
利用 LR 插件完成性能测试脚本

性能测试过程中，最耗费经历的就是编写性能测试脚本的过程，在大部分的测试工具中都是采用录制的方式，通过录制产生脚本，然后根据需要进行修改，以及参数化。有些时候为了能够完成某一个功能的脚本，需要将录制下来的脚本进行大手术，给编写脚本的人带来了很大的麻烦。

这篇文章向大家介绍了一种新的方式产生脚本，对于 ISV（独立软件开发商）和一些有代码的测试厂商带来了很大的方便，特别是一些 c/s 结构的产品采用此种方式更加有效。文章介绍了如何利用 Loadrunner Add_In 的方式进行脚本的开发，如果想采用这种方式是需要有几个前提条件的：

- 要有被测试程序的源代码
- 被测试程序的架构要清晰，对于一些分层不够好的产品，不太适宜用这种方法。
- 编写脚本的人员要了解 Loadrunner 的使用。
- 当然，Loadrunner 及插件需要正版的。

一、 插件的安装

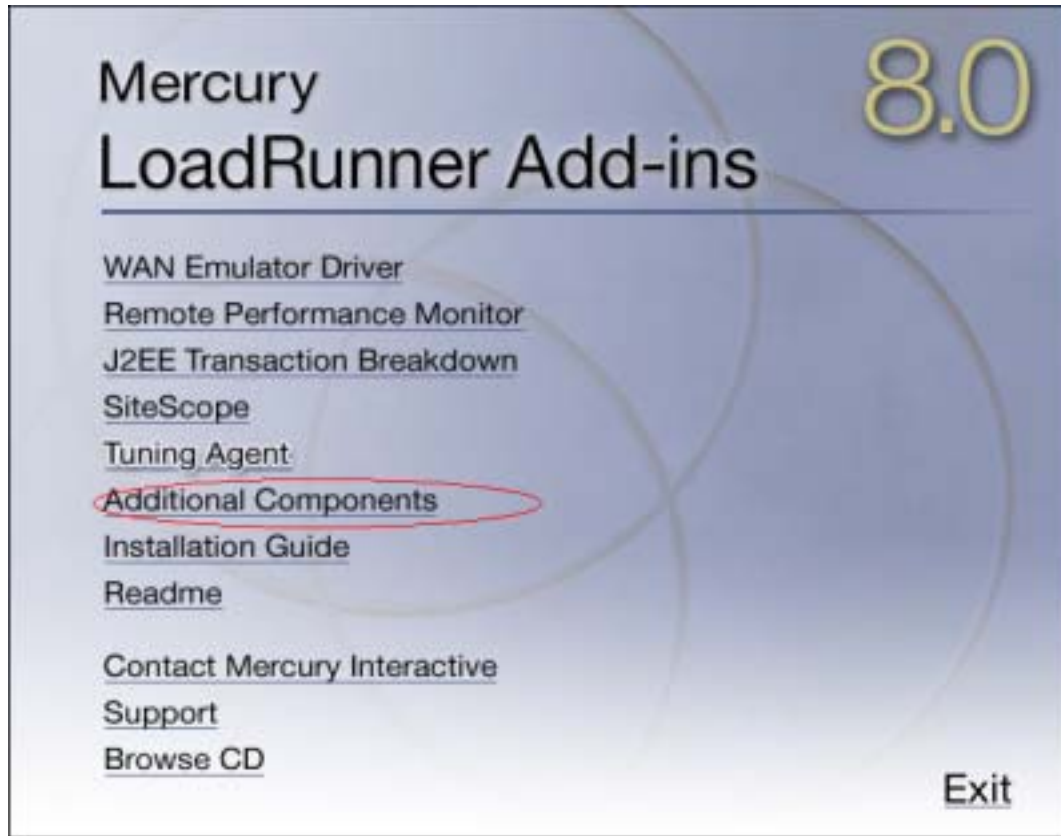
首先安装次序不要错，在安装插件之前要安装好相应的开发软件，例如：Visual Basic 或者 Visual Studio .net 或者 Jbuilder。LR 的插件是在相应软件的 IDE 环境中使用的。

- 支持的语言环境

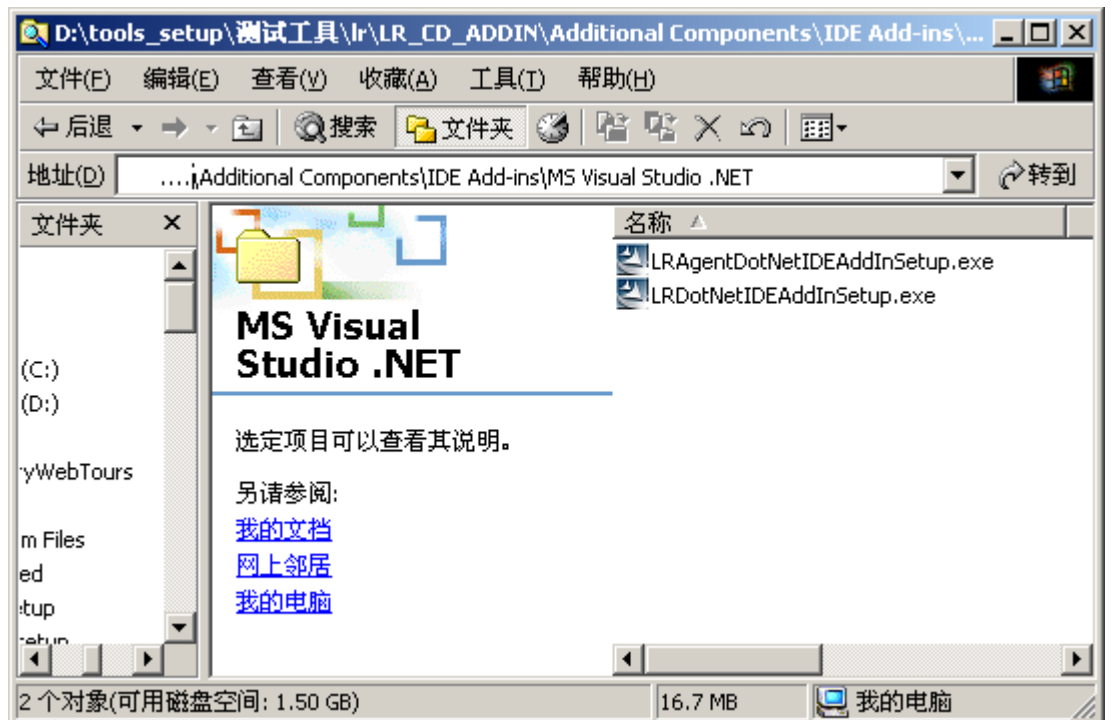
在这里首先向大家说明一下，插件能够支持的语言，我所了解的是能够支持 Visual Basic 或者 Visual Studio .net 或者 Jbuilder，每种使用方法大致相同，本文中重点介绍一下 vb 和 .net 的环境下如何使用。

- 安装过程

一般 Loadrunner 的安装盘分为两部分，一部分是 Loadrunner 主要的功能包括脚本生成器，控制器，分析器，Agent 等。另一部分是 Loadrunner 的插件，这些插件包括了各种监控插件，以及 IDE 插件，我们要使用的就是 IDE 插件



点击后进入附加组件的目录,在此我们可以选择相关的内容,我们使用的目录是:...\Additional Components\IDE Add-ins\MS Visual Studio .NET,安装文件:LRDotNetIDEAddInSetup.exe



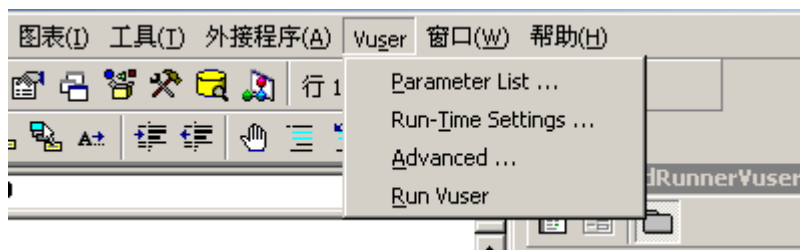
安装过程与其他软件一样,很简单,不需要再描述了。安装完以后,需要重新

启动计算机，
在安装过程中一定要注意所有 IDE 环境要关闭。

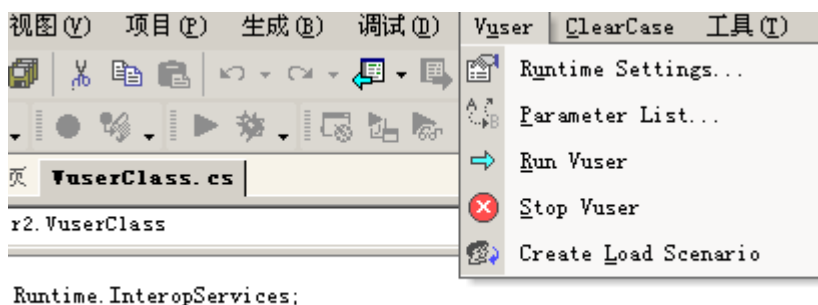
二、 认识插件

安装完以后，让我们来看看 IDE 环境变成了什么样子!!

VB 环境：



.net 环境：



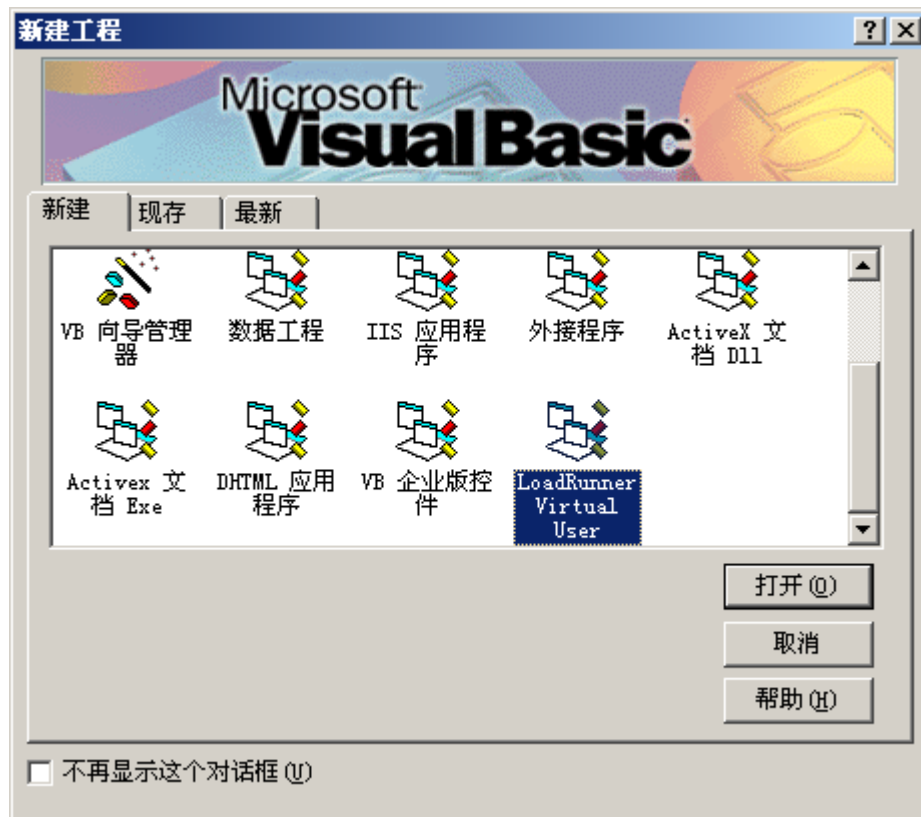
- 菜单中会多出一项【Vuser】的菜单。这个菜单就是我们主要使用的内容。
- 另外，在对象库中也会增加 loadrunner 相应的对象。这些对象的使用和软件中其他对象的使用没有什么分别。这些对应都是可以被 loadrunner 所识别的，同时也是可以并发安全的。

在成功安装了插件后，我们该如何进行性能测试工程的建立呢？接下来将分为几个部分来描述如何使用这些菜单

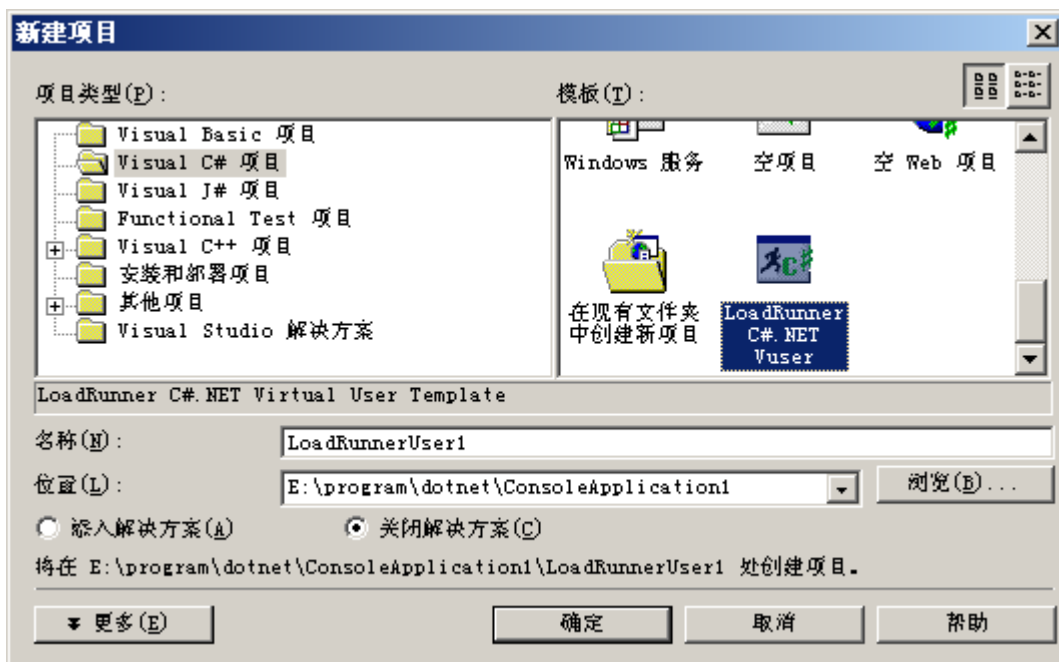
三、 建立 loadrunner 工程

在成功安装插件后，在新建工程的时候，就会多出一个选项，我们需要建立该工程。

Vb:

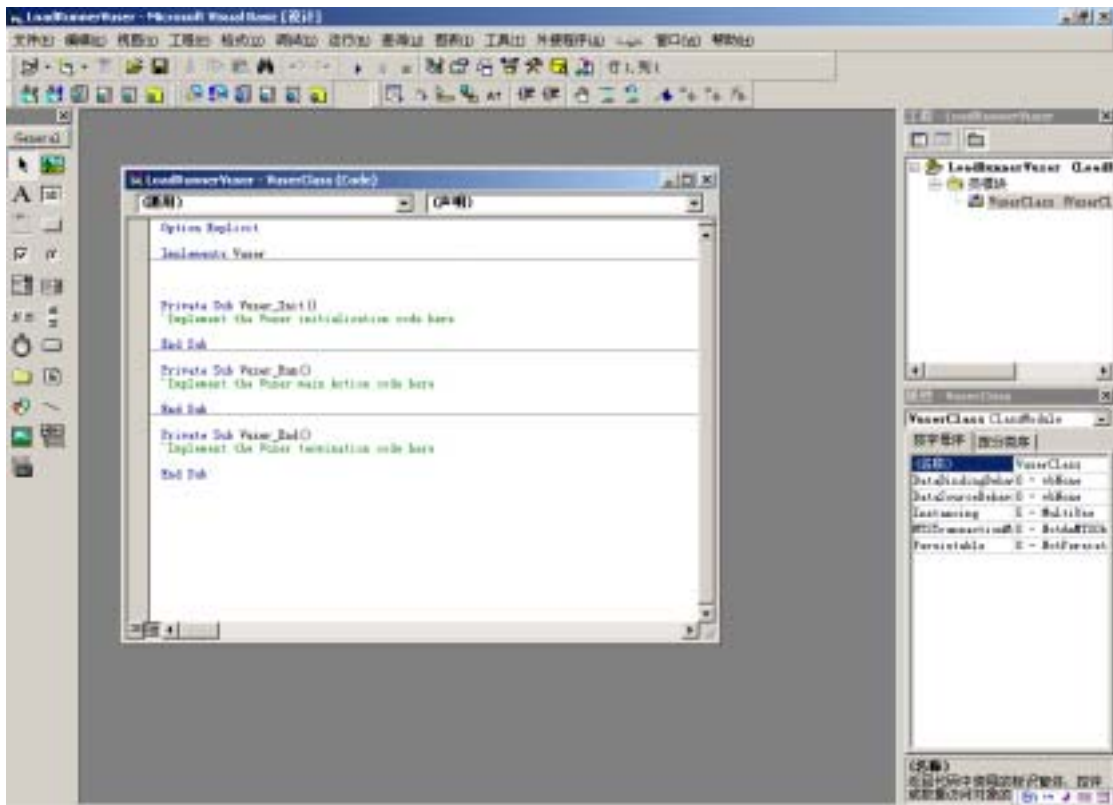


.net



在.net 中，可以支持 vb.net，vc.net，以及 c#语言，应该能够满足普通使用的需要了。

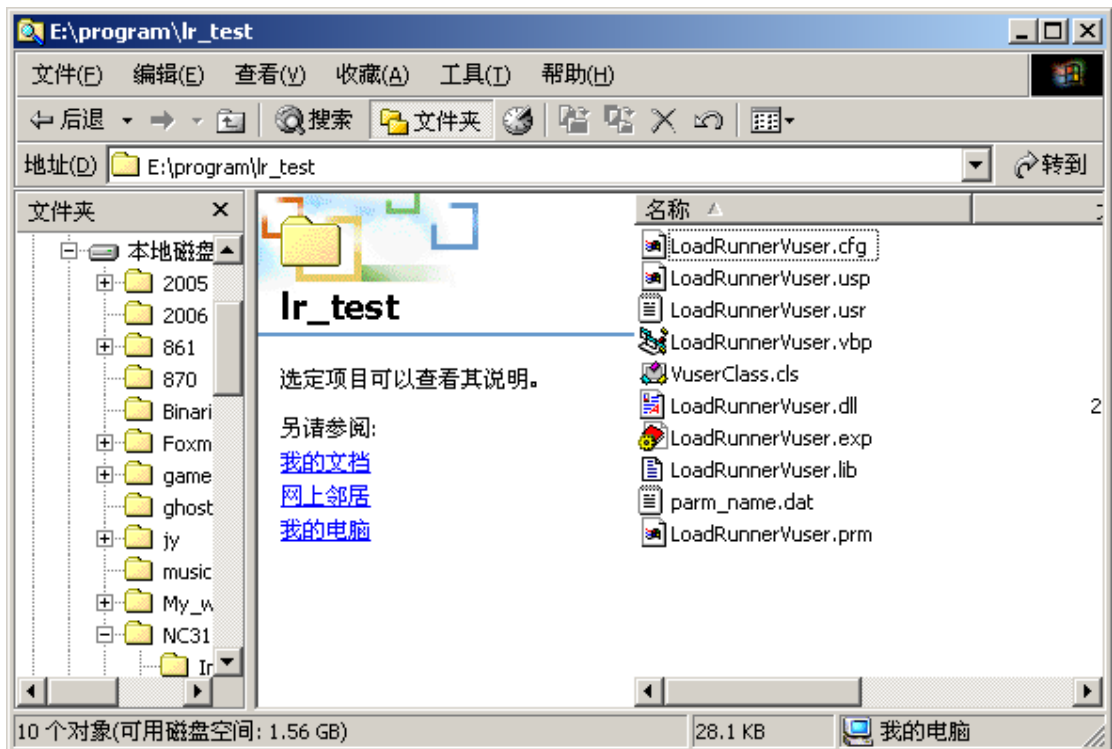
在建立工程成功后，开发环境会自动创建一些脚本，大家就会看到十分熟悉的框架结构。以 VB 为例：



与在 loadrunner VU Generator 工具一样，脚本共分为三个部分，分别为 Init()、run()、End ()。在右侧的列表中我们可以看到，只有一个 VuserClass 的类。同很多 vb 程序一样，可以修改名称，增加类等等，但是默认的三个函数名称和属性不要进行修改，只需要将内容填写在相应的位置就可以了。

建立了工程后，我们在从目录中仔细观察一下，该工程的目录和普通 vb 程序到底有什么样的区别。

首先要将编写的工程进行编译，生成.dll 文件，此时我们来解读这个工程目录。我们从下面的图中可以看到，除了 vb 工程生成的一般文件外，还多了几个新的文件。



- .usr 文件，此文件是 loadrunner 插件自动生成的文件，该文件是在 Lr Controller 工具中所要选择的脚本。也就是我们最终要使用的性能测试脚本，在后面我们还会讲到调用这个文件。利用记事本打开 .usr 文件：

[General]

Type=VbCom

DefaultCfg=LoadRunnerVuser.cfg

RunType=Com

PROGID=LoadRunnerVuser.VuserClass

CLSID={4790271A-F1BC-48C7-BA9F-E0E6D90B5952}

InterfaceID=IVuser

IID={56D136D3-506D-11D1-956B-0080C82A2A0D}

MajorVersion=5

MinorVersion=0

ParameterFile=LoadRunnerVuser.prm

[Actions]

Init=

Run=

End=

[CustomAction]

init_func=Init

end_func=End

[Transactions]

[Rendezvous]

[Components]

E:\program\lr_test\LoadRunnerVuser.dll=

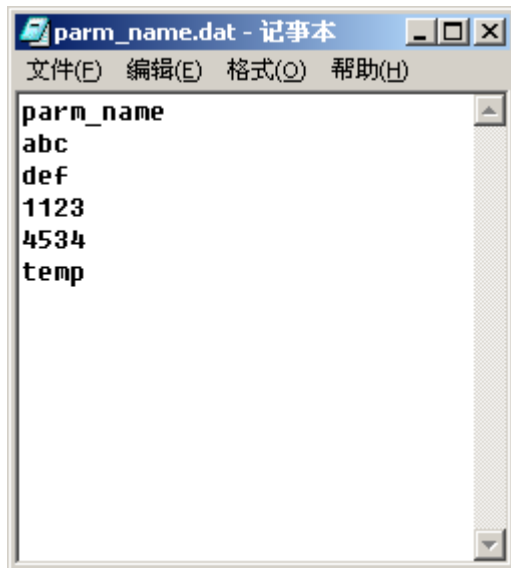
C:\WINNT\system32\MSVBVM60.DLL=

我们可以清楚的看到，该文件实际上并不是实现的脚本，而是一个工程的头信息的文件，记录了这个性能测试脚本所需要支持的类和文件，脚本的同步点名称、事务名称，CLSID 等等。描述了脚本中包含的主要内容，这些内容会关联到其他的文件上。如果你是高手，你可以直接修改此文件达到你的目的。

- LoadRunnerVuser.prm
此文件记录了脚本中所有的参数，以及参数的属性，具体某个参数的值是在下面的.dat 文件中记录的。

```
[parameter:parm_name]
Delimiter=","
ParamName="parm_name"
TableLocation="Local"
ColumnName="Col 1"
GenerateNewVal="EachIteration"
Table="parm_name.dat"
Type="Table"
value_for_each_vuser=""
OriginalValue=""
auto_allocate_block_size="1"
SelectNextRow="Sequential"
StartRow="1"
OutOfRangePolicy="ContinueWithLast"
```

- parm_name.dat 文件，parm_name.dat 文件是参数文件，说明性能测试脚本中有一个参数叫 parm_name。
打开此文件，我们可以看到这个文件中，记录的是参数的值。



每一行是一个值，与 VU Generator 工具的参数一样

- 最后要说明一下 .dll 文件，这个文件是脚本运行需要支持的文件，任何一个 Generator 机器上都要有此文件，所以这个文件非常重要，每一次编译的结果都要部署在所有的 Generator 机器上。

以上是生成的文件，在环境部署的部分我们还会详细介绍这些文件的使用。

四、 增加必要的内容

框架生成后，我们就需要根据测试的内容增加相关的代码了，下面介绍的函数都是以 c# 为例进行介绍的，其他的语言与此差不太多。

我们假设一个场景：要测试 100 个用户同时登录系统的情况。后面都将围绕着这个主题进行。

首先，无论使用什么内容都要先进行对象的创建：

```
LoadRunner.LrApi lr;  
lr = new LoadRunner.LrApi ();
```

1、 事务

事务是测试最核心的单元，所以事务的创建非常重要，下面是一个事务的定义过程。

```
LoadRunner.LrApi lr;  
lr = new LoadRunner.LrApi ();  
lr.start_transaction("syslogi n"); //事务开始, 事务名称为 syslogi n  
// x x x x 此处添加事务 syslogi n 的程序，例如：x x  
Client_syslogi n("servername", "Username", "userpassword", "2005-1-12");  
lr.end_transaction("syslogi n", lr.PASS); // 事务结束
```

事务的开始和结束与 VU Generator 工具的脚本基本上是一样的，需要注意的是，中间所调用的内容应该是所测试应用程序调用的组件，这样才能说明所编写的脚本是在

测试应用程序，否则不能说明任何问题。例如：上面的 Client_syslogin 函数就是应用程序所使用的函数。

这样对被测试的应用程序要求就比较高，应用程序分的层次比较清晰。让我们来看看应用程序是如何实现的：

```
private void buttonlogin_Click(object sender, System.EventArgs e)
{
    string srvname, username, userpws, logindate;
    srvname=text1.text;
    username=text2.text;
    userpws=text3.text;
    logindate=text4.text;
    Client_syslogin(srvname, username, userpws, logindate);
}

private void Client_syslogin(srvname, username, userpws, logindate)
{
    连接数据库；
    判断用户名称和密码是否正确；
    返回用户权限；
}
```

很明显，应用程序将业务逻辑判断和 UI 层进行了分离，所以我们对 Client_syslogin 函数的测试就是我们最终的目的。

2、参数

上面讲到了事务是如何定义和实现的，接下来我们讲讲如何在代码中进行参数化，此过程与定义一般变量很相似，仅仅在于此变量的值是来源于参数表中的数据。我们继续上面的例子：

```
LoadRunner.LrApi lr;
lr = new LoadRunner.LrApi ();
lr.start_transaction("syslogin"); //事务开始, 事务名称为 syslogin
/*
    从参数列表中取出相关的值，我们假设在参数列表中已经存在了
    srvname_parm, username_parm, userpws_parm, logindate_parm 这样几个变量，每个变量分别有
    100 个值
*/
string srvname, username, userpws, logindate //定义变量
srvname = lr.eval_string("<srvname_parm >");
username = lr.eval_string("<username_parm >");
userpws = lr.eval_string("<userpws_parm >");
logindate = lr.eval_string("<logindate_parm >"); //从参数列表中取出正确的数据传递给变量
```

```
Client_syslogin(srvname, username, userpws, logindate); //传递参数  
lr.end_transaction("syslogin", lr.PASS); // 事务结束
```

这样脚本在并发运行的时候就能够按照参数取值的要求进行不同数据的测试了。

3、同步点、延时

```
同步点和延时相对比较简单，  
lr.rendezvous("abc"); //增加名称为 abc 的同步点  
lr.think_time(1000); //增加延时
```

我们把脚本中主要的内容给大家介绍完了，其余的部分，例如参数如何定义，参数取值属性如何进行等等内容，与 VU Generator 工具的使用方法是一样的。

在.net 继承开发环境中，我们不但可以生成脚本，而且还能够利用向导来创建一个简单的场景，来测试你所编写好的一个脚本，很方便。

以上只是简单的介绍了一下如何使用这些函数，在 loadrunner 提供的对象库中还有很多很多函数，例如编写 web 产品时，代理的设置，端口设置，请求，响应以及 cookie 相关的内容，等等都可以充分利用这些组件提供的方法来实现。我们在这里就不一一描述了。

五、 环境部署

当我们有了这些脚本，接下来就是如何来运行这些脚本。

1、 Visual Basic 开发的脚本

对于利用 vb 开发的脚本，部署的工作是很繁琐的，除了要按照一般的方式要在各个负载端安装上 Load Generator 工具外，还要拷贝相关的 dll 到负载端机器上，同时要注册。然后我们就可以在 Controller 工具中选择.usr 文件，之后所有的工作就是使用 Controller 了

在此处需要注意地点，我们采用这种方式，经常会产生一个问题：我们一起就绪后，发现只有生成脚本的机器能够正常运行，而其他的负载端仍然无法正常运行，总是提示 xxx.dll 文件找不到，这是为什么呢？让我们回到第三章，仔细看看.usr 文件记录了什么？

大家一定会发现，此时的 usr 文件记录的是 dll 的绝对路径，所以当我们的 dll 拷贝到另外一台机器上，就会出现路径不同的情况，导致了 Controller 在运行的时候不能找到说到的.dll 文件。知道原因的就在，就知道如何进行修改了，让.user 中的路径和文件实际存在的路径保持一直就 ok 了。

2、 Visual Studio .net

对于利用.net 开发的脚本，就没有那么麻烦了，只是需要简单的拷贝，但是对于.net 用户则会遇到另外一个问题。我们在开发.net 应用程序的时候，很多时候都是通过读取一个 exe 文件对应的 config 文件得到相关的配置信息，那么此时这个.exe 文件到底是什么呢？可以告诉大家 Loadrunner Controller 工具中在运行脚本是所调用的.exe 是 mdrv.exe 程序，也就是如果你的程序需要通过 exe 读取配置文件，需要将此配置文件和 mdrv.exe 文件放在一个目录下：

```
...\...Mercury Interactive\Mercury LoadRunner\bin
```

六、 注意事项

下面就是本人的一些经验了，大家在使用的的时候要多多注意的地方。

- 命名

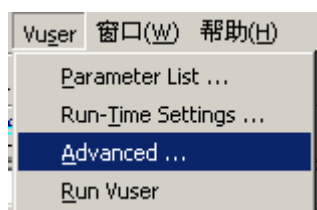
由于生成的文件主要是工程项目和 dll 文件，一定要有一个命名的规范，首先是不能重复，其次是这些名称直接关系到生成的.dll 对应的类的名称。

在 vb 中统一名称是一件非常关键的事情，在创建了工程后，系统会自动工程和类名称为：LoadrunnerVuser，我们如果需要修改名称需要分为几个步骤：

第一步：修改工程名称，修改类名称，保存工程，此时会弹出如下的对话框：

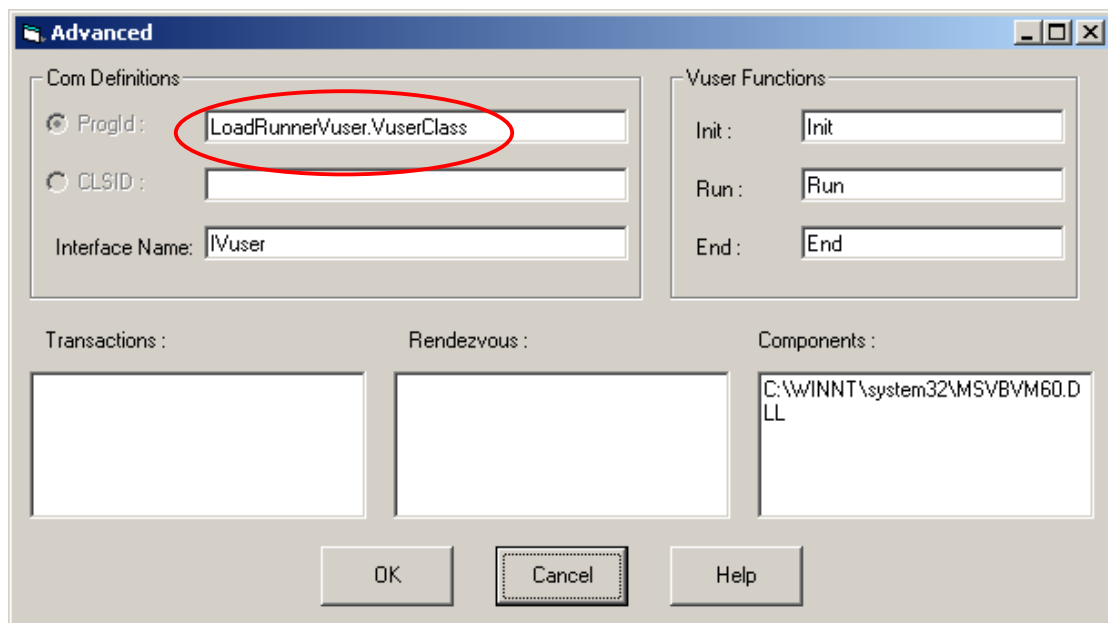


第二步：提示确定后，我们需要到下面的界面中修改对应的类名称



选择菜单上的 Advanced 项目

弹出 Advanced 窗口



手工修改此处的 progID：工程名.类名。

至此，我们修改名称完毕，可以生成我们需要的脚本了。

- 不写不能并发的语句

从上面的讲解大家可以看到，我们所编写的测试脚本是没有人工交互的，而且所有脚本都是为了进行并发的性能测试而编写的，所以脚本中如果存在不能并发的函数和方法，就会报错，一般都是内存错误，这一点希望大家一定要注意，一般采用线程安全的方式编写的.dll 都可以，就是有一些直接写库、直接写文件的方式很容易产生此问题。

- 提到并发就不能不提到日志

由于我们的脚本都是类函数，没有操作界面，所以大家在进行调试的时候会感到比较困难，那么该如何解决此问题呢？

一般我们会有两种方式：

一种是采用 loadrunner 提供的函数，将结果返回到相关的日志中

另一种是采用自己编写日志的方式，记录需要记录的内容。此时就需要注意写文件的操作是不能并发的，所以还需要再次处理一下。

以上是向大家介绍了一下如何用另外一种方式进行性能测试，这种测试方法不是对所有的产品所有的程序都适合的，大家可以根据自己的需要进行选择。