

# JMETER 中文手册

## 1. 简介

Apache JMeter 是 100%纯 java 桌面应用程序，被设计用来测试客户端/服务器结构的软件（例如 [web 应用程序](#)）。它可以用来测试包括基于静态和动态资源程序的性能，例如静态文件，Java Servlets，Java 对象，[数据库](#)，[FTP 服务器](#)等等。JMeter 可以用来在一个服务器、网络或者对象上模拟重负载来测试它的强度或者分析在不同的负载类型下的全面性能。

另外，JMeter 能够通过让你们用[断言](#)创建测试脚本来验证我们的应用程序是否返回了我们期望的结果，从而帮助我们回归测试我们的程序。为了最大的灵活性，JMeter 允许我们使用正则表达式创建断言。

### 1.1 历史

Apache 软件组织的 Stefano Mazzocchi 是 JMeter 的创始人。他编写它起初是为了测试 Apache JServ 的性能（一个已经被 Apache Tomcat 工程所替代的工程）。我们重新设计 JMeter 来增强用户界面并增加功能测试的能力。

### 1.2 未来

我们希望看到作为开发者利用它的可插入架构使 JMeter 的功能快速扩展。未来发展的主要目标是在没有影响 JMeter 的负载测试能力的情况下尽可能使 JMeter 成为最实用的回归测试工具。

## 2. 入门

开始使用 JMeter 最容易的方法是首先[下载最新版](#)并且安装它。这个版本包含所有你在构建和运行 Web，FTP，JDBC，和 JNDI 等测试时使用需要的所有文件。

如果你想执行 JDBC 测试，你当然需要从供应商得到适当的 JDBC 驱动。JMeter 没有提供任何 JDBC 驱动。

你可能需要下载的其它软件：

- [BeanShell](#) - BeanShell 函数和测试元件需要
- [Java Activation Framework](#) - JavaMail 需要

- [Java Mail](#) - Mail 可视化, Mail Reader 和 WebService (SOAP) 取样器需要
- [JMS](#) - JMS 取样器需要
- [General Java download page](#)



详细参见 [JMeter Classpath](#) 一章安装附加的 jar 包

- 下一步, 开始使用 JMeter 并且参见用户手册[创建一个测试计划](#)一章使自己更加熟悉 JMeter 基础 (例如, 添加和删除元件)。

最后, 参见如何构建一个明确类型的测试用例的适合章节。例如, 如果你对 Web 应用测试感兴趣, 那就参见[创建一个 Web 测试计划](#)。其他具体的测试计划章节是:

- [高级 Web 测试计划](#)
- [JDBC](#)
- [FTP](#)
- [JMS 点到点](#)
- [JMS 主题](#)
- [LDAP](#)
- [LDAP 扩展](#)
- [WebServices \(SOAP\)](#)

一旦你熟练创建和执行 JMeter 测试计划, 通过你的测试计划你会观察到给你更多帮助的各种元件的配置(定时器, 监听器, 断言, 和其他)。

## 2.1 需求

JMeter 需要最小需求的运行环境。

### 2.1.1 Java 版本



JMeter 需要一个完全适当的 JVM1.4 或者更高

因为 JMeter 仅使用 Java 标准 API, 请不要把因为 JRE 实现版本而无法运行 JMeter 的 bug 报告提交。

### 2.1.2 操作系统

JMeter 是 100%纯 Java 应用程序并且能够正确的在任何有适当的 Java 实现的操作系统上运行。

JMeter 在下列环境已经被测试：

- Unix (Solaris, Linux, 等)
- Windows (98, NT, 2000, xp)
- OpenVMS Alpha 7.3+

## 2.2 可选

如果你计划做 JMeter 开发或者想使用 SUN 的 java 标准扩展包，你将需要下列更多的可选包。

### 2.2.1 Java 编译器

如果你想编译 JMeter 源代码或者开发 JMeter 插件，你将需要一个完整的适当的 JDK1.4 或者更高。

### 2.2.2 SAX XML 解析器

JMeter 使用 Apache 的 [Xerces XML](#) 解析器，你可以选择告诉 JMeter 使用一个不同的 XML 解析器。这样做，把第三方的解析器的类包包含在 JMeter 的 [classpath](#) 中，并更新 [jmeter.properties](#) 文件里的解析器实现的全类名。

### 2.2.3 Email 支持

JMeter 有有限的 Email 能力。它能够发送基于测试结果的 Email，并且支持 POP/IMAP 取样器。它现在不支持 SMTP 取样。为了能够支持 Email，需要添加 Sun 的 JavaMail 包和 activation 包到 JMeter [classpath](#)。

### 2.2.4 SSL 加密

为了测试一个使用 SSL 加密 (HTTPS) 的 web 服务器，JMeter 需要一个提供 SSL 实现 (例如 Sun 的 Java Secure Sockets Extension - JSSE)。包含需要的加密包到 JMeter 的 [classpath](#)。同样，通过注册 SSL 提供者更新 [system.properties](#) 文件。

JMeter 默认协议等级 TLS (传输层安全性)。这可以通过修改在 [jmeter.properties](#) 或者 [user.properties](#) 文件中的 "https.default.protocol" 来改变。

JMeter 被配置接受所有的证书，不管是否信赖和合法时间等。这允许在测试服务器最大灵活性。

如果服务器需要一个客户端证书，这是可以提供的。

为了更好的管理证书，也要有一个 SSL 管理器。



JMeter 代理服务器(见下)不支持记录 SSL(https)。

## 2.2.5 JDBC 驱动

如果你需要 JDBC 测试, 需要添加厂商的 JDBC 驱动到 [classpath](#)。确认文件是一个 **jar 文件**，而不是 zip。

## 2.2.6 Apache SOAP

Apache SOAP 需要 mail.jar 和 activation.jar. 你需要下载并拷贝这两个 jar 文件到你 jmeter/lib 目录. 一旦文件放到那里, JMeter 会自动找到它们。

## 2.2.7 BeanShell

为了运行 BeanShell 函数或者任何 BeanShell 测试元件（取样器，定时器等），你需要从 <http://www.beanshell.org/> 下载 beanshell 的 jar 文件并拷贝 jar 文件到 jmeter/lib 目录，JMeter 会自动找到它。

## 2.2.8 ActiveMQ 3.0 类库

详细参见 <http://activemq.apache.org/initial-configuration.html>。



详细参见 [JMeter Classpath](#) 一章安装附加的 jar 包

## 2.3 安装



避免在一个有空格的路径安装 JMeter。这将导致远程测试出现问题。

我们推荐大多数用户运行 [最新版本](#)。

要安装一个构建版本，简单解压 zip/tar 文件到你想安装 JMeter 的目录。保证一个 JRE/JDK 正确的安装并且设置环境变量 JAVA\_HOME，其它不需要做什么了。

安装目录接口应该看到的像这样一些东西（2.3.1 版本）：

```
jakarta-jmeter-2.3.1
jakarta-jmeter-2.3.1/bin
jakarta-jmeter-2.3.1/docs
jakarta-jmeter-2.3.1/extras
jakarta-jmeter-2.3.1/lib/
jakarta-jmeter-2.3.1/lib/ext
jakarta-jmeter-2.3.1/lib/junit
jakarta-jmeter-2.3.1/printable_docs
```

如果你想的话你可以重命名父目录（例如 `jakarta-jmeter-2.3.1`），但是不要改变任何子目录名。父目录路径中不能包含任何空格；如果包含，你运行客户端-服务器模式会有问题。

## 2.4 运行 JMeter

要运行 JMeter，运行 `jmeter.bat`（for Windows）或者 `jmeter`（for Unix）文件。那些文件在 `bin` 目录下。稍微暂停后，JMeter GUI 会显示出来。

在 `bin` 目录你会发现有些附加脚本很有用。Windows 脚本文件（CMD 文件需要 Win2K 或者更新）：

- `jmeter.bat` - 运行 JMeter(默认在 GUI 模式)
- `jmeter-n.cmd` - 使用一个 JMX 文件运行非 GUI 测试
- `jmeter-n-r.cmd` - 使用一个 JMX 文件远程运行一个非 GUI 测试
- `jmeter-t.cmd` - 在 GUI 模式使用一个 JMX 文件
- `jmeter-server.bat` - 以服务器模式启动 JMeter

注意：LAST 可以在 `jmeter-n.cmd`，`jmeter-t.cmd` 和 `jmeter-n-r.cmd` 中使用，意味着最后一次测试会运行。

Unix 脚本文件；可以运行在大多 Linux/Unix 系统。

- `jmeter` - 运行 JMeter(默认在 GUI 模式)
- `jmeter-server` - 以服务器模式启动 JMeter

### 2.4.1 JMeter Classpath

JMeter 自动从下列目录中的 `jar` 文件发现类。

- `JMETER_HOME/lib` - 用来放使用的 `jar` 文件
- `JMETER_HOME/lib/ext` - 用来放 JMeter 组件和扩展

如果你开发新的 JMeter 组件，你可以压缩它们成 jar 包并拷贝到 JMeter 的 /lib/ext 目录。JMeter 将会自动发现在这里的任何 jar 文件中的 JMeter 组件。

支持的 jar 文件(类库)应该放在 lib 目录。

如果你不想把扩展 jar 包放到 lib/ext 目录，可以在 jmeter.properties 中定义 **search\_paths** 属性。不要使用 lib/ext 给那些公用的 jar 包；它仅仅是存放 JMeter 组件。

其他 jar 包（例如 JDBC，和任何 JMeter 代码需要支持的类库）应该被代替放在 lib 目录 – 而不是 lib/ext 目录。



#### 注意

JMeter 会发现.jar 文件，而不是.zip 文件。

你可以在 \$JAVA\_HOME/jre/lib/ext 安装有用的 jar 文件，或者(自从 2.1.1 版本)你可以在 jmeter.properties 中设置 user.classpath 属性。

注意设置 CLASSPATH 环境变量将不起作用。这是因为 JMeter 使用“java -jar”启动，并且 java 命令无记录忽略 CLASSPATH 变量，并且当使用 -jar 选项时 -classpath/-cp 选项也被使用。[所有的 java 程序都是这样，不仅仅是 JMeter。]

## 2.4.2 使用代理服务器

如果你从防火墙/代理服务器后测试，你需要提供给 JMeter 防火墙/代理服务器的主机名和端口号。这样做，从命令行使用以下参数运行 jmeter.bat/jmeter 文件：

-H [代理服务器主机名或者 ip 地址]

-P [代理服务器端口]

-N [非代理主机] (例如: \*.apache.org|localhost)

-u [代理证书用户名- 如果需要]

-a [代理证书密码 - 如果需要]

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

或者，你使用 --proxyHost, --proxyPort, --username, and --password



JMeter 也有自己的内建 HTTP 代理服务器，来记录 HTTP(不是 HTTPS) 浏览器会话。这是和上面的代理设置描述不混淆的，它是在 JMeter 发出 HTTP 或者 HTTPS 请求时使用的。

### 2.4.3 非用户界面模式 (命令行模式)

为了不相互影响测试，你可以选择运行没有用户界面的 JMeter。这样做，使用下列命令选项：

-n 这是指定 JMeter 在非用户界面模式运行

-t [包含测试计划的 JMX 文件的名字]

-l [记录取样结果的 JTL 文件的名字]

-r 运行在 jmeter.properties 文件里所有的远程服务器（或者通过在命令行覆盖属性指定远程服务器）

这个脚本也允许我们指定可选的防火墙/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]

例如：`jmeter -n -t my_test.jmx -l log.jtl -H my.proxy.server -P 8000`

### 2.4.4 服务器模式

为了分布测试，在服务器模式运行 JMeter，并且通过用户界面控制每一台服务器。



`jmeter-server/jmeter-server.bat` 脚本使用适当的 classpath 为你开始远程注册。如果失败，参见关于 JMeter 服务器启动细节。

运行 `jmeter-server/jmeter-server.bat`，加上下列选项命令：

这个脚本也允许我们指定可选的防火墙/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]

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

## 2.4.5 通过命令行覆盖属性

Java 系统属性，JMeter 属性，和日志属性可以通过命令行直接覆盖(代替更改 `jmeter.properties` 文件)。这样做，使用下列选项：

`-D[prop_name]=[value]` - 定义一个 java 系统属性值。

`-J[prop name]=[value]` - 覆盖一个 JMeter 属性。

`-L[category]=[priority]` - 覆盖一个日志设置，设置一个特殊目录为给定的优先级。

`-L` 标志也可以使用没有目录名来设置根目录日志等级。

例如：

```
jmeter -Duser.dir=/home/mstover/jmeter_stuff \
```

```
-Jremote_hosts=127.0.0.1 -Ljmeter.engine=DEBUG
```

```
jmeter -LDEBUG
```



### 注意

命令行参数在启动时较早被处理，但是在日志系统被设置以后。尝试使用 `-J` 标志更新 `log_level` 或者 `log_file` 属性无效。

## 2.4.6 日志和错误信息

如果 JMeter 发现一个错误，一个消息将被写入日志文件。日志文件名在 `jmeter.properties` 文件中定义。一般定义为 `jmeter.log`。并且在 JMeter 启动目录，例如 `bin`。

当在 Windows 下运行时，如果你不设置 Windows 显示文件扩展名，文件名会仅显示为 JMeter。[你可以做一些事都很容易地发现伪装成文本文件的病毒和垃圾文件...]

还有记录错误，`jmeter.log` 文件记录一些测试运行信息。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
```

日志文件对发现错误原因很有帮助，作为 JMeter 不会打断一个测试来显示一个错误对话框。

## 2.4.7 命令行选项目录

调用 JMeter 的 “jmeter -?”命令将打印所有命令选项的一个列表。列表如下：

- h, --help 打印使用信息并退出
- v, --version 打印版本信息并推出
- p, --propfile {argument} 使用的 JMeter 属性文件
- q, --addprop {argument} 附加的属性文件
- t, --testfile {argument} 运行的 JMeter 测试文件(. jmx)
- l, --logfile {argument} 日志取样文件
- n, --nongui 非用户界面运行 JMeter
- s, --server 运行 JMeter 服务器
- H, --proxyHost {argument} 设置 JMeter 使用的代理服务器
- P, --proxyPort {argument} 设置 JMeter 使用的代理服务器端口
- u, --username {argument} 设置 JMeter 使用的代理服务器用户名
- a, --password {argument} 设置 JMeter 使用的代理服务器密码
- J, --jmeterproperty {argument}={value} 定义附加的 JMeter 属性

- D, --systemproperty {argument}={value} 定义附加的 System 属性
  - S, --systemPropertyFile {filename} 一个属性文件被做为系统属性添加
  - L, --loglevel {argument}={value} 定义日志等级: [category=]level
- 例如 jorphan=INFO or jmeter.util=DEBUG
- r, --runremote 从非用户界面模式启动远程服务器
  - d, --homedir {argument} 使用的 JMeter 目录

## 2.5 配置 JMeter

如果你希望改变 JMeter 运行时的属性你需要改变在/bin 目录的 jmeter.properties 文件，或者创建你自己的 jmeter.properties 文件并且在命令行指定它。



### 注意

自从 2.1.2,你能够通过 JMeter 属性 user.properties 在文件中定义附加的 JMeter 属性，user.properties 默认值是 user.properties。如果在当前目录被发现，这个文件被自动加载。类似的，system.properties 被用来更新系统属性。

### 参数

属性	描述	需要
ssl.provider	你可以为你的 SSL 实现指定类。如果你想使用来自 sun 的 JSSE ， 是这样：com.sun.net.ssl.internal.ssl.Provider。JMeter 默认提供 https 支持。如果你正在使用 JDK1.4， 或者你使用带 JSSE 类的 jar 文件在 JMeter 的 classpath 里的 JDK1.4。	No
com.sun.net.ssl.internal.ssl.Provider。	JMeter 默认提供 https 支持是在你使用 JDK1.4 或者你使用把 JSSE 类的 jar 包放到 JMeter classpath 中的	No
xml.parser	你可以指明一个你的 XML 解析器实现。默认值是：org.apache.xerces.parsers.SAXParser	No
remote_hosts	逗号分割远程 JMeter 主机列表。如果你在一个分布式环境运行 JMeter，列出你用 JMeter 远程主机运行的机器。这允许你使用机器的用户界面控制那些服	No

	务器。
not_in_menu	在 JMeter 选项屏中你不想看到的组件列表。 如果 JMeter 被添加越来越多的组件， 你会希望定制 JMeter 只出现那些你感兴趣的组件。你可以在这儿 No 列出那些类名和他们的类标签(JMeter 的用户界面出现的字符串)， 它们将在选项屏中不出现。
search_paths	列出那些 JMeter 搜索 JMeter 附加类的路径(以;分割)；例如附加的取样器。被添加到 lib/ext 目录的任 No 何 jar 包都被发现。
user.classpath	JMeter 搜索的公用类库的路径列表。被添加到 lib 目 No 录的任何 jar 包都被发现。
user.properties	附加的 JMeter 属性文件名。 初始化属性文件后它 No 们被添加，但是在-q 和-J 选项被处理之前。
system.properties	附加的系统属性文件名。 -S 和-D 选项被执行前添 No 加。

又见 jmeter.properties 文件注释，在你改变其它设置时会给你更多的信息。

## 3. 创建一个测试计划

一个测试计划描述了一系列 Jmeter 运行时要执行的步骤。一个完整的测试计划包含一个或者多个线程组，逻辑控制器，取样发生控制，监听器，定时器，断言和配置元件。

### 3.1 添加和删除元件


在一个树上通过右击可以添加元件到一个测试计划，并且从“添加”列表中选择一个新元件。另外，元件可以从文件加载并且通过选择“打开”选项添加。

为了删除元件，确保元件被选中，正确在元件上右击，并且选择“删除”选项。

### 3.2 加载和保存元件


为了从文件加载元件，右击将要加载元件到的已经存在的树元件，并选择“打开”选项。选择你的元件保存的文件。JMeter 会加载元件到树中。

为了保存树元件，在一个元件上右击，选择“保存”选项。JMeter 会保存已选的元件，加上所有下面的子元件。用这种方法，你能够保存测试树的片段，单独元件，或者整个测试计划。

 工作台不会自动保存测试计划，但是它可以同上被单独保存


### 3.3 配置树元件

在测试树中的任何元件都在 JMeter 的右侧框架显示配置。那些配置允许你配置测试元件的细节行为，对于一个元件什么能被配置依赖于它是一个什么类型的元件。

 可以通过拖拉测试树周围的元件操作测试树。

### 3.4 保存测试计划

虽然这不是必须的，我们推荐你在运行前保存测试计划。为了保存测试计划，从文件菜单选择保存测试计划(使用最新版本, 你不再需要首先选择测试计划元件)。

 JMeter 允许你保存整个测试计划树或者仅它的一部分. 为了仅保存测试计划树中特殊“支”位置的元件, 从从这个“支”开始的地方选择树中的测试计划元件，然后右击鼠标访问保存菜单项. 另外选择合适的测试计划元件并从编辑菜单选择保存。

### 3.5 运行一个测试计划

为了运行一个测试计划，从“运行”菜单项选择“开始”。为了停止你的测试计划，从同样的菜单选择“停止”。JMeter 不会自动给它是否正在运行任何显示。如果 JMeter 运行，一些监听器使它变明显，但是唯一确定的方法是检查“运行”菜单。如果“开始”不可用，“停止”可用，证明 JMeter 正在运行你的测试计划(或者，至少，它认为它是)。

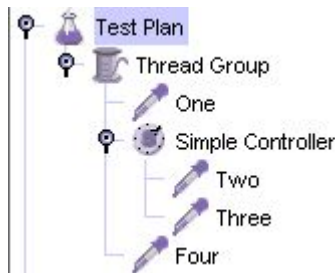
有两个类型的停止命令：

- 停止 (Control + ‘.’) – 立刻停止所有的线程
- 关闭 (Control + ‘,’) – 请求所有线程在当前任务结束后停止

### 3.6 作用域规则

JMeter 测试树包含元件总是分等级和顺序的。在测试树中的一些元件是严格分级(监听器，配置元件，后置处理器，前置处理器，断言，定时器)，一些主要

是有序的(控制器, 取样器)。当你创建测试计划时, 你将创建一个有序的取样请求(通过取样器)列表, 那些请求描述了一组步骤的执行。那些请求常组织在也有序的控制器中。给出如下测试树:

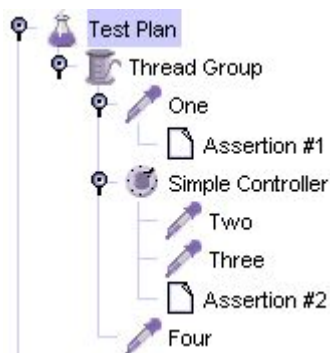


### 测试树例子

请求的顺序是 One, Two, Three, Four。

一些控制器影响它的子元件的顺序, 你可以在组件参考读到特定的控制器。

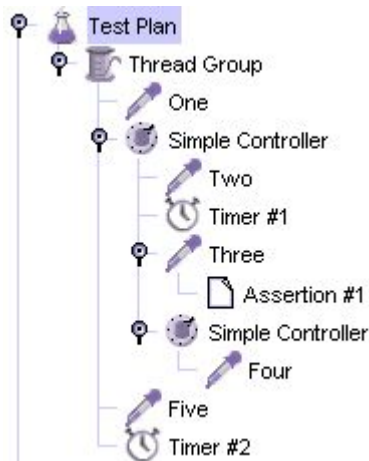
其他元素是分等级的。例如, 一个断言在测试树中是分等级的。如果它的父元件是请求, 它就被应用于那个请求。如果它的父元件是控制器, 它就影响所有那个控制器下的所有请求。如下测试树:



### 分级例子

Assertion #1 仅被应用于请求 One, Assertion #2 仅被应用于 请求 Two 和 Three。

另一个例子, 这次使用定时器:



### 复杂的例子

在这个例子中，请求的命名表现它们被执行的顺序。Timer #1 应用于 请求 Two, Three, 和 Four (注意对于分等级的元件怎样的顺序是不相关的)。Assertion #1 应用于请求 Three。Timer #2 对所有请求有效。

希望那些例子使你弄清了配置（分等级的）元件如何被应用。如果你想每个请求都被树分叉拒绝，到它的父元件，到它的父元件的父元件，等等，每次收集所有它的父元件的配置元件，你将看到它如何工作的。

元件首部管理器，Cookie 管理器和授权管理器的配置和默认元件的配置被视为是不同的。默认元件配置的设置被并入取样器可以到达的一组值里了。然而来自管理器的设置没有并入。如果多于一个管理器在一个取样器范围中，仅仅一个被使用，但是现在没有办法指定那个被使用。

## 3.7 错误报告

JMeter 把警告和错误信息报告在 jmeter.log 文件中, 也有一些测试运行本身的信息. 只是偶尔地, JMeter 对于某些错误是无法捕捉和记录的, 这些信息会显示在命令台上。如果一个测试的执行并不是你所期待的, 请检查日志, 也许错误会被报告 (例如: 也许在函数调用上有语法错误)。

取样错误 (例如: HTTP 404 - 找不到文件) 是会被正常的记录在日志中的, 取而代之的, 他们会被当作取样结果的属性来储存, 取样结果的状态能被许多不同的监听器所得到。

## 4. 测试计划元件

测试计划对象有一个叫做“功能测试”复选框。如果被选择，它会使 JMeter 记录来自服务器返回的每个取样的数据。如果你在测试监听器中选择一个文件，这个数据将被写入文件。如果你尝试一个较小的测试来保证 JMeter 配置正确并且你的服务器正在返回期望的结果，这是很有用的。这样的后果就是这个文件会快速的增大，并且 JMeter 的效率会影响。如果你不做压力测试这个选项应该关闭（默认关闭）。

如果你不记录数据到文件，这个选项就没有不同了。

## 4.1 线程组

线程组元件是任何测试计划的起点。一个测试计划的所有元件必须在一个线程组下。由名字可以看出，线程组元件控制 JMeter 运行测试时使用的线程数。线程组管理允许你：

- 设置线程数
- 设置 ramp-up period
- 设置执行测试的次数

每个线程会作为一个整体执行测试计划并完全独立于他测试线程。多线程用来模拟到达服务器程序的同步连接。

Ramp-up period 告诉 JMeter 多久开始“ramp-up”选择的全部线程。如果使用 10 个线程，ramp-up period 是 100 秒，那么 JMeter 用 100 秒使所有 10 个线程启动并运行。每个线程会在上一个线程启动后 10 秒（100/10）启动。如果有 30 个线程和一个 120 秒的 ramp-up period，那么每个连续的线程会延迟 4 秒。

Ramp-up 需要要充足长以避免在启动测试时有一个太大的工作负载，并且要充足小以至于最后一个线程在第一个完成前启动（除非你想那样发生）。

使用 ramp-up=线程数启动，并上下调整到所需的。

默认，线程组被配置仅循环一次。

1.9 版本引入了一个测试运行\*调度器\*。单击在线程组面板底部的复选框来显示额外的文本域，在里面你可以输入启动和结束时间。当测试启动时，如果必须 JMeter 会等待启动时间到达。在每个周期 结束，JMeter 检验结束时间是否到达，如果是，运行停止，如果不是测试被允许继续，直到迭代限制到达。

另外你可以使用启动延迟和持续时间文本域。注意启动延迟会覆盖启动时间，持续时间会覆盖结束时间。

## 4.2 控制器

JMeter 有两种控制器：取样器和逻辑控制器。

取样器告诉 JMeter 发送请求到服务器。例如，如果你要 JMeter 发送一个 HTTP 请求添加一个 HTTP 请求取样器。你也可以通过添加一个或者多个配置元件到一个取样器来定制一个请求。更多信息，见取样器。

逻辑控制器让你定制当发送请求时 JMeter 使用的判断逻辑。例如，你可以添加交替控制器来在两个 HTTP 请求取样器之间交替。更多信息，见逻辑控制器。

## 4.2.1 取样器 (Sampler)

取样器告诉 JMeter 发送请求到服务器。JMeter 取样器包括：

- FTP 请求
- HTTP 请求
- JDBC 请求
- Java object 请求
- LDAP 请求
- SOAP/XML-RPC 请求
- Webservice (SOAP) 请求

每个取样器有一些你可以设置的属性。**你可以通过添加一个或多个配置元件到取样器来进一步定制它。**注意 JMeter 发送请求按照取样器出现在树中的顺序。

如果你想发送多个相同类型的请求（例如，HTTP Request）到相同的服务器，可以考虑使用一个默认配置元件。每个控制器有一个或者多个默认配置元件（见下）。

记得添加一个监听器到线程组来查看/保存你的请求结果到磁盘。

如果你对使用 JMeter 平台的基础验证器到你的请求响应感兴趣，添加一个断言到请求控制器。例如，在压力测试一个 web 程序时，服务器会返回一个成功的 HTTP 响应代码，但是这个页面有错误或者被忽略部分。你可以添加断言来检查某个 HTML 标签，一些 错误字符串，等等。JMeter 允许你使用正则表达式创建断言。

JMeter 内建取样器

## 4.2.2 逻辑控制器

保持

逻辑控制器让你定制当发送请求时 JMeter 使用的判断逻辑。**逻辑控制器还可以作为下列任何元件的子元件：取样器（请求）、配置元件、和其他逻辑控制器。**



逻辑控制器可以改变来自它们的子元件的请求顺序。它们可以修改请求本身，导致 JMeter 重复请求，等。

理解逻辑控制器在测试计划中的效果，考虑下列测试树：

- 测试计划
  - 线程组
    - 仅一次控制器
      - 登录请求(一个 HTTP 请求)
    - 加载搜索页面(HTTP 取样器)
    - Interleave Controller
      - 搜索“A”(HTTP 取样器)
      - 搜索“B”(HTTP 取样器)
      - HTTP 默认请求(配置元件)
    - HTTP 默认请求(配置元件)
    - Cookie 管理器(配置元件)

这个测试的第一件事就是登录请求仅在第一次经过时被执行。随后的迭代会忽略它。这应使用仅一次控制器。

登陆后，下一个取样器加载搜索页面（假设一个用登录的 web 应用程序，并到达搜索页面去搜索）。这仅是一个简单的请求，不会被任何逻辑控制器过滤。

加载搜索页面后，我们要做一个搜索。事实上，我们想做两个不同的搜索。然而，在每个搜索之间我们想要自己重新加载搜索页面。我们通过 4 个简单 HTTP 元件这样做。(load search, search “A”, load search, search “B”). Instead, we use the Interleave Controller which passes on one child request each time through the test. It keeps the ordering (ie - it doesn't pass one on at random, but “remembers” its place) of its child elements. Interleaving 2 child requests may be overkill, but there could easily have been 8, or 20 child requests.

注意 HTTP 默认请求属于插入控制器。假如“Search A”和“Search B”共享同样的 PATH 信息（一个 HTTP 请求说明中包括域，端口，方法，协议路径和参数，附加其他可选项）。两个搜索请求访问同样的后端搜索引擎（比方说 Servle，或者 cgi 脚本），这样是说得通的。与其两者都配置使用相同信息的 HTTP 取样器，我们可以抽象那些新到一个单独的配置元件。当内部控制器通过“Search A”或者“Search B”传递时，它会从 HTTP 默认请求配置元件中获得值填充空白。所以我们可以为那些请求保留 PATH 域为空，然后把那些信息放到配置元件。在这个例子中，这至多是一个很小的好处，但它显示了这个特性。

在这个树中下一个元件是另一个 HTTP 默认请求，这个时间被添加到线程组本身。这个线程组有一个内建的逻辑控制器，因此它正好使用这个配置元件做为上面的描述。它填充任何穿过的请求的空白。在 web 程序中你所有的 HTTP 取样器元件 DOMAIN 域为空，这是极度有用的，替代的，把那些信息放到 HTTP 默认请求元

件中，添加到线程组。通过这样 做，你可以在一个同的服务器通过改变你测试计划中的一个域来测试你的程序。另外，你必须编辑每个取样器。

最后一个元件是一个 HTTP Cookie 管理器。一个 Cookie 管理器应该添加到所有 web 测试上-否则 JMeter 会忽略 Cookie。通过在线程组级添加它，我们可以确定所有的线程分享同样的 Cookie。

逻辑控制器可以组合达到不同的结果。见内建逻辑控制器列表。

## 4.3 监听器

监听器提供访问 JMeter 收集当 JMeter 运行的关于测试计划的信息。

图形结果监听器在一张图上绘制响应时间。

“查看结果树”监听器显示了请求和响应取样器的细节，并且以基础的 HTML 和 XML 显示响应表现。

其他监听器提供了摘要或者集合信息。

另外，监听器可以指导它们收集的数据到一个文件供以后用。在 JMeter 中每一个监听器提供一个域来指出存储数据的文件。

在测试中监听器可以添加到任何位置。它们仅仅会从它们等级或者它们以下等级的元件收集数据。

伴随 JMeter 有很多有趣的监听器。

## 4.4 定时器

默认，JMeter 线程发送请求时不在请求间暂停。我们建议你通过添加一个可用的定时器到你的线程组来指定一个延迟。如果你不添加延迟，JMeter 会在短时间内产生太多请求，可能会压倒你的服务。

定时器会使 JMeter 在一个线程开始每个请求间延迟一段时间。

如果你选择添加多于一个定时器到一个线程组，JMeter 会在执行取样器前获得定时器数量并暂停那个时间量。

## 4.5 断言

断言允许你断言关于从测试服务器收到的响应的行为。使用断言你本质上你可以测试你的应用程序返回你期望的结果。

例如，你可以断言一个查询的响应会包含一些特殊的文本。你指定的文本可能是 Perl 风格的正则表达式，并且你可以指出这个响应是包含这个文本，还是匹配整个响应。

你可以添加一个断言到任何取样器。例如你可以添加一个断言到 HTTP 请求检查文本“</HTML>”。JMeter 会检查在 HTTP 响应中表现的文本。如果 JMeter 没有找到这个文本，它会标记这个为一个失败的请求。

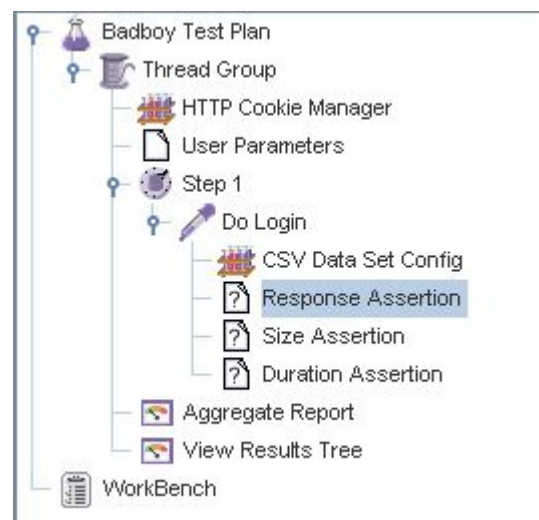
为了查看断言结果，添加一个断言监听器到线程组。

Assertion--断言，通常是用于对每一个 request sampler 进行额外验证的工具。下面通过一个例子来介绍一下常用的几个 Assertions。

假设现在要进行一个登陆的压力测试，下面是对 pass 的几个定义：

1. 正确 login 后，必须收到返回的身份验证和用户个人信息下载的完成信息；
2. 响应时间不能超过 150 毫秒
3. 返回的结果大小不能小于 750bytes

根据以上的要求，我们在对应的 request sampler 下面添加了以下的 Assertions。



#### A. Response Assertion

**Response Assertion**

**Name:** Response Assertion

**Comments:**

**Response Field to Test**

☒ Text Response
 ☐ URL Sampled
 ☐ Response Code
 ☐ Response Message
 ☐ Response Headers
 ☐ Ignore

**Pattern Matching Rules**

☒ Contains
 ☐ Matches
 ☐ Equals
 ☐ Not

**Patterns to Test**

Patterns to Test
loginMsg="OK"
profileMsg="OK"
tokenMsg="OK"

Response Field to Test -- 标示被检查对象是什么？

Pattern Matching Rules -- 标明被检查对象与验证内容之间的关系，Contains（包含关系）；Matches（匹配关系）；Equals（相等关系）；Not（非关系）

Pattern to Test -- 需要验证的内容列表

## B. Duration Assertion

**Duration Assertion**

**Name:** Duration Assertion

**Comments:**

**Duration to Assert**

**Duration in milliseconds:** 123

Duration to Assert -- 允许的响应时间的最大值

## C. Size Assertion

Size Assertion

Name:

Comments:

Size to Assert

Type of Comparison

☐ =
☐ !=
☒ >
☐ <
☐ >=
☐ <=

Size in bytes:

Size to Assert -- 对于返回结果文件大小的标准定义

## 二、结果分析

添加了以上的 Assertions 后，我运行了一次脚本。以下是存放结果的. jtl 文件的内容：

```

timestamp,elapsed,label,responseCode,responseMessage,threadName,dataT
ype,success,failureMessage,bytes,Latency
11/04/08 13:22:03,172,Do Login,200,OK,Thread Group 1-1,text,false,The
operation lasted too long: It took 172 milliseconds, but should not have
lasted longer than 150 milliseconds.,835,172
11/04/08 13:22:03,156,Do Login,200,OK,Thread Group 1-2,text,false,Test
failed: text expected to contain /loginMsg="OK"/,633,156
11/04/08 13:22:03,156,Do Login,200,OK,Thread Group 1-3,text,false,The
operation lasted too long: It took 156 milliseconds, but should not have
lasted longer than 150 milliseconds.,839,156
11/04/08 13:22:03,156,Do Login,200,OK,Thread Group 1-4,text,false,The
operation lasted too long: It took 156 milliseconds, but should not have
lasted longer than 150 milliseconds.,836,156
11/04/08 13:22:03,78,Do Login,200,OK,Thread Group 1-5,text,true,,779,78
11/04/08 13:22:03,63,Do Login,200,OK,Thread Group 1-7,text,false,Test
failed: text expected to contain /loginMsg="OK"/,570,63
11/04/08 13:22:03,141,Do Login,200,OK,Thread Group 1-6,text,false,The
result was the wrong size: It was 721 bytes, but should have been greater
than 750 bytes.,721,141

```

11/04/08 13:22:03, 78, Do Login, 200, OK, Thread Group 1-9, text, false, The result was the wrong size: It was 724 bytes, but should have been greater than 750 bytes., 724, 78

11/04/08 13:22:03, 204, Do Login, 200, OK, Thread Group 1-10, text, false, The operation lasted too long: It took 204 milliseconds, but should not have lasted longer than 150 milliseconds., 1137, 204

11/04/08 13:22:03, 2985, Do Login, 200, OK, Thread Group 1-8, text, false, Test failed: text expected to contain /loginMsg="OK"/, 570, 2985

我们可以看到，每一个 Sampler 都会经过 Assertions 的判断，只要有一个不符合条件的都会标示为 False。

## 4.6 配置元件

配置元件配合取样器工作。虽然它不发送请求（除了 HTTP 代理服务器），但是它可以添加或者修改请求。

一个配置元件能访问有所代替元件所在的树分支的内部。例如，如果你在一个简单逻辑控制器里面设置一个 HTTP Cookie 管理器，Cookie 管理器很容易访问“web Page 1”和“web Page 2”HTTP 请求。但是不能访问“web Page 3”。

同样，一个在树枝内部的配置元件比在父支的同样元件有更高的优先级。例如，我们定义两个 HTTP 默认请求元件，“Web Defaults 1”和“Web Defaults 2”。如果我们把“Web Defaults 1”放置在一个循环控制器内部，仅“Web Page 2”可以访问它。另一 HTTP 请求会使用“Web Defaults 2”，如果我们把它放置在线程组（所有其他分支的父支）。

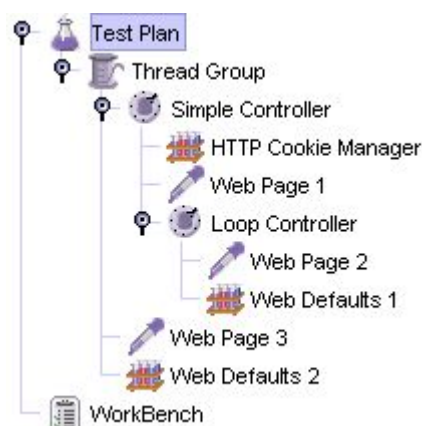


图 1 - 测试计划展示配置元件的可达性

## 4.7 前置处理器元件


前置处理器在取样器请求建立前执行一些行为。如果前置处理器隶属于取样器元件，那么它会仅在那个取样器元件运行前执行。前置处理器最常用在取样请求运行前修改它的设置，或者更新不能从响应文本提取的变量。当前置处理器执行时，详细信息见作用域规则。

## 4.8 后置处理器元件

后置控制器在取样器请求建立后执行一些行为。如果后置处理器隶属于取样器元件，那么它会仅在那个取样器元件运行后执行。后置处理器最常用来处理响应数据，常用来从它里面提取数值。\\\\\\\\\\\\\\\\详细见作用域规则关于前置处理器执行。

## 4.9 执行顺序

1. 配额制元件
2. 前置处理器
3. 定时器
4. 取样器
5. 后置处理器（如果 SampleResult 不为空）
6. 断言（如果 SampleResult 不为空）
7. 监听器（如果 SampleResult 不为空）

 Please note that Timers, Assertions, Pre- and Post-Processors are only processed if there is a sampler to which they apply. Logic Controllers and Samplers are processed in the order in which they appear in the tree. Other test elements are processed according to the scope in which they are found, and the type of test element. [Within a type, elements are processed in the order in which they appear in the tree].

For example, in the following test plan:

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

- Timer 2
- Post-Processor 2

The order of execution would be:

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

## 4.10 Properties and Variables

JMeter properties are defined in `jmeter.properties` (see Getting Started – Configuring JMeter for more details).

Properties are global to `jmeter`, and are mostly used to define some of the defaults JMeter uses. For example the property `remote_hosts` defines the servers that JMeter will try to run remotely. Properties can be referenced in test plans – see Functions – read a property – but cannot be used for thread-specific values.

JMeter variables are local to each thread. The values may be the same for each thread, or they may be different.

If a variable is updated by a thread, only the thread copy of the variable is changed. For example the Regular Expression Extractor Post-Processor will set its variables according to the sample that its thread has read, and these can be used later by the same thread. For details of how to reference variables and functions, see Functions and Variables

Note that the values defined by the Test Plan and the User Defined Variables configuration element are made available to the whole test plan at startup. If the same variable is defined by multiple elements, then the last one takes effect. Other elements such as the User Parameters Pre-Processor or Regular Expression Extractor Post-Processor may be used



to redefine the same variables. These redefinitions only apply to the current thread.

Note that global variables cannot be updated during a test. The `setProperty` function can be used to define a JMeter property. These are global to the test plan, so can be used to pass information between threads.

## 5. 创建一个 Web 测试计划

在这一部分，你将学会如何创建一个基础的[测试计划](#)来测试一个 Web 站点，你将创建 5 个用户向 Jackrta 网站上的两个网页发送请求。同样，你要告诉用户运行测试两次。这样，总的 HTTP 发送请求为（5 个用户\*2 次请求\*重复 2 次）=20。要创建这个测试计划，你将会用到下面的元件：[线程组](#)，HTTP 请求，HTTP 请求默认值和图形结果。

要创建更好的测试计划，可以参考[创建一个高级 web 测试计划](#)。

### 5.1 添加用户

处理每个 JMeter 测试计划的第一步就是添加[线程组](#)元件。这个线程组会告诉 JMeter 你想要模拟的用户数量，用户应该发送请求的频率和应该发送的数量。

下一步来添加一个线程组：

首先选择这个测试计划，用鼠标右键点击然后在得到的菜单中选择“添加”--> “线程组”。

这时你应该看到这个线程组已经在测试计划下面了，如果没有看到，就点击测试计划元件展开这个测试计划树。

下一步，你需要修改这些默认的属性。如果你还没有选择线程组元件，则从测试计划树型结构中选择它。这时你应该看到 JMeter 窗口右边的线程组控制面板了。

**Thread Group**

Name:

Action to be taken after a Sampler error

☒ Continue ☐ Stop Thread ☐ Stop Test

Thread Properties

Number of Threads (users):

Ramp-Up Period (in seconds):

Loop Count: ☐ Forever

☐ Scheduler


图 5.1. 线程组默认值

首先给这个线程组起一个有意义的名字。在名称域中，输入 Jakarta Users.

下一步，增加用户的数量（称为线程）为 5。

**域 Ramp-Up Period:**，使用默认值为 0。这个属性表示每个用户启动的迟延时间。例如，如果你输入 Ramp-Up Period 为 5 秒，JMeter 将会在 5 秒结束前完成启动所有的用户。所以，如果你有五个用户并且 Ramp-Up Period 为五秒，那么开始用户的延迟就是 1 秒。(5 个用户 / 5 秒 = 1 用户每秒)。如果你设置其值为 0，JMeter 将会立即启动你所有的用户。

**域 Loop Count:**取消标记为“永远”的复选框选择并设置循环次数为 2。这个属性表示你的测试的重复次数。如果你设置为 1，JMeter 将你的测试只运行一次。要让 JMeter 不断的运行，你要选择“永远”这个复选框。

 在大多数的应用程序中，你需要手动来接受你在控制面板中所做的修改。但在 JMeter 中，如果你做了修改，控制面板可以自动的接受。如果你修改的元件的名字，当你离开控制面板后树型菜单自动更新(例如，当你选择另外一个树元件)。

见图 5.2 为完整的 Jakarta Users 线程组。

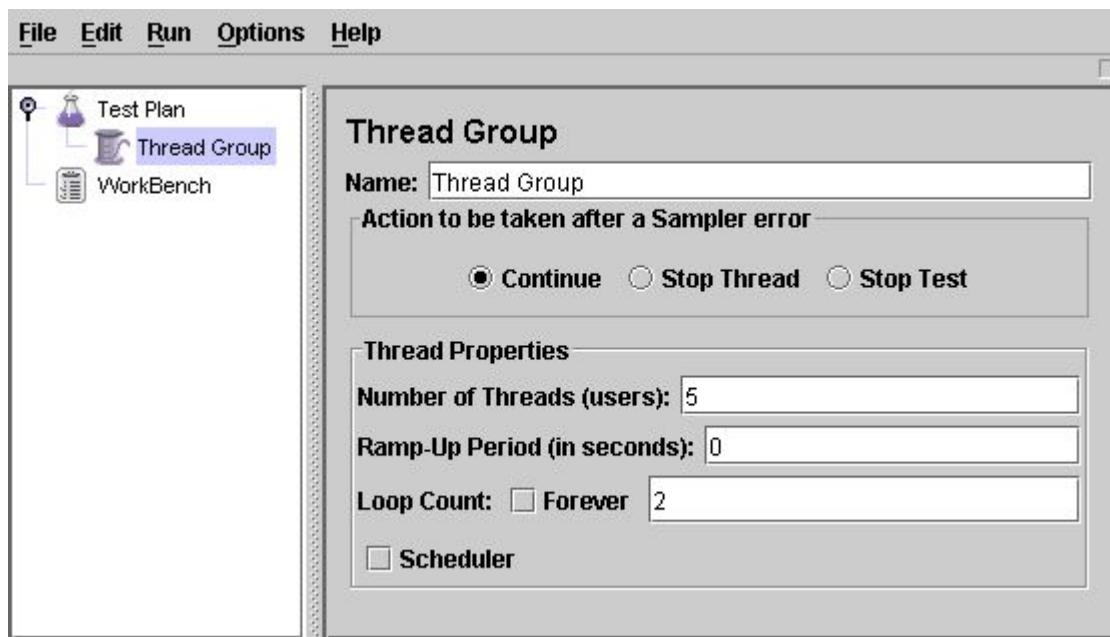


图 5.2. Jakarta Users 线程组

## 5.2 添加默认 HTTP 请求属性

我们已经定义了用户，现在要定义他们的行为了。在这一部分，你将学会对你的 HTTP 请求设置默认值。然后在 5.3 节，用你在这里指定的默认设置来添加 HTTP 请求元件。

首先选择 Jakarta Users(就是刚才的线程组)元件，右键点击并在弹出的菜单中选择“添加”-->“配置元件”-->“HTTP 请求默认值”。然后选择这个新元件来显示其控制面板（见图 5.3）。

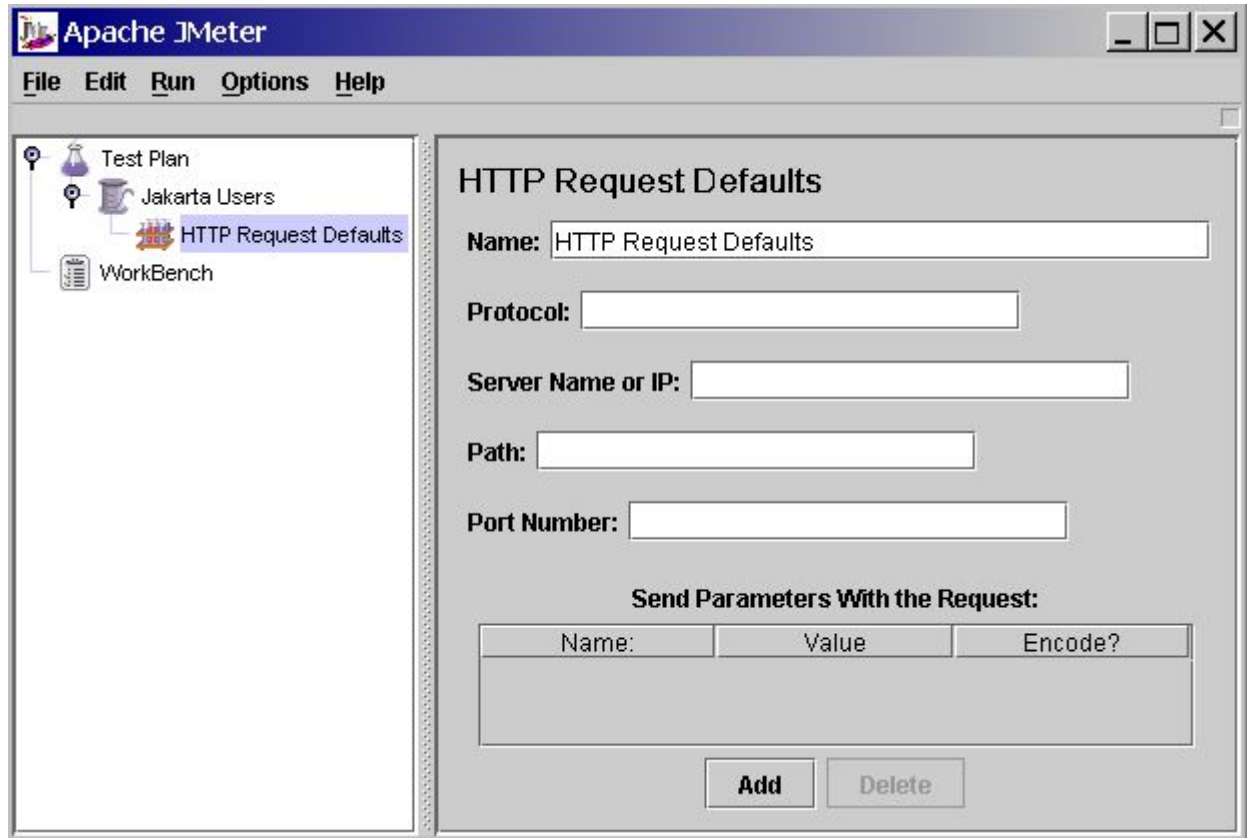


图 5.3. HTTP 请求默认值

跟大多数的 JMeter 元件一样， HTTP 请求默认值控制面板也有一个名称域。在这个例子中将它保留为默认值。

下面这个文本域是 Web 服务器的服务器/IP。对于你创建的测试计划，所有的 HTTP 请求都将发送到相同的 Web 服务器 `jakarta.apache.org`。向这个域中输入这个域名，这是唯一一个需要我们去修改它的默认值的文本域，其它的文本域都保留它们的默认值。

⚠ HTTP 请求默认值元件并不告诉 JMeter 来发送 HTTP 请求，它仅仅定义这个 HTTP 请求所用的默认值。

见图 5.4 完整的 HTTP 请求默认值元件

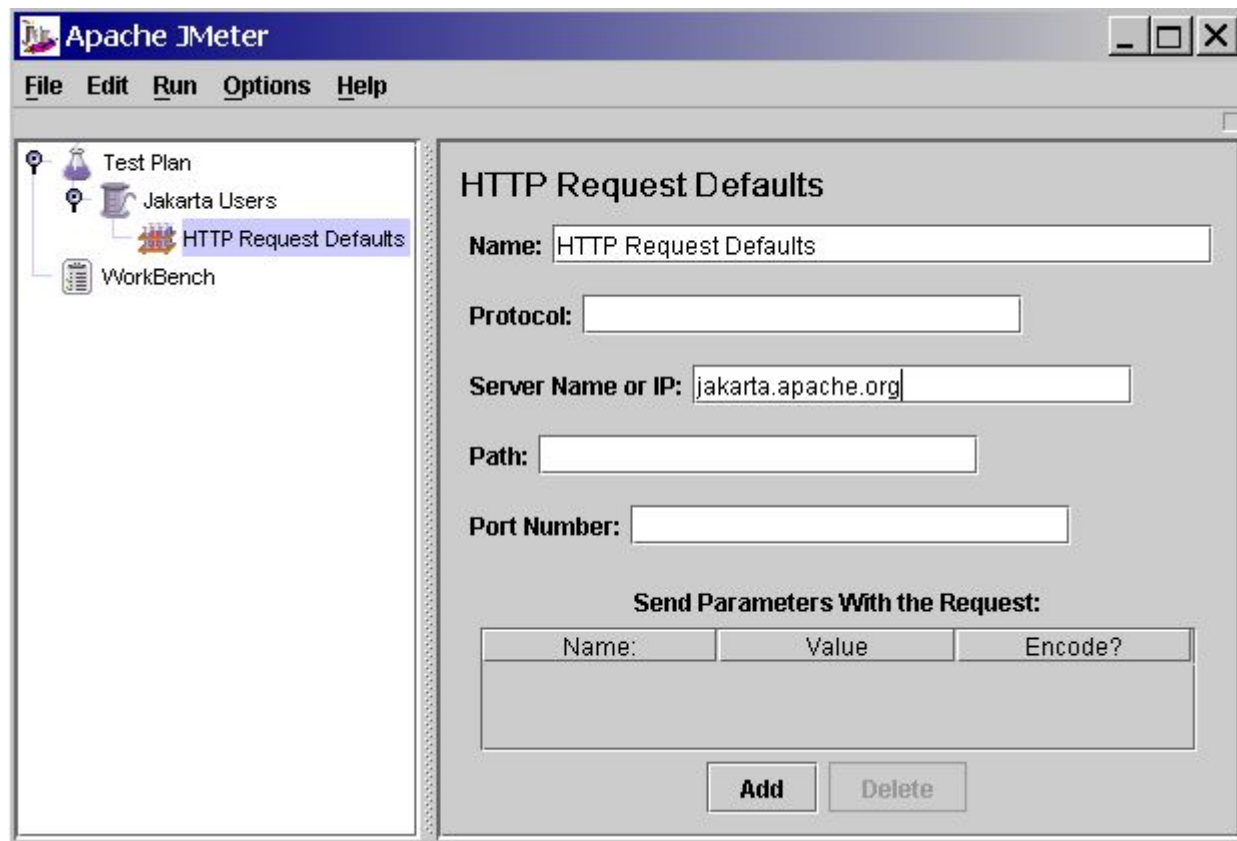


图 5.4. 测试计划的 HTTP 默认值

## 5.3 添加 Cookie 支持

除非你的应用程序明确的不使用 Cookies，几乎所有的网站应用程序都会使用 cookie 支持。要添加 cookie 支持，可以简单的在你的测试计划中给每一个线程组添加一个 HTTP Cookie 管理器。这样确保每个线程组有自己的 cookies，但是共享跨越所有的 HTTP 请求对象。

添加 HTTP Cookie 管理器，简单地选择这个[线程组](#)，选择添加—>配置元件—>HTTP Cookie 管理器，也可以从编辑菜单或通过右键点击来实现添加。

## 5.4 添加 HTTP 请求

在这个测试计划中，我们需要实现两个 HTTP 请求。第一个就是 Jakarta 网站首页 (<http://jakarta.apache.org/>)，第二个就是工程向导网页 (<http://jakarta.apache.org/site/guidelines.html>)。



JMeter 按照它们在树的出现的次序来发送请求。

首先给 Jakarta Users 元件添加第一个 HTTP 请求（添加--> 取样器--> HTTP 请求）。然后从树中选择 HTTP 请求元件并修改下列属性（见图 5.5）：（取样器就是 sampler）

- 更改名称域为“Home Page”。
- 设置路径域为 “/”。记得你不必设置服务器名域，因为你已经在 HTTP 默认请求元件中指定了这个值。

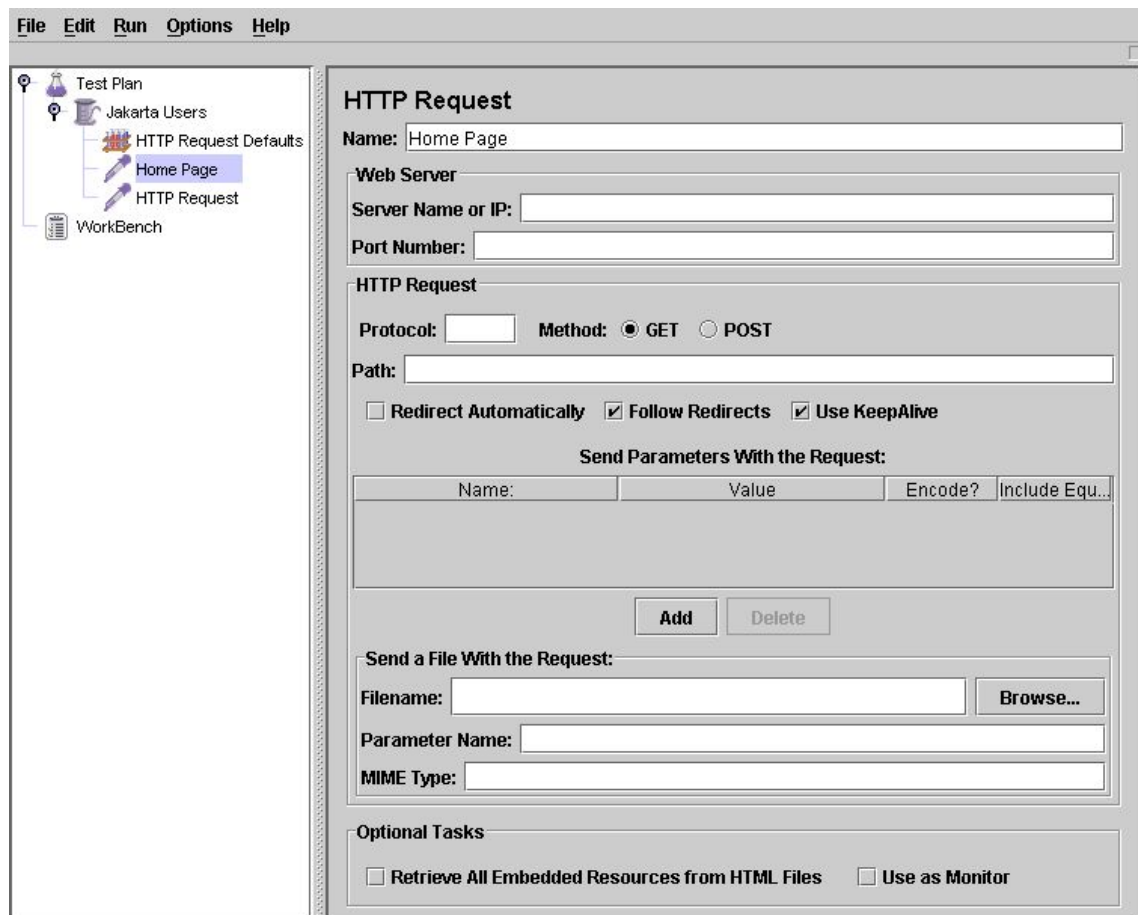


图 5.5. Jakarta 首页的 HTTP 请求

下一步，添加每二个 HTTP 请求并修改下列属性（见图 5.6）：

- 更改名称域为“Project Guidelines”。
- 设置路径域为 “/site/guidelines.html”。

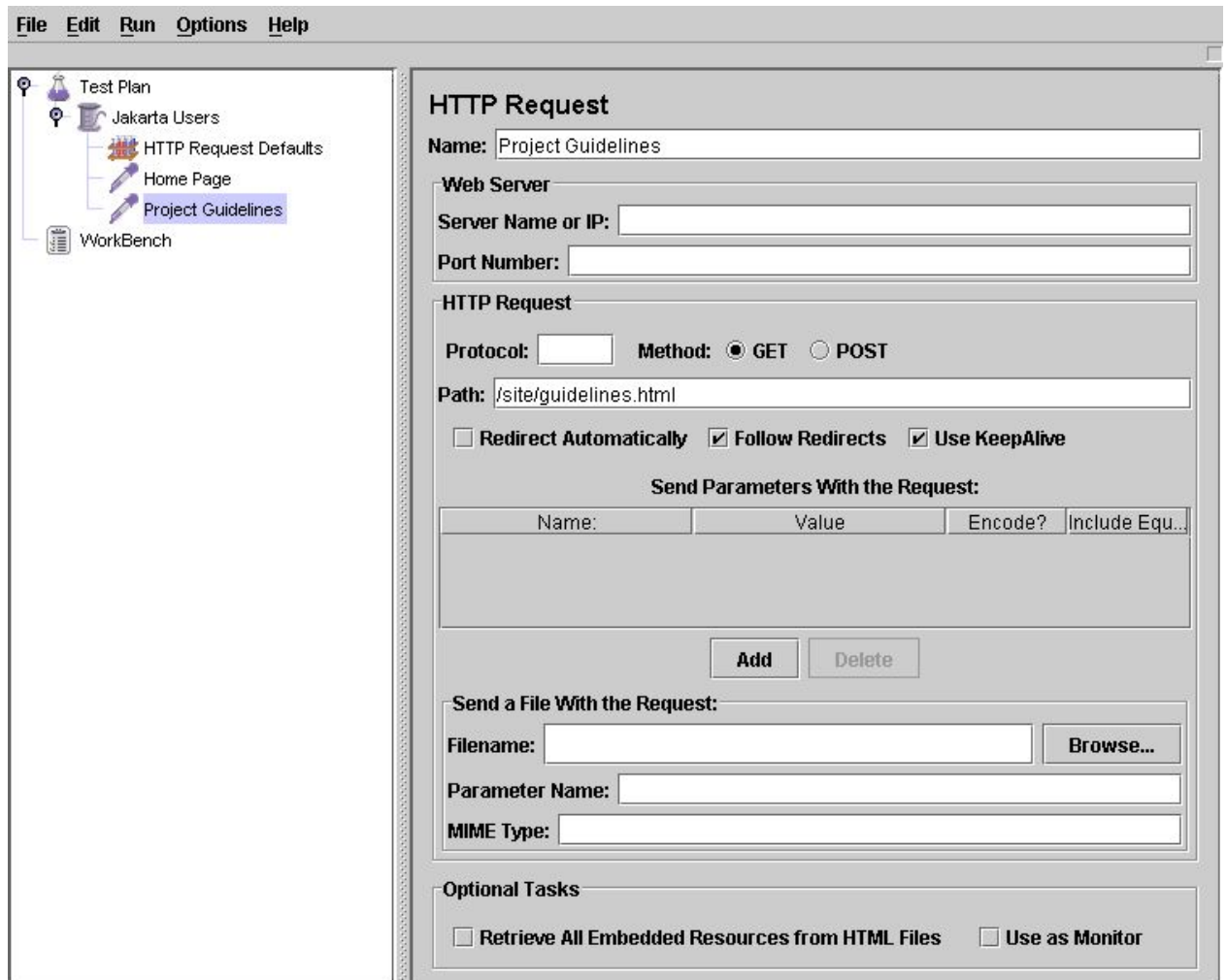


图 5.6. Jakarta 工程 Guidelines 页的 HTTP 请求

## 5.5 添加一个监听器来浏览/储存测试结果

最后一个需要添加到测试计划的元件是监听器。这个元件的用途是将所有的 HTTP 请求结果存储在一个文件中并显现出数据的可视模型。

选择 Jakarta Users 元件，然后添加一个“图形结果”监听器（添加--> 监听器 --> 图形结果）。接着，你需要指定一个文件路径和输出文件名。你可以在文件名域中输入或选择浏览按钮并选择一个路径然后输入文件名。





图 5.7. 图像结果监听器

## 5.6 登录一个 web 站点

在这它不是一个例子, 但是一些网站需要在许可你做某些操作前登录. 在一个 web 浏览器中, 登录会表现为一个包含用户名, 密码和一个提交按钮的表单. 这个按钮产生一个 POST 请求, 传递表单的值作为参数. It's not the case here, but some web-sites require you to login before permitting you to perform certain actions. In a web-browser, the login will be shown as a form for the user name and password, and a button to submit the form. The button generates a POST request, passing the values of the form items as parameters.

在 JMeter 中这样做, 添加一个 HTTP 请求, 并设置为 POST 方法. 你还要通过表单知道域的名字和目标页面. 通过查看那登陆页面的代码可以发现 它们. To do this in JMeter, add an HTTP Request, and set the method to POST. You'll need to know the names of the fields used by the form, and the target page. These can be found out by inspecting the code of the login page. [如果这比较难做, 你可以使用 JMeter 代理记录器来记录登录顺序. If this is difficult to do, you can use the JMeter Proxy Recorder to record the login sequence.] 设置路径为提交按钮的目标. 单击添加按钮并输入用户名和密码资料. 有时登录表单包含附加的隐藏域. 它们也需要被添加. Set the path to the target of the submit button. Click the Add button twice and enter the



username and password details. Sometimes the login form contains additional hidden fields. These will need to be added as well.

HTTP Request

Name: Login

Comments:Example login page

Web Server

Server Name or IP: www.sample.com

Port Number:

HTTP Request

Protocol (default http):

Method: POST

Content encoding:

Path: /loginform.html

☒ Redirect Automatically

☐ Follow Redirects

☒ Use KeepAlive

☐ Use multipart/form-data for HTTP POST

Send Parameters With the Request:

Name:	Value	Encode?	Include Equ...
username	johndoe	<input type="checkbox"/>	<input checked="" type="checkbox"/>
password	secret	<input type="checkbox"/>	<input checked="" type="checkbox"/>

Add

Delete

Send a File With the Request:

Filename:

Browse...

Value for "name" attribute:

MIME Type:

Optional Tasks

☐ Retrieve All Embedded Resources from HTML Files

☐ Use as Monitor

Embedded URLs must match:

Figure 5.8. Sample HTTP login request

## 6. 创建一个高级 web 测试计划

在这章，你将学到如何创建高级测试计划来测试 web 站点。

如果需要一个基础的测试计划例子，见创建一个 web 测试计划 。

## 6.1 用 URL 重写处理用户会话

如果你的 web 应用程序使用 URL 重写, 而不是 cookies 保存会话信息, 那么为了测试你的站点你将需要做一点额外的工作。

为了正确响应 URL 重写, JMeter 需要解析从服务器得到的 HTML 并获取唯一的会话 ID。利用适当的 HTTP URL 重写修改器来完成这些。简单地在修改器中输入你的会话 ID 参数名, 它会找到它并添加它到每一个请求。如果请求已经有一个值, 它将会被替代。如果“缓存会话 Id?”被选中, 那么最后创建的会话 ID 将被保存, 并且如果 HTTP 的上次取样不包含一个会话 ID, 它将会被使用。

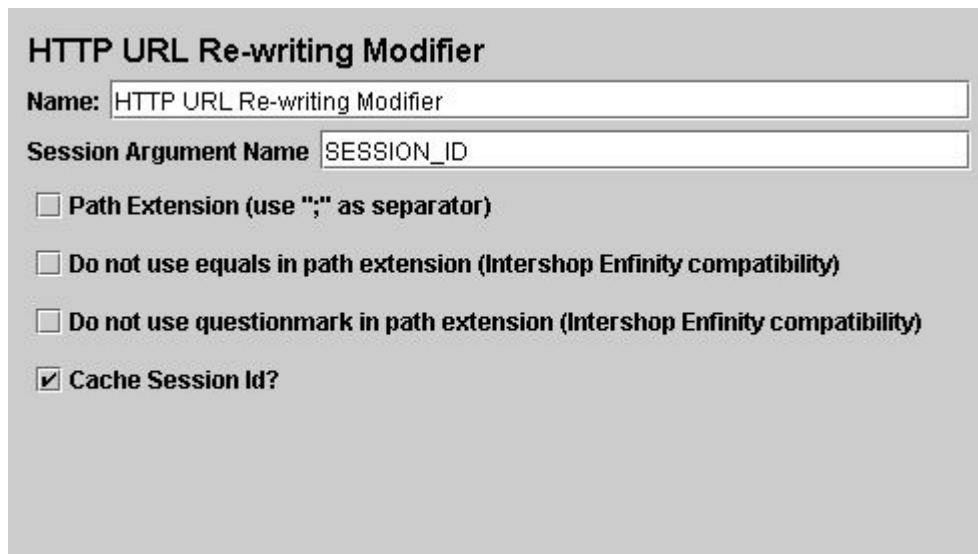
### URL 重写例子

下载[这个例子](#)。在图 1 中展示了一个使用 URL 重写的测试计划。注意 URL 重写修改器附被添加到简单控制器, 因此确认它仅影响简单控制器下的请求。



图 1 - 测试树

在图 2 中, 我们看到了 URL 重写修改器的 GUI, 它仅仅有一个让用户指定会话 ID 参数名的文本域。有一个复选框来指示会话 ID 将被化为为路径 (以“;”隔开), 这样胜过使用一个请求参数。



**HTTP URL Re-writing Modifier**

Name:

Session Argument Name

☐ Path Extension (use ";" as separator)

☐ Do not use equals in path extension (Intershop Enfinity compatibility)

☐ Do not use questionmark in path extension (Intershop Enfinity compatibility)

☒ Cache Session Id?

图 2 - 请求参数


## 6.2 使用消息头管理器

HTTP 消息头管理器允许你定制 JMeter 在 HTTP 请求消息头发送什么信息。这个消息头包括像 "User-Agent", "Pragma", "Referer" 等属性。

HTTP 消息头管理器像 HTTP Cookie 管理器一样, 如果你因为一些原因你不希望在你的测试里为不同的 HTTP 请求对象指定不同的消息头, 可以添加到线程组等级。

# 7. 创建一个数据库测试计划

在这一部分, 你将学会如何去创建一个基础的[测试计划](#)来测试一个数据库服务器。你会创建 10 个用户来给数据库服务器发送 2 次 SQL 请求。同样, 你也可以让用户运行他们的测试三次。这样总的 JDBC 请求数量就是 (10 用户) \* (2 次请求) \* (重复 3 次) = 60。要构建这个测试计划, 你将会用到下面的元件: [线程组](#), JDBC 请求, 图形结果。

 这个例子使用了 MySQL 数据库驱动。要使用这个驱动, 它所包含的 .jar 文件必须复制到 ../lib/directory 下 (详情参见 [JMeter's ClassPath](#))。

## 7.1 添加用户

处理每个 JMeter 测试计划的第一步就是添加[线程组](#)元件。这个线程组会告诉 JMeter 你想要模拟的用户数量，用户应该发送请求的频率和应该发送的数量。

开始来添加一个线程组：首先选择这个测试计划，点击鼠标右键得到添加菜单，然后选择添加-->线程组。

这时你应该看到这个线程组已经在测试计划下了，如果没有看到，就点击测试计划元件展开这个测试计划树。

下一步，你需要修改这些默认的属性。如果你还没有选择线程组元件，则从测试计划树型结构中选择它。这时你应该看到 JMeter 窗口右边的线程组控制面板了（见下图 7.1）。

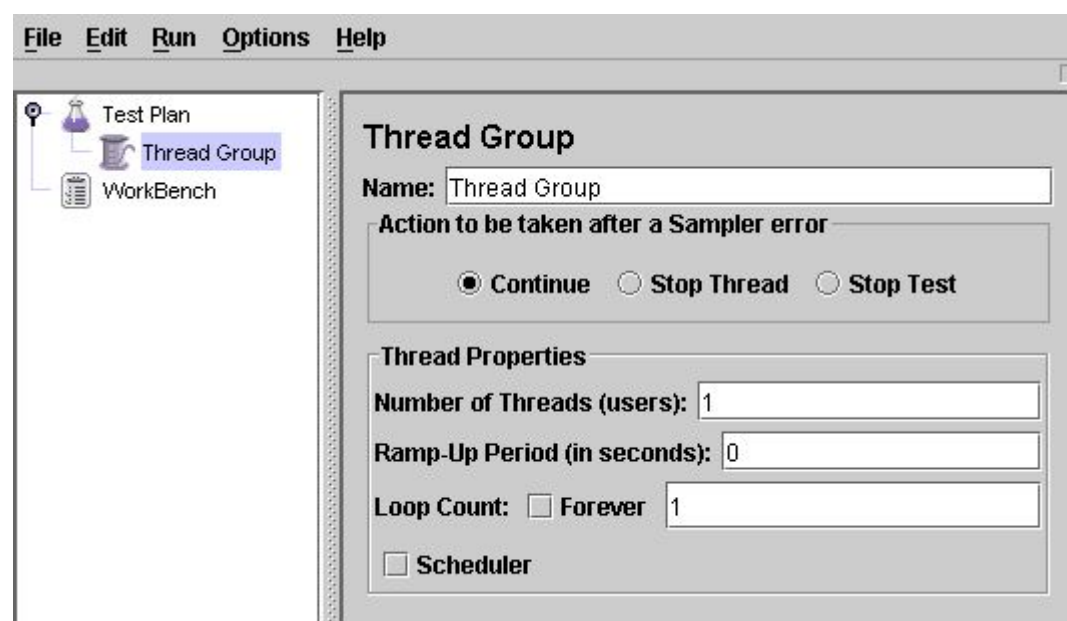



Figure 7.1. Thread Group with Default Values

首先给这个线程组起一个有意义的名字。在名称域中，输入 JDBC Users。

 你将需要一个可用的数据库，数据库表，和表的用户使用权限。  
在这个例子中，数据库是 'mydb'，表名是 'Stocks'。

接下来，将用户的数量（即线程）增加到 10。

在下一个 Ramp-Up Period 文本域中，使用默认值 0。这个属性表示每个用户启动的延迟时间。例如，如果你输入 Ramp-Up Period 为 5 秒，JMeter 将会在五秒结束前完成启动所有的用户。所以，如果你有五个用户并且 Ramp-Up Period 为五秒，那么启动用户的延迟就是 1 秒。（5 个用户 / 5 秒 = 1 用户每秒）。如果你设置其值为 0，JMeter 将会立即启动你所有的用户。

最后，输入循环次数为 3。这个属性告诉 JMeter 你的测试运行多少次。如果你设置为 1，JMeter 将你的测试只运行一次。要让 JMeter 不断的运行，你要选择“永远”这个复选框。

⚠ 在大多数的应用程序中，你需要手动来接受你在控制面板中所做的修改。但在 JMeter 中，如果你做了修改，控制面板可以自动的接受。如果你修改的元件的名字，树型菜单当你离开控制面板后自动更新。（例如，当你选择另外一个树元件。）

见图 7.2 完整的 JDBC Users 线程组。

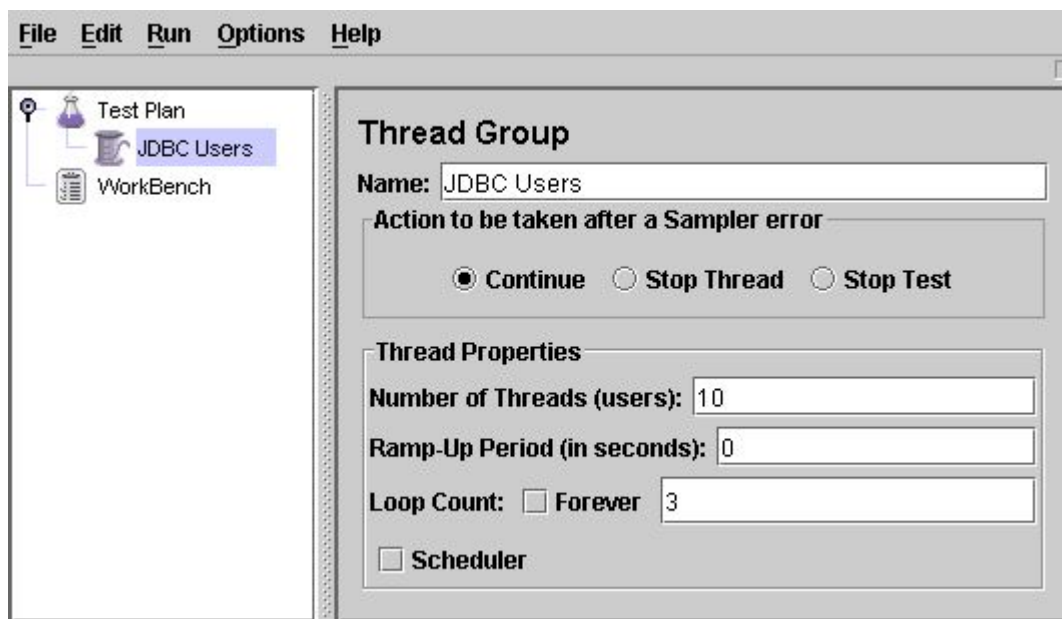


图 7.2. JDBC Users 线程组

## 7.2 添加 JDBC 请求

现在我们已经定义了用户，是时候定义他们。在这一部分，我们将会详细说明 JDBC 请求的执行。

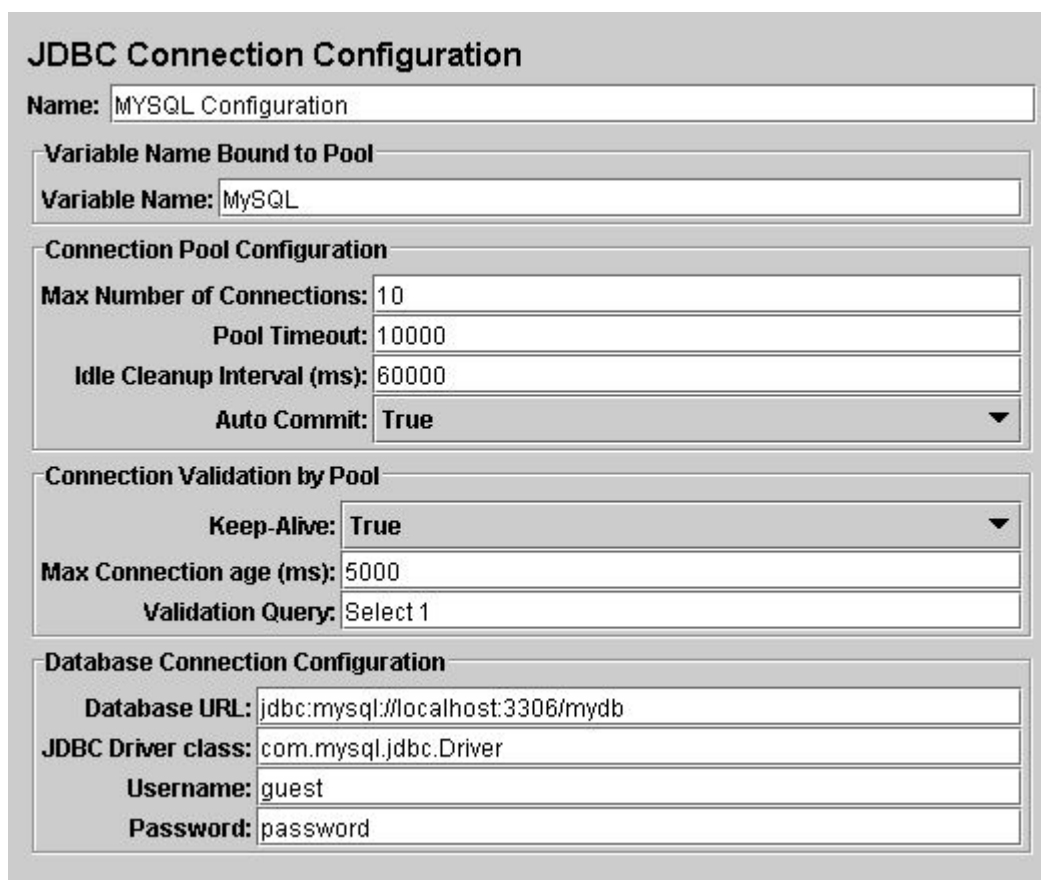
首先选择 JDBC 用户元件，点击鼠标右键，得到添加菜单，然后选择添加-->配置元件-->JDBC 连接配置。接着，选择这个新元件来显示它的控制面板（见图 7.3）。

设定下面的文本域(我们这里假定用一个名为 test 的本地 MySQL 数据库)。

- 绑定到池变量。这需要能够唯一标识这个配置。这是用来被 JDBC 取样器识别这个配置来使用。
- 数据库 URL: jdbc:mysql://localhost:3306/test
- JDBC 驱动类: com.mysql.jdbc.Driver
- 用户名: guest

- 密码: guest 的密码

剩下的域保留默认值。



**JDBC Connection Configuration**

**Name:** MySQL Configuration

**Variable Name Bound to Pool**

**Variable Name:** MySQL

**Connection Pool Configuration**

**Max Number of Connections:** 10

**Pool Timeout:** 10000

**Idle Cleanup Interval (ms):** 60000

**Auto Commit:** True

**Connection Validation by Pool**

**Keep-Alive:** True

**Max Connection age (ms):** 5000

**Validation Query:** Select 1

**Database Connection Configuration**

**Database URL:** jdbc:mysql://localhost:3306/mydb

**JDBC Driver class:** com.mysql.jdbc.Driver

**Username:** guest

**Password:** password

图 7.3. JDBC 配置

再次选择 JDBC 用户元件。点击鼠标右键，得到添加菜单，然后选择添加-->Sampler-->JDBC 请求。然后，选择这个新元件来显示其控制面板（见图 7.4）。

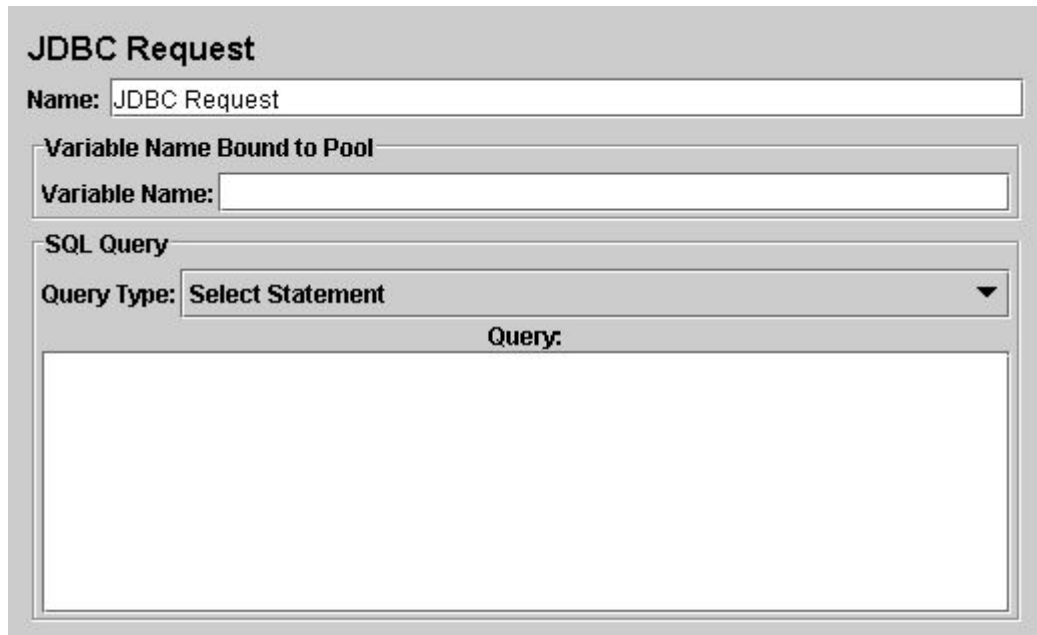
The image shows the 'JDBC Request' configuration window in JMeter. It has a title bar 'JDBC Request'. Below it, there are several fields: 'Name:' with the value 'JDBC Request', 'Variable Name Bound to Pool' (empty), 'Variable Name:' (empty), 'SQL Query' (empty), 'Query Type:' with a dropdown menu showing 'Select Statement', and a large 'Query:' text area at the bottom.

图 7.4. JDBC 请求

在我们这个测试计划中，我们将发送 2 个 JDBC 请求。第一个是向 Eastman Kodak stock, 第二个是向 Pfizer stock (很显然需要改变这些例子来适合你的特殊的数据库)。下面有插图文字说明。

⚠ JMeter 发送请求的次序就是你向树中添加它们的次序。

开始编辑下列的属性 (见图 7.5) :

- 修改名称为“Kodak”
- 输入池名:MySQL (在配置元件里面一样)
- 输入 SQL 查询字符串域

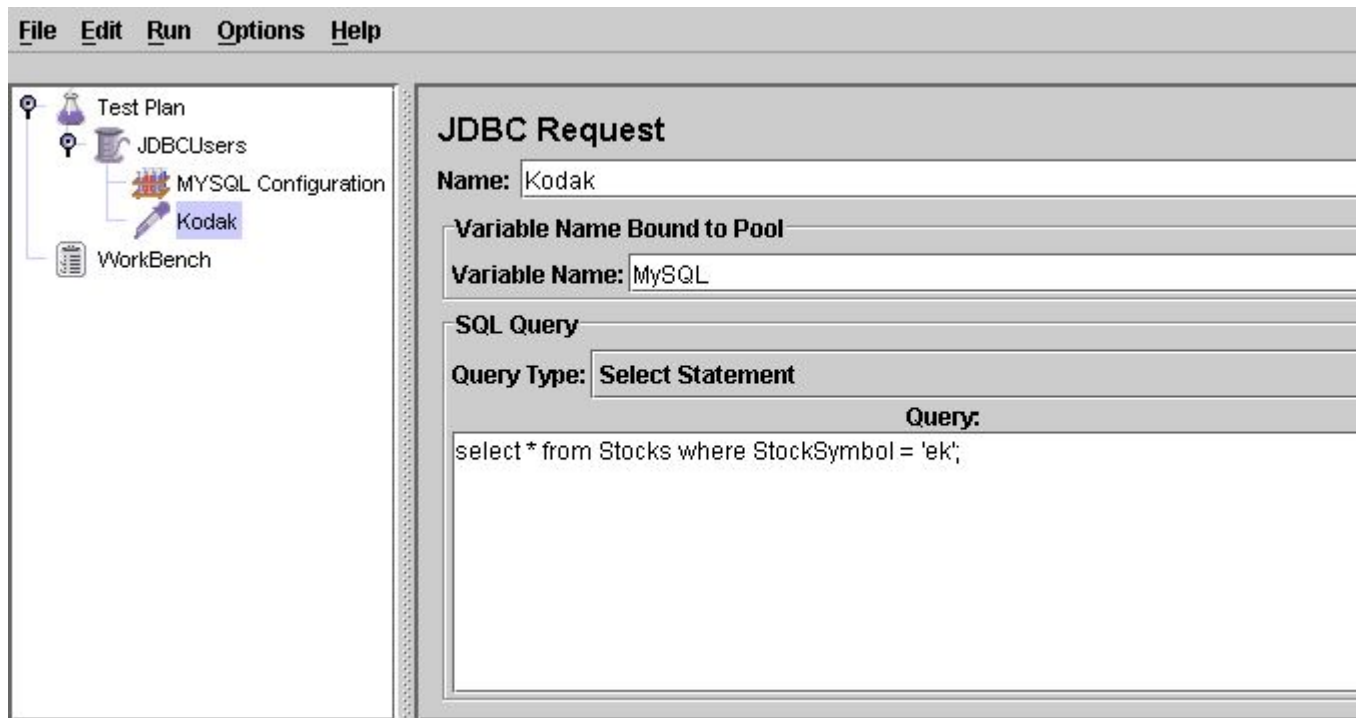


图 7.5. Eastman Kodak stock 的 JDBC 请求

下一步，添加第二个 JDBC 请求并编辑下列的属性（见图 7.6）：

- 修改名字为“Pfizer”
- 输入 SQL 查询字符域

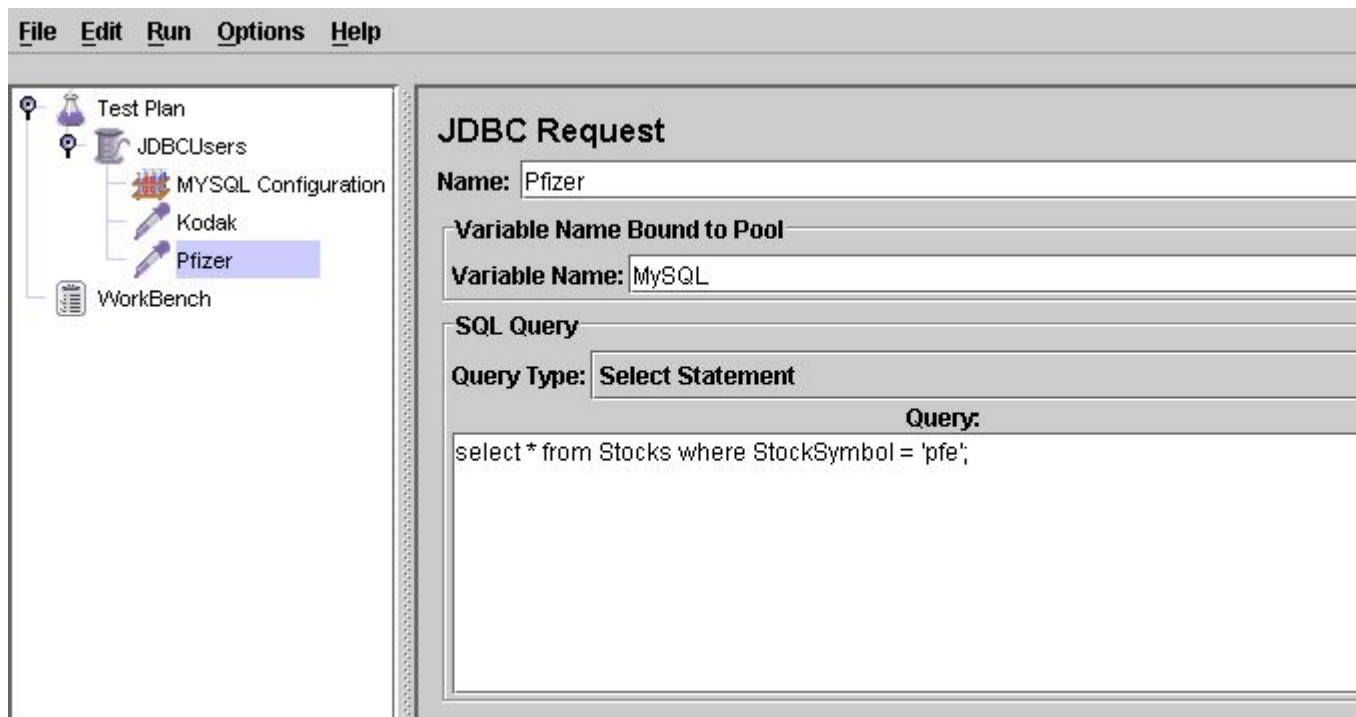


图 7.6. Pfizer stock 的 JDBC 请求



## 7.4 添加一个监听器浏览/保存测试结果

你需要添加到你测试计划的最后元件是一个监听器。这个元件责任是储存所有你的 JDBC 请求结果到文件，并且展示一个可视数据模型。

选择 JDBC 用户元件，添加一个图形结果监听器（添加-->监听器-->图形结果）。

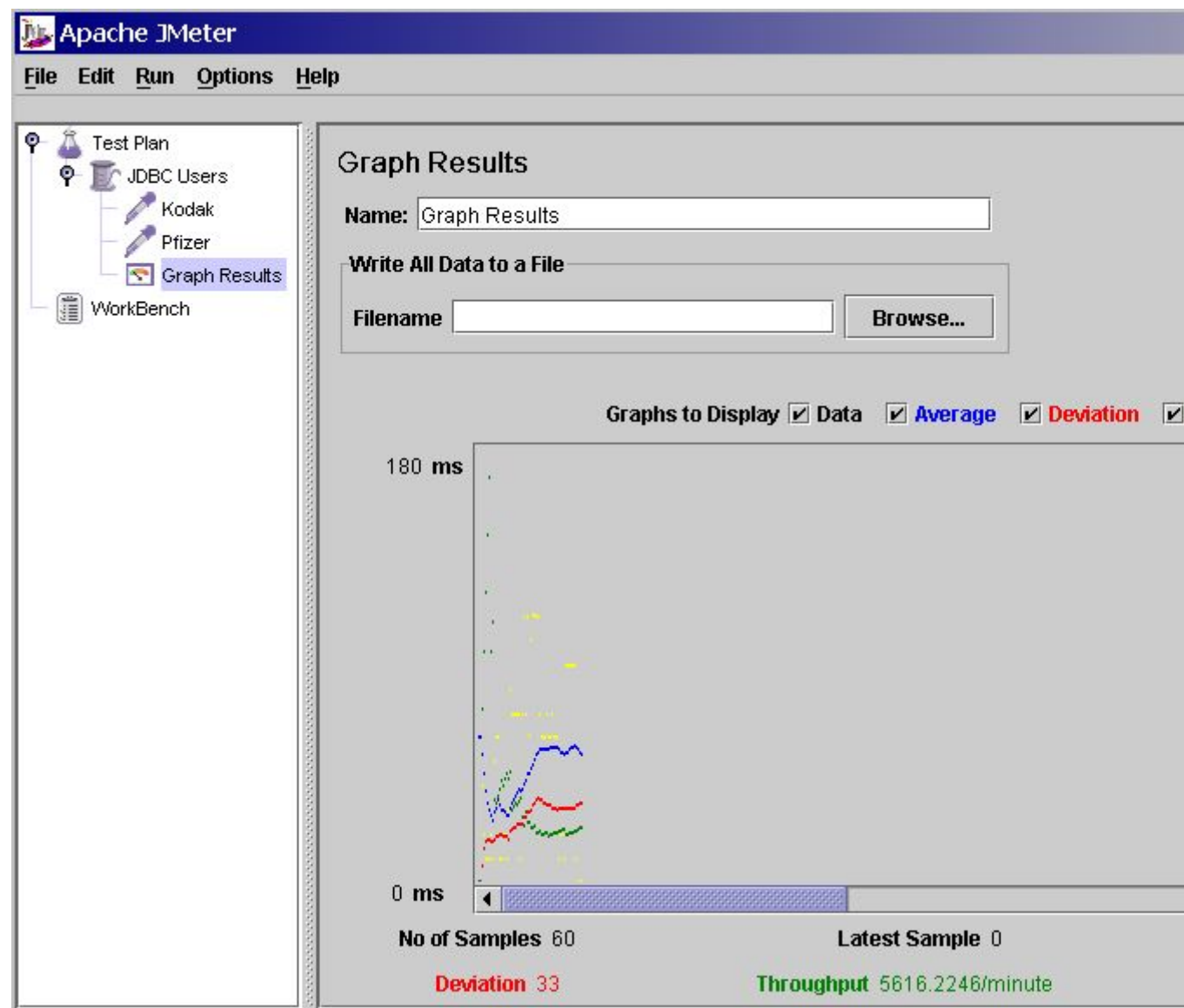


图 7.7. 图像结果监听器

## 8 创建一个 FTP 测试计划

在这章，你将学习到如何创建一个基本的[测试计划](#)来测试 FTP 站点。你将为在 O'Reilly 的 FTP 站点上的两个文件创建四个发送请求的用户。同样，你将告诉

用户运行测试两次。所以整个测试数目是（4 个用户）\*（2 个请求）\*（重复 2 次）=16 个 FTP 请求。为了构造测试计划，你将需要使用下列元件：[测试线程](#)，FTP 请求，FTP 默认请求和 Spline Visualizer。

⚠ 这个例子使用 O'Reilly 的 FTP 站点，[www.oro.com](http://www.oro.com)。当运行这个例子时请考虑周到，并且（如果可能）考虑再次运行其他 FTP 站点。

## 8.1 添加用户

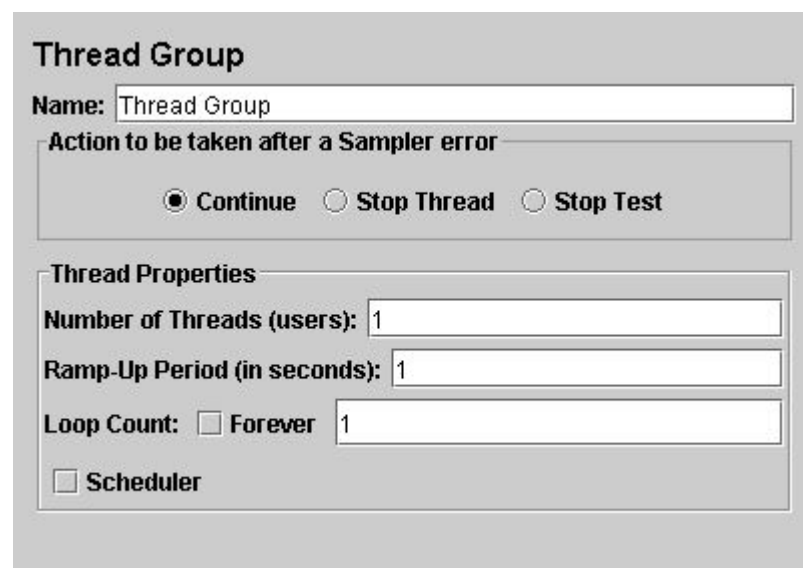
你想处理每个 JMeter 测试计划的第一步是添加线程组元件。线程组告诉 JMeter 你想模拟的用户数，用户发送请求的频率，和发送请求的数量。

顺便说一下，首先选择测试计划，右键点击得到“添加”菜单，并且选择“添加”-->“线程组”，通过这种方式添加线程组。

现在你应该看到了测试计划下的线程组元件了。如果你看不到这个元件，单击测试计划元件展开测试计划树。

下一步，你需要修改默认配置。如果你还没有选择线程组元件，在树里选择它。现在在 JMeter 窗口右部你应该可以看到线程组控制面板。

（见下图 8.1）



**Thread Group**

Name:

Action to be taken after a Sampler error

☒ Continue ☐ Stop Thread ☐ Stop Test

Thread Properties

Number of Threads (users):

Ramp-Up Period (in seconds):

Loop Count: ☐ Forever

☐ Scheduler

图 8.1 使用默认值的线程组

首先给线程组起一个更加有意义的名字。在名称域，输入 O'Reilly Users。

下一步，增加用户数（称作线程）到四个。

在下一个文本域——Ramp-Up Period，使用默认值 0 秒。这个属性告诉 JMeter 启动每个用户之间的时间间隔。例如，你输入 Ramp-Up Period 为五秒，JMeter 将会在最后 5 秒结束前启动所有你的用户。所以，如果我们有 5 个用户和一个 5 秒的 Ramp-Up Period，那么启动用户的延迟就是 1 秒（5 用户/5 秒=1 用户每秒）。如果你设置为那个值为零，那么 JMeter 将会立刻启动所以你的用户。

最后，清除标为“永远”的复选框，并且在循环次数文本域中输入 2。这个属性告诉 JMeter 重复你的测试的次数。如果你输入循环次数为 0，那么 JMeter 将会运行你的测试一次。为了让 JMeter 重复运行你的测试计划，选择永远复选框。

⚠ 在大部分应用程序中，你必须在控制面板中手工改变。然而，在 JMeter 中，控制面板中自动接受你做的改变。如果你修改元件名，这个树会在你离开控制面板前自动使用新的文本更新这个树（例如，当你选择另一个树元件时）。

见图 8.2 完整的 O'Reilly Users 线程组。

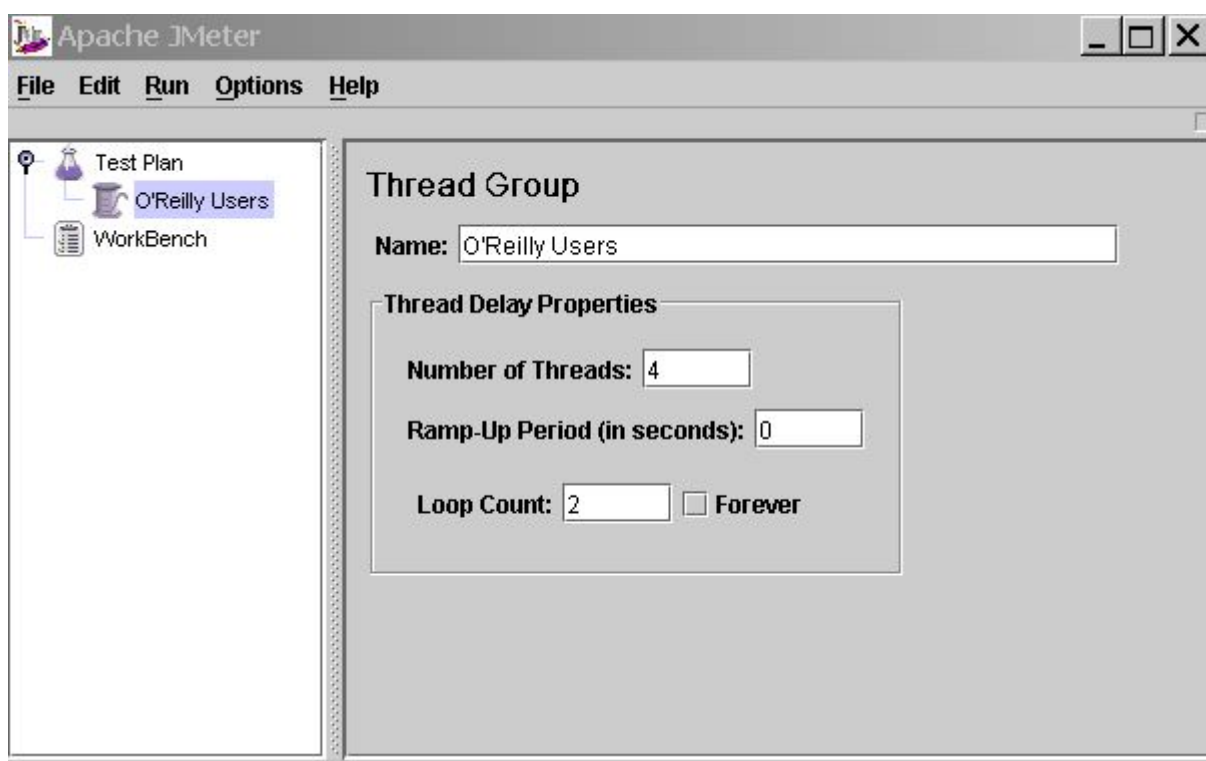


图 8.2 O'Reilly Users 线程组

## 8.2 添加默认 FTP 请求配置

既然我们已经定义了我们的用户，是时候定义他们要执行的任务了。在这一节，你将为你的 FTP 请求指定默认设置。然后在 8.3 节，你将会添加使用你在这里指定的一些默认设置的 FTP 请求元件。

首先选择 O'Reilly Users 元件。右键点击得到“添加”菜单，然后选择“添加”→“配置元件”→“FTP 默认请求”。于是选择新的元件预览它的控制面板（见图 8.3）。

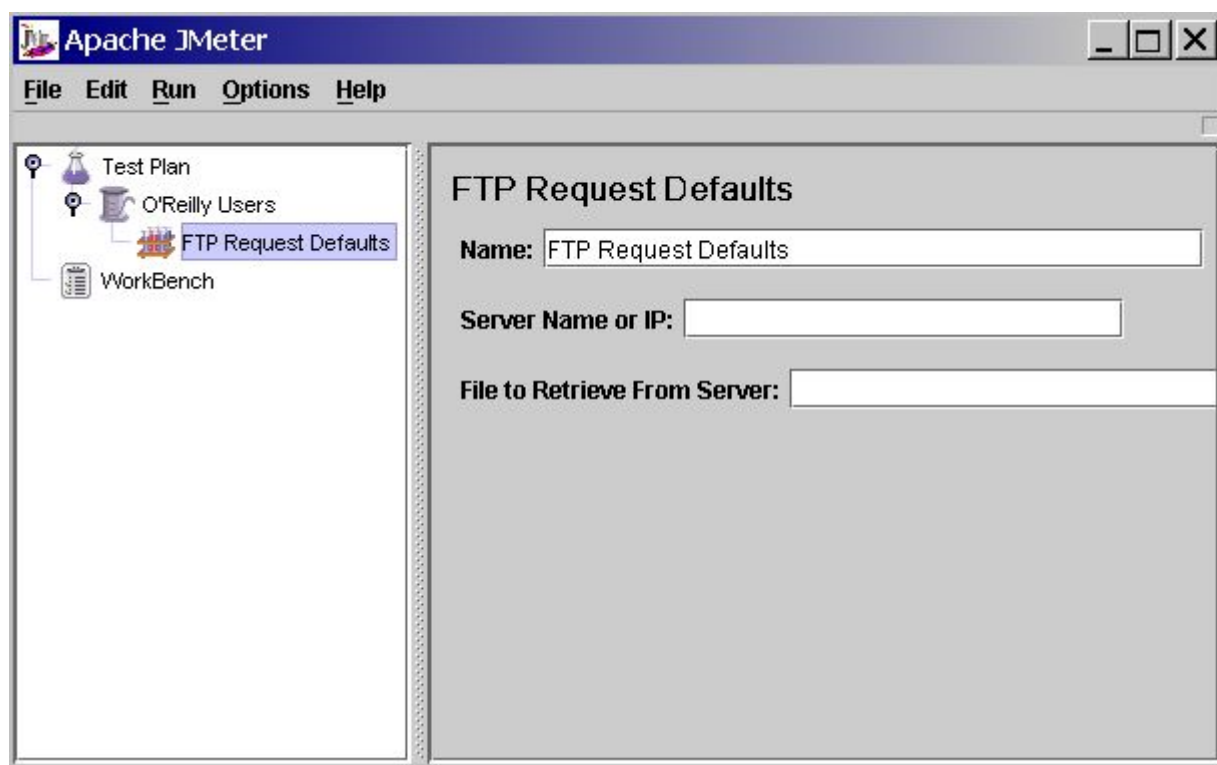


图 8.3 FTP 默认请求

像大多数 JMeter 元件一样，FTP 默认请求控制面板有一个你可以修改的名称域。在这个例子中，保持这个文本域使用默认值。

忽略下一个文本域，它是 FTP 服务器的服务器名/IP。为了你正在构建的测试计划，所有的 FTP 请求将会发送到相同的 FTP 服务器，ftp.oro.com。输入域名到这个文本域。这是唯一一个需要我们去修改它的默认值的文本域，所以保持剩余的文本域使用它们的默认值。

⚠ FTP 默认请求元件没有告诉 JMeter 发送一个 FTP 请求。它只是简单定义了 FTP 请求元件使用的默认值。

见图 8.4 完整的 FTP 默认请求元件。

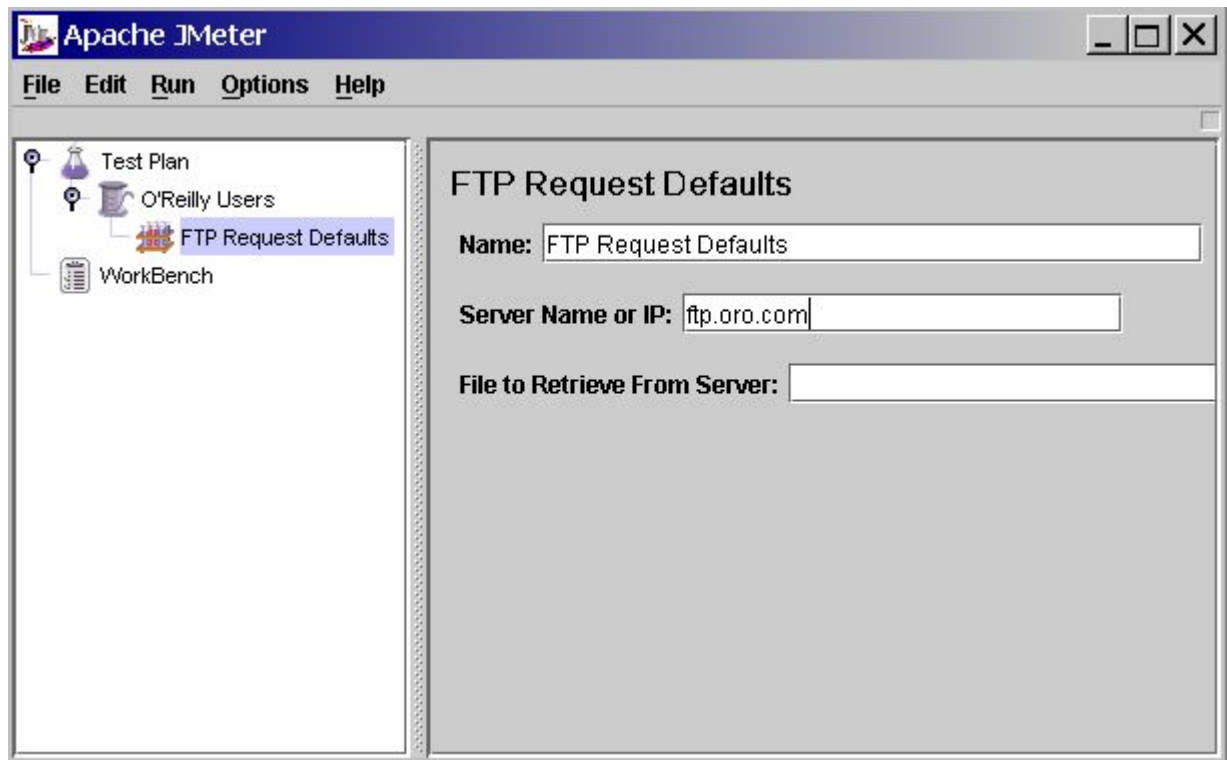


图 8.4 我们测试计划的 FTP 默认

## 8.3 添加 FTP 请求

在我们的测试计划中,我们需要制作两个 FTP 请求。第一个是 O' Reilly 下的 mSQL 下的 java 下 README 文件 (<ftp://ftp.oro.com/pub/mssql/java/README>), 第一个文件是 tutorial 文件 (<ftp://ftp.oro.com/pub/mssql/java/tutorial.txt>)。

⚠ JMeter 按照它们在树中出现的顺序发送请求。

首先添加第一个 FTP 请求到 O' Reilly Users 元件 (添加 --> 取样器 --> FTP 请求)。然后在树中选择 FTP 请求元件, 并且编辑下列属性 (见图 8.5) :

1. 修改名称域为"README"。
2. 修改 File to Retrieve From Server 文本域为"pub/mssql/java/README"。
3. 修改用户名域为"anonymous"。
4. 修改密码域为"anonymous"。

⚠ 因为你已经在 FTP 默认请求元件中指定了服务器名, 所以你不需设置这个值了。

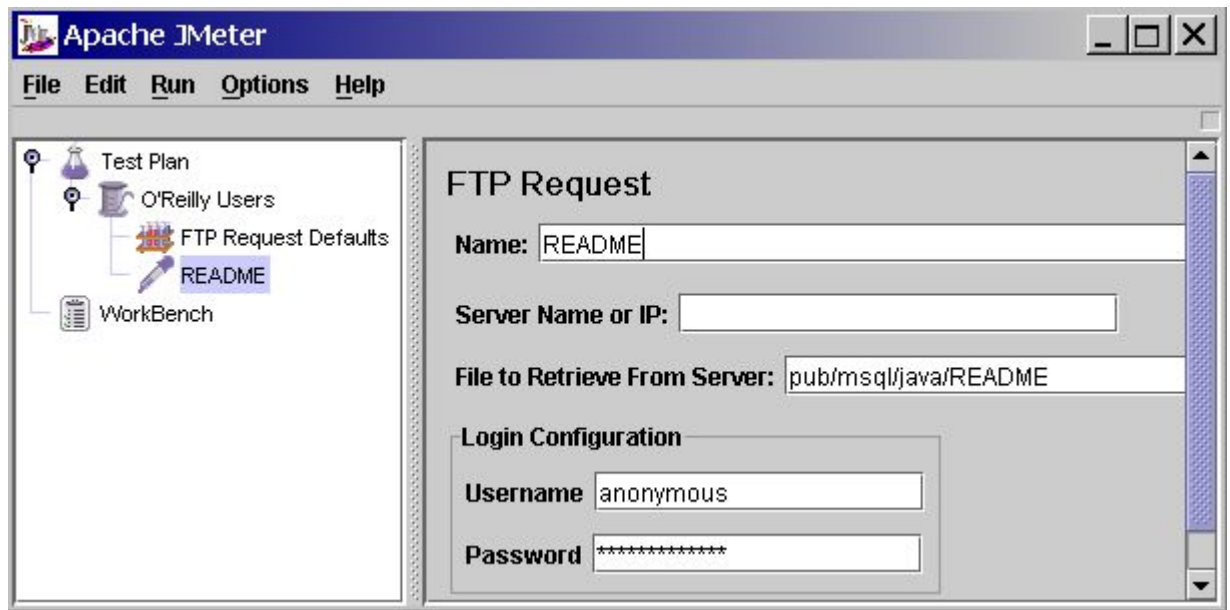


图 8.5 O'Reilly mSQL java README 文件的 FTP 请求

下一步，添加第二个 FTP 请求，并修改下列属性（见图 8.6）：

1. 修改名称域为“tutorial”。
2. 修改 File to Retrieve From Server 文本域为“pub/msql/java/tutorial.txt”。
3. 修改用户名为“anonymous”。
4. 修改密码域为“anonymous”。

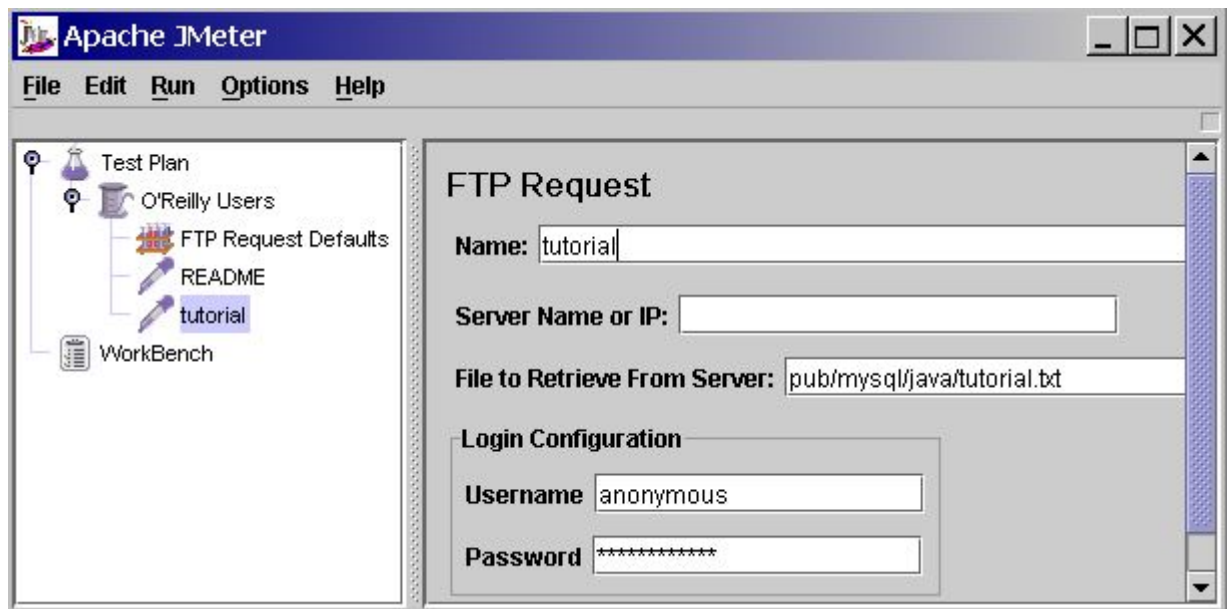


图 8.6 O'Reilly mSQL java tutorial 文件的 FTP 请求

## 8.4 添加一个监听器浏览/保存测试结果

你需要添加到你测试计划的最后元件是一个监听器。这个元件责任是储存所有你的 FTP 请求结果到文件，并且展示一个可视数据模型。

选择 O'Reilly Users 元件，添加一个 Spline Visualizer 监听器（添加 --> 监听器 --> Spline Visualizer）。

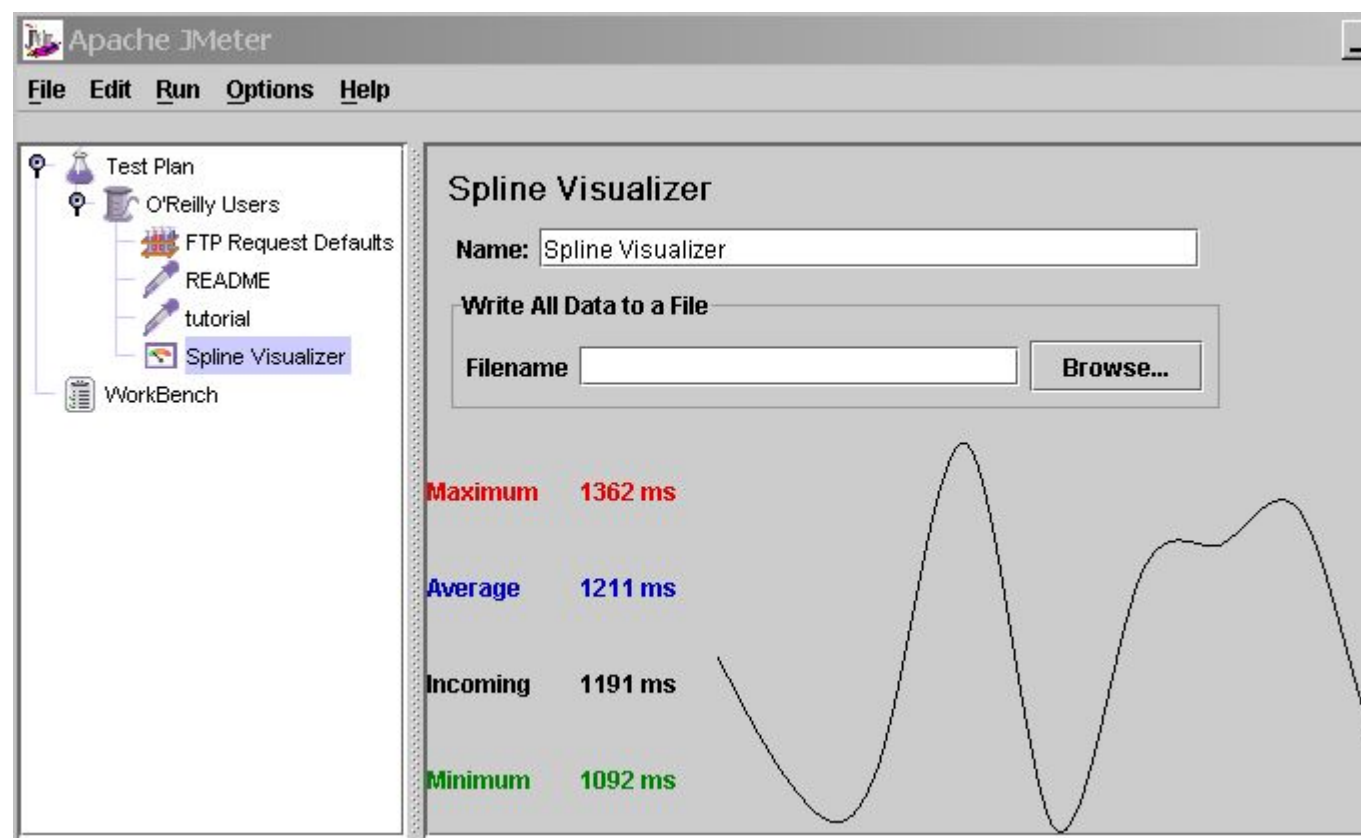


图 8.7 Spline Visualizer 监听器

## 9 构建一个 LDAP 测试计划

在这一节，你将学习到如何创建一个基本的测试计划来测试一个 LDAP 服务器。你将为在 LDAP 上的四个测试创建四个用户发送请求。同样，你要告诉用户运行测试两次。所以，整个请求次数是（4 用户）x（4 请求）x（重复 2 次）=32 LDAP 请求。为了构造测试计划，你将使用下列元件：线程组，LDAP 请求，LDAP 请求默认值和表格视图结果。

这个例子，假定在你的本地机器上已经安装了 LDAP 服务器。



## 9.1 添加用户

你想使用 JMeter 测试计划的第一步是添加一个线程组元件。线程组告诉 JMeter 你想要模拟的用户数，用户多长时间发送一次请求，和它们发送多少个请求。

继续进行，通过首先选择测试计划添加线程组，单击鼠标右键得到“添加”菜单，然后选择“添加”-->“线程组”来添加一个线程组。你现在应该在测试计划下看到线程组。如果你没有看到这个元件，那么通过单击测试计划元件展开测试计划树。

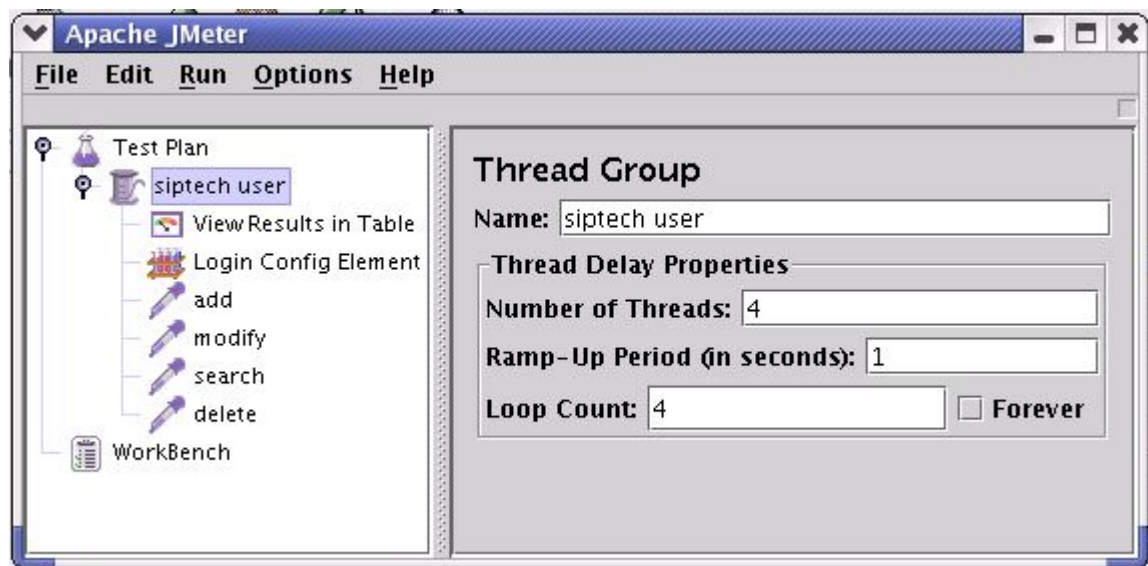


图 9.1 线程组默认值

## 9.2 添加登录配置元件

开始选择 Siptech User 元件。单击鼠标右键得到添加菜单，然后选择“添加”-->“配置元件”-->“登录配置”元件。然后选择这个新元件来查看它的控制面板。

像大多 JMeter 元件一样，登录配置元件控制面板有名称域你可以修改。在这个例子中，保留它为默认值。



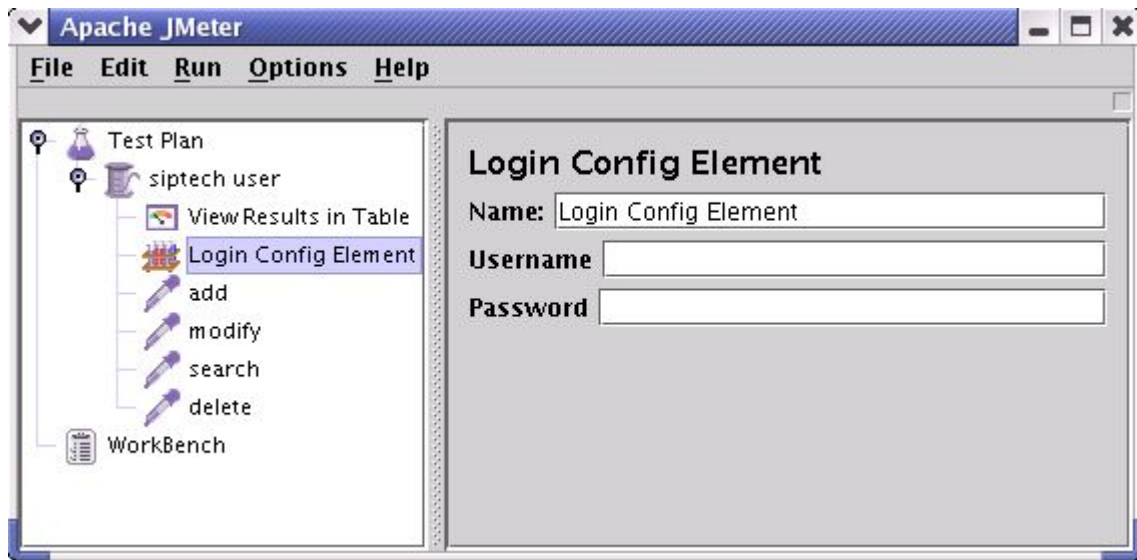


图 9.2 登录配置元件测试计划

### 9.3 添加 LDAP 请求默认值

开始选择 Siptech User 元件。单击鼠标右键得到添加菜单，然后选择“添加”-->“配置元件”-->“LDAP 请求默认值”。选择这个新元件来查看它的控制面板。

像大多 JMeter 元件一样，LDAP 请求默认值控制面板有名称域你可以修改。在这个例子中，保留它为默认值。

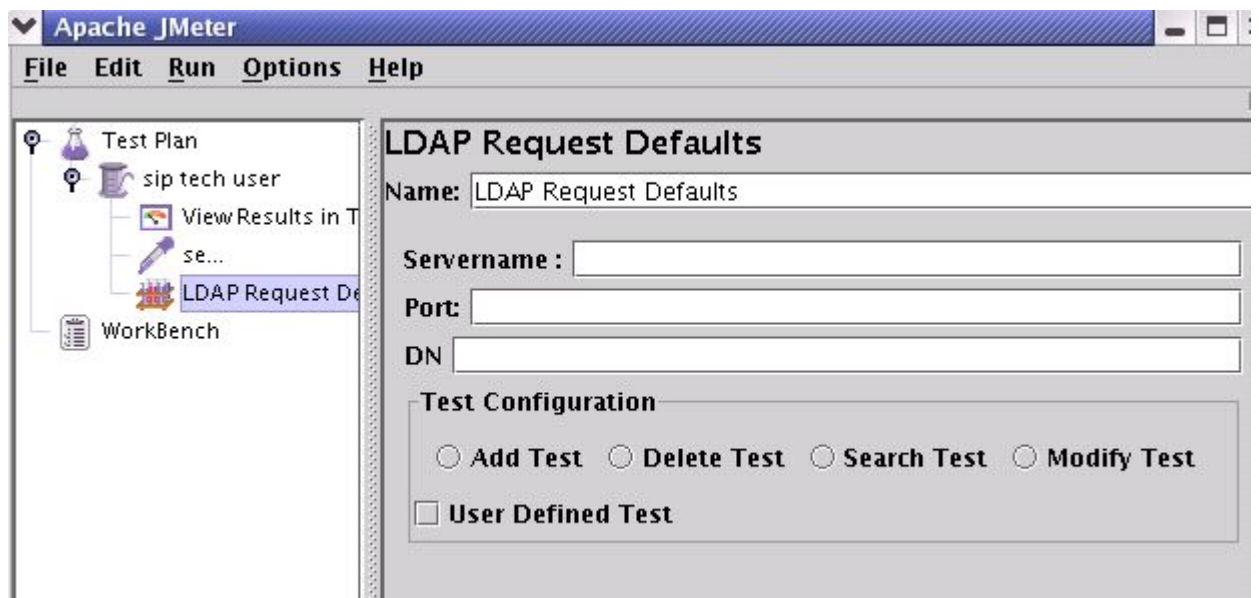


图 9.3 LDAP 请求默认值测试计划



在 DN 域输入“你服务器的根 DN”。  
在 LDAP 服务器的服务器名域输入“localhost”。

端口为 389。  
那些就是 LDAP 请求的默认值。

## 9.4 添加 LDAP 请求

在我们测试计划我们需要准备四个 LDAP 请求。

1. Inbuilt Add Test
2. Inbuilt Modify Test
3. Inbuilt Delete Test
4. Inbuilt Search Test

JMeter 以添加它们到树的顺序发送请求。开始添加第一个 LDAP 请求到 Siptech User 元件（添加-->LDAP 请求）。然后，在树中选择 LDAP 请求元件，编辑下列属性

1. 更改名称为 Inbuilt-Add Test
2. 选择添加测试单选按钮

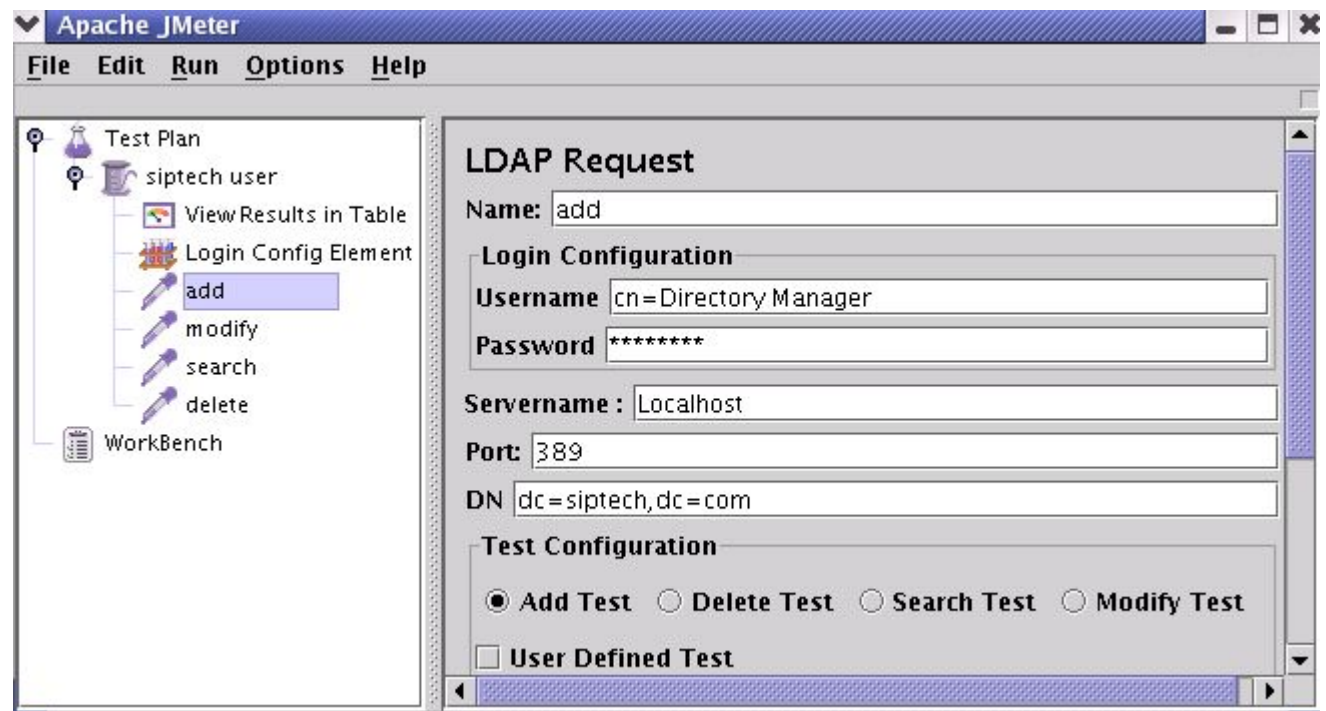


图 9.4.1 Inbuilt Add test LDAP 请求

你不需要设置服务器域和端口域,用户名,密码和 DN,因为你已经在 Login Config Element 和 LDAP 请求默认值中指定了。

下一步，添加第二个 LDAP 请求，编辑下列属性

1. 更改名称为 Inbuilt-Modify Test
2. 选择修改测试单选按钮



图 9.4.2 Inbuilt Modify test LDAP 请求

1. 更改名称为 Inbuilt-Delete Test
2. 选择删除测试单选按钮



图 9.4.3 Inbuilt-Delete Test LDAP 请求

1. 更改名称为 Inbuilt-Search Test
2. 选择搜索测试单选按钮

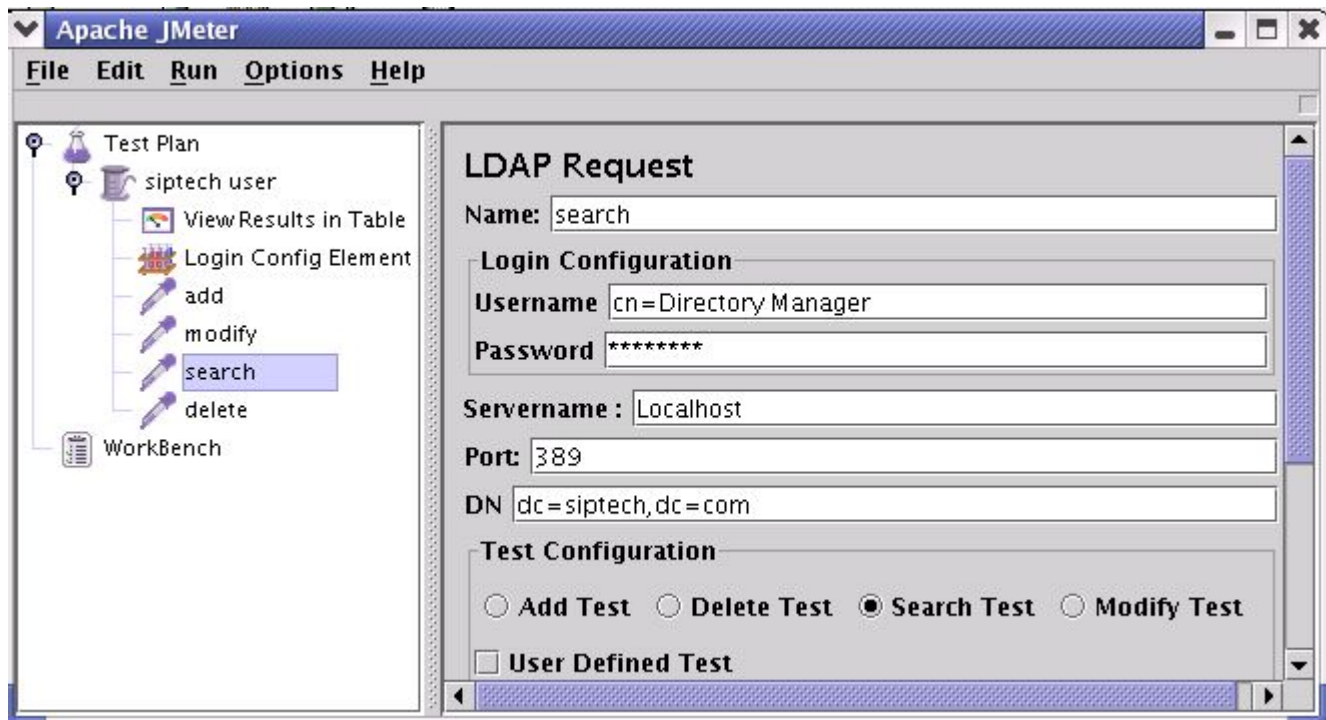


图 9.4.4 Inbuilt-Search Test LDAP 请求

## 9.5 添加一个监听器浏览/保存测试结果

你需要添加到你测试计划的最后元件是一个监听器。这个元件责任是保存所有你的 LDAP 请求结果到一个文件，并且显示一个可视化数据模型。选择 Siptech Users 元件，添加一个表格视图结果（添加—>表格视图结果）。

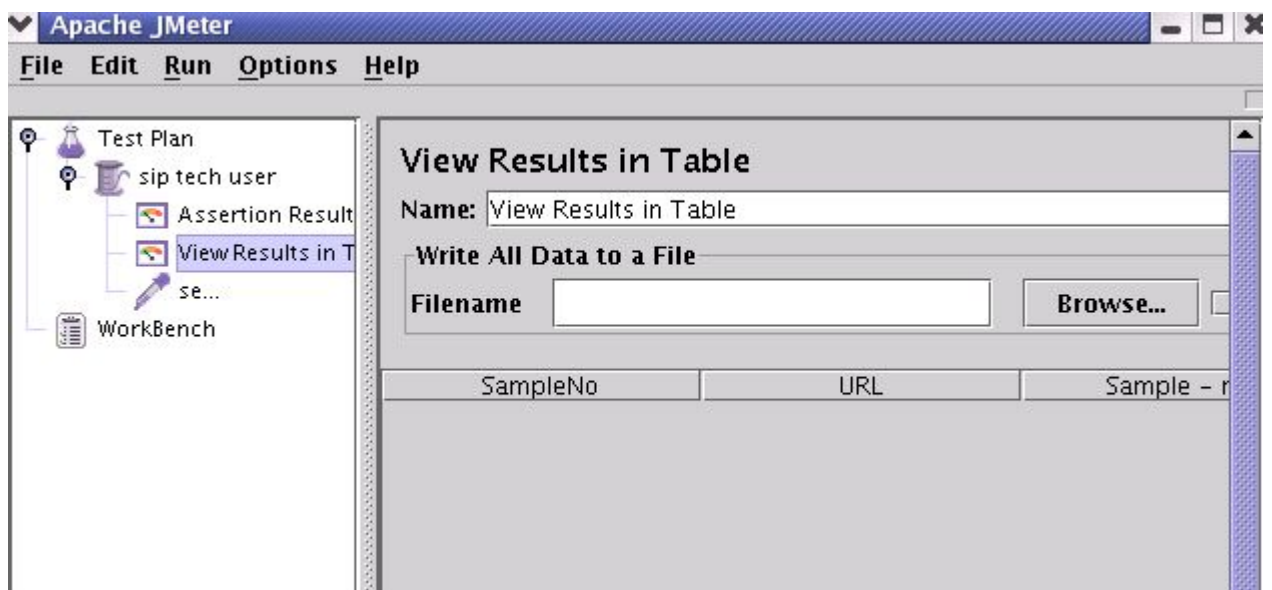



图 9.5 表格视图结果监听器


## 9.6 保存测试计划

虽然它不是需要的，但是我们推荐你在运行前保存测试计划到一个文件。为了保存测试计划，从文件菜单选择保存测试计划（使用最新版本，它不再需要首先选择测试计划元件）。

 JMeter 允许你保存这个测试计划树或者仅仅其中一部分。为了仅保存在测试计划树上的特殊“分支”，选择在树中用来启动“分支”的测试计划元件，然后右击在菜单项中选择“保存”。或者，选择合适测试计划元件，然后从编辑菜单选择保存。

## 9.7 运行测试计划

从运行菜单，选择运行。

 如果你测试正在运行，JMeter 在右手上方的角落点亮一个绿正方形显示。当所有测试停止，那个方块变成灰色。即使你选择了“停止”，绿光依然会继续持续，直到所有测试都已经退出。

# 10 构建一个 Web 服务测试计划

在这章，你将学习如何创建一个测试 web 服务的测试计划。你将创建五个发送请求到一个页面的用户。同时，你将告诉用户运行他们的测试两次。所以整个请求是（5 用户）\*（1 请求）\*（重复 2 次）=10HTTP 请求。为了构造测试计划，你将需要使用以下元件：测试计划、Web 服务（SOAP）请求（beta 版代码）和图表结果。

General notes on the webservicessampler. 现在实现使用 Apache SOAP 驱动程序，需要来自 sun 的 activation.jar 和 mail.jar 包。由于协议限制，JMeter 没有包含这些 jar 文件到二进制版本。请查阅 SOAP 文档的未来细节。

如果取样器表现出从 web 服务中得到一个错误，仔细检查 SOAP 消息，确认格式正确。细节方面，确认 xmlns 属性和 WSDL 是一样的。如果 xml 命名空间是不同的，web 服务将会可能返回一个错误。Xmethods 为那些想要测试他们的测试计划的人包含了一系列公用的 web 服务。

## 10.1 添加用户

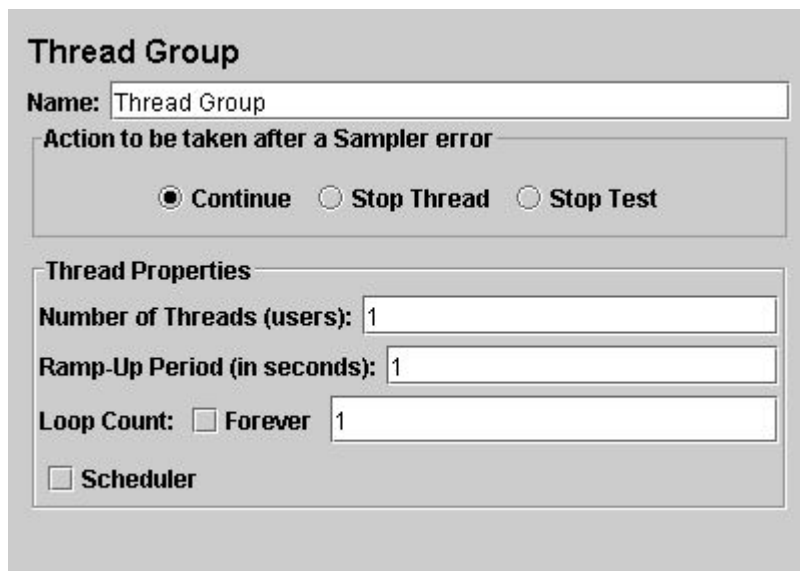
你想处理每个 JMeter 测试计划的第一步是添加线程组元件。线程组告诉 JMeter 你想模拟的用户数，用户发送请求的频率，和发送请求的数量。

顺便说一下，首先选择测试计划，右键点击得到 Add 菜单，并且选择 Add->ThreadGroup，通过这种方式添加线程组。

现在你应该看到了测试计划下的线程组元件了。如果你看不到这个元件，单击测试计划元件展开测试计划树。

下一步，你需要修改默认配置。如果你还没有选择线程组元件，在树里选择它。现在在 JMeter 窗口右部你应该可以看到线程组控制面板。

(见下 10.1)



**Thread Group**

Name:

Action to be taken after a Sampler error

☒ Continue ☐ Stop Thread ☐ Stop Test

Thread Properties

Number of Threads (users):

Ramp-Up Period (in seconds):

Loop Count: ☐ Forever

☐ Scheduler

图 10.1 使用默认值的线程组

首先给线程组起一个更加有意义的名字。在 name 文本域，输入 O'Reilly Users。

先一步，增加用户数（调用线程）到四个。

在下一个文本域——Ramp-Up Period，使用默认值 0 秒。这个属性告诉 JMeter 启动每个用户之间的时间间隔。例如，你输入 Ramp-Up Period 为五秒，JMeter 将会在最后 5 秒结束前启动所有你的用户。所以，如果我们有 5 个用户和一个 5 秒的 Ramp-Up Period，那么启动用户的延迟就是 1 秒(5 用户/5 秒=1 用户每秒)。如果你设置为那个值为零，那么 JMeter 将会立刻启动所以你的用户。

最后，清除标为“Forever”的复选框，并且在循环次数文本域中输入 2。这个属性告诉 JMeter 重复你的测试的次数。如果你输入循环次数为 0，那么 JMeter 将会运行你的测试一次。为了让 JMeter 重复运行你的测试计划，选择 Forever 复选框。



⚠ 在大部分应用程序中，你必须在控制面板中手工改变。然而，在 JMeter 中，控制面板中自动接受你做的改变。如果你修改元件名，这个树会在你离开控制面板前自动使用新的文本更新这个树（例如，当你选择另一个树元件时）。

见图 10.2 完整的 Jakarta Users 线程组。

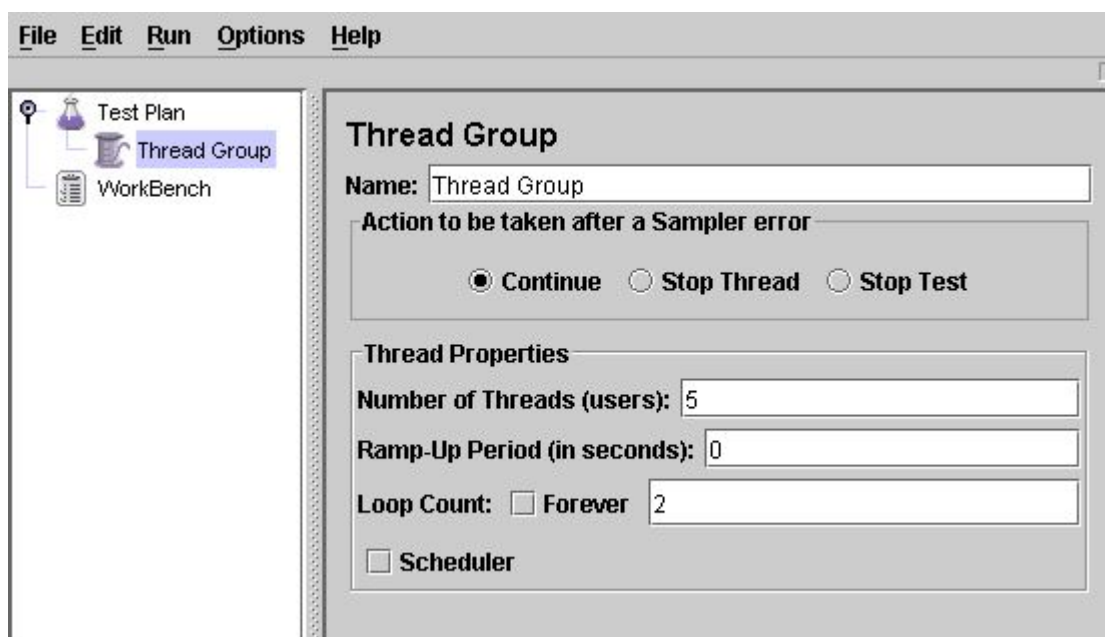


图 10.2 Jakarta Users 线程组

## 10.2 添加 web 服务请求

在我们的测试计划，我们将使用一个 .NET web 服务。自从你在使用 web 服务取样器，我们将不用深究写一个 web 服务的细节。如果你不知道如何写一个 web 服务，使用 google 搜索 web 服务 并自己去熟悉写 java 和 .NET 的 web 服务。应该注意的是 .NET 和 Java 的 web 服务实现有很大的不同。这个主题包含在用户手册太宽了。请参考其他源代码来得到更好的不同之处的概念。

⚠ JMeter 发送请求是以它们出现在书中的顺序。

开始添加 Web 服务 (SOAP) 请求 (Beta 代码) 取样器到 Jakarta Users 元件（添加-->取样器-->Web 服务 (SOAP) 请求 (Beta 代码)）。然后，在树中选择 web 服务请求元件并设置下列属性（见图 10.5）：

1. 改变名称文本域为“WebService(SOAP) Request (Beta Code)”。
2. 输入 WSDL 地址并点击“加载 WSDL”。

## WebService(SOAP) Request

Name:

WSDL URL

**Load WSDL**

Web Methods



**Configure**

Protocol:

Server Name or IP:

Port Number:

Path:

SOAPAction

Soap/XML-RPC Data

**File with SOAP XML Data (overrides above text)**

Filename

**Browse...**

Note: Parsing XML is CPU intensive. Therefore, do not set the thread count too high. In general, 10 threads will consume 100% of the CPU on a 900mhz Pentium 3. On a pentium 4 2.4ghz cpu, 50 threads is the upper limit. Your options for increasing the number of clients is to increase the number of machines or use multi-cpu systems.

Message Folder

☒ **Memory Cache**

☐ **Read SOAP Response**

If read response is unchecked, the sampler will not read the response or set the SampleResult. This improves performance, but it means the response content won't be logged.

☐ **Use HTTP Proxy**

Proxy Host

Proxy Port

If Use HTTP Proxy is checked, but no host or port are provided, the sampler will look at command line options. If no proxy host or port are provided by either, it will fail silently.



图 10.3 Web 服务请求

如果 WSDL 文件加载正确，“Web 方法”下拉框将会增加元素。如果下拉框仍然为空，它说明得到 WSDL 文件有问题。你可以使用浏览器读取 XML 测试 WSDL。例如，如果你测试一个 IIS web 服务，URL 会像这样：

<http://localhost/myWebService/Service.asmx?WSDL>。在这点，“SOAPAction”，“URL”，“SOAPData”应该为空。

下一步，选择 web 方法并点击“配置”。取样器应该会填充“URL”和“SOAPAction”文本域。加入 WSDL 文件有效，正确的 SOAPAction 会输入。

最后一步是在“SOAP/XML-RPC”文本域粘贴 SOAP 消息。

## 11. 构建一个 JMS 点对点测试计划

JMS：JMS（Java Messaging Service）是 Java 平台上有关面向消息中间件(MOM)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。

在本节中，你将学会如何创建一个测试计划来测试 JMS 点对点的解决方案。测试的建立是一个有五个线程的线程组，通过每个请求队列发送 4 个消息。一个固定的回复队列将用来监听应答消息。每个 1 到 10 次迭代。构建测试计划，你将使用下列元件：线程组，JMS 点对点图形结果。

大概介绍一下 JMS。现在有两种 JMS 取样器。一个使用 JMS 主题，另一个使用队列。主题消息通常被称作发布/订阅消息。它一般使用的情况是一个生产者发布消息，多个订阅者来消费。

### 11.1 添加一个线程组

你想使用 JMeter 测试计划的第一步是添加一个线程组元件。线程组告诉 JMeter 你想要模拟的用户数，用户多长时间发送一次请求，和它们发送多少个请求。

继续进行，通过初次的选择测试计划添加线程组，单击鼠标右键得到一个菜单，然后选择添加—>线程组来添加一个线程组。

你现在应该在测试计划下看到了线程组。如果你没有看到这个元件，然后通过单击测试计划元件展开测试计划树。

下一步，你需要修改默认属性。如果你还没有选择线程组元件，那么在这个树中选择它。你现在应该在 JMeter 窗口的右边部分看到了线程组控制面板。（见下图：11.1）

**Thread Group**

Name:

Action to be taken after a Sampler error

☒ Continue ☐ Stop Thread ☐ Stop Test

Thread Properties

Number of Threads (users):

Ramp-Up Period (in seconds):

Loop Count: ☐ Forever

☐ Scheduler


图 11.1 使用默认值的线程组

开始，为我们的线程组提供一个更加有描述性的名字。在 name 域，输入 Point-to-Point。

下一步，增加用户数（即线程）到 5。

在下一个域中，Ramp-Up 周期，保持默认值 0 秒。这个属性告诉 JMeter 启动每个用户之间有多长延迟。例如，如果你输入 Ramp-up 周期为 5 秒，JMeter 会到 5 秒末完成启动所有你的用户。所以如果我们有五个用户和一个 5 秒的 Ramp-up 周期，那么启动用户之间的延迟将会是 1 秒（5 用户 / 5 秒=1 用户每秒）。如果你设置为那个值为零，那么 JMeter 将会立刻启动所以你的用户。

最后，清除标为“Forever”的复选框，并且在循环次数域中输入 4。这个属性告诉 JMeter 重复你的测试的次数。如果你输入循环次数为 0，那么 JMeter 将会运行你的测试一次。为了让 JMeter 重复运行你的测试计划，可以选择 Forever 复选框。

 在大部分应用程序中，你必须在控制面板中手工改变。然而，在 JMeter 中，控制面板中自动接受你做的改变。如果你修改元件名，这个树会在你离开控制面板前自动使用新的文本更新这个树（例如，当你选择另一个树元件时）。

## 11.2 添加点对点取样器

确认你需要的 jar 文件在 JMeter 的 lib 目录下。如果它们不在，停止 JMeter，拷贝 jar 文件过去，然后重启 JMeter。

开始添加 JMS 点对点取样器到 Jakarta 用户元件（添加-->JMS 点对点）。然后，在树中选择 JMS 点对点取样器元件。在构建例子中将提供一个使用 ActiveMQ3.0 工作的配置。

### 11.3 添加一个监听器浏览/保存测试结果

你需要添加到你测试计划的最后元件是一个监听器。这个元件责任是保存所有你的 HTTP 请求结果到一个文件，并且显示一个可视化数据模型。

选择 Jakarta Users 元件，添加一个图形结果监听器（添加-->图形结果）。下一步，你需要指定一个目录和一个输出文件名。你可以，选择浏览按钮，浏览一个目录，然后输入一个文件名。



图 11.2 图形结果监听器

### 11.4 保存测试计划

虽然它不是需要的，但是我们推荐你在运行前保存测试计划到一个文件。为了保存测试计划，从文件菜单选择保存测试计划（使用最新版本，它不再需要首先选择测试计划元件）。



JMeter 允许你保存这个测试计划树或者仅仅其中一部分。为了仅保存在测试计划树上的特殊“分支”，选择在树中用来启动“分支”的测试计划元件，然后右击在菜单项中选择“保存”。或者，选择合适测试计划元件，然后从编辑菜单选择保存。

## 11.5 运行测试计划

从运行菜单，选择运行。



如果你测试正在运行，JMeter 在右手上方的角落点亮一个绿正方形显示。当所有测试停止，那个方块变成灰色。即使你选择了“停止”，绿光依然会继续持续，直到所有测试都已经退出。

一旦 JMeter 完成你的测试计划，从运行菜单选择停止。

如果你在监听器中选择一个文件保存结果，那么你将会有一个能够在任何 visualizer 中打开的文件。每个 visualizer 以它们自己的风格显示结果。



有可能会在多于一个的 visualizer 中打开同一个文件。这是没有问题的。JMeter 会确保在测试运行时没有取样器记录到同一文件多于一次。

## 11.6 ActiveMQ3.0 的类库

下面是必须在 JMeterlib\ext 目录提供的类库。

1. activation.jar
2. activeio-1.0-SNAPSHOT.jar
3. activemq-3.0.jar
4. activemq-core-3.0.jar
5. commons-logging-1.0.3.jar
6. concurrent-1.3.4.jar
7. geronimo-spec-j2ee-jacc-1.0-rc4.jar
8. geronimo-spec-j2ee-management-1.0-rc4.jar
9. geronimo-spec-jms-1.1-rc4.jar
10. geronimo-spec-jta-1.0.1B-rc4.jar
11. jms.jar
12. jndi.jar
13. log4j-1.2.8.jar

## 12. 创建 JMS 主题测试计划

在这章，你将学习如何创建一个测试计划去测试 JMS 提供者。你将创建五个订阅者和一个发布者。你将创建两个线程组并且设置一个为重复 10 次。消息总数是  $(6 \text{ 线程}) \times (1 \text{ 消息}) \times (\text{重复 } 10 \text{ 次}) = 60 \text{ 个消息}$ 。为了构造测试计划，你将使用以下元件：线程组、JMS 发布者、JMS 订阅者和图标结果。

一般在。当前有两个 JMS 取样器。一个使用 JMS 主题，另一个是使用 JMS 队列。主题消息是通常说的发布/订阅消息。在案例里它一般用在一个被生产者发布消息和多个订阅者接收消息的地方。队列消息一般被用在发送者期望得到一个响应时的事务。消息系统和普通的 HTTP 请求有很大不同。在 HTTP 中，单个用户发送一个请求并且得到一个响应。消息系统可以工作在同步和异步模式。

### 12.1 添加用户

第一步是添加线程组元件。线程组告诉 JMeter 你想要模拟的用户数，用户多久发送一次请求，它们发送多少请求。

接着首先选择测试计划添加线程组元件，单击鼠标右键得到 Add 菜单，并且选择 Add --> ThreadGroup。

你现在可以在测试计划下看到线程组元件。如果看不到这个元件，然后通过单击测试计划元件“展开”测试计划树。

下一步，你需要修改默认属性。如果你没有选择在树中的线程组，就选择它。你现在可以在 JMeter 窗口右部分看到线程组控制面板（见下 12.1）。

**Thread Group**

Name:

Action to be taken after a Sampler error

☒ Continue ☐ Stop Thread ☐ Stop Test

Thread Properties

Number of Threads (users):

Ramp-Up Period (in seconds):

Loop Count: ☐ Forever

☐ Scheduler


图 12.1 具有默认值的线程组

开始为线程组提供一个更有描述性的名字。在这个 name 文本域，输入 Subscribers。

下一步，增加用户数（叫做线程）到 5。

在下一个文本域——Ramp-Up Period，使用默认值 0 秒。这个属性告诉 JMeter 启动每个用户之间的时间间隔。例如，你输入 Ramp-Up Period 为五秒，JMeter 将会在最后 5 秒结束前启动所有你的用户。所以，如果我们有 5 个用户和一个 5 秒的 Ramp-Up Period，那么启动用户的延迟就是 1 秒（5 用户/5 秒=1 用户每秒）。如果你设置为那个值为零，那么 JMeter 将会立刻启动所以你的用户。

最后，清除标为“Forever”的复选框，并且在循环次数文本域中输入 2。这个属性告诉 JMeter 重复你的测试的次数。如果你输入循环次数为 0，那么 JMeter 将会运行你的测试一次。为了让 JMeter 重复运行你的测试计划，选择 Forever 复选框。

 在大部分应用程序中，你必须在控制面板中手工改变。然而，在 JMeter 中，控制面板中自动接受你做的改变。如果你修改元件名，这个树会在你离开控制面板前自动使用新的文本更新这个树（例如，当你选择另一个树元件时）。

## 12.2 添加 JMS 订阅者和发布者

确认在 JMeter 的 lib 文件夹下有需要的 jar 包。如果没有，关闭 JMeter，拷贝 jar 文件过去，重启 JMeter。

开始添加 JMS Subscriber 取样器到 Jakarta Users 元件 (Add --> Sampler --> JMS Subscriber)。然后，在树中选择 JMS Subscriber 元件，并且编辑下列属性：

1. 改变 Name 域为“sample subscriber”
2. 如果 JMS 提供者使用 jndi.properties，选择这个复选框
3. 输入 InitialContextFactory 的类名
4. 输入提供者 URL，
5. 输入连接工厂名。请参考 JMS 提供者的文档信息
6. 输入消息主题名
7. 如果 JMS 提供者需要认证，选择“required”并且输入用户名和密码。例如，Orion JMS 需要认证，然而 ActiveMQ 和 MQSeries 不需要
8. “ActiveMQ and MQSeries”中输入 10. 因为性能原因，the sampler will aggregate messages, since small messages will arrive very quickly. If the sampler didn't aggregate the messages, JMeter wouldn't be able to keep up.
9. 如果你需要读取响应，选择这个复选框
10. There are two client implementations for subscribers. If the JMS provider exhibits zombie threads with one client, try the other.

**JMS Subscriber**

Name:

☒ Use jndi.properties file

JNDI Initial Context Factory

Provider URL

Connection Factory

Topic

Authentication ☐ Required ☒ Not Required

User

Password

Number of samples to aggregate

☐ Read Response

Client ☒ Use TopicSubscriber.receive() ☐ Use MessageListener.onMessage()

图 12.2 JMS Subscriber

1. 改变 Name 域为“sample publisher”

2. 如果 JMS 提供者使用 `jndi.properties`, 选择这个复选框
3. 输入 `InitialContextFactory` 的类名
4. 输入提供者 URL,
5. 输入连接工厂名。请参考 JMS 提供者的文档信息
6. 输入消息主题名
7. 如果 JMS 提供者需要认证, 选择“required”并且输入用户名和密码。例如, Orion JMS 需要认证, 然而 ActiveMQ 和 MQSeries 不需要
8. “ActiveMQ and MQSeries”中输入 10. 因为性能原因, the sampler will aggregate messages, since small messages will arrive very quickly. If the sampler didn't aggregate the messages, JMeter wouldn't be able to keep up.
9. Select the appropriate configuration for getting the message to publish. If you want the sampler to randomly select the message, place the messages in a directory and select the directory using browse.
10. Select the message type. If the message is in object format, make sure the message is generated correctly.



**JMS Publisher**

Name:

☒ Use jndi.properties file

JNDI Initial Context Factory

Provider URL

Connection Factory

Topic

Authentication ☐ Required ☒ Not Required

User

Password

Number of samples to aggregate

Configuration ☒ From file ☐ Random File ☐ Textarea

Message Type ☒ Text Message ☐ Object Message

**File**

Filename

**Random File**

Filename

**Text Message**

图 12.3. JMS Publisher

## 12.3 添加一个监听器浏览/保存测试结果

你需要添加到你测试计划的最后元件是一个监听器。这个元件责任是储存所有你的 HTTP 请求结果到文件，并且展示一个可视数据模型。

选择 Jakarta Users 元件，添加一个 Graph Resultsr 监听器（Add --> Listener --> Graph Results）。Next, you need to specify a directory and filename of the output file. You can either type it into the filename field, or select the Browse button and browse to a directory and then enter a filename.



图 12.4 Graph Results 监听器

## 12.4 保存测试计划

虽然它不是需要的，但是我们推荐你在运行前保存测试计划到一个文件。为了保存测试计划，从 File 菜单选择 Save Test Plan（使用最新版本，它不再需要首先选择测试计划元件）。

⚠ JMeter 允许你保存这个测试计划树或者其中一部分。为了仅保存在测试计划树上的特殊“分支”，选择在树中用来启动“分支”的测试计划元件，然后右击在菜单项中选择“Save”。或者，选择合适测试计划元件，然后从 Edit 菜单选择 Save。

## 12.5 运行测试计划

从 Run 菜单，选择 Run。



如果你测试正在运行，JMeter 在右手上方的角落点燃一个绿正方形显示。当所有测试停止，那个方块变成灰色。即使你选择了“stop”，绿光依然会继续停留，知道所有测试都已经停止。

一旦 JMeter 完成你的测试计划，从 Run 菜单选择 Stop。

如果你在你的监听器中选择一个文件保存结果，然后你将有一个能够在任何可视化工具下打开的文件。

每一可视化工具会使用它自己的风格去显示结果。



如果可能在多个可视化工具中打开同一个文件。这是不是问题。JMeter 会保证在测试运行期间没有取样会再次被记录于同一文件。

## 13 构建一个监视器测试计划

在这一节，你讲学习如何创建一个测试计划来监视 web 服务器。监视器对一个压力测试和系统管理很有用。使用压力测试，监视器可以提供一些关于服务器性能的附加信息。它也会使看出服务器性能和在客户端响应时间直接的关系更加容易。作为一个系统管理工具，监视器提供很容易的方法从一个控制台监视多个服务器。监视器被设计和 Tomcat 5 下的状态 servlet 一起运行。理论上说任何支持 JMX (Java Management Extension) 的 servlet 容器都可以指定状态 servlet 来提供同样的信息。

因为一些人要使用伴随其他 servlet 或者 EJB 容器的监视器，Tomcat 的状态 servlet 将不要做任何修改工作在其他的容器来进行内存统计。为了得到线程信息，你将需要改变 MBeanServer 的查找来从新得到 MBeans。

### 13.1 添加一个服务器

第一步添加一个线程组元件。线程组告诉 JMeter 你想要模拟的用户数。因为使用 JMeter 作为监听器，所以总是为 1. 对于那些不熟悉服务器监视器的人这是非常重要的。作为一个一般规则，对于单个服务器使用多个线程是严重的并且会造成重大的压力。

继续进行，通过初次的选择测试计划添加线程组，单击鼠标右键得到添加菜单，然后选择添加-->线程组。

你现在应该在测试计划下看到线程组。如果你没有看到这个元件，那么通过单击测试计划元件展开测试计划树。

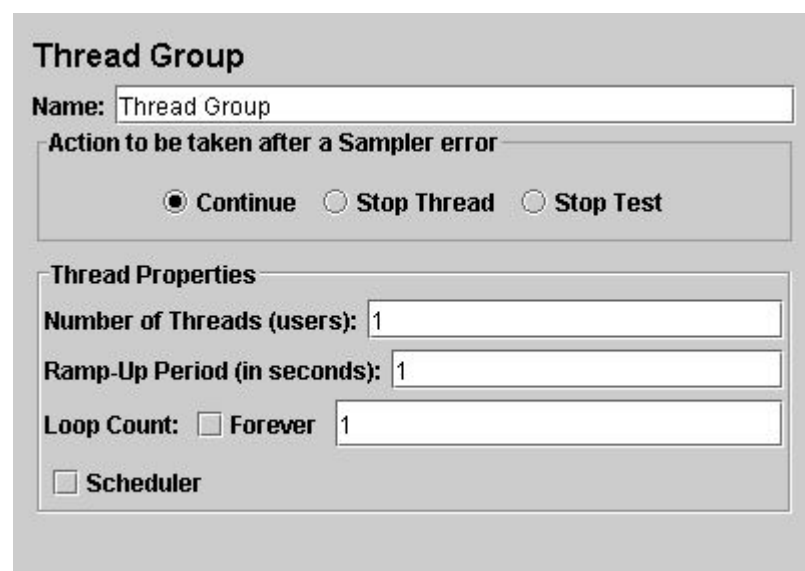


图 10.1 使用默认值的线程组

输入服务器名。

## 13.2 HTTP 认证管理

添加 HTTP 授权管理器到线程组元件（添加-->HTTP 授权管理器）。输入你的 web 服务器的用户名和密码。重要提示：监视器仅能工作于 Tomcat5 的 5.0.19 和更新版本。如何建立 Tomcat 的相关说明，情参考 tomcat 5 文档。

1. 保持基础 URL 为空
2. 输入用户名
3. 输入密码

## 13.3 添加 HTTP 请求

添加 HTTP 请求到线程组元件（添加-->取样器-->HTTP 请求）。然后，在树中选择 HTTP 请求元件，并编辑下列属性：

1. 修改名称域为“Server Status”
2. 输入 IP 地址和主机名
3. 输入端口号
4. 如果使用 Tomcat，设置 Path 域为“/manager/status”

5. 添加名为大写的“XML”请求参数，给它一个小写的“true”值
6. 选择取样器的底部“Use as Monitor”

## 13.4 添加固定定时器

添加一个定时器到这个线程组（添加-->定时器-->固定定时器）。在“线程延迟”方框输入 5000 毫秒。一般使用间隔少于 5 秒会给你服务器添加压力。在你你的产品环境部署监视器前找出一个可接受的间隔。

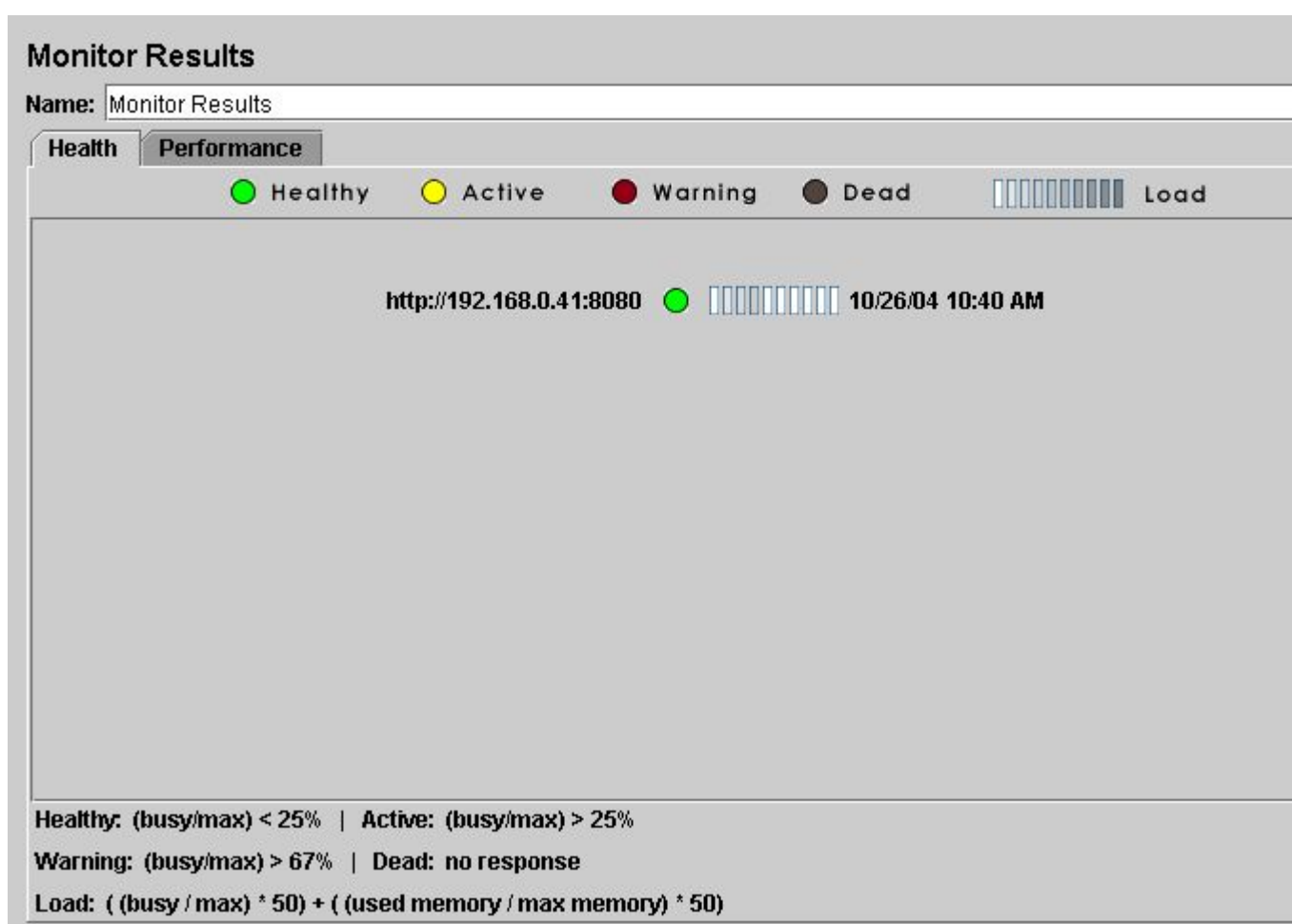
## 13.5 添加一个监听器保存测试结果

如果你想保存来自服务器的结果，添加一个简单的数据监听器。如果你想保存计算的统计表，在监听器输入一个文件名。如果你想保存产生数据和统计表，确认你使用不同的文件名。

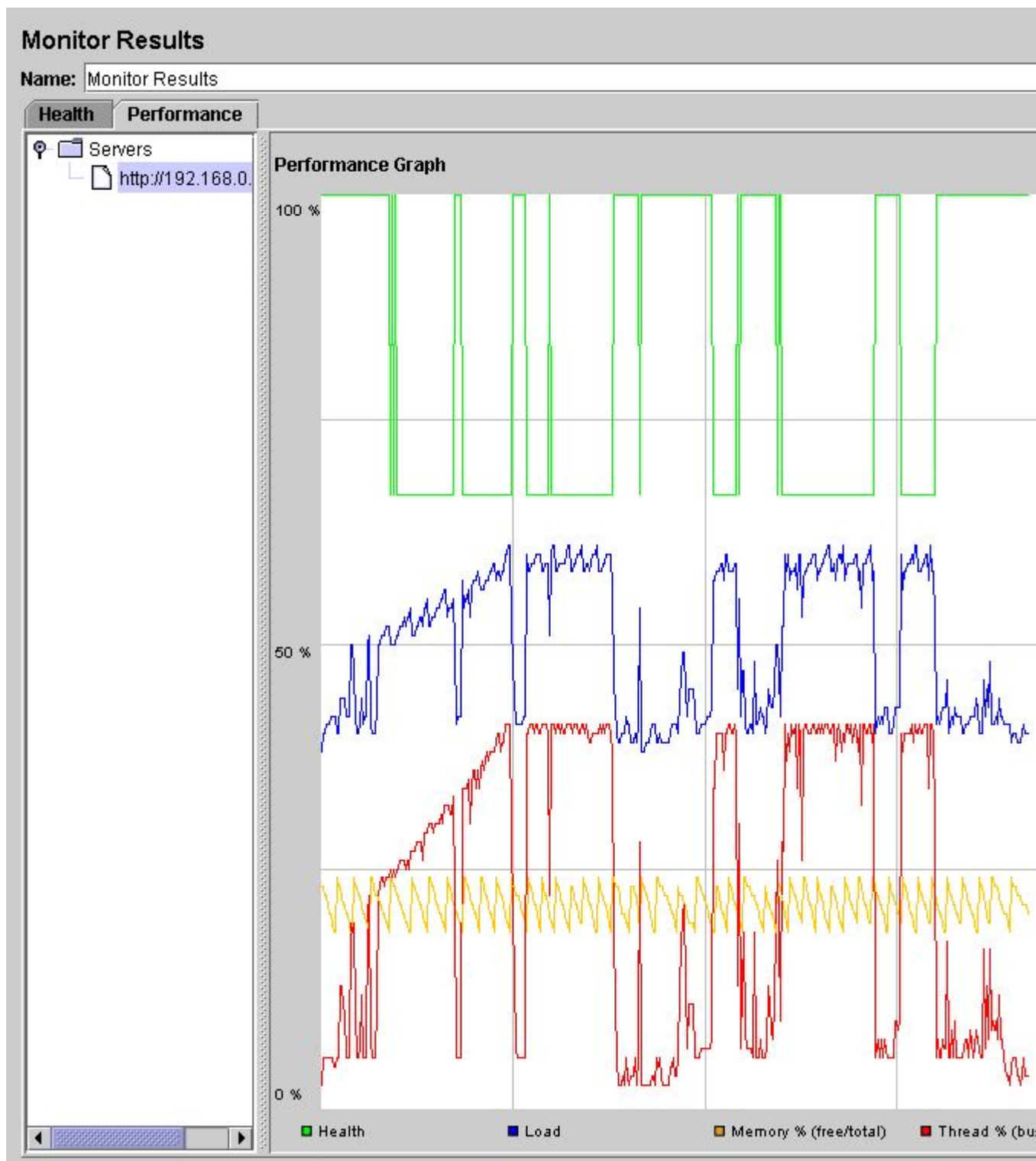
选择线程组元件，添加一个简单数据记录器监听器（添加-->监听器-->简单数据记录器）。下一步，你需要指定一个目录和一个输出文件文件名。你可以在文件名域输入它，也可以选择浏览按钮，浏览一个目录，然后加入一个文件名。

## 13.6 添加监视器结果

通过选择测试计划元件添加监听器（添加-->监听器-->监视器结果）。在监视器结果监听器中有两个 tab。第一个是“健康”，它显示了监视器受到的最后取样的状态。第二个 tab 是“性能”，它显示了服务器性能的历史视图。



一个关于健康情况的快速注释会被计算出来。典型地，一个服务器内存用完或者达到最大线程数，它就要崩溃。如果是 Tomcat 5，一旦线程到达最大，请求将被放置到一个队列直到一个线程可用。在容器之间线程关系的重要性改变很大，所以现在使用 50/50 的实现更加保守。一个更加 有效管理线程的容器可能看不到任何性能的下降，但是使用的内存也肯定会显示一些影响。




性能图像显示为不同的线条。空闲内存线显示在当前已分配存储块剩余多少空闲内存。Tomcat 5 返回最大内存，但是它没有被绘制。在一个调试好的环境，服务器应该从不达到最大内存。

注意在图形的两边都有标题。在左边是百分比，右边是死亡/健康。如果内存线尖峰上升和下降迅速，它可能指示 memory thrashing。在其它情况，使用 Borland

OptimizeIt 或者 JProbe 是一个好方法。你想看到的是对于负载，内存和线程的一个规则的图案。任何不确定路线的状态常常都预示了不良的性能或者某个种类的一个 bug。


## 13.7 保存测试计划

虽然它不是需要的，但是我们推荐你在运行前保存测试计划到一个文件。为了保存测试计划，从文件菜单选择保存测试计划（使用最新版本，它不再需要首先选择测试计划元件）。

 JMeter 允许你保存这个测试计划树或者仅仅其中一部分。为了仅保存在测试计划树上的特殊“分支”，选择在树中用来启动“分支”的测试计划元件，然后右击在菜单项中选择“保存”。或者，选择合适测试计划元件，然后从编辑菜单选择保存。


## 13.8 运行测试计划

从运行菜单，选择运行。

 如果你测试正在运行，JMeter 在右上方的角落点亮一个绿正方形显示。当所有测试停止，那个方块变成灰色。即使你选择了“停止”，绿光依然会继续持续，直到所有测试都已经退出。

一旦 JMeter 完成你的测试计划，从运行菜单选择停止。

如果你在监听器中选择一个文件保存结果，那么你将会有一个能够在任何 visualizer 中打开的文件。每个 visualizer 以它们自己的风格显示结果。

 有可能会在多于一个的 visualizer 中打开同一个文件。这是没有问题的。JMeter 会确保在测试运行时没有取样器记录到同一文件多于一次。

## 14. 监听器介绍

监听器是显示取样器结果的组件。结果可以显示在树、表格、图表或者简单的写入一个日志文件。为了观察来自提供的取样器的响应内容，可以添加“观察结果树”或者“在表格观察结果”监听器到测试计划。为了图形化观察响应时间，可以添加图形结果



⚠ 不同的监听器使用不同的方法显示响应信息。然而，如果他们其中一个被指点，他们所有使用相同的原始数据写入到输出文件。

“配置”按钮可以指定那些域被写入文件，和是否把它作为一个 CSV 或者 XML 文件。CSV 文件比 XML 文件小得多，所有如果产生大量的取样建议使用 CSV 文件。

如果你仅期望记录某几个取样，可以添加监听器作为取样器的一个子节点。或者你可以使用简单控制器去组织取样器集，并且添加监听器到那个控制器。相同的文件可以被多个取样器使用-但是确定它们都使用相同的配置！

## 14.1 屏幕捕获

JMeter 能够保存任何监听器作为一个 PNG 文件。在左边的面板选择监听器。单击 edit -> Save As Image

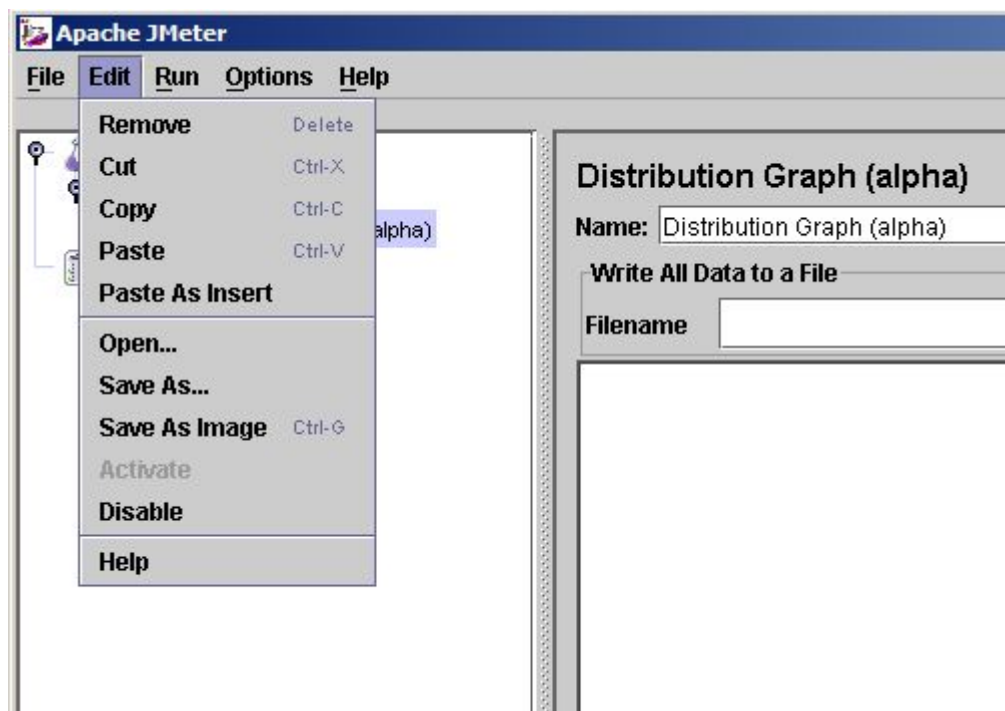


图 1-Edit -> Save As Image

## 14.2 非 GUI 测试运行

当在非 GUI 模式运行时，使用 -l 标志为测试运行创建一个顶级监听器。

## 14.3 资源使用

监听器。为了最小的资源使用，删除所有的监听器，并且使用-1 标志运行测试在非 GUI 模式来定义仅一个监听器。这样在测试完成之后日志文件可以被重新读取到一个监听器。

## 14.4CSV 日志格式

CSV 日志格式依赖于在配置中被选择的数据项。仅那些指定的数据项被记录在文件。列的表现顺序是固定的，如下：

- \*时间标志-自从 1970-1-1 的毫秒数
- \*用时-毫秒
- \*标签-取样器标签
- \*响应代码-例如 200、404
- \*响应消息-例如 OK
- \*线程名
- \*数据类型
- \*成功与否-true 或者 false
- \*失败消息-如果要的话
- \*字节数-在取样中的字节数
- \*URL

XML 文件格式如下：

## 14.6XML 日志格式 2.0

原始 XML (2.0) 格式如下（转行可以不相同）：

```
<?xml version="1.0" encoding="UTF-8"?>
<testResults version="1.2">
<sampleResult timeStamp="1144365463297" dataType="text"
threadName="Listen 1-1" label="HTTP Request" time="1502"
responseMessage="OK" responseCode="200" success="true">
  <sampleResult timeStamp="1144365464238" dataType="text"
threadName="Listen 1-1" label="http://www.apache.org/style/style.css"
time="171" responseMessage="OK" responseCode="200" success="true">

<property xml:space="preserve" name="samplerData">
GET http://www.apache.org/style/style.css
</property>
<binary>
body, td, th {
    font-size: 95%;
    font-family: Arial, Geneva, Helvetica, sans-serif;
```

```

        color: black;
        background-color: white;
    }
    ...
</binary>
</sampleResult>

</sampleResult>
...
</testResults>

```

## 14.6 XML 日志格式 2.1

更新的 XML (2.1) 格式如下 (转行可以不相同) :

```

<?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/images/asf_logo_wide.g
if</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/images/asf_logo_wide.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">
...

<html>
    <head>
...
    </head>
    <body>
...
    </body>

```

```

</html>
</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>

-- nonHTTP 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>

```

🟢 取样节点名字可以是“sample”或者“httpSample”。

## 14.7 Sample Attributes

取样器属性意义如下：

属性	内容
by	字节数
de	数据编码
dt	数据类型
ec	Error count (0 or 1, unless multiple samples are aggregated)
hn	Hostname where the sample was generated
lb	标签
lt	延时 (ms) -不是所有的取样器支持这个
na	所有线程组中的活跃线程数
ng	在这个线程组中活跃的线程数
rc	响应代码

rm	响应消息
s	是否成功
sc	Sample count (1, unless multiple samples are aggregated)
t	用时 (ms)
tn	线程名
ts	时间标志

varname Value of the named variable (versions of JMeter after 2.3.1)

JMeter2.1 和 2.1.1 版本保存响应代码为“rs”，但是读取它期望是“rc”。这个 bug 已经被修复，所以为“rc”；“rc”或者“rs”都可以被读取。



Versions of JMeter after 2.3.1 allow additional variables to be saved with the test plan. Currently, the variables are saved as additional attributes. The testplan variable name is used as the attribute name.

## 14.8 保存响应数据

像上面展示的那样，如果需要响应数据可以被保存为 XML 日志文件。然而，这将使文件相当大，并且文本必须被编码才可以被安静的验证 XML。同样图片不会被包括。

另一个解决方案是使用后置处理器保存响应结果到文件。这样为每个取样产生一个新的文件，并且保存文件为取样器名。文件名会被包含在一个取样日志输入。当取样日志文件被加载时如果需要数据将从文件从新得到。

## 14.9 Loading (reading) response data

To view an existing results file, you can use the File “Browse...” button to select a file. If necessary, just create a dummy testplan with the appropriate Listener in it.

Results can be read from XML or CSV format files. When reading from CSV results files, the header (if present) is used to determine which fields were saved. In order to interpret a header-less CSV file correctly, the appropriate JMeter properties must be set.

# 15. 远程测试

如果你的 JMeter 客户机不可能高效的模拟充足的用户加压力到你服务器，an option exists to control multiple, remote JMeter engines from a single JMeter GUI client。通过远程运行 JMeter，你可以跨越很多低配置的电脑复制一个测试，并在服务器模拟一个大的负载。一个 JMeter GUI 客户端实例能够控制任何多个远程 JMeter 实例，并从它们收集所有的数据。这种方法有如下特性：

- 保存测试取样到一个本地机器
- 从单个机器管理多个 JMeter 引擎。

注意当你在你的应用程序服务器上真正执行 JMeter 引擎时，你需要注意这样一个事实。这就是它将会在应用程序服务器添加处理开销，所以你的测试结果会稍微收到影响。推荐的方法是在同一个以太网段有一个或者多个机器作为你要配置运行 JMeter 引擎的应用程序服务器。这将在没有影响应用程序性能的前提下最小化网络在测试结果中的影响。

### 第一步：启动服务器

为了在远程节点运行 JMeter，通过运行 `JMETER_HOME/bin/jmeter-server` (unix) 或者 `JMETER_HOME/bin/jmeter-server.bat` (windows) 脚本在你希望运行的所有机器上启动 JMeter 服务器组件。

### 第二步：添加服务器 IP 到你的客户端配置文件

在 JMeter 控制机上编辑配置文件。在 `/bin/jmeter.properties` 文件中找到属性 `"remote_hosts"`，并添加你运行的 JMeter 服务器的 IP。添加多个服务器，逗号分隔。

注意你可以使用 `-J` 命令行参数来指定远程计算机。例如：

```
jmeter -Jremote_hosts=host1,127.0.0.1,host2
```

### 第三步：启动 JMeter 客户端

现在你准备启动 JMeter

## 15.2 技巧

如果你运行 Suse Linux，这些技巧会有帮助。默认安装会启用防火墙。如果那样的话，远程测试会不能完全工作。下列技巧是由 Sergey Ten 贡献。

如果你看到连接被拒绝，通过下列参数开启调试。

## 15.3 使用不同的端口

默认, JMeter 使用标准的 RMI 端口 1099。可能需要改变它。为了这样工作成功, 所有下列需要保持一致:

- 在服务器, 使用新的端口数启动 rmiregistry
- 在服务器, 启动 JMeter 时指定 server\_port
- 在客户端, 更新 remote\_hosts 属性包含新的远程主机: 端口设置

自从 JMeter2.1.1 以后, jmeter-server 脚本提供了更改端口支持。例如架设你想使用 1664 端口 (可能 1099 已经被使用)。

在 Windows 下 (在 DOS 窗口)

```
C:\JMeter> SET SERVER_PORT=1664
```

```
C:\JMeter> JMeter-SERVER [其他参数]
```

在 Unix 下

```
$ SERVER_PORT=1664 jmeter-server [其他参数]
```

[注意使用大写字母标示环境变量]

## 15.4 Using sample batching

Listeners in the test plan send their results back to the client JMeter which writes the results to the specified files. By default, samples are sent back as they are generated. This can place a large load on the network and the JMeter client. There are some JMeter properties that can be set to alter this behaviour.

- mode - sample sending mode - default is Standard
  - Standard - send samples as soon as they are generated
  - Hold - hold samples in an array until the end of a run. This may use a lot of memory on the server.
  - Batch - send saved samples when either the count or time exceeds a threshold
  - Statistical - send a summary sample when either the count or time exceeds a threshold. The samples are summarised by thread group name and sample label. The following fields are accumulated:
    - elapsed time
    - latency
    - bytes
    - sample count
    - error count

Other fields that vary between samples are lost.



The following properties apply to the Batch and Statistical modes:

- `num_sample_threshold` - number of samples in a batch (default 100)
- `time_threshold` - number of milliseconds to wait (default 60 seconds)

## 16. 最佳实践

### 16.1 限制线程数

你的硬件能力会限制你有效运行 JMeter 的线程数。它也会依赖于你服务器的速度（一个更快的服务器因为它更加快速的返回一个请求所以会使 JMeter 工作更加努力）。JMeter 工作越多，它的时间信息就越不准确。JMeter 做越多的动作，每个线程必须等待访问 CPU 的时间越长，定时信息越长。如果你需要大规模的负载测试，考虑在多个机器上运行多个非用户界面的 JMeter。

### 16.2 何处放置 Cookie 管理器

见构建一个 Web 测试

### 16.3 何处放置授权管理器

见构建一个高级的 Web 测试

### 16.4 使用代理服务器

详细参见 HTTP 代理服务器设置代理服务器。要做的最重要的事是过滤你感兴趣的所有请求。例如，在记录图片请求方面并没有观点（JMeter 被致使 下载一个页面的所有图片-见 HTTP 请求）。这些仅仅会能乱你的测试计划。大多数可能，所有你的文件共享一个扩展，例如 `.jsp`, `.asp`, `.php`, `.html` 或者相似的。你应该通过输入 `".\jsp"` 作为一个 `"Include Pattern"` 包含这些。

另外，你可以通过输入 `".\gif"` 作为一个 `"Exclude Pattern"`。依赖于你的应用程序，这可能是个好方法，也可能不是。你可能也必须包含样式表，javascript 文件和其他包含文件。测试你的设置来验证你记录的是你想要的，然后清楚，重新启动。

The Proxy Server expects to find a ThreadGroup element with a Recording Controller under it where it will record HTTP Requests to. This

conveniently packages all your samples under one controller, which can be given a name that describes the test case.

现在，经历测试用力的几步。如果你有一个以前定义的测试用力，使用 JMeter 记录你的行为来定义你的测试用例。一旦你完成一个明确步骤系列，使用一个适当的命名文件保存这个测试用例。然后，

代理服务器软件一个最实用的特性是你可以从被记录的取样中提取出某些通用的元件。通过在测试计划级别或者用户自定义变量元件中定义一些自定义变量，你可以让 JMeter 在你记录的取样中自动的替换那些值。举例来说，如果你在服务器“xxx.yyy.com”，测试一个应用，那么你可以使用“xxx.yyy.com”定义一个变量叫做“server”，然后在你记录的取样那个值被发现的任何地方都会被替换为“\${server}”。

## 16.5 用户变量

一些测试计划需要为不同的用户/线程使用不同的值。例如，你可能想要测试一个为每个用户指定一个唯一的登录序列。这通过 JMeter 提供的设备很容易完成。

例如：

- 创建一个包含用户名和密码的文本文件，以逗号分隔。把这个文件放到你的测试计划的相同目录
- 添加一个 CSV 数据集配置元件到测试计划。创建 USER 和 PASS 变量
- 在设当的取样器使用\${USER}替代登录名，使用\${PASS}替代密码

CSV 数据集元件会为每个线程读取一个新行。

## 16.6 减少资源需求

减少资源使用的一些建议：

- 使用非用户界面模式：jmeter -n -t test.jmx -l test.jtl
- 尽可能少的使用监听器；如果使用-l 标志
- 不要使用函数模式
- 使用 CSV 输出而不是 XML
- 仅保存你需要的数据
- 尽可能少的使用断言

## 16.7 BeanShell 服务器

BeanShell 拦截器有一个非常有用的特性——它扮演一个很容易通过 telnet 或者 http 访问的服务器



那是不安全的。连接到那个端口的任何人都可以发出任何 BeanShell 命令。这能够提供无限制的访问 JMeter 应用程序和主机。\*如果那些端口不是受保护拒绝访问不要允许这个服务器。

如果你想使用这个服务器，在 jmeter.properties 文件定义下列属性：

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

在上面的例子中，服务器会被启动，并且监听 9000 和 9001 端口。9000 端口会用来为 http 访问。9001 端口会用来为 telnet 访问。startup.bsh 文件会通过服务器处理，并且可以用来定义各种函数和启动变量。startup 文件为设置和打印 JMeter 和系统属性定义方法。这是你将在 JMeter 控制台看到的：

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

作为一个实际例子，假设你有一个在非图形界面模式长期运行的 JMeter，并且你想在测试中改变多个时间的吞吐量。测试计划包含了一个不变的吞吐量定时器，它定义了一个期限属性，例如 `$_P(throughput)`。下列 BeanShell 命令可以用来改变测试：

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

这个脚本可以存储到一个文件（throughput.bsh）的话，使用 bshclient.jar 发送到服务器。例如：

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

## 16.8 BeanShell scripting

### 16.8.1 Overview

Each BeanShell test element has its own copy of the interpreter (for each thread). If the test element is repeatedly called, e.g. within a loop, then the interpreter is retained between invocations unless the "Reset bsh.Interpreter before each call" option is selected.

Some long-running tests may cause the interpreter to use lots of memory; if this is the case try using the reset option.

You can test BeanShell scripts outside JMeter by using the command-line interpreter:

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

### 16.8.2 Sharing Variables

Variables can be defined in startup (initialisation) scripts. These will be retained across invocations of the test element, unless the reset option is used.\

Scripts can also access JMeter variables using the `get()` and `put()` methods of the "vars" variable, for example: `vars.get("HOST");`  
`vars.put("MSG", "Successful");`. The `get()` and `put()` methods only support variables with String values, but there are also `getObject()` and `putObject()` methods which can be used for arbitrary objects. JMeter variables are local to a thread, but can be used by all test elements (not just Beanshell).

If you need to share variables between threads, then JMeter properties can be used:

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

The sample .bshrc files contain sample definitions of `getprop()` and `setprop()` methods.

Another possible method of sharing variables is to use the "bsh.shared" shared namespace. For example:

```
if (bsh.shared.myObj == void)
```

```
Unknown macro: { // not yet defined, so create it}
```

```
bsh.shared.myObj.process();
```

Rather than creating the object in the test element, it can be created in the startup file defined by the JMeter property "beanshell.init.file". This is only processed once.

## 17. 求助！我的老板让我负担测试我们的web 应用程序

这是一个相当开放的命题。首先有很多问题要问，另外还有一些必须的资源。你会需要一些硬件运行基准/负载测试。许多工具证明是有用的。可以考虑一些产品。最后，为什么 Java 是实现负载测试/基准产品的一个好选择。

### 17.1 要问的问题

我们预期的平均用户数（正常负载）是多少？

我们预期的峰值用户数是多少？

什么时候压力测试你应用程序是个好时间（例如，下班时间或者每个周末），记住这可能彻底崩溃你们一个或多个服务器？

我们侧应用程序是否有状态？如果是这样，你的应用程序如何管理它（cookies, session 重写，或者其他方法）？

### 17.2 资源

下列资源会非常有用。记住如果你不定位这些资源，你将成为这些资源。As you already have your work cut out for you, it is worth knowing who the following people are, so that you can ask them for help if you need it.

## 17.2.1 网络

谁熟悉我们的网络技术？如果你遇到任何防火墙或者代理问题，这会变的很重要。一个专用测试网络（那会有一个非常低的网络延迟）是非常好的事情。知道谁可以为你设立一个是非常有用的（如果你感觉这是必须的）。如果应用程序不能预期测量，谁可以添加额外的硬件？

## 17.2.2 应用程序

谁知道应用程序的功能如何？正常顺序是

- 测试（低尺度--我们可以基准测试我们应用程序？）
- 基准测试（平均用户数）
- 负载测试（最大用户数）
- 破坏测试（最大极限是多少？）

测试过程可以从黑盒测试到白盒测试（不同点是第一个不需要应用程序知识[所以称为“黑河”]，然而第二个需要应用程序知识）进行。在这个过程不难发现应用程序的问题，所以它准备否定你的工作。

## 17.3 我应该使用什么平台运行基准测试/负载测试？

这应该是一块普遍使用的有一个标准的（例如 vanilla）软件安装硬件。记住，如果你公布你们结果，你的客户会做的第一件事雇佣一个研究生验证它们。You might as well make it as easy for this person as you possibly can.

对于 Windows，Windows XP Professional 应该是最小的（其它的不支持多线程超过 50-60 个连接，你可能期望比那更多的用户）。

非常好的免费平台包括 linux，BSD 和 Solaris Intel。如果你有跟多一点钱，还有一些商业的 linux。如果你能适应它，一个商业的 Unix（Solaris 等）可能是最好的选择。

For non-Windows platforms, investigate "ulimit -n unlimited" with a view to including it in your user account startup scripts (.bashrc or .cshrc scripts for the testing account).

如果你进行更高尺度的基准测试/压力测试，这个平台会成为限制因素。所以使用你们可用的最好的硬件和软件是有价值。记得在你公布基准测试时包括硬件/软件配置。

不要忘记 JMeter 的批处理模式。如果你有一个服务器，它支持 Java 但可能没有一个快的绘图实现，或者你需要远程登录，这会是有用的。批处理模式相比远程显示或者客户端-服务器模式可以减少网络流量。批处理文件可以被加载到工作站的 JMeter 来分析，或者你可以使用 CSV 输出和输入数据到电子表格。

## 17.4 工具

下列工具会都很有用。熟悉它们将会很有用的。这应该包括实验它们，并阅读适当的文档（man-pages, info-files, application --help messages 和任何提供的文档）。

### 17.4.1 ping

它常用来确定你是否能到达目标站点。Options can be specified so that 'ping' provides the same type of route reporting as 'traceroute'.

### 17.4.2 nslookup/dig

当用户通常会使用一个人们易读的网络地址，但是你可能希望当执行基准测试/压力测试时避免 DNS 查找的开销。这就用来确定你目标站点的唯一地址（IP 地址）。

### 17.4.3 traceroute

如果你不能“ping”通你的目标站点，这个命令多半可以用来确定问题（可能是防火墙或者代理）。它也可以用来估算整个网络延时（本地运行应该给出最小可能网络延迟-记住你的用户运行的将会是一个繁忙的 Internet）。通常，跳跃越少越好。

## 17.5 其他产品还有什么？

有很多商业的产品，但都有相当高的价格。如果你可以适应它，这或许也是可以走的路线。如果那些产品不能做你想要的，或者你限制在一个预算，下列工具就值得一看。事实上，你或许应该开始试试 Apache ab 工具，如果你的需求不是特别的复杂它可以很好的工作。

## 17.5.1 Apache 'ab'工具

你应该明确从使用这个开始。它很好的处理 HTTP 'get' 请求，做一点工作也可以处理 HTTP 'post' 请求。它用'C' 语言写的，它运行的非常好，并且提供很好的性能报告(if basic)。

## 17.5.2 HttpUnit

这个工具值得一看。它是一个用来执行 HTTP 测试/基准测试的类库（因此更值得开发人员关注）。它想要用来连同 JUnit 代替 web 浏览器（因此是非 GUI 的）。

## 17.5.3 微软 WAS

这个工具也很值得一看。它有一个优秀的用户界面，但是它也许不能完全做你想做的。If this is the case, be aware that the functionality of this product is not likely to change.

## 17.5.4 JMeter

如果你有非标准的需求，那么这个解决方案提供了以开源社区来提供它们（当然，如果你正在阅读这个文档，你可能已经热衷于这个工具）。这个产品免费和你的需求一起进行。

## 17.6 为什么是 Java?

为什么不是 Perl 或者 C?

Perl 多半是一个很好的选择，只可惜基准程序包似乎给出相当模糊的结果。还有，使用 Perl 模拟多个用户是一个很棘手的问题（多连接可以通过从一个 shell 脚本分支多个进程模拟，但是那些都不是线程，它们是进程）。然而，Perl 社区是很强大的。如果你发现某个东西已经写了一些好像有用的东西，这可能是一个很好的解决方案。

C, 当然是一个很好的选择（check out the Apache ab tool）。但是你需要准备所有测试你应用程序需要的常规网络，线程，状态管理码。

Java 给你了（免费）你应用程序需要的常规网络，线程，状态管理码。Java 有 HTTP, FTP 和 HTTPS-也有 RMI, IIOP 和 JDBC（没有提到 cookies, URL 编码和 URL 重定向）。另外 Java 提供了自动化垃圾回收和字节码安全。



And once Microsoft moves to a CLR (common language run-time) a Windows Java solution will not be any slower than any other type of solution on the Windows platform.

## 18 组件参考

### 18.1 Samplers

- FTP Request
- HTTP Request
- JDBC Request
- Java Request
- SOAP/XML-RPC Request
- WebService(SOAP) Request
- LDAP Request
- LDAP Extended Request
- Access Log Sampler
- BeanShell Sampler
- BSF Sampler
- TCP Sampler
- JMS Publisher
- JMS Subscriber
- JMS Point-to-Point
- JUnit Request
- Mail Reader Sampler
- Test Action
- 18.2 Logic Controllers
  - Simple Controller
  - Loop Controller
  - Once Only Controller
  - Interleave Controller
  - Random Controller
  - Random Order Controller
  - Throughput Controller
  - Runtime Controller
  - If Controller
  - While Controller
  - Switch Controller
  - ForEach Controller
  - Module Controller
  - Include Controller
  - Transaction Controller

- Recording Controller
- 18.3 Listeners
  - Sample Result Save Configuration
  - Graph Full Results
  - Graph Results
  - Spline Visualizer
  - Assertion Results
  - View Results Tree
  - Aggregate Report0
  - View Results in Table
  - Simple Data Writer
  - Monitor Results
  - Distribution Graph (alpha)
  - Aggregate Graph
  - Mailer Visualizer
  - BeanShell Listener
  - Summary Report
- 18.4 Configuration Elements
  - CSV Data Set Config
  - FTP Request Defaults
  - HTTP Authorization Manager
  - HTTP Cache Manager
  - HTTP Cookie Manager
  - HTTP Request Defaults
  - HTTP Header Manager
  - Java Request Defaults
  - JDBC Connection Configuration
  - Login Config Element
  - LDAP Request Defaults
  - LDAP Extended Request Defaults
  - TCP Sampler Config
  - User Defined Variables
  - Simple Config Element
- 18.5 Assertions
  - Response Assertion
  - Duration Assertion
  - Size Assertion
  - XML Assertion
  - BeanShell Assertion
  - MD5Hex Assertion
  - HTML Assertion
  - XPath Assertion
  - XML Schema Assertion
  - BSF Assertion

- 18.6 Timers
  - Constant Timer
  - Gaussian Random Timer
  - Uniform Random Timer
  - Constant Throughput Timer
  - Synchronizing Timer
  - BeanShell Timer
- 18.7 Pre Processors
  - HTML Link Parser
  - HTTP URL Re-writing Modifier
  - HTML Parameter Mask
  - HTTP User Parameter Modifier
  - User Parameters
  - Counter
  - BeanShell PreProcessor
  - BSF PreProcessor
- 18.8 Post-Processors
  - Regular Expression Extractor
  - XPath Extractor
  - Result Status Action Handler
  - Save Responses to a file
  - Generate Summary Results
  - BeanShell PostProcessor
  - BSF PostProcessor
- 18.9 Miscellaneous Features
  - Test Plan
  - Thread Group
  - WorkBench
  - SSL Manager
  - HTTP Proxy Server
  - HTTP Mirror Server
  - Property Display
  - Debug Sampler
  - Debug PostProcessor
- 18.10 Reports
  - Report Plan
  - Report Table
  - HTML Report Writer
  - Report Page
  - Line Graph
  - Bar Chart

# 19. 函数

JMeter 函数是一些能够转化在测试树中取样器或者其他配置元件的域的特殊值。一个函数调用就像这样：

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

“\_\_functionName”匹配函数名。

圆括号周围设置函数的参数，例如`${__time(YMD)}`实际参数因函数而不同。不需要参数的函数使圆括号内

为空，例如`${__threadNum}`。

函数列表：

- `regexFunction` - 正则表达式计数器
- `counter`
- `threadNum` - 得到线程数
- `intSum` - 添加变量
- `StringFromFile` - 从文件读取一行
- `machineName` - 得到本地计算机名
- `JavaScript (Apache Rhino)`
- `random number`
- `CSVRead` - 从 CSV 文件读取
- 读取一个属性
- `P` - 读取一个属性
- `setProperty` - 设置一个属性
- `log` - 记录一个日志
- `logn` - 记录一个日志
- `BeanShell` - 运行 BeanShell
- `split` - 分隔一个字符串为变量
- `XPath` - 使用一个 xpath 表达式
- `time` - 返回一些格式的当前时间
- `jexl` - 执行一个 jexl 表达式

## 19.1 函数可以做什么

有两类函数：自定义静态值（或者变量），和内置函数。

自定义静态值允许当一个测试树编译并且提交运行时自定义变量被它们的静态值代替。这个替代在测试运行开始时发生一次。这可以用来替换所有 HTTP 请求中的 DOMAIN 域，例如 — 做一个简单的问题来使用同样的测试改变 for example - making it a simple matter to change a test to target a different server with the same test.

这用不使用函数的替代是可能的，但是不方便而且 intuitive。它需要用户创建默认配置元件来填充取样器空值。自定义函数允许 one to replace only part of any given value, not just filling in blank values. 使用内建的函数用户可以计算基于在运行时前一个相应数据的新值， which thread the function is in, the time, and many other sources. These values are generated fresh for every request throughout the course of the test.



函数在线程之间是共享的。在测试计划中每个函数调用是被一个单独函数实例调用的。

## 19.2 函数被用来做什么

一个用户自定义函数可以写在任何测试组件的任何域中。一些域不允许随机字符串，因为他们期望数字，然而不期望一个函数。然而，大多数域会允许函数。内建的函数允许写进任何非控制器测试组件的任何域。这包含取样器，定时器，监听器，修改器，断言，前置处理器，后置处理器和配置元件。



当使用变量 / 函数 When using variable/function references in SQL code (etc), remember to include any necessary quotes for text strings, i.e. use  
SELECT item from table where name='\${VAR}.'  
not  
SELECT item from table where name=\${VAR}  
(unless VAR itself contains the quotes)


## 19.3 书写函数字符串

自定义函数使用这种格式： `${varName}`。在测试计划树元件中，一个自定义值两列的表格 a two-column table of user-defined values is kept, matching up variable names with static values. Referencing the variable in a test element is done by bracketing the variable name with `'${' and '}'`.

内建函数使用同样的风格书写，但惯例，内建参数以 `__` 开始来避免和用户值名 \*冲突。一些函数使用参数配置它们，并且包含在括弧内，以逗号分隔。如果函数没有参数，括弧可以省略。A further complication for argument values that themselves contain commas is that the value should be escaped as necessary. Thus, if you need to include a comma in your parameter value, escape it like so: `'\,'`.

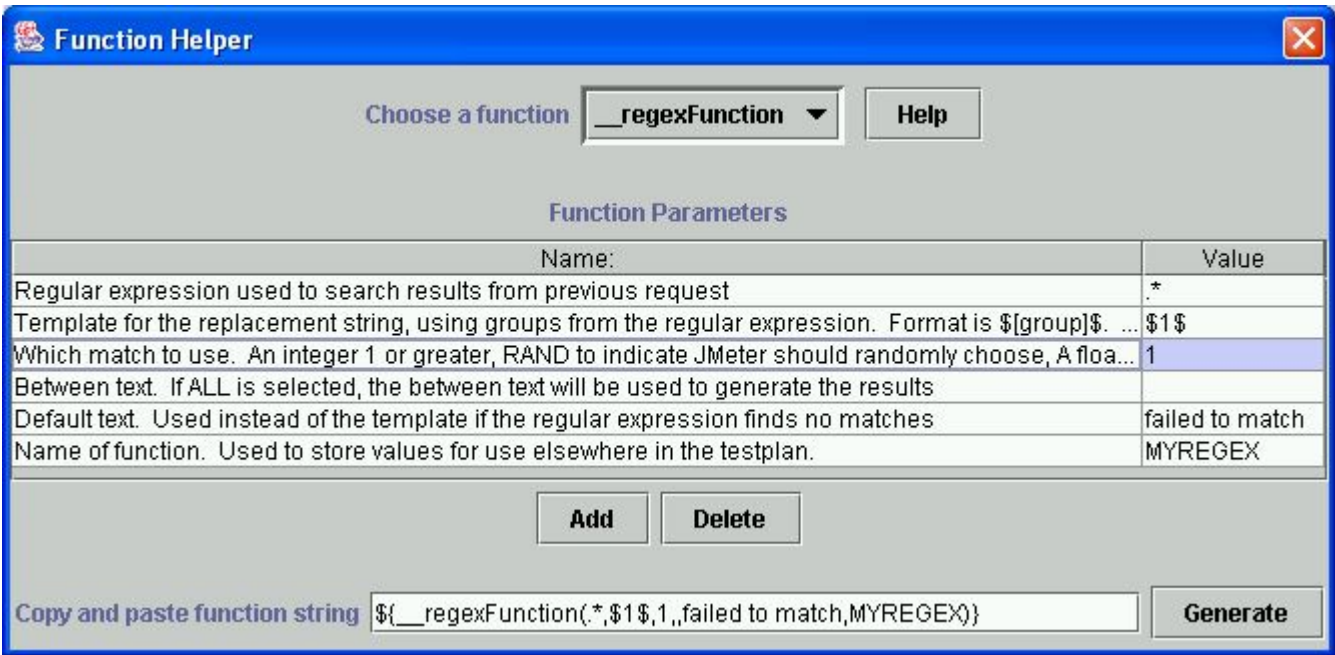
JMeter 提供了一个工具来帮助你使用各种内建函数构建函数调用，然后你可以拷贝粘贴。 It will not automatically escape values for you, since

functions can be parameters to other functions, and you should only escape values you intend as literal.

 \* 如果你使用和内建函数同样的名字定义一个自定义变量，你的静态变量会覆盖内建函数。

## 19.4 函数助手对话框

函数助手对话框从 JMeter 工具菜单中是可用的。



函数助手对话框

使用函数助手，你可以从下拉中选择一个函数，并赋予值。

对于它的参数，表格的左列提供了简短的参数描述，右边的列是你为那个参数填写的值。不同的函数使用不同的参数。

当你完成这些，点击“生成”按钮，适当的字符串就生成出来了，你可以拷贝粘贴到你喜欢的测试计划中。

## 19.5 函数

# 19.5.1 \_\_regexFunction

正则函数用来解析使用任何正则表达式的前响应（通过用户提供）。这个函数返回使用填充的变量值的模板字符串。

\_\_regexFunction 存储值以备以后使用。在第六个参数中，你可以指定一个参考名。在以后这个函数执行时，同样的值会在稍后使用自定义值规则时被得到。例如，如果你输入“refName”作为第六个参数，你将可以这样使用：

- `${refName}` to refer to the computed result of the second parameter ("Template for the replacement string") parsed by this function
- `${refName_g0}` to refer to the entire match parsed by this function.
- `${refName_g1}` to refer to the first group parsed by this function.
- `${refName_g#}` to refer to the n th group parsed by this function.
- `${refName_matchNr}` to refer to the number of groups found by this function.

## 参数

属性	描述	是否需要
第一个参数	第一个参数使用应用于响应数据的正则表达式。It will grab all matches. Any parts of this expression that you wish to use in your template string, be sure to surround in parentheses. Example: <code>&lt;a href="(.)"&gt;. This will grab the value of the link and store it as the first group (there is only 1 group). Another example: <code>&lt;input type="hidden" name="(.)" value="(.)"&gt;</code>. This will grab the name as the first group, and the value as the second group. These values can be used in your template string</code>	Yes
Second argument	This is the template string that will replace the function at run-time. To refer to a group captured in the regular expression, use the syntax: <code>Yes \$[group_number]\$. Ie: \$1\$, or \$2\$. Your template can be any string.  The third argument tells JMeter which match to use. Your regular expression might find numerous matches. You have four choices:</code>	
Third argument	<ul style="list-style-type: none"><li>• An integer - Tells JMeter to use that match. '1' for the first found match, '2' for the second, and so on</li><li>• RAND - Tells JMeter to choose a match at random.</li><li>• ALL - Tells JMeter to use all matches, and create a template string for each one and then append them all together. This option is little used.</li><li>• A float number between 0 and 1 - tells JMeter to find the Xth match using the formula: <code>(number_of_matches_found * float_number)</code> rounded to nearest integer.</li></ul>	No, default=1
Fourth	If 'ALL' was selected for the above argument value, then this argument will No	

argument be inserted between each appended copy of the template value.

Fifth argument Default value returned if no match is found No

A reference name for reusing the values parsed by this function.

Sixth argument Stored values are `${refName}` (the replacement template string) and `${refName_g#}` where `"#"` is the group number from the regular expression ("0" can be used to refer to the entire match). No

Seventh argument Input variable name. If specified, then the value of the variable is used as the input instead of using the previous sample result. No

## 19.5.2 \_\_counter

counter 每次调用产生一个新数字，从 1 开始每次增加 1。counter 可以配置来保持单独模拟每个用户的值，或者对于所有的用户使用同一个 counter。如果每个用户的值是单独增加的，那在整个测试计划过程中就像记录数字迭代一样。一个全局的 counter 就像记录请求运行了多少次。

counter 使用一个数字变量来计数，它有一个最大值 2,147,483,647。

counter 函数实例现在完全是单独的。[JMeter 2.1.1 和较早版本使用一个固定的线程变量来保持跟踪每个用户的计数，所以多个 counter 函数操作同样的值。] 全局 counter - "FALSE" - 通过每个 counter 实例单独维护。

### 属性


属性	描述	是否需要
第一个参数	TRUE 如果你希望每个模拟的用户的 counter 保持单独并分隔其他用户设为 TRUE。 FALSE for a global counter.	是
第二个参数	A reference name for reusing the value created by this function. Stored values are of the form <code>\${refName}</code> . This allows you to keep one counter and refer to its value in multiple places. [For JMeter 2.1.1 and earlier this parameter was required.]	No

## 19.5.3 \_\_threadNum

thread number 函数简单返回当前执行的线程数量。那些数字是单独的线程组，意味着从函数的视角来看线程组 1 的线程 #1 和在另一个线程组的线程#1 是不可区分的。


这个函数没有参数。



 这个函数不可以运行在那些任何运行在一个单独线程的配置元件中(例如 自定义变量)。 Nor does it make sense to use it on the Test Plan.

## 19.5.4a \_\_intSum

intSum 函数用来计算两个或者更多 int 值的和。

 JMeter2.3.1 或者更早版本需要被表现的参考名，这个参考名现在是可选的，但是它必须不是一个非法的整数

### 属性

属性	描述	是否需要
第一个参数	第一个 int 值。	是
第二个参数	第二个 int 值。	是
第 n 个参数	第 n 个 int 值。	否
最后一个参数	为通过这个函数计算的值重用的参考名。如果指定，参考名必须包含至少一个非数字字符，否则它将被作为另一个 int 值对待添加进去。	否

## 19.5.4b \_\_longSum

longSum 函数用来计算两个或者更多 long 值的和。

### 属性

属性	描述	是否需要
第一个参数	第一个 long 值。	是
第二个参数	第二个 long 值。	是
第 n 个参数	第 n 个 long 值。	否
最后一个参数	为通过这个函数计算的值重用的参考名。如果指定，参考名必须包含至少一个非数字字符，否则它将被作为另一个 long 值对待添加进去。	No

# 19.5.5 \_\_StringFromFile

StringFromFile 函数用来从文本文件读取字符串。对于运行需要许多变量数据的测试非常有用。例如当测试一个银行程序时，100 个或者 1000 个不同的帐号数字可能是需要的。

另见 CSV 测试配置元件，它很容易使用。然而，那个元件现在不支持多个输入文件。

每次调用，它会从文件读取下一行。当文件到达最后，它会从开头再次开始读取，如果没有到达最大循环数。如果在一个测试脚本对于函数有多个参考名，每个都会单独打开文件，甚至文件名是相同的。[如果这个值在其他任何地方再次使用，对于每个函数的调用使用不同的变量名。]

如果打开文件或者读取文件发生错误，函数返回字符串“\*ERR\*”

## 属性

属性	描述	是否需要
文件名	Path to the file name. (The path can be relative to the JMeter launch directory) If using optional sequence numbers, the path name should be suitable for passing to DecimalFormat. See below for examples.	Yes
Variable Name	A reference name - refName - for reusing the value created by this function. Stored values are of the form \${refName}. Defaults to "StringFromFile_".	No
Start sequence number	Initial Sequence number (if omitted, the End sequence number is treated as a loop count) No	No
	End sequence number Final sequence number (if omitted, sequence numbers can increase without limit)	No

The file name parameter is resolved when the file is opened or re-opened.

The reference name parameter (if supplied) is resolved every time the function is executed.

## Using sequence numbers:

When using the optional sequence numbers, the path name is used as the format string for java.text.DecimalFormat. The current sequence number is passed in as the only parameter. If the optional start number is not specified, the path name is used as is. Useful formatting sequences are:

- insert the number, with no leading zeros or spaces  
000 - insert the number packed out to 3 digits with leading zeros if necessary

Examples:

```
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
```

If more digits are required than there are formatting characters, the number will be expanded as necessary.

\*To prevent a formatting character from being interpreted, enclose it in single quotes. Note that "." is a formatting character, and must be enclosed in single quotes \*(though #. and 000. work as expected in locales where the decimal point is also ".")

In other locales (e.g. fr), the decimal point is "," - which means that "#." becomes "nnn,".

See the documentation for DecimalFormat for full details.

If the path name does not contain any special formatting characters, the current sequence number will be appended to the name, otherwise the number will be inserted according to the formatting instructions.

If the start sequence number is omitted, and the end sequence number is specified, the sequence number is interpreted as a loop count, and the file will be used at most "end" times. In this case the filename is not formatted.

```
${_StringFromFile(PIN#'. 'DAT,,1,2)} - reads PIN1.DAT, PIN2.DAT
```

```
${_StringFromFile(PIN.DAT,,2)} - reads PIN.DAT twice
```

Note that the "." in PIN.DAT above should not be quoted. In this case the start number is omitted, so the file name is used exactly as is.

## 19.5.6 \_\_machineName

machineName 函数返回本地主机名。

### 属性

属性	描述	是否需要
变量名	为重通过这个函数计算的这个值的一个参考名	否

## 19.5.7 \_\_javaScript


javaScript 函数执行一个 JavaScript 代码片段，并返回它的值。

JMeter 的 Javascript 函数调用一个单独的 JavaScript 解析器。Javascript 是一种脚本语言，所以你可以做计算等。

语言详细，请参考 Mozilla Rhino 概述


下列变量对于脚本是可用的：

- ctx - JMeterContext 对象
- vars - JMeterVariables 对象
- threadName - 字符串
- sampler - 当前取样器对象 (如果有的话)
- sampleResult - 前一个 SampleResult 对象 (如果有的话)
- props - JMeter Properties 对象

 JMeter 不是一个浏览器，并不能在下载页面解析 JavaScript。

属性

属性	描述	是否需要
被执行的 JavaScript 表达式。例如：		
表 达 式	<ul style="list-style-type: none"><li>•<ul style="list-style-type: none"><li>○ new Date() - return the current date and time</li><li>○ Math.floor(Math.random()*(\$ {maxRandom}+1)) - a random number between 0 and the variable maxRandom</li><li>○ \$ {minRandom}+Math.floor(Math.random()*(\$ {maxRandom}-\$ {minRandom}+1)) - a random number between the variables minRandom and maxRandom</li><li>○ "\$ {VAR}"=="abcd"</li></ul></li></ul>	Yes
	Variable Name	No
	A reference name for reusing the value computed by this function.	No
		No
		No

 Remember to include any necessary quotes for text strings and JMeter variables. Also, if the expression has commas, please make sure to escape them. For example in:  
\${\_\_javaScript('\$ {sp}'.slice(7,99999))}  
the comma after 7 is escaped.

19.5.8 \_\_Random

random 函数返回在最小值和最大值之间的一个随机数。

## 属性

属性	描述	是否需要
最小值	一个数字	是
最大值	一个更大的数	是
变量名	A reference name for reusing the value computed by this function.	否

## 19.5.9 \_\_CSVRead

CSVFile 函数从 CSV 文件返回一个字符串 (c.f. StringFromFile )

注意: 1.9.1 以下版本仅支持一个文件。JMeter 自从 1.9.1 版本支持多文件名。

**在大多数情况下，较新的 CSV 数据集配置元件更早使用**

当一个文件名第一次遇到时，文件会被打开，并读取到一个内部数组。如果一个空行被发现，这被认为是文件的结尾 - 这允许使用跟随注释 (N.B. 这个特性是在 1.9.1 以后引进的)

所有接下来同样文件名的参考使用相同内部数组。N.B. 文件名大小写对于函数是有效的，甚至 OS 不关心，所以 CSVRead(abc.txt, 0) 和 CSVRead(aBc.txt, 0) 会指向不同的内部数组。

\*ALIAS 特性允许同样的文件被打开多次，并允许使用更短的文件名。

每个线程都有它自己的内部指针指向文件数组的当前行。当一个线程第一次指向这个文件，它会被分配到数组的下一个空行，所以每个线程会和其他线程访问不同的行。 [Unless there are more threads than there are rows in the array.]

注意: the function splits the line at every comma by default. If you want to enter columns containing commas, then you will need to change the delimiter to a character that does not appear in any column data, by setting the property: csvread.delimiter

## 属性

属性	描述	是否需要
File Name	The file (or *ALIAS) to read from	Yes
Column number	The column number in the file. 0 = first column, 1 = second etc. "next" - go to next line of file. *ALIAS - open a file and assign it to the alias	Yes

For example, you could set up some variables as follows:

- 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)}`

This would read two columns from one line, and two columns from the next available line. If all the variables are defined on the same User Parameters Pre-Processor, then the lines will be consecutive. Otherwise, a different thread may grab the next line.



The function is not suitable for use with large files, as the entire file is stored in memory. For larger files, use CSV Data Set Config element or StringFromFile .

## 19.5.9 \_\_property

property 返回 JMeter 的属性的值。如果这个属性值没找到，没有默认值提供，它会返回属性名。当提供一个默认值时，就不需要提供一个函数名 - 参数可以为空，并它被忽略。

For example:

- `${__property(user.dir)}` - 返回 user.dir 值
- `${__property(user.dir,UDIR)}` - 返回 user.dir 值并保存在 UDIR 中
- `${__property(abcd,ABCD,atod)}` - 返回属性 abcd 的值(如果未定义, 返回"atod") 并保存在 ABCD 中
- `${__property(abcd,,atod)}` - 返回属性 abcd 的值 (如果未定义, 返回"atod")但并不保存它

### 属性

属性	描述	是否需要
属性名	要得到的属性名	是
变量名	A reference name for reusing the value computed by this function.	否
默认值	这个属性的默认值	否

## 19.5.10 \_P

This is a simplified property function which is intended for use with properties defined on the command line. Unlike the \_\_property function,

there is no option to save the value in a variable, and if no default value is supplied, it is assumed to be 1. The value of 1 was chosen because it is valid for common test variables such as loops, thread count, ramp up etc.

For example:

Define the property value:

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

Fetch the values:

```
${__P(group1.threads)} - return the value of group1.threads  
${__P(group1.loops)} - return the value of group1.loops  
${__P(hostname,www.dummy.org)} - return value of property hostname or  
www.dummy.org if not defined
```

In the examples above, the first function call would return 7, the second would return 1 and the last would return www.dummy.org (unless those properties were defined elsewhere!)

**Parameters**

Attribute	Description	Required
Property Name	The property name to be retrieved.	Yes
Default Value	The default value for the property. If omitted, the default is set to "1".	No

**19.5.11 \_\_log**

log 函数把一条信息记入日志，并返回空字符串

**参数**

属性	描述	是否需要
被记录的字符串	一个字符串	是
日志等级	OUT, ERR, DEBUG, INFO (默认), WARN or ERROR	否
Throwable 文本	如果非空，创建一个 Throwable 传递到 logger	否

OUT 和 ERR 日志等级用来直接输出到 System.out 和 System.err

相应的，既然这样，输出总是被打印 - 它不依赖当前的日志等级

## 19.5.12 \_\_logn

logn 函数把一条信息记入日志，并返回空字符串

### 参数

属性	描述	是否需要
被记录的字符串	一个字符串	是
日志等级	OUT, ERR, DEBUG, INFO (默认), WARN or ERROR	否
Throwable 文本	如果非空，创建一个 Throwable 传递到 logger	否

相应的，既然这样，输出总是被打印 - 它不依赖当前的日志等级

## 19.5.13 \_\_BeanShell

BeanShell 函数执行通过它的脚本并返回结果。

请注意 BeanShell 的 jar 文件并没有包含在 JMeter 中；它需要单独下载。

\*使用 BeanShell 的全部详细信息，请参见 BeanShell 站点  
<http://www.beanshell.org/> \*

Note that a different Interpreter is used for each independent occurrence of the function in a test script, but the same Interpreter is used for subsequent invocations. This means that variables persist across calls to the function.

A single instance of a function may be called from multiple threads. However the function execute() method is synchronised.

If the property "beanshell.function.init" is defined, it is passed to the Interpreter as the name of a sourced file. This can be used to define common methods and variables. There is a sample init file in the bin directory: BeanShellFunction.bshrc.

The following variables are set before the script is executed:

- log - the logger for the BeanShell function ★
- ctx - the current JMeter context variable
- vars - the current JMeter variables
- threadName - the threadName
- Sampler the current Sampler, if any



- SampleResult - the current SampleResult, if any

★ means that this is set before the init file, if any, is processed. Other variables vary

from invocation to invocation.

## Parameters

Attribute	Description	Required
BeanShell script	A beanshell script (not a file name)	Yes
Name of variable	A reference name for reusing the value computed by this function.	No

Example:

```
${__BeanShell(123*456)} - returns 56088
```

```
${__BeanShell(source("function.bsh"))} - processes the script in
function.bsh
```

## 19.5.14\_split

split 函数按照分隔符分隔字符串，并返回原始字符串。如果任何分隔符相邻，返回“?”。这个分隔的字

符串返回在变量\${VAR\_1}，\${VAR\_2}等中。变量的数量返回在\${VAR\_n}中。自从 JMeter 2.1.2 以后，

一个连续的分隔符被作为一个缺少的变量处理，并以“?”返回。同时，允许它和 ForEach 控制器一起工作

的更好，\_\_split now deletes the first unused variable in case it was set by a previous

split.

例如:

Define VAR="a|c|" in the test plan.

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

This will return the contents of VAR, i.e. "a|c|" and set the following variables:

VAR\_n=4 (3 in JMeter 2.1.1 and earlier)

VAR\_1=a

VAR\_2=?

VAR\_3=c

VAR\_4=? (null in JMeter 2.1.1 and earlier)

VAR\_5=null (in JMeter 2.1.2 and later)

\*\*

属性	描述	是否需要
分隔的字符串	一个分隔的字符串，例如"a b c"	是
变量名	A reference name for reusing the value computed by this function.	是
分隔符	分隔符字符，例如   。 If omitted, , is used. Note that , would need to be specified as \, .	否

### 19.5.15\_XPATH

XPATH 函数读取一个 XML 文件并匹配 XPath，每个节点的匹配都是迭代遍历，就像 CVSReader。

例如：

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

这会匹配在 build.xml 文件的所有 target 元素，并返回 name 变量

#### 参数

属性	描述	是否需要
要解析的 XML 文件	要解析的 XML 文件	是
XPath	在 XML 文件一个匹配节点的 XPath 表达式	是

## 19.5.16\_setProperty

setProperty 函数设置 JMeter 属性的值。来自函数的默认返回值是空字符串。所以这个函数调用可以在任

何函数可用的地方使用。

原始值可以通过设置第三个可选参数为“true”来返回。

属性对于 JMeter 是全局的，所以可以用来在任何线程和线程组之间传递。

### 参数

属性	描述	是否需要
Property Name	被设置的属性名	是
Property Value	属性的值	是
True/False	原始值是否被返回	否

## 19.5.17\_time

这个时间函数以不同的格式返回当前时间

### 参数

属性	描述	是否需要
Format	这个格式被传递给 SimpleDateFormat。这个函数支持各种不同的简写别名，见下文	否
Variable	要设置的变量名	否

如果这个格式字符串被删除，那么函数以毫秒返回当前时间。否则，当前时间被传递给 SimpleDateFormat

。支持下列简写别名：

- YMD = yyyyMMdd
- HMS = HHmmss
- YMDHMS = yyyyMMdd-HHmmss
- USER1 = whatever is in the Jmeter property time.USER1
- USER2 = whatever is in the Jmeter property time.USER2

这些默认值可以通过适当的设置 JMeter 属性来改变，例如 time.YMD=yyMMdd

## 19.5.18\_jexl

jexl 函数返回执行 [Commons JEXL 表达式](#) 的结果。

### 参数

属性	描述	是否需要
表达式	要被计算的表达式	是

## 19.6 预先定义变量

大多说变量通过函数或者测试元件例如自定义变量调用来设置;假如用户已经完全控制使用的变量名了。

然而一些变量通过 JMeter 内部定义。它们列表如下

- cookiename - 包含 cookie 的值
- JMeterThread.last\_sample\_ok - 最后一个取样是否正确 - 正确/不正确

## 19.7 预先定义属性

属性被定义在 jmeter.properties 文件内, 或者在命令行上。一些附加属性通过 JMeter 定义。如下列出。

START 属性也可以被拷贝给使用其他相同名字的变量。

- START.MS-开始时间, 使用 ms 为单位
- START.YMD-开始时间, 使用 yyyyMMdd 格式
- START.HMS-开始时间, 使用 HHmmss

and many other sources. These values are generated fresh for every request throughout the

course of the test.

# 20. 正则表达式

## 20.1 概述

JMeter 包含了软件 [Apache Jakarta ORO](#) 模式匹配。

在 Jakarta web 站点上有一些文档, 例如模式匹配字符概要信息。

还有一个是建立在 OROMatcher 使用手册基础上的文档, 它也很有用。

这里的模式匹配很接近 Perl 里的模式匹配。一个完全安装的 Perl 包括了大量的正则表达式的文档, 可以查找 `perlrequick`, `perlretut`, `perlre`, `perlref`。

它最值得强调的是在响应断言上作为测试元件使用时“包含”和“匹配”之间的不同:

“包含”意味着正则表达式匹配至少目标的一些部分, 所以 `'a1.t'` 是“匹配” `'alphabet'`。因为正则表达式匹配 `'phabe'` 字符串。

✔ 不像 Perl, 正则表达式不需要嵌套在 `//` 内。

## 20.2 例子

提取单个字符串

假如你想匹配下列 web 页面的一部分:

`name="file" value="readme.txt"`, 并且想提取 `readme.txt`

一个合适正则表达式是:

`name="file" value="(.)+"`

特定字符如下:

- `(and)` - 它们会嵌套到被返回的匹配字符串部分
- `.` - 匹配任何字符。 `+` - 一次或者多次。 `?` - 不是贪婪的, 例如: 当第一次匹配成功后停止

✔ 没有 `?`, `+` 第一匹配成功之后会继续, “直到发现最后一个可能” - 可能并不是想要的。

提取多个字符串

假如你想匹配下列 web 页面的一部分:

`name="file.name" value="readme.txt"`, 并且想提取 `file.name` 和 `readme.txt`

一个合适正则表达式是:

`name="(.)+" value="(.)+"`

这就创建了两个组, 它们将被 JMeter 正则表达式提取器模板为 `$1$` 和 `$2$`。

JMeter 正则表达式提取器在附加变量中保存组的值。

例如，假设：

- 参考名：MYREF
- 表达式：name="(.+?)" value="(.+?)"
- 模板：\$1\$2\$



不要在//中嵌套正则表达式

下列变量将会被设置：

- MYREF: file.namereadme.txt
- MYREF\_g0: name="file.name" value="readme.txt"
- MYREF\_g1: file.name
- MYREF\_g2: readme.txt

那些变量在 JMeter 测试计划中稍后会谈到，例如\${MYREF}，\${MYREF\_g1}等

## 20.3 命令行模式

模式匹配行为表现在多种细微的不同的情况，但是依赖于多行和单行修改者的设置。

会有四种可以的组合：

- 默认行为：'.' 匹配任何字符除了"\n"，^匹配仅这个字符串开始的，\$匹配仅在结尾的或者一个新行的结尾。
- 单行修改者(?s)：视字符串为一个单独行。'.' 匹配任何字符，甚至"\n"，^匹配仅这个字符串开始的，\$匹配仅在结尾的或者一个新行的结尾。
- 多行修改者(?m)：视字符串为一个多行集。'.' 匹配任何字符除了"\n"，\$能够匹配在字符串内开始或者结束的任何行。
- 双修改者(?sm)：视字符串为一个单独行，但是探测多行。'.' 匹配任何字符，甚至"\n"。然而，^和\$能够匹配在字符串内开始或者结束的任何行。

## 21. 测试结果分析

### 一、Listener 的使用

用过 LoadRunner 的人应该都知道，LoadRunner 会为我们提供一大堆图标和曲线。但是在 Jmeter 里，我们只能找到几个可怜的 Listener 来方便我们查看测试结果。但是，对于初学者来说，一些简单的结果分析工具可以使更容易理解性能测试结果的分析原理。所以，千万别小看 这几个简单的 Listener 啊。

## A. Aggregate Report 聚合报告

<b>Aggregate Report</b>								
<b>Name:</b> Aggregate Report								
<b>Comments:</b>								
<b>Write results to file / Read from file</b>								
<b>Filename</b>		\\Blog\\Jmeter\\测试结果分析\\Do Login_1.jtl			<b>Browse...</b>		<b>Log/Display Only:</b> <input type="checkbox"/> Errors <input checked="" type="checkbox"/> Successes	
Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Thro
Do Login	10	418	156	2985	63	2985	90.00%	
TOTAL	10	418	156	2985	63	2985	90.00%	

我们可以看到，通过这份报告我们就可以得到通常意义上性能测试所最关心的几个结果了。

Samples -- 本次场景中一共完成了多少个 Transaction

Average -- 平均响应时间

Median -- 统计意义上的响应时间的中值

90% Line -- 所有 transaction 中 90% 的 transaction 的响应时间都小于 xx

Min -- 最小响应时间

Max -- 最大响应时间

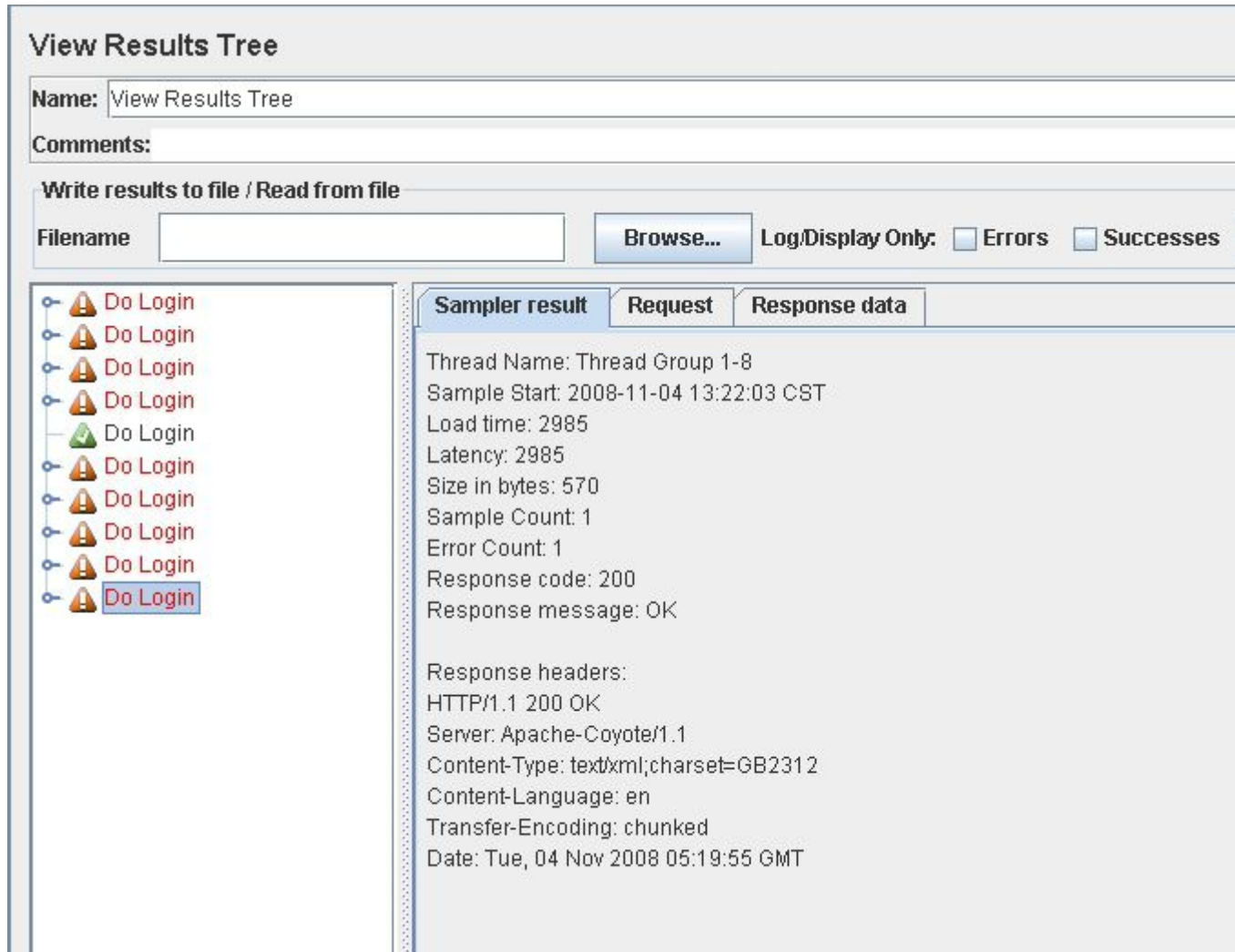
PS：以上时间的单位均为 ms

Error -- 出错率

Troughput -- 吞吐量，单位：transaction/sec

KB/sec -- 以流量做衡量的吞吐量

## B. View Results Tree 以树状列表查看结果



通过这个 Listener，我们可以看到很详细的每个 transaction 它所返回的结果，其中红色是指出错的 transaction，绿色则为通过的。

如果你测试的场景会有很多的 transaction 完成，建议在这个 Listener 中仅记录出错的 transaction 就可以了。要做到这样，你只需要将 Log/Display: 中的 Errors 勾中就可以了。

## 二、.jtl 文件的分析

在性能测试过程中，我们往往需要将测试结果保存在一个文件当中，这样既可以保存测试结果，也可以为日后的性能测试报告提供更多的素材。

Jmeter 中，结果都存放在 .jtl 文件。这个 .jtl 文件可以提供多种格式的编写，而一般我们都是将其以 csv 文件格式记录，这样做是因为 csv 文件格式看起来比较方便，更重要的是这样做可以为二次分析提供很多便利。

我这里所说的二次分析是指除了使用 Listener 之外，我们还可以对 .jtl 文件进行再次分析。



#### a. 设置 jtl 文件格式

我们从 jmeter 官方网站中下载下来的 Jmeter 解压后是可以直接使用的。但是，使用默认配置生成的 jtl 文件内容并不能满足我们的需要。于是 我们必须进行必要的设置。在 2.2 版本中，如果要修改 jtl 设置必须要到 jmeter.properties 文件中设置；但是在 2.3 版本中，我们只需要 在界面上设置就可以了。你只需要选择某个 Listener，点击页面中的 configure 按钮。此时，一个设置界面就会弹出来，建议多勾选如下 项：Save Field Name, Save Assertion Failure Message。

#### b. jtl 文件中的各项

经过了以上设置，此时保存下来的 jtl 文件会有如下项：

timeStamp, elapsed, label, responseCode, responseMessage, threadName, dataType, success, failureMessage, bytes, Latency

请求发出的绝对时间，响应时间，请求的标签，返回码，返回消息，请求所属的线程，数据类型，是否成功，失败信息，字节，响应时间

其中聚合报告中的，吞吐量=完成的 transaction 数/完成这些 transaction 数所需要的时间；平均响应时间=所有响应时间的总和/完成的 transaction 数；失败率=失败的个数/transaction 数

温馨提示：在 jmeter2.2 和 2.3 版本中，都存在的一个问题是当我们重新打开 jmeter，使用某个 Listener 来查看 jtl 文件 时，jmeter 是会报错的。因此当你使用命令行方式完成了一个场景的测试后，你得到的只是一堆保存在 jtl 文件中的原始数据。所以知道聚合报告中的各项 的来源是可以方便大家摆脱测试工具来进行结果的分析。