JMETER入门指南

1. jmeter概述

1.1 什么是jmeter

Apache JMetier是Apache组织开发的基于Java的压力测试工具。一款非常优秀的开源的性能测试工具。

1.2 jmeter适用场景

适用的测试领域：

用于对软件做压力测试，它最初被设计用于Web应用测试，但后来扩展到其他测试领域。它可以用于测试静态和动态资源，例如静态文件、 Java小服务程序、 CGI脚本、Java对象、数据库、 FTP服等器，等等。 JMeter可以用于对服务器、网络或对象模拟巨大的负载，来自不同压力类别下测试它们的强度和分析整体性能。

JMetor能够对应用程序做功能/回归测试，通过创建带有断言的脚本来验证你的程序返回了你期望的结果。为了最大限度的灵活性， JMeter允许使用正则表达式创建断言。

Apache jmeter可以用于对静态的和动态的资源（文件， Servlet» Perl脚本,java对象，数据库和查询，FTP服务器等等）的性能进行测试。

接口测试

数据库压力测试 一

批量产生测试数据

2. jmeter安装配置

2.1 下载安装

官网下载地址：http://jmeter.apache.org/download_jmeter.cgi

jmeter是免安装的，下载解压配置环境变量即可使用。

解压之后状态

2.2 环境配置（可不配）

jdk1.8环境配置：Java -version 查看jdk版本。

jmeter环境配置（不配置不影响）

1）桌面上选择“我的电脑”(右键),高级, 环境变量, 在“系统变量”—>“新建”, 在变量名中输：

JMETER_HOME，变量值中输入：D:\apache-jmeter-2.11

2）再修改CLASSPATH变量，变量值中添加%JMETER_HOME%\lib\ext\ApacheJMeter_core.jar;%JMETER_HOME%\lib\jorphan.jar;%JMETER_HOME%\lib\logkit-1.2.jar; 然后确定即可。

2.3 jmeter启动

启动两种方式，window批处理文件双击bat即可，或者可执行jiar。

3. jmeter安装插件

3.1 安装插件管理工具JMeter Plugins Manager

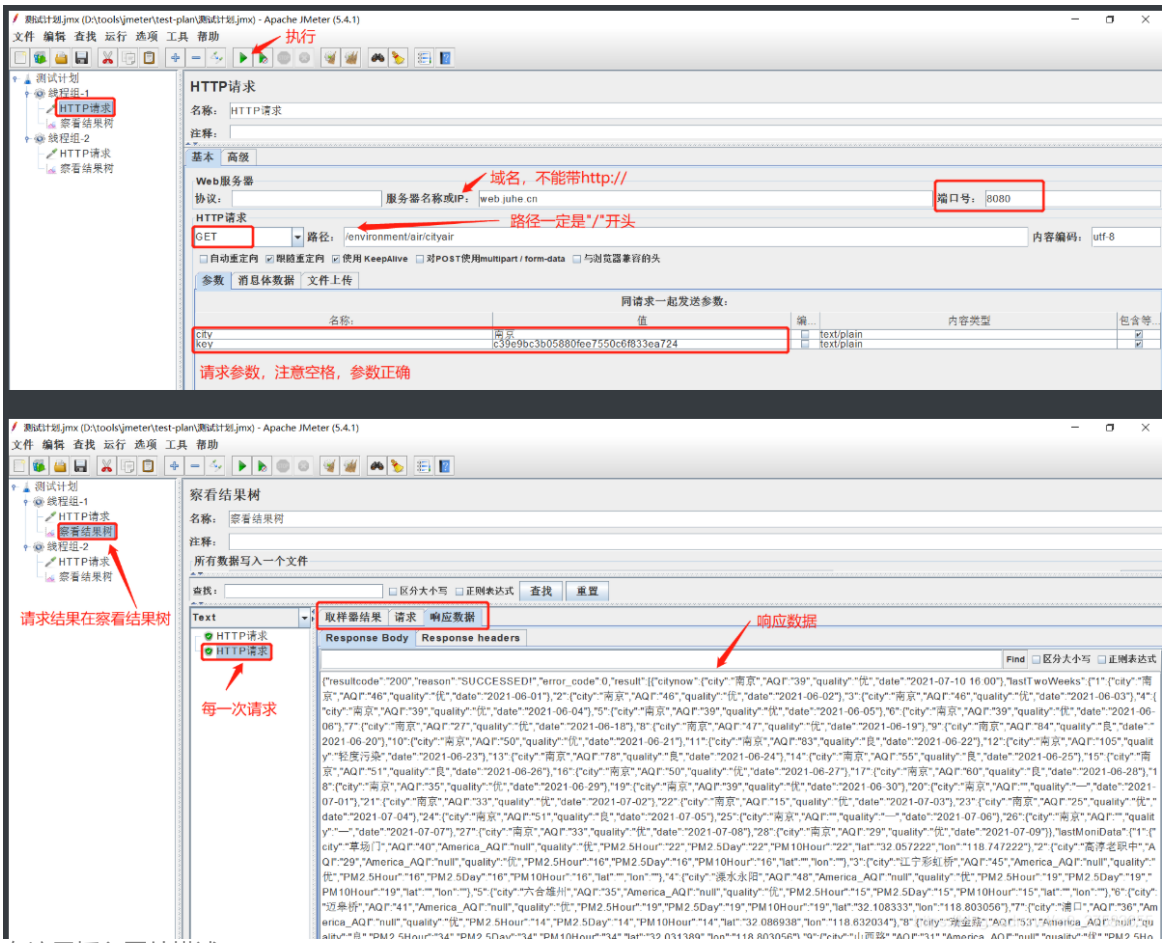
官网下载地址：https://jmeter-plugins.org/install/Install/
将下载的jar包放在jmeter安装目录对应的lib-ext目录下，重启jmeter即可。

3.2 通过Plugins Manager安装插件
打开，选项-Plugins Manager。
Installed Plugins 已安装plugins
Available Plugins 待安装的plgins

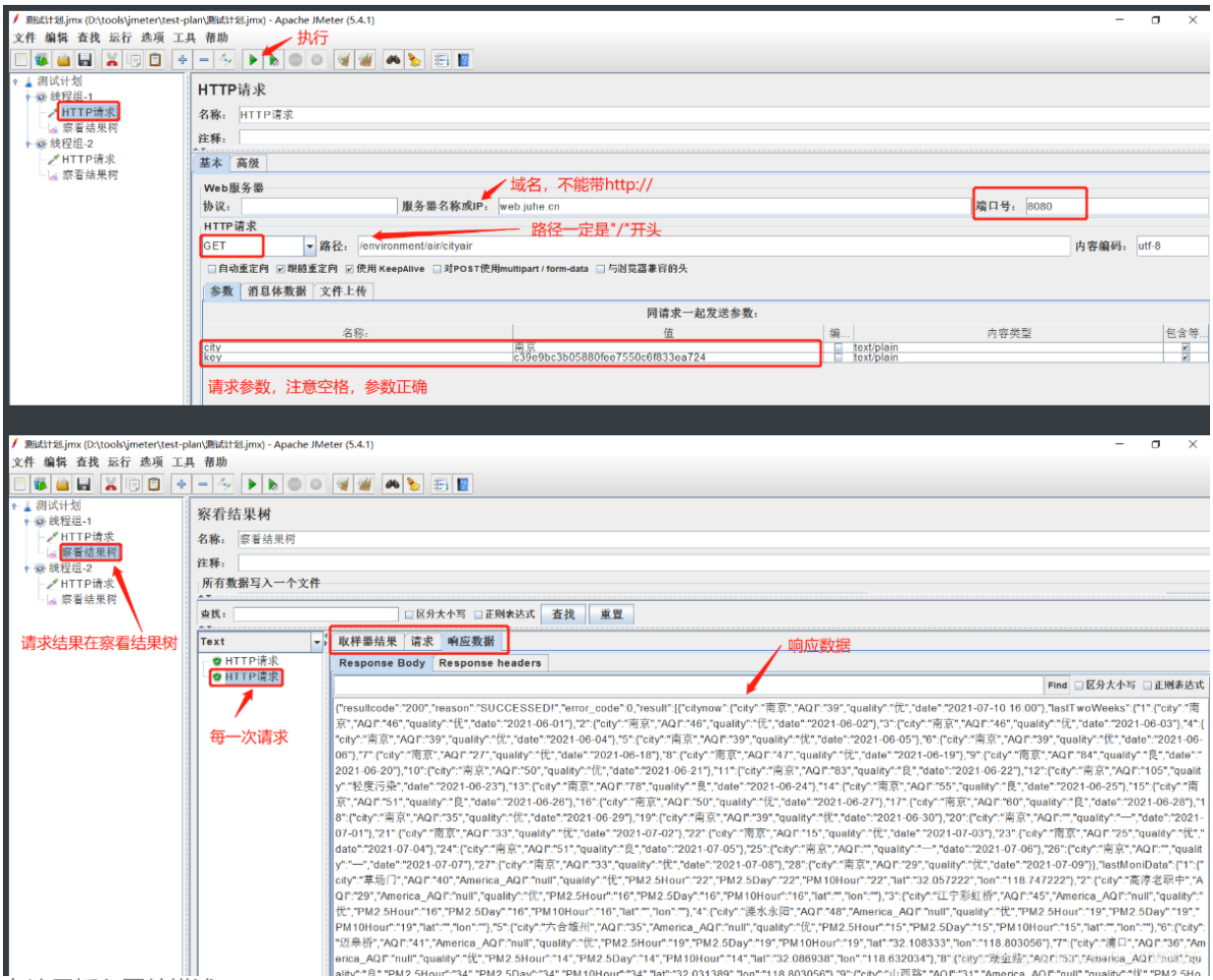
4. Jmeter与loadrunner的区别
简单几个词概括一下吧，一目了然。

两者都是通过中间代理，监控，收集并发客户端指令，录制脚本，并发请求。都可以功能性能压测。
jmeter，开源，轻量级，安装便捷，偏向于功能技术。
loadrunner，收费，重量级，偏向于业务，强大的图表系统。

5. jmeter使用教程
5.1 jmeter发送GET请求
创建测试计划-线程组-HTTP请求与察看结果树，接口测试还没postman与restclient好用，毕竟侧重点不一样。
几个注意的点：
HTTP请求：服务器名称或IP这里不能带http://，下边路径哪里一定要"/"开头，端口不要忘了，参数填正确，注意空格。



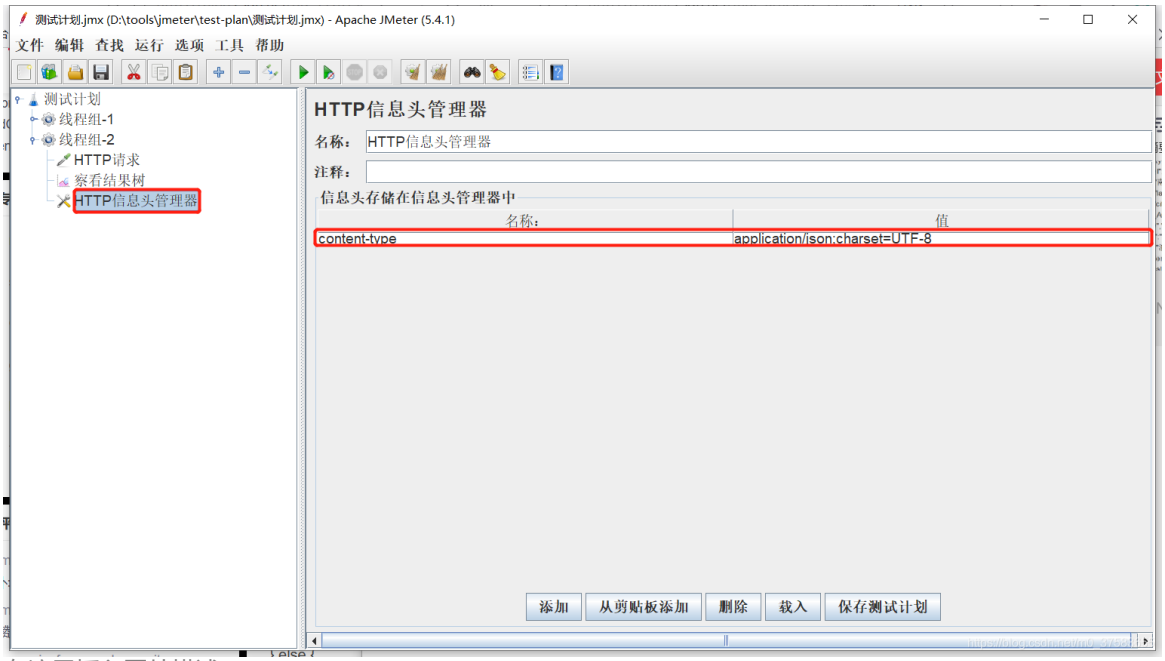
在这里插入图片描述



在这里插入图片描述

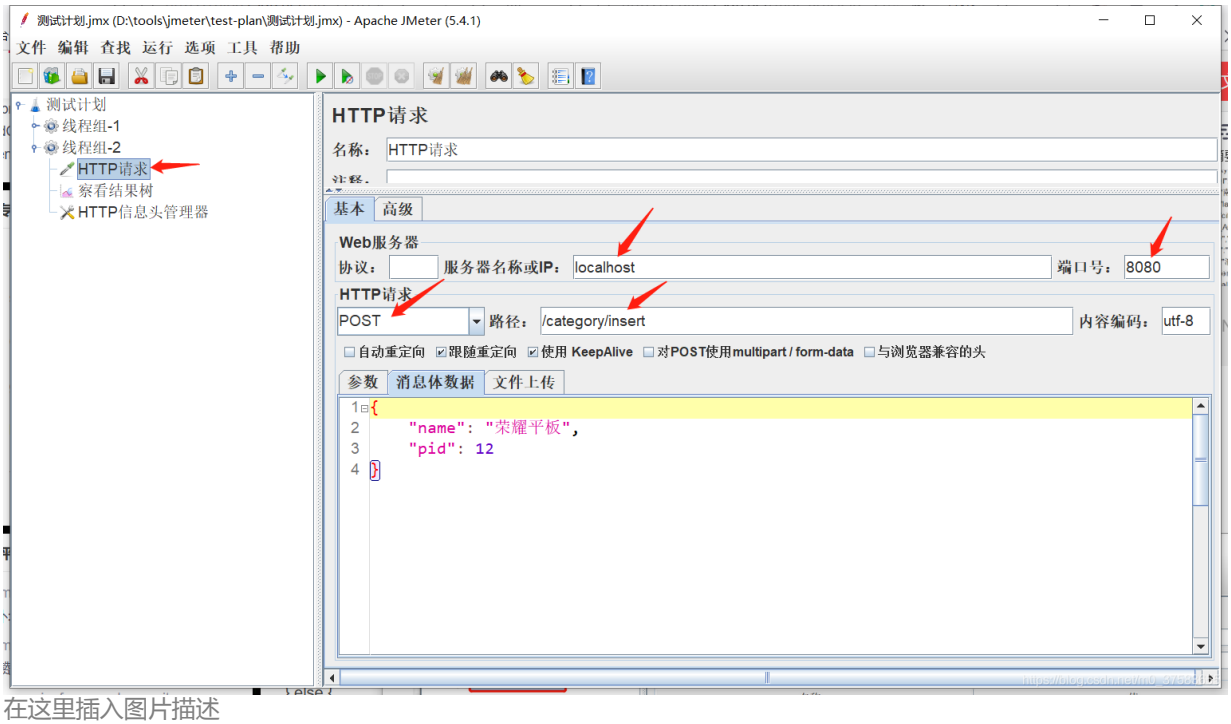
5.2 jmeter发送POST请求

HTTP信息头管理器



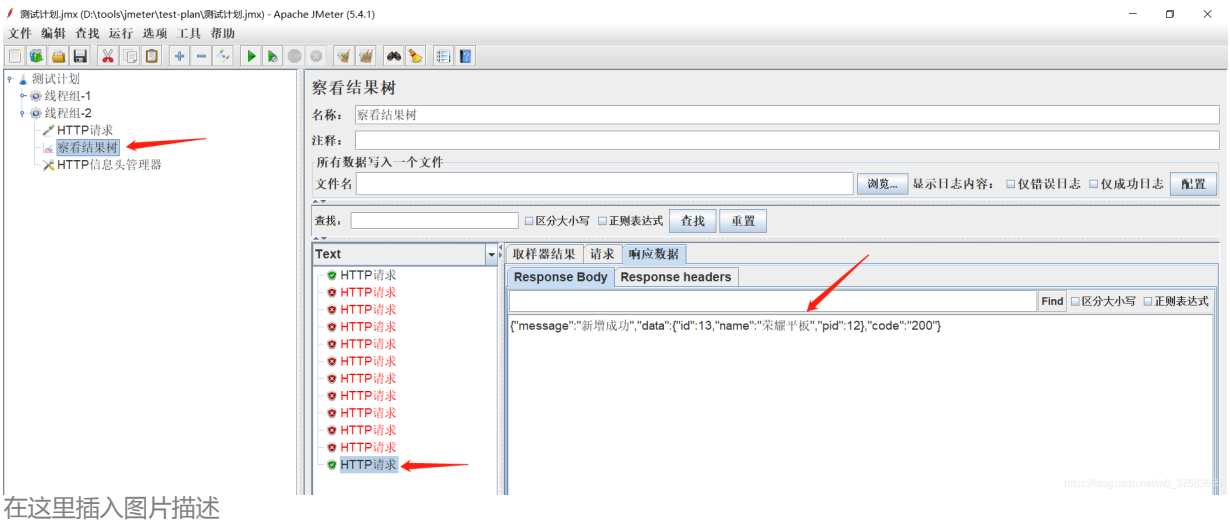
在这里插入图片描述

HTTP请求



在这里插入图片描述

察看结果树



在这里插入图片描述

原文链接：https://blog.csdn.net/m0_37583655/article/details/118630641

jmeter详解

jmeter常用组件

测试计划： 起点，所有组件的容器

线程组： 代表有一定数量的用户

取样器： 向服务器发送请求的最小单元

逻辑控制器： 结合取样器实现一些复杂的逻辑

前置处理器： 在请求之前的工作

后置处理器： 在请求之后的工作

定制器： 负责在请求质检的延迟间隔。固定、高斯、随机

配置元件： 配置信息

断言： 用于判断请求是否成功

监听器： 负责收集结果

执行顺序： 测试计划>>>线程组>>>配置元件>>>前置处理器>>>定时器>>>取样器>>>后置处理器>>>断言>>>监听器

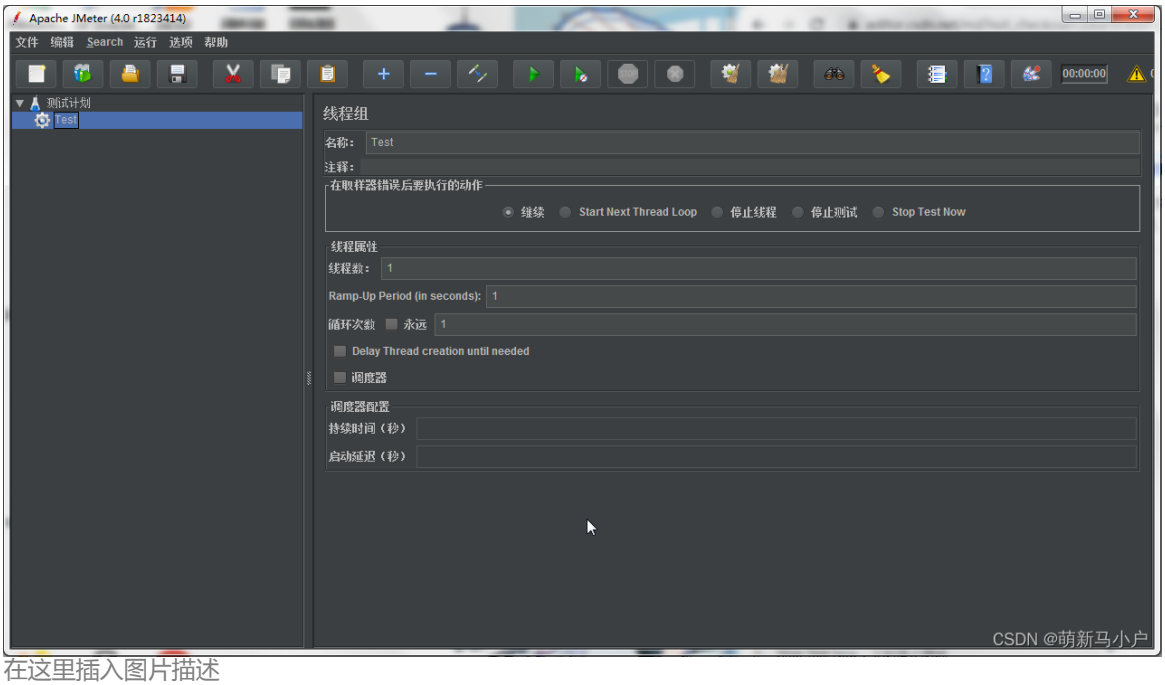
作用域： 辅助组件（除测试计划、线程组、取样器之外的组件）作用于父组件、同级组件，以及同级组件下的所有子组件

一、Threads(Users)

1、线程组

线程组元件可以理解为一个测试计划的开始（jmeter其它的元件都要放在线程组下）

右击测试计划>>>添加>>>Threads(Users)>>>线程组



在这里插入图片描述

在取样器错误后要执行的动作

（1）、继续：继续执行接下来的操作

（2）、Start Next Thread Loop： 开始下一次循环

（3）、停止线程：退出该线程，不在执行此线程的操作

（4）、停止测试：等待当前执行的采样器结束后，结束整个测试

（5）、Stop Test Now：立刻停止测试

线程属性

（1）、线程数：相当于模拟的用户数量，一个用户占一个线程，模拟200个用户就是200个线程

注：进行参数化时，需配置对应的线程数量

（2）、Ramp-Up Period (in seconds)：设置多长时间内启动全部线程。例如线程数为100，时间设定为10s，那么就是10s加载 100个线程，每秒启动的线程数=100/10=10

（3）、循环次数： 如果填具体的数值，就是循环对应的次数，例如线程数为200，循环次数为10，则每个线程发10次请求；如果选择“永远”，则一直执行下去，直到手动停止

（4）、Delay Thread creation until needed：默认不勾选，测试开始时，所有的线程就被创建完。勾选此项，延迟线程创建，直到需要才创建

调度器配置

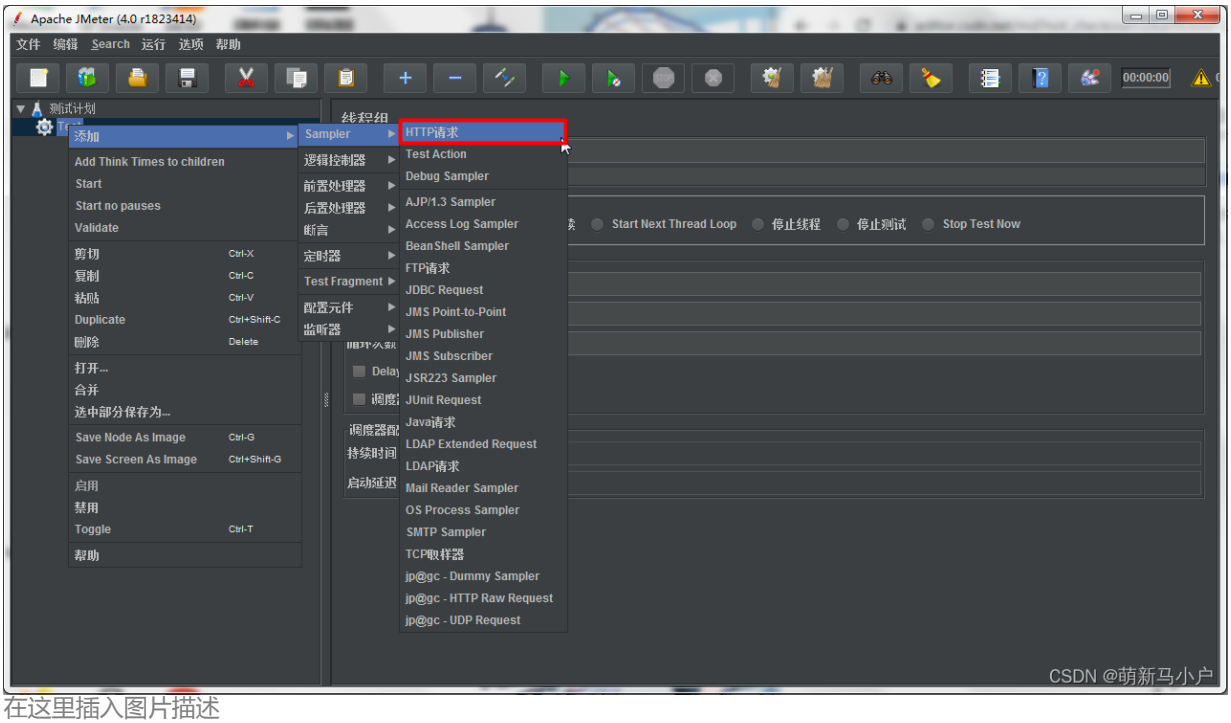
（1）、持续时间（秒）： 脚本持续运行的时间

（2）、启动延迟（秒）：脚本延迟启动的时间

二、Sampler

1、HTTP请求

选择线程组右键>>>添加>>>Sampler>>>HTTP请求



在这里插入图片描述

Basic

（1）、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

（2）、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www.baidu.com

（3）、端口号：目标服务器的端口号，默认值为80

HTTP请求

（1）、方法：发送http请求的方法，例：GET\POST

（2）、路径：目标的URL路径（不包括服务器地址和端口）

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

（4）、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

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

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

parameter： 请求 URL 中添加参数，函数定义中参数，而argument指的是函数调用时的实际参数，简略描述为：parameter=形参(formal parameter)，

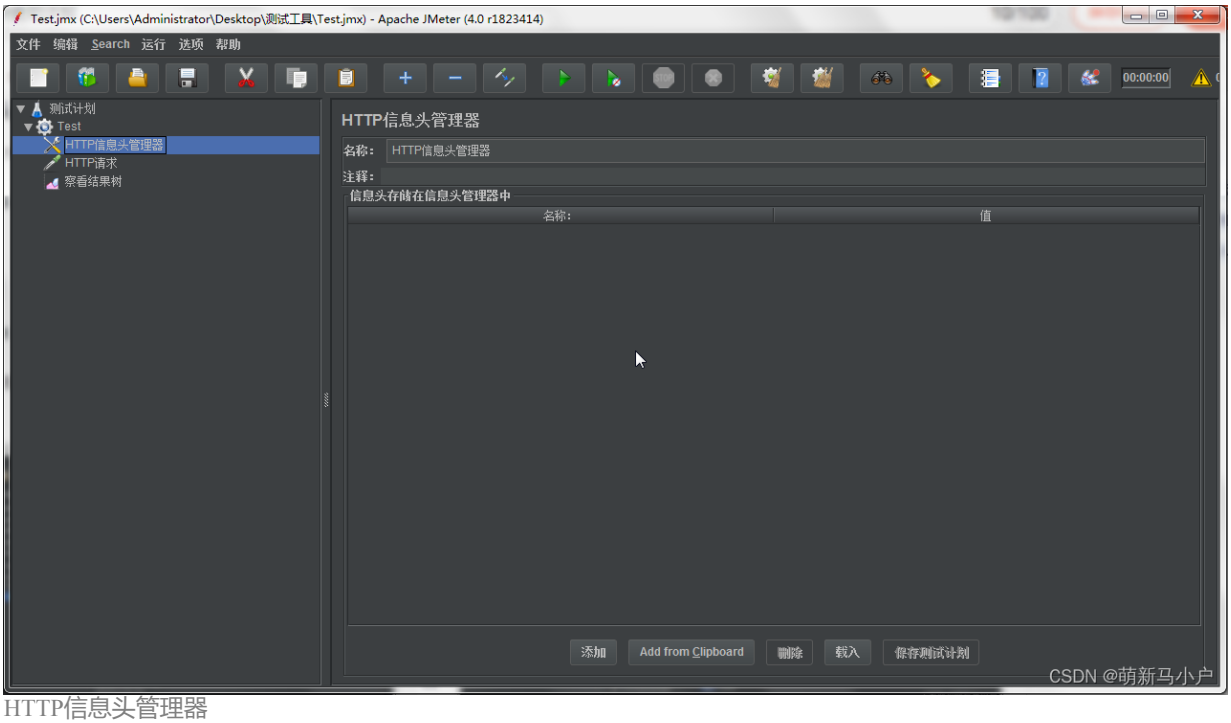
Body Data： 实体数据，就是请求报文里面主体实体的内容，一般我们向服务器发送请求，携带的实体主体参数，可以写入这里

Files Upload： 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

三、配置元件

1、HTTP信息头管理器

设置jmeter发送的HTTP请求头所包含的信息



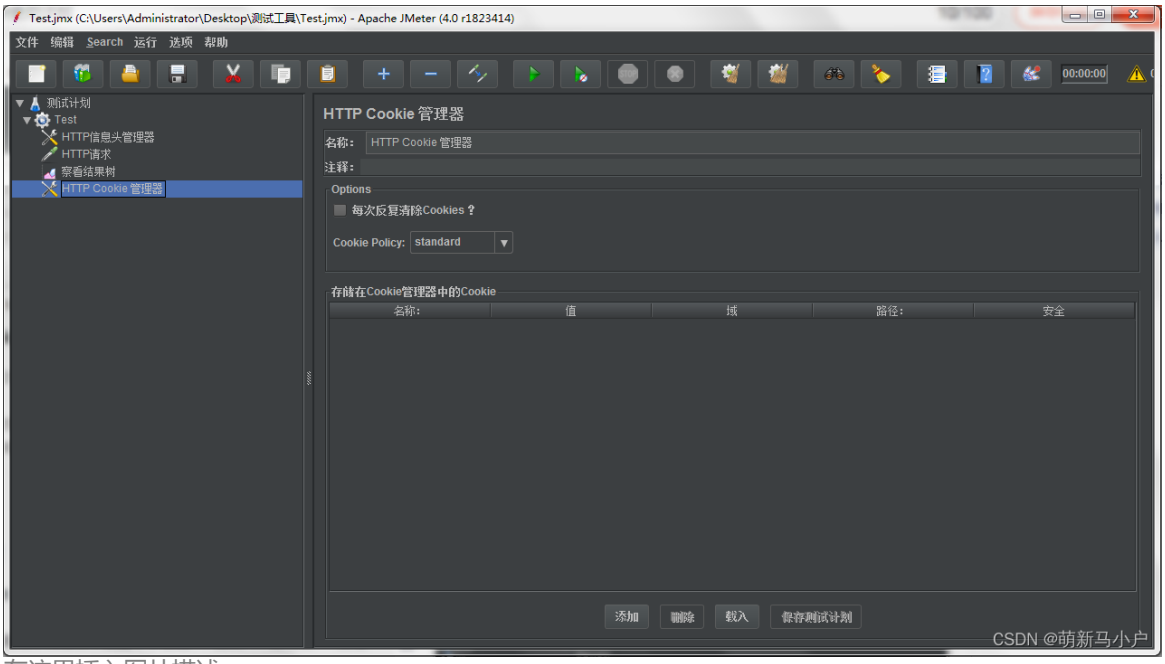
HTTP信息头管理器

右键>>>添加>>>配置元件>>>HTTP信息头管理器

信息头，也就是请求头，会跟随HTTP请求一起发送到服务器。比如需要传输User-Agent 、cookie、token或其他某些信息，或是需要伪造请求头的时候。

2、HTTP Cookie管理器

右键>>>添加>>>配置元件>>>HTTP Cookie管理器



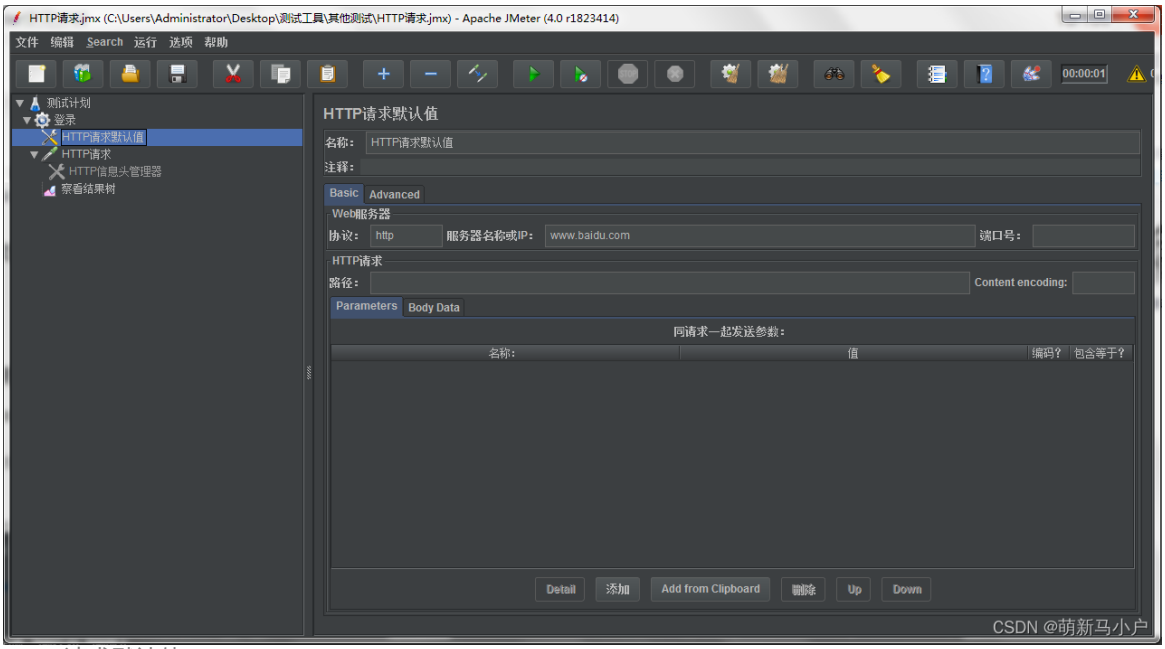
在这里插入图片描述

如果你有一个 HTTP 请求，其返回结果里包含一个 cookie，那么 Cookie 管理器会自动将该 cookie 保存起来，而且以后所有的对该网站的请求都使用同一个 cookie。

3、HTTP请求默认值

管理公用的HTTP请求配置数据

线程组右键>>>添加>>>配置元件>>>HTTP请求默认值



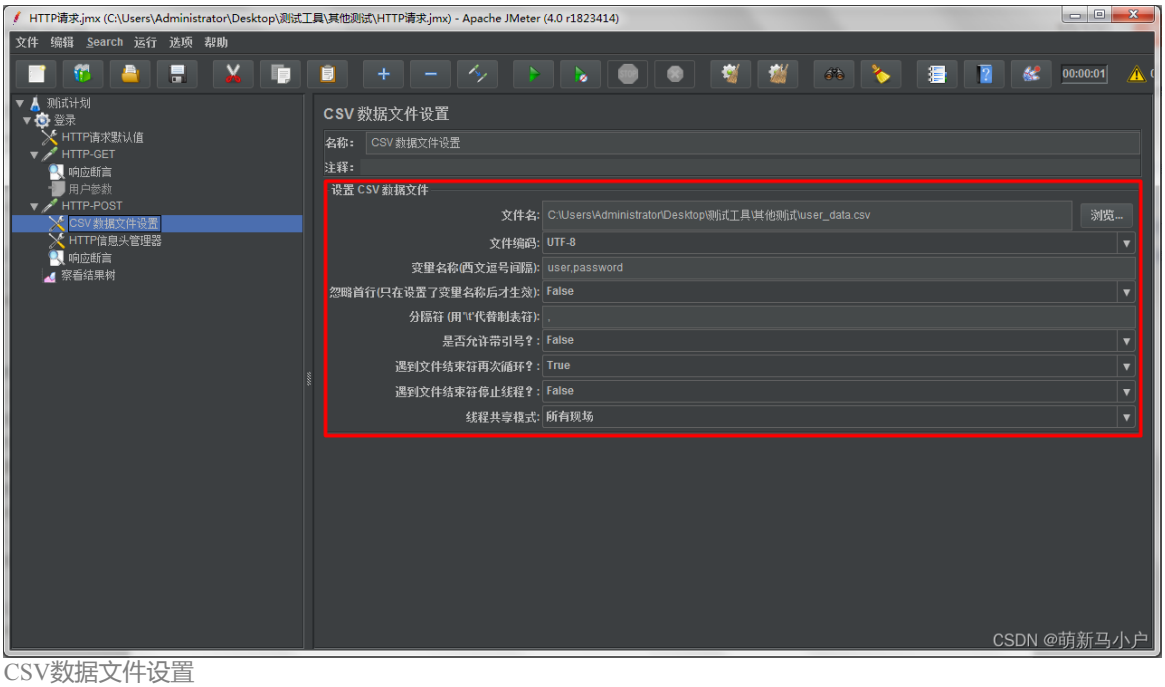
HTTP请求默认值

配置线程组下所有【HTTP请求】的请求行和请求体的默认值，与【HTTP请求】放在同级目录。配置后，每个【HTTP请求】无需重复配置，特殊的请求也可以单独配置，单独配置的优先级更高

4、CSV 数据文件设置

CSV 数据文件变量是指从外部 csv 文件读取数据出来作为变量

选择请求>>>添加>>>配置元件>>>CSV 数据文件设置



CSV数据文件设置

设置CSV数据文件

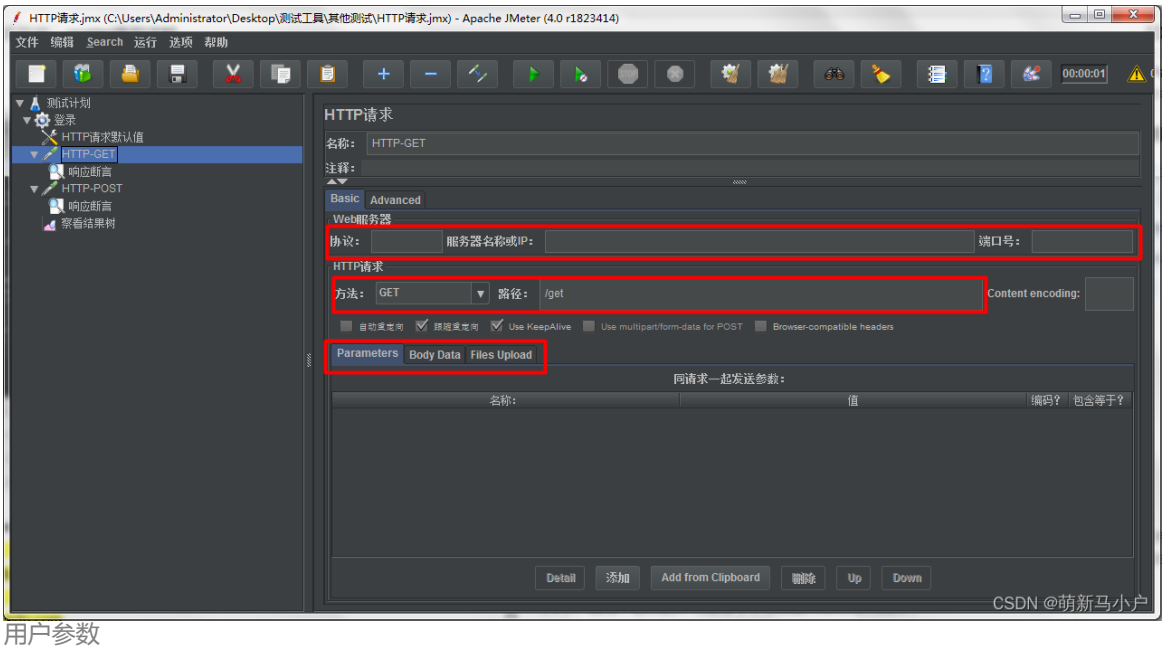
（1）、文件名：csv文件路径，可以是绝对路径或者相对路径

（2）、文件编码：编码格式，与所选文件编码格式保持一致
（3）、变量名称：如果文件中只有一个变量，直接写变量名，如果有多个变量，用英语的逗号隔开
（4）、是否允许带引号？：
设置为true，参数文件包含引号时，实际的数据为引号中的数据。比如参数文件中的数据为"1"，当使用该参数时，实际取得值为1
设置为false，参数文件包含引号时，实际取得值为全部的值。比如参数文件中的数据为"1"，当使用该参数时，实际取得值为"1"
（5）、遇到文件结束符再次循环？：
设置为true后，参数文件中的数据循环使用，测试按照线程组中的设置执行。比如csv 文件共有 10 条记录，但线程数有 15 个，循环 10 次后，重头开始循环取值
设置为false后，参数文件不再循环遍历取值
（6）、遇到文件结束符停止线程：当执行完参数文件中所有参数后，直接停止线程
（7）、线程共享模式：
所有线程（All threads）：参数文件对所有线程共享，这包括同一测试计划中的不同线程组（测试计划下的所有线程组下的所有线程共享参数文件，所有线程之前参数取值互相影响,线程在同一次迭代下取值相同）
当前线程组（Current thread group）： 只对当前线程组中的线程共享（当前线程组下的所有线程公用一个参数文件，同一个线程组下的线程之前取值相互影响,线程在同一次迭代下取值相同）
当前线程（Current thread）： 仅当前线程获取(即每个线程获取一个参数文件，各个线程之间参数取值互不影响,线程在同一次迭代下取值相同)
注：
1)当参数文件的位置与线程组在同级下，线程组下存在循环控制器时，循环控制器下的参数取值相同
2)线程组下存在循环控制器时，当参数文件在循环控制器下，循环控制器下每次迭代时重新取值
3)线程组下存在仅一次控制器，参数文件在仅一次控制器下，当参数在仅一次控制器下取值一次之后，之后无论哪次迭代参数取值都不变，类似于unique once
注：创建CSV文件最好用notepad创建，编码格式为UTF-8

四、前置处理器

jmeter支持的变量
jmeter支持的变量类型：用户自定义变量；函数生成变量；BeanShell 变量；数据文件变量

用户自定义变量
选中请求>>>添加>>>前置处理器>>>用户参数



用户参数

变量引用格式为：${user}

函数生成变量

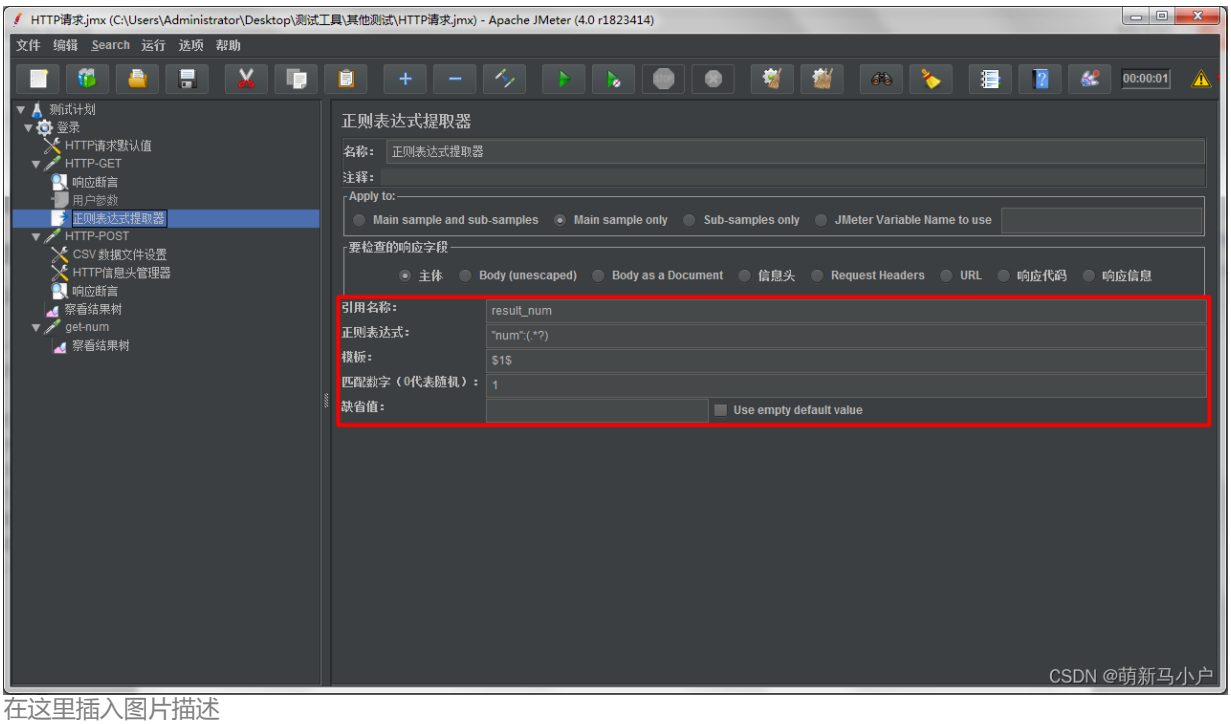
内置的函数

点击菜单栏选项>>>函数助手对话框>>>下拉选择>>>选择对应函数

五、后置处理器

1、正则表达式提取器

接口需要关联时，提取接口指定数据



在这里插入图片描述

选择请求右键>>>添加>>>后置处理器>>>正则表达式提取器

（1）、引用名称：请求要引用的变量名称，如填写 result_num

（2）、正则表达式：匹配需要的内容。

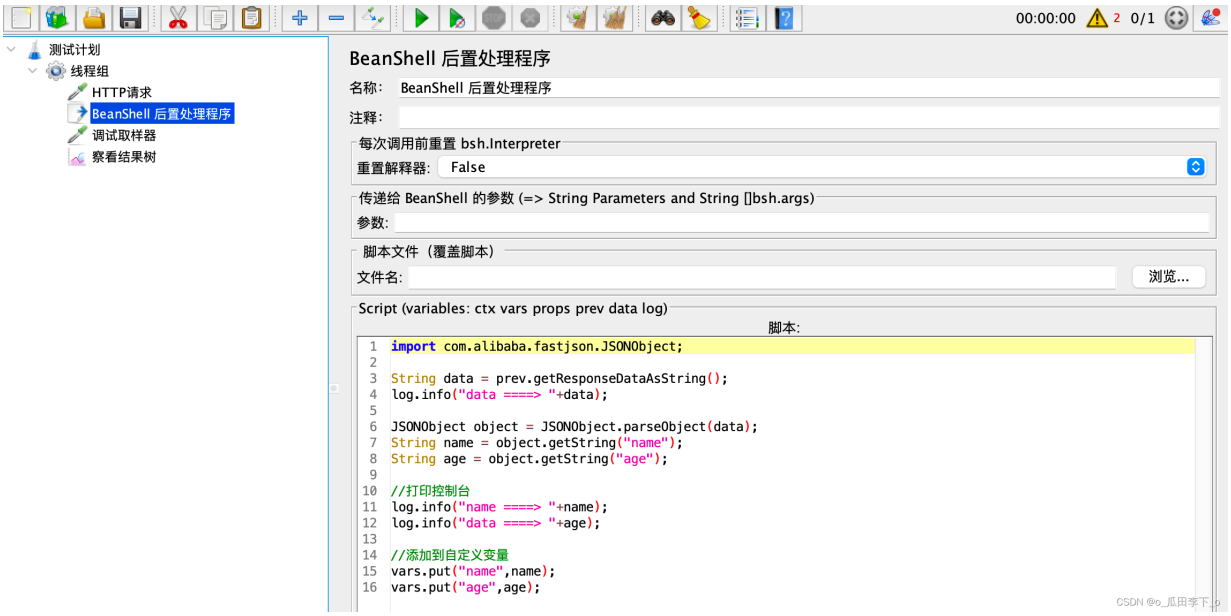
（3）、模板：

（4）、匹配数字：0 代表随机取值，1 代表全部取值，

（5）、缺省值：如果参数没有取得到值，那默认给一个值让它取

2、beanshell 后置处理器

1、添加后置处理器



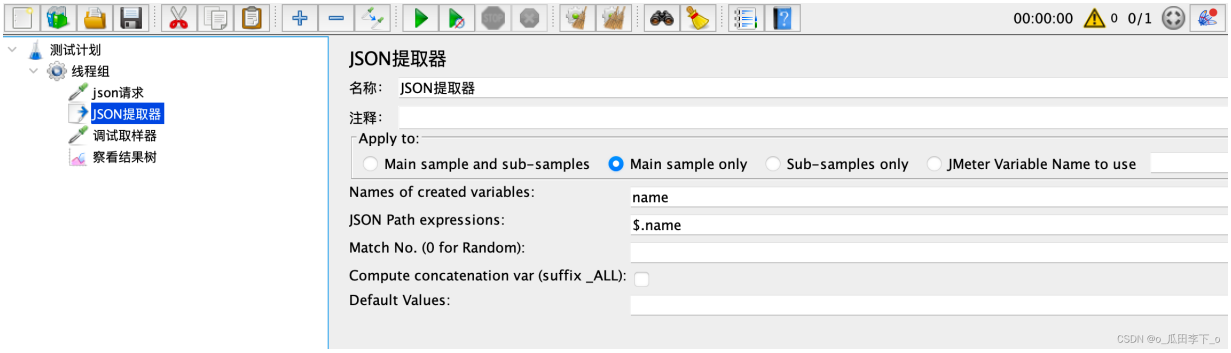
2、通过脚本操作变量，将下列脚本写入后置处理器的脚本中：

```
//获取上一取样器返回取样器结果信息
String response=prev.getResponseDataAsString();
//获取返回取样器状态码
String status=prev.getResponseCode();
//输出结果--写入信息到jmeber.log文件--括号中的信息会输出至log控制台上
log.info(status+"\t"+response);
log.info("--------------------------------------------");
```

3、点击运行，然后查看结果树，我们会发现状态码 200 和一行下划线出现在日志中：

3、JSON提取器

1、添加JSON处理器



2、JSON 提取器参数说明：

name of created variables:创建变量的名称,该名称后面调用时使用${变量名}引用，如：${live_id}

JSONPath Expression：JSON表达式

Match Numbers：匹配数字（0代表随机，1代表第一个，-1代表所有），可为空即默认第一个

Default Value：未取到值的时候默认值

Compute concatenation var(suffix_ALL)：是否统计所有，即将匹配到的所有值保存，名为“变量名_ALL”，使用场景需要获取的

值有多个，后面需要对这一组数据进行操作。

3、使用后置beanshell取样器获取一下变量值：

4、点击运行，可以看到控制台的log中提取到msg2=success，如下图：

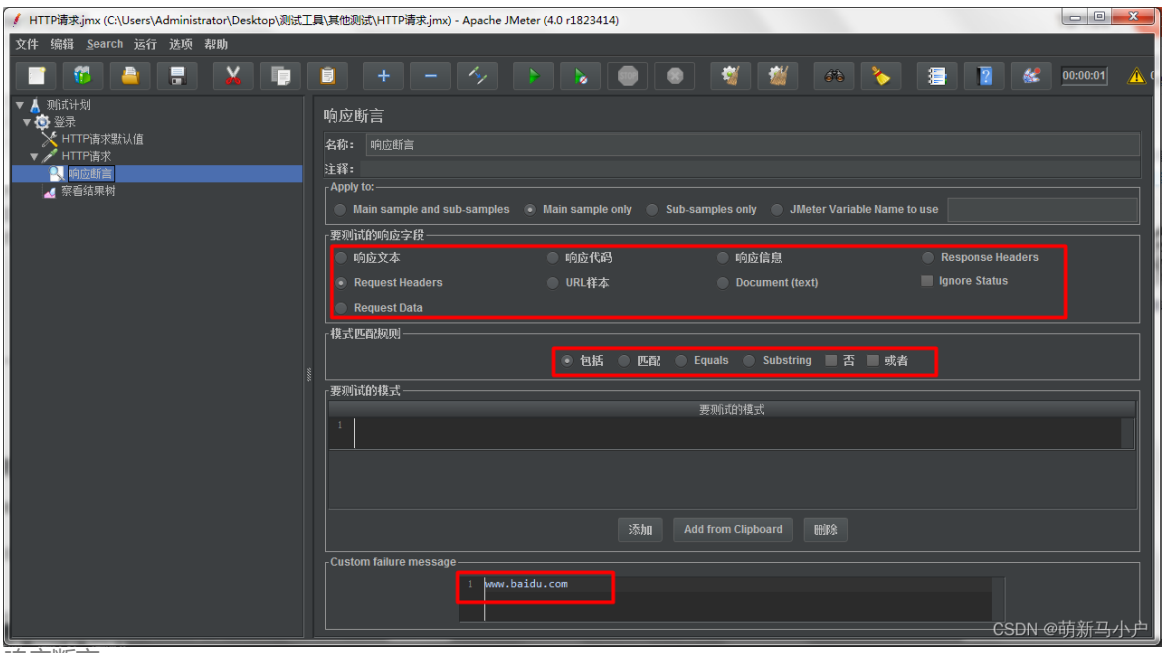
备注：JMeter中日志查看方法为 选项-->勾选"日志查看"，勾选之后才能查看日志，默认不显示日志

六、断言

1、响应断言

检查点，比较预期结果与实际结果

选择一个请求右键>>>添加>>>断言>>>响应断言



响应断言

断言成功，查看结果为绿标；断言失败，查看结果为红标

apply to

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

（2）、Main sample only：只作用于父节点取样器

（3）、sub-samples only：只作用于子节点取样器

（4）、JMeter Variable Name to use：作用于jmeter变量（输入框内输入变量名称）

要测试的响应字段

（1）、响应文本(Text Response)：从服务器返回的响应文本，比如body，包含HTTP头

（2）、响应代码(Response Code) ：比如 200、404

（3）、响应消息(Response Message) ：比如 OK

（4）、Response Headers：响应头，比如 Set-Cookie 头

（5）、Document(text)：通过Apache Tika追踪的各种类型文档的文本

（6）、lonore Status：指示JMeter设置sampler status的初始状态为success。sample status是否成功，由已Response status和断言结果决定，当选中Ignore Status时，Response status被强制设置为success，不执行进一步的断言判断。仅第一次断言时使用

模式匹配规则

（1）、包括：响应内容包括需要匹配的内容即代表响应成功，支持正则表达式

（2）、匹配：响应内容要完全匹配需要匹配的内容即代表响应成功，大小写不敏感，支持正则表达式。

（3）、Equals：响应内容要完全等于需要匹配的内容才代表成功，大小写敏感，需要匹配的内容是字符串正则表

达式

（4）、Substring：返回结果包含指定结果的字串，但是 subString 不支持正则字符串

（5）、否：不进行匹配

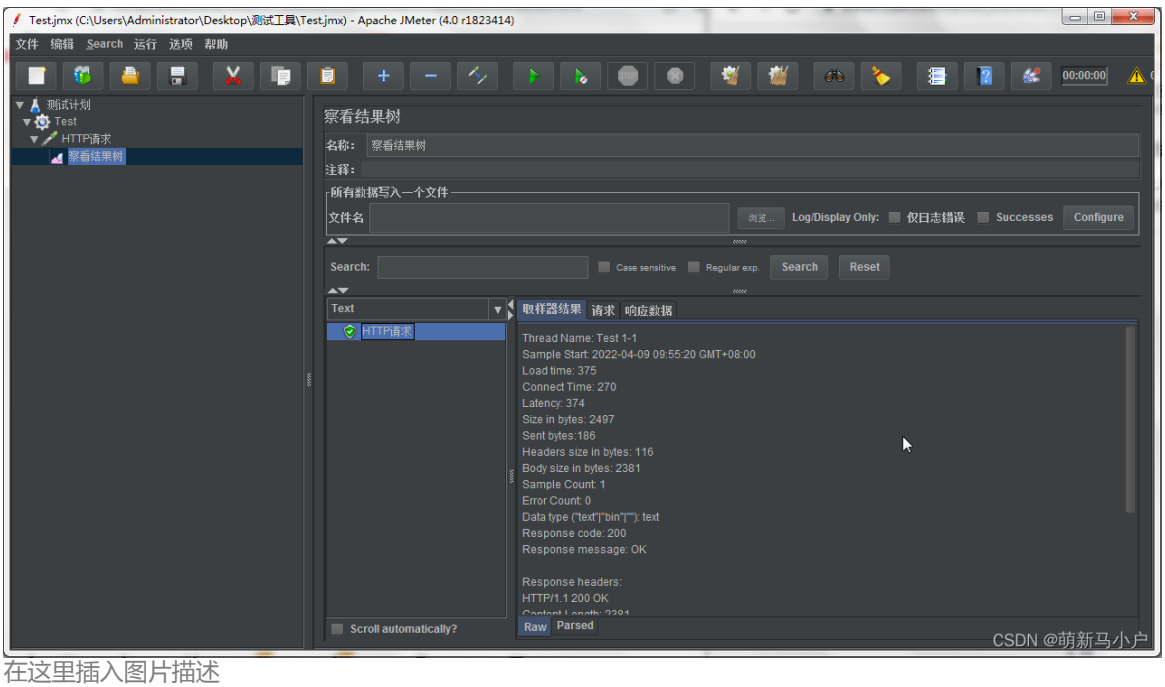
Custom failure message： 自定义失败消息

七、监听器

1、察看结果树

察看请求发送和返回的信息

查看结果树可以放在线程组下或者某个配置下，右键>>>添加>>>监听器>>>查看结果树



在这里插入图片描述

所有数据写入一个文件

（1）、文件名：可以通过浏览，选择一个文件，这样jmeter在执行的过程中，会将所有的信息输出到文件，也支持打开一个结果文件进行浏览

（2）、Log/Display Only（显示日志内容）：

仅日志错误：表示只输入报错的日志信息

Successes：表示只输出正常响应的日志信息

不勾选：表示输出所有的信息

（3）、Configuer：配置需要输出的内容

Search： 在输入框中输入想查询的信息，点击查找（Search），可以在请求列表中进行查询，并在查询出的数据上加上红色的边框

结束数显示类型切换： 通过结果树上面的下来看可以进行切换，包含多种显示方式，默认Text

取样器结果： 取样器的详细结果，可以切换取样器的显示方式Raw/Parsed

请求： 显示当前取样器发送的详细请求内容，支持查找

响应数据： 显示请求得到的响应内容，支持查找

Scroll automaticlly?： 当执行的取样器较多，设置是否滚屏显示

原文链接：https://blog.csdn.net/qq_45138120/article/details/124056704

JMeter参数化的种方式

案例：邮箱登录操作，参数化登录的用户名

方法一、依赖Jmeter自带的函数助手

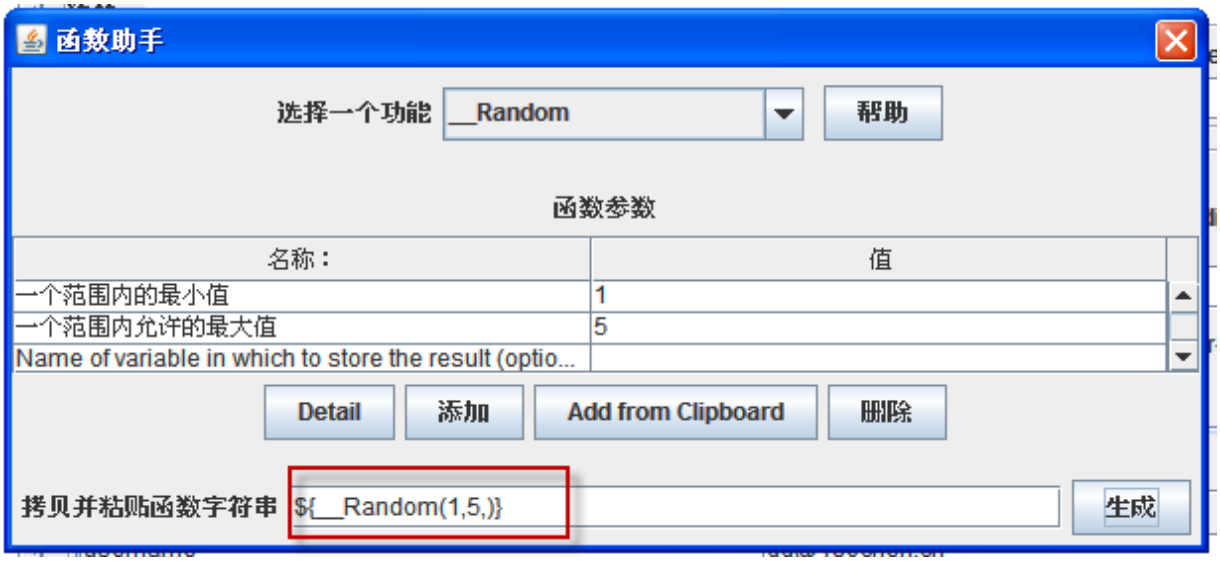
选项-->函数助手对话框，即可打开函数助手弹窗

（1）比如使用函数_Random：（循环控制器加以辅助，生成多个表单）

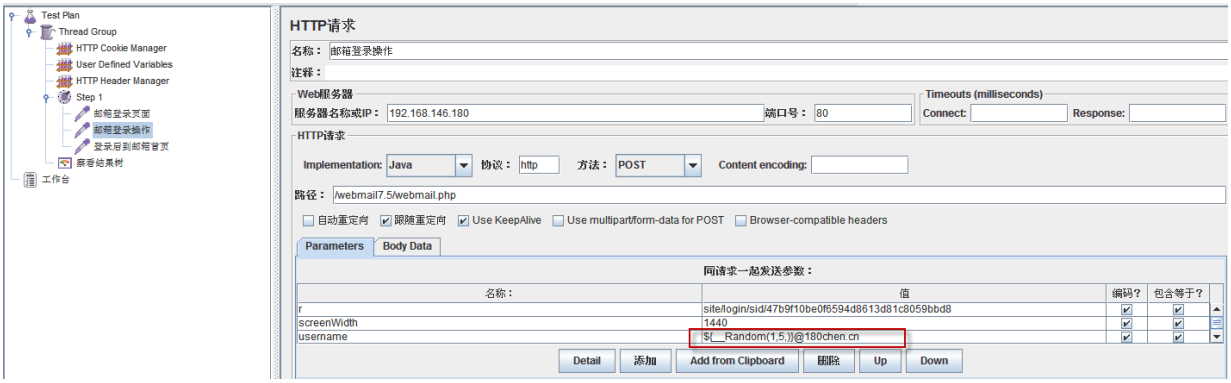
输入最小值、最大值，点击下方的【生成】按钮，即会生成：${__Random(1,5,)}

生成的随机数可能重复：

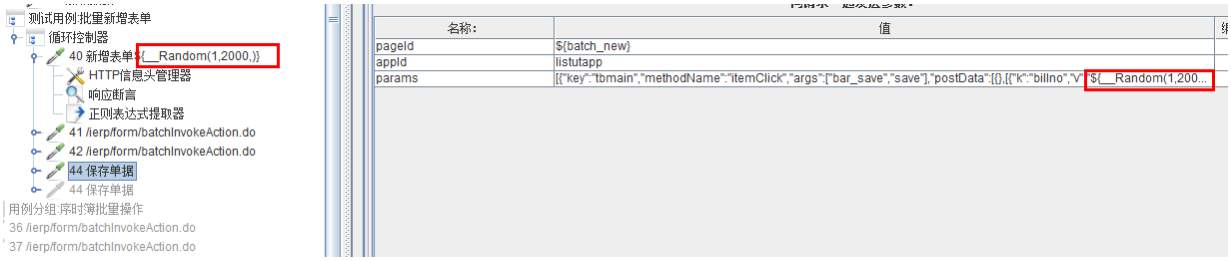
解决办法：加时间戳



假如邮箱是163的，则用户名参数的值直接填入：${__Random(1,5,)}@163.com，我测试的邮箱是内网测试用的域名是180chen.cn，则填写${__Random(1,5,)}@180chen.cn

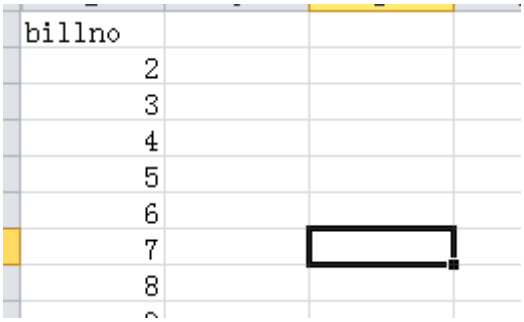


案列：随机生成单据编号名



（2）比如使用函数__CSVRead（##进过测试，每次都只取到第一行的数据**）**

1、先准备csv文件，可以先在excel文件里准备好数据，保存的时候格式选择csv格式；

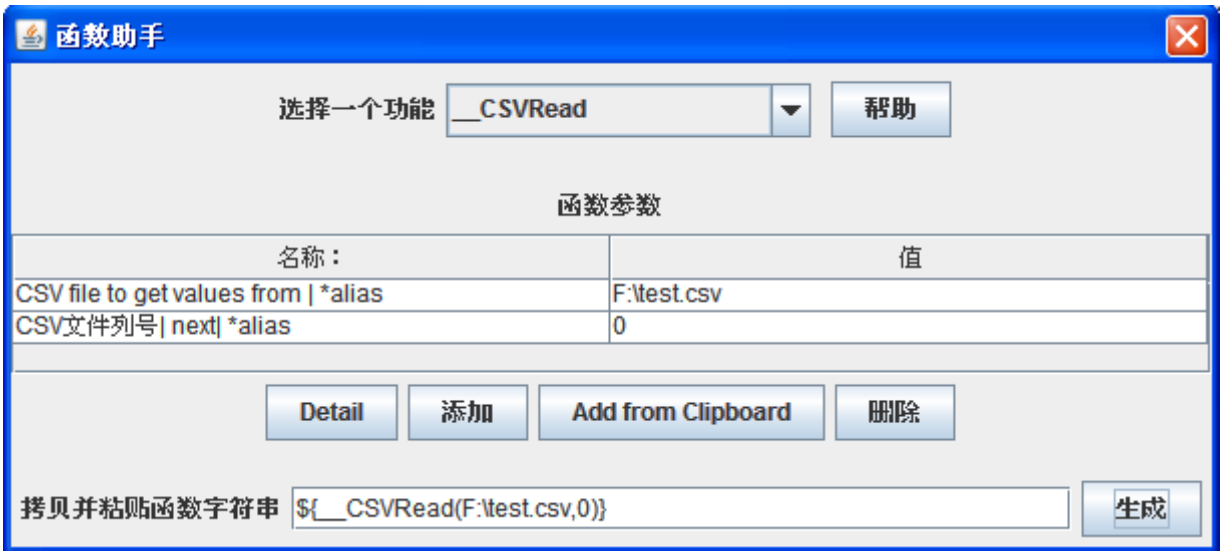


输入csv文件所在的路径:F:\test.csv，输入参数所在的列（注意：此处列数是从0开始数的，第一列是用户名，对应的列号为0，第二列是密码，对应的列号为1...），生成被调用的函数：${__CSVRead(F:\test.csv,0)}

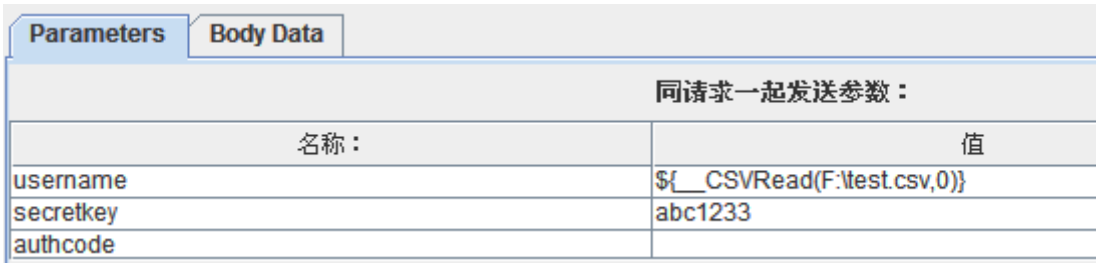
##第一列是单据的编

生成调用的函数：${__CSVRead(F:\test.csv,0)}

##此时第一行如果是变量名，那么变量的值会被取到

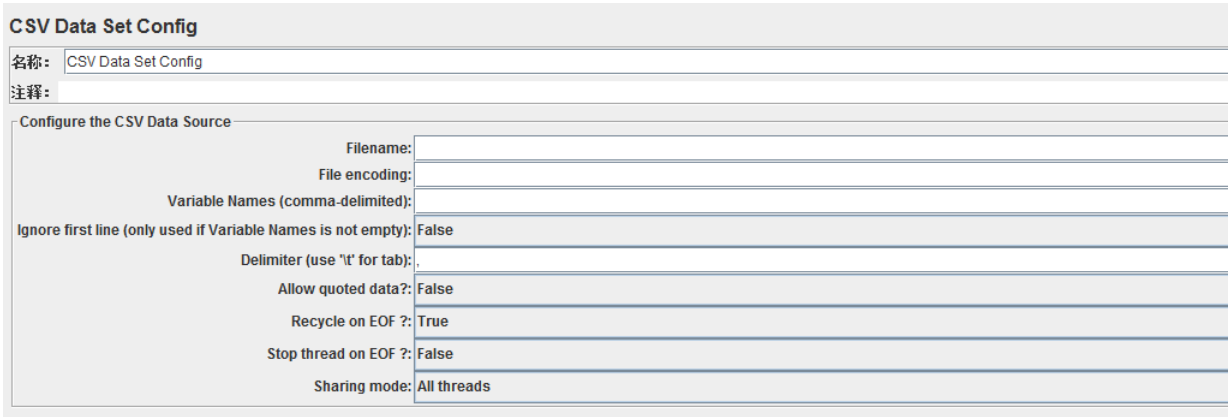


在参数对应的值处填入：${__CSVRead(F:\test.csv,0)}



方法二、CSV Data Set Config

选中取样器，右键：添加-配置元件-CSV Data Set Config，从csv文件中读取



Filename:csv文件所在的路径以及名称如：F:\test.csv ；（其实不一定要csv文件，亲测txt格式的文件也可以）

File encoding：给出页面的编码方式，可以不填写；这里以百度为例，它的源代码里<meta http-equiv="content-type" content="text/html;charset=gb2312"> ，所以这里File encoding:gb2312

Variable Names(comma-delimited)：给出变量名如：name,pwd;这里的变量名是给后面引用用的，如要用到这个文件的值，可以利用变量名来引用：${name},${pwd},如test.csv文件中有这样的数据:1@180chen.cn,abc1233,那${name}就可以引用到1@180chen.cn,${pwd}就可以引用到abc1233

Delimiter(use '\t' for Tab):这个是用来隔开变量的分隔符，如上面的name,pwd,那分隔符就是“,”

Allow quoted data?：是否允许引用数据，---这个目前还未弄明白，设置成True或者False都能正常引用数据。

Recycle on EOF?：到了文件尾是否循环，True---继续从文件第一行开始读取，False---不再循环

Stop thread on EOF?：到了文件尾是否停止线程，True---停止，False---不停止，注：当Recycle on EOF设置为True时，此项设置无效。

当Recycle on EOF 选择flase时，Stop thread on EOF选择true，线程4个，参数3个，那么只会请求3次

当Recycle on EOF 选择flase时，Stop thread on EOF选择flase，线程4个，参数3个，那么会请求4次，但第4次没有参数可取，不让循环，所以第4次请求错误

ignorefirstline：忽略变量名

Sharing mode：共享模式，All threads---所有线程，Current thread group—当前线程组，Current thread—当前线程。这个地方和LoadRunner中的迭代取之相反，经试验得出来的结果是：

All threads：测试计划中所有线程，假如说有线程1到线程n（n>1），线程1取了一次值后，线程2取值时，取到的是csv文件中的下一行，即与线程1取的不是同一行。

Current thread group：当前线程组，假设有线程组A、线程组B，A组内有线程A1到线程An，线程组B内有线程B1到线程Bn。取之情况是：线程A1取到了第1行，线程A2取第2行，现在B1取第1行，线程B2取第2行。

Current thread：当前线程。假设测试计划内有线程1到线程n（n>1），则线程1取了第1行，线程2也取第1行。

综上：CSV Data Set Config实现的功能跟之前用的：${__CSVRead(F:\test.csv,0)}这个函数实现的功能大体上是一样的。

#注意：单个线程的话每次读取的都是第一行

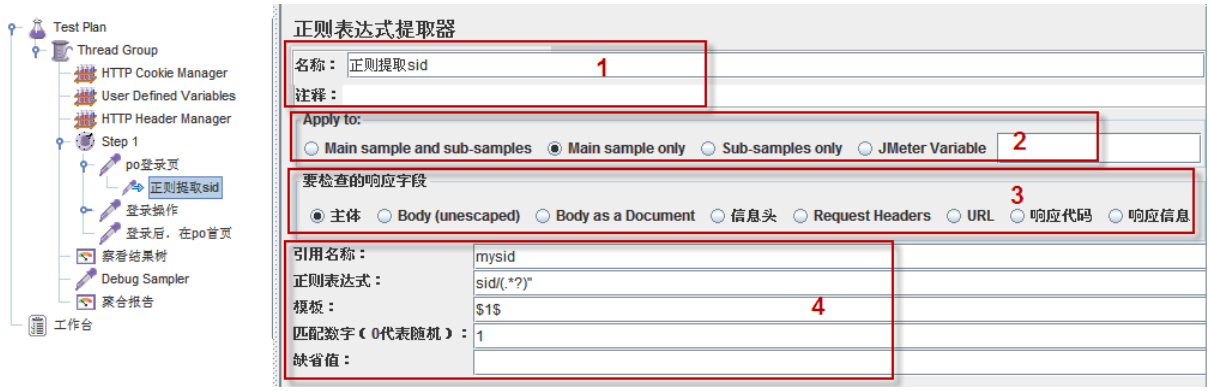
方法三、用户定义的变量

选中取样器，右键：添加-配置原件-用户定义的变量

在列表中填入名称和值，在别处就可以使用${name}、${pwd}来引用

方法四、正则表达式提取器获取

在打开登录页的时候服务器有返回一个sid，获取到sid后，然后登录进入到首页；所以在这个请求下添加后置处理器：右键-添加-后置处理器-正则表达式提取器



第1部分：名称+注释，可不修改，只是为了方便自己识别

第2部分：apply to 默认即可

第3部分：要检查的字段：主体等选择，一般我们选择主体，即服务器返回给我们的页面主体信息

第4部分：

【引用名称】：就是参数名称，在别处引用；如输入mysid，别处引用的时候使用${mysid}

【正则表达式】：表达式中()内的内容就是要提取的。如sid/(.*?)"，表示查找sid/字符串之后的内容，直到出现第一个"时结束；（注意括号里的表示提取的内容）

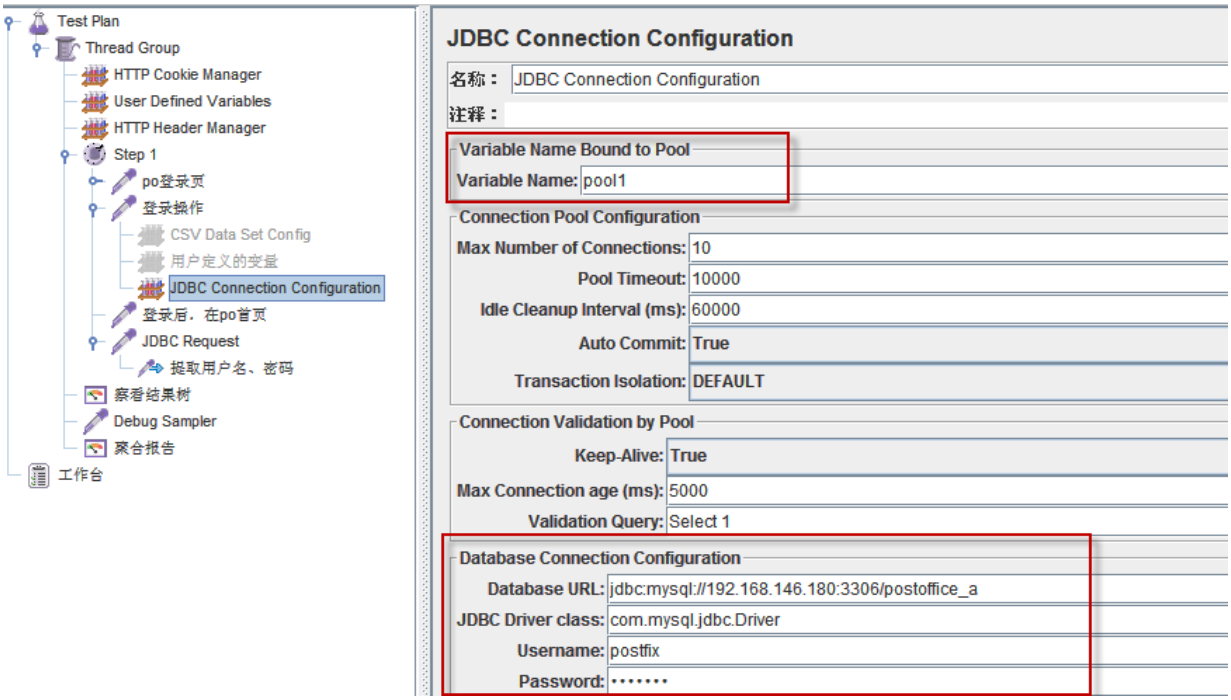
【模板】：用$$引用起来，如果在正则表达式中有多个正则表达式（多个括号括起来的东东），则可以是1（表示只有一组数据），2等等，表示解析到的第1个、第2个值给mysid

【匹配数字】：0代表随机取值，-1代表所有，1代表全部取值

【缺省值】：如果参数没有取到值，则使用此处的缺省值
注意：运行脚本后，在“察看结果树”监听器中，[响应数据]标签页先搜索sid出现的位置，及出现的规律，如出现的时候前面会有“sid/”字符串； 调试正则提取表达式的时候，可添加Debug Sampler来查看是否正确提取到对应的值（右键-添加-Sampler-Debug Sampler）

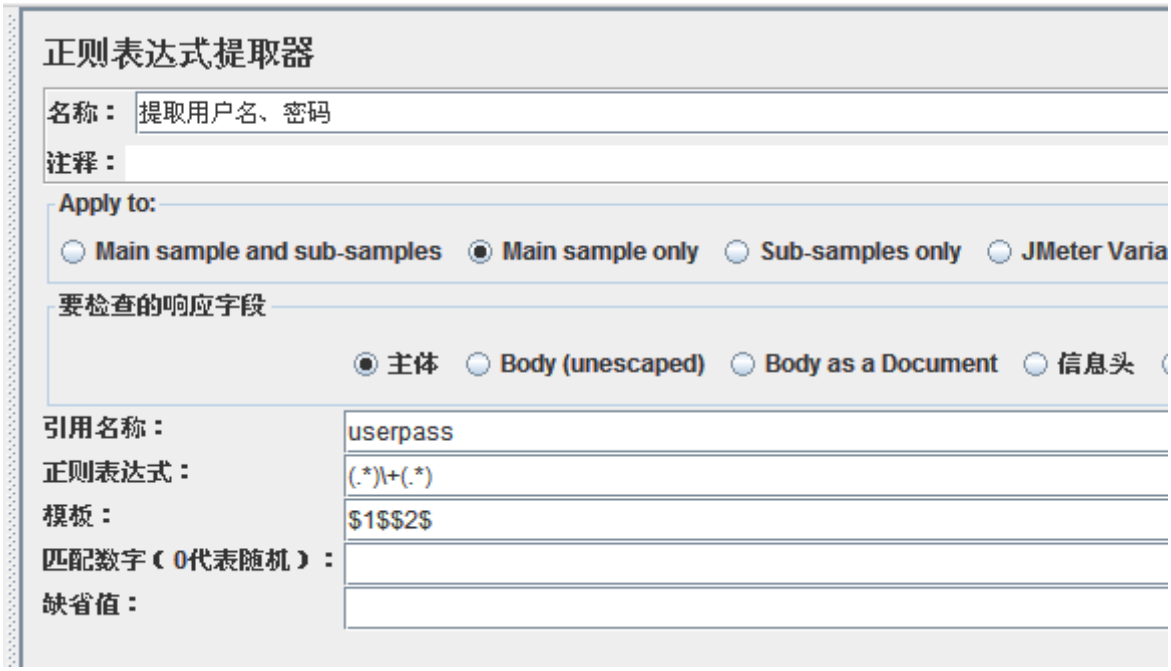
方法五、从数据库获取

1) 将其中的mysql-connector-java-5.1.34-bin.jar放到Jmeter的lib目录下
2) 添加“配置元件”->“JDBC Connection Configuration”，设置下列参数：
Variable Name：连接池名称
Database URL：jdbc:mysql://host:port/db（jdbc:mysql://ip地址:端口号/数据库名称）
JDBC Driver class：com.mysql.jdbc.Driver
username：连接数据库的用户名（如root）
password：连接数据库的密码



3) 添加“Sampler”->“JDBC Request”，在SQL Query中输入查询语句，如下：
select concat(domain,'+',po_pwd) as userpass from domain;

Variable Name：连接池名称
注意：该值要和JDBC Connection Configuration中配置的Variable Name值对应，否则会提示：
No pool found named: 'pool12', ensure Variable Name matches Variable Name of JDBC Connection Configuration ;
4) 在JDBC请求中添加“后置处理器”->“正则表达式提取器”，以提取用户名、密码为例，设置正则表达式提取器的参数：
引用名称：userPass
正则表达式：(.*)\+(.*)
模板：$1$$2$
使用时，userPass_g1即为用户名，userPass_g2即为对应用户名的密码；



注意：

（1）+在正则表达式中是关键字，所以需要\转义。

（2）匹配数字，填0或不填，表示随机读取，如果填正整数，如1，则不同虚拟用户或循环固定读取某行数据。

JMeter控制器的使用

前言：

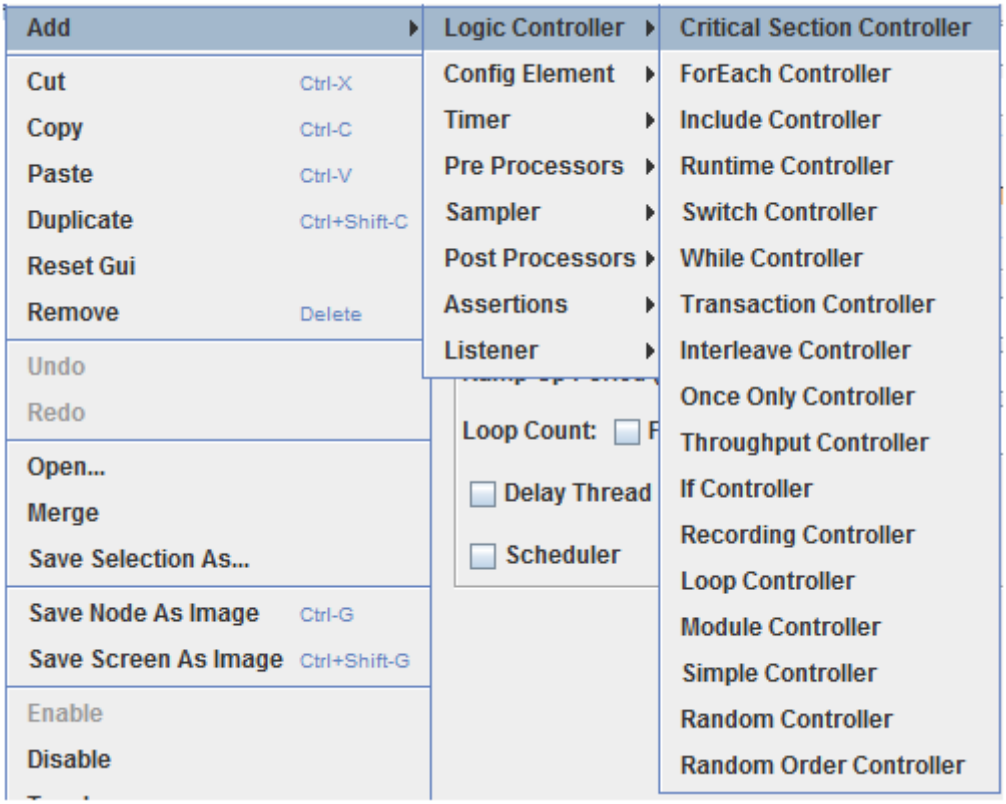
1、 Jmeter官网对逻辑控制器的解释是：“Logic Controllers determine the order in which Samplers are processed.”。意思是说，逻辑控制器可以控制采样器(samplers)的执行顺序。由此可知，控制器需要和采样器一起使用，否则控制器就没有什么意义了。放在控制器下面的所有的采样器都会当做一个整体，执行时也会一起被执行。

2、JMeter中的Logic Controller分为两类：

a) 控制测试计划执行过程中节点的逻辑执行顺序，如：Loop Controller、If Controller等；

b) 对测试计划中的脚本进行分组、方便JMeter统计执行结果以及进行脚本的运行时控制等，如：Throughput Controller、Transaction Controller。

3、Jmeter提供如下这么多的控制器：



一、简单控制器（Simple Controller）：

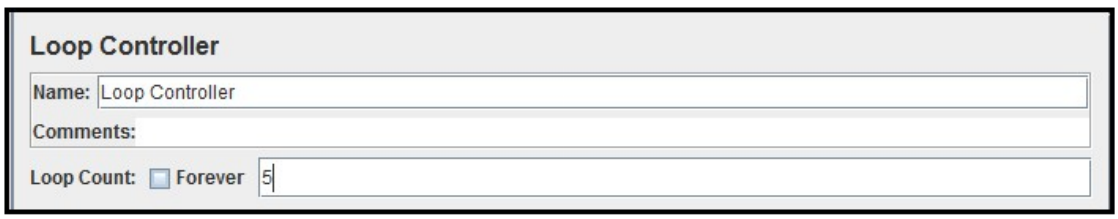
作用：这是Jmeter里最简单的一个控制器，它可以让我们组织我们的采样器和其它的逻辑控制器（分组功能），提供一个块的结构和控制，并不具有任何的逻辑控制或运行时的功能。

二、循环控制器（Loop Controller）：

作用：指定其子节点运行的次数，可以使用具体的数值（如下图，设置为5次），也可以使用变量

1、Forever选项：勾选上这一项表示一直循环下去

2、如果同时设置了线程组的循环次数和循环控制器的循环次数，那循环控制器的子节点运行的次数为两个数值相乘的结果。



三、仅一次控制器（Once Only Controller）：

作用：在测试计划执行期间，该控制器下的子结点对每个线程只执行一次，登录场景经常会使用到这个控制器。

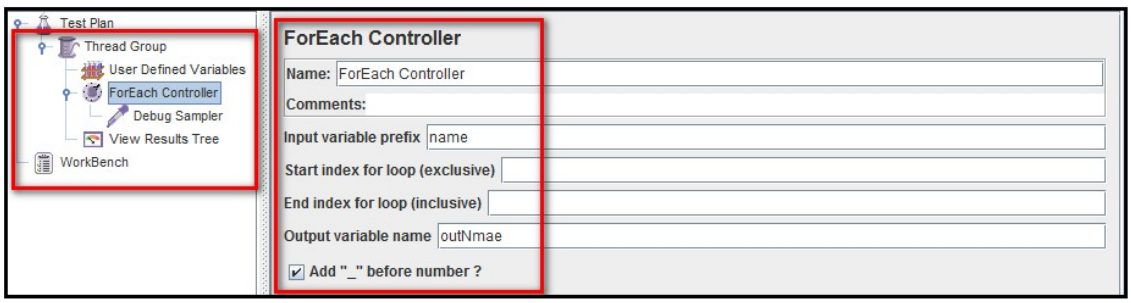
注意：将Once Only Controller作为Loop Controller的子节点，Once Only Controller在每次循环的第一次迭代时均会被执行。

四、ForEach控制器（ForEach Controller）：

作用：ForEach控制器一般和用户自定义变量一起使用，其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。如下图：

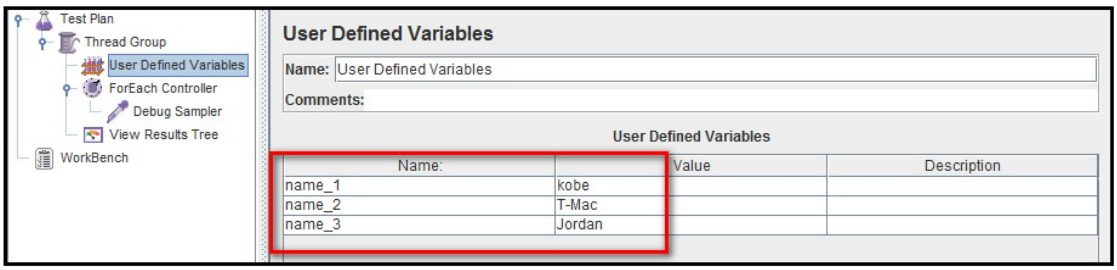
参数:

- Input Variable Prefix：输入变量前缀
- Output variable name：输出变量名称
- Start index for loop(exclusive)：循环开始的索引（
- 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)：循环结束的索引
- Add"_"before number：输入变量名称中是否使用"_"进行间隔。



用户自定义变量：

变量名前缀为ForEach Controller中Input variable prefix定义的name + 下划线（上图中我们勾选了下划线）+数字编号



执行结果：

总共执行了3次，每次执行时会把获取到的变量值赋值给输出变量outNmae，其它地方可以通过${outNmae}进行调用。

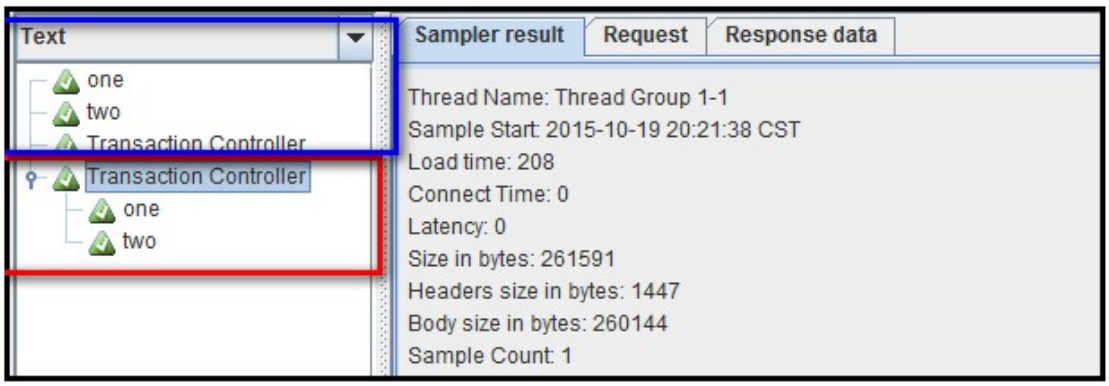
五、事务控制器（Transaction Controller）：

作用： 事务控制器会生产一个额外的采样器，用来统计该控制器子结点的所有时间。

事务控制器定义的事务是否成功取决于子事务是否都成功，子事务其中任何一个失败即代表整个事务失败。

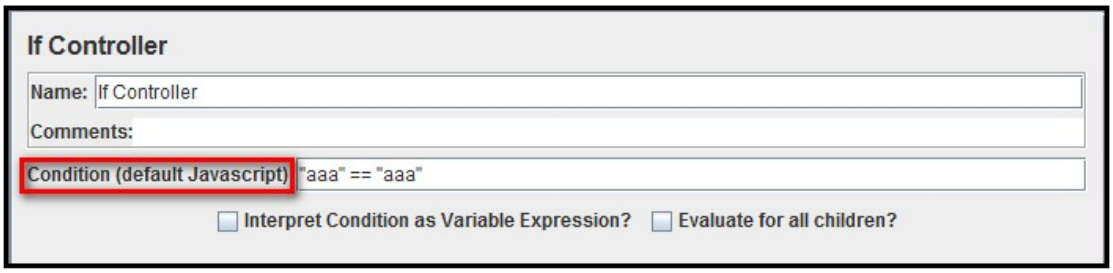
参数：

- Generate parent sample：(选中这个参数结果展示如下图红框，否则显示为下图蓝框)
- Include duration of timer and pre-post processors in generated sample：选中这一项会统计定时器(timer)的时间，否则只统计采样器(sample)的时间



六、If 控制器（If Controller）：

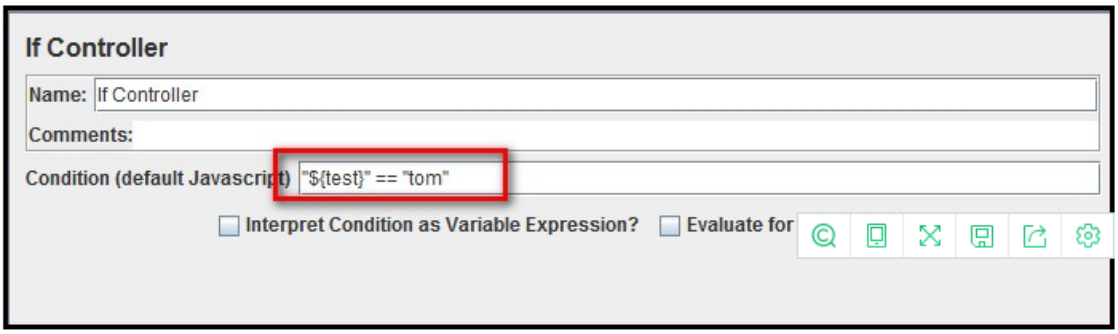
作用：根据给定表达式的值决定是否执行该节点下的子节点，默认使用javascript的语法进行判断(如下图红框内的文字)。



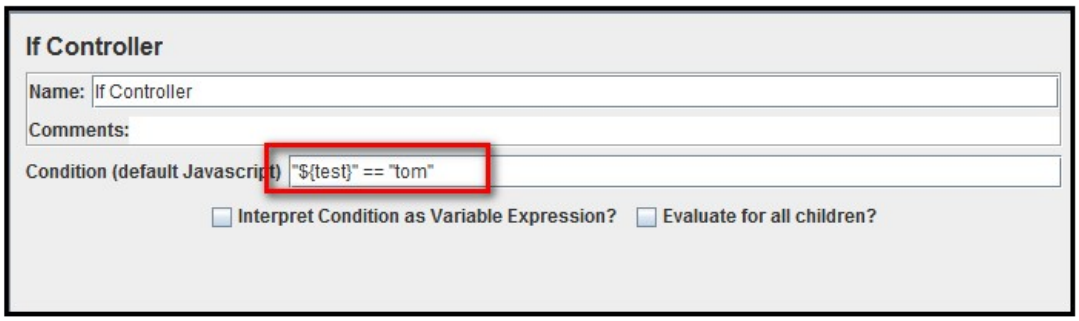
参数：

- Interpret Condition as Variable Expression?：选中这一项时表示：判断变量值是否等于字符串
- true（不区分大小写）
- Evaluate for all children：如果选中这一项，在每个子结点执行前都会计算表达式
-

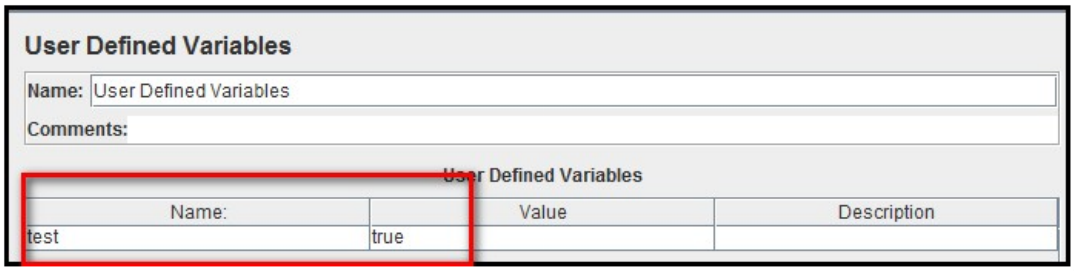
示例一：使用变量的方式进行判断：

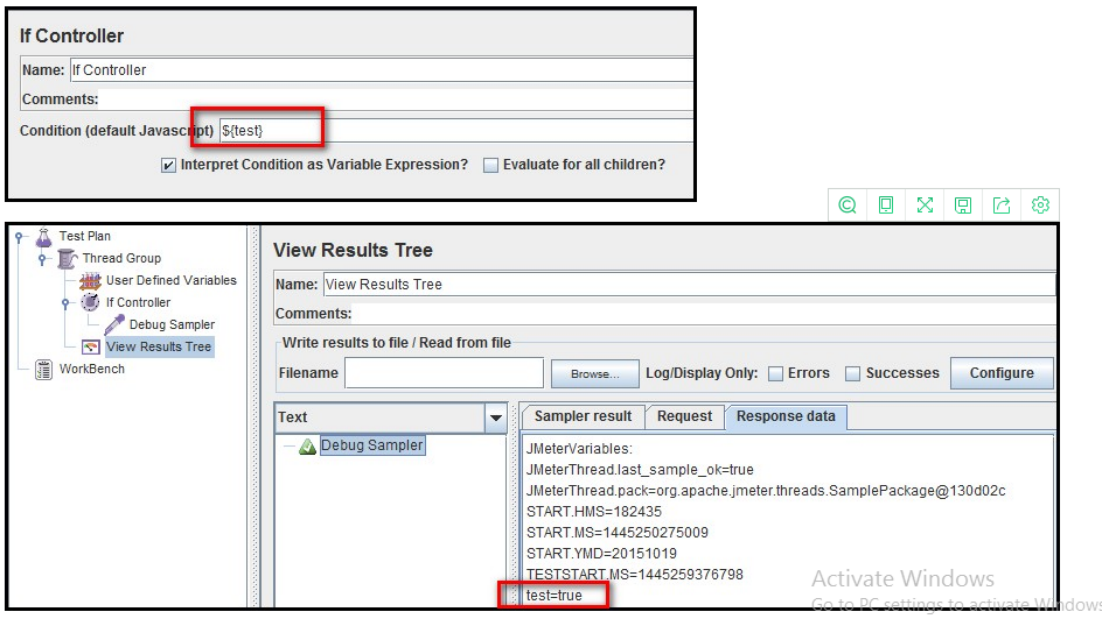


示例二：选中Interpret Condition as Variable Expression ?



示例二：选中Interpret Condition as Variable Expression ?





七、Switch控制器（Switch Controller）：

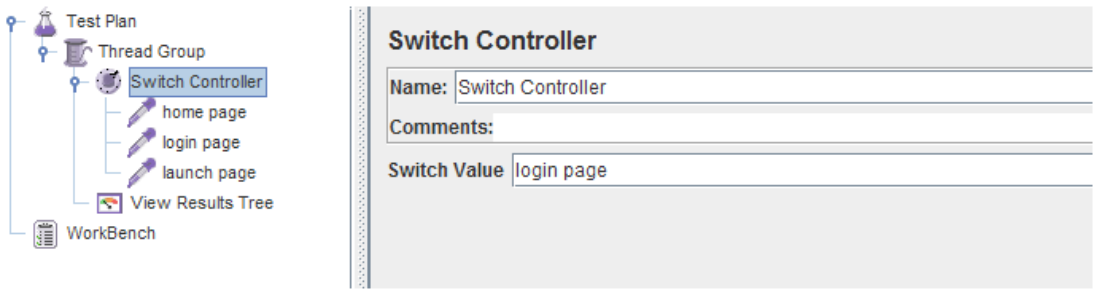
作用：Switch控制器通过给该控制器中的Value赋值，来指定运行哪个采样器。有两种赋值方式：

- 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。
- 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。

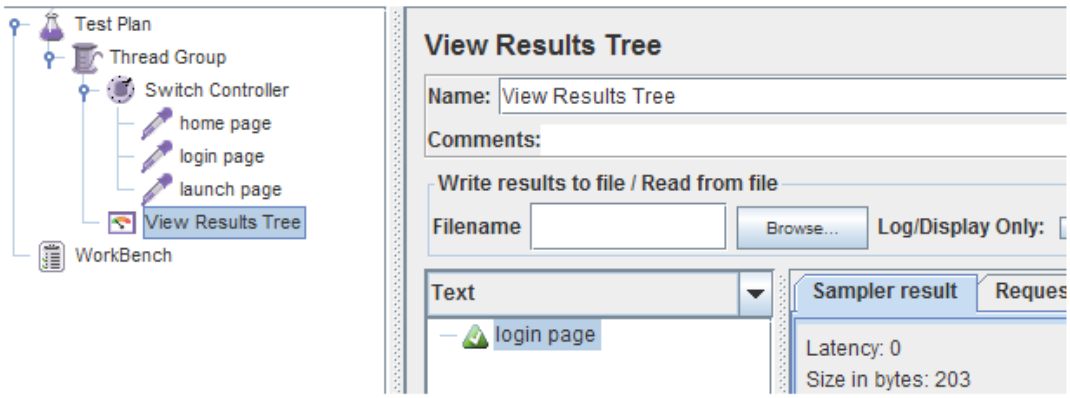
当Value为空时，默认执行第1个子节点元素。

示例：

1、Switch Controller选择的值为login page



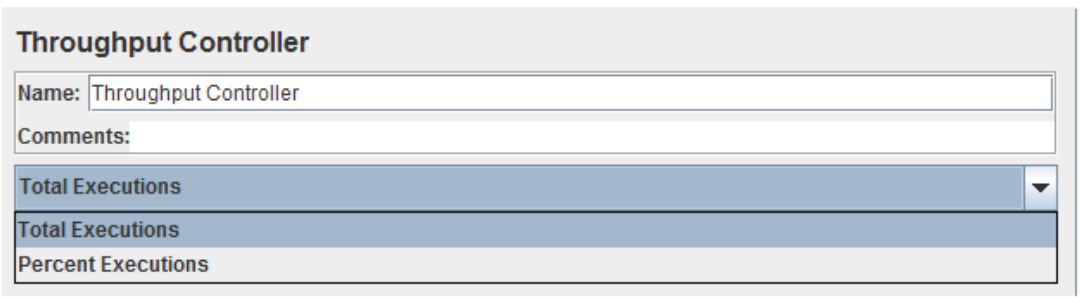
2、执行结果：



八、吞吐量控制器(Throughput Controller):

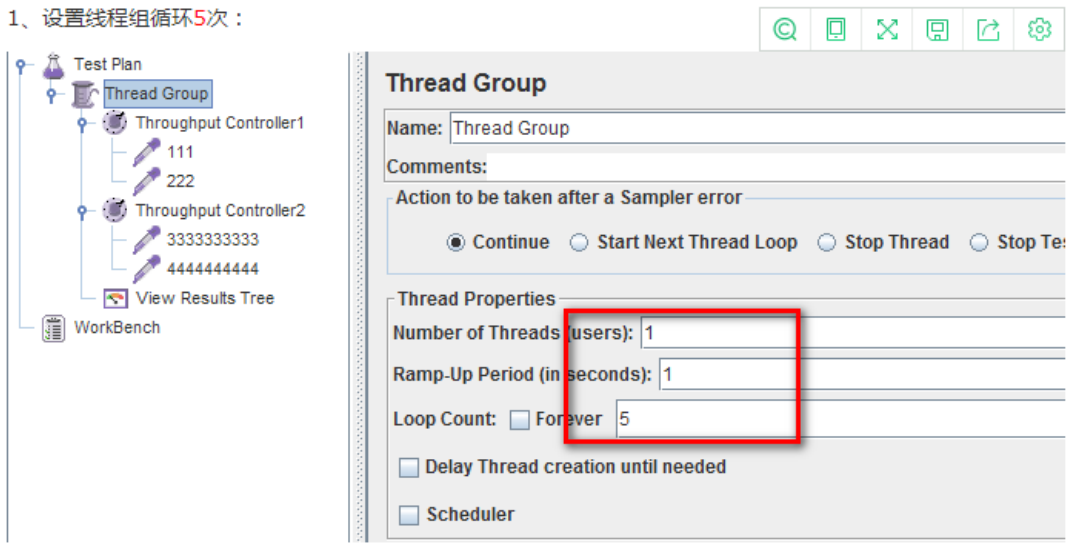
作用：控制其下的子节点的执行次数与负载比例分配，也有两种方式：

- Total Executions：设置运行次数
- Percent Executions：设置运行比例(1~100之间)

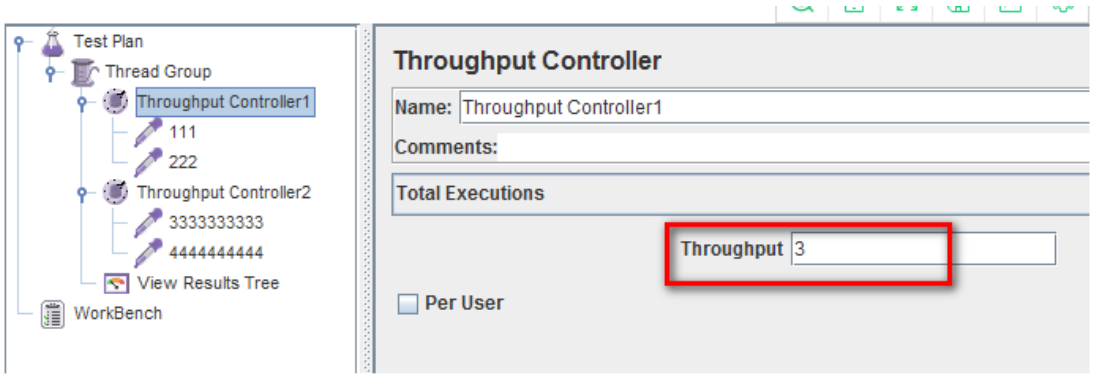


示例：

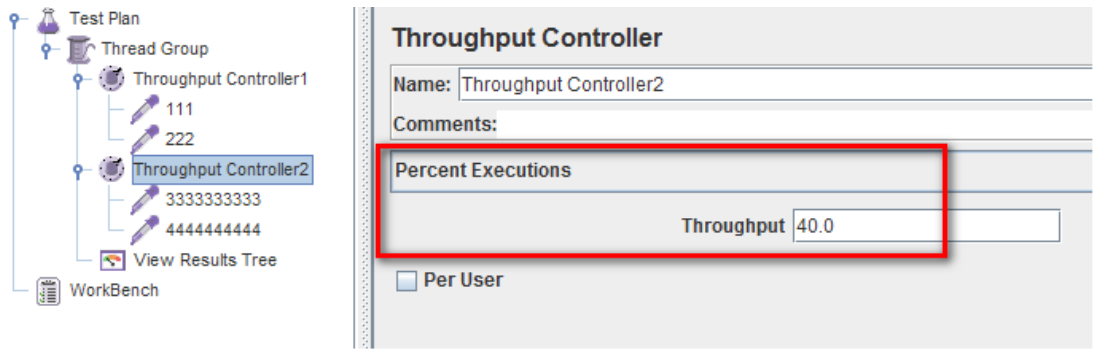
1、设置线程组循环5次：



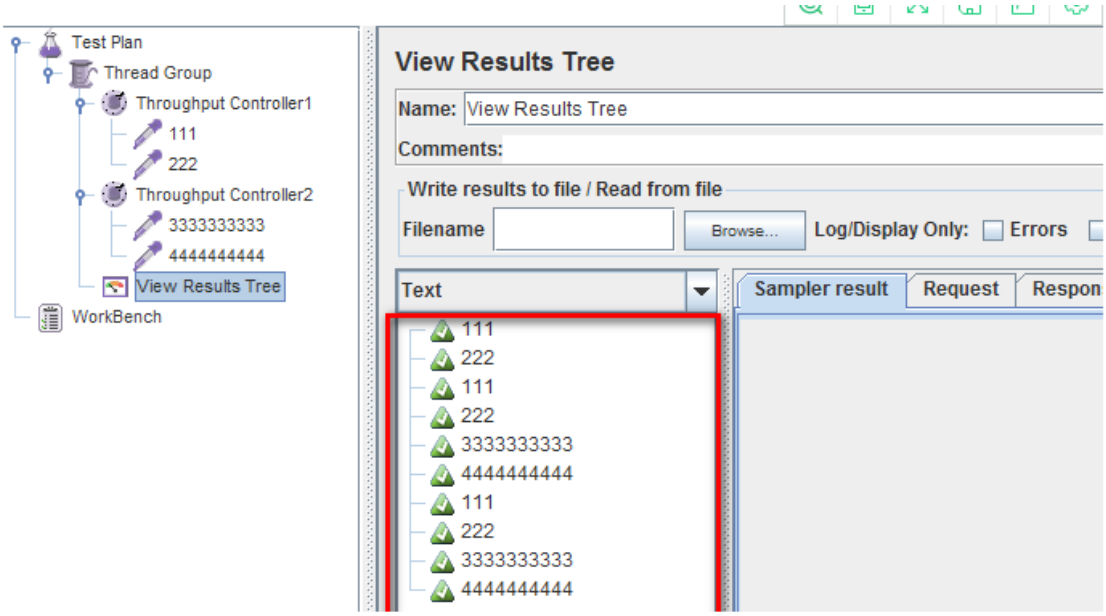
2、Throughput Controller1的子结点执行3次：



3、Throughput Controller2的子结点执行（40% * 线程组循环次数5）= 2次：



执行结果：



九、随机控制器(Random Controller):

作用：随机执行其下的所某个子结点

十、随机顺序控制器(Random Order Controller):

作用：随机执行其下的所有子结点