

第十章节 性能测试工具篇

1. JMeter介绍


环境要求：

JMeter is compatible with Java 8 or higher. We highly advise you to install latest minor version of your major version for security and performance reasons.

Apache *JMeter* 是 Apache 组织基于 Java 开发的压力测试工具，用于对软件做性能测试

1.1 安装JMeter

1. 下载tar包，解压即可。

 **apache-jmeter-5.5.tgz**
81.52MB

👁

解压完成后：

D:\software\apache-jmeter-5.5\apache-jmeter-5.5

名称

backups

bin

docs

extras

lib

licenses

printable_docs

LICENSE

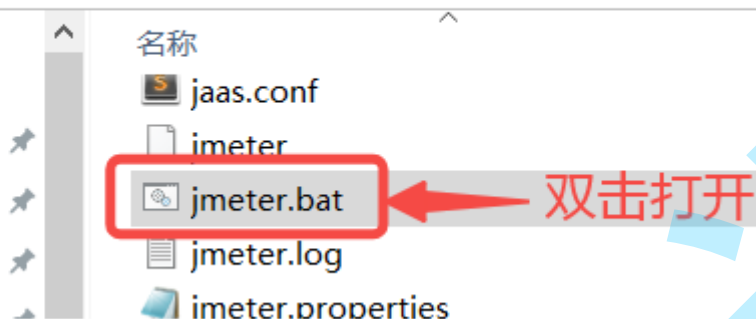
NOTICE

README.md

1.2 打开JMeter

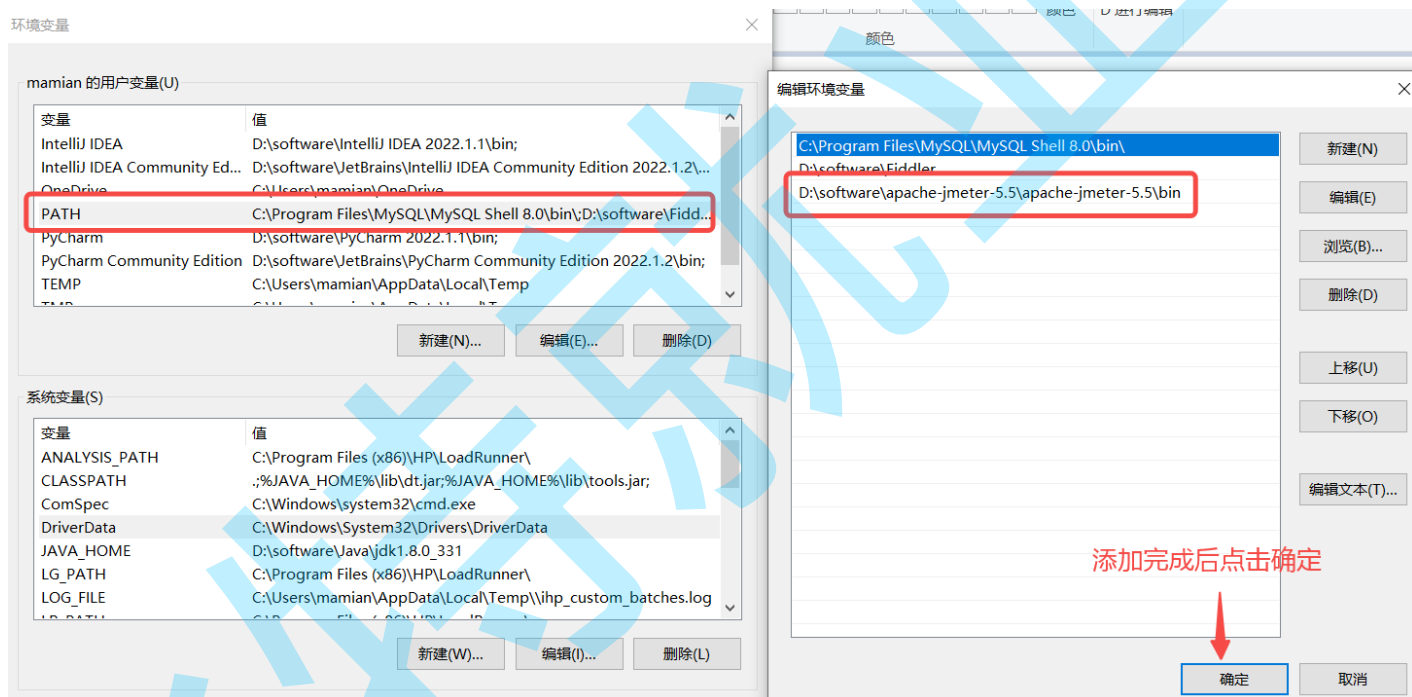
方式一：点击bat文件

D:\software\apache-jmeter-5.5\apache-jmeter-5.5\bin



方式二：命令行启动（推荐）

step1: 添加JMeter系统环境变量



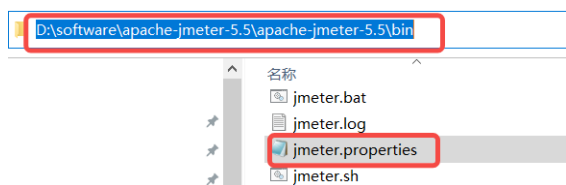
step2: 保存后打开命令行工具



输入命令jmeter即可启动JMeter工具

1.3 JMeter基础配置

- 修改字体为中文



```
# http://jmeter.apache.org/usermanual/properties_reference.html
# A local copy can be found in
# printable_docs/usermanual/properties_reference.html

#Preferred GUI language. Comment out to use the JVM default locale's language.
language=zh_CN
```

在jmeter的bin目录下，修改文件中的内容：language=zh_CN

1.4 JMeter基本使用流程

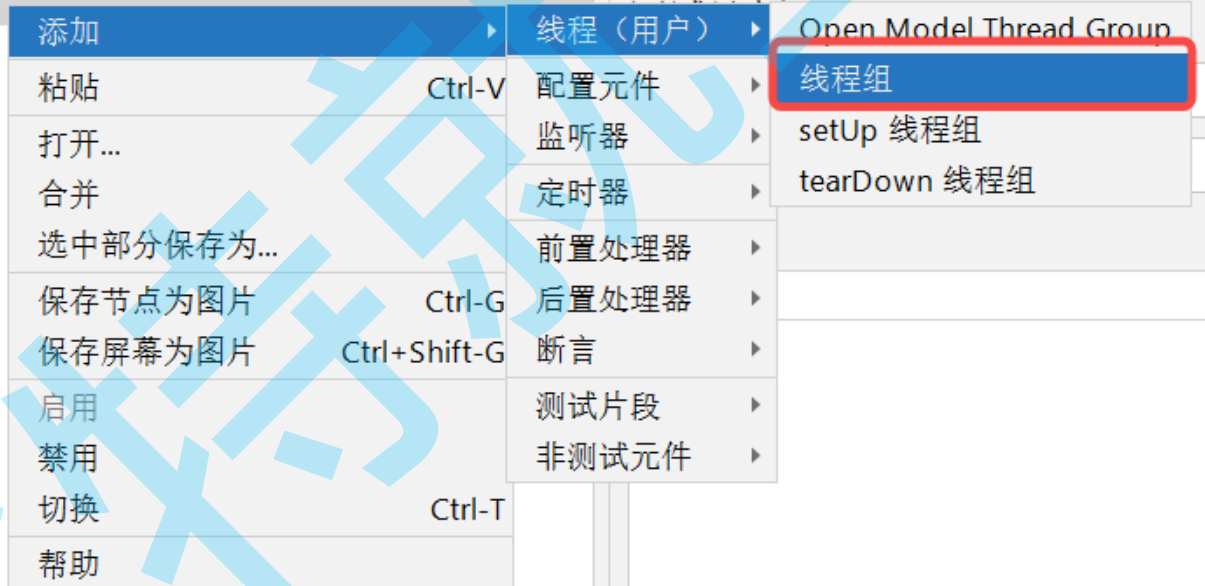
- 1) 启动JMeter
- 2) 在“测试计划”下添加“线程组”

Apache JMeter (5.5)

文件 编辑 查找 运行 选项 工具 帮助



Test Plan



- 3) 在“线程组”下添加“HTTP”取样器

Apache JMeter (5.5)

文件 编辑 查找 运行 选项 工具 帮助



Test Plan

线程组

线程组

添加

为子线程添加响应时间

启动

不停顿启动

验证

剪切

复制

粘贴

Ctrl-X

Ctrl-C

Ctrl-V

取样器

逻辑控制器

前置处理器

后置处理器

断言

定时器

测试片段

测试活动

HTTP请求

Debug Sampler

JSR223 Sampler

AJP/1.3 取样器

Access Log Sampler

BeanShell 取样器

Bolt Request

3) 填写“HTTP请求”的相关请求数据

Apache JMeter (5.5)

文件 编辑 查找 运行 选项 工具 帮助



Test Plan

线程组

HTTP请求

HTTP请求 命名

名称: 博客项目登陆接口

注释: 第一个接口的性能测试配置

添加接口: http://192.168.47.135:8653/blog_system/login

Web服务器

协议: http

服务器名称或IP: 192.168.47.135

端口号: 8653

HTTP请求

GET

路径: /blog_system/login

内容编码:

☐ 自动重定向

☒ 跟随重定向

☒ 使用 KeepAlive

☐ 对POST使用multipart / form-data

☐ 与浏览器兼容的头

参数 消息体数据 文件上传

请求参数

同请求一起发送参数:

名称:

值

编码?

内容类型

包含等于?

username

admin

☐ text/plain

☒

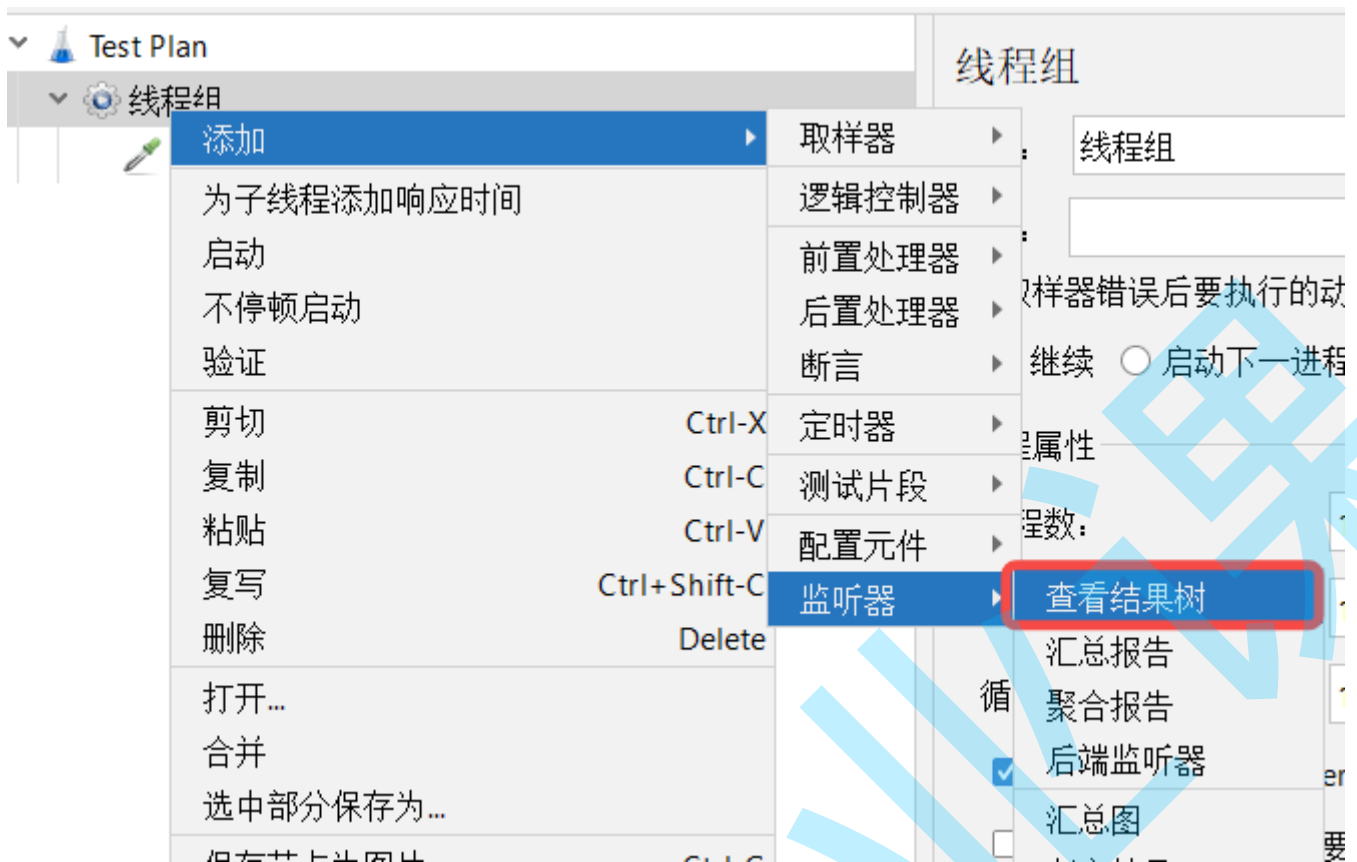
password

123

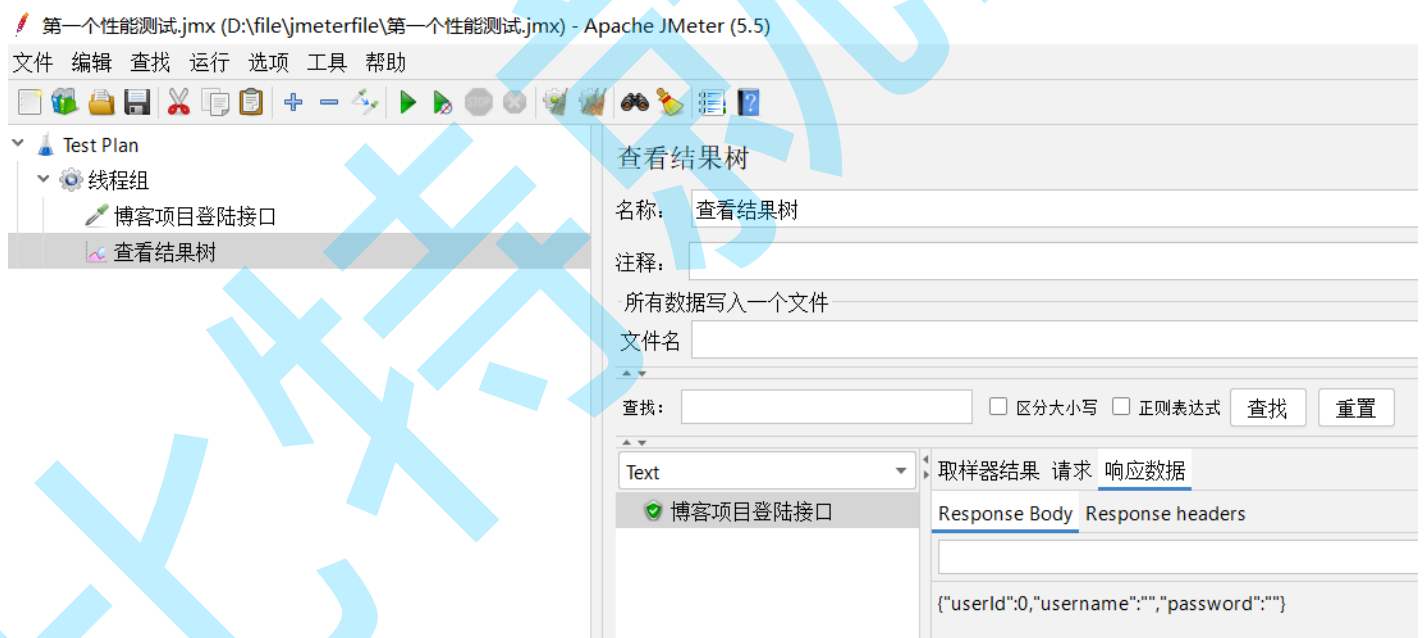
☐ text/plain

☒

4) 在“线程组”下添加“查看结果树”监听器



5) 点击“启动”按钮运行，查看接口测试结果



1.5 JMeter元件作用域和执行顺序

在JMeter中，元件的作用域和执行顺序是非常重要的概念。

作用域：

JMeter元件的作用域主要由测试计划的树形结构中的元件父子关系来确定。

执行顺序：

取样器(sampler)元件内组件不依赖其他元件就可执行，因此取样器不存在作用问题
元件作用域只对它的子节点有作用，
其他作用域默认根据测试计划中树形结构来定；

2. 重点组件

2.1 线程组

控制JMeter将用于执行测试的线程数，也可以把一个线程理解为一个测试用户。

线程组

名称：

线程组

注释：

在取样器错误后要执行的动作

☒

继续

☐

启动下一进程循环

☐

停止线程

☐

停止测试

☐

立即停止测试

线程属性

线程数：

1

Ramp-Up时间（秒）：

1

循环次数

☐ 永远

1

☒ Same user on each iteration

☐ 延迟创建线程直到需要

☐ 调度器

持续时间（秒）

启动延迟（秒）

线程数：一个线程即一个测试用户，设置发送的请求次数

Ramp-up时间（秒）:设置性能测试运行时间，单位为秒

循环次数：

- 配置指定次数：控制脚本循环执行的次数
- 配置循环永远
 - 需要调度器配置使用
 - 运行时间：脚本执行时间
 - 延迟启动时间：脚本等待指定时间才能运行

2.2 HTTP取样器

HTTP请求

名称: 博客项目登陆接口

注释: 第一个接口的性能测试配置

基本 高级

Web服务器

协议: http 服务器名称或IP: 192.168.47.135 端口号: 8653

HTTP请求

GET 路径: /blog_system/login 内容编码:

☐ 自动重定向 ☒ 跟随重定向 ☒ 使用 KeepAlive ☐ 对POST使用multipart / form-data ☐ 与浏览器兼容的头

参数 消息体数据 文件上传

同请求一起发送参数:

名称:	值	编码?	内容类型	包含等于?
username	admin	<input type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>
password	123	<input type="checkbox"/>	text/plain	<input checked="" type="checkbox"/>

添加必需的配置：

- http协议
- http主机名/IP
- 端口
 - http协议端口号80
 - https端口号443
- 请求方法
- 路径（目录+参数）
- 内容编码（默认的ISO国际标准，但对中文支持不友好，可以使用utf-8）
- 参数
 - 参数可以拼在路径里，也可以卸载参数中
 - POST参数要放到消息体数据中{wd:test}

2.3 查看结果树

查看结果树

名称: 查看结果树

注释:

所有数据写入一个文件

文件名: 浏览... 显示日志内容: ☐ 仅错误日志 ☐ 仅成功日志 配置

查找: ☐ 区分大小写 ☐ 正则表达式 查找 重置

Text 取样器结果 请求 响应数据

博客项目登陆接口

博客项目登陆接口

Response Body Response headers

Find ☐ 区分大小写 ☐ 正则表达式

取样器结果：统计请求相关的信息

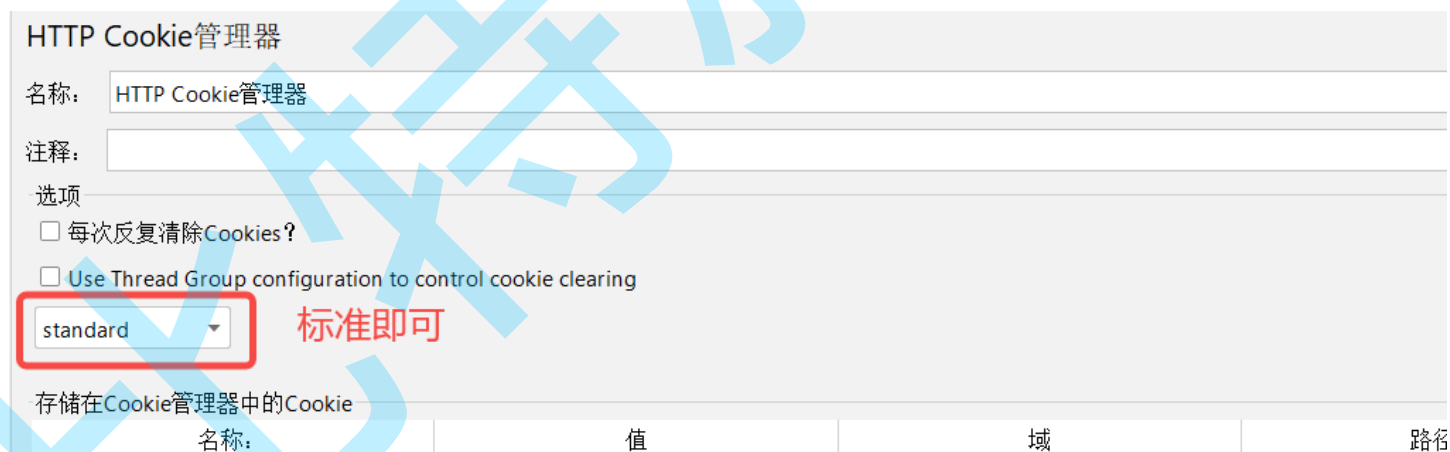
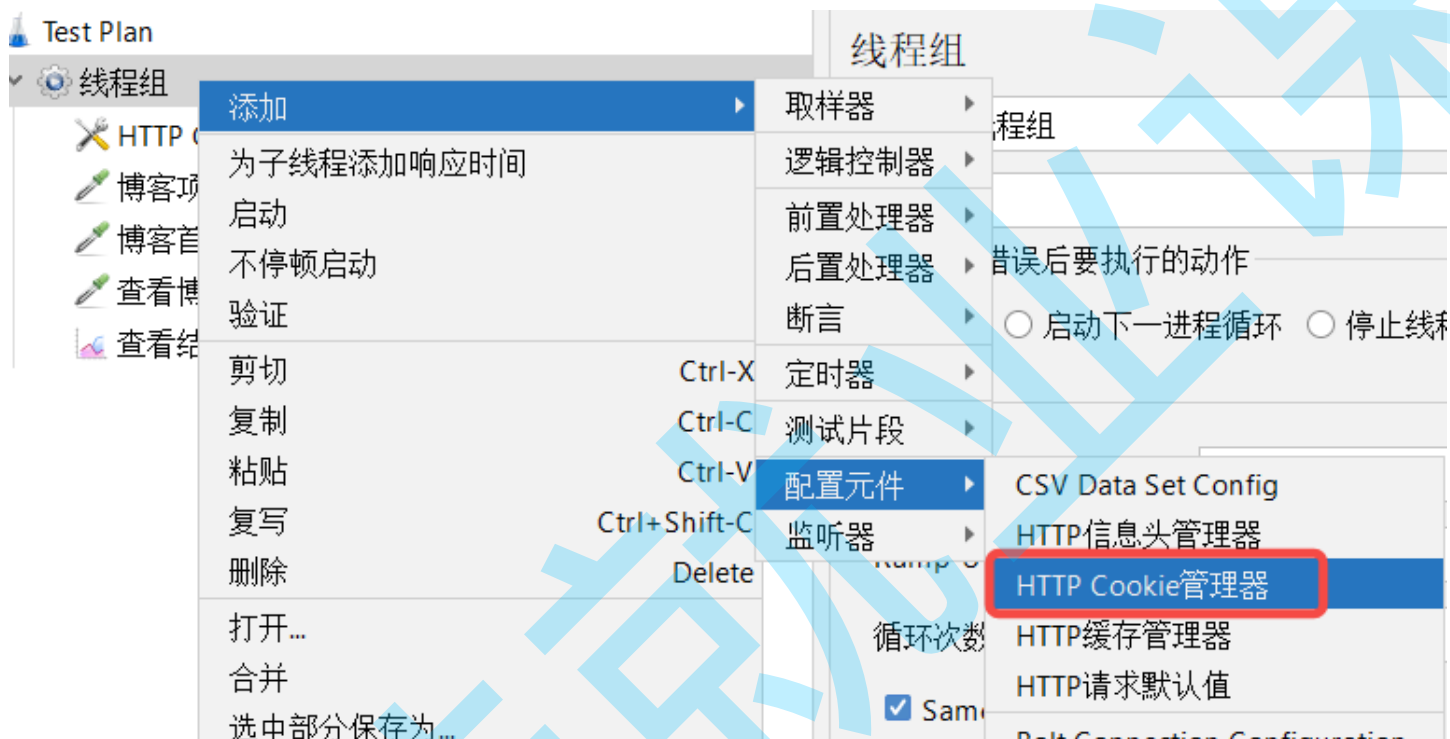
- 1 Thread Name：线程组名称

- 2 Sample time:发送请求时间
- 3 load time: 响应时间
- 4 Response code : 接口响应状态码

请求：HTTP请求的请求头和请求体的详细信息

响应：HTTP响应的响应头和响应体的详细信息

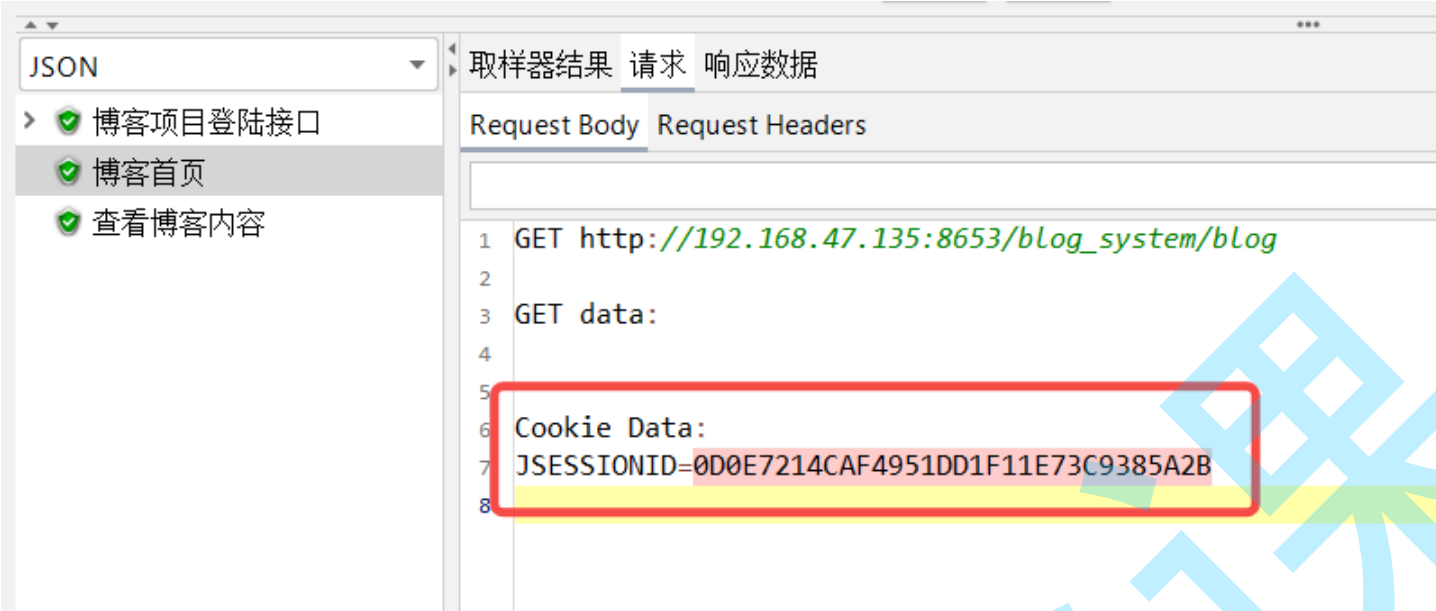
2.4 HTTP Cookie管理器



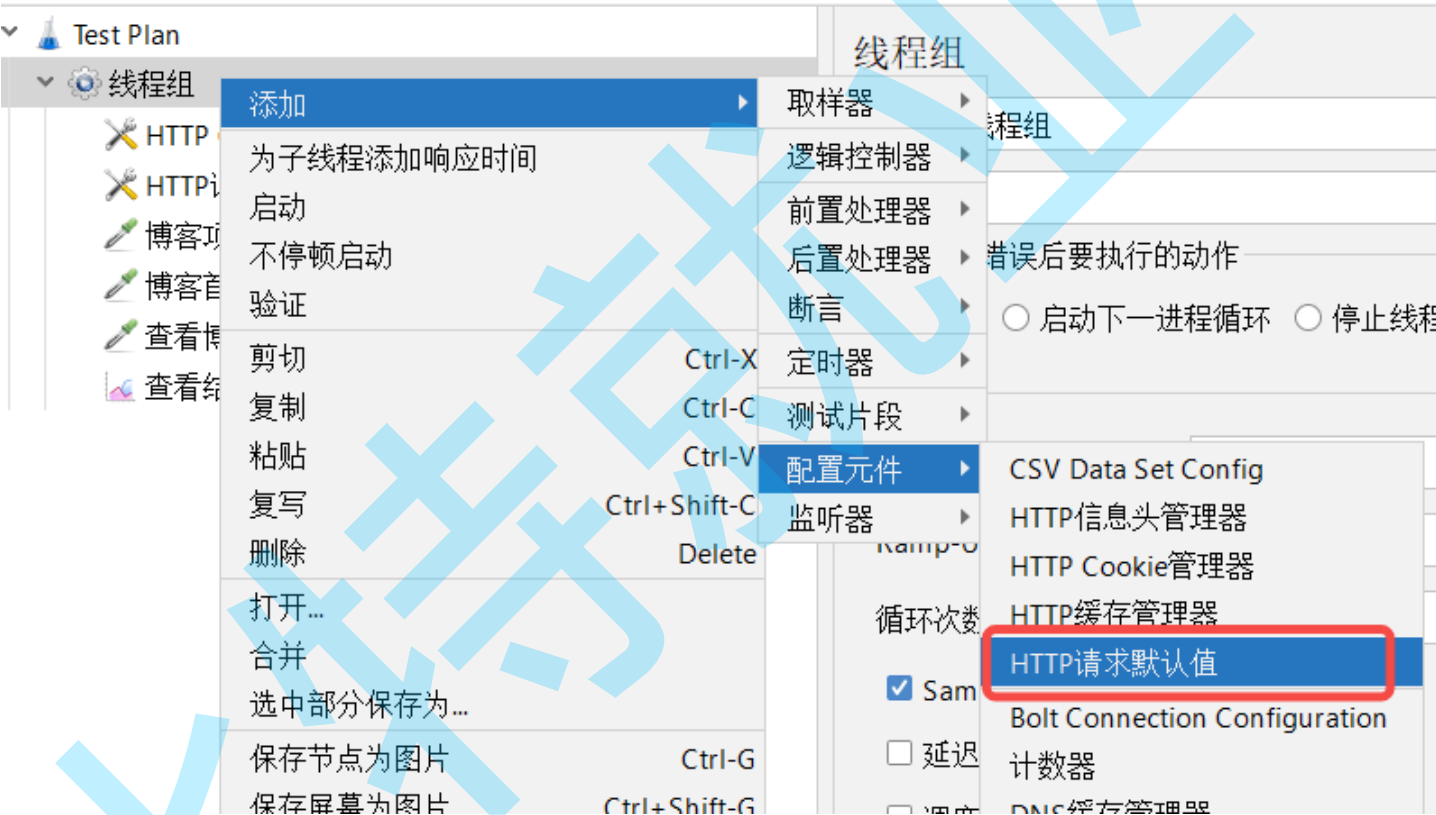
HTTP Cookie管理器像Web浏览器一样存储和发送Cookie。如果HTTP请求并且响应包含cookie,则Cookie管理器会自动存储该cookie,并将其用于将来对该特定网站的所有请求。每个JMeter线程都有自己的"cookie存储区"。因此,正在测试使用cookie存储会话信息的网站,则每个JMeter线程都将拥有自己的会话。此类Cookie不会显示在Cookie管理器显示屏上,可以使用"查看结果树监听器"查看。

缓存配置可选择standard(标准)或compatibility(兼容的),当然也可以手工添加一些cookie.

添加了HTTP Cookie管理器后,会自动存储并发送Cookie



2.5 HTTP请求默认值



博客中涉及到的接口协议、IP、端口号全都一样，可以单独抽取出来存放在默认值中，其他接口就可以省略不写协议、IP、端口号

HTTP请求

名称: 博客项目登陆接口

注释: 第一个接口的性能测试配置

基本 高级

Web服务器

协议: 服务器名称或IP: 端口号:

HTTP请求

POST 路径: /blog_system/login 内容编码:

☐ 自动重定向 ☒ 跟随重定向 ☒ 使用 KeepAlive ☐ 对POST使用multipart / form-data ☐ 与浏览器兼容的头

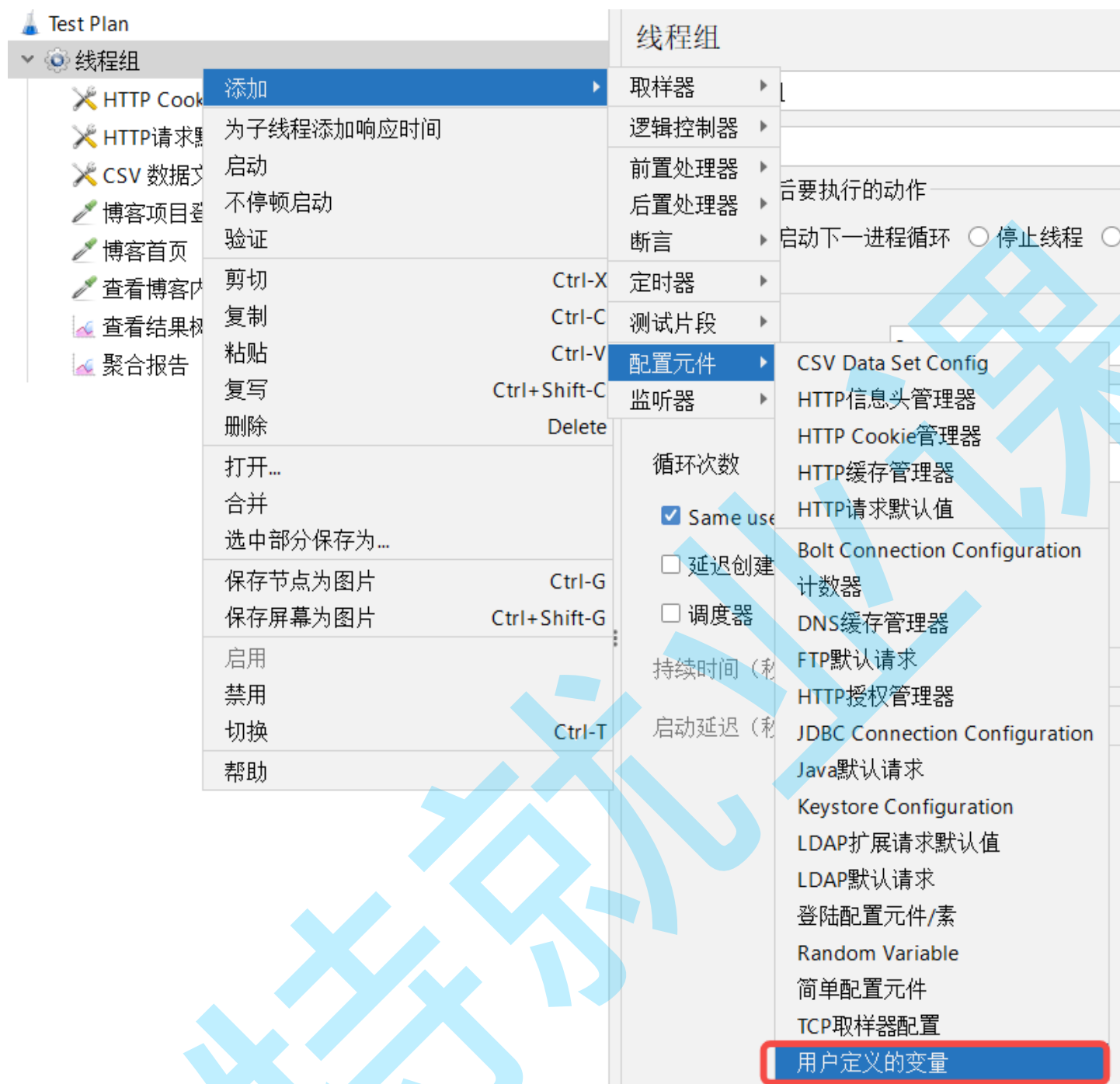
参数 消息体数据 文件上传

同请求一起发送参数:

名称 值 编码? 内容类型 内容长度?

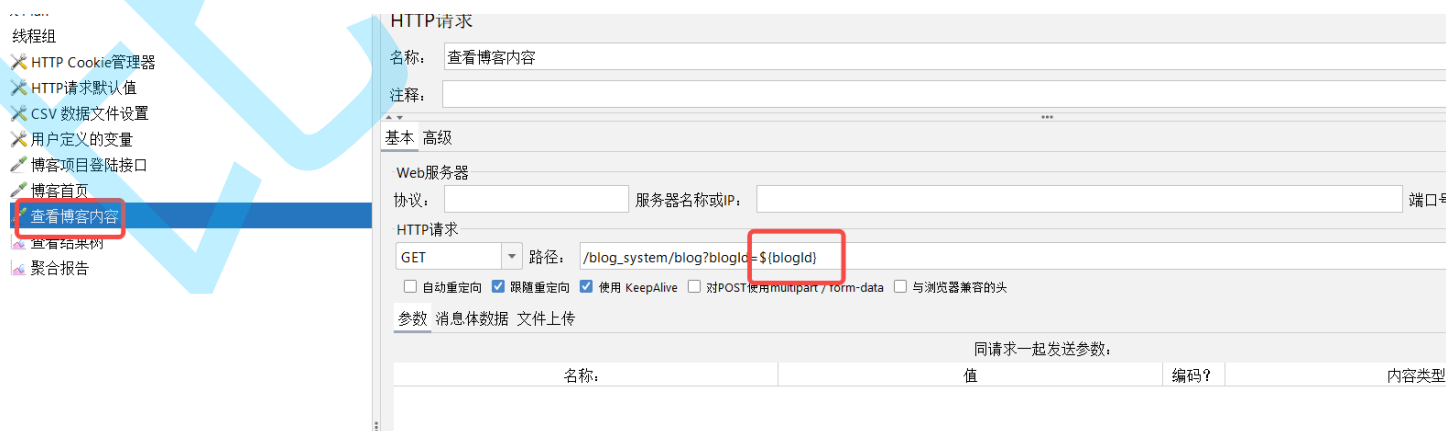
2.6 用户定义的变量

添加方式：线程组—配置元件—用户定义的变量



有时我们只想要在固定的场景里使用参数化，改动后不希望影响到其他的脚本。

使用：在HTTP请求的取样器中引入定义的变量。\${参数名}

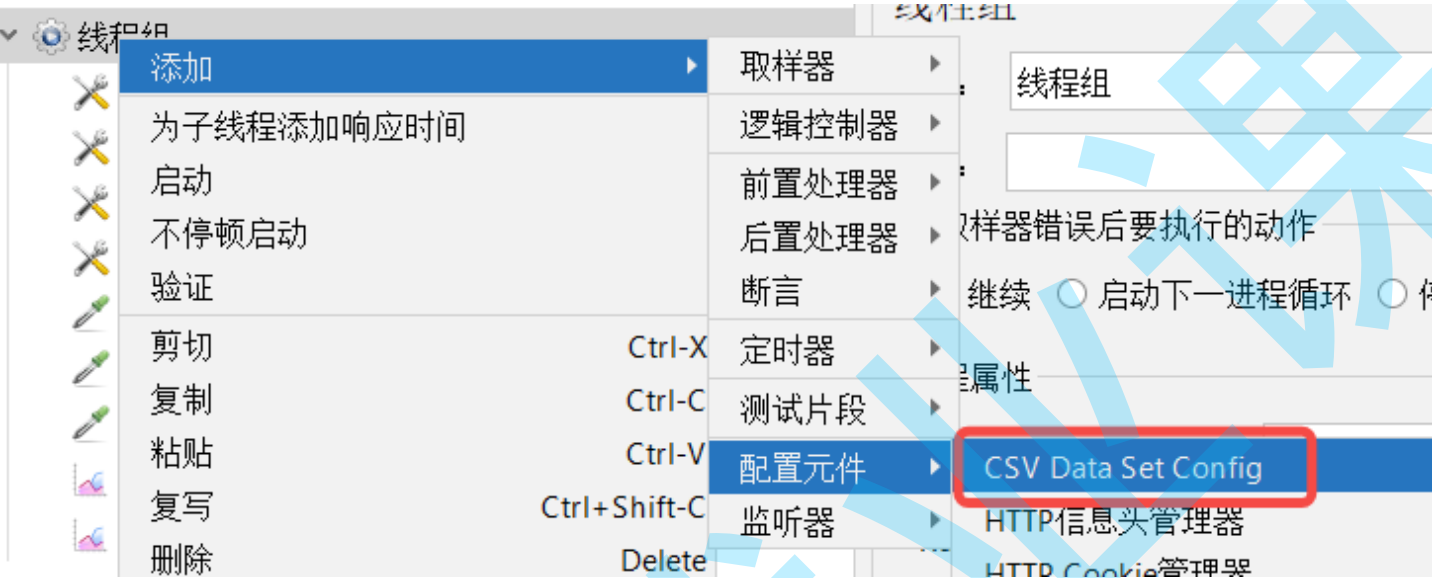


适用场景：变量需要在多个脚本中使用，方面统一管理和修改

2.7 CSV数据文件设置

以登陆接口为例，当我们执行登陆接口的性能测试时，手动配置了用户名和密码为固定的username和password，然而实际使用中不可能只有一个用户登陆，为了模拟更真实的登录环境，我们需要提供更多的用户username和password来实现登录操作

添加方式：线程组——配置元件——CSV数据文件设置



2.7.1 操作步骤

1) CSV数据文件设置

CSV 数据文件设置

名称: CSV 数据文件设置

注释:

设置 CSV 数据文件

文件名: D:/file/jmeterfile/test.csv 浏览...

文件编码: UTF-8

变量名称(西文逗号间隔): username,password

忽略首行(只在设置了变量名称后才生效): False

分隔符(用 \t 代替制表符): ,

是否允许带引号?: False

遇到文件结束符再次循环?: True

遇到文件结束符停止线程?: False

线程共享模式: 所有线程

- 文件名：填写csv文件的路径。建议使用绝对路径。
- 文件编码：UTF-8
- 变量名称：从csv数据文件中读起的数据需要保存到的变量名。有多个变量时用逗号分隔
- 是否忽略首行：是否从csv数据文件第一行开始读取。
- 分隔符：要求与csv数据文件中多列的分隔符一致
- 遇到文件结束符再次循环：若选择为True当数据不够的时候会从头取。若选择False，则需要勾选下面的配置，遇到文件结束符停止线程，这里如果不勾选，请求将会报错。

2) 编写test.csv文件，示例：

test.csv

	A	B	C	D	E	F	G	H
1	admin	123						
2	zhangsan	123						
3	lisi	123						
4								

3) 修改登陆接口及其他涉及到username和password获取的参数

博客项目登陆接口

Web服务器

协议: 服务器名称或IP:

HTTP请求

POST 路径: /blog_system/login

☐ 自动重定向 ☒ 跟随重定向 ☒ 使用 KeepAlive ☐ 对POST使用multipart / form-data ☐ 与浏览器兼容的头

参数 消息体数据 文件上传

名称	值	同请求一起发送参数:	编码?	
username	\$(username)		<input type="checkbox"/>	text/plain
password	\$(password)		<input type="checkbox"/>	text/plain

修改完该配置后，登陆接口发起请求时将从csv文件中获取配置好的username和password参数，获取顺序为从上往下依次获取

4) 修改线程组中线程数，使得每次取到的username和password都不一样

线程组

名称: 线程组

注释:

在取样器错误后要执行的动作

☒ 继续 ☐ 启动下一进程循环 ☐ 停止线程 ☐ 停止测试 ☐ 立即停止测试

线程属性

线程数: 3

Ramp-Up时间(秒): 1

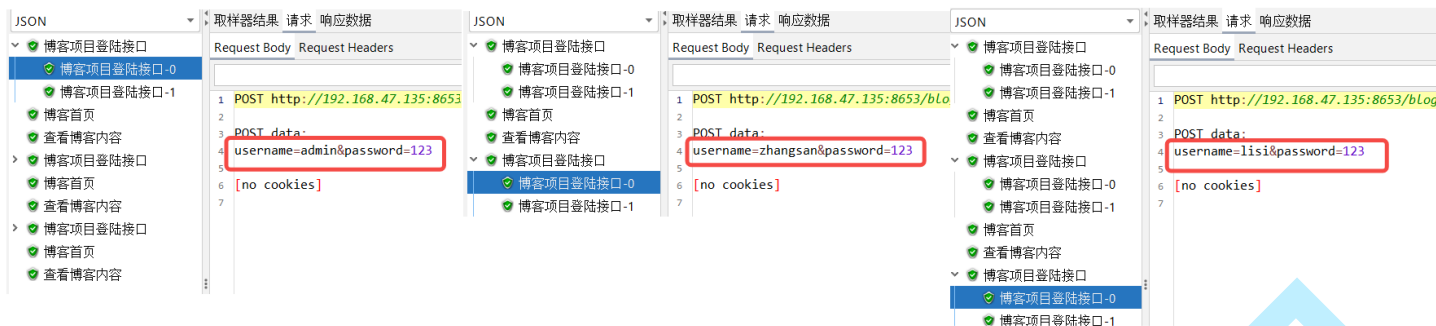
循环次数 ☐ 永远 1

☒ Same user on each iteration

☐ 延迟创建线程直到需要

☐ 调度器

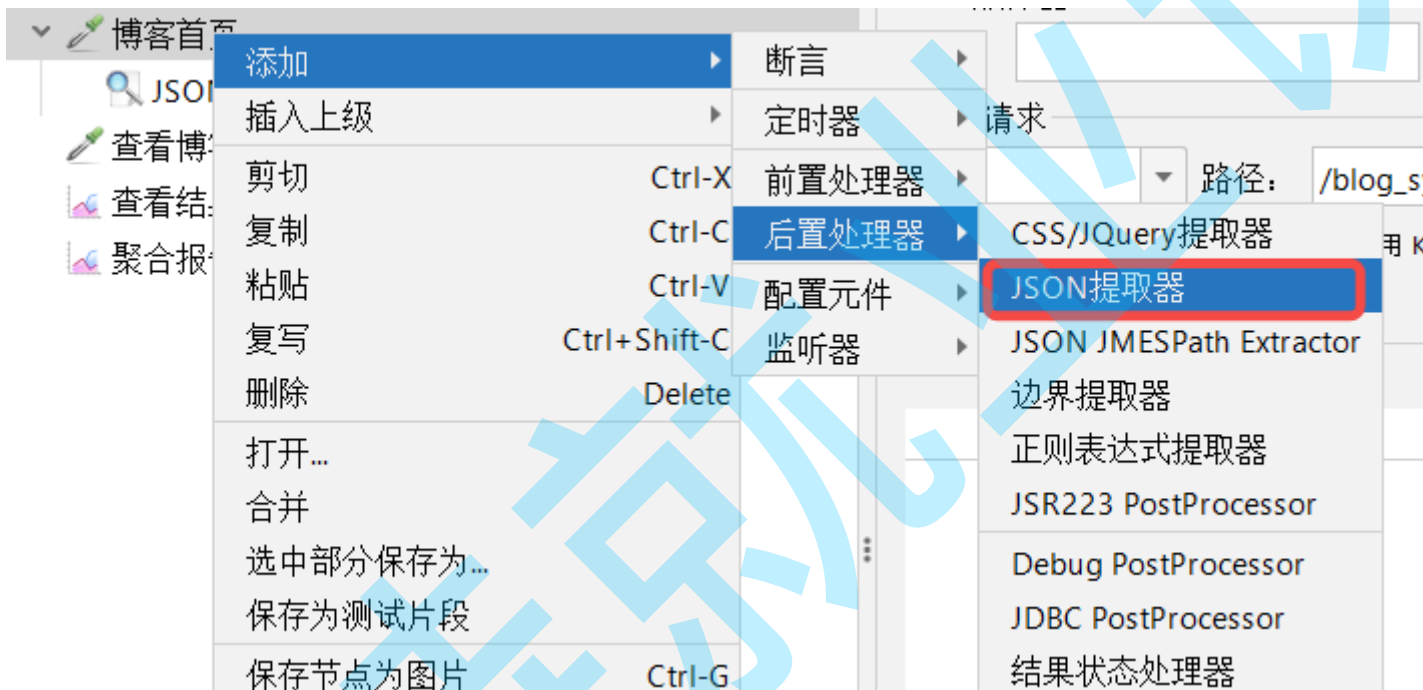
5) 运行结果



2.8 JSON提取器

接口响应成功，通过提取返回值对应字段，可用于其他接口的参数配置

1) 添加JSON提取器



针对某一个HTTP请求接口添加JSON提取器

案例：以博客首页为例

```
1  [
2    {
3      "postTime": "2024-04-18 05:20:16",
4      "title": "dddddd",
5      "blogId": 13,
6      "userId": 3,
7      "content": "# 在这里写下一篇博客\r\nndddd"
8    },
9  ],
10 {
11   "postTime": "2022-10-22 02:38:21",
12   "title": "同学，请问你今天学习了么",
13   "blogId": 12,
14   "userId": 3,
```

```

14         "content": "今天是2022年10月22日17:42分，为了能够早日将最新版本的测试课件呈现
    给同学们，我已经开始奋..."
15     }
16 ]

```

JSON操作符参考：

Operator	Description
\$	表示根元素
@	当前元素
*	通配符。所有节点
..	选择所有符合条件的节点
.<name>	子元素
['<name>' (, '<name>')]	括号表示子元素或子元素列表
[<number> (, <number>)]	数组索引或索引列表
[start:end]	数组切片操作符
[?(<expression>)]	过滤器表达式。表达式必须评估为布尔值。

参考文档：<https://github.com/json-path/JsonPath>

获取相应中的所有blogId元素：\$..blogId

获取第一个blogId元素：\$.[0]blogId

2) 添加JSON配置

JSON提取器

名称：JSON提取器

注释：

Apply to:

☐ Main sample and sub-samples
 ☒ Main sample only
 ☐ Sub-samples only
 ☐ JMeter Variable Name to use

Names of created variables:

JSON Path expressions:

Match No. (0 for Random):

Compute concatenation var (suffix_ALL): ☐

Default Values:

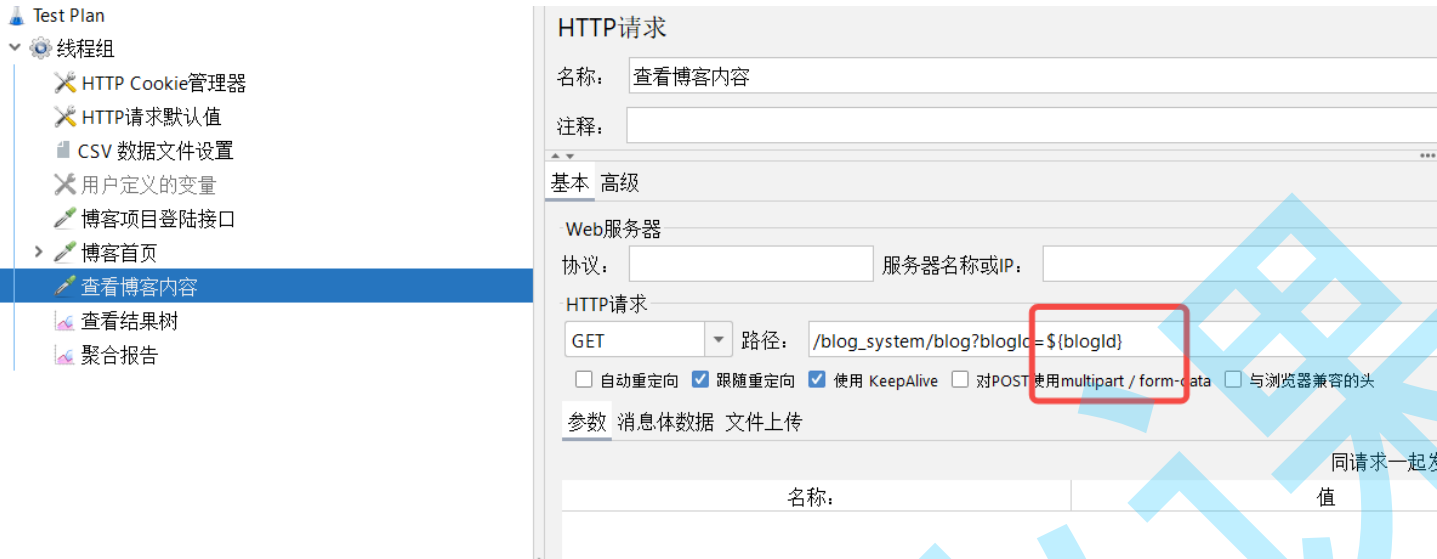
blogId

给提取到的数据存储在該变量下

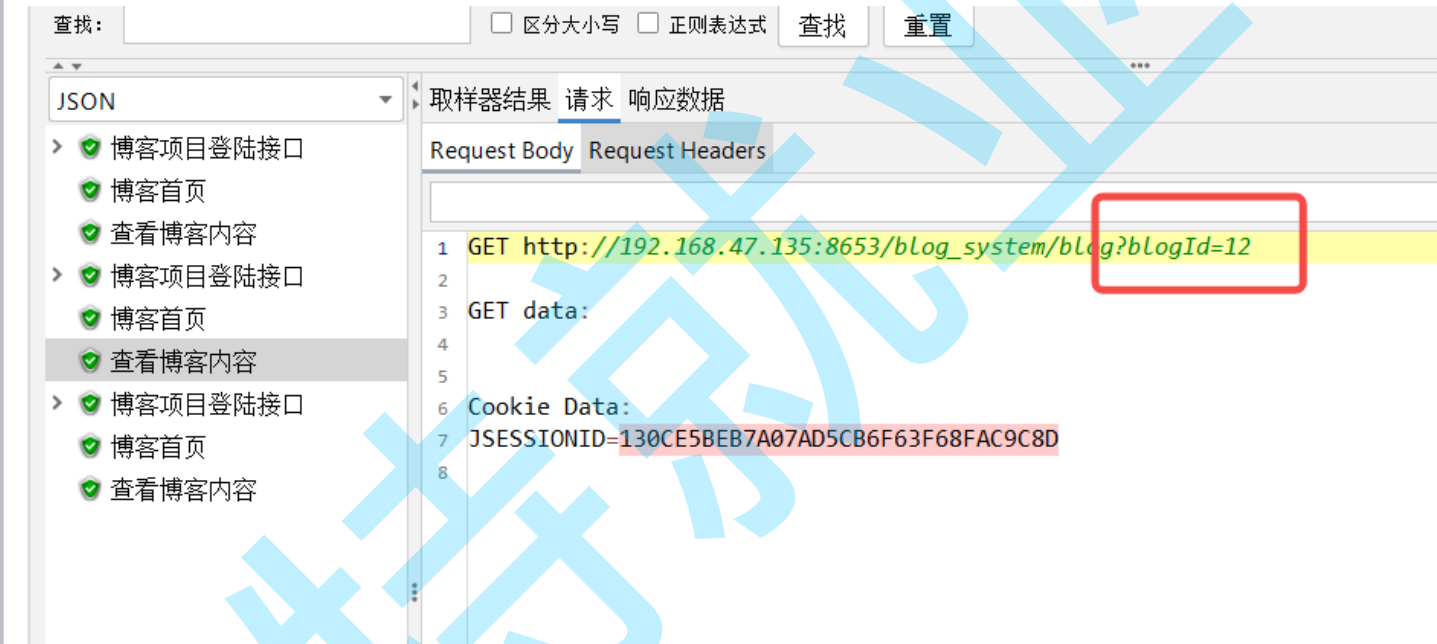
\$.[1]blogId

json提取

3) 配置json提取的参数



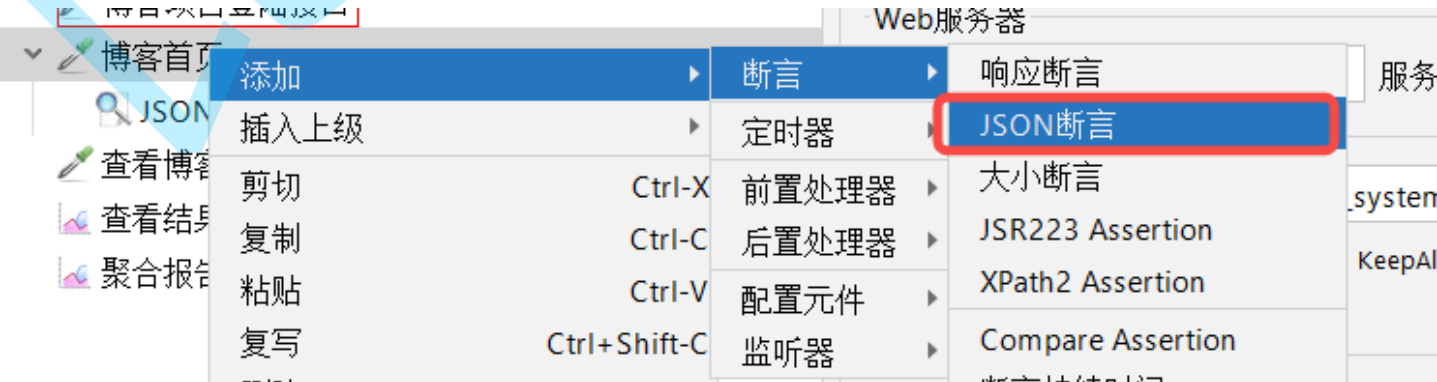
运行脚本后，所有的查看博客内容接口对应的blogId统一替换为12：



2.9 JSON断言

接口发送请求成功，响应码为200并不能完全代表接口请求成功，我们更多需要关注接口响应数据是否符合预期。

1) 添加JSON断言：



针对某一个HTTP请求接口添加JSON断言

2) 添加JSON配置

同JSON提取器语法配置

JSON断言

名称:

注释:

Assert JSON Path exists:

☒ Additionally assert value

☐ Match as regular expression

Expected Value:

1

同学，请问你今天学习了.*

Expect null ☐

Invert assertion (will fail if above conditions met) ☐

匹配断言值
若选上，表示支持断言匹配

添加断言值

期望结果为空

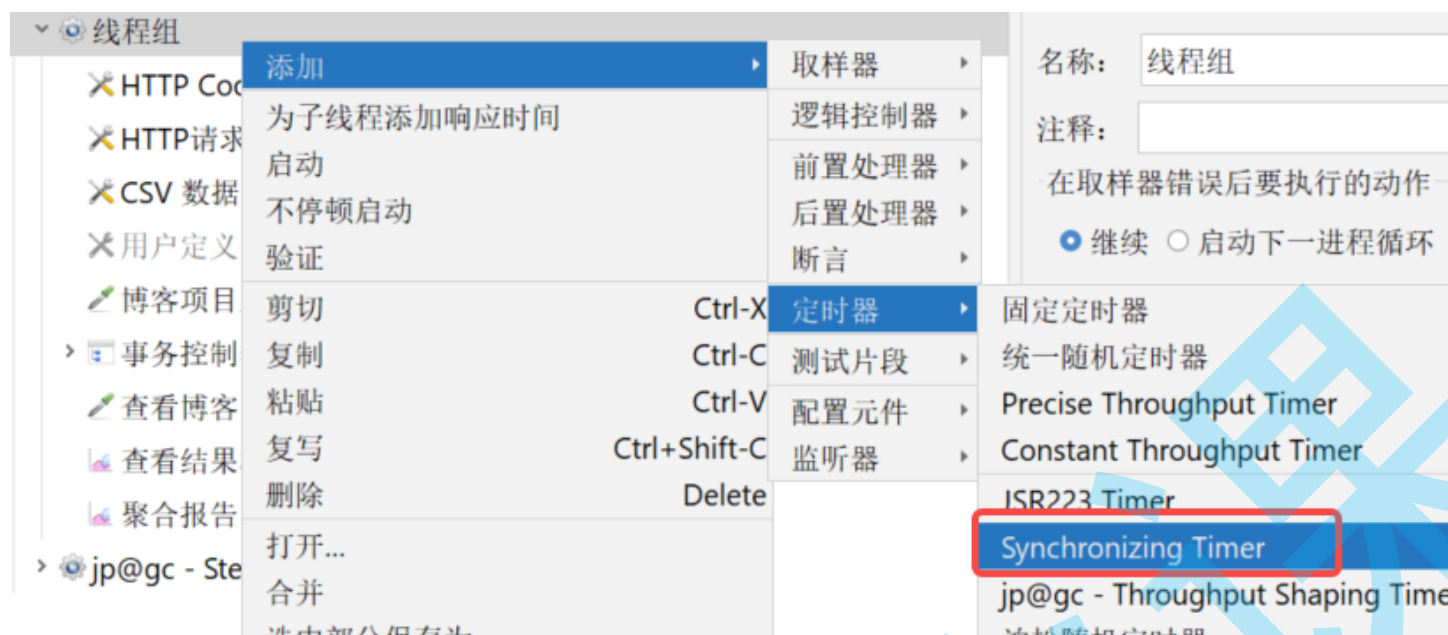
结果取反

注意：

- 1) 若不选Additionally assert value，表示添加断言值，则可用来判断字段是否存在
- 2) 选择Additionally assert value，则必须添加Expected Value期望的断言值
- 3) 若不选Match as regular expression正则匹配，则Expected Value必须填写完整，少一个字符都会导致断言失败
- 4) 若选择Match as regular expression正则匹配，则Expected Value可以仅写上部分关键词即可断言成功

2.10 同步定时器（集合点）

为了达到并发的效果，需要添加同步定时器



JMeter同步定时器的作用主要在于模拟多用户并发访问的场景，确保多个线程能够同时执行某个操作，以达到真正的并发效果。

当多个线程同时启动时，它们可能会在不同的时间间隔内执行，这样就无法达到真正的并发效果。同步定时器的作用就是将这些线程的执行时间同步，使它们在同一时间点执行。它可以在多个线程之间制造一定的延迟，直到同时到达指定时间点，再同时执行后续的操作。

此外，同步定时器可以理解为集合点，当线程数量达到指定值后，再一起释放，可以瞬间产生很大的压力。这样，可以更好地模拟真实的用户并发访问场景，提高测试的准确性和可靠性。

在性能测试过程中，为了真实模拟多个用户同时进行操作以度量服务器的处理能力，可以使用同步定时器来设置集合点。不过，虽然通过加入集合点可以约束请求同时发送，但不能确保请求同时到达服务器，所以只能说是较真实模拟并发。

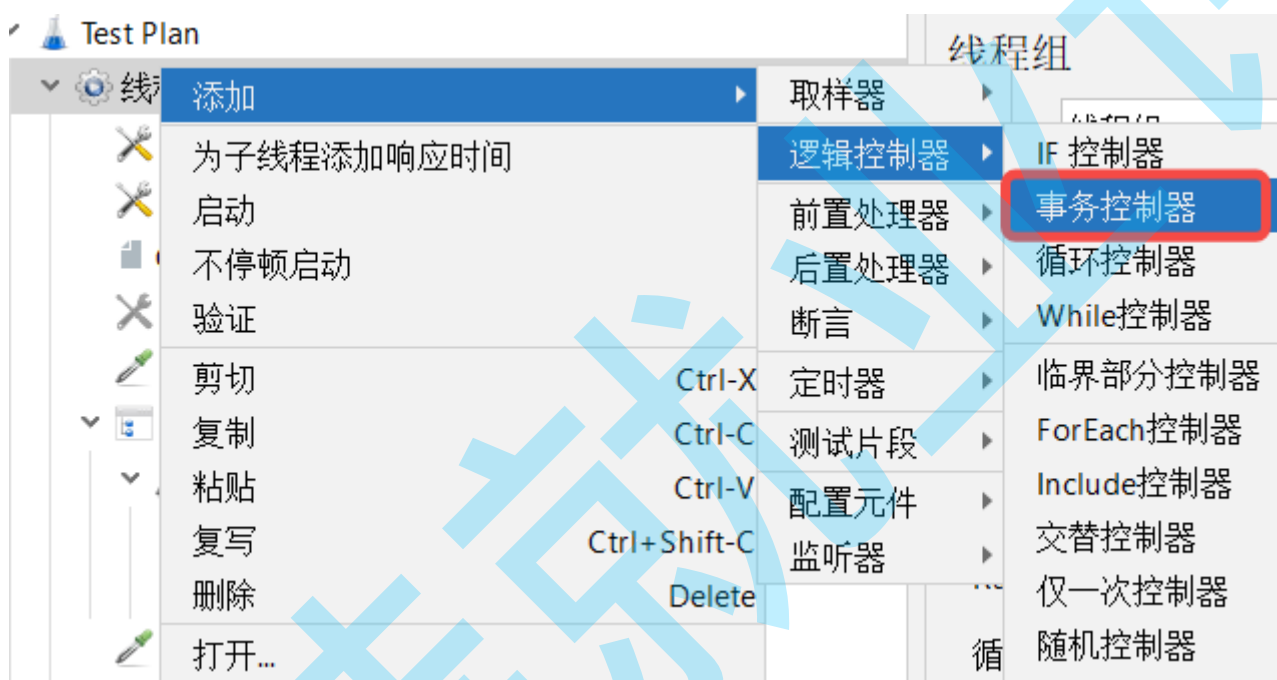


现实生活中，红绿灯就相当于一个集合点，有人先到达，有人后达到，但必须等到绿灯后所有人才能开始过人行道。

2.11 事务控制器

JMeter事务控制器的作用主要用于测试执行嵌套测试元素所花费的总时间。这相当于模拟用户进行一系列操作的测试。

在进行页面性能测试或API性能测试时，事务控制器是一个非常重要的工具。它可以帮助测试人员更准确地评估系统性能，尤其是在涉及多个接口或操作的复杂场景中。例如，在订单提交的过程中，可能需要调用多个接口，并且某些接口可能依赖于前一个接口的结果。在这种情况下，使用事务控制器可以将这些接口统一视为一个事务进行性能测试，从而得到更接近真实场景的性能测试结果。



若不添加事务控制器，则一个接口即一个事务。

添加了事务控制器后，可以将多个接口统一放到一个事务控制器下作为一个事务。

2.12 Jmeter插件

下载Jmeter插件功能：

<https://jmeter-plugins.org/install/Install/>



- [Install](#)
- [Browse Plugins](#)
- [Documentation](#)
- [Usage Statistics](#)
- [Support Forums](#)
- [JMX Editor](#)



Installing Plugins

The easiest way to get the plugins is to install [Plugins Manager](#). Then you'll be able to install any other plugins just by clicking

Download [plugins-manager.jar](#) and put it into `lib/ext` directory, then restart JMeter.

If you experience any issues with plugins installation, don't hesitate to ask at [Support Forums](#).

点击下载插件

将下载好的插件放到jmeter下lib/ext文件夹下：

脑 > Data (D:) > software > apache-jmeter-5.5 > apache-jmeter-5.5 > lib > ext				
名称	修改日期	类型	大小	
jmeter-plugins-cmd-2.2.jar	2024/4/19 10:39	Executable Jar File	14 KB	
jmeter-plugins-fifo-0.2.jar	2024/4/19 10:39	Executable Jar File	18 KB	
jmeter-plugins-graphs-basic-2.0.jar	2024/4/19 10:39	Executable Jar File	9 KB	
jmeter-plugins-ffw-2.0.jar	2024/4/19 10:39	Executable Jar File	15 KB	
jmeter-plugins-functions-2.2.jar	2024/4/19 10:39	Executable Jar File	30 KB	
jmeter-plugins-pde-0.1.jar	2024/4/19 10:39	Executable Jar File	9 KB	
jmeter-plugins-dummy-0.4.jar	2024/4/19 10:39	Executable Jar File	14 KB	
jmeter-plugins-casutg-2.10.jar	2024/4/19 10:39	Executable Jar File	68 KB	
jmeter-plugins-tst-2.6.jar	2024/4/19 10:39	Executable Jar File	17 KB	
jmeter-plugins-manager-1.10.jar	2024/4/19 10:26	Executable Jar File	887 KB	
ApacheMeter_bolt.jar		Executable Jar File	20 KB	

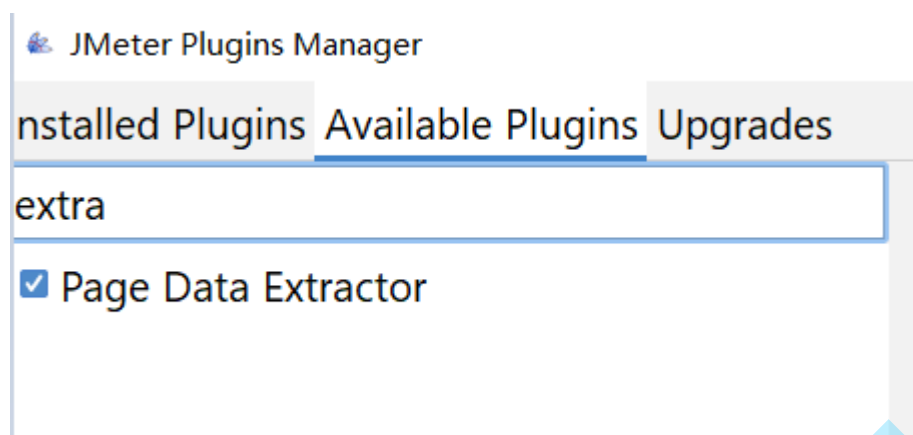
此时，jmeter界面右上角将会展示一个小蝴蝶形状的工具，该工具即jmeter插件功能，点击该功能可以下载jmeter中支持的各种插件：



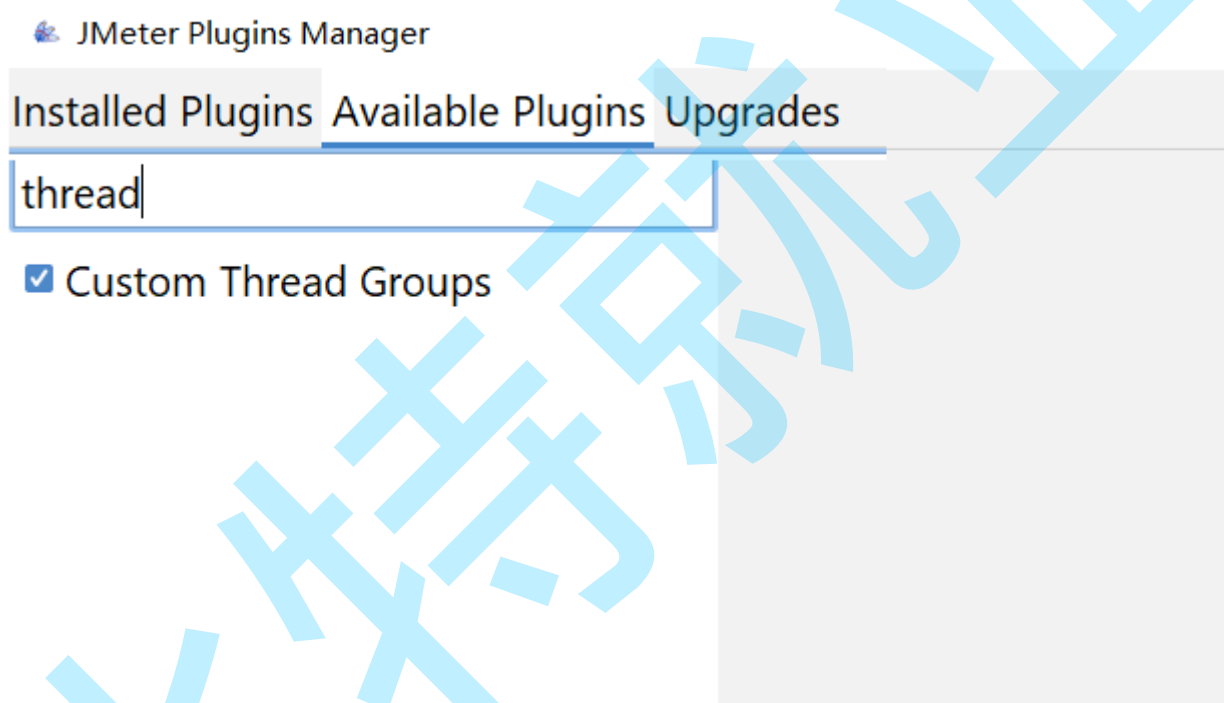
在真实企业压测场景中，我们通常为一点一点的逐步增加线程数，因此需要安装新的插件来支持线程数的配置。

通过插件管理工具下载其他插件：

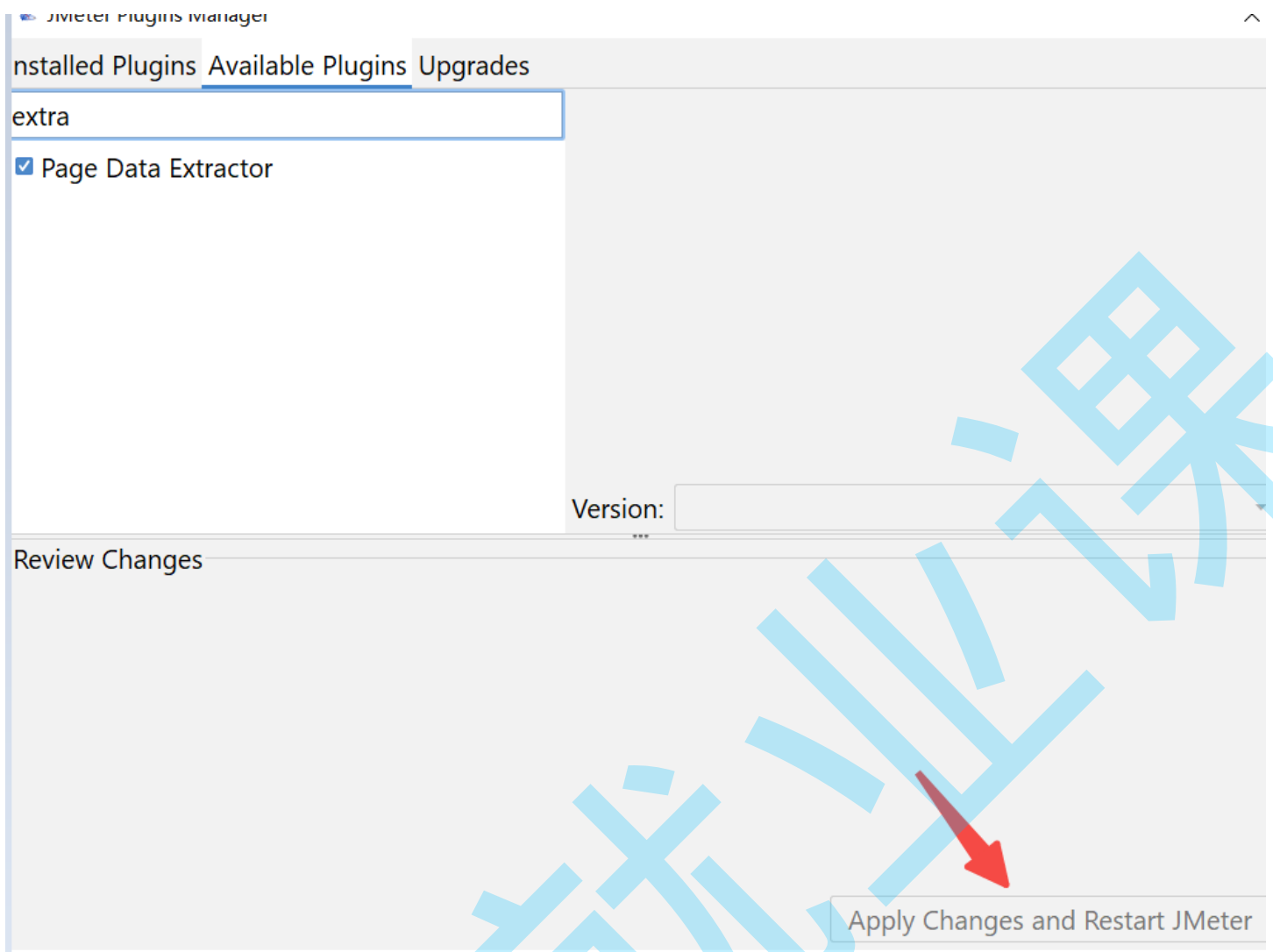
1) 在插件中下载其他监听器插件



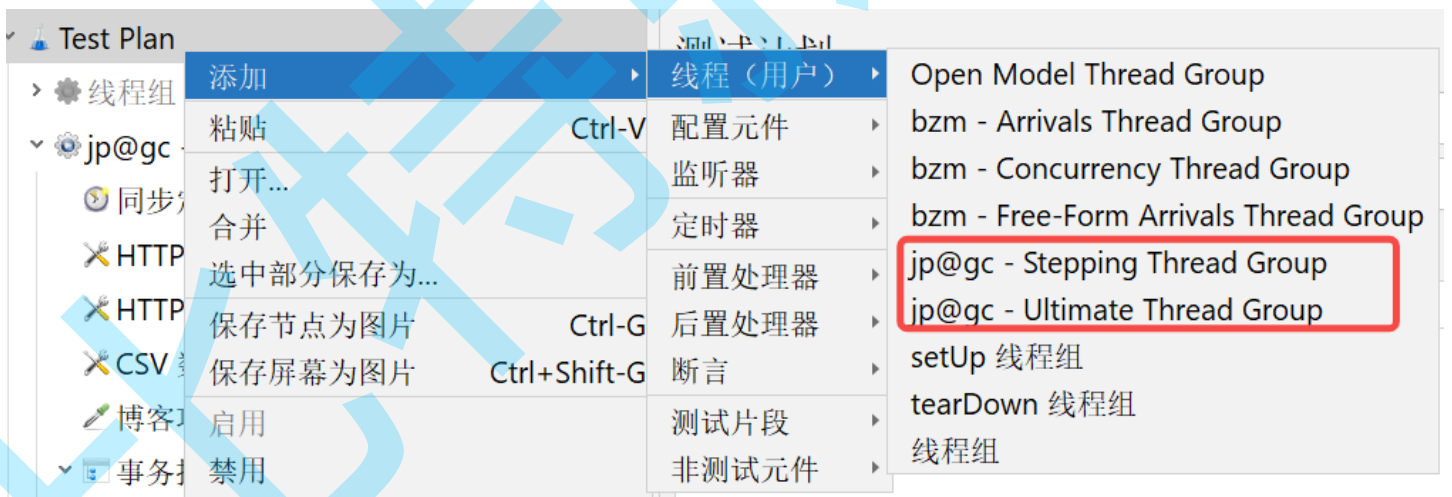
2) 在插件中下载线程组插件



点击Apply Changes and Restart JMeter等待下载完成并重启JMeter:



下载完成后再线程和监听器中可以看到新增的元素：



2.12.1 Stepping Thread Group

梯度压测线程组

jp@gc - Stepping Thread Group

名称: jp@gc - Stepping Thread Group

注释:

在取样器错误后要执行的动作

☒ 继续 ☐ 启动下一进程循环 ☐ 停止线程 ☐ 停止测试 ☐ 立即停止测试

Threads Scheduling Parameters

This group will start threads:
First, wait for seconds;
Then start threads;
Next, add threads every
Then hold load for seconds.
Finally, stop threads every



This group will start: 启动多少个线程，同线程组中的线程数

First, wait for: 等待多少秒才开始压测，一般默认为0

Then start: 一开始有多少个线程数，一般默认为0

Next,add: 下一次增加多少个线程数

threads every: 当前运行多长时间后再次启动线程,即每一次线程启动完成之后的持续时间;

using ramp-up:启动线程的时间;若设置为5秒,表示每次启动线程都持续5秒

thenhold loadfor: 线程全部启动完之后持续运行多长时间

finally,stop/threadsevery: 多长时间释放多少个线程; 若设置为5个和1秒, 表示持续负载结束之后每1秒钟释放5个线程

2.13 常见监听器

2.13.1 聚合报告

从聚合报告可以看到性能测试过程中整体的数据变化

聚合报告

名称: 聚合报告

注释:

所有数据写入一个文件

文件名

浏览...

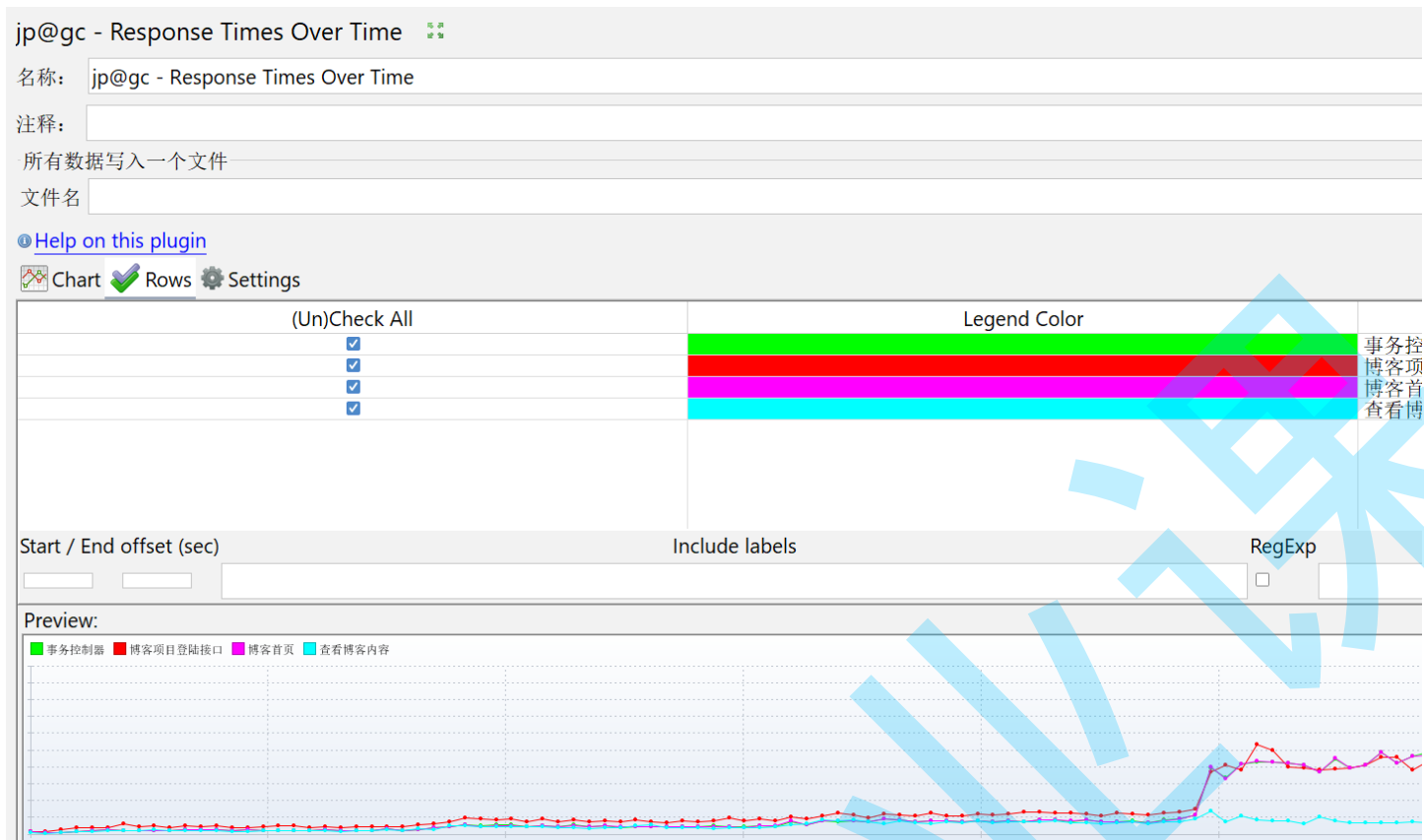
显示日志内容: ☐ 仅错误日志 ☐ 仅成功日志

Label	# 样本	平均值 ↑	中位数	90% 百分位	95% 百分位	99% 百分位	最小值	最大值	异常 %	吞吐量	接收 KB/sec	发送 KB/sec
HTTP请求	20	6	7	8	8	8	6	8	0.00%	8.0/sec	11.29	1.05
总体	20	6	7	8	8	8	6	8	0.00%	8.0/sec	11.29	1.05

指 标	说 明
Samples	发起的 HTTP 请求调用数
Average	平均响应时间，单位为毫秒
Median	请求调用响应时间的中间值，也就是 50%请求调用的响应时间，单位为毫秒
90%Line	90%请求调用的响应时间，单位为毫秒
95%Line	95%请求调用的响应时间，单位为毫秒
99%Line	99%请求调用的响应时间，单位为毫秒
Min	请求调用的最小响应时间，单位为毫秒
Max	请求调用的最大响应时间，单位为毫秒
Error%	调用失败的请求占比。调用失败一般指响应断言失败或者请求调用出错
Throughput	TPS/QPS，每秒处理的事务数
KB/sec	每秒网络传输的流量大小，单位为 KB。这个指标是以网络传输的大小来衡量网络的吞吐量

2.13.2 Response Times Over Time

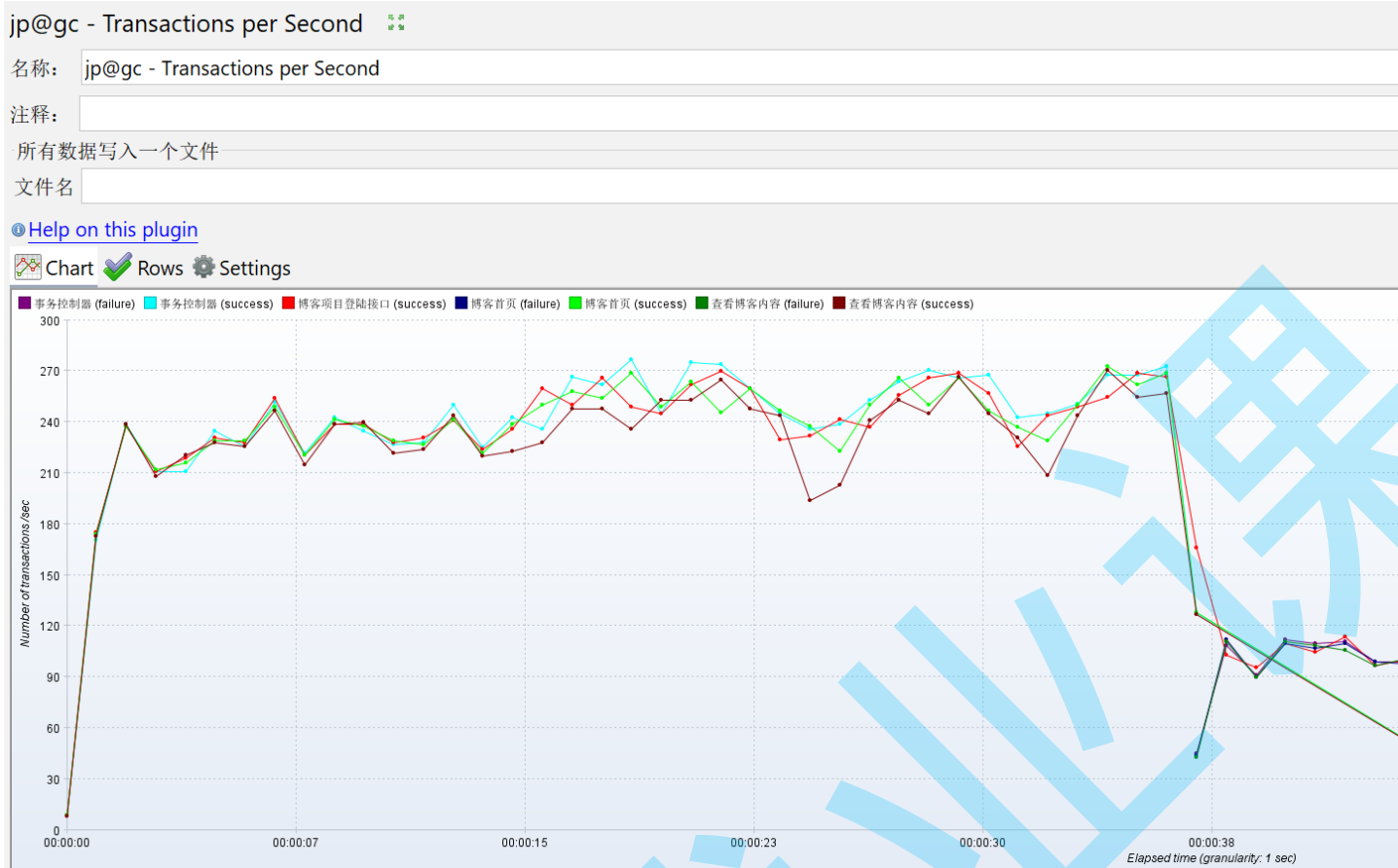
Response Times Over Time主要用于监听整个事务运行期间的响应时间。在测试过程中，它可以帮助测试人员观察并分析响应时间的实时平均值以及整体响应时间的走向。通过这一监听器，测试人员能够更直观地了解系统在不同时间点的响应性能，从而发现可能存在的性能问题或瓶颈。



Response Times Over Time的图形展示中，横坐标通常代表运行时间，而纵坐标则代表响应时间（单位是毫秒）。测试人员可以根据图形中的趋势线来判断响应时间的稳定性以及是否存在大的波动。例如，如果响应时间在某个时间点突然增加，这可能意味着系统在该时间点遇到了性能问题。

2.13.3 Transactions per Second (TPS)

JMeter中的Transactions per Second (TPS) 监听器是一个用于分析系统吞吐量的重要工具。TPS，即每秒事务数，表示一个客户机向服务器发送请求后服务器做出反应的过程。这个指标反映了系统在同一时间内处理业务的最大能力。TPS值越高，说明系统的处理能力越强。



在使用TPS监听器时，横坐标通常代表运行时间，而纵坐标则代表TPS值。通过监听器展示的图表，可以清晰地看到TPS值随时间的变化情况。在图表中，红色通常表示通过的TPS，而绿色可能表示失败的TPS。这有助于我们快速识别系统性能的变化和瓶颈。

3. 测试报告

JMeter测试报告是一个全面而详细的文档，它提供了关于测试执行结果的详细信息，帮助用户全面评估系统的性能并进行性能优化。

生成性能测试报告的命令：

- 1 `Jmeter -n -t 脚本文件 -l 日志文件 -e -o 目录`
- 2
- 3 `-n` : 无图形化运行
- 4 `-t` : 被运行的脚本
- 5 `-l` : 将运行信息写入日志文件，后缀为jtl的日志文件
- 6 `-e` : 生成测试报告
- 7 `-o` : 指定报告输出目录

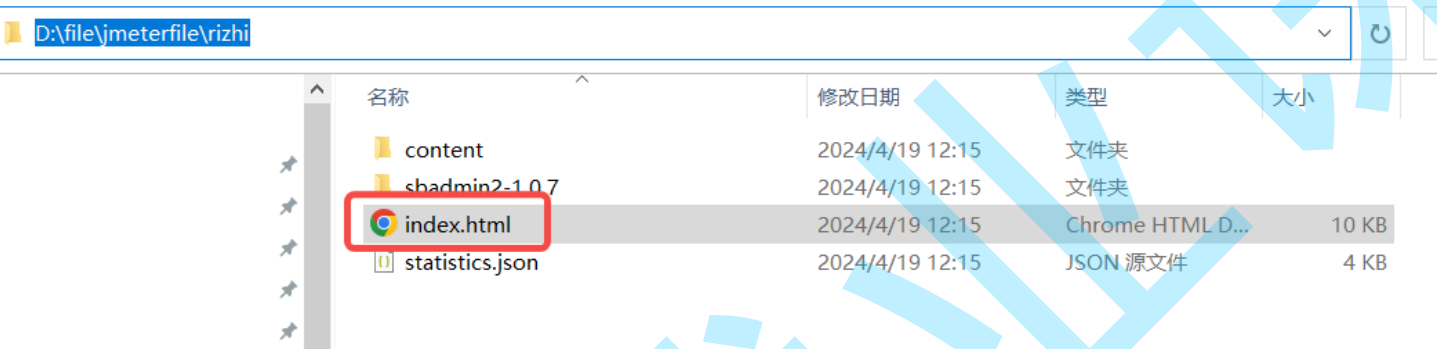
注意：日志文件和目录可以不存在，若为已经存在的情况下需要保证内容为空，否则会出现错误！

```
D:\file\jmeterfile>jmeter -n -t 第一个性能测试.jmx -l test.jtl -e -o rizhi/.
An error occurred: Cannot write to 'D:\file\jmeterfile\rizhi\.' as folder is not empty
errorlevel=1
```

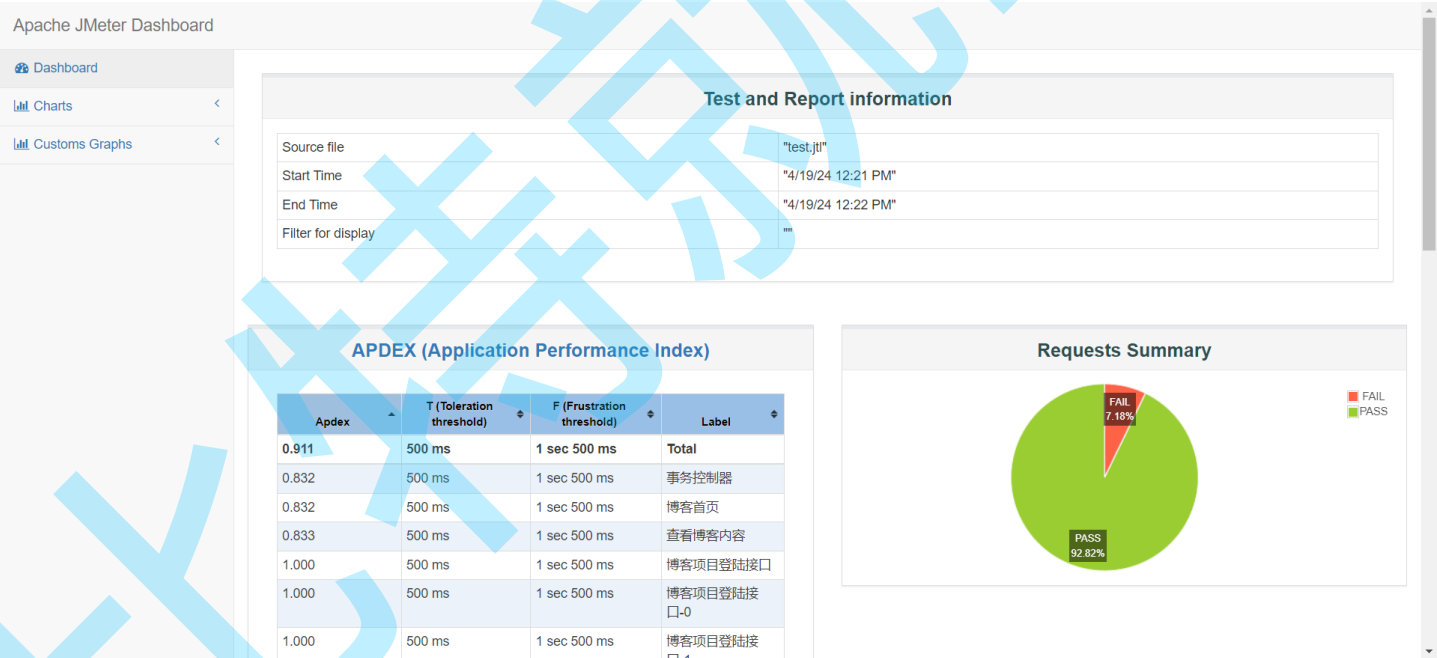
正确演示示例：

```
D:\file\jmeterfile>mkdir rizhi
D:\file\jmeterfile>jmeter -n -t 第一个性能测试.jmx -l rizhi.jtl -e -o rizhi/.
Creating summariser <summary>
Created the tree successfully using 第一个性能测试.jmx
Starting standalone test @ April 19, 2024 12:15:03 PM CST (1713500103356)
Waiting for possible Shutdown/StopTestNow/HeapDump/ThreadDump message on port 4445
summary +      1 in 00:00:00 =      6.1/s Avg:      43 Min:      43 Max:      43 Err:      0 (0.00%) Active: 1 Started: 1 Finishe
d: 0
summary +      8 in 00:00:01 =     13.8/s Avg:      4 Min:      3 Max:      7 Err:      3 (37.50%) Active: 0 Started: 3 Finish
ed: 3
summary =      9 in 00:00:01 =     12.0/s Avg:      8 Min:      3 Max:     43 Err:      3 (33.33%)
Tidying up ...    @ April 19, 2024 12:15:04 PM CST (1713500104316)
... end of run
```

性能测试报告生成成功后，在rizhi文件夹下将出现以下内容：



双击index.html文件，界面展示如下：



4. 性能分析

通过三大指标来分析性能问题

4.1 响应时间

如果响应时间超过了要求，代表系统到了瓶颈

注意事项：分析在多少线程的情况下发生了超标

响应时间变化的原因：

系统不稳定，有时快有时慢

随着并发压力变大而慢慢变慢，响应时间变高

4.2 错误率（可靠性）

高并发场景下，系统是否能够正常处理业务

要求：99.99%可靠，99.9999%

错误率高的原因：

接口请求错误

服务器无法继续处理，达到了瓶颈（代码写的不好，内存泄漏、硬件资源等）

后端系统限流（系统里配置了不能超过多少并发）、熔断、降级

什么是熔断、降级？

熔断：防止系统因某个服务的故障而整体崩溃。当电商平台上用户支付时，收银台发现某个支付渠道，如微信支付失败率突增，超时严重，那么就可以临时把这个支付方式熔断掉

降级：主动关闭一些非核心功能，以确保核心功能的正常运行。某次腾讯视频挂了的时候，用户名称默认显示腾讯用户，这也是一种降级方式，用兜底名称做展示

4.3 吞吐量

吞吐量越大，性能越好；吞吐量相对稳定或者变低，可能系统达到了性能瓶颈

吞吐量变化规律：

波动很大：代表系统性能不稳定

慢慢变高，再趋于稳定：和并发量强相关。如果并发量小于吞吐量，慢慢增大并发量，吞吐量也会随之增加

慢慢变低，并发量也减少了：要么说明性能测试要结束了，并发减少；也可能是系统变的卡顿，从而导致响应时间变慢，导致单个线程发起的并发量变少