使用JMeter进行负载测试——终极指南

2014/11/26 | 分类： [WEB开发](http://www.importnew.com/cat/web-development), [教程](http://www.importnew.com/cat/tutorial) | [2 条评论](http://www.importnew.com/13876.html#comments) | 标签： [JMETER](http://www.importnew.com/tag/jmeter), [教程](http://www.importnew.com/tag/%e6%95%99%e7%a8%8b)

分享到：**30**

本文由 [ImportNew](http://www.importnew.com/) - [张涛](http://www.importnew.com/author/zhangtao) 翻译自 [JavaCodeGeeks](http://www.javacodegeeks.com/2014/11/jmeter-tutorial-load-testing.html)。欢迎加入[翻译小组](http://group.jobbole.com/category/feedback/trans-team/)。转载请见文末要求。

这篇教程讨论的是JMeter，它是一款基于Java的、集合了几个应用程序、具有特定用途的负载和性能测试工具。

本篇主要涉及的内容:

* 解释一下JMeter的用途
* JMeter的实现方式以及采用的技术
* 安装与配置
* 介绍用户界面的基本特征
* 详细介绍不同的测试计划的配置方法
* 如何分析结果数据
* 涉及几个重要的最佳实践

所有的例子都是基于Java 8 update 0.20，[Eclipse](http://res.importnew.com/eclipse) Luna 4.4, JMeter 2.11, MySQL Community Server 5.6.21的开发环境。那我们就开始吧！

目录

[引言](http://www.importnew.com/13876.html#introduction)  
[采用的技术](http://www.importnew.com/13876.html#technologies)  
[教程内容简介](http://www.importnew.com/13876.html#about)  
[安装](http://www.importnew.com/13876.html#installation)  
[用户界面](http://www.importnew.com/13876.html#ui)  
[可用的请求](http://www.importnew.com/13876.html#requests)  
[测试计划和测试组件](http://www.importnew.com/13876.html#test_plan)  
[执行顺序](http://www.importnew.com/13876.html#order)  
[启动与停止](http://www.importnew.com/13876.html#run)  
[HTTP请求的测试计划](http://www.importnew.com/13876.html#http)  
[MySQL测试计划](http://www.importnew.com/13876.html#mysql)  
[JUnit Request测试计划](http://www.importnew.com/13876.html#junit)  
[记录测试](http://www.importnew.com/13876.html#recording)  
[插件](http://www.importnew.com/13876.html#plugins)  
[最佳实践](http://www.importnew.com/13876.html#best_practices)  
[总结](http://www.importnew.com/13876.html#summary)  
[下载](http://www.importnew.com/13876.html#download)  
[资源](http://www.importnew.com/13876.html#resources)

引言

[JMeter](http://jmeter.apache.org/)使用了不同技术和协议，是一款可以进行配置和执行负载测试、性能测试和压力测试的工具。

它能够模拟不同类型的请求，例如各种类型的数据库，FTP, HTTP, HTTPS 或者其他的服务端应用。

最好先来了解一下如下概念:

* **负载测试**: 这类测试使系统或者应用程序在预先设计好的极端场景下测试运行。这类测试用来评估系统或者程序在极端条件下的行为。
* **性能测试**: 这种测试被用来检测系统的性能表现，包括特定情况下，系统的响应能力和稳定性。
* **压力测试**: 这类测试通过载入更多的外部资源，并使系统组件超越其所设定的能力范围，试图使系统挂掉。

JMeter最初的用途是用来测试web应用的，但是它的功能已得到扩展，如今已经可以针对绝大部分的程序进行测试，同时可以在你的程序中进行功能测试。

采用的技术

JMeter是一款Java桌面应用程序，它的用户界面采用Swing Java API实现。基于这两点，JMeter是一个跨平台工具，能够运行在任何安装了Java虚拟机的操作系统(Windows, Linux, Mac)的设备上。  
它的框架支持并发和多线程或者线程组的执行。这对于配置负载测试和压力测试非常有用。  
它是可扩展的，提供了大量的可用插件。  
JMeter是Apache软件基金会下的一个子项目，是完全免费和开源的(<http://www.apache.org/licenses/>)。

教程内容简介

这篇文章是针对那些没有JMeter使用经验的开发人员和软件设计人员，介绍如何使用JMeter的教程。

同时针对可能不同的配置和测试计划，提供了一些例子和使用说明。在下一个章节中，我们将会看到如何安装JMeter，并配置、运行它，如何保存和分析不同测试计划的结果。

安装

为了JMeter在你的电脑上安装并运行，你需要按照如下步骤进行操作:

* 首先必须安装java环境。需要在你的电脑上安装JRE 1.6或者更高版本。只要到Oracle官网下载并安装适合您的系统的安装包: <http://www.oracle.com/technetwork/java/javase/downloads/index.html>。
* 设置JAVA\_HOME环境变量，使其指向jre的安装目录。Windows用户而言:

|  |  |
| --- | --- |
| 1 | JAVA\_HOME=C:\Program Files\Java\jdk1.8.0\_20 |

注意设置正确你的Java版本。

* 将java编译路径添加到系统路径下。Windows用户而言，将*C:\Program Files\Java\jdk1.7.0\_25\bin*添加到系统变量Path尾部。

然后测试java环境在你的设备上是否安装成功，使用如下命令行:

|  |  |
| --- | --- |
| 1 | java -version |

应该输出类似如下信息:

|  |  |
| --- | --- |
| 1  2  3 | java version "1.8.0\_20"  Java(TM) SE Runtime Environment (build 1.8.0\_20-b26)  Java HotSpot(TM) 64-Bit Server VM (build 25.20-b23, mixed mode) |

* 从<http://jmeter.apache.org/download_jmeter.cgi>下载最新版本的JMeter，
* 将它解压到例如C:\JMeter下(Windows)。
* 文件夹的结构应该这样:

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8 | apache-jmeter-2.9  apache-jmeter-2.9bin  apache-jmeter-2.9docs  apache-jmeter-2.9extras  apache-jmeter-2.9lib  apache-jmeter-2.9libext  apache-jmeter-2.9libjunit  apache-jmeter-2.9printable\_docs |

* 然后你就可以进入到C:\jmeter\apache-jmeter-2.11\bin(对于Windows用户)，执行jmeter.bat
* 可以看到JMeter GUI的启动界面:

GUI概述

JMeter的GUI非常直观易用。有文件处理的通用菜单，例如：新建，打开，保存，另存为等。还有启动和停止测试计划，配置的可用的菜单项。每一个测试计划，线程组，测试计划节点(我们将会在下一章看到更多这方面的内容)都可以通过点击鼠标右键，提供不同的上下文菜单选项。

也可以改变用户界面的语言，有许多语言可供选择。

正如我们所说的，图形用户界面是非常直观和易用的，为每个菜单和动作提供了易用理解的工具提示。除此之外，暂且不谈它的质量，那为什么JMeter是如此广泛使用？因为它很容易理解，并且易学。

可用的请求

JMeter几乎提供任何一种系统的测试配置。总的来说，下列协议被包含在内：

* Web: HTTP, HTTPS网站 ‘web 1.0′ web 2.0 (ajax, flex 和 flex-ws-amf): 我们以后会看到如何测试一个潜在的网页及其行为
* Web 服务: SOAP / XML-RPC.
* 通过JDBC驱动的数据库。几乎支持任何一种数据库，你需要的就是一个合适的驱动程序包，并放到在JMeter安装目录的正确位置。我们后续将做一个MySQL数据库测试。
* 路径: LDAP.
* 基于JMS的面向消息的服务
* 使用POP3, IMAP, SMTP协议的邮件服务.
* FTP服务
* 使用JUnit和Java应用程序的进行的功能测试

测试计划与组件

基本上，使用JMeter可以进行如下操作:创建，配置，执行测试计划及结果分析。测试计划是对已配置的本地或远程服务器（或客户端）的请求集执行的具体说明。

在一个测试计划的配置中，你可以指定多个输入和输出参数，可以配置使测试计划成功执行或失败的基本条件。

JMeter测试计划是由不同的组件组成。这里有一个最重要组件的列表，指导你如何使用JMeter的用户界面添加它们以及这些插件的作用：

线程组（ThreadGroup）

一个线程组基本上是不同的测试计划元素的组合，它是一个测试计划的核心，它控制着基本核心参数。

为了创建一个测试计划，首先你不得不去创建一个线程组，配置如下参数:线程数量，过渡时期，循环次数和正常情况或者错误情况下的行为：

* **线程数**：执行测试计划的线程数，这个参数对于配置负载和压力测试非常重要。
* **过渡期**：JMeter开始启动所有线程所需时间。
* **循环次数**:即迭代次数，也就是测试计划被执行的次数
* **错误行为**：错误场景下的行为模式：阻止当前线程，停止整个测试，继续执行…

一个线程组也有配置选项的开始和结束时间。通过单击复选框“Scheduler”，弹出带有调度参数的面板，可以为你的测试配置开始和结束时间。

一旦配置完成，你就可以开始添加其他测试计划元素到线程组，例如采样器，侦听器和定时器。我们将在下一章解释这一切。

采样器（Samplers）

采样器用于发送请求到不同类型的服务器。它们是每一个测试计划的基本要素，一切都围绕这些采样器而工作：采样器执行请求（基于配置的请求），这些请求产生一个或多个响应，后续将被分析。

这里有一个在JMeter可用的采样器列表（在本教程我们将看到其中一些）：

* 访问日志采样器
* AJP采样器
* Bean shell取样器
* BSF采样器
* 调试采样器
* FTP采样器
* HTTP采样器
* Java采样器
* JDBC采样器
* JMS(几个)采样器
* JSR223采样器
* JUnit采样器
* LDAP(几个)采样器
* 邮件阅读器
* MongoDB采样器
* 操作系统进程取样器
* SMTP采样器
* SOAP
* TCP采样器
* 测试行动

在上面的列表中我们可以看到，有很多不同类型的采样器。这个清单是不完整的,因为存在多种其他不同于JMeter插件的实现技术。每个采样器的配置取决于它所执行的请求；这意味着一些采样器有一些共性的配置，而另外一些采样器由于它们各自请求的不同而完全不同。

逻辑控制器(Logic Controllers)

逻辑控制器允许你配置一个线程组内不同采样器的执行顺序。该列表包含了在JMeter所有可用的逻辑控制器：

* 简单控制器
* 循环控制器
* 一次性控制器
* 交错控制器
* 随机控制器
* 随机顺序控制器
* 流量控制器
* 运行时控制器
* I控制器
* While控制器
* Switch控制器
* ForEach控制器
* 模块控制器
* include控制器
* 事务控制器
* 记录控制器

监听器(Listeners)

监听器提供不同的方式查看由采样器请求产生的结果。监听器以报表、树型结构、或简明的日志文件的形式分析结果。

还可以在测试计划中的任何地方添加监听器，但他们只会在各自的应用范围内解析和收集来自采样器的数据。

这些都是JMeter可用的监听器：

* 样品结果配置保存
* 全图景结果集
* 图表结果集
* 样条线可视化工具
* 断言结果集
* 树形结果集
* 整合报告
* 表格结果集
* 简单数据输出
* 监测结果集
* 分布图(alpha)
* 整合图
* Mailer可视化工具
* Beanshell监听器
* 总结报告

定时器(Timers)

可以使用定时器来定义请求之间的等待时间。如果不指定，JMeter会一个请求完成后立即执行下一个请求，没有任何等待时间。

可在JMeter使用的计时器如下：

* 恒定的定时器
* 高斯随机定时器
* 均匀随机定时器
* 恒定吞吐量定时器
* 同步定时器
* jsr223定时器
* Beanshell定时器
* BSF定时器
* 泊松随机定时器

断言(Assertions)

断言通过验证采样器请求产生的响应，来验证测试计划的有效性。它基本上类似于单元测试断言，用来检测被测试应用程序的响应质量。你可以为每个测试计划配置任何生效的断言。

这里是一个在JMeter可以使用的断言列表：

* Bean shell断言
* BSF断言
* 比较断言
* jsr223断言
* 响应断言
* Duration断言
* XML Assertion
* BeanShell Assertion
* MD5Hex Assertion
* HTML Assertion
* XPath Assertion
* XML Schema Assertion

配置节点(Configuration nodes)

通过使用配置元素你可以将不同的参数传递给取样器请求。他们提供了创建变量（不同的和动态的）的一种方式，这些参数之后被采样器所使用。在采样器被执行前，参数所属节点启动时，这些参数被执行；这就是为什么采样器可以依赖这些变量。

这里是一个在JMeter使用的所有配置节点列表：

* 计数器
* CSV数据集配置
* FTP请求缺省值
* HTTP授权管理
* HTTP缓存管理
* HTTP cookie管理
* HTTP代理服务器
* HTTP请求缺省值
* HTTP头部管理
* Java请求缺省值
* 密钥库配置
* JDBC连接值
* 登录配置元素
* LDAP请求缺省值
* LDAP扩展请求缺省值
* TCP采样器配置
* 用户自定义变量
* 简单配置元素
* 随机变量

前置处理器(Pre processors)

在采样器执行前,前置处理器被触发。他们可用于从响应中提取变量，这些变量后续将通过配置元素被采样器所使用。

下面都是可以用来作为前置处理器的元素：

* HTML链接解析器
* HTTP URL重写修改器
* HTTP用户参数修改器
* 用户参数
* JDBC前置处理器
* jsr223前置处理器
* 正则表达式的用户参数
* Beanshell前置处理器
* BSF的前置处理器

后置处理器(Post processors)

后置处理器是取样器被执行后被触发执行的元素。他可用于解析响应数据，提取变量，以便后续使用  
下列元素可用于后置处理器:

* 正则表达式提取器
* XPath提取器
* Result status动作处理器
* jsr223 后置处理器
* JDBC 后置处理器
* BSF后置处理器
* jQuery/CSS 提取器
* Beanshell 后置处理器
* Debug后置处理器

测试计划元素执行顺序

测试计划的元素是有序的，通过以下方式执行：

1–配置节点

2–前置处理器

3–定时器

4–取样器

5–后置处理器（只在有结果可用情况下执行）

6–断言（只在有结果可用情况下执行）

7–监听器（只在有结果可用情况下执行）

一个测试计划可以包含一个或多个测试计划。通过功能性或者技术逻辑将测试组织在一起是一种常见的做法。

运行/停止测试计划

运行一个测试计划，你只需要点击“play”按钮：

点击按钮“Play no pauses”,开始一个测试并忽略所有暂停:

通过点击“停止”按钮，可以停止测试：

也可以禁用执行测试计划。你只需要“切换”的测试计划的状态：

你可以看到在上面的截图中看到，不可执行的计划呈现灰色，但可以修改他们或重新配置。

Web（HTTP）测试计划

在这一章中，我们将看到如何创建一个通过HTTP测试一个特定的网页的测试计划。为此目的，我将用[http://www.wikipedia.org](http://www.wikipedia.org/)网站。

有了足够理论知识，现在我们将看到如何配置一个HTTP请求的测试计划。

* 重命名“测试计划”，起一个你喜欢的名字

在这一步中，配置测试计划可能会影响线程的行为，您可以添加可能使用到的外部类库。不过我们不打算在这个例子中那样做。

* 添加线程组

在这一步，你需要创建一个负责执行所有测试组件的线程组，并配置主要属性：线程数，秒级的过渡时期，迭代次数。  
在菜单中这样操作

**Test Plan -> Add -> Thread ->ThreadGroup**

你也可以指定错误情况下行为模式（继续，停止，停止当前线程……）

* HTTP请求添加取样器

在这一步，给要测试HTTP请求添加采样器：

**Add Sampler->HTTP Request**

我们需要为这种类请求设置不同的属性：

在这个例子中，为要测试的服务器设置名称:www.wikipedia.org（不用指明HTTP或HTTPS协议）。如果需要的话,你应该修改端口，协议，执行方法等。在我们的案例中这是不必要的。这有几个关于配置代理，超时和不同的头信息的参数，但在我们的例子中，使用默认值。

* 添加结果监听器

为了后续能够看到结果,这是有必要的；就如我们之前在本教程中看到那样，对于我们的http请求的测试计划，有几种可选配置，我们使用结果树：

**Add Listener -> View Results Tree**

在我们的示例中，我们没有改变任何东西，保留所有配置属性的默认值。

* 保存并运行测试计划

**File->Save**（或单击“Control + s”）

**Run->Start**（或单击“play”）

* 在监听器中查看结果

我们可以看到，所有的请求已经完成，如预期那样，他们都提供有意义的响应，如此看来，我们测试的网页工作正常（如果这是我们的预期行为）。我们可以玩的转这些结果,检查所有返回的数据。

我们已经配置了一个测试计划,它发送大量请求到指定的服务器并解析响应。接下来我们将看到如何使用其他类型的测试组件，如定时器和断言。

* 添加定时器

为了增加一个定时器，我们只要右击测试计划并添加一个定时器：

**Add Timer->Constant Timer**

我们配置500毫秒。每一个请求都将在上一个请求执行完成后等待500毫秒。

* 添加断言持续时间

在测试计划中，单击右键并添加一个断言持续时间，如果响应时间超过100毫秒,配置错误断言。针对the main sample only选项的配置如下：

**Add assertion->Duration assertion**

显然这是因为我们配置的断言持续时间只有100毫秒，如果我们改变这个数字到2000毫秒，几乎所有请求都将按时送达。

* 添加size assertion

**Add assertion->Size Assertion**

我们需确保响应信息的尺寸大于5000字节：

如果我们运行测试计划，我们将会看到响应信息总是大于设定的值，如果响应信息尺寸小于设定值，断言也会失败。

数据库测试计划指南

我们将看到如何配置，运行一个针对MySQL数据库的测试，并对结果进行分析。这个例子是关于MySQL的，也可以使用任何其他类型的数据库如Oracle，MongoDB或别的。

* 安装MySQL

如果你没有在你的电脑安装MySQL，你可以从<http://dev.mysql.com/downloads/>下载安装MySQL服务器，这是很容易的。

* 下载MySQL驱动程序

下载MySQL连接器（mysql-connector-java-5.1.6），将它复制到JMeter安装目录下的lib目录下，本例中是C:\jmeter\apache-jmeter-2.11\lib

* 启动MySQL

启动数据库服务器，你可以对它进行查询。

* 创建数据库

你可以创建一个你想要的数据库,我们这里只是一个非常简单的例子：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | create database jmeter;    use jmeter;    create table jmeter\_stuff(id int, name varchar(50), description varchar(50));    insert into jmeter\_stuff(id, name, description)VALUES(1, "dani", "the boss");    insert into jmeter\_stuff(id, name, description)VALUES(2, "topo", "the worker");    insert into jmeter\_stuff(id, name, description)VALUES(3, "tupac", "the other"); |

* 创建一个测试计划

跟上一章节一样，进行如下操作创建测试计划:一个新的线程组，一个JDBC类型的配置元素和一个新的JDBC类型请求，以同样的方式添加一个监听器来解析和查看结果。

唯一的区别是采样器的类型，在这种情况下使用JDBC采样器。

|  |  |
| --- | --- |
| 1 | select \* from jmeter\_stuff; |

* JDBC连接配置

添加一个新的JDBC连接配置到采样器：

**点击右键 JDBC采样器->添加连接配置**

改变参数如下：

|  |  |
| --- | --- |
| 1  2  3  4  5 | Variable Bound to Pool= same name as in the sampler  Database URl=jdbc:mysql://localhost:3306/jmeter  JDBC Driver.mysql.jdbc.Driver  username=  password= |

设置用户名密码

* 保存并执行。
* 利用之前配置的监听器查看结果。

您可以以类似在HTTP请求的测试计划相同的方式验证和检查输出结果。只要你需要，你也可以添加前置处理器，后置处理器，断言或其他任何一种元素。

您可以以类似在HTTP请求的测试计划相同的方式验证和检查输出结果。只要你需要，你也可以添加前置处理器，后置处理器，断言或其他任何一种元素。

在这种情况下，输出的是SELECT语句的结果。  
你可以用这种取样器用于更新数据，以及修改采样器配置的SQL语句。  
在下面的屏幕捕获的是请求响应结果的基本信息：

JUnit测试计划

在这一章中，我们将解释如何结合JUnit使用JMeter。从负载和性能的角度来看，JMeter为执行功能测试，提供了非常强大的机制。

首先，需要创建一个像下面的JUnit测试：

|  |  |  |  |
| --- | --- | --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36 | | public class JMeterTest  {        public JMeterTest()      {          // just for testing purposes          System.out.println( "calling the constructor" );      }        @Before      public void setUp() throws Exception      {          // just for testing purposes          System.out.println( "calling setup" );      }        @After      public void tearDown() throws Exception      {          // just for testing purposes          System.out.println( "calling tearDown" );        }        @Test      public void testMe()      {          for( int i = 0; i < 30; i++ )          {              // just asserting randomly in order to check what happens in each case              int randomNumber = (int)( Math.random() \* 100 );              System.out.println( randomNumber );              assertEquals( 0, randomNumber % 2 );          }      }  } | |
| 1 |  | |

为了查看JMeter如何使用他们,本试验主要包含两个断言，当然这本教程的目的不是详细解释单元测试或JUnit测试。

* 添加JUnit测试路径

将单元测试打成一个JAR包，把它复制到JMeter的安装目录下的/lib/JUnit下。确保JAR文件包含你想让JMeter执行的测试类的.class文件。在文章的最后，你可以找到一个带有pom.xml文件的Java项目，你可以直接使用他去生成的JAR文件。现在你只需要编译代码：

|  |  |
| --- | --- |
| 1 | mvn compile |

或者直接在Eclipse或你喜欢的其他IDE中导出你的项目，生成包括字节码JAR文件。

* 添加JUnit采样器请求

入前所述，创建一个测试计划，添加一个JUnit类型采样器：Add Sampler->Junit Request ，配置并执行你想要测试的方法。

在我们的例子中，我们将运行基于JUnit 4的方法，但你也可以勾选框使用JUnit 3。

* 就像我们在前几章一样，添加一个监听器，以查看结果
* 运行测试计划
* 查看结果

在这里，可以看到我们刚刚创建的JUnit测试结果：

如上述截图所示，我们所有的测试（一个测试，10次循环）都失败了。这是预期的行为，因为我们希望我们的测试失败。其实我们只是写了个单元测试，没有功能上的意义，是无用的。它的目的是要说明如何配置JMeter去执行这些类的单元测试。

在这一章所解释的内容 有助于你使用JMeter提供的便利性的功能进行应用程序的功能测试和业务逻辑的测试.它也很有趣，因为它可以将Java语言的所有优势与JMeter的多线程能力相结合。

HTTP测试记录

现在我们将说明如何配置JMeter记录HTTP请求。为此我们需要Firefox安装在我们的机器，我是用版本32。

* 让Firefox使用JMeter代理

进入Firefox网络设置，选择手动代理选项和设置为本地主机服务器和端口8080（实际上，这加深了对你的JMeter配置）。之后选中“为所有协议使用代理服务器”。

* 跟以前一样，首先我们创建一个测试计划。

我们添加了一个新的线程组，给它起名。配置线程组线程数量为50，过渡时期为10秒，循环次数等于1。

* 添加HTTP请求默认值

在刚刚创建的线程组中添加一个新的配置元素，选择“HTTP Request Defaults”。

在这里你应该填写你想测试服务器名称。

* 添加记录控制器

现在我们添加一个记录控制器到工作台。

选择工作台，不是测试计划，并添加一个非测试元件的”Test Script Recorder”。

你现在应该让工作台保存起来，并且与你的测试计划无关。保存测试计划不意味着保存了工作台。

配置记录如下：

* 添加定时器

如果你想为你的记录器添加一个定时器（常量类型），那么该定时器将由每个被记录的HTTP请求所使用。

* 开始记录

通过点击记录器启动按钮，你将启动JMeter代理服务器。

* 记录显示

如果你为了生成一个HTTP请求而访问一个网页，例如使用Firefox（配置为在本章的第一步解释之后）访问http://www.oviedin.com，那么你会看到不同的请求入口出现在你的测试计划记录控制器中。

这些记录实际上是之前访问的web页面的HTTP请求。那些包含在过滤器配置测试脚本配置过的记录将被保存。可以在你的后续测试计划中重新配置使用。

JMeter插件

不同的插件应用的场景和用户不同，其中的一些需要安装，有些不需要安装。

一些非常有用的插件，它们通过一些高级选项来解析结果，并利用图表分析结果 例如<http://jmeter-plugins.org/wiki/GraphsGeneratorListener/>。

也有一些可用插件，能够将JMeter与你的持续集成工具连接，直接从CI软件运行JMeter，执行测试计划，例如”<https://wiki.jenkins-ci.org/display/JENKINS/Performance+Plugin>“。

正如我所说的，随着插件数量的上升，不可能列出所有的插件。所以你在实现自己的插件功能时，尽可能的去互联网上查找一下是否有可用的插件，这对自己是非常有价值的。

你也可以创建你自己的JMeter插件，但这超出了本文的范围。

JMeter最佳实践

在文章末尾，我们要写下两个很有用的技巧和最佳实践：

* 虽然JMeter UI是一个用于配置和创建不同测试计划非常有用和直观的工具，但是有经验的用户更想使用非GUI模式执行测试计划，存储他们的结果。

你可以通过命令行：

|  |  |
| --- | --- |
| 1 | jmeter -n -t test.jmx -l test.jtl. |

* 假设该测试就是你想要执行的测试计划。你必须提供该测试计划的完整存储路径。
* 监听器对解析结果是非常棒的，但他们也都是资源密集型，需要消耗大量内存，在你的配置使用尽可能少的监听器是有好处的。标记 -l 表示删除所有监视器，如果还没添加监视器，那么这个选项是不需要的。
* 如果你对测试期间发生了的错误感兴趣，可以在结果树视图中查看”View only errors“标记。
* 使用变量和循环在相同的采样代替配置几个类似的样品中的变量和参数，他们使用的是唯一的区别。
* 使用同一个采样器的变量和循环变量，而不是创建好几个这样的采样器，他们间只是变量值不同而已。
* 当创建的JUnit测试计划时，建议提供有意义的错误和断言信息，以便JMeter输出尽可能被人理解的内容。当进行单元测试时，通常情况下都会给出这样建议，而且完全适用这种情况。
* 使用Stop（Control + ‘.’）。这将尽可能的立即停止线程。
* 使用Shutdown（Control + ‘,’）。这就要求在当前任何工作结束后的停止线程。其他工具提示在[https://wiki.apache.org/jmeter/jmetershortcuts](https://wiki.apache.org/jmeter/JMeterShortcuts.)都可以找到。

总结

到这里就结束了。我们讲了如何安装JMeter，以及如何配置它用于执行不同类型的测试，如HTTP请求，数据库，基于JUnit功能测试。还有很多种JMeter提供的测试类型没有包含在本教程中。本教程的目的是提供一个完整的应用程序说明，以及教你如何配置你的第一个测试计划。

我们也解释了JMeter UI的主要组成部分，以及如何使用它，以及非常好用的JMeter测试计划元素。在最后一章中，我们提到可以为JMeter安装不同插件；对于有高级需求的经验用户来说这是一个非常有趣的。

下载

所有JMeter模板，本文中使用的SQL脚本和Java代码可以在下面的链接下载：[JMeter教程](http://a3ab771892fd198a96736e50.javacodegeeks.netdna-cdn.com/wp-content/uploads/2014/10/jmeter-tutorial.zip)。

资源

* <http://jmeter.apache.org/>
* <http://jmeter-plugins.org/>
* [href=”http://en.wikipedia.org/wiki/Software\_performance\_testing](http://en.wikipedia.org/wiki/Software_performance_testing)
* <http://www.tutorialspoint.com/jmeter/>
* [http://jmeter.apache.org/usermanual/get-started.html (section 2.4.3 Non-Guide Mode)](http://jmeter.apache.org/usermanual/get-started.html)
* <https://wiki.apache.org/jmeter/FrontPage>

原文链接： [JavaCodeGeeks](http://www.javacodegeeks.com/2014/11/jmeter-tutorial-load-testing.html) 翻译： [ImportNew.com](http://www.importnew.com/)- [张涛](http://www.importnew.com/author/zhangtao)  
译文链接： <http://www.importnew.com/13876.html>  
[ **转载请保留原文出处、译者和译文链接。**]