

# 如何使用 Jmeter 做接口测试

## 修订记录

修订时间	修订人员	修订原因
2018.1.28	鲁健	新建
2018.5.7	鲁健	完成初稿, 后续待完善细节

Github 地址：

[https://github.com/monleylu/How\\_To\\_Use\\_Jmeter\\_To\\_Test\\_Interface.git](https://github.com/monleylu/How_To_Use_Jmeter_To_Test_Interface.git)

## 初识 Jmeter

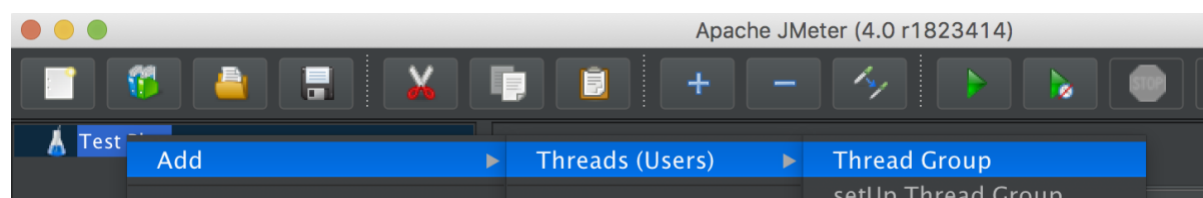
Jmeter 是一套开源的性能测试工具，纯 java 编写，所以拥有良好的跨平台运行能力，最初 jmeter 设计用于 web 测试，现在已经支持各种协议测试，包括 soap、rest、ftp 等。

Jmeter 提供图形界面和命令行运行两种方式，图形界面方便开发调试脚本，命令行模式多用于进行性能测试。同时命令行模式也方便用于集成测试。

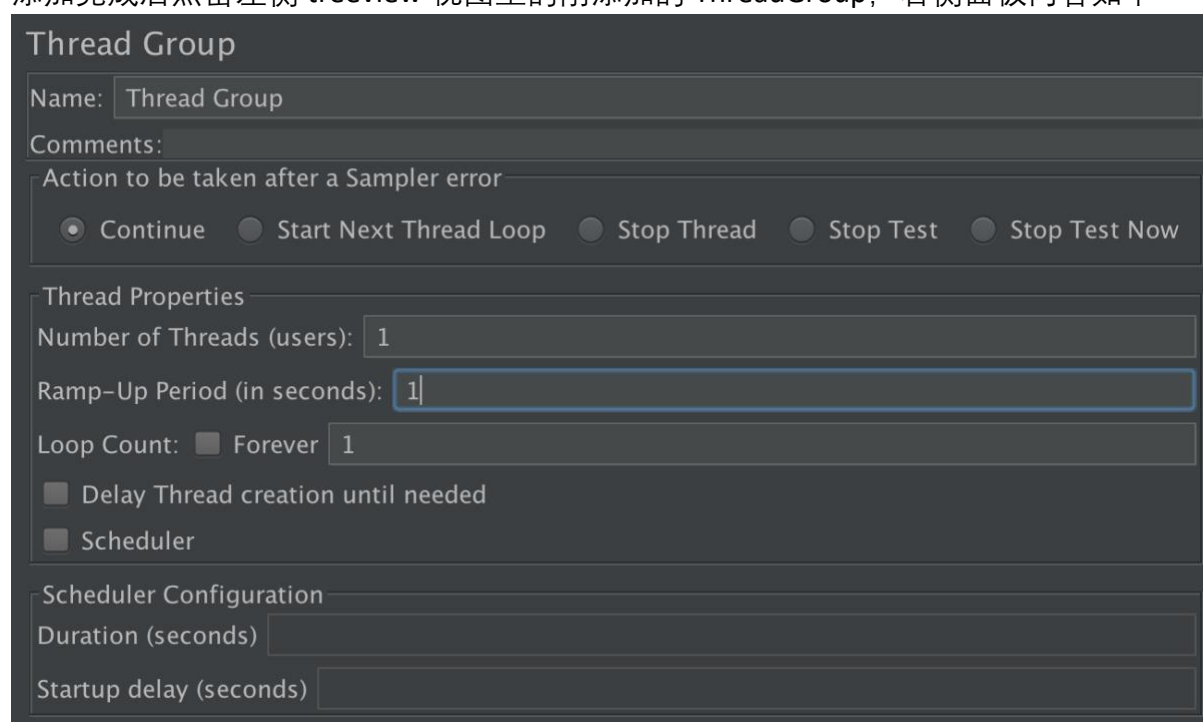
### class1：创建 Http 测试计划

目的：创建一个最简单的 http 测试接口，并查看响应结果

1. 选中“Test Plan”，右键“Test Plan”，添加 Thread Group，如下图所示。



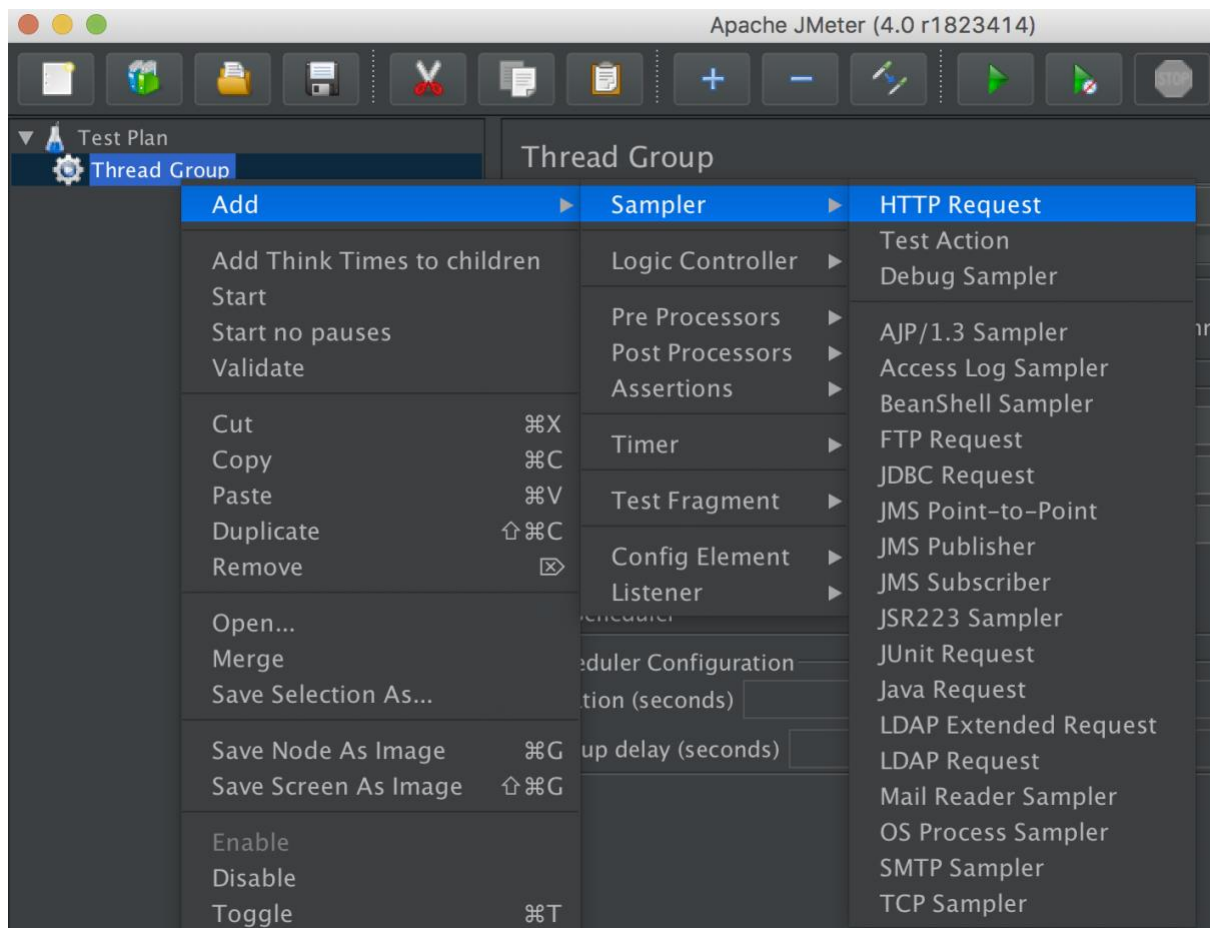
添加完成后点击左侧 treeview 视图里的刚添加的 ThreadGroup，右侧面板内容如下



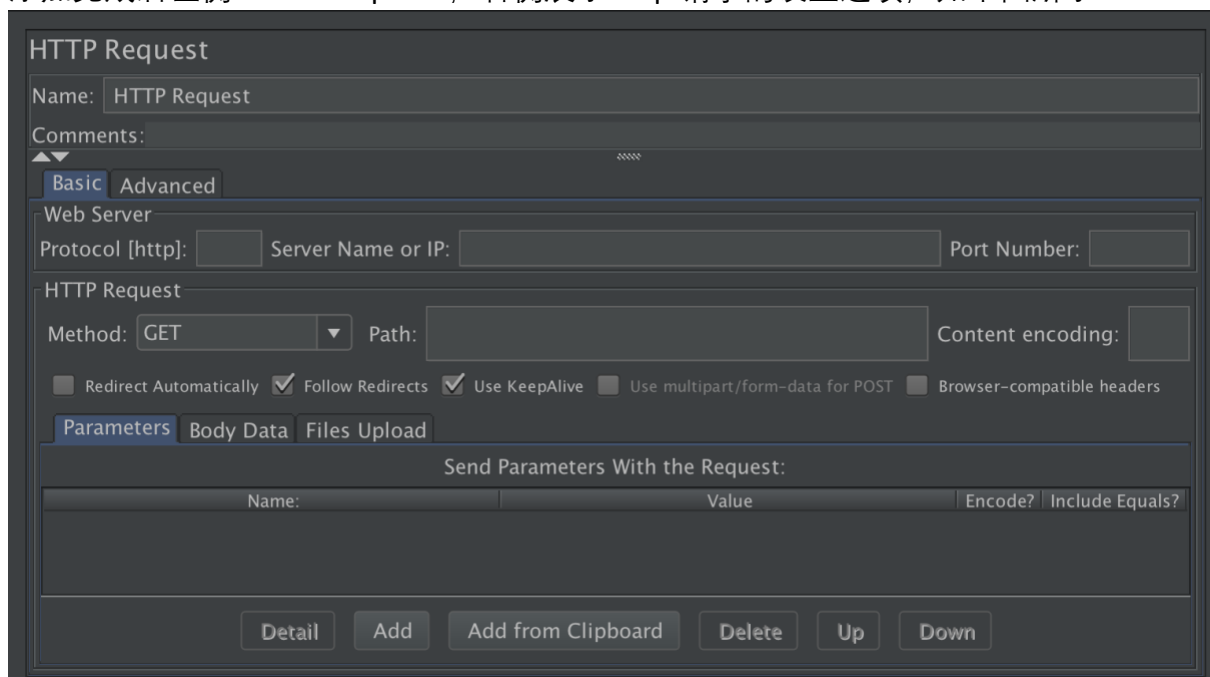
“Action to be taken after a Sampler error” 面板下的选项分别代表当前进程组下面的 sampler 出错时如何继续操作

“Thread Properties”下面的“Number of Threads”代表多少并发请求数，“Ramp-Up Period”表示多久启动完毕所有的进程。如果设置为零，则所有进程同时启动。

2. 选中“Thread Group”，右键添加“HTTP Request”，如下图所示



添加完成后左侧“HTTP Request”，右侧展示 http 请求的设置选项，如下图所示



其中“Web Server”字段填写要远程访问的服务地址信息，“Protocol”为访问协议类型，默认为“http”，也可以填写“https”或者“File”，“Server Name or IP”填写要访问的域名或者 IP 地址，注意不要包含协议类型，如“http：//”。“Port Number”填写端口号，“Path”字段填写请求的 url 路径。

例子：

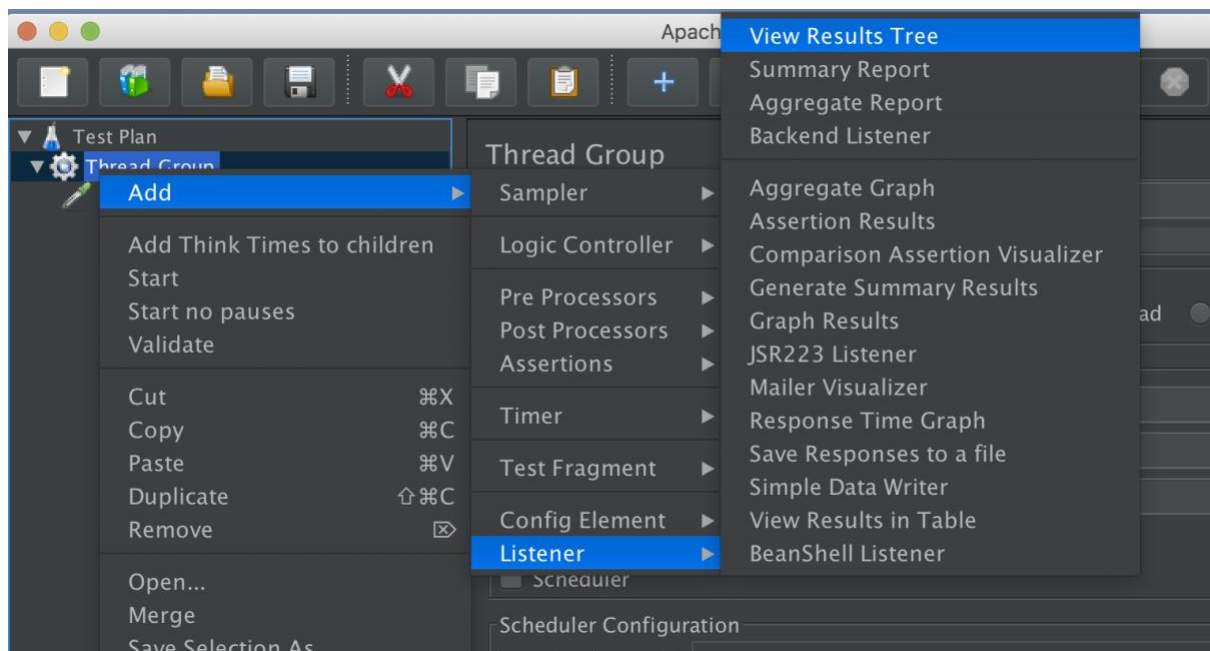
访问百度搜索“jmeter”

The screenshot shows the 'HTTP Request' configuration in JMeter. The 'Name' field is '百度搜索方式一'. The 'Web Server' section has 'Protocol [http]:' set to 'http', 'Server Name or IP:' set to 'www.baidu.com', and 'Port Number:' is empty. The 'HTTP Request' section has 'Method:' set to 'GET' and 'Path:' set to '/s?wd=jmeter'. The 'Content encoding:' field is empty. Below the 'Path' field, there are checkboxes for 'Redirect Automatically' (unchecked), 'Follow Redirects' (checked), 'Use KeepAlive' (checked), 'Use multipart/form-data for POST' (unchecked), and 'Browser-compatible headers' (unchecked). The 'Parameters' tab is selected, showing a table with columns 'Name', 'Value', 'Encode?', and 'Include Equals?'. The table is currently empty. At the bottom, there are buttons for 'Detail', 'Add', 'Add from Clipboard', 'Delete', 'Up', and 'Down'.

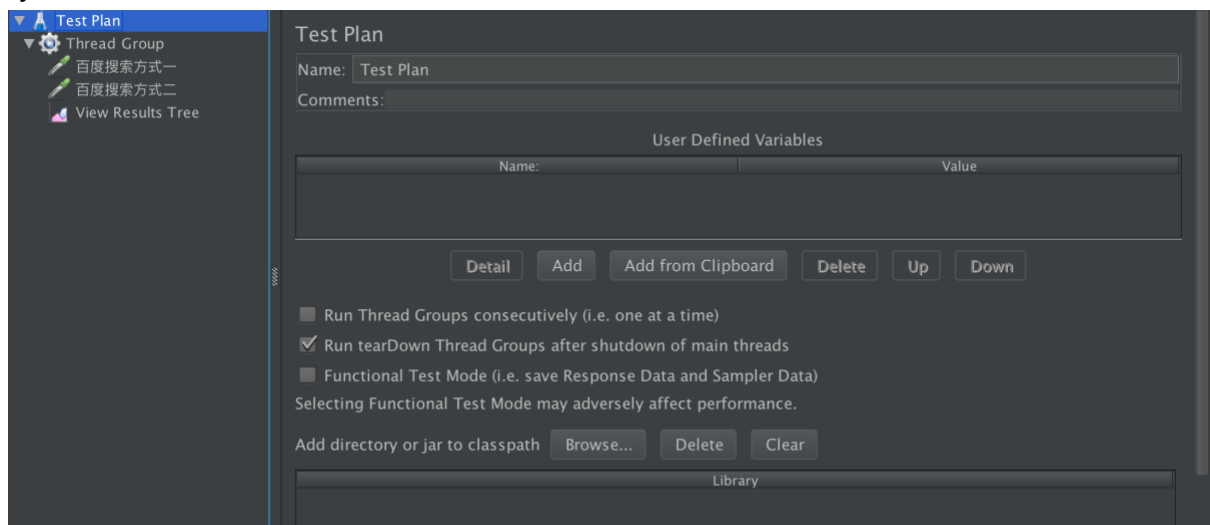
The screenshot shows the 'HTTP Request' configuration in JMeter. The 'Name' field is '百度搜索方式二'. The 'Web Server' section has 'Protocol [http]:' set to 'http', 'Server Name or IP:' set to 'www.baidu.com', and 'Port Number:' is empty. The 'HTTP Request' section has 'Method:' set to 'GET' and 'Path:' set to '/s'. The 'Content encoding:' field is empty. Below the 'Path' field, there are checkboxes for 'Redirect Automatically' (unchecked), 'Follow Redirects' (checked), 'Use KeepAlive' (checked), 'Use multipart/form-data for POST' (unchecked), and 'Browser-compatible headers' (unchecked). The 'Parameters' tab is selected, showing a table with columns 'Name', 'Value', 'Encode?', and 'Include Equals?'. The table contains one row with 'Name' as 'wd', 'Value' as 'jmeter', 'Encode?' as unchecked, and 'Include Equals?' as checked. At the bottom, there are buttons for 'Detail', 'Add', 'Add from Clipboard', 'Delete', 'Up', and 'Down'.

上面两张截图功能一样都是访问使用百度搜索关键字 jmeter，第一张是把请求内容放到了 path 中，第二张是把访问的参数放到了 Parameters 中。两种方式都可以正常访问。

3. 选中“Thread Group”，右键添加“View Results Tree”，如下图所示

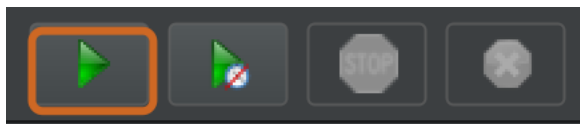


如上，我们就创建好了一个最简单的测试 http 协议接口的测试计划，完整界面如下所示

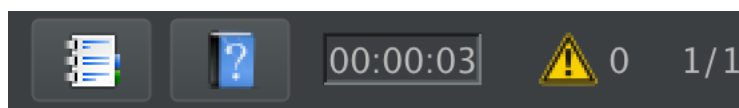


#### 4. 运行接口测试并查看测试结果

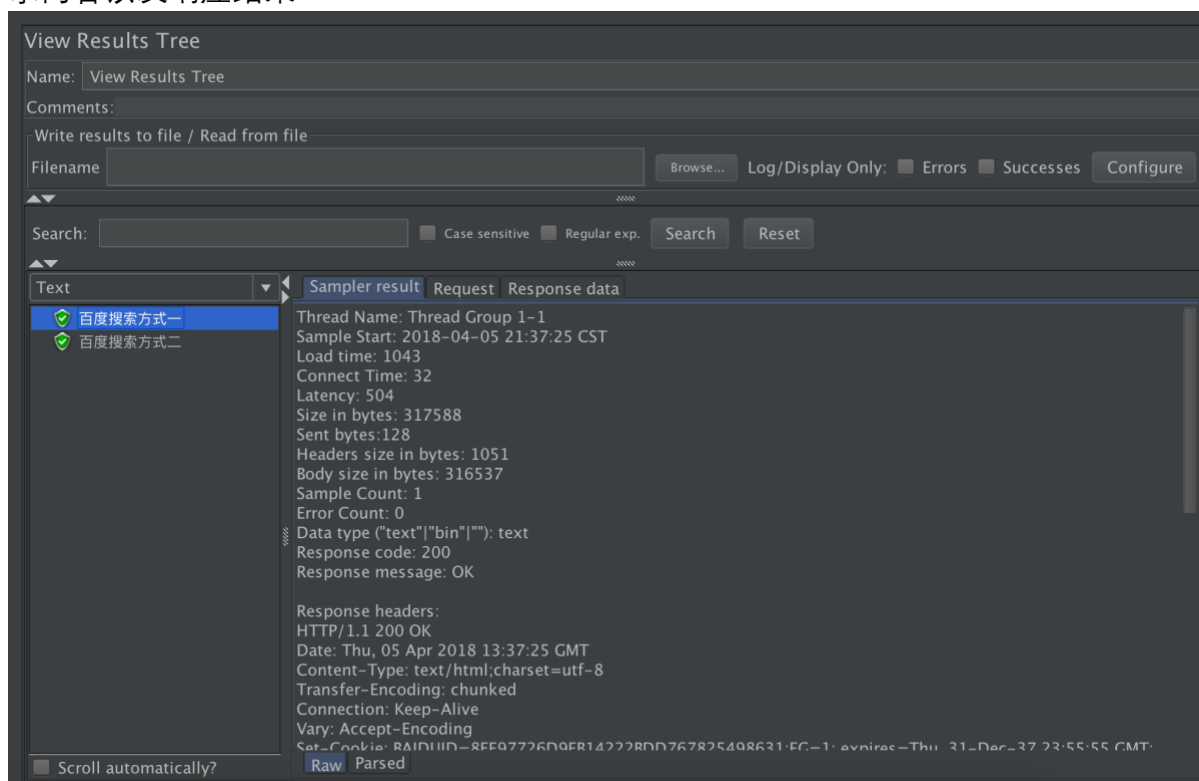
点击快捷菜单栏的绿色开始按钮或者菜单栏的 Run—Start，就会开始运行所写的接口测试了



运行过程中，右上侧的图标显示是否有告警错误、已运行时长、目前运行的进程数目，点击黄色感叹号图标，会展示 jmeter 运行过程中的日志，感叹号后面的数字表示有多少错误发生



运行结束或者运行过程中查看各个接口响应结果，点击左侧“View Results Tree”，右侧展示运行的请求结果，点击不同的请求名称就可以查看运行请求的数据，包括请求内容以及响应结果



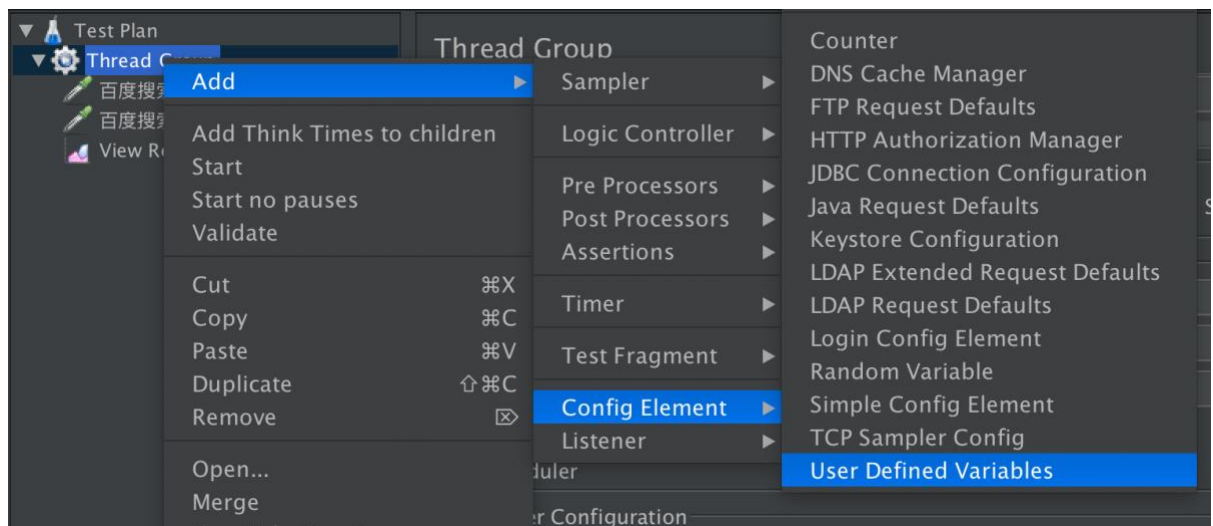
这样我们就完成一个简单的并且完整的请求响应测试

## class2：参数化

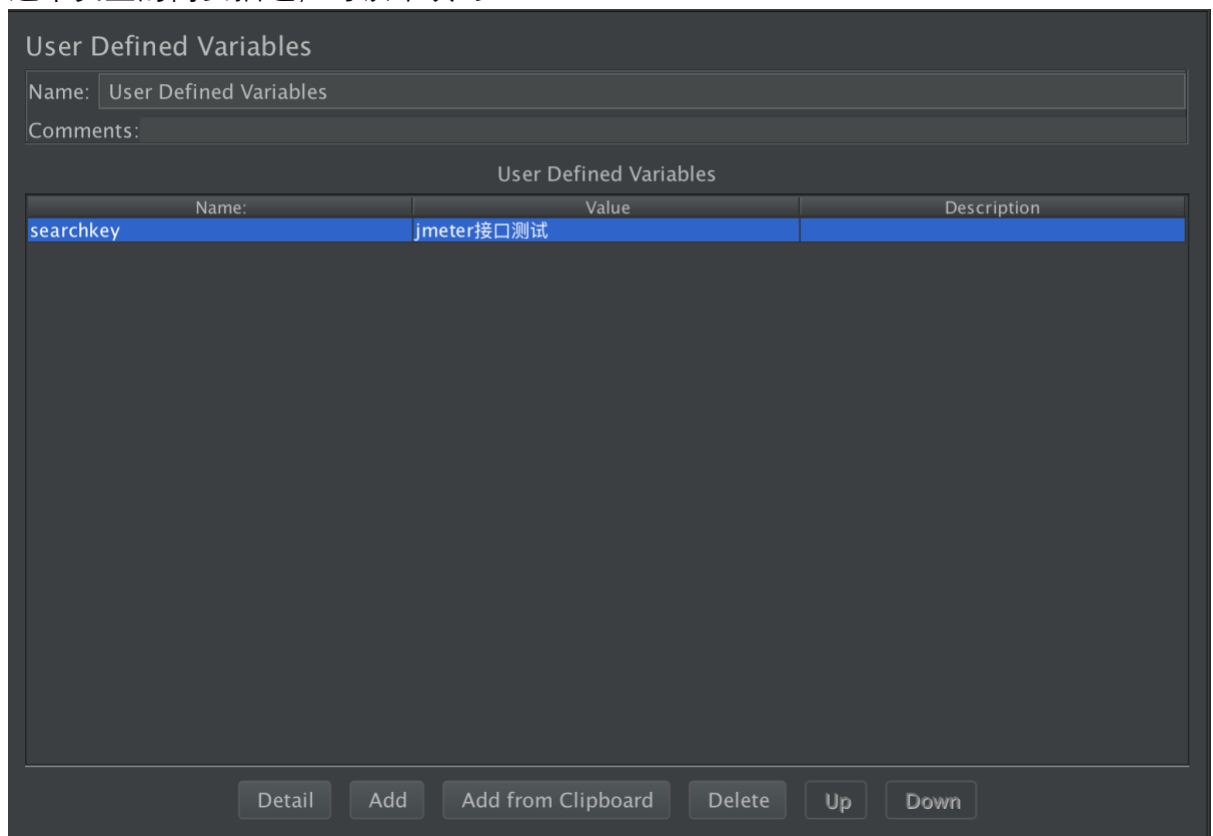
还是以百度搜索内容为例，搜索的参数需要改变，可以直接到各个 sample 中去找要修改的参数，也可以在环境变量中直接修改参数即可

- 对搜索的关键字参数化

在 Thread Group 元素上右键添加“User Defined Variables”，如下图所示



点击左侧“User Defined Variables”，右侧展示 UDV 的面板内容，点击下方的添加按钮“Add”，“name”列填写变量名称，“value”列填写变量内容，“description”列填写对这个变量的简要描述，可以不填写



点击需要使用关键字的 httpsample，修改“wd”参数的值，改成“\${变量名称}”格式，如下图所示

### HTTP Request

Name: 百度搜索方式一

Comments:

Basic Advanced

Web Server

Protocol [http]: Server Name or IP: www.baidu.com Port Number:

HTTP Request

Method: GET Path: /s?wd=\${searchkey} Content encoding:

☐ Redirect Automatically ☒ Follow Redirects ☒ Use KeepAlive ☐ Use multipart/form-data for POST ☐ Browser-compatible headers

Parameters Body Data Files Upload

Send Parameters With the Request:

Name:	Value	Encode?	Include Equals?
-------	-------	---------	-----------------

Detail Add Add from Clipboard Delete Up Down

点击运行，查看请求参数，如下图所示，搜索的关键字已经改成了我们预先设置的内容了

### View Results Tree

Name: View Results Tree

Comments:

Write results to file / Read from file

Filename Browse... Log/Display Only: ☐ Errors ☐ Successes Configure

Search: ☐ Case sensitive ☐ Regular exp. Search Reset

Browser

百度搜索方式一

Sampler result Request Response data

GET http://www.baidu.com/s?wd=jmeter接口测试

GET data:

[no cookies]

Request Headers:

Connection: keep-alive

Host: www.baidu.com

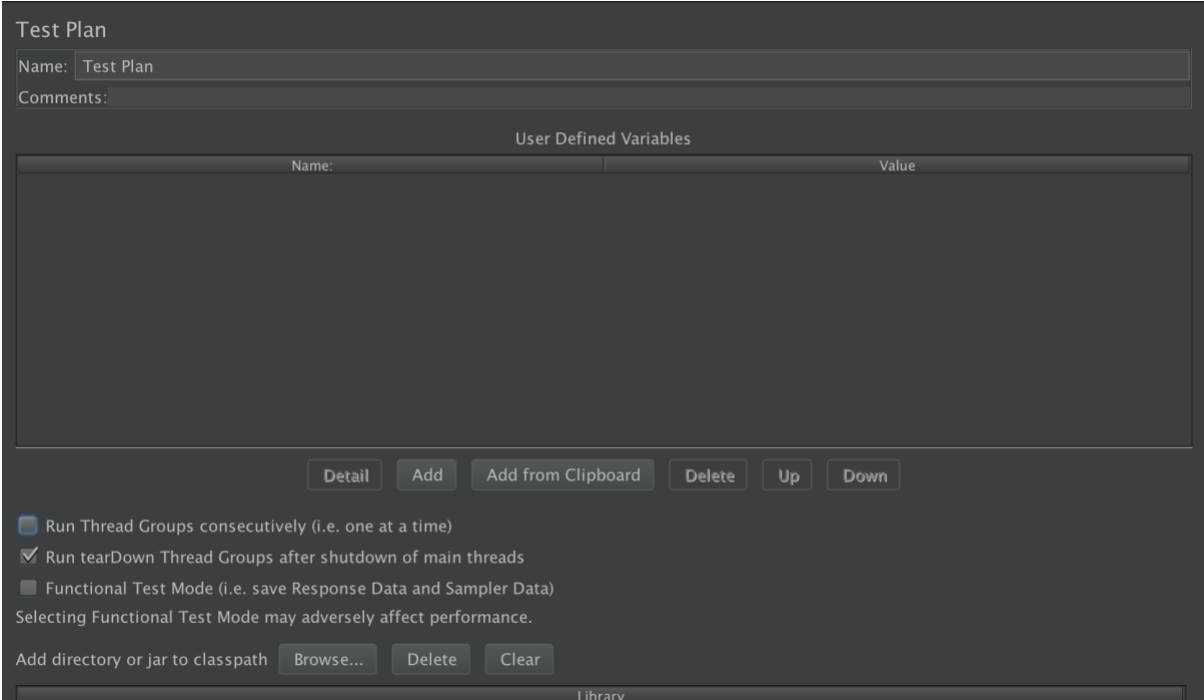
User-Agent: Apache-HttpClient/4.5.5 (Java/1.8.0\_121)

☐ Scroll automatically? Raw HTTP

PS :



左侧元素树的“Test Plan”元素也包含一个“User Defined Variables”，这边也可以添加变量，如下图所示

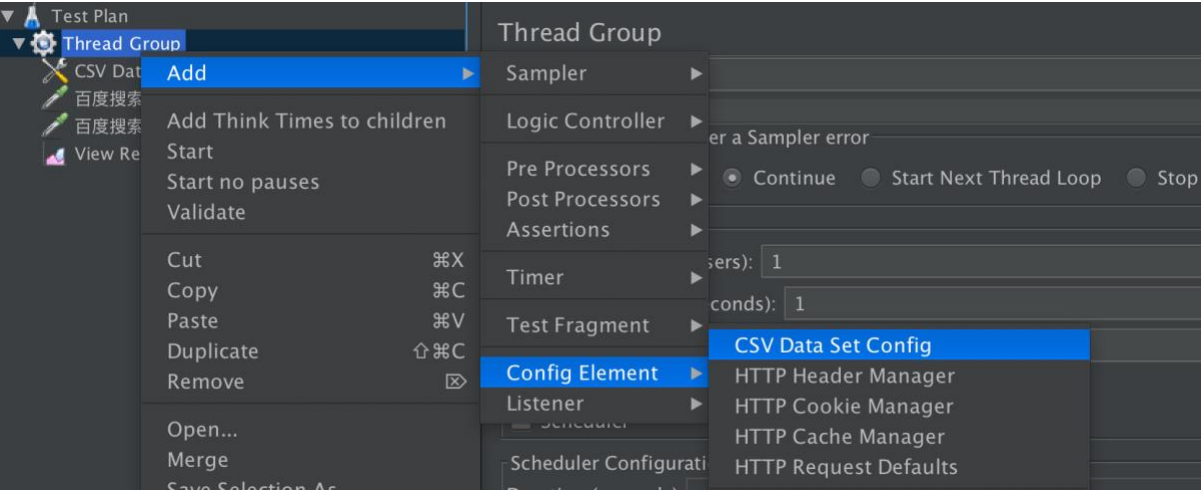


“Test Plan”和 “User Defined Variables”定义的变量都是全局有效的，变量名称也可以相同，相同变量名称时值等于最后定义的变量内容。

➤ 从文件读取变量值

如搜索关键字每次查询都想改变，而且变动的值是确定的时候，可以通过“CSV Data Set Config”读取文件的方式获取需要的值

右键 Thread Group 添加“CSV Data Set Config”，如下图所示



点击 Browse 按钮，选择待读取的文件，文件内容格式如下图所示

Filename 为待读取的文件名称

File encoding 为文件编码格式，一般选择 UTF-8 格式，因为我们创建的文本文件基本都是这种格式，如果不是根据实际文件格式选择即可

Variable names 为给 jmeter 使用的变量名称，这个字段可以为空也可以填写想要填写的值，这个字段配合下面的“ignore first line”来设置读取的 csv 文件第一行是否使用。

当这个字段不设置为空时，“ignore first line”字段不生效，这时候 csv 文件的第一行会被当作变量名称，如例子中的第一行变量名称为 `${searchkeyfirst}` 和 `${searchkeysecond}`；

当这个字段设置不为空时，如 searchkey,searchkey2，当“ignore first line”字段为 True，第一行就会被忽略，第一次变量读到的值就是 test1 和 jmeter1，；当“ignore first line”字段为 False，第一次变量读到的值就为 searchkeyfirst 和 searchkeysecond。

Delimiter 字段表示每行每个字段之间分隔符是什么，例子中文件是以逗号分割的，这边也就填写逗号。一般这个字段分隔符没有什么限制，可以用任意值作为分隔符，如逗号、分号、\t 等等，不过从实际使用情况看，建议还是以逗号作为分隔符最好

Allow quoted data 字段表示使用允许使用双引号，当字段为 True 时，可以使用双引号包含含有 delimiter 字符的字段

Recycle on EOF 字段和 Stop thread on EOF 两个字段配合使用

当 Recycle on EOF 为 False，Stop thread on EOF 为 False 时，当进程数多余 csv 行数，即读到文件末尾后，变量值都会被设置成 <EOF>，可以通过修改 bin 目录下的 jmeter.properties 文件中的 csvdataset.eofstring 属性改变默认值；

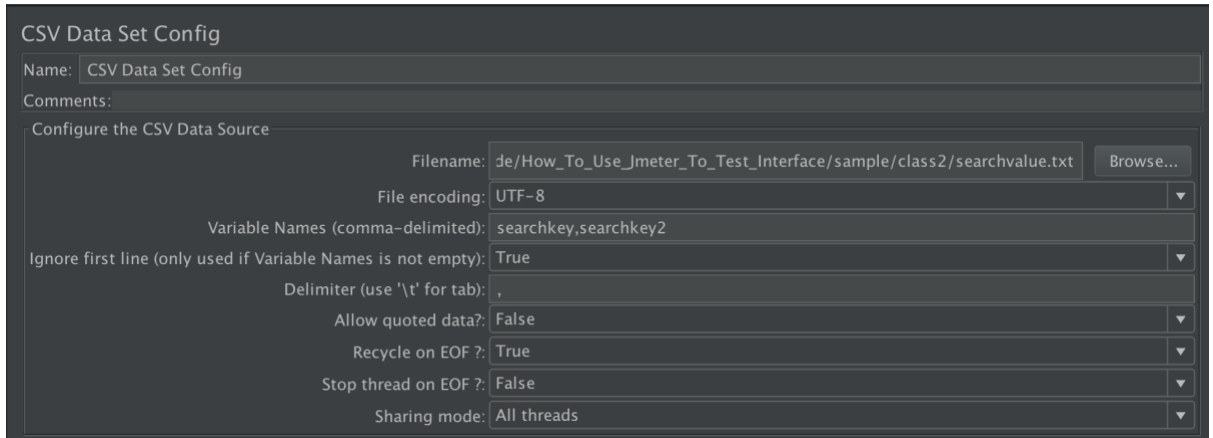
当 Recycle on EOF 为 False，Stop thread on EOF 为 True 时，当读到文件末尾时，如果还有进程未启动执行，即进程数大于 csv 文件行数，剩余未启动的进程不会启动；

当 Recycle on EOF 为 True，不论 Stop thread on EOF 为 False 还是 True，都会有多少进程启动多少进程，循环读取文件里的变量值。

Sharing mode 表示 csv 文件里的内容共享模式。All threads – 默认所有进程都共享这些数据。

Current thread group – 每个进程组之间共享

Current thread - 数据不共享，每个进程单独一份数据拷贝，都从第一行数据开始读取



csv 文件内容如下图所示，不同列之间使用逗号隔开，注意需要是英文的逗号

```
1 searchkeyfirst,searchkeysecond
2 test1,jmeter1
3 test2,jmeter2
4 test3,jmeter3
5 test4,jmeter4
```

## class3：进程间共享数据

章节 2 里讲的 csv 文件参数化就是其中一种共享数据方式，另外可以通过\_\_setProperty 或者 props 设置进程间共享数据

点击“Test Plan”添加“setUp Thread Group”，选中“setUp Thread Group”添加“BeanShell Sampler”，“setUp Thread Group”和普通“Thread Group”的不同就是会第一个执行，和 Junit 单元测试的 setUp 一样。

添加如下代码：

```
1. //wiki:https://docs.oracle.com/javase/8/docs/api/java/util/Properties.html
2.
3. boolean boolvar =true;
4.
5. //无论 ifelse 条件，searchwd1 的值永远都会是最后一个设置的值，哪怕程序并未走入后面的流程
6. if(boolvar){
7.     log.info("-----true----");
8.     ${__setProperty(searchwd1,monleyluttrue,,)};
9.     log.info("-----true2222----");
10. }else{
11.     log.info("-----false----");
12.     ${__setProperty(searchwd1,monleylufalse,,)};
13.     log.info("-----false2222----");
14. }
15.
16.
```

```

17. if(boolvar){
18.     props.setProperty("searchwd2","monleylu2true");
19. }else{
20.     props.setProperty("searchwd2","monleylu2false");
21. }
22.
23.
24. log.info("-----");
25. //另外一种获取数据的方式
26. //这里可以观察到，程序并未走进 else 流程，但是 searchwd1 的值仍然被设置成了 monleylufalse
27. //这并不是程序 bug，是这种设置的加载方式
28. ${__property(searchwd1,data,)};
29. log.info(vars.get("data"));

```

这里是分别设置了两个进程间需要传递的参数。

\_\_setProperty 设置的参数通过 \${\_\_P(searchwd1)} 获取；

Props 设置的参数需要通过 props.getProperty 获取

```

1. vars.put("searchwd2",props.getProperty("searchwd2"));
2. log.info("-----" + vars.get("searchwd2"));

```

PS：

\_\_setProperty 方法不支持条件语句，如在特定条件下设置参数，只支持固定需要动态传递的参数直接写好，props 可结合 ifelse 等语句条件设置

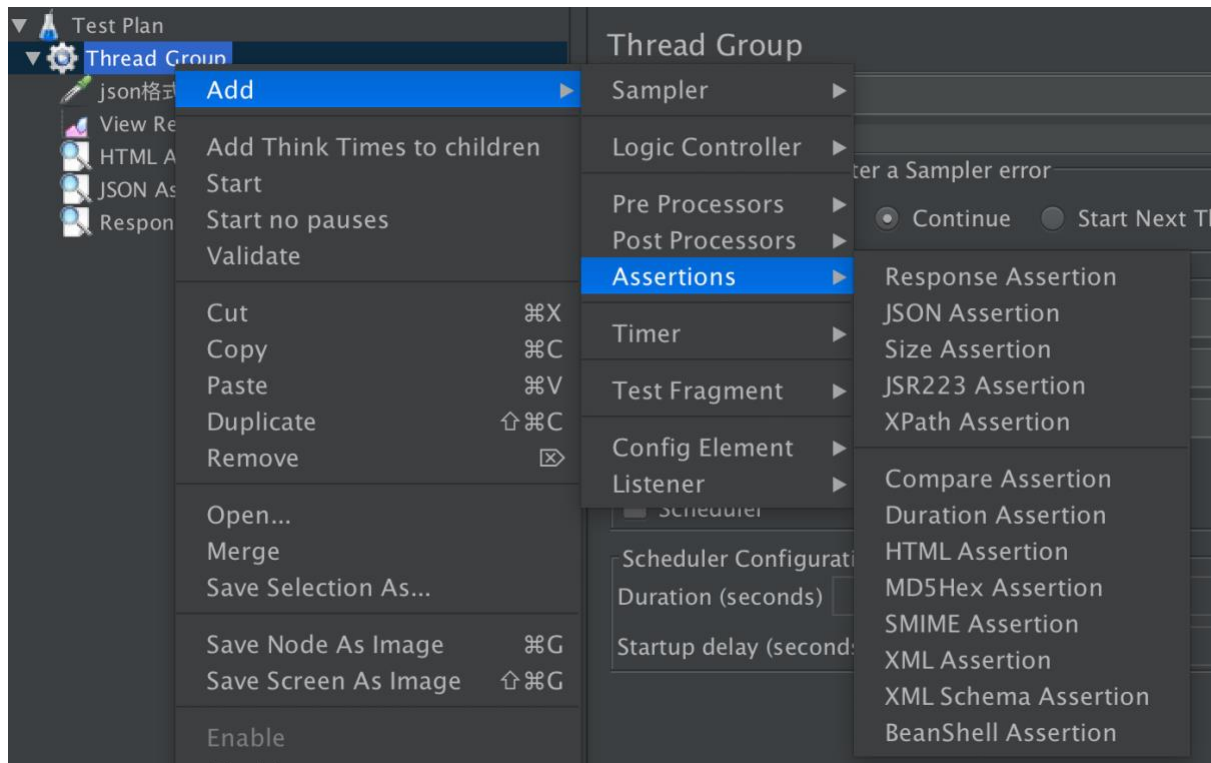
## class4：响应处理

对响应进行处理有好多方式，如获取指定数据时，可以使用正则或者 beanshell 等，如对结果进行判断可以使用断言等。下面会对常见使用方式做简单介绍

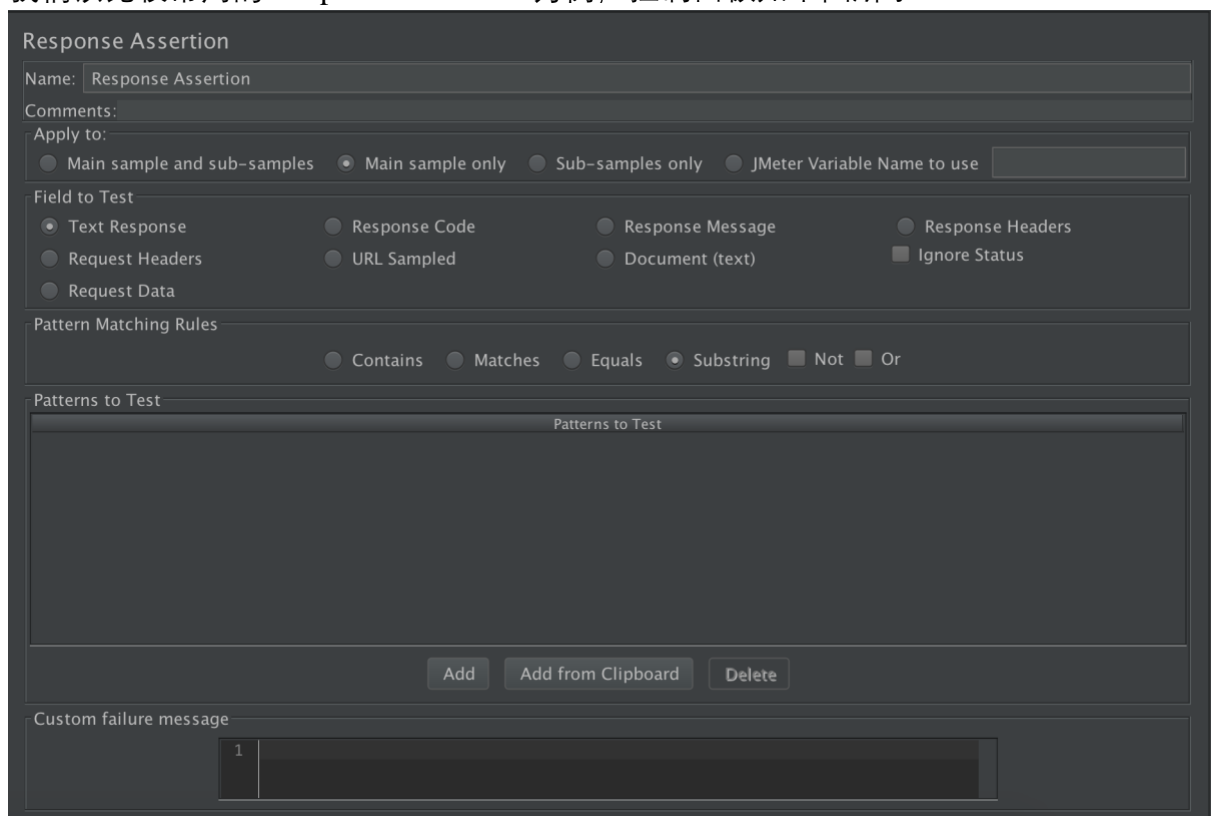
### 断言

断言提供了对响应数据进行处理的一种方式，并且会在每个请求结束后依次对同一级别的请求进行断言处理。如果需要对特定请求进行断言，则需要将断言添加为该请求的一个子属性。

断言支持的种类繁多，如下图所示



我们以比较常用的 Response Assertion 为例，控制面板如下图所示



Apply to 表示这个断言要应用到哪个请求上或者响应上，默认为“Main sample only”，如果断言也需要应用到子请求上需要勾选“Main sample and sub-samples”，根据具体请求情况选择不同的应用模式。

Field to Test 断言指定的内容，具体如面板所示，这里要注意有个“Ignore status”的复选框，默认情况下 Jmeter 会将响应状态码为 4XX 和 5XX 的请求标记为失败，如果勾选了这个复选框，则会在进行断言判断之前强制默认请求为成功。

Pattern Matching Rules 表示匹配规则，“contains”和“matches”为正则匹配，“equals”和“substring”为文本匹配。“contains”和“matches”的区别是：

“contains”：当整个文本部分或者全部匹配正则表达式时为 true，否则为 false；

“matches”：当整个文本匹配正则表达式时为 true，否则为 false；

如接口响应为

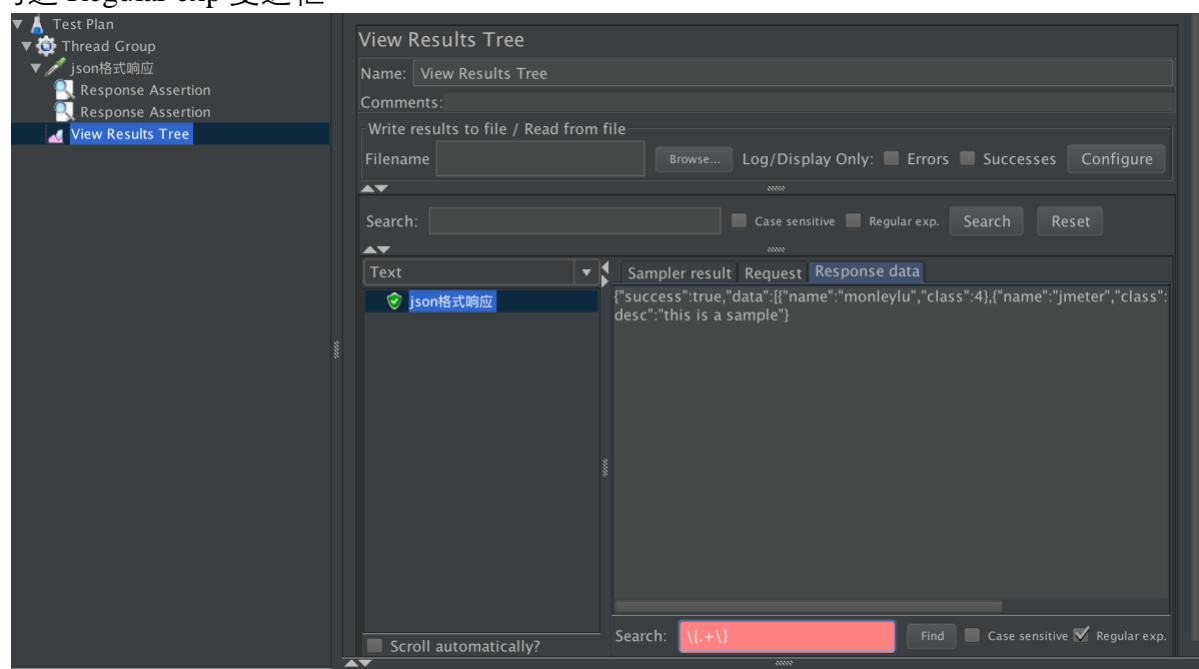
```
1. {"success":true,"data":[{"name":"monleylu","class":4},{ "name":"jmeter", "class":5},{ "name":"tutorial", "class":6}], "desc":"this is a sample"}
```

当正则内容为 'data':\{.+} 时，选择“contains”时，结果断言为 true，其他选项为 false；

当正则内容为 \{.+} 时，选择“matches”时，结果断言为 true，当然这时候选择“contains”也为 true，因为的确也是包含的意思。

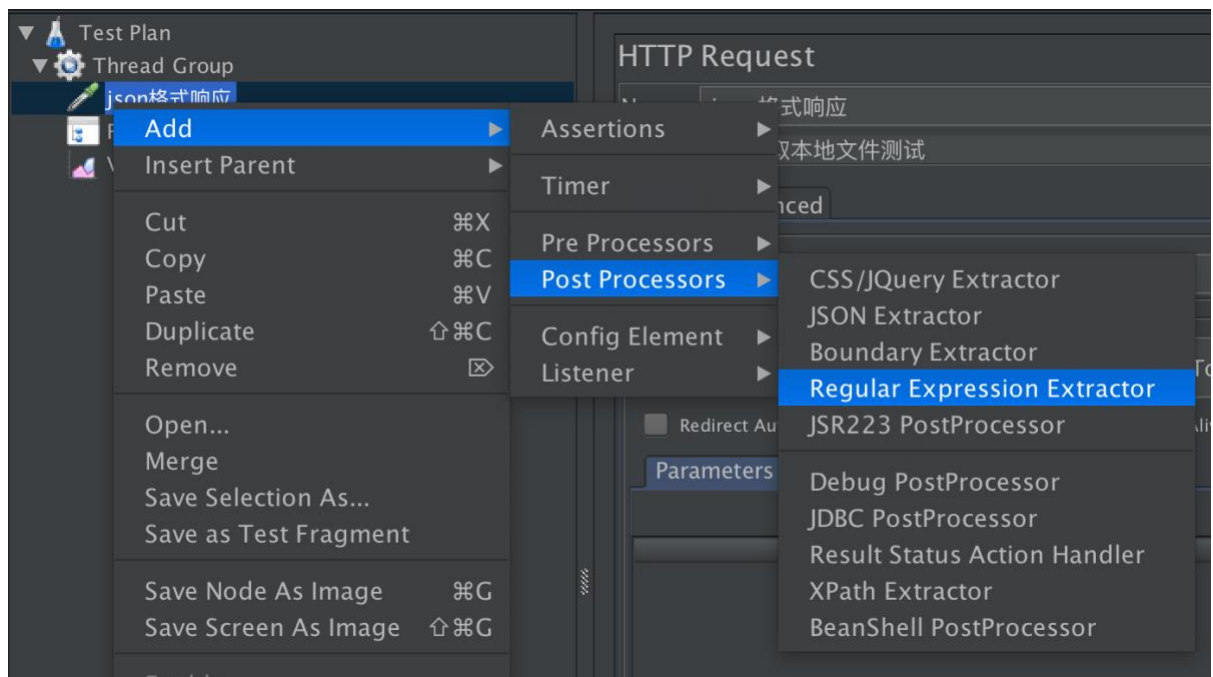
注意当匹配文字中还有正则表达式字符时，需要使用反斜杠 \ 转义。

Jmeter 自带了一个正则表达式测试的功能，可以方便的测试正则表达式正确性。注意勾选 Regular exp 复选框



## 正则处理器

正则表达式处理器属于后置处理器的一种，允许用户使用 Perl-type 标准的正则语法匹配响应内容并保存匹配内容。如下图所示添加正则表达式处理器



Regular Expression Extractor

Name: Regular Expression Extractor

Comments:

Apply to:

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

Field to check

☒ Body ☐ Body (unescaped) ☐ Body as a Document ☐ Response Headers ☐ Request Headers ☐ URL ☐ Response Code ☐ Response Message

Name of created variable: regName

Regular Expression: "name": "(.+?)"

Template (\$i\$ where i is capturing group number, starts at 1): \$1\$

Match No. (0 for Random): 0

Default Value: ☐ Use empty default value

属性	描述	必填
Name of created variable	Jmeter 用于存储匹配结果的变量名称。 PS：不同匹配组以 <b>[refname]_g#</b> 格式存储， <b>[refname]</b> 表示用户填写的变量名称， <b>#</b> 表示匹配组，从 0 开始，0 表示全部匹配内容，1 表示第一个匹配组，依此类推	是
Regular Expression	用于匹配响应结果的正则表达式，至少需要包含一组"( )"括号以匹配字符响应的一部分，除非匹配全部表达式	是

Template	模版用于从正则表达式匹配结果中获取匹配组数据，规则为 ' <b>\$1\$</b> ' 表示第一个组，' <b>\$2\$</b> ' 表示第二个组，' <b>\$0\$</b> ' 表示正则匹配的所有结果	是
Match No. (0 for Random)	由于正则表达式可能多次匹配，这个参数用于表示使用哪次匹配数据，有三种参数方式 数字 0:表示随机取一个数据； 正整数 N:表示取第 N 次匹配的数据 负数：一般会 and ForEach 控制器联合使用循环所有的匹配数据	是
Default Value	如果正则表达式没有匹配到数据，这个值会被赋值给变量	否

如果 **Match No** 设置为非负数的自然数，并且正则表达式匹配到响应结果，那么如下变量会被赋值：

- refName - the value of the template
- refName\_gn, where n=0,1,2 - the groups for the match
- refName\_g - the number of groups in the Regex (excluding 0)

如果没有匹配到响应结果，那么 refName 会被设置为默认值（除非用户没有填写匹配失败赋值默认值），并且如下的变量会被删除：

- refName\_g0
- refName\_g1
- refName\_g

如果 **Match No** 设置为负数，那么所有的匹配结果都会被处理，并且如下变量会被初始化赋值：

- refName\_matchNr - the number of matches found; could be 0
- refName\_n, where n = 1, 2, 3 etc. - the strings as generated by the template
- refName\_n\_gm, where m=0, 1, 2 - the groups for match n
- refName - always set to the default value
- refName\_gn - not set



beanshell

参考自定义开发扩展章节

## class5:自定义扩展开发

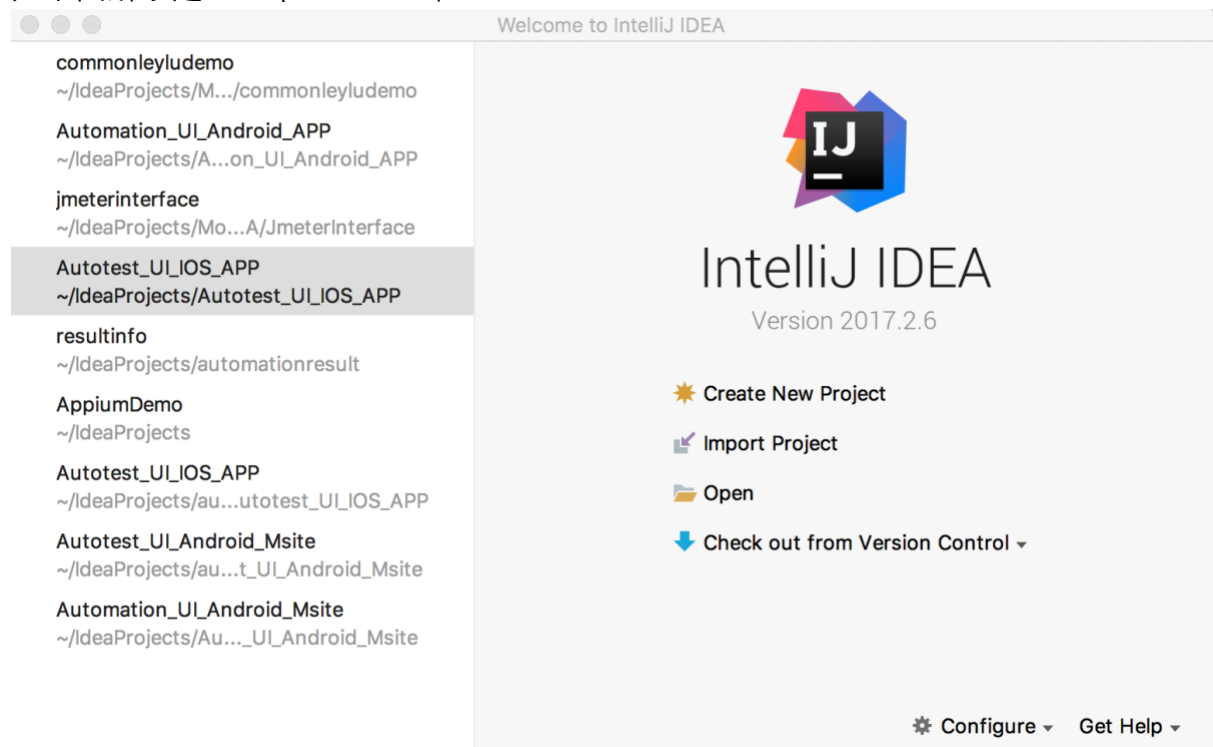
jmeter 支持 groovy、java、jexl、javascript 四种语言扩展脚本。都是跑在 jvm 之上的脚本语言。可以通过前置处理/后置处理器的 JSR223 或者 BeanShell 增强 jmeter 的处理能力。

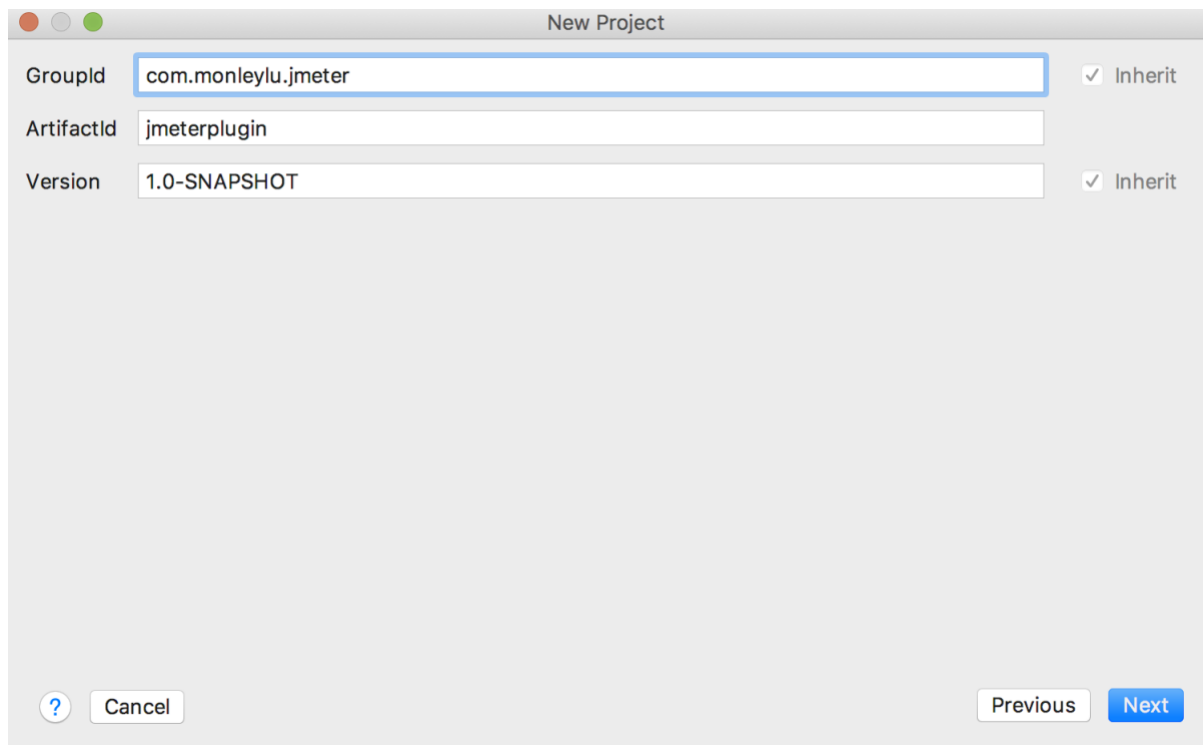
以下例子以 beanshell 为例，编写 java 扩展。

涉及一些开发工具，如 intellij idea 或者 eclipse，maven 或 gradle 等，这些知识点百度下吧。

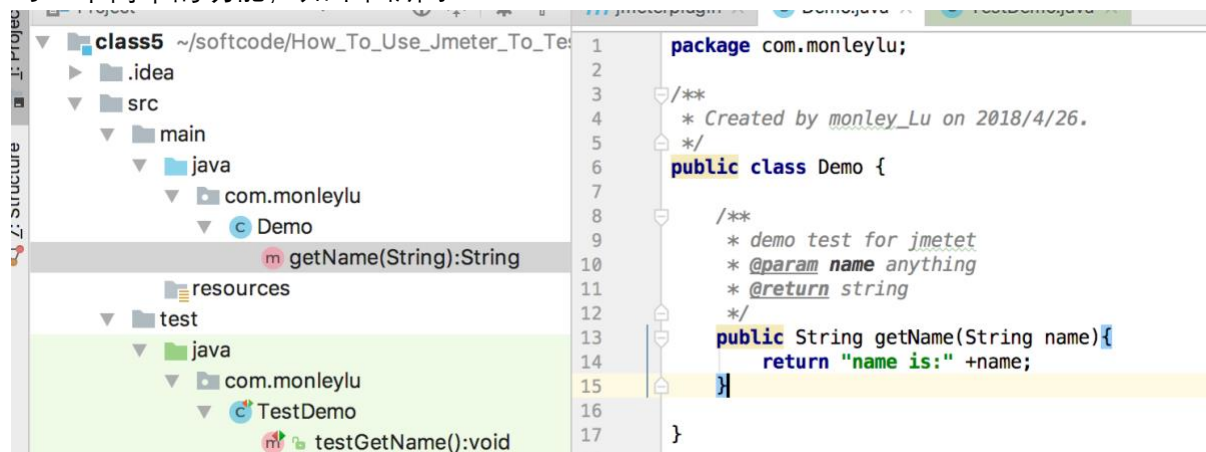
简单以 intellij idea 和 maven 写这个扩展包。

如下图所示建立一个 maven 工程
















写一个简单的功能，如下图所示





测试通过后打包


Maven Projects





 jmeterplugin


 Lifecycle


 clean


 validate


 compile


 test



 package



 verify

 install


 site

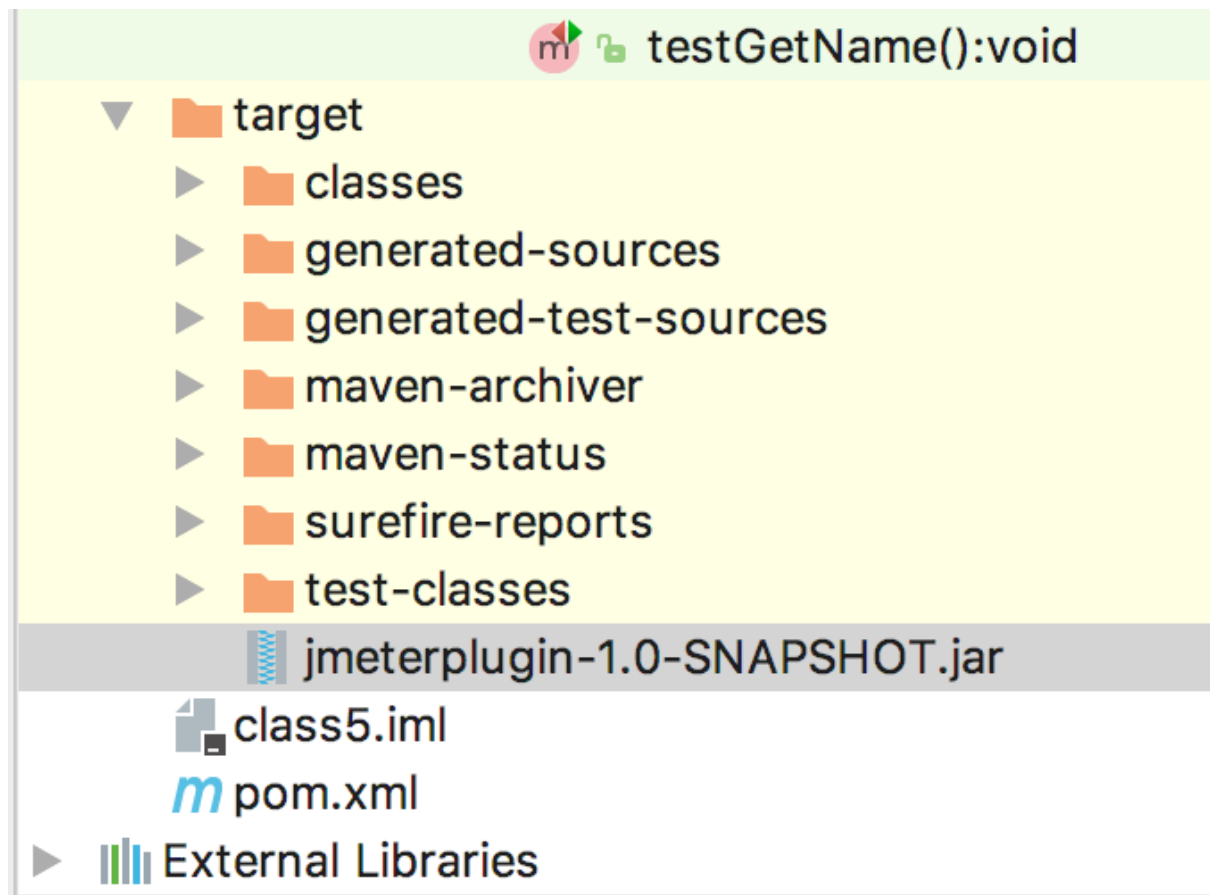
 deploy

  Plugins

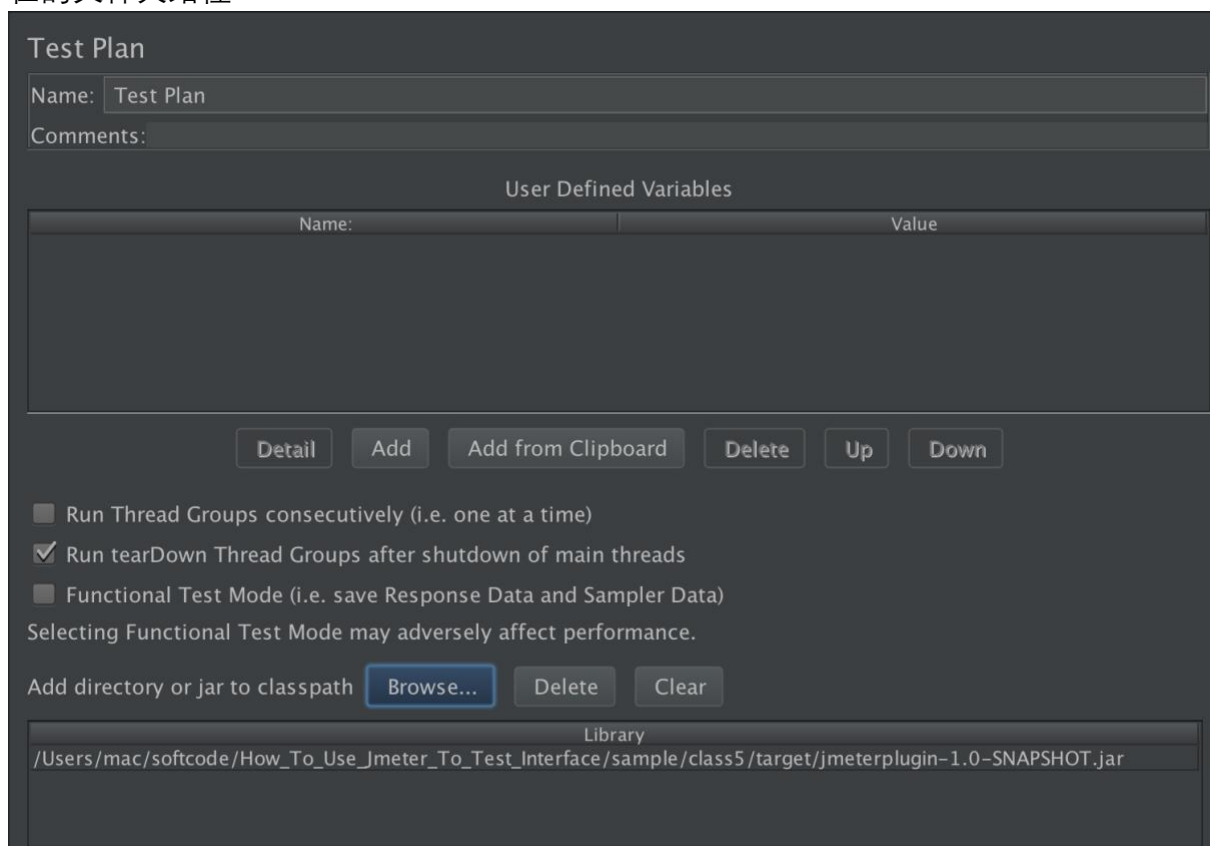
  Dependencies

Maven Projects

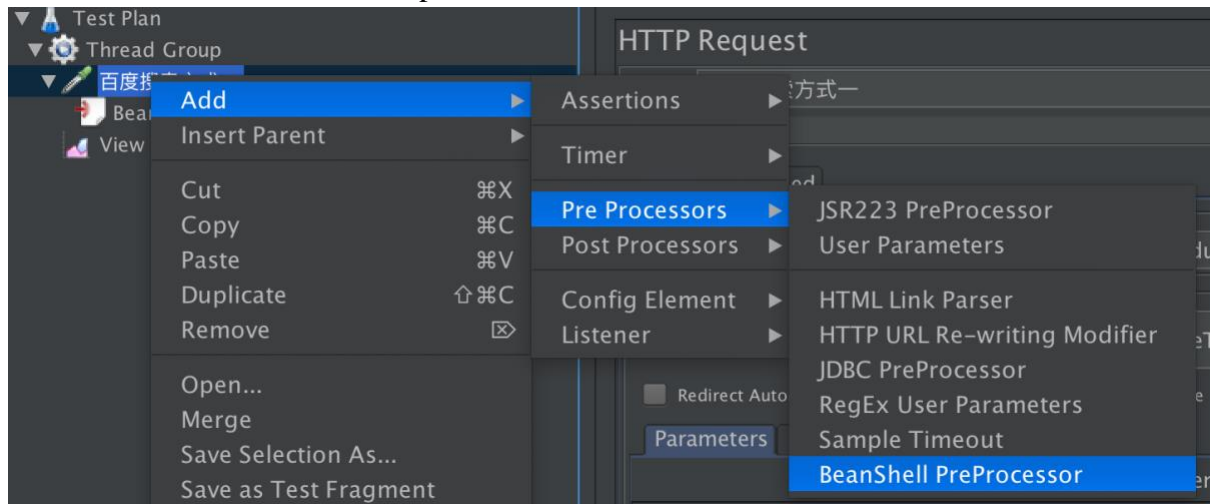
 Ant Build



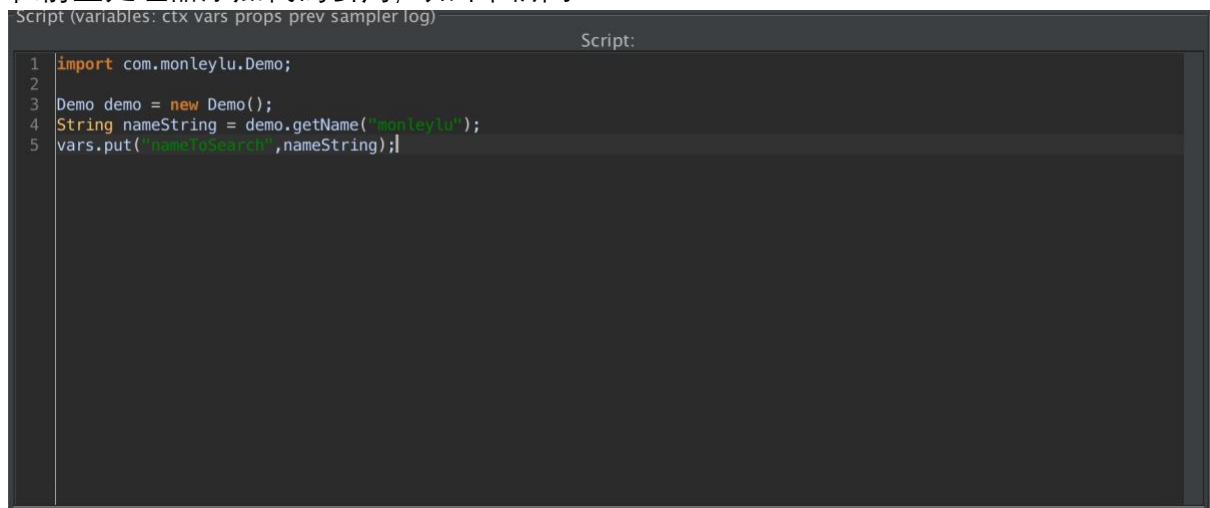
打开 jmeter，点击 TestPlan，点击最下面的 Browse 按钮，引入扩展包 jar 或者 jar 包所在的文件夹路径



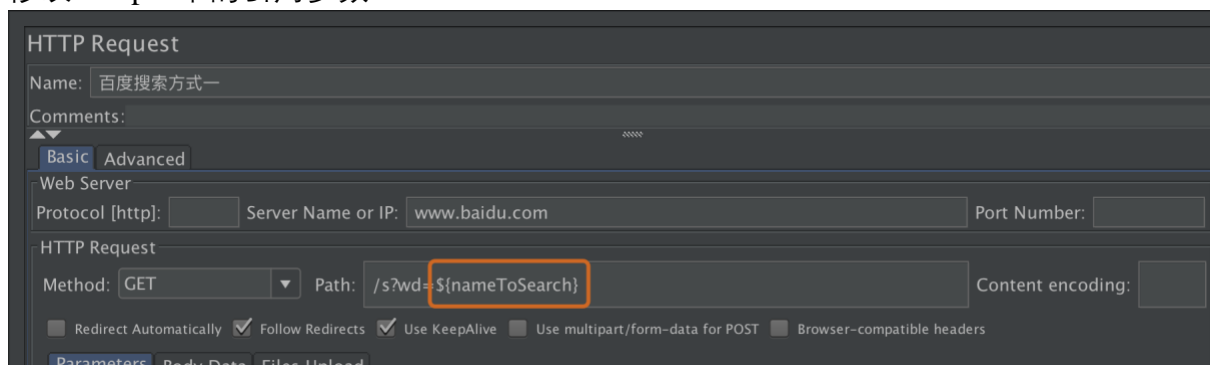
选择需要使用扩展代码的 sample，右键添加前置处理器 BeanShell PreProcessor



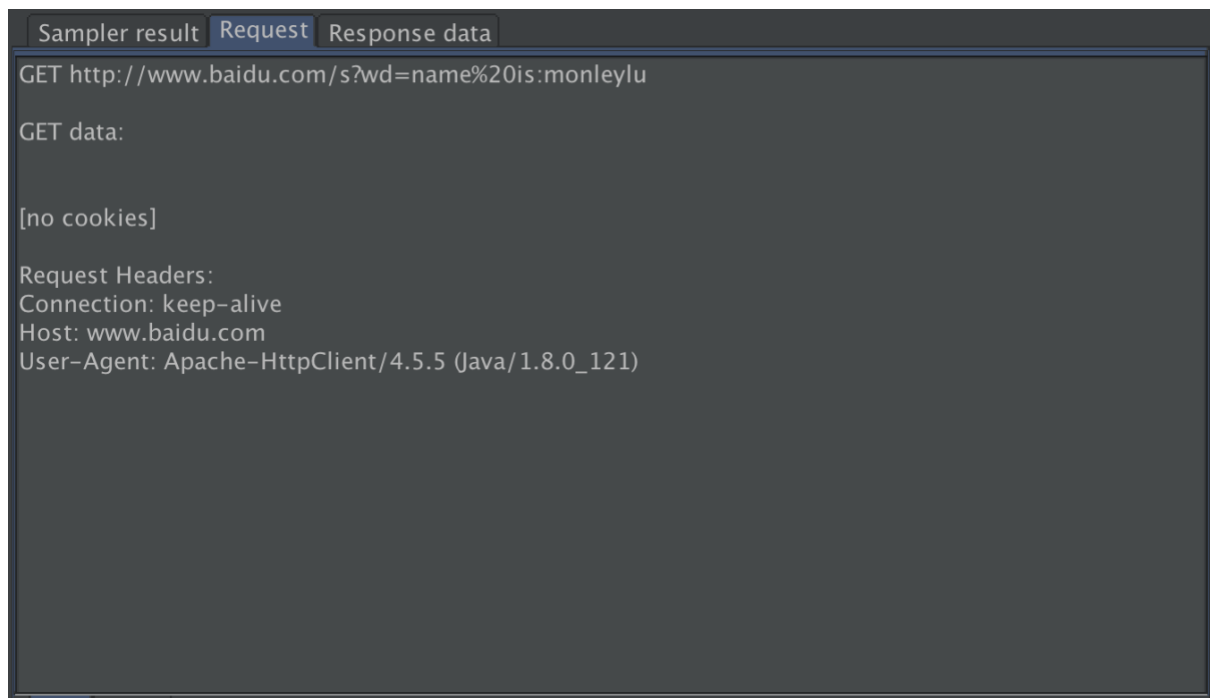
在前置处理器添加代码引用，如下图所示



修改 sample 中的引用参数



执行查看程序运行情况，如下图所示，中间多个%20是因为我们的字符串中间有个空格，传输时会对空格编码



上面介绍了通过 TestPlan 页面添加指定 jar 包到 jmeter，如果存在多个 jmeter 脚本都需要引入相同的扩展包，可以通过修改 jmeter 参数的方式，让 jmeter 在启动时就引入指定位置的扩展包，通过修改 jmeter 软件所在目录 user.properties 文件里的 user.classpath 变量指向扩展包即可。

## class6:结合 jenkins、ant 构建持续集成自动化测试

首先需要安装 ant，配置环境变量，这个自行百度配置；

安装完 ant 后，拷贝 jmeter 跟目录下 extras 目录下的 ant-jmeter-\*.jar 拷贝到 ant 安装根目录 lib 目录中。

主要的给 ant 使用构建文件如下，如下是个完整使用的文件，也可以参考 jmeter 根目录下的 extras 目录下的 build.xml 文件，也是一个完整的配置样例

```
1. <?xml version="1.0" encoding="UTF-8" standalone="no"?>
2. <!--
3. user tutorial
4. https://github.com/jfifield/ant-jmeter
5. -->
6.
7. <project name="ant-jmeter-test" default="RunAll_Prd" basedir=".">
8.   <tstamp>
9.     <format property="time" pattern="yyyyMMddhhmm" />
10.   </tstamp>
11.   <!-- 需要改成自己本地的 Jmeter 目录-->
12.   <property name="jmeter.home" value="/root/jmeter/apache-jmeter-3.2" />
13.   <!-- jmeter 生成 jtl 格式的结果报告的路径-->
14.   <property name="jmeter.result.jtl.dir" value="report_jtl" />
15.   <!-- jmeter 生成 html 格式的结果报告的路径-->
16.   <property name="jmeter.result.html.dir" value="report_html" />
17.   <!-- 测试环境变量-->
18.   <property name="AutomationEnv_PRD" value="prd" />
19.   <property name="AutomationEnv_PRE" value="pre" />
```

```

20.     <property name="AutomationEnv_SIT" value="sit" />
21.
22.     <!-- 生成的报告的前缀-->
23.     <property name="ReportName" value="TestReport" />
24.     <property name="jmeter.result.jtlName" value="${jmeter.result.jtl.dir}/${Report
Name}.jtl" />
25.     <property name="jmeter.result.htmlName" value="${jmeter.result.html.dir}/${Repo
rtName}.html" />
26.
27.     <!-- 生产环境-->
28.     <target name="RunAll_Prd">
29.         <antcall target="ToursAll">
30.             <param name="AutomationEnv" value="${AutomationEnv_PRD}"/>
31.             <param name="DataSrc" value="${DataSrc}"/>
32.         </antcall>
33.         <antcall target="report" />
34.         <antcall target="copyData" />
35.     </target>
36.
37.     <!-- 灰度环境-->
38.     <target name="RunAll_Pre">
39.         <antcall target="ToursAll">
40.             <param name="AutomationEnv" value="${AutomationEnv_PRE}"/>
41.             <param name="DataSrc" value="${DataSrc}"/>
42.         </antcall>
43.         <antcall target="report" />
44.         <antcall target="copyData" />
45.     </target>
46.
47.     <!-- 跟团度假 -->
48.     <target name="ToursAll">
49.         <echo message="workspace path is : ${basedir}" />
50.         <echo message="Automation Environment is : ${AutomationEnv}"/>
51.         <echo message="DataSrc is : ${DataSrc}"/>
52.         <taskdef name="jmeter" classname="org.programmerplanet.ant.taskdefs.jmeter.
JMeterTask" />
53.         <jmeter jmeterhome="${jmeter.home}" resultlog="${jmeter.result.jtlName}">
54.             <testplans dir="./tours/" includes="tours.jmx" />
55.             <!-- 环境参数标识 -->
56.             <property name="AutomationEnv" value="${AutomationEnv}"/>
57.             <!-- 代码构建路径 -->
58.             <property name="CodeWorkSpace" value="${basedir}"/>
59.             <!-- 接口用例数据源文件 -->
60.             <property name="DataSrc" value="${DataSrc}"/>
61.         </jmeter>
62.     </target>
63.
64.
65.     <!-- 生成的报告的前缀-->
66.     <target name="report">
67.         <xslt in="${jmeter.result.jtlName}"
68.             out="${jmeter.result.htmlName}"
69.             style="${jmeter.home}/extras/jmeter-results-detail-
report_21.xsl" />
70.         <!-- 因为上面生成报告的时候，不会将相关的图片也一起拷贝至目标目录，所以，
需要手动拷贝 -->
71.         <copy todir="${jmeter.result.html.dir}">
72.             <fileset dir="${jmeter.home}/extras">
73.                 <include name="collapse.png" />
74.                 <include name="expand.png" />

```

```

75.         </fileset>
76.     </copy>
77. </target>
78.
79. <!-- 复制执行过程中的测试数据以及自定义测试结果到报告文件夹里去 -->
80. <target name="copyData">
81.     <!-- 复制 jmeter 原始测试数据到报告里去 -->
82.     <copy todir="${jmeter.result.html.dir}/${jmeter.result.jtl.dir}"/>
83.         <fileset dir="${jmeter.result.jtl.dir}" />
84.     </copy>
85.     <!-- 复制测试数据到报告里去 -->
86.     <copy todir="${jmeter.result.html.dir}/data/">
87.         <fileset dir="data" />
88.     </copy>
89.     <!-- 复制测试自定义结果到报告里去 -->
90.     <copy todir="${jmeter.result.html.dir}/results/">
91.         <fileset dir="results" />
92.     </copy>
93. </target>
94.
95. </project>

```

如果是命令行运行 ant 构建，是使用 `ant -f build_linux.xml -DDataSrc=toursSrcId.csv` 方式，使用 jenkins 则如下图配置



The image shows the Jenkins 'Build' configuration page for an 'Invoke Ant' step. The configuration includes the following fields:

- Targets:** RunAll\_Prd
- Build File:** build\_linux.xml
- Properties:** DataSrc=toursSrcId.csv
- Java Options:** (empty)

Each field has a dropdown arrow and a help icon. At the bottom, there is an 'Add build step' button.

## class7:优化测试报告

默认我们使用的接口测试的报告和性能测试的报告，性能测试报告关注的是指标，如并发数量，QPS 等，接口测试重点关注的是每个接口的入参出参以及响应状态。所以对测试报告需要做下处理，方便查看每个接口的响应。

默认使用的接口报告模板是 jmeter 根目录 extras 下的 jmeter-results-detail-report\_21.xsl 模板，这个模板在可以满足大部分接口测试需要展示的数据需求。

优化过的模板主要修改了当接口报错时，展示响应的参数和响应，以及所有接口请求运行过程数据。

主要改动如下



```

348.         <xsl:if test="$failureCount > 0">
349.             <h3><xsl:value-of select="@lb" /><a><xsl:attribute name="name"><xsl:value-of select="@lb" /></xsl:attribute></a></h3>
350.
351.             <table style="table-layout: fixed; word-break: break-all; overflow-wrap: break-
352. word;" align="center" class="details" border="0" cellpadding="5" cellspacing="2" width="95%">
353.                 <tr valign="top">
354.                     <th>Response</th>
355.                     <th>Failure Message</th>
356.                     <xsl:if test="$showData = 'y'">
357.                         <th>Request Url</th>
358.                         <th>Request Data</th>
359.                         <th>Response Data</th>
360.                     </xsl:if>
361.                 </tr>
362.
363.                 <xsl:for-each select="/testResults/*[@lb = current()/@lb][attribute::s='false']">
364.                     <tr>
365.                         <td><xsl:value-of select="@rc | @rs" /> - <xsl:value-of select="@rm" /></td>
366.                         <td><xsl:value-of select="assertionResult/failureMessage" /></td>
367.                         <xsl:if test="$showData = 'y'">
368.                             <td><xsl:value-of select="./java.net.URL" /></td>
369.                             <td><xsl:value-of select="./queryString" /></td>
370.                             <td><xsl:value-of select="./responseData" /></td>
371.                         </xsl:if>
372.                     </tr>
373.                 </xsl:for-each>
374.
375.             </table>
376.         </xsl:if>
377.     </xsl:for-each>
378. </xsl:if>
379. <xsl:template name="detailAll">
380.
381.     <h2>DetailAll</h2>
382.     <h3>这个地方隐藏了一个将所有请求入参结果解析出来的页面</h3>
383.     <button onclick="document.getElementById('showtime').setAttribute('style','')">ShowMe</button>
384.     <div id="showtime" style="display:none;">
385.         <xsl:for-each select="/testResults/*[not(@lb = preceding::*/@lb)]">
386.
387.             <h3><xsl:value-of select="@lb" /><a><xsl:attribute name="name"><xsl:value-of select="@lb" /></xsl:attribute></a></h3>
388.
389.             <table style="table-layout: fixed; word-break: break-all; overflow-wrap: break-
390. word;" align="center" class="details" border="0" cellpadding="5" cellspacing="2" width="95%">
391.                 <tr valign="top">
392.                     <th>Response</th>
393.                     <th>Request Url</th>
394.                     <th>Request Data</th>
395.                     <th>Response Data</th>
396.                 </tr>
397.
398.                 <xsl:for-each select="/testResults/*[@lb = current()/@lb]">
399.                     <tr>
400.                         <td><xsl:value-of select="@rc | @rs" /> - <xsl:value-of select="@rm" /></td>
401.                         <td><xsl:value-of select="./java.net.URL" /></td>
402.                         <td><xsl:value-of select="./queryString" /></td>
403.                         <td style="white-space: nowrap;text-overflow: ellipsis;overflow: hidden;"><xsl:value-of select="./responseData" /></td>
404.                     </tr>
405.                 </xsl:for-each>
406.             </table>
407.
408.         </xsl:for-each>
409.     </div>
410. </xsl:template>
411.
412.
413.
414.
415.
416.
417.
418.

```

最终效果如下

## Load Test Results

Date report: date not defined

Designed for use with [JMeter](#) and [Ant](#).

## Summary

# Samples	Failures	Success Rate	Average Time	Min Time	Max Time
685	15	97.81%	719 ms	NaN	NaN

## Pages

URL	# Samples	Failures	Success Rate	Average Time	Min Time	Max Time	
InitTestData	1	0	100.00%	52 ms	NaN	NaN	+
InitTestCookie	1	0	100.00%	227 ms	NaN	NaN	+
GetTestData [5]	1	0	100.00%	39 ms	NaN	NaN	+
GetTestData [43]	1	0	100.00%	28 ms	NaN	NaN	+
GetTestData [8]	1	0	100.00%	26 ms	NaN	NaN	+
GetTestData [9]	1	0	100.00%	44 ms	NaN	NaN	+
Tours_GetInfoApp [210579581] [m.tunui.com]	1	0	100.00%	43 ms	NaN	NaN	+
Tours_GetDefaultCityInfo [210579581]	1	0	100.00%	25 ms	NaN	NaN	+
Tours_GetCalendar [210579581]	1	0	100.00%	140 ms	NaN	NaN	+
Tours_GetInfoApp [210292553] [m.tunui.com]	1	0	100.00%	20 ms	NaN	NaN	+
Tours_GetDefaultCityInfo [210292553]	1	0	100.00%	32 ms	NaN	NaN	+
Tours_GetCalendar [210292553]	1	0	100.00%	126 ms	NaN	NaN	+

## Failure Detail

## Tours\_Promotion [210581556]

Response	Failure Message	Request Data	Response Data
200 - OK	Test failed: text expected not to contain /("success":true,"errorCode":710000,"msg":"OK","data":{"promotion":null})/	{"orderId":0,"productId":210581556,"planDate":"2018-01-26","adultNum":2,"childNum":2,"freeChildNum":0,"bookCityCode":2500,"departureCityCode":1602,"backCityCode":1602,"selectedResources":{"unitPrice":0,"insurance":{"1729289017"},"extendParams":{"eyJldWlkjoiNjQ4OWE1YjgtNmVMOs00YTazlTkoNTctNTk5NDY2MTM1Y2Y1In0u003d","giftAddressId":11188462,"orderTotalPrice":0,"promotionTotalPrice":0,"gt":{"gtRes":1716990400,"journeyId":8812766},"house":{"seqNum":1,"houseId":1870272447,"houseCount":1,"adultNum":2,"hotelType":0,"ratePlanId":0,"journeyId":8812766,"hotelId":1395930228}},"sessionId":"b8917580e32faab33ff53159caf71e1a_","isOnlySelected":0}	{"success":true,"errorCode":710000,"msg":"OK","data":{"promotion":null}}

## Tours\_Promotion [210448015]

Response	Failure Message	Request Data	Response Data
200 - OK	Test failed: text expected not to contain /("success":true,"errorCode":710000,"msg":"OK","data":{"promotion":null})/	{"orderId":0,"productId":210448015,"planDate":"2017-12-16","adultNum":2,"childNum":2,"freeChildNum":2,"bookCityCode":1602,"departureCityCode":1602,"backCityCode":1602,"selectedResources":{"unitPrice":0,"insurance":{"1729287894"},"extendParams":{"eyJldWlkjoiZTQ0ZjlmYWU0WZiZi00ZjVhLW1lNWU0NTcxZDhmMTg1NmZln0u003d","giftAddressId":11188462,"orderTotalPrice":0,"promotionTotalPrice":0,"gt":{"gtRes":1454214976,"journeyId":4547151}}},"sessionId":"b8917580e32faab33ff53159caf71e1a_","isOnlySelected":0}	{"success":true,"errorCode":710000,"msg":"OK","data":{"promotion":null}}

## DetailAll

这个地方隐藏了一个将所有请求入参结果解析出来的页面

[ShowMe](#)

## InitTestData

Response	Request Url	Request Data	Response Data
200 - OK			

## InitTestCookie

Response	Request Url	Request Data	Response Data
200 - OK	http://occamrazor.dev.tunui.org/automation/getAllTestCookies?ID=10		{"total":1,"rows":[{"ID":10,"Name":"AndroidAP...

## GetTestData [5]

Response	Request Url	Request Data	Response Data
200 - OK	http://occamrazor.dev.tunui.org/automation/getTestcaseWithRelateProduct?ID=5		{"testCase":{"ID":5,"testCaseName":"M9boss...

## GetTestData [43]

## class8:录制手机 APP 接口

这一步和平常用 charles 或者 fiddler 抓无线端报文内容类似，如果要抓 https 报文，需要安装指定的 ca 证书，在根目录 bin 下 ApacheJMeterTemporaryRootCA.crt,电脑和手机端都要安装，之后就可以正常录制 https 流量，网站和 app 的 https 请求都可以录制