

性能测试 PTS

使用手册

使用手册

测试机

用户从已购买的ECS、RDS、SLB或非阿里云服务器中选取能够用于性能测试使用的服务器，添加至性能测试的测试环境里。

待测系统

部署在测试机上的应用系统。

监控机

从测试机中选取需要进行服务器资源监控的服务器。

事务

事务是性能测试脚本的一个重要特性，要度量服务器的性能需要定义事务；在性能测试中是通过方法来实现事务的，即将业务操作或者代码放在方法里面，通过框架将函数置为事务。

TPS

TPS(Transaction Per Second)每秒系统能够处理的交易或事务的数量，它是衡量系统处理能力的重要指标。

并发

并发分为狭义和广义两类。

- 狭义并发即所有的用户在同一时刻做同一件事情或操作，这种操作一般针对同一类型的业务，或者所有用户进行完全一样的操作，目的是测试数据库和程序对并发操作的处理。
- 广义并发即多个用户对系统发出了请求或者进行了操作，但是这些请求或操作可以是不同的，但对整个系统而言，仍然是有很多用户在同时进行操作。

狭义并发强调对系统的请求操作是完全相同的，多适用于负载测试、压力测试；广义并发不限制对系统的请求操作，多适用于混合场景、稳定性测试。

场景

性能测试过程中为了模拟真实用户的业务处理过程，在系统中构建的基于事务、脚本、虚拟用户、运行设置等一系列动作的集合称之为性能测试场景。性能测试中场景包含了脚本、施压模式、用户数、日志级别、步调时间等。

响应时间

响应时间是指从客户端发一个请求开始，到客户端接收到服务端返回的响应所经历的时间，响应时间由请求发送时间、网络传输时间和服务器处理时间三部分组成。在性能测试结果分析中，性能场景中事务的响应时间可以通过监控得到，事务响应时间分为事务最小响应时间、事务平均响应时间、事务最大响应时间。

并发用户数

模拟真实业务逻辑步骤的虚拟用户，虚拟用户模拟的操作步骤都被记录在虚拟用户脚本里，脚本用于描述用户在场景中执行的操作。

请求状态

请求状态反映了HTTP压测结果的HTTP状态码，状态码含义如下：

- 成功200：服务器已成功处理了请求并提供了请求的网页。
- 成功204：服务器成功处理了请求，但没有返回任何内容。
- 重定向3xx：需要客户端采取进一步的操作才能完成请求。
- 客户端错误4xx：表示请求可能出错，妨碍了服务器的处理。
- 服务器错误5xx：表示服务器在处理请求时发生内部错误，这些错误可能是服务器本身的错误而不是请求出错。

CPU

CPU资源是指性能测试场景运行的时间段内应用服务系统的CPU资源占用率，CPU资源是判断系统处理能力及应用运行是否稳定的重要参数。

Load

系统平均负载指在特定时间间隔内运行队列中的平均进程数。如果一个进程满足以下条件就会位于运行队列中：

- 它没有在等待I/O操作的结果。
- 它没有主动进入等待状态，也就是没有调用“wait”。
- 没有被停止，例如等待终止。

访问企业版控制台

操作指南

访问控制台

使用流程



一套完整的性能测试过程包括脚本创建、场景创建、场景执行和结果查看，按先后顺序执行。

访问控制台

- 脚本管理：提供简单易用的性能测试脚本模板。
- 场景管理：提供多种模拟真实场景的施压方式，提供丰富的性能指标监控和多种操控方式。
- 监控集：提供公网非阿里云监控机维护入口。
- 结果报表：提供完善的性能结果报表和强大的性能分析日志。

模块

- 脚本：脚本是执行性能测试的基础，脚本里包括需要压测的服务器地址、压测的url、压测的参数和压测的类型。
- 场景：场景需要绑定脚本来运行，一个场景可以绑定多个脚本，可以在场景中设置并发压测用户数、施压模式，场景开始执行后可以实时查看性能指标。
- 环境：可选择公网非阿里云服务器作为监控机。
- 结果：结果自动保存可随时查看。

访问企业版控制台

测试环境

返回企业版测试环境

添加监控服务器

- 1. 下载Radar性能监控工具，此监控工具分为Windows和Linux版本。

Radar下载地址：

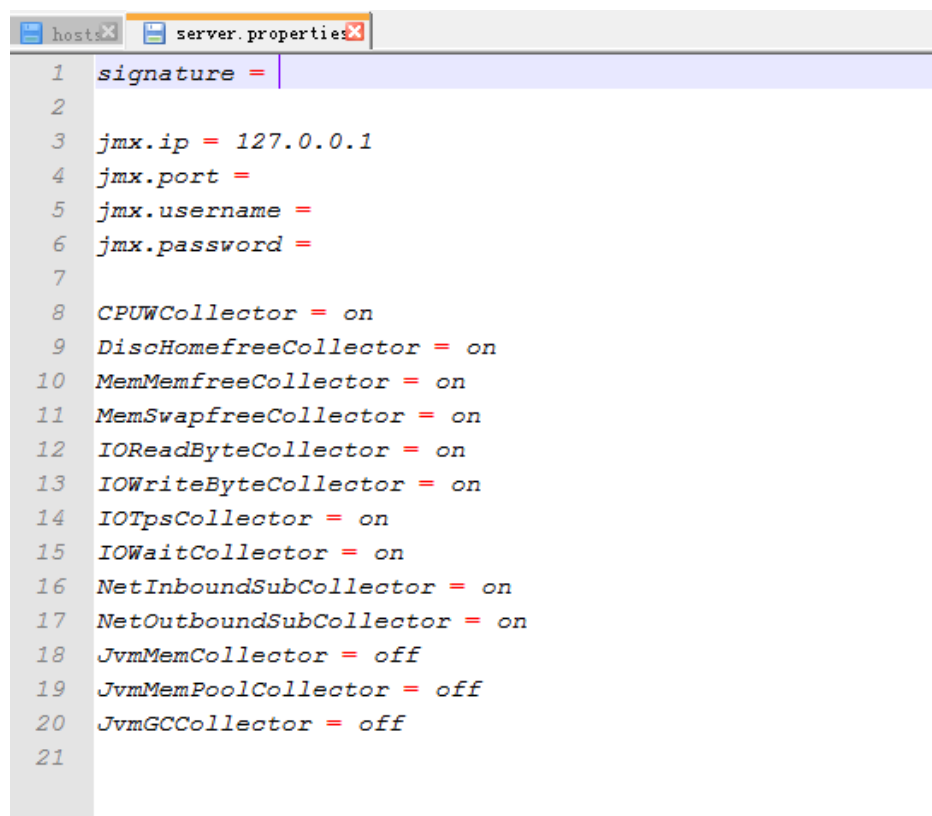
radar-for-Windows.zip

Windows安装Java教程：

Linux安装Java教程：

<http://bbs.aliyun.com/read/252573.html>

- 2.解压Radar工具下载包(Linux用户执行unzip radar-for-linux.zip命令进行解压)；
- 3.配置server.properties文件，修改signature字段等于性能测试用户设置中的用户标识；

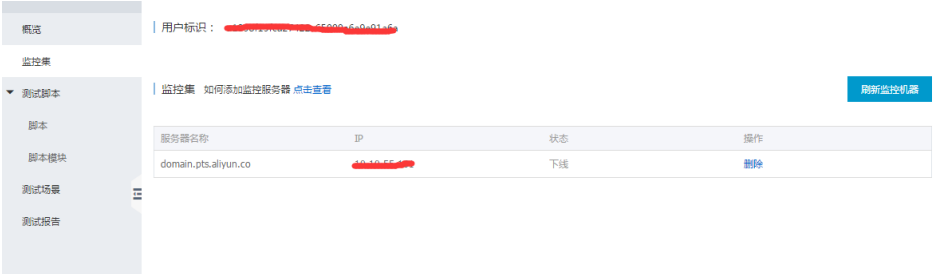


- 4.启动Radar ;

Windows版：直接点击监控工具目录下start.bat文件

Linux版：进入监控工具目录，运行chmod +x radar.sh赋予执行权限和./radar.sh start启动Radar（ 可通过agent.log日志，查看监控工具启动运行信息）

- 5.刷新监控集，可查看安装监控工具并成功启动后的监控服务器状态信息；



FAQ:

- 1. 添加监控服务器不成功？
- 2. 任务执行看不到监控指标？

其他问题咨询：在线帮助

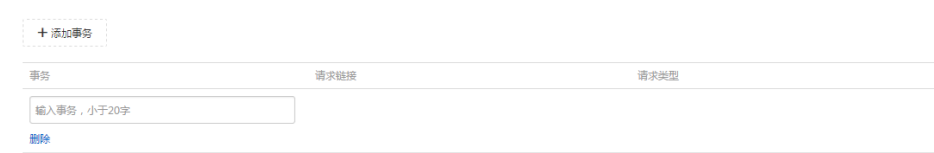
返回企业版测试环境

脚本编写

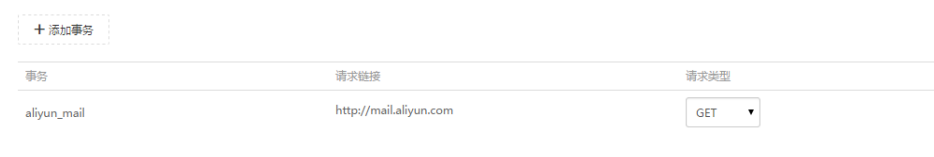
返回企业版测试脚本

添加事务

首先需创建事务，事务名可根据实际业务制定，如图所示：

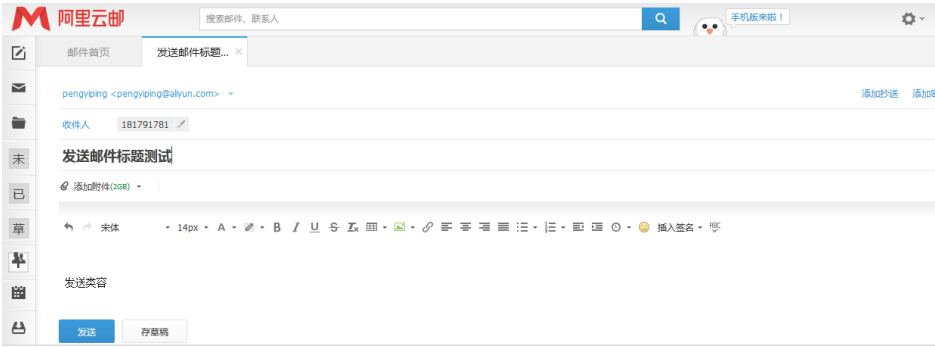


然后可添加待测系统链接，同时可选择Http请求类型（GET或POST），如图所示：

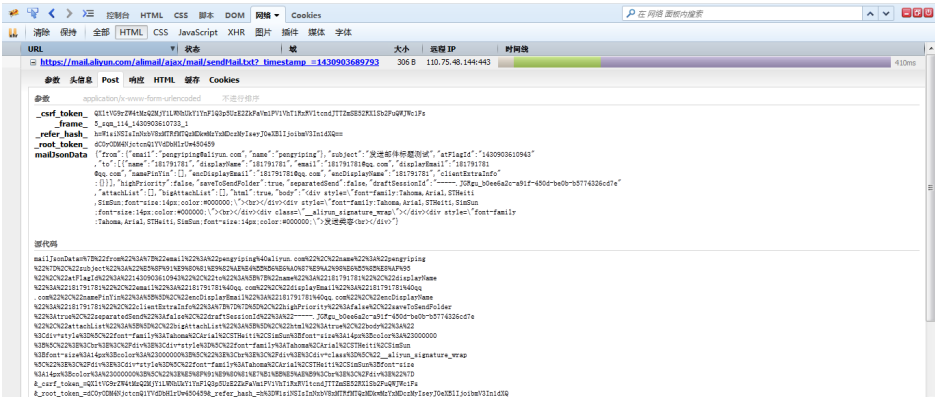


请求类型如选择的是GET方式，直接输入待测系统链接地址即可；如选择的是POST方式，需要在【请求链接】的【高级属性】编辑栏中Header项中添加Content Type请求头及Body项添加POST请求主体内容，步骤如下：

- 用抓包工具（例如Firebug）捕获待测系统的POST请求，如图所示：



- 查看POST请求主体内容，如图所示：



- 把POST请求主体内容添加到【高级属性】编辑栏的Body项（在Header项中POST请求默认添加Content Type请求头，值为application/x-www-form-urlencoded），如图所示：



高级属性

帮助链接

Header	Body	Cookie	检查点
<pre>mailJsonData=%7B%22from%22%3A%7B%22email%22%3A%22pengyiping%40aliyun.com%22%2C%22name%22%3A%22pengyiping%22%7D%22subject%22%3A%22%E5%8F%91%E9%80%81%E9%82%AE%E4%BB%B6%E6%A0%87%E9%A2%98%E6%B5%8B%E8%AF%95%22%2C%22atFlagId%22%3A%221430903610943%22%2C%22to%22%3A%5B%7B%22name%22%3A%22181791781%22%2C%22displayName%22%3A%22181791781%22%2C%22email%22%3A%22181791781%40qq.com%22%2C%22displayEmail%22%3A%22181791781%40qq.com%22%2C%22namePinYin%22%3A%5B%5D%22%2C%22encDisplayEmail%22%3A%22181791781%40qq.com%22%2C%22encDisplayName%22%3A%22181791781%40qq.com%22%2C%22clientExtraInfo%22%3A%7B%7D%7D%5D%2C%22highPriority</pre>			

确定

取消

如果使用ECS内网IP进行压测，施压机IP会临时写入用户ECS安全组，以确保内网环境的互联互通，压测完毕后施压机IP会从ECS安全组自动删除，同时请选择与测试机在同一地域的施压集群，跨地域无法进行内网压测。SLB无此限制。

返回企业版测试脚本

7

如果待测系统通过域名进行访问，可在此进行域名绑定，一个IP地址可绑定多个域名。
注意：IP地址只能输入环境管理中已添加的ECS、SLB或验证的公网服务器的IP，输入其他IP在压测时不会生效。

返回企业版测试脚本

返回企业版测试脚本

按模板编写脚本参数化

如果想让测试数据更丰富可对脚本进行参数化。参数化步骤如下：

- 上传参数文件，目前**仅支持csv格式**。请使用UE或Notepad工具（不要用Excel）进行编辑，每列参数化列值以英文逗号分隔；注意如果参数文件内容包含中文，参数文件请使用UTF-8无BOM格式编码。参数文件第一行为参数变量，可增加多个参数变量，首行不得超过255个字符；从第二行开始为参数值，可增加多个参数值，脚本运行时会读取对应参数变量的参数值按从上往下的顺序执行。可上传多个参数文件，上传完毕后参数文件默认为启用。每个参数文件大小不得超过60M。一个脚本最多10个参数文件，不得重名，重名会覆盖。

A	B	C	D
name	email	→ 参数变量	
test1	test1@taobao.com	→ 参数值	
test2	test2@taobao.com		

参数文件：

上传文件

参数文件说明（参数文件仅支持CSV格式，60M以内）

文件名	创建时间	循环使用	操作
demotest.csv	2015-01-30 13:57:27	已启用	取消循环 删除

注：参数文件循环启用，执行任务会循环使用该参数文件的参数值。参数文件循环不启用，执行任务只会运行一次该参数文件的参数值，然后任务停止。

- 在请求链接中选中要进行参数化的参数值，然后点击参数化按钮进行参数化。

事务	请求链接	请求类型
cloudtest 添加链接 上移 下移 删除	http://cloud.tmall.com/order/tryApply.htm?_tb_token_=nklAPB Bm5knV&action=tryApplyAction&event_submit_do_apply=1& productId=43&productName=%E6%80%A7%E8%83%BD% E6%B5%8B%E8%AF%95%E6%9C%8D%E5%8A%A1%28P TS%29&name=test1&email=test1@taobao.com&wangwang =&scene=&reason= 点击进行参数化	POST

- 在参数化窗口选择参数文件及参数变量，参数变量可通过系统自动读取参数文件获得也可手工输入。

参数格式化

替换内容

test1

参数文件

demotest.csv

参数变量

name

手动输入

确定

取消

- 系统自动完成替换，替换格式为 “%%_参数文件名：参数变量%%” 。

事务	请求链接	请求类型
cloudtest 添加链接 上移 下移 删除	http://cloud.tmall.com/order/tryApply.htm?_tb_token_=nklAPB Bm5knV&action=tryApplyAction&event_submit_do_apply=1& productId=43&productName=%E6%80%A7%E8%83%BD% E6%B5%8B%E8%AF%95%E6%9C%8D%E5%8A%A1%28P TS%29&name=%_demotest.csv:name%%&email=test1@t aobao.com&wangwang=&scene=&reason=	POST

- 一个参数文件或参数变量可用于多个不同链接里，一个链接也可对应多个不同的参数文件或参数变量。

按手工编写脚本参数化

- 上传参数文件，跟按模板编写一样。
- 导入如下类库：

```
from com.aliyun.pts import DsvReader
from com.aliyun.pts import ParamManager
```
- 加载参数化文件：

```
params = ParamManager.getInstance()
```

```
params.addProvider(DsvReader(u' query-user1.csv' ))
```

- 参数化文件指针向下移一行，注意只需要调用一次：

```
params.nextRecord(u' query-user1.csv' )
```
- 获取参数化值name = params.getParamValue(u' query-user1.csv:userpassword')

代码示例：

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# PTS Script Version 1.0
# Trunner auto-generated test script at Tue Jul 12 09:53:55 CST 2016

from java.lang import String
from java.util import Random
from java.util import Date

# PTS脚本SDK：框架API、常用HTTP请求/响应处理API
from util import PTS
from HTTPClient import NVPair
from HTTPClient import Cookie
from HTTPClient import HTTPRequest
from HTTPClient import CookieModule
from HTTPClient import ShutdownException

# PTS对参数化的相关支持
from com.aliyun.pts import DsvReader
from com.aliyun.pts import ParamManager

# 脚本初始化段，可以设置压测引擎的常用HTTP属性
PTS.HttpUtilities.setUrlEncoding('utf-8')
PTS.HttpUtilities.setFollowRedirects(True)
PTS.HttpUtilities.setTimeout(120000)
# PTS.HttpUtilities.setKeepAlive(False)
# PTS.HttpUtilities.setUseCookieModule(False)
# PTS.HttpUtilities.setProxyServer('localhost', 8888)
# PTS.Context.setParamDirectory("/Users/fei/Work/trunner/data")

# 支持socket测试, 如TCP\UDP等协议
# import socket

# 设置系统编码
import sys
reload(sys)
sys.setdefaultencoding('utf-8')

params = ParamManager.getInstance()

expiryDate = Date()
expiryDate.year += 10

params.addProvider(DsvReader(u'FileName.csv'))
class TestRunner:
```

```
# TestRunner对象的初始化方法，每个线程在创建TestRunner后执行一次该方法
def __init__(self):
    self.threadContext = PTS.Context.getThreadContext()

# 主体压测方法，每个线程在测试生命周期内会循环调用该方法
def __call__(self):
    PTS.Data.delayReports = 1
    params.nextRecord(u'FileName.csv')

    PTS.Logger.info("username=" + params.getParamValue(u'FileName.csv:Field') + "")

    # 请在此处调用事务方法
    #statusCode = self.action_name()
    #PTS.Framework.setExtraData(statusCode)
    PTS.Data.report()
    PTS.Data.delayReports = 0

# 执行事务的方法，会在__call__主体压测方法中被调用到
#def action_name(self):
# 请输入方法内容

# 调用施压引擎施压。第一个参数是事务名，可以为中文；第二个参数是执行事务方法的方法名；第三个统一写TestRunner
#PTS.Framework.instrumentMethod(u'tran_name', 'action_name', TestRunner)
```

手工编写脚本参数化技巧

在需要参数化的字符串中用 “%%_参数文件名：参数变量%%” 进行替换，保存脚本（记住要保存脚本）后，会自动转换成可运行的脚本。例如将参数化用户名的地方进行日志输出：PTS.Logger.info(‘username=%%_UserNameList.csv:username%%’),保存脚本后，会将按照手工编写脚本参数化的步骤自动进行转换。

返回企业版测试脚本

返回企业版测试脚本

高级属性

- 编辑脚本时可使用高级属性来设置脚本请求的header、body及cookie信息，支持参数化，参数文件上传后需手工输入%%_FileName.csv:ColumnName%%进行替换，与URL参数化后的效果类似。



- 当请求类型为get时可设置 header、cookie：

高级属性 ×

Header

Cookie

添加

Header名称	Header值	操作
----------	---------	----

确定

取消

- 当请求类型为post时可设置header、body及cookie：

高级属性 ×

Header

Body

Cookie

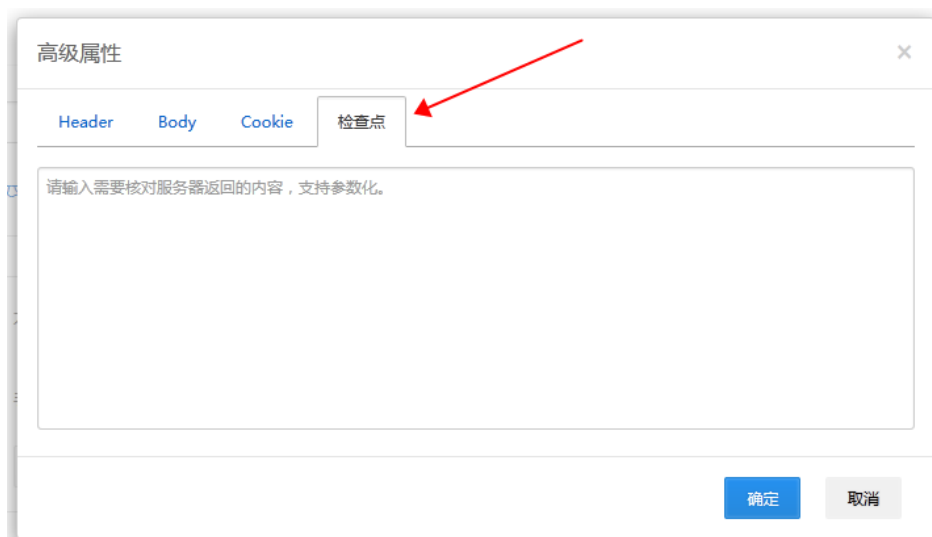
添加

Header名称	Header值	操作
----------	---------	----

确定

取消

此外，如果想自定义校验信息来验证服务端的返回内容是否正确，可通过设置检查点来完成。注意，只有当返回码为200时才会进一步校验检查值，如果返回码非200直接算失败，更丰富的使用可通过手工编写脚本来进行。



检查点内容目前只支持字符串，不支持正则表达式，支持参数化。

样例

1、json校验

服务端返回内容：`{ "code" : " success" , " result" :123}`

检查点设置：`"code" : " success"`，只要返回结果中包含此字符串就会认为请求成功，反之认为请求失败。

2、html校验

服务端返回内容：

`<html>`

`<head>`

`</head>`

`<body>`

`<div id=" code" >success</div>`

`<div id=" result" >123</div>`

`</body>`

`</html>`

检查点设置：`<div id=" code" >success</div>`，只要返回结果中包含此字符串就会认为请求成功，反之认为请求失败。

3、参数化

如果要参数化校验上方案例中result的返回值，检查点设置：`"result":%%_xxx.csv:result%%`

返回企业版测试脚本

返回企业版测试脚本

快速启动

脚本编写完成后可通过脚本列表上的快速启动运行脚本，系统会自动创建脚本对应的场景与任务。执行时长默认1分钟，并发用户数上限2000。

施压集群地域可选择，如想通过私网进行压测请选择与测试机在同一地域的施压集群并确保设置的待测系统IP为ECS内网地址或SLB私网地址，如想体验公网压测可任意选择施压集群并确保设置的待测系统IP为公网地址。**温馨提示：**如果使用公网压测会对ECS、SLB产生公网流量，ECS、SLB会按照现有的公网流量计费策略进行计费。

快速启动

(请通过配置场景和任务来获取更多的压力模型和监控指标)

×

执行时间：

1分钟

并发用户数：

施压集群：

ⓘ 您本次任务将会消耗0VUM，预估价格0元， 预估剩余免费额度0VUM

确定

取消

返回企业版测试脚本

如果测试场景必须要先登录，可选择启用“是否登录”功能，启用后每次执行时会先进行登录动作并只执行一次，然后再进行事务里的其他动作。另外，登录动作也可以直接写在事务里，这样的话每次运行事务都会执行一次登录动作。

是否登录：

是

否

+ 添加登录链接

请求链接	请求类型	操作
http://	GET	上移 下移 删除 高级属性

返回企业版测试脚本

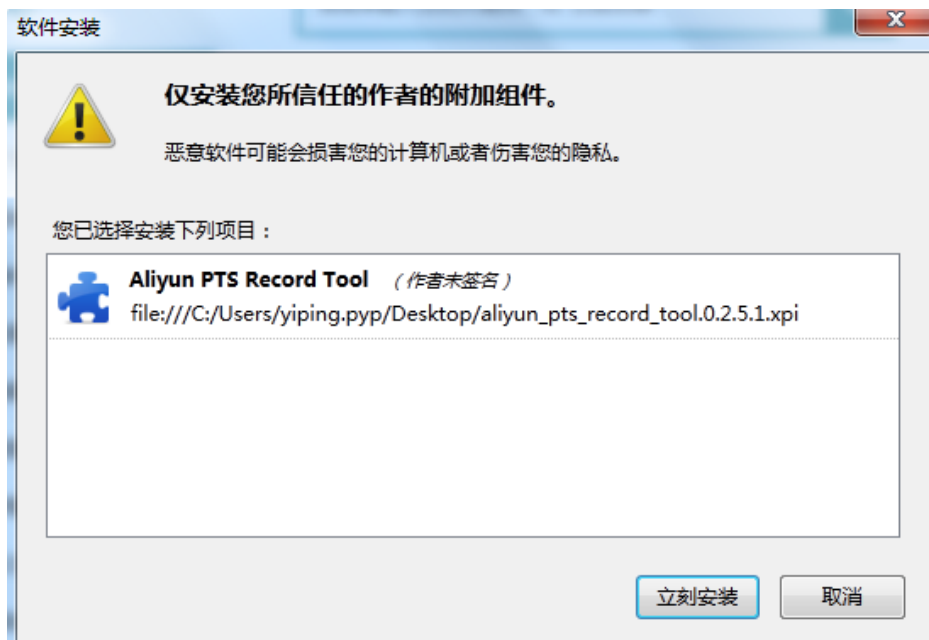
返回企业版测试脚本

录制工具安装说明

性能测试基于浏览器插件提供Chrome和Firefox（支持37及以下版本）两个版本的录制工具，工具下载后的安装步骤如下。

Firefox：（支持37及以下版本）

- 1、下载文件拖动至Firefox浏览器窗口，在提示框里选中组件点击安装。

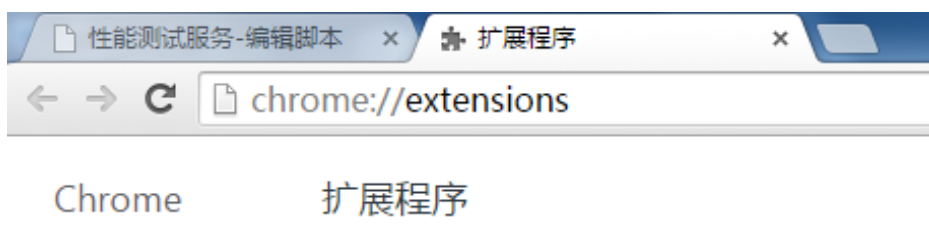


- 2、安装完成后浏览器右上角会出现性能测试Logo。



Chrome：

- 1、URL框内输入chrome://extensions/，进入扩展程序安装页面。



- 2、下载文件拖动至Chrome浏览器窗口，浏览器弹出确认新增扩展程序框，点击添加安装。



- 3、安装完成后浏览器右上角会出现性能测试Logo。



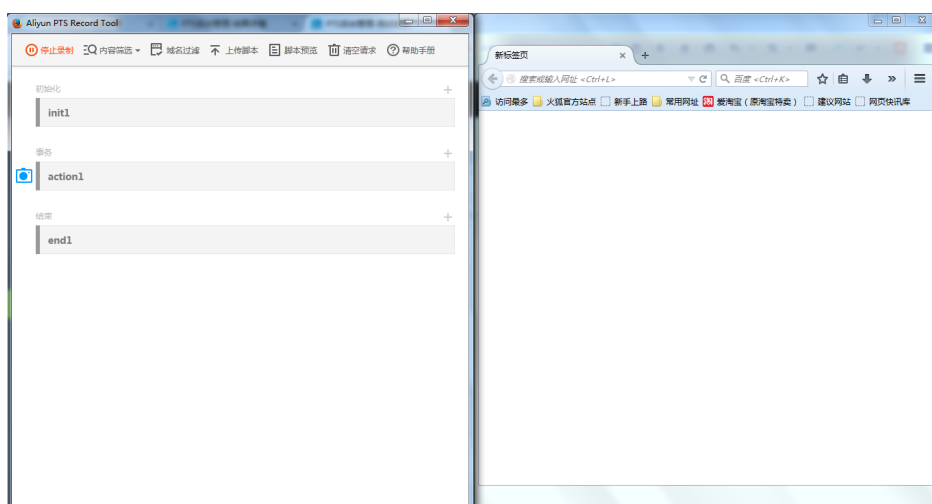
返回企业版测试脚本

返回企业版测试脚本

录制工具

对于复杂的业务，例如登陆、考试、订购、购买、发帖、回帖、退出等业务，由于捕获请求内容或者手工编写脚本工作量稍大，可以使用性能测试基于Firefox（**支持37及以下版本**）和Chrome浏览器插件录制工具。通过使用此工具，用户在被测系统中进行手工操作业务，录制工具会将用户的操作行为进行录制，录制完成以后，自动生成脚本，根据业务规则可能稍微修改一下脚本，就可以运行脚本了，录制下来的脚本模拟了用户真实的操作行为，极大地方便用户的使用。

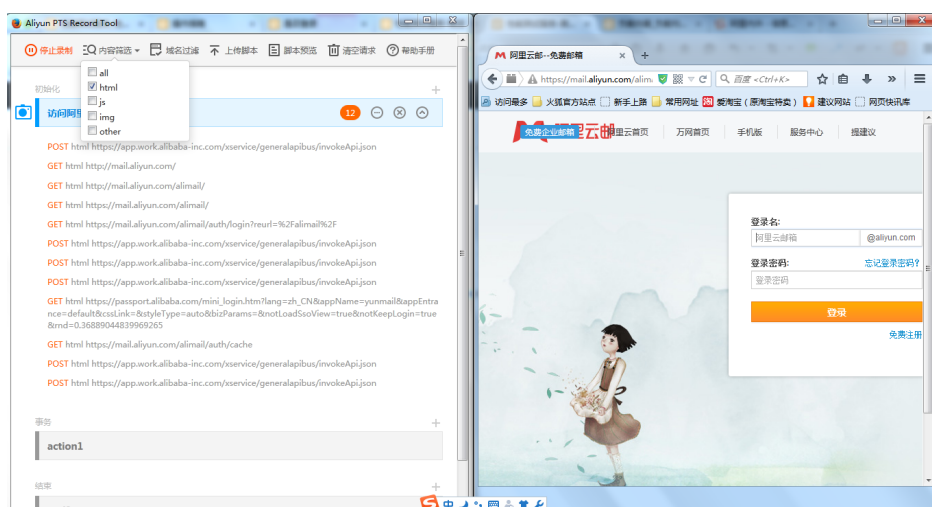
- 工具安装后打开浏览器点击右上角性能测试录制工具Logo,弹出录制工具框和浏览器



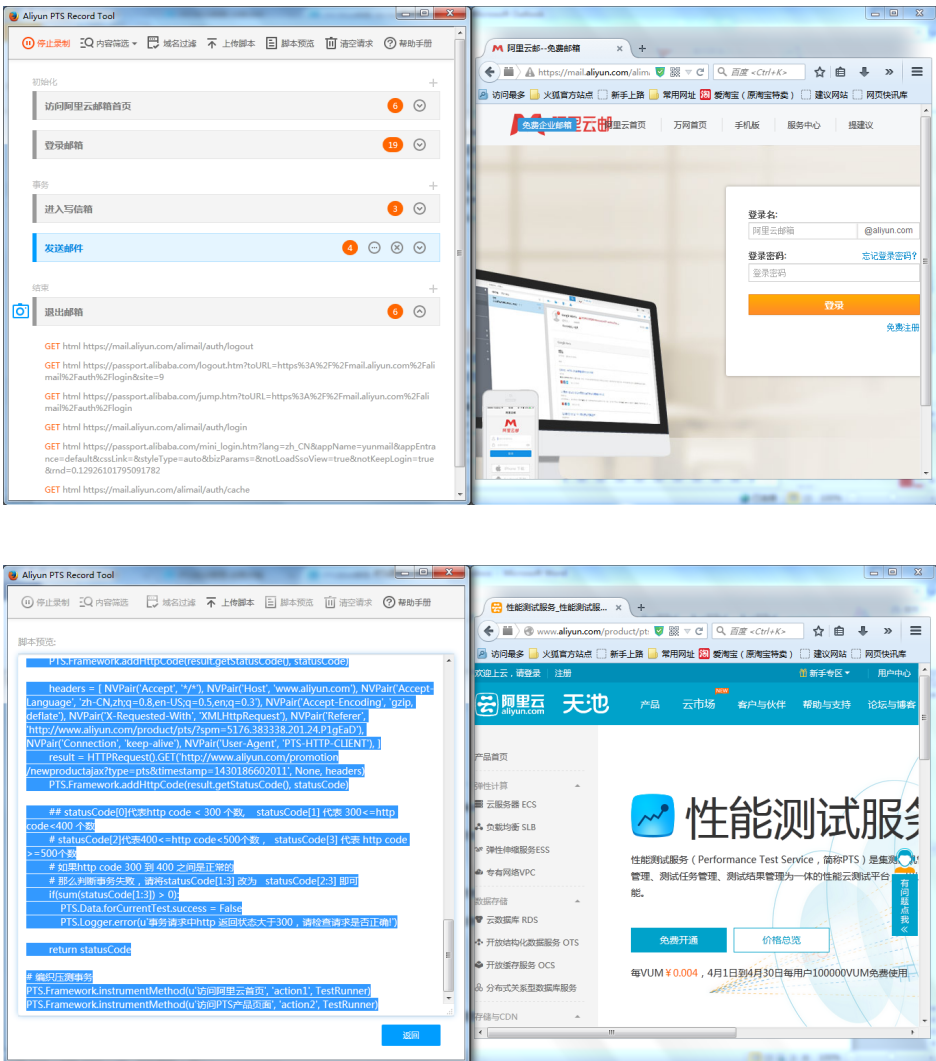
- 定义事务名，在浏览器输入URL进行访问操作，录制工具会自动记录访问操作过程中的HTTP请求



- 录制工具默认只显示HTML类型HTTP的录制请求，如需显示其他类型请求请点击内容过滤选择需要显示的类型请求。



- 录制完成后点击停止录制按钮，如果需要预览录制生成的脚本，请点击脚本预览按钮。



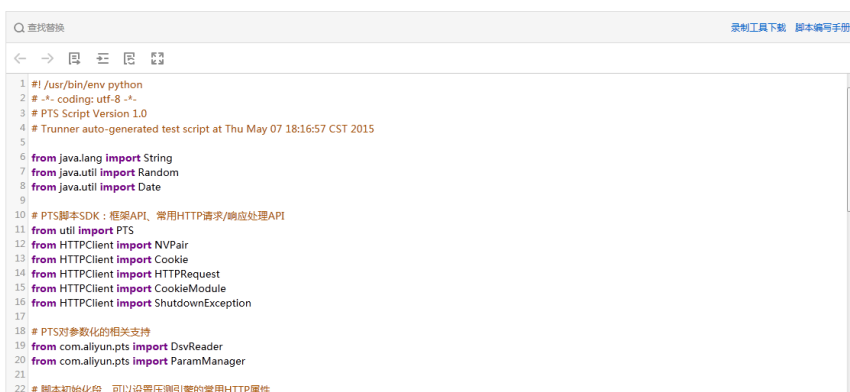
- 预览脚本确认没问题后就可以点击上传脚本按钮保存脚本到性能测试云端，完成脚本录制过程。

返回企业版测试脚本

返回企业版测试脚本

手工脚本

对于非HTTP/HTTPS协议的业务（如：TCP协议等）或者某些内容不能录制(如:加密算法，非浏览器访问的应用等)，需要进行手工编写代码开发脚本。

编写方式：  模板编写  手工编写A screenshot of the PTS script editor interface. It features a search bar at the top left, navigation icons, and a toolbar with '录制工具下载' and '脚本编写手册'. The main area displays a Python script with line numbers 1 through 22. The script includes comments in Chinese and Python code for imports and initialization. The code is color-coded: comments are orange, and code is black. The script starts with a shebang and encoding declaration, followed by version and generation information. It then imports various modules from the PTS SDK, including util, HTTPClient, and com.aliyun.pts. The script ends with a comment about the initialization phase.

```
1 #!/usr/bin/env python
2 #-*- coding: utf-8 -*-
3 # PTS Script Version 1.0
4 # Trunner auto-generated test script at Thu May 07 18:16:57 CST 2015
5
6 from java.lang import String
7 from java.util import Random
8 from java.util import Date
9
10 # PTS脚本SDK：框架API、常用HTTP请求/响应处理API
11 from util import PTS
12 from HTTPClient import NVPair
13 from HTTPClient import Cookie
14 from HTTPClient import HTTPRequest
15 from HTTPClient import CookieModule
16 from HTTPClient import ShutdownException
17
18 # PTS对参数化的相关支持
19 from com.aliyun.pts import DsvReader
20 from com.aliyun.pts import ParamManager
21
22 # 脚本初始化段，可以设置压测引擎的常用HTTP属性
```

手工脚本语言为Jython，目前支持的Jython版本为2.5.3。Jython是Python的Java语言实现，它使用Python的语法和类库，运行在JVM中，和同一个JVM中的Java类可以实现无缝互操作，因此使用Jython作为脚本语言可以最大程度的利用Python的简洁、高效，同时保留对Java语言的全面兼容。

- Jython语法

Jython程序由一系列语句组成，语句组成了代码块，代码块组成了方法、函数，然后再通过类把数据、方法和函数封装起来。和其它高级语言一样，Jython的语句也是由一些最基本的词（token）组成。Token可以是标识符（identifiers）、关键字（keywords）、字面值（literals）、操作符（operators）和分割符（delimiters），这些token通过Jython的语言执行器进行词法分析产生，而词法分析器通过字符方式读入Jython脚本文件。

- 脚本框架

性能测试的性能测试脚本是一个TestRunner类，这个类会被每一个并发线程初始化。测试进程首先加载脚本并执行脚本中顶格的语句，同时定义TestRunner这个测试类。然后每个线程会实例化一个TestRunner类，调用类中的init方法一次，继而循环调用TestRunner类的call方法。最后线程结束时，会调用类中的del方法。init和del方法都是可选的，只有call方法是必需的。

```
#第一部分：执行器声明和脚本编码声明
#!/usr/bin/env python
#-*- coding: utf-8 -*-
#第二部分：Jython类库、Java类库和自定义类的导入
# PTS Script Version 1.0
# PTS脚本SDK：框架API、常用HTTP请求/响应处理API
from util import PTS
from HTTPClient import NVPair
from HTTPClient import Cookie
from HTTPClient import HTTPRequest
from HTTPClient import CookieModule

#第三部分：测试进程级别的脚本语句和初始化
# 脚本初始化段，可以设置压测引擎的常用HTTP属性
#PTS.HttpUtilities.setKeepAlive(False)
#PTS.HttpUtilities.setUrlEncoding('GBK')
#PTS.HttpUtilities.setFollowRedirects(False)
#PTS.HttpUtilities.setUseCookieModule(False)
PTS.HttpUtilities.setUseContentEncoding(True)
PTS.HttpUtilities.setUseTransferEncoding(True)
```

```
#第四部分：TestRunner测试类
# 脚本执行单元类，每个VU/压测线程会创建一个TestRunner实例对象
class TestRunner:
    # TestRunner对象的初始化方法，每个线程在创建TestRunner后执行一次该方法
    def __init__(self):
        self.threadContext = PTS.Context.getThreadContext()
        self.init1()
        self.init_cookies = CookieModule.listAllCookies(self.threadContext)
    # 主体压测方法，每个线程在测试生命周期内会循环调用该方法
    def __call__(self):
        PTS.Data.delayReports = 1
        for c in self.init_cookies:
            CookieModule.addCookie(c, self.threadContext)
        statusCode = self.action1()
        PTS.Framework.setExtraData(statusCode)
        statusCode = self.action2()
        PTS.Framework.setExtraData(statusCode)
        PTS.Data.report()
        PTS.Data.delayReports = 0
    # TestRunner销毁方法，每个线程循环执行完成后执行一次该方法
    def __del__(self):
        for c in self.init_cookies:
            CookieModule.addCookie(c, self.threadContext)
        self.end1()
    # 定义请求函数
    def init1(self):
        .....
    def action2(self):
        .....
def action2(self):
    .....

#第五部分：instrumentMethod语句
# 编织压测事务
PTS.Framework.instrumentMethod(u'action1', 'action1', TestRunner)
PTS.Framework.instrumentMethod(u'action2', 'action2', TestRunner)
```

返回企业版测试脚本

返回企业版测试脚本

脚本调试

脚本开发完成后，可回放脚本测试虚拟用户脚本的基本业务功能是否成功，通过回放请求快照及执行日志信息查看请求成功或失败后需要解决的错误和问题。

返回企业版测试脚本

返回企业版测试场景

保存

- 21

测系统IP为ECS内网地址或SLB私网地址，如想体验公网压测可任意选择施压集群并确保设置的待测系统IP为公网地址。温馨提示：如果使用公网压测会对ECS、SLB产生公网流量，ECS、SLB会按照现有的公网流量计费策略进行计费。

场景操作

- 执行：立即启动场景，生成一个可执行任务。
- 停止：立即停止运行中的场景。
- 增减并发用户数：场景启动后实时增加并发用户数。
- 实时监控：实时监控性能指标。
- 删除：只能删除未启动的场景。
- 取消排期：取消定时场景。
- 查看日志：进入日志分析页面。

返回企业版测试场景

一次性加载所有并发用户。

施压模式：☒ 常规模式 ☐ 梯度模式 ☐ 目标模式

* 并发用户数：

返回企业版测试场景

分批增加或减少用户。递增用于控制任务启动时用户增加的频率，递减用于任务停止时用户减少的频率。如果只输入初始用户数、最大用户数没有启用递增、递减，则施压效果与常规模式一致。

施压模式：☐ 常规模式 ☒ 梯度模式 ☐ 目标模式

* 初始用户数：

* 最大用户数：

是否递增：

☐ 否 ☒ 是

每隔：

时

分

秒

增加用户：

是否递减：

☐ 否 ☒ 是

每隔：

时

分

秒

减少用户：

返回企业版测试场景

达到指定目标阈值后停止压测。例如下图配置运行时的效果是：事务“Login”的响应时间达到100ms或服务器CPU占用率达到80%后停止压测。

施压模式：☐ 常规模式 ☐ 梯度模式 ☒ 目标模式

* 事务名：

Login

目标类型：

RT

* 目标值：

100

ms

CPU限制阈值：

80

%

* 初始用户数：

1

(压力控制设置)

* 最大用户数：

100

(压力控制设置)

* 每隔：

10

秒 (压力控制设置)

* 增加用户：

1

(压力控制设置)

返回企业版测试场景

步调时间即场景运行频率控制时间，一旦设置，场景每次运行后根据所设步调时间及压测请求响应时间综合判断是否需要停顿，如需停顿则在停顿时间到达后再启动下一次运行，注意非思考时间。

如想控制压测请求的发送频率可设置步调时间，一旦设置在指定的单位时间内只会发送一次压测请求。例如，场景所选脚本中包含一个或多个事务，场景每运行一次响应时间为0.5秒，如果希望每秒只发送两次请求则步调时间为0或空，如果希望每秒只发送一次请求则步调时间为1秒。

如果步调时间设置为1秒而响应时间大于1秒，则步调时间不会生效，按响应时间发送压测请求。

返回企业版测试场景

测试结果

测试任务执行完毕后系统会自动生成测试结果报告，由概览、业务指标、ECS指标、RDS指标、监控集指标五部分构成，同时提供查看日志入口。

返回企业版测试结果

返回企业版测试结果

查看日志

日志信息主要由压测日志、压测步骤两部分组成，压测日志显示压测进程的日志信息，压测步骤显示压测过程信息。

查看日志

返回测试结果列表

任务名：

执行时间：2014-10-28 10:00:14 到 2014-10-28 10:00:59

压测日志

场景名	脚本名	状态	压测日志
		成功	实时查看 日志分析

压测步骤

时间	步骤详情	操作状态
2014-10-28 10:00:14	任务启动中	成功
2014-10-28 10:00:20	压测集群启动压测	成功
2014-10-28 10:00:59	任务停止中	成功
2014-10-28 10:00:59	压测集群停止压测	成功

其中压测日志又分为实时查看、日志分析两部分。任务一旦执行可在实时查看中查阅最新的压测日志，默认保存前后1000行。任务运行5分钟后可在日志分析中查看更完整的压测日志，并能根据条件进行日志筛选。注意，日志内容与场景设置中的日志级别保持一致，所有日志默认保存七天。

日志查看：

服务端IP：10.175.199.28，查询行数：【后100】，任务ID：【200129111】，场景ID：【20012549】，日志名称：【worker.log】，查询日志结果：
2015-10-20 10:52:10,931 200129111445308624 INFO IZ23zq88js0Z-43419-1346 thread-0 [run-365555, test-20012087]: http://hangzhou.com/pts/SelectRds.jsp?count=3000 -> 200 OK, 162 bytes
2015-10-20 10:52:10,933 200129111445308624 INFO IZ23zq88js0Z-43419-1346 thread-0 [run-365557, test-20012087]: http://hangzhou.com/pts/SelectRds.jsp?count=10 -> 200 OK, 159 bytes
2015-10-20 10:52:10,935 200129111445308624 INFO IZ23zq88js0Z-43419-1346 thread-0 [run-365557, test-20012087]: http://hangzhou.com/pts/SelectRds.jsp?count=30 -> 200 OK, 160 bytes
2015-10-20 10:52:10,937 200129111445308624 INFO IZ23zq88js0Z-43419-1346 thread-0 [run-365558, test-20012087]: http://hangzhou.com/pts/SelectRds.jsp?count=50 -> 200 OK, 160 bytes
2015-10-20 10:52:10,940 200129111445308624 INFO IZ23zq88js0Z-43419-1346 thread-0 [run-365559, test-20012087]: http://hangzhou.com/pts/SelectRds.jsp?count=100 -> 200 OK, 159 bytes
2015-10-20 10:52:10,942 200129111445308624 INFO IZ23zq88js0Z-43419-1346 thread-0 [run-365560, test-20012087]: http://hangzhou.com/pts/SelectRds.jsp?count=1000 -> 200 OK, 161 bytes
2015-10-20 10:52:10,945 200129111445308624 INFO IZ23zq88js0Z-43419-1346 thread-0 [run-365561, test-20012087]: http://hangzhou.com/pts/SelectRds.jsp?count=3000 -> 200 OK, 162 bytes
2015-10-20 10:52:10,947 200129111445308624 INFO IZ23zq88js0Z-43419-1346 thread-0 [run-365562, test-20012087]: http://hangzhou.com/pts/SelectRds.jsp?count=10 -> 200 OK, 159 bytes
2015-10-20 10:52:10,950 200129111445308624 INFO IZ23zq88js0Z-43419-1346 thread-0 [run-365563, test-20012087]: http://hangzhou.com/pts/SelectRds.jsp?count=30 -> 200 OK, 159 bytes
2015-10-20 10:52:10,951 200129111445308624 INFO IZ23zq88js0Z-43419-1346 thread-0 [run-365564, test-20012087]: http://hangzhou.com/pts/SelectRds.jsp?count=50 -> 200 OK, 160 bytes
2015-10-20 10:52:10,954 200129111445308624 INFO IZ23zq88js0Z-43419-1346 thread-0 [run-365565, test-20012087]: http://hangzhou.com/pts/SelectRds.jsp?count=100 -> 200 OK, 160 bytes
2015-10-20 10:52:10,957 200129111445308624 INFO IZ23zq88js0Z-43419-1346 thread-0 [run-365566, test-20012087]: http://hangzhou.com/pts/SelectRds.jsp?

全部日志类型 请输入关键字 日志时间范围： 2015-10-20 10:37:04 - 2015-10-20 10:52:12 搜索

时间	内容
2015-10-20 10:52:11,158	content : shut down level : INFO thread : thread-0
2015-10-20 10:52:11,154	content : http://hangzhou.com/pts/SelectRds.jsp?count=10 -> 200 OK, 159 bytes level : INFO thread : thread-0
2015-10-20 10:52:11,152	content : http://hangzhou.com/pts/SelectRds.jsp?count=3000 -> 200 OK, 161 bytes level : INFO thread : thread-0
2015-10-20 10:52:11,151	content : http://hangzhou.com/pts/SelectRds.jsp?count=1000 -> 200 OK, 161 bytes level : INFO thread : thread-0
2015-10-20 10:52:11,148	content : http://hangzhou.com/pts/SelectRds.jsp?count=100 -> 200 OK, 160 bytes level : INFO thread : thread-0
2015-10-20 10:52:11,146	content : http://hangzhou.com/pts/SelectRds.jsp?count=50 -> 200 OK, 160 bytes level : INFO thread : thread-0
2015-10-20 10:52:11,144	content : http://hangzhou.com/pts/SelectRds.jsp?count=30 -> 200 OK, 160 bytes level : INFO thread : thread-0
2015-10-20 10:52:11,141	content : http://hangzhou.com/pts/SelectRds.jsp?count=10 -> 200 OK, 159 bytes level : INFO thread : thread-0

共365654条,每页显示8条 < 1 2 3 >

返回企业版测试结果

1.Alibaba Cloud Performace Testing 支持RAM简介

阿里云性能测试接入RAM可以支持子账户操作使用，子账户创建的脚本、场景、任务等都属于父账户，子账户必须通过父账户在RAM控制台授权以后才能使用。

2.操作步骤

- a.父账户登录阿里云控制台(<https://ak-console.aliyun.com/#/accesskey>) 创建固定AccessKey和AccessSecret,已创建请忽略,此操作必须。
- b.父账户登录RAM控制台(<https://ram.console.aliyun.com/#/overview>)，创建授权策略:
- b1: 性能测试服务级别授权策略，系统有默认的性能测试访问策略，名称叫做：AliyunPTSFullAccess，此操作可以忽略。

b2: 获取SLB/ECS/RDS资源授权策略，系统有默认的只读访问策略，名称叫做：
AliyunSLBReadOnlyAccess, AliyunECSReadOnlyAccess, AliyunRDSReadOnlyAccess, 此操作可以忽略。

b3: 创建群组，分别创建群组性能测试服务级别群组、ECS只读群组、SLB只读群组、RDS只读群组。分别将对应的权限服务级别访问和资源访问加入到上述群组中，有的子账户可能已经被设置了资源所有权限群组或者子账户不需要获取阿里云机器资源功能，那么ECS/SLB/RDS只读群组可以省略，但是性能测试服务级别群组必须创建。

b4: 创建子账户，将子账户加入到上述群组中去，此操作必须，然后通过子账户登录RAM控制台就可以使用性能测试服务。

c. 授权策略可以通过自定义策略来控制子账户性能测试使用，例如: IP/SSL/MFA/Time限制，如下策略定义限制某段IP才能访问PTS，一般情况下这些策略很少使用，具体可以参照RAM帮助文档：

https://help.aliyun.com/product/8315075_ram.html

IP限制：

```
{
  "Statement": [
    {
      "Action": "pts:*",
      "Effect": "Allow",
      "Resource": "*",
      "Condition": {
        "IpAddress": {
          "acs:SourceIp": "42.0.9.45/90"
        }
      },
      "Bool": {
        "acs:SecureTransport": "false"
      }
    }
  ],
  "Version": "1"
}
```

访问企业版控制台首页

并发用户

阿里云性能测试单个场景默认提供最高10000并发的施压能力（5台施压实例 * 2000/台），如果用户的单个场景需要使用更大的并发，需要提前与阿里云客户经理或者技术支持提出申请。

施压集群

阿里云性能测试提供华东1(杭州)、华北2(北京)、华北1(青岛)和华南1(深圳)四个地域的施压机器。

参数文件

阿里云性能测试单个脚本的参数文件最多支持10个。

结果日志

性能测试的测试结果日志分析最长保存7天。

金融云用户

阿里云性能测试已对金融云用户全面开放，由于金融云与公有云存在安全隔离，所以金融云应用仅支持外网性能测试。

[访问性能测试控制台](#)

性能测试不仅只提供性能测试工具，还有在线的阿里性能测试专家为您解答。性能测试过程中，难免会遇到各种各样的问题，性能测试相关的、性能测试产品相关的问题都欢迎您来咨询。在线咨询的方式有三种，按优先级排序如下：

1、旺旺群名称：性能测试技术支持，群号：1473075831

工作日，旺旺群会得到最及时的回复，阿里性能测试专家等候您的光临。

2、阿里云工单系统：进入工单系统提交工单

阿里云专业的售后工程师团队为您提供7*24小时技术服务

3、性能测试论坛：进入阿里云性能测试论坛

您可以通过论坛了解产品动态，学习性能测试知识，分享性能测试心得，也可以通过论坛来提问。

[访问性能测试控制台](#)

脚本录制调试指南

[返回企业版测试脚本](#)

工具简述

使用性能测试进行测试的时候，针对简单URL请求，很多用户都可以直接按照模板来进行编写，相当简单；但

对于某些稍微复杂的业务，例如登陆、考试、订购、购买、发帖、回帖、退出等业务，由于捕获请求内容或者手工编写脚本工作量稍大，给用户带来不便，在此背景下，性能测试项目组开发了基于Firefox（支持37及以下版本）和Chrome浏览器插件录制工具。

通过使用此工具，用户在被测系统中进行手工操作业务，录制工具会将用户的操作行为进行录制，录制完成以后，自动生成脚本，根据业务规则可能稍微修改一下脚本，就可以运行脚本了，录制下来的脚本模拟了用户真实的操作行为，极大地方便用户的使用。

使用指南

工具下载与安装

一、Chrome录制工具下载与安装

1.chrome录制工具下载

下载地址：chrome_pts_record_tool

或登录阿里云进入性能测试页面 —> 点击“测试脚本” —> 点击“新增脚本” —> 编写方式选择“手工编写” —> 点击“录制工具下载” —> 点击下载。

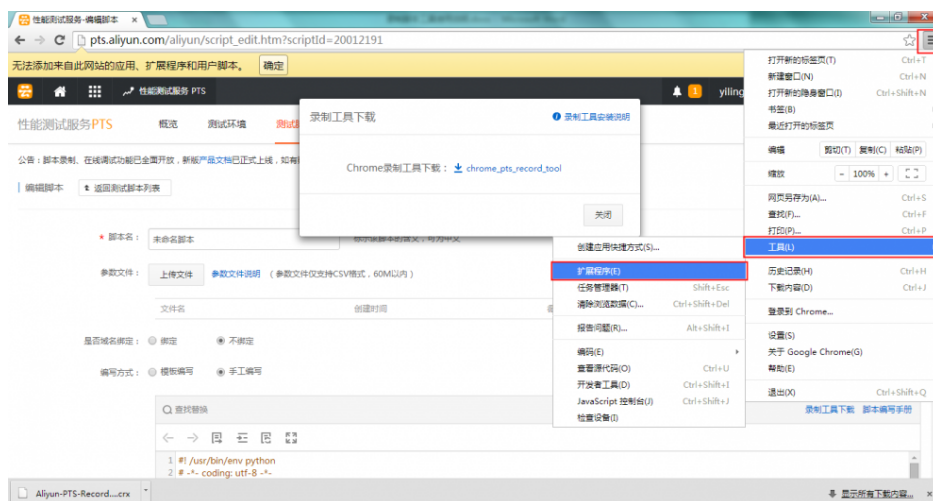
如下图展示：



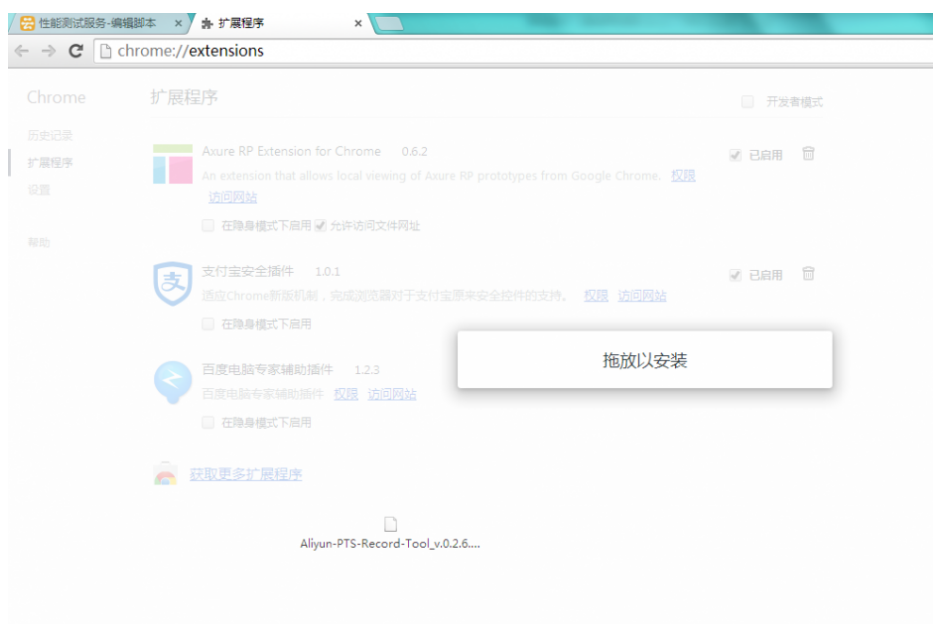
2.点击继续，会提示如下信息，需手动安装：



3.首先打开Chrome扩展程序：



4.将下载的文件：Aliyun-PTS-Record-Tool_v0.2.6.3.crx直接拖动到扩展程序中：



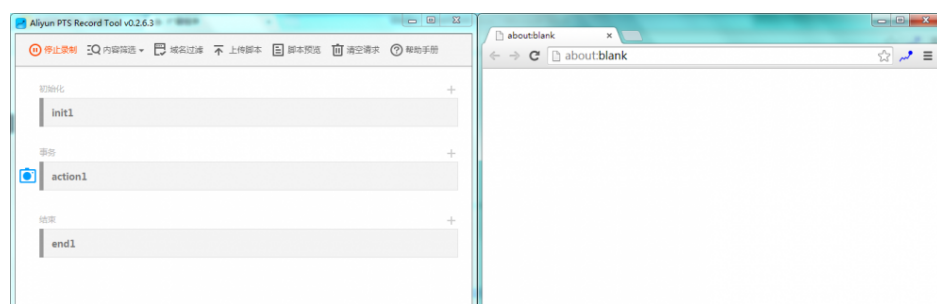
5.提示信息确认是否添加，点击“添加”：



6.完成安装后，扩展程序中显示安装的性能测试录制工具，同时浏览器右上角会显示性能测试录制工具Logo：



7.点击性能测试录制工具Logo，即可打开工具，进行脚本录制操作，安装完成！



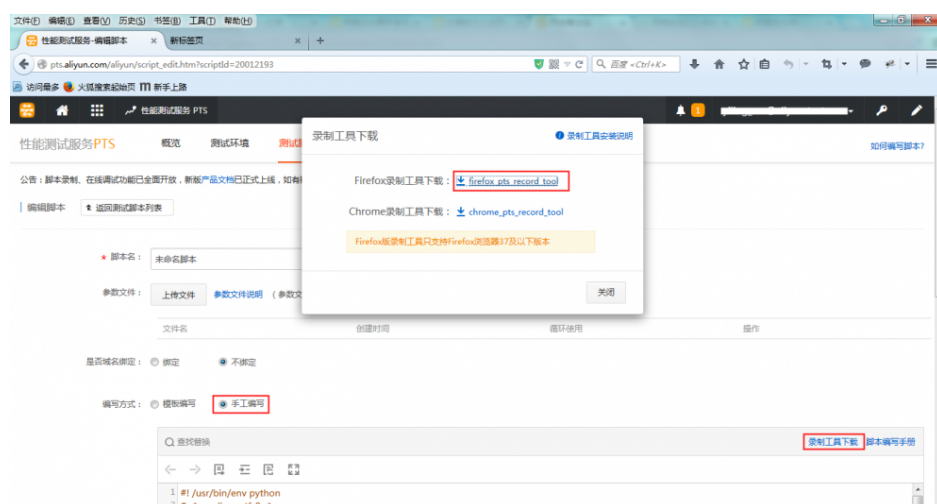
二、Firefox录制工具下载与安装

1.Firefox录制工具下载

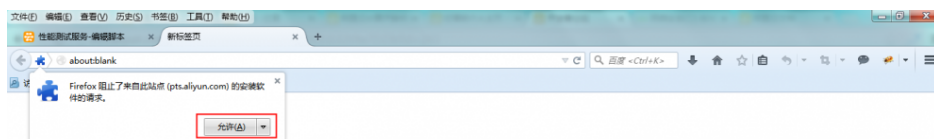
下载地址：firefox_pts_record_tool

或使用Firefox浏览器登录阿里云进入性能测试页面 —> 点击“测试脚本” —> 点击“新增脚本” —> 编写方式选择“手工编写” —> 点击“录制工具下载” —> 点击下载。

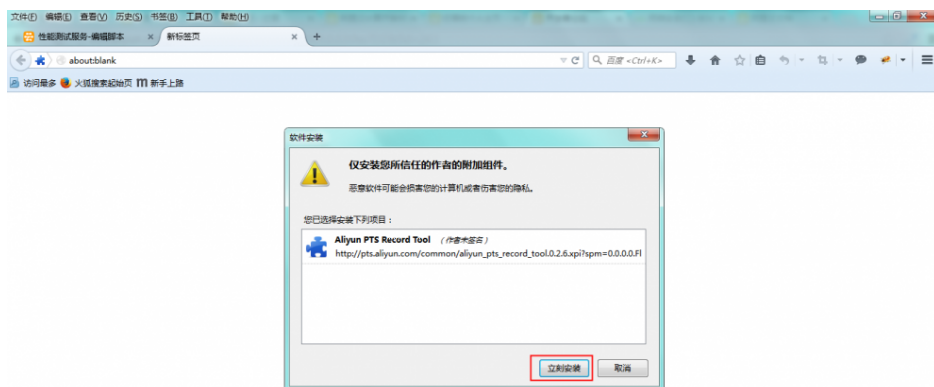
如下图所示：



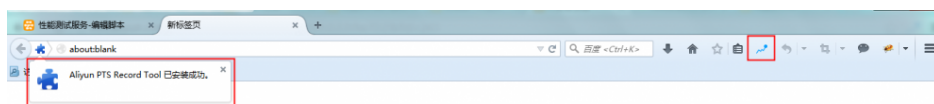
2.打开新的浏览器标签页，提示是否允许下载安装，点击“允许”：



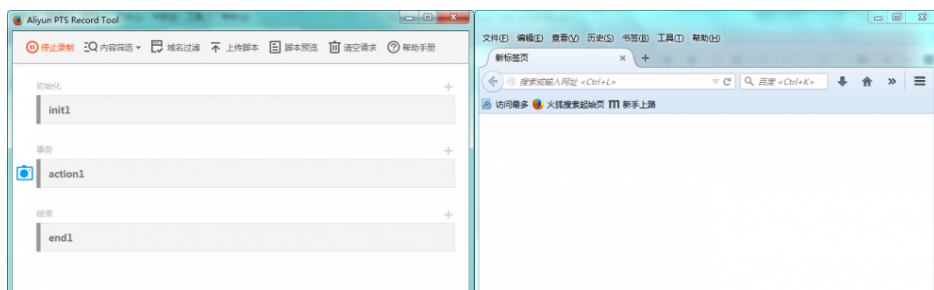
3.下载完成后，提示用户是否安装，点击立刻安装”：



4.安装成功后，会显示提示信息，同时浏览器右上角会有性能测试录制工具Logo：



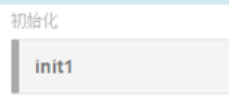
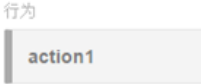
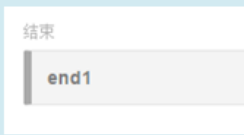
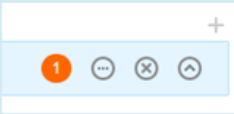
5.点击性能测试录制工具Logo，即可打开工具，进行脚本录制操作，安装完成！



功能界面介绍

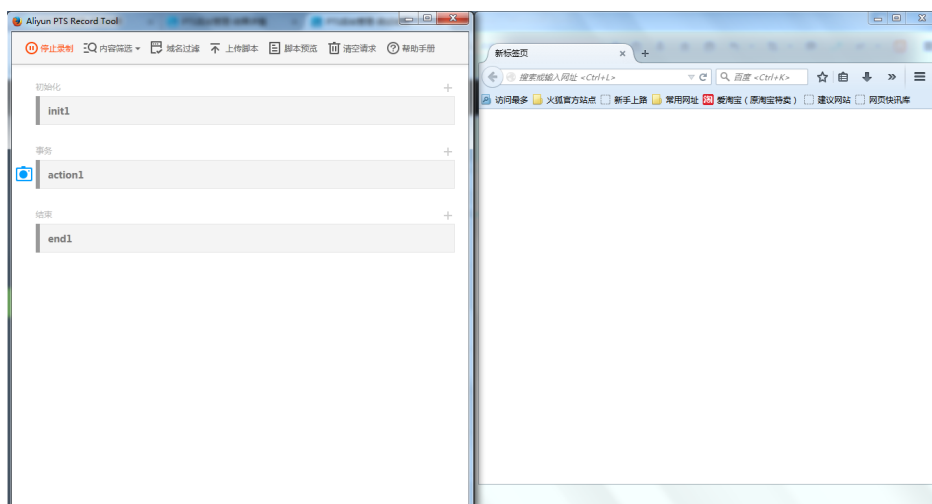


编号	图标或操作	用途	使用场景
1		开始录制按钮	在浏览器点击 PTS Logo 图标打开录制工具后会弹出录制工具框和浏览器框,默认已经开启录制
2		停止录制按钮	如果想暂停和停止录制可以点击停止录制按钮,录制工具会停止记录 HTTP 协议请求
3		内容筛选下拉框	如果用户想对已经录制的请求进行筛选,可以选择下拉框请求类型进行筛选 默认选择 HTML 类型
4		域名过滤按钮	脚本录制完成后,可以点击域名过滤按钮对录制的请求按域名进行过滤设置
5		上传脚本按钮	脚本录制完成后,可以点击上传脚本按钮,把脚本保存到云端

6	 脚本预览	脚本预览按钮	脚本录制完成后, 可以点击脚本预览查看录制生成的脚本代码
7	 清空请求	清空请求按钮	清空录制的所有 HTTP 请求
8	 帮助手册	帮助手册按钮	如需查看帮助手册可点击
9		虚拟用户初始化	虚拟用户初始化指的是每个虚拟用户启动时, 会运行初始化操作, 无论任务执行多长时间, 此初始化操作只运行一次, 一般情况下可以将类似登录操作、初始化连接操作等放在此初始化里面。
10		虚拟用户行为	虚拟用户行为在测试任务中会一直不断的循环运行, 直到运行任务截止时间才结束, 因此在 10 分钟之内, 根据响应时间和相关设置, 将会运行很多次, 行为中可包含定义多个事务, 例如查询、购买等。
11		虚拟用户结束	虚拟用户结束指的是在测试任务结束的时候, 每个虚拟用户执行此操作, 只执行一次, 比如登录注销、关闭连接等操作
12		录制到当前事务标识	表示当前录制请求在 action1 事务, 如果想在 action2 进行录制时, 需用鼠标双击 action2 或单击 action2 最左边粗线条。注意单击 action2 只表示选中此 action2, 进行相关的操作, 非在当前 action2 进行录制
13		事务的功能和编辑	 当前事务包含请求数  点击可展开编辑功能, 详见编号 14  删除当前事务  展开和折叠当前事务的请求  添加新的事务
14		相关操作	重命名: 重新命名 init、Action、end 名字 新增: 新增新的 init、Action、end 清空请求: 清空此 init、Action、end 下所有录制的请求

录制使用

- 步骤1：打开浏览器点击右上角性能测试录制工具Logo,弹出录制工具框和浏览器，截图如下：

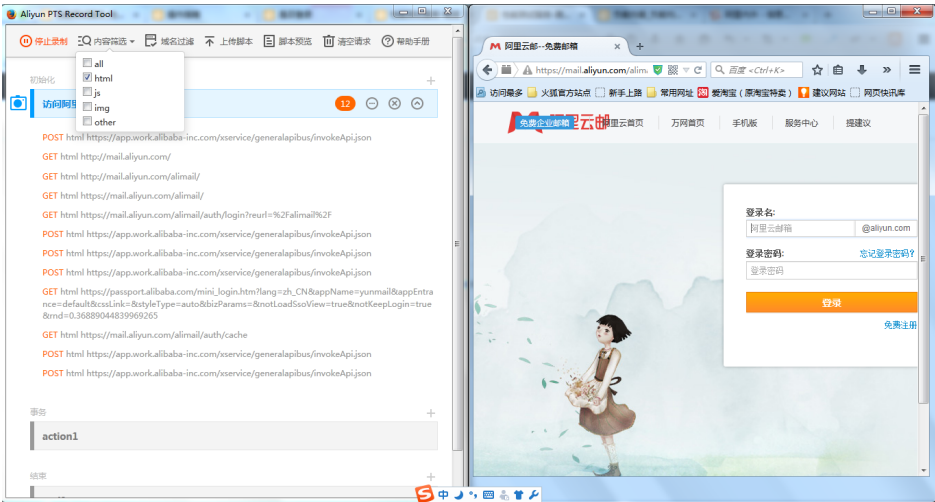


备注:录制工具打开后默认启用开始录制。

- 步骤2：定义事务名，在浏览器输入URL进行访问操作，录制工具会自动记录访问操作过程中的HTTP请求。



- 步骤3：录制工具默认只显示HTML类型HTTP的录制请求，如需显示其他类型请求请点击内容过滤选择需要显示的类型请求。



- 步骤4：录制完成后点击停止录制按钮，如果需要预览录制生成的脚本，请点击脚本预览按钮，



- 步骤5：预览脚本确认没问题后就可以点击上传脚本按钮保存脚本到性能测试云端，完成脚本录制过程。

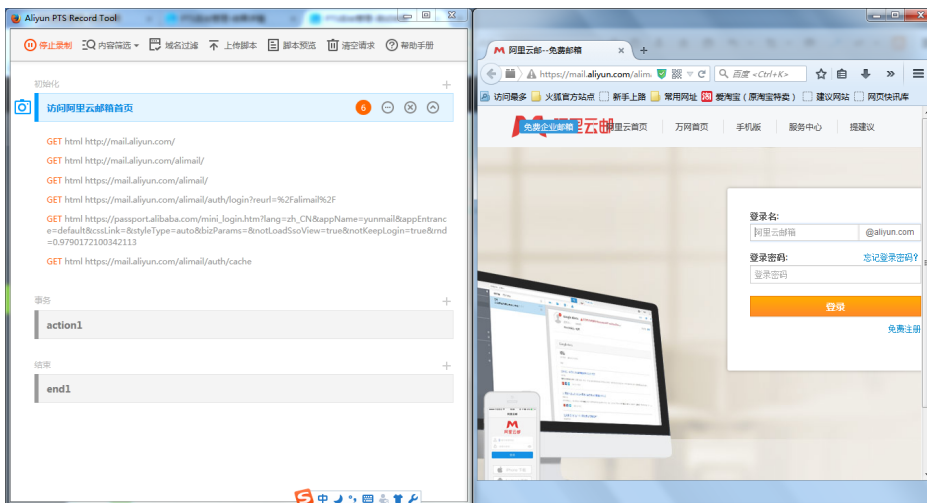
阿里云邮箱发送邮件脚本示例

业务调研

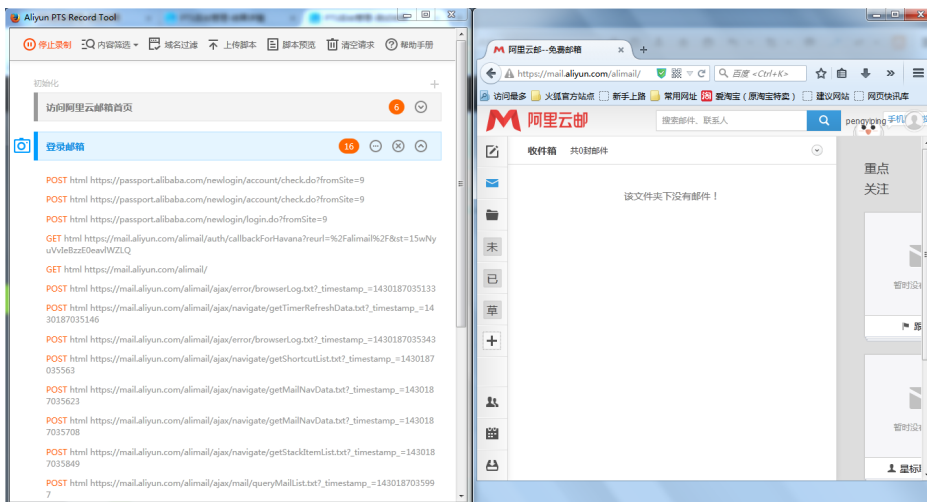
假设针对阿里云邮箱系统进行压测，邮箱最主要的业务是发邮件，发邮件流程是登陆邮箱，发送一封又一封的邮件，而在业务高峰的时候，可能会有很多人在给不同的人发邮件，邮件主题和内容都可能不一样，发完邮件后，退出邮箱系统，采用性能测试录制工具将此过程录制下来。

录制业务流程请求

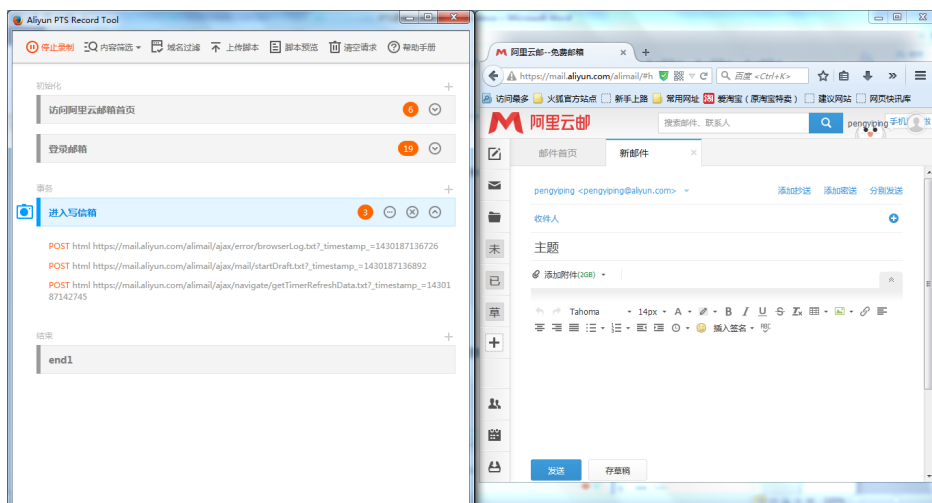
- 步骤1：访问阿里云邮箱



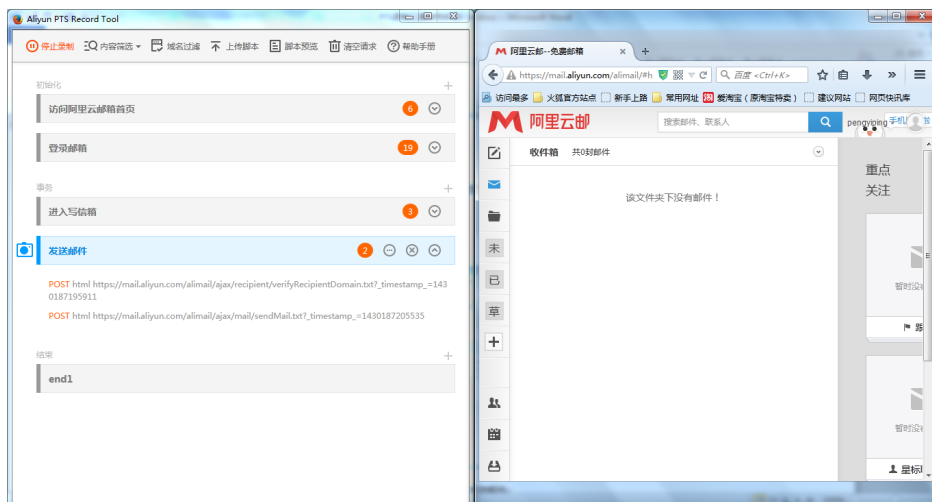
- 步骤2：登陆邮箱



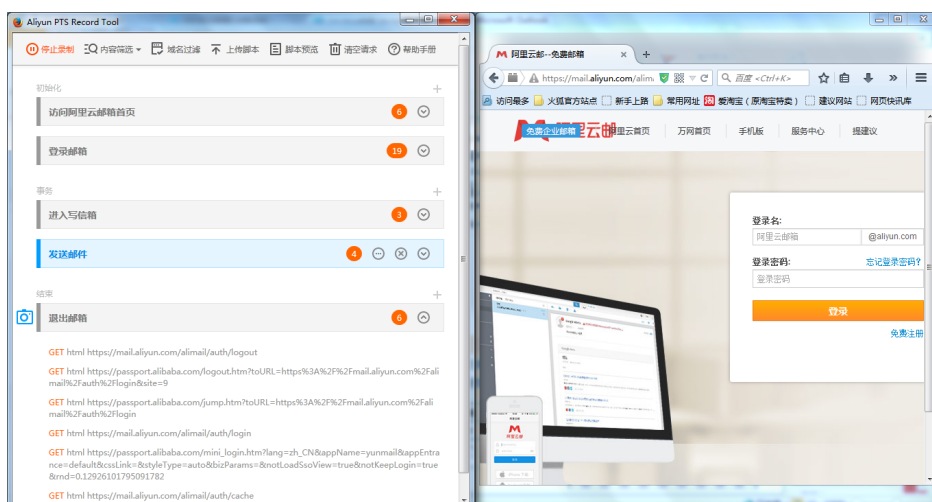
- 步骤3：进入写邮件页面



- 步骤4：发送邮件



- 步骤5：退出邮箱



- 步骤6：保存录制脚本到性能测试云端

添加参数化

参数化定义：将脚本中录制下来的特定值用一个变量来代替，该变量的值是可变的。例如收件人邮箱、标题内容等。

参数化作用：主要是模拟用户真实访问系统和避免一些系统约束导致虚拟用户运行失败。

参数化策略：

- 是否需要唯一性约束
- 是否需要当前时间
- 是否需要从缓存中获得或模拟缓存策略
- 是否受其他字段约束

如果测试脚本需要发送邮件给不同的收件人，并且不同的收件人邮箱标题和内容不一样。这种情况下就需要我们对录制下来的收件人、邮箱标题、邮箱内容进行参数化操作：

- 步骤1：准备参数化文件--CSV格式，用UE或Notepad工具（不要用Excel）进行编辑，每列参数化列值以逗号分隔；注意如果参数化文件内容包含中文，参数化文件请使用UTF-8无BOM格式编码保存。

sendmail_addr	mail_title	mail_cont
18112345@aliyun.com	标题1	内容1
18112346@aliyun.com	标题2	内容2
18112347@aliyun.com	标题3	内容3

- 步骤2：上传参数化文件mail.csv到性能测试脚本编辑页面

步骤3：在脚本添加导入相关类，并加载参数化文件

```
from com.aliyun.pts import DsvReader
from com.aliyun.pts import ParamManager
params = ParamManager.getInstance()
params.addProvider(DsvReader(u"mail.csv"))
```

步骤4：在脚本中找到需要参数化的URL请求，进行参数化 需要参数化内容：
: %5B%22181791781%40qq.com

```
headers = [ NVPair('Accept', '*/*'), NVPair('Host', 'mail.aliyun.com'), NVPair('Accept-Languag
result = HTTPRequest().POST('https://mail.aliyun.com/alimail/ajax/recipient/verifyRecipientDon
'''emails=%5B%22181791781%40qq.com%22%5D&_csrf_token_=QXltVG9rZW4t
```

提取参数化文件的参数值：

```
Import urllib
Sendmail_addr=urllib.quote(params.getParamValue(u'mail.csv:sendmail_addr').encode('utf-8'))
```

备注：如果参数化内容需要URL转码，需要使用urllib.quote方法转码 替换需要参数的内容：

```
[ NVPair('Accept', '/*/*'), NVPair('Host', 'mail.aliyun.com'), NVPair('Accept-Language',
TTPRequest().POST('https://mail.aliyun.com/alimail/ajax/recipient/verifyRecipientDomain.
'''emails=''' + Sendmail_addr + '''%22%5D&_csrf_token_=QXltVG9rZW4tODE5MzE
ork.addHttpCode(result.getStatusCode(), statusCode)
```

添加文本检查点

文本检查点是通过对HTTP请求响应的文本内容进行检查是否包含预先设置的文本内容；用户可以通过设置文本检查点来判断事物是否成功。发送邮件的HTTP请求成功后响应文本内容包含了 “success” 关键字，通过文本检查点来判断发送邮件事务是否成功。

```
headers = [ NVPair('Accept', '/*/*'), NVPair('Host', 'mail.aliyun.com'), NVPair('Accept-Language',
result = HTTPRequest().POST('https://mail.aliyun.com/alimail/ajax/recipient/verifyRecipientDomain
'''emails=''' + Sendmail_addr + '''%22%5D&_csrf_token_=QXltVG9rZW4tODE5MzE
PTS.Framework.addHttpCode(result.getStatusCode(), statusCode)
if(not PTS.HttpUtilities.checkResponse(200, 'success')):|
    PTS.Data.forCurrentTest.success = False
```

```
#当响应返回码为200，且返回body内容包含success则返回True
PTS.HttpUtilities.checkResponse(200, " success ")
```

添加关联

关联定义：在录制脚本时，录制工具会拦截客户端和服务端之间的会话生成脚本，生成的脚本都是一成不变的。但是业务流程中的HTTP请求中存在一些上下文依赖的动态数据，比如说向服务器发送一个login请求，并发给服务器用户名和口令的数据，服务器端验证数据正确后发送给客户端一个sessionID，不同的客户端登录会返回不同的sessionID，客户端下一次请求的时候就会发送上次请求返回的sessionID给服务器端进行验证合法性。关联就是把脚本中某些固定数据转变成获取自服务器发送的、动态的、每次都一样的数据。

关联作用：主要是模拟更真实用户访问系统的行为保证业务的逻辑性和准确性、防止服务器校验导致脚本运行失败。

关联策略：

在脚本中请求中对动态值进行参数化

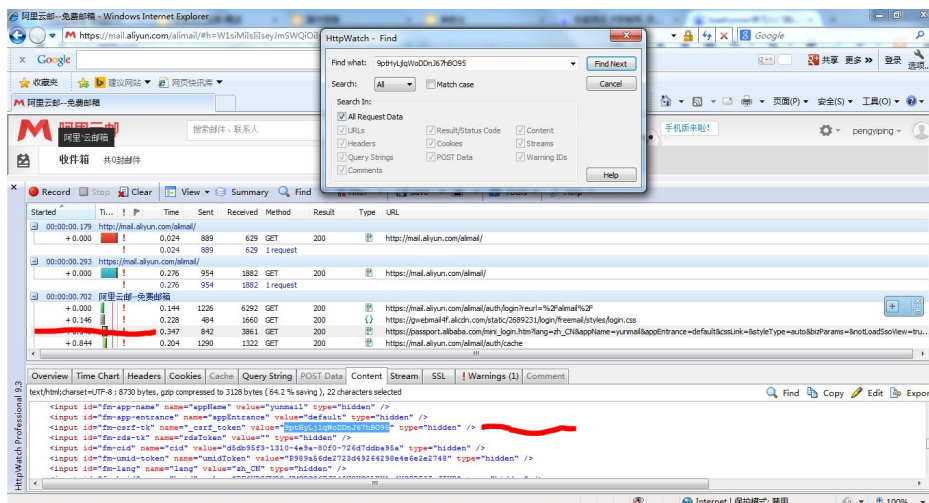
- 决定要捕捉的动态值（通过两个脚本比较、开发建议、脚本经验）
- 创建一个参数保存捕捉的动态值

- 对脚本中固定值进行参数化
- 验证脚本执行的准确性

例阿里云登录提交请求POST，标红的表单数据都是：

```
loginId=pengyiping&password=xxxxx&checkCode=&appName=yunmail&appEntrance=
=default&hsid=EF6YBOFVO2-HMCZGQ6EJ940UOUC9LBY1-4XCSDI8I-JIHB&cid=d5db95f
3-1310-4e9a-80f0-726d7ddba95a&rdsToken=&umidToken=P989a56de2723d49264298e
4e6e2e2748&isRequiresHasTimeout=false&isRDSReady=true&isUMIDReady=true&umi
dGetStatusVal=255&lrfcf=7c349dcd12e63422cfc45b10f3a7a95&lang=zh_CN&scene=&
sMobile=false&screenPixel=1366x768&navUserAgent=Mozilla%2F4.0+(compatible%3B
+MSIE+8.0%3B+Windows+NT+6.1%3B+WOW64%3B+Trident%2F4.0%3B+GTB7.5%3B+
SLCC2%3B+.NET+CLR+2.0.50727%3B+.NET+CLR+3.5.30729%3B+.NET+CLR+3.0.30729%
3B+Media+Center+PC+6.0%3B+.NET4.0%3B+.NET4.0E)&navAppVersion=&navPlatfor
m=Win32&_csrf_token=9ptHyLjLqWoDDnJ67hBO95
```

可以通过HTTP协议抓包工具（httpwath、firebug、fiddler）查找需要关联的动态表单数据具体在前面哪个请求中返回：



通过抓包工具查找到具体请求后，在性能测试脚本请求后面添加获取动态值方法。

https://passport.alibaba.com/mini_login.htm?lang=zh_CN&appName=yunmail&appEntrance=default
&cssLink=&styleType=auto&bizParams=¬LoadSsoView=true¬KeepLogin=true&rnd=0.87515
22711732308请求响应内容：

```
<input id="fm-csrf-tk" name="_csrf_token" value="9ptHyLjLqWoDDnJ67hBO95" type="hidden" />
<input id="fm-rds-tk" name="rdsToken" value="" type="hidden" />
<input id="fm-cid" name="cid" value="d5db95f3-1310-4e9a-80f0-726d7ddba95a" type="hidden" />
<input id="fm-umid-token" name="umidToken" value="P989a56de2723d49264298e4e6e2e2748" type="hidden" />
<input id="fm-lang" name="lang" value="zh_CN" type="hidden" />
<input id="fm-hsid" name="hsid" value="EF6YBOFVO2-HMCZGQ6EJ940UOUC9LBY1-4XCSDI8I-JIHB" type="hidden" />
```

脚本示例：抓取动态关联值：

```
headers = [ NVPair('Accept', '/*/*'), NVPair('Host', 'passport.alibaba.com'), NVPair('Accept-Language', 'zh-CN,zh;');
result = HTTPRequest().GET('https://passport.alibaba.com/mini_login.htm?lang=zh_CN&appName=yunmail&appEntrance=de
csrf_token=PTS.HttpUtilities.valueFromTextBetween(result.getText(), 'name="csrf_token" value="', '"')
cid=PTS.HttpUtilities.valueFromTextBetween(result.getText(), 'name="cid" value="', '"')
umidToken=PTS.HttpUtilities.valueFromTextBetween(result.getText(), 'name="umidToken" value="', '"')
csrf_token=PTS.HttpUtilities.valueFromTextBetween(result.getText(), 'name="_csrf_token" value="', '"')

headers = [ NVPair('Accept', '/*/*'), NVPair('Host', 'passport.alibaba.com'), NVPair('Accept-
result = HTTPRequest().POST('https://passport.alibaba.com/newLogin/account/check.do?fromSite=
lrfcf=PTS.HttpUtilities.valueFromTextBetween(result.getText(), "lrfcf":"', '"')
```

使用动态关联值：

```
headers = [ NVPair('Accept', '/*/*'), NVPair('Host', 'passport.alibaba.com'), NVPair('Accept-Language',
result = HTTPRequest().POST('https://passport.alibaba.com/newLogin/Login.do?fromSite=9', \
    '''loginId=pengviping&password=!qaz2wsx&checkCode=&appName=yunmail&appEntr
    '''&hsid=''' +hsid.encode('utf-8')+'''&cid=''' +cid.encode('utf-8')+'''&rdsTc
    '''&umidToken=''' +umidToken.encode('utf-8')+'''&isRequiresHasTimeout=false&
    '''&umidGetStatusVal=255&lrfcf=''' +lrfcf.encode('utf-8')+'''&lang=zh_CN&sce
    '''&screenPixel=1366x768&navLanguage=zh-CN&navUserAgent=Mozilla%2F5.0+(WinC
    '''_csrf_token=''' +csrf_token.encode('utf-8'), headers)
PTS.Framework.addHttpCode(result.getStatusCode(), statusCode)
if(not PTS.HttpUtilities.checkResponse(200, "success":true)):
    PTS.Data.forCurrentTest.success = False
```

FAQ

1. 问：性能测试录制工具可以录制非Chrome、Firefox浏览器以外的其他应用HTTP/HTTPS协议请求吗？

答：不能，目前只能基于Firefox（支持37及以下版本）和Chrome浏览器进行录制，其他浏览器IE、QQ、Sogou等都是不可以的。

2. 问：行为和事务有何区别？

答：行为指的是一系列业务操作或请求组合，而事务是衡量行为里面一些业务操作或请求所需要的时间，在进行压测的时候，可以在性能测试实时监控图表中显示事务的时间。默认情况下，行为和事务是默认的名字，双击行为名称，可以进行修改事务的名字。

3. 问：脚本录制下来以后，运行没问题，但业务没有做成功，可能原因是？

答：这个问题的产生是多方面的，在性能测试领域叫做“脚本空转”，可能的原因有4点：

- （1）数据是一次性的，这样的话录制的时候就用掉了，再运行脚本的时候，数据是无效的，因此业务是做不成功的，例如订单支付、活期销户；可以通过参数化解决。
- （2）需要关联，有些业务关联性非常强，不进行关联的话，业务也是不成功的，例如用户ID，TokenID等；可以通过关联进行解决。
- （3）一些重要的请求丢失，默认录制工具会将非html类型的请求过滤掉，例如一些json类型的，而这些json类型也是必须的，可以通过选中此类型请求进行解决。
- （4）其他方面的原因等

4. 问：需要关联的数据来自于哪个请求，如何快速查找？

答：正在开发易于关联的数据查找，将会在后续版本中实现，目前最好的办法是咨询开发人员、自己定位、寻求性能测试技术人员技术支持。

5. 问：哪些数据需要关联？

答：一般情况下，最快的办法是咨询开发人员；也可以通过录制两份脚本进行对比，不一样的数据可能就需要关联；对于有一定经验的相关人员可以通过猜测，例如用户ID，TokenID等。

6. 问：参数化和关联必须有什么？

答：不一定，看业务规则，如果业务没有关联的话，可以不需要，为了更加真实的模拟实际业务场景，建议尽可能的将数据进行参数化。

7. 问：部分Chrome浏览器用户安装插件后，重新打开Chrome浏览器录制工具插件自动灰掉，不能使用

答：由于非来自Chrome网上应用店扩展程序，已停用的扩展程序均显示为灰色，且无法重新启用，需要在扩展程序中删除后重新安装。

[返回企业版测试脚本](#)

[返回企业版测试脚本](#)

脚本调试使用手册

1. 功能概述

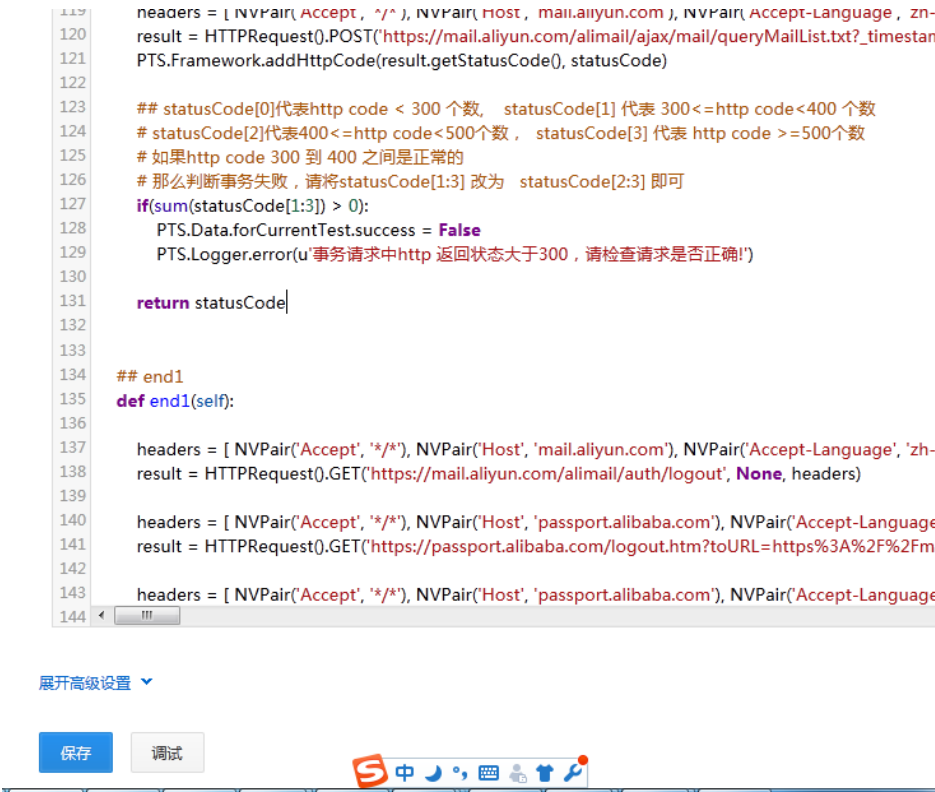
脚本开发完成后，需通过回放脚本、调试脚本代码逻辑和验证虚拟用户的基本业务功能是否成功，在性能测试中通过回放请求快照及执行日志信息来查看请求成功或失败后需要解决的错误和问题。

2. 脚本常见错误类型



3. 脚本调试使用流程

- i. 脚本编辑完成后，点击调试按钮



- ii. 调试运行完成后查看调试结果
调试运行完毕后自动跳转到调试详情页面，查看请求快照和执行日志。

调试详情

返回测试脚本

请求快照

执行日志

URL	状态	类型	域	时间线
▶ GET login	200 OK	text/html; charset=UTF-8	mail.aliyun.com	 151ms
▶ GET mini_login.htm?lan...	200 OK	text/html; charset=UTF-8	passport.alibaba.com	
▶ GET cache	200 OK	text/html; charset=UTF-8	mail.aliyun.com	
▶ POST check.do?fromSit...	200 OK	text/javascript; charset=UTF-8	passport.alibaba.com	
▶ POST login.do?fromSite...	200 OK	text/javascript; charset=UTF-8	passport.alibaba.com	
▶ GET callbackForHavana?...	302 Moved Temporarily	text/html; charset=UTF-8	mail.aliyun.com	 25ms
▶ GET alimail	200 OK	text/html; charset=UTF-8	mail.aliyun.com	 33ms
▶ POST sendMail.txt?_tim...	200 OK	text/html; charset=UTF-8	mail.aliyun.com	 26ms

0ms

请求开始时间

请求各阶段的开始时间和持续时间：

+0ms

0ms

域名解析

+0ms

20ms

建立连接

+20ms

127ms

等待响应

+147ms

4ms

接收数据

 64ms

调试详情

返回测试脚本

请求快照

执行日志

URL	状态	类型	域	时间线
GET login	200 OK	text/html; charset=UTF-8	mail.aliyun.com	151ms

头信息

响应

<!DOCTYPE html>
<html>
 <head>
 <meta http-equiv="X-UA-Compatible" content="IE=edge,chrome=1" />
 <meta name="description" content="阿里云邮, 阿里云邮箱, 专业云端邮箱, 安全稳定, 全球畅通。支持3g超大附件, 垃圾邮件拦截率超过99%, 阿里云邮箱, qq邮箱, gmail, hotmail, 雅虎邮箱, 139邮箱, 新浪邮箱 />
 <meta name="keywords" content="邮件, 邮箱, 电子邮件, 电子邮箱, 免费邮箱, 阿里云邮, 阿里云邮箱, 免费电子邮箱, mail, email, 个人邮箱, 企业邮箱, 云端, 邮箱app />
 <link rel="shortcut icon" href="/static/2693276/images/favicon.ico" type="image/x-icon"/>
 <link rel="bookmark" href="/static/2693276/images/favicon.ico" type="image/x-icon"/>
 <title>阿里云邮--免费邮箱</title>
 <link rel="stylesheet" type="text/css" href="https://gwebmail4f.alicdn.com/static/2693276/login/freemail/styles/login.css"/>
 <script type="text/javascript" src="https://gwebmail4f.alicdn.com/static/2693276/scripts/jquery-1.10.2.js"></script>
 <script type="text/javascript" src="https://gwebmail4f.alicdn.com/static/2693276/scripts/jquery.easing.1.3.js"></script>
 </head>
<body>

请求快照可查看具体某个HTTP请求响应码状态、类型、请求响应时间分解，点击某个请求可展开查看请求和响应报文详细信息。

调试详情

返回测试脚本

请求快照

执行日志

日志详情:

2015-05-06 10:17:50,690 400273401430878669 INFO v125013177-27458-20 : Java(TM) SE Runtime Environment 1.6.0_32-b05; OpenJDK (Taobao) 64-Bit S
20.23.2.ali927.e15.x86_64
2015-05-06 10:17:50,701 400273401430878669 INFO v125013177-27458-20 : time zone is CST (+0800)
2015-05-06 10:17:53,596 400273401430878669 INFO v125013177-27458-20 : script version: 1
2015-05-06 10:17:53,599 400273401430878669 INFO v125013177-27458-20 : start time is 1430878673599 ms since Epoch
2015-05-06 10:17:53,806 400273401430878669 INFO v125013177-27458-20 thread-0: https://mail.aliyun.com/alimail/auth/login -> 200 OK, 16996 bytes
2015-05-06 10:17:53,889 400273401430878669 INFO v125013177-27458-20 thread-0: https://passport.alibaba.com/mini_login.htm?lang=zh_CN&appName=LoadSsoView=true&KeepLogin=true&cmd=0.0054699630776792185 -> 200 OK, 8731 bytes
2015-05-06 10:17:53,933 400273401430878669 INFO v125013177-27458-20 thread-0: https://mail.aliyun.com/alimail/auth/cache -> 200 OK, 2199 bytes
2015-05-06 10:17:53,964 400273401430878669 INFO v125013177-27458-20 thread-0: https://passport.alibaba.com/newlogin/account/check.do?fromSite=2015-05-06 10:17:54,032 400273401430878669 INFO v125013177-27458-20 thread-0: https://passport.alibaba.com/newlogin/login.do?fromSite=9 -> 200
2015-05-06 10:17:54,060 400273401430878669 WARN v125013177-27458-20 thread-0: https://mail.aliyun.com/alimail/auth/callbackForHavana?reurl=%2Fure the next URL is https://mail.aliyun.com/alimail/auth/loginFail?reurl=%2Ffailmail%2F]

标准输出:

[INFO][SCRIPT:20,016,255] Transaction: name=action2, id=1
logincsrf_token= SYvNGiPAQNxcgggc8WqJo1
logincid= 3d9c7b2e-e259-491f-a99b-bcc33e2e770a
loginumidToken= P0abb17f9d552fd5ad897c302064f6360
loginhsid= EF6YEOFVO2-BXPZ20QPJS61P5J0TE5N1-1TU74C9I-KPMO2
loginlrfcf= 117452f73f718b51955c0971ecff79b5
login Failed!
loginst=
csrf_token=
root_token=

执行日志包含日志详情和标准输出两个模块,日志详情内容主要包含用户业务运行相关日志，如压测进程启动、业务请求开始执行时间、状态、响应内容大小、请求执行线程、PTS.Logger.*日志函数输出等信息；标准输出内容主要包含“标准正常输出”和“标准错误输出”两部分，标准正常输出主要包含事务、print、system.out语句等输出信息，标准错误输出包含脚本编写错误、运行时异常等输出信息。

4. 常用脚本调试验证方法

4.1 手工验证

i. 手工业务验证

在脚本执行期间或执行完成后，通过手工检查脚本业务功能是否成功实现，比如：通过被测程序后台管理检查脚本模拟用户是否登录成功；检查脚本下载文件是否在压力机本地；检查脚本新增业务记录或者修改业务记录是否与查询的业务记录一致等。

ii. 数据库更新验证

在脚本执行期间或执行完成后，通过数据库查询数据更新状态是否和预期脚本实现的状态一致。

iii. 回放快照记录

在脚本执行完成后，通过查看性能测试回放快照的请求响应状态及响应内容来判断请求是否成功处理响应。注：此功能只限于HTTP协议。



- iv. 日志输出记录

在脚本执行完成后，通过查看性能测试执行日志输出记录查看脚本请求、事务处理等过程详细信息来查看和验证脚本正确性。

[调试详情](#)
[返回测试脚本](#)

[请求快照](#)
[执行日志](#)

日志详情:

```

2015-05-06 10:17:50.690 400273401430878669 INFO v125013177-27458-20 : Java(TM) SE Runtime Environment 1.6.0_32-b05: OpenJDK (Taobao) 64-Bit Sr
20.23.2.al927.e15.x86_64
2015-05-06 10:17:50.701 400273401430878669 INFO v125013177-27458-20 : time zone is CST (+0800)
2015-05-06 10:17:53.596 400273401430878669 INFO v125013177-27458-20 : script version: 1
2015-05-06 10:17:53.599 400273401430878669 INFO v125013177-27458-20 : start time is 1430878673599 ms since Epoch
2015-05-06 10:17:53.806 400273401430878669 INFO v125013177-27458-20 thread-0: https://mail.aliyun.com/alimail/auth/login -> 200 OK, 16996 bytes
2015-05-06 10:17:53.889 400273401430878669 INFO v125013177-27458-20 thread-0: https://passport.alibaba.com/mini_login.htm?lang=zh_CN&appName=
Params=~LoadSsoView=true&KeepLogin=true&rnd=0.0054699630776792185 -> 200 OK, 8731 bytes
2015-05-06 10:17:53.933 400273401430878669 INFO v125013177-27458-20 thread-0: https://mail.aliyun.com/alimail/auth/cache -> 200 OK, 2199 bytes
2015-05-06 10:17:53.964 400273401430878669 INFO v125013177-27458-20 thread-0: https://passport.alibaba.com/newlogin/account/check.do?fromSite=
2015-05-06 10:17:54.032 400273401430878669 INFO v125013177-27458-20 thread-0: https://passport.alibaba.com/newlogin/login.do?fromSite=9 -> 200
2015-05-06 10:17:54.060 400273401430878669 WARN v125013177-27458-20 thread-0: https://mail.aliyun.com/alimail/auth/callbackFortHavana?reurl=%2F
ure the next URL is https://mail.aliyun.com/alimail/auth/loginFail?reurl=%2Falimail%2F

```

4.2 自动验证

- i. HTTP响应状态码验证

通过模板编辑和录制工具生成的脚本会自动对事物的每个HTTP请求响应码进行验证。性能测试默认请求响应码中包含300以上的响应码，定义事务失败。如果业务需求认为300到400之间的响应码是可接受的，可手工修改事务成功或失败判断语句。

```

## statusCode[0]代表http code < 300 个数， statusCode[1] 代表 300<=http code<400 个数
# statusCode[2]代表400<=http code<500个数， statusCode[3] 代表 http code >=500个数
# 如果http code 300 到 400 之间是正常的
# 那么判断事务失败，请将statusCode[1:3] 改为 statusCode[2:3] 即可
if(sum(statusCode[1:3]) > 0):
    PTS.Data.forCurrentTest.success = False
    PTS.Logger.error(u'事务请求中http 返回状态大于300，请检查请求是否正确!')

return statusCode

```

注：此功能只限于HTTP协议。

- ii. 检查点验证

通过设置检查点可以验证服务器响应内容是否和预期内容一致来判断请求或事物是否成功。

手工编写脚本示例代码：

```

headers = [ NVPair('Accept', '*/*'), NVPair('Host', 'passport.alibaba.com'), NVPair('Accept-Language', 'zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3'), NVPair('User-Agent', 'Mozilla/5.0 (Windows NT 6.0; rv:2.0.1 Gecko/20100101 Firefox/4.0.1)') ]
result = HTTPRequest().POST('https://passport.alibaba.com/newlogin/login.do?fromSite=9', {'loginId=barneyzhang&password=1q2w3e4r5t&check

```

```

###添加登录请求检查点，判断登录业务是否成功
if(PTS.HttpUtilities.checkResponse(200, "success:true") == True):
    print 'Login Success!'
else:
    print 'login Failed!'

```

示例代码中通过checkResponse函数检查登录请求响应码和响应内容来判断登录请求是否成功；登录成功输出“login Success !”，登录失败输出“login Failed”；

模板编写脚本示例：

高级属性 [帮助链接](#)

Header

Cookie

检查点

```
"success":true|
```

确定

取消

查看执行日志标准输出：

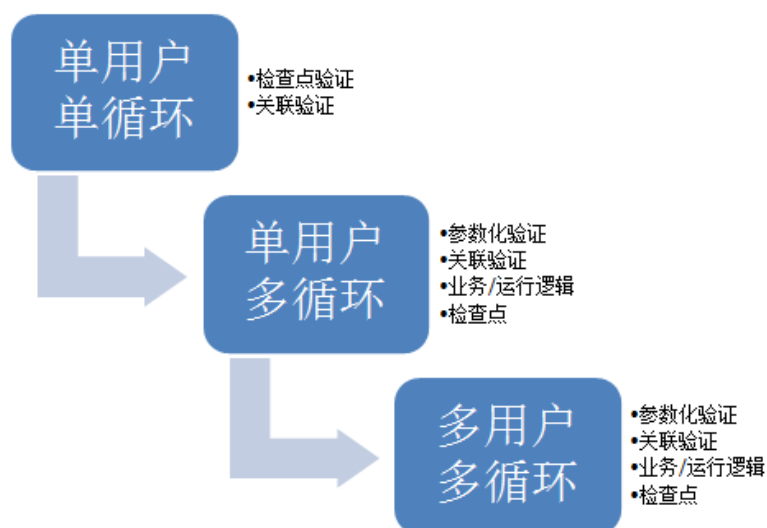
标准输出:

```
[INFO][SCRIPT:20,016,255] Transaction: name=action2, id=1  
loginsrf_token= 5YvNGiPAQNxcgggc8WqJo1  
logincid= 3d9c7b2e-e259-491f-a99b-bcc33e2e770a  
loginumidToken= P0abb17f9d552fd5ad897c302064f6360  
loginhsid= EF6YEOFVO2-BXPZ20QPJS61P5J0TE5N1-1TU74C9I-KPMO2  
loginlrfcf= 117452f73f718b51955c0971ecff79b5  
login Failed!
```

通过标准输出日志我们可以看到login Failed信息，表示登录请求失败。

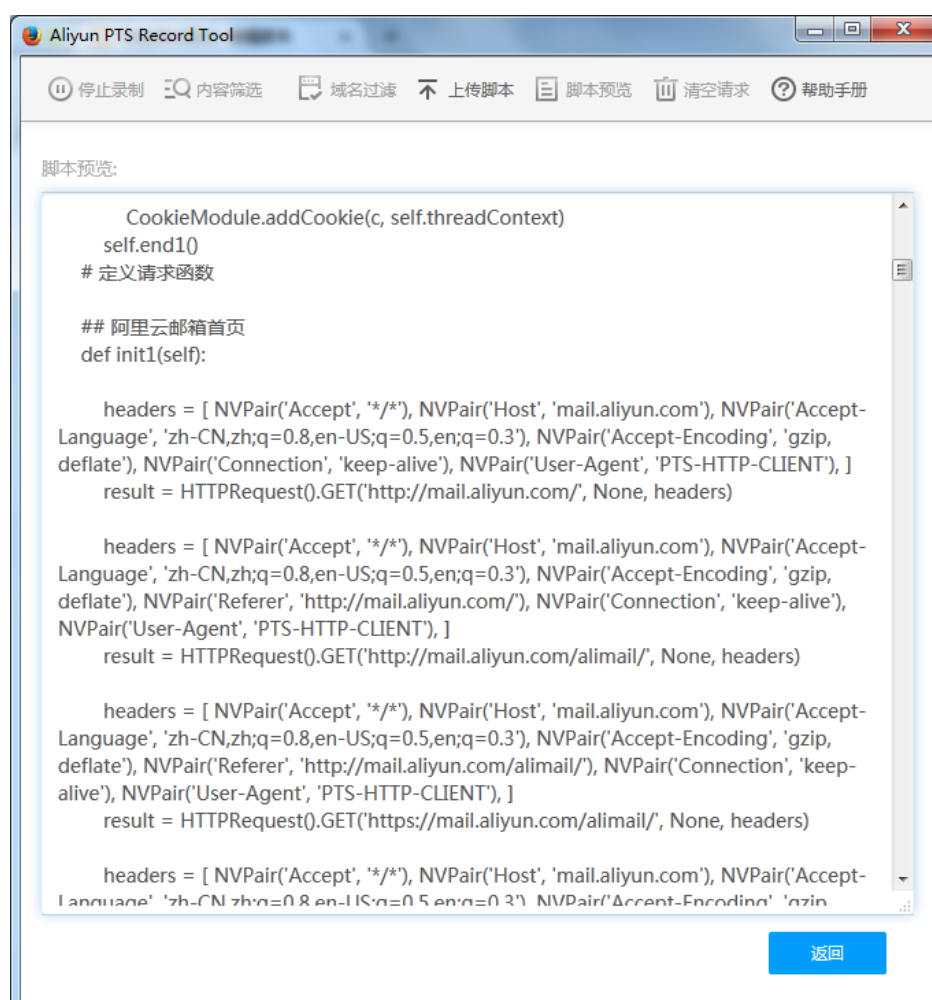
5. 脚本调试使用案例-阿里云邮箱发送邮件脚本

性能测试脚本调试验证通用流程：



5.1 录制脚本

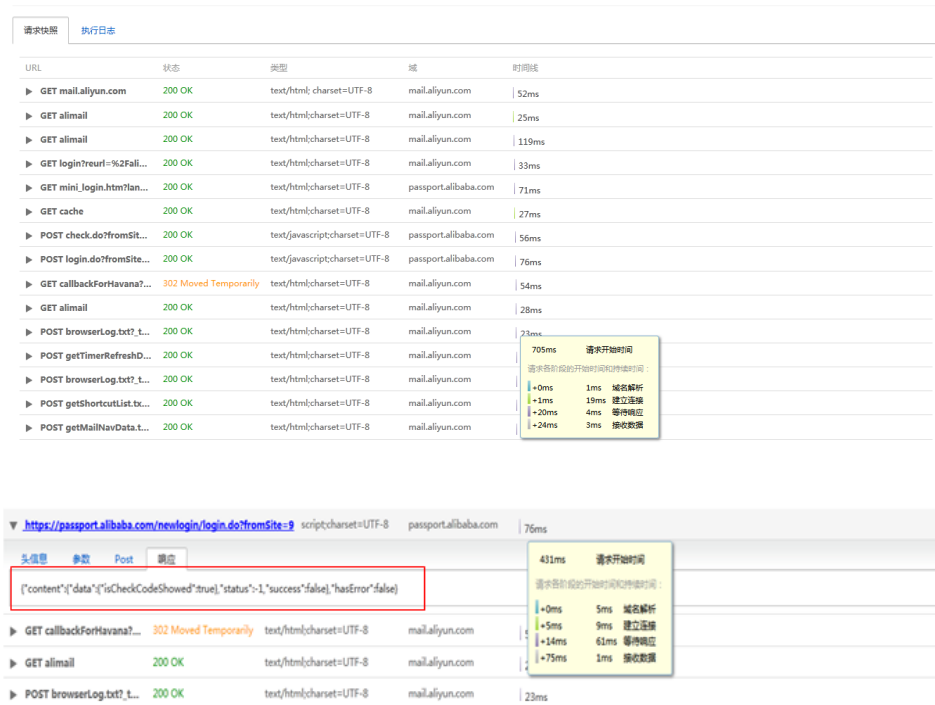
录制阿里云邮箱发送功能整个业务流程请求，上传保存。



5.2 登录手工回放、检查点验证调试

点击调试按钮，回放录制请求，可以手工或者添加检查点进行验证脚本正确性。

登录请求手工验证：



登录请求响应码虽然返回200，但是通过请求响应内容检查登录失败（HTTP200响应码表示服务器成功处理请求并响应，但是从业务角度来说200响应并不表示请求业务处理成功）。

添加检查点自动验证：

```
###{"content":{"data":{"appEntrance":"yunmail","resultCode":100,"st":"1OIYu6ER4DLAB9_PKRlue0w","accountInfoLogKey":"303",
###{"havanaCode":303,"status":0,"success":true,"hasError":false}
headers = [ NVPair('Accept', '*/*'), NVPair('Host', 'passport.alibaba.com'), NVPair('Accept-Language', 'zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3'), NVPair('Accept-
result = HTTPRequest().POST('https://passport.alibaba.com/newlogin/login.do?fromSite=9', "loginId=...&password=...&checkCode=&apph
PTS.Framework.addHttpCode(result.getStatusCode(), statusCode)

if(PTS.HttpUtilities.checkResponse(200,"success":true)):
    PTS.Logger.info(u'阿里云邮箱登录成功！')
else:
    PTS.Logger.error(u'阿里云邮箱登录失败！')
    PTS.Logger.info(u'登录请求失败响应数据：'+Result.getText())
```

在登录请求后面添加PTS.HttpUtilities.checkResponse(200,'success':true)检查点函数验证服务器是否成功响应请求及请求内容是否包含'success':true文本。如果登录失败输出自定义失败标志和失败请求的响应数据到日志详情模块。

添加检查点后进行调试，执行日志如下：

```
v125013176-50491-9 thread-0 [ run-0, test-1 ]: https://passport.alibaba.com/newlogin/login.do?fromSite=9 -> 200 OK, 99 bytes
{ v125013176-50491-9 thread-0 [ run-0, test-1 ]: 阿里云邮箱登录失败！
v125013176-50491-9 thread-0 [ run-0, test-1 ]: 登录请求失败响应数据：{"content":{"data":{"titleMsg":"请输入验证码"},"status":-1,"success":false,"hasError":false}}
```

通过日志详情看到通过检查点验证阿里云邮箱登录失败，失败响应内容中提示“请输入验证码”。

5.3 登录关联调试

阿里云邮箱登录业务规则需要，登录请求Body表单部分字段数据需进行关联，关联完成后进行脚本调试验证关联是否成功。

登录请求提交表单内容：

```
loginId=xxxx&password=xxxxx&checkCode=&appName=yunmail&appEntrance=default&hsid=EF6YBOFVO2-HMCZGQ6EJ940UOUC9LBY1-ZEZ05I8I-8FC9&cid=0b7a5b17-c123-49a5-8b1b-68d0b220379b&rdsToken=&umidToken=Pf17afcbd614a49ddc57be94bb448ef8c&isRequiresHasTimeout=false&isRDSReady=true&isUMIDReady=true&umidGetStatusVal=255&lrfcf=71a33e2b38f7ffc450148460143eedd1&lang=zh_CN&scene=&isMobile=false&screenPixel=1366x768&navlanguage=zh-CN&navUserAgent=Mozilla%2F5.0+(Windows+NT+6.1%3B+WOW64)+AppleWebKit%2F537.36+(KHTML%2C+like+Gecko)+Chrome%2F41.0.2272.118+Safari%2F537.36&navAppVersion=&navPlatform=Win32&_csrf_token=D608Nh4BxHeOIbtH14vyf2
```

注：标红的字体为服务器动态返回值，需要参数化关联值。

添加关联：

```
###<input id="fm-csrf-tk" name="_csrf_token" value="YuS9JdM4FY9pSC9ATgq5d" type="hidden" />
###<input id="fm-cid" name="cid" value="332de424-f851-4adb-b6b7-60bdc60d2b57" type="hidden" />
###<input id="fm-umid-token" name="umidToken" value="Paa35da66c37758e8c4e75b7ccb5180d5" type="hidden" />
###<input id="fm-hsid" name="hsid" value="EF6YPWXXVO2-TWPZ5M4VKR98K2BJMD973-VC7WVK9I-C6YP4" type="hidden" />
headers = [ NVPair('Accept', '*/*'), NVPair('Host', 'passport.alibaba.com'), NVPair('Accept-Language', 'zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3'), N
result = HTTPRequest().GET('https://passport.alibaba.com/mini_login.htm?lang=zh_CN&appName=yunmail&appEntrance=default&cssLink=&

self._csrf_token=PTS.HttpUtilities.valueFromBodyBetween("_csrf_token" value="");
if(self._csrf_token==""):
    PTS.Logger.info(u'_csrf_token关联值为空或关联错误，请检查！')
else:
    PTS.Logger.info(u'_csrf_token : '+self._csrf_token)

self.cid=PTS.HttpUtilities.valueFromBodyBetween("cid" value="");
if(self.cid==""):
    PTS.Logger.info(u'cid关联值为空或关联错误，请检查！')
else:
    PTS.Logger.info(u'cid : '+self.cid)

self.umidToken=PTS.HttpUtilities.valueFromBodyBetween("umidToken" value="");
if(self.umidToken==""):
    PTS.Logger.info(u'umidToken关联值为空或关联错误，请检查！')
else:
    PTS.Logger.info(u'umidToken : '+self.umidToken)
```

```
self.hsid=PTS.HttpUtilities.valueFromBodyBetween("hsid" value="");
if(self.hsid!=""):
    PTS.Logger.info(u'hsid关联值为空或关联错误，请检查！')
else:
    PTS.Logger.info(u'hsid : '+self.hsid)

##("content":{"data":{"lrfcf":"71a33e2b38f7ffc450148460143eedd1","resultCode":100,"status":0,"success":true,"hasError":false}
headers = [ NVPair('Accept', '*/*'), NVPair('Host', 'passport.alibaba.com'), NVPair('Accept-Language', 'zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3'), NV
result = HTTPRequest().POST('https://passport.alibaba.com/newlogin/account/check.do?fromSite=9', ""loginId=pengyiping&password=&checkC

self.lrfcf=PTS.HttpUtilities.valueFromBodyBetween("lrfcf":"");
if(self.lrfcf!=""):
    PTS.Logger.info(u'lrfcf关联值为空或关联错误，请检查！')
else:
    PTS.Logger.info(u'lrfcf : '+self.lrfcf)
```

参数化登录请求表单中动态值：

```
loginId=xxxx&password=xxxx&checkCode=&appName=yunmail&appEnt
rance=default&hsid=self.hsid &cid= self.cid&rdsToken=&umidToken= self.
umidToken
&isRequiresHasTimeout=false&isRDSReady=true&isUMIDReady=true&um
idGetStatusVal=255&lrfcf=self.lrfcf&lang=zh_CN&scene=&isMobile=false
&screenPixel=1366x768&navlanguage=zh-CN&navUserAgent=Mozilla%2F
5.0+(Windows+NT+6.1%3B+WOW64)+AppleWebKit%2F537.36+(KHTML%
2C+like+Gecko)+Chrome%2F41.0.2272.118+Safari%2F537.36&navAppVers
ion=&navPlatform=Win32&_csrf_token=self._csrf_token
```

关联完成后，调试脚本结果：

日志详情：

```
2015-05-12 14:46:08.572 400296331431413167 INFO v101093172-1844-14 : Java(TM) SE Runtime Environment 1.6.0_32-b05: OpenJDK (Taobao) 64-Bit Server VM (20.0-b12-internal;
0.23.2.a1927.e15.x86_64
2015-05-12 14:46:08.584 400296331431413167 INFO v101093172-1844-14 : time zone is CST (+0800)
2015-05-12 14:46:11.708 400296331431413167 INFO v101093172-1844-14 : script version: 1
2015-05-12 14:46:11.711 400296331431413167 INFO v101093172-1844-14 : start time is 143141317111 ms since Epoch
2015-05-12 14:46:11.793 400296331431413167 INFO v101093172-1844-14 thread-0: http://mail.aliyun.com/ -> 200 OK, 222 bytes
2015-05-12 14:46:11.841 400296331431413167 INFO v101093172-1844-14 thread-0: http://mail.aliyun.com/alimail/ -> 200 OK, 701 bytes
2015-05-12 14:46:11.989 400296331431413167 INFO v101093172-1844-14 thread-0: http://mail.aliyun.com/alimail/ -> 200 OK, 847 bytes
2015-05-12 14:46:12.029 400296331431413167 INFO v101093172-1844-14 thread-0: https://mail.aliyun.com/alimail/auth/login?reurl=%2Falimail%2F -> 200 OK, 17569 bytes
2015-05-12 14:46:12.103 400296331431413167 INFO v101093172-1844-14 thread-0: https://passport.alibaba.com/mini_login.htm?lang=zh_CN&appName=yunmail&appEntrance=de
arams=-LoadSsoView=true-KeepLogin=true&rnd=0.0027902944861823986 -> 200 OK, 8731 bytes
2015-05-12 14:46:12.104 400296331431413167 INFO v101093172-1844-14 thread-0: _csrf_token : HigEslLdoQkckEVKfGfE581
2015-05-12 14:46:12.104 400296331431413167 INFO v101093172-1844-14 thread-0: cid : 6ecf9b17-4b91-40d9-a605-6c1248a3a8d4
2015-05-12 14:46:12.105 400296331431413167 INFO v101093172-1844-14 thread-0: umidToken : P673c37bf98d6cd8eabdb6cf1a08dba13
2015-05-12 14:46:12.105 400296331431413167 INFO v101093172-1844-14 thread-0: hsid : EF6YBQFVQ2-TWPZSOWTKQCS80IKWULT2-P90GYK9I-NFAR4
2015-05-12 14:46:12.133 400296331431413167 INFO v101093172-1844-14 thread-0: http://mail.aliyun.com/alimail/auth/cache -> 200 OK, 2200 bytes
2015-05-12 14:46:12.217 400296331431413167 INFO v101093172-1844-14 thread-0: https://passport.alibaba.com/newlogin/account/check.do?fromSite=9 -> 200 OK, 125 bytes
2015-05-12 14:46:12.221 400296331431413167 INFO v101093172-1844-14 thread-0: lrfcf : e19799ce876f66bfc0bea49e1fa9a02
2015-05-12 14:46:12.221 400296331431413167 INFO v101093172-1844-14 thread-0: starting, will do 1 run
2015-05-12 14:46:12.402 400296331431413167 INFO v101093172-1844-14 thread-0: [run-0, test-1]: https://passport.alibaba.com/newlogin/login.do?fromSite=9 -> 200 OK, 180 bytes
2015-05-12 14:46:12.404 400296331431413167 INFO v101093172-1844-14 thread-0: [run-0, test-1]: 阿里云邮箱登录成功！
```

通过日志详情看到关联的服务器动态值输出及登录请求检测点日志输出。

5.4 发送邮件检查点、关联、参数化调试

阿里云邮箱发送邮件功能检测点添加、动态值关联、参数化验证。

发送邮件请求表单内容：

```
mailJsonData=%7B%22from%22%3A%7B%22email%22%3A%22xxxx%40aliyun.com%22%2C%22name%22%3A%22xxxx%22%7D%2C%22subject%22%3A%22sendmailtest%22%2C%22atFlagId%22%3A%221431398280800%22%2C%22to%22%3A%5B%7B%22name%22%3A%22181791781%22%2C%22displayName%22%3A%22181791781%22%2C%22email%22%3A%22181791781%40qq.com%22%2C%22displayEmail%22%3A%22181791781%40qq.com%22%2C%22namePinYin%22%3A%5B%5D%2C%22encDisplayEmail%22%3A%22181791781%40qq.com%22%2C%22encDisplayName%22%3A%22181791781%22%2C%22clientExtraInfo%22%3A%7B%7D%5D%2C%22highPriority%22%3Afalse%2C%22saveToSendFolder%22%3Atrue%2C%22separatedSend%22%3Afalse%2C%22draftSessionId%22%3A%22---.JIKS6_fdf6fd5a-3365-42c4-ab4c-227d724aca8a%22%2C%22attachList%22%3A%5B%5D%2C%22bigAttachList%22%3A%5B%5D%2C%22html%22%3Atrue%2C%22body%22%3A%22%3Cp+style%3D%5C%22margin%3A+0px%3B+clear%3A+both%3B%5C%22%3E%3Cspan+style%3D%5C%22font-family%3A+Tahoma%2CArial%2CSTHeiti%2CSimSun%3B+font-size%3A+14px%3B+color%3A+rgb(0%2C+0%2C+0)%3B%5C%22%3Esendmailtest%3C%2Fspan%3E%3C%2Fp%3E%22%7D&_csrf_token_=QXltVG9rZW4tMjk1MDMzLXFjaFdob2o2dkFyVURyUlJ2S0pMa2dwcHlNeU1PTWQ1RVBaQlZlNlNKeUJsVmdlcnd0&_root_token_=dC0yNjQ0NTYtU2h0YzNwMFZDag239592&_refer_hash_=h%3DW1siNSIsInNxbV83M18xNDMxMzk4MjgwMzE3Iix7InR5cGUiOiJuZXcifV1d&_frame_=5_sqm_73_1431398280317_1
```

请求Body表单中红色为需要参数化内容：发件人地址181791781%40qq.com和发件人名字181791781，参数化后实现给不通的人发送邮件。紫色内容为需要参数化之前关联的动态值。

参数化Body表单中的发件人地址和名字：

上传参数化文件，参数化发件人地址和发件人名字

sendmail_addr	sendmail_displayname
181791781@aliyun.com	181791781
181791782@aliyun.com	181791782
181791783@aliyun.com	181791783

```

19 ###导入参数化函数、转URL码函数相关类
20 from com.aliyun.pts import DsvReader
21 from com.aliyun.pts import ParamManager
22 import urllib
23 ###初始化参数文件
24 params = ParamManager.getInstance()
25 params.addProvider(DsvReader(u"mailaddr.csv"))

# 主体压测方法，每个线程在测试生命周期内会循环调用该方法
def __call__(self):
    PTS.Data.delayReports = 1
    for c in self.init_cookies:
        CookieModule.addCookie(c, self.threadContext)
    ###每次迭代取下一行值
    params.nextRecord(u'mailaddr.csv')

###参数化取值及转URL码
sendmail_addr=urllib.quote(params.getParamValue(u'mailaddr.csv:sendmail_addr').encode('utf-8'))
sendmail_displayname=urllib.quote(params.getParamValue(u'mailaddr.csv:sendmail_displayname').encode('utf-8'))
###输出参数化
PTS.Logger.info(sendmail_addr)
PTS.Logger.info(params.getParamValue(u'mailaddr.csv:sendmail_addr').encode('utf-8'))
PTS.Logger.info(sendmail_displayname)
PTS.Logger.info(params.getParamValue(u'mailaddr.csv:sendmail_displayname').encode('utf-8'))

mailJsonData=%7B%22from%22%3A%7B%22email%22%3A%22xxxx%40aliyun.com%22%2C%22name%22%3A%22xxxx%22%7D%2C%22subject%22%3A%22sendmailtest%22%2C%22atFlagId%22%3A%221431398280800%22%2C%22to%22%3A%5B%7B%22name%22%3A%22"+sendmail_displayname+"%22%2C%22displayName%22%3A%22"+sendmail_displayname+"%22%2C%22email%22%3A%22"+sendmail_addr+"%22%2C%22displayEmail%22%3A%22"+sendmail_addr+"%22%2C%22namePinYin%22%3A%5B%5D%2C%22encDisplayEmail%22%3A%22"+sendmail_addr+"%22%2C%22encDisplayName%22%3A%22"+sendmail_displayname+"%22%2C%22clientExtraInfo%22%3A%7B%7D%7D%5D%2C%22highPriority%22%3Afalse%2C%22saveToSendFolder%22%3Atrue%2C%22separatedSend%22%3Afalse%2C%22draftSessionId%22%3A%22-----JIKS6_fdf6fd5a-3365-42c4-ab4c-227d724aca8a%22%2C%22attachList%22%3A%5B%5D%2C%22bigAttachList%22%3A%5B%5D%2C%22html%22%3Atrue%2C%22body%22%3A%22%3Cp+style%3D%5C%22margin%3A+0px%3B+clear%3A+both%3B%5C%22%3E%3Cspan+style%3D%5C%22font-family%3A+Tahoma%2CArial%2CSTHeiti%2CSimSun%3B+font-size%3A+14px%3B+color%3A+rgb(0%2C+0%2C+0)%3B%5C%22%3Esendmailtest%3C%2Fspan%3E%3C%2Fp%3E%22%7D&_csrf_token_=QXltVG9rZW4tMjk1MDMzLXFjaFdob2o2dkFyVURyUlJ2S0pMa2dwchINeU1PTWQ1RVBaQjZlNlNKeUJsVmdlcnd0&_root_token_=dC0yNjQ0NTYtU2h0YzNwMFZDag239592&_refer_hash_=h%3DW1siNSIsInNxbV83M18xNDMxMzk4MjgwMzE3Iix7InR5cGUlOiJuZXcifV1d&_frame_=5_sqm_73_1431398280317_1

```

添加关联：

```

headers = [ NVPair('Accept', '*/*'), NVPair('Host', 'mail.aliyun.com'), NVPair('Accept-Language', 'zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3'), NVPa
result = HTTPRequest().GET('https://mail.aliyun.com/alimail/auth/callbackForHavana?reurl=%2Falimail%2F&st='+self.st, None, headers)
PTS.Framework.addHttpCode(result.getStatusCode(), statusCode)

self._csrf_token_ = PTS.HttpUtilities.valueFromCookie('_csrf_token_')
if(self._csrf_token_==""):
    PTS.Logger.info(u'_csrf_token_关联值为空或关联错误，请检查！')
else:
    PTS.Logger.info(u'_csrf_token_:'+self._csrf_token_)

headers = [ NVPair('Accept', '*/*'), NVPair('Host', 'mail.aliyun.com'), NVPair('Accept-Language', 'zh-CN,zh;q=0.8,en-US;q=0.5,en;q=0.3'), NVPa
result = HTTPRequest().GET('https://mail.aliyun.com/alimail/', None, headers)
PTS.Framework.addHttpCode(result.getStatusCode(), statusCode)
|
self._root_token_ = PTS.HttpUtilities.valueFromBodyBetween("_root_token_" value="");
if(self._root_token_==""):
    PTS.Logger.info(u'_root_token_关联值为空或关联错误，请检查！')
else:
    PTS.Logger.info(u'_root_token_:'+self._root_token_)

```

mailJsonData=%7B%22from%22%3A%7B%22email%22%3A%22xxxx%40aliyun.com%22%2C%22name%22%3A%22xxxx%22%2C%22subject%22%3A%22sendmailtest%22%2C%22atFlagId%22%3A%221431398280800%22%2C%22to%22%3A%5B%7B%22name%22%3A%22'+sendmail_displayname+'"%22%2C%22displayName%22%3A%22'+sendmail_displayname+'"%22%2C%22email%22%3A%22'+sendmail_addr+'"%22%2C%22displayEmail%22%3A%22'+sendmail_addr+'"%22%2C%22namePinYin%22%3A%5B%5D%2C%22encDisplayEmail%22%3A%22'+sendmail_addr+'"%22%2C%22encDisplayName%22%3A%22'+sendmail_displayname+'"%22%2C%22clientExtralInfo%22%3A%7B%7D%7D%5D%2C%22highPriority%22%3Afalse%2C%22saveToSendFolder%22%3Atrue%2C%22separatedSend%22%3Afalse%2C%22draftSessionId%22%3A%22----.JIKS6_fdf6fd5a-3365-42c4-ab4c-227d724aca8a%22%2C%22attachList%22%3A%5B%5D%2C%22bigAttachList%22%3A%5B%5D%2C%22html%22%3Atrue%2C%22body%22%3A%22%3Cp+style%3D%5C%22margin%3A+0px%3B+clear%3A+both%3B%5C%22%3E%3Cspan+style%3D%5C%22font-family%3A+Tahoma%2CArial%2CSTHeiti%2CSimSun%3B+font-size%3A+14px%3B+color%3A+rgb(0%2C+0%2C+0)%3B%5C%22%3Esendmailtest%3C%2Fspan%3E%3C%2Fp%3E%22%7D&_csrf_token_="'+self._csrf_token_+'"&_root_token_="'+self._root_token_+'"&_refer_hash_=h%3DW1sInSIslInNxbV83M18xNDMxMzk4MjgwMzE3Iix7InR5cGUiOiJ1ZXcifV1d&_frame_=5_sqm_73_1431398280317_1+</p>
</div>
<div data-label="Text">
<p>参数化和关联完成后，脚本调试结果：</p>
</div>
<div data-label="Text">
<pre>
thread-0: _csrf_token : elu308ahdNvgZohAHQjYA
thread-0: cid : 8ca722d8-17fd-4dc5-81cb-69598fe395d0
thread-0: umidToken : Pdf034f579bf4b5ab4eaeefbe2d8f4790
thread-0: hsid : EF6YEOFVO2-BXPZ20QPJS61P5J0TE5N1-4PIE2L9I-EWUR4
thread-0: https://mail.aliyun.com/alimail/auth/cache -> 200 OK, 2199 bytes
thread-0: https://passport.alibaba.com/newlogin/account/check.do?fromSite=9 -> 200 OK, 125 bytes
thread-0: lrfcf : 2e29e35f0381b5a475584dceee507507
thread-0: starting, will do 1 run
thread-0 [run-0, test-1]: https://passport.alibaba.com/newlogin/login.do?fromSite=9 -> 200 OK, 180 bytes
thread-0 [run-0, test-1]: 阿里云邮箱登录成功！
thread-0 [run-0, test-1]: st : 1Ds2CqMSdTTYWBWJjEVUtWyyw
6 thread-0 [run-0, test-1]: https://mail.aliyun.com/alimail/auth/callbackForHavana?reurl=%2Falimail%2F&st=1Ds2CqMl.aliyun.com/alimail/]
thread-0 [run-0, test-1]: _csrf_token_ : QXltVG9rZW4tMzI2NjYxLVpFQWZhbnZzYnISSEtiVGJGdzU5QlhqSzN3a1Z6eGQ
thread-0 [run-0, test-1]: https://mail.aliyun.com/alimail/ -> 200 OK, 26070 bytes
thread-0 [run-0, test-1]: _root_token_ : dC0yOTIyODEteE5EZFQ1VHlxNA821654
</pre>
</div>
<div data-label="Text">
<p>日志信息提示关联数据及登录事务已经成功。</p>
</div>
</div>

参数化和关联完成后，脚本调试结果：

```

thread-0: _csrf_token : elu308ahdNvgZohAHQjYA
thread-0: cid : 8ca722d8-17fd-4dc5-81cb-69598fe395d0
thread-0: umidToken : Pdf034f579bf4b5ab4eaeefbe2d8f4790
thread-0: hsid : EF6YEOFVO2-BXPZ20QPJS61P5J0TE5N1-4PIE2L9I-EWUR4
thread-0: https://mail.aliyun.com/alimail/auth/cache -> 200 OK, 2199 bytes
thread-0: https://passport.alibaba.com/newlogin/account/check.do?fromSite=9 -> 200 OK, 125 bytes
thread-0: lrfcf : 2e29e35f0381b5a475584dceee507507
thread-0: starting, will do 1 run
thread-0 [ run-0, test-1 ]: https://passport.alibaba.com/newlogin/login.do?fromSite=9 -> 200 OK, 180 bytes
thread-0 [ run-0, test-1 ]: 阿里云邮箱登录成功！
thread-0 [ run-0, test-1 ]: st : 1Ds2CqMSdTTYWBWJjEVUtWyyw
6 thread-0 [ run-0, test-1 ]: https://mail.aliyun.com/alimail/auth/callbackForHavana?reurl=%2Falimail%2F&st=1Ds2CqMl.aliyun.com/alimail/]
thread-0 [ run-0, test-1 ]: _csrf_token_ : QXltVG9rZW4tMzI2NjYxLVpFQWZhbnZzYnISSEtiVGJGdzU5QlhqSzN3a1Z6eGQ
thread-0 [ run-0, test-1 ]: https://mail.aliyun.com/alimail/ -> 200 OK, 26070 bytes
thread-0 [ run-0, test-1 ]: _root_token_ : dC0yOTIyODEteE5EZFQ1VHlxNA821654

```

日志信息提示关联数据及登录事务已经成功。

```
!]: https://mail.aliyun.com/alimail/ajax/mail/startDraft.txt?_timestamp_=1431398280731 -> 200 OK, 260 bytes
!]: 181791781%40aliyun.com
!]: 181791781@aliyun.com
!]: 181791781
!]: 181791781
!]: https://alimei-sub.alibaba.com/subscribe?sign=73aff64c506644efac9d002802bf388d&data=%7B%22appnam
```

参数化的具体值日志输出。

```
[ run-0, test-3 ]: https://mail.aliyun.com/alimail/ajax/mail/sendMail.txt?_timestamp_=1431398327758 -> 200 OK, 251 bytes
[ run-0, test-3 ]: 发送邮件失败！
[ run-0, test-3 ]: 发送邮件请求失败响应数据：2693705|2{"version":"2693705","log":"","status":4,"msg":"以下邮件地址不存在：181791781@a
```

发送邮件检查点日志输出提示发送邮件失败，失败请求响应数据提示邮件地址不存在。
修改参数化文件内容后重新上传后，调试结果如下：

```
thread-0 [ run-0, test-3 ]: https://mail.aliyun.com/alimail/ajax/mail/sendMail.txt?_timestamp_=1431398327758 -> 200 OK, 1516 bytes
thread-0 [ run-0, test-3 ]: 发送邮件成功！
thread-0 [ run-0, test-3 ]: https://alimei-sub.alibaba.com/subscribe?sign=73aff64c506644efac9d002802bf388d&data=%7B%22appname%
SEI5dmhJeQ396472%22%2C%22account%22%3A%22pengyiping%40aliyun.com%22%7D&timeout=18000&_timestamp_=1431398329688
```

5.5 单用户多循环脚本调试

在脚本栏点击快速启动，使用单个用户多次迭代循环脚本验证参数化、关联操作等。



快速启动任务默认执行时间为1分钟，输入1并发用户后点击确定启动任务执行，测试结果了可以查看任务执行情况 & 日志查看：

```

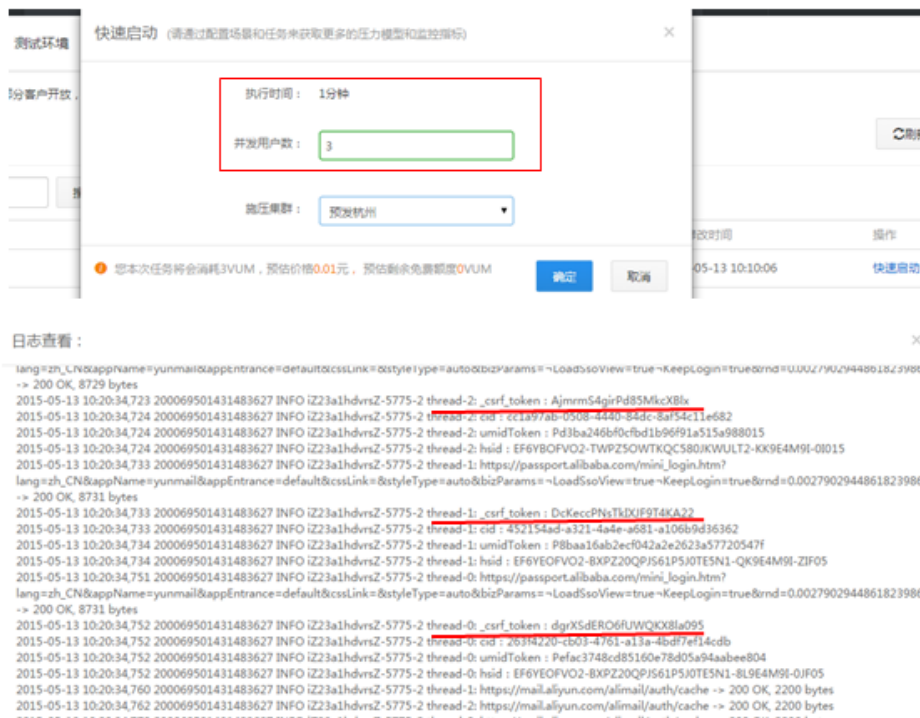
thread-0 [ run-0, test-20009869 ]: 181791781%40qq.com
thread-0 [ run-0, test-20009869 ]: 181791781@qq.com
thread-0 [ run-0, test-20009869 ]: 181791781
thread-0 [ run-0, test-20009869 ]: https://mail.aliyun.com/alimail/ajax/recipient/verifyRecipientDo
thread-0 [ run-0, test-20009869 ]: https://mail.aliyun.com/alimail/ajax/mail/sendMail.txt?_timestam
thread-0 [ run-0, test-20009869 ]: 发送邮件成功!
thread-0 [ run-0, test-20009869 ]: https://alimei-sub.alibaba.com/subscribe?sign=73aff64c506644efac
essionId%22%3A%22d02NTMxNDgtexQYmNFPUGFFSg487801%22%2C%22account%22%3A%22pengyiping%40aliyun.com%22
OK, 126 bytes
thread-0 [ run-1, test-20009868 ]: https://mail.aliyun.com/alimail/ajax/error/browserLog.txt?_times
thread-0 [ run-1, test-20009868 ]: https://mail.aliyun.com/alimail/ajax/mail/startDraft.txt?_timest
thread-0 [ run-1, test-20009869 ]: 181791782%40qq.com
thread-0 [ run-1, test-20009869 ]: 181791782@qq.com
thread-0 [ run-1, test-20009869 ]: 181791782
thread-0 [ run-1, test-20009869 ]: https://mail.aliyun.com/alimail/ajax/recipient/verifyRecipientDo
thread-0 [ run-1, test-20009869 ]: https://mail.aliyun.com/alimail/ajax/mail/sendMail.txt?_timestam
thread-0 [ run-1, test-20009869 ]: 发送邮件成功!
thread-0 [ run-1, test-20009869 ]: https://alimei-sub.alibaba.com/subscribe?sign=73aff64c506644efac
essionId%22%3A%22d02NTMxNDgtexQYmNFPUGFFSg487801%22%2C%22account%22%3A%22pengyiping%40aliyun.com%22
OK, 126 bytes
thread-0 [ run-2, test-20009868 ]: https://mail.aliyun.com/alimail/ajax/error/browserLog.txt?_times
thread-0 [ run-2, test-20009868 ]: https://mail.aliyun.com/alimail/ajax/mail/startDraft.txt?_timest
thread-0 [ run-2, test-20009869 ]: 181791783%40qq.com
thread-0 [ run-2, test-20009869 ]: 181791783@qq.com
thread-0 [ run-2, test-20009869 ]: 181791783
thread-0 [ run-2, test-20009869 ]: https://mail.aliyun.com/alimail/ajax/recipient/verifyRecipientDo
thread-0 [ run-2, test-20009869 ]: https://mail.aliyun.com/alimail/ajax/mail/sendMail.txt?_timestam
thread-0 [ run-2, test-20009869 ]: 发送邮件成功!

```

在日志查看中可以看到单用户多循环迭代运行情况下参数值变化情况、以及事务检查点信息情况。

5.6 多用户多循环脚本调试

在脚本栏点击快速启动，使用多个用户多次迭代循环脚本验证参数化、关联操作等。



Thread-X线程号代表不同的并发用户数，通过日志可以看出不同并发用户数的各自关联动态值。

```

thread-1 [ run-0, test-20009868 ]: https://mail.aliyun.com/alimail/ajax/error/browserLog.txt?_times
thread-2 [ run-0, test-20009868 ]: https://mail.aliyun.com/alimail/ajax/mail/startDraft.txt?_timest
thread-2 [ run-0, test-20009869 ]: 181791781%40qq.com
thread-2 [ run-0, test-20009869 ]: 181791781@qq.com
thread-2 [ run-0, test-20009869 ]: 181791781
thread-2 [ run-0, test-20009869 ]: 181791781
thread-1 [ run-0, test-20009868 ]: https://mail.aliyun.com/alimail/ajax/mail/startDraft.txt?_timest
thread-1 [ run-0, test-20009869 ]: 181791782%40qq.com
thread-1 [ run-0, test-20009869 ]: 181791782@qq.com
thread-1 [ run-0, test-20009869 ]: 181791782
thread-1 [ run-0, test-20009869 ]: 181791782
thread-2 [ run-0, test-20009869 ]: https://mail.aliyun.com/alimail/ajax/recipient/verifyRecipientDo
thread-1 [ run-0, test-20009869 ]: https://mail.aliyun.com/alimail/ajax/recipient/verifyRecipientDo
thread-2 [ run-0, test-20009869 ]: https://mail.aliyun.com/alimail/ajax/mail/sendMail.txt?_timestam
thread-2 [ run-0, test-20009869 ]: 发送邮件失败!
thread-2 [ run-0, test-20009869 ]: 发送邮件请求失败响应数据: 2693705|2{"&quot;version&quot;:&quot;26
;email&quot;:&quot;pengyiping@aliyun.com&quot;,&quot;cost&quot;:3}
thread-0: https://alimail-sub.aliyunc.com/subscribe?sign=/3aff64c506644efac9d002802bf3888&data=%/B%2
TQTG4ZYKX3WVZ2UA804640%22%2C%22account%22%3A%22pengyiping%40aliyun.com%22%7D&timeout=18000&_timest
thread-0: starting, will run forever
thread-0 [ run-0, test-20009868 ]: https://mail.aliyun.com/alimail/ajax/error/browserLog.txt?_times
thread-0 [ run-0, test-20009868 ]: https://mail.aliyun.com/alimail/ajax/mail/startDraft.txt?_timest
thread-0 [ run-0, test-20009869 ]: 181791783%40qq.com
thread-0 [ run-0, test-20009869 ]: 181791783@qq.com
thread-0 [ run-0, test-20009869 ]: 181791783
thread-0 [ run-0, test-20009869 ]: 181791783
thread-0 [ run-0, test-20009869 ]: https://mail.aliyun.com/alimail/ajax/recipient/verifyRecipientDo
thread-1 [ run-0, test-20009869 ]: https://mail.aliyun.com/alimail/ajax/mail/sendMail.txt?_timestam
thread-1 [ run-0, test-20009869 ]: 发送邮件成功!
thread-0 [ run-0, test-20009869 ]: https://mail.aliyun.com/alimail/ajax/mail/sendMail.txt?_timestam
thread-0 [ run-0, test-20009869 ]: 发送邮件成功!
thread-0 [ run-0, test-20009869 ]: https://alimail-sub.aliyunc.com/subscribe?sign=/3aff64c506644efac
essionId%22%3A%22d0c02NT0ZMT0TG4ZYKX3WVZ2UA804640%22%2C%22account%22%3A%22pengyiping%40aliyun.com%22%

```

通过日志可以看出不同并发用户数的使用的具体参数化值及检测点日志输出信息。

[返回企业版测试脚本](#)

SDK使用手册

[返回性能测试控制台](#)

包：PTS

PTS包封装了大部分用于编写HTTP手工测试脚本的类和API，类主要有：Framework、Context、Data、Thread、Logger和HttpUtilities这六个。

类：Framework

Framework类主要用于测试引擎对脚本框架进行内部处理，使用户可以通过API对脚本中的函数进行性能计数（instrument），也提供了对HTTP返回码和响应处理时间分布的处理函数。一般用户无需自己使用这个类的方法进行操作。通过录制或者模版生成的脚本会自动加入Framework代码，只有用户完全手工编写脚本时才会用到Framework内的方法。

方法：instrumentMethod

定义：def instrumentMethod(tran_name, method_name, c)对“c”这个类（在性能测试脚本中固化为“TestRunner”这个类）的“method_name”方法进行编织，产生一个性能事务，对应性能测试平台上的事务名称为“tran_name”，这样在脚本执行时会自动在“method_name”方法执行前后加上时间戳，可以计算该方法的执行时间。示例：u’事务名’是性能测试页面显示的事务名，中文字符串前需要加u，viewPage1是TestRunner中定义的一个成员方法。

```
PTS.Framework.instrumentMethod (u'事务名', 'viewPage1', TestRunner)
```

方法：addHttpCode

定义：def addHttpCode(code, codeArray)

对脚本中HTTP请求返回的状态码进行累加计数，code是当前需要累加的状态码。codeArray是长度为4的int数组，codeArray[0]表示状态码小于300的个数，codeArray[1]表示状态码在[300, 400)区间的个数，codeArray[2]表示状态码在[400, 500)区间的个数，codeArray[3]表示状态码在500及以上的个数。

示例：

```
statusCode = [0L, 0L, 0L, 0L]
headers = [ NVPair('Accept', '*/'), NVPair('Connection', 'Keep-Alive'), NVPair('User-Agent', 'PTS-HTTP-CLIENT'), ]
result = HTTPRequest().GET('http://item.taobao.com/item.htm?spm=a217i_h.1303267.1998177633-0.10.hx5m95&id=12817144777', None, headers)
PTS.Framework.addHttpCode(result.getStatusCode(), statusCode)
```

方法：setExtraData

定义：def setExtraData(codeArray=None)

对脚本中HTTP的额外计数信息进行累加，包括状态码和响应时间分布。

示例：

```
statusCode = self.action3()
PTS.Framework.setExtraData(statusCode)
statusCode = self.action4()
PTS.Framework.setExtraData(statusCode)
```

类：Context

方法：getThreadContext

定义：def getThreadContext()

返回线程级上下文对象，使用该线程上下文对象可以对测试脚本的线程对象进行操作。

示例：

```
# 获取线程上下文中的Cookie信息
self.threadContext = PTS.Context.getThreadContext()
self.init_cookies = CookieModule.listAllCookies(self.threadContext)
# 往线程上下文中加入Cookie信息
self.threadContext = PTS.Context.getThreadContext()
CookieModule.addCookie(c, self.threadContext)
```

方法：setParamDirectory

定义：def setParamDirectory(dir)

测试脚本设置参数文件目录，通常情况不需要使用，默认为脚本目录下的data目录。如果用户有自定义的参数文件路径，可以通过该方法进行设置。

类：Data

开关值：delayReports

定义：

delayReports，“0”表示事务数据收集完之后立即汇报，“1”表示将数据收集延迟汇报，通常我们需要在脚本中根据请求返回内容对事务是否成功进行判断，所以需要将该开关设置为“1”。

示例：

```
# 延迟汇报数据
PTS.Data.delayReports = 1
# 事务调用
.....
.....
# 汇报数据
PTS.Data.report()
# 关闭延迟汇报数据
PTS.Data.delayReports = 0
```

开关值：forCurrentTest.success定义：

forCurrentTest.success，用于设置当前事务是否成功。True表示事务成功，False表示事务失败。

示例：

如果HTTP请求返回码不是200或者300，或者返回内容中不存在“test”字符串，就将当前事务标记为失败，事务默认为成功。

```
if(not PTS.HttpUtilities.checkResponse([200,300], "test")):
```

```
PTS.Data.forCurrentTest.success = False
```

类：Thread

方法：sleep

定义：def sleep(millis)

当前调用方法的线程进入睡眠状态，millis毫秒。

示例：

```
# 线程睡眠1秒  
PTS.Thread.sleep(1000)
```

方法：getThreadNumber

定义：def getThreadNumber()

返回当前调用线程的线程号，注意该线程号是当前施压进程启动线程的逻辑线程号，从0开始往上递增，不代表实际操作系统的线程号。

方法：getRunNumber

定义：def getRunNumber()

返回当前调用线程的迭代次数。

类：Logger

定义：

脚本日志类，所有的输出请使用Logger类进行，Logger可以输出级别为INFO、WARN、ERROR的三种日志。INFO包含WARN包含ERROR。根据用户在性能测试场景里设置的日志级别，用户可以看到不同的日志，比如用户在场景设置了INFO，则可以看到通过所有三种级别方法打印出的日志，如果设置为WARN，则只能看到通过warn和error方法打出的日志，如果设置成ERROR，则只能看到通过error方法打出的日志。建议用户在调试阶段可以将日志级别设为INFO，正式压测时设成WARN或者ERROR。

示例：

```
# 打印INFO级别的日志  
PTS.Logger.info(u'INFO级别的日志')  
  
# 打印WARN级别的日志  
PTS.Logger.warn(u'WARN级别的日志')
```

```
# 打印ERROR级别的日志
PTS.Logger.error(u'ERROR级别的日志')
```

类：HttpUtilities

方法：setKeepAlive

定义：def setKeepAlive(flag)

设置底层HTTP客户端引擎在发送请求时使用长连接还是短连接，默认使用长连接，flag为True或者False。

示例：

```
# 强制HTTP客户端引擎使用短连接，每次循环之后关闭连接
PTS.HttpUtilities.setKeepAlive(False)
```

方法：setUrlEncoding

定义：def setUrlEncoding(code)

设置底层HTTP客户端引擎在发送请求时使用的URL编码方式，默认使用GBK编码。如果用户的URL已经经过URL编码，则修改设置无效。

示例：

```
# 使用UTF-8来进行URL编码
PTS.HttpUtilities.setUrlEncoding('UTF-8')
```

法：setTimeout

定义：def setTimeout(time)

设置底层HTTP客户端引擎控制HTTP请求的Connection和reading Response超时时间，时间单位为毫秒。

示例：

```
# 设置超时时间120000毫秒
PTS.HttpUtilities.setTimeout(120000)
```

方法：setFollowRedirects

定义：def setFollowRedirects(flag)

设置底层HTTP客户端引擎针对类似302这样的跳转响应的处理方式。如果设置成True，则引擎会自动跟随跳转

内容进行继续请求，否则不再跟随。按照模板编写和手工编写引擎默认为True，录制下来的脚本引擎默认为False。示例：

```
# 跟随跳转请求
PTS.HttpUtilities.setFollowRedirects(True)
```

方法：valueFromCookie

定义：def valueFromCookie(cookieName, threadContext=None)

获取Cookie管理器中cookieName对应的值，当HTTP响应返回使用SET-COOKIE来设置Cookie时，这些Cookie会自动存储到Cookie管理器。

示例：

```
# 获取Cookie管理器中名字为JSESSIONID的Cookie值
PTS.HttpUtilities.valueFromCookie('JSESSIONID')

# 获取Cookie管理器中名字为JSESSIONID的Cookie值（仅在当前线程中查找）
PTS.HttpUtilities.valueFromCookie('JSESSIONID', self.threadContext)
```

方法：valueFromHeader

定义：def valueFromHeader(header)

获取HTTP响应中Header名为header的值。

示例：

```
# 获取HTTP响应中Header名为Location的值
PTS.HttpUtilities.valueFromHeader('Location')
```

方法：valueFromBodyInput

定义：def valueFromBodyInput(tokenName)

获取HTTP响应中标签名tokenName的input的值。

示例：

```
# 获取HTTP响应表单中input名称为username的控件值
PTS.HttpUtilities.valueFromBodyInput('username')
```

方法：valueFromHiddenInput

定义：def valueFromHiddenInput(tokenName)

获取HTTP响应中标签名为tokenName的隐藏控件的值

方法：valueFromTextBetween

定义：def valueFromTextBetween(text, left, right)

获取text中被left和right字符串包围的中间字符串，如果有多个匹配则返回第一个。

示例：

```
<html> .....<body>.....<img token=' tb_token' >KDJAK</img> </body> </html>
```

比如上面这段HTML代码，调用下面这个方法会返回“ KDJAK”

```
# 返回KDJAK
PTS.HttpUtilities.valueFromTextBetween(text, "'tb_token'>";, "<");
```

方法：valuesFromTextBetween

定义：def valuesFromTextBetween(text, left, right)

功能和valueFromTextBetween类似，返回被left和right包围的字符串列表。

示例：

```
<xml><cars><car name=" VW" /> <car name=" BMW" /> <car name=" BENZ" />
</cars></xml>
```

比如上面这段xml代码，调用下面这个方法会返回：[“VW” , “BMW” , “BENZ”]

```
PTS.HttpUtilities.valuesFromTextBetween(text, "name=\"", "\"")
```

方法：valueFromBodyBetween

定义：def valueFromBodyBetween(left,right)

获取HTTP响应中被left和right字符串包围的中间字符串，如果有多个匹配则返回第一个。

示例：

```
<html> .....<body>.....<img token=' tb_token' >KDJAK</img> </body> </html>
```

比如上面这段HTML代码，调用下面这个方法会返回“ KDJAK”

```
# 返回KDJAK
PTS.HttpUtilities.valueFromBodyBetween("'tb_token'>";, "<");
```

方法：valuesFromBodyBetween

定义：def valuesFromBodyBetween(left, right)

功能和valueFromBodyBetween类似，返回被left和right包围的字符串列表。

示例：

```
<xml><cars><car name=" VW" /> <car name=" BMW" /> <car name=" BENZ" />
</cars></xml>
```

比如上面这段xml代码，调用下面这个方法会返回：["VW" , "BMW" , "BENZ"]

```
PTS.HttpUtilities.valuesFromBodyBetween("name=\"", "\"")
```

方法：setProxyServer

定义：def setProxyServer(ip, port)

设置底层HTTP引擎使用IP为ip，端口为port的代理进行请求。

示例：

```
PTS.HttpUtilities.setProxyServer('localhost', 8888)
```

方法：setUseCookieModule

定义：def setUseCookieModule(flag)

设置脚本中是否使用CookieModule来进行cookie管理，默认使用。如果用户想在HTTP的header里自己设置Cookie，则需要通过调用这个方法将CookieModule禁用。

示例：

```
PTS.HttpUtilities.setUseCookieModule(False)
```

方法：checkResponse

定义：def checkResponse(code, expectedText=None)

对HTTP响应进行脚本，返回码是否为code，响应body里面是否包含expectedText。只有当返回码符合且expectedText存在时返回True。当expectedText为None时（可以不传入该参数），不进行文本校验。

示例：

```
# 当响应返回码为200，且返回body包含test则返回True
```

```
PTS.HttpUtilities.checkResponse(200, "test")
# 当响应返回码为200，且返回body包含有中文"测试"且返回True
PTS.HttpUtilities.checkResponse(200, u"测试")
```

方法：checkResponseByRegex

定义：def checkResponseByRegex(code, expectedPattern=None)

对HTTP响应进行脚本，返回码是否为code，响应body里面是否包含expectedPattern这样的正则字符串。只有当返回码符合且expectedPattern能够匹配时，返回True。当expectedPattern为None时（可以不传入该参数），不进行文本校验。

示例：

```
# 当响应返回码为200，且返回body包含IP地址则返回True
PTS.HttpUtilities.checkResponseByRegex(200, "^(([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])\\.){3}([0-9]|[1-9][0-9]|1[0-9]{2}|2[0-4][0-9]|25[0-5])$")
```

类：DsvReader

定义：

参数化文件读取操作类。

方法：DsvReader

定义：public DsvReader(fileName)

DsvReader类构造函数，初始化类成员变量

类：ParamManager

定义：

参数化功能管理类，提供对用户上传的参数文件进行管理。

方法：getInstance

定义：public static ParamManager getInstance()

返回参数化功能管理类静态实例对象。

示例：

一般情况下，下列代码放在全局位置，既class TestRunner上面

```
params = ParamManager.getInstance()
```

方法：addProvider

定义：public void addProvider(ParamProvider_Instance)

添加参数读操作类实例对象。

示例：

一般情况下，下列代码放在全局位置，既class TestRunner上面

添加mailaddr.csv文件读取操作实例对象

```
params.addProvider(DsvReader(u'mailaddr.csv'))
```

方法：nextRecord

定义：public boolean nextRecord (paramFileName)

public boolean nextRecord(paramFileName,bCircle)

根据参数文件名加载当前参数位置的下一个位置的参数值

示例：

一般情况下，下列代码放在全局位置，既class TestRunner上面

添加mailaddr.csv文件读取操作实例对象

```
params = ParamManager.getInstance()
params.addProvider(DsvReader(u'mailaddr.csv'))
```

一般情况下，下列代码放在 class TestRunner 成员函数里面，例如action里面

加载mailaddr.csv下一个位置的参数值，参数值循环，如果已到最后一个值，会继续取第一个值

```
params.nextRecord(u'mailaddr.csv')
```

加载mailaddr.csv下一个位置的参数值,参数值不循环，如果已到最后一个值，此函数值返回False

参数化数据用完后，要想脚本停止运行请添加下列代码（需要导入：from HTTPClient import ShutdownException）：

```
if (not params.nextRecord(u'mailaddr.csv', False)):
    raise ShutdownException("param file EOF")
```

方法：getParamValue

定义：public String getParamValue(paramName)

根据参数化文件和参数名取当前位置的参数值。

示例：

读取mailaddr.csv参数文件的sendmail_addr字段当前位置的值

```
params.getParamValue (u'mailaddr.csv:sendmail_addr')
```

附录一：HTTP示例脚本

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# PTS Script Version 1.0

# PTS脚本SDK：框架API、常用HTTP请求/响应处理API
from util import PTS
from HTTPClient import NVPair
from HTTPClient import Cookie
from HTTPClient import HTTPRequest
from HTTPClient import CookieModule

# 脚本初始化段，可以设置压测引擎的常用HTTP属性
#PTS.HttpUtilities.setKeepAlive(False)
#PTS.HttpUtilities.setUrlEncoding('GBK')
#PTS.HttpUtilities.setFollowRedirects(False)
#PTS.HttpUtilities.setUseCookieModule(False)

# 脚本执行单元类，每个VU/压测线程会创建一个TestRunner实例对象
class TestRunner:

# TestRunner对象的初始化方法，每个线程在创建TestRunner后执行一次该方法
def __init__(self):
self.threadContext = PTS.Context.getThreadContext()

self.action1()

self.action2()

self.init_cookies = CookieModule.listAllCookies(self.threadContext)

# 主体压测方法，每个线程在测试生命周期内会循环调用该方法
def __call__(self):
PTS.Data.delayReports = 1

for c in self.init_cookies:
```

```
CookieModule.addCookie(c, self.threadContext)

statusCode = self.action3()
PTS.Framework.setExtraData(statusCode)

statusCode = self.action4()
PTS.Framework.setExtraData(statusCode)

PTS.Data.report()
PTS.Data.delayReports = 0

# TestRunner销毁方法，每个线程循环执行完成后执行一次该方法
def __del__(self):

    for c in self.init_cookies:
        CookieModule.addCookie(c, self.threadContext)

    self.action5()

    self.action6()

# 定义请求函数
def action1(self):

    headers = [ NVPair('Accept', '*/*'), NVPair('Connection', 'Keep-Alive'), NVPair('User-Agent', 'PTS-HTTP-CLIENT'), ]
    result = HTTPRequest().GET('http://www.ptstest.com', None, headers)

    headers = [ NVPair('Accept', '*/*'), NVPair('Connection', 'Keep-Alive'), NVPair('Content-Type', 'application/x-www-
form-urlencoded'), NVPair('Host', 'www.ptstest.com'), NVPair('User-Agent', 'PTS-HTTP-CLIENT'), ]
    result = HTTPRequest().POST('http://www.ptstest.com/s', 'wd=test', headers)

def action2(self):

    headers = [ NVPair('Accept', '*/*'), NVPair('Connection', 'Keep-Alive'), NVPair('User-Agent', 'PTS-HTTP-CLIENT'), ]
    result = HTTPRequest().GET('http://www.ptstest.com/s?wd=test', None, headers)

    headers = [ NVPair('Accept', '*/*'), NVPair('Connection', 'Keep-Alive'), NVPair('Content-Type', 'application/x-www-
form-urlencoded'), NVPair('Host', 'www.ptstest.com'), NVPair('User-Agent', 'PTS-HTTP-CLIENT'), ]
    result = HTTPRequest().POST('http://www.ptstest.com/s', 'wd=test', headers)

def action3(self):
    statusCode = [0L, 0L, 0L, 0L]

    headers = [ NVPair('Accept', '*/*'), NVPair('Connection', 'Keep-Alive'), NVPair('User-Agent', 'PTS-HTTP-CLIENT'), ]
    result = HTTPRequest().GET('http://item.taobao.com/item.htm?spm=a217i_h.1303267.1998177633-
0.10.hx5m95&id=12817144777', None, headers)
    PTS.Framework.addHttpCode(result.getStatusCode(), statusCode)

    headers = [ NVPair('Accept', '*/*'), NVPair('Connection', 'Keep-Alive'), NVPair('Content-Type', 'application/x-www-
form-urlencoded'), NVPair('Host', 'www.ptstest.com'), NVPair('User-Agent', 'PTS-HTTP-CLIENT'), ]
    result = HTTPRequest().POST('http://www.ptstest.com/s', 'wd=test', headers)
    PTS.Framework.addHttpCode(result.getStatusCode(), statusCode)

if(not PTS.HttpUtilities.checkResponse(200)):
```

```

PTS.Data.forCurrentTest.success = False

return statusCode

def action4(self):
    statusCode = [0L, 0L, 0L, 0L]

    headers = [ NVPair('Accept', '*/*'), NVPair('Connection', 'Keep-Alive'), NVPair('User-Agent', 'PTS-HTTP-CLIENT'), ]
    result = HTTPRequest().GET('http://item.taobao.com/item.htm?spm=a217i_h.1303267.1998177633-0.10.hx5m95&id=12817144777', None, headers)
    PTS.Framework.addHttpCode(result.getStatusCode(), statusCode)

    headers = [ NVPair('Accept', '*/*'), NVPair('Connection', 'Keep-Alive'), NVPair('Content-Type', 'application/x-www-form-urlencoded'), NVPair('Host', 'www.ptstest.com'), NVPair('User-Agent', 'PTS-HTTP-CLIENT'), ]
    result = HTTPRequest().GET('http://www.ptstest.com/s?wd=test', None, headers)
    PTS.Framework.addHttpCode(result.getStatusCode(), statusCode)

    if(not PTS.HttpUtilities.checkResponse(200)):
        PTS.Data.forCurrentTest.success = False

    return statusCode

def action5(self):

    headers = [ NVPair('Accept', '*/*'), NVPair('Connection', 'Keep-Alive'), NVPair('User-Agent', 'PTS-HTTP-CLIENT'), ]
    result = HTTPRequest().GET('http://www.ptstest.com/s?wd=test', None, headers)

    headers = [ NVPair('Accept', '*/*'), NVPair('Connection', 'Keep-Alive'), NVPair('Content-Type', 'application/x-www-form-urlencoded'), NVPair('Host', 'www.ptstest.com'), NVPair('User-Agent', 'PTS-HTTP-CLIENT'), ]
    result = HTTPRequest().POST('http://www.ptstest.com/s', 'wd=test', headers)

def action6(self):

    headers = [ NVPair('Accept', '*/*'), NVPair('Connection', 'Keep-Alive'), NVPair('User-Agent', 'PTS-HTTP-CLIENT'), ]
    result = HTTPRequest().GET('http://www.ptstest.com', None, headers)

    headers = [ NVPair('Accept', '*/*'), NVPair('Connection', 'Keep-Alive'), NVPair('Content-Type', 'application/x-www-form-urlencoded'), NVPair('Host', 'www.ptstest.com'), NVPair('User-Agent', 'PTS-HTTP-CLIENT'), ]
    result = HTTPRequest().POST('http://www.ptstest.com/s', 'wd=test', headers)

# 编织压测事务
PTS.Framework.instrumentMethod(u'事务1', 'action3', TestRunner)
PTS.Framework.instrumentMethod(u'事务2', 'action4', TestRunner)

```

附录二：TCP Socket示例脚本

```

#!/usr/bin/env python
# -*- coding: utf-8 -*-
# PTS TCP Socket Script Template Version 1.0

```

```
from util import PTS
import socket

class TestRunner:
    # TestRunner对象的初始化方法，每个线程在创建TestRunner后执行一次该方法
    def __init__(self):
        return

    # 主体压测方法，每个线程在测试生命周期内会循环调用该方法
    def __call__(self):
        PTS.Data.delayReports = 1

        statusCode = self.action1()
        PTS.Framework.setExtraData(statusCode)
        PTS.Data.report()
        PTS.Data.delayReports = 0
        # TestRunner销毁方法，每个线程循环执行完成后执行一次该方法
    def __del__(self):

        return
    # 业务函数
    def action1(self):

        statusCode = [0L, 0L, 0L, 0L]

        #以下为socket协议脚本
        #创建TCP Socket
        sk = socket.socket(socket.AF_INET,socket.SOCK_STREAM)
        #连接到服务器，服务器IP及端口号
        sk.connect(('localhost',2000))
        #将发送内容送到服务器
        sendContent = 'This is tcp socket test,{Status:success}'
        sk.send(sendContent);
        #接收服务器返回内容
        RecvContent = sk.recv(1024)
        PTS.Logger.info('Content come from server is:' + RecvContent)

        #关闭连接
        sk.close()
        #socket协议脚本结束

        #抓返回值
        value = PTS.HttpUtilities.valueFromTextBetween(RecvContent,'{Status:','}')

        #判断事务是否成功
        if not value:
            PTS.Data.forCurrentTest.success = False
            PTS.Logger.info('Have no value')
        elif value.lower() == 'success':
            PTS.Data.forCurrentTest.success = True
            PTS.Logger.info('Success flag1:' + value)
        else:
            PTS.Data.forCurrentTest.success = False
            PTS.Logger.info('Success flag2:' + value)

        return statusCode
```

```
# 编织压测事务
PTS.Framework.instrumentMethod(u'事务名', 'action1', TestRunner)
```

附录三：UDP Socket示例脚本

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
# PTS UDP Socket Script Version 1.0

from util import PTS
import socket

class TestRunner:
    # TestRunner对象的初始化方法，每个线程在创建TestRunner后执行一次该方法
    def __init__(self):

    return

    # 主体压测方法，每个线程在测试生命周期内会循环调用该方法
    def __call__(self):
        PTS.Data.delayReports = 1

        statusCode = self.action1()
        PTS.Framework.setExtraData(statusCode)
        PTS.Data.report()
        PTS.Data.delayReports = 0
        # TestRunner销毁方法，每个线程循环执行完成后执行一次该方法
    def __del__(self):

    return
    # 业务函数
    def action1(self):

        statusCode = [0L, 0L, 0L, 0L]

        #创建UDP Socket
        s = socket.socket(socket.AF_INET,socket.SOCK_DGRAM)
        #设置服务器IP地址及端口号
        port = 20000
        host = 'localhost'
        msg = ""0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz
        mnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
        PQRS
        TUVWXYZ0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz
        op
        qrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
        UVWX
        YZ0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz
        tuvxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ
        W
        XYZ0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyz
```

```
klmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ"
OPQRSTUVWXYZ""

#用Connect提高效率
s.connect((host, port))
#将内容发送给服务器

s.send(msg)
#设置接收超时,单位秒
s.settimeout(30)
#接收服务器返回内容
RecvContent,ADDR = s.recvfrom(1024)
PTS.Logger.info('Content come from server:'+RecvContent)

#关闭连接
s.close()

#socket协议脚本结束

#抓返回值
value = PTS.HttpUtilities.valueFromTextBetween(RecvContent,'{Status:','}')

#判断事务是否成功
if not value:
PTS.Data.forCurrentTest.success = False
PTS.Logger.info('Have no value')
elif value.lower() == 'success':
PTS.Data.forCurrentTest.success = True
PTS.Logger.info('Success flag1:'+value)
else:
PTS.Data.forCurrentTest.success = False
PTS.Logger.info('Success flag2:'+value)

return statusCode

# 编织压测事务
PTS.Framework.instrumentMethod(u'事务名', 'action1', TestRunner)
```

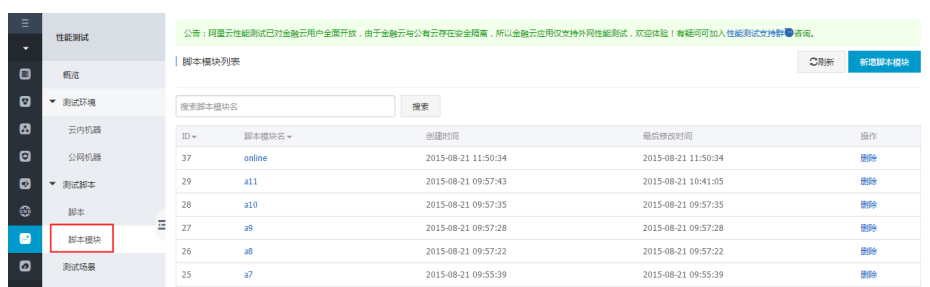
返回性能测试控制台

返回性能测试控制台

脚本模块使用说明

1.编写脚本模块

进入性能测试页面，选择脚本模块标签，点击新增脚本模块：



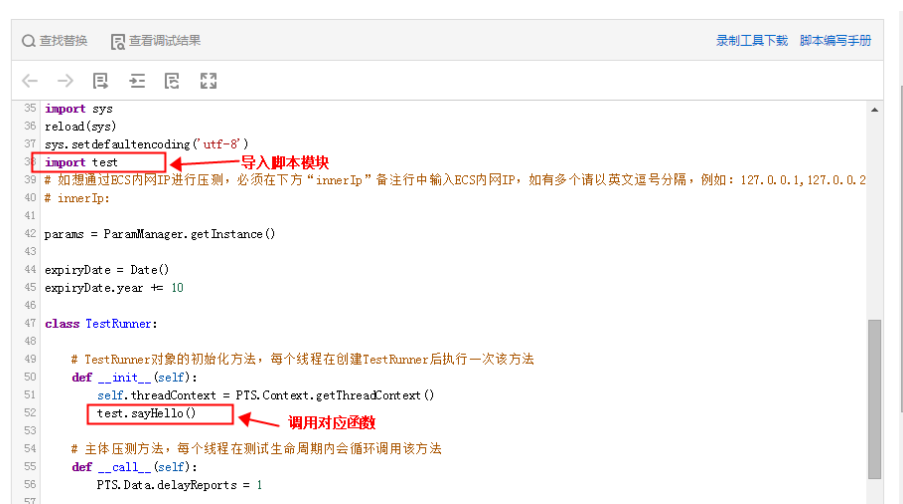
输入脚本模块名，编辑脚本模块，编辑完成后点击保存按钮。

例如定义一个函数sayHello,打印输出Hello World！：



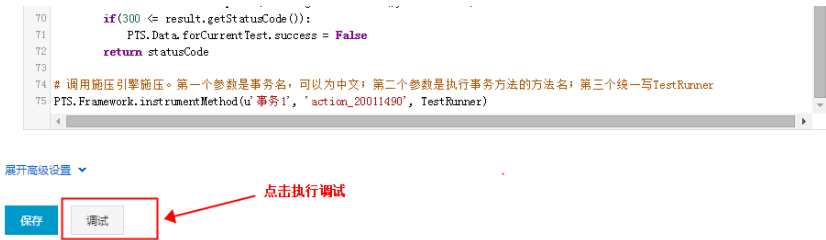
2.编辑脚本

通过手工编写脚本，导入需要的脚本模块，模块名为脚本模块名称，在需要调用模块函数的地方编写代码，调用相应函数：



3.调试运行脚本

编辑完脚本，为了查看脚本是否符合预期，启动任务前，先对脚本进行调试，点击调试按钮，启动调试：

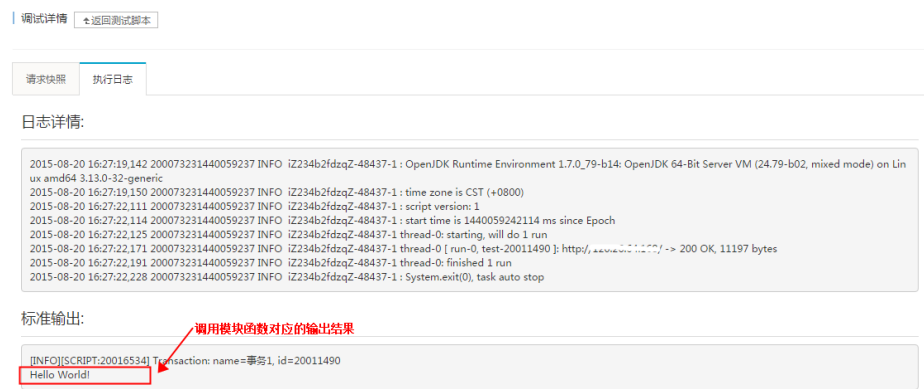


4.查看调试结果

调试成功后，会调整到调试详情页，显示请求快照：



执行日志中，可以看到函数输出的结果：



5.执行压测

方法1：通过快速启动脚本，进行压测

方法2：通过自动义增加场景，增加任务，启动任务，进行压测

注：使用print执行任务，日志中不会打印出对应的结果，可在编辑脚本中使用Logger，详见脚本SDK编写

返回性能测试控制台