

Table of Contents

性能测试课程	1.1
第1章-性能测试理论	1.2
性能测试概述	1.2.1
性能测试策略	1.2.2
性能测试指标	1.2.3
性能测试流程	1.2.4
第2章-性能测试工具	1.3
常用性能测试工具	1.3.1
JMeter环境搭建	1.3.2
JMeter功能概要	1.3.3
JMeter元件作用域和执行顺序	1.3.4
JMeter使用示例	1.3.5
JMeter参数化	1.3.6
JMeter断言	1.3.7
JMeter关联	1.3.8
JMeter录制脚本	1.3.9
JMeter直连数据库	1.3.10
JMeter逻辑控制器	1.3.11
JMeter定时器	1.3.12
JMeter分布式	1.3.13
JMeter测试报告	1.3.14
JMeter性能测试常用图表	1.3.15
第3章-项目实战	1.4
项目介绍和部署	1.4.1
性能测试需求分析	1.4.2
性能测试计划	1.4.3
测试用例设计	1.4.4
测试脚本开发	1.4.5
建立测试环境	1.4.6
执行测试脚本	1.4.7
性能测试监控	1.4.8
性能分析和调优	1.4.9
编写测试报告	1.4.10
第4章-Locust框架	1.5
Locust介绍和安装	1.5.1
Locust使用	1.5.2

程序员-软件测试

性能测试课程

课程大纲

章节	知识点	
第 1 章 性能测试理论	1. 性能测试概述 2. 性能测试策略	3. 性能测试指标 4. 性能测试流程
第 2 章 性能测试工具JMeter	1. 常用性能测试工具 2. JMeter环境搭建 3. JMeter基本使用 4. JMeter参数化、断言、关联	5. JMeter录制脚本 6. JMeter直连数据库 7. JMeter分布式 8. JMeter测试报告
第 3 章 项目实战	1. 项目介绍和部署 2. 性能测试需求分析 3. 性能测试计划 4. 测试用例设计 5. 测试脚本开发	6. 建立测试环境 7. 执行测试脚本 8. 性能测试监控 9. 性能分析和调优 10. 编写测试报告
第 4 章 Locust框架	1. Locust介绍和安装	2. Locust使用

课程目标

1. 掌握性能测试理论中常见的性能测试场景、性能测试指标和性能测试流程；
2. 掌握使用JMeter工具发送http接口请求，并掌握断言、关联、参数化等功能；
3. 熟悉性能测试的流程；

程序员-软件测试

第1章-性能测试理论

目标

- 3. 熟悉
- 4. 熟悉

程序员-软件测试

性能测试概述

目标

1. 知道进行性能测试的原因
2. 理解性能测试的概念
3. 理解性能测试和功能测试的区别与联系

1. 为什么要进行性能测试

1.1 业务需求

- 电商双11活动/微信春晚抢红包/12306春运订票
- 当前服务器配置是否支持20000人同时使用
- 技术选型，如编程语言选择Java? Python? PHP?

1.2 招聘需求

- 面试：会性能测试吗?
- 招聘信息：要求会使用性能测试工具Jmeter、LoadRunner

提示：要解决以上问题，必须要进行相关性能测试

2. 性能测试的概念

2.1 什么是性能?

性能：就是软件质量属性中的“效率”特性

效率特性：

- 时间特性：指系统处理用户请求的响应时间
- 资源特性：指系统在运行过程中，系统资源的消耗情况
 - CPU
 - 内存
 - 磁盘IO（磁盘的写入Input和读取Output，简称IO）

2.2 什么是性能测试?

概念：使用自动化工具，模拟不同的场景，对软件各项性能指标进行测试和评估的过程就是性能测试。

1. 后台处理程序的性能（代码性能）
2. 中间件、数据库、架构设计等是否存在瓶颈
3. 服务器资源消耗（CPU、内存、磁盘、网络）

中间件：是提供系统软件和应用软件之间连接的软件。如：Tomcat、Apache...

2.3 性能测试目的

1. 评估当前系统能力
 - 例如：验收第三方提供的软件

程序员-软件测试

- 例如：获取关键的性能指标，与其他类似产品进行比较
- 2. 寻找性能瓶颈，优化性能
- 3. 评估软件是否能够满足未来的需要

3. 性能测试与功能测试

3.1 焦点不一样

- 功能测试：验证软件系统操作功能是否符合产品功能需求规格，主要焦点在功能（正向、逆向）；
- 性能测试：验证软件系统是否满足业务需求场景，主要焦点是业务场景的满足(时间、资源)；

3.2 关系

- 功能测试和性能测试是相辅相成的，对于一款优秀的软件产品来讲，它们是不可减少的2个重要测试环节；
- 注意：一般新项目中，先功能测试通过后，再进行性能测试。

程序员-软件测试

性能测试策略

目标

1. 理解性能测试中常见测试策略

1. 性能测试策略

1. 基准测试
2. 负载测试
3. 稳定性测试
4. 其他：并发测试、压力测试、容量测试等

1.1 基准测试

基准测试：

- 狭义上讲：也是单用户测试，测试环境确定以后，对业务模型中的重要业务做单独的测试，获取单用户运行时的各项性能指标。（进行基础的数据采集）
- 广义上讲：是一种测量和评估软件性能指标的活动。你可以在某个时刻通过基准测试建立一个已知的性能水平（称为基准线），当系统的软硬件环境发生变化之后再进行一次基准测试以确定那些变化对性能的影响。

基准测试数据的用途：

1. 为多用户并发测试和综合场景测试等性能分析提供参考依据
2. 识别系统或环境的配置变更对性能响应带来的影响
3. 为系统优化前后的性能提升/下降提供参考指标

1.2 负载测试

说明：通过逐步增加系统负载，测试系统性能的变化，并最终确定在满足系统的性能指标情况下，系统所能够承受的最大负载量的测试。

负载：指向服务器发送的请求数量，请求越多，负载越高

注意：负载测试关注的重点是逐步增加压力

1.3 稳定性测试

说明：稳定性测试是指，在服务器稳定运行（用户正常的业务负载下）的情况下进行长时间测试，并最终保证服务器能满足线上业务需求。时长一般为1天、一周等。

1.4 其他测试策略

性能测试中，测试策略其实有很多种，但是掌握基础的用法后，其他不同名称的测试策略只是基础用法的一个变形用法。

- 并发测试：并发测试是指在极短的时间内，发送多个请求，来验证服务器对并发的处理能力。如：抢红包、抢购、秒杀活动等。
- 压力测试：压力测试是在强负载（大数据量、大量并发用户等）下的测试，查看应用系统在峰值使用情况下操作行为，从而有效地发现系统的某项功能隐患、系统是否具有好的容错能力和可恢复能力。压力测试分为高负载下的长时间（如24小时以上）的稳定性压力测试和极限负载情况下导致系统崩溃的破坏性压力测试。
- 容量测试：关注软件的极限压力下的各个极限参数值，例如：最大TPS，最大连接数，最大并发数，最多数据条数等。

程序员-软件测试

程序员-软件测试

性能测试指标

目标

1. 掌握常用的性能测试指标
2. 理解性能测试指标中每种指标的含义

1. 性能测试指标介绍

1.1 什么是指标

说明：一些经过运算得出的结果，来衡量某种操作性能统称；比如：错误率 0.5%

1.2 性能指标

1. 响应时间
2. 并发数
3. 吞吐量
4. 点击数
5. 错误率
6. 资源利用率
7. PV和UV

2. 常用性能指标

2.1 响应时间

说明：响应时间指用户从客户端发起一个请求开始，到客户端接收到从服务器端返回的结果，整个过程所耗费的时间。

组成：响应时间 = 网络时间 + 应用程序处理时间

2.2 并发数

说明：并发测试的用户数

扩展：

- 系统用户数：系统注册的总用户数据
- 在线用户数：某段时间内访问系统的用户数，这些用户并不一定同时向系统提交请求
- 并发用户数：某一物理时刻同时向系统提交请求的用户数

2.3 吞吐量

说明：吞吐量（Throughput）指的是单位时间内处理的客户端请求数量，直接体现软件系统的性能承载能力

注意：

1. 从业务角度来看，吞吐量也可以用“业务数/小时”、“业务数/天”、“访问人数/天”、“页面访问量/天”来衡量
2. 从网络角度来看，还可以用“字节数/小时”、“字节数/天”等来衡量网络的流量
3. 从技术指标来看，可以用每秒事务数（TPS）和每秒查询数（QPS）来衡量服务器具体性能处理能力

TPS

程序员-软件测试

说明：Transactions Per Second，每秒事务数（单位时间内系统处理的客户端请求的事务次数）

计算：TPS = 并发数/平均响应时间

事务：就是业务请求，对应一个或者多个操作。如支付请求，包括服务器查询用户余额，支付安全校验等多个操作。一个业务请求发送给服务器后，最终会定位到服务器对应的业务请求的代码，既有可能是一段代码也有可能是多段代码。

TPS归属吞吐量

QPS

说明：QPS(Query Per Second)每秒查询数

应用：控制服务器每秒处理指定请求数（如：控制服务器达到每秒60QPS，服务器的性能各项性能指标是否正常）。（衡量web服务器处理能力一个重要指标）

2.4 点击数

说明：点击数是衡量Web服务器处理能力的一个重要指标。

提示：

1. 点击数不是通常一般人认为的访问一个页面就是1次点击数，点击数是该页面包含的元素（图片、链接、框架等）向Web服务器发出的请求数量。
2. 通常我们也用每秒点击次数（Hits per Second）指标来衡量Web服务器的处理能力。

注意：只有web项目才有此指标。

2.5 错误率

说明：错误率指系统在负载情况下，失败业务的概率。错误率=(失败业务数/业务总数)*100%。

提示：

1. 不同系统对错误率要求不同，但一般不超过千分之五；
2. 稳定性较好的系统，其错误率应该由超时引起，即为超时率。

2.6 资源利用率

说明：是指系统各种资源的使用情况，一般用“资源的使用量/总的资源可用量×100%”形成资源利用率的数据。

提示：通常，没有特殊需求的话

- 1). 建议CPU不高于80%(±5)
- 2). 内存不高于80%
- 3). 磁盘不高于90%
- 4). 网络不高于80%

程序员-软件测试

性能测试流程

目标

1. 掌握性能测试流程

思考：为什么要掌握性能测试流程？

1. 功能测试需要按照流程来推进，性能测试也同样，一套完整的测试流程是一次成功性能测试的基石
2. [面试]简述下性能测试过程...

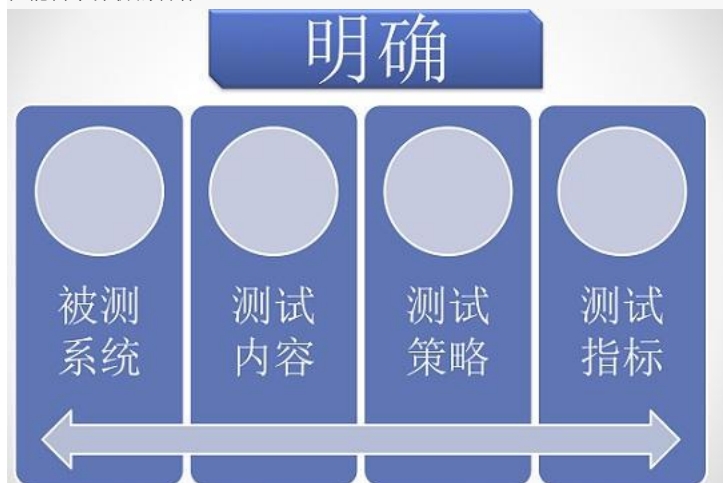
1. 性能测试流程

1. 性能测试需求分析
2. 性能测试计划及方案
3. 性能测试用例
4. 测试脚本编写/录制
5. 建立测试环境
6. 执行测试脚本
7. 性能测试监控
8. 性能分析和调优
9. 性能测试报告总结

提示：使用不同的性能测试工具时，主要流程是不变的。

说明：性能需求分析是整个性能测试工作开展的基础，性能需求分析做的好不好直接影响到性能测试的结果。

性能需求分析的目标：



1. 熟悉被测系统
 - 熟悉被测系统的业务功能

- 熟悉被测系统的技术架构
- 2. 明确性能测试内容
 - 从业务角度明确测试内容
 - 确定关键业务。即：用户使用频率较高的业务功能
 - 从技术角度明确测试内容
 - 如：通常逻辑复杂度较高的业务也是CPU密集运算较大的地方，考量服务器CPU在预定性能指标下是否达标

10

程序员-软件测试

- 如：通常数据量较大的业务很占用系统内存，考量服务器内存存在预定性能指标下是否达标
- 3. 明确性能测试策略
 - 负载测试
 - 稳定性测试
 - 并发测试
- 4. 明确性能测试的指标
 - 无明确需求指标
 - 通过查找相关资料，和类似的系统对比，以及对未来流量的预估，确定性能测试需求的指标
 - 有明确需求指标
 - 例如，类似如下指标
 - 下订单业务并发20个用户
 - 平均响应时间要小于等于3s
 - 事务成功率为100%
 - CPU使用率小于等于85%
 - 只需要根据执行分析与预期指标做对比，如果有不满足的，就需要分析问题所在

1.2 性能测试计划及方案

说明：性能测试实施第一份文档，也是最重要的一份文档。

主要内容：

1. 项目背景
2. 测试目的
3. 测试范围
4. 测试策略
5. 风险控制
6. 交付清单
7. 进度与分工

1.3 性能测试用例

程序员-软件测试

性能测试用例模板

1.4 测试脚本编写/录制

用例名称	获取首页数据			
用例编号	Index001			
用例描述	TPS达到100的情况下，进入首页的时间不超过5s			
前置条件	首页相关的商品数据已经配置完成			
用例步骤	动作	期望的性能		
1	进入商城首页	<5s		
2				
并发用户数与事务响应				
并发用户数	事务平均响应时间	事务最大响应时间	平均每秒处理事务数（TPS）	事务成功率
5				
10				
30				
50				
100				
并发用户数与服务器性能				
并发用户数	CPU利用率	内存利用率	磁盘IO情况	其他参数
5				
10				
30				
50				
100				
并发用户数与数据库性能				
并发用户数	CPU利用率	内存利用率	磁盘IO情况	其他参数
5				
10				
30				
50				
100				

说明：性能测试用例编写完成以后，接下来就需要结合用例的需要，进行测试脚本的编写工作。

提示：录制或编写，根据不同的工具要注意代码冗余。

1.5 建立测试环境

说明：在进行性能测试之前，需要先完成性能测试环境的搭建工作，测试环境一般包括硬件环境、软件环境及网络环境

提示：一般情况下可以要求运维和开发工程师协助完成

1.6 执行测试脚本

说明：先保证脚本调试通过之后，才能进入正式压测阶段

执行测试脚本时，要先进行性能运行场景的设置，再运行脚本

1.7 性能测试监控

性能监控就是监控服务器的各项性能指标。例如：监控CPU、内存、网络、TPS、磁盘IO等

1.8 性能分析和调优

说明：性能测试分析人员经过对结果的分析以后，有可能提出系统存在性能瓶颈。

提示：

1. 调优人员(开发人员、数据库管理员、系统管理员、网络管理员、性能测试分析人员)相关人员对系统进行调整;

程序员-软件测试

2. 验证-性能测试人员继续进行第二轮、第三轮.....的测试，与以前的测试结果进行对比，从而确定经过调整以后系统的性能是否有提升。

系统调优由易到难的先后顺序如下：

1. 硬件问题
2. 网络问题
3. 应用服务器、数据库等配置问题
4. 源代码、数据库脚本问题
5. 系统构架问题

1.9 性能测试报告总结

性能测试总结要包含以下内容：

1. 性能测试需求覆盖情况，测试过程回顾，及测试中出现的问题（如何去分析、调优、解决的）---基本要求
2. 性能测试过程中遇到各类风险是如何控制的; 目前是否还有其他的性能风险存在
3. 经过该项目性能测试后，有那些经验和教训等内容

目标

- 3. 掌握
- 4. 掌握

目标

1. 了解市面主流测试工具
2. 掌握JMeter工具的优缺点

1. 主流性能测试工具

- LoadRunner
- JMeter [本阶段学习]

1.1 LoadRunner

- HP LoadRunner是一种工业级标准性能测试负载工具，可以模拟上万用户实施测试，并在测试时可实时检测应用服务器及服务器硬件各种数据，来确认和查找存在的瓶颈
- 支持多协议：Web(HTTP/HTML)、Windows Sockets、FTP、ODBC、MS SQL Server等协议
- 最初是Mercury公司采用C语言编写,现被HP公司收购

优点：

1. 多用户（支持数量单位万）
2. 详细分析报表
3. 支持ip欺骗

缺点：

1. 收费
2. 体积庞大（单位GB）
3. 无法定制功能

1.2 JMeter

- JMeter是Apache组织开发的基于Java的开源软件，用于对系统做功能测试和性能测试。
- 它最初被设计用于Web应用测试，但后来扩展到其他测试领域，例如静态文件、Java 程序、shell 脚本、数据库、FTP、Mail等。

优点：

1. 免费
2. 开源
3. 跨平台
4. 丰富
5. 应用
6. 易学

缺点：

2. 分析

程序员-软件测试

JMeter环境搭建

目标

1. 掌握如何搭建Jmeter环境
2. 掌握JMeter基本使用流程

1. 安装JDK

JDK(Java Development Kit) 是 Java 语言的软件开发工具包

1.1 JDK下载

- 官网: <http://www.oracle.com/>
- JDK8下载地址: <http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html>

提示: 下载时注意电脑系统是32位或64位

1.2 安装JDK

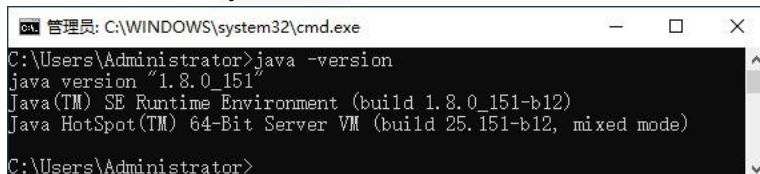
双击安装包进行安装, 所有步骤选择默认选项即可

1.3 配置环境变量

- 
- 在Path中添加: 

1.4 校验

打开命令行窗口, 输入 `java -version`, 校验命令能否正常执行以及版本信息是否一致



```
管理员: C:\WINDOWS\system32\cmd.exe
C:\Users\Administrator>java -version
java version "1.8.0_151"
Java(TM) SE Runtime Environment (build 1.8.0_151-b12)
Java HotSpot(TM) 64-Bit Server VM (build 25.151-b12, mixed mode)
C:\Users\Administrator>
```

2. 安装JMeter

2.1 下载

官网下载地址: http://jmeter.apache.org/download_jmeter.cgi

2.2 安装

把下载的安装包, 解压到某一目录即可

提示: 安装目录中不要出现空格, 这将导致远程测试出现问题

2.3 环境配置

程序员-软件测试

Jmeter环境

1. 新建JMeter_HOME变量, 并添加jmeter所在目录 如: -> F:\Jmeter\apache-jmeter-5.1.1
2. PATH-> .;%JMeter_HOME%\lib\ext\ApacheJMeter_core.jar;%JMeter_HOME%\lib\jorphan.jar;%JMeter_HOME%\lib\logkit-2.0.jar;

2.4 启动验证

JMeter启动有多种方式，进入JMeter安装目录下的bin目录

- 双击 `j`
- 双击 `ApacheJMeter.jar` 选择使用 `java` 程序打开
- 命令行输入: `java -jar ApacheJMeter.jar`

目标

3. 了解

1. JMeter文件目录介绍

1.1 bin目录

存放可执行文件和配置文件

```
jmeter.bat: windows的启动文件
jmeter.log: 日志文件
jmeter.sh: linux的启动文件
jmeter.properties: 系统配置文件
jmeter-server.bat: windows分布式测试要用到的服务器配置
jmeter-serve: linux分布式测试要用到的服务器配置
```

1.2 docs目录

docs: 是JMeter的api文档，可打开api/index.html页面来查看

1.3 printable_docs目录

- printable_docs的**usermanual**子目录下的内容是JMeter的用户手册文档
- usermanual下**component_reference.html**是最常用到的核心元件帮助文档。

提示: *printable_docs*的*demos*子目录下有一些常用的**JMeter**脚本案例，可以作为参考

1.4 lib目录

该目录用来存放JMeter依赖的jar包和用户扩展所依赖的jar包

2. 修改默认配置

2.1 汉化配置

实现JMeter界面的汉化包含两种方式:

1. 临时性
2. 永久性

- 临时性: 启动JMeter->选择菜单'Options'->Choose Language->Chinese (Simplified)
- 永久性:
 - 找到jmeter安装目录下的bin目录,
 - 打开jmeter.properties文件, 把第37行修改为"language=zh_CN",
 - 重启JMeter即可

程序员-软件测试

2.2 修改主题

JMeter默认主题是黑色的，可以通过以下步骤修改:

启动JMeter -> 选择菜单'选项' -> 外观 -> Windows(选择自己喜欢的主题即可)

程序员-软件测试

JMeter元件作用域和执行顺序

目标

1. 熟悉元件的类型与各自的功能
2. 熟悉元件之间彼此作用域
3. 掌握元件执行顺序

1. 元件的基本介绍

元件：多个类似功能组件的容器（类似于类）。

常见的元件类型有：

1. 取样器
2. 逻辑控制器
3. 前置处理器
4. 后置处理器
5. 断言
6. 定时器
7. 测试片段
8. 配置元件
9. 监听器

组件：实现独立的某个功能（类似于方法）

2. 元件作用域

在JMeter中，元件的作用域是靠测试计划的树形结构中元件的父子关系来确定的。

提示：核心是取样器，其他组件都是以取样器为核心运行的，组件添加的位置不同，生效的取样器也不同。

作用域的原则

1. 取样器：元件不和其他元件相互作用，因此不存在作用域的问题；
2. 逻辑控制器：元件只对其子节点中的取样器和逻辑控制器作用；
3. 其他六大元件：除取样器和逻辑控制器元件外，如果是某个取样器的子节点，则该元件对其父子节点起作用；
4. 如果其父节点不是取样器，则其作用域是该元件父节点下的其他所有后代节点（包括子节点，子节点的子节点等）；

提示：以上元件中还没开始学习，暂时理解jmeter这种树形结构影响作用域即可。

3. 元件执行顺序

1. 配置元件(config elements)
2. 前置处理程序(Per-processors)
3. 定时器(timers)
4. 取样器(Sampler)
5. 后置处理程序(Post-processors)
6. 断言(Assertions)
7. 监听器(Listeners)

提示：

1. 前置处理器、后置处理器、断言等元件功能对取样器起作用（如果在它们的作用域内没有任何取样器，则不会被执行）
2. 如果在同一作用域范围内有多个同一类型的元件，则这些元件按照它们在测试计划中的上下顺序依次执行

目标

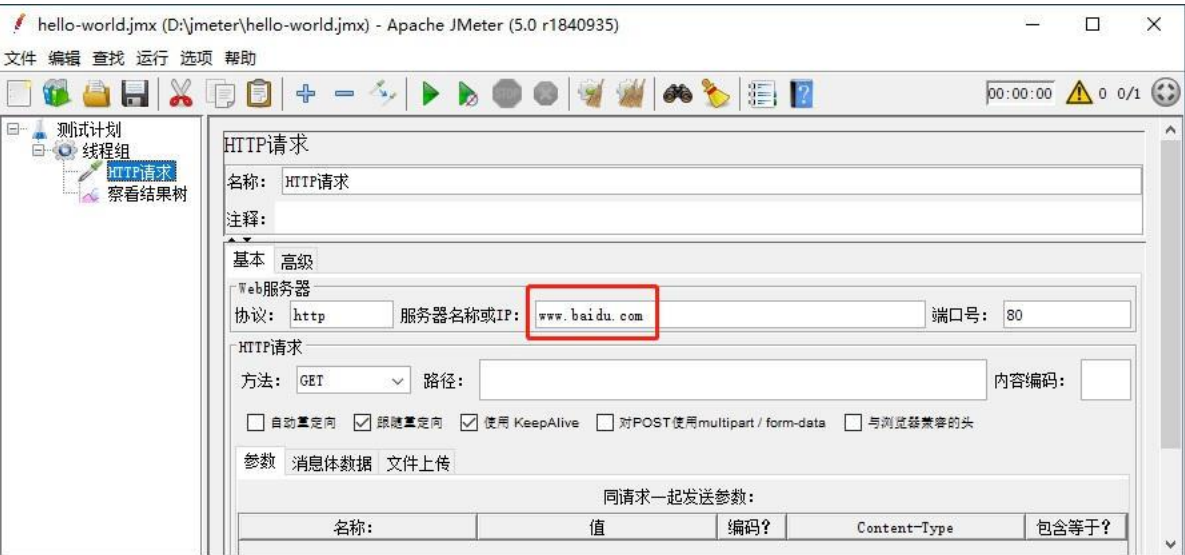
3. 掌握

1. JMeter第一个案例

需求：使用JMeter访问百度首页接口，并查看请求和响应信息

操作步骤

- 1. 启动JMeter
- 2. 在‘测试计划’下添加‘线程组’
- 3. 在‘线程组’下添加‘HTTP请求’取样器
- 4. 填写‘HTTP请求’的相关请求数据
- 5. 在‘线程组’下添加‘察看结果树’监听器
- 6. 点击‘启动’按钮运行，并查看结果



2. 重点组件

- 线程组
- HTTP取样器
- 察看结果树

提示：无论哪个案例基本都需要以上三个组件，在这里先讲解下以上组件。

2.1 线程组

程序员-软件测试

说明：线程组是控制JMeter将用于执行测试的线程数，也可以把一个线程理解为一个测试用户。

2.1.1 添加线程组

位置：右键点击‘测试计划’ --> 添加 --> 线程（用户） --> 线程组

2.1.2 线程组的特点

- 模拟多人操作
- 线程组可以添加多个，多个线程组可以并行或串行
- 取样器（请求）和逻辑控制器必须依赖线程组才能使用
- 线程组下可以添加其他元件下组件

2.1.3 线程组分类

- 线程组
普通的、常用的线程组，可以看做一个虚拟用户组，线程组中的每一个线程都可以理解为一个虚拟用户
- setUp线程组
一种特殊类型的线程组，可用于执行预测试操作
- tearDown线程组
一种特殊类型的线程组，可用于执行测试后工作

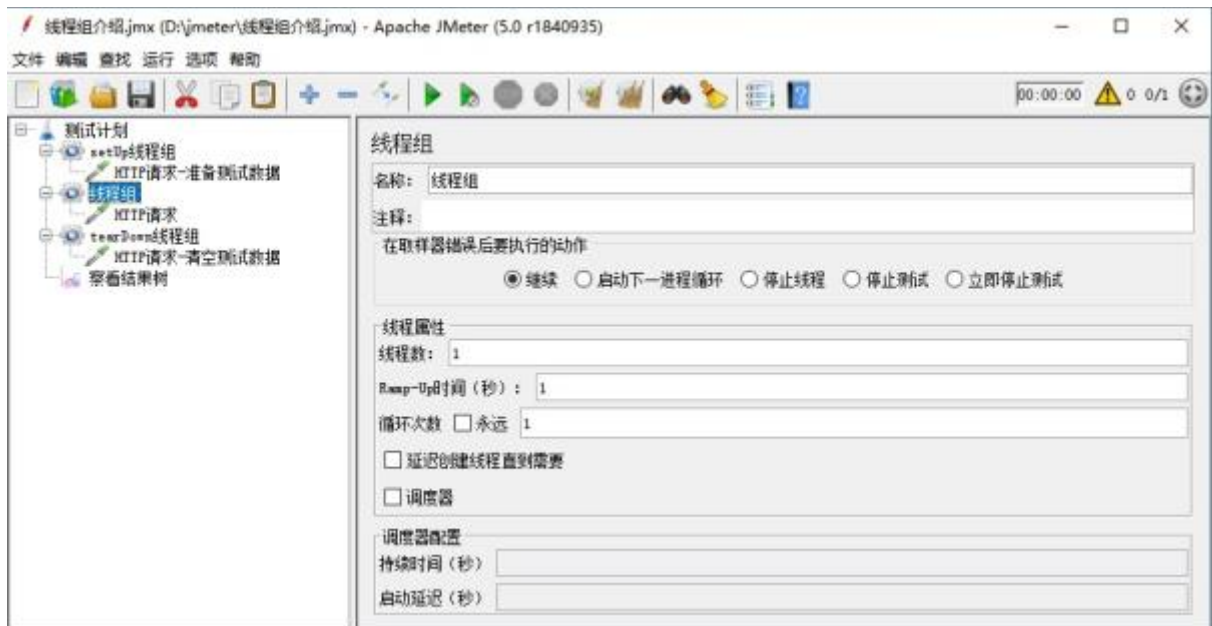
2.1.4 线程组参数详解

2.1.5 取样器错误后要执行的动作

- 继续：如果取样器里的执行出现错误失败的时候，请求不会停止，继续执行。
- 启动下一进程循环: 忽略错误，线程当前循环错误，执行下一个循环。
- 停止线程： 只限当前线程停止，不影响其他线程执行
- 停止测试： 当前执行的线程全部执行完毕后结束
- 立即停止测试： 立刻停止

2.1.6 线程属性

- 线程数：虚拟用户数
- Ramp-Up时间(秒)：启动全部虚拟用户数所需要的时间
- 循环次数：指定次数或勾选永远
- 延迟创建线程直到需要：测试开始的时候，所有线程就被创建完。勾选了此选项，那么线程只会在合适的需要用到时候创建。



24

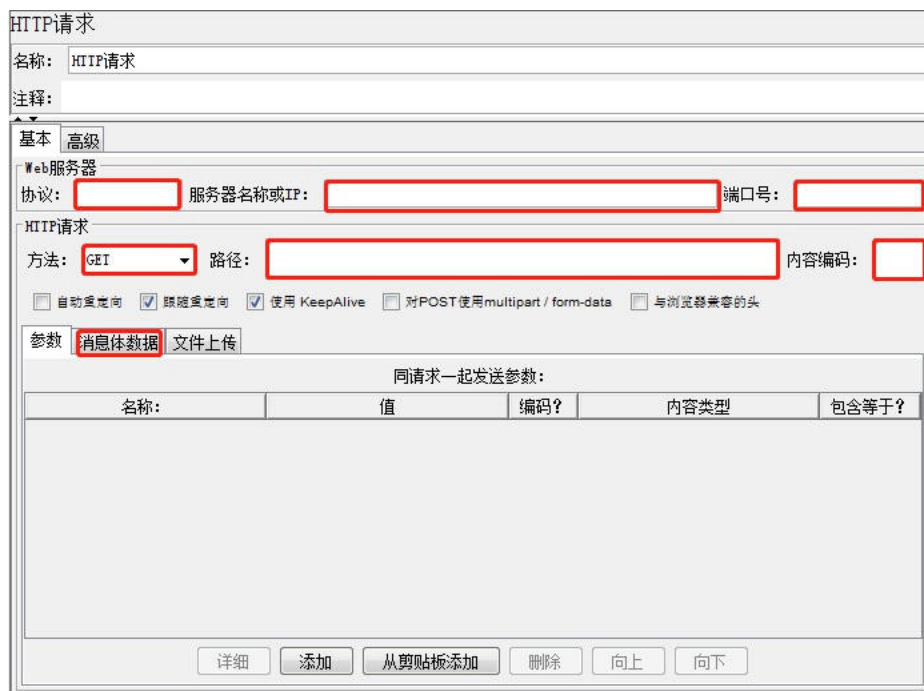
程序员-软件测试

- 调度器：勾选后，调度器配置才能使用；

2.1.7 调度器配置

- 持续时间（秒）：设置脚本压测持续时间
- 启动延迟（秒）：启动延迟时间

2.2 HTTP请求

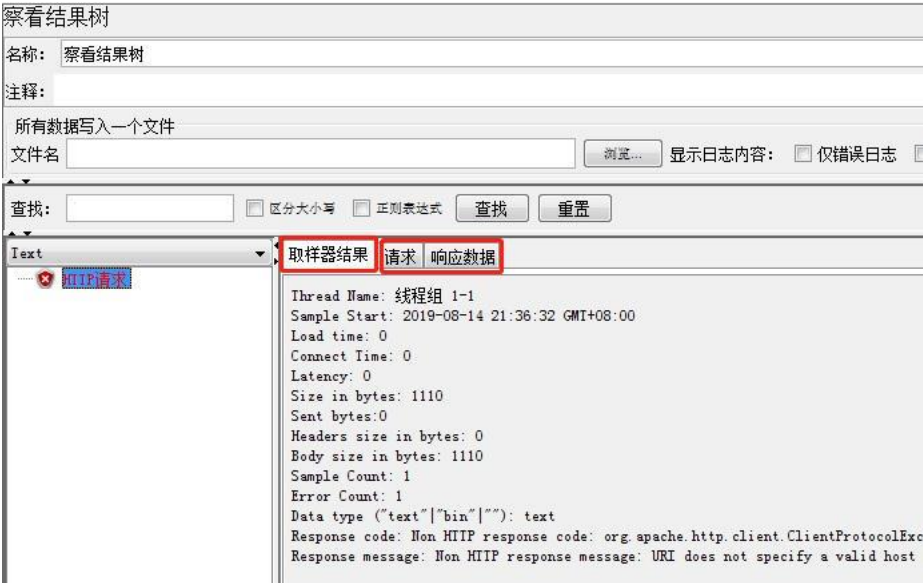


位置： 选中线程组->右键->添加->取样器->HTTP请求 作用： 向服务器发送http及https请求

协议：向目标服务器发送HTTP请求时的协议，可以是http或者是https，默认值为http。
服务器名称或IP：HTTP请求发送的目标服务器名称或IP地址。
端口号：目标服务器的端口号，默认值为80。
方法：发送HTTP请求的方法，可用方法包括GET、POST、HEAD、PUT、OPTIONS、TRACE、DELETE等。
路径：目标URL路径（不包括服务器地址和端口）
Content encoding：内容的编码方式，默认值为iso8859
同请求一起发送参数：GET请求时url中附带参数可以通过此方式添加
消息体数据：POST/PUT请求JSON数据存放地

2.3 查看结果树

程序员-软件测试



位置： 选中测试计划/线程组->右键->添加->监听器->察看结果树 作用： 查看请求请求和响应结果

- 取样结果：查看响应信息头信息、响应状态码
- 请求：查看请求相关信息（url、方法、参数）
- 响应：查看响应信息

程序员-软件测试

JMeter参数化

目标

1. 掌握JMeter中常用的参数化方式

思考：如果循环访问某一请求10次，要求每次请求发送不同的参数值，该怎么做？

1. JMeter参数化常用方式

- 用户定义的变量
- 用户参数
- CSV Data Set Config
- 函数

2. 用户定义的变量

添加方式：测试计划 --> 线程组--> 配置元件 --> 用户定义的变量

2.1 场景

- 请求：<https://www.baidu.com:443>
- 要求：使用用户定义的变量配置被测系统的协议、域名和端口

2.2 操作步骤

1. 添加线程组
2. 添加用户定义的变量
3. 添加HTTP请求
4. 添加查看结果树

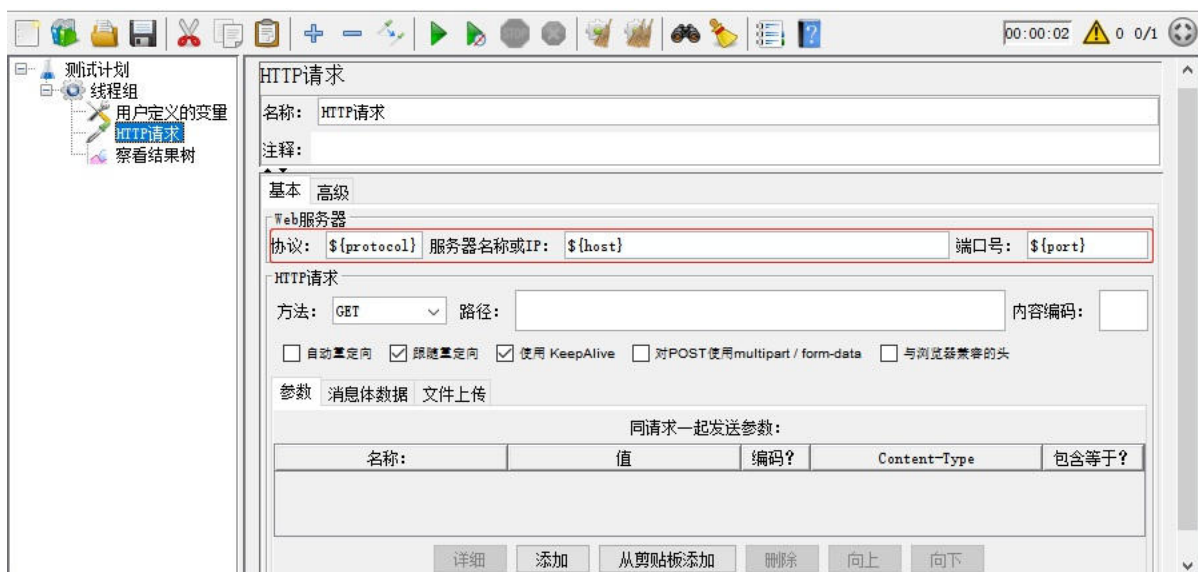
用户定义的变量



HTTP请求设置

27

程序员-软件测试



3. 用户参数

添加方式：测试计划 --> 线程组--> 前置处理器 --> 用户参数

3.1 场景

- 请求：<https://www.baidu.com>
- 要求：第一次请求附带参数：name="张三"&age=28;第二次请求附带参数：name="李四"&age=30

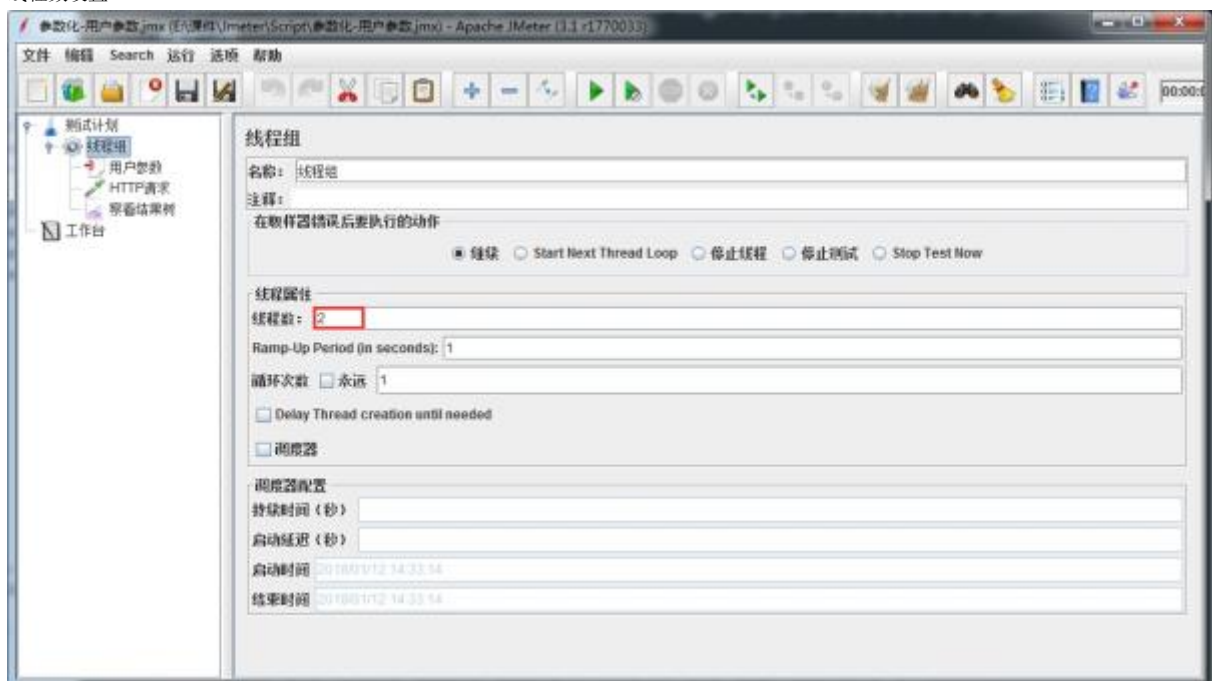
2.2 操作步骤

1. 添加线程组
2. 添加用户参数
3. 添加HTTP请求
4. 添加查看结果树

线程组设置

程序员-软件测试

线程数设置: 2



用户参数设置

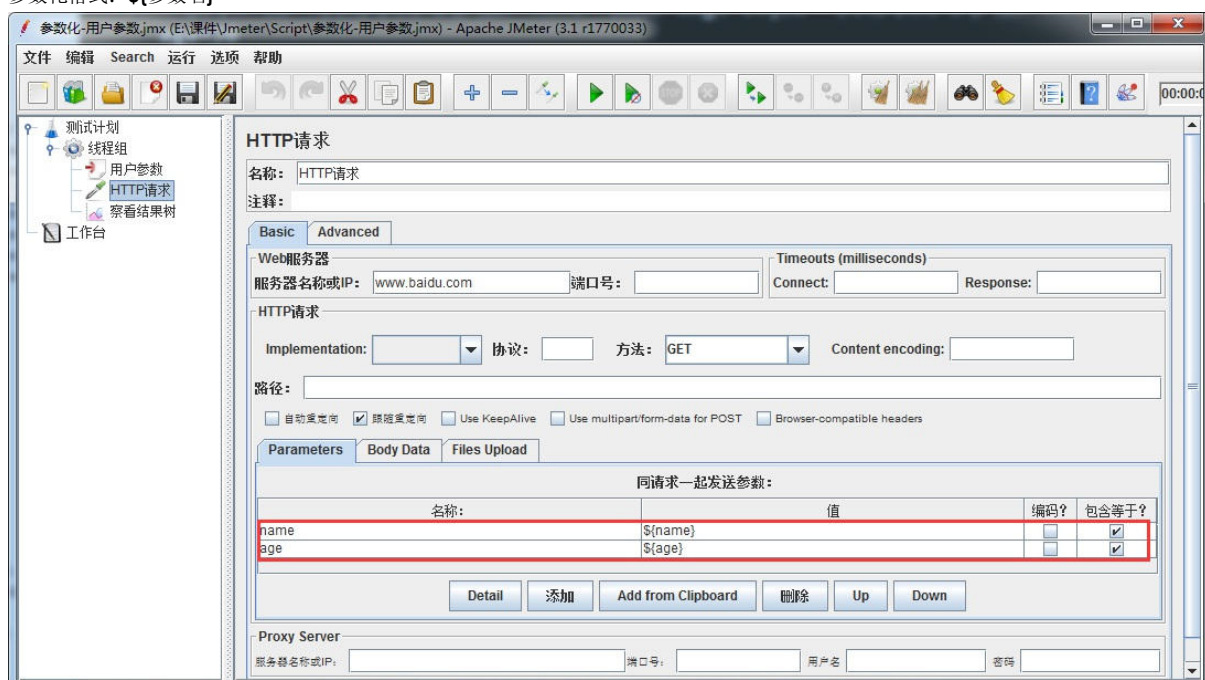


HTTP请求设置

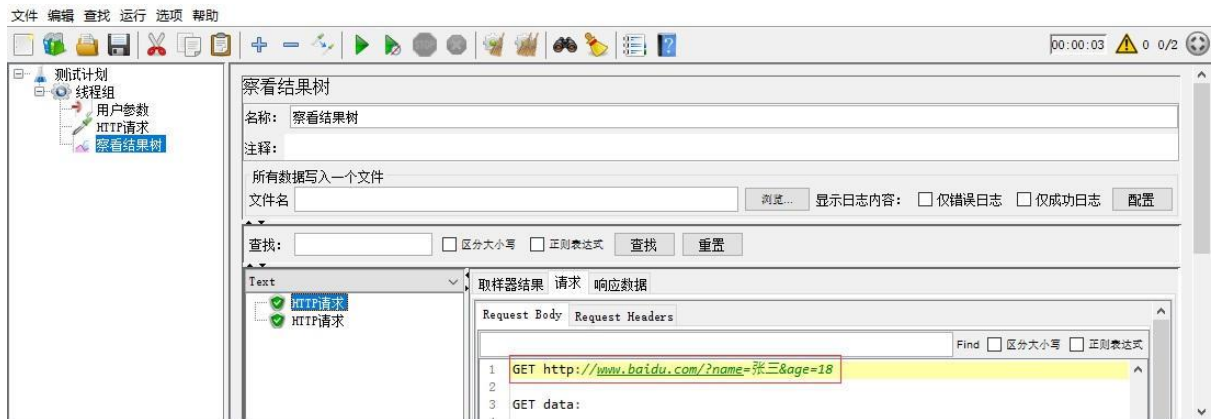
29

程序员-软件测试

参数化格式: \${参数名}



运行结果



3. CSV 数据文件设置

添加方式：测试计划 --> 线程组 --> 配置元件 --> CSV 数据文件设置

3.1 场景

- 请求：<https://www.baidu.com>
- 要求：循环3次,每次请求时附带参数username,password,code的值不相同

操作步骤

1. 定义CSV数据文件
2. 添加线程组
3. 添加CSV 数据文件设置
4. 添加HTTP请求
5. 添加查看结果树

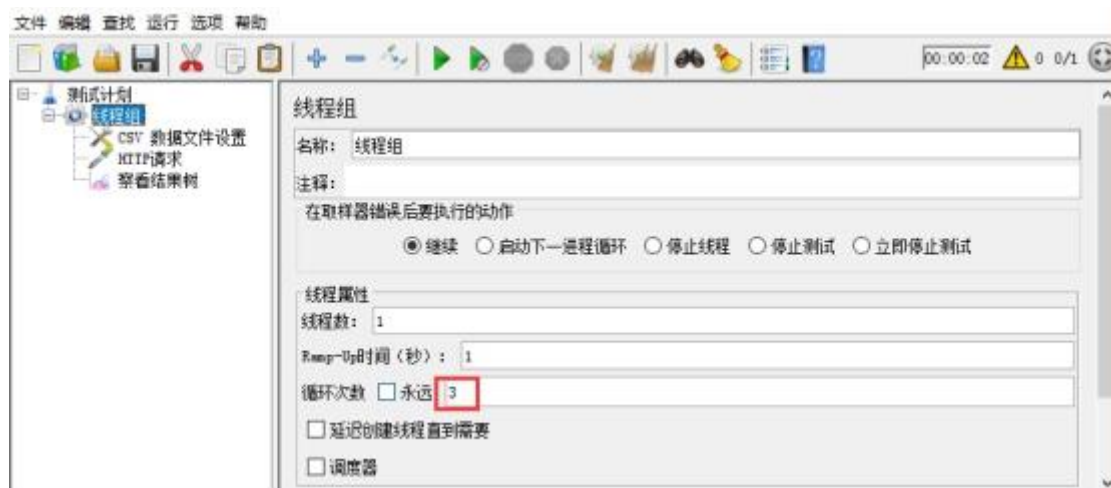
30

程序员-软件测试

定义CSV数据文件

```
test_data.csv
1 user01,123456,0000
2 user02,123456,1111
3 user03,123456,2222
```

线程组设置



CSV 数据文件设置



HTTP请求设置



3.2 参数详解(CSV 数据文件设置)

- 文件名: CSV文件路径
- 文件编码: 文件编译字符编码, 一般设置UTF-8
- 变量名称: 多个变量时, 使用英文逗号分隔
- 忽略首行: **True**为忽略, **False**为不忽略, 默认值: **False**
- 分隔符: 如文件中使用的是逗号分隔, 则填写逗号; 如使用的是制表符, 则填写\t;
- 是否允许带引号: CSV文件中的内容是否允许带引号
- 遇到文件结束符再次循环: 当读取文件到结尾时, 是否再从头读取文件, **False**=当读取文件到结尾时, 停止读取文件
- 遇到文件结束符停止线程: 当“遇到文件结束符再次循环”一项为**False**时起效; **True**:当读取文件到结尾时, 停止进程
- 线程共享模式: 共享模式一般默认即可
 - 所有线程: 该文件在所有线程之间共享, 所有线程循环取值, 线程一取第一行, 线程二取下一行
 - 当前线程组: 各个线程组分别循环取值
 - 当前线程: 每个文件分别为每个线程打开

4. 函数(__counter)

计数函数, 一般做执行次数统计使用;

位置: 在菜单中选择--> 选项 --> 函数助手对话框

4.1 函数助手



32

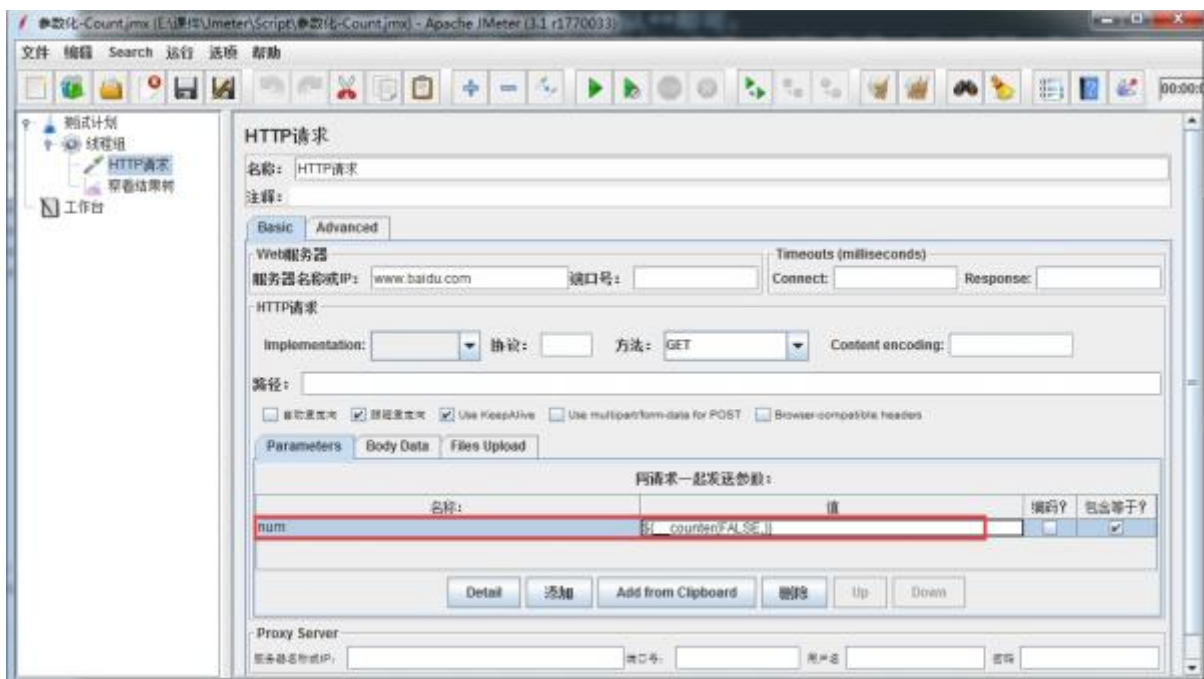
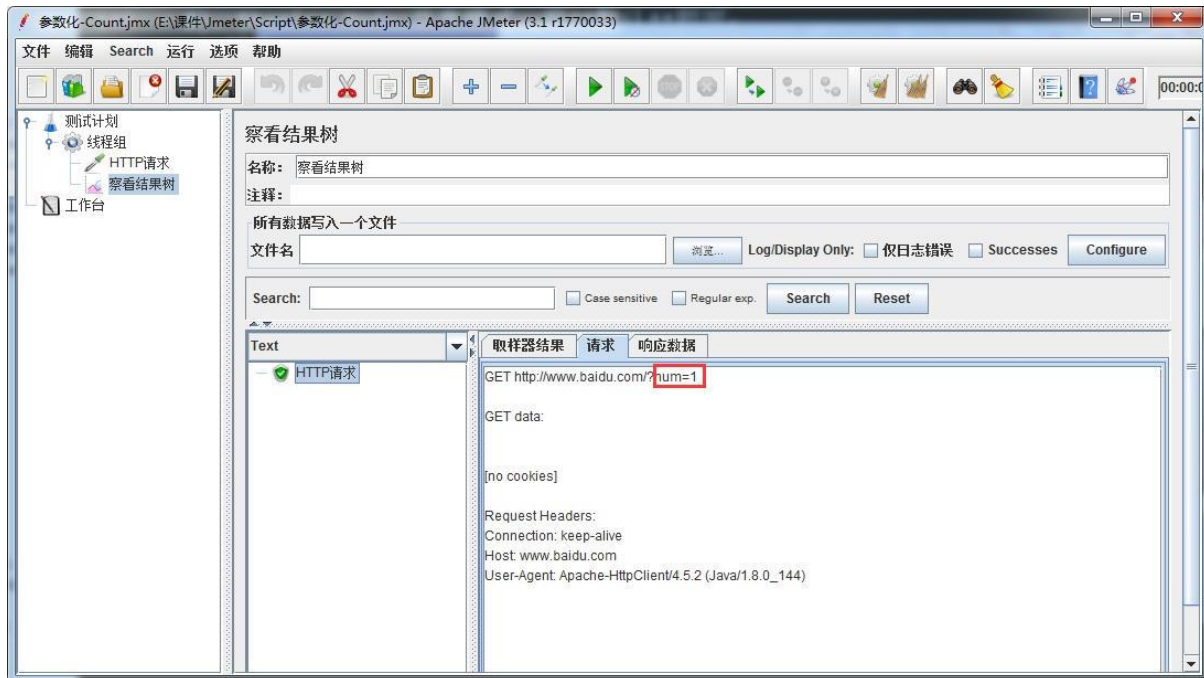
程序员-软件测试

参数设置

- **TRUE**, 每个用户有自己的计数器; **FALSE**, 使用全局计数器
- **Name of variable in which to store the result (optional)**:用于存储结果的变量名(可选)
- 生成-复制

4.2 参数化调用

4.3 运行结果



5. CSV和用户定义的变量作用域问题

33

程序员-软件测试

CSV的作用域是针对线程的，只有两种情况：

- 对所有线程组中的线程生效
父节点是测试计划，并且线程共享模式是“所有线程”时，对所有线程组中的线程生效
- 对当前线程组中的线程生效
父节点是某个线程组时，只会对当前线程组生效

用户定义的变量作用域针对的是测试计划

无论用户定义的变量组件放在哪里，他都会针对整个测试计划生效

程序员-软件测试

JMeter断言

目标

1. JMeter断言

1.1 断言的概念

断言：让程序判断预期结果和实际结果是否一致。

提示：JMeter断言是在请求的返回层面增加一层判断机制；因为请求成功了，并不代表结果一定正确，因此需要检测机制提高测试准确性。

1.2 JMeter中常用断言

- 响应断言
- JSON断言
- 持续时间断言(Duration Assertion)

2. 响应断言

添加方式：测试计划 --> 线程组--> HTTP请求 --> (右键添加) 断言 --> 响应断言

2.1 案例

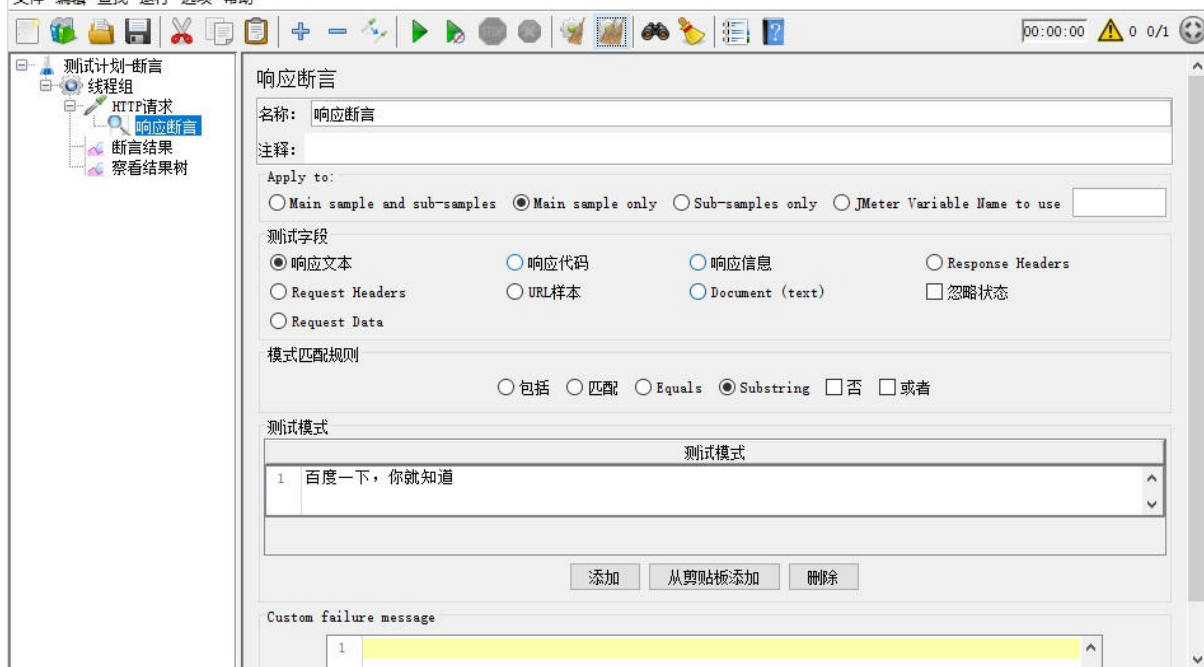
场景

- 请求：<https://www.baidu.com>
- 检查：让程序检查响应数据中是否包含“百度一下，你就知道”

操作步骤

1. 添加线程组
2. 添加HTTP请求
3. 添加响应断言
4. 添加断言结果
5. 添加查看结果树

响应断言



2.2 参数详解

Apply to: 适用范围

- Main sample and sub-samples: 作用于父节点取样器及对应子节点取样器;
- Main sample only: 仅作用于父节点取样器;
- Sub-samples only: 仅作用于子节点取样器;
- JMeter Variable: 作用于jmeter变量(输入框内可输入jmeter的变量名称);

测试字段: 要检查的项

- 响应文本: 来自服务器的响应文本, 即主体, 不包括任何HTTP头
- 响应代码: 响应的状态码, 例如: 200
- 响应信息: 响应的信息, 例如: OK
- Response Headers: 响应头部
- Request Headers: 请求头部
- Request Data: 请求数据
- URL样本: 响应的URL
- Document(text): 响应的整个文档
- 忽略状态: 忽略返回的响应状态码

模式匹配规则

- 包括: 文本包含指定的正则表达式
- 匹配: 整个文本匹配指定的正则表达式
- Equals: 整个返回结果的文本等于指定的字符串(区分大小写)
- Substring: 返回结果的文本包含指定字符串(区分大小写)
- 否 : 取 反
- 或 者 : 如 果存在多个测试模式, 勾选代表逻辑或 (只要有一个模式匹配, 则断言就是OK), 不勾选代表逻辑与 (所有都必须匹配, 断言才是OK)

注意: Equals和Substring模式是普通字符串, 而不是正则表达式

程序员-软件测试

测试模式

即填写你指定的结果 (可填写多个), 按钮【添加】、【删除】是进行指定内容的管理

3. JSON断言

该组件用来对JSON文档进行验证，验证步骤如下：

1. 首先解析JSON数据，如果数据不是JSON，则验证失败。
2. 使用Jayway JsonPath 1.2.0中的语法搜索指定的路径。如果找不到路径，就会失败。
3. 如果在文档中找到JSON路径，并且要求对期望值进行验证，那么它将执行验证操作。

添加方式：测试计划 --> 线程组--> HTTP请求 --> (右键添加) 断言 --> JSON断言

3.1 案例

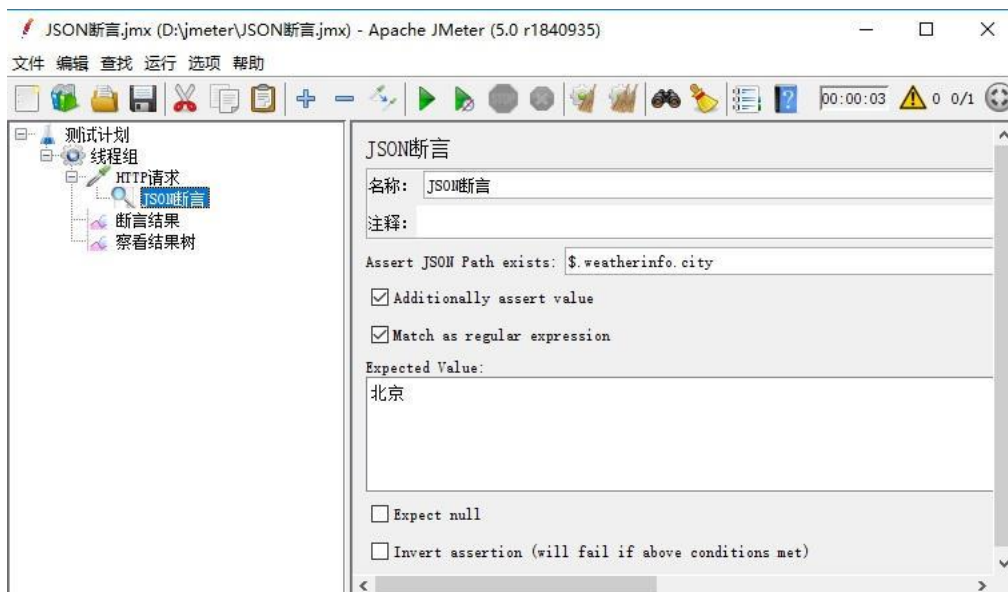
场景

- 请求：<http://www.weather.com.cn/data/sk/101010100.html>
- 检查：让程序检查响应的JSON数据中，city对应的内容是否为“北京”

操作步骤

1. 添加线程组
2. 添加HTTP请求
3. 添加JSON断言
4. 添加断言结果
5. 添加查看结果树

JSON断言



3.2 参数详解

- Assert JSON Path exists: 用于断言的JSON元素的路径
- Additionally assert value: 如果您想要用某个值生成断言，请选择复选框
- Match as regular expression: 如果需要使用正则表达式，请选择复选框

程序员-软件测试

- Expected Value: 期望值，用于断言的值或用于匹配的正则表达式的值
- Expect null: 如果希望为空，请选择复选框
- Invert assertion (will fail if above conditions met): 反转断言(如果满足以上条件则失败)

3.4 JsonPath语法介绍

参考文档：<https://github.com/json-path/JsonPath>

4. 断言持续时间

添加方式：测试计划 --> 线程组--> HTTP请求 --> (右键添加) 断言 --> 断言持续时间

4.1 案例

场景

- 请求：<https://www.jd.com>
- 检查：让程序检查响应时间是否大于**500**毫秒

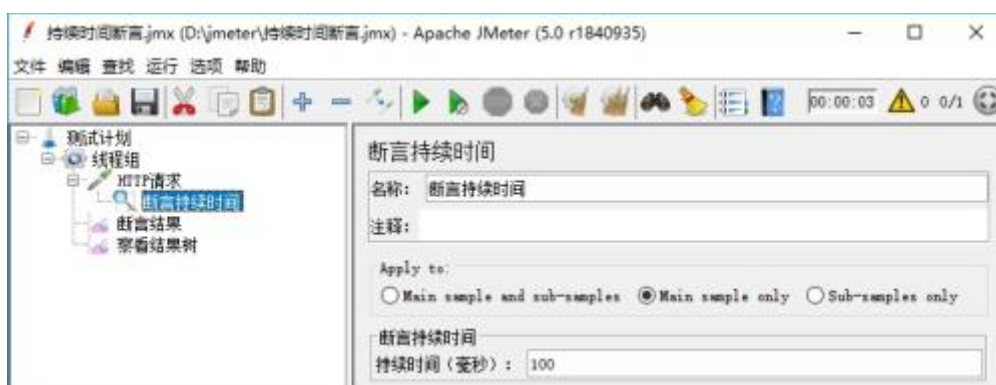
操作步骤

1. 添加线程组
2. 添加HTTP请求
3. 添加断言持续时间
4. 添加断言结果
5. 添加查看结果树

断言持续时间

4.2 参数详解

- 持续时间（毫秒）：在将每个响应标记为失败之前允许的最大毫秒数



程序员-软件测试

JMeter关联

目标

1. 掌握关联的用法

4. 掌握

5. 掌握

1. 关联

当请求之间有依赖关系，比如一个请求的入参是另一个请求返回的数据，这时候就需要用到关联处理。 JMeter可以通过“后置处理器”中的一些组件来处理关联。

常用的关联方法：

- 正则表达式提取器
- XPath提取器
- JSON提取器

2. 正则表达式提取器

添加方式：测试计划 --> 线程组--> HTTP请求 --> (右键添加) 后置处理器 --> 正则表达式提取器

2.1 场景

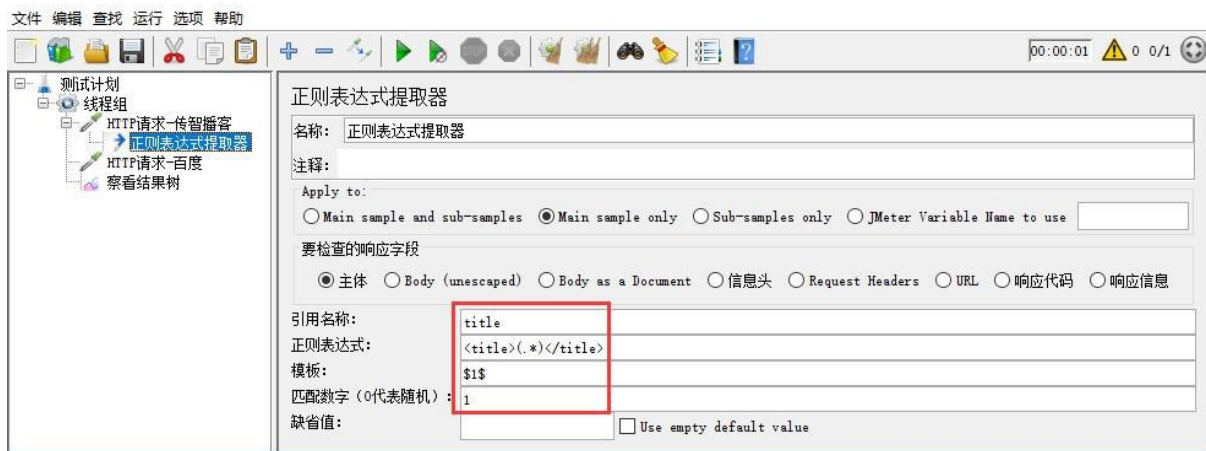
- 请求：<http://www.itcast.cn/>，获取网页的title值
- 请求：<https://www.baidu.com/>，把获取到的title作为请求参数

2.2 操作步骤

- 1.添加线程组
- 2.添加HTTP请求-传智播客
- 3.添加正则表达式提取器
- 4.添加HTTP请求-百度
- 5.添加查看结果树

正则表达式提取器

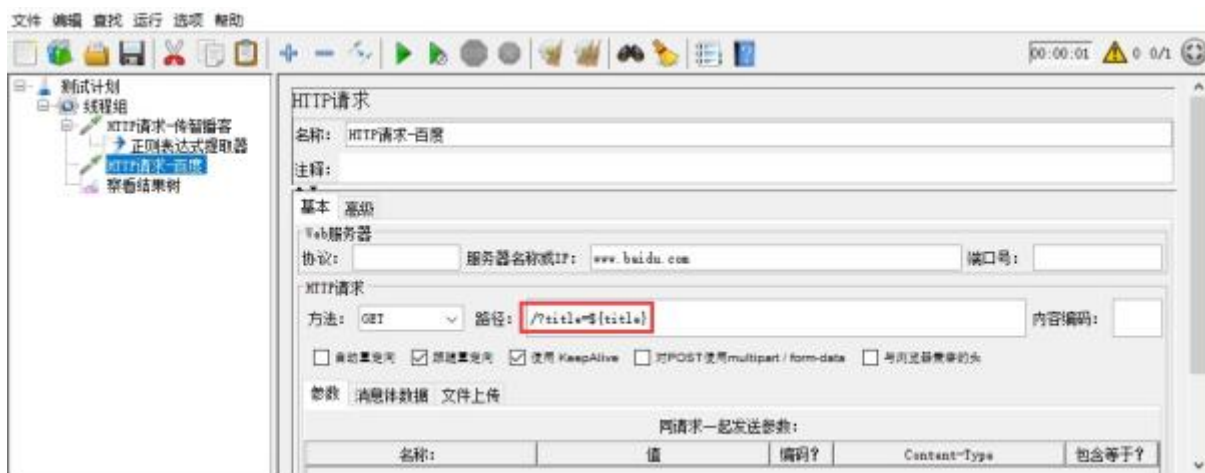
程序员-软件测试



HTTP请求设置(百度)

2.3 参数设置（正则表达式提取器）

- 引用名称：下一个请求要引用的参数名称，如填写title，则可用\${title}引用它
- 正则表达式
 - ()：括起来的部分就是要提取的。
 - .：匹配任何字符串。
 - +：一次或多次。
 - ?：不要太贪婪，在找到第一个匹配项后停止。
- 模板：用\$\$引用起来，如果在正则表达式中有多个正则表达式，则可以是\$2\$\$3\$等等，表示解析到的第几个值给title。如：\$1\$表示解析到的第1个值
- 匹配数字：0代表随机取值，-1代表全部取值，1代表取第一个值
- 缺省值：如果参数没有取得到值，那默认给一个值让它取。



3. XPath提取器

添加方式：测试计划 --> 线程组--> HTTP请求 --> (右键添加) 后置处理器 --> XPath提取器

3.1 场景

- 请求：<http://www.itcast.cn/>，获取网页的title值
- 请求：<https://www.baidu.com/>，把获取到的title作为请求参数

40

程序员-软件测试

3.2 操作步骤

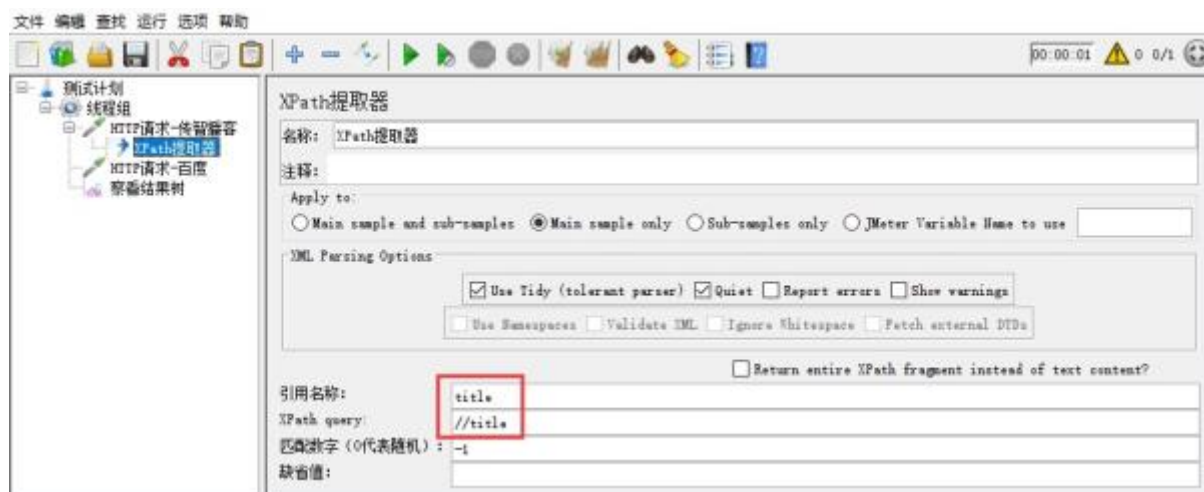
1. 添加线程组
2. 添加HTTP请求-传智播客
3. 添加XPath提取器
4. 添加HTTP请求-百度
5. 添加查看结果树

正则表达式提取器

3.3 参数设置（XPath提取器）

- Use Tidy (tolerant parser): 如果勾选此项，则使用Tidy将HTML响应解析为XHTML。当需要处理的页面是HTML格式时，必须选中该选项，当需要处理的页面是XML或XHTML格式（例如，RSS返回）时，取消选中该选项。
- 引用名称：存放提取出的值的参数
- XPath Query：用于提取值的XPath表达式

- 匹配数字：如果XPath路径查询导致许多结果，则可以选择提取哪个作为变量
 - 0：表示随机
 - -1：表示提取所有结果(默认值)，它们将被命名为<变量名>_N(其中N从1到结果的个数)
 - X：表示提取第X个结果。如果这个x大于匹配项的数量，则不返回任何内容。将使用默认值
- 缺省值：参数的默认值



4. JSON提取器

添加方式：测试计划 --> 线程组--> HTTP请求 --> (右键添加) 后置处理器 --> JSON提取器

4.1 场景

1. 请求获取天气的接口，<http://www.weather.com.cn/data/sk/101010100.html>
2. 获取返回结果中的城市名称
3. 请求：<https://www.baidu.com/s?wd=北京>，把获取到的城市名称作为请求参数

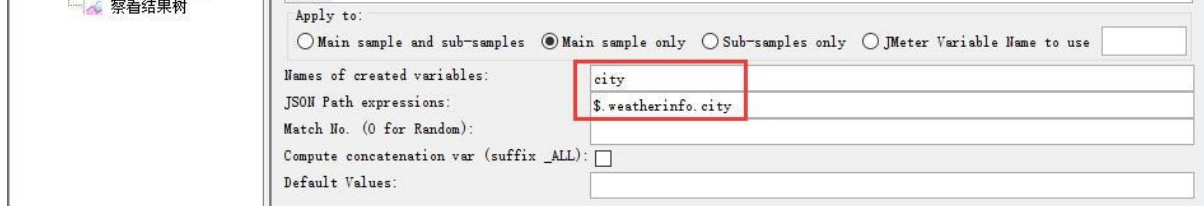
4.2 操作步骤

1. 添加线程组
2. 添加HTTP请求-天气
3. 添加JSON提取器

程序员-软件测试

4. 添加HTTP请求-百度
5. 添加查看结果树

JSON提取器



4.3 参数设置

- Names of created variables: 存放提取出的值的参数
- JSON Path Expressions: JSON路径表达式

5. 跨线程组关联

当有依赖关系的两个请求（一个请求的入参是另一个请求返回的数据），放入到不同的线程组中时，就不能使用提取器保存的变量来传递参数值，而是要使用Jmeter属性来传递。

5.1 Jmeter属性的配置方法

函数实现：

1. __setProperty函数：将值保存成jmeter属性
2. __property函数：在其他线程组中使用property函数读取属性

备注：setProperty函数需要通过BeanShell取样器来执行（BeanShell取样器作用：执行函数和java脚本）

5.2 场景

1. 线程组1：请求获取天气的接口，<http://www.weather.com.cn/data/sk/101010100.html>
2. 获取返回结果中的城市名称
3. 线程组2：请求：<https://www.baidu.com/s?wd=北京>，把获取到的城市名称作为请求参数

5.3 操作步骤

1. 添加线程组1
2. 添加HTTP请求-天气
3. 添加JSON提取器
4. 添加BeanShell取样器（将JSON提取器提取的值保存为Jmeter属性）
5. 添加HTTP请求-百度（读取Jmeter属性）
6. 添加查看结果树

程序员-软件测试

JMeter录制脚本

目标

1. 掌握JMeter如何直连数据库

1. JMeter录制脚本

应用场景:

在没有接口文档的旧项目当中，快速录制web页面产生的http接口请求，帮助编写接口测试脚本

代理服务器原理介绍

代理服务器的原理主要拦截和转发请求与响应数据。

操作步骤：

1. 开启windows操作系统的浏览器代理



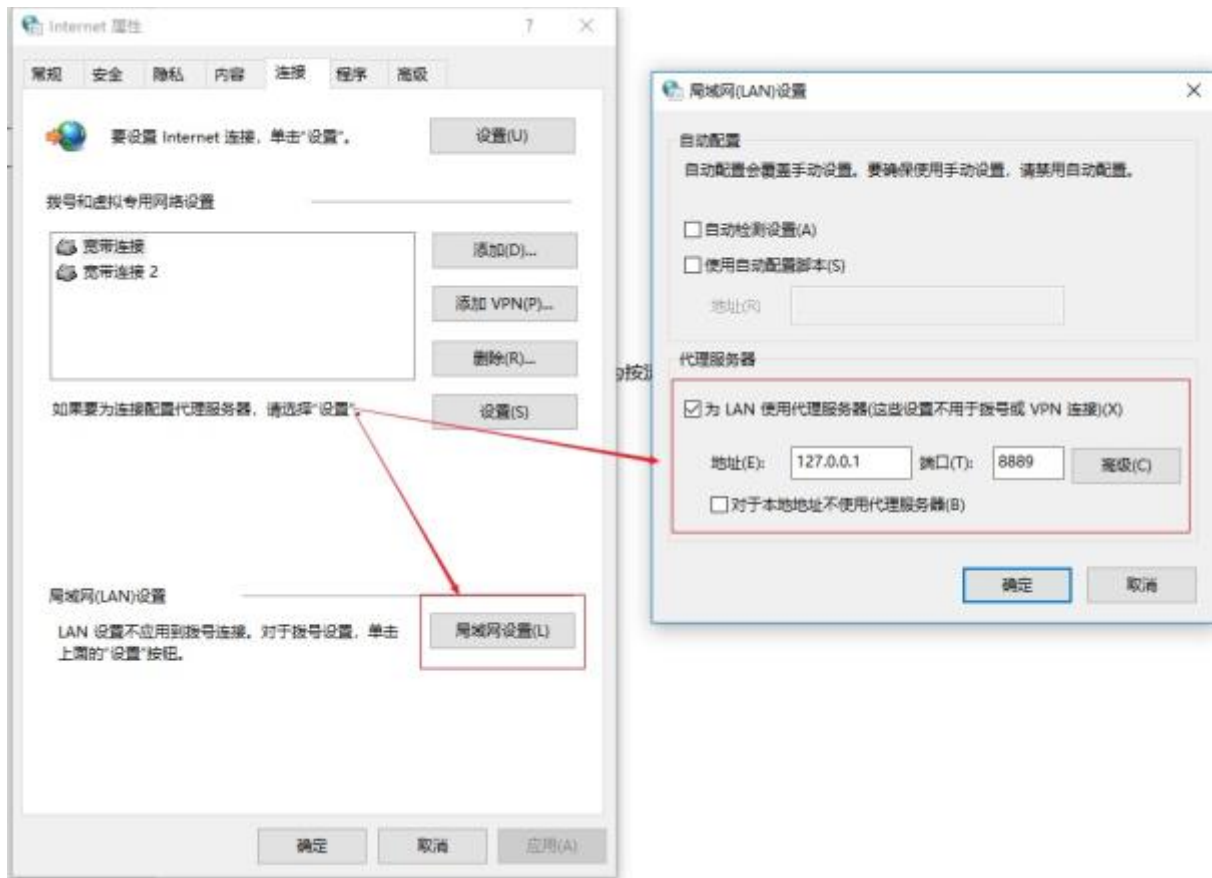
开启后，windows操作系统中所有的http请求都会发送给设置的代理服务器。如果这个代理服务器没有启动，那么会提示网络连接错误。

保存后，访问页面会提示

程序员-软件测试

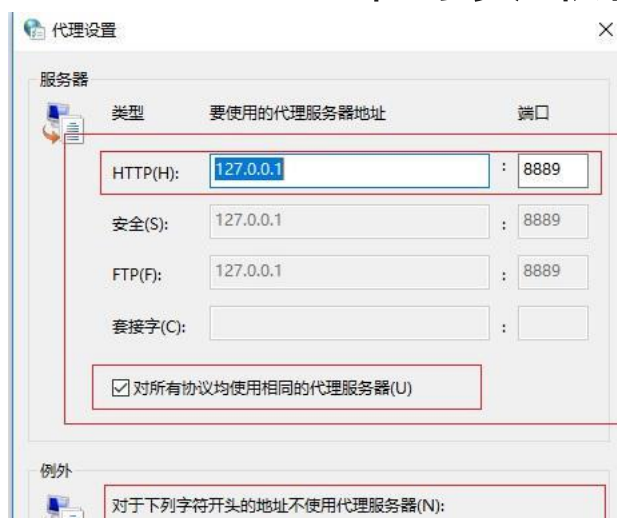


如果是windows7系统，那么在internet选项的连接中配置代理



45

程序员-软件测试



1. 在jmeter当中添加非测试元件HTTP代理服务器，并进行配置

- 在jmeter当中添加HTTP代理服务器：测试计划(右键)->非测试元件->HTTP代理服务器

配置代理服务器的参数

- State:
1. 设置端口：代理服务程序端口程序
 2. 启动按钮
- Test Plan Creation:
1. 目标控制器：录制的脚本放到那个容器
 2. 分组：
 - 1). 不对样本分组：对所有录制的取样器不分组。
 - 2). 在组间添加分组：在取样器分组之间添加以名为 "---" 的控制器。
 - 3). 每个组放入一个新的控制器：每个分组放到一个新的简单控制器下。
 - 4). 只存入每个组的第一个样本：只要每个分组的第一个请求会被录制。
 - 5). Put each group in a new transaction controller:
(每个分组创建一个事务控制器，那个分组的所有取样器都保存在控制器下。)

46

程序员-软件测试

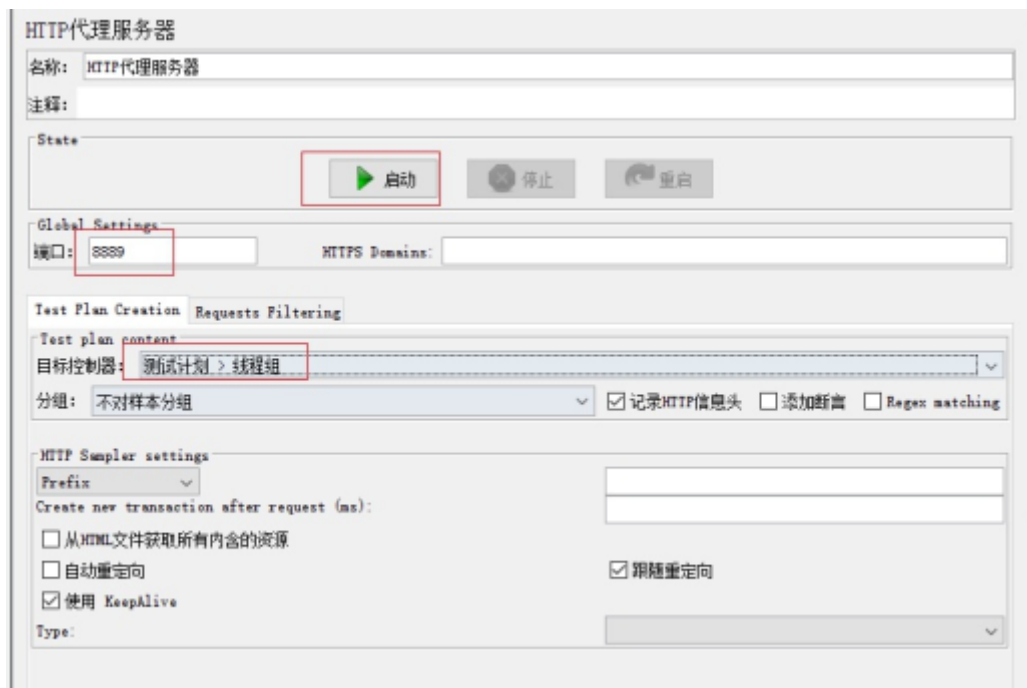
- Requests Filtering:
- 包含模式：url匹配正则表达式，包含此项 如：.*localhost.*
- 排除模式：url匹配正则表达式，不包含此项 如：.*.css *.jpg *.jpeg *.png *.js

注意事项：

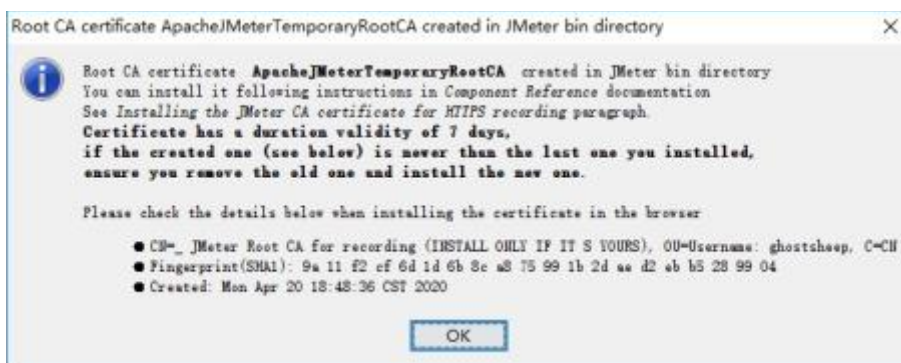
jmeter的代理服务器组件配置中，配置的端口必须和windows的浏览器代理的端口号保持一致

1. 启动代理服务器，开始录制

先点击启动

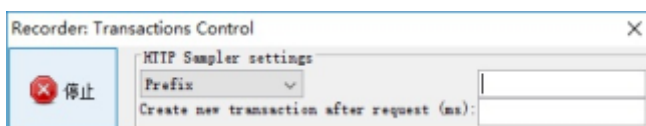


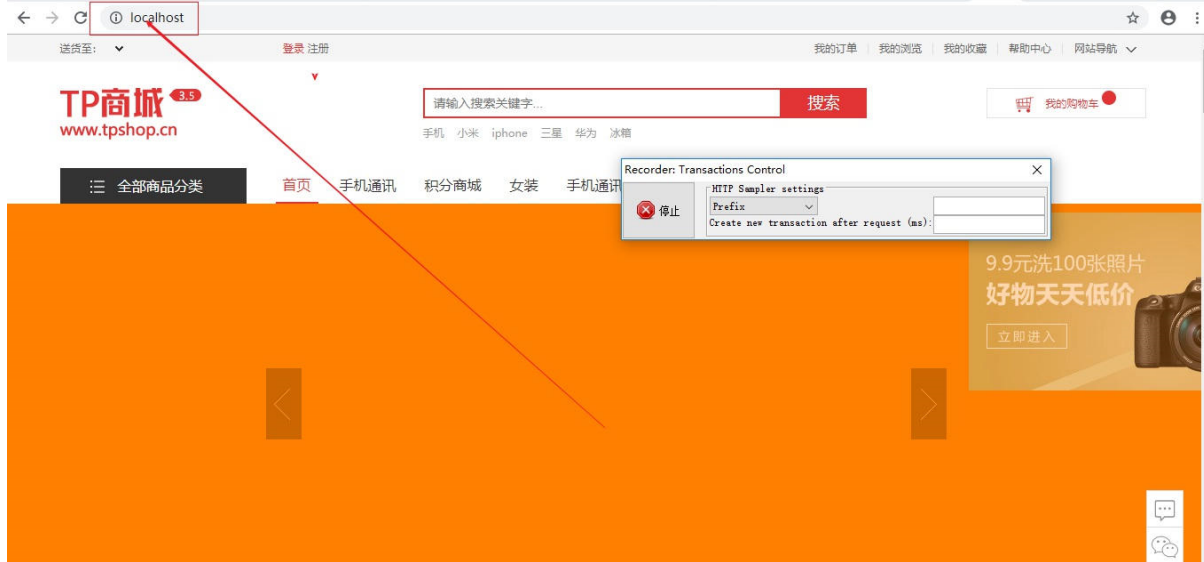
等待一段时间后，会弹出一个证书确认窗口，这个证书用于抓取https接口请求。
后续如果需要抓取https的请求时，需要按照这个窗口的提示添加证书到操作系统当中



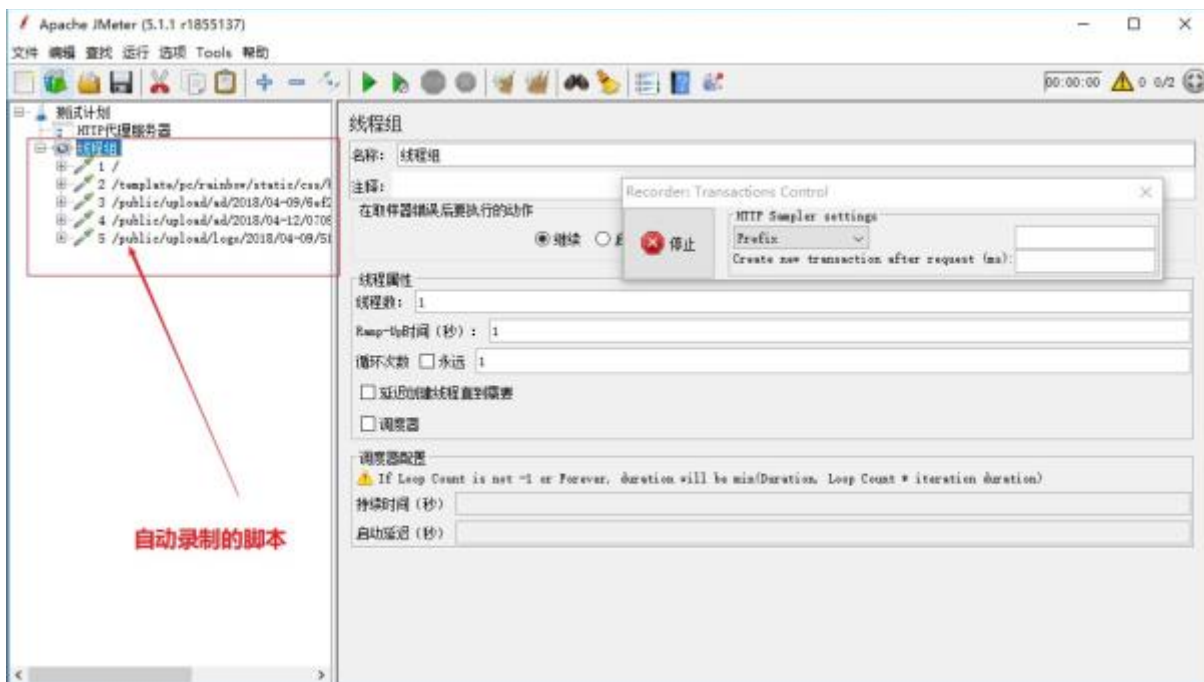
最后的效果：

1. 在浏览器页面中进行操作，成功后，就能在jmeter当中看到抓取到的接口请求了。
访问TPSHOP商城主页





回到jmeter查看录制结果



目标

1. 掌握JMeter如何直连数据库

1. JMeter直连数据库

1.1 场景

1. 连接tpshop商城数据库获取商品名包含：小米手机5 的商品id

1.2 准备工作

启动tpshop数据库服务器

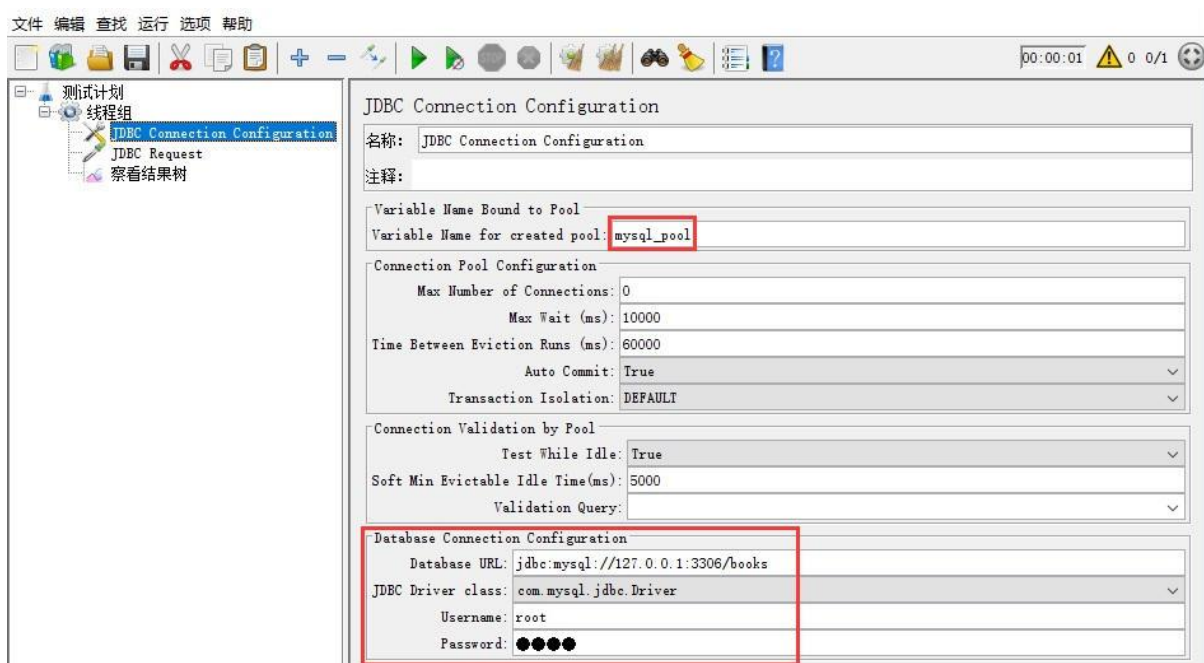
```
库名: tpshop2.0  
表名: tp_goods  
商品id字段: goods_  
id  
商品名字段: goods_name
```

添加MySQL驱动jar包

在测试计划面板点击“浏览...”按钮，将你的JDBC驱动添加进来

配置数据库连接信息

添加方式：测试计划 --> 线程组--> (右键添加) 配置元件 --> JDBC Connection Configuration



程序员-软件测试

主要参数：

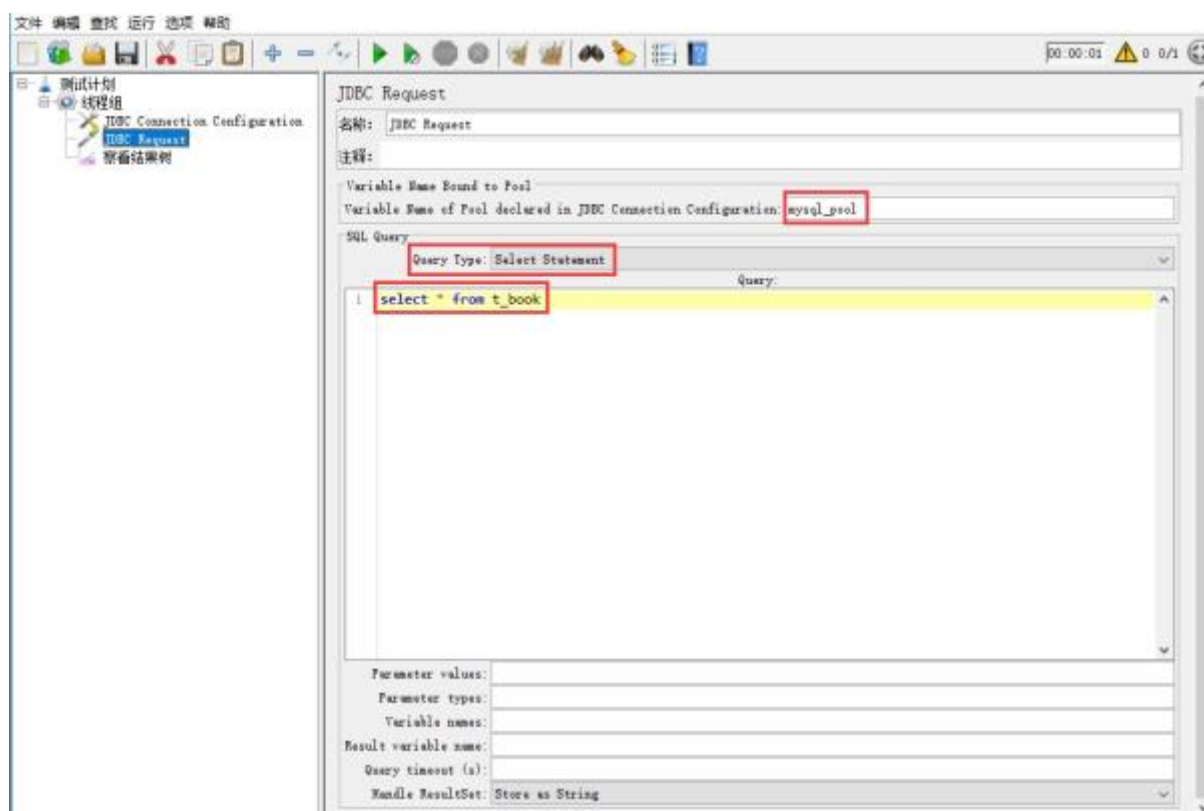
- Variable Name: mysql数据库连接池名称（JDBC请求时要引用）
- Database URL: jdbc:mysql://localhost:3306/tpshop2.0
 - jdbc:mysql: (MySQL固定格式)
 - //127.0.0.1:(数据库ip地址)

- 3306:(MySQL默认端口, 如改变, 请如实填写)
- books: 要连接的数据库名称
- JDBC DRIVER class: com.mysql.jdbc.Driver (MySQL驱动包位置固定格式)
- Username: root(连接数据库用户名, 如实填写)
- Password: (MySQL数据库密码, 如实填写, 如果密码为空不写)

1.3 操作步骤

- 1.添加线程组
- 2.添加 JDBC Connection Configuration
- 3.添加 JDBC request
- 4.添加查看结果树

JDBC request设置

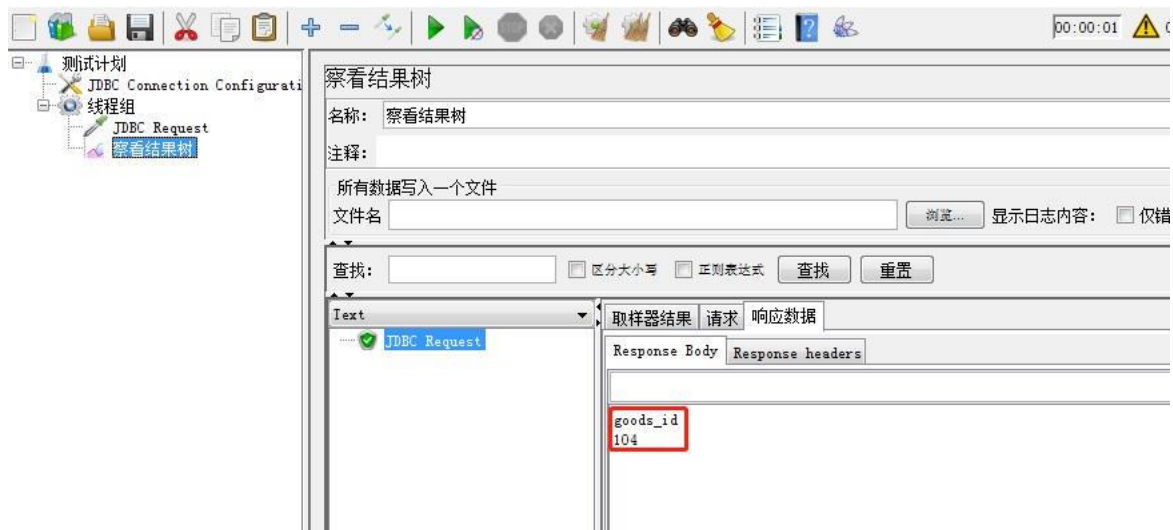


主要参数:

- Variable Name: 数据库连接池的名字, 需要与JDBC Connection Configuration的Variable Name Bound Pool名字保持一致
- Query: 填写的sql语句末尾不要加“;”
- Parameter values: 参数值
- Parameter types: 参数类型
- Variable names: 保存sql语句返回结果的变量名
- Result variable name: 创建一个对象变量, 保存所有返回的结果
- Query timeout: 查询超时时间
- Handle result set: 定义如何处理由callable statements语句返回的结果

程序员-软件测试

运行结果



目标

1. 掌握JMeter逻辑控制器

1. 逻辑控制器

逻辑控制器可以按照设定的逻辑控制取样器的执行顺序

1.1 常用的逻辑控制器

- 如果（If）控制器
- 循环控制器
- ForEach控制器

2. 如果（If）控制器

If控制器用来控制它下面的测试元素是否运行

添加方式：测试计划 --> 线程组--> (右键添加) 逻辑控制器 --> 如果（If）控制器

2.1 案例

需求

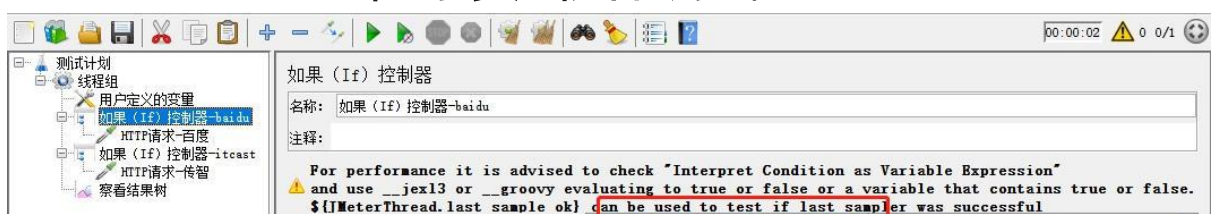
1. 使用‘用户定义的变量’定义一个变量name，name的值可以是‘baidu’或‘itcast’
2. 根据name的变量值实现对应网站的访问

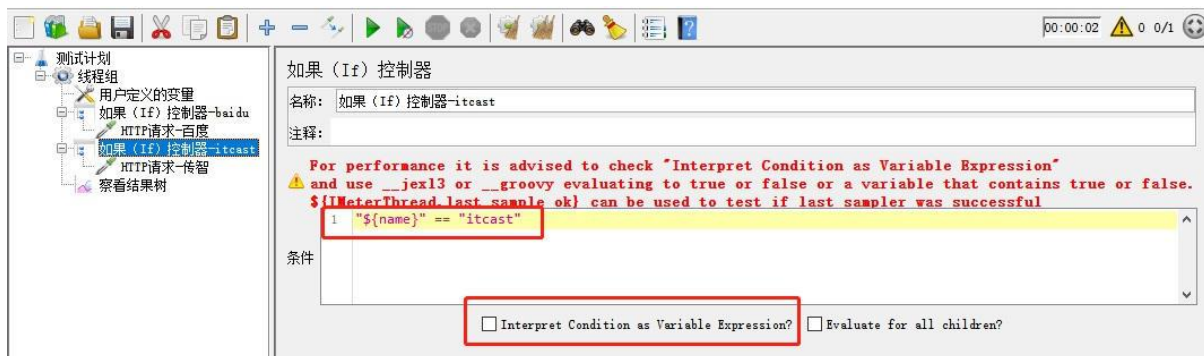
操作步骤

1. 添加线程组
2. 用户定义的变量
3. 添加If控制器，判断name是否等于baidu
4. 添加HTTP请求，用来访问百度
5. 添加If控制器，判断name是否等于itcast
6. 添加HTTP请求，用来访问传智播客
7. 添加查看结果树

If控制器

程序员-软件测试





3. 循环控制器

通过设置循环次数，来实现循环发送请求

添加方式：测试计划 --> 线程组--> (右键添加) 逻辑控制器 --> 循环控制器

3.1 案例

需求

1. 循环访问百度10次

操作步骤

1. 添加线程组
2. 添加循环控制器
3. 添加HTTP请求
4. 添加查看结果树

循环控制器



思考

程序员-软件测试

线程组属性可以控制循环次数，那么循环控制器有什么用？

线程组属性控制组内所有取样器的执行次数，而循环控制器可以控制组内部分取样器的循环次数，后者控制精度更高

4. ForEach控制器

ForEach控制器一般和用户自定义变量或者正则表达式提取器一起使用，其在用户自定义变量或者从正则表达式提取器的返回结果中读取一系列相关的变量。该控制器下的取样器都会被执行一次或多次，每次读取不同的变量值。

添加方式：测试计划 --> 线程组--> (右键添加) 逻辑控制器 --> ForEach控制器

4.1 案例

1. 有一组关键字 [hello,python,测试]，使用用户定义的变量存储
2. 要依次取出关键字，并在百度搜索，例如：<https://www.baidu.com/s?wd=hello>

1. 添加线程组
2. 用户定义的变量
3. 添加ForEach控制器

4. 添加HTTP请求

5. 添加查看结果树

用户定义的变量

需求

操作步骤



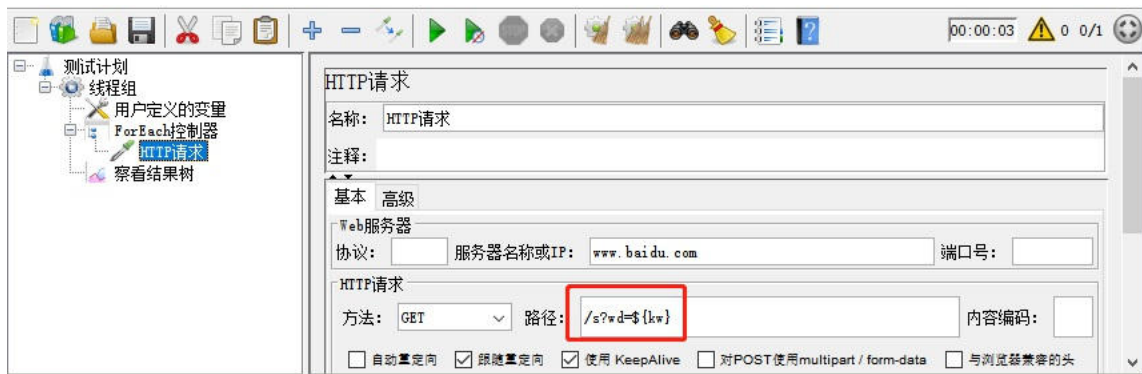
ForEach控制器



HTTP请求

54

程序员-软件测试



4.2 案例

需求

1. 访问传智播客首页<http://www.itcast.cn>，获取首页中的地址信息，并全部保存下来
2. 要依次取出关键字，并在百度搜索，例如：<https://www.baidu.com/s?wd=地址1>

操作步骤

1. 添加线程组
2. 添加HTTP请求1（传智播客）
3. 添加正则表达式提取器
4. 添加ForEach控制器
5. 添加HTTP请求2（百度）
6. 添加查看结果树

程序员-软件测试

JMeter定时器

目标

1. 掌握如何使用同步定时器
2. 掌握如何使用常数吞吐量定时器

1. 同步定时器（Synchronizing Timer）[集合点]

提示：在JMeter中叫做同步定时器，在其他软件中又叫集合点。

2. 如何测试电商网站中的抢购活动、秒杀活动？
思考？
 1. 如何模拟多个用户同时抢一个红包？

1.1 介绍

SyncTimer的目的是阻塞线程，直到阻塞了n个线程，然后立即释放它们。

同步定时器相当于一个储蓄池，累积一定的请求，当在规定的时间内达到一定的线程数量，这些线程会在同一个时间点一起并发，所以可以用来做大数据量的并发请求。

添加方式：测试计划 --> 线程组--> HTTP请求 --> (右键添加) 定时器 --> Synchronizing Timer

1.2 案例

场景

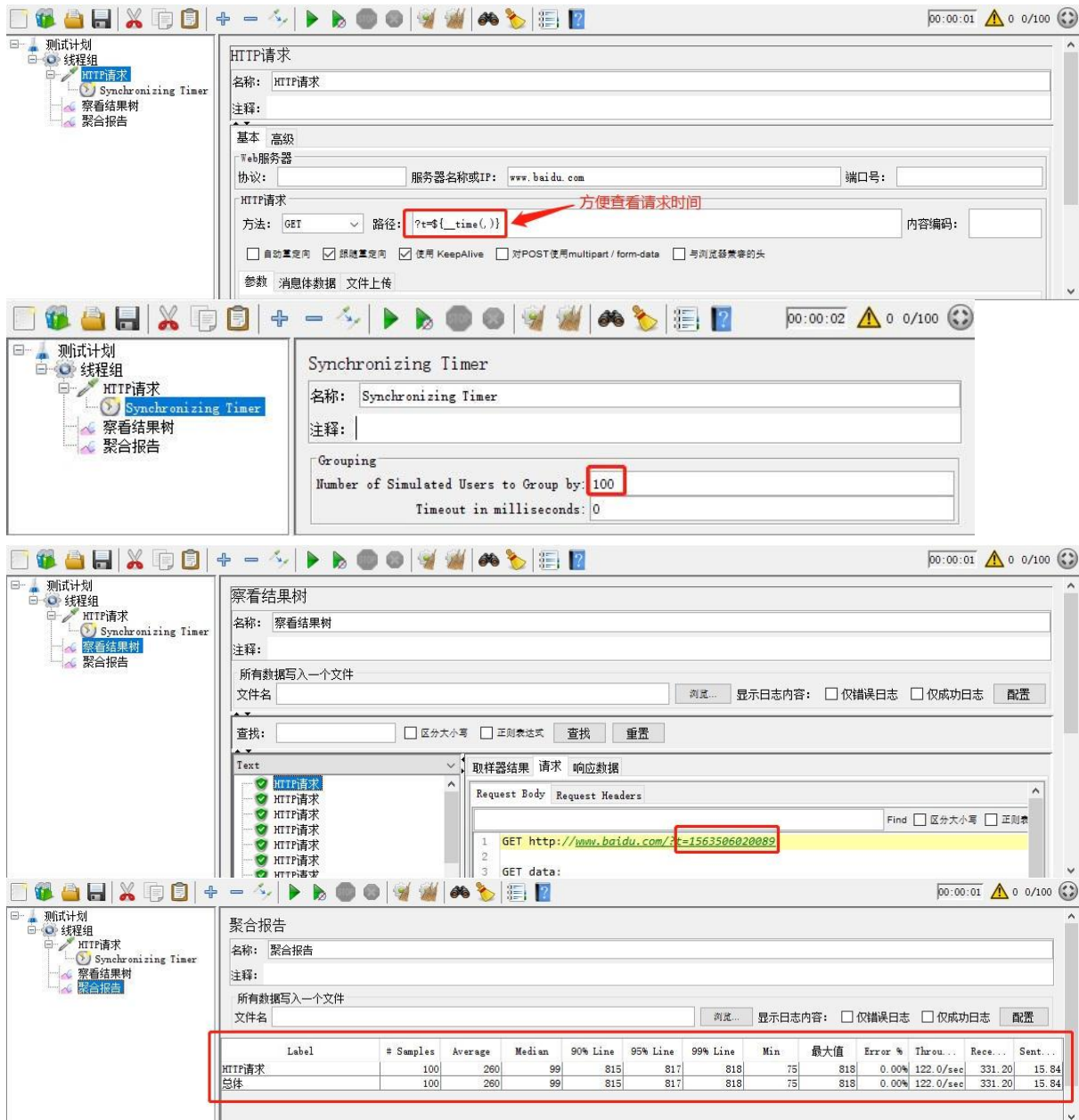
模拟100个用户同时访问百度首页，统计高并发情况下运行情况

操作步骤

1. 添加线程组，设置线程数=100
2. 添加HTTP请求
3. 添加同步定时器
4. 添加查看结果树
5. 添加监听器-聚合报告

操作截图

程序员-软件测试



1.3 注意事项

问题: 当用户数不能整除集合点组件的一组用户数属性时, 如果超时时间是 0, 会导致程序挂起, 怎么避免挂起?

实现:

- 方案1: 点击 stop 强行终止, 但是不建议
- 方案2: 修改一组用户数, 能够做到整除(治标不治本)
- 方案3: 修改超时时间, 不设置为 0, 即便一组用户数填充不满, 只要超时, 也会执行(建议)

2. 常数吞吐定时器 (Constant Throughput Timer)

2.1 介绍

常数吞吐量定时器可以让JMeter以指定数字的吞吐量（以每分钟的样本数为单位，而不是每秒）执行。吞吐量计算的范围可以为指定为当前线程、当前线程组、所有线程组。

57

程序员-软件测试

添加方式：测试计划 --> 线程组--> HTTP请求 --> (右键添加) 定时器 --> Constant Throughput Timer

2.2 案例

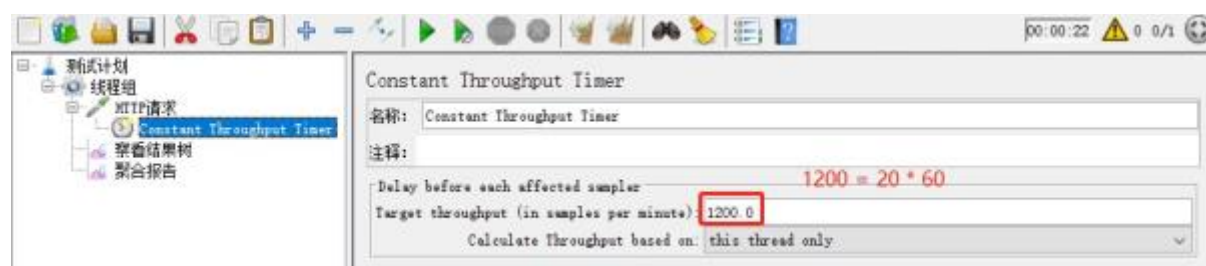
场景

一个用户以 **20QPS (20 次/s)** 的频率访问百度首页，持续一段时间，统计运行情况

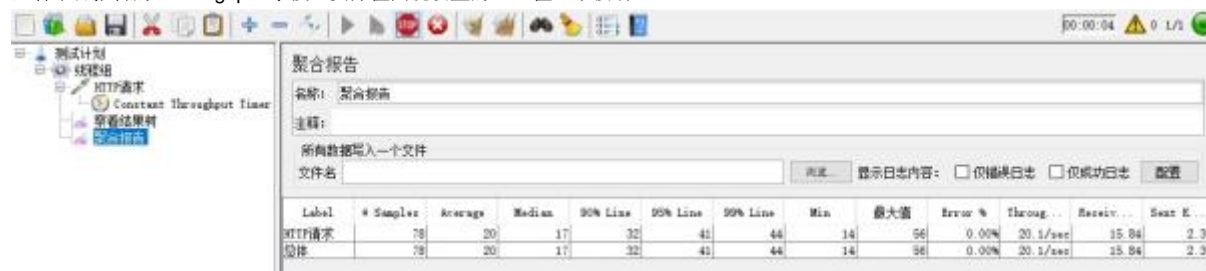
操作步骤

1. 添加线程组，循环次数设置成永远
2. 添加HTTP请求
3. 添加常数吞吐定时器
4. 添加查看结果树
5. 添加监听器-聚合报告

操作截图



查看聚合报告的 Throughput 字段，实际值围绕设置的QPS值上下波动



程序员-软件测试

JMeter分布式

目标

1. 了解JMeter分布式的应用场景
2. 掌握JMeter分布式操作技巧

1. JMeter分布式测试

在使用JMeter进行性能测试时，如果并发数比较大(比如项目需要支持10000并发)，单台电脑的(CPU和内存)可能无法支持，这时可以使用JMeter提供的分布式测试的功能。

1.1 JMeter分布式执行原理

- JMeter分布式测试时，选择其中一台作为控制机(Controller)，其它机器做为代理机(Agent)。
- 执行时，控制机会把脚本发送到每台代理机上，代理机拿到脚本后就开始执行，代理机执行时不需要启动JMeter界面，可以理解它是通过命令行模式执行的。
- 执行完成后，代理机会把结果回传给控制机，控制机会收集所有代理机的信息并汇总。

1.2 代理机（Agent）配置

1. Agent机上需要安装JMeter
2. 修改服务端口
 - 注意：非必须。如果是在同一台机器上演示需要使用不同的端口，多台机器可以不修改
 - 打开bin/jmeter.properties文件，修改`server_port`，比如：`server_port=2001`
3. 将RMI SSL设置为禁用
 - 打开bin/jmeter.properties文件，修改为：server.rmi.ssl.disable=true
4. 运行Agent上的jmeter-server.bat文件，启动JMeter

1.3 控制机（Controller）配置

1. 修改JMeter的bin目录下jmeter.properties配置文件，修改`remote_hosts`
 - 示例：`remote_hosts=192.168.182.100:1099,192.168.182.101:1099`
 - IP和Port是Agent机的IP以及自定义的端口，多台Agent之间用","隔开
2. 将RMI SSL设置为禁用
 - 打开bin/jmeter.properties文件，修改为：server.rmi.ssl.disable=true
3. 启动JMeter
4. 选择菜单：运行-->远程启动/远程全部启动

1.4 案例

一台控制机和两台执行机，做分布式；要求控制机启动，两台执行机执行，反馈结果；

实现步骤

1. 配置代理机一，并启动
2. 配置代理机二，并启动
3. 配置控制机，并启动
4. 添加线程组
5. 添加HTTP请求
6. 添加聚合报告

程序员-软件测试

1.5 备注

1. 修改完端口要重启JMeter
2. 控制机和代理机最好分开，由于控制机需要发送信息给代理机并且会接受代理机回传的测试数据，所以控制机自身会有消耗
3. 参数文件：如果使用csv进行参数化，那么需要把参数文件在每台slave上拷一份且路径需要设置成一样的；
4. 每台机器上安装的JMeter版本和插件最好都一致，否则会出一些意外的问题；

程序员-软件测试

JMeter测试报告

目标

1. 掌握聚合报告各项指标含义
2. 掌握JMeter如何生成测试报告

1. 聚合报告

位置： 测试计划->右键->监听器->聚合报告

10. 异
11. 吞
12. 接
13. 发

注释:													
所有数据写入一个文件													
文件名										显示日志内容: <input type="checkbox"/> 仅捕获日志 <input type="checkbox"/> 仅成功日志 <input type="checkbox"/> 配置			
Label	# 样本	平均值	中位数	90% 百分位	95% 百分位	99% 百分位	最小值	最大值	异常 %	吞吐量	接收 KB/sec	发送 KB/sec	
学院-List...	1	11	11	11	11	11	11	11	0.00%	90.9/sec	43.41	18.92	
学院-List...	1	11	11	11	11	11	11	11	0.00%	90.9/sec	43.41	18.92	
学院-成绩...	1	15	15	15	15	15	15	15	0.00%	66.7/sec	24.67	13.15	
学院-组合...	1	15	15	15	15	15	15	15	0.00%	66.7/sec	24.67	13.15	
学院-删除...	2	495	428	563	563	563	428	563	0.00%	2.0/sec	0.37	0.42	
班级-新建...	1	1053	1053	1053	1053	1053	1053	1053	0.00%	57.0/min	0.52	0.53	
班级-更新...	1	646	646	646	646	646	646	646	0.00%	1.5/sec	0.52	0.49	
班级查询-所有...	1	27	27	27	27	27	27	27	0.00%	37.0/sec	19.06	7.20	
班级查询-指定...	1	10	10	10	10	10	10	10	0.00%	100.0/sec	33.89	20.51	
班级查询-学...	1	13	13	13	13	13	13	13	0.00%	76.9/sec	39.59	17.58	
班级查询-学...	1	14	14	14	14	14	14	14	0.00%	71.4/sec	36.76	16.88	
班级查询-学...	1	12	12	12	12	12	12	12	0.00%	83.3/sec	42.89	20.10	
班级查询-学...	1	22	22	22	22	22	22	22	0.00%	45.5/sec	23.39	10.74	
班级查询-组合...	1	18	18	18	18	18	18	18	0.00%	95.6/sec	21.59	14.76	
班级-删除...	2	495	495	499	499	499	495	499	0.00%	2.0/sec	0.37	0.46	
学生-成绩...	1	472	472	472	472	472	472	472	0.00%	2.1/sec	0.90	0.85	
学生-新建...	1	771	771	771	771	771	771	771	0.00%	1.3/sec	0.93	1.23	
student-学...	5	0	0	1	1	1	0	1	0.00%	6.4/sec	0.16	0.00	
学生-更新...	1	648	648	648	648	648	648	648	0.00%	1.5/sec	0.85	0.96	
学生查询-所有...	1	34	34	34	34	34	34	34	0.00%	29.4/sec	20.19	6.29	
学生查询-指定...	1	21	21	21	21	21	21	21	0.00%	47.6/sec	24.51	21.34	
学生查询-学...	1	26	26	26	26	26	26	26	0.00%	35.7/sec	24.52	8.96	
学生查询-学...	1	18	18	18	18	18	18	18	0.00%	55.6/sec	38.14	14.21	
学生查询-学...	1	17	17	17	17	17	17	17	0.00%	58.8/sec	27.92	13.96	
学生-删除...	2	496	375	538	538	538	375	538	0.00%	2.2/sec	0.40	0.53	
学生-成绩...	1	183	183	183	183	183	183	183	0.00%	5.5/sec	1.01	1.24	
student-学...	1	527	527	527	527	527	527	527	0.00%	1.9/sec	0.35	0.79	
总计	73	129	6	527	646	776	0	1053	0.00%	7.6/sec	1.73	1.14	

1. Label: 每个请求的名称(勾选: 在标签中包含组名称, 显示线程组名-取样器名)
2. #样本: 各请求发出的数量
3. 平均值: 平均响应时间(单位: 毫秒)。默认是单个Request的平均响应时间
4. 中位数: 中位数, 50% <= 时间
5. 90%百分比: 90% <= 时间

2. 生成html测试报告

JMeter支持生成HTML测试报告, 以便从测试计划中获得图表和统计信息。

2.1 命令

61

程序员-软件测试

```
jmeter -n -t [jmx file] -l [result file] -e -o [html report folder]
eg: jmeter -n -t hello.jmx -l result.jtl -e -o ./report
```

参数描述:

- -n: 非GUI模式执行JMeter
- -t [jmx file]: 测试计划保存的路径及jmx文件名, 路径可以是相对路径也可以是绝对路径
- -l [result file]: 保存生成测试结果的文件, jtl文件格式
- -e: 测试结束后, 生成测试报告
- -o [html report folder]: 存放生成测试报告的路径, 路径可以是相对路径也可以是绝对路径

注意: result.jtl和report会自动生成, 如果在执行命令时result.jtl和report已存在, 必须用先删除, 否则在运行命令时就会报错

2.2 查看测试报告

打开index.html，就可以看到页面左侧有三个菜单：



2.3 Dashboard

Test and Report informations

- Source file ---- 生成报告的源文件
- Start Time ---- 开始时间
- End Time ---- 结束时间

Test and Report informations	
Source file	"a.jtl"
Start Time	"6/24/19 10:40 AM"
End Time	"6/24/19 10:40 AM"
Filter for display	""

APDEX (应用性能指标)

APDEX (Application Performance Index)

Apdex	T (Toleration threshold)	F (Frustration threshold)	Label
1.000	500 ms	1 sec 500 ms	Total
1.000	500 ms	1 sec 500 ms	登陆 (发送登陆数据) 并返回登陆成功提示
1.000	500 ms	1 sec 500 ms	登陆成功后返回到首页
1.000	500 ms	1 sec 500 ms	会员名退出成功
1.000	500 ms	1 sec 500 ms	返回到首页
1.000	500 ms	1 sec 500 ms	打开首页
1.000	500 ms	1 sec 500 ms	登陆页面输入用户名与密码

https://blog.csdn.net/qq_24394093

- 计算每笔交易APDEX的容忍和满足阈值基于可配置的值，范围在 0-1 之间，1表示达到所有用户均满意
- T(Toleration threshold): 容忍或满意阈值
- F(Frustration threshold): 失败阈值

Requests Summary (请求总结)

成功与失败请求占比，KO指失败率，OK指成功率



https://blog.csdn.net/qq_24394093

2.4 Chart (图表)

它包括Over Time(时间变化)、Throughput (吞吐量)、Response Times (响应时间)

程序员-软件测试

JMeter性能测试常用图表

1. 掌握TPS算法
1. 了解jmeter下载插件方式
2. 理解常用性能测试图表

1. 常用平均并发数计算公式

需求：



PV: (Page View) 即页面访问量, 每打开一次页面PV计数+1, 刷新页面也是。PV只统计页面访问次数。
UV(Unique Visitor), 唯一访问用户数, 用来衡量真实访问网站的用户数量。
一般用UV统计用户活跃度, 用PV统计用户访问页面的频率

1.1 普通计算方法

计算公式: $TPS = \text{总请求数} / \text{总时间}$

按照需求所示, 在2019年第32周, 有4.13万的浏览量, 那么总请求数, 我们可以认为估算为4.13万 (1次浏览都至少对应1个请求)

总请求数 = 4.13 万请求数 = 41300 请求数

总时间: 由于不知道每个请求的具体时间, 我们按照普通方法, 我们可以按照一周的时间进行计算

总时间 = 1天 = 1 * 24 小时 = 24 * 3600 秒

套入公式可得:

$TPS = 41300 \text{ 请求数} / 24 * 3600 \text{ 秒} = 0.48 \text{ 请求数/秒}$

结论: 按照普通计算方法, 我们在测试环境对相同的系统进行性能测试时, 每秒能够发送0.48请求就可以满足线上的需要。

1.2 二八原则计算方法

二八原则就是指80%的请求在20%的时间内完成

计算公式: $TPS = \text{总请求数} * 80\% / (\text{总时间} * 20\%)$

按照公式进行计算

$TPS = 41300 * 0.8 \text{ 请求数} / 24 * 3600 * 0.2 \text{ 秒} = 1.91 \text{ 请求数/秒}$

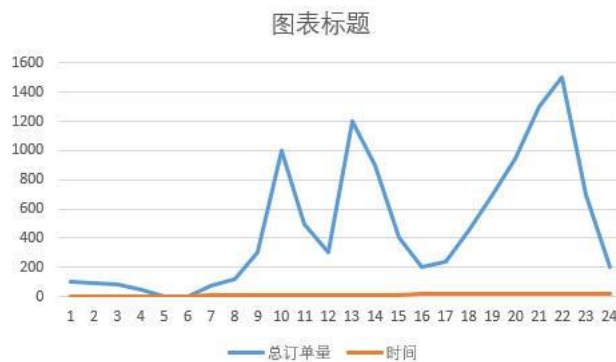
结论: 按照二八原则计算, 在测试环境我们的TPS只要能达到1.91请求数每秒就能满足线上需要。二八原则的估算结果会比平均值的计算方法更能满足用户需求。

1.3 按照业务数据进行计算

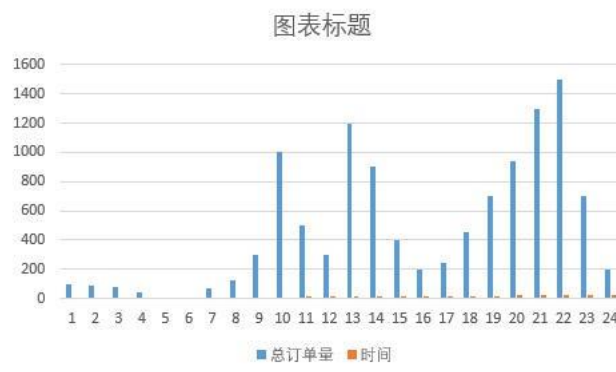
程序员-软件测试

业务数据: 有的公司会统计一定时间内的所有业务数据, 我们可以根据这个业务数据曲线图, 统计出最多请求的数量和时间比例。

曲线图看趋势



直方图看数据和趋势



计算模拟用户正常业务操作（稳定性测试）的并发量：

根据这些数据统计图，可以得出结论：

- 大部分订单在8点-24点之间，因此系统的有效工作时长为16个小时
- 从订单数量统计，8-24点之间的订单占一天总订单的98%左右（40474个）

结合二八原则计算公式： $TPS = \text{总请求数} \times 80\% / (\text{总时间} \times 20\%)$

需要在测试环境模拟用户正常业务操作（稳定性测试）的并发量为：
 $TPS = 40474 \times 0.8 \text{ 请求数} / 16 \times 3600 \times 0.2 \text{ 秒} = 2.81 \text{ 请求数/秒}$

计算模拟用户峰值业务操作（压力测试）的并发量：

根据这些数据统计图，可以得出结论：

- 订单最高峰在在21点-22点之间，一小时的订单总数大约为8853个

计算压力测试的并发数： $TPS = \text{峰值请求数} / \text{峰值时间} \times \text{系数}$

需要在测试环境模拟用户峰值业务操作（压力测试）的并发量为：
 $TPS = 8853 \text{ 请求数} / 3600 \text{ 秒} \times 3 (\text{系数}) = 7.38 \text{ 请求数/秒}$

2. 插件管理包工具

程序员-软件测试

说明：下载jmeter插件管理工具包（可以用此包下载jmeter插件）

2.1 应用步骤

1. 下载包管理工具jar包
2. 将包管理工具jar包添加到jmeter中

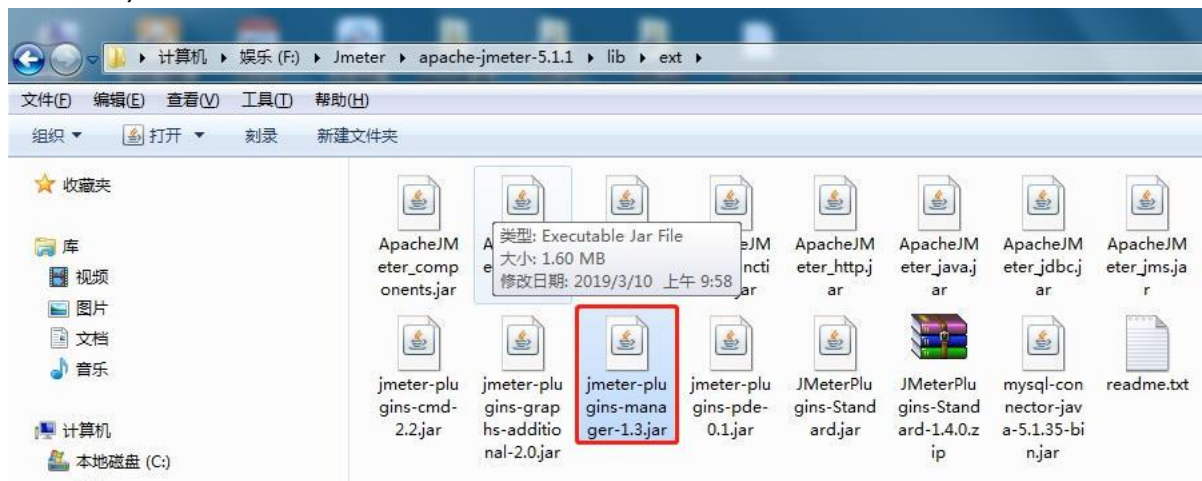
3. 下载

下载: <https://jmeter-plugins.org/install/Install/>



将jar包添加到jmeter中

提示: 存放到jmeter安装目录 lib\ext\目录下



3. 性能测试常用图表及组件

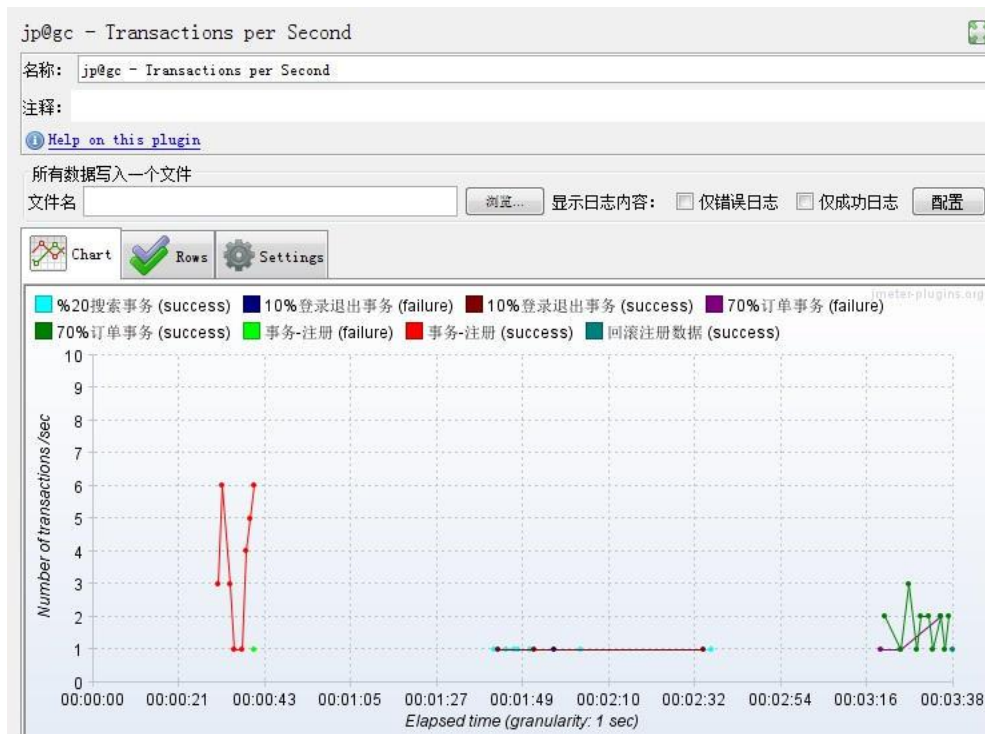
1. Concurrency Thread Group 线程组
2. Transactions per Second 每秒事务数
3. Bytes Throughput Over Time 吞吐量
4. PerfMon Metrics Collector 性能指标收集器

3.1 Concurrency Thread Group 线程组

说明: 阶梯线程组

添加方式: 测试计划 --> 线程(用户) --> Concurrency Thread Group

将线程状态记录到文件中（将线程启动和线程停止事件保存为日志文件）

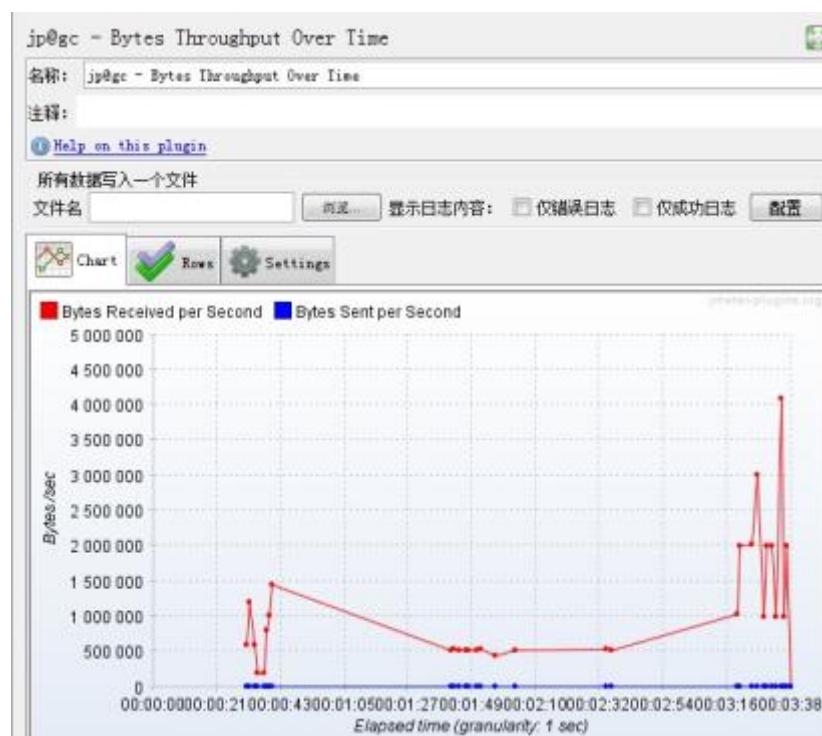


提示: 需要配合事务控制器完成, 如果不使用事务控制器, 默认1个请求为1个事务

3.3 Bytes Throughput Over Time

说明: 查看服务器吞吐流量 单位/字节

添加方式: 测试计划 --> 线程组--> 监听器--> Bytes Throughput Over Time



3.4 PerfMon Metrics Collector

69

程序员-软件测试

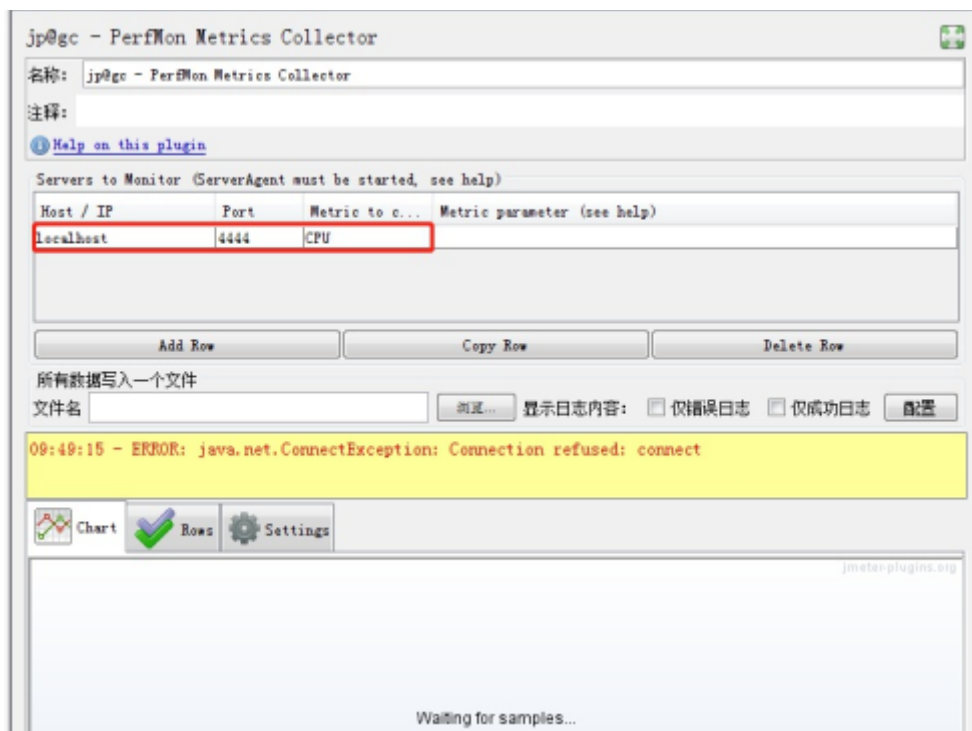
说明: 用来监控服务端性能的工具, 包括cpu、内存、磁盘、网络等性能数据

添加方法: 线程组--> 监听器--> jp@gc - PerfMon Metrics Collector

注意: 使用之前需要在服务器端安装监听服务程序并启动

监控服务器服务程序:

1. 下载安装包ServerAgent-2.2.3.zip, 链接地址: <https://github.com/undera/perfmon-agent>
2. 解压ServerAgent-2.2.3.zip
3. 启动, 如果是windows运行startAgent.bat, 如果是linux运行startAgent.sh
4. 启动这个工具后, jmeter的插件jp@gc - PerfMon Metrics Collector就可以收集服务端的资源使用率, 并在jmeter中查看了



目标

1. 熟悉项目的功能模块和技术架构
2. 掌握如何进行性能测试点的提取
3. 掌握性能测试计划包含的主要内容
4. 掌握如何编写性能测试用例
5. 熟练掌握如何编写JMeter测试脚本
6. 知道如何建立性能测试环境
7. 掌握如何执行测试脚本
8. 掌握性能测试监控关键指标
9. 知道如何进行性能测试瓶颈分析
10. 知道如何进行性能调优
11. 掌握性能测试报告包含的主要内容

目标

1. 熟悉项目的功能模块
2. 熟悉项目的技术架构
3. 了解搭建轻商城项目环境的流程

1. 轻商城项目介绍

1.1 背景

轻商城项目是一个现在流行的电商项目。我们需要综合评估该项目中各个接口的性能，并给出优化建议，以满足公司未来的发展需要。

1.2 简介

- 轻商城是一个支持web和微信小程序的前后端分离架构的项目。
- 前端使用VUE技术框架开发，即支持微信小程序，也支持手机移动端，还支持web页面。
- 后端使用了SpringBoot框架进行开发，MySQL做数据库。
- 目前还在开发完善阶段。

2. 项目功能架构

- 小商城功能
 - 首页
 - 专题列表、专题详情
 - 分类列表、分类详情
 - 品牌列表、品牌详情
 - 新品首发、人气推荐
 - 优惠券列表、优惠券选择
 - 团购
 - 搜索
 - 商品详情、商品评价、商品分享
 - 购物车
 - 下单
 - 订单列表、订单详情、订单售后
 - 地址、收藏、足迹、意见反馈
 - 客服
- 管理平台功能
 - 会员管理
 - 商城管理
 - 商品管理
 - 推广管理
 - 系统管理
 - 配置管理
 - 统计报表

3. 项目技术架构

72

程序员-软件测试

技术栈

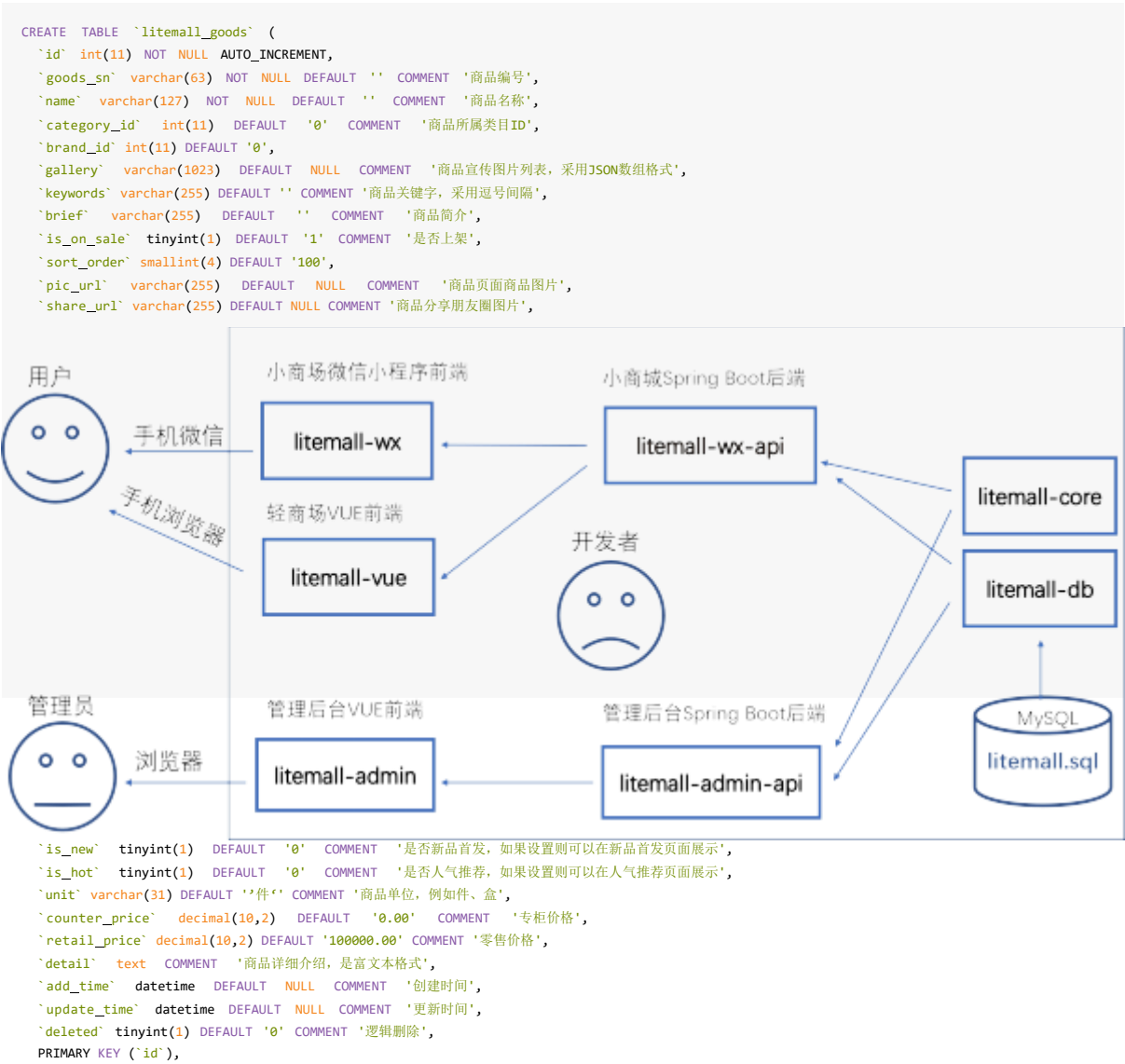
- Spring Boot
- Vue

- 微信小程序

技术架构图

4. 熟悉数据库设计

- 1. 熟悉数据库设计结构，便于后期对数据库的性能监控
- 2. 性能测试的过程中，数据库容易出现性能瓶颈



```
KEY `goods_sn` (`goods_sn`),
KEY `cat_id` (`category_id`),
KEY `brand_id` (`brand_id`),
KEY `sort_order` (`sort_order`)
) ENGINE=InnoDB AUTO_INCREMENT=1181005 DEFAULT CHARSET=utf8mb4 COMMENT='商品基本信息表';
```

73

程序员-软件测试

5. 轻商城项目搭建

5.1 准备工作

- 安装JDK
- 安装MySQL
- 安装Nginx
- 安装node.js

5.2 项目搭建步骤

1. 获取项目源代码
 - 包括前端代码和后端代码
 - 实际工作中项目源代码由开发提供，项目所需要的配置文件、启动项目的顺序也由开发提供文档介绍
2. 构建轻商城后端代码
 - 编译、打包
 - 打包成jar包或war包
3. 构建前端代码
 - 使用node.js打包
 - 部署包中包含HTML、JS、CSS等文件
4. 初始化MySQL数据库
 - 项目启动前需要先初始化数据库
 - 执行初始化数据库的sql文件

```
source /usr/local/litemall/litemall-db/litemall.sql
```

5. 启动轻商城后台管理系统的后端服务

```
java -jar litemall-all.jar
```

6. 部署轻商城前端服务
 - 可以使用Nginx服务器
7. 通过浏览器访问启动的前端，测试项目是否能够正常运行

程序员-软件测试



程序员-软件测试

性能测试需求分析

目标

1. 知道如何获取有效的需求
2. 掌握如何进行性能测试点的提取
3. 能够确定性能测试的目标

1. 性能测试需求分析

- 性能测试需求分析与传统的功能测试需求有所不同
- 功能测试需求分析：重点在于分析被测系统的功能是否满足产品功能需求规格（正向、逆向）
- 性能测试需求分析：重点在于分析被测系统是否能满足特定的业务需求场景（时间、资源）
 - 需要从业务场景、程序代码、服务器、硬件配置等多个维度分析系统可能存在性能瓶颈

1.1 如何获取有效的需求

1. 客户方提出
 - 能够提出明确需求的一般是金融、银行、电信、医疗等企业，他们一般对系统的性能要求高，并且对性能也非常了解
 - 提示：需要评估性能需求的合理性
2. 根据历史数据分析
 - 通过分析历史运营数据收集用户信息，如：
 - 注册用户数、日活、月活，计算用户的增长速度
 - 每月、每周、每天的峰值业务量是多少
 - 用户频繁使用的功能模块是哪些

2. 性能测试点的提取

2.1 性能测试点的提取规则

1. 用户频繁使用的业务功能
2. 非常关键的业务功能
3. 特殊交易日或峰值交易的业务功能
4. 核心业务发生重大调整的业务功能
5. 资源占用非常高的业务功能

2.2 轻商城性能测试点的提取

编号	功能模块	业务功能	功能描述	优先级
T01	登录	登录	用户通过用户名和密码登录	高
T02	首页	进入首页	获取商城首页数据	高
T03	商品	搜索商品	通过关键字搜索商品	高

T04	商品	查看商品详情	点击商品进入商品详情页面	高
T05	购物车	添加购物车	把商品加入购物车	高
T06	购物车	查看购物车	用户查看购物车内的商品	高
T07	订单	商品结算	对已选择的商品进行结算	高
T08	订单	提交订单	用户提交商品订单	高
T09	订单	查看我的订单	用户查看订单列表	高

76

程序员-软件测试

3. 确定性能测试目标

轻商城作为一个新开发的项目，性能测试目标包括：

1. 确定核心业务功能的TPS
2. 对业务流程（多接口组合）进行压测
3. 系统能在实际系统运行压力的情况下，稳定的运行24小时

期望的TPS和最大响应时间：

编号	功能模块	业务功能	TPS	响应时间
T01	登录	登录	20	3s
T02	首页	进入首页	100	5s
T03	商品	搜索商品	40	3s
T04	商品	查看商品详情	100	3s
T05	购物车	添加购物车	20	3s
T06	购物车	查看购物车	20	3s
T07	订单	商品结算	10	3s
T08	订单	提交订单	10	3s
T09	订单	查看我的订单	40	2s

程序员-软件测试

性能测试计划

目标

1. 掌握性能测试计划包含的主要内容

1. 测试背景

轻商城是公司新开发的一个电商项目，为了保证项目上线后能够稳定的运行，且在后期推广中能够承受用户的增长，需要对项目进行性能测试。

2. 测试目的

对新电商项目进行性能测试的核心目的包括：

- 确定核心业务功能的TPS
- 对业务流程（多接口组合）进行压测
- 系统能在实际系统运行压力的情况下，稳定的运行24小时

3. 测试范围

通过对性能测试需求的调研和分析，确定被测系统的测试范围如下

编号	功能模块	业务功能	功能描述	优先级
T01	登录	登录	用户通过用户名和密码登录	高
T02	首页	进入首页	获取商城首页数据	高
T03	商品	搜索商品	通过关键字搜索商品	高
T04	商品	查看商品详情	点击商品进入商品详情页面	高
T05	购物车	添加购物车	把商品加入购物车	高
T06	购物车	查看购物车	用户查看购物车内的商品	高
T07	订单	商品结算	对已选择的商品进行结算	高
T08	订单	提交订单	用户提交商品订单	高
T09	订单	查看我的订单	用户查看订单列表	高

4. 测试策略

4.1 基准测试

先做基准测试，确定估算的标准。

4.2 负载测试

- 通过逐步增加系统负载，测试系统性能的变化，并最终确定在满足系统的性能指标情况下，系统所能承受的最大负载量的测试。
- 分别模拟5、10、30、50、100个用户对系统进行负载测试，查看不同并发时系统软件各项指标是否符合需求。

4.3 稳定性测试

78

程序员-软件测试

- 用200用户对系统进行7*24小时的不间断稳定性测试，查看服务器日志内有无异常和报错；系统软件各项指标中间有无异常波动；是否存在内存溢出之类的问题。
- 验证系统长期运行的稳定性以及是否存在内存溢出之类的问题

5. 风险控制

风险类型	风险描述	风险级别	应对方案
环境风险	部署出现问题，联调进度缓慢	中	更换环境、增加资源配置
数据风险	构造测试数据时间较长	中	开发人员协助
交付风险	发现比较严重的Bug	中	延长测试时间，增加对应人员

6. 交付清单

性能测试计划、测试脚本、性能缺陷统计和性能测试报告等。

7. 进度与分工

阶段	事项	开始时间	结束时间	状态	责任人
需求阶段	需求评审			完成	多方参与
	系统架构图			完成	开发
	需求调研			完成	性能测试人员
准备阶段	环境交付			完成	运维、开发
	应用部署			完成	运维、开发
	数据准备			完成	开发、DBA、测试
	脚本开发			完成	性能测试人员
实施阶段	执行压测			未完成	性能测试人员
	服务监控			未完成	运维、测试
	数据收集			未完成	性能测试人员
结束	报告评审			未完成	多方评审

程序员-软件测试

测试用例设计

目标

1. 编写性能测试用例

用例名称	获取首页数据			
用例编号	Index001			
用例描述	TPS达到100的情况下，进入首页的时间不超过5s			
前置条件	首页相关的商品数据已经配置完成			
用例步骤	动作	期望的性能		
1	进入商城首页	<5s		
2				
并发用户数与事务响应				
并发用户数	事务平均响应时间	事务最大响应时间	平均每秒处理事务数（TPS）	事务成功率
5				
10				
30				
50				
100				
并发用户数与服务器性能				
并发用户数	CPU利用率	内存利用率	磁盘IO情况	其他参数
5				
10				
30				
50				
100				
并发用户数与数据库性能				
并发用户数	CPU利用率	内存利用率	磁盘IO情况	其他参数
5				
10				
30				
50				
100				

程序员-软件测试

测试脚本开发

目标

1. 熟练掌握如何编写JMeter测试脚本

1. 测试脚本开发

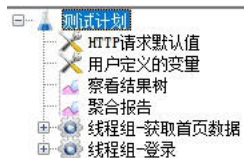
使用JMeter编写测试脚本并调试

1.1 常用测试元件

1. 取样器-HTTP请求
2. 配置元件-HTTP请求默认值
3. 配置元件-用户定义的变量
4. 后置处理器-JSON提取器
5. 断言-响应断言
6. 断言-JSON断言
7. 监听器-察看结果树
8. 监听器-聚合报告

1.2 初始化工作

1. 创建测试用例结构
2. 设置HTTP请求默认值
3. 用户定义的变量
4. 添加监听器-察看结果树
5. 添加监听器-聚合报告



实现测试用例

获取首页数据

操作步骤：

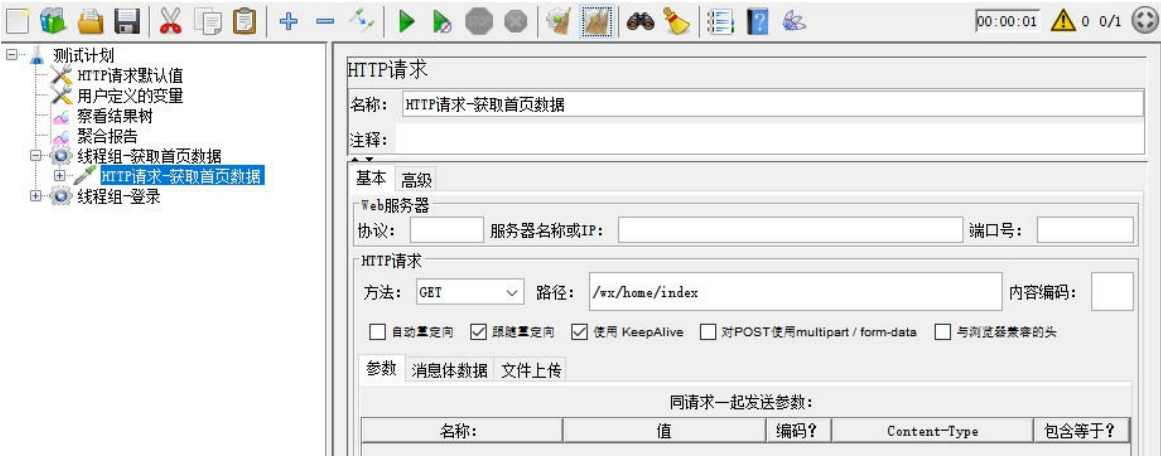
1. 在线程组下，添加取样器'HTTP请求-获取首页数据'，并填写请求数据
2. 在取样器下，添加'响应断言'断言响应状态码
3. 在取样器下，添加'JSON断言'断言errno和errmsg
4. 发送请求，调试脚本

实现截图：

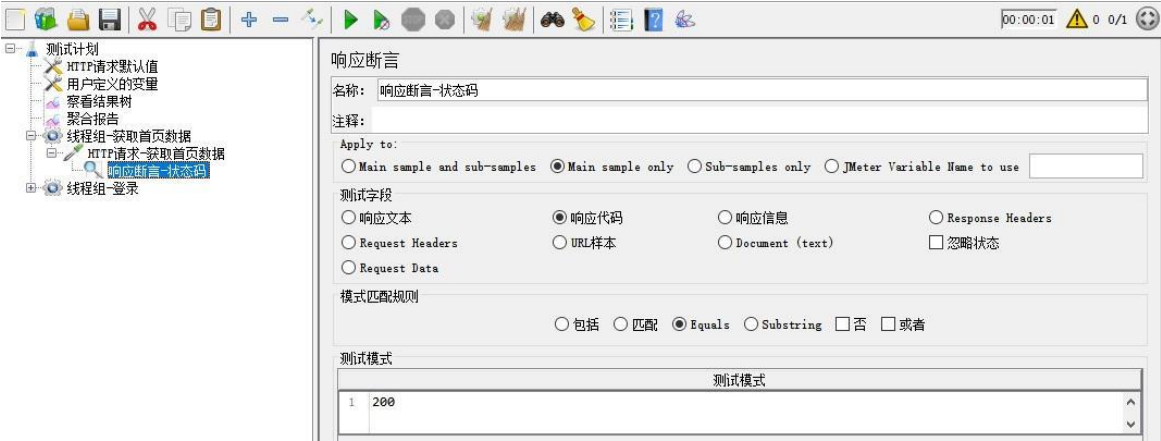
1. 添加'HTTP请求-获取首页数据'的请求

81

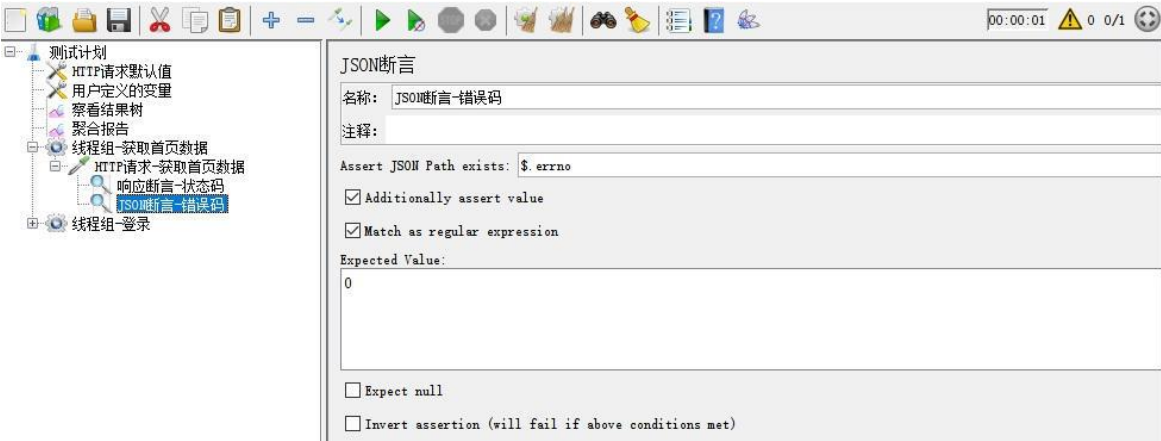
程序员-软件测试



2. 添加'响应断言-响应状态码'

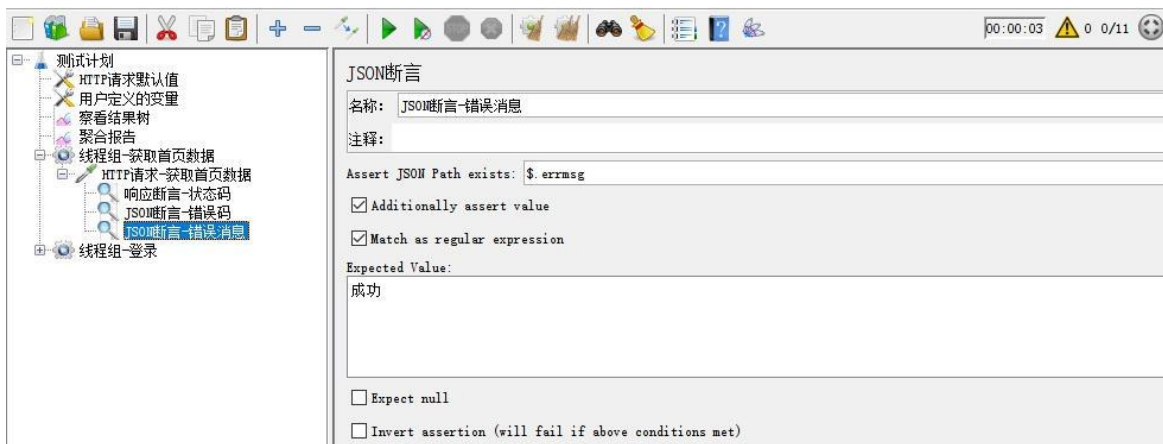


3. 添加'JSON断言-错误码'



4. 添加'JSON断言-错误消息'

程序员-软件测试



程序员-软件测试

建立测试环境

目标

1. 知道如何建立性能测试环境

1. 建立测试环境

- 在进行性能测试之前，需要先完成性能测试环境的搭建工作，测试环境一般包括硬件环境、软件环境及网络环境
- 一般情况下可以要求运维和开发工程师协助完成

1.1 性能测试环境的特点

1. 性能测试对测试环境的独立性要求更高，更为严格
 - 如果某环境下运行多个系统，就很难判断其中的某个环境对资源的占用情况
2. 尽量保持性能测试环境与真实生产环境的一致性

1.2 如何保证测试环境与生产环境的一致性

1. 硬件环境
 - 包括服务器环境、网络环境等
2. 软件环境
 - 版本一致性：包括操作系统、数据库、被测应用程序、第三方软件等
 - 配置一致性：包括操作系统、数据库、被测应用程序、第三方软件等
3. 使用场景的一致性
 - 基础业务数据的一致性
 - 业务操作模式的一致性：尽量模拟真实场景下用户的使用情况

2. 构造测试数据

压测环境中的数据量尽量与生产环境中数据量一致，为了快速创建大量数据，可以直接操作数据库进行添加

2.1 构造商品数据

通过编写Python脚本，构造10万条商品记录
参考代码：

```
import pymysql

conn = pymysql.connect(host="127.0.0.1", user="litemall", password="litemall123456", database="litemall", port=3306)
cursor = conn.cursor()
```



```

goods_sql = """
INSERT INTO `litemall_goods` (`id`, `goods_sn`, `name`, `category_id`, `brand_id`, `gallery`,
`keywords`, `brief`, `is_on_sale`, `sort_order`, `pic_url`, `share_url`, `is_new`, `is_hot`, `unit`,
`counter_price`, `retail_price`, `detail`, `add_time`, `update_time`, `deleted`)
VALUES ({}, {}, '小米手机-{}'.format('1008016', '1001000'), '["http://182.92.81.159:8080/wx/storage/fetch/0b51pf15ee8c6010vkd8.jpg","\
"http://182.92.81.159:8080/wx/storage/fetch/ykaptr4vntbofoi912f5.jpg","\
"http://182.92.81.159:8080/wx/storage/fetch/u5zc8sp2t4f9y9uwn179.jpg"]',
'手机,Android', '小米10 双模5G 骁龙865 1亿像素8K电影相机', '1', '100', 'http://182.92.81.159:8080/wx/storage/fetch/re5jul69plklzusso97
t.png', '', '1', '0', '个', '100.00', '99.00', '<p>小米10 双模5G 骁龙865 1亿像素8K电影相机 对称式立体声 8GB+256GB 冰海蓝 拍照智能游戏手机
!!!!!!!!!!!!</p>',
'2020-03-11 14:19:50', '2020-03-26 14:55:58', '0');
"""

```

84

程序员-软件测试

```

goods_attr_sql = """
INSERT INTO `litemall_goods_attribute` (`goods_id`, `attribute`, `value`, `add_time`, `update_time`, `deleted`)
VALUES
({}, '产地', '中国山东', '2018-10-26 21:27:13', '2018-10-26 21:27:13', '0'),
({}, '尺寸', '200*230cm/ 220*240cm', '2018-10-26 21:27:13', '2018-10-26 21:27:13', '0'),
({}, '颜色', '条纹绿格', '2018-10-26 21:27:13', '2018-10-26 21:27:13', '0'),
({}, '执行标准', 'GB/T 22844-2009', '2018-10-26 21:27:13', '2018-10-26 21:27:13', '0');
"""

goods_product_sql = """
INSERT INTO `litemall_goods_product` (`goods_id`, `specifications`, `price`, `number`, `url`, `add_time`, `update_time`, `deleted`)
VALUES ({}, '['标准']', '999.00', '1000', 'http://182.92.81.159:8080/wx/storage/fetch/o4531ja3h9sgq5ib32f4.jpg', '2020-03-11 14:
19:50', '2020-03-11 14:19:50', '0');
"""

goods_spec_sql = """
INSERT INTO `litemall_goods_specification` (`goods_id`, `specification`, `value`, `pic_url`, `add_time`, `update_time`, `deleted`)
VALUES ({}, '规格', '标准', '', '2020-03-11 14:19:50', '2020-03-11 14:19:50', '0');
"""

goods_id_start = 200000
for i in range(100000):
    # 商品
    goods_id = goods_id_start + i
    print("i={} goods_id={}".format(i, goods_id))
    sql = goods_sql.format(goods_id, goods_id, goods_id)
    cursor.execute(sql)

    # 商品参数
    sql = goods_attr_sql.format(goods_id, goods_id, goods_id, goods_id)
    cursor.execute(sql)

    # 商品货品
    sql = goods_product_sql.format(goods_id)
    cursor.execute(sql)

    # 商品规格表
    sql = goods_spec_sql.format(goods_id)
    cursor.execute(sql)

    conn.commit()

cursor.close()
conn.close()

```

程序员-软件测试

执行测试脚本

目标

1. 执行测试脚本的测试机

先保证脚本调试通过之后，才能进入正式压测阶段。通常会选择Windows或者Linux环境来执行：

- Windows环境：操作界面化、直观、易上手，但是软件占用机器资源较多，导致资源使用率不高；可支持并发较低。
- Linux环境：命令行操作，结果查看不太方便，但资源利用率相对较高；可支持较高并发。

2. 分布式执行

如果单台压测机的并发量不能够满足要求，则可以通过分布式压测来提高并发量。 JMeter工具支持分布式压测，即多台机器同时执行同一个脚本，然后统计结果。

2.1 分布式压测条件

分布式压测，机器分为控制机和执行机，有几个条件必须满足：

- 执行机和控制机必须在同一个网段之内
- 压测机也必须安装相同版本的JMeter和JDK

2.2 修改JMeter配置信息

1. 修改控制机jmeter.properties配置文件中的 `remote_hosts=xx` xx表示压力机的IP和端口，多个用英文逗号分隔
2. 修改执行机jmeter.properties配置文件中的 `server_port=1099` 端口号要和控制机配置文件中保持一致

2.3 启动执行机

打开jmeter目录bin\jmeter-server.bat文件进行启动

2.4 启动控制机

打开jmeter目录bin\jmeter.bat，添加执行脚本，通过运行-->远程启动或者远程全部启动，即可启动对应执行机或者全部执行机

程序员-软件测试

性能测试监控

目标

2. 知道

1. 性能测试监控关键指标

1. 系统指标：系统指标则与用户场景及需求直接相关
 - 并发用户数：某一物理时刻同时向系统提交请求的用户数
 - 平均响应时间：系统处理事务的响应时间的平均值。对于系统快速响应类页面，一般响应时间为3秒左右
 - 吞吐量
2. 服务器资源指标：资源指标与硬件资源消耗直接相关
 - CPU使用率：一般可接受上限为85%
 - 内存利用率：一般可接受上限为85%
 - 磁盘I/O
 - 网络带宽
3. Java应用：
 - JVM监控：JVM内存、Full GC频率
4. 数据库：
 - 慢查询
 - 缓存命中率
 - 数据池连接数
 - mysql锁
5. 压测机资源：
 - CPU
 - 内存
 - 磁盘空间

2. 性能监控工具

-
- 网络

要对性能测试指标进行监控，可以使用系统自带的监控工具，也可以使用第三方监控工具或者监控平台。

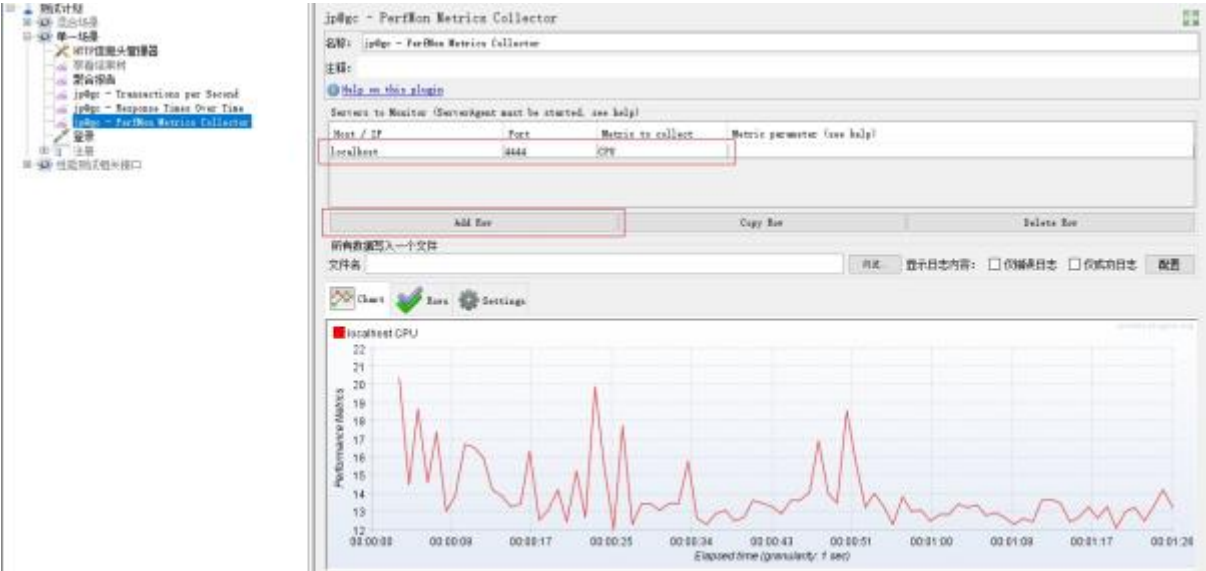
1. 系统指标：
 - 通过性能测试工具(如LoadRunner、JMeter等)以图形化方式监控
2. 服务器资源指标：
 - 使用Jmeter性能监控插件PerfMon 进行监控
 - 使用Linux命令监控：top、free、vmstat、sar、iostat等

- Nmon: 全面监控linux系统资源使用情况，包括CPU、内存、I/O等，可独立于应用监控。
3. Java应用:
 - jvisualvm
 4. 数据库
 5. 压测机资源:
 - Windows自带“任务管理器”

3. 服务器资源

程序员-软件测试

使用JMeter性能监控插件PerfMon Metrics Collector监控服务器资源



4. MySQL监控

4.1 Mysql常用监控指标

- 慢查询SQL
 - 慢查询: 指执行速度低于设置的阈值的SQL语句
 - 作用: 帮助定位查询速度较慢的SQL语句，方便更好的优化数据库系统的性能

4.2 开启MySQL慢查询日志

- slow_query_log: 慢查询日志开启状态[ON:开启，OFF:关闭]
- slow_query_log_file: 慢查询日志存放位置
- long_query_time: 慢查询时长设置（超过该时长才会被记录，单位：秒）

1. 查询相关参数配置
参数说明:
设置步骤:

```
mysql> show variables like 'slow_query%';
+-----+-----+
| Variable_name | Value |
+-----+-----+
| slow_query_log | OFF   |
| slow_query_log_file | /var/lib/mysql/iZ2ze6id2tw1o2zn0mznxZ-slow.log |
+-----+-----+
2 rows in set

mysql> show variables like 'long_query_time';
```

```
+-----+-----+
| Variable_name | Value |
+-----+-----+
| long_query_time | 10.000000 |
+-----+-----+
1 row in set
```

2. 开启慢查询并配置

```
# 开启慢查询日志
mysql> set global slow_query_log='ON';
# 设置慢查询日志存放位置
mysql> set global slow_query_log_file='/data/slow_query.log';
```

88

程序员-软件测试

```
# 设置慢查询时间标准，设置之后会在下次会话才生效
mysql> set global long_query_time=1;
```

5. JVM监控

使用本地jvisualvm远程监控服务器

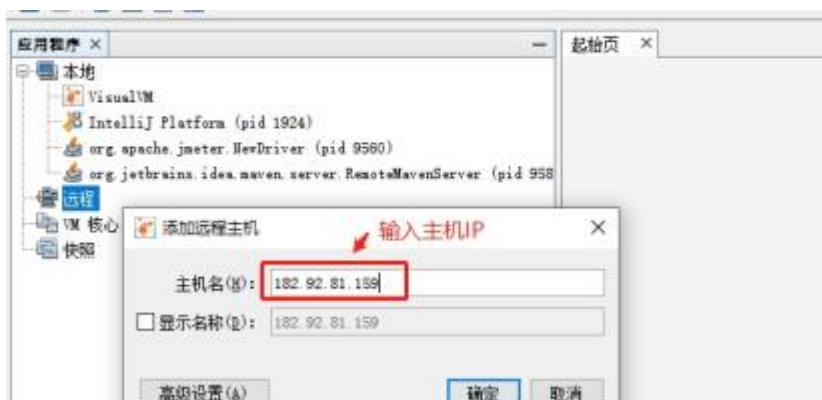
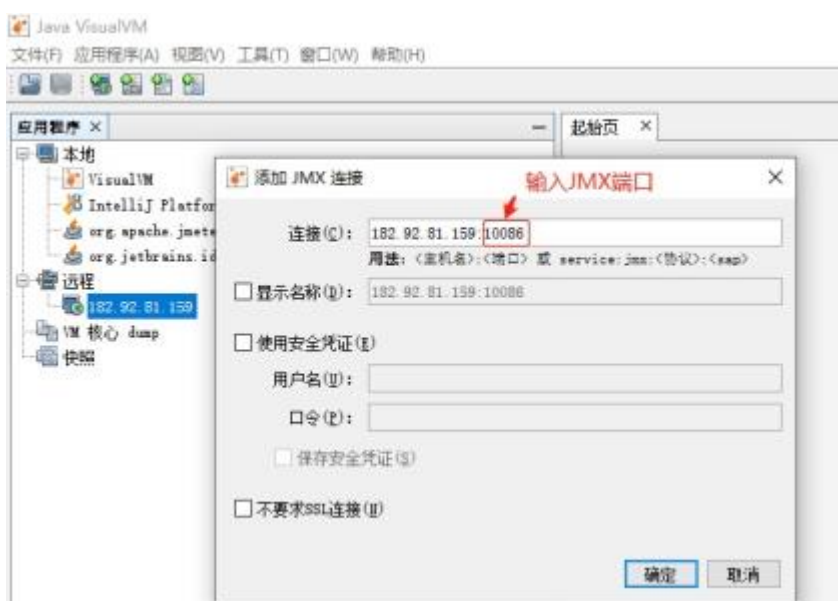
1. 添加应用程序启动参数，并启动服务

```
-Dcom.sun.management.jmxremote
-Djava.rmi.server.hostname=182.92.81.159
-Dcom.sun.management.jmxremote.port=10086
-Dcom.sun.management.jmxremote.ssl=false
-Dcom.sun.management.jmxremote.authenticate=false
```

2. 进入本地jdk安装目录bin目录，找到jvisualvm.exe并启动
3. 右键“远程”选择“添加远程主机”，并输入主机IP

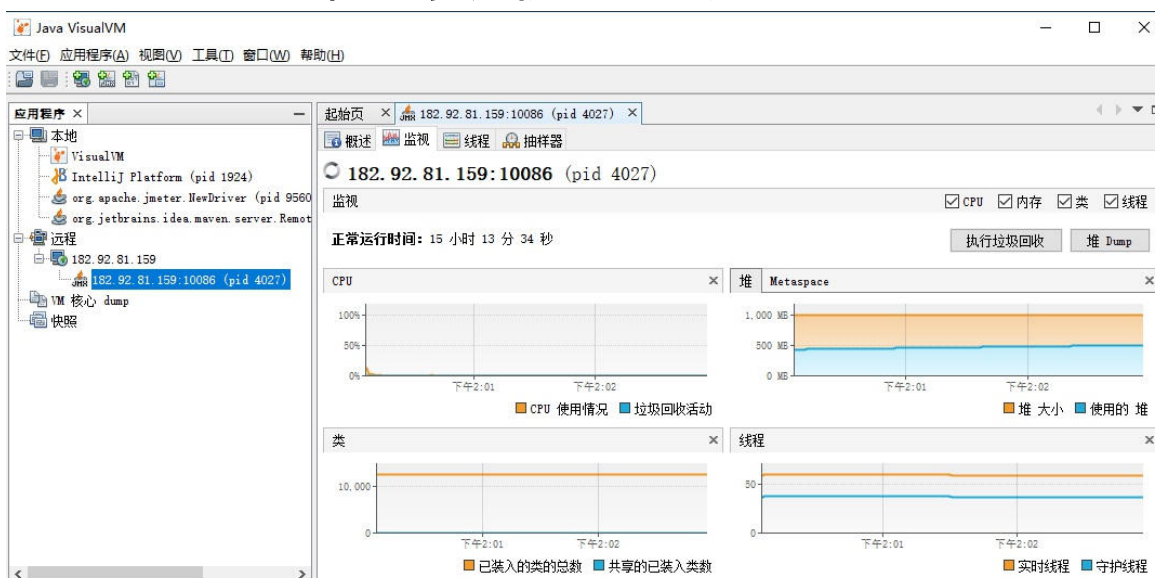
4. 右键主机选择“添加JMX连接”，并输入JMX端口

5. 连接成功后在主机下会有对应的连接显示，双击查看监控信息



89

程序员-软件测试



jmx服务会同时启动其它两个随机端口，需要把对应的端口开放

程序员-软件测试

性能分析和调优

目标

2. 知道

1. 性能测试瓶颈分析

在实际的性能测试中，会遇到各种各样的问题，比如TPS压不上去，导致这种现象的原因很多，作为测试人员应配合开发人员进行分析尽快找出瓶颈的所在。

常见性能瓶颈分析：

1. 服务器资源分析

- CPU瓶颈分析

- CPU已压满（接近100%），需要再看其他指标的拐点出现的时刻是否与CPU压满的时刻基本一致

- 内存瓶颈分析

- 内存不足时，操作系统会使用虚拟内存，从虚拟内存读取数据，影响处理速度

- 磁盘I/O瓶颈分析

- 磁盘I/O成为瓶颈时，会出现磁盘I/O繁忙，导致交易执行时在I/O处等待

- 网络带宽

- 如果接口传递的数据包过大，超过了带宽的传输能力，就会造成网络资源竞争，导致TPS上不去

2. JVM瓶颈分析

- 分析JVM的内存

3. 数据库瓶颈分析

- 慢查询

- 数据库的连接池设置太小，导致数据库连接出现排队

- 数据库出现死锁

4. 程序内部实现机制

JMeter单机负载能力有限，如果需要模拟的用户请求数超过其负载极限，也会导致TPS压不上去

5. 压测机

2. 性能调优

2.1 性能调优的步骤

1. 确定问题：根据性能监控的数据和性能分析的结果，确定性能存在的问题（要求）
2. 确定原因：确定了问题之后，对问题进行分析，找出问题的原因
3. 确定调整目标和解决方案（改服务器参数配置/增加硬件资源配置/修改代码）
4. 测试解决方案
5. 分析调优结果

注意：性能测试调优并不是一次完成的过程，针对同一个性能问题，上面的五步可能要经过多次循环才能最终完成性能调优的目标（即：测试发现问题-找原因-调整-验证-分析-再测试。。。）

程序员-软件测试

3. 性能调优案例

3.1 获取首页数据

场景描述

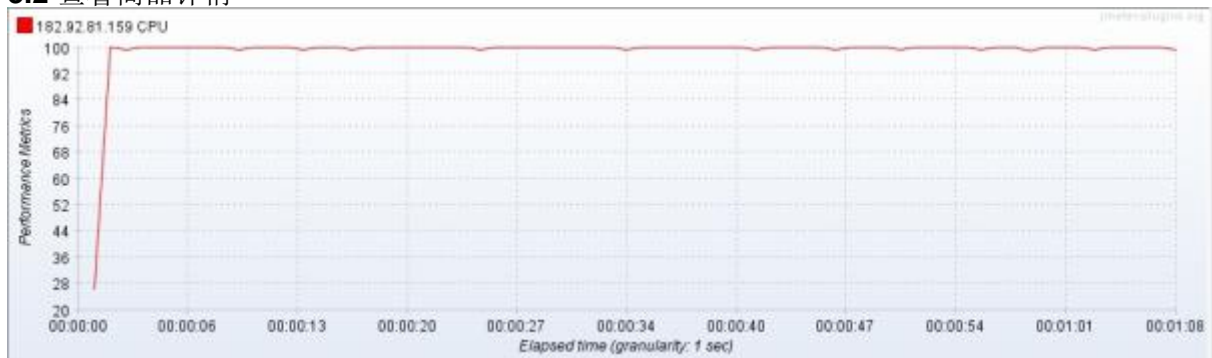
进入首页后，加载首页的相关数据，包括：轮播图、频道、优惠券、团购专区、品牌商直供、新品首发、热卖商品、专题精选等数据

测试结果数据

1. CPU已接近100%
2. 一次请求中需要查询很多数据

1. 提升服务器配置
2. 分批次、异步加载首页数据，首页底部的数据（如：新品首发、热卖商品、专题精选等数据）等用户向下滑动页面时再加载

3.2 查看商品详情



进入商品详情页面时，加载商品的详细信息

测试结果数据

问题分析

解决方案

场景描述



问题分析

1. 网络带宽已跑满
2. 一次请求中返回了全部数据

1. 提升服务器网络带宽
2. 分批次、异步加载商品数据

3.3 搜索商品

进入首页，在搜索框中输入关键字搜索商品

测试结果数据

1. 搜索关键字“床”时，出现慢查询SQL语句

查看慢查询语句

```
cat /var/lib/mysql/localhost-slow.log
```

解决方案

场景描述

```
select id, `name`, keywords, `desc`, pid, icon_url, pic_url, `level`, sort_order, add_time, update_time, deleted
from litemall_category
WHERE ( id in (1008009, 1008009, 1008008, 1008008, 1015000, 1015000, 1008009, 1008009, 1008009, 1008008, 1036000) and `level` = 'L2'
and deleted = 0);
```

问题分析

1. 找出搜索商品接口对应的SQL语句（通过查看代码实现或者从日志中获取查询SQL）

日志信息：

```
11:21:39.380 logback DEBUG o.s.web.servlet.DispatcherServlet - GET "/wx/goods/list?keyword=%E5%BA%8A&page=1&limit=10&categoryId=0", parameters={masked}
11:21:39.380 logback DEBUG o.s.w.s.m.m.a.RequestMappingHandlerMapping - Mapped to public java.lang.Object org.linlinjava.litema
ll.wx.web.WxGoodsController.list(java.lang.Integer,java.lang.Integer,java.lang.String,java.lang.Boolean,java.lang.Boolean,java.
lang.Integer,java.lang.Integer,java.lang.Integer,java.lang.String,java.lang.String)
```

```
11:21:39.382 logback DEBUG org.mybatis.spring.SqlSessionUtils - Creating a new SqlSession
11:21:39.383 logback DEBUG org.mybatis.spring.SqlSessionUtils - SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@22d838e7] was not registered for synchronization because synchronization is not active
11:21:39.389 logback DEBUG o.m.s.t.SpringManagedTransaction - JDBC Connection [com.alibaba.druid.proxy.jdbc.ConnectionProxyImpl@5b5faeed] will not be managed by Spring
11:21:39.389 logback DEBUG o.l.l.d.d.l.selectByExampleSelective_COUNT - ==> Preparing: SELECT count(0) FROM litemall_goods WHERE (keywords LIKE ? AND is_on_sale = ? AND deleted = ?) OR (`name` LIKE ? AND is_on_sale = ? AND deleted = ?)
11:21:39.391 logback DEBUG o.l.l.d.d.l.selectByExampleSelective_COUNT - ==> Parameters: %床%(String), true(Boolean), false(Boolean), %床%(String), true(Boolean), false(Boolean)
11:21:39.396 logback DEBUG o.l.l.d.d.l.selectByExampleSelective_COUNT - <== Total: 1
11:21:39.398 logback DEBUG o.l.l.d.d.l.selectByExampleSelective - ==> Preparing: select id, `name`, brief, pic_url, is_hot, is_new, counter_price, retail_price from litemall_goods WHERE ( keywords like ? and is_on_sale = ? and deleted = ? ) or( `name` like ? and is_on_sale = ? and deleted = ? ) order by add_time desc LIMIT ?
11:21:39.399 logback DEBUG o.l.l.d.d.l.selectByExampleSelective - ==> Parameters: %床%(String), true(Boolean), false(Boolean), %床%(String), true(Boolean), false(Boolean), 10(Integer)
11:21:39.403 logback DEBUG o.l.l.d.d.l.selectByExampleSelective - <== Total: 10
11:21:39.403 logback DEBUG org.mybatis.spring.SqlSessionUtils - Closing non transactional SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@22d838e7]
11:21:39.404 logback DEBUG org.mybatis.spring.SqlSessionUtils - Creating a new SqlSession
11:21:39.405 logback DEBUG org.mybatis.spring.SqlSessionUtils - SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5677e0a5] was not registered for synchronization because synchronization is not active
11:21:39.406 logback DEBUG o.m.s.t.SpringManagedTransaction - JDBC Connection [com.alibaba.druid.proxy.jdbc.ConnectionProxyImpl@5b5faeed] will not be managed by Spring
11:21:39.406 logback DEBUG o.l.l.d.d.l.selectByExampleSelective - ==> Preparing: select category_id from litemall_goods WHERE ( keyword like ? and is_on_sale = ? and deleted = ? ) or( `name` like ? and is_on_sale = ? and deleted = ? )
11:21:39.407 logback DEBUG o.l.l.d.d.l.selectByExampleSelective - ==> Parameters: %床%(String), true(Boolean), false(Boolean), %床%(String), true(Boolean), false(Boolean)
11:21:39.409 logback DEBUG o.l.l.d.d.l.selectByExampleSelective - <== Total: 11
11:21:39.410 logback DEBUG org.mybatis.spring.SqlSessionUtils - Closing non transactional SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@5677e0a5]
11:21:39.411 logback DEBUG org.mybatis.spring.SqlSessionUtils - Creating a new SqlSession
11:21:39.411 logback DEBUG org.mybatis.spring.SqlSessionUtils - SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@2d594a27] was not registered for synchronization because synchronization is not active
11:21:39.413 logback DEBUG o.m.s.t.SpringManagedTransaction - JDBC Connection [com.alibaba.druid.proxy.jdbc.ConnectionProxyImpl@5b5faeed] will not be managed by Spring
11:21:39.414 logback DEBUG o.l.l.d.d.l.selectByExample - ==> Preparing: select id, `name`, keywords, `desc`, pid, icon_url, pic_url, `level`, sort_order, add_time, update_time, deleted from litemall_category WHERE ( id in ( ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ? ) and `level` = ? and deleted = ? )
11:21:39.415 logback DEBUG o.l.l.d.d.l.selectByExample - ==> Parameters: 1008009(Integer), 1008009(Integer), 1008008(Integer), 1008008(Integer), 1015000(Integer), 1015000(Integer), 1008009(Integer), 1008009(Integer), 1008009(Integer), 1008008(Integer), 1036000(Integer), L2(String), false(Boolean)
11:21:39.417 logback DEBUG o.l.l.d.d.l.selectByExample - <== Total: 4
11:21:39.418 logback DEBUG org.mybatis.spring.SqlSessionUtils - Closing non transactional SqlSession [org.apache.ibatis.session.defaults.DefaultSqlSession@2d594a27]
11:21:39.419 logback DEBUG o.s.w.s.m.m.a.RequestResponseBodyMethodProcessor - Using 'application/json', given [*/] and supported [application/json, application/*+json, application/json, application/*+json]
11:21:39.420 logback DEBUG o.s.w.s.m.m.a.RequestResponseBodyMethodProcessor - Writing [{errno=0, data={total=11, pages=2, limit=10, page=1, list=Page(count=true, pageNum=1, pageSize=10, s (truncated)...}]
11:21:39.424 logback DEBUG o.s.web.servlet.DispatcherServlet - Completed 200 OK
```

2. 分析具体的SQL语句

```
-- 1.查询满足条件的商品总数量
SELECT count(0) FROM litemall_goods WHERE (keywords LIKE '%床%' AND is_on_sale = 1 AND deleted = 0) OR (`name` LIKE '%床%' AND is_on_sale = 1 AND deleted = 0);

--
-- 2.分页获取满足条件的商品列表
select id, `name`, brief, pic_url, is_hot, is_new, counter_price, retail_price from litemall_goods WHERE ( keywords like '%床%' and is_on_sale = 1 and deleted = 0 ) or( `name` like '%床%' and is_on_sale = 1 and deleted = 0) order by add_time desc LIMIT 10;

--
-- 3.查询满足条件的商品分类id
select category_id from litemall_goods WHERE ( keywords like '%床%' and is_on_sale = 1 and deleted = 0) or( `name` like '%床%' and is_on_sale = 1 and deleted = 0);

--
-- 4.根据商品分类id获取商品分类信息
select id, `name`, keywords, `desc`, pid, icon_url, pic_url, `level`, sort_order, add_time, update_time, deleted from litemall_category WHERE ( id in (1008009, 1008009, 1008008, 1008008, 1015000, 1015000, 1008009, 1008009, 1008009, 1008008, 1036000) and `level` = 'L2' and deleted = 0);
```

程序员-软件测试

3. 定位问题

- 当搜索关键字匹配到大量的商品时，第3条SQL语句会返回大量重复数据
- 第4条SQL语句中的in查询条件中同样包含大量重复的商品分类id

- 第3条和第4条SQL都会出现查询时间较长

解决方案

1. 优化第3条SQL语句，因该SQL语句的查询结果大部分都是重复的，可以进行去重处理

```
select DISTINCT category_id
from litemall_goods
WHERE (keywords like '%床%' and is_on_sale = 1 and deleted = 0)
or ('name' like '%床%' and is_on_sale = 1 and deleted = 0);
```

3.4 JVM内存溢出

场景描述

请求测试接口（/wx/index/oom），模拟内存溢出

测试结果数据



问题分析

1. JVM内存占用随着时间的推移占用越来越多，直至内存溢出，系统退出

解决方案

程序员-软件测试

1. 排查代码存在的问题，及时释放无用的对象

编写测试报告

目标

编写测试报告的要点

- 1. 结构清晰
- 2. 描述简洁
- 3. 图文混合
- 4. 数据对比

1. 项目概况

litemall是公司新开发的一个电商项目，为了保证项目上线后能够稳定的运行，且在后期推广中能够承受用户的增长，需要对项目进行性能测试。

2. 测试目的

对新电商项目进行性能测试的核心目的包括：

- 确定核心业务功能的TPS
- 对业务流程（多接口组合）进行压测
- 系统能在实际系统运行压力的情况下，稳定的运行24小时

3. 测试范围

通过对性能测试需求的调研和分析，确定被测系统的测试范围如下：

编号	功能模块	业务功能	功能描述	优先级
T01	登录	登录	用户通过用户名和密码登录	高
T02	首页	进入首页	获取商城首页数据	高
	商品	搜索商品	通过关键字搜索商品	
	商品	查看商品详情	点击商品进入商品详情页面	

4. 测试环境及工具

4.1 性能测试环境的基本配置

设备	IP地址	硬件配置	软件配置
		CPU：2核	

服务器	182.92.81.159	RAM: 4GB Disk: 200GB	JDK 1.8 MySQL 5.7
压测机	127.0.0.1	CPU: 4核 RAM: 8GB	Windows10 JMeter5.0

4.2 测试工具

- 负载工具：JMeter
- 监控工具：PerfMon Metrics Collector

5. 测试记录及结果分析

进入首页后，加载首页的相关数据，包括：轮播图、频道、优惠券、团购专区、品牌商直供、新品首发、热卖商品、专题精选等

测试结果数据

1. CPU已接近100%
2. 一次请求中需要查询很多数据

1. 提升服务器配置

5.1 单场景负载测试-获取首页数据

场景描述

数据



问题分析

解决方案

2. 分批次、异步加载首页数据，首页底部的数据（如：新品首发、热卖商品、专题精选等数据）等用户向下滑动页面时再加载

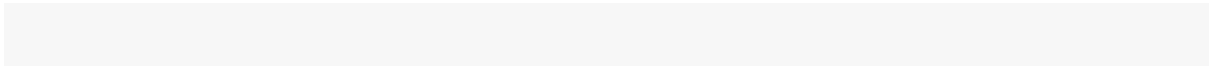
6. 测试结论

98

程序员-软件测试

第4章-**Locust**框架

目标



程序员-软件测试

Locust介绍和安装

目标

1. 了解Locust的相关特点

1. Locust简介

Locust是一个开源的性能测试工具，主要思想就是模拟一群用户访问你的系统。

1.1 特点

1. 在代码中定义用户行为
 - 不需要安装笨重的软件，只是简单的Python代码
2. 分布式和可扩展
 - Locust支持在多台机器上的运行负载测试，因此可用于模拟数百万用户的请求
3. 经过验证和战斗测试
 - Locust被用于许多真实的项目中
4. Locust有一个整洁的HTML+JS的用户界面，实时显示相关测试细节
 - 由于用户界面是基于网络的，它是跨平台的和容易扩展

2. Locust安装

安装命令：

```
pip install locustio==0.12.2
```

程序员-软件测试

Locust使用

目标

1. 掌握如何使用Locust编写测试脚本

1. 案例演示环境说明

某系统包含以下接口：

- - URL: <http://182.92.81.159:1880/bms/login>
请求方式: POST
请求参数: {"username": "admin", "password": "123456"}
 - URL: <http://182.92.81.159:1880/bms/index>
请求方式: GET
- 3. 获取用户信息
 - URL: <http://182.92.81.159:1880/bms/profile>
请求方式: GET
 - URL: <http://182.92.81.159:1880/bms/logout>
请求方式: POST

2. 编写测试脚本

1. 定义任务（接口请求）
2. 定义任务集（用户行为）
3. 定义Locust类（用户）

2.1 定义任务

locust里发送请求是基于requests实现的，请求方法、参数、响应对象和requests使用方式一样。

1. 登录

2. 首页

4. 退出

实现步骤：

```
def index(l):
    l.client.get("/index")

def login(l):
    l.client.post("/login", data={"username": "admin", "password": "123456"})
```

2.2 定义任务集

定义一个用户行为（任务集），包含多个具体的任务。

```
from locust import TaskSet

class UserBehavior(TaskSet):
    tasks = {index: 3, profile: 1}
```

101

程序员-软件测试

```
def on_start(self):
    login(self)

def on_stop(self):
    logout(self)
```

如何定义？

- 一个用户行为类，要继承TaskSet类，表示一个任务集
- on_start: 前置方法(前置任务)，在所有任务之前调用
- on_stop: 后置方法(后置任务)，当任务集停止时调用
- tasks: 用来添加任务，它是一个dict类型，key表示任务的方法名，value表示挑选执行的权重，数值越大执行频率越高

2.3 定义Locust类

定义一个Locust类，这个类代表用户。

如何定义？

- 自定义的Locust类继承了Locust类，这个类代表用户，生成一个实例，模拟用户发送http请求
- task_set: 该属性指向TaskSet类，定义用户的行为
- min_wait: 用户执行任务之间等待时间的下界，单位：毫秒，默认值：1000
- max_wait: 用户执行任务之间等待时间的上界，单位：毫秒，默认值：1000
- host: 被测应用的网址，例如：<http://localhost:8080/bms>
- weight: 用户被选中的概率，权重越大，被选中的机会就越大。默认值：10

2.4 示例代码

```

from locust import HttpLocust, TaskSet

def login(l):
    l.client.post("/login", data={"username": "admin", "password": "123456"})

def logout(l):
    l.client.post("/logout")

def index(l):
    l.client.get("/index")

def profile(l):
    l.client.get("/profile")

class UserBehavior(TaskSet):
    tasks = {index: 3, profile: 1}

    def on_start(self):
        login(self)

    def on_stop(self):
        logout(self)

```

102

程序员-软件测试

```

class WebsiteUser(HttpLocust):
    task_set = UserBehavior
    min_wait = 5000
    max_wait = 9000
    host = "http://182.92.81.159:1880/bms"

```

3. 运行Locust

运行命令：

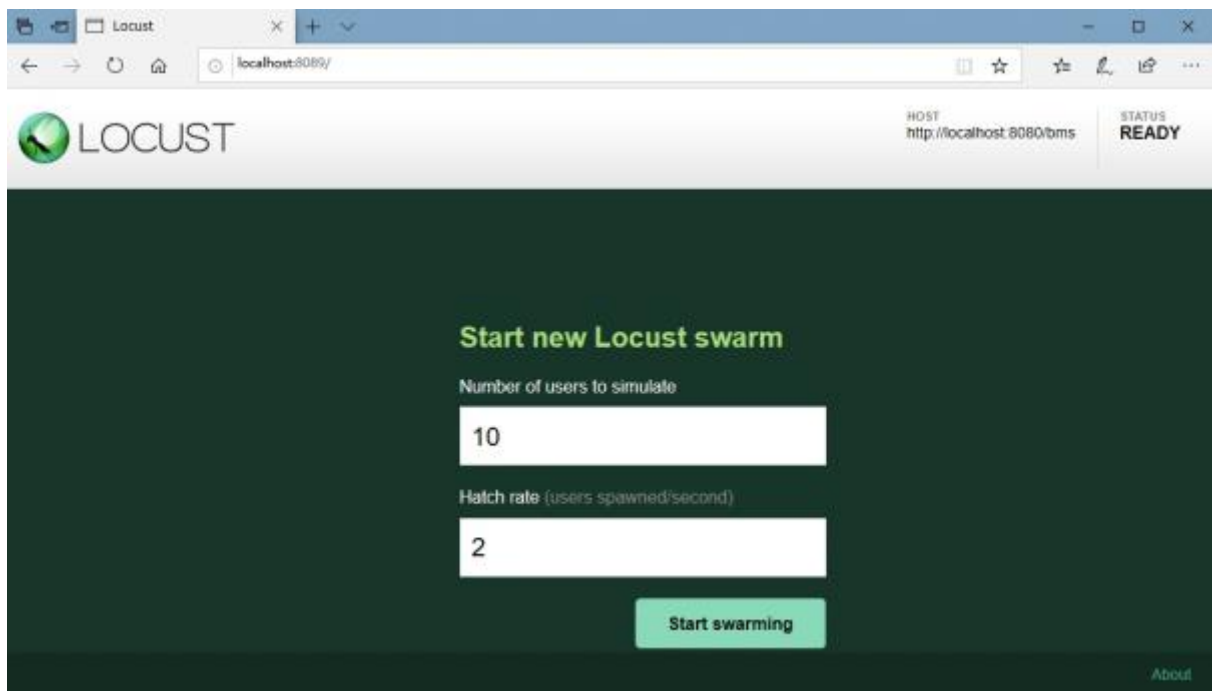
```
locust -f locust_files/my_locust_file.py --host=http://example.com
```

参数介绍：

- -f: 用来指定locust文件所在路径
- --host: 用来指定测试应用的网址

3.1 打开Locust的web界面

使用上面的命令行启动Locust之后，打开浏览器并访问：<http://localhost:8089> (如果你在本地运行Locust)。可以看到如下界面



参数说明：

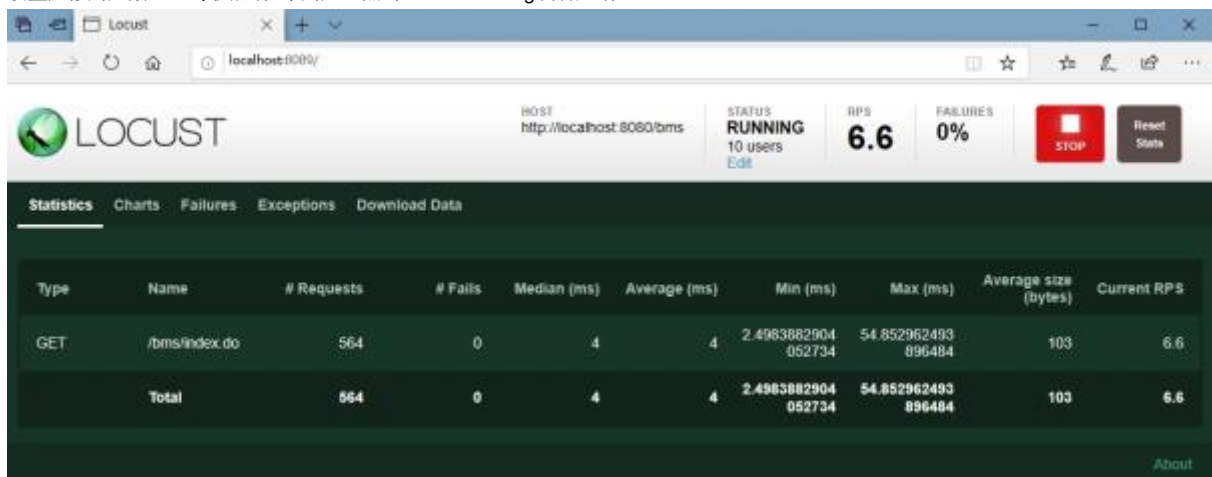
- Number of users to simulate: 要模拟的用户数量
- Hatch rate (users spawned/second): 孵化率(用户生成/秒)，即每秒启动虚拟用户数
- 点击Start swarming 开始运行性能测试

效果展示

103

程序员-软件测试

设置虚拟用户数10，每秒启动2个用户，点击Start swarming 开始运行

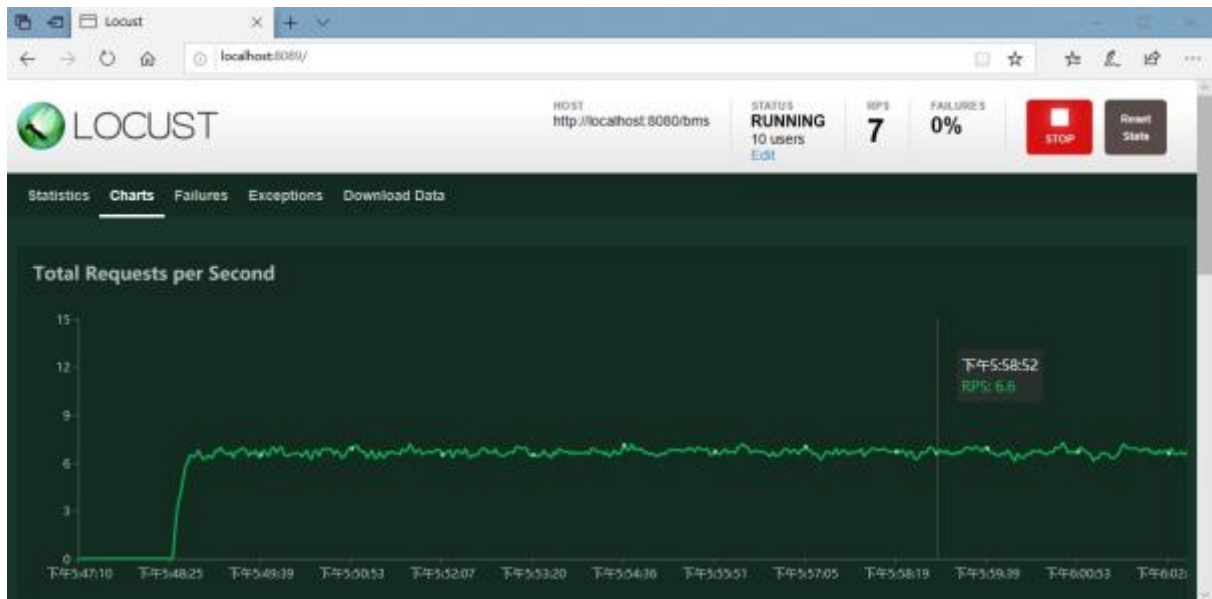


界面说明：

- Type: 请求类型
- Name: 请求路径
- Requests: 当前请求的数量
- Fails: 当前请求失败的数量

- Median (ms): 中间值, 单位毫秒, 一半服务器响应时间低于该值, 而另一半高于该值
- Average (ms): 所有请求的平均响应时间, 毫秒
- Min (ms): 请求的最小的服务器响应时间, 毫秒
- Max (ms): 请求的最大服务器响应时间, 毫秒
- Average size (bytes): 平均单个请求的大小, 单位字节
- Current RPS: 每秒钟请求的个数
- 点击Edit可以编辑请求用户数
- 点击STOP按钮可以停止测试
- 点击New test可以重新开始测试

Charts图表展示



三个图标分别是:

- Total Requests per Second: 每秒发送请求数
- Response Times(ms): 平均响应时间
- Number of Users: 虚拟用户数

程序员-软件测试