目 录

如何解决在使用敏捷过程中带来的技术挑战

作者：邵育亮

在企业中采用敏捷开发是一个非常具有挑战性的工作。敏捷不是一个软件，只需要点击 install 就可以安装完毕的。它需要一个企业从文化，技术以及组织结构等个个方面去接受敏捷带来的变化。我们这里来看一下企业在开发环境，自动化测试和持续集成等方面遇到的挑战。我们目前又是如何来解决这些问题的。

开发环境的建立

每个技术部门的经理，技术团队的负责人都希望能够降低构建开发环境的时间。但是我们还是发现我们的开发人员仍然在一遍又一遍地花费大量的时间来努力把项目变得更加有序。我认为缺少文档是我们花费大量时间来构建开发环境的重要原因。第二个重要的原因就是在这个构建的过程中，充斥着大量的人工的环节。那我们如何来应对这个挑战呢？对于文档的建立，我有一些个人的信条 - 简单，关注细节和自动化。简单是要让我们很方便创建，维护，查看和分发文档。我们使用 wiki 来管理我们在构建开发环境过程中产生的文档内容。每个 wiki 的页面都有一个所有人，然后会在每一个迭代中更新。关注细节是要让文档不会产生歧义。文档中的每一个细节都会关系到开发人员如何构建代码，以及如何与其他团队的集成。我在我们的 wiki 页面中抓取了一些信息如下。

- 一些列需要安装的软件：在我们的案例中，这包括 JDK，Eclipse 开发平台，Apache Ant，Apache Axis 和 SQL Server
- 每一个包都包括路径（网络磁盘/互联网/局域网/等等）以及获取资格。比如 apache ant，它的路径在我们的 SVN 代码库里面。
- 每一个包获取的系统和本地的变量需要在机器上配置好。比如说，ANT 需要设置 ANT_HOME 变量，axis 需要设置 AXIS2_HOME 变量。
- 第三方库的列表：包括所有第三方的 jar 包和 dll 文件，以及其他的一些文件。比如用来连接 SQLServer 的 JDBC 包和负责和 IBM WebsphereMQ 通信的第三方 Jar 包等等。

…

也许建立开发环境最重要的一方面就是自动化。自动化有很多好处 - 如果不考虑配置的问题，它可以显著的提高我们构建开发环境的效率。不过自动化也面临一些挑战：

1、缺乏对于机器的管理权限。系统管理员可能出于安全和公司政策的考虑，不会给予开发人员管理权限。这个会导致无法安装软件，设置环境变量和执行脚本。

2、需要和外部的组织合作：外部的组织会提供软件，提供权限等等

3、需要平衡现有的服务：许多大型的企业已经建立很多中间件（消息队列，ESB，应用服务接收器）。

上面说的这些挑战也不一定会经常发生，但是还是要提一下。开发人员需要的任何软件，无论他们从哪里获得都加入到源码控制中。确保所有的软件包都有一个统一的管理。这个让我们的开发人员可以很轻松的获取自己需要的软件而不需要在公司里到处询问。

自动化测试和持续集成

开发人员总有各种各样的理由不做自动化测试。我们总能听到这么些声音：

- “我有一款我喜欢的工具”- 这个可能是一个开发人员自己开发的工具，或者是来自于一个开源组织的。这个工具可能也会有缺陷，但是开发人员不愿意承认。
- “没有测试数据”-特别是我们需要从不同的系统去收集数据的时候

工具的问题可以有很多解决的办法。你可以告诉他们使用一款标准的测试工具类似 Junit 或者是 Nunit 的好处。支出这些工具可以很好的和测试脚本集成并且持续稳定的运行。在一个大型的企业里面，不可能所有的开发人员只用一个工具，应该是有一个工具系列。我们要做的就是提供统一的测试脚本，比如脚本编译成 Junit 的 class，执行并生成报告并邮件通知相关人员。当开发人员从 SVN 代码库中下载一份代码的拷贝的时候，测试代码也会被同时下载下来，开发人员在工作的过程中会不断的增加测试用例脚本。我们团队有一个统一的目录来存放测试脚本和测试集。在代码审查的过程中，我们要保证除了产品的代码 review 之外，测试代码也会被 review。作为代码审查的一部分，测试代码也需要被重构。测试代码中的数据不应该是 hard code，而是需要可配置的。

在 SOA 的项目中，集成了应用服务器，过去的服务，数据源，打包完毕的应用和测试脚本等等。这些测试脚本可以通过 Mock 的技术来实现底层的集成测试。最终你希望你的测试脚本是针对真实的集成环境。很多时候，测试数据分布在不同的系统上是个很大的问题。缺乏数据，特别是那些需要从不同系统获取的数据是导致我们无法充分进行自动化测试的问题。我们可会议这样来解决这个情况，我们可以使用本地的数据，开发人员从不同的数据源获取数据并生成本地的 copy。这个听上去是一个完全手工的操作，但是如果数据的是新的，或者是 ETL 的任务，脚本可以让这些工作变得简单。我们可以通过各种脚本在不同的数据源中获取数据。这样的话，我们的测试脚本和我们的数据可以分离，即使我们要更换我们的数据源，也不会影响到我们的测试脚本了。

定义我们为什么这么做

在一个大型的企业中去指定一个开发，或者测试的规范是非常棘手的。代码可能在开发环境运行的很好，但是不能在测试环境中运行。这个可能是因为网络，安全，系统稳定性等各个因素导致的，或者是因为更高质量要求的测试以及终端用户。如果我们希望在迁移代码的过程中的影响最小，我们的规范应该包括：

- 构建一个良好的测试套件，其中更多的是测试数据可配置，并且可以反复的执行。
- 把测试集加入到持续集成的流程中。例如我们在我们的 CruiseControl 的 ant 脚本中，增加了执行 Junit 单元测试的脚本。
- 自动化测试包括功能和性能
- 代码审查人员不单单包括开发很远，还需要包括基础架构的成员，例如数据库管理员，系统运维人员等等。

【淘测试专栏】针对数据进行自动化测试

作者：小稚

【摘要】

针对 web 的测试，如果做 UI 层的自动化测试，需要等界面都已经设计好且没有大的改动时才能编写脚本，写 UI 脚本又比较耗时，在进行项目功能测试的时候常常没有足够的时间来写界面自动化脚本，那么有什么方式能将自动化测试提前化又能方便的测试上层功能呢？

也许你要说 service 层（Model）和 action 层（controller）的接口测试，这两项确实能将测试提前化，但是就我个人的测试体会来说，service 层一般只提供一些底层的服务，里面包含的业务逻辑不是很多，经常会发现测试完后根本覆盖不了多少功能，不能减少功能测试的周期；而 action 层测试由于环境等因素比较复杂，经常会出现投入产出比不理想的情况。基于以上的一些考虑，针对一些特殊场景的 http 测试，我们引入了针对数据的自动化测试方法。

【使用场景】

针对都是通过发送 http 请求，返回 html 页面或者是直接返回数据的场景。

校验点：返回数据的正确性。

【实现】

一般的 http 访问返回页面的类型都是以.htm、.jsp、.asp 等结尾，后面跟上各种参数，例如：在浏览器中输入下面这个网址：

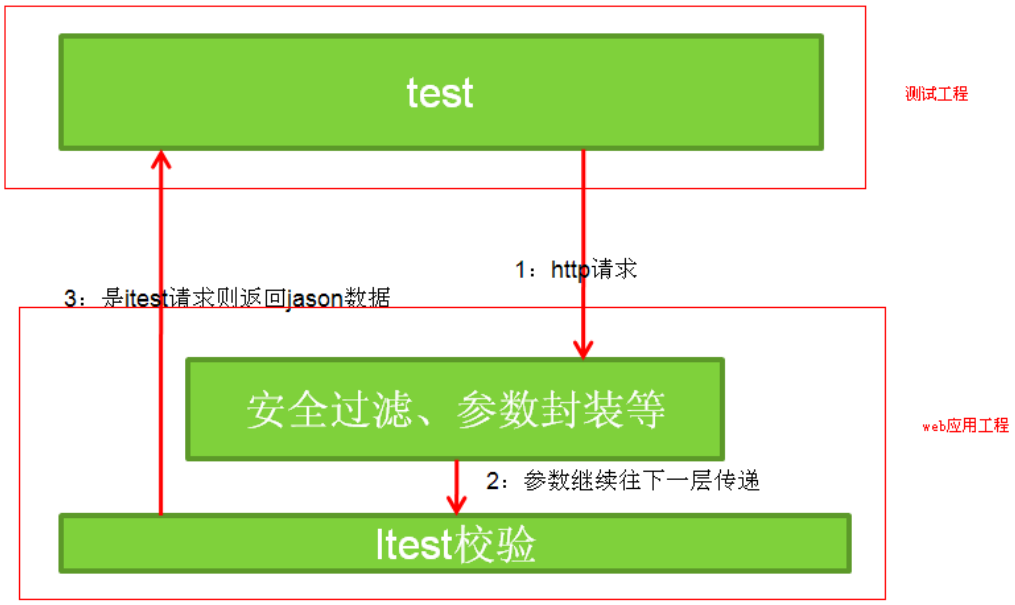
http://tradecard.wangwang.taobao.com/tradecard/buyer/chatCard.htm?loginId=cntaobaoclientautotest01&uid=cntaobaoschiffer_yu&gid=9409512100&itemsId= 返回 html 页面

针对这样的请求和响应，我们设想，能不能发送一个特殊的请求，不返回 html 页面，而是返回页面上展示的那些数据呢？例如返回 json 格式的字符串等不依赖页面展示的数据类型。

接下来，我们就按照这种想法继续往下设计。

1、请求和响应之间做一个约定，如果 http 请求中以.do（实际可以依项目情况而定）结尾，后面跟上各种参数（与原 http 请求一致），最后加上一个我们自定义的 itest（这里的 itest 仅仅是个标识）参数，服务端遇到这个请求之后认为是一个测试请求，不返回页面，而是以 json 数据格式返回页面需要展示的数据。

2、接收服务器返回的数据，将返回的数据与预期的数据进行对比，校验结果的正确性。



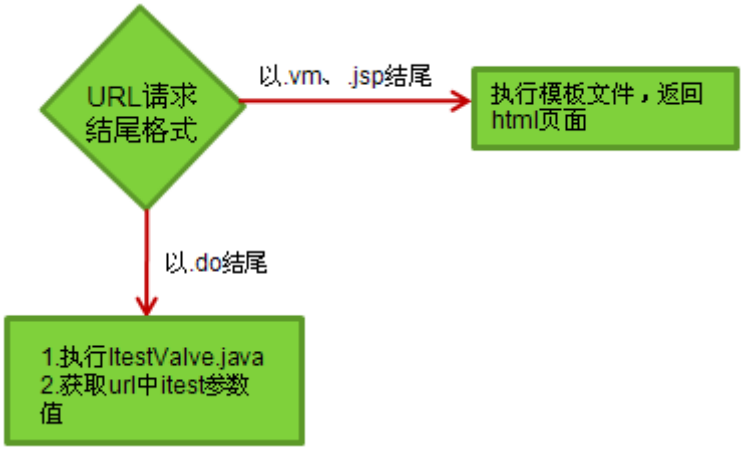
服务端代码举例：

1、配置一个阀门，判断请求是以什么格式结尾。

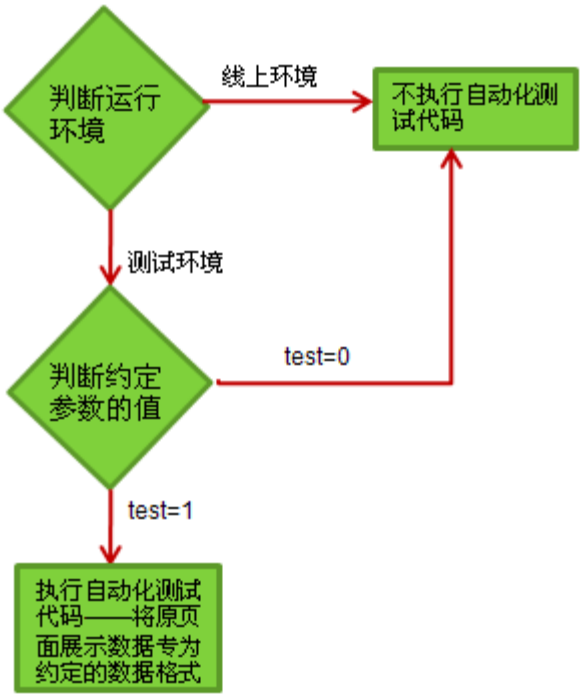
例如下面这段代码：

```
<choose>
    <when>
        <!-- 执行带模板的screen，默认有layout。 -->
        <pl-conditions:target-extension-condition extension="null, vm, jsp" />
        <performAction />
        <performTemplateScreen />
        <renderTemplate />
    </when>
    <when>
        <!-- 执行不带模板的screen，默认无layout。 -->
        <pl-conditions:target-extension-condition extension="do" />
        <performAction />
        <performScreen />
        <!-- 用于执行Itest自动测试的Valve -->
        <valve class="[redacted]ItestValve" p:cando="$![redacted]itest.cando}"/>
    </when>
    <otherwise>
        <!-- 将控制交还给servlet engine。 -->
        <exit />
    </otherwise>
</choose>
```

判断如果是以.do 格式结尾，则执行不带模板的 screen，即不返回 html 页面，而是执行 ItestValve.java 这个类；同时获取 http 请求中的 itest 参数的值。



2、ItestValve.java 中判断程序运行环境是测试环境还是线上环境，测试环境（这个参数可以根据自己配置）则执行自动化测试，线上环境则不执行自动化测试；还需要判断 http 请求中 itest 的值是否为 1，itest=1 则执行自动化测试，不为 1 则不执行自动化测试。执行自动化测试：将原有应用 context 中的 map 数据转换为 json 返回。



代码实现如下

```
String itest = request.getParameter("itest");

// 不执行自动化测试
if (StringUtil.isBlank(itest) || !"1".equals(itest.trim()) || !"1".equals(cando.trim())) {
    pipelineContext.invokeNext();
    return;
}

// 执行自动化测试
ObjectMapper om = new ObjectMapper();
StringWriter sw = new StringWriter();
try {

    // 将context中的map转换为JSON，返回给客户端。
    TurbineRunDataInternal rundata = (TurbineRunDataInternal) getTurbineRunData(request);
    PullableMappedContext pmc = (PullableMappedContext) rundata.getContext();
    om.writeValue(sw, pmc.getMap());
} catch (Exception e) {

    // 如果发生异常，则将异常返回。
    om.writeValue(sw, ExceptionUtil.getStackTrace(e));
} finally {

    // 返回JSON数据
    response.getWriter().write(sw.toString());
    sw.close();
}
```

测试代码举例：

1、拼接测试 URL。

String url=

"http://tradecard.wangwang.daily.taobao.net:8080/tradecard/buyer/cartItemList.do?

loginId=cntaobaotbtestuser250&uid=cntaobaoclienttestcauto1&gid=&itemsId=1604398968,1500003926799&itest=1"；该请求以.do 格式结尾，最后一个参数为 itest，其值为 1.实际测试代码中，url 中的参数值可以设置成可配置的，将代码与测试数据分离。

2、发送 http 请求并获取返回字符串。

HttpRequest interfaceHttpRequest = new HttpRequest("get", url, "",false, false);

ResponseInfo response = interfaceHttpRequest.call();

String s=response.responseBody.toString();

返回的结果中可以看出，包含了很多信息，其中有些数据界面上是不展现的，我们只需要提取那些页面上展现的数据进行校验。

{"itemsId":"1604398968,1500003926799","opp":{"parent":
{"parent":null,"nick":"clienttestcautol","subAccount":false,"site":"cntaobao","nickWithSite":"cntaobaoclienttestcautol","urlNic
{"parent":
{"parent":null,"nick":"tbtestuser250","subAccount":false,"site":"cntaobao","nickWithSite":"cntaobaotbtestuser250","urlNick":"tb
[{"id":1604398968,"quantity":0,"title":"小稚的测试哦。不要删了","price":1212.0,"imageUrl":"i4/T1LWtcXgRdXXbOhXk._111116.jpg"}]}

3、校验返回的字符串

校验方法与普通的测试校验方法基本一致，将返回的字符串于预期设定的字符串进行对比，一致则表示测试通过，不一致则表示测试失败。

例如上面举例的测试 http 请求中包含两个 itemid，其中一个宝贝不属于双方，预期结果是只显示一个宝贝。

校验方法：首先校验是否包含宝贝 A，其次校验是否不包含宝贝 B。

assertThat(s, containsString("title\":\"小稚的测试哦。不要删了")); //校验第一个宝贝名称

assertThat(s, containsString("price\":1212.0")); //校验第一个宝贝价格

assertThat(s.indexOf("title\":\"小稚测试哈"), is(-1)); //校验第二个宝贝信息不显示

【实践】

该方法已经在一淘的旺旺发展部针对 web 应用的测试中普遍运用到。在项目开发过程中进行测试代码的编写，功能测试的时候，开始用测试脚本辅助功能测试，并负责每次部署后的功能回归。

【优势】

1、针对 http 请求返回页面的服务测试起来很方便，测试环境搭建简单，测试代码简洁。

2、较之 UI 自动化，该方法能将自动化测试提前，缩短项目周期。在还没提交测试之前，先写好测试脚本；UI 层的调整不会影响脚本的执行，稳定性好。

3、较之 Jmeter 工具，能动态的准备测试数据和校验多种类型的返回值（如实体类）。

4、该测试接近功能测试，能覆盖大部分功能测试用例，节省功能测试时间。在机器上部署每日回归测试，可以监控应用的可用性。

【扩展】

目前该方法应用场景较局限--暂时只适用于单纯的 http 请求返回页面或者是数据的应用。针对表单（action 等）和消息推送（notify）等应用，还在继续研究中。

【总结】

不同应用场景可以选择不同测试方法，每种方法都有他的优点和局限性，针对数据进行自动化测试的方法，优点和缺点都在上面列出来了，有兴趣的同学不妨一试，分享下自己的使用心得吧。

别再纠结于那些自动化测试不得不面对的现实

作者：刘毅

前言&摘要

工作中总难免遇到一些不想见到的问题，但是遇到问题总需要去解决。解决问题的时候我们不提倡一条路走到黑，但是也绝不鼓励“朝三暮四”、“朝秦暮楚”等各种浅尝辄止。就像我们在做自动化测试的过程中，有些问题是始终无法回避的，如：测试工具缺陷、测试数据使用难以及开发人力投入较多等诸多妖魔鬼怪。我们若不去逐个斗过，也不知道自己战斗力到底有多强，最终也无法取得真经、修成正果。

关键字：自动化测试 QTP SELENIUM 测试数据 测试工具

摆一摆自动化测试中的问题

大约前几天，一位同事为了 Selenium 处理模态窗口问题而纠结，而她之前是常年与 QTP 为伍的，她悻悻地说：这破玩意，以前用 QTP 的时候根本就不会存在这个问题！我奸笑一声，对她狠狠地说：既然 QTP 那么好用，那你之前还不好好写你的脚本，现在领导觉得 QTP 在你的系统可能不适用了才开始推行 Selenium，你还抱怨……那你得怨恨到什么时候呢？作为一个习惯了我“叫兽”风格的同事，她对我给予的挖苦不以为意，嘿嘿一笑继续忙她的了。当然，她最终搞定了这个问题，自然地，欢呼雀跃一下也是在所难免的。我后来跟她说：其实你之前只要多花一点时间把你的 QTP 开发质量提高一个档次，现在也不会有人逼着你花时间来解决这个问题，而且接下来前面还有什么问题等着你谁也不知道；现在是花很多时间，以前也要花很多时间，什么时候花都一样，自动化测试在我们这里绝不是应付了事，所以，我们始终都得面对开发投入这个现实！而且，早一点投入，产出效益也会更加明显一些。

再往前几个月，有一个开发经理向我们组同事咨询 QTP 在自动化测试中的使用情况，可碰巧被咨询的这位同事是个 QTP 盲。出于对技术仔的仰慕，我主动找到他，问他想了解一些什么东西，他说他想在集成环境中用 QTP 做集成测试。我就问：你们的单元测试做的咋样嘛？他说：还不错啊，用 JUnit 尝试了一下，不过 JUnit 测试完了还是无法确保移交的版本质量和可测性。我又问：你们尝试过 Selenium 么？那玩意是开源的，对你们来说应该很好用，如果用 JAVA

来做，估计你们单元测试代码可以复用不少呢。他回了一句：测试数据不好弄啊，数据又少又复杂，不知道 QTP 在这方面有没有什么优势？我就瞬间丢失了作为一个粉丝应有的姿态，嘟囔了一句：优势个屁，如果想绕过测试数据这关来做你的集成测试，你趁早还是别做了，早点移交给我们，让我们多点系统测试时间吧。当然这个我不敢明说，只好书面的回了一句：不管我们用什么工具，测试数据的使用、管理都是我们不得不面对的问题，这是测试的核心要素，所以没有什么 UI 工具能跨越业务逻辑来使用测试数据。

再往前一两年的样子，部门里有个 QTP 技术仔同事私下里和我探讨，他对 QTP 运行环境要求高和运行速度慢、对象经常莫名其妙不识别等等问题深恶痛绝。他问我：据说 LR 运行不是基于页面的，你懂吗？我说：略懂，LR 是基于协议的，靠模拟客户端请求去执行测试的。他立马来了劲，兴奋地说：那就是不受页面对象变化的影响咯？如果这样，你觉得我们如果用 LR 去写脚本做回归测试会不会很爽？至少速度快，受环境影响非常小。我一身冷汗，心想别让我又造了孽，误导了别人吧，赶紧劝阻：千万别，要是这样，WEB 页面最基本的 JS 控制项都被可以忽略了，测试结果根本不可信啊。看他将信将疑，我赶快找了几份生产缺陷事件报表发给他看，后来他研究了很久，好歹总算把这个想法给否定了。同样，我们可以看得出，无论技术是否过硬，测试工具本身的弱点是无法回避的，但是既然被称为工程师，我们始终要面对这些问题，并且必须要尝试解决它们。否则，就算是换一种测试工具那又能如何，这世界上存在完美的东西么？

该如何面对自动化测试的学习

其实稍微想一下我们就知道，自动化测试的学习和研究也是我们不得不面对的问题之一。《师说》有云：“人非生而知之者，孰能无惑？”所以我们在自动化测试学习或者实施的过程中也难免会遇到很多不可预知的问题，遇到问题的时候是该“求鱼”还是该“求渔”，是该自力更生还是求助于人，是该谷歌还是谷歌，工具和技术应该学到什么水平……这些都是应该事先弄清楚的。

平常闲逛的时候，经常在坛子里面看到有人在那里喊：“来个高手来教我 QTP 啊”，“广州有自动化测试的高手吗？带带我吧”——哦哟，拜托你们不要再卖萌了啦，人家会受不了的啦！在我看来这基本上都是不肯面对现实，不愿意主动去学习的表现，到坛子里只是找个倾诉的对象或者说精神的寄托而已。因为你真的不知道哪位菩萨心肠的大师会去主动找到你，给你传道授业解惑。如果

自己都没有仔细的去动手尝试，没有去找到自己的困惑点在哪里，再高的高手也无法三两句话能给你解释清楚，你还真以为可以随随便便移植一下记忆么？就像一个虔诚的信徒在哪里求上帝：上帝啊，求求你，让我中六合彩吧！上帝回答他：那你（TMD，语气助词）也得先买注彩票先！道理很简单：若要得到别人帮助，必须先保证在求助于别人之前自己已经尽了最大的努力了；不愿意一点点把基础打牢的人是浮躁的人，若无神奇因缘，指望别人将他们的问题一扫而光基本是不可能的事情。

我熟悉的测试工具不多，但是也了解不少，我比较讨厌比较这这那那的工具，在抵制工具至上论的同时，我也不介意反复再三地宣扬我的观点：我们是做测试的，但绝大多数不是做测试工具的测试的，所以没必要为了工具的好坏去争论；我们以我们的被测系统为核心，我们以满足自己的测试需求为根本目标，一切工具都只是整个测试过程中的一些手段而已，没有什么事情是以手段为核心而对根本目标置之不理的。我们部门有位对 WEB 开源测试工具比较熟悉的同事，他自己出了一本关于 Selenium 实践经验的书，据说口碑还不错，只可惜他在封面上就开始痛批 QTP 的种种不是，我看了之后感觉五味杂陈，不知该以之为荣还是该为之伤心。之所以该伤心，并不单单是因为他对 QTP 工具本身的贬低，因为他这么说恰恰说明我可能在 QTP 的使用上比他熟悉得多。我主要是对这种妄下结论，排斥性太强的看法与做法很不以为然，说难听一点，简直是死脑筋嘛，这不是我们学习自动化测试工具与方法所应有的态度。我觉得要想全面学习自动化测试，一定要抛开测试工具的门户之见，否则很容易陷入偏执的僵局，真的会被一叶障目。

一个人如果对自动化测试若只是初窥门径或者尚未入门则罢了，若是要熟练掌握一种测试工具的使用，那就该从基本的功能到框架的搭建都能了然于胸。熟练之后，如果有精力的话，就可以去深入研究和分析自动化测试抛开形形色色的工具之后又该是什么一种意识形态，有什么规律和章法。若非如此，我们对自动化测试的认识始终停留在对一种测试工具的理解上，而不是对测试的理解上。这样，你把 QTP 的录制、参数化叫做你的测试框架，把 Selenium 使用 JUnit 的框架叫做你的测试框架，把我们建设起来的自动化测试管理平台工具叫做你的测试框架。可惜对真正帮助组织我们的自动化测试开发、运行，优化我们的自动化测试开发、运行的一套方法没有去摸索，或者没有将之与我们的系统测试结合起来

落地执行，更没有去做方法论上的提炼升华与实例化的开发。如此这般，我们就错过了很多凌驾于测试工具本身的，有价值的、有趣的一些东西，而得到这部分东西，哪怕只是其中的一部分，我们就会明白，只要是做测试，核心的东西还是操作描述、测试数据、预期结果那点东西；知道这一点，就会知道用什么工具来做自动化，差别只是在手段上而已。真正精通自动化测试的人，其实并不见得他们对测试工具本身使用或者编码的技术是最强的人，但他们至少是懂得如何把“测试”和“自动化”这两个关键字有机结合的一些人；在他们看来，在自动化实施过程中，规划和需求要比技术和工具本身重要得多。

可能有人会说：先莫唱高调，你觉得我们该如何学习和研究呢？我的回答是：很简单，就是把自己实践得到的经验加上别人实践得到的经验揉合一下，其实总结起来有三点（当然如果你愿意的话，你可以去分成十三点、二百五十小点去说）：

● 不停的实践：使用一两种工具在不同类型、不同架构、不同行业的系统上去实践。真能用心学习和实践，一年半载便足够了，至少工具驾驭上应该不输现在 90%的人，但像 QTP 的使用，能够坚持数年录制、参数化而不改变的人，不在我们讨论的范围之内。我见到过有位仁兄，09 年的时候还在在论坛询问一些很简单很基础的 QTP 工具使用问题，10 年就已经自己搞了一些很有技术含量的专题讲座了，学习速度让我很惊讶也很佩服。这都是他通过一点点摸索尝试获得的成果，相信将来更大的成就对他来说也不会很远。

● 多听多看：听别人如何说，看别人如何做。除了要多于自己身边的同事多学习沟通之外，平时抽空多到网上去看看别人的说法和做法，结合自己在实践中的感受和困惑进行思考总结。拿句套话来说，这是一个“去伪存真”的过程，面对海量的信息，如果缺了有效的鉴别，很可能就会被别人的观点所迷惑或者造成更多的困惑。其实无论作者观点的对与错、是否有价值并不在于他这篇文章本身，我们只有认真地读了、想了、实践了，有所斩获了，才能不枉作者码字的一番心血。

● 看书不如看帮助文档：无论是谁写的书，写的什么书，书中所讲述的观点都是依照作者的知识、阅历去陈述的。而最可悲的是有些作者著书的动机可能会不单纯，或为名、或为利，而知识共享与传承神马的对他们来说都是次要的。这样一来，书中可能会充斥着“绝对”、“一定”这类字眼，将一些不成熟甚至是错误的观点灌输给读者。时过境迁，作者未必不知道当初自己错了或者有疏漏，

但是鲜有人肯出面勘误澄清，所以说：读书有风险，购买须谨慎。别的行业我不敢说，但是关于自动化测试，我见过这样的工具说明书，好几本，读起来感觉就是花几十块钱雇了个蹩脚的翻译，把英文的帮助文档给丢三落四的翻译了一通。至于英文的著述如何，我就没再研究了，总之学的过程中若想看参考说明书，我还是建议配上一本词典，去看帮助文档吧。如果刚入门就想凭借一本书变精通，那么观念里面有些东西可能以后很难改变了，无论它是对的还是错的。

例行总结

最后，我想说句话与大家共勉：不要盲目崇拜什么权威、什么技术专家，只要能够勇于面对你本该面对的问题，并且努力解决它，很快你就是专家。当然，你这个专家与其他专家一样，都是可以被很快超越的，所以也不要迷恋自己专家的头衔；继续虚心学习，方能立于不败之地。这些都是很俗套、很简单的道理，看得开，一切都是风轻云谈、海阔天空；看不开，则有可能走火入魔、害己害人！

武侠崇尚的是：天下武功，唯快不破！同样，在自动化测试领域中，要想做高手，那么你要快快地工作，快快地遇到问题，快快地解决问题，快快地进步吧；然后快快地被赏识，快快地涨薪升职吧。

性能测试前的调查工作

作者：思齐

客户端与服务器是通过什么样的协议通信的？软件的网络拓展什么样的？各个服务器的配置是什么样、宽带是多少？你选取的业务是系统处理的关键业务吗？该关键业务每天处理的数据量可能有多大？预计日峰值处理是多少？预计月峰值处理是多少？你算出来的场景启动方式和持续时间是怎么来的？你的脚本构成方式是什么样的，是否在关键点的地方进行了判断？脚本里的数据是否真实的反应了用户的数据？你的脚本排除了 session 部分，并保证全是 Post 请求吗，还是想测试用户第 1 次访问时的性能？

如果我一口气连续地问这么多问题，你能有理有据地回答出来吗？如果这些问题不能有理有据地回答出来，那么你度量的性能测试结果是可信的吗？

答案是我觉得不能。这就是为什么我们大多数做出的性能测试结果，总是令人怀疑的原因。当然我也很怀疑曾经我所做出来的性能测试结果，因为，上面所提到过的问题，在我做过的性能工作中我都不能回答出来。以前的性能测试，我也只能说仅仅是例行公事的做了，但做过之后我是心虚的。

我现在坚持认为，在做性能测试之前，我们应该像福尔摩斯一样，四处侦探调查一下性能测试工作的前提条件和环境。从网上搜集的信息来看，也许我们需要做下面几方面的调查。

1、测试需求可信性调查

什么是可信的测试需求呢，前辈 jackie 提出了以下观点，在下深以为然，在此仅简单引用，如需了解其理论依据，请前往地址（http://www.cnblogs.com/jackei/archive/2006/12/12/589473.html）查看。

1.1 明确的数字，而不模糊的语句

1.2 有凭有据，合理，有实际意义

1.3 相关人员达成一致

2、测试对象调查

测试对象调查，个人觉得主要从以下几个方面去深入：

2.1 关键业务是哪些，为什么这些是关键业务？

2.2 这些业务每天处理量是多少？

2.3 峰值处理量可能是多少？

2.4 峰值访问时间是在哪个时间段？

3、测试环境调查

3.1 硬件环境

主要是服务器（中间件服务器、数据库服务器）的配置情况。收集并归纳这些服务器的配置以及性能情况，可以为以后项目的服务器度量提供参考。并且，如内存、虚拟内存、CPU、磁盘 IO 活动频率的初始值也是结果分析一个参照点。

如下面的测试环境 CheckList：

项目名称	
项目简介	
硬件配置	CPU： 内存： 硬盘： 网卡
软件配置	操作系统： 应用平台： 数据库： 中间件： 杀毒软件： 防火墙：
网络配置	IP： 子网掩码：
测试工具：	LR（9.5）

网址：http://www.51testing.com/?uid-217803-action-viewspace-itemid-248283 上面有一个更加详细的 checkList 表单，大家可以参考一下。

3.2 软件运行环境

并发测试时服务器的运行要消耗比较多的资源，如果服务器充斥着各种各样的软件，同时运行着多种服务，那么此服务器做出来的测试结果肯定是不准确的，同时这也根本不符合正式服务器的情况，这时候做出来的结果当然也是毫无意义的。这个条件看起来是不会发生的，但我们在 XX 项目进行性能测试时发现，该服务器上配置了大量的服务（比如其它系统服务、Oracle 服务、杀毒软件运行中等），使得我们进行的性能测试的结果毫无意义，途耗时间和成本。当然，我的意思是在这种环境下，最好不要进行测试执行，但我们可以做性能脚本开发等工作，而不是完全停止掉性能测试的工作。

3.3 网络拓展图

Loadrunner 帮助文档性能测试计划部分有这么一段话：画出一个示意图来说明应用程序的结构，如果可能最好从已有的文档中提取，如果你的程序只是一个大型网络结构中的一部分，你应该确定该系统中有那些组件将应用于测试，并使用图解法分析之。

从网上的资料来看，性能测试结果受网络拓展图的变化而变化，如果系统是一个不断调优的过程，并且网络拓展图多次变化，那么最后度量出来的网络拓展，将是系统进行配置的最好方式。同时，理解网络拓展图，对于性能脚本的处理、性能结果分析（没看到相关文章的介绍）有着较好的作用。

3.4 中间件的配置

例如 IIS 的相关配置的初始值，包括应用程序池配置、CPU 的限制、关闭时间和启动时间间隔设置、对内存的限制及进程回收时间的设置等做调查。建立配置基线，随着各个参数值的不断地优化，性能测试的结果也将发生着变化。记录该变化的过程，可以很好的佐证“性能测试的结果是可信并变化着的”。可参考：http://www.automationqa.com/uchome/space.php?uid=4&do=blog&id=116。

3.5 服务器操作账户调查

调查可能使用到服务器的账户人员，在性能测试过程中需要锁定这些账户，防止在性能测试过程中有其它用户登录服务器，进行类似于杀毒、重启 IIS 等操作。很遗憾地是，上面所说的事情在我的性能测试过程中经常出现，更悲吹的是

有一次 3 个小时的场景过程中，网管正在用不认识的杀毒软件杀毒 VNC 连接远程杀毒，害得测试过程中 CPU 突然高升，找了半天才知道是杀毒软件在做恶。还有更令人无语的是，有一次在性能测试时，配置管理员，竟然重启了 IIS 以便更新一个小的服务。所以，服务器的账户在性能测试之前，我悲吹的历史告诉我最好能够严格控制。

4、测试脚本调查

在脚本录制完之后，也许我们还应该反反复复问问自己下面这几个问题：

4.1 测试脚本关键处是否加入了检查点（关于检查点的重要性我想这不用做太多介绍吧）？

4.2 测试脚本是否去掉了检查点所耗费的时间？

要知道是代码都会花时间，如果你加入的代码较多，那么也许你需要利用方法从 lr_wasted_time（waste）LR 中去掉了时间。

4.3 测试脚本是否真实的反应了测试用例的要求？

性能测试人员业务不熟，对于测试人员给出的业务场景有时理解不到位，实现了偏位的脚本，所以性能测试人员需要跟测试人员确认这一点。

4.4 测试脚本中的数据是否真实的反应了用户实际情况的数据？

数据应尽量模拟真实场景，真实的数据往往会带来一些想不到的问题。

4.5 测试脚本定义了足够的事务，以利于分析？

我觉得原则应该是一个操作动作一个事务，但令人遗憾的是至今有些人还是几个动作一个事务。一个动作一个事务的优点是，更好分析，而缺点是要多花点时间。

4.6 测试脚本是否在有 Cache 的基础上进行的？

有一篇文章介绍了带 IE Cache 和 Cookie 的区别的，文章标题好像叫：清空本地缓存与不清空本地缓存的差别

4.7 测试脚本运行后真实地在界面上起作用了吗？

同时登录到界面去看看，回放后界面是不是有相应的效果。因为 LR 是基于服务器与客户端协议回归的，只要服务器能够返回 200 的状态码（我举的是 HTTP 协议），那么 LR 都会通过脚本。这里有一个现象就是如果服务器实际返回的是 error，但是状态码为 200 那么脚本也会通过。这就是为什么要加检查点和回放后要到界面去看看的原因。

5、测试脚本中的数据调查

也就是通常所说的参数化和 IP 欺骗。测试数据和请求发起应尽量模拟真实情况，并且数据应该合理并够用。

以下问题，是一个真实的案例，在性能测试前期由于没有将登录账号进行参数化，到后期中是由于自动化功能测试发现现象，最终由性能测试发现问题的一个示例。

详细描述：

多个不同用户（100 个左右），在一个相同时间段内登录系统，应用服务器 IIS 连接池被占满，系统无法再次登录。

6、风险调查

测试过程中可能出现的风险，性能需求分析可能面临的风险等。性能测试的结果并不能百分百代表真实的效果，我觉得也只能说在性能测试的环境下，排除各种可能干扰影响实际环境下，这个结果才是可信的。所以，环境风险，也许应该在一开始就提出来。

7、性能计数器的调查

调查需要初始监控那些计数器，不是所有计数器我们都要监控的，我们需要择优选择。

有些情况下，Loadrunner 是监控不了的，如果监控不了需要想其它办法。

如果是出现了某些硬件瓶颈的现象，那么也许我们需要更进一步采用相关的计数器来进行监控，这时也许我们应该更加深入地调查相关的计数器。

8、运行时设置

你是否正确地设置了“F4”，不正确地设置 F4 将会也将会影响到性能测试结果。如何正确设置“F4”，老外有一篇文章已经很好的介绍了，详细请查看这个连接：http://www.myloadtest.com/vugen-runtime-settings/#comment-129077。

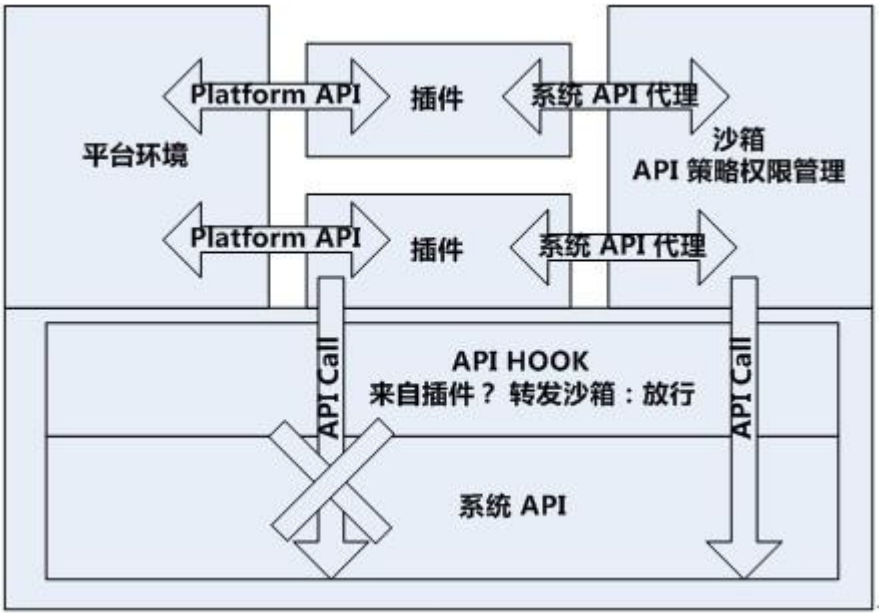
有时候觉得，做测试就跟做刑侦一样，抓取一些蛛丝麻迹，用一条线将其串连起来，然后深入调查和了解，再然后几个击破，最终抓得那个“元凶”。而环境的重要性对于刑侦来说意义是重大的，相同的测试前的调查工作我觉得也应该是重要的，本文从在性能测试的角度，综合网上各位达人的不同说法，对性能测试工作开展之前的调查工作做了描述，由于做的性能我还是比较少的，如有不正之处，还望各位看官指正。

【淘测试专栏】应用软件插件与沙箱的测试

作者：药尘

前言

为了方便的扩展各自平台环境的功能，更好的为终端用户服务，不少服务提供商都向开发者提供了插件开发的功能。在插件中，开发者除了调用服务商对外封装的平台（Platform）API 外，还可以调用系统 API。系统 API 的调用，如文件、注册表的写操作，可能对终端用户的操作系统产生影响。为了控制插件中系统 API 调用的影响，服务商要求插件以独立进程的方式在沙箱中运行。沙箱是一套隔离环境，通过在操作系统底层进行某些系统 API Hook 操作后，根据 API 策略权限管理机制，决定这些 API 是否继续向下调用。平台环境、插件、沙箱与操作系统的关系如下图：



由于涉及多个进程，需要通过进程间通信来完成一些信息交换。

插件及沙箱的测试策略

插件及沙箱的使用对象包括开发者和终端用户。测试过程中也需要从两个角度出发，针对不同的测试项目采用不同的测试方法实现。整体的测试策略如下：

测试角度	测试项目	测试方法	测试要求
开发者	开发文档	文档测试	文档中环境配置、API 说明用词准确 示例 DEMO 是能正确运行
	平台 API	接口测试	各种支持语言调用 API 接口，返回结果正确
	系统 API	接口测试	按照沙箱权限控制策略管理机制返回结果

	平台环境	功能测试	对于插件中的运行时错误，能良好的处理 能方便的加载调试开发中的插件 开发后的插件能顺利上传、更新到应用商店
终端用户	平台环境	功能测试	顺利从应用商店订阅、下载、安装插件 已安装的插件能正确更新 运行环境 API 版本与开发 API 版本不同时能正确处理
	插件	接口测试	平台 API、系统 API 调用结果与开发模式下相同

API 的接口测试方法

无论是平台（Platform）API，还是系统 API，都可以采用接口测试的方式进行。由于实际的调用参数，是由第三方代码决定的，可能由于其对 API 不熟悉或者代码错误等因素传入不符合要求的参数，因此平台 API 应有良好的参数校验机制，能处理各种参数，具有良好的健壮性。在接口测试中，应对所有的输入参数进行等价类划分，配合上边界值分析，将接口的实际返回结果与预期结果（Expect Result）进行比较。

平台 API 是内部 API 的对外封装，当内部接口存在错误时，API 也会返回错误。因此有条件的接口测试，还可以从平台环境内部调用内部 API，将内部 API 结果与平台 API 结果比较，可以帮助定位接口问题原因。

系统 API 的结果由操作系统返回，可能受到操作系统权限策略（如 UAC）、其他进程、调用参数的影响。下面以沙箱控制文件读写权限为例进行说明如何验证沙箱的读写权限。

沙箱限制文件的读写，通过 Hook 打开文件 API 调用，根据权限策略机制改变该函数中文件句柄的申请权限。权限申请能否成功，取决于如下几个因素：

1、插件中函数申请的权限

2、沙箱中，对权限的修改

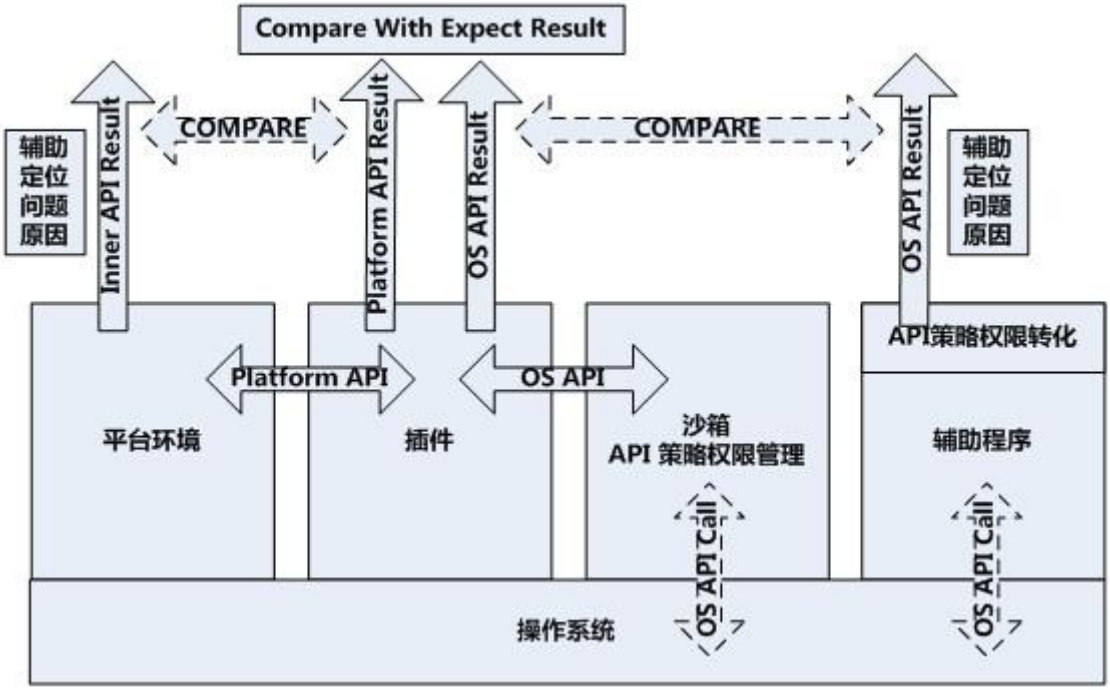
3、操作系统的文件访问权限

4、文件是否存在、是否被打开，是否开放读写权限

对于所有因素，可以采用决策表的方式，对预期的结果进行分析。其中，沙箱对权限的修改结果是测试的主要目的。在测试数据准备阶段，需要一个工具在沙箱外[沙箱策略可能会影响数据准备过程]完成因素 4 中涉及的数据准备。

为了突出因素 2 的影响，可以通过扩展辅助程序的功能来辅助结果判定。通过给沙箱中的插件和沙箱外的辅助程序传递相同的参数，可以排除因素 1、3、4，

再通过插件和辅助程序调用文件的读写、删除、重命名、替换 API，比较这 2 个程序调用 API 的结果，可以方便的判断沙箱的权限策略，甚至可以把沙箱权限转化规则，在辅助程序中做一次转化，以方便用程序进行比较。最终的测试框架结构，形如下图：



说在最后

插件和沙箱的测试，进程间的通信和 API 的测试是其重点。进程间通信，主要通过功能测试为主，日志输出为辅的方式进行，多考虑资源争抢及协议兼容问题。API 测试，通过接口测试和辅助程序判定的方式，可以方便的定位问题的原因。此外需要根据平台环境及沙箱的实现特点，做好完整的兼容性规划，对于沙箱的实现细节，也应做细致的分析。在做好相关的 API 测试覆盖测试的同时，测试工程师应充分评估沙箱设计和覆盖的验证 API 是否完善，是否存在漏洞。

探秘 QTP 的 Windows 标准对象——QTP 封装的属性简介

作者：文青山

摘要：本文简介了标准 Windows 控件的属性含义，以及如何利用这些属性来达到自己的测试目的方法和示例。

关键词：QTP

查看 QTP 帮助文档标准 Windows 控件中的任一对象，比如 Window 或 Dialog 对象，我们可知其 QTP 封装的属性有以下内容：

属性名称	属性描述
abs_x	x 轴的屏幕像速位置
abs_y	y 轴的屏幕像速位置
Class Name	对象的名称
enabled	该对象属性是否可见（true/false）
focused	光标是否定位于该控件上面
Foreground	该对象是否处于激活状态
HasBorder	是否有边框
HasCaption	是否有 title bar
HasHScroll	是否有水平滚动条
HasSizebox	是否定义边框的面积
HasSystemMenu	在 title bar 上是否有 menu
HasVScroll	是否有垂直滚动条
height	对象的高度
HScrollPageSize	水平滚动条的面积，如果对象没有水平滚动条，那么 GetROProperty 会返回空
HScrollPosition	水平滚动条的位置，如果对象没有水平滚动条，那么 GetROProperty 会返回空
hWnd	句柄
index	QTP 智能属性标识，从 0 开始
is child window	该窗口是否为子窗口(true/false)
is owned window	该窗口是否有从属窗口(true/false)
IsMdiChildWindow	该窗口是否为 MDI 窗口
IsPopupWindow	该窗口是否为 pop-up 窗口
IsToolWindow	是否有工具栏
LeftScrollbar	水平滚动条的初始位置是否在左边
location	QTP 智能属性标识，从 0 开始
MaxHScrollPos	水平滚动条所处的最大位置，如果对象没有水平滚动条，那么 GetROProperty 会返回空
Maximizable	是否可以最大化
Maximized	是否可以最小化

MaxVScrollPos	垂直滚动条所处的最大位置数，如果对象没有垂直滚动条，那么 GetROProperty 会返回空
MinHScrollPos	垂直滚动条所处的最小位置数，如果对象没有垂直滚动条，那么 GetROProperty 会返回空
NativeClass	对象窗口的名称
RegExpWndClass	MFC 对象窗口的名称
RegExpWndTitle	窗口的标题
RightAligned	对象是否有"right-aligned"属性
RightToLeftLayout	是否从右上方显示
RightToLeftReading	是否从右上方读取文件
text	某对象的文字信息
Topmost	是否置于所有窗口的关方
visible	对象是否可见(true/false)
width	对象的宽度
window id	对象的 windows id
WindowExtendedStyle	对象的外部 windows 样式
WindowStyle	对象的 windows 的样式
x	在窗口中 x 轴的像速位置
y	在窗口中 y 轴的像速位置

文字颜色为红色的部分，其它标准 Windows 对象都是有这些属性的，这些属性可以称之为 Windows 标准对象的公共属性。

背景色标黄的部分，我认为是可能需要经常用到的属性，这些属性的使用方向，个人觉得大概如下。

一、对象的识别

如：

abs_x	x 轴的屏幕像速位置
abs_y	y 轴的屏幕像速位置
x	在窗口中 x 轴的像速位置
y	在窗口中 y 轴的像速位置
index	QTP 智能属性标识，从 0 开始
location	QTP 智能属性标识，从 0 开始

这几组对象可用来做为对象识别的手段之一，即对象通过其它属性来描述，不能找到对象，并且点击不到时，可以利用此属性来描述确定并定义该对象，但 abs_x 和 abs_y 是在屏幕中的坐标位置，而 x、y 是窗口中的对象的位置。前者获取的位置，就如低级录制时获取的一样，而后者的位置是不随屏幕变化的，但却受控件位置的更改而变化。Index 和 location 为 QTP 自带的智能属性标识，其巨

大作用也不废话了，另外我在《WEB-QTP 随想录》中也有一些介绍，不太清楚智能属性标识的可以去看看。

通过 x 和 y 来识别对象的实例：

```
Dim filePath
filePath="%SystemRoot%\system32\calc.exe" "定义启动软件的地址
If Not Window("regexpwndtitle:=计算器").Exist(2) Then"如果不存在计算器，则启动该软件
    SystemUtil.Run filePath
End If
Window("regexpwndtitle:=计算器").WinButton("x:=57","y:=169").Click "通过 x,y 来操纵控件
```

你可以在 if 语句这里设置一个断点，然后手动改变一个窗口的位置，看看 Window("regexpwndtitle:=计算器").WinButton("x:=57","y:=169").Click 语句是否起到了作用。

另外，上面是通过描述性编程来实现的，当然我更建议的还是用 QTP 的对象仓库，如果你用 index 或 location 属性来描述编程，再我看来这并不是一个好注意。

二、界面测试

如：

HasHScroll	是否有水平滚动条
height	对象的高度
IsToolWindow	是否有工具栏
Maximizable	是否可以最大化
Maximized	是否可以最小化
Topmost	是否置于所有窗口的关方
width	对象的宽度

有些界面的测试用例，比如窗口的大小的比例是否符合黄金比例法则，窗口是否有最大化或最小化按钮，窗口是否处于所有窗口的最前面等用例就需要用到上面的属性来判断。

如“要求计算器窗口的最大化按钮不可用”的用例

```
Dim isMax
isMax=Window("regexpwndtitle:=计算器").CheckProperty("maximizable",true)
```

```
If isMax Then
    print "窗口可以最大化"
else
    print "窗口不可以最大化"
End If
```

三、易用性测试

如：

focused	光标是否定位于该控件上面

比如进入某个窗口后，光标需要位于第一个输入框中的用例。这只是一个高亮功能，但是这个功能我们通常认为可以省略用户思考的时间，同时也将起到提示作用，所以我们把这类用例归纳为易用性用例。

下面这个示例演示了利用这个属性来判断的实例：

验证 SQL2000 SQL Server 服务管理器启动后，光标是否默认在第一个输入框。


```
Dim filePath "Sql  服务管理器的路径
Dim isFoc"光标是否位于其中
filePath="C:\Program Files\Microsoft SQL Server\80\Tools\Binn\sqlmangr.exe"
"保证环境的初始性
If Window("SQL Server  服务管理器").Exist Then
    SystemUtil.CloseProcessByName "sqlmangr.exe"
End If
"启动软件
```

```
SystemUtil.Run filePath
 isFoc=Window("SQL Server  服务管理器").WinEdit("服务器(V):").CheckProperty("focused",true)
If isFoc Then
    print "软件启动后，光标默认是位于服务器输入框中"
else
    print "软件启动后，光标没有位于服务器输入框中"
End If
```

注意，为了提高编码效率，这里我是采用对象库来编写的，所以这段代码直接复制过去是执行不了的，你需要自己再添加一下对象库。

四、等待某个对象出现

如：

enabled	该对象属性是否可见（true/false）
visible	对象是否可见(true/false)

有时候，我们在执行某个操作之后，某个按钮变为置色状态或不可见状态，怎么去判断这个控件处于这个状态呢，当然也就可能要用到上面的属性了。

下面的示例演示了“验证 SQL2000 SQL Server 服务管理器启动后，开始/继续按钮处于不可用状态”。

```
"判断开始/继续(s)按钮是否可用
Dim isEnabled
isEnabled=window("SQL Server  服务管理器").WinButton("Button").CheckProperty("enabled",true)
If isEnabled Then
    print "软件启动后，默认开始/继续按钮可用，服务未启动"
else
    print "软件启动后，默认开始/继续按钮不可用，服务已启动"
End If
```

五、测试成功的判断依据

如：

text	文字信息

如获取某个输入框对象的文字信息与预期的文字信息是否一致，一致则 Pass，不一致则记录 Fail 的用例。

```
"判断启动后，服务器输入框中的字符是否为“23F7CB1691A8445”
Dim isEditRight
isEditRight=window("SQL Server  服务管理器
").WinButton("Button").CheckProperty("text",true)
If   isEditRight Then
   print "默认服务器为：23F7CB1691A8445 Pass"
else
   print "默认服务器不为：23F7CB1691A8445 Fail"
End If
```

六、childObjects 的应用

如：

Class Name	对象的名称
NativeClass	对象窗口的名称
text	某对象的文字信息

选择这三个属性来查找或操作某对象，通常选择的属性包括但不限于这 3 个，良好的属性选择其实决定了使用该方法查找到该对象的速度。

下面这个示例演示了如何利用 Class Name 属性来做一些特别酷的操作。

```
"查找窗口中所有的 Edit 对象，并赋值为 testing
set EditDesc = Description.Create()
EditDesc("Class Name").Value = "WinEdit"
Set EditS=Window("SQL Server  服务管理器").ChildObjects(EditDesc)
For i=0 to EditS.count-1
   EditS(i).set "testing"
Next
```

同事常常问一些问题，这些问题大多都是诸如“XX 这种场景下，我该怎么判断？XX 这种场景下，怎么识别不到对象了呢？”，我认为这些问题大多都是同事不知道 QTP 对象的属性的含义以及怎么灵活运用这些含义造成的。了解这些属性的含义和常用的方向，我觉得是我们做好自动化测试的一个必备技术条件

之一，如果不能灵活运用这些属性或者根据不知道这些属性的含义，那么你还是静下心来看看这些属性的含义，对这些属性都做一下试验吧！

采访测试专家 Jim Sivak

作者：飞燕儿 译

这次，我们提出一些问题，不断的与测试专家探讨。Jim 已经在计算机技术领域工作了 35 年了，这其中包括最近在 McAfee 做了四年的资深 QA 经理。他的测试职业生涯开始于 SpaceShuttle，在那些岁月里总是围绕着 warehouse 系统，cyclotrons，radars，操作系统和现在的安全软件。他是 ASQ 的一个资深成员，并且获得过软件质量工程师（CSQE）称号。

在我们面试的第一轮中，我们要获得他对于忽略安全测试风险的想法；在移动程序和设备安全测试方面的错误思想；malware 的进化；管理 QA 的期望；SQAG 的含义等等更多的内容。明天回来审核第二部分

uTest：我们了解到你最近结束了在 Macfee 四年的工作，继而加入了 Unidesk.. 我想问你的第一个问题就是：你要们 Unidesk 主要开发哪一个行业的产品？你在这里最期望担当什么新角色？

JS：Unidesk 主要从事虚拟桌面领域。我们的产品让公司在使用虚拟桌面时的感觉和物理桌面没什么大的差别。基于我们的技术，个性化桌面非常易于管理。我们的产品让企业使用虚拟桌面，跟物理桌面的外观、感觉、容量一样。虚拟桌面将会成为 IT 部门最好的朋友，系统的变化和补丁更新后，可以自动被复制到每个相关联的桌面。因为 Unidesk 是一个刚刚成立不久的公司，我有机会制定公司 QA 的流程及目标，决定战略和战术策略。使用新技术和以前的经验，能够很好的完成我每天的工作内容。

Utest：你在 McAfee 一定了解了许多关于 WEB 易被攻击的地方吧（如安全危险）。回顾过去这些年，在关于业务和顾客来说，处理安全方面问题采用的方法中，哪种最令你意外？

JS：Mike，你这个问题问得非常好。最令我意外的是有时候公司存在像鸵鸟一样总是把头埋在沙子里的员工。已经拥有了攻击和技术，相关的资料准备好了，可以使用了，但是在一些安全方面的问题发生前，安全问题的等级总是被设置为较低优先级。看看那些每天基本上出现的问题就了解了。在家，有多少人会打开 email 并且回复说："你的信用卡申请已经为你准备好了"。

uTest：如果用户更关心 web 上存在的威胁的话，这样看起来似乎是安全的，而对比 mobile 的话就相反？在你看来，mobileapp、社会媒体和第三方集成软件会怎样影响安全。

JS：这些新测试方法中出现安全错误和过失的机会很大，仅看一下平常所发生的事就知道了，如一些人在他们不知情的情况下，发送了恶意的链接给他们的网友，或者当在下载以及在设备上安装这些下载软件时，往往有可能带有恶意软件。人们总是假想自己的电话是安全的，或者假想他们的 tablet 是永远不会被黑客攻击的。再次，软件供应商需要将严重的安全问题都解决掉，不是总等到有大问题发生后再来解决。这一切都要归结到一个实际问题那就是我们的用户是人，还有就是很多事情都是我们想当然的。

uTest：安全威胁（如病毒、恶意软件等等）或者用于解决这些安全问题的技术哪一个演变得更快？为什么是这种？以及这对最终用户来说有怎样的暗示？

JS：在我看来，非常不幸的是恶意软件总是存在。尽管试图在问题发生前进行检测，大多数的技术都是可用的--首先威胁存在，而且解决方案/检测随后才进行的。

UTest：下面这个问题看起来像是面试问题，就是你在 McAfee 面临最大的测试挑战是什么，你又是怎样克服的？

JS：我们团队开发的产品是一个企业级的项目，这个项目，主要是为有成百上千资产的客户使用，而这些客户，他们的客户需要浏览几百万个系统。你不能向你的 boss 要求拿钱出来建一个有 10 万台桌子的实验室。所以我们面临的是大规模的挑战。面对这个非常松散的环境，你怎样测完成产品的测试？我们采用的就是模拟和联系测试项目来解决这个问题。我们通常会不断地改善我们复杂的测试工作区。

uTest：测试团队总是会面对市场、销售、产品等部门的时间限制压力（特别是测试管理者）。团队是怎样做到其他部门所期望的结果？

JS：一旦你创建了一套可靠的跟踪产品的管理方法，那些测试目标应该就能很好的管理起来。这些数据和流程都是非常重要的。如果你能证实你的数据测试点非常可靠的话，那么其他人就很难反驳你了，即使反对的人挥手不同意，也会束手无策。从一个流程的角度来看，创建一套架构，其他团队能够清楚地知道完

成产品需要什么，或者 FeatureComplete 的真正定义是什么。例如，把 QA 团队放在一个可拥护的位置。

uTest：按照你的观点，当谈到软件测试时，有什么做法会减少收益吗？换句话说，经过某一阶段的测试后，产品真的变得不一样了吗？

JS：噢，你问的这个问题可能是大家都疑惑的问题----你做测试吗？尽管我的定义可能有一点点不同于其他人的，但是确实是有出现收益减少的情况。由于业务决定必须上线产品，开发则不会再修改那些存在的问题，那么我会停止测试。换而言之，如果所发现的那些 bug 被延迟解决，那么就停止测试。如果在一个版本里面发现的 bug，开发不修改，那么继续测试下去也是没什么意义的。自然而言，软件没有代码错误是不可能的，因此测试总能够发现更多的 bug。当版本发布时，通过业务提供的数据，我们会让业务决定减少收益哪些地方存在。

uTest：我看见你在 2011 年 STPcon 上你所赠予的，所以我了解 WAG 和 SWAG 之间的区别。你能为没能参加这个活动的人解释一下这些首字母缩略词都代表什么含义？为什么对测试团队来说如此重要呢？

JS：当我第一次碰到这些词汇术语时，我也记不住，但是它们是有一些实际意义的。WAG 是"wildassguess"的首字母缩略词，而 SWAG 是"scientificwildassguess"首字母缩写。它们的意思就是他们字面上所指的含义。当多次被问到"要测试多长或者需要多大的努力或者我们产品什么时候能发布出去"时，测试团队只能是猜测而已。在问一些基于测试团队已经提出的流程方面的问题前，，下一步就是申请做一些在测试成熟度方面的质量分析。它们仍然是猜测 SWAGs，但是已经非常有把握了。

uTest：在过去的 30 多年里，你已经为公平分享平台招聘了许多的软件测试人员。那么，你招聘的人员需要具备哪些技能及品质呢？

JS：我理想中的合适人选需要具备以下三个特质：他们爱专研，并且不断得把事情做得更好（不满于现状），具有怀疑精神。要成为一个成功的 QA 工程师，就必须不断地提升自己以及自己所在的团队。最好的团队就是不断地尝试新方法和流程。测试就是在不断地解决疑惑—你怎样确认它是正常工作的，你怎样判断什么可以中止以及在什么时候中止，初始化工作实际上什么开始。我认为一些其他的技能：如编程能力，产品业务知识等等也能在工作中学习获得。

Utest：你们跟开发团队合作过程中，坚持使用各种各样的不同的方法（敏捷、瀑布等方法）从测试的角度看，这些方法哪一个是最能提高工作效率的？有没有某种方法在测试程序中不适用的？

JS：你可以发现测试方法中任何一种方法都有成功或失败的地方，如果这些方法都能被正确地执行，我认为没有哪一种方法比其他方法更高效。每种方法都会给测试团队带来不同的挑战，没有单一的方法适用在任何地方。关键一点是你按照怎样的软件生命周期在执行。把你在瀑布模型中学到的测试技术用于敏捷流程中，这不可能让你成功。你必须灵活地运用软件开发生命周期。

从测试的角度看，我认为瀑布模型比其他模型更适用于 SDLC 中。如果业务流程支持，并且要求对需求更有追溯性，许多精确的文档以及正式的测试周期比并发性测试更适用。敏捷并不一定适用你所的项目，并且很少有时间允许你采取正确的措施。

效能实际上取决于客户的满意度，跟生命开发周期没什么关系。

uTest：至今为止你所在的测试团队犯过的最严重的错误是什么？

JS：MiKe，这问题很有趣。我所经历过的，犯得最大的错误是在使用策略上，而不是在进行策略计划上。我所说的意思是：由于进度紧，资源缺乏，团队人员把所有的努力都放在当前的任务中，从一个版本到下一个版本。没有时间对计划进行改进，没有时间将所有的测试策略更改进行统一优化。由于要有可追溯性，所以项目中有更小的改动，但是更多的地方都存在迷糊。团队要求能够站在足够的高度抛开时间来考虑所有的一切产品质量相关事。只有这样他们才能确定是不是按照预定目标进行的流程，以至于花一年的时间来完成，但有巨大的投资回报率的机会。

Utest：测试自动化是你所擅长的一方面。那么你对一些刚打算设计测试自动化套件的公司有什么建议么？

JS：我想引用“七个习惯。。。stevencovey”中的一句话：做有把握的事。换句话说：你首先要想好你为什么要自动化，以及你自动化想要达到什么目的。短期项目进行自动化，成本是非常高的，自动化用于在长期项目中才会觉得非常的高效。但是你必须要有一个计划----计划中包含你做的一些事，这些事能估量你的目标流程。不要因为开发商或供应商不断地告诉自动化能解决你所有的问

题，你就信以为真的做自动化。你要弄明白：你抛开自动化你要达到什么目的，而且非常重要的是，你要了解成本和这长期做下来的收益。

uTest：你的测试生涯开始于 SpaceShuttle（不是你常说的“开始”工作）。那么在这个职位中你要负担什么样的任务和责任？你又是怎样为你的软件 QA 职业做准备的呢？

JS：我开始在测试和仪表组中工作。我们的职责就测试设计以及为工程师发展制定发展方向。

那时，Rockwell 开始从生产带图记录仪到生产电脑，并且聘用了大量有生产新型电脑经验的年轻大学生。我和不同的高级测试工程师们一起工作，当我们测试起落装置、船员舱、飞行液压甚至磁瓦的时候，他们会指导我。在那期间，两个主要课程为我提供了：1）当你要测试和记录所有重要事情的详细信息 2）测试步骤，查看测试任务，以及创造性思维。这里有大量的 2）的例子：测试船员舱压力输送----你不能得到空间容量，因此接下来最重要的事就是给船舱加压到 2 倍大气压力，然后根据给你的一个主要测试参数的气压差来进行测试。

RapidFire：

适合工作的地方：东海岸还是西海岸？

Mike，这是偏离主题的问题。。因为东海岸的天气很糟糕，因为你没有到外面去的激情，所以工作机会很多。相反地，西海岸天气非常适宜（至少在加利福利亚南部是这样的。）以至于你有想到沙滩边去玩的欲望。

WebService 功能测试

作者：冰雪

摘要：由于 webService 的平台无关性，使用越来越多，所以对 webService 的测试也越来越受关注，本文整理了 webService 的功能测试方法，包括编码方式和工具方式。

关键词：webService 测试

1、WebService 简述

WebService 是一种革命性的分布式计算技术，本质上就是服务提供方发布一些服务（实现一定功能的函数接口）到网络上，服务使用方如果使用到该服务的功能，直接在网络上调用服务接口就可以了，无需自己重新开发。

那么，服务使用方（我们称之为客户端）是如何使用服务发布方发布（我们称之为服务端）的 webService 服务的呢？这里简要介绍：WebService 发布后，其服务是封装在一个 wsdl（Web Services Description Language，Web 服务描述语言）文件中，客户端发请求主要是向发布好的 wsdl 地址以 SOAP 方式发请求，调用过程如下：

服务端生成服务描述文件，以供客户端获取。

客户端取得服务端的服务描述文件，解析该文件从而获得服务端的服务信息以及调用方式。

客户端指定调用方法和参数，生成恰当的 SOAP 请求消息，发往服务端，并等待服务端返回的 SOAP 回应消息。

服务端接收客户端发来的 SOAP 请求消息，解析其中的方法调用和参数格式。并根据 wsdl 的描述，完成指定功能，将返回值放入 SOAP 回应消息返回给用户。

客户端解析得到的返回值。

使用 WebService 的优点是一次开发多次使用，且由于 WebService 的平台无关性特性，使用越来越多，所以对 webService 的测试也就显得越来越重要。

2、WebService 功能测试方法

WebService 测试最基本的是功能测试，即验证功能的正确性。另外由于服务发布后会由多个客户端进行调用使用，所以性能测试也是一个重要测试内容。

本文只涉及了功能测试部分。功能测试分为编码方式（通过编码测试接口的正确性）和工具方式（通过测试工具测试接口的正确性）。

对于编码方式：WebService 常用的框架有 axis、xfire、cxf 等，对应的有相应的测试方法。

对于工具方式，商用的就不考虑了，免费的工具中有 TestMaker（较复杂，需要学习 java 和 Python，学习曲线长）、WebInject（需要懂得 soap 原理，开发人员用得多）、WSCaller（工具过于简单，只能进行简单的功能测试）、soapUI 开源版本（该工具由 Java 语言开发，有 Eclipse 插件。脚本语言 Groovy 是类 Java 的轻量级脚本语言。组织目录由 TestSuit 和 TestCase 构成）。其中的 soapUI 使用较多，尤其是对懂得 Java 的人来说上手较快。本文只对 soapUI 工具测试 WebService 的方式进行介绍。

3、编码方式

针对 WebService 常用的框架 axis、xfire、cxf，该部分描述了 4 种 webService 功能测试方法及其具体步骤。

使用的开发和测试环境是 eclipse3.6.2。

具体步骤中使用了一个 HelloWorld 的例子进行说明，其中包含两个接口：

```
public interface HelloWorldService {
        public String sayHello(String name);
        public String sayWords(String... words);
}
```

函数功能如下：

```
public class HelloWorldServiceImpl implements HelloWorldService {

    public String sayHello(String name) {
        String words = "Hello, World!";
        if(StringUtils.hasText(name)){
            words += " " + name;
        }
        System.out.println(words);
        return words;
```

```
    }

    public String sayWords(String... words) {
        String theWords = "Hello, ";
        if(words != null){
            theWords += StringUtils.join(words);
        }
        else{
            theWords += "nothing to say";
        }
        System.out.println(theWords);
        return null;
    }

}
```

3.1 使用 Axis 测试接口（wsdl 服务地址）

该部分使用 wsdl 的服务地址进行测试。

注：axis 包，是个 webservice 的工具。

官方地址：http://ws.apache.org/axis/

3.1.1 导入相应的包

需要导入的包如下 14 个：

activemq-4.0-M3.jar

axiom-api-1.2.7.jar

axiom-impl-1.2.7.jar

axis-1.4.jar

axis2-1.4.jar

commons-httpclient-3.1-rc1.jar

data-management-cli-1.2.2-app.jar

dspace-lni-client-1.5.1-jar-with-dependencies.jar

jsr173_1.0_api.jar

neethi-2.0.4.jar

openejb-itests-standalone-client-3.1.jar

woden-api-1.0M8.jar

wsdl4j-1.6.2.jar

XmlSchema-1.4.1.jar

3.1.2 编写测试代码

Helloworld 的测试代码如下：

```
package com.ruijie.samples.helloworld; （包名可以任意取）
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
import javax.xml.namespace.QName;
public static void main(String [] args) {
try {
String endpoint =
"http://localhost:8089/HelloWorldServiceComponent/HelloWorldService";
Service    service = new Service();
Call          call        = (Call) service.createCall();
call.setTargetEndpointAddress( new java.net.URL(endpoint) );
call.setOperationName(new
QName("http://HelloWorldService.sca.helloworld.samples.ruijie.com/", "sayHello"));
String ret = (String) call.invoke( new Object[] { "aaa" } );
System.out.println("Sent 'aaa', got '" + ret + "'");
} catch (Exception e) {
System.err.println(e.toString());
}
}
```

或者：

```
package com.ruijie.samples.helloworld; （包名可以任意取）
import org.apache.axis.client.Call;
import org.apache.axis.client.Service;
```

```
import javax.xml.namespace.QName;
import javax.xml.rpc.ServiceException;
import java.net.MalformedURLException;
import java.rmi.RemoteException;
public static void main(String[] args) throws RemoteException,
ServiceException, MalformedURLException {
Service service = new Service();
Call call = (Call)service.createCall();
call.setTargetEndpointAddress(new
java.net.URL("http://localhost:8089/HelloWorldServiceComponent/HelloWorldServi
ce"));
call.setOperationName(new
QName("http://HelloWorldService.sca.helloworld.samples.ruijie.com/", "sayHello"));
call.setUseSOAPAction(true);
call.setReturnType(org.apache.axis.encoding.XMLType.XSD_STRING);
call.setSOAPActionURI("http://HelloWorldService.sca.helloworld.samples.ruijie.com
/sayHello");
String k = (String)call.invoke(new Object[]{}); //因为返回值是 String 类型，所
以这里调用的返回值也是 String 类型
System.out.println(">>> "+k); //返回值输出
}
```

3.2 使用 Xfire 测试接口（wsdl 文件）

前置：创建测试工程。New-project-Dynamic Web Project

3.2.1 导入相应的包

需要导入的包如下 6 个：

xfire-all-1.2.6.jar

spring-core-2.0-rc2.jar

commons-httpclient-3.1-rc1.jar

data-management-cli-1.2.2-app.jar

dspace-lni-client-1.5.1-jar-with-dependencies.jar

XmlSchema-1.4.1.jar

3.2.2 将 wsdl 文件导入到工程的 src 目录下

3.2.3 编写测试代码

Helloworld 的测试代码如下：

```
package com.ruijie.samples.helloworld;（包名可以任意取）



import org.codehaus.xfire.client.Client;
import org.springframework.core.io.ClassPathResource;
import org.springframework.core.io.Resource;



 public class WebServiceClientTest {


        public static void main(String[] args) throws Exception {
            WebServiceClientTest test = new WebServiceClientTest();
            test.testClient();
        }

        public void testClient() throws Exception {
            String wsdl = "HelloWorldService.wsdl"; //对应的 WSDL 文件
            Resource resource = new ClassPathResource(wsdl);
            Client client = new Client(resource.getInputStream(), null); //根据
WSDL 创建客户实例

            Object[] objArray = new Object[1];
            objArray[0] = "tianxx";
```

```
        //调用特定的 Web Service 方法
        Object[] results = client.invoke("sayHello", objArray);
        System.out.println("result: " + results[0]);
    }
}
```

3.3 使用 Cxf 的 java2wsdl 工具测试接口

该部分描述了使用 cxf 的 java2wsdl 工具（命令行方式）测试 webservice 接口的步骤。

3.3.1 下载安装 JDK 并配置环境变量

也可直接使用开发环境 eclipse 的 jdk。

3.3.1.1 打开下载地址

https://cds.sun.com/is-bin/INTERSHOP.enfinity/WFS/CDS-CDS_Developer-Site/en_US/-/USD/ViewProductDetail-Start?ProductRef=jdk-6u7-oth-JPR@CDS-CDS_Developer

3.3.1.2 下载 JDK

（我下载的版本是 jdk-6u7-windows-i586-p.exe）

点击运行安装，我安装的目录是默认的安装目录：C:\Program Files\Java\jdk1.6.0_07

3.3.1.3 配置环境变量

JAVA_HOME=C:\Program Files\Java\jdk1.6.0_07

CLASSPATH=.;C:\ProgramFiles\Java\jdk1.6.0_07\lib\tools.jar;C:\ProgramFiles\Java\jdk1.6.0_07\lib\dt.jar;C:\Program Files\Java\jdk1.6.0_07\bin;

path=C:\Program Files\Java\jdk1.6.0_07\bin;（这个放到 path 的开头）

3.3.1.4 验证安装

写一个简单的 java 程序来验证是否已安装成功：

```
public class hello
{
 public static void main(String args[])
 {
 System.out.println("Hello");
```

```
    }
}
```

将程序保存为文件名为 hello.java 的文件。

打开命令提示符窗口，进入到 hello.java 所在目录，键入下面的命令

javac hello.java

java hello

此时若打印出来 hello 则安装成功，若没有打印出这句话，仔细检查以上配置是否正确。

3.3.2 下载配置 cxf

3.3.2.1 打开下载地址

http://cxf.apache.org/download.html

3.3.2.2 下载 cxf

（我下载的版本是 apache-cxf-2.4.0.zip）

3.3.2.3 解压缩

解压缩到一个目录，我解压的目录是 G:\apache-cxf-2.4.0

3.3.2.4 配置环境变量

配置环境变量%CXF_HOME%= G:\apache-cxf-2.4.0（以我的目录为例）

并在 PATH 后加上;%CXF_HOME%\bin

3.3.2.5 验证安装

打开 cmd，进入 G:\apache-cxf-2.4.0\bin 目录，运行 wsdl2java 命令可成功。

注：Win7 下转换目录需要输入/d 参数，如 cd /d g:\apache-cxf-2.4.0\bin

3.3.3 使用 wsdl2java 工具

wsdl2java -d G:\src -client http://www.webservicex.net/CurrencyConvertor.asmx?WSDL

-d 生成文件存放到哪里

-client 生成客户端类

其他参数使用请参照：https://cwiki.apache.org/CXF20DOC/wsdl-to-java.html

3.3.4 测试代码编写

A、到 eclipse 中建立一个工程。

B、将上一章节 3 中使用 wsdl2java 工具生成的代码拷贝到工程目录中，或直接用工具生成代码到工程中。

C、下载需要的 jar 包，导入到工程中。

包下载地址：http://www.jarfinder.com/

方法：运行 wsdl2java 工具自动生成的 client（此时服务需要处于可访问状态，可通过访问 wsdl 验证是否可访问），按照错误提示导入需要的包，直到 client 可运行成功，说明需要的包已经导入完全。

其中一个提示没有告诉具体的错误原因：No method error：com.sun.xml.bind.api.JAXBRIContext.Instance。需要下载 jaxb-impl-2.1.13.jar（即最新的包），若没有最新的，2.1.8 也可以，但是早期版本不支持，比如 2.0.1 就不可以。

我导入的包是下面 12 个（工程右键选择 Properties-Java Build Path-Add External JARs 或者直接导入到工程相应目录下——工程名\WebContent\WEB-INF\lib）：

axis2-jws-api-1.4.1.jar

axis2-saaj-1.2.jar

axis2-saaj-api-1.3.jar

commons-logging-1.1.1.jar

jaxb-api-2.1.jar

jaxb-impl-2.1.8.jar

jaxb-xjc-2.1.7.jar

jaxws-api-2.1-1.jar

jaxws-rt-2.1.4.jar

ow2-bundles-externals-jaxb2-1.0.9.jar

streambuffer-0.8.jar

wstx-asl-3.1.1.jar

D、编辑修改 client 代码，进行接口测试。（使用 Junit 即可）

3.4 Dynamic Client

Cxf 也提供了一个不需要显性的生成 java stub，参考下面的网址：

https://cwiki.apache.org/CXF20DOC/dynamic-clients.html

wsdl2java 静态方式调用和动态调用的区别：如果自己用 java 调用，那就自己生成 stub，然后来调用；如果用 Dynamic 的话，其实也是 Cxf 帮你后台生成而已。

3.4.1 导入相应的包

需要导入的包如下 14 个：

commons-logging-1.1.1.jar

cxf-bundle-2.1.3.jar

geronimo-activation_1.1_spec-1.0.2.jar

geronimo-annotation_1.0_spec-1.1.1.jar

geronimo-javamail_1.4_spec-1.5.jar

geronimo-stax-api_1.0_spec-1.0.1.jar

jaxb-api-2.1.jar

jaxb-impl-2.1.8.jar

jaxb-xjc-2.1.8.jar

neethi-2.0.4.jar

wsdl4j-1.6.2.jar

wstx-asl-3.2.7.jar

xml-resolver-1.2.jar

XmlSchema-1.4.2.jar

注：

JDK 版本 1.5

TOMCAT 版本 5.X

上述条件必须有，否则会在客户端提示包错误。

还有要特别注意在 JDK（仅供程序出问题时参考（本示例未做该项处理））\%JAVA_HOME%\jre\lib\endorsed 目录或者 %tomcat_home%\common\endorsed 目录下必须要用到两个包：

jaxb-api-2.1.jar

jaxws-api-2.1.jar

否则会提示如下错误：

JAXB 2.0 API jar is being loaded (from jar:file:/C:/Program%20Files/JetBrains/IntelliJ%20IDEA%209939/lib/javaee.jar!/javax/xml/bind/annotation/XmlSchema.class), but this RI (from jar:file:/C:/dev/AgileMark.svn.work/lib/jaxb-impl-2.1.9.jar!/com/sun/xml/bind/v2/model/impl/ModelBuilder.class) requires JAXB 2.1 API jar.

3.4.2 编写测试代码

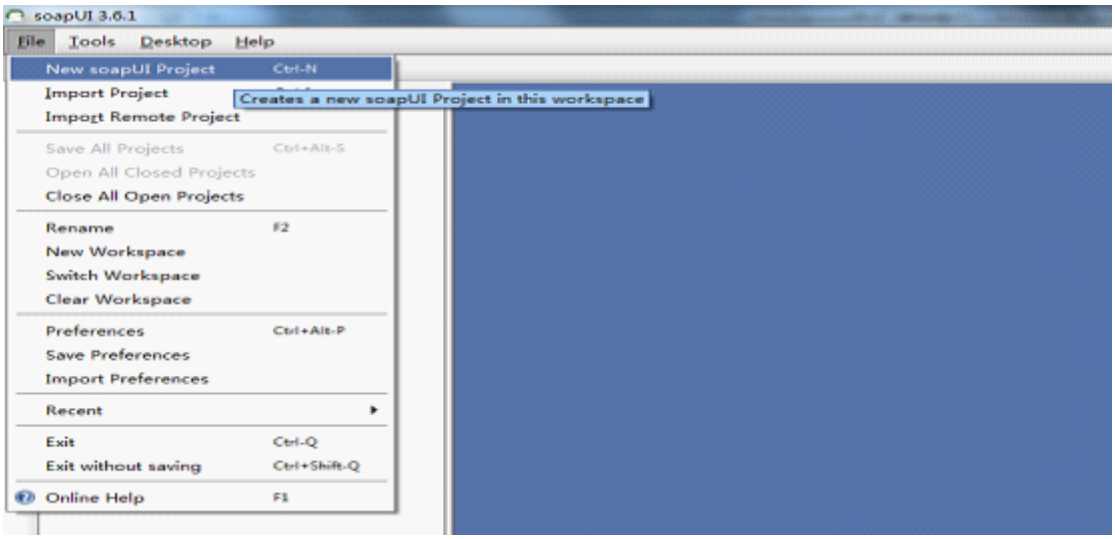
```
package dynamicSample;
import org.apache.cxf.jaxws.endpoint.dynamic.JaxWsDynamicClientFactory;
import org.apache.cxf.endpoint.Client;
public class SampleTest{
        public static void main(String[] args) throws Exception {
        SampleTest test = new SampleTest();
        test.testClient();
    }
        public void testClient() throws Exception {
        JaxWsDynamicClientFactory dcf =
JaxWsDynamicClientFactory.newInstance();
//          Client client = dcf.createClient("HelloWorldService.wsdl");
        org.apache.cxf.endpoint.Client client =
dcf.createClient("http://localhost:8089/HelloWorldServiceComponent/HelloWorldSer
vice?wsdl");
        Object[] res = client.invoke("sayHello", "tianxx");
        System.out.println("Response: " + res[0]);
    }

}
```

4、工具 soapUI 方式

下面以测试服务 HelloWorldService 为例，详细说明使用工具 soapUI 进行 WebService 功能测试的操作步骤。

4.1 将 WebService 导入工程

单击‘File’->‘ New soapUI Project’，如下图：



B、在弹出的对话框中输入待测试的 ws 信息，然后点击 [OK] 到下一步

Project Name：HelloWorldService

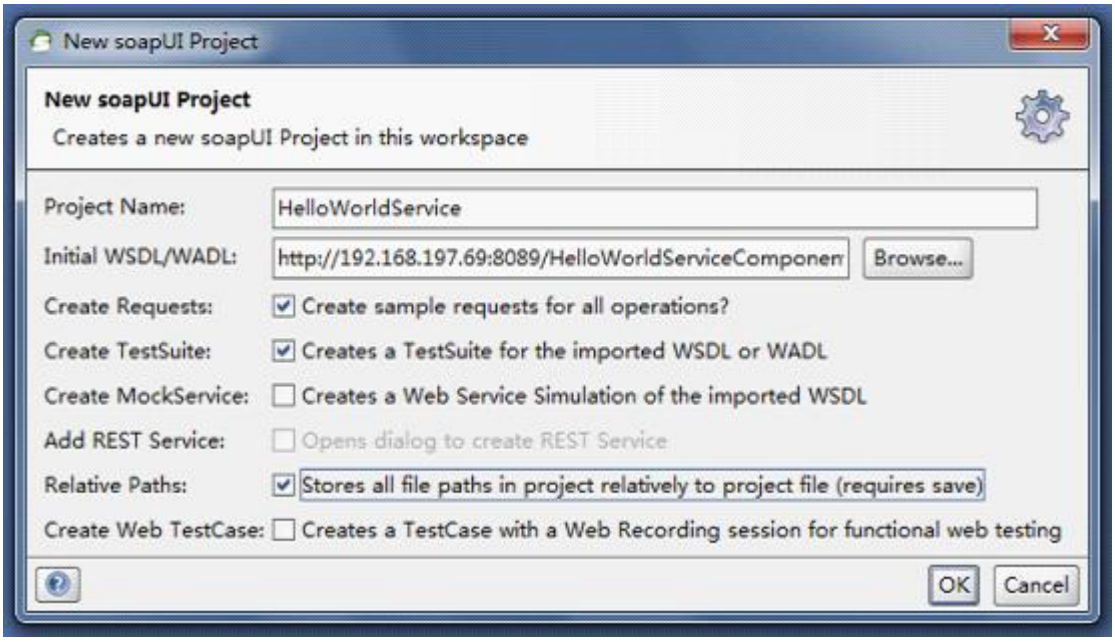
Initial WSDL/WADL:

http://192.168.197.69:8089/HelloWorldServiceComponent/HelloWorldService?wsdl

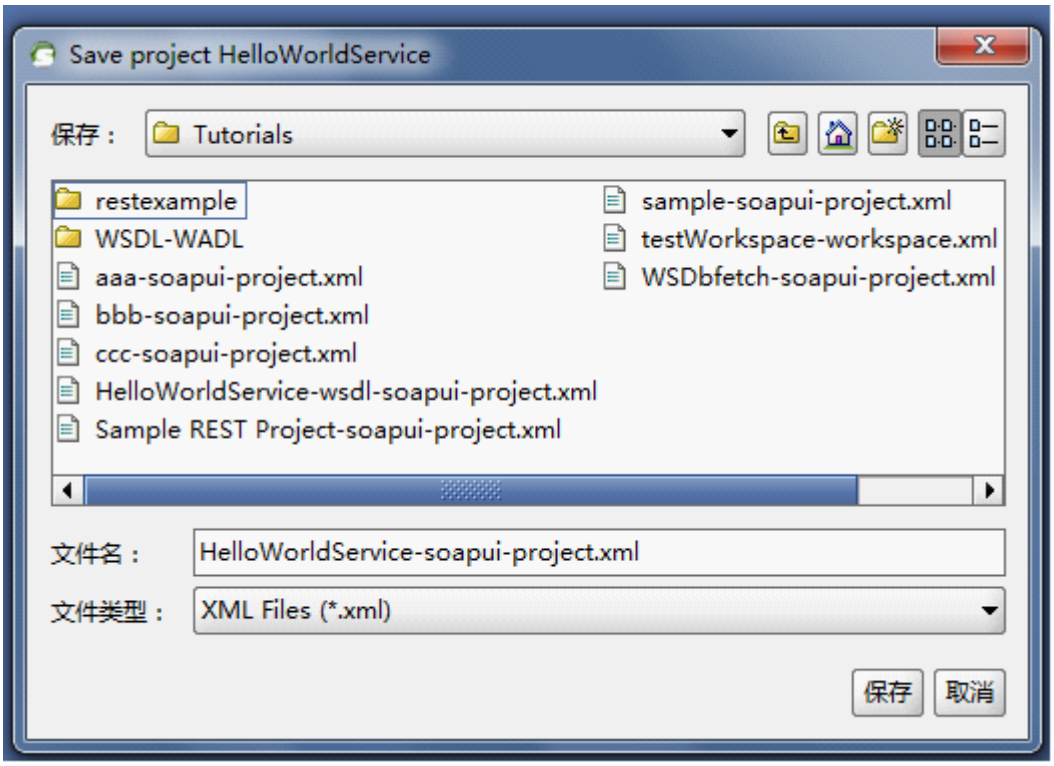
Create Requests：选中

Create TestSuite：选中

Relative Paths：选中



C、保存 project



4.2 创建初始的 TestSuit 和 TestCase

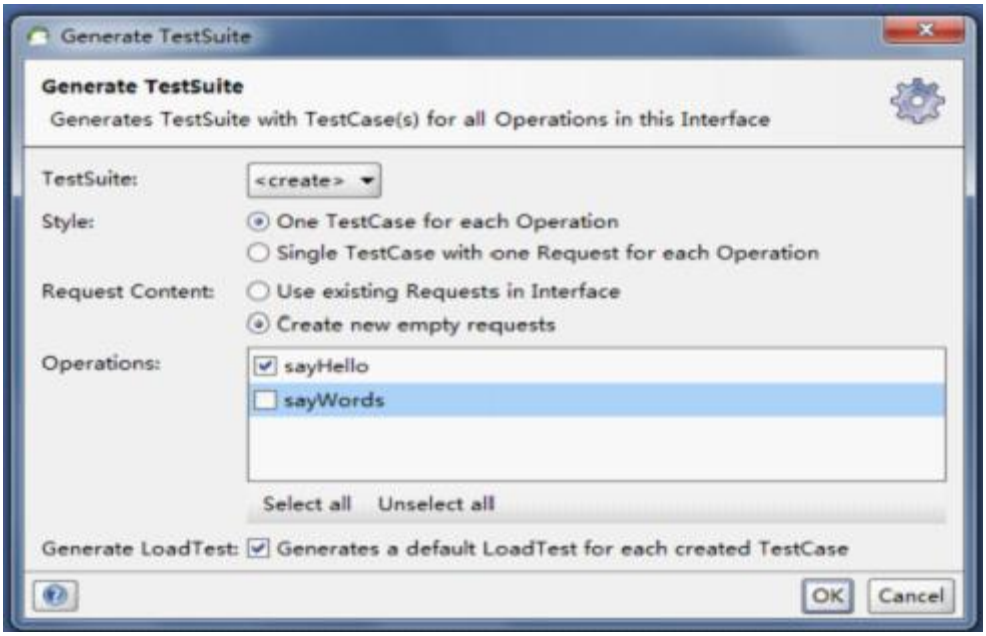
A、生成初始的测试用例

选择 One TestCase for each Operation：每个接口创建一个用例

选择 Create new empty requests：创建一个空的请求

Operations：选择待测试的方法

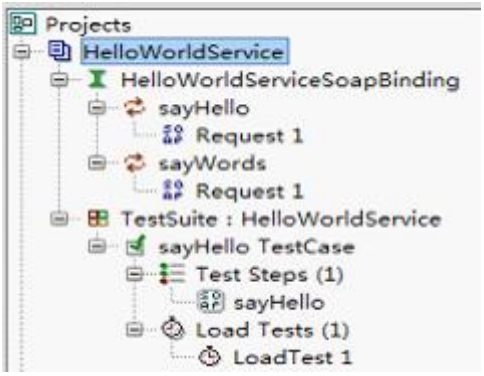
选择 Generates a default LoadTest for each created TeseCase：每个用例生成一个负载测试（为后面性能测试做准备）



B、生成 TestSuite

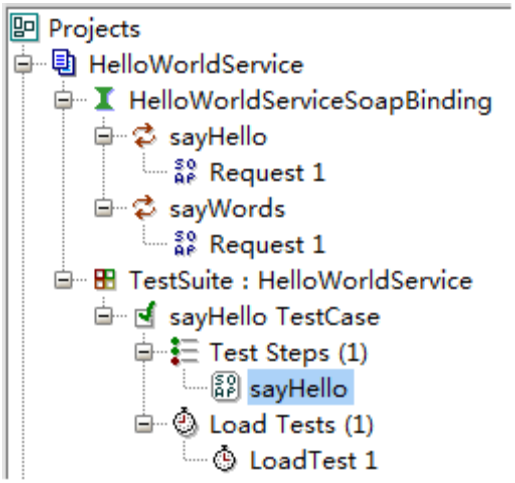
这里可以根据习惯更改 TestSuite 的名称，如 TestSuite：HelloWorldService。

C、在 soapUI 的左侧生成如下目录

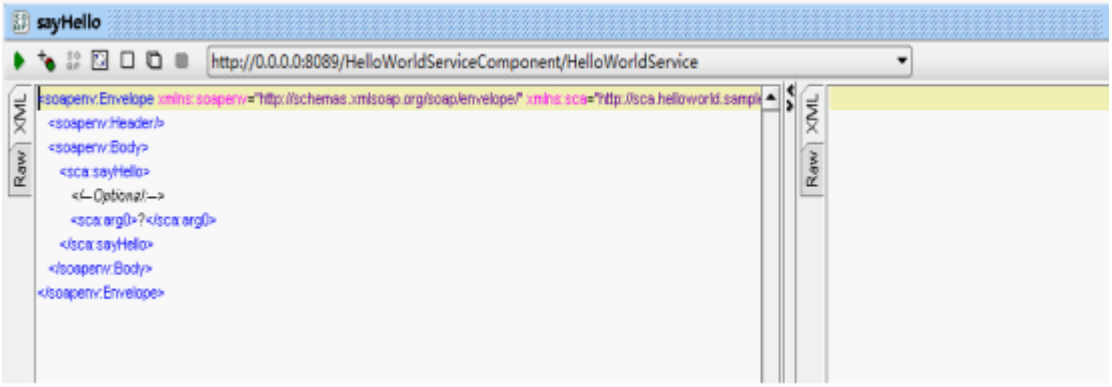


4.3 设置 TestCase 的参数

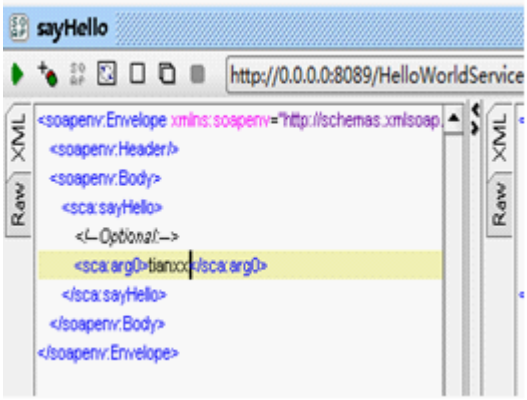
A、创建项目的时候我们选择了 Create sample requests for all operations，所以每个接口方法都会自动创建一个请求，如下图：



B、双击它就可以打开编辑面板，左边是请求内容，右边是响应内容。

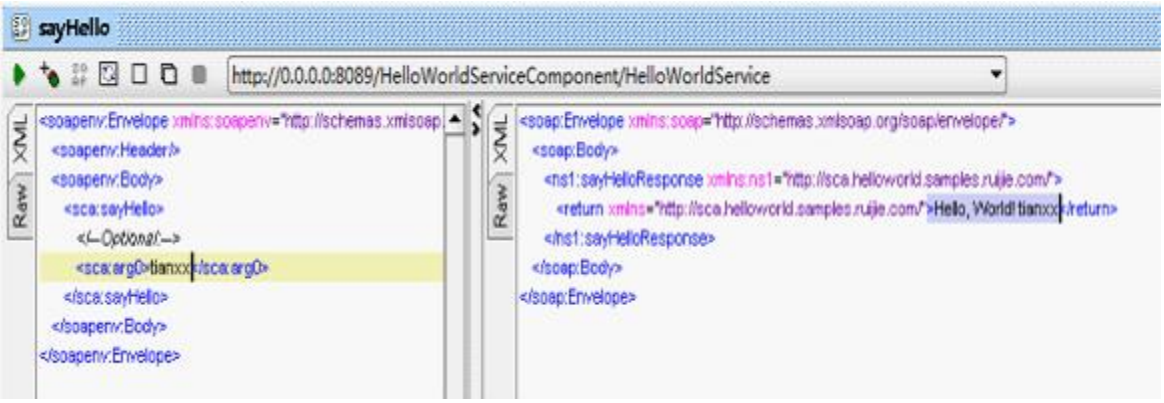


打问号的地方是需要输入的参数，我们可以输入任意参数，这里输入 tianxx，见下图：



4.4 执行 TestCase

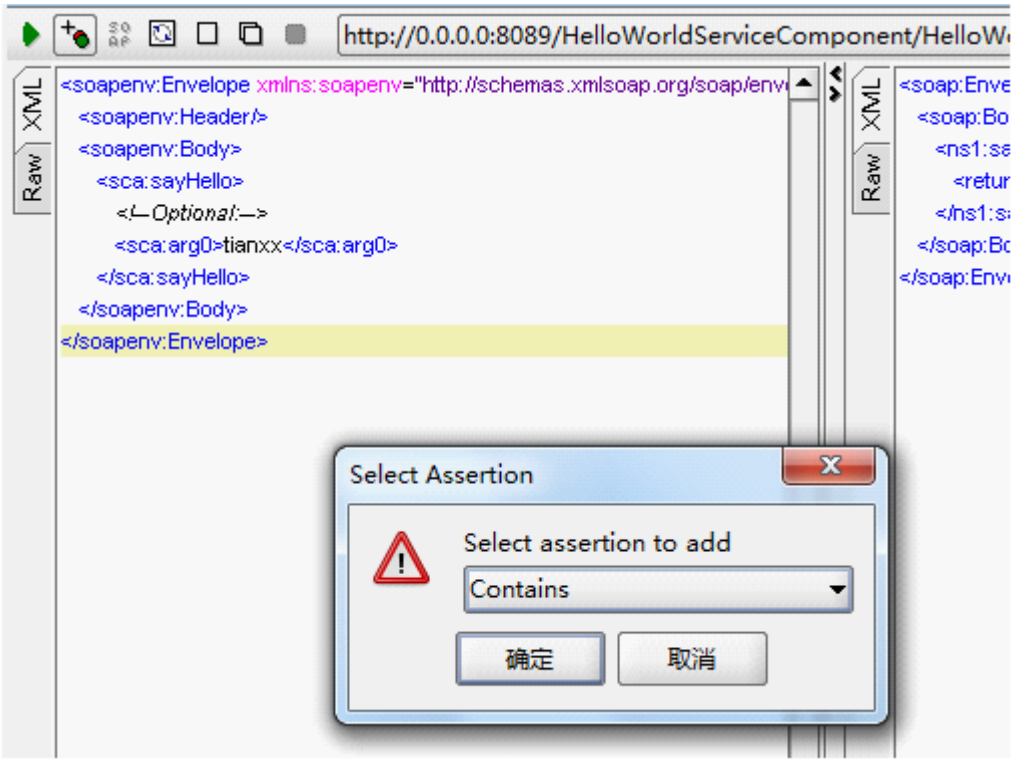
单击  按钮执行，右侧 Response 页面查看结果。



根据反馈的结果判断请求是否发送成功（输入了参数 tianxx，返回了结果 Hello,World! tianxx）。

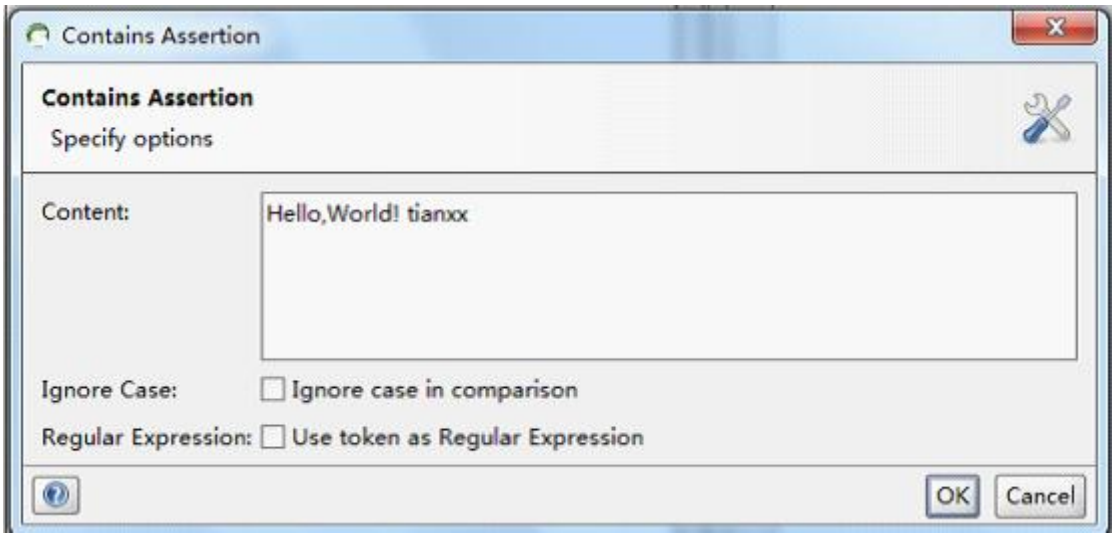
4.5 设置断言

为在测试中不用人为的进行接口功能是否正确的判断，因此加入断言 Assertions。可由程序直接对返回结果进行判断。点击下图中绿色执行按钮后面的“+”来增加断言。

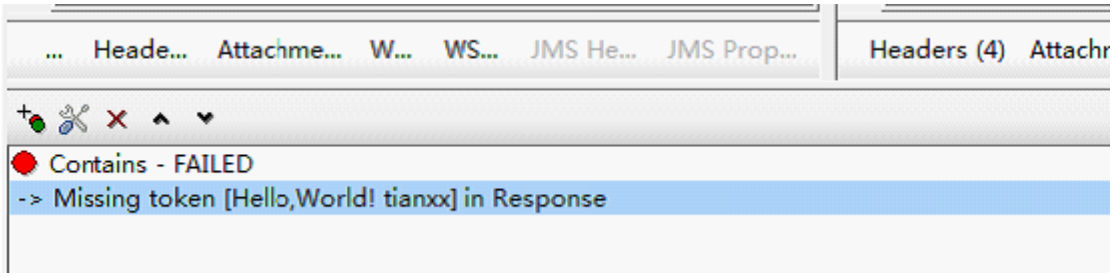


在弹出的“Secect Assertion”对话框中，选择“Contains”的断言，确定后弹出“Contains Assertion”对话框，在 Content 中填入内容，此处是表示返回的结果报文里应包含的字段，我们输入“Hello,World! tianxx”，点击 OK。插入断

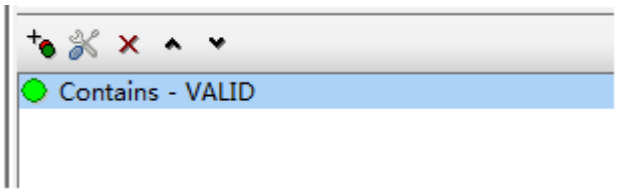
言后，程序会在运行用例时，自动帮我们校验返回的结果报文是否包含“Hello,World! tianxx”内容。



工具下方的结果输出区域显示运行结果如下：



运行失败，因为实际结果应该是“Hello,”和“World!”之间应该有个空格，我们修改（双击 Contains - FAILED）断言，在两个单词之间增加个空格，再次运行，结果如下：



4.6 执行 TestSuit

可以一次执行一个或多个 TestSuit。使用脚本语言 Groovy 控制。

5、总结

5.1 测试方法对比分析

编码方式适合做复杂的功能测试，根据难易程度，方法的使用顺序推荐：url 方式、wsdl 文件方式、wsdl2java 方式（动态、静态）。

soapUI 工具使用简单，但是开源版本功能没有收费版本那么强大，对于要求非常高的测试还是要去具体分析下是否适用，如果是简单测试下基本功能（比如输入下参数值验证下结果）soapUI 还是比较好用的工具。

5.2 后记

本文算是对自己前段工作做的一个笔记或叫总结，不对的地方还请不吝赐教。

我只是一名测试人员

作者：MR.曾

前言：

之前有发过一篇名为水杯测试-误人子弟的博文，很多朋友比较认可，也有部分朋友提出了反对的意见，我看了下，大部分集中在水杯测试考验的是测试员的发散思维能力，那今天我还是写篇博文，目的很直接，水杯测试不能体现出一名测试员的发散思维能力，甚至起到相反的效果，也就是测试能力越强的人，给出的答案越不尽人意，而测试能力较差的往往还能提出一些独特的思维角度，好奇了吗？继续看下去。

正文：

在此，感谢刘德宝刘老师在成都 51 沙龙上说的一句话，“水杯从软件测试的角度能够想到的点，海了去了，而实际上在现实中能倒水，能喝水，没有毒就够了”正是这句话给了我灵感，也期待以后在成都举行的测试沙龙或者聚会，如果有朋友邀请的话我想我没有理由拒绝呵呵。

先不谈测试，我们先来分析一个案例，假设你现在位于成都市九龙宾馆 3 楼某个会议室，这时，我问一个问题。

Q1：你要走出这间会议室，你会怎么做

Q2：你要走出这间宾馆，你会怎么做。

为了避免大家犯职业病，我就直接给答案了，要走出会议室自然是用脚从门口走出去，要走出宾馆，可以走楼梯，也可以走电梯。对这两个答案大家没有异议吧，个别比较优秀的，可以说下跳窗，跳楼等等非常规做法。

Q3：你要离开成都市，你会怎么做。

这个问题是关键，对成都不熟悉的人，他们会很自然的想出很多答案，打的，包车，公交车到车站，坐飞机，坐火车等等。

假如你是一个对成都熟悉的人，你会怎么走。这个地点换一下也可以，就换成你所在的城市也一样可以。

初步估算一下，从成都某个宾馆到离开成都的路线，没有千条也有八百条了，各种小路大路，各种组合，各种交通工具都能达到这样的效果，让你来说，一时半会，你的答案很有可能不如对成都不熟悉的人。没有其他的原因，只因为你对

这个城市太了解了。你所考虑的可能性太多了，多到了你没有办法选择，你甚至不知道到底想到了哪些，你只知道可能性实在太多，太多。

现在我们把这个案例换成测试，会议室和宾馆对应的是测试的范围，不管是做什么行业的软件测试，总是有个范围的，区别只是这个范围大小，而我们要做的只是在这个范围内进行发散，有了局限后，我们发散的思维虽然少但是却很实在，而不是漫无边际，而对水杯测试的范围是比针对离开成都还要大的多的。往往我们不是不知道怎么测，而是不知道该从何说起，从何测起。

你可以从 ERP 的角度去测试它的可管理性，也可以从保险角度测水杯里的水蒸发挥发保温情况，还能从 WEB 测试他的界面，还有手机的扩展，可以说各行各业的测试你都可以放在上面，如果是新人，他们的测试思维尚未成熟，能够给你的答案离不开日常或者离开的有限，再加之水杯测试被誉为经典测试，可想而知网络上似乎已经有标准的答案了，那么你能从新人哪里得到什么信息可想而知，而如果是老人，根据他们的经验，任何测试都是有范围有局限的，不怕局限小就怕局限大，而水杯就属于局限无限大的情况，他们可以考虑到成百上千的测试点，但是需要时间去理清自己的思绪，而面试或者笔试的场合是不允许这种情况的，即处于工作 1-3 年测试经验的人对于水杯测试的答案未必比新人好，加之思维惯性对测试需求的依赖性还有个人风格的特性，他们可能比新人的答案还不如，比如我自己来回答水杯测试，我心里就很抵触，即使我知道网络上的答案即使我也有我自己的测试风格和思维特性，但是我不想答，或者不愿意去想，遇见的几次这道题目，我要么是敷衍过去要么直接不答。

我个人比较喜欢高效率的去完成事情，也喜欢高质量的去利用时间，对于这类无意义，无价值的题目我确实是无法理解面试官的用意是为何。

国内的测试行业已经发展了有一段时间了，有 10 年测试经验的人也有很多了，但我相信即使是 10 年，20 年测试经验的人也不敢说自己精通测试，试问有多少行业？有多少类型的软件？即使你很厉害半年吃透一种业务半年换个工作，10 年，也不过是 20 种业务，相比软件行业不过是沧海一粟，谈何精通，排除业务不谈，单从技术来讲，测试技术与日俱增，数据库，语言，脚本，各种工具，用例的设计方法，矩阵等等，谁敢说自己精通？我们能做的只是精通测试职业中的一种或几种而已。对于测试，没有人敢说自己精通。

而一位做了 3 年 WEB 测试的测试员去应聘一个手机测试的职位，在面试时发现这一道题，请你根据水杯测试设计用例，我来假想一下，这名测试员根据自己 3 年的 WEB 经验，设计了很多很精彩的用例包括，界面是否美观，色彩是否友好，水杯是否方便运用，水杯的握把是否符合人体学等等从 WEB 方面散发出来的思想，而面试官没有做过 WEB 测试，但面试官有 5 年的手机测试的经验，隔行如隔山，这位面试官没有办法理解为什么要做这些测试，而他希望的答案是从手机行业出发，手机应用软件的跨平台使用上，于各种其他软件的兼容性上等方向的用例，这样的结果是否很好笑，两人之间完全是牛头不对马嘴。

这样的事未必就是不可能的，相反我觉得可能性还比较大，这段时间接触过几次面试或笔试，虽然没怎么遇见水杯测试，但是却发现一件事，面试还好，由人控制，而笔试，工作一年的是那份题，应届生是那份题，工作十年的还是那份题，测试路上走的越远的人，离测试的标准答案就越远，我一直深以为然，在校时，遇见的测试题目都是有唯一的标准的答案的，工作后我发现完全不是那么回事，学校说，测试需要依赖文档，工作中没有文档就不测了吗？学校说，测试用例讲究少用例覆盖多面积，那产品测试中的用例颗粒度怎么说？学校中要学 QTP，LR 工作中用到的可能是 SELENIUM，或者干脆自己公司开发的工具，1 年的人说测试是找 BUG，5 年的人说测试是控制项目的质量，10 年的人说测试就是测试。

手机测试的笔试题上，16 道题测试占 4 道，4 道系统题，4 道代码题，4 道数据库题 oracle，

应届生或许都能答上，毕竟刚丢下书本没多久，做了 WEB 测试 5 年的人，能答上数据库，其余的很有悬念，测试题只能说不留空，毕竟 5 年已经足够一个人形成自己的测试理念。无疑单从成绩上来讲恐怕这名 5 年的测试员会不如应届生，但这能证明什么？

最后，刘老师在沙龙上透露了一个情况，很多企业埋怨招不到人，我想对这些企业说，不是你们招不到人，而是你们于有能力的人擦肩而过，测试是个特殊的行业，特殊在范围，手机测试是测试，WEB 测试是测试，嵌入式是测试，只要有程序的影子都是测试，不要单一的用技术来衡量一名测试员的水平，如果要衡量请用测试本身来衡量，不要问你会什么什么语言，你会什么什么系统，测试行业的特殊性决定了我们可以学习多方面的姿势但并不需要去深入了解，测试本

身的能力强硬，其他的技术只是辅助测试的一种手段，对测试有深入理解的人，往往会给你们惊喜，我们不需要去了解代码的结构，不需要去了解程序设计的思想，也不需要去了解工具怎么做出来的，只需要了解这段代码是什么意思，这个工具怎么用，怎么用这些技术来达到测试的效果，仅此而已，对测试的要求不应该是技术，而是学习能力，因为我们不需要精通，只需要能用就足够。

埋怨招不到人的企业，不如调整下你们招人的思路，100 名测试员，有一半是符合你们的经验要求的，1/4 是符合你们行业的要求的，1/8 是符合你们技术要求的，1/16 是真的需要跳槽的，1/32 是希望进入贵公司的。64/1 是没有被你们对测试错误的理解而吓走的，试问是招不到测试的人呢？还是在将好的测试员往门外推呢。

合格的测试员，是能够短时间内能够使用一门技术工作的人，这一点除了测试员应该了解，也是希望招到好测试员应该了解的。

不要说要自学或者要怎么怎么样，这样的做法是很愚蠢的，只是浪费时间，试问一名正在从事手机测试的测试员，他去自学 QTP 等 WEB 测试工具，你认为他的时间是哪里来的？没错挤出来的，一天 24 小时，8 小时工作，8 小时睡觉，4 小时吃饭上班下班的路上，如果没有加班，还有 4 小时的时间，再排除我们一些生活琐事，我们还能有 1-2 个小时的时间可以控制，这 1-2 小时你可以选择学习 QTP 也可以选择学习手机测试相关的知识，选择前者，你的生活势必劳累，并且工作技术上勉强跟的上部队，选择学习手机测试相关的，第一天学习的东西第二天就能投入使用，工作上的表现必然出类拔萃，这是一个很简单的选择题，另外，说实话不管是学习 QTP 还是学习手机，只要学习我都认为，这样的人很优秀，只是得不到应得的对待方式，工具也好，技术也罢，都是为了测试工作，都只是测试职业的一种辅助存在，不要本末倒置。

我只会为了工作而去学习，我所学习的必然是我现在所在工作的行业，脚踏实地不好高慕远这是我认为的成功的标准之一。

请用我们测试的知识，来笔试或者面试我们，测试员不是神，只是一个职业，我们有义务有责任去学好我们测试的知识，有义务去学好我们工作中要用到的技术，但是作为面试者，作为一个尚未进贵公司的人员，我们真的没有义务去放弃我们现在的工作，放弃我们现在的方向而去满足你们的要求的。

我们只是一群测试人员，不要对我们要求太高，更不要把我们当做万能的，我们不是变形金刚更不是神。

关于编写测试用例的几点思考

作者：王洪臣

从事测试工作的各位同仁，对怎么写一个测试用例再也熟悉不过了。测试用例人人会写，但什么是好的用例，怎样去设计一个实用且规范的用例，可能就见仁见智了。下面我根据自己的理解，并结合在工作中的实际情况来谈一下对于测试用例写作的几点思考。

首先，编写测试用例的前提是基于对需求的充分理解。从哪些方面去获取详尽的需求呢？如果公司研发流程比较规范的话，在测试准备阶段，测试人员就能拿到一份关于需求的说明文档，即软件规格说明书。那么，软件规格说明书就是我们获取需求的首要途径。通过研读规格说明书，大概了解下所测系统的测试项目，再结合质量模型的相关特性，测试人员就可以初步确定从哪几个方面来进行用例的设计，比如功能、性能和可靠性等方面。如果没有相关的需求文档，那我们就通过与那些熟悉需求的人员进行交流，如开发人员，需求分析人员等，来获取被测系统的详细需求。测试类型确定以后，测试人员就可以进行用例的设计工作了。

接下来，谈一谈用例设计的方法。拿到需求规格以后，可以将其划分为几个相对独立的需求片断。对每个需求片断，采用不同的用例设计方法来进行用例的设计。常用的几种黑盒用例设计方法有等价类边界值，判定表，因果图，状态转移与流程分析等。每一种用例的设计方法都有其适用的情况，也有不适合的情况。这个要根据具体的需求片断来进行筛选。

最后，是关于测试用例的写作。按照上面设计出来的内容，采用文档化的形式表现出来，就是用例的写作过程。拿功能测试类型来说，对于一个具体的系统功能测试项目，如果该测试项目功能比较复杂，我们可以把它再进一步细分为几个测试子项目。对于每个测试子项，来设计若干个用例来进行测试。用例的编号可以参照方案中规定的格式来编写，要确保编号的唯一性和易识别等特性，至少要保证在我们测试组内风格要统一。测试标题可以按照所测子项来进行标识，主要用来说明该用例所测的测试点。然后是设计测试数据及测试步骤。在进行步骤的设计过程中，应该使用简洁明了的语言把过程叙述清楚，确保用例的执行人员

能够完全理解。最后是预期结果，也就是该用例的检查点。检查点说明也要详尽，确保执行人员能够判断出该用例是否执行成功。

用例编写完成以后，测试小组内应该相互评审下，主要检查用例对功能、性能等类型的覆盖情况，用例编写风格是否统一等。评审过后，再进行相应的修改。之后再组织与开发和需求分析人员的用例评审，来确保用例达到公司的要求。在后续的测试执行阶段，如果发现测试数据等问题，测试人员应该及时的修改测试用例，并通知给有关的执行人员。如果发生了需求变更，也要增加或者修改用例，始终要保证用例与用户需求的一致性。

关于集成、互操作性、兼容性与可移植性测试的辩析

作者：王晓芹

在软件测试领域，由于集成测试、互操作性测试、兼容性测试与可移植性测试四个概念意思相近，很容易让人产生混淆。本文提供了对四个术语的解释，以便大家对它们的理解。

要解释以上四个术语，首先需要了解两个概念：组件和系统。组件是构成系统的部分；而系统是组件的集合，组件组织在一起后完成一个特定的功能或一套功能。

1、集成

集成关注的是将各组件组成一个完整系统的过程。在软件中，我们通常所关注的集成包括两个层面，首先是模块级集成，有时也被认为是组件集成测试或小集成，其二是系统集成，有时也被叫作集成测试或大集成。

集成测试关注的不仅仅是组件之间的接口是否得到实现，而且还关注集成后的组件（已形成系统）是否按照既定要求工作，这种要求涉及集成系统的功能和非功能方面。

图 1 显示了两个互相作用的组件形成一个集成系统，在这里，集成测试关心的是两个组件结合起来后所形成的集成系统作为一个整体后是否能按预期的行为进行工作。

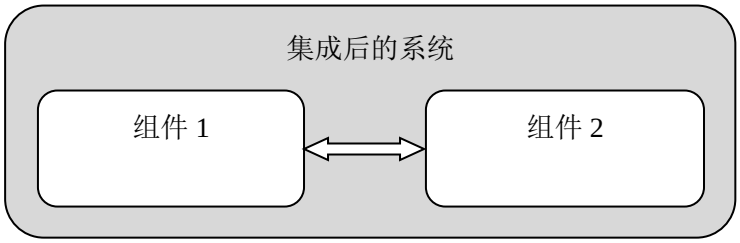


图 1 集成

举个一个集成测试的例子，比如对光圈控制组件和快门控制组件进行集成测试，这里的集成测试就是为了确保他们可以作为照相机控制系统的一部分能够正确地执行。

2、互操作性

互操作性是两个或多个系统（组件）进行交换、使用信息的能力。因此，互操作性关注的是系统之间进行通信的能力，并且它要求被交互的信息可被接收系统所理解，但是它不关注通信的双方作为整体做的其它事情。两个组件/系统的互操作性可能是好的，但这两个组件/系统作为整体是否能发挥很好的功能则与互操作性无关。因此互操作性只涉及接口，而不涉及通信组件/系统的双方作为整体功能的实现，可以说互操作性测试是集成测试的子集。

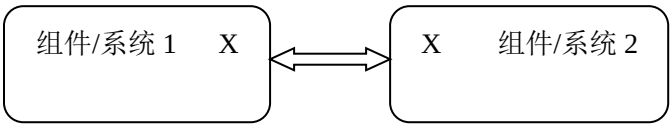


图 2 可互操作性

图 2 显示了带有接口的两个通信组件/系统在各自的系统里处理信息，其中在标识有 X 点的位置提供了收发系统所使用的信息，互操作测试只限于检测这些信息是否正确地从一个源系统发出并且以正确地状态到达另一个系统。

举一个互操作性测试的例子，两个独立的的订票系统进行航班信息传递，互操作性测试将验证这些信息是否到达了目标系统，并且要保证目标系统接收到的信息含义与源系统相同，但它并不测试目标系统接下来是否以合理的方式使用这个订票信息。

3、兼容性

兼容性测试关注的是两个或多个系统或组件在共享同一环境时是否能正确执行既定功能，这两个组件不需要相互通信，它们只是简单地驻留于同一个运行环境，因此，兼容性不关心互操作性。两个组件或系统可以执行完全独立的功能，既便两个组件集成在一起不能够正确执行，它们也是兼容的。

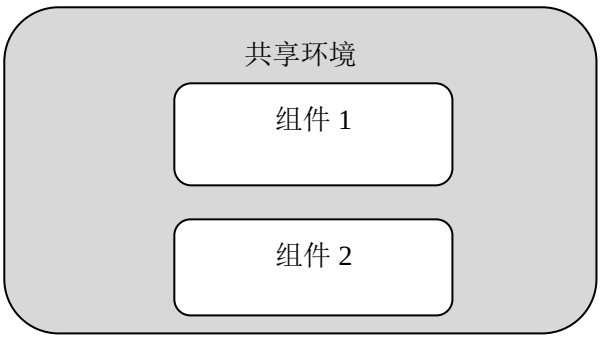


图 3 兼容性

图 3 显示的是同一环境下的两个组件，只要两者能在环境中运行（简单驻留），不冲突、并且不影响另一方的行为，它们就互相兼容。比如大家所熟知的字处理程序和计算器程序，它们是两个具有独立功能的应用程序，只要他们能在一个 PC 机上同时正确工作，它们就是兼容的。

4、可移植性

可移植性测试关注的是将组件或系统在不同环境下进行移动时的难易程度。如图 4，组件 X 被放置在两个不同的环境下，只要组件 X 能在这两个不同环境下工作，就被认为是可以移植的组件。

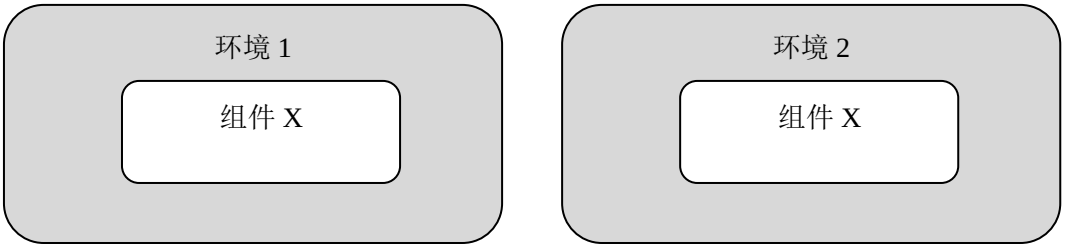


图 4 可移植性

例如，对于一个工作于 WIN 98 及 PC 机环境下的计算机游戏程序，然后验证它是否也可工作于 WIN XP 的 PC 机上，这就是可移植性测试。