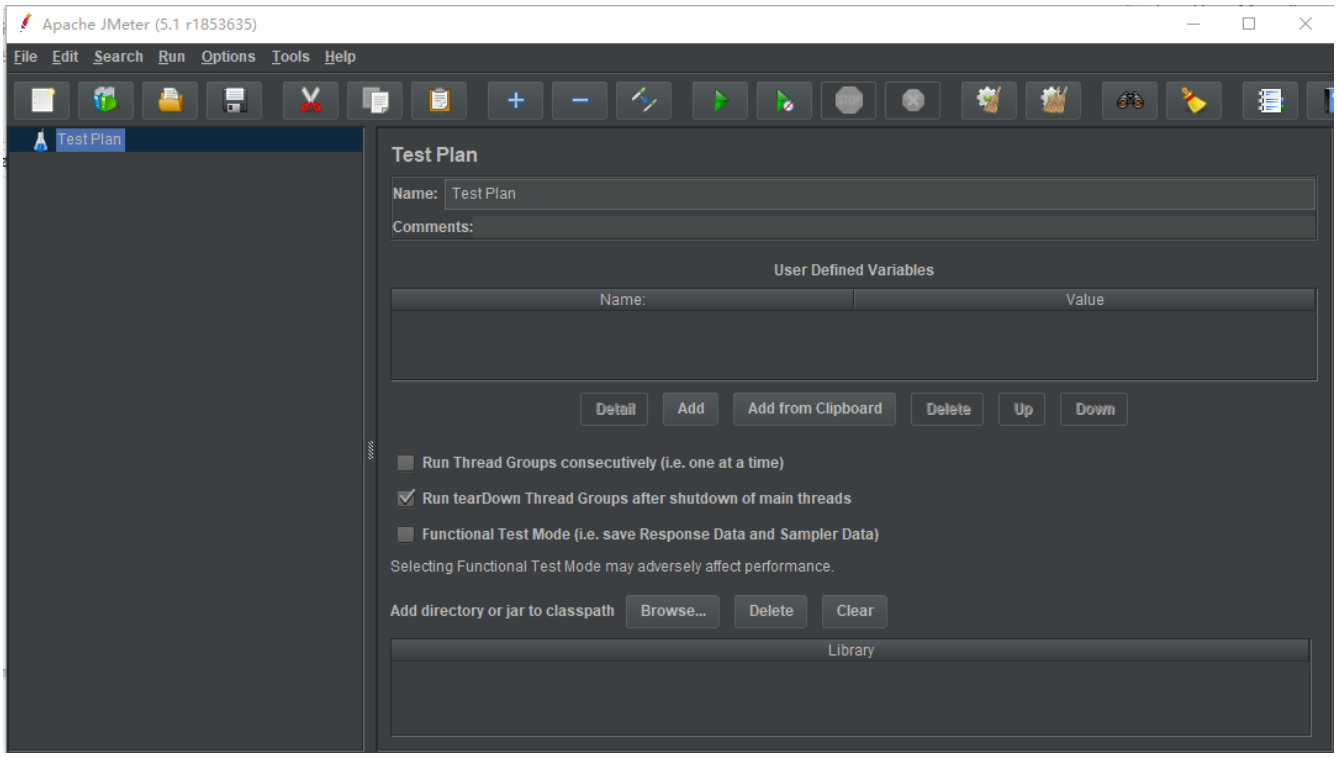
jmeter简易教程

jmeter是我们在测试写的代码性能的时候经常用到的一个小工具

1、下载与安装

- 下载地址在Apache官网：http://jmeter.apache.org/
- 解压，进入bin下运行jmeter.bat即可进入jmeter界面



2、教程出处

https://www.cnblogs.com/imyalost/p/7062784.html

3、目录

3、元件的作用域及执行顺序

jmeter各元件的作用域及执行的顺序;

4、Sampler之SOAP/XML-RPC Request

取样器中关于SOAP/XML-RPC Request的用法;

5、Sampler之HTTP请求

取样器中关于HTTP请求的用法;

6、http请求之content-type

取样器中关于HTTP请求的补充说明;

7、Sample之JDBC Request

取样器中关于JDBC请求的用法;

8、JDBC Request之Query Type

取样器中关于JDBC请求的补充说明;

9、目录结构

jmeter目录结构等简单介绍;

10、参数化

jmeter参数化的4种方式;

11、关联之正则表达式提取器

jmeter关联之正则表达式提取器的用法;

12、关联之XPath Extractor

jmeter关联之XPath Extractor的用法;

13、配置元件之计数器

jmeter配置元件中关于计数器的用法;

14、配置元件之HTTP属性管理器

jmeter配置元件中关于http属性管理器的用法;

15、函数助手

jmeter内置函数助手的简单介绍;

16、定时器

jmeter八大元件之定时器的介绍；

17、断言

jemter八大元件之断言的介绍；

18、逻辑控制器

jmeter八大元件之逻辑控制器的介绍；

19、常见问题及解决方法

jmeter使用过程中常见问题及解决方案的说明；

20、阶梯式加压测试

jmeter扩展插件Stepping Thread Group的简单介绍；

21、jmeter常用插件介绍

jmeter插件Transactions per Second、Response Times Over Time、PerfMon Metrics Collector的下载安装及使用；

22、内存溢出原因及解决方法

关于jmeter做压力负载测试时候遇到内存溢出的原因和解决方法；

23、jmeter分布式测试

关于高并发情况下分布式测试的一些技术点和注意事项；

24、dubbo接口测试

利用jmeter的dubbo插件进行dubbo接口测试和性能测试；

25、linux环境运行jmeter并生成报告

linux环境，非GUI模式运行jmeter脚本进行性能测试，并生成测试报告的介绍；

26、jmeter生成HTML格式性能测试报告

jmeter生成HTML格式的性能测试报告的2种方式，以及可视化图表解析内容；

4、详细内容

（1）基础介绍

参考书籍：段念《软件性能测试与案例剖析》——第二版

推荐一本书《零成本实现web性能测试——基于Apache—jmeter》，主要内容是一些关于jmeter的实战使用，想学习的可以去看看。。。

jmeter是一款优秀的开源性能测试工具，目前最新版本3.0版本，官网文档地址：http://jmeter.apache.org/usermanual/index.html

一、优点

1、开源工具，可扩展性非常好

2、高可扩展性，用户可自定义调试相关模块代码

3、精心简单的GUI设计，小巧灵活

4、完全的可移植性和100%纯java

5、完全swing和轻量组件支持（预编译的HAR使用javax.swing.*）包

6、完全多线程框架，允许通过多个线程并发取样以及单独的线程对不同的功能同时取样

7、支持脚本取样器

二、安装及下载

这里附一个最新的jmeter官网下载地址：

http://jmeter.apache.org/download_jmeter.cgi

该链接是3.0版本的jmeter安装包

jmeter本身不需要安装，只需要配置好JDK环境，然后在在jmeter文件中的bin文件中打开jmeter.bat文件即可

最新版本，建议配置的JDK最好用1.7及以上版本

三、基础构成

1、组成部分

1）负载发生器：产生负载，多进程或多线程模拟用户行为

2）用户运行器：脚本运行引擎，用户运行器附加在进程或线程上，根据脚本模拟指定的用户行为

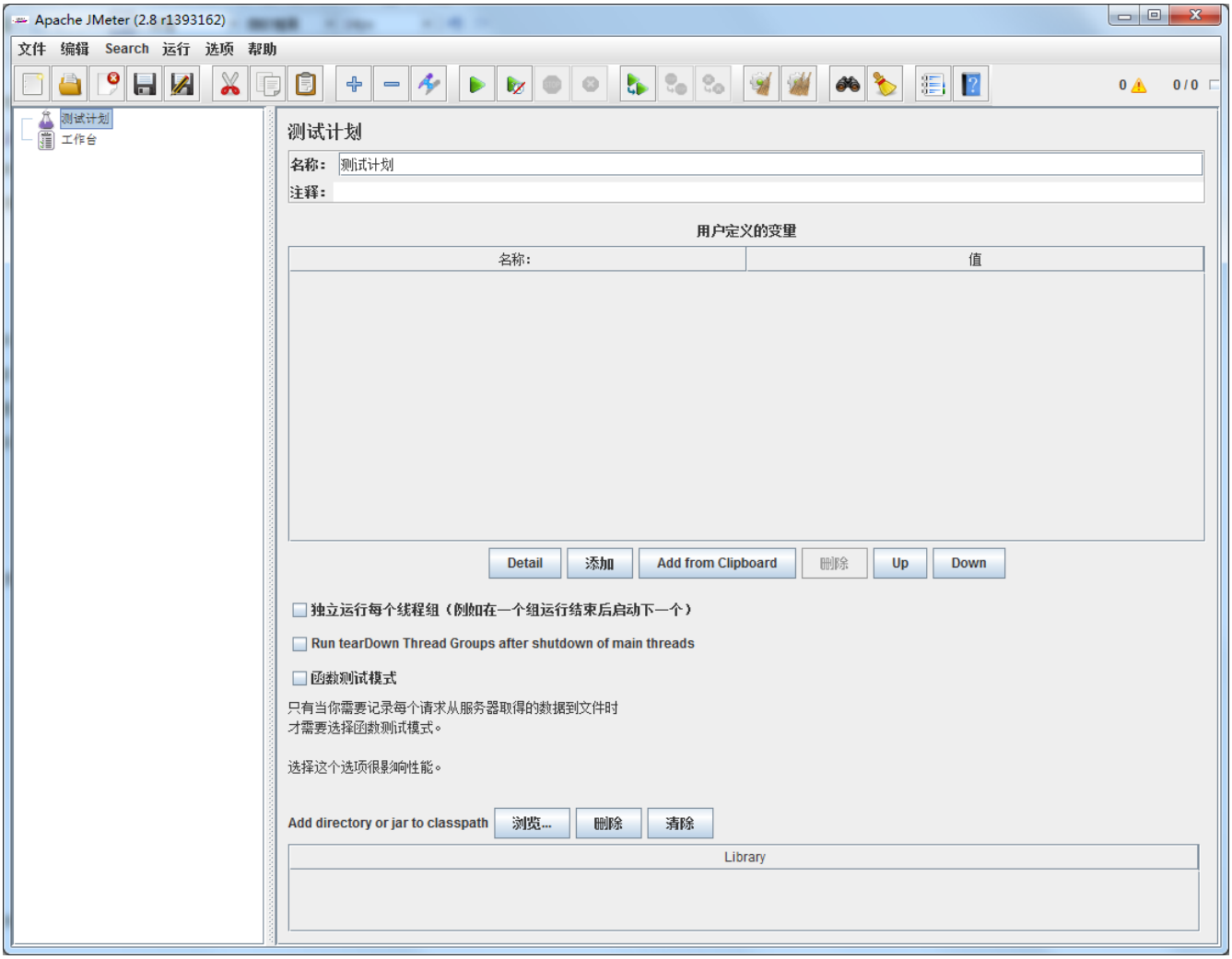
3）资源生成器：生成测试过程中服务器、负载机的资源数据

4）报表生成器：根据测试中获得的数据生成报表，提供可视化的数据显示方式·

2、主要概念

2.1测试计划（test plan）

描述一个性能测试，包含本次测试所有相关功能



2.2.threads（users）线程

Setup thread group：

一种特殊类型的线程，可用于执行预测试操作。即执行测试前进行定期线程组的执行

Teardown thread group：

一种特殊类型的线程，可用于执行测试后动作。即执行测试结束后执行定期的线程组

以上两个线程组，举个例子：loadrunner的脚本除了action里是真正的脚本核心内容，还有初始化“环境”的初始化脚本和测试完毕后对应的清除信息的脚本块，与其对应

Thread group：

通常添加使用的线程，一般一个线程组可看做一个虚拟用户组，其中每个线程为一个虚拟用户

2.3测试片段（test fragment）

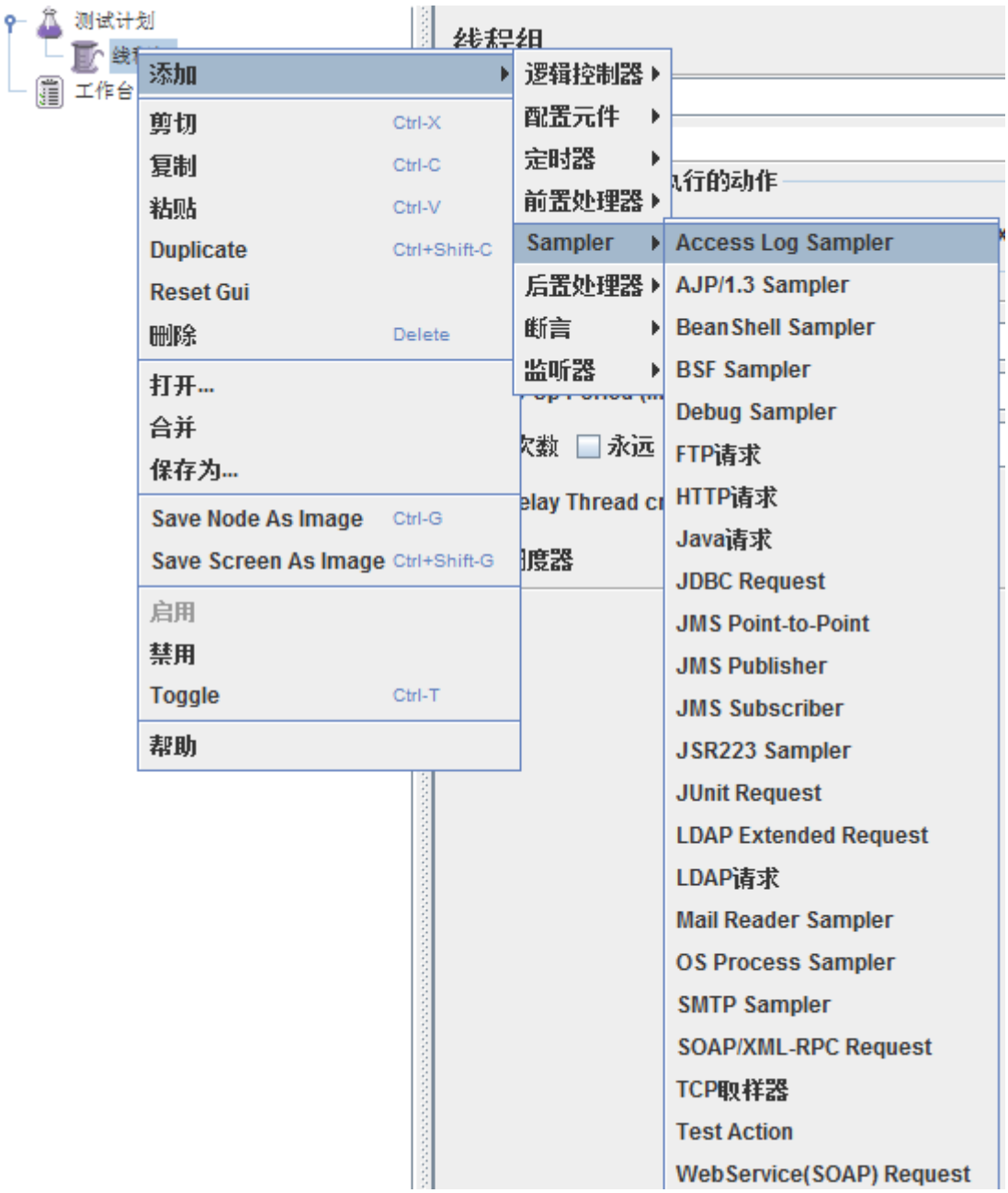
2.5版本之后新增的一个选项，是一种特殊的线程组，在测试树上与线程组一个层级，但是它不被执行，除非它是一个模块控制器或者被控制器所引用时才会被执行

2.4控制器

Jmeter有2种控制器：取样器（sampler）和逻辑控制器（Logic Controller）

作用：用这些原件驱动处理一个测试

1）取样器（Sampler）



是性能测试中向服务器发送请求，记录响应信息，记录响应时间的最小单元，JMeter 原生支持多种不同的sampler

如 HTTP Request Sampler 、 FTP Request Sampler 、TCP Request Sampler 、JDBC Request Sampler 等

每一种不同类型的 sampler 可以根据设置的参数向服务器发出不同类型的请求。

Java Request Sampler 和 Beanshell Request Sampler 是两种特殊的可定制的 Sampler （暂不讨论）

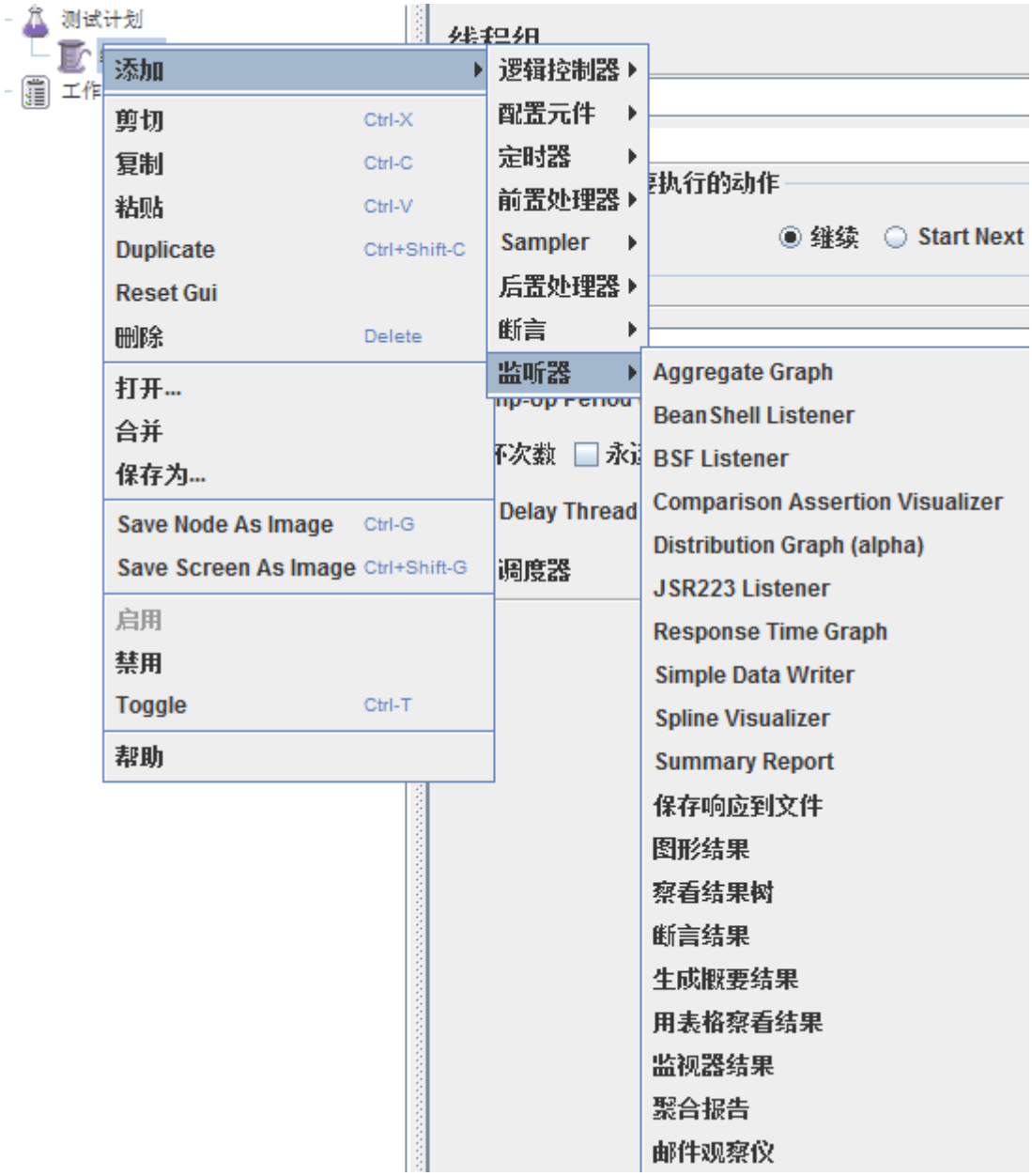
2）逻辑控制器（Logic Controller）

包含两类原件：

一类是控制Test Plan中Sampler节点发送请求的逻辑顺序控制器，常用的有：If Controller、Swith Controller、Loop Controller、Random Controller等

另一类是用来组织和控制Sampler节点的，如Transaction Controller、Throughput Controller等

2.5监听器（Listener）



对测试结果进行处理和可视化展示的一系列组件，常用的有图形结果、查看结果树、聚合报告等

以上的五类原件就可以构成一个简单的性能测试脚本

下面再介绍几种jmeter提供的其他组件：

2.6配置原件（Config Element）

用于提供对静态数据配置的支持。CSV Date Set Config可以将本地数据文件形成数据池（Date Pool），而对应于HTTP Request Configuration

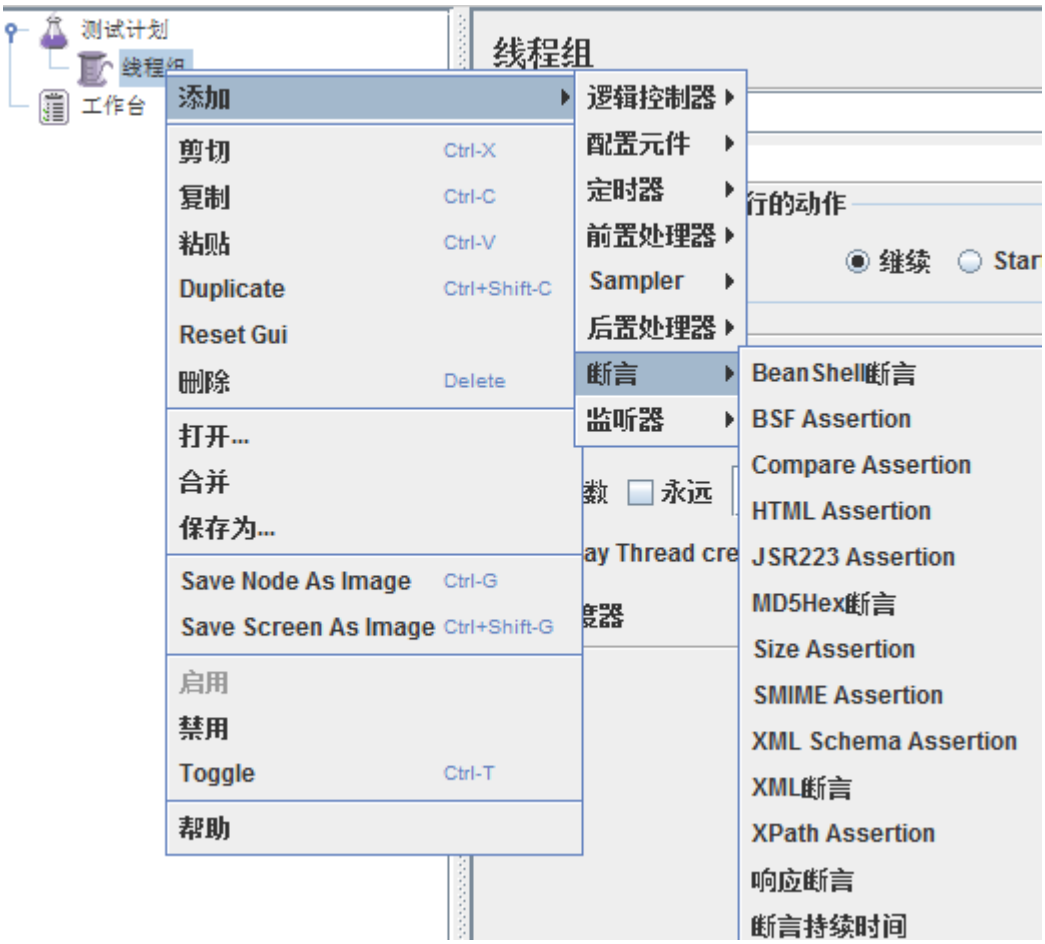
和TCP Request Sample等类型的Configuration元件则可以修改这些Sample的默认数据等

2.7定时器（Time）

用于操作之间设置等待时间，等待时间使性能测试中常用的控制客户端QPS的手段，jmeter定义了Constant Times、

Constant Throughput Times、Guass Ramdon Times等不同类型的Times

2.8断言（Assertions）



用于检查测试中得到的响应数据等是否符合预期，Assertions一般用来设置检查点，用以保证性能测试过程中的数据交互与预期一致

2.9前处理器（Pre Processors）

用于在实际请求发出之前对即将发出的请求进行特殊处理。

例如：Count处理器可以实现自增操作，自增后生成的的数据可以被将要发出的请求使用，而HTTP URL Re—Writing Modifier处理器则可以实现URL重写，

当URL中有sessionID一类的session信息时，可以通过该处理器填充发出请求实际的sessionID。

2.10后处理器（Post Processors）

用于对Sampler发出请求后得到的服务器响应进行处理。一般用来提取响应中的特定数据（类似loadrunner中的关联）。

例如：Regular Expression Extractor用于提取响应数据中匹配某正则表达式的数据段，并将其填充在参数中，Xpath Extractor则可以用于提取响应数据中通过给定Xpath值获得的数据。。。

（2）录制脚本

对大多数刚开始接触性能测试的人来说，代码功力可能不是太好，我们可以通过工具，录制脚本来进行测试，以达到我们的目的

一般来讲，录制脚本有两种方法

一、利用badboy进行脚本录制

1、下载安装

badboy官网地址：http://www.badboy.com.au

提示：官网下载时候会有用户邮件验证的，直接continue跳过，下载即可

安装：和一般的Windows安装程序没区别，无脑下一步就行；安装完成后一般都会在桌面和开始菜单里面有badboy的快捷方式，如果没有，在badboy安装目录下找到badboy.exe文件，双击启动即可

启动：启动badboy之后，界面如下

2、录制

1）如上图，在地址栏（红色标注区域）中输入你需要录制的web应用的URL，这里以http://www.baidu.com为例子

2）点击开始录制按钮（地址栏上方圈出来的地方）开始录制

3）开始录制后，你可以在badboy内嵌的浏览器（界面右侧）对被测应用进行操作，所有操作过程都会记录在界面左侧的编辑窗口（黄色标注区域）

录制的脚本并不是一行行代码，而是一个web对象，有点类似于loadrunner中VuGen中的tree view视图

4）录制完成后，点击工具栏中的停止按钮（绿色标注区域），完成脚本的录制

5）点击file→save或者export to jmeter，将文件保存为jmeter的脚本格式：.jmx；启动jmeter，打开刚录制保存的文件，就可以进行测试了

二、利用jmeter代理服务器进行脚本录制

1、启动jmeter：在测试计划中添加线程组，线程组中添加逻辑控制器→录制控制器

2、工作台：添加非测试元件→http代理服务器

3、端口（代理服务器监听端口）：设置为8080（一般来说）

目标控制器：测试计划——线程组

分组选择：每个组放入一个新的控制器

4、http代理服务器：右键单击，添加定时器→高斯随机定时器（告知jmeter在其生成的http请求中自动增加一个定时器）

定时器会使相应的取样器被延迟：上一个请求发送被响应且延时指定时间后，下一个被定时器影响的取样请求才会被发送

如果在代理服务器中使用了高斯随机定时器，则应在其中的**固定延迟偏移**里添加：**${T}**：用于自动引用记录的延迟时间

5、打开浏览器，网络设置，将局域网设置中的代理服务器设为localhost，端口设置为8080

6、代理服务器配置后之后，点击启动，代理服务器就会开始记录所接受的http请求

7、在浏览器地址栏输入需要测试的地址并进行相关操作，录制完成后，停止http代理服务器，在录制控制器上点击右键，保存录制的脚本

注意：别忘了将代理服务器设置恢复原样

8、脚本录制完毕，启动jmeter，就可以进行测试了

（3）元件的作用域及执行顺序

jmeter是一个开源的性能测试工具，它可以通过鼠标拖拽来随意改变元件之间的顺序以及元件的父子关系，那么随着它们的顺序和所在的域不同，它们在执行的时候，也会有很多不同。

jmeter的test plan通过图形化的方式表达脚本，域代码方式的脚本不同，图形方式表达的脚本中无法使用变量和函数等描述元件的作用域，因此jmeter主要依靠test plan中元件的相对位置、

父子关系以及元件本身的类型来决定test plan中各元件的执行顺序；原件在test plan中的位置不同，可能导致该元件的行为有很大差异。（新版jmeter都可以自主选择语言，对号入座即可）

1、元件的作用域

jmeter中共有8类可被执行的元件（test plan和thread group不属于元件），其中，sampler（取样器）是不与其他元件发生交互的作用的元件，Logic Controller

（逻辑控制器）只对其子节点的sampler有效，而其他元件需要与sampler等元件交互。

Config Elements（配置元件）：影响其范围内的所有元件

Pre-porcessors（前置处理器）：在其作用范围内的每一个sampler元件之前执行

Timer（定时器）：对其作用范围内的每一个sampler有效

Post-porcessors（后置处理器）：在其作用范围内的每一个sampler元件之后执行

Assirtions（断言）：对其作用范围内的每一个sampler元件执行后的结果执行校验

Listener（监听器）：收集其作用范围内的每一个sampler元件的信息并且呈现出来

在jmeter中，元件的作用域是靠test plan的树形结构中元件的父子关系来确定的，其原则如下：

1） sampler不与其他元件相互作用，因此不存在作用域问题

2） Logic Controller只对其子节点中的sampler和Logic Controller作用

3） 除sampler和Logic Controller外的其他元件，如果是某个sampler的子节点，则该元件仅对其父节点作用

4） 除sampler和Logic Controller外的其他元件，如果其父节点不是sampler，则其作用域是该元件父节点下的其他所有后带节点（包括子节点，子节点的子节点等）

2、元件的执行顺序

在同一作用域范围内，test plan中的元件按照以下顺序执行：

1） **Config Elements**

2） **Pre-porcessors**

3） **Timer**

4） **Sampler**

5） **Post-porcessors**（除非Sampler得到的返回结果为空）

6） **Assirtions**（除非Sampler得到的返回结果为空）

7） **Listener**（除非Sampler得到的返回结果为空）

注意:Pre-porcessors、Post-porcessors和Assirtions等元件仅对Sampler作用，如在它们作用域内没有任何Sampler，则不会被执行；

如果在同一作用域范围内有多个同一类型的元件，则这些元件按照它们在test plan中的上下顺序依次执行。

（4）Sampler之SOAP/XML-RPC Request

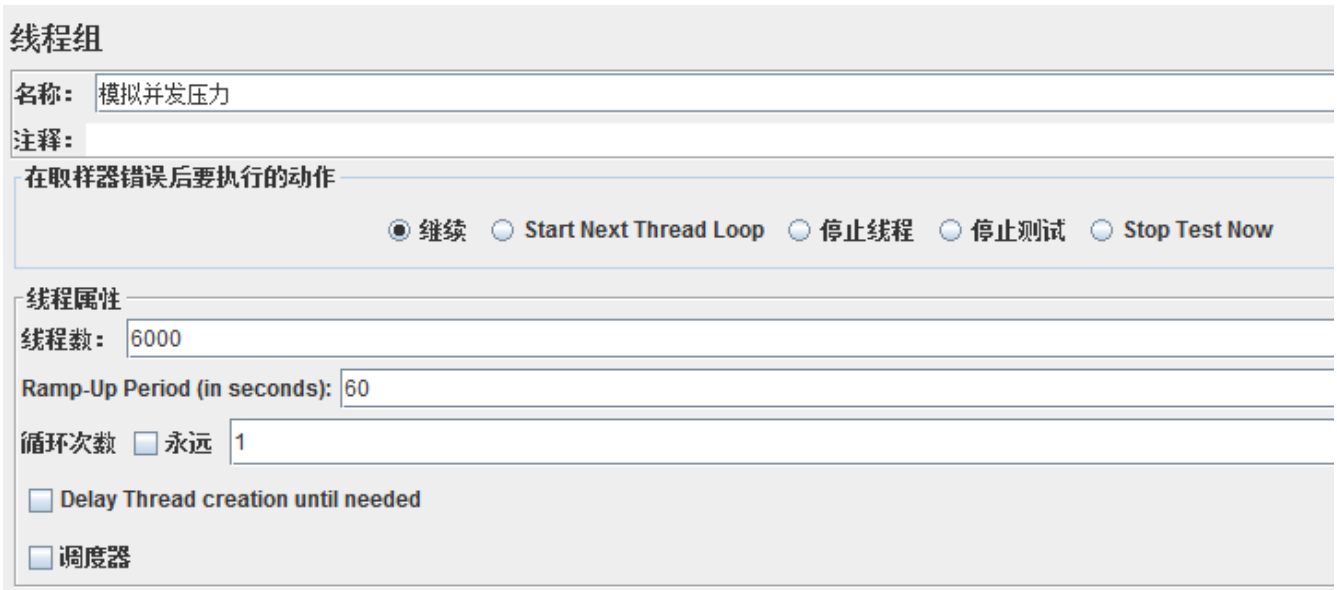
一、建立一个测试计划（test plan）

之前有说过，jmeter打开后会自动生成一个空的test plan，用户可以基于该test plan建立自己的test plan

一个性能测试的负载必须有一个线程组完成，而一个测试计划必须有至少一个线程组。添加线程组操作如下：

在测试计划处右键单击：添加→Threads（Users）→线程组

每个测试计划都必须包含至少一个线程组，当然，也可以包含多个，多个线程组的运行在jmeter中采用的是并行的方式，即：同时被初始化且同时执行其下的sampler



线程组主要包含三个参数：

线程数：虚拟用户的数量，一个线程指一个线程或者进程

Ramp—Up Period(in seconds)：准备时长。设置的线程数需要多久全部启动，比如上图，线程数为6000，启动时间为60，那么需要60S内启动6000个线程；

循环次数：每个线程发送请求的次数。如上图，6000个线程，每个线程发送1次，如果勾选了永远，那么它将永远发送下去，直到停止脚本；

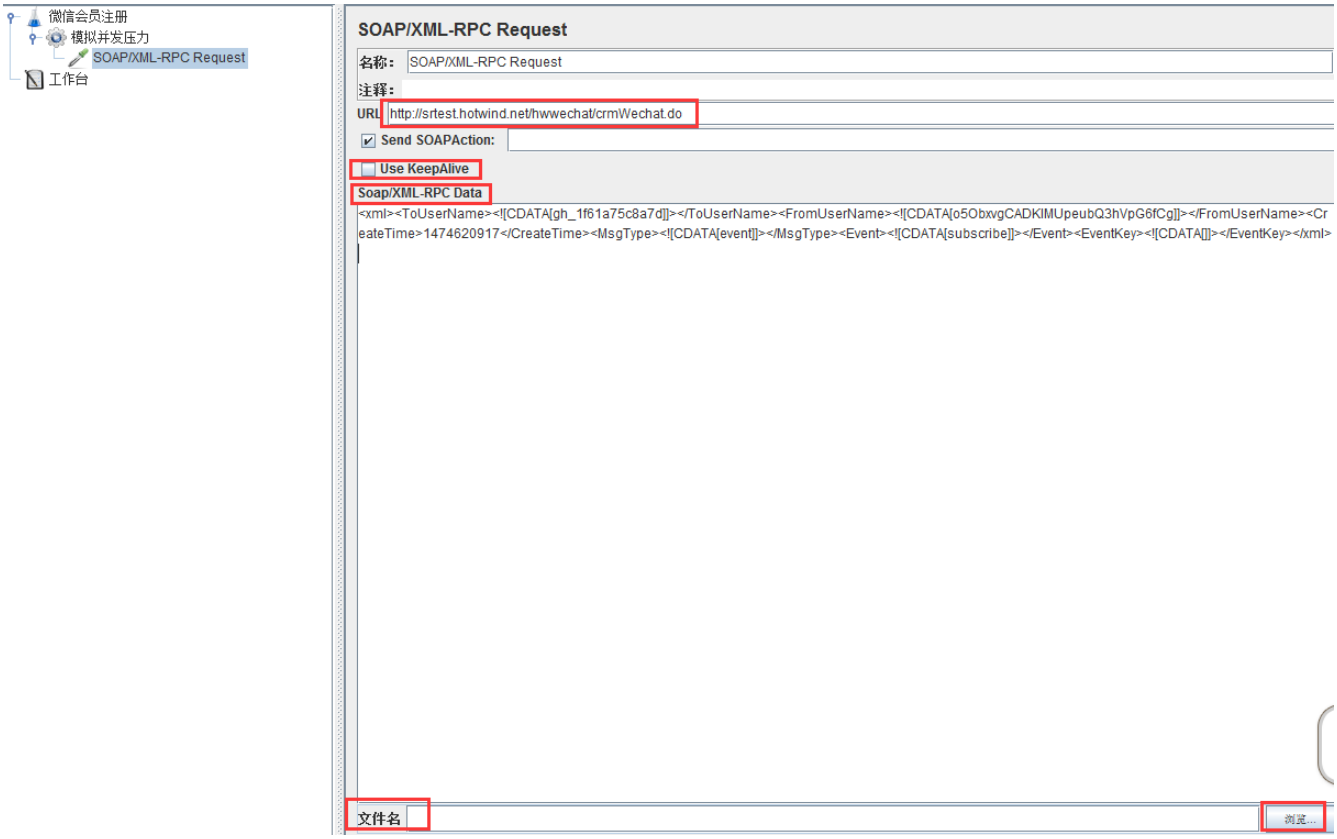
设置合理的线程数对能否达到测试目标有决定性影响。比如在本例中，如果线程数太少，则无法达到设定的要求；

另外，设置合理的循环次数也很重要，除了给定的设置循环次数和永远，还可以通过勾选**调度器**，设置开始和结束时间来控制。

二、添加sampler

添加完线程组后，在线程组上右键单击：添加→Sampler→SOAP/XML-RPC Request（SOAP/XML-RPC：都是报文中不同的数据格式）

前面说过，取样器（Sampler）是与服务器进行交互的单元。一个取样器通常进行三部分的工作：向服务器发送请求，记录服务器的响应数据和记录相应时间信息



这里解释一下，因为微信H5界面的会员注册，向微信端发送的是xml文件，所以这里我选择的取样器是SOAP/XML-RPC Request

上面的图中，选择SOAP/XML-RPC Request取样器，然后URL一栏输入我们需要进行加压的URL

然后默认选项，Use KeepAlive的意思是：保持连接，这个是http协议报文中的一个首部字段，之前的关于HTTP协议的随笔写过

下面的SOAP/XML-RPC Data输入需要发送的xml格式的文件就行（也可以导入xml文件的文件夹进行读取），下面是xml和json的区别：

xml	json
xml是一种可扩展标记语言 提供描述结构化数据的方法 用于定义数据本身结构和数据类型 简单讲，xml就是一个文件	常见的报文大多是json类型，json本身是一种数据类型 特点是键值对：即每个键对应一个值 其键值对数据由逗号分隔 花括号{}保存对象 方括号【】保存数组

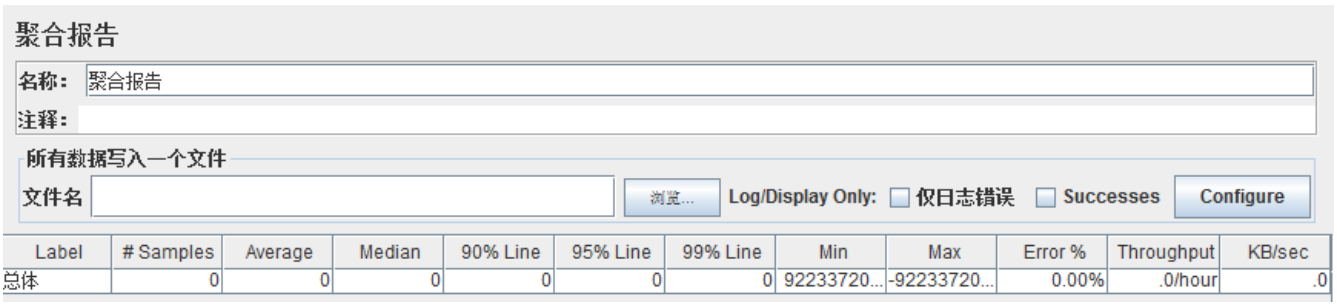
添加完取样器和具体的地址参数之后，接下来就是添加监听器，对测试结果进行获取

三、添加监听器

在线程组上右键单击，添加你需要的监听器，一般常用的就是结果树和聚合报告

添加后启动线程组进行测试，等线程执行完成后，根据结果树中的请求和响应结果（成功或者失败）就可以分析我们的测试是否成功，以及根据聚合报告结果来确认我们这次确认是否达成了预期结果。

四、聚合报告简析



Aggregate Report： **JMeter** 常用的一个 Listener，中文被翻译为“聚合报告”

Label：每个 JMeter 的 element（例如 HTTP Request）都有一个 Name 属性，这里显示的就是 Name 属性的值

#Samples：表示你这次测试中一共发出了多少个请求，如果模拟10个用户，每个用户迭代10次，那么这里显示100

Average：平均响应时间——默认情况下是单个 Request 的平均响应时间，当使用了 Transaction Controller 时，也可以以Transaction 为单位显示平均响应时间

Median：中位数，也就是 50% 用户的响应时间

90% Line：90% 用户的响应时间

Note：关于 50% 和 90% 并发用户数的含义，请参考下文

http://www.cnblogs.com/jackei/archive/2006/11/11/557972.html

Min：最小响应时间

Max：最大响应时间

Error%：本次测试中出现错误的请求的数量/请求的总数

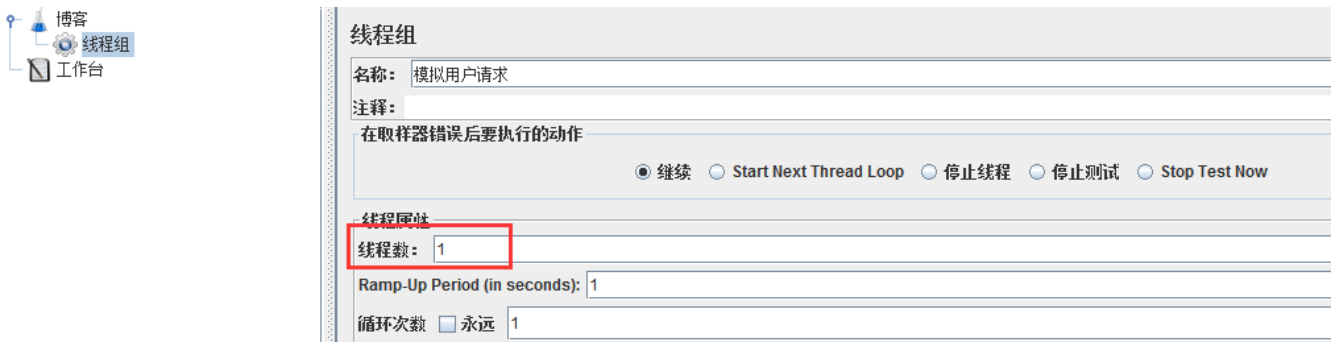
Throughput：吞吐量——默认情况下表示每秒完成的请求数（Request per Second），当使用了 Transaction Controller 时，也可以表示类似 **LoadRunner** 的 Transaction per Second 数

KB/Sec：每秒从服务器端接收到的数据量，相当于LoadRunner中的Throughput/Sec

（5）HTTP请求

启动jmeter，默认有一个测试计划，然后，修改计划名称，尽量使其变得有意义，容易看懂，然后，新建一个线程组

这里线程数我设置为1，方便演示



然后，添加一个http信息头管理器

这里解释一下为什么要添加http信息头管理器：

JMeter不是浏览器，因此其行为并不和浏览器完全一致。这些JMeter提供的配置元件中的HTTP属性管理器用于尽可能模拟浏览器行为，在HTTP协议层上发送给被测应用的http请求

（1）HTTP Request Defaults（请求默认值）

用于设置其作用范围内的所有HTTP的默认值，可被设置的内容包括HTTP请求的host、端口、协议等

（2）HTTP Authorization Manager（授权管理器）

用于设置自动对一些需要NTLM验证的页面进行认证和登录

（3）HTTP Cache Manager

用于模拟浏览器的Cache行为。为Test Plan增加该属性管理器后，Test Plan运行过程中会使用Last-Modified、ETag和Expired等决定是否从Cache中获取相应的元素

（4）HTTP Cookie Manager（cookie管理器）

用于管理Test Plan运行时的所有Cookie。HTTP Cookie Manager可以自动储存服务器发送给客户端的所有Cookie，并在发送请求时附加上合适的Cookie

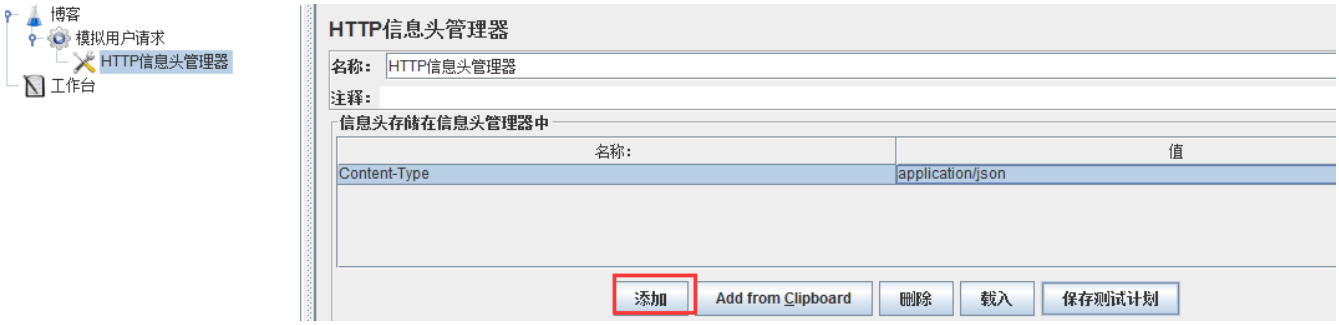
同时，用户也可以在HTTP Cookie Manager中手工添加一些Cookie，这些被手工添加的Cookie会在发送请求时被自动附加到请求

（5）HTTP Header Manager（信息头管理器）

用于定制Sampler发出的HTTP请求的请求头的内容。不同的浏览器发出的HTTP请求具有不同的Agent

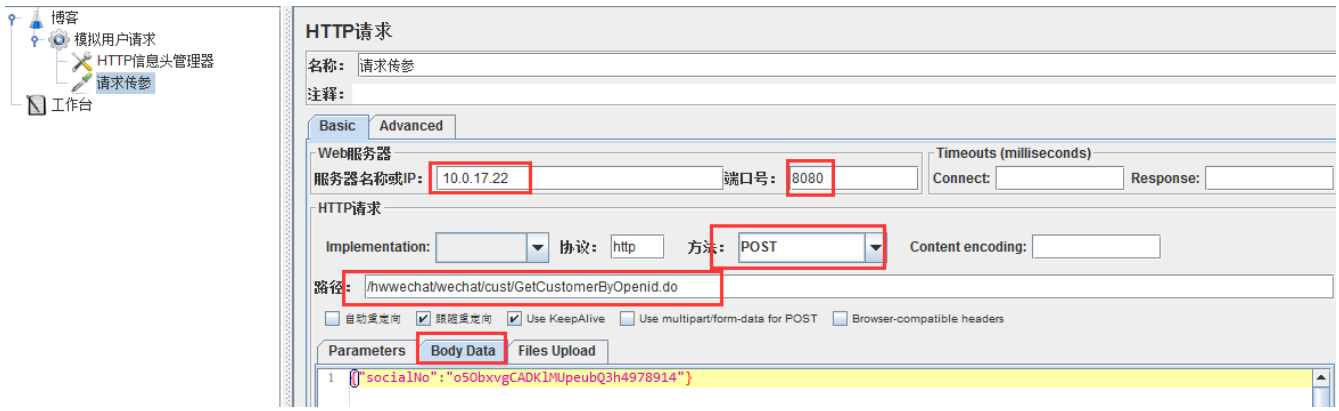
访问某些有防盗链的页面时需要正确的Refer...这些情况下都需要通过HTTP Header Manager来保证发送的HTTP请求是正确的

http信息头管理器添加好之后，需要填入信息头的名称以及对应的值，如下



Content-Type意思可以理解为参数名称、类型，值下面输入对应的参数类型就行了，这里我测试时候需要传输json类型，因此就填入了application/json

接着，添加Sampler（取样器）→http请求



按照截图所示，填入测试的服务器地址、端口、所用的户协议、方法，这里方法我用的是POST，然后填入路径，选择Body Data；

关于http请求的的属性参数说明：

1）名称：用于标识一个sample。建议使用一个有意义的名称

2）注释：对于测试没任何影响，仅用来记录用户可读的注释信息

3）服务器名称或IP：http请求发送的目标服务器名称或者IP地址，比如http://www.baidu.com

4）端口号：目标服务器的端口号，默认值为80，可不填

5）协议：向目标服务器发送http请求时的协议，http/https，大小写不敏感，默认http

6）方法：发送http请求的方法(链接：http://www.cnblogs.com/imyalost/p/5630940.html)

7）Content encoding：内容的编码方式（Content-Type=application/json;charset=utf-8）

8）路径：目标的URL路径（不包括服务器地址和端口）

9）自动重定向：如果选中该项，发出的http请求得到响应是301/302，jmeter会重定向到新的界面

10）Use keep Alive：jmeter 和目标服务器之间使用 Keep-Alive方式进行HTTP通信（默认选中）

11）Use multipart/from-data for HTTP POST：当发送HTTP POST 请求时，使用

12）Parameters、Body Data以及Files Upload的区别：

1. parameter是指函数定义中参数，而argument指的是函数调用时的实际参数

2. 简略描述为：parameter=形参(formal parameter)， argument=实参(actual parameter)

```
**3.**在不很严格的情况下，现在二者可以混用，一般用argument，而parameter则比较少用
```

While defining method, variables passed in the method are called parameters.

当定义方法时，传递到方法中的变量称为参数.

While using those methods, values passed to those variables are called arguments.

当调用方法时，传给变量的值称为引数.（有时argument被翻译为“引数”）

4、Body Data指的是实体数据，就是请求报文里面主体实体的内容，一般我们向服务器发送请求，携带的实体主体参数，可以写入这里

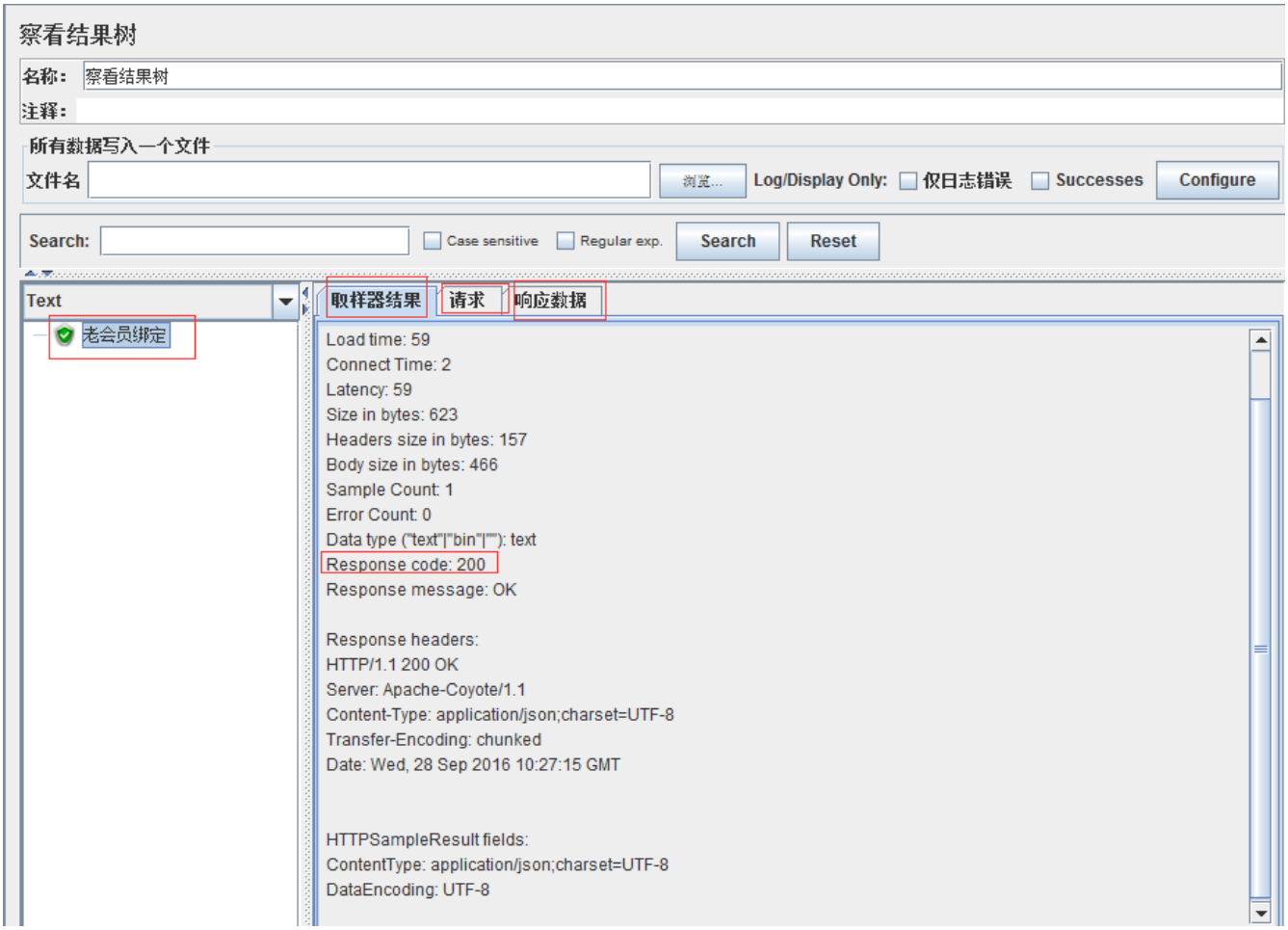
5、Files Upload指的是：从HTML文件获取所有有内含的资源：被选中时，发出HTTP请求并获得响应的HTML文件内容后还对该HTML

进行Parse 并获取HTML中包含的所有资源（图片、flash等）：（默认不选中）

如果用户只希望获取特定资源，可以在下方的Embedded URLs must match 文本框中填入需要下载的特定资源表达式，只有能匹配指定正则表达式的URL指向资源会被下载

接下来可以给这个测试计划添加一个监视器，常用的监视器有“查看结果树”和“聚合报告”

添加好监视器，点击运行，开始测试

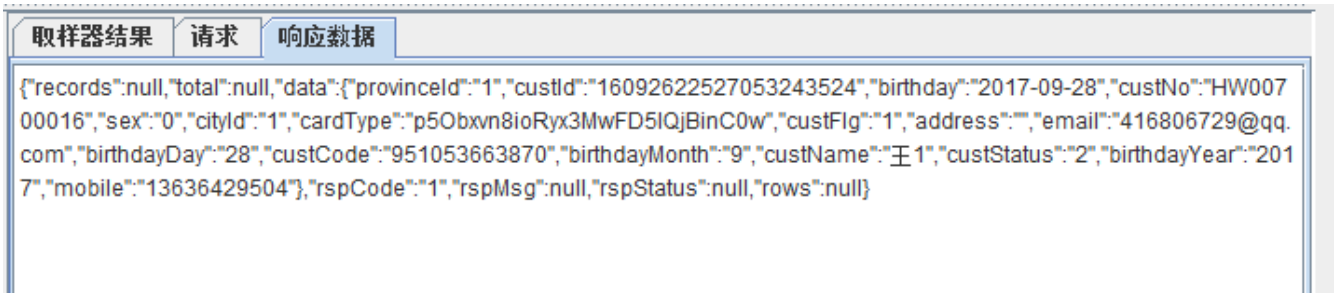


如上，测试结束后，如果我们的请求成功发送给服务器，那么结果树里面的模拟请求会显示为绿色，可以通过取样器结果里面的响应状态码信息来判断

也可以点击请求模块，查看我们发送的请求

里面有我们发送的请求的方法、协议、地址以及实体主体数据，以及数据类型，大小，发送时间，客户端版本等信息

响应数据：里面包含服务器返回给我们的响应数据实体，如下图



（6）HTTP请求之content-type

本文讲三种content-type以及在Jmeter中对应的参数输入方式

第一部分：目前工作中涉及到的content-type 有三种：

content-type：在Request Headers里，告诉服务器我们发送的请求信息是哪种格式的。

1 content-type:**application/x-www-form-urlencoded**

默认的。如果不指定content-type，默认使用此格式。

参数格式：key1=value1&key2=value2

2 content-type:application/json

参数为json格式

{

"key1":"value1",

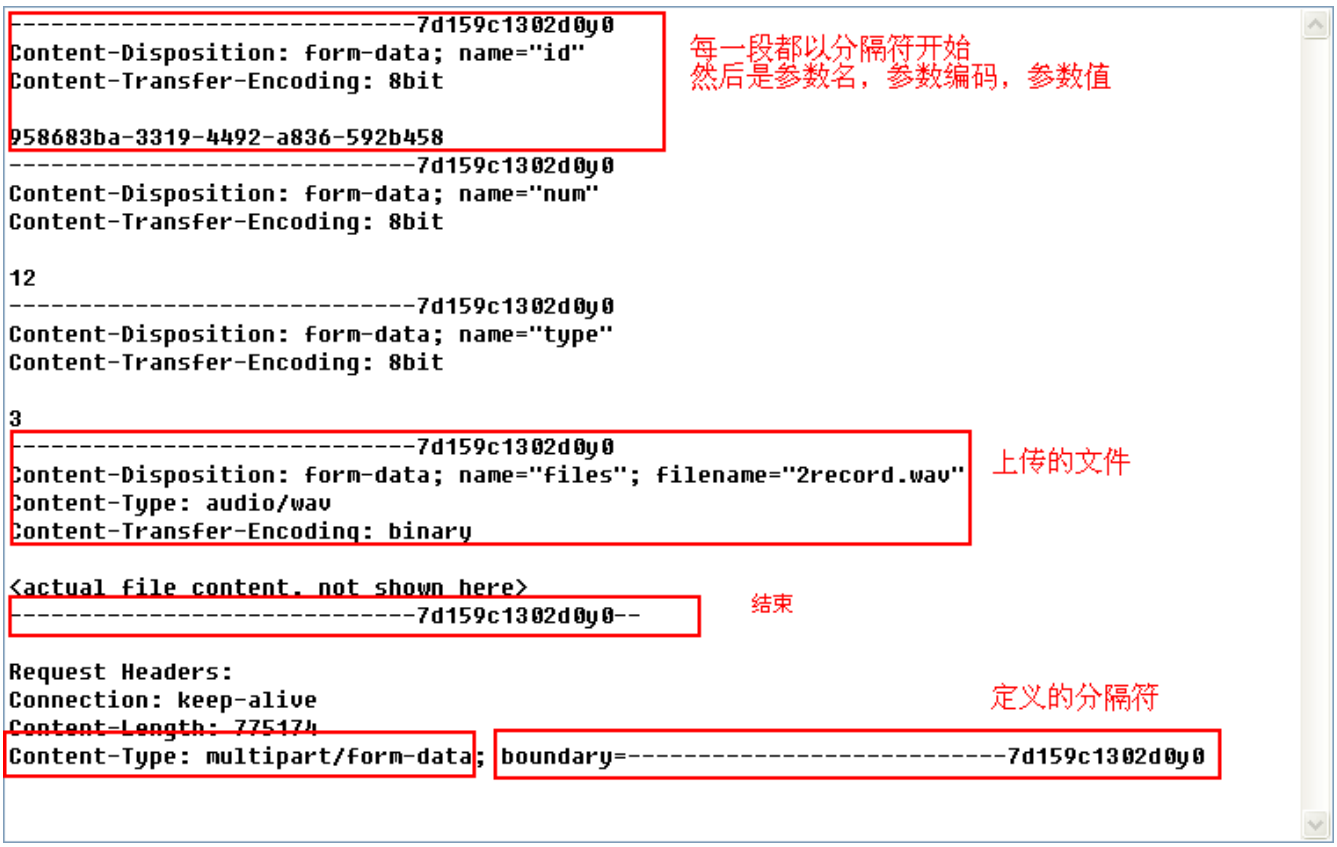
"key2":"value2"

}

3 content-type:multipart/form-data [dinghanhua]

上传文件用这种格式

发送的请求示例：



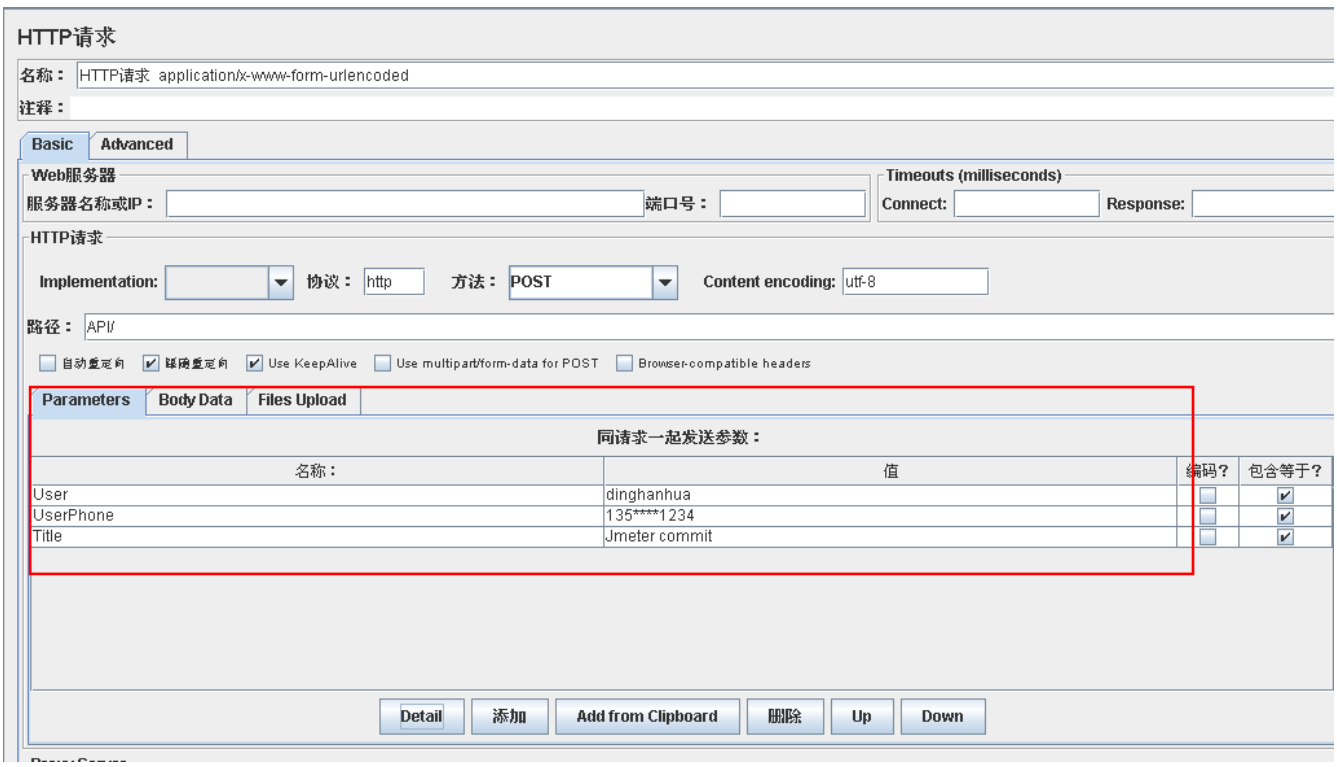
第二部分 不同的content-type如何输入参数

1 content-type:application/x-www-form-urlencoded

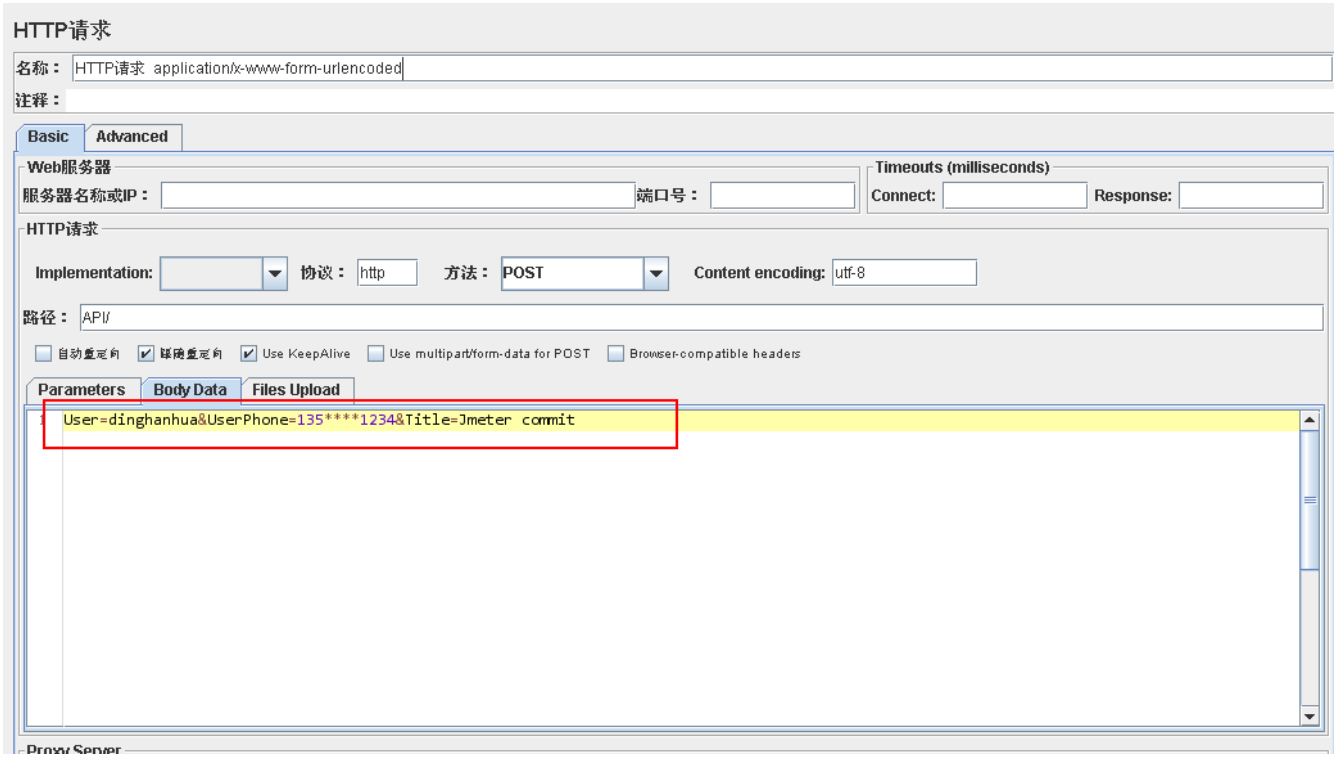
参数可以在Parameters或Body Data里输入，格式不同，如下图所示。**

这两个参数输入的tab页只能使用一个，某一个有数据后不能切换到另一个。

Parameters：

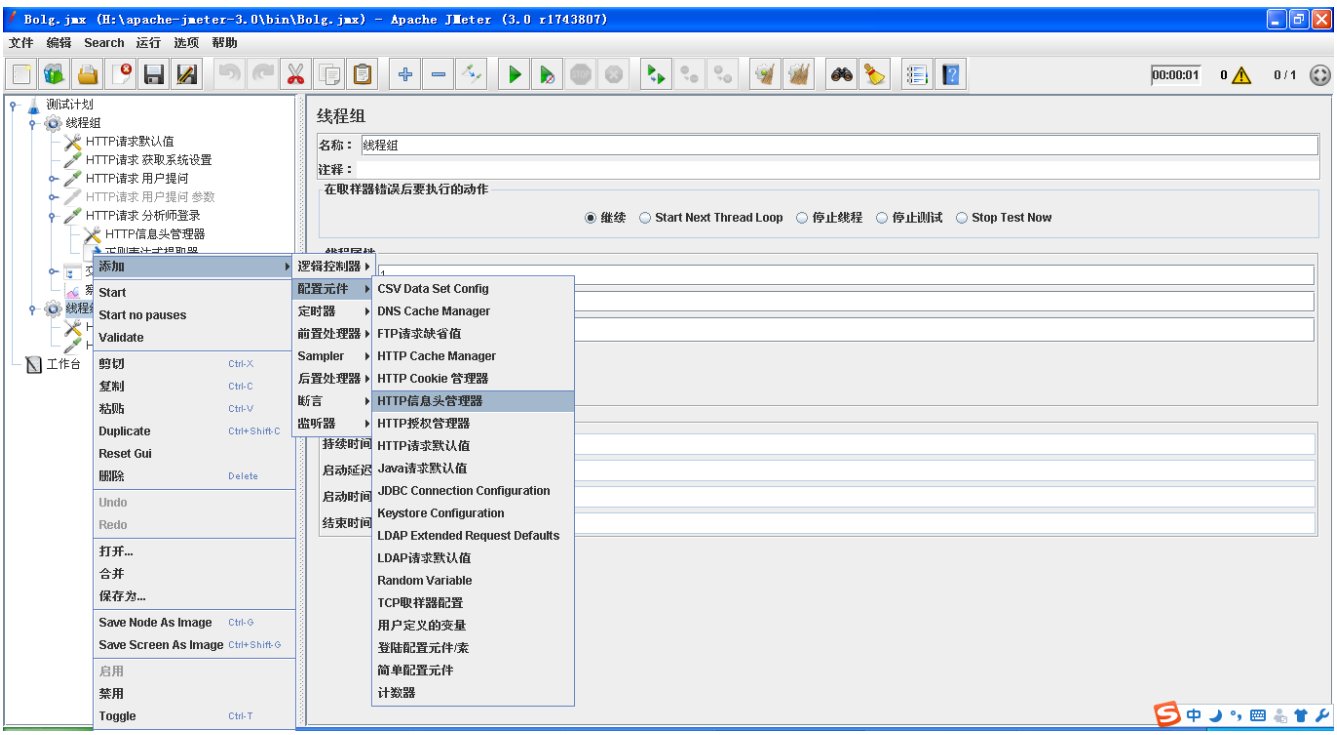


Body Data：

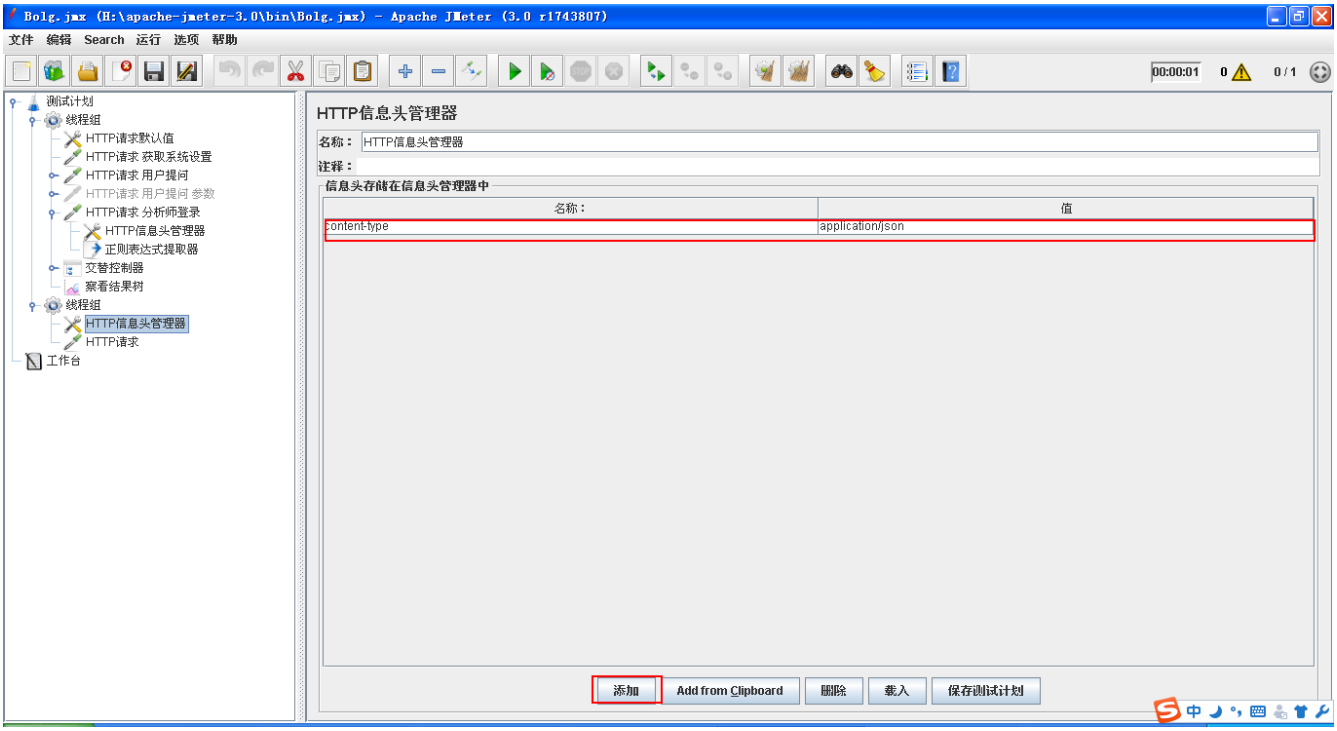


2 content-type:application/json

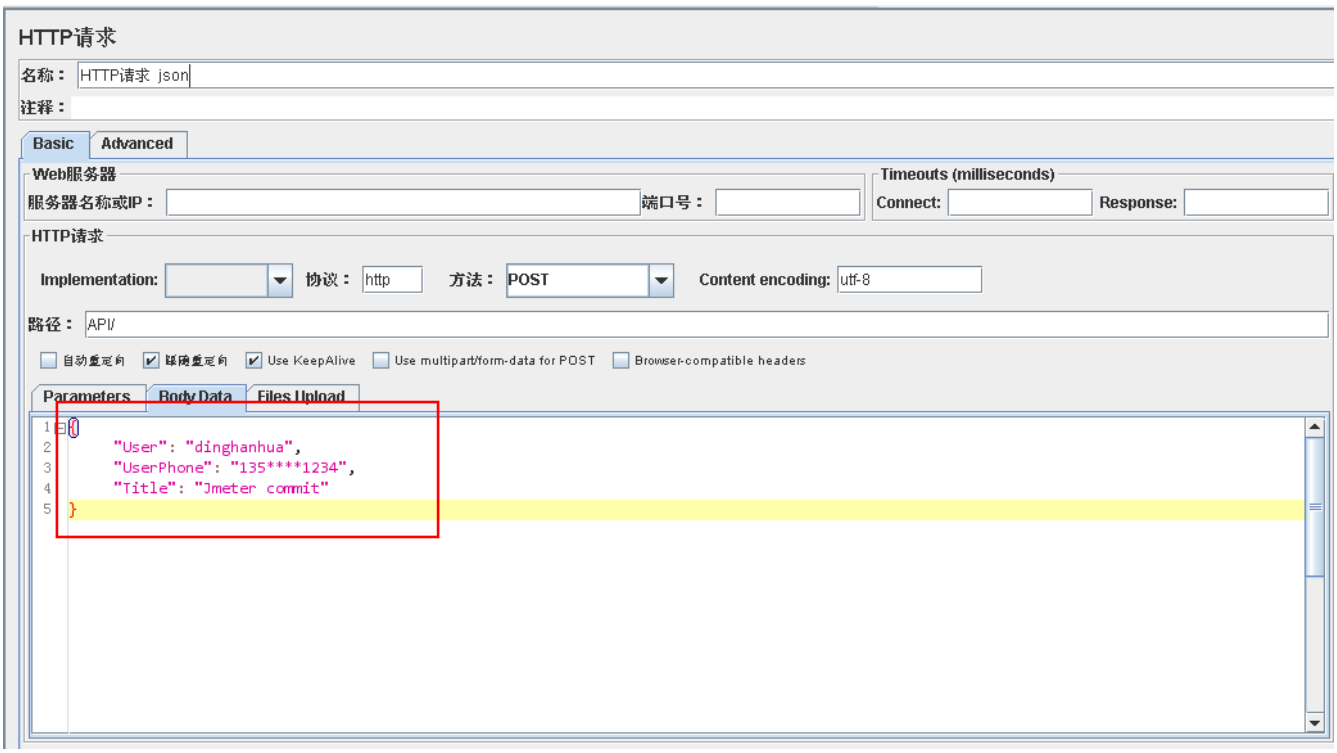
2.1 首先添加信息头管理。http请求上点击右键》添加》配置元件》 HTTP信息头管理器



2.2 信息头编辑页面，点击添加，输入content-type application/json



2.3 在http请求，Body Data中输入json格式的参数

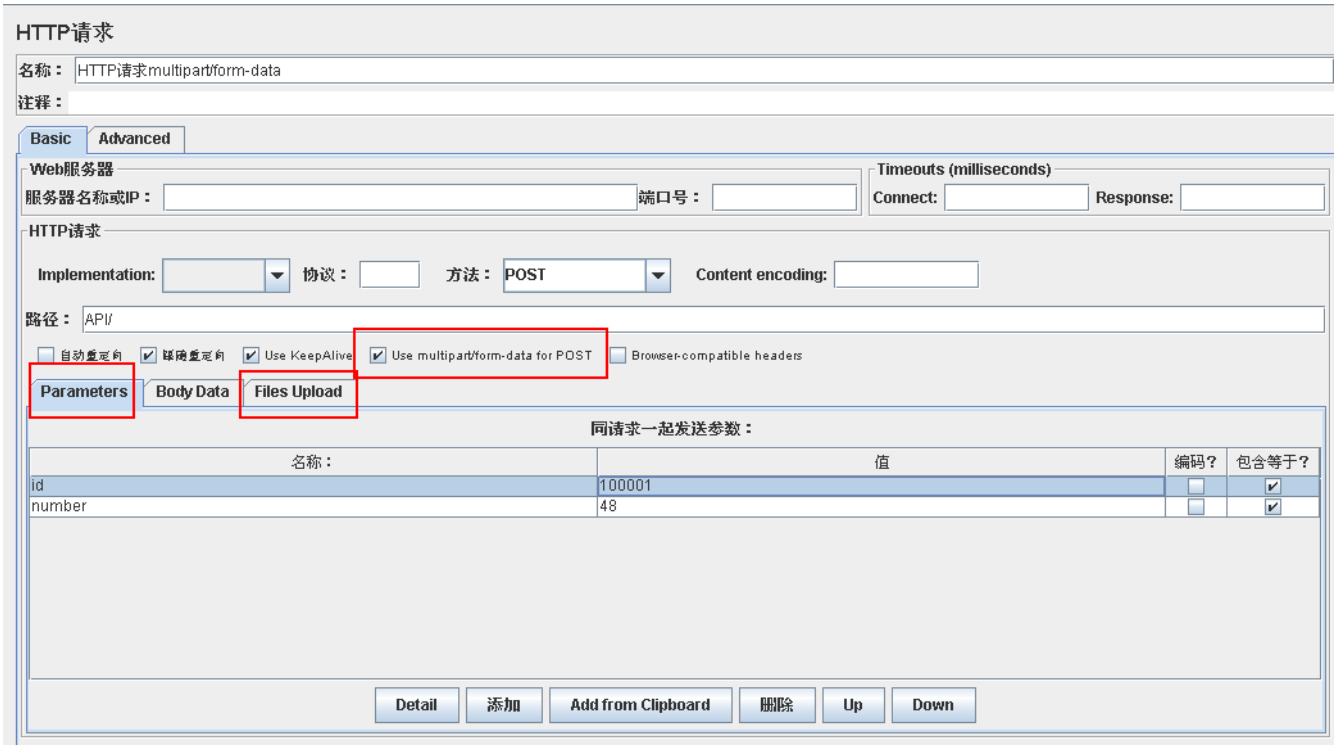


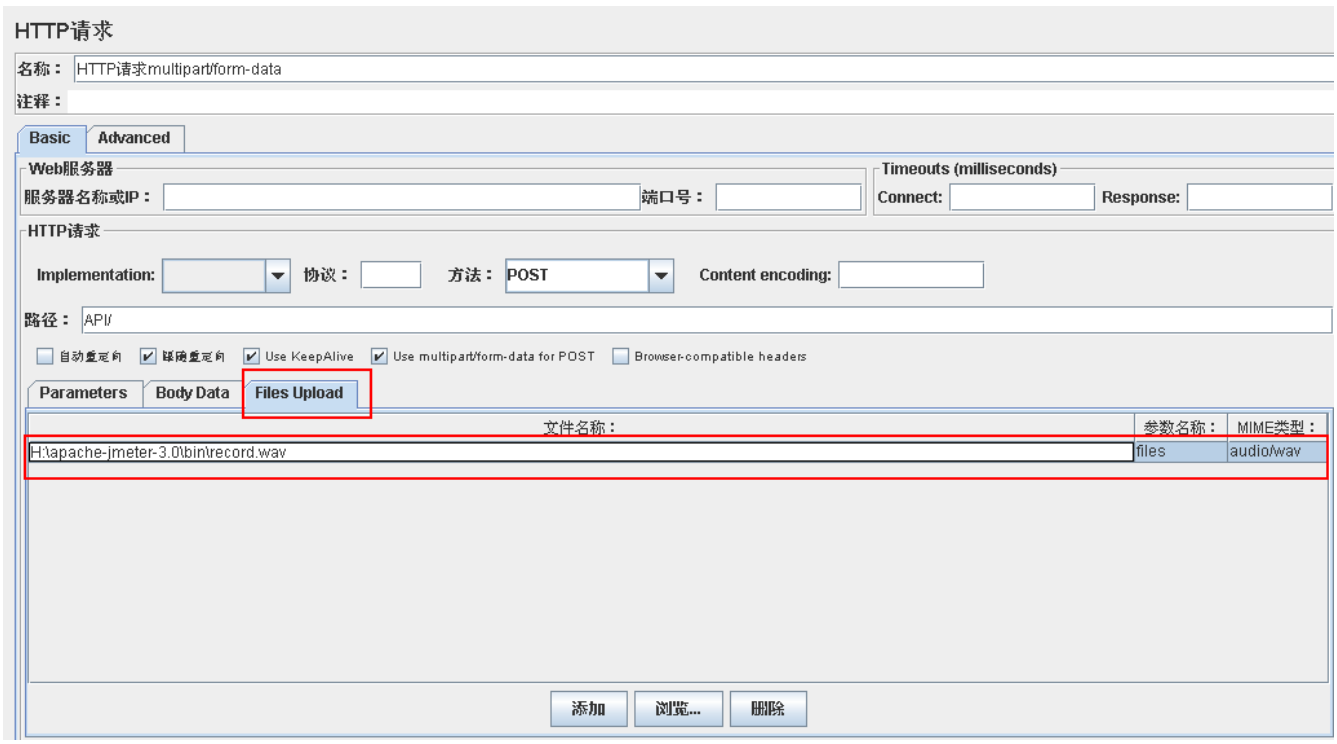
3 content-type:multipart/form-data [dinghanhua]

在http请求编辑页面，选中Use multipart/form-data for POST

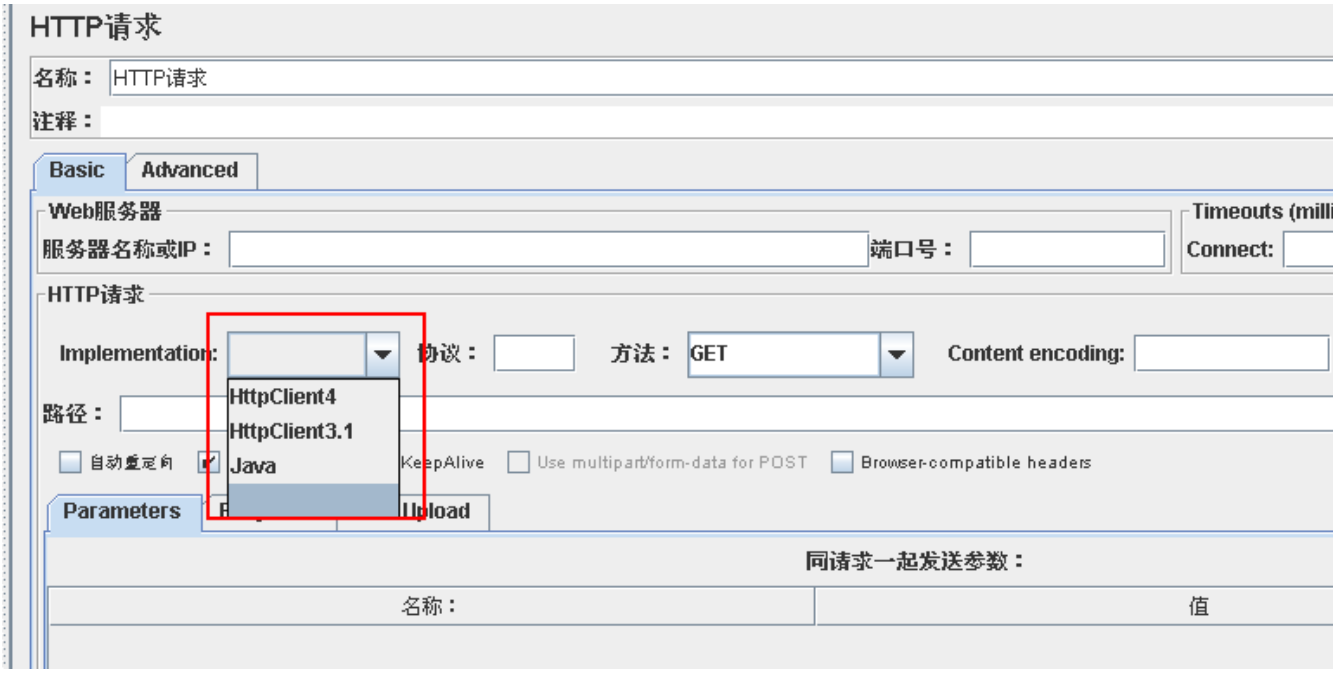
Parameters中输入除了上传的文件以外的参数：参数名和参数值

Files Upload中上传文件，参数名和MIME类型





上传文件如果不成功，修改Implementation为java试一下。



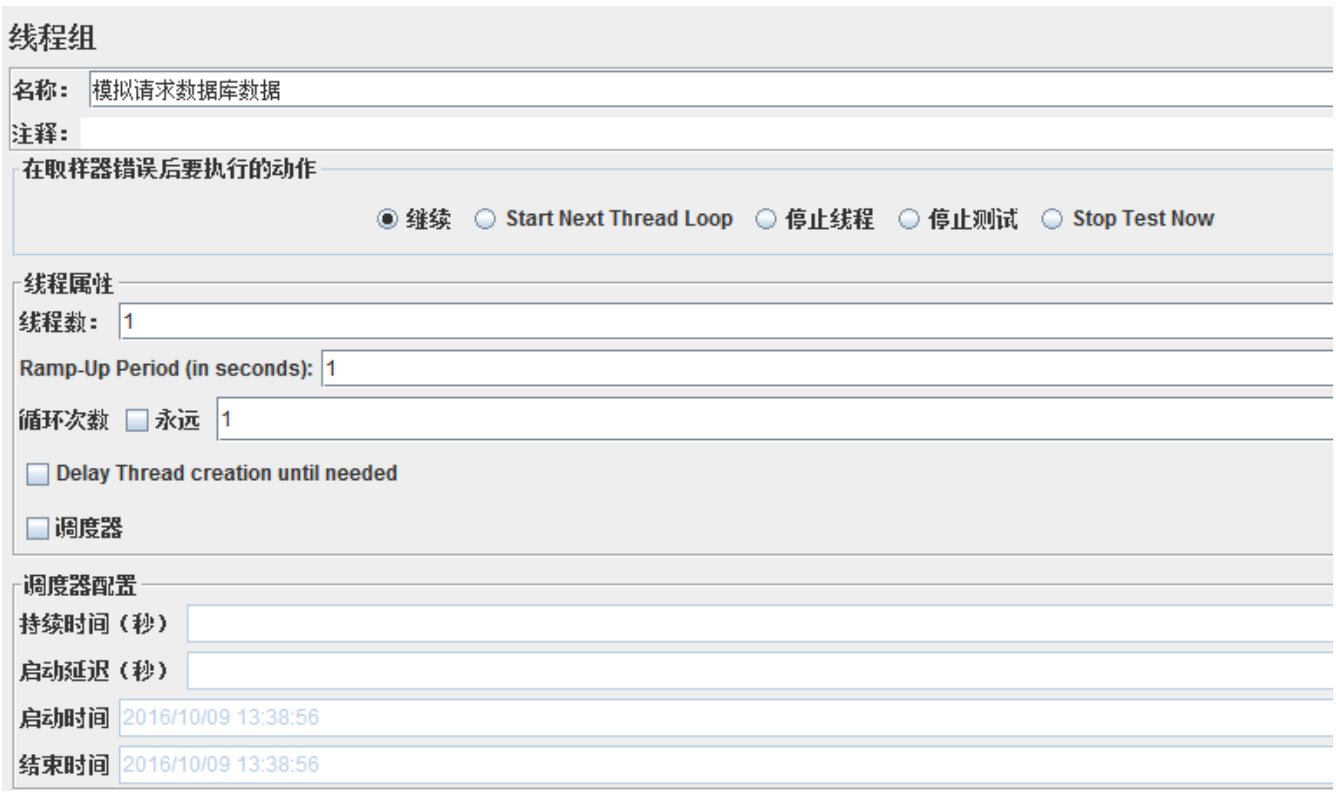
（7）Sample之JDBC Request

jmeter中取样器（Sampler）是与服务器进行交互的单元。一个取样器通常进行三部分的工作：向服务器发送请求，记录服务器的响应数据和记录响应时间信息

有时候工作中我们需要对数据库发起请求或者对数据库施加压力，那么这时候就需要用到**JDBC Request**

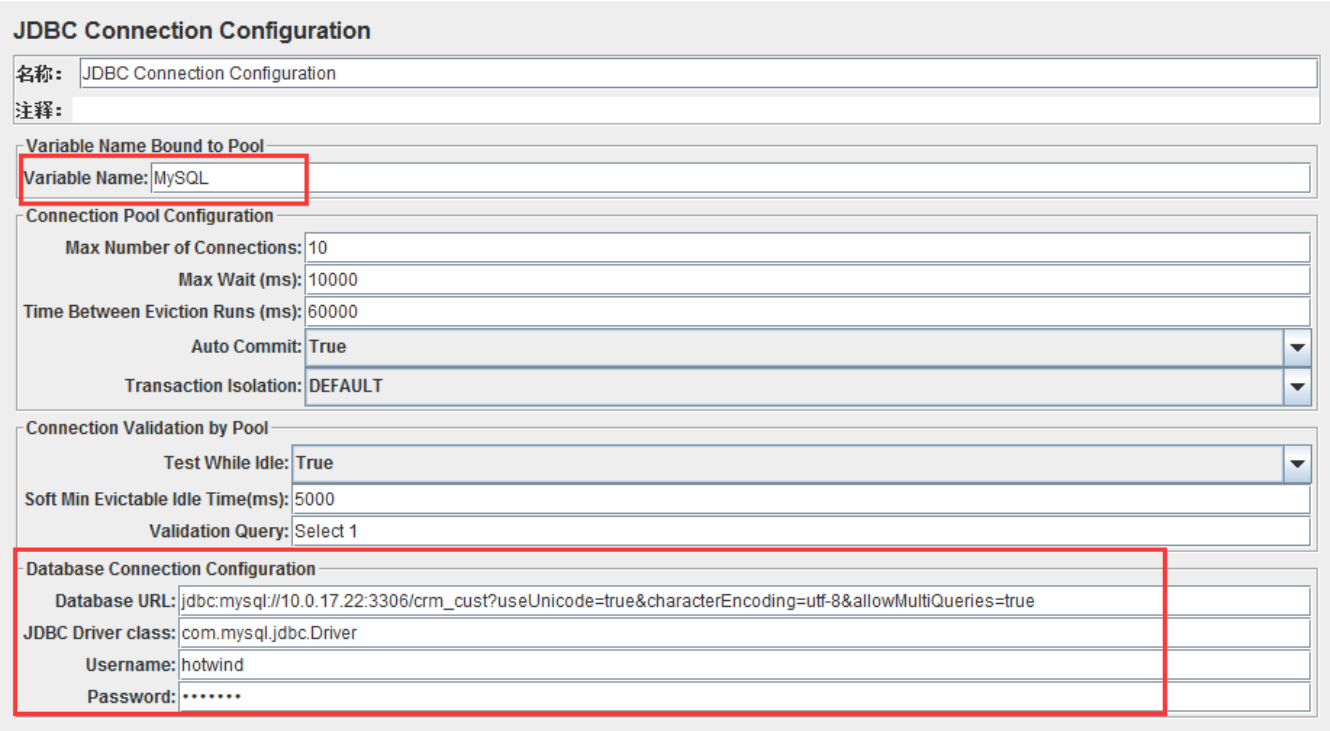
JDBC Request可以向数据库发送一个请求（sql语句），一般它需要配合JDBC Connection Configuration配置元件一起使用

首先，还是先建立一个测试计划，添加线程组



为了方便，这里线程数我设置为1，然后在线程组上面右键单击选择配置元件→ **JDBC Connection Configuration（JDBC连接配置）**

JDBC Connection Configuration界面如下：



Variable Name（变量名）： 这里写入数据库连接池的名字

Database URL：数据库连接地址

JDBC Driver class：数据库驱动（可以将需要连接的数据库驱动jar包复制到jmeter的lib/目录下，然后在设置测试计划界面，最下面的Library中导入）

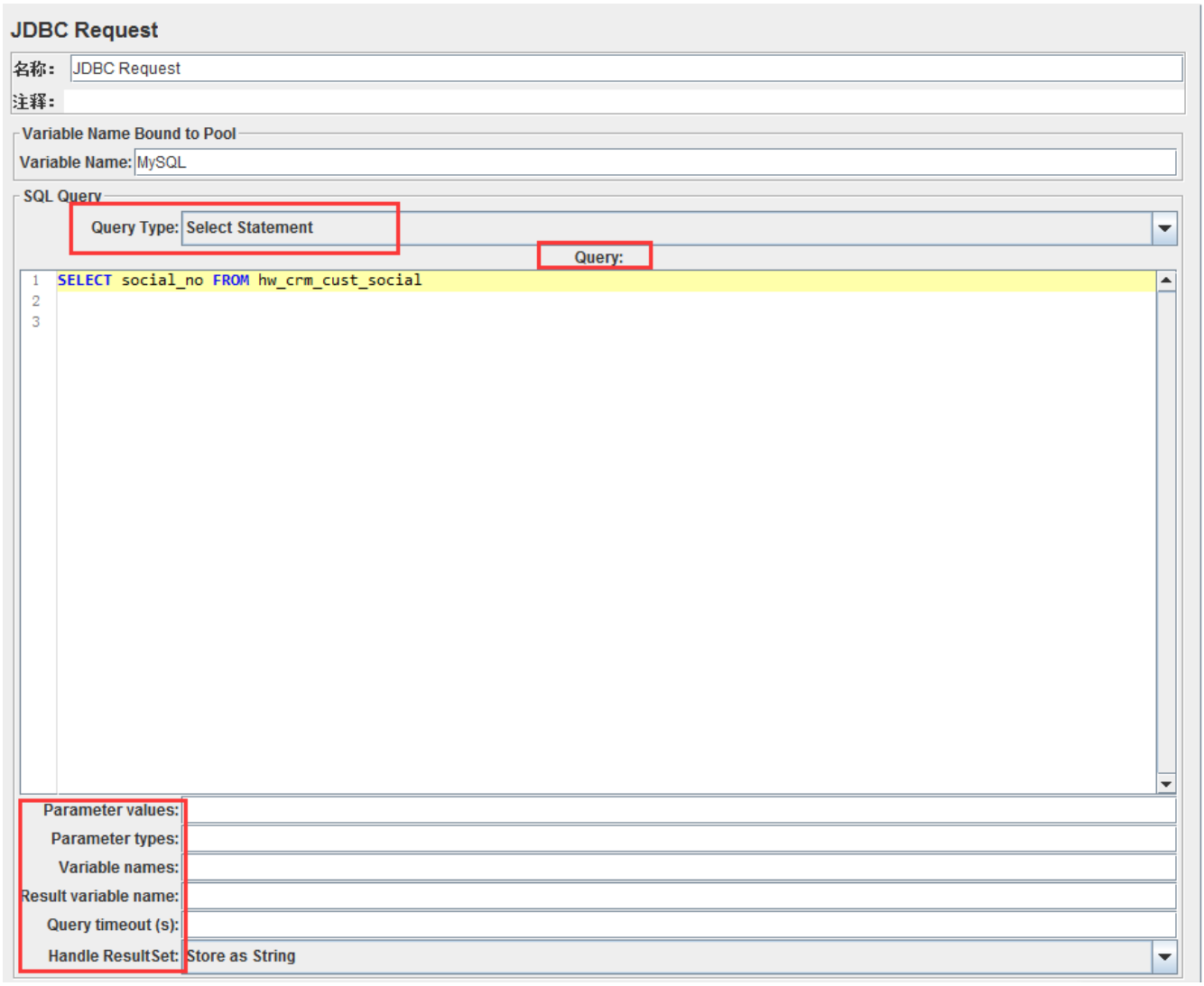
Username：数据库登录名

Password：数据库登陆密码

这里顺带说说不同数据库的驱动类和URL格式：

Datebase	Driver class	Datebase URL
MYSQL	com.mysql.jdbc.Driver	jdbc:mysql://host:port/{dbname}
PostgreSQL	org.postgresql.Driver	jdbc:postgresql:{dbname}
Oracle	oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:@//host:port/service OR jdbc:oracle:thin:@(description=(address=(host={mc-name}) (protocol=tcp)(port={port-no})) (connect_data=(sid={sid})))
Ingres(2006)	ingres.jdbc.IngresDriver	jdbc:ingres://host:port/db[;attr=value]
MSSQL	com.microsoft.sqlserver.jdbc.SQLServerDriver 或者 net.sourceforge.jtds.jdbc.Driver	jdbc:sqlserver://IP:port;databaseName=DBname 或者 jdbc:jtds:sqlserver://localhost:1433/"+"library"

设置好JDBC连接配置后，添加JDBC请求，界面如下：



Variable name：这里写入数据库连接池的名字（和JDBC Connection Configuration名字保持一致 ）

Query：里面填入查询数据库数据的SQL语句（填写的SQL语句末尾不要加“；”）

parameter valus：数据的参数值

parameter types：数据的参数类型

cariable names：保存SQL语句返回结果的变量名

result cariable name：创建一个对象变量，保存所有返回结果

query timeout：查询超时时间

handle result set：定义如何处理由callable statements语句返回的结果

完成了上面的操作后，就可以添加监听器，来查看我们的请求是否成功了

这是请求内容，即SQL语句

这是响应数据，正确的显示了我查询的该表的对应字段的数据

jmeter这个工具还是蛮强大的，每个组件的作用都不同，不同组合的情况下，可以实现95%的性能测试的辅助工作。。。。。。

（8）JDBC Request之Query Type

工作中遇到这样一个问题：

需要准备10W条测试数据，利用jmeter中的JDBC Request向数据库中批量插入这些数据（只要主键不重复就可以，利用函数助手中的**Random**将主键的ID末尾五位数随机插入）；

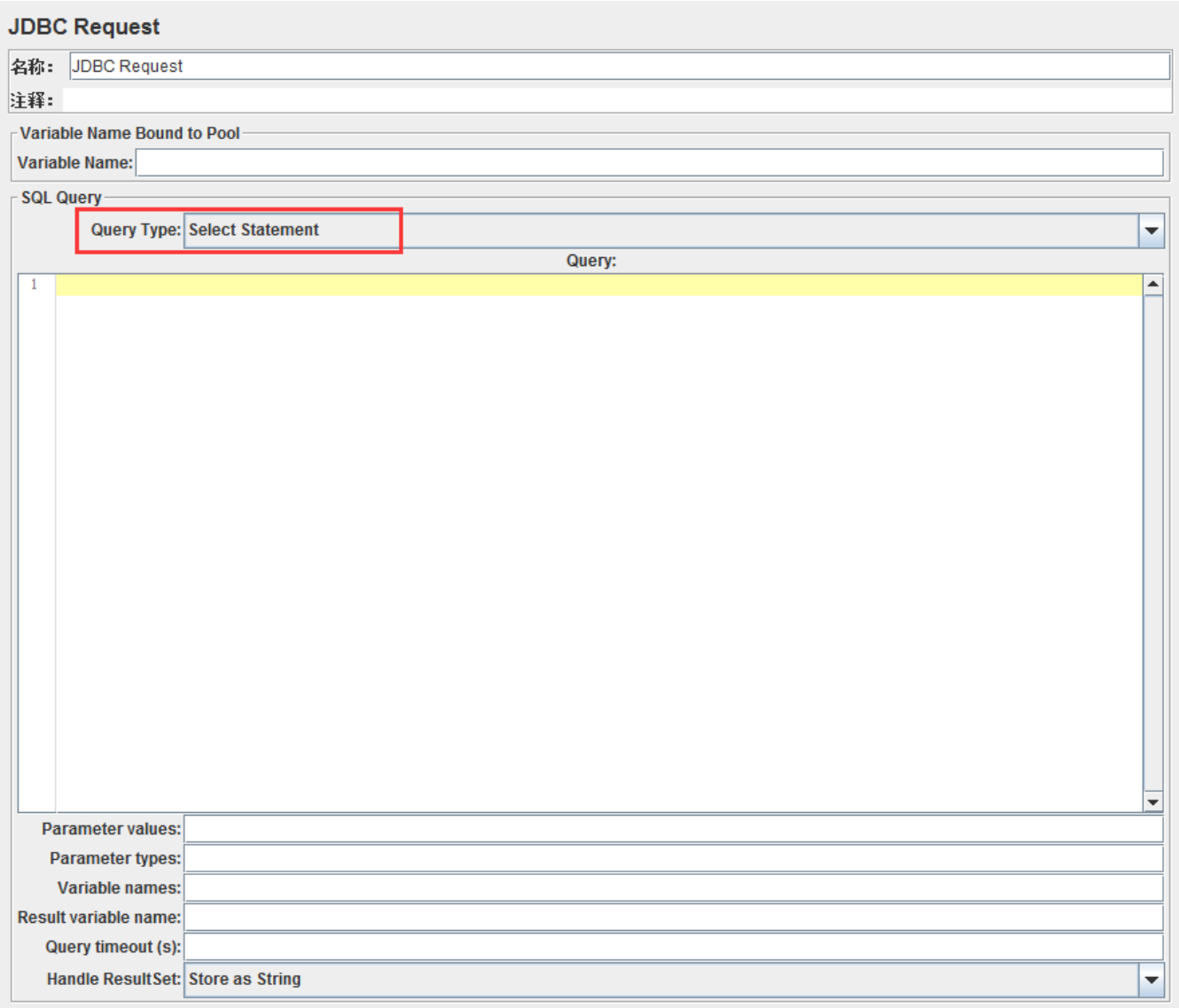
响应数据报错：Can not issue data manipulation statements with executeQuery().后来查阅了很多资料，才发现跟JDBC Request中的Query Type类型选择有关；

最后得出的结论是：如果SQL语句是update、insert等更新语句，应该使用statement的execute()方法；如果使用statement的executeQuery()就会出现报错。

PS：之前的博客有简单介绍JDBC Request怎样使用，传送门：
http://www.cnblogs.com/imyalost/p/5947193.html

下面主要说jmeter的JDBC Request请求中的**Query Type**：

JDBC Request界面如下：



其中Query Type（SQL语句类型）包含十个类型，每个类型作用都不同，下面分别介绍。

1、Select statement

这是一个查询语句类型；如果JDBC Request中的Query内容为**一条**查询语句，则选择这种类型。

PS：多个查询语句(不使用参数的情况下)可以放在一起顺序执行，需要设置Query Type为：Callable Statement；

如果Query Type为：select Statement，则只执行第一条select语句。

2、Update statement

这是一个更新语句类型（包含insert和update）；如果JDBC Request中的Query内容为**一条**更新语句，则选择这种类型。

PS：如果该类型下写入多条update语句，依然只执行第一条（原因同上，具体下面介绍）。

3、Callable statement

这是一个可调用语句类型，CallableStatement 为所有的 DBMS 提供了一种以标准形式调用已储存过程的方法。

已储存过程储存在数据库中，对已储存过程的调用是 CallableStatement 对象所含的内容。

这种调用是用一种换码语法来写的，有两种形式：一种形式带结果参数，另一种形式不带结果参数；结果参数是一种输出 (OUT) 参数，是已储存过程的返回值。

两种形式都可带有数量可变的输入（IN 参数）、输出（OUT 参数）或输入和输出（INOUT 参数）的参数，问号将用作参数的占位符。

在 JDBC 中调用已储存过程的语法如下所示。注意，方括号表示其间的内容是可选项；方括号本身并不是语法的组成部份。

{call 过程名[(?, ?, ...)]}，返回结果参数的过程的语法为： {? = call 过程名[(?, ?, ...)]}；

不带参数的已储存过程的语法类似：{call 过程名}。

更详细的使用方法可参考这篇文章：**http://blog.csdn.net/imust_can/article/details/6989954**

4、Prepared select statement

statement用于为一条SQL语句生成执行计划（这也是为什么select statement只会执行第一条select语句的原因），如果只执行一次SQL语句，statement是最好的类型；

Prepared statement用于绑定变量重用执行计划，对于多次执行的SQL语句，Prepared statement无疑是最好的类型（生成执行计划极为消耗资源，两种实现速度差距可能成百上千倍）；

PS：PreparedStatement的第一次执行消耗是很高的. 它的性能体现在后面的重复执行。

更详细的解释请参考这一篇文章：**http://blog.csdn.net/jiangwei0910410003/article/details/26143977**

5、Prepared update statement

Prepared update statement和Prepared select statement的用法是极为相似的，具体可以参照第四种类型。

6、Commit

commit的意思是：将未存储的SQL语句结果写入数据库表；而在jmeter的JDBC请求中，同样可以根据具体使用情况，选择这种Query类型。

7、Rollback

rollback指的是：撤销指定SQL语句的过程；在jmeter的JDBC请求中，同样可以根据需要使用这种类型。

8、AutoCommit(false)

MySQL默认操作模式就是autocommit自动提交模式。表示除非显式地开始一个事务，否则每条SQL语句都被当做一个单独的事务自动执行；

我们可以通过设置autocommit的值改变是否是自动提交autocommit模式；

而AutoCommit(false)的意思是AutoCommit（假），即将用户操作一直处于某个事务中，直到执行一条commit提交或rollback语句才会结束当前事务重新开始一个新的事务。

9、AutoCommit(true)

这个选项的作用和上面一项作用相反，即：无论何种情况，都自动提交将结果写入，结束当前事务开始下一个事务。

10、编辑（${}）

jmeter中的JDBC请求中的SQL语句是无法使用参数的，比如： SELECT * FROM ${table_name} 是无效的。

如果需实现同时多个不同用户使用不同的SQL，可以通过把整条SQL语句参数化来实现；（把SQL语句放在csv文件中，然后在JDBC Request的Query 中使用参数代替 ${SQL_Statement}）。

备注：后面的七项项涉及到数据库的事务控制等知识点，如果有不明白的地方请自行查询相关知识，或等待几天，我会将数据库事务管理等知识发布上来，供参考。。。

（9）jmeter目录结构

之前了解过jmeter的目录结构，但只知道一些常用的配置文件，看到一篇介绍的比较详细的博客，就转载过来，当然，其实是自己懒得再去搜集更多资料慢慢看了，时间不够用。。。

原文链接：http://www.cnblogs.com/zichuan/p/6938772.html，作者：**zzz紫川**

首先得了解一下这些东西，以后才能快速的找到某些配置文件进行修改（举个例子，改配置只是其中之一）

一、bin目录 examples: 目录中有CSV样例

jmeter.bat windows的启动文件

jmeter.log jmeter运行日志文件

jmeter.sh linux的启动文件

jmeter.properties 系统配置文件

jmeter-server.bat windows分布式测试要用到的服务器配置

jmeters-server linux分布式测试要用的服务器配置

其中系统配置文件中的SSL设置重点关注如下几个：

指定HTTPS协议层

https.default.protocol=TLS

指定SSL版本

https.default.protocol=SSLv3

设置启动的协议

https.socket.protocols=SSLv2Hello SSLv3 TLSv1

缓存控制，控制SSL是否可以在多个迭代中重用

https.use.cached.ssl.context=true

二、docs目录

接口文档目录。例C:\apache-jmeter-3.0\docs\api下的index.html

三、extras目录

扩展插件目录。提供了对Ant的支持，可以使用Ant来实现自动化测试，例如批量脚本执行，产生html格式的报表，测试运行时，可以把测试数据记录下来，jmeter会

自动生成一个.jtl文件，将该文件放到extras目录下，运行"ant -Dtest=文件名 report"，就可以生成测试统计报表。

四、lib目录

所用到的插件目录，里面均为jar包。jmeter会自动在jmeter_HOME/lib和ext目录下寻找需要的类，lib下存放JMeter所依赖的外部jar，

如：httpclient.jar、httpcore.jar、httpmime.jar等等。

其中lib\ext目录下存放有Jmeter依赖的核心jar包，ApacheJMeter_core.jar、ApacheJMeter_java.jar在写client端需要引用，JMeter插件包也在此目录下。

lib\junit下存放junit测试脚本.

五、Licenses目录

jmeter证书目录

六、Printable_docs目录

用户使用手册，例C:\apache-jmeter-3.0\printable_docs下的index.html

（10）参数化

参数化是自动化测试脚本的一种常用技巧。简单来说，参数化的一般用法就是将脚本中的某些输入使用参数来代替，在脚本运行时指定参数的取值范围和规则；

这样，脚本在运行时就可以根据需要选取不同的参数值作为输入。这种方式通常被称为数据驱动测试（Data Driven Test），参数的取值范围被称为数据池（Data Pool）。

jmeter的test plan中，支持如下**4种参数化方式**：

函数助手：_CSVRead

CSV Data Set Config：CSV数据控件

User Defined Variables：用户定义的变量

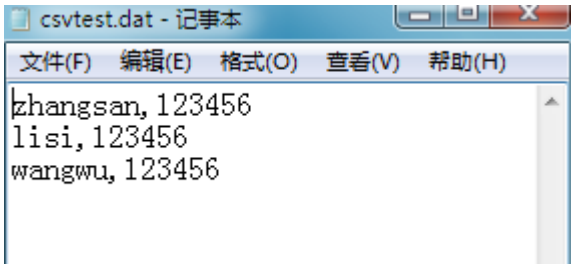
User Variables：用户参数

首先新建一个测试脚本，可以通过工具（badboy）录制或者自己手动编写

登录请求的界面如下：

这里我们对登录的用户名密码进行参数化，将用户名密码写入txt文档，**保存为.dat格式，编码类型选择UTF-8**；

因为配置元件——CSV Data Set Config对参数化的格式要求比较严格，用户名密码一一对应，之间用**半角英文逗号隔开**

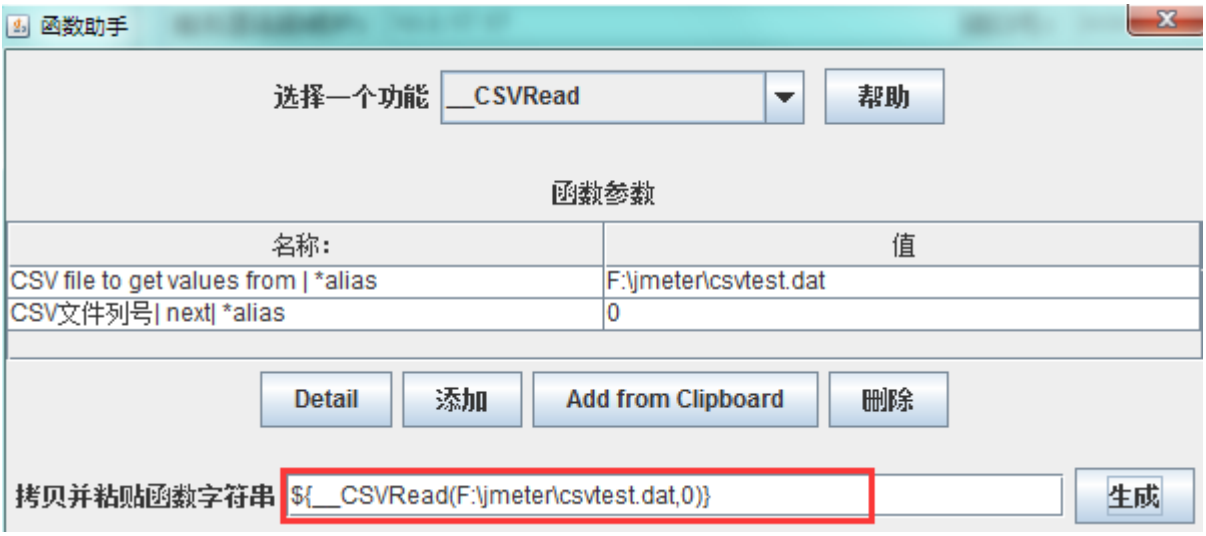


然后将保存的.dat文件放入计算机的某个盘里，这里我放入路径为：F:\jmeter\csvtest.dat

下面具体介绍参数化常用的的两种方法：

一、函数助手：_CSVRead

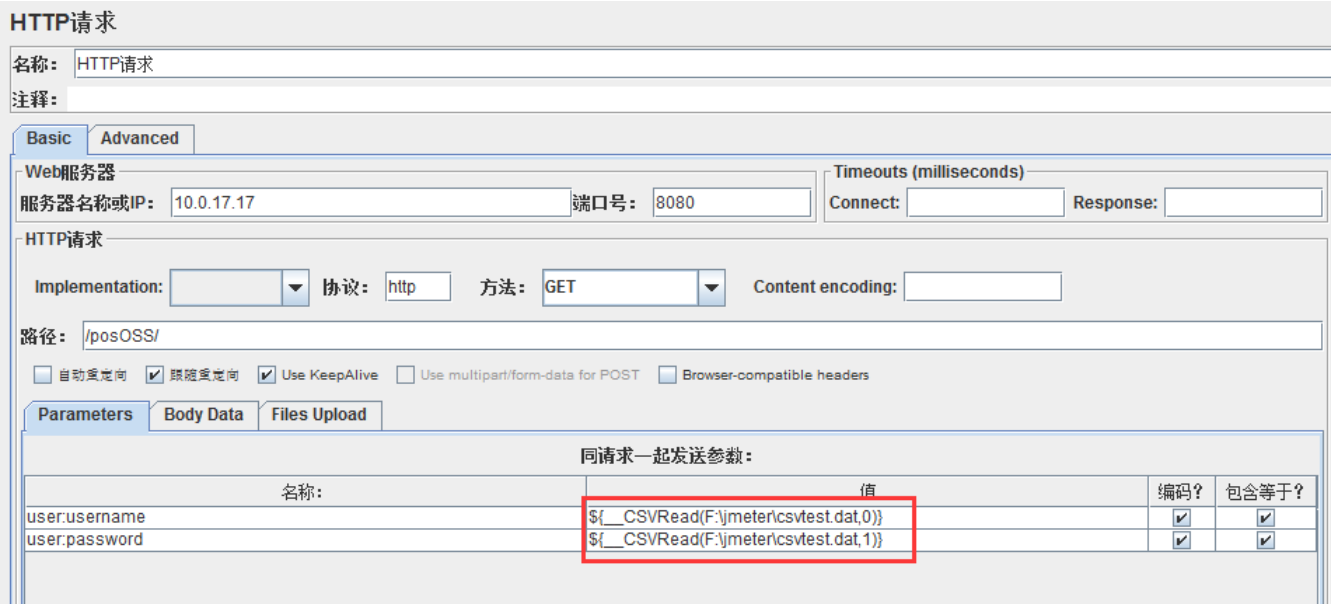
点击jmeter的界面，功能栏选项→ 函数助手对话框→ _CSVRead



CSV file to get values from | *alias：CSV文件取值路径，即这里需要写入之前的需要参数化的参数的文件路径

CSV文件列号| next|*alias：文件起始列号：CSV文件列号是从0开始的，第一列为0，第二列为1，以此类推。。。

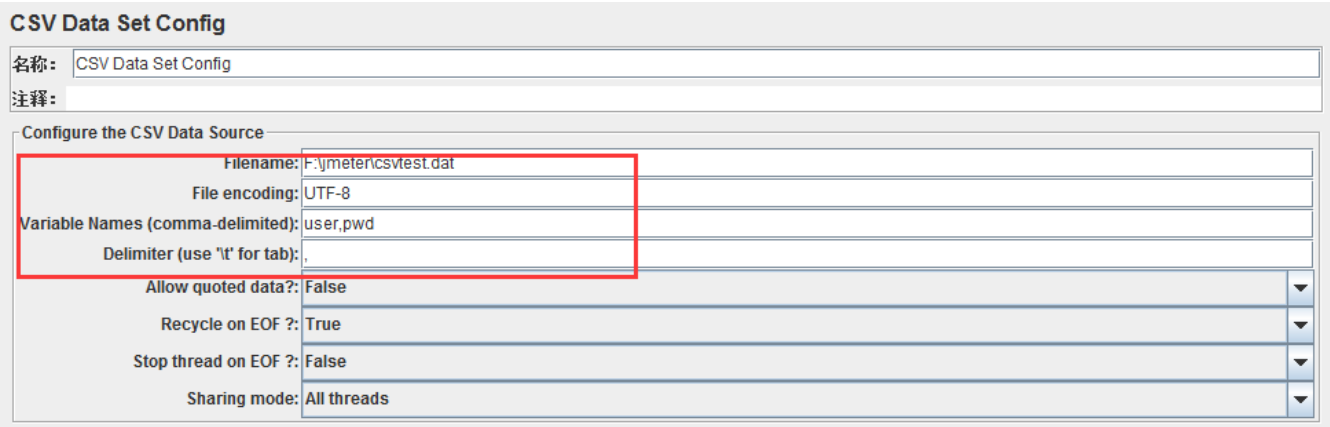
函数字符串：即生成的参数化后的参数，可以直接在登陆请求中的参数中引用，第一列为用户名，函数字段号为0，第二列为密码，函数字段号为1，以此类推进行修改使用即可



替换参数化后的参数，然后修改线程数，执行脚本，通过监听器里结果树的请求内容，可以看到请求的参数都是参数化后的数据

二、配置元件——CSV Data Set Config

点击线程组添加配置元件→ CSV Data Set Config：



说明：

Filename：F:\jmeter\csvtest.dat文件名，保存参数化数据的文件目录，可选择相对或者绝对路径（建议填写相对路径，避免脚本迁移时需要修改路径）；

File encoding:UTF-8，F:\jmeter\csvtest.dat文件的编码格式，在保存时保存编码格式为UTF-8即可;

Variable Names(comma-delimited)：对对应参数文件每列的变量名，类似excel文件的文件头，起到标示作用，同时也是后续引用的标识符，建议采用有意义的英文标示;

(如：有几列参数，在这里面就写几个参数名称，每个名称中间用分隔符分割，这里的 user,pwd，可以被利用变量名来引用：user,user,{pwd};

Delimitet：参数文件分隔符，用来在"Variable Names"中分隔参数，与参数文件中的分隔符保持一致即可;

Allow quote data：是否允许引用数据，默认false，选项选为"true"的时候对全角字符的处理出现乱码；

Recycle on EOF? ：是否循环读取参数文件内容；因为CSV Data Set Config一次读入一行，分割后存入若干变量中交给一个线程，如果线程数超过文本的记录行数，那么可以选择从头再次读入;

▵ Ture：为true时，当已读取完参数文件内的测试用例数据，还需继续获取用例数据时，此时会循环读取参数文件数据（即：读取文件到结尾时，再重头读取文件）；

▵False：为false时，若已至文件末尾，则不再继续读取测试数据；通常在"线程组线程数* 线程组循环次数 > 参数文件行数"时，选用false（即：读取文件到结尾时，停止读取文件）；

Stop thread on EOF?: 当Recycle on EOF为False时（读取文件到结尾），停止进程，当Recycle on EOF为True时，此项无意义;

▵若为ture，则在读取到参数文件行末尾时，终止参数文件读取线程;

▵若为false，此时线程继续读取，但会请求错误，因此时读取的数据为EOF;

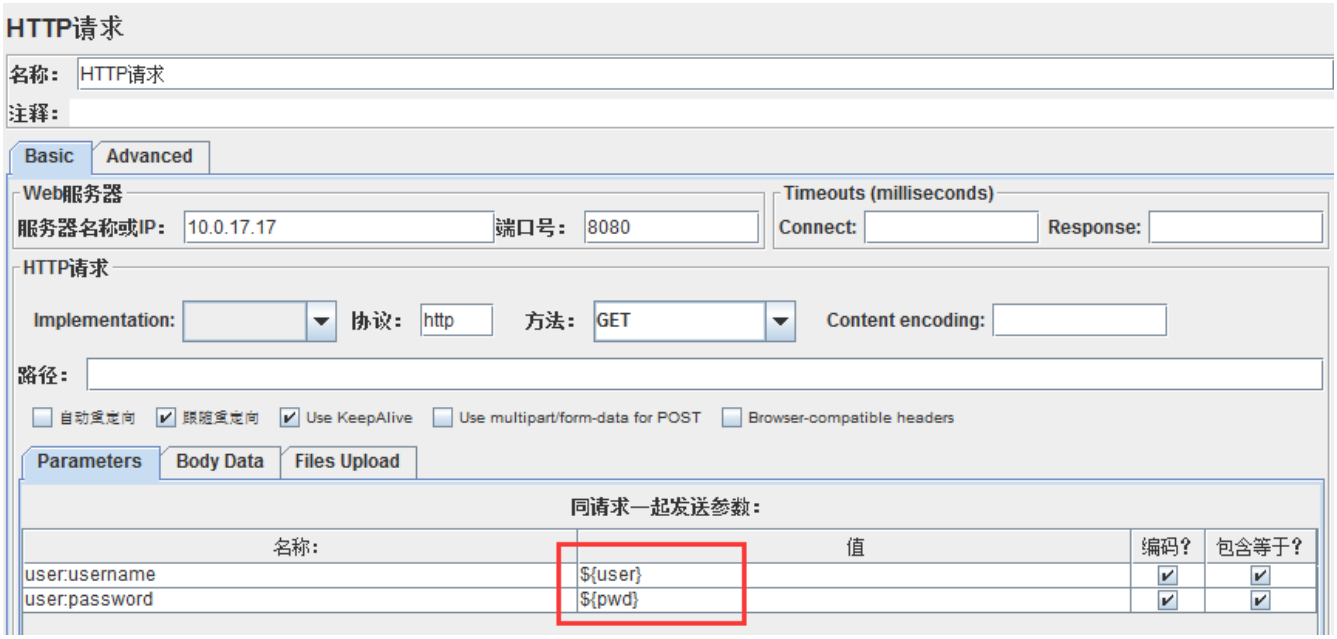
Sharing mode:共享模式，即参数文件的作用域，有以下几种方式:

▵All threads:当前测试计划中的所有线程中的所有的线程都有效，默认;

▵Current thread group:当前线程组中的线程有效;

▵Current thread:当前线程有效;

完成之后，将刚才生成的参数写入参数对应的值里面:



以上两种常见的参数化的方法，推荐使用CSV控件方法（因为函数助手参数化功能相比其较弱）

三、配置元件——User Defined Variables

点击线程组添加配置元件→ User Defined Variables（用户定义的变量）：

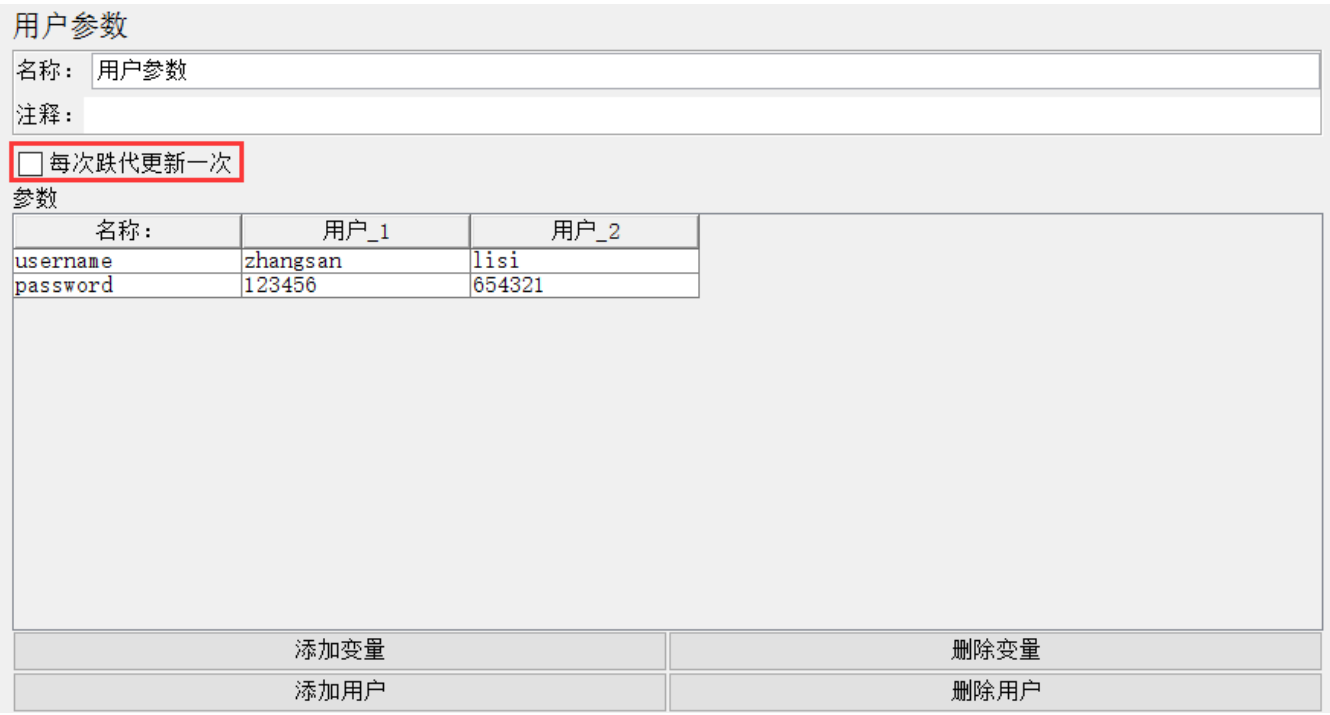
如上图所示，在该参数组中已经定义了两个参数，通过界面下方的添加、删除按钮可以向参数列表增加和删除参数，Up和Down可以上下移动参数的位置；

PS：User Defined Variables中定义的参数值在test plan执行过程中不能发生取值的改变，因此一般仅将test plan中不需要随迭代发生改变的参数（只取一次的参数）

设置在此处；例如：被测应用的host和port值。

四、前置处理器——User Variables

点击线程组添加前置处理器——User Variables（用户参数）：



如上图所示，在该参数组中已经设置了两个参数，username和password分别有2组不同的取值，通过页面下方的四个按钮，可以增加删除参数的可能取值。

PS：User Variables中设置的参数可以在test plan执行过程中发生变化。

以上就是jmeter参数化的四种方式，其中：

1、函数助手_CSVRead的参数化功能相比CSV Data Set Config较弱；

2、CSV Data Set Config适用于参数取值范围较大的时候使用，该方法具有更大的灵活性；

3、User Defined Variables一般用于test plan中不需要随请求迭代的参数设置；

4、User Variables适用于参数取值范围很小的时候使用；

PS：相比于loadrunner来说，jmeter参数化有以下不同：

1.jmeter参数文件第一行没有列名称

2.参数文件的编码，尽量保存为UTF-8（编码问题在使用CSV Data Set Config参数化时要求的比较严格）

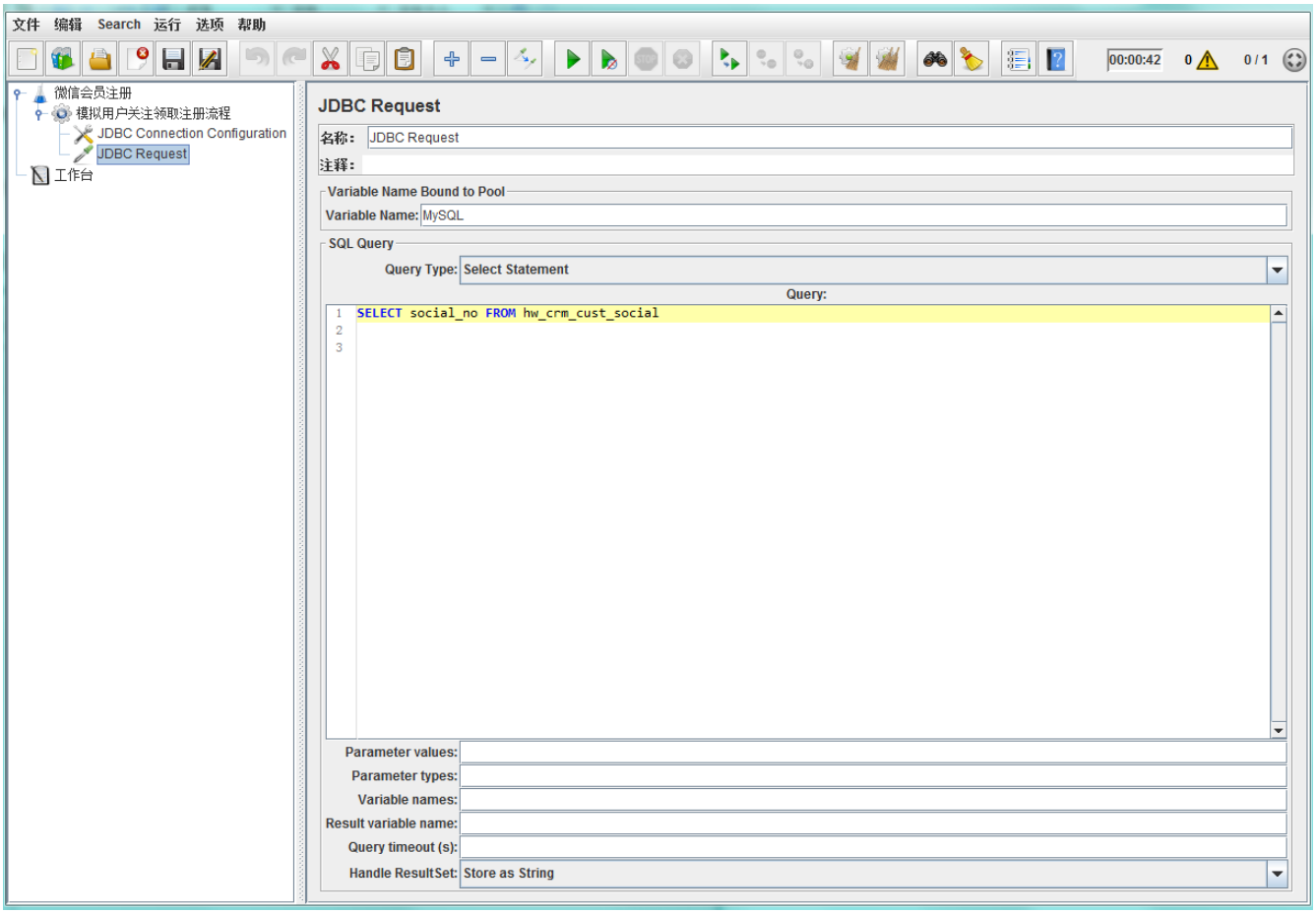
3.Jmeter的参数化没有LoadRunner做的出色，它是依赖于线程设置的（只有CSV Data Set Config参数化方法才有）

（11）关联之正则表达式提取器

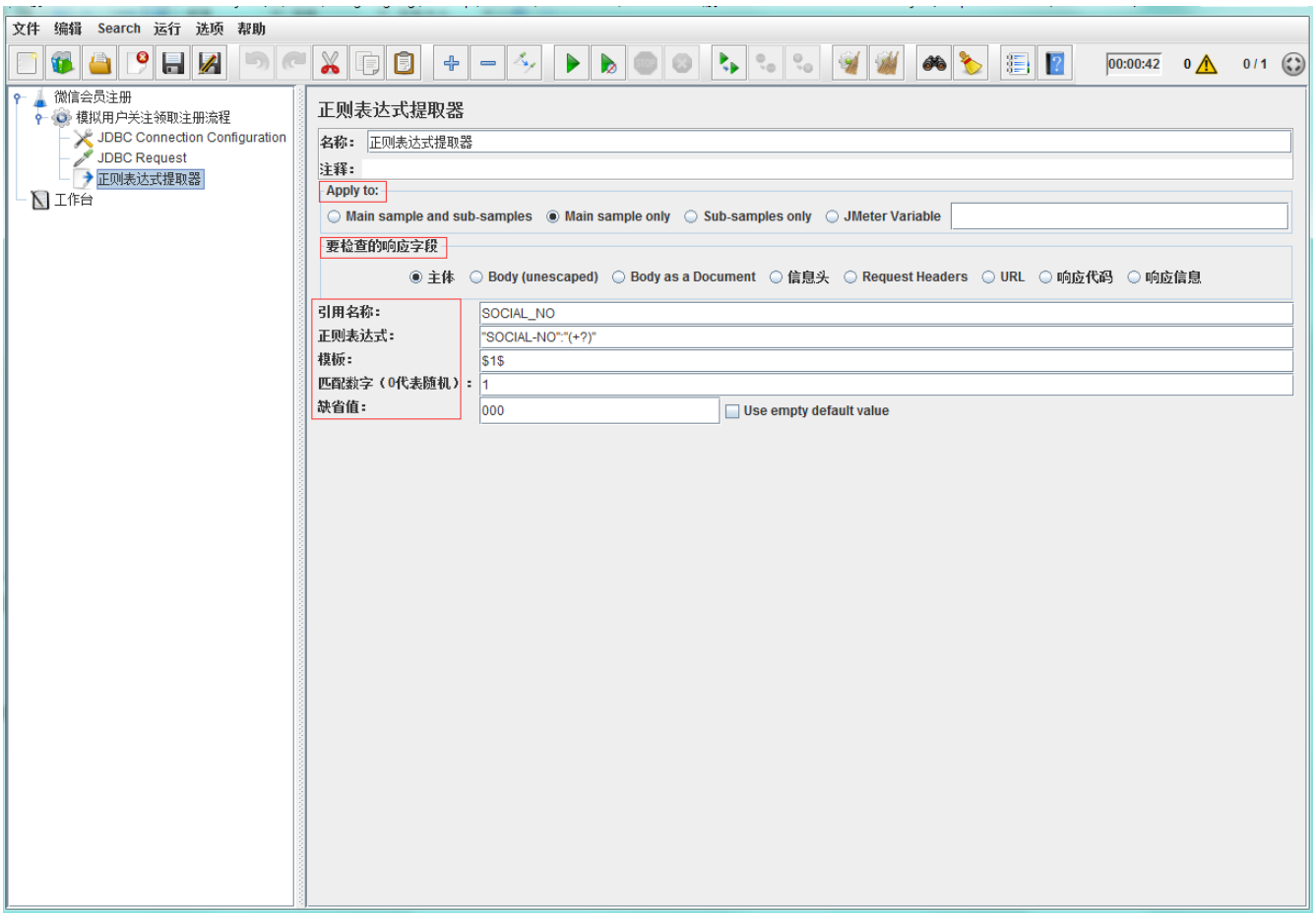
如果有这样的情况：一个完整的操作流程，需要先完成某个操作，获得某个值或数据信息，然后才能进行下一步的操作（也就是常说的关联/将上一个请求的响应结果作为下一个请求的参数）；

在jmeter中，可以利用正则表达式提取器来帮助我们完成这一动作。

首先：在默认的测试计划中添加一个线程组，然后添加取样器，这里我以JDBC请求做例子：



然后：右键添加后置处理器→正则表达式提取器，正则表达式提取器界面如下：



说明：

后置处理器：在请求结束或者返回响应结果时发挥作用

正则表达式提取器：允许用户从服务器的响应中通过使用perl的正则表达式提取值。该元素会作用在指定范围取样器，用正则表达式提取所需值，生成模板字符串，并将结果存储到给定的变量名中。

APPly to:作用范围（返回内容的断言范围）

Main sample and sub-samples:作用于父节点的取样器及对应子节点的取样器

Main sample only：仅作用于父节点的取样器

Sub-samples only:仅作用于子节点的取样器

JMeter Variable:作用于jmeter变量(输入框内可输入jmeter的变量名称)

要检查的响应字段：需要检查的响应报文的范围

主体：响应报文的主体

Body(unescaped):主体，响应的主体内容且替换了所有的html转义符，注意html转义符处理时不考虑上下文，因此可能有不正确的转换，不太建议使用

Body as a Document：从不同类型的文件中提取文本，注意这个选项比较影响性能

Response Headers：响应信息头

Request Headers:请求信息头

URL：统一资源定位符，即Internet上用来描述信息资源的字符串

Response Code:响应状态码，比如200、404等

Response Message:响应信息

PS：jmeter的中文翻译有时候不太准确，建议尽量选择语言格式为英文（为了方便说明，这里选择中文语言，当然，自己明白最好，不用纠结这个）

引用名称（Reference Name）：Jmeter变量的名称，存储提取的结果；即下个请求需要引用的值、字段、变量名（例子中我提取的是SOCIAL_NO）

引用方法：引用方法:${引用名称}

正则表达式（Regular Expression）：使用正则表达式解析响应结果，**“（）”表示提取字符串中的部分值**，请不要使用“||”，除非你本身需要匹配这个字符。

下面是常用的正则表达式操作符：

操作符	含义
.	匹配任何单个字符
?	出现在该符号之前的项目是可选的，最多匹配一次（匹配0或1次）
*	匹配出现0次或多次的项目
+	匹配1次或多次先前项目
{N}	精确匹配N次的先前项目
{N,}	先前的项目匹配N或者更多次
{N,M}	先前的项目匹配至少N次，但不多于M此
^	匹配行开始的空字符串；也表示不在列表范围内的字符
$	匹配行末的空字符串
\b	匹配词两边的空字符串
\w	匹配字母、数字、下划线或汉字
\<	匹配任何词开头的空字符串
\>	匹配任何词结尾的空字符串

模板（Template）：从匹配的结果中创建一个字符串，这是通过正则表达式匹配出来的一组值，意为使用提取到的第几个值（可能有多个值匹配，因此使用模板）；从1开始匹配，以此类推.

参数可以在取值模板组合使用，例如："11-22"作为模板得到的值是使用"-"连接的第一个待匹配内容与第二个待匹配内容组合而成的字符串。

匹配数字（Match No）：正则表达式匹配数据的结果可以看做一个数组，表示如何取值：0代表随机取值，正数n则表示取第n个值（比如1代表取第一个值），负数则表示提取所有符合条件的值。

缺省值：匹配失败时候的默认值；通常用于后续的逻辑判断，一般通常为特定含义的英文大写组合，比如：ERROR

最后，根据上面的说明，完成配置，然后可以先添加一个监视器（查看结果树），检查是否取到了对应的值；

提取到的参数，调用时用${SOCIAL_NO_1}，${SOCIAL_NO_2}...，如果想要得到匹配出的参数的个数，用${SOCIAL_NO_matchNr}，如果想随机选取一个，只需要将

匹配数字设为0，使用${SOCIAL_NO}调用即可。

同类型博客推荐：http://www.cnblogs.com/wuyepiaoxue/p/5661194.html

http://blog.csdn.net/meami/article/details/50495148

（12）关联之XPath Extractor

之前的博客，有介绍jmeter如何对请求进行关联的一种常见用法，即：后置处理器中的正则表达式提取器，下面介绍另一种关联方法，XPath Extractor！

所谓**关联**，从业务角度讲，即：某些操作步骤与其相邻步骤存在一定的依赖关系，导致某个步骤的输入数据来源于上一步的返回数据，这时就需要"关联"来建立步骤之间的联系。

简单来说，就是：将上一个请求的响应结果作为下一个请求的参数。。。

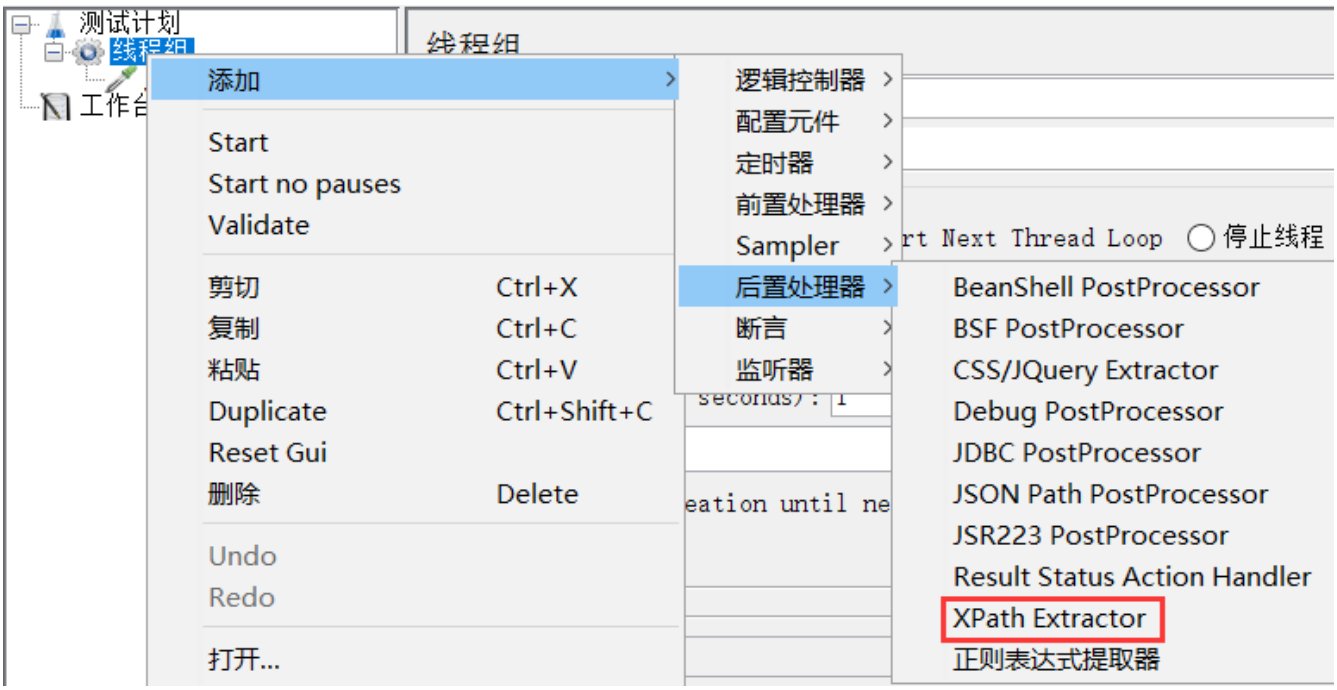
jmeter提供的对关联的支持包括以下2个方面：

①能够将返回页面上的指定内容保存在参数中；

②能够将GET或POST方法中的数据使用该参数来替换；

XPath Extractor的使用方法与正则表达式提取器(Regular Expression Extractor)类似，只不过该Expression中指定的不是正则表达式，而是给定的XPath路径。

首先，新建一个线程组，然后右键-添加-后置处理器-XPath Extractor：

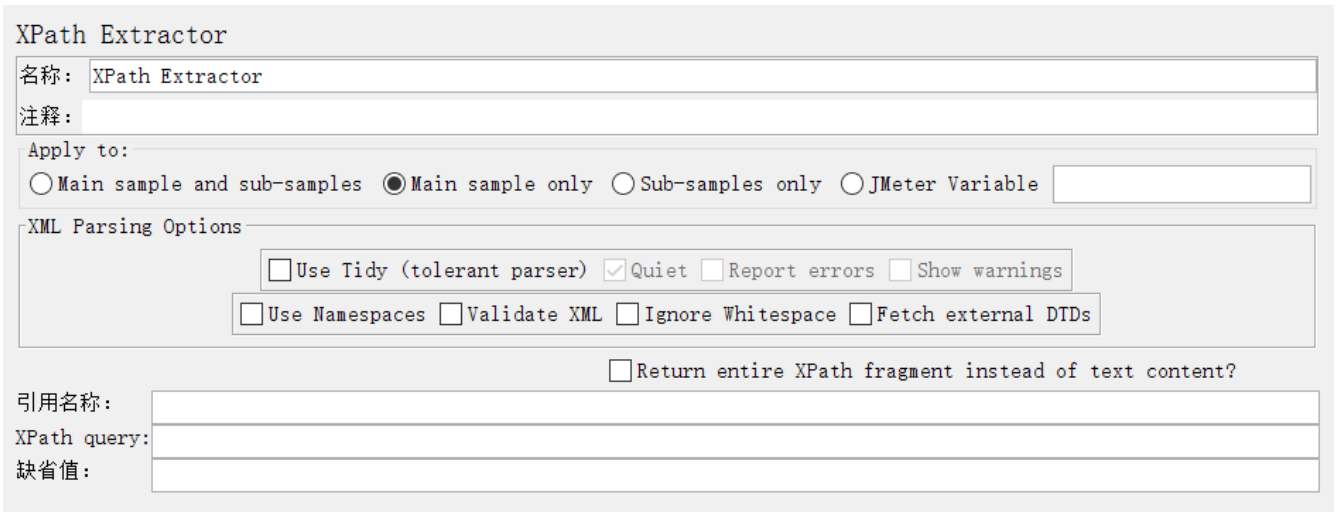


这里简单介绍下**jmeter后置处理器的作用**：

后置处理器(Post Processor)本质上是一种对sampler发出请求后接受到的响应数据进行处理（后处理）的方法，结合之前我介绍过的jmeter元件的作用域和执行顺序，

必须将后置处理器元件放在合适的位置才能达到预期的效果。

XPath Extractor界面如下：



APPly to:作用范围（返回内容的断言范围）

Main sample and sub-samples:作用于父节点的取样器及对应子节点的取样器

Main sample only：仅作用于父节点的取样器

Sub-samples only:仅作用于子节点的取样器

JMeter Variable:作用于jmeter变量(输入框内可输入jmeter的变量名称)

XML Parsing Options：要解析的XML参数

Use Tidy：当需要处理的页面是HTML格式时，必须选中该选项；如果是XML或XHTML格式（例如RSS返回），则取消选中；

Quiet表示只显示需要的HTML页面，Report errors表示显示响应报错，Show warnings表示显示警告；

Use Namespaces：如果启用该选项，后续的XML解析器将使用命名空间来分辨；

Validate XML：根据页面元素模式进行检查解析；

Ignore Whitespace：忽略空白内容；

Fetch external DTDs：如果选中该项，外部将使用DTD规则来获取页面内容；

Return entire XPath fragment of text content：返回文本内容的整个XPath片段；

Reference Name：存放提取出的值的参数。

XPath Query：用于提取值的XPath表达式。

Default Value：参数的默认值。

PS:XPath是XML/XHTML中常用的选取给定节点和节点集的方法。

正则表达式提取器和XPath Extractor的区别：

①正则表达式提取器可以用于对页面任何文本的提取，提取的内容是根据正则表达式在页面内容中进行文本匹配；

②XPath Extractor则可以提取返回页面任意元素的任意属性；

③如果需要提取的文本是页面上某元素的属性值，建议使用XPath Extractor;

④如果需要提取的文本在页面上的位置不固定，或者不是元素的属性，建议使用正则表达式提取器。

（13）配置元件之计数器

先说说利用jmeter生成数据的几种方法：

1、CSV Data Set Config

这个元件被用来在参数化生成数据时使用，简单高效，容易生成有序数；只需要新建excel，然后通过拖拽、复制黏贴等方式产生不同的数据，然后读取调用即可。

但它也有不足之处，如下：

①如果数据库中某些表的某些字段不允许重复（比如订单号），那么在完成一轮测试后，再次测试需要重新手动构造新的不重复的数据；

②excel只有数字格式才可以通过拖拽生成增长数据；

③数据量过大时，容易被excel修改为科学计数法；

PS：关于该元件以及参数化，请参考之前的博客：http://www.cnblogs.com/imyalost/p/6229355.html

2、${_Random}

${_Random}是jmeter函数助手里面自带的一个函数，作用是返回指定的最大值和最小值之间的一个随机数。

缺点：数值可能会重复出现；

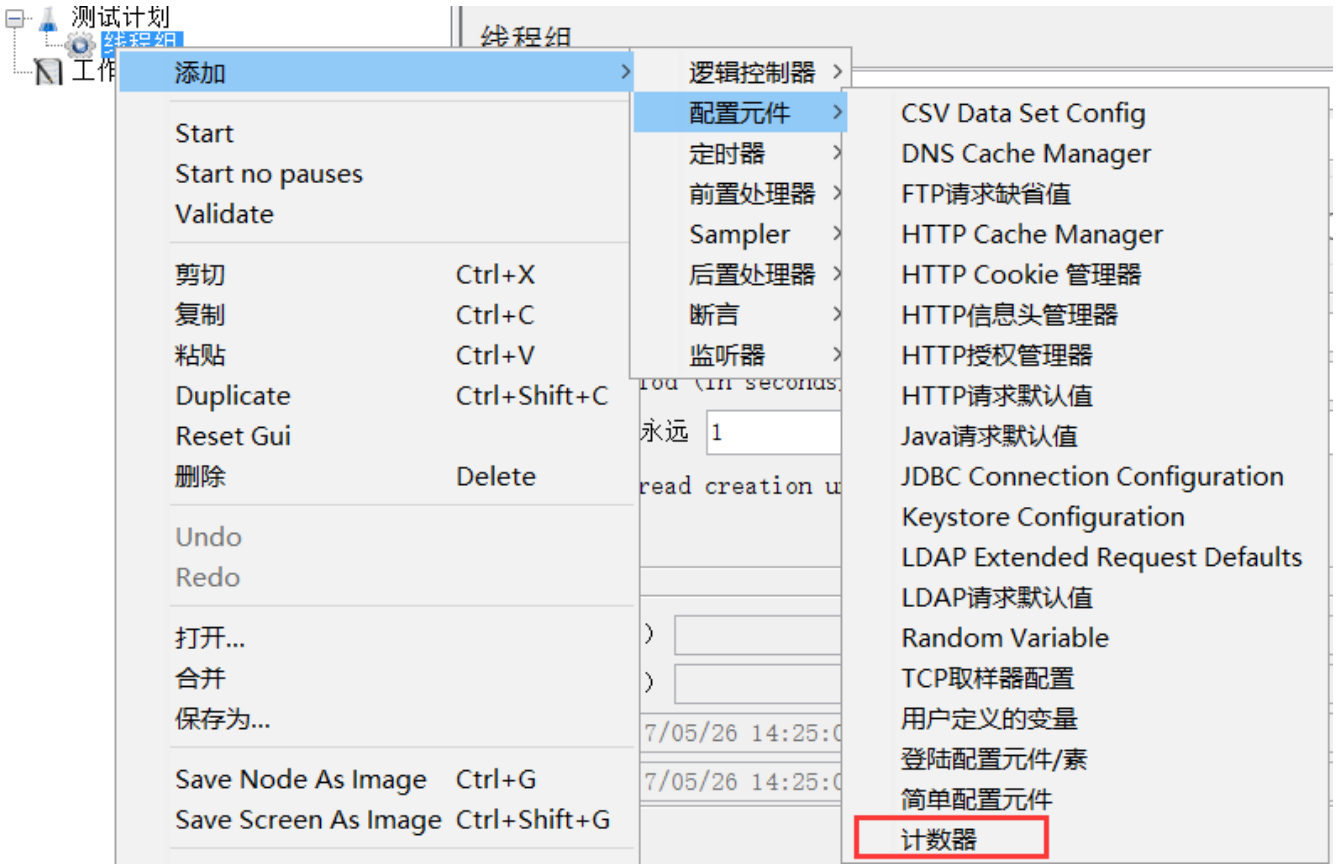
PS：关于jmeter函数助手，请参考之前的博客：http://www.cnblogs.com/imyalost/p/6802173.html

如果需要引用的数据量较大，且要求不能重复或者需要自增，那么可以使用计数器来实现。

计数器（counter）：允许用户创建一个在线程组之内都可以被引用的计数器。

计数器允许用户配置一个起点,一个最大值,增量数，循环到最大值,然后重新开始,继续这样,直到测试结束。计数器使用long存储的值,所取的范围是2^63——2^63-1.

1、启动jmeter，添加线程组，右键添加配置元件——计数器



计数器界面如下：

启动（start）：给定计数器的起始值、初始值，第一次迭代时，会把该值赋给计数器

PS：英文版是Start，Jmeter的中文语言将Start翻译成了“启动”，有些歧义

递增(Increment)：每次迭代后，给计数器增加的值

最大值(Maximum)：计数器的最大值，如果超过最大值，重新设置为初始值(Start)，默认的最大值为Long.MAX_VALUE,2^63-1 **（如果持续压测，建议最好不要设置最大值）**

Number format：可选格式，比如000，格式化为001，002；默认格式为Long.toString()，但是默认格式下，还是可以当作数字使用

引用名称(Reference Name)：用于控制在其它元素中引用该值，形式：$(reference_name}

与每用户独立的跟踪计数器(Track Counter Independently for each User)：全局的计数器，如果不勾选，即全局的，比如用户#1 获取值为1，用户#2获取值还是为1；

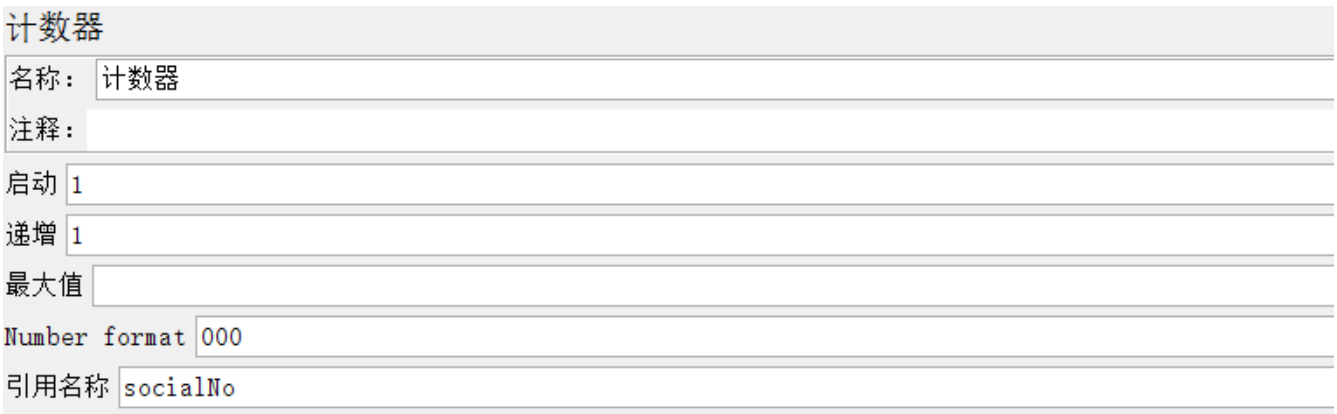
如果勾选，即独立的，则每个用户有自己的值：比如用户#1 获取值为1，用户#2获取值为2。

每次迭代复原计数器（Reset counter on each Thread Group Iteration）：可选，仅勾选与每用户独立的跟踪计数器时可用；

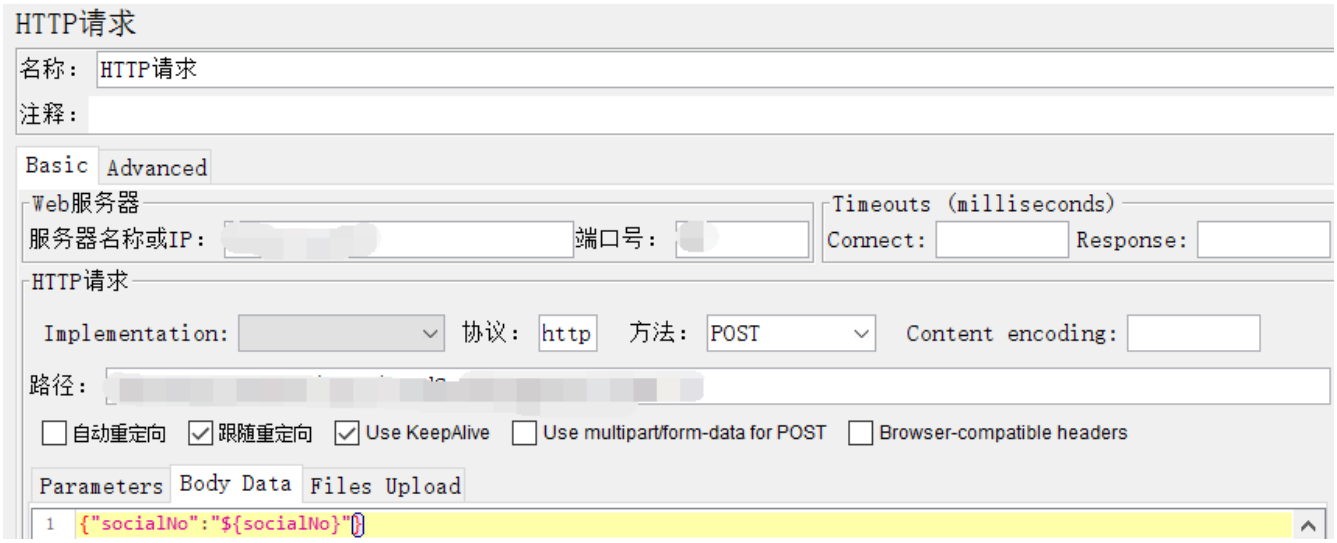
如果勾选，则每次线程组迭代，都会重置计数器的值，当线程组是在一个循环控制器内时比较有用。

2、具体过程

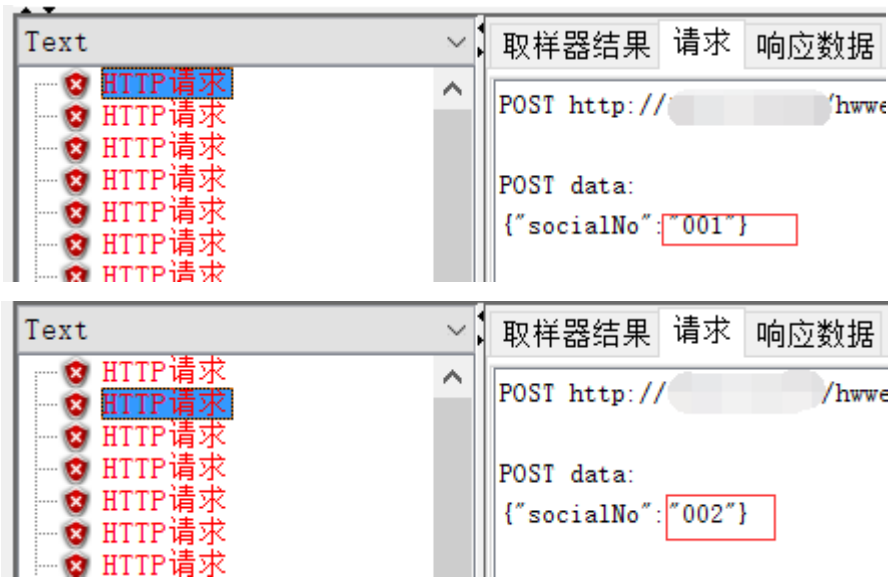
①计数器设置



②取样器设置



③结果树请求内容



从上图可以看出，计数器成功的生成了我们所需的值。

PS：以上就是计数器的使用方法；为了方便演示，请求地址和路径是随便选用的，忽略红色报错即可。。。

（14）配置元件之HTTP属性管理器

jmeter是一个开源灵活的接口和性能测试工具，当然也能利用jmeter进行接口自动化测试。在我们利用它进行测试过程中，最常用的sampler大概就是Http Request，

使用这个sampler时，一般都需要使用配置元件里的http属性管理器，其作用就是用于尽可能的模拟浏览器的行为，在http协议层上定制发送给被测应用的http请求。

jmeter提供以下五种http属性管理器：

HTTP Cache Manager：Cache管理器

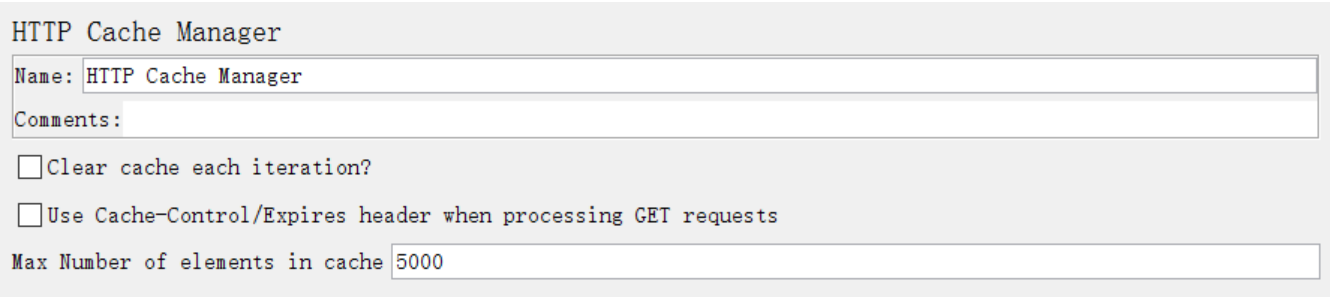
HTTP Cookie Manager：cookie管理器

HTTP Header Manager：信息头管理器

HTTP Authorzation Manager：授权管理器

HTTP Request Defaults：请求默认值

1、HTTP Cache Manager



Clear cache each iteration?（每次迭代清空缓存）:如果选择该项，则该属性管理器下的所有Sampler每次执行时都会清除缓存；

Use Cache-Control/Expires header when processing GET requests:在处理GET请求时使用缓存/过期信息头；

Max Number of elements in cache（缓存中的最大元素数）:默认数值为5000，当然可以根据需要自行修改；

PS：如果Test Plan中某个Sampler请求的元素是被缓存的元素，则Test Plan在运行过程中会直接从Cache中读取元素，这样得到的返回值就会是空。

在这种情况下，如果为该Sampler设置了断言检查响应体中的指定内容是否存在，该断言就会失败！

为test plan增加该属性管理器后，test plan运行过程中会使用Last-Modified、ETag和Expired等决定是否从Cache中获取对应元素。

Cache：一般指的是浏览器的缓存

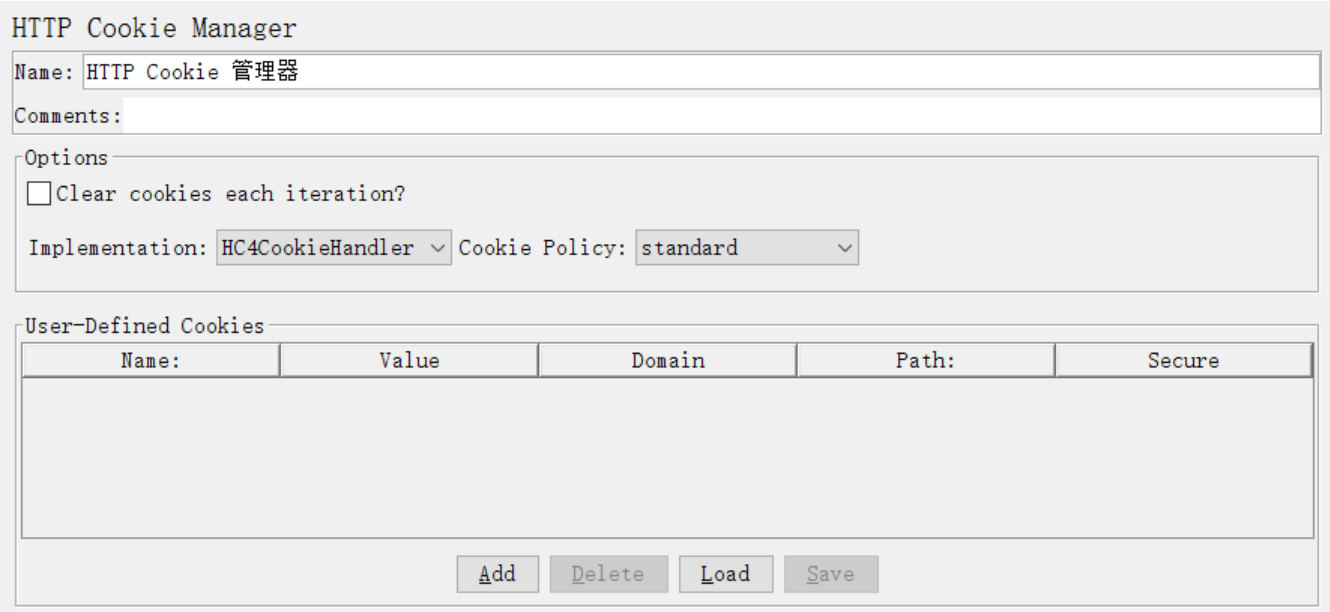
Last-Modified：文件在服务端最后被修改的时间

ETag：在HTTP协议规格说明中定义为：被请求变量的实体标记

Expired：给出的日期/时间之后；一般结合Last-Modified一起使用，用于控制请求文件的有效时间

PS：上面提到的几个字段，都是HTTP协议里面的报文首部的字段，感兴趣的请自行查阅相关内容，或可参考这篇博客：浏览器缓存详解

2、HTTP Cookie Manager



Clear cookie each iteration?（每次迭代时清除自己会话区域的所有cookie）；

Implementation：实现方式；

Cookie Policy：cookie的管理策略，建议选择compatibility,兼容性强；

PS：对于JMeter来说，一个test plan只能有一个cookie管理器。因为当多个magager存在时，JMeter没有方法来指定使用那个manager；

同时，一个cookie manager中的存储的cookie也不能被其他cookie manager所引用，所以同一个计划中不建议使用多个cookie manager；

如果你想让JMeter的cookie manager支持跨域， 修改JMeter.property :CookieManager.check.cookies=false；

HTTP cookie Manager管理cookie有两种方法：

①、它可以像浏览器一样存储和发送cookie，如果发送一个带cookie的http请求，cookie manager会自动存储该请求的cookies，并且后面如果发送同源站点的http请求时，

都可以用这个cookies；每个线程都有自己的“cookie存储区域”，所以当测试一个使用cookie来管理session信息的web站点时，每个JMeter线程都有自己的session；

PS：以这种自动收集的方式收集到的cookie不会在cookie manager中进行展示，但是运行后通过查看结果树可以查看到cookie信息，接受到的cookie会被自动存储在线程变量中，

但是从Jmeter2.3.2版本后，默认不再存储，如果你想要manager自动存储收集到 的cookie，你需要修改JMeter.property:CookieManager.save.cookies=true；

存储的时候，cookie的key会以“COOKIE_”为前缀命名（默认情况），如果你想自定义这个前缀，修改JMeter.property:CookieManager.name.prefix= ；

②、除了上面说的自动收集，还可以手动添加cookie，点击界面下方的Add按钮，然后输入cookie的相关信息；

PS：一般不建议手动添加，可以将cookie通过浏览器插件（比如Firefox的firebug）导出cookie，然后点击界面下方的load按钮，载入刚才导出的cookie文件即可。

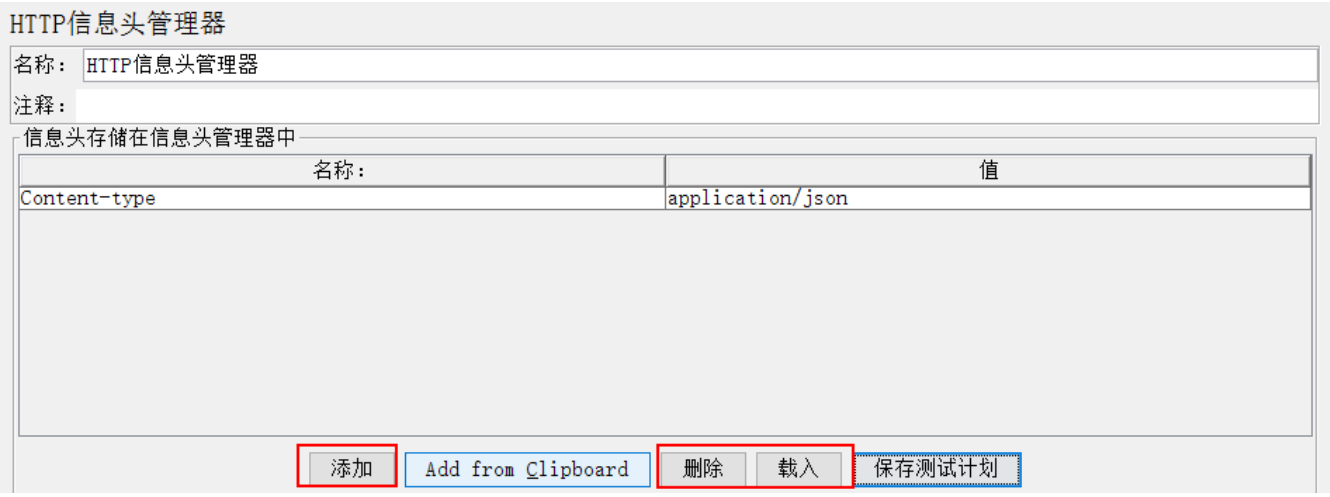
关于Cookie：

cookie一般分为2种：**持久cookie**（Permanent cookie）和**会话cookie**（Session cookie）：

持久cookie：保存在客户端本地硬盘上，在浏览器被关闭后仍然存在；

会话cookie：通常保存在浏览器进程的会话中，一旦浏览器会话结束或关闭，cookie就不再存在。

3、HTTP Header Manager



通常Jmeter向服务器发送http请求的时候，后端需要一些验证信息，比如说web服务器需要带过去cookie给服务器进行验证，一般就是放在请求头（header）中，或者请求传参

需要定义参数格式等；因此对于此类请求，在Jmeter中可以通过HTTP信息头管理器，在添加http请求之前，添加一个HTTP信息头管理器，发请求头中的数据通过键值对的形式放到

HTTP信息头管理器中，在往后端请求的时候就可以模拟web携带header信息。

PS：可以点击添加、删除按钮等来新增和删减信息头的数据，也可通过载入按钮来将信息头数据加载进去（信息头数据较多时推荐使用）。

4、HTTP Authorzation Manager

该属性管理器用于设置自动对一些需要验证的页面进行验证和登录；

基础URL：需要使用认证页面的基础URL，如上图，当取样器访问它时，jmeter会使用定义的username和password进行认证和登录；

用户名：用于认证和登录的用户名；

密码：用于认证和登录的口令；

域：身份认证页面的域名；

Realm：Realm字串；

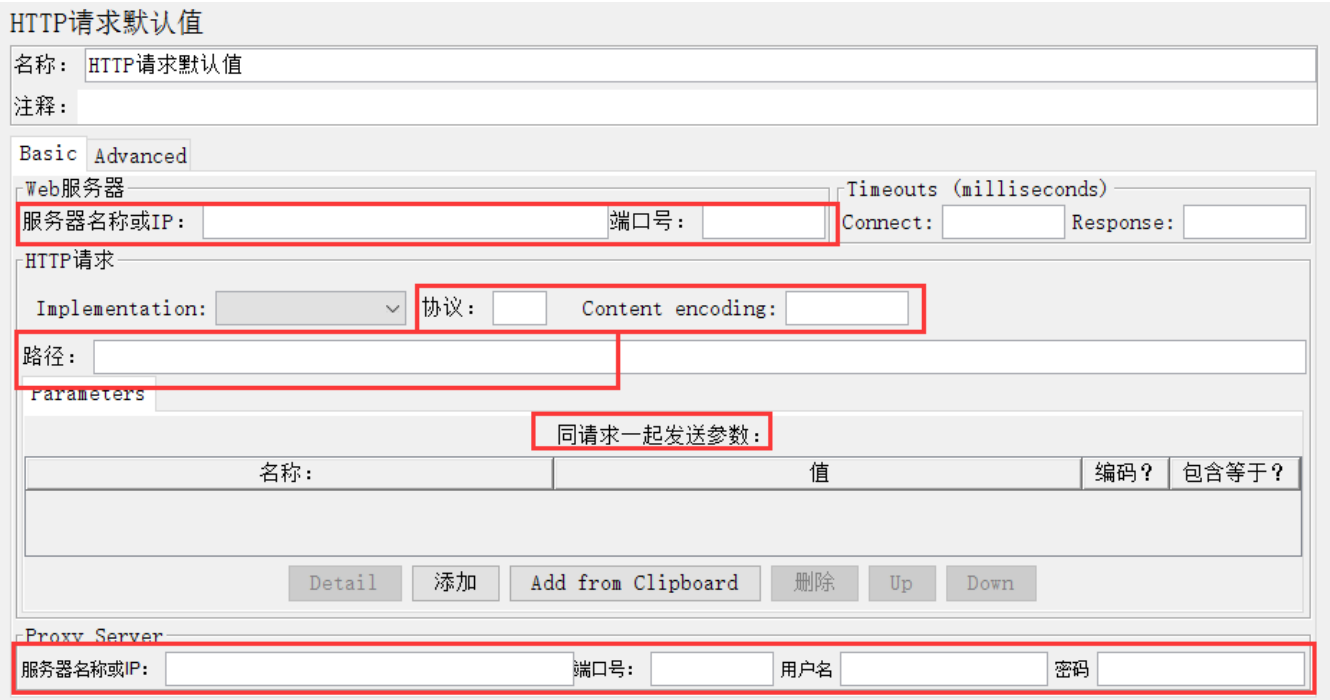
Mechanism：机制；jmeter的http授权管理器目前提供2种认证机制：BASIC_DIGEST和KERBEROS：

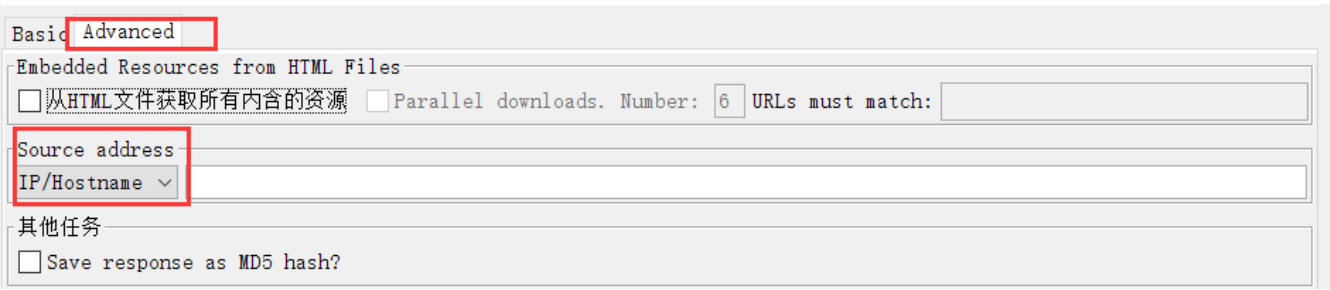
BASIC_DIGEST：HTTP协议并没有定义相关的安全认证方面的标准，而BASIC_DIGEST是一套基于http服务端的认证机制，保护相关资源避免被非法用户访问，

如果你要访问被保护的资源，则必需要提供合法的用户名和密码。它和HTTPS没有任何关系（前者为用户认证机制，后者为信息通道加密）。

KERBEROS：一个基于共享秘钥对称加密的安全网络认证系统，其避免了密码在网上传输，将密码作为对称加密的秘钥，通过能否解密来验证用户身份；

5、HTTP Request Defaults





HTTP请求默认值，这个属性管理器用于设置其作用范围内的所有HTTP Request默认值，包括：

服务器请求或IP：请求发送的目标服务器名称或地址；

端口：目标服务器的端口号，默认80；

协议：箱目标服务器发送请求所采用的协议，HTTP或HTTPS，默认HTTP；

Content encoding：内容的编码方式，默认值为iso8859；

路径：目标URL路径（不包括服务器地址和端口）；

同请求一起发送参数： 对于带参数的URL，jmeter提供了一个简单的对参数化的方法：用户可以将URL中所有参数设置在本表中，表中的每一行是一个参数值对；

从HTML文件获取所有有内含的资源：该选项被选中时，jmeter在发出HTTP请求并获得响应的HTML文件内容后，还对该HTML进行Parse 并获取HTML中包含的

所有资源（图片、flash等），默认不选中；如果用户只希望获取页面中的特定资源，可以在下方的Embedded URLs must match 文本框中填入需要下载的特定资源表达式，

这样，只有能匹配指定正则表达式的URL指向资源会被下载。

注意事项：

①、一个测试计划中可以有多个Defaults组件，多个Defaults组件的默认值会叠加；

②、两个default中都定义的"Server Name or IP"，显示在发送请求时只能使用一个；

参考博客：http://www.cnblogs.com/puresoul/p/4853276.html

http://blog.chinaunix.net/uid-29578485-id-5604160.html

（15）函数助手

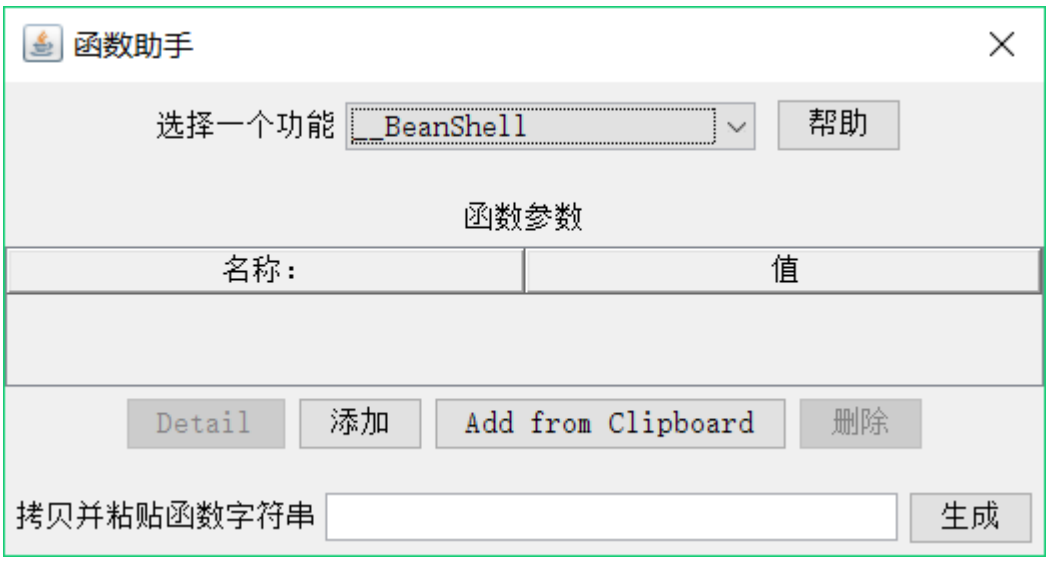
jmeter作为一个开源的性能测试工具，作用还是蛮强大的，找到一篇对jmeter中函数助手解释蛮详细的一篇博客，感觉不错，转载过来，希望对大家有所帮助。

由于时间和版本问题，其中有些内容和排版我做了修改和重新整理，使其更符合最新的jmeter版本。

原文地址：http://blog.csdn.net/fanjeff/article/details/46873159

一、使用jmeter函数助手

启动jmeter后，可以在JMeter的选项菜单中找到函数助手对话框（快捷键：Ctrl+Shift+F1），如下图所示：



打开函数助手，可以从下拉列表中选择一个函数，并为其参数设定值，不同函数要求的参数也不同；表格的左边一列是函数参数的简要描述，右边一列是供用户填充参数的值。

二、常用JMeter函数

1、__regexFunction

正则表达式函数可以使用正则表达式（用户提供的）来解析前面的服务器响应（或者是某个变量值），函数会返回一个有模板的字符串，其中携带有可变的值。

__regexFunction还可以被用来保存值，以便供后续使用。

在函数的第6个参数中，可以指定一个引用名；在函数执行以后，可以使用用户定义值的语法来获取同样的值。例如，如果输入"refName"作为第6个参数，那么可以使用：

${refName}来引用第2个参数（Template for the replacement string）的计算结果，这依赖于函数的解析结果；

${refName_g0}来引用函数解析后发现的所有匹配结果；

${refName_g1}来引用函数解析后发现的第一个匹配组合；

${refName_g#}来引用函数解析后发现的第n个匹配组合；

${refName_matchNr}来引用函数总共发现的匹配组合数目；

参数如下表所示：

函数参数	描述	是否必需
第1个参数	第1个参数是用于解析服务器响应数据的正则表达式，它会找到所有匹配项；如果希望将表达式中的某部分应用在模板字符串中，一定记得为其加上圆括号。例如，，这样就会将链接的值存放到第一个匹配组合中（这里只有一个匹配组合）。又如，<input type="hidden"name="(.)"value="(.)">，在这个例子中，链接的name作为第一个匹配组合，链接的value会作为第二个匹配组合，这些组合可以用在测试人员的模板字符串中。	是
第2个参数	这是一个模板字符串，函数会动态填写字符串的部分内容。要在字符串中引用正则表达式捕获的匹配组合，请使用语法：[groupnumber][groupnumber]。例如11或者 22，模板可以是任何字符串。	是
第3个参数	第3个参数告诉JMeter使用第几次匹配；测试人员的正则表达式可能会找到多个匹配项，对此，有4种选择：n 整数，直接告诉JMeter使用第几个匹配项；n “1”对应第一个匹配，“2”对应第二个匹配，以此类推；n RAND，告诉JMeter随机选择一个匹配项；n ALL，告诉JMeter使用所有匹配项，为每个匹配项创建一个模板字符串，并将它们连接在一起n 浮点值0到1之间，根据公式（找到的总匹配数目*指定浮点值）计算使用第几个匹配项，计算值 向最近的整数取整	否，默认值为1
第4个参数	如果在上一个参数中选择了“ALL”，那么这第4个参数会被插入到重复的模板值之间	否
第5个参数	如果没有找到匹配项返回的默认值	否
第6个参数	重用函数解析值的引用名，参见上面内容	否
第7个参数	输入变量名称。如果指定了这一参数，那么该变量的值就会作为函数的输入，而不再使用前面的采样结果作为搜索对象	否

2、__counter

每次调用计数器函数都会产生一个新值，从1开始每次加1。计数器既可以被配置成针对每个虚拟用户是独立的，也可以被配置成所有虚拟用户公用的。如果每个虚拟用户的计数器

是独立增长的，那么通常被用于记录测试计划运行了多少遍。全局计数器通常被用于记录发送了多少次请求，计数器使用一个整数值来记录，允许的最大值为2,147,483,647。

目前计数器函数实例是独立实现的（JMeter 2.1.1及其以前版本，使用一个固定的线程变量来跟踪每个用户的计数器，因此多个计数器函数会操作同一个值）。

全局计数器（FALSE）每个计数器实例都是独立维护的。

参数如下表所示：

函数参数	描述	是否必需
第1个参数	True，如果希望每个虚拟用户的计数器保持独立，与其他用户的计数器相区别；false，全局计数器；	是
第2个参数	重用计数器函数创建值的引用名。可以这样引用计数器的值：${refName}。这样一来，就可以创建一个计数器后，在多个地方引用它的值（JMeter 2.1.1及其以前版本，这个参数是必需的）	否

3、__threadNum

函数__threadNum只是简单地返回当前线程的编号。线程编号不依赖于线程组，这就意味着从函数的角度看来，某个线程组的线程#1和另一个线程组的线程#1是没有区别的。

另外，该函数没有参数。这一函数不能用在任何配置元件中（如用户定义的变量），原因在于配置元件是由一个独立线程运行的。另外在测试计划中使用也是没有意义的。

4）__intSum

函数__intSum可以被用来计算两个或者更多整数值的合。

参数如下表所示：

函数参数	描述	是否必需
第1个参数	第1个整数值	是
第2个参数	第2个整数值	是
第*n*个参数	第*n*个整数值	否
最后一个参数	重用函数计算值的引用名。如果用户指定了这一参数，那么引用名中必须包含一个非数字字母，否则它会被当成另一个整数值，而被函数用于计算	否

JMeter 2.3.1及其以前版本，要求必须有引用名参数。后续JMeter版本中，引用名是可选的参数，但是引用名不能是整数值。

5、__longSum

函数__ longSum可以被用来计算两个或者更多长整型值的合。

参数如下表所示：

函数参数	描述	是否必需
第1个参数	第1个长整型值	是
第2个参数	第2个长整型值	是
第*n*个参数	第*n*个长整型值	否
最后一个参数	重用函数计算值的引用名。如果用户指定了这一参数，那么引用名中必须包含一个非数字字母，否则它会被当成另一个长整型值，而被函数用于计算	否

6、__StringFromFile

__StringFromFile可以被用来从文本文件中读取字符串，这对需要大量可变数据的测试很有用。例如，当测试一个银行系统时，测试人员可能需要100条甚至1000条账户信息。

使用配置元件CSV Data Set Config ，也能达到相同的目的，而且方法更简单，但是该配置元件目前不支持多输入文件。

每次调用函数，都会从文件中读取下一行。当到达文件末尾时，函数又会从文件开始处重新读取，直到最大循环次数。如果在一个测试脚本中对该函数有多次引用，那么每一次引用

都会独立打开文件，即使文件名是相同的（如果函数读取的值，在脚本其他地方也有使用，那么就需要为每一次函数调用指定不同的变量名）。

如果在打开或者读取文件时发生错误，那么函数就会返回字符串"**ERR**"。

参数如下表所示：

函数参数	**描述**	**是否必需**
文件名	文件名（可以使用相对于JMeter启动目录的相对路径）。如果要在文件名中使用可选的序列号，那么文件名必须适合转成十进制格式。参考下面的例子	是
变量名	一个引用名（refName）的目的是复用这一函数创建的值。可以使用语法${refName}来引用函数创建的值。默认值为“StringFromFile_”	否
初始序列号	初始序列号（如果省略这一参数，终止序列号会作为一个循环计数器）	否
终止序列号	终止序列号（如果省略这一参数，序列号会一直增加下去，不会受到限制）	否

当打开或者重新打开文件时，文件名参数将会被解析；每次执行函数时，引用名参数（如果支持）将会被解析。

使用序列号：当使用可选的序列号时，文件名需要使用格式字符串Java.text.DecimalFormat；当前的序列号会作为唯一的参数，如果不指明可选的初始序列号，

就使用文件名作为起始值。一些有用的格式序列如下：

#：插入数字，不从零开始，不包含空格。

000：插入数字，包含3个数字组合，不从零开始。

例如：

①pin#'.'dat -> pin1.dat, ... pin9.dat, pin10.dat, ... pin9999.dat

②pin000'.'dat -> pin001.dat ... pin099.dat ... pin999.dat ... pin9999.dat

③pin'.'dat# -> pin.dat1, ... pin.dat9 ... pin.dat999

如果不希望某个格式字符被翻译，测试人员需要为它加上单引号；注意"."是格式字符，必须被单引号所包含。

如果省略了初始序列号，而终止序列号参数将会作为循环计数器，文件将会被使用指定的次数。例如：

${_StringFromFile(PIN#'.'DAT,,1,2)}：读取 PIN1.DAT, PIN2.DAT。

${_StringFromFile(PIN.DAT,,,2)}：读取 PIN.DAT 两次。

7、__machineName

函数__machineName返回本机的主机名。

参数如下表所示：

函数参数	描述	是否必需
变量名	重用函数计算值的引用名	否

8、__javaScript

函数_*javaScript可以用来执行*[*avaScript*](#)*代码片段（非ava），并返回结果值。JMeter的*javaScript函数会调用标准的JavaScript解释器；

JavaScript会作为脚本语言使用，因此测试人员可以做相应的计算；在脚本中可以访问如下一些变量：

Log：该函数的日志记录器;

Ctx：JmeterContext对象;

Vars：JmeterVariables对象;

threadName：字符串包含当前线程名称 （在2.3.2 版本中它被误写为"theadName"）；

sampler：当前采样器对象（如果存在）；

sampleResult：前面的采样结果对象（如果存在）；

props：JMeter属性对象;

Rhinoscript允许通过它的包对象来访问静态方法;例如，用户可以使用如下方法访问JMeterContextService静态方法：

Packages.org.apache.jmeter.threads.JMeterContextService.getTotalThreads()

JMeter不是一款浏览器，它不会执行从页面下载的JavaScript;

参数如下表所示：

函数参数	描述	是否必需
JavaScript代码片段	待执行的JavaScript代码片段；例如：n new Date()：返回当前日期和时间n Math.floor(Math.random()*(${maxRandom}+1))：在0和变量maxRandom之间的随机数n ${minRandom}+Math.floor(Math.random()*(maxRandom−maxRandom−{minRandom}+1))：n 在变量 minRandom和maxRandom之间的随机数n "${VAR}"=="abcd"	是
变量名	重用函数计算值的引用名	否

请记得为文本字符串添加必要的引号。另外，如果表达式中有逗号，请确保对其转义。例如，{**javaScript('{**javaScript('{sp}'.slice(7\,99999))}，对7之后的逗号进行了转义。

9、__Random

函数__Random会返回指定最大值和最小值之间的随机数。

参数如下表所示：

函数参数	描述	是否必需
最小值	最小数值	是
最大值	最大数值	是
变量名	重用函数计算值的引用名	否

10、__CSVRead

函数__CSVRead会从CSV文件读取一个字符串（请注意与StringFromFile 的区别）。

JMeter 1.9.1以前的版本仅支持从单个文件中读取，JMeter 1.9.1及其以后版本支持从多个文件中读取。在大多数情况下，新配置元件CSV Data Set更好用一些。

当对某文件进行第一次读取时，文件将被打开并读取到一个内部数组中；如果在读取过程中找到了空行，函数就认为到达文件末尾了，即允许拖尾注释（JMeter 1.9.1版本引入）；

后续所有对同一个文件名的引用，都使用相同的内部数组。另外，文件名大小写对函数调用很重要，哪怕操作系统不区分大小写，CSVRead(abc.txt,0)和CSVRead(aBc.txt,0)

会引用不同的内部数组；使用*ALIAS特性可以多次打开同一个文件，另外还能缩减文件名称。

每一个线程都有独立的内部指针指向文件数组中的当前行；当某个线程第一次引用文件时，函数会为线程在数组中分配下一个空闲行。如此一来，任何一个线程访问的文件行，

都与其他线程不同（除非线程数大于数组包含的行数）。

默认下，函数会在遇到每一个逗号处断行；如果希望在输入列中使用逗号，那么需要换一个分隔符（设置属性csvread.delimiter实现），且该符号没有在CSV文件任何列中出现。

参数如下表所示：

函数参数	描述	是否必需
文件名	设置从哪个文件读取（或者*ALIAS）	是
列数	从文件的哪一列读取。0=第一列， 1=第二列，依此类推。“next”为走到文件的下一行。 *ALIAS为打开一个文件，并给它分配一个别名	是

例如，可以用如下参数来设置某些变量：

①COL1a ${__CSVRead(random.txt,0)}

②COL2a {**CSVRead(random.txt,1)}{**CSVRead(random.txt,1)}{__CSVRead(random.txt,next)}

③COL1b ${__CSVRead(random.txt,0)}

④COL2b {**CSVRead(random.txt,1)}{**CSVRead(random.txt,1)}{__CSVRead(random.txt,next)}

上面例子会从一行中读取两列，接着从下一行中读取两列。如果所有变量都在同一个前置处理器中（用户参数上定义），那么行都是顺序读取的。否则，不同线程可能会读取不同的行。

这一函数并不适合于读取很大的文件，因为整个文件都会被存储到内存之中。对于较大的文件，请使用配置元件CSV Data Set或者StringFromFile 。

11、__property

函数__property会返回一个JMeter属性的值。如果函数找不到属性值，而又没有提供默认值，则它会返回属性的名称。

例如：

${__property(user.dir)}：返回属性user.dir的值；

${__property(user.dir,UDIR)}：返回属性user.dir的值，并保存在变量UDIR中；

${__property(abcd,ABCD,atod)}：返回属性abcd的值 （如果属性没有定义，返回"atod"），并保存在变量ABCD中；

${__property(abcd,,atod)}：返回属性abcd 的值（如果属性没有定义，返回"atod"），但是并不保存函数的返回值；

参数如下表所示：

函数参数	描述	是否必需
属性名	获取属性值、所需的属性名	是
变量名	重用函数计算值的引用名	否
默认值	属性未定义时的默认值	否

12、_P

函数*P是一个简化版的属性函数，目的是使用在命令行中定义的属性。不同于函数*property，本函数没有提供选项用于设置保存属性值的变量。

另外，如果没有设置默认值，默认值自动设为1。之所以选择1，原因在于它对于很多常见测试变量都是一个合理值，例如，循环次数、线程数、启动线程耗时间等。

例如：定义属性值：

jmeter -Jgroup1.threads=7 -Jhostname1=www.realhost.edu

获取值如下：

${__P(group1.threads)}：返回属性group1.threads的值；

${__P(group1.loops)}：返回属性group1.loops 的值；

${__P(hostname,www.dummy.org)}：返回属性hostname的值，如果没有定义该属性则返回值www.dummy.org；

在上面的例子中，第一个函数调用返回7，第二个函数调用返回1，而最后一个函数调用返回www.dummy.org（除非这些属性在其他地方有定义）；

参数如下表所示：

函数参数	描述	是否必需
属性名	获取属性值、所需的属性名	是
默认值	属性未定义时的默认值。如果省略此参数，默认值自动设为1	否

13、__log

函数__log会记录一条日志，并返回函数的输入字符串。

参数如下表所示：

函数参数	描述	是否必需
待记录字符串	一个字符串	是
日志级别	OUT、ERR、DEBUG、INFO（默认）、WARN或者ERROR	否
可抛弃的文本	如果非空，会创建一个可抛弃的文本传递给记录器	否
注释	如果存在，注释会在字符串中展示，用于标识日志记录了什么	否

OUT 和ERR的日志级别，将会分别导致输出记录到System.out和System.err中。在这种情况下，输出总是会被打印（它不依赖于当前的日志设置）。

例如：

${__log(Message)}：写入日志文件，形如"...thread Name : Message"；

${__log(Message,OUT)}：写到控制台窗口；

{**log({**log({VAR},,,VAR=)}：写入日志文件，形如"...thread Name VAR=value"；

14、__logn

函数__logn会记录一条日志，并返回空字符串。

参数如下表所示：

函数参数	描述	是否必需
待记录字符串	一个字符串	是
日志级别	OUT，ERR，DEBUG， INFO（默认），WARN 或者ERROR	否
可抛弃的文本	如果非空，会创建一个可抛弃的文本传递给记录器	否

OUT和ERR的日志级别，将会分别导致输出记录到System.out和System.err中。在这种情况下，输出总是会被打印（它不依赖于当前的日志设置）。

例如：{**logn(VAR1={**logn(VAR1={VAR1},OUT)}：将变量值写到控制台窗口中。

15、__BeanShell

函数__BeanShell会执行传递给它的脚本，并返回结果。关于BeanShell的详细资料，请参考BeanShell的Web站点：http://www.beanshell.org/ 。

需要注意，测试脚本中每一个独立出现的函数调用，都会使用不同的解释器，但是后续对函数调用的援引会使用相同的解释器；这就意味着变量会持续存在，并跨越函数调用。

单个函数实例可以从多个线程调用。另外，该函数的execute()方法是同步的。如果定义了属性"beanshell.function.init"，那么它会作为一个源文件传递给解释器。

这样就可以定义一些通用方法和变量。在bin目录中有一个初始化文件的例子：BeanShellFunction.bshrc。

如下变量在脚本执行前就已经设置：

log：函数BeanShell(*)的记录器；

ctx：目前的JMeter Context变量；

vars：目前的JMeter变量；

props：JMeter属性对象；

threadName：线程名（字符串）；

sampler：当前采样器（如果存在）；

sampleResult：当前采样器（如果存在）；

"*"意味着该变量在JMeter使用初始化文件之前就已经设置了。其他变量在不同调用之间可能会发生变化；

参数如下表所示：

函数参数	描述	是否必需
BeanShell脚本	一个BeanShell脚本（不是文件名）	是
变量名	重用函数计算值的引用名	否

例如：

${__BeanShell(123*456)}：回56088；

${__BeanShell(source("function.bsh"))}：行在function.bsh中的脚本；

请记得为文本字符串及代表文本字符串的JMeter变量添加必要的引号；

16、__plit

函数__split会通过分隔符来拆分传递给它的字符串，并返回原始的字符串。如果分隔符紧挨在一起，那么函数就会以变量值的形式返回"?"。

拆分出来的字符串，以变量${VAR_1}、{VAR_2}...以此类推的形式加以返回。JMeter 2.1.2及其以后版本，拖尾的分隔符会被认为缺少一个变量，会返回"?"。

另外，为了更好地配合ForEach控制器，现在__split会删除第一个不用的变量（由前一次分隔符所设置）。

在测试计划中定义变量VAR="a||c|"：{**split({**split({VAR},VAR),|}，该函数调用会返回VAR变量的值，例如"a||c|"，并设定VAR_n=4（3，JMeter 2.1.1及以前版本）、

VAR_1=a、VAR_2=?、VAR_3=c、VAR_4=?（null，JMeter 2.1.1及其以前版本）、VAR_5=null（JMeter 2.1.2及其以后版本）变量的值。

参数如下表所示：

函数参数	描述	是否必需
待拆分字符串	一个待拆分字符串，例如“a|b|c”	是
变量名	重用函数计算值的引用名	否
分隔符	分隔符，例如“**|**”。如果省略了此参数，函数会使用逗号做分隔符。需要注意的是，假如要多此一举，明确指定使用逗号，需要对逗号转义，如“**\,**”	否

17、__XPath

函数__XPath读取XML文件，并在文件中寻找与指定XPath相匹配的地方。每调用函数一次，就会返回下一个匹配项。到达文件末尾后，会从头开始。

如果没有匹配的节点，那么函数会返回空字符串，另外，还会向JMeter日志文件写一条警告信息；整个节点列表都会被保存在内存之中。

例如：

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

这会找到build.xml文件中的所有目标节点，并返回下一个name属性的内容。

参数如下表所示：

函数参数	描述	是否必需
XML文件名	一个待解析的XML文件名	是
XPath	一个XPath表达式，用于在XML文件中寻找目标节点	是

18、__setProperty

函数__setProperty用于设置JMeter属性的值。函数的默认返回值是空字符串，因此该函数可以被用在任何地方，只要对函数本身调用是正确的。

通过将函数可选的第3个参数设置为"true"，函数就会返回属性的原始值。属性对于JMeter是全局的，因此可以被用来在线程和线程组之间通信。

参数如下表所示：

函数参数	描述	是否必需
属性名	待设置属性名	是
属性值	属性的值	是
True/False	是否返回属性原始值	否

19、__time

函数__time可以通过多种格式返回当前时间。

参数如下表所示：

函数参数	描述	是否必需
格式	设置时间所采用的格式	否
变量名	待设置变量名	否

如果省略了格式字符串，那么函数会以毫秒的形式返回当前时间。其他情况下，当前时间会被转成简单日期格式。包含如下形式：

YMD = yyyyMMdd；

HMS = HHmmss；

YMDHMS = yyyyMMdd-HHmmss；

USER1 = JMeter属性time.USER1；

USER2 = JMeter属性time.USER2；

用户可以通过修改JMeter属性来改变默认格式，例如：time.YMD=yyMMdd。

20、_jexl

函数_jexl可以用于执行通用JEXL表达式，并返回执行结果。感兴趣的读者可以从下面这两个网页链接获取更多关于JEXL的信息。

http://commons.apache.org/jexl/reference/syntax.html。

http://commons.apache.org/jexl/reference/examples.html#Example_Expressions。

参数如下表所示：

函数参数	描述	是否必需
表达式	待执行的表达式。例如，6*(5+2)	是
变量名	待设置变量名	否

如下变量可以通过脚本进行访问：

log：函数记录器；

ctx：JMeterContext对象；

vars：JMeterVariables对象；

props：JMeter属性对象；

threadName：字符串包含当前线程名称 （在2.3.2 版本中它被误写为"theadName"）；

sampler：当前的采样器对象（如果存在）；

sampleResult：前面的采样结果对象（如果存在）；

OUT - System.out，例如 OUT.println("message");

JEXL可以基于它们来创建类，或者调用方法，例如：

①Systemclass=log.class.forName("java.lang.System");

②now=Systemclass.currentTimeMillis();

需要注意的是，Web站点上的JEXL文档错误地建议使用"div"做整数除法。事实上"div"和"/"都执行普通除法。

JMeter 2.3.2以后的版本允许在表达式中包含多个声明。JMeter 2.3.2及其以前的版本只处理第一个声明（如果存在多个声明，就会记录一条警告日志）。

21、__V

函数__V可以用于执行变量名表达式，并返回执行结果。它可以被用于执行嵌套函数引用（目前JMeter不支持）。

例如，如果存在变量A1、A2和N=1，则：

${A1}：能正常工作；

{A{A{N}}：无法正常工作（嵌套变量引用）；

{**V(A${N})}：可以正常工作。A{**V(A${N})}：可以正常工作。A{N}变为A1，函数 __V返回变量值A1；

参数如下表所示：

函数参数	描述	是否必需
变量名表达式	待执行变量名表达式	是

22、__evalVar

函数__evalVar可以用来执行保存在变量中的表达式，并返回执行结果。

如此一来，用户可以从文件中读取一行字符串，并处理字符串中引用的变量。例如，假设变量"query"中包含有"selectcolumnfromcolumnfrom{table}"，

而"column"和"table"中分别包含有"name"和"customers"，那么${__evalVar(query)}将会执行"select name from customers"。

参数如下表所示：

函数参数	描述	是否必需
变量名	待执行变量名	是

23、__eval

函数__eval可以用来执行一个字符串表达式，并返回执行结果。如此一来，用户就可以对字符串（存储在变量中）中的变量和函数引用做出修改。

例如，给定变量name=Smith、column=age、table=birthdays、SQL=select columnfromcolumnfrom{table} where name='${name}'，

那么通过{**eval({**eval({SQL})}，就能执行"select age from birthdays where name='Smith'"。这样一来，就可以与CSV数据集相互配合；

例如：将SQL语句和值都定义在数据文件中。

参数如下表所示：

函数参数	描述	是否必需
变量名	待执行变量	是

24、__char

函数**char会将一串数字翻译成Unicode字符，另外还请参考下面**unescape()函数。

参数如下表所示：

函数参数	描述	是否必需
Unicode字符编码（十进制数或者十六进制数）	待转换的Unicode字符编码，可以是十进制数或者十六进制数	是

Unicode字符编码（十进制数或者十六进制数） 待转换的Unicode字符编码，可以是十进制数或者十六进制数；

例如：

①${__char(0xC,0xA)} = CRLF

②${__char(165)} = ï¿½ (yen)

25、__unescape

函数**unescape用于反转义Java-escaped字符串，另外还请参考上面的**char函数。

参数如下表所示：

函数参数	描述	是否必需
待反转义字符串	待反转义字符串	是

例如：

①${__unescape(\r\n)} = CRLF

②${__unescape(1\t2)} = 1[tab]2

26、__unescapeHtml

函数__unescapeHtml用于反转义一个包含HTML实体的字符串，将其变为包含实际Unicode字符的字符串。支持HTML 4.0实体。

例如，字符串"<Français>"变为"<Fran?ais>"。

如果函数不认识某个实体，就会将实体保留下来，并一字不差地插入结果字符串中。例如，">&zzzz;x"会变为">&zzzz;x"。

参数如下表所示：

函数参数	描述	是否必需
待反转义字符串	待反转义字符串	是

27、__escapeHtml

函数__escapeHtml用于转义字符串中的字符（使用HTML实体）。支持HTML 4.0实体。

例如，"bread" & "butter"变为"bread" & "butter"。

参数如下表所示：

函数参数	描述	是否必需
待转义字符串	待转义字符串	是

28、__FileToString

函数__FileToString可以被用来读取整个文件。每次对该函数的调用，都会读取整个文件。

如果在打开或者读取文件时发生错误，那么函数就会返回字符串"**ERR**"。

参数如下表所示：

函数参数	描述	是否必需
文件名	包含路径的文件名（路径可以是相对于JMeter启动目录的相对路径）	是
文件编码方式（如果不采用平台默认的编码方式）	读取文件需要用到的文件编码方式。如果没有指明就使用平台默认的编码方式	否
变量名	引用名（refName）用于重用函数创建的值	否

三、变量

1、预定义变量

大多数变量都是通过函数调用和测试元件（如用户定义变量）来设置的；在这种情况下用户拥有对变量名的完整控制权。但是有些变量是JMeter内置的。例如：

Cookiename：包含Cookie值。

JMeterThread.last_sample_ok：最近的采样是否可以（true/false）。

2、预定义变量属性

JMeter属性集是在JMeter启动时通过系统属性初始化的；其他补充JMeter属性来自于jmeter.properties、user.properties或者命令行。

JMeter还另外定义了一些内置属性。下面是具体列表。从方便的角度考虑，属性START的值会被复制到同名变量中去。

START.MS：以毫秒为单位的JMeter启动时间；

START.YMD：JMeter启动日期格式yyyyMMdd；

START.HMS：JMeter启动时间格式HHmmss；

TESTSTART.MS：以毫秒为单位的测试启动时间；

请注意：START变量/属性表征的是JMeter启动时间，而非测试的启动时间。它们主要用于文件名之中。

文章出处：《零成本实现性能测试-jmeter》

（16）定时器

知识来源有点复杂，其他测试工作者的博客，百度百科，搜集的电子文档，个人理解等等，限于水平和理解能力，可能有些内容有错误的地方。。。

jmeter提供了很多元件，帮助我们更好的完成各种场景的性能测试，其中，定时器（timer）是很重要的一个元件，最新的3.0版本jemter提供了9种定时器（之前6种），下面一一介绍：

一、定时器的作用域

1、定时器是在每个sampler（采样器）之前执行的，而不是之后（无论定时器位置在sampler之前还是下面）；

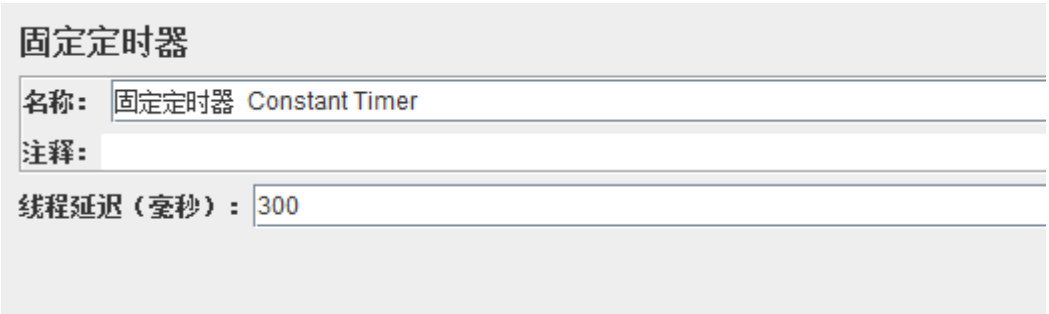
2、当执行一个sampler之前时，所有当前作用域内的定时器都会被执行；

3、如果希望定时器仅应用于其中一个sampler，则把定时器作为子节点加入；

4、如果希望在sampler执行完之后再等待，则可以使用Test Action；

二、定时器的作用

1、固定定时器（Constant Timer）



如果你需要让每个线程在请求之前按相同的指定时间停顿，那么可以使用这个定时器；需要注意的是，固定定时器的延时不会计入单个sampler的响应时间，但会计入事务控制器的时间。

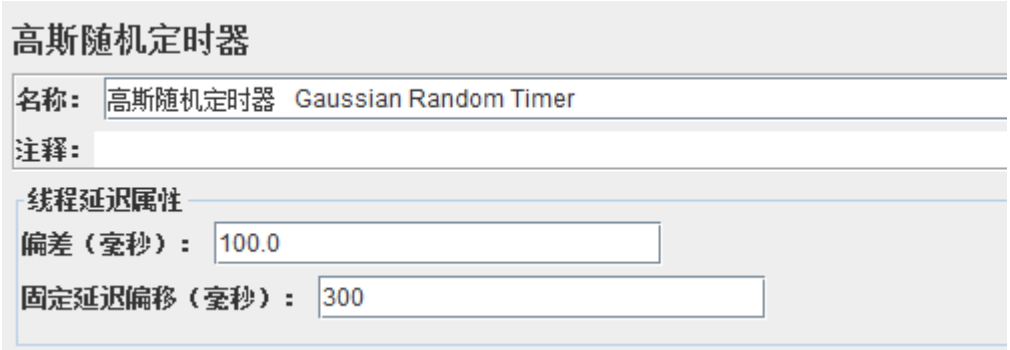
对于“java请求”这个sampler来说，定时器相当于loadrunner中的pacing（两次迭代之间的间隔时间）；

对于“事务控制器”来说，定时器相当于loadrunner中的think time（思考时间：实际操作中，模拟真实用户在操作过程中的等待时间）。

这里附上一个传送门，对loadrunner中的pacing和think time有比较全面的解释：https://zhidao.baidu.com/question/1431215934913423459.html

我们通常说的响应时间，应该大部分情况下是针对某一个具体的sampler（http请求），而不是针对一组sampler组合的事务 。

2、高斯随机定时器（Gaussian Random Timer）



如需要每个线程在请求前按随机时间停顿，那么使用这个定时器，上图表示暂停时间会分布在100到400之间，计算公式参考：Math.abs((this.random.nextGaussian() * 300) + 100)

传送门（什么是高斯随机分布）：https://zhidao.baidu.com/question/89318504.html

3、均匀随机定时器（Uniform Random Timer）

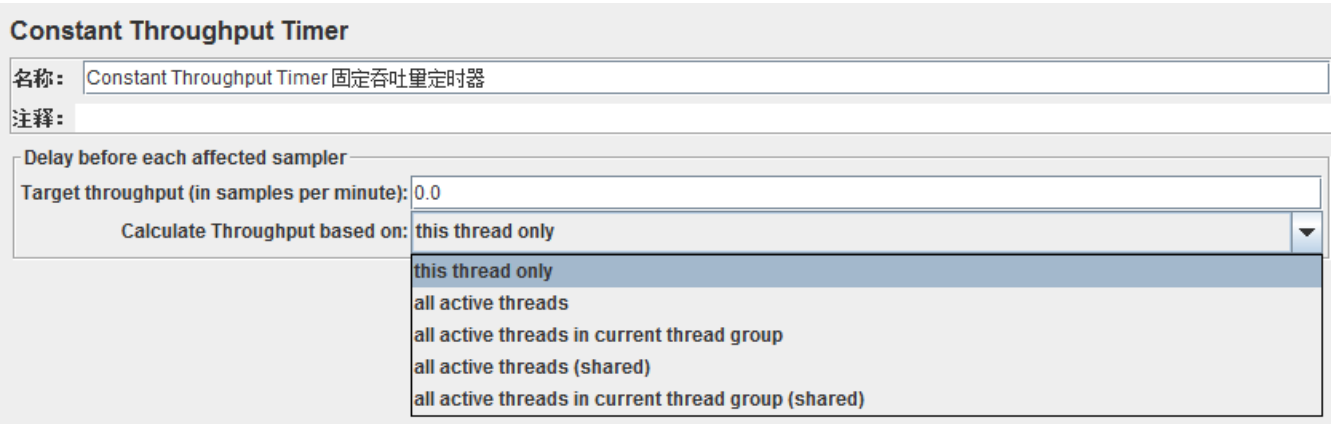
和高斯随机定时器的作用差异不大，区别在于延时时间在指定范围内且每个时间的取值概率相同，每个时间间隔都有相同的概率发生，总的延迟时间就是随机值和偏移值之和。

下面表示的是随机延迟时间的最大值是100毫秒：

（1）Random Delay Maximum(in milliseconds):随机延迟时间的最大毫秒数

（2）Constant Delay Offset(in milliseconds):暂停的毫秒数减去随机延迟的毫秒数

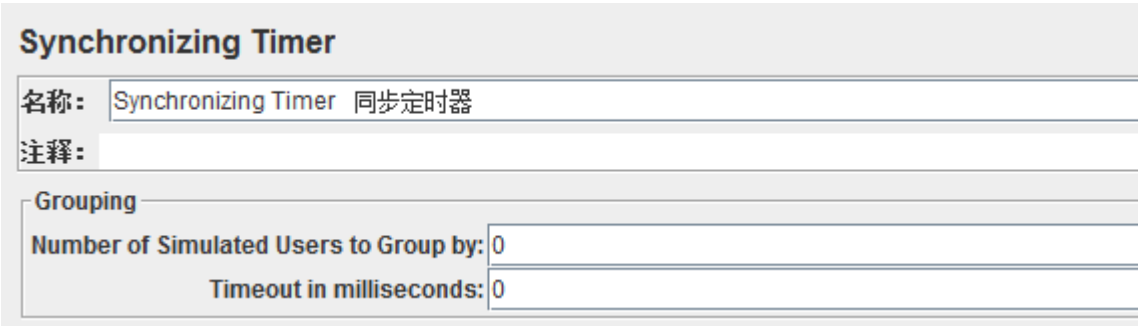
4、固定吞吐量定时器（Constant Throughput Timer）



可以让JMeter以指定数字的吞吐量（即指定TPS，只是这里要求指定每分钟的执行数，而不是每秒）执行。

吞吐量计算的范围可以为指定为当前线程、当前线程组、所有线程组等范围，并且计算吞吐量的依据可以是最近一次线程的执行时延。这种定时器在特定的场景下，还是很有用的。

5、同步定时器（Synchronizing Timer）

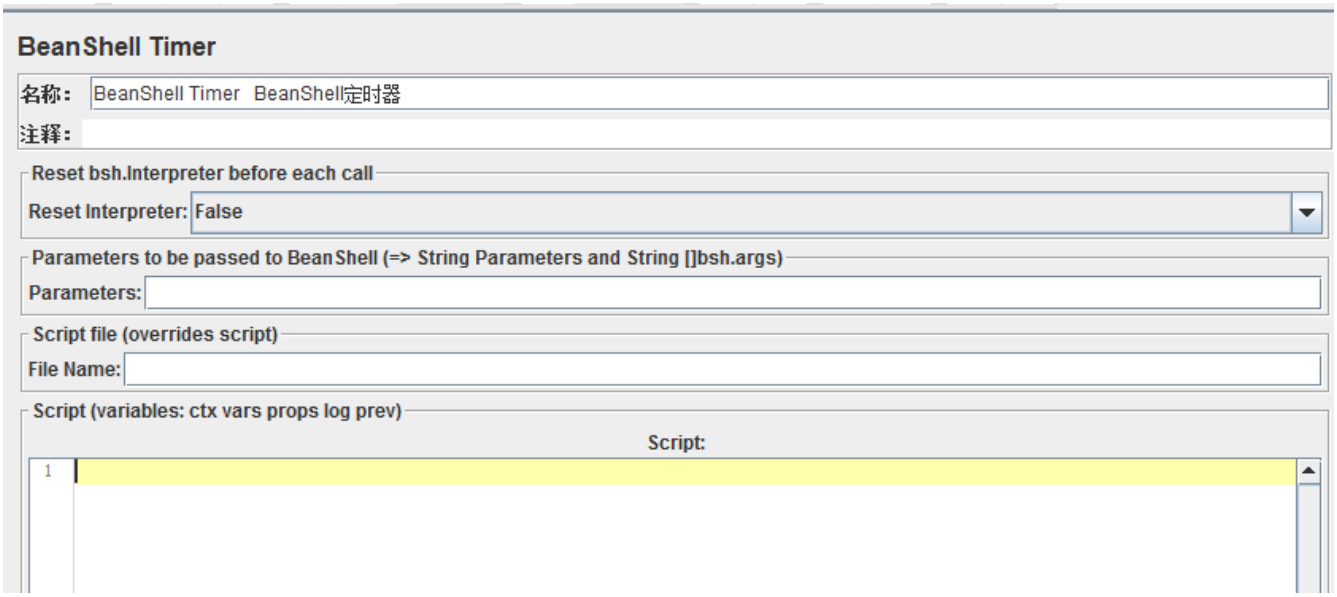


这个定时器和loadrunner当中的集合点（rendezvous point）作用相似，其作用是：阻塞线程，直到指定的线程数量到达后，再一起释放，可以瞬间产生很大的压力（人多力量大- -哈哈！）

（1）Number of Simulated Users to Group by:模拟用户的数量，即指定同时释放的线程数数量

（2）Timeout in milliseconds:超时时间，即超时多少毫秒后同时释放指定的线程数

6、BeanShell定时器（BeanShell Timer）



这个定时器，一般情况下用不到，但它可以说是最强大的，因为可以自己变成实现想要做的任何事情，例如：希望在每个线程执行完等待一下，或者希望在某个变量达到指定值的时候等待一下。

这里给大家介绍下BeanShell：

BeanShell是一种松散类型的脚本语言（这点和JS类似），一种完全符合java语法的java脚本语言，并且又拥有自己的一些语法和方法。

传送门（另外一位博客园作者的博客）：http://www.cnblogs.com/jssy/archive/2006/10/23/537101.html

7、泊松随机定时器（Poisson Random Timer）

这个定时器在每个线程请求之前按随机的时间停顿，大部分的时间间隔出现在一个特定的值，总的延迟就是泊松分布值和偏移值之和。

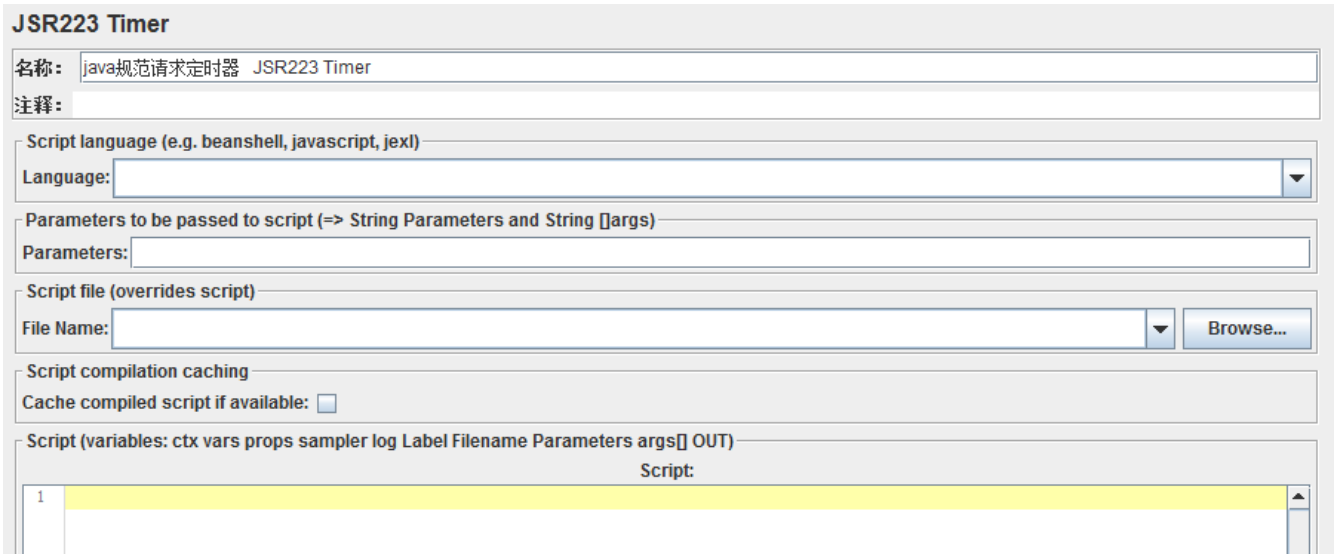
上面表示暂停时间会分布在100到400毫秒之间：

（1）Lambda(in milliseconds):兰布达值

（2）Constant Delay Offset(in milliseconds):暂停的毫秒数减去随机延迟的毫秒数

传送门（什么是泊松随机数）：http://baike.baidu.com/link?url=CJ0_Qtuilzp3a4Xos9N7V_hFQjaf_zb_aM1wggqxlYGDGWjtKsp6JSjRIQ110lE38sQOKYcgNUMjRuMAPGb3xK

8、JSR223定时器（JSR223 Timer）

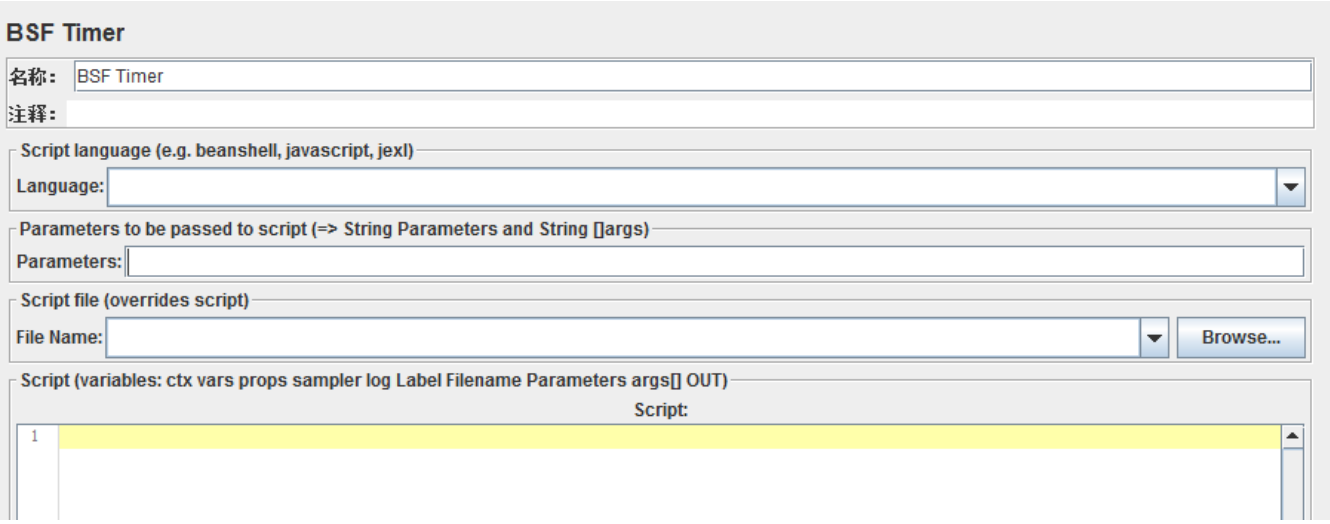


在jemter最新的版本中，新增了这个定时器，可以这么理解，这个定时器相当于BeanShell定时器的“父集”，它可以使用java、JavaScript、beanshell等多种语言去实现你希望完成的事情；

我们都知道jemter是一种开源的纯java工具，可以自己构件各个组件，jar包去完成各种事情。

传送门（关于JSR223）：http://wenku.baidu.com/link?url=GUFnww9nb_1D6MlFd1YksYrNVk1NXF74ov8kJL06MmqVdmH_Q9v4YnWK-_gZ-04zL4QEqD9VN48OrXi4JyXpxosNZd8LBfIWhyhhxgUbrAC

9、BSF定时器（BSF Timer）



BSF Timer，也是jmeter新的版本中新增的定时器，其使用方法和JSR223 Timer很相似，只需要在jmeter的lib文件夹导入其jar包，就可以支持脚本语言直接访问Java对象和方法的一定时器。

有了它，你就能在java application中使用javascript, Python, XSLT, Perl, tcl, ...等一大堆scripting language. 反过来也可以；

就是在这些scripting language中调用任何已经注册过了的JavaBean,java object。它提供了完整的API实现通过Java访问脚本语言的引擎。

由于本人对java了解不深，只能通过查阅相关资料，简单描述下其作用，不足之处，希望指正。

传送门（BSF）：http://baike.baidu.com/link?url=0RRkO1WqT1SdaXIzohqnEU8lcilpc_Sqwy7HtfpzCdCX1kyyLC5qttpF8jayTWFZi_tCbFbzMEw8FxHFYnIGYK

（17）断言

jmeter中有个元件叫做断言（Assertion），它的作用和loadrunner中的检查点类似；

用于检查测试中得到的响应数据等是否符合预期，用以保证性能测试过程中的数据交互与预期一致。

使用断言的目的：在request的返回层面增加一层判断机制；因为request成功了，并不代表结果一定正确。

使用断言的方法：

▵在选择的Sampler下添加对应的断言（因为不同类型的断言检查的内容不同）；配置好响应的检查内容（根据断言情况而定，有的断言控制面板不需要添加任何内容，如XML Assertion）。

▵添加一个断言结果的监听器（从监听器中添加），通过“断言结果”可以看到是否通过断言；对于一次请求，如果通过的话，断言结果中只会打印一行请求的名称；

如果失败，则除了请求的名称外，还会有一行失败的原因（不同类型的断言，结果不同）。

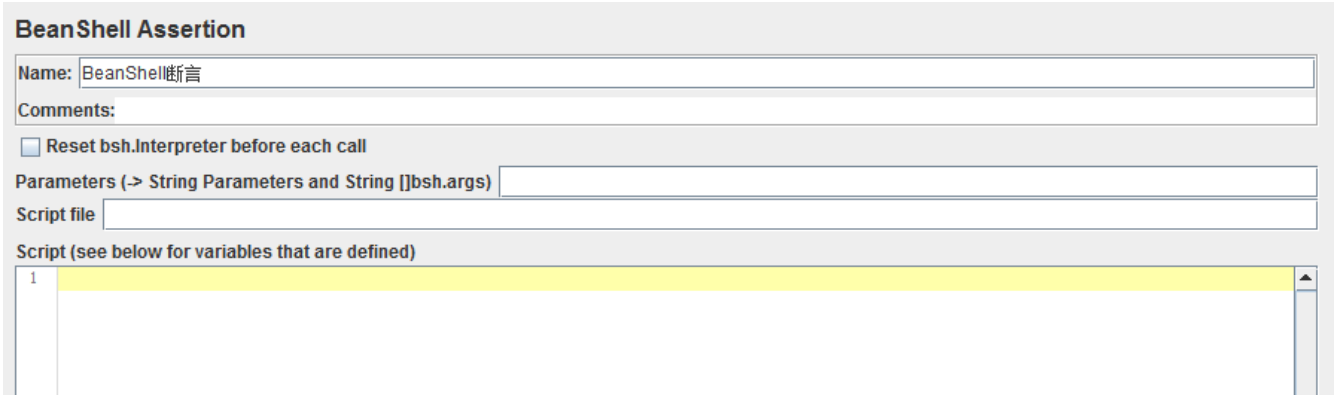
PS：一个Sampler可以添加多个断言，根据你的检查需求来添加相应的断言，当Sampler下所有的断言都通过了，那么才算request成功。

最新版本的3.0jmeter中有13种不同的断言，下面简单介绍下每个断言各自拥有什么样的作用以及它们的适用场景：

1、BeanShell断言

BeanShell之前关于定时器的随笔中有介绍过，是一种松散类型的脚本语言（这点和JS类似），一种完全符合java语法的java脚本语言，并且又拥有自己的一些语法和方法；

作用对象：针对sampler中的Bean Shell sampler而使用的断言



Name:断言的名字（可以用一个比较容易理解和分辨的名称）

Comments：注释（对这个断言进行一个解释，备注）

Reset bsh.interpreter before each call:在每次调用Bean Shell之前重置bsh.interpreter类（bsh.interpreter是Bean Shell脚本语言的一种类，也可以理解为一种解析器）

Parameters（String Parameters and String []bsh.args）: String参数（String []bsh.args是主类main函数的形式参数,是一个String 对象数组，可以用来获取命令行用户输入进去的参数）

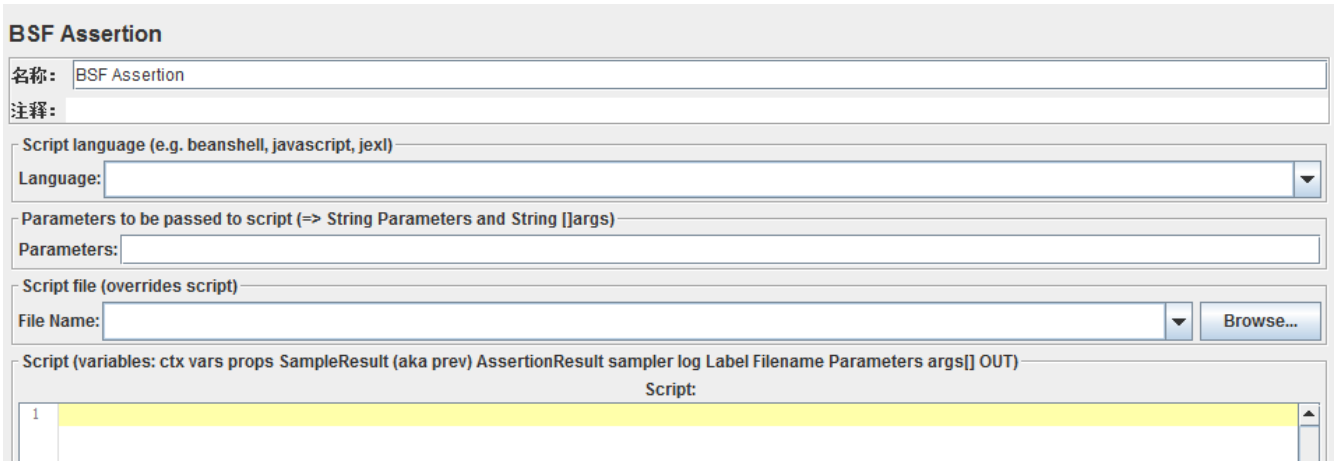
Script file：脚本文件（可以填入脚本文件路径）

Script（see below for variables that are defined）:参照下文定义的变量（使脚本文件参照定义的变量来运行）

2、 BSF断言

BSF(Bean Scripting Framework)之前也介绍过，是一个支持在Java应用程序内调用脚本语言 (Script)，并且支持脚本语言直接访问Java对象和方法的一个开源项目；

作用对象：针对sampler中的BSF sampler而使用的断言



Script language（e.g.beanshell,javascirpt,jexl）:脚本语言（可以从下面的下拉框中选择对应的脚本语言JavaScript、beanshell等）

parameters to be passed to script（=> String Parameters and String []args）：（传递给脚本的参数→可以理解为使用BSF断言脚本时候一起引用的参数 ）

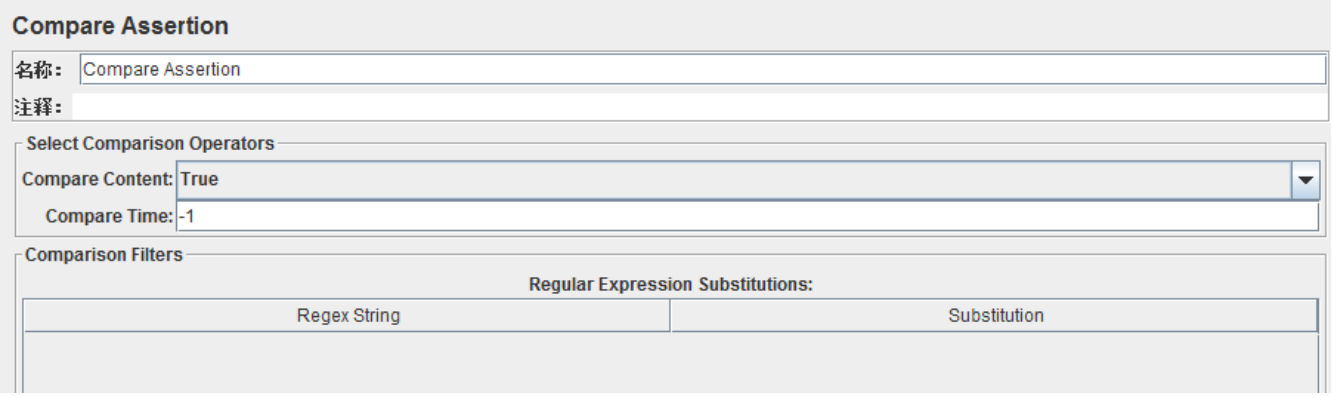
Script file（overrides script）：重写脚本（可以通过选择脚本文件的状态，是浏览调用已有的脚本还是在在下方的输入框内写入脚本；）

Script：下面的输入框表示可以输入变量类型，运用的脚本（取样结果、断言结果、取样日志文件等参数）

3、比较断言（compare assertion）

这是一种比较特殊的断言元件，针对断言进行字符串替换时使用；

作用对象：需要替换的字符串



Select Comparison Operators:选择比较运算符

Compare Content:可以选择比较的内容类型（true/false或者自定义，编辑）

Compare Time：比较时间（可以设定比较的时间，单位为秒，默认为-1）

Comparison Fitters:比较修改工具

regular expression substitutions:替换正则表达式

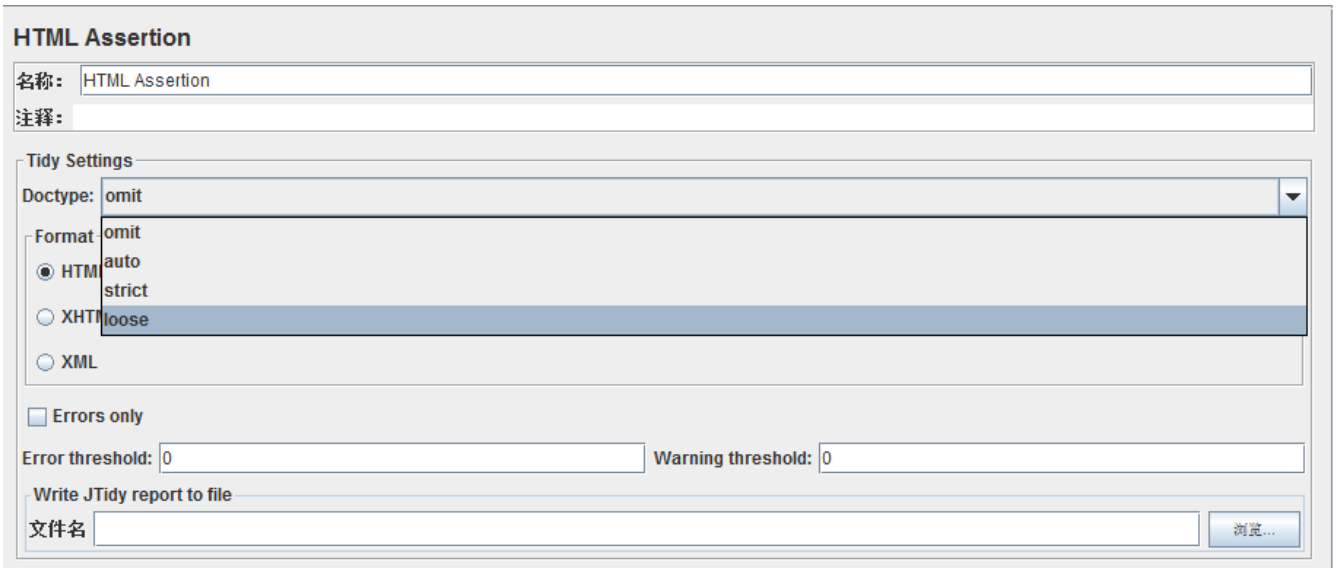
Regex String:要替换的字符串（可从断言结果中选择）

substitutions：替换的字符串（替换结果）

4、HTML断言

对响应类为XML类型的文件进行断言；

作用对象：针对sampler中的SOAP/XML-RPC Request而使用的断言



Tidy Settings: Tidy 环境（Tidy是一个HTML语法检查器和打印工具，可以将HTML转换为XML类型的文件）

Doctype:文档类型（可通过下拉框选择不同文档类型→ omit疏忽遗漏的/auto动态的/strict严格的/loose宽泛的。。。。。。我也不太懂这里什么意思GG）

Format：文件格式（可选择HTML/XHTML/XML三种不同类型的文件格式来检查返回内容）

Errors only：误差校正（能接受的最大值）

Error threshold：误差/错误范围（可选择误差/错误数量的范围，最大值）

Warning threshold：警告范围（可选择误差警告的数量范围，最大值）

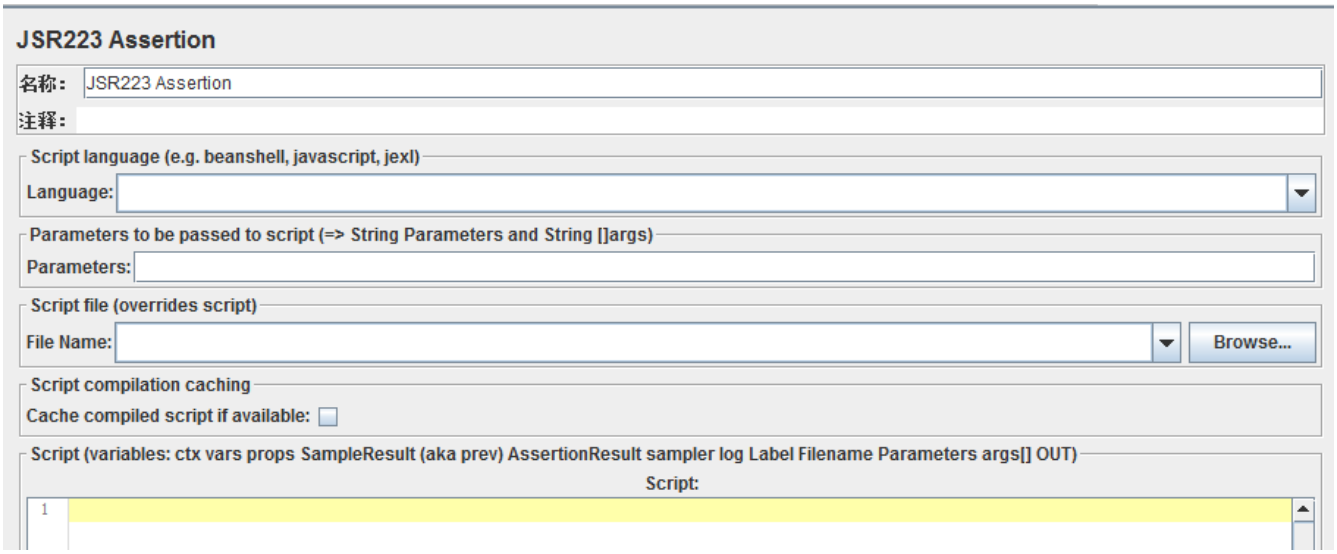
如果勾选“Error only”这里忽略Warning，只对误差作统计检查；如果对返回内容的检查结果不超过指定结果，则断言通过，否则失败。

Write JTidy report to file:写入JTidy报告的文件（JTidy是Tidy的一个java移植，可以将它当成一个处理HTML文件的DOM解析器）

5、JSR223断言

JSR223即Java 规范请求，是指向JCP(Java Community Process)提出新增一个标准化技术规范的正式请求；

作用对象：针对sampler中的JSR223 sampler而使用的断言



Script language（e.g.beanshell,javascirpt,jexl）:脚本语言（可以从下面的下拉框中选择对应的脚本语言JavaScript、beanshell等）

parameters to be passed to script（=> String Parameters and String []args）：（传递给脚本的参数→可以理解为使用JSR223断言脚本时候一起引用的参数 ）

Script file（overrides script）：重写脚本（可以通过选择脚本文件的状态，是浏览调用已有的脚本还是在在下方的输入框内写入脚本；）

Script：下面的输入框表示可以输入变量类型，运用的脚本（取样结果、断言结果、取样日志文件等参数）

6、MD5Hex断言

MD5是一种消息摘要算法，用以提供消息的完整性保护（具体关于MD5的知识请自行查询）；

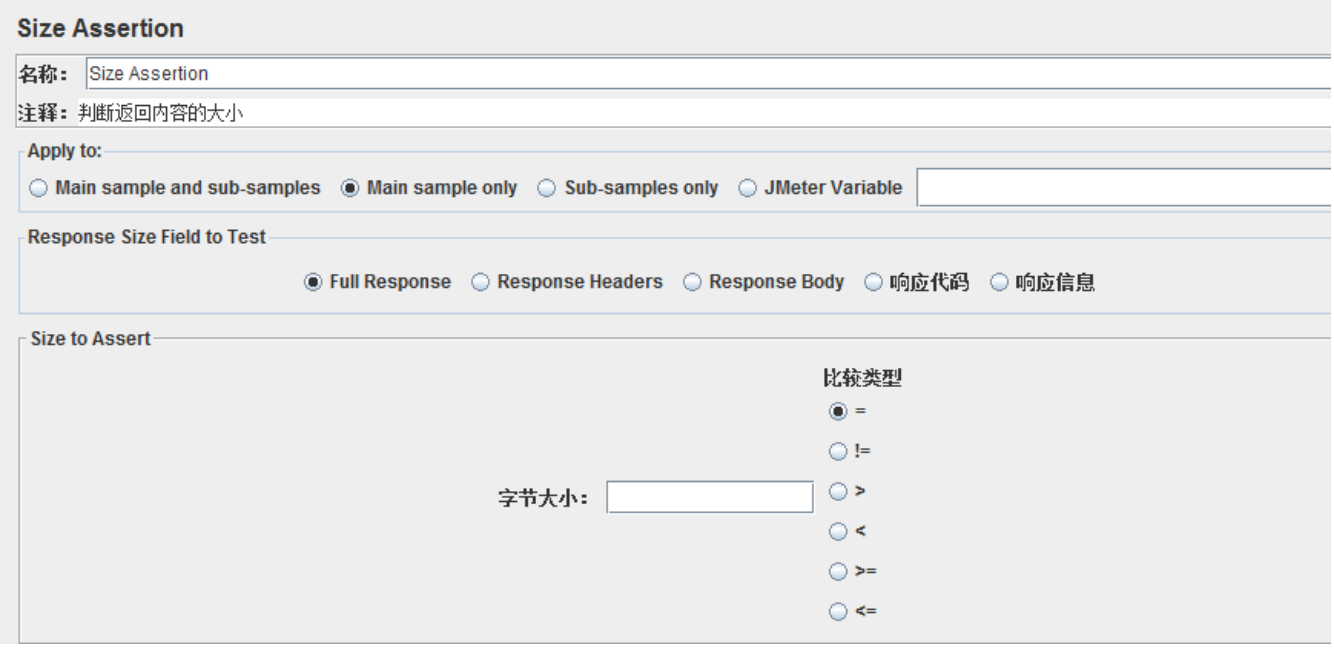
作用对象：针对参数类型为MD5Hex加密的参数的断言

MD5Hex：将已被MD5加密的参数写入其中，添加取样器等其他元件

7、Size断言

用于判断返回内容的大小；

作用对象：返回信息，响应报文



APPly to:应用范围（返回内容的断言范围）

Main sample and sub-samples:作用于父节点取样器及对应子节点取样器

Main sample only：仅作用于父节点取样器

Sub-samples only:仅作用于子节点取样器

JMeter Variable:作用于jmeter变量(输入框内可输入jmeter的变量名称)

Response Size Field to Test:响应字节的测试范围（可以选择用于判断的响应范围）

Full Response:全部响应

Response Headers:响应头部

Response Body:响应主体

响应代码：响应报文相关的代码

响应信息：响应报文的信息

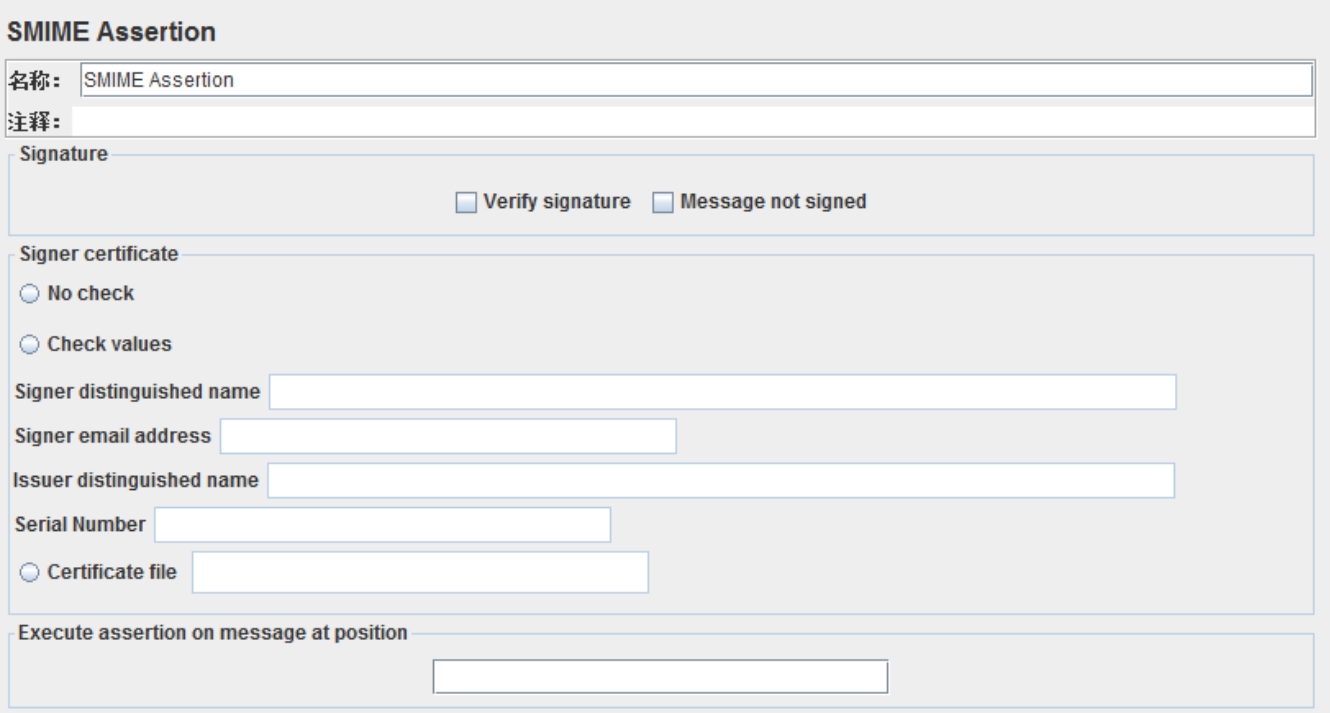
Size to Assert:断言字节范围

字节大小单位为：字节；比较顺序是①返回内容的大小②比较类型③指定字节大小

8、SMIME断言

SMIME是一种多用途网际邮件扩充协议，相比于之前的SMAP邮件传输协议，增加了安全性，对邮件主题进行保护；

作用对象：针对采用了该种邮件传输协议的信息



signature:签名（可选择对协议的签名验证状态）

Verify signature:验证签名

Message not signed:没有签名消息

Signer certificate：签名证书（因为SMIME协议增加了安全传输，需要证书验证）

No check：不检查

Check values:检查

Signer distinguished name:签名证书者名称（证书注册者的名称）

Sigmer email address:签名者的邮件地址（注册的邮件地址）

Issuer distinguished name:发行者名称（由谁发行的证书）

Serial Number:证书序号

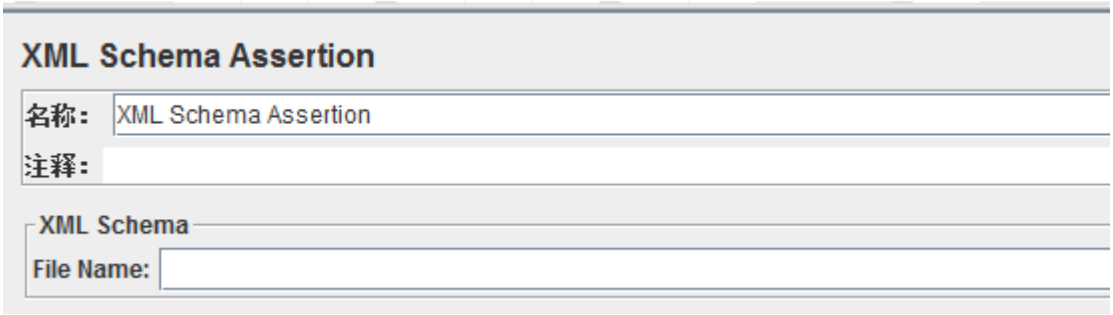
Certificate file:选择证书文件

Execute assertion message at position:执行断言消息的位置（在返回消息的具体哪个位置执行断言）

9、XML概要断言

亦可以称为XML模型断言/XML数据类型断言；XML Schema 定义了两种主要的数据类型：①xml document schema 文档架构 ;② 文档架构xml-schema xml模式

作用对象：返回结果为XML概要断言的2中数据类型的消息



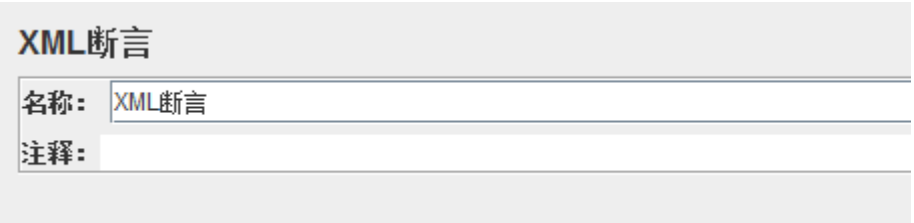
XML Schema：XML概要模型

File Name:文件名（写入需要断言的文件名称）

10、XML断言

XML(可扩展标记语言) 提供一种描述结构化数据的方法。与主要用于控制数据的显示和外观的 HTML 标记不同，XML 标记用于定义数据本身的结构和数据类型；

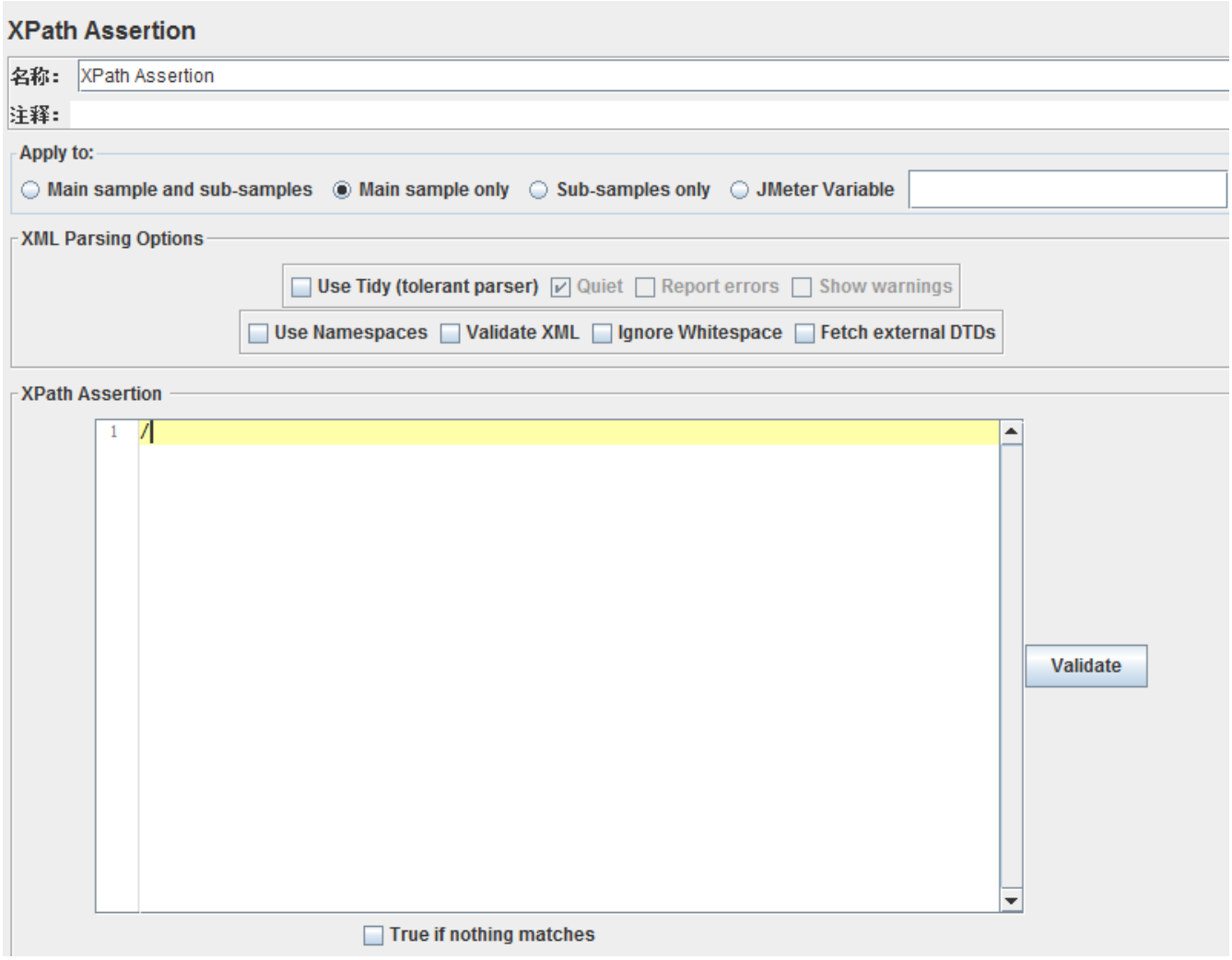
作用对象：判断返回结果是否和xml的格式即<></>成对出现



11、XPath断言

XPath即为XML路径语言，它是一种用来确定XML（标准通用标记语言的子集）文档中某部分位置的语言。XPath基于XML的树状结构，提供在数据结构树中找寻节点的能力。

作用对象：针对返回信息为XPAth的数据类型进行断言



Apply to:适用范围

Main sample and sub-samples:主要样本和次级样本

Main sample only：仅主要样本

Sub-samples only:仅次级样本

JMeter Variable:jmeter变量(输入框内可输入jmeter的变量名称)

XML Parsing Options：XML解析选项

Use Tidy(tolerant parser):使用Tidy（容错解析器），默认选择quiet（不显示）

Quiet：不显示

Report errors：错误报告

Show warnings:显示错误

Use Namespaces:使用名称空间

Validate XML:验证XML（文件包/数据）

Ignore Whitespace:忽略空格（这允许你指定语法分析器可以忽略哪个空格，而哪个空格是重要的）

Fetch external DTDs:获取外部DTDs（一些XML元素具有属性，属性包含应用程序使用的信息，属性仅在程序对元素进行读、写操作时，提供元素的额外信息，这时候需要在DTDs中声明）

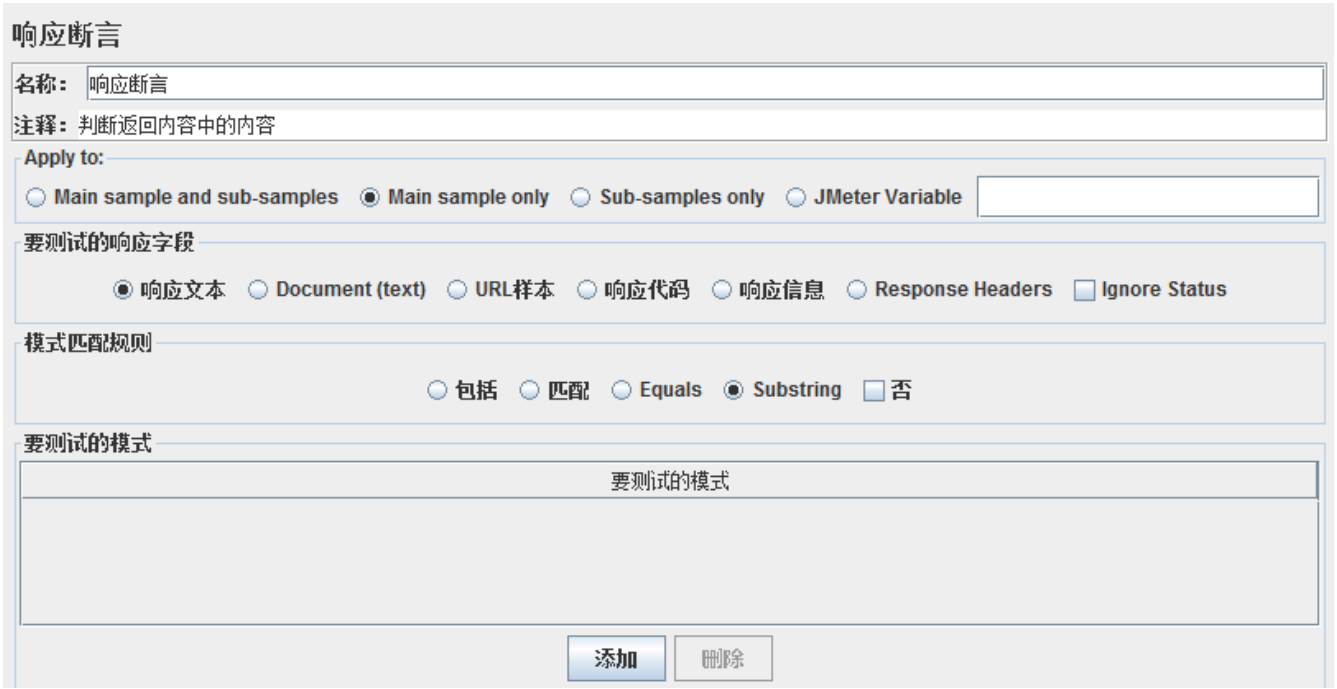
XPath Assertion:输入框中写入xpath断言，点击Validate验证其正确性

True if nothing matches:确认都不匹配

12、响应断言

判断返回内容中的内容

作用对象：响应报文中的所有对象



APPly to:适用范围

Main sample and sub-samples:作用于父节点取样器及对应子节点取样器

Main sample only：仅作用于父节点取样器

Sub-samples only:仅作用于子节点取样器

JMeter Variable:作用于jmeter变量(输入框内可输入jmeter的变量名称)

要测试的响应字段：要检查的项

响应报文

Documeng(text)：测试文件

URL样本

响应代码

响应信息

Response Headers:响应头部

Ignore status：忽略返回的响应报文状态码

模式匹配规则：

包括：返回结果包括你指定的内容

匹配：（好像跟Equals查不多，弄不明白有什么区别）

Equals：返回结果与你指定结果一致

Substring：返回结果是指定结果的字串

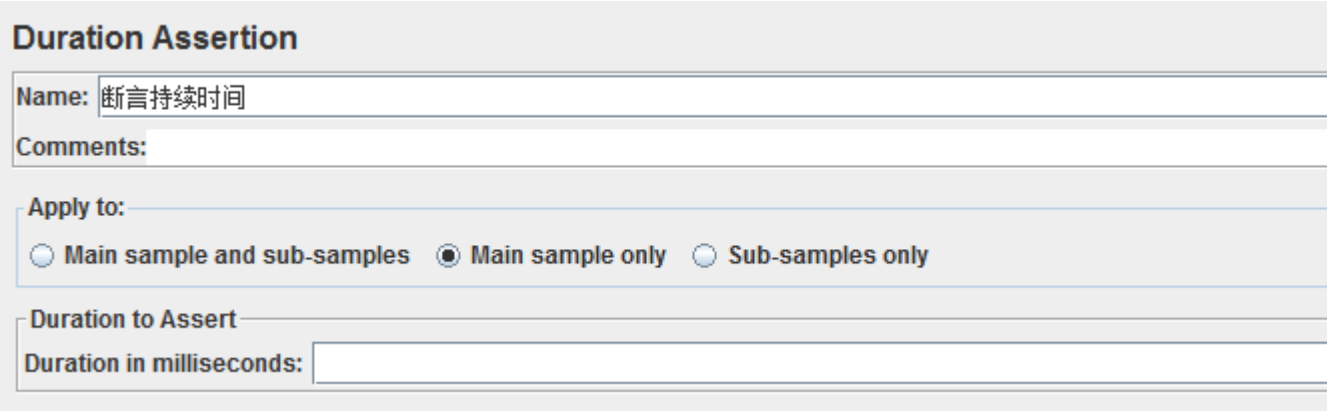
否：不进行匹配

要测试的模式:即填写你指定的结果（可填写多个）,按钮【添加】、【删除】是进行指定内容的管理

13、断言持续时间

用于判断服务器的响应时间

作用对象:服务器



APPly to:适用范围

Main sample and sub-samples:作用于父节点取样器及对应子节点取样器

Main sample only：仅作用于父节点取样器

Sub-samples only:仅作用于子节点取样器

Duration to assert：持续断言

Duration in milliseconds：响应时间设置（单位：毫秒），如果响应时间大于设置的响应时间，则断言失败，否则成功!

（18）逻辑控制器

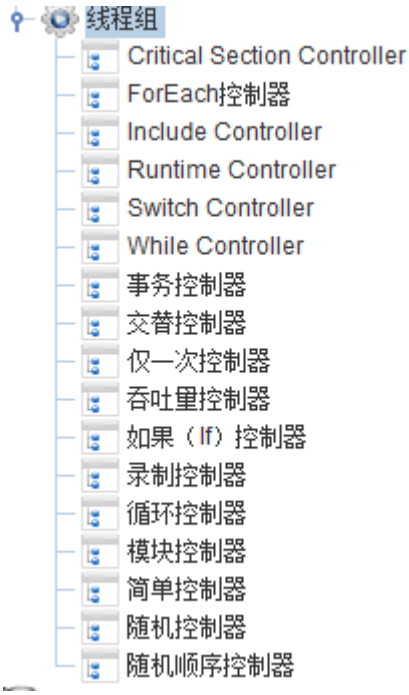
jmeter中逻辑控制器（Logic Controllers）的作用域只对其子节点的sampler有效，作用是控制采样器的执行顺序。

jmeter提供了17种逻辑控制器，它们各个功能都不尽相同，大概可以分为2种使用类型：

①.控制测试计划执行过程中节点的逻辑执行顺序，如：Loop Controller（循环控制器）、If Controller（如果if控制器）等；

②.对测试计划中的脚本进行分组，方便Meter统计执行结果以及进行脚本的运行时控制等，如：Throughput Controller（吞吐量控制器）、Transaction Controller（事务控制器）等

jmeter提供如下17种逻辑控制器：



一、临界区控制器（critical section Controller）

作用：临界区控制器确保其子节点下的取样器或控制器将被执行（只有一个线程作为一个锁）

名称和注释很简单，就是给控制器添加一个备注，使人明白这个控制器的解释含义的意思

Lock name：锁名称，这里可以填入其子节点下执行的线程的名称，这个线程作为一个全局锁存在

2、遍历循环控制器（ForEach Controller）

作用：用来遍历当前元素的所有可执行场景；在用户自定义变量中读取一系列相关的变量，该控制器下的采样器或控制器都会被执行一次或多次，每次读取不同的变量值；

输入变量前缀：在其中输入需要遍历的用户参数（User Parameter）

Start index for loop(exclusive)：循环指数开始（唯一）→ 遍历查询的变量范围，开始的值（这里如果不填写，默认从1开始，如果没有1开始的变量，执行时会报错）

End index for loop(inclusive)：循环指数结束（包含）→ 遍历查询的变量范围，结束的值

输出变量名称：将遍历查询到的符合条件的用户参数赋值给输入变量（Vname），然后就可以在控制器下的取样器使用，格式为${输出变量名}

Add"*"*before number：**输入变量名称中是否使用"*进行间隔

PS：这个控制器一般配合配置元件→ 正则表达式提取器来一起使用，可对页面上的某些元素进行重复处理。

3、包含控制器（Include Controller）

作用：用于引用外部的jmx文件；从而控制多个测试计划组合

include Test Plan：包含测试计划的文件名，可以点击浏览，从文件夹保存的JMX文件夹目录下选择对应的JMX文件

使用方法：创建一个测试计划，下面可添加取样器/控制器等，然后保存测试计划，为了方便起见, 线程组也可以添加外部JMX文件中用于调试；

如果测试使用Cookie或用户定义的变量,这些应放置在顶层（包括文件）,否则无法正常工作；此元素不支持变量/函数在文件名字段中；但是,如果属于包含控制器定义的内容，则使用前缀路径名。

当使用包含控制器中包含相同的JMX文件,则要确保文件名不同,以避免无法读取；如果文件不能被发现，那么控制器会尝试打开文件名相对于JMX启动目录。

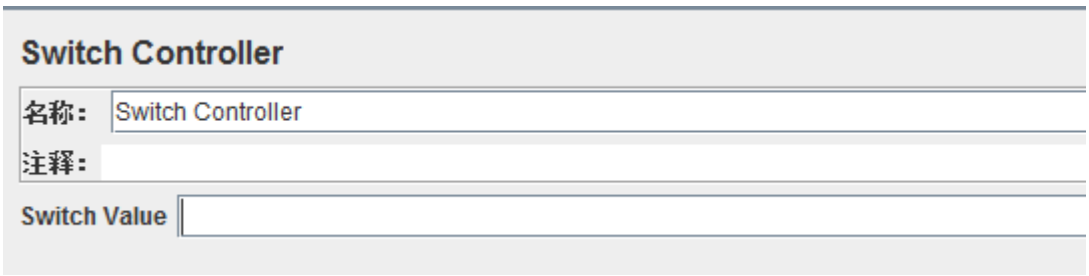
4、生命周期/运行周期控制器（Runtime Controller）

作用：用于控制该控制器下的取样器/控制器的运行时间

Runtime（seconds）：运行时间，单位为：秒

5、转换控制器（Switch Controller）

作用：通过给该控制器中的value赋值，来指定运行哪个取样器（也可以理解为开关控制器）



Switch value：控制器具体赋值的value值字段

有两种赋值方式：

①．第一种是数值，Switch控制器下的子节点从0开始计数，通过指定子节点所在的数值来确定执行哪个元素。

②．第二种是直接指定子元素的名称，比如采样器的Name来进行匹配。当指定的名称不存在时，不执行任何元素。

当Value为空时，默认执行第1个子节点元素。

6、当/判断控制器（While Controller）

作用：运行其子节点下的取样器/控制器，直到条件为“假”

Condition(function or variable):条件（函数或变量）：里面可填入判断依据的条件，参照

使用方法：可能的条件值有：

①.空白：最后一个示例循环失败时退出循环

②.最后一个值：最后一个示例循环失败时退出循环。 如果之前的最后一个示例只是循环失败,不进入循环。

③.否则：退出时(或不输入)循环条件等于字符串“ 假 ”

条件可以是任何变量或函数，最终等于字符串” 假 ”。需要注意的是：条件是评估两次,一次取样前,一次随机取样

7、事务控制器（transaction controller）

作用：生成一个额外的采样器来测量其下测试元素的总体时间；值得注意的是，这个时间包含该控制器范围内的所有处理时间，而不仅仅是采样器的

Generate parent sample：生成父样本（不同的模式选择）

include duration of timer and pre-post processors in generated sample：包含时间的计时器和前后处理器生成的示例（不同的模式选择）

对于Jmeter2.3以上的版本，有两种模式的操作

①.事务采样器是添加到其下采样器后面的

②.事务采样器是作为其下采样器的父采样器

生成的事务采样器的测量的时间包括其下采样器以及其他的一切时间。由于时钟频率问题，这个时间可能略大于单个采样器的时间之和；

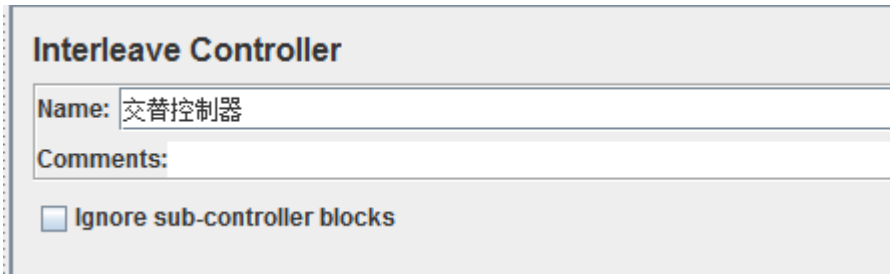
时钟开始时间介于控制器记录开始时间与第一个采样器开始之间，时钟结束时间亦然。

事务采样器只有在其子采样器都成功的情况下才显示成功。

在父模式下，事务控制器下的各个采样器只有在结果树里才能看到；同时，子采样器的数据也不会在CSV文件中显示，但是在XML文件中可以看到。

8、交替控制器（creatleave controller）

作用： 交替控制，使得该控制器包含的取样器步骤交错执行在每个循环中

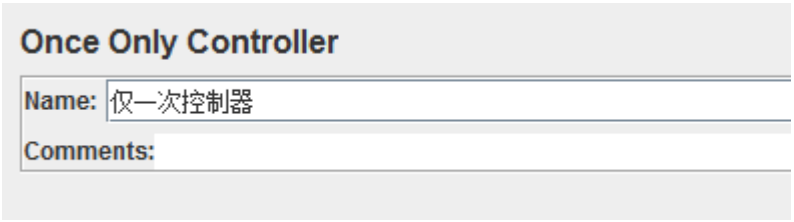


忽略子控制器模块（Ignore sub-contorller blocks）：如果勾选此项,交错控制器将sub-controllers像单一请求元素一样，一次只允许一个请求/控制器

使用方法：假使该控制器下有2个取样器A和B，交替执行A和B2个请求，即每次传递一个子请求到这个测试，按子元件的排列顺序

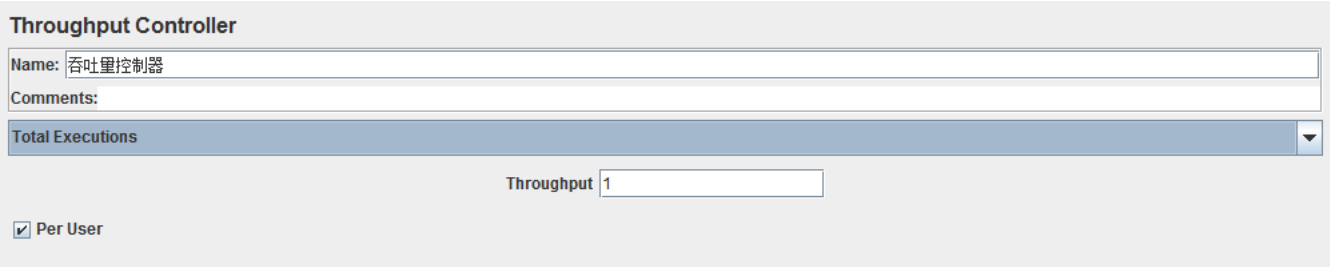
9、仅一次控制器（once only controller）

作用：在多线程循环的时候，将使其子节点下的取样器请求只运行一次



10、流量控制器（throughput controller）

作用：jmeter自带的翻译这里是错误的，因为它并不能控制吞吐量（吞吐量的概念请自行百度）；其实质作用是允许用户控制执行的频率



总共有两种执行模式：百分比执行和总执行

总执行（Total Executions）：使控制器停止执行一定数量的测试计划

百分比执行（Percent Executions）：使控制器按一定比例执行迭代的测试计划

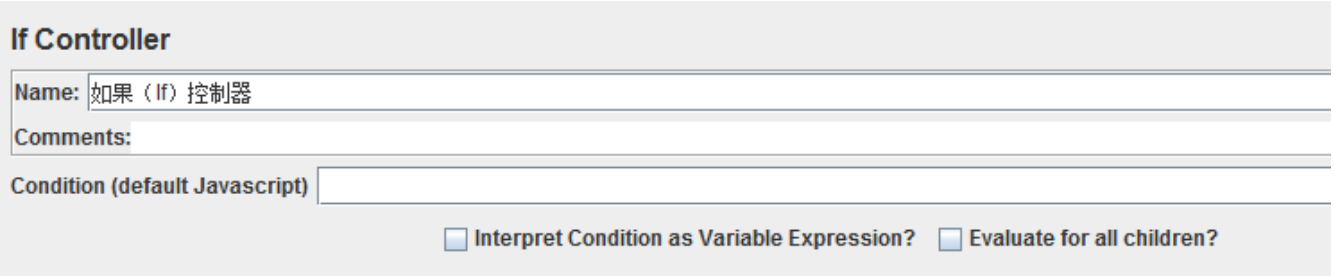
流量（Throughput）：对应上面的执行数量或者比例

每个用户（Per User）：每个用户

如果勾选此项,将导致控制器计算是否应该执行在每个用户(每个线程)的基础上；如果不加以控制,那么将计算全球所有用户

11、IF控制器（If Controller）

作用：允许用户控制该控制器下面的取样器/控制器是否执行该节点下的子节点；



条件（默认JavaScript）（Condition(default javascript)**）**:使用JavaScript的函数或变量进行评估判断条件为真或假

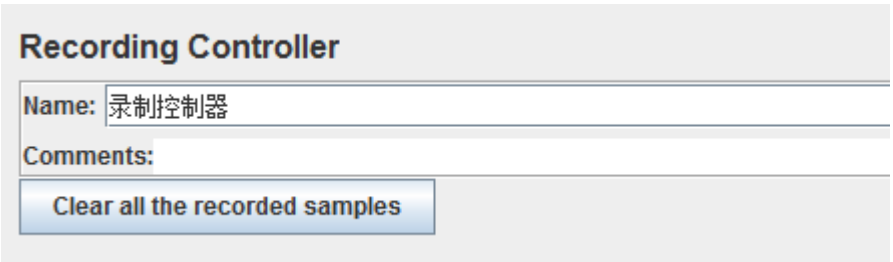
条件解释为变量表达式（interpret condition as variable expression）:如果勾选该项，那么变量表达式会进行求值，并与“ture”或“false”进行比较，而无需使用JavaScript

对所有子条件执行（evaluate for all children）:如果勾选该项，则该controller在没一个子节点执行时执行一次；

默认情况下，该控制器可以对包含在其下面的所有可运行的元素进行执行，但只在入口执行一次

12、**录制控制器**（Recording Controller）

作用：类似代理服务器的作用，在测试执行期间记录测试样本

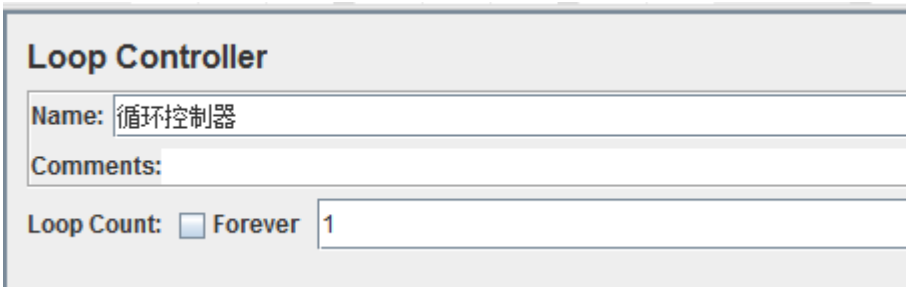


清除所有记录的样本(Clear all the recorded samples):点击可以清除所有已经记录的测试样本

一般情况下，在测试执行时候，它没有效果，但是在执行HTTPS测试脚本时，会记录下所有测试样本

13、循环控制器（Loop Controller）

作用：该控制器下的取样器请求可以循环运行

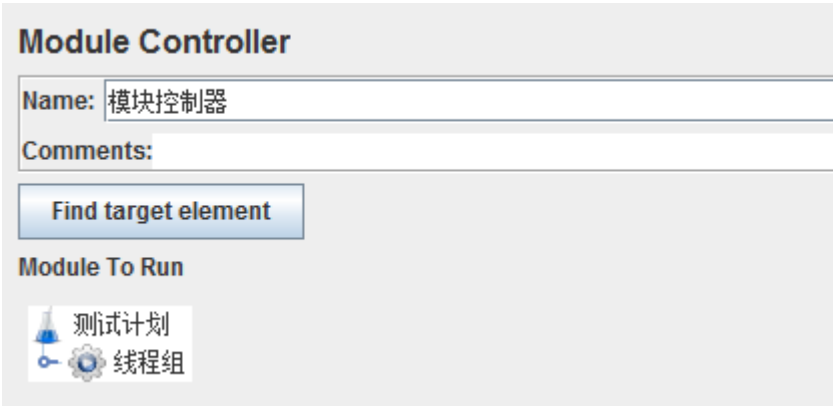


循环次数(Loop Count):在输入框中输入需要循环的次数，控制器下的请求即可循环运行

永远(forever)：如果勾选该项，那么控制器下的请求可一直运行

14、**模块控制器**（Module Controller）

作用：测试控制器子节点下的某一个模块，而不是整个测试计划



寻找目标元素(Find target element)：寻找测试计划中需要特定测试的元素，模块；也可理解为该控制器可以控制已经封装好的模块元素

▵：一个测试计划由一个控制器和所有的测试元素（取样器等）组成，测试计划可以位于任何线程组或工作台；如果计划位于线程组，则可以禁用其他控制器，防止正在运行的测试计划被影响（除了模块控制器）

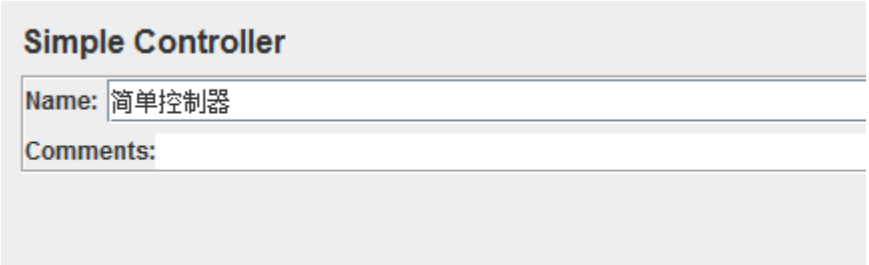
模块控制器的优势在于：当存在多个线程组时，该控制器可以轻松切换，只需要选择对应的取样器，方便快捷，替代了创建很多测试计划的繁琐操作

▵：任何一个模块所用的控制器名字必须唯一，因为其名字被用来找到目标控制器时重新加载；出于这个原因，最好保证控制器名字不同，否则执行测试时候可能发生意外

▵：模块控制器与远程测试不应使用或非gui测试与工作台部件，因为工作台测试元素并没有测试计划的一部分 jmx 文件。 任何这样的测试就会失败

15、**简单控制器**（Simple Controller）

作用：用来组合取样器和其他逻辑控制器



简单控制器是最基本的控制器，对jmeter测试运行没有任何影响，可用来命名某些操作

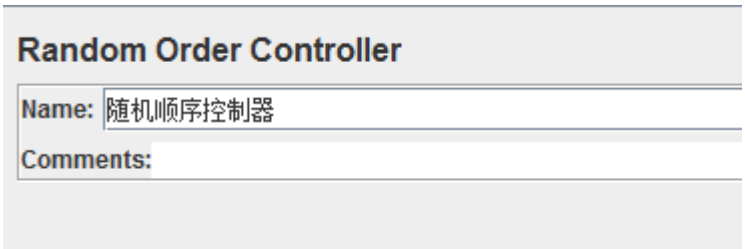
16、**随机控制器**（Random Controller）

作用：类似交替控制器，但该控制器随机选取某一个取样器请求并执行

忽略子控制器模块(Ignore sub-controller blocks)：如果勾选此项,交错控制器将sub-controllers像单一请求元素一样，一次只允许一个请求/控制器

17、**随机顺序控制器**（Random Order Controller）

作用：类似于简单控制器，将执行每个子节点下的取样器请求一次，但是执行是随机的



查询了很多资料，参考了其他博客作者的内容，可以说结果很不理想，大部分都是直接将jmeter官网文档用翻译词典翻译出来就贴上去了，错别字，解释语句不通；

偶尔看见一篇，也是寥寥几笔，关于这些控制器的具体使用方法，我自己也尝试的使用了一下，还是不难的，可能会有点误差，请谅解。。。

官网文档地址：http://jmeter.apache.org/usermanual/component_reference.html#logic_controllers

参考博客：http://www.cnblogs.com/kuihua/p/5537083.html

http://www.cnblogs.com/puresoul/p/4886574.html

（19）常见问题

jmeter作为一个开源的纯Java性能测试工具，工作中极大的方便了我们进行接口、性能测试，但使用过程中也遇到了很多的问题，下面就记录一下自己遇到的问题，后续会不断更新。。。

1、获取日志

在使用jmeter过程中，如果想获得更详细的日志，可以修改jmeter\bin\jmeter.properties文件中的一个属性：所有log_level.jmeter的后缀由info改为debug，如下：

```
log_level.jmeter=INFO
log_level.jmeter.junit=DEBUG
#log_level.jmeter.control=DEBUG
#log_level.jmeter.testbeans=DEBUG
#log_level.jmeter.engine=DEBUG
#log_level.jmeter.threads=DEBUG
#log_level.jmeter.gui=WARN
#log_level.jmeter.testelement=DEBUG
#log_level.jmeter.util=WARN
#log_level.jmeter.protocol.http=DEBUG
# For CookieManager, AuthManager etc:
#log_level.jmeter.protocol.http.control=DEBUG
#log_level.jmeter.protocol.ftp=WARN
#log_level.jmeter.protocol.jdbc=DEBUG
#log_level.jmeter.protocol.java=WARN
#log_level.jmeter.testelements.property=DEBUG
log_level.jorphan=INFO
```

2、jmeter安装

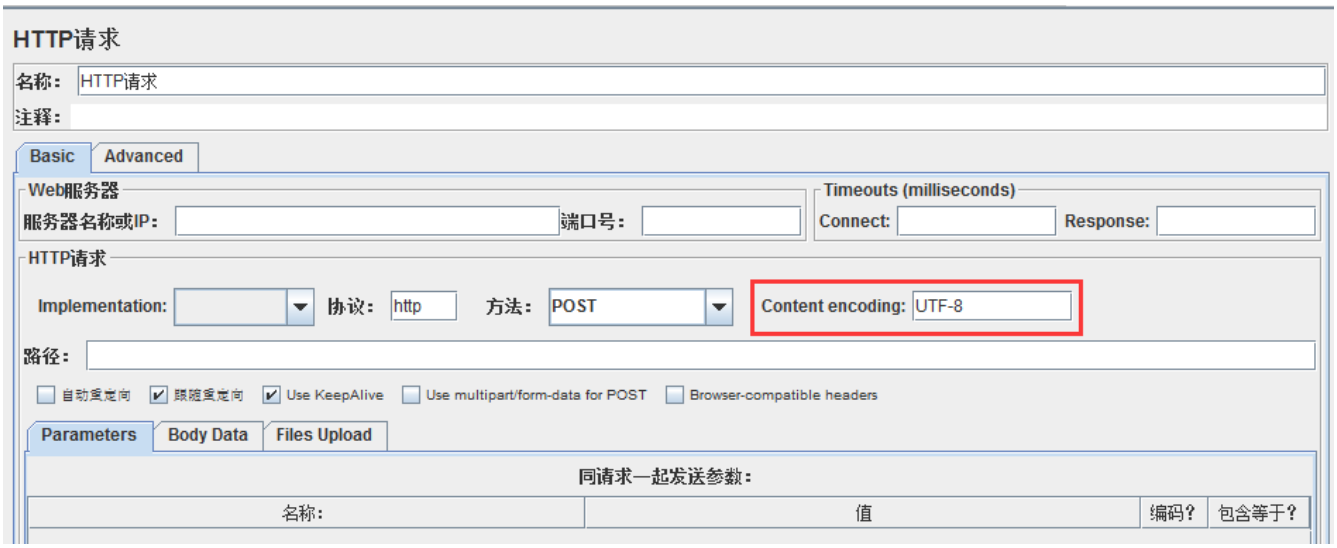
安装使用jmeter时候不需要设置classpath以及class变量，只需要默认安装好JDK即可（通常情况下），然后解压jmeter安装包，启动jmeter\bin\jmeter.bat程序即可；

因为jmeter是以Java_jar的方式启动，而且会忽略该变量，这对所有Java程序都适用。

3、请求/响应数据显示乱码

有时候在发送请求/查看响应数据时，服务端接收到的请求中包含乱码，导致无法解析报错，解决方法有如下几种：

①请求数据显示乱码，可以在请求中如下设置：



②返回数据包含乱码时，可以修改jmeter\bin\jmeter.properties文件中的一个属性：将encoding=后面的编码格式改为utf-8，如下：

```
# List of extra HTTP methods that should be available in select box
#httpsampler.user_defined_methods=VERSION-CONTROL,REPORT,CHECKOUT,CHECKIN,UNCHECKOUT,MKW

# The encoding to be used if none is provided (default ISO-8859-1)
#sampleresult.default.encoding=utf-8

# Network response size calculation method
# Use real size: number of bytes for response body return by webserver
# (i.e. the network bytes received for response)
# if set to false, the (uncompressed) response data size will used (default before 2.5)
```

PS：此模块更详细的原因说明，可参考这篇博客：http://blog.csdn.net/cakushin7433/article/details/53039566

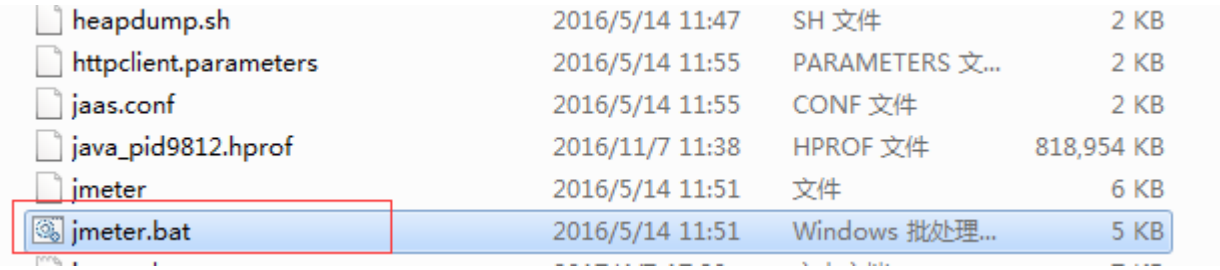
4、内存OOM（OutOfMemoryError：内存溢出）

在执行压力测试时候，有时候会遇到OutOfMemoryError这样的异常；JMeter是一个纯Java开发的工具，内存是由java虚拟机JVM管理；如果出现了内存溢出的问题，

可以通过调整JVM内存相关的参数进行优化。

具体过程如下：

①找到jmeter.bat文件，也就是我们启动jmeter的脚本：



②打开jmeter.bat文件，对一下这些配置项进行编辑：

```
rem See the unix startup file for the rationale of the following parameters,
rem including some tuning recommendations
set HEAP=-Xms512m -Xmx512m
set NEW=-XX:NewSize=128m -XX:MaxNewSize=128m
set SURVIVOR=-XX:SurvivorRatio=8 -XX:TargetSurvivorRatio=50%
set TENURING=-XX:MaxTenuringThreshold=2
rem Java 8 remove Permanent generation, don't settings the PermSize
if %current_minor% LEQ "8" (
    rem Increase MaxPermSize if you use a lot of Javascript in your Test Plan :
    set PERM=-XX:PermSize=64m -XX:MaxPermSize=128m
)
```

③参数调整：

调整堆内存的大小：

将默认的set HEAP=-Xms512m -Xmx512m，调整为set HEAP=-Xms1024m -Xmx1024m；

调整堆内存中新生带的大小：

将默认的set NEW=-XX:NewSize=128m -XX:MaxNewSize=128m，调整为set NEW=-XX:NewSize=256m -XX:MaxNewSize=256m；

调整堆内存中永久带的大小：

将默认的set PERM=-XX:PermSize=64m -XX:MaxPermSize=128m，调整为set PERM=-XX:PermSize=128m -XX:MaxPermSize=256m；

调整后重启jmeter，问题一般可以得到解决（参数的调整不能一概而论，具体根据测试机的硬件配置来决定）。

5、Listener使用技巧

listener作为一个收集sampler的结果数据和呈现结果的文件，其本身会在每次sampler运行完成后执行一次，即一个test plan中的listener数量越多，运行时listener本身带来的资源消耗

就越大（尤其是view results in table以及view results tree等）。

因此实际执行test plan时，应首先禁用不需要的listener，再开始执行；更好的方式是每次运行时将生成的结果写入结果文件中，方便以后用不同的listener展现保存的结果数据。

当然，在并发量较大的情况下，一般的测试机限于配置等因素，无法支撑较大的并发数，可以用以下的方法来进行测试，方法如下：

去掉listener，为sampler添加断言（一般是响应断言），根据断言结果来判断请求是否成功，测试报告以plugins插件中的报告形式或文本形式写入文件中来提升测试效率。

PS：这个方法是我认识的一个妹子她之前说的一种方法，感兴趣的可以去看看她的博客，链接：http://www.cnblogs.com/sunshine2016/

6、调试test plan

很多测试人员在初始进行性能测试时，脚本都是录制得到的，但录制的脚本一般都包含很多对本次测试来说无用的sampler，以及录制的sampler需要重新修改参数等内容，才能使用。

所以调试test plan就很有必要，常用的有以下2种方法：

①使用listener观察sampler的请求和相应

录制的脚本，一般都需要剔除无用的sampler，然后修改参数，进行调试，才能用于测试执行，一般用于调试的listener是结果树，可以在测试计划中将线程组的数量修改为1，然后执行。

listener显示的每一个sampler结果为绿色（表示通过），但jmeter仅根据http返回码来判断sampler执行是否成功，这样无法判断sampler语义上的错误；因此，一般都是在sampler

中插入对应的检查点（Assertion：断言），根据返回的内容，来判断sampler是否真正成功。

②使用http Mirror server观察sampler发出的请求

在调试和修改sampler时，经常会为其增加一些额外的设置，例如额外的信息头、cookie管理器等，但设置完成后直接运行脚本进行测试，并不能保证请求真的和我们预期的一致。

如果不想将请求发送给被测应用，可以使用http mirror server组件（http镜像服务器）。

http mirror server可以启动一个镜像服务器，其可以把所有接收到的请求原封不动的返回，这样就可以查看发出的请求的具体内容。

使用方法如下：

点击工作台，右键添加→http mirror server，如有必要修改服务器端口（一般修改为localhost：8080，方便调试），然后启动镜像服务器；

其次修改需要调试的sampler，将其请求发送到mirror server启动的端口，运行测试计划，即可以从listener中查看响应数据。

PS：其实http mirror server更大的作用是检查浏览器是否发送了特殊的http头，启动mirror server，使用浏览器访问该server，则可以在返回页面看到浏览器发送请求的完整内容。

（20）阶梯加压测试

性能测试中，有时需要模拟一种实际生产中经常出现的情况，即：从某个值开始不断增加压力，直至达到某个值，然后持续运行一段时间。

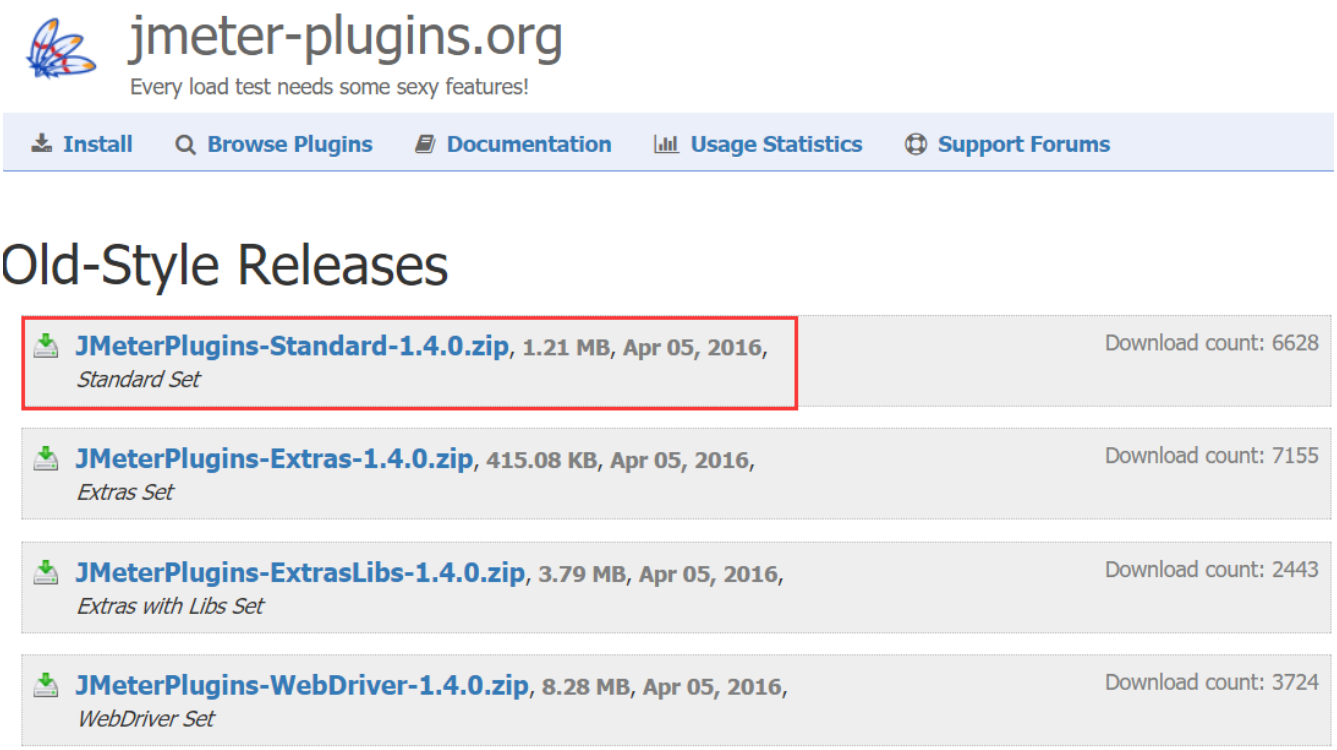
在jmeter中，有这样一个插件，可以帮我们实现这个功能，这个插件就是：**Stepping Thread Group**

1、下载配置方法

Stepping Thread Group是jmeter插件的一种，其作用就是模拟实际的生产情况，不断对服务器施加压力，直至到某个值，然后持续运行一段时间。

下载地址：https://jmeter-plugins.org/downloads/old/

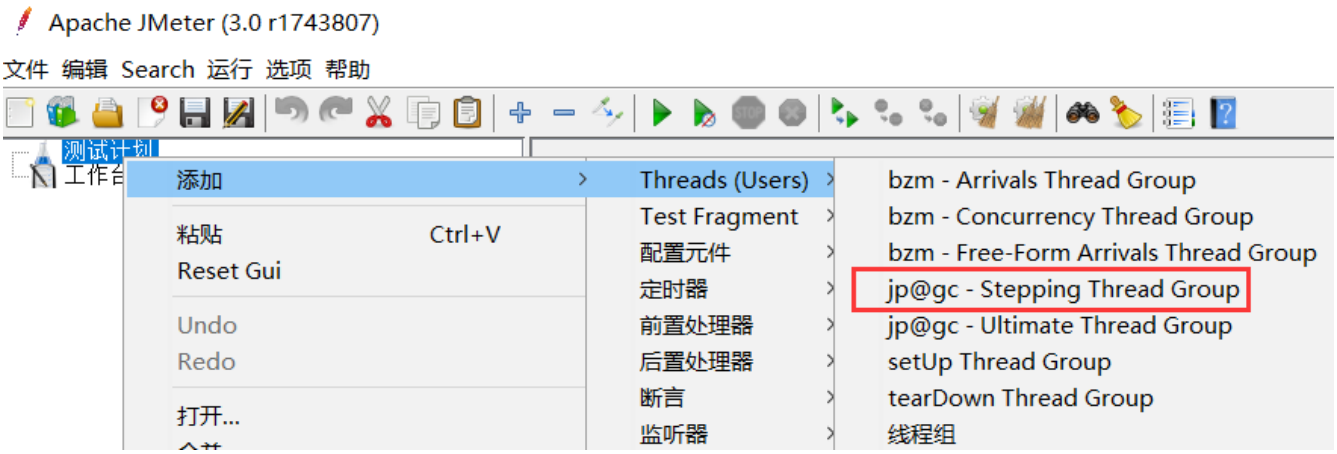
下载界面如下：



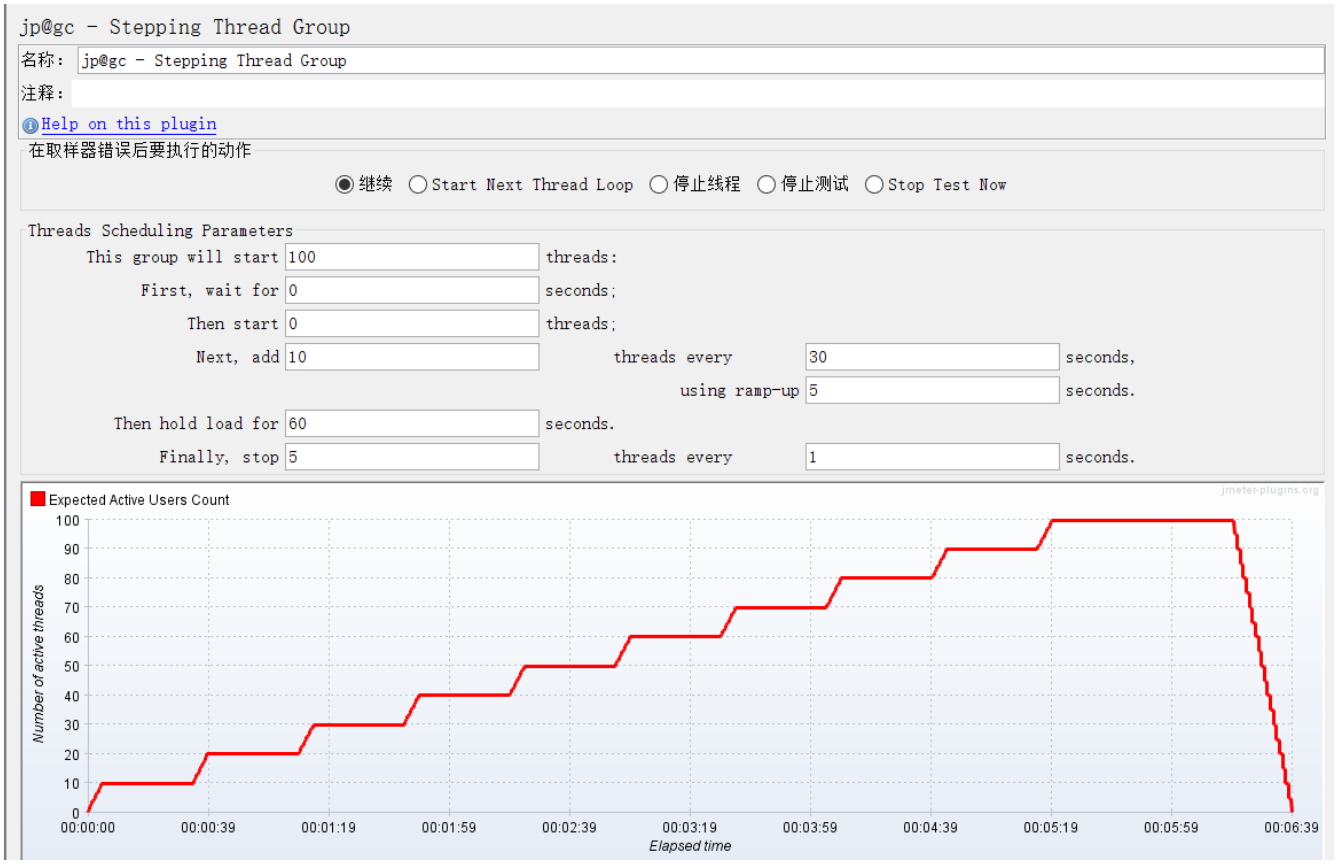
下载后需要解压，然后将JMeterPlugins-Standard.jar包放在**jmeter安装目录的jmeter-3.0\lib\ext路径**下，重新启动jemter即可。

2、使用介绍

启动jmeter，添加线程组——jp@gc - Stepping Thread Group，如下图：



Stepping Thread Group界面如下：



功能如下：

This group will start 100 threads：设置线程组启动的线程总数为100个；

First,wait for N seconds：启动第一个线程之前，需要等待N秒；

Then start N threads：设置最开始时启动N个线程；

Next,add 10 threads every 30 seconds,using ramp-up 5 seconds：每隔30秒，在5秒内启动10个线程；

Then hold load for 60 seconds：启动的线程总数达到最大值之后，再持续运行60秒；

Finally,stop 5 threads every 1 seconds：每秒停止5个线程；

三、相关插件

Stepping Thread Group插件相对来说比较旧，在plugins插件组中，还有一个类似的优化过的插件，叫做：**Concurrency Thread Group**

相关介绍以及下载地址如下：https://jmeter-plugins.org/wiki/ConcurrencyThreadGroup/

其实最好的办法，是直接下载jmeter的第三方插件Plugin Manager（其中包含了很多扩展支持插件），解压后将其放入jmeter安装目录的jmeter-3.0\lib\ext路径下，然后重启即可。

下载地址：https://jmeter-plugins.org/wiki/PluginsManager/

jmeter的第三方扩展插件功能是很丰富的，也算一定程度上弥补了jmeter作为开源工具的某些不足之处，具体的作用还是需要在实战中摸索实践。。。

（21）jmeter常用插件介绍

jmeter作为一个开源的接口性能测试工具，其本身的小巧和灵活性给了测试人员很大的帮助，但其本身作为一个开源工具，相比于一些商业工具（比如LoadRunner），在功能的全面性上就稍显不足。

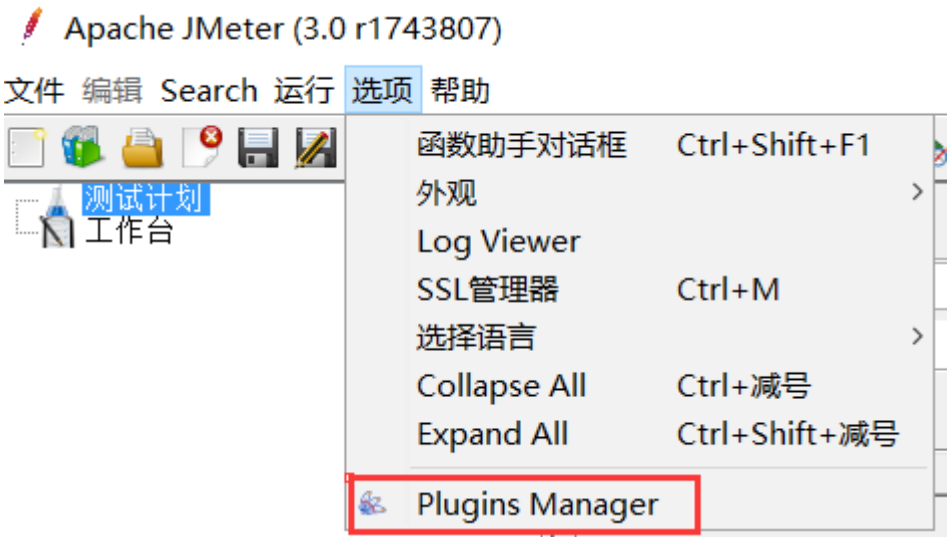
这篇博客，就介绍下jmeter的第三方插件**jmeter-plugins.org**和其中常用的几种插件使用方法。

一、下载安装及使用

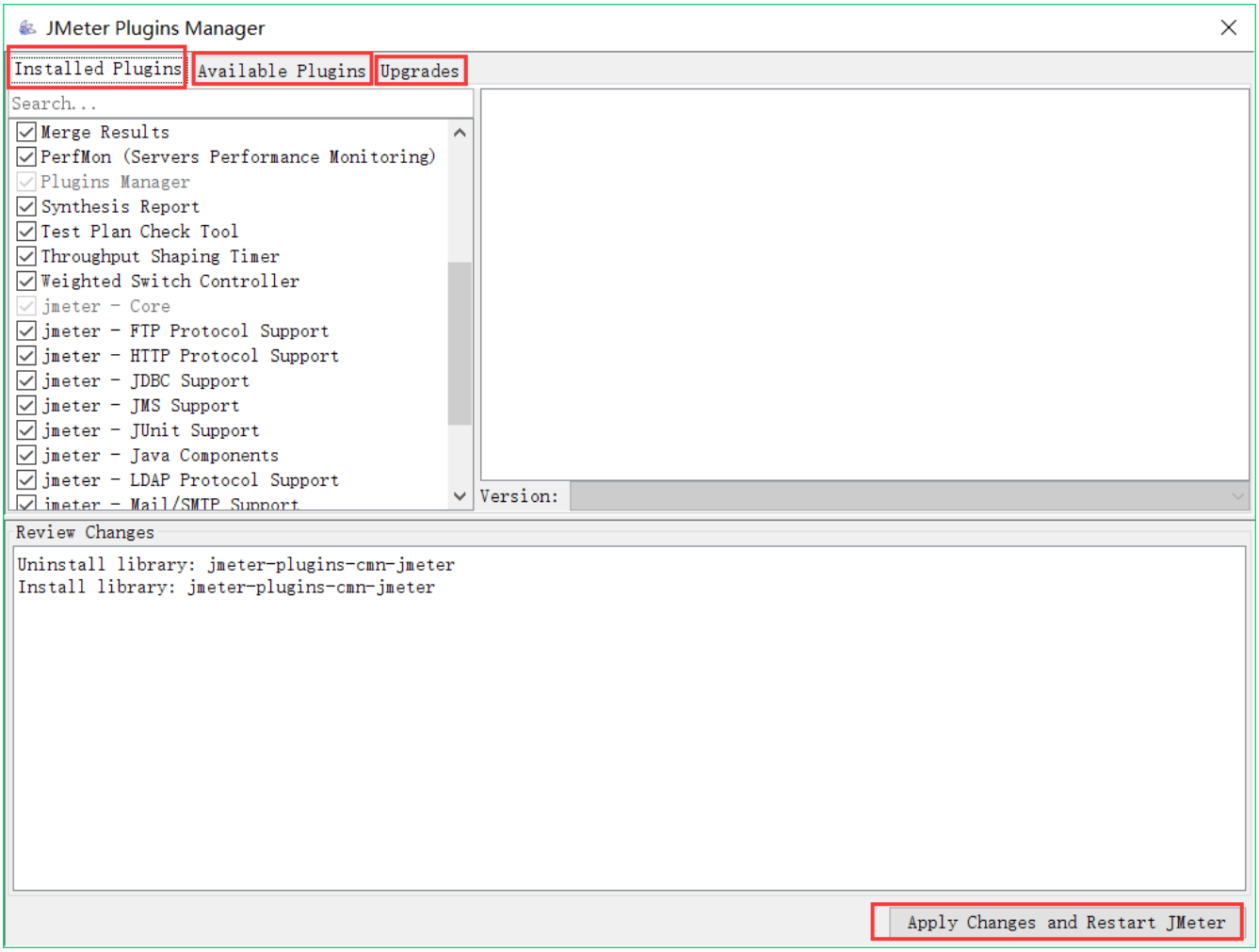
下载地址：jmeter-plugins.org

安装：下载后文件为plugins-manager.jar格式，将其放入jmeter安装目录下的lib/ext目录，然后重启jmeter，即可。

启动jemter，点击选项，最下面的一栏，如下图所示：



打开后界面如下：



Installed Plugins（已安装的插件）：即插件jar包中已经包含的插件，可以通过选中勾选框，来使用这些插件；

Available Plugins（可下载的插件）：即该插件扩展的一些插件，可以通过选中勾选框，来下载你所需要的插件；

Upgrades（可更新的插件）：即可以更新到最新版本的一些插件，一般显示为加粗斜体，可以通过点击截图右下角的Apply Changes and Restart Jmeter按钮来下载更新；

PS：一般不建议进行更新操作，因为最新的插件都有一些兼容问题，而且很可能导致jmeter无法使用（经常报加载类异常）！！！

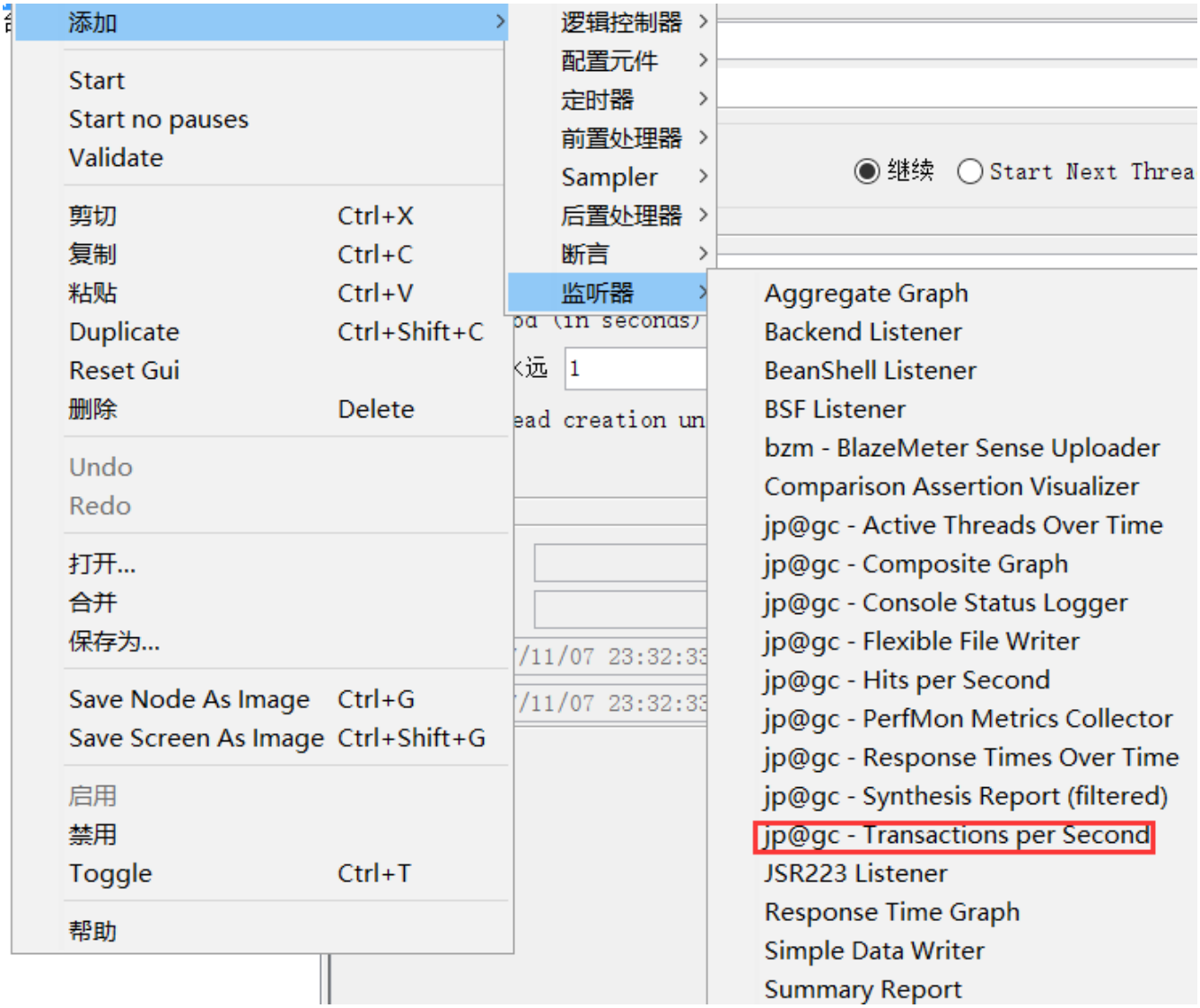
建议使用jmeter最新的3.2版本来尝试更新这些插件。。。

二、Transactions per Second

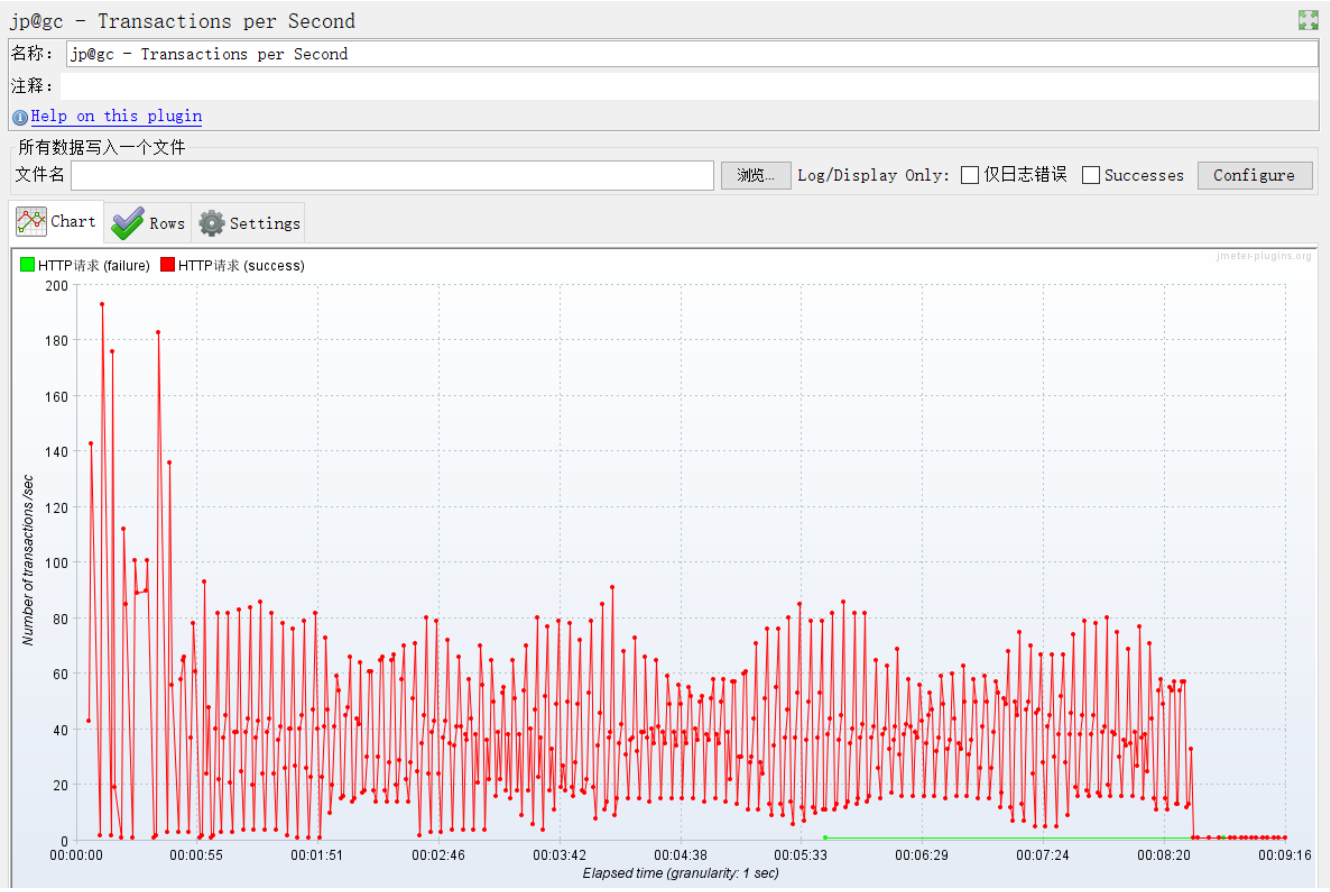
即**TPS：每秒事务数**，性能测试中，最重要的2个指标之一。该插件的作用是在测试脚本执行过程中，监控查看服务器的TPS表现————比如**整体趋势、实时平均值走向、稳定性**等。

jmeter本身的安装包中，监视器虽然提供了比如聚合报告这种元件，也能提供一些实时的数据，但相比于要求更高的性能测试需求，就稍显乏力。

通过上面的下载地址下载安装好插件后，重启jmeter，从监视器中就可以看到该插件，如下图所示：



某次压力测试TPS变化展示图：

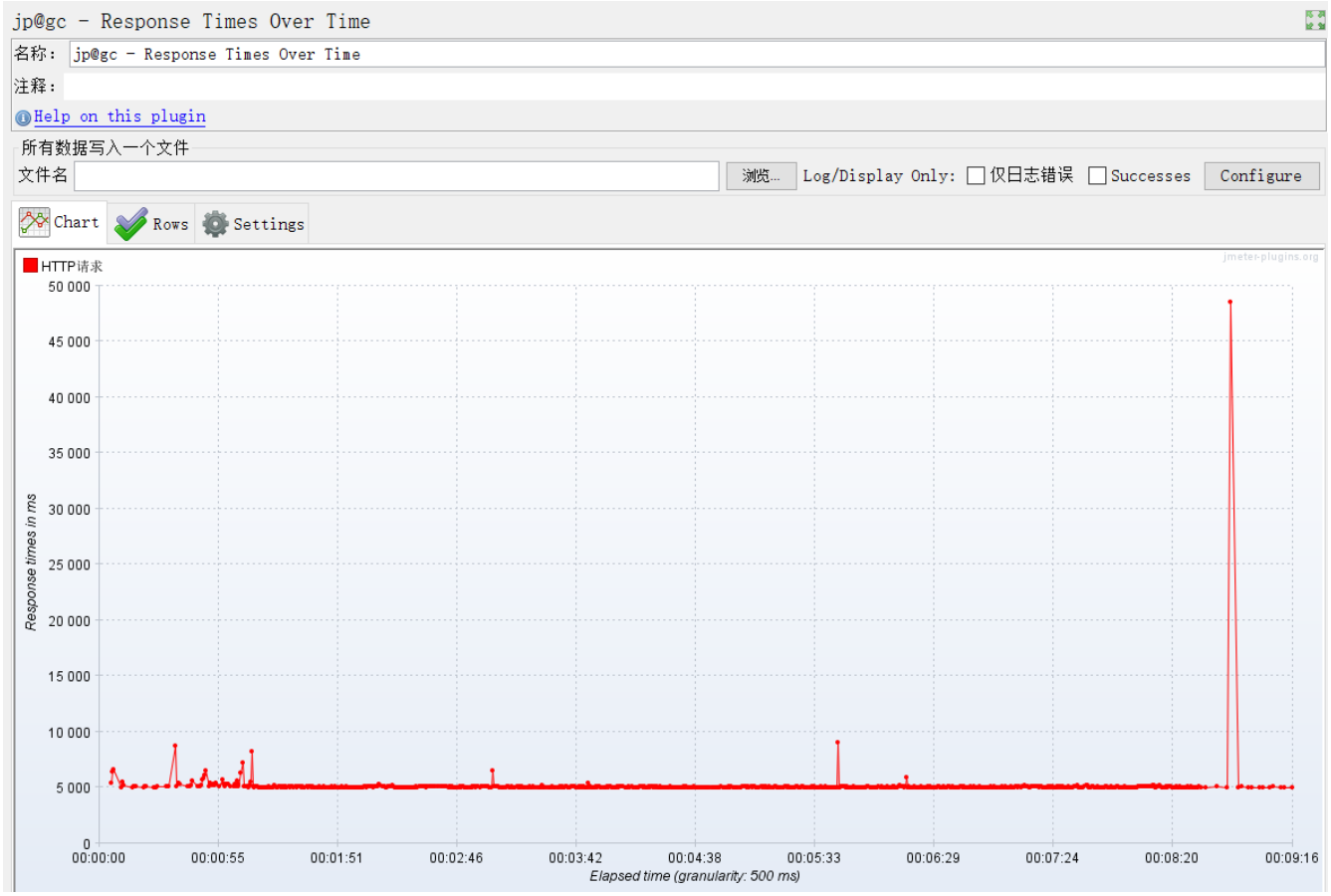


三、Response Times Over Time

即**TRT：事务响应时间**，性能测试中，最重要的两个指标的另外一个。该插件的主要作用是在测试脚本执行过程中，监控查看**响应时间的实时平均值、整体响应时间走向**等。

使用方法如上，下载安装配置好插件之后，重启jmeter，添加该监视器，即可实时看到实时的TRT数值及整体表现。

某次压力测试TRT变化展示图：

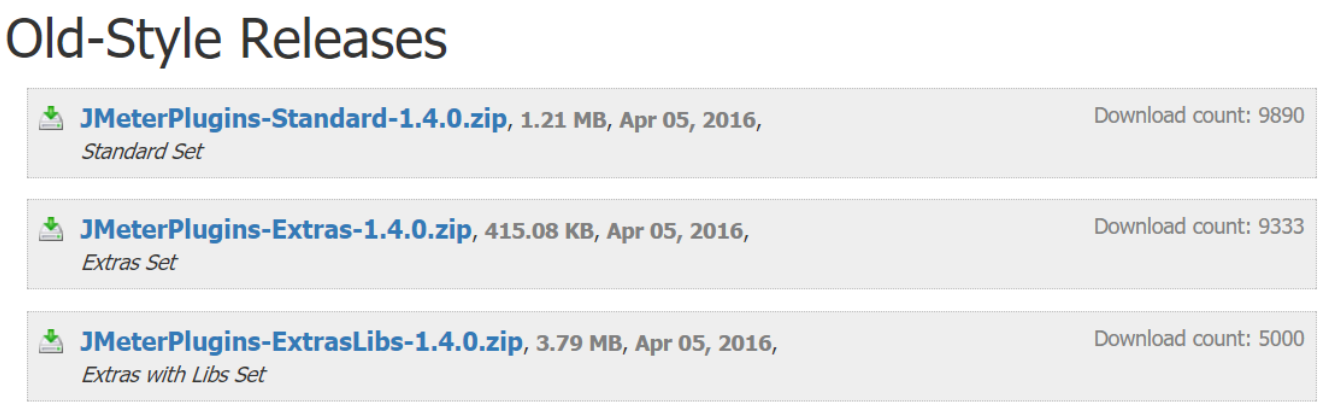


四、PerfMon Metrics Collector

即**服务器性能监控数据采集器**。在性能测试过程中，除了监控TPS和TRT，还需要监控服务器的资源使用情况，比如**CPU、memory、I/O**等。该插件可以在性能测试中实时监控服务器的各项资源使用。

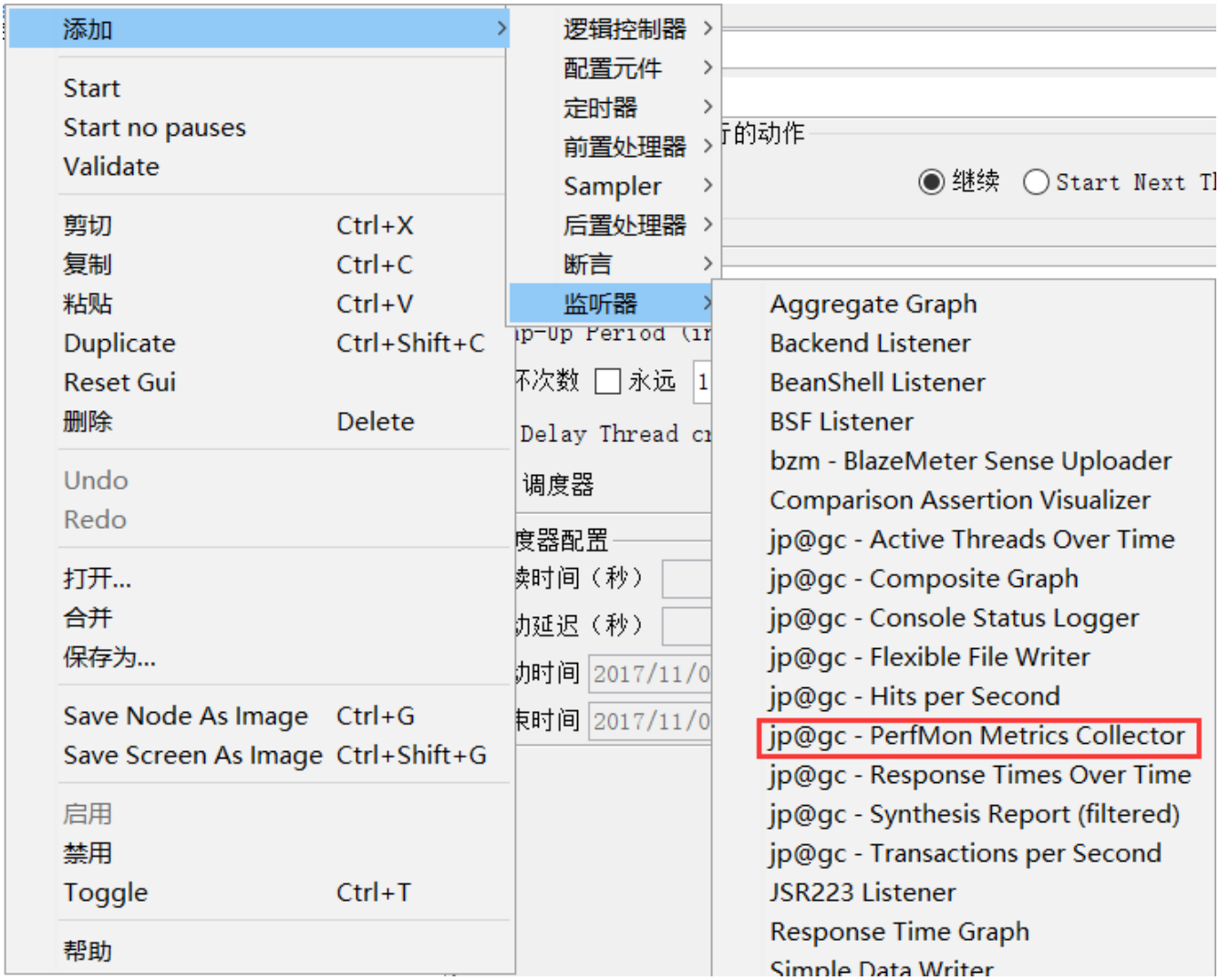
下载地址：http://jmeter-plugins.org/downloads/all/或链接：http://pan.baidu.com/s/1skZS0Zb 密码：isu5

下载界面如下：



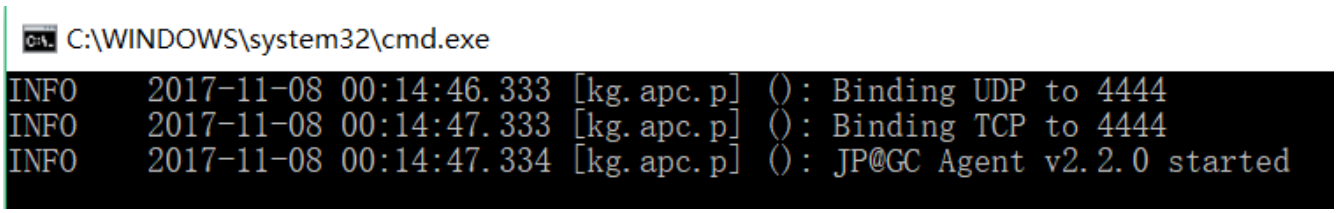
其中JMeterPlugins-Standard和JMeterPlugins-Extras是**客户端**的插件，ServerAgent是**服务端**的插件。

下载成功后，复制**JmeterPlugins-Extras.jar**和**JmeterPlugins-Standard.jar**两个文件，放到jmeter安装文件中的lib/ext中，重启jmeter，即可看到该监视器插件。如下图：



将ServerAgent-2.2.1.jar上传到被测服务器，解压，进入目录，Windows环境，双击ServerAgent.bat启动；linux环境执ServerAgent.sh启动，默认使用4444端口。

如出现如下图所示情况，即表明服务端配置成功：

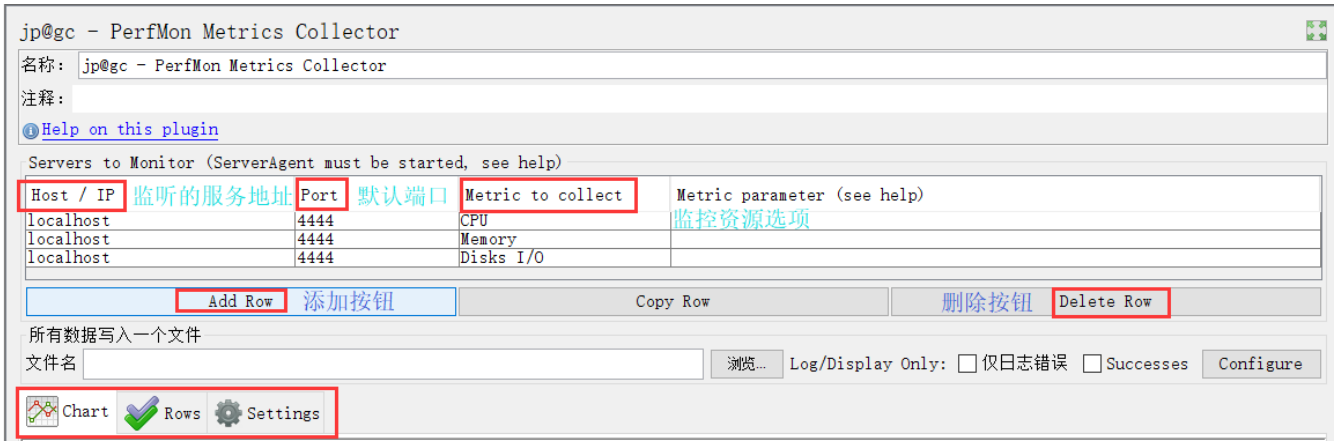


1、服务端启动校验

CMD进入命令框，观察是否有接收到消息，如果有，即表明ServerAgent成功启动。

2、客户端监听测试

给测试脚本中添加jp@gc - PerfMon Metrics Collector监听器，然后添加需要监控的服务器资源选项，启动脚本，即可在该监听器界面看到资源使用的曲线变化。如下图所示：



在脚本启动后，即可从界面看到服务器资源使用的曲线变化，Chart表示主界面显示，Rows表示小界面以及不同资源曲线所代表的颜色，Settings表示设置，可选择自己需要的配置。

PS：注意测试脚本需要持续运行一段时间，才可以看到具体的曲线变化，否则ServerAgent端会断开连接！

上面的几个插件为最常见的一些插件，具体的使用方法请在实践中自行探索。

软件测试交流群：497127619，欢迎加入，谢谢。。。

（22）内存溢出原因及解决方法

jmeter是一个java开发的开源性能测试工具，在性能测试中可支持模拟并发压测，但有时候当模拟并发请求较大或者脚本运行时间较长时，压力机会出现卡顿甚至报异常————内存溢出，

这里就介绍下如何解决内存溢出及相关的知识点。。。

首先来看看我们常说的内存泄漏、内存溢出是什么？

内存泄露是指你的应用使用资源之后没有及时释放，导致应用内存中持有了不需要的资源，这是一种状态描述；

内存溢出是指你应用的内存已经不能满足正常使用了，堆栈已经达到系统设置的最大值，进而导致崩溃，这事一种结果描述；

通常都是由于内存泄露导致堆栈内存不断增大，从而引发内存溢出。

在利用jmeter测试过程中，如果内存溢出的话，一般会出现这个提示：**java.lang.OutOfMemoryError: Java heap space：意思就是堆内存溢出，不够用了。**

说到堆栈内存，顺带简单介绍下**堆栈的相关知识**：

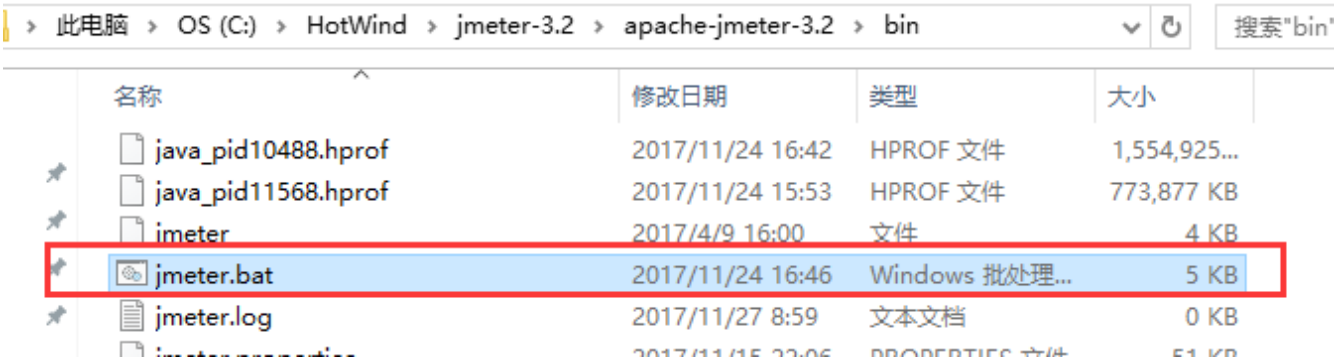
名称	堆（heap）	栈（stack）
定义	一种数据项按序排列的数据结构	
存储类型	函数的参数值、局部变量等值	存放对象
特性	队列优先，先进先出	先进后出OR后进先出
内存分配原理	new分配的内存，速度较慢，存放在二级缓存中，生命周期由垃圾回收算法决定	使用一级缓存，由系统自动分配
优点	动态分配内存，垃圾回收器自动回收，空间较大，较灵活，便于使用	调用完毕立即释放，速度较快
缺点	存取速度较慢（堆大小受限于系统中的虚拟内存）	存在栈中的数据大小和生存周期必须是确定的，缺乏灵活性，且数据不可共享

更详细的内容请参照这里：[百度百科：堆栈]

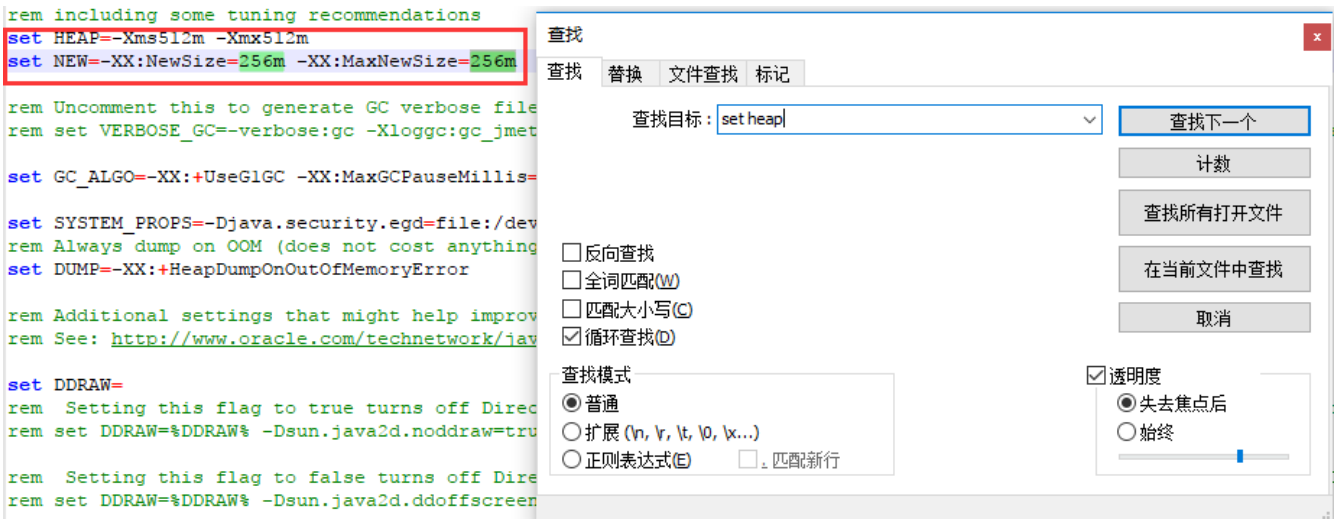
内存溢出解决方法：调整堆内存大小

步骤：

1、打开jmeter安装文件（可以用notepad++打开），bin目录下的jmeter.bat文件：



2、找到set HEAP开头的内容，根据具体需要修改堆（heap）值大小，以及NEW分配的内存值大小：



这里默认值为：

set HEAP=-Xms512m -Xmx512m set NEW=-XX:NewSize=256m -XX:MaxNewSize=256m

将其修改为：

set HEAP=-Xms512m -Xmx4096m set NEW=-XX:NewSize=256m -XX:MaxNewSize=512m

注意：一般而言，堆的最大值不要超过物理内存的一半，否则容易导致jmeter运行变慢、卡顿甚至内存溢出（因为java本身的垃圾回收机制是动态分配内存，

调整时候其本身会占用很多内存），NEW分配的内存，不宜太大！

3、修改完成后，关闭文件，重启jmeter既可以：

PS：当需要模拟的线程数较大时，就需要根据具体情况采用分布式压测的方式了，这种修改堆大小的方法只适用一部分情况，并不是万能的！

（23）jmeter分布式测试

jmeter用了一年多，也断断续续写了一些相关的博客，突然发现没有写过分布式测试的一些东西，这篇博客就介绍下利用jmeter做分布式测试的一些技术点吧，权当参考。。。

关于jmeter的介绍和元件作用，之前的博客介绍过，很多其他同行的博客也够详细的，这里不做介绍，对jmeter不甚了解的可以参考之前的博客：jmeter：菜鸟入门到进阶系列

jmeter官方文档：用户手册

jmeter源码：Apache JMeter

一、为什么要使用分布式测试

按照一般的压力机配置，jmeter的GUI模式下（Windows），最多支持300左右的模拟请求线程，再大的话，容易造成卡顿、无响应等情况，这是限于jmeter其本身的机制和硬件配置。

有时候为了尽量模拟业务场景，需要模拟大量的并发请求，这个时候单台压力机就显得有心无力。针对这个情况，jmeter的解决方案是支持分布式压测，即将大量的模拟并发分配给

多台压力机，来满足这种大流量的并发请求场景。

二、分布式压测的原理

1、分布式测试中，选择一台作为管理机（Contorller），其他的机器作为测试执行的代理机（Agent）；

2、执行测试时，由Contorller通过命令行将测试脚本发给Agent，然后Agent执行测试（不需要启动GUI），同时将测试结果发送给Contorller；

3、测试完成，可以在Contorller上的监听器里面看到Agent发来的测试结果，结果为多个Agent测试结果汇总而成；

三、分布式设置步骤

1、修改Contorller配置

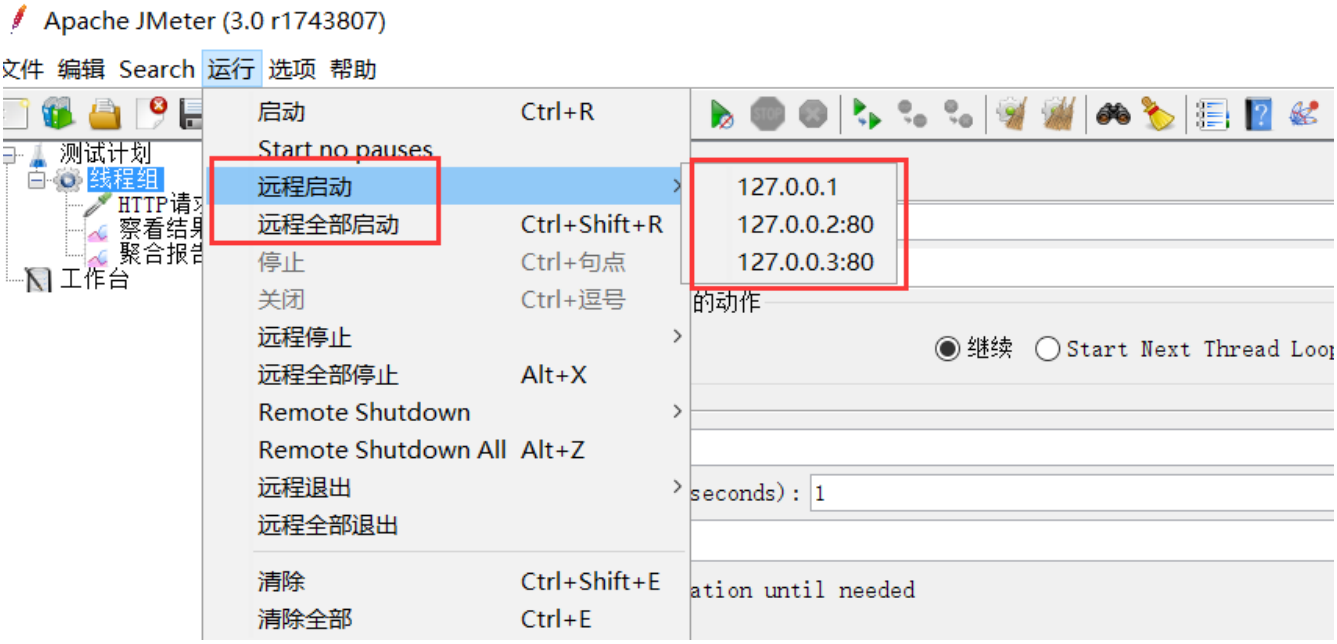
打开Contorller机下jmeter安装文件下的bin目录：jmeter.properties文件，搜索remote_hosts=127.0.0.1，将Agent机的IP和端口写在后面，比如：

remote_hosts=127.0.0.1,127.0.0.2:80,127.0.0.3:80

其中127.0.0.2和127.0.0.3为Agent机的IP，每个Agent机之间用英文半角逗号隔开，修改保存。

2、启动jemter

启动jmeter后，设置线程组、配置元件、取样器、监听器等原件，点击“运行-远程启动”：



可以选择远程启动一个Agent机，或者选择远程全部启动，这样，就可以进行分布式测试了。

PS：上面的例子中，127.0.0.2和127.0.0.3为举例说明，具体实践请修改为对应的Agent机IP以及端口。

四、注意事项

1、保持Contorller和Agent机器的JDK、jmeter以及插件等配置版本一致；

2、如果测试数据有用到CSV或者其他方式进行参数化，需要将data pools在每台Agent上复制一份，且读取路径必须保持一致；

3、确保Contorller和Agent机器在同一个子网里面；

4、检查防火墙是否被关闭，端口是否被占用（防火墙会影响脚本执行和测试结构收集，端口占用会导致Agent机报错）；

5、分布式测试中，通过远程启动代理服务器，默认查看结果树中的响应数据为空，只有错误信息会被报回；

6、如果并发较高，建议将Contorller机设置为只启动测试脚本和收集汇总测试结果，在配置文件里去掉Contorller机的IP；

7、分布式测试中，如果1S启动100个模拟请求，有5个Agent机，那么需要将脚本的线程数设置为20，否则模拟请求数会变成500，和预期结果相差太大。

（24）dubbo接口测试

最近工作中接到一个需求，需要对一个MQ消息队列进行性能测试，测试其消费能力，开发提供了一个dubbo服务来供我调用发送消息。

这篇博客，介绍下如何利用jmeter来测试dubbo接口，并进行性能测试。。。

一、Dubbo简介

dubbo是一个分布式服务框架，致力于提供高性能和透明化的RPC远程服务调用方案，以及SOA服务治理方案。其核心部分包含如下几点：

1、远程通讯：提供对多种基于长连接的NIO框架抽象封装，包括多种线程模型，序列化，以及“请求-响应”模式的信息交换方式；

2、集群容错：提供基于接口方法的透明远程过程调用，包括多协议支持，以及软负载均衡，失败容错，地址路由，动态配置等集群支持；

3、自动发现：基于注册中心目录服务，使服务消费方能动态的查找服务提供方，使地址透明，使服务提供方可以平滑增加或减少机器；

4、dubbo简化模型

二、Dubbo插件

1、jmeter本身并不支持dubbo接口的测试，需要下载第三方插件，然后将jar包放入${JMETER_HOME}\lib\ext路径下，重启即可。

插件下载地址：jmeter-plugins-dubbo

2、选择自己需要的插件，为了适配性考虑，建议1.3.2及以上的版本。

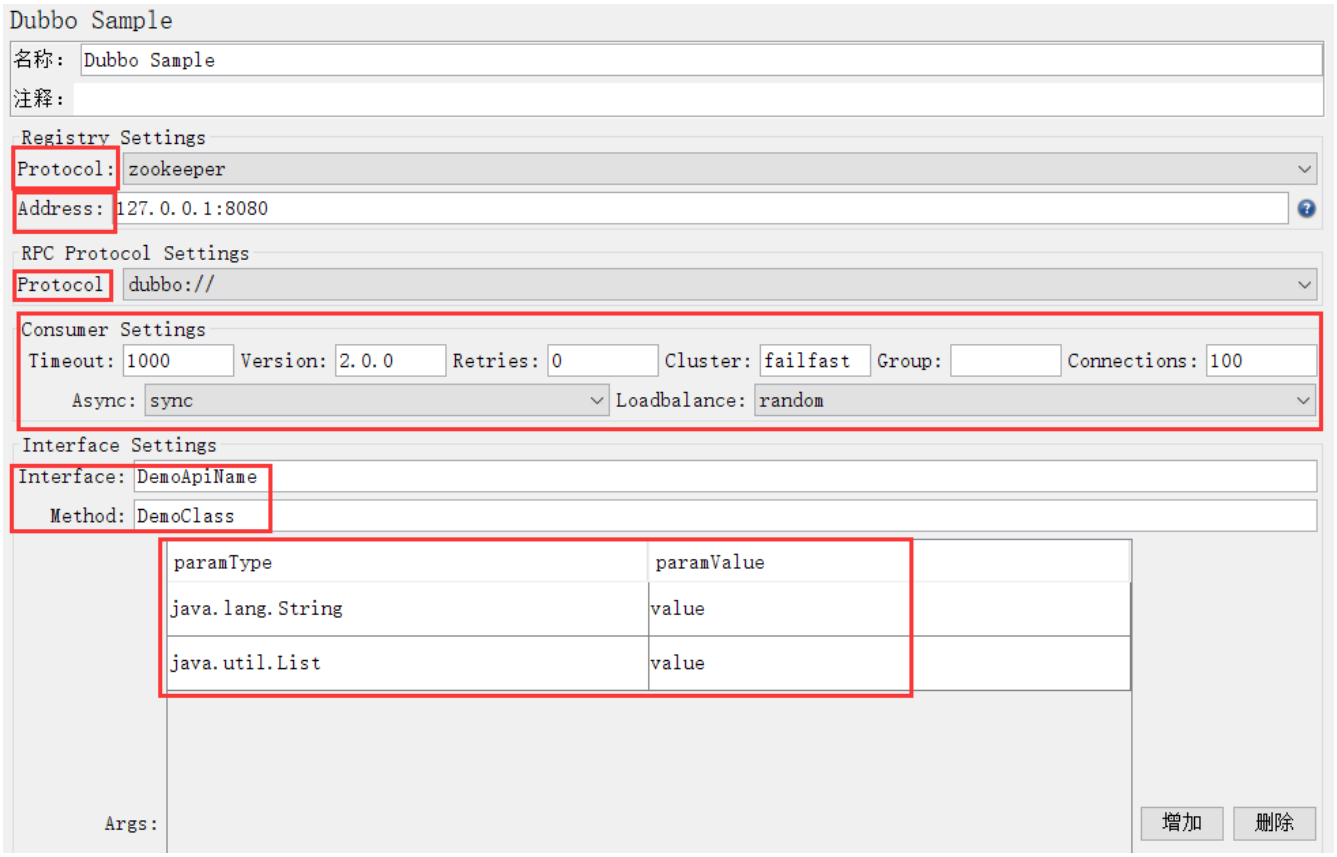
各版本更新说明：changelog

3、参数说明

简单参数对照表、复杂参数对照表

三、Dubbo Sample

启动jmeter，添加线程组→Sampler→Listener，dubbo-sample界面如下：



各参数说明如下：

Protocol：注册协议，包括zookeeper、multicast、Redis、simple；

Address：注册地址，dubbo服务的IP+Port：

①、当使用zk，address填入zk地址，集群地址使用","分隔；

②、使用dubbo直连，address填写直连地址和服务端口；

Protocol：使用的dubbo协议，包括dubbo、rmi、hessian、webservice、memcached、redis，根据自己的协议类型选择对应的选项即可；

Timeout：请求超时时间，单位ms，根据dubbo具体配置填写；

Version：版本，dubbo不同版本之间差异较大，不同版本之间不能互相调用，这里指定dubbo版本，是为了方便识别和说明；

Retries：异常重试次数（类似这种分布式服务通信框架，大多都有重试机制，是为了保证事务成功率）；

Cluster：集群类型，包括failover、failfast、failsafe、failback、failking；

Group：组类型，如果有的话，根据配置填写即可；

Connections：连接数，同上，根据配置填写；

Async：服务处理类型，包括sync（同步）、async（异步），根据配置填写；

Loadbalance：负载均衡策略，包括random（随机）、roundrobin（轮询）、leastactive（最少活跃数）、consistenthash（一致性哈希）；

Interface：接口名（因为dubbo服务大多是开发根据规范自行命名的，因此这里需要填写完整的接口名+包名）；

Method：当前接口下的方法名，按照开发提供的API文档填写即可；

Args：接口报文，根据API文档填写，如上图所示，添加输入行，输入对应的参数类型和值即可（参数类型和值如何定义填写，请参考上面的链接）；

①、paramType：参数支持任何类型，包装类直接使用`java.lang`下的包装类，小类型使用：`int``、``float``、``shot``、``double``、``long``、``byte``、``boolean``、``char`，自定义类使用类完全名称；

②、paramValue：基础包装类和基础小类型直接使用值，例如：int为1，boolean为true等，自定义类与`List`或者`Map`等使用json格式数据；

以上即为利用jmeter的dubbo插件进行dubbo接口的测试，当然，同样可以进行性能测试，更多使用方式请自行探索。。。

（25）linux环境运行jmeter并生成报告

jmeter是一个java开发的利用多线程原理来模拟并发进行性能测试的工具，一般来说，GUI模式只用于创建脚本以及用来debug，执行测试时建议使用非GUI模式运行。

这篇博客，介绍下在linux环境利用jmeter进行性能测试的方法，以及如何生成测试报告。。。

一、为什么要非GUI模式运行

jmeter是java语言开发，实际是运行在JVM中的，GUI模式运行需要耗费较多的系统资源，一般来说，GUI模式要占用10%-25%的系统资源。

而使用非GUI模式（即linux或dos命令）可以降低对资源的消耗，提升单台负载机所能模拟的并发数。

启动jmeter，提醒如下：

```
================================================================================
Don't use GUI mode for load testing, only for Test creation and Test debugging !
For load testing, use NON GUI Mode:
   jmeter -n -t [jmx file] -l [results file] -e -o [Path to output folder]
& adapt Java Heap to your test requirements:
   Modify HEAP="-Xms512m -Xmx512m" in the JMeter batch file
================================================================================
```

二、环境准备

1、安装JDK

关于如何在linux环境安装JDK，可参考我之前的博客：linux下安装JDK

2、安装jmeter

官方地址：Download Apache JMeter

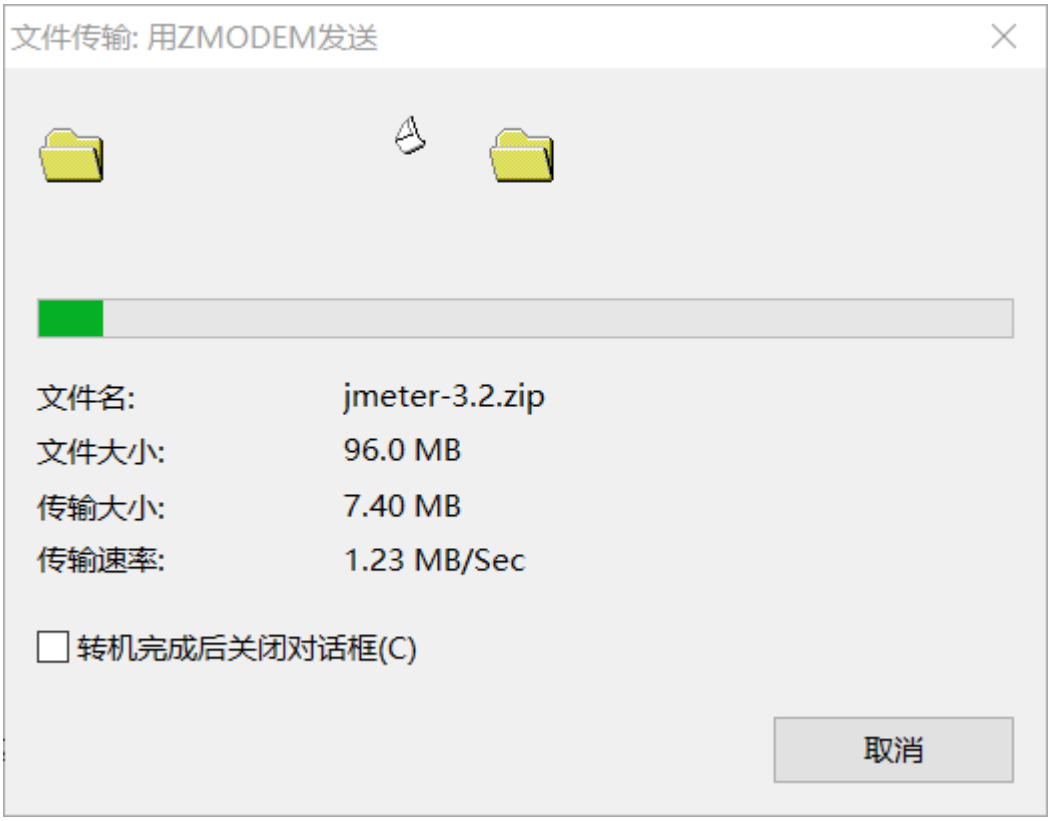
下载压缩包，然后将安装包上传至linux服务器，一般有以下2种方式：

①、通过FileZilla或其他类似工具上传至linux服务器；

②、直接将zip文件拖至linux服务器；

方法如下：

输入命令 yum install -y lrzsz ，安装linux下的上传和下载功能包，然后将jmeter压缩包拖进去即可，示例如下：



然后输入命令 unzip jmeter-3.2.zip ， 进行解压。

3、配置环境变量

输入命令 vim /etc/profile ， 在最下面添加如下内容：

```
export JAVA_HOME=/usr/java/1.8.0_181
export CLASSPATH=.:JAVA_HOME/lib/dt.jar:$JAVA_HOME.lib/tools.jar:$JRE_HOME.lib
export PATH=$PATH:$JAVA_HOME/lib
export PATH=/jmeter/apache-3.2/bin/:$PATH
export JMETER_HOME=/jmeter/jmeter-3.2
export
CLASSPATH=$JMETER_HOME/lib/ext/ApacheJMeter_core.jar:$JMETER_HOME/lib/jorphan.jar:$CLASSPAT
H
export PATH=$JMETER_HOME/bin:$PATH
```

保存后，输入命令 source /etc/profile ,使修改的配置生效。

4、授予权限

在执行jmeter脚本执行，首先要确保监控工具、jmeter以及相关的文件有相应的权限，否则会报错，常见的报错如下：

①、文件没有权限

②、无法打开目录下的文件

③、编码格式错误

查看文件或工具是权限的命令如下：

```
# 查看当前目录下所有文件的权限
ls -l
# 查看当前目录下所有文件的权限
ll
# 查看某个文件的权限
ls -l filename
# 查看某个目录的权限
ls ld /path
```

5、linux文件颜色代表的含义

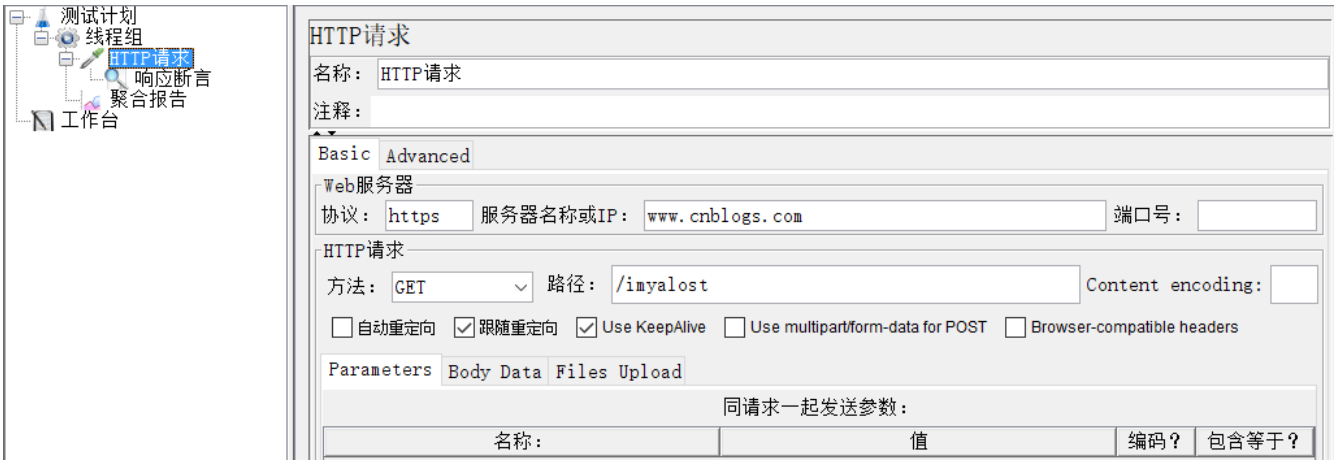
在linux中，不同颜色的文件代表不同的含义，下面是linux中不同颜色的文件代表的含义：

```
# 白色：普通的文件
# 蓝色：目录
# 绿色：可执行的文件
# 红色：压缩文件或者包文件
# 青色：连接文件
# 黄色：设备文件
# 灰色：其他的文件
```

三、运行jmeter

1、启动jmeter，创建脚本

这里以访问我博客首页为例：



脚本保存为test.jmx，然后将文件上传至linux服务器。

2、运行脚本

将脚本上传至linux服务器，然后进入jmeter的bin目录下，输入命令 jmeter -n -t test.jmx -l test.jtl ，运行jmeter脚本。

PS：常用命令解析：

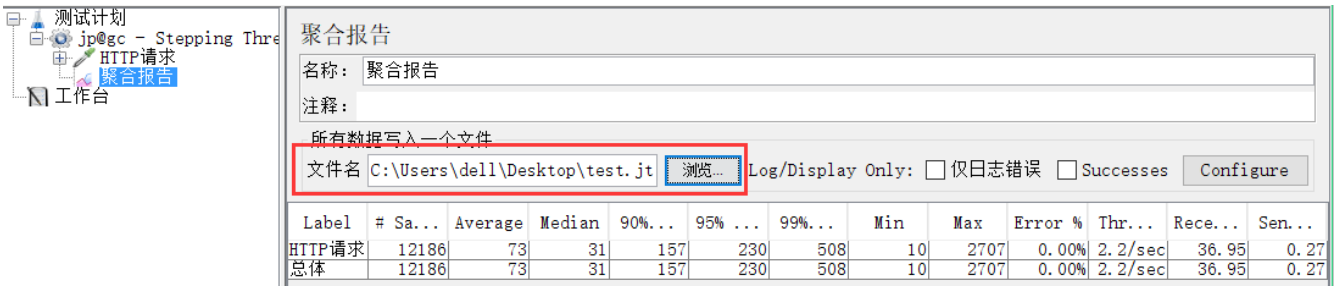
```
# 常见命令说明
-h 帮助：打印出有用的信息并退出
-n 非 GUI 模式：在非 GUI 模式下运行 JMeter
-t 测试文件：要运行的 JMeter 测试脚本文件
-l 日志文件：记录结果的文件
-r 远程执行：启动远程服务
-H 代理主机：设置 JMeter 使用的代理主机
-P 代理端口：设置 JMeter 使用的代理主机的端口号
```

运行结果如下图：

```
[root@izbp1jbg0c2bbcmcba0exoz bin]# ./jmeter -n -t /root/script/test.jmx -l /root/test.jtl
Creating summariser <summary>
Created the tree successfully using /root/script/test.jmx
Starting the test @ Fri Oct 19 00:28:20 CST 2018 (1539880100682)
Waiting for possible Shutdown/StopTestNow/Heapdump message on port 4445
summary +    714 in 00:00:09 =   79.6/s Avg:    35 Min:    10 Max:   530 Err:     0 (0.00%) Active: 4 Started: 4 Finished: 0
summary +   6205 in 00:00:30 =  207.0/s Avg:    50 Min:    10 Max:   557 Err:     0 (0.00%) Active: 16 Started: 16 Finished: 0
summary =   6919 in 00:00:39 =  177.6/s Avg:    49 Min:    10 Max:   557 Err:     0 (0.00%)
summary +   1438 in 00:00:06 =  231.3/s Avg:    75 Min:    11 Max:   348 Err:     0 (0.00%) Active: 0 Started: 20 Finished: 20
summary =   8357 in 00:00:45 =  185.0/s Avg:    53 Min:    10 Max:   557 Err:     0 (0.00%)
Tidying up ...    @ Fri Oct 19 00:29:06 CST 2018 (1539880146220)
... end of run
```

3、查看测试报告

启动jmeter，新建一个线程组，添加所需的监听器，导入脚本运行产生的.jtl文件，如下：



以上，即为在linux环境中运行jmeter脚本进行压测，并生成测试报告的的过程，具体操作，请自行实践，本文仅供参考。。。

（26）jmeter生成HTML格式性能测试报告

性能测试工具Jmeter由于其体积小、使用方便、学习成本低等原因，在现在的性能测试过程中，使用率越来越高，但其本身也有一定的缺点，比如提供的测试结果可视化做的很一般。

不过从3.0版本开始，jmeter引入了Dashboard Report模块，用于生成HTML类型的可视化图形报告（3.0版本的Dashboard Report模块会中文乱码，因此建议使用3.0以上的版本）。

这篇博客，简单介绍下在利用jmeter进行性能测试时，生成HTML的可视化测试报告。。。

一、生成HTML测试报告的两种方式

1、利用已有.jtl文件生成报告

之前的博客介绍过如何在linux环境运行jmeter并生成报告，如果已经有经过测试生成的.jtl文件，可以利用该文件直接生成HTML可视化测试报告。

进入jmeter的bin目录下，输入如下命令：

```
jmeter -g test.jtl -o /path
# -g：后跟test.jtl文件所在的路径
# -o：后跟生成的HTML文件存放的路径
```

PS：如果是在Windows环境命令行运行，必须指定生成的HTML文件存放文件夹，否则会报错；如果是linux环境，如指定路径下不存在该文件夹，会生成对应的文件夹存放报告文件！

2、无.jtl文件生成测试报告

如果还未生成.jtl文件，则可以通过如下命令，一次性完成测试执行和生成HTML可视化报告的操作，进入jmeter的bin目录下，输入如下命令：

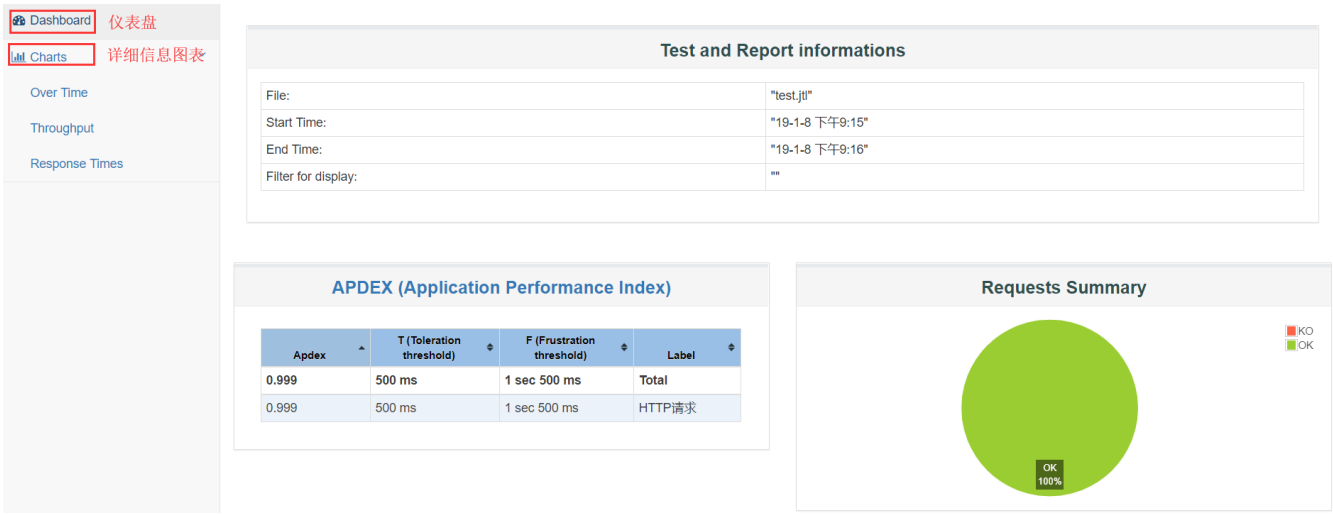
```
jmeter -n -t test.jmx -l test.jtl -e -o /path
# -n：以非GUI形式运行Jmeter
# -t：source.jmx 脚本路径
# -l：result.jtl 运行结果保存路径（.jtl），此文件必须不存在
# -e：在脚本运行结束后生成html报告
# -o：用于存放html报告的目录
```

我本地Windows环境执行截图如下：

```
E:\jmeter-3.2\bin>jmeter -n -t C:\Users\dell\Desktop\DEMO\test.jmx -l C:\Users\dell\Desktop\DEMO\test.jtl -e -o C:\Users\dell\Desktop\DEMO\test
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/E:/jmeter-3.2/lib/log4j-slf4j-impl-2.8.1.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/E:/jmeter-3.2/lib/ext/jmeter-plugins-dubbo-1.3.5-jar-with-dependencies.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Creating summariser <summary>
Created the tree successfully using C:\Users\dell\Desktop\DEMO\test.jmx
Starting the test @ Tue Jan 08 21:15:34 CST 2019 (1546953334889)
Waiting for possible Shutdown/StopTestNow/Heapdump message on port 4445
summary +   1281 in 00:00:25 =   51.9/s Avg:   111 Min:    44 Max:  1171 Err:     0 (0.00%) Active: 10 Started: 10 Finished: 0
summary +   2442 in 00:00:21 =  116.9/s Avg:   126 Min:    47 Max:   561 Err:     0 (0.00%) Active: 0 Started: 20 Finished: 20
summary =   3723 in 00:00:46 =   81.7/s Avg:   121 Min:    44 Max:  1171 Err:     0 (0.00%)
Tidying up ...    @ Tue Jan 08 21:16:20 CST 2019 (1546953380889)
... end of run
```

PS：（linux系统和windows系统命令一样）需要注意的是，生成的**.jtl**文件路径下，不能存在同名的**.jtl**文件，否则会执行失败。

执行完毕后，用浏览器打开生成的文件目录下的index文件，效果展示如下：



二、图表信息详解

测试报告分为两部分，Dashboard和Charts，下面分开解析。

1、Dashboard（概览仪表盘）

①、Test and Report informations

Test and Report informations

测试和报告信息

File:	生成报告的源文件	"test.jtl"
Start Time:	开始时间	"19-1-8 下午9:15"
End Time:	结束时间	"19-1-8 下午9:16"
Filter for display:		""

②、APDEX (应用性能指标)

关于APDEX的相关信息，请参考这里：应用性能指标；英文原文，参考这里：Apdex-Wikipedia

APDEX (Application Performance Index)

满意度，范围是0-1，1最高

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
0.999	500 ms	1 sec 500 ms	Total
0.999	500 ms	1 sec 500 ms	HTTP请求

③、Requests Summary

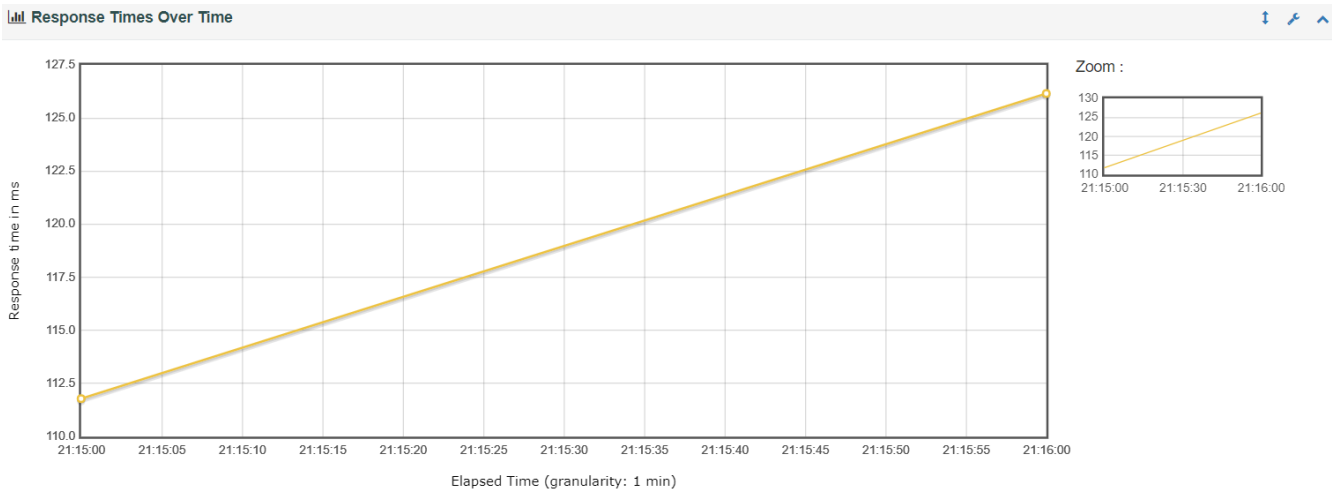
2、Charts（详细信息图表）

PS：由于详细信息图表有点多，这里我挑几个性能测试过程中比较关键的图表解析！

Over Time

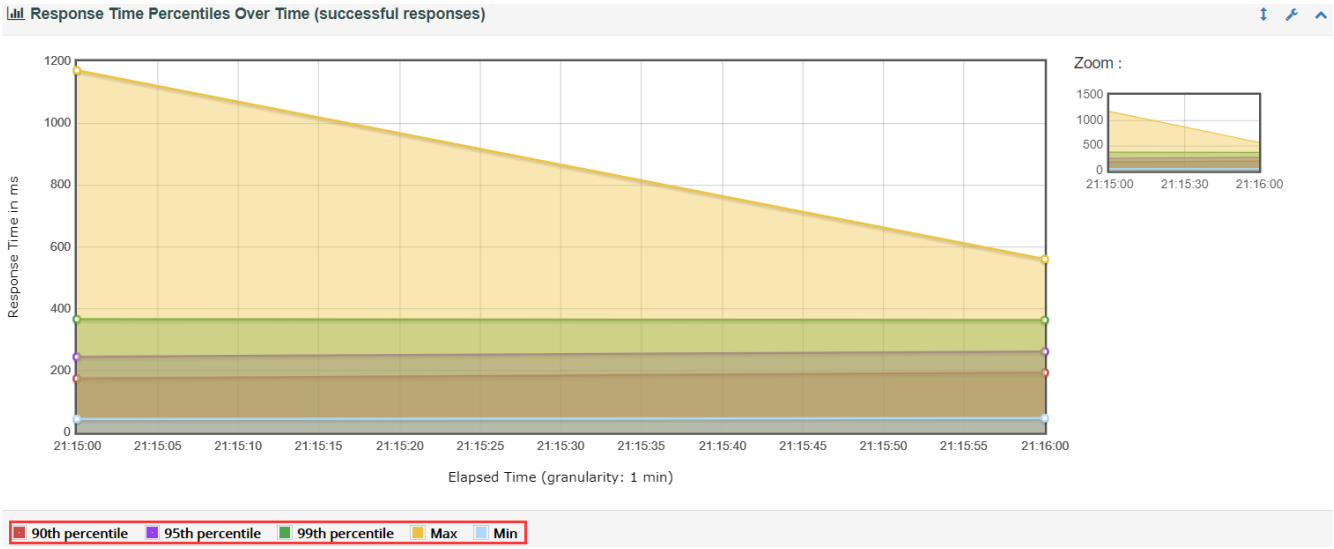
①、Response Times Over Time（脚本运行期间的响应时间变化趋势图）

说明：可以根据响应时间和变化和TPS以及模拟的并发数变化，判断性能拐点的范围。



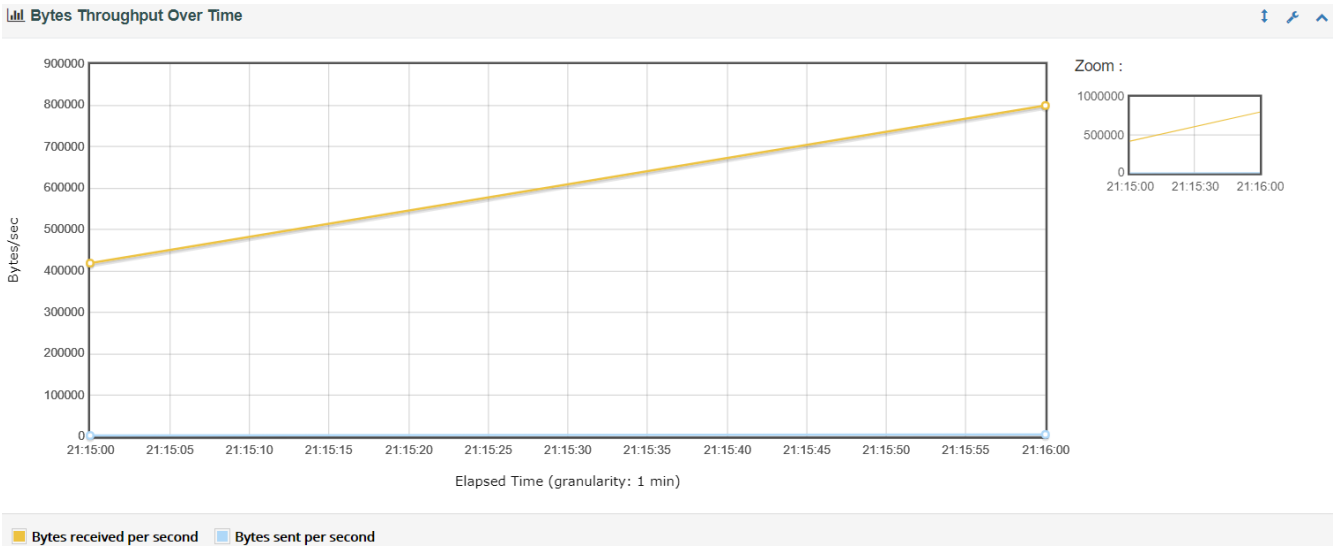
②、 Response Time Percentiles Over Time (successful responses)

说明：脚本运行期间成功的请求响应时间百分比分布图，可以理解为聚合报告里面不同%的数据，图形化展示的结果。



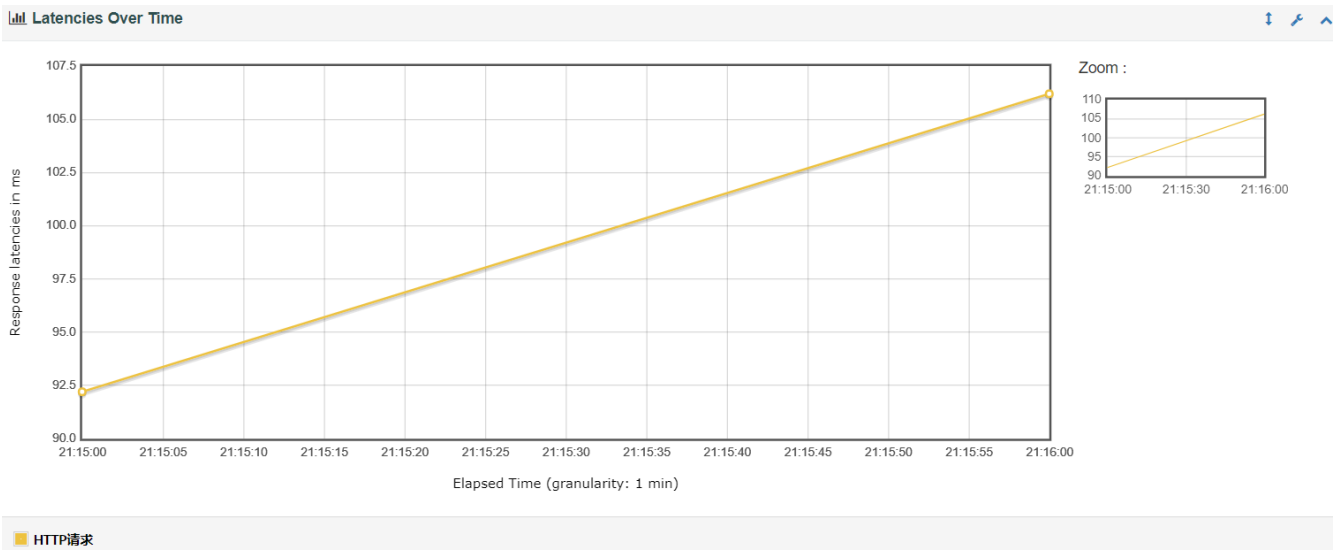
③、Bytes Throughput Over Time（脚本运行期间的吞吐量变化趋势图）

说明：在容量规划、可用性测试和大文件上传下载场景中，吞吐量是很重要的一个监控和分析指标。



④、 Latencies Over Time（脚本运行期间的响应延时变化趋势图）

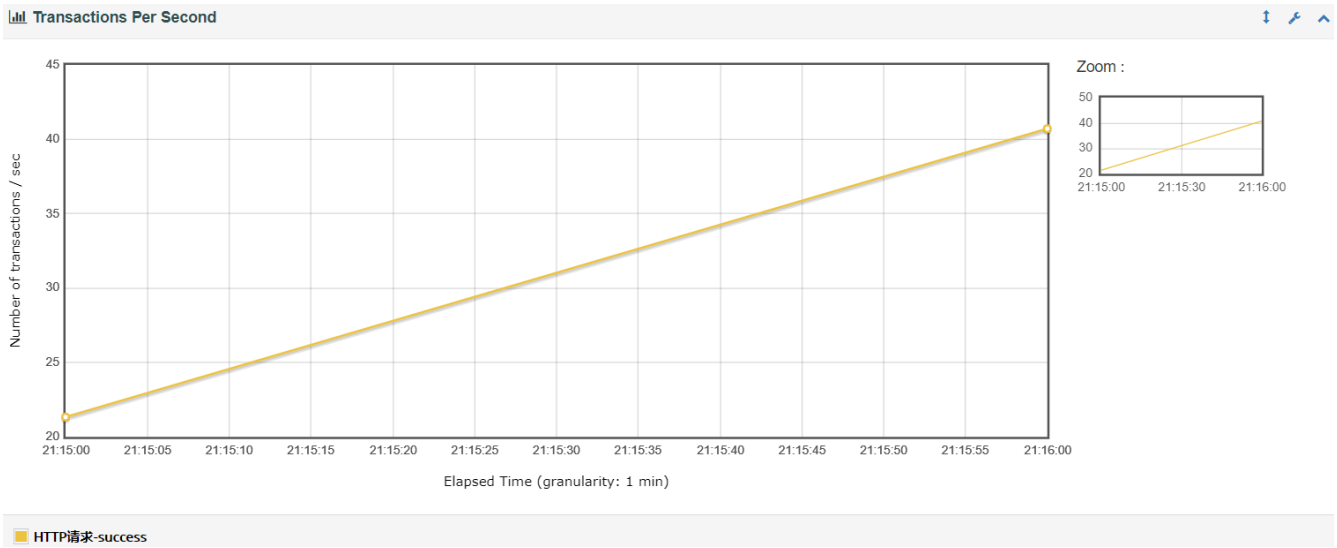
说明：在高并发场景或者强业务强数据一致性场景，延时是个很严重的影响因素。



Throughput

①、Transactions Per Second（每秒事务数）

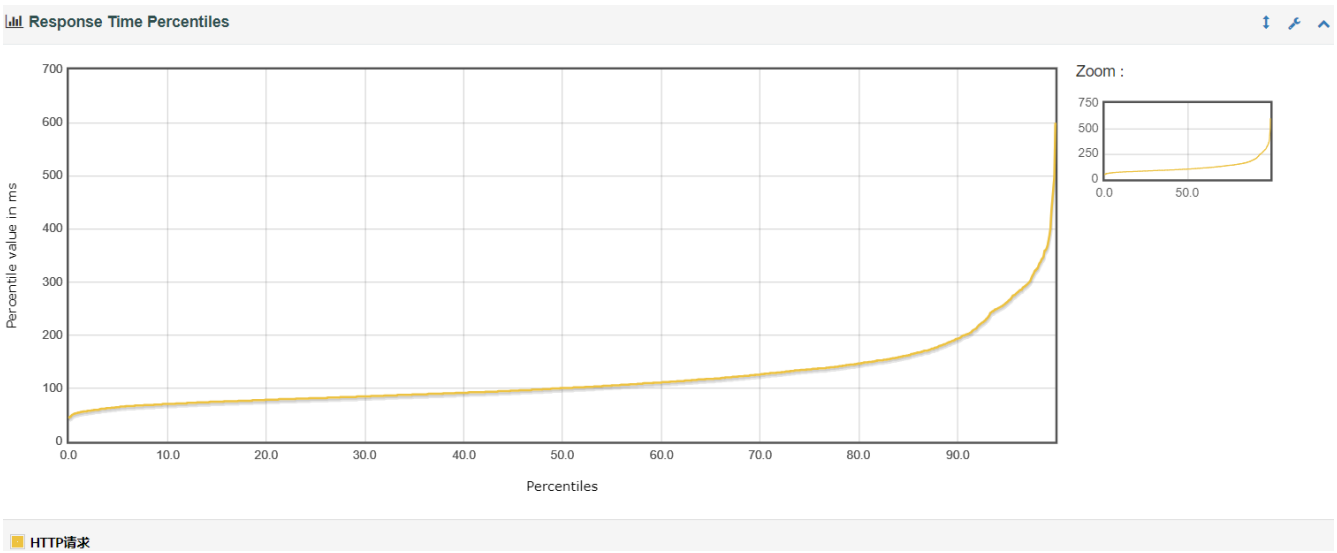
说明：每秒事务数，即TPS，是性能测试中很重要的一个指标，它是用来衡量系统处理能力的一个重要指标。



Response Times

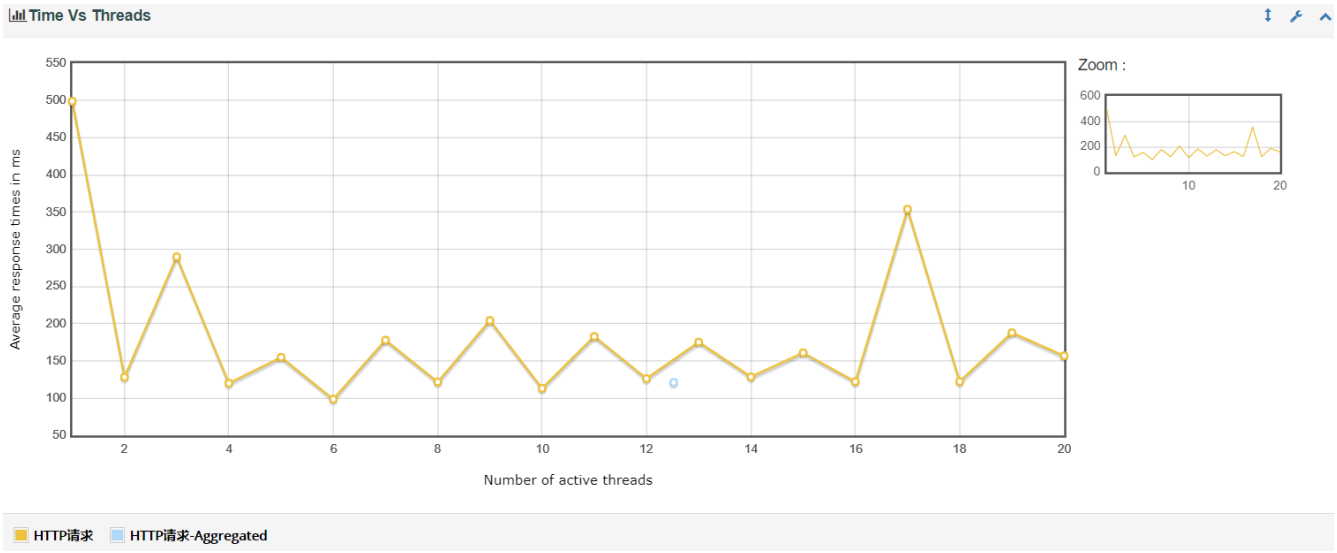
①、 Response Time Percentiles（响应时间百分比分布曲线图）

说明：即响应时间在某个范围内的请求在所有请求数中所占的比率，相比于平均响应时间，这个值更适合用来衡量系统的稳定性。



②、Time Vs Threads（平均响应时间和线程数的对应变化曲线）

说明：可以通过这个对应的变化曲线来作为确定性能拐点的一个参考值。



以上内容，即为jmeter生成HTML格式测试报告的方法以及报告内容解析，个人觉得这个图表可以进行再次开发，变得更灵活和易用。。。