

## 1. 简介

Apache JMeter 是 100% 纯 java 桌面应用程序，被设计用来测试 C/S 结构的软件（例如 web 应用程序）。它可以被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 可以用来模拟一个在服务器、网络或者对象上大的负载来测试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造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。他写出它起初是为了测试 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 来增强用户界面和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危机 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 测试时使用的所有文件。

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

其它你可能需要下载的软件：

- [BeanShell](#)
- [Java Activation Framework](#) - JavaMail 需要
- [Java Mail](#) - mail 显示 and SOAP 测试需要
- [JMS](#) - JMS 取样器
- [General Java download page](#)



详细参见安装的 jar 包中的 JMeter Classpath 一章

下一步，开始使用 JMeter 并且参见用户手册构建测试计划一章使自己更加熟悉 JMeter 基础（例如，添加和删除元素）。

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，那就参见构建一个 Web 测试计划。其他测试计划的细节是 JDBC，FTP，and JNDI。

一旦你熟练构建和执行 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素的配置（定时器，监听器，断言，和其他）。

### 2.1 需求

JMeter 需要运行环境匹配的最小需求。

#### 2.1.1 Java 版本

JMeter 需要一个完整适当的 JVM 1.3 或更高的版本。我们现在尽力与 JVM 1.3 保持兼容，然而 JMeter 在 1.4 或者更高运行的会更好。

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。



Java 1.3 不包括 SSL (HTTPS) 支持 - 你将需要下载 JSSE。同样，它不会像其他更高版本的 Java 那样好的运行。为了更好的结果使用 Java 1.4 或者 1.5。

### 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 JDK1.3 或者更高。

### 2.2.2 SAX XML 解析器

JMeter 使用 Apache's 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。它能够发送给你测试结果的 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 jmeter.properties。

为了更好的管理证书，也要有一个 SSL 管理器。



#### 注意

如果你在 JDK1.4 上运行，你将不需要下载 JSSE，因为 SUN 已经集成它到 JDK1.4 中做为标准类库了。

JMeter 代理服务器(见下)不支持记录 SSL(https)。

### 2.2.5 JDBC 驱动

你需要添加你的厂商的 JDBC 驱动到 classpath，如果你需要 JDBC 测试。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。



详细参见安装的 jar 包中的 JMeter Classpath 一章

## 2.3 安装

快速安装 JMeter。细节依赖你下载的发布文件。



#### 注意

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。

### 2.3.1 下载最新版本

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。

### 2.3.2 下载夜晚构建

如果你不介意使用 beta 版软件，你可以下载运行[最新夜晚构建](#)。

要安装一个夜晚构建，解压\_bin 和\_lib zip/tar 文件到相同的目录结构。保证一个 JRE/JDK 正确的安装并且设置环境变量 JAVA\_HOME，JMeter 就可以正确的运行了。

## 2.4 运行 JMeter

要运行 JMeter，运行 jmeter.bat (for Windows) 或者 jmeter (for Unix) 文件。JMeter 必须从 JMeter 的 bin 目录（那些文件没有发现的地方）启动。如果 jmeter.bat 文件能够的话，它试图改变到一个适当的目录。

### 2.4.1 JMeter Classpath

JMeter 自动从在它的/lib 和 /lib/ext 目录中的 jar 包发现类。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 将会自导发现在这里的任何 jar 包的 JMeter 组件。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 给那些有用的 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。



#### 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

如果你在防火墙/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：	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 JDK1.3 时候。	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	No



	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。	No
search_paths	列出那些 JMeter 搜索 JMeter 附加类的路径(以;分割);例如增加的取样器。被添加到 lib/ext 目录的任何 jar 包都被发现。	No
user.classpath	JMeter 搜索的有用类库的路径列表。被添加到 lib 目录的任何 jar 包都被发现。	No
user.properties	附加的 JMeter 属性文件名。初始化属性文件后它们被添加，但是在-q 和-J 选项被处理之前。	No
system.properties	附加的系统属性文件名。-S 和-D 选项被执行前被添加。	No

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，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“list”列表中选择一个新元件。或者，元件从文件加载并且通过选择“open”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“remove”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“open”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“save”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，你能够保存测试树段，单独元件，或者这个测试计划。

#### 3.3 配置树元件

在测试树中的任何元件控制在 JMeter 的右手结构。那些控制允许你配置测试元件的细节行为，什么被配置为一个依赖元件类型的元件。



可以通过拖拉测试树周围的元件操作测试树。

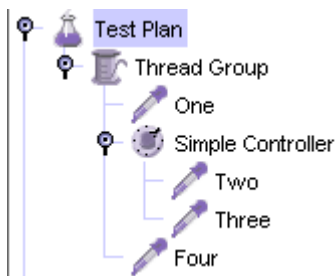
#### 3.4 运行一个测试计划

为了运行一个测试计划，从“run”菜单项选择“start”。为了停止你的测试计划，从同样的菜单选择“stop”。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“run”菜单。如果“start”不可用，“stop”可用，证明 JMeter 正在运行你的测试计划(或者，至少，它认为它是)。

#### 3.5 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，配置元素，后置处理器，前置处理器，断言，定时器)，一些主要是顺序的(控制器，取样器)。当你创建你的测试计划时，你

将创建一个描述被执行的步骤集的取样请求有序列表。那些请求常组织在也有序的控制器中。给出如下测试树：

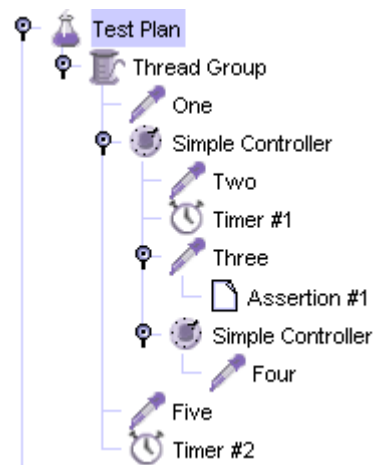


Example test tree

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 看到详细的控制器。

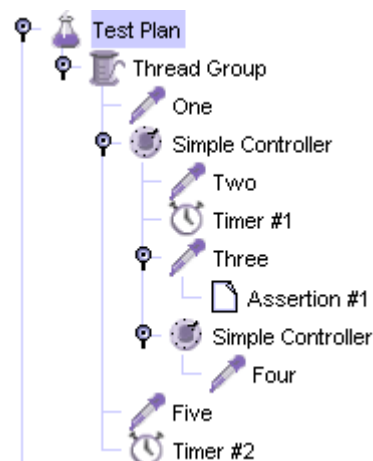
其他元素是分等级的。例如，一个断言在测试树中是分等级的。如果你的父元件是请求，它就被应用于那个请求。如果它的父元件是控制器，它就影响所有那个控制器下的所有请求。如下测试树：



Hierarchy example

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

其它例子，这次使用定时器：



complex example

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 Header Manager, Cookie Manager 和 Authorization manager 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并入取样器到达的值的集里。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。

[Comments](#) ([Hide](#))

### 3.6 Error reporting

#### 3.6 錯誤報告

JMeter reports warnings and errors to the jmeter.log file, as well as some information on the test run itself. Just occasionally there may be some errors that JMeter is unable to trap and log; these will appear on the command console. If a test is not behaving as you expect, please check the log file in case any errors have been reported (e.g. perhaps a syntax error in a function call).

JMeter 把警告和錯誤訊息回報在 jmeter.log 這個檔案中，就像測試本身在執行時產生的某些資訊。只是偶爾地，JMeter 對於某些錯誤是無法捕捉和記錄的，這些資訊都會顯示在執行命令台上。如果一個測試的執行並不是你所期待的，那麼當錯誤發生時，請你檢查記錄檔（例如：也許在函數的調用上有語法上的錯誤）。

Sampling errors (e.g. HTTP 404 – file not found) are not normally reported in the log file. Instead these are stored as attributes of the sample result. The status of a sample result can be seen in the various different Listeners.

取樣錯誤（例如：HTTP 404 – 找不到檔案）是不會被正常的記錄在記錄檔中的，取而代之的，他們會被當作取樣結果的屬性來儲存，取樣結果的狀態能被許多不同的監聽器所得知。

## 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，它将导致 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。你尝试一个小的运行来保证 JMeter 配置正确并且你的服务器正在返回期望的结果是很有用的。

### 4.8 后置处理器元件

一个后置控制器在一个取样器请求被建立后执行一些操作。如果一个后置处理器附属一个取样器元件，它仅在取样器元件运行后执行。后置处理器最多用来处理响应数据，常用来从它里面摘录数值。见范围规则 关于前置处理器执行细节

### 4.9 执行顺序

1. 定时器 – 任何个
2. 取样器
3. 后置处理器（如果 SampleResult 不为空）
4. 断言（如果 SampleResult 不为空）
5. 监听器（如果 SampleResult 不为空）

## 5. 创建一个网站测试计划

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网站，你将会创建 5 个用户向 Jackrta 网站上的两个网页发送请求。当然，你也可以让每个用户发送两次。这样，总的 HTTP 发送请求为（5 个用户\*2 次请求\*重复 2 次）=20。要构建这个测试计划，你将会用来下面的元素：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的测试计划网站。

### 5.1 添加用户

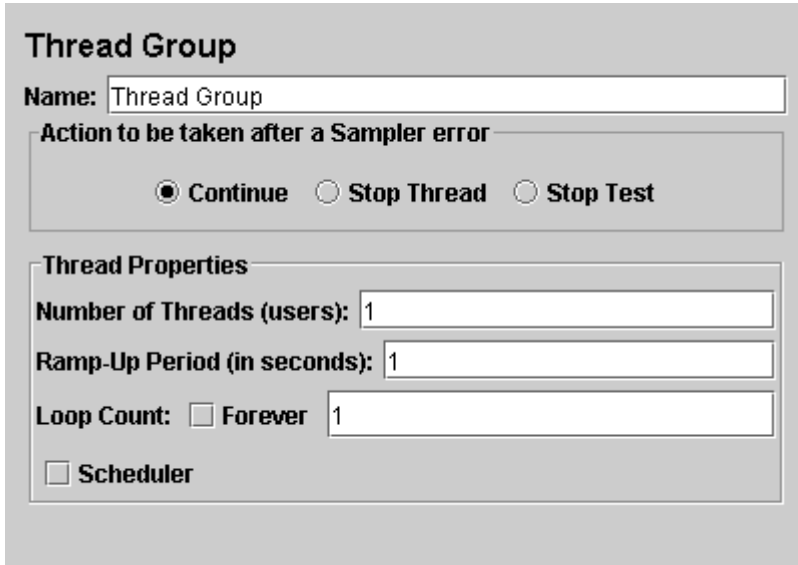
处理每个 JMeter 测试计划的第一步就是添加 线程组元件。这个线程组会告诉 JMeter 你想要模拟的用户数量，用户应该发送请求的频率和应该发送的数量。



进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，增加用户的数量为 5。

在下一个 the Ramp-Up Period 文本域中，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，那么开始用户的延迟就是 1 秒。(5 个用户 / 5 秒 = 1 用户每秒)。JMeter 将会立即启动你所有的用户，如果你设置其值为 0。

最后，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，树型菜单自动更新当你离开控制面板后。（例如，当你选择另外一个树元件。）

图 5.2 为完整的 Jakarta Users 线程组。

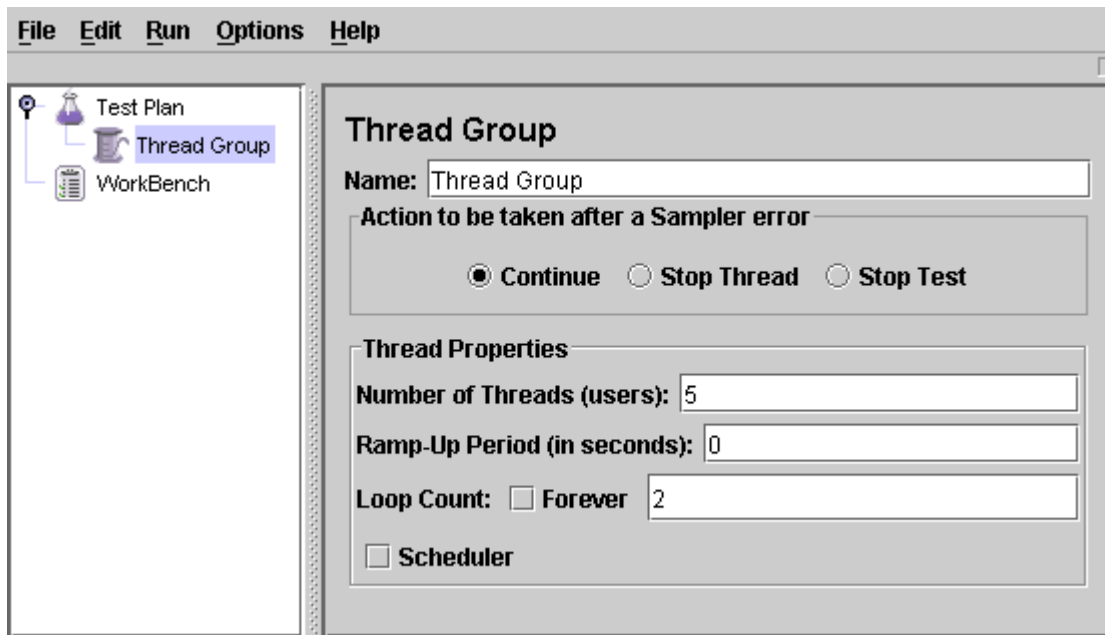


图 5.2. Jakarta Users 线程组

## 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 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）。

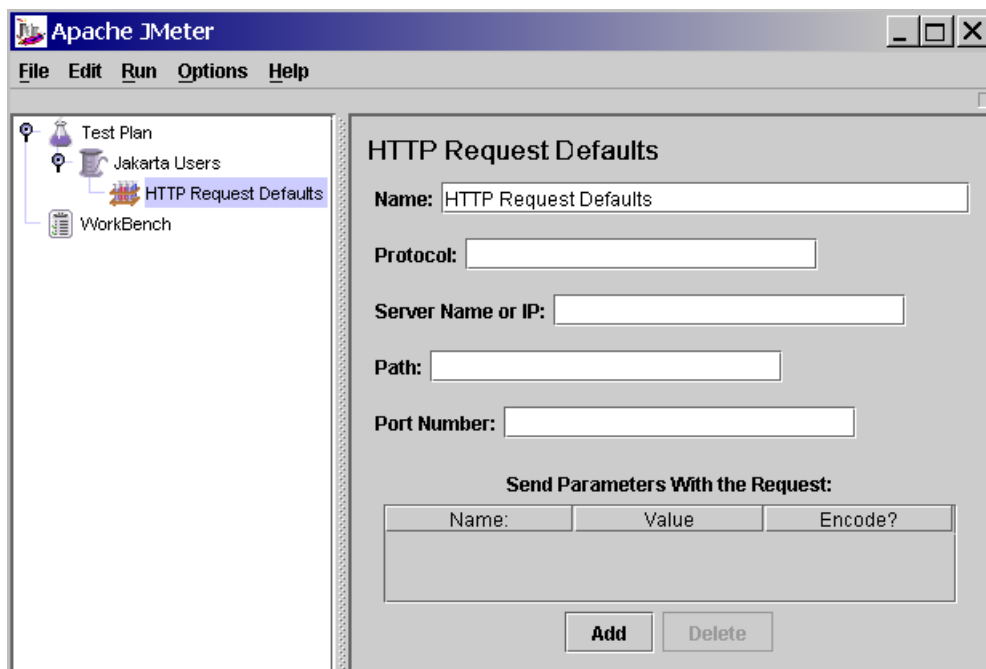


图 5.3. HTTP 请求默认值

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 Server 的 Server 名字/IP。对于这个测试计划中，所有的 HTTP 请求都将发送到相同的网站服务器 jakarta.apache.org。向文本域中输入名字，这是唯一的一个需要我们去修改它的默认值，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。

图 5.4 表示为完整的 HTTP 请求默认值元件

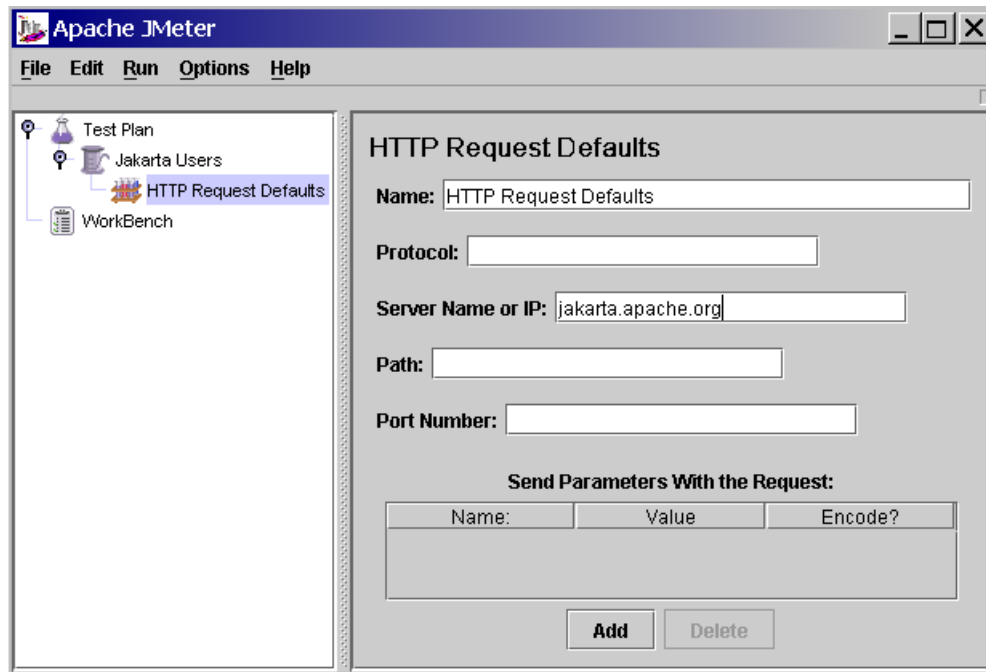


图 5.4. 测试计划的 HTTP 默认值

### 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。这样确信每个线程组有自己的 cookies，但是通过所有交互的 HTTP 请求 对象变成共享。

添加 HTTP Cookie 管理器，简单地，选择这个 线程组，选择添加--> 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 请求元件并修改正面的属性（看图 5.5）：

更改名称域为“Home Page”。

设置路径域为 “/”。



你不必要设计服务器的名称域，因为你已经在 HTTP 请求默认值元件中设定过了。

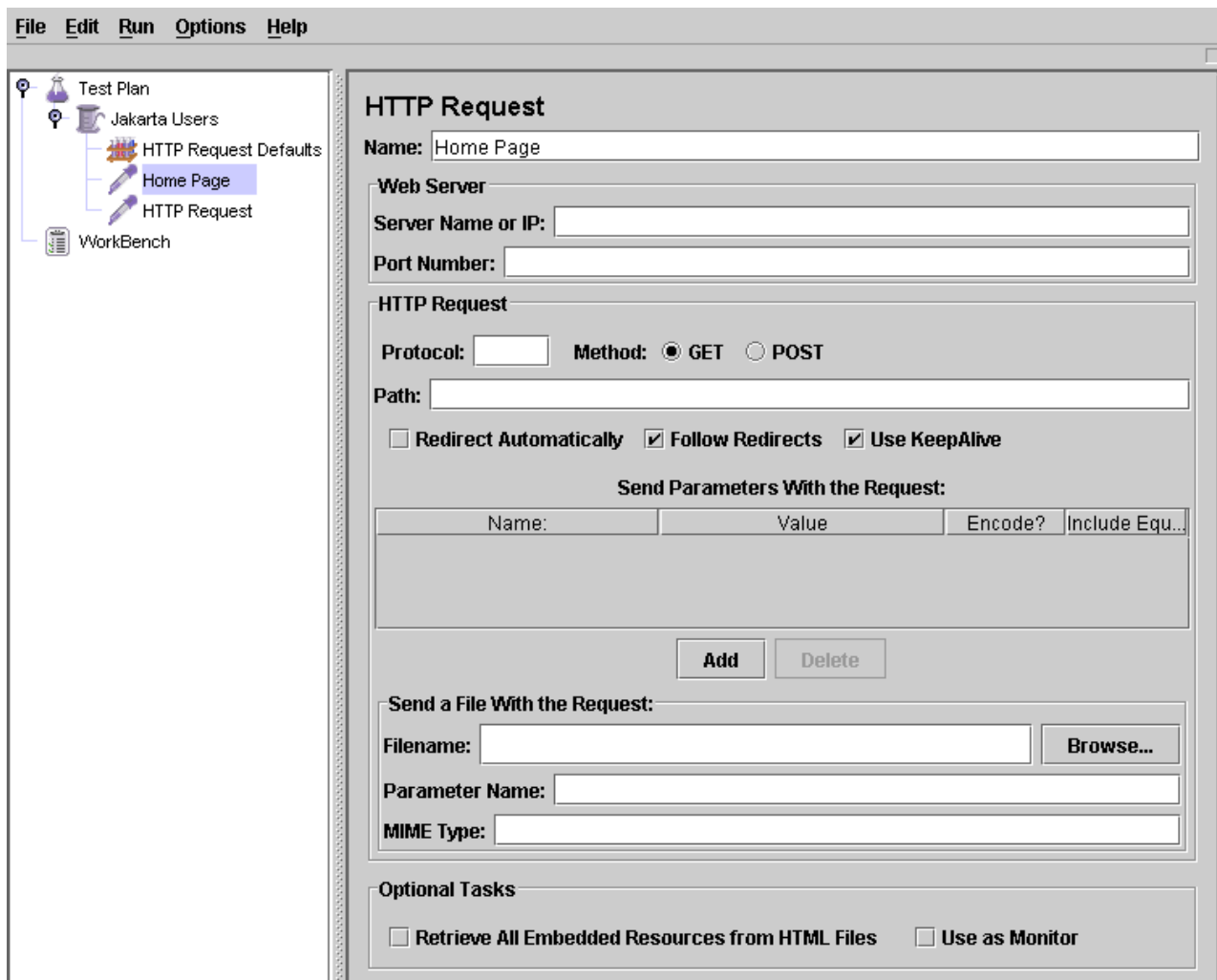


图 5.5. Jakarta 首页的 HTTP 请求

下一步，添加每二个 HTTP 请求并修改下面的属性（见图 5.6）：

更改名称域为“Project Guidelines”。

设置路径域为 “/site/guidelines.html”。

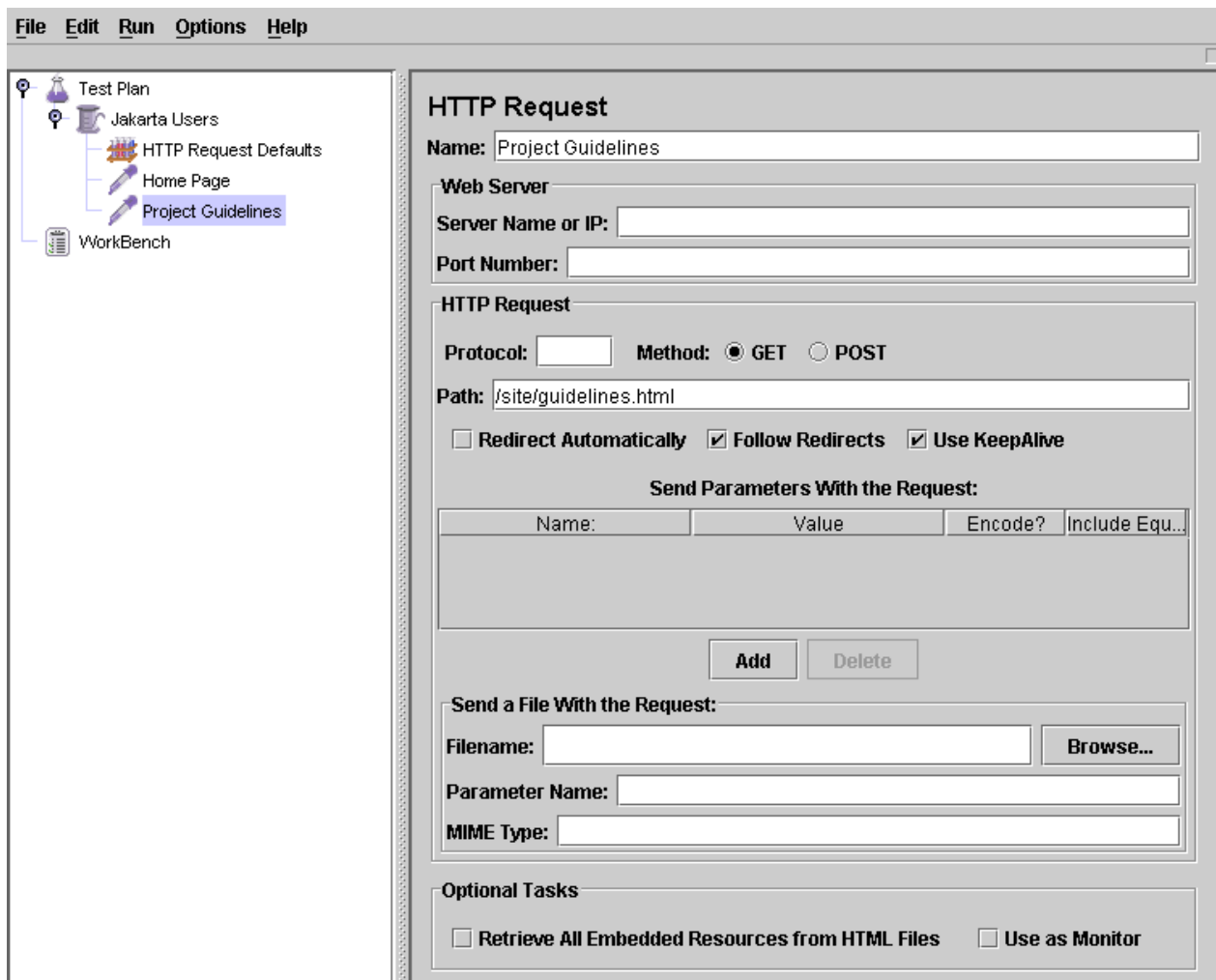


图 5.6. Jakarta 工程 Guidelines 页的 HTTP 请求

### 5.5 添加一个监听器到试图储存测试结果

最后一个你需要给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 → 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。

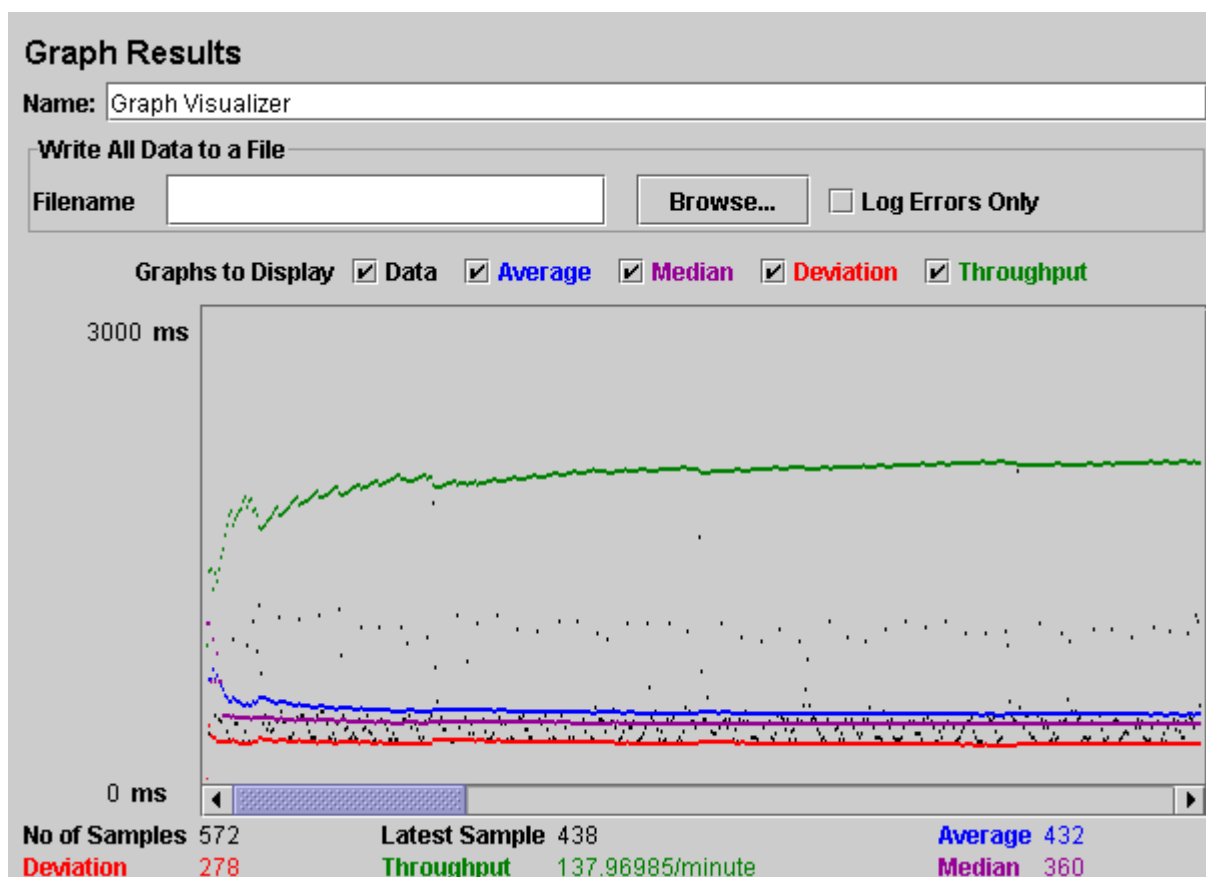


图 5.7. 图像结果监听器

## 5.6 保存测试计划

尽管它并不必要，我们还是建议你在运行测试计划前将它保存在一个文件里边。通过选择文件菜单中的“保存测试计划”来保存（在最新版本中你不需要先选择测试计划元件）。

⚠ JMeter 允许你保存整个测试计划树，也可以只保存其中的一部分。要保存特别树枝中的一些元件，首先选择树枝的起始元件，然后在右键弹出的菜单中选择保存为菜单项。同样的，也可以选择合适的元件，然后选择编辑菜单中的“另存为”。

## 5.7 运行测试计划

从 Run 菜单中选择 Run。

⚠ 如果测试运行正确，JMeter 会在上方显现一个绿色的长方形区域。当所有的测试结束时，它将会变成灰色。即使在你选择了“stop”后，这个绿色的灯还将保持，直到所有的线程结束。

一旦 JMeter 已经完成测试计划，选择“run”菜单中的“stop”。

如果你选择了一个文件来保存你监听器中的结果，那么你将有一个文件，它可以在任何的视图中打开。每一个视图将以它自己的样子显示结果。

⚠ 相同的文件可以在多个视图中打开，这是没有问题的。在测试运行期间，JMeter 确信没有例子被多次保存在同一个文件中。

## 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 测试 web 站点。



如果需要一个基础的[测试计划](#)例子，见 [构建一个 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 重写优于 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。

为了响应正确到 URL 重写，JMeter 需要解析从服务器接受的 HTML 和重新得到唯一的会话 ID。利用适当的 HTTP URL 重写修改器 来完成这些。简单地输入你的会话 ID 参数名到修改器，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“Cache Session Id?”被选中，那么最后被发现的会话 ID 将被保存，并且如果 HTTP 的上次取样不包含一个会话 ID 将会被使用。

URL 重写例子

下载 [这个例子](#)。在图 1 展示了一个使用 URL 重写的测试计划。注意 URL 重写修改器附属于线程组，因此确定它对在那个线程组的每一个请求有效。

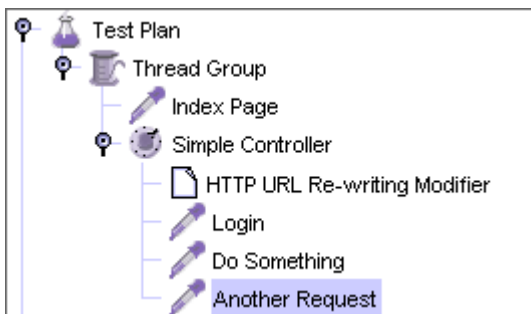


图 1 - 测试树

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

HTTP 消息头管理 让你定制 JMeter 在 HTTP 请求消息头发送的信息。这个消息头包括像“User-Agent”，“Pragma”，“Referer”等属性。

HTTP 消息头管理 好像 HTTP Cookie 管理，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，可以被添加到线程组水平。

## 7. 创建一个数据库测试计划

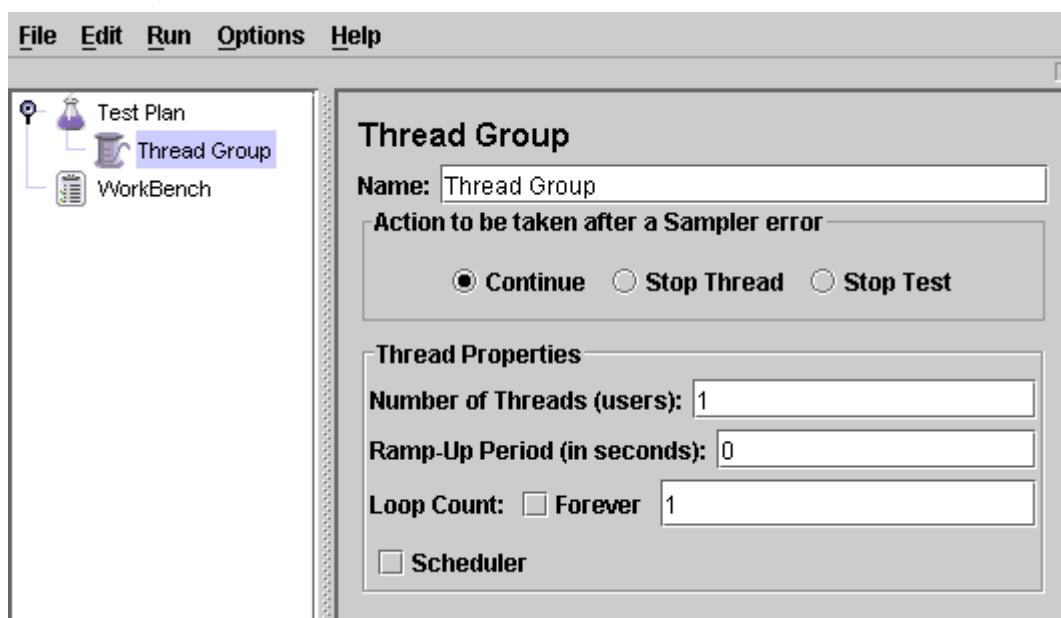
在这一部分，你将学会如何去创建一个基础的测试计划来测试一个数据库服务器。你会创建 10 个用户来给数据库服务器发送 2 次 SQL 请求。同样，你也可以让用户运行他们的测试三次。这样总的 JDBC 请求数量就是  $(10 \text{ 用户}) * (2 \text{ 次请求}) * (\text{重复 } 3 \text{ 次}) = 60$ 。要构建这个测试计划，你将会用到下面的元件：线程组，JDBC 请求，图形结果。

⚠ 这个例子使用了 MySQL 数据库驱动。要使用这个驱动，它所包涵的 .jar 文件必须复制到 `./lib/directory` 下(详情参见 JMeter's ClassPath)。另外我们期望在运行这个测试计划的时候有的堆栈跟踪数量。

## 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。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(见图 7.1)。



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

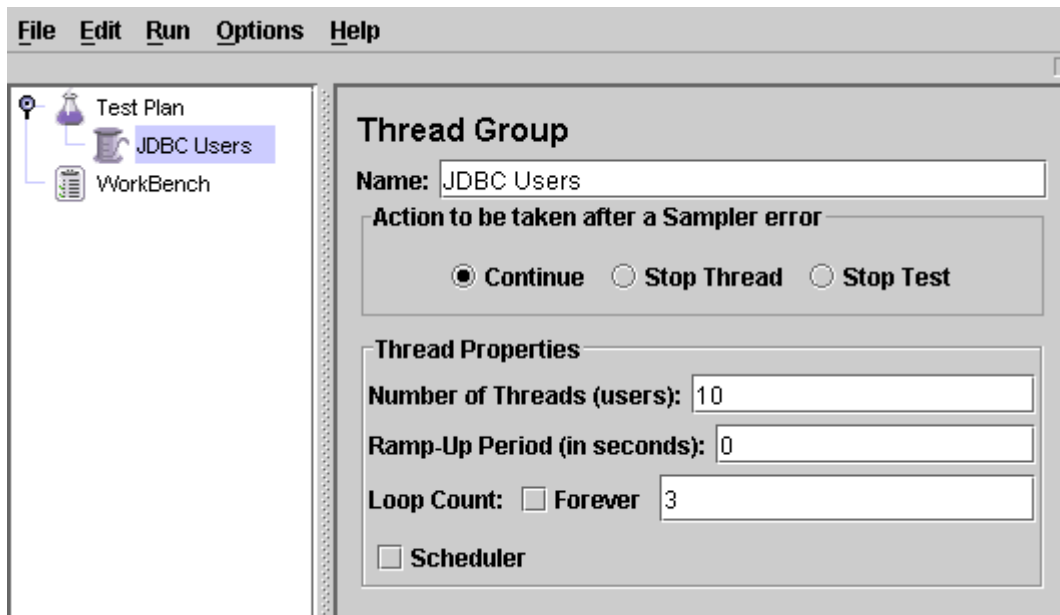
⚠ 你将需要一个可用的数据库，数据库表，和表的用户使用权限。在这个例子中，数据库是 'mydb'，表名是 'Stocks'。

接下来，将用户的数量（即 threads）增加不 10。在下一个 the Ramp-Up Period 文本域中，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，那么开始用户的延迟就是 1 秒。 $(5 \text{ 个用户} / 5 \text{ 秒} = 1 \text{ 用户每秒})$ 。JMeter 将会立即启动你所有的用户，如果你设置其值为 0。最后，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，树型菜

单自动更新当你离开控制面板后。（例如，当你选择另外一个树元件。）

图 7.2 为完整的 JDBC Users 线程组。



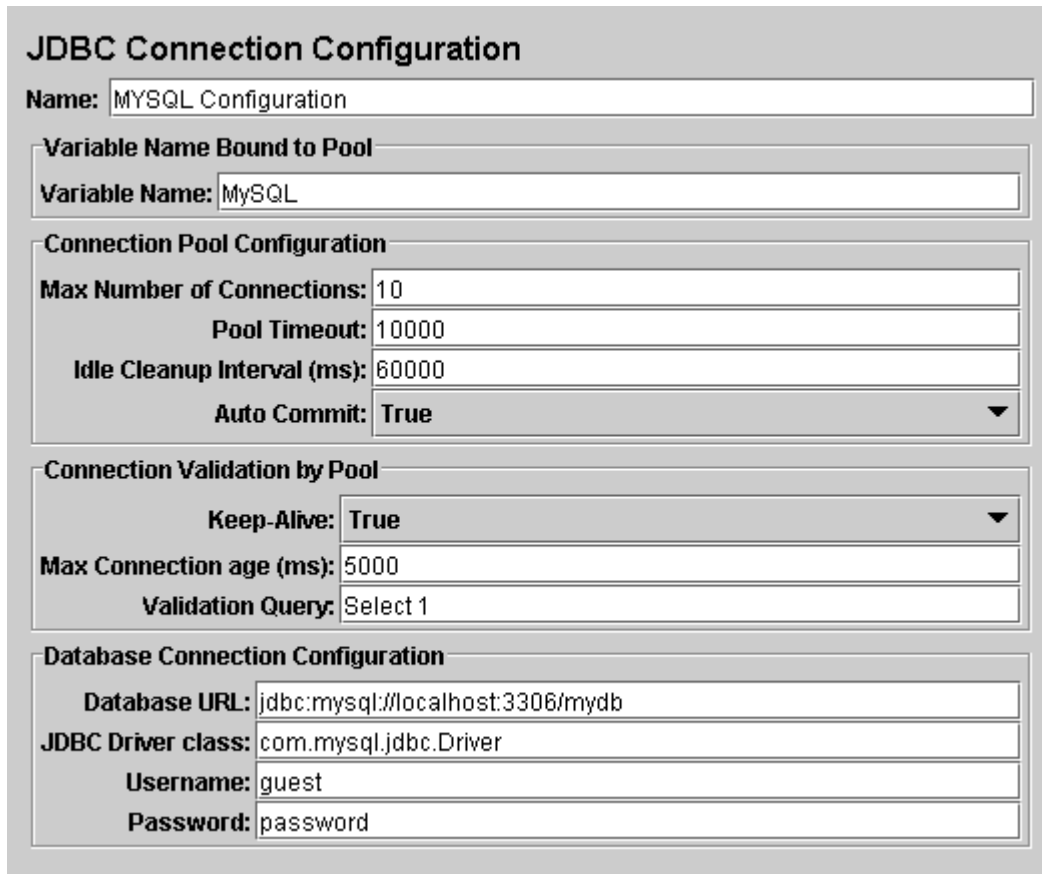
## 7.2 添加 JDBC 请求

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，我们将会详细说明 JDBC 请求。

首先选择 JDBC 用户元件，右键点击，在弹出的菜单中选择 Add --> Config Element --> JDBC Connection Configuration。然后，选择这个新的元件来显示它的控制面板（见图 7.3）。设定下面的文本域的值（我们这里假定用一个本地的 MySQL 数据库名为 test）。

- Variable name bound to pool. 这需要能够唯一标识这个配置。
- Database URL: jdbc:mysql://localhost:3306/test
- JDBC Driver class: com.mysql.jdbc.Driver
- Username: guest
- Password: password for guest

剩下的文本域我们可以保留默认的值。



**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

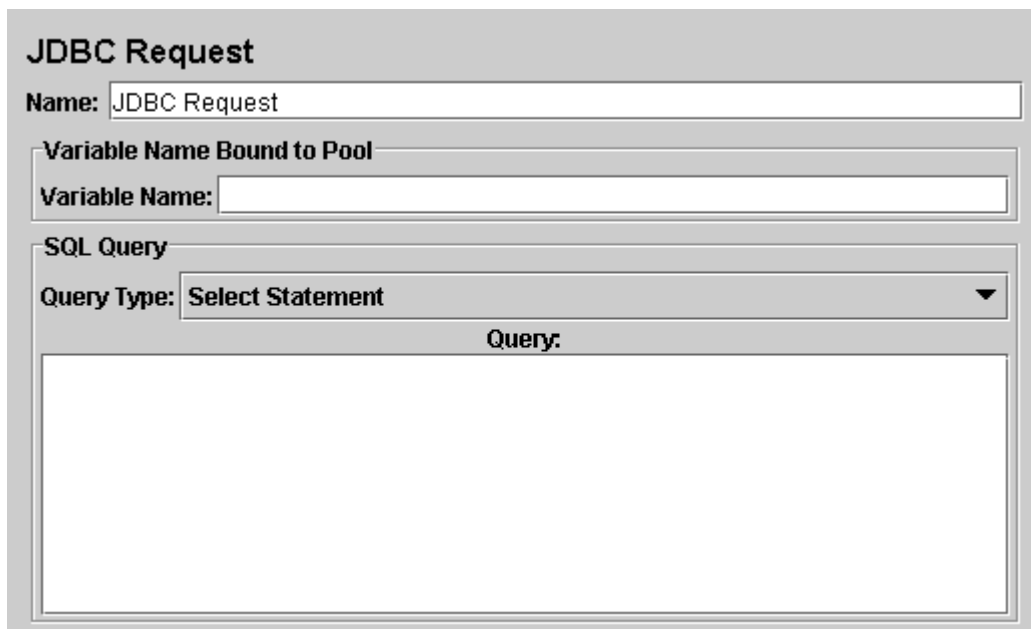
JDBC Driver class: com.mysql.jdbc.Driver

Username: guest

Password: password

Figure 7.3. JDBC Configuration

再次选择 JDBC 用户元件。右键点击，并在弹出的菜单中选择 Add --> Sampler --> JDBC Request。然后，选择一个新的元件来显示其控制面板（见图 7.4）。



**JDBC Request**

Name: JDBC Request

Variable Name Bound to Pool

Variable Name:

SQL Query

Query Type: Select Statement

Query:

Figure 7.4. JDBC Request

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)。下面的插图文字说明。



JMeter 发送请求的次序就是你向树中添加它们的次序。

首先修改下面的属性值（见图 7.5）：

- 修改名字 Name 为“Kodak”
- 输入 Pool Name:MySQL(在配置元件里面一样)
- 输入 SQL Query String(数据库查询字符串)

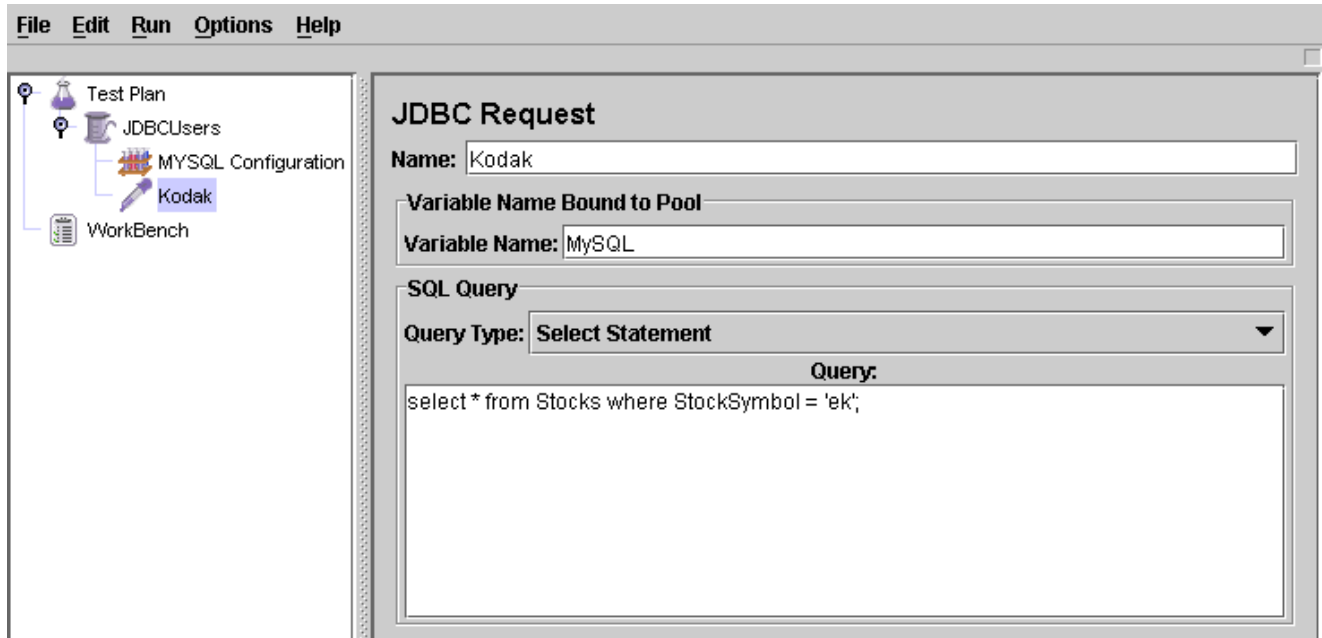


Figure 7.5. JDBC Request for Eastman Kodak stock

然后，添加第二个 JDBC 请求并编辑正面的属性（见图 7.6）：

- 修改名字 Name 为“Pfizer”
- 输入 SQL Query 语句

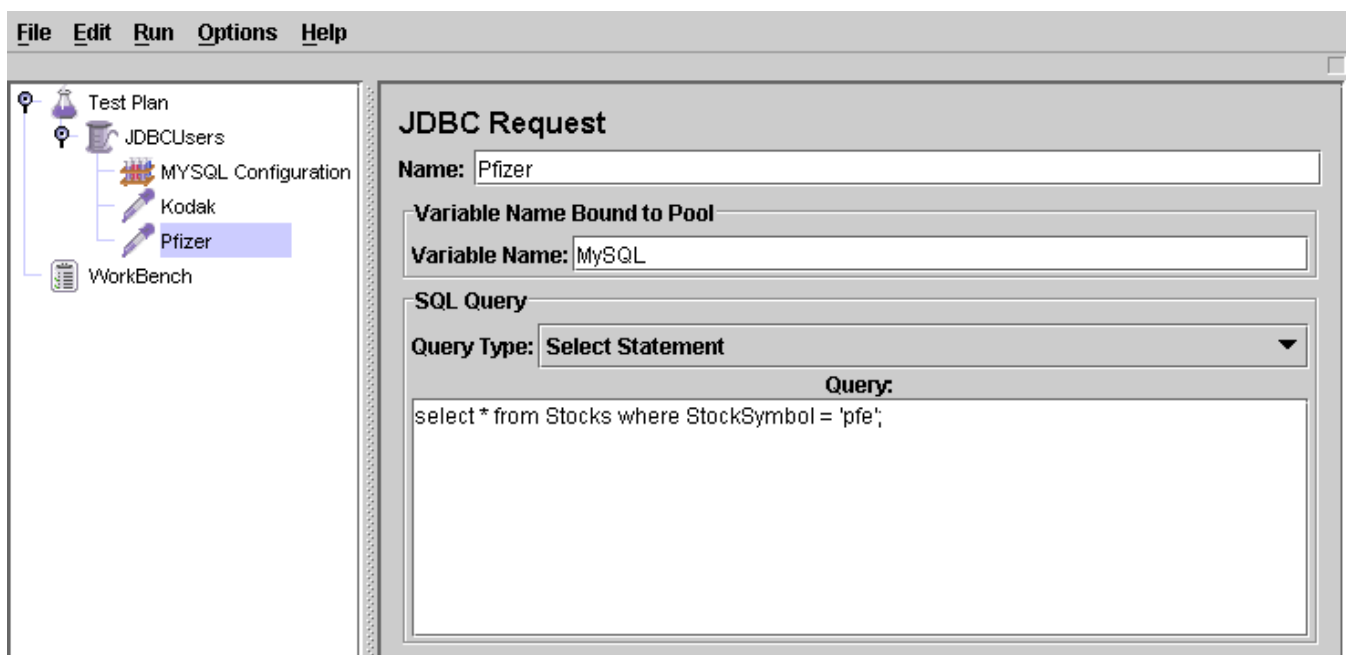


Figure 7.6. JDBC Request for Pfizer stock

#### 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 Users 元件，添加一个 Graph Results 监听器（Add --> Listener --> Graph Results）。

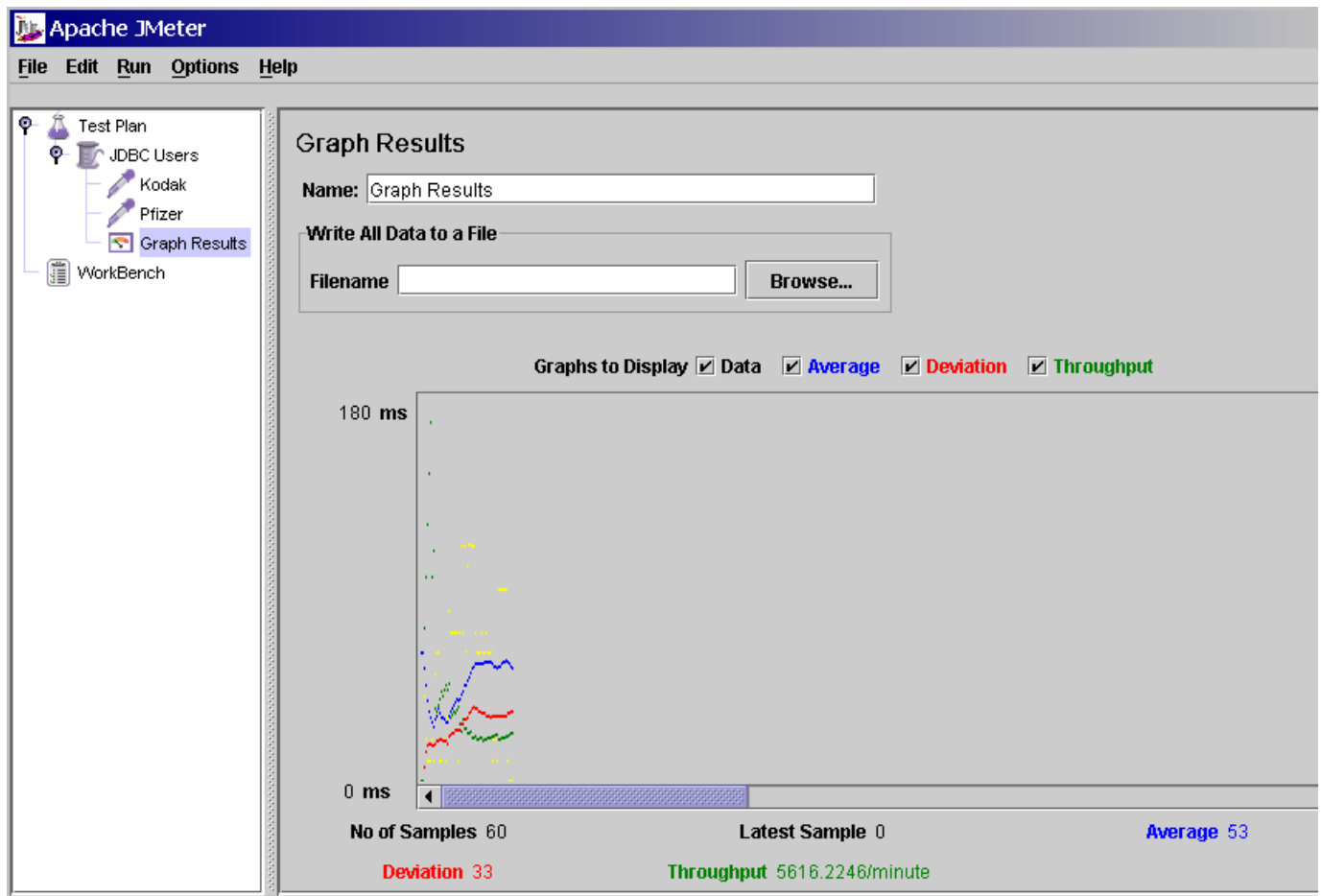


Figure 7.7. Graph results Listener

## 7.5 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。

## 7.6 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。

## 7.7 JDBC 设置

不同的数据库和 JDBC 驱动程序需要不同的 JDBC 设置。JDBC 执行的提供者来定义数据库 URL 和数据库驱动程序类。



下面是一些可能的设置。要得到详细的说明请看 JDBC 驱动程序文档。

Database	Driver class	Database URL
MySQL	com.mysql.jdbc.Driver	jdbc:mysql://host:port/{dbname}
PostgreSQL	org.postgresql.Driver	jdbc:postgresql:{dbname}
Oracle	oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:user/pass@//host:port/service
Ingres (2006)	ingres.jdbc.IngresDriver	jdbc:ingres://host:port/db[:attr=value]



上面的可能不正确，请查看相应的 JDBC 驱动程序文档。

## 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。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 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。

（见下图 8.1）

**Thread Group**

Name: Thread Group

Action to be taken after a Sampler error

☒ Continue ☐ Stop Thread ☐ Stop Test

Thread Properties

Number of Threads (users): 1

Ramp-Up Period (in seconds): 1

Loop Count: ☐ Forever 1

☐ 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 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 8.2 完整的 O'Reilly Users 线程组。

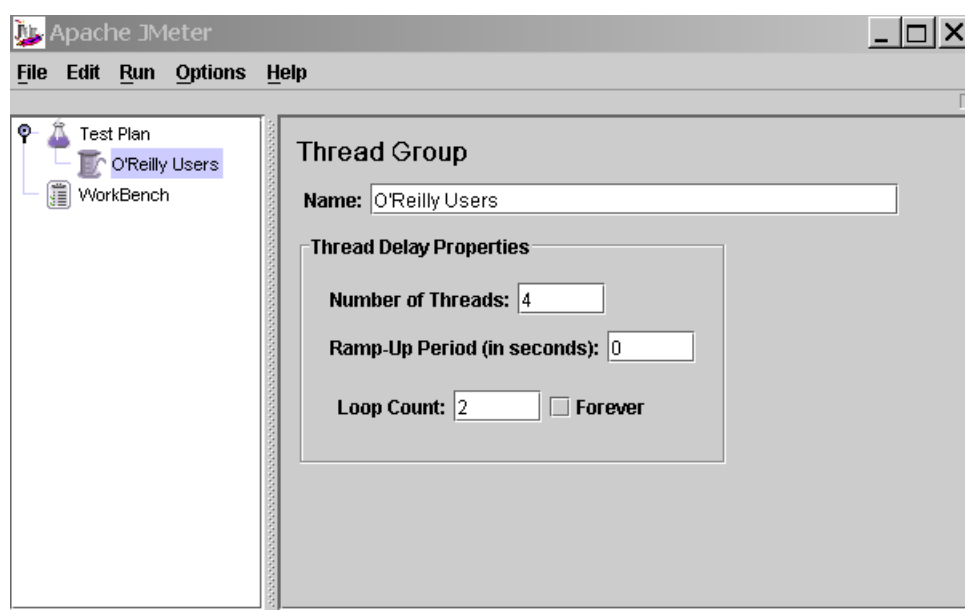


图 8.2 O'Reilly Users 线程组

## 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 Add 菜单, 然后选择 Add --> Config Element --> FTP Request Defaults。于是选择新的元件预览它的控制面板 (见图 8.3)。

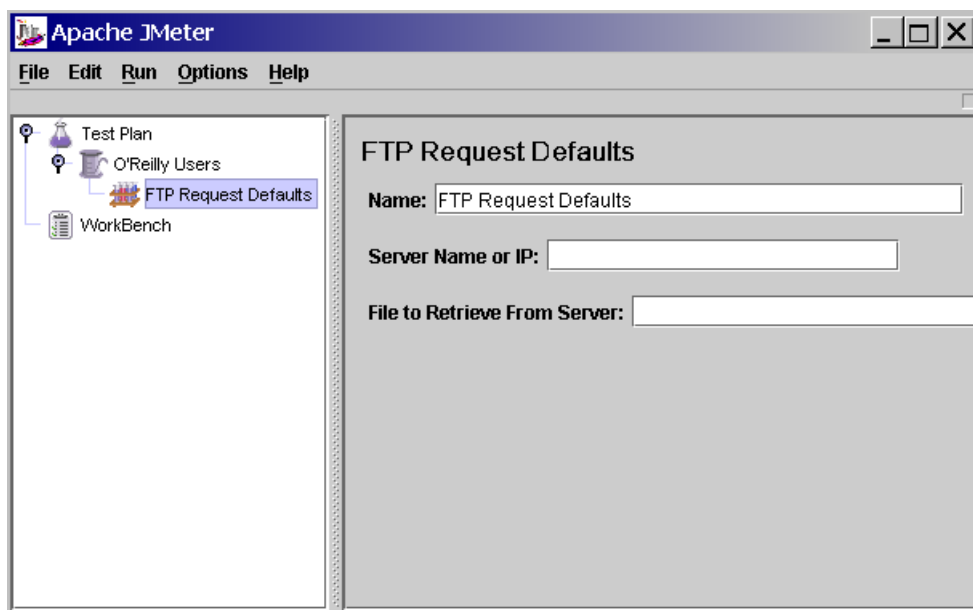


图 8.3 FTP 默认请求

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 name 文本域。在这个例子里，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。输入这个域名到这个文本域。这是我们定制一个默认的唯一文本域，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。

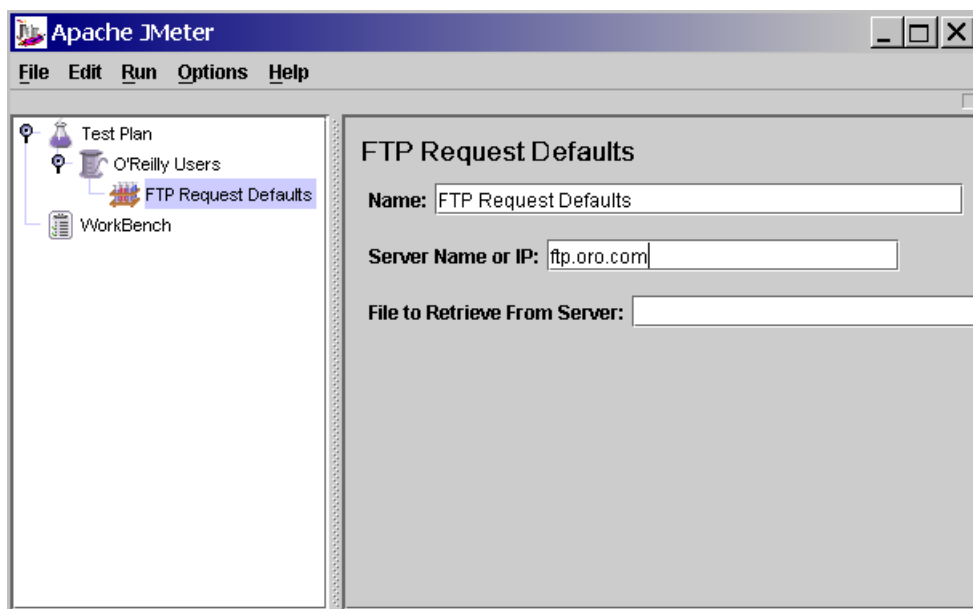


图 8.4 我们测试计划的 FTP 默认

### 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/msql/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 (Add --> Sampler --> FTP Request)。然后早树中选择 FTP 请求元件，并且编辑下列属性 (见图 8.5)：

1. 修改 Name 文本域为“README”。
2. 修改 File to Retrieve From Server 文本域为“pub/msql/java/README”。
3. 修改 Username 文本域为“anonymous”。
4. 修改 Password 文本域为“anonymous”。



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

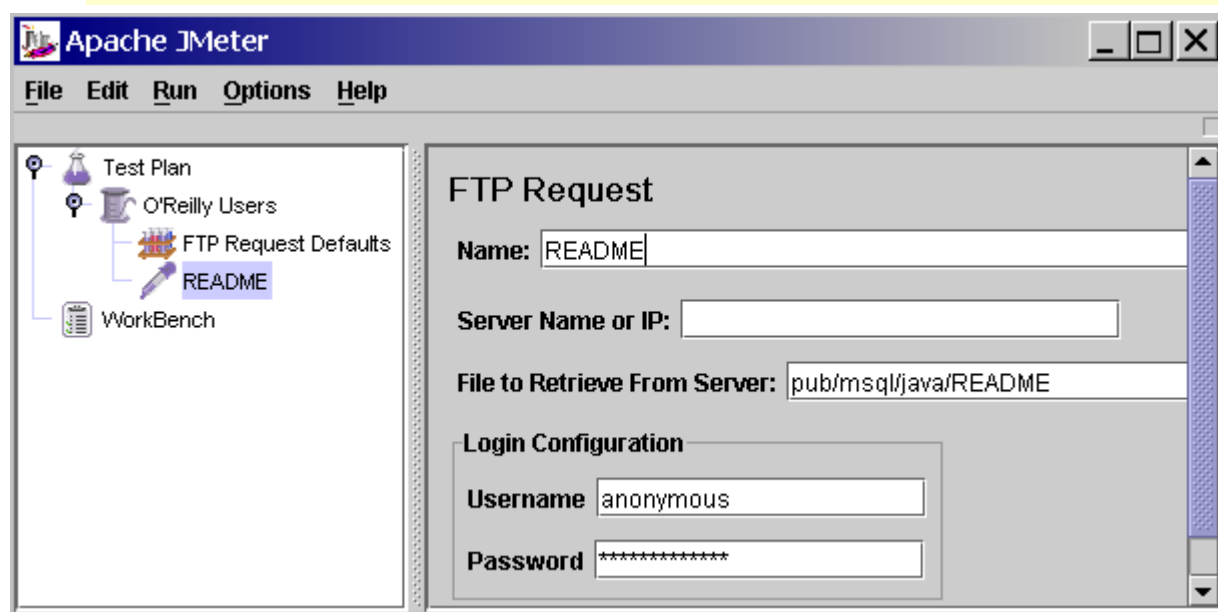


图 8.5 O'Reilly mSQL java README 文件的 FTP 请求

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 Name 文本域为“tutorial”。
2. 修改 File to Retrieve From Server 文本域为“pub/msql/java/tutorial.txt”。
3. 修改 Username 文本域为“anonymous”。
4. 修改 Password 文本域为“anonymous”。

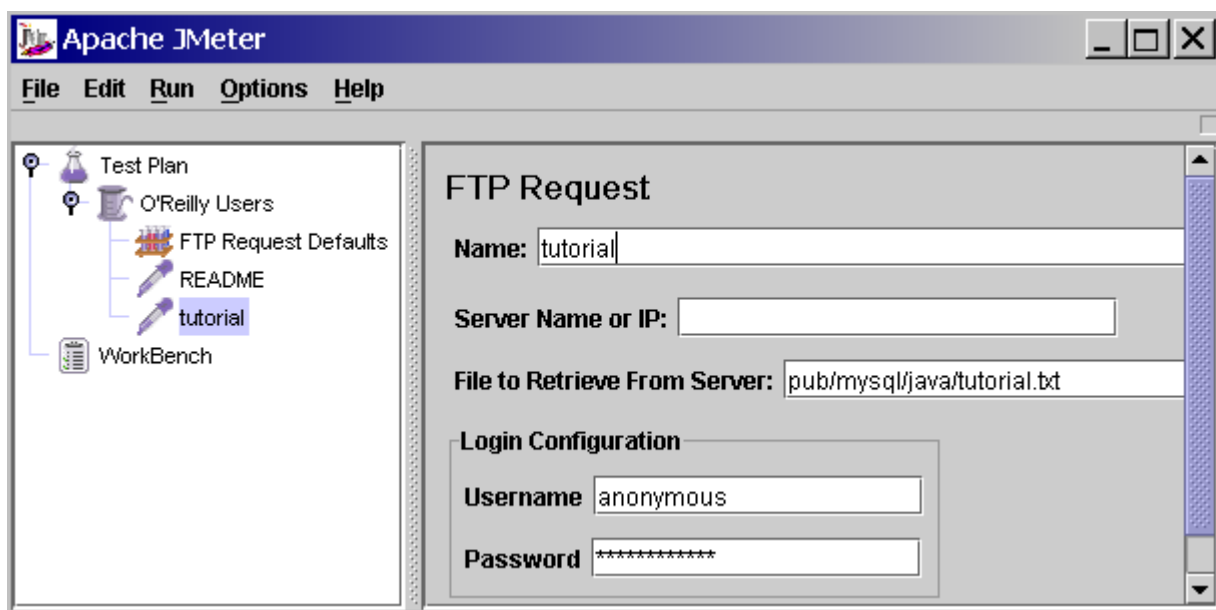


图 8.6 O'Reilly mSQL java tutorial 文件的 FTP 请求

#### 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 (Add --> Listener --> Spline Visualizer)。

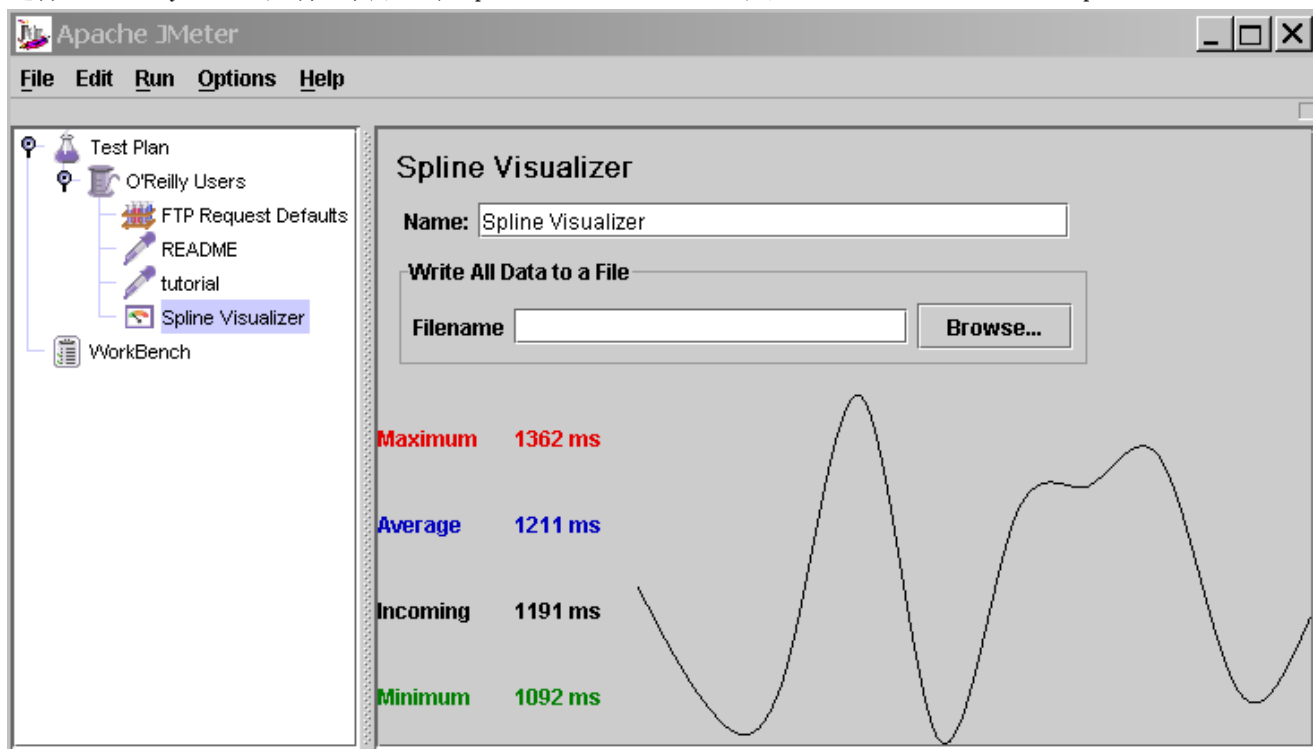


图 8.7 Spline Visualizer 监听器

#### 8.5 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。

## 8.6 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。

## 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，你要告诉用户运行他们的测试两次。所以，整个请求次数是（4 用户）x（4 请求）x（重复 2 次）=32LDAP 请求。构建测试计划，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。

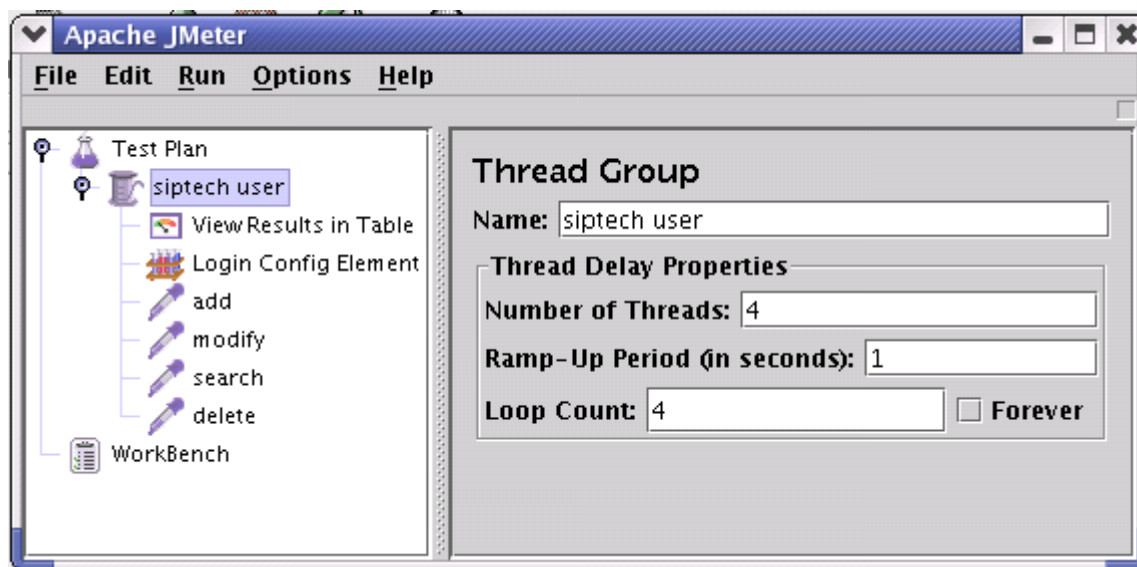


图 9.1 线程组

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-->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。



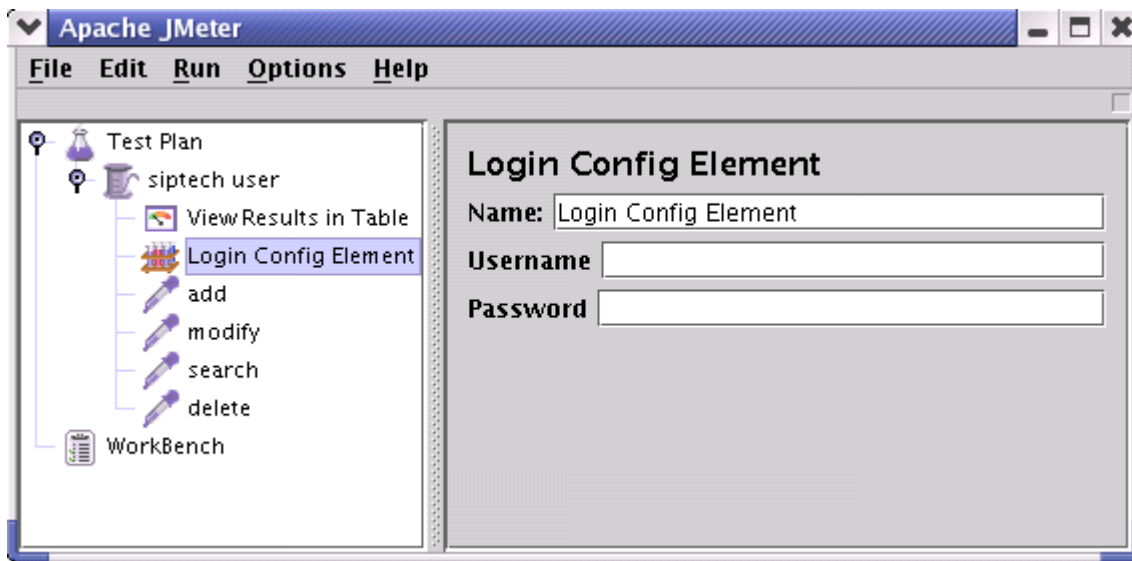


图 9.2 登录配置元素测试计划

置元素测试计划

### 9.3 添加 LDAP 请求默认值

开始选择 Sip tech User 元素。单击鼠标右键得到添加菜单，然后选择添加-->LDAP 请求默认值。选择这个新元素来查看它的控制面板。

像大多 JMeter 元素一样，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。

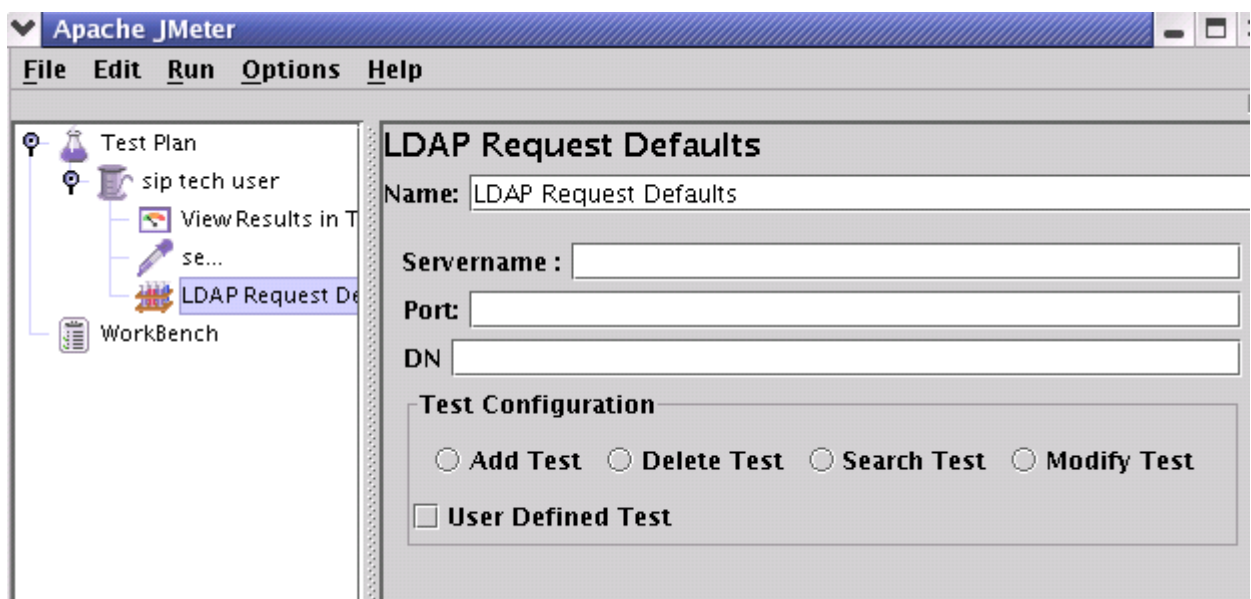


图 9.3

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

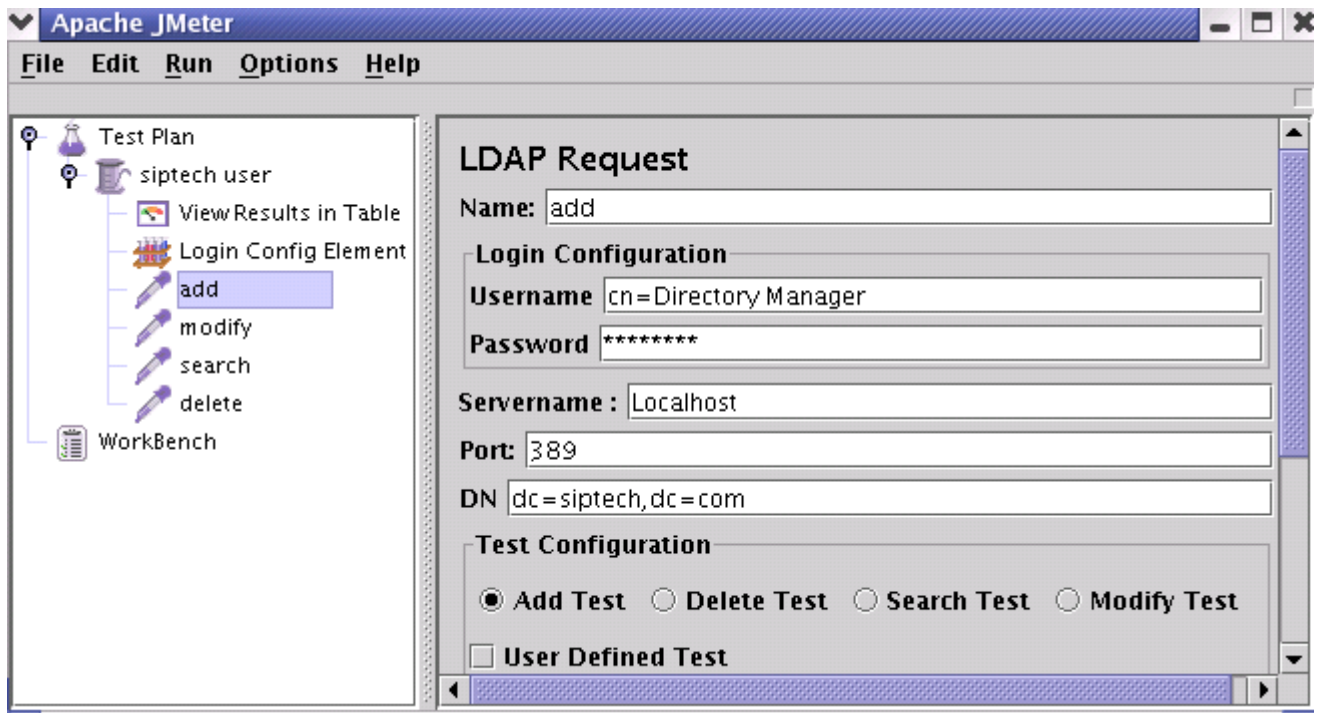


图 9.4.1 Inbuilt Add test LDAP 请求

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

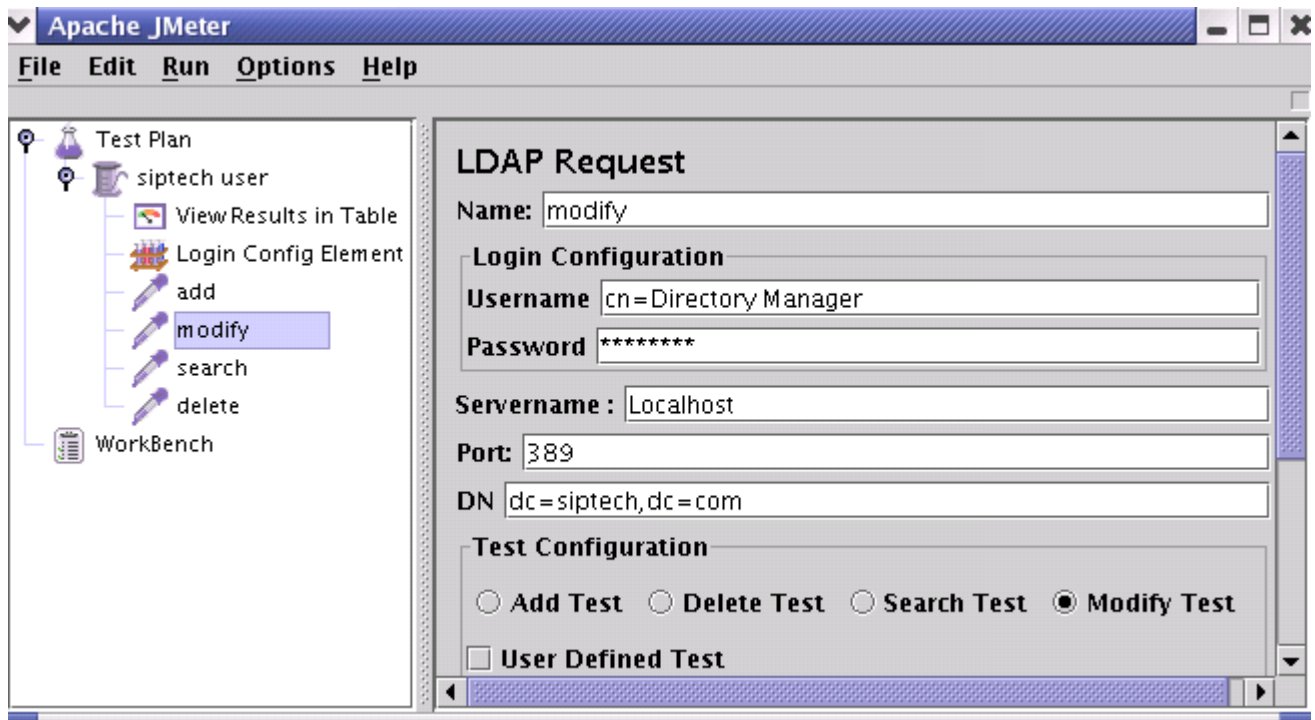


图 9.4.2 Inbuilt Modify test LDAP 请求

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

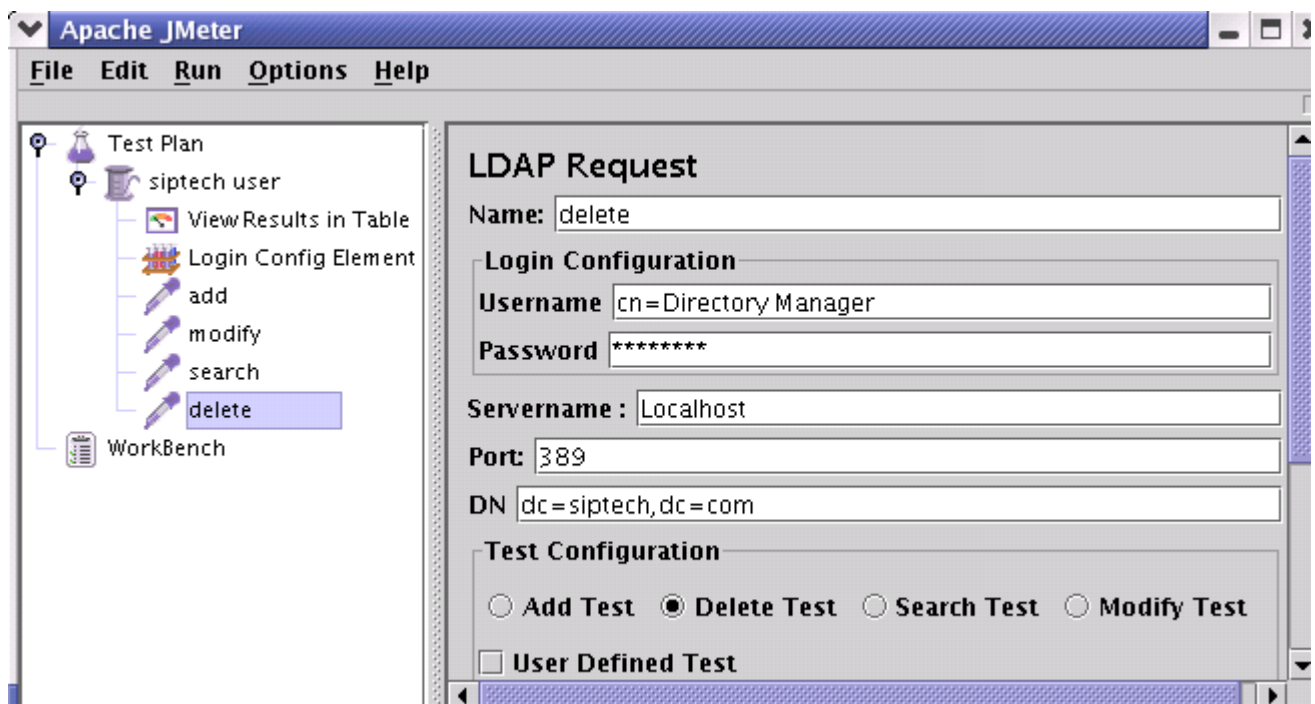


图 9.4.3 Inbuilt-Delete Test LDAP 请求

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

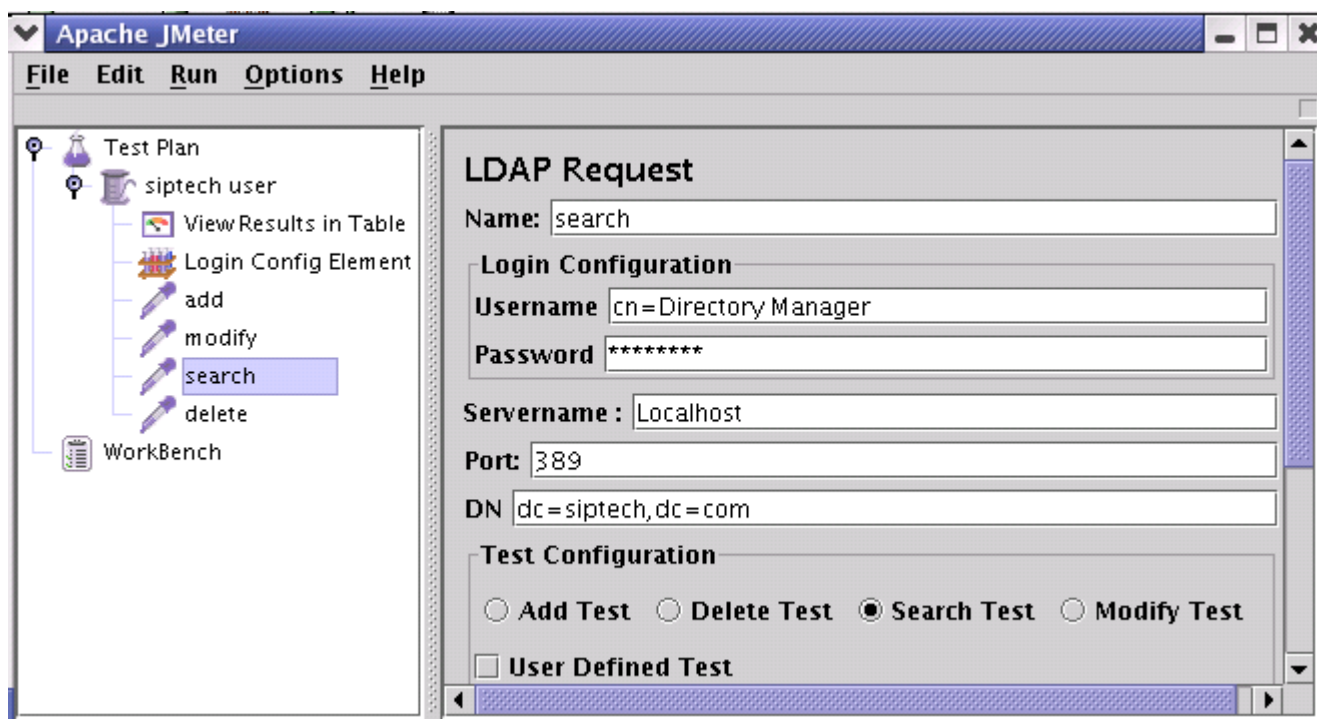


图 9.4.4 Inbuilt-Search Test LDAP 请求

### 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）。

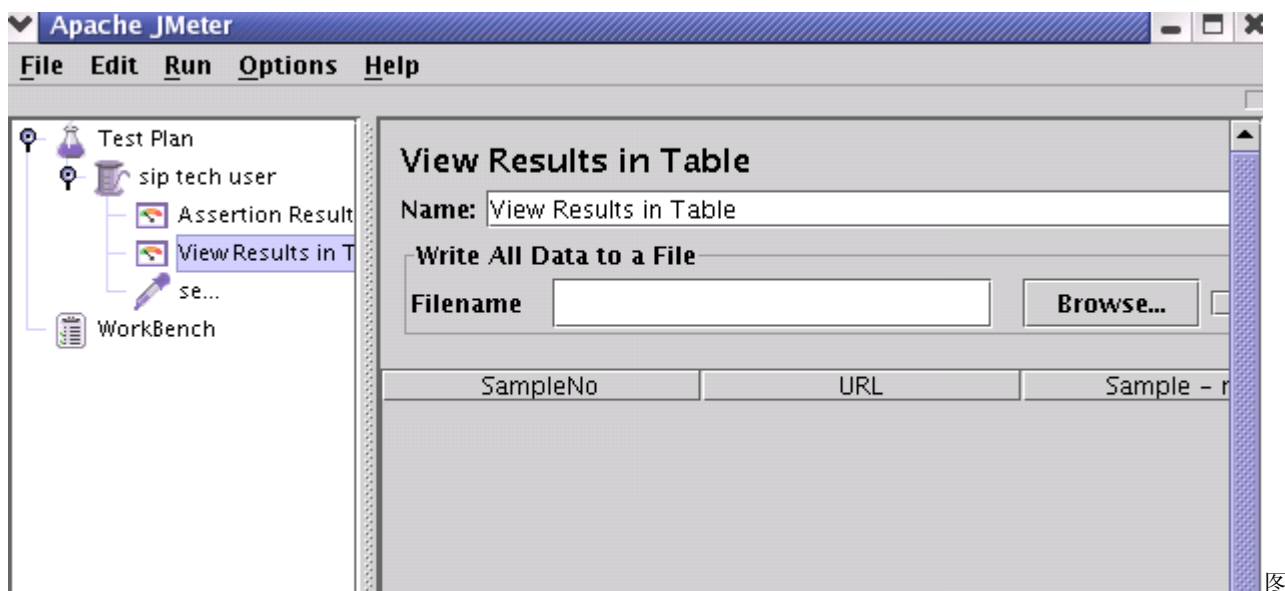


图 9.5 表格视图结果监听器

### 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。

## 9 添加一个 LDAP 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 LDAP 服务器。你会创建 4 个用户来给 LDAP 服务器发送 4 次请求。同样，你也可以让用户运行他们的测试 2 次。这样总的 LDAP 请求数量就是（4 用户）\*（4 次请求）\*（重复 2 次）=32。要构建这个测试计划，你将会用到下面的元件：线程组，LDAP 请求，LDAP 请求默认值，用表格查看结果。

⚠ 这个例子假定你在你的个人机器上已经安装了 LDAP 服务器

### 9.1 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。

进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。

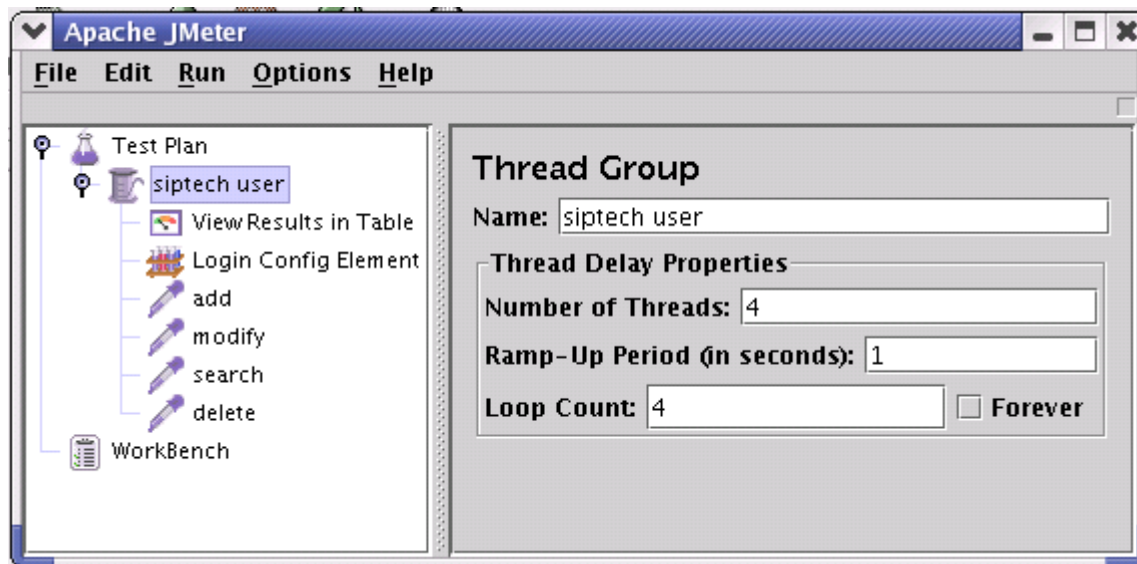


Figure 9.1. Thread Group with Default Values

### 9.2 添加 Login Config Element

首先选择 Siptech Users 元件，右键点击，在弹出的菜单中选择 Add --> ConfigElement --> Login Config Element。然后，选择这个新建的元件使它的控制面板显示出来。

像所有的 JMeter 元件一样，这个 Login Config Element 控制面板有一个名字域需要你来修改，在这个例子，我们取它的默认值。

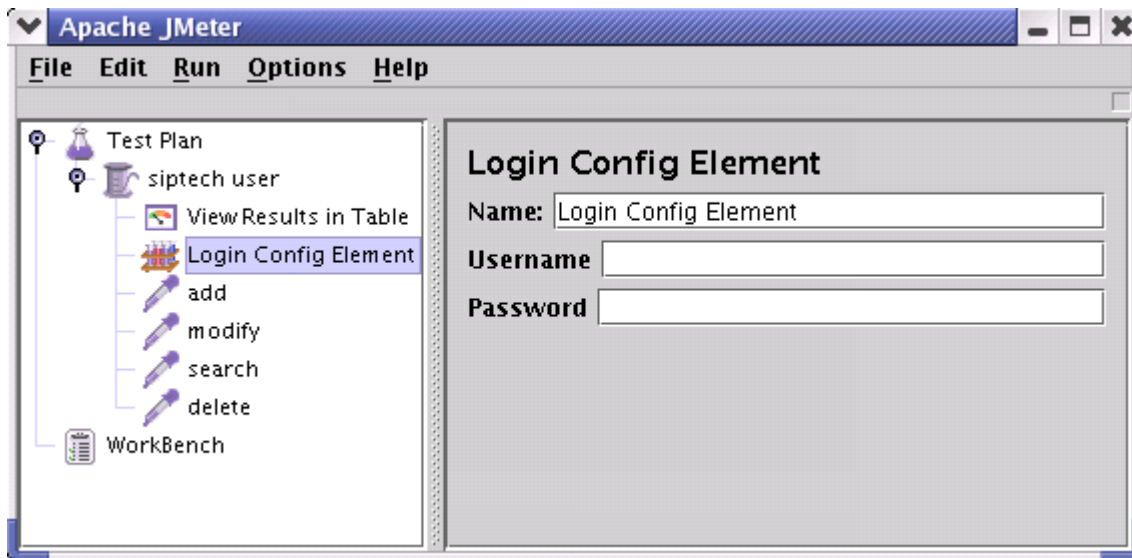


Figure 9.2 Login Config Element for our Test Plan

⚠ 在 UserName 域中输入你的服务器用户名  
在 password 域中输入你的服务器密码  
LDAP 请求的值为默认值。

### 9.3 添加 LDAP 请求默认值

首先选择 Sip tech Users 元件，右键点击，在弹出的菜单中选择 Add --> Config Element --> LDAP Request Defaults。然后，选择这个新的元件使它的控制面板显示出来。

像所有的 JMeter 元件一样，这个 Login Config Element 控制面板有一个名字域需要你来修改，在这个例子，我们取它的默认值。

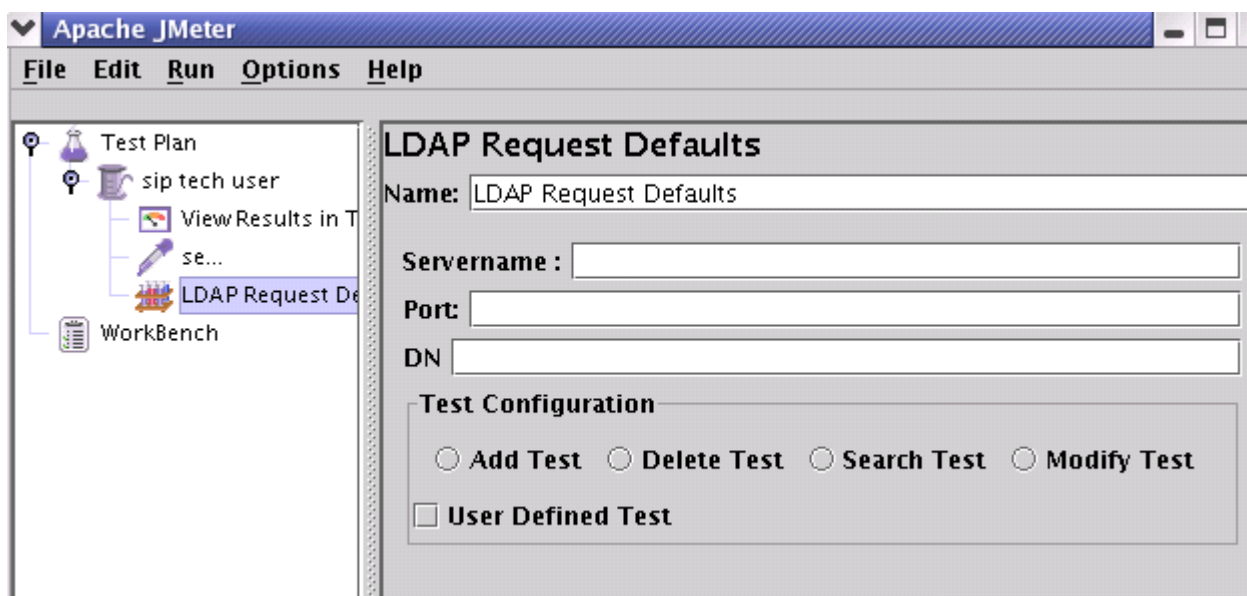


Figure 9.3 LDAP Defaults for our Test Plan

⚠ 在 DN 域中输入你的服务器 Root Dn  
在 LDAP Server's Servername 域中输入 "localhost"  
端口设为 389。



LDAP 请求的值为默认值。

#### 9.4 添加 LDAP 请求

在我们这个测试计划， 我们需要创建 4 个 LDAP 请求。

1. Inbuilt 添加测试
2. Inbuilt 修改测试
3. Inbuilt 删除测试
4. Inbuilt 搜索测试



JMeter 发送请求的次序就是你向树中添加它们的次序。

首先给 Siptech Users 添加第一个 LDAP 请求(Add → Sampler → LDAP Request)。然后，在树型结构中选择这个 LDAP 请求元素修改下面的属性。

1. 修改名字 Name 为“Inbuilt-Add Test”。
2. 选择 Serch Test 单行框。

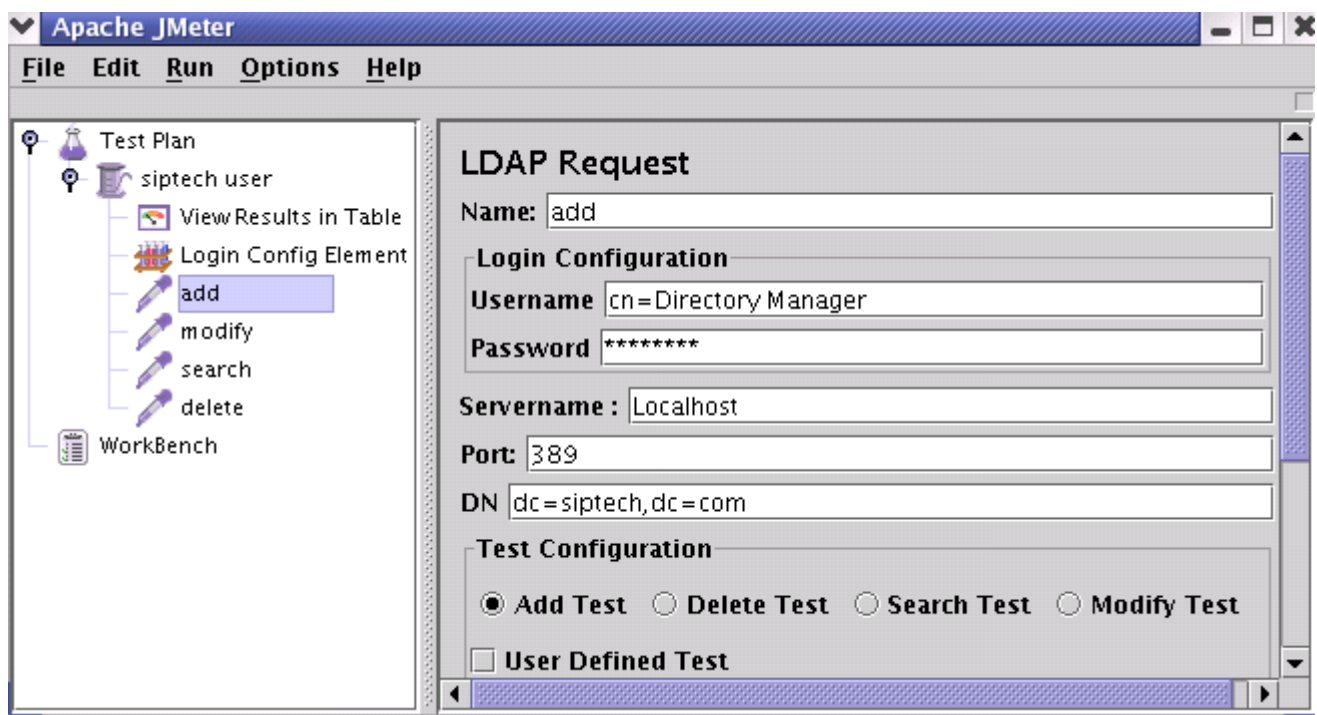


Figure 9.4.1 LDAP Request for Inbuilt Add test

你不必设置 Server Name 域, port 域, Username, Password 和 DN 域，因为你已经在 Login Config Element 和 LDAP 请求默认值中确认了这些值。

下一步，添加第二个 LDAP 请求，并修改下面的属性值。

1. 修改名字 Name 为“Inbuilt-Modify Test”。
2. 选择 Serch Test 单行框。

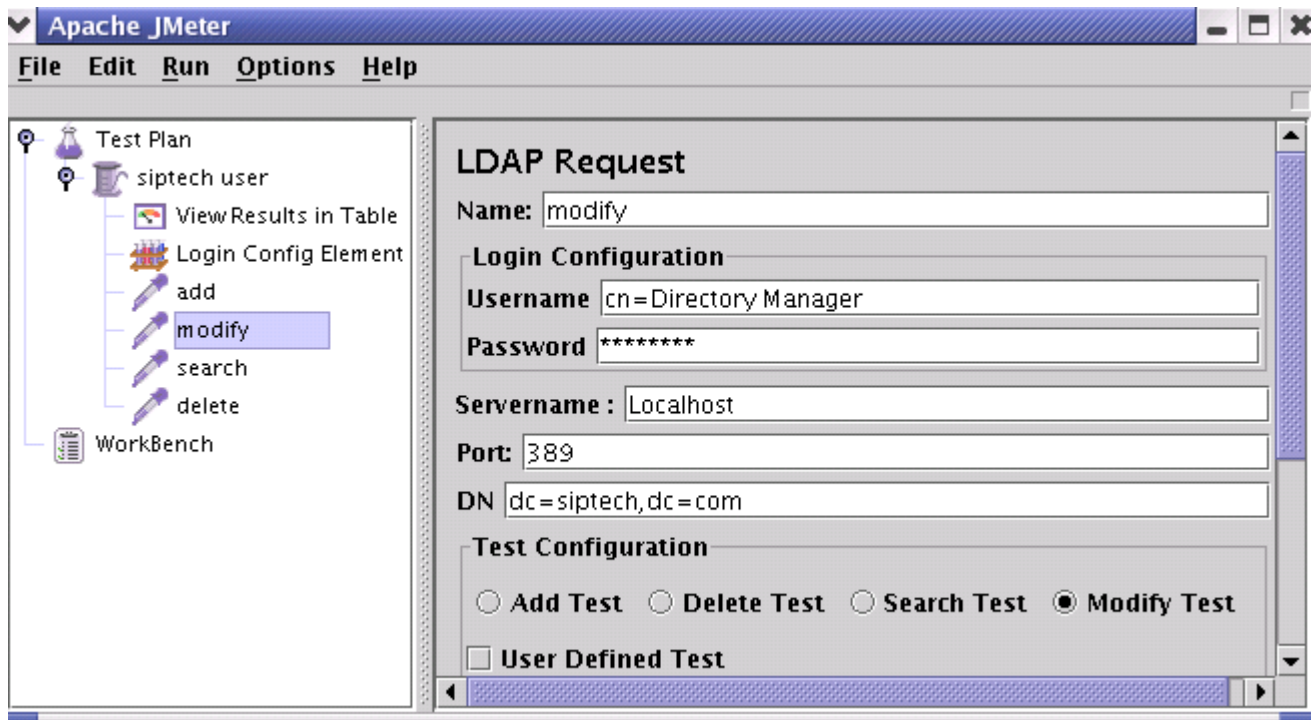


Figure 9.4.2 LDAP Request for Inbuilt Modify test

1. 修改名字 Name 为“Inbuilt-Delete Test”。
2. 选择 Serch Test 单行框。

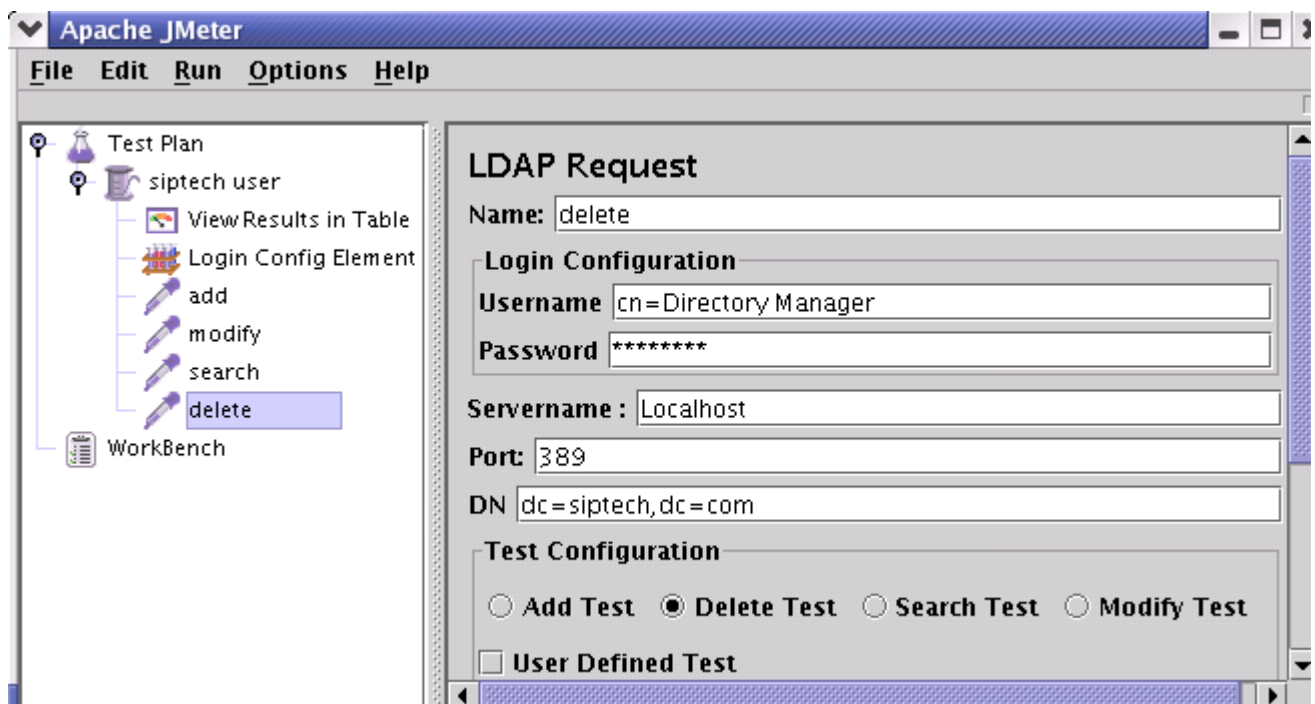


Figure 9.4.3 LDAP Request for Inbuilt Delete test

1. 修改名字 Name 为“Inbuilt-Serch Test”。
2. 选择 Serch Test 单行框。

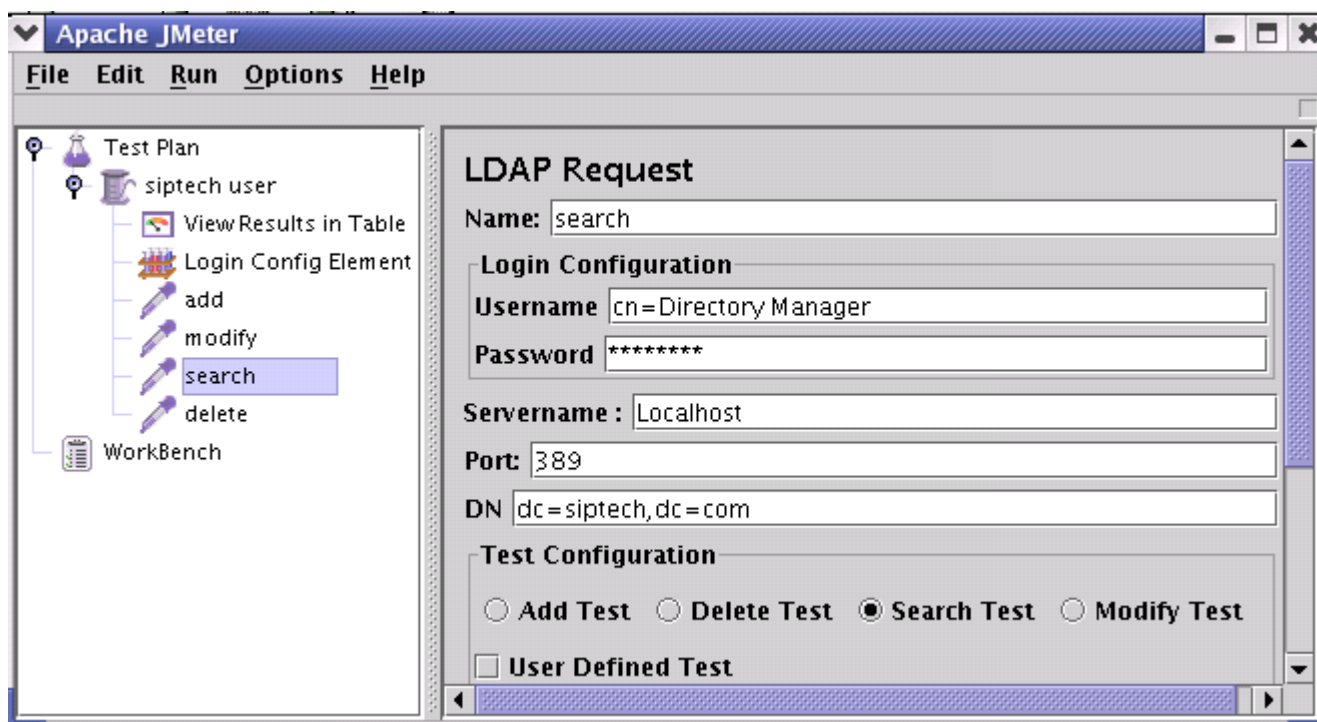


Figure 9.4.4 LDAP Request for Inbuilt Search test

### 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。这个元件责任是储存所有你的 LDAP 请求结果到文件，并且展示一个可视数据模型。

选择 Siptech Users 元件，添加一个 Graph Results 监听器（Add --> Listener --> View Results in Table）。

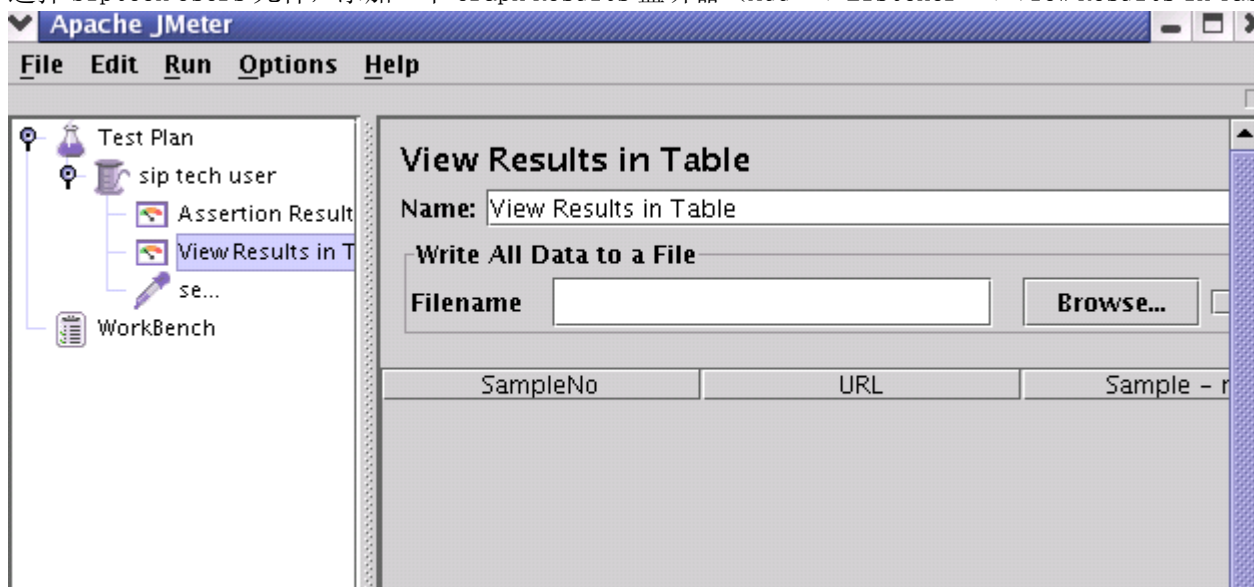



Figure 9.5 View result in Table Listener

### 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 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。

## 9.7 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。

## 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: Thread Group

Action to be taken after a Sampler error

☒ Continue ☐ Stop Thread ☐ Stop Test

Thread Properties

Number of Threads (users): 1

Ramp-Up Period (in seconds): 1

Loop Count: ☐ Forever 1

☐ 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。

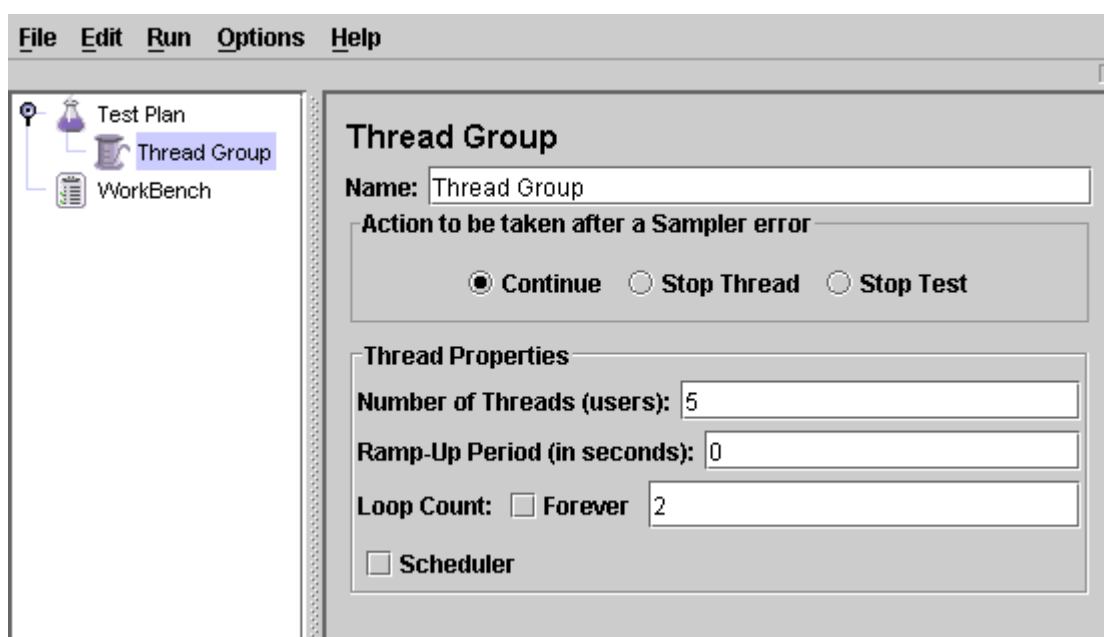


图 10.2 Jakarta Users 线程组

## 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。

### 构建一个 JMS 点对点测试计划

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 点对点和图形结果。

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。

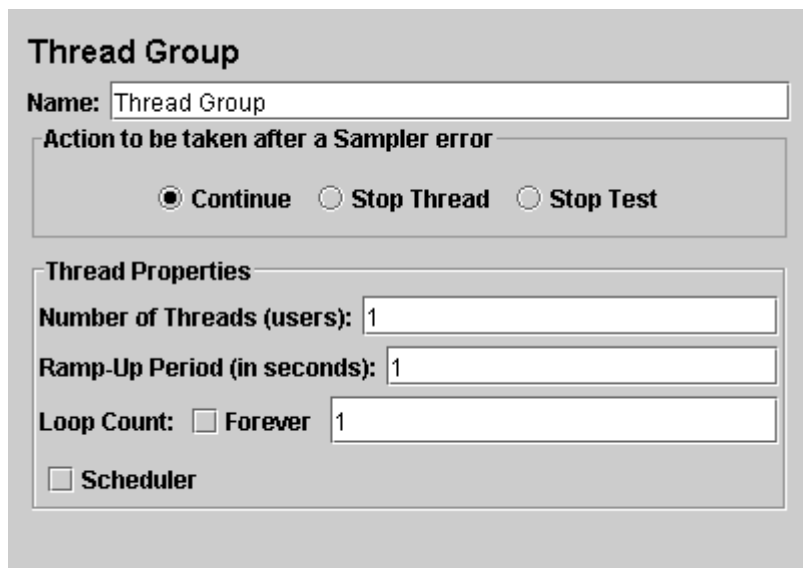
### 添加一个线程组

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）



The screenshot shows the 'Thread Group' configuration window in JMeter. It has a 'Name' field set to 'Thread Group'. Below it, a section titled 'Action to be taken after a Sampler error' contains three radio buttons: 'Continue' (selected), 'Stop Thread', and 'Stop Test'. Another section titled 'Thread Properties' contains three input fields: 'Number of Threads (users)' set to 1, 'Ramp-Up Period (in seconds)' set to 1, and 'Loop Count' set to 1 with the 'Forever' checkbox selected. At the bottom, there is an unchecked checkbox for '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 ActiveMQ 3.0 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
14. spring-1.1.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 线程）x（1 消息）x（重复 10 次）=60 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）。

见图 8.2 完整的 O'Reilly Users 线程组。

Unable to render embedded object: File (threadgroup2.png) not found.

## 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：

#改变 Name 域为“sample subscriber”

#如果 JMS 提供者使用 jndi.properties，选择这个复选框

#输入 InitialContextFactory 的类名

#输入提供者 URL，

#输入连接工厂名。请参考 JMS 提供者的文档信息

#输入消息主题名

#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

#“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.

#如果你需要读取响应，选择这个复选框

#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

#改变 Name 域为“sample publisher”

#如果 JMS 提供者使用 jndi.properties，选择这个复选框

#输入 InitialContextFactory 的类名

#输入提供者 URL，

#输入连接工厂名。请参考 JMS 提供者的文档信息

#输入消息主题名

#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

#“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.

#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.

#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**

When I run Optimizelt with the JMS sampler, I see several thousand instances of com.evermind.\_wk.class. During the test, the number of objects increases until the end of the test. If I click "garbage collect" button in Optimizelt, the instances are not cleaned up. Once the test plan is finished, these instances can be cleared. This explains why memory usage in JMeter increases steadily as the test runs.

So far, writing the sampler and testing it against Orion has been a great learning experience. Once it's all done, I need to report these findings to the Orion team, so they can fix these bugs.

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，你将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。监视器设计使用运行在 Tomcat 5 下

### 13.1

#### 13.2 HTTP 认证管理

#### 13.3 添加 HTTP 请求

添加 HTTP 请求到线程组元件（添加-->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-->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。在你在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-->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。

Unable to render embedded object: File (monitor\_health.png) not found.

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。

Unable to render embedded object: File (monitor\_screencap.png) not found.

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。

### 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



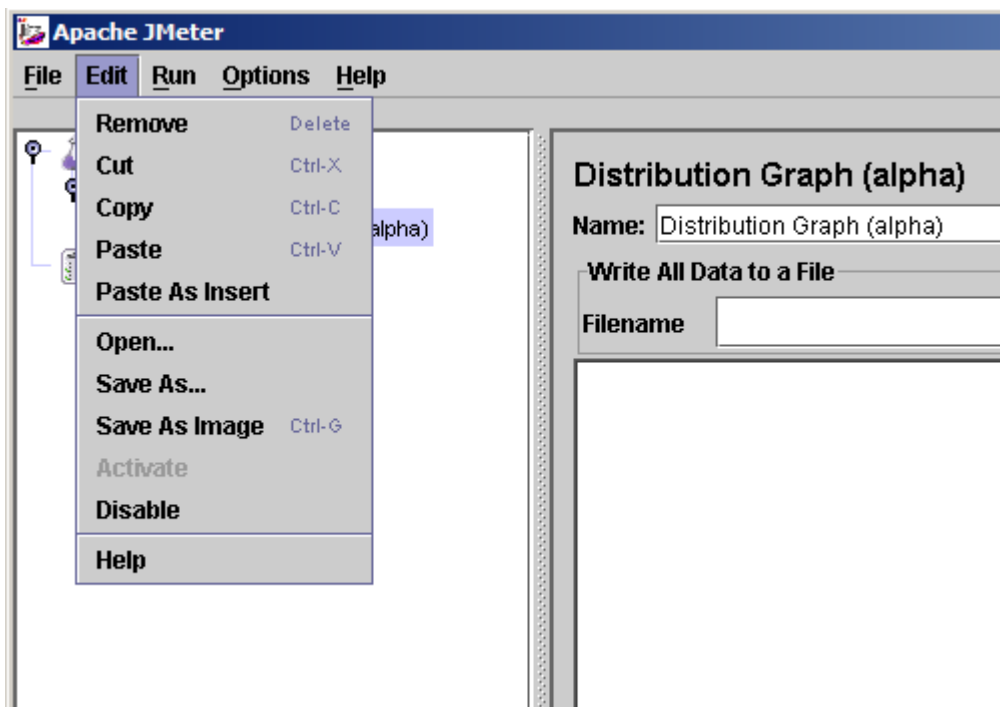


图 1-Edit -> Save As Image

## 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 -l 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。

## 14.4 CSV 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：

## 14.6 XML 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>
  </httpSample>
</testResults>

```

```

    <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.gif</responseData>
    <responseFile class="java.lang.String">Mixed1.html</responseFile>
    <cookies class="java.lang.String"></cookies>
    <method class="java.lang.String">GET</method>
    <queryString class="java.lang.String"></queryString>
    <url>http://www.apache.org/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”。

取样器属性意义如下：

属性	内容
t	用时（ms）
lt	延时（ms）-不是所有的取样器支持这个
ts	时间标志
s	是否成功
lb	标签
rc	响应代码
rm	响应消息
tn	线程名
dt	数据类型
de	数据编码
by	字节数
ng	在这个线程组中活跃的线程数
na	所有线程组中的活跃线程数

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。

#### 14.7 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。