

目 录

1. (完成) 监控tomcat	2
1.1、Loadrunner.....	2
1.2、Jmeter	6
2. (完成) 监控Windows资源	9
2.1、Loadrunner.....	9
2.2、Jmeter	10
3. (完成) 监控Linux资源	12
3.1、Loadrunner	12
3.2、Jmeter	13
4. (完成) 集合点.....	14
4.1、Loadrunner	14
4.1.1、脚本.....	14
4.1.2、场景运行	14
4.2、Jmeter	14
4.2.1、场景设置	14
4.2.2、注意事项	14
5. (完成) 参数化.....	15
5.1、Loadrunner	15
5.1.1、脚本.....	15
5.1.2、场景运行	15
5.2、Jmeter	15
6. (完成) 接口测试	17
6.1、Loadrunner	17
6.2、Jmeter	18
6.2.1、HTTP请求	18
6.2.2、事务控制器.....	18
7. (完成) 关联函数	19
7.1、Loadrunner	19
7.2、Jmeter	20
8. (部分完成) 数据库性能测试	22
8.1、Loadrunner (未写)	22
8.2、Jmeter (完成)	22
9. (完成) 检查点.....	24
9.1、Loadrunner	24
9.2、Jmeter	27
9.2.1、固定值	27
9.2.2、动态值	28
10. (部分完成) JAVA性能测试	30
10.1、Java Vuser (--未写--)	30
10.2、BeanShell Sampler	30
10.2.1、例子1 (固定值)	31
10.2.2、例子2 (动态值)	33
11. (持续中) 常用函数 (loadrunner)	33

11.1、lr_eval_string	33
11.2、lr_output_message.....	34
11.3、strcmp	34
12. (部分持续中) 工具设置	35
12.1、Loadrunner (--未写--)	35
12.1.1、多机负载设置	35
12.2、Jmeter (持续中)	35
12.2.1、线程组设置	35
12.2.2、并发设置.....	36
12.2.3、多机负载测试	37

1. (完成) 监控tomcat

1.1、Loadrunner

1、首先确保能成功登陆进tomcat控制台

操作：打开Tomcat的status页面，方法为编辑Tomcat的conf目录下的tomcat-users.xml文件，在文件中添加

```
<?xml version='1.0' encoding='utf-8'?>
```

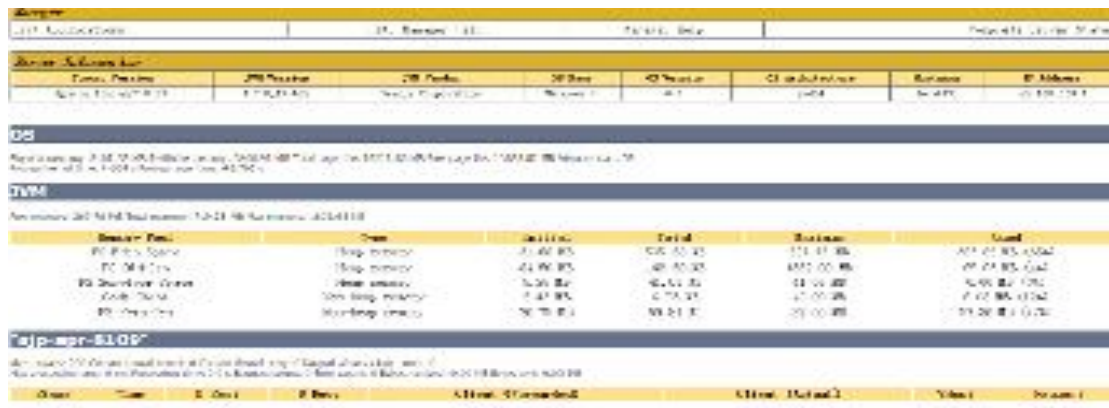
```
<tomcat-users>
```

```
    <role rolename="manager-gui"/>
```

```
    <user username="tomcat2" password="tomcat" roles="manager-gui"/>
```

</tomcat-users>

接着启动tomcat服务，输入http://localhost:9999/manager/status，界面如下



The screenshot shows the Tomcat Manager status page. It includes a top navigation bar, a status table, and a detailed JVM status section.

Context Path	URL	State	Restart	Stop
/	localhost:8080	Running	Restart	Stop

Category	Item	Value
JVM	Free memory	104,400,000
	Total memory	1,048,576,000
	Max memory	1,048,576,000
	Used memory	944,176,000
HTTP	Current threads	10
	Max threads	100
	Queue size	100
	Accepting connections	Yes

2、监控思路

- A、在Action脚本中，使用web_set_user("用户名","密码","tomcat服务器所在的IP地址:端口")
- B、脚本中编写web_url(); 模拟访问Tomcat的url 并登录
- C、利用关联函数web_reg_save_param()动态地捕获想要的参数
- D、最后利用打点函数lr_user_data_point(“监控指标名”,“监控指标值”);记录用户自定义的数据样本

3、脚本内容

```
double atof(const char * string);
```

```
web_reg_save_param("JVM_FreeMemory", //关联取值
```

```
    "LB=Free memory: ",  
    "RB= MB",  
    "Ord=1",  
    LAST);
```

```
web_reg_save_param("JVM_TotalMemory",
```

```
    "LB=Total memory: ",  
    "RB= MB",  
    "Ord=1",  
    LAST);
```

```
web_reg_save_param("JVM_MaxMemory",
```

```
    "LB=Max memory: ",  
    "RB= MB",  
    "Ord=1",  
    LAST);
```

```
web_reg_save_param("HTTP_MaxThreads",
```

```
    "LB=Max threads: ",  
    "RB= ",
```

```

        "Ord=1",
        LAST);
web_reg_save_param("HTTP_MaxProcessingTime",
    "LB=Max processing time: ",
    "RB= s",
    "Ord=1",
    LAST);
web_reg_save_param("HTTP_RequestCount", //HTTP请求计数
    "LB=Request count: ",
    "RB= ",
    "Ord=1",
    LAST);
web_reg_save_param("HTTP_BytesReceived", //收到的字节数
    "LB=Bytes received: ",
    "RB= MB",
    "Ord=1",
    LAST);
web_set_user("admin","admin","localhost:9999");
lr_think_time(5);
web_url("status", //访问Tomcat的url 并登录
    "URL=http://localhost:9999/manager/status",
    "Resource=0",
    "Referer=",
    "Snapshot=t1.inf",
    "Mode=HTML",
    LAST);
lr_user_data_point("Tomcat_JVM_FreeMemory",atof(lr_eval_string("{JVM_FreeMemory}")));
lr_user_data_point("Tomcat_JVM_TotalMemory",atof(lr_eval_string("{JVM_TotalMemory}")));
lr_user_data_point("Tomcat_JVM_MaxMemory",atof(lr_eval_string("{JVM_MaxMemory}")));
lr_user_data_point("Tomcat_HTTP_MaxThreads",atof(lr_eval_string("{HTTP_MaxThreads}")));
lr_user_data_point("Tomcat_HTTP_MaxProcessingTime",atof(lr_eval_string("{HTTP_MaxProcessingTime}"))); //
lr_user_data_point("Tomcat_HTTP_ProcessingTime",atof(lr_eval_string("{HTTP_ProcessingTime}")));
lr_user_data_point("Tomcat_HTTP_RequestCount",atof(lr_eval_string("{HTTP_RequestCount}")));
lr_user_data_point("Tomcat_HTTP_BytesReceived",atof(lr_eval_string("{HTTP_BytesReceived}"))); //记录用户自定义的数据样本

```

写法见Loadrunner附件之“**MonitorTomcat**”。

4、运行脚本再查看回放日志

```

manager/status" [MsgId: WMSG-26659]
Action.c(51): Found resource "http://localhost:9999/manager/images/tomcat.gif" in HTML "http://localhost:9999/manager/
status" [MsgId: WMSG-26659]
Action.c(51): Web_UPS("status") was successful, 18750 body bytes, 1119 header bytes, 12 chunking overhead bytes
[MsgId: WMSG-26385]
Action.c(59): Notify: Data Point "Tomcat_JVM_FreeMemory" value = 149.1100.
Action.c(61): Notify: Data Point "Tomcat_JVM_TotalMemory" value = 686.7500.
Action.c(63): Notify: Data Point "Tomcat_JVM_MaxMemory" value = 1802.6800.
Action.c(65): Notify: Data Point "Tomcat_HTTP_MaxThreads" value = 200.0000.
Action.c(67): Notify: Data Point "Tomcat_HTTP_MaxProcessingTime" value = 0.0000.
Action.c(69): Notify: Data Point "Tomcat_HTTP_ProcessingTime" value = 0.0000.
Action.c(71): Notify: Data Point "Tomcat_HTTP_RequestCount" value = 0.0000.
Action.c(73): Notify: Data Point "Tomcat_HTTP_BytesReceived" value = 0.0000.
Ending action Action.

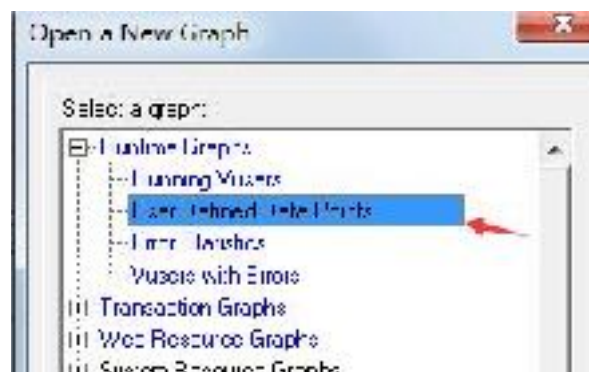
```

注意：将以上脚本代码放在新插入一个**Action**里面，并且放在录制来的业务脚本后面。

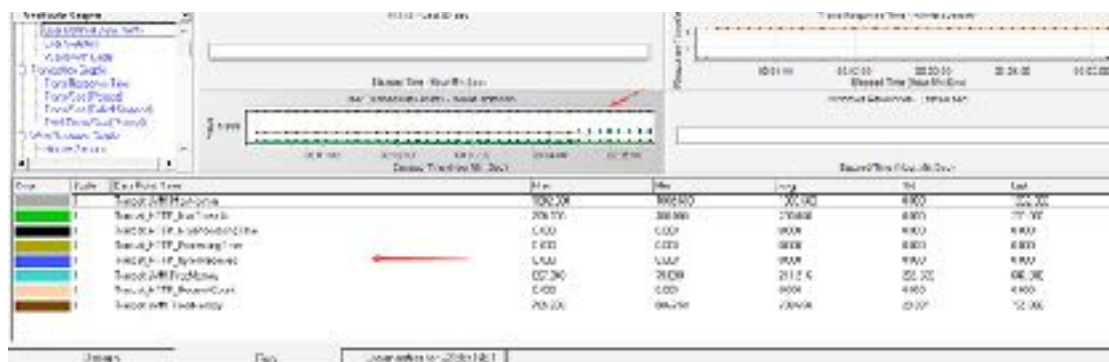
5、创建Controller等与平时的性能测试流程一样就好了

6-1、接着在Controller监控场景图中，添加“运行时图-用户定义的数据点图”

操作：添加“Open a New Graph...”然后点选“User Defined DatePoints”，如下图所示



6-2、这样用户自定义的数据图点就显示出来了，如下图所示



7、最后在Analysis生成结果分析报告中，可以看到“运行时图-用户定义的数据点图”是显示有数据的，如下图所示



图1

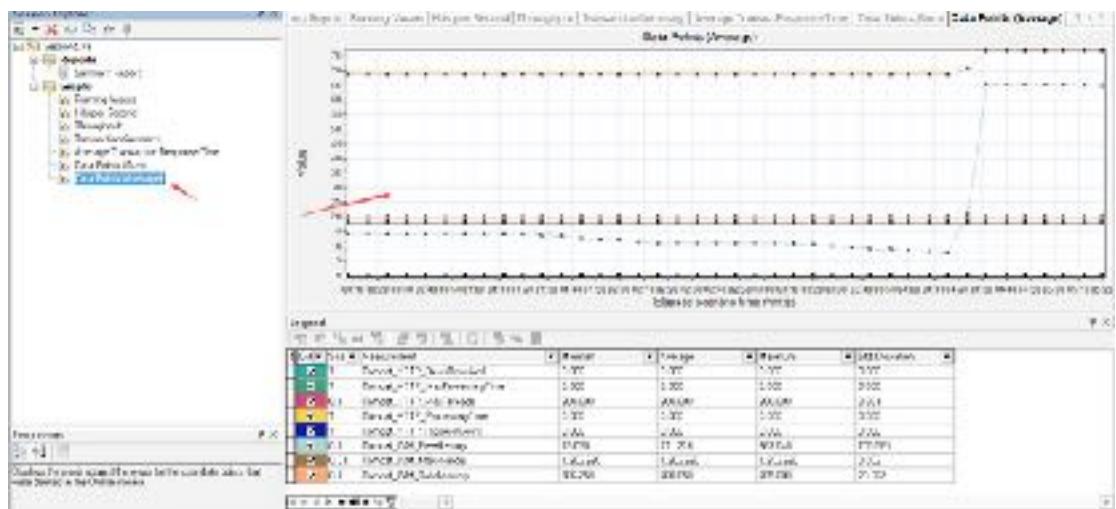


图2

1.2、Jmeter

使用jmeter的tomcat监视器功能，可以通过向tomcat的status页面发送get请求，得到资源使用信息，然后转换为直观的图像方式，这样的话，就可以监视到服务器的资源使用情况，不过需要注意的是，要使用tomcat监视器功能，就必须在要监视的服务器上装有tomcat。

1、配置/conf/tomcat-users.xml，内容如下

```
<?xml version='1.0' encoding='utf-8'?>
<tomcat-users>
  <role rolename="manager-gui"/>
  <role rolename="manager-script"/>
  <role rolename="manager-jmx"/>
  <role rolename="manager-status"/>
</tomcat-users>
```

```

<user username="admin" password="admin" roles="manager-gui"/>
<user username="admin" password="admin" roles="manager-script"/>
<user username="admin" password="admin" roles="manager-jmx"/>
<user username="admin" password="admin" roles="manager-status"/>
</tomcat-users>

```

2、配置/conf/context.xml，内容如下

```

<Context>
  <WatchedResource>WEB-INF/web.xml</WatchedResource>
  <Manager pathname="/manager" debug="0" privileged="true" docBase="/home/
tomcat-9900/webapps/manager" />
  <Valve className="org.apache.catalina.valves.CometConnectionManagerValve" />
</Context>

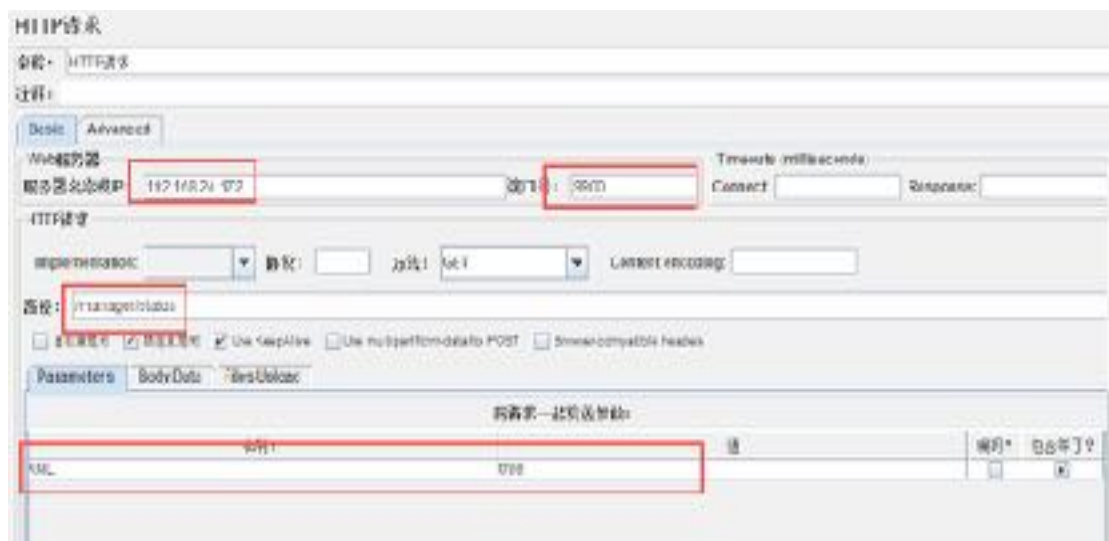
```

注意：配置完成后先访问<http://192.168.24.172:9900/manager/status>，检查是否配置成功。

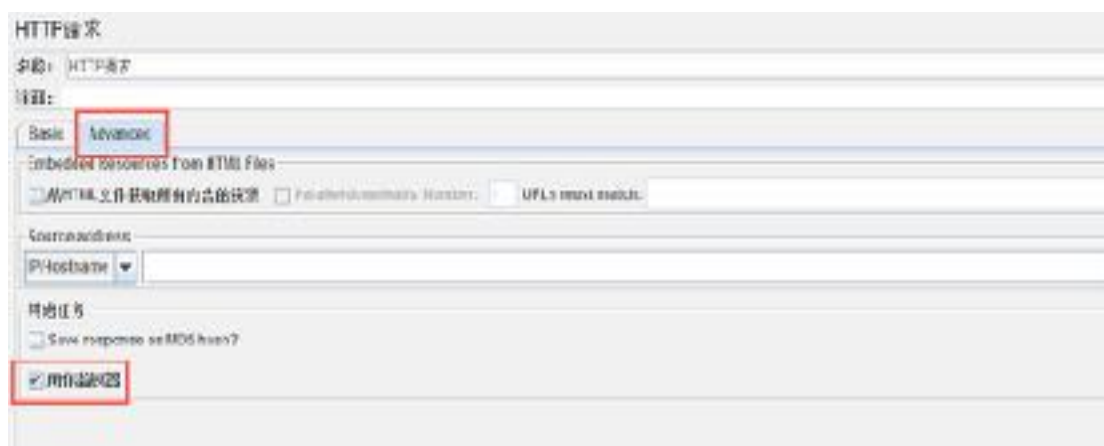
3、打开Jmeter在线组下添加“配置元件-HTTP授权管理器”，在HTTP授权管理的“基础URL”中输入<http://192.168.24.172:9900/manager/status>，用户名为上面新建的admin，密码是admin，其他默认，如下图所示



4、接着在当前线程组下添加“HTTP请求”，输入被检测服务器中的TOMCAT访问IP和端口号；路径输入 /manager/status；请求参数输入XML（必须大写），值为true，如下图A1所示并且将此HTTP请求设置为“用作监视器”，如下图A2所示



图A1



图A2

5、接着在当前HTTP请求下添加“监视器结果”和“察看结果树”，如下图所示



6、最后运行JMETER脚本，此时进入到“监视器结果”界面可以看到被测服务器IP和资源（图片），如下图所示



写法见Jmeter附件之“附件02（监控Tomcat资源）”。

2.（完成） 监控Windows资源

2.1、 Loadrunner

1、打开服务器Windows 的两个服务

Remote Procedure Call （RPC）

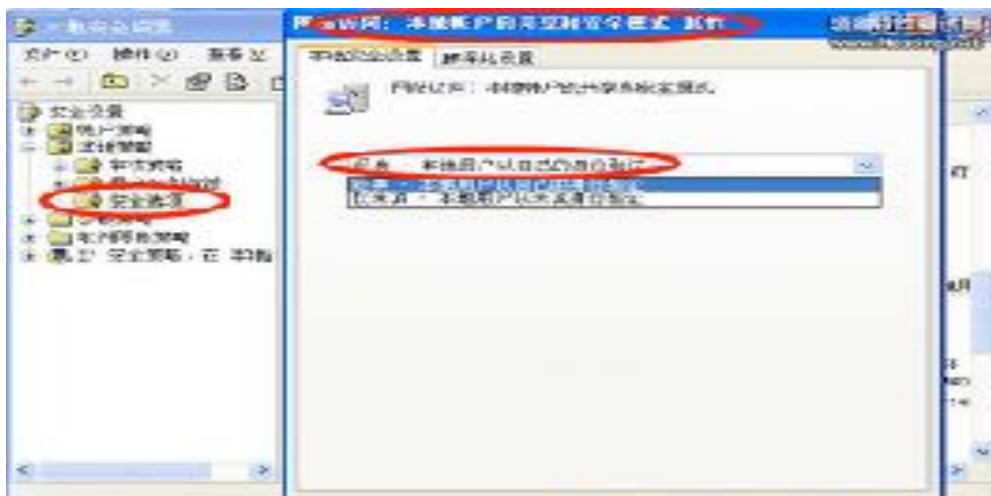
Remote Registry

路径：控制面板-管理工具-服务

Remote Packet Capture Protocol v.0...	Allow...	手动	本地系统
Remote Procedure Call (RPC)	RPC...	已启动	网络服务
Remote Procedure Call (RPC) Locator	在 W...	手动	网络服务
Remote Procedure Call (RPC) Net	已启动	自动	本地系统
Remote Registry	停止...	已启动	本地服务

2、修改服务器的本地策略，改为经典模式

路径：控制面板-管理工具-本地安全策略



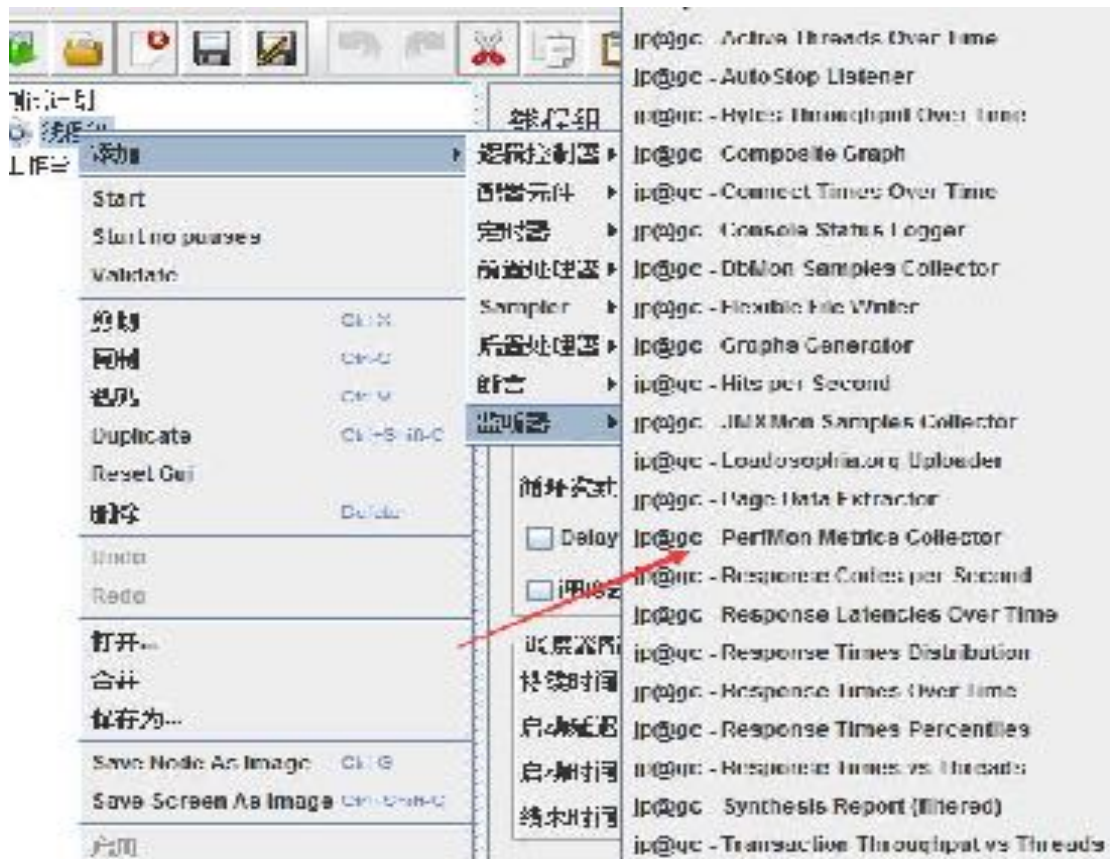
3、在客户端进行测试，在“运行”栏中输入服务器的ip地址，后面跟上C\$，表示服务器C盘下的系统资源目录，如：“\\192.168.96.135\C\$”，看看是否可以访问服务器C盘目录，通常情况下可能需要输入用户名和密码，填充服务器端的账户和密码就 ok，如下所示

JMeterPlugins-Standard-1.3.1 (Jmeter插件)

ServerAgent-2.2.1 (监听工具)

2、解压Jmeter的2个插件，将JMeterPlugins-Extras.jar和JMeterPlugins-Standard.jar复制到jmeter安装目录下的lib\ext下

3、启动Jmeter后在“监听器”中就显示出可以监控的资源，如下图所示



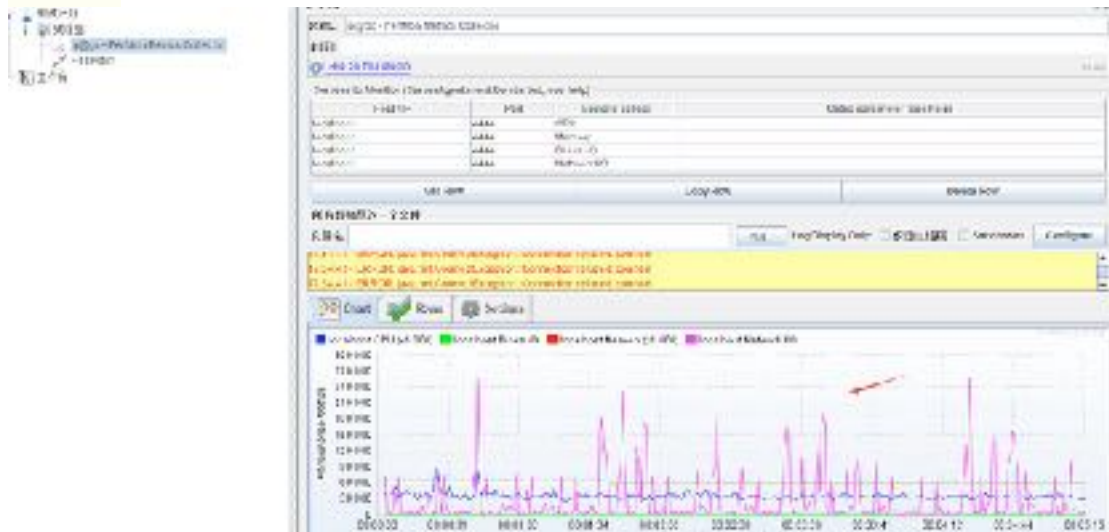
4、现在需要将ServerAgent-2.2.1解压后放在需要监控的服务器上并启动startAgent.bat服务

5、现在回到Jmeter，在线程组中添加要监听的服务器资源，比如CPU和内存和网络硬盘I/O，操作如下：

A、首先添加HTTP请求并输入属性后，再在当前HTTP所在的线程组上添加jp@gc - PerfMon Metrics Collector，界面如下图所示



B、接着在jp@gc - PerfMon Metrics Collector页面中添加要监听的服务器资源，端口号默认是4444。为了能监控到资源变化，我将线程组设置为无限循环，最后启动脚本，结果如下图所示



注：上图中报错是未启动ServerAgent-2.2.1时就执行了Jmeter脚本。

写法见Jmeter附件之“附件06（监控windows资源）”。

3.（完成） 监控Linux资源

3.1、 Loadrunner

先检查Linux服务器上是否已安装和启动了NFS和POETMAP服务，如果未安装请让运维人员在服务器上进行安装再进行接下来的操作。添加Linux服务的操作与Windows类似，这里不再累述。如果添加Linux服务时提示为rpc服务在客户端添加失败，如果遇到这种情况基本可以确定远程没有获取到rpc服务看看系统防火墙，cat /etc/sysconfig/iptables的配置，果然需要加入rpc访问策略，为此我关闭了防火墙服务，所以就不需要设置防火墙策略了，命令为：

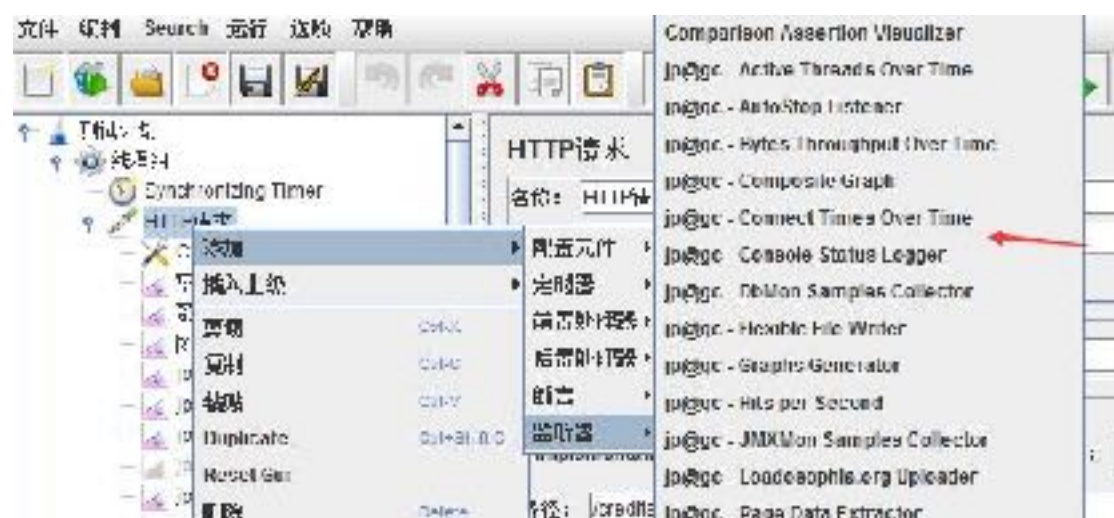
chkconfig iptables on //永久性启动防火墙

chkconfig iptables off //永久性关闭防火墙

关闭防火墙之后再在LoadRunner场景中添加Linux资源即可。

3.2、Jmeter

- 1、首先下载JMeterPlugins-Extras-1.4.0和JMeterJMeterPlugins-Standard-1.3.1插件（下载地址：<https://jmeter-plugins.org/downloads/old/>）
- 2、然后将解压出来的JAR包放到Jmeter/lib/ext目录下
- 3、再下载ServerAgent-2.2.1.zip（默认端口4444，下载地址：<http://www.cnblogs.com/ljsjddddd/p/5709052.html>）并部署到Linux下
- 4、在Linux下进入到ServerAgent目录下启动服务：sh startAgent.sh
- 5、打开Jmeter在HTTP请求下添加“监视器”即可看到JP@开头的监视元素



- 6、监控Linux服务器资源添加“jp@gc-PerMon Metrics Collector”，然后输入Linux的IP，端口号4444,和要监控的资源，如下图所示



注意：监控完一次服务器资源后，ServerAgent会自动关闭，需要重新启动。否则第二次执行Jmeter脚本会报错。

写法见Jmeter附件之“附件04（监控Linux资源）”。

4.（完成）集合点

4.1、Loadrunner

4.1.1、脚本

在提交函数上方添加集合点（**lr_rendezvous**），集合点不要写在事务统计时间函数里面。

4.1.2、场景运行

A、1个脚本多个Action（每个都有一个集合点）运行场景：多次集合并发；

B、多个脚本1个Action（一个集合点）运行场景：多个脚本用户均到达集合点才并发。

4.2、Jmeter

4.2.1、场景设置

在Jmeter中，集合点叫定时器，使用的是“定时器-Synchronizing Timer”,直接将它移放到HTTP请求上方即可，然后设置相应属性值：在Synchronizing Timer界面中的“Number of Simulated Users to Group by:”输入并发数（表示当X个用户到达此集合点时就开始并发）。



4.2.2、注意事项

集合点中的并发数最好能被线程组中设置的用户数整除（线程用户数%并发数=0），

不然跑脚本会报错。集合点在线程组中可以随意拖动，放在请求前面。

集合点可以和线程组的“线程属性”和“调度器”结合使用，主要是为了保证所有的用户能到达集合点并能发生并发。

5.（完成）参数化

5.1、Loadrunner

5.1.1、脚本

LR参数化的变量有三种方式：

- A、**each occurrence** 一次迭代中脚本里面的参数每出现一次，其值生成一次；
- B、**each iteration** 一次迭代中脚本里面的参数只生成一次，无论出现多少次；
- C、**once** 随机的数字只出现一次，即已得到的数字, 之后不会再得到。

5.1.2、场景运行

- A、场景中1个用户，脚本迭代3次

脚本：Sequential + each iteration 成功3个，取值正确且唯一。

- B、场景中10个虚拟用户，脚本迭代1次

脚本1：Unique + each iteration + AbortVuser 成功10个，取值正确且唯一；

脚本2：Sequential + each iteration 第一个成功，后面9个全部失败。

5.2、Jmeter

- A、在HTTP请求上添加“配置元件-CSV Data Set Config”，如下图所示



B、元件添加成功后，使用TXT新建一个CSV文件（注意，这里必须使用TXT文件然后另存为CSV文件，不然执行时会报错全是乱码）；

C、在CSV文件中输入多个值（这里是手机号，一行显示一条手机号，我在CSV文件中写了5个不同的手机号），然后保存CSV文件（记住存放路径）；

D、新建CSV文件完成后，切换到刚才添加成的CSV Data Set Config窗口，在“Filename”输入CSV的存放地址（包括CSV文件名称），在Variable Names (comma-delimited)中输入自定义的变量名称（此变量会用到HTTP请求4的参数值中，我这里设置为phonenum），其他默认，如下图所示



现在切换到HTTP请求窗口中，在Parameters标签页面中添加4个参数（因为此接口带4个参数）imagecode、token、type、phone。参数值分别是zds4、\${token}（HTTP请求1中的正则表达式）、fql、\${phonenum}（刚刚设置的参数化），如下图所示



6. (完成) 接口测试

6.1、Loadrunner

1、选择HTTP协议；

2、使用`web_custom_request`函数获得参数值；

```
web_custom_request("get_to_seller_fee", //参数名随便起
    "URL=http://one.1v1.one:8089/creditsys/wap/payBill", //访问地址
    "Method=GET", //请求类型是GET
    "Resource=0",
    "Referer=",
    "Mode=HTTP"
    "EncType=text/html;charset=UTF-8",
    "Body=",
    LAST);
```

3、再使用`web_submit_data`函数发送参数；

```
web_submit_data("repayPlanReturn", //事务名，随便起
    "Action=http://one.1v1.one:8089/creditsys/alipay/repayPlanReturn", //访问地址
    "Method=POST", //请求类型为POST
    "RecContentType=application/json", //返回格式为json
    "Referer=http://one.1v1.one:8089/creditsys/alipay/repayPlanReturn", //引用地址
    "Mode=HTML",
    ITEMDATA, //下面编辑POST请求的参数值
    "Name=trade_status", "Value=TRADE_SUCCESS", ENDITEM,
```

```
"Name=to_seller_fee", "Value={get_amount}", ENDITEM,  
.....  
LAST);
```

4、使用关联（web_reg_save_param）替换返回字符串中的值。关联函数写在GET和POST函数之前，注意左右边界，JSON返回字符串中的双号要用“\”进行处理。

```
web_reg_save_param("get_amount", //获取to_seller_fee值，变量名是get_amount  
"LB=\\\\"amount\\\\"":", //关联  
"RB=",  
"Search=All",  
LAST);
```

5、运行脚本检查结果是否正确。

写法见Loadrunner附件中的脚本“lr-script-01”。

6.2、Jmeter

6.2.1、HTTP请求

HTTP单接口测试可以直接添加HTTP接口输入输入访问地址，选择访问方式（GET/POST）。如果是GET方式就不需要添加参数，而如果是POST方式则需要添加参数，对于返回信息中存在动态值而且需要使用这些动态值，则要使用正则表达式来得到动态值。

6.2.2、事务控制器

HTTP的多接口测试，特别是多个接口之间存在关联的访问，就需要用到事务控制器。不然在执行脚本时，会随机调用接口，这样不符合科学。比如现在有2个HTTP接口，第一个是获取信息，第二个是使用获取的信息作为参数传个服务器。结构如下图所示



操作：

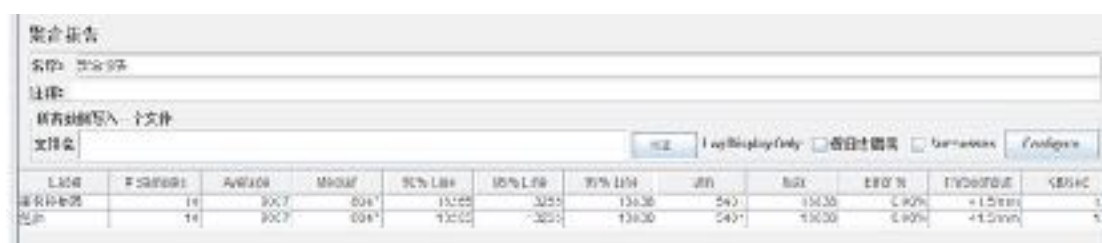
A、将有关联的HTTP请求放在一个线程组下；

B、然后在该线程组上添加“逻辑控制器-事务控制器”并移动到关联的接口上面，如上图所示；

C、接着设置“事务控制器”属性，将两个选项选上即可，如下图所示



注意：在“聚合报告”和“用表格查看结果”中只能看到“事务控制器”，无法看到事务控制器包含的HTTP请求，如下图所示



Label	# of Samples	Average	Min	90% Line	95% Line	99% Line	Std	Bias	Error	Throughput	Delay
事务控制器	10	300.7	204	15.00	32.5	130.30	54.0	130.30	0.00%	~1.5/min	0.0
总计	10	300.7	204	15.00	32.5	130.30	54.0	130.30	0.00%	~1.5/min	0.0



Sample #	Start Time	Thread Name	Label	Sample Time (ms)	Status	Delay	Latency	Connect Time (ms)
1	17:24:22.470	Thread 1-1	事务控制器	1639	Success	263.4	0	0
2	17:24:27.469	Thread 1-2	事务控制器	2782	Success	253.4	0	0
3	17:24:34.469	Thread 1-3	事务控制器	2899	Success	253.4	0	0
4	17:24:40.467	Thread 1-4	事务控制器	1747	Success	261.4	0	0
5	17:24:45.469	Thread 1-5	事务控制器	249	Success	253.4	0	0
6	17:24:50.467	Thread 1-6	事务控制器	1169	Success	253.4	0	0
7	17:24:57.468	Thread 1-7	事务控制器	1189	Success	261.4	0	0
8	17:25:04.468	Thread 1-8	事务控制器	604	Success	253.4	0	0
9	17:25:09.468	Thread 1-9	事务控制器	1129	Success	263.4	0	0
10	17:25:16.469	Thread 1-10	事务控制器	2472	Success	253.4	0	0

事务控制器包含的多个HTTP请求可以在“察看结果树”中查看到。

7.（完成）关联函数

7.1、Loadrunner

函数：[web_reg_save_param](#)替换返回字符串中的值。关联函数写在GET和POST函数之

前，注意左右边界，JSON返回字符串中的双号要用“\”进行处理。

```
web_reg_save_param("get_amount", //获取to_seller_fee值，变量名是get_amount
    "LB=\\\"amount\\\":",
    "RB=",
    "Search=All",
    LAST);
```

得到的动态返回值可以放在web_submit_data函数的Value处，用{关联名称}代替。

7.2、Jmeter

在Jmeter中，关联叫做“正则表达式”，当返回的信息中存在动态值而且会被其他接口使用，此时就需要使用“正则表达式”来保存动态值。比如此时调用一个接口返回一些信息：
{"msg": "\u6210\u529f", "code": 0, "detail": {"token": "a53b93507c9d4f3fa43de4a44073bbb6"}}。返回信息中的token是动态值，需要正则下。具体操作如下：

A、在有次信息返回的接口处添加“[后置处理器-正则表达式提取器](#)”，如下图所示



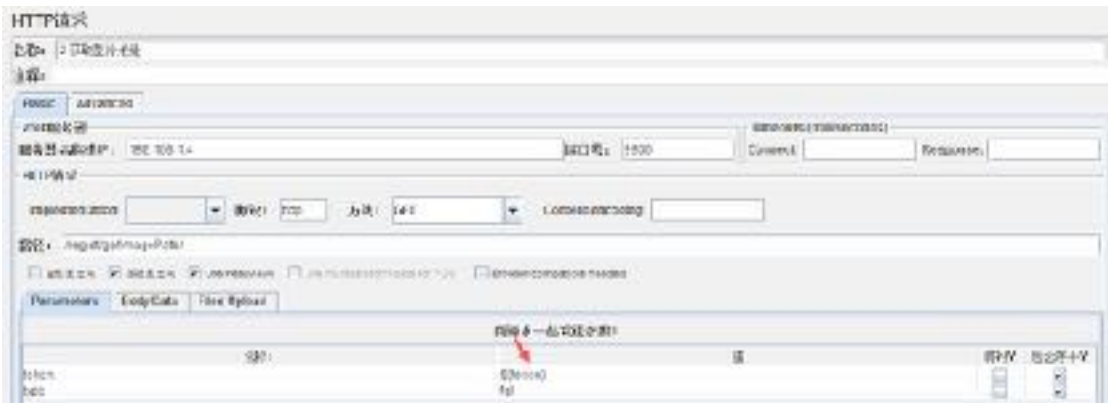
B、输入提取器中的引用名称（请随意），此名称即为该正则的参数名，会在其他接口

中引用到。刚才我们得到了响应结果：{"msg": "\u6210\u529f", "code": 0, "detail": {"token": "a53b93507c9d4f3fa43de4a44073bbb6"}}。正则的用法与LR的手动关联类似，{"token": "a53b93507c9d4f3fa43de4a44073bbb6"}需要关联，那么正则表达式的写法就是：**"token": "(.+?)"**，如下图所示



C、运行脚本，如果在“察看结果树”中看到绿色则表示正则成功；

D、正则运行成功过，打开需要应用此正则的HTTP接口，添加参数名和参数值（用{正则引用名}表示），如下图所示



E、运行线程组，如果一片绿色则代表成功了.....

写法见Jmeter附件之“附件01”。

8.（部分完成）数据库性能测试

8.1、Loadrunner（未写）

8.2、Jmeter（完成）

数据库驱动

Database	Driver class	Database URL
MySql	com.mysql.jdbc.Driver	jdbc:mysql://host:port/{dbname}
PostgreSQL	org.postgresql.Driver	jdbc:postgresql:{dbname}
Oracle	oracle.jdbc.driver.OracleDriver	jdbc:oracle:thin:user/pass@//host:port/service
Ingres (2006)	ingres.jdbc.IngresDriver	jdbc:ingres://host:port/db[;attr=value]
MSSQL	com.microsoft.sqlserver.jdbc.SQLServerDriver 或者 net.sourceforge.jtds.jdbc.Driver	jdbc:sqlserver://IP:1433;databaseName=DBname 或者 jdbc:jtds:sqlserver://localhost:1433/"+"library"

1、安装对应数据库的JDBC驱动包，我这里用的是MySQL，所以驱动包名是mysql-connector-java-3.0.17-ga-bin.jar

2、将驱动包复制到jmeter/lib目录下，重启JMeter

3、在JMeter中先新建线程组，然后在此线程组下新增“配置元件-JDBC Connection Configuration”，并设置此配置元件，配置如下（如下图所示）

Database URL: jdbc:mysql://192.168.24.172:3306/abc

JDBC Driver class: com.mysql.jdbc.Driver

Username: root（数据库用户）

Password: 111111

JDBC Connection Configuration

名称: JDBC Connection Configuration

名称:

Variable Name Bound to Pool

Variable Name: abc

Connection Pool Configuration

Max Number of Connections: 10

Max Wait (ms): 10000

Time Between Eviction Runs (ms): 60000

Auto Commit: True

Transaction Isolation: DEFAULT

Connection Validation by Pool

Test While Idle: True

Soft Min Evictable Idle Time(ms): 5000

Validation Query: Select 1

Database Connection Configuration

Database URL: jdbc:mysql://192.168.24.172:3306/abc

JDBC Driver class: com.mysql.jdbc.Driver

Username: root

Password: *****

4、接着进入到MySQL服务器，在abc数据库的shell界面，输入命令：

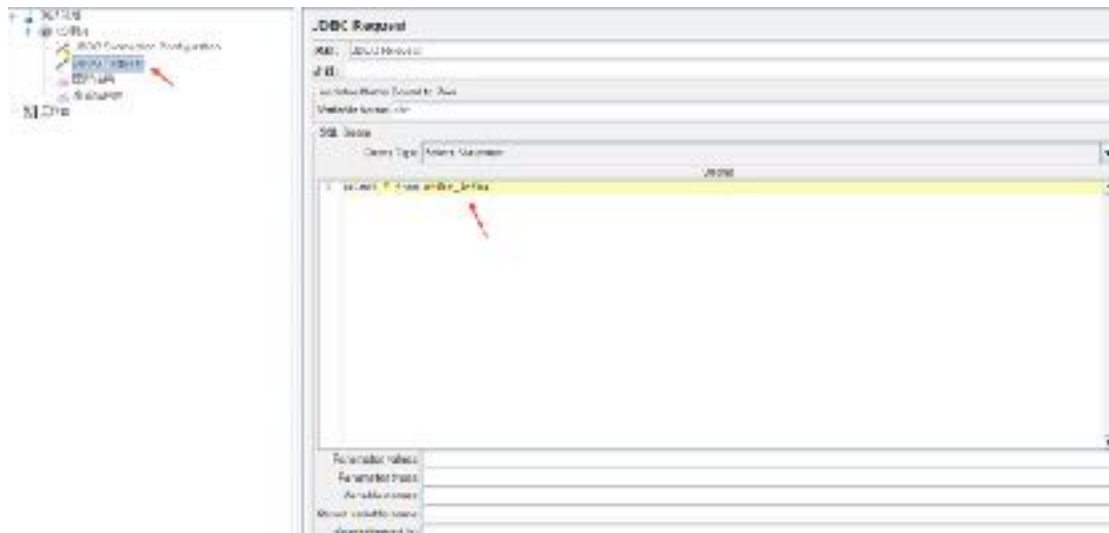
Grant select,insert,update,delete on abc.* to root@192.168.24.145 identified by '111111';

abc是数据库名，将本地IP和用户root授予访问MySQL服务器权限并且将密码设置为111111。

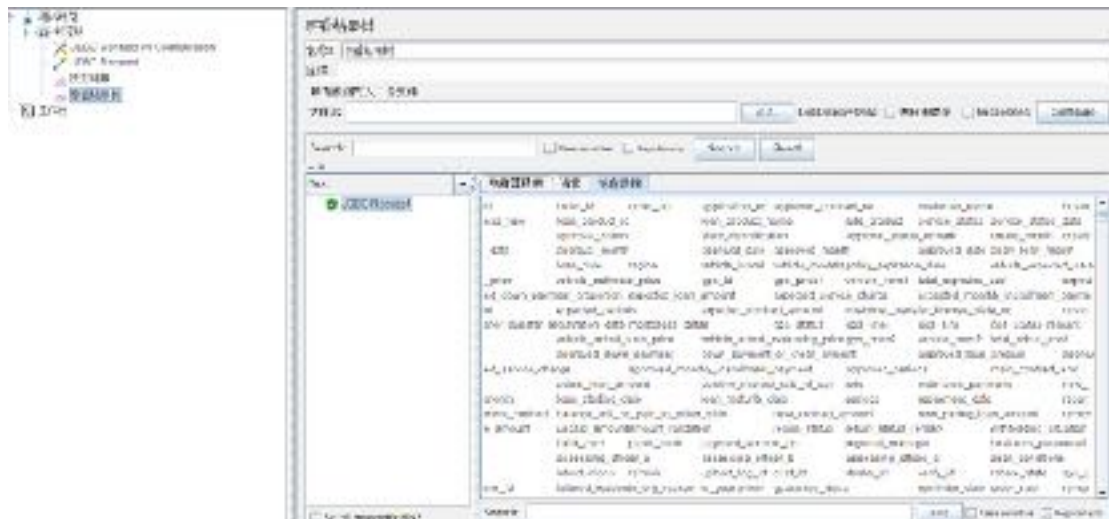
如果授权成功则显示如下图所示

```
mysql> grant select,insert,update,delete on abc.* to root@192.168.24.145 identified by '111111';
Query OK, 0 rows affected (0.14 sec)
```

5、此时重启jmeter，再在线程组下新增“Sampler-JDBC Request”，在SQL Query输入要执行的SQL语句，如下图所示



6、然后在该线程组下添加“察看结果树”和“图形结果”即可，最后执行脚本。然后再在“察看结果树”区域查看结果，如下图所示

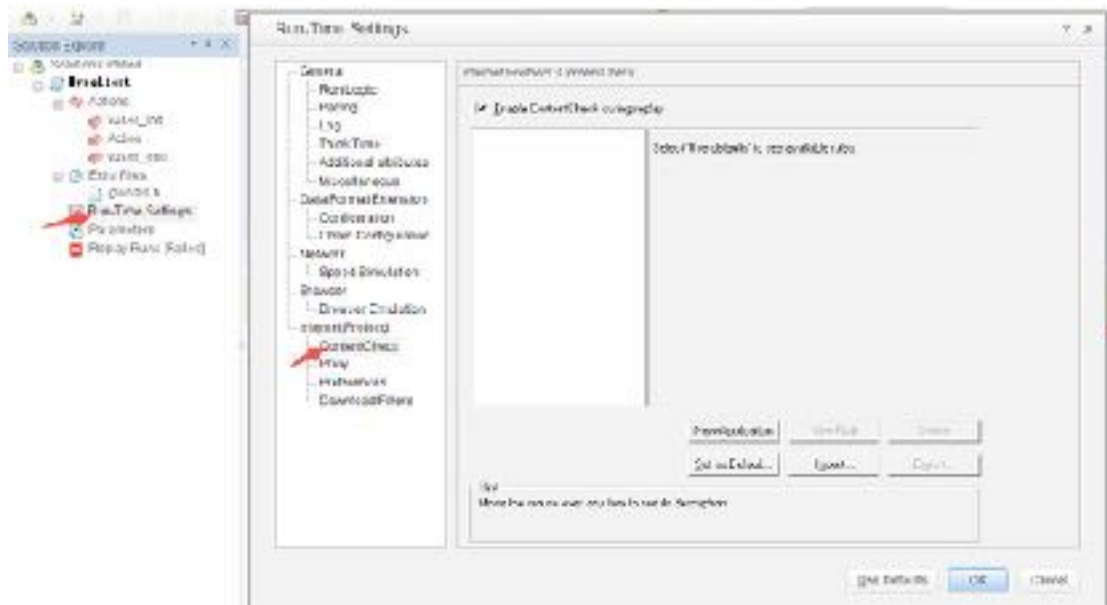


写法见jmeter附件之“附件03（JDBC）”。

9.（完成）检查点

9.1、Loadrunner

1、在Loadrunner的Generator组件中，双击打开Run-Time Settings窗口，再点开ContentCheck按钮，如下图所示



2、接着选中上图中的**Internet Protocol:ContentCheck**选项（如上图所示），最后点击OK按钮

3、接着可以在录制时点击添加检查点按钮（如下图所示），完成录制。



注意：不知道为什么我录制脚本时，点这个按钮LR就挂了。所以我都是录制完成后在脚本中手动写检查点函数。

4.现在打开录制的脚本（录制时未手动添加检查点）发现，init、action、end脚本中都有检查点函数（**web_reg_find**）。去掉不需要的检查点然后在修改检查内容，有两种方法修改检查对象

检查点函数放在请求函数上方，不然脚本会报错：找不到检查点。

A、直接修改**web_reg_find**函数的属性**Text**（这里是文本检查点，其他检查点方式请自行百度），比如我这里需要判断登录成功后是否有“常用模板”信息出现（如下图所示）



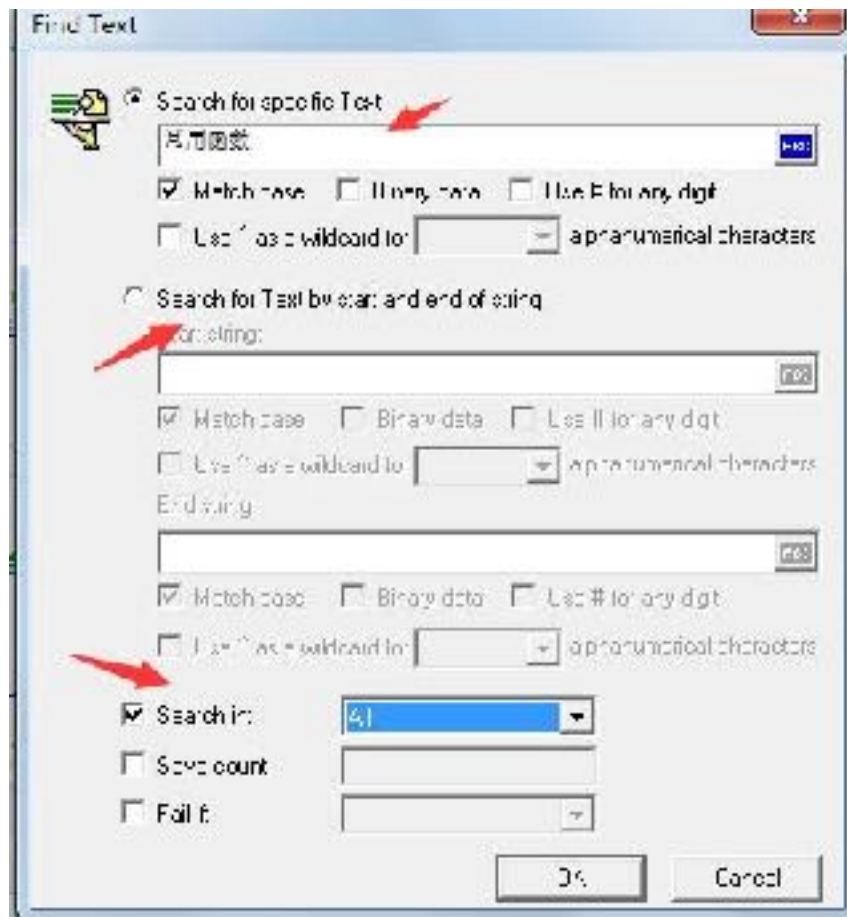
函数写法如下

```
web_reg_find("Text=常用函数",  
    LAST);  
web_submit_data("login",  
    "Action=1",  
    "Method=POST",  
    "Referer=",  
    "Content-Type=text/html",  
    "Cookie=...",  
    "Encoding=UTF-8",  
    "Data=...");
```

注意：**web_reg_find**函数要放在请求函数上方，否则在回放脚本时会报错提示找不到检查对象。

B、另外一种方法，是在请求函数上方点击鼠标右键，在点击insert-new steps，接着在右侧搜索输入框中输入web_reg_find，再点击此函数，然后在弹出的窗口中输入要进行检查的TEXT内容（如下图所示），完成后的函数内容如下所示

```
web_reg_find("Search=All", //在所有位置进行搜索查询  
    "Text=常用模板", //检查对象  
    LAST);
```



在上图中有2种查询对象，一种是TEXT，另外一种字符串（这种方法我没试过）。

5.最后执行脚本，绿色代表成功。

9.2、Jmeter

在使用Jmeter进行接口测试时，有的时候不好确认返回值是否正确或者调用接口是否成功，这种情况下我们就可以使用检查点来进行验证。另外调用接口有时会返回Success或者fail字符串，有的时候却是一个动态。这种情况下，我们还可以结合正则表达式提取器和响应断言来做验证。下面我将返回值分为固定值（Success或者fail字符串）和动态值（返回值是随机生成）来进行讲解。

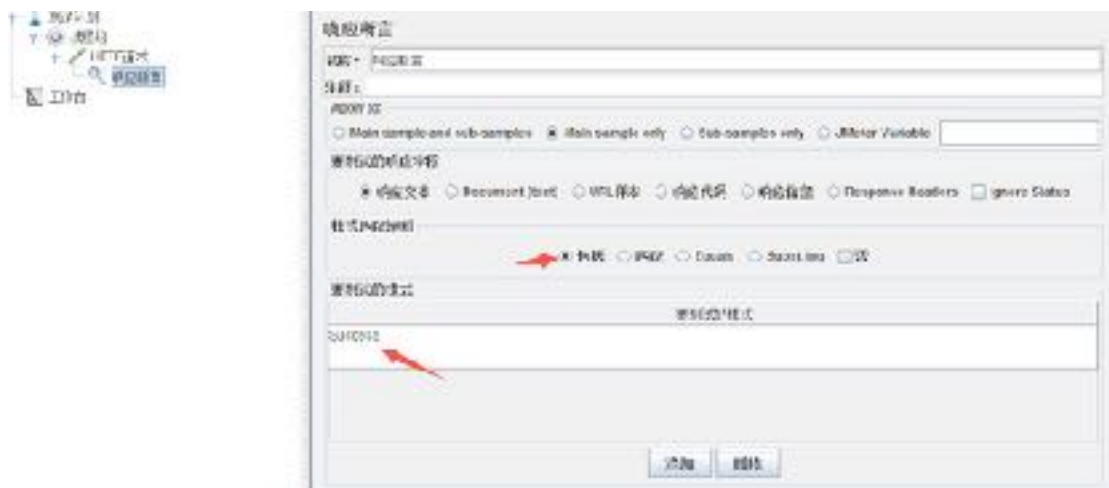
9.2.1、固定值

现在调用一个接口，如果调用成功就返回Success否则返回fail。具体操作如下：

- 1、在请求接口下方点击鼠标右键再点击“添加”-“断言”-“响应断言”（看下图）



2. 在“响应断言”界面中，“模式匹配规则”选择为“包括”，“要测试的模式”区域点击添加按钮后输入Success或者fail（不能同时输入），如下图所示



3. 现在在添加一个“断言结果”监听器，最后运行脚本。

9.2.2、动态值

调用接口时返回的图片名称是随机生成的，现在需要检查是否成功的返回来一个随机名称的图片。返回值是{"msg": "\u6210\u529f", "code": 0, "detail": {"image_path": "images/regist/

20160918/img1474190825.9.jpg"}}。"img1474190825.9.jpg"就是随机生成的图片名称，由于此返回值不是一个固定值，所以不能使用上一章节介绍的检查方法（用的话肯定报错）。此时我们可以使用正则表达式提取器将返回值保存起来，然后去运行脚本，如果能符合正则规则的字符串就表示调用接口成功。具体操作如下

1、首先在请求接口下方添加正则表达式提取器，然后设置正则规则（变量名称是\${token}），如下图所示

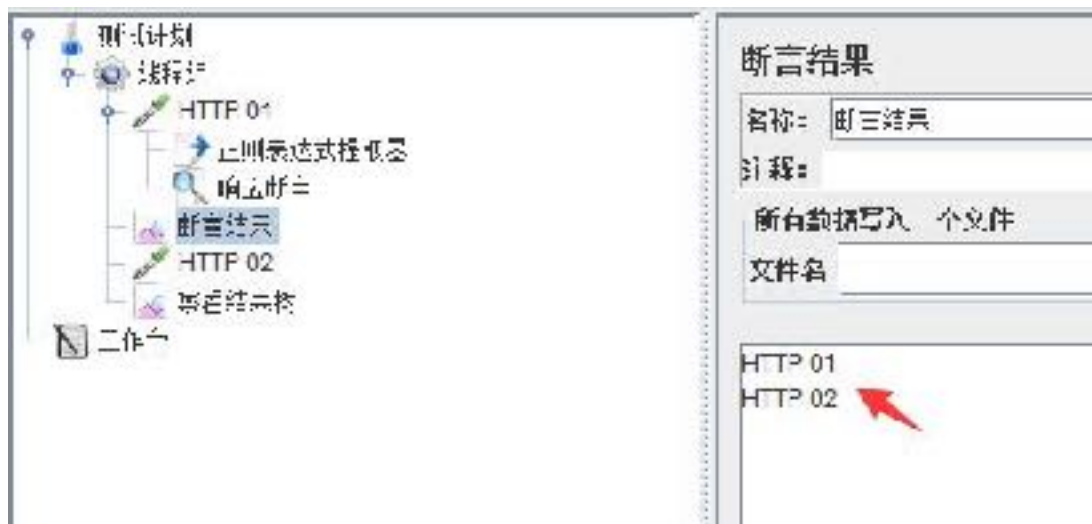


注意：正则表达式的具体写法请自行学习。

2、接着在HTTP01接口下方添加“响应断言”（添加-断言-响应断言），在“要测试的模式”区域输入正则表达式提取器的变量名称：\${token}，其他默认，如下图所示



3、现在可以在HTTP01请求下方添加一个“断言结果”（添加-监听器-断言结果）来检查断言是否成功。现在运行脚本，如果返回值正确，断言结果显示如下图所示



如果返回错误，断言结果显示为如下图所示



写法见Jmeter附件之“附件07 动态检查点”

10.（部分完成）JAVA性能测试

10.1、Java Vuser（--未写--）

Loadrunner自带的用于Java程序的性能测试。

10.2、BeanShell Sampler

针对JAVA程序的性能测试可以直接在Jmeter中调用JAR再编写JAVA测试脚本进行性能测试。使用的工具是BeanShell Sampler。

10.2.1、例子1（固定值）

本文档以MD5加密算法进行讲解。

1. 首先在eclipse中新建一个MD5类，代码如下

```
package md5; //包名
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
public class md5test { //类名
    public static String getMd5(String plainText) { //方法名
        try {
            MessageDigest md = MessageDigest.getInstance("MD5");
            md.update(plainText.getBytes());
            byte b[] = md.digest();
            int i;
            StringBuffer buf = new StringBuffer("");
            for (int offset = 0; offset < b.length; offset++) {
                i = b[offset];
                if (i < 0)
                    i += 256;
                if (i < 16)
                    buf.append("0");
                buf.append(Integer.toHexString(i));
            }
            //32位加密
            return buf.toString();
            // 16位的加密
            //return buf.toString().substring(8, 24);
        } catch (NoSuchAlgorithmException e) {
            e.printStackTrace();
            return null;
        } } }
}
```



2. 从eclipse将上面的clasee导出为JAR包

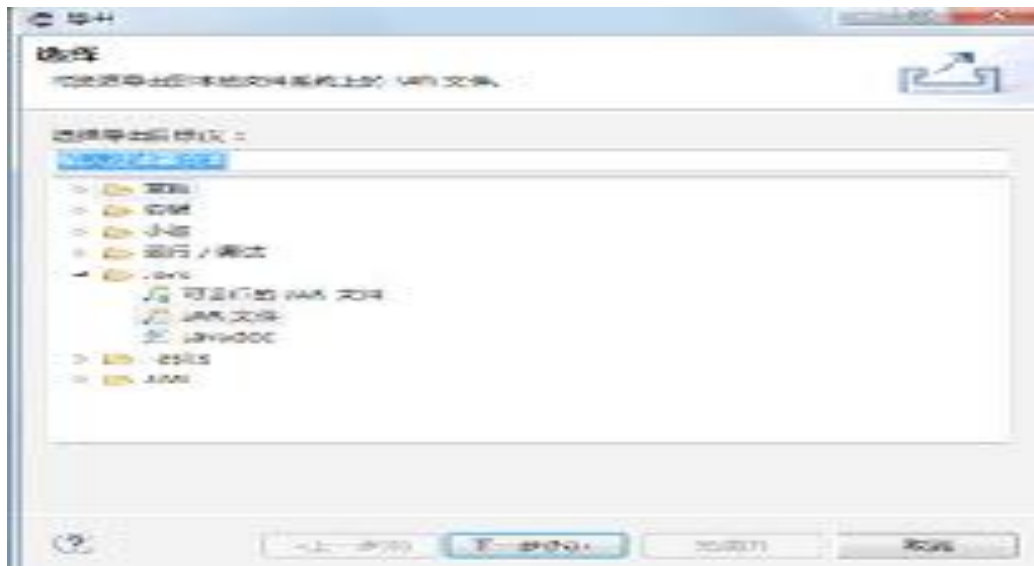


图1



起个JAR名字并保存起来

3.将导出来的JAR包存放到Jmeter\lib文件夹中

4.接着启动Jmeter并在“线程组”下面添加“Sampler-BeanShell Sampler”

5.然后在“BeanShell Sampler”的“Script”区域输入以下代码内容

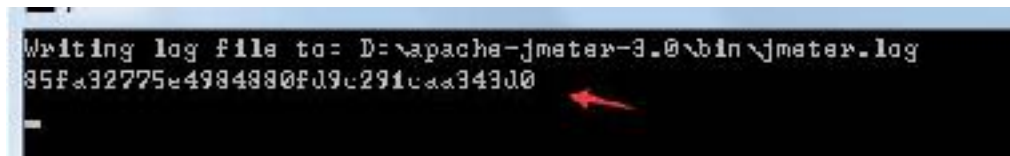
```
import md5.md5test;
```

```
String pass = md5test.getMd5("llslksks");
```

//md5test是类名，getMd5是方法名，llslksks是要加密的字符串

```
System.out.println(pass);
```


6.现在可以运行Jmeter了，加密后的结果显示在Jmeter的CMD中，如下图所示



```
Writing log file to: D:\apache-jmeter-3.0\bin\jmeter.log
85fa32775e4984880f09c291caa343d0
```

以上就是加密后的结果。

写法见Jmeter附件之“附件05-2（BeanShell）”，“附件05-1是MD5算法的JAR包”。

10.2.2、例子2（动态值）

有些时间MD5加密的原始值都是动态值，每个不一致，那么我们就需要先获取原始值（用正则表达式），然后再将原始值传到BeanShell的getMD5方法中。具体操作如下所示

- 1、需要一个GET方法的获取动态值的接口
- 2、对GET接口获取的值进行正则表达式提取，自定义一个变量token
- 3、在BeanShell脚本中将上面的md5test.getMd5("llslksks")中的“llslksks”换成\${token}，写法：md5test.getMd5("\${token}")
- 4、运行脚本，在CMD中即可看到结果。

写法见Jmeter附件之“附件05-3（BeanShell+正则表达式）”。

11.（持续中）常用函数（loadrunner）

11.1、lr_eval_string

作用：返回脚本中的一个参数当前的值，返回值类型：char

写法：lr_eval_string("{参数名}")

例如：关联函数{str}，要想获得这个{str}的当前值就如下书写：

lr_eval_string("{str}") ;

或者先定义一个字符串变量 str然后将{str}值赋予str，写法如下

char * str = lr_eval_string("{str}") ;

写法见Loadrunner附件中的脚本“lr-script-02”。

11.2、lr_output_message

作用：打印结果，有两种写法

A、直接打印参数结果：`lr_output_message("{参数名}")`；

B、赋值给变量后打印结果，写法如下

```
char * str;  
str = lr_eval_string("{参数名}");  
lr_output_message("XX参数是%s",str);
```

注意：

A、在VuGen中默认使用{}的字符串称为参数，参数必须在双引号中才能用

B、`lr_output_message()` 方法放置位置很重要，要放在获取参数值函数的后面

写法见Loadrunner附件中的脚本“lr-script-02”。

11.3、strcmp

作用：字符串比对函数

写法：`strcmp(字符串A, 字符串B)`

While判断条件如下：

```
while((strcmp(str1,str3))||(strcmp(str2,str3))){} 
```

写法见LR附件中的脚本“lr-script-03”中的注释。

12.（部分持续中）工具设置

12.1、Loadrunner（--未写--）

12.1.1、多机负载设置

12.2、Jmeter（持续中）

12.2.1、线程组设置

Jmeter在做并发时很消耗测试机内存，为了降低对测试机的内存负担，我们可以设置线程组的创建方式。使用线程组中的“Delay Thread Creation until needed”选项，如下图所示



中文翻译：延迟线程创建，直到需要。

例子1：300线程，并发50

①不设置Delay Thread Creation until needed，平均TPS和响应时间分别是3769.6和76ms，失败率14332（0.63%），其中线程池耗尽错误856次；

②设置Delay Thread Creation until needed，平均TPS和响应时间分别是3680.0和78ms，错误率11089（0.52%），线程池耗尽错误162次；

例子2：50线程，并发50

①不设置Delay Thread Creation until needed，平均TPS和响应时间分别是815.1和

57ms，，失败率0（0.00%），其中线程池耗尽错误0次；

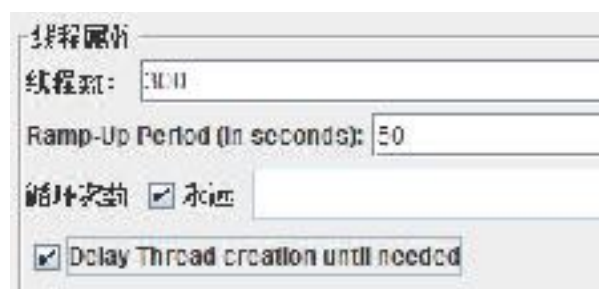
②设置Delay Thread Creation until needed，平均TPS和响应时间分别是936.4和50ms，失败率0（0.00%），其中线程池耗尽错误0次；

综上所述，当虚拟用户数比较大时而测试机内存较小时，可以选中此选项来降低对测试机的压力，得到有效的测试数据。

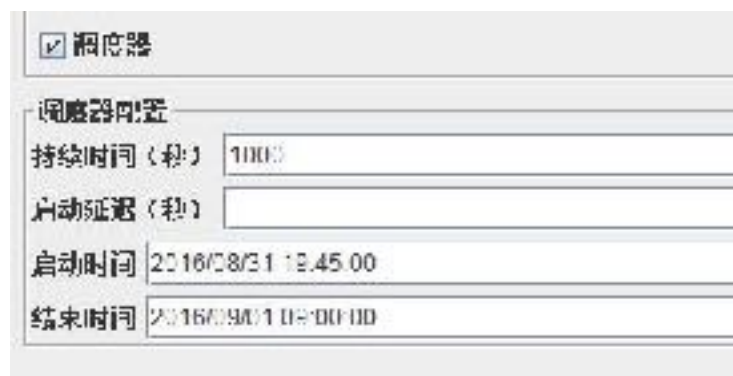
12.2.2、并发设置

在Jmeter中进行并发压力测试需要集合点和脚本运行方式。集合点的设置请参照本文档的第6.2.1章节。脚本运行方式是在“线程组”中进行设置，具体操作如下：

A、在线程组属性设置区域，输入“线程数”和“Ramp-Up Period（in seconds）”值再选中“循环次数”后面的“永远”选项，如下图所示



B、接着选中“调度器”选项，然后在“调度器配置”区域中输入“持续时间（秒）”或者设置“启动时间”和“结束时间”，如下图所示



启动时间:测试计划什么时候启动，启动延迟会覆盖它。当启动时间已过，手动运行脚本时当前时间也会覆盖它（但启动时间页面显示不会变）；

结束时间：测试计划什么时候结束，持续时间会覆盖它；

持续时间（秒）：测试计划持续多长时间，会覆盖结束时间；

启动延迟（秒）：测试计划延迟多长时间启动，会覆盖启动时间。

多机负载设置

12.2.3、多机负载测试