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：

* 功能测试
* 性能测试（包括负载/压力测试）
* 用户界面测试
* 兼容性测试
* 安全测试
* 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。