**Loadrunnr相关参考文章：**

**LoadRunner响应时间与用户体验时间不一致问题的深入分析**

**https://blog.csdn.net/hualusiyu/article/details/65937808**

**loadrunner事务响应时间的组成**

**https://blog.csdn.net/he\_jian1/article/details/41749623**

**LoadRunner监控mysql利器－SiteScope**

**https://www.jianshu.com/p/fce30e333578**

**loadrunner场景中按scenario和group执行的区别**

group：多个脚本之间按照独立设置模式跑，各个脚本可以单独设置虚拟用户、运行时间等

scenario：多个脚本之间按照相同的模式跑，将总的虚拟用户数按照一定的比例分配给各个脚本

[**LoadRunner场景的一种设置方法**](http://www.blogjava.net/xingcyx/archive/2007/07/18/130962.html)

在设置LoadRunner测试场景时，很多时候都需要设置用户逐渐递增的场景。可能大家通常的做法都是逐步增加用户数，并且分成多次去运行场景，比如第一次运行用100用户，第二次用200用户，第三次用300用户等等。  
其实，还可以有另外一种方法，就是针对同一个脚本设置多个组，使用组策略(点击Edit Schedule按钮，选择Schedule by Group)，第一个组为初始用户数，如100，第二个组以50或100递增。每个用户场景都持续稳定地运行一段时间，这样可以把几种用户场景下的测试在一次场景的运行过程中就一次性执行完毕，而且还有一个好处是可以把多个用户场景下的性能结果都整合到同一个结果中，也方便对比、分析。

是选择HTML还是URL录制，有以下参考原则： （1）基于浏览器的应用程序推荐使用HTML-based script。 （2）不是基于浏览器的应用程序推荐使用URL-based script。 （3）如果基于浏览器的应用程序中包含了JavaScript并且该脚本向服务器产生了请求，比如DataGrid的分页按钮等，也要使用URL-based script方式录制。 （4）基于浏览器的应用程序中使用了HTTPS安全协议，使用URL-based script方式录制。

事务（Transaction）是这样一个点，我们为了衡量某个action的性能，需要在action的开始和结束位置插入这样一个范围，这就定义了一个transaction，LoadRunner 运行到该事务的开始点时，LoadRunner 就会开始计时，直到运行到该事务的结束点，计时结束。这个事务的运行时间在结果中会有反映。

插入事务操作可以在录制过程中进行，也可以在录制结束后进行。LoadRunner 可以在脚本中插入不限数量的事务。

举个例子：比如一个单据,把从登录到保存成功退出整个作为一个脚本,对于需要关注的保存时间,定义为单独的事务,以取得响应时间，事务脚本函数如下：



脚本里面有2个函数，解释一下：1.几个函数的解释：

1） int web\_url (const char \*Name, const char \* url, <List of Attributes>,

[EXTRARES, <List of Resource Attributes>,] LAST );

这个函数load 指定的web页面 .

\*Name：页面的name；

􀁺url：页面的url，Resource：指示the URL是否是一个资源。0，不是，1，是。

􀁺RecContentType：录制脚本过程中，Header响应的类型，e.g. text/html, application/x- javascript

􀁺Referer  参考web页的the URL

􀁺Snapshot - snapshot 文件名(扩展名inf), correlation的时候要的。

􀁺Mode  录制的级别: HTML or HTTP

􀁺Last- 属性列表的结束标志。

web\_url("Folder.jsp\_4",

"URL=http://172.17.16.5/xpc71/jsp/com/folder/Folder.jsp",

"Resource=0",

"RecContentType=text/html",

"Referer=http://172.17.16.5/xpc71/LoginAction.do",

"Snapshot=t10.inf",

"Mode=HTML",

LAST);

lr\_log\_message("abc");

输出字符到下方窗口，供调试用

[**实现LoadRunner多个场景的顺序执行**](http://www.cnblogs.com/pent/archive/2007/04/16/715587.html)

注：以下内容部分总结自51testing论坛。

**应用场景**  
假设有3个不同的测试场景，分别为并发登录、核心业务、可靠性测试，3个场景有先后执行顺序。由于白天测试机器另有用处，只能在晚上进行性能测试，这时我们的期望是能否把测试场景都设定好之后晚上自动运行，第二天我们回来看测试结果呢？  
答案是肯定的，可以有两种方式实现。

**第一种，相对简单**充分利用LR Controller里面Group的功能。  
新建一个场景把3个脚本都添加进来，在Edit Schedule中选择“Schedule by Group”的方式，在StartTime中设置3个脚本的运行顺序为“Start when Group xxx finished”，并在“Scenario Start Time”中设定场景在晚上的运行启动时间。设定完定时执行场景后，点击StartScenario按钮，会出现一个倒计时窗口，这样在固定的某个时间上，测试场景中的3个脚本将乖乖的按照设定的先后顺序进行测试。注意，如果没有点击StartScenario按钮激活测试，是不会真正进行测试的。（感谢Athenst朋友的提醒，^\_^）

**第二种，比较灵活**我们把应用场景稍微扩展一下，假设其中1、3场景只有一个测试脚本，而核心业务场景由数据录入、数据查询、数据上报3个脚本组成，同样的，3个场景仍需按顺序进行测试。这时如果采用第一种方式，由于第2个场景有3个脚本，所以第三个脚本的启动时间就是一个问题了。由于Controller中每个脚本都对应一个Group，而且GroupName不能重复，这时第三个场景的StartTime中“Start when group finished”则只能是选择第二个场景中的某个Group，而并非是第二个场景的3个脚本都完成之后再进行，无法达到我们的初衷。  
这时，可以通过命令行的方式来进行。  
首先创建并设置好3个测试场景，再创建一个一个批处理程序按先后顺序调用这3个场景进行测试，最后通过Windows的定时任务设定批处理的执行时间。  
批处理示例如下：  
cls  
SET M\_ROOT="D:\Program Files\MI\Mercury LoadRunner\bin\"  
%M\_ROOT%\wlrun.exe -TestPath "D:\Program Files\MI\Mercury LoadRunner\scenario\Test\TestScen\_1.lrs" -Run  
%M\_ROOT%\wlrun.exe -TestPath "D:\Program Files\MI\Mercury LoadRunner\scenario\Test\TestScen\_2.lrs" -Run  
%M\_ROOT%\wlrun.exe -TestPath "D:\Program Files\MI\Mercury LoadRunner\scenario\Test\TestScen\_3.lrs" -Run  
这种方式比较灵活，但需要注意在Result　Settings中设置“Automatically create a results directory for each scenario execution”，以免后面的测试结果覆盖了前面的。

另外补充一下，如果想对某个脚本进行50、100、150...等用户数递增的测试，也可以用以上方法实现，但需要注意的是将事务名称区分开以便进行分析。

.bat文件如下  
"C:\Program Files\Mercury Interactive\LoadRunner\bin\wlrun.exe" -TestPath "C:\Program Files\Mercury Interactive\LoadRunner\scenario\50609\_60.lrs" -ResultName "D:\temp2\50609\_60" -Run  
"C:\Program Files\Mercury Interactive\LoadRunner\bin\wlrun.exe" -TestPath "C:\Program Files\Mercury Interactive\LoadRunner\scenario\50609\_6005.lrs" -ResultName "D:\temp2\50609\_6005" -Run

**引言**

我们经常遇到这样的情况，一个系统采用了不同的算法，或是服务器使用了不同的设置，要求我们对不同的算法或服务器设置都设置测试场景进行测试，通过不同场景的测试结果得出最佳的算法或服务器设置。假设有100个这样的场景，你是否会去手动一个一个运行这100个场景呢？

**实现**

Controller(wlrun.exe)支持命令行参数，所有你只需要通过命令行给wlrun.exe传入场景文件的路径参数，实现多多个场景顺序执行。  
  
1.新建并保存所有要执行的场景，注意：每个场景必须设置结果存放路径，菜单"Results - Results Settings",设置结果路径，然后勾选“Automatically create a results directory for each scenario execution”  
  
2.新建一个bat文件，写入如下内容：

SET LR\_PATH="D:\Mercury\LoadRunner\bin\wlrun.exe"  
%LR\_PATH%\wlrun.exe -TestPath "E:\Test\Scenario01.lrs" -Run  
%LR\_PATH%\wlrun.exe -TestPath "E:\Test\Scenario02.lrs" -Run  
%LR\_PATH%\wlrun.exe -TestPath "E:\Test\Scenario03.lrs" -Run

3.双击bat文件运行。

**说明**

关于wlrun.exe的参数说明在Loadrunner中没有找到，是否还支持其他参数尚不清楚

**经实践，loadrunner多个场景运行时的批处理命令：（已经验证）**

**"C:\Program Files\HP\LoadRunner\bin\wlrun.exe" -TestPath "E:\testScenario1.lrs" –Run**

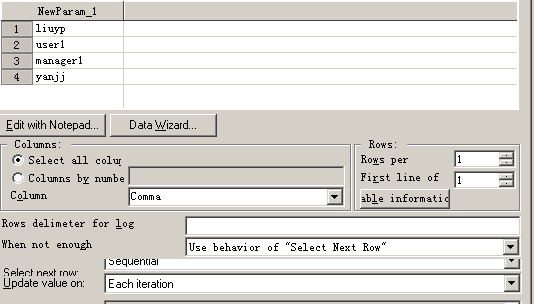
**ping –n 10 127.0.0.1**

**"C:\Program Files\HP\LoadRunner\bin\wlrun.exe" -TestPath "E:\testScenario2.lrs" –Run**

**其中：ping –n 10 127.0.0.1为延时命令，作用为第一个命令处理完后延迟10s后再执行下一命令；**

1. **Select next row=** sequential，**Update Value on=** each iteration，在某次循环中所有用户取值相同。所有用户第一次循环取第一行值，第二次循环取第二行值。

**如下图设置时：**



场景中设置3个虚拟用户时，则

第一个虚拟用户第一次循环时取值为liuyp

第一个虚拟用户第二次循环时取值为user1

第二个虚拟用户第一次循环时取值为liuyp

第二个虚拟用户第二次循环时取值为user1

第三个虚拟用户第一次循环时取值为liuyp

第三个虚拟用户第二次循环时取值为user1

理发店模式<http://www.cnblogs.com/jackei/archive/2006/11/20/565527.html>

**协议“顾问”—Protocol Advisor**

Protocol Advisor是LoadRunner9.5新添加的功能，你可以使用Protocol Advisor来帮助你决定采用什么样的协议来录制脚本。Protocol Advisor可以扫描你的应用程序，检查其中使用到的协议，并把他们显示在列表中，列出来的协议可以作为指引，指导我们应该采用什么样的协议来录制应用程序。

测试新手在使用LoadRunner时常常问的问题是“为什么我录制不了脚本？”、“我应该采用什么协议来录制？”。现在有了Protocol Advisor，问这些问题的人应该会少很多了吧！与其盲目选择协议去录制，不如在录制之前先运行Protocol Advisor，让Protocol Advisor告诉你应该采用什么样的协议。

在“预录制”过程中，Protocol Advisor记录所有找到的协议，然后把他们从高层次到低层次的顺序列出来。

但是需要注意的是，Protocol Advisor也不是万能的，不可尽信，例如它通常都会把COM/DCOM、Java、.NET、WinSocket、LDAP这些协议列出来，但是未必适合选择作为录制的协议。

　3）如何选择？什么时候选择url的录制方式：

　　\* 不是基于浏览器的

　　\* 基于浏览器但是包含javascrīpt，并且发送了请求到[服务器](http://server.chinaitlab.com/)

　　\* 使用了https [安全](http://security.chinaitlab.com/)[协议](http://cisco.chinaitlab.com/List_11.html)

　　4）两种脚本的内容比较：

　　\* http脚本直观，易于理解和维护

\* url 内容多，可伸缩性强，记录了更详细的用户操作信息。

**如何选择两种模式？http://www.cnblogs.com/fnng/archive/2013/02/28/2937805.html**

1、基于浏览器的应用程序推荐使用HTML-Based Script。

2、不是基于浏览器的应用程序推荐使用URL-Based Script。

3、如果基于浏览器的应用程序中包含了Java Script，applet等并且该脚本向服务器产生了请求，比如DataGrid的分页按钮等，也要使用URL-Based Script方式录制。

4、基于浏览器的应用程序中使用了HTTPS安全协议，使用URL-Based Script方式录制。

5、录制过程中不要使用浏览器的“后退”功能，LoadRunner对其支持不太好。

**关于思考时间**

**观点一：**

**两种思路，两种需求，并不矛盾**

|  |
| --- |
| 有思考时间的场景适用于评价真实生产实际中被测项的性能情况，该数据适合给业务部门看。 无思考时间的场景用于评价在服务器的最大负载能力，这数据对于被测项在生产中的运维监控具有指导意义，适合提供给技术部门中的生产运维人员  **观点二：** |

1：思考时间肯定是要留下的，这样更接近实际的用户使用、也减小服务器的压力。如果纯粹为了测系统的最大压力，可以不加。  
2：将思考时间函数不放到你的事务中，适当的放在在两个事务之间，这样是不是即不影响事务的时间，而切合实际情况呢。

**观点三：**

如果你做的是 负载和压力测试的话，就把think time 去掉；  
如果做得是客户验收，领导想看下真实环境下的稳定性测试，那就把 think time 加上，但结果相对来说是肯定比较乐观的。

[**LoadRunner**](javascript:;) 思考时间与事务响应时间的区别与关系  
6af K:` V0 思考时间lr\_think\_time 就是一个事务要开始时思考的时间;比如 你要点击一个 登录按钮 我们都要点击这个按钮要先思考下 就是人为脑袋思维的延迟，  
xq­Q v]k"^‑?[0还有手指点击鼠标的这个动作的时间 一般是1-5秒，这就是思考时间，[**性能测试**](javascript:;)模拟思考时间就是模拟真实人为动作的方式来做压力测试。  
a7Cw2Z#w+t0一般在脚本中思考时间是这样写比较合理，在一个事务的结束点另一个事务的起始点，两者中间定义思考时间。51Testing软件测试网tc‑N0S%o5yM'W  
lr\_end\_transaction("登录", LR\_AUTO);  
~/Vc$Q3t0lr\_think\_time(3);51Testing软件测试网,M)[ AgU UH:~%M  
lr\_start\_transaction("计算连接");51Testing软件测试网8K"Yg(Ik]$\_5r  
正常情况下思考时间越短，对服务器的压力会越大。51Testing软件测试网8q[%\_F [

51Testing软件测试网p`:zoDH&U$Q

事务的响应时间就是指从客户端发起一个请求开始，到客户端接收到从服务器端返回的响应结束，这个过程所耗费的时间。51Testing软件测试网w-~5od}  
响应时间= 网络响应时间+ 应用程序响应时间  
1U z|^#F bg]3G&ej0一般在测试结果分析时 要分析事务的响应时间 要过滤掉 思考时间。

**Analysis（结果分析）**

1、结果分析中可以对思考时间进行设置，是否在事务响应时间中添加思考时间或是去掉思考时间。

2、设置是否计算思考时间：File->set Global Filter中选择Think Time设置项，将lnclude think time值去掉勾选。这样分析中会自动将思考时间去除，这样更清楚，明确的分析系统事物的响应时间。

**url 模式录制脚本web\_concurrent\_start和web\_concurrent\_end是起到什么作用**

很多地方都没有把这个东西解释清楚。其实比较容易理解的。就是并发组这个概念把人说晕了。  
  
简单的说：  
这两个函数是在URL中标记一个页面请求的，注意：这里我说的是页面（page），并不是请求。  
在LR请求一个页面里，由于使用URL的方式录制，会把一个页面中的元素分成几个[**web**](http://www.51testing.com/?uid-228105-action-viewspace-itemid-222012)函数做处理。所以，LR中实现了web\_concurrent\_start和web\_concurrent\_end。实现的作用是：  
从web\_concurrent\_start开始标记，当脚本运行到web\_concurrent\_start时，后续的脚本都不会立即被执行，直到web\_concurrent\_end出现。才把这中间的所有的脚本一起执行。  
所谓并发组也是指把这一组函数一起执行起来。  
  
如果你用 lr\_start\_transaction和 lr\_end\_transaction来替换，脚本完全可以跑通。中间的脚本是从上到下执行的，而不是一起执行的。  
[**其他**](http://www.51testing.com/?uid-228105-action-viewspace-itemid-222012)的没有作用。

**关于关联**

**Loadrunner函数：**

3.lr\_vuser\_status\_message  
+H,t@-HV2X:NC448910在controller中显示[**日志**](javascript:;)信息51Testing软件测试网aDmyrG(xy\

51Testing软件测试网 Z(E7r2r[y7KE

显示当前参数值和当前迭代次数  
J T‑^"a!V)^448910**lr\_vuser\_status\_message("Param is:%s,%d Times Iteration",lr\_eval\_string("{NewParam}"),**51Testing软件测试网'M‑e/|P:\_\  
**++iteration);**  
&d,z6d1t9K448910return**0;**51Testing软件测试网Y8Al,\_0`?Q'i  
**}**

**Loadrunner消息函数：**

lr\_message51Testing软件测试网‑\.X-R&sr&]e"^[2@  
vugen的回放[**日志**](javascript:;)显示与否，取决于runntime settings-Log-enable logging51Testing软件测试网zgJzM+f~  
是否勾选always send messages51Testing软件测试网XjfZNv  
显示格式为直接显示，不带位置及行号，同样双击也不会跳到脚本对应位置  
(Z'{Dws­z448910日志文件output.txt

1@1@3LD+Hq6LW2s44891051Testing软件测试网HM5Hu8XO

\*x0Fhg5C5l ]-~448910lr\_log\_message  
3v]6Lbv.\_ScF$L448910类似于lr\_message,帮助里说其不会显示在output window，试了一下，结果还是显示了？51Testing软件测试网J6\_"o;z+`:g

\*}^;A&r.v5@]‑M448910lr\_error\_message51Testing软件测试网NJiLltD~5B2E\_  
显示错误信息，标红显示，如果  
$](\zEc(P0A448910runtime setting >general >miscellaneous>fail open transactions on lr\_error\_message勾选  
KbIv/Y.\_7Egl448910则调用该函数将导致之前打开的事务的状态为fail，并输出错误信息  
6HaL @ O4HN448910runtime settings里的loging设置不会对其有影响51Testing软件测试网{W e\‑A+?4\^ M  
显示格式为：位置加行号：错误信息，如Action.c(4): Error: error,双击跳到错误行51Testing软件测试网R2X4f;r`&W"B@d

51Testing软件测试网W9H T1^/i%[U;v

lr\_output\_message51Testing软件测试网9~U\_,\)a9nb b  
同样runntime settings-Log-enable logging  
‑tw/w}L/F9e }0v\448910勾选always send messages时才会在output window显示，同时写入output.txt51Testing软件测试网 b&~0FZ0G  
d\*~$T  
显示格式为：位置加行号：错误信息，如Action.c(5): output message51Testing软件测试网 \_?2e1y  
UQ­P  
双击跳到该函数所在行51Testing软件测试网9Y2h?(A\*g#g.[7kRJHx

9^$Q1W5p3Y3gi;q448910lr\_vuser\_status\_message51Testing软件测试网"D5Ay"wpR"j1]5c8s  
发送信息到controller中的Vuser 中的status列

Loadrunner操作：

1. 脚本录制

录制协议的选择

Init、action、end部分

1. 脚本处理

参数化不同参数化的执行效果；可以在log中查看执行的参数；关联方法

思考时间

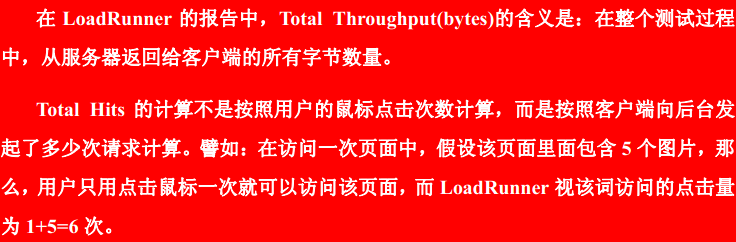
1. 场景设置及运行

场景中各设置、按scenario和group执行的区别、监控windows资源

场景类型的选择：http://blog.csdn.net/he\_jian1/article/details/7753838

1. 结果分析

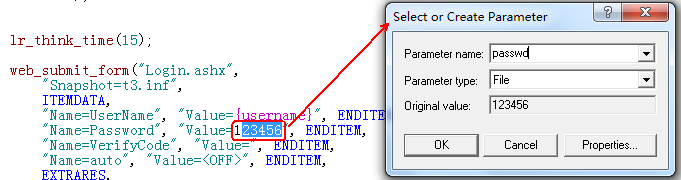
Loadrunner中参数记录默认只显示100条，超过100条的记录可以被操作但是无法显示，可以修改loadrunner安装路径中的config文件夹中的vugen.ini文件中的MaxVisibeLines为想显示的记录条数；



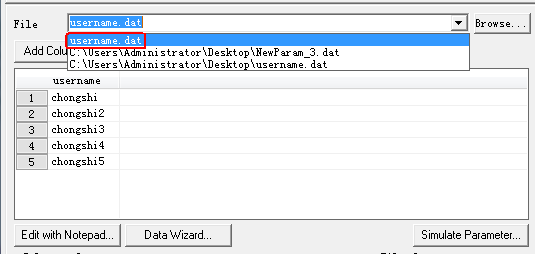
**参数化之间的关联**

　　前面我们已经对用户名进行了参数化，或对密码进行了参数化，这样是不是脚本就能正常跑了，不好说。因为用户名和密码不是一一对应关系，每次运行脚本时取的用户名和密码没有对应上的话肯定就会出问题。

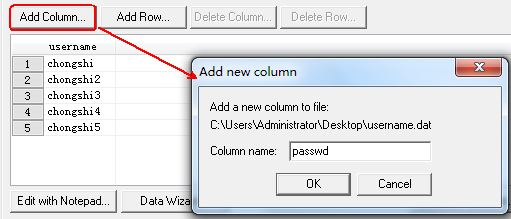
假设，我们已经对用户名已经进行了参数化，参数名为【**username**】，下面设置密码参数化与用户名关联。



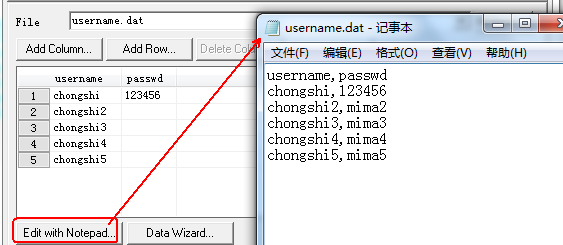
点击“**Properites…**”会打开编辑用户名参数化窗口。**File**列表框中，刚才保存用户名信息的文件"**username.dat**"。



点击“**Add Column…**”,添加新的一列信息，用于放置密码。



点击“**Edit with Notepad**”再次编辑参数化数据文件，使用户名密码建立一一对应关系。



完成之后，我们已经成功对用户名和密码进行了参数化，并且让用户名和密码形成了对应关系。

问题及解决方法(关于Simulate a new user on each iteration设置项)：

一些Web虚拟用户脚本录制后立刻回放没有任何问题，但是当设置迭代次数大于1时，如果进行回放则只能成功迭代一次。为什么从第二次迭代开始发生错误?

解决办法：这种现象多是由于在“Run-time Setting”的“Browse Emulation”的设置中，勾选了“Simulate a new user on each iteration”及其下面的选项“Clear cache on each iteration”这两个选项的含义是每次迭代时模拟一个新的用户及每次迭代时清除缓存。由于脚本迭代时，init和end只能执行一次，如果每次迭代都模拟一个新的用户并清除缓存，则用户登录信息将一并清除，因此迭代时可能会发生错误。

另：参考<http://www.51testing.com/html/59/16059-16782.html>

Loadrunner常用函数：

Web\_url()

Web\_link()

Web\_submit\_form()

Web\_submit\_data()

Web\_custom\_request()

1. **web\_add\_cookie()**

主要负责为vuser脚本添加一个cookie信息，在脚本中可以删除掉cookie信息。

例如在DM中，在录制脚本初始处：web\_add\_cookie("userId=admin; DOMAIN=192.168.1.24");该句脚本可以删除。

2、**web\_url()**

该函数可以模拟用户请求，是在脚本中最常使用的函数之一，它实现HTTP请求中的GET方法。

web\_url()函数的基本语法如下所示：

web\_url(“在测试结果中显示的名称”，“URL=需要访问的超链接地址”，LAST)

3、**web\_link()**

web\_link()函数用例模拟用户单击一个超链接的操作。当使用web\_link()函数时，只要写出正确的链接名，VuGen会自动查找并访问页面中该链接所指向的URL地址，它实现HTTP请求中的GET方法。

web\_link()函数的基本语法如下：

web\_link(“在测试结果中显示的名称”，“TEXT=需要单击的超链接名”，LAST)；

4、**Web\_submit\_form()**

Web\_submit\_form()函数会自动检测在当前页面上是否存在form表单，然后让后面的ITEMDATA数据进行传送。当一个页面上有多个表单时无法录制得到Web\_submit\_form()

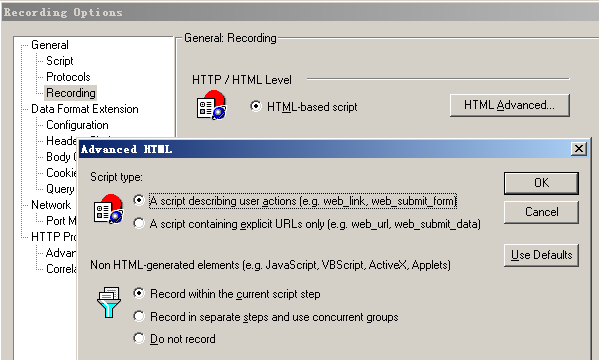
函数，LR会自动使用Web\_submit\_data()替代。它实现HTTP请求中的POST方法

5、**Web\_submit\_data()**

Web\_submit\_data()函数与Web\_submit\_form()函数功能相同，只是函数使用方式及内容具体处理方式不同。它实现HTTP请求中的POST方法。

6、**Web\_custom\_request()**

当请求比较特殊，VuGen无法简单使用Web\_url()、Web\_link()、Web\_submit\_form()、Web\_submit\_data()这几个函数进行表述时，那么录制后就会出现Web\_custom\_request()函数，这个函数的作用是自定义HTTP规则。Web\_custom\_request()函数既可以实现HTTP中的GET方法，又可以实现POST方法。



A script describing user actions

基于解释用户行为的脚本，这种脚本录制出来有些类似于QTP基于对象识别操作的模式，具体描述了用户进行了什么样的操作。Web\_link()函数是使用A script describing user actions模式才可以得到的。

A script containing explicit URLs only

该选项确定基于URL请求的脚本录制类型。与基于用户行为模拟的不同之处在于，这种录制方式不考虑任何用户操作，只考虑客户端发出的请求，注重系统实际上做了什么。使用A script containing explicit URLs only录制模式时，web\_link()函数被web\_url()函数代替。

**针对ajax请求录制时遇到的问题**

最近利用loadrunner进行一个项目的性能测试测试，通过页面录制后得到测试脚本；

在对脚本进行回放操作时，脚本成功回放，没有任何错误；但是，在查看后台记录时，却查看不到任何的操作记录；

后来，切换到tree view界面中，对比录制和回放的结果；发现很多操作的回放响应结果不正确；最后，发现回放的信息头和录制的信息头相比，缺少部分信息头字段，将这些缺少的信息头添加在脚本中，再次回放，发现脚本中操作成功执行了。

后在网上了解到，缺少的信息头字段X-Requested-With: XMLHttpRequest是定义这个请求是ajax请求的，我们的项目中采用了ajax，而loadrunner录制时，不能录制ajax请求，所以需要手动添加X-Requested-With: XMLHttpRequest，标明请求为ajax请求，loadrunner才会成功执行操作。

# Loadrunner 协议中Web(click and script) 与 Web(HTTP/HTML)协议的区别

WEB(Click and Script)是基于页面操作的，方式跟QTP的方式差不多，所以当页面中有applet等程序的时候，由于无法从控件操作的角度捕获，所以只能用WEB(HTTP/HTML)方式来录制。

Web (Click and Script) 协议的录制是基于GUI的、用户实际操作界面过程的脚本，记录的是浏览器和服务器的WEB对话，你选择了该协议后，VuGen记录的是你在WEB界面上的操作的直观过程。例如，当你点击“提交”按钮提交信息时，VuGen会生成web\_button函数，当你在编辑框中输入时，VuGen生成web\_edit\_field函数

Web(Click and Script)的Vusers支持非HTML代码，比如客户端的Javascript。VuGen会创建直观的脚本来精确的模拟你在web页面上的操作过 程。相反，Web(HTTP/HTML)协议不支持Javascript，VuGen只是把Javascript作为web\_url函数的一个资源。

Web(Click and Script)的Vusers能够自动处理大多数关联（correlations）的问题，大大减少脚本编程的时间。通常情况下，你不需要再去定义关联的规则或者在录制后手动再去做关联。

而WEB(HTTP/HTML)是从纯消息的角度来录制，因此无论页面内用了什么技术，只要将客户端与服务端交互的消息录制下来即可

当你选择Web(HTTP/HTML)协议录制时，VuGen记录的是在Internet上传送的“浏览器发出的HTTP请求和服务器的响应”的数据，脚本中包含了你的浏览器请求的数据详细信息，而不是操作过程的信息。

对于大部分的应用，包括使用Javascript的应用，使用Web(Click and Script)协议；对于使用applets和vbscript的基于浏览器的应用，或者非浏览器的web应用，使用Web(HTTP/HTML)协议

# WEB(HTTP/HTML)协议中HTML与URL录制方式的选择

HTML方式为loadrunner默认和推荐的录制模式，是将类属于一个页面的请求放在一个web\_url中，录制的脚本比较简短，便于阅读；而URL方式中将客户端向服务器端发送的每一个请求都放在一个web\_url中，录制的脚本相对比较长，不便于阅读，但比较直观；

HTML录制方式下，mode值为‘html’，URL录制方式下，mode值为‘http’；

基于浏览器的应用建议使用HTML；

不是基于浏览器的应用、包含javascript且javascript脚本向服务器发送了请求的应用程序、应用了https安全协议的基于浏览器的应用程序，建议使用URL录制方式；

# 在Loadrunner场景中，Vuser\_init和Vuser\_end中的参数值的取值问题

当在场景中迭代运行时，

对于每一个虚拟用户来说：

Vuser\_init中的脚本仅运行一次，

Action中的脚本进行迭代

Vuser\_end中的脚本仅运行一次

虽然Vuser\_init中的脚本仅运行一次，但是Vuser\_init中的参数却随着虚拟用户的每次迭代，按照该参数所设置的迭代方式进行迭代；

同样，Vuser\_end中的参数也是如此；

在实际运用过程中就会产生问题：

如UC中，登录用户参数化为NewParam，且该登录脚本放在Vuser\_init中；

用户退出脚本放在Vuser\_end中，且退出用户和登录用户同为一个参数NewParam；那么在场景中运行时，因用户登录脚本放在Vuser\_init中，用户登录后（假设取值为user1），在场景中就以该用户进行迭代；但同时NewParam值也是随着迭代在进行，所以，当停止脚本时，运行Vuser\_end中的用户退出脚本，此时退出的用户却是经过无数次迭代后的NewParam，而不是user1，这样就会导致user1不能被成功退出；

解决该问题，可以设置NewParam的更新方式为once，则此问题便可解决；