Fidder 和 jmeter 使用手册

Fiddler:

分为六大模块：一.菜单栏、二.工具栏、三.数据报文规划区、四.功能页 五.命令框、六.状态栏

一. 菜单栏：

File:

Capture Traffic：默认勾选，勾选此项才可抓包，与点击左下角状态栏的 Capture 效果一样

New Viewer：开启一个新的 fiddler 的 viewer

Load Archive…：用于重新加载之前捕获到的 SAZ 文件格式保存的流量。

Recent Archives：查看最近之前捕获到的 SAZ 文件格式保存的流量

Save：保存

Import Sessions…：从目标文件夹及其子文件夹加载所有 SAZ 文件。缓存和重用密码。 支持导入从其他工具获得的流量

Export sessions：支持用 fiddler 把捕捉到的 sesison 用多种方式保存。CURL 脚本由 CURL 回放;Capture Traffic：默认勾选，勾选此项才可抓包，与点击左下角状态栏的 Capture 效果一样

Exit：退出 Fiddler

Edit:

Copy：用来拷贝请求的相关信息。

Remove：主要是用来移除左侧边栏中的 session

Select All：全选左侧边栏中 session

Undelete：恢复之前删除的 session

Paste as Sessions：把剪切板里的 sesisongs 复制到 web sessions 中，把以前的会话粘贴回来

Mark：自定义不同 session 的显示颜色

Unlock for Editing:把锁定的 session 进行解锁,可以进行编辑,默认情况下是不可进行编辑的,默认可以看到选定的 session 前是“锁”的图形 ，点击此按钮后变成可编辑按钮 。

Find Sessions…：搜索 session

Rules:

Hide Image Requests：可以隐藏图片请求，让图片类的 session 不在 session 框中显示出来

Hide CONNECTs：可以隐藏 CONNECT 方法的请求，让这类 session 不在 session 框中显示出来

Automatic Breakpoints：自动断点，控制是否自动在 Before Request 或 After Request 处断点，来修改请求或响应的内容

Customize Rules…：来打开 fiddler script 工具，调取脚本操作，多用于网络修改，其他自定义时使用

Require proxy authentication：若选中此项，则所有未提交 Require proxy authentication 的请求头的请求会返回 HTTP/407 响应，要求客户安装证书

Apply GZIP Encoding：请求 GZIP 编码，若选中此项，则只要请求包含了 gzip 标识的 Accept-Encoding 请求头就会对除了图片以外的所有相应使用 GZIP HTTP 进行压缩

Remove All Encoding：若选中此项，会删除所有请求相应的 http 内容编码和传输编码

Hide 304s：在 session 框中隐藏所有的 304 的 session

Request Japanese Content：把所有的 Accept-Encoding 请求头设置替换成 ja 标示，标示客户端希望以日语的形式发送

Automatically Authenticate：自动进行身份验证 鉴权

User-Agents：选择相应的用户代理模式，默认是选择 disabled。

Performance：此项提供影响 web 性能的简单选项。

Tools:

Options…：打开 Options 窗口，是 fiddler 抓包的一些设置项，包括对抓取接口是 Http 还是 Https 的设置，获取证书，设置代理端口号等功能

WinINET Options…：打开 IE 浏览器的 options 进行设置

Clear WinINET Catch：清空 IE 和其他应用中所使用的 WinINET 的缓存文件

Clear WinINET Cookies：清空 IE 和其他应用中所使用的 WinINET 的 Cookies 文件

TextWizard：文本向导工具，是一个非常好用的可以轻松将 text 文本 encode 和 decode 的小工具

Reset Script：重置脚本

Sandbox：fiddler sanbox 官方文档

View IE Cache：查看 IE 浏览器缓存文件夹

New Session Clipboard…：打开一个新的 session 剪贴板，可以把侧边栏中的 session 拖到这个剪贴板中具体来查看

HOSTS：主机重定向工具。若在其中勾选 Enable 框，然后在下面加入 host 配置，点击保存之后，这个配置并不会修改到本地 hosts 中，取消勾选就会失效，若点击 Import Windows Hosts File 将会导入本地的 host 文件内容。

//view：（statistics 后面详细看）

Show Toolbar：显示工具栏，默认是勾选的

Default Layout：默认 layout，session 在左，请求和响应在右边的上下处

Stacked Layout：session 在上，请求在下方

Wide layout：session 在上，请求和响应在下方的左右处

Tabs：打开标签页面，其中有三个标签可以打开，分别是 Preferences（fiddler 偏好属性），AutoSave（fiddler 自动保存的设置），APITest（api 的测试）

Statistics：查看一个请求的统计数据

Inspectors：嗅探，用来查看会话的内容，上面是请求，下面是响应

Composer：设计构造，在 Composer 中进行请求的修改，可以把 session 框中的数据先清除，然后点击 Composer 中的 Excute 按钮来发送请求，请求出现在 session 框中

Minimize To Tray：最小化托盘

Squish Session List：挤压 session 框

AutoScroll Session List：自动滚动会话列表，默认是勾选此项的，勾选此项后，session 框中的每出现新的 session，session 框中就会不断向下滚动，若不勾选此项，就很方便具体某一个 session 的定位，即使出现了新的 session 也不会自动向下滚动

Refresh：刷新功能，按 F5 刷新

Help：

Help：进入 fiddler 的帮助的网页中

Get Fiddler Book…：fiddler book 的网页

Discussions：fiddler 的讨论网页，这个需要魔法上网

Http Preferences：进入 http preferences 相关网站

Troubleshoot…：会捕获所有请求，对于哪些被过滤的请求用删除线表示出来并给出原因，使用时候会打开一个网页

Get Priority Support…：打开网页购买 fiddler 的优先级服务

Check for Updates…：检查软件更新情况

Send Feedback…：意见反馈

About：当前 fiddler 的相关信息

二、工具栏：

WinConfig windows：

使用了一种叫做“AppContainer”的隔离技术，使得一些流量无法正常捕获，在 fiddler 中点击 WinConfig 按钮可以解除这个诅咒，这个与菜单栏 Tools→Win8 Loopback Exemptions 功能是一致的

云朵标志图：

此按钮来给选定的 session 添加注释

Replay： 重发按钮，选定请求重发按钮

（✕）移除按钮，其中有 Remove all 移除所有，Images，CONNECTs，Non-200s，Non-Browser，Complete & Unmarked，Duplicate response bodies，这些都是移除 session 中的这些状态的选项

Go： 重跑 sessions，依据断点暂停

Stream：流模式是一种实时通信模式，请求之后实时的返回，更接近浏览器真实行为，但 fiddler 默认是缓冲模式而不是流模式

Decode： 解码，这里可以将 session 中乱码进行解码方便查看

Keep All sessions All sessions：这里可以保持 session 框中存在多少个 sessions

Any Process：（不了解咋用） 点击此按钮并且拖动到你想要捕获的浏览器从而实现只捕获某个浏览器的请求

Find ：查询

Save： 保存按钮，保存所有的 session 成 SAZ 文件

计时功能： 手动点击运行，手动点击暂停终止

Browser： 打开浏览器来查看响应数据

Clear Cache ：清除 WinINET 的缓存，按住 CTRL 键点击可以清除已经存在的 cookies

TextWizard ：此工具可以将某一编码过的或者未编码过的字串拿到此处解码和编码，在菜单栏中的 Tools 中也有此项可以打开

Tearoff： 此功能用来将右边栏里的请求和响应部分给单独拆成一个新窗口，方便视察

MSDN Search… ：在网页版的微软开发中去搜索

问号标志：帮助

online：鼠标悬停显示本机的一些 ip 信息

“×”用来关闭工具栏的按钮，在 View 中选择第一个可以打开工具栏

三、底端状态栏

Capturing：此处与菜单栏中 File→Capture Traffic 效果是一致的，默认底端状态栏此处是有 Caturing，有它才表示 fiddler 捕获请求

All Processes：（不会） 这里有 All Processes，Web Browsers，Non-Browser，Hide All 几个选项，这个几个选项顾名思义，但要注意的是这些不是筛选当前 session 框中的 session，而是选中需要筛选的状态之后，后面的请求会按照此状态来筛选

数字/数字： 第一个数字表示这一个请求，第二个数字表示 session 框中共有多少 session

数字后的白色框：此处显示请求的 url 网址

四、Request 栏：

在 request 栏中有 9 个大的标签页，分别是 Inspector，AutoResponder，Composer，Fiddler Orchestra Beta，Fiddler Script，Log，Filters，Timeline，Statistics

这里是查看某个 session 的请求和响应，响应的话专门置为一栏讲解，请求的话又可分为 10 个小的标签页。并且右键点击这 10 个标签页可以查看 Inspector 的属性还有诸如 copy as image 和隐藏标签页的功能

headers

：这里是请求头中的信息，包括 cache，cookies 等信息，点击右边黄色的 Raw 可以以新窗口的形式来显示原生头信息，而 Header Definitions 可以查看 fiddler 官方的头信息的网页版帮助文档，可能需要魔法上网

TextView 方式显示传送过去的请求体数据

SyntaxView 方式显示传送过去的请求体数据

webforms：网页表单方式显示传送过去的请求体数据

hexview：十六进制视图的方式显示传过去的数据

auth：显示请求中的身份认证信息

cookies：显示该请求的 cookies 信息

raw（具体看）：显示该原生的请求体

json ：显示请求

xml ：显示请求

autoresponder：重定向，本机代替服务器发送响应

Composer 和 Inspector 都可以篡改数据，Inspector 是篡改输入的数据，但是 Composer 却可以篡改 Cookies 中的数据，并通过 Execute 发送重新篡改后的请求，界面上的控件比较简单

composer

Fiddler Orchestra Beta：这个功能应该还是 beta 测试阶段，暂未开放

FiddlerScriptfiddler： 在 Web 前端开发时候经常使用，用户再修改请求头信息时候经常需要设置断点，但是设置后会在断点处停下，之后点击重启才行

log：查看 fiddler 的 event log 信息，不同请求的 log 信息应该是一致的，每当更新一次页面，event log 会自动刷新一次，若将上方的 any process 拖动到指定浏览器后，fiddler 会单独记录该浏览器页面的通信信息

Filters：这个可以用来过滤 session 中的请求

Timeline：可以查看请求响应的时间轴

Statistics（具体看）：里头包含该 session 请求的统计数据，包括请求次数，请求与响应字节数，信息头和体各有多少字节，以及连接时间点，响应信息类型，最后，下面有个全球性能估计的数据。

五、response 栏

这里包含 13 个小的标签页面，分别是 Transformer，Header，TextView，SyntaxView，ImageView，HexView，WebView，Auth，Caching，Cookies，Raw，JSON，XML

Transformer：这里显示了响应体的字节数，这里头的 Chunked Transfer-Encoding 和 HTTP Compression 是分块传输编码和 HTTP 压缩技术，这就需要 http 的知识了

Headers：这里可以看到响应头部分，包括 http 的协议，返回状态码，连接情况等

重定向，本机代替服务器发送响应响应体信息

SyntaxView：这里返回 SyntaxView 形式的响应体信息

ImageView：如果是返回图片的话这里将有显示，并且左边会显示图片的信息

HexView：这里会显示响应体的十六进制信息

WebView：这里会用网页的形式来显示响应体的信息

Auth：显示身份验证

Caching：显示缓存信息

Cookies：显示 Cookies 信息

Raw：显示原生的响应信息

JSON：响应体中的数据 json 显示

XML：响应体中的数据 XML 显示

六、命令行

查找相关 session

七、列表栏

--session id（响应状态）

Result --响应状态码

Protocol --Session 使用的协议

Host --接受请求的服务器主机名和端口号

URL --请求 URL 的路径

Body --相应题中包含的字节数

Caching --响应头中 Expires 和 Cache-Control 字段的值

Content-Type -- 响应 Content-type 头

Process --数据流对应的本地进程

Custom --FiddlerScript 所设置的 ui-CustomColumn 标志位的值

Comments --注释信息

八、额外知识：

1、状态码（表示网页服务器超文本传输协议响应状态的 3 位数字代码）：

分类：

1XX(100~101)：信息提示，表示请求已被成功接收，继续处理

2XX(200~206)：成功，表示请求已被成功接收、理解、处理

3XX(300~305)：重定向，要完成要求，需进行下一步处理

4XX(400~415)：客户端错误，请求有语法错误或无法实现

5XX(500~505)：服务器错误，服务器未能实现合法的请求

常见状态码：

200：OK，服务器成功处理请求

301/302：重定向，请求的 URL 被移走

304：未修改，客户的缓存资源是最新的，需要客户端使用缓存

404：未找到资源

401：禁止访问

501：服务器遇到一个错误，使其无法对请求提供服务

204：

返回的 HTTP 响应中只有一些 header 和一个状态行，没有实体的主题内容（没有响应 body）

作用：

在不获取资源的情况下了解资源的情况

通过查看 HTTP 响应中的状态码看某个对象是否存在

通过查看 header 测试资源是否被修改

206：

206 状态码表示已经处理了部分的 GET 请求（有发送 GET 方法的 HTTP 请求，web 服务器才会返回 206）

应用场景：

迅雷、HTTP 下载工具都使用 206 状态码实现断点续传

将大文档分为多个下载段同时下载，如在线看视频可以边看边下载

301：

服务器返回 301 时表示请求的网页已经永久性转移到了另一个地址

应用场景：

网站更换域名

302：

当访问一个 URL 时，服务器要我们访问另外一个资源，这时浏览器会继续发送一个 HTTP，请求访问新的资源

301 与 302 的区别：

301 表示旧地址的资源已经被永久地移除了，资源已经无法再访问，搜索引擎会把权重算到新的地址

302 表示旧地址还在，仍然可以访问，重定向只是临时地从旧地址跳转到新地址，搜索引擎会把权重算到旧地址上

304：

代表上次的文档已经被缓存，还能够继续使用

如果不想使用本地缓存，Crtl+F5 强制刷新页面

400：

状态码 40 表示请求的语法错误，发送的 HTTP 请求中数据有错，无法被服务器所理解

应用场景：

查询快递，参数不对，服务器会返回 400

401：

状态码 401 指未授权的错误，有些网页采用 HTTP 基本认证，需要在 HTTP 请求中带上认证首部（authorization header），否则会返回 401

403：

表示 Web 客户端发送的请求被 Web 服务器拒绝了

404：

当输入一个 URL，此 URL 的域名正确，但资源不存在，服务器会返回 404，告诉浏览器资源不在了

此 404 页面是可以自定义的

500：

代表服务器内部错误，错误原因过多，如代码错误、数据库....

503：

表示服务器暂时不可以，此状态是临时的，并且在一段时间后会恢复

200：请求成功

302：资源临时移动，客户端继续使用原有的 URL

206：服务器成功处理了部分 GET 请求

405：客户端请求的方法被禁止

2、Get、post 请求：

区别：

（1）、GET 请求的数据会暴露在地址栏中，而 POST 请求则不会；

（2）、Get 请求传输数据量小，Post 请求传输数据量大；

GET　请求获取 Request-URI 所标识的资源

POST　在 Request-URI 所标识的资源后附加新的数据

HEAD　请求获取由 Request-URI 所标识的资源的响应消息报头

PUT　请求服务器存储一个资源，并用 Request-URI 作为其标识

DELETE　请求服务器删除 Request-URI 所标识的资源

TRACE　请求服务器回送收到的请求信息，主要用于测试或诊断

CONNECT 保留将来使用

OPTIONS 请求查询服务器的性能，或者查询与资源相关的选项和需求

3、Raw、statistics（内容）：

Get:请求类型

Host:主机地址

Connection: keep-alive：连接生命周期

User-Agent：用户代理

Accept：接受

Sec-Fetch-Site: Sec 获取站点：

cross-site：跨站点

Sec-Fetch-Mode：秒取数模式

no-cors：非 cors

Sec-Fetch-Dest：

Script：脚本

Referer：推荐人

Accept-Encoding: gzip, deflate, br: 接受编码

Accept-Language: 接受语言：

Cookie: 浏览器用来保存少量数据的一种机制。

ACTUAL PERFORMANCE：实际业绩

ClientConnected:客户连接：

ClientBeginRequest:客户开始请求：

GotRequestHeaders:

Determine Gateway：确定网关

DNS Lookup：域名解析

HTTPS Handshake:HTTPS 握手：

ServerConnected:服务器已连接：

ClientDoneRequest:客户需求：

ClientDoneResponse:客户响应

Jmeter

测试计划

测试计划：用来描述一个性能测试，所有内容都是基于这个计划的。

线程（用户）

一般常用线程组（Thread Group）：可以理解成为虚拟用户组：

线程数：虚拟用户数。一个虚拟用户占用一个进程或线程。设置多少虚拟用户数在这里也就是设置多少个线程数。

准备时长： 设置的虚拟用户数需要多长时间全部启动。

循环次数：每个线程发送请求的次数。

setup thread group：可用于执行预测试操作。这些线程的行为完全像一个正常的线程组元件。类似 Loadrunner 中的 init

teardown thread group：可用于执行测试后动作。这些线程的行为完全像一个正常的线程组元件。类似 Loaderrunner 中的 end

jmeter 操作术语

控制器

JMeter 有两种类型的控制器：取样器（sample）和逻辑控制器（Logic Controller），用这些原件来驱动处理一个测试。

采样器（Samplers）：采样器是 JMeter 测试脚本的基础单元，用户可以用它来向服务器发出一个特定的请求，采样器会在超时前等待服务器的响应。

逻辑控制器（Logic Controllers）：用户通过逻辑控制器来控制 JMter 测试脚本的执行顺序，以便测试能够按照用户期望的顺序和逻辑执行。

监听器（Listeners）：监听器被用来手机测试结果信息，并以用户指定的方式加以展示。它是用来对测试结果数据进行处理和可视化展示的一系列元件。 图行结果、查看结果树、聚合报告。都是我们经常用到的元件。

聚合报告解析

Listeners-Aggregate Report

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

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

配置元件（Configuration Elements）：配置元件被用来设置一些 JMeter 测试脚本公用的信息。

断言（Assertions）：断言被用来验证服务器实际返回的信息与用户期望的情况是否相符。

定时器（Timers）：定时器被用来保存 Jmeter 测试脚本与时间相关的一些信息，例如思考时间（Think Time）。

测试片段（Test Fragment）

测试片段元素是控制器上的一个种特殊的线程组，它在测试树上与线程组处于一个层级。它与线程组有所不同，因为它不被执行，除非它是一个模块控制器或者是被控制器所引用时才会被执行。

前置处理器（Per Processors）

用于在实际的请求发出之前对即将发出的请求进行特殊处理。例如，HTTP URL 重写修复符则可以实现 URL 重写，当 RUL 中有 sessionID 一类的 session 信息时，可以通过该处理器填充发出请求的实际的 sessionID 。

后置处理器（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 测试工具中的关联概念）。例如，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。

Jmeter 工具和其他性能工具在原理上完全一致，工具包含 4 个部分：

（1）负载发生器：用于产生负载，通常以多线程或是多进程的方式模拟用户行为。

（2）用户运行器：通常是一个脚本运行引擎，用户运行器附加在线程或进程上，根据脚本要求模拟指定的用户行为。

（3）资源生成器：用于生成测试过程中服务器、负载机的资源数据。

（4）报表生成器：根据测试中霍地的数据生成报表，提供可视化的数据显示方式。

使用方法：

添加了线程组之后，就开始添加请求了，右键线程组， 添加 -> sampler -> http 请求

一个 http 请求就是一个接口，开始配置各个参数：

端口号我们设置为 80，这样一个 get 请求就设置好了。

下面在来一个 post 方式的请求：

post 方式的提交跟 get 不同的是，有 request body，在 Body Data 里设置 json 格式的数据，作为提交到服务器的参数。

三、那么我们提交了请求，现在就应该看看结果了，这时候在所在请求上，右键 -> 添加 -> 监听器 -> 观察结果树

在实际工作中，我们都是一个请求对应一个结果树。

下面有三个选项卡，通过查看请求跟响应数据可以看到我们想要的数据。返回的请求结果。

四、经常遇到入参变动，这时候可以考虑，将其设置为变量，jmeter 提供了三种方式设置变量

1、用户自定义变量 ：

在所在的请求上右键，添加、配置元件、用户自定义变量。

定义好了，如何引用呢？ ${domain} 通过这个就可以了。

2、用 CVS 文件读取：

在所在的请求上右键，添加、配置元件、CVS Data Set Config。

五、有一种情况就是，服务器域名、端口、协议、编码都是相同的，那么我们没有必要每一个请求都添加一遍，这个时候可以使用，http 请求默认值：

右键测试计划、添加、http 默认请求。

六、我们在接口请求的时候经常是处于登录状态下的，没有 cookie 或者 token，无法请求一些私密性的接口：

所在请求上右键、添加、配置元件、cookie 管理器，在里面设置三个参数，key value domain。

七、Jmeter 断言提供了几种方法，我喜欢 bean shell 函数，比较灵活：

所在请求上右键、添加、断言、bean shell 断言，

String response = "";

// Str 就是接口的一个字段， response 就是接口内容

String Str = "\"externalSignInProviders\":[{\"provider\":\"WeChat\",\"providerDisplayName\":\" 微 信 \"},{\"provider\":\"QQ\",\"providerDisplayName\":\"QQ\"},{\"provider\":\"GitHub\",\"providerDisplayName\":\"GitHub\"}]";

```
response = prev.getResponseDataAsString();
if(response == ""){
// 设置断言失败为真
Failure = true;
FailureMessage = "系统无响应，获取不到响应数据！";
log.info(FailureMessage);
}
else if(response.contains(Str) == false){
// 设置断言失败为真
Failure = true;
String Msg = "\n 系统返回响应结果与期望结果不一致！请排查是性能问题，还是程序代码问题";
FailureMessage = Msg + "\n" + "期望结果:\n" + Str + "\n" + "响应内容: \n" + response +"\n";
log.info(FailureMessage);
}else{
// 设置断言失败为假
Failure = false;
log.info("期望结果:\n" + Str + "\n" + "响应内容: \n" + response +"\n"+"一致通过");
}
```

八、有的时候，上一个请求返回的接口中提取指定的值，用来做下一次请求的参数，就叫做接口关联，我喜欢使用 json 提取。还可以使用正则表达式提取。

所在请求上右键、添加、后置处理器、json extractor,

先看看接口结果：

{"publicKey":"MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQCpOwHYbg/NOPO3nzMD3dndwSOMccuMeXCHgVlGOoYyFwLdS24Im2e7YyhB0wrUsyYfO/nhzCzBK8ZC9eCWqdOaHbdgOQT6CuFQBMjbyGYvlVYU2ZP7kG9Ft6YV6oc9ambuO7nPZh+bvXHOzDKfiO2prknrScAKC0XhadTHT3AlOQIDAQAB","externalSignInProviders":[{"provider":"WeChat","providerDisplayName":"微信"},{"provider":"QQ","providerDisplayName":"QQ"},{"provider":"GitHub","providerDisplayName":"GitHub"}],"externalSignedIn":null}

根据接口的结构，这样添加就可以了。

*注意请求的顺序，必须是提供参数的请求在先，接收参数的请求靠后。

File:	edit:	search:	run:
新建	添加子进程	搜索	开始，别停顿
模板	应用命名策略	复位	远程启动(remote start)
打开最近的	切		远程停止(remot stop)
合并	复制		关闭(shutdown)
将测试计划保存为	合并		远程出口(remote exit)
将选择保存为	将节点保存为图像（save node as image）		
保存为测试碎片	保存屏幕为图像（save screen as image）		
重新开始	使能（enable）		
退出	肘节（toggle）		

Options:	tools:
外观和感觉(look and feel)	创建一个帮助垃圾堆(creat a help dump)
日志查看器(log viewer)	创建一个线程转储(creat a thread dump)

对数杆(log lever)

Ssl 管理程序(ssl manager)

崩溃(collapse)

全部展开(expand all)

全部放大(zoom all)

缩小(缩小)

运行前自动保存(save automatically before run)

Help:

这个节点是什么(what is this node)

启用调试(enable debug)

停止调试(disable debug)

有用的链接(useful links)

关于 apache jmeter(about apache jmeter)

函数助手对话框(function helper dialog)

生成 html 报告(generate html report)

编译 jsr223 测试元素(compile jsr223 test element)

出口交易报告(export transactions for report)

生成示意图视图(export transactions for report)

从 curl 导入(import from curl)