　　目的：对**[Java](javascript:;" \t "_self)**程序进行**[测试](javascript:;" \t "_self)**；

**一、核心步骤**

　　1.创建一个Java工程；

　　2.将**[JMeter](javascript:;" \t "_self)**的lib目录下的jar文件添加进此工程的Build Path；

　　3.创建一个类并实现JavaSamplerClient接口或继承AbstractJavaSamplerClient，并重写：

|  |
| --- |
| public Arguments getDefaultParameters()：设置可用参数及的默认值；  public void setupTest(JavaSamplerContext arg0)：每个线程测试前执行一次，做一些初始化**[工作](javascript:;" \t "_self)**；  public SampleResult runTest(JavaSamplerContext arg0)：开始测试，从arg0参数可以获得参数值；  public void teardownTest(JavaSamplerContext arg0)：测试结束时调用；  4.Export为Runnable Jar File； |

　　5.将此jar包放入JMETER\_HOMElibext目录；

　　6.以管理员身份打开JMeter；

　　7.创建线程组、Java Request、结果树，进行测试；

**二、实例**

　　利用JMeter对服务进行**[性能测试](javascript:;" \t "_self)**，服务为：将输入的两个参数通过IO存入文件；

　　1、编写Java代码

　　服务：

|  |
| --- |
| package [**test**](javascript:;);  import java.io.File;  import java.io.PrintWriter;  public class OutputService {  public static void output(String filename，int a， int b) throws Exception {  PrintWriter out = new PrintWriter(new File(filename));  out.write(a+":"+b);  out.close();  }  } |

　　测试类：

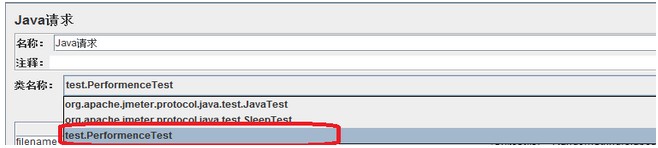
|  |
| --- |
| package test;  import org.apache.jmeter.config.Arguments;  import org.apache.jmeter.protocol.java.sampler.JavaSamplerClient;  import org.apache.jmeter.protocol.java.sampler.JavaSamplerContext;  import org.apache.jmeter.samplers.SampleResult;  public class PerformenceTest implements JavaSamplerClient {  private SampleResult results;  private String a;  private String b;  private String filename;  // 设置传入的参数，可以设置多个，已设置的参数会显示到Jmeter的参数列表中  public Arguments getDefaultParameters() {  Arguments params = new Arguments();  params.addArgument("filename"， "0");//设置参数，并赋予默认值0  params.addArgument("a"， "0");//设置参数，并赋予默认值0  params.addArgument("b"， "0");//设置参数，并赋予默认值0  return params;  }  // 初始化方法，实际运行时每个线程仅执行一次，在测试方法运行前执行  public void setupTest(JavaSamplerContext arg0) {  results = new SampleResult();  }  // 测试执行的循环体，根据线程数和循环次数的不同可执行多次  @Override  public SampleResult runTest(JavaSamplerContext arg0) {  b = arg0.getParameter("b"); // 获取在Jmeter中设置的参数值  a = arg0.getParameter("a"); // 获取在Jmeter中设置的参数值  filename = arg0.getParameter("filename"); // 获取在Jmeter中设置的参数值  results.sampleStart();// jmeter 开始统计响应时间标记  try {  OutputService test = new OutputService();  test.output(filename，Integer.parseInt(a)， Integer.parseInt(b));  results.setSuccessful(true);  // 被测对象调用  } catch (Throwable e) {  results.setSuccessful(false);  e.printStackTrace();  } finally {  results.sampleEnd();// jmeter 结束统计响应时间标记  }  return results;  }  // 结束方法，实际运行时每个线程仅执行一次，在测试方法运行结束后执行  public void teardownTest(JavaSamplerContext arg0) {  }  public static void main(String[] args) {  // TODO Auto-generated method stub  Arguments params = new Arguments();  params.addArgument("a"， "0");//设置参数，并赋予默认值0  params.addArgument("b"， "0");//设置参数，并赋予默认值0  JavaSamplerContext arg0 = new JavaSamplerContext(params);  PerformenceTest test = new PerformenceTest();  test.setupTest(arg0);  test.runTest(arg0);  test.teardownTest(arg0);  }  }  Export 为 Runnable Jar File； |

**2、设置JMeter**

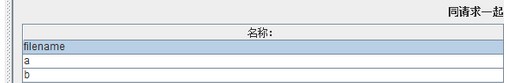
　　以管理员身份打开JMeter，并创建Java Request后，结构如下图所示：



　　在Java请求中发现了自己新建的测试类：



　　我们在JMeter中发现参数有三个：



　　我们在响应的数值中填入：



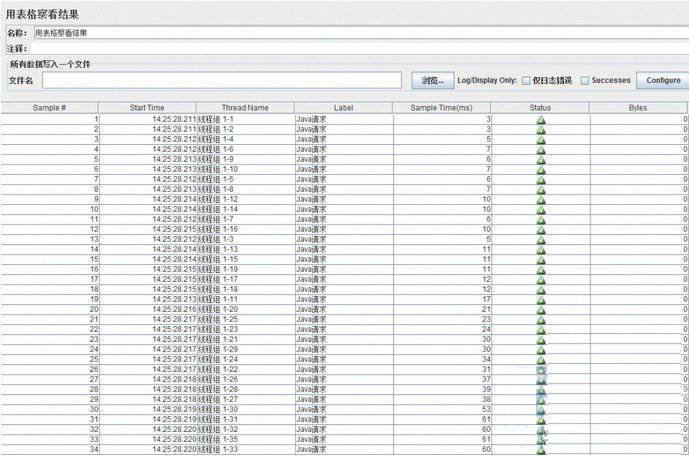
http://www.spasvo.com/ckfinder/userfiles/images/0020131118_07(1).jpg　　大家一定很奇怪，怎么会有一些看不懂的东西，这是JMeter提供的函数，我们可以在  
　　因为我们要做的是性能测试，因此我们需要开多个线程并发测试，因此随机数很重要；  
　　常用的函数为：  
　　(1)\_Random生成随机整数；  
　　(2)\_RandomString生成随机字符串；  
　　在线程组中设置并发线程数为10000，保存后即可运行；  
　　图形结果如下所示：



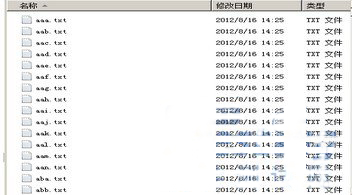
　　聚合报告：



　　表格查看结果：



　　结果成功，成功生成了接近10000个文件：



　　3.期间遇到的问题

　　3.1 Export为Jar File导致Java Request的类名称无法找到；

　　3.2 JMeter java.lang.OutOfMemoryError: PermGen space问题：

　　在JMeter.bat 中修改：

　　set HEAP=-Xms512m -Xmx1024m

　　set NEW=-XX:NewSize=128m -XX:MaxNewSize=128m

　　set SURVIVOR=-XX:SurvivorRatio=8 -XX:TargetSurvivorRatio=50%

　　set TENURING=-XX:MaxTenuringThreshold=2

　　set RMIGC=-Dsun.rmi.dgc.client.gcInterval=600000 -Dsun.rmi.dgc.server.gcInterval=600000

　　set PERM=-XX:PermSize=256m -XX:MaxPermSize=512m

　　3.3 JMeter需要以管理员身份打开，否则会出现：

