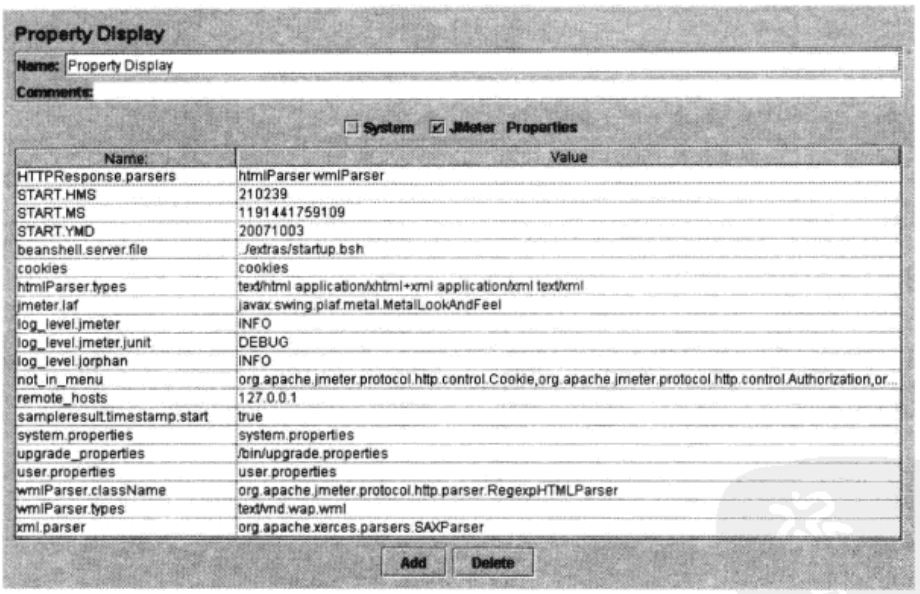
5. HTTP Mirror Server

HTTP Mirror Server 是一个非常简单的 HTTP 服务器，它简单地将接收到的数据反射回去，如图 10-93 所示，这对检查 HTTP 请求内容有用。

图 10-93　HTTP Mirror Server

6. Property Display

Property Display 展示了系统或者 JMeter 属性的值，如图 10-94 所示。测试人员可以在 Value 列中输入新值，来改变对应属性的值。该测试元件只能放在工作台下。



Name:	Value
HTTPResponse.parsers	htmlParser wmlParser
START.HMS	210239
START.MS	1191441759109
START.YMD	20071003
beanshell.server.file	../extras/startup.bsh
cookies	cookies
htmlParser.types	text/html application/xhtml+xml application/xml text/xml
jmeter.laf	javax.swing.plaf.metal.MetalLookAndFeel
log_level.jmeter	INFO
log_level.jmeter.junit	DEBUG
log_level.jorphan	INFO
not_in_menu	org.apache.jmeter.protocol.http.control.Cookie,org.apache.jmeter.protocol.http.control.Authorization,or...
remote_hosts	127.0.0.1
sampleresult.timestamp.start	true
system.properties	system.properties
upgrade_properties	/bin/upgrade.properties
user.properties	user.properties
wmlParser.className	org.apache.jmeter.protocol.http.parser.RegexpHTMLParser
wmlParser.types	text/vnd.wap.wml
xml.parser	org.apache.xerces.parsers.SAXParser

图 10-94　Property Display

参数如表 10-77 所示。

表 10-77　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否

7. Debug Sampler

通过 Debug Sampler 可以产生一个包含所有 JMeter 变量和/或者属性的采样，如图 10-95 所示。这些值可以在监听器 View Results Tree 的响应数据面板中看到。

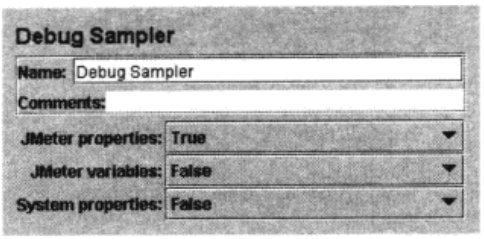


图 10-95 Debug Sampler

参数如表 10-78 所示。

表 10-78 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
JMeter properties	是否包含 JMeter 属性	是
JMeter variables	是否包含 JMeter 变量	是
System properties	是否包含系统属性	是

8. Debug PostProcessor

Debug PostProcessor 会创建一个子采样，其中含有前一采样器属性的细节。该测试元件仅供 JMeter 开发者使用。

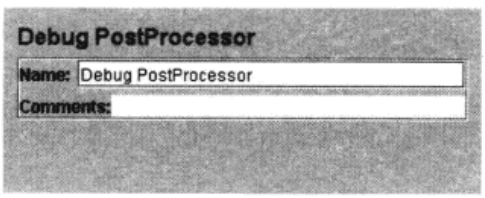


图 10-96 Debug PostProcessor

参数如表 10-79 所示。

表 10-79 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否

9. HTTP 代理服务器（HTTP Proxy Server）

通过代理服务器，JMeter 可以观察和录制测试人员使用浏览器访问待测 Web 应用系统时所做的各项操作，如图 10-97 所示。JMeter 会创建测试采样对象，并将它们直接存储到测试人员的测试计划中，这一动作是实时的（因此测试人员在录制操作时，就可以看到录制产生的采样器）。

要使用代理服务器，需要将 HTTP 代理服务器添加到工作台下。在测试树中选中工作台，单击鼠标右键以获取 Add 菜单（Add→Non-Test Elements→HTTP Proxy Server）。

另外，测试人员还需要配置浏览器，让其使用 JMeter 代理端口作为 HTTP 和 HTTPS 请求的代理。不要使用 JMeter 作为其他请求类型的代理（FTP 等），因为 JMeter 代理无法处理它们。

当录制 HTTPS 时，JMeter 代理服务器会使用虚拟证书，以便接受来自于浏览器的 SSL 连接。该证书不是浏览器通常信任的证书之一，而且不是针对正确的主机，因此浏览器将会弹出一个对话框，询问测试人员是否接受该证书。例如，服务器名“www.example.com”与证书名“JMeter Proxy”不匹配，有人可能正在尝试窃听。再如，“JMeter Proxy”证书由未知证书颁发机构“JMeter Proxy”所签署，不可能验证它是否是正确的证书。测试人员需要接受该证书，以便允许 JMeter 代理中途截取 SSL 通信，从而完成录制工作。测试人员应该仅暂时接受该证书。

如下属性可以用来改变使用的证书。

- proxy.cert.directory：在其中寻找证书的目录（默认 = JMeter bin/）。
- proxy.cert.file：key store 文件的名称（默认为“proxyserver.jks”）。
- proxy.cert.keystorepass：key store 密码（默认为“password”）。
- proxy.cert.keypassword：证书 key 密码（默认为“password”）。
- proxy.cert.type：证书类型（默认为“JKS”）。
- proxy.cert.factory：工厂（默认为“SunX509”）。
- proxy.ssl.protocol：使用的协议（默认为“SSLv3”）。

如果测试人员的浏览器当前正在使用代理（如公司内网中使用代理），那么测试人员需要在 JMeter 启动前告诉 JMeter 使用这个代理，即使用命令行选项-H 和-P。在运行录制产生的测试计划时，同样需要此项设置。

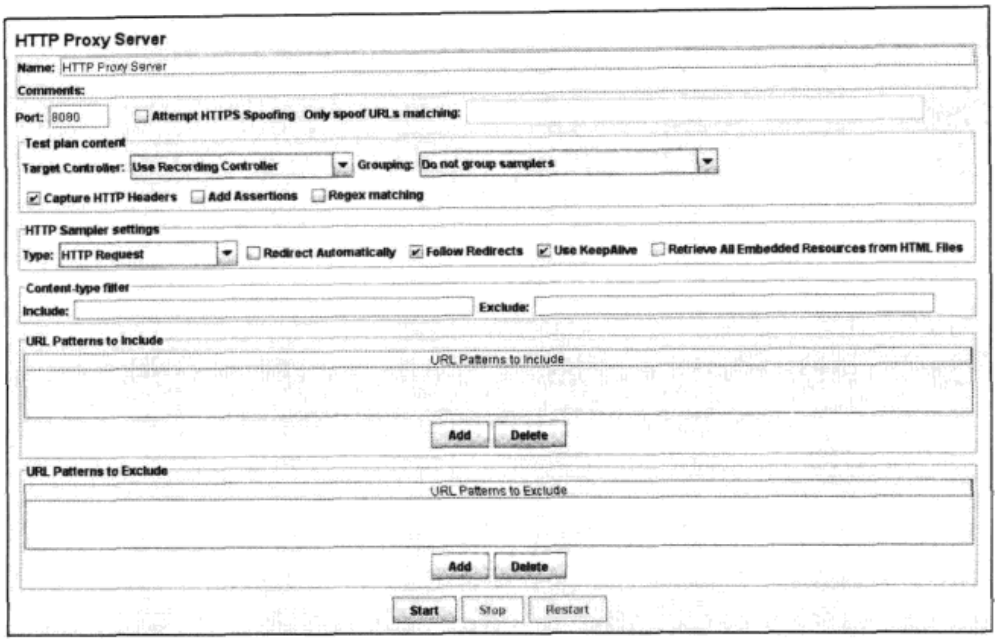


图 10-97　HTTP 代理服务器（HTTP Proxy Server）

参数如表 10-80 所示。

表 10-80　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Port	代理服务器监听的端口。8080 是默认端口，但是测试人员可以改变它的值	是
Attempt HTTPS Spoofing	请注意 HTTPS 欺骗应该不再需要。当测试人员启用 HTTPS 欺骗后，将会发生如下事情： ■ 来自于客户端的所有匹配（参考下面的内容）HTTP 请求将被转为 HTTPS（在代理和 Web 服务器之间） ■ 将检查所有文本响应数据，并将出现的字符串 https://替换为 http://；默认 HTTPS 端口（443）同样将被移除（如果存在的话） 因此如果测试人员想要使用这一特性，那么当测试人员在客户端中浏览时，在浏览器中应该输入“http://...”来代替“https://...”。JMeter 会请求和录制所有匹配 HTTPS 的东西，无论是否应该这么做	是
Only Spoof URLs matching	如果指定了该参数，那么参数值必须是正则表达式（java.util.regex），用于匹配将被欺骗的 HTTP URL(s)。例如 VK，如果测试人员想欺骗 http://a.b.c/service/，而非 http://a.b.c/images，那么测试人员可以使用表达式 http://a.b.c/service/.*。请注意表达式以“.*”结尾，因为它必须匹配整个 URL	否

续表

属性	描述	是否必需
Target Controller	用于存放代理录制产生的采样器的控制元件。默认情况下，它会寻找一个 Recording Controller，并将它们存放其下	是
Grouping	是否通过单次单击，为请求组合采样器（接收的请求没有显著的时间间隔），而如何在录制过程中标识组合方式： ■ Do not group samplers：顺序存储所有录制产生的采样器，不做任何组合 ■ Add separators between groups：添加一个控制器名为“----------”以便在组合间添加一个可见的分隔。另外采样器被顺序存储 ■ Put each group in a new controller：为每一个组合创建一个简单控制器，并将对应组合的所有采样器放在其下 ■ Store 1st sampler of each group only：只有每个组合的第一个请求将被录制。这些采样器中的“Follow Redirects”和“Retrieve All Embedded Resources...”标志将被打开 ■ Put each group in a new transaction controller：为每个组合创建一个新的事务控制器，并将对应组合的所有采样器放在其下 proxy.pause 属性定义了 JMeter 将两个请求认定为不同“单击”所需的最小时间间隔。默认为 1000（毫秒）即 1 秒。如果测试人员使用组合，请确保测试人员在两次单击操作间有足够间隔时间	是
Capture HTTP Headers	头部是否应该加入到测试计划中？如果选中了该复选框，则会为每个 HTTP 采样器添加一个信息头管理器（Header Manager）。代理服务器会从产生的信息头管理器中，移除 Cookie 和授权头部。默认情况下，它还会移除 If-Modified-Since 和 If-None-Match 头部。通常当录制需要下载所有内容的操作时，这些都是用来判断浏览器缓存中的项目是否是最新的。要设置哪些额外的头部将被移除，可以定义 JMeter 属性 proxy.headers.remove，其值为逗号分隔的头部列表	是
Add Assertions	为每一个采样器添加一个空白断言	是
Regex matching	当替换变量时，使用正则匹配	是
Type	产生哪种类型的采样器（Java default 或者 HTTPClient）	是
Redirect Automatically	在产生的采样器中设置 Redirect Automatically	是
Follow Redirects	在产生的采样器中使用 Follow Redirects	是
Use KeepAlive	在产生的采样器中使用 KeepAlive	是

续表

属性	描述	是否必需
Retrieve all Embedded Resources from HTML Files	在产生的采样器中设置 Retrieve all Embedded Resources	是
Content Type filter	依据 content-type（例如“text/html [;charset=utf-8]”）来过滤请求。该域使用正则表达式，以便检查 content-type 中是否包含它们（无须匹配整个域）。首先检查 Include 过滤器，接着检查 Exclude 过滤器。被过滤排除的采样，不会被存储	否
Patterns to Include	正则表达式，用于匹配采样的完整 URL。通过它可以过滤正在录制的请求。所有经过的请求中，只有符合 Include/Exclude 域要求的才会被录制。如果 Include 和 Exclude 域都为空，则所有请求都会被录制（这就意味着会录制大量请求，包括页面、图片、stylesheets 等）。如果 Include 域中存在至少一条条目，则只有符合一条或者多条 Include Patterns 的请求才会被录制	否
Patterns to Exclude	正则表达式，用于匹配采样的 URL。任何匹配一条或者多条 Exclude Patterns 的请求，将不会被录制	否
Start	启动代理服务器。一旦代理服务器启动并且可以录制请求后，JMeter 会将如下信息写到控制台中：“Proxy up and running!”	无意义
Stop	停止代理服务器	无意义
Restart	停止并重启代理服务器。当测试人员希望 change/add/delete 一个 Include/Exclude 过滤器时，这会有用	无意义

Include 和 Exclude Patterns 被作为正则表达式对待（使用 Jakarta ORO）。它们会匹配每一个浏览器请求的主机名、端口号（实际或者隐藏的）路径和询问（如果存在）。如果测试人员正在浏览的 URL 是：“http://jakarta.apache.org/jmeter/index.html?username=xxxx”，那么正则表达式的匹配对象是：“jakarta.apache.org:80/jmeter/index.html?username=xxxx”。

如果测试人员想要包含所有.html 文件，则使用的正则表达式应该为：“.*\.html(\?.*)?”或者“.*\.html”，即测试人员知道这里没有询问字符串或者只关心没有询问字符串的 html 页面。

如果这里存在任何 Include Patterns，那么 URL 必须匹配至少一个 Patterns，否则它就不会被录制。如果这里存在任何 Exclude Patterns，那么 URL 不能匹配任何一个 Patterns，否则它就不会被录制。组合使用 Include 和 Exclude，测试人员可以只录制自己所关心的请求，而排除掉其他请求。

被正则表达式所匹配的字符串，必须与完整的 host+path 字符串相同。因此，“\.html”不会匹配 j.a.o/index.html。

从 JMeter 2.3.2 版本开始，可以捕获 binary POST 数据。要配置哪些 content-types 会被当做 binary，可以更新 JMeter 属性 proxy.binary.types。默认设置如下：

```
# These content-types will be handled by saving the request in a file:
proxy.binary.types=application/x-amf,application/x-java-serialized-objec
t
# The files will be saved in this directory:
proxy.binary.directory=user.dir
# The files will be created with this file filesuffix:
proxy.binary.filesuffix=.binary
```

测试人员可以让代理在录制得到的脚本中添加定时器。要做到这一点，测试人员可以直接在 HTTP 代理服务器元件内创建一个定时器。代理会为每个它录制的采样添加一个该定时器的副本，或者为每个组合的第一个采样添加一个定时器副本（如果使用分组）。JMeter 将在定时器副本的属性中扫描变量${T}是否存在，接下来它会使用距上一次采样录制的时间间隔（单位为毫秒）来代替找到的变量${T}。

当测试人员做好录制准备后，单击“Start”按钮。

小贴士

测试人员需要编辑浏览器的代理设置，以便指向合适的服务器和端口（服务器即运行 JMeter 的机器，而端口号来自于图 10-97 中的代理服务器控制面板）。

1）录制得到的采样器放在哪里

JMeter 将录制得到的采样器放在测试人员所选定的目标控制器（Target Controller）中。如果测试人员选择默认选项“Use Recording Controller”，那么它们就会被存储到测试对象树中找到的第一个录制控制器（Recording Controller）下（因此请确保在开始录制前，已经添加了一个录制控制器）。

如果代理看上去没有录制任何采样，这可能是由于浏览器并没有实际使用该代理。要检查是否是这种原因所导致，可以尝试停止代理。如果浏览器还能继续下载页面，就证明它没有通过代理来发送请求。请仔细检查浏览器选项设置。如果测试人员所录制的服务器与 JMeter 运行在同一台机器上，那么请确保浏览器没有设置“Bypass proxy server for local addresses”（这一说法是针对 IE 7 的，但是其他浏览器也有类似选项）。如果 JMeter 不能录制 http://localhost/或者 http://127.0.0.1/此类浏览器 URL，则可以尝试使用非回送主机名或者 IP 地址，例如 http://myhost/或者 http://192.168.0.2/。

2）处理 HTTP 请求默认值（HTTP Request Defaults）配置元件

如果 HTTP 代理服务器在存储采样的控制器中直接找到了 HTTP 请求默认值，或者直接在它的父控制器中找到了 HTTP 请求默认值，那么录制得到的采样器中已指定默认值的域就为空。

测试人员可以进一步控制这一行为，即将 HTTP 请求默认值直接放到 HTTP 代理服务器下，那么该配置元件中的非空值就会覆盖其他 HTTP 请求默认值中的值。

3）替换用户自定义变量

同样地，如果 HTTP 代理服务器在存储采样的控制器中直接找到了 User Defined Variables（UDV），或者直接在它的父控制器中找到了 User Defined Variables（UDV），那么在录制得到的采样中，任何与变量值相同的参数都会被对应的变量所替换。另外，测试人员可以将 User Defined Variables（UDV）直接放到 HTTP 代理服务器下，以便覆盖将被替换的值。

请注意这种匹配是大小写敏感的。

用变量来替换：默认情况下，代理服务器会搜寻所有出现的 UDV 值。如果测试人员定义变量“WEB”的值为“www”，那么找到的字符串“www”将被${WEB}所替换。为了避免在所有地方都这么做，请选中“Regex matching”复选框。这会告诉代理服务器将值作为正则表达式对待（使用 ORO）。

如果测试人员仅期望匹配某个整字符串，那么就使用^$封装它，例如^thus$。

如果测试人员仅期望在字符串的开头匹配/images，就使用值“^/images”。 Jakarta ORO 还支持零宽度前瞻（zero-width look-ahead），因此测试人员可以通过使用“^/images(?=/)”来匹配“/images/...”，却将尾巴上的/留在输出中。请注意当前 Jakara ORO 版本不支持背后查找（look-behind），例如“(?<=...)或者(?<!...)”。

如果将变量解释为模式（Patterns）时存在问题，那么就会被记录到 jmeter.log 中，因此当 UDVs 工作不正常时，请检查这一文件。

当测试人员完成录制测试采样后，就停止代理服务器（单击“Stop”按钮）。请记住重置测试人员的浏览器代理设置。现在测试人员可能希望排序和重新排序测试脚本，添加定时器、监听器、一个 Cookie 管理器等。

4）怎样才能录制服务器的响应

将查看结果树（View Results Tree）监听器作为代理服务器的子元件，如此则会在监听器中展示服务器响应。另外，测试人员还可以添加一个保存响应到结果（Save Responses to a file）监听器，它能将响应保存到文件中去。

5）Cookie 管理器

如果测试人员使用 Cookies 来测试服务器，那么当测试人员完成测试计划录制后，别忘了为其添加 HTTP Cookie 管理器（HTTP Cookie Manager）。在录制期间，浏览器会处理所有 Cookies，但是 JMeter 在测试运行期间需要一个 Cookie 管理器来管理完成 Cookie 处理。JMeter

会传递录制期间浏览器发送的所有 Cookies，但并不将它们存储在测试计划中，因为它们在两次运行间可能会发生改变。

6）授权管理器

JMeter 会传递浏览器发送的所有授权头，但并不将它们存储在测试计划中。如果站点要求授权，测试人员就需要添加一个授权管理器，并为其填充一些必要的条目。

7）上载文件

一些浏览器（如 Firefox 和 Opera）在上传文件时，并不包含文件的全名。这可以导致 JMeter 代理服务器失败。 其中一种解决办法就是确保所有待上传文件都被放置在 JMeter 工作目录中，或者在包含待上传文件的目录下启动 JMeter。

10.9　本章小结

本章详细介绍了 JMeter 的各类测试元件，包括监听器、逻辑控制器、配置元件、定时器、前置处理器、后置处理器、采样器等。了解各类测试元件的特性和适用范围，对于创建 JMeter 测试脚本很有帮助。本章内容较多，读者朋友并不需要掌握所有测试元件，只需了解与实际工作相关的测试元件即可。

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。

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. 使用函数可以做些什么

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。

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。

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。

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）。

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。

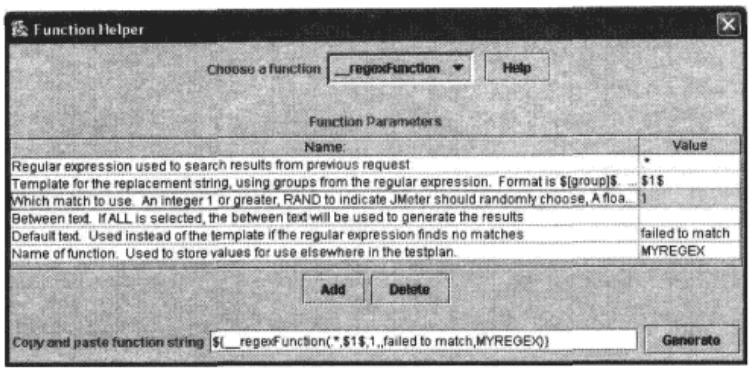


图 11-1　函数助手（Function Helper）对话框

使用函数助手，测试人员可以从下拉列表中选择一个函数，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，<a href="(.*)">。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，<input type="hidden" name="(.*)" value="(.*)">。在这个例子中，链接的 name 作为第一个匹配组合，链接的 value 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：$[group_number]$。例如1或者 2。测试人员的模板可以是任何字符串	是
第 3 个参数	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： ■ 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 ■ RAND，告诉 JMeter 随机选择一个匹配项 ■ ALL，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，默认值为 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看来，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。

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

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 ()
```

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： ■ new Date()：返回当前日期和时间 ■ Math.floor(Math.random()*(${maxRandom}+1))：在 0 和变量 maxRandom 之间的随机数 ■ ${minRandom}+Math.floor(Math.random()*(${maxRandom}-${minRandom}+1))：在变量 minRandom 和 maxRandom 之间的随机数 ■ "${VAR}"=="abcd"	是
变量名	重用函数计算值的引用名	否

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）。

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）。

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。

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 -Jhostname1=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。

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。

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。

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)} = ï¿½ (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、perlreref）。

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'。

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。

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。

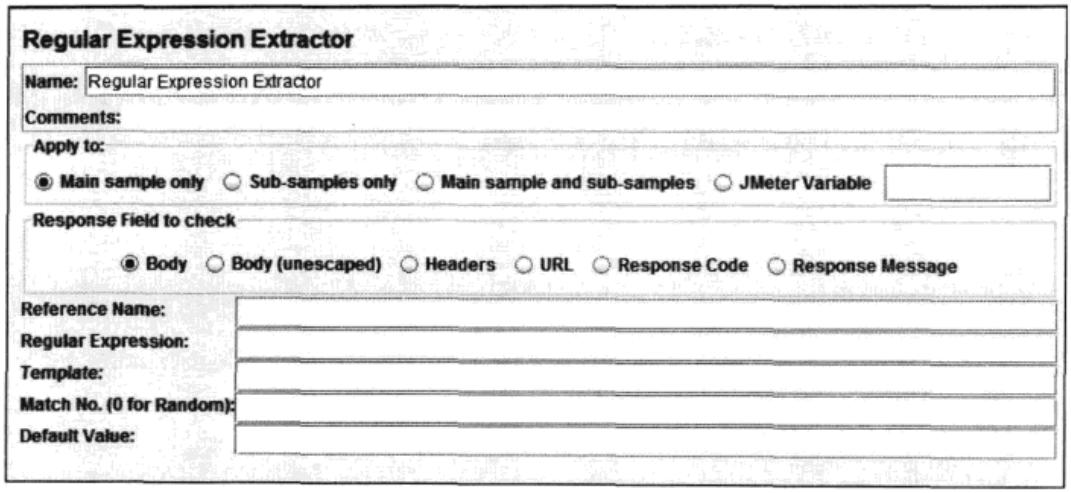


图 11-2　JMeter 正则表达式提取器

例如，

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

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。

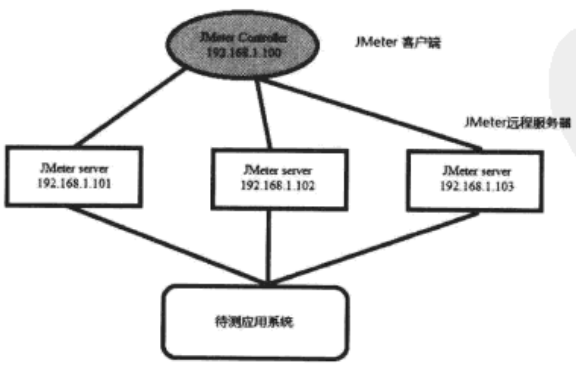


图 11-3 JMeter 远程测试原理图

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。

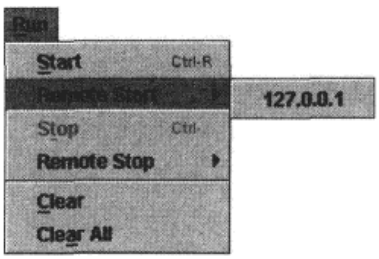


图 11-4　运行菜单

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 claspath 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。这样就会对网络和 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。

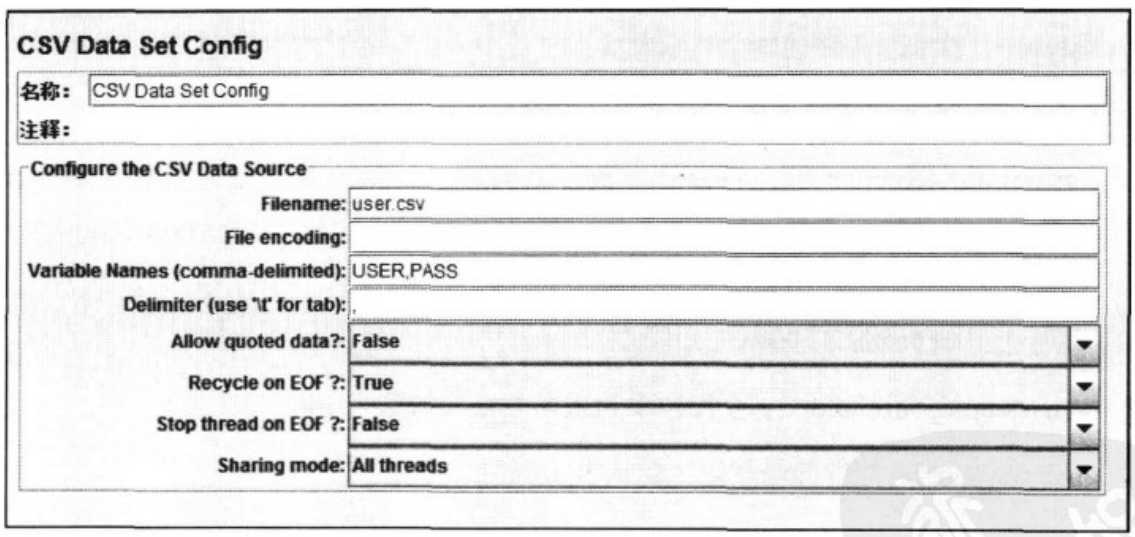


图 11-5 CSV DataSet 配置元件

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，它表现得的一台服务器，可以通过 Telnet 或者 HTTP 访问。

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。

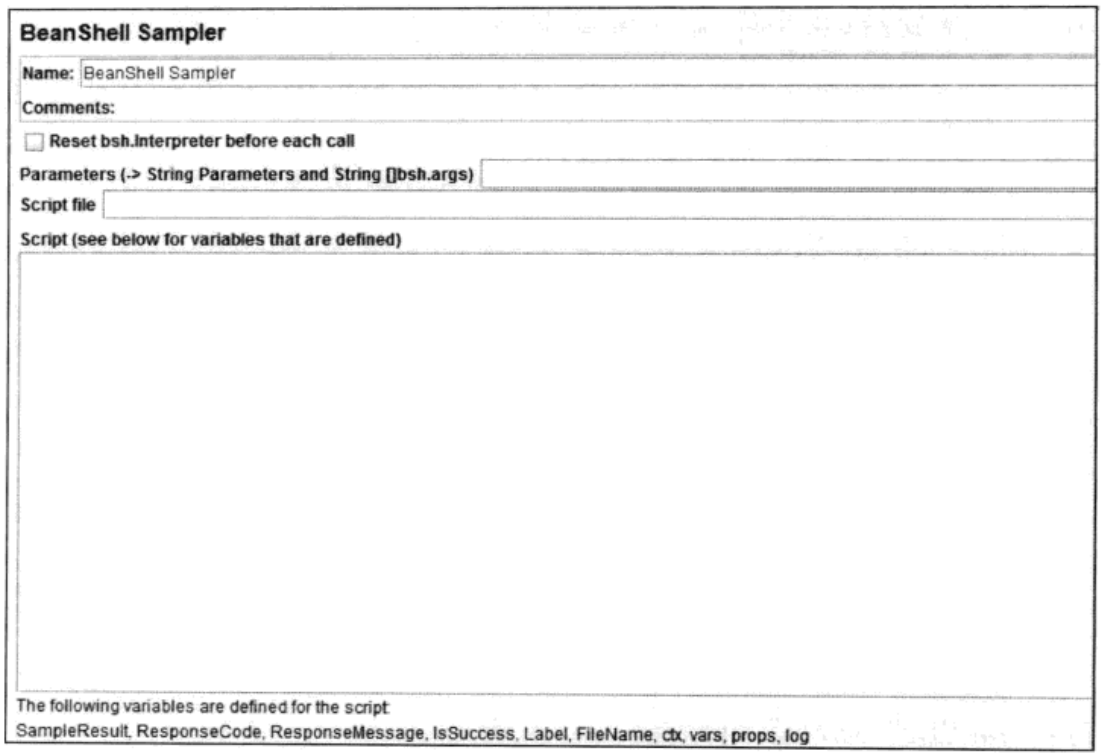


图 11-6　BeanShell 采样器

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.Interperter file.bsh
[parameters]
```

或者：

```
$ java -cp bsh-xxx.jar bsh.Interperter
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去。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}”。

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。

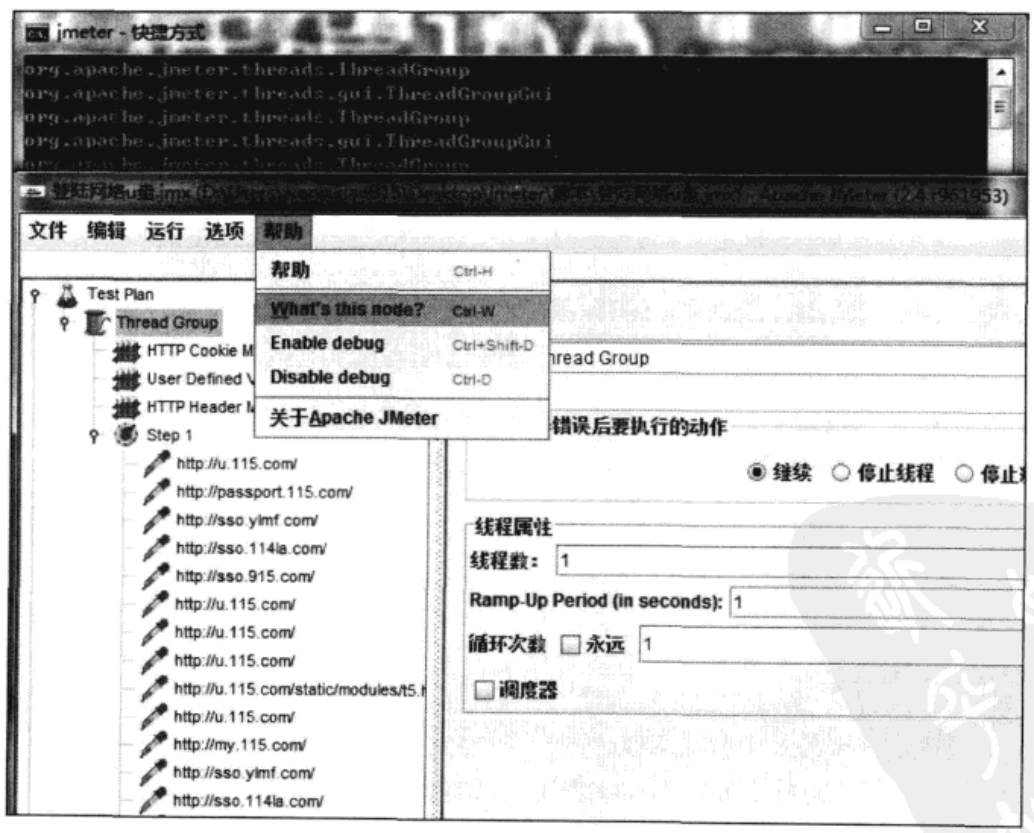


图 11-7 BeanShell 采样器

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；“详解 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。

第 12 章

性能测试结果分析

在 JMeter 测试脚本执行完成后，让很多初学者感到畏惧和害怕的阶段到来了——性能测试结果分析。因此可能很多读者朋友都认为本章是最重要和最困难的一章。其实不然，前期的性能测试需求分析、性能测试案例设计，以及 JMeter 测试计划的设计都很重要。正所谓“不积跬步，无以致千里；不集小流，无以成江海”，没有扎实的前期工作，性能测试结果分析就是“神马浮云”。

实际上，性能测试从头到尾都是一项谨慎细致的工作。工作的各个环节紧密关联，牵一发而动全身，因此要做好性能测试结果分析，首要的任务是做好前期的各项工作。

12.1 如何分析性能测试结果

在完成 JMeter 测试脚本执行后，首先要做的就是判断收集到的测试数据是否真实有效。实际性能测试中有很多情况会导致测试数据失效，例如，运行 JMeter 的机器性能存在瓶颈、网络拥塞，甚至于测试脚本本身设计存在问题，等等。对无效的测试数据进行分析，纯粹是浪费时间。那么该如何判断测试数据是否有效呢？作者推荐按照如下步骤进行分析：

（1）分析在整个性能测试执行期间，测试环境是否稳定正常。如果测试环境在性能测试执行过程中出现过异常，那么测试数据就会受到“污染”，由此得到的分析结果也变得毫无价值。

例如，测试期间运行 JMeter 的机器 CPU 占用率经常达到 100%（或者内存占用很高）、测试网络出现拥塞导致响应延迟、待测系统参数配置错误（JDBC 连接池等）……

（2）检查 JMeter 测试脚本参数设置是否合理，检查 JMeter 运行模式是否合理。JMeter 测试脚本参数的设置非常重要，它会直接影响测试数据的准确性。例如，有初学者经常将线程组

的参数 Ramp-Up Period(in seconds)设置为 0 或者 1。如此一来，JMeter 就会瞬间启动该线程组下的全部虚拟用户，会为待测服务器施加巨大的压力。轻则导致服务器响应时长超长，重则导致部分虚拟用户等待响应超时而报错。正确的做法应该是逐步加压，而不是瞬间达到目标压力。另外初学者还容易犯的一个错误，就是以 GUI 模式运行大量并发测试。相对于非 GUI 模式而言，GUI 模式对机器 CPU 和内存的占用会高得多。对于大并发量测试，最好采用非 GUI 模式。

（3）检查测试结果是否暴露出了系统瓶颈。没有必要在一切正常的测试结果上纠缠不止，应该重点关注异常的测试结果。如果测试结果显示一切正常，那么首先需要考虑的是并发用户数是否足够多，对待测服务器施加的压力是否足够大。另外还需要考虑，待测系统是否存在什么机制屏蔽掉了大部分压力，例如，禁止同一个用户多次执行某项操作，或者在多次查询同一组数据时使用缓存技术。对于这类情况，需要理清实际情况多加分析，以免漏掉本该发现的性能测试缺陷。

确定测试结果有效后，接下来就需要对测试数据进行深入分析了。面对庞杂的原始数据，该如何进行分析呢？作者认为这需要遵循一定的原则，简而言之，就是“由表及里，由内而外，抽丝剥茧”，如图 12-1 所示。

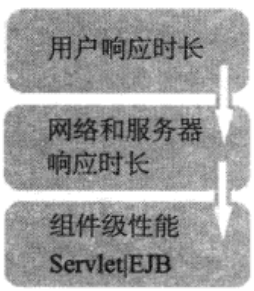


图 12-1　性能测试数据分析原则示意图

对一个软件应用系统而言，最终用户不会去关心系统的内部架构、技术实现等内部细节，他们只关心系统是否正确，是否响应及时。而且当系统出现性能下降时，最直观的表现就是响应时长变长。因此应该首先从原始测试数据中查看系统响应时长，判断它是否满足用户的期望，以此作为性能分析的起点。如果系统响应时长不满足用户期望，则说明系统的性能出了问题。发现系统存在性能问题后，就需要进一步分析哪个环节出了问题。

目前的企业级 B/S 架构应用系统的架构颇为复杂，甚至存在不同企业的架构体系大不一样的情况。不过万变不离其宗，不同 B/S 架构应用系统的基本构成还是相似的。

如图 12-2 所示为 B/S 架构应用系统的基本构成。从图中可以看出，用户从客户端发出的请求数据包经网络到达应用服务器，在由应用服务器处理后转交数据库处理，最终响应结果由原路返回。在整个处理过程中，可以把响应时长分为两段：一段是 T_s，即服务器响应时长，包括

应用服务器和数据库服务器的响应时长；另一段是 T_n，即网络响应时长。

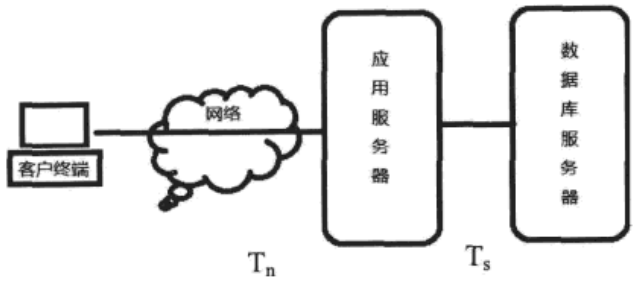


图 12-2　B/S 架构应用系统响应时长分解图

JMeter 是开源性能测试工具，自然也无法如 LoadRunner 一般设计得非常完备，JMeter 工具本身无法区分 T_s 和 T_n，它只能统计总响应时长。不过测试人员可以用其他办法来加以区分，例如，使用最简单的 ping 获取 T_n，再用总响应时长减去 T_n 即可得到 T_s。此刻，我们已经将响应时长分成了两部分，当然也就能将问题细分成两种网络瓶颈问题和服务器瓶颈问题。

接下来还需要对服务器瓶颈问题进一步细分为：应用服务器瓶颈问题和数据库服务器响应问题。要进一步分析问题根源，需要动态监视应用服务器和数据库服务器。关于应用服务器的监控，本书前面的内容有所涉及，请参考第 9 章“服务器监控测试脚本开发”。而对于数据库服务器的监控，有多种开源和商业监控工具可供选择，作者比较熟悉的一种工具是赛门铁克 Veritas i3 APM。该工具可以查找指定时间段最耗服务器资源的 TopSQL，这对于定位数据库性能缺陷非常有帮助，如图 12-3 所示。

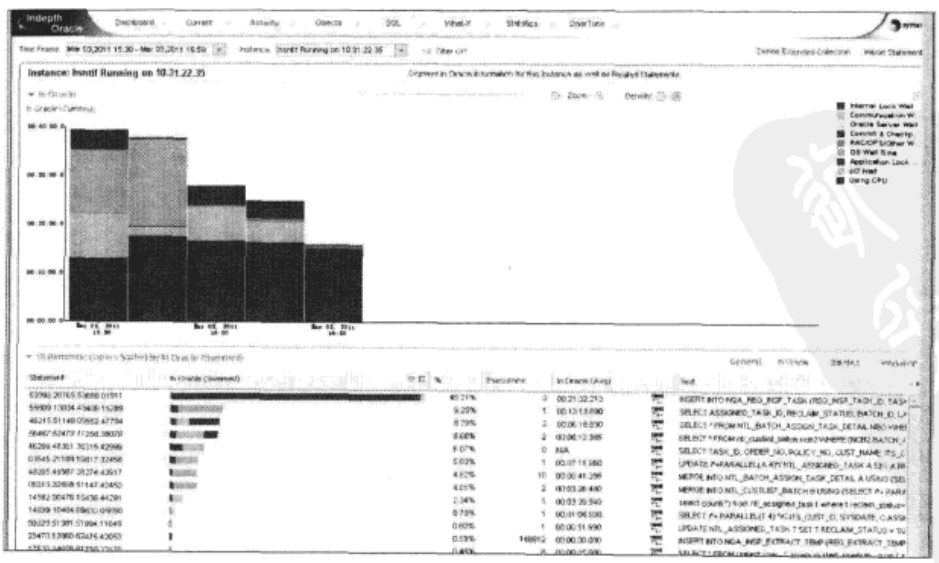


图 12-3　数据库监控工具——赛门铁克 Veritas i3 APM

另外，应用服务器日志对于定位性能测试瓶颈也很有用，对于各种报错信息测试人员都不应轻易放过。

12.2　如何借助监听器发现性能缺陷

12.2.1　监听器——性能测试分析的基石

性能测试分析离不开合适的监听器，精心挑选的监听器可以帮助测试人员识别性能缺陷，甚至还能有助于定位缺陷。JMeter 目前支持的监听器包括以下几种：

1. 图形结果（Graph Results）

图形结果监听器将收集的所有 JMeter 采样信息绘制在一个简单图形之中。在图形的底部，标明了各项统计信息，即当前采样响应时长（黑色）、当前平均采样响应时长（蓝色）、当前采样响应时长标准差（红色）、当前吞吐率（绿色），单位为毫秒。吞吐率表征服务器每分钟处理的实际采样数。该值计算时会包含测试人员在测试脚本中加入的任何延迟，以及 JMeter 工具自身的处理时间。之所以要统计该值，是因为它表征了每分钟服务器实际处理了多少请求。测试人员可以通过增加并发线程数或减少脚本中的延迟，来找到系统支持的最大吞吐率。

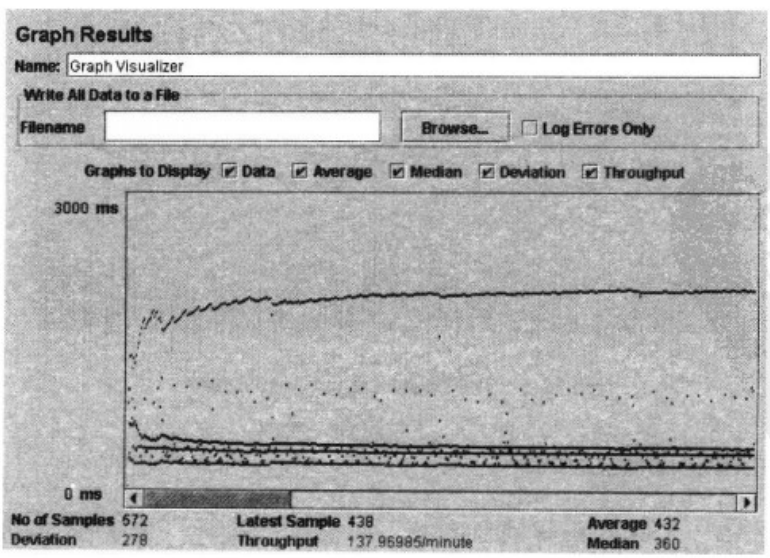


图 12-4　图形结果（Graph Results）

下面简要介绍了图形中的元素。

- Data：绘制实际的数据值。

- Average：绘制平均值。
- Median：绘制中间值。
- Deviation：绘制标准差（用于衡量数据变化）。
- Throughput：绘制单位时间的采样数。

底部的独立图片用于展示当前值。“Latest Sample”显示当前采样响应时长，在上面的图中作为“Data”绘制。

2. 样条视图（Spline Visualizer）

样条视图会提供从测试开始到结尾的所有采样时间的视图（如图 12-5 所示），它并不会关心已经发生多少次采样。样条有 10 个点，每一个点表征 10%的采样，并使用样条逻辑加以连接，形成一根连续的曲线。

图形会自动调整以便适应窗口。这一点在比较图形时，应该加以考虑。

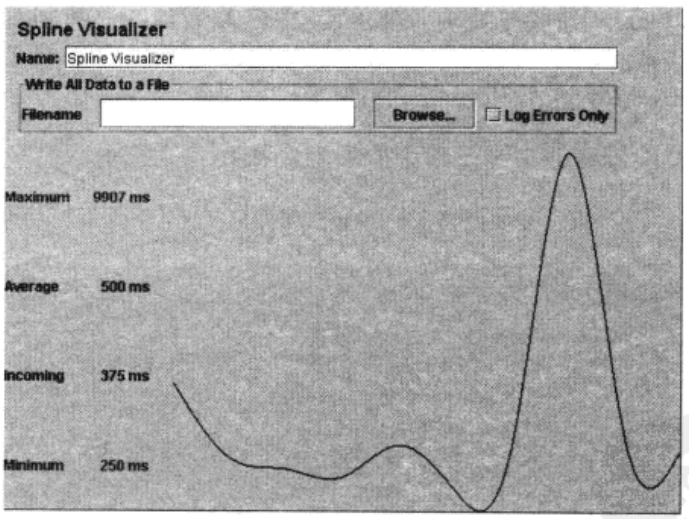


图 12-5 样条视图（Spline Visualizer）

3. 断言结果（Assertion Results）

断言结果视图会展示每个采样携带的标签，如图 12-6 所示。它同样会报告测试计划中断言产生的失败。

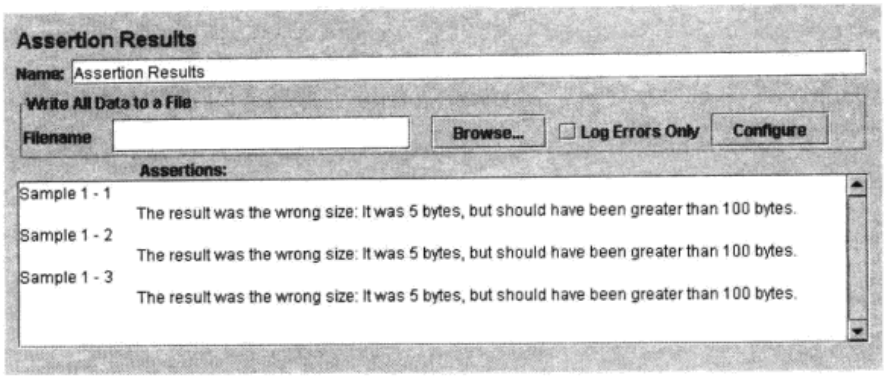


图 12-6　断言结果（Assertion Results）

4．查看结果树（View Results Tree）

查看结果树会以树的方式来展示所有采样响应结果，测试人员可以通过它来查看任何采样的响应。除了展示响应结果之外，测试人员还能看到获取响应所耗费的时间，以及一些响应代码。需要注意的是，Request 面板中只显示 JMeter 添加的头部。它不会展示由 HTTP 协议实现添加的头部（如 Host）。

查看响应有多种可选方式（HTML、HTML（Download Embedded Resources）、JSON、Regexp Tester、Text、XML），用户可以在左侧面板底部的下拉列表框中选择。

用户还可以创建附加选项。对应的类必须实现接口：org.apache.jmeter.visualizers.ResultRenderer，并且/或者继承抽象类：
org.apache.jmeter.visualizers.SamplerResultTab，JMeter 必须要能访问该类编译后得到的文件（如将其放入 lib/ext 目录）。

默认的“Text”视图会展示响应中包含的全部文字内容。需要注意的是，该选项只有在响应的内容类型是文字形式时才生效。如果内容类型以如下形式开头，那么它就被认为是二进制形式，否则被认为是文字形式。

```
image/
audio/
video/
```

如果没有指定内容类型（Content-Type），那么对应的内容就不会在任何 Response Data 面板中展示。在这种情况下，测试人员可以使用“Save Responses to a file”来保存数据。请注意在采样结果中仍然可以找到响应数据，因此也可以使用后置处理器来访问响应数据。

如果响应数据超过 200KB，那么就不会被显示。要改变这一限制，请设置 JMeter 属性 view.results.tree.max_size。测试人员同样可以使用“Save Responses to a file”，将整个响应保存

到一个文件之中。

HTML 视图会尝试以 HTML 格式展现服务器响应。监听器处理得到的 HTML 与通过浏览器看到的页面会有所差别，它能迅速提供一个近似的页面，帮助用户进行响应结果评估。任何图形都不会被下载。如果选择了 HTML（Download Embedded Resources）选项，那么监听器可能会下载 HTML 引用的图形和 style-sheets 等，如图 12-7 所示。

XML 视图将会以树的形式展示服务器响应，如图 12-8 所示。任何 DTD 节点或者 Prolog 节点都不会在树中展示；不过，响应中可能会包含这些节点。

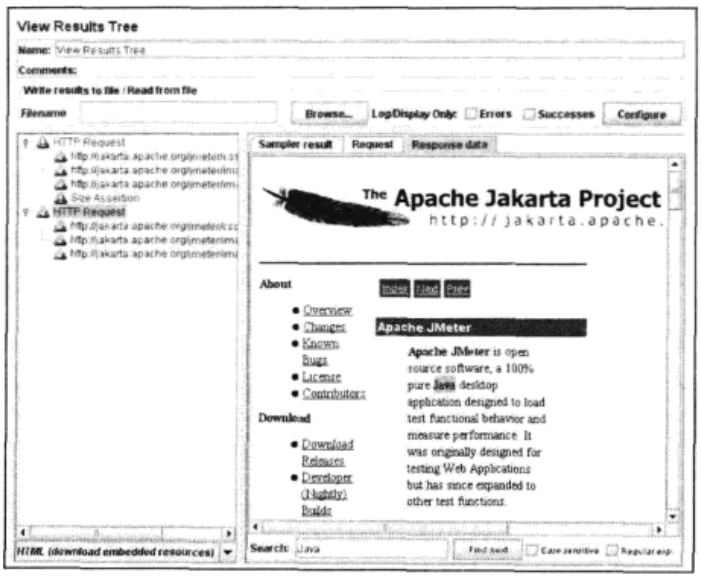


图 12-7　查看结果树（View Results Tree）的控制面板

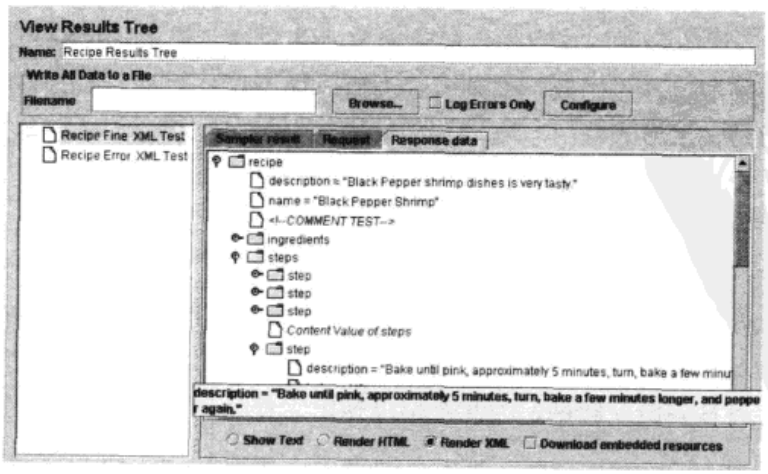


图 12-8　XML 视图

JSON 视图以树的形式展示服务器响应（同样会处理集成在 JavaScript 中的 JSON）。

大多数视图都支持对展示的数据进行搜索；搜索结果将会被高亮显示。例如，请参见图 12-7 中对“Java”的搜索结果。需要注意的是，在 TEXT 和 HTML 视图中的搜索结果可能会不同。

Regexp Tester 视图只对文本响应有效。它在上半部分面板中展示原始文本。用户可以使用“Test”按钮，以便将正则表达式应用到上半部分面板中去，匹配结果将会在下半部分面板中显示，如图 12-9 所示。例如，正则表达式(JMeter\w*).*应用到 JMeter 工具的官方主页上，将会获得如下匹配项：

```
Match count: 26
Match[1][0]=JMeter - Apache JMeter</title>
Match[1][1]=JMeter
Match[2][0]=JMeter" title="JMeter" border="0"/></a>
Match[2][1]=JMeter
Match[3][0]=JMeterCommitters">Contributors</a>
Match[3][1]=JMeterCommitters
... and so on ...
```

[]中的第一个数字是匹配编号；第二个数字表示组合。Group [0]匹配整个正则表达式。Group [1]匹配第一个组合，例如本例子中的(JMeter\w*)。

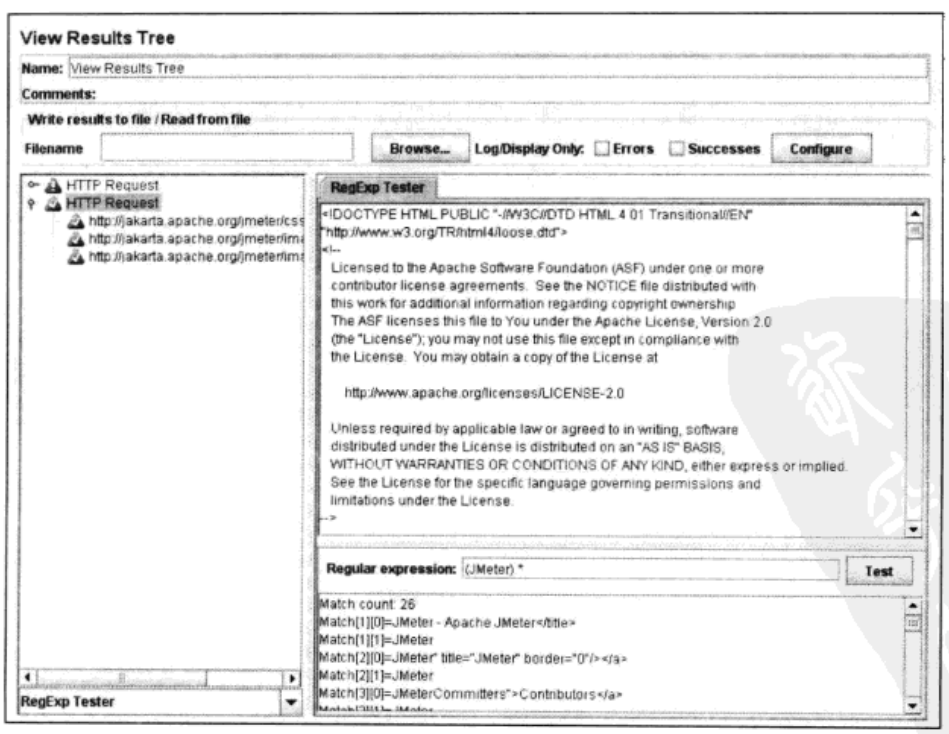


图 12-9　Regexp Tester 视图

5. 聚合报告（Aggregate Report）

聚合报告会为测试中的每一个不同采样，在表格中创建一行统计值，如图 12-10 所示。对每一个采样，它都会统计服务器响应信息，并提供请求数目、Min、Max、Average、Error %、Throughput（request/second）及 Throughput（Kilobytes per second）等统计值。一旦测试结束，那么吞吐率（Throughput）就是贯穿整个测试阶段的统计值。

吞吐率是从采样目标（如 HTTP 采样中的远程服务器）的角度来计算的。JMeter 会计算请求产生需要的总时间，如果同一个线程中存在其他采样器和定时器，就会增加总的时间，从而减小吞吐率的值。因此两个名称不同（其他完全相同）的采样器，相对于两个名称相同的采样器而言，吞吐率会减半。因此用户需要为采样器正确命名，才能通过聚合报告获取正确的值。

计算 Median（中间值）和 90% Line（90 %阈值）会占用更多内存。JMeter 2.3.4 及其以前版本，每个采样的细节信息都是独立存储的，这就意味着需要占用很多内存。新版本 JMeter 将同一时刻的采样绑定在一起，因此占用的内存会减少很多。不过，对于需要数秒才能完成的采样而言，意味着相同时刻的采样数会变少，在这种情况下就会需要更多内存。聚合报告与 Summary Report 的功能完全相同，不过监听器 Summary Report 不会存储单个采样的信息，因此只需要固定大小的内存。

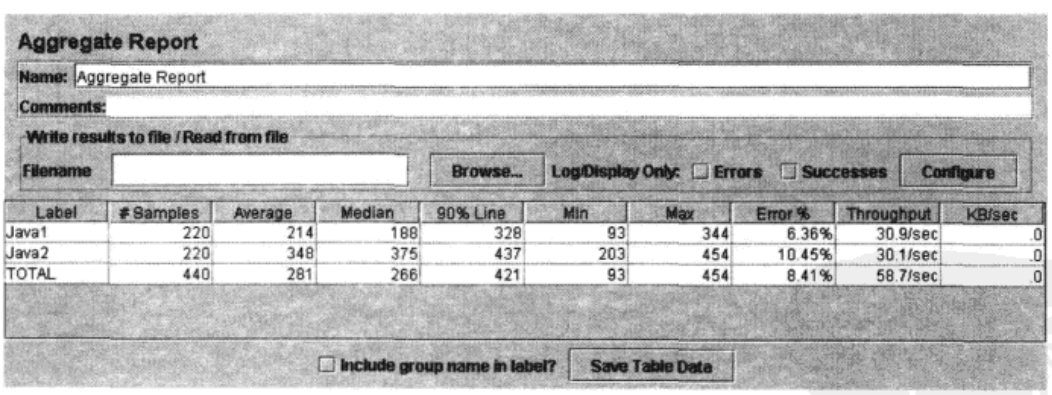


图 12-10 聚合报告

- Label：采样标签。如果选中了“Include group name in label?”复选框，那么线程组的名称就会作为前缀，如图 12-11 所示。 这样就能在必要的时候区分线程组不同而标签相同的采样。
- # Samples：标签名相同的总采样数。

- Average：一系列采样结果的平均响应时长。
- Median：一系列采样结果响应时长的中间值。50% 的采样响应时长不超过该值，剩下的采样响应时长不会比该值少。
- 90% Line：90%的采样响应时长不超过该值，剩下的采样响应时长不会比该值少。
- Min：标签名相同的采样中，最小的响应时长。
- Max：标签名相同的采样中，最大的响应时长。
- Error %：采样发生错误的比率。
- Throughput：该吞吐率以每秒/分钟/小时发生的采样数来衡量。单元时间已经选定，因此显示的吞吐率至少是 1.0。如果将吞吐率保存到 CSV 文件中，它以请求数/秒的格式保存，例如 30.0 请求/分钟被保存为 0.5。
- KB/sec：该吞吐率以每秒 KB 来衡量。

响应时长都以毫秒为单位。

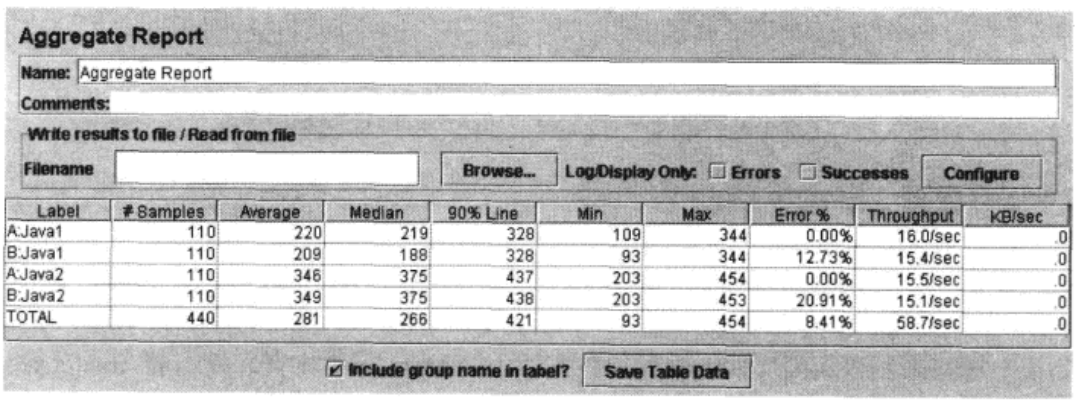
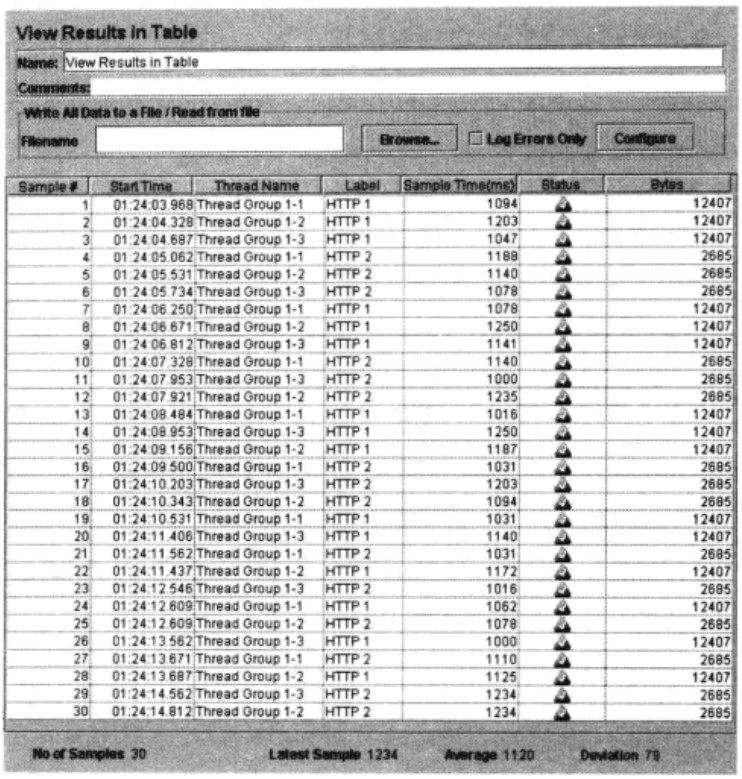


图 12-11　聚合报告（选中“Include group name in label?”复选框）

6. 用表格查看结果（View Results in Table）

该监听器为每一个采样结果创建一行（如图 12-12 所示）。同查看结果树（View Results Tree）相似，该监听器会占用较多内存。


View Results in Table

Name: View Results in Table

Comments:

Write All Data to a File / Read from file

Filename Browse... Log Errors Only Configure

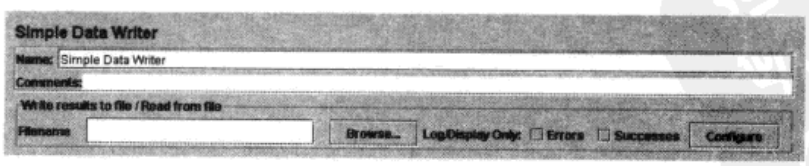
Sample #	Start Time	Thread Name	Label	Sample Time(ms)	Status	Bytes
1	01:24:03.968	Thread Group 1-1	HTTP 1	1094		12407
2	01:24:04.328	Thread Group 1-2	HTTP 1	1203		12407
3	01:24:04.687	Thread Group 1-3	HTTP 1	1047		12407
4	01:24:05.062	Thread Group 1-1	HTTP 2	1188		2685
5	01:24:05.531	Thread Group 1-2	HTTP 2	1140		2685
6	01:24:05.734	Thread Group 1-3	HTTP 2	1078		2685
7	01:24:06.250	Thread Group 1-1	HTTP 1	1078		12407
8	01:24:06.671	Thread Group 1-2	HTTP 1	1250		12407
9	01:24:06.812	Thread Group 1-3	HTTP 1	1141		12407
10	01:24:07.328	Thread Group 1-1	HTTP 2	1140		2685
11	01:24:07.953	Thread Group 1-3	HTTP 2	1000		2685
12	01:24:07.921	Thread Group 1-2	HTTP 2	1235		2685
13	01:24:08.484	Thread Group 1-1	HTTP 1	1016		12407
14	01:24:08.953	Thread Group 1-3	HTTP 1	1250		12407
15	01:24:09.156	Thread Group 1-2	HTTP 1	1187		12407
16	01:24:09.500	Thread Group 1-1	HTTP 2	1031		2685
17	01:24:10.203	Thread Group 1-3	HTTP 2	1203		2685
18	01:24:10.343	Thread Group 1-2	HTTP 2	1094		2685
19	01:24:10.531	Thread Group 1-1	HTTP 1	1031		12407
20	01:24:11.406	Thread Group 1-3	HTTP 1	1140		12407
21	01:24:11.562	Thread Group 1-1	HTTP 2	1031		2685
22	01:24:11.437	Thread Group 1-2	HTTP 1	1172		12407
23	01:24:12.546	Thread Group 1-3	HTTP 2	1016		2685
24	01:24:12.609	Thread Group 1-1	HTTP 1	1062		12407
25	01:24:12.609	Thread Group 1-2	HTTP 2	1078		2685
26	01:24:13.562	Thread Group 1-3	HTTP 1	1000		12407
27	01:24:13.671	Thread Group 1-1	HTTP 2	1110		2685
28	01:24:13.687	Thread Group 1-2	HTTP 1	1125		12407
29	01:24:14.562	Thread Group 1-3	HTTP 2	1234		2685
30	01:24:14.812	Thread Group 1-2	HTTP 2	1234		2685

No of Samples 30 Latest Sample 1234 Average 1120 Deviation 79

图 12-12 用表格查看结果（View Results in Table）

7. 简单保存数据（Simple Data Writer）

该监听器可以将测试结果数据保存到文件中（如图 12-13 所示），而不在 JMeter UI 中显示。它提供了一种高效记录数据的方法，可以不受 GUI 的限制。当以非 GUI 模式运行 Jmeter 时，可以使用-l 标签来创建数据文件。保存的数据域由 Jmeter 属性定义。请查看 jmeter.properties 文件以便获取更多信息。


Simple Data Writer

Name: Simple Data Writer

Comments:

Write results to file / Read from file

Filename Browse... Log/Display Only: Errors Successes Configure

图 12-13 简单保存数据（Simple Data Writer）

8. 监视器结果（Monitor Results）

监视器结果是一种新的监听器，用于展示服务器状态，如图 12-14 所示。它是针对 Tomcat 5 设计的，但是任何 Servlet 容器都可以部署该状态 Servlet，并使用这一监视器。该监听器有两个

主要选项卡：一个是“Health”选项卡，用于展示一个或者多个服务器的状态；另一个是“Performance”选项卡，用于展示某个服务器最近 1000 个采样的性能。用于计算负载值的方程式，内置于该监听器之中。

目前，对该监听器的使用主要受限于系统内存。这里有一个关于内存使用的快速基准，对于 100 台服务器的 1000 个数据点进行缓冲，需要 10MB 的 RAM。在一台 1.4GHz 主频、1GB 内存的笔记本电脑上，该监视器能够处理数百台服务器的监控任务。

这里有一个通用规则，即用户必须小心设置监视服务器的时间间隔。时间间隔小于 5 秒则监视太频繁，可能对服务器的性能产生冲击。如果采用 5 秒间隔，且缓存 1000 个数据点，那么该监视器每分钟会检测服务器 12 次，或者每小时 720 次。这就意味着缓存能够展示每台机器最近一小时的历史性能。

该监听器要求 Tomcat 5 或者更高版本。用户可以使用浏览器来检查 Tomcat Status Servlet 是否工作正常。

关于如何使用该监听器的更多细节，请参考本书“服务器监控测试脚本开发”一章。

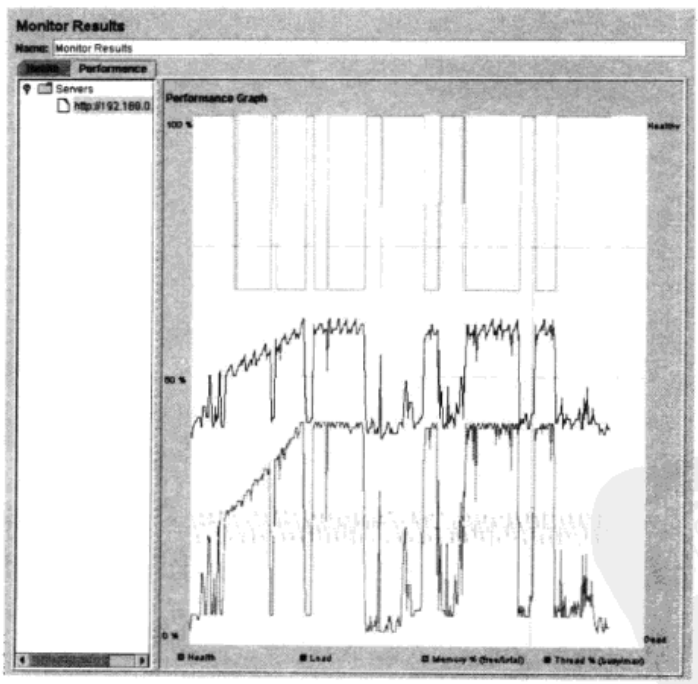


图 12-14　监视器结果（Monitor Results）

9. 分布图形（Distribution Graphs）

分布图形监听器会针对每个独立响应时长绘制一柱形图形，如图 12-15 所示。间隔粒度 System.currentTimeMillis()是 10ms，因此 90%阈值应该落在图形宽度以内。在图形中绘制有两

条阈值线：50%和 90%。这就意味着有 50%的响应时长落在 0 和 50%阈值线之间。90%阈值线也是这个含义。假设有多个针对 Tomcat 的测试，使用 30 个线程发送 600KB 请求，那么分布图完全可以支持这一情况，50%和 90%阈值线都会落在图形宽度之内。性能良好的应用系统，图形中的响应时长应该聚集在一起。而对于构建很差（存在内存泄漏）的应用系统，图形中的响应时长会发生跳跃起伏，在这一情况下，阈值线可能会超出图形的宽度。作者建议在这一情况下，上报缺陷并要求开发人员修复 WebApp 系统，接着重新测试。如果性能测试计划产生的分布图中，响应时长没有明显聚集或者存在某种规律，那么可能意味着存在内存泄漏。要想确认问题是否存在唯一的办法，就需要使用相关专业工具加以检测。

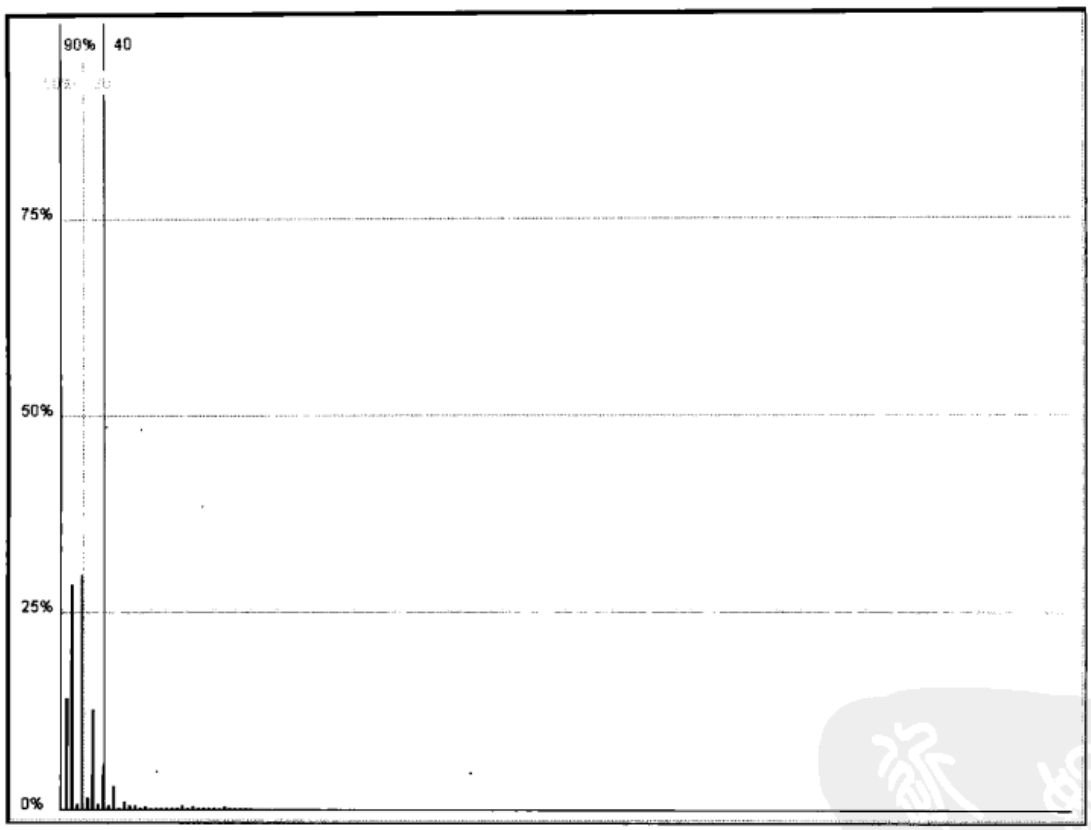


图 12-15　分布图形（Distribution Graphs）

10. 聚合图形（Aggregate Graphs）

聚合图形几乎等同于聚合报告，两者最主要的区别是聚合图形提供了生成对应柱状图，和将图形保存为 PNG 文件的方法，如图 12-16 所示。默认情况下，聚合图形会生成 450×250 像素的柱状图。

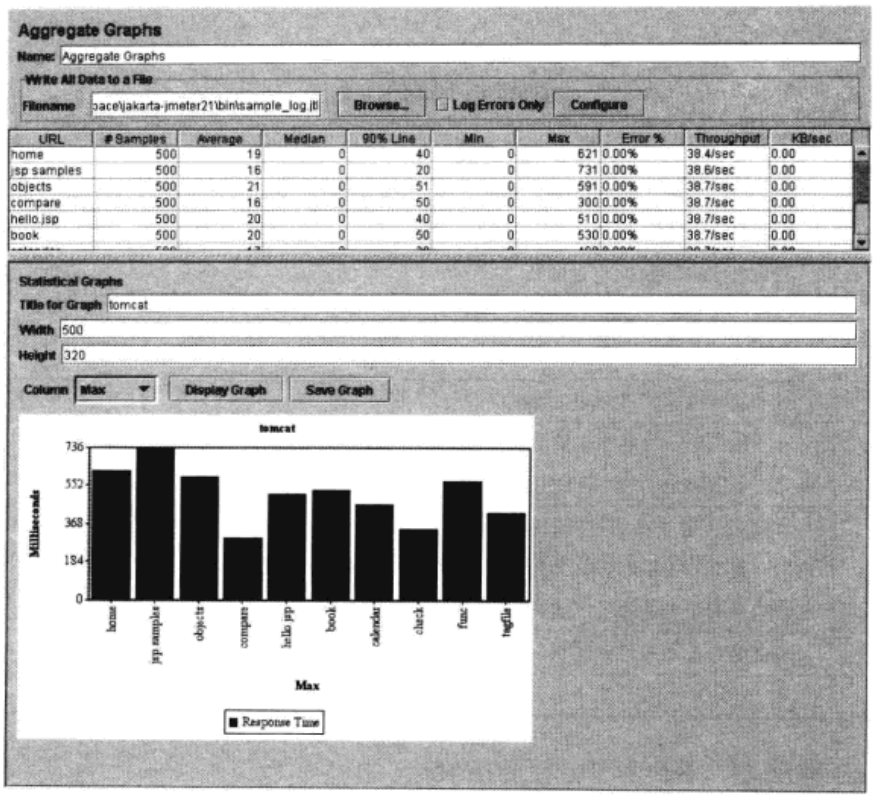


图 12-16　聚合图形（Aggregate Graphs）

11. 邮寄者查看器（Mailer Visualizer）

在测试运行期间从服务器接收到太多失败响应时，可以设置邮寄者查看器用于发送提醒邮件，如图 12-17 所示。

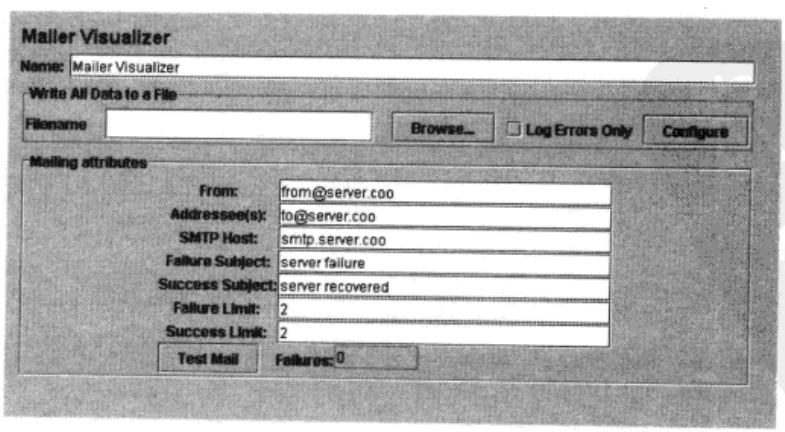


图 12-17　邮寄者查看器（Mailer Visualizer）

参数如表 12-1 所示。

表 12-1 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
From	发送邮件的地址	是
Addressee(s)	目标 E-mail 地址，逗号分隔	是
SMTP Host	SMTP 服务器（E-mail 转发者）的 IP 地址或者主机名	否
Failure Subject	对于失败消息的 E-mail 主题行	否
Success Subject	对于成功消息的 E-mail 主题行	否
Failure Limit	一旦超过这一失败响应的数目限制，一封失败提醒邮件就会被发送。例如，设置该值为 0，则发生第一个错误时，就发送提醒邮件	是
Success Limit	一旦超过这一成功响应的数目限制，且前面达到过失败响应数目限制，一封成功提醒邮件就会被发送。如此邮寄者只会以失败—成功—失败—成功的序列形式，发送提醒邮件	是
Test Mail	单击这一按钮发送一封测试邮件	否
Failures	一个字段用于保存迄今为止总共收到的失败消息数目	否

12. BeanShell 监听器（BeanShell Listener）

通过 BeanShell 监听器可以使用 BeanShell 来处理采样、保存信息等，如图 12-18 所示。

关于 BeanShell 的全面介绍，请访问 BeanShell 的 Web 站点：http://www.beanshell.org/。

该测试元件支持 ThreadListener 和 TestListener 两种方法。这应该在初始化文件中定义。请查看 BeanShellListeners.bshrc 文件中的定义范例。

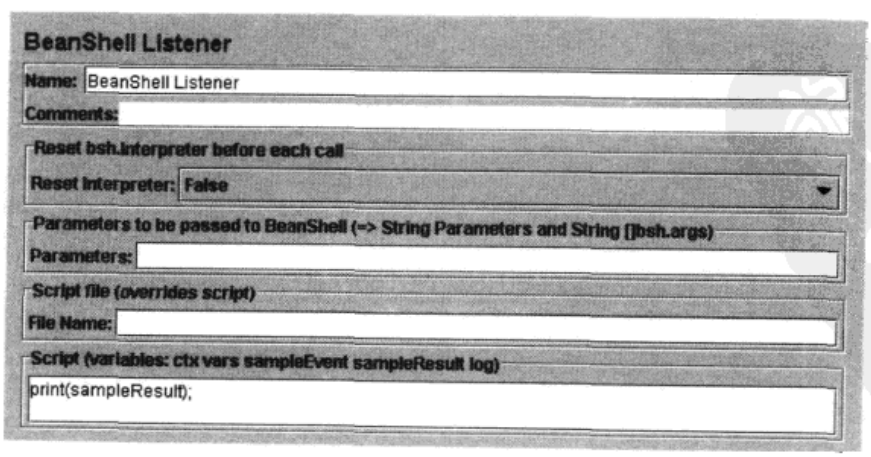


图 12-18 BeanShell 监听器（BeanShell Listener）

参数如表 12-2 所示。

表 12-2　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Reset bsh.Interpreter before each call	如果设置了该选项，那么每一次采样都会导致重新创建解释器。对于某些长时间运行的脚本，可能有必要这么做。请参见本书“详解 JMeter 最佳实践经验”一节	是
Parameters	传递给 BeanShell 脚本的参数。这些参数被存储于如下变量之中： ■ Parameters：字符串，以单个变量的形式包含参数 ■ bsh.args：字符串数组包含参数，以空格加以分隔	否
Script file	文件，包含待运行的 BeanShell 脚本	否
Script	待运行的 BeanShell 脚本。脚本返回值将被忽略	是（除非提供了脚本文件）

在唤醒脚本之前，BeanShell 解释器已经初始化了部分变量。

- log - (Logger)：可以用来记录日志文件。
- ctx - (JMeterContext)：可以通过它来访问 Context。
- vars - (JMeterVariables)：可以通过它来读/写变量：

```
vars.get(key); vars.put(key,val);
vars.putObject("OBJ1",new Object());
```

- props - JMeter 属性，例如：

```
props.get("START.HMS"); props.put("PROP1","1234");
```

- sampleResult - (SampleResult)：可以通过它来访问前面的采样结果。
- sampleEvent -(SampleEvent)：可以通过它来访问当前采样事件。

关于上述变量可以使用的方法，更多细节请参考对应 Javadoc。

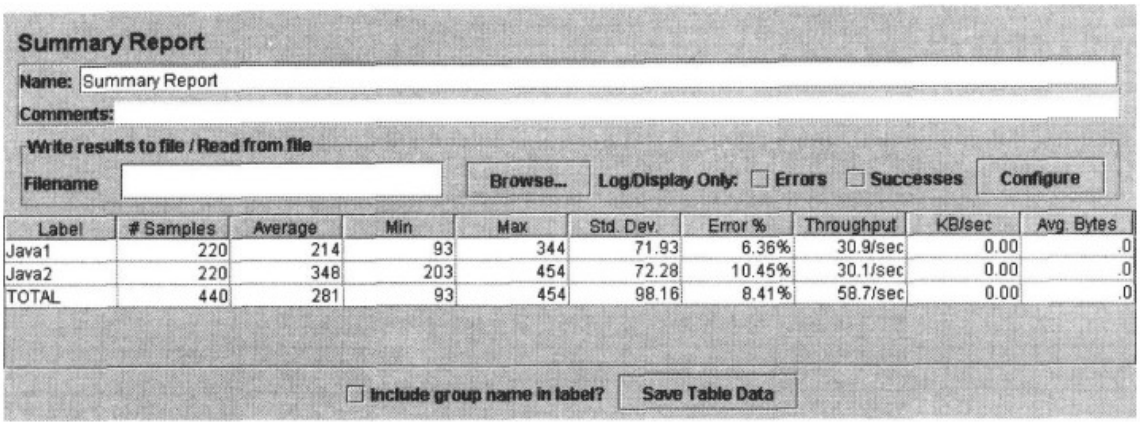
如果定义了属性 beanshell.listener.init，那么它可以用于加载初始化文件。在该文件中可以定义 BeanShell 脚本将会用到的方法。

13. 概要报告（Summary Report）

概要报告会为测试中每一个不同名称的请求，在表格中单独创建一行，如图 12-19 所示。这一点与聚合报告相同，区别在于消耗的内存较少。

吞吐率是从采样目标（例如 HTTP 采样中的远程服务器）的角度来计算的。JMeter 会计算

请求产生需要的总时间，如果同一个线程中存在其他采样器和定时器，就会增加总的时间，从而减小吞吐率的值。因此两个名称不同（其他完全相同）的采样器，相对于两个名称相同的采样器而言，吞吐率会减半。因此用户需要为采样器正确命名，才能通过概要报告获取正确的值。



Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
Java1	220	214	93	344	71.93	6.36%	30.9/sec	0.00	.0
Java2	220	348	203	454	72.28	10.45%	30.1/sec	0.00	.0
TOTAL	440	281	93	454	98.16	8.41%	58.7/sec	0.00	.0

图 12-19　概要报告（Summary Report）

- Label：采样标签。如果选中了“Include group name in label?”复选框，那么线程组的名称就会作为前缀，如图 12-20 所示。这样就能在必要的时候区分线程组不同而标签相同的采样。
- # Samples：标签名相同的总采样数。
- Average：一系列采样结果的平均响应时长。
- Min：标签名相同的采样中，最小的响应时长。
- Max：标签名相同的采样中，最大的响应时长。
- Std. Dev.：采样响应时长的标准差。
- Error %：采样发生错误的比率。
- Throughput：该吞吐率以每秒/分钟/小时发生的采样数来衡量。单元时间已经选定，因此显示的吞吐率至少是 1.0。如果将吞吐率保存到 CSV 文件中，它以请求数/秒的格式保存，例如 30.0 请求/分钟被保存为 0.5。
- KB/sec：该吞吐率以每秒 KB 来衡量。
- Avg. Bytes：以字节为单位的采样响应平均大小（在 JMeter 2.2 错误地以 KB 为单位显示值）。

响应时长都以毫秒为单位。

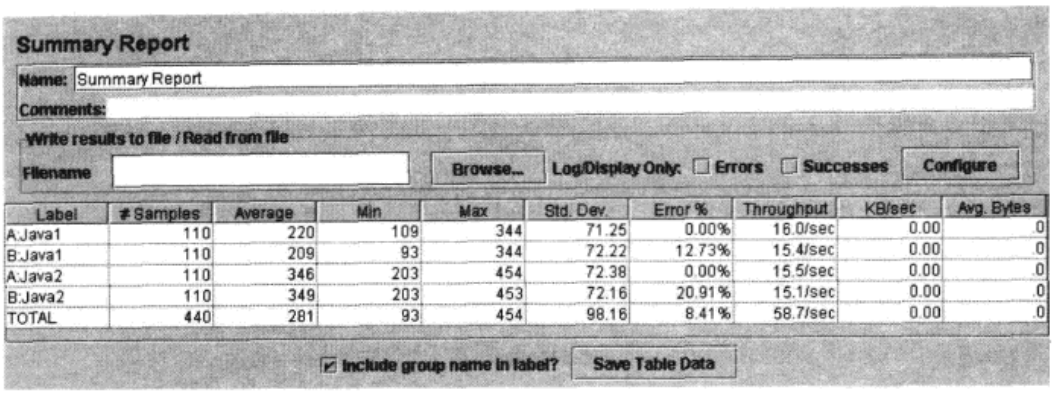


图 12-20　概要报告选中“Include group name in label?”复选框）

14．保存响应到文件（Save Responses to a file）

该测试元件可以放在测试计划中的任意位置。对于每一个在其作用域范围内的采样，它都会创建一个保存响应数据的文件。它主要用于功能测试，另外还可以用于响应消息太大的情况（无法在查看结果树中展示）。文件名由指定前缀加一个数字构成（除非这一功能被关闭了）。文件扩展名是通过文档类型创建的（如果清楚的话）。如果不清楚，就将文件扩展名设置为“unknown”。如果编号功能被禁止了，而且添加后缀功能也被禁止了，那么 file prefix 将被作为整个文件名。在必要的时候，可以产生固定的文件名。产生的文件名将保存在采样响应中，必要时还可以保存到测试日志输出文件中。

当前采样会首先保存，紧跟着是子采样。如果提供了变量名，那么文件名将会以子采样出现的顺序加以保存，如图 12-21 所示。

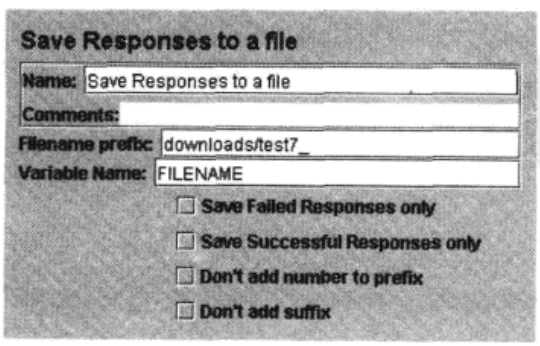


图 12-21　保存响应到文件（Save Responses to a file）

参数如表 12-3 所示。

表 12-3 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Filename prefix	生成文件名的前缀；可以包含一个目录名	是
Variable Name	变量名，该变量中包含生成的文件名（那么该变量可以用在测试计划之中）。如果有子采样，那么就会在变量名后加后缀。例如，假如变量名是 FILENAME，那么父采样的文件名将被保存到变量 FILENAME 中，而子采样的文件名将保存到 FILENAME1、FILENAME2 等变量中	否
Save Failed Responses only	如果选中该复选框，只有失败的响应会被保存	是
Save Successful Responses only	如果选中该复选框，只有成功的响应会被保存	是
Don't add number to prefix	如果选中该复选框，就不会为前缀加上编号。如果选中该复选框，那么请确保前缀是唯一的或者记录的文件可以被覆盖	是
Don't add suffix	如果选中该复选框，那么就不会添加后缀。如果选中该复选框，那么请确保前缀是唯一的或者记录的文件可以被覆盖	是

15. BSF 监听器（BSF Listener）

通过 BSF 监听器可以将 BSF 脚本代码应用于采样结果，如图 12-22 所示。

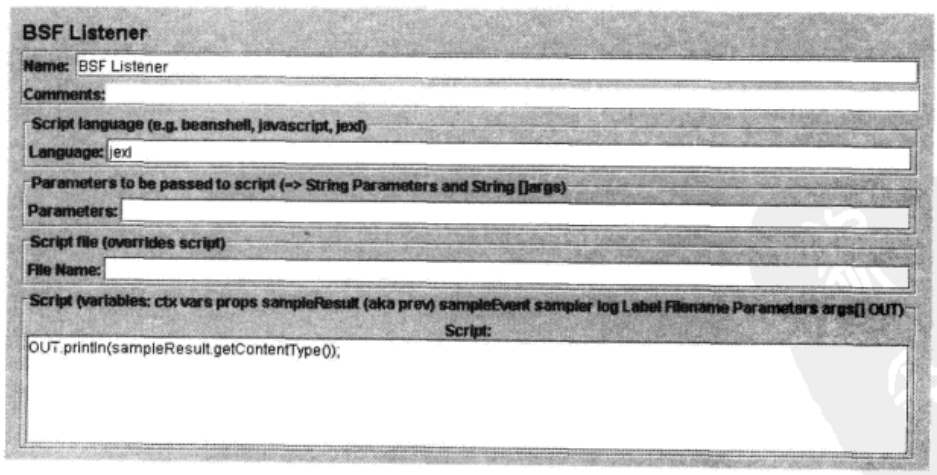


图 12-22 BSF 监听器（BSF Listener）

参数如表 12-4 所示。

表 12-4　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	否
Language	指明 BSF 使用的语言	是
Parameters	传递给脚本的参数。这些参数被存储于如下变量之中。 ■ Parameters：字符串，以单个变量的形式包含参数 ■ bsh.args：字符串数组包含参数，以空格加以分隔	否
Script file	文件，包含待运行的脚本	否
Script	待运行的脚本。脚本返回值将被忽略	是（除非提供了脚本文件）

BSF 监听器将会使用 BSFEngine.exec()方法对脚本（或者文件）进行处理，脚本执行不会有返回值。

在唤醒脚本之前，已经初始化了部分变量。注意它们是 BSF 变量，它们可以在脚本中直接使用。

- log - (Logger)：可以用来记录日志文件。
- Label：字符串标签；
- Filename：脚本文件名（如果存在）。
- Parameters：参数（字符串形式）。
- args[]：参数，字符串数组形式（以空格分隔）。
- ctx - (JMeterContext)：可以通过它来访问 Context；
- vars - (JMeterVariables)：可以通过它来读取/访问 JMeter 变量：

```
vars.get(key); vars.put(key,val); vars.putObject("OBJ1",new Object());
vars.getObject("OBJ2");
```

- props - JMeter 属性，例如：

```
props.get("START.HMS");
props.put("PROP1","1234");
```

- sampleResult-(SampleResult)：可以通过它来访问采样结果。
- sampleEvent- (SampleEvent)：可以通过它来访问采样事件。
- sampler - (Sampler)：可以通过它来访问最近的采样器。

■ OUT - System.out：例如 OUT.println("message")。

关于上述变量可以使用的方法，更多细节请参考对应 Javadoc。

16. JSR223 监听器（JSR223 Listener）

通过 JSR223 监听器可以将 JSR223 脚本代码应用于采样结果，如图 12-23 所示，更多细节请参考 BSF 监听器。

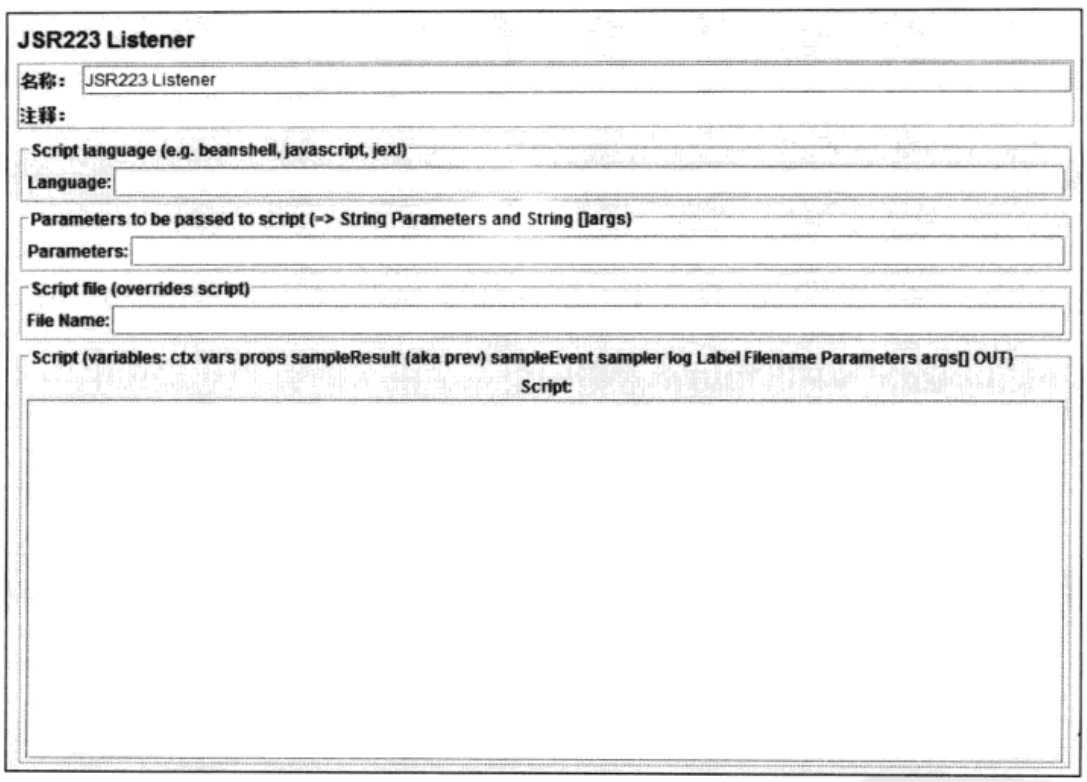


图 12-23 JSR223 监听器（JSR223 Listener）

17. 生成概要结果（Generate Summary Results）

该测试元件（如图 12-24 所示）可以被放置在测试计划中的任何位置。它会产生一个关于测试运行的概要，并输出到日志文件和/或者标准输出中。该测试元件将在合适的时间点，间隔 *n* 秒（默认 3 分钟）输出一次，因此，同一时刻的多个测试运行将被同步处理。间隔时间由属性“summariser.interval”定义（参考 jmeter.properties 文件）。该测试元件主要用于非 GUI 模式。输出如下面一般：

```
label+171 in  20.3s=8.4/s Avg:  1129 Min:  1000 Max:  1250 Err:     0 (0.00%)
label+263 in  31.3s=8.4/s Avg:  1138 Min:  1000 Max:  1250 Err:     0 (0.00%)
```

```
label=434 in  50.4s=8.6/s Avg:  1135 Min:  1000 Max:  1250 Err:     0 (0.00%)
label+263 in  31.0s=8.5/s Avg:  1138 Min:  1000 Max:  1250 Err:     0 (0.00%)
label=697 in  80.3s=8.7/s Avg:  1136 Min:  1000 Max:  1250 Err:     0 (0.00%)
label+109 in  12.4s=8.8/s Avg:  1092 Min:    47 Max:  1250 Err:     0 (0.00%)
label =   806 in  91.6s =    8.8/s Avg:  1130 Min:    47 Max:  1250 Err:     0
(0.00%)
```

“label”是对应测试元件的名称。“+”意味着该行是差值行，例如，展示从上一次输出以后发生了多少变化。“=”意味着该行是合计行，例如，展示总的运行数。JMeter 日志文件中的条目还包含时间戳。上面的例子中，“806 in 91.6s = 8.8/s”意味着在 91.6 秒内记录了 806 次采样，可以计算出每秒 8.8 次采样。Avg (Average)，Min(imum)和 Max(imum)的时间单位为毫秒。“Err”指出错误数目（还以百分比形式显示）。最后两行出现在测试结尾，它们不会等待合适的时间点。请注意初始和最后的差值可能会小于指定的时间间隔（在上面的例子中是 30 秒）。第一个差值通常比较小，原因在于 JMeter 会等待间隔时间点。最后一个差值也会比较小，因为测试通常不会碰巧在某个间隔时间点结束。

标签（Label）用于将采样结果组合起来。因此假如测试人员有多个线程组，而又想跨越线程组进行概要统计，那么就使用相同的标签，或者将该测试元件直接放在测试计划之下（如此所有线程组都在它的作用域范围内）。通过合理使用标签（Label），以及将生成概要结果（Generate Summary Results）合理添加到测试计划中，可以实现不同的概要组合。

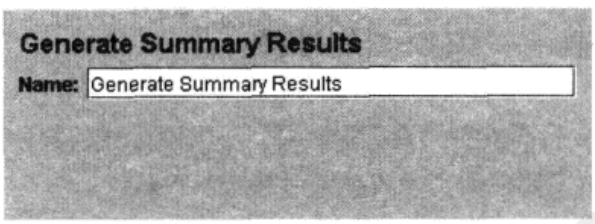


图 12-24　生成概要结果（Generate Summary Results）

参数如表 12-5 所示。

表 12-5　参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件。它会作为标签（Label）出现在输出中。所有标签相同的测试元件的细节信息，将被合在一起	是

18. 比较断言查看器（Comparison Assertion Visualizer）

比较断言查看器可以用于展示任何 Compare Assertion 测试元件的结果，如图 12-25 所示。

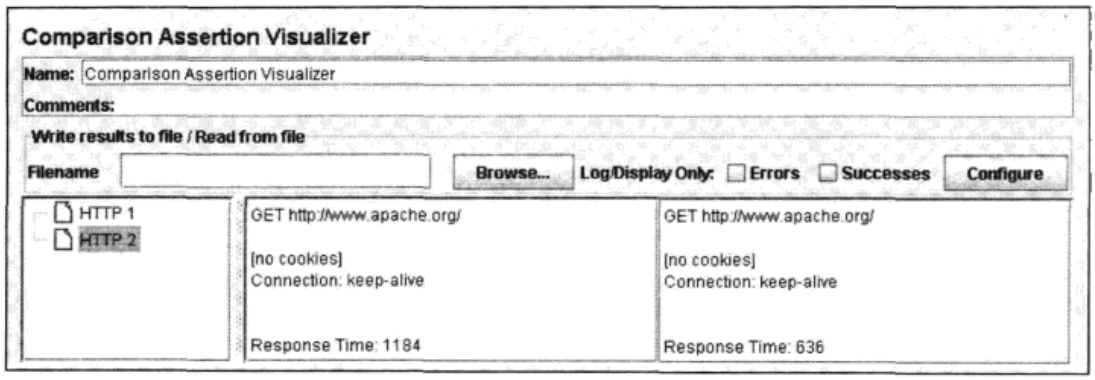


图 12-25 比较断言查看器（Comparison Assertion Visualizer）

参数如表 12-6 所示。

表 12-6 参数描述

属性	描述	是否必需
Name	该元件的描述性名称，用于在测试树中标识该元件	是

12.2.2 巧用监听器——识别性能缺陷

上一节介绍了 JMeter 目前支持的监听器种类。不同监听器的特性各异，为了更快更便捷地识别性能缺陷，测试人员需要根据实际情况灵活挑选监听器。

实际工作（Web 性能测试）中最常用到的监听器包括：图形结果（Graph Results）、查看结果树（View Results Tree）、样条视图（Spline Visualizer）、聚合报告（Aggregate Report）、用表格查看结果（View Results in Table）、简单保存数据（Simple Data Writer）、监视器结果（Monitor Results）、分布图形（Distribution Graphs）、聚合图形（Aggregate Graphs）、概要报告（Summary Report）。

下面将会介绍一些经验，希望能够帮助读者朋友快速入门。

1. 受运行 JMeter 的机器硬件性能限制，尽量不要使用 JMeter GUI 模式运行性能测试

绘制图形的监听器更是会占用大量内存。测试人员应该使用非 GUI 模式运行 JMeter，并使

用简单保存数据（Simple Data Writer）记录测试数据，待测试执行完毕后，再使用其他监听器对数据进行分析。在测试运行期间，测试人员需要关注 JMeter 占用 CPU 和内存的情况（如图 12-26 所示），以便确保测试数据真实有效。

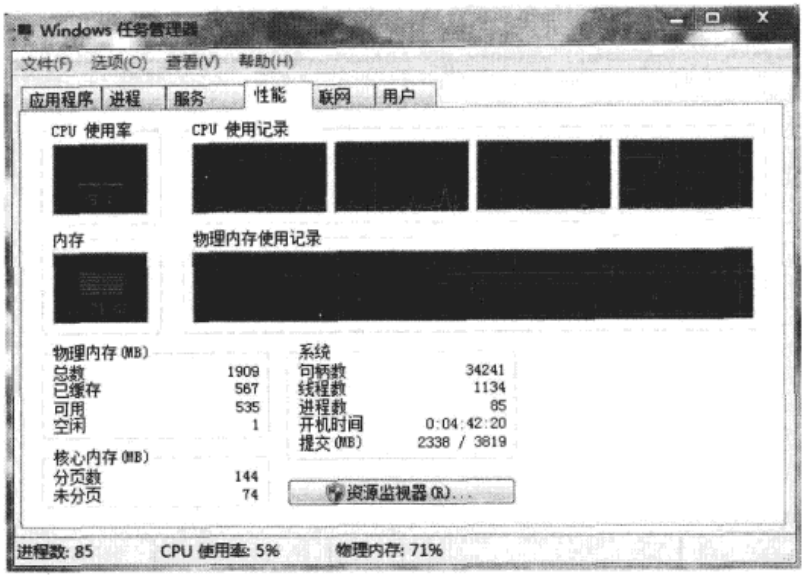


图 12-26　监控 JMeter 测试机 CPU 和内存占用情况

2. 图形结果（Graph Results）——性能测试的王者利器

图形结果主要包含 4 种统计信息，即当前采样响应时长（黑色）、当前平均采样响应时长（蓝色）、当前采样响应时长标准差（红色）、当前吞吐率（绿色），如图 12-4 所示。其中最有价值的是：平均采样响应时长、采样响应时长标准差、吞吐率。

（1）通过观察平均采样响应时长曲线，用户可以直观地看到，随着并发压力的加大，以及性能测试时间的延长，系统性能所发生的变化。正常情况下，平均采样响应时长曲线应该是平滑的，并大致平行于图形下边界。

下面是几种可能存在性能问题的平均采样响应时长曲线的示意图。

①平均响应时长在初始阶段跳升，而后逐渐平稳下来，如图 12-27 所示。这说明系统在初始阶段响应较慢，导致这种现象的原因有一种：一是系统在初始阶段存在性能缺陷，需要进一步优化，如数据库查询缓慢。二是系统有缓存机制，而性能测试数据在测试期间没有变化，如此一来同样的数据在初始阶段的响应时长肯定较慢，而后续响应时长基本一致。这属于性能测试数据准备的问题，不是性能缺陷，需调整后再继续测试。三是系统架构设计导致的固有现象，例如，

在系统接收到第一个请求后，才去建立应用服务器到数据库的连接，后续一段时间内不会释放连接。这种情况是否属于性能缺陷，需要测试人员与系统架构师、开发人员、最终用户等角色一起确认。

图 12-27　平均响应时长曲线（1）

②平均响应时长持续变大，图形变得越来越陡峭，如图 12-28 所示。对于这种情况，基本可以肯定存在性能缺陷，如内存泄漏。测试人员需要及时上报缺陷，并协助开发定位问题。测试人员可以通过监控系统日志、监控应用服务器状态等常见方法，来尝试定位问题。

图 12-28　平均响应时长曲线（2）

③平均响应时长在性能测试期间，突然发生跳变，然后又恢复正常，如图 12-29 所示。对于这一情况，既可能源于系统性能缺陷，又可能是由于测试环境不稳定所造成的。分析问题，首先要排除测试环境不稳定带来的影响，一是检查应用服务器状态（CPU 占用、内存占用等），如果待测系统与其他应用系统共用服务器，需排除其他系统带来的影响。二是检查测试环境网络是否存在拥塞。如果能够排除测试环境的影响，测试人员接下来应该通过多种手段（监控应用服务器、监控数据库、监控系统日志等）来确认是否存在性能缺陷，例如，对某些特定测试

数据的处理存在问题。

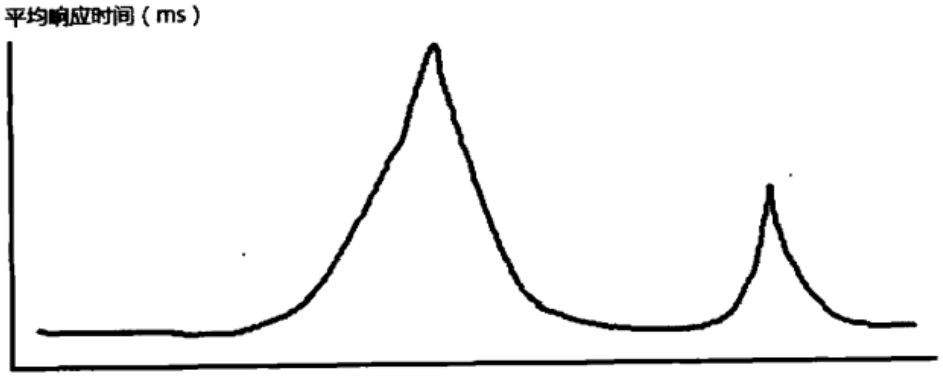


图 12-29　平均响应时长曲线（3）

（2）通过观察采样响应时长标准差，可以判断采样数据的分布是否均匀。当采样数据分布不均时，提示可能存在性能缺陷。标准差（Standard Deviation），也称均方差（Mean Square Error），是各数据偏离平均数的距离的平均数，它是离均差平方和平均后的方根，用 σ 表示。标准差是方差的算术平方根。标准差能反映一个数据集的离散程度。平均数相同的，标准差未必相同。如图 12-30 所示为标准差的计算公式。标准差越高，表示测试数据越离散；反之，标准差越低，代表测试的数据越平稳。显然，理想的采样响应时长标准差曲线，也应该是平滑的。

$$\alpha = \sqrt{\frac{\sum_{i=1}^{n}(X_i - X)^2}{n-1}}$$

图 12-30　标准差计算公式

（3）吞吐率表征服务器每分钟处理的实际采样数。测试人员可以通过增加并发线程数或减少脚本中的延迟，来找到系统支持的最大吞吐率。接着将系统实际支持的最大吞吐率和预期吞吐率进行比较，以便确认系统表现是否满足用户需求。参考图 12-4 中的绿色吞吐率曲线，该曲线随着并发压力逐步加大，逐渐趋于平滑，这就说明系统达到了其支持的最大吞吐率。如果进一步加大压力，系统响应会变慢，甚至发生崩溃。如果系统支持的最大吞吐率不能达到系统设计预期值，测试人员应该及时上报性能缺陷，并协助开发人员定位问题。

3. 查看结果树（View Results Tree）——用于调试性能测试脚本

某些情况下，测试人员需要关心服务器的响应数据。例如，保险公司客服人员在保单查询页面上输入条件，查询符合要求的保单数据，接着对某个保单进行操作。在这种情况下，用户下一步的操作依赖于上一步的查询结果，如此就需要用到查看结果树。通过它，测试人员可以

检视查询结果（请参考图 12-7 和图 12-9），并利用正则表达式，从响应结果中提取数据，以供后续测试使用。关于 JMeter 正则表达式的相关内容，请参考本书“详解 JMeter 正则表达式”一节。在大并发量性能测试过程中，查看结果树并不适合用于分析测试结果。

4. 样条视图（Spline Visualizer）——用于实时观察系统响应时长的变化

测试人员可以使用样条视图实时观察系统响应时长的变化。样条视图与图形结果（Graph Results）的平均采样响应时长曲线有些近似，都会体现系统响应时长的变化。不过，二者也有区别，样条视图表征系统响应时长的直接变化，而平均采样响应时长曲线表征平均值变化。二者是正相关的，但是相对于平均采样响应时长曲线而言，样条视图的变化应该更剧烈一些，更陡峭一些。关于样条视图的分析方法，请参考前面介绍的平均采样响应时长曲线的分析方法，分析的基本方法是一致的。需要注意的是，样条视图更适合于对系统某一项功能的性能测试数据进行分析，例如对某保险公司电话销售系统的保单查询功能单独进行分析。

5. 聚合报告（Aggregate Report）——综合判断系统性能是否满足要求

聚合报告以表格的形式展示一系列统计值，包括请求数目、最小响应时长、最大响应时长、平均响应时长、错误率、中间值、90 %阈值、吞吐率（每秒完成请求数）及吞吐率（每秒千字节）。聚合报告会为每一个不同采样，创建一行统计值。这样测试人员就可以通过聚合报告，综合判断系统性能是否满足要求。我们最关心的几个统计值，包括平均响应时长、90 %阈值、吞吐率（每秒完成请求数）、错误率。

如表 12-7 所示为是某大型保险公司电话销售系统的性能指标要求，出于保护商业机密的原因，作者将其中部分关键数据隐去了。不过，这并不妨碍我们以此为例。

表 12-7　电话销售系统的性能指标要求

关键功能	平均使用次数	平均用户数	高峰段用户数	平均响应时长	可接受最长响应时长	使用时间段
Xxx 功能	X 次	20000	23000	<4 s	≤7 s	8:00-21:00
Xxx 功能	X 次	20000	23000	<3 s	≤6 s	8:00-21:00
Xxx 功能	X 次	20000	23000	<3 s	≤6 s	8:00-21:00
Xxx 功能	X 次	20000	23000	<2 s	≤4 s	8:00-21:00
Xxx 功能	X 次	20000	23000	<6 s	≤15 s	8:00-21:00
Xxx 功能	X 次	20000	23000	<3 s	≤6 s	8:00-21:00

假设我们要判断表格中第一项功能是否满足预期要求，那么需要按如下步骤进行判断。

（1）错误率应该为 0。如果错误率不为 0，测试人员就必须仔细分析，这提示系统可能存在缺陷，例如无法正确处理某些类型的测试数据。

（2）聚合报告统计的“90 %阈值”，应该小于等于表格中的“可接受最长响应时长”，即小于等于 7 秒。为什么使用“90 %阈值”而非“最大响应时长”？原因在于，在长达数小时的性能测试过程中，数万甚至数十万次操作中，偶尔出现一两次“最大响应时长”超过“可接受最长响应时长”是可以接受的。导致这种情况发生的原因很多，比如测试网络中出现了瞬时拥塞。要去定位这样的偶发性事件是很困难的，甚至是不可重现的。

（3）聚合报告统计的“平均响应时长”，应该小于等于表格中的“平均响应时长”，即小于等于 4 秒。

（4）测试人员需要先计算系统设计吞吐率，系统设计吞吐率=(高峰段用户数/平均用户数)×(平均用户数×平均使用次数)/系统使用时间。“(平均用户数×平均使用次数)”表示系统某项功能一天之内平均的操作次数；“(平均用户数×平均使用次数)/系统使用时间)”表示系统某项功能一天之内平均的吞吐率；“(高峰段用户数/平均用户数)”表示高峰时段业务增加的比例。用“高峰时段业务增加的比例”乘以“系统某项功能一天之内平均的吞吐率”，就可以得到高峰时段系统需要支持的吞吐率。系统设计吞吐率应该小于等于聚合报告统计的吞吐率(每秒完成请求数)。

小贴士

①如果聚合报告统计的吞吐率（每秒完成请求数）没有达到系统设计吞吐率，则需要进一步检查是否对系统施加了足够大的压力。测试人员可以使用图形结果做进一步分析，以便更直观地观察吞吐率变化情况。

②概要报告的分析办法同聚合报告一致。

6. 用表格查看结果（View Results in Table）——逐条分析性能测试数据

如果测试人员从“图形结果”或者“样条视图”中看到了异常情况，而想进一步定位问题，就可能会用到“用表格查看结果”。原因很简单，监听器“图形结果”和“样条视图”没有准确的时间刻度，测试人员只能粗略估计采样发生的时间。

在“用表格查看结果”中，测试人员可以看到采样时间、系统响应时长、字节数等，如图 12-12 所示。这能帮助测试人员确定问题采样发生的准确时刻。

7. 监视器结果（Monitor Results）——寻找最容易说服开发人员的证据

假如测试人员问一位有经验的测试人员，测试工作中最困难的是什么？答案一定是如何证

明缺陷真的存在。让我们先看一个故事："通用汽车有一个品牌叫 Pontiac，曾经收到过一个投诉。顾客说他们家每天晚饭后要吃冰激凌，惯例是全家决定了吃什么口味然后他开车去商店购买。问题出在他新买的汽车身上。他每次开车去买冰激凌，如果买的是香草味的，车就无法启动；如果是其他口味，就没有问题。汽车公司的经理虽然很怀疑事情的真实性，但还是派了一个工程师去解决这个投诉。"如果在现实工作中，作为测试人员的测试人员，发现了一个类似的问题。测试人员该如何向开发人员清晰描述，并说服他真的存在缺陷呢？

监视器结果也许就能帮上测试人员的大忙，与其向开发费劲地描述"系统性能正在变差，××功能的响应时长大于 x 秒"，不如将有问题的服务器状态展示给他看。如果内存占用曲线显示，内存占用率越来越高，那么毫无疑问存在内存泄漏。如果监视器结果指出服务器线程被大量占用，而得不到释放，导致后续的用户请求无法处理，那么毫无疑问肯定存在缺陷。将监视器结果的截图作为缺陷存在的证据，无疑将提高缺陷的质量，并能帮助开发人员定位问题。

8. 分布图形（Distribution Graphs）——通过观察系统响应时长分布规律，寻找隐藏起来的缺陷

如果聚合报告中的各项统计值都满足系统性能要求，是否意味着不存在缺陷？答案是不一定。性能良好的应用系统，图形中的响应时长应该聚集在一起。而对于构建很差（存在内存泄漏）的应用系统，图形中的响应时长会发生跳跃起伏，在这一情况下，阈值线可能会超出图形的宽度。响应时长不聚集或者无规律，经常发生跳跃起伏，提示系统可能存在隐藏缺陷。最常见的问题是内存泄漏，或者系统对某类测试数据的处理效率太低，也可能是由于测试网络延迟造成的影响，还可能是共用服务器的其他系统带来的影响……

9. 聚合图形（Aggregate Graphs）——寻找加载最慢的页面

聚合图形几乎等同于聚合报告，唯一的区别是聚合图形内嵌了柱状图，如图 12-16 所示。通过柱状图，测试人员可以直观地找到加载最慢的页面。如此一来，开发人员对于优化系统性能就有了具体的目标。不过，优秀的测试人员还可以做得更好，比如向开发人员指出××页面的某处有优化的空间。如何做到了？答案就是大名顶顶的 YSlow。YSlow 是一款由 YAhoo 发布的基于 FireFox 的插件。该插件可以分析网站的页面，并告诉测试人员为了提高网站性能，如何基于某些规则而进行优化。安装 YSlow 前，需要在 FireFox 先安装另外一个流行的 Web 开发工具 Firebug，安装完 YSlow 后，就可以利用 YSlow 来分析网页的 HTML 代码及 JavaScript 代码，并对其进行优化。YSlow 可以分析任何网站，并为每一个规则产生一个整体报告，如果页面可以进行优化，则 YSlow 会列出具体的修改意见，如图 12-31 所示。

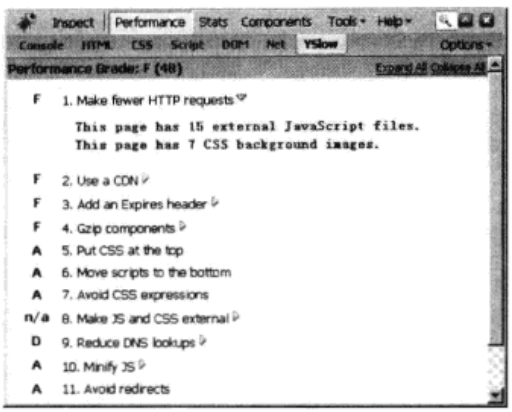


图 12-31　YSlow 图例（1）

YSlow 还具有统计分析功能，可以分析总共下载的网页大小、缓存及 Cookies，如图 12-32 所示。

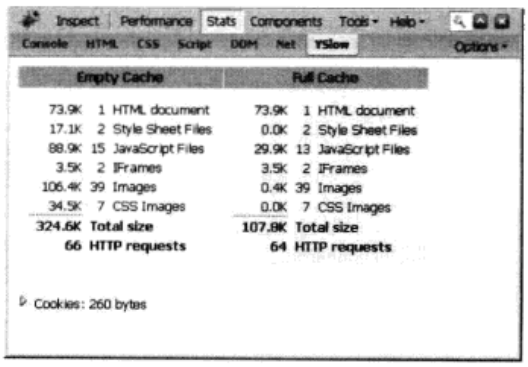


图 12-32　YSlow 图例（2）

YSlow 可以列出页面所有组件，包括组件类型、URL、失效日期、压缩状态、装载时间、大小等，如图 12-33 所示；还可以查看 HTTP 响应头的任何部分。

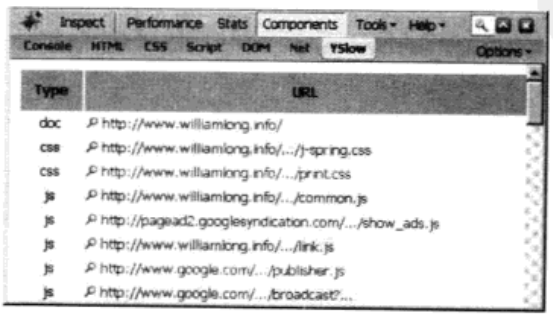


图 12-33　YSlow 图例（3）

如果需要了解更多 YSlow 的特点，请访问 http://developer.yahoo.com/yslow/。

12.3 借助 Ant 实现批量测试和报表生成

Ant 是一种基于 Java 的 build 工具。从理论上来说，它有些类似于（UNIX）C 中的 make ，但没有 make 的缺陷。编著本书时的最新版本为：Ant 1.8.2。Ant 工程的配置文件采用 XML 格式描述，支持多平台编译，比较适合大型工程。在使用 Ant 之前，先要从 http://www.apache.org/dist/ant/binaries/ 下载 ZIP 包（如 apache-ant-1.8.2-bin.zip），接着解压、安装到本地目录。另外，还需要设置环境变量 ANT_HOME，将其值设为 Ant 安装目录的路径，并在系统 Path 目录中添加%ANT_HOME%/bin，在 cmd 窗口中输入 ant-version，如果显示如下信息，则表示安装已经成功。

```
D:\Users\wensujian815>ant -version
Apache Ant version 1.7.0 compiled on December 13 2006
```

将 JMeter 项目对应的.jmx 文件，放入 extras 目录下，即可使用 Ant 实现测试的自动化。几个有用的 Ant 命令如下：

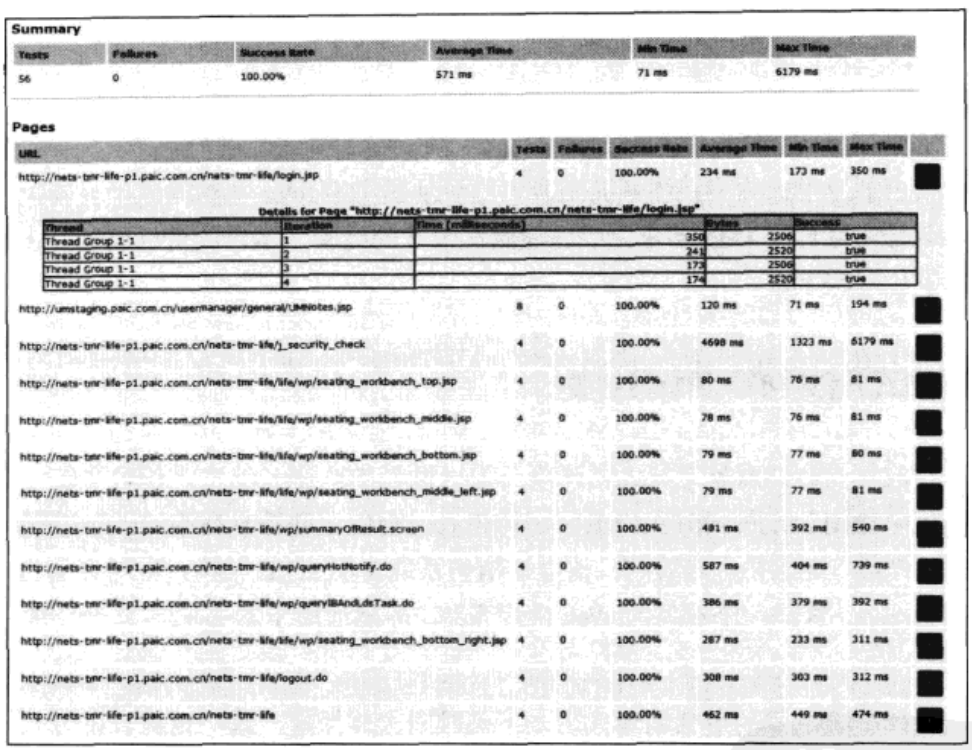
```
ant -Dtest=测试案例名称          ；运行测试案例并生成报表
ant -Dtest=测试案例名称 run     ；只运行测试案例
ant -Dtest=测试案例名称 report；只生成报表（需要把测试采样数据.jtl 文件放到 ant 目录下）
```

接下来，让我们以登录功能（某大型保险公司电话销售系统）的 JMeter 测试计划为例：

```
D:\jmeter\jakarta-jmeter-2.4\extras>ant -Dtest=NETS-TMR-LIFE_pir
Buildfile: build.xml
run:
     [echo] funcMode = false
   [jmeter] Executing test plan:
   D:\jmeter\jakarta-jmeter-2.4\extras\NETS-TMR-LI
FE_pir.jmx ==> D:\jmeter\jakarta-jmeter-2.4\extras\NETS-TMR-LIFE_pir.jtl
   [jmeter] Created the tree successfully using
D:\jmeter\jakarta-jmeter-2.4\ext
ras\NETS-TMR-LIFE_pir.jmx
   [jmeter] Starting the test @ Thu Mar 03 19:47:23 CST 2011 (1299152843473)
   [jmeter] Waiting for possible shutdown message on port 4445
   [jmeter] Tidying up ...    @ Thu Mar 03 19:47:32 CST 2011 (1299152852131)
   [jmeter] ... end of run
_message_xalan:
report:
     [xslt] Processing
```

```
    D:\jmeter\jakarta-jmeter-2.4\extras\NETS-TMR-LIFE_pir.jtl
 to D:\jmeter\jakarta-jmeter-2.4\extras\NETS-TMR-LIFE_pir.html
    [xslt] Loading stylesheet
    D:\jmeter\jakarta-jmeter-2.4\extras\jmeter-result
s-detail-report_21.xsl
all:
BUILD SUCCESSFUL
Total time: 11 seconds
```

下面是运行生成后的报表，如图 12-34 所示。



Summary

Tests	Failures	Success Rate	Average Time	Min Time	Max Time
56	0	100.00%	571 ms	71 ms	6179 ms

Pages

URL	Tests	Failures	Success Rate	Average Time	Min Time	Max Time
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/login.jsp	4	0	100.00%	234 ms	173 ms	350 ms
http://umstaging.paic.com.cn/usermanager/general/UMNotes.jsp	8	0	100.00%	120 ms	71 ms	194 ms
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/j_security_check	4	0	100.00%	4698 ms	1323 ms	6179 ms
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/life/wp/seating_workbench_top.jsp	4	0	100.00%	80 ms	76 ms	81 ms
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/life/wp/seating_workbench_middle.jsp	4	0	100.00%	78 ms	76 ms	81 ms
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/life/wp/seating_workbench_bottom.jsp	4	0	100.00%	79 ms	77 ms	80 ms
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/life/wp/seating_workbench_middle_left.jsp	4	0	100.00%	79 ms	77 ms	81 ms
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/wp/summaryOfResult.screen	4	0	100.00%	491 ms	392 ms	540 ms
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/wp/queryHotNotify.do	4	0	100.00%	587 ms	404 ms	739 ms
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/wp/queryIBAndLdsTask.do	4	0	100.00%	386 ms	379 ms	392 ms
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/life/wp/seating_workbench_bottom_right.jsp	4	0	100.00%	287 ms	233 ms	311 ms
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/logout.do	4	0	100.00%	308 ms	303 ms	312 ms
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life	4	0	100.00%	462 ms	449 ms	474 ms

Details for Page "http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/login.jsp"

Thread	Iteration	Time (milliseconds)	Bytes	Success
Thread Group 1-1	1	350	2506	true
Thread Group 1-1	2	241	2520	true
Thread Group 1-1	3	173	2506	true
Thread Group 1-1	4	174	2520	true

图 12-34　借助 Ant 生成的 JMeter 测试结果报表

12.4　本章小结

本章的内容非常重要，各位读者朋友应该仔细阅读，以便掌握分析 JMeter 性能测试结果的方法。本章首先介绍了性能测试分析模型，让读者朋友有整体上的把握。接着作者详细介绍了 JMeter 目前支持的各种监听器，它们是 JMeter 性能测试结果分析的基石。在熟悉各种监听器的使用方法之后，作者介绍了如何在实际工作（Web 性能测试）中，使用常用监听器来发现性能缺陷。最后介绍了如何借助 Ant 来实现批量测试和报表生成。

第 13 章

JMeter性能测试实战——电话销售系统

通过学习前面章节的内容，读者朋友应该已经掌握了 JMeter 性能测试工具的使用方法。目前所欠缺的就是实际使用 JMeter 来完成性能测试工作的经验。本章的目标就是补上这一短板，帮助读者朋友们尽快上手。

13.1 测试背景和测试目标

本章介绍的性能测试对象是某大型保险公司（世界 500 强）的电话销售系统。该电话销售系统支持数万电话销售人员同时在线工作，销售人员会电话联系目标客户并推销保险产品，其中包括车险、产险、养老险、寿险等。该电话销售系统使用 J2EE 架构，中间件层为 WebLogic，数据库采用 Oracle。这是一个典型的 Web 应用系统，显然适用于本书 12.1 节介绍的性能测试分析模型。

测试背景：保险公司的电话销售人员上下班时间大体保持一致，也就是说每天上班时，销售人员会在很短一段时间内集体登录电话销售系统。这种风暴式的登录会对系统带来很大的压力，严重时会导致部分用户无法及时登录系统。而且随着公司电话销售业绩的快速提升，同时在线的销售人员也会大幅增长，也就是说登录风暴只会越来越严重。当问题累积到一定程度全面爆发时，业务部门的投诉绝对会让 IT 部门“吃不了兜着走”。

测试目标：协助开发人员分析登录系统的流程中，哪一步骤存在瓶颈，并给予改进建议。

13.2 分析确定性能测试指标

如表 13-1 所示为 IT 部门对业务部门承诺的关键功能（登录系统）的性能指标。如何将宽泛的 IT 承诺指标转换为测试人员可以使用的性能测试指标，这里有一定的技巧。作者所在的测

试部门总结出了一套转换模型，在这里简要介绍一下。它并不一定适用于测试人员所在的测试组织，但存在一定的通用性。

表 13-1　关键功能的性能指标

关键功能	平均使用次数	平均用户数	高峰段用户数	平均响应时长	可接受最长响应时长	使用时间段
登录（身份验证）		20000	23000	<4 s	≤7 s	8:00～21:00

1．如何确定性能测试的并发虚拟用户数

测试环境平均并发数=（高峰段用户数×10%）/n

n 是生产环境和测试环境服务器配置折算比，例如 n=公倍数((生产 Web 服务器数/测试 Web 服务器数), (生产 APP 服务器数/测试 APP 服务器数)) ×(生产服务器内存/测试服务器内存)，一般算下来 $n=4$。请注意 $n=4$ 仅仅是一个经验值，并不一定适用于测试人员所在的测试组织。

10%的含义是我们假定所有用户中，只有 10%的用户在同一时刻做同一件事情，例如登录系统。请注意 10%仅仅是一个经验值，并不一定适用于测试人员所在的测试组织。

2．如何确定性能测试的持续时长

IT 部门对业务部门承诺的系统可用时间段为：8:00～！21:00，那么用户的登录操作是否平均分布在 8:00～21:00 这 13 个小时以内呢？答案当然是否定的。作者调研生产用户习惯后，发现登录操作集中发生在 8:50～9:20 这半个小时以内。因此我们可以将性能测试的持续时长，确定为半个小时。

3．如何确定性能测试的存量数据

准备性能测试的存量数据是一件复杂而具有挑战性的工作。这项工作并没有通用的模型可供参考，只能凭借测试人员的经验，它需要测试人员对系统架构非常熟悉才行。本例中的存量数据就是 23000 个测试账号，即数据库用户表中应该有 23000 条以上的记录，而测试人员并不需要保证这 23000 条记录每一条都能真实完成系统登录。

4．如何确定测试人员重点观察的性能指标

从 IT 部门对业务部门承诺的关键功能（登录系统）的性能指标中，我们可以找到两个具体要求：

（1）平均响应时长<4 s。

（2）可接受最长响应时长≤7 s。

现在要做的就是将这两个指标与 JMeter 监听器的统计口径关联起来。

（1）平均响应时长←→平均响应时长（JMeter 聚合报告）。

（2）可接受最长响应时长←→90%阈值（JMeter 聚合报告）。

除了平均响应时长和 90%阈值这两项统计值之外，我们还需要关注哪几项统计值呢?答案是吞吐率（每秒完成请求数）和错误率。在图形结果中吞吐率应该是先逐渐攀升最后趋于平稳的一条曲线，而错误率则应该基本为 0。

于是我们得出结论，本例中测试人员应该重点关注 4 项 JMeter 监听器的统计指标：

（1）平均响应时长（JMeter 聚合报告）<4 s。

（2）90 %阈值（JMeter 聚合报告）≤7 s。

（3）吞吐率，应该是先逐渐攀升最后趋于平稳的一条曲线。

（4）错误率，趋近于 0，如果不为 0，应该认真分析。

13.3 录制创建性能测试脚本

让我们采用本书 3.5.2 节介绍的方法，用 Badboy 工具录制/导出 JMeter 测试脚本。我们得到的测试脚本如图 13-1 所示。

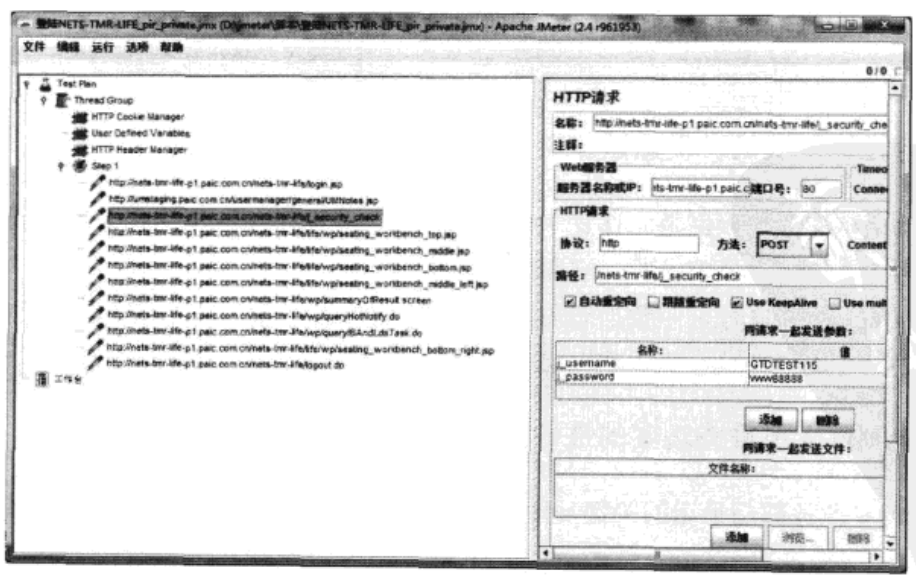


图 13-1 使用 Badboy 工具录制/导出 JMeter 测试脚本

从图 13-1 中我们可以看出，Badboy 工具录制/导出的 JMeter 测试脚本中包含登录操作涉及的所有 Web 页面。另外还添加了 3 个配置元件：HTTP Cookie Manager、User Defined Variables、HTTP Header Manager。此时，创建性能测试脚本的工作是否已经结束了？答案是没有，我们还需要添加和修改一些测试元件。

（1）添加 CSV Data Set Config 配置元件，用于读取关键测试数据。本例中的关键测试数据就是登录系统所用到的账户/密码。我们添加一个 CSV Data Set Config 配置元件，指定读取的 CSV 数据文件，将账户/密码信息存放在 CSV 数据文件中。账户/密码组合应该尽可能地多提供，这样才能真实地模拟用户登录操作。

（2）修改用户校验页面对应的采样器，让它使用来自于 CSV 数据文件的账户/密码信息。

（3）添加事务控制器“登录”和“退出”，以便衡量整个登录流程所需的时长。

（4）添加监听器“聚合报告”，以便统计和分析性能测试结果。

修改后的 JMeter 测试脚本如图 13-2 所示。

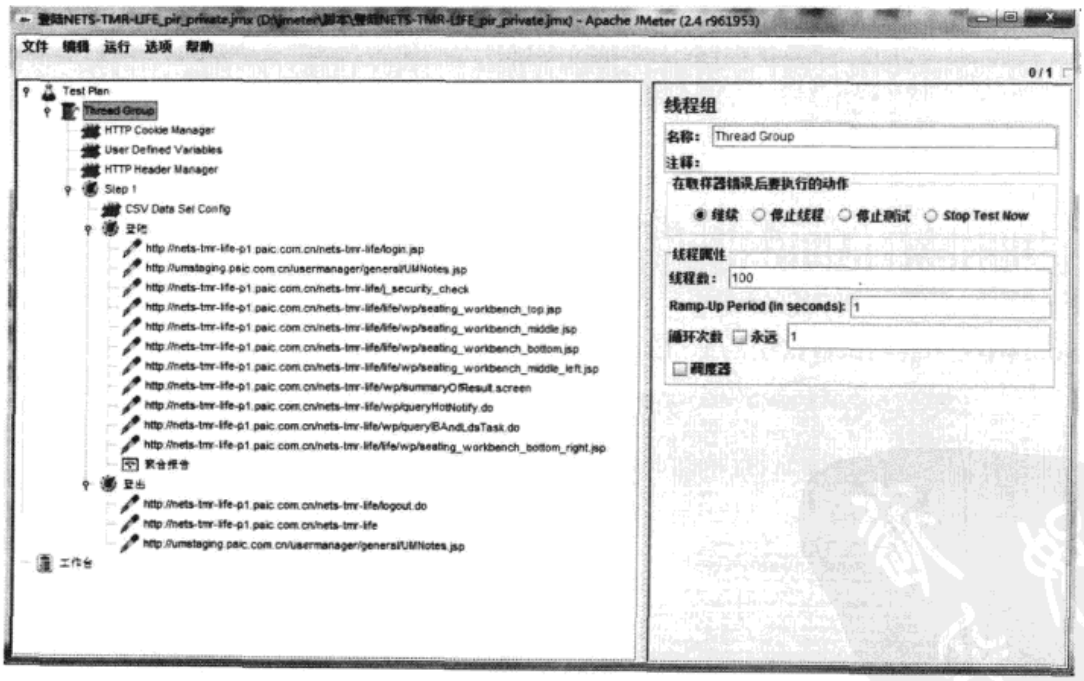


图 13-2　使用 badboy 工具录制/导出 JMeter 测试脚本

下面让我们首先看看如何设置 CSV Data Set Config，如图 13-3 所示。在 Filename 域中填写 CSV 文件的路径；在 Variable Names 域中填写参数组合，其中参数的排序要与数据文件中的数据排序保持一致；在 Delimiter 域中填写数据分隔符；其余参数的含义请参见本书 10.1 节的内容。

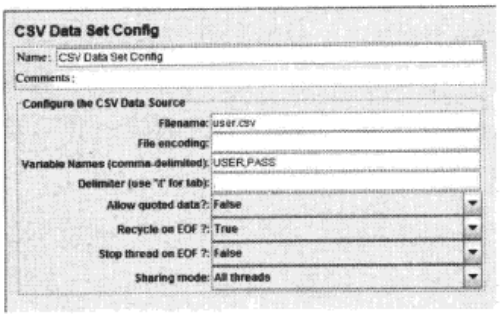


图 13-3　设置 CSV Data Set Config

如此设置 CSV Data Set Config 的目标，就是读取数据文件 user.csv 中的数据，提取登录用户名和对应的密码，分别放在 USER 和 PASS 两个参数之中。数据文件 user.csv 中存放的数据格式如图 13-4 所示。

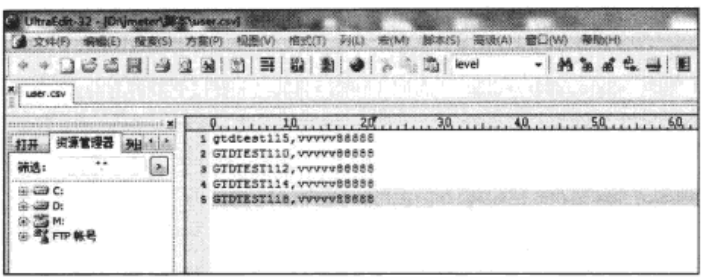


图 13-4　数据文件 user.csv 中存放的数据格式

接下来，让我们看看如何修改用户校验页面对应的采样器，如图 13-5 所示。j_username 和 j_password 两项关键测试数据引用 USER 和 PASS 两个参数的值。

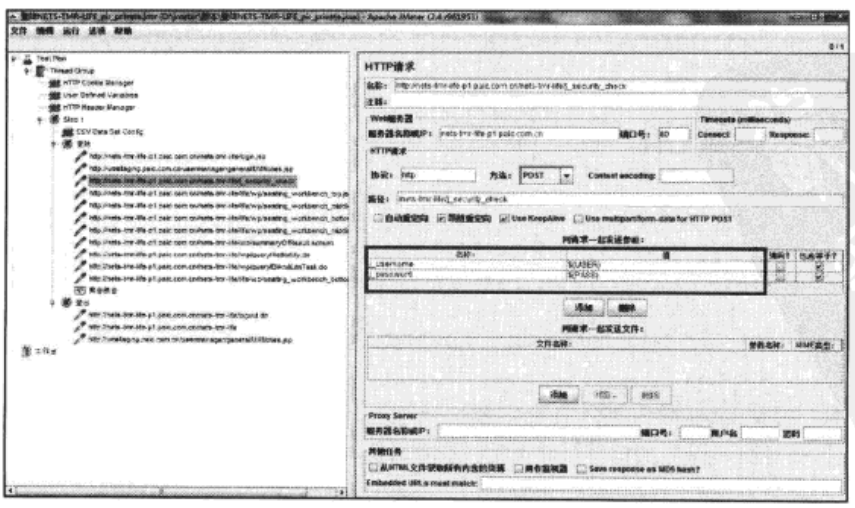


图 13-5　修改用户校验页面对应的采样器

在添加“登录”、“退出”两个事务控制器及“聚合报告”监听器后，还有一项关键的工作，那就是修改线程组（Thread Group）的各项参数，如图 13-6 所示。

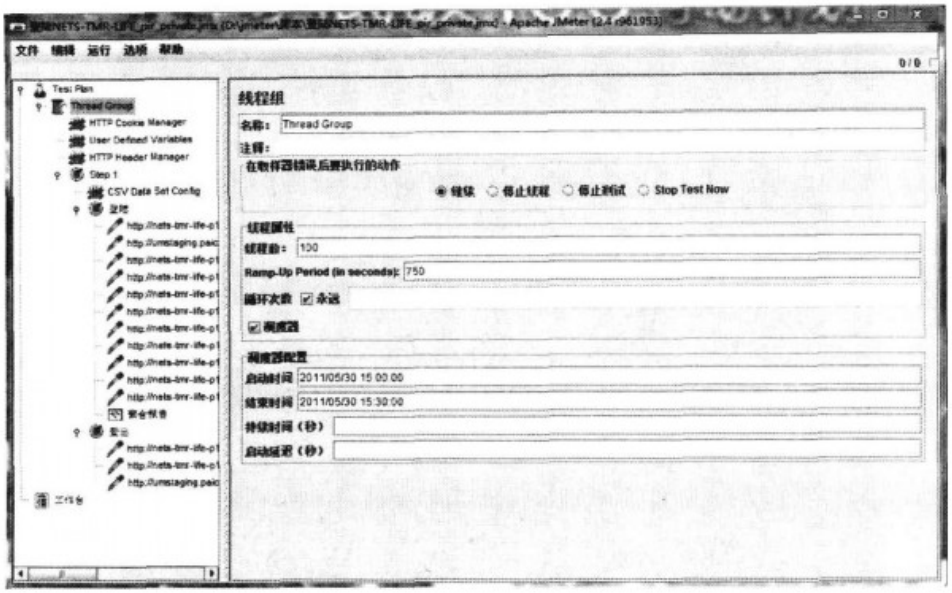


图 13-6 修改线程组的各项参数

其中线程数由 13.2 节介绍的计算公式得出（此例中 n 取 23，原因在于测试环境的 Web 和应用服务器并不是独享的，需要与其他应用系统共用）。Ramp-Up Period 参数设置为 750 秒，即在 750 秒内启动所有并发线程。这里应该选中调度器，并设置测试的启动和结束时间，设置总的测试时长为半个小时。

13.4　运行性能测试脚本

认真阅读过本书前面章节的读者朋友，应该清楚地知道 JMeter 有多种运行方式，其中包括 GUI 模式、非 GUI 模式、远程模式。GUI 模式适用于并发线程数较少，且需要实时观察监听器统计数据变化的场景；非 GUI 模式适用于并发线程数较多，无须实时观察监听器统计数据变化的场景；远程模式适用于并发线程数很多，单台 JMeter 执行机无法支持的场景。

本例中的并发线程数达到了 100 个，采用 GUI 模式对 JMeter 执行机的性能要求较高，一般情况下我们选择非 GUI 模式来运行 JMeter，如果还不能满足需求，可以使用远程模式。在测试期间我们应该观察 JMeter 执行机的 CPU 和内存占用情况，甚至于本地网络内的负载情况，以免影响测试结论。

采用 GUI 模式时，JMeter 在压力较大的情况下，可能会报告“Out of heap”错误。

现在让我们看看本例中如何使用非 GUI 模式运行 JMeter。首先在 DOS 下访问 JMeter 的 bin 目录，如图 13-7 所示。我们使用 dir 命令查询该目录下的文件，检查是否存在 jmeter–n.cmd 文件。

图 13-7　修改线程组的各项参数

如果该目录下存在 jmeter–n.cmd 文件，接着我们就使用 jmeter-n 命令来实现非 GUI 模式下运行 JMeter 测试脚本，如图 13-8 所示。

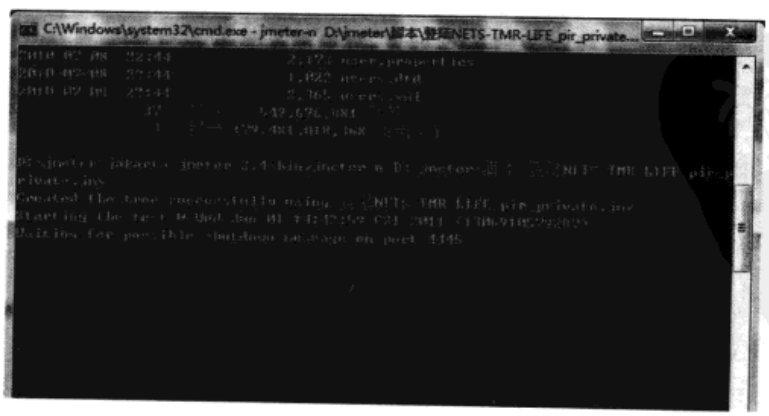


图 13-8　使用 jmeter-n 命令实现非 GUI 模式下运行 JMeter 测试脚本

在测试运行期间，我们可以随时使用 shhtdown 命令来终止测试，如图 13-9 所示。

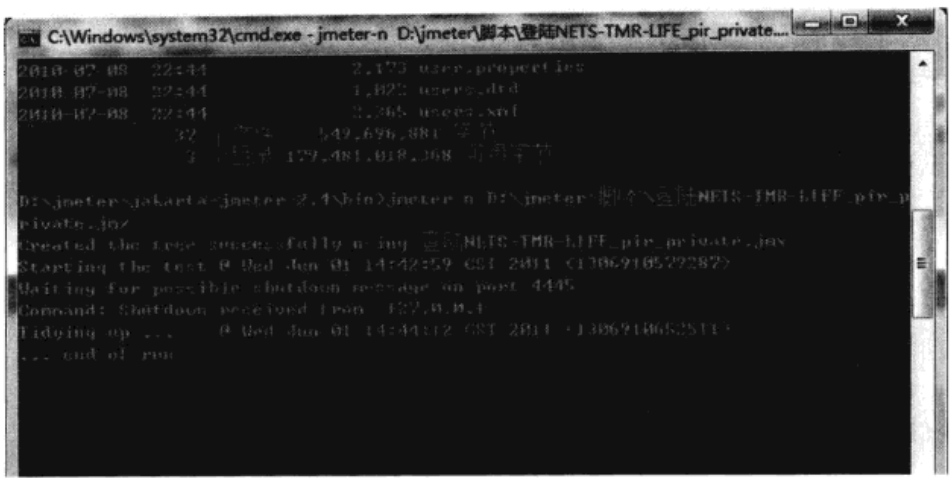


图 13-9　测试运行期间，使用 shutdown 命令终止测试

JMeter 测试脚本运行完毕后，会产生两个重要结果文件：xxx.jtl 和 xxx.log，如图 13-10 所示。其中 xxx.jtl 存放测试记录的结果数据，xxx.log 存放测试期间记录的日志文件。

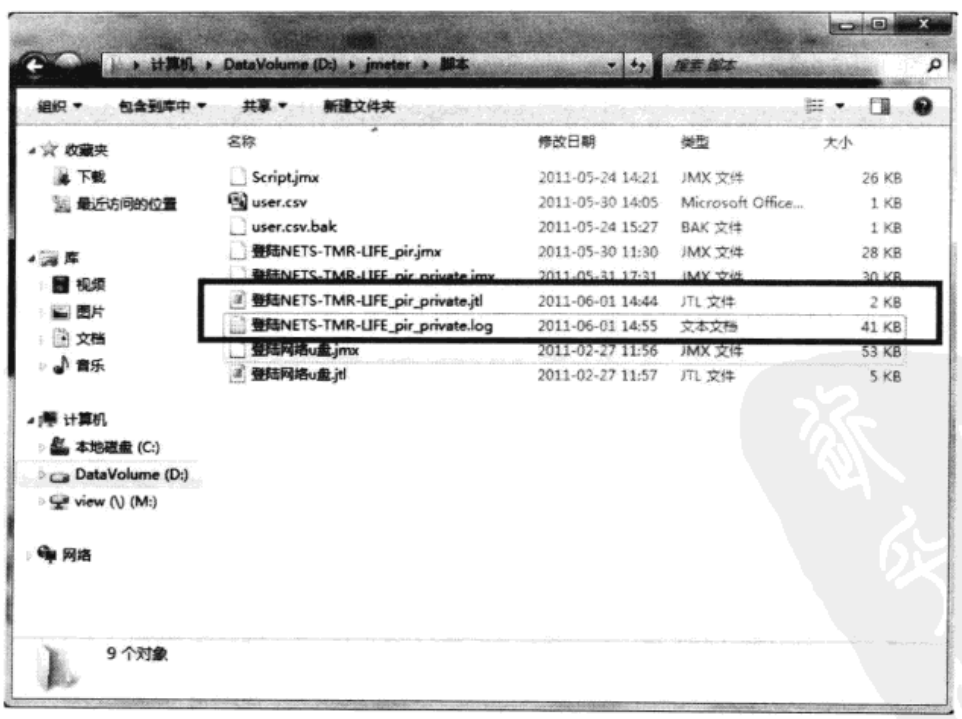


图 13-10　运行 JMeter 测试脚本产生的两个重要结果文件

13.5 分析性能测试结果

性能测试脚本执行完毕后，接下来要做的就是对性能测试结果进行分析。

（1）确认性能测试脚本是否已经执行完毕，如图 13-11 所示。从图中可以看出测试启动于 14:02:49，结束于 14:35:50，整个测试时长略长于 30 分钟，符合预期。

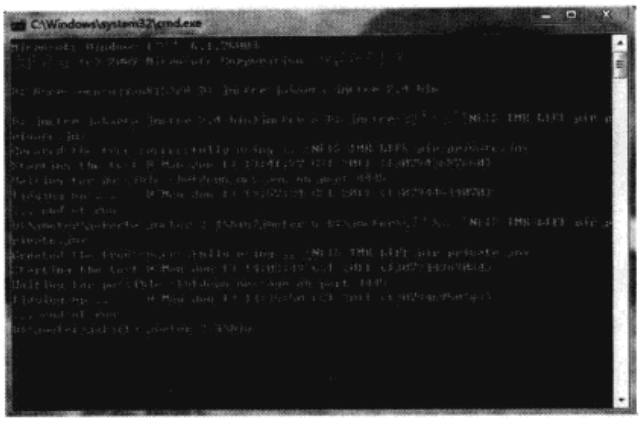
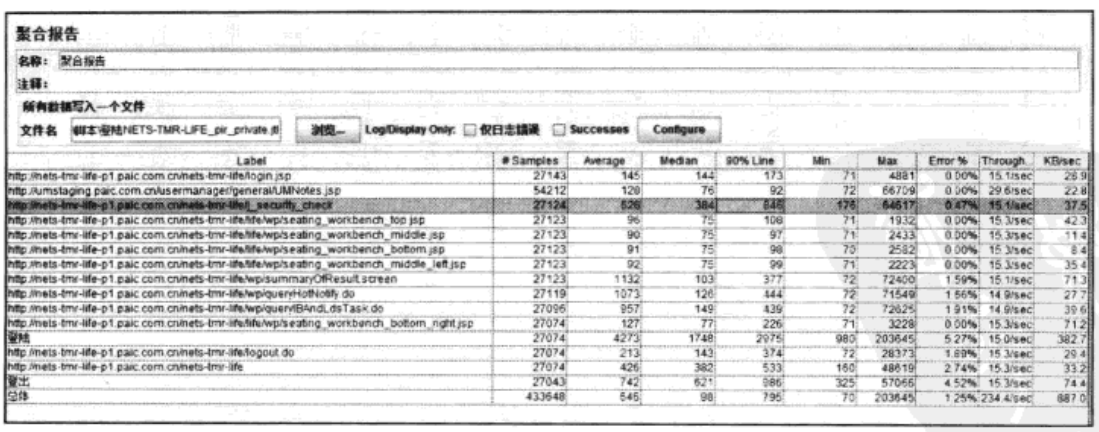


图 13-11 确认性能测试脚本是否已经执行完毕

（2）以 GUI 模式启动 JMeter，并使用监听器“聚合报告”来分析 JMeter 收集的性能测试数据，如图 13-12 所示。


聚合报告

名称：聚合报告

注释：

所有数据写入一个文件

文件名 ...\脚本\登陆\NETS-TMR-LIFE_pir_private.jtl 浏览... Log/Display Only: 仅日志错误 Successes Configure

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Through	KB/sec
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/login.jsp	27143	145	144	173	71	4881	0.00%	15.1/sec	28.9
http://umstaging.paic.com.cn/usermanager/general/UMNotes.jsp	54212	128	76	92	72	66709	0.00%	29.6/sec	22.8
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/j_security_check	27124	628	384	848	176	64617	0.47%	15.1/sec	37.5
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/life/wp/seating_workbench_top.jsp	27123	96	75	108	71	1932	0.00%	15.3/sec	42.3
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/life/wp/seating_workbench_middle.jsp	27123	90	75	97	71	2433	0.00%	15.3/sec	11.4
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/life/wp/seating_workbench_bottom.jsp	27123	91	75	98	73	2582	0.00%	15.3/sec	8.4
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/life/wp/seating_workbench_middle_left.jsp	27123	92	75	99	71	2223	0.00%	15.3/sec	35.4
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/wp/summaryOfResult.screen	27123	1132	103	377	72	72400	1.59%	15.1/sec	71.3
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/wp/queryHotNotify.do	27119	1073	126	444	72	71549	1.56%	14.9/sec	27.7
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/wp/queryIBAndLdsTask.do	27096	957	149	439	72	72625	1.91%	14.9/sec	39.6
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/life/wp/seating_workbench_bottom_right.jsp	27074	127	77	226	71	3228	0.00%	15.3/sec	71.2
登陆	27074	4273	1748	2975	980	203645	5.27%	15.0/sec	382.7
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life/logout.do	27074	213	143	374	72	28373	1.89%	15.3/sec	29.4
http://nets-tmr-life-p1.paic.com.cn/nets-tmr-life	27074	426	382	533	160	48619	2.74%	15.3/sec	33.2
登出	27043	742	621	986	325	57066	4.52%	15.3/sec	74.4
总体	433648	545	98	796	70	203645	1.25%	234.4/sec	887.0

图 13-12 使用监听器“聚合报告”来分析性能测试数据

在图 13-12 中，我们需要关心几个重要数据：平均响应时长（Average）、90 %阈值（90% Line）、错误率（Error%）。下面分别进行介绍：①登录的平均响应时长为 4273 毫秒（约 4.3 秒），超过

了对用户承诺的 4 秒时长，但超出幅度不大，也就是说对此需要加以改善。

②登录的 90%阈值为 2975 毫秒（约 3 秒），远远小于对用户承诺的可接受最长响应时长 7 秒，是符合预期的。不过需要注意的是，JMeter 统计的最大响应时长（Max）高达 20 秒，远远超过了对用户承诺的 7 秒。总结一下，系统对 90%的用户响应很快，但是对个别用户存在系统缓慢的情况，对此需要提醒开发加以注意。

③登录的错误率高达 5.27%需要引起我们的高度关注，其中 j_security_check、summaryOfResult.screen、queryHotNotify.do 和 queryIBAndLdsTask.do 的错误率分别为 0.47%、1.59%、1.66%和 1.91%。另外我们在图中还可以发现一个规律，也就是说错误集中在平均响应时间最长的几个 URL 上。由此我们可以初步怀疑系统在压力较大的情况下，无法及时处理某些请求。但这只是初步怀疑，还需要其他证据来加以佐证。现在让我们看看日志文件：登录 NETS-TMR-LIFE_pir_private.log，如图 13-13 所示。从图中我们不难看出存在大量的 HTTP 500 错误，导致 HTTP 500 内部服务器错误的原因很多，但是这里我们几乎可以断定是性能缺陷导致的。原因在于笔者将并发用户数设为 10 后，再次运行该性能测试脚本，发现日志文件中没有记录 HTTP 500 错误。另外笔者对比 Web 服务器日志，发现其中的记录可以支持这一结论。

由于担心泄露公司商业秘密，笔者隐去了 Web 服务器的日志。这里介绍一个小技巧，即分析性能测试数据时，测试人员需要综合参考 JMeter 运行日志和待测系统服务器的日志记录。

```
709 2011/06/13 14:12:28 ERROR - jmeter.protocol.http.sampler.HTTPSampler: Cause: java.io.IOException: Server returned HTTP response coc
704 2011/06/13 14:12:28 INFO  - jmeter.protocol.http.sampler.HTTPSampler: Error Response Code: 500
705 2011/06/13 14:12:28 ERROR - jmeter.protocol.http.sampler.HTTPSampler: readResponse: java.io.IOException: Server returned HTTP respc
706 2011/06/13 14:12:28 ERROR - jmeter.protocol.http.sampler.HTTPSampler: Cause: java.io.IOException: Server returned HTTP response coc
707 2011/06/13 14:12:28 INFO  - jmeter.protocol.http.sampler.HTTPSampler: Error Response Code: 500
708 2011/06/13 14:12:30 INFO  - jmeter.threads.JMeterThread: Thread started: Thread Group 1-61
709 2011/06/13 14:12:30 ERROR - jmeter.protocol.http.sampler.HTTPSampler: readResponse: java.io.IOException: Server returned HTTP respc
710 2011/06/13 14:12:30 ERROR - jmeter.protocol.http.sampler.HTTPSampler: Cause: java.io.IOException: Server returned HTTP response coc
711 2011/06/13 14:12:30 INFO  - jmeter.protocol.http.sampler.HTTPSampler: Error Response Code: 500
712 2011/06/13 14:12:30 ERROR - jmeter.protocol.http.sampler.HTTPSampler: readResponse: java.io.IOException: Server returned HTTP respc
713 2011/06/13 14:12:30 ERROR - jmeter.protocol.http.sampler.HTTPSampler: Cause: java.io.IOException: Server returned HTTP response coc
714 2011/06/13 14:12:30 INFO  - jmeter.protocol.http.sampler.HTTPSampler: Error Response Code: 500
715 2011/06/13 14:12:31 ERROR - jmeter.protocol.http.sampler.HTTPSampler: readResponse: java.io.IOException: Server returned HTTP respc
716 2011/06/13 14:12:31 ERROR - jmeter.protocol.http.sampler.HTTPSampler: Cause: java.io.IOException: Server returned HTTP response coc
717 2011/06/13 14:12:31 INFO  - jmeter.protocol.http.sampler.HTTPSampler: Error Response Code: 500
718 2011/06/13 14:12:31 ERROR - jmeter.protocol.http.sampler.HTTPSampler: readResponse: java.io.IOException: Server returned HTTP respc
719 2011/06/13 14:12:31 ERROR - jmeter.protocol.http.sampler.HTTPSampler: Cause: java.io.IOException: Server returned HTTP response coc
720 2011/06/13 14:12:31 INFO  - jmeter.protocol.http.sampler.HTTPSampler: Error Response Code: 500
721 2011/06/13 14:12:31 ERROR - jmeter.protocol.http.sampler.HTTPSampler: readResponse: java.io.IOException: Server returned HTTP respc
722 2011/06/13 14:12:31 ERROR - jmeter.protocol.http.sampler.HTTPSampler: Cause: java.io.IOException: Server returned HTTP response coc
723 2011/06/13 14:12:31 INFO  - jmeter.protocol.http.sampler.HTTPSampler: Error Response Code: 500
724 2011/06/13 14:12:31 ERROR - jmeter.protocol.http.sampler.HTTPSampler: readResponse: java.io.IOException: Server returned HTTP respc
725 2011/06/13 14:12:31 ERROR - jmeter.protocol.http.sampler.HTTPSampler: Cause: java.io.IOException: Server returned HTTP response coc
726 2011/06/13 14:12:31 INFO  - jmeter.protocol.http.sampler.HTTPSampler: Error Response Code: 500
727 2011/06/13 14:12:32 ERROR - jmeter.protocol.http.sampler.HTTPSampler: readResponse: java.io.IOException: Server returned HTTP respc
728 2011/06/13 14:12:32 ERROR - jmeter.protocol.http.sampler.HTTPSampler: Cause: java.io.IOException: Server returned HTTP response coc
729 2011/06/13 14:12:32 INFO  - jmeter.protocol.http.sampler.HTTPSampler: Error Response Code: 500
730 2011/06/13 14:12:32 ERROR - jmeter.protocol.http.sampler.HTTPSampler: readResponse: java.io.IOException: Server returned HTTP respc
731 2011/06/13 14:12:32 ERROR - jmeter.protocol.http.sampler.HTTPSampler: Cause: java.io.IOException: Server returned HTTP response coc
732 2011/06/13 14:12:32 INFO  - jmeter.protocol.http.sampler.HTTPSampler: Error Response Code: 500
733 2011/06/13 14:12:33 ERROR - jmeter.protocol.http.sampler.HTTPSampler: readResponse: java.io.IOException: Server returned HTTP respc
734 2011/06/13 14:12:33 ERROR - jmeter.protocol.http.sampler.HTTPSampler: Cause: java.io.IOException: Server returned HTTP response coc
735 2011/06/13 14:12:33 INFO  - jmeter.protocol.http.sampler.HTTPSampler: Error Response Code: 500
736 2011/06/13 14:12:34 ERROR - jmeter.protocol.http.sampler.HTTPSampler: readResponse: java.io.IOException: Server returned HTTP respc
737 2011/06/13 14:12:34 ERROR - jmeter.protocol.http.sampler.HTTPSampler: Cause: java.io.IOException: Server returned HTTP response coc
```

图 13-13　日志文件数据

（3）查看应用系统对应数据库的历史统计信息，发现数据库并未承受太大压力，在前 10 条 Top Sql 中未发现与登录功能相关的语句。这就证明系统瓶颈并不在数据库上，可以缩减定

位问题的范围。

由于担心会泄露公司商业秘密，在本例中未展示实际数据库监控信息，读者朋友可以参考图 12-3。

（4）查看应用系统的 Web 服务器、APP 服务器日志和监控记录，我们可以发现 Web 服务器日志中存在较多错误记录。作为测试人员，应该将这些错误记录作为软件缺陷的一部分提交开发人员进行分析。

由于担心泄露公司商业秘密，笔者这里隐去了 Web 服务器的日志。

13.6 上报性能测试缺陷

在此例中作为测试人员，我们发现应用系统存在有明显的性能缺陷，需要上报性能缺陷。在这里我们简单总结一下应该如何描述缺陷。

缺陷名称：

“登录”系统在高并发压力情况下，无法及时响应用户请求。

缺陷内容：

登录的平均响应时长为 4273 毫秒（约 4.3 秒），超过了对用户承诺的 4 秒时长，但超出幅度不大，需要对此加以改善；系统对 90%的用户响应很快，但是对个别用户存在系统响应缓慢的情况，对此需要提醒开发同事加以注意。

登录的错误率高达 5.27%需要引起高度关注，其中 j_security_check、summaryOfResult.screen、queryHotNotify.do 和 queryIBAndLdsTask.do 的错误率分别为 0.47%、1.59%、1.66%T 1.91%。Web 服务器日志及 JMeter 运行日志均提示，系统在高并发压力情况下，无法及时响应用户请求（存在大量 HTTP 500 错误）。具体日志信息请见缺陷附件。

通过 APM 检查数据库历史统计信息，发现数据库并未承受太大压力，在前 10 条 Top Sql 中未发现与登录功能相关的语句。测试人员推断性能瓶颈不在数据库。另外，测试方建议重点分析 j_security_check、summaryOfResult.screen、queryHotNotify.do 和 queryIBAndLdsTask.do。

缺陷附件：

- Web 服务器日志.doc。

- APP 服务器日志.doc。
- JMeter 运行日志.doc。
- APM 数据库监控信息.doc。

笔者认为如此上报性能缺陷，可以极大地方便开发人员定位问题。即便是最挑剔的开发人员，也会对测试人员的专业技能表示钦佩。

13.7　本章小结

本章以某大型保险公司电话销售系统的登录功能为例，详细演示了一个性能测试应该包含的所有步骤。首先是如何分析性能测试指标，接着是录制/创建性能测试脚本，并在特定测试场景下执行性能测试脚本；最关键的是如何使用 JMeter 监听器来分析性能测试数据；最后作者展示了如何准确地上报一个性能测试缺陷。

反侵权盗版声明

举报电话：（010）88254396；（010）88258888
传　　真：（010）88254397
E-mail：dbqq@phei.com.cn
通信地址：北京市万寿路 173 信箱
　　电子工业出版社总编办公室
邮　　编：100036

[General Information]
书名=零成本实现WEB性能测试　基于APACHE JMETER
作者=温素剑编著
页数=344
出版社=北京市：电子工业出版社
出版日期=2012.02
SS号=12939325
DX号=000008232582
URL=http://book2.duxiu.com/bookDetail.jsp?dxNumber=000008232582&d=A27545D02F42B992B3553311B4F82FB8