目 录

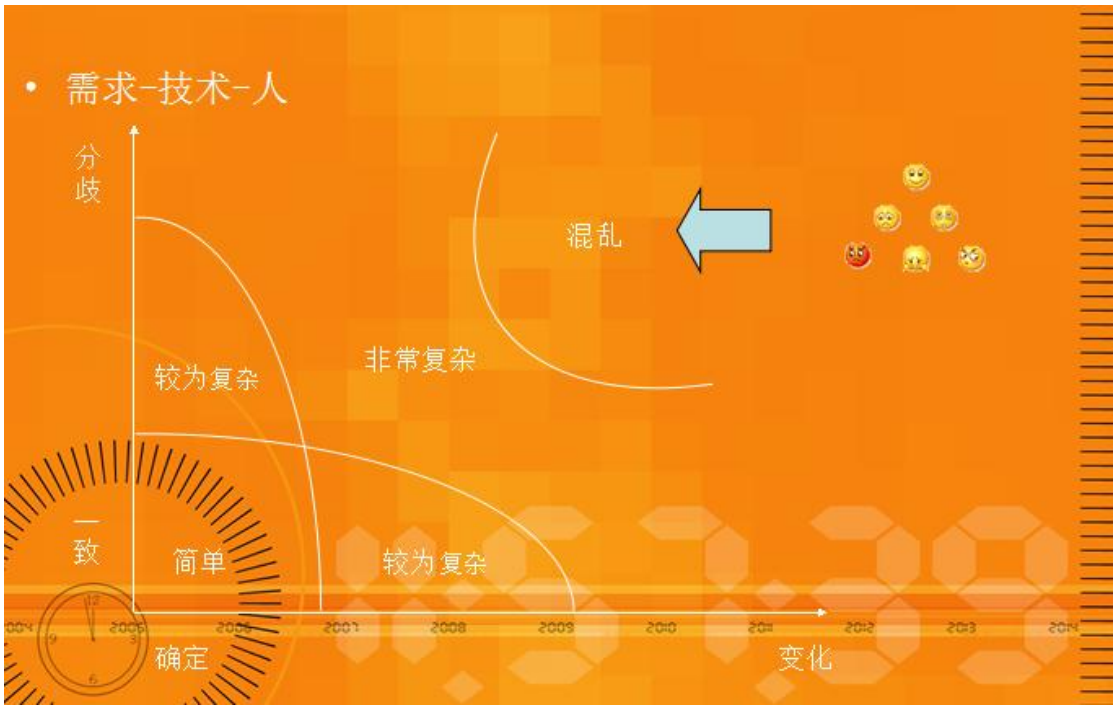
我在华为做敏捷测试的那些流程

◆作者：刘利方

一、开发和测试的通性困扰？

面对复杂性（客户）：不断地修改计划、不断地增加预算、低劣的产品质量……

面对复杂性（项目组成员）：经常加班到深夜、提交的产品不合格……



二、敏捷开发中的敏捷测试目的：

敏捷宣言

个体和交互比过程和工具更有价值；能工作的软件比全面的文档更有价值；顾客的协作比合同谈判更有价值；及时响应变更比遵循计划更有价值。

其核心是:以人为本,发挥人的主观能动性.

三、传统测试和华为敏捷测试区分：

3.1、传统的测试

1.守门员:质量保证者，阻止那些不可靠的、无效的、充满 BUG 的版本发布。

2.信息提供者:提供大量积极的、关于项目开发的状态的信息。告诉大家哪些功能正常工作、哪些功能不能正常工作、哪些 BUG 必须处理。

3.2、华为敏捷测试

测试和开发的角色界线变得模糊。有些人主要做测试工作，有些人主要做开发工作，但是在快速推进的过程中，所有人都会被号召起来测试或支持测试的工作。

更多职责:帮助开发人员理解需求,尽早确定测试规范。

3.3、敏捷测试中测试人员扮演的角色

1.测试是项目的“车头灯”，它告诉大家现在到哪了，正在往哪个方向走。

2.测试为项目组提供信息，使得项目组基于可靠的信息作出正确的决定。

3.测试人员不作出项目发布的决定。

4.测试员不保证质量，整个项目组对质量负责。

5.测试不是抓虫子的游戏，它的目的不是纠缠在错误中，而是帮助找到目标。

四、敏捷测试用例的设计和评审要素:

4.1、基于需求的用例场景来设计测试用例:

1.基于需求的用例场景来设计测试用例是最直接有效的方法，因为它直接覆盖了需求，而需求是软件的根本，验证对需求的覆盖是软件测试的根本目的。

2.把测试用例当成“活”的文档,因为需求是“活”的、善变的。因此在设计测试用例方面应该符合敏捷的“及时响应变更比遵循计划更有价值”这一原则。

3.测试用例的设计不是一个阶段，测试用例的设计也需要迭代，在软件开发的不同的阶段都要回来重新审视和完善测试用例。

4.2、敏捷测试用例设计原则

通常我们所看到的测试用例的设计是其中一项。

测试用例可以写得很简单，也可以写得很复杂。最简单的测试用例是测试的纲要，仅仅指出要测试的内容，如探索性测试中的测试设计，仅会指出需要测试产品的哪些要

素、需要达到的质量目标、需要使用的测试方法等。而最复杂的测试用例就像银行取款机系统中工作指令系统界面一样，会指定输入的每项数据，期待的结果及检验的方法，具体到界面元素的操作步骤，指定测试的方法和工具等等。

测试用例写得过于复杂或过于详细，会带来两个问题：一个是效率问题，一个是维护成本问题。另外，测试用例设计得过于详细，留给测试执行人员的思考空间就比较少，容易限制测试人员的思维。

测试用例写得过于简单，则可能失去了测试用例的意义。过于简单的测试用例设计其实并没有进行“设计”，只是把需要测试的功能模块记录下来而已，它的作用仅仅是在测试过程中作为一个简单的测试计划，提醒测试人员测试的主要功能包括哪些而已。测试用例的设计的本质应该是在设计的过程中理解需求，检验需求，并把对软件系统的测试方法的思路记录下来，以便指导将来的测试。

大多数测试团队编写的测试用例的粒度介于两者之间。而如何把握好粒度是测试用例设计的关键，也将影响测试用例设计的效率和效果。我们应该根据项目的实际情况、测试资源情况来决定设计出怎样粒度的测试用例。

软件是开发人员需要去努力实现敏捷化的对象，而测试用例则是测试人员需要去努力实现敏捷化的对象。要想在测试用例的设计方面应用“能工作的软件比全面的文档更有价值”这一敏捷原则，则关键是考虑怎样使设计出来的测试用例是能有效工作的。

4.3、敏捷中测试用例评审：

1. 同行评审是最敏捷的检查测试用例的方式，它主要强调测试用例设计者之间的思想碰撞、互补，通过讨论、协作来完成测试用例的设计，原因很简单，测试用例的目的是尽可能全面地覆盖需求，而测试人员总会存在某方面的思维缺陷，一个人的思维总是存在局限性。因此需要一起设计测试用例，从而体现敏捷的“个体和交互比过程和工具更有价值”。

2. 除了同行评审，还应该尽量引入用户参与到测试用例的设计中来，让他们参与评审，从而体现敏捷的“顾客的协作比合同谈判更有价值”这一原则。

备注：这里顾客的含义比较广泛，关键在于你怎样定义测试，如果测试是对产品的批判，则顾客应该指最终用户或顾客代表（在内部可以是市场人员或领域专家）；如果测试是指对开发提供帮助和支持，那么顾客显然就是程序员了。

五、华为常用敏捷的测试方法和测试流程:

1. 进行敏捷测试主要分以下几个步骤:

敏捷测试采用的功能测试方法是我们常用的基本的测试方法，例如:

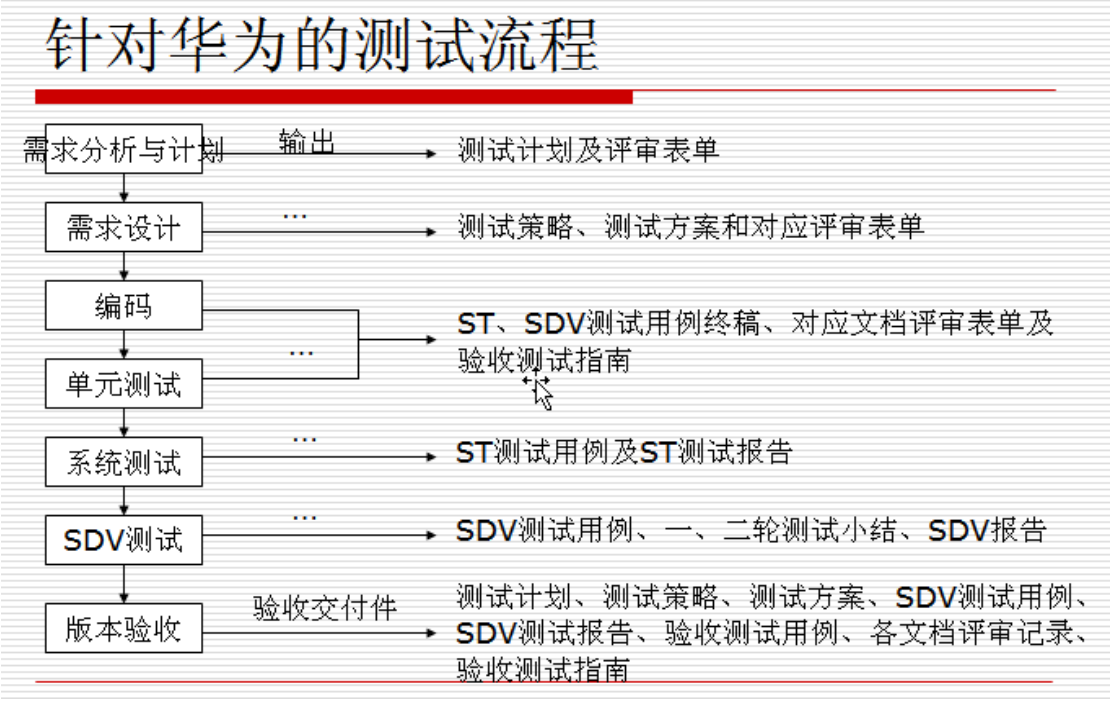
等价类划分、

边界值分析法、

错误推测法……

2. 敏捷测试和传统测试不同的是:

- 需要高度的迭代工作、
- 频繁得到客户的反馈，
- 需要动态调整测试计划、
- 测试的执行。



3. 敏捷测试过程中性能测试方法和考虑:

性能测试的方法?

性能测试、负载测试、压力测试、配置测试、并发测试

- 性能测试：通过模拟生产运行的业务压力量和使用场景组合，测试系统的性能

是否满足生产性能要求。

- 负载测试：通过在被测系统上不断增加压力，直到性能指标，例如“响应时间”超过预定指标或者某种资料使用已经达到饱和状态。
- 配置测试：通过对被测系统的软/硬 件环境的调整，了解各种不同环境对系统性能影响的程度，从而找到系统各项资源的最优分配原则。
- 压力测试：测试系统在一定的饱和状态下，例如 CPU、内存等在饱和使用情况下，系统能够处理的会话能力，以及系统是否会出现错误。
- 并发测试：通过模拟用户的并发访问，测试多用户并发访问同一个应用、同一个模块或者数据记录时是否存在死锁或者其他性能问题。

六、与您共勉

最后送大家一句话，兵无常势，水无常形，能因敌变化而取胜者谓之神，相信在测试的过程程中只有找对了方法，不管是传统的测试还是现当今国外比较流程的敏捷测试，只要应用得当就是好的测试。

如何与开发沟通功能实现

◆ 作者：搜狗测试 Deadwalk

测试工程师日常工作中，经常会与其他团队角色进行沟通，这其中难免会出现一些沟通的问题，这些问题需要更多地沟通技巧来解决。本次小编想跟大家分享一下：如何与开发沟通功能实现。

某测试同学为了测试一个功能，需要了解功能的实现逻辑，所以她满脸笑容地找到开发同学后说道:”你给讲讲 Cookie 同步是怎么实现的吧!”

开发同学不耐烦道：“说了你也不懂。”

以上情景相信不少同学遇到过吧，小编分享下自己在与开发沟通功能实现方面的技巧:

a) 沟通的时机很重要

小编以前做开发的时候，最大的感受就是来自于实现功能的压力巨大，特别是项目比较紧急的时候，所以当开发同学正在赶一个需求时，需要注意力高度集中，特别是遇到棘手的问题。此时，如果测试同学去沟通实现，很有可能碰一鼻子灰。

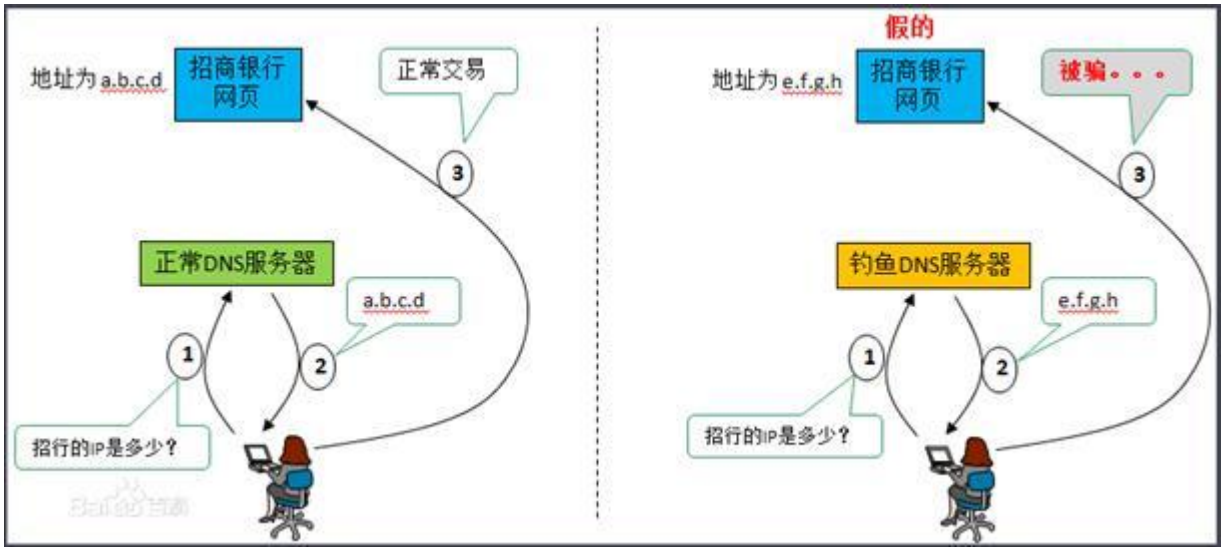
建议：在对方有空的时候或者精神相对不紧张的时候，沟通一般会比较顺畅。

b) 提前准备很重要

小编见过有些测试同学在沟通实现前，没有对要沟通的功能做一丝的准备和思考，直接找到开发同学说你给我讲讲这个功能如何实现的，这个时候开发同学可能要花大量的时间先做基本的背景知识普及和讲解，而本身应该沟通的功能实现却被忽略了，而且测试同学后续产生的提问也是有边没边地乱冒一通，自然是效率低下。

建议:

1) 在沟通功能实现前，测试同学提前做一些背景知识的了解，例如：在上例中，我们要了解下 Cookie 字段有哪些；Cookie 一般在网站的登录中使用；Cookie 分为文件 Cookie 和内存 Cookie 等等。



(以上图示是测试同学在进行 DNS 劫持测试前，对 DNS 劫持的了解画成了图例)

2) 提前准备好要沟通的问题列表，可以记在本子上或电子版，沟通时一个接一个地提问，免得现想问题浪费时间。

2. SystemRankUrl清除历史数据时不被清除，只有用户在下拉列表中点击“x“时才会被删除
--被清除是指从数据库里删除？
是，userrank删除后是标记该URL的deleteflag=1，因为历史记录里的url没有删，所以这里删除数据库没有意义
3. 在获取鼠标对应元素的属性时，如果遇到iframe嵌套iframe，需要根据iframe转换鼠标坐标以获得元素的属性。
--求解释，和地址栏的关系
没关系
4. SE_DragDrop执行过程中会调用IE_DragDrop；同样的SE_DragEnter执行过程中也会调用IE_DragEnter。
--写明这句话的用意何在？
没关系
5. 当后台的某个标签页访问的URL为下载链接时，浏览器如何判断此标签页需要变为激活的标签页？
当点击下载链接后，IE会触发DISPID_FILEDOWNLOAD事件，在此事件的回调函数中，SE将此标签页标识为前台打开。

(以上图示是测试同学在沟通前准备的问题列表，黄色底色为问题，绿色底色为问题答案)

c) 第一个问题很重要

在沟通实现时的第一个问题，如果是比较”大而泛”，沟通结果一般不会太理想，比如：“Cookie 同步功能是怎么实现的？”

开发同学更喜欢回答一些具体的、技术性的、非开放性的问题。上例中如果我们换个提问方式:

QA: 浏览器不同进程之间是如何传递数据的?

DEV: 通过发送消息的方式，使用 FileMappming 进行多进程传递。

QA: 传递的时机是什么时候?

DEV: 用户在浏览器中登录网站时，触发了 Cookie 的读写操作时。

QA: 浏览器是怎么检测到用户产生了登录行为?

DEV: 浏览器对网络返回值中的 Set-Cookie 字段进行了检测，一旦发现该字段，则会解析其内容并进行保存和同步操作。

……（借着以上问题继续展开沟通）

建议: 沟通时第一个问题不要提开放性的问题。

d) 提问的方式很重要

以前小编和一位开发大牛聊过测试和开发沟通实现的议题，开发大牛表示，他们更愿意测试同学是思考后带着一些想法来沟通的，即便这种想法是错的，也是乐意欢迎的。所以，我们可以准备一些自己对这个功能实现过程的猜测，然后用自己猜测的功能实现来进行提问。

例如: 我们不知道 Cookie 数据如何进行多进程传递时，我们可以猜测 Cookie 数据是不是保存在数据库中，不同进程读取同一份数据库来进行数据的交互，然后带着这种猜测来进行发问:

QA: “多进程数据同步是不是用数据库进行数据交换的？”

DEV: “通过数据库进行数据交互可能存在这锁的问题，所以我们不是通过这种方法而是用 FileMappming 文件映射的方式……”

……（顺畅地交流起来）

e) 纸上的交流比口头交流好

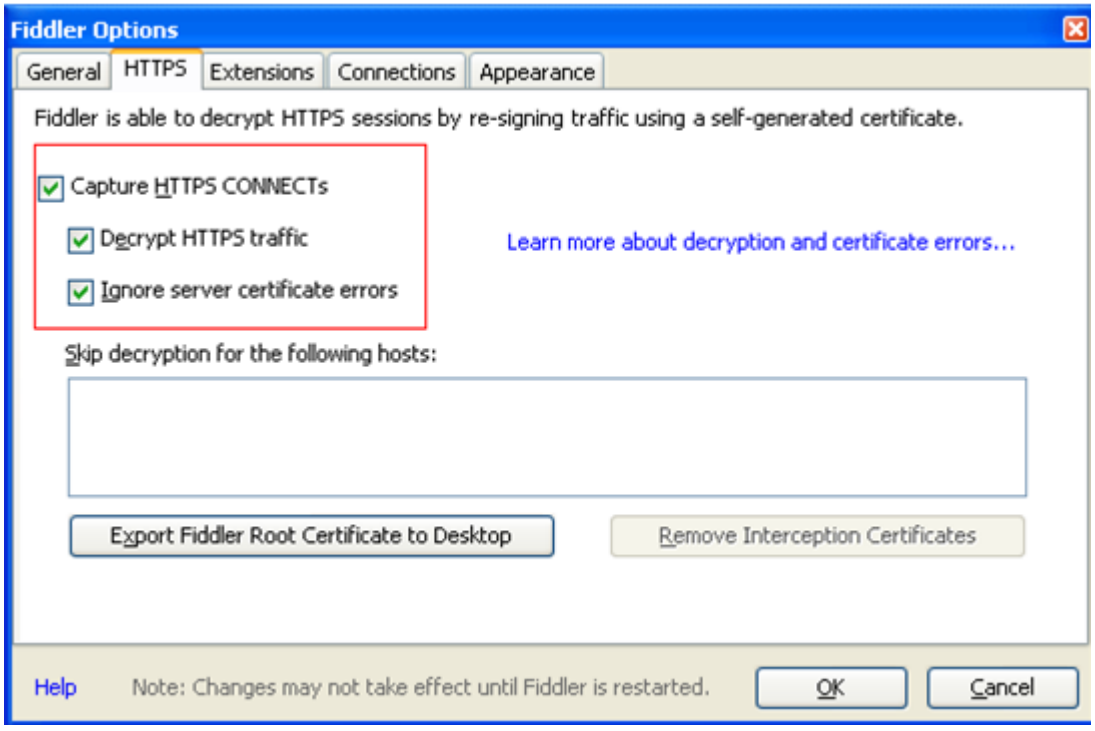
在沟通过程中，简单的问题可以口头交流，但是复杂的问题建议在纸上画着流程图或者程序的框架图会更加容易交流沟通。所以，避免只停留在口头上的沟通。

Fiddler 使用十二个小技巧

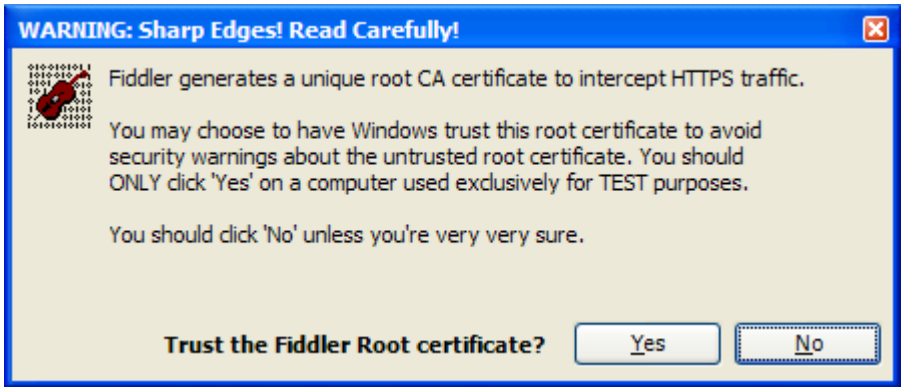
◆作者：搜狗测试 Janice

技巧一：全新安装 **Fiddler**

默认下，Fiddler 不会捕获 HTTPS 会话，需要你设置下，打开 Fiddler Tool->Fiddler Options->HTTPS tab



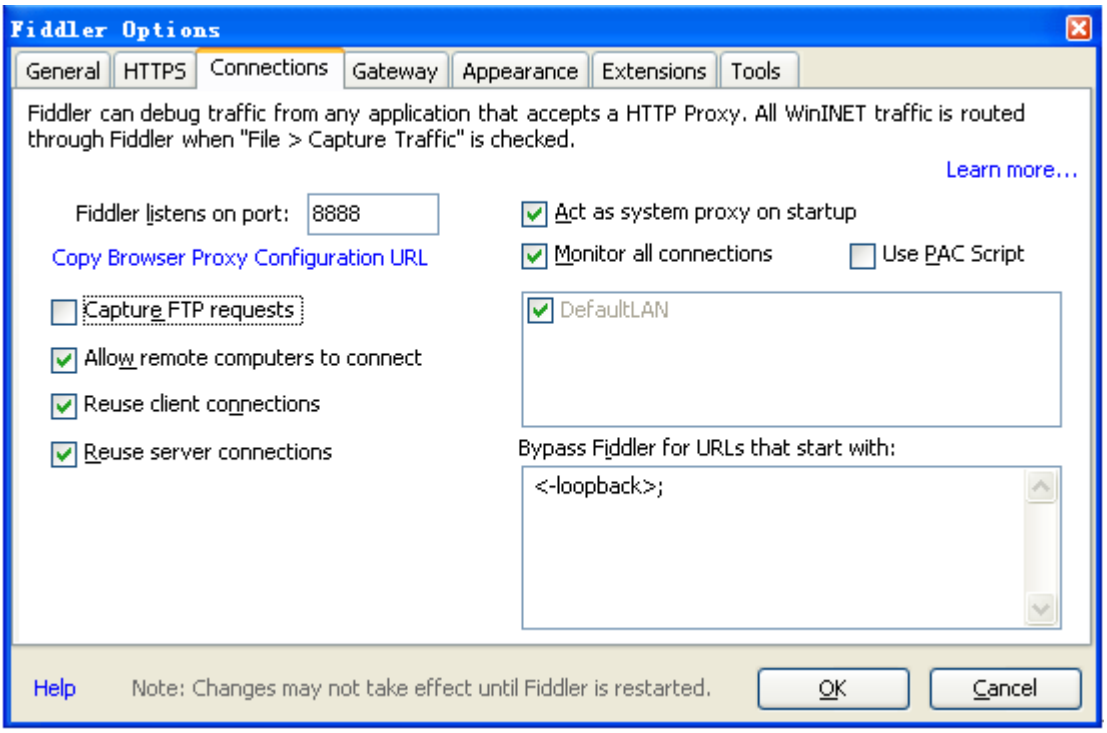
选中 checkbox，弹出如下的对话框，点击"YES"



点击"Yes"后，就设置好了

技巧二：Fiddler 连接手机代理配置:

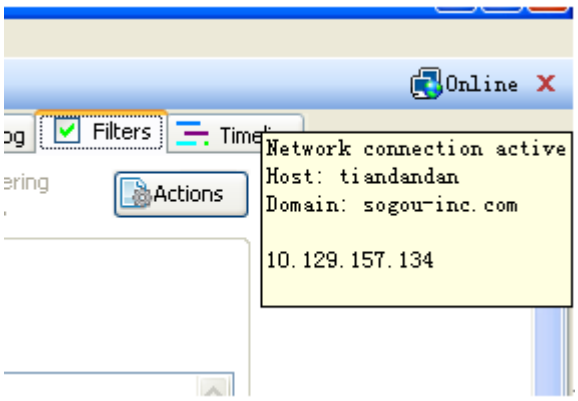
1) 依次打开 Fiddler->Tools->Fiddler Options 在【Connection】面板里将 Allow remote computers to connect 勾选，设置端口号【默认为 8888，可修改其他端口号】；点击【OK】按钮，关闭 Fiddler 并重新打开 Fiddler。



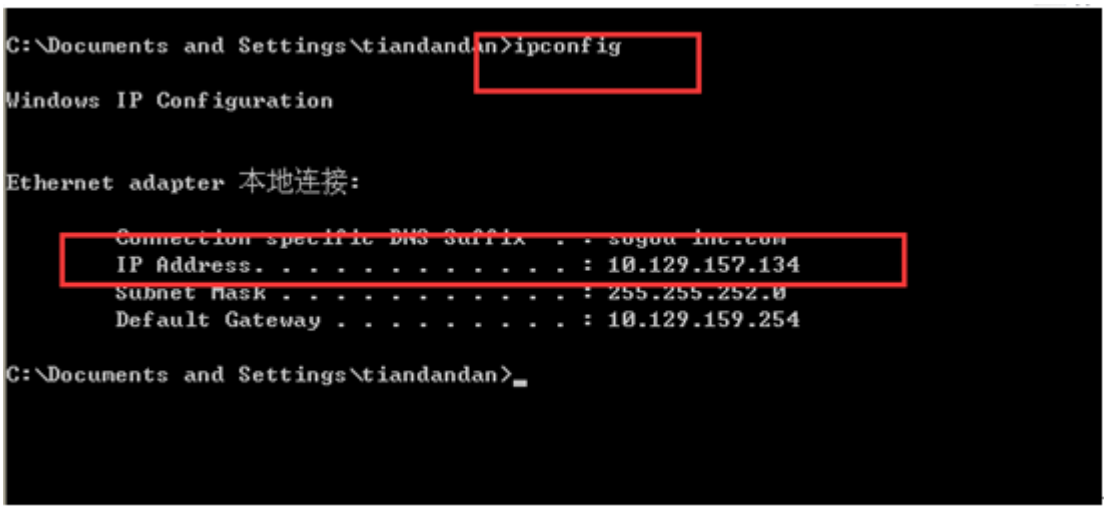
2) 得到本机 IP 用于手机端配置，windows->运行->cmd->ipconfig；得到本机 IP

简便获取本机的 IP 有俩种办法:

①鼠标放置在 fiddler 右上角的 Online，就会显示本机的 IP、Host 等一些信息，如下图



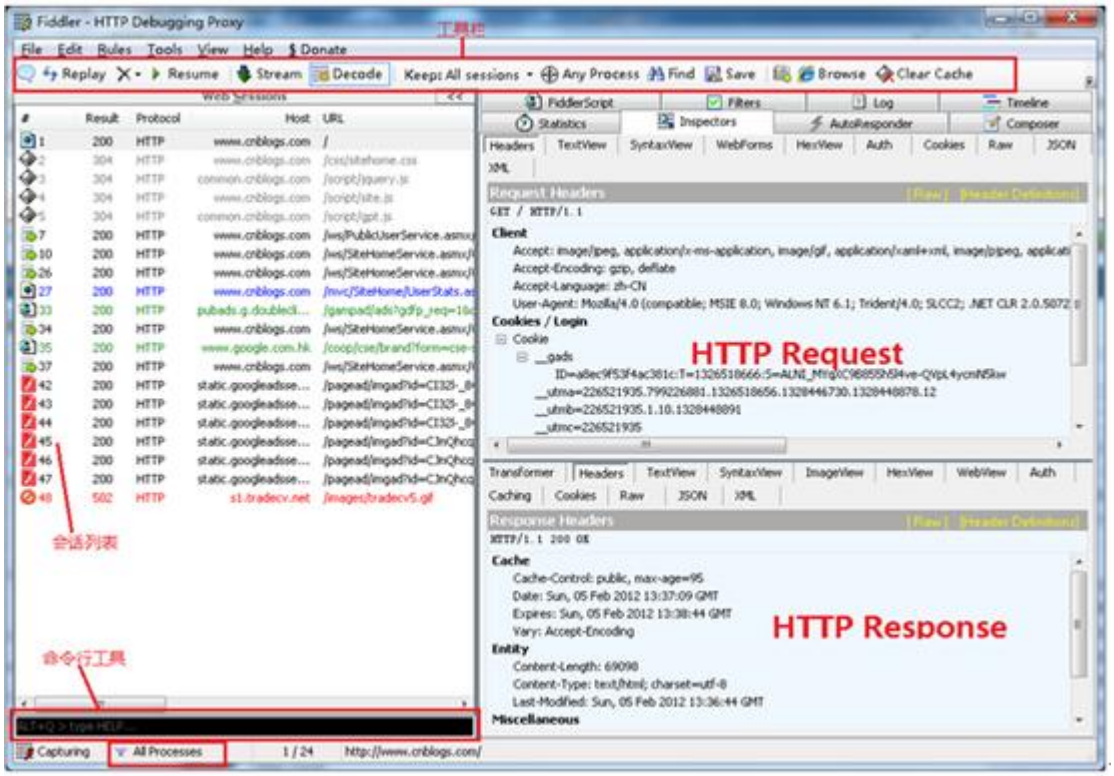
②点击 windows->运行->cmd->ipconfig；得到本机 IP



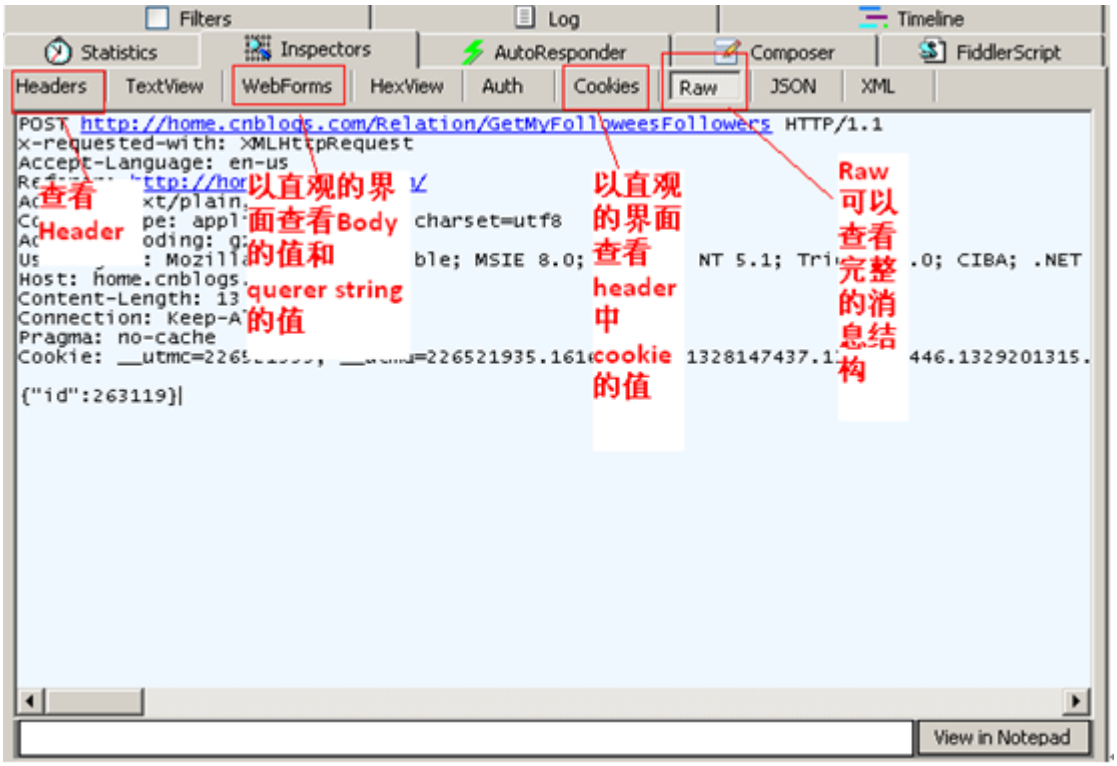
3) 手机端配置：代理设置更改为【手动】，输入【代理服务器主机名】（对应 PC 端的 IP 地址），输入【代理服务器端口号】（Fiddler 配置的端口号）

4) 全部配置完毕，现在用手机打开助手，就能在 Fiddler 中监听到手机发送的请求

技巧三：**Fiddler 的基本界面**



nspectors tab 下有很多查看 Request 或者 Response 的消息。其中 Raw Tab 可以查看完整的消息，Headers tab 只查看消息中的 header.如下图

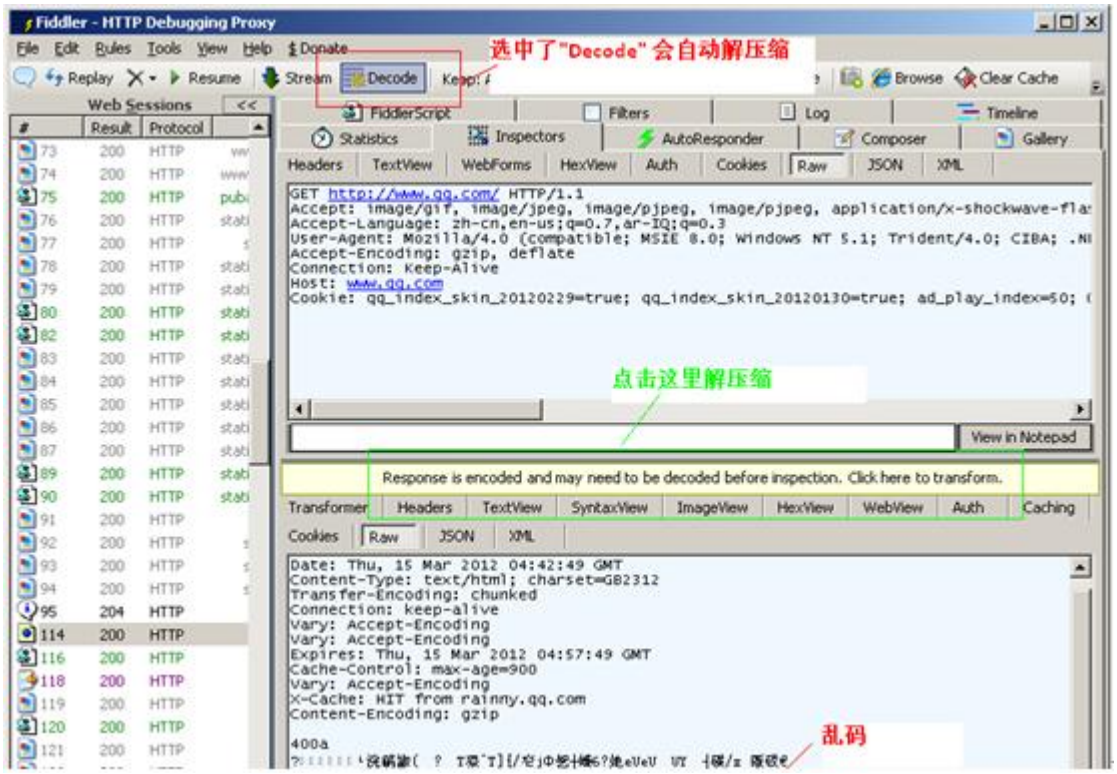


技巧四：启动 Fiddler 后 Response 是乱码，怎么办？

有时候我们看到 Response 中的 HTML 是乱码的，这是因为 HTML 被压缩了，我们可以通过两种方法去解压缩。

1. 点击 Response Raw 上方的"Response is encoded any may need to be decoded before inspection. click here to transform"

2. 选中工具栏中的"Decode"。这样会自动解压缩，解压后重启就可以了



技巧五：QuickExec 命令行的使用

Fiddler 的左下角有个命令行工具叫做 QuickExec，允许你直接输入命令。

常见的命令有:

help: 打开官方的使用页面介绍，所有的命令都会列出来

cls: 清屏

Select: 选择会话的命令

?.png: 用来选择 png 后缀的图片

bpu: 截获 request

技巧六：Fiddler 中创建 AutoResponder 规则

Fiddler 的 AutoResponder tab 允许你从本地返回文件，而不用将 http request 发送到服务器上

看个实例.

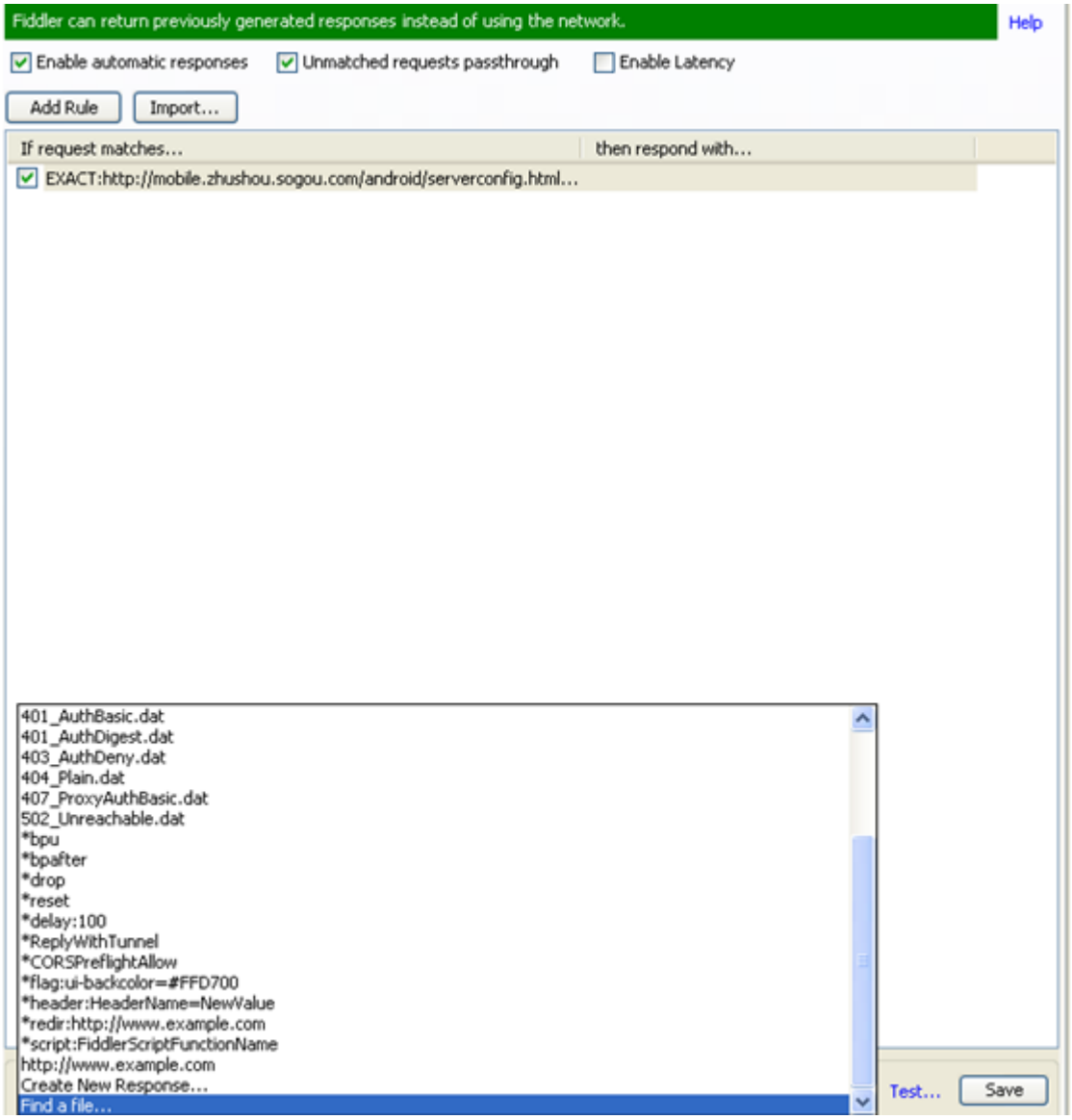
1. 进入助手首页，把 serverconfig.html 连接保存到本地桌面，选中该条请求，右键—>copy àJust Url

2. 选中该条请求，点击右侧选择 AutoResponder，点击 Add Rule，把该条请求添加进去，或者把这个会话拖到 AutoResponer tab 下

3. 选择 Enable automatic reaponses 和 Unmatched requests passthrough

4. 更改本地保存的配置文件，在下面的 Rule Editor 下面选择 Find a file...选择本地保存的图片.最后点击 Save 保存下

5. 重新首次进入助手，查看 serverconfig 返回的数据中就是自己修改的

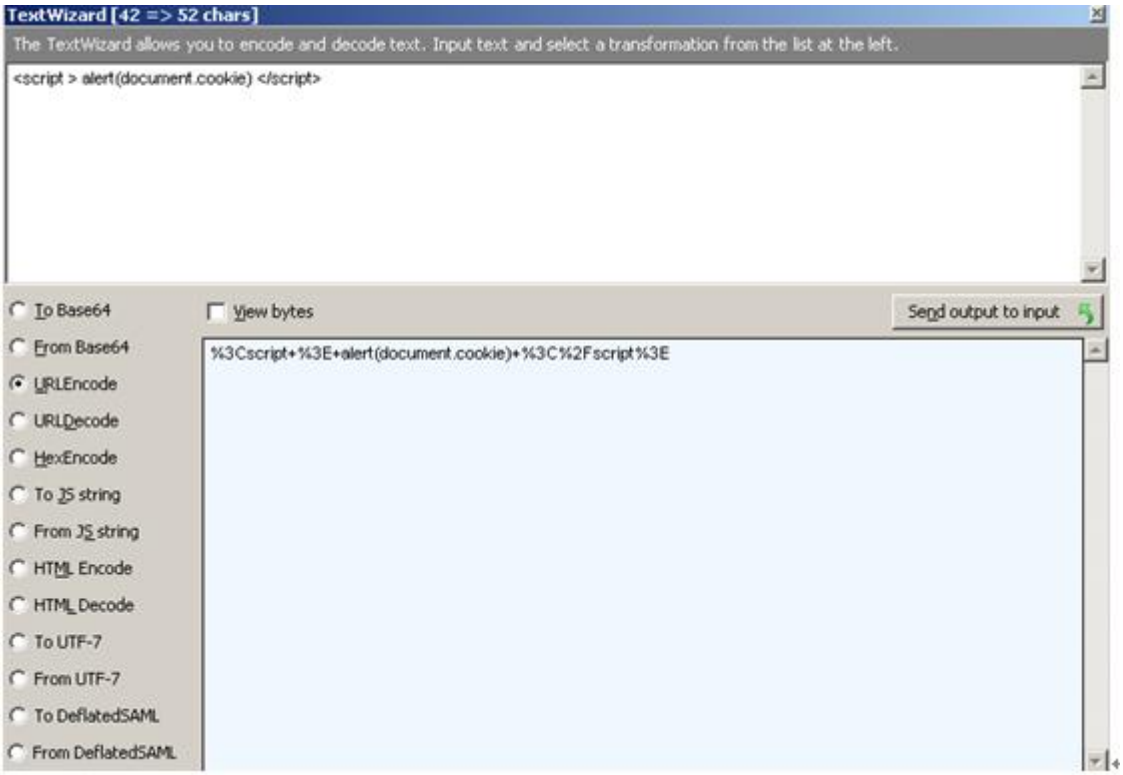


技巧七：Fiddler 中如何过滤会话

每次使用 Fiddler，打开一个网站，都能在 Fiddler 中看到几十个会话，看得眼花缭乱。最好的办法是过滤掉一些会话，比如过滤掉图片的会话。Fiddler 中有过滤的功能，在右边的 Filters tab 中，如下图就是只显示带有 zhushou.sogou.com 的请求

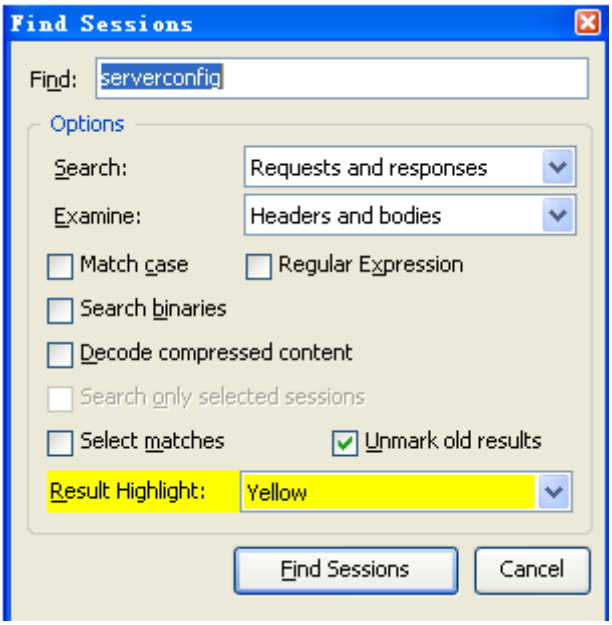
技巧八：Fiddler 中提供的编码小工具

点击 Fiddler 工具栏上的 TextWizard，这个工具可以 Encode 和 Decode string.



技巧九：Fiddler 中查询会话

用快捷键 Ctrl+F 打开 Find Sessions 的对话框，输入关键字查询你要的会话。查询到的会话会用黄色显示



技巧十：Fiddler 中保存会话

有些时候我们需要把会话保存下来，以便发给别人或者以后去分析。保存会话的步骤如下：

选择你想保存的会话，然后点击 File->Save->Selected Sessions

CustomRules.js

CustomRules.js 中的主要方法：

Static function OnBeforeRequest(oSession： Session)//在这个方法中修改 Response 的内容，

Static function OnBeforeResponse(oSession： Session)

//在个方法中包含 Fiddler 命令。在 Fiddler 界面中左下方的 QuickExec Box static function OnExecAction(sParams： String[])

实例：修改 sogouid

① 在菜单栏中点击 Rules—》CustomRules，打开 CustomRules.js 脚本

② 添加如下脚本，（）里面写上接口名字，url 中填上请求数据

③ 更改你想要改的 sogouid 就可以

把这段脚本放在 oSession.uriContains（）方法下，并且点击"Save script"，这样该条

接口下的 sogouid 就会变成自己更改的 id

if(oSession.uriContains("install.html?")){

oSession.url="mobile.zhushou.sogou.com/m/install.html?uid=d15449c17bbded35c98973670c1e1e0c&vn=3.11.2&channel=sogouinputgx&sogouid=e9ed8a54201e5481e20f6760804772c3&stoken==IhTefovaz0ppdInTQxRlnQ&cellid=&sc=0&iv=311";

技巧十一：修改 session 在 Fiddler 的显示样式

把这段脚本放在 OnBeforeRequest(oSession: Session)方法下，并且点击"Save script"，这样所有的 cnblogs 的会话都会显示绿色。

if (oSession.uriContains("mobiletoolhit.gif?")) {oSession["ui-color"] = "green"; }

技巧十二：如何在 Fiddler Script 中修改 Request 中的 body

方法一：

Static function OnBeforeRequest(oSession: Session) {

if(oSession.uriContains("http://www.cnblogs.com/TankXiao/")) {

//获取 Request 中的 body 字符串

Var strBody=oSession.GetRequestBodyAsString();

//用正则表达式或者 replace 方法去修改 string

```
strBody=strBody.replace("1111","2222");
//弹个对话框检查下修改后的 body
FiddlerObject.alert(strBody);
//将修改后的 body，重新写回 Request 中
oSession.utilSetRequestBody(strBody);  }}
```

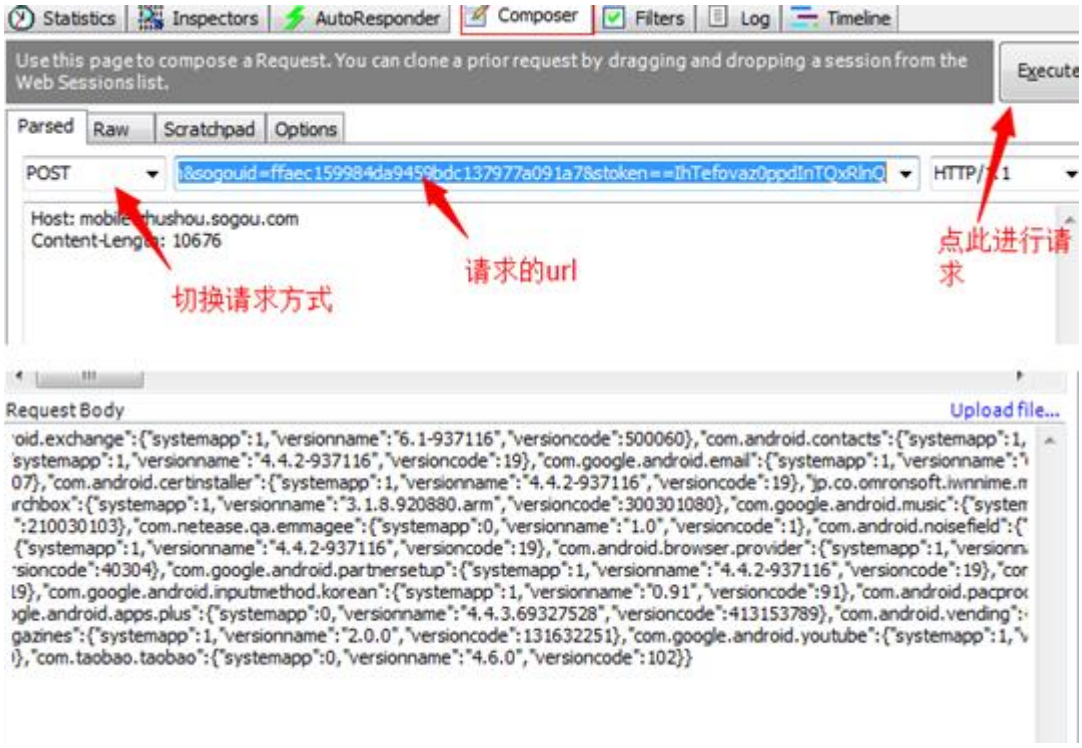
方法二:

提供一个简单的方法，可以直接替换 body 中的数据

```
oSession.utilReplaceInRequest(“ 1111”,“2222”);
```

post 请求模拟

测试时如果客户端的数据出错，可能需要单独对某个 url 进行请求，看返回数据是否正确，通过客户端再去请求有时需要复杂的环境需求，整一次挺费劲，通过单独的请求看数据是否正确可以节省一些时间，当然 get 请求可以直接用浏览器就能看到返回数据了，post 数据由于有 body 数据，需要借助工具才行，fiddler 的 composer 功能就是为此而生的，如下图，在上方填写 url，选择请求方式，在下方的 body 中添加 post 的数据后，点击 execute 就是请求了，在通过 inspectors 就能看到本次模拟请求的返回值了。



搜狗灰盒用例设计思想和方法

◆ 作者：搜狗测试 deadwalk

前言

本章内容旨在向大家介绍桌面组的测试设计思路和方法，不论你是新加入测试的初学者，还是从事一段测试的实践者，通过阅读这篇文章相信可以帮助你更加深入地了解桌面组的测试。

我们在测试过程中经常会遇到以下问题：

- 如何进行测试设计？
- 测试设计的过程是怎样的？
- 什么是测试对象？为什么要进行测试对象的拆分？
- 常见的测试用例设计方法有哪些？
- 如何进行开发实现了解在灰盒层面进行测试？
- 为什么要进行灰盒层面的测试？
- 如何提升自己的测试发散度？

以上诸多问题可能是新加入测试的同学或者从事一段时间测试之后，同学们常见的问题，下面我们将逐一解答大家的问题。

一、测试流程介绍

当大家拿到一个新产品或新功能时，有没有考虑如何进行测试？

答案是多种多样的。有些人可能会说要严格的按照软件工程的控制过程，进行单元测试、集成测试、系统测试等等；有些人可能会说直接进行黑盒的测试设计，然后执行

黑盒测试用例就行，保证黑盒测试做好就行；有些人甚至可能会说不用做复杂的测试设计，直接把产品拿来就用，当做一个实际用户来进行产品的使用即可。

这些答案，我觉得是仁者见仁智者见智，不同项目的质量要求、不同项目的迭代周期，不同项目所处阶段决定了它的测试方法、测试流程。之前曾听说有些项目组(如手机输入法)每天就进行大量的随机测试，但是同样能够保证软件的质量。从这个问题，我们可以看到测试流程在整个测试中的重要性，首先我们先讨论这个测试流程。

是不是测试流程一定要具备类似于传统软件行业的双 V 模型(后续会做介绍)等规范化的流程才能称之为测试流程？我个人的意见是不一定。互联网软件行业最大的一个特征就是快速迭代，通过不断地、持续地进行增量式的迭代开发，在较短的周期内进行软件需求、设计、开发、测试并上线。虽然这个过程中各方没有规范化的文档，没有按照规范化的流程来做事，但是通过快速迭代吸引到大量的用户，这个收益和战略意义远比完备的文档要大得多。

所以，我们首先要想大家介绍的是，桌面测试组目前的开发模型和测试流程。

1. 常见的软件开发模型：瀑布模型和双 V 模型

我们先了解一下常见的软件开发模型有哪些。一般常见的软件开发模型有：瀑布模型、V 模型、双 V 模型，这类开发模型被应用于中小型企业的软件开发过程中。不太常见的有 RUP 模型、螺旋模型等，这类一般用于大型的复杂软件开发过程，例如航天相关软件、银行系统等。

(参考资料：http://baike.baidu.com/view/8300.htm)

a.瀑布模型:

该模型给出了固定的顺序，将生存期活动从上一个阶段向下一个阶段逐级过渡，如同流水下泻，最终得到所开发的软件产品，投入使用。

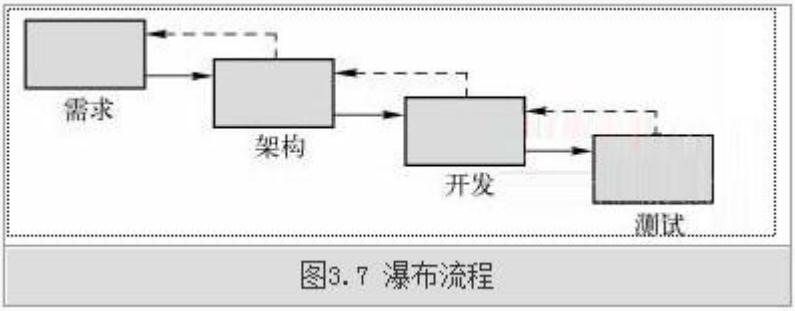
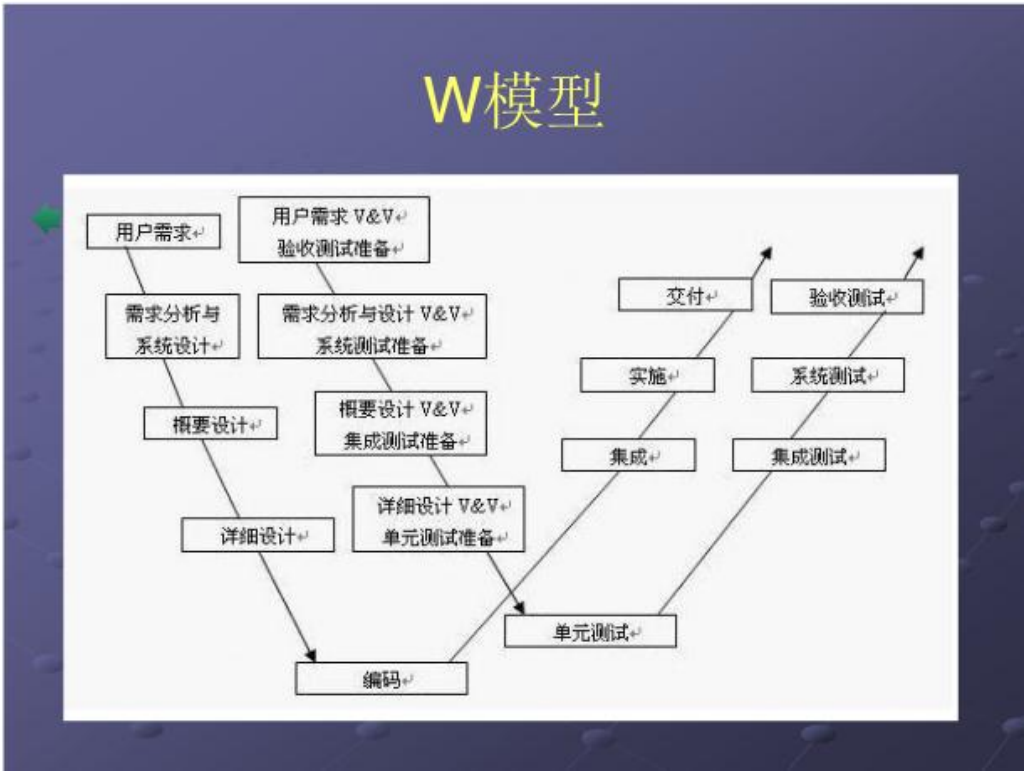
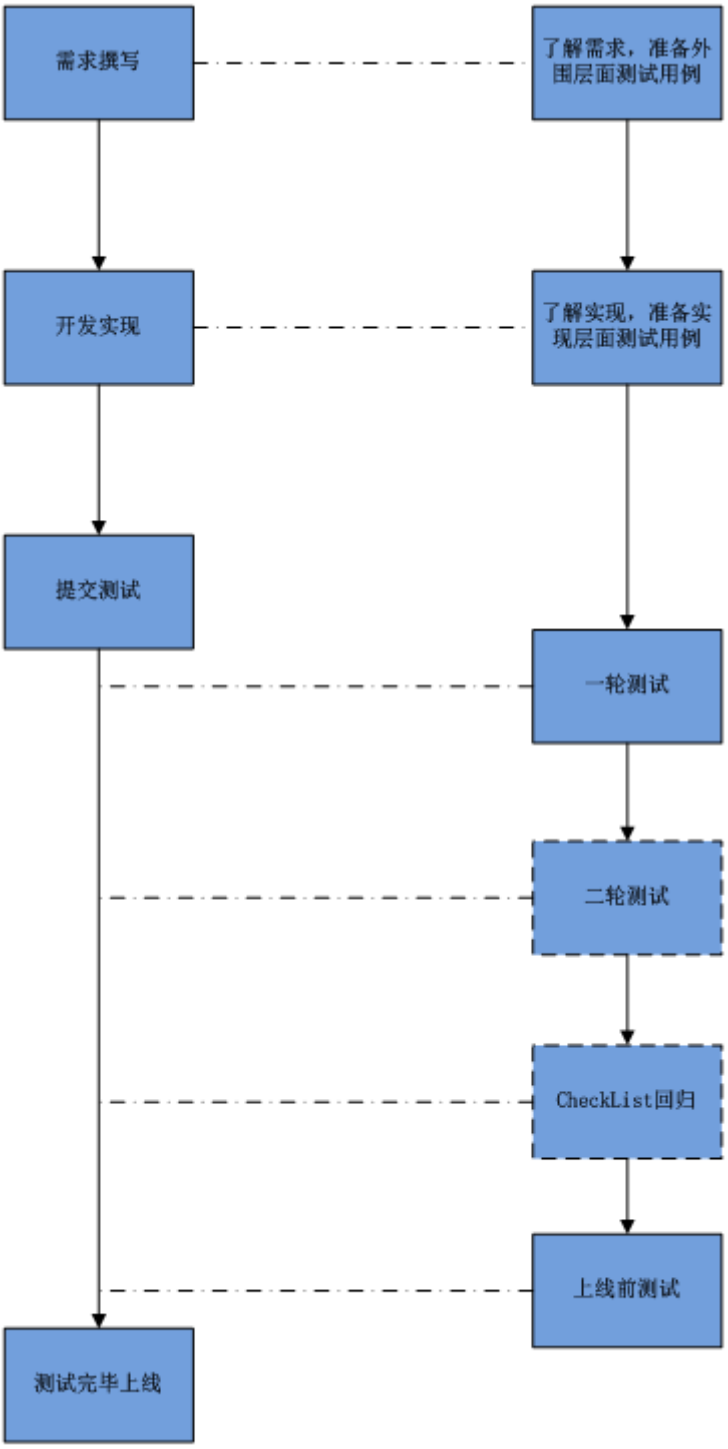


图3.7 瀑布流程

b.双 V 模型:



2. 搜狗桌面组的开发模型:



a.相同点：

i.测试尽早介入。可以看到在需求撰写和制定的阶段，测试人员就已经介入，进行需求的了解和外围黑盒测试设计。这一点与双 V 模型是类似的，测试越早介入，越容易提早发现问题。

ii.都需要经过一个需求->开发->测试的过程。

b.不同点：

i.与其他模型相比，这个过程中没有里程碑的概念(里程碑是指上一个阶段所有的产出达到标准后才能进入下一个阶段)。

在产品需求已经给出之后，开发可能已经着手进行产品的开发或者技术准备，测试也已经着手进行测试用例准备，此时出现需求考虑不全等问题，可以随时进行问题的沟通和改正。它所带来的问题是已经设计的用例可能要被改写甚至删除。

ii.没有单元测试、集成测试阶段。

传统的软件开发流程中，单元测试和集成测试有着重要的地位和作用，它可以在开发过程的前期，投入一定的成本进行相关函数和模块的测试，将缺陷在早期即被发现。

Note:可能会有同学会问，既然单元测试和集成测试这么好，为什么我们不去做？

1)单元测试和集成测试需要相应的详细设计文档、概要设计文档来进行支持，同时编写相应的测试代码才可以进行，这需要投入不少的时间和成本。

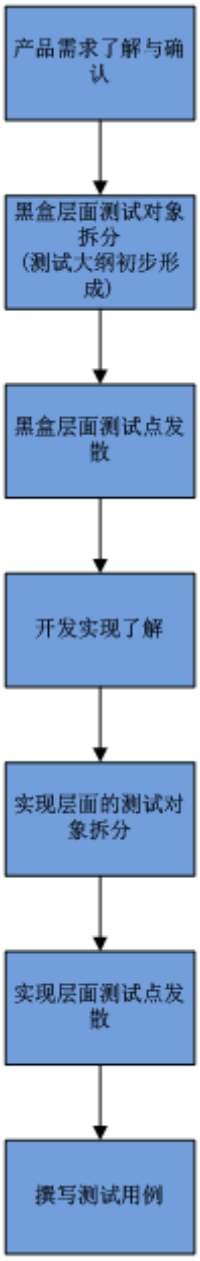
2)单元测试仅能发现功能的逻辑是否正确，但是不能发现功能的逻辑是否合理。

在互联网软件快速迭代的特性下，这个过程是无法满足要求的。

综上所述，桌面组的测试过程是一个全程参与产品需求、设计和测试的过程，同时由于互联网快速迭代的特性，决定了测试过程并不是严格遵照软件工程的开发模型而进行的。快速迭代、周期较短、受众用户广从而对产品质量有着更高的要求，决定了我们要走一条有自己特色的测试发展路线。

二、测试用例设计过程

桌面组的测试用例设计过程一般会经过这样几个过程：产品需求了解与确认、黑盒层面测试对象拆分、黑盒层面测试点发散、开发实现的了解、实现层面的测试对象拆分、实现层面测试点发散，撰写测试用例。



1. 产品需求的了解和确认:

a.测试的本质:

测试的本质是使用人工或者自动手段来运行或测试某个系统的过程，其目的在于检验它是否满足规定的需求或弄清预期结果与实际结果之间的差别。需求是测试的出发点，测试用例的设计是基于需求的，所以在测试开始之前，了解需求、搞清需求、评审需求是测试首要做的事情。

(源文档 <http://baike.baidu.com/view/16563.htm>)

b.需求确认:

作为一名光荣而自豪的测试工程师，公司赋予我们的第一项职责即是对产品需求文档进行把关。产品需求文档可能存在着各种先天不足，例如需求描述不够详细、需求考虑不够全面，需求设计不合理，甚至需求写得比较简单或者没有需求，所以我们要对需求进行确认。

i.对需求中不明确的地方进行确认:

需求文档中常见的一类问题是需求未做明确的说明，导致我们设计测试用例时不知道预期结果应该是什么样。这类不明确预期的、容易产生歧义的需求点我们都应该提出问题尽早确认，例如以下问题:

实例：我的最爱页面需求

1)需求中"不支持自定义换肤"，这个自定义换肤是指什么功能？

2)需求中"用户可根据背景搭配适合的字体颜色"，如何根据背景搭配适合的字体颜色，是程序自动搭配还是用户主动选择搭配？

3)起始页 8 宫格和 12 宫格的智能判断逻辑中，分辨率的阈值是多少？

4)……

ii.对需求中未考虑的隐性需求进行确认。

需求文档中另一类最常见的问题是需求考虑不全，隐性需求未被考虑。这类隐性需求往往不是功能表现层面能够预见到的，它属于一些不常见情况或特殊情况。

举例说明：起始页九宫格会对页面进行截图，我们往往会预见到的是截图应该长什么样，高多少，宽多少、鼠标 hover 显示什么样的效果，但是不容易预见到截图失败应该显示什么样，首次打开起始页未进行截图时显示什么样。所以我们会产生如下的问题列表:

1)起始页九宫格中的网站未截取到截图时，默认显示什么样？

2)鼠标 Hover 到起始页标题上是否有 Tooltip 提示？

3)起始页标题文字长度是否有限制？

4)切换皮肤是否需要统计 Pingback？

5)覆盖安装旧版本后，皮肤如何显示？

6)……

Note:确认需求的过程中可以参考测试知识库的需求 Review 列表(http://10.14.143.25/index.php?id=107)，帮助我们进行隐性需求的补充和确认。

iii.对需求中的合理性问题进行评审。

需求文档中同时存在的一种问题是：产品的需求设计并不一定是合理的。请记住，产品人员设计的需求不一定就是合理的需求，开发人员设计的程序框架也不一定是合理的，作为测试我们有责任指出问题，凡事都要带一个怀疑的态度。

举例说明：浏览器 2.1 版本时，产品人员当时设计的一个需求：当浏览器占用的内存较多时，弹出气泡提示用户内存占用较多，并给出按钮让用户选择清理内存。产品人员的本意是想通过弹泡告诉用户内存需要清理，让用户接受清理内存时所有页面会被重新加载(因为技术原因需要杀死所有进程并重新创建进程以达到清理内存的目的)。但是，我们仔细思考会发现需求并不合理，当用户正在浏览网页的时候，突然以弹泡的形式打扰用户，这是一种极不友好的用户体验；很多用户又都是初级用户，并不理解内存大了能怎样，不明不白地点击按钮致使页面全部被重新加载，留下的只是不解和抱怨。后来经过测试人员的推进，这个需求最终被取消了。

c.需求确认方法：

一般需求确认的方法采用邮件的形式，也可以进行口头直接沟通，最终目的是将需求确认并且公示全组。邮件形式的话，可以将需求确认邮件发送给对应的产品人员，同时抄送给对应的开发和测试组，邮件标题为"【需求确认】XXX 功能"，邮件内容为具体问题的描述，形式不限。

作为一名光荣而自豪的测试工程师，公司赋予我们的第二项职责，监督产品的需求文档的更新，务必确认沟通并确认后的需求更新至对应的需求文档中，口头沟通的需求务进行公示。

2. 黑盒层面进行测试对象拆分:

a.测试对象拆分介绍：

i.测试对象拆分是什么？(what)

1)测试对象拆分即是将一个测试对象，根据产品表现形式或者功能实现方式，划分

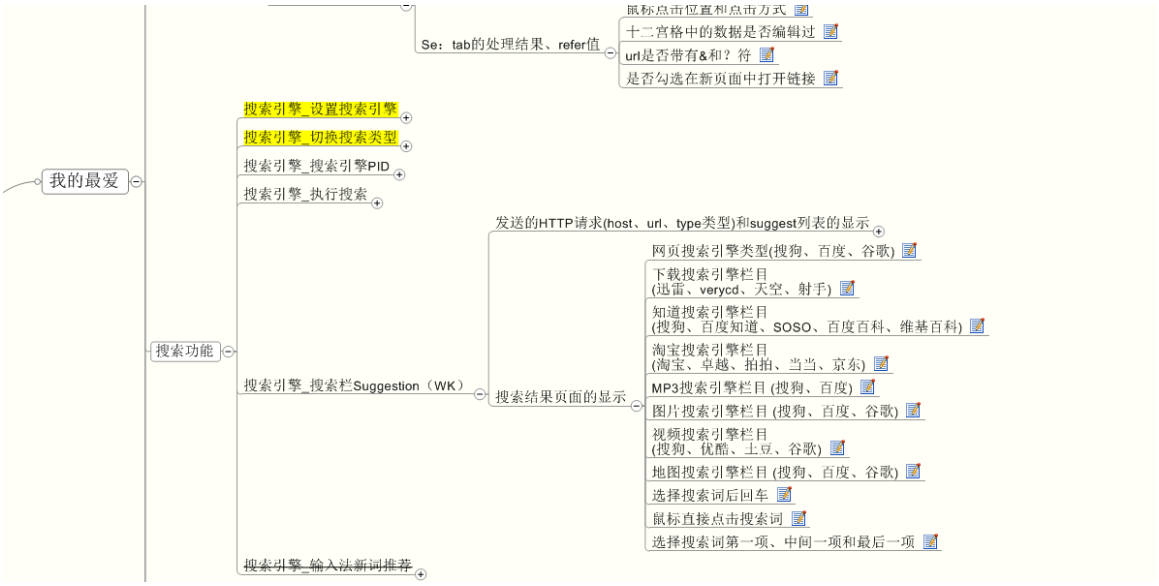
为更细粒度的测试对象的过程。

2)根据产品表现形式拆分。

举例说明：搜狗浏览器最常使用的一个功能：我的最爱起始页(不熟悉的同学可以安装搜狗浏览器后打开这个页面使用)。根据产品表现形式可以分为我的最爱、页面导航、TAB 页切换等大的功能；针对我的最爱又可以进一步划分为搜索引擎、天气预报、九宫格、最近关闭页面、换肤等功能。

3)根据功能实现方式拆分。

举例说明：我的最爱起始页中搜索引擎已经被我们拆分为一个测试对象，它还可以拆分为设置搜索引擎、切换搜索引擎类型、搜索栏 Suggestion 和执行搜索等。在搜索栏输入文字后，搜索栏会自动显示一个下拉列表，这个下拉列表可以显示输入内容相关的提示文字。例如输入"搜狗"，下拉列表会显示搜狗输入法、搜狗浏览器、搜狗地图、搜狗搜索等等提示。这个功能的实现根据后期了解，我们可以划分为发送 HTTP 请求、接受 HTTP 返回结果并解析和搜索 suggestion 列表的显示。(第 4 章我们会详细介绍为什么要进行开发实现的了解，以及为什么要进行实现层面的拆分)



ii.为什么要做测试对象拆分？(why)

1)有助于对测试对象进行测试点归类，不会造成检查点遗漏。

举例说明：我们测试一个窗口，我们会测试窗口显示是否正确。但是可能会忽略了窗口的标题文字、窗口的 icon 显示、窗口在任务栏的状态，窗口的位置.....甚至窗口获取焦点后在任务栏的闪烁等等。

2)有助于对测试对象进行测试点发散。

举例说明：如上例所说，如果我们拆分到窗口标题，那么可能会考虑切换标签页后，窗口标题的内容显示；如果我们拆分到窗口的 icon，那么可能会考虑 16*16 的 icon、256*256 的 icon，win7 系统下 icon 的显示等。

这种思考方法是从一个点联想有关系的多个点的过程。可能有同学会反问上述举例中的窗口标题、窗口 icon 都是系统处理的，我们有必要进行测试吗？如果你在 BUG 库筛选一下浏览器、输入法等产品早期的 BUG，相信你会得到答案。

iii.如何进行测试对象拆分？(how)

1)抽象和概括功能点。在需求层面对产品的功能进行概括，形成比较独立的测试对象(一级模块划分)

2)针对已经拆分的较为独立的测试对象进一步进行拆分。

测试对象拆分举例：

i.实例：天气预报扩展

ii.拆分过程:

刚拿到天气预报这个扩展，我们可能会觉得这个扩展没什么可测的，就是一个预报天气的扩展而已。那么有这么几个问题是否会进行考虑？

1)天气预报的背景是否进行了考虑？如何对天气预报背景进行考虑？

2)天气预报的天气描述文字是否进行了考虑？类似于"多云转阴"的天气描述信息最长能显示几个字符？

3)天气预报中的温度是否进行了考虑？温度最低温度能显示零下多少度？

以上看似非常简单的问题，实际测试过程都发现过问题，例如：天气描述文字最长可以为"雷阵雨转中到大雨"8 个字符，由于开发未做处理导致文字显示断行不够美观；温度显示中，由于未对零下摄氏温度的显示做考虑，导致-20℃~-10℃文字折行等。这些问题可以通过对测试对象拆分，由大到小逐层的发散和考虑，逐步达到全面覆盖。

天气预报扩展是一个比较简单的扩展，但是测试的思考方法和思路与其他复杂模块是类似的，接下来我们一步步剖析和实战拆分过程。

1) 抽象和概括功能点。

抽象和概括功能点是一个归纳和概括需求的过程。我们可能在一篇需求文档中看到许多的需求点，这些需求点往往是比较零散和细碎的，我们需要对这些需求点进行归纳，以便我们更好的理解和组织测试用例。这就好比我们整理书籍的时候要把与上课相关的书放在一个抽屉里，把课外的小说放在另外一个抽屉里一样。如果没有归纳和概括的过程，只是从头到尾一股脑地按照产品需求文档给出的需求点设计，那么会缺少一个统看全局的过程，缺少对整个功能轮廓的理解，测试用例的组织也会零散缺少条理，这样不利于我们去查遗补漏。

Note: 有一种归纳方法是这样的，我们将自己视为一名产品人员，将产品需求有着

相关性的概括为一个功能；然后重新审视一遍需求，确保各个需求点都在所概括的功能之内覆盖。

举例说明：天气预报扩展分析其需求，大致有以下几点：

a)打开天气预报窗口，显示当天气温显示；

b)打开天气预报窗口，显示未来 4 天天气；

c)天气预报可以设置不同的城市，设置城市后显示对应天气的信息；

d)点击工具栏天气预报之后，浏览器弹出气泡显示天气预报内容。

e)当显示天气预报内容之后，点击其他区域，天气预报窗口消失。

f)天气预报窗口不再显示时，工具栏仍然可以实时显示天气信息。

g)右键点击工具栏天气预报之后，选择菜单中的访问"中国天气网"，浏览器可以新开页面访问中国天气网。

上述需求中，a)、b)、c)我们很容易思考到并概括出当天气温显示、未来 4 天天气显示、城市设置三个功能；但是 d)、e)、f)、g)没有被涵盖到，而如果将以上需求点分别概括为不同功能后，那么显然没有达到我们合理组织的目的。

对此，我们分析 a)、b)、d)都是在显示天气预报窗口时才能显示的，我们可以归纳为一个天气预报窗口的显示；与窗口显示相反的联想到窗口的消失，对此我们可以归纳为一个天气预报窗口的关闭用于覆盖 e)；城市的设置与窗口显示和关闭都是不相干的，它后台所做的是城市设置的保存和读取，所以我们单独归纳为天气预报设置(c)；因为工具栏和天气预报窗口在表现层面是两个不同的对象，他们并不是相互依赖才能展示的，所以我们将工具栏的显示(f)归纳为一个单独功能；剩下 g)还没有被覆盖，所以我们单独归纳外部链接为一个单独的功能。

通过以上分析和概括，我们得到工具栏显示、天气预报窗口的显示、天气预报窗口的关闭、天气预报设置和外部链接五部分。

Note：对功能点的概括思路可以建立在功能之上，而不仅限于通过 UI 进行概括，所见即所得，例如：有些同学可能会用比较直观的方法来概括功能，"查看当前天气情况"，或者为"点击工具栏天气预报扩展"等。这类功能的概括太过具体，可能会不容易发散到其他测试点，比方说，点击工具栏天气预报提示，可能很难考虑到从工具栏的更多

菜单中打开天气预报提示等，所以我们需要对功能进行抽象和概括。

抽象和概括测试对象之后，每一个对象还可以继续细分。

举例说明：我们将天气预报的显示按照如图所示的红线部分进行归类和划分，可以划分为：城市名称的显示、日期的显示、实时天气的显示、未来 4 天天气的显示。进一步观察和 Review 会发现我们容易忽略的几个部分，如背景图的显示、Loading 加载条显示。所以，根据以上的分析，我们得到下图的拆分结果：

Note：测试发散推荐使用 MindManager，可以很清晰地看到思路展开过程。

举例说明：我们对工具栏天气的显示进行拆分，得到天气图标、地区、温度、Tooltip 提示文字等。

通过以上的分析，我们得到以下天气预报的框架图（其中天气预报的设置和外包链接未做详细拆分介绍）。

2) Review 需求

通过概括和抽象功能点之后，我们基本完成了测试大纲的建立。但是概括和抽象的过程不可避免会遗漏一些不容易考虑到的需求，所以我们还需要 Review 测试大纲。

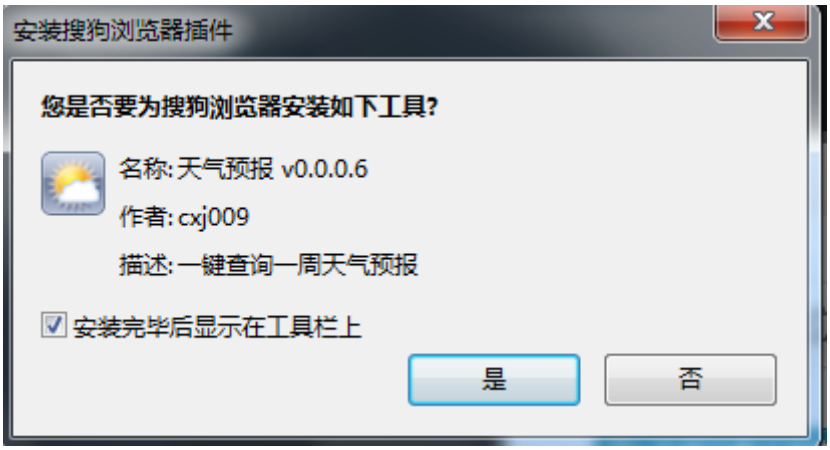
Review 的方法:

a)可以对照需求文档再过一遍，看看有没有遗漏的功能;

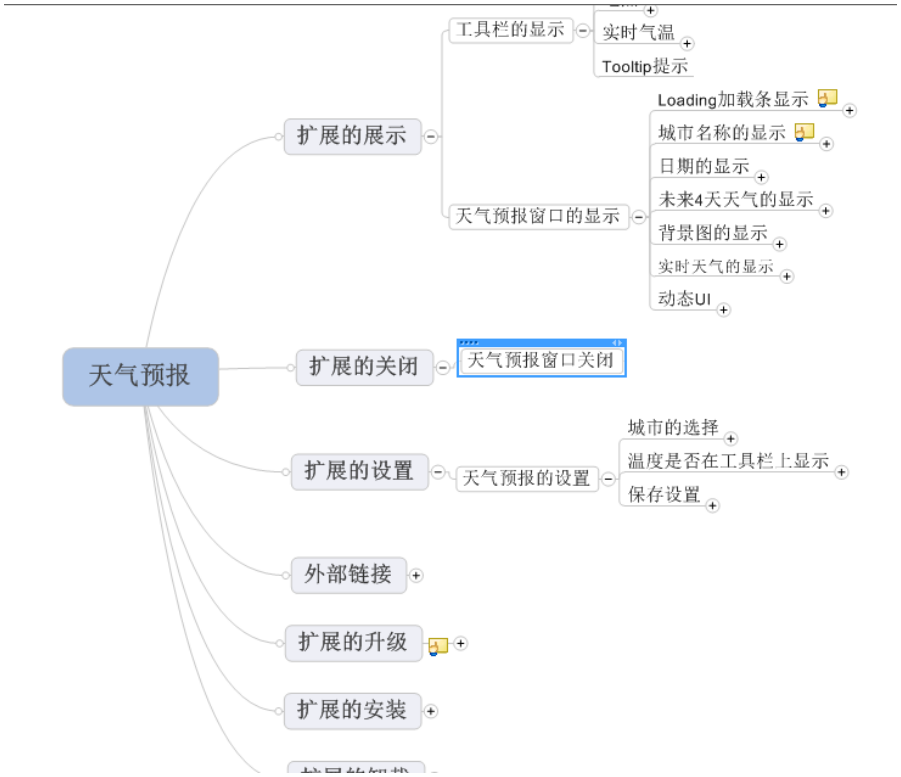
b)访问测试知识库的公共用例(搜狗内部系统)看看有没有常见的容易遗漏的需求。

思考题：如果测试设计时，经常能够在测试知识库中发现自己容易遗漏的问题，如何改进自我避免遗漏呢?

举例说明：通过 Review 需求，我们了解到天气预报扩展文件每次安装时都会弹出如下安装对话框，该对话框中的作者、描述、版本号和数字签名都需要验证。



所以，天气扩展的安装需要验证。通过安装功能进而联想到扩展的升级，所以在原有测试大纲中增加天气扩展的安装、升级和卸载，重新组织测试大纲结构得到如下内容:



拆分的越细，能够考虑和发散的测试点越多，所以可以继续拆分的考虑继续拆分下去，例如：实时天气的显示通过界面 UI 布局，拆分为实时温度、湿度、风力、天气描述、温度范围等，那么实时天气的显示进一步被拆分为对应的测试对象。

拆分的原则可以灵活掌握，如果继续细分下去对于测试发散度没有太大帮助的话，可以不继续拆分。

Note: 拆分粒度的建议---属于 UI 层面的测试对象，建议拆分的粒度到最小的元素，例如一个按钮、一段文字、一个图标、一个单选框控件等。属于功能实现层面的测试对象，建议拆分的粒度到一个不可继续拆分的功能逻辑，例如写文件、写注册表，再往下细分是具体的接口函数了，这些函数大多是已封装好的系统函数或第三方函数，可以不用拆分。

3) 多维度影响的考虑

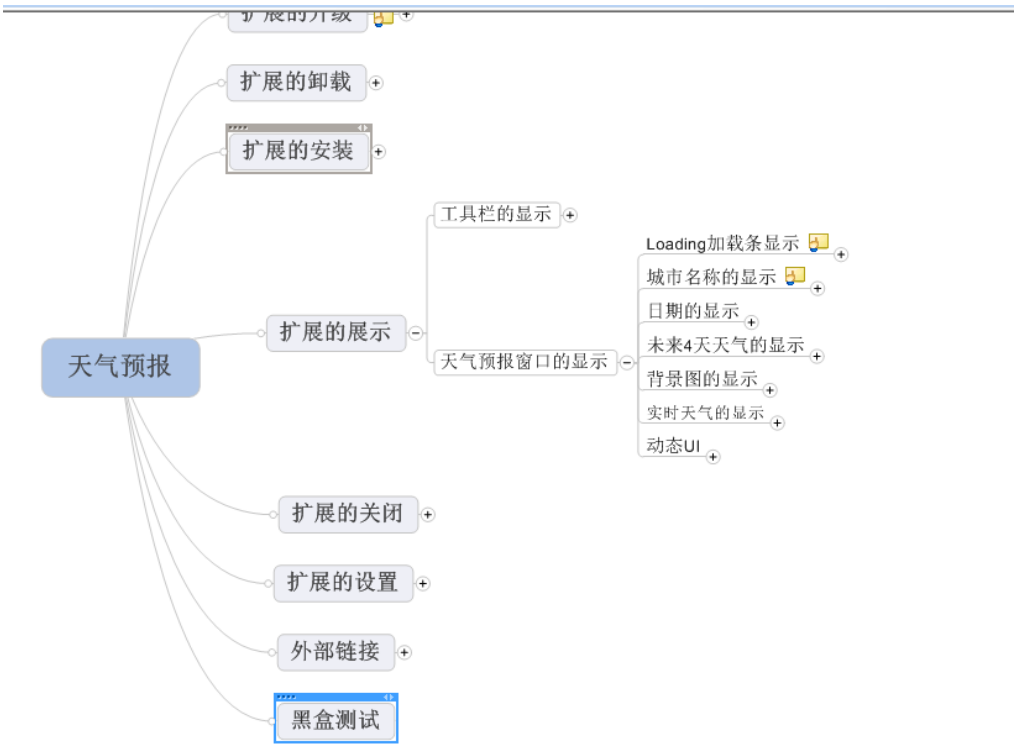
由于我们尽可能的对一个测试对象进行细化和拆分，针对每一个小的测试对象进行测试和发散，但是这也带来一个问题：将各个功能组合起来的集成测试(黑盒测试理论叫系统测试)不够，例如：

- 覆盖安装为旧的浏览器版本，天气预报扩展能否使用？
- 登录通行证账户后，同步通行证数据，自动下载并加载天气预报扩展后，天气预报的地理位置是否与当前一致？
- 在教育网环境下，开启全网加速功能后，天气预报的实时天气能否正常显示？

通过以上问题，我们看到功能与功能之间相互作用的影响，用户实际使用的复杂场景也是我们设计中需要考虑的。在此我们通过一个事例来说明相关项考虑不够导致上线事故的问题：

浏览器 2.0 版本对收藏功能进行了重构，我们整个测试过程中对于收藏的添加、删除、拖拽、编辑、显示、重命名等各方面进行了测试，但是忽略了这样一条路径：用户安装 1.4 版本浏览器后，覆盖安装到 2.0 版本浏览器。在发现 2.0 版本是试用版之后，又重新覆盖安装到 1.4 较为稳定的版本使用，结果导致收藏丢失。丢失的原因是由于数据库格式被升级后，旧版本浏览器无法识别。

为了解决类似以上的问题，我们在测试大纲中增加一项"黑盒测试"，在这个黑盒测试中，可以着重考虑功能之间相互影响的测试、用户使用场景下的测试：



至此，一个功能黑盒层面的测试大纲拆分过程结束了。

3. 黑盒层面进行用例发散：

a.黑盒层面的用例发散：

在上面文章中，我们介绍了测试大纲的建立和拆分过程，这个过程帮助我们梳理和整理了待测试功能在大方向上的思路和测试范围，我们可以把测试大纲比作"骨架"；接下来，我们要在"骨架"的基础上丰富内容，形成饱满的内容。

为了保证功能和功能之间的测试被覆盖到，我们可以首先在黑盒测试这里进行用例发散。用例的发散主要考虑别的功能模块对当前模块有什么影响，以及当前模块对别的模块有什么影响，顺着这个思路进行用例相关的发散。例如：

i.常见的我们会考虑覆盖安装对当前模块的影响；

ii.全屏模式对当前模块的影响；

iii.切换标签页对当前模式的影响；

以上思路主要是针对当前模块，考虑哪些模块与当前模块有关联，再加上平时测试

积累的测试经验进行猜测和发散。

另外一种发散方法可以考虑用户使用的场景进行发散，例如:

i.当前用户经常在网吧上网，那么通过通行证进行数据同步是其经常使用的场景。我们可以考虑当用户在网吧电脑上首次安装搜狗浏览器，登录通行证之后，使用天气扩展插件，测试天气扩展可以自动同步并使用等。

ii.当前用户为老年人一族，上网目的主要是查看新闻或者查看天气预报。该类型用户的特点是字体比较大，系统中 DPI 设置为最大的 120，页面缩放比例也是 120%以上，那么这种情况下我们会考虑系统分辨率在 1024*768 分辨率下天气扩展的显示正常，没有错乱。

以上的思路是根据用户使用场景进行思考和发散。

b.根据拆分的测试对象进行用例发散:

在黑盒层面的发散补充之后，我们将在上例中拆分的每一个模块进行逐级的用例发散。

举例说明：天气预报窗口的显示，它下面包含了 loading 加载条的显示、城市名称的显示、日期的显示等，我们针对每一个模块进行展开考虑。

i.Loading 加载条的显示。针对这个模块，我们思考什么情况下会显示 Loading 的加载条，什么测试点会影响到加载条。

我们可能会考虑到的是:

1)第一次打开天气扩展时，需要考虑天气扩展读取文件过程显示 Loading 加载条。

2)第二次或者多次反复打开天气扩展，Loading 加载条的显示。(较好的用户体验是不再显示 Loading 加载条)

3)更换城市时，读取城市天气的过程需要 Loading 加载条进行场景切换。

4)天气信息已经加载完毕后，验证 Loading 加载条消失。

……

ii.城市名称的显示。针对城市名称，我们同样思考什么情况会影响到城市名称的显示，或者我们可以使用黑盒用例常见的"边界值"进行发散:

1)长度为 2 个字、3 个字、最长为 7 个字的。

2)当前系统默认字体为微软雅黑等其他字体后，可能会影响到城市的显示，需要进行考虑。

依次类推，日期的显示、天气的显示、温度的显示、天气说明的显示均可以参考"等价类"、"边界值"的方法进行考虑。

iii.天气预报的窗口。除了上述对于天气预报窗口内的文字、图标、背景的测试外，我们还需要考虑天气预报窗口的测试:

1)什么情况会影响到窗口的显示。

2)什么情况会影响到窗口的位置。

3)什么情况会影响到窗口的消失。

根据测试经验和之前总结的公共用例库，我们可以发散到如下的测试点:

影响窗口显示的影响因素可能有:

a)入口：包括从工具栏、从扩展插件盒子，从菜单栏更多菜单、通过右键菜单打开天气预报扩展。

b)当前系统的分辨率，可能影响

c)当前浏览器窗口的大小(最大化、全屏、非最大化)

d)当前浏览器窗口宽度小于天气预报窗口宽度

……

影响窗口位置的影响因素可能有:

a)当前浏览器窗口在屏幕的位置(左侧、右侧、上方、下方)

b)当前系统为双屏浏览器

c)存在置顶窗口时，天气扩展窗口能显示在置顶窗口下方。

……

Note: 进行测试发散时，不要忽略了数据来源的正确性。例如：在天气扩展中测试日期显示的时候，我们会测试日期的文字显示正确，没有错行；日期的显示位置居中等

等，但是不要忘记验证当天到底是星期几！

经过测试发散，我们在"骨架"上补充了"血肉"，这些"血肉"就是一个个的测试点：

4. 开发实现了解：

a.我们为什么要了解开发实现？

这是一个新来同学经常问到的问题，我们得从黑盒测试和白盒测试的作用进行解释。黑盒测试是一个从需求层面、用户使用层面进行功能测试的过程，测试的目的是验证产品实现与需求是一致，而程序的执行过程和实现方法对于测试人员是不可见的，所以我们称之为黑盒测试。白盒测试是一个从代码层面，针对代码逻辑实现进行测试的过程，测试的目的是验证函数的输出结果与开发设计文档是一致的。

是不是仅仅做好黑盒测试就可以保证产品质量没有任何问题了呢？答案是否定的。

互联网产品的特点是什么？快速迭代，开发和测试周期比较短。

如何能够做到在尽可能快速迭代、周期比较短的情况下，保证产品的质量得到最大限度的覆盖和保证呢？经过桌面组多年的实际项目经验，我们归纳总结出一套方法：除了进行黑盒层面的测试之外，对于复杂的功能模块进行实现层面了解，补充实现层面的可以测试的测试点，尽可能在测试广度上和测试深度上进行覆盖，我们称之为灰盒测

试。

与传统意义上的测试不同的是：与黑盒测试相比，要测得多，测得深；与白盒测试相比，不需要花费较大的精力写专门的测试代码，而是由了解实现发散实现层面的测试点，然后通过外围黑盒测试来构造，比白盒测试要快。

举例说明：浏览器的升级功能

升级功能是一个常见的软件功能，输入法要进行程序的升级，浏览器也要进行程序的升级。那么升级功能我们不了解实现时考虑到的测试点有哪些？

1) 未了解实现的测试点可能有：

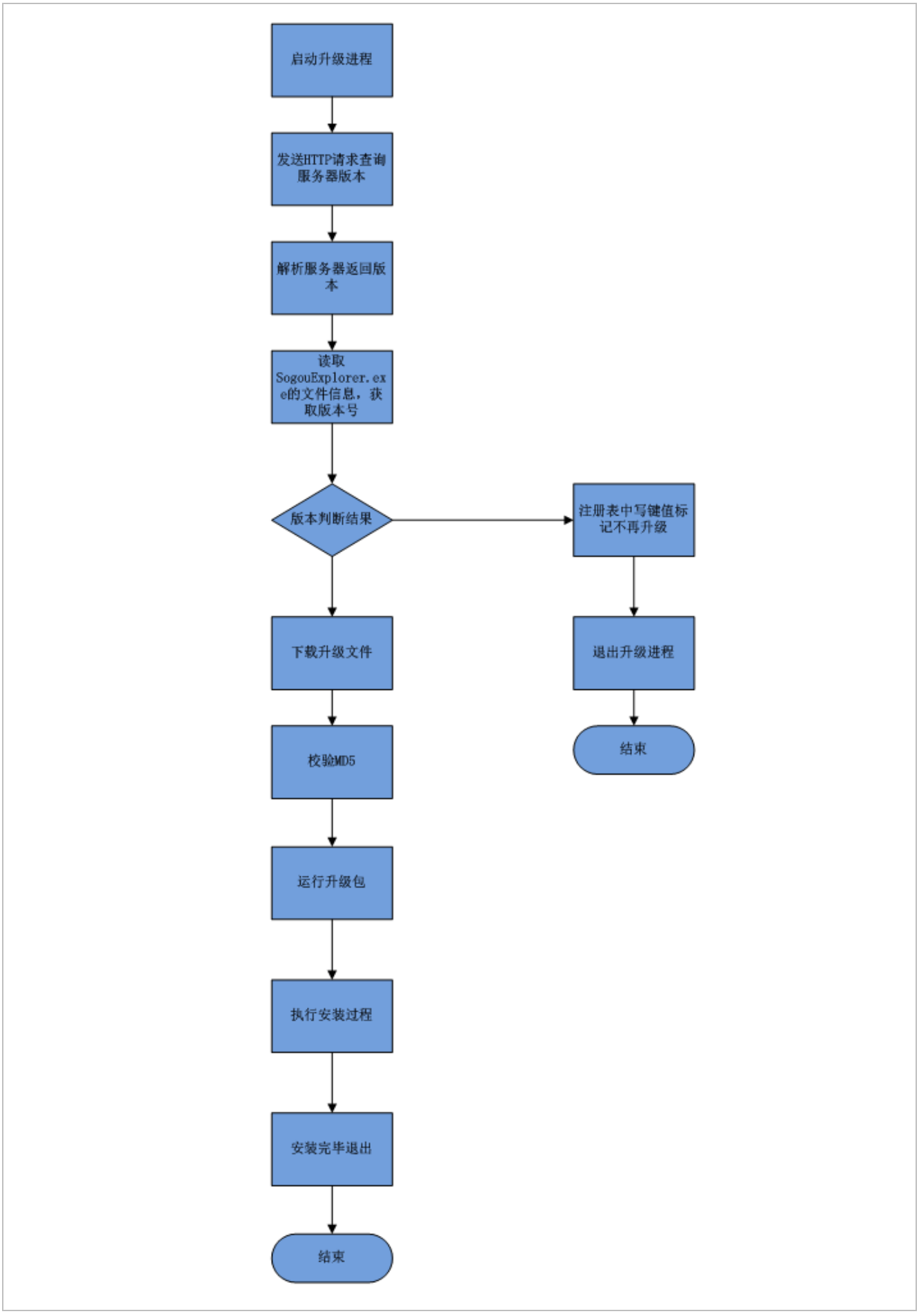
已安装旧的版本情况下，执行升级，验证可以升级到新版本

升级时的 UI 显示

升级时的各类操作(取消、关闭)

升级后程序原有功能正常

2) 开发实现的了解：


启动升级进程
发送HTTP请求查询服务器版本
解析服务器返回版本
读取SogouExplorer.exe的文件信息，获取版本号
版本判断结果
注册表中写键值标记不再升级
退出升级进程
结束
下载升级文件
校验MD5
运行升级包
执行安装过程
安装完毕退出
结束

3) 新增加的测试点:

经过开发实现的了解，我们知道功能需要读取注册表键值判断升级是否启动、需要发送网络请求查询版本信息、需要进行 MD5 计算校验文件完整性。所以我们会联想到如下的测试点:

启动升级程序:

当天首次启动浏览器，验证升级程序运行

非当天首次启动浏览器，验证升级程序不会运行

升级程序所在路径是中文路径或者带有空格路径，验证升级程序能够启动运行

验证升级程序带有数字签名，不会被杀毒软件和防火墙拦截

……

发送 HTTP 查询请求:

验证发送的请求带有当前浏览器的版本号、渠道号和机器名，以便可以控制待升级的版本和渠道

……

版本比较:

服务器没有返回信息时，验证升级程序不会崩溃

服务器返回 502/404/403 等异常信息时，验证升级程序不会崩溃

DNS 无法解析情况下，验证升级程序不会崩溃

服务器返回的版本号大于当前版本

服务器返回的版本号等于当前版本

服务器返回的版本号小于当前版本

……

下载升级文件:

下载过程中断网，验证升级程序不会卡死、崩溃，同时可以重试

下载文件的下载路径为中文路径或带有空格路径

……

运行安装包:

当前账户为受限账户，验证安装包以管理员身份运行安装

当前浏览器正在运行时运行安装包，验证已有浏览器会被关闭

……

b.了解开发实现测试设计的弊端是什么?

上面我们讨论了为什么要了解开发实现，这有利于我们挖掘更多的测试点。但是了解开发实现的弊端是什么？了解开发实现仅能够帮助我们了解一个函数或者一个功能是如何一步一步运行的，但是不能帮助我们辨别这种过程是否合理。所以，请谨记我们了解实现的目的是挖掘更多的测试点，但是不要忘了或者忽略外围黑盒的测试，要同时做到测试的广度(黑盒)和测试的深度(灰盒)。

c.是不是所有的功能都要了解开发实现?

了解实现的目的是帮助我们挖掘更多的测试点，但是有些模块在了解实现后并不能对测试发散度有所帮助，所以并不是所有的功能都有必要挖掘到开发实现，比如弹泡提示、Tooltip 显示、菜单功能等这些简单的功能；对于有些复杂的功能，例如通行证同步、皮肤系统等复杂模块，可以考虑开发实现了解。

d.了解实现的几种方式:

i.开发给出相应的设计文档。

由于项目进度压力，开发人员能力高低、模块的复杂程度等因素的影响下，有些开发人员可以给出详细的设计文档，有些开发可能连指导文档都是没有的。对于能够给出设计文档的开发人员，我们鼓掌欢迎；对于不能够给出设计文档的，也请同学们互相体谅。

ii.直接与开发进行沟通。

作为一名光荣而自豪的测试工程师，公司赋予我们的另一项职责即是与开发人员进行实现层面的沟通。请记住我们了解实现的目的是更好地测试一个产品。

思考题：如何与开发进行开发实现层面的沟通?

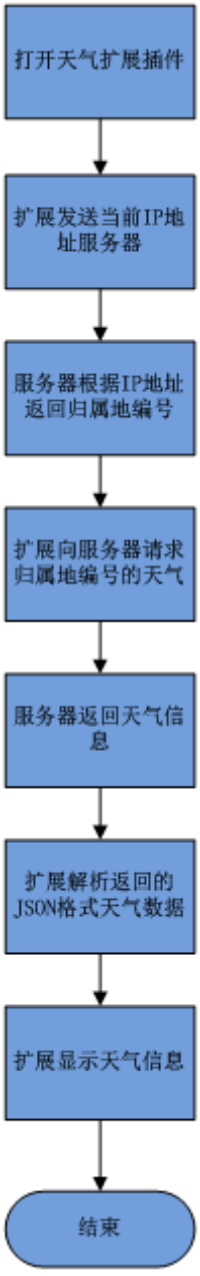
iii.由有经验的测试人员进行代码调研，给出调研结果。

iv.自己阅读代码了解实现。

5. 实现层面测试对象拆分

a.根据实现画出流程图:

举例说明：经过实现层面的了解沟通，我们在实现层面了解到实时天气信息的获取过程是这样实现的:



Note:绘制流程图是一个了解实现很好的习惯。它不但有助于我们系统全面地了解实现过程，同时还便于我们根据流程图拆分实现层面测试对象。

b.归纳实现过程:

为了便于梳理和组织测试用例，同时也为了便于我们查缺补漏，我们仍然需要对实现过程进行归纳和概括。在画出流程图之后，流程图中每一步都可以看做是一个子功能，都可以作为一个测试对象。

举例说明：服务器根据 IP 地址返回归属地这个逻辑是在服务器上运行的，而我们的测试对象是客户端的天气扩展，我们测试的重点是天气扩展在接收到服务器不同数据情况下能够正常处理并显示，而服务器的功能逻辑不在我们的测试范围内。因此，我们把服务器上的功能逻辑忽略，而客户端在解析服务器不同数据最终会作用在天气信息的显示上，所以将天气信息的解析和显示看做是一个过程，解析结果影响着显示，解析最终的结果反映在显示上。

1).归属地查询请求过程:

扩展向服务器发送归属地查询请求，如下图 61.4.185.48 地址请求信息，发送的请求会携带当前机器的 IP 地址。

2).归属地结果解析过程:

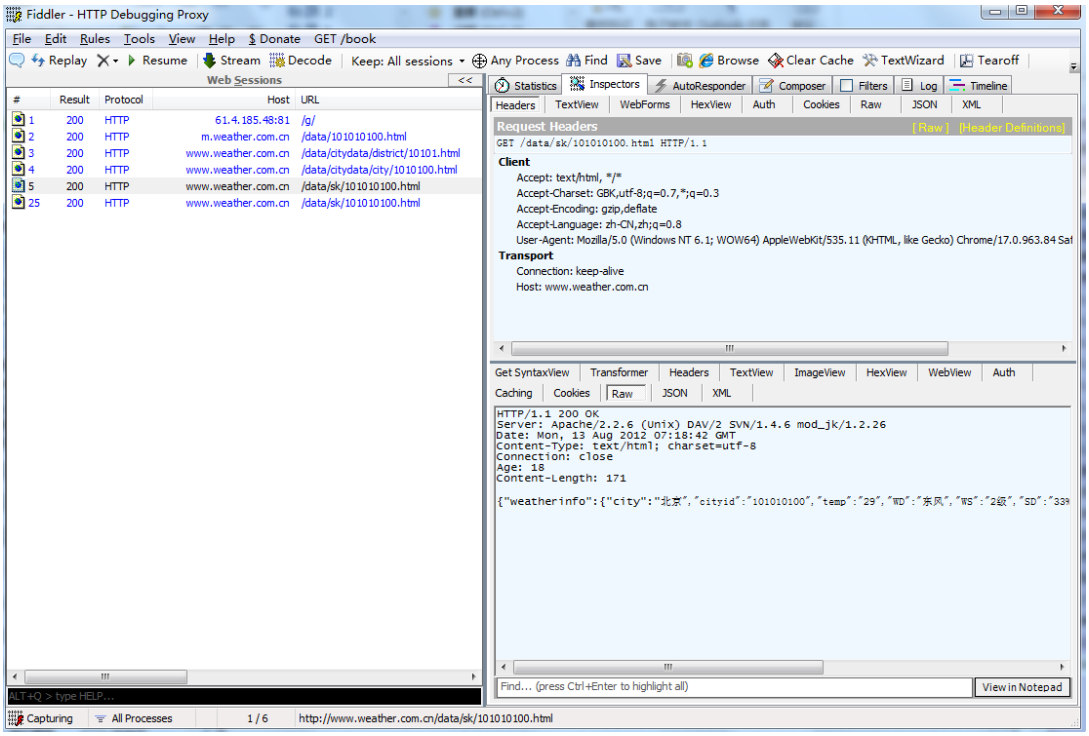
服务器收到查询请求之后，根据 IP 地址查询到归属地返回城市编号。

3).天气信息请求过程:

扩展根据返回的归属地编号向 www.weather.com.cn 发送请求，获取天气信息，如下图所示第 5 条。

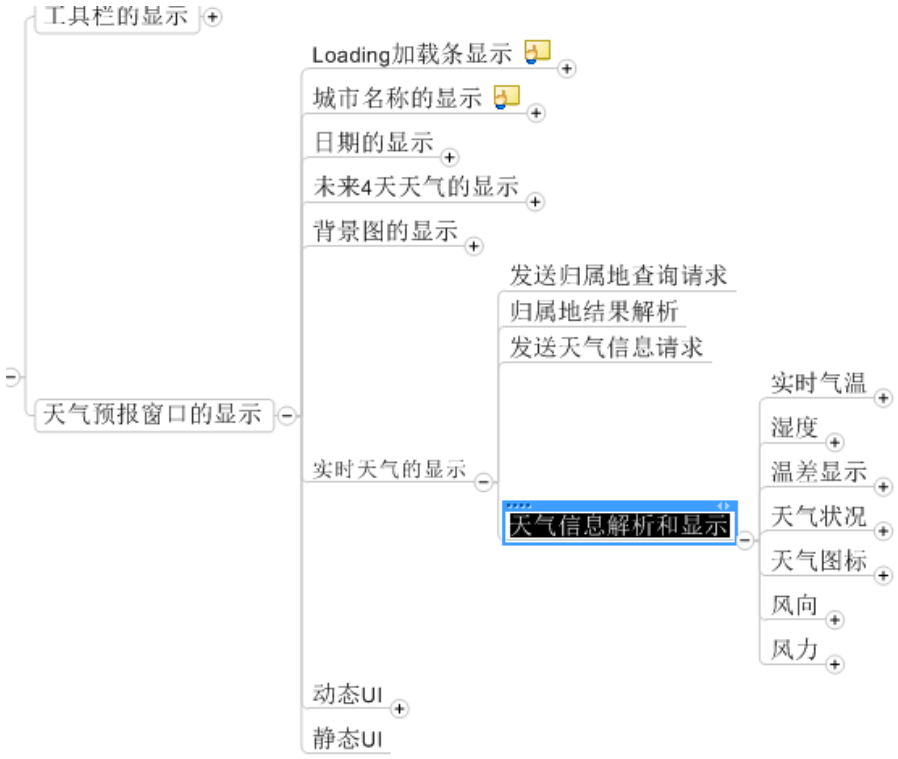
4).天气信息解析和显示过程:

服务器将天气信息以 JSON 格式返回给扩展，扩展收到信息后解析 JSON 数据，分别获取到城市名称、当前温度、当天温度、风力、湿度、日期、未来 7 日信息、天气描述信息等。

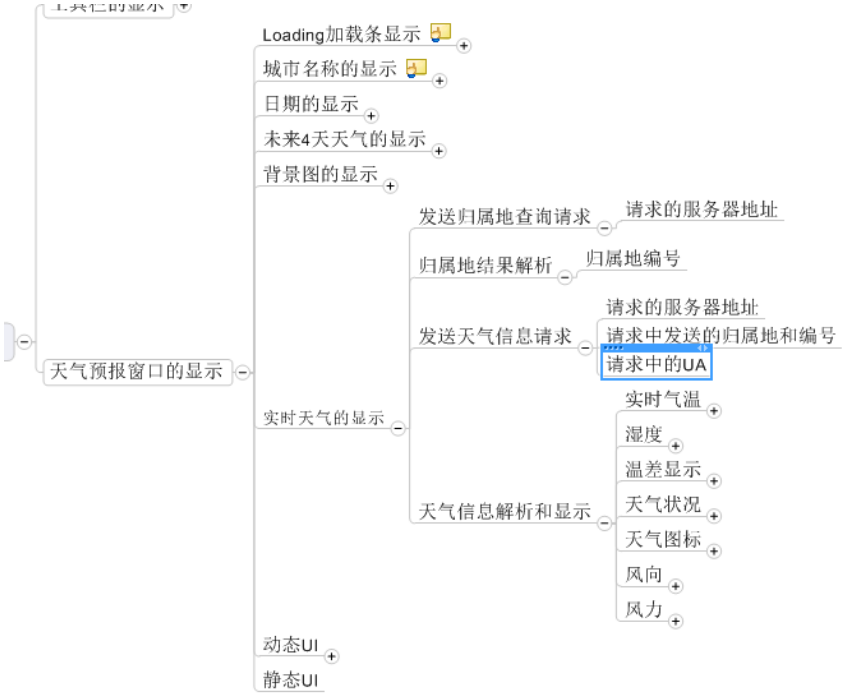


c.更新测试大纲:

根据以上的总结，我们将测试大纲中实时天气的显示进一步拆分为发送归属地查询请求、归属地结果解析、发送天气信息请求、天气信息解析和显示几个部分，将原有已拆分的实时气温、湿度、温差显示等划分在天气信息解析和显示下，如图:



发送归属地查询请求又可以细分为：请求的 URL、请求中的参数(例如是否有 H 机器码参数、V 版本号参数等)。根据这样的思路，我们细分为：



Note：对比一下，经过这样的拆分有什么好处？

i.我们会考虑一些容易疏漏的测试点。

比如，我们要测试每次打开扩展至少都会发送重新获取归属地的请求，我们要测试请求的服务器地址是正确的，我们要测试请求中的参数是正确的(对于 pingback 测试，尤其要关注 hrv 和自定义字段的发送是正确的)。

ii.我们会自然而然考虑上一个过程对下一个过程的影响。

例如：天气信息返回的内容为空对显示的影响；天气信息返回超时对显示和扩展的影响；

iii.我们会考虑一些比较深的测试点。

例如：返回的天气信息不是 JSON 格式对显示的影响；返回的数据中包含 JSON 的敏感字符，对显示的影响等。

6. 实现层面测试点发散

a.发散方法：

发散方法可以从上一个过程对下一个过程的影响、深层次的实现对外围功能的影响

进行发散考虑。在这个事例中，实现层面我们看到了两个新的测试对象，HTTP 请求的发送与接收，JSON 数据的解析与显示。我们可以借鉴之前的测试公共用例库进行发散补充，比如 HTTP 常见的返回码(502\404\403\401 等)、HTTP 常见的网络异常(返回超时、返回内容为空、返回内容不正确等)。再比如 JSON 数据常见的测试点(返回的内容不是 JSON 格式、返回的字段中包含 JSON 特殊字符、返回的 JSON 数据为空等)

Note: 实现层面的发散方法与黑盒层面是类似的，需要使用常见的用例设计方法等价类、边界值等，或者通过已总结的测试公共用例库、或者通过积累大量的测试经验。

7. 撰写具体测试点的测试用例

a.撰写测试用例的过程:

经过以上的步骤，对测试大纲抽象、概括、整理和发散，一个待测试功能的思路已经基本成型，测试设计工作基本已经完成了 2/3，后续所要做的事只是针对每一个测试点去详细写一条用例的内容，如前提条件、操作步骤、预期结果等。

Note:推荐仍然在 mind manager 中写测试用例，在每一个条目下按下 ctrl+t，在右侧的文字输入栏粘贴用例模板，然后写每一个字段内容即可。用例全部写完之后，用 mind manager 的导出功能，将测试用例导出为一份 word 文档即可。

Appium+Robotframework 实现手机应用的自动化测试-Android 篇

◆ 作者：王 东

摘要： 本人主要介绍了如何使用 Robotframework 结合 Appium 实现对 Android 应用的自动化测试，从概念介绍，如何安装和部署到示例脚本一应具有，相信读者通过本文可基本掌握其用法。

一、Appium 简介

使用 Appium 已经有一段时间了，我是结合之前用了很久的 Robotframework 来写 Appium 的自动化脚本，对 Android 和 iOS 的原生应用都已经实现了自动化测试，现把自己的一些经验和心得分享出来，由于本人水平有限，难免有不足和错误之处，欢迎读者指正。

以下的介绍主要翻译自 Appium 官网，我想这才能最接近 Appium 创建者的本意吧。

Appium 是一个开源，跨平台的自动化测试工具，它支持原生的，混合的和移动 web App，可以在 iOS,Androidhe 和 FireFoxOS 的模拟器以及它们的真机中进行测试。它**支持的操作系统有** iOS，Android 和 FirefoxOS。

为什么用 Appium?

1. 由于在所有的平台中使用了标准的自动化 API,所以不需要为了自动化而且重新编译或修改 App。

2. 可以使用自己最熟悉最喜欢的语言,比如 Java, Objective-C, JavaScript with Node.js, PHP, Python, Ruby, C#, Clojure 或者 Perl 结合 WebDriver API 和其语言特定的客户端库以及工具来写 Appium。

3. 可以使用任何测试框架。

当使用 Appium 时，实际上意味着在利用唯一的，免费的和开源的已经成为事实上的标准的 WebDriver 协议。不要把自己封闭起来。

如果使用苹果的 UIAutomation 库，就只能使用 JavaScript 编写测试，并只能通过 Instruments 来运行测试。同样的，使用谷歌的 UiAutomator，只能使用 Java 来编写测试。Appium 是最大程度上的真正的跨平台的原生移动自动化框架。

如果你是一个 Appium 新手，或者想要以上内容的完整描述，请阅读 Introduction to Appium Concepts。

二、安装条件

需要建立设定的移动平台用于运行测试。请看以下的平台要求:

如果你想通过 npm install 运行 Appium,对 Appium 有所贡献(因为它是开源的哦)，需要 node.js 和 npm 0.10 或更高版本(使用 n 或 brew install node 安装 Node.js,确保没有使用 sudo 来安装 Node 或者 Appium,否则就会遇到麻烦)。建议使用最新的稳定版本。

可以使用 appium-doctor 来检查是否所有的 Appium 依赖项都没有问题，运行 appium-doctor 并提供参数 --ios 或者--android 来检查所有的依赖项是否正确安装。如果从源代码处运行，则要使用 ./bin/appium-doctor.js 或者 node bin/appium-doctor.js.

请下载你熟悉的语言的 Appium 客户端用于编写测试。Appium 客户端是 WebDriver 客户端的简单扩展。可以在 Appium 客户端列表看到客户端列表及其下载链接说明。

Android 安装条件:

Android SDK API >=17(附加的功能需要 18/19),本人当前使用的是 18.

因为 Appium 支持在 OS X,Linux 和 Windows 中运行 Android 测试，所以请确保你使用的操作系统满足安装条件，本人为了方便就直接安装在了 Windows 7 中，以下是这三种操作系统所需的安装条件，请点击进行了解。

Linux

OS X

Windows

FirefoxOS 安装条件:

Appium 可真强大，这个操作系统也支持，不过本人还没用过，请点击右侧的链接进行了解。Firefox OS Simulator

三、Appium 在 Windows 中的具体安装步骤

由于前两篇文章主要是翻译自 Appium 官网，由于本人英文不是很好，所以还请读者见谅，也感谢大家的厚爱和支持。

好了，让我们开始在 Windows 中开始安装 Appium 吧。

官网上说先要装 Node.js，还要装 Apache Ant 和 Apache Maven,Git 以及 cURL，不过我的经验是这些不是必须的，可以不装，当以后需要时再装也不迟，这样一开始安装比较容易和上手。

废话少说，直接开始安装步骤:

1.安装 android 的 sdk 包，(http://developer.android.com/sdk/index.html), 运行依赖 sdk 中的'android'工具。并确保你安装了 Level17 或以上的版本 api，建议至少安装到 19，我安装到了 22。

2.设置 ANDROID_HOME 系统变量为你的 Android SDK 路径，并把 tools platform-tools 两个目录加入到系统的 Path 路径里。因为这里面包含有一些执行命令。

3.安装 java 的 JDK，并设置 JAVA_HOME 变量为你的 JDK 目录。

4.安装 Appium for Windows 版：下载路径 http://appium.io/downloads.html，我用的是最新的 1.4 版本，很好。

5.安装好 Python 2.7 版本，虽然 Appium 支持很多语言，但个人最偏爱 Python，还有一个原因是 RobotFramework 也支持 Python，接下来我是用 RobotFramework 来写 Appium 的哦。

6.安装 Appium Client 中的 Python 支持包，打开命令行，输入：pip install Appium-Python-Client，也可以直接下载 python-client-master.zip，然后将其解压缩，打开命令行，先切到解压缩所在的路径，之后输入：python setup.py install 来完成安装。

如果都没有报错，那恭喜你，成功了！如果遇到问题，请仔细查看错误提示，一般都能解决。

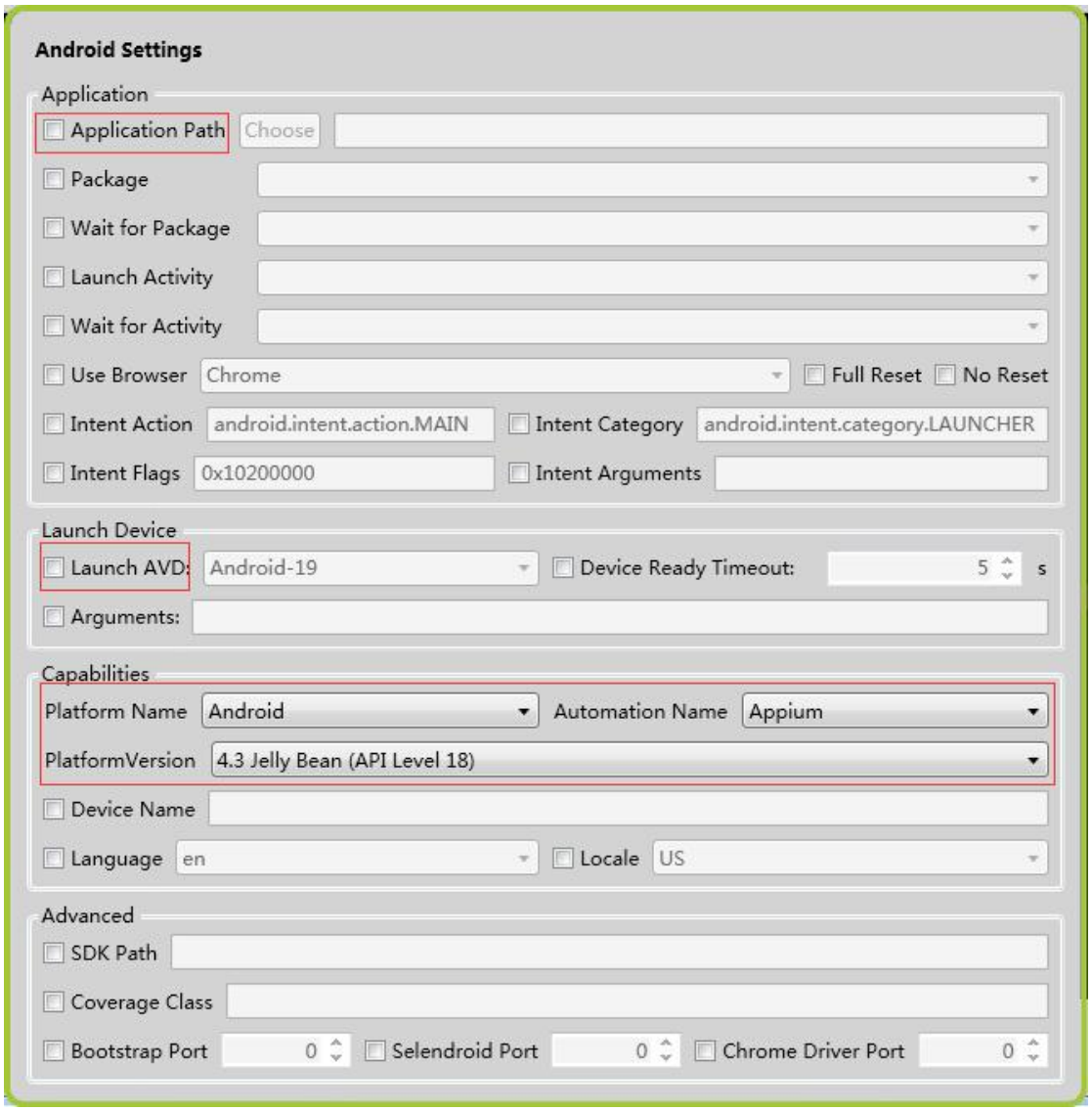
四、Windows 中启动 Appium 和模拟器

4.1.启动 Appium

安装好了之后，在桌面或者菜单中找到 Appium，分别双击或点击打开 Appium.exe，如果一切正常，接着会出现一个 Appium 启动后的界面窗口，如下图所示。

4.1.1 Android Settings

点击左上角的第一个机器人图标，弹出 Android Settings 窗口，如下图所示。



在该窗口可以对将要进行测试的 Android 应用进行设置，因为接下来我们写脚本来跑测试，所以可以统统不进行设置，保持其默认值即可。如果不在脚本中设置，则需要在这里设置下，不过大部分保持默认值即可，需要进行设置主要有 5 个:

1.Application Path: 点击 Choose 按钮后会弹出选择 Android 应用的路径选择框，选择需要测试的应用即可。

2.Launch AVD: 如果有多个模拟器，这里选择一个作为测试用的模拟器。

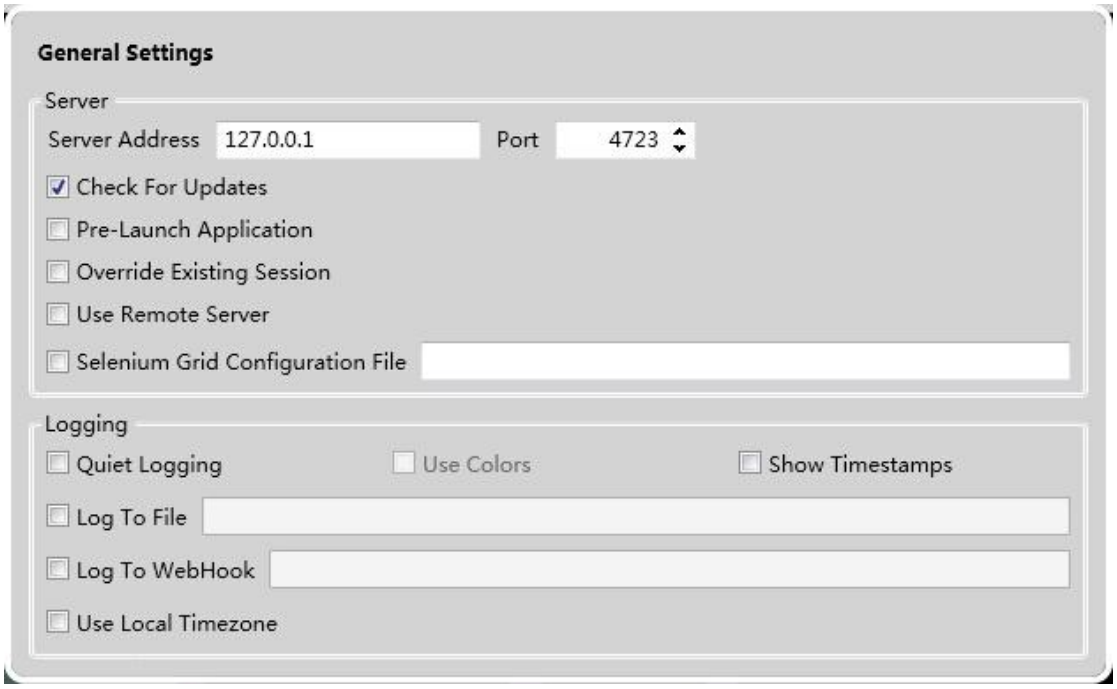
3.Platform Name:这个当然选择 Android，因为现在是在说 Android 的自动化测试。

4.Automation Name:当然选择 Appium，我们不是在玩 Appium 吗？

5.PlatformVersion:这个当然选择和模拟器中一样的版本啦。

4.1.2 General Settings

点击 Appium 左上角的第二个齿轮图标则弹出 General Settings 窗口，如下图所示。



如果就在本机安装了 Appium，则都可以保持默认值，如果把 Appium 安装在了其他机器，则需要设置 Server Address，值就是 Appium 所在机器的 IP 地址，端口号一般无需更改。

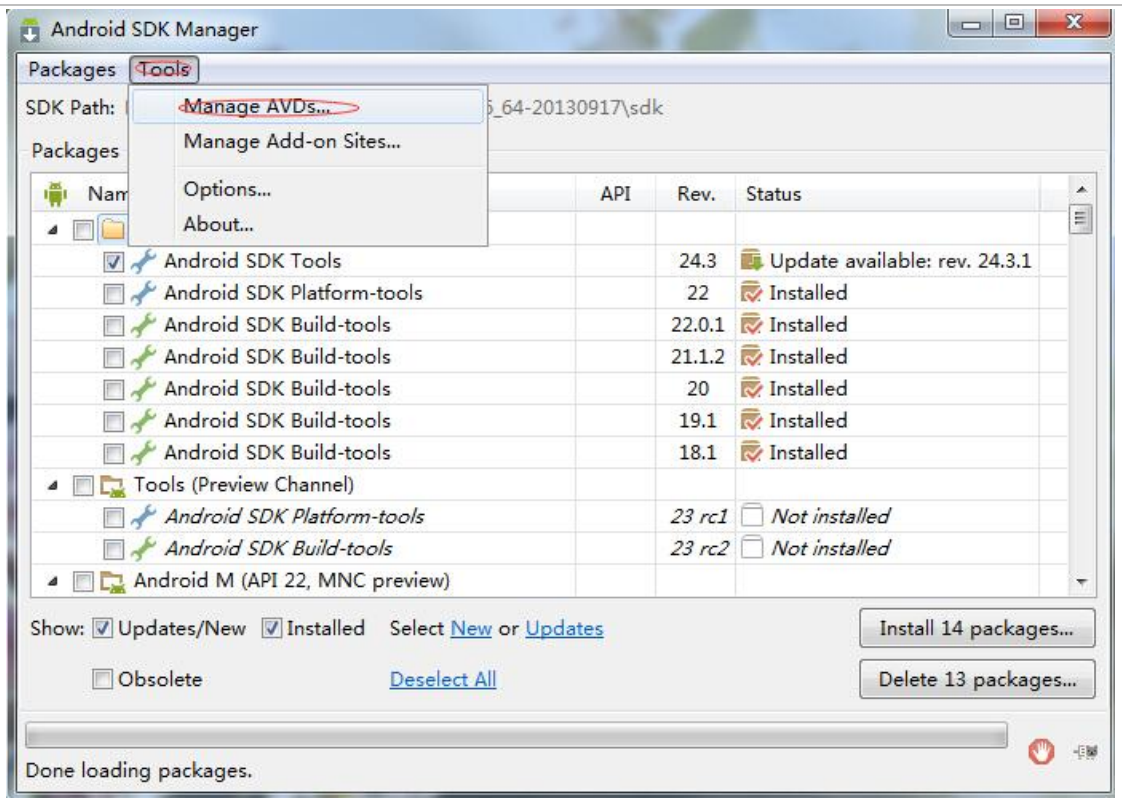
4.1.3 启动 Appium

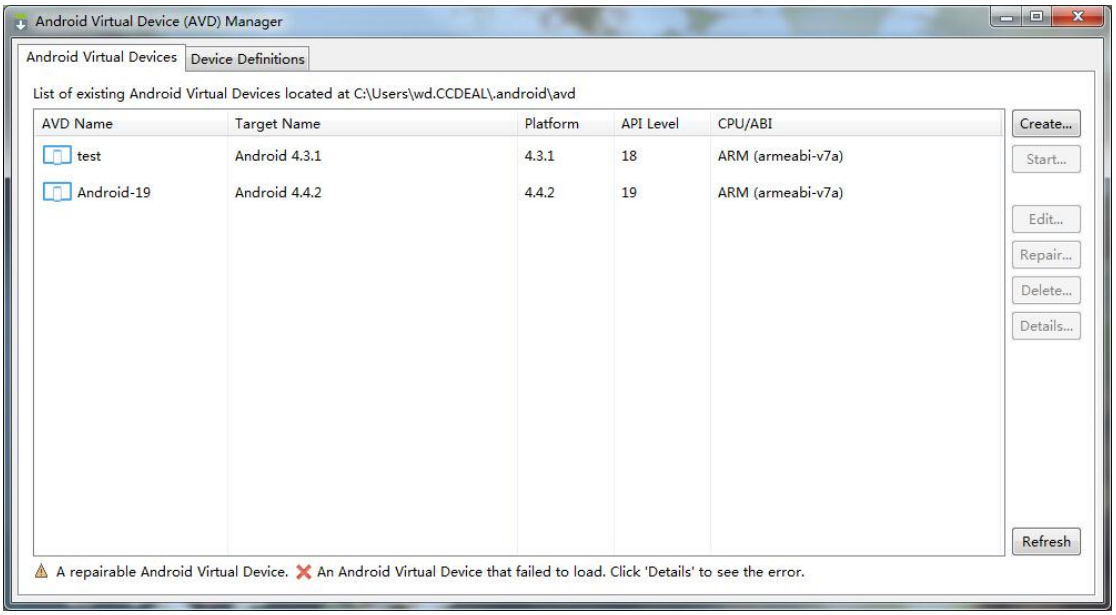
这次直接跑 Appium，所以进行相关的设置，点击 Appium 右上角的正方形图标启动 Appium，如果能看到如下图所示的信息就表示启动成功了。

4.2 启动模拟器

4.2.1 启动虚拟设备管理器

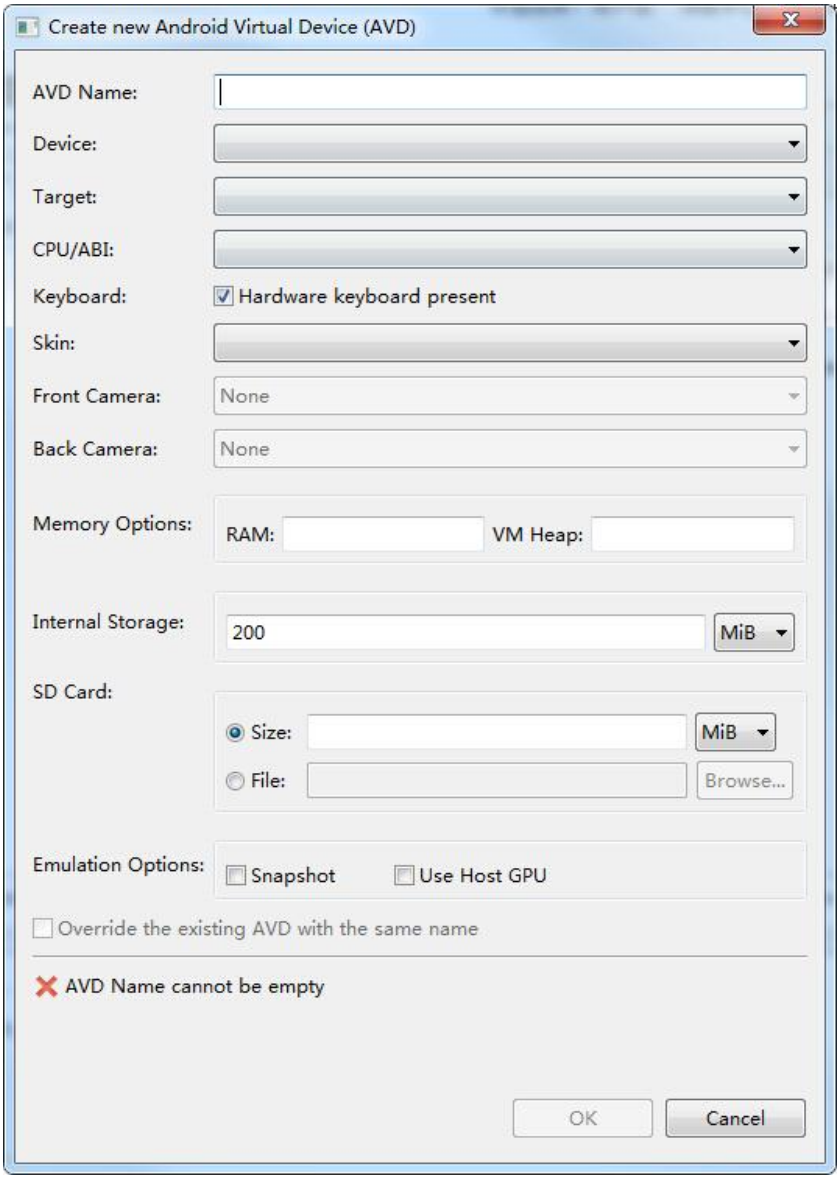
进入到 ANDROID_HOME\sdk 目录中，然后双击 AVD Manager.exe 文件，如果没有该文件，则可双击 SDK Manager.exe 文件，然后在打开的窗口中点击菜单 Tools，接着点击其 Manage AVD…子菜单项，这样就打开了 Android Virtual Device(AVD) Manager，如下图所示。这个是 Android 虚拟设备管理器，利用它我们可以创建，编辑和启动具体的模拟器。





4.2.2 创建和编辑模拟器

点击 Create 按钮用于创建一个新的模拟器，这将会打开一个创建新的模拟器的窗口，如下图所示。

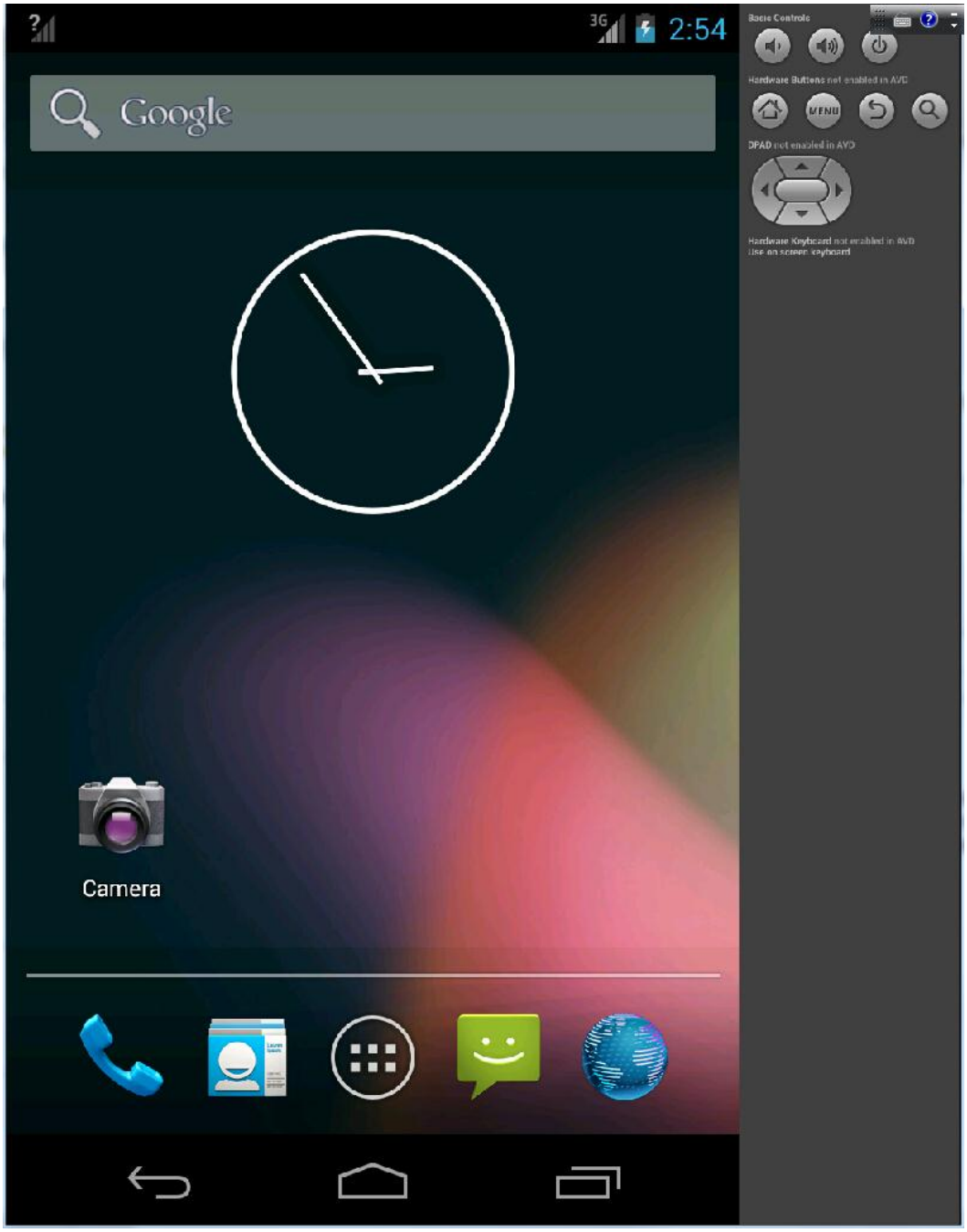


在这个窗口中输入或者选择所需的模拟器的参数，这里需要注意的是 Target 要选择和 Appium 或者脚本中相同的参数，Device 要选择符合对应的 Target，另外要保证 CPU/ABI 的值不能为空，Memory Options 中的 RAM 不要设置过大，我一般设置为 256 或者 512，VM Heamp 设置为 64，其他选项都保持默认值即可，最后点击底部的 OK 按钮就创建完毕了。创建成功后会出现在列表中。选中该项，可进行编辑操作，和创建类似，不再赘述。

4.2.3 启动模拟器

确定都没有问题了，选中需要启动的模拟器，点击 Start 按钮，在弹出的 Launch Options 对话框中直接点击 Launch 按钮，就会出现启动模拟器的进度条，之后出现模拟器的窗口，刚开始该窗口中一片漆黑，那是因为模拟器还没有启动完毕，等待一会儿，

可以趁机泡杯茶，就看到启动成功了，如下图所示。完全就是一个 Android 手机的感觉有木有。



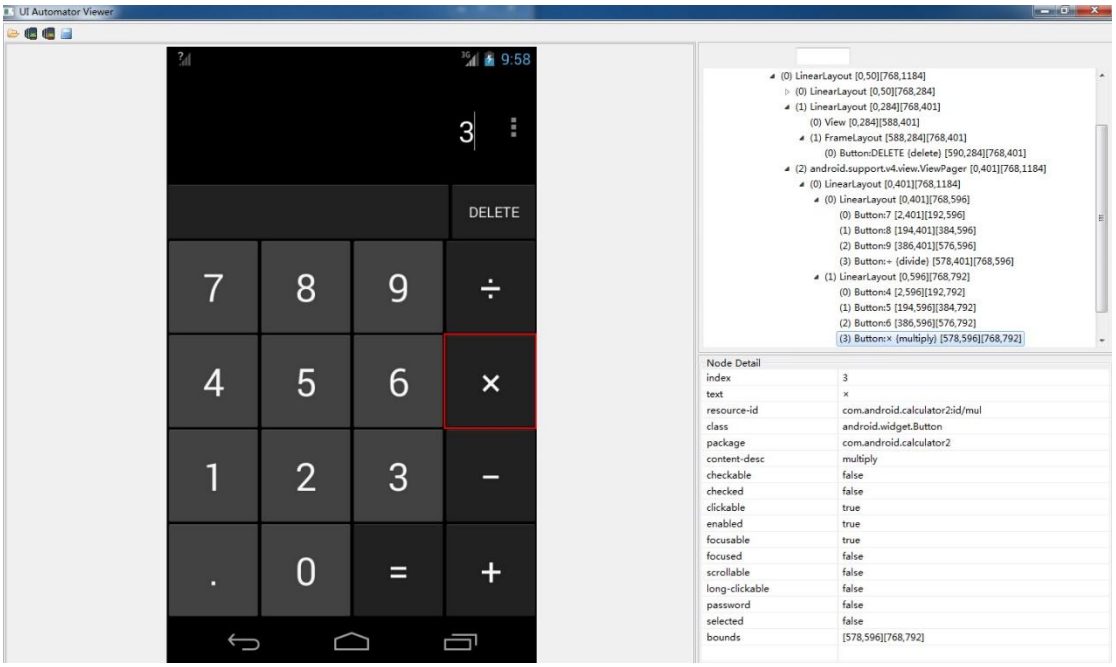
五、一个必不可少的工具介绍

万事具备，接下来我们就要开始编写测试脚本了。

不过，有个重要的问题还没有解决。我们知道 RobotFramework 在 web 网页中主要依靠元素的 id，name 或者 xpath 来定位页面上的元素，我们依赖浏览器的插件 firepath 来做到这一点，但在 Android 应用中，怎么定位页面上的元素呢？现在，这个关键的工

具就该出场了，它就是 uiautomatorviewer.bat，这个工具位于%ANDROID_HOME%\tools 目录下，双击运行它。

如果模拟器还没有启动，也把它启动起来，等模拟器启动完毕出现手机的待机界面时，对模拟器进行操作，我们打开计算器，点击数字 6，再点击乘号，这时点击 uiautomatorviewer.bat 左上角的第二个图标 Device Screenshot，这个图标点击后能对模拟器进行截图，但更神奇的是当我们将光标移动到截图中的元素后，在其右侧就会自动出现该元素的属性，如下图所示。



经常会用到的是 resource-id，text，class 属性，resource-id 在 AppiumLibrary(后面的文章会介绍)中其实就是 id 属性，想起来了吧，在 RobotFramework 中我们不是经常使用 id 吗？不过要注意：只有当测试的应用的版本设置为 4.3 及以上，才能看到 resource-id 哦。

有了这个方便好用的工具，接下来的编写代码脚本的工作就简单多了。

六、AppiumLibrary 介绍和安装

Appium 是个好东东，Android，iOS 都支持，并且居然 RobotFramework 也支持 Appium 了，这就是本文要介绍的 AppiumLibrary。

通过前面的文章大家知道可以使用多种语言来写 Appium 的测试脚本，但如果从编写效率和学习曲线上来说，当然是用如 RobotFramework 的关键字的方式最为简单，所

以 AppiumLibrary 就出现了，从 Appium 官网可知最新的版本是 1.2.5，在官网有其介绍和如何安装，为了方便不喜欢看英文的朋友，下面我把安装的方法简单介绍下。

Appium 的安装有两种方式，第一种就是使用 pip 指令，具体为：**pip install robotframework-appiumlibrary**，当然前提条件是安装好了 Python2.7，并且也安装了 pip 工具。

第二种方式就是使用 setup.py，可以官网页面右侧的下载链接下载后再进行解压缩，在命令行模式下进行到解压缩后所在的目录，最后执行指令 **python setup.py install**，或者先把 git 安装好，然后依次执行三条指令:

git clone https://github.com/jollychang/robotframework-appiumlibrary.git

cd robotframework-appiumlibrary

python setup.py install

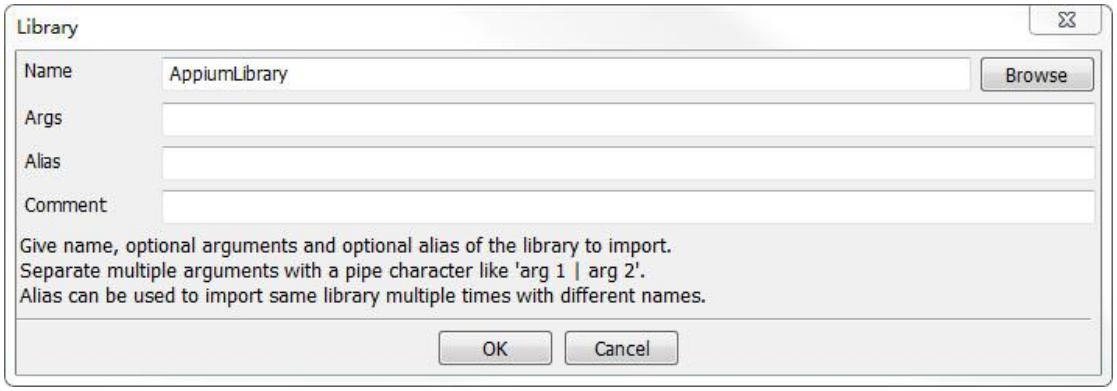
没有提示错误就是安装成功了。

七、RIDE 中 AppiumLibrary 的配置

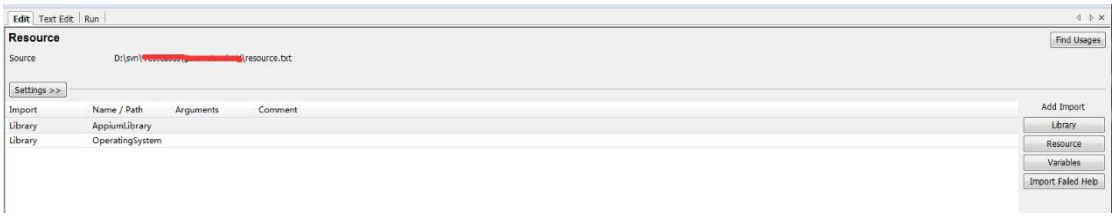
可能很多朋友已经迫不及待的想要用 RobotFramework+AppiumLibrary 来写 Android App 的测试脚本了，那我们也废话少说，直接开始。

首先打开 RIDE，这是编写 RobotFramework 测试脚本的集成环境，如果没有则先安装好，当然 RobotFramework 也要安装好，这些大家可以在网上查找相关的资料来解决如何安装的问题。

其次，新建一个 resource.txt 文件，这个文件是用来放置共用的脚本的，之后点击 Edit 选项卡，再点击右侧的 Library 按钮用于添加 AppiumLibrary 库，在弹出的的 Name 输入框中输入 AppiumLibrary，如下图所示。



最后点击 OK 按钮。如果在 Settings 下方出现黑色的 AppiumLibrary 行，就表明设置成功了，如下图所示。



八、一个简单的例子

万事具备，只欠编码！

下面看一个简单的示例，这个示例验证 Android 手机自带的通讯录的添加联系人的操作是否成功。这个例子是 Appium 官网自带的示例，有兴趣的同学也可以自己下载来研究和学习，下载地址：示例代码下载

首先请看 resource.txt 文件的代码：

```
*** Settings ***
Library           AppiumLibrary

*** Variables ***
${REMOTE_URL}     http://localhost:4723/wd/hub
${PLATFORM_NAME}    Android
${PLATFORM_VERSION}    4.4.4
${DEVICE_NAME}    Android Emulator
${APP}            ../../../sample-code/apps/ContactManager/ContactManager.apk

*** Keywords ***
add new contact
    [Arguments]    ${contact_name}    ${contact_phone}    ${contact_email}
    Open Application    ${REMOTE_URL}    ${PLATFORM_NAME}    ${PLATFORM_VERSION}    ${DEVICE_NAME}    ${APP}
    Click Element    accessibility_id=Add Contact
    Input Text    id=com.example.android.contactmanager:id/contactNameEditText    ${contact_name}
    Input Text    id=com.example.android.contactmanager:id/contactPhoneEditText    ${contact_phone}
    Input Text    id=com.example.android.contactmanager:id/contactEmailEditText    ${contact_email}
    Click Element    accessibility_id=Save
```

这里重点要说的是 Variables 下的五个变量，它们都是 Open Application 关键字的参数，用于在测试执行时提供给 Appium 相关的参数设置，之前的文章曾经说过可以不设置，在测试脚本中进行设置，这五个就是进行相关设置的参数。下面分别做下解释说明。

${REMOTE_URL}：远程 URL，指的是 Appium 所在的地址以及端口号，之后的 /wd/hub 为固定格式，请不要改动。

${PLATFORM_NAME}：平台名称，我们在 Android 中进行测试，只能写

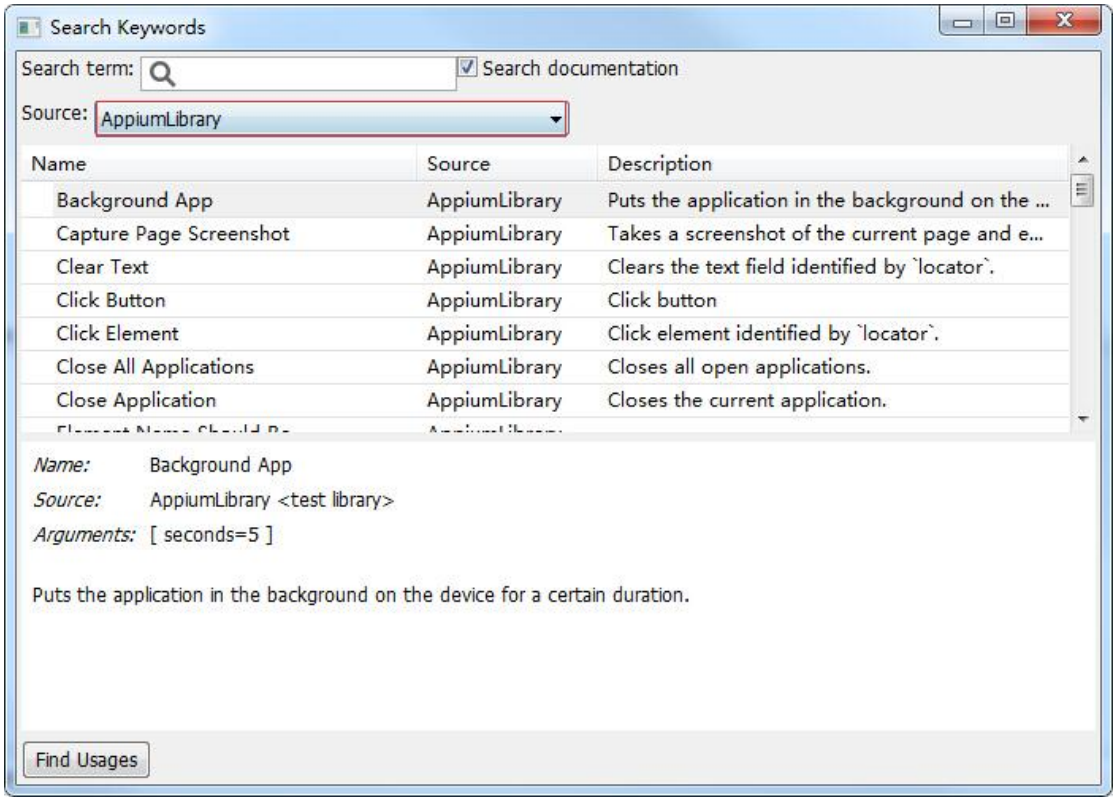
Android。

${PLATFORM_VERSION}：平台版本，也就是 Android 的版本号，这个要和模拟器中设定的版本相同即可。

${DEVICE_NAME}：设备名称，就是运行中的模拟器的名称，如果不知道，可以通过在命令行中输入 adb devices 指令取得。

${APP}：要测试的 App 的全路径，注意这个路径指的是相对于 Appium 所在的机器的路径，这里的 App 需要从是 Appium 官网下载，下载地址：示例代码下载，下载后在 sample-code\apps\ContactManager 目录下可找到 ContactManager.apk。

接下来的 Keywords 就简单了，可以打开 AppiumLibrary 得知每个关键字的具体含义，也可以更直接的在 RIDE 中查看关键字的含义，点击左上方的 K 图标，然后弹出 Search Keywords 窗口，再选择 Source 为 AppiumLibrary，如下图所示。



如何获取元素在之前的文章说过了，怎么样？写起来是不是很简单？

接下来再来看看 contacts.txt 里面的脚本：

```
*** Settings ***
Resource          test_android_contact_resource.txt

*** Test Cases ***
add_contact
    [Documentation]    demo for android_contacts(https://github.com/appium/sample-code/blob/master/sample-code/examples/python/android_contacts.py)
    [Tags]    demo
    add new contact    Appium User    someone@appium.io    5555555555
    Page Should Contain Text    Appium User
```

这个就更简单了，调用 resource.txt 文件里面的 add new contact 关键字，参数就是要添加的联系人的姓名，邮件和电话，最后一行是验证添加是否成功。

好了，有兴趣的朋友可以自己试试哦。

参考文献

1. https://github.com/appium

2. https://github.com/jollychang/robotframework-appiumlibrary

3. http://appium.io/slate/en/master/

作者简介

王东，男，有 11 年的软件测试和 8 年多的团队管理经验。对软件质量管理，软件测试有丰富经验，擅长使用 Robotframework 和 Appium 进行 Web 和移动 App 的自动化测试。

测试女巫之石头变宝石篇之三

◆作者：王平平

一、前言：

测试女巫又要开始念新咒语啦，先带着大家复习一下前五次我们学习的“咒语”和“法术”：“找到 bug 产生的原因”以及“提高 bug 产出效率”，“通过自动化节省人力”，“总结工作流程篇”和“总结芯片的特点和科学对比两款芯片的优劣”为例介绍 DOE，柏拉图，主效应，交互作用，相关性，鱼骨图，Xbar Chart，One Way ANOVA，Two Way ANOVA，QFD，DFMEA，量测系统，Johnson Transformation 假设检定，双样本检定，并着重介绍如何将这些方法应用到我们软件测试中。

这次我们学习新的“魔法”是什么呢？在我们实际工作中有两个需求：第一个需求：对很重要功能的“预警”需求，即对于某些重要功能，需要做长时间监控，在发现超过标准值前提出预警，不要等问题变得很坏的时候才给出警告，这样做可以使相关工程师有足够的时间去解决问题。

第二个需求，对于重要功能尤其是性能方面的功能，在客户无法给出具体的标准的前提下，如何根据收集的数据，根据科学的统计学的方法制定标准？或者根据客户给出的标准，使用科学的统计学的方法推算出出问题的机率？

对于这两个需求，我们可以分别用 6 sigma 专业的统计学的工具：统计过程控制（Statistic Process Control）SPC, X-Bar-S 控制图以及推估不良率/推估标准这三几个工具来完美呈现！

在使用这些工具前我们有一些背景知识需要了解，所以这次我们需要学习新的基本知识和工具为：

标准差和平均值的概念（这两个概念在我们此文章中用到工具都需要使用的基本概念），趋中定律（Central Limit Theorem），也是使用 SPC 和推估不良率/推估标准前需要了解的概念，最后是我们需要学习的目的工具：即统计过程控制（SPC），X-Bar-S 控制

图，推估不良率或者推估标准

二、6 sigma 常用工具基础知识介绍

1、标准差以及平均数的基本概念

1) 平均数：指的是一组数据中所有数据之和除以这组数据的个数，它是反映数据集中趋势的一项指标,它是描述数据集中位置的一个统计量，既可以用它来反映一组数据的一般情况、和平均水平，也可以用它进行不同组数据的比较，以看出组与组之间的差别。

公式 ：$$A_n = \frac{a_1 + a_2 + a_3 + \cdots + a_n}{n}$$

2) 标准差：标准差定义是总体各单位的数值与其平均数离差平方的算术平均数的平方根。它反映组内个体间的离散程度。

假设有一组数值 $X_1,X_2,X_3,.....Xn$（皆为实数），其平均值

（算术平均值）为 μ

公式:$$\sigma = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(x_i - \mu)^2}$$

简单来说，标准差是一组数据平均值分散程度的一种度量。一个较大的标准差，代表大部分数值和其平均值之间差异较大；一个较小的标准差，代表这些数值较接近平均值

例如，两组数的集合 {0,5,9,14} 和 {5,6,8,9} 其平均值都是 7 ，但第二个集合具有较小的标准差。

例如有 7 个人的月薪的平均值是 3000 元，第 8 个人月薪是 100000 元，那平均月薪是 50000 元；如果计算标准差就会发现此第 8 个人的离散程度差别比较大。所以在统计学中如希望对一组数据有一个全面的认识，需要考察两个参数：平均数和标准差

所以综上：在统计工作中，平均数（均值）和标准差是描述数据资料集中趋势和离散程度的两个最重要的测度值。

以下我们举个形象的例子来说明平均数和标准差:

平均数：决定了曲线的位置(抛物线中轴线即为平均数)

标准差：决定了曲线的形状，即曲线是高耸还是扁平

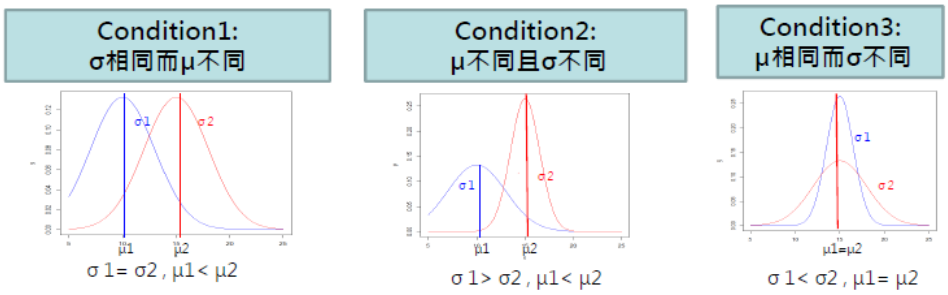
a. 标准差越小，曲线越高耸，表明数值集中在平均值附近的数值越多。

b. 标准差越大，曲线越扁平，表明数值集中在平均值附件的数值越少

下图是标准差和平均值三种情况的对比：

其中 μ 指的是平均值； δ 指的是标准差。

標準差 &平均值三種情形比較:



趋中定理（Central Limit Theorem）

1) 作用：它是所有统计学中的推论工具的基础理论，它主要阐述的是样本和母体之间的关系；换句话说如何通过对样本的研究来推论母体的分布。

从平均数和标准差的角度来说：即通过对于样本的平均数和标准差来推论未知母体的平均数和标准差。

2) 定义：假设存在一个母体的数据，平均值为 μ，标准差为 δ；随机从母体抽出一组样本（注意样本的数量 n，n>30），则此样本的平均数仍然为 μ，标准差为 $\sigma/\sqrt{n}$

注意：我们在实际使用中，是通过“样本”的平均数和标准差，根据趋中定理的公式来推论“未知母体”的平均数和标准差统计过程控制

1) 定义：

统计过程控制（简称 SPC）是一种借助数理统计方法的过程控制工具。

它对生产过程进行分析评价，根据反馈信息及时发现系统性因素出现的征兆，并采取措施消除其影响，使过程维持在仅受随机性因素影响的受控状态，以达到控制质量的目的。

目的为使过程平均值能符合目标值，并持续降低过程变异，是过程质量改善手法主要利用分析抽样样本数据，以判断过程是否处于稳定状态，当发现过程存在异常时，能适时采取矫正行动将异常原因排除掉。

2) 作用:

我们可以思考这个工具如何运用到实际工作中，我们可以用这个工具来长期检测某个重要功能其品质是否存在异常，是否在我们的“管制范围内”，如果超过“管制范围”就启动我们的警告机制。

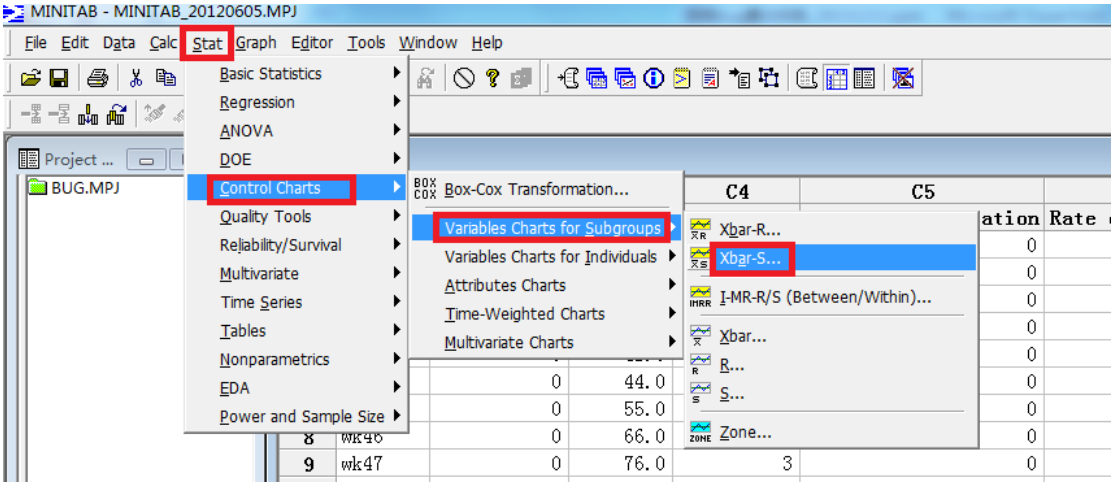
X-Bar-S 控制图

控制图是统计过程控制中非常具有功效的工具，它由三条管制界限组成的图形：中心线，上管制界限，下管制界限。将我们的样本资料绘入图中，判断样本数据是否在管制界面内。

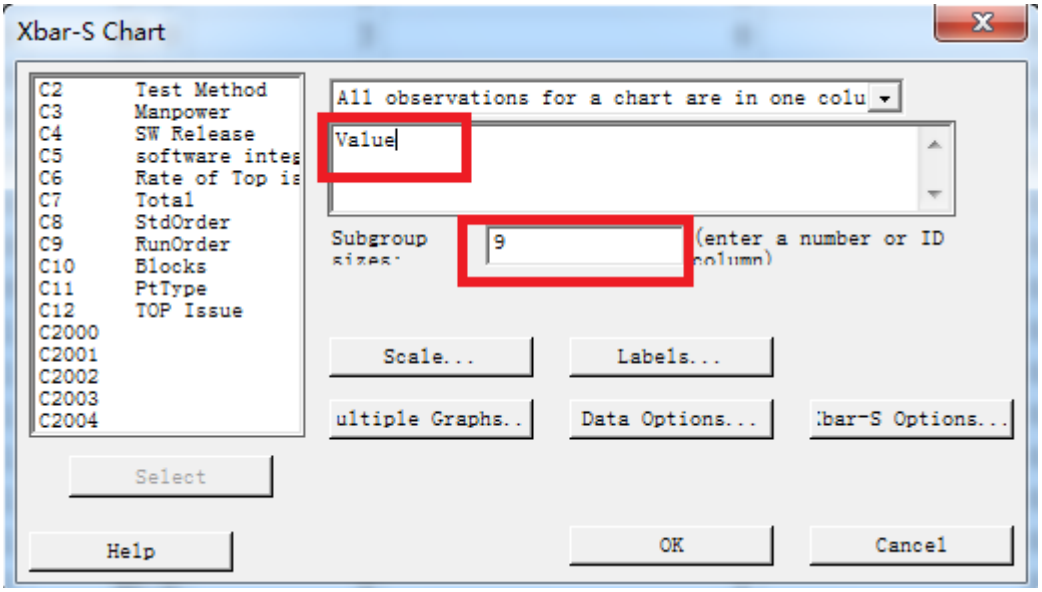
而 X-Bar-S 是众多控制图的其中一种：X 主要指的是平均控制图，而 S 指的是标准差控制图。使用 Minitab 画图的过程如下:

1) 搜集资料，例如医院对 10 名病人研究其血压水平，需要监视病人的血压的平均值和标准差。

2) 打开 Minitab 选择路径：Stat->Control charts->Variables Charts for Subgroups->Xbar-S



3) 将测量值选中到第一个红框中，将人数：9 输入第二个红框，如下图



4) 点击 Option 按键出现如下界面，此界面给出了判断方法，如下图:

常见的 8 种异常情形如下:

一点落在 3 s 外(OOC) (0.27%)

连续 9 点落在同一边 (0.39%)

连续 6 点递增或递减 (0.27%)

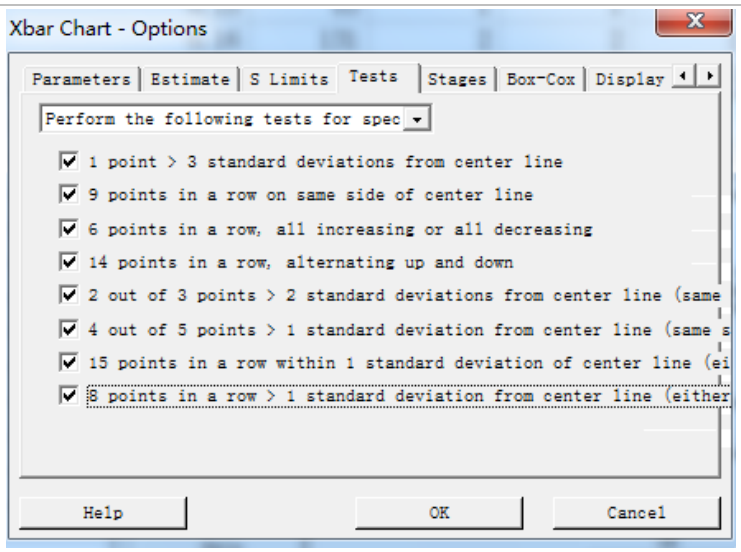
连续 14 点上下循环 (0.27%)

3 点内有 2 点落在 2 s 外(同一边) (0.36%)

5 点内有 4 点落在 1 s 外(同一边) (0.26%)

连续 15 点若在上下 1 s 的范围内 (0.31%)

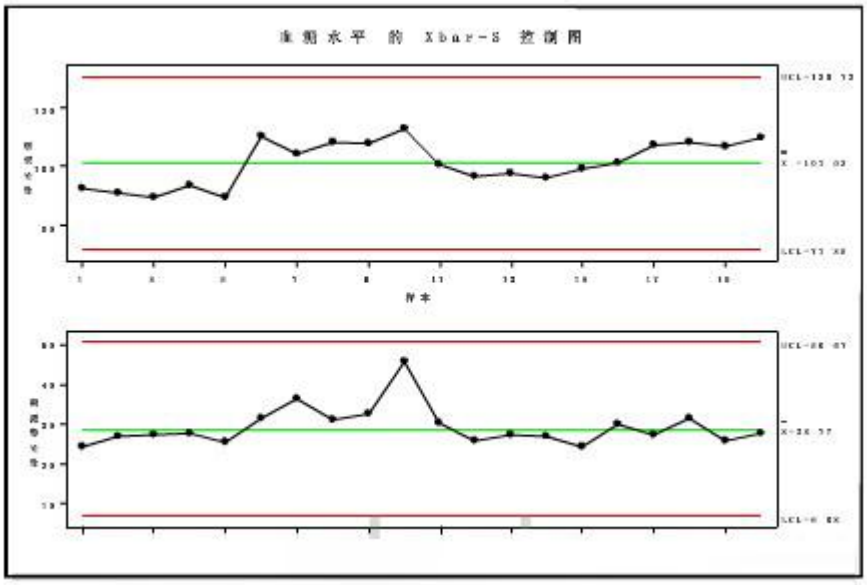
连续 8 点落在上下 1 s 的范围外 (0.02%)



5) 结果如下图:

即 X 控制图（平均值控制图）绘制在屏幕的上半部分，S 控制图（标准差控制图）绘制的下半部分，红线即为控制限制范围，如果抽样的数值均在红线范围内则说明不管是平均数（集中趋势）还是标准差（离散程度）都是在可控制的范围内

注意如果超过了红线的范围但是没有超过标准值，也要进行预警，因为它已经超过了可控制的范围，后续超过标准值的可能性非常大，这时就可以提前进行预警



推估不良率或者推估标准

1) 推估不良率

a. 使用的原理是:

“累积分布函数”（Cumulative distribution function）

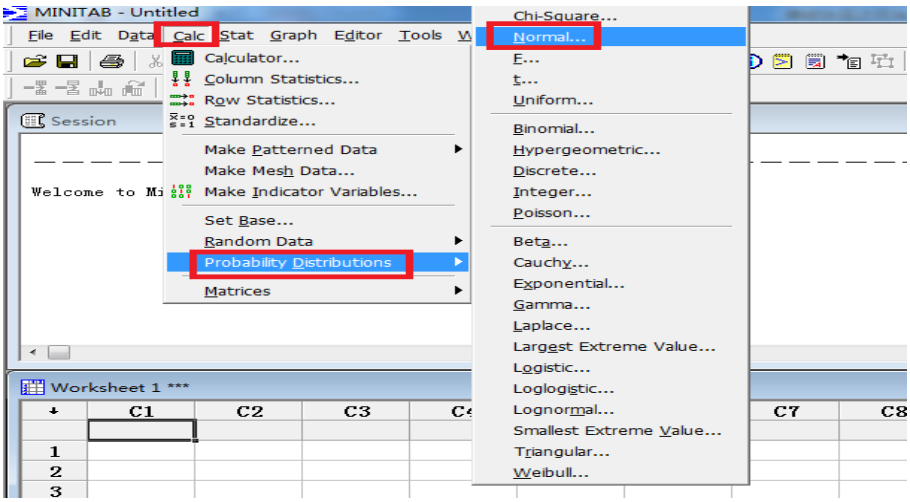
即小于等于 a 的值，其出现概率的和。使用公式即为：F(a)=P(x<=a)类比到我们实际的使用，可以这样理解，我们测试期望得到的数值为 a，使用这个统计学理论，可以得到小于或者大于 a 值的概率；

即只要我有了样本的数据，以及标准值，通过这样函数我就可以计算样本不符合标准值的机率，进而可以实现“推估不良率”

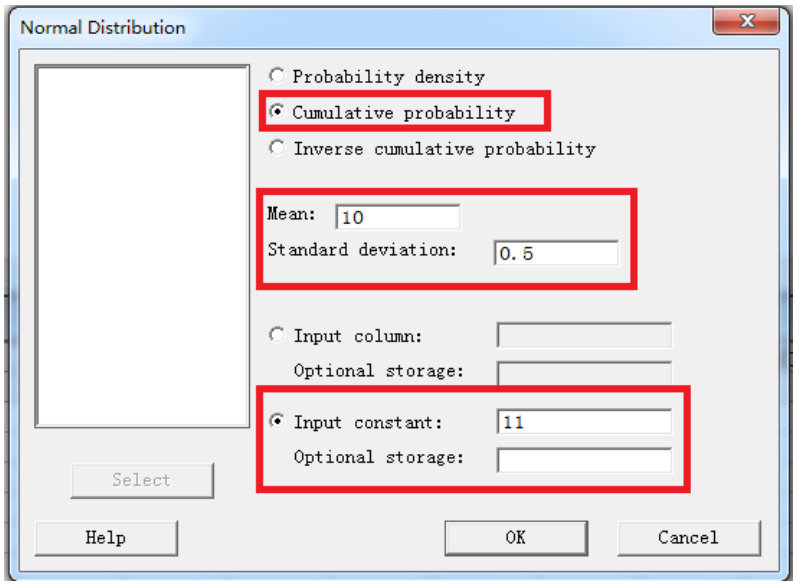
b. Minitab 操作步骤如下：

(1) 进入如下路径：Minitab->Calc->Probability Distributions->Normal，

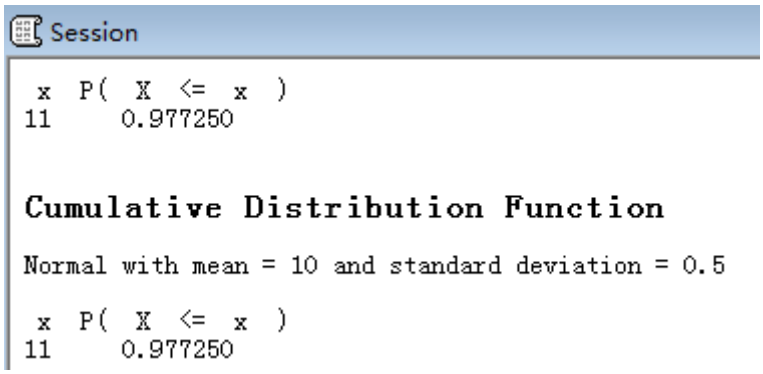
如下图：



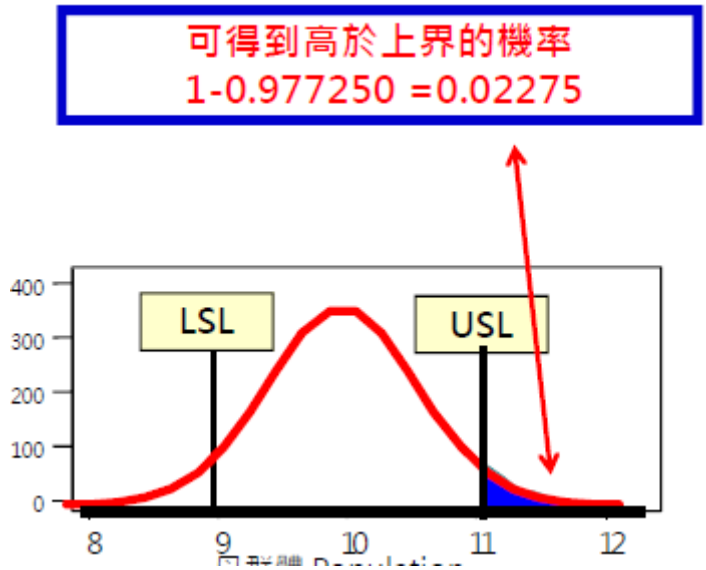
(2) 在以下红框标出的两个编辑框，输入样本的平均数（Mean）和标准差（Standard deviation），在 Input constant 输入规格值上界如下图：



(3) 点击 OK 得到如下结果



(4) 分析此结果，我们计算出来的值：0.977250 对应下图中的 USL（规格值上界）如果想得到高于“规格值上界”的机率 1-0.977250 即可。



综上就可以得到超过规格值上界的机率是 0.02275 即 2.275%

同样的道理计算 LSL 这里就不再赘述。

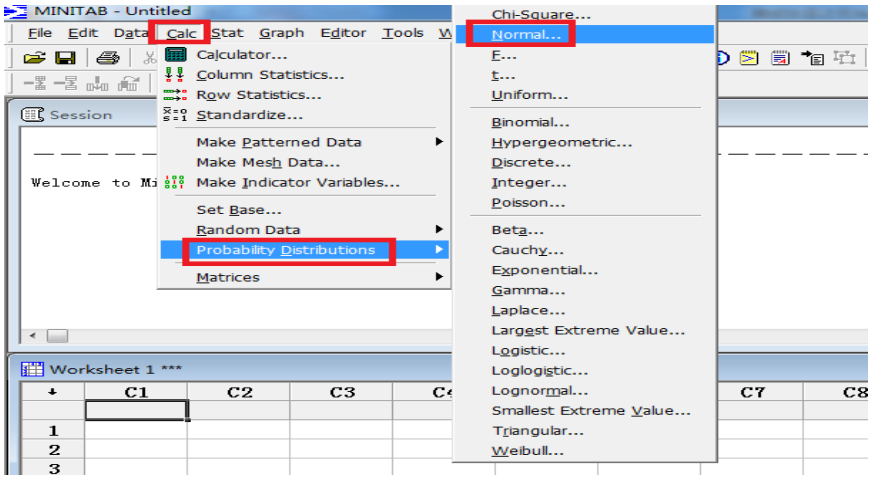
2) 推估标准

a. 使用原理:

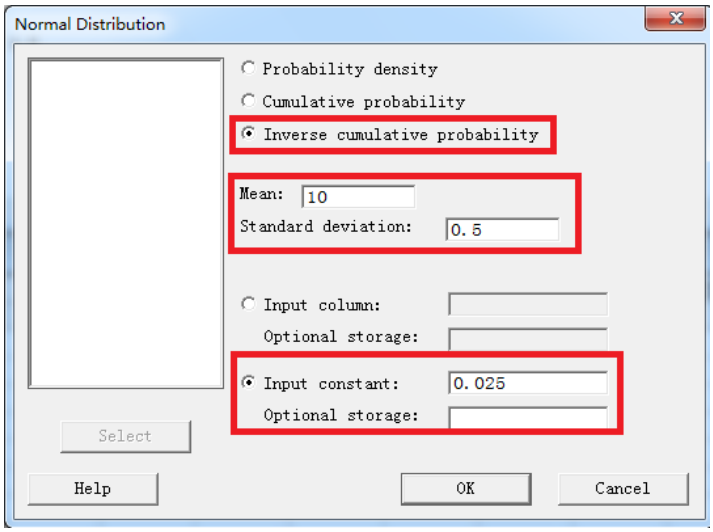
“逆累积分布函数”(Inverse Cumulative distribution function)。原理同“累积分布函数”，是通过给出的概率来推估标准，相当于“累积分布函数”的反推。

b. Minitab 操作步骤如下:

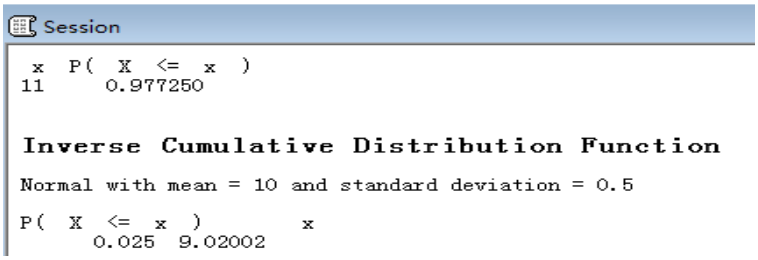
(1) 进入如下路径：Minitab->Calc->Probability Distributions->Normal，如下图:



(2) 在以下红框标出的两个编辑框，输入样本的平均数（Mean）和标准差（Standard deviation），在 Input constant 输入概率的上界如下图:



(3) 点击 OK 得到如下结果



(4) 可以得到规格下界(LSL)为 9.02002

应用到实际工作中之追踪重要功能稳定性篇

根据之前做项目的经验，对于一个项目来说最重要的功能一般都要长时间追踪，确认其稳定性；例如对于路由器产品来说，无线网络连接的稳定性就是其“关键功能”，需要长时间追踪。长时间的稳定性主要确认其无线网络协议中丢包率来体现。对于这个需求，我们可以使用控制图来进行预警。

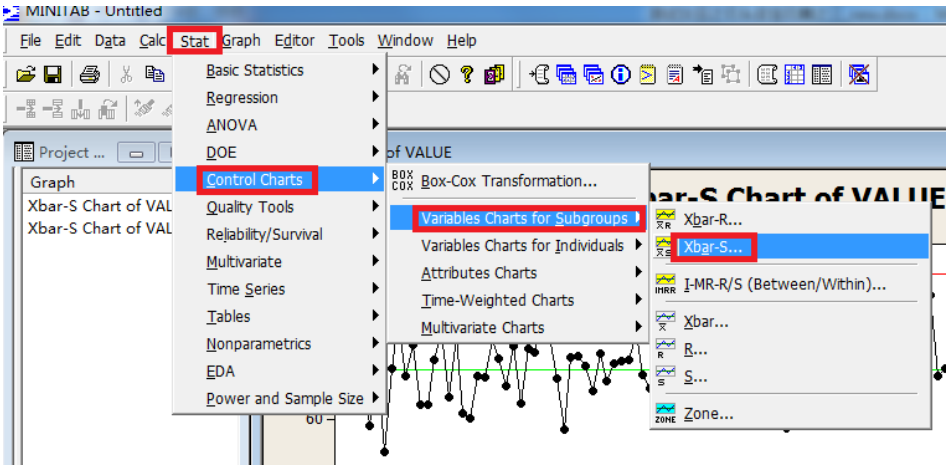
搜集 A 项目 WIFI 的丢包率

1) 使用 110 个硬件，每个硬件测试 3 天，每天都固定测试 3 个小时，搜集 WIFI 连接测试的丢包数量，如【表 1】

MBSN	Days	VALUE
A1	1	61.1691
A1	2	68.2734
A1	3	54.5554
A1	8	65.3780
A10	1	62.8145
A10	2	62.3917
A10	3	64.5663
A12	1	69.1502
A12	2	64.3184
A12	3	65.3251
A13	1	72.0517
A13	2	63.0915
A13	3	55.4428
A14	1	59.0064
A14	2	54.2987
A14	3	56.1569
A15	1	72.7284
A15	2	55.1195
A15	3	65.0368
A16	1	61.9264
A16	2	70.0858
A16	3	52.2920
A17	1	65.5234
A17	2	60.9574
A17	3	64.2606

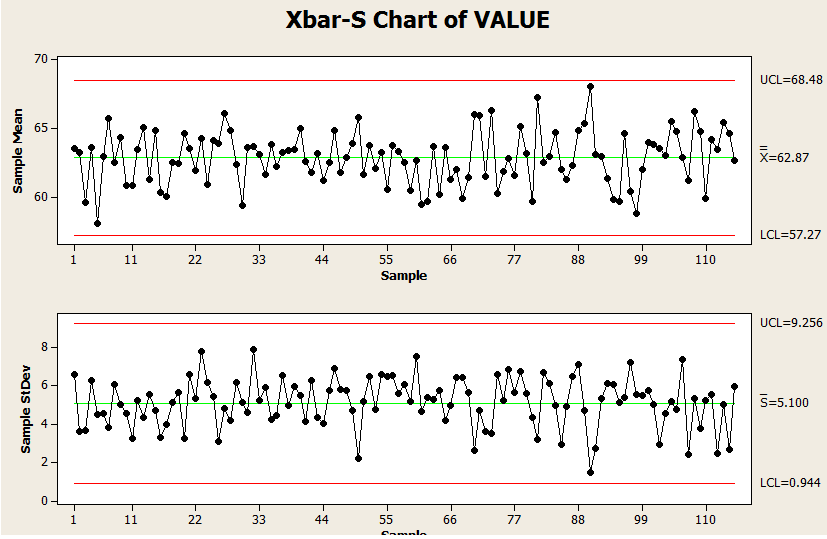
【表 1】

2) 使用 Minitab 中的 XBar-S 此工具如下图



3) 点击 ok 后就可以看到下图，有两个控制图，第一个是平均值的控制图；第二个图是标准差的控制图；从这两张图可以看出其丢包的数量均在红色管制线之内，但是第

88 个硬件的丢包数量已经接近管制线，可以将这个第 88 个硬件发出软性警告，建议硬件工程师分析一下这个硬件是否有出现异常的趋势。



总结方法

1) 首先我们要确认我们要监控的重要功能是什么例如上面的例子我们需要研究的是某一款路由器的无线连接性能，以丢数据包多少来衡量“无线连接性能”

2) 然后我们需要确定此功能如何分类例如上面的例子，我们确认以不同的硬件，每个硬件测试不同的时间来分类，注意要利用我们学习的“趋中定律”的核心理论：想要了解母体的状态，至少抽样的样本要大于 30 个，这个例子我们抽样的样本是 110 个远远大于 30 个，所以我们研究此 110 个样本来推论母体是可靠的。

3) 然后根据你的定义分类，搜集数据

例如上面的例子就是开始搜集不同的硬件，在不同时间段测试的丢包的数量

4) 最后使用 Minitab 画出“XBar-s”图，注意画的是平均数和标准差的控制图，然后根据控制图的状况，决定是否需要给出“预警信号”应用到实际工作之如何重要功能的规格值

实际工作中需要判断的例子

在项目规划初期，经常会有这样的需求：对于某款产品的重要功能客户无法给出确切的标准，需要我们自己定义标准。如何定义一个即合理又对我们自己有利的标准呢？例如对于一款路由器产品，天线的发射功率是非常重要的一项功能，如果客户无法给出确切的标准，我们如何定义标准呢？

首先我们需要搜集类似产品的测试数据，例如我们搜集的数据是某一产品的发射功率的测试结果，注意我们搜集的是 35 个结果，且是随机搜集的，只有这样才保证样本充足的数量，进而研究这 35 个样本才能反映出母体的实际情况。将测试的结果搜集如下【图 5】

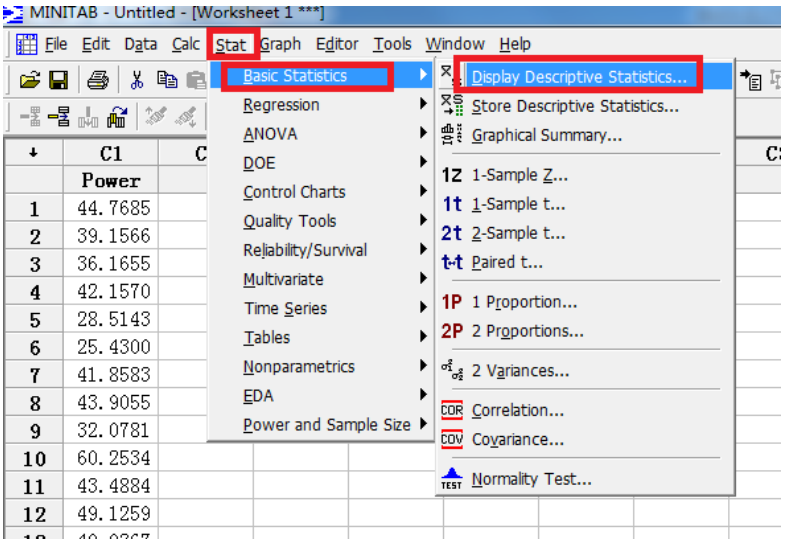
C1
Power
44.7685
39.1566
36.1655
42.1570
28.5143
25.4300
41.8583
43.9055
32.0781
60.2534
43.4884
49.1259
40.0367
40.6685
40.4098
38.3380
39.4176
38.8720
20.8071
33.1967
24.0945
42.5941
40.7834
35.5942
41.7327
34.1390

【图 5】

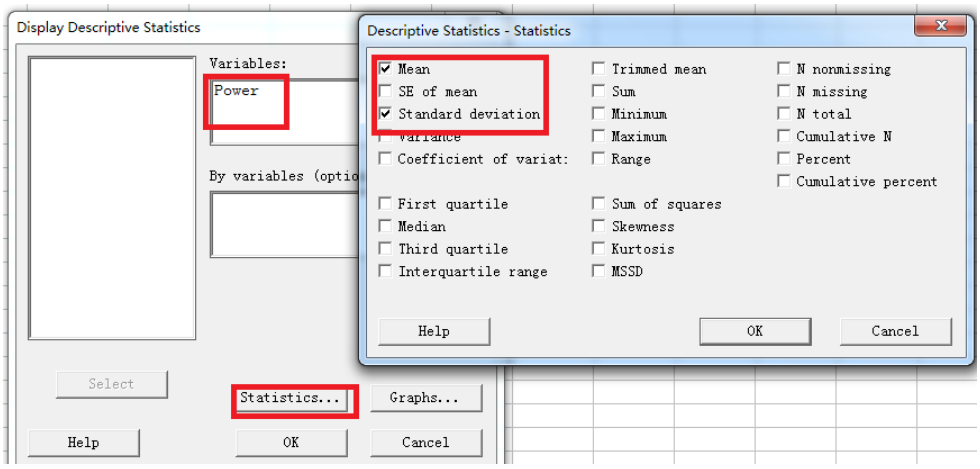
分析步骤

1) 先计算出上面数据的平均数和标准差，方法如下:

进入 Minitab 的菜单:Stat->Basic Statistics ->Display descriptive Statistics 如【图 6】



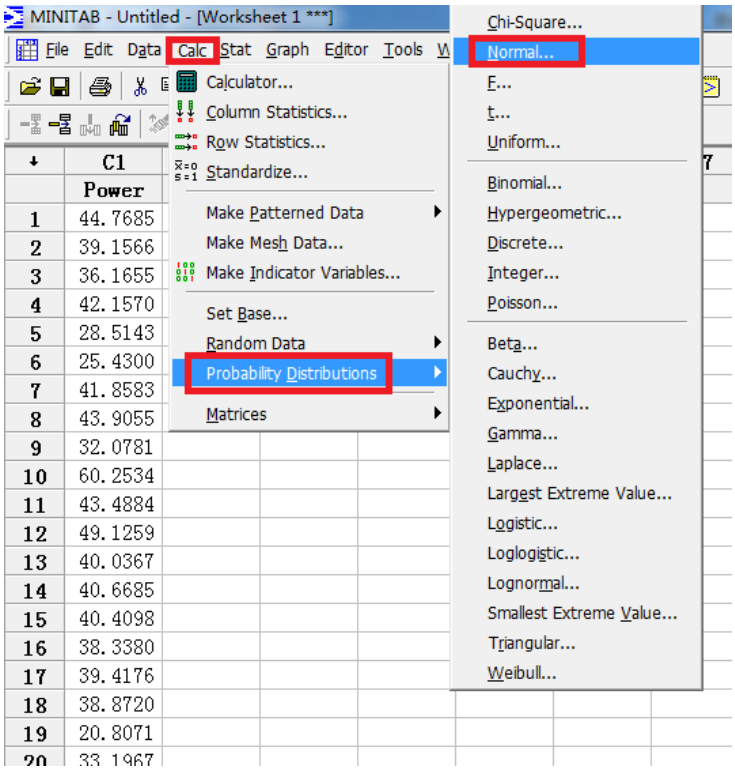
2) 选择 Power 这个数据，点击 Statistics 此按钮，只需要勾选 mean 以及 Standard deviation 点击 OK 就可以计算出此样本的平均数以及标准差



Descriptive Statistics: Power

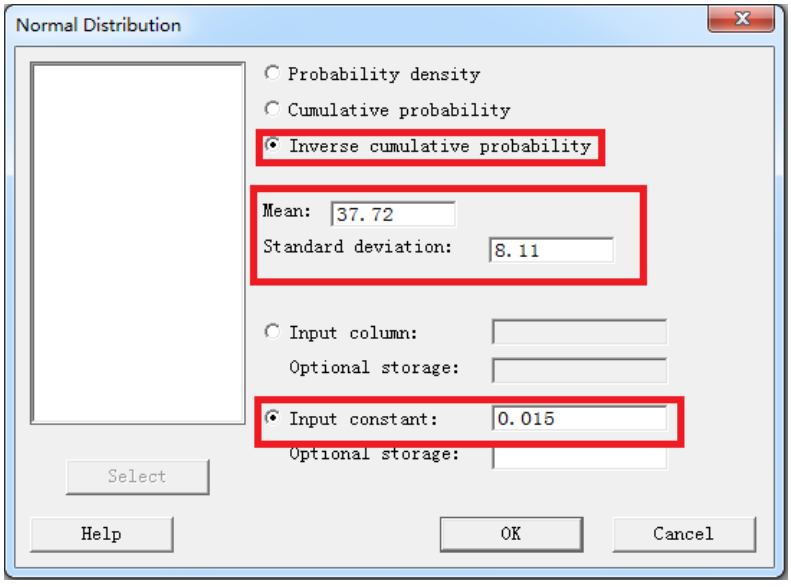
Variable	Mean	StDev
Power	37.72	8.11

3) Minitab 的菜单:Calc->Probability Distribution->Normal 如【图 6】



【图 6】

4) 输入 2)中计算的平均数以及标准差，注意我们要首先定义可接受的不符合标准的比率是多少，例如我们定义超过标准的比率为 3%，【图 7】，则 Minitab 设置如下:

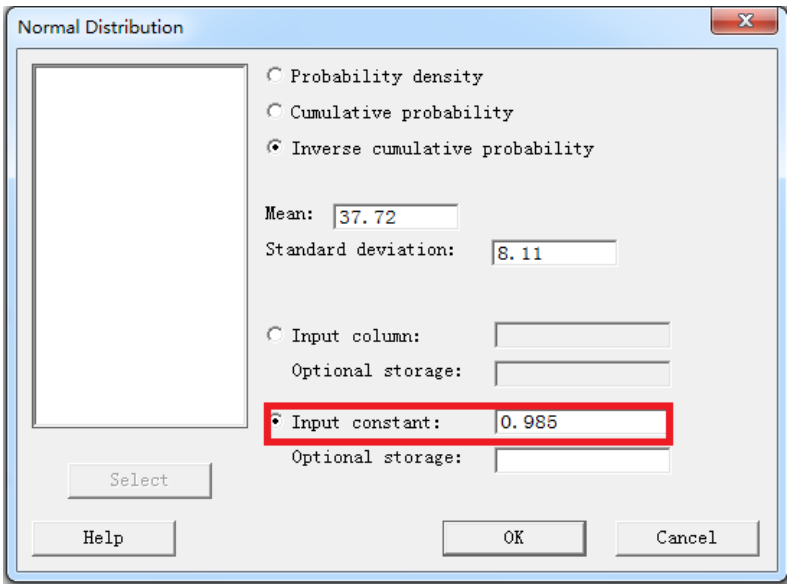


【图 7】

3) 点击 OK 后，得到上界值，如【图 8】

```
Inverse Cumulative Distribution Function

Normal with mean = 37.72 and standard deviation = 8.11

P( X <= x )        x
      0.015   20.1206
```

4) 下界值的计算方法:



【图 8】

5) 点击 OK 键后计算出下界值为:

```
Inverse Cumulative Distribution Function

Normal with mean = 37.72 and standard deviation = 8.11

P( X <= x )        x
      0.985   55.3194
```

6) 结论

综上我们得到天线发射功率的规格值是: 20.1206 < SPEC < 55.3194 我们就可以告诉用户如果接受此功能成功运行的比率是 97%的情况下，我们需要制定的标准为 20.1206 < SPEC < 55.3194

总结方法

1) 首先要明确我们需要设立的标准是什么

2) 搜集之前项目有关这个标准的数据，这个数据就是样本，所以至少要搜集 30 笔资料

3) 使用 Minitab 计算此样本资料的平均数和标准差，定义可接受的不符合标准的比率是多少

4) 使用 Minitab 的 Probability Distribution 输入平均数和标准差，以及输入相关的比率，即可以得到此标准

总结

我们在这一期中介绍了两个重要的知识点，

第一个是在项目进行前如有这样的需求：即对某一个重要功能制定合理且能说服客户的标准，因为客户无法给出非常确切的标准值，但是我们可以先可以与客户讨论，他能接受的“不合格标准的比率”，在得到客户提供的比率后，使用“逆累积分布函数”原理即 Minitab 的 Probability Distribution 此工具，根据之前项目搜集的资料，进行分析并得出我们既可以达成且客户也能接受的很科学的标准。

第二个是项目进行中的需求：即“提前预警机制”对于重要功能需要时时监控，在还未出现严重问题前给出预警，使相关的工程师有足够的时间去解决问题，去监控这个重要功能，可以让项目因为这个重要功能而产生风险的风险大大降低。相信不管是项目经理还是开发经理，对于这个“预警机制”都是大大支持的！

这次使用的工具非常简单，但是我还是要强调，“简单”的工具下隐藏了“不简单”的统计学原理，如果希望后续在大家自己的工作中灵活的运营这些工具，必须要理解这些统计学的原理。

还是那句话：对于 6 sigma 可以带来的奇妙旅程，我将不懈的坚持探索，怀着赤子之心去探索如何使用这些工具去改善我的工作，反之在不断的使用这些工具的过程中又大大加深了我对统计学原理的认识，所以“路漫漫其修远兮，吾将充满欢喜的上下而求索”！

参考文献:

1. 有关标准差基本概念介绍的网站：http://baike.baidu.com/link?url=4v_m3jJhHUKllt1lA0tRe_I-pVurV1tNNrztJ6_PbZf0Me5rJr-bxIdp4fRCdsrsSIRd-hSFBglCEyZCkunk6K

2. 有关介绍统计过程控制的网站：

http://baike.baidu.com/link?url=M9BLOK736USXqiqCHa2uhW624zYJpD-UXRAotTwo9PyTn-AWZv3GDHofg2Gz_uOqC42RK1GoKdLf-Wr30fWJR0_sgWYIj_ItTG-JSdB1Amm

手机应用程序测试策略规避

◆ 译者：why5256

想你的应用程序没有 bug? 注意了,近一半的问题都是客户发现的。

为什么不是开发人员发现这些缺陷呢? 都怪糟糕的测试，实际上一些流行的测试策略是会破坏你的应用程序的。

幸运的是糟糕的测试是很容易避免的。这里有五种最常见的移动应用程序测试错误方式以及如何去做的例子。

1. 公测

当应用程序进行公测，开发商发布警告，启动程序，看发生什么。但本质上，实际用户做了 beta 版的测试者。

为何要避免

有几个理由可以说明公测是危险的，首先，你无法控制用户体验。你只是发布了一个应用程序，但是不知道如何响应客户动作，网络环境以及市场需求。

这是一个巨大的风险，如果你的用户体验糟糕的话，你的应用程序口碑和你的品牌形象都会受到伤害。

其次，公测没有一个系统的方法来记录和解决问题。即使再忠实的客户也不可能让他们用一致的方式报告崩溃和其他问题。

一些开发商用雇佣测试人员来为他们的应用程序做出反馈，来作为变通手段。

那么问题是?

所有的应用程序测试者，无论是有组织的用户还是众包雇员，都是在不受控制，可变条件下使用应用程序的。导致在德国连接 3G 网络死机的原因与在巴西使用 LTE 导致

死机的原因并没有什么关系。

所以不要把你的程序抛在那些有可能你无法跟踪发生了什么的地方，更不用说解决那些肯定会出现的问题了。

2. 接入点映射

欢迎使用接入点映射，一个理论上确实非常好的测试想法。

开发商雇佣测试人员（哦哦，我们已经排除了一个坏的开始）来进行公测。

但是无论何时何地，测试者都是通过绕特定区域街道开车或步行，来观察在不同位置以及网络中的执行情况，而不是使用应用程序。

为何要避免

接入点映射要比公测稍微可控，但是条件仍然不理想。

即使你雇佣了大量的测试人员在他们自己所在城市和社区进行接入点映射测试，但他们最终用户体验还是对应他们各自的条件。

在应用程序使用中涉及的位置，设备以及网络创建的一个个个性化的体验，只适用于这个人在特定的某天时间。

换句话说，从在周三下午的托莱多使用 3G 网络进行应用程序测试的 Bob 那里收集的任何数据，都不适用于在周三上午三藩市使用 Wi-fi 连接网络的 Suzie。

接入点映射测试是一个卑鄙的测试方法，因为能感觉到不同条件下真实人的真实数据是不错的，问题是，这些条件不能适用于所有用户，导致这个方法只能是部分有效。

3. 去带宽

一旦你意识到任何形式的公测都是不可依赖的时候，就是进入实验室模仿在受控环境条件下的真实条件的时候了。

许多开发商以及企业进入测试实验室但是走的不够远，进行所谓什么“部分仿真测试”。仿真获取了一些，但不是所有影响应用的真实世界条件。

为何要避免

部分仿真往往忽略了重要的环境因素，创建不完整的测试并没有捕捉到全方位的用户体验。

考虑到网络带宽，应用程序通常是在静态带宽的条件下进行测试，但现实生活中，带宽却很少是静态的。用户可以在 3G 和 4G 网络，4G 网络和 Wi-Fi 之间互相切换来感受不断波动的信号强度。

延迟也是一个很重要的因素——对于很多应用程序性能这也是主要决定因素。与移动环境的其他方面一样，延迟是高度动态的。这取决于很多因素，如路由器等网络设备间的信号交换，编码技术，和网络协议。

既然现实世界的移动环境是如此的不同，那么在实验室中创建静态条件来测试移动应用程序还有一定价值的。

4. 忽略抖动

在一个移动应用程序测试环境中抖动很难作为代表，静态变量的带宽或延迟更容易创建。因此一些测试淡化抖动值。

为何要避免

不考虑流媒体需求，冒着严重到令人失望的终端用户体验的风险（更不用说失去潜在的收入和推荐）去测试你的应用程序。

当评价应用程序是如何执行的时候，需要考虑两个关键领域：如何让程序自身进行操作以及如何在特定网络上进行程序操作。换一种方式，忽略抖动就意味着忽略网络性能。

归功于带宽，视频特别容易抖动，流媒体质量很大程度上受位置，网络类型，服务提供商和其他等因素的影响。

5. 纯功能检测

纯功能检测是开发者只测试应用程序那些功能性的元素，并没有将性能纳入到测试过程中去。

为何要避免

一个移动应用程序的成功不单只基于功能。

一个应用程序的功能必须做什么（就是说，当某个功能被选中或者某个按钮被按下时发生什么）。一个应用程序的执行，另一方面，是要做的怎样（就是说，在一个特定网络上使用时要如何快速的反应）。

测试当用户发出命令时会发生什么是功能测试，测试应用程序如何快速响应要求，从另一方面是性能测试。

为获得整体应用程序能力的一个三维视图，功能和性能两个都必须进行测试。

你当然希望应用程序功能正常，测试功能而不测试性能将永远得不到你的应用程序可能（或不能）给你的全部画面。正如我们迄今所看到的，整体应用程序的性能很大程度受外界因素的影响，如网络性能。

这就意味着，为创建最准确的测试可能，功能，性能和外部的影响都要考虑在内。一定要考虑一下的内容:

- 什么网络条件是被虚拟化了的?
- 那些条件是基于实际网络的么?
- 你模拟了多个网络条件了么?
- 你代表的是分布式用户群么?

最后一点至关重要，功能测试往往忽视需要虚拟化不同用户的不同限制条件。

基于云的移动应用程序测试也应谨慎，请记住，功能云测试只是提供一个单一定位视点，并不代表整个用户群。云测试也不能对真实的用户如何在网络上使用应用程序给出准确的画像，因为云连接往往比家庭或者其他网络的速度更快。

要采取哪些措施

所以如果要公测的话，静态带宽测试，部分仿真，纯功能测试要避免。那怎么在往市场推出应用程序前进行准确的测试呢?

1. 做功课

在测试你的应用程序前有很多工作要做。

你首先要非常了解影响功能，性能和用户体验的各种因素。

研究网络条件，基础设施，用户位置，以及一旦开始就需要考虑在内的其他环境条件。

对于如何，何时，何地使用你的应用程序才能帮助你创建一个虚拟的能准确代表真实世界用户体验的测试环境，要有一个彻底的三维理解。

2. 去虚拟

创建虚拟条件包括前一步中发现的所有因素。

创建包含用户在现实生活中所经历的各种各样的变化的虚拟网络条件是非常重要的。

你创建的虚拟网络与功能性能工具应该能够无缝集成，用来进一步提高测试的真实性与可靠性。

3. 分析与优化

接下来是分析你的结果的时候了。寻找功能和性能两方面的故障，以及可以归因于网络故障的任何错误的解决方法。

最后，开发系统来测试分析优化你的应用程序。

做正确的移动应用测试

为了确保你的移动应用程序的性能最佳，关键是要建立一个能准确反映现实世界条件的测试环境。

- 不推出未经事先测试的应用程序
- 不要浪费时间与金钱在接入点映射上
- 请记住现实世界中的带宽是可变的
- 考虑视频网络可能会影响应用程序的功能和性能
- 永远要对功能和性能进行测试
- 深入研究影响你最终用户的环境因素
- 创建三维，真实世界的测试环境
- 分析你的测试结果，并持续优化系统

考虑到在现实条件下你的应用程序的功能和执行，通过仿真所有这些虚拟测试条件，你就可以准确有效的预测你的应用程序一旦推出用户体验如何。

WebUI 自动化(PageObject_Python)

◆ 作者：姜林斌

前言:

2014 年的上下半年对我来说可谓冰火两重天，上半年 Lead 并全程跟进一个银行的现场实施项目各种被虐！下半年从 WB 的坑跳了出来专攻 PC 端 Web UI 自动化(有些前辈告诉我说这是更大的一坑，可是坑总会有人去把它们填平的)，码农性发，每每敲至深夜，总体感觉是累，但我很快乐！~特以此篇总结和犒劳 2014 年且行且珍惜且快乐的自己！

1、Web UI 自动化的意义和难点

Web UI 与最终用户最近，基于用户场景的 UI 自动化测试还是有其重要的意义的。使用 UI 自动化测试对产品的关键功能路径进行验证及回归，比起传统的 QA 手工执行 Test，case 可以更快地得到反馈，也让当前发布版本更透明。

理想状况下，我们应该将所有可以固化下来的 Test case 都自动化起来，而让我们昂贵的手工测试关注于更有挑战性的新功能的测试。让机器做已知领域的事儿，让人工关注未知不稳定的领域。

UI 层的测试距离交付最近，但是成本也最高。编写和维护 UI 自动化测试需要付出比其他自动化测试（如接口/单元自动化）更高昂的成本，

相比较系统的其他部分，UI 是一个多变的层，如果 UI 自动化测试没有构建好，即使界面的一个微小改动，整个测试集可能就天崩地裂。对于 UI 自动化测试，可维护性是要考虑的首要因素。

在产品的迭代和更新过程中，若自动化脚本的可维护性太差 那还不如不做，产出/

投入性价比太低或几乎没有，这样的自动化团队注定是走不远的。

2、Webdriver 简介

Selenium 2.0 主要的特性就是与 WebDriver API 的集成。

WebDriver 旨在提供一个更简单，更简洁的编程接口以及解决一些 Selenium-RC API 的限制。Selenium-Webdriver 更好的支持页面本身不重新加载而页面的元素改变的动态网页。WebDriver 的目标是提供一个良好设计的面向对象的 API，提供了对于现代先进 Web 应用程序测试问题的改进支持。

WebDriver 与 Selenium-RC 相比如何驱动浏览器

Selenium-WebDriver 使用每个浏览器自身对自动化的支持来直接调用浏览器。这些直接调用怎么做取决于你所使用的浏览器。

Selenium-RC 对于每个支持的浏览器采用相同的方式。当浏览器加载的时候，它“注入”浏览器的 javascript 功能，然后使用 javascript 来驱动浏览器内的应用程序。

WebDriver 不使用这个技术。再一次,它直接使用浏览器的内建自动化支持来驱动浏览器。

WebDriver 和 Selenium-Server

你可能需要或者可能不需要 Selenium Server。取决于你打算如何使用 Selenium-WebDriver。如果你只用 WebDriver API 你不需要 Selenium-Server。如果你的浏览器和测试运行在同一机器，你的测试只使用 WebDriver API，你不需要运行 Selenium-Server，WebDriver 会直接运行浏览器。

但是有一些原因你需要使用 Selenium-Server，你使用 Selenium-Grid 来在多个机器或者虚拟机中分布运行您的测试。您想连接一个远程的特定的浏览器

<此处均为引用他人>

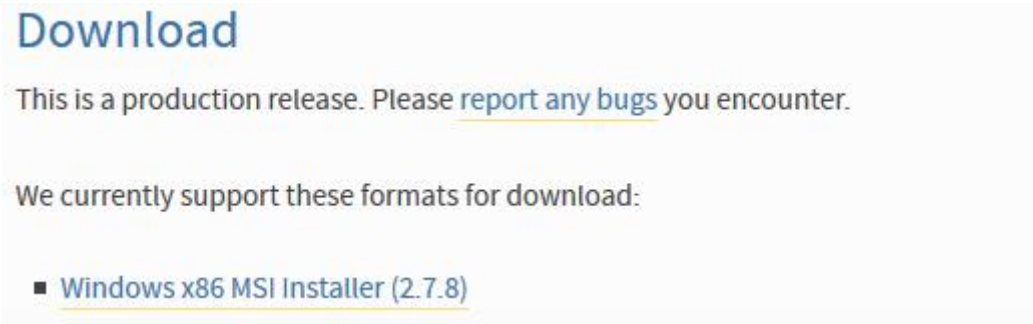
3、Webdriver+Python 自动化环境搭建

3.1：安装 python

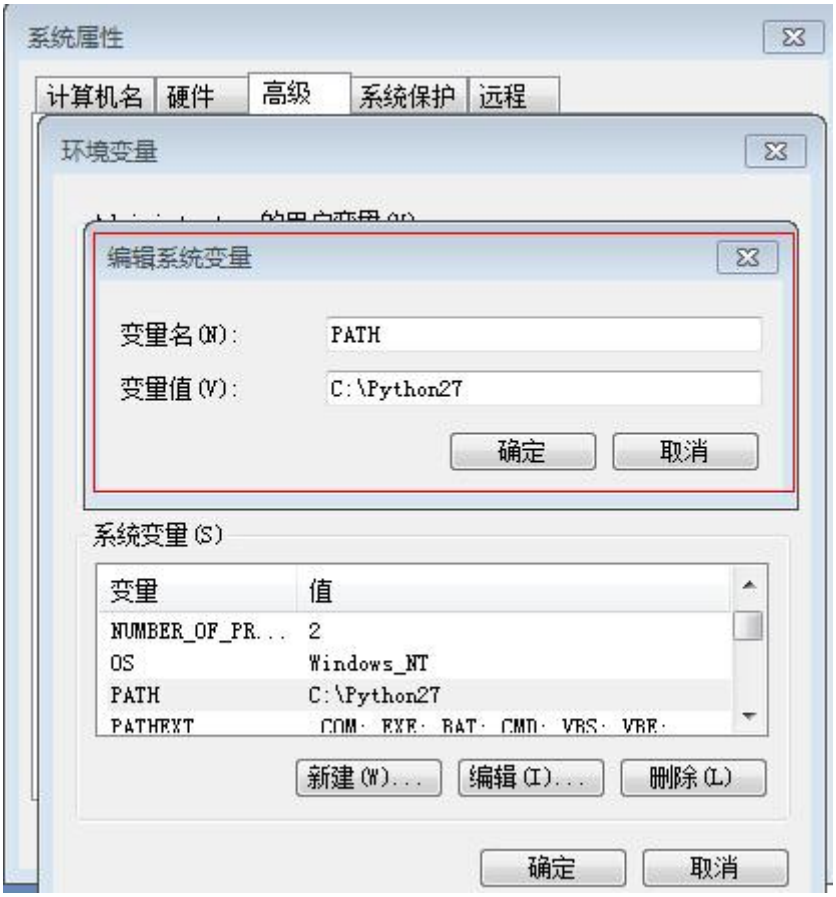
python2.7 版本(最新的 python 版本是 3.4，但用户体验没有 2.7 版本的好，我们选择用 2.7 版本)

下载地址：https://www.python.org/download/releases/2.7.8/

下载 Windows x86 MSI Installer (2.7.8)安装包



设置系统环境变量



3.2: 下载安装 setuptools 工具集

在命令行方式下，进入目录“C:\Python27\Scripts”，执行 python ez_setup.py。

3.3: 下载 pip 【python 的安装包管理工具】

https://pypi.python.org/pypi/pip

解压 pip 包并进入到解压后的文件目录下。

执行 python setup.py install

再切换到 C:\Python27\Scripts 目录下输入:

C:\Python27\Scripts > easy_install pip

3.4: 安装 selenium

在 Python 安装目录下的 Scripts 目录下执行 pip install -U selenium 安装 selenium

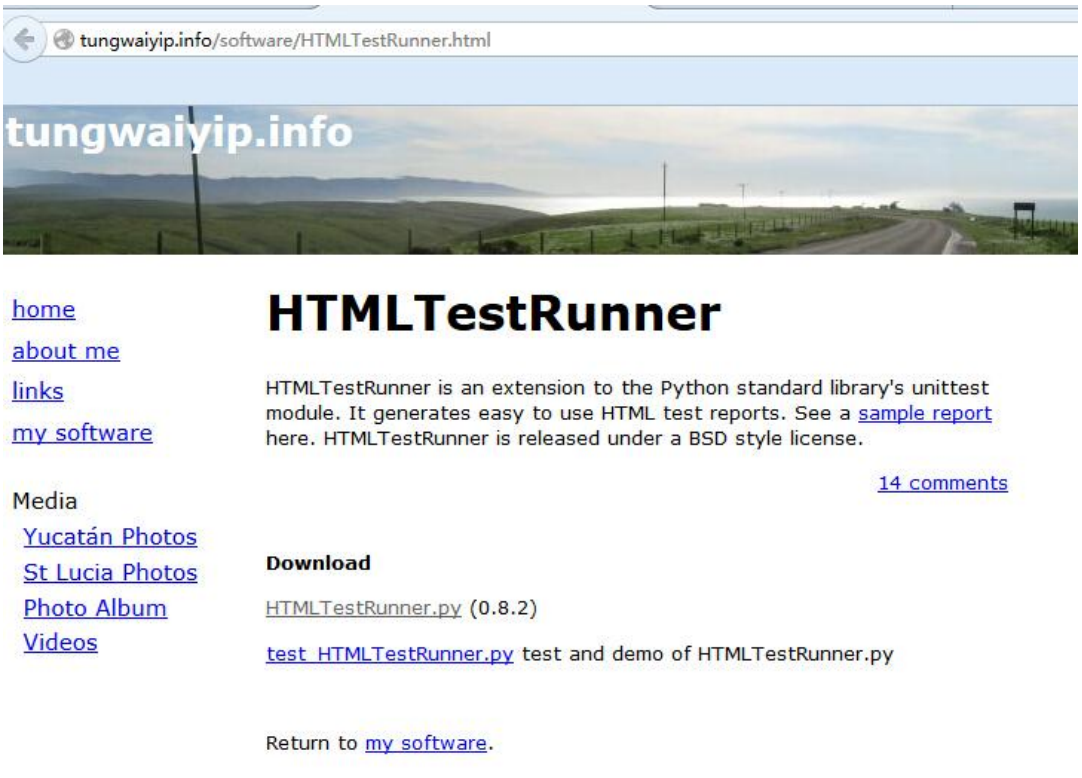
若未联网也可以下载 selenium for python 解压后放置在 Lib\site-pageages 目录下

```
C:\Python27\Scripts>pip install -U selenium
Downloading/unpacking selenium
  Running setup.py (path:c:\users\admini~1.pc-\appdata\local\temp\pip_build_Admi
nistrator\selenium\setup.py) egg_info for package selenium

Installing collected packages: selenium
  Running setup.py install for selenium

Successfully installed selenium
Cleaning up...
```

3.5: 下载并安装测试报告文件包

下载 HTMLTestRunner.py 放置在 Lib 目录下:

http://tungwaiyip.info/software/HTMLTestRunner.html



3.6: 下载并配置 openpyxl (操作 Excel 文件)

https://pypi.python.org/pypi/openpyxl/下载最新版的 openpyxl 包

PS:由于 xlwt 和 xlrd 不支持 excel2007 及更高版本，并且不支持对现有 excel 文件进行修改;

所以我们选用功能工强大的 openpyxl。

在 Dos 窗口中分别进入到包解压后的目录，执行 python setup.py install 即可。

3.7: 下载安装脚本开发工具。

(Note++，Eclipse Pydev，IDLE，Wing IDE，PythonWin，PythonWokrs，PyCharm 请根据个人喜好选择)

个人建议使用 JetBrains 公司的 PyCharm。

下载地址如下：http://www.jetbrains.com/pycharm/

注意事项：webdriver 在驱动 Firefox 浏览器时 在 FireFox 启动时会预装载十几 M 的配置文件，过程非常耗时。

为解决此问题，请同学们最好用英文版的 FireFox 并安装在默认路径下，安装完成后在 Dos 窗口里执行 firefox.exe –p 选择 default 用户后确定即可。

4、Webdriver API(python)

4.1: 根据 id/name/class/css_selector/xpath 定位元素

这是最基础的方法，给各位同学的建议是 id/name/class_name 优先使用 最后迫不得已使用 xpath。

示例网上有很多，哥们在这里就不班门弄斧了。

4.2: 在不同浏览器窗口间切换焦点

为了在新的浏览器窗口上操作或返回之前的浏览器窗口，我们必须根据 windowhandle 切换焦点。

```
#切换焦点到ImageView页面
def switch_to_imageview():
    oldwindowhandle=driver.current_window_handle
    handles=driver.window_handles
    for handle in handles:
        if handle!=oldwindowhandle:
            driver.switch_to.window(handle)

    time.sleep(1)

```

4.3: 等待页面元素加载完成后展示

```
    #等待指定的图片显示出来
    WebDriverWait(driver, 10).until(lambda the_driver: the_driver.find_element_by_id(canvas_id).is_displayed())
```

4.4: 若想操作 iframe 上的元素，必须切换焦点到 iframe 上

```
    #等待浮出层出现
    WebDriverWait(driver, 10).until(lambda the_driver: the_driver.find_element_by_id(session_manger_frame_id).is_displayed())
    #切换焦点到frame以选择session
    driver.switch_to.frame(session_manger_frame_id)
    driver.switch_to.frame(session_list_frame_id)
    try:
        for session in driver.find_element_by_id('dllSession').find_elements_by_tag_name('option'):
            if session.get_attribute('value')!=0:
                if session.text==session_name:
                    session.click()
                    time.sleep(1)
    except Exception:
        print 'exception occur when selecting session'
    #切换焦点到最里层frame以选择trainee用户
    driver.switch_to.frame(trainee_list_frame_id)
    #选择用户或用户组
    try:
        for role in driver.find_element_by_id('dllSelsect').find_elements_by_tag_name('option'):
            if role.text==None:
                pass
            elif role.text=='somovTrainee':
                role.click()
                time.sleep(1)
    except Exception:
        print 'exception occur when selecting role group'
```

4.5: 查找多个类似元素，根据 index 定位元素

```
#选择第一个case
def select_the_first_case():
    WebDriverWait(driver, 10).until(lambda the_driver: the_driver.find_element_by_xpath(up_session_xpath).is_enabled())
    caselist=driver.find_element_by_id(sessions_list_id).find_elements_by_class_name(case_class)
    caselist[0].click()
```

4.6: 模拟鼠标

```
888     #trainer做良性marker
889     def trainer_lable_benign_marker(canvas_index,Xoffset,Yoffset,Xsize,Ysize):
890         canvas_id='canvasDivContainer'+str(canvas_index)
891         canvas=driver.find_element_by_id(canvas_id)
892         #等待指定的图片显示出来
893         WebDriverWait(driver, 10).until(lambda the_driver: the_driver.find_element_by_id(canvas_id).is_displayed())
894         #在图片上右击 等待右键工具栏浮出层
895         ActionChains(driver).move_to_element_with_offset(canvas,Xoffset,Yoffset).context_click().perform()
896         WebDriverWait(driver, 10).until(lambda the_driver: the_driver.find_element_by_xpath(benign_marker_xpath).is_displayed())
897         #选中恶性marker
898         driver.find_element_by_xpath(benign_marker_xpath).click()
899         #在图片上标注恶性marker
900         ActionChains(driver).move_to_element_with_offset(canvas,Xoffset,Yoffset).click_and_hold().perform()
901         ActionChains(driver).move_by_offset(Xsize,Ysize).release().perform()
902         time.sleep(2)
```

4.7：Python 操作 Excel

个人建议使用 Openpyxl（xlrd 和 xlwt 不太好使哦），安装方式自己选吧 嘿嘿~~

4.8：Python 操作 sqlite 数据库

Python 自带的有 sqlite 数据库的方法只需要导入 sqlite 库即可

import sqlite3

4.9：python 通过 jdbc 方式连接 sql server 数据库

需要安装 pyodbc 库的

4.10：python 读取本地主机信息

自行安装鼓捣 psutil 库吧

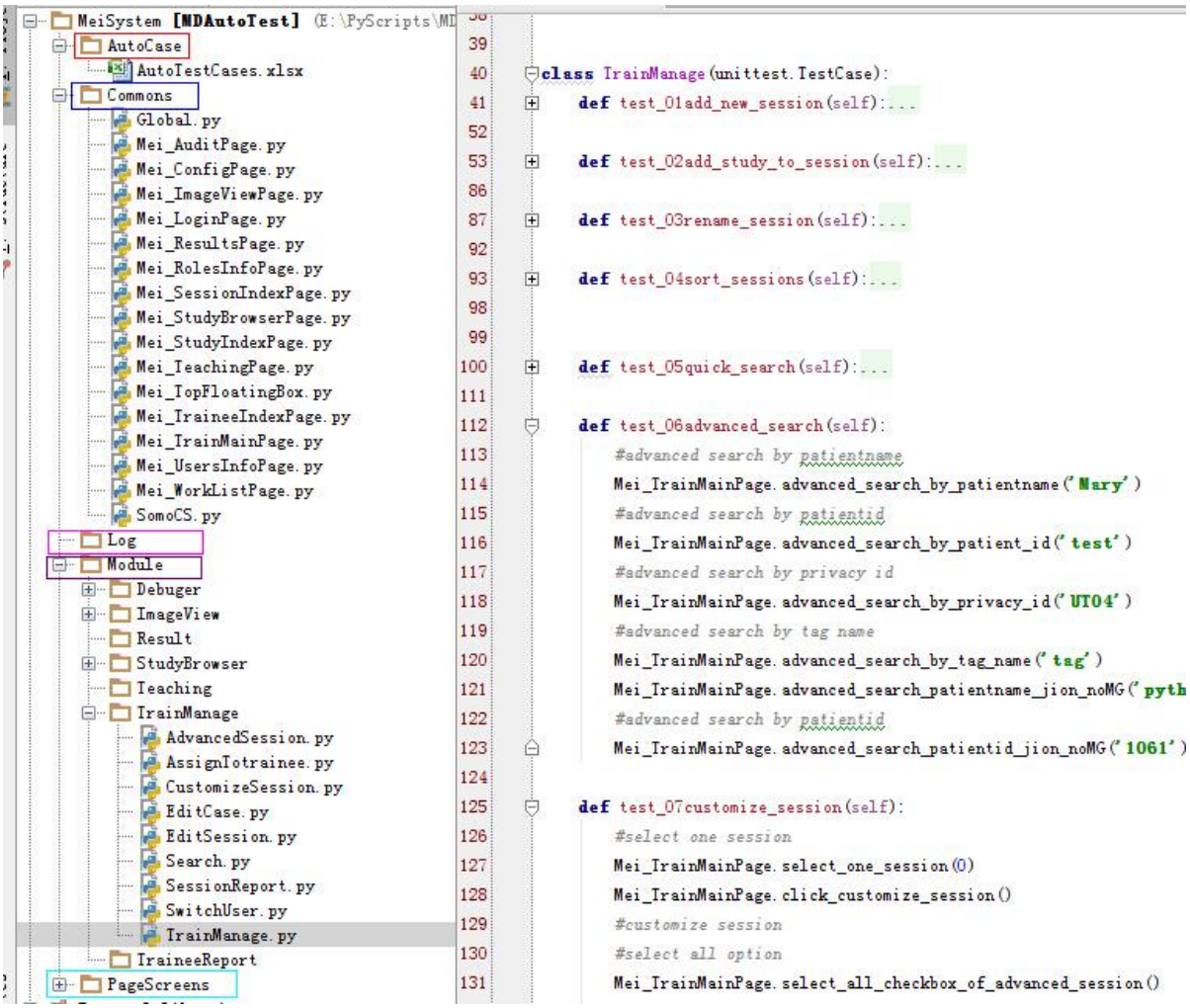
4.11：webdriver 执行 JS

```
1374    #点击refresh图标展开USB文件列表
1375    def click_refresh_to_expand_usblist():
1376
1377        try:
1378            driver.execute_script('javascript:refreshTree()')
1379            time.sleep(1)
1380        except Exception:
1381            print 'expand usbfile list error'
1382
```

5：PageObject 模式介绍

Page Object 模式是 Selenium 中的一种测试脚本的设计模式，将每一个页面设计为一个 Class 或基础库，其中包含页面中需要测试的元素（按钮，输入框，浮出层，图标，下拉框等）以及操作各种元素的方法。

通过以下截图和说明，逐步了解 Page Object 的意义。



Ps: 此为整个自动化项目的目录结构。

1: AutoCase 保存自动化用例，我们用 excel 管理。

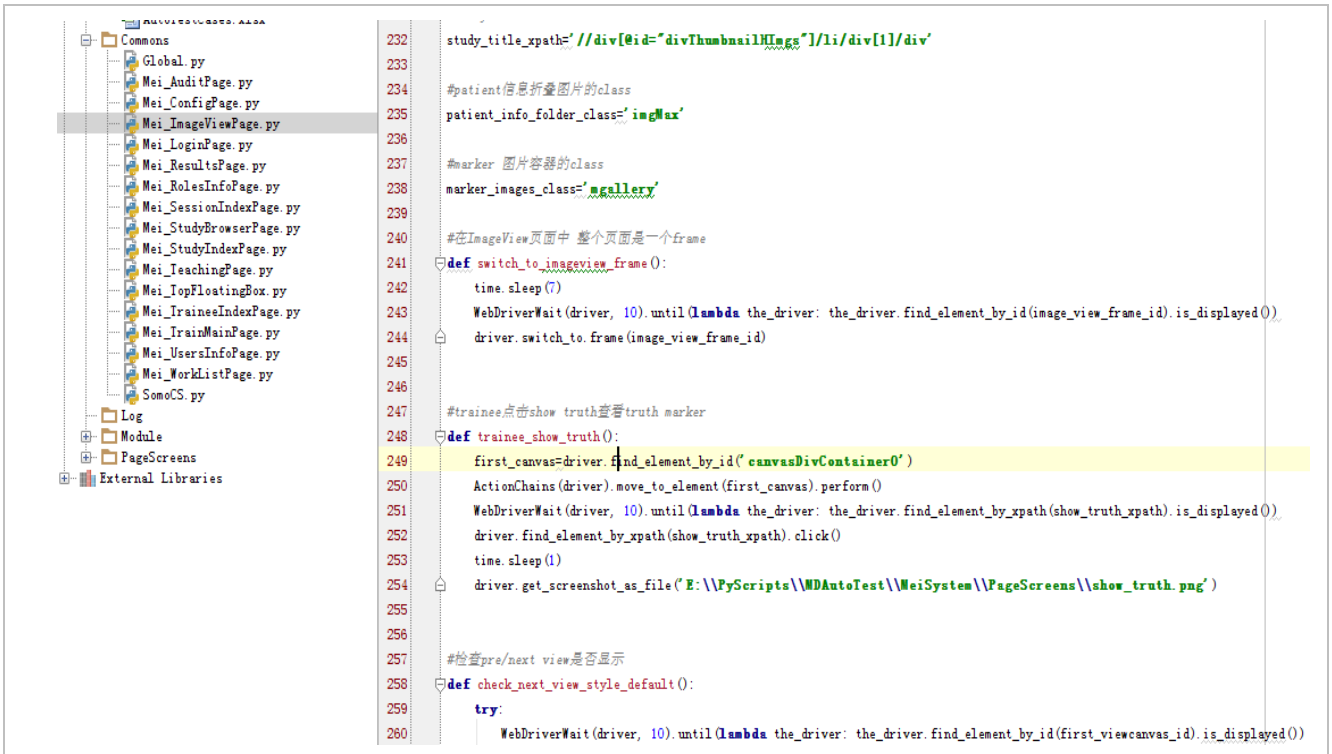
2: Commons 保存封装页面元素及其方法的基础库文件。

3: Log 保存自动化执行过程中的输出信息。

4: Module 用例层，保存组织好的脚本方法以模拟自动化用例。

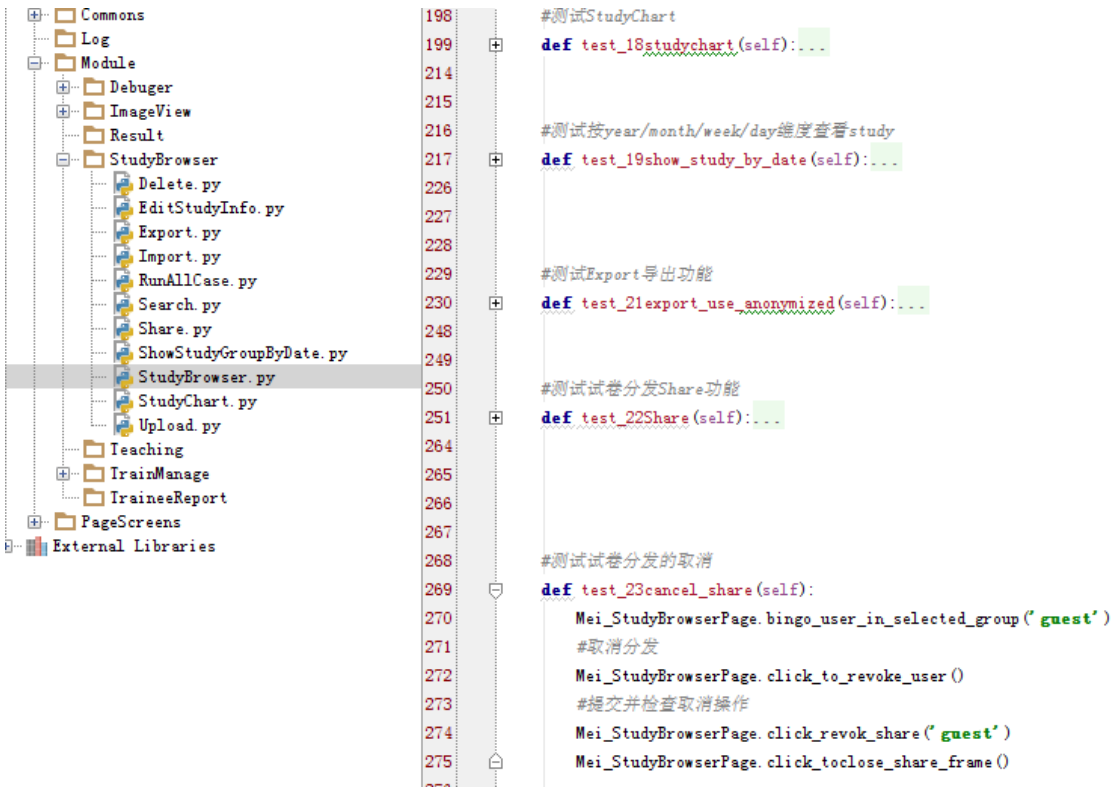
5: PageScreens 保存页面截图。

重点关注 Commons 和 Module，嘿嘿。。。



Page 基础库文件中集成了页面上所有的元素以及操作各种元素的方法。

UI 层是经常变化的，版本更新迭代过程中肯定会遇到功能点的变更，我们在维护脚本时，若元素发生了变更 我们只需要更改 Page 基础库里对应的元素，无需对用例层(Module)作任何改动；若新增了功能点，在此处添加相应的新增元素和其操作方法即可，用例层添加方法的调用即可。



用例层组织调用方法:

UI 改动，用例层改动非常小，这样组织脚本更便于维护。

脚本设计我们使用了 PageObject 模式，而运行脚本，我则选择了使用 PyUnit。

PyUnit 介绍如下:

在 Python 中进行单元测试需要用到自动单元测试框架 PyUnit，Python2.1 及其以后的版本都将 PyUnit 作为一个标准模块（即 python 的 unittest 模块），如果你很 out，那么你需要从 PyUnit 网站下载源码安装后才能使用。

(一)、Python 单元测试范例

测试最基本的原理是比较预期结果是否与实际执行结果相同，如果相同则测试成功，否则测试失败。为了更好地理解自动测试框架 PyUnit，下面会以对 Widget 类进行测试为例说明之:

#widget.py

#将要被测试的类 Widget

class Widget:

```
def __init__(self, size = (40, 40)):
self._size = size
def getSize(self):
return self._size
def resize(self, width, height):
if width < 0   or height < 0:
raise ValueError, "illegal size"
self._size = (width, height)
def dispose(self):
pass
```

(二)、测试用例 TestCase

软件测试中最基本的组成单元式测试用例（test case），PyUnit 使用 TestCase 类来表示测试用例，并要求所有用于执行测试的类都必须从该类继承。TestCase 子类实现的测试代码应该 是自包含的（self contained），即测试用例既可以单独运行，也可以和其它测试用例构成集合共同运行。TestCase 类中常用的函数或方法有:

setUp: 进行测试前的初始化工作。

tearDown: 执行测试后的清除工作。

failedinfo: 表示不成立打印信息 faliedinfo，为可选参数。

self.assertEqual(value1, value2, failedinfo): 会无条件的导致测试失败，不推荐使用。

self.assertTrue(, failedinfo): 断言 value1 == value2。

self.assertFalse(, failedinfo): 断言 value 为真。

self.assertRaises(ValueError, self.widget.resize, -1, -1): 断言肯定发生异常，如果没发生异常，则为测试失败。参数 1 为异常，参数 2 为抛出异常的调用对象，其余参数为传递给可调用对象的参数。

TestCase 在 PyUnit 测试框架中被视为测试单元的运行实体，Python 程序员可以通过

它派生自定义的测试过程与方法（测试单元），利用 Command 和 Composite 设计模式，多个 TestCase 还可以组合成测试用例集合。PyUnit 测试框架在运行一个测试用例时，TestCase 子类定义的 setUp()、runTest()和 tearDown()方法被依次执行，最简单的测试用例只需要覆盖 runTest()方法来执行特定的测试代码就可以了。

1、静态方法

一个测试用例只对软件模块中一个方法进行测试，采用覆盖 runTest()方法来构造测试用例，这在 PyUnit 中称之为静态方法，举例说明如下:

```
#static.py
from widget import Widget
import unittest
#执行测试的类
class WidgetTestCase(unittest.TestCase):
def runTest(self):
widget = Widget()
self.assertEqual(widget.getSize(), (40, 40))
#测试
if __name__ == "__main__":
testCase = WidgetTestCase()
testCase.runTest()
```

如果采用静态方法，Python 程序员就不得不为每个要测试的方法编写一个测试类，该类通过覆盖 runTest()方法来执行测试，并在每个测试类中生成一个待测试的对象，这样会非常繁琐与笨拙。

2、动态方法

鉴于静态方法的缺陷，PyUnit 提供了另一种高帅富的解决方法，即动态方法，只编写一个测试类来完成对整个软件模块的测试，这样对象的初始化工作可以在 setUp()方法中完成，而资源的释放则可以在 tearDown()方法中完成，举例说明如下:

```
#dynamic.py

from widget import Widget

import unittest

class WidgetTestCase(unittest.TestCase):

def setUp(self):

self.widget = Widget()

def tearDown(self):

self.widget.dispose()

self.widget = None

def testSize(self):

self.assertEqual(self.widget.getSize(), (40, 40))

def testResize(self):

self.widget.resize(100, 100)

self.assertEqual(self.widget.getSize(), (100, 100))
```

动态方法不再覆盖 runTest()方法，而是为测试类编写多个测试方法，按照惯例这些方法通常以 test 开头但这不是必须的，在创建 TestCase 子 类的实例时必须给出测试方法的名称来为 PyUnit 测试框架指明运行该测试用例时应该调用测试类中的哪些方法，这通常会结合测试用例集 TestSuite 一起使用。

三、测试用例集 TestSuite

完整的单元测试很少只执行一个测试用例，开发人员通常需要编写多个测试用例才能对某一软件功能进行比较完成的测试，这些相关的测试用例称为一个测试用例 集，在 PyUnit 中是用 TestSuite 类来表示的。PyUinit 测试框架允许 Python 程序员在单元测试代码中定义一个名为 suite()的全 局函数，并将其作为整个单元测试的入口，PyUnit 通过调用它来完成整个测试过程：

```
def suite():
```

```
suite = unittest.TestSuite()
suite.addTest(WidgetTestCase("testSize"))
suite.addTest(WidgetTestCase("testResize"))
return suite
```

也可以直接定义一个 TestSuite 的子类，并在其初始化方法__init__中完成所有测试用例的添加:

```
class WidgetTestSuite(unittest.TestSuite)
def __init__(self):
unittest.TestSuite.__init__(self, map(WidgetTestCase, ("testSize", "testResize")))
```

这样只需要在 suite()方法中返回该类的一个实例就可以了:

```
def suite():
return WidgetTestSuite()
```

在 PyUnit 测试框架中，TestSuite 类可以看成是 TestCase 类的一个容器，用来对多个测试用例进行组织，这样多个测试用例可以自动在一次 测试中全部完成。事实上，TestSuite 除了可以包含 TestCase 外，也可以包含 TestSuite，从而可以构成一个更庞大的测试用例集:

```
suite1 = mysuite1.TheTestSuite()
suite2 = mysuite2.TheTestSuite()
alltests = unittest.TestSuite((suite1, suite2))
```

四、实施测试 TestRunner

编写测试用例（TestCase）并将它们组织成测试用例集（TestSuite）的最终目的只有一个：实施测试并获得最终结果。PyUnit 使用 TestRunner 类作为测试用例的基本执行环境，来驱动整个单元测试过程。但是 Python 开发人员在进行单元测试时一般不直接使用 TestRunner 类，而是使用其子类 TextTestRunner 来完成测试，并将测试结果以文本方式显示出来。举例说明如下:

```
#text_runner.py
```

```
from widget import Widget

import unittest

#执行测试的类

class WidgetTestCase(unittest.TestCase):

def setUp(self):

self.widget = Widget()

def tearDown(self):

self.widget.dispose()

self.widget = None

def testSize(self):

self.assertEqual(self.widget.getSize(), (40, 40))

def testResize(self):

self.widget.resize(100, 100)

self.assertEqual(self.widget.getSize(), (100, 100))

#测试

if __name__ == "__main__":

#构造测试集

suite = unittest.TestSuite()

suite.addTest(WidgetTestCase("testSize"))

suite.addTest(WidgetTestCase("testResize"))

#执行测试

runner = unittest.TextTestRunner()

runner.run(suite)
```

使用如下命令执行该单元测试:

$python text_runner.py

默认情况下，TextTestRunner 将结果输出到 sys.stdout/sys.stderr 上，但是如果在创建 TextTestRunner 类实例时将一个文件对象传递给了构造函数，则输出结果将被重定向到该文件中。

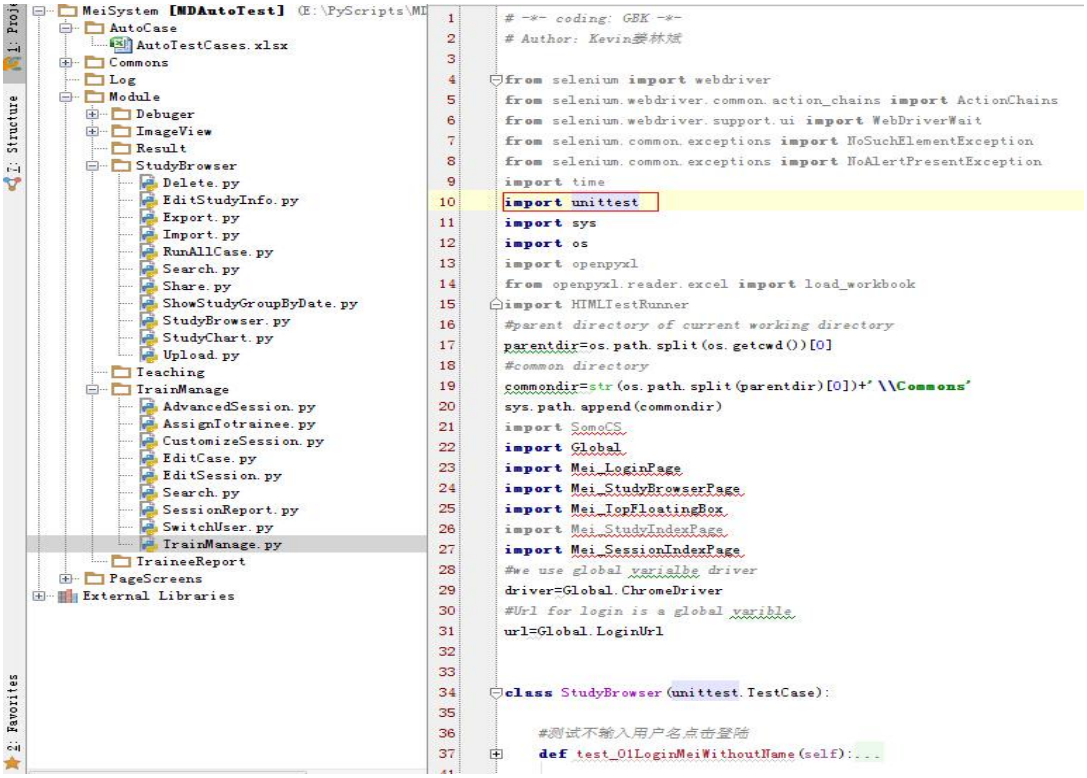
五、大道至简 main()

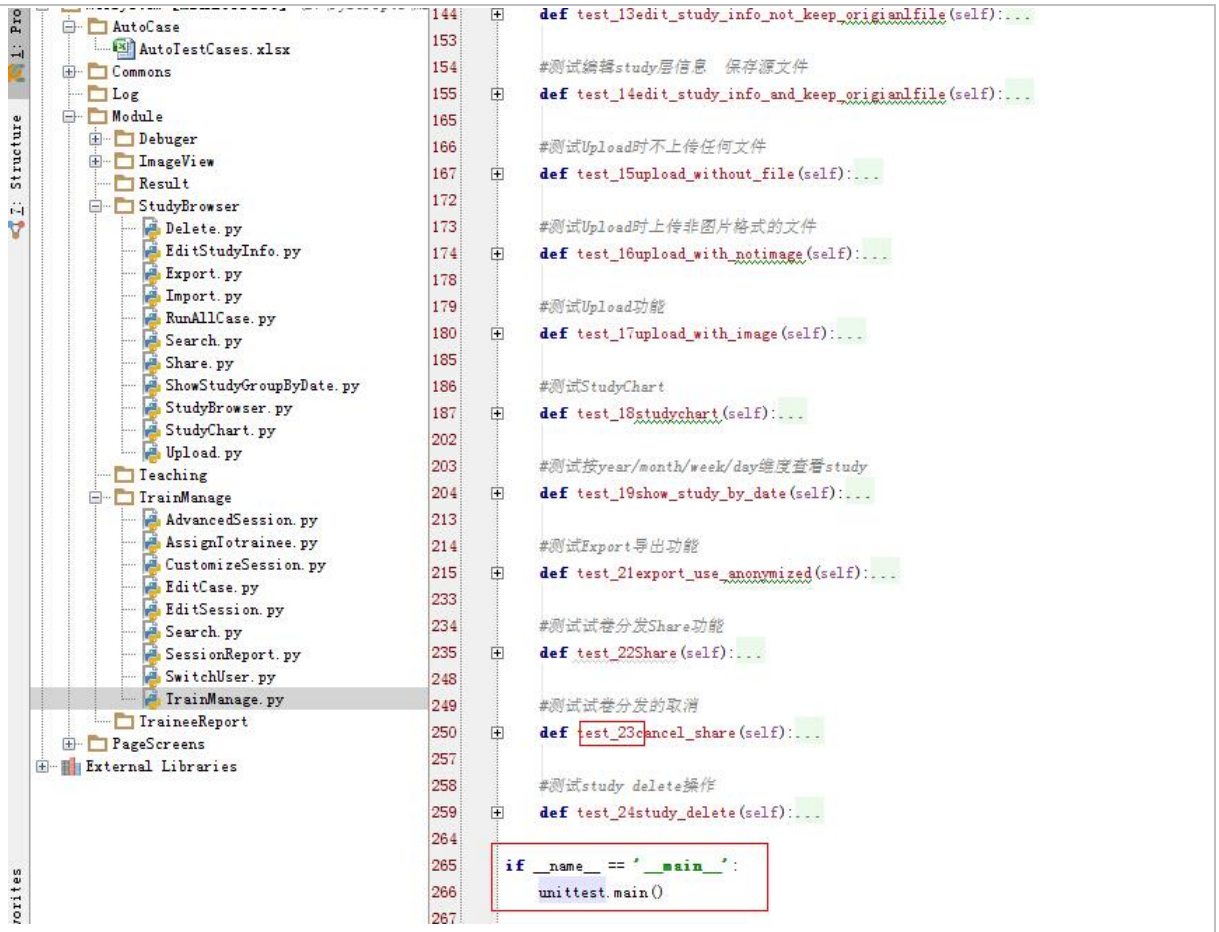
PyUnit 模块中定义了一个名为 main 的全局方法，使用它可以很方便地将一个单元测试模块变成可以直接运行的测试脚本，main()方法使用 TestLoader 类来搜索所有包含在该模块中的测试方法，并自动执行它们。如果 Python 程序员能够按照约定（以 test 开头）来命令所有的测试方 法，那么只需要在测试模块的最后加入如下几行代码即可:

if __name__ == "__main__":

unittest.main()

下面贴出已经调试通过的例子:





注意点:

1: 使用 Pyunit 需要先导入 unittest 模块

Import unittest

2: 测试方法必须以 test 开头

3: Pyunit 是以方法名字排序后的先后顺序来执行的，建议用例层方法的名字如下;test_00x<用户名>_<功能点名称>

使用 PyCharm 运行脚本后，其自动生成的测试报告如下:

脚本开发的规范性:

1: 页面元素唯一标识自解释(尽量可以从唯一标识的描述可看出是根据什么方法定位哪个元素)。

示例如下:

```
77      #note信息栏class属性
78      note_info_class='viewtitleInfo'
79
80      #note信息输入框
81      note_input_id='inputNotes'
82
83      #finding content栏刷新图标
84      refresh_finding_info_xpath='//a[@title="Refresh"]'
```

2: 基础方法加注释并自解释(若为分角色的方法，在方法名称中包含角色名称)。

示例如下:

```
284     #放大marker的附加图
285   + def large_marker_image():...
291
292     #查看marker的附加图片
293   + def view_and_select_marker_image(marker_image_index):...
317
318
319     #检查pre view默认时是否为灰色
320   + def check_pre_view_default():...
334
335
336     #检查默认layout
337   + def check_layout_default():...
```

3: 用例组织添加注释以解释用例的执行步骤

```
140     def test_08trainer_set_sixstep_case(self):
141         #customize case
142         Mei_IrainMainPage.expand_cases_of_session(0)
143         #select the case
144         Mei_IrainMainPage.select_the_first_case()
145         #set 6 step to show view
146         Mei_IrainMainPage.click_customize_session()
147         Mei_IrainMainPage.switch_to_customize_case()
148         Mei_IrainMainPage.first_step_images('RCC','LCC')
149         Mei_IrainMainPage.second_step_images('RMLO','LMLO')
150         #set enabled new report
151         Mei_IrainMainPage.set_enable_new_report()
152         Mei_IrainMainPage.submit_customize()
153
154     def test_09trainer_mark_truth_marker(self):
155         #select sessionC
156         Mei_IrainMainPage.select_one_session(0)
157         #expand SessionC
158         Mei_IrainMainPage.expand_cases_of_session(0)
159         #dobule click the first case
160         Mei_IrainMainPage.select_the_first_case()
```

4: 自动化用例尽量详细(目前自动化用例包括: 用例序号,功能点名称,操作步骤,检查点详述,检查点期望结果,测试结果)

1	1.1 studybrowser_main\ 测试时间：			测试者：	
2	用例序号	功能点名称	操作步骤	检查点详述	检查点期望结果
3	1	登陆系统(公共方法)	1)浏览器地址栏输入URL	当前URL的正确性	Current_Url当前URL地址
4			2)输入用户名	用户名输入框	当前页面存在用户名输入框
5			3)输入用户名对应的密码	密码输入框	当前页面存在密码输入框
6			4)点击登陆图标	登陆按钮标示	当前页面存在登陆图标
7	2	登陆成功后默认设置	无	URL跳转成功	成功跳转到指定URL地址
8	3	通过Storage跳转到StudiesBrowser总数展示页	5)点击Studybrowser图标	Studybrowser图标在当前页面存在	页面成功跳转
9			无	跳转后的URL检查	成功跳转到指定URL地址
10			无	跳转后Title检查	跳转后的窗口的title正确
11	4	跳转到StudiesList主页	6)点击"AllStudies"图标	"AllStudies"图标可点击	页面成功跳转
12			无	跳转后的URL检查	成功跳转到指定URL地址
13			无	im/export,up/download,share功能图标	登陆完成后，页面展示im/export,up/download,share图标
14	5	Studies统计报表生成(storage特有)	7)点击报表图标	报表图标可点击	点击报表图标后，弹出Studies统计报表浮出层
15			无	Studies报表浮出层检查	当前页面存在浮出层
16	6	按年/月/周/日统计Studies(storage特有)	8) 点击by Year图标	以年为单位的统计报表	生成以年为单位的统计报表
17			9)点击by Month图标	以月为单位的统计报表	生成以月为单位的统计报表
18			10)点击by Week图标	以周为单位的统计报表	生成以周为单位的统计报表
19			11)点击by Day图标	以日为单位的统计报表	生成以日为单位的统计报表
20			12)点击关闭统计浮出层的图标	报表浮出层成功关闭检查	Studies报表浮出层成功关闭

5: 每个脚本包含脚本开发者的联系方式(如邮箱/电话)

```
1  # -*- coding: GBK -*-
2  # Author: Kevin蒙林斌
3  #Phone:18258101665
4  #Email: linbin_jiang@md-technologies.com
```

程序猿最痛苦的两件事儿，一是看没有注释的代码；二是让自己添加注释。

哈哈 ~~ 哥们不觉得写注释是痛苦的(没有一丢丢儿恶心自己成全别人的成分哈)，无任何注释的代码 其维护成本是相当高的 读起来差不多就是 bull shit!

6、题外话：

1-----吐槽现实中的敏捷

敏捷软件开发价值观:

个体和互动　高于　流程和工具

工作的软件　高于　详尽的文档

客户合作　　高于　　合同谈判

响应变化　　高于　　遵循计划

敏捷的价值观是左侧高于右侧的，但是……并不代表右侧的这些不是必须的。

现实的敏捷型团队中很多都是下面这种情况:

有个体和互动　无　流程和工具

有工作的软件　无　任何文档

有扯淡的客户需求　无　合理的合同谈判

有响应变化　　无 任何项目计划

在敏捷团队里 很多时候 都会让哥们想到脚本里的“死循环” 有木有？！

在现实敏捷团队里，我想我们会情不自禁地深思以下问题（当然，我只是从 QA 的角度去想问题的）:

1: 测试人员何时介入？

2: 客户需求在团队里如何扩散并统一同步？

3: 新进员工如何快速融入这样的团队？

4: 产品的实施和维护如何进行？

5: 是我最想问的 也可能是团队 leader 要去考虑的，敏捷是否真的适合自己的团队？

有没有比敏捷开发更好模式去提高生产率和质量？

附加--敏捷宣言遵循的原则

1: 最重要的目标是通过持续不断地及早交付有价值的软件使客户满意。

2: 欣然面对需求变化，即使在开发后期也一样。为了客户的竞争优势，敏捷过程掌控变化。

3: 频繁地交付可工作的软件，相隔几周或一两个月，倾向于采取较短的周期。

4: 业务人员和开发人员必须相互合作，项目中的每一天都不例外。

5: 激发个体的斗志，以他们为核心搭建项目。提供所需的环境和支援，辅以信任，从而达成目标。

6: 不论团队内外，传递信息效果最好效率也最高的方式是面对面的交谈

7: 可工作的软件是进度的首要度量标准。

8: 敏捷过程倡导可持续开发，责任人，开发人员和用户要能够共同维持其步调稳定延续。

9: 坚持不懈地追求技术卓越和良好的设计，敏捷能力由此增强。

10: 以简洁为本，它是极力减少不必要工作量的艺术。

11：最好的架构，需求和设计出自自组织团队。

12：团队定期地反思如何能提高成效，并依此调整自身的行动。

最后 哥们只想说 任何事物，无论生活 感情 技术 流程 团队 适合自己的 才是最好的！

原创测试文章系列（三十九） 下篇精彩预览

- 高效地测评软件的用户体验
- Appium+Robotframework 实现手机应用的自动化测试
- 测试女巫之石头变宝石篇之三
- 手机应用程序测试策略规避
- 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

马上阅读