1. Web 测试中关于登录的测试

请问，你为自己写过的用例怀疑过吗？

前两天听一个朋友说他同事写了 100 个用例，结果有 92 个是无效的，差点被公司开了，本人以前也写过不少用例，但现在忽然怀疑我的用例了，觉得越来越糊涂了，拿登陆框来说吧，我写了 7 个用例，但总感觉不好，在网上找了篇文章，分享下，希望对大家有帮助。

快捷键的使用是否正常：

1. TAB 键的使用是否正确

2.上下左右键是否正确

3.界面如果支持 ESC 键 看是否正常的工作

3.ENTER 键的使用是否正确切换时是否正常。

布局美感

界面的布局是否符合人的审美的标准

具体因人而依

输入框的功能：

输入合法的用户名和密码可以成功进入

输入合法的用户名和不合法密码不可以进入，并给出合理的提示

输入不合法的用户名和正确密码不可以进入，并给出合理的提示

输入不合法的用户名和不正确的密码不可以进入，并给出合理的提示

不合法的用户名有：不正确的用户名，，使用了字符大于用户名的限制

正常用户名不允许的特殊字符 空的用户名，系统（操作系统和应用系统）的保留字符

不合法的密码有：空密码（除有特殊规定的），错误的密码，字符大于密码的限制

正常密码不允许的特殊字符，系统（操作系统和应用系统）的保留字符

界面的链接：

对于界面有链接的界面，要测试界面上的所有的链接都正常或者给出合理的提示

补充

输入框是否支持 复制和黏贴 和移动

密码框显示的不要是具体的字符，要是一些密码的字符

验证用户名前有空格是否可以进入，一般情况可以。

验证用户名是否区分大小写。（有的软件是区分大小写的）

验证必填项为空，是否允许进入。

验证登录的次数是否有限制。从安全角度考虑，有些安全级别高的软件会考虑这方面的限制。

2. 搜索功能测试用例设计

对被测试点进行分解，把测试用例分解为多个测试场景

场景编号	场景描述	预期结果
场景一	页面检查	正确
场景二	默认条件搜索	查询结果正确
场景三	修改可选条件搜索	查询结果正确
场景四	修改输入条件搜索	查询结果正确
场景五	修改区间条件搜索	查询结果正确
场景六	组合可选、输入条件搜索	查询结果正确
场景七	操作后检查搜索条件及查询结果	查询结果正确
场景八	错误、空记录搜索	查询结果为空

测试步骤描述

按照已经分解的测试场景顺序，逐个描述测试场景的测试步骤

测试场景一：

步骤编号	具体描述
1	进入搜索（高级搜索）页面
2	界面共性测试
3	退出

测试场景二：

步骤编号	具体描述
1	进入搜索（高级搜索）页面
2	点击“搜索”按钮，显示查询结果列表
3	检查查询结果列表，每页显示记录条数正确、文字折行显示正确、页面布局美观
4	检查查询结果列表，列标题项、列显示内容、排序方式符合需求定义
5	检查查询结果列表，符合默认查询条件结果集
6	点击查询结果列表链接、复选框、全选框响应正确
7	退出

测试场景三：

步骤编号	具体描述
1	进入搜索（高级搜索）页面
2	逐一选择各个查询条件可选项，如：“全部”、“类别 1”等，点击“搜索”，查询结果正确
3	组合各个查询条件可选项，如：价格＋产品，点击“搜索”，查询结果正确
4	退出

测试场景四：

步骤编号	具体描述
1	进入搜索（高级搜索）页面
2	逐一输入文本域条件，模糊查询值，点击“搜索”，查询结果正确
3	逐一输入文本域条件，完全匹配值，点击“搜索”，查询结果正确
4	逐一输入文本域条件，中文值，点击“搜索”，查询结果正确
5	逐一输入文本域条件，字母大、小写值，点击“搜索”，查询结果正确
6	逐一输入文本域条件，数字类型值，点击“搜索”，查询结果正确
7	逐一输入文本域条件，全角、半角值，点击“搜索”，查询结果正确
8	组合各个文本域查询条件，点击“搜索”，查询结果正确
9	退出

3. 翻页功能测试用例

翻页功能我们常碰到的一般有以下几个功能：

1、首页、上一页、下一页、尾页。

2、总页数，当前页数

3、指定跳转页

4、指定每页显示条数

当然，有一些是少于多少页，全部以数字的形式显示，多于多少页后，才出现下一页的控件。

本文暂且用以上四点来做为通用的用例来设计吧。

对于 1 翻页链接或按钮的测试，主要要检查的测试点有：

1、有无数据时控件的显示情况

2、在首页时，首页和上一页是否能点击

3、在尾页时，下一页和尾页是否能点击

4、在非首页和非尾页时，四个按钮功能是否正确

5、翻页后，列表中的记录是否仍按照指定的排序列进行了排序

对于 2 总页数，当前页数，主要要检查的测试点有：

1、总页数是否等于总的记录数/指定每页条数

2、当前页数是否正确

对于 3 指定跳转页，主要要检查的测试点有：

1、是否能正常跳转到指定的页数

2、输入的跳转页数非法时的处理

对于 4 指定每页显示条数，主要要检查的测试点有：

1、是否有默认的指定每页显示条数

2、指定每页的条数后，列表显示的记录数，页数是否正确

3、输入的每页条数非法时的处理

分析完上面的测试点，应该可以进行用例的设计了。

step 1: 列表无记录
expect: 1、四个翻页控件变灰不可点击
2、列表有相应的无数据信息提示
3、不可指定页数
4、不可指定跳转页
5、总页数显示为 0
6、当前页数显示为 0

step 2: 列表的记录数<=指定的每页显示条数
expect: 1、四个翻页控件变灰不可点击
2、总页数显示为 1
3、当前页数显示为 1

step 3: 列表的记录数>指定的每页显示条数
expect: 1、默认在首页，当前页数为 1
2、列表的数据按照指定的排序列正确排序
3、记录数与数据库相符
4、总页数＝记录数/指定的每页显示条数

step 4: 列表的记录数>指定的每页显示条数，在首页
expect: 1、首页变灰不可点击
2、上一页变灰不可点击
3、下一页可点击，从（每页指定条数＋1）条记录开始显示，当前页数+1
4、尾页可点击，显示最后页的记录

step 5: 列表的记录数>指定的每页显示条数，在中间的某页
expect: 1、首页可点击，显示 1 到每页指定条数的记录
2、上一页可点击，显示上一页的记录
3、下一页可点击，从后一页的记录
4、尾页可点击，显示最后页的记录
5、列表的数据按照指定的排序列正确排序
6、当前页数为所在页

step 6:列表的记录数>指定的每页显示条数，在尾页
expect: 1、首页可点击，显示 1 到每页指定条数的记录
2、上一页可点击，显示上一页的记录
3、下一页变灰不可点击
4、尾页变灰不可点击
5、列表的数据按照指定的排序列正确排序
6、当前页数为最后一页的页数

step 7:输入每页显示条数为正整数
expect: 1、每页显示条数更新成指定的条数
2、超过指定的条数的记录分页显示

3、总页数更新成列表的记录数/每页显示条数

step 8:输入每页显示条数为 0
expect: 1、提示“每页显示条数必须为大于 1 的整数”
2、提示后每页显示条数恢复为上次生效的条数

step 9:输入每页显示条数为负数
expect: 1、提示每页显示条数必须为大于 1 的整数
2、提示后每页显示条数恢复为上次生效的条数

step 10:输入每页显示条数长度超过数据库指定的长度<<<maxlen>>>
expect: 1、提示每页显示条数不能超过<<<maxlen>>>位
2、提示后每页显示条数恢复为上次生效的条数

step 11:输入每页显示条数为字符串，如中文翻页数
expect: 1、提示每页显示条数必须为大于 1 的整数
2、提示后每页显示条数恢复为上次生效的条数

step 12:输入每页显示条数为特殊字符，如%
expect: 1、提示每页显示条数必须为大于 1 的整数
2、提示后每页显示条数恢复为上次生效的条数

step 13:输入每页显示条数为 html 字符串，如
expect: 1、提示每页显示条数必须为大于 1 的整数
2、提示后每页显示条数恢复为上次生效的条数

step 14:输入跳转的页数为存在的页数
expect: 1、正确跳转到指定的页数

step 15:输入跳转的页数不存在或非法值
expect: 1、跳转的页数值置为 1，显示第一页的数据
以上的用例是将总页数，当前页数都揉进了翻页控件的测试用例中了

4. 输入框的测试

最近在测试Web的输入框的时候，老是不知道从何处下手，去网上搜罗了一些资料，当然网上对输入框的测试资料少之又少，所以我作了一个简单的总结，总的情况有一下几个方面：

1.验证输入与输出的是否信息一致；

2.输入框之前的标题是否正确；

3.对特殊字符的处理，尤其是输入信息徐需要发送到数据库的。特殊字符包括：'（单引号）、"（双引号）、[]（中括号）、()（小括号）、{}（大括号）、;（分号）、<>（大于小于号）……

4.对输入框输入超过限制的字符的处理，一般非特殊的没有作出限制的在 255byte 左右；

5.输入框本身的大小、长度；

6.不同内码的字符的输入；

7.对空格、TAB 字符的处理机制；

8.字符本身显示的颜色；

9.密码输入窗口转换成星号或其它符号；

10.密码输入框对其中的信息进行加密，防止采用破解星号的方法破解；

11.按下 ctrl 和 alt 键对输入框的影响；

12.对于新增、修改、注册时用的输入框，有限制的，应该输入时作出提示，指出不允许的或者标出允许的；

13.对于有约束条件要求的输入框应当在条件满足时输入框的状态发生相应的改变，比如选了湖南就应该列出湖南下面的市，或者选了某些条件之后，一些输入框会关闭或转为只读状态；

14.输入类型；根据前面的栏位标题判断该输入框应该输入哪些内容算是合理的。例如，是否允许输入数字或字母，不允许输入其他字符等。

15.输入长度；数据库字段有长度定义，当输入过长时，提交数据是否会出错。

16.输入状态；当处于某种状态下，输入框是否处于可写或非可写状态。例如，系统自动给予的编号等栏位作为唯一标识，当再次处于编辑状态下，输入框栏位应处于不可写状态，如果可写对其编辑的话，可能会造成数据重复引起冲突等。

暂时，就能想这么多，看大家谁还有观点，互相学习下！

17.如果是会进行数据库操作的输入框，还可以考虑输入 SQL 中的一些特殊符号如单引号等，有时会有意想不到的错误出现

18.输入类型　输入长度　是否允许复制粘贴　为空的情况　空格的考虑　半角全角测试　对于密码输入框要考虑显示的内容是*　输入错误时的提示信息及提示信息是否准确

19.可以先了解你要测试的输入框在软件系统的某个功能中所扮演的角色，然后了解其具体的输入条件，在将输入条件按照有效等价类，无效等价类，边界值等方法进行测试用例的设计。

20.关键字有大小写混合的情况；

21.关键字中含有一个或多个空格的情况，包括前空格，中间空格（多个关键字），和后空格；

22.关键字中是否支持通配符的情况（视功能而定）；

23.关键字的长度分别为 9、10、11 个字符时的情况；

24.关键字是 valid，但是没有匹配搜索结果的情况；

25.输入 html 的标签会出现哪些问题？输入&lt；html&gt； 会出现什么问题呢？（这条是我自己发现的，在网上也没找到类似的东东，呵呵，大家凑合着看吧）

安全测试方面：

给出一些特别的关键字，比如 or 1=1， 这样的关键字如果不被处理就直接用到数据库查询中去，后果可想而知。

5. Web 测试的常用的检查点

1，页面连接检查每一个连接是否都有对应的页面，并且页面之间切换正确。

2，相关性检查删除/增加一项是否会对其他项产生影响，如果产生影响，这些影响是否都正确。

3，检查按扭的功能是否正确如update，cancel，delete，save等功能是否正确。

4，字符串长度检查输入超过需求说明的字符串长度的内容，看系统是否检查字符串长度，会不会出错。

5，字符类型检查在应该输入指定类型的内容的地方输入其他类型的内容（如在应该输入整形的地方输入其他字符类型），看系统是否检查字符类型，是否报错。

6，标点符号检查输入内容包括各种标点符号，特别是空格，各种引号，回车健，看系统是否处理正确。

7，中文字符处理在可以输入中文的系统输入中文（简体或繁体），看是否会出现乱码或出错。

8，检查带出信息的完整性在查看信息和update信息时，查看所填写的信息是否全部带出，带出信息和添加的是否一致。

9，信息重复检查在一些需要命名，且名字应该唯一的信息输入重复的名字或id，看系统有没有处理，是否报错，重名包括是否区分大小写，以及在输入内容的前后输入空格，系统是否作出正确处理。

10，检查删除功能在一些可以一次删除多个信息的地方，不选择任何信息，按‘delete’，看系统如何处理，是否报错，然后选择一个或多个信息，进行删除，看是否做正确处理。

11，检查添加和修改的一致，检查添加和修改信息的要求是否一致，例如添加要求必添的项，修改也应该必填，添加规定的整形的项，修改也必须为整形。

12，检查修改重名，修改时把不能重名的项改为已存在的内容看是否会处理，同时，也要注意，会不会报和自己重名的错。

13，重复提交表单一条已经成功提交的记录，back后再提交，看系统会如何处理。

14，检查多次使用back健的情况在有back的地方，back，回到原来的页面，再back，重复几次，看是否会报错。

15，search检查在有search功能的地方输入系统存在和不存在的内容，看search结果是否正确，如果可以输入多个search条件，可以同时添加合理和不合理的条件，看系统处理是否正确。

16，输入信息位置注意在光标停留的地方输入信息时，光标和所输入信息是否会跳到别的地方。

17，上传 和下载文件检查上传和下载文件的功能是否实现，上传是否能打开。对上传文件的格式有什么规定，系统是否有解释信息，并检查系统是否能够做到。

18，必填项检查应该填写的项没有填写的时候系统是否都做了处理，对必填项是否提示

信息，如在必填项前面加*.

19，快捷键检查是否支持常用快捷，如 Ctrl+C，Ctrl+V，BackSpace 等，对一些不允许的输入信息的字段，如选人，选日期对快捷方式是否做了限制。

20，回车检查在输入结束后直接按回车键，看系统如何处理，是否会报错。

性能测试

2.1.连接速度测试用户连接到Web 应用系统的速度根据上网方式的变化而变化，他们或许是电话拨号，或是宽带上网。当下载一个程序时，用户可以等较长的时间，但如果仅仅访问一个页面就不会这样。如果Web 系统响应时间太长（例如超过 5 秒钟），用户就会因没有耐心等待而离开。

另外，有些页面有超时的限制，如果响应速度太慢，用户可能还没来得及浏览内容，就需要重新登陆了。而且，连接速度太慢，还可能引起数据丢失，使用户得不到真实的页面。

2.2.负载测试负载测试是为了测量Web 系统在某一负载级别上的性能，以保证Web 系统在需求范围内能正常工作。负载级别可以是某个时刻同时访问Web 系统的用户数量，也可以是在线数据处理的数量。例如：Web 应用系统能允许多少个用户同时在线？如果超过了这个数量，会出现什么现象？Web 应用系统能否处理大量用户对同一个页面的请求？

6. 用户及权限管理功能常规测试方法

1) 赋予一个人员相应的权限后，在界面上看此人员是否具有此权限，并以此人员身份登陆，验证权限设置是否正确（能否超出所给予的权限）；

2) 删除或修改已经登陆系统并正在进行操作的人员的权限，程序能否正确处理；

3) 重新注册系统变更登陆身份后再登录，看程序是否能正确执行，具有权限是否正确；

4) 在有工作组或角色管理的情况下，删除包含用户的工作组或角色，程序能否正确处理；

5) 不同权限用户登录同一个系统，权限范围是否正确；

6) 覆盖系统所有权限设定；

7) 能否添加信息为空的用户(其中包括空用户名及空口令、空用户名非空口令、非空用户名及空口令) ；

8) 能否添加长用户名及长口令，如果允许，新用户能否正确登录；

9) 系统是否允许删除系统管理员这一特殊用户或修改系统管理员口令，删除或修改后系统的实际情况；

10) 登录用户能否修改自己的权限；

11) 添加用户（有标识或编号）:标识相同，用户名不同；标识相同，用户名相同；标识不同，用户名相同；标识不同，用户名不同；

12) 登录用户能否修改本人（或其他人）的信息，删除本人（或其他人）；

13) 修改用户的信息（包括权限，口令，基本信息等），对其他模块的影响；

14) 修改用户信息：修改后的用户信息和已经存在的用户信息相同；修改后的用户信息和已经存在的用户信息不同；

15) 不给用户授权，是否允许登录；

15) 改某些设置时，是否会影响具有上级权限及相同权限人员的设置；

16) 系统管理员修改了某些数据，以其他人员身份登录时数据是否改变；

17) 用户能否同时属于多个组，各个组的权限能否交叉；删除后重新添加的用户是否具有以前的权限；更改用户各项属性（包括权限）看对权限是否有影响。

7. Web 测试之兼容性测试

WEB 测试之兼容性测试

发表于: 2009-8-30 17:20　　作者: 战王　　来源: 中博 IT 教育网

1. 软件兼容性测试

兼容性测试是指待测试项目在特定的硬件平台上，不同的应用软件之间，不同的操作系统平台上，在不同的网络等环境中能正常的运行的测试。

兼容性测试的目的：待测试项目在不同的操作系统平台上正常运行，包括待测试项目能在同一操作系统平台的不同版本上正常运行；待测试项目能与相关的其他软件或系统的“和平共处”；待测试项目能在指定的硬件环境中正常运行；待测试项目能在不同的网络环境中正常运行。

兼容性测试无法做到完全的质量保证，但对于一个项目来讲，兼容性测试是必不可少的一个步骤。

2. Web 兼容性测试的主要类型

Web 兼容性测试主要是针对不同的操作系统平台，浏览器，以及分辨率进行的测试。

2.1. 操作系统兼容性测试

常见的操作系统有 Windows，Unix，Linux 等，对于普通用户来讲，最常用的是 Windows 操作系统。Windows 操作系统包括 Windows XP，windows 2003，vista，Win2000/NT，Windows9x 等等。用户使用操作系统的类型，直接决定了我们操作系统平台兼容性测试的操作系统平台数量，进行操作系统平台的兼容性测试的主要目的就是保证我们的待测试项目在该操作系统平台下能正常运行。

对于一些特殊项目（比如定制项目），可以指定某一类型的操作系统版本，这些都应该在需求规格说明书中指明，针对这些指明的操作系统版本必须进行兼容性测试。

大部分的其他项目，是不指定操作系统版本的，针对这样的项目，我们应当针对当前的主流操作系统版本进行兼容性测试，在确保主流操作系统版本兼容性测试的前提下在对非主流操作系统版本进行测试，尽量保证项目的操作系统版本的兼容性测试的完整性。

2.2. 浏览器兼容性测试

浏览器是 Web 系统中对核心的组成构件，来自不同厂家的浏览器对 Javascrīpt、ActiveX 或不同的 HTML 规格有不同的支持，即使是同一厂家的浏览器，也存在不同的版本的问题。不同的浏览器对安全性和 JAVA 的设置也不一样。

目前最为常用的浏览器为：IE 6.0 IE 7.0.但由于操作习惯的问题，还有相当一部分用户喜欢使用腾讯的 TT，以及 firefox 浏览器，这些浏览器同样也存在各个版本的问题。这个对于 Web 系统来讲是一个相当大的挑战。

对于一些特殊项目（比如定制项目），可以指定某一类型的浏览器（包括版本），这些都必须在需求规格说明书中指明。针对这些指明的浏览器必须进行兼容性测试。但大部分的项目，是不能指定浏览器的，针对这样的项目，那么我们必须针对当前的主流浏览器（含版本），在确保主流浏览器的兼容性测试通过的前提下，再对非主流浏览器（含版本）进行测试，尽量保证项目的浏览器的兼容性测试的完整性。

2.3. 分辨率兼容性测试

分辨率的测试是为了页面版式在不同的分辨率模式下能正常显示，字体符合要求而进行的测试。

用户使用什么模式的分辨率，对于我们来讲是未知的。通常情况下，在我们的需求规格说明书中会建议某些分辨率。对于测试来讲，必须针对需求规格说明书中建议的分辨率进行专门的测试。现在常见的分辨率是 1024×768，800×600。对于需求规格说明书中规定的分辨率，测试必须保证测试通过，但对于其他分辨率，原则上也应该尽量保证，但由于这个在需求规格说明书中没有加以约束，所以在一定程度上，开发往往会拒绝进行调整。对于需求规格说明书中没有规定分辨率的项目，测试应该在完成主流分辨率的兼容性测试的前提下，尽可能进行一些非主流分辨率的兼容性测试，在一定程度上保证大部分。

8. Web 测试-sql 注入

Web 安全性测试—SQL 注入

因为要对网站安全性进行测试，所以，学习了一些 sql 注入的知识。

在网上看一些 sql 注入的东东，于是想到了对网站的输入框进行一些测试，本来是想在输入框中输入<script>alter("abc")<script>,但是输入框有字符限制，只好输入<script>,结果网站出大问题了，呵呵，终于又出现了个 bug。

另一个就是在 URL 最后随意输入一些字符或数字，结果，新闻模块那出现了问题，暴露了网站的一些信息，又一个 bug，高兴下……

下面把今天看到的有关 SQL 注入方面的知识整理如下：

SQL 注入是一种攻击方式，在这种攻击方式中，恶意代码被插入到字符串中，然后将该字符串传递到 SQL Server 的实例以进行分析和执行。任何构成 SQL 语句的过程都应进行注入漏洞检查，因为 SQL Server 将执行其接收到的所有语法有效的查询。一个有经验的、坚定的攻击者甚至可以操作参数化数据。

SQL 注入的主要形式包括直接将代码插入到与 SQL 命令串联在一起并使其得以执行的用户输入变量。一种间接的攻击会将恶意代码注入要在表中存储或作为元数据存储的字符串。在存储的字符串随后串连到一个动态 SQL 命令中时，将执行该恶意代码。

注入过程的工作方式是提前终止文本字符串，然后追加一个新的命令。由于插入的命令可能在执行前追加其他字符串，因此攻击者将用注释标记“--”来终止注入的字符串。执行时，此后的文本将被忽略。

以下脚本显示了一个简单的 SQL 注入。此脚本通过串联硬编码字符串和用户输入的字符串而生成一个 SQL 查询：

```
var Shipcity;
ShipCity = Request.form. ("ShipCity");
var sql = "select * from OrdersTable where ShipCity = '" + ShipCity + "'";
```

用户将被提示输入一个市县名称。如果用户输入 Redmond，则查询将由与下面内容相似的脚本组成：

```
SELECT * FROM OrdersTable WHERE ShipCity = 'Redmond'
```

但是，假定用户输入以下内容：

```
Redmond'; drop table OrdersTable--
```

此时，脚本将组成以下查询：

```
SELECT * FROM OrdersTable WHERE ShipCity = 'Redmond';drop table OrdersTable--'
```

分号(;)表示一个查询的结束和另一个查询的开始。双连字符(--)指示当前行余下的部分是一个注释，应该忽略。如果修改后的代码语法正确，则服务器将执行该代码。SQL Server 处理该语句时，SQL Server 将首先选择 OrdersTable 中的所有记录（其中 ShipCity 为 Redmond）。然后，SQL Server 将删除 OrdersTable。

只要注入的 SQL 代码语法正确，便无法采用编程方式来检测篡改。因此，必须验证所有用户输入，并仔细检查在您所用的服务器中执行构造 SQL 命令的代码。本主题中的以下各部分说明了编写代码的最佳做法。

验证所有输入:

始终通过测试类型、长度、格式和范围来验证用户输入。实现对恶意输入的预防时，请注意应用程序的体系结构和部署方案。请注意，设计为在安全环境中运行的程序可能会被复制到不安全的环境中。以下建议应被视为最佳做法：

如果一个用户在需要邮政编码的位置无意中或恶意地输入了一个 10 MB 的 MPEG 文件，应用程序会做出什么反应？

如果在文本字段中嵌入了一个 DROP TABLE 语句，应用程序会做出什么反应？

测试输入的大小和数据类型，强制执行适当的限制。这有助于防止有意造成的缓冲区溢出。

输入字符 在 Transact-SQL 中的含义

; 查询分隔符。

' 字符数据字符串分隔符。

-- 注释分隔符。

/* ... */ 注释分隔符。服务器不对/*和*/之间的注释进行处理。

xp_ 用于目录扩展存储过程的名称的开头，如 xp_cmdshell。

9. Web 测试中书写用例时要考虑的检查点

通常书写 Test Case 时需要考虑的检查点.

对于屏幕显示来说包括：

检查显示的布局；

检查域和按钮的顺序；

检查域的尺寸；

检查字体的大小和风格；

检查文本的含义；

检查拼写错误；

检查屏蔽域；

检查只读域；

检查图片；

检查按钮的状态；

检查按钮的尺寸；

检查按钮的图标和名字；

检查是否有重复的图标；

检查指针是否在第一个可输入域；

检查 TAB 键的次序；

对于域来说包括：

检查可编辑性；

检查域间的移动；

检查分界条件；
检查有效分界符；
检查无效分界符；
检查连续多个有效分界符；
检查仅一个分界符输入；
检查多余空格的截取；
检查只读域和屏蔽域在 TAB 时的状态；
对于数字域来说包括：
检查正数值；
检查负数值；
检查零值；
检查小数点；
检查特殊字符加数字；
检查字母加数字；
检查 ASCII 值；
检查重复值；
检查空值；
对于字符域来说包括：
检查仅有字母；
检查仅有数字；
检查字母数字；
检查允许的特殊字符；
检查禁止的特殊字符；
检查包含特殊字符的字母数字；
检查 ASCII 值；
对于字母域来说包括：
检查字母；
检查数字值；
检查字母数字值；
检查特殊字符；
检查 ASCII 值；
对于时间域来说包括：
检查字符?和/；
检查其他特殊字符；
检查字母数字值；
检查正确的格式；
检查错误的格式；
检查错误的日期数字；
检查正确的日期数字；
检查日历表；

10. 手机电子邮件测试用例

序号	测试项目	测试方法	测试标准

1	电子邮件设置	IP 地址设置	默认的为 010.000.000.172，自己设置必须设置为 010.000.000.172	如果需要手动设置必须设置成 010.000.000.172
		拨号设置(GSM)\	默认为 17266，手动设置必须设置成 17266	手动设置必须设置成 17266
		节点设置(GPRS)	必须设置为 cmwap	必须设置为 cmwap
		用户名设置	如果设置，必须设置为 WAP	如果设置必须设置为 WAP
		用户密码设置	如果用户名做了设置，此项必须设置为 wap	如果用户名做了设置，此项必须设置为 wap
		上网数据模式设置(GPRS/GSM)	选择 GPRS 或 GSM 方式	如果有这两项可选择，必须能够使用。
		端口类型选择	可以选择安全和不安全。	一般选择不安全
2	用户设置	帐户设置	将用户的邮箱地址、接收邮件服务器（POP3）、发送邮件服务器（SMTP）、邮箱帐户名以及密码设置完成	能够成功设置
		下载设置	设置好各个限制项目：如单个信件下载最大字节、全部信件下载最大字节等等	能够成功设置
3	编写新邮件		选择输入法编写文本，并选择插入的附件（格式为 txt/gif/jpg/bmp 等）	所有的输入法都能实现，插入附件功能必须实现
4	发送邮件		编辑好邮件后，输入对方的电子邮件，按确认键进行发送	邮件能够成功发送并要有保存提示或自动进入已发信箱（如果此功能已经设定）
5	删除邮件		在邮件列表中，选择某条邮件，按选项菜单，选择删除项，再按确认键。	能够删除所选邮件
6	回复邮件		在邮件列表中选择某条邮件，在选项中选择回复，然后进行编辑，确认后发送。	能够实现邮件的回复操作。
7	转发邮件		在收件箱中选择某条邮件，在选项中选择转发，然后进行编辑，并输入第三方的邮箱地址或者从地址簿中选择号码，按确认键进行发送。	能够实现邮件的转发。

11. 记事本与日历的测试用例

序号	测试项目		测试方法	判定标准
1	记事本测试	新建	在记事本中新建一个文本文档。	必须能够正常新建一个文档。
		编辑	对新建文档或者已有的文档进行编辑。	必须能够编辑文档，编辑时

				必须能够正确输入汉字、英文、数字、字母、标点符号等。
		保存	保存已经编辑的文档。	编辑完成后必须能够保存已编辑的内容。
		删除	删除已经保存的文档。	必须能够删除已经保存的文档，删除前必须有确认信息，删除成功与否必须有相应的提示。
2	日程表测试	新建	在记事本中新建一个日程安排。	必须能够正常新建一个日程。
		编辑	对新建文档或者已有的日程进行编辑。	必须能够编辑文档，编辑时必须能够正确输入汉字、英文、数字、字母、标点符号等。
		保存	保存已经编辑的日程。	编辑完成后必须能够保存已编辑的内容。
		删除	删除已经保存的日程。	必须能够删除已经保存的文档，删除前必须有确认信息，删除成功与否必须有相应的提示。
		日程提示设定	设定提示时间、提示模式。	必须能够设定日程提示的时间和提醒模式，设定完成后必须能够准时提醒。
		时间、日期设定	设定当前时间、日期。	必须能够设定当前的时间和日期，必须保证日程表里的万年历时间准确无误，至少保证日程表中每年的每个月（阳历与农历）的第一天与最后一天的正确性，另外对应的星期也必须准确。特别需要关注闰年。

12. Web 测试总结

13. 让 web 站点崩溃最常见的七大原因

磁盘已满

导致系统无法正常运行的最可能的原因是磁盘已满。一个好的网络管理员会密切关注磁盘的使用情况，隔一定的时间，就需要将磁盘上的一些负载转存到备份存储介质中（例如磁带 。

日志文件会很快用光所有的磁盘空间。Web服务器的日志文件、SQL*Net的日志文件、JDBC日志文件，以及应用程序服务器日志文件均与内存泄漏有同等的危害。可以采取措施将日志文件保存在与操作系统不同的文件系统中。日志文件系统空间已满时Web服务器也会被挂起，但机器自身被挂起的几率已大大减低。

C 指针错误

用C或C++编写的程序，如Web服务器API模块，有可能导致系统的崩溃，因为只要间接引用指针（即，访问指向的内存）中出现一个错误，就会导致操作系统终止所有程序。另外，使用了糟糕的C指针的Java模拟量（analog）将访问一个空的对象引用。Java中的空引用通常不会导致立刻退出JVM，但是前提是程序员能够使用异常处理方法恰当地处理错误。在这方面，Java无需过多的关注，但使用 Java对可靠性进行额外的度量则会对性能产生一些负面影响。

内存泄漏

C/C++程序还可能产生另一个指针问题：丢失对已分配内存的引用。当内存是在子程序中被分配时，通常会出现这种问题，其结果是程序从子程序中返回时不会释放内存。如此一来，对已分配的内存的引用就会丢失，只要操作系统还在运行中，则进程就会一直使用该内存。这样的结果是，曾占用更多的内存的程序会降低系统性能，直到机器完全停止工作，才会完全清空内存。

解决方案之一是使用代码分析工具（如 Purify）对代码进行仔细分析，以找出可能出现的泄漏问题。但这种方法无法找到由其他原因引起的库中的泄漏，因为库的源代码是不可用的。另一种方法是每隔一段时间，就清除并重启进程。Apache 的 Web 服务器就会因这个原因创建和清除子进程。

虽然 Java 本身并无指针，但总的说来，与 C 程序相比， Java 程序使用内存的情况更加糟糕。在 Java 中，对象被频繁创建，而直到所有到对象的引用都消失时，垃圾回收程序才会释放内存。即使运行了垃圾回收程序，也只会将内存还给虚拟机 VM，而不是还给操作系统。结果是：Java 程序会用光给它们的所有堆，从不释放。由于要保存实时（Just In Time，JIT）编译器产生的代码，Java 程序的大小有时可能会膨胀为最大堆的数倍之巨。

还有一个问题，情况与此类似。从连接池分配一个数据库连接，而无法将已分配的连接还回给连接池。一些连接池有活动计时器，在维持一段时间的静止状态之后，计时器会释放掉数据库连接，但这不足以缓解糟糕的代码快速泄漏数据库连接所造成的资源浪费。

进程缺乏文件描述符

如果已为一台 Web 服务器或其他关键进程分配了文件描述符，但它却需要更多的文件描述符，则服务器或进程会被挂起或报错，直至得到了所需的文件描述符为止。文件描述符用来保持对开放文件和开放套接字的跟踪记录，开放文件和开放套接字是 Web 服务器很关键的组成部分，其任务是将文件复制到网络连接。默认时，大多数 shell 有 64 个文件描述符，这意味着每个从 shell 启动的进程可以同时打开 64 个文件和网络连接。大多数 shell 都有一个内嵌的 ulimit 命令可以增加文件描述符的数目。

线程死锁

由多线程带来的性能改善是以可靠性为代价的，主要是因为这样有可能产生线程死锁。线程死锁时，第一个线程等待第二个线程释放资源，而同时第二个线程又在等待第一个线程释放资源。我们来想像这样一种情形：在人行道上两个人迎面相遇，为了给对方让道，两人同时向一侧迈出一步，双方无法通过，又同时向另一侧迈出一步，这样还是无法通过。双方都以同样的迈步方式堵住了对方的去路。假设这种情况一直持续下去，这样就不难理解为何会发生死锁现象了。

解决死锁没有简单的方法，这是因为使线程产生这种问题是很具体的情况，而且往往有很高的负载。大多数软件测试产生不了足够多的负载，所以不可能暴露所有的线程错误。在每一种使用线程的语言中都存在线程死锁问题。由于使用Java进行线程编程比使用C容易，所以 Java程序员中使用线程的人数更多，线程死锁也就越来越普遍了。可以在Java代码中增加同步关键字的使用，这样可以减少死锁，但这样做也会影响性能。如果负载过重，数据库内部也有可能发生死锁。

如果程序使用了永久锁，比如锁文件，而且程序结束时没有解除锁状态，则其他进程可能无法使用这种类型的锁，既不能上锁，也不能解除锁。这会进一步导致系统不能正常工作。这时必须手动地解锁。

服务器超载

Netscape Web 服务器的每个连接都使用一个线程。Netscape Enterprise Web 服务器会在线程用完后挂起，而不为已存在的连接提供任何服务。如果有一种负载分布机制可以检测到服务器没有响应，则该服务器上的负载就可以分布到其它的 Web 服务器上，这可能会致使这些服务器一个接一个地用光所有的线程。这样一来，整个服务器组都会被挂起。操作系统级别可能还在不断地接收新的连接，而应用程序（Web 服务器）却无法为这些连接提供服务。用户可以在浏览器状态行上看到 connected（已连接）的提示消息，但这以后什么也不会发生。

解决问题的一种方法是将 obj.conf 参数 RqThrottle 的值设置为线程数目之下的某个数值，这样如果越过 RqThrottle 的值，就不会接收新的连接。那些不能连接的服务器将会停止工作，而连接上的服务器的响应速度则会变慢，但至少已连接的服务器不会被挂起。这时，文件描述符至少应当被设置为与线程的数目相同的数值，否则，文件描述符将成为一个瓶颈。

数据库中的临时表不够用

许多数据库的临时表（cursor）数目都是固定的，临时表即保留查询结果的内存区域。在临时表中的数据都被读取后，临时表便会被释放，但大量同时进行的查询可能耗尽数目固定的所有临时表。这时，其他的查询就需要列队等候，直到有临时表被释放时才能再继续运行。

这是一个不容易被程序员发觉的问题，但会在负载测试时显露出来。但可能对于数据库管理员（DataBase Administrator，DBA）来说，这个问题十分明显。

此外，还存在一些其他问题：设置的表空间不够用、序号限制太低，这些都会导致表溢出错误。这些问题表明了一个好的 DBA 对用于生产的数据库设置和性能进行定期检查的重要性。而且，大多数数据库厂商也提供了监控和建模工具以帮助解决这些问题。

另外，还有许多因素也极有可能导致 Web 站点无法工作。如：相关性、子网流量超载、糟糕的设备驱动程序、硬件故障、包括错误文件的通配符、无意间锁住了关键的表。

14. Web 应用程序是否存在跨站点脚本漏洞

到目前为止，对于跨站点脚本攻击具有很大的威胁这一点大家并无异议。如果您很精通 XSS 并且只想看看有什么好的测试方法可供借鉴，那么请直接跳到本文的测试部分。如果您对此一无所知，请按顺序认真阅读！如果某个怀有恶意的人（攻击者）可以强迫某个不知情的用户（受害者）运行攻击者选择的客户端脚本，那么便会发生跨站点脚本攻击。“跨站点脚本”这个词应该属于用词不当的情况，因为它不仅与脚本有关，而且它甚至不一定是跨站点的。所以，它就是一个在发现这种攻击时起的一个名字，并且一直沿用至今。从现在开始，我们将使用它常见的缩写名称“XSS”。

XSS 攻击的过程涉及以下三方：

- 攻击者
- 受害者
- 存在漏洞的网站（攻击者可以使用它对受害者采取行动）

在这三方之中，只有受害者会实际运行攻击者的代码。网站仅仅是发起攻击的一个载体，一般不会受到影响。可以用多种方式发起 XSS 攻击。例如，攻击者可通过电子邮件、IM 或其他途径向受害者发送一个经过经心构造的恶意 URL。当受害者在 Web 浏览器中打开该 URL 的时侯，网站会显示一个页面并在受害者的计算机上执行脚本。

XSS 漏洞是什么样的呢？

作为一名 Web 开发人员或测试人员，您肯定知道 Web 应用程序的技术基础是由 HTTP 和 HTML 组成的。HTTP 协议是 HTML 的传输机制，可使用代码设计 Web 页面布局和生成页面。

如果 Web 应用程序接受用户通过 HTTP 请求（如 GET 或 POST）提交的输入信息，然后使用输出 HTML 代码在某些地方显示这些信息，便可能存在 XSS 漏洞。下面是一个最简单的例子：

1. Web 请求如下所示：

GET http://www.somesite.com/page.asp?pageid=10&lang=en&title=Section%20Title

2. 在发出请求后，服务器返回的 HTML 内容包括：

<h1>Section Title</h1>

可以看到，传递给“title”查询字符串参数的用户输入可能被保存在一个字符串变量中并且由 Web 应用程序插入到 <h1> 标记中。通过提供输入内容，攻击者可以控制 HTML。

3. 现在，如果站点没有在服务器端对用户输入加以过滤（因为总是可以绕过客户端控件），那么恶意用户便可以使用许多手段对此漏洞加以滥用：

攻击者可以通过摆脱 <h1> 标记来注入代码：

http://www.somesite.com/page.asp?pageid=10&lang=en&title=Section%20Title</h1><script>alert(‘XSS%20attack’)</script>

这个请求的 HTML 输出将为：

<h1>Section Title</h1><script>alert(‘XSS attack’)</script>

即便是这个最简单的例子，攻击者也可以利用此连接完成数不清的事情。让我们看看会有哪些潜在的威胁，然后讨论一些更高级的测试方法。

XSS 攻击的威胁有多么严重？

由于能够在生成的 Web 页面中注入代码，能想到的威胁有多么严重，就可以有多么严重的威胁。攻击者可以使用 XSS 漏洞窃取 Cookie，劫持帐户，执行 ActiveX，执行 Flash 内容，强迫您下载软件，或者是对硬盘和数据采取操作。

只要您点击了某些 URL，这一切便有可能发生。每天之中，在阅读来自留言板或新闻组的受信任的电子邮件的时侯，您会多少次地单击其中的 URL？

网络钓鱼攻击通常利用 XSS 漏洞来装扮成合法站点。可以看到很多这样的情况，比如您的银行给你发来了一封电子邮件，向您告知对您的帐户进行了一些修改并诱使您点击某些超链接。如果仔细观察这些 URL，它们实际上可能利用了银行网站中存在的漏洞，它们的形式类似于 http://mybank.com/somepage?redirect=<script>alert(‘XSS’)</script>，这里利用了“redirect”参数来执行攻击。

如果您足够狡猾的话，可以将管理员定为攻击目标，您可以发送一封具有如下主题的邮件：“求救！这个网站地址总是出现错误！”在管理员打开该 URL 后，便可以执行许多恶意操作，例如窃取他（或她）的凭证。

好了，现在我们已经理解了它的危害性 -- 危害用户，危害管理员，给公司带来坏的公共形象。现在，让我们看看本文的重点 -- 测试您的网站是否存在这些问题。

测试 XSS 漏洞

多年以来，我一直是一名全职的安全顾问，已经做过无数次的这种测试了。我将好的测试计划归结为两个字：彻底。对于你我来说，查找这些漏洞与能够有机会在 Bugtraq 或 Vulnwatch 上吹嘘一番没有任何关系；它只与如何出色完成负责的工作有关。如果这意味着对应用程序中所有的单个查询字符串参数、cookie 值 以及 POST 数据值进行检查，那么这只能表明我们的工作还不算太艰巨。

显然，一次完整的安全性检查所涉及的内容通常远远超出寻找 XSS 漏洞那样简单；它需要建立整体的威胁模型，测试溢出漏洞、信息泄漏、错误处理、SQL 注入、身份验证和授权错误。好在执行这样彻底的工作时，各个领域之间都存在重叠。比如，在测试 XSS 漏洞时，经常会同时找出错误处理或信息泄漏问题。

我假设您属于某个负责对 Web 应用程序进行开发和测试的小组。在这个幸运的位置上，您可以混合使用黑盒和白盒方法。每种方法都有它自己的优点，结合使用时甚至能相互提供支持。

1. 按顺序准备您的工具包

测试工作也可以是自动化的，但是我们在这里只讨论手动操作。手动测试的必备工具包括：

• Paros proxy (http://www.parosproxy.org)，用于截获 HTTP 通信数据

• Fiddler (http://www.fiddlertool.com/fiddler)，用于截获 HTTP 通信数据

• Burp proxy (http://www.portswigger.net/proxy/)

• TamperIE (http://www.bayden.com/dl/TamperIESetup.exe)，用于修改 GET 和 POST

我们以上至少列出了三种 Web 代理软件。也可以寻找其他不同的类似产品，因为每种产品都有它自己的独到之处。下面，您需要在 Web 浏览器发出 HTTP 请求之前截获这些请求，并修改它们以注入 XSS 测试代码。上面所有这些工具都可以完成这项任务，某些工具还会显示返回的 HTML 源代码（如果您选择了截获服务器响应）。

截获客户端发出的 GET 和 POST 请求非常重要。这样可以绕过所有的客户端 javascript. 输入验证代码。我在这里要提醒所有 Web 开发人员 -- 客户端安全控制是靠不住的。应该总是在服务器端执行有效性验证。

2. 确定站点及其功能 -- 与开发人员和 PM 交流

绘制一些简单的数据流图表，对站点上的页面及其功能进行描述。此时，可以安排一些与开发人员和项目经理的会议来建立威胁模型。在会议上尽可能对应用程序进行深入探讨。站点公开了 Web 服务吗？是否有身份验证表单？有留言板吗？有用户设置页面吗？确保列出了所有这些页面。

3. 找出并列出所有由用户提供输入的点

对站点地图进行进一步细化。我通常会为此创建一个电子表格。对于每个页面，列出所有查询字符串参数、cookie 值、自定义 HTTP 标头、POST 数据值和以其他形式传递的用户输入。不要忘记搜索 Web 服务和类似的 SOAP 请求，并找出所有允许用户输入的字段。

分别列出每个输入参数，因为下面需要独立测试每个参数。这可能是最重要的一个步骤！如果阅读下面的电子表格，您会看到我已经在示例站点中找出了一大堆这样的东西。如

forwardURL 和 lang 这样的查询字符串。如 name、password、msgBody、msgTitle 和这样的 POST 数据，甚至某些 Cookie 值。所有这些都是我们感兴趣的重要测试内容。

4. 认真思考并列出测试用例

使用已经得到的电子表格并列出各种用来测试 XSS 漏洞的方法。我们稍候将讨论各种方法，但是现在先让我们看看我的电子表格的屏幕截图，请注意，我列出了页面上允许的每个值以及每个值的所有测试类型。这种记录测试的方法仅是我自己的习惯，您可以使用自己的方法。我喜欢记录所有东西，以便我能知道已经做了哪些工作和哪些工作没有做。

5. 开始测试并注意输出结果

在查找漏洞的过程中，最重要的部分并不是您是否找到了漏洞。而是您是否真正知道究竟发生了哪些事情。对于 XSS，只需检查 HTML 输出并看看您输入的内容在什么地方。它在一个 HREF 标记中吗？是否在 IFRAME. 标记中？它在 CLSID 标记中吗？在 IMG SRC 中吗？某些 Flash 内容的 PARAM NAME 是怎样的？

我会检查所有这些情况，如果您对所输入内容的目的十分了解，可以调整您的测试来找出问题。这意味着您可能需要添加一个额外的封闭括号“>”来让某个标记变得完整，或者添加一个双引号来关闭标记内的一个元素。或者，您可能需要使用 URL 或 HTML 来编码您的字符，例如将双引号变为 %22 或 "。

嗯，并不那么容易，这个站点看来防范比较严密。现在该怎么办呢？

那么，也许您的简单测试用例 <script>alert(‘hi’)</script> 并不能产生期望中的警告对话框。仔细想想这个问题并在可能的情况下与开发人员进行交流。也许他们对输入中的尖括号、单引号或圆括号进行了过滤。也许他们会过滤“script”这个词。重新研究为何输入会产生这样的输出，并理解每个值（查询字符串、cookie、POST 数据）的作用。“pageId=10”这样的查询字符串值可能对输出没有影响，因此不值得花费时间测试它。有时，最好试着注入单个字符（例如尖括号、双引号标记或者圆括号），看看应用程序是否过滤这些字符。然后，便可以知道您面对的过滤级别究竟如何。接着，可以调整测试方法，对这些字符进行编码并重试，或者寻找其他注入办法。

改变测试用例

我恐怕无法充分对此进行说明：研究输入的值会输出什么样的 HTML 页面非常重要。如果它们不能产生输出，那么不要在它们上面浪费时间。如果可以，请进行研究，因为您需要根据输出对测试进行相应的修改。我使用了各种变化形式来找出能接受和显示脚本代码的参数。因为这涉及太多内容，因此在这里无法一一进行讨论，但是请务必注意以下几种情况：

1. >'"><script>alert(‘XSS')</script>

2. >%22%27><img%20src%3d%22javascript.:alert(%27XSS%27)%22>

3. >'"><img%20src%3D%26%23x6a;%26%23x61;%26%23x76;%26%23x61;%26%23x73;%26%23x63;%26%23x72;%26%23x69;%26%23x70;%26%23x74;%26%23x3a;alert(%26quot;XSS%26quot;)>

4. AK%22%20style%3D%22background:url(javascript.:alert(%27XSS%27))%22%20OS%22

5. %22%2Balert(%27XSS%27)%2B%22

6. <table background="javascript.:alert(([code])"></table>

7. <object type=text/html data="javascript.:alert(([code]);"></object>

8. <body nload="javascript.:alert(([code])"></body>

有许多变化形式可以尝试。关键在于了解程序究竟使用何种方式处理输入和显示输出页面。如同这些例子所展示的，常见的变化形式经常是在脚本代码前面加上 “>””，以尝试封闭网站可能在输出中生成的标记。还可以对代码进行 URL 编码，尝试绕过服务器端的输入过滤功能。此外，因为尖括号“<>”通常会在输入时被过滤和从输出中删除，所以还必须尝试不需要尖括号的 XSS，例如 ”&{alert('XSS')};”

持久和动态

找出一个成功的 XSS 颇费周折，因为在开始时 XSS 攻击可能并不是那么显而易见的。随便举一个例子，如果向网站添加一条新留言并在“msgTitle”值中注入代码，在提交数据后，您可能不会立即看到脚本代码被执行。但是，当您访问留言板的时侯，将会在 HTML 页面中使用“msgTitle”值并将其作为脚本代码执行。这种情况被称作持久性 XSS 攻击，如果包含脚本代码的值将被保存到客户端或者后端系统并在稍候的时间被执行，便会发生此种攻击。

而与此相对的是动态 XSS 攻击，这种攻击会立即执行并只发生一次。比如，如果某个链接或 GET 请求在某个用来控制页面输出的查询字符串中包含了脚本代码，那么在点击链接后会立即显示输出。

总结

XSS 测试通常只是整个 Web 应用程序安全性审查工作的一小部分，但是在执行测试时必须细致和彻底。在多年的工作中，我一直强调使用电子表格或其他方式来记录站点的所有页面，以及每个页面接受的输入值（查询字符串、cookie、POST 数据、SOAP），这是在测试前必须进行的一个重要步骤。与此同等重要的是，理解输入以及它在输出的 HTML 页面上的呈现方式。如果您知道了应用程序处理输入的方式，就会非常迅速地完成许多工作。不要浪费时间测试那些不会作为输出显示的输入。与开发人员和 PM 进行交流，并在开始测试前建立完善的威胁模型。

15. Web 测试总结（全）

在Web工程过程中，基于Web系统的测试、确认和验收是一项重要而富有挑战性的工作。基于Web的系统测试与传统的软件测试不同，它不但需要检查和验证是否按照设计的要求运行，而且还要测试系统在不同用户的浏览器端的显示是否合适。重要的是，还要从最终用户的角度进行安全性和可用性测试。然而，Internet和Web媒体的不可预见性使测试基于Web的系统变得困难。因此，我们必须为测试和评估复杂的基于Web的系统研究新的方法和技术。

本文将 web 测试分为 6 个部分：

1. 功能测试
2. 性能测试（包括负载/压力测试）
3. 用户界面测试
4. 兼容性测试
5. 安全测试

6. 接口测试

本文的目的是覆盖 web 测试的各个方面，未就某一主题进行深入说明。

1 功能测试

1.1 链接测试

链接是 Web 应用系统的一个主要特征，它是在页面之间切换和指导用户去一些不知道地址的页面的主要手段。链接测试可分为三个方面。首先，测试所有链接是否按指示的那样确实链接到了该链接的页面；其次，测试所链接的页面是否存在；最后，保证 Web 应用系统上没有孤立的页面，所谓孤立页面是指没有链接指向该页面，只有知道正确的 URL 地址才能访问。

链接测试可以自动进行，现在已经有许多工具可以采用。链接测试必须在集成测试阶段完成，也就是说，在整个 Web 应用系统的所有页面开发完成之后进行链接测试。

采取措施：采用自动检测网站链接的软件来进行。

推荐软件：

Xenu Link Sleuth 免费 绿色免安装软件

HTML Link Validator 共享（30 天试用）

1.2 表单测试

当用户通过表单提交信息的时候，都希望表单能正常工作。

如果使用表单来进行在线注册，要确保提交按钮能正常工作，当注册完成后应返回注册成功的消息。如果使用表单收集配送信息，应确保程序能够正确处理这些数据，最后能让顾客能让客户收到包裹。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。

当用户使用表单进行用户注册、登陆、信息提交等操作时，我们必须测试提交操作的完整性，以校验提交给服务器的信息的正确性。例如：用户填写的出生日期与职业是否恰当，填写的所属省份与所在城市是否匹配等。如果使用了默认值，还要检验默认值的正确性。如果表单只能接受指定的某些值，则也要进行测试。例如：只能接受某些字符，测试时可以跳过这些字符，看系统是否会报错。

1.3 数据校验

如果系根据业务规则需要对用户输入进行校验，需要保证这些校验功能正常工作。例如，省份的字段可以用一个有效列表进行校验。在这种情况下，需要验证列表完整而且程序正确调用了该列表(例如在列表中添加一个测试值，确定系统能够接受这个测试值)。

在测试表单时，该项测试和表单测试可能会有一些重复。

1.2 和 1.3 的采取措施：第一个完整的版本采用手动检查，同时形成 WinRunner（QTP）脚本；回归测试以及升级版本主要靠 WinRunner（QTP）自动回放测试。

1.4 cookies 测试

Cookies通常用来存储用户信息和用户在某应用系统的操作，当一个用户使用Cookies访问了某一个应用系统时，Web服务器将发送关于用户的信息，把该信息以Cookies的形式存储在客户端计算机上，这可用来创建动态和自定义页面或者存储登陆等信息。

如果 Web 应用系统使用了 Cookies，就必须检查 Cookies 是否能正常工作。测试的内容可包括 Cookies 是否起作用，是否按预定的时间进行保存，刷新对 Cookies 有什么影响等。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。

采取措施：

1 采用黑盒测试：采用上面提到的方法进行测试

2 采用查看 cookies 的软件进行（初步的想法）

可以选择采用的软件

IECookiesView v1.50

Cookies Manager v1.1

1.5 数据库测试

在 Web 应用技术中，数据库起着重要的作用，数据库为 Web 应用系统的管理、运行、查询和实现用户对数据存储的请求等提供空间。在 Web 应用中，最常用的数据库类型是关系型数据库，可以使用 SQL 对信息进行处理。

在使用了数据库的 Web 应用系统中，一般情况下，可能发生两种错误，分别是数据一致性错误和输出错误。数据一致性错误主要是由于用户提交的表单信息不正确而造成的，而输出错误主要是由于网络速度或程序设计问题等引起的，针对这两种情况，可分别进行测试。

采取措施：暂时没有更好的测试方法

考虑结合到 1.2 和 1.3 的测试中

1.6 应用程序特定的功能需求

最重要的是，测试人员需要对应用程序特定的功能需求进行验证。尝试用户可能进行的所有操作：下订单、更改订单、取消订单、核对订单状态、在货物发送之前更改送货信息、在线支付等等。这是用户之所以使用网站的原因，一定要确认网站能像广告宣传的那样神奇。

采取措施：深刻理解需求说明文档

1.7 设计语言测试

Web设计语言版本的差异可以引起客户端或服务器端严重的问题，例如使用哪种版本的HTML等。当在分布式环境中开发时，开发人员都不在一起，这个问题就显得尤为重要。除了HTML的版本问题外，不同的脚本语言，例如Java、JavaScript、 ActiveX、VBScript或Perl等也要进行验证。

暂时没有方法测试，可以多参考一点讨论组内的更新信息

2 性能测试

2.1 连接速度测试

用户连接到Web应用系统的速度根据上网方式的变化而变化，他们或许是电话拨号，或是宽带上网。当下载一个程序时，用户可以等较长的时间，但如果仅仅访问一个页面就不会这样。如果Web系统响应时间太长（例如超过 5 秒钟），用户就会因没有耐心等待而离开。

另外，有些页面有超时的限制，如果响应速度太慢，用户可能还没来得及浏览内容，就需要重新登陆了。而且，连接速度太慢，还可能引起数据丢失，使用户得不到真实的页面。

2.2 负载测试

负载测试是为了测量 Web 系统在某一负载级别上的性能，以保证 Web 系统在需求范围内能正常工作。负载级别可以是某个时刻同时访问 Web 系统的用户数量，也可以是在线数据处理的数量。例如：Web 应用系统能允许多少个用户同时在线？如果超过了这个数量，会出现什么现象？Web 应用系统能否处理大量用户对同一个页面的请求？

2.3 压力测试

负载测试应该安排在 Web 系统发布以后，在实际的网络环境中进行测试。因为一个企业内部员工，特别是项目组人员总是有限的，而一个 Web 系统能同时处理的请求数量将远远超出这个限度，所以，只有放在 Internet 上，接受负载测试，其结果才是正确可信的。

进行压力测试是指实际破坏一个 Web 应用系统，测试系统的反映。压力测试是测试系统的限制和故障恢复能力，也就是测试 Web 应用系统会不会崩溃，在什么情况下会崩溃。黑客常常提供错误的数据负载，直到 Web 应用系统崩溃，接着当系统重新启动时获得存取权。

压力测试的区域包括表单、登陆和其他信息传输页面等。

负载/压力测试应该关注什么

测试需要验证系统能否在同一时间响应大量的用户，在用户传送大量数据的时候能否响应，系统能否长时间运行。可访问性对用户来说是极其重要的。如果用户得到“系统忙”的信息，他们可能放弃，并转向竞争对手。系统检测不仅要使用户能够正常访问站点，在很多情况下，可能会有黑客试图通过发送大量数据包来攻击服务器。出于安全的原因，测试人员应该知道当系统过载时，需要采取哪些措施，而不是简单地提升系统性能。

瞬间访问高峰

如果您的站点用于公布彩票的抽奖结果，最好使系统在中奖号码公布后的一段时间内能够响应上百万的请求。负载测试工具能够模拟 X 个用户同时访问测试站点。

每个用户传送大量数据

网上书店的多数用户可能只订购 1-5 书，但是大学书店可能会订购 5000 本有关心理学介绍的课本? 或者一个祖母为她的 50 个儿孙购买圣诞礼物(当然每个孩子都有自己的邮件地址) 系统能处理单个用户的大量数据吗?

长时间的使用

如果站点用于处理鲜花订单，那么至少希望它在母亲节前的一周内能持续运行。如果站点提供基于 web 的 email 服务，那么点最好能持续运行几个月，甚至几年。可能需要使用自动测试工具来完成这种类型的测试，因为很难通过手工完成这些测试。你可以想象组织 100 个人同时点击某个站点。但是同时组织 100000 个人呢。通常，测试工具在第二次使用的时候，它创造的效益，就足以支付成本。而且，测试工具安装完成之后，再次使用的时候，只要点击几下。

采取措施：采用测试工具 WAS、ACT 协助进行测试

3 用户界面测试

3.1 导航测试

导航描述了用户在一个页面内操作的方式，在不同的用户接口控制之间，例如按钮、对话框、列表和窗口等；或在不同的连接页面之间。通过考虑下列问题，可以决定一个 Web 应用系统是否易于导航：导航是否直观？Web 系统的主要部分是否可通过主页存取？Web 系统是否需要站点地图、搜索引擎或其他的导航帮助？

在一个页面上放太多的信息往往起到与预期相反的效果。Web 应用系统的用户趋向于目的驱动，很快地扫描一个 Web 应用系统，看是否有满足自己需要的信息，如果没有，就会很快地离开。很少有用户愿意花时间去熟悉 Web 应用系统的结构，因此，Web 应用系统导航帮助要尽可能地准确。

导航的另一个重要方面是 Web 应用系统的页面结构、导航、菜单、连接的风格是否一致。确保用户凭直觉就知道 Web 应用系统里面是否还有内容，内容在什么地方。

Web 应用系统的层次一旦决定，就要着手测试用户导航功能，让最终用户参与这种测试，效果将更加明显。

3.2 图形测试

在 Web 应用系统中，适当的图片和动画既能起到广告宣传的作用，又能起到美化页面的功能。一个 Web 应用系统的图形可以包括图片、动画、边框、颜色、字体、背景、按钮等。图形测试的内容有：

（1）要确保图形有明确的用途，图片或动画不要胡乱地堆在一起，以免浪费传输时间。Web 应用系统的图片尺寸要尽量地小，并且要能清楚地说明某件事情，一般都链接到某个具体的页面。

（2）验证所有页面字体的风格是否一致。

（3）背景颜色应该与字体颜色和前景颜色相搭配。

（4）图片的大小和质量也是一个很重要的因素，一般采用 JPG 或 GIF 压缩，最好能使图片的大小减小到 30k 以下

（5）最后，需要验证的是文字回绕是否正确。如果说明文字指向右边的图片，应该确保该图片出现在右边。不要因为使用图片而使窗口和段落排列古怪或者出现孤行。

通常来说，使用少许或尽量不使用背景是个不错的选择。如果您想用背景，那么最好使用单色的，和导航条一起放在页面的左边。另外，图案和图片可能会转移用户的注意力。

3.3 内容测试

内容测试用来检验 Web 应用系统提供信息的正确性、准确性和相关性。

信息的正确性是指信息是可靠的还是误传的。例如，在商品价格列表中，错误的价格可能引起财政问题甚至导致法律纠纷；信息的准确性是指是否有语法或拼写错误。这种测试通常使用一些文字处理软件来进行，例如使用 Microsoft Word 的"拼音与语法检查"功能；信息的相关性是指是否在当前页面可以找到与当前浏览信息相关的信息列表或入口，也就是一般 Web 站点中的所谓"相关文章列表"。

对于开发人员来说，可能先有功能然后才对这个功能进行描述。大家坐在一起讨论一些新的功能，然后开始开发，在开发的时候，开发人员可能不注重文字表达，他们添加文字可能只是为了对齐页面。不幸的是，这样出来的产品可能产生严重的误解。因此测试人员和公关部门一起检查内容的文字表达是否恰当。否则，公司可能陷入麻烦之中，也可能引起法律方面的问题。测试人员应确保站点看起来更专业些。过分地使用粗体字、大字体和下划线可能会让用户感到不舒服。在进行用户可用性方面的测试时，最好先请图形设计专家对站点进行评估。你可能不希望看到一篇到处是黑体字的文章，所以相信您也希望自己的站点能更专业一些。最后，需要确定是否列出了相关站点的链接。很多站点希望用户将邮件发到一个特定的地址，或者从某个站点下载浏览器。但是如果用户无法点击这些地址，他们可能会觉得很迷惑。

3.4 表格测试

需要验证表格是否设置正确。用户是否需要向右滚动页面才能看见产品的价格？把价格放在左边，而把产品细节放在右边是否更有效? 每一栏的宽度是否足够宽，表格里的文字是否都有折行？是否有因为某一格的内容太多，而将整行的内容拉长?

3.5 整体界面测试

整体界面是指整个 Web 应用系统的页面结构设计，是给用户的一个整体感。例如：当用户浏览 Web 应用系统时是否感到舒适，是否凭直觉就知道要找的信息在什么地方？整个 Web 应用系统的设计风格是否一致？

对整体界面的测试过程，其实是一个对最终用户进行调查的过程。一般 Web 应用系统采取在主页上做一个调查问卷的形式，来得到最终用户的反馈信息。

对所有的用户界面测试来说，都需要有外部人员（与 Web 应用系统开发没有联系或联系很少的人员）的参与，最好是最终用户的参与。

采取措施：手动测试，参与人员最好有外部人员

4 兼容性测试

4.1 平台测试

市场上有很多不同的操作系统类型，最常见的有Windows、Unix、Macintosh、Linux等。Web应用系统的最终用户究竟使用哪一种操作系统，取决于用户系统的配置。这样，就可能会发生兼容性问题，同一个应用可能在某些操作系统下能正常运行，但在另外的操作系统下可能会运行失败。

因此，在 Web 系统发布之前，需要在各种操作系统下对 Web 系统进行兼容性测试。

4.2 浏览器测试

浏览器是 Web 客户端最核心的构件，来自不同厂商的浏览器对 Java，、JavaScript、ActiveX、 plug-ins 或不同的 HTML 规格有不同的支持。例如，ActiveX 是 Microsoft 的产品，是为 Internet Explorer 而设计的，JavaScript 是 Netscape 的产品，Java 是 Sun 的产品等等。另外，框架和层次结构风格在不同的浏览器中也有不同的显示，甚至根本不显示。不同的浏览器对安全性和 Java 的设置也不一样。

测试浏览器兼容性的一个方法是创建一个兼容性矩阵。在这个矩阵中，测试不同厂商、不同版本的浏览器对某些构件和设置的适应性。

4.3 分辨率测试

页面版式在 640x400、600x800 或 1024x768 的分辨率模式下是否显示正常？字体是否太小以至于无法浏览？或者是太大？文本和图片是否对齐？

4.4 Modem/连接速率

是否有这种情况，用户使用 28.8 modem下载一个页面需要 10 分钟，但测试人员在测试的时候使用的是 T1 专线？用户在下载文章或演示的时候，可能会等待比较长的时间，但却不会耐心等待首页的出现。最后，需要确认图片不会太大。

4.5 打印机

用户可能会将网页打印下来。因此网也在设计的时候要考虑到打印问题，注意节约纸张和油墨。有不少用户喜欢阅读而不是盯着屏幕，因此需要验证网页打印是否正常。有时在屏幕上显示的图片和文本的对齐方式可能与打印出来的东西不一样。测试人员至少需要验证订单确认页面打印是正常的。

4.6 组合测试

最后需要进行组合测试。600x800 的分辨率在 MAC 机上可能不错，但是在 IBM 兼容机上却很难看。在 IBM 机器上使用 Netscape 能正常显示，但却无法使用 Lynx 来浏览。如果是内部使用的 web 站点，测试可能会轻松一些。如果公司指定使用某个类型的浏览器，那么只需在该浏览器上进行测试。如果所有的人都使用 T1 专线，可能不需要测试下载施加。(但需要注意的是，可能会有员工从家里拨号进入系统) 有些内部应用程序，开发部门可能在系统需求中声明不支持某些系统而只支持一些那些已设置的系统。但是，理想的情况是，系统能在所有机器上运行，这样就不会限制将来的发展和变动。

采取措施：根据实际情况，采取等价划分的方法，列出兼容性矩阵

5 安全测试

即使站点不接受信用卡支付，安全问题也是非常重要的。Web 站点收集的用户资料只能在公司内部使用。如果用户信息被黑客泄露，客户在进行交易时，就不会有安全感。

5.1 目录设置

Web 安全的第一步就是正确设置目录。每个目录下应该有 index.html 或 main.html 页面，这样就不会显示该目录下的所有内容。我服务的一个公司没有执行这条规则。我选中一幅图片，单击鼠标右键，找到该图片所在的路径"…com/objects/images"。然后在浏览器地址栏中手工输入该路径，发现该站点所有图片的列表。这可能没什么关系。我进入下一级目录"…com/objects" ，点击 jackpot。在该目录下有很多资料，其中引起我注意的是已过期页面。该公司每个月都要更改产品价格，并且保存过期页面。我翻看了一下这些记录，就可以估计他们的边际利润以及他们为了争取一个合同还有多大的降价空间。如果某个客户在谈判之前查看了这些信息，他们在谈判桌上肯定处于上风。

5.2 SSL

很多站点使用 SSL 进行安全传送。你知道你进入一个 SSL 站点是因为浏览器出现了警告消息，而且在地址栏中的 HTTP 变成 HTTPS。如果开发部门使用了 SSL，测试人员需要确定是否有相应的替代页面(适用于 3.0 以下版本的浏览器，这些浏览器不支持 SSL。当

用户进入或离开安全站点的时候，请确认有相应的提示信息。是否有连接时间限制？超过限制时间后出现什么情况？

5.3 登录

有些站点需要用户进行登录，以验证他们的身份。这样对用户是方便的，他们不需要每次都输入个人资料。你需要验证系统阻止非法的用户名/口令登录，而能够通过有效登录。用户登录是否有次数限制？是否限制从某些 IP 地址登录？如果允许登录失败的次数为 3，你在第三次登录的时候输入正确的用户名和口令，能通过验证吗？口令选择有规则限制吗？是否可以不登陆而直接浏览某个页面？

Web 应用系统是否有超时的限制，也就是说，用户登陆后在一定时间内（例如 15 分钟）没有点击任何页面，是否需要重新登陆才能正常使用。

5.4 日志文件

在后台，要注意验证服务器日志工作正常。日志是否记所有的事务处理？是否记录失败的注册企图？是否记录被盗信用卡的使用？是否在每次事务完成的时候都进行保存？记录 IP 地址吗？记录用户名吗？

5.5 脚本语言

脚本语言是常见的安全隐患。每种语言的细节有所不同。有些脚本允许访问根目录。其他只允许访问邮件服务器，但是经验丰富的黑客可以将服务器用户名和口令发送给他们自己。找出站点使用了哪些脚本语言，并研究该语言的缺陷。还要需要测试没有经过授权，就不能在服务器端放置和编辑脚本的问题。最好的办法是订阅一个讨论站点使用的脚本语言安全性的新闻组。

6 接口测试

在很多情况下，web 站点不是孤立。Web 站点可能会与外部服务器通讯，请求数据、验证数据或提交订单。

6.1 服务器接口

第一个需要测试的接口是浏览器与服务器的接口。测试人员提交事务，然后查看服务器记录，并验证在浏览器上看到的正好是服务器上发生的。测试人员还可以查询数据库，确认事务数据已正确保存。

这种测试可以归到功能测试中的表单测试和数据校验测试中

6.2 外部接口

有些 web 系统有外部接口。例如，网上商店可能要实时验证信用卡数据以减少欺诈行为的发生。测试的时候，要使用 web 接口发送一些事务数据，分别对有效信用卡、无效信用卡和被盗信用卡进行验证。如果商店只使用 Visa 卡和 Mastercard 卡， 可以尝试使用 Discover 卡的数据。(简单的客户端脚本能够在提交事务之前对代码进行识别，例如 3 表示 American Express，4 表示 Visa，5 表示 Mastercard，6 代表 Discover。)通常，测试人员需要确认软件能够处理外部服务器返回的所有可能的消息。

这种情况在远程抄表中可能会体现到

6.3 错误处理

最容易被测试人员忽略的地方是接口错误处理。通常我们试图确认系统能够处理所有错误，但却无法预期系统所有可能的错误。尝试在处理过程中中断事务，看看会发生什么情况？订单是否完成？尝试中断用户到服务器的网络连接。尝试中断 web 服务器到信用卡验证服务器的连接。在这些情况下，系统能否正确处理这些错误？是否已对信用卡进行收费？如果用户自己中断事务处理，在订单已保存而用户没有返回网站确认的时候，需要由客户代表致电用户进行订单确认。

采取措施：在理解需求的基础上，充分发挥想象力，尽量比较全面的列出各种异常情

况。

7 结论

无论你在测试 internet、intranet 或者是 extranet 应用程序，web 测试相对于非 web 测试来说都是更具挑战性的工作。用户对 web 页面质量有很高的期望。在很多情况下，就像业务功能一样，页面用于维护和发展公共关系，所以第一印象非常重要。

16. 理解 web 性能测试术语

在软件系统日益复杂的今天，性能已经成为软件质量的重要衡量标准之一，这一点尤其体现在和WEB相关的系统上。接下来介绍一些WEB性能测试中的术语，这些术语都是WEB性能测试中出现频繁的比较高的词汇，只有掌握这些基础的性能知识才可以进一步开展测试工作。这些术语主要有并发用户，并发用户数量，请求响应时间，事务响应时间，吞吐量，吞吐率，TPS，点击率，资源利用率等。

并发用户：并发一般分为 2 种情况。一种是严格意义上的并发，即所有的用户在同一时刻做同一件事情或者操作，这种操作一般指做同一类型的业务。比如在信用卡审批业务中，一定数目的拥护在同一时刻对已经完成的审批业务进行提交；还有一种特例，即所有用户进行完全一样的操作，例如在信用卡审批业务中，所有的用户可以一起申请业务，或者修改同一条记录。

另外一种并发是广义范围的并发。这种并发与前一种并发的区别是，尽管多个用户对系统发出了请求或者进行了操作，但是这些请求或者操作可以是相同的，也可以是不同的。对整个系统而言，仍然是有很多用户同时对系统进行操作，因此也属于并发的范畴。

可以看出，后一种并发是包含前一种并发的。而且后一种并发更接近用户的实际使用情况，因此对于大多数的系统，只有数量很少的用户进行“严格意义上的并发”。对于 WEB 性能测试而言，这 2 种并发情况一般都需要进行测试，通常做法是先进行严格意义上的并发测试。严格意义上的用户并发一般发生在使用比较频繁的模块中，尽管发生的概率不是很大，但是一旦发生性能问题，后果很可能是致命的。严格意义上的并发测试往往和功能测试关联起来，因为并发功能遇到异常通常都是程序问题，这种测试也是健壮性和稳定性测试的一部分。

用户并发数量：关于用户并发的数量，有 2 种常见的错误观点。一种错误观点是把并发用户数量理解为使用系统的全部用户的数量，理由是这些用户可能同时使用系统；还有一种比较接近正确的观点是把在线用户数量理解为并发用户数量。实际上在线用户也不一定会和其他用户发生并发，例如正在浏览网页的用户，对服务器没有任何影响，但是，在线用户数量是计算并发用户数量的主要依据之一。

请求响应时间：指的是客户端发出请求到得到响应的整个过程的时间。在某些工具中，请求响应时间通常会被成为"TLLB"，即"Time to last byte",意思是从发起一个请求开始，到客户端接收到最后一个字节的响应时间所耗费的时间。请求响应时间过程的单位一般为"秒"或者"毫秒".

事务响应时间:事务可能由一系列请求组成,事务的响应时间主要是针对用户而言,属于宏观上的概念，是为了向用户说明业务响应时间而提出的.例如:跨行取款事务的响应时间就是由一系列的请求组成的.事务响应时间和后面的业务吞吐率都是直接衡量系统性能的参数.

吞吐量:指的是在一次性能测试过程中网络上传输的数据量的总和.吞吐量/传输时间,就是吞吐率.

TPS:每秒钟系统能够处理的交易或者事务的数量.它是衡量系统处理能力的重要指标.

点击率:每秒钟用户向WEB服务器提交的HTTP请求数.这个指标是WEB应用特有的一个指标:WEB应用是"请求-响应"模式,用户发出一次申请,服务器就要处理一次,所以点击是

WEB应用能够处理的交易的最小单位.如果把每次点击定义为一个交易,点击率和TPS就是一个概念.容易看出,点击率越大,对服务器的压力越大.点击率只是一个性能参考指标,重要的是分析点击时产生的影响。需要注意的是,这里的点击并非指鼠标的一次单击操作,因为在一次单击操作中,客户端可能向服务器发出多个HTTP请求.

资源利用率:指的是对不同的系统资源的使用程度,例如服务器的 CPU 利用率,磁盘利用率等.资源利用率是分析系统性能指标进而改善性能的主要依据,因此是 WEB 性能测试工作的重点.

资源利用率主要针对 WEB 服务器,操作系统,数据库服务器,网络等,是测试和分析瓶颈的主要参考.在 WEB 性能测试中,更根据需要采集相应的参数进行分析.

17. Web 安全测试入门

18. 测试工作总结

19. Web 应用系统易出问题的原因和测试要点

web应用系统是目前最常见的应用系统之一，例如电子商务网站，就是一种典型的web应用系统，关于测试要点，我认为可以有以下几点：

当我们在进行web应用系统的测试时，我们可以做这样一个假设：如果我们是某个电子商务网站的用户，我们会对这个网站有哪些期望呢？

1,有足够的性能，不要在并发用户很多的时候响应速度很慢；

2,有足够好的兼容性，当我们使用 IE 以外的浏览器的时候，网站仍旧能够正常使用；

3,有足够的安全性。至少我们不希望自己的用户名和密码被别人轻易获得；

4,链接的正确性。当我们点击购买一本图书时，我们不希望出现的是是张 CD 的页面；

于是总结起来无外乎，性能，兼容性，安全性，正确性，我认为这是我们测试人员应该关注的要点。

这几方面容易出问题的原因，我想可能跟以下几点有关吧。

1,网站用户的数量可能在某个时间段迅速增加，其增加的速度和用户的总数可能会超过当初设计的极限；

2,用户使用环境的复杂性，系统就有 WINDOWS,LINUX 和其他系统，浏览器又有 IE ,FIREFOX ,NETSCAPE,OPERA 等等；而有的用户显示器可能还仅仅支持 800*600 等等状况的复杂性；

3,网络的人为攻击，病毒泛滥等

4,在一个 web 页面存在大量的连接，且这些链接是在不断更新，难免会出现错误；

由于这些问题的存在，在编写测试计划和测试用例，搭建测试环境和执行测试时，我认为，应该基于 web 应用系统的测试时的特征，有针对性地进行测试工作。

另外，对于web应用系统来说，还可以分为服务器端测试和客户端测试两部分，毕竟web是由服务器端和客户端组成的。

我想，在服务器端，重点进行的应该是性能测试，负载测试和安全测试吧？！

在客户端，则要在兼容性测试上做好工夫。

其实对于 web，最普遍的性能测试应该是负载测试，通过负载测试，测试人员就可以知道系统如何完成预期的或者超过预期的行为。例如：某个电子商务网站设计时，考虑能同时在线的用户为 5000 人。那测试员就需要知道当同时在线 5000 人时，系统的响应状况，也要知道，如果在某个时间段同时在线用户超过系统设计值，假设达到了 10000 人，系统的响应情况。如果同时在线 5000 人时，系统响应速度很慢，以至于很少有用户有足够的耐心来等待完成，那么我想这个 web 系统将不会被用户接受。

对 web 的性能测试，主要还是借助于测试工具，LoadRunner，我想，最好也是要系统学习一下的。

20. 使用 JMeter 测试 web 的应用

2.1 测试环境

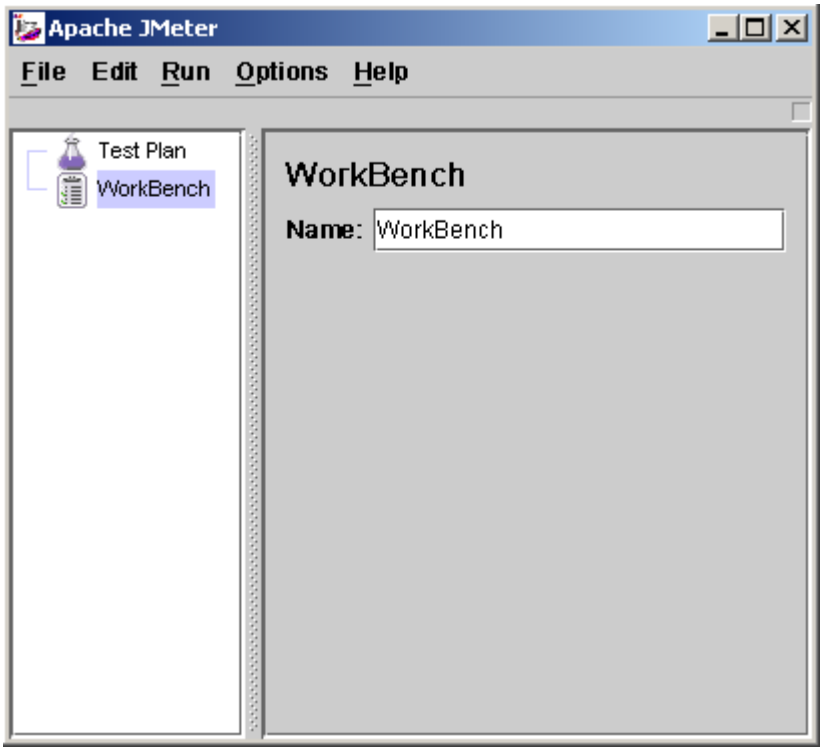
作者使用了Tomcat作为Web服务器进行测试，被测试的内容是一个jsp文件和一个servlet，jsp文件调用JavaBean、打印相关信息，servlet接受用户参数、调用javabean、输出相关信息。详细的内容请参考作者提供的JMeter.war的内容。

2.2 安装启动 JMeter

大家可以到通过http://apache.linuxforum.net/dist/jakarta/jmeter/binaries/jakarta-jmeter-1.9.1.zip 下载 JMeter 的release版本，然后将下载的.zip文件解压缩到C:/JMeter（后面的文章中将使用%JMeter%来引用这个目录）目录下。

现在，请使用%JMeter%/bin 下面的 jmeter.bat 批处理文件来启动 JMeter 的可视化界面，下面的工作都将在这个可视化界面界面上进行操作。下面的图片是 JMeter 的可视化界面的屏幕截图。

图一： JMeter 打开时的屏幕截图



2.3 建立测试计划（Test Plan）

测试计划描述了执行测试过程中 JMeter 的执行过程和步骤，一个完整的测试计划包括一个或者多个线程组(Thread Groups)、逻辑控制（Logic Controller）、实例产生控制器（Sample Generating Controllers）、侦听器(Listener)、定时器（Timer）、比较（Assertions）、配置元素（Config Elements）。打开 JMeter 时，它已经建立一个默认的测试计划,一个 JMeter 应用的实例只能建立或者打开一个测试计划。

现在我们开始填充一个测试计划的内容，这个测试计划向一个 jsp 文件和一个 servlet 发出请求，我们需要 JMeter 模拟五个请求者（也就是五个线程），每个请求者连续请求两次，下面的章节介绍了详细的操作步骤。

2.4 增加负载信息设置

这一步，我们将向测试计划中增加相关负载设置,是 Jmeter 知道我们需要模拟五个请求者，每个请求者在测试过程中连续请求两次。详细步骤如下：

1. 选中可视化界面中左边树的 Test Plan 节点，单击右键，选择 Add'Thread Group，界面右边将会出现他的设置信息框。

2. Thread Group 有三个和负载信息相关的参数：

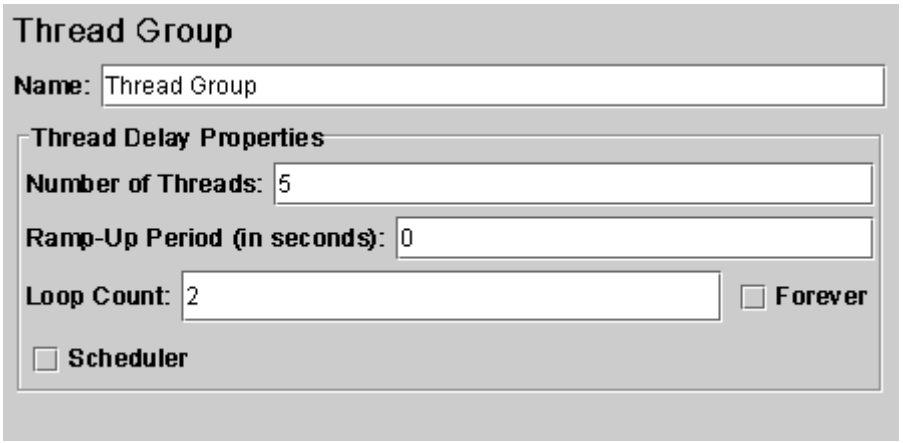
Number of Threads： 设置发送请求的用户数目

Ramp-up period： 每个请求发生的总时间间隔，单位是秒。比如你的请求数目是 5，而这个参数是 10，那么每个请求之间的间隔就是 10 / 5，也就是 2 秒

Loop Count： 请求发生的重复次数，如果选择后面的 forever（默认），那么 请求将一直继续，如果不选择 forever，而在输入框中输入数字，那么请求将重复 指定的次数，如果输入 0，那么请求将执行一次。

根据我们演示例子的设计，我们应该将 Number of Threads 设置为 5，Ramp-up period 设置为 0（也就是同时并发请求），不选中 forever，在 Loop Count 后面的输入框中输入 2，设置后的屏幕截图如下：

图二：设置好参数的 Thread Group。



2.5 增加默认 Http 属性（可选）

实际的测试工作往往是针对同一个服务器上Web应用展开的,所以Jmeter提供了这样一种设置，在默认Http属性设置需要被测试服务器的相关属性，以后的http请求设置中就可以忽略这些相同参数的设置,减少设置参数录入的时间。

我们这里将采用这种属性。你可以通过下面的步骤来设置默认 http 属性：

1. 选中可视化界面中左边树的 Test Plan 节点，单击右键，选择 Add'config element'http request defaults，界面右边将会出现他的设置信息框。

2. 默认 http 属性的主要参数说明如下：

protocal：发送测试请求时使用的协议

server name or ip：被测试服务器的 ip 地址或者名字

path： 默认的起始位置。比如将 path 设置为 / jmeter，那么所有的 http 请求的 url 中都将增加 / jmeter 路径。

port number： 服务器提供服务的端口号

我们的测试计划将针对本机的 Web 服务器上的 Web 应用进行测试，所以 protocal 应该是 http，ip 使用 localhost，因为这个 web 应用发布的 context 路径是 / jmeter，所以这里的 path 设置为 / jmeter，因为使用 Tomcat 服务器，所以 port number 是 8080。设置后的屏幕截

图如下：

图三： 测试计划中使用的默认 Http 参数

2.6 增加 Http 请求

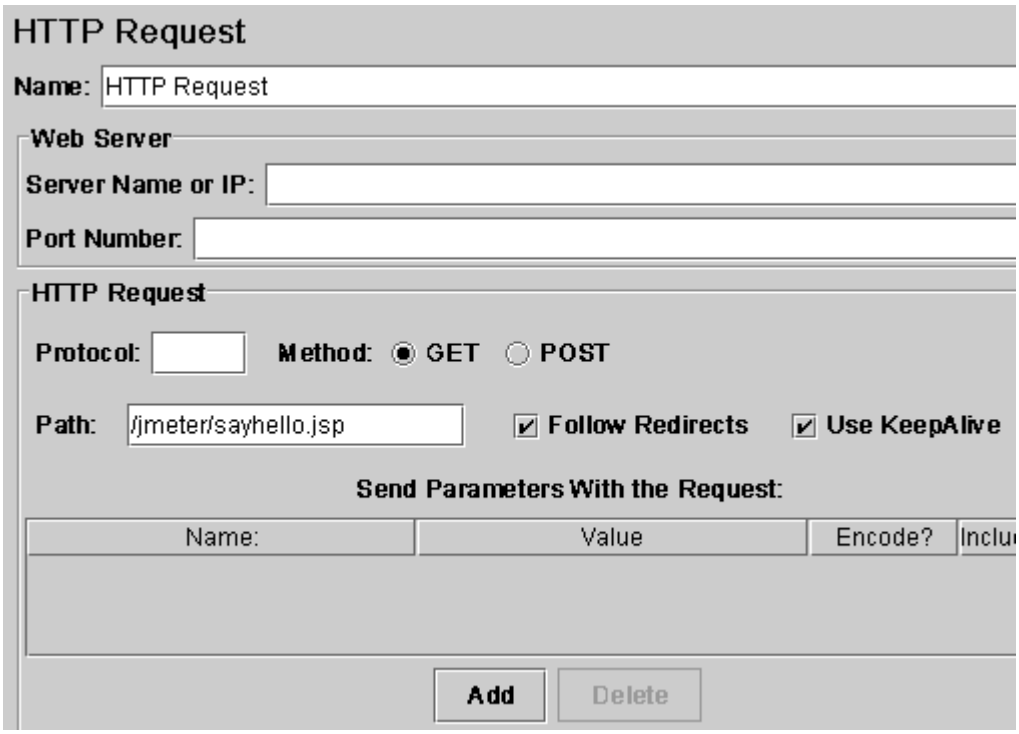
现在我们需要增加 http 请求了，他也是我们测试的内容主体部分。你可以通过下面的步骤来增加性的 http 请求：

1. 选中可视化界面中左边树的 Thread Group 节点，单击右键，选择 Add'sampler'http request，界面右边将会出现他的设置信息框。

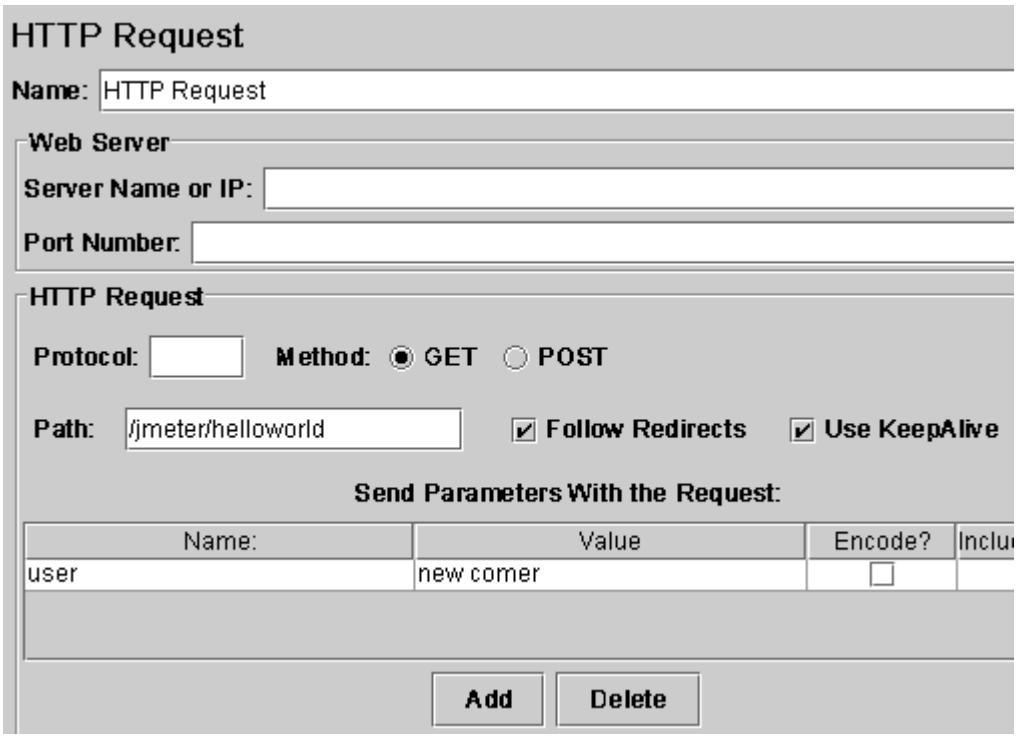
2. 他的参数和 2.5 中介绍的 http 属性差不多，增加的属性中有发送 http 时方法的选择，你可以选择为 get 或者 post。

我们现在增加两个 http 请求，因为我们设置了默认的 http 属性，所以和默认 http 属性中相同的属性不再重复设置。设置后的屏幕截图如下：

图四：设置好的 jsp 测试请求



图五：设置好的 Servlet 测试请求（带参数）



2.7 增加 Listener

增加 listener 是为了记录测试信息并且可以使用 Jmeter 提供的可视化界面查看测试结果，里面有好几种结果分析方式可供选择，你可以根据自己习惯的分析方式选择不同的结果显示方式，我们这里使用表格的形式来查看和分析测试结果。你可以通过下面的步骤来增加 listener：

1. 选中可视化界面中左边树的 Test Plan 节点，单击右键，选择 Add'listener'view result in table，界面右边将会出现他的设置信息和结果显示框。

2. 你可以设置界面上面的 filename 属性设置将测试结果保存到某个文件中界面下面将使用表格显示测试结果，表格的第一列 sampleno 显示请求执行的顺序和编号，url 显示请求发送的目标，sample－ms 列显示这个请求完成耗费的时间，最后的 success 列显示改请求是否成功执行。

界面的最下面你还可以看到一些统计信息，最关心的应该是 Average 吧，也就是相应的平均时间。

2.8 开始执行测试计划

现在你可以通过单击菜单栏 run -> Start 开始执行测试计划了。下面这两个图是作者第一次、第二次执行该测试计划的结果图:

图六：第一次执行后的结果显示

SampleNo	URL	Sample - ms	Success
1	jsp Request	2653	☑
2	jsp Request	2363	☑
3	jsp Request	2643	☑
4	jsp Request	2644	☑
5	jsp Request	2654	☑
6	servlet Request	21	☑
7	servlet Request	20	☑
8	servlet Request	30	☑
9	servlet Request	91	☑
10	jsp Request	60	☑
11	jsp Request	60	☑
12	servlet Request	70	☑
13	servlet Request	10	☑
14	servlet Request	10	☑
15	jsp Request	20	☑
16	jsp Request	20	☑
17	jsp Request	20	☑
18	servlet Request	10	☑
19	servlet Request	10	☑
20	servlet Request	0	☑

No of Samples 20 **Latest Sample** 0 **Average** 670 **Deviation**

图七：第二次执行的结果显示

SampleNo	URL	Sample - ms	Success
1	jsp Request	0	☑
2	servlet Request	0	☑
3	jsp Request	61	☑
4	jsp Request	51	☑
5	servlet Request	20	☑
6	jsp Request	41	☑
7	servlet Request	0	☑
8	jsp Request	40	☑
9	servlet Request	0	☑
10	servlet Request	40	☑
11	jsp Request	60	☑
12	servlet Request	0	☑
13	jsp Request	50	☑
14	servlet Request	10	☑
15	jsp Request	70	☑
16	servlet Request	10	☑
17	jsp Request	30	☑
18	servlet Request	0	☑
19	jsp Request	70	☑
20	servlet Request	10	☑

No of Samples 20 **Latest Sample** 10 **Average** 28 **Deviation**

大家可以看到第一次执行时的几个大时间值均来自于 jsp request，这可以通过下面的理由进行解释：jsp 执行前都需要被编译成.class 文件。所以第二次的结果才是正常的结果。