

性能测试

--性能测试工具LoadRunner使用—参数化

内容回顾

■ LoadRunner开启

■ 选择协议

- 协议分析
- 协议确定
- 单协议脚本/多协议脚本

内容回顾

— 录制选项设置

- **HTML-based script**

- 基于 **HTML** 的脚本级别：为每个 **HTML** 用户操作生成单独的步骤和函数。
步骤直观且脚本容易理解和维护

- **URL-based script**

- 基于 **URL** 的脚本级别：录制“客户端向服务器发送请求后，服务器返回给客户端的所有浏览器请求和资源”。它自动将每个 **HTTP** 资源（即所有操作）录制为 **URL** 步骤（即由 `web_url` 语句构成的脚本）。

较上一方式，记录了更详细的客户端操作信息，甚至可捕获非 **HTML** 形式应用程序，如小程序、非浏览器程序。但生成的脚本内容长且多，显示不直观

内容回顾

- `web_url()`
- `web_submit_data()`
- `web_custom_request()`

内容回顾

■ Run-time Settings

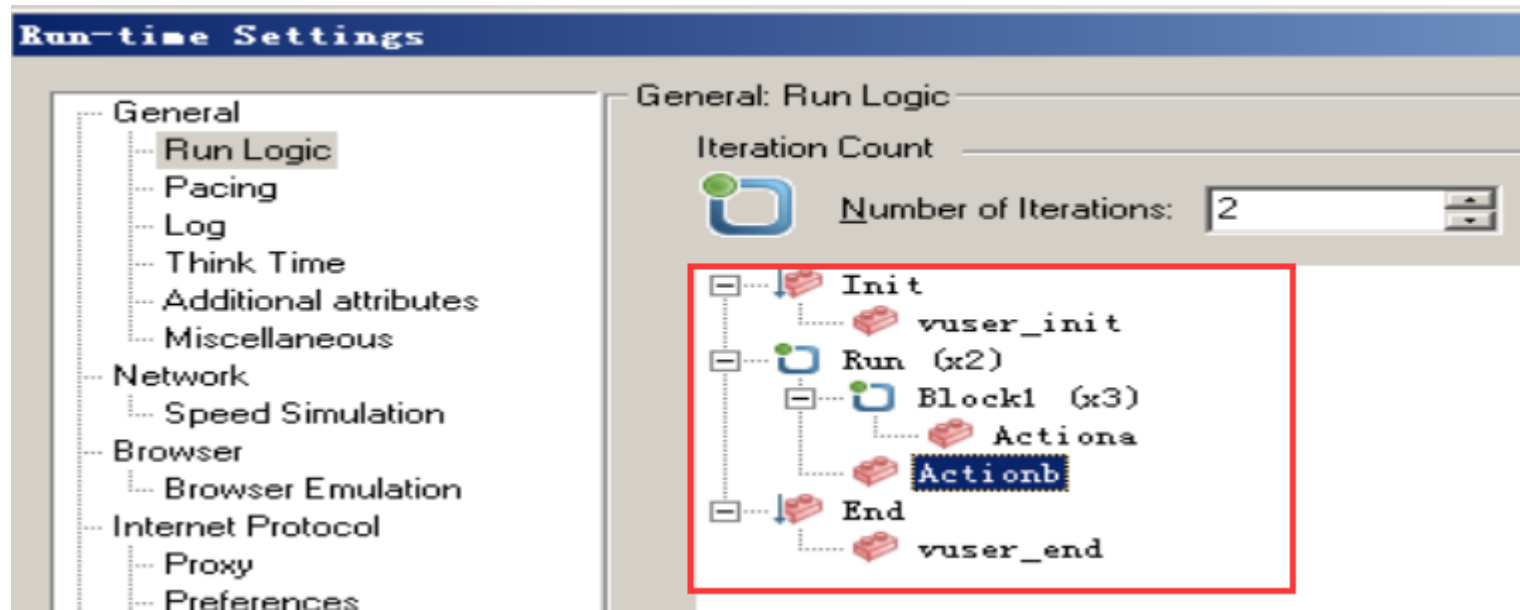
- Run Logic
- Pacing
- Log
- Think time
- Speed Simulation
- Browser Emulation

内容回顾—代码逻辑

■ 思考：

- Actiona中的代码： `lr_output_message("A");`
- Actionb中的代码： `lr_output_message("B");`
- 设置代码逻辑如下
- 请问：运行结果是什么？

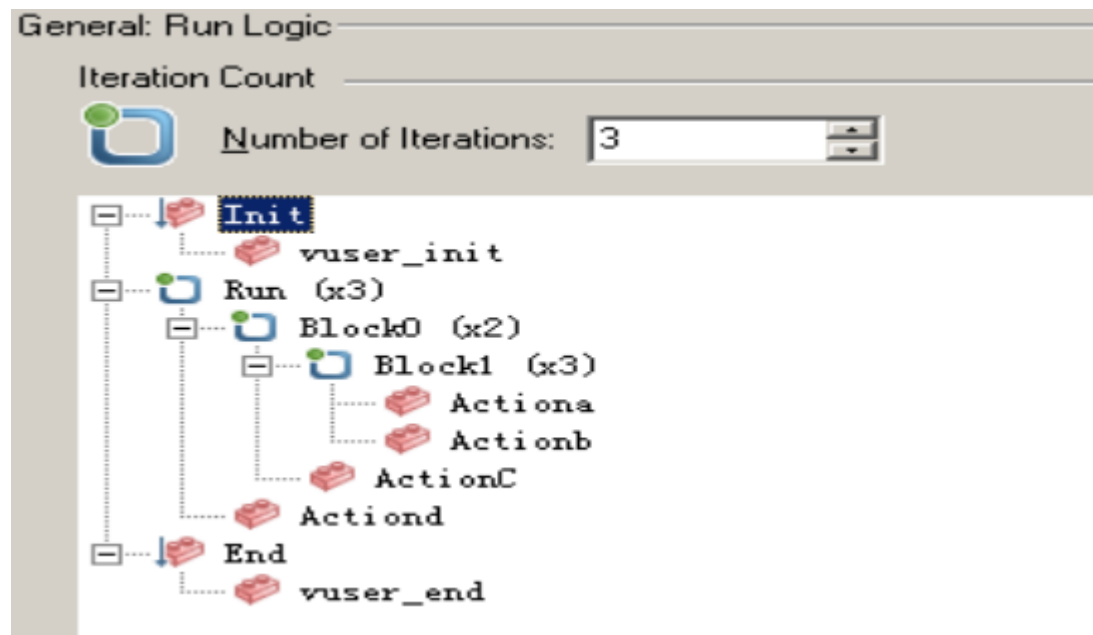
A A A B A A A B



内容回顾—代码逻辑

■ 思考：

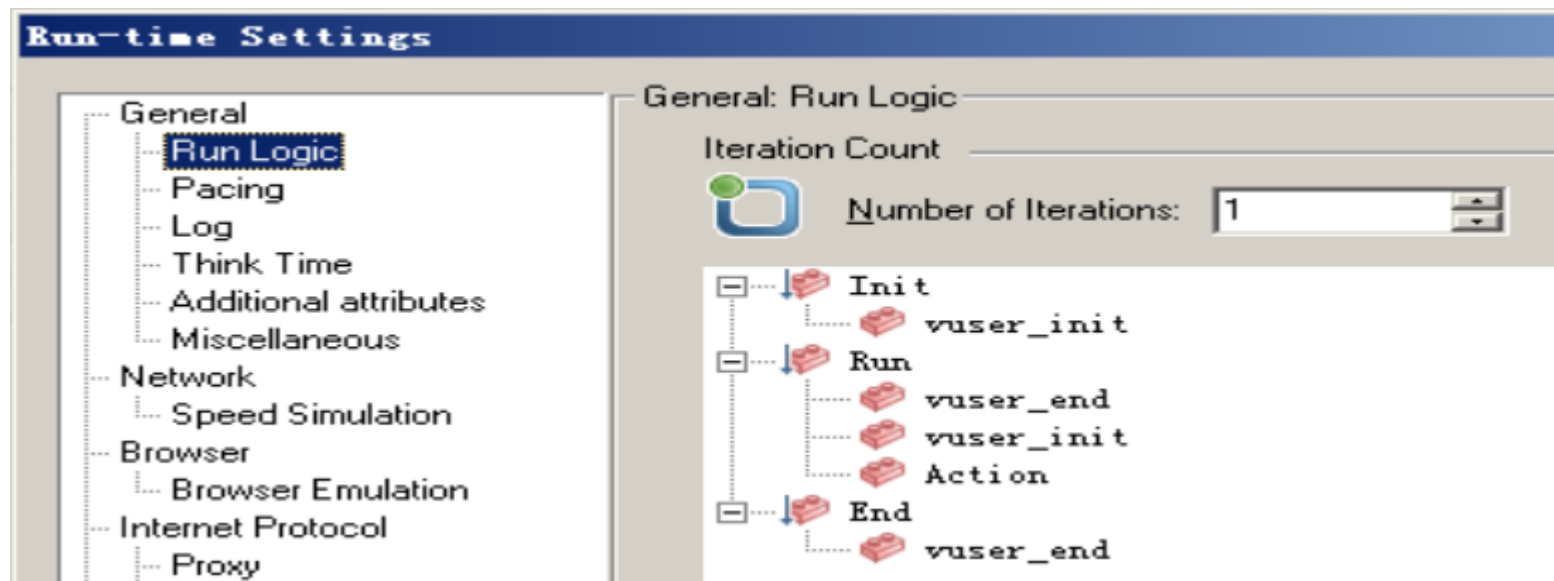
- Actiona, Actionb, Actionc, Actiond 分别是输出 A, B, C, D, 代码逻辑如下, 请问输出结果是什么



- A B A B A B C A B A B A B C D (3遍)

内容回顾—代码逻辑

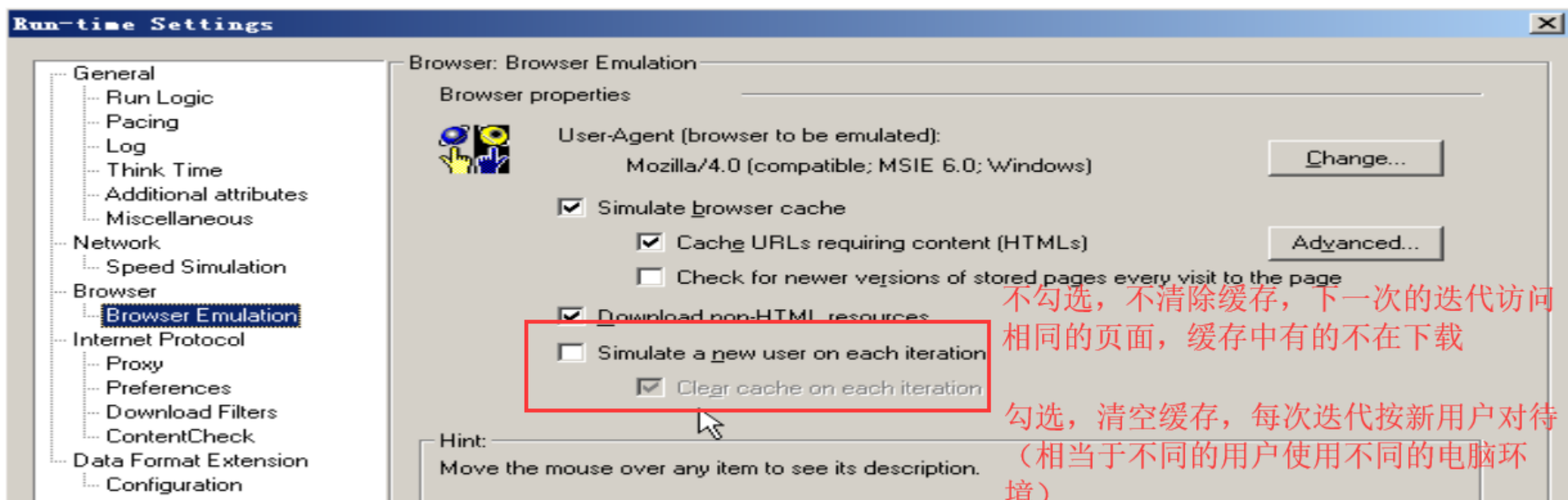
- 代码执行顺序一定是vuser_init,action, vuser_end吗?
- 如下逻辑, 代码的执行顺序是怎样的?



内容回顾

■ Browser Emulation

— 浏览器仿真



补充知识：函数和变量

■ 函数类型说明：

- web:基于HTTP协议（与协议相关的函数）
- lr:通用函数
- lr. :java或.net中使用的函数
- 自定义函数

补充知识：函数和变量

■ 局部变量和全局变量

- 在init、action、end中定义的变量就是局部变量
- 在globals.h中定义的变量是全局变量

■ 什么时候定义全局变量

- 整个过程中固定不变的，例如URL地址、KEY、其他
- 支持指针、数组、控制流等

思考

- 如果模拟50名用户登录订票系统，每人登录名称是否相同？
- 每人订票信息（出发地、目的地、出发日期、航班信息）是否相同？
- 所有属于个人的信息都不相同，怎么办？

——参数化

目录

- 什么是参数化
- 为什么进行参数化
- 怎样进行参数化

什么是参数化

■ 参数化

- 用参数替代常量，可更加真实的模拟实际用户操作并简化脚本

目录

- 什么是参数化
- 为什么进行参数化
- 怎样进行参数化

为什么进行参数化

- 每次使用数据不同时，参数化更方便脚本执行

目录

- 什么是参数化
- 为什么进行参数化
- 怎样进行参数化

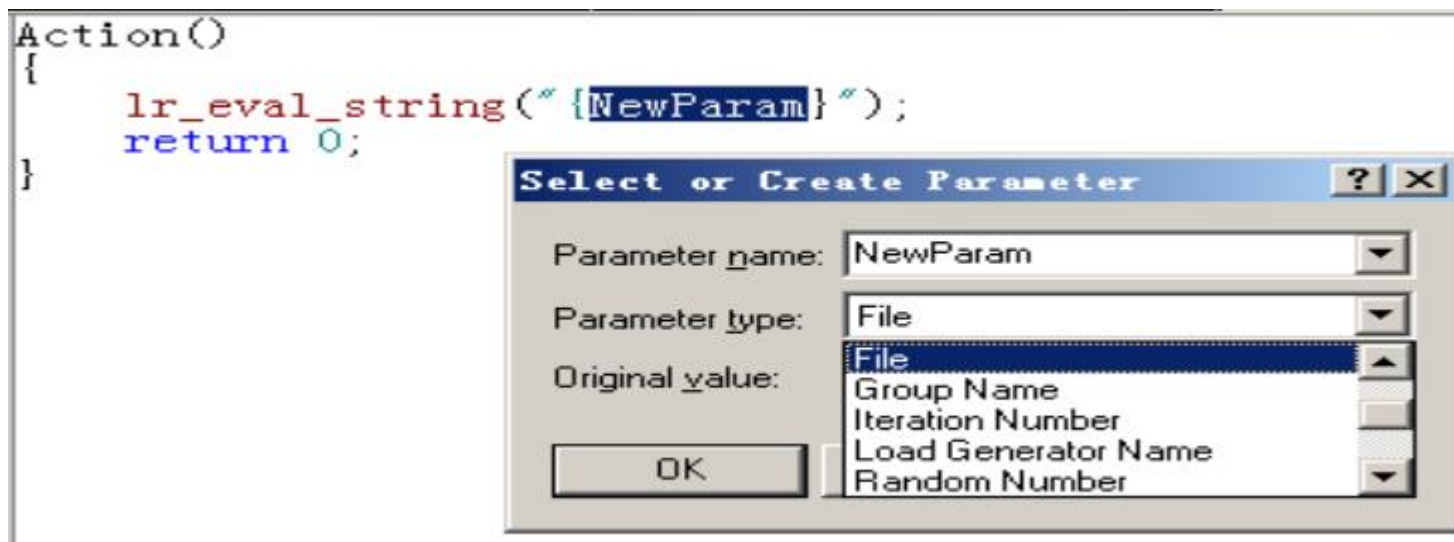
怎样进行参数化

■ 脚本参数化的两种方式

- 鼠标右键参数化方式：先替换常量再建参数化列表
- 建参数化列表方式：先建参数化列表再替换常量

怎样进行参数化

1 替换常量



Parameter name:参数名称，一般起通俗易懂的名称

Parameter type:参数类型

怎样进行参数化

■ 参数名称

```
lr_think_time(9);

web_submit_data("user-login-L3plbnRhby93d3cv.html_2",
    "Action=http://192.168.56.1:9090/zentao/www/user-login-L3plbnRhby93d3cv.html",
    "Method=POST",
    "TargetFrame=",
    "RecContentType=text/html",
    "Referer=http://192.168.56.1:9090/zentao/www/user-login-L3plbnRhby93d3cv.html",
    "Snapshot=t2.inf",
    "Mode=HTML",
    ITEMDATA,
    "Name=account", "Value={account}", ENDITEM,
    "Name=password", "Value={password}", ENDITEM,
    "Name=referer", "Value=/zentao/www/", ENDITEM,
    LAST);

web_url("www",
    "URL=http://192.168.56.1:9090/zentao/www/",
    "TargetFrame=",
    "Resource=0",
    "RecContentType=text/html",
    "Referer=",
    "Snapshot=t3.inf",
```

怎样进行参数化

■ 参数类型

- **DateTime**: 需要输入日期/时间的地方, 使用**DateTime**来替代。选择一种格式或定制格式即可
- 用**Vuser**组的名称替换参数。创建方案时, 要指定**Vuser**组的名称, 否则运行**VuGen**的脚本时, 组名始终为“无
- **Load Generator Name**: 在实际运行中, **LoadRunner** 使用该虚拟用户所在**Load Generator** 的机器名来代替
- **Iteration Number**: 用当前的迭代编号替换参数, 迭代编号的格式可以自己设置

怎样进行参数化

- **Random Number:** 随机数。在属性设置中可以设置产生随机数的范围
- **Unique Number:** 用一个随机生成的整数替换参数，可以通过指定最小和最大值，设置随机数的范围
- **Vuser ID:** 使用该虚拟用户的ID来代替参数值，该ID由Controller来控制。在VuGen中运行脚本时，VuGen将会是-1

怎样进行参数化

- **File:**可以在参数属性中编辑参数文件，也可以直接选择已编辑好的参数文件，还可以从现成的数据库中提取，这是最常用的一种参数化方式
- **User Defined Function:** 从用户开发的dll 文件提取数据

怎样进行参数化—基本解释

Parameter List

.....<D> NewParam

参数列表

Parameter type: File 参数类型

File NewParam.dat Browse...

Add Column... Add Row... Delete Column... Delete Row...

NewParam	
1	Value
2	value2

Edit with Notepad... Data Wizard... Simulate Parameter...

Select column

☒ By number: 1

☐ By name:

选择数据的方式

Select next row: Sequential

Update value on: Each iteration

When out of values: Continue with last value

Allocate Vuser values in the Controller

☒ Automatically allocate block size

☐ Allocate values for each Vuser

怎样进行参数化—基本解释

Edit with Notepad... Data Wizard... Simulate Parameter...

Select column:
☒ By number: 1
☐ By name:

File format:
Column: Comma
First data: 1

Select next row:
Update value on:
Sequential
Random
Unique

When out of values:
Continue with next value

Allocate Vuser values in the Controller:
☒ Automatically allocate block size

Select column:
☒ By number: 1
☐ By name:

File format:
Column: Comma
First data: 1

Select next row:
Update value on:
Sequential
Each iteration
Each iteration
Each occurrence
Once

When out of values:
Continue with next value

Allocate Vuser values in the Controller:
☒ Automatically allocate block size

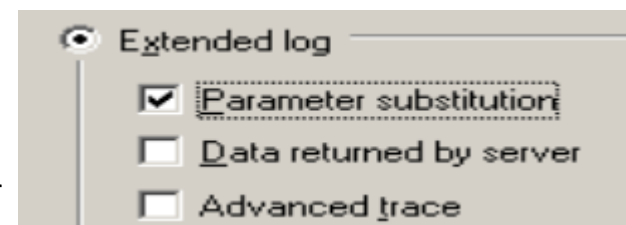
怎样进行参数化—基本使用

■ 实践一：

1 新建脚本，Action中写入：`lr_eval_string("{NewParam}")`;

说明：`lr_eval_string`:返回字符串参数值

2 将运行设置中log设置中勾选 **Parameter substitution**



3 设置参数值：甲、乙、丙、丁、戊

4 Run Logic 设置3次；参数更新方式选择顺序和迭代（如下页图）

5 运行查看结果

怎样进行参数化—基本使用

Parameter List

.....<D> NewParam

Parameter type: File

File NewParam.dat

Add Column... Add Row... Delete Column... Delete

NewParam	
1	甲
2	乙
3	丙
4	丁
5	戊

Edit with Notepad... Data Wizard...

Select column:

☒ By number: 1

☐ By name:

Select next row: Sequential

Update value on: Each iteration

怎样进行参数化—基本使用

- 结果：每次迭代切换不同的参数值

```
Action.c(3): Notify: Parameter Substitution: parameter "NewParam" = "甲"  
Ending action Action.  
Ending iteration 1.  
Starting iteration 2.  
Notify: Next row for parameter NewParam = 2 [table = NewParam].  
Notify: Getting new value for parameter 'NewParam': table = 'NewParam.dat' column = '0' row = '  
Starting action Action.  
Action.c(3): Notify: Parameter Substitution: parameter "NewParam" = "乙"  
Ending action Action.  
Ending iteration 2.  
Starting iteration 3.  
Notify: Next row for parameter NewParam = 3 [table = NewParam].  
Notify: Getting new value for parameter 'NewParam': table = 'NewParam.dat' column = '0' row = '  
Starting action Action.  
Action.c(3): Notify: Parameter Substitution: parameter "NewParam" = "丙"
```

怎样进行参数化—基本使用

■ 实践二：将上述实例设置迭代5次，查看结果

- 结果：代码迭代5次，每次获取一个参数的值，得到的值分别是甲、乙、丙、丁、戊

■ 实践三：将上述实例设置迭代6次，查看结果

- 结果：甲、乙、丙、丁、戊、甲（当迭代数据不够时，回到第一个数据开始取）

怎样进行参数化—基本使用

- 实践四：在Action中写如下代码，并在run logic 中设置迭代3次，查看结果

```
lr_eval_string({param});
```

```
lr_eval_string({param});
```

- 结果：甲甲、乙乙、丙丙
- 为什么不是甲、乙、丙、丁、戊、甲

* 每次触发迭代才更新数据，不迭代不更新数据

怎样进行参数化—代码逻辑+参数

- 实践5：在Action中写如下代码，在Run Logic 中设置迭代次数：2，
查看结果

```
int i;  
for(i = 0;i < 10;i++){  
    lr_eval_string("{NewParam}");  
    lr_eval_string("{NewParam}");  
}
```

return 0;

- 结果：20个甲，20个乙
- 为什么？
- 原因：只有Action迭代时，数据才更新

怎样进行参数化—代码逻辑+参数

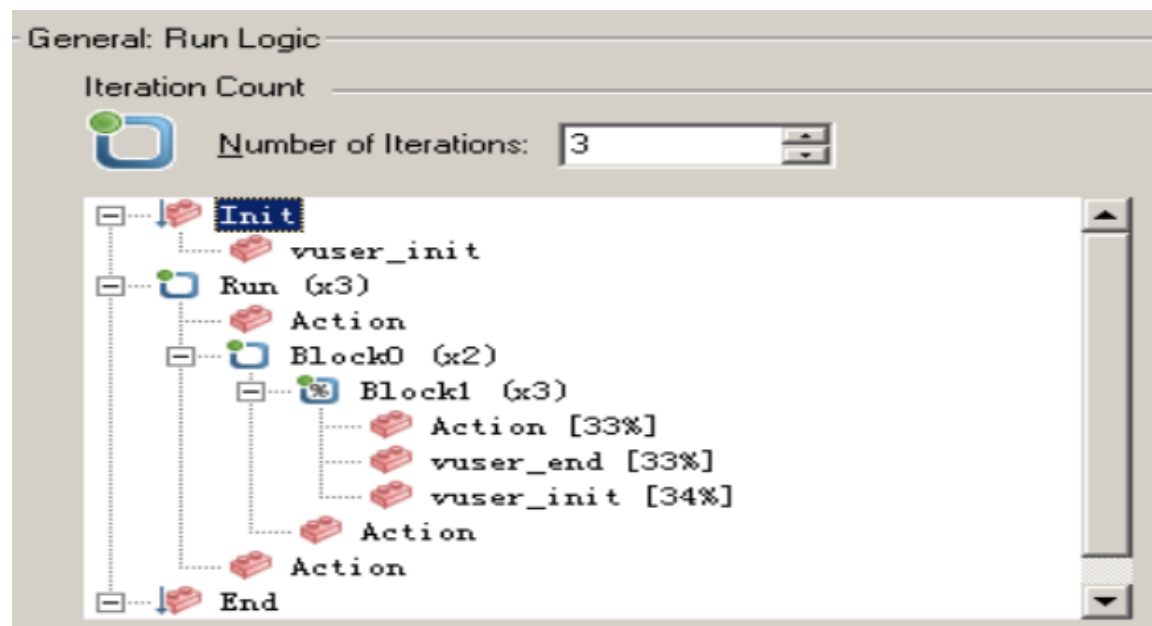
- 实践6: Action 内什么都不写, 设置迭代3次; vuser_init和vuser_end中分别写: lr_eval_string("{New Param}"); 查看运行结果
 - 结果: 甲、丙
 - 为什么?
 - 第一次执行时, 打印了甲, 执行Action, 第二次执行Action时, 参数为乙, 第三次执行时, 参数为丙, 接着执行vuser_end, 打印出丙

怎样进行参数化—代码逻辑+参数

- 实践7-1：在Action中写如下代码，init和end中为空，在Run Logic中设置如图，最后打印的参数是什么？

```
int i;  
for(i = 0;i < 10;i++)  
    lr_eval_string("{NewParam}");  
lr_eval_string("{NewParam}");  
return 0;
```

— 最后打印的是：丙

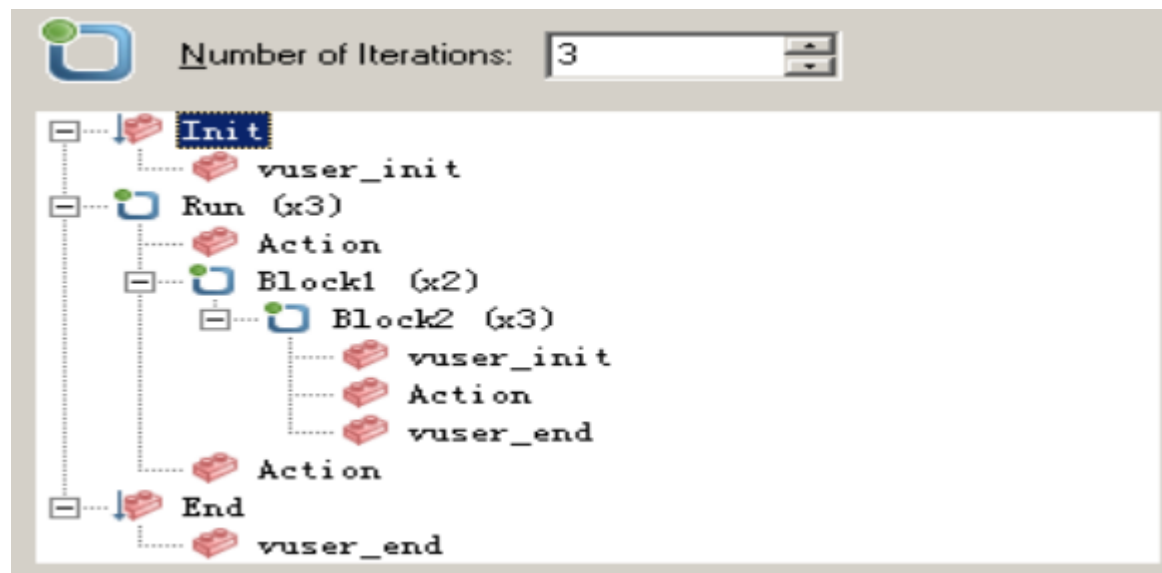


怎样进行参数化—代码逻辑+参数

■ 实践7-2：如果将Block1设置成Sequential，打印结果是什么？

— 结果：99个甲，99个乙，
99个丙

* 只有run迭代才更新数据



怎样进行参数化—每次取值更新

■ 实践8: action中代码如下:

```
lr_eval_string("{NewParam}");
```

```
lr_eval_string("{NewParam}");
```

vuser_init和vuser_end中为空, Run Logic中设置迭代次数: 3, 参数列表如上, 更新方式选择顺序+每次取值更新



问: 运行结果是什么

结果: 甲 乙 丙 丁 戊 甲

为什么?

原因: 每次取值更新

怎样进行参数化—每次取值更新

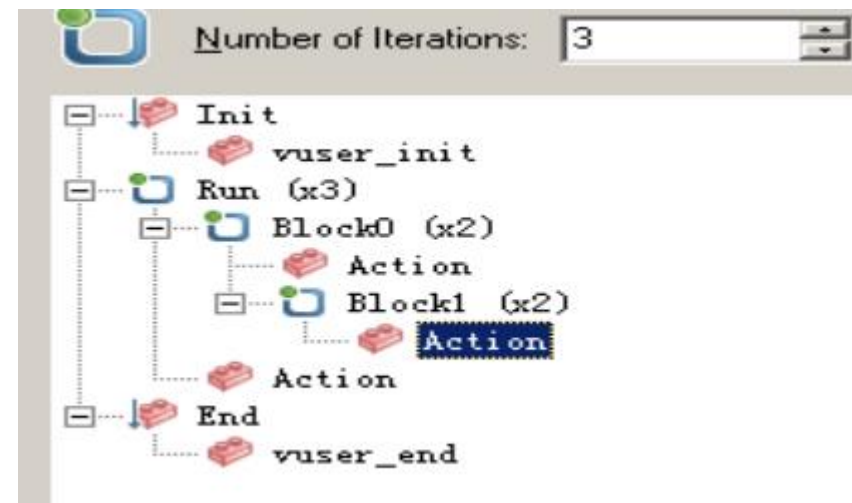
■ 实践9：代码设置如下：

- `vuser_init` 和 `vuser_end` 中代码：`lr_eval_string("{NewParam}");`
- `action` 中代码为空，设置迭代次数3，参数更新方式：顺序+每次取值更新
- 程序运行结果中输出参数值分别是什么？
- 甲 乙
- 原因：只有 `action` 迭代3次，`init` 和 `end` 只执行1次

怎样进行参数化—每次取值更新

- 实践10：在action中写如下代码，init 和end中保持默认，参数设置方式如上例，在Run Logic 中设置如图，最后打印的参数值，最后一个是什么？

```
int i;  
for(i = 1;i < 10;i++)  
    lr_eval_string("{NewParam}");  
lr_eval_string("{NewParam}");  
return 0;
```



- 最后一个参数值：戊
- 原因：参数值更新210次，更新到最后正好是戊

怎样进行参数化—唯一

- 实践11：如上逻辑都不变，将参数更新方式改为Sequential + Once，

程序执行结果是什么

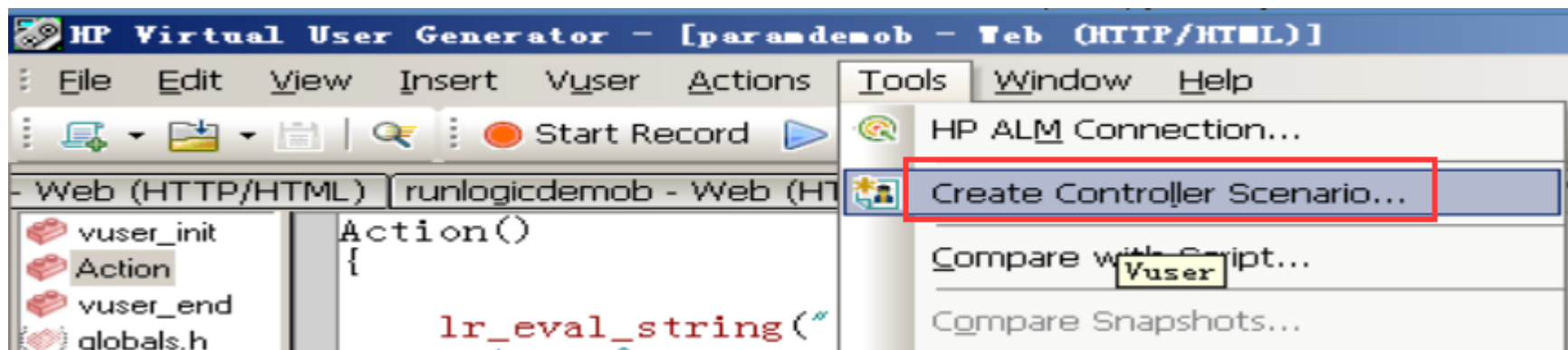
Select next row:	Sequential
Update value on:	Once

- 全部是：甲
- Once:无论程序怎么执行，参数值都不更新

怎样进行参数化—参数结合场景

■ 实践12: action中的代码: `lr_eval_string({NewParam});`

- 参数设置顺序+迭代, Run Logic设置6次
- 创建场景, 10个用户同时执行, 设置细节见下图
- 查看执行结果



怎样进行参数化—参数结合场景

Create Scenario

Select Scenario Type

☐ Goal Oriented Scenario

☒ **Manual Scenario**

Number of Vusers:

Load Generator:

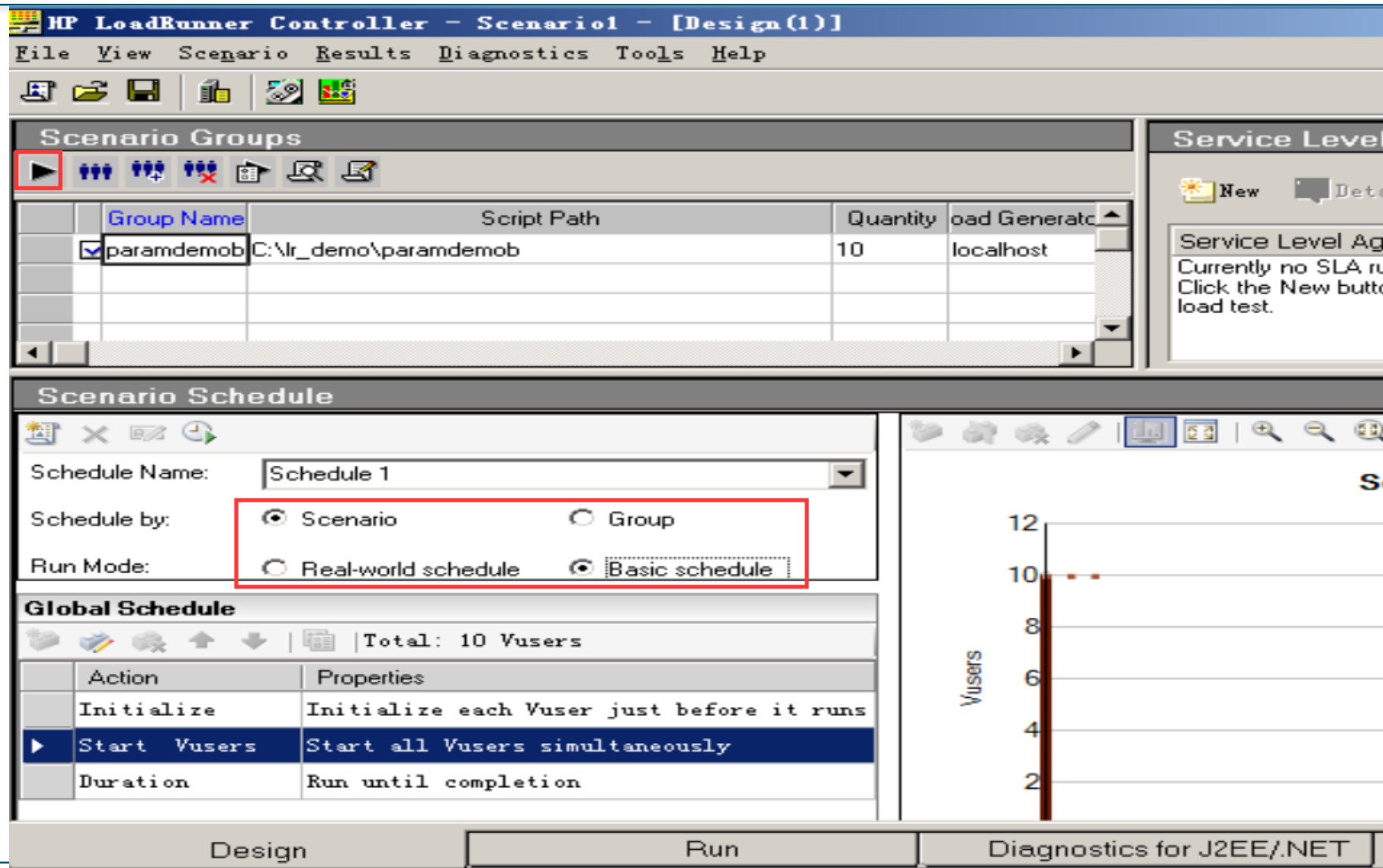
Group Name:

Result Directory:

☒ Add script to current scenario

OK Cancel

怎样进行参数化—参数结合场景



The screenshot displays the HP LoadRunner Controller interface for a scenario named "Scenario1". The interface is divided into several sections:

- Scenario Groups:** A table listing the groups used in the scenario. The "paramdemob" group is selected, with a quantity of 10 and a load generator of localhost.
- Scenario Schedule:** A section for defining the scenario's execution schedule. The "Schedule Name" is "Schedule 1". The "Schedule by" options are "Scenario" (selected) and "Group". The "Run Mode" options are "Real-world schedule" and "Basic schedule" (selected).
- Global Schedule:** A table listing the actions and their properties for the scenario.

Group Name	Script Path	Quantity	Load Generator
paramdemob	C:\lr_demo\paramdemob	10	localhost

Action	Properties
Initialize	Initialize each Vuser just before it runs
Start Vusers	Start all Vusers simultaneously
Duration	Run until completion

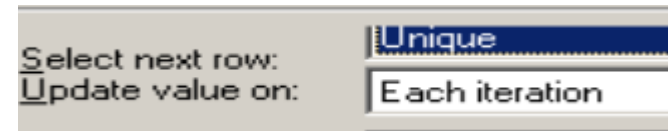
The "Global Schedule" section also shows a total of 10 Vusers. The "Design" tab is active, and the "Run" and "Diagnostics for J2EE/.NET" tabs are also visible.

怎样进行参数化—随机+迭代

- 结果：每个用户使用相同的数据，执行6次
 - 思考：这是否符合实际情况？什么样的参数更新方式更能模拟现实？
- 实践13：将参数更新方式改成随机+迭代，其他设置不变，查看结果
 - 每位用户使用的数据都是随机的
- 随机的做法，让每次读取都是随机的，这样可以保证相对离散

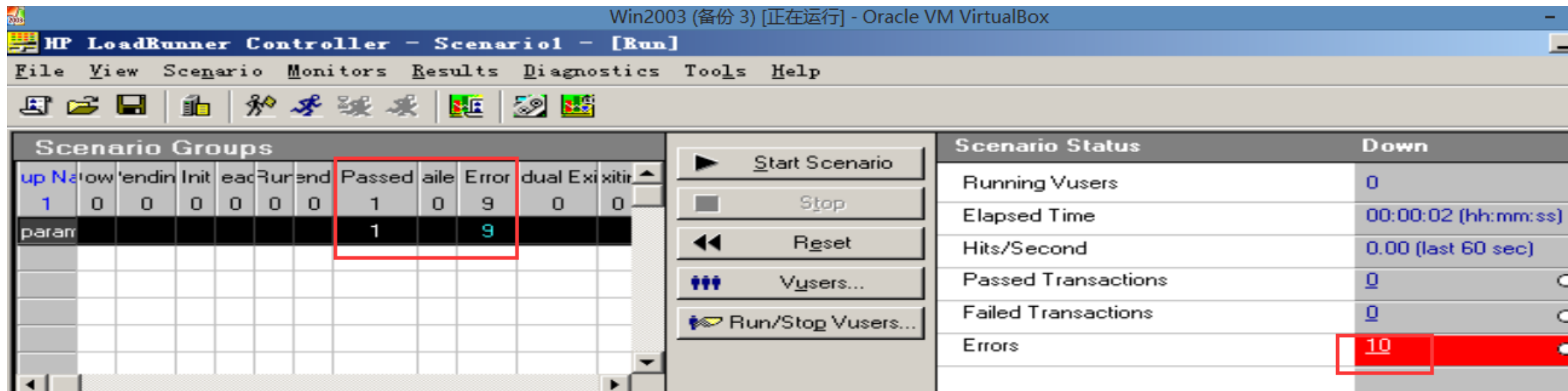
怎样进行参数化—唯一+迭代

■ 实践14：如上设置中，数据更新方式唯一+迭代



其他设置不变，运行同样的场景，查看结果

■ 结果：passed:1 ;Error:9



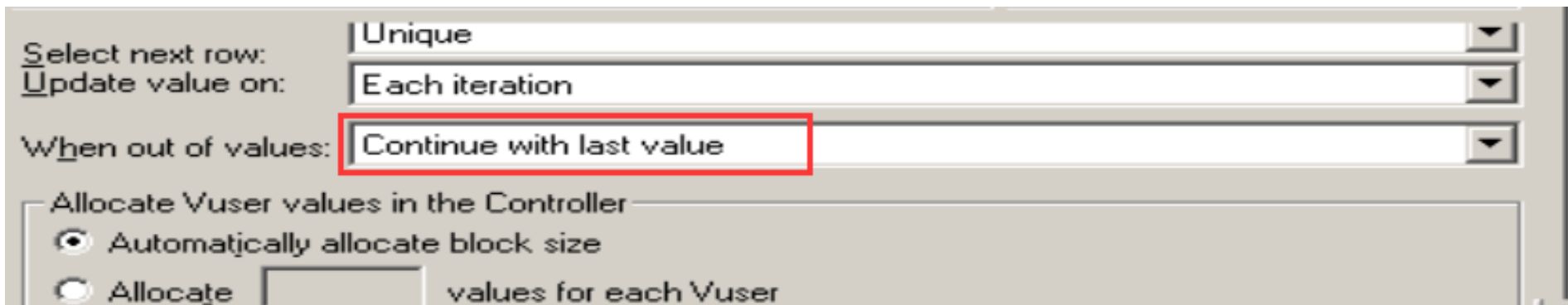
The screenshot shows the HP LoadRunner Controller interface for a scenario named 'Scenario1'. The 'Scenario Groups' table on the left displays the results of the run, with a red box highlighting the 'Passed' and 'Error' columns. The 'Scenario Status' table on the right shows the overall status of the run, with a red box highlighting the 'Errors' row.

Scenario Groups	Up	Down	Init	Each	Run	End	Passed	Fail	Error	Aborted	Exit
1	0	0	0	0	0	0	1	0	9	0	0

Scenario Status	Down
Running Vusers	0
Elapsed Time	00:00:02 (hh:mm:ss)
Hits/Second	0.00 (last 60 sec)
Passed Transactions	0
Failed Transactions	0
Errors	10

怎样进行参数化—唯一+迭代

- 原因：5个参数，6次迭代，当参数不够用时，使用最后一个参数



The screenshot shows a configuration window with the following settings:

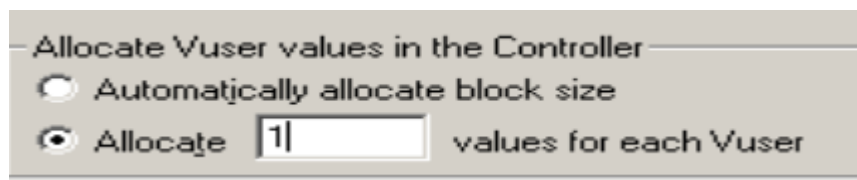
- Select next row: Unique
- Update value on: Each iteration
- When out of values: Continue with last value (highlighted with a red box)
- Allocate Vuser values in the Controller:
 - ☒ Automatically allocate block size
 - ☐ Allocate values for each Vuser

怎样进行参数化—唯一+迭代

- 实践15：将如上设置中Run Logic改成3次，其他设置不变，查看运行结果
- 结果：Passed:2 Error:8

怎样进行参数化—分配虚拟用户值方式

- 实践16：将分配虚拟用户值的方式改为 Allocate 1 values for each Vuser，其他设置不变，查看运行结果



- 结果：5个passed；5个error
- Automatically allocate block size: 自动分配参数块大小
- Allocate 1 values for each Vuser: 每个虚拟用户分配1个参数值

怎样进行参数化—两个及以上参数设置

■ 实践16：需求：使用用户名、密码登录系统

- 分析：涉及到两个以上的参数怎么做？
- 怎样让用户取得的用户名更离散（接近现实）
- 如果参数选择随机，第二个参数值怎样对应第一个参数值

怎样进行参数化—两个及以上参数设置

做法：

1 两个参数指向同一个参数值文件

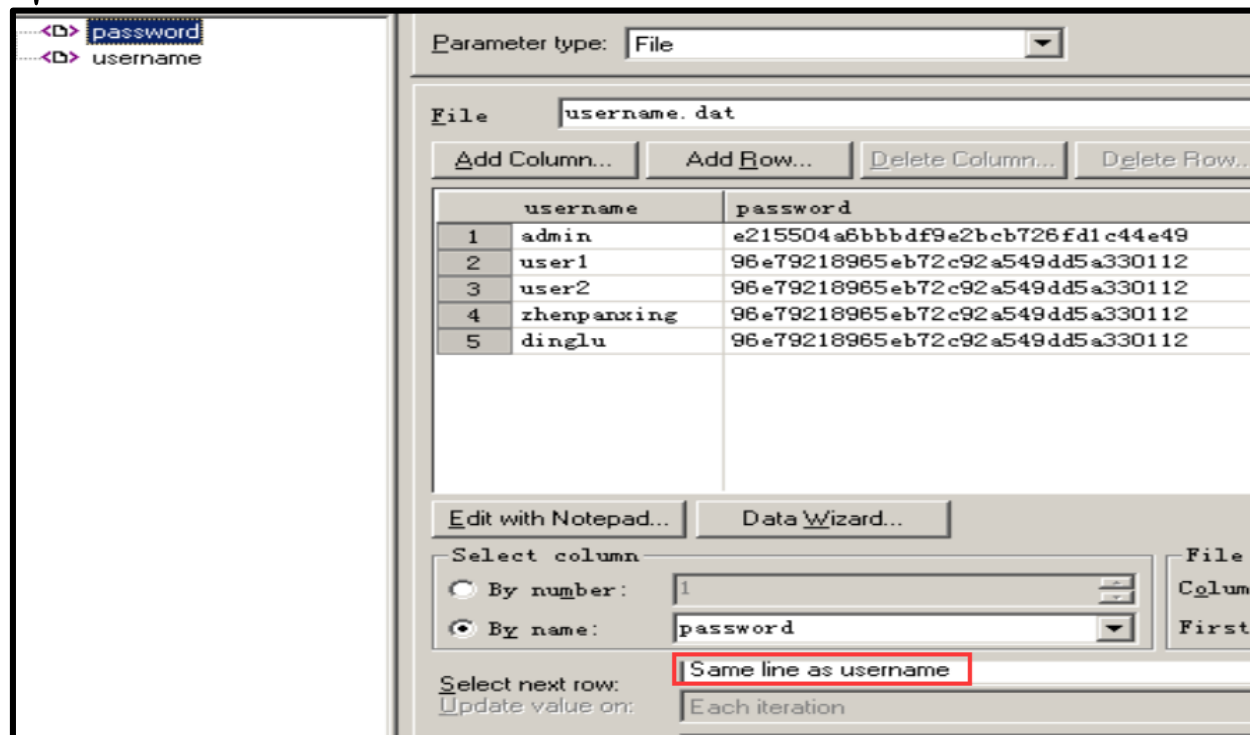
2 每个参数选择自己的列

3 参数值更新方式：第一个参数

随机+迭代；

第二个参数选择

same line as username



怎样进行参数化—两个及以上参数设置

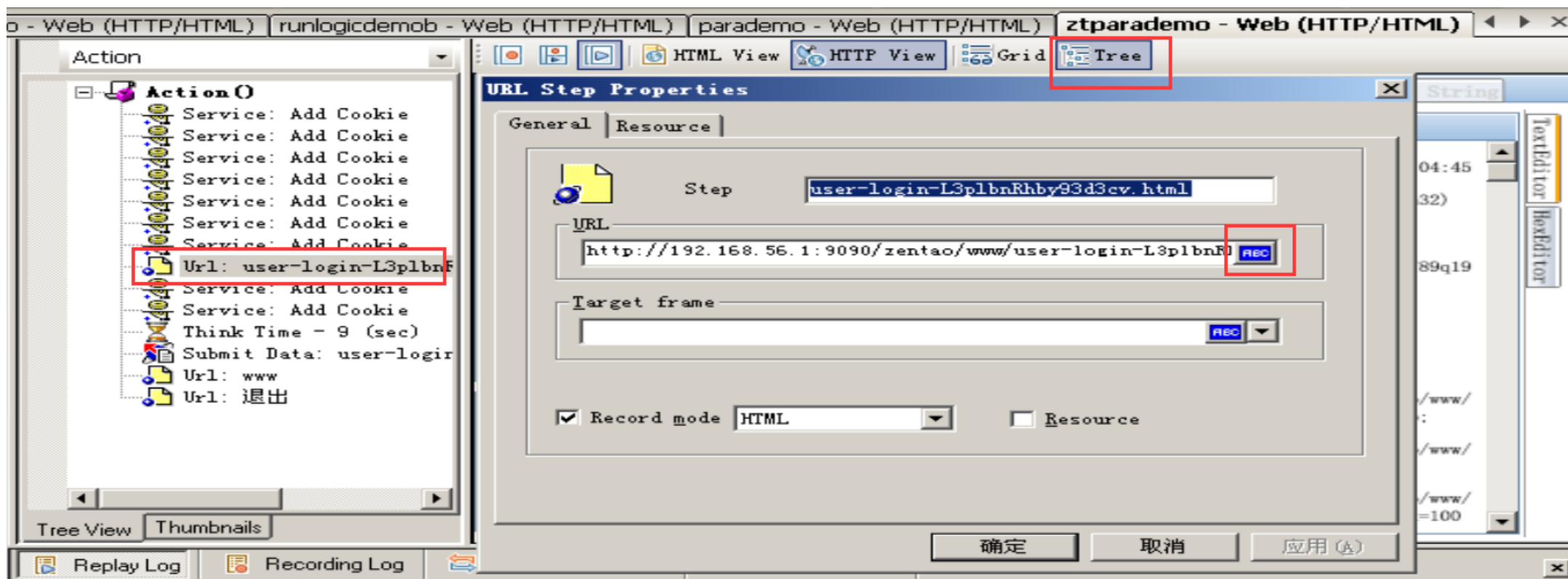
- 分析：当多个用户一起执行登录操作时，设置随机+迭代有可能会使用同一个用户名登录，怎样做？
 - 设置值更新方式：顺序+迭代

怎样进行参数化——生成大规模数据

- Excel拖拽
- 数据库插入

怎样进行参数化—哪些可以参数化

- Tree视图方式下，每条访问记录属性中，各个被填项最后有ABC字样的都可以参数化



内容总结

- 什么是参数化
- 为什么进行参数化
- 怎样进行参数化
 - 基本设置
 - 数据更新方式
 - 参数设置+不同运行逻辑结合
 - 不同参数设置+场景
 - 数据量、哪些可以参数化



Question
