第三十九期(下)　2015年10月

51测试天地

高效地测评软件的用户体验

小明的测试故事系列

说说我在搜狗的收获

填充磁盘空间的工具和方法

零基础完成Loadrunner压力测试

平淡中的伟大

大规模项目将采用CCPM

Robot FrameWork持续集成测试实战

微信扫一扫
测试知识全知道

软件测试人的精神家园：www.51testing.com
中国软件测试黄埔军校：www.51testing.net
软件测试整体解决方案：www.51testing.cn

目 录 Contents

2015 年 10 月

高效地测评软件产品的用户体验

◆作者：李伊祈

摘要： 良好的用户体验是交互产品成功的关键。为了在与用户体验相关的质量要素方面改善产品，首先要能够高效并可靠地测量用户体验。另一方面，量化地测评一个产品的用户体验本身并不是最终目的，而是为了解决实践中的各种问题。本文简略介绍了用户体验调查问卷（User Experience Questionnaire，简称 UEQ），讨论了几种需要通过测量用户体验来解决的典型问题，并以例子展示如何应用 UEQ 这一工具来解答这些问题。本文使用的一般方法可以推广到其他标准化的问卷。

一、引言

成功的软件产品和服务必须具备足够好的用户体验。对于同一产品，不同的人或许会在用户体验这一软件质量指标上有颇不同的评价。这是由于个体特质不同，比方说需要和能力上的差异。因而用户体验是一种主观经验。要测量这一主观体验，一个低成本而高效率的方法是使用经过验证的标准化问卷。

用户体验是一系列独特的质量标准。它不仅包括经典的可用性标准，如效率，可控制性和可习得性，还包括非目标取向的质量标准，如激发度，趣味性，新奇性，情感和审美等（见 Preece, Rogers & Sharp, 2002）。相对于用户体验这一复杂抽象而多维度的概念，这些质量标准较为单一。它们分别描述了用户体验的一个清楚的明确定义的方面，因此容易通过问卷测量。

量化地测评一个产品的用户体验可以出于数种不同的动机。归结起来，它们源于以下几个在实践中自然提出的问题。

持续地监测进展：与旧版本相比，重新设计过的产品版本是否有效地改善了用户体验？

与竞争对手比较：与市面上的直接竞争对手相比，产品在用户体验方面做得怎样？

测验用户体验是否足够好：产品在用户体验方面是否符合用户的一般期望？

锁定有待提升的区域：产品的什么地方需要改善才能提升用户体验？

以下我们将以用户体验调查问卷（User Experience Questionnaire，以下简称 UEQ，

参见 Laugwitz, Schrepp & Held，2008）为例，对应用标准化问卷来调查和解答这些问题展开探讨。

二、什么是 UEQ?

UEQ 是一个语义差异量表（即由若干对反义词构成的双向形容词量表，被测者根据对词的理解和感受，在量表上选择位置来表达他们的观点和评价等主观印象），它的主要目的是实现快速而直接的用户体验测量。这份标准化问卷较短，既可以作为在线问卷使用，也可以用传统的纸笔方式作答。

UEQ 最初以德语写成，诞生于 2005 年。其理论基础是上文中提到的从用户体验的定义中导出的质量标准。研究小组以分析经验数据的方法制定 UEQ，其首要目标就是为了实证地检验从这些质量标准构建出来的维度在实际应用中的有效性。最开始，研究小组请一群可用性专家进行头脑风暴，通过此方式收集了 229 对描述用户体验相关方面的形容词词组（每对反义词为一个题项）。然后，通过专家评估将这 229 对词合并缩减到 80 对，构成问卷的雏形。接下来，这一包含 80 个题项的原始版本被用于多个测评交互产品的调查中，以便收集数据。这些产品包括统计软件包，手机联系地址簿，在线协作软件以及商务应用软件等等。最后，通过主成分分析的方法，从收集到的数据中析出几个主要因子（也就是量表的维度）以及代表这些因子的题项。被析出的 26 个题项则构成了 UEQ。

UEQ 的内在一致性及效度在 11 个可用性测试（共 144 名参与者）和一个较大型的网上调查（722 名参与者）中得到了研究。这些测试和调查的结果表明 UEQ 有足够高的内在一致性（以 Cronbach’s Alpha 度量）以及良好的效度。

由此，UEQ 的现行版本包含以下 6 个维度，以 26 个项的形式出现。

吸引力：对产品的整体印象，即“用户喜欢还是不喜欢该产品？”

代表这一维度的项有：令人不快的/令人愉快的，好的/差的，令人厌恶的/招人喜爱的，不合意的/合意的，吸引人的/无吸引力的，引起好感的/令人反感的

明晰度：产品是否容易熟悉？

代表这一维度的项有：费解的/易懂的，易学的/难学的，复杂的/简单的，一目了然的/令人眼花缭乱的

效率：用户是否不费多余的力气就能够完成他们的任务？

代表这一维度的项有：快的/慢的，低效的/高效的，不实用的/实用的，井然有序的/杂乱无章的

可靠性：用户是否觉得与机器的互动在自己的掌握之中？

代表这一维度的项有：无法预测的/可预见的，妨碍的/支持性的，可靠的/靠不住的，符合预期的/不合期望的

激发度：人机互动是否令人兴奋，是否能激发用户使用交互产品的动机？

代表这一维度的项有：有价值的/低劣的，乏味的/带劲的，无趣的/有趣的，令人兴奋的/令人昏昏欲睡的

原创性：产品是否突破传统，是否有创造力？

代表这一维度的项有：富创造力的/平淡无奇的，独创的/俗套的，传统的/新颖的，保守的/创新的

每一项的答题形式是数值化的语义差异量表，分为 7 个评分等级。例如：

	1	2	3	4	5	6	7	
令人不快的	o	o	o	o	o	o	o	令人愉快的

被测者选择最能代表他的意见的圆圈，其对应分值就是该项的得分[1]。计算同属一个维度的所有项的平均分，就得到用户在对应的维度上的得分。某一维度上的最终得分越高，则被测试的软件在相应的方面表现得越好。

UEQ 的完整问卷（含问卷指导语）、使用指南以及一个专门为数据分析而制作好的 Excel 表格分析工具均可在 ueq-online.org 上免费下载获得。由于 UEQ 是语义差异量表，强调被测者对题项的形容词的理解，因此被测者最好以母语回答问卷。因此，研究小组翻译并验证了几种语言版本（例如汉语，印度尼西亚语，英语，西班牙语，葡萄牙语等等）。这些语言版本（包括问卷、使用指南和数据分析工具）同样可以下载。

[1] 为了防止被测者不读题而只是勾选某一个位置的选项，量表中有的项是消极的词语在左（对应小数值），积极的词语在右（对应大数值，如本例），但有的项恰好相反。与量表配套的数据分析工具已经考虑到了这一点，因此输入原始数据时对于每一项都只需直接输入量表上方对应的数字，数据分析工具会自动转换颠倒的项的分数，使得高得分代表积极的含义。

也正是由于对词语的理解在填答语义差异量表时的重要性，研究小组一直致力于收集各种语言版本的经验数据，在比较实际数据显示的内在结构的基础上检验所翻译的版本的有效性。研究小组所收集的实践数据大部分是由世界各地应用过 UEQ 的软件测评人员提供的。每个语言版本的数据积累得越多，研究小组就越有可能发现翻译的不足之处并往正确的方向修改。尽管 UEQ 的德语版本的信度和效度已经得到大量数据事实的支持，其它语言版本收集到的数据仍未达到做出可靠的严谨的科学论断所要求的数量。由于 UEQ 近年来在全世界得到广泛的应用，如果每个应用 UEQ 的测评人员都能把调查得到的平均值，方差，样本容量和所测试的软件类型等信息反馈给 UEQ 的研究小组，将会大大地推进改善 UEQ 的多语言版本的工作，帮助研究小组持续地提高问卷的质量。

应用 UEQ 不必耗费多少时间和人力物力。通常 3 至 5 分钟就足够让一名参与者阅读完问卷指导语并填完问卷。数据分析的工作也可以用我们提供的 Excel 表格迅速快捷地完成。只要在指定的位置输入收集到的原始数据，表格就会自动计算并显示各统计量和结果图表。

三、典型问题一：持续地监测进展

大部分软件产品在其整个生命周期中都要经历若干次重新设计。软件开发方通常会先交付一个大致上已经完成的初始版本，然后在客户反馈的基础上，在若干个发布周期中逐渐改进。于是，修改后的版本是否比原版本好，或者至少与原版本有可比性（例如，新版本提供更多功能，但也因此而更复杂），就成为一个很自然的问题。

使用标准化问卷工具，例如 UEQ，能较简单地回答这样的问题。测试人员需要做的只是抽取一个有代表性的用户样本，用 UEQ 收集经验数据，然后比较两个版本在 UEQ 的每个维度上的平均得分。

图 1 是同一商务应用软件的两个版本（一新一旧）用 UEQ 所测得的结果[2]。参与者首先分别使用这两个版本完成了他们的可用性测试的题目，然后填答 UEQ。

[2] 与量表上标示的数值不同（可能得分范围为 1 至 7 分）数据分析工具给出的结果是换算后的分数，最小值为-3，最大值为 3，以保持与语义一致，即负分为消极含义，正分为积极含义。

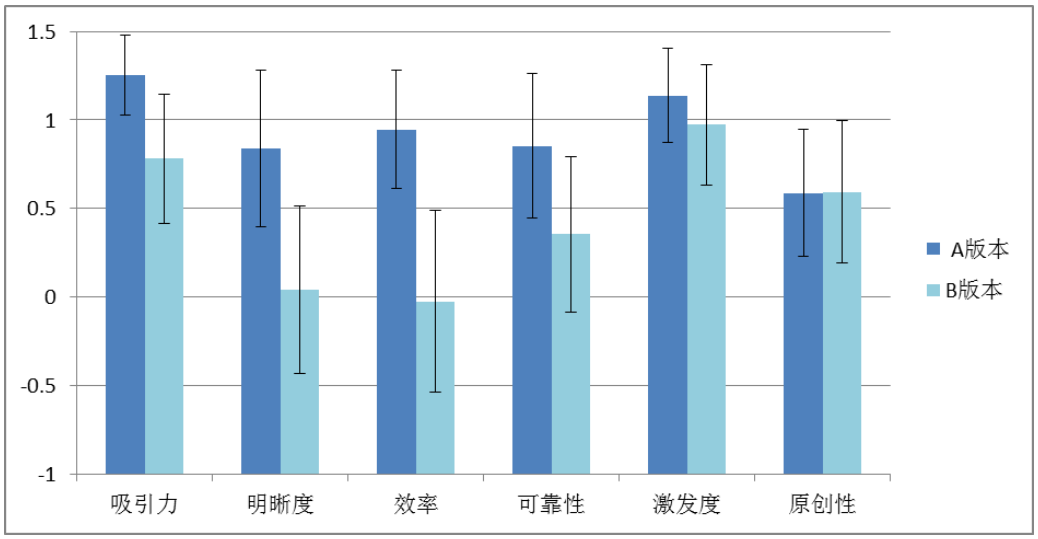


图 1 产品两个版本的 UEQ 结果。误差线代表 95%置信区间

A 版本显得比 B 版本在除了原创性以外的其他所有维度上都要优越。不过，数据显示的差异是实际差异的反映，还是只是由随机偏差引起的呢？尤其当样本较小时，检验观察到的差异是否在统计学意义上显著是绝对必要的。就以上的例子而言，相应的统计学检验（t 检验）显示，新旧版本间的差异在吸引力，明晰度，效率和可靠性这几方面在 5%的水平上显著。

基于这一简便的比较两个产品版本的方法，对产品的用户体验进行持续的监测就非常直截了当。

四、典型问题二：与市场上的直接竞争对手比较

软件产品开发的目标往往不仅仅是要表现优秀，而且要超越市场上的直接竞争对手。一个新产品是否在用户体验方面赢得激烈的市场竞争的问题与前一类问题类似，只不过比较的对象不是同一软件产品的不同版本，而是相互竞争的软件产品。要解答这类问题，最大的困难在于收集竞争对手的产品的用户体验数据。如果是典型的企业内部专属软件，这一般是不可能的，因为实际上无法通过正当手段获取竞争对手的产品的使用权限。但如果是现代的基于网络的应用软件，获取竞争对手的数据往往容易得多，因为竞争对手的产品演示通常至少可以在网络上找到。

这一类型的比较测评需要花费的成本也相当有限。只要解决了获取对方的软件方案的问题，测试程序就和前一类问题基本一样。如果能够联系到足够大的用户群体，就能很快地从这些潜在的参与者中抽取到所需要的样本，这样整个测评过程在几天内就能完成。

五、典型问题三：测试产品的用户体验是否足够好

一个新产品的开发启动后，一个典型的问题是产品是否具备足够的用户体验以满足用户的一般期望。用户的期望是在与其他典型的软件产品的互动中形成的。这些产品并不一定与开发中的产品从属同一范畴。在过去的几年中，一般用户每天都接触现代的互联网网站和现代的交互设备，例如平板电脑和智能手机。这些互动经验极大地提高了用户对专业软件（例如商务应用软件）的用户体验的期望。因此，要解答一个新产品的用户体验是否足够好的问题，可以比较该产品与一个足够大的常用软件产品群体的结果。换言之，一个产品与常用软件产品群体的普遍水平（即基准数据库）比较，其相对位置反映了它的用户体验与普通用户的期望的关系。

在过去的几年间，研究小组为 UEQ 研制了一个基准。这个基准包含了 163 个应用了 UEQ 的产品测评的数据，其中复杂的商务应用软件 98 个，开发工具 4 个，网上商店或网上服务 37 个，社交网络 3 个，手机应用软件 13 个，以及其他类别的软件产品 8 个。由此，我们的基准数据库一共包含了 4818 份答卷的数据。

根据基准，我们在每个维度上定义了 5 个质量等级:

优秀：排名在前 10%内

良好：排名在前 10% 之外，但在 75%之上

高于平均水平：排名低于前 25%，但在 50% 之上

低于平均水平：排名低于前 50%，但在最末的 25%之上

差：排名在最末的 25%内

数据分析的电子表格已经包含了基准的信息。只要输入调查得到的原始数据，电子表格就会自动计算并显示所有统计量，并给出与基准的比较的结果。

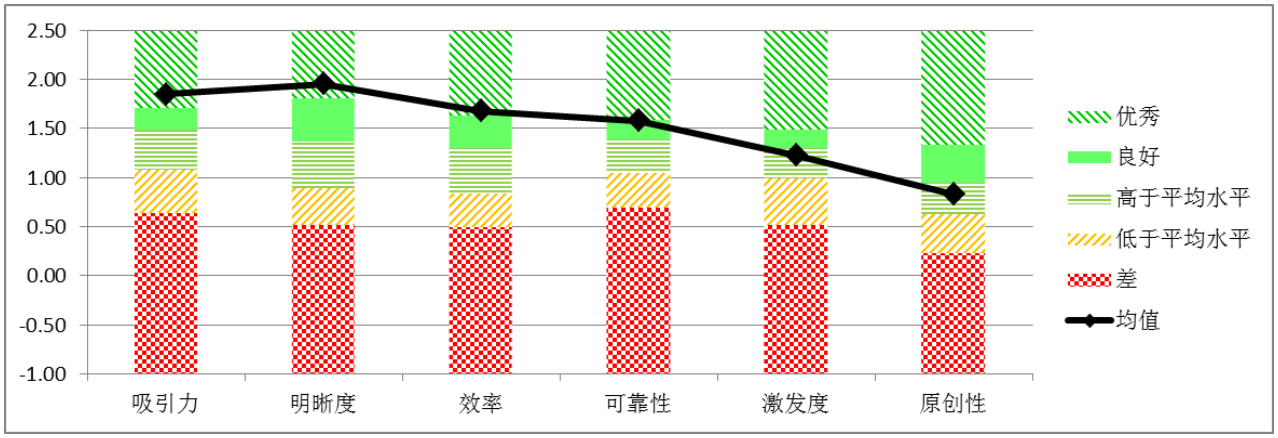


图 2 由供数据分析的电子表格生成的基准图

有了基准，要知道新产品是否具备在市场上优胜的用户体验就不再是难事，只要将 UEQ 应用在对一个较大的有代表性的用户样本的调查中就可以了（一般而言，20 到 30 个用户就能提供较可靠的结果）。将产品在各个维度上的调查结果与基准比较，就能推断出产品相对的优势和弱点。

基准数据库集中了所有研究小组目前为止收集到的数据。虽然这已经是一个很大的数据库，是市场上所有的软件产品的一个样本，但它并不一定在任何国家和地区内都具有代表性，因为不能排除社会和文化等因素的影响。但是，基准库包含的数据越多，来源越广，其它影响因素就越难对基准造成干扰，这时与基准的比较就越有意义。因此，软件测评人员将他们的测评结果反馈给研究小组，不仅仅能促进问卷质量的提高，并且能帮助研究小组制定更准确，更有代表性的基准，从而使问卷以后的使用者得到更准确和更有预测力的结果。

六、典型问题四：锁定有待提升的区域

使用 UEQ 这样的标准化量表来收集关于用户体验的量化数据效率很高。然而，对效率的追求不免带来一些缺点。我们只能在较高的层面上得到关于 UEQ 所测量的维度的信息，但到底产品的哪项特性需要改进以及怎样改进才能提高用户体验等问题有时候无法直接回答。这种测评结果只提供高层面的信息的情况与可用性测试不一样。可用性测试通常能够识别一系列具体问题，例如需要更改的地方，但不能给出一个关于用户如何看待产品的总体印象（尤其是因为可用性测试很耗时耗力，而且只能对较小容量的用户样本进行测试）。

不过，用 UEQ 这样的标准化问卷至少能大致上推测在哪个区域改进软件会带来最

大的效果。对于每个被测评的产品，UEQ 能给出它在 6 个用户体验的质量指标上的表现。从这一模式中至少能推断在哪个区域内更有可能发现软件存在的用户体验问题。

参考文献

1. Preece, J., Rogers, Y., Sharp, H.: Interaction design: Beyond human-computer interaction. Wiley, New York (2002)

2. Laugwitz, B., Held, T., Schrepp, M.: Construction and evaluation of a UX questionnaire. In: Holzinger, A. (ed.) USAB 2008, pp. 63-76, LNCS 5298. Springer Verlag (2008)

3. UEQ Online: www.ueq-online.org (last visited: 20.07.2015)

小明的测试故事系列

◆作者：搜狗测试 Deadwalk

第一篇 Json 导致的事故总结

人物简介:

小明，男，25 岁，一个普通的不能再普通的大学毕业生，刚刚参加工作两年，在某互联网公司担任测试工程师一职。与其他刚毕业的同学一样，爱好看电影、听音乐、爬山……还有倒腾电子数码产品。人生格言是：”我不敢肯定，但是我和胜利有个约定”，目前最大的愿望是：挥洒青春，扎根北京。

大熊，男，32 岁，资深测试工程师，在某互联网公司从事测试工作长达 8 年之久，是小明的 Leader。为人严肃认真，平时上班总是板着脸，同事从未见他笑过。爱好不详、婚姻状况不详，因为体重 180 斤再加上脸比较黑，所以人送外号“大熊”。

今天的故事是这样的….

大熊：小明，今天有个测试任务你测一下。

小明：什么任务?

大熊：浏览器搜索栏推荐列表的测试任务。

功能需求：当用户鼠标点击搜索栏时，搜索栏会向搜索服务器请求最热门搜索词，服务器返回内容后，浏览器将内容以下拉列表的方式展示出来。

小明：好的。

三天后，该功能测试完毕上线了……

大熊：小明，你给我过来！(四川话口音)

小明：老大，什么事？

大熊：刚接到反馈，线上用户只要打开搜索栏就会主进程崩溃，你是怎么测的！

小明：我那天测的时候没有这个问题啊。

大熊：你当时都测试了哪些用例？

小明：不就是点击搜索栏能够弹出下拉列表嘛，另外还测试了：

返回条目的个数；

点击下拉列表内容能够跳转到对应搜索结果；

切换别的搜索引擎不会有下拉列表；

服务器返回的内容是中文、英文、数字的时候，下拉列表内容也是对应的内容。

服务器返回的数据超长的情况。

大熊：服务器返回的数据格式是什么？有没有测试格式异常？

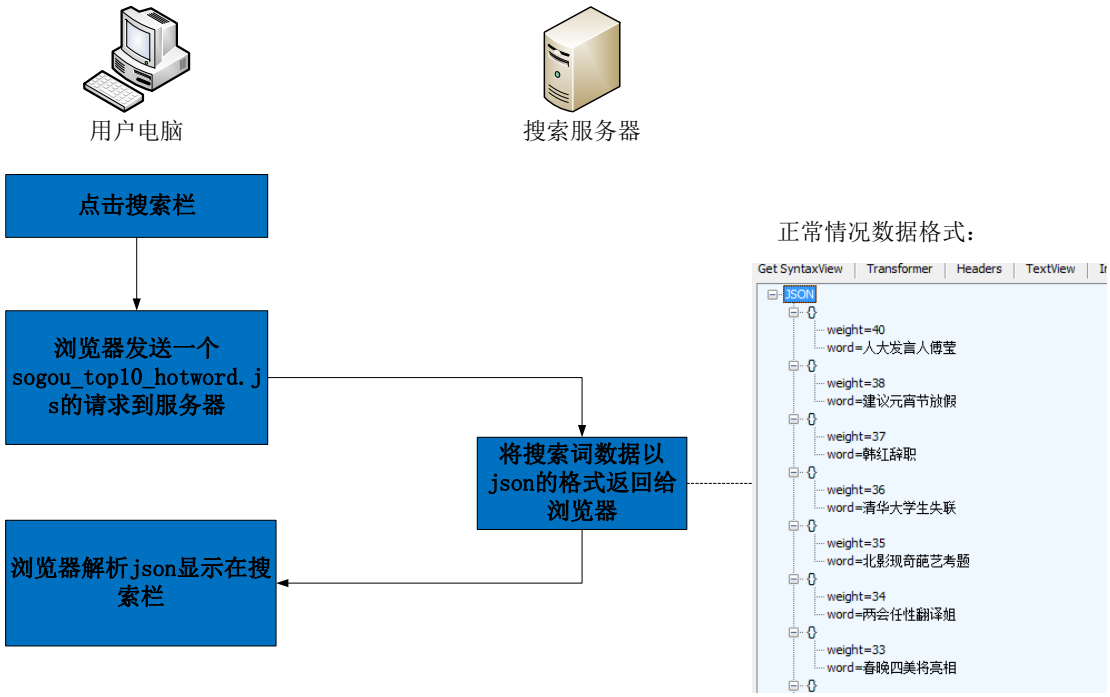
小明：服务器返回的数据格式是 json 格式，没有测试格式异常。格式异常这种情况

测它有意义吗？怎么可能会出现返回的数据异常啊，这个服务器都是我们公司自己的服务器。

大熊什么话都没说，让小明先去查明崩溃原因。

不久之后，小明了解到了崩溃原因，原来是：

正常情况下：



出现问题时：因为搜索服务器出现了异常，返回给浏览器的数据格式不是 json，而是一段 html，而浏览器仍然当做 json 去解析，所以发生了异常崩溃了。

小明：老大，我知道错了，Json 格式异常也需要测。

大熊强忍胸中的怒火，在电脑上打开了一份文件，那是一份很长的事故列表，其中的内容是这样写的：

2013 年 10 月，一款叫做桌面助手程序在获取天气预报数据时，由于服务器返回的 json 格式数据异常，导致桌面助手频繁崩溃。该问题造成了比较大的影响，Leader 被罚 1000 元，测试团队上下做了深刻的反省和总结。

2012 年 3 月，浏览器升级程序在下载一个升级策略.dll 文件时 ，该文件在传输过程中被江西运营商加入了一段 html 的广告，导致升级程序加载.dll 文件时异常，造成江西一带用户无法升级。

2010 年 11 月，公司大 BOSS 川总反馈，在搜狗浏览器搜索栏输入双引号，浏览器崩溃。崩溃原因是返回的数据因为双引号未转义原因，将 json 数据格式配对破坏，导致解析失败崩溃。事后测试组 Leader 和测试人员被当季度罚绩效考核不合格。

……..

看到这份列表，小明半天没有说出话来。

大熊问到：你从这件事得到了什么总结？

小明思考片刻，理了理头绪，娓娓道来：

测试客户端时，要考虑服务器出现异常情况时，不会对客户端造成影响，例如服务器 502 挂掉了。

测试功能时要了解到网络传输过程中的数据格式，除了使用等价类、边界值考虑常见的中英文数字等数据之外，还要对数据格式异常进行测试，例如：json 数据缺少{;xml 数据缺少<等情况

接第 2 点，还要考虑返回的数据为空。

测试功能时还要考虑到网络传输过程中的异常情况，如断网、直接拔网线等。

大熊点点头，继续问道：如何构造这些异常情况呢？

小明：不知道….

大熊：用 Fiddler 拦截请求，具体用法去查知识库！另外，本季度 PM 成绩从 B 开始，以示惩罚。

后来，该事故的处罚结果为：大熊作为 Leader 连带罚款 1000 元，小明季度奖金取消。

第二篇 Fiddler 超时处理

前文回顾：

小明在测试任务浏览器搜索栏推荐列表的测试中，因为没有考虑到 json 格式异常的情况，导致浏览器上线后出现大范围的崩溃，大熊连带被罚 1000 元，小明绩效考核扣

除一个季度的奖金，本人对此也做了深刻地反省。

今天的故事是这样的…

大熊：小明，搜索栏崩溃这个问题开发已经改了，你回去好好测啊，一定要注意测异常格式！

小明：老大，放心吧！

一天后，该功能上线，再次出现崩溃问题……

此刻，小明低着头站在大熊的工位旁，无言以对。

大熊听到这个事情，起先是诧异，然后是无奈，现在是愤怒。

大熊：你搞撒子嘛，老人跟你说了要测试异常字符的嘛，怎么还有这个问题？(四川口音)

小明：老大，我确实测了异常格式的，当时测的时候没有问题。

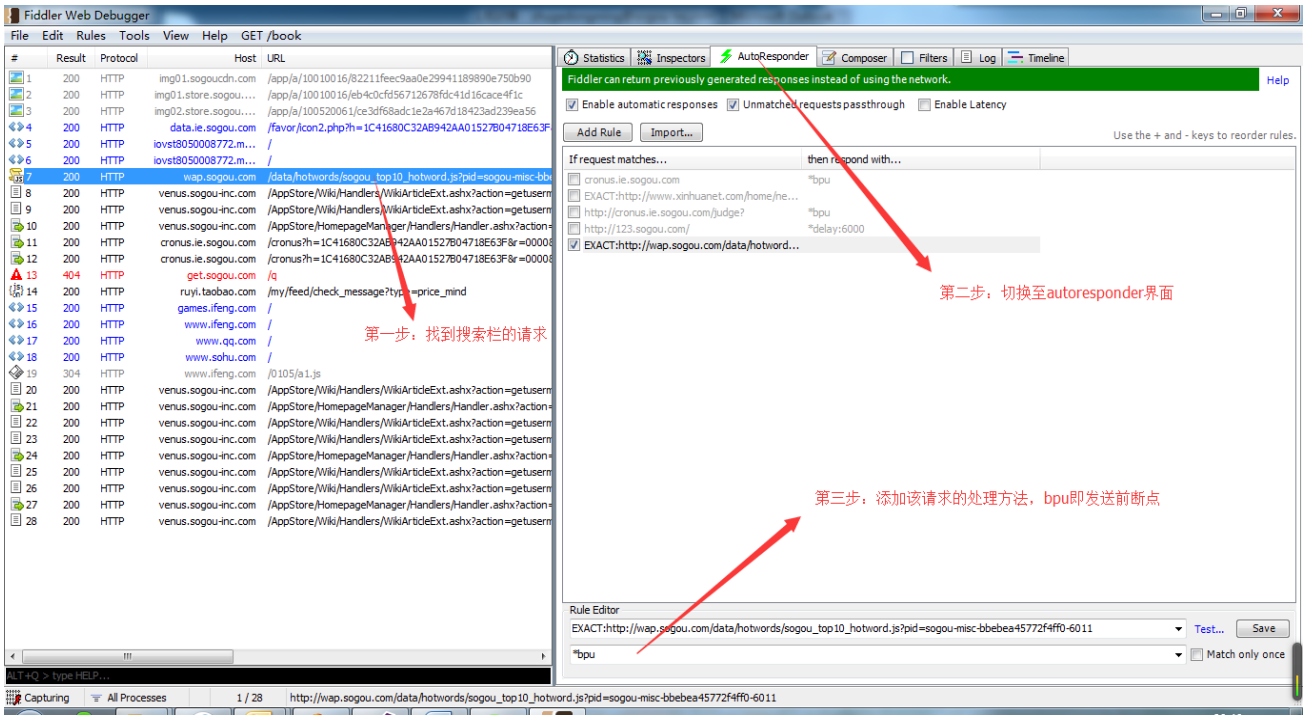
大熊：那你说为什么一上线还是崩溃呢？

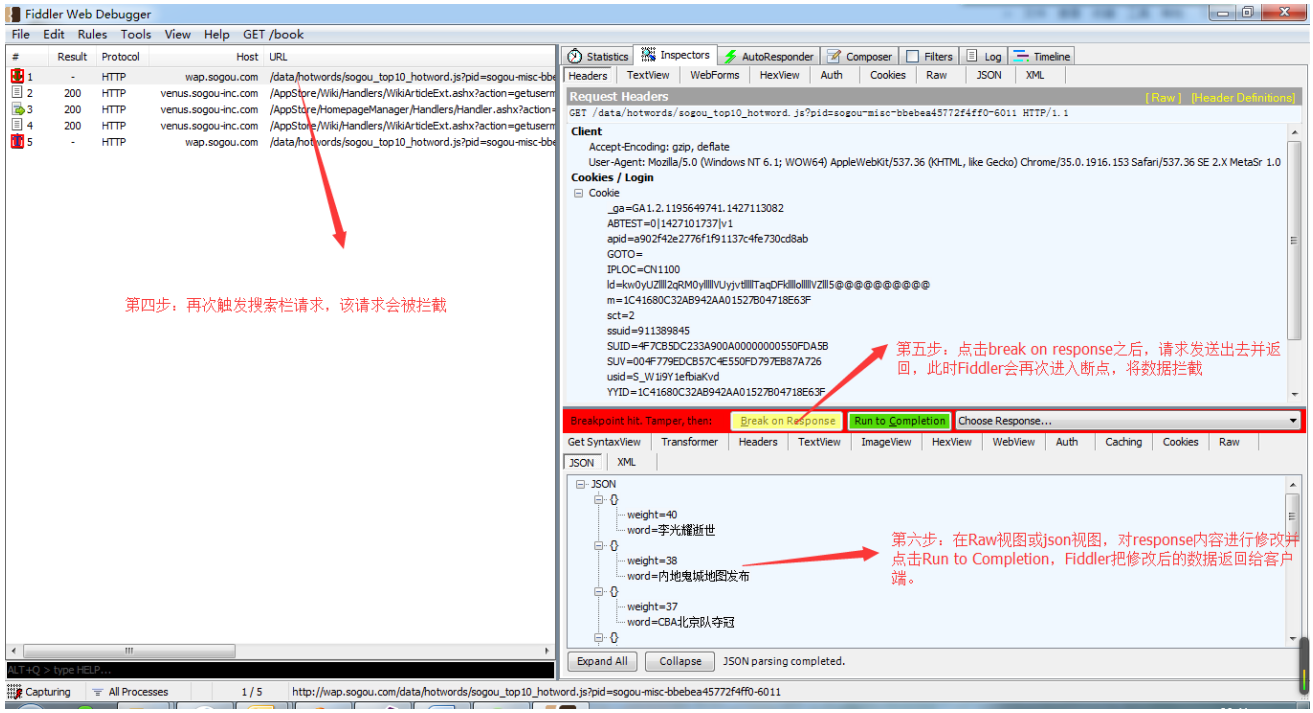
小明低下了头，心中充满了委屈。

大熊：你怎么测的？

小明：我用 Fiddler 拦截住 sogou_top10_hotword.js 的请求，然后在 Fiddler 中把 Response 的返回值改为非 json 格式……

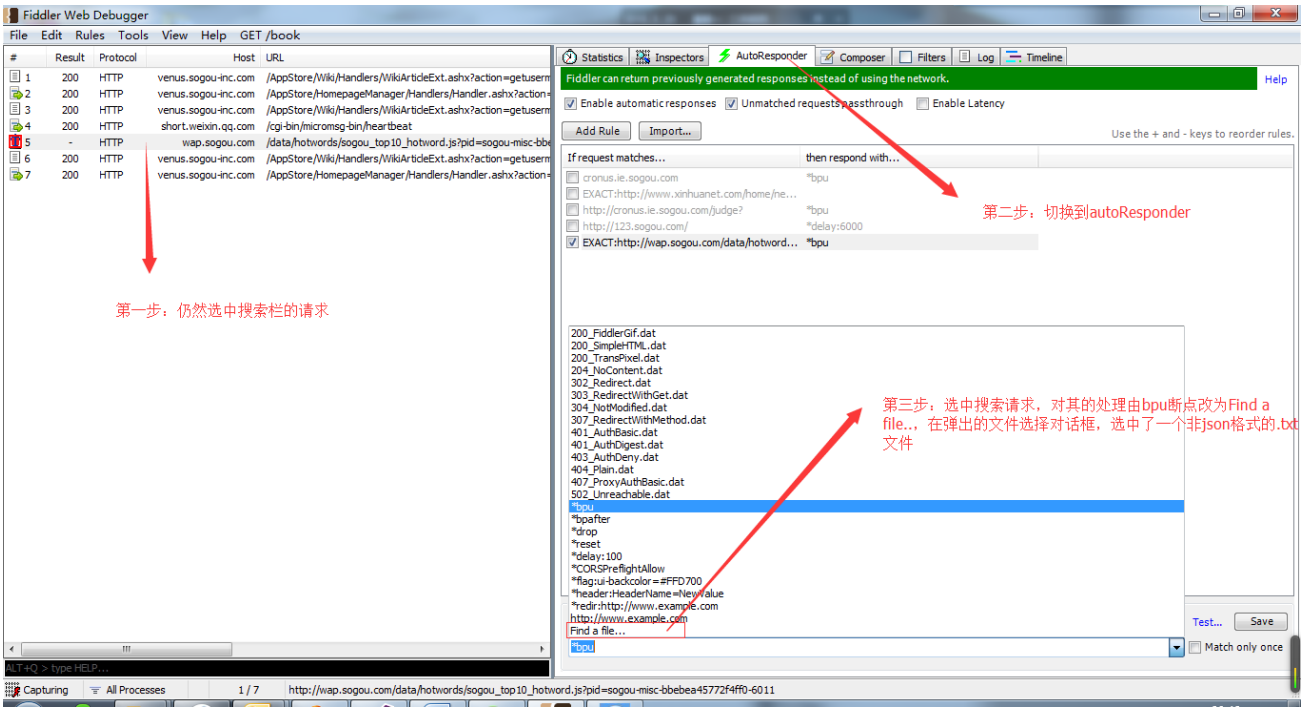
大熊照着小明的做法试了一下，果然没有崩溃：





多年的测试经验告诉大熊，问题可能出在 Fiddler 的拦截处理方法上。

大熊换做如下方法：



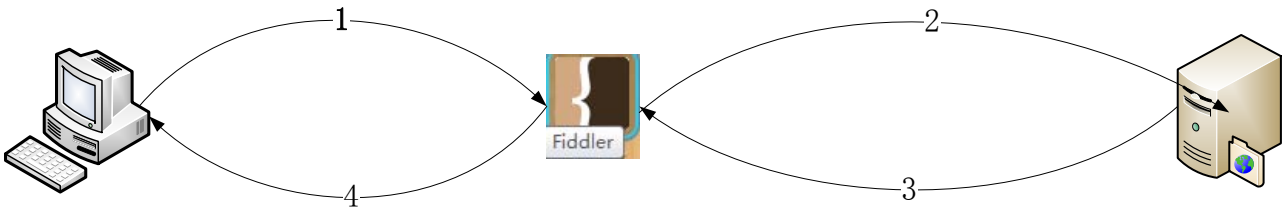
如此操作之后，崩溃问题重现了。

小明见状，终于知道是自己的问题。他虽然很惭愧，但是求知的欲望还是占了上风。

小明：“老大，我知道错了，我认罚。但是我还想请教下您这是为什么？第一种测试方法为什么没有问题呢？”

大熊刀子嘴豆腐心，见小明如此虚心请教，也不由地平静下来，毕竟只是发脾气对结果没有任何帮助。

大熊在纸上画了起来....



1).当用户点击搜索栏后，浏览器触发请求，如上图 1 过程。

2).Fiddler 用 bpu(发送前断点)拦截下请求，如上图 Fiddler 位置出断点。

3).Fiddler 中选择 Break on response(接受请求断点)，即为将请求放行，发送给服务器；当服务器返回给 Fiddler 后再拦截断点，即上图 2 和 3 过程。

4).小明在 Fiddler 上修改返回内容再点击 Run to Completion，则为将请求放行返回给客户端，如上图 4 过程。

问题原因:

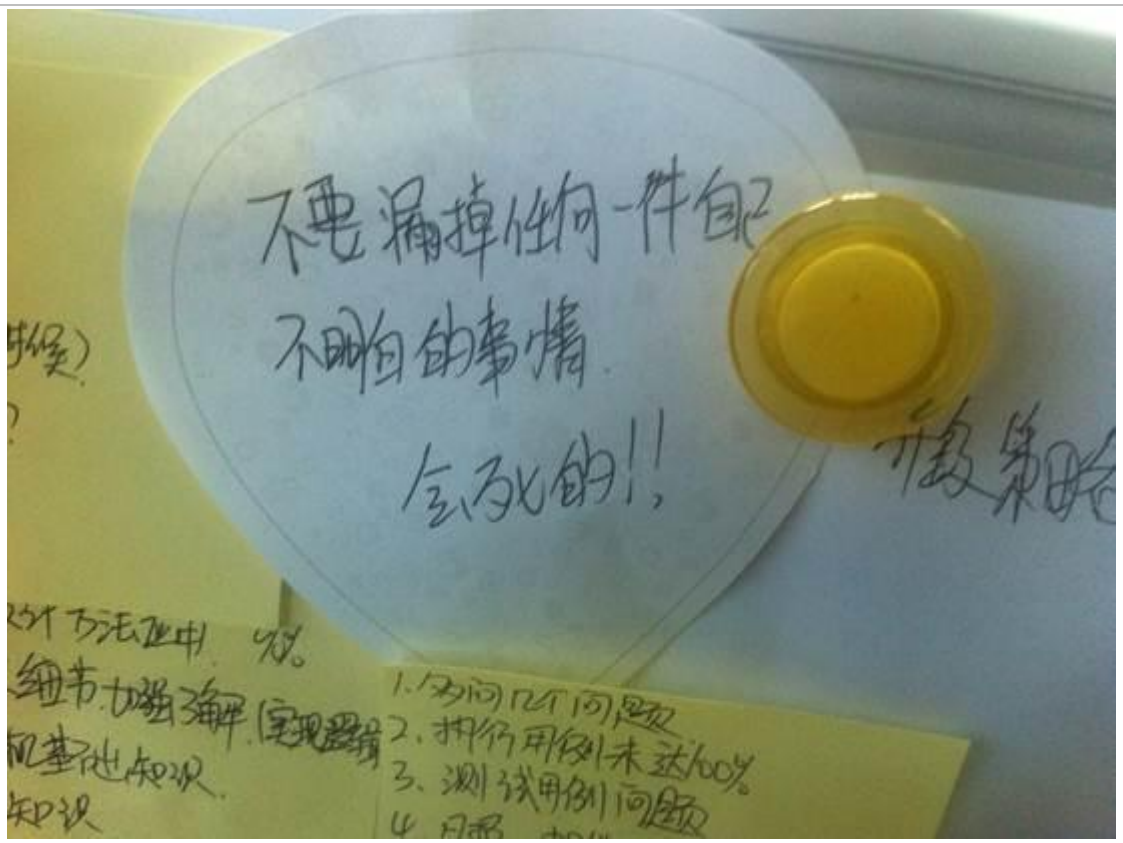
客户端有一个超时机制，即发送的请求 2 秒内没有返回则不予处理。因为小明在 Fiddler 上修改数据，所以花费时间超过了 2 秒，所以客户端未进行处理，不会崩溃。大熊用的办法是让 Fiddler 自动直接按照规则返回.txt 文件中的内容，时间极短，没有超时，所以触发了客户端的处理逻辑进而崩溃。

小明：谢谢老大的指导，我以后不会再犯这个错误了。

大熊看着小明深深地鞠了一躬，想想自己当初也是这么一路走来的，训斥的话到了嘴边又咽了下去。

事后，该问题仍然判为严重事故，Leader 连带罚款 3000 元，同时小明该季度绩效考核成绩不合格，并且给每位同事买饮料以示自罚。

小明心里明白，虽然这次事故让自己吃了点苦头，但是长远来看对自己是一种成长，于是他写了张贴纸贴在工位上以提醒自己。



第三篇 搜索栏第一候选搜索无效的事故总结

前文回顾:

小明在搜索栏热词的测试中，因为不知道请求返回超时机制，导致浏览器上线后再次出现大范围崩溃，大熊连带被罚 3000 元，小明该季度绩效考核成绩不合格，并且给每位同事买饮料以示自罚。

今天的故事是这样的……

大熊：小明，6.0 搜索栏要加动效，你回去好好测啊，不要再有线上崩溃的问题了，如果再有，你懂的(@￣—￣@)……

小明：嗯嗯，知道了，老大。

……

小明：开发，麻烦看一下这个 bug……

开发：好的

……

开发：小明，为了改这个 bug，我把搜索栏的代码整理了下，帮忙回归下吧，回归范围是：下拉列表显示（包括搜索引擎列表、热词列表），点击选中项是否正确响应。多谢 O(∩_∩)O~

小明：热词的轮播和 pingback 要不要看一下？

开发：轮播不用，pingback 的话可以看一下。

小明：那对 GDI 泄露有影响吗？

开发：应该没有，那块代码没有动，不过你要是有时间，可以看一下。

小明：好。你到底改了什么呢？

开发：之前搜索引擎是一个菜单，热词使用 GUI 画出来的，现在把搜索引擎和热词都变成列表，它们统一使用一个控件去画里面的每一项。

小明：哦哦，我就回归你之前提到的那几个地方就可以了，是不？

开发：嗯。

两天后，该功能测试完毕上线了……

大熊：小明，过来！

小明：（弱弱的快步走过来）老大(⊙o⊙)…

大熊：内测反馈，搜索栏输入关键字，第一候选回车搜索无效，怎么回事？？？

小明：(⊙ o ⊙)啊？我回去看一下……

五分钟后……

小明：老大，确实不好用，包括热词是一样的~~~~(>_<)~~~~，我先提个 bug 给开发，了解下原因……

大熊：嗯，你懂得 o(￣ヘ￣o#)!

小明：哦(。··)ノ……

1 小时后，开发改完了 bug，小明也了解到了原因，原来是：

回车搜索是编辑框先响应键盘上下键及鼠标 hover 时的消息，然后传给了列表，列表又响应了回车触发了搜索，而点击搜索是列表直接响应触发了搜索，这两个路径触发搜索的原理是不一样的，开发由于整理了代码，把列表响应回车触发搜索的判断语句写成了 0<a< span="">≤9,而正确的写法是 0≤a≤9。</a<>

大熊：你当时是怎么测的？

小明：我以为点击和回车搜索都是一样的，所以把点击都验证了，而回车只是随机验的，没有验到第一个（弱弱的答道）……

大熊：为什么以为点击和回车搜索是一样的呢？

小明：呃(⊙o⊙)…

大熊：现在有什么感想？

小明：1.不要自己想当然，遇到不确定的事情，要和开发确认。

2.对于这种开发重构代码的问题，要把测试用例中与该改动相关的主路径测一遍

3.以后在测试过程或测试范围回归的过程中，遇到边界值这种问题时，重点把边界值测一下

4.不要因为需求或者改动小就大意，越小的地方越容易出错

事后，该问题依然判为严重事故，Leader 连带罚款 3000 元，同时小明该季度奖金再次取消，并且给组内每位同事买一星期酸奶以示自罚。

年轻的小明仍然在路上……

本文中部分内容为虚构，如有雷同，纯属巧合。

说说我在搜狗的收获

◆ 作者：搜狗测试 Deadwalk

岁月如梭，时光飞逝，一转眼我已经在测试这一行业摸爬滚打 7 年了。作为一个工作 7 年的 tester，我想说说成长，说说我在搜狗的成长，说说我在测试人生的成长。

讲故事前先做一下自我介绍，我叫诸葛东明，31 岁，搜狗浏览器测试组 Leader。2006 年开始北漂生活，2008 年加入搜狗，见证了搜狗浏览器的诞生，然后陪着它一起走到今天。

7 年之间，我已经数不清上线了多少个版本、运行了多少遍测试用例、提交了多少个 BUG、奋战了多少个通宵达旦，但是忘不了每次战友离开时那淡淡的忧伤。

曾经多次有人问我，为什么你还不跳槽？

因为一份坚持，从我的 leader、从我的 BOSS 身上学到的那份坚持。也许它听起来有点冠冕堂皇，但听我慢慢道来。

故事之一：

时间大概是在 2009 年。在浏览器各项指标中，项目组上下一直极为重视浏览器的稳定性指标，也就是浏览器的崩溃率。为了改善这一崩溃率，只是通过常规的手工测试手段是保证不了的，这需要使用自动化技术。

起先，我们使用了 BHO 技术来完成浏览器内核的自动化测试，自动化脚本可以使得浏览器自动地进行前进、后退、导航和刷新等操作。但是这一技术的缺陷是无法进行浏览器内核以外功能的自动化操作，所以随着新功能不断地增多，BHO 技术已经无法满足。

之后，我们尝试使用业界比较成熟的 QTP 进行自动化测试，通过控件识别+键盘快

捷键等方式，内核之外的功能也逐步纳入到稳定性测试之中。但是随着浏览器 2.0 版本的发布，内核变为 Trident+Webkit 双内核，QTP 无法有效识别 Webkit 内核的控件。

此外，更多的困难也随着项目的行进不断地暴露出来:

QTP 软件体积庞大，随着测试机由两台不断扩充到几十台，每次部署到新环境非常耗时间。

QTP 自动化脚本日积月累，已经庞大到几十个组合动作，脚本维护成本巨大。

QTP 所使用的 vbscript 脚本无法支持多线程等功能，这使得自动化脚本所能操作的对象比较有局限性。

我个人的测试工作已经忙得没日没夜，同时还要维护自动化脚本。

更为致命的一点，自动化脚本的作用受到质疑。每次浏览器上线前，自动化测试没有发现什么问题，但是上线后仍然有大量的崩溃问题，这些崩溃问题在测试环境没有被提前发现。

诸多的困难之下，我逐渐对自动化丧失信心，开始质疑这一方法的可行性。在我的学习经历中，所接受到的知识是自动化技术是用于解决重复性的、有预期结果的测试用例回归，我们只能让机器按照我们提前设定好的步骤去执行，然后对比实际结果是不是符合预期。而使用自动化技术进行随机性的操作去发现未知的问题，这行不通。

因为这个问题，我和我的老大鲁剑争论了多次，我坚持认为证明自动化发现不了未知的问题，过去一年多的实践就是最好证明。而鲁剑始终坚信自动化可以发现影响浏览器的稳定性问题，未来可以作为评估浏览器的上线标准。

我放弃了，但是鲁剑没有放弃。

他后来做了两件事：第一，让测试开发林飞使用 python 重写稳定性自动化脚本，以此来克服 QTP 的诸多问题。第二，让林飞每天查看浏览器的崩溃栈，根据栈信息分析可能的操作路径，然后将这些操作路径转化为自动化脚本。这项工作大概持续了一个月之久，林飞通过每天不断地动作补充，建立了三百个庞大的浏览器动作组合脚本。基于 python 面向对象的特性和更为高效的随机算法，稳定性脚本在效率、问题发现能力和脚本可维护性上都取得了进步。

通过这个脚本，我们多次在测试阶段就发现了潜在的崩溃问题，避免了问题的遗

漏。这一通过随机浏览自动化测试的方式，已经成为国内浏览器厂商必备的评估方法。

故事之二:

时间大概也是在 2009 年，距离搜狗浏览器第一个版本上线后的半年。有一天，公司突然发全员邮件，告知王小川已不再管理搜索团队，只负责桌面团队的管理工作。这意味着什么，小川管理的团队拦腰砍半，原因可能是老张 Charles 和小川意见不合，不支持研发搜狗浏览器。

一般人遇到这种情况，自己努力工作却不被上级老板支持，也许就此放弃收拾收拾就走人了，但是在我眼里的小川是这样的:

他不但没有消极应对，反倒在浏览器上加大了精力投入。他那时每天会花 2 小时对浏览器进行测试，经常会报一些路径复杂的 BUG 给我们。作为测试你会懂那份压力的，你的大 BOSS 掌管着一个上百人的公司，他不但每天要处理各个产品线的管理事务，而且每天都在测试你测试的产品，发现你发现不了的 BUG！与小川共事多年的 Better 说，从未见过小川在浏览器上投入如此大的精力，他甚至已经不管那时的搜狗音乐盒项目了。

那时候我下班都很晚，基本上是晚上 11 点以后。工位上已经没有多少人，只剩下浏览器的开发和测试。小川也没走，他有时会走到我们的工位上转转聊聊，自邮件一事之后，我见他嘴角多了不少裂纹，那是着急上火的原因。

隐忍一年之后，搜狗浏览器 2.0 上线。上线当天还出现了一点小意外，小川在向 Charles 演示浏览器的时候出现了 BUG，我们紧急赶到 Charles 的会议室来处理，我虽然难忘在大 BOSS 前处理 BUG 的窘境，但是更加难忘小川紧坐在 Charlse 旁，不断地讲解着浏览器的作用和价值，眼神中流露着那份渴求得到认可的期望。

最终浏览器被老板认可，小川重新执掌搜索部门，搜狗得到新生，分拆独立之后，从 200 人的搜狐研发部门，一路扩充为今天 3000 人的搜狗公司。

逆境之下，坚持不懈，不忘初衷。这就是我在搜狗最大的收获。

最后，我想分享一段摘文来结束今天所讲的故事，希望对各位小伙伴们有所启发。

“未来的某一天，他们会回顾这段共同度过的时光，对于那些痛苦的时刻，只是过眼云烟，或者付之一笑，他们会把这段时光看做人生中奇妙的巅峰时刻。”

--摘录自:[美]沃尔特·艾萨克森. “史蒂夫·乔布斯传

填充磁盘空间的工具和方法

◆作者：搜狗测试 UncleCui

平时在测试时（比如测试安装、保存文件到本地等），可能需要构造本地硬盘空间不足的情况，这里介绍一个方法。

一、fsutil 命令

1.以管理员的身份运行 cmd

2.进入相应的盘符

3.命令：fsutil file createnew hello.txt 100000，其中“hello.txt”创建的文件，“100000”创建的文件大小

4.一般创建文件的大小为盘符空间大小的从右向左数第 8 位小 1 就可以了

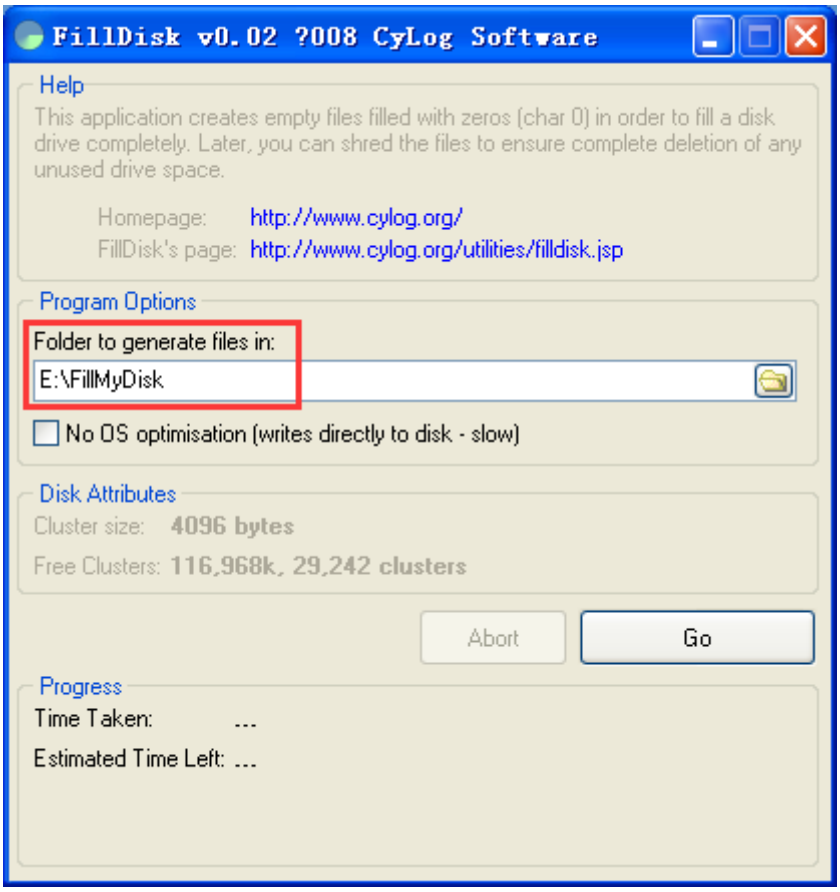
这个方法有个小缺点，就是要自己计算生成文件的大小，有一点点麻烦。

二、FIllDisk.exe

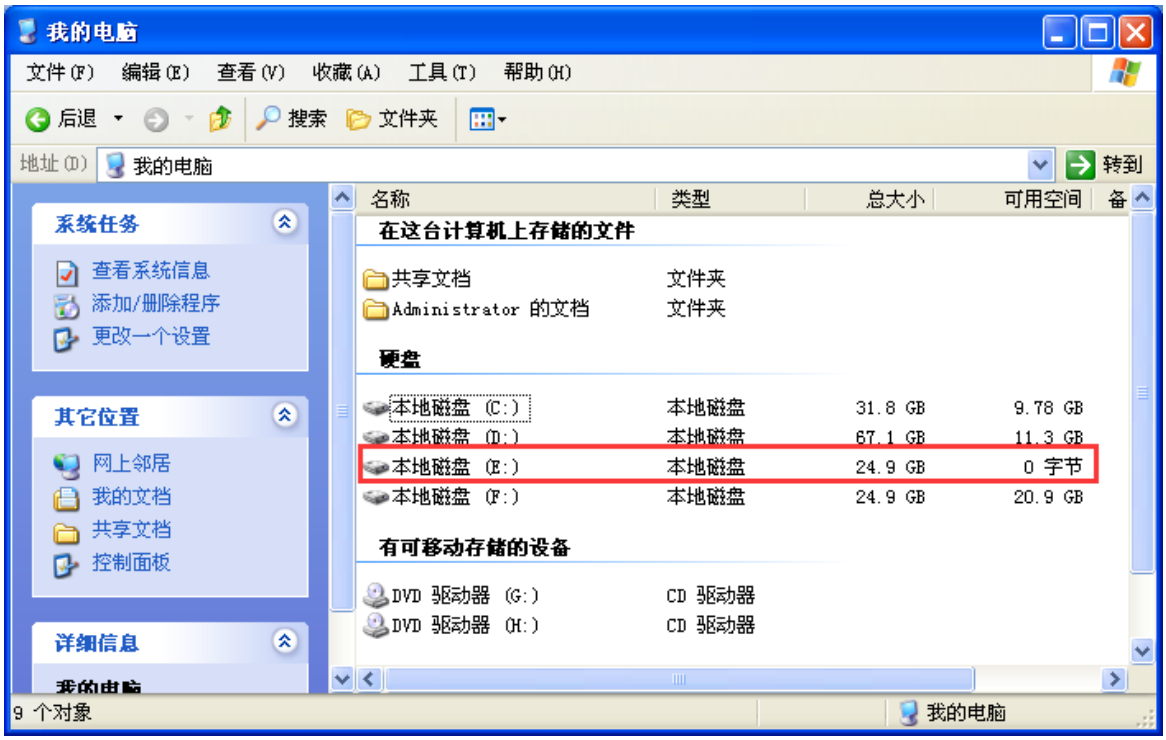
这里再介绍一个工具：FillDisk.exe

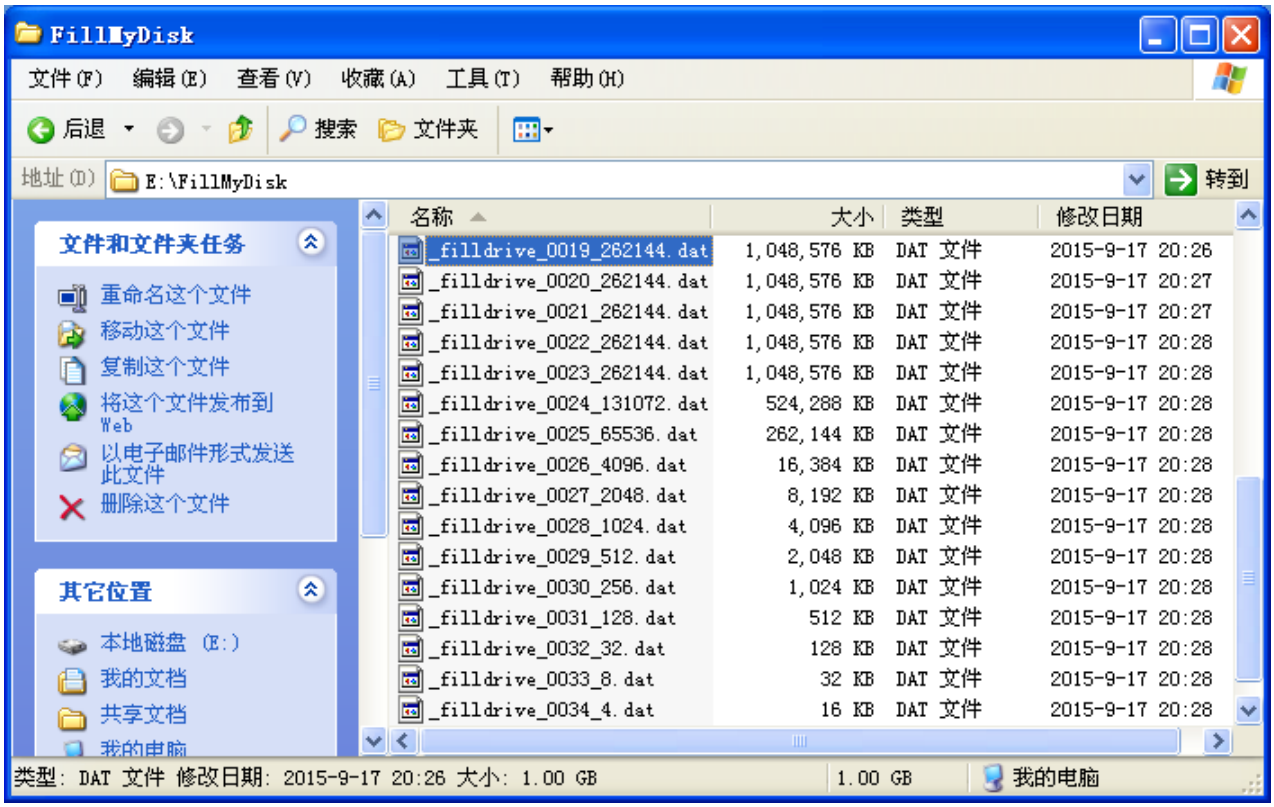
官方介绍：FillDisk is a simple utility that fills up a disk with data. It starts with1GB files, then drops to 512MB, 256MB and so on until it fills up completely the disk with files.（FillDisk 工具的工作原理就是先生成 1GB 的文件，当磁盘空间不足 1GB 时，就生成 512MB 的，再不足就生成 256MB 的，以此类推，逐渐生成小文件，直至磁盘空间被全部填满。）

使用方法很简单，打开 FillDisk.exe，填写文件的生成路径（路径要已存在，否则请先在本地手动创建，否则会报错），点击 Go 按钮就可以了。


FillDisk v0.02 ?008 CyLog Software
Help
This application creates empty files filled with zeros (char 0) in order to fill a disk drive completely. Later, you can shred the files to ensure complete deletion of any unused drive space.
Homepage: http://www.cylog.org/
FillDisk's page: http://www.cylog.org/utilities/filldisk.jsp
Program Options
Folder to generate files in:
E:\FillMyDisk
No OS optimisation (writes directly to disk - slow)
Disk Attributes
Cluster size: 4096 bytes
Free Clusters: 116,968k, 29,242 clusters
Abort
Go
Progress
Time Taken: ...
Estimated Time Left: ...

这是执行结束后的效果:


我的电脑
文件(F) 编辑(E) 查看(V) 收藏(A) 工具(T) 帮助(H)
后退 搜索 文件夹
地址(D) 我的电脑
转到
系统任务
查看系统信息
添加/删除程序
更改一个设置
其它位置
网上邻居
我的文档
共享文档
控制面板
详细信息
名称 类型 总大小 可用空间 备
在这台计算机上存储的文件
共享文档 文件夹
Administrator 的文档 文件夹
硬盘
本地磁盘 (C:) 本地磁盘 31.8 GB 9.78 GB
本地磁盘 (D:) 本地磁盘 67.1 GB 11.3 GB
本地磁盘 (E:) 本地磁盘 24.9 GB 0 字节
本地磁盘 (F:) 本地磁盘 24.9 GB 20.9 GB
有可移动存储的设备
DVD 驱动器 (G:) CD 驱动器
DVD 驱动器 (H:) CD 驱动器
9 个对象
我的电脑



这个方法也有个小缺点，就是如果磁盘剩余空间很大，填满会比较耗时。

三、fsutil+FillDisk

把 fsutil 命令和 FillDisk 工具结合起来使用就会达到很理想的效果，先用 fsutil 命令生成一个大文件，再用 FillDisk 工具填满剩余的空间，这样就得到了一个可用空间为 0 字节的本地磁盘，还可以删除 FillDisk 生成的文件，制造可用空间为其它数据的情况。

零基础完成 Loadrunner 压力测试

◆作者：三千笔

摘要： 最近笔主带着两位新入职的同事进行了公司新平台的压力测试，工具选择的当然是 Loadrunner，小笔发现有很多刚入门 Loadrunner 的小白都会遇到很多相似的问题，但是这些问题并不能在各大搜索网站上得到完善的解决。因此，小笔选中了 51testing 这个流量给力认可度高的专业测试平台给各位 loadrunner 新手提拱一份参考，希望能够帮助到有需要的朋友。

在如今的大数据时代，软件、测试、自动化测试都在扮演者不可或缺的重要角色，我们开发一个平台要求的已经不仅仅是功能要正确，更要考虑的是随着访问量的增加给客户带来的压力体验。

OK，引文部分已经完成，下面我们一起走进 Loadrunner 的压力测试吧。

跟着小笔一起动手来完成此次的压力测试吧！一个完整的压力测试三部曲：

1. 脚本录制—>2. 场景设计—>3. 结果分析

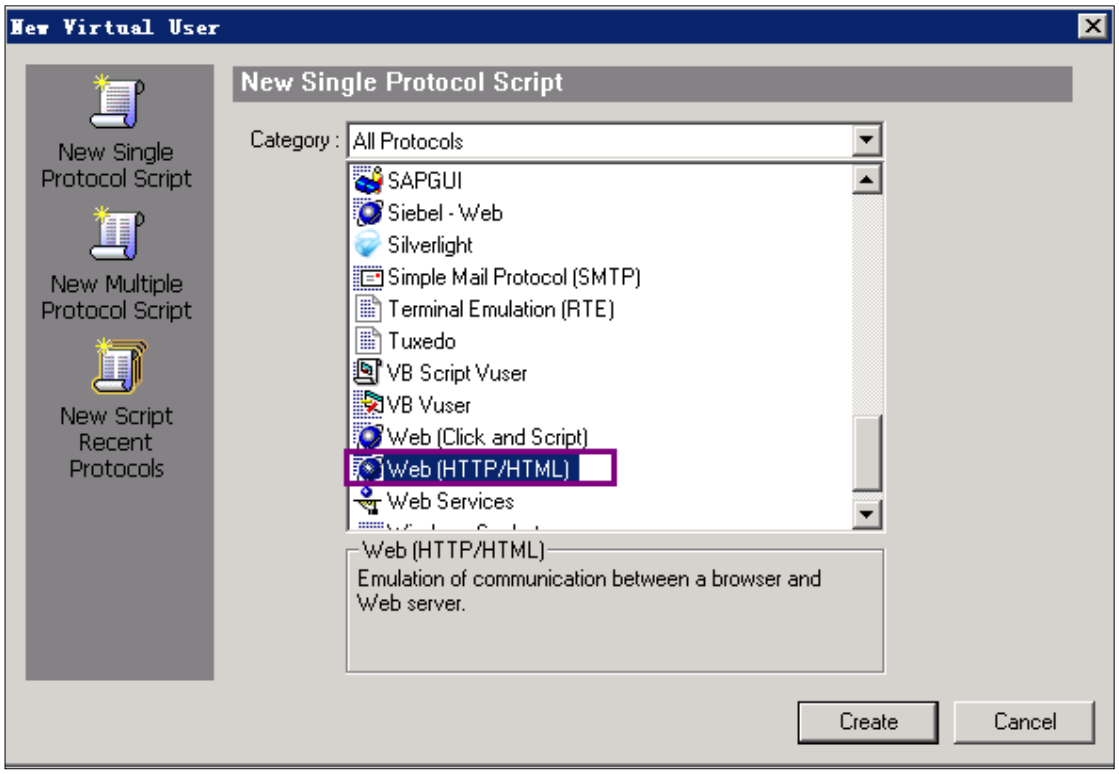
场景介绍：此处我们选择最具有代表意义的多用户并发登录系统，我们测试 150 个用户并发登录平台 A 的时候给系统增加的压力情况。

测试背景： Windows Server 2008+Loadrunner11+IE8

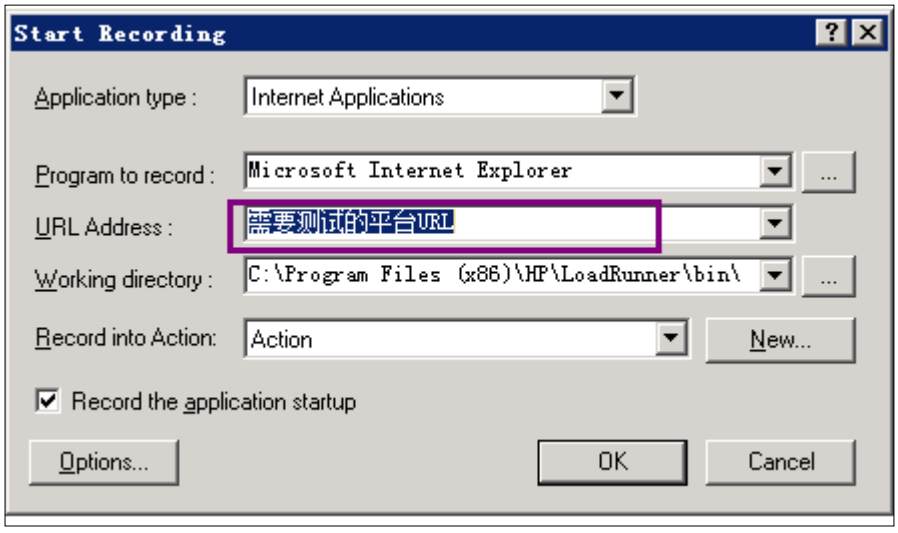
1. 录制脚本（Virtual User Generator）

安装好 Loadrunner 后（安装比较容易，在此暂且省略），打开 Virtual User Generator 进行脚本录制，录制时相关设置：

Step 1、Catalog 选择 ‘Web(HTTP/HTML)’，点击[Create] 按钮。



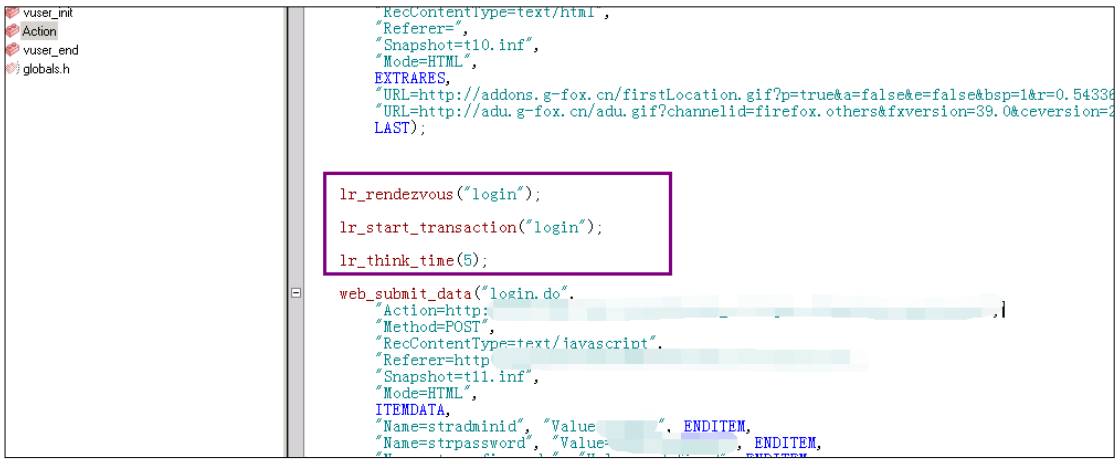
Step 2、[URL Address]的值输入需要测试系统的地址，点击[OK]按钮。



Step3、开始进行登录系统的脚本录制，一般情况下，我们在录制的过程中需要切分action，不同的操作放在相对应的 action 里，此处因为操作简单，我们暂且不去细分。

Step4、生成脚本

Step5、优化脚本：添加集合点，事务，思考时间。



事务：定义一个 action 的范围，以便对此 action 进行某种操作。比如对该 action 进行计时操作。

语句：lr_start_transaction("login");

集合点：正如字面意思，等待所有的事务集合到一起进行的操作，用来执行负载测试。要实现此操作，可以同步 Vuser 以便恰好在同一时刻执行任务。通过创建集合点，可以配置多个 Vuser 同时执行某个操作。当某个 Vuser 到达该集合点时，将进行等待，直到参与该集合的全部 Vuser 都到达。指定数量的 Vuser 均到达后，释放所有这些 Vuser。

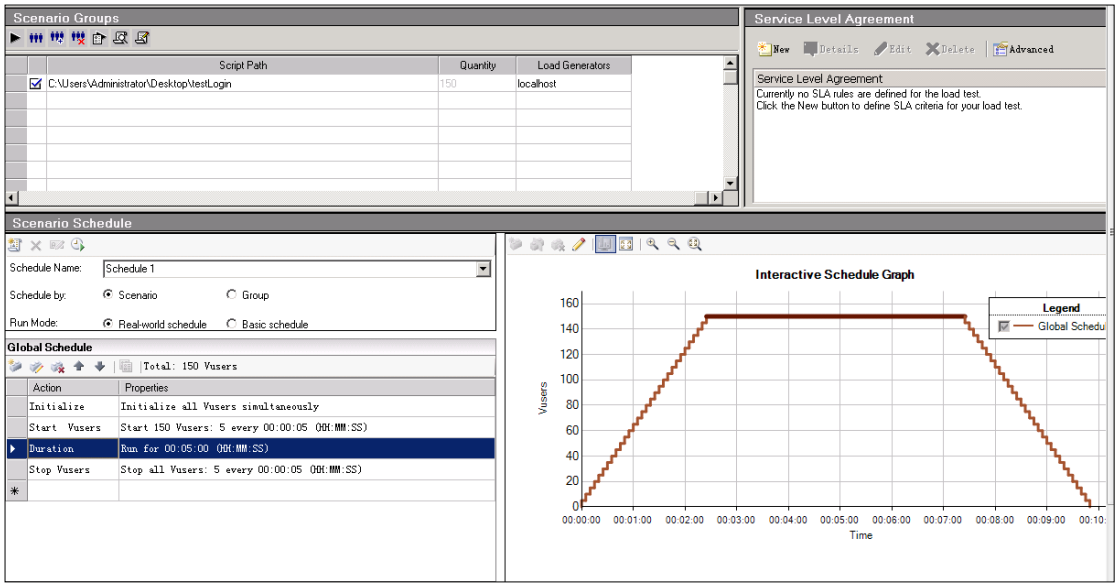
语句：lr_rendezvous("login");

思考时间：思考时间即等待时间，是一种延迟操作，很好理解。

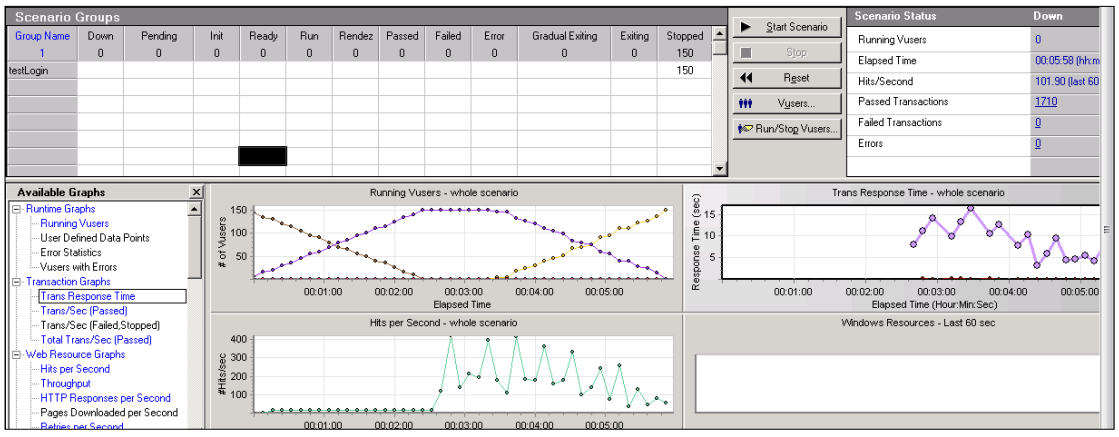
语句：lr_think_time(5);

2. 场景设计（Controller）

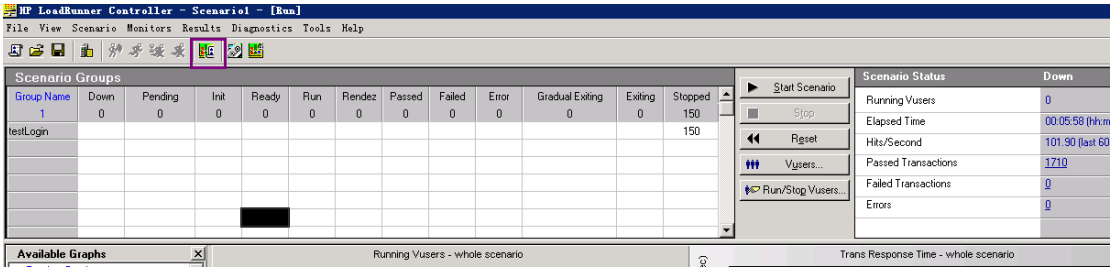
Step1、打开 controller，添加上面优化好的脚本，设置场景模式。（此处命名为 testLogin）设置场景如下：



Step2、点击【Start Scenario】运行脚本，结果如下：



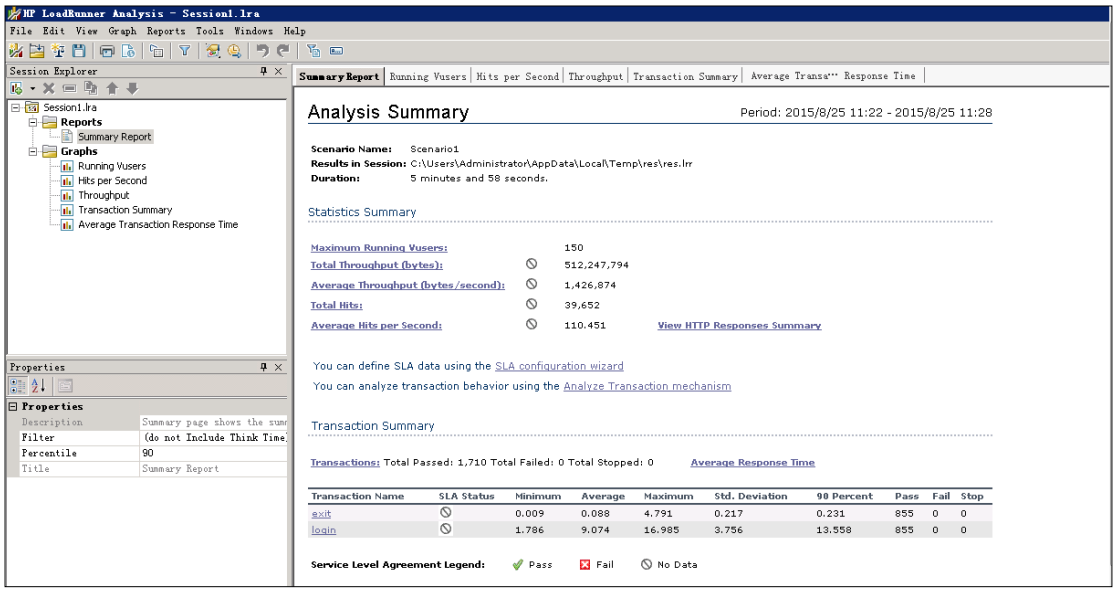
Step 3、点击紫色框中按钮，生成测试结果报告。



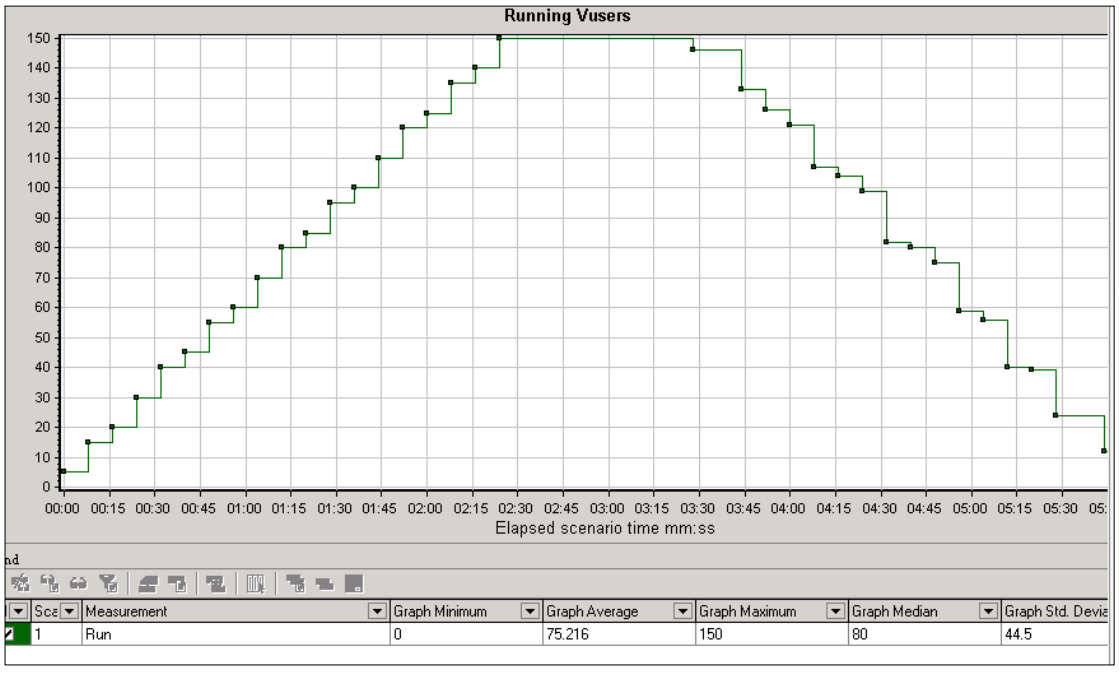
3. 结果分析（Analysis）

Analysis 可以说是 Loadrunner 压力测试的重点和难点，所以对于新手而言 analysis 不是测试的结束，而是开始。因此，对于各项测试结果我们要做出准确的理解和判断。在本次的实践中，我们做的是一个比较简单的场景，那么针对此场景的各项结果如下:

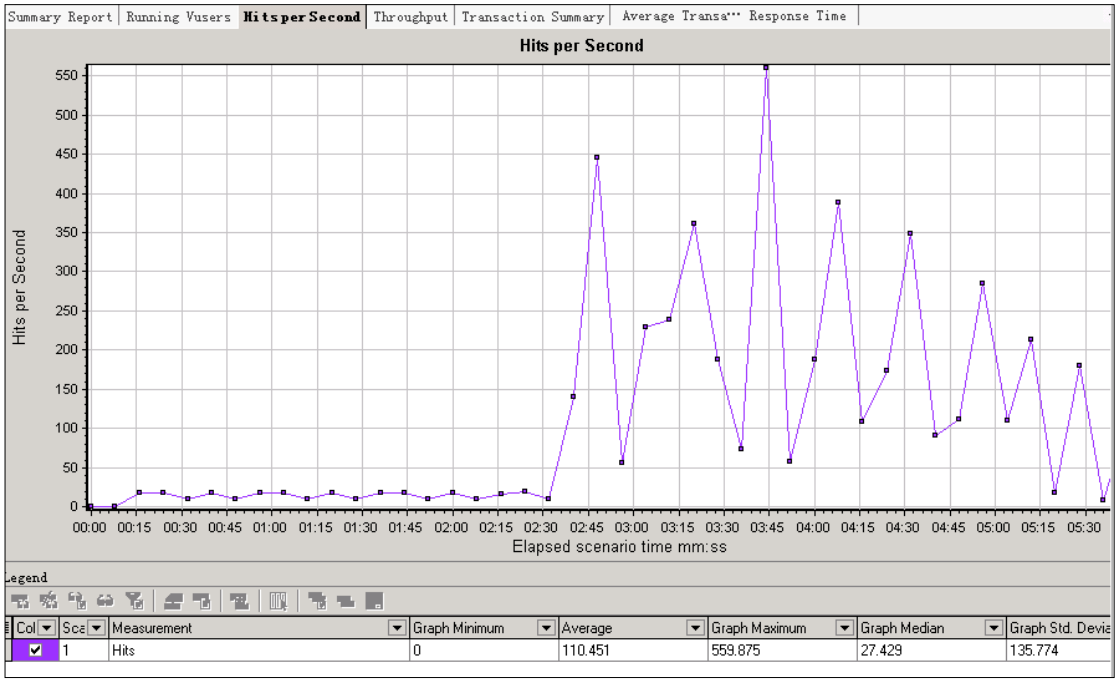
【测试报告分析摘要】，这里显示了实际测试过程中，总体的测试结果。我们可以选择更过的图来分析系统的负载情况。



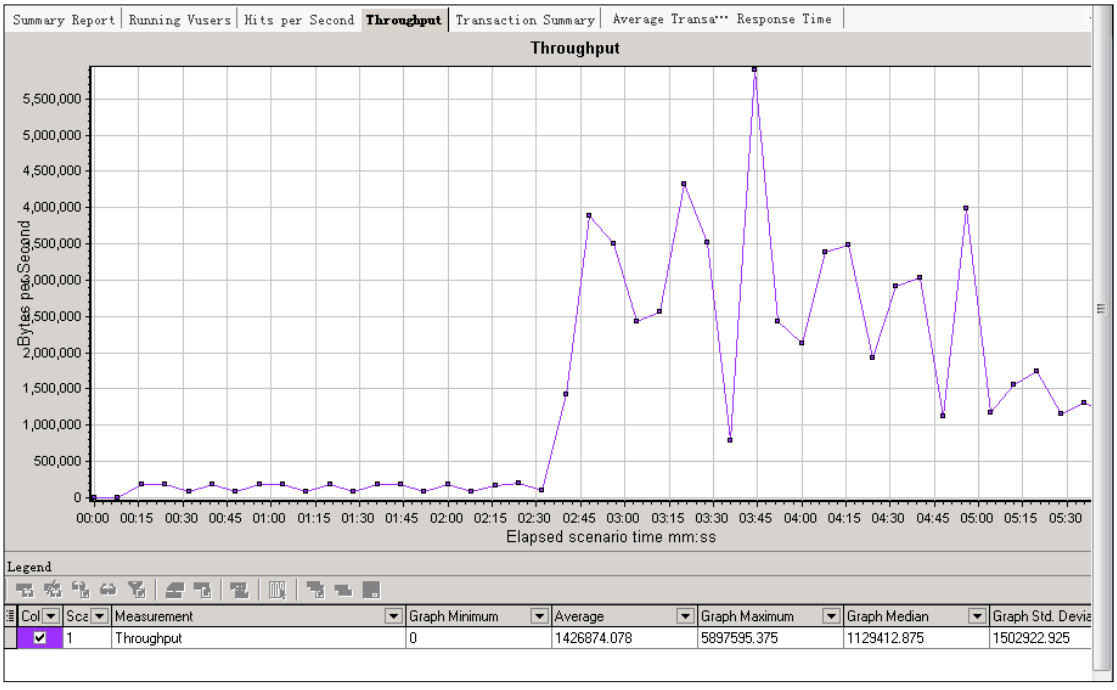
【Running Vuser】结果分析：Vuser 是并发测试选取的虚拟用户，从下图中可以看出，Vuser 是每 5 秒增加 5 个，在 02:20 秒的时候达到了顶峰值 150，持续运行了一分钟后，逐渐退出系统。



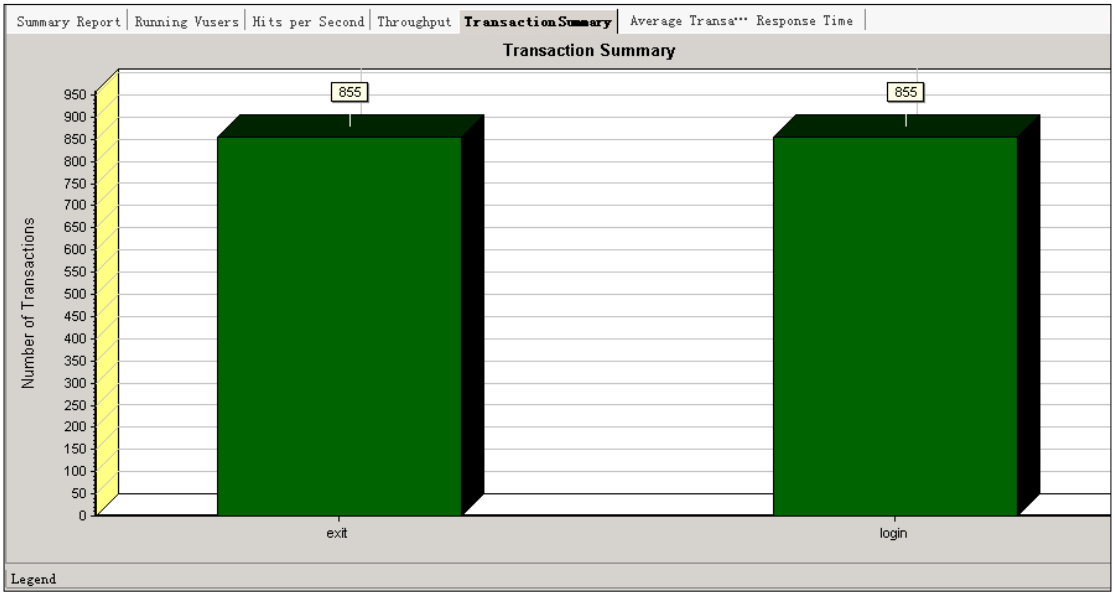
【Hits per Second】结果分析：每秒提交的 HTTP 请求数量，在本场景中执行的时间比较短，因此结果不是很明显，建议大家此处可以放宽执行时间，这样得到的结果比较准确。



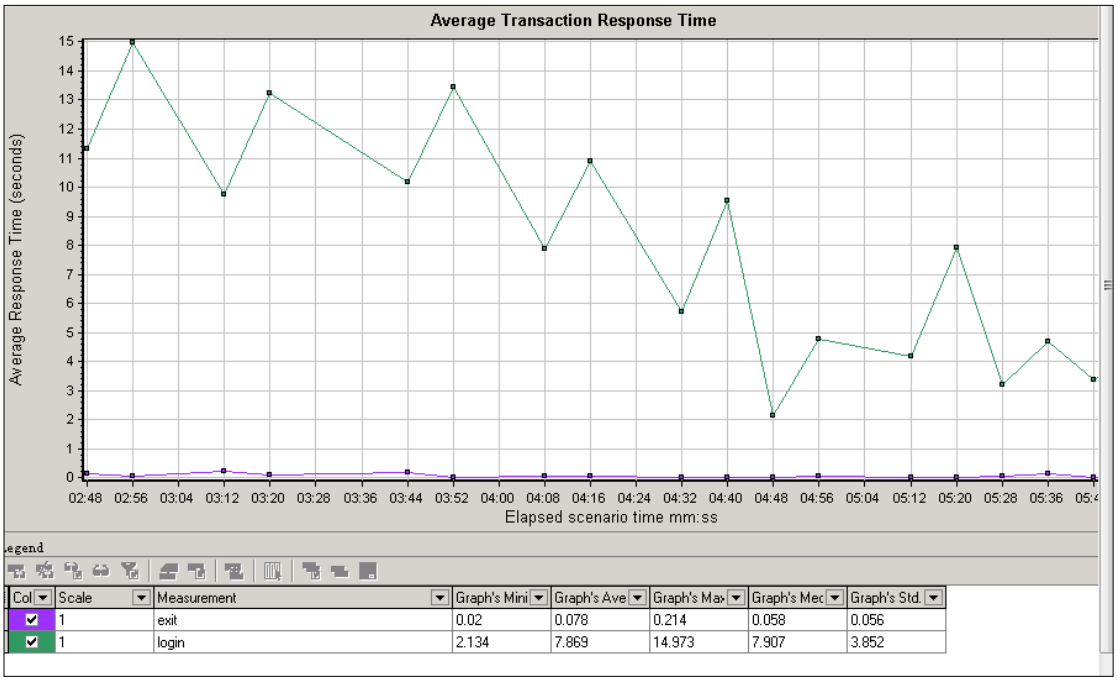
【Throughput】结果分析：吞吐量是指返回的应用层数据的值，吞吐量单位是以字节数为准，表示 Vuser 在任何给定的某一秒上从服务器获得的数据量。借助此图我们可以依据服务器吞吐量来评估 Vuser 产生的负载量。该数据越小说明系统的带宽依赖就越小，通过这个数据可以确定是不是网络出现了瓶颈。



【Tansaction summary】结果分析：事务概要说明，统计执行的事务数量，比如在本次场景中，login 和 exist 这两个事务的值都是 855 次。同事也监控了事务的 Pass 数和 Fail 数，了解负载的事务完成情况。通过的事务数越多，说明系统的处理能力越强；失败的事务数越小说明系统越可靠。这个比较容易理解，不多阐述。



【Average Transaction Response Time】- 事务响应时间结果分析：这里需要注意的一个问题是因为在 Transaction Response Times 里面是场景运行时记录的响应时间的最大值最小值与平均值，而 Average Transaction Response Time 是按照采样率每隔几秒钟取一个值画出来的图，然后根据图来记录最大值最小值和平均值，在报告中也可以看到，Average Transaction Response Time 中写的是图最大值、图小值和图平均值。如果将采样率设置小一些，这两个值就会比较接。所以，抽象率是关键。那么下图现实的结果可以看出，login 这个 action 最大值是 14.978，最小值是 2.134，平均值是 7.869；exist 最小值是 0.02，最大值 0.214，平均值是 0.078 。这些时间是可以接受的压力响应的时间。



本次测试过程中常见问题汇总：

之所以加上问题汇总是因为笔主觉得大家在做压力测试的时候，这类问题的出现率很高，所以，在此稍微总结一下。

问题 1：averager esponse time 响应时间过长？（与实际偏差甚大完全不合理）

解决方法：导致此问题的原因很多，但是我们可以从以下几类去分析：1、是否在脚本中添加了多长时间的思考时间。2、事务和集合点的先后顺序是否正确，正确的顺序是把集合点放在事务前面，反之则也会增加事务响应时间的值。3、网速问题，网速一般不会造成太大的偏大，但是不排除并发量很大的情况下造成的延误。

问题 2：LoadRunner 超时错误

解决方法：首先在运行环境中对超时进行设置，默认的超时时间可以设置长一些，再设置多次迭代运行，如果还有超时现象，需要在“Runtime Setting”>“Internet Protocol: Preferences”>“Advanced”区域中设置一个“winlnet replay instead of sockets”选项，再回放是否成功。

问题 3：LoadRunner 脚本中出现乱码

解决方法：重新录制脚本，在录制脚本前，打开录制选项配置对话框进行设置，在“Recording Options”的“Advanced”选项里先将“Surport Charset”选中，然后选中支持“UTF-8”的选项。

问题 4：在录制过程中 IE 页面上，某些控件显示有问题，导致录制不了。

解决方法：一般情况下，将被测系统的 URL 加入到可信任站点即可解决此类问题。

问题 5：Error -27796 ： Failed to connect to server ‘XXXX’

这个问题可以说是经常遇到但是不易被解决的难题，我们大致可以这样去排查

（1）检查 run time setting 中的请求超时时间 Preferences 中点击 Options ‘HTTP-request connect timeout’，‘HTTP-request receieve timeout’，‘Step download timeout’，查看其值是否为 1000、1000、10000；run time setting 设置完了后记住还需要在 control 组件的 option 的 run time setting 中设置相应的参数；

（2）Browser Emulation 中的 Download non-HTML resources 选项去掉，点击 OK 即可

如果还不能解决的话，继续尝试第 3 种方法

（3）设置 runt time setting 中的 internet protocol-preferences 中的 advaced 区域有一个 winlnet replay instead of sockets 选项，选项后再回放就成功了。

如果实在不行的话就试试重启大法吧，因为有些问题的确可能是因为工具问题，网络问题，机子问题等等。

总结：用 Loadrunner 进行压力测试难免会遇到各种问题，细心排查总能一一解决，所以笔者想对刚刚踏入这一行业的朋友说，不急不燥认真去思考，问题总能被解决。希望此篇文章对大家有所帮助，任何问题都可以留言喔。

参考文献:

1. 于涌.《精通软件性能测试与 Loadrunner 最佳实践》.人民邮电出版社.2013.6
2. http://www.51testing.com/html/index.html

作者简介

原名王海兰，笔名三千笔，就职于亚信科技（南京），担任软件测试一职位，业余喜欢读书写作.

平淡中的伟大

◆ 作者：那些失去的年华

说到这个题目，首先我得说说我们测试在开发组的地位。一个测试人员在一个开发组中的地位是很平凡的；他没有开发人员编写代码成功实现功能，所带给产品发展的瞩目成就，也没有一个构架师对于产品构架让人来的一目了然。我们测试所保证的是对一个项目或者产品质量的监控和品质的提升。所以我想表达的是对测试职业的一种发声。

我从事测试这一行有两年多的经验，算起大千的 IT 世界，我不过只是里面的一只小菜鸟。但是毕竟每个人的故事都不一样，不要认为你是一只菜鸟，所以你的故事就会平凡，至少我是不会这么认为的。

当然说到测试人生，我有自己的理解。算起工作环境应该有 3 到 4 个。还记得我进第一家公司的时候，当时老总就握着我的手对我说，我们公司派你出去到 xx 公司协助做性能测试，这对你以后的提升有很大的帮助；当时的我还不知道什么叫外派，什么叫无助。于是我就带着感激的心应下了。当我第二天去别的公司进行测试时，我才知道我是真的错了，就像一个无助的孩子找不到回家的路。但是我并没有慌，我需要的是思考接下来怎么安排自己的工作。对于性能测试，当时我才刚刚接触到皮毛，而外派公司对我的要求确是能够进行性能调优；我并没有气馁，我知道挑战才刚刚开始，于是我积极的去询问能帮助到我的人，寻找对于性能测试的方法等等；我感觉那段时间是我最充实也是最努力的一段时间。

当我感到对自己的性能测试技能还能够接受的时候，另一个挑战又来了，公司的老总打来电话说外派公司说我技术不行，需要换新的测试人员；当时我听到确实是很气愤，但我明白技术才能说明问题；于是我向公司老总阐明了意图，表明我是不会进行性能的调优，也确实需要公司的协助；其实我内心想法是实话说吧，如果不行炒掉我也没

话说。但公司同意派人来协调我，这也让我真正感受到了公司的爱。至此，大家都该明白，去执行一个项目本来就没有一帆风顺的，就像人的一生。而对于测试这一行，我们需要的其实是对这门技术的热爱。

而对于测试这条路我想说的是：首先你得有一颗热爱的心。因为我们测试这一行，对于技术的本质要求其实不高，而需要你掌握的知识面却要很广；比如说你要会写文档，会搭建测试环境，会管理测试团队，有自己的测试思维等等。测试行业需要的是一个全面的测试人员，这样才能更适应市场的发展

其次我认为在测试道路上，你要对测试技术和 IT 业界保持足够的好奇心，不断的去丰富自己，了解新的技术。再次也是最重要的一点就是要有目标；以前在公司里，老总都会要求我们写周报告，当时我就很反感，但是慢慢我就发现，你不去做你就写不出来，你写出来了你就会去做。在测试这条路上，执行力和目标性强的人都会有所成就的。

最后，我想说，我也是向成功迈进的青年，我们有的是活力和青春。重要的是把握自己人生的方向和目标，然后一如既往的走下去！

大规模项目将采用 CCPM

◆译者：why5256

一、CCPM

CCPM（Critical Chain Project Management：关键链条项目管理）是将项目的各个任务的预算进行最大限度的控制，预先设置相应程度的缓冲时间的管理手段。

比如在派对上要做十人份的火腿三明治来招待客人的话，需要 20 分钟来切肉，撒盐和胡椒，放小麦面粉，鸡蛋，面包粉，要用 10 分钟将卷心菜切碎拌上蛋黄酱，要用 5 分钟将黄油和芥末涂到面包上，要用 20 分钟用色拉油炸猪排，要用 5 分钟将卷心菜和炸面包切成三等分，这样预估的话一共要用 60 分钟。

往往这些工序中是包含某些程度的富余时间的，假设这些工序中包含 20%的富余时间，如果把这些工序中的富余时间给去掉再来计划时间的话，猪肉的准备时间就变成 16 分钟，卷心菜的准备时间变成 8 分钟，面包的准备时间变成 4 分钟，炸猪排时间变成 16 分钟，切完时间变成 4 分钟，一共需要 48 分钟。

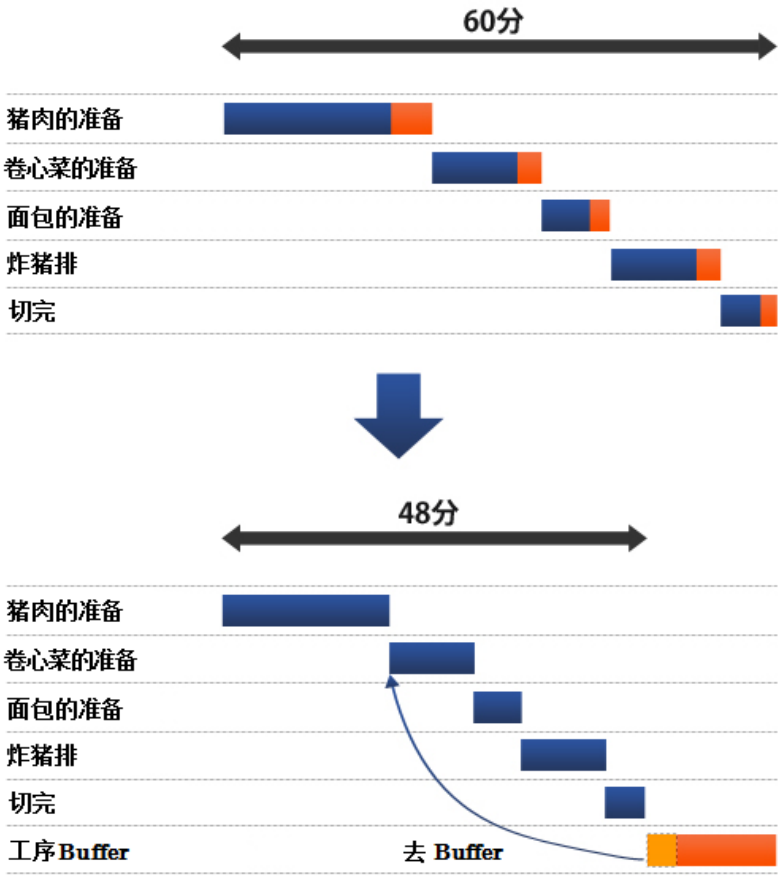


图 1　CCPM

将缩短的 12 分钟作为工序的预估缓冲时间，再对各道工序制定最大限度控制的计划来做三明治。最后结果，卷心菜切细多用了 2 分钟，炸猪排多用了 3 分钟，那么用预估的缓冲时间来实际来补充的话，预估的缓冲时间还剩 12 分钟-（2+3）分钟=7 分钟。比原来预计的 60 分钟还早 7 分钟，即用了 53 分钟就做好了火腿三明治。

为什么这么管理呢？因为人们习惯着眼眼前的目标，将缓冲的时间包含进计划，然后再按照计划进行的话就容易了（帕金森法则）。另外，暑假作业也是有人会在暑假结束前一直慢悠悠不急于做完，也有人提早做完了也不说的。

在将任务中包含的安全富余时间去掉基础上再把各任务衔接，取而代之的是设置在项目后的缓冲时间。这样的话，因为各个任务尽可能不使用缓冲时间来在努力去完成，即使拿掉这个缓冲时间，任务也能在整个项目的预计期间内完成。

二、CCPM 经营管理的效果显著

事实上我们（系统集成商）从 2011 年开始引入 CCPM，取得了连续 2 年刷新过去最大利润的成果。通常，公司的预算（销量，利润）是各个部门预算的叠加（反过来公

司各部门预算的分配也是一样的）。

① 各部门预算之和等于公司预算

这个公式就是说所有部门预算达成了整个公司的预算才能达成。反言之，哪个部门不好的话，公司预算也有不能达成的风险。

因此决定公司预算以后，把各部门的预算的累计值的 5%作为努力目标，这样即使哪个部门不好，也可以通过缓冲来去掉它来达成公司预算。

② 各部门的预算之和等于公司预算加上努力目标（各部门预算的 5%）

乍一看各部门的预算再加上努力目标很相似，不过不是每个部门的努力目标，而是全体的努力目标。因为只是自己部门的缓冲的话就能放心使用了，但对于全体的共用缓冲使用起来就会有抗拒感。

公司经营采用 CCPM 的结果，即使有些部门超出预算，但是最终仍然有缓冲残余，使得预算超出决算。效果超出预期，以后我们对大规模项目都使用 CCPM。

三、CCPM 具体使用方法

引入 CCPM 的管理手段的时候，考虑下如何制定计划进行管理吧。

- 最大限度的控制计划（ABP: Aggressive but Possible)

「最大限度的控制计划」是什么东西呢？如果说去除安全富余时间，但是不同人残存的富余时间和做过的工作类型都是不同的，一般来说概率是 50%，也就是说计划合适与不合适也是各占一半，得到这样的数字来说，最大限度的精度也不是很高的。

要想得到高精度的「最大限度的计划」的秘诀是好好的分解 WBS。假定被分解的每个任务顺利执行了来制定计划就是对整个计划的最大限度控制。

- 正式计划 (HP: Highly Possible)

不管怎样，如果预估最大限度控制的计划（内部目标）和正式计划（对外公开）两个的话，多少还有缓冲是不能决定的。按照制定计划顺序，应该是制定了最大控制计划以后，加上富余预估来制定正式计划。实际上是制定了正式计划后，再制定去除富余的最大限度控制计划，把差额作为缓冲才是现实的。

- 得出缓冲的方法

执行下面的步骤来得出缓冲吧。

① WBS 展开

② 预估每个任务的费用，整体做正式计划

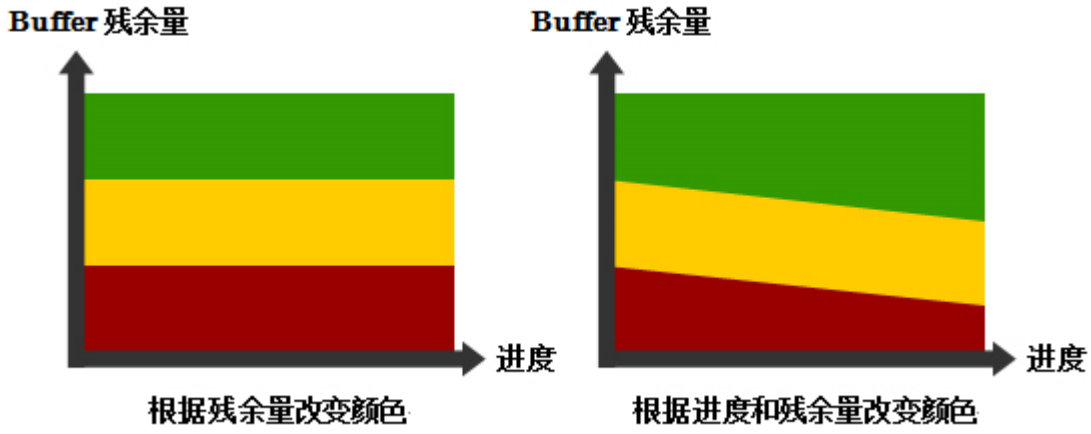
③ 去除各个任务的富余，整体做最大限度控制计划

④ 正式计划减去最大限度控制计划的差额作为缓冲预备

这时难的是步骤③，何种程度的富余时间是依存人的，也有人是没有任何富余时间的。因此，我们在做正式计划后一律使用 20%作为缓冲时间来对抗超预算的情况。

- 管理上的注意点

在用 CCPM 进行项目管理时，要对各个任务的进度进行管理的同时进行缓冲的管理。根据缓冲数值的残余量对应颜色绿⇒黄⇒红的改变也是常有的做法。这是单纯根据残余量来改变颜色，和根据进度和残余量的关系来改变颜色的方法。



例如，像表 1 那样有 4 个任务的项目，各个任务的计划值一共是 3000，预估缓冲按照 600 算（就是执行预算是 3600）。任务 A 使用 100 缓冲量，任务 B 比计划还少使用预算量 50，任务 C 已使用 250 缓冲量了，到任务 C 完成的时间点缓冲残余量还剩 300。如果再稍微努力下，不止能对任务单位缓冲量进行把握，而且对任务单位，月单位的缓冲量都可以进行管理。

任务	计划	实绩	Buffer使用
任务 A	400	500	100
任务 B	1000	950	-50
任务 C	600	850	250
任务 D	1000		
合计	3000	2300	300

表 1 任务单位 Buffer 使用量管理

四、CCPM 课题

CCPM 的课题是如何进行最大限度控制计划成本估算。因为即使按照把各任务的安全富余去掉来统计，或者砍掉正式报价的 20%，还是有时间不富余的情况下执行任务的感觉。所以更想以看不见的巧妙形式来拥有安全富余时间。

重要的是，使用缓冲了即使颜色变红时也不是问题的事情要跟全员共享。因为缓冲变成负数时才是超出预算，使用缓冲了也不要有出言责备的气氛要特别注意下吧。

Robot FrameWork 持续集成测试实战

◆ 作者：姜林斌

一、64 位 PC 上部署 RobotFrameWork+Jenkins

1.1　安装 Python 2.7(64 位)版本

1.2　安装 pip

安装命令: python get-pip.py

1.3　安装 AutoItX-V3(AutoItLibrary 库依赖于它)

1.4　安装 robot framework

安装命令: pip install robotframework

1.5　安装 wxPython(支撑 Ride 的运行库)

1.6　安装 robotframework-ride(GUI 界面管理)

安装命令: pip install robotframework-ride

1.7　安装 selenium2library

安装命令: pip install robotframework-selenium2library

1.8　安装 pyodbc

安装命令:pip install pyodbc

1.9　安装 requests(支持 RequestLibrary 库)

安装命令:pip install requests

1.10　安装 RequestsLibrary(http request 库)

安装命令: pip install -U robotframework-requests

1.11 安装 AutoItLibrary

安装命令:pip install AutoItLibrary

1.12 安装 Openpyxl

安装命令:pip install openpyxl

1.13 安装 Psutil

安装命令:pip install psutil

1.14 安装 JDK 并配置环境变量

1.15 下载 jenkins.war 包

二、常用的三方库及其 API

2.1 Selenium2Library(浏览器&页面元素操作)

2.1.1 打开浏览器 Open Browser

示例: Open Browser http://192.168.10.206/somoStorage Chrome

2.1.2 跳转到指定 RUL 地址 Go To

示例: Go To http://192.168.10.206/somoStorage

2.1.3 点击元素 Click Element

示例: Click Element id= id=Main

Click Element xpath=//div[@id="divLoading"]/img

2.14 点击超链接 Click Link

示例: Click Link添加

Click Link //a[contains(.,'添加')]

2.1.5 上传文件 Choose File

示例: Choose File xpath=//div[@id="divFiles"]/div[1]/input E:\\TestData\\14.jpg

2.1.6 清空输入框 Clear Element Text

示例：Clear Element Textxpath=//li[@id="Monday"]/div[3]/div[2]/span[2]/input

2.1.7 向输入框中填写内容 Input Text

示例：Input Text xpath=//li[@id="Monday"]/div[3]/div[2]/span[2]/input ${极限总号额}

2.1.8 对话框作确定操作 Confirm Action

示例：Confirm Action

2.1.9 执行 JS 方法 Execute JavaScript

示例：Execute JavaScript javascript:show_cate();

2.1.10 等待元素可见 Wait Until Element Is Visible

示例：

Wait Until Element Is Visible xpath=/html/body/div[1]/table/tbody/tr/td/div/div[2]/iframe 10s error=等待元素可见超时

2.1.11 等待元素可用 Wait Until Element Is Enabled

示例：Wait Until Element Is Enabled xpath=/html/body/div[4]/div/div/div[1]/div[1]/div[2] 20s error=等待门诊预约元素超时

2.1.12 锚定某个元素 Focus

示例：Focus id=btnGeneralSave

2.1.13 切换焦点到 frame 上 Select Frame

示例：Select Frame xpath=/html/body/div[1]/table/tbody/tr/td/div/div[2]/iframe

2.1.14 切换焦点到窗口 Select Window

```
@{handles}=    list windows
select window    ${handles[1]}
```

2.1.15For 循环使用

```
@{location_list}=    Create List    A0   A-1
```

```
: FOR ${locator}      IN   @{location_list}
```

更改最危灶位置 ${locator} ${locator}

Take Screenshot ${locator}.jpg

还有很多很多 在此不一一列举 请志同道合的同学自行研究哈 ^_^!

2.2 Pyodbc 库 操作数据库(Sql Server)

Pyodbc 操作 Sql Server 示例:

```
# 在 Sql Server 数据库中执行查询
def get_select_result_in_sql_server(db_name, sql_select):
conn=pyodbc.connect(DRIVER='{SQL Server}',SERVER='FARIDAH-
PC',DATABASE=db_name,UID='sa',PWD='sa')
cursor = conn.execute(sql_select)
row = cursor.fetchone()
conn.commit()
select_count = row[0]
conn.close()
return select_count
```

这里仅举一个简单的栗子 请查看 pyodbc 库的源码学习更多的 API。

若使用 Oracle 数据库请考虑 cx_Oracle+DatabaseLibrary 结合。

若使用 My Sql 数据库请自行配置 pyodbc+ mysql-connector-odbc 驱动。

2.3 Requests 库(模拟 http 协议的请求发送)

发送 Get 请求:获取(Github 的公共时间线)

```
requests.get('https://github.com/timeline.json')
```

发送一个 HTTP POST 请求

```
requests.post("http://httpbin.org/post")
```

其他方法：

```
requests.put("http://httpbin.org/put")

requests.delete("http://httpbin.org/delete")

requests.head("http://httpbin.org/get")

requests.options("http://httpbin.org/get")
```

响应状态码：

```
request.options("http://httpbin.org/get").status_code
```

响应内容：

```
request.options("http://httpbin.org/get").content
```

为 URL 传递参数：

```
payload = {'key1': 'value1', 'key2': 'value2'}

requests.get("http://httpbin.org/get", params=payload)
```

错误与异常

遇到网络问题（如：DNS 查询失败、拒绝连接等）时，Requests 会抛出一个 ConnectionError 异常。

遇到罕见的无效 HTTP 响应时，Requests 则会抛出一个 HTTPError 异常。

若请求超时，则抛出一个 Timeout 异常。

若请求超过了设定的最大重定向次数，则会抛出一个 TooManyRedirects 异常。

所有 Requests 显式抛出的异常都继承自 requests.exceptions.RequestException 。

2.4 Openpyxl 库(操作 Excel 文件)

示例：

```
# 对指定单元格赋值

def WriteExcel(result, locator,sheetname):

    # 加载 excel 文件
```

```
book = load_workbook(autocase)
# 定位工作 sheet 单
    sheet = book.get_sheet_by_name(sheetname)
if result == True:
    # 对单元格赋值
        sheet.cell(locator).value = 'Pass'
    elif result == False:
        sheet.cell(locator).value = 'Fail'
    # 保存 excel 文件
    book.save(autocase)
#显示表名，表行数，表列数
sheet.title
sheet.get_highest_row()
sheet.get_highest_column()
# 建立存储数据的字典
data_dic = {}
#把数据存到字典中
for rx in range(ws.get_highest_row()):
    temp_list = []
    pid = sheet.cell(row = rx,column = 0).value
    w1 = sheet.cell(row = rx,column = 1).value
    w2 = sheet.cell(row = rx,column = 2).value
    w3 = sheet.cell(row = rx,column = 3).value
    w4 = sheet.cell(row = rx,column = 4).value
```

```
    temp_list = [w1,w2,w3,w4]

   # 获取指定单元格的值

   Sheet. cell("A1").value
```

2.5 AutoItLibrary 库(模拟鼠标键盘操作)

在指定坐标位置单击:

Mouse Click LEFT 696 383

在指定坐标右击:

Mouse Click RIGHT 300 300

拖拽元素:

Mouse Click Drag LEFT 300 300 600 600 Speed=1

滚动鼠标中间键:

Mouse WheelDOWN 10

Mouse WheelUP 7

2.6 Psutil 库(获取系统资源数据)

CPU:

```
# 获得 CPU 时间使用信息

psutil.cpu_times()

scputimes(user=191488.046875, system=81560.75, idle=3844134.0)

# 获得 CPU 物理个数和逻辑 CPU 数量

psutil.cpu_count()

psutil.cpu_count(logical=False)
```

Memory:

```
# 获得内存缓冲区(缓存)大小

psutil.swap_memory()
```

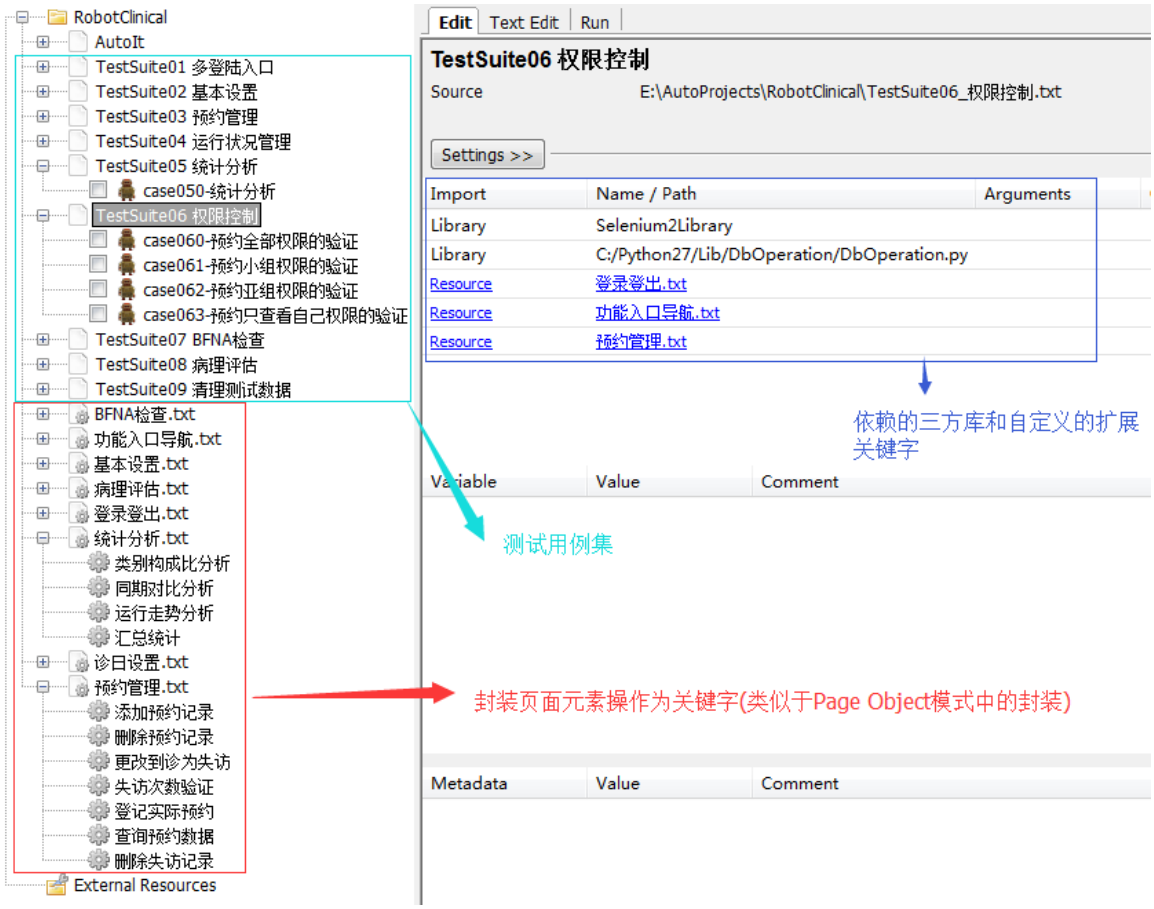
```
    sswap(total=17007190016L, used=5249916928L, free=11757273088L, percent=30.9, sin=0, sout=0)

    # 获得虚拟内存大小
psutil.virtual_memory()

    svmem(total=8504537088L, available=4915023872L, percent=42.2, used=3589513216L,
free=4915023872L)

    Disk:

    # 获得磁盘分区信息

    psutil.disk_partitions()
[sdiskpart(device='C:\\', mountpoint='C:\\', fstype='NTFS', opts='rw,fixed'), sdiskpart(device='D:\\',
mountpoint='D:\\', fstype='NTFS', opts='rw,fixed'), sdiskpart(device='E:\\', mountpoint='E:\\', fstype='NTFS',
opts='rw,fixed'), sdiskpart(device='G:\\', mountpoint='G:\\', fstype='', opts='cdrom')]

    # 获取指定磁盘的使用信息
psutil.disk_usage('C:\\')
sdiskusage(total=160956411904L, used=70494846976L, free=90461564928L, percent=43.8)

    # 获取磁盘 IO 读写(吞吐量)信息
psutil.disk_io_counters()

    sdiskio(read_count=4680532, write_count=7875453, read_bytes=86995578880L,
write_bytes=146009722880L, read_time=2546110150L, write_time=4543259250L)

    NetWork

    # 获得 net 连接信息

    psutil.net_connections()

    # 获得网络吞吐信息
psutil.net_io_counters(pernic=True)
```

三：使用 Robot+Jenkins 进行持续自动化测试(UI+接口)

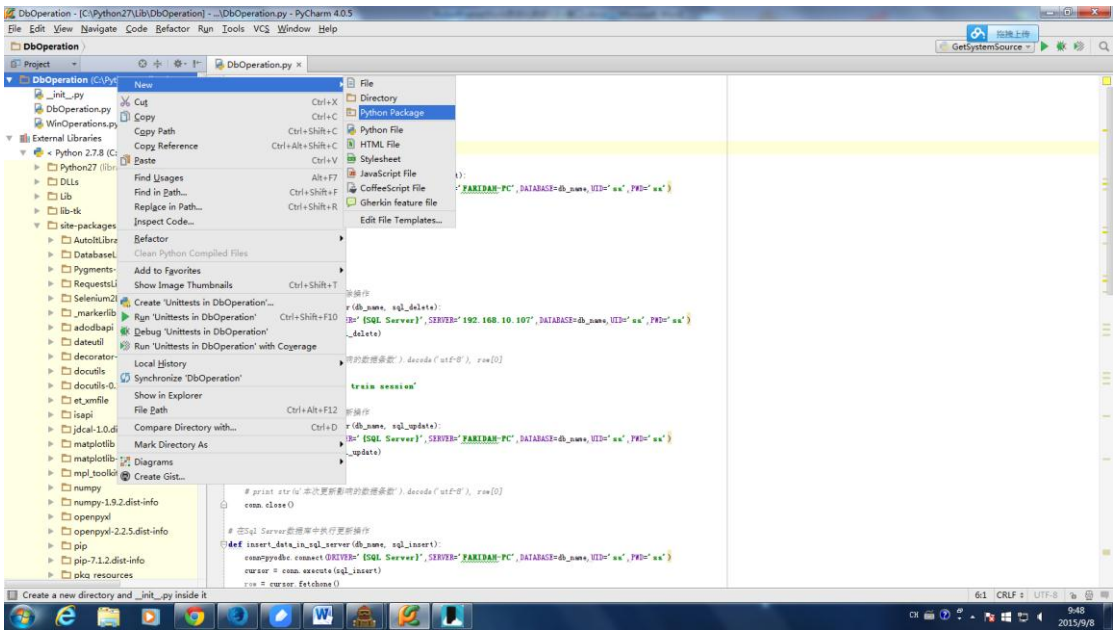
3.1 Web UI 自动化测试

以邵逸夫医院预约与病理管理系统为例：

3.1.1: RobotFrameWork 整体架构目录

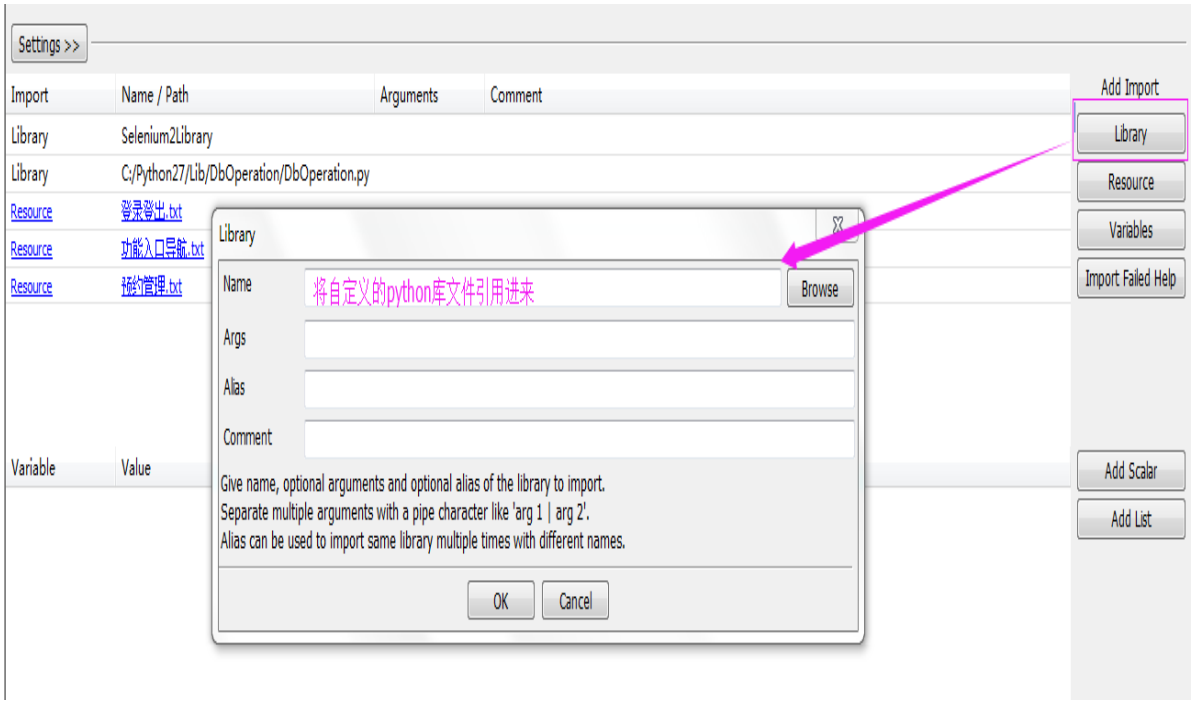


3.1.2 自定义 python 库



步骤 1: 新建 Python Package

步骤 2：在 Robot 中添加 py 文件



3.2 WebService 接口功能自动化测试

封装基于 requests 库的自定义三方库 HttpRequests 其中提供的 API 如下：

3.2.1 返回 get 请求的 text

```
def text_of_get(url, params):
    json_data = {}
    parameters = params.split(',')
    counts = len(parameters)-1
    # 将字符串转换为 json 格式
    for index in range(0, counts):
        name = parameters[index]
        value = parameters[index+1]
        json_data[name] = value
    get_request = requests.get(url, params=json_data)
    # 返回请求的 text
```

```
    return get_request.text
```

3.2.2 返回 post 请求的 text

```
def text_of_post(url, params):
    json_data = {}
    parameters = params.split(',')
    counts = len(parameters)-1
    # 将字符串转换为 json 格式
    for index in range(0, counts):
        name = parameters[index]
        value = parameters[index+1]
        json_data[name] = value
    post_request = requests.post(url, params=json_data)
    # 返回请求的 text
    return post_request.text
```

3.2.3 返回 get 请求的 status_code

```
def status_code_of_get(url, params):
    json_data = {}
    parameters = params.split(',')
    counts = len(parameters)-1
    # 将字符串转换为 json 格式
    for index in range(0, counts):
        name = parameters[index]
        value = parameters[index+1]
        json_data[name] = value
```

```
    get_request = requests.get(url, params=json_data)
    # 返回请求的 text
    return get_request.status_code
```

3.2.4 返回 post 请求的 status_code

```
def status_code_of_post(url, params):
    json_data = {}
    parameters = params.split(',')
    counts = len(parameters)-1
    # 将字符串转换为 json 格式
    for index in range(0, counts):
        name = parameters[index]
        value = parameters[index+1]
        json_data[name] = value
    post_request = requests.post(url, params=json_data)
    # 返回请求的 text
    return post_request. status_code
```

3.2.5 在 Robot FrameWork 中集成 HttpRequests

Robot FrameWork 中的测试用例步骤

1	${airlines}=	text_of_get	http://www.webxml.com.cn/webservices/Don	startCity,杭州,lastCity,北京,theDate,2015-10-18,userID,
2	${http_code}=	status_code_of_get	http://www.webxml.com.cn/webservices/Don	startCity,杭州,lastCity,北京,theDate,2015-10-18,userID,
3	Comment	# 将返回的text记录到日志		
4	Log	${airlines}		
5	Log	${http_code}		
6	Comment	# 判断返回的状态码是否为200		
7	${code}=	Convert To String	${http_code}	
8	Should Be Equal	${code}	200	
9	Comment	# 判断返回的text中是否包含HU7278航班		
10	Should Contain	${airlines}	HU7278	
11				
12				

日志文件信息：

3.3 创建 Jenkins Jobs 进行持续自动化测试

3.3.1 安装并配置 JDK

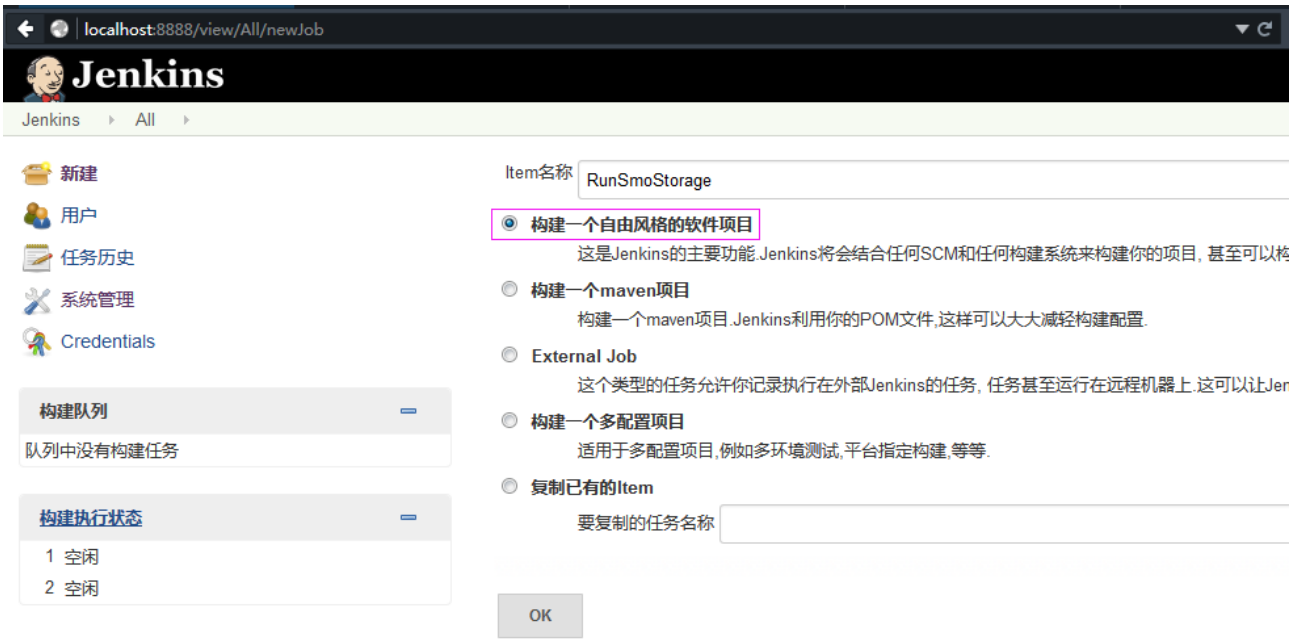
3.3.2 使用批处理文件启动 Jenkins 服务

脚本内容如下：

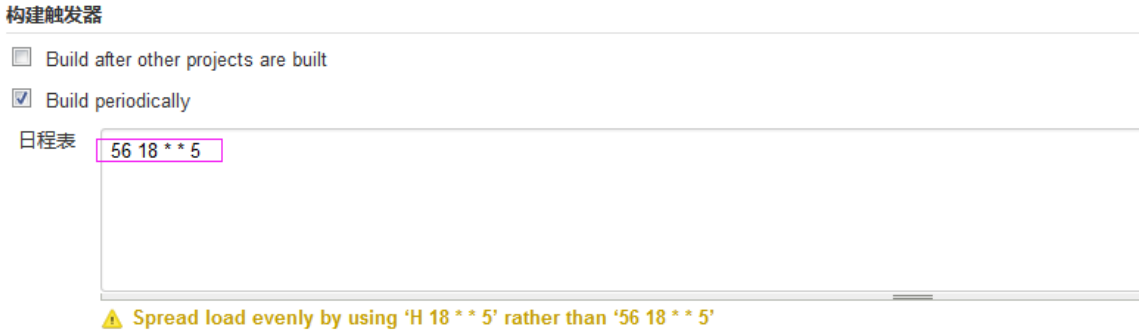
```
set JENKINS_HOME=D:\JenkinsHome

cd /d %JENKINS_HOME%

java -jar %JENKINS_HOME%\jenkins.war --httpPort=8888
```

3.3.3 创建 Job 任务

构建自由风格的 Job

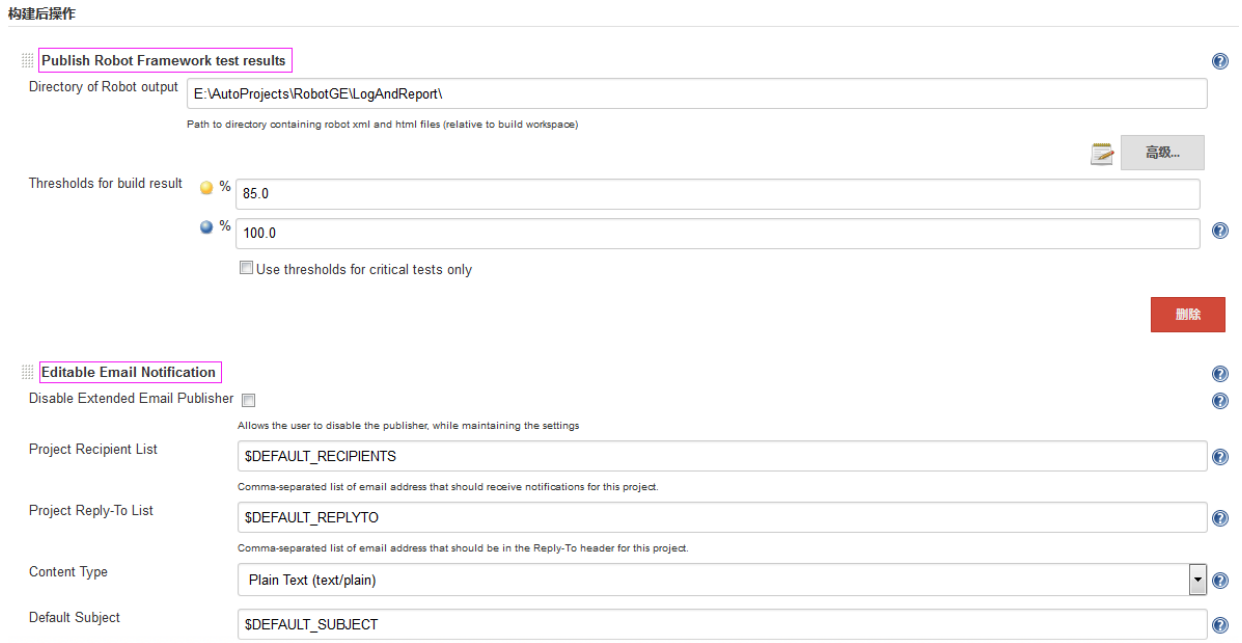


配置任务触发条件(示例为每周五下午 6 点 56 分执行):



配置任务，即：执行指定的自动化脚本

配置构建完成后的任务(示例为发布 Robot 的测试报告并发送邮件到指定的邮箱):

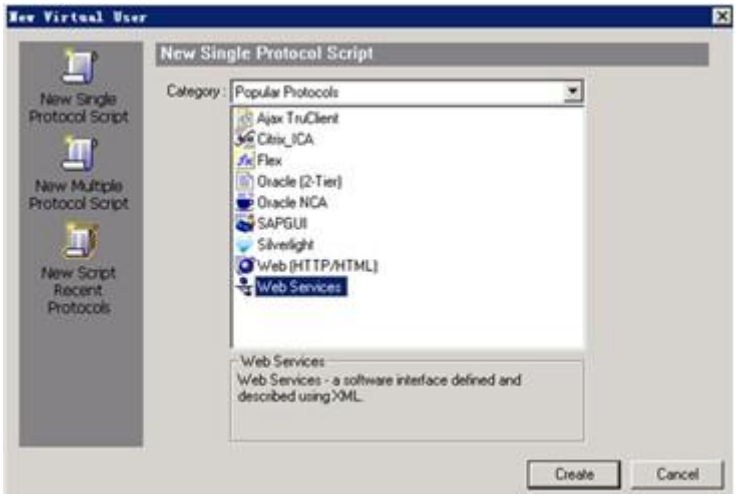


四：接口性能测试

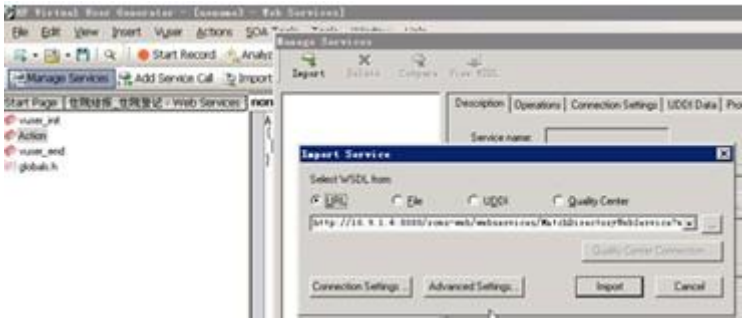
4.1 使用 LoadRunner 进行接口性能测试

4.1.1. 如何使用 loadrunner 测试 webservice 接口？

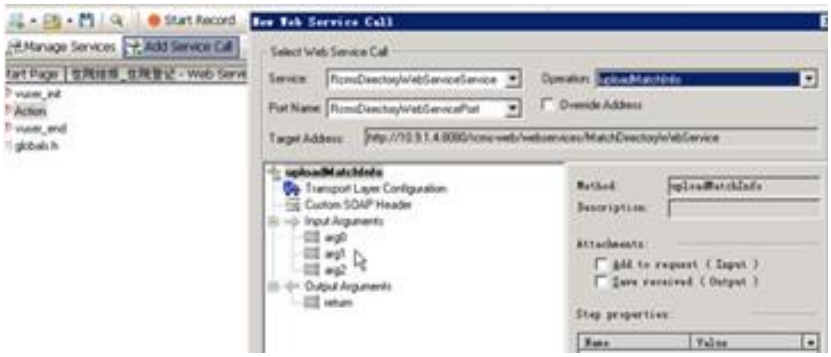
1）新建测试项目时，选择 Web Services



2）点击 Manage Services。导入测试接口的 url。



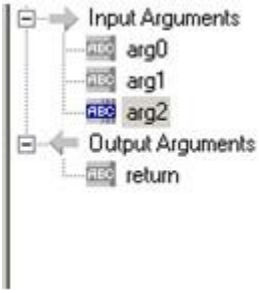
3）点击 Add Service Call，选择需要测试的具体方法。

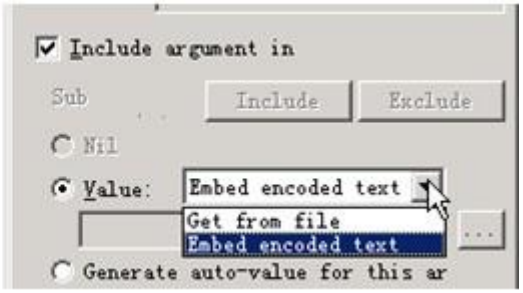


4）指定输入参数和输出参数的值，输入参数一般有两种，分别是字符串和 xml 文件。接口的返回值放到输出参数里面，可以通过设置检查点的方式来验证返回值是否正确。

5）设置好输入参数和输出参数后，点击 OK 后，就会生成测试脚本。

4.1.2. 如果输入参数为 xml 文件，如何处理？





如果输入参数为 xml 文件的话，有两种处理方式。方式一：选择现有的 xml 文件。方式二：将 xml 当成字符串，进行 base64 编码，当成字符串传入。

采用第一种方式的话，没办法进行参数化，大多数测试场景下都需要对 xml 里面具体的元素值进行参数化。所以第一种方式大部分情况下不适用。

采用第二种方式的话，可以对 xml 的元素值进行参数化。更加灵活。

4.1.3. 输入参数为 xml 文件，如何对元素值进行参数化？

思路是这样的：首先获得需要传入的 xml，将 xml 进行分解，然后将 xml 里面需要参数化的值进行参数化。定义几个变量，将参数的值赋给变量。然后使用 strcat 方法将变量和分解的 xml 进行字符串连接。然后将连接好的 xml 字符串进行 base64 编码，将编码后的字符串当做输入参数。问题解决。

示例：

步骤 1：获得需要传入的 xml，将 xml 进行分解。

```
<?xml version="1.0" encoding="UTF-8"?><opRegister><hisOutpatientSn>
hisOutpatientSn
</hisOutpatientSn><compType>11</compType><personalNum>
personalNum
```

```
</personalNum><name>test</name><age>78</age><certNum>
certnum
</certNum><cardNum>3709820000010001</cardNum><visitOrgCode>370982206</visitOrgCod
e><visitOrgName>test</visitOrgName></opRegister>
```

步骤 2：将 xml 里面需要参数化的值进行参数化。hisOutpatientSn 元素值参数化为 t 开头的随机码。Personalcode 元素值参数化为 personalcode。Certnum 的元素值参数化为 certnum。

```
<?xml version="1.0" encoding="UTF-8"?><opRegister><hisOutpatientSn>t
{NewParam}
{NewParam_1}
</hisOutpatientSn><compType>11</compType><personalNum>
{personalcode}
</personalNum><name>test</name><age>78</age><certNum>
{certnum}
</certNum><cardNum>3709820000010001</cardNum><visitOrgCode>370982206</visitOrgCod
e><visitOrgName>test</visitOrgName></opRegister>
```

步骤 3：定义几个变量，将参数的值赋给变量。

```
    char plain[500],temp[500],temp2[500],*str1,*str2,*str3,*str4;
    str1=lr_eval_string("{NewParam}");
    str2=lr_eval_string("{NewParam_1}");
    str3=lr_eval_string("{personalNum}");
    str4=lr_eval_string("{certNum}");
```

步骤 4：使用 strcat 方法将变量和分解的 xml 进行字符串连接。

```
strcpy(temp,"<?xml version=\"1.0\" encoding=\"UTF-8\"?><ipRegister><hisInpatientSn>t");
strcat(temp,str1);
strcat(temp,str2);
strcat(temp,"</hisInpatientSn><compType>23</compType><personalNum>");
strcat(temp,str3);
strcat(temp,"</personalNum><name>test</name><age>33</age><certNum>");
strcat(temp,str4);
strcat(temp,"</certNum><cardNum>3709820000010001</cardNum><visitOrgCode>370982
206</visitOrgCode><visitOrgName>test</visitOrgName></opRegister>"
```

步骤 5：将连接好的 xml 字符串进行 base64 编码。

```
b64_encode_string(temp,"plain");
```

注：需要自己引入 base64 编码和解码函数的头文件。

步骤 6：将编码后的字符串填到需要填写入参值的地方。

```
BEGIN_ARGUMENTS,
"arg0=test",
"arg1=1",
"xml:arg2="
    "<arg2 base64Mode=\"encoded\">{plain}</arg2>",
END_ARGUMENTS,
```

问题搞定。

注：定义变量的时候，使用 char *定义的话，运行脚本到 strcpy 方法时可能会报错：

```
Error: C interpreter run time error: Action.c (11): Error -- memory violation : Exception
ACCESS_VIOLATION received.
```

原因是使用 strcpy 的时候，src 和 dest 所指向的内存区域不能重叠，并且 dest 必须要有足够的空间来容纳 src。只是单纯的定义个 char *a;的话，这时的 a 是一个非存在或非确定的地址，必须要为 a 分配确定的内存地址。

4.1.4. Webservice 返回值为 xml 时，如何设置检查点？

Webservice 测试，返回值有时是经过 base64 编码之后的 xml 格式的字符串。设置检查点需要先将返回的字符串进行解码操作，然后设置检查点检查 xml 中的元素值。

示例：

步骤 1：设置接口的返回值存到参数 Param_return 中。

```
BEGIN_RESULT,
"return=Param_return",
END_RESULT,
```

步骤 2：将 Param_retur 的值进行 base64 解码，解码之后的值放到参数 plain 中。

```
b64_decode_string( lr_eval_string("{Param_return}"), "plain" );
```

步骤 3：查看返回值的 xml 格式。

```
lr_output_message("plain: %s", lr_eval_string("{plain}") );
```

执行脚本，会在控制台输出类似以下内容。

```
Action.c(27): plain: <result>
  <datas>
    <opPreComp>
      <compType>11</compType>
      <totalAmount>30</totalAmount>
      <calcCanPay>0</calcCanPay>
      <compAmount>0.0</compAmount>
      <accountPayAmount>0</accountPayAmount>
      <fundPayAmount>0.0</fundPayAmount>
      <ownAmount>30.0</ownAmount>
    </opPreComp>
  </datas>
  <serviceDescription>opPreCalculate</serviceDescription>
  <msgs>
    <msg>鍣?00 鍣摹幟鍌ｈ皘鐢儿垚鍔?/msg>
  </msgs>
</result>
```

如果有乱码的话，可以将输出结果先转换成 utf-8 再输出。

```
lr_convert_string_encoding(lr_eval_string("{plain}"),"utf-8",NULL,"my");
lr_output_message("plain: %s", lr_eval_string("{plain}") );
```

结果会变成：

```
Action.c(29): plain: <result>
  <datas>
    <opPreComp>
      <compType>11</compType>
      <totalAmount>30</totalAmount>
      <calcCanPay>0</calcCanPay>
      <compAmount>0.0</compAmount>
      <accountPayAmount>0</accountPayAmount>
      <fundPayAmount>0.0</fundPayAmount>
      <ownAmount>30.0</ownAmount>
    </opPreComp>
  </datas>
  <serviceDescription>opPreCalculate</serviceDescription>
  <msgs>
    <msg>【100】接口调用成功</msg>
  </msgs>
</result>
```

步骤 4：参照输出的 xml 格式设置检查点，比如可以检查 compType 元素的值为 11.

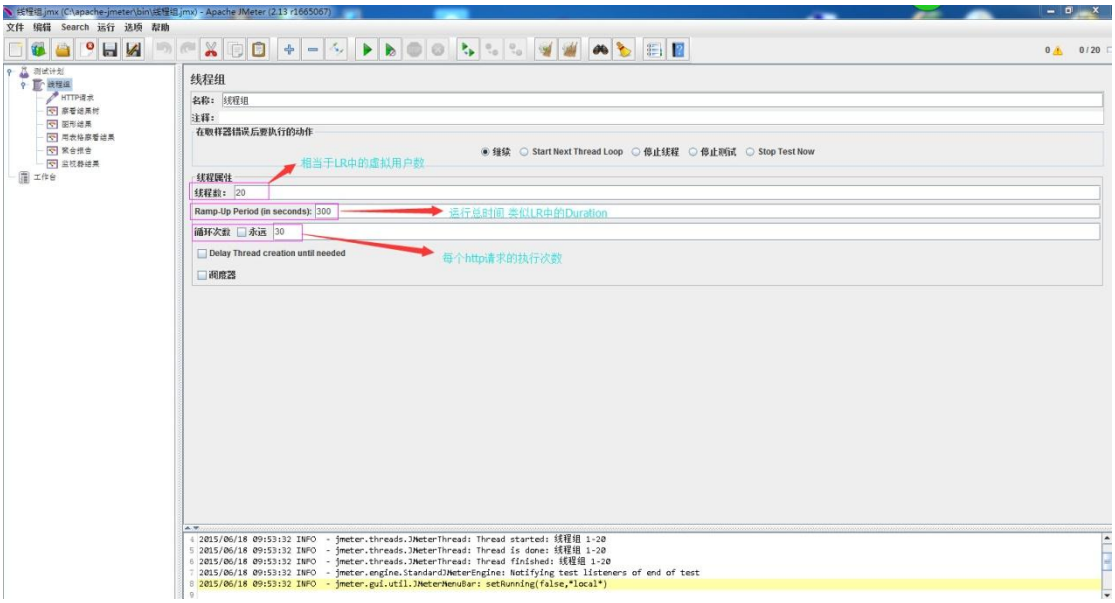
```
lr_xml_find("Xml={plain}",
        "Query=/result/datas/opPreComp/compType",
        "Value=11",
        LAST);
```

步骤 5：将输出语句注释掉，只保留 base64 编码和检查点语句。

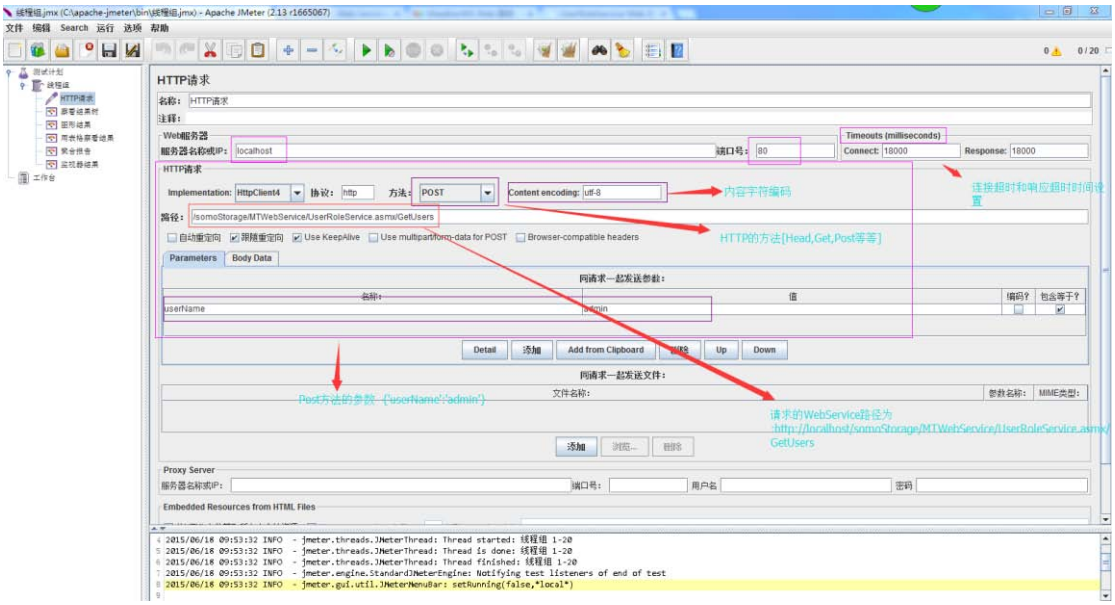
注：一般都是对非汉字的元素值设置检查点。因为 loadrunner 返回的汉字是乱码，有时会破坏 xml 标签样式。如上面的返回值就把的结束标签的样式破坏了。需要注意的是就算用 lr_convert_string_encoding 将"【100】接口调用成功"转换为 utf-8 再去做检查点判断时，也检查不到该内容，可能是内码转换后格式还是有些不同导致的，所以不要采用先转内码再对转化后的内码做检查点的方式。

4.2 使用 Jmeter 进行接口性能测试

1：新建测试计划--→添加线程组



2：添加 sampler—添加 HTTP 请求



设置服务器名为:localhost

HTTP 请求栏:

Implemetation：HttpClient4

协议为:http

方法：POST

Content encoding:utf-8

路径：/somoStorage/MTWebService/FileLibService.asmx/GetPatientStudy

添加参数:

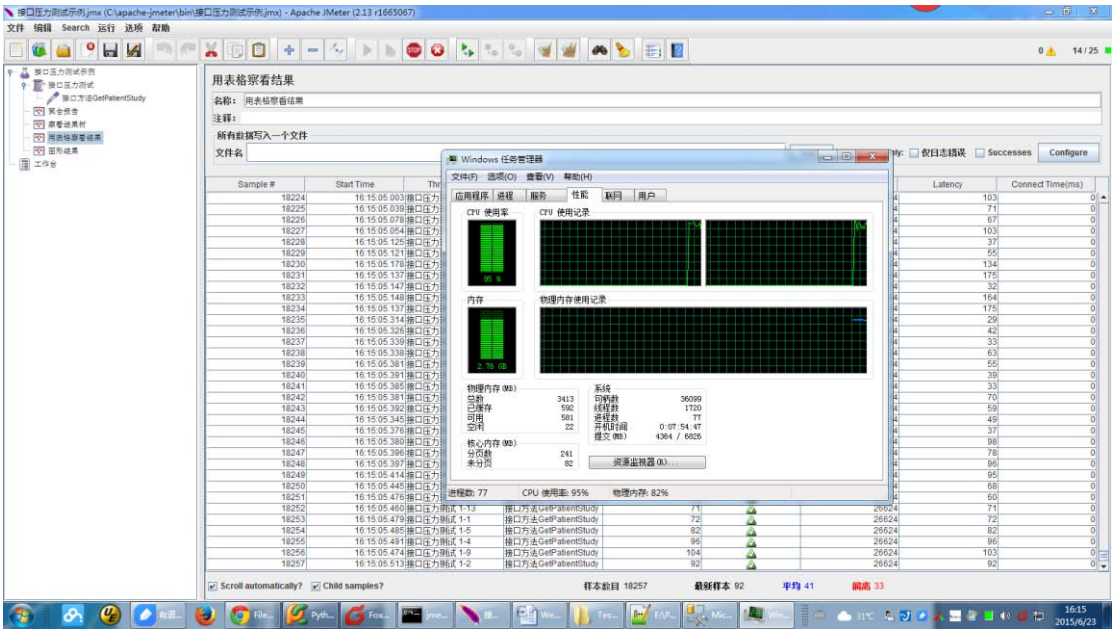
参数名: studyInstanceUID

参数值: 1.2.276.0.7230010.3.1.4.3856525867.14768.1367466243.413

在测试计划中添加监听器:

1:监听结果树　2: 结果汇总表格 3: 聚合报告

Windows 资源使用情况:



由于 Jmeter 负载机和 WebService 服务器均为本机 可能会出现内存溢出的情况，若 Jmeter 日志中报错 java.lang.OutOfMemoryError: Java heap space

调整 jmeter.bat 文件 HEAP 配置项　示例如下:

C:\apache-jmeter\bin\jmeter.bat - Notepad++ [Administrator]

文件(F) 编辑(E) 搜索(S) 视图(V) 格式(M) 语言(L) 设置(T) 宏(O) 运行(R) 插件(P) 窗口(W) ?

jmeter.bat

```
rem Unfortunately TechTips no longer seem to be available

rem See the unix startup file for the rationale of the following parameters,
rem including some tuning recommendations
set HEAP=-Xms1024m -Xmx1024m
set NEW=-XX:NewSize=512m -XX:MaxNewSize=512m
set SURVIVOR=-XX:SurvivorRatio=8 -XX:TargetSurvivorRatio=50%
set TENURING=-XX:MaxTenuringThreshold=2
rem Java 8 remove Permanent generation, don't settings the PermSize
if %current_minor% LEQ "8" (
    rem Increase MaxPermSize if you use a lot of Javascript in your Test Plan :
    set PERM=-XX:PermSize=512m -XX:MaxPermSize=512m
)

set CLASS_UNLOAD=-XX:+CMSClassUnloadingEnabled
rem set DEBUG=-verbose:gc -XX:+PrintTenuringDistribution

rem Always dump on OOM (does not cost anything unless triggered)
set DUMP=-XX:+HeapDumpOnOutOfMemoryError

rem Additional settings that might help improve GUI performance on some platforms
rem See: http://java.sun.com/products/java-media/2D/perf_graphics.html
```

从 14 年 8 月至今，历经菜渣，全手工写 Script 用 Robot+扩展封装库，再到 Jenkins 持续集成，终成型并稳定 UI 自动化测试框架，期间槽点颇多，心累不已....IT 有风险，切码且珍惜。

51testing 博为峰教育

高校软件测试一体化解决方案

毕业生难 就业竞争 压力大

毕业生人数连年创新高，学什么专业，找什么工作，成为毕业生心头的一块大石

学校难 学生培养 无从下手

软件测试领域接触少，缺乏课程体系、教学经验，不熟知企业实际项目运作形式，无法提供学生实践环境等

企业难 招聘人才 满意度低

软件测试方向应届生少，学生知识面窄，缺乏实践经验，无法快速承担企业工作需求

培养软件测试紧缺人才，博为峰怎么做？

高校软件测试实验室建设

课程+工具+实验+项目+培训一体化实验室平台，快速提升软件测试教学水平

高校大学生就业培训项目

校企学生三赢模式，定向培训，保证就业，解决学生就业问题

高校大学生软件测试实训

校内或校外中短期集中培训，快速掌握理论知识，积累实战经验

为什么选择博为峰？

上海博为峰软件技术股份有限公司（51Testing）成立于2004年，是中国领先的软件测试服务供应商，总部位于上海，在北京、深圳、成都、南京、西安、杭州分别设有分支机构。公司专业从事软件测试工具研发、培训、外包等服务，为高等院校提供软件测试人才培养并解决大学生就业问题。

- 2004年推出国内首家软件测试工程师培训，开班400多期，培养人数超20000人
- 帮助合格毕业学员成功就业，就业率高达到99.5%
- 为226家国内外知名企业提供企业内训
- 服务700余家国内外各行业企业，对国内外企业的用人需求及行业发展把握精准
- 自研软件测试实验室平台TestPlatform荣获国家创新基金
- 自研软件测试实验室平台TestPlatform荣获上海市高新技术成果转化项目认定
- 自研软件测试实验室平台TestPlatform荣获上海市创新基金

实验室建设咨询电话：400-888-0051（全国）

实训及就业培训咨询电话：

博为峰中国上海 40088-51518　博为峰中国北京 40088-40018　博为峰中国深圳 40088-51008

博为峰中国成都 40088-90051　博为峰中国南京 40082-15251　博为峰中国西安 40082-15107