目 录 (五十二期·上)

Python 实现 K-means 聚类算法.....01

软件测试中的认知偏差:你受到影响了吗?.....07

使用 Fiddler 进行 APP 弱网测试.....10

Pytest+Allure2+Jenkins 持续集成.....20

数据迁移测试教程：一份完整的指南.....33

Selenium 报错集锦（代码迁移）.....45

JMeter RabbitMQ 采样器 AMQP 详解与实战.....49

如果您也想分享您的测试历经和学习心得，欢迎加入我们~(*^▽^*)

投稿邮箱：editor@51testing.com

Python 实现 K-means 聚类算法

◆作者：咖啡猫

题记：最近有幸参与了一个机器学习的项目，我的主要工作是帮助进行数据预处理，期间用 Python 实现了 K-means 聚类算法，感觉收获很多特此记录下来和大伙儿分享。

一 机器学习项目的主要流程

机器学习项目的主要流程有五步:

1.数据提取

2.数据清洗

3.特征工程

4.训练模型

5.验证模型并优化

之前讲到的 PYTHON 爬虫可以算是第一步数据提取里面的内容，数据提取的作用就是想方设法获取源数据。获得源数据后，由于源数据里可能包含很多的脏数据，比如房屋面积是零、年龄是负数等等情况，所以需要对源数据进行清洗，使数据变得更加整齐有效。然后开始提取数据信息，也就是我们说的特征工程。特征工程是提升模型强度的关键因素，特征选取的越好，模型效果就越好。然后训练模型，就是找出特征 x 和标签 y 之间的函数关系式。最后验证这个函数关系式的效果好不好，并进行优化。

二 聚类算法简介

我这次所做的工作就是第二步数据清洗。需求是这样的：需要将二维坐标系上的点集按距离远近分类，距离近的分成一组。以方便后续算法工程师对各个分组进行特征的提取工作。

为实现目标，常采用的就是 K-means 聚类算法。K-means 聚类算法的思路如下:

1.选择分类数 K，也就是我们希望把数据分成 K 组。一般根据经验来选取，比如衣服就分成 S M L 三类，或者根据聚类结果和 K 的函数式，选择让结果最好的那个 K 值。

2.随机选择每类数据的中心点。中心点的选择对结果影响也很大，如果选取不好就会找到局部最小值，这里一般有两种处理方法：一是多次取平均值，二是采用改进算法。

3.计算数据集中所有点到各个中心点的距离，它们离哪个中心点最近，就把它们划分到哪一类里面去。

4.在每一类中通过求平均值的方法寻找新的中心点

5.循环执行 3、4 步，直到分类结果收敛为止。(即分类结果不变为止)

三 源数据格式

源数据是一个存放在 e 盘下的 txt 文档，其中每一行数据代表点的横坐标和纵坐标，横坐标和纵坐标之间用 TAB 键分隔，文档有多少行就代表有多个点。如下:

四 Python 代码实现

```
import numpy
def loadDataSet(fileName):
dataMat = []
fr = open(fileName)
for line in fr.readlines():
curLine = line.strip().split('\t') #strip 去除首尾的空格，split 以 tab 作为分隔符分割数据。返回一个 list 数据
fltLine =[float(i) for i in curLine]
```

```
    dataMat.append(fltLine)
    return dataMat
    # 计算两个向量的距离，用的是欧几里得距离
    def distEclud(v1, v2):
    return numpy.sqrt(numpy.sum(numpy.square(v1 - v2)))
    # 构建聚簇中心，取 k 个(此例中为 2)随机质心
    def randCent(dataSet, k):
    n = shape(dataSet)[1] #等价于 dataSet.shape[1]，得到数组的列数。同理，dataSet.shape[0]得到数组的行数。shape 函数就是读取矩阵的长度
    centroids = mat(zeros((k,n)))    # 每个质心有 n 个坐标值，总共要 k 个质心
    for j in range(n):
    minJ = dataSet[:,j].min() #第 j 列的最小值
    maxJ = dataSet[:,j].max() #第 j 列的最大值
    rangeJ = maxJ - minJ
    centroids[:,j] = minJ + rangeJ * random.rand(k, 1) #官方文档中给出的用法是:numpy.random.rand(d0,d1,…dn) 以给定的形状创建一个数组，并在数组中加入在[0,1]之间均匀分布的随机样本。
    return centroids
    #def kMeans(dataSet, k, distMeas=distEclud, createCent=randCent):
    def kMeans(dataSet, k):
    m = shape(dataSet)[0] #等价于 dataSet.shape[0] 第一维的长度，即行数
    clusterAssment = mat(zeros((m, 2)))  # 用于存放该样本属于哪类及质心距离，第一列是该数据所属中心点，第二列是该数据到中心点的距离
    centroids = randCent(dataSet, k)
    clusterChanged = True  #判断聚类是否已经收敛，当 clusterChanged 的值为 False 时已经收敛，跳出 While 循环。
    while clusterChanged:
    clusterChanged = False
    for i in range(m):  # 每个数据点分配到离它最近的质心，range（start,stop,step）用于生成一个整数列表
    minDist = inf #inf 代表正无穷   -inf 代表负无穷
    minIndex = -1
    for j in range(k):
    distJI = distEclud(numpy.array(centroids[j, :]), numpy.array(dataSet[i, :])) #求第 j 行的中心点到第 i 行的数据之间的距离
```

```
        if distJI < minDist:
        minDist = distJI;
        minIndex = j
        if clusterAssment[i, 0] != minIndex:
        clusterChanged = True
        clusterAssment[i, :] = minIndex, minDist ** 2 #   x**y 返回 x 的 y 次幂   x//y 取 x 除以 y 的整数部分
        print(centroids)
        for cent in range(k):   #重新计算中心点
        ptsInClust = dataSet[nonzero(clusterAssment[:, 0].A == cent)[0]]   # 筛选出第一列等于 cent 的所有数据，即找到属于这个聚类的所有数据点。   nonzero 函数是 numpy 中用于得到数组 array 中非零元素的位置（数组索引）的函数
        centroids[cent, :] = mean(ptsInClust, axis=0)   # mean 是求平均值的函数，
        # numpy.mean(a, axis, dtype, out，keepdims )   axis 不设置值，对 m*n 个数求均值，返回一个实数
        #axis = 0: 压缩行，对各列求均值，返回 1* n 矩阵 axis =1 ： 压缩列，对各行求均值，返回 m *1 矩阵
        return centroids, clusterAssment
        def show(dataSet, k, centroids, clusterAssment):   #绘图展示，这块还没看
        from matplotlib import pyplot as plt
        numSamples, dim = dataSet.shape
        mark = ['or', 'ob', 'og', 'ok', '^r', '+r', 'sr', 'dr', '<r', 'pr']
        for i in range(numSamples): #画样本点，属于同一中心的用一种标记
        markIndex = int(clusterAssment[i, 0])
        plt.plot(dataSet[i, 0], dataSet[i, 1], mark[markIndex])
        mark = ['Dr', 'Db', 'Dg', 'Dk', '^b', '+b', 'sb', 'db', '<b', 'pb']
        for i in range(k): #画中心点
        plt.plot(centroids[i, 0], centroids[i, 1], mark[i], markersize=12)
        plt.show()
        def main():
        dataMat =mat(loadDataSet('E:/test.txt')) #mat 转换成矩阵操作
        myCentroids, clustAssing = kMeans(dataMat, 2) #函数返回一个元组，元组的括号可以省略
        # print(myCentroids)
        show(dataMat, 2, myCentroids, clustAssing)
```

```
if __name__ == '__main__':
main()
```

代码要点精讲

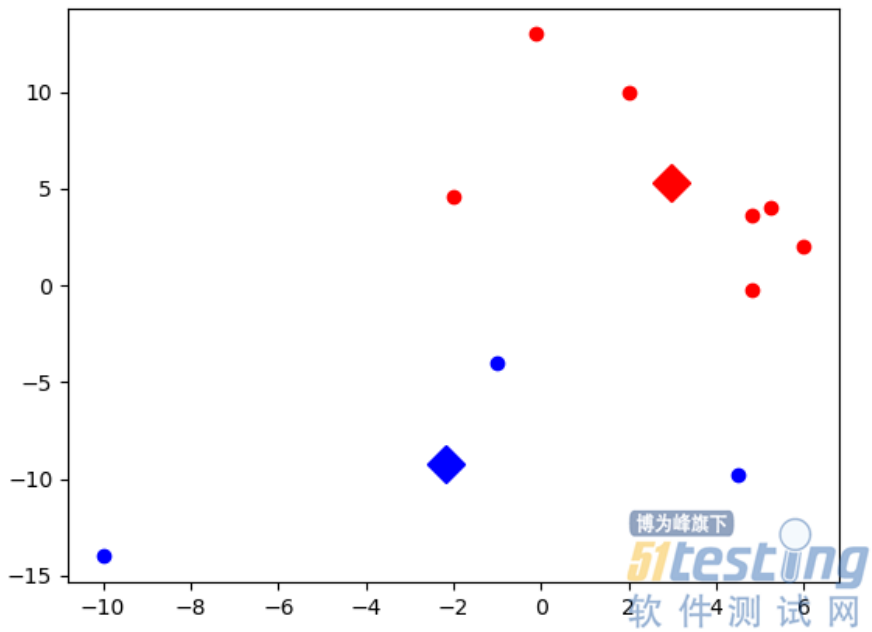
main（）函数：这是程序的起点，它先对用 loadDataSet 函数对源数据做加载，再要用 kMeans（）做聚类，最后用 show（）将聚类结果以图表形式展示出来。

loadDataSet（）函数：用来加载源数据，先用 open 函数打开源文件，然后逐行读取源文件的内容，每行内容用 TAB 作为分隔符，每行内容形成一个列表，附加到 dataMat 中，最后返回一个对象是列表的列表 dataMat。比如这里的 txt 数据形成一个 10*2 的矩阵给了 dataMat。

kMeans()函数：分类结果存放在 clusterAssment 中，中心点存放在 centroids 中，当标志位 clusterChanged 为 false 时，表示聚类结束。While 循环表示每次聚类，一次 WHILE 里有双重 for 循环，表示每次聚类时，都要计算所有数据点到中心点的距离，数据到哪个中心点距离最短，就划分到哪类里面去。划分完毕后重新计算中心点。

Show（）函数：用于聚类结果的展示。导入 matplotlib 库，应用其中的 plot 函数绘图，先绘制样本点再绘制中心点。属于同一中心点的样本点用同种标记。

六、聚类结果



从图中我们可以看到十个样本点被分成了两类。红色的是一类，蓝色的是一类，这

个结果符合我们最初的预期。当数据量加大的时候，也有同样的效果，这里就不赘述了。

后记:

曾经作为测试小白的我有很长一段时间也认为测试不需要懂代码，特别是功能测试只需要懂业务，然后会跑用例就够了。但是当真正遇到一些稍微高大上一点的项目的时候，才知道测试光懂业务还远远不够，知其然还要知其所以然，才能更好地把控软件产品质量。通过这次“兼职”数据处理工程师，在我明白这个聚类算法的逻辑后，才能思考怎么测试才能更好了把控产品质量，比如说采用大数据量、采用一些垃圾数据做干扰、采用让它收敛到局部最小的 K 值等等。。

测试路上没有最好，只有更好。与各位同仁共勉!

软件测试中的认知偏差:你受到影响了吗?

◆ 译者：侯 峥

为了保证质量，测试领域的技术正在飞速的发展。

“持续集成、数字转换、生命周期自动化、将质量移以最小化成本”等一些神奇的词语正在推广普及。虽然我们谈到了这些问题，但我们仍然可以听到“为什么以及如何遗漏了缺陷”这一根本问题，并且这些问题仍然没有得到回答。

与此同时，感觉到最明显的缺陷似乎已经被忽略了。

为什么呢?

虽然我们都希望认为自己很有逻辑，有条理，理性，悲催的是，我们都受到认知偏见的影响，它影响我们日常生活和专业工作的思维方式。

简述认知偏见

根据维基百科——“认知偏差是一种在判断中偏离规范或理性的系统性模式。个体通过对输入的感知创造出自己的“主观社会现实”。

一个人对社会现实的建构，而非客观的输入，可能决定他在社会世界中的行为。因此，认知偏差有时会导致感知扭曲、错误判断、不合逻辑的解释或不理性。”

好吧，这是一个很好的定义，但是它如何影响思维以及它对测试领域的测试人员意味着什么呢?

好了，当测试人员开始任何测试时，他们已经被自己的偏见所影响——根据要测试的内容、可能出现的潜在缺陷、开发人员、程序的进程等等来构建思维和判断，还有其他因素。

对我们来说，了解不同类型的偏见是非常必要的，这样我们就能更清楚地意识并深

入地思考如何有效地管理它们。

软件测试中要了解的认知偏差类型

我们需要在自身内部寻找不同的认知偏见，下面将详细解释其中一些偏见。

1) 雷同的偏见

人们很容易根据相似情况的相似程度来做出判断。

例如，作为测试人员，我们常常认为 web 应用程序会有类似的错误，而客户机-服务器应用程序会有类似的一系列错误。

作为测试人员，我们自然会根据项目的性质来寻找类似的错误。不幸的是，由于这种本性，有时我们会因为固定思维而忽略最明显的问题。

2) 正向的偏见

这是一种我们的大脑拒绝思考其他选择的行为。

这意味着，测试人员倾向于只验证预期的行为，从而忽略了负面测试。

在编写测试用例时，我们倾向于用正常流程覆盖所有需求，而忽略了异常流程，因为并不是所有异常流程都在需求中被特别提及。

它们在需求中是隐性的，实际上也不可能记录所有的用户行为。

3) 认知的偏见

这是一种通过确认我们的认知和假设来搜索和解释信息的倾向行为。

通常，在测试中，我们肯定会遇到这样的情况，即我们认为某个特定开发人员的代码在默认情况下会比其他开发人员的代码有更多的缺陷，因此我们将花费大量时间测试他开发的模块。在这些认知的影响下，会增加遗漏其他人员开发模块中的缺陷的风险。

4) 从众效应

“从众效应”指的是人们传播的行为或观点。

当相信某件事的人达到一定数量时，另一个人也相信同一件事的可能性就会自动增加。这在我们的日常生活中很常见。

最常见的例子是当我们购买一些产品。 我们通常会根据别人的想法来选择产品，

而不是仅根据自己的判断选择

在测试领域中也显示了完全相同的行为。在我们的同行群体中，如果有人认为某个模块是无缺陷的，我们会不自觉地倾向于相信某个模块是无缺陷的，并且我们在验证过程中对该模块的关注程度会大大降低。

5) 专注测试某些功能而忽略其他功能点

这是一种行为，而测试人员在不寻找缺陷时，往往会忽略最明显的缺陷。

想象一下这样一种情况，你让一群人数一数有多少人穿了某种特定的衣服，你可以观察到人们会全神贯注地数，却看不到周围其他任何重要或重要的东西。

联想到测试，例如在一个增强项目中，其中一个屏幕是新开发的，那么测试人员自然倾向于更多地关注新开发的屏幕，而忽略其他关键集成功能。

6) 消极的偏见

消极偏见是人类倾向于关注不好的经历而不是好的。

消极偏见在测试领域的影响是什么？如果测试人员曾经漏测过，那么是他是很难判断什么时候结束测试的。他们不能确定什么时候能够确认产品没有缺陷。这个是产品经理或者项目经理决定产品是否上线的要根据测试经理的建议的主要原因之一。

结论

希望您对软件测试中的认知偏见有更好的了解，包括它的影响，以及如何消除这种影响?

需要注意的一个重要事实是，我们对自己的偏见是看不见的，而我们可以识别他人的偏见(这本身就是盲点认知偏见)。然而，我们可以避免偏见，在很大程度上，我们可以在任何需要的地方谨慎的思考来避免问题。

❖ 拓展学习

■自动化测试框架开发，职场精英必修课>>

http://testing51.mikecrm.com/govLasx

使用 Fiddler 进行 APP 弱网测试

◆作者：枫　叶

一、安装 Fiddler

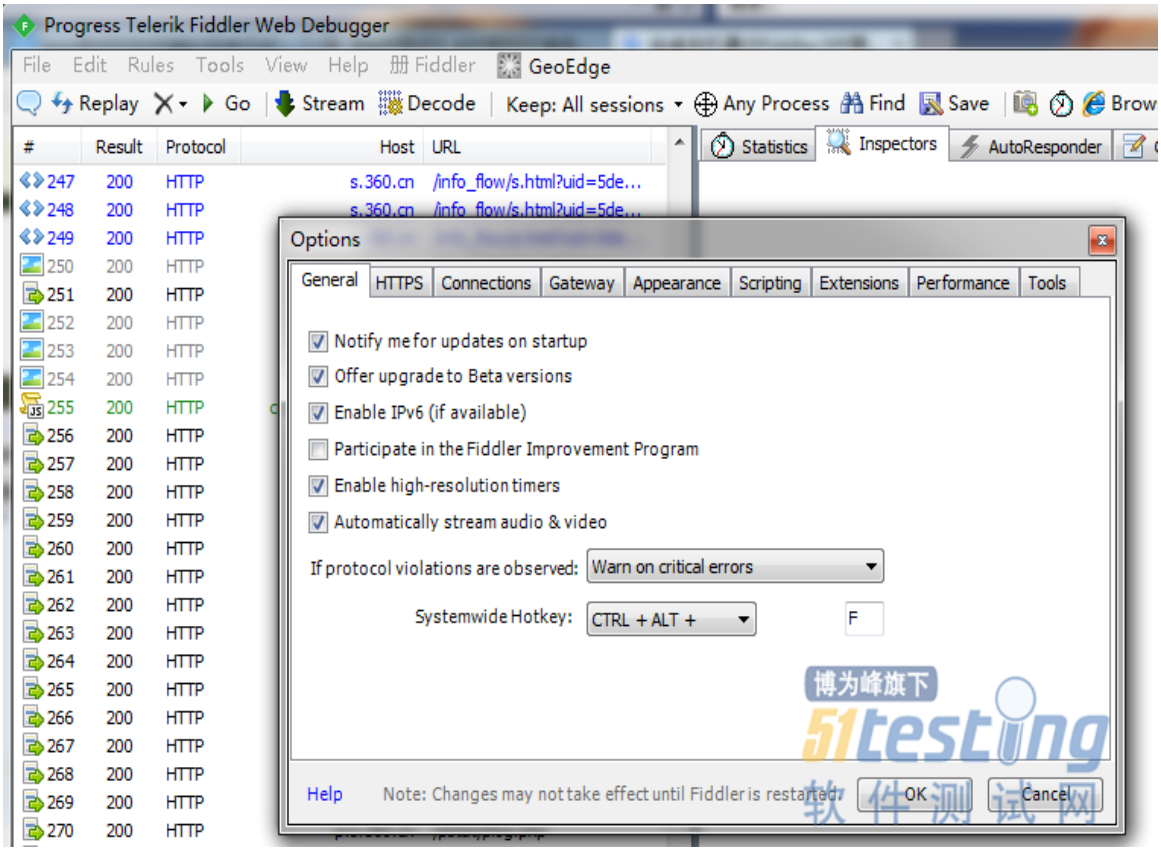
网上说要先安装.NET Framwork4，应该是由于本机已装，所以在安装 Fiddler 时并没有相关提示。

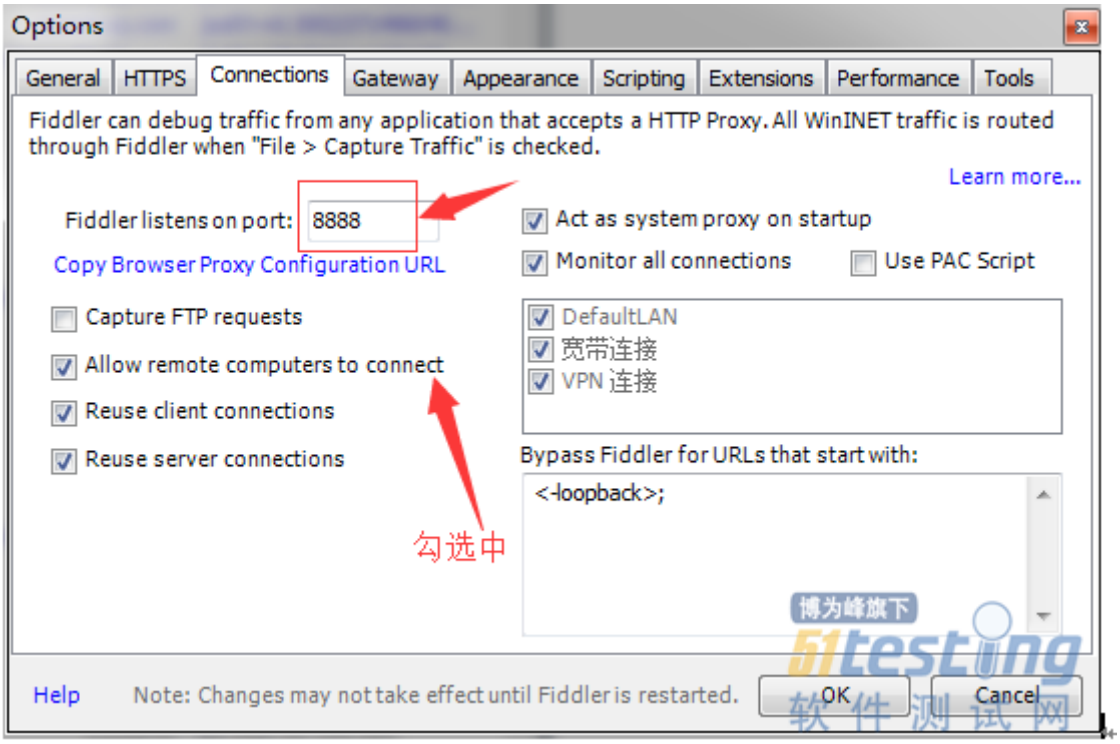
Fiddler 安装包：https://www.telerik.com/download/fiddler/fiddler4

二、Fiddler 通过代理连上手机

首先电脑和手机要使用同一个无线网。

1、Fiddler 工具->选项



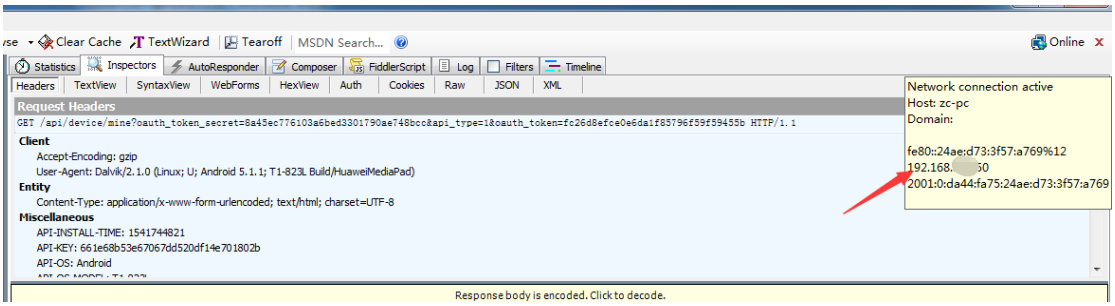


注意：fiddler 监听端口设置为：8888

2、需要安装 fiddler 证书

使用手机浏览器访问 http://[电脑 IP 地址]:[fiddler 设置的端口号]，即可以下载 fiddler 的证书并安装。

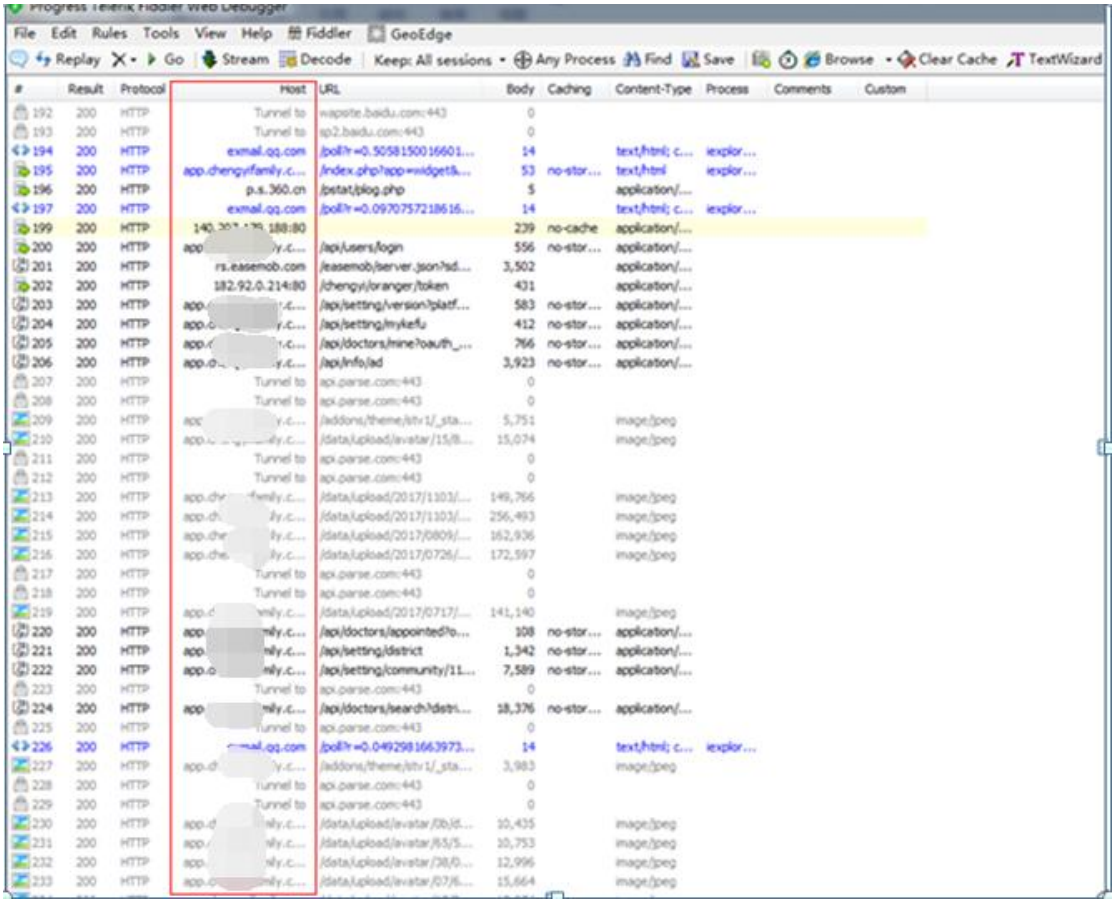
查看电脑 IP 的方法：直接在 cmd 下 ipconfig，或者鼠标滑过 fiddler 的 online 也可以看到 IP 地址。



3、打开手机设置 WLAN

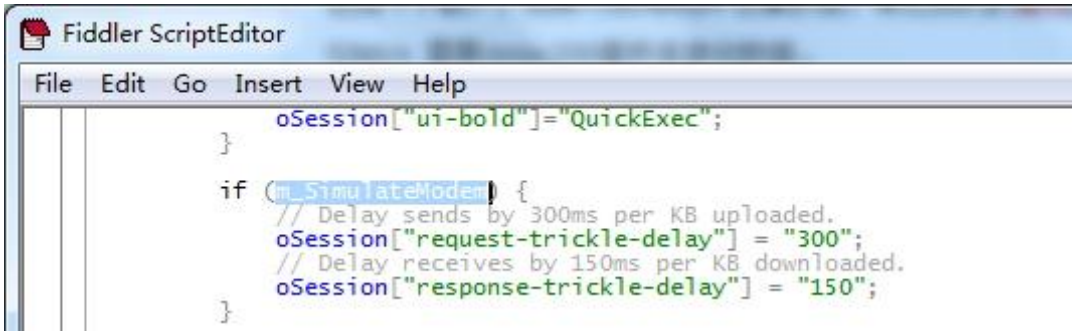
长按 wifi 名称，选择“管理网络设置”，勾选【显示高级设置】，代理设置选择【手动】，输入电脑的 IP 地址和端口，端口为 fiddler 中设置的 8888

4、手机打开浏览器网页，或者要测试的 APP，fiddler 软件里会获取相关的地址



5、修改参数模拟网速

利用 fiddler 通过代理连接上手机之后，进入 Fiddler->Rules->Customize Rules，点击弹出的 CustomRules.js 文件，找到 m_SimulateModem，也就是下面的这段:

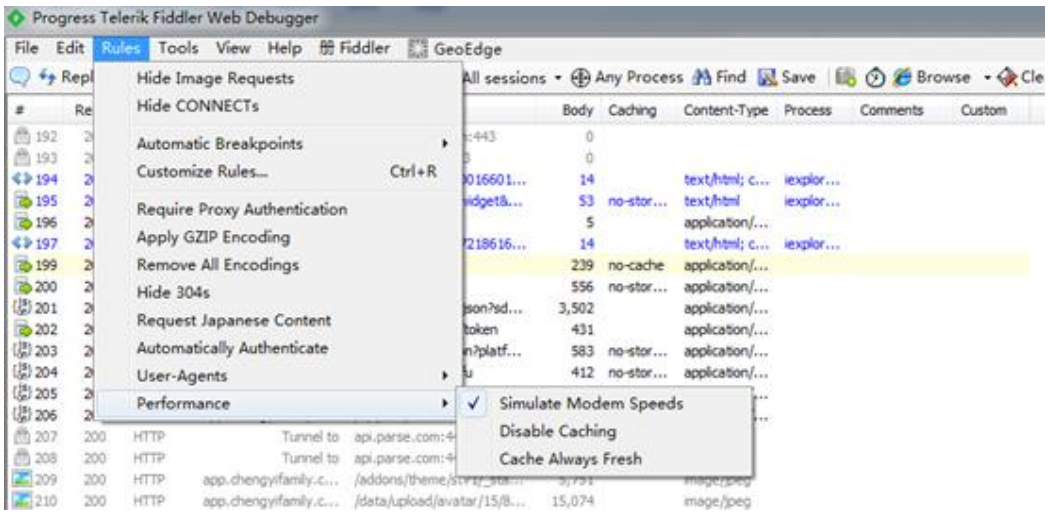


修改代码如下:

```
if (m_SimulateModem) {
    // Delay sends by 300ms per KB uploaded.
    oSession["request-trickle-delay"] = "2962";
    // Delay receives by 150ms per KB downloaded.
    oSession["response-trickle-delay"] = "833";
}
```

6、确定设定的参数

设置完之后，再勾选 Rules -> Performances -> Simulate Modem Speeds

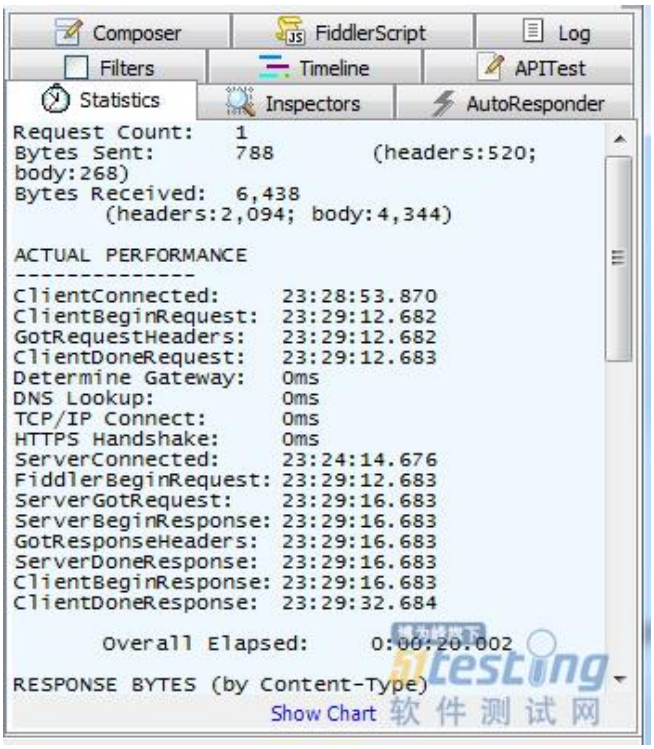


7、注意停掉 PC 与手机上面上网的应用

把 PC 与手机上面上网的进程杀掉，如果上网的应用太多了，那网速肯定也会受到影响，这样出来的报告，就会不准确。

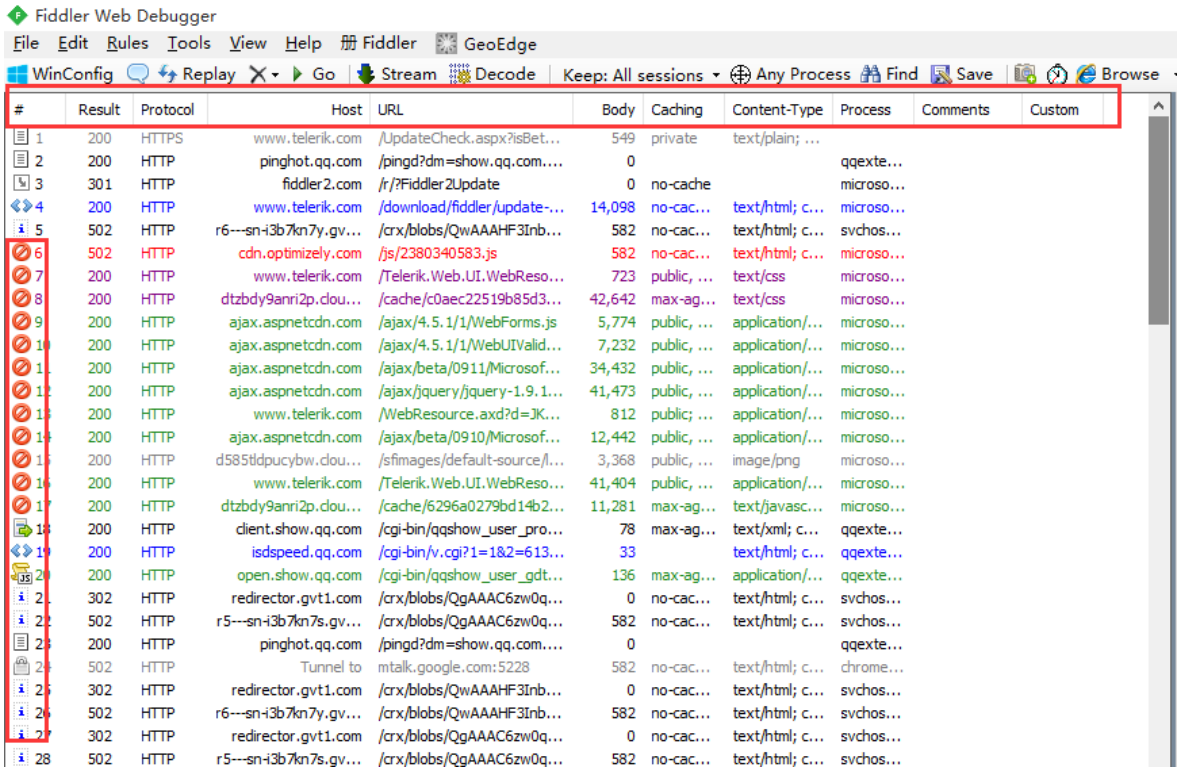
8、进行抓包

点击抓包数据中的一条记录，在右侧的 statistics 中就会显示当前界面相应数据。bytes Sent 是指发送的请求数，Bytes Received:返回的数据量，Overall Elapsed:总耗时。



三. Fiddler 工具使用说明

1、Fiddler 开始工作了，抓到的数据包就会显示在列表里面，以下总结这些是什么意思:



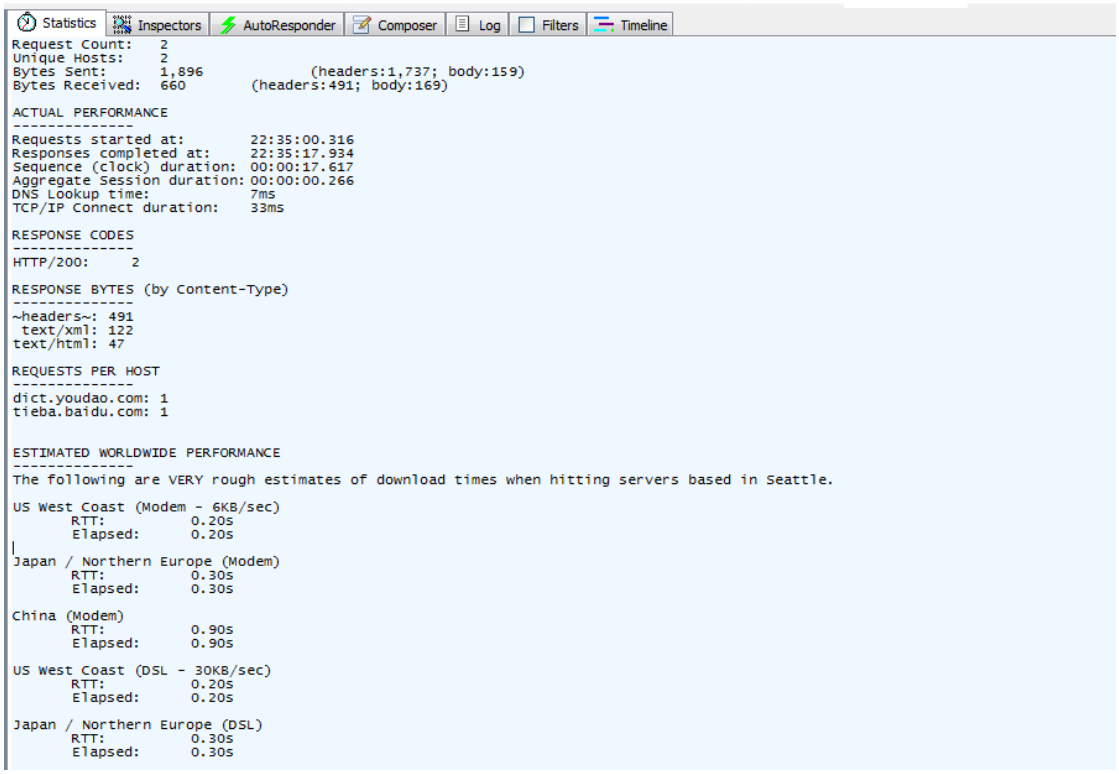
#: 抓取 HTTP Request 的顺序，从 1 开始，以此递增

- Result: HTTP 状态码
- Protocol: 请求使用的协议，如 HTTP/HTTPS/FTP 等
- Host: 请求地址的主机名
- URL: 请求资源的位置
- Body: 该请求的大小
- Caching: 请求的缓存过期时间或者缓存控制值
- Content-Type: 请求响应的类型
- Process: 发送此请求的进程，进程 ID
- Comments: 允许用户为此回话添加备注
- Custom: 允许用户设置自定义值。

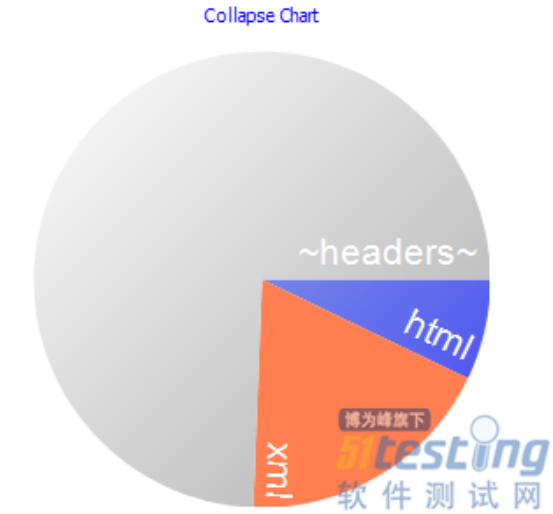
2、Fiddler 的统计选项卡中显示了当前 Session 的基本信息，在选项卡的最上方显示的是文本信息，最下方是个饼图。使用 Statistics 页签，用户可以通过选择多个会话来得来这几个会话的总的信息统计，比如多个请求和传输的字节数。

选择第一个请求和最后一个请求，可获得整个页面加载所消耗的总体时间。从条形图表中还可以分别出哪些请求耗时最多，从而对页面的访问进行访问速度优化。

如下所示:



饼图如下:



统计选项卡的一些信息含义如下解释:

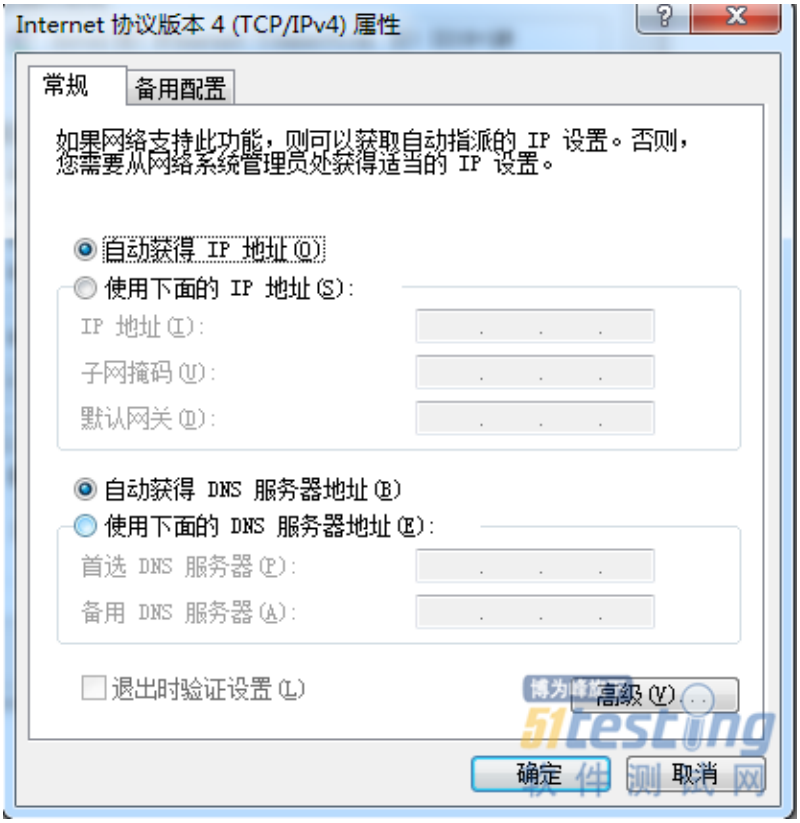
- Request Count: 选中的 session 数;
- Unique Hosts: 流量流向的独立目标主机数。如果所有选中的流量都发送到相同的服务器上，则不会显示该字段。
- Bytes sent: HTTP 请求头和请求体中向外发送的字节总数。后面括号中分别给出了头和 body 各自的字节数。
- Bytes received: HTTP 请求头和请求体中接收到的所有字节数。在全部计数后面的括号中给出了请求头和请求体各自的字节数。
- Requests started at: Fiddler 接收到的第一个请求的第一个字节的时间点。
- Responses completed at: Fiddler 发送到客户端的最后一个响应的最后一个字节的时间点。
- Sequence(clock) duration: 第一个请求开始到最后一个响应结束之间的 “时钟时间”。
- Aggregate session duration: 所有选中的 session 从请求到响应之间的时间的和。
- DNS Lookup time: 所有选中的 session 解析 DNS 所花费的时间的总和。
- TCP/IP Connect duration: 所有选中 session 建立 TCP/IP 连接所花费的时间总和。
- HTTPS Handshake duration: 所有选中 session 在 HTTPS 握手上所花费的时间总和。
- Response Codes: 选中 session 中各个 HTTP 响应码的计数。
- Response Bytes by content-type: 选中 session 中响应的各个 Content-Type 的字节数。
- Estimated Performance: 选中的流量在不同语种(local)地区和连接方式下所需时间的初步估计。

四、实际操作 Fiddler 遇到的问题及解决办法

问题: 由于办公区域路由改造，获取 IP 地址从自动改成固定 IP，使得 Fiddler 连接

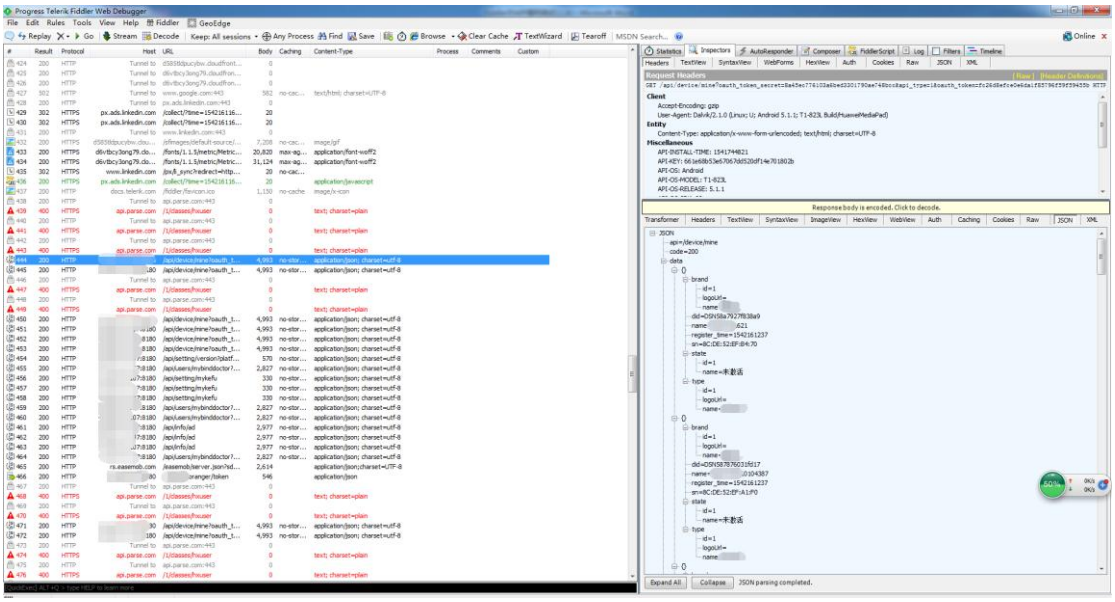
手机的时候，手机连不上网络，无法获取 APP 请求。

办法：1、将固定 IP 改为“自动获取 IP 地址”

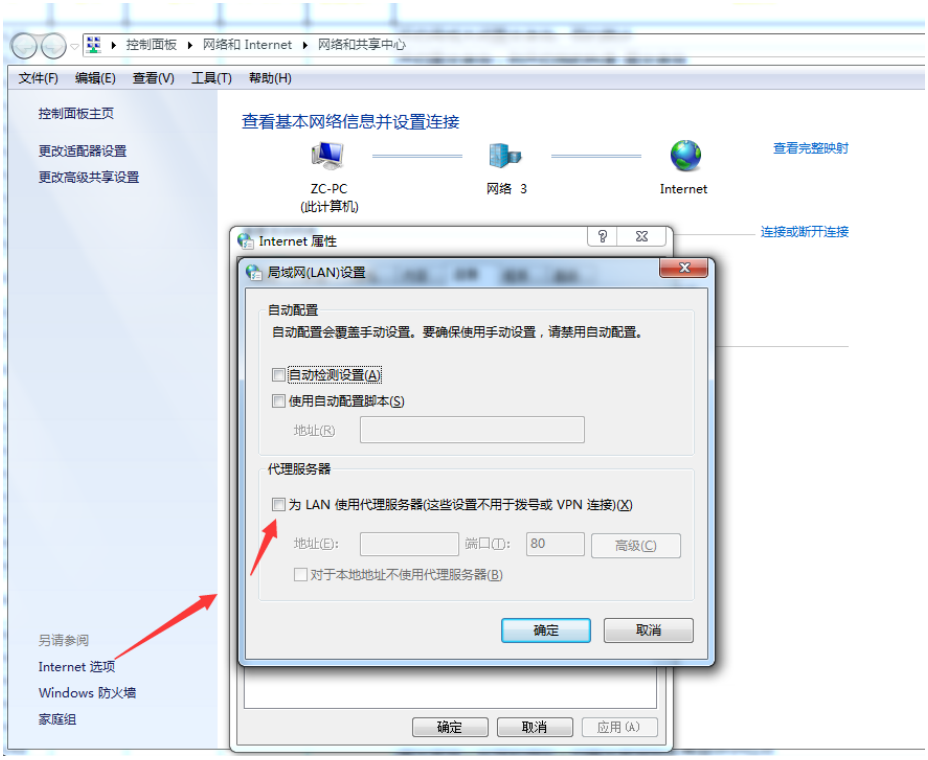


2、查看电脑 IP 地址，CMD->IP config

3、手机的代理输入刚才查看到的电脑 IP。这样可以解决 fiddler 连接手机抓取不到手机 app 的问题了！



注意：需要检查代理服务器设置为不使用代理。打开 Internet 选项——连接——局域网设置，取消代理服务器的勾选。


控制面板 ▸ 网络和 Internet ▸ 网络和共享中心
文件(F) 编辑(E) 查看(V) 工具(T) 帮助(H)
控制面板主页
更改适配器设置
更改高级共享设置
查看基本网络信息并设置连接
查看完整映射
ZC-PC
(此计算机)
网络 3
Internet
连接或断开连接
Internet 属性
局域网(LAN)设置
自动配置
自动配置会覆盖手动设置。要确保使用手动设置，请禁用自动配置。
自动检测设置(A)
使用自动配置脚本(S)
地址(R)
代理服务器
为 LAN 使用代理服务器(这些设置不用于拨号或 VPN 连接)(X)
地址(E):
端口(T):
80
高级(C)
对于本地地址不使用代理服务器(B)
确定
取消
确定
取消
应用(A)
另请参阅
Internet 选项
Windows 防火墙
家庭组

Pytest+Allure2+Jenkins 持续集成

◆ 作者：晴 空

一：环境配置

安装 pytest

pytest 官网地址：https://github.com/pytest-dev/pytest/

pytest 第三方插件：http://plugincompat.herokuapp.com/

前置条件：已安装 Python 环境

使用 pip 安装 pytest，在 Dos 窗口中执行 pip install -U pytest 命令：

```
C:\Users\DYK>allure generate  D:\pytestreoprt -o d:\report
Report successfully generated to d:\report
C:\Users\DYK>pip install  -U pytest
Requirement already up-to-date: pytest in d:\python37\lib\site-packages (4.0.2)
Requirement already satisfied, skipping upgrade: setuptools in d:\python37\lib\site-packages (from pytest) (39.0.1)
Requirement already satisfied, skipping upgrade: six>=1.10.0 in d:\python37\lib\site-packages (from pytest) (1.11.0)
Requirement already satisfied, skipping upgrade: pluggy>=0.7 in d:\python37\lib\site-packages (from pytest) (0.8.0)
Requirement already satisfied, skipping upgrade: attrs>=17.4.0 in d:\python37\lib\site-packages (from pytest) (18.2.0)
Requirement already satisfied, skipping upgrade: atomicwrites>=1.0 in d:\python37\lib\site-packages (from pytest) (1.2.1
)
Requirement already satisfied, skipping upgrade: more-itertools>=4.0.0 in d:\python37\lib\site-packages (from pytest) (4
.3.0)
Requirement already satisfied, skipping upgrade: colorama; sys_platform == "win32" in d:\python37\lib\site-packages (fro
m pytest) (0.4.1)
Requirement already satisfied, skipping upgrade: py>=1.5.0 in d:\python37\lib\site-packages (from pytest) (1.7.0)

C:\Users\DYK>
```

安装 Allure2

前置条件：已部署 java 环境

allure 是一个轻量级的，灵活的，支持多语言，多平台的 report 框架

Allure2 官网地址：https://github.com/allure-framework/allure2

Win10 安装 allure2：需要在 Power Shell 窗口中执行 scoop install allure 命令：

Win 键+X 调出 Power Shell 窗口

Power Shell 窗口执行如下命令（先安装 scoop）:

iex (new-object net.webclient).downloadstring('https://get.scoop.sh')

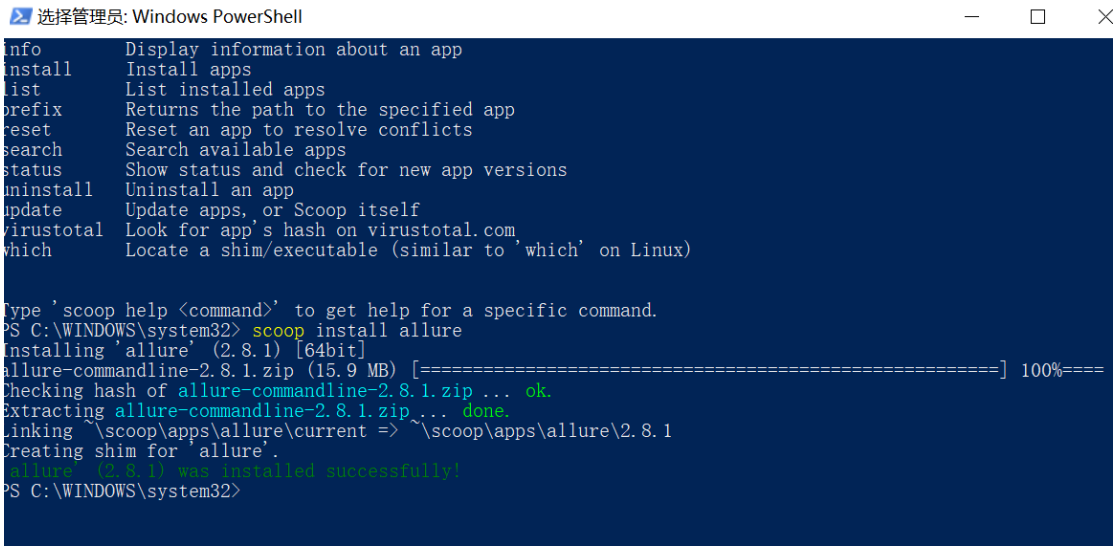
执行后，使用 scoop 命令查看是否正确安装（下图说明安装正确）:

```
PS C:\WINDOWS\system32> scoop
Usage: scoop <command> [<args>]

Some useful commands are:

alias       Manage scoop aliases
bucket      Manage Scoop buckets
cache       Show or clear the download cache
checkup     Check for potential problems
cleanup     Cleanup apps by removing old versions
config      Get or set configuration values
create      Create a custom app manifest
depends     List dependencies for an app
export      Exports (an importable) list of installed apps
help        Show help for a command
home        Opens the app homepage
info        Display information about an app
install     Install apps
list        List installed apps
prefix      Returns the path to the specified app
reset       Reset an app to resolve conflicts
search      Search available apps
status      Show status and check for new app versions
uninstall   Uninstall an app
update      Update apps, or Scoop itself
virustotal  Look for app's hash on virustotal.com
which       Locate a shim/executable (similar to 'which' on Linux)
```

Power Shell 窗口执行 scoop install allure 命令（如下图安装成功）:



注意：win7 下要执行 pip install pytest-adaptor-allure 安装 allure 的适配器！

二：Pytest 的参数详解

2.1：--collect-only

使用--collect-only 选项可以展示在给定的配置下哪些测试用例会被执行。

```
Kevinqingkong:~ mc$ cd /Users/mc/Desktop/CRMAuto-New/TestSuites
Kevinqingkong:TestSuites mc$ pytest --collect-only
============================= test session starts ==============================
platform darwin -- Python 3.7.0, pytest-4.0.1, py-1.7.0, pluggy-0.8.0
rootdir: /Users/mc/Desktop/CRMAuto-New/TestSuites, inifile:
plugins: xdist-1.25.0, rerunfailures-5.0, metadata-1.7.0, html-1.19.0, forked-0.2, allure-pytest-2.5.4
collected 4 items
<Package '/Users/mc/Desktop/CRMAuto-New/TestSuites'>
  <Module 'test_create_invoice.py'>
    <Function 'test_create_new_invoice'>
  <Module 'test_create_payback.py'>
    <Function 'test_create_new_contract'>
  <Module 'test_create_payment.py'>
    <Function 'test_create_new_payment'>
  <Module 'test_create_purchase.py'>
    <Function 'test_create_new_purchase'>

============================ no tests ran in 0.11 seconds ============================
Kevinqingkong:TestSuites mc$
```

2.2：-k

-k 选项允许我们使用表达式指定希望执行的测试用例。

```
Kevinqingkong:TestSuites mc$ pytest -k "pay" --collect-only
============================= test session starts ==============================
platform darwin -- Python 3.7.0, pytest-4.0.1, py-1.7.0, pluggy-0.8.0
rootdir: /Users/mc/Desktop/CRMAuto-New/TestSuites, inifile:
plugins: xdist-1.25.0, rerunfailures-5.0, metadata-1.7.0, html-1.19.0, forked-0.2, allure-pytest-2.5.4
collected 4 items / 2 deselected
<Package '/Users/mc/Desktop/CRMAuto-New/TestSuites'>
  <Module 'test_create_payback.py'>
    <Function 'test_create_new_contract'>
  <Module 'test_create_payment.py'>
    <Function 'test_create_new_payment'>

============================ 2 deselected in 0.05 seconds ============================
Kevinqingkong:TestSuites mc$
```

2.3: -m

-m(marker)用于标记测试并分组以便快速选择并执行测试用例。

使用的前提条件是必须使用@pytest.mark.marker_name 标记测试用例

-m 可以使用多个 marker_name(标记名称)，当然它也支持 and not or 这些规则。

2.4: -x(--exitfirst)

正常情况下，pytest 会运行每个收集到的测试用例。如果某个测试函数被断言失败或者触发了外部异常，则该测试用例的运行就会终止，pytest 将其标记为失败后会继续下一个测试用例。通常来说这是我们期望的运行模式。但是在 debug 时，我们会希望失败时立即终止整个会话，此时，-x 选项就可以满足我们的需求了。

2.5: --maxfail=num

-x 选项的特点是一旦遇到失败就会立即终止会话。如果我们允许 pytest 失败几次后再停止，那么就果断使用--maxfail 选项吧。

例子: pytest --maxfail=3(pytest 执行过程中失败 3 次后终止会话)

2.6: -s(--capture=method)

-s 选项允许终端在测试过程中输出某些结果，包括任何附和标准的输出流信息。-s 等价于--capture=no。正常情况下，所有的测试输出都会被捕获。

2.7: --lf(--last-failed)

当一个或多个测试用例失败时，如果我们希望定位到最后一个失败的测试用例重新执行，此时，可以使用--lf 选项。

2.8: --ff(--failed-first)

--ff 和--lf 选项的作用差不多，不同之处在于--ff 会运行完剩余的测试用例。

2.9: -v(--verbose)

使用-v 选项，输出的信息会更详细。

2.10: -q(--quit)

-q 选项和-v 选项作用相反，它会简化输出信息。

2.11: -l(--showlocals)

使用-l 选项，失败的测试用例由于被堆栈追踪，所以局部变量及其值都会显示出来。

2.12: --tb=style

--tb 选项觉得了捕获到失败时输出信息的显示方式。

推荐的 style 类型有 short，line，no。

Short 模式：仅输出 assert 的一行一级系统判定内容。

line 模式只使用一行输出显示所有的错误信息。

no 模式则直接屏蔽全部回溯信息。

2.13: --duration=N

--duration=N 选项可以加快测试节奏。它不关心测试如何执行，只统计测试过程中哪几个阶段是最慢的，展示最慢的 N 个阶段，耗时越长越靠前。如果指定 duration=0，将所有阶段按耗时从长到短排序后显示。

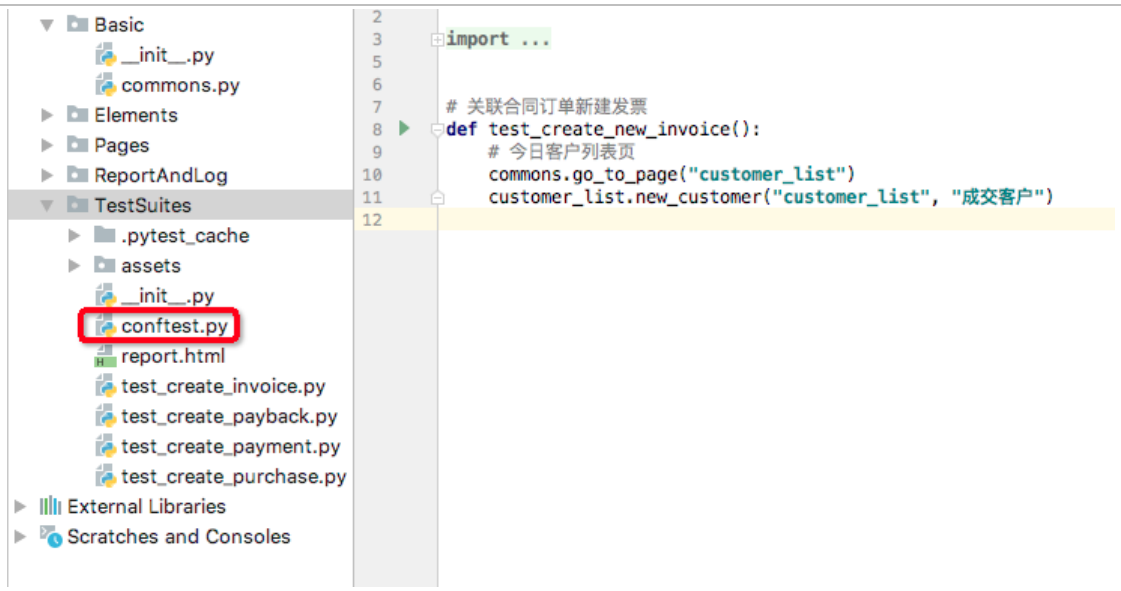
三：Pytest 的精髓 Fixture

fixture 是在测试函数运行前后，又 pytest 执行的外壳函数。fixture 中的代码可以定制，满足多变的测试需求，包括定义传入测试中的数据集，配置测试前系统的初始状态，为批量测试提供数据源等。

@pytest.fixture()装饰器用于声明函数是一个 fixture。如果测试函数的参数列表中包含 fixture 名，那么 Pytest 会检测到，并在测试函数运行之前执行该 fixture。fixture 可以完成任务，也可以返回数据给测试函数。

3.1: 通过 conftest.py 共享 fixture

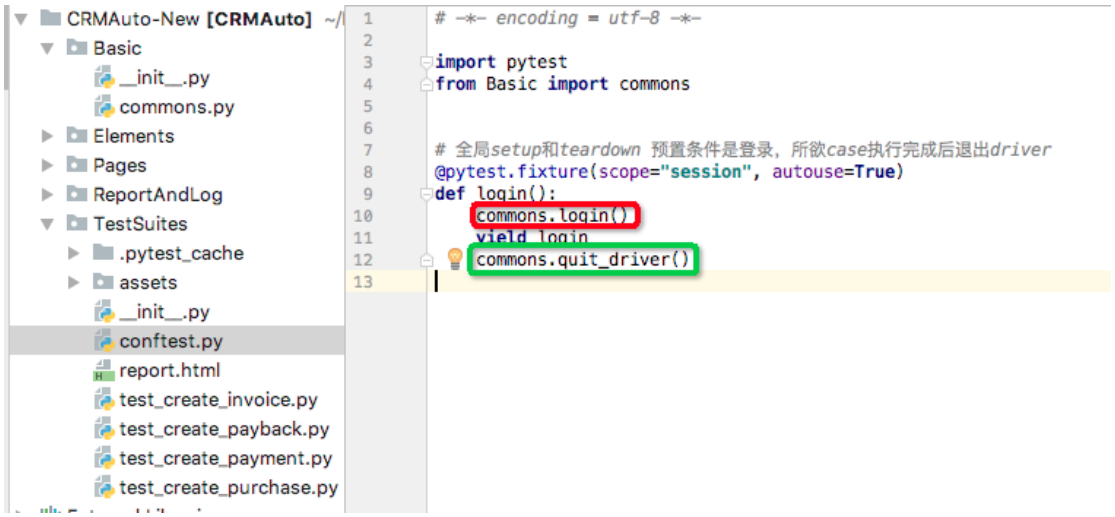
fixture 可以放在单独的测试文件中。如果希望多个测试文件共享 fixture，可以在用例层目录下新建一个 conftest.py 文件，将 fixture 放在其中。



3.2: 使用 fixture 执行预置&销毁逻辑

fixture 函数会在测试函数之前执行，但如果 fixure 函数包含 yield，那么系统会在 yield 处停止，转而运行测试函数，等测试函数执行完毕后再回到 fixture，执行 yield 后面的代码。

因此，可以将 yield 之前的代码视为配置过程(setup)，将 yield 之后的代码视为清理过程(teardown)。物流测试过程中发生了什么，yield 之后的代码都会被执行。



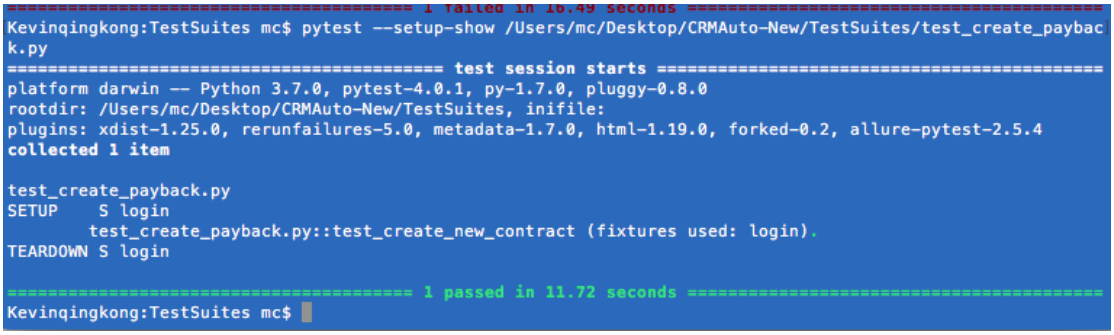

```
# -*- encoding = utf-8 -*-
import pytest
from Basic import commons
# 全局 setup 和 teardown 预置条件是登录，所欲 case 执行完成后退出 driver
@pytest.fixture(scope="session", autouse=True)
def login():
```

```
commons.login()
yield login
commons.quit_driver()
```

上面代码的例子是所有用例执行前先执行系统的登录操作，所有用例执行完成后退出 driver。

3.3: 使用--setup-show 回溯 fixture 的执行过程

我们编写 fixture 时如果希望看到测试过程中执行的是什么以及执行的先后顺序，pytest 提供--setup-show 选项可以实现我们想要的。



3.4: 使用 fixture 传递测试数据

fixture 非常适合存放测试数据，并且它可以返回任何数据。

```
@pytest.fixture()
def save_test_data():
return {"name": "kevin", "passwd": "hello1234"}
```

3.5: 指定 fixture 作用域

fixture 包含一个叫 scope 的可选参数，用于控制 fixture 执行配置和销毁逻辑的频率。

@pytest.fixture(scope=作用域)有 4 个可选值。

Scope=’function’:

函数级别的 fixture 每个测试函数只需要执行一次，配置代码在测试用例执行之前执行，销毁代码在测试用例运行之后运行。function 是 scope 的默认值。

Scope=’class’:

类级别的 fixture 每个测试类只执行一次，无论测试类里有多少类方法都可以共享这

个 fixture。

Scope=’module’:

模块级别的 fixture 每个模块执行需要运行一次，无论模块里有多少个测试函数，类方法或其他 fixture 都可以共享这个 fixture。

Scope=’session’:

会话级别的 fixture，每次会话只需要执行一次，一次 pytest 会话中的所有测试函数，方法都可以共享这个 fixture。

作用范围虽然由 fixture 自身定义，但是要注意 scope 参数是在定义 fixture 时定义的，因此，使用 fixture 的测试函数无法改变 fixture 的作用域。fixture 智能使用同级别的 fixture，或者比自己级别更高的 fixture。比如：函数级别的 fixture 可以可使用同级别的 fixture，也可以使用类级别，模块级别，会话级别的 fixture，但反过来不行！

3.6：fixture 的 autouse 选项

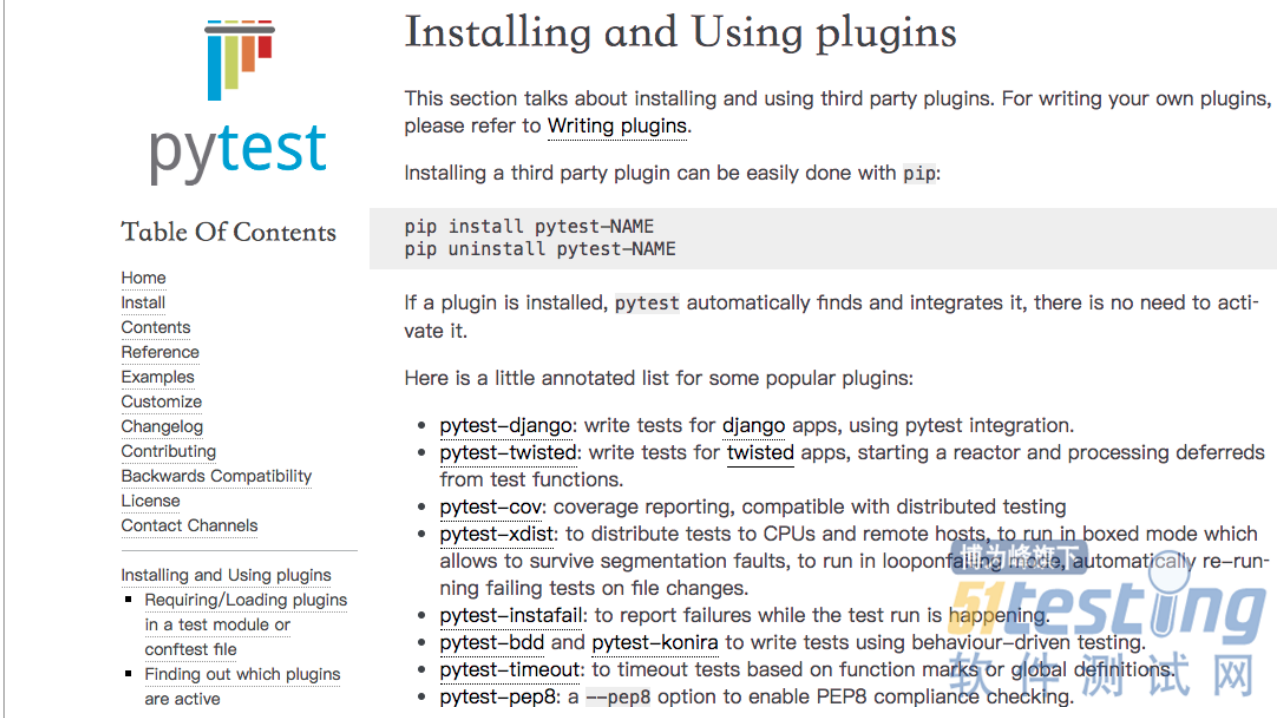
fixture 的可选项 autouse 可以决定是不是所有的测试用例自动使用该 fixture。

例子:

```
# 全局 setup 和 teardown  预置条件是登录，所欲 case 执行完成后退出 driver
@pytest.fixture(scope="session", autouse=True)
def login():
    commons.login()
    yield login
    commons.quit_driver()
```

四：Pytest 的插件

可以从 https://docs.pytest.org/en/latest/plugins.html 查看下载 pytest 的插件。



pytest

Table Of Contents

Home
Install
Contents
Reference
Examples
Customize
Changelog
Contributing
Backwards Compatibility
License
Contact Channels

Installing and Using plugins
- Requiring/Loading plugins in a test module or conftest file
- Finding out which plugins are active

Installing and Using plugins

This section talks about installing and using third party plugins. For writing your own plugins, please refer to Writing plugins.

Installing a third party plugin can be easily done with `pip`:

```
pip install pytest-NAME
pip uninstall pytest-NAME
```

If a plugin is installed, `pytest` automatically finds and integrates it, there is no need to activate it.

Here is a little annotated list for some popular plugins:

- pytest-django: write tests for django apps, using pytest integration.
- pytest-twisted: write tests for twisted apps, starting a reactor and processing deferreds from test functions.
- pytest-cov: coverage reporting, compatible with distributed testing
- pytest-xdist: to distribute tests to CPUs and remote hosts, to run in boxed mode which allows to survive segmentation faults, to run in looponfailing mode, automatically re-running failing tests on file changes.
- pytest-instafail: to report failures while the test run is happening.
- pytest-bdd and pytest-konira to write tests using behaviour-driven testing.
- pytest-timeout: to timeout tests based on function marks or global definitions.
- pytest-pep8: a `--pep8` option to enable PEP8 compliance checking.

4.1: pytest-repeat

如果希望在一个会话中重复允许测试用例，可以使用 pytest-repeat 插件。

(如果测试执行总是断断续续失败，可以尝试这个插件)

执行 pip(3) install -U pytest-repeat 可安装最新版本

```
Kevinqingkong:TestSuites mc$ pip3 install pytest-repeat
Collecting pytest-repeat
  Downloading https://files.pythonhosted.org/packages/c9/47/569e74afc10b2c58bfb358e4573639d34550b2ae2f18960c9db0c5aa500b/pytest_repeat-0.7.0-py2.py3-none-any.whl
Requirement already satisfied: pytest>=2.8.7 in /usr/local/lib/python3.7/site-packages (from pytest-repeat) (4.0.1)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/site-packages (from pytest>=2.8.7->pytest-repeat) (1.11.0)
Requirement already satisfied: atomicwrites>=1.0 in /usr/local/lib/python3.7/site-packages (from pytest>=2.8.7->pytest-repeat) (1.2.1)
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.7/site-packages (from pytest>=2.8.7->pytest-repeat) (18.2.0)
Requirement already satisfied: more-itertools>=4.0.0 in /usr/local/lib/python3.7/site-packages (from pytest>=2.8.7->pytest-repeat) (4.3.0)
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/site-packages (from pytest>=2.8.7->pytest-repeat) (40.4.3)
Requirement already satisfied: pluggy>=0.7 in /usr/local/lib/python3.7/site-packages (from pytest>=2.8.7->pytest-repeat) (0.8.0)
Requirement already satisfied: py>=1.5.0 in /usr/local/lib/python3.7/site-packages (from pytest>=2.8.7->pytest-repeat) (1.7.0)
Installing collected packages: pytest-repeat
Successfully installed pytest-repeat-0.7.0
Kevinqingkong:TestSuites mc$
```

4.2: pytest-xdist

通常测试都是依次执行，因为有些资源依次只能被一个用例访问。如果我们的测试用例不需要访问共享资源，那么就可以通过并行执行来提高执行速度。

使用 pytest-xdist 可以指定处理器进程数目来同时执行多个测试，如果我们将 pytest-

xdist 和 selenium-grid 结合起来可以将测试在多台机器上执行。

执行 pip(3) install pytest-xdist 安装此插件。

4.3: pytest-timeout

默认情况下，pytest 里的测试执行是没有时间限制的。如果测试过程中涉及会消失的资源，比如 web 服务，那么最好为测试执行时间加上时间限制。

Pytest-timeout 允许我们指定超时时间或者直接在测试代码中标注超时时间。

测试用例上标注的超时时间优先级高于命令行上的超时时间优先级。

4.4: pytest-rerunfail

如果我们有测试用例失败后重跑的需求，那 pytest-rerunfail 插件值得一试。

4.5: pytest-instafail

默认情况下，pytest 会在所有测试执行完毕后显示错误和失败用例的堆栈信息。如果测试执行时间很长，而我们希望及时看到错误或堆栈回溯信息而不是等所有用例执行完之后查看，那么就 pip install pytest-instafail 安装插件吧

它的使用也比较简单只需要 pytest --instafail 执行时加上选项即可。

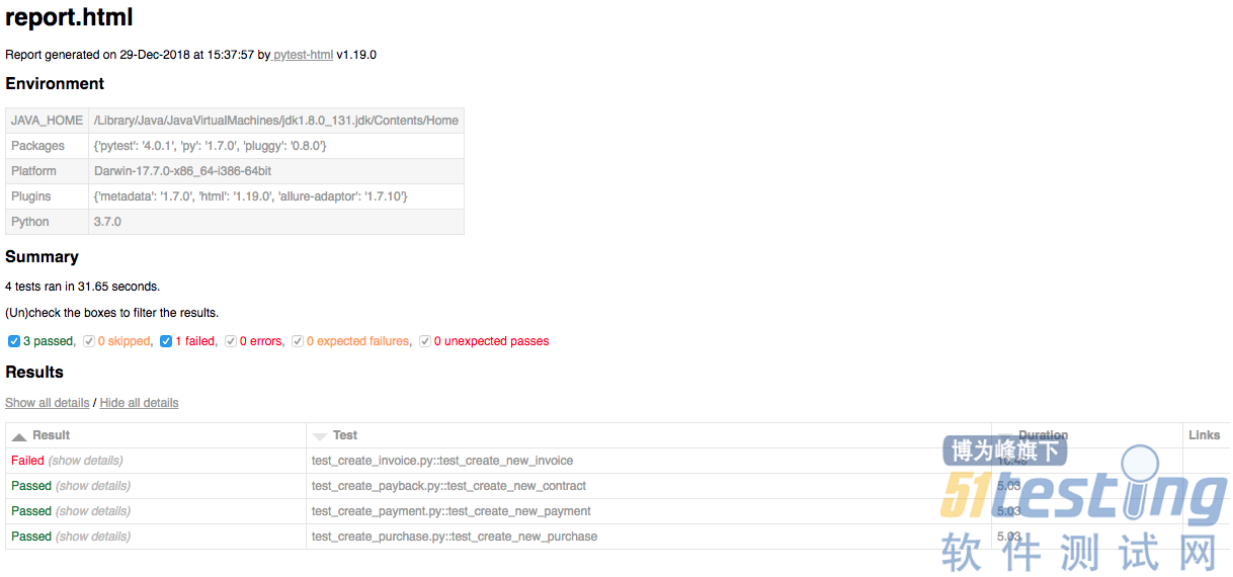
4.6: pytest-html

Pytest-html 对持续集成或长时间执行的测试非常有用。它可以为 pytest 测试生产一个现实测试结果的网页。这个 HTML 报告可以对测试结果(通过，跳过，失败，错误，预期失败，预期失败但通过)进行筛选，还可以按测试名称，持续时间，结果状态来排序。

HTML 报告还可以定制一些元素，如截图，输出信息。

Pip install -U pytest-html 安装插件

Pytest 用例绝对路径 --html=html 报告文件的绝对路径



report.html

Report generated on 29-Dec-2018 at 15:37:57 by pytest-html v1.19.0

Environment

JAVA_HOME	/Library/Java/JavaVirtualMachines/jdk1.8.0_131.jdk/Contents/Home
Packages	{'pytest': '4.0.1', 'py': '1.7.0', 'pluggy': '0.8.0'}
Platform	Darwin-17.7.0-x86_64-i386-64bit
Plugins	{'metadata': '1.7.0', 'html': '1.19.0', 'allure-adaptor': '1.7.10'}
Python	3.7.0

Summary

4 tests ran in 31.65 seconds.

(Un)check the boxes to filter the results.

3 passed, 0 skipped, 1 failed, 0 errors, 0 expected failures, 0 unexpected passes

Results

Show all details / Hide all details

Result	Test	Duration	Links
Failed (show details)	test_create_invoice.py::test_create_new_invoice	15.03	
Passed (show details)	test_create_payback.py::test_create_new_contract	5.03	
Passed (show details)	test_create_payment.py::test_create_new_payment	5.03	
Passed (show details)	test_create_purchase.py::test_create_new_purchase	5.03	

五：Allure(漂亮的测试报告)

Allure 框架是一个灵活的轻量级多语言测试报告工具，它不仅显示了在整洁的 Web 报告表单中测试内容的非常简洁的表示，而且允许参与开发过程的每个人从每天的测试执行中提取最大的有用信息。

从 dev/qa 的角度来看，Allure 报告缩短了常见缺陷的生命周期：测试失败可以分为 bug 和中断的测试，还可以配置日志、步骤、固定装置、附件、计时、历史以及与 tms 和 bug 跟踪系统的集成，因此负责的开发人员和测试人员将掌握所有信息。

从管理者的角度来看，Allure 提供了一个清晰的“大图”，其中包含了哪些特性、缺陷在哪里聚集、执行的时间轴是什么样子的，以及许多其他方便的事情。

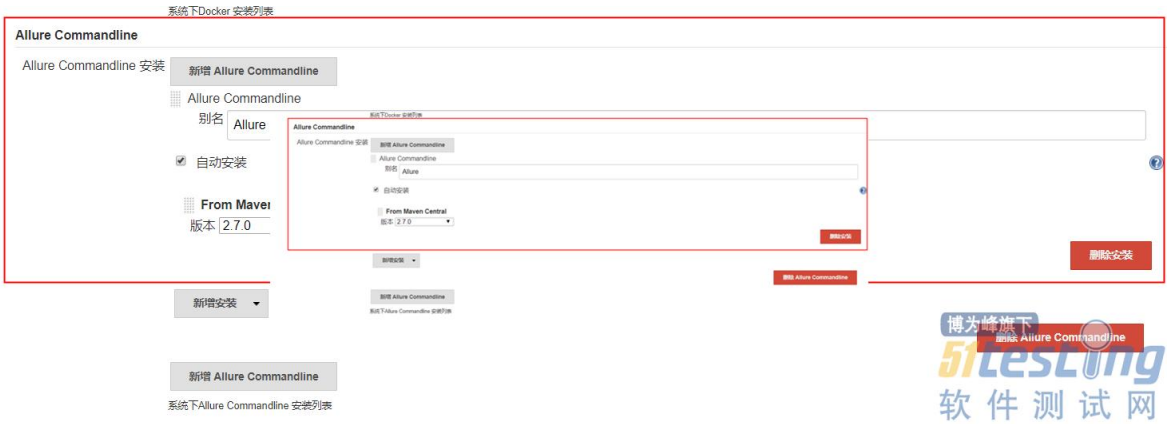
Jenkins 配置 Allure

- 安装 allure 插件

系统管理-->插件管理-->可安装插件　搜索 allure 进行安装

- 安装 Allure Commandline

安装完 allure 插件后，进入系统管理-->全局工具配置，安装 Allure Commandline



新建 job-->增加构建步骤-->Windows 批处理命令，输入：

```
python -m pytest E:\future-his\TestCase  -n 2  --alluredir  D:\\report
```

说明：

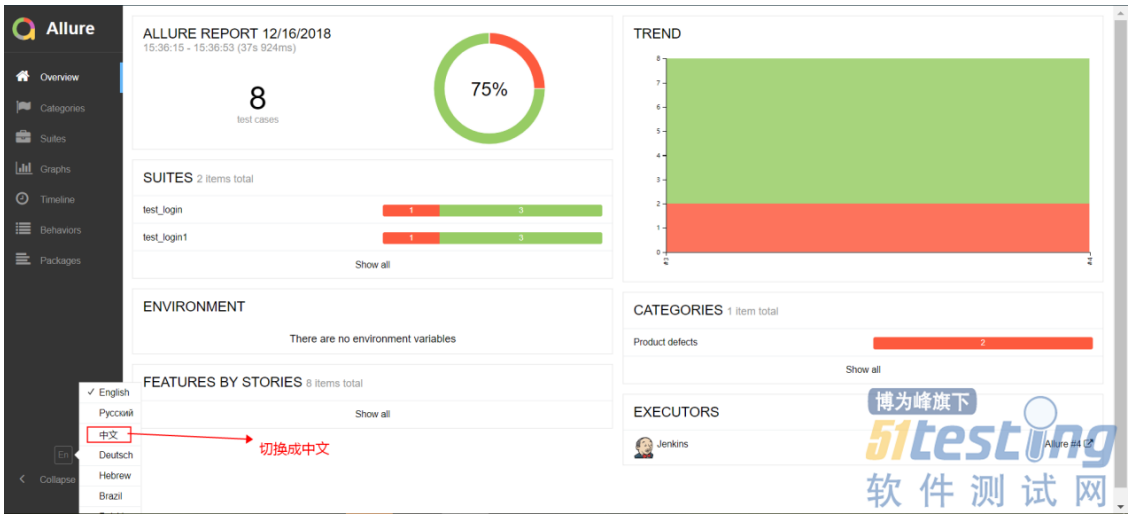
E:\future-his\TestCase：测试用例所在目录（unittest 脚本）

D:\\report ：测试报告输出目录

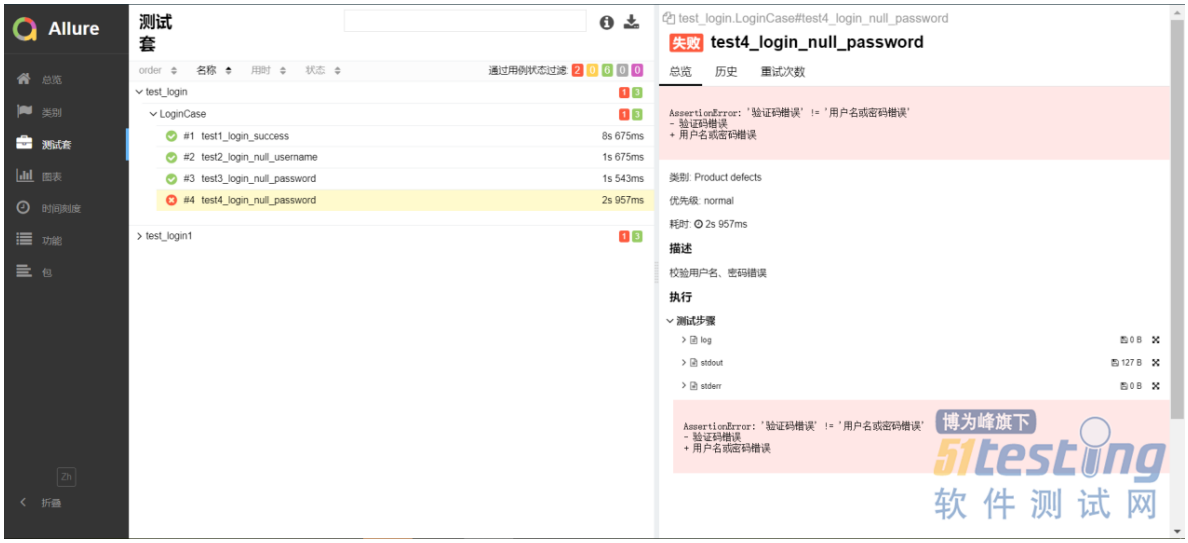
-n 2 ：表示开启 2 个线程（未开启 seleniumGrid 的情况下可去掉）

新增构建后操作，Allure report，输入 report 所在目录名称：与 Windows 批处理命令中设置的报告目录名称保持一致（这里使用的是 report 名称）

执行构建，查看报告：



查看报告详情：


Allure
测试套
总览
类别
测试套
图表
时间测度
功能
包
折叠
order
名称
用时
状态
通过用例状态过滤:
test_login
LoginCase
#1 test1_login_success 8s 675ms
#2 test2_login_null_username 1s 675ms
#3 test3_login_null_password 1s 543ms
#4 test4_login_null_password 2s 957ms
test_login1
test_login.LoginCase#test4_login_null_password
失败
test4_login_null_password
总览
历史
重试次数
AssertionError: '验证码错误' != '用户名或密码错误'
- 验证码错误
+ 用户名或密码错误
类别: Product defects
优先级: normal
耗时: 2s 957ms
描述
校验用户名、密码错误
执行
测试步骤
log
stdout
127 B
stderr
博为峰旗下
51testing
软件测试网

数据迁移测试教程：一份完整的指南

◆译者：咖啡猫

数据迁移测试概览

我们常常遇到这样的情况：随着技术的更新，比如软件升级到下一个版本或者更换不同的数据库等，随之而来应用系统也要部署到不同的服务器上。

这实际上意味着什么？

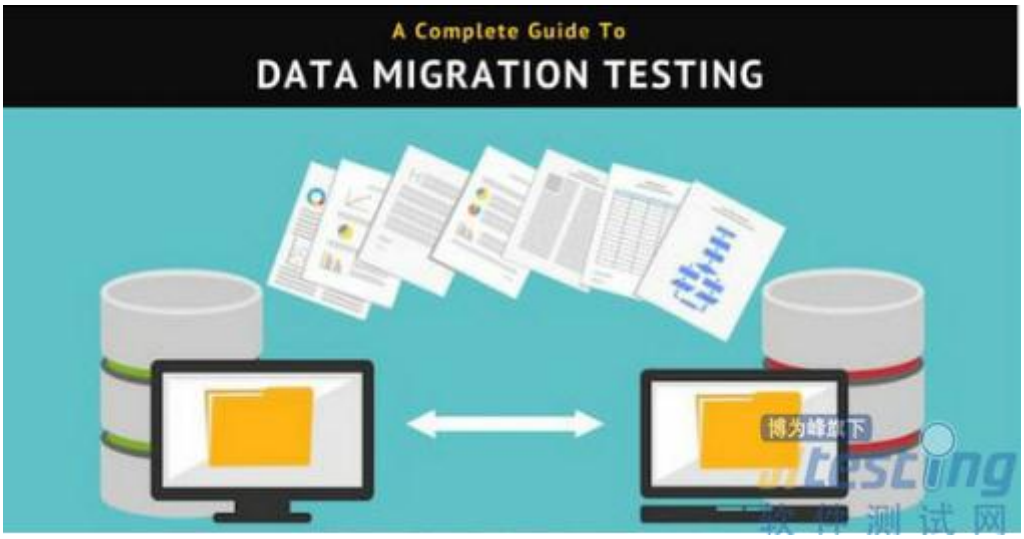
在这种情况下测试团队被期望做些什么呢？

站在测试的角度上，这意味着伴随着从现有系统到新系统的成功迁移，对应用系统一定要彻底地重新进行一次端到端的测试。

本系列教程：

- 数据迁移测试
- 迁移测试的类型

在这种情况下，系统测试需要被执行，并且不管是旧系统的数据还是新数据都需要用来作测试数据。伴随着功能的新增和修改，现有的功能点都需要被验证。



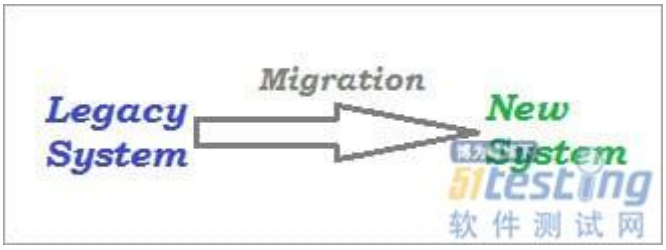
当用户的全部数据要被迁移到新系统时，就不仅仅是迁移测试了，我们可以把这种测试类型称为数据迁移测试。因此，迁移测试的对象包括：旧数据、新数据、新旧数据混合、旧功能以及新功能。

旧的应用系统通常被称为“遗留应用”。伴随着新应用的生成，遗留应用也会被强制执行测试，直到新应用变得稳定和完善。在新系统上进行全面的迁移测试会揭露一些新的问题，这些问题通常在遗留应用中不会被发现。

什么是迁移测试?

迁移测试是指当遗留系统向新系统无缝过渡（宕机时间小、数据完整性好、没有数据丢失）时，进行的一种验证性的测试过程，以确保新系统所有指定的功能特性和非功能特性都满足要求。

简单描述一下系统迁移:



为什么要做迁移测试?

我们都知道，应用迁移到一个全新的系统上可能有很多原因，比如：系统加固、技术废弃、系统优化或者其他任何原因。

当一个正在使用中的系统被迁移到一个新系统上去时，最起码要达到以下几点要求:

1.因为迁移给用户造成的任何类型的中断和不变都要被避免或者最小化。比如：宕机、数据丢失等。

2.需要确保用户可以继续使用软件的所有特性，尽管软件在迁移期间被造成较小的损伤或者零损伤。比如：功能的改变，特定功能的移除

3.另外，对于在实际的系统迁移过程中可能会发生的任何故障错误都做好预期和排除也十分重要。

因此，为了确保系统的平滑过渡且过渡中消除了上述任何缺点，在实验室进行迁移

测试是非常有必要的。

这种测试拥有其自身的重要性，并且当数据出现时它当扮演者一个非常重要的角色。

从技术上讲，基于以下几个目标迁移测试同样需要被执行:

- 确保新系统/升级系统的兼容性。新系统需要对遗留系统支持的所有软硬件平台兼容。同时，新系统对于新的软硬件平台的兼容性也需要被测试。
- 确保所有现存功能都能像遗留系统一样正常运行。当与遗留系统进行比较时，新系统的运行方式没有改变。
- 由于迁移引起大量缺陷 bug 的概率很高，很多 bug 往往和数据相关联，因此这类 bug 需要在测试过程中被发现出来并修复掉。
- 确保新系统的响应时间小于或者等于遗留系统的响应时间
- 确保测试过程中服务器、硬件、软件之间的连接是完好的且没有中断的，不同组件之间的数据流在任何情况下都不会被中断。

什么时候需要做迁移测试?

在迁移前和迁移后都需要做迁移测试。

在测试实践中，迁移测试的不同阶段是这样定义的:

1.预迁移测试

2.迁移测试

3.后迁移测试

除此之外，下面的测试作为完整迁移测试活动的一部分也需要被执行:

1.后向兼容性验证

2.回滚测试

在执行测试之前，对于每一个测试人员清晰地理解如下几点是非常有必要的:

1.新系统的每一个部分分别发生了哪些改变（服务器、前端、数据库、调度计划、数据流、功能点等等）

2.了解团队设计的实际的迁移策略。迁移如何发生，系统后台如何一步步地发生变化以及迁移过程中需要执行哪些脚本。

因此对新系统和旧系统进行一个彻底的学习是非常必要的，然后才能设计测试用例及测试场景来覆盖上述测试阶段和编写测试策略。

数据迁移测试的策略

设计迁移测试的测试策略包含一系统的活动并且很多方面需要被考虑到。这是为了更有效的执行迁移测试并且尽可能减少错误和风险的发生。

迁移测试包括哪些活动

1) 形成专业的团队

构建一个由专业知识扎实且实践经验丰富的成员组成的专业化团队。并且为成员们提供迁移系统相关知识的培训。

2) 业务风险分析-可能错误分析

不应该在迁移后影响了现有业务的正常开展，因此-开业务风险分析会时需要引入利益相关方（测试经理、业务分析师、架构师、产品经理、业务经理等等）并识别风险、制定可实施的减轻风险的方案。测试应该包括一些可以发现风险的场景并验证是否已经实施了缓解措施。执行“可能错误分析”需要采取合适的“错误猜测方法”并且围绕这些错误来设计测试用例以期在测试过程中揭露它们

3) 迁移范围分析和识别

分析界定清楚迁移测试的范围，也就是什么时候系统的哪些方面需要被测试

4) 为迁移选择合适的工具

在制定测试策略、选择手工还是自动化时，明确将要使用什么工具。比如：用于对比源数据和目标数据的自动化工具

5) 为迁移选择合适的测试环境

为迁移前测试和迁移后测试选择独立的测试环境，以方便后续测试的进行。理解并记录下遗留系统和新系统的技术要点，确保测试环境按此建立。

6) 迁移测试详尽文档的制定和检查

准备迁移测试的详细文档，文档应该清晰地描述了测试方案、测试范围、测试模式（手工还是自动化）、测试方法（黑盒测试、白盒测试）、测试周期、测试进度、造数据的方法和使用现网数据的方法（敏感数据需要被隐藏）、测试环境说明书、测试人员资质等，并且与利益相关方对文档进行一次检查

7) 迁移系统的产品发布

在文档中分析并记录下产品迁移需要做的事项并提前发布该文档

迁移测试的不同阶段

下面给出的是迁移测试的不同阶段。

第一阶段：预迁移测试

在迁移数据前，一系列的测试行为作为预迁移测试阶段的一部分被执行。在一些简单的应用系统中这是被忽略或者不考虑的。但是当复杂的应用系统被迁移时，预迁移测试活动就变得十分必要了。

这一阶段需要做的工作如下所示：

- 设置一个清晰地数据范围—什么数据被包括，什么数据被排除，哪些数据需要被转换等
- 在遗留系统和新系统之间形成数据映射关系—为遗留系统中的每种数据找到在新系统中相关联的类型并把它们映射在一起—高水平的映射
- 如果新系统有些字段是强制性必填的，但是遗留系统中没有，确保遗留系统没有的字段是空值—低水平映射
- 学习新系统的数据结构—字段名称、类型、最小值和最大值、长度、是否为强制字段等
- 遗留系统中的一些数据表应该被标记。如果迁移后的系统中有新增或删除表格的话这些也是需要被验证的
- 每个表格、视图中有大量的记录都应该在遗留系统中被标记
- 学习新系统及他们连接点的接口。接口的数据流应该是高度安全且不可被破坏的

- 为新系统的新条件而准备测试场景、测试用例
- 让测试人员执行一系列测试场景、测试用例并且保存结果和运行日志。迁移后同样需要执行这一过程以保证遗留系统的数据和功能是完好的。
- 数据的数量应该被纪录下来，因为迁移完成后需要确认是否有数据丢失。

第二阶段 迁移测试

“迁移指导”应该是被迁移团队严格依据迁移活动来制定的。理想情况下，迁移活动在数据备份完成后开始，这样的话任何时候遗留系统都可以被恢复。

验证“迁移指导”文档同样是数据迁移测试的一部分。验证文档是否通俗易懂，所有的脚本和步骤都被清晰地记录到文档中而没有任何模糊不清的地方。各种类型的文档错误，与实际执行步骤不相符的地方同样要被重视起来，这样这些问题才能被报告上去且得到修复。迁移脚本、指导还有其他与实际迁移相关的信息需要从版本控制数据库中拿出来为执行做准备。

记下从迁移开始到系统成功恢复之间的时间也是一个要被执行的测试用例，因此“迁移时间”需要被纪录到最终的测试报告中（测试报告是作为迁移测试结果的一部分要被交付出去的并且这些信息在产品发布时非常有用）。在测试环境中纪录的宕机时间将被外推来近似计算真实系统的宕机时间。真实系统是指迁移活动将被执行的那个遗留系统。

在测试过程中，所有环境组件都会被从网络中移除以便进行迁移活动。因此纪录宕机时间对迁移测试是非常有必要的。理想情况下，宕机时间应该和迁移时间相同。

一般地，迁移活动应该被定义在迁移指导文档中，包括如下内容:

- 实际迁移的应用系统
- 防火墙、端口、主机、硬件、软件配置都根据新系统（迁移目标系统）而修改
- 数据泄漏，安全性检查要被执行
- 所有应用组件的连接需要被检查

建议测试人员在系统后台或者通过执行白盒测试的方法来验证上述问题。

一旦指南中定义的迁移活动被完成了，所有的服务器都会重启并且与验证迁移成功

性相关的基本测试都完成了，这就确保了所有端到端的系统都被合适地链接并且所有组件都能和对方通信，数据库启动并且正常运行，前端能和后端正常通信。这些测试需要被趁早验证并记录到迁移测试详细文档中。

如果应用软件支持不同的平台，在这种情况下，迁移需要在不同平台上独立地验证。

验证迁移脚本也是迁移测试的一部分。有时同样需要使用“白盒测试”的方法在独立的测试环境里验证。

因此迁移测试是一种白盒测试和黑盒测试的有机结合

一旦迁移相关的验证完成了并且相关用例都通过了，那么测试团队可以接着进行后迁移测试了。

第三阶段 后迁移测试

一旦应用系统被成功迁移，就要开始做后迁移测试了。

这时，端到端的系统测试需要在测试环境中执行，测试人员使用遗留系统和新系统的数据作为测试数据执行测试用例和测试场景。

除此之外，还有一些指定的项目需要在迁移环境中验证，如下:

（所有这些都需要作为测试用例记录在测试文档中）

1.检查是否遗留系统中的所有数据都在计划的宕机时间内被迁移到新的应用系统中了。为了验证这个，需要比较遗留系统和新系统的数据库中每个表和视图，同时记录下移动一万条记录的时间。

2.检查所有计划的变更（字段和表格的新增或者删除），新系统都做了相应的更新。

3.数据从遗留系统到新系统应该维持其原有的值和格式除非没有指定要这样做。为了确保这点，需要对比新旧系统数据库的数值是否一致。

4.针对新应用系统测试迁移数据。这里需要覆盖最大数目的用例。为确保数据迁移测试百分之百的覆盖率，使用自动化测试工具来验证为佳。

5.检查数据库的安全性。

6.对所有可能的样本记录检查数据完整性。

7.检查并确保在遗留系统中支持的功能在新系统中同样支持。

8.检查应用中的数据流，这里的应用包括所有组件。

9.不同组件之间的接口应该被广泛测试，因为当数据在组件之间传递时它不能被修改、丢失、破坏，完整性测试用例是用来验证这个的。

10.检查遗留系统数据的冗余情况。没有遗留系统的数据在迁移过程中被重复。

11.检查数据不匹配的情况，比如数据类型改变，存储格式的改变等等

12.在遗留系统中进行的所有字段级别的检查在新系统中也都要检查一遍。

13.新系统中任何数据新增都不能反映到遗留系统中。

14.支持通过新应用系统更新遗留系统的数据，并且一旦更新成功不能反映到遗留系统中。

15.支持通过新应用系统删除遗留系统的数据，并且一旦删除成功在遗留系统中不能再次被删除。

16.验证对于遗留系统的某些改变是否支持作为新系统的一部分的新交付的功能

17.验证来自遗留系统的用户可以继续使用所有新功能和旧功能，尤其是那些有改动的功能点。执行在预迁移测试中执行过的测试用例

18.在系统中创建新的用户并执行测试以确保遗留系统和新系统的功能支持新用户访问并且运行良好

19.使用多种数据样本执行功能测试（比如不同年龄组，来自不同区域的用户等）

20.同样需要验证对于新特性已经使能了“特性标志位”并且通过控制特征标志位来开关系统的新特性

21.性能测试对于确认新系统在性能上没有降级是非常重要的。

22.同样需要执行负载和压力测试以确保系统的稳定性。

23.为了验证软件升级没有带来任何安全隐患因此需要执行安全测试，尤其是在系统迁移过程中发生改变地方进行安全测试。

可用性也是需要被验证的一方面。当软件界面布局/前后端系统发生改变时，验证相对遗留系统用户是否觉得新系统更加方便快捷。

由于后迁移测试的范围非常广泛，最好是隔离出重要的测试用例来优先执行以验证迁移是否成功，然后才执行剩下的用例。

同时，我建议将那些端到端的功能测试用例和其他可能的用例自动化执行，这样可以节省测试时间并且很快拿到可用的测试结果。

给测试工程师一些设计后迁移测试用例的建议

- 当应用被迁移时，并不意味者对于整个新系统来说所有的用例都要重写一遍。那些为遗留系统设计的测试用例同样可以用在新系统上。因此，尽可能使用旧的测试用例，并在需要时将遗留测试用例转换为新应用程序的用例。
- 当新系统中有些特性发生改变时，相关的测试用例也需要被修改。
- 当新系统中新增了一些功能时，新的测试用例也要被设计好。
- 当新系统中删掉了部分功能时，相关的遗留系统的测试用例就不需要在新系统中执行，这部分用例应该被标记无效并和有效用例隔离开。
- 使用相关术语设计测试用例时，应该保证用例的可靠性和一致性。在测试用例中应该也包含一些（批判性的数据、极限数据、脏数据）这样才能保证在执行时没有遗漏。
- 当新系统的的 UI 设计与遗留系统不同时，那么 UI 相关的测试用例就需要被修改以适应新系统的设计。至于是更新原有用例还是写新用例，测试人员可以根据改动量的大小来决定。

后向兼容性测试

系统迁移同样要求测试工程师去验证“后向兼容性”。其中，新系统要求兼容至少两个版本的旧系统并且确保在这些版本上的所有功能运行良好。

后向兼容性测试是为了确保:

1.新系统是否支持它前面两个旧版本所支持的功能。

2.可以从之前的两个版本成功迁移到新版系统上，并且迁移期间不会遇到任何困难。

因此通过执行兼容性方面的测试用例来确保系统的后向兼容性是非常必要的。另

外，与后向兼容性相关的测试用例应该被提前设计好并写入测试文档中。

回滚测试

万一执行迁移的过程中遇到任何问题或者在迁移过程的某个时间点遇到了一个错误，那么应该容许回退到遗留系统并且快速恢复遗留系统的功能而不影响用户使用和之前支持的功能的运行。

因此为了验证这一点，迁移失败的测试场景也需要被设计来作为消极测试的一部分并且回滚机制也需要被测试。恢复到遗留系统的总时间也需要被记录下来并且写入测试报告中。

回滚完成后，主要的功能和自动化回归测试应该被执行以便确然迁移没有任何不良影响并且本次回归可以成功地恢复到原有遗留系统。

迁移测试总结报告

测试总结报告应该在完成测试后输出并且报告应该包括迁移测试不同阶段的不同用例/场景的测试结果状态（通过/失败）和测试过程日志。

应该清晰地记录如下活动所花时间:

1.迁移总时间

2.系统应用宕机时间

3.迁移一万条记录花费的时间

4.回滚时间

除以上信息外，任何观察和建议都可以记录到报告中。

数据迁移测试中面临的挑战

在这项测试中面临的主要挑战是数据，下面就列出一些挑战:

1）数据质量

我们会发现将遗留系统中的数据放到新系统中使用的话，数据质量都很差，这种情况下，应该提高数据质量以满足业务标准的要求。

这些因素会导致数据质量不好，比如：过多假设、迁移后的数据转换、遗留系统自身的数据无效、失败的数据分析等。这些脏数据会导致高昂的运营成本，激增的数据完

整性风险并最终导致背离业务目标。

2）数据不匹配

数据从遗留系统迁移到新系统中可能会发生数据不匹配的情况。这可能是因为数据类型的改变，数据存储格式的改变，使用数据的目的被重新定义了。

这就导致需要付出巨大的努力去做一些必要的更改，无论是修正不匹配的数据或者接受这种不匹配或者略作调整以满足要求。

3）数据丢失

数据从遗留系统迁移到新系统时可能会丢失。也许是强制字段丢失也许是非强制字段丢失。如果是非强制字段丢失了，那整条记录仍然是有效的并且可以继续使用。如果是强制字段丢失了，整条记录都会失效并且无法恢复。这将导致巨大的数据丢失并且不得不从备份数据库或者抓取的日志中取回数据。

4）数据量巨大

大量的数据需要一些时间才能在迁移活动的宕机窗口期内完成迁移。比如：电信系统中的刮奖卡，智能网络平台上的用户等。这里的挑战在于等到遗留系统的数据被清洗，大批量的新数据又将被创造，这些新数据同样是需要被迁移的。自动化才是大数据迁移的终极解决方案。

5）用实际数据模仿真实线上环境

在测试实验室中模拟真实线上环境是又一大挑战。因为在线上真实环境里，测试人员会面临在测试过程中没有遇到过的各种各样的关于真实数据真实系统的问题。因此，迁移系统中数据的取样、真实环境的拷贝、大量数据的识别等过程在执行数据迁移测试是都是非常重要的。

6）模仿大数据量

团队需要非常仔细地学习线上系统中的数据并且能够提出典型的分析和数据采样。

比如：用户按年龄分组有 10 岁以下、10 岁-30 岁等，只要有可能，只要是来自线上的数据都要被获取。如果没有线上数据，那么就需要在测试环境中造数据。此时自动化工具在创造大批量数据时是需要被使用的。

减小数据迁移风险的建议

下面是一些可以用来减小数据迁移风险的建议:

- 标准化遗留系统中的数据。这样迁移时标准的数据在新系统中才是可用的。
- 提高数据的质量。这样当迁移时，有高质量的数据用于测试就能给人一种作为终端用户在测试的感觉。
- 迁移之前要清洗数据。这样当迁移时，重复的数据才不会出现在新系统中，这样可以保证整个系统的洁净度。
- 再次检查约束条件、存储过程、以及用于产生精确结果的复合查询语句。这样迁移时，新系统中才会返回正确的数据
- 选择正确的自动化工具执行遗留系统与新系统之间的数据/记录的比对检查。

结论:

因此，考虑到执行数据迁移测试的复杂性，由于任何一个验证方面的小失误都会导致整个产品迁移的失败，在迁移前和迁移后进行认真彻底的系统学习和分析是非常重要的。而且要用强健的测试工具和有能力的测试工程师来设计有效的迁移策略。

正如我们知道的，迁移对系统的质量有着很大的影响，整个团队都要付出大量的努力来验证整个系统在所有方面的情况，比如功能、性能、安全性、易用性、可用性、健壮性、兼容性等，这样才能确保迁移测试的成功。

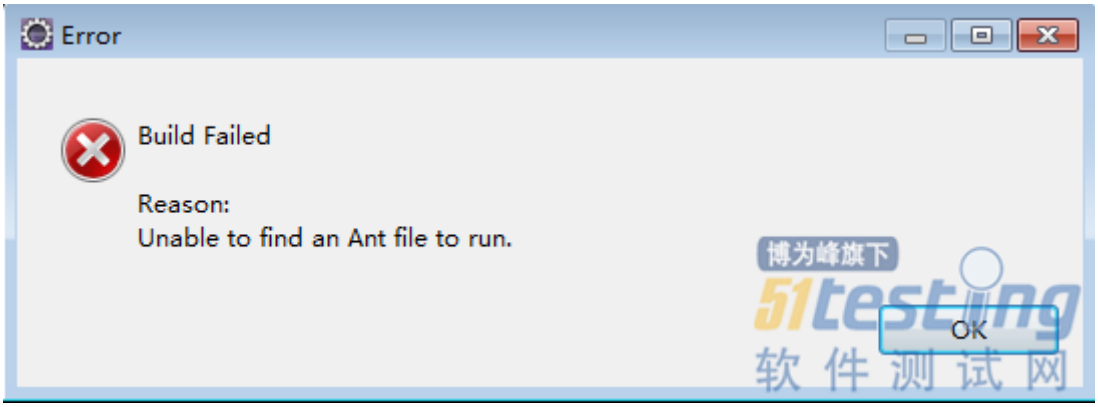
“不同类型的迁移测试”会在实际工作中经常发生，如何掌控这类测试将会在我们本系列的下一个教程的介绍。

Selenium 报错集锦（代码迁移）

◆作者：桃 子

场景：前端时间在家里环境编写了一部分脚本，中间耽搁了一段时间，最近想在单位重新开始弄，所以将代码考到单位环境后出现了一系列报错问题，对这些问题梳理总结如下。

1、build failed ，提示“unable to find an ant file to run”



解决：我采用方法 2 解决成功

1. 请检查你项目的工程名和项目文件夹名称的大小写一不一致。
2. 可能是要运行的文件没有在工程中导致。可以尝试:右键工程->new->file->advanced->link to file in file system-> 找到文件并加入。
3. 默认的行为是在当前目录下找 build.xml，如果你的文件不是叫这个名字那就用｛省略 ant 路径｝-f start.xml build 这样参数，它运行 start.xml 里面的 build target。

出现问题原因：这个文件时我从电脑 A 拷贝到电脑 B，缺少相应文件导致

2、python 出现"No module named "XXX""的解决办法

```
  File "C:\Users\Administrator\AppData\Local\Programs\Python\Python36\lib\site-packages\pip\_internal\command
    from pip._internal.cli.base_command import Command
  File "C:\Users\Administrator\AppData\Local\Programs\Python\Python36\lib\site-packages\pip\_internal\cli\bas
    from pip._internal.download import PipSession
  File "C:\Users\Administrator\AppData\Local\Programs\Python\Python36\lib\site-packages\pip\_internal\downloa
    class PipSession(requests.Session):
AttributeError: module 'pip._vendor.requests' has no attribute 'Session'
```

```
<terminated> F:\download\ass.py
Traceback (most recent call last):
  File "F:\download\ass.py", line 19, in <module>
    from pip import locations
ImportError: cannot import name 'locations'
```

出现这个问题的原因:

1. 环境中没有安装 pip 文件

2. 安装了，环境路径错误

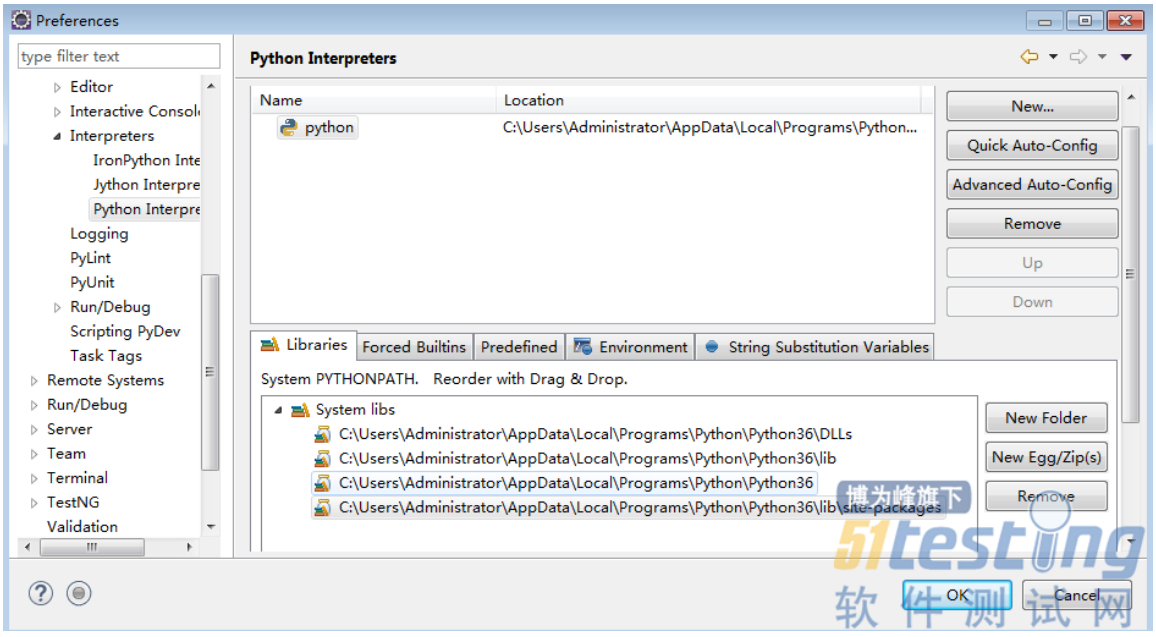
解决如下:

首先执行升级命令 升级到最新

python -m pip install -U pip

再到 site-packages 目录下找 pip 包

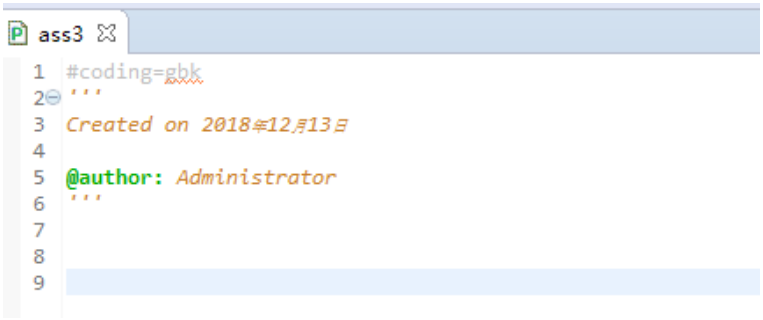
查看编译环境是否能找到自己安装的包的路径，确认是文件夹下的目录



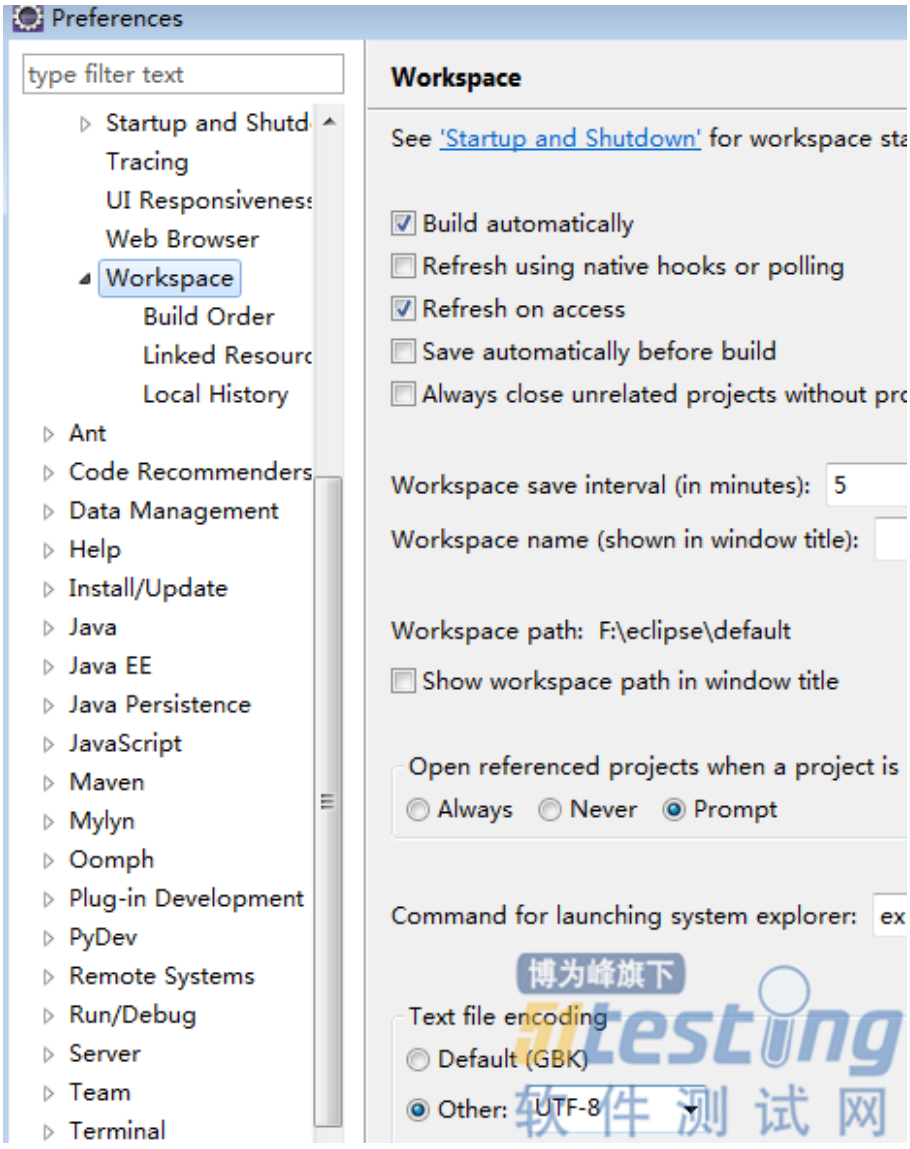
3、Non-UTF-8 code starting with '\xc4'

```
  File "F:\eclipse\default\ass3\src\ass3\ass3.py", line 2
SyntaxError: Non-UTF-8 code starting with '\xc4' in file F:\e
```

解决（1）: 在程序最上方加上语句，# coding=gbk



解决（2）：在 preference 下进行修改



4、ocr 识别过程中报错　tesseract is not installed

```
pytesseract.pytesseract.TesseractNotFoundError: tesseract is not installed or it's not in your path
```

这个问题无论在初始编译时或者在后来坏境变更调试时都会遇到的问题。

解决：问题原因是源码中的默认路径位置与文件位置不同，需要更改一下

然后将源码中的：

```
tesseract_cmd = 'tesseract'
```

更改为：

```
tesseract_cmd = r'C:\Program Files (x86)\Tesseract-OCR\tesseract.exe'
```

5、如果下拉菜单的内容项提示定位不到，可以试试 move_to_element()办法进行

```
D:\software\eclipse\class1\wx\ass.py
Traceback (most recent call last):
  File "D:\software\eclipse\class1\wx\ass.py", line 66, in <module>
    browser.find_element_by_xpath(".//ul[@id='user-menu']/li[7]/a/span").click()
  File "C:\Python34\lib\site-packages\selenium-3.8.0-py3.4.egg\selenium\webdriver\remote\webeleme
```

下拉框是鼠标移上去直接弹出的，那么我们可以使用 move_to_element()进行操作

6、切换窗口句柄时，提示 list index out of range python

```
#browser.implicitly_wait(30)
time.sleep(5)
browser.switch_to_window(browser.window_handles[1])
browser.find_element_by_xpath("//form/div/div/descendant::a[@href='/adview']").click()
```

后来想了一下，是因为新窗口没有打开，所以就不存在窗口 2，所以才会提示列表超出范围，感觉有点开窍了

7、进入三级页面提示 503 Service Temporarily Unavailable，如果手动刷新页面重新加载成功

网上看都是如何配置及原因的，没告诉如何解决

于是我想，如果是这样的话，执行刷新操作应该可以规避这个问题。

503 Service Temporarily Unavailable

nginx

语句：driver.refresh()

总结：通过这一系列的问题，有点感触分享一下，遇到问题后最好先分析一下报错属于哪一类，锻炼自己不通过网上找答案，提高自己的分析能力；另外，有些时候开拓一下思路，比如上面的问题 9，如果只是想怎么从根本上取消 503 提示，再加上对这个环境不熟悉，估计破费周折，但是换个思路加一行刷新代码，程序就可以往下进行了。

JMeter RabbitMQ 采样器 AMQP 详解与实战

◆ 作者：王 练

摘要： JMeter 是性能测试中使用非常广泛的工具之一，其中让 JMeter 能够大放异彩的扩展插件异常重要。本文介绍测试 RabbitMQ 的扩展插件：AMQP 采样器。不同于其他采样器，该插件的使用需要从源码中编译后使用。本文首先对 RabbitMQ、AMQP、MQTT 等基本概念进行了简要说明。之后将 AMQP 插件的安装过程，源码编译方法等进行详细说明。之后对该插件的使用方法进行详解，详细介绍采样器中每个字段的含义。最后通过一个实际的 JMeter 用例说明 AMQP 采样器的使用方法。

本文虽然主要介绍 AMQP 采样器，其实也间接介绍了 JMeter 的插件知识，从这个角度来看，JMeter 有非常好的扩展性，能够实现测试中非常丰富的需求。

一、RabbitMQ 简介

要了解 RabbitMQ，首先要了解几个基本概念：MQ、JMS、AMQP、MQTT。

MQ 是 Message Queue 的简称，即消息队列。队列我们可以理解为管道。以管道的方式做消息传递。消息传递作为基本通信机制已经在全世界成功运用。无论是人与人、机器与人还是机器与机器之间，消息传递一直都是唯一常用的通信方式。在双方（或更多）之间交换消息有两种基本机制。

首先出现的是 Java 消息传递服务（Java Messaging Service (JMS)）。JMS 是最成功的异步消息传递技术之一。随着 Java 在许多大型企业应用中的使用，JMS 就成为了企业系统的首选。它定义了构建消息传递系统的 API。

为了通用性，高级消息队列协议（Advanced Message Queueing Protocol (AMQP)）应运而生。JMS 非常棒而且人们也非常乐意使用它。微软开发了 NMS（.NET 消息传递服务）来支持他们的平台和编程语言，它效果还不错。但是碰到了互用性的问题。两套使用两种不同编程语言的程序如何通过它们的异步消息传递机制相互通信呢。此时就需要

定义一个异步消息传递的通用标准。JMS 或者 NMS 都没有标准的底层协议。它们可以在任何底层协议上运行，但是 API 是与编程语言绑定的。AMQP 解决了这个问题，它使用了一套标准的底层协议，加入了许多其他特征来支持互用性，为现代应用丰富了消息传递需求。

后来又出现了消息队列遥测传输（Message Queueing Telemetry Transport (MQTT)）。已经有了面向基于 Java 的企业应用的 JMS 和面向所有其他应用需求的 AMQP。为什么我们还需要第三种技术？它是专门为小设备设计的。计算性能不高的设备不能适应 AMQP 上的复杂操作，它们需要一种简单而且可互用的方式进行通信。这是 MQTT 的基本要求，而如今，MQTT 是物联网（IOT）生态系统中主要成分之一。

JMS、AMQP、MQTT 都是 MQ 的协议，是需要遵循的一套标准。正如需要 TCP/IP 来实现 OSI 七层规范一样，真正在使用的是这些协议的实现。JMS 本身有一套 Java API 可以实现 JMS，同时 ActiveMQ 也支持 JMS。事实上，现在流行的 MQ 实现，都支持多种协议，比如阿里巴巴的 RocketMQ，支持 JMS、MQTT。

今天介绍的 RabbitMQ 是一个开源的 AMQP 实现，服务器端用 Erlang 语言编写，支持多种客户端，如：Python、Ruby、.NET、Java、JMS、C、PHP、ActionScript、XMPP、STOMP 等，支持 AJAX。用于在分布式系统中存储转发消息，在易用性、扩展性、高可用性等方面表现不俗。作为一款优秀的 AMQP 实现，在很多系统中作为消息中间件使用。

测试 RabbitMQ 在实际系统中的性能有很多方法，其中使用 JMeter 的 AMQP 插件是比较简单的实现方案。

二、插件安装说明

JMeter 插件安装非常简单，只要将扩展插件的 jar 包拷贝到\lib\ext 文件夹下即可。不过 AMQP 插件只有源码，需要自行编译。

首先在如下路径下载源码：https://github.com/jlavallee/JMeter-Rabbit-AMQP

如果安装了 ant，并且了解 ivy，可以直接使用工程的 ivy.xml 进行编译，生成 JMeterAMQP.jar。或者根据 ivy.xml 的依赖配置，手工下载依赖的 jar 包，具体需要的 jar 如下：

amqp-client-3.5.1.jar，ApacheJMeter_core.jar，

avalon-logkit-2.0.jar，commons-codec-1.4.jar，

commons-collections-3.2.1.jar，commons-httpclient-3.1.jar，

commons-io-1.4.jar，commons-jexl-2.1.1.jar，

commons-lang3-3.1.jar，commons-logging-1.1.1.jar，

commons-net-1.4.1.jar，jorphan-2.6.jar。

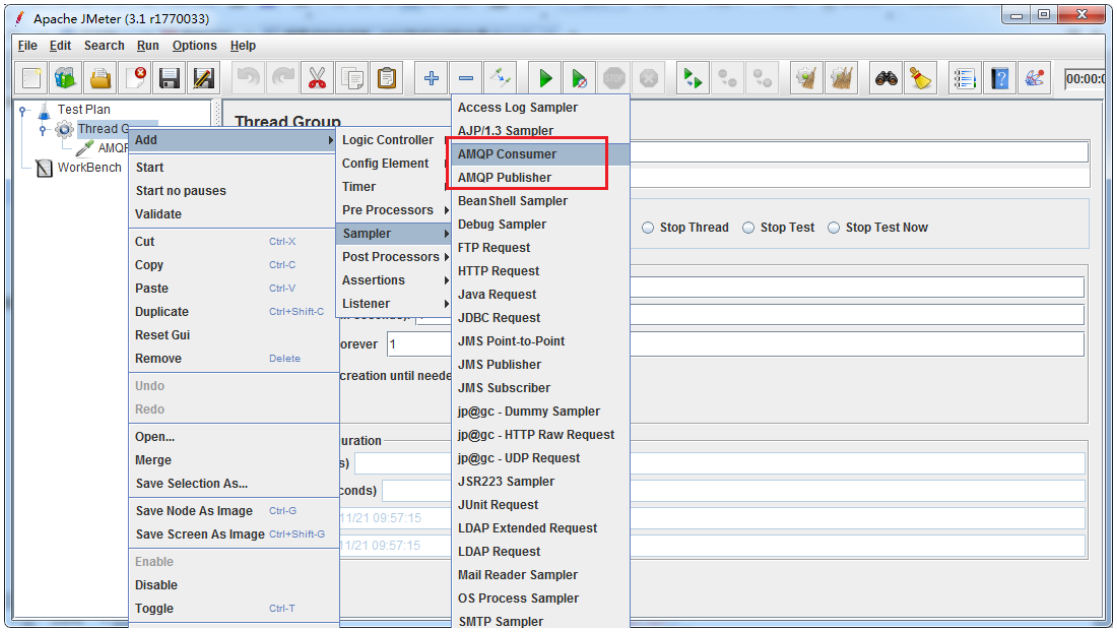
加入到 build path 后，编译生成 JMeterAMQP.jar。

之后将 JMeterAMQP.jar 拷贝到 JMeter 的\lib\ext 路径。

常见问题：拷贝 JMeterAMQP.jar 后，打开 JMeter 无法看到 AMQP 采样器，日志报错如下：

```
2018/11/21 09:54:44 ERROR - jmeter.JMeter: Uncaught exception:  java.lang.ArrayIndexOutOfBoundsException: -3
  at org.apache.jorphan.gui.MenuScroller.refreshMenu(MenuScroller.java:552)
  at org.apache.jorphan.gui.MenuScroller.access$300(MenuScroller.java:55)
  at org.apache.jorphan.gui.MenuScroller$MouseScrollListener.mouseWheelMoved(MenuScroller.java:578)
  at java.awt.Component.processMouseWheelEvent(Unknown Source)
```

需要将 amqp-client-3.5.1.jar 也拷贝到\lib\ext 路径下。再次打开 JMeter 就可以添加 AMQP 采样器了。



直接下载的 AMQP 插件自己的 Publisher 和 Consumer 传递消息没有问题，和其他组件配合使用时中文会出现乱码。修改源码中对消息体的获取，以 Consumer 为例，修改类 com.zeroclue.jmeter.protocol.amqp.AMQPConsumer，将获取返回结果的代码修改如下内容：

```
102                 if (getReadResponseAsBoolean()) {
103                     String response = new String(delivery.getBody());
104                     result.setSamplerData(response);
105                     result.setResponseMessage(response);
106                 }
```

修改为：

```
103             if (getReadResponseAsBoolean()) {
104                 String responseBefore = new String(delivery.getBody());
105                 String response = new String(responseBefore.getBytes("GBK"),"UTF-8");
106                 result.setSamplerData(response);
107                 result.setResponseMessage(response);
108             }
```

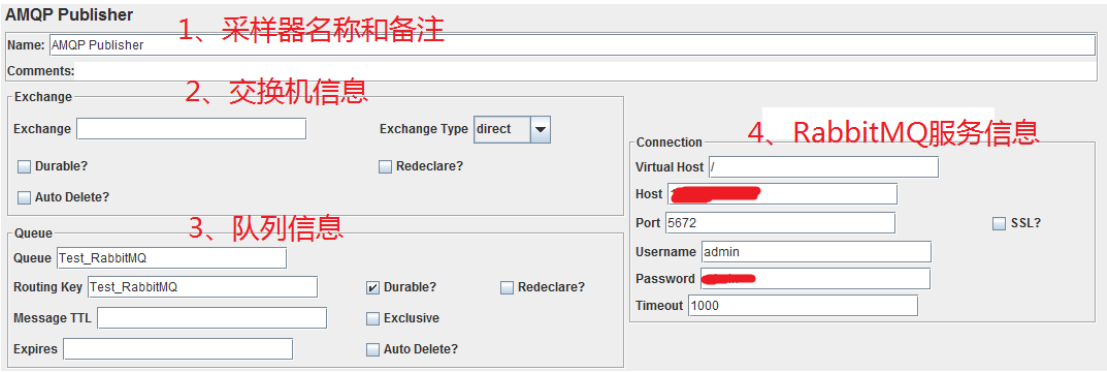
重新编译后替换 JMeterAMQP.jar 即可解决中文乱码问题。

三、插件使用说明

AMQP 插件在 JMeter 中的使用，与其他采样器一致，直接增加到线程组中即可。根据对消息队列的贡献不同，分为发布者和消费者，前者作用是向指定的消息队列中发送消息，后者是在指定的消息队列中收取消息。在 AMQP 插件中分别为 AMQP Publisher 和 AMQP Consumer。

3.1、AMQP Publisher

添加后的 AMQP Publisher 采样器如图所示。分两部分进行介绍。



上图的 4 个方面是消息的基本信息。包括如下内容:

1、采样器的名称和备注。这是 JMeter 所有采样器的属性，用来进行线程组的采样管理。建议取有意义的名字。

2、交换机信息。如果需要发送的 MQ 消息是以交换机的方式进行消息传递，需要配置。如果不需要则配置“Exchange”属性为空即可。具体的属性含义如下。

- Exchange：交换机名称，默认为 jmeterExchange。RabbitMQ 消息传递通过交换机或队列模型进行，需要根据实际系统定义情况进行配置。

- Exchange Type：交换机类型，规定了消息的路由策略，根据实际的消息交换机类型进行配置，可选为 direct,topic,headers,fanout。
- Durable：交换机的持久化属性。持久化队列和非持久化队列的区别是，持久化队列会被保存在磁盘中，固定并持久的存储，当服务重启后，该队列会保持原来的状态在 RabbitMQ 中被管理，而非持久化队列不会被保存在磁盘中，服务重启后队列就会消失。
- Auto Delete：交换机的自动删除属性。如果该交换机没有任何订阅的消费者的话，该交换机会被自动删除。这种队列适用于临时队列。
- Redeclare：该交换机是否可以重新声明。当声明的交换机 Durable 或者 Auto Delete 与已经存在交换机不一致，会出现类似如下错误:

Caused by: com.rabbitmq.client.ShutdownSignalException: channel error; protocol method: #method<channel.close>(reply-code=406, reply-text=PRECONDITION_FAILED - inequivalent arg 'durable' for exchange 'jmeterExchange' in vhost '/': received 'false' but current is 'true', class-id=40, method-id=10)

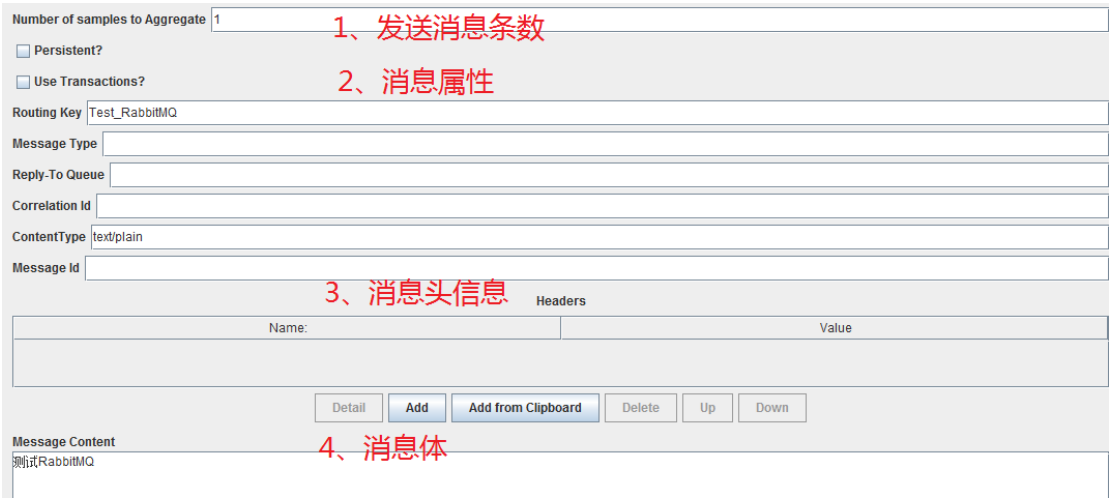
所以 AMQP 插件增加了重新声明属性，当交换机声明与已经存在的不一致时，如果勾选了可重新声明，会将原交换机删除，重新定义一个新的。

3、队列信息。如果需要发送的 MQ 消息是以队列的方式进行消息传递，需要配置。如果不需要则配置“Queue”属性为空即可。具体的属性含义如下。

- Queue：队列名称，默认为 jmeterQueue。
- Routing Key：队列的路由关键字，通过路由关键字的设定，可以完成多个消息统一处理的需求。对于队列的方式，需要与队列名称一致。
- Durable、Auto Delete、Redeclare 属性的作用与交换机类似。
- Exclusive：队列的排他性属性。如果想创建一个只有自己可见的队列，即不允许其它用户访问，可以将一个 Queue 声明成为排他性的（Exclusive Queue）。
- Message TTL：Time-To-Live，消息过期时间，单位为毫秒。
- Expires：队列的超期时间，单位为毫秒。该值必须为正数(与消息 TTL 不同，该值不可以为 0)，所以如果该参数设置为 1000 ，则表示该队列如果在 1 秒钟之内未被使用则会被删除。

4、RabbitMQ 服务信息。RabbitMQ 服务的连接信息。具体包括如下属性。

- Virtual HOST：使用的 RabbitMQ 服务虚拟主机信息。通过虚拟主机可以在服务器上划分多个虚拟空间，起到消息隔离的作用，默认为根目录“/”。
- Host：RabbitMQ 服务所在主机 IP 或名称，使用名称时需要保证 DNS 正确性。
- Port：RabbitMQ 服务所在端口，默认为 5672。
- Username：使用 RabbitMQ 服务的用户名。
- Password：使用 RabbitMQ 服务的密码。
- Timeout：连接 RabbitMQ 服务的超时时间，单位为毫秒。



上图的 4 个方面是具体发送的消息信息。

1、发送消息条数。Number of samples to Aggregate 表示 AMQP 采样器工作一次发送几条消息。

2、消息属性。具体包括如下信息。

Persistent：消息的持久化属性。设置了持久化的消息，即使当时 Consumer 没有在监听，等 Consumer 启动后也能够收到该消息。反之，如果当时 Consumer 没有监听，则无法获取消息。

Use Transactions：是否将发送消息通道设置为事务模式。事务能够解决 Publisher 与消息服务之间消息确认的问题，只有消息成功被服务接受，事务提交才能成功，否则我们便可以在捕获异常进行事务回滚操作同时进行消息重发，但是使用事务机制的话会降低 RabbitMQ 的性能。

Routing Key：消息的路由关键字。队列方式需要与队列名称一致，交换机方式根据实际需求设定。

Message Type：消息类型属性。可以自定义消息的类型，如 String,Object 等。

Reply-To Queue：用于指定回复的队列的名称。该属性是业务流程的需要，不会自动创建指定的回复队列。

Correlation Id：消息的关联 ID。当存在多个计算节点时，通过该属性是的每个线程确定收到的消息与该线程对应。

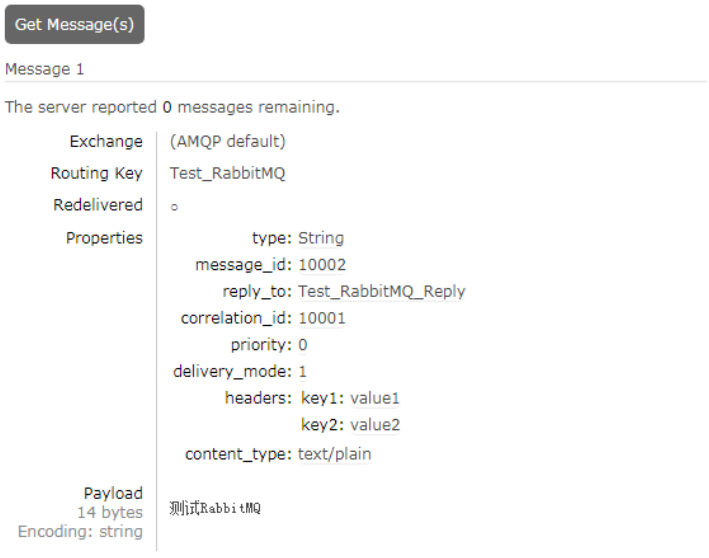
ContentType：消息扩展类型。与 HTTP 中的相同字段意义一致，可以是 text/plain，JSON 等。

Message Id：消息的 ID。通过设定消息 ID，完成业务上的某些需求。

3、消息头信息。Key-value 格式设置头信息，可以是任意的进行业务逻辑的数据。

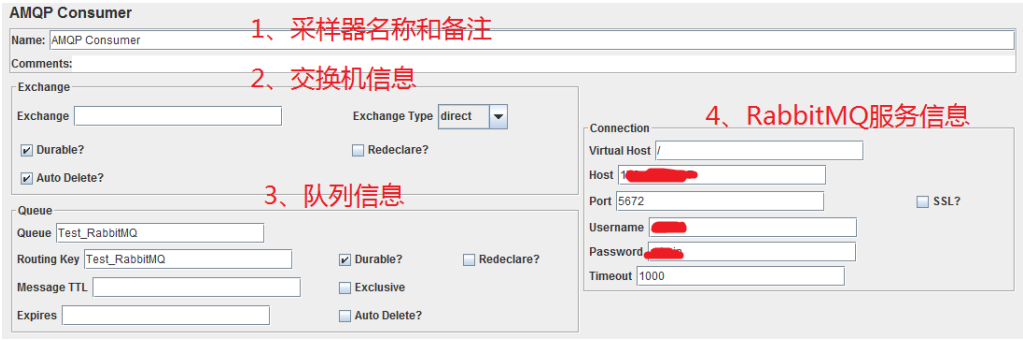
4、消息体。实际发送的消息内容。

通过 AMQP Publisher 采样器发送的消息，通过 RabbitMQ 的 Web 管理界面可以进行查看，下面是在管理界面获取一个消息的示例。

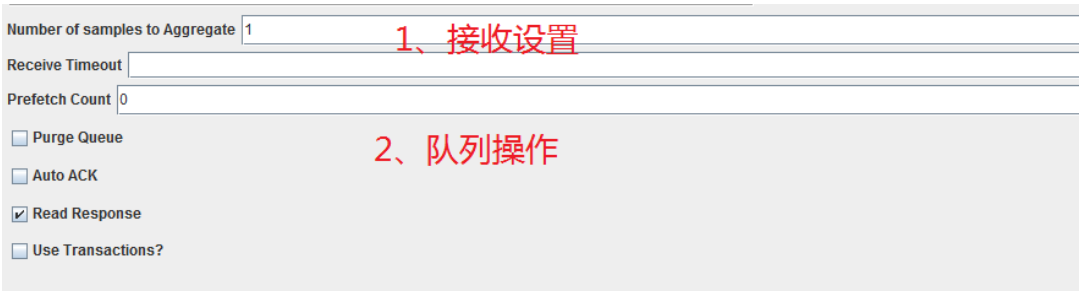


3.2、AMQP Consumer

添加后的 AMQP Consumer 采样器如图所示。大部分参数与 AMQP Publisher 一致，分两部分简单介绍。



上图的 4 个方面是消息的基本信息。包括 1.采样器的名称和备注，2.交换机信息，3.队列信息，4.RabbitMQ 服务信息，参数与 AMQP Publisher 一致。测试中需要按实际需要进行配置。



上图的 2 个方面是具体接收的消息设置。

1、接收设置。接收消息本身的属性。

Number of samples to Aggregate 表示 AMQP 采样器工作一次消费几条消息。

- Reccive Timeout: 接收超时时间。即等待设置时间没有收到消息则退出，单位毫秒。
- Prefetch Count: 直观理解是预先获取的消息条数。Prefetch Count 允许为每个 Consumer 指定最大的 unacked messages 数目。简单来说就是用来指定一个 Consumer 一次可以从消息中心中获取多少条 message 并缓存。一旦缓冲区满了，会停止投递新的 message 给该 Consumer，直到它发出 ack。

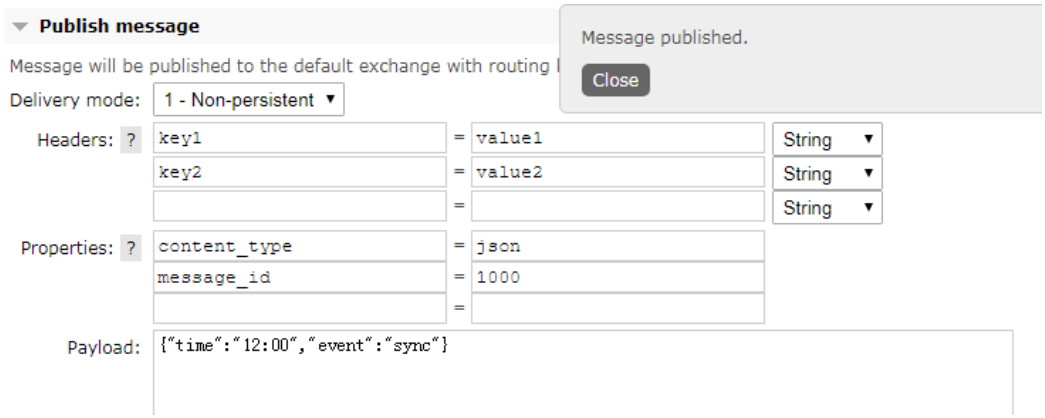
2、队列操作。接收消息时对队列本身的处理。具体包括如下信息。

- Purge Queue: 清除队列。消费消息后将队列清空。
- Auto ACK: 自动应答属性。在订阅消息的时候可以指定应答模式，当自动应答等于 true 的时候，表示当消费者一收到消息就表示消费者收到了消息，消费者收到了消息就会立即从队列中删除。

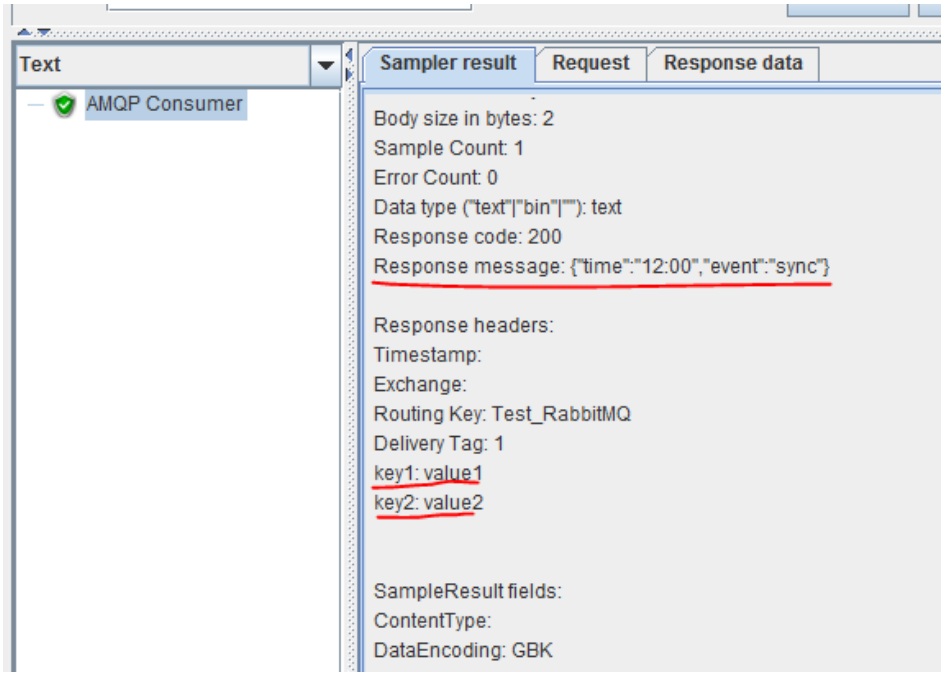
- Read Response：读取消息内容。一般情况都是要读取消息体的。

3、Use Transactions 与 AMQP Publisher 含义一致。

通过 AMQP Consumer 采样器可以获取在 RabbitMQ 的 Web 管理界面发布的消息。下面是在管理界面发布的消息。

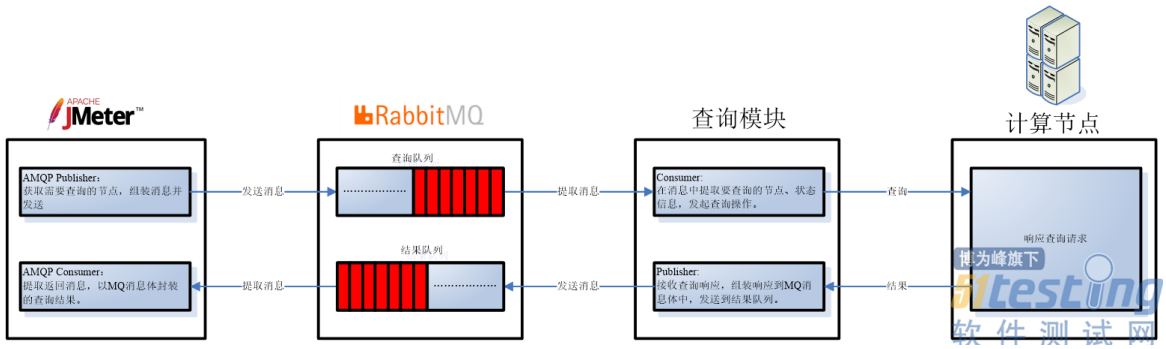


之后，通过 AMQP Consumer 采样器获取该消息。



四、实战 AMQP 采样器

根据实际业务的测试需求，AMQP 采样器可以单独使用，也可以组合与其他 JMeter 元件配合完成测试。下面以一个使用 AMQP 测试业务模块处理能力的完整用例进行说明，业务流程是定时完成对集群中计算节点的状态查询操作，通过查询获取计算节点的状态（存活、资源占用等），查询操作通过 RabbitMQ 消息发送。目前需要测试查询操作的相关性能，设计的主要测试流程如下。



1）通过 JMeter 的 AMQP Publisher 采样器，向 MQ 消息中心的查询队列发送消息，消息中携带节点 IP 地址和需要获取的状态信息。

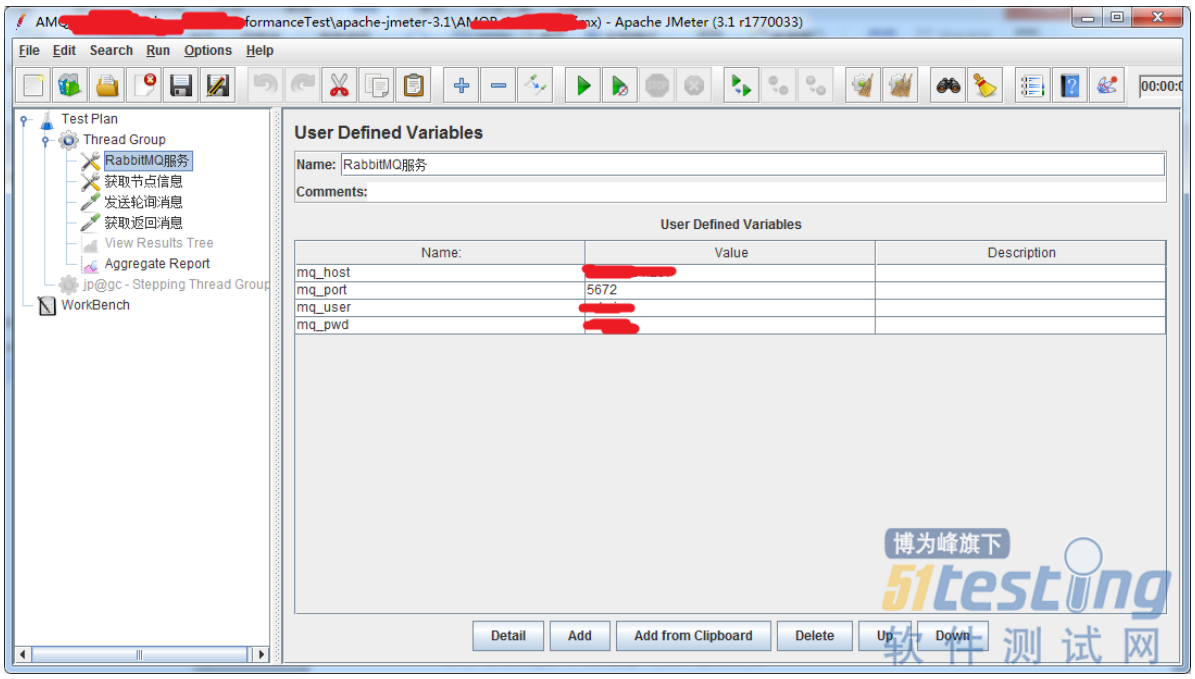
2）查询模块处理消息。将节点 IP、状态信息提取出来， 并通过 Reply-To Queue 字段提取返回队列的名称。之后对该节点发起查询操作。

3）计算节点对查询操作做出响应。

4）查询模块接收查询结果，将结果整理为 MQ 消息体，发送到返回队列中。

5）通过 JMeter 的 AMQP Consumer 采样器，处理返回的消息。

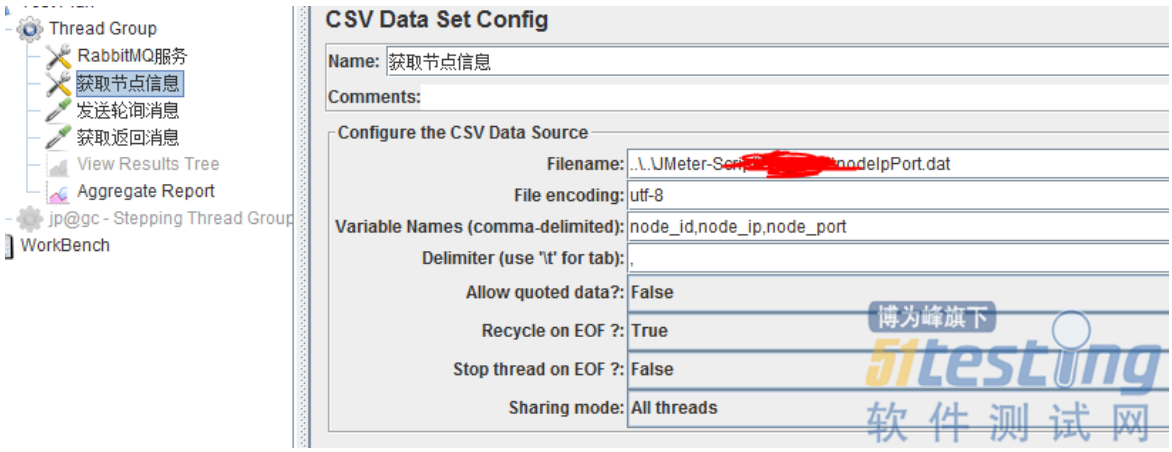
完整的 JMeter 脚本编写如下。



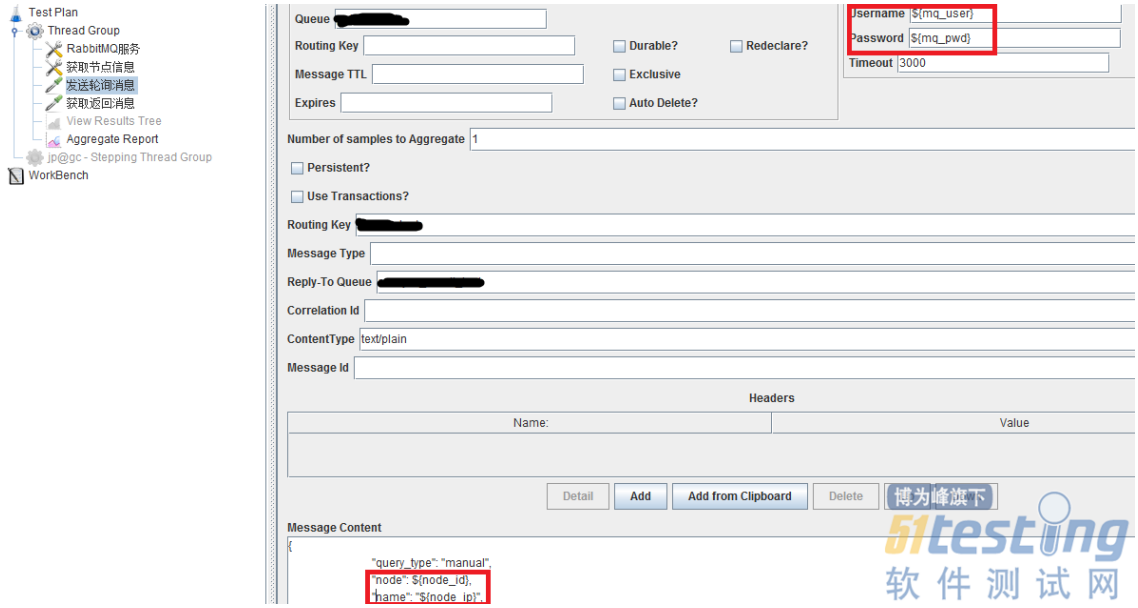
JMeter 元件说明如下:

1）变量定义元件。定义连接的 RabbitMQ 地址、端口、用户名和密码。

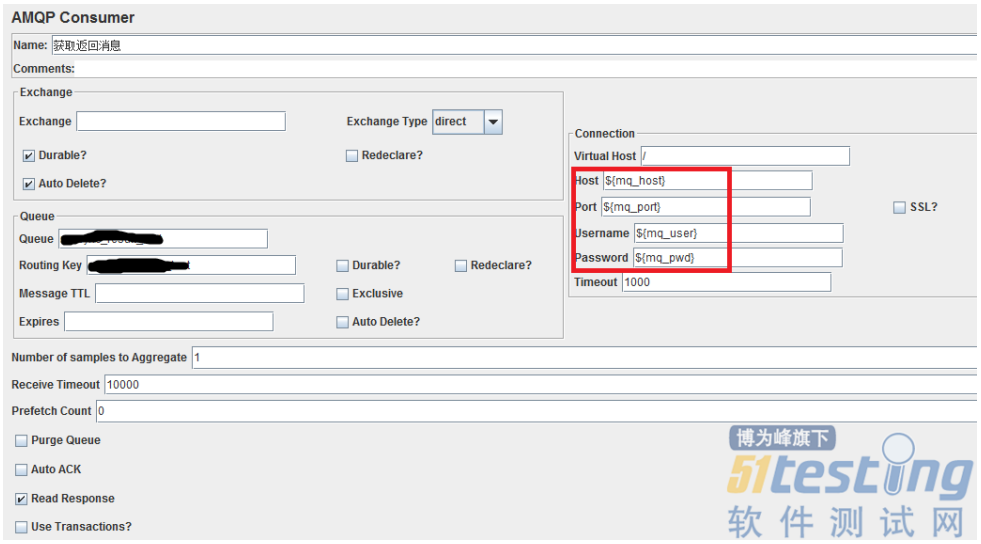
2）获取节点信息元件是一个 CSV Data Set Config，用来在配置文件中获取节点的 IP 地址、端口、管理 ID。如下图。并发线程的个数，与能够查询的节点个数，会影响最终的测试结果。如果仅仅放置一个待查询节点，那么该节点本身的响应有可能成为处理的瓶颈，从而无法验证查询模块自身的极限。



3）发送轮询消息元件是一个 AMQP Publisher 采样器。主要属性配置如下图。



4）获取返回消息元件是一个 AMQP Consumer 采样器。主要属性配置如下图。



5）Aggregate Report 为最终的聚合报告。以一次 50 线程并发为例，测试结果如下。

Label	# Samples	Average	Median	90% Line	95% Line	99% Line	Min	Max	Error %	Throughput	Received KB/...	Sent KB/sec
发送轮询消息	2500	0	0	1	1	1	0	5	0.00%	216.8/sec	216.12	0.00
获取返回消息	2500	218	166	292	393	1251	0	1473	0.00%	213.7/sec	0.42	0.00
TOTAL	5000	109	1	257	292	1226	0	1473	0.00%	424.6/sec	212.09	0.00

由于 JMeter 所在客户端能力有限，仅做脚本正确性验证，不能作为业务模块性能的最终结果（可以说明该查询操作每秒的 TPS 最少为 210）。

五、总结

JMeter 安装容易，使用简单，通过扩展插件能够增加非常丰富的功能，适应各种协议的测试。本文主要介绍了 RabbitMQ 采样器 AMQP Publisher 和 AMQp Consumer 的相关知识。从插件的编译，界面，到实际测试用例中的使用，都给出了详细的介绍。同时，由于插件由源码编译得来，可以根据需要进行定制化，提供更为贴心的服务。