

# 软件测试开发职位内推Q群：485353510

## 目录

目录.....	1
前言.....	2
一、脚本录制(Jmeter): .....	3
在“工作台”添加“HTTP 代理服务器” .....	3
端口: .....	3
分组: .....	4
记录 HTTP 信息头: .....	4
添加断言: .....	4
Regex matching: .....	4
在浏览器中录制.....	5
二、脚本录制(Badboy): .....	5
Badboy 使用: .....	5
三、参数化: .....	6
方法一: 使用“函数助手”添加从文件中读取字符串的函数。 .....	6
方法二: 使用“CSV Data Set Config”元件.....	8
四、使用关联参数: .....	8
五、添加检查点: .....	9
六、设置思考时间: .....	10
七、设置集合点: .....	10
八、使用 cookies: .....	10
九、模拟浏览器操作: .....	11
HTTP Cache 管理器.....	11
HTTP 请求的“从 HTML 文件获取所有内含的资源” .....	11
Embedded URLs must match.....	11
十、控制器实现脚本逻辑: .....	12
If 控制器: .....	12
随机控制器: .....	12
随机顺序控制器: .....	13
循环控制器: .....	13
十二、设置场景.....	14
设置测试计划: .....	14
添加线程组: .....	15
线程组参数分析: .....	15
线程运行状态显示: .....	16
十三、响应数据.....	17
图形结果: 添加“监视器”→“图形结果” .....	17
察看结果树: 添加“监视器”→“察看结果树” .....	18
聚合报告: 添加“监视器”→“聚合报告” .....	19
Summary Report: 添加“监视器”→“Summary Report” .....	19
将响应情况保存到文件中以供统计: .....	20

# 软件测试开发职位内推Q群：485353510

十四、服务器资源监控 (Linux) .....	20
Linux 服务器上加入监控脚本: .....	20
Jmeter 上使用监控: .....	21
将监控到的资源情况保存到文件中以供统计: .....	22
十五、统计分析.....	23
附录.....	26
后记.....	27

## 前言

一直以来都希望能有一套能够基本满足常规性能测试需求，并有效产生报表的工具，用以部分替代 LoadRunner 的依赖。所以专门针对 jmeter 进行了评估和研究，在评估过程中完成了一份使用说明；经过代码研究，对 jmeter 进行了改进，主要是增加了 linux 资源监控功能和报表功能。由于时间仓促，对增加的代码只进行了单元测试。

本手册可用于面向 B/S WEB 应用测试的工程师使熟悉 jmeter 使用，章节安排按照脚本设计、场景设置、查看监控三部分顺序组织。十四、十五两章内容是关于增进的监控和报表功能的，不适用于 apache 网站提供的原 jmeter。

讲解内容主要是使用上的，不涉及性能测试分析的内容。

# 软件测试开发职位内推Q群：485353510

## 一、脚本录制(Jmeter):

Jmeter 脚本（.jmx）为 xml 格式，树形结构，由元件组成，使用“取样器”产生请求。

在“工作台”添加“HTTP 代理服务器”

HTTP代理服务器

名称：

HTTP代理服务器

注释：

端口：

8080

☐ Attempt HTTPS Spoofing

Optional URL match string:

Test plan content

目标控制器：

使用录制控制器

分组：

不对样本分组

☒ 记录HTTP信息头 ☐ 添加断言 ☐ Regex matching

HTTP Sampler settings

Type: HTTP请求

☐ 自动重定向 ☒ 跟随重定向 ☒ Use KeepAlive ☐ 从HTML文件获取所有内含的资源

Content-type filter

Include:

Exclude:

包含模式

包含模式

添加

删除

排除模式

排除模式

添加

删除

启动

停止

重启

端口：

代理服务器的端口，默认 8080，可自行修改，但不要与其它应用端口冲突

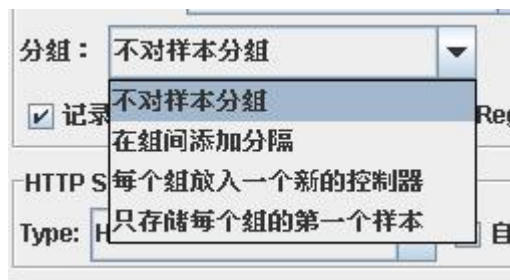
目标控制器：录制的脚本存放的位置，可选择项为测试计划中的线程组

# 软件测试开发职位内推Q群：485353510



分组：

对请求进行分组。“分组”的概念是将一批请求汇总分组，可以把 url 请求理解为组。



“不对样本分组”：所有请求全部罗列

“在组间添加分隔”：加入一个虚拟的以分割线命名的动作，运行同“不对样本分组”，无实际意义

“每个组放入一个新的控制器”：执行时按控制器给输出结果

“只存储每个组的第一个样本”：对于一次 url 请求，实际很多次 http 请求的情况，这个选项很好用，因为我们常常是不关心后面的那些请求的。

**记录 HTTP 信息头：**

录制 request 的 head 信息

**添加断言：**

录制时加入空的检查点

**Regex matching:**

录制时加入空的正则匹配

# 软件测试开发职位内推Q群：485353510

## 在浏览器中录制

启动 HTTP 代理服务器后，打开浏览器（IE，Firefox，Opera 等），添加代理，地址填写本机 ip 或 host name，端口填写刚刚设置的代理端口，在浏览器中进行正常网页浏览，即可录制下对应的操作。

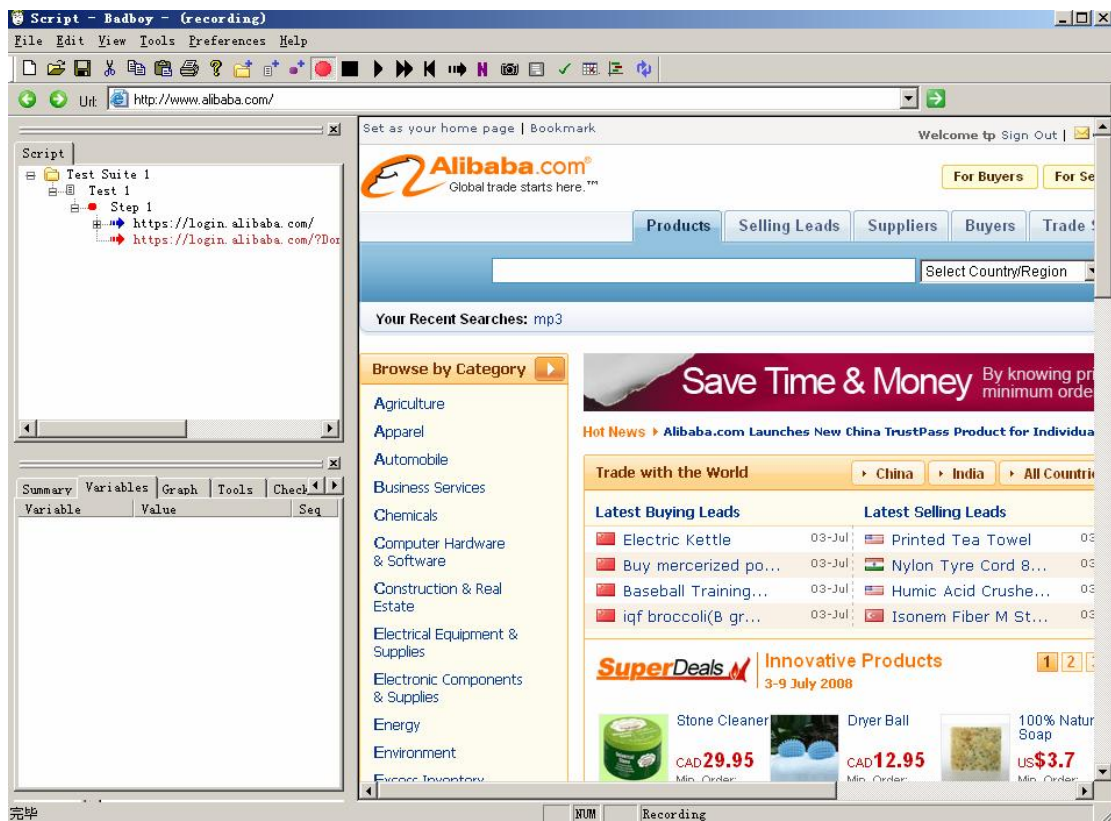
## 二、脚本录制 (Badboy)：

使用 jmeter 的代理服务器可以顺利录制完整的 http 请求，包括请求的头；但无法录制 cookies，对于网站脚本，大多需要 cookies 支持，可以使用 badboy 来录制。Badboy 是澳大利亚的一个软件公司设计的，有免费版可以下载。当然不通过录制也可以自己添加 cookies

### Badboy 使用：

下载安装包，一键安装，“开始”->“程序”->“badboy”

# 软件测试开发职位内推Q群：485353510



点击红色的 record 按钮开始录制，输入 url，与在浏览器中操作一样。左边的脚本栏会显示录制下来的请求。

录制完成后，“File” -> “Export to Jmeter”，保存成 jmeter 的脚本 jmx 文件；

用 jmeter 打开脚本，可以看到比较完整的请求。

## 三、参数化:

方法一：使用“函数助手”添加从文件中读取字符串的函数。

通过菜单“选项”→“函数助手对话框”调出“函数助手”。选择“\_\_StringFromFile”

软件测试开发职位内推Q群：485353510

函数助手

选择一个功能

\_\_StringFromFile

帮助

函数参数

名称：	值
输入文件的全路径	E:\project\param.dat
函数名称。用于存储在测试计划中其他的方式使用的...	keywords
Start file sequence number	
Final file sequence number	

添加

删除

拷贝并粘贴函数字符串

\$\_StringFromFile(E:\project\param.dat,keywords,,)

生成

填写文件路径；可以填入命名以便在其它地方使用该函数。点击生成。“函数助手”保留最近一次生成的函数信息。拷贝字符串，粘贴到需要调用该函数的地方即可，如：

同请求一起发送参数：	
名称：	值
q	\$_StringFromFile(E:\project\param.dat,keywords,,)

在其它地方调用时使用函数填写的命名\${命名}，如\${keywords}：

HTTP请求

协议：

方法：

GET

Content en

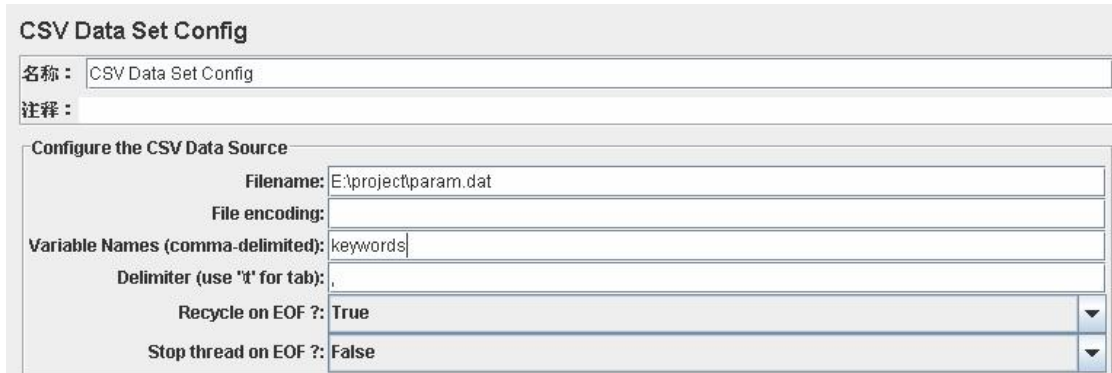
路径：

/bin/search?sell\_offer?q=\${keywords}

# 软件测试开发职位内推Q群：485353510

## 方法二：使用“CSV Data Set Config”元件

在脚本中添加“配置元件” → “CSV Data Set Config”



**CSV Data Set Config**

名称： CSV Data Set Config

注释：

**Configure the CSV Data Source**

Filename: E:\project\param.dat

File encoding:

Variable Names (comma-delimited): keywords

Delimiter (use '\t' for tab): .

Recycle on EOF?: True

Stop thread on EOF?: False

Filename 与 Variable Names 的定义与“\_\_StringFromFile”函数一样，填写后即可使用。

调用时使用 Variable Names 的命名\${填写的 Variable Names}，如\${keywords}：



**HTTP请求**

协议： 方法： GET Content encoding:

路径： /bin/search?sell\_offer?q=\${keywords}

## 四、使用关联参数：

在 http 请求下加入“后置处理器”->“正则表达式提取器”：

引用名称即使用的参数名；填入正则表达式；模板选取匹配的组；匹配数字为匹配的个数，负数表示全部匹配；缺省值为没有匹配到时的取值。



正则表达式提取器

名称：正则表达式提取器

注释：

要检查的响应字段

☒ 主体

☐ 信息头

☐ URL

☐ 响应代码

☐ 响应信息

引用名称：

product\_id

正则表达式：

name="chkProductIds" id="chk(.+?)"

模板：

\$1\$

匹配数字（0代表随机）：

-1

缺省值：

示例中用正则表达式匹配出产品 id 作为后续使用的参数。

提取到的参数，调用时用\${product\_id\_1}，\${product\_id\_2}，\${product\_id\_3}……；

如果想要得到匹配出的参数的个数，用\${product\_id\_matchNr}；如果想随机选取其中一个，只需将匹配数字设为 0，使用\${product\_id}调用即可。

可以一次匹配多组；示例中只匹配了一个，假如正则表达式为 name="chkProductIds" id="chk(.+?)" value="(.+?)"，就会有两组参数。想获得匹配到的组个数用\${product\_id\_g}。模板针对的是匹配到的字符串再做组的区分，比如希望\${product\_id}取出的是第二组参数的值，用\$2\$。

五、添加检查点：

在脚本中添加“断言”→“响应断言”

响应断言

名称：响应断言

注释：

要测试的响应字段

☒ 响应文本

☐ URL 样本

☐ 响应代码

☐ 响应信息

☐ Response Headers

☐ Ignore Status

模式匹配规则

☒ 包括

☐ 匹配

☐ Equals

☐ 否

要测试的模式

要测试的模式

<product id=(\*) value=1>

使用正则表达式进行检查，可以选择正则的模式匹配规则，以及检查的文字段。

## 六、设置思考时间：

在脚本中添加“定时器”→“固定定时器”

固定定时器	
名称：	固定定时器
注释：	
线程延迟（毫秒）：	300

设定延迟时间。

## 七、设置集合点：

在脚本中添加“定时器”→“Synchronizing Timer”

Synchronizing Timer	
名称：	Synchronizing Timer
注释：	
Grouping	
Number of Simulated Users to Group by:	12

设置集合点处的并发用户数，即“多少”个用户达到集合点后再执行。

## 八、使用 cookies：

访问的页面需要 cookies 时，在脚本中加入“配置元件”->“HTTP Cookies 管理器”，

要点：cookies 管理器元件需要位于需要使用 cookies 的请求的上一级节点。



在示例中加入了一个事务控制器，在该任务控制器下的节点都使用同一个 cookies 管理器

## 九、模拟浏览器操作：

### HTTP Cache 管理器

一般来说不建议使用，因为在 LR 中我们一般也不让页面 cache，避免由于 cache 导致压力不足，结果过于乐观。

### HTTP 请求的“从 HTML 文件获取所有内含的资源”

如果勾选了该项，则对 html 文件所调用的外部链接也会产生压力，响应的服务必须是在待测系统范围内，即该服务也是受压的系统之一。一般来说当不具备该服务环境时不勾选。

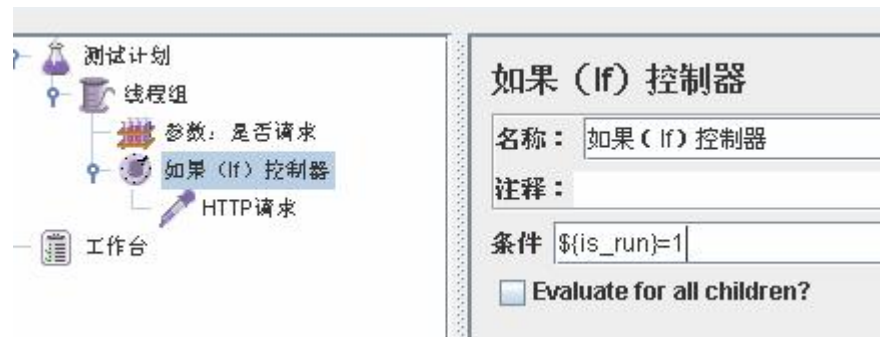
### Embedded URLs must match

这个用于匹配 html 中的资源链接，正则表达式，只有匹配符合的才会去请求比如正则表达式 [http://www.alibaba.com/.\\*](http://www.alibaba.com/)，意味着只有 [http://www.alibaba.com](http://www.alibaba.com/) 相关的链接才会被请求。

## 十、控制器实现脚本逻辑：

### If 控制器：

条件符合时执行控制器的子节点内容

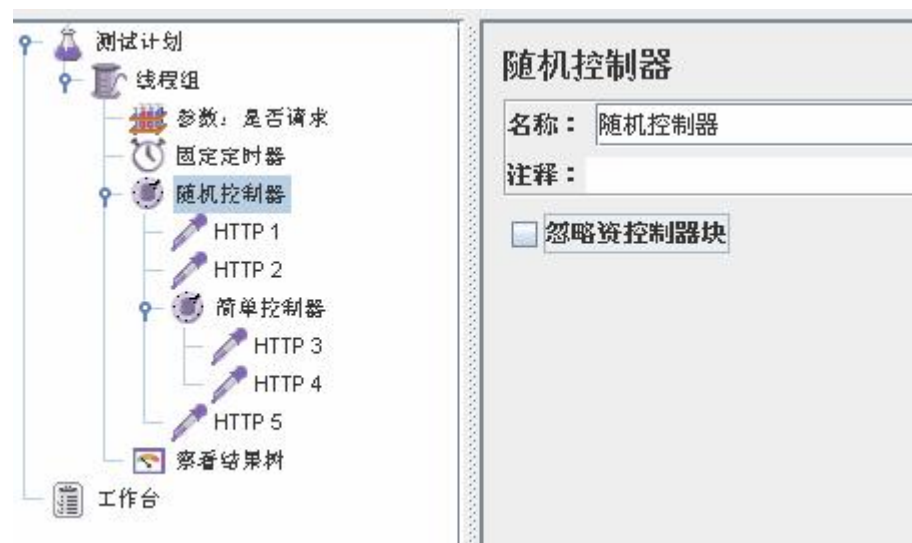


示例：判断`${is_run}=1` 是否成立，如果成立就执行 http 请求；复选框 “Evaluate for all children?” 表示是否对所有子节点使用判断条件，不选择的话，只对第一个子节点生效。

无 else 判断，可以用非条件来实现。

### 随机控制器：

随机执行某一个子节点内容



示例中随机控制器下有四个子节点：HTTP 1、HTTP 2、简单控制器、HTTP 3，简单控制器下有两个子节点：HTTP 3、HTTP 4；

# 软件测试开发职位内推Q群：485353510

当复选框“忽略控制器块”未勾选时，简单控制器是作为一个节点跟其它随机控制器子节点一起参与随机执行；勾选时，简单控制器下的子节点直接参与其它随机控制器子节点一起参与随机执行。

## 随机顺序控制器：

子节点全部执行，但顺序随机。

## 循环控制器：

设置执行控制器子节点的次数，也可以设置永远执行。

十二、设置场景

设置测试计划：

测试计划

名称：

测试计划

注释：

用户定义的变量

名称：	值
-----	---

添加

删除

☐ 独立运行每个线程组（例如在一个组运行结束后启动下一个）

☐ 函数测试模式

只有当你需要记录每个请求从服务器取得的数据到文件时才需要选择函数测试模式。

选择这个选项很影响性能。

Add directory or jar to classpath

浏览...

删除

清除

Library

测试计划就是一个完整的场景

“独立运行每个线程组”：勾选以后所有的线程组都是顺序执行的了。一般不勾选，让所有的线程组并发启动。

“函数测试模式”：勾选后会有详细的请求记录，消耗资源，影响客户端性能。一般不勾选。

用户定义的变量：全局变量

添加线程组：

线程组

名称：

线程组1

注释：

在取样器错误后要执行的动作

☒ 继续

☐ 停止线程

☐ 停止测试

线程属性

线程数：

1

Ramp-Up Period (in seconds):

1

循环次数 ☐ 永远 

1

☒ 调度器

调度器配置

启动时间

2008/06/24 09:33:39

结束时间

2008/06/24 09:33:39

持续时间（秒）

启动延迟（秒）

取样器错误后要执行的动作：继续，停止线程，停止测试

线程数：可理解为当前线程组下脚本运行的“并发用户数”。

Ramp-Up Period (in seconds)：开始运行时线程数在“设定的时间”内由 0 增加到设置值。

循环次数：当前线程组下脚本运行循环次数；“永远”选项，无限次循环

启动时间：脚本自动启动时间

结束时间：脚本自动结束时间

持续时间（秒）：持续运行的时段

启动延迟（秒）：延迟指定时间后启动

线程组参数分析：

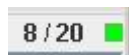
1. 取样器错误后执行的动作：“停止线程”为停止当前的线程运行；“停止测试”为停止测试计划运行。
2. Ramp-Up Period 表示从 0 增加到指定线程数的时间，是线性增加，如：线程数为 20 个，Ramp-Up Period 为 50s， $50/20=2.5\text{s}/\text{个}$ ，所以是每隔 2.5s 增加一个线程。
3. 循环次数：勾选“永远”，启动后必须手工“停止”才会停止；次数达到时若调度器未运行完毕，则调度器无效，停止执行；调度器中运行完毕，次数尚未达到，则次数设置

# 软件测试开发职位内推Q群：485353510

无效，停止执行。

4. 调度器：在手工启动后生效，设置生效原则：启动延迟的优先级高于启动时间，持续时间间优先级高于结束时间，设置针对“未来时间”有效，针对“过去时间”无效；即启动延迟与启动时间同时设置如果不一致，则以启动延迟为准，持续时间与结束时间同时设置如果不一致，则以持续时间为准，设置为过去的时间则不生效。

线程运行状态显示：

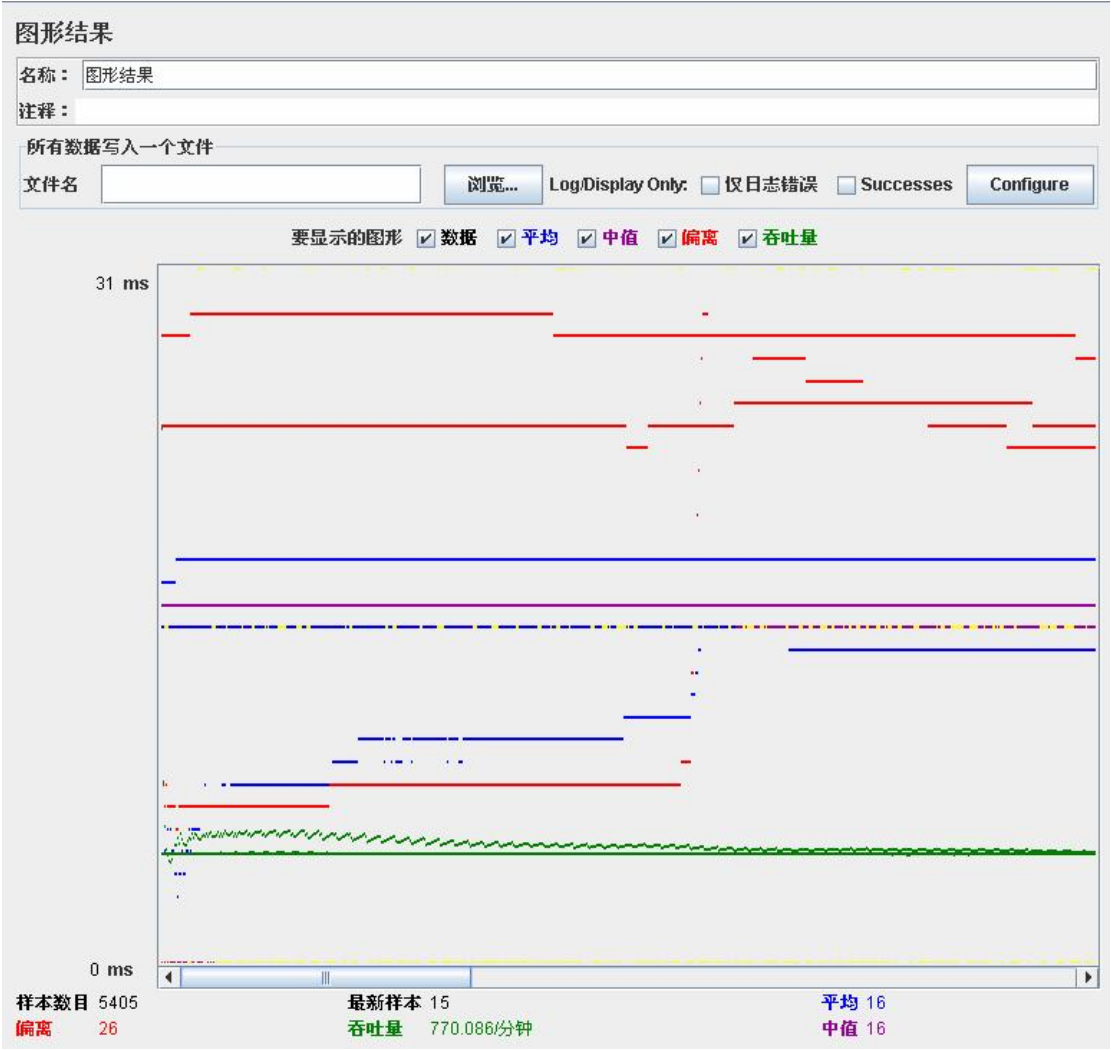


在 jmeter 右上角处。图示含义为：绿色表示正在运行；当前图例表示共 20 个线程，已启动 8 个。



十三、响应数据

图形结果：添加“监视器”→“图形结果”



显示内容含义：

样本数目：运行时得到的取样器响应结果个数

最新样本：最近一个取样器结果的响应时间

平均：所有取样器结果的响应时间平均值

偏离：所有取样器结果的响应时间标准差

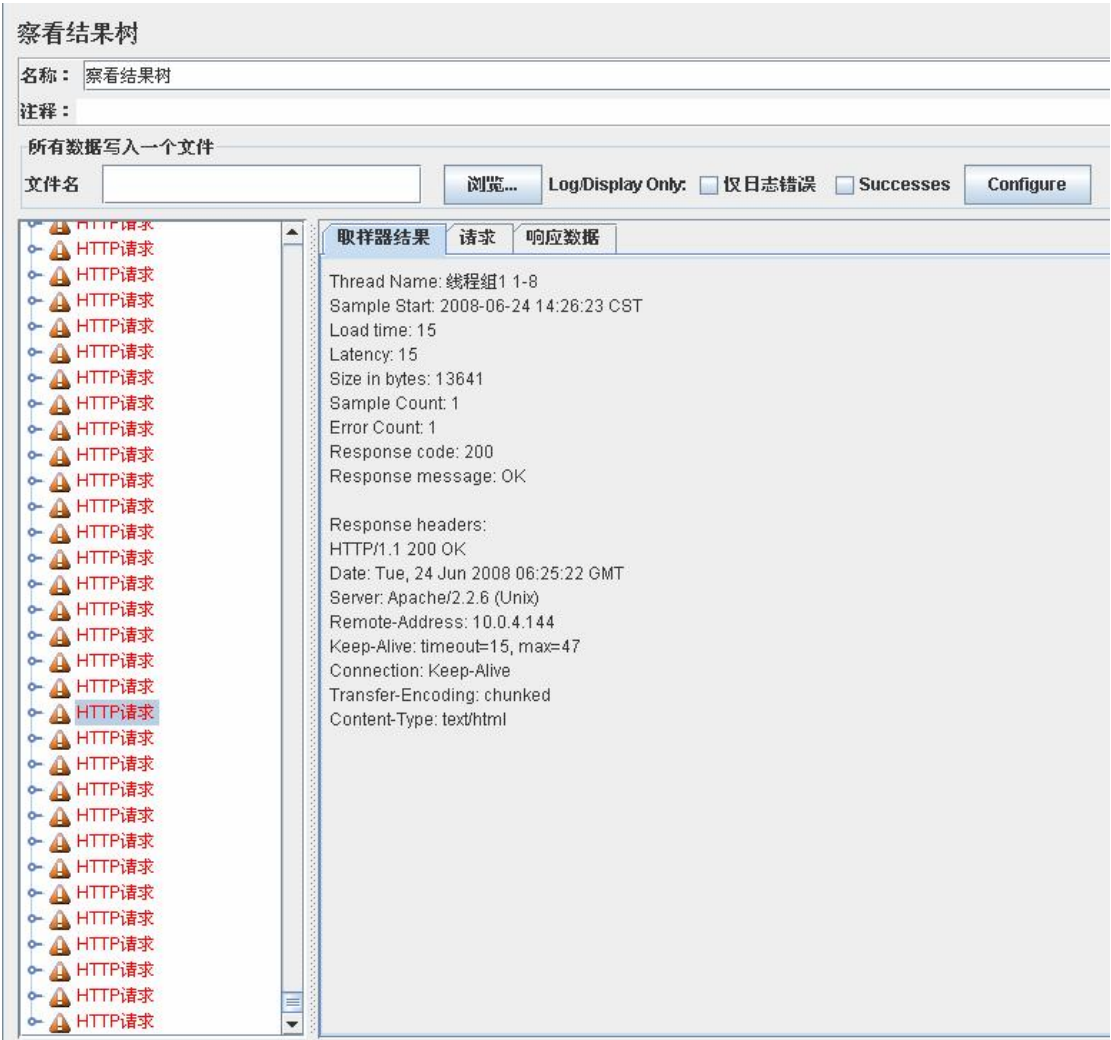
吞吐量：每分钟响应的取样器结果个数

中值：所有取样器结果的响应时间中间值

显示图线为随时间变化曲线，但 x 轴不是时间轴，是取样器个数的均匀分布轴

# 软件测试开发职位内推Q群：485353510

察看结果树：添加“监视器”→“察看结果树”



显示内容含义：

取样器结果：显示的是取样器相关参数（客户端参数与响应参数）

请求：http request

响应数据：http response data

# 软件测试开发职位内推Q群：485353510

聚合报告：添加“监视器”→“聚合报告”

聚合报告

名称：

聚合报告

注释：

所有数据写入一个文件

文件名

浏览...

Log/Display Only: ☐ 仅日志错误 ☐ Successes 

Configure

Label	# Samples	Average	Median	90% Line	Min	Max	Error %	Throughput	KB/sec
HTTP请求	5981	16	15	31	0	391	99.93%	9.6/sec	128.2
总体	5981	16	15	31	0	391	99.95%	9.6/sec	128.2

显示内容含义：

- Label: 取样器名称
- Samples: 运行时得到的取样器响应结果个数
- Average: 所有取样器结果的响应时间平均值
- Median: 所有取样器结果的响应时间中间值
- 90%Line: 所有取样器结果的响应时间 90%线
- Min: 所有取样器结果的响应时间最小值
- Max: 所有取样器结果的响应时间平均值
- Error%: 出错的取样器结果占有所有取样器结果的比例
- Throughput: 每秒钟响应的取样器结果个数
- KB/sec: 每分钟响应的数据流量

Summary Report: 添加“监视器”→“Summary Report”

Summary Report

名称：

Summary Report

注释：

所有数据写入一个文件

文件名

res/h

浏览...

Log/Display Only: ☐ 仅日志错误 ☐ Successes 

Configure

Label	# Samples	Average	Min	Max	Std. Dev.	Error %	Throughput	KB/sec	Avg. Bytes
h3-res.html	972	286	8	753	101.63	0.00%	4.8/sec	107.85	22855.0
总体	972	286	8	753	101.63	0.00%	4.8/sec	107.85	22855.0

显示内容含义：

- Label: 取样器名称
- Samples: 运行时得到的取样器响应结果个数

# 软件测试开发职位内推Q群：485353510

Min: 所有取样器结果的响应时间最小值

Max: 所有取样器结果的响应时间平均值

Std.Dev.: 所有取样器结果的响应时间标准差

Error%: 出错的取样器结果占有所有取样器结果的比例

Throughput: 每秒钟响应的取样器结果个数

KB/sec: 每分钟响应的数据流量

Avg.Bytes: 所有取样器返回 http response data 字节数的平均值

将响应情况保存到文件中以供统计:

给单个取样器添加监视器后, 在“所有数据写入一个文件”下方的输入框输入响应情况记录的保存文件, 或者浏览选择; 可使用全路径, 也可使用相对路径, 相对路径基准为脚本保存路径; 多个取样器使用一个监视器时, 得到的统计结果是累加起来的。

## \* 十四、服务器资源监控 (Linux)

Linux 服务器上加入监控脚本:

登录到被监控的服务器上, 将 status 脚本放上去。该脚本执行时会在所在路径生成 status.xml 文件, 可以直接将 status 脚本放在 web server 的目录下, 也可以用软链接来链到 status.xml 文件。下面介绍在基于 apache 的 web server 上的配置方法:

在 apache 的配置文件 httpd.conf 中找到 DocumentRoot, 一般默认是 apache 目录下的 htdocs, 将 status 脚本放到该目录下;

更改执行权限:

```
chmod 744 status
```

启动该脚本:

```
./status start
```

启动起来之后就会在当前目录下产生 status.xml 文件

不需要监控时, 停止该脚本:

```
./status stop
```

# 软件测试开发职位内推Q群：485353510

Jmeter 上使用监控:

使用

打开 jmeter，建立一个线程组，添加一个 http 请求，ip 就是要监控的服务器地址，端口号就是 apache 侦听的 http 端口，协议是“http”，路径是“/status.xml”，勾选“用作监视器”；

HTTP请求

名称： HTTP请求

注释：

Web服务器

服务器名称或IP： 10.0.4.144 端口号：

HTTP请求

协议： http 方法： GET Content encoding:

路径： /status.xml

☒ 自动重定向 ☐ 跟随重定向 ☒ Use KeepAlive ☐ Use multipart/form-data for HTTP POST

同请求一起发送参数：

名称：	值	编码？	包含等于？
-----	---	-----	-------

添加 删除

同请求一起发送文件：

文件名称：	参数名称：	MIME类型：
-------	-------	---------

添加 浏览... 删除

其他任务

☐ 从HTML文件获取所有内含的资源 ☒ 用作监视器 ☐ Save response as MD5 hash?

Embedded URLs must match:

再为该 http 请求添加一个“固定定时器”组件和一个“监视器结果”组件，“固定定时器”的延时要设置为大于 1 秒的时间，即数据的采样时间。

线程组

HTTP请求

固定定时器

监视器结果

工作台

名称： 固定定时器

注释：

线程延迟（毫秒）： 1500

在线程组中循环次数设置勾选“永远”；

# 软件测试开发职位内推Q群：485353510

**线程组**

名称： 线程组

注释：

在取样器错误后要执行的动作

☒ 继续 ☐ 停止线程 ☐ 停止测试

**线程属性**

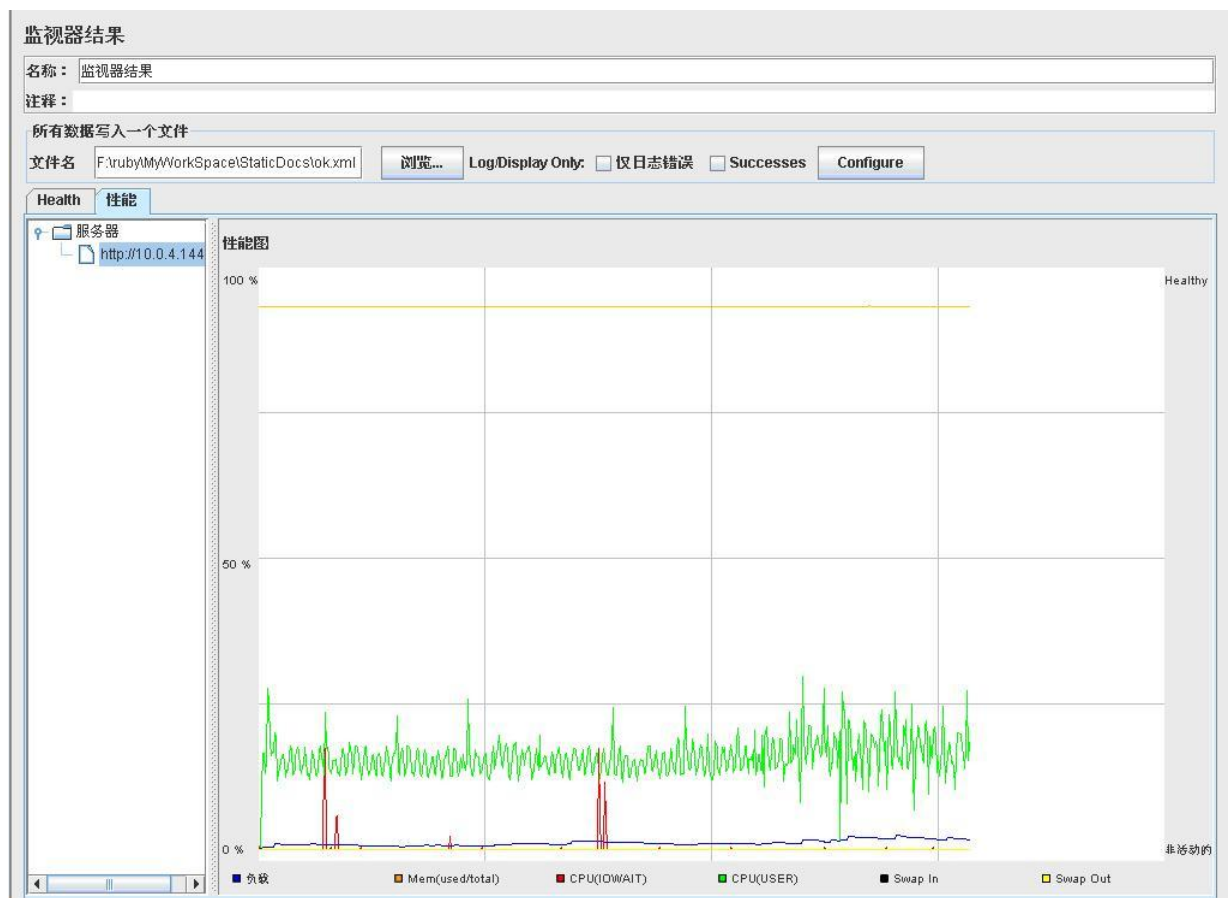
线程数： 1

Ramp-Up Period (in seconds): 1

循环次数 ☒ 永远

☐ 调度器

Run 一下，就可以在监视器结果上看到刚刚添加的监控服务器了，目前已经监控了 6 个参数：cpu%user,cpu%iowait,load,mem%(used/total),swap in,swap out。



将监控到的资源情况保存到文件中以供统计：

方法同“将响应情况保存到文件中以供统计：”

## \* 十五、统计分析

将记录响应情况和资源监控情况的文件生成 html 图表报告以便进行分析（注意：记录文件格式为 xml，若不完整是无法解析的；即脚本运行中生成的文件是无法解析的，必须在脚本运行完毕后才能进行以下处理）；生成方法如下：

设置 jmeter 保存的文件地址

打开 jmeter 目录的 bin 目录下的 Jmeter-DataChart.ini 文件，按照格式：

response=响应文件路径

resources=监控资源文件路径

进行填写。比如有 2 个响应文件 res/r1.jtl，res/r2.jtl，1 个监控资源文件 res/linux.jtl，则填写文件内容为：

```
response=res/r1.jtl
```

```
response=res/r2.jtl
```

```
resources=res/linux.jtl
```

保存退出。

双击运行 bin 目录下的 Jmeter-DataChart.bat，进行文件的解析和图表生成。解析处理成功后，在 bin 目录下会生成一个名为 CurDataChart-Index.html，双击打开，可以看到相应的图表链接，是原文件名+ -res.html 的文件





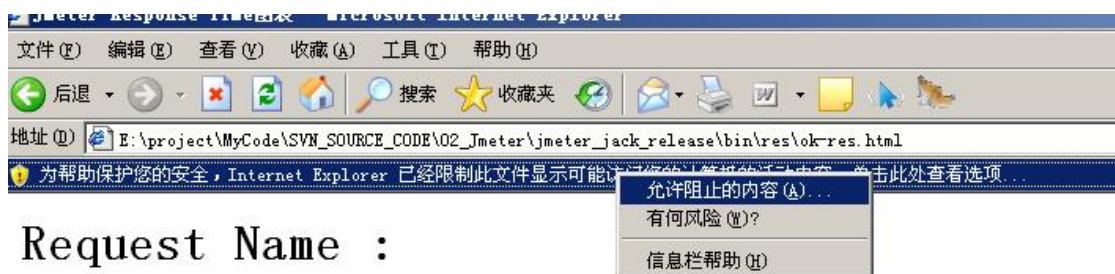
Jmeter Result

response:[res/h-res.html](#)

response:[res/ok-res.html](#)

resources:[res/linux-res.html](#)

点击链接打开报告，IE 提示脚本风险



Request Name :

点击允许，即可看到完整的图表：

响应图表：

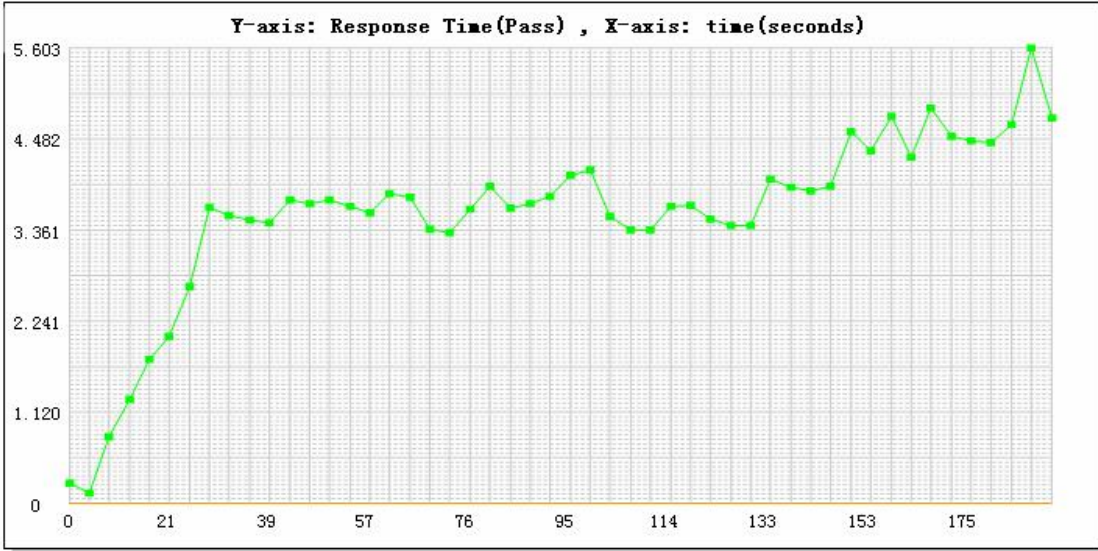


Request Name : ok.xml

Response Time Data Table

Response Status	Maximum	Minimum	Median	90%Line	Average	Standard Deviation
PASS	5.603	0.062	3.689	4.607	3.57	1.035
FAIL	0	0	0	0	0	0

Response Time Data Chart

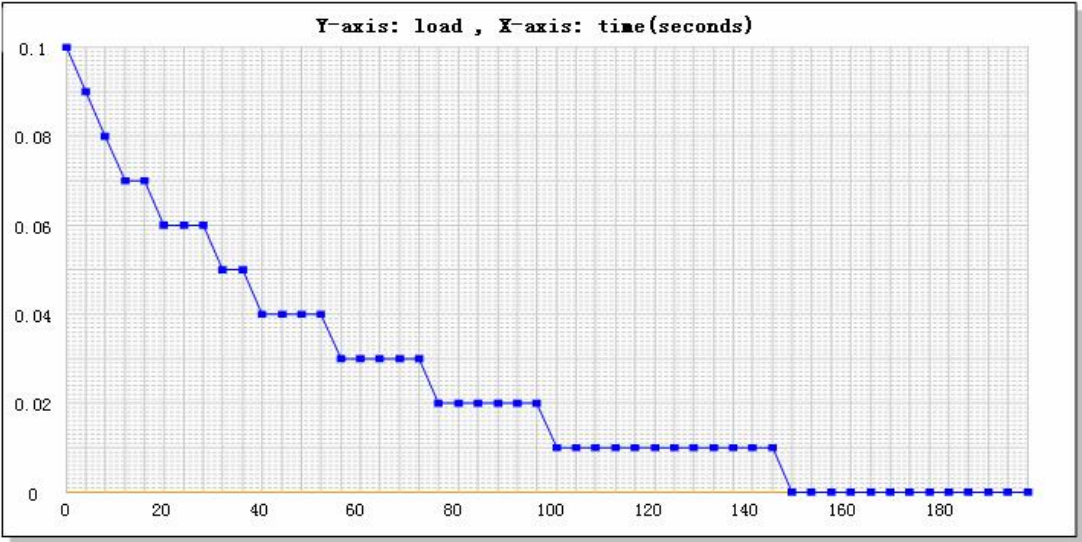


资源监控图表：

10.0.4.144 Linux Resources Data Table

Resource Name	Maximum	Minimum	Median	90%Line	Average	Standard Deviation
load	0.1	0	0.01	0.06	0.02	0.02
cpu%user	7.98	0	0.25	0.5	0.54	1.21
mem%	91.45	91.44	91.45	91.45	91.44	1.14
cpu%iowait	0	0	0	0	0	0
swap in	0	0	0	0	0	0
swap out	0	0	0	0	0	0

10.0.4.144 Linux Resources Data Chart



注意：因为该图表采用 vml 编写，在不支持 vml 的浏览器上无法显示。

附录

监控器实现的是 linux 系统上的资源监控，要求：服务器上至少有一个 web server（因需要通过 http 请求获取资源 xml），用于生成服务器资源 xml 的脚本中使用了 sysstat 包中的指令，所以需要安装 sysstat 包。

如果被监控的 linux 上没有安装 sysstat 包，是无法正常生成监控数据的，这里附上 sysstat 包的安装方法：

# 软件测试开发职位内推Q群：485353510

首先到 <http://perso.wanadoo.fr/sebastien.godard/> 下载最新的版本，最好是源码包，

比如 sysstat-5.1.1.tar.gz

1.解包：

```
tar zxvf sysstat-5.1.1.tar.gz
```

2.安装：

```
cd sysstat-5.1.1
```

```
make config
```

这步可以省略，有些发行版中会出错；如果不用这个命令，可以直接安装到其默认的/usr/local/lib 目录中

```
make
```

```
编译
```

```
make install
```

```
安装
```

## 后记

从开始 jmeter 评估到写完使用手册，历时约三周，中间由于很多其它事情加入，真正利用的时间不到一周。为了能对 jmeter 的功能进行改良和加强，花了大约 4 天时间来阅读代码，之后用 2 天时间理清监控器并开始修改代码，耗时 3 天完成了 linux 上的监控器；又经历了一些小的修改和调整，基本成型。数据报表利用 vml 生成线图，用 js 求取各种如极值、中值、标准差等数学运算值，用去大约四天，将图表样板与生成脚本全部完成。

一到十三的内容基本都是在评估时写成的，可以适用与 apache 提供的 jmeter；而十四、十五这部分内容是经过自行开发后得到的，仅适用于自行维护开发的 jmeter 版本。