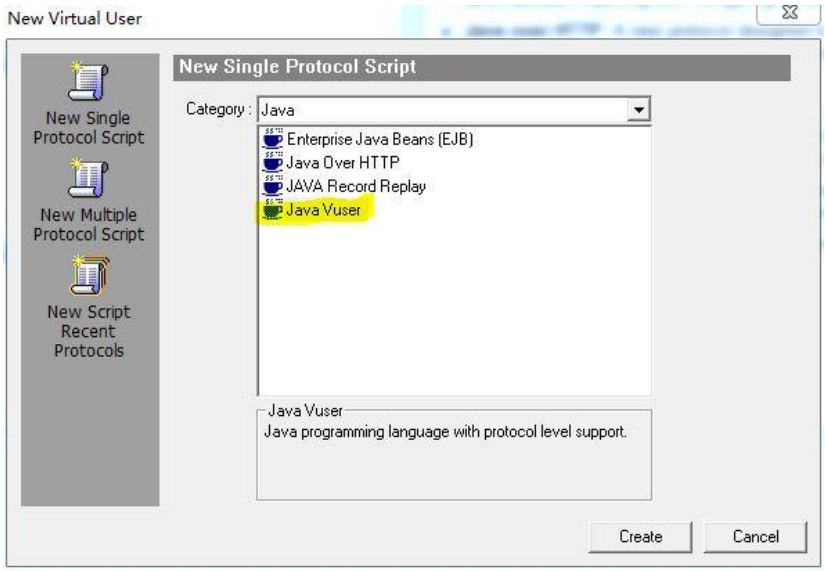
使用 Loadrunner 测试数据库性能 plus

之前写过一个文档，使用 LR 通过 JDBC 测试数据库性能，但那个文档主要介绍的是在 MyEclipse 里编写类后，在 LR 中如何加载如何设置，此种方法有一定的局限性，例如把写好的类放在 LR 中调用，都是在 action 中调用类的所有方法，若方法中包含完整的建立连接和断开连接，每次迭代的时候就会不断重复这个操作，既耗费资源，又耗费时间，对于要求高并发的测试场景，显然不是最好的办法，因此重新考虑将 java 的脚本直接放在 LR 中进行编辑。

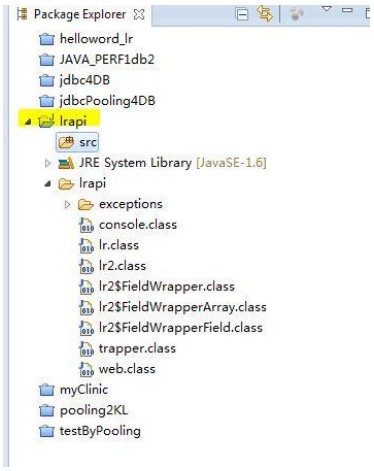
首先我们要明确一个问题，使用 MyEclipse 编写类的手段仍然是不可或缺的，因为 LR 在编写 JAVA 脚本时不能对其他脚本进行调用，如果需要外部方法，则必须通过引入 jar 的办法，所以如果 LR 脚本里需要其他类和方法的支持，就必须提前把这个 class 通过 import 的办法加载到这个脚本里。另外为了 LR 使用，必须使用 J2EE 而不能使用 J2SE，因为虽然在验证效果上 J2EE 和 J2SE 都能体现 java 功能和数据连接，但是发布的时候，J2EE 能够根据创建的包的层次，分别发布每一个 class 文件，而 J2SE 则会把所有的 class 发布在一个 jar 里。

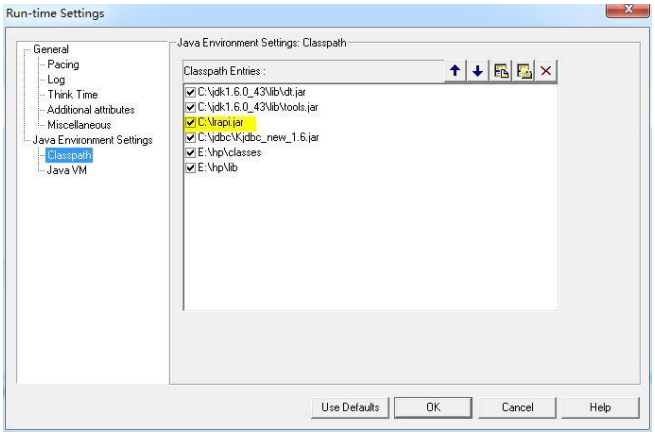
接下来我们开始准备 LR 的编辑环境，当然脚本协议仍然是 java vuser，



1. 准备编辑环境
    a) 引入 lrapi.jar

打开编辑器之后，为了能有更多的 java 编辑提示和引用提示，需要导入一个 jar 包。方法是，找到 lr 系统安装目录下的 classes 目录，（%loadrunner%\classes），里面有一个 lrapi 目录，把这个目录完整复制到 j2se 的一个项目里，然后把这个 j2se 工程发布成一个 jar 包，最后再把这个 jar 包 import 到 LR 脚本中。



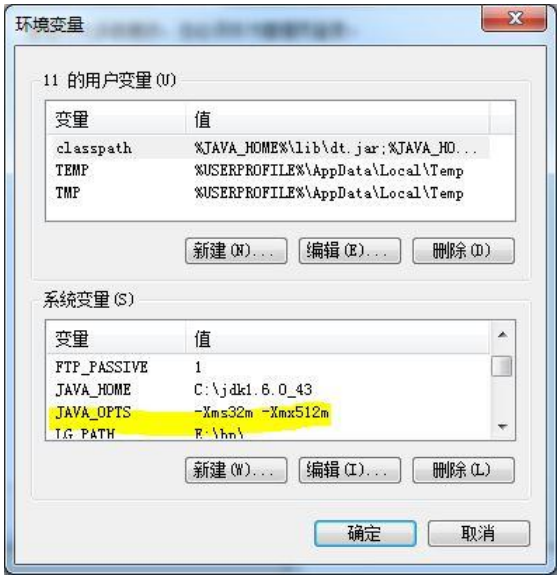


其他需要的 jar 包，例如 jdbc 等，根据需要同样引入到环境中。

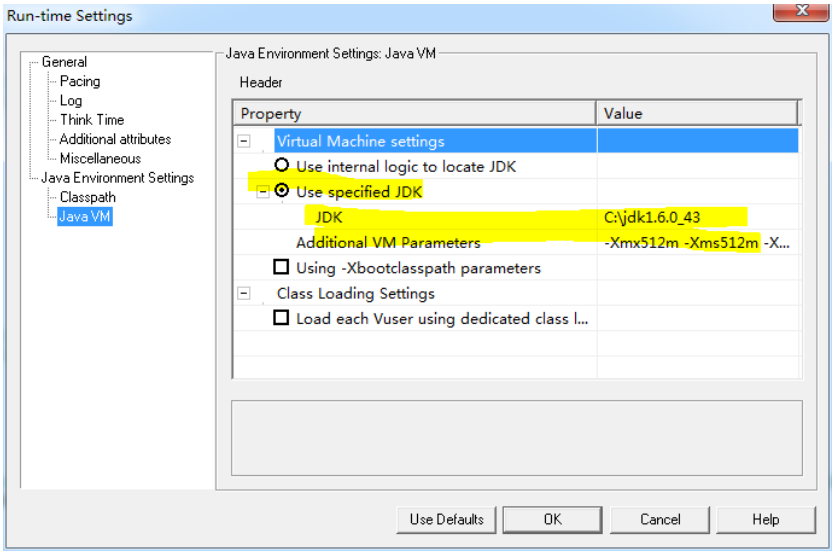
b) 调整 jvm 参数

由于需要大量运行 java 程序，所以为了避免出现堆错误和内存溢出，需要对 jvm 进行设置，根据运行位置的不同，共有 3 个地方需要添加 jvm 参数。

i. 在系统运行，需要修改系统环境变量，增加一个系统环境变量 JAVA_OPTS，如图：

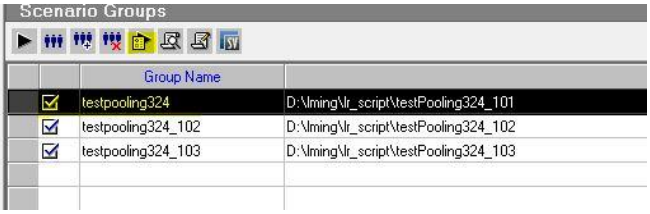


ii. 在 lr 脚本调试时，编辑“用户运行时”设置，如图：

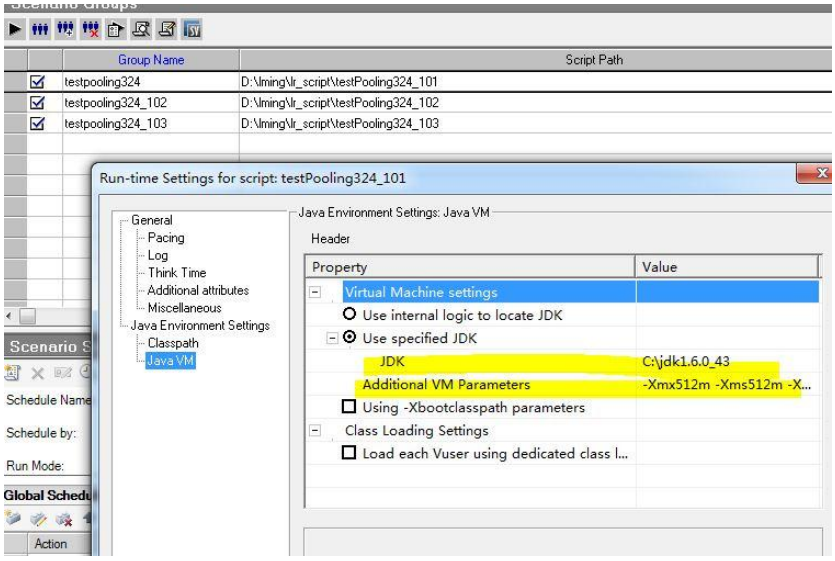


注意选择指定的 jdk（1.6）和添加 VM 参数，参数内容：
-Xmx512m -Xms512m -Xmn200m -XX:PermSize=64m -Xss128k

iii. 在编辑 lr 场景时，同样要设置 ii 的内容，如图：



注意如果如图，每个脚本都是独立的，需要每个脚本各自设置。编辑的内容和脚本调试时一样。



2. 编辑脚本

由于是在 LR 的编辑器里编辑需要的内容，因此要充分发挥这个环境的特点，在这个 Actions 类中包含 init()，action()，end()3 个方法，而和普通的 http 协议一样，也是 init()和 end()只执行一遍，action()会根据场景设置不断循环迭代，因此

为了提高执行效率减少 jdbc 连接断开所产生的资源消耗，可以把创建连接和断开连接写到 init 和 end 里，把具体的查询语句写到 action 里。

首先需要在 Actions 里建立公共变量：

```
public class Actions
{

    static Connection conn = null;
       java.sql.PreparedStatement statement = null;
       java.sql.ResultSet rs = null;

        public  int init() throws Throwable {
```

接下来我们具体讲如何在 init 里编写连接初始化，通过 jdbc 连接数据库实际上有两种办法，一个是使用 DriverManager，一个是使用 DataSource。

DriverManager：

```
Class.forName("");
conn =    (Connection)DriverManager.getConnection("url","user","pass");
```

这里得到了 conn 属性，在 action 中所有的查询都是使用这个连接属性。

DataSource:

```
        PoolDataSource k = newPoolDataSource();
        k.setHostName("");
        k.setPort(1234);
        k.setDatabaseName("");
        k.setUser("user");
        k.setPassword("pass");
//      k.setUrl("jdbc:url");
        k.setMaxActive(2);
        k.setMinIdle(1);
        k.setLoginTimeout(3000);
        PooledConnection pc = k.getPooledConnection();
        conn = (Connection)pc.getConnection();
```

同样也是得到 conn 属性，以上两种形式在 init 里任选一种即可。

打开的 conn 要在 end()中关闭，

```
public int end() throws Throwable {
  conn.close();
        return 0;
}//end of end
```

在 action 中调用 conn 执行操作，

```
statement = conn.prepareStatement("select * from table);
rs = (ResultSet) statement.executeQuery();
if(rs.next()){
                    System.out.println("select1 OK");
                }
```

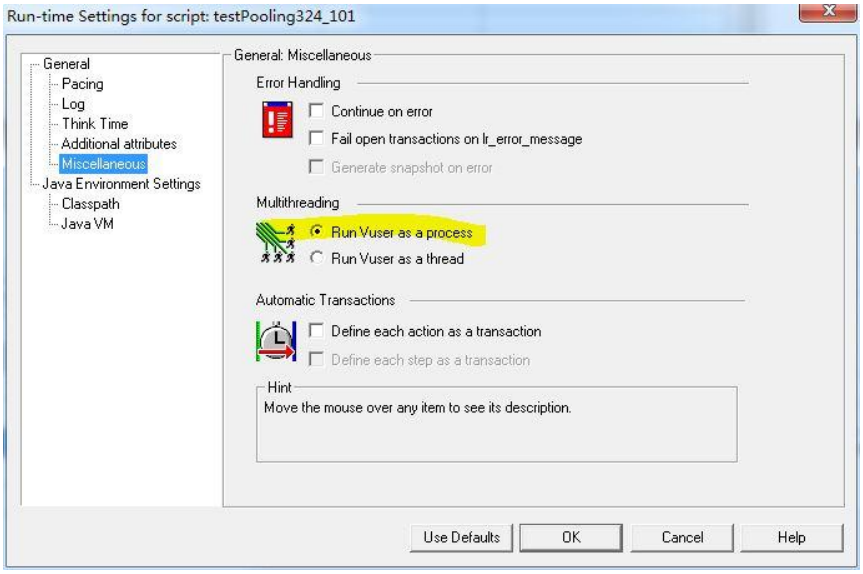
```
else{
  System.out.println("select1 fail");
}
```

通过自己写对 rs 的状态判断，获得输出内容。如果需要获得查询内容，则需要 import 外部 jar 包，存储数据对象的 get 方法和 set 方法。这里可以根据需要灵活增加。

以上为编辑脚本的基本方法，完成编辑后可先做单步调试，确保脚本编写正确。

3. 执行测试

执行测试的方法和其他 LR 性能测试的方法有太大区别，但是需要注意的是，这个场景必须选择以进程方式运行，否则使用线程方式会产生大量的错误。



以进程方式运行即可。

4. 需要注意

这种数据库性能测试方法需要注意的是，无论用哪个方式连接，都是在 init 中获得 conn 属性，而 init 每个脚本只会执行一次，因此每个脚本实际上只做了一次连接，若数据库会根据连接做某些判断，则这种方法无法模拟，次方法实际上脱离了中间件等网络应用环节，因此某些网络行为也无法模拟。

这种测试方法只是用于纯粹验证数据库执行操作时的性能情况，若数据库的执行需要网络中某些特定因素，则无法达到预期效果。

以下为完整的 LR 脚本例子：


lrTestDB.zip