
第六章 JMeter 性能测试

一、 相关术语

1 JMeter 是做什么的

jmeter用于测试软件的性能。

2 线程组

线程数：虚拟用户数

3 Sampler 取样器

- 用来模拟用户操作，向服务器（被测系统）发出请求。
- http 请求
 - ✓ 发送 http 请求
- 跟随重定向
 - ✓ 会沿用之前的 sessionid。
- Debug Sampler
 - ✓ 用于显示正则表达式、参数等的结果。
- Beanshell Sampler
 - ✓ 于输出参数值。

4 断言

- 用来验证结果是否正确，相当于检查点。
 - ✓ 常用响应断言。

5 前置处理器

- 请求发送前可能会做一些环境或者参数的准备工作，比如在对数据库进行操作前需要先建立一个数据库连接。

6 后置处理器

- 一般放在取样器之后，用来处理服务器的返回结果。
- 正则表达式处理器
 - ✓ 用于解决关联问题；
 - ✓ 要放在请求以下。

7 配置元件

- CSV Data Set Config (CSV 数据集配置)
 - ✓ 用于参数化。
- http cookie 管理器
 - ✓ 录制之前添加，用于解决 sessionid 存储于 cookie 中的情况。
- 用户自定义变量

8 逻辑控制器

- 事务控制器
 - ✓ 用于添加事务。
- 循环控制器
 - ✓ 用于设置迭代次数。

9 定时器

- Synchronizing Timer (同步计时器)
 - ✓ 用于设置集合点。
- 固定定时器、Uniform Random Timer
 - ✓ 可用于设置思考时间，需要每一次思考时都加此定时器，不常用。
- 高斯随机定时器
 - ✓ 产生服从正态分布的随机时间间隔，一般用于模拟思考时间，需要每一次思考时都加此定时器，不常用。

10 监听器

- 用于对测试进行监控，查看运行结果，常用察看结果树。


二、 搭建 Jmeter 环境

- 控制机上安装 JDK 并配置环境变量（不能仅使用 JRE，负载机可以）
- 复制 jmeter 文件夹到计算机
- 安装 badboy
 - ✓ 与 jmeter 无关。
 - ✓ 用于录制脚本。
 - ✓ 常用于导出脚本到 jmeter。

三、 录制脚本

1 badboy 录制脚本

- 输入 url 地址后，单击绿色箭头
 - ✓ 红色原点表示“开始录制”，默认已选中；

- ✓ 黑色四方块：停止录制，需要手工停止。
- 回放脚本
 - ✓ 右击 step1→Play All，或使用工具栏按钮。
- 将脚本导出为.jmx 格式
 - ✓ File→Export to Jmeter
- jmeter 导入 jmx 脚本
 - ✓ 打开 jmeter
 - ✧ \apache-jmeter-3.0\bin\jmeter.bat
 - ✓ 文件→打开
- jmeter 回放脚本
 - ✓ 右击 ThreadGroup 下的 Step1→添加→监听器→察看结果树
 - ✓ 单击“察看结果树”，单击工具栏中的图标
 - ✓ 查看图形方式的回放结果
 - ✧ 单击“察看结果树”→选择下拉列表中的“HTML”→选中某个 url“→点击“响应数据”

2 jmeter 代理录制脚本

- 为计算机设置 IP
- 在浏览器中设置代理服务器
 - ✓ 打开浏览器→工具→Internet 选项→连接→局域网设置→勾选“为 LAN 设置代理服务”，地址输入 Jmeter 的 IP，端口 8888（jmeter 默认）
- 添加线程组到测试计划
 - ✓ 添加“配置元件”→http cookie 管理器，否则影响关联效果
- 在工作台下添加“非测试元件”→HTTP 代理服务器
 - ✓ 目标控制器：测试计划→线程组
 - ✓ 分组：不对样本分组
- 启动，操作软件后开始录制
 - ✓ 需要使用真实 IP，不能使用 localhost
- 录制完成后，停止

3 自己编写脚本

- 需要对请求非常了解。

四、 增强 Jmeter 脚本

1 添加断言

- 【例 1】检查注册是否成功
 - ✓ 先根据“结果树”中的“HTML”中的“响应数据”，找到需要检查的网页中的文本

- ✓ 复制上述内容，根据“结果树”中的“Text”中的“响应数据”，确定最终确定要检查的文本（可能含有标记）
- ✓ 在线程组下找到需要检查的 url，右击“添加”→断言→响应断言
 - ✧ 要测试的响应字段：响应文本
 - ✧ 模式匹配规则：包括
 - ✧ 要测试的模式：添加
 - 预期结果如“Thank you, zhsan1”
 - ✧ 注意
 - 找不到会报错，找到无反应

2 事务

- 右击 step1→添加→逻辑控制器→事务控制器。
- 拖动事务位置，将 url 拖动到合适的事务中。

3 参数化

- 配置元件→CSV Data Set Config
 - ✓ Filename：参数文件名
 - ✧ 可用相对或绝对路径，相对路径须在...jmeter 安装位置\bin 中；
 - ✧ 当前目录下使用./；参数文件中不能使用列名。
 - ✓ File encoding：文件编码格式
 - ✧ 常用且建议 UTF-8
 - ✓ Variable Names(comma-delimited)
 - ✧ 以英文逗号间隔的列名，与数据的间隔符可以不一致
 - ✓ Delimiter(use '\t' for tab)
 - ✧ 指定数据间隔符，Tab 用'\t'(不加引号)，默认逗号
 - ✓ Allow quota data
 - ✧ 表示数据中是否有分隔符
 - ✓ Recycle on eof
 - ✧ 参数不够用时，是否从头开始重新循环
- 【例 1】注册 200 个账号，客户名使用用户名。
 - ✓ 手动编辑参数文件，写入参数，但不要写列名
 - ✓ 找到需要参数化的输入数据的 url
 - ✧ 在前面的位置，右击“添加”→配置元件→CSV Data Set Config
 - ✧ 将参数的“值”改为“\${参数名}”
 - 使用参数时不加引号
 - ✓ 负载测试
 - ✧ 单击“Thread Group”，设置“线程数”为并发人数
 - 一个线程对应一个模拟用户
 - ✧ Ramp-Up Period（in second）
 - 线程启动开始运行的时间间隔，单位是秒，即所有线程在多长时间开始运行。
 - 如设置线程数为 200，数据为 75 秒，相当于 15s 加载 40 个用户。

4 关联

➤ 后置处理器→正则表达式提取器

✓ 引用名称

✧ 根据指定规则找到的字符串存放到此名表示的参数中。

✓ 正则表达式

✧ 一个用()表示一个模式；left(?)right: ?表示遇到行中的第一个 right 就作为右边界，然后不断向后寻找。

✓ 模板

✧ \$1\$指第一个模板(模式)，\$2\$第二个...，此项必填；

✧ 允许 \$1\$ \$2\$ 的写法。

✓ 匹配数字

✧ 找到的第某项，1 为第 1 项，-1 为所有项，0 是随机

● 参数名_matchNr (区分大小写) 中存着找到几项，必写-1

● 参数名_gi 表示找到的第 i 项，i 从 1 开始，g0 表示包含左右边界在内的整个字符串

■ g 是关键字

● 有多个模板时，参数名_i_gj 表示找到的第 i 行第 j 列数据

➤ 正则表达式案例

✓ 网页内容

```
<table>
  <tr><td>hello</td><td>zhsan</td><td>123</td>
  <tr><td>hello</td><td>lisi</td><td>5326</td>
</table>
```

✓ 正则表达式: <tr><td>hello</td><td>(.)</td><td>(.)</td>

✓ 【例 1】模板: \$1\$, 匹配数字: 1

```
test=zhsan, test_g=2, test_g0=<tr><td>hello</td><td>zhsan</td><td>123</td>
test_g1=zhsan
test_g2=123
```

✓ 【例 2】模板: \$1\$, 匹配数字: 2

```
test=lisi, test_g=2, test_g0=<tr><td>hello</td><td>lisi</td><td>5326</td>
test_g1=lisi
test_g2=5326
```

✓ 【例 3】模板: \$2\$, 匹配数字: 1

```
test=123, test_g=2, test_g0=<tr><td>hello</td><td>zhsan</td><td>123</td>
test_g1=zhsan
test_g2=123
```

✓ 【例 4】模板: \$2\$, 匹配数字: 2

```
test=5326, test_g=2, test_g0=<tr><td>hello</td><td>lisi</td><td>5326</td>
test_g1=lisi
test_g2=5326
```

✓ 【例 5】模板: \$1\$ \$2\$, 匹配数字: -1

```
test_1=zhsan 123
test_1_g=2, test_1_g0=<tr><td>hello</td><td>zhsan</td><td>123</td>
```

```
test_1_g1=zhsan (第 1 行第 1 列)
test_1_g2=123
test_2=lisi 5326
test_2_g=2, test_2_g0=<tr><td>hello</td><td>lisi</td><td>5326</td>
test_2_g1=lisi
test_2_g2=5326
test_matchNr=2
```

➤ 【例 6】?的用法

- ✓ 正则表达式: `<td>(.*?)</td>`与`<td>(.*?)</td>`
- ✓ 查看正则表达式的结果
 - ✧ 在线程组中添加 Debug Sampler, 移到正则表达式提取器之后
 - ✧ 在线程组中添加 BeanShell Sampler, 移到正则表达式提取器之后, 写入下面的代码之一
 - `${参数名};或"字符串${参数名}"`;
 - 结果出现在请求中, 最后一个数据出现在响应中
 - 如果省略分号, 则数据都不出现在响应中, 此外还会报错, 但是数据还会正常显示
 - `ResponseMessage="字符串${参数名}"`;
 - 结果出现在响应中, 最后一个数据出现在响应中
 - `log.error(数据)`;
 - 向日志中写数据
 - 点击“警告”图标可以清除日志
 - `log.info(数据)`;
 - 向日志中写数据

➤ 【例 7】录制订票的脚本, 保证脚本正确实现业务。

- ✓ 找到查看结果树中出错的请求, 在这个 url 的请求中查找会话值
- ✓ 在 step 下找到哪个 url 的发送数据中有会话值
- ✓ 在查看结果树的 Text/源代码界面中, 倒着找那个 url 中的响应数据中有会话值
- ✓ 找到 sessionid 的左右边界
- ✓ 在响应的 url 处添加后置处理器→正则表达式提取器
- ✓ 在请求的 url 处, 将 userSession 参数的值改为“`${引用名称}`”
 - ✧ 不输入引号

5 自定义函数

➤ 编写 Java 类

- ✓ 编写自己需要的 static 函数, 并加入 main 方法, 先编译与运行正确
- ✓ 删除 main 方法后保存, 右击项目→导出→Java→可运行的 JAR 文件
 - ✧ 启动配置
 - 选择测试的类名
 - ✧ 导出目标
 - 必须放在`\apache-jmeter-3.0\lib\ext`目录中, 名字可与类名不同
 - ✧ 忽略错误
 - 找不到 mian 方法

- ✓ 【例 1】编写一个输出字符串左边多个字符的函数。
- Jmeter 导入包
 - ✓ 测试计划→Add directory or jar to classpath 处：浏览，找到包
- BeanShell 引用函数
 - ✓ import 包名.类名
 - ✓ vars.put("新参数名","常量值或参数名");
 - ✧ 常量值或参数结果存入新参数中
 - ✧ 无法在 BeanShell 的请求和响应中看到新参数值
 - ✧ BeanShell 后面的 url 以及请求中可以使用该参数
 - ✓ 可以在日志中看到参数值
 - ✧ log.error(包名.类名.函数名(参数));
 - ✓ 【例 2】输出航班号。
 - ✧ import str.StrMethod; //包名.类名
 - ✧ log.error(str.StrMethod.split("\${banci}"," checked ")[0]);
 - ✧ vars.put("hb","常量值或参数名");
 - 结果存入 hb 参数

6 循环控制器

- 右击 Step→添加→逻辑控制器→循环控制器
 - ✓ 循环最好放在事务之外。
 - ✓ CSV Data 要放在循环之中。
- 添加计数器，查看循环次数
 - ✓ 添加配置元件→计数器
 - ✓ 计数器要放在循环内部
- 使用参数的方式
 - ✓ 点击 Step1 中的参数化→Sharing mode: Current Thread
 - ✧ 不选择 Current Thread，会导致有重复买票的情况
- 【例】参数化始发地和目的地，实现每人买 3 张票。
 - ✓ 为方便看那个用户订的票，参数化 lastname，使用用户名即可
 - ✓ 需要关联班次，注意班次使用了多次

7 用户自定义变量

- 根据业务流程制作成测试脚本，想要移植到其他测试环境时，由于数据发生了相应的变化，例如 IP 地址、请求路径等，这时候可以将 IP 地址、请求路径等做成用户自定义变量。
- Thread Group→配置元件→用户自定义变量
- 【例】将脚本中的 localhost 改为服务器 IP。
 - ✓ HTTP 请求中的名称不需要改

五、 场景设计

1 集合点

- 通过定时器完成。
- Synchronizing Timer（同步定时器）
 - ✓ 用来保证我们的取样器在同一时刻向服务器发起负载。
 - ✓ Number of Simulated Users to Group by
 - ✧ 设置同步的线程数量。
 - ✓ Timeout in milliseconds
 - ✧ 超时时间，单位为毫秒。
- 【例】参数化登录，为登录设置集合点，运行负载测试（取消每人订 3 张票）。

2 IP 欺骗

- 在 jmeter 所在计算机中添加多个 IP
 - ✓ netsh interface ip add address "本地连接" 172.16.0.2 255.255.0.0
- 创建参数化文件，存储多个 ip 地址
 - ✓ IP 地址必须跟上面添加的计算机 IP 完全一致。
- 配置元件
 - ✓ 指定参数化信息
 - ✓ 放在所有请求的最前面
- HTTP 请求
 - ✓ Basic
 - ✧ Implementation→选择 Java 以外的内容，否则可能看不到 IP
 - ✓ Advanced
 - ✧ SourceAddress→IP/HOSTNAME
 - \${ip 参数名}
 - ✓ 查看"请求"
 - ✧ X-LocalAddress 后面的即为 IP
 - ✧ cmd→netsta -au | find "你自己的 ip 网段号"

3 多机联合负载

- 【例 1】使用多机联合负载运行 200 用户的负载测试。
 - ✓ 两台计算机：控制机（这里充当负载机）、负载机，配置同网段 IP 且连通，双向关闭防火墙
 - ✓ 每台计算机各设置 100 个 IP
 - ✓ 每台计算机准备一份 ip 参数文件
 - ✧ 各含 100 个 ip，不重复，不冲突
 - ✧ 文件名必须相同
 - ✓ 每台计算机准备一份 users.txt 账号密码文件
 - ✧ 名字必须相同
 - ✓ 修改脚本和参数文件的目录为 jmeter\bin

- ✓ 负载机搭建 jmeter 环境（至少安装 Jre）
- ✓ 在 JMeter 控制机运行远程负载机
 - ◇ 控制机
 - 打开 jmeter.properties 文件，搜索 “remote_hosts”，加上远程 JMeter 负载机的 “IP: 1099”，重启 jmeter 生效，本机直接写 ip 或 127 均可
 - 控制机要执行测试的话，需要打开 jmeter-server.bat
 - ◇ 负载机
 - 打开 jmeter-server.bat;
 - ◇ 每个负载机均衡负担设置的线程数
 - 重新启动 jmeter
 - 控制机上线程数设置为：总并发数/负载数。
- ✓ 运行测试



- ✓ 点击黄色三角形图标，可以显示日志，旁边数字表示出错数。
- ✓ 绿色圆圈处的 100/100 中，后一个 100 表示总的线程数（虚拟用户数），前一个 100 表示当前的在线用户数。

六、 场景监控与结果分析

1 添加监听器

- 图形结果、用表格察看结果、聚合报告

2 监控硬件资源

- 解压 ServerAgent-2.2.1.zip 到被监控计算机中
 - ✓ Windows 服务器运行 startAgent.bat 文件
 - ✓ Linux 服务器运行 startAgent.sh 文件
- 将 JMeterPlugins-Standard.jar 包复制到 jmeter 安装目录下的 lib\ext 下
- 重启 jmeter
- 选择监听器 jp@gc-PerfMon Metrics Collector
 - ✓ 单击 Add Row，添加服务器的 ip，选择要监控的 CPU、内存、硬盘、网络等资源

3 结果分析

- 察看结果树-取样器结果
 - ✓ Thread Name

-
- ◇ 线程组名称
 - ✓ Sample Start
 - ◇ 启动开始时间
 - ✓ Load time
 - ◇ 加载时长
 - ✓ Connect time
 - ◇ 连接时长
 - ✓ Latency
 - ◇ 等待时长
 - ✓ Size in bytes
 - ◇ 发送的数据总大小
 - ✓ Headers size in bytes
 - ◇ 发送数据的头部部分大小
 - ✓ Body size in bytes
 - ◇ 发送数据的正文部分大小
 - ✓ Sample Count
 - ◇ 发送统计
 - ✓ Error Count
 - ◇ 错误统计
 - ✓ Response code
 - ◇ 状态码
 - ✓ Response message
 - ◇ 响应信息
 - ✓ Response headers
 - ◇ 响应的头部信息
 - 聚合报告
 - ✓ Label
 - ◇ 线程组中的步骤名
 - ✓ #Samples
 - ◇ 表示一共发出的请求数
 - ✓ Average
 - ◇ 平均响应时间，默认情况下是单个 Request 的平均响应时间（ms）
 - ✓ Median
 - ◇ 中间值，有一半的服务器响应时间低于该值而另一半高于该值。
 - ✓ 90%line
 - ◇ 90%请求的响应时间。
 - ✓ Min
 - ◇ 服务器响应的最短时间。
 - ✓ Max
 - ◇ 服务器响应的最长时间。
 - ✓ Error%
 - ◇ 测试出现的错误请求数量百分比。
 - ◇ 确认是否允许错误的发生或者错误率允许在多大的范围内。
 - ◇ 若出现错误就要看服务端的日志，配合开发查找定位原因。

-
- ✓ Throughput
 - ✧ 简称 tps, 吞吐量
 - ✧ 默认情况下表示每秒处理的请求数, 也就是指服务器处理能力, tps 越高说明服务器处理能力越好。
 - ✧ 吞吐量默认以 requests/second 来衡量, 即每秒多少个请求。
 - ✓ Kb/sec
 - ✧ 以 KB/second 来衡量的吞吐量。
 - 用表格查看结果
 - ✓ Sample#
 - ✧ 每个请求的序号
 - ✓ Start Time
 - ✧ 每个请求开始时间
 - ✓ Thread Name
 - ✧ 每个线程的名称 (代表一个虚拟用户)
 - ✓ Label
 - ✧ Http 请求名称
 - ✓ Sample Time
 - ✧ 每个请求所花时间, 单位毫秒
 - ✓ Status
 - ✧ 请求状态, 如果为勾则表示成功, 如果为叉表示失败。
 - ✓ Bytes
 - ✧ 请求的字节数
 - ✓ Latency
 - ✧ 等待时长
 - ✓ Connect time
 - ✧ 连接时长
 - ✓ 样本数目
 - ✧ 也就是请求个数, 成功的情况下等于你设定的并发数目乘以循环次数
 - ✓ 最新样本
 - ✧ 表示服务器响应最后一个请求的时间
 - ✓ 平均
 - ✧ 每个线程请求的平均时间
 - ✓ 偏离
 - ✧ 服务器响应时间变化、离散程度测量值的大小

七、 非 GUI 运行测试

1 修正脚本和设置

1.1 修改请求名称

- 为方便将来的结果分析, 建议修改请求名称

1.2 解决察看结果树中的汉字乱码问题

- 修改 jmeter-properties 文件
 - ✓ sampleresult.default.encoding=utf-8

1.3 日志中显示参数和断言信息

- 修改 jmeter-properties 文件
 - ✓ log_level.jmeter=DEBUG
- 日志中的部分文本含义
 - ✓ .ResponseAssertion
 - ✧ 断言
 - ✓ FileServer: Read
 - ✧ 读文件
- 注意
 - ✓ 运行负载测试时不显示
 - ✓ 在调试的时候可以调整日志级别来看输出,但是正常情况下要将日志的级别调高(INFO),减少日志输出,以提高 jmeter 本身性能

1.4 设置资源监控图的保存位置

- 察看结果树
 - ✓ 无需设置保存文件
 - ✓ 可以删除“察看结果树”
- jp@gc - PerfMon Metrics Collector
 - ✓ 设置保存文件
 - ✧ 直接输入路径,不点“浏览”

2 非 GUI 运行脚本

- 进入 jmeter/bin 命目录后,输入如下命令
 - ✓ jmeter -n -t jmx 脚本文件名 -r -l 结果文件名.jtl
 - ✧ -n: 非 GUI 方式运行。nongui
 - ✧ -t: 指定运行的测试脚本地址与名称,可以使用相对路径。
 - ✧ -r: 开启远程负载机
 - 远程机器列表在 jmeter.properties 中指定 remote_hosts
 - 若使用-r,则必须事先开启 jmeter-server.bat
 - 若省略-r,表示不使用远程负载机
 - ✧ -l: 记录测试结果到文件,指定文件地址与名称,可以是相对路径,也可以是绝对路径。

3 导出测试结果

3.1 导出图形结果

- jmeter -g 结果文件名.jtl -o 图形结果存储目录
 - ✓ 无需选监听器（硬件资源监控器需要保留）
 - ✓ 不要导出硬件资源图

3.2 性能结果图

- 打开结果目录中的 index.html 即可查看，IE 浏览器如果看不了，可以换更高版本或换浏览器。

3.3 解决导出结果中的汉字乱码问题

- 替换文件
 - ✓ apache-jmeter-3.0\lib\ext\ApacheJMeter_core.jar

3.4 硬件资源图

- 非 GUI 运行结束正常导出结果，不用导出 PerfMon Metrics Collector
 - ✓ 运行结束后打开 PerfMon Metrics Collector 视图，“浏览”数据文件，再将图存为图片

4 测试结果图

- dashboard 仪表盘
 - ✓ Apdex(Application Performance Index)
 - ◇ 应用程序性能满意度的标准
 - ◇ APDEX 是由 APDEX 公司推出的衡量企业应用程序性能满意度标准的计算方式，将用户的满意度用数字衡量，范围在 0-1 之间。
 - 0 表示所有用户均不满意
 - 1 表示所有用户都满意
 - 设定请求样本目标响应时间为 t ，则可容忍的响应时间上限设定为目标响应时间 t 的 4 倍即 $4t$ ，Apdex 计算公式定义为：(满意的样本数量+可容忍样本数量的一半)/总样本数量
 - 如：总样本数量为 1000，目标时间 $t=3s$ ，假设有 750 个样本响应时间小于等于 t ，150 个样本响应时间在 $3s-12s$ 之间，100 个样本响应时间超过 $12s$ ，则用户满意度为： $(750+150/2)/1000=0.825$
 - ◇ Satisfied（满意）
 - ◇ Toleration threshold（可容忍门槛/上限）
 - ◇ Frustrated threshold（烦躁上限）
 - ✓ Request Summary
 - ◇ 样本请求的成功、失败百分占比图表。

✓ Statistics

✧ 此部分结果展示的是每个样本事务的一些常见的性能测试指标,跟我们通常看到的聚合报告的表格展示非常相近,多了成功与失败的占比。

✓ Errors

✧ 执行结果的错误情况,根据不同的错误类型进行展示。

✧ 四列分别对应:发生错误的类型、错误数量、类型错误占比(相对于错误总数)、类型错误样本占比(相对于所有的请求样本数量)。

➤ Charts

✓ Over Time

✧ Response Times Over Time

● 随时间推移,样本请求响应时间的变化。

✧ Bytes Throughput Over Time

● 随时间推移,网络数据传输(发送、接收,单位:字节)速率的变化。

✧ Latencies Over Time

● 随时间推移,请求样本延迟响应的变化。

✓ Throughput

✧ Hits Per Second

● 每秒点击数。

✧ Codes Per Second

● 随时间推移,每秒响应的状态码数量。

✧ Transactions Per Second

● 每秒响应的事务数。

✧ Response Time Vs Request

● 每秒请求总样本数量的响应时间分位数分布。

✧ Latency Vs Request

● 随每秒样本请求数量变化,延迟请求的成功、失败响应时间。

✓ Response Times

✧ Response Time Percentiles

● 响应时间百分位数分布。

✧ Active Threads Over Time

● 随时间变化,激活线程数变化。

✧ Time Vs Thread

● 随活动线程数变化,平均响应时间变化曲线。

✧ Response Time Distribution

● 响应时间分布。