第 25 章

Software Engineering: A Practitioner's Approach, Eighth Edition

测试 WebApp

要点浏览

概念：WebApp 测试是一组相关的活动，这些活动都具有共同的目标：发现 WebApp 在内容、功能、可用性、导航性、性能、容量及安全方面存在的错误。为实现这个目标，要同时应用包括评审及运行测试的测试策略。

人员：所有参加 WebApp 测试的 Web 工程师和项目的其他利益相关者（经理、客户、最终用户）。

重要性：如果最终用户遇到错误，就会动摇对 WebApp 的信心，他们会转向其他地方寻找需要的内容及功能，WebApp 就会失败。因此，Web 工程师一定要在 WebApp 上线前尽可能多地排除错误 。

步骤：在进行 WebApp 测试时，首先关注用户可见的方面，之后进行技术及内部结构方面的测试。要进行 7 个步骤的测试：内容测试、界面测试、导航测试、构件测试、配置测试、性能测试及安全测试。

工作产品：在某些情况下，需要制定 WebApp 测试计划。在每种情况下，都需要为每一个测试步骤开发一组测试用例，并且要对记录测试结果的文档进行维护，以备将来使用。

质量保证措施：虽然你从来都不能保证你已经完成了需要的每一项测试，但是，你能够肯定测试已经发现了错误（并且已经改正）。此外，如果你已经制定了一份测试计划，就可以对照测试计划进行检查，以确保所有计划的测试都已经完成。

关键概念

兼容性测试
构件级测试
配置测试
内容测试
数据库测试
负载测试
导航测试
性能测试
计划
质量维度
安全性测试
策略
压力测试
可用性测试
用户界面测试

WebApp 项目的工作普遍都很紧迫。利益相关者由于担心来自其他 WebApp 的竞争，迫于客户的要求，并担心他们将失去市场，因而迫使 WebApp 仓促上线。其结果是，在 Web 开发过程中，技术活动通常开始较晚，例如，有时给 WebApp 测试所剩的时间很短，这可能是一个灾难性的错误。为了避免这种错误的发生，Web 工程团队一定要确保每个工作产品都具有高质量。Wallace 和他的同事 [Wal03] 在谈及这一点时讲道：

> 不应该等到项目完成时才进行测试。在你写第一行代码之前就开始测试。进行持续及有效的测试，这样你将开发出更耐用的 Web 站点。

由于不能在传统意义上对需求模型及设计模型进行测试，因此除了可运行的测试，Web 工程团队还应该进行正式技术评审（第 20 章），目的是在 WebApp 交付最终用户使用之前发现并改正错误。

25.1 WebApp 的测试概念

测试是为了发现（并最终改正）错误而运行软件的过程。这个首次在

第 22 章介绍的基本原理对 WebApp 也是一样的。事实上，由于基于 Web 的系统及应用位于网络上，并与很多不同的操作系统、浏览器（或其他个人通信设备）、硬件平台、通信协议及“暗中的”应用进行交互作用，因此错误的查找是一个重大的挑战。

为了了解 Web 工程环境中的测试目标，我们必须考虑 WebApp 质量的多种维度[⊖]。在此讨论中，我们考虑在讨论 Web 开发的测试工作中都特别关注的质量维度，同时，我们也考虑作为测试结果所碰到的错误的特性以及为发现这些错误所采用的测试策略。

25.1.1 质量维度

良好的设计应该将质量集成到 WebApp 中。通过对设计模型中的不同元素进行一系列技术评审，并应用本章所讨论的测试过程，对质量进行评估。评估和测试都要检查下面质量维度中的一项或多项 [Mil00a]：

- *内容*。在语法及语义层对内容进行评估。在语法层，对基于文本的文档进行拼写、标点及文法方面的评估；在语义层，（所表示信息的）正确性、（整个内容对象及相关对象的）一致性及清晰性都要评估。
- *功能*。对功能进行测试，以发现与客户需求不一致的错误。对每一项 WebApp 功能，都要评定其正确性、不稳定性及与相应的实现标准（例如，Java 或 AJAX 语言标准）的总体符合程度。
- *结构*。对结构进行评估，以保证它能正确地表示 WebApp 的内容及功能，具备可扩展性，并支持新内容、新功能的增加。
- *可用性*。对可用性进行测试，以保证接口支持各种类型的用户，各种用户都能够学会及使用所有需要的导航语法及语义。
- *导航性*。对导航性进行测试，以保证检查所有的导航语法及语义，发现任何导航错误（例如，死链接、不合适的链接、错误链接）。
- *性能*。在各种不同的操作条件、配置及负载下，对性能进行测试，以保证系统响应用户的交互并处理极端的负载情况，而且没有出现操作上不可接受的性能下降。
- *兼容性*。在客户端及服务器端，在各种不同的主机配置下通过运行 WebApp 对兼容性进行测试，目的是发现针对特定主机配置的错误。
- *互操作性*。对互操作性进行测试，以保证 WebApp 与其他应用和数据库有正确接口。
- *安全性*。对安全性进行测试，通过评定可能存在的弱点，试图对每个弱点进行攻击。任何一次成功的突破都被认为是一个安全漏洞。

提问 我们如何在 WebApp 及其所处的环境中评定质量？

引述 创新对于软件测试人员是苦乐参半的事情。当我们似乎知道如何测试一项特定的技术时，恰好出现了一项新技术（WebApp），以前的好办法都不灵了。

James Bach

541

人们已经制定了 WebApp 测试的策略及多种战术，用来测试这些质量维度，在本章后面将对这些进行讨论。

25.1.2 WebApp 环境中的错误

提问 是什么使得在 WebApp 运行中遇到的错误不同于传统软件中遇到的错误？

对于成功的 WebApp 测试，它们所遇到的错误具有很多独特的特点 [Ngu00]：

1. 由于 WebApp 测试发现的很多类型的错误都首先表现在客户端中

⊖ 第 19 章讨论的通用软件质量维度同样可以应用于 WebApp。

（即通过在特定浏览器或个人通信设备上实现的接口），因此 Web 工程师看到了错误的征兆，而不是错误本身。

2. 由于 WebApp 是在很多不同的配置及不同的环境中实现的，因此要在最初遇到错误的环境之外再现错误可能是很困难的，或是不可能的。
3. 虽然许多错误是不正确的设计或不合适的 HTML（或其他程序设计语言）编码的结果，但很多错误的原因都能够追溯到 WebApp 配置。
4. 由于 WebApp 位于客户 / 服务器体系结构之中，因此在三层体系结构（客户、服务器或网络本身）中追踪错误是很困难的。
5. 某些错误应归于静态的操作环境（即进行测试的特定配置），而另外一些错误可归于动态的操作环境（即瞬间的资源负载或时间相关的错误）。

542 上述 5 个错误特点说明：在 WebApp 测试的错误诊断过程中，环境起着非常重要的作用。在某些情况（例如，内容测试）下，错误的位置是明显的；但对于很多其他类型的 WebApp 测试（例如，导航测试、性能测试、安全测试），错误的根本原因很难确定。

25.1.3 测试策略

WebApp 测试策略采用所有软件测试所使用的基本原理（第 22 章），并建议使用面向对象系统所使用的策略和战术（第 24 章）。下面的步骤对此方法进行了总结：

1. 对 WebApp 的内容模型进行评审，以发现错误。
2. 对接口模型进行评审，保证其适合所有的用例。
3. 评审 WebApp 的设计模型，发现导航错误。
4. 测试用户界面，发现表现机制和导航机制中的错误。
5. 对所选择的功能构件进行单元测试。
6. 对贯穿体系结构的导航进行测试。
7. 在各种不同的环境配置下实现 WebApp，并测试 WebApp 对于每一种配置的兼容性。
8. 进行安全性测试，试图攻击 WebApp 或其所处环境的弱点。
9. 进行性能测试。
10. 通过可监控的最终用户群对 WebApp 进行测试；对他们与系统的交互结果进行以下方面的评估，包括内容和导航错误、可用性、兼容性、WebApp 的安全性、可靠性及性能等方面的评估。

关键点 WebApp 测试的总体策略在这里可以总结为 10 个步骤。

网络资源 WebApp 测试方面的优秀文章可以在 www.stickyminds.com/testing.asp 找到。

由于很多 WebApp 在不断进化，因此 WebApp 测试是 Web 支持人员所从事的一项持续活动，他们使用回归测试，这些测试是从首次开发 WebApp 时所开发的测试中导出的。

25.1.4 测试计划

对某些 Web 开发人员来说，使用计划一词（在任何环境中）是不受欢迎的。这些开发
543 人员不做计划，而只是抱着这样的想法开始工作——希望开发一个招人喜爱的 WebApp。更严格的方法则认识到计划工作建立了所有工作遵循的路线图，这种努力是值得的。Splain 和 Jaskiel[Spl01] 在他们关于 WebApp 测试的书中谈道：

除了最简单的网站以外，其他 WebApp 对某种测试计划的需要很快就变得很明显。通常，在随机测试中发现的最初的错误数量很大，以致于在第一次检测到错误时不能对所有的

错误进行定位，这就增加了 Web 站点或 WebApp 测试人员的负担。他们不仅要幻想虚构的新测试，还必须记住以前的测试是如何运行的，以对 Web 站点或应用进行可靠的重复测试，并确保已知的错误都被排除，同时没有引入新的错误。

每位 Web 工程师面临的问题是：我们如何“幻想虚构的新测试”，这些测试应该集中在什么地方？对这些问题的回答都包含在测试计划中。

WebApp 的测试计划确定了：（1）当测试开始时所应用的任务集合[⊖]；（2）执行每项测试任务所生产的工作产品；（3）以何种方式对测试结果进行评估、记录，以及在进行回归测试时如何重用。在某些情况下，测试计划被集成到项目计划中；而有些情况下测试计划是单独的文档。

关键点　测试计划确定了测试任务集和将被开发的工作产品，以及结果被评估、记录及重用的方式。

引述　一般情况下，在其他应用中使用的软件测试技术与在基于 Web 的应用中使用的软件测试技术相同……两种测试类型的不同之处在于 Web 环境中的技术变量增加了。

Hung Nguyen

25.2　测试过程概述

在对 Web 工程进行测试时，首先测试最终用户能够看到的内容和界面。随着测试的进行，再对体系结构及导航设计的各个方面进行测试。最后，测试的焦点转到测试技术能力——WebApp 基础设施及安装或实现方面的问题，这些方面对于最终用户并不总是可见的。

图 25-1 将 WebApp 的测试过程与 WebApp 的设计金字塔（第 17 章）相并列。需要注意的是，当测试流从左到右、从上到下移动时，首先测试 WebApp 设计中的用户可见元素（金字塔的顶端元素），之后对内部结构的设计元素进行测试。 544

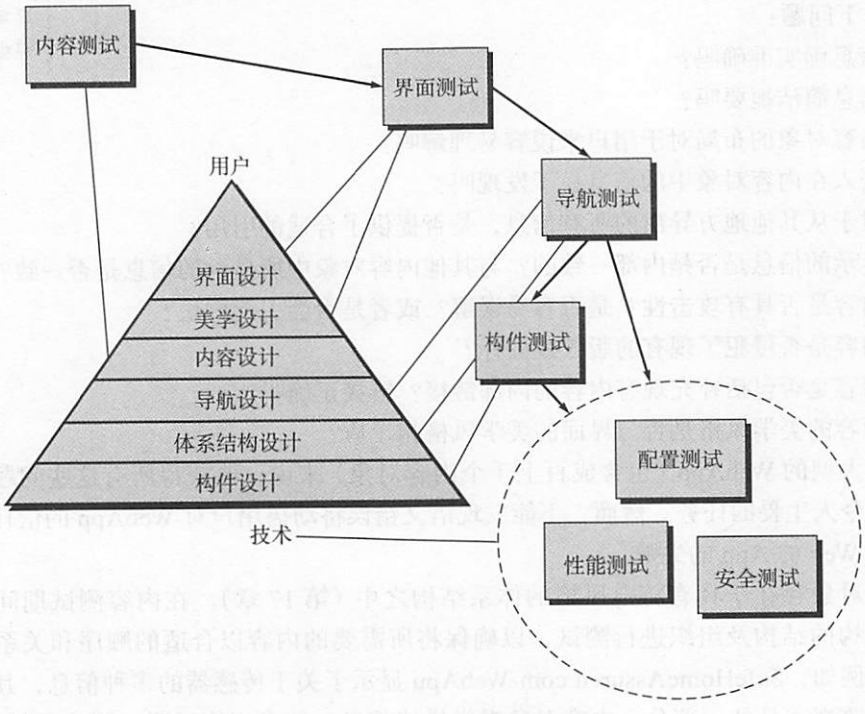

图 25-1　测试过程

⊖ 第 3 章对任务集合进行了讨论。在本书中也使用相关的术语——工作流——来描述完成软件工程活动所需要的一系列任务。

25.3 内容测试

WebApp 内容中的错误可以小到打字错误，也可以大到不正确的信息、不合适的组织、或者违背知识产权法。内容测试试图在用户碰到这些问题及很多其他问题之前就发现它们。

建议 虽然正式的技术评审不是测试的组成部分，但应执行内容评审，以确保内容的质量。

内容测试结合了评审和可运行的测试用例的生成。采用评审来发现内容中的语义错误（在 25.3.1 节讨论）；可运行的测试用于发现内容错误，这些错误可被跟踪到动态导出的内容（这些内容由从一个或多个数据库中获取的数据驱动）。

25.3.1 内容测试的目标

内容测试具有三个重要的目标：（1）发现基于文本的文档、图形表示和其他媒体中的语法错误（例如，打字错误、文法错误）；（2）发现当导航发生时所展现的任何内容对象中的语义错误（即信息的准确性和完备性方面的错误）；（3）发现展示给最终用户的内容的组织或结构方面的错误。

关键点 内容测试的目标包括：（1）发现内容中的语法错误；（2）发现语义错误；（3）发现结构错误。

545

为了达到第一个目标，可以使用自动的拼写和语法检查。然而，很多语法上的错误会逃避这种工具的检查，而必须由审查人员（测试人员）人为发现。实际上，大型网站会借助专业稿件编辑器，以发现打字错误、语法错误、内容一致性错误、图形表示错误和交叉引用错误。

提问 要发现内容中的语义错误，应该提出和回答哪些问题？

语义测试关注每个内容对象所显示的信息。评审人员（测试人员）必须回答以下问题：

- 信息确实准确吗？
- 信息简洁扼要吗？
- 内容对象的布局对于用户来说容易理解吗？
- 嵌入在内容对象中的信息易于发现吗？
- 对于从其他地方导出的所有信息，是否提供了合适的引用？
- 显示的信息是否是内部一致的？与其他内容对象中所显示的信息是否一致？
- 内容是否具有攻击性？是否容易误解？或者是否会引起诉讼？
- 内容是否侵犯了现有的版权或商标？
- 内容是否包括补充现有内容的内部链接？链接正确吗？
- 内容的美学风格是否与界面的美学风格相矛盾？

对于大型的 WebApp（包含成百上千个内容对象）来说，要获得所有这些问题的答案可能是一项令人生畏的任务。然而，不能发现语义错误将动摇用户对 WebApp 的信任，并且会导致基于 Web 的 App 的失败。

内容对象存在于具有特定风格的体系结构之中（第 17 章）。在内容测试期间，要对内容体系结构的结构及组织进行测试，以确保将所需要的内容以合适的顺序和关系展现给最终用户。例如，SafeHomeAssured.com WebApp 显示了关于传感器的多种信息，其中传感器是安全和监视产品的一部分。内容对象提供描述信息、技术规格说明、照片和相关的信息。SafeHomeAssured.com 内容体系结构的测试试图发现这种信息的表示方面的错误（例如，用传感器 Y 的照片来描述传感器 X）。

546

软件工具　Web 内容测试工具

[目标] Web 内容测试工具的目标是识别错误，这些错误会阻止 Web 页以可读和有组织的方式显示 Web 内容。

[机制] 这些工具通常提示输入 Web 资源的 URL 进行测试。每个工具提供错误列表（例如，没有遵循标记语言标准）及如何改正这些错误的建议。

[代表性工具][⊖]

- http://validator.w3.org/，在线的 WC3 工具，可检查 Web 页标记语言的有效性（HTM，XHTML，SMIL，MathML）。
- http://jigsaw.w3.org/css-validator/，在线的 WC3 工具，可检查 CSS 样式表，并用 CSS 样式表检查文档。
- http://validator.w3.org/feed/，在线的 WC3 工具，可检查 Atom 语法或 RSS 文件。

25.3.2　数据库测试

现代 WebApp 要比静态的内容对象做更多的事情。在很多应用领域中，WebApp 要与复杂的数据库管理系统接口，并构建动态的内容对象，这种对象是使用从数据库中获取的数据实时创建的。

例如，用于金融服务的 WebApp 能够产生某种特殊产权（例如，股票或互惠基金）的复杂的文本信息、表格信息和图形信息。当用户已经申请了某种特殊的产权信息之后，就会自动创建表示这种信息的复合内容对象。为了完成此任务，需要下面的步骤：（1）查询股票数据库；（2）从数据库中抽取相关的数据；（3）抽取的数据一定要组织为一个内容对象；（4）将这个内容对象（代表由某个最终用户请求的定制信息）传送到客户环境显示。每个步骤的结果都可能发生错误，并且一定会发生。数据库测试的目标就是发现这些错误，然而，数据库测试会由于以下多种原因而变得复杂：

1. 客户端请求的原始信息很少以能够输入数据库管理系统（DBMS）的形式（例如，结构化查询语言 SQL）表示出来。因此，应该设计测试，用来发现将用户的请求翻译成 DBMS 可处理格式的过程中所产生的错误。
2. 数据库可能离装载 WebApp 的服务器很远。因此，应该设计测试，用来发现 WebApp 和远程数据库之间的通信所存在的错误[⊖]。
3. 从数据库中获取的原始数据一定要传递给 WebApp 服务器，并且被正确地格式化，以便随后传递给客户端。因此，应该设计测试，用来证明 WebApp 服务器接收到的原始数据的有效性，并且还要生成另外的测试，证明转换的有效性，将这种转换应用于原始数据，能够生成有效的内容对象。
4. 动态内容对象一定以能够显示给最终用户的形式传递给客户端。因

提问　哪些问题使 WebApp 的数据库测试变得复杂了？

引述　对于频繁停机、在事务处理的中途停止或可用性差的网站，我们不可能有信心。因此，测试在整个开发过程中具有关键作用。
Wing Lam

547

⊖ 这里提到的工具只是此类工具的例子，并不代表本书支持采用这些工具。在大多数情况下，工具名称被各自的开发者注册为商标。

⊖ 当遇到分布式数据库，或者需要访问数据仓库（第 1 章）时，这些测试可能变得非常复杂。

此，应该设计一系列测试，用来发现内容对象格式方面的错误，以及测试与不同的客户环境配置的兼容性。

考虑这4种因素，对图25-2中所列出的每一个“交互层”[Ngu01]，都应该使用测试用例的设计方法。测试应该保证：（1）有效信息通过界面层在客户与服务器之间传递；（2）WebApp正确地处理脚本，并且正确地抽取或格式化用户数据；（3）用户数据被正确地传递给服务器端的数据转换功能，此功能格式化为合适的查询（例如，SQL）；（4）查询被传递到数据管理层[⊖]，此层与数据库访问程序（很可能位于另一台机器）通信。

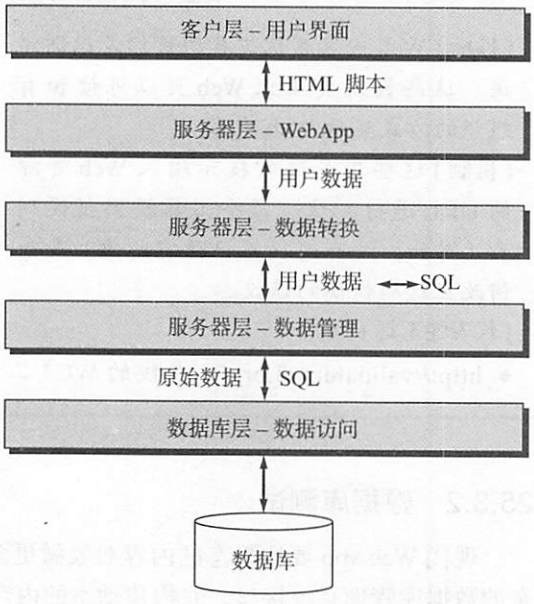

图25-2　交互层

通常使用可复用的构件来构造图25-2所示的数据转换层、数据管理层和数据库访问层，这些可复用的构件都分别进行了合格性确认，并且被打成一个包。如果是这种情况，那么WebApp的测试便集中在图25-2所示的客户层与前两个服务器层（WebApp和数据转换）之间交互的测试用例的设计。

548

应该对用户界面层进行测试，以确保对每一个用户查询都正确地构造了HTML脚本，并且正确地传输给服务器端。还应该对服务器端的WebApp层进行测试，以确保能够从HTML脚本中正确地抽取用户数据，并且正确地传输给服务器端的数据转换层。

应该对数据转换功能进行测试，以确保创建了正确的SQL，并且传给合适的数据管理构件。

为合理地设计这些数据库测试所需要了解的主要技术的详细讨论超出了本书的范围。感兴趣的读者应参看[Sce02]、[Ngu01]和[Bro01]。

25.4　用户界面测试

在Web系统开发过程中，需要在三个阶段对WebApp的用户界面进行验证与确认。在需求分析阶段，对界面模型进行评审，确保与利益相关者的需求及分析模型的其他元素相一致；在设计阶段，对界面设计模型进行评审，确保已经达到了为所有用户界面建立的通用质量标准（第15章），并且正确描述了特定于应用的界面设计问题；由于用户交互是通过界面的语法和语义来表示的，因此，在测试阶段，重点转移到特定于应用的用户交互方面的执行。另外，测试提供了对可用性的最终评估。

建议　除了面向WebApp的详细设计说明以外，这里记录的界面策略可应用于所有类型的客户/服务器软件。

25.4.1　界面测试策略

界面测试检查用户界面的交互机制，并从美学角度对用户界面进行确认。界面测试的总体测试策略是：（1）发现与特定的界面机制相关的错误（例如，未能正确执行菜单链接的错

⊖ 数据管理层一般合并了SQL调用层接口（SQL-CLI），如Microsoft OLE/ADO或Java数据库连接（JDBC）。

误，或者输入数据格式的错误）；（2）发现界面实现导航语义方式的错误、WebApp 的功能性错误或内容显示错误。为了实现此策略，必须启动下面的一些战术步骤：

- 对界面要素进行测试，确保设计规则、美学和相关的可视化内容对用户有效，且没有错误。
- 采用与单元测试类似的方式测试单个界面机制。例如，设计测试用例对所有的表单、客户端脚本、动态 HTML、脚本、流内容及应用的特定界面机制（例如，电子商务应用中的购物车）进行测试。 549
- 对于特殊的用户类，在用例或导航语义单元（NSU，第 17 章）的环境中测试每一种界面机制。
- 与选择用例及 NSU 有所不同，此方法要对所有界面进行测试，发现界面的语义错误。正是在这个阶段，进行一系列的可用性测试。
- 在多种环境（例如，浏览器）中对界面进行测试，确保其兼容性。

25.4.2 测试界面机制

当用户与 WebApp 交互时，是通过一种或多种界面机制来发生交互。在下面的段落中，对每一种界面机制，我们简要介绍一下测试时需要考虑的内容 [Spl01]。

链接。对每个导航链接进行测试，以确保获得了正确的内容对象或功能[⊖]。测试包括与界面布局（例如，菜单条、索引项、每个内容对象的内部链接、外部 WebApp 的链接）相联系的所有链接。

建议 外部链接的测试应该贯穿 WebApp 的整个生命期，支持策略的一部分应该是有规律的、定期的链接测试。

表单。在宏观层次上进行测试，以确保：（1）对表单中的标识域给出正确标记，并且为用户可视化地标识出强制域；（2）服务器接收到了表单中包括的所有信息，并且在客户端与服务器之间的传输过程中没有数据丢失；（3）当用户没有从下拉菜单或按钮组中进行选择时，使用合适的默认项；（4）浏览器功能（例如，“回退”箭头）没有破坏输入到表单中的数据；（5）执行对输入数据进行错误检查的脚本工作正常，并且提供了有意义的错误信息。

建议 每当流行的浏览器的新版本发布时，都应该重新进行客户端脚本的测试及与动态 HTML 相关的测试。

在更具有目标性的层次上，测试应该确保：（1）表单域有合适的宽度和数据类型；（2）表单建立了合适的安全措施，防止用户输入的文本字符串长度大于某预先定义的最大值；（3）对下拉菜单中的所有合适的选项进行详细说明，并按照对最终用户有意义的方式排序；（4）浏览器“自动填充”特性不会导致数据输入错误；（5）Tab 键（或其他键）能够使输入焦点在表单域之间正确移动。

客户端脚本。当脚本运行时，使用黑盒测试发现处理中的一些错误。由于脚本输入通常来自作为表单处理组成部分所提供的数据，因此这些测试通常与表单测试联合进行。 550

动态 HTML。运行包含动态 HTML 的每一个网页，确保动态显示正确。另外，应该进行兼容性测试，以确保动态 HTML 在支持 WebApp 的环境配置中工作正常。

弹出窗口。进行一系列测试，以确保：（1）弹出窗口具有合适的大小和位置；（2）弹出窗口没有覆盖原始的 WebApp 窗口；（3）弹出窗口的美学设计与界面的美学设计相一致；

⊖ 这些测试可以作为界面测试或导航测试的一部分。

（4）附加到弹出窗口上的滚动条和其他控制机制被正确定位，并具有所需的功能。

CGI 脚本。一旦已经接收到经过验证的数据，黑盒测试的重点就集中在数据的完整性（当数据被传递给 CGI 脚本）和脚本处理。此外，进行性能测试以确保服务器端的配置符合 CGI 脚本多重调用的处理要求 [Spl01]。

流内容。测试应该证明流数据是最新的、显示正确、能够无错误地暂停，并且很容易重新启动。

cookie：服务器端的测试和客户端的测试都是需要的。在服务器端，测试应该确保一个 cookie 被正确构建（包含正确的数据），并且在请求特定的内容和功能时，此 cookie 能够被正确地传输到客户端。此外，测试此 cookie 是否具有合适的持久性，确保有效日期正确。在客户端，用测试来确定 WebApp 是否将已有的 cookie 正确地附到了特定的请求上（发送给服务器）。

特定应用的界面机制。测试是否与界面机制定义的功能和特性清单相符合。例如，Splaine 和 Jaskiel[Spl01] 为电子商务应用中所定义的购物车功能提出了下面的检查单：

- 对能够放置到购物车中的物品的最小数量和最大数量进行边界测试（第 23 章）。
- 对一个空购物车的“结账”请求进行测试。
- 测试从购物车中正确地删除一件物品。
- 测试一次购买操作是否清空了购物车中的内容。
- 测试购物车内容的持久性（这一点应该作为客户需求的一部分详细说明）。
- 测试 WebApp 在将来某个日期是否能够记起购物车的内容（假设没有购买活动发生）。

551

25.4.3 测试界面语义

一旦对每一个界面功能都已经进行了“单元”测试，就可以将界面测试的关注点转移到界面的语义测试。界面的语义测试是要“评价设计在照顾用户、提供清楚的指导、传递反馈并保持语言与方法的一致性方面做得如何”[Ngu00]。

从头到尾重新考察一下界面设计模型，我们就能够得到前面段落中所蕴涵问题的部分答案。然而，一旦实现了 WebApp，就应该对每个用例场景（针对每一类用户）进行测试。本质上，用例就变成了设计测试序列的输入。测试序列的目的是发现那些妨碍用户获得与用例相关的目标的错误。

25.4.4 可用性测试

可用性测试也评价用户在多大程度上能够与 WebApp 进行有效交互，以及 WebApp 在多大程度上指导用户行为、提供有意义的反馈、并坚持一致的交互方法。从这种意义上说，可用性测试与界面语义测试是相似的（25.4.3 节）。可用性检查和测试不是集中在某交互目标的语义上，而是要确定 WebApp 界面在多大程度上使用户的生活变得轻松[⊖]。

可用性测试可以由测试人员设计，但是测试本身由最终用户进行。可用性测试可能发生在多种不同的抽象级别：（1）对特定的界面机制（例如，

网络资源 可用性测试的有价值的指导可以在 http://www.keyrelevance.com/articles/usability-tips.htm 找到。

⊖ 这种问题常用术语“用户友好性”来表示。当然，问题在于一个用户对于“友好”界面的感觉可能根本不同于另一个用户。

表单）的可用性进行评估；（2）对所有网页（包括界面机制、数据对象及相关的功能）的可用性进行评估；（3）考虑整个 WebApp 的可用性。

可用性测试的第一步是确定一组可用性类别，并对每一类可用性建立测试目标。下面的测试类别和目标（以问题的形式书写）举例说明了这种方法[⊖]：

提问　可用性的哪些特性变成了测试的焦点？描述了哪些特殊的目标？

552

交互性。交互机制（例如，下拉菜单、按钮、指针）容易理解和使用吗？

布局。导航机制、内容和功能放置的方式是否能让用户很快地找到它们？

可读性。文本是否很好地编写，并且是可理解的[⊜]？图形表示是否容易理解？

美学。布局、颜色、字体和相关的特性是否使 WebApp 易于使用？用户对 WebApp 的外观是否“感觉舒适”？

显示特性。WebApp 是否使屏幕的大小和分辨率得到了最佳使用？

时间敏感性。是否能够及时使用或获取重要的要素、功能和内容？

个性化。WebApp 是否能够适应多种用户或个别用户的特殊要求？

可访问性。残疾人是否可以使用该 WebApp？

对于上面每一种可用性，都需要设计一系列测试。在某些情况下，“测试”可以是对网页的可视化审查；而在有些情况下，可以重新执行界面的语义测试，但在下面的实例中，可用性是极为重要的。

作为一个例子，我们考虑对交互和界面机制进行可用性评估。Constantine 和 Lockwood [Con03] 建议应该对下列界面要素进行可用性评审和测试：动画、按钮、颜色、控制、对话、域、表单、框架、图形、标签、链接、菜单、消息、导航、页、选择器、文本和工具条。评估每个要素时，可以由执行测试的用户对其进行定性分级。图 25-3 描述了用户可能选择的一系列评估“级别”。这些级别可以应用于每个单独的要素、所有的网页或者整个 WebApp。

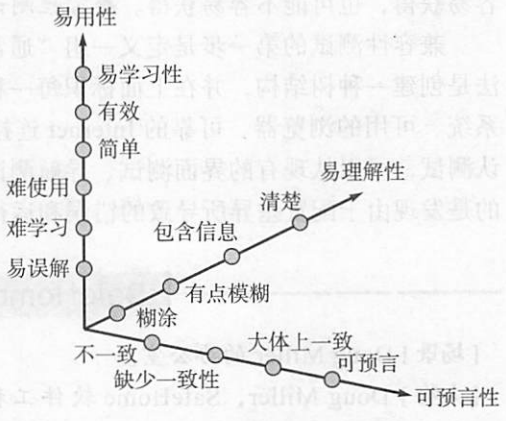

图 25-3　可用性的定性评估

553

软件工具　Web 用户界面测试工具

[目标] 使用 Web 用户界面测试工具的目的是确定网页或网站中的可用性问题或可访问性问题。

[机制] 这里列出了两种类型的工具。有些工具提示输入 Web 页的 URL，并遵照一组启发式，提供了改正失败的建议列表。当用户在网站的网页上工作时，有些工具捕捉用户动作，并且提醒用户。

[代表性工具][⊛]

- http://www.usabilla.com/，usabilla 是一个

⊖ 关于可用性的其他信息参见第 15 章。

⊜ FOG 可读性指数和其他工具可以用来定量地评估可读性。更多的细节参见 http://developer.gnome.org/gdp-style-guide/stable /usability-readability.html。

⊛ 这里提到的工具只是此类工具的例子，并不代表本书支持采用这些工具。在大多数情况下，工具名称被各自的开发者注册为商标。

在线工具，它允许开发者跟踪用户的动作，并在使用网页期间收集用户的意见。

- http://www.google.com/analytics/，Google Analytics 是一个在线工具，可提供综合的网站数据跟踪和分析工具，用于评估网站的可用性。
- http://valet.webthing.com/access/url.html，Web Valet 提供了在线服务，检查网页的可访问性问题。
- http://wave.webaim.org/，Wave 提供了在线服务，通过对网页进行标记来说明可访问性问题。
- http://www.sidar.org/hera/index.php.en，Hera 提供了一个在线服务，运用 Web 内容访问指南来检查网页的可访问性问题。

25.4.5　兼容性测试

不同的计算机、显示设备、操作系统、浏览器和网络连接速度都会对 WebApp 的运行造成很大的影响。每一种计算配置都可能使客户端的处理速度、显示分辨率和连接速度有所不同。操作系统反复无常的行为也可能导致 WebApp 的处理问题。不管 WebApp 中 HTML 的标准化程度如何，不同的浏览器有时会产生稍微不同的结果。特殊的配置所需要的插件可能容易获得，也可能不容易获得。兼容性测试试图在 WebApp 上线前发现这些问题。

兼容性测试的第一步是定义一组“通常遇到”的客户端计算配置和它们的变体。实际做法是创建一种树结构，并在上面标识每一种计算平台、典型的显示设备、此平台支持的操作系统、可用的浏览器、可靠的 Internet 连接速度及类似信息。下一步，导出一系列兼容性确认测试，可以从现有的界面测试、导航测试、性能测试和安全性测试中导出。这些测试的目的是发现由于配置差异所导致的错误和运行问题。

554

SafeHome　WebApp 测试

[场景] Doug Miller 的办公室。

[人物] Doug Miller，SafeHome 软件工程团队经理；Vinod Raman，SafeHome 产品软件工程团队的成员。

[对话]

Doug：对于 SafeHomeAssured.com 电子商务 WebApp 0.0 版，你是怎样看的？

Vinod：外包供应商已经做了很好的工作。Sharon（供应商的开发经理）告诉我，他们正在按我们说的进行测试。

Doug：我希望你和团队的其他人能够对电子商务网站做一点非正式的测试。

Vinod（作苦相）：我想我们将雇用第三方测试公司对 WebApp 进行确认测试。我们仍然在致力于推出产品软件。

Doug：我们将雇用测试供应商进行性能测试和安全性测试，并且我们的外包供应商已经在进行测试了。只是想从另外一种角度看看是否会有帮助，况且，我们想控制成本，所以……

Vinod（叹息）：你在期待什么？

Doug：我想确信界面和所有的导航都是可靠的。

Vinod：我想我们可以从每个主要界面功能的用例开始。

学习 SafeHome
详细说明你需要的 SafeHome 系统
购买一套 SafeHome 系统
取得技术支持

Doug：很好。但是，要走通所有的导航路径，才能得出结论。

Vinod（浏览记录用例的笔记本）：是的，

当你选择“详细说明你需要的 SafeHome 系统”时，此系统将使你：

选择 SafeHome 构件

获得 SafeHome 构件建议

我们要当心每一条路径的语义。

Doug：测试时，需要检查出现在每一个导航节点的内容。

Vinod：当然，还有功能元素。谁在测试可用性？

Doug：哦，测试供应商将配合可用性测试。我们已经雇用了一个市场调查公司列出 20 个进行可用性研究的典型用户，但是，如果你发现了任何可用性问题……

Vinod：我会转给他们。

Doug：谢谢！ Vinod。

25.5 构件级测试

构件级测试也称功能测试，它集中于一系列的测试，试图发现 WebApp 功能方面的错误。每个 WebApp 功能都是一个软件模块（用多种编程语言或脚本语言中的一种实现的），并且可以用第 23 章讨论的黑盒（及在某些情况下的白盒）技术对其进行测试。

构件级测试用例通常受表单级的输入驱动。一旦定义了表单数据，用户就可以选择按钮或其他控制机制来启动运行。在测试基于表单的输入和应用于表单的功能方面，适合采用等价类划分、边界值分析和路径测试（第 23 章）。

除了这些测试用例设计方法，还可以使用称为强制错误测试（forced error testing）[Ngu01] 的技术导出测试用例，这些测试用例故意使 WebApp 构件进入错误条件，目的是发现错误处理过程中发生的错误（例如，不正确或不存在的错误提示信息，由于错误的发生导致 WebApp 失败，由错误的输入而导致的错误输出，与构件处理有关的副作用）。 555

每个构件级测试用例详细说明了所有的输入值和由构件提供的预期输出。将测试过程中产生的实际输出数据记录下来，以供将来的支持和维护阶段参考。

25.6 导航测试

用户在 WebApp 中旅行的过程与访问者在商店或博物馆中漫步的过程很相似。可走很多路径，可以有很多站，可学习和观看很多事情，启动很多活动，并且可以做决策。如我们所讨论的那样，每个访问者到来时都有一系列的目标，在这个意义上，导航过程是可预测的。同时，导航过程又可能是无法预测的，因为访问者受到他所看到的或学到的某件事的影响，可能选择一条路径或启动一个动作，而这对于最初的目标并不是典型的路径或动作。导航测试的工作是：（1）确保允许 WebApp 用户经由 WebApp 游历的机制都是功能性的；（2）确认每个导航语义单元（NSU）都能够被合适的用户类获得。

25.6.1 测试导航语法

实际上，导航测试的第一个阶段在界面测试期间就开始了。应对导航功能进行测试，以确保每个导航都执行了预计的功能。Splaine 和 Jaskiel[Spl01] 建议应该对下面的每个导航功能进行测试：链接及所有类型的锚、重定向（当用户请求一个不存在的 URL 时）、书签、框架和框架集、站点地图以及内部搜索引擎。

引述 我们没有迷路，我们面临定位挑战。

John M. Ford

前面已经提到的某些测试可以由自动工具执行（例如，链接检查），而另外一些要手工设计和执行。导航测试的目的始终是确保在 WebApp 上线之前发现导航功能方面的错误。

25.6.2　测试导航语义

在第 17 章，我们将导航语义单元（NSU）定义为“一组信息和相关的导航结构，在完成相关的用户需求的子集时，这些导航结构会相互协作”[Cac02]。每个 NSU 由一系列连接导航节点（例如，网页、内容对象或功能）的导航路径（称为“导航的路”）定义。作为一个整体，每个 NSU 允许用户获得特殊的需求，这种特殊的需求是针对某类用户，由一个或多个用例定义的。导航测试应检查每个 NSU，以确保能够获得这些需求。

提问　我们测试 NSU 时，必须提出和回答哪些问题？

在测试每个 NSU 时，Web 工程团队一定要回答下面的问题：

- 此 NSU 是否没有错误地全部完成了？
- 在为此 NSU 定义的导航路径的上下文中，（为一个 NSU 定义的）每个导航节点是否都是可达的？

556

- 如果使用多条导航路径都能完成此 NSU，每条相关的路径是否都已经被测试？
- 如果使用用户界面提供的指导来帮助导航，当导航进行时，它们方向正确并可理解吗？
- 是否具有返回到前一个导航节点及导航路径开始位置的机制（不同于浏览器的“回退”箭头）？
- 大型导航节点（即一个长的网页）中的导航机制工作正常吗？
- 如果一个功能在一个结点上运行，并且用户选择不提供输入，那么 NSU 的剩余部分能完成吗？
- 如果一个功能在一个结点上运行，并且在功能处理时发生了一个错误，那么 NSU 能完成吗？
- 在到达所有节点之前，是否有办法终止导航，然后又能返回到导航被终止的地方并从那里继续？
- 从站点地图可以到达每个节点吗？节点的名字对最终用户有意义吗？
- 如果可以从某外部的信息源到达 NSU 中的一个节点，那么有可能推移到导航路径的下一个节点吗？有可能返回到导航路径的前一个节点吗？
- 运行 NSU 时，用户知道他在内容体系结构中所处的位置吗？

建议　如果我们在 Web 工程的分析和设计中没有创建 NSU，则可以将用例应用于导航测试用例的设计，需要提出和回答的一系列问题是一样的。

如同界面测试和可用性测试，导航测试应该由尽可能多的不同的支持者进行。测试的早期阶段由 Web 工程师进行，但后来的测试应该由其他的项目利益相关者、独立的测试团队进行，最后应该由非技术用户进行，目的是彻底检查 WebApp 导航。

软件工具　Web 导航测试工具

[目标] 使用 Web 导航测试工具的目标是识别网站中不可达的任何毁坏的链接或网页。

[机制] 这些工具提示输入 Web 页的 URL，并扫描其标记语言，查找不能返回正确网页类型的链接。某些工具试图抓取整个网站来寻找深层链接中的错误。

[代表性工具]⊖

- http://validator.w3.org/checklink/，一个在线的 WC3 链接检查工具，它可分析 HTML 和 XHTML 文档中无效的链接。
- http://www.relsoftware.com/，Rel Link Checker Lite 的下载网站，Rel Link Checker Lite 是一个免费工具，可用于确认无效链接和孤立文件。

557

25.7 配置测试

配置的可变性和不稳定性是使 Web 工程面临挑战的重要因素。硬件、操作系统、浏览器、存储容量、网络通信速度和多种其他的客户端因素对每个用户都是难以预料的。另外，某个用户的配置可能会有规律地改变（例如，操作系统升级、新的 ISP 和连接速度），其结果可能是客户端环境容易出错，这些错误既微妙又重要。如果两个用户不是在相同的客户端配置中工作，那么一个用户对 WebApp 的印象及与 WebApp 的交互方式可能与另一个用户的体验有很大不同。

配置测试的工作不是去检查每一个可能的客户端配置，而是测试一组很可能的客户端和服务器端配置，确保用户在所有配置中的体验都是一样的，并且将特定于特殊配置的错误分离出来。

25.7.1 服务器端问题

在服务器端，以配置测试用例验证所计划的服务器配置（即 WebApp 服务器、数据库服务器、操作系统、防火墙软件、并发应用）能够支持 WebApp，而不会发生错误。

当设计服务器端的配置测试时，Web 工程师应该考虑服务器配置的每个构件。在服务器端的配置测试期间，需要提出及回答以下问题：

提问 进行服务器端配置测试时，必须提出和回答哪些问题？

- WebApp 与服务器操作系统完全兼容吗？
- 当 WebApp 运行时，系统文件、目录和相关的系统数据是否正确创建？
- 系统安全措施（例如，防火墙或加密）允许 WebApp 运行并对用户提供服务，而不发生冲突或性能下降吗？
- 是否已经利用所选择的分布式服务器配置⊖（如果存在一种配置）对 WebApp 进行了测试？
- 此 WebApp 是否与数据库软件进行了适当的集成？是否对数据库的不同版本敏感？
- 服务器端的 WebApp 脚本运行正常吗？
- 系统管理员错误对 WebApp 运行的影响是否已经得到检查？
- 如果使用了代理服务器，在站点测试时，是否已经明确这些代理服务器在配置方面的差异？

558

25.7.2 客户端问题

在客户端，配置测试更多地集中在 WebApp 与配置的兼容性上，这些配置包括下面构件

⊖ 这里提到的工具只是此类工具的例子，并不代表本书支持采用这些工具。在大多数情况下，工具名称被各自的开发者注册为商标。

⊖ 例如，可能使用单独的应用服务器和数据库服务器，两台机器之间通过网络连接进行通信。

的一种或多种改变 [Ngu01]：硬件、操作系统、浏览器软件、用户界面构件、插件及连通服务（例如，电缆、DSL、WiFi）。除了这些构件，其他配置变量包括网络软件、ISP 的难以预测的变化及并发运行的应用。

为了设计客户端配置测试，Web 工程团队必须将配置变量的数量减少到可管理的数目[㊀]。为了实现这一点，要对每类用户进行评估，以确定此类用户可能遇到的配置。此外，工业市场上的共享数据可以用来预测最可能的构件组合，然后，在这些环境中测试 WebApp。

软件工具　Web 配置测试工具

[目标] 使用 Web 配置测试工具的目标是确定当用不同的 Web 浏览器和操作系统组合显示页面时可能会出现的问题。

[机制] 这些工具提示输入 Web 页的 URL，并允许从很多浏览器和操作系统组合中进行选择。此类工具将显示网页的缩略图，就如同在所选择的每种浏览器版本中所显示的那样。

[代表性工具][㊁]

- http://browsershots.org/，Browsershots 提供在线服务，可以利用很多不同的浏览器和操作系统对网站进行测试。
- http://testingbot.com/，TestingBot 提供了有限免费试用的在线服务，允许使用很多不同的浏览器和操作系统对网站进行测试。

25.8　安全性测试[㊂]

引述　对于管理业务和存储资产来说，Internet 是一个危险的地方。电脑黑客、解密高手、窥探者、骗子、小偷、故意破坏者、病毒发布者和无赖程序承办商都可以自由行动。

Dorothy, Peter Denning

WebApp 的安全性测试是一个复杂的主题，在有效地完成安全性测试之前，必须要对该主题有充分的了解[㊃]。WebApp 和其所处的客户端和服务器端环境对于一些人来说是很有吸引力的攻击目标，这些人包括：外部的电脑黑客，对单位不满的员工，不诚实的竞争者，以及其他想偷窃敏感信息、恶意修改内容、降低性能、破坏功能或者给个人、组织或业务制造麻烦的人。

559

安全性测试用于探查在某些方面所存在的弱点，比如客户端环境，以及当数据从客户端传到服务器并从服务器再传回客户端时所发生的网络通信及服务器端环境。这些领域中的每一个都可能会受到攻击。发现可能会被怀有恶意的人利用的弱点，这是安全性测试人员的任务。

在客户端，弱点通常可以追溯到早已存在于浏览器、电子邮件程序或通信软件中的缺陷。在服务器端，薄弱环节包括拒绝服务攻击和恶意脚本，这些恶意脚本可以被传到客户端，或者用来使服务器操作丧失能力。另外，服务器端数据库能够在没有授权的情况下被访问（数据窃取）。

建议　如果 WebApp 是业务关键的，用来维护敏感的数据，或者很可能成为电脑黑客的目标，则将安全性测试外包给擅长于此的供应商是一个好主意。

㊀ 在每一种可能的配置构件的组合中运行测试是非常耗费时间的。

㊁ 这里提到的工具只是此类工具的例子，并不代表本书支持采用这些工具。在大多数情况下，工具名称被各自的开发者注册为商标。

㊂ 在第 27 章中，安全性测试还被作为安全性工程的一部分进行讨论。

㊃ 由 Cross 和 Fisher[Cro07]、Andrew 和 Whittaker[And06] 及 Trivedi[Tre03] 编写的书籍提供了关于此主题的有用信息。

为了防止这些（和很多其他）攻击，可以使用防火墙（firewalls）、鉴定（authentication）、加密（encryption）和授权（authorization）技术。应该设计安全性测试，探查每种安全性技术来发现安全漏洞。

关键点　应该将安全性测试设计为检验防火墙、鉴定、加密和授权。

在设计安全性测试时，需要深入了解每一种安全机制的内部工作情况，并充分理解所有网络技术。在很多情况下，应将安全性测试外包给擅长这些技术的公司。

软件工具　Web 安全性测试工具

[目标] Web 安全性测试工具的目标是帮助识别网站中可能存在的安全性问题。

[机制] 这些工具通常可以下载，并在开发环境中运行。它们检查应用，查找可以注入可能改变网站功能的有害数据脚本。有些工具可以作为探查工具来使用。

[代表性工具] ⊖

- http://www.mavitunasecurity.com/communityedition/，工具 Netsparker 的下载网站，此工具检查 WebApp 是否存在 SQL 注入漏洞。
- http://enyojs.com/，自由工具 N-Stalker 的下载站点，该工具使用网络攻击特征库对网站执行很多安全性检查。
- http://code.google.com/p/skipfish/，skipfish 工具的下载站点，此工具通过抓取网站的页面给出安全漏洞报告。

25.9　性能测试

你的 WebApp 要花好几分钟下载内容，而竞争者的站点下载相似的内容只需几秒钟，没有什么比这更让人感到不安了；你正设法登录到一个 WebApp，却收到“服务器忙”的信息，建议你过一会儿再试，没有什么比这更让人感到烦恼了；WebApp 对某些情形能够立即做出反应，而对某些情形却似乎进入了一种无限等待状态，没有什么比这更让人感到惊慌了。所有这些事件每天都在 Web 上发生，并且所有这些都是与性能相关的。 560

使用性能测试来发现性能问题，这些问题可能是由以下原因产生的：服务器端资源缺乏、不合适的网络带宽、不适当的数据库容量、不完善或不强大的操作系统能力、设计糟糕的 WebApp 功能以及可能导致客户 – 服务器性能下降的其他硬件或软件问题。性能测试的目的是双重的：（1）了解系统如何对负载（即用户的数量、事务的数量或总的数据量）做出反应；（2）收集度量数据，这些数据将促使修改设计，从而使性能得到改善。

25.9.1　性能测试的目标

设计性能测试来模拟现实世界的负载情形。随着同时访问 WebApp 用户数量的增加，在线事务数量或数据量（下载或上载）也随之增加，性能测试将帮助回答下面的问题：

- 服务器响应时间是否降到了值得注意的或不可接受的程度？
- 在什么情况下（就用户、事务或数据负载来说），性能变得不可接受？

⊖ 这里提到的工具只是此类工具的例子，并不代表本书支持采用这些工具。在大多数情况下，工具名称被各自的开发者注册为商标。

- 哪些系统构件应对性能下降负责？
- 在多种负载条件下，对用户的平均响应时间是多少？
- 性能下降是否影响系统的安全性？
- 当系统的负载增加时，WebApp 的可靠性和准确性是否会受影响？
- 当负载大于服务器容量的最大值时会发生什么情况？
- 性能下降是否对公司的收益有影响？

建议　至少在被最终用户察觉到问题前，WebApp 性能的很多方面是难于测试的，包括网络负载、网络接口硬件的反复无常的行为及类似的问题。

为了得到这些问题的答案，要进行两种不同的性能测试：（1）负载测试，在多种负载级别和多种组合下，对真实世界的负载进行测试。（2）压力测试，将负载增加到强度极限，以此来确定 WebApp 环境能够处理的容量。在下面的两节中将分别考虑每种测试的策略。

561

25.9.2　负载测试

建议　如果一个 WebApp 使用多个服务器提供巨大容量，则必须在多服务器环境中进行负载测试。

负载测试的目的是确定 WebApp 和其服务器环境如何响应不同的负载条件。在进行测试时，下面变量的排列定义了一组测试条件：

N，并发用户的数量

T，每单位时间的在线事务数量

D，每次事务服务器处理的数据负载

每种情况下，在系统正常的操作范围内定义这些变量。当每种测试条件运行时，收集下面的一种或多种测量数据：平均用户响应时间，下载标准数据单元的平均时间，或者处理一个事务的平均时间。Web 工程团队对这些测量进行检查，以确定性能的急剧下降是否与 *N*、*T* 和 *D* 的特殊组合有关。

负载测试也可以用于为 WebApp 用户估计建议的连接速度。以下面的方式计算总的吞吐量 *P*：

$$P = N \times T \times D$$

例如，考虑一个大众体育新闻站点。在某一给定的时刻，2 万个并发用户平均每两分钟提交一次请求（事务 *T*）。每一次事务需要 WebApp 下载一篇平均长度为 3KB 的新文章，因此，可如下计算吞吐量：

$$P = (20\,000 \times 0.5 \times 3\text{KB})/60 = 500\text{KB/s} = 4\text{Mb/s}$$

因此，服务器的网络连接将不得不支持这种数据传输速度，应对其进行测试，以确保它能够达到所需要的数据传输速度。

25.9.3　压力测试

关键点　压力测试的目的是更好地理解：当系统所承受的压力超过它的操作极限时，系统是如何失效的？

压力测试是负载测试的继续，但是，在压力测试中，我们强迫变量 *N*、*T* 和 *D* 满足操作极限，然后超过操作极限。这些测试的目的是回答下面的问题：

- 系统“逐渐”降级吗？或者，当容量超出时，服务器停机吗？
- 服务器软件会给出“服务器不可用”的提示信息吗？更一般地说，用户知道他们不能访问服务器吗？
- 服务器队列请求增加资源吗？一旦容量要求减少，会释放队列所占用的资源吗？

- 当容量超过时事务会丢失吗？
- 当容量超过时数据完整性会受到影响吗？ 562
- N、T 和 D 的哪些值迫使服务器环境失效？如何来证明失效？自动通知会被发送到位于服务器站点的技术支持人员那里吗？
- 如果系统失效，需要多长时间才能回到在线状态？
- 当容量达到 80% 或 90% 时，某些 WebApp 功能（例如，计算密集的功能、数据流动能力）会被停止吗？

有时将压力测试的变型称为脉冲 / 回弹测试（spike/bounce testing）[Spl01]。在这种测试中，增加负载以达到最大容量，然后迅速回落到正常的操作条件，然后再增加。通过回弹系统负载，测试者能够确定服务器如何调度资源来满足非常高的需求，然后当一般条件再现时释放资源（以便为下一次脉冲做好准备）。

软件工具　Web 性能测试工具

[目标] Web 性能测试工具的目标是查找使系统性能低下的瓶颈或者模拟可能导致网站完全失败的条件。

[机制] 在线工具提示输入 Web 资源的 URL 地址。某些工具自动构造一系列模拟的负载测试。某些工具收集网页负载的统计数据以及导航网站时服务器的响应时间。

[代表性工具] ⊖

- http://loadimpact.com/，LoadImpact 是一个在线工具，可以在 Web 服务器上使用模拟的用户负载进行负载影响测试。
- http://www.websitepulse.com/help/testtools.website-test.html/，WebSitePulse 是一个在线工具，可以测量服务器的可用性和网站的响应时间。
- http://www.websiteoptimization.com/services/analyze/，Web Page Analyzer 是一个在线工具，可以测量网站的性能，并提供一个建议的更新列表来改进装载次数。
- http://developer.yahoo.com/yslow/，yslow 是一个在线工具，可以基于高性能网站的开发规则对网页进行分析，并给出改进建议。
- http://tools.pingdom.com/ftp/，pingdom 是一个在线工具，它通过单独分析构件元素，对网页装载时间的瓶颈进行测量。

25.10　小结

WebApp 测试的目标是对每一种 WebApp 质量维度进行检查，发现可能导致质量失效的错误或问题。应该对内容、功能、结构、可用性、导航性、性能、兼容性、互操作性、容量和安全性方面进行重点测试，在 WebApp 设计完成后进行评审，一旦实现了 WebApp，就对其进行测试。 563

WebApp 测试策略检查每个质量维度，从考察内容、功能或导航“单元”开始。一旦单独的单元都已经被确认，重点就转到测试整个 WebApp。为了完成这项工作，很多测试来自

⊖ 这里提到的工具只是此类工具的例子，并不代表本书支持采用这些工具。在大多数情况下，工具名称被各自的开发者注册为商标。

于用户的看法，并由用例所包含的信息所驱动。应编写 WebApp 测试计划，并确定测试步骤、工作产品（例如，测试用例）以及测试结果的评价机制。测试过程包括 7 个不同的测试类型。

内容测试（和评审）主要对内容的多种分类进行测试，目的是发现语义或语法上的错误，这些错误会影响内容的准确性，或对展示给最终用户的方式有影响。界面测试检查用户与 WebApp 之间通信的交互机制，并对界面的美学方面进行确认。目的是发现由于实现糟糕的交互机制、遗漏、不一致或界面语义不明确所导致的错误。

导航测试使用从建模活动中得出的用例，在测试用例的设计中，测试用例对照导航设计检查每个使用场景。对导航机制进行测试，以确保识别并改正妨碍用例完成的任何错误。构件测试检查 WebApp 中的内容和功能单元。

配置测试试图发现针对特殊的客户或服务器环境的错误和兼容性问题，然后进行测试，发现与每种可能配置有关的错误。安全性测试包括一系列测试，探测 WebApp 及其环境中存在的弱点，目的是发现安全漏洞。性能测试包括一系列测试，当对服务器端资源容量的要求增加时，评估 WebApp 的响应时间及可靠性。

习题与思考题

25.1 是否存在应该完全忽视 WebApp 测试的任何情况？

25.2 用你自己的话讨论在 WebApp 环境下测试的目标。

25.3 兼容性是一项重要的质量维度，必须对哪方面进行测试才能确保 WebApp 具有兼容性？

25.4 哪些错误趋向于更加严重，是客户端错误还是服务器端错误？为什么？

25.5 WebApp 的哪些元素可以进行“单元测试”？哪些类型的测试只能在 WebApp 元素被集成之后进行？

564 25.6 开发正式的书面测试计划是否总是必要的？给出解释。

25.7 是否可以公平地说，总的 WebApp 测试策略开始于用户可见的元素，然后移向技术元素？这种策略存在例外吗？

25.8 内容测试是传统意义上的真正测试吗？给出解释。

25.9 描述与 WebApp 的数据库测试有关的步骤。数据库测试是否对客户端和服务器端活动有影响？

25.10 与界面机制相关的测试和界面语义测试之间的区别是什么？

25.11 假设你正在开发满足老年人需要的在线药房（YourCornerPharmacy.com）。药房提供了典型的功能，但也为每位客户维护一个数据库，使得它可以提供药品信息和潜在的药品互作用警告。讨论针对此 WebApp 的任何特殊的可用性测试。

25.12 假设你已经为 YourCornerPharmacy.com（习题 25.11）实现了一个药品互作用检查功能。讨论必须进行的构件级测试的类型，以确保这项功能工作正常。（注意：实现这项功能将必须使用数据库。）

25.13 导航语法测试和导航语义测试之间的区别是什么？

25.14 测试 WebApp 可能在服务器端遇到的每一种配置，这样做可能吗？在客户端呢？如果不可能，Web 工程师如何选择有意义的配置测试集合？

25.15 安全性测试的目标是什么？谁来进行这种测试活动？

25.16 YourCornerPharmacy.com（习题 25.11）已经取得了广泛成功，在运行的前两个月里用户数量剧增。对于固定的服务器端资源集合，画一张图，将可能的响应时间描述为用户数量的函数。对图进行标注，指出“响应曲线”的兴趣点。

25.17 为使应用成功，CornerPharmacy.com（习题 25.11）已经实现了一个特殊的服务——单独处理

处方的重新填写。平均情况下，1000 个并发用户每两分钟提交一次重填请求，WebApp 下载 500B 的数据块来响应。此服务器需要具有的吞吐量是多少 Mb/s？

25.18 负载测试和压力测试之间的区别是什么？

扩展阅读与信息资源

WebApp 测试方面的文献仍在不断发表。这方面最新出版的书籍中，由 Andrews 和 Whittaker(《How to Break Web Software》，Addison-Wesley，2006)、Ash（《The Web Testing Companion》，Wiley，2003）、Nguyen 和他的同事（《Testing Applications for the Web》，second edition，Wiley，2003）、Dustin 和他的同事（《Quality Web Systems》，Addison-Wesley，2002）以及 Splaine 和 Jaskiel[SPL01] 撰写的书论述得最全面。Whitaker 和他的同事（《How Google Tests Software》，Addison-Wesley，2012）描述了另外的 Web 测试实践。Mosley（《Client-Server Software Testing on the Desktop and the Web》，Prentice Hall，1999）论述了客户端和服务器端的测试问题。

David（《Selenium 2 Testing Tools: A Beginner's Guide》，Packit Publishing，2012）和 Stottlemeyer（《Automated Web Testing Toolkit》，Wiley，2001）介绍了 WebApp 测试策略和方法方面的有用信息，并对自动化测试工具进行了有价值的讨论。Graham 和他的同事（《Experiences of Test Automation》，Addison-Wesley，2012 和《Software Test Automation》，Addison-Wesley，1999）介绍了自动化工具方面的其他资料。

微软（《Performance Testing Guidance for Web Applications》，Microsoft Press，2008）和 Subraya（《Integrated Approach to Web Performance Testing》，IRM Press，2006）对 WebApp 性能测试进行了详细论述。Hope 和 Walther（《Web Security Testing Cookbook》，O'Reilly，2008）、Skoudis（《Counter Hack Reloaded》，second edition，Prentice Hall，2006）、Andreu（《Profession Pen Testing for Web Applications》，Wros，2006）、Chirillo（《Hack Attacks Revealed》，second edition，Wiley，2003）、Splaine（《Testing Web Security》，Wiley，2002）以及 Klevinsky 和他的同事（《Hack I.T.：Security through Penetration Testing》，Addison-Wesley，2002）为必须设计安全性测试的人提供了非常有用的信息。另外，讲述软件安全测试的书籍通常能够为那些必须测试 WebApp 的人提供重要的指导。有代表性的书籍包括：Engebretson（《Basics of Hacking and Penetration Testing》，Syngress，2011）、Basta 和 Halton（《Computer Security and Penetration Testing》，Thomson Delmar Learning，2007）、Wysopal 和他的同事（《The Art of Software Security Testing》，Addison-Wesley，2006）以及 Gallagher 和他的同事（《Hunting Security Bugs》，Microsoft Press，2006）。

在网上可以获得大量 WebApp 测试的信息资源，最新参考文献可以在 SEPA 网站 www.mhhe.com/pressman 找到。