# 性能测试的指标

性能测试主要是为了帮助系统达到以下目的：

**多：**可以支持更多的用户访问

**快：**接口、页面响应速度快

**好：**访问都能成功，错误率低

**省：**在达到预期效果的前提下，尽可能节省资源

性能测试的指标包括：

**响应时间**

**并发用户数**

**吞吐量**

**系统性能计数器**

## 1.1响应时间

响应时间是指系统对请求做出相应所需要的时间

它包括以下几部分：

1. **客户端呈现时间**
2. **请求/响应的网络传输时间**
3. **应用服务器的处理时间**
4. **数据库服务器的处理时间**

那么响应时间应该多少合适呢？

对于一个web系统，普遍接受的响应时间标准为2/5/8秒3个级别。2秒以内用户体验很好，5秒用户可以接受，8秒是用户能接受的上线时间

## 1.2并发用户数

用户数主要分为：

**注册用户数：**系统注册的用户总数

**在线用户数：**某段时间内用户在线总数，即系统正在使用的人数，这部分人可能只在浏览(已登录)，不一定与服务器产生交互

**并发用户数：**同一时刻向服务器发送请求的用户数

一般来说，并发用户数和在线用户数的比例一般为5%~20%，这个比例是并发用户数和在线用户数的瞬时值之间的预估换算，那么平均值和峰值呢？

这里实际上是有一个估算公式：

C是平均并发用户数，Cmax是峰值并发用户数，n是T段时间内所有用户访问系统的次数，L用户单次登录使用系统的平均时间(即从登录到退出的平均时间)，T为考察的时间段长度(比如一天或8个小时)

C≈n\*L/T

Cmax≈C+3\*

eg：一个OA系统，共3000个用户，平均每天约有400个用户访问该系统，每个用户一般只在8小时内使用该系统，且从登录到退出系统的平均时间为4小时

C≈n\*L/T=400\*4/8=200

Cmax≈C+3\*=200+3\*=243

## 1.3吞吐量

吞吐量是指单位时间内，系统能够处理用户请求的数量

吞吐率是单位时间为1s下系统能处理的请求数，即吞吐量的速率

衡量吞吐量的指标主要是TPS和QPS:

TPS：事务每秒请求数

QPS：查询每秒请求数

PV：即page view，页面浏览量

例如：针对单个接口，TPS可以认为等同于QPS，而访问一个html页面会请求服务器3次，一次html，一次js，一次css，那么这次访问就产生了一次T和三次Q

计算公式：

当没有遇到性能瓶颈时（吞吐量在到达性能瓶颈后将不再上升）：

QPS=U\*R/T

U是用户数，R是每个用户数在测试期间发出的请求数，T表示测试时长

## 1.4系统性能计数器

主要是描述服务器的资源使用情况，如内存，CPU，磁盘资源等

## 1.5性能测试的场景提取

功能测试有一个覆盖率的概念，即尽可能将所有功能覆盖，那么性能测试是否有类似的概念呢？当然不是，一个系统各功能被用户使用的频率是不同的，通常我们只对核心/高频业务进行性能测试

下面哦我们举一个提取出来的性能测试场景案例：

比如电商网站查询订单：

场景操作流程：

1. 用户访问首页——登录——每个用户一次
2. 查看订单——点击订单管理

场景性能指标：

1. 验证最大在线用户数
2. 错误率<0.5%
3. 请求响应时间<5s
4. 场景运行10分钟（不加同步定时器，即不需要瞬时高并发）
5. 服务器资源cpu使用率<80%，内存使用率<80%

# jmeter

## 2.1环境安装



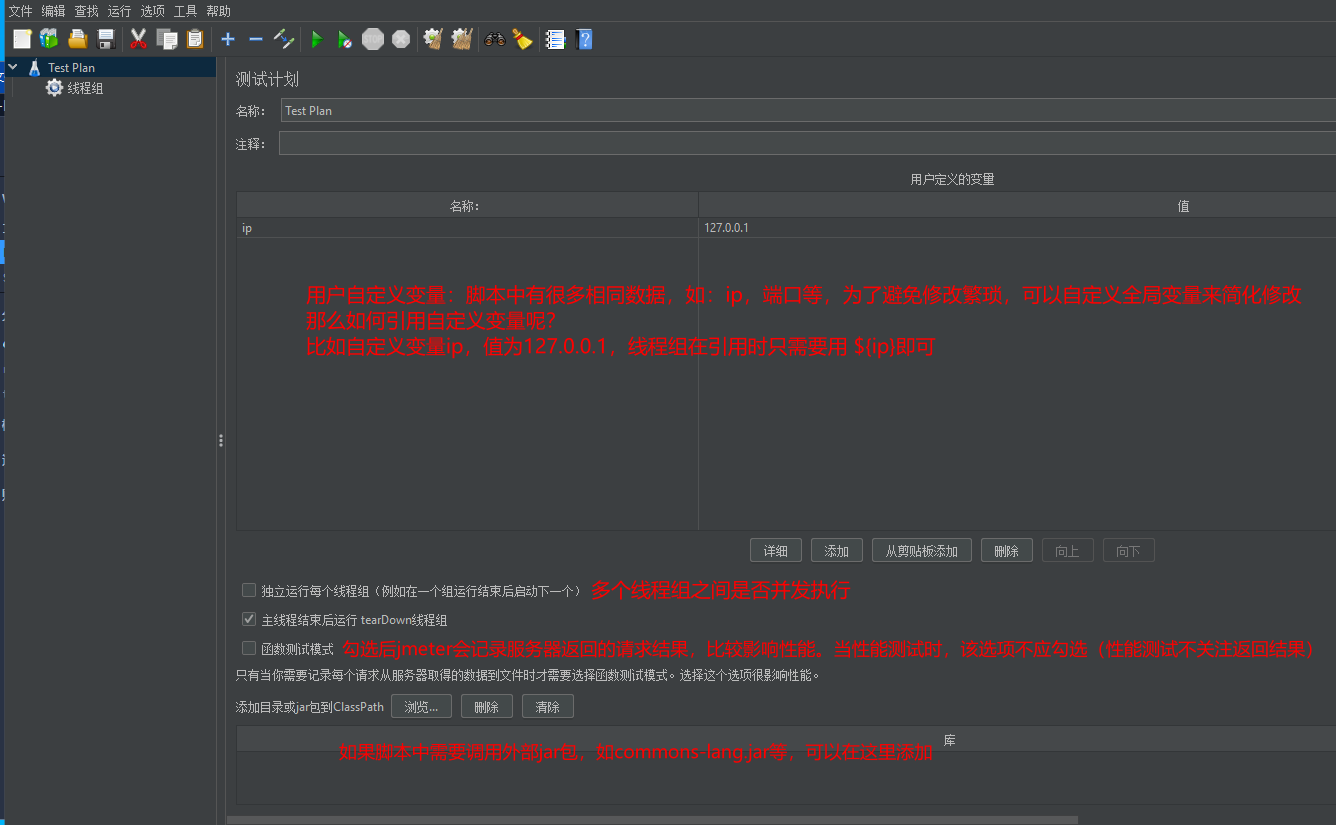
jmeter提示我们不要使用GUI模式去进行测试，这是因为GUI界面对性能会有消耗，GUI界面应当只用于创建和调试Test用例

jmeter是基于java研制的，因此我们需要设置其堆大小

## 2.2测试计划Test Plan

测试计划是jmeter中最大的一个元件，它是其他元件的容器，其他元件如线程组等都会放在Test Plan下

测试计划下可以包含多个线程组，多个线程组之间可以并行执行，也可以顺序执行

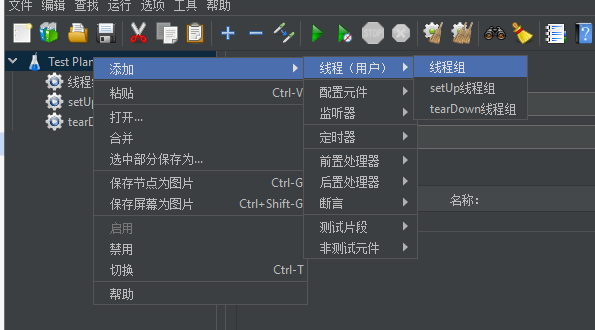


## 2.3线程组

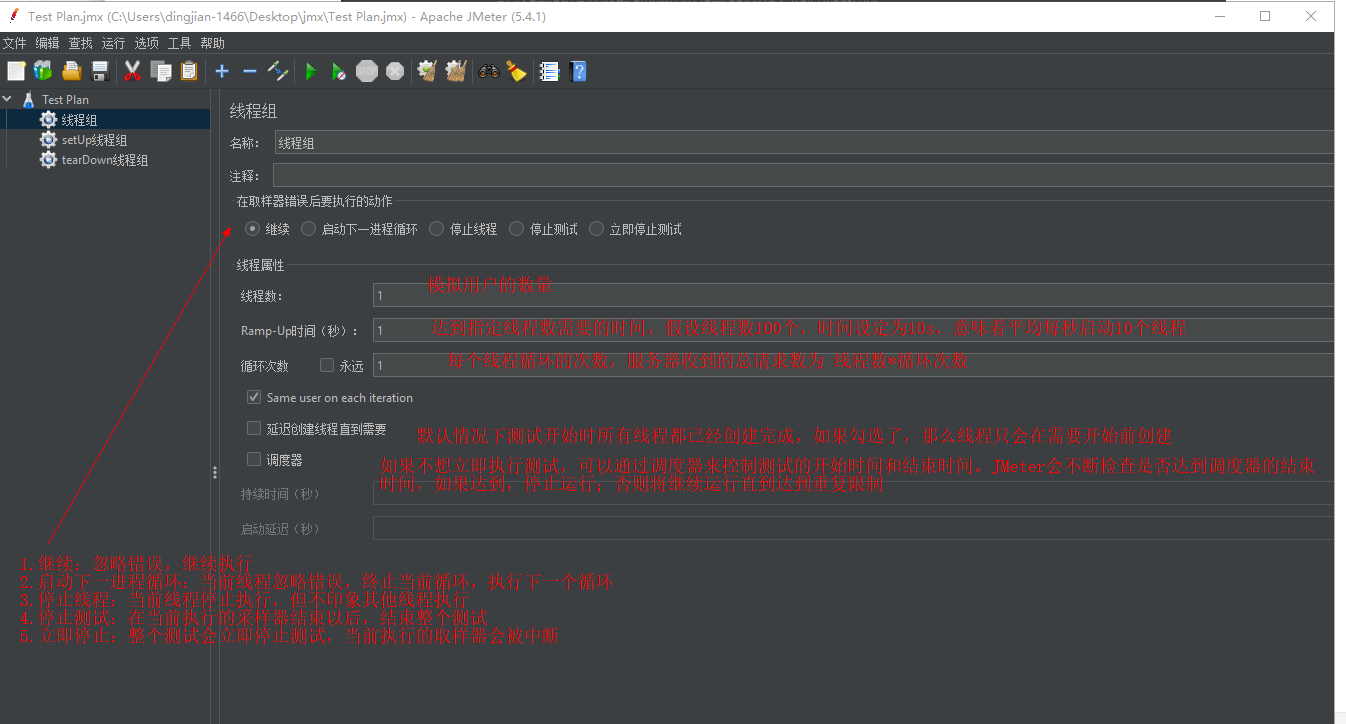
线程组中可以有多个线程，每个线程就是一个”虚拟用户”，因此线程组可以看作一个虚拟用户组

线程组主要分为三种：

1. setup线程组：用于测试前的初始化操作。比如打开数据库连接、用户登录等操作
2. 线程组：用于执行测试操作，脚本都是放在线程组中才能执行的
3. teardown线程组：用于退出测试后的环境清除。比如关闭数据库连接、用户退出登录等

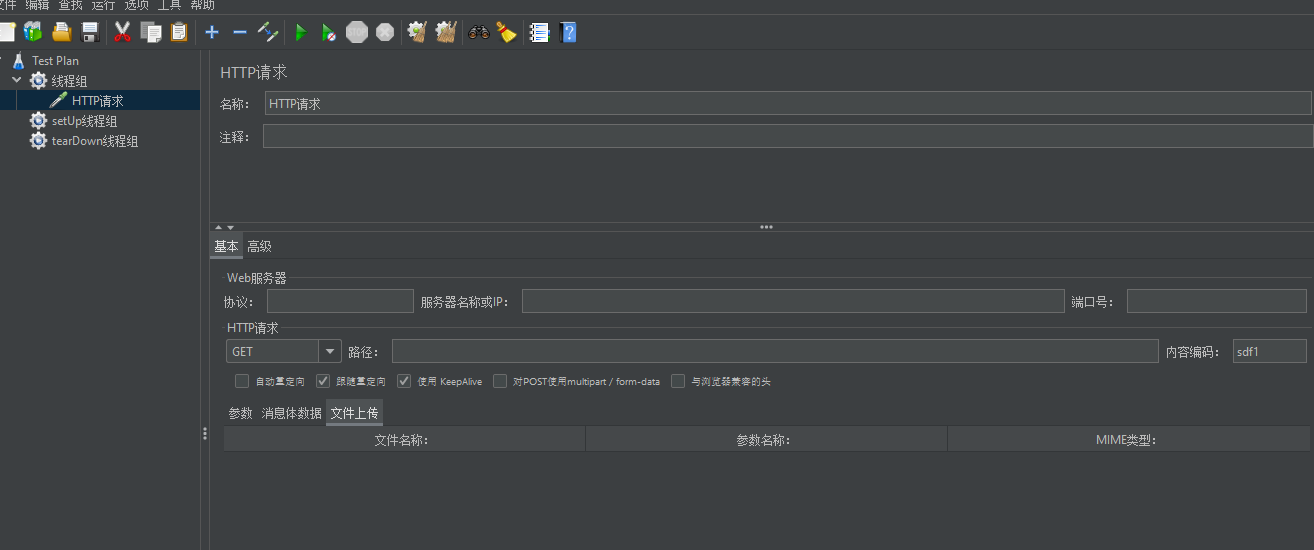


无论这三种线程组如何放置，执行顺序永远是setup线程组，普通线程组，teardown线程组



## 2.4采样器

采样器是允许jmeter将多种类型的请求发送到服务器的组件，它被用于模拟用户对目标服务器的请求。采样器支持多种协议，如HTTP，FTP，JDBC，websocket(需要下载插件)等，下面以最常用的HTTP为例：



## 2.5脚本开发

### 2.5.1HTTP代理服务器录制

jmeter支持录制的方式，通过jmeter代理录制操作来开发脚本

1. 右键点击测试计划，创建线程组
2. 右键点击测试计划，创建“非测试元件”—“HTTP代理服务器”
3. 设置“HTTP代理服务器”的目标控制器为刚刚创建的线程组
4. 打开浏览器，设置代理，代理ip地址是127.0.0.1，端口号默认8888
5. 回到jmeter界面，设置排除/包含模式，避免录制到不需要的资源如.js.css等
6. 点击启动，此时已经开始录制
7. 访问系统页面，执行要录制的操作

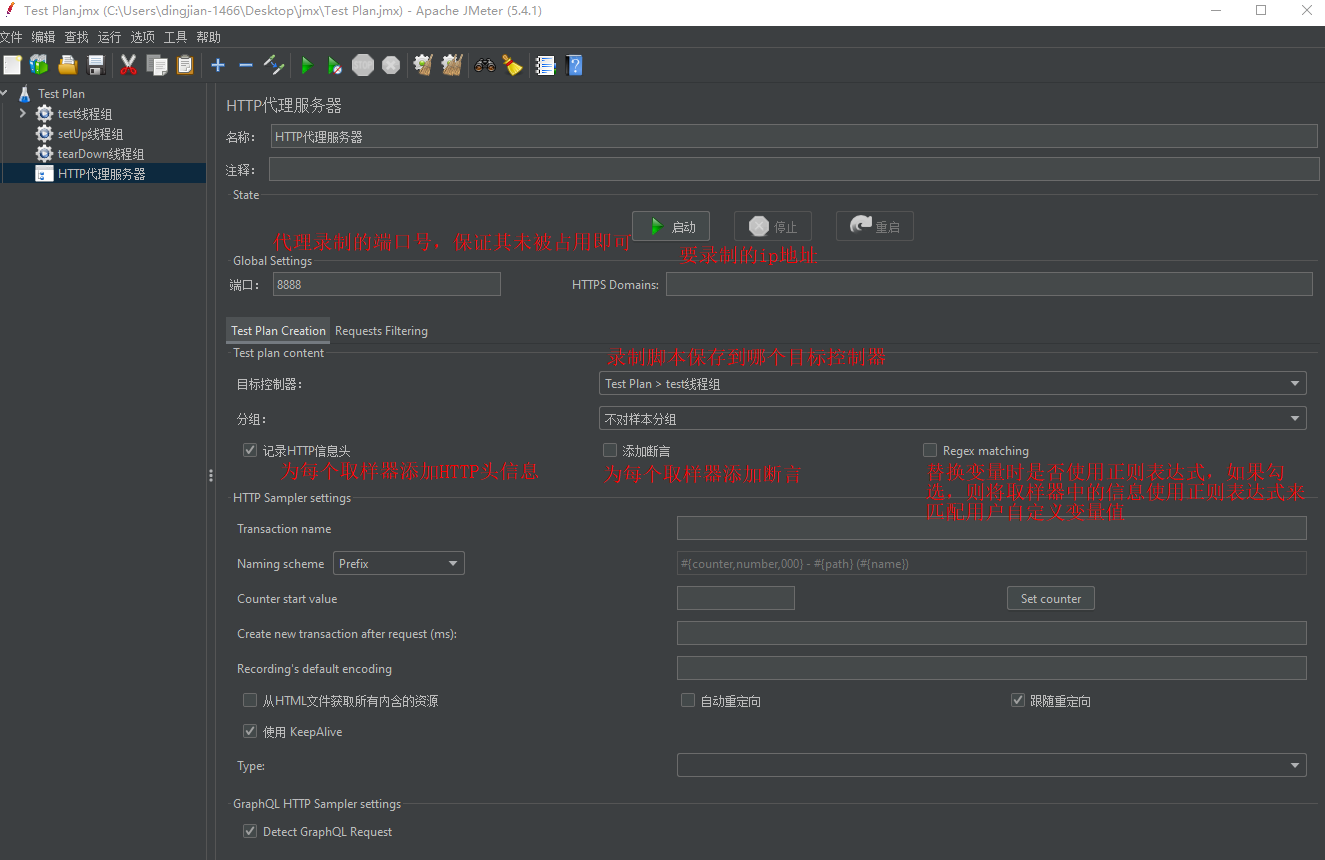


图 2-5-1

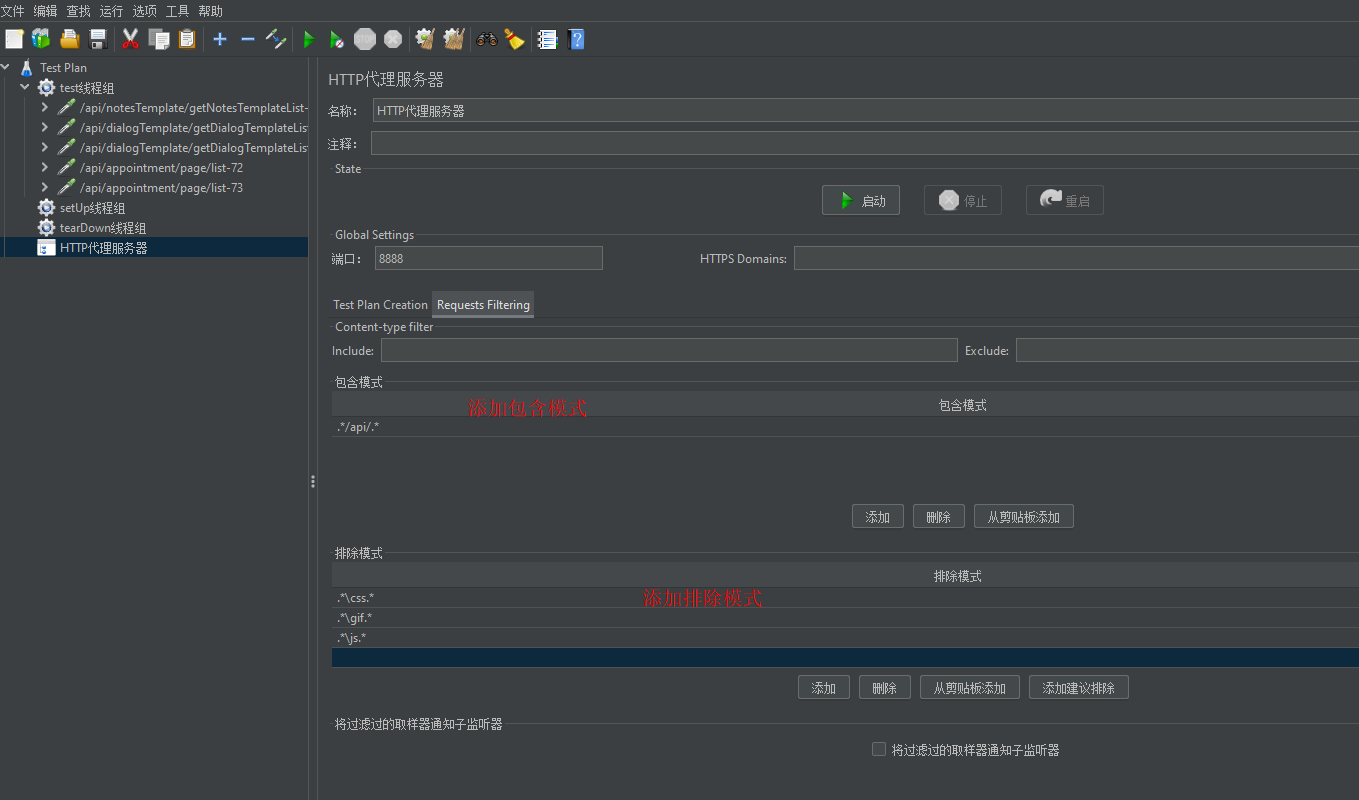


图 2-5-2

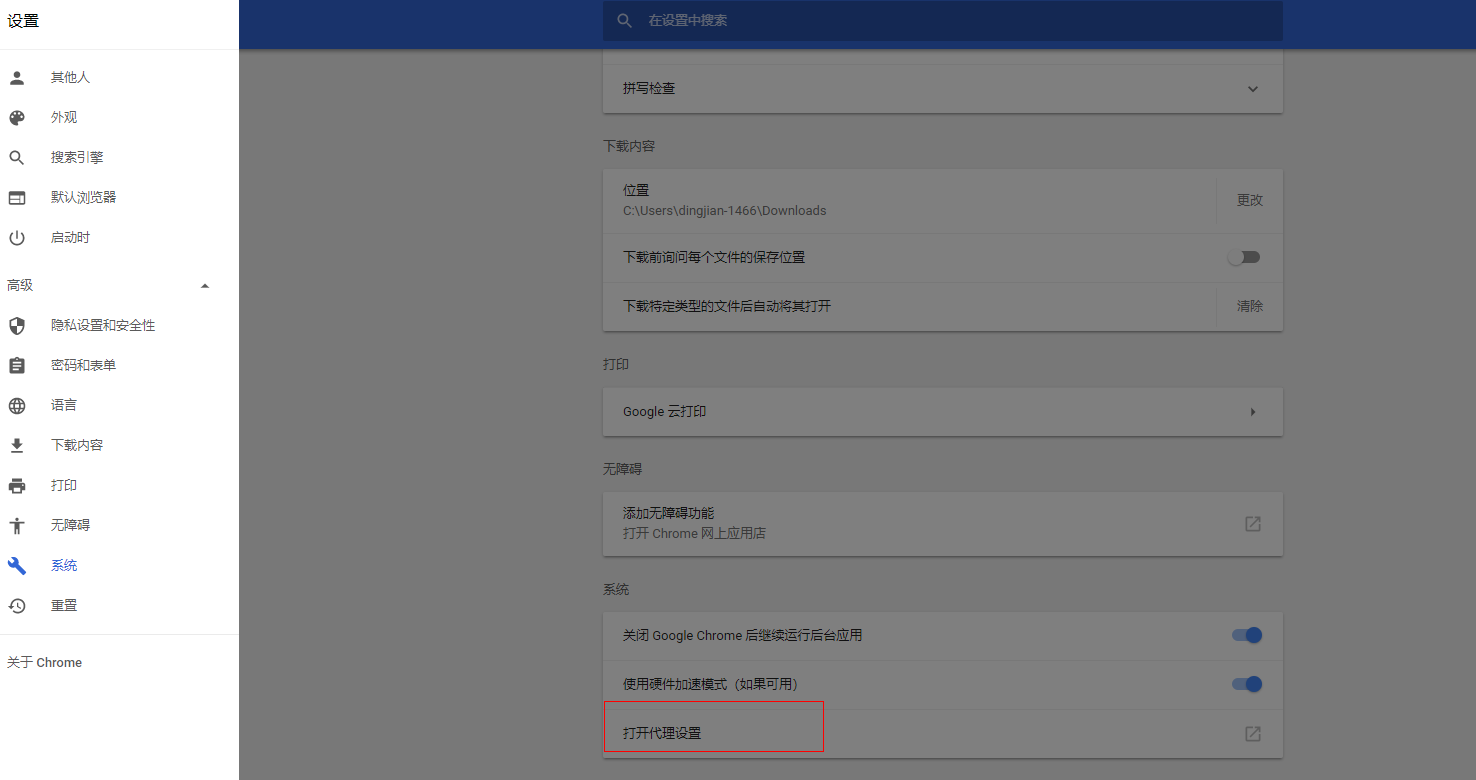
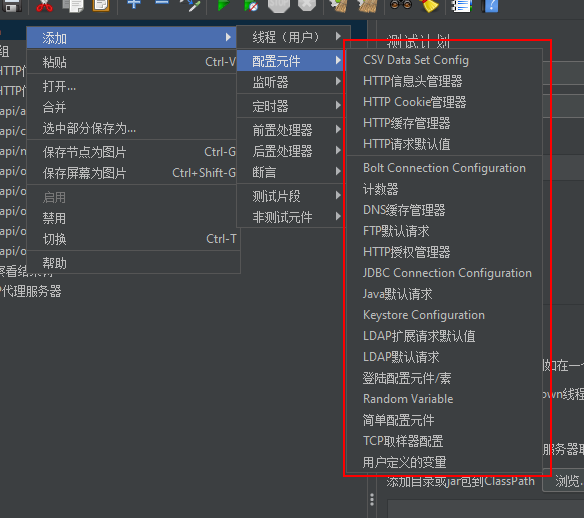


图 2-5-3



图 2-5-4

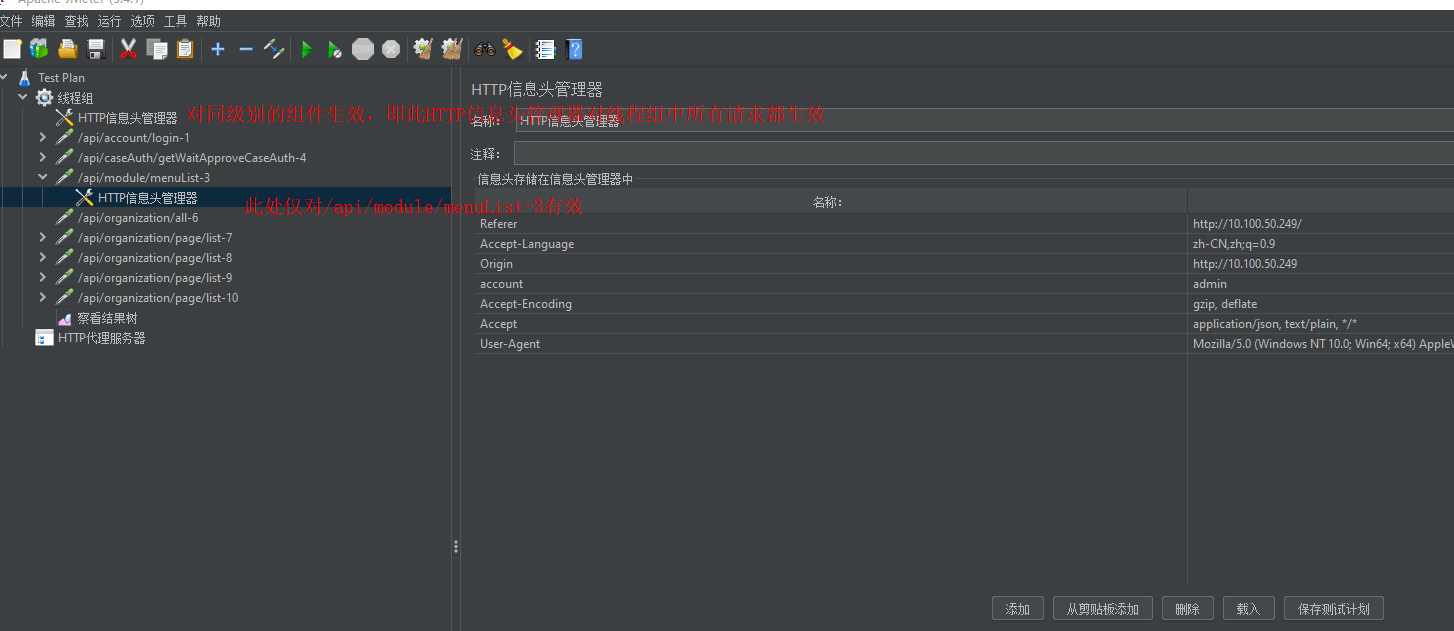
## 2.6配置元件



### 2.6.1HTTP消息头管理器

刚才我们录制了多个HTTP请求，他们各自有各自的请求头，那么如何对这些请求头进行统一管理呢？比如我们希望给每个http头都增加一个参数，该怎么操作？

我们可以添加一个HTTP信息头管理器来对http头进行管理，如下：



需要注意Http头管理器将会对 与自己同级别的组件 生效。

如果同一层级配置了多个HTTP头管理器，请求头参数取多个头管理器的全集，重复的请求头参数以第一个HTTP头管理器为准

### 2.6.2cookie管理器

cookie管理器可以模拟web浏览器对cookie进行存储和发送，帮助我们自动管理cookie，如果一个http请求的响应中带有cookie，cookie管理器将自动保存该cookie并在所有后续发送到该站点的请求中使用该cookie，每个线程都有自己的cookie存储区

### 2.6.3HTTP请求默认值

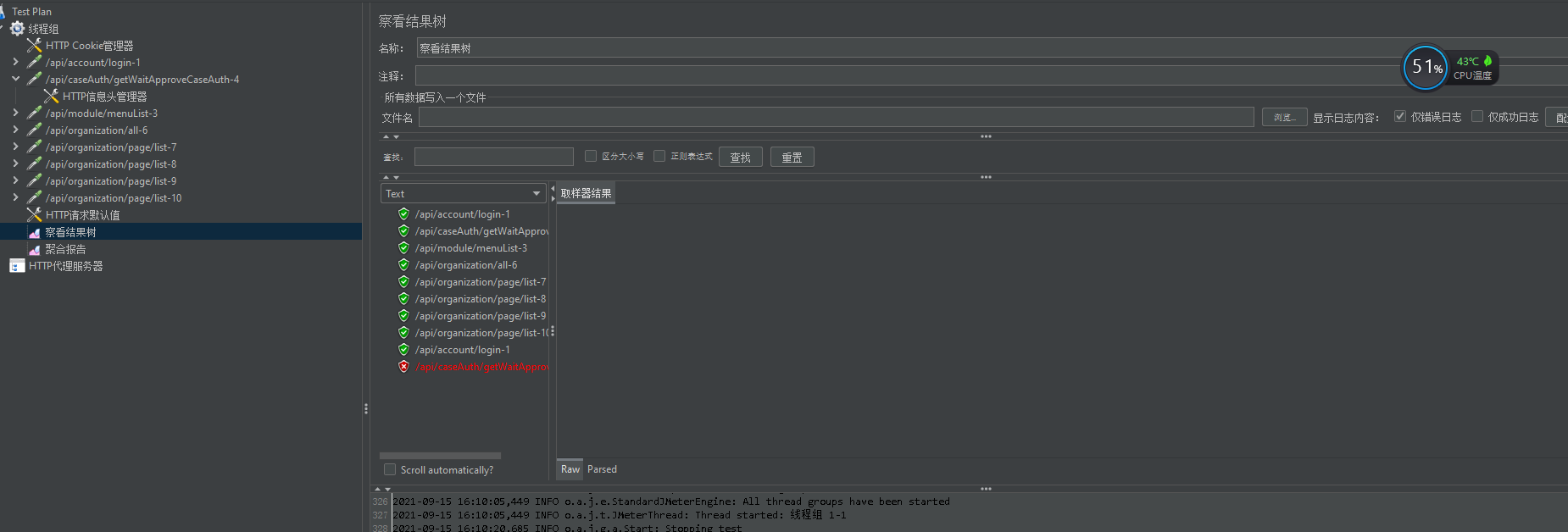
主要用于填写接口中重复出现的参数，比如域名、端口号等，这样在编写多个http接口时，不需要重复填写这些参数

请求默认值中的参数优先级是最低的，如果在http采样器中设置了重复的参数，以http采样器的为准

## 2.7监听器

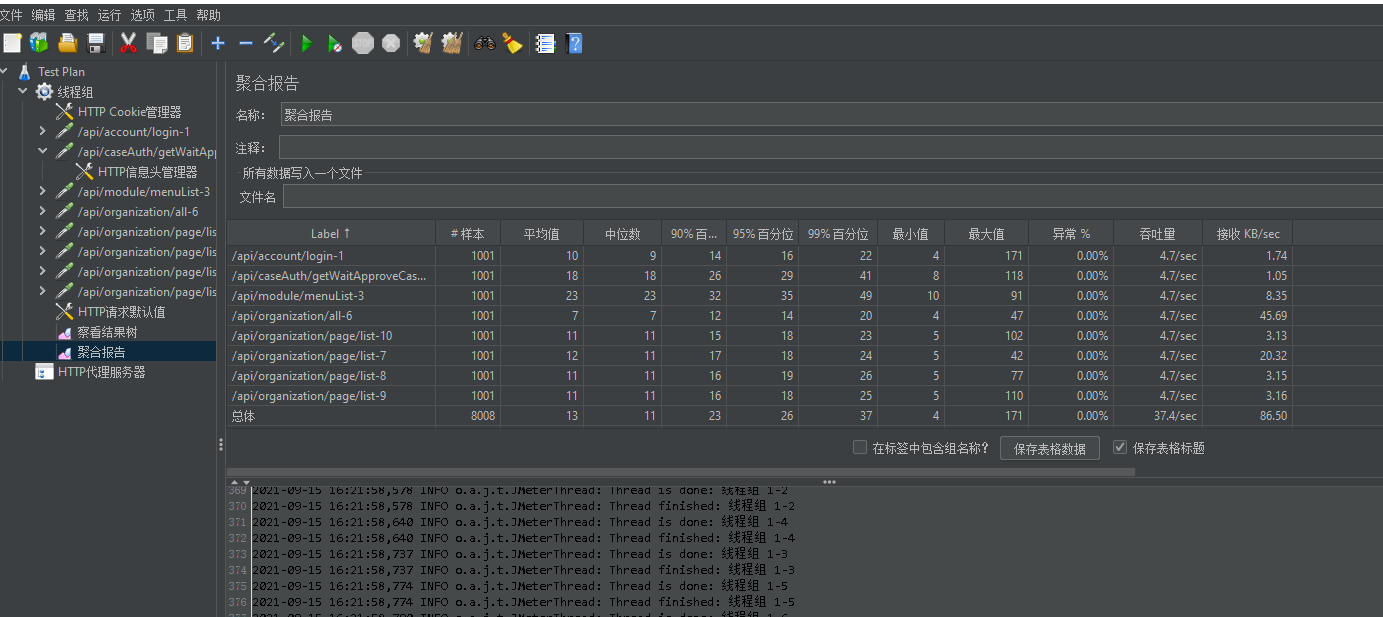
### 2.7.1查看结果树

查看结果树可以看到每个请求的请求过程和结果，包括请求头、请求体、响应头、响应体等，查看结果树还可以筛选出错误的请求，防止因请求次数太多而找不到错误的请求



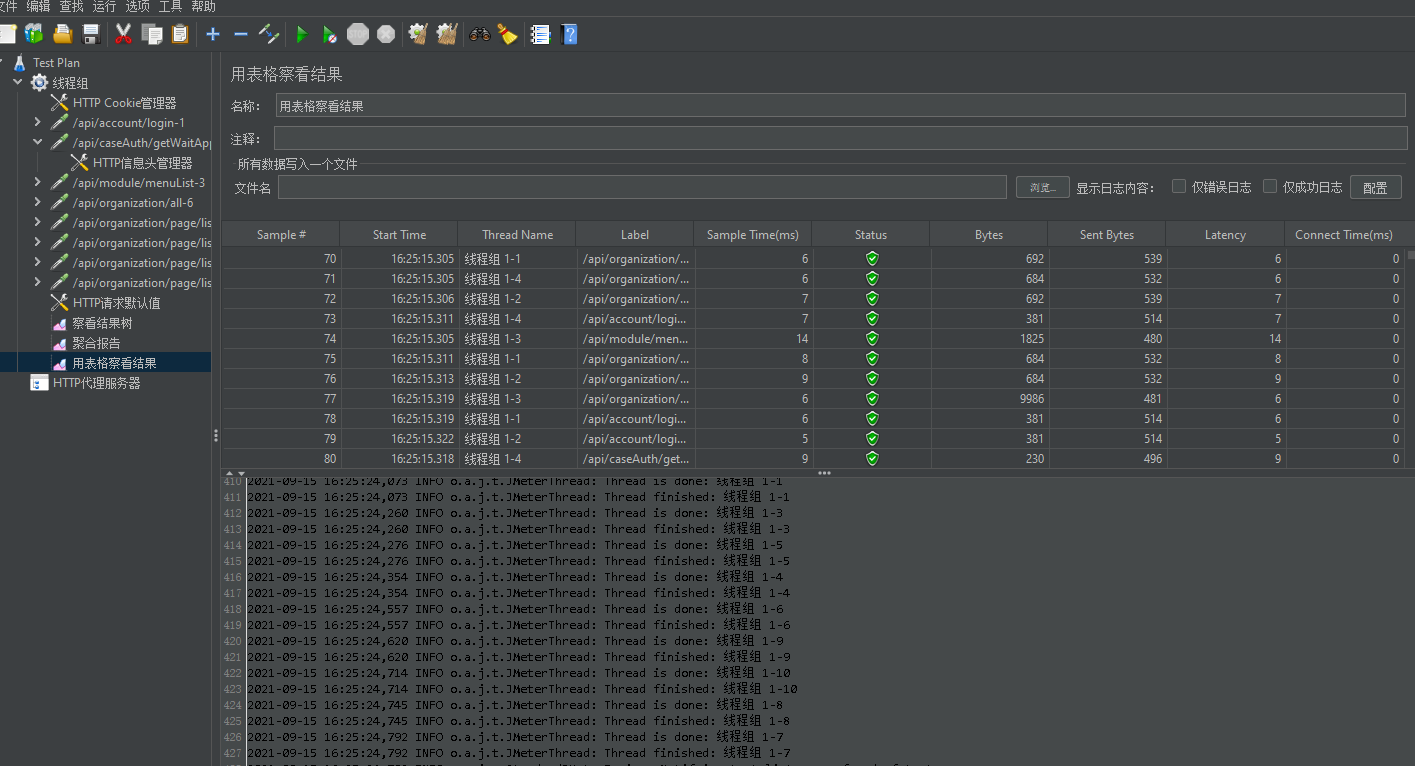
### 2.7.2聚合报告

对于运行的结果做总结汇总，单位是ms：



### 2.7.3用表格查看结果

用表格查看结果将会列出每一个线程的每一次循环的详细内容：

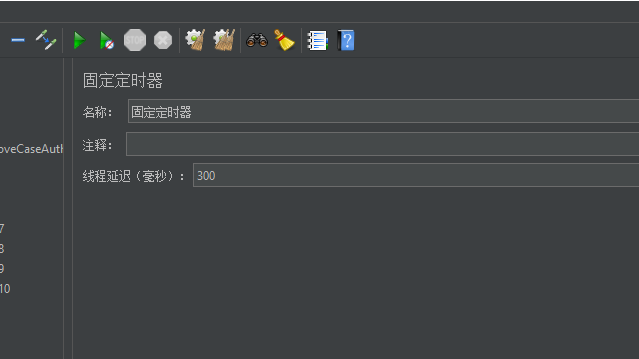


## 2.8定时器

jmeter并发测试时每个请求之间是没有延迟的，那么实际场景中用户点击不会那么快，就需要使用定时器来模拟这些场景

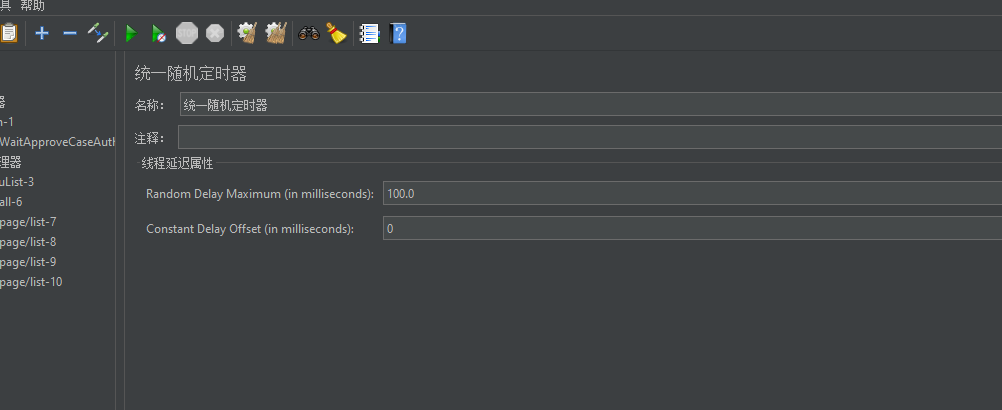
### 2.8.1固定定时器

这是比较常用的一种定时器，通过ThreadDelay设定每个线程请求之前的等待时间，单位为毫秒。如果希望定时器对所有请求有效，就将定时器放在线程组下；如果希望仅对单个请求有效，则将定时器放在对应请求下



### 2.8.2统一随机定时器

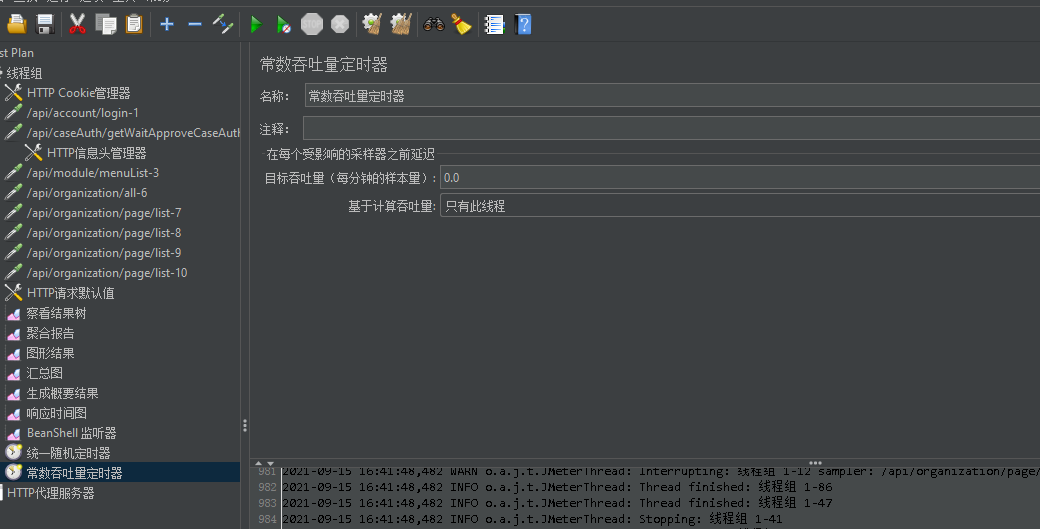
其产生的延迟时间是一个随机值，总延迟时间=随机延迟时间+固定offset延迟时间



### 2.8.3固定吞吐量定时器

按指定吞吐量执行，以每分钟为单位。目标吞吐量指每分钟发送的请求数，可以选择作用的线程，比较常用的三个取值如下：

1. **只有此线程：**设置每个线程的吞吐量。总吞吐量=线程数×该值
2. **当前线程组的所有活动线程：**吞吐量被分摊到当前线程组的所有活动线程上
3. **所有活动线程：**吞吐量被分配到所有线程组的所有活动线程上

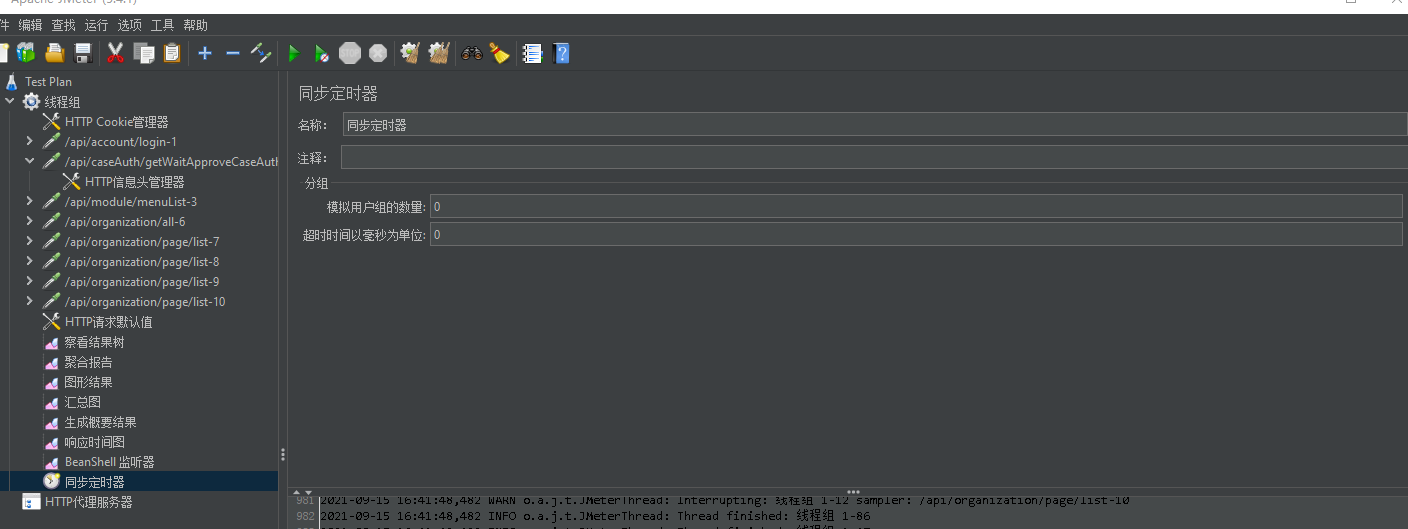


### 2.8.4同步定时器

用来设置集合点，阻塞线程，直到指定线程数量到达后，再一起释放，可以瞬间差生很大的压力，用来模拟秒杀等场景。

**模拟用户数量：**即同时释放的线程数量，取值范围应当是[0,线程数]。若设置为0，则等同于设置为线程组中的线程数量；若设置的值大于线程数量，则线程永远在等待，不会执行。

**超时时间：**超过多少毫秒后同时释放线程数，若设置为0，则定时器将会等待线程数达到设置的模拟用户数量后再释放；若设置大于0，如果超过超时时间后线程数还未达到模拟用户数，定时器将不再等待，直接释放



## 2.9参数化技术

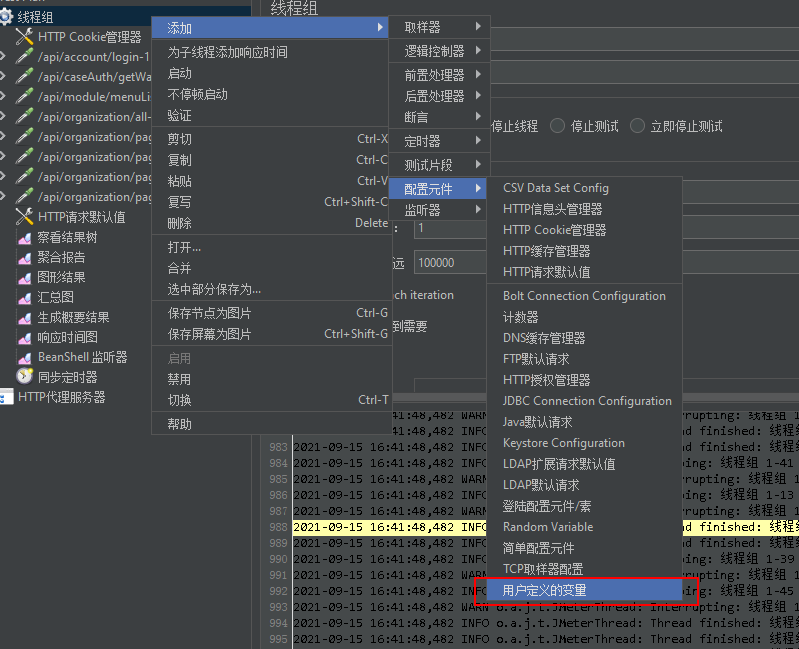
参数化技术主要是为了应对如下场景：

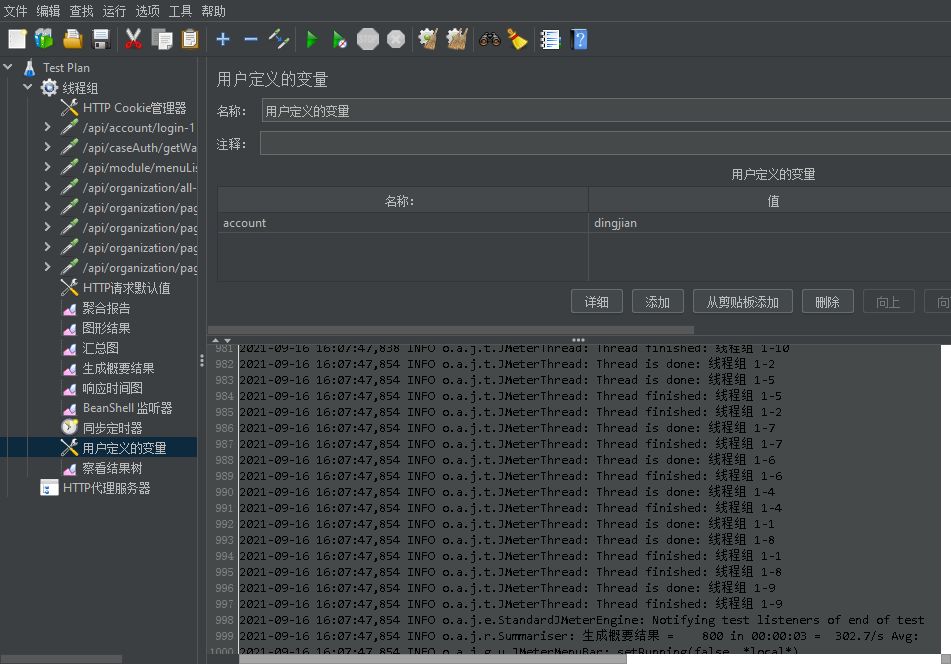
1. 多个请求使用同一个ip地址，若更换服务器ip，则每个请求都要修改ip地址
2. 注册账号，如果希望批量注册账号，但又不希望用户名重复
3. 模拟多个用户登录，需要用到不同的用户信息，可请求中的用户登录信息是写死的
4. 上一个请求的结果用于下一个请求的参数传入时

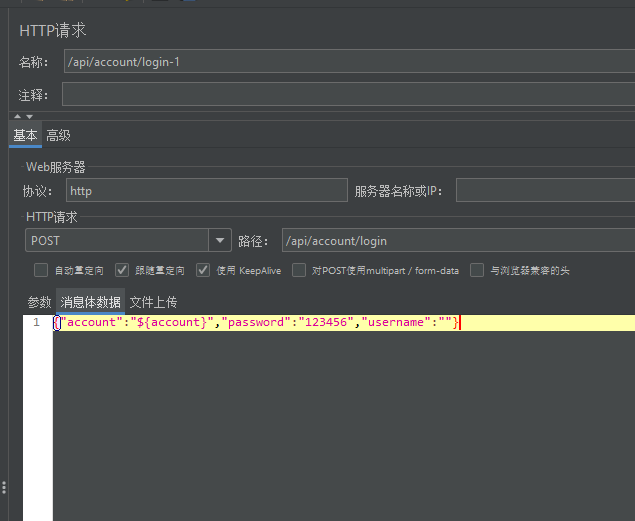
**在jmeter中，用${变量名}来获取参数值**

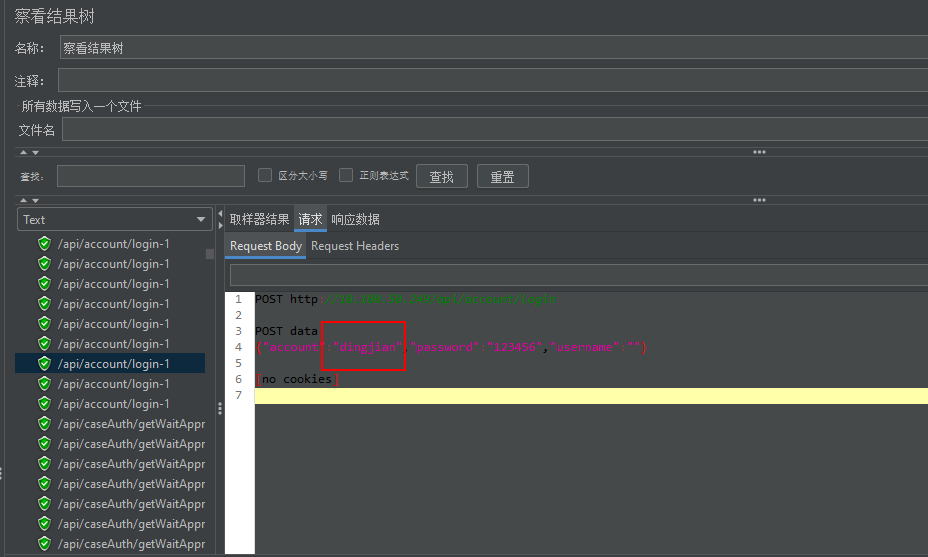
那么jmeter中有几种参数化方式呢？主要有如下几种：

### 2.9.1用户定义的变量

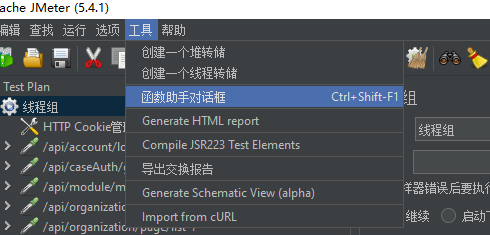


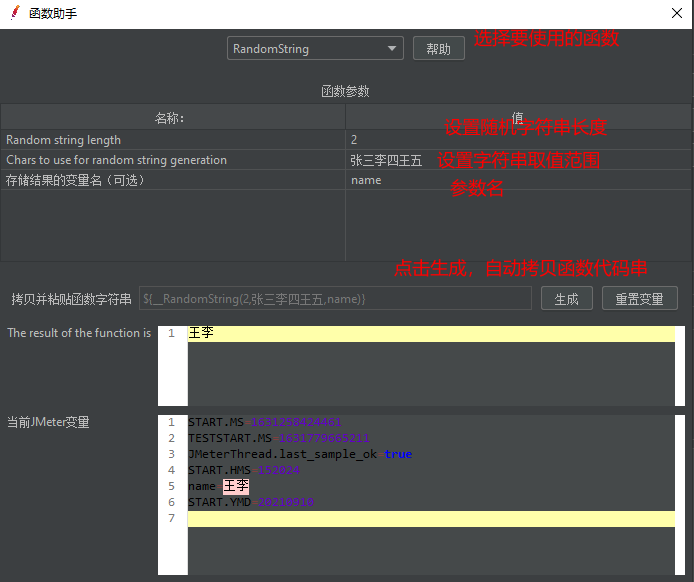






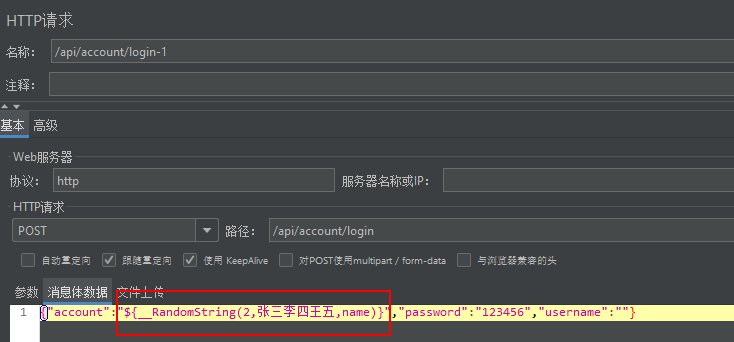
### 2.9.2函数助手





点击生成，自动拷贝函数字符串，用于引用

使用图例：

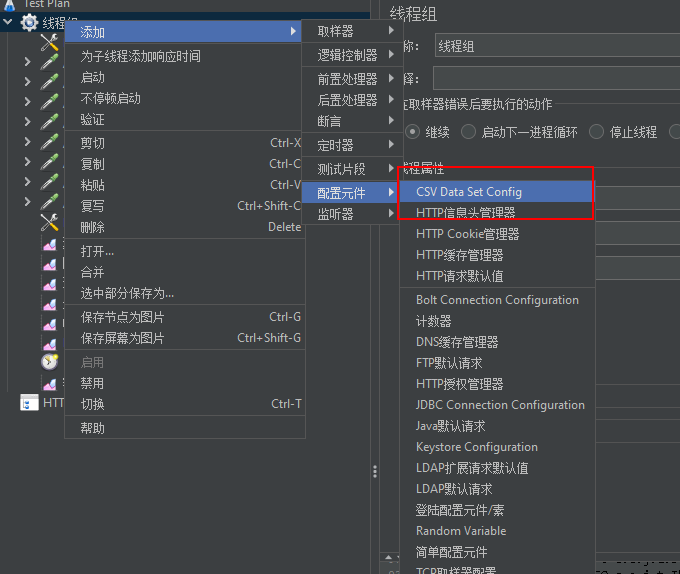


### 2.9.3 csv数据文件

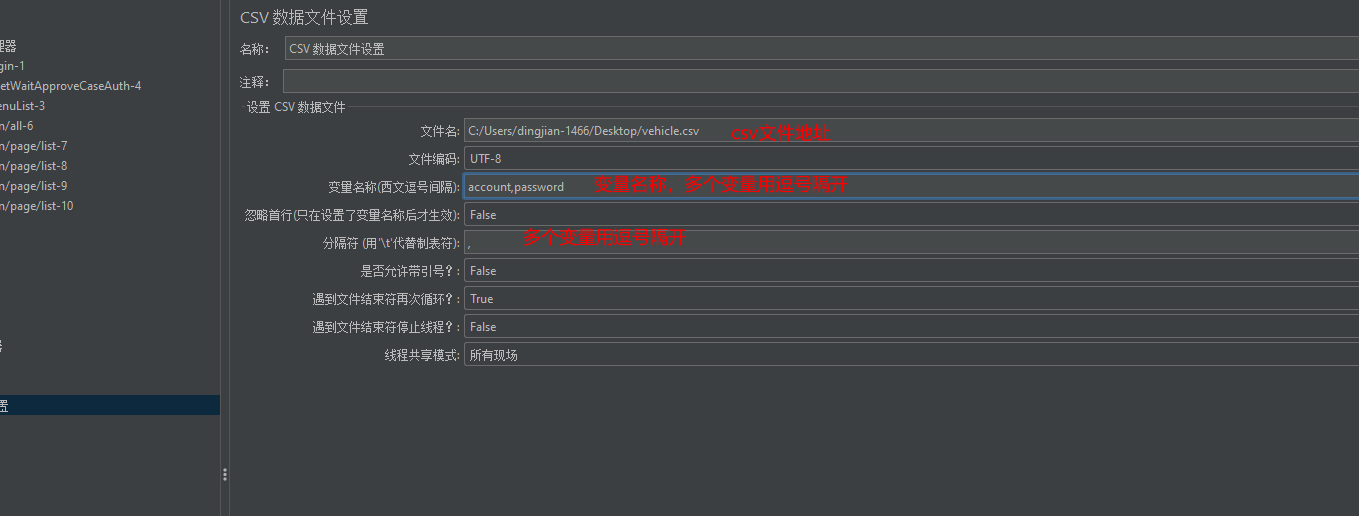
1. 准备csv参数化文件

可以从数据库导出csv格式文件，也可以自己创建csv格式文件

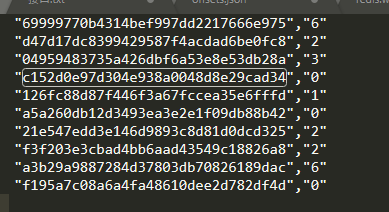
1. jmeter创建CSV Data Set Config元件

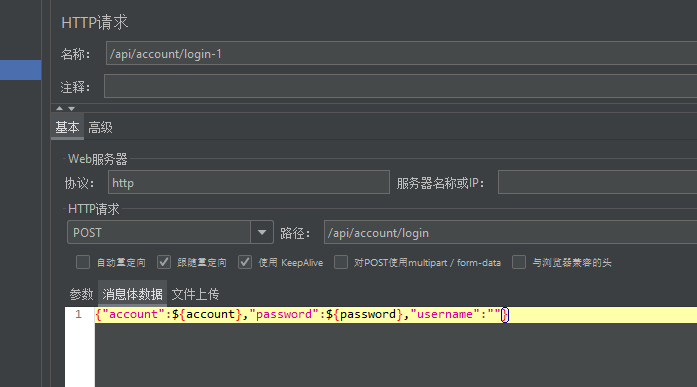


1. 填写csv配置信息



csv文件如下（这里带了双引号，因此变量值也会带双引号）：





## 2.10断言

断言是用于判断接口返回的响应结果是否正确的一种元件

### 2.10.1响应断言（常用）

响应断言可以对各种返回类型的结果进行断言，比如html、json等



## 分布式使用

单机使用jmeter会有性能瓶颈，单台机器能承受的用户数是有限的，因此jmeter支持分布式使用

