web 应用程序测试方法和测试技术详述

1. 概述

随着 web 应用的增多，新的模式解决方案中以 web 为核心的应用也越来越多， 很多公司各种应用的架构都以 B/S 及 web 应用为主，但是有关 WEB 测试方面的内容并没有相应的总结，所以我在这里对 web 的测试方法和采用的测试技术进行总结，便于内部交流。

测试方法尽量涵盖 web 程序的各个方面，测试技术方面在继承传统测试技术的技术上结合 web 应用的特点。

l 相关的测试和实现技术也有着很大的关系，由于本公司使用 J2EE 体系，也许例子中只有 JAVA 平台可以使用，.NET 平台测试技术暂时不涉及，如果你有请与我联系。

2. 测试方法

说明：测试方法的选择取决你的测试策略。

一般的 web 测试和以往的应用程序的测试的侧重点不完全相同，基本包括以下几个方面。

当然圆满的完成测试还要有好的团体和流程等的方方面面的支持，你同样应该对这些方面进行注意。

有些测试方法设计到了流程，哪些应该在你的测试团队建设中建立。

2.1 界面测试

现在一般人都有使用浏览器浏览网页的经历，用户虽然不是专业人员但是对界面效果的印象是很重要的。如果你注重这方面的测试，那么验证应用程序是否易于使用就非常重要了。很多人认为这是测试中最不重要的部分，但是恰恰相反界面对不懂技术的客户来说那相当关键，慢慢体会你会明白的。

方法上可以根据设计文档，如果够专业的话可以专业美工人员，来确定整体风格页面风格，然后根据这个可以页面人员可以生成静态的 HTML，CSS 等甚至生成几套不用的方案来讨论，或者交给客户评审，最后形成统一的风格的页面/框架。注意不要靠程序员的美术素养形成你的 web 风格，那样可能会很糟糕。

主要包括以下几个方面的内容：

- Ø 站点地图和导航条 位置、是否合理、是否可以导航等内容布局 布局是否合理，滚动条等简介说明 说明文字是否合理，位置，是否正确；
- Ø 背景/色调 是否正确、美观，是否符合用户需求；
- Ø 页面在窗口中的显示是否正确、美观（在调整浏览器窗口大小时，屏幕刷新是否正确）表单样式 大小，格式，是否对提交数据进行验证（如果在页面部分进行验证的话）等；
- Ø 连接 连接的形式，位置，是否易于理解等。

web 测试的主要页面元素

- Ø 页面元素的容错性列表（如输入框、时间列表或日历）；
- Ø 页面元素清单（为实现功能，是否将所需要的元素全部都列出来了，如按钮、单选框、复选框、列表框、超连接、输入框等等）；
- Ø 页面元素的容错性是否存在；
- Ø 页面元素的容错性是否正确；
- Ø 页面元素基本功能是否实现（如文字特效、动画特效、按钮、超连接）；

- Ø 页面元素的外形、摆放位置（如按钮、列表框、核选框、输入框、超连接等）；
- Ø 页面元素是否显示正确（主要针对文字、图形、签章）；
- Ø 元素是否显示（元素是否存在）；
- Ø 页面元素清单（为实现功能，是否将所需要的元素全部都列出来了，如按钮、单选框、复选框、列表框、超连接、输入框等等）。

测试技术

- Ø 通过页面走查，浏览确定使用的页面是否符合需求。可以结合兼容性测试对不用分辨率下页面显示效果，如果有影响应该交给设计人员提出解决方案；
- Ø 可以结合数据定义文档查看表单项的内容，长度等信息；
- Ø 对于动态生成的页面最好也能进行浏览查看。如 Servelet 部分可以结合编码规范，进行代码走查。是否支持中文，如果数据用 XML 封装要做的工作会多一点等等。

2.1.l 界面测试要素:

符合标准和规范,灵活性,正确性,直观性,舒适性,实用性,一致性

2.1.l.1 直观性:

- Ø 用户界面是否洁净,不唐突,不拥挤.界面不应该为用户制造障碍.所需功能或者期待的响应应该明显,并在预期出现的地方；
- Ø 界面组织和布局合理吗?是否允许用户轻松地从一个功能转到另一个功能?下一步做什么明显吗?任何时刻都可以决定放弃或者退回,退出吗?输入得到承认了吗?菜单或者窗口是否深藏不露?
- Ø 有多余功能吗?软件整体抑或局部是否做得太多?是否有太多特性把工作复杂化了?是否感到信息太庞杂?
- Ø 如果其他所有努力失败,帮助系统真能帮忙吗?

2.1.l.2 一致性

- Ø 快速键和菜单选项.在 Windows 中按 F1 键总是得到帮助信息；
- Ø 术语和命令.整个软件使用同样的术语吗?特性命名一致吗?例如,Find 是否一直叫 Find,而不是有时叫 Search?
- Ø 软件是否一直面向同一级别用户?带有花哨用户界面的趣味贺卡程序不应该显示泄露技术机密的错误提示信息；
- Ø 按钮位置和等价的按键.大家是否注意到对话框有 OK 按钮和 Cancle 按钮时,OK 按钮总是在上方或者左方,而 Cancle 按钮总是在下方或右方?同样原因,Cancle 按钮的等价按键通常是 Esc,而选中按钮的等价按钮通常是 Enter.保持一致。

2.1.l.3 灵活性

- Ø 状态跳转：灵活的软件实现同一任务有多种选择方式；
- Ø 状态终止和跳过：具有容错处理能力；
- Ø 数据输入和输出：用户希望有多种方法输入数据和查看结果.例如,在写字板插入文字可用键盘输入,粘贴,从 6 种文件格式读入,作为对象插入,或者用鼠标从其他程序拖动。

2.1.l.4 舒适性

- Ø 恰当：软件外观和感觉应该与所做的工作和使用者相符；
- Ø 错误处理：程序应该在用户执行严重错误的操作之前提出警告,并允许用户恢复由于错误操作导致丢失的数据，如大家认为 undo /redo 是当然的；

Ø 性能：快不见得是好事，要让用户看得清程序在做什么，它是有反应的。

2.2 功能测试

功能测试是测试中的重点，主要包括一下几个方面的内容：

Ø 连接：这个连接和界面测试中的连接不同。那里注重的是连接方式和位置，如是图像还是文字放置的位置等，还是其他的方式；这里的连接注重功能，如是否有连接，连接的是否是说明的位置等；

Ø 表单提交：应当模拟用户提交，验证是否完成功能，如注册信息，要测试这些程序，需要验证服务器能正确保存这些数据，而且后台运行的程序能正确解释和使用这些信息。还有数据正确性验证，异常处理等，最好结合易用性要求等。B/S 结构实现的功能可能主要的就在这里，提交数据，处理数据等如果有固定的操作流程可以考虑自动化测试工具的录制功能，编写可重复使用的脚本代码，可以在测试、回归测试时运行以便减轻测试人员工作量；

Ø Cookies 验证：如果系统使用了 cookie，测试人员需要对它们进行检测。如果在 cookies 中保存了注册信息，请确认该 cookie 能够正常工作而且已对这些信息已经加密。如果使用 cookie 来统计次数，需要验证次数累计正确。关于 cookie 的使用可以参考浏览器的帮助信息。如果使用 B/S 结构 cookies 中存放的信息更多；

Ø 功能易用性测试 完成了功能测试可以对应用性进行了解，最好听听客户的反映，在可以的情况下对程序进行改进是很有必要的，和客户保持互动对系统满意度也是很有帮助的。

2. 2.1 测试技术

功能测试的测试技术可是很多的，我们可以结合实际环境选择使用。

Ø 白盒测试技术(White Box Testing)： 深入到代码一级的测试，使用这种技术发现问题最早，效果也是最好的。该技术主要的特征是测试对象进入了代码内部，根据开发人员对代码和对程序的熟悉程度，对有需要的部分进行在软件编码阶段，开发人员根据自己对代码的理解和接触所进行的软件测试叫做白盒测试。这一阶段测试以软件开发人员为主，在 JAVA 平台使用 Xunit 系列工具进行测试，Xunit 测试工具是类一级的测试工具对每一个类和该类的方法进行测试。

Ø 黑盒测试技术（Black Box Testing）：黑盒测试的内容主要有以下几个方面，但是主要还是功能部分。主要是覆盖全部的功能，可以结合兼容，性能测试等方面进行，根据软件需求，设计文档，模拟客户场景随系统进行实际的测试，这种测试技术是使用最多的测试技术涵盖了测试的方方面面，可以考虑以下方面

- 正确性 (Correctness)：计算结果，命名等方面。
- 可用性 (Usability)：是否可以满足软件的需求说明。
- 边界条件 (Boundary Condition)：输入部分的边界值，就是使用一般书中说的等价类划分，试试最大最小和非法数据等等。
- 性能 (Performance)： 正常使用的时间内系统完成一个任务需要的时间，多人同时使用的时候响应时间在可以接受范围内。J2EE 技术实现的系统在性能方面更是需要照顾的，一般原则是 3 秒以下接受，3-5 秒可以接受，5 秒以上就影响易用性了。如果在测试过程中发现性能问题，修复起来是非常艰难的，因为这常常意味着程序的算法不好，结构不好，或者设计有问题。因此在产品开发的开始阶段，就要考虑到软件的性能问题
- 压力测试 (Stress)： 多用户情况可以考虑使用压力测试工具，建议将压力和性能

测试结合起来进行。如果有负载平衡的话还要在服务器端打开监测工具,查看服务器 CPU 使用率，内存占用情况，如果有必要可以模拟大量数据输入，对硬盘的影响等等信息。如果有必要的话必须进行性能优化(软硬件都可以)。这里的压力测试针对的是某几项功能。

- 错误恢复 (Error Recovery)：错误处理，页面数据验证，包括突然间断电，输入脏数据等。
- 安全性测试(Security)：这个领域正在研究中，防火墙、补丁包、杀毒软件等的就不必说了，不过可以考虑。破坏性测试时任意看了一些资料后得知,这里面设计到的知识\内容可以写本书了,不是一两句可以说清的，特别是一些商务网站，或者跟钱有关，或者和公司秘密有关的 web 更是需要这方面的测试，在外国有一种专门干这一行的人叫安全顾问，可以审核代码，提出安全建议，出现紧急事件时的处理办法等，在国内没有听说哪里有专门搞安全技术测试的内容。
- 兼容性 (Compatibility)：不同浏览器，不同应用程序版本在实现功能时的表现不同的上网方式，如果你测试的是一个公共网站的话。

兼容性测试内容详述：

Ø 硬件平台

Ø 浏览器软件和版本：浏览器插件，浏览器选项，视频分辨率和色深，文字大小，调制解调器速率.

- 软件配置 (Configuration)：如 IE 浏览器的不用选项-安全设定最高，禁用脚本程序等等，你们的程序在各种不用的设置下表现如何。

2. 2.2 单元测试技术(Unit Test):

2. 2.2.1 下面是对白盒测试和单元测试的区别的论述:

单元测试和白盒测试是不同的，虽然单元测试和白盒测试都是关注功能，他们都需要代码支持，但是级别不同。白盒测试关注的是类中一个方法的功能，是更小的单位，但是完成一个单元测试可能需要 N 多类，所以说作单元测试需要写驱动和稳定桩，比如查询单元是一个查询包，包括 N 多的测试类、测试数据，运行他需要提供数据的部分，输入参数和发出命令的驱动等等，是比类大的一个整体进行的。

另一个明显的区别是白盒测试不会关注类接口,但是单元测试主要的内容就是类接口测试。

不过很多时候是很少区分的，因为这两种技术实现起来有很多相互关联的部分。不过要看你对质量的关注程度来决定。

2. 2.2.2 功能测试边界测试\越界测试技术详述

Ø 边界条件

边界条件是指软件计划的操作界限所在的边缘条件，如果软件测试问题包含确定的边界，那么数据类型可能是：数值、速度、字符、地址、位置、尺寸、数量。同时，考虑这些类型的下述特征：第一个/最后一个、最小值/最大值、开始/完成、超过/在内、空/满、最短/最长、最慢/最快、最早/最迟、最大/最小、最高/最低、相邻/最远。

Ø 越界测试

通常是简单加 1 或者很小的数(对于最大值)和减少 1 或者很小的数(对于最小值)，例如：第一个减 1/最后一个加 1、开始减 1/完成加 1、空了再减/满了再加、慢上加慢/快上加快、最大数加 1/最小数减 1、最小值减 1/最大值加 1、刚好超过/刚好在内、短了再短/长了再长、早了更早/晚了更晚、最高加 1/最低减 1。

另一些该注意的输入：默认，空白，空值，零值和无；非法，错误，不正确和垃圾数据。

2. 2.2.3 状态测试技术

Ø 软件可能进入的每一种独立状态；

Ø 从一种状态转入另一种状态所需的输入和条件；

Ø 进入或退出某种状态时的设置条件及输入结果。

具体测试方法可以参考如下：

Ø 每种状态至少访问一次；

Ø 测试看起来最常见最普遍的状态转换；

Ø 测试状态之间最不常用的分支；

Ø 测试所有错误状态及其返回值；

Ø 测试随机状态转换。

2. 2.2.4 竞争条件测试技术

竞争条件典型情形参考如下:

Ø 两个不同的程序同时保存或打开同一个文档；

Ø 共享同一台打印机,通信端口或者其他外围设备；

Ø 当软件处于读取或者修改状态时按键或者单击鼠标；

Ø 同时关闭或者启动软件的多个实例；

Ø 同时使用不同的程序访问一个共同数据库。

2. 2.3 负载\压力测试(StressTest)

在这里的负载\压力和功能测试中的不同，他是系统测试的内容，是基本功能已经通过后进行的。可以在集成测试阶段，亦可以在系统测试阶段进行。

使用负载测试工具进行，虚拟一定数量的用户看一看系统的表现，是否满足定义中的指标。

负载测试一般使用工具完成，loadrunner，webload，was，ewl，e-test 等，主要的内容都是编写出测试脚本，脚本中一般包括用户常用的功能，然后运行，得出报告。所以负载测试包括的主要内容就不介绍了。

负载测试技术在各种极限情况下对产品进行测试 (如很多人同时使用该软件，或者反复运行该软件)，以检查产品的长期稳定性。例如，使用压力测试工具对 web 服务器进行压力测试。本项测试可以帮助找到一些大型的问题，如死机、崩损、内存泄漏等，因为有些存在内存泄漏问题的程序，在运行一两次时可能不会出现问题，但是如果运行了成千上万次，内存泄漏得越来越多，就会导致系统崩滑。用 J2EE 实现的系统很少但是并不是没有内存问题。

Ø 无论什么工具基本的技术都是利用线程技术模仿和虚拟用户，在这里主要的难点在与测试脚本的编写，每种工具使用的脚本都不一样，但是大多数工具都提供录制功能就算是不会编码的测试人员同样可以测试。

Ø 对负载工具的延伸使用可以进行系统稳定性测试，系统极限测试，如使用 100 的 Load Size 连续使用 24 小时，微软定义的通过准则是通过 72 小时测试的程序一般不会出现稳定性的问题。

2. 2.4 回归测试 (Regression Test)

过一段时间以后，再回过头来对以前修复过的 Bug 重新进行测试，看该 Bug 是否会重新出现。

Ø 回归测试技术可以在测试的各个阶段出现，无论是单元测试还是集成测试还是系统测试。是对以前问题进行验证的过程。

Ø 回归测试的目的就是保证以前已经修复的 Bug 不会再出现。实际上，许多 Bug 都是在回归测试时发现的，在此阶段，我们首先要检查以前找到的 Bug 是否已经更正了。值得注意

的是，已经更正的 Bug 也可能又回来了，有的 Bug 经过修改之后可能又产生了新的 Bug。所以，回归测试可保证已更正的 Bug 不再重现，不产生新的 Bug。

2. 2.5 Alpha 和 Beta 测试 (Alpha and Beta Test):

在正式发布产品之前往往要先发布一些测试版，让用户能够反馈出相关信息，或者找到存在的 Bug，以便在正式版中得到解决。

特别是在有客户参加的情况下，对系统进行测试可能会出现一些我们没有考虑的情况，还可以解决一些客户实际关心的问题。

不同的测试技术区分

2.3 覆盖测试技术

说明:测试覆盖率可以看出测试的完成度，在测试分析报告中可以作为量化指标的依据，测试覆盖率越高效果越好。

覆盖测试可以是程序代码的执行路径覆盖，亦可以是功能实现的步骤覆盖（可以理解成流程图的路径覆盖）。

该技术可以用在任何测试阶段，包括单元测试、集成测试、系统测试。

使用该技术时可以使用以上的任何测试方法和测试技术。

2.4 白盒测试和黑盒测试技术

白盒测试技术 (White Box Testing)：该技术主要的特征是测试对象进入了代码内部，根据开发人员对代码和对程序的熟悉程度，对有需要的部分进行测试。在软件编码阶段，开发人员根据自己对代码的理解和接触所进行的软件测试叫做白盒测试。这一阶段测试以软件开发人员为主，使用 Xunit 系列工具进行测试，可以包括很多方面如功能性能等。

黑盒测试 (Black Box Testing)：测试的主体部分，黑盒测试的内容主要有以下几个方面，但是主要还是功能部分。主要是覆盖全部的功能，可以结合兼容，性能测试等方面进行，包括的不同测试类型请参考以上内容。

2.5 手工测试和自动化测试

手工测试 （Manual Testing）：即依靠人力来查找 Bug。方法可以参考上边的测试，也可以根据对实现技术及经验等进行不同的测试。

自动测试 （Automation Testing）：使用有针对工具实行。可以作出自动化测试的计划，对可以进行自动化测试的部分编写或者录制相应的脚本，可以加入功能，容错，表单提交等，可以参考 MI,Rational 或者其他类测试工具说明。

根据权威的软件测试经验，手工测试还是主要的测试方法，自动测试不够灵活，在这里不再详述。微软的测试过程 80％还是手工完成。

自动测试永远也代替不了手工测试，但是手工测试的工作量很大是不争的事实。

2.6 根据 RUP 标准按阶段区分测试

单元测试：在上边有详细的叙述，还有针对单元测试和集成测试的论述，请参考。

集成测试：分为功能集成测试和系统集成测试，相互有调用的功能集成，在系统环境下功能相互调用的影响等，使用方法可以任意选用上面的内容。注重功能方面。

系统测试：在功能实现的基础上，可以加入兼容性，易用性，性能等等。

验收测试：可以包括 Alpha 和 Beta 测试，在这里就不再详述。

3. 存在风险及解决方法

说明：测试不能找出所有的问题，只是尽量在开发阶段解决大多数的问题而已。

测试风险如下：

软硬件的测试环境提供上也对测试结果有很大的影响；

测试团队的水平，经验，合作效果等；
整个开发流程对测试的重视程度，测试的进入时间等；
由于测试环境操作系统，网络环境，带宽等情况可能产生的测试结果可能不同这是就需要经验以及对测试环境的保护等方面下一些功夫。

4. 软件缺陷的原则

软件缺陷区别于软件 bug,它是在测试过程中出现的对系统有影响的，但是在设计中没有的或者对修改后的 bug 测试和开发人员有不同意见等。

Ø 软件未达到产品说明书标明的功能；
Ø 软件出现了产品说明书指明不会出现的错误；
Ø 软件功能超出产品说明书指明范围；
Ø 软件未达到产品说明书虽未指出但应达到的目标；
Ø 软件测试员认为软件难以理解、不易使用、运行速度缓慢，或者最终用户认为不好。

5. 文档测试

产品说明书属性检查清单

Ø 完整：是否有遗漏和丢失？完全吗？单独使用是否包含全部内容？
Ø 准确：既定解决方案正确吗？目标明确吗？有没有错误？
Ø 精确：不含糊，清晰。描述是否一清二楚？还是自说自话？容易看懂和理解吗？
Ø 一致：产品功能描述是否自相矛盾？与其他功能有没有冲突?
Ø 贴切：描述功能的陈述是否必要?有没有多余信息?功能是否满足的客户要求?
Ø 合理：在特定的预算和进度下，以现有人力，物力和资源能否实现?
Ø 代码无关：是否坚持定义产品，而不是定义其所信赖的软件设计，架构和代码?
Ø 可测试性：特性能否测试?测试员建立验证操作的测试程序是否提供足够的信息?

产品说明书用语检查清单

说明：对问题的描述通常表现为粉饰没有仔细考虑的功能----可归结于前文所述的属性。从产品说明书上找出这样的用语，仔细审视它们在文中是怎样使用的。产品说明书可能会为其掩饰和开脱,也可能含糊其词----无论是哪一种情况都可视为软件缺陷。

Ø 总是，每一种，所有，没有，从不。如果看到此类绝对或肯定的，切实认定的叙述，软件测试员就可以着手设计针锋相对的案例。
Ø 当然，因此，明显，显然，必然。这些话意图诱使接受假定情况，不要中了圈套。
Ø 某些，有时，常常，通常，惯常，经常，大多，几乎。这些话太过模糊，"有时"发生作用的功能无法测试。
Ø 等等，诸如此类，依此类推。以这样的词结束的功能清单无法测试，功能清单要绝对或者解释明确，以免让人迷惑，不知如何推论。
Ø 良好，迅速，廉价，高效，小，稳定。这些是不确定的说法，不可测试。如果在产品说明书中出现，就必须进一步指明含义。
Ø 已处理，已拒绝，已忽略，已消除。这些廉洁可能会隐藏大量需要说明的功能。
Ø 如果...那么...(没有否则)。找出有"如果...那么..."而缺少配套的"否则"结构的陈述，想一想"如果"没有发生会怎样。