TURING 图灵交互设计丛书

Web Anatomy

Interaction Design Frameworks that Work

网站设计解构

有效的交互设计框架和模式

[美] Robert Hoekman, Jr.
Jared Spool 著

向怡宁 译

人民邮电出版社
北 京

图书在版编目（CIP）数据

网站设计解构 : 有效的交互设计框架和模式 / (美) 霍克曼 (Hoekman,R.), (美) 斯普乌尔 (Spool,J.) 著; 向怡宁译. — 北京 : 人民邮电出版社, 2010.8
(图灵交互设计丛书)
书名原文: Web Anatomy: Interaction Design Frameworks that Work
ISBN 978-7-115-23375-2

Ⅰ. ①网… Ⅱ. ①霍… ②斯… ③向… Ⅲ. ①网站－设计 Ⅳ. ①TP393.092

中国版本图书馆CIP数据核字(2010)第126519号

内 容 提 要

本书主要介绍了交互设计框架化体系结构的各个组成元素，并使用成功与不成功的网站作为案例，深入剖析了它们的功能以及工作原理，目的是解决 Web 项目中反复出现的三类问题：如何将高层面的程序目标转化为低层面的设计细节，如何创新，以及如何用低成本换来高回报。书中案例形象生动，语言诙谐幽默，是 Web 交互设计师必备的完整指南。

本书适合各层次 Web 设计人员和开发人员使用。

图灵交互设计丛书

网站设计解构：有效的交互设计框架和模式

◆ 著 [美] Robert Hoekman, Jr. Jared Spool
译 向怡宁
责任编辑 傅志红
执行编辑 李 静

◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街14号
邮编 100061 电子函件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京 印刷

◆ 开本：800×1000 1/16
印张：12.25
字数：200千字 2010年8月第1版
印数：1－3 500册 2010年8月北京第1次印刷

著作权合同登记号 图字：01-2009-1523号

ISBN 978-7-115-23375-2

定价：59.00元

读者服务热线：(010)51095186 印装质量热线：(010)67129223
反盗版热线：(010)67171154

版 权 声 明

Acknowledgements 致　谢

Robert Hoekman, Jr. 感谢以下人士——

Wendy Sharp，再次感谢，你真是一位伟大的编辑和朋友；Jacqueline Aaron，因为你又完成了一次卓越的审稿；Kathleen Cunningham，因为你非凡的图书设计；感谢 Nancy Reunzel，令人尊敬的出版人，因为，唔，你是我们尊敬的出版人；以及其他许多人，包括但不限于 Glenn Bisignani、Nancy Davis、Gary Paul Prince，还有 New Riders 公司其他头脑冷静、处乱不惊的诸位——现在我要说啦——没有你们，这本书不可能完成。

当然了，我还要感谢我的妻子 Christine，她在我第四次叫嚷着“上帝，这本该死的书什么时候才能完成”的时候，满怀鼓励地对我点头微笑。（事实证明，写作的过程并不总是充满了魅力和玫瑰。）

* * * * * *

Jared Spool 感谢以下人士——

User Interface Engineering 公司令人叫绝的团队（所有离去和留下的人），尤其是那些为本书引述的研究成果出过力的人。UIE 的座右铭就是“通过令人震撼的体力劳动来推动知识向前发展”。收集那些人们如何使用技术的统计数据的确是一项繁重的工作。这些人包括 Will Schroeder、Lori Landesman、Carolyn Snyder、Tara Scanlon、Nina Gilmore、Matthew Klee、Joshua Porter 还有 Christine Perfetti，正是他们（以及其他很多人）把研究变成了现实。我们在本书中看到的一切都源于这些人的努力和付出。

我还要感谢 Wendy Sharp 以及 New Riders 的团队耐心地纠正书中的错误，正是他们的奉献让这本书得以成形。我还要衷心地感谢本书的合著者。Robert 包揽了所有吃力不讨好的工作，是一个值得信赖的好伙伴。他在工作中永远动力十足，而且（除了有时我会让他抓狂之外）整个过程中我们得到了极大的乐趣！

还要特别感谢我的孩子 Ari 和 Reed，当我努力为他们创造一个更美好的世

界时，他们给予我大力的支持和鼓舞。最后，我应该给 Dana Chisnell 赠送一座荣誉雕像，以纪念她在此过程中给予的支持和做出的贡献。

* * * * * *

我们两人要共同感谢所有的先行者们，感谢你们在提倡设计模式和组件的标准和最佳实践方面所付出的艰苦卓绝的努力。这些人包括 Christian Crumlish、Luke Wroblewski、Teresa Neil、Bill Scott、Martijn van Welie、Dan Brown 以及 Eight Shapes 公司的 Nathan Curtis，还有 Jenifer Tidwell，等等。没有你们的努力，互联网将变成庞大、无组织的一团乱麻。唔，真的会相当地乱。

Contents 目 录

第一部分 框架体系简介

第二部分 框架体系


LOGO
SEARCH
PRODUCT GALLERY
USER ID
PASSWORD
SIGN IN
CONTENT
NEWS
NEWS

第一部分

框架体系简介

1. 框架体系呼之欲出

1998 年，可用性专家 Rolf Molich[①] 分别给 9 个团队 3 周的时间对 Web 邮箱 www.hotmail.com 进行评估。该实验是被他称为 CUE（Comparative Usability Evaluations，相对可用性测试）系列实验[②]的一部分，试图建立起一套切实可行的可用性测试标准。在每一项测试中，Rolf 都会请多个可用性团队对同一个设计进行自由评估。

这次测试被称为 CUE-2，因为它是该系列实验中的第二项测试。在各团队记录下的结果中，反映出一个令人惊诧的趋势。

尽管可用性专家们在检测界面问题时宣称使用的方法非常科学，但是可用性评估本身却“并不十分科学”。

---

① Rolf Molich 于 1984 年投身于可用性方面的研究，1993 年在丹麦创立了可用性咨询公司 DialogDesign。他同时也在 Nielsen Norman Group（由 Donald Norman、Jakob Nielsen 和 Bruce Tognazzini 三位著名用户体验专家创办的可用性咨询公司）的大型可用性测试项目中担任首席研究员。Rolf Molich 也是“启发式评估法”的发明人之一（另一位发明人是 Jakob Nielsen），并著有 *User Friendly Computer Systems*（《用户友好的计算机系统》）一书，英文版本名为 *Usable Web Design*（《易用网页的设计》）。——译者注（以下未特别说明的均为译者注）

② 该实验包含了 7 项研究案例（CUE-1 至 CUE-7），聘请了 60 多个专业可用性团队为同一个应用程序进行测试或评估。

在一次接受 Christine Perfetti[①]（她当时还在 Jared 的公司 User Interface Engineering[②]即 UIE）的采访时，Rolf 谈起了自己的评估，说道：

> CUE-2 的各团队总共提交了 310 个不同的可用性问题。提到次数最多的问题有 7 个团队提交。只有 6 个问题被超过半数的团队提交，而有 232 个问题（75%）只被提交了一次。许多标记为“严重”的问题都只有一个团队提交。即使是那些所有团队都测试过的常见任务，得到的结果也完全不同——这些任务中有 70% 左右的发现都是唯一的。

在 2003 年进行的 CUE-4（参见 http://www.dialogdesign.dk/Summary3.htm）中，共有 17 个团队被聘请对 www.hotelpenn.com[③]进行评估，该网站有一个基于 Flash 的酒店客房预定系统，由 iHotelier 开发[④]。在这 17 个团队中，9 个进行可用性测试，其余的 8 个进行专家评审。

所有团队共提交了 340 个可用性问题。然而，这些问题中只有 9 个被超过半数的团队提交。而且共有 205 个问题（占提交结果的 60%）只被发现过一次，其中又有 61 个被标记为“严重”或者“极为严重”的问题。

思考一下。

为了在评估过程中发现所有“严重”的可用性问题，Hotmail 项目聘请了 9 个可用性团队。而在 CUE-4 中，为了找出所有 61 个严重问题，Hotel Penn 则需要聘请 17 个可用性团队。17 个啊！

在被问到开发团队如何确信已经准确标识出网站中存在的问题时，Rolf 总结道：

> 很简单：他们根本没法确定！

那么在今后创建可用性网站的过程中，可用性测试是否仍将扮演重要角色？ Rolf 继续解释道：

---

① Christine Perfetti 是一位颇受欢迎的讲师和技术指导，经常出席世界各地的会议并且是最受欢迎的发言人之一。她曾任 UIE 公司副总裁兼行政董事，后自己创办 Perfetti Media。

② User Interface Engineering（用户界面工程）是一家专注于网站和产品可用性方面的咨询培训公司，由资深用户体验专家 Jared M. Spool（也就是本书的第二作者）于 1988 年创立。公司网站是 http://www.uie.com/。

③ www.hotelpenn.com 是宾夕法尼亚酒店的网站。宾夕法尼亚酒店是世界上最著名的酒店之一，总部位于纽约曼哈顿核心区。

④ iHotelier 现已更名为 TravelCLICK。这家 1989 年创立的公司专为宾馆及酒店提供一站式电子商务解决方案，在全球 140 多个国家拥有 15000 多位客户。网址为 http://www.travelclick.net/。

> （我们）应该主要在项目的中期阶段使用它们，以此来增进同事间的信任。然后应采取其他更合算且有效的预防措施，例如为界面进行高可用性的模块化分组，以成熟的标准和准则为基础进行评审，以及情境式调查[①]。

我们认为 Rolf 的回答并不完整。首先，为了判断你是否在解决最重要的问题——解决这些问题可以让我们获得最大的收益，可用性测试也许并不比一位专家或者一位评估人员执行的启发式评估[②]来得更为准确或可靠，但可用性测试可以而且确实已经为我们理解人们的网络交互行为提供了巨大的帮助，它仍然应当被视为一种必不可少的工具。

其次，任何评估或观测的方法都应当被放在上下文情境中分析。“页面浏览量”和“平均页面花费时间”这两个指标都曾经被视为判断网站效率高低的衡量标准，但它们必须结合被访问页面自身的目标来考虑，否则就毫无意义。一个用户访问了一系列的页面，这种举动是因为任务流程很有效呢，还是因为他根本找不到想要的内容？用户在某个页面上花费了大量的时间，是因为他们很感兴趣呢，还是因为遇到了困难？ NYTimes.com（纽约时报的网站）无疑希望读者能在网页上逗留足够长的时间，阅读整篇文章或者浏览所有的标题，但 Google 的目标却是让用户能在搜索页面中尽快找到需要的内容，然后绝尘而去。较长的“花费时间”在 NYTimes.com 上可能预示着一篇高质量的文章，而在 Google.com 上却可能预示着开发团队一次重大的失误。

不管怎样，可用性评估无法告诉设计师怎样才能做出好的设计，它只能帮助我们发现已有设计中存在的问题。从这一点来看，下面这段 Rolf 的陈述颇能引起我们的共鸣：

> 我希望有一天我们能拥有庞大的资源库，里面存放着经过现实用户的周详测试而且证明可用的通用界面模块（building block）。我还希望我们能向大家展示，由可用性专家把这些模块组装成完全成熟的网站，从而高效地生产出大量具有极高可用性的网站。

---

① 情境式调查（Contextual Inquiry）是一种以用户为中心的设计方法（User-Centered Design，UCD）。它要求研究者在自然情境下旁观研究对象的实际工作，并记录下详细的信息。在该过程中，研究者应避免影响到研究对象的日常工作环境。研究者试图通过这种观察和讨论帮助他们设计出能协助、缩短甚至削减用户工作流程的产品。

② 启发式评估法（Heuristic Evaluation Method）的发明人正是 Rolf Molich 和 Jakob Nielsen。它是一种“打折”的可用性观测方法，用于在计算机软件中标识用户界面设计方面的问题。观测者依据已经过证实的可用性原则（启发式方法）对界面执行的交互进行评估。这种评估方式如今被广泛应用于新媒体领域，尤其适用于需要在短期内设计出用户界面，或者由于预算金额的限制无法进行其他界面测试的情况。

庞大的资源库。界面模块。周详的测试。

如果你正在疑惑 Rolf 的故事和手中的这本书有什么关系，上面这三个词已经包含了答案。这是一本关于模块的书。这里没有原理，没有概念，没有代码，*只有模块*。

本书的第一个目标是近距离地观摩这些界面模块——认出它们，剥离它们，分析它们的功能以及工作原理。只有通过这些，身为设计师的我们才能将它们“组装成完全成熟的网站”并且创造出“极高可用性”。

但这只是第一步，我们还有其他的一些目标。准确地说，我们希望本书能够解决在每个 Web 项目中都反复出现的三种问题。

首先，我们对一个应用程序目标的理解是*高层面*的，而要想把它转化为*低层面*的设计细节，这个过程可能会非常艰难。这就好像拿走你的魔杖，绑住一只手，然后要你把蒸汽变成石头一样。在整个过程中，弄明白从哪里开始可能是最困难的一步，而即使你认为一切都已搞定，也很难确保不会有任何遗漏。当你忙着迎合业务目标的时候，又如何能保证自己完全满足了用户的需求呢？

其次是标准与创新的问题。在绝大多数情况下，*标准就意味着无趣*。我们也许都会同意，在设计项目中最大快人心的部分就是凭空创造出一个前人尚未想到的解决办法。这种时刻非常令人兴奋——心如撞鹿、肾上腺素急速上升。但是大部分项目中，这种情况非常罕见。这是因为，即使是最具创新精神的项目，能称得上“前所未见”的地方可能也只占整个项目的 20%，其余都是基于标准的支撑功能。这些工作不会让我们心如撞鹿、肾上腺素急速上升——它们只是“苦力活”，是每个项目都必须要做的一部分。也正因为如此，我们往往不太重视它们。

比如说，此刻某个团队正在创建一个新的设计，里面包含了某些令人惊叹、前所未见的功能。但要想运用这些突破性的成果，用户需要先注册并登录。

注册功能并不新颖，并不会让人激动，对开发也没有太多挑战。但是无数团队都不得不一而再、再而三地设计这个功能，就好像之前从未创建过一样。这使得设计“注册”的工作非常枯燥乏味，但是忽视它却会给开发者带来很大的麻烦。因为它不是项目中最“帅”的部分，所以往往得不到重视，结果却可能给用户带来难以使用的挫折体验，甚至造成公司在收入上的损失。

而另一方面，创新也可能会产生问题。如果你已经致力于可用性界面设计

一段时间了，你可能会和我们注意到同样的事情：可用性和创新常常会互斥。酷和易用有时候是一对欢喜冤家。

Live.com[①]在它首次发布的时候就遇到了这种麻烦，其中使用了一种有争议的无限滚动设计模式。开发者的本意是，如果用户希望查看更多搜索结果，这个方案将为他们省去被迫翻页、等待载入新页面的麻烦，而可以直接在单个页面中即时载入后续的搜索结果，这样当用户往下滚动页面时就能随时看到新的内容。然而，实际的结果并不如人所愿。用户们并不熟悉这一变化，因此当他们发现这个页面似乎永远也拉不到头时，很容易就会产生挫折感，而且很快就会厌恶这一创意。一句话，他们觉得它的表现不像 Google。无限长的页面滚动可能确实很酷，但是它对用户来说太莫名其妙，因此完全没有达到预期的效果[②]。（第 4 章详细讲述了这个故事。）

每个人都渴望自己的产品能在市场上引起轰动，这需要我们创造出光芒四射的界面，但不能因此而牺牲可用性。要想做好这一点可能非常困难。当我们打破陈规时，实际上是在冒险，因为很可能用户根本无法理解我们设计出的界面。框架体系为我们提供了一种方法，把无趣的支撑功能进行标准化，这样就能避免重复开发，从而有更多的时间进行真正的发明创造。

最后是“低成本、高回报”的问题。Web 开发的团队越来越精简（很多团队的规模都只有 10 年前的一半），但公司对每个团队的期望却越来越多。与此同时，项目也变得比以往更为精密和复杂。为了节省时间和精力，开发团队必须开始考虑使用以前曾经完成过的功能。就像我们之前所说的，开发者不得不反复设计注册功能，就好像之前从未曾创建过一样。它确实被创建过（全世界的开发者都在往自己的产品里添加注册功能，估计已经不下数百万次了），但尽管它就在那里，却仍然要从头再来一遍。所有这些重复创造、重复开发的工作导致效率极为低下。为了降低在这方面耗费的精力（并且给刺激、有趣的创新部分留下更充裕的时间），开发团队需要可重复利用的设计。

重用是如今最应该优先考虑的事。

---

① Live.com 本是 Windows Live Personalized Experience（Windows Live 个性化体验）的用户自定义门户，由微软于 2005 年 11 月发布。它是最初发布的 Windows Live 服务之一。但是随着微软的战略调整，Live.com 被 My.Live.com 所取代，而原来的域名 Live.com 则指向了搜索引擎 Bing（取代了之前的 Live Search）。因此，作者文中所提到的 Live.com 实际上指的是 2009 年 6 月首次发布的 Bing。

② 如今的 Bing 图片搜索仍然采用了无限滚动的交互方式。不过所谓的“无限滚动”仍然是有限的：无论搜索到多少结果，Bing 最多显示前 1 000 个，而且不提供翻页功能。

1.1 可重用策略

要想解决我们之前所述的所有问题，核心理念其实非常简单：Web 开发团队需要日积月累、稳步实施可重用策略。

可重用策略可以划分为三种资源库：模式、组件，以及交互设计的框架体系。这些资源库能让开发团队充分利用丰富的成品资源，从而提高工作的速度和效率。

我们发现，那些成功实施可重用策略的团队已经尝到了实实在在的甜头。首先，他们可以调用已经（按最基本水平）实现的成品集，快速地拼凑起有效的设计雏形，从而达到让设计尽快启航的目的。这些团队完成整个设计的时间似乎也更短，而且可以周全地涵盖一切有利于提升用户体验的细节和精妙之处。其次，尽管他们在无趣的支撑功能上并未花费太多时间，其设计成果却似乎具备更高的可用性，而且在所有功能中的表现都始终如一。最后，团队进行迭代设计[①]更为快捷，这让他们有机会在设计尚具可塑性的时候对它进行精细的调整。

在团队的可重用策略中，模式、组件和交互设计的框架体系三者扮演着不同的角色，但都举足轻重。在下一章，我们会更为详细地讨论每一种资源，不过首先请允许我在这里进行一个简要的介绍。

1.1.1 模式：预期行为的锦囊

设计模式是重用拼图中的第一块零件，它的灵感来自于 Christopher Alexander[②]提出的建筑模式这一概念［见其 1977 年所著的 *A Pattern Language: Towns, Building, Construction*（《建筑模式语言：城镇、建筑、构造》）一书，牛津大学出版社］。Alexander 观察人们生活和工作的具体行为，然后创建出一系列关于楼房建筑如何支持这些行为的可重用描述。模式并不会使建筑师们落入一成不变的窠臼，相反地，它为他们提供资源以确保所有细节的正确性。

① 迭代设计是一种具有更高成功率和生产率的软件开发方式。在迭代式设计中，整个开发过程被分解为一系列周期短小、固定的小项目，被称为一系列的“迭代”。每一次迭代都包括需求分析、设计、实践与测试。通过这种方法，开发工作可以在需求全部确定之前启动，并在一次迭代中完成系统的一部分功能或开发，然后通过客户的反馈来细化需求，并开始新一轮的迭代。

② Christopher Alexander 于 1936 年出生于奥地利维也纳。他是当代建筑大师，以其设计理论和丰富的建筑设计作品而闻名于世。Alexander 认为，建筑的使用者比建筑师更清楚他们需要什么，因此创造并以实践验证了模式语言（与 Sarah Ishikawa 和 Murray Silverstein 合作）。建筑模式语言赋予所有人设计和建造的能力。1958 年 Alexander 移居美国，目前是加州大学伯克利分校的终身荣誉教授。

如今的设计模式也与之类似。比如，让我们假设一位正在订票的用户需要输入日期。有哪些支持输入日期的设计呢？一个带自动分析功能的文本框？分别表示年、月、日的三个数字下拉列表？可以直接点选日期的弹出式日历？

针对同一种行为，不同的选择体现出了不同的设计。当开发团队指定一种最适合他们（及其用户）的设计时，就能将其定义为一个模式。日后，当团队需要响应类似的行为时，就能以相似的方式进行响应，利用之前的工作成果来满足用户已经确立的需求。图 1-1 为一个模式文档。

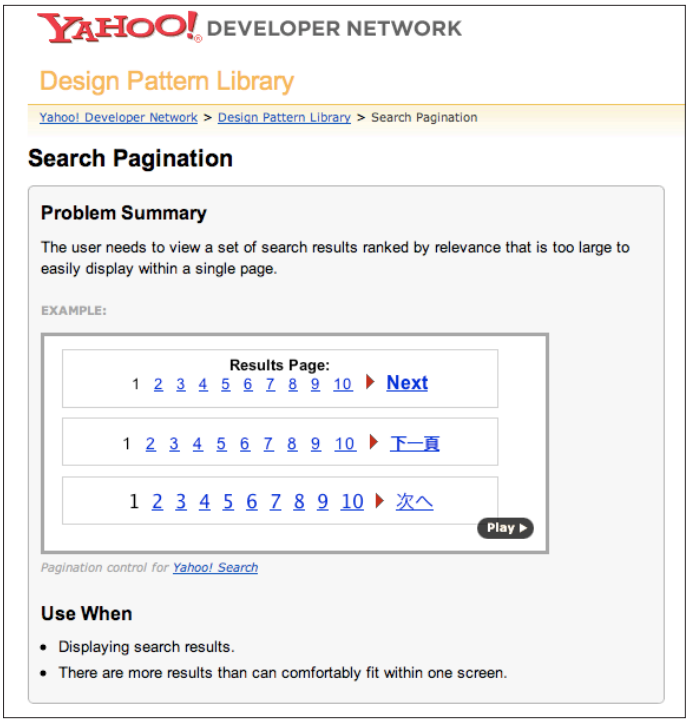

图1-1 雅虎设计模式库[①]定义的一个模式

1.1.2 组件：高效利用可重用的代码

除了模式之外，开发者们还需要一种简便的方式来重用具体的代码。

我们选定了可用的设计模式以后，就需要考虑具体的实现问题了。要想让弹出式日历能顺利工作，屏幕上必须得显示日期。日历必须得响应鼠标的点击。它的外观还得与其他的界面元素保持一致。这些都是组件大显身手的地方。

① 雅虎设计模式库（Yahoo Design Pattern Library）目前已发布了 50 个雅虎认为最优的设计模式，包括分页系统、导航和页面布局等。网址为 http://developer.yahoo.com/ypatterns/。

组件会从像素级别来详细指定设计响应。它们通常以代码的形式来体现，因此组件实际上也体现了具体的交互行为。它们是具备了诸如字体、颜色和布局等样式元素的功能性设计方案，如图 1-2 所示。

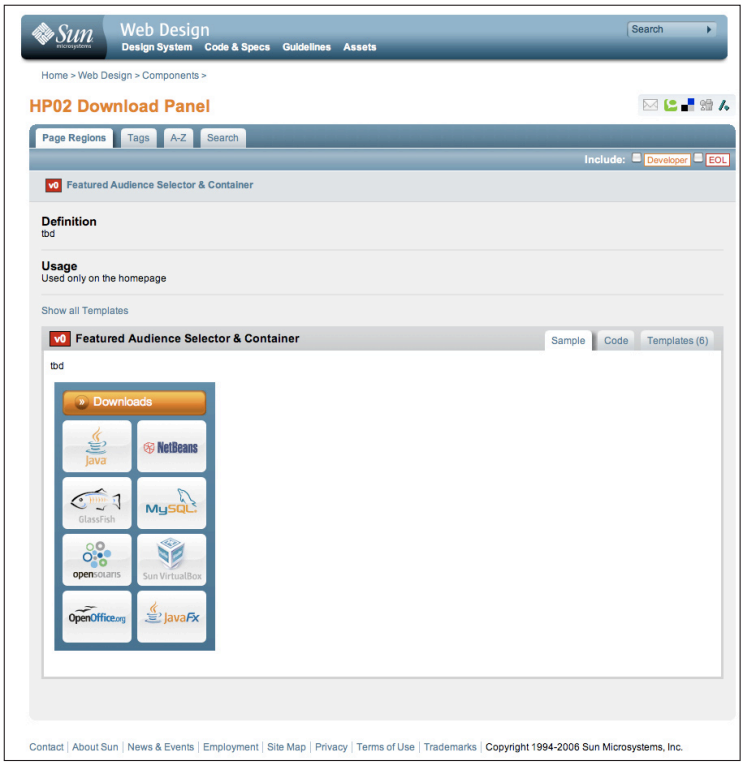

图1-2
Sun.com[①]上某个组件的存档记录

开发人员用组件来拼凑出设计的所有细节。组件构建完成以后，就变成了随时待命的现成元素，能轻易地嵌入到任何新界面中去。这使得每一个环节的开发速度都得到了提高，从早期的原型阶段到最后的部署阶段莫不如此。简而言之，组件就是将设计模式进行完整代码化、模块化后得到的可执行版本。

1.1.3 框架体系：最后的拼图定乾坤

交互设计的框架体系（也就是本书主题）是这个三位一体中的最新成员。如果说设计模式是某个常见问题的通用解决方案，那么交互设计的框架体系则是一

① Sun.com 是 Sun 公司的网站。Sun 这个名字其实来自于 Stanford University Network（斯坦福大学网络）的首字母缩写。该公司推出的著名产品包括 Solaris 操作系统、Java 平台技术及 MySQL 数据库系统等。Sun 在行业中被认为是最具创造性的企业之一，不断尝试新的软件方式和定价模式，但在 2009 年，该公司以 74 亿美金的价格被 Oracle（甲骨文）公司收购。

系列设计模式再加上其他元素和信息，用以指导完整的系统或站点上下文的设计。

和人体一样，每一个 Web 应用都由一系列相互合作的子系统所组成。每一个子系统都包含许多独立的单元，每个单元都具有各自的功能和用途。如果对这些成功的（和不成功的）网站及应用进行详细的解剖和分析，我们不仅能够标识出那些在不同环境下满足用户需求的常用元素，而且能够更加深刻地理解人类行为，从而改善这些标准，同时在不牺牲可用性的前提下将我们的设计提升到新的高度。

图 1-3 为一个框架体系。

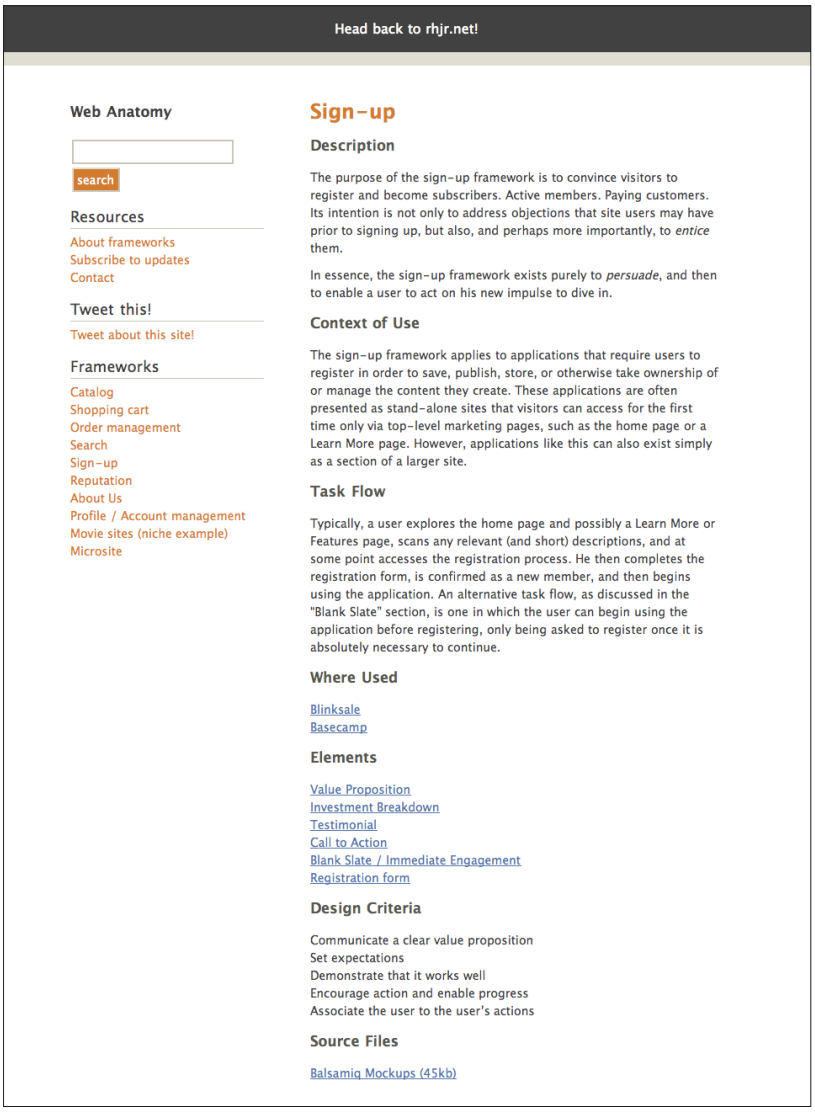

图1-3
一个框架体系的存档文件

换句话说，如果仔细观察那些已然运作良好的案例，并且分析其背后的原因，我们就能同时解决两个问题：其一是明确设计的切入点；其二是了解如何才能创建出更好、更强、更快而且和那些老牌经典一样易于使用的交互设计。

比如，在第 4 章中我们将描述一个注册框架，包含了数个常用于鼓励用户注册的设计元素。其中每一个元素都可以被看作是一个独立的设计模式。例如“电梯陈述”（关于应用程序本身价值的一段简要声明）[①]，它可以让人们快速、高效地了解这个网站或应用能提供什么。然而，尽管“电梯陈述”消除了用户脑海中的疑点，解决了问题，但它本身其实并没有多大意义。事实上它隶属于另一个更大的问题——如何劝服人们在崭新的 Web 应用中注册。

框架体系不去为“狭隘的”问题提供“狭隘的”解决方案，它处理的是较为复杂的问题。它从产品的整体背景出发，为设计确定指导方针。

当一位用户到达某个新网站，犹豫是否注册的时候，注册框架为他做出清楚的介绍、解答疑问、指明起点，并提供注册的方法。没有哪一个单独的设计模式能处理所有这一切。实际上，真正解决问题的是这些元素的组合。进一步说，任何独立的设计模式都不能告诉我们如何满足用户的所有需求，也不能告诉我们为什么需要首先处理这些需求。

为了弥补这一点，框架体系为包含该模式的整个子系统进行了描述。一个注册子系统需要有输入用户 ID 和密码的模式。但它也需要有重新获得密码的模式、建立用户档案的模式、创建新 ID 的模式，以及修改密码的模式。

开发团队通过观察其他设计并提取共性的方式来标识、记录自己的框架体系。这些元素变成了一个清单，列出了一个完整的系统所需要的一切，从而帮助团队确保拥有所有正确的模式，以便开展设计工作。

框架体系是一种高度的抽象。它不涉及具体的品牌化或视觉设计需求——那是组件来完成的事情。相应地，组件又以独立的设计模式为基础。确切地说，框架体系是某种范围更广的结构化系统，从一开始就能帮助设计师对模式进行选择。我们相信，这种和模式、组件相配合的结构化系统，就是 Rolf 所说的“界面模块”。

---

① “电梯陈述”（elevator pitch）是一个风险投资行业的术语，指的是关于产品、服务或项目的摘要性概述。原意是指在和投资家同乘电梯的时间内，创业者用极为简要的方式介绍自己的项目并争取投资。在本书第 5 章和第 7 章都有关于“电梯陈述”的进一步讨论。

1.2 超越常规

除了能有效地加快迭代，产生易用的设计之外，框架体系还能帮助我们更深刻地理解现有标准背后的逻辑依据。通过这种“解剖镜”，设计师可以反向追溯影响当今各种设计决定的最初逻辑原理，然后将自己的理解转化为准则，应用到更加新颖的设计上。我们将在第 2 章更为深入地讨论框架体系的这一方面内容。

1.2.1 分解工作量

需要提到的是，重用带来的好处并不是无偿的。识别可重用的元素需要花费时间和精力，为它们备案、存档也颇为耗时，而且保持资源库的更新也是一项长期而且劳心费力的任务。

但如果把它们分为模式、组件和框架体系三位一体的形式，事情就好办多了。因为这样一来，我们就能让设计师和开发人员对工作进行分配。

组件更接近于最后的实现工作，因此我们通常能让开发人员来管理这个库。而交互式设计框架体系则重点关注用户体验，因此更适合让设计人员来负责。模式库则介于设计和开发两者之间。

小型的公司也许无需耗费大量精力来保持资源库的更新，但是规模较大的企业则需要付出更大的努力。管理者需要鼓励团队成员为资源库标识新的元素（或者直接新增元素），同时还要确保已录入的元素能与时俱进。由于资源库是一种共享的资源，整个团队也应当分担监管的义务。这种劳动分工能够避免让某人独自身负重担，同时也有利于让资源库在团队成员的工作中随时保持活跃，提醒他们可以利用其中的资源。

构建完成以后，这些资源库就能为设计团队提供强有力的援助。它们能为整个设计流程注入活力，加快产品交付的过程，同时为优秀的设计扫清障碍。长期实施可重用策略节省下来的费用无疑要高于最初的开发投入。

以上三种资源库聚集到一起，就组成了我们所谓的可重用铁三角（The Reuse Trinity）。我们会在下一章深入描述这三种资源，并详察三者之间的关系。在此之后，我们会对数个常用框架进行挖掘，以弄清它们的来龙去脉。

1.2.2 古老问题有新解

通过把 Web 视作多个结构化系统，并标识其中与你自己的项目相关的结构

化系统，不仅便于你快速开展设计工作，还有助于积累经验，设计出业界前沿的解决方案：适合用户使用，以低投入实现高产出，缩短项目周期，而且从一开始就确保设计的可用性。

通过框架体系，我们能清楚地看到基于当今标准的设计方针。我们同时也会看到组合更佳用户体验的可能性。但这里却没有什么包治百病的良方。

换句话说，交互设计的框架体系并不深奥。它们只是系统框架而已，是服务于系统及其上下文背景的设计指南，可以无缝地构成完整的解决方案。它们可以（而且应该）进行调整，赋予独特的风格，也能被定制。它们就是一个易用设计的构成模块。而更好的是，它们可以告诉我们如何演变和进化。

1.2.3 问题，答案与灵感

当我们致力于编写本书时，Robert 曾在数次讲座和会议中谈到过交互设计的框架体系，会议期间 Robert 记录了许多听众和与会者提出的问题。我们据此对本书的内容和结构不断地进行修改，以期涵盖并解决这些问题。我们发现，仅仅写一本有关框架体系的参考工具书是远远不够的，更为重要的是能清楚地解释它们为什么能产生如此重大的影响、它们的组成及其原因，在自己的设计项目中使用框架时需要考虑哪些问题、如何标识和共享框架如何备案和记录框架，如何使框架融入设计过程，以及如何能充分利用它们的潜能，从而达到启发创新的目的。

因此，在本书的一开始，我们会近距离地观察各种模式、组件和框架体系，了解它们如何相互协作以形成一个完整的可重用策略。随后我们将察看若干重要的框架体系，揭示它们的历史和效能，讲解如何使用这种崭新的方式来剖析网页，并且深入理解这些框架能够成为标准的原因，以及从中学到的知识和经验。在此之后，我们会演示如何利用框架体系来完成一个项目，同时强调你能进行的调整，以便充分利用这些新的资源。最后，我们会说明在你自己的公司或组织中，应当如何标识并开始使用框架体系。

我们衷心希望，你在读完本书之后能受到鼓舞和激励，从而着手构建一个庞大的、属于你自己的、经过周详测试的界面模块资源库。

2. 可重用铁三角

在本章中，我们将深入探究可重用策略中的三个组成部分，以便在后续的内容中你能更好地理解设计模式、组件和交互设计框架体系三者之间是如何相互关联、相互协作的。

可重用铁三角的诞生并非轻而易举，产生的顺序也绝非符合逻辑。模式的概念最初始于 Christopher Alexander 于 1977 年写的书，其后又被 Luke Wroblewski [①]、Bill Scott [②]、Martijn van Welie [③]、Theresa Neil [④]、Christian

① Luke Wroblewski 是一位界面设计师、战略家及作家。他曾是美国伊利诺斯州大学界面设计专业的老师，后任职 eBay，领导组织了多项网站改进工作。现任雅虎公司产品构思与设计部门高级总监，并创立了 LukeW Interface Designs。著有 *Site-seeing: A Visual Approach to Web Usability*（《Web 可用性的可视化》）、*Web Form Design: Filling in the Blanks*（《Web 表单设计》）等。个人网站是 http://www.lukew.com。

② Bill Scott 有着长达 25 年的从业经验，在设计和开发领域均有建树。他曾在雅虎任 Ajax 技术推广专家一职，并负责管理雅虎设计模式库，现任 Netflix 公司的用户界面工程总监。与 Theresa Neil 合著有 *Designing Web Interfaces: Principles and Patterns for Rich Interactions*（《Web 界面设计》）一书。

③ Martijn van Welie 现任飞利浦高级交互设计顾问一职。写了很多用户界面设计模式方面的文章。其个人网站是 http://www.welie.com。

④ Theresa Neil 是一位用户体验咨询师。她曾与 Bill Scott 共事，创建了 Sabre 用户体验团队。自 2001 年起，她为各种新技术及财富 500 强公司组织设计了 80 多个网站、桌面程序和手机应用，客户包括美国公共广播公司、PayPal 及 Pervasive。她也是《Web 界面设计》的作者之一。其个人网站是 http://www.theresaneil.com。

Crumlish[①]、Jenifer Tidwell[②]等模式倡导者和众多业界专家所普及和发扬光大。正是在他们的努力下，模式这一概念目前已被推广到了 Web 设计实践的前沿。而组件——表示模式自然演变的完整的、生产就绪的页面元素——则出现得颇晚，至少在软件设计领域是如此（与开发相比）。

事实上，尽管有关组件的想法雏形已经酝酿了数年的时间，但它才刚开始作为一种概念被标准化。这要特别感谢来自 EightShapes[③]的两位设计师 Nathan Curtis（著有《模块化网页设计：为用户体验设计和存档而创建可重用的组件》一书，New Riders 出版社）和 Dan Brown（著有《设计沟通十器》一书，New Riders 出版社）所付出的努力。而框架体系则是拼图的最后一块，在你手中这本书里首次以正规文档的形式记录下来。然而，在实际工作中，我们应该首先考虑框架体系，模式其次，最后是组件。如图 2-1 所示，这也是它们在网页设计过程中最为有效的使用和思考顺序。

设计模式实际上就是隶属于大型框架体系的生产模式。而组件则是针对具体某个系统对模式进行实现后的产物。它将模式具象化为网站或应用界面中的一个部件，使之能够通过网络被直接交互。

接下来我们会按照构思这些元素的顺序逐一进行深入讨论。无论你对模式和组件是否熟悉，预先了解这些框架体系的基本组成部分，都将有助于我们建立起正确的大局观。

---

① Christian Crumlish 是一位极其多产的作家和信息设计师。他早在 1994 年就开始参与交互分析和设计的工作，是 Adaptive Path 的总裁及创始人之一，如今主要负责雅虎设计模式库。他著作等身，自 1990 年起，出版了包括畅销书 *Internet for Busy People*（《为忙碌人士讲解 Internet》）、*The Power Of Many: How The Living Web Is Transforming Politics, Business, And Everyday Life*（《大众的力量：网站如何改变政治、经济和日常生活》）、*Designing Social Interface: Principles, Patterns, and Practices for Improving the User Experience*（《社会化网站界面设计：提升用户体验的原则、模式及实践》）（与 Erin Malone 合著）等近 30 本著作。其个人网站是 http://mediajunkie.com。

② Jenifer Tidwell 目前在技术计算软件厂商 MathWorks 公司任交互设计师及软件工程师一职。她从 1997 年开始研究用户界面模式，同时也是一名 RIA（Rich Internet Application）技术的倡导者。著有 *Designing Interfaces: Patterns for Effective Interaction Design*（《设计界面：有效的交互设计模式》）一书。其个人网站是 http://jtidwell.net。

③ EightShapes 是 Nathan Curtis 和 Dan Brown 共同创立的用户体验设计公司，总部在华盛顿。该公司提供计划、研究、信息架构、交互设计、用户测试等多方面的服务，同时为客户创造既能满足用户需求，又能在商业目标和技术可行性上取得平衡的解决方案。公司网站是 http://eightshapes.com。

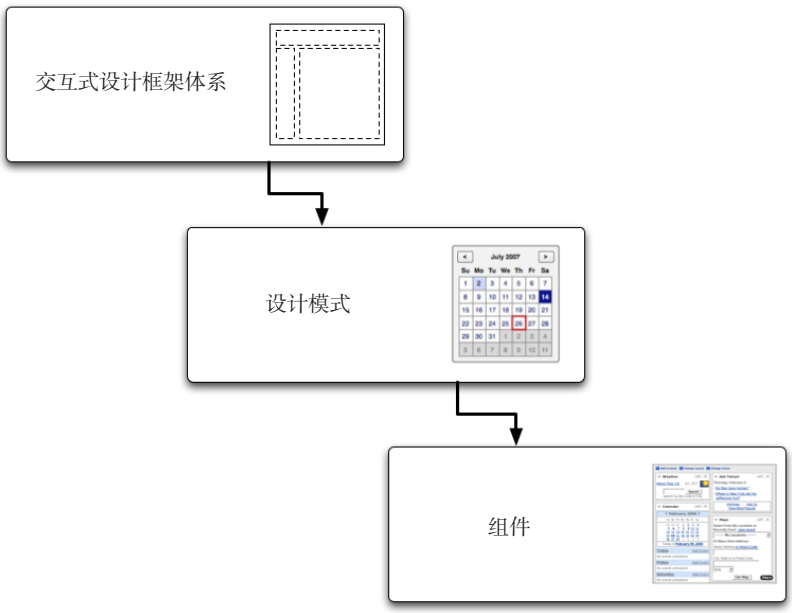

图2-1
可重用铁三角

2.1 设计模式

设计模式其实是一种针对某个常见问题的常用解决方案。比如，分页模式就为我们呈现了一种标准的交互接口，它将搜索结果以多个页面显示，同时在每个结果页的底部加上指向不同页面的链接。这种设计通常会包含 Previous（上一页）和 Next（下一页）按钮、通往其他各个结果页的数字序号链接以及一个指示当前页面的视觉标志，如图 2-2 所示。

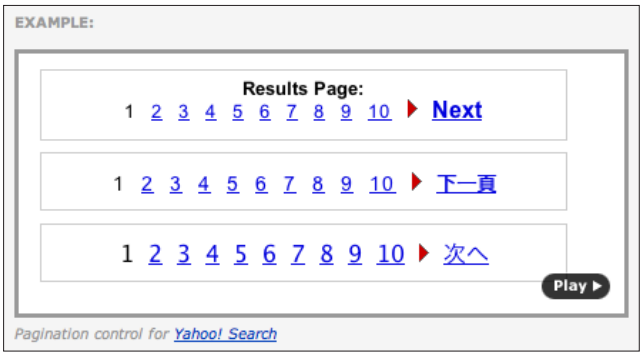

图2-2
来自雅虎设计模式库的分页界面示例

似乎很眼熟？本应如此。在每一个雅虎搜索结果页面的底部，你都能见到它。

也许雅虎并不是第一个使用这种设计的网站，但是雅虎的版本被使用的频率如此之高，其他几乎所有的搜索系统都或多或少沿用了它。这使它变成了一种模式——取得了巨大成功的模式。实际上，它已经被不计其数的模式资源库（无论是公共库还是私有库）所收录和引用。

设计模式所带来的首要好处就是，用户能够将自己在某一个网站上的体验转化为通用的操作经验，运用到其他所有使用相同模式的网站中去。在雅虎上进行了数次搜索之后，用户就能很轻松地理解其他任何使用了相似设计的分页界面，不管是在哪个网站。

而另一方面，设计师们从中也获益匪浅。因为模式能够为大量典型的设计问题提供“罐装”的解决方案。不必为每一个新网站都思考和创造新的搜索导航系统，我们只需要拿出分页模式，做一些小小的调整就可以了。

数年来，许多成熟的设计模式已经成为互联网中的模块，这无疑是明智之举。而框架体系的构建前提就是成功的模式。

2.1.1 设计模式六要素

要想更好地理解模式，我们不妨看看在一份标准的设计模式描述文档中都包含些什么，如图 2-3 所示。Jared 的公司 UIE 列出了以下六要素。

1. 模式名称

如果我们正在讨论一个元素，它使用户能进入到网站受密码保护的区域，那么我们也许会称呼它为“用户名和密码控件”、“两行式登录元素”或者“登录元素”。

模式名称的选择务必要小心谨慎。在此之前，设计中出现了太多的无名元素，以至于在讨论中经常会有类似“那些我们常常放在左边的小方块”这样的说法。而之所以有模式名称，其目的就是为了促进清晰的交流和沟通，这样在会议、设计文档或者其他地方我们就能明确地称呼某个具体元素。

我们发现，为模式命名需要技巧、创造力以及一点点运气。开发团队往往在一开始为某个模式起了名字，过了一阵子发现大家经常使用的却是另一个名字。

比如，某个开发团队将他们的应用程序的对象属性编辑器正式定名为“Infobox”（信息框），之后却发现团队里根本没人这么说。所有人都叫它“Properties”（属性）。

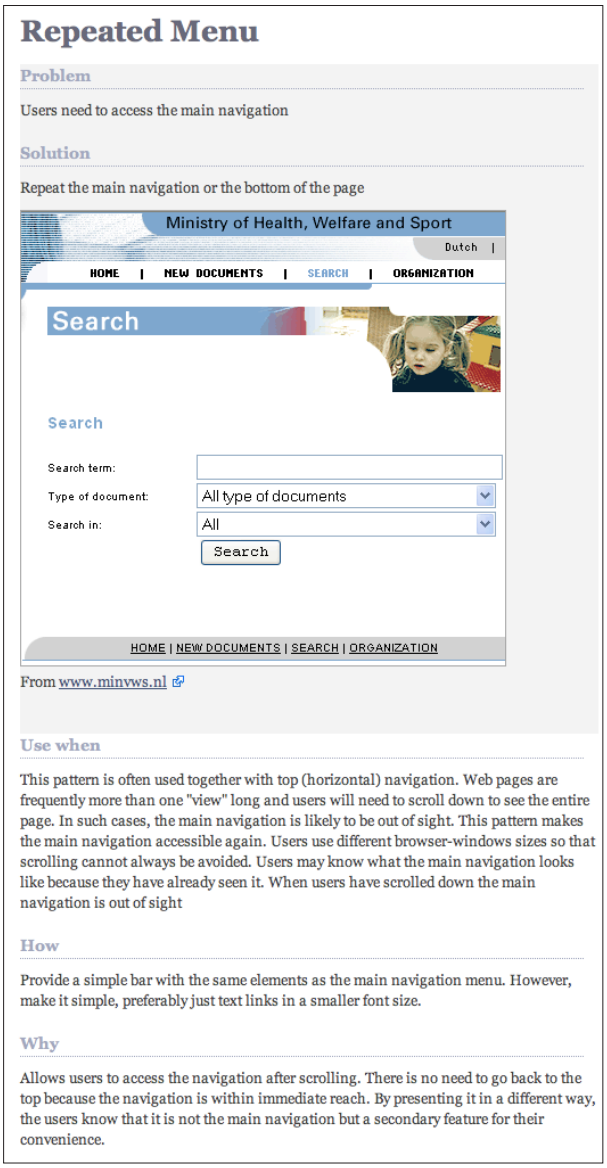

图2-3 来自Welie.com公用模式资源库[①]的一份模式文档

有些开发团队在模式的描述中为其添加了一系列的昵称、同义词以及代称（also-known-as，简称 aka）。这能帮助团队成员确定他们讨论的对象。我们对事物的称呼常常会随着时间而改变，因此如果模式名称能和当前用词随时保持对应和更新，会带来很大的好处。

① Welie.com 是 Martijn van Welie（见本章第 2 段）的个人网站。Martijn 在网站上公布了自己所整理的满足用户需求、程序需求和上下文设计的三大模式资源库，共计 120 余个模式。

2. 描述

描述对于一个好的模式来说至关重要。通过描述，那些对该元素不太熟悉的团队成员就能准确地理解大家正在讨论的内容。

由于一图胜千言，界面截图也非常有价值。如果某个模式在同一个网站中有多种表现形式，那么各来一张截图会有极大帮助。

比如，一个登录元素可能会有如下描述（伴随着合适的界面截图）：

> 一个两行的表单元素，用于采集用户的 ID 和密码，从而使他们能够进入网站内受密码保护的区域。

描述无需像文学作品那样精雕细琢，但它应当包含足够的信息来解释该元素存在的理由，并说明如何将它和网站上的其他元素进行区分。

3. 上下文情境

与一般的设计指南或样式参考文档相比，设计模式的主要优点之一在于它强调了每一种模式所使用的模式库中的上下文情境。在构思新设计的时候，设计师们可以利用上下文描述来确定该模式是否运用得当。

例如我们的登录元素，有关它的上下文情境可能会是如下描述：

> 无论何时，只要网站中的某位用户希望从公有区域转向访问私密信息，我们将使用登录元素。在面向公众的页面中，只要有足够 155 像素 ×210 像素的空间，就可以显示该模式。

当然，在这里还需要包括在不使用登录元素时的描述：

> 如果在某些面向公众的页面中，垂直方向无法提供足够的空间，我们将在页首的 banner 横幅处使用单行的登录元素。或者在网站受密码保护的区域中，不使用登录元素。

上下文情境是不断变化的。当开发团队加入了更多元素，开发新的应用程序，发现新的用户需求时，都需要对“上下文情境”一项进行频繁的更新。理想的情况是，在某个模式生命周期的任何一个阶段，设计师都能通过阅读此项而迅速了解该元素是否适用于手头的工作。

4. 曾于何处使用

“曾于何处使用”是模式文档中另一个不断变化的部分，它列出了那些使用过这一模式的实例。模式每一次将其转化为生产系统时，都应当对此项进行更新。开发团队成员可以查看已经实现出来的成品，了解某个模式的运转情况。

5. 工作方式

开发团队在这里将描述该元素技术层面的内容：

> 用户在标记有 User Name 的输入框中键入他们的用户 ID，在标记有 Password 的输入框中键入密码（密码内容会被遮盖）。如果他们愿意，可以点击 Remember Me 复选框，以便在重复访问时系统能预先为其填写 User Name 输入框。当就绪后，用户点击标记有 Log in 的按钮。如果用户名和密码有效，则显示该用户的个人页面。如果无效，则显示错误页面（参见“登录错误”模式）。

需要的细节数量取决于控件的复杂级别，以及团队成员对它的熟悉程度（如果是他们自己经常使用的元素，就不需要像不常见元素那样进行详尽的描述）。曾有一个可用性团队向我们展示了利用视频捕捉来创建演示短片，他们通过这种方式来描述元素的运作机能。

与该元素产生交互的其他模式也会提及，此举能帮助设计师更为全面地考虑问题，便于在最后对设计进行整合。

6. 其他必备模式

很少有能完全独立存在的模式。一个模式的出现，通常都意味着设计师还需要考虑其他模式来支持它。

比如说，如果一个设计需要“登录元素”模式，那么它很可能还需要下面这些：

- 创建新用户 ID 的模式；
- 修改密码的模式；
- 重新获得密码的模式；
- 从网站的受密码保护区域退出的模式；
- 当输入的用户名或密码不正确时，显示错误信息的模式。

所有这些模式都会列在“必备模式”项中，并附有它们为什么“必备”的相关解释（如果不是很明显的话）。

设计模式的文档中还可以包括竞争性举措、模式历史、可用性测试结果、用户反馈和讨论记录，等等。

2.1.2 模式资源库

如图 2-4 所示，模式资源库就是模式文档经过整理、分类后的集合，通常在网上或者企业内部进行公开。

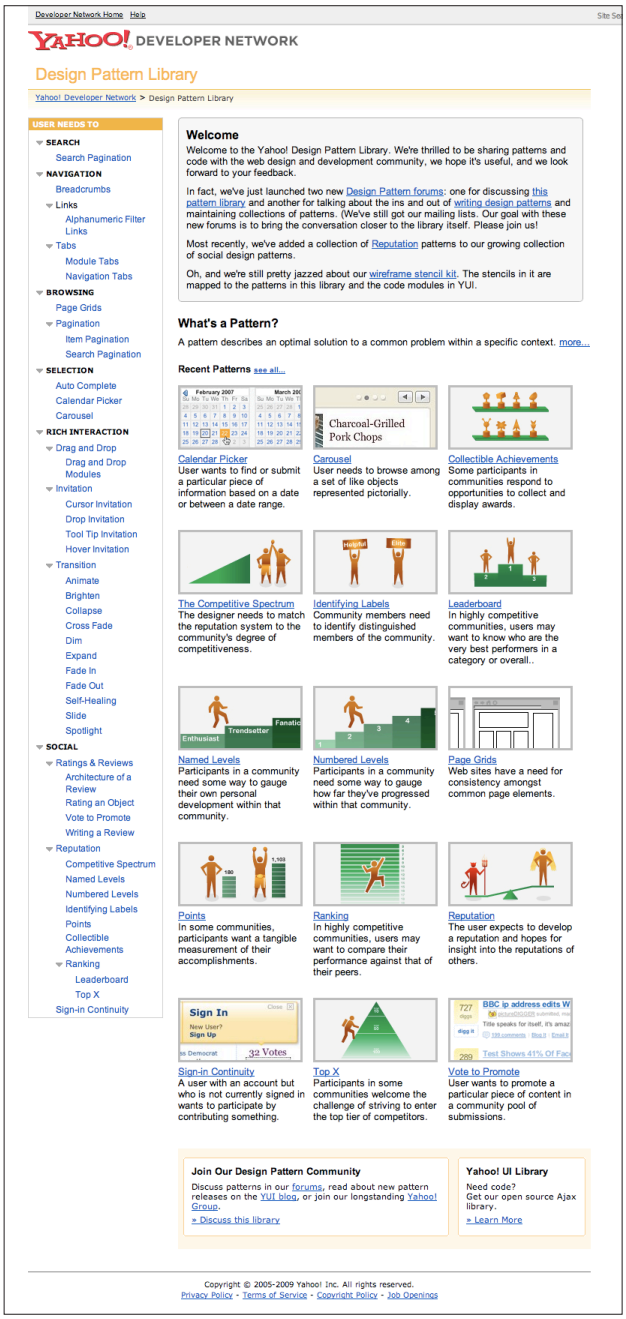

图2-4
雅虎设计模式库是最流行的公用资源库之一

以下是一些公用的模式资源库。

- 雅虎设计模式库：http://developer.yahoo.com/ypatterns；
- Designing Interfaces（是同名图书的资源网站，Jennifer Tidwell 著，O'Reilly 出版社）：http://designinginterfaces.com；
- Welie.com：http://www.welie.com。

在准备使用模式时，许多开发团队都会偏向这些现成的、公用的模式资源库。然而，尽管这些资源库的资料通常都很齐全，而且免费，但它们却无法说明项目中特定的技术限制和业务需求。因此对于具体的项目来说，公用模式库的用处可能并不大。最有用的模式库应该重点关注项目特定的规范和需求。

公用的模式资源库倾向于提供通用的模式，而不会针对某个具体的应用。它们为那些公认的标准 Web 交互提供了低级别的基础性建议，而企业往往会根据自己的应用或网站对这些模式进行“定制”，把过于通用的公用模式转化为真正适用于专属设计团队的模式。

尽管如此，这些通用的模式仍然能为我们带来极大的好处。除了能提供基础性建议，为“定制”模式提供一个高起点之外，这种模式库对于“单干”的设计师来说无疑是一种绝佳的资源，不管在企业内工作还是在外面做顾问都是如此。顾问常常会因为客户不同而接触到各式各样的项目，而公用模式资源库能够为其提供适用于大多数网站的无尽资源。

内部模式资源库中的模式则与此不同。它们具有更强的针对性，只对应企业内部的网站，是设计团队手中威力强大的工具。团队可以在内联网中利用 wiki[①]或者微型网站来组建资源库，使其贴近自身的设计风格，这样企业中的其他团队在实现新设计的时候就能直接从中取用。

构建一个模式资源库的主要困难在于协调运作。首先，个人或团队需要从自己的所有案例中翻找、标识出已经在使用的模式，其次他们还需要建立稳固的公布和共享机制，为每一种模式存档备案、为资源库制订长期的维护计划，最后还得为其进行宣传和推广。这可不是什么微不足道的小任务，而且很可能还会经常遭遇到其他更为紧迫、优先级更高、更直接面对客户的项目的排挤。

① wiki 一词来源于夏威夷语 wee kee wee kee，意思是“快”。它其实是一种在网络上开放、可供多人协同创作的超文本系统，由 wiki 之父 Ward Cunningham 于 1995 年所创。基本上，wiki 包含一套能简易制作、修改 HTML 网页的系统，再加上一套记录和编排所有改变的系统，并且提供还原改变的功能。wiki 网站允许任何造访者轻松地添加、删除、编辑所有的内容，而且通常都不用登录，因此特别适合团队合作的写作方式。

不过，模式库构建成功以后，它带来的回报将远远超过构建时所付出的努力。如果对某个交互的外观或功能有疑问，开发团队可以立刻派开发人员来回答问题。而且模式是可重用策略的核心板块，随着它的就位，设计师们就能避免在已经非常普遍的方案上浪费时间，而把更多的精力用来迎接项目中那些更为独特的挑战。

2.2 组件

要想介绍组件，也许最简单的办法就是直接引用其倡导者、EightShapes公司网站上的解释（参见 http://unify.eightshapes.com/users-guide/what-you-get/wireframe-components/）：

> 组件是页面设计的一部分。
>
> 组件由通用的最基层元素（例如文本、链接、按钮、复选框和图片）相组合而成，它们是在页面的界面设计中可以使用（或重复使用）的有意义的组成单元。其他描述这种页面内集合的常用术语包括模块、元件、控件，甚至是分子。

而在 Nathan Curtis[①]的一次演讲“Creating a Component Library”（创造组件资源库）中，其释义也大致类似：

> 鉴于我们通常都以一个完整页面或页面状态为单位来观察，同时把页面上无法继续划分的部件（例如一个 logo、页首图片或者按钮）称为“元素”——我们可以说，组件是由元素相互组合而成的具有明确目的、可重用的独立结构。标签式导航条、搜索结果、文章内容都是组件。

模式一般都能在各网站间通用，而组件则只对应特定的某一个网站，非常具体。模式适合于交互设计师和其他运用草图、线框图或其他基础手段进行构思的人，而组件则专用于那些负责构建这些设计的人。它们的代码完整，能够重复使用，而且一个模式可以派生出多个组件。开发人员只需要把某个组件直接插入到页面中，定制其中的内容，就可以得到一个完整的页面区域。

2.2.1 组件六要素

组件资源库目前还不是十分流行，因此对现有资源的调查和最佳案例的推

① Nathan Curtis 是 EightShapes 的创始人之一（见本章第 3 段）。目前他已经创建了一个基于组件的模块化体系，面向规模各异的开发团队，客户包括 Comcast、Sun、Discovery、Cisco、eBay 等。个人博客是 http://nathancurtis.com。

测尚不足以得出完善的结论。我们暂时以 Sun 公司网站中提供的公用组件资源库为例，列出以下需要包括的六要素（更多内容参见 2.2.2 节）。

1．组件名称

在 Sun 的资源库中，组件名称都是以页面标题的形式来展现的。不过在模式资源库里，也可以包含一个更为精确的“组件名称”项，提供一系列可供替换的名称。

2．组件版本号

组件的版本号与组件名称和示例（参见后文描述）的标题密切相关。和软件更新时的发布版本说明一样，版本号可以翔实地记载从一个版本到下一个版本中发生的变化。如果需要对之前实现的内容进行更新，版本号能引起开发人员的注意，而且确保开发团队能维持系统的一致性和连贯性。

3．定义

类似于模式里面的“描述”项，组件的“定义”会描述组件的目的和用途。在图 2-5 中，组件 B01 Features 的定义如下：

> 首页专题是 sun.com 网站首页中一个较为复杂而又重要的部分。它可以看作是首页推介元素的容器，循环显示首页推介的 3 个专题内容。

4．使用方法

“使用方法”项描述了组件应于何处使用，并包含相关信息。在 Sun 的“D05 Primary Index Nav”（基本索引导航）页面中，其使用方法如下：

> 在任何索引页上使用。如果没有额外内容，可以选择是否加入 See All（查看所有）链接。

该组件被限制只能在索引页上使用，同时可以从两种方式中选择一种来实现：带 See All 链接或者不带。使用方法项注明了这两点内容。

5．示例

示例为我们提供了一个鲜活、生动的组件实例。组件是经过实现后的页面元素，因此在基于网页的文档中，你完全可以添加那些带有完整功能的版本。通过实际的示例，团队内部的所有人都能直观地了解该组件的工作方式（它还能协助质保小组审核已实现版本的正确性）、外观，以及要实现它应使用何种代码。

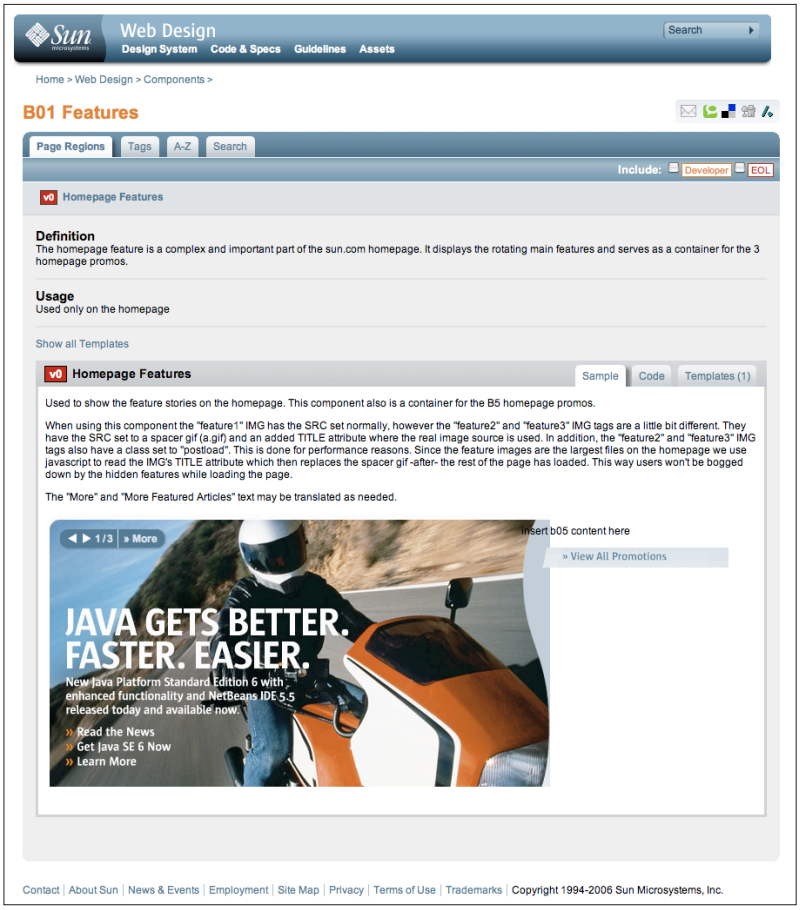

图2-5
Sun.com的一份组件文档

6．代码

除了一个能运行的示例之外，组件文档中还应该包含一个“代码”项，链接到该组件的已实现版本（包括使用不同编程语言的版本）。如果某个开发团队用Ruby on Rails[①]开发了一个应用程序，用苹果的Cocoa[②]开发了另一个，而两者都可以使用这个组件，那么它的代码项就应当包括用这两种语言创建的版本。

① Ruby on Rails简称为RoR或Rails，是一个用Ruby语言写的、严格按照MVC结构开发的开源网络应用框架。Ruby on Rails由David Heinemeier Hanson从37 Signals公司的项目管理工具BaseCamp里面分离出来，并于2004年7月以开源方式发布。由于开发人员可以专注于应用程序自身的设计，省去那些花在了解及配置基础框架上面的时间，Ruby on Rails在发布之后短时间内就迅速受到很多开发人员的欢迎。

② Cocoa是苹果公司为Mac OS X操作系统所创建的面向对象的编程环境，与Carbon和Java处于同一层。它从20世纪80年代NeXT开发的编程环境NeXTSTEP和OPENSTEP演变而来，包含一组面对对象的软件库以及一个运行环境，而且它还与其他的应用程序环境共用一个集成开发环境。

2.2.2 组件资源库

组件资源库可以用和模式库同样的方式来构建和发布——建立 wiki，标识已完成的组件实例，然后为其存档备案。

如图 2-6 所示，Sun 公司的网站提供了一个公用的组件资源库，可以作为组件如何存档的参考。你能在 www.sun.com/webdesign 上找到这个资源库。

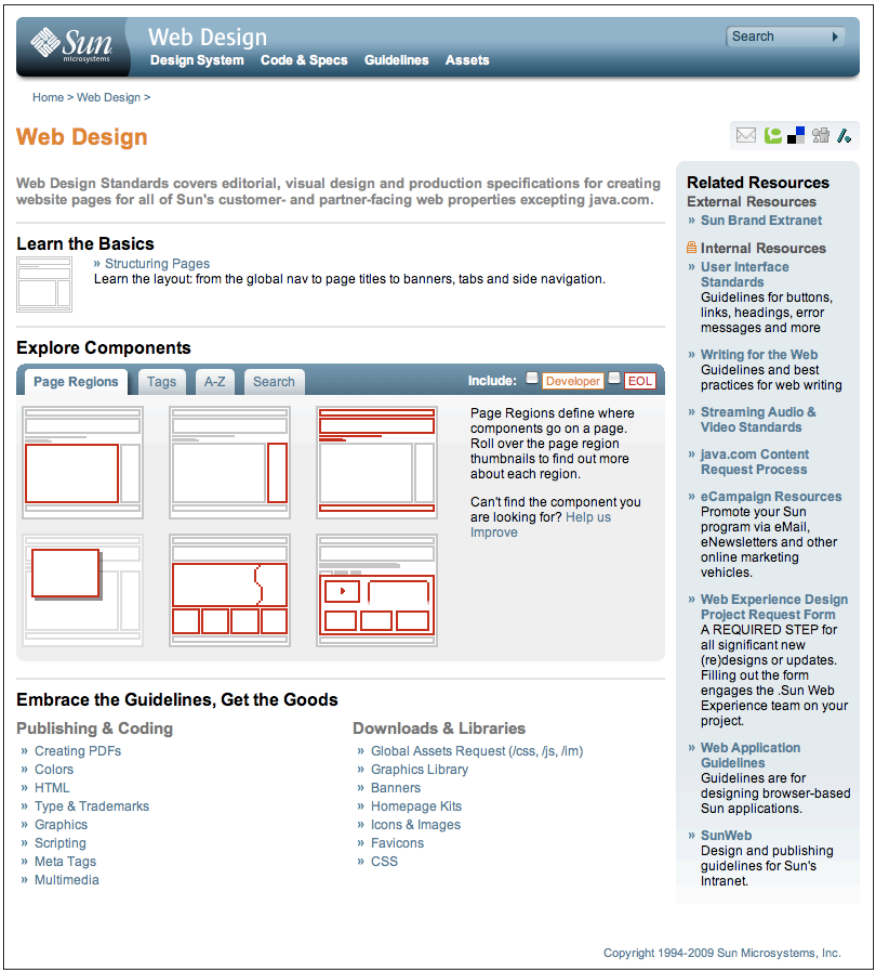

图2-6 Sun.com组件资源库的首页，网址是www.sun.com/webdesign

而最有帮助的是，Sun 还提供了一个有关如何使用组件的页面（参见 http://www.sun.com/webdesign/structuring-pages.html），以一个标准的 Sun 内容页面模板为例，对其中的各个区域进行了直观的描述。同时它还提供了一个用这些组件构建的页面示例（参见 http://www.sun.com/software/products/mysql/index.jsp）。

此外，Nathan Curtis 也在 SlideShare.net[①]上发布了他的“Creating a Component Library”演讲稿。网址是 http://www.slideshare.net/nathanacurtis/creating-a-component-library-298610。

2.3 交互设计框架体系

在互联网上可能存在着数百万个设计模式，它们每天都在被人使用，其中一些很流行。就此而言，它们似乎就是所谓的界面模块的理想“后备”，但其实模式仍然存在着一些无法摆脱的局限。

设计模式能够解决那些范围较小的具体问题（这也正是我们期待的结果），但是我们无法从中得知这些问题之间是否有联系、是否会顾此失彼。尽管在一般的模式描述文档中有一个上下文情境项，但它也只是讨论了模式能在何处适用，其中既没有提到该模式会如何影响整个应用程序，也没有提到它和系统其余部分的联系。而在必备条件项中，也只是列出了与该模式密切相关的其他模式，并没有揭示其表面下的深层原因。

我们希望创造出成功的设计，然而设计模式缺乏足够的上下文信息，为我们留下了许多悬而未决的问题。每个页面应该显示多少条搜索结果？应该如何处理错误（例如用户输入错误或者零结果搜索）？用户怎样才能开始新搜索？每条结果应该提供多少信息？哪些信息更为重要？单凭模式是无法回答这些问题的。

比如，分页界面能让用户从一个页面前往另一个页面，然后还能再回来。这的确解决了一个问题，为一个操作提供了支持。然而要想追求可用的设计，还有很多其他问题需要考虑。

要想回答这些问题，我们不能只着眼于分页界面，而必须考虑整个搜索系统。换言之，我们不能只依赖于设计模式，而必须了解模式是如何构成系统，而系统又是如何构成整个应用的。

那么组件呢？从本质上来说，它们也只是某个公司（或组织）定义的模式的代码完成版本而已，因此也同样爱莫能助。

要想回答这些问题，我们需要审视可重用铁三角中凌驾于设计模式之上的更高层次：交互设计的框架体系。

---

① SlideShare.net 是一个发布、分享演讲稿或幻灯片的网络社区，有些类似于 YouTube。用户能在其中上传或下载幻灯片或 PDF 文档，为幻灯片插入同步的声音，还能将幻灯片嵌入到博客、个人网站以及类似 LinkedIn、Facebook 和 twitter 等社交网站中。

2.3.1 交互设计框架体系包含的元素

框架体系的结构与设计模式类似。实际上，我们把框架体系视为设计模式在逻辑上的进化结果，它是我们构建一系列成熟解决方案的下一个必经步骤。本书所呈现的框架体系就是我们推荐大家使用的框架体系，包括名称、描述、在何处使用它们以及如何使用，等等。简而言之，每一个框架体系都会包含本节所述的部分或者全部内容，如图 2-7 所示。

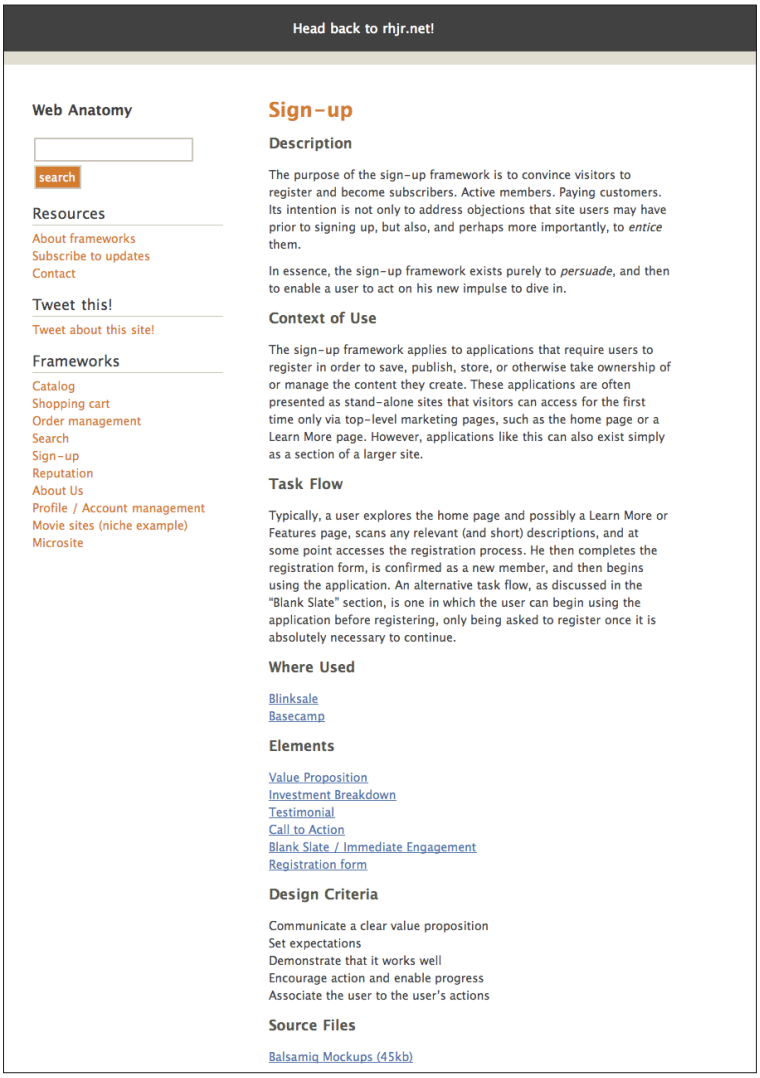

图2-7 来自http://web-anatomy.rhjr.net/signup[1]的注册框架文档

① 这是本书作者 Robert Hoekman Jr. 在他的个人网站（http://rhjr.net）上为本书建立的资源网站，列出了书中涉及的所有 5 个框架范例。

除此之外，我们还将讨论这些框架体系所能满足的用户需求，以及相关的人类行为，同时还会探索解决这些设计问题的各种途径。此举是为了让你能深刻地理解在你自己的项目中应用框架体系时应当考虑什么，以及如何充分利用从中归纳出的设计标准。

当然，你也没有必要为自己的每一个框架体系都总结出一篇见解深刻的论文。实际上，框架体系文档的复杂程度和模式文档相当即可。

只要遵循以下各项就可以了。

1. 描述

描述项不仅描述了框架本身（这是其首要目的），同时也描述了框架应满足的需求。例如在下一章将介绍的目录框架，它描述了人们在选择目标时必经的三个步骤，而这一事实也正是目录框架存在的原因，因为它对该行为提供了直接的支持。

在本书每个框架的描述项中，我们都尽力囊括这类信息。但在你构建自己的框架资源库时，仅仅照本宣科是不够的，必须加以变化。关键是要了解你希望框架做什么。目录框架支持的是人们选择对象的过程，搜索框架支持的是导航未能提供帮助时寻找内容的过程，注册框架则鼓励用户从访问者转变为客户。

2. 上下文情境

该项描述了在使用给定框架时用户可能遇到的问题，或者他们希望满足的需求。例如，当某个用户面对一个必须登录才能使用的 Web 应用（就好像一个四周都是围墙的花园）时，可能会考虑是否值得去注册。而注册框架将会处理此时用户心中产生的各种疑问。它会介绍该应用的功能和优点，注册时须付出的努力或成本，以及如何开始注册，等等。

除此之外，该项还会描述框架在网站信息架构中所处的位置。例如注册框架经常会在顶级的市场营销页面中出现。

3. 任务流程

许多框架通常都由一系列按顺序排列的交互构成。其中一些还会提供必要的信息，用户只要参考这些信息就能自己解决产生的疑问。在这种情况下，用户必须遵守特定的任务流程。任务流程部分所描述的正是这些内容。

当然也有例外，例如那些针对特定主题的网站（比如电影网站，参见第 7 章），或者其他根据惯例相结合，提供完备解决方案的模式集合。一个典型的电影网站包括一个预告片视频、演职员信息、剧情梗概以及其他内容。这种网站本身通常不会包含任务（至少用户访问这类网站不是为了完成工作任务），因此任务流程部分对于电影网站的框架来说并不重要，因此也不会被包括进来。但是框架是仍然存在的，因为这些元素相互协作，共同实现了电影网站的目标，也就是劝说访问者来看这部电影，所以它们构成了框架。

4．其他必备框架

其他必备框架一项列出了与当前描述的框架配合使用时不可或缺的其他框架。例如，一个电子商务框架可能由网站中特定于产品销售与服务的元素构成（出于本书的目的，我们将它与目录框架进行了整合）。然而，为了更好地支持该过程中的任务，电子商务框架还需要其他框架的配合：搜索框架，提供寻找商品的第二渠道；购物车框架，用于处理购买信息；付款框架，用于付款结算。这些都是电子商务框架的其他必备框架。

5．相关框架

相关框架指的是那些与当前描述的框架有着相似的目的或者支持相似的用户或业务目标的框架。除了搜索框架和购物车框架之外，一个电子商务网站通常还会包括一个订单管理系统和顾客账户系统。这些系统可以作为框架进行存档和备案，然后在相关框架项中进行相互链接。

6．构成元素

构成元素项是一个框架中最重要的两个部分之一（另一个是设计标准项，参见下文），因为它列出了所有从属于该框架的设计模式。在后面五章中你将会发现，构成元素是框架体系的主要组成部分，该项中列出的所有模式都会指向各自的详细文档。正是通过这种方式，框架资源库包裹在模式资源库之外，与模式资源库紧密地结合在一起（我们将在 2.3.3 节中进行深入讨论）。虽然我们说是框架体系（而不是模式）构筑了整个上下文情境，为最终的解决方案指明了方向，但它们依然主要由模式构成。

没有模式的框架是不存在的。二者相辅相成、紧密协作。

7．设计标准

框架体系文档中的另一个重要元素是设计标准项，它列出了框架中一系列

设计的导向性方针。这听上去也许简单，但它可能是整个框架中最难以准确界定的部分。

设计标准准确地标识出为了网站用户而必须实现的目标，形成一系列如规章制度般的指示，表达了设计背后的动机。

例如，注册框架（参见第 5 章）包含以下设计标准：

- 表达了明确的价值声明；
- 建立起用户的预期；
- 证明本应用程序运转良好；
- 鼓励行动并确保获得进展；
- 让用户与其操作相联系。

每一条标准都是针对设计的指令。

之所以要列出设计标准，有两个很重要的原因。

首先，框架体系能让我们逆向设计人类的行为，而这种理解正好能通过种种设计标准表达出来。定义设计标准能帮助我们更加深刻地理解框架的价值，也只有这样，你在自己的项目中应用框架时才能做到有的放矢。在设计中，哪怕是最细微、看上去最无关紧要的细节，也可能会导致产品在可用性、客户转化率、操作愉悦性甚至产品价值等方面产生重大的差异。标准能让我们知道在一个设计中哪些地方是最重要的，这能确保我们对设计进行正确的修改和调整。

其次，框架体系能将人类行为映射成一系列具体的目标，其中的每一个目标都有可能激发我们的灵感，创造出新颖的解决方案以达到同样的效果。而以此为基础的创新又能实现整个框架真正的目标，解决真正的问题。简而言之，框架体系的设计标准能告诉你需要实现哪些目标，至于实现的方式则完全可以自由选择。

本书共涉及了 5 份框架文档，我们将以每份文档的设计标准项为例，演示上述内容的实现过程——如何运用与众不同、独一无二的方式来满足某个标准框架的既定目标。我们希望这不仅能帮助你理解如何构思和应用设计标准，还能启发你设计出充满新意的解决方案。

既然设计标准这么重要，那么另一个问题也随之出现：这些标准的产生和推断过程似乎非常复杂，令人费解。但我们并不这么认为。

在为设计制订标准的过程中，唯一的窍门就是自问这个框架能为你或者网

站的其他用户带来什么。

比如，要想知道注册框架为我们带来了什么，先浏览一下该框架中包含的元素。其中一个元素就是**价值声明**（value proposition）——几乎每一个要求用户注册的网站上都会出现。如图 2-8 所示，有关估价和结算的 Web 应用 Ballpark[①]（www.getballpark.com）就在网站中使用了让您能更方便地发送预算单和结算单”这样一条语句。那么这段陈述为我们带来了什么呢？很明显，它的目的是为了表现该 Web 应用的价值，告诉用户这个应用有什么用，以及它为什么很重要。在这个例子中，框架中的某个元素和某条设计标准包含了同一个词。该元素被称为价值声明，而设计标准则能明确地表达价值声明。换句话说，有时候元素仅仅只是为了满足某条设计标准而存在的。

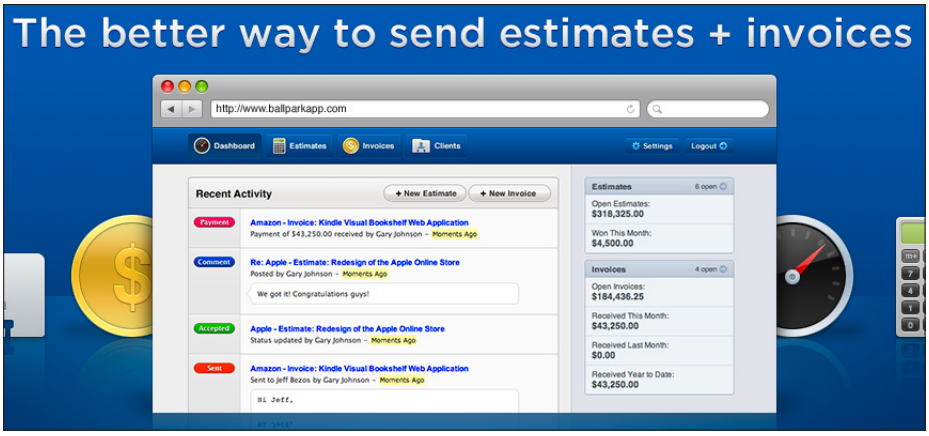

图2-8 GetBallpark.com 上的价值声明元素

而另一个注册元素（将在第 5 章进行讨论），就是注册表单本身。注册表单能让用户创建账户，这非常明显。但是，从这一事实中我们能推断出什么样的设计标准呢？要想解决这个问题，请先自问用户为什么必须注册。用户必须注册，只有这样才将让他们在网站上执行的操作和他们自身“绑”在一起，使他们在下次登录时能够找到之前的数据、进行个性化的定制，等等。那么该框架的又一条设计标准出现了，那就是“让用户和他们的操作相联系”。

而搜索框架（参见第 4 章）则可能是一个较为复杂的例子，因为框架中的元素并没有明显地表现出它们各自的用途。如果要为一个有效的搜索系统设计设立标准，你首先必须了解人们搜索的原因。有时候，网站的导航无法呈现给

① Ballpark 是一个网络在线应用。那些通过网络提供服务的公司或个人（例如设计工作室或自由撰稿人）可以在线制订项目预算，与客户协调，通过 PayPal 进行结算并获得报酬。

用户想看的内容，而为了弥补这一点，搜索提供了找到内容的第二种途径。但是，如果用户自认能带来帮助的关键字并不在寻找的内容之中，那么这第二种途径很容易失败。因此，设计标准变成了：将内容和用户的用词进行关联。如果你没有理解人们为什么搜索、以及是什么让搜索能够成功，那么将很难得到上述的结论。但话又说回来，如果相关的问题已经过很好的研究，那么只要经过一些细致的调查，你自己也能得到答案。所以，首先自问有哪些因素有助于成功搜索，你就能找到线索，然后将其加入到设计标准中去。

的确，有些标准会比其他标准更难发现。但通过以上的解释，我们希望你能明白在这个过程中除了提出问题，然后将答案转化为相关标准之外，并没有其他复杂之处。

在接下来五章的设计标准部分都会阐明当存档自定义框架时如何做到这一点。

与此同时，让我们转而看看框架更为概念化的一面——让框架行之有效的哲学基础。

2.3.2 框架体系的特质

著名的信息架构师 Liz Danzico[①]（Happy Cog Studios[②]，Bobulate.com）曾经做过一次题为“The Framework Age”（框架时代）的演讲。演讲的主题是 Web 设计师们已经抛弃了严格支持剧本化的用户行为的设计，转而开始为之设计一种弹性的平台。Liz 的演讲重点关注于一种不断变化的网页范型，这和本书并无联系；但在演讲的过程中，她提到了用户定义框架的 3 个特质。

为了说明这些特质，Liz 谈到了古典音乐和莫达尔爵士乐之间的区别。

她解释到，在古典音乐中，每一个音符都是既定的。作曲家煞费苦心地为每一件乐器谱写乐曲的每一个小节中的每一个音符。要想成为大师级的乐手，技巧、风度和个人特色当然都很重要，但从根本上来说，任何乐手都得有能力和自制力来演奏出那些音符。每一次都必须完美、准确。在古典音乐中，只要有一个音符出现错误，那就是失败。

---

① Liz Danzico 具有设计师、教育工作者、编辑、信息架构师等多个头衔，同时她在 Happy Cog Studios 任用户体验顾问。

② Happy Cog Studios 是一个网站设计公司，为客户提供网站架构、界面设计、品牌塑造以及内容开发等多方位的服务。客户包括 Mozilla、WordPress、Blogger 等。公司网站是 http://www.happycog.com。

然而爵士乐[①]则完全不同。而莫达尔爵士乐[②]更是极端中的极端。

在录制那张首开先河的唱片 Kind of Blue[③]之前，传奇小号手 Miles Davis[④]走进录音棚，拿出了 6 张纸。在这些纸上只提供了少量的信息——歌曲的调式、速度以及使音乐不偏离方向的一些限制。除此之外什么也没有。

Miles Davis 并没有要求这些音乐家按照乐谱逐个音符、循规蹈矩地演奏，而是让他们跟随着律动进行谱曲。在演奏中进行创作。

他希望他们尽情表演，即兴发挥，让他们自己沉浸到音乐中。

尽管录音棚里没有任何人用这种方式演奏过，但是他们作为音乐家的进化，事实上也是爵士音乐的进化，就摆在面前的这 6 张纸上。

而通过 Miles Davis 简单却具革命性的要求，莫达尔爵士乐，一种本质上诞生于框架的音乐被推向了世界。

每首乐曲的骨架（轮廓、结构）是确定的，但其他的一切都是开放的。音乐家可以完全自由地在框架中演奏，试验，大展拳脚。

通过这个例子，Liz 阐述了她为框架定义的三个特质。

1．存在

首先，框架是存在的。用 Liz 的话说，它们“可检测，却又不固定”。这句话的意思是说它们存在，而且可以标识，但是它们的表现却绝不是一成不变的。

---

① 爵士乐是一种源自美国黑人的音乐形式（类似蓝调音乐），20 世纪早期产生于新奥尔良，基本乐器包括铜管、簧管乐器和鼓，后来加入了贝斯和吉他。早期的爵士乐是低下阶层的音乐，因此自由奔放，但凡缺乏精致的地方一律用音量和激情予以补偿。30 年代后，爵士乐吸引了白人阶层，同时诞生了可重复的简短旋律模式（riff），乐手可以在这种 riff 模式的基础上进行较长时间的独奏（多为即兴）。如今爵士乐已经成为一种主流的音乐风格，更诞生了无数技巧出神入化的大师。

② 莫达尔爵士乐（Modal Jazz）是上世纪 50 年代发展出来的一种爵士乐的分支类型。这种形式的爵士乐将 8 个音符随意组合构成不同音阶，在节奏上或和声上强调即兴色彩。演奏时乐手们不会按照事先排练好的乐谱演奏，而是凭着听觉和乐感自由即兴发挥，甚至可以说是在演奏中创造乐曲。

③ Kind of Blue 是美国爵士音乐家 Miles Davis 于 1959 年 8 月出版的录音室唱片。这张唱片是有史以来最为畅销的爵士录音唱片之一，获得过无数殊荣，对爵士、摇滚和古典乐都产生过极大的影响。

④ Miles Dewey Davis，小号手、爵士乐演奏家、作曲家、指挥家，20 世纪最有影响力的音乐人之一。第二次世界大战以后的每一次爵士乐发展运动，Davis 始终站在最前线。随着爵士乐逐渐被人们所接受，Davis 作品的影响力逐渐扩大，获得了不朽的声誉和滚滚财源，是当代成功音乐人的典范。

就像每一种音乐类型中都能找到框架的影子一样，几乎在每一个纵向市场中都能发现网站设计的足迹。这使得它们很容易被发现，存档和备案，但却仍然能根据不同的表现而保持其独立性。50 亿个搜索系统可能都提供了（本质上）完全相同的分页界面，但出于某些原因，任何一个搜索系统都会和其他搜索系统存在着或多或少的差别。可检测，却又不固定。

2．可累加

其次，框架是可累加的。设计师能以它们为基础，针对具体的解决方案以有效的方式对设计的规模进行缩放，同时将一系列框架串联起来，构成整个网站。

3．增强表现力

最后，框架是表现力的推动者。它们使设计师能够为作品赋予自己的风格。自定义设计，表演。

框架不会把用户限制在那些生搬硬套的规则中，它们允许即兴发挥。尽管框架所包含的元素始终不会有太大的变化，但就像设计模式一样，框架的每一次实现都必须适用于相应的特定环境。比如最常见的搜索结果页面，它已经成为每一个搜索系统中不可缺少的标准部分，但要想使用它，则必须考虑整个应用的上下文情境。设计师们的美学素养在这里派上了用场。正是在这里，设计师开始注入个人风格，调整细微之处，运用娴熟的技巧。正是在这里，设计师开始表演。

4．鼓励创新

因此，框架体系完全可以作为我们设计的起点，同时一定要认识到，以框架为标准并不代表着创新精神的消逝。除了提供一整套彼此关联的界面解决方案之外，框架还能让我们领悟如何提高水准，登上新的台阶。

看看当今主流的在线零售网站。你也许已经注意到，它们都在使用一种极为相似的信息架构。例如，当访问 Target.com[①]的主页时，你会四处寻找有关你想要的产品的链接（比如运动装备），寻找可能会包含该产品的商品种类，扫描搜索结果，找到一件希望深入了解的商品，然后点击进入详情页面仔细查看。

① Target.com 隶属美国 Target 零售公司，成立于 1902 年（前身为 Dayton 干货公司）。Target 的市场定位是高级折扣零售店，是美国第二大零售商，仅次于沃尔玛。2009 年，Target 在财富 500 强中排名第 28。目前的 Target.com 由 Target 和 Amazon 共同管理，但双方的合作协议将于 2011 年到期。

互联网上的任何一家零售网站都支持这一核心的任务流程。（当然，炉火纯青的 Google 搜索让我们几乎能够完全摈弃在线购物的这一流程，但那是另一码事。）

为什么会这样？

因为在线的购物体验符合我们购物的一贯心理模式——实际上，它和我们在日常生活中的购物方式完全相同。这里面没有任何特别之处。Target.com、Barnesandnoble.com[①]、Amazon.com 和其他许多在线零售商都支持常见的用户行为。

而这么做的理由是，一些人注意到了用户们惯常的行为，决定要对其进行在线的支持，于是便设计出一些东西把这些实际的行为照搬到网上。这些零售商们决定要全盘模拟现实世界中的购物行为，但其实并无这个必要。事实上，只要他们真正理解了现实世界中的人类行为，就可以创造出完全不同的解决方案，用更有趣的方式来解决问题。

而你的机会正在于此。

正是心理因素导致了现有的各种标准化的解决方案，但它同样也能衍生出其他更为引人注目的设计。在决策过程中保持以这种心理影响为中心，你就有能力构思出不可思议却又对用户同样有效的设计。

简而言之，框架从不限制创新，它们鼓励创新。

2.3.3 第一个框架体系资源库

在本书的创作期间，框架体系仍然处于初期发展阶段。事实上，你手中的这本书正是第一部从用户体验而非代码的角度来讨论框架的作品。因此，我们也没有听说过任何公开的框架体系资源库（之前曾筹备过的一些资源库都属私有）。

于是我们创建了一个，如图 2-9 所示。地址是 http://webanatomy.rhjr.net。

“Web 解剖学”框架体系资源库包含了本书中所有框架实例的文档记录。我们希望以你的反馈、想法和贡献为基础，让它持续成长、不断壮大。

正如你在该网站中所看到的，框架资源库基本上就是模式资源库的一个外

① Barnesandnoble.com 隶属美国 Barnes & Noble 公司，前身是 1873 年 Charles Barnes 创办的印刷行。目前它是美国最大的图书零售商，产品范围包括书报杂志、DVD、漫画、礼品、游戏和音乐。该公司于 20 世纪 80 年代末期开始图书的在线销售，但直到 1997 年才发布公司网站。如今在网上销售 100 多万种商品。

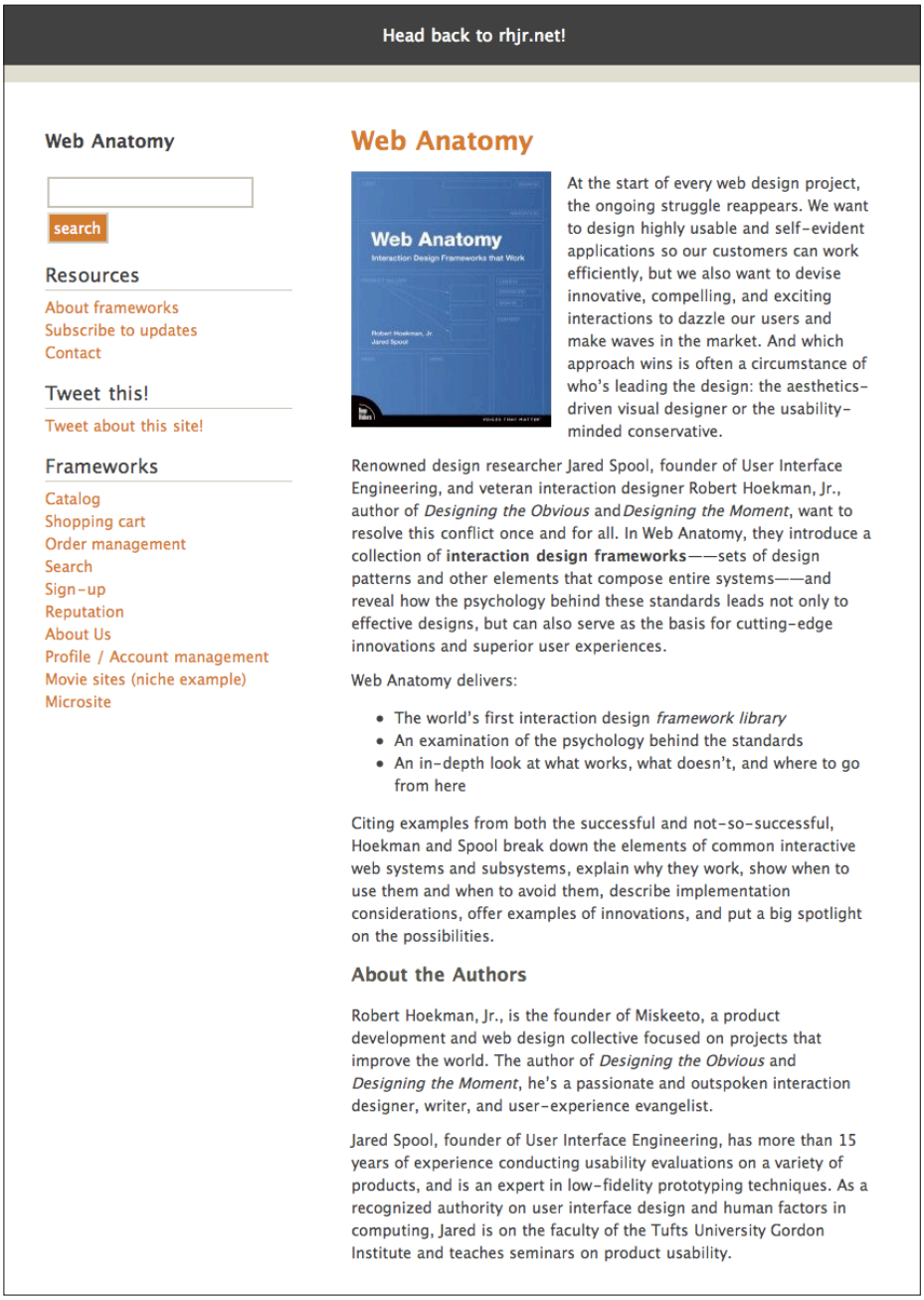

图2-9
位于http://web-anatomy.rhjr.net上的首个公用框架体系资源库

包装。如果你已经为自己的公司（或组织）创建了模式资源库，那么一切将非常简单。即使你还没有创建，也完全可以让二者齐头并进：先利用 wiki 或者其他方式来创建框架文档，然后将文档中构成元素项里列出的元素链接到对其进

行描述的相应模式文档上。

在“Web 解剖学”资源库中，尽可能将框架所包含的元素链接到各个公共模式库。如果某个列出的元素无法对应任何公共资源库，我们才会尝试对其进行描述。希望其他的模式资源库在日后能够对这些元素进行归档，以方便我们进行链接。

作为第一个公共框架资源库，它对应了本书中的每一个实例。如果你把它和本书结合起来使用，不仅能了解如何为框架存档，还能了解如何与其他人共享框架。而就其本身而言，我们也相信使用者能够获得在本章和前一章中所提到的一切好处。

我们当然希望它不会是最后一个公共框架资源库。我们还希望其他个人、公司或组织也能够开放他们的资源库，供大家自由使用。

你自己的资源库

像你所看到的那样，我们的资源库由静态的网页构成，而不是通过可供大家任意编辑的 wiki 页面来表示。事实上，Web 解剖学资源库构建在 WordPress[①] 之上，这是一种流行的内容管理系统（content-management system）和博客工具，而且它具有自定义的 WordPress 主题。

你可以使用任何喜爱的工具来创建自己的资源库。不过重要的是，同公司内的所有人都应当有访问的权限，而且应使用一种能支持快速更新的方法来记录资源库（相信我们，此举的理由非常明显，你绝不会希望用纸来记录框架资源库）。

2.4 自然环境下的框架

Robert 曾在 Jared 的 UIE Web App Summit[②] 上举办过一次有关框架体系的

① WordPress 是一个使用 PHP 语言开发的免费、开源的博客平台，于 2003 年创立，2004 年开始受到用户的普遍欢迎，现在已发展成为全球最知名的博客服务提供商之一。WordPress 功能强大，插件众多，可以说是目前世界上最先进的博客系统，目前开发的其他很多系统都参考了它。除了提供免费的博客平台以便用户自由建立托管博客以外，WordPress 还免费提供了使用 PHP 语言和 MySQL 数据库开发的开源博客系统引擎以供下载，用户可以在支持 PHP 和 MySQL 的服务器上安装该系统以便建立自己的独立博客。同时，WordPress 提供了大量的主题（theme）插件，用户可以自由下载收费或免费的主题，使自己的博客更为美观、更具特色。WordPress 的网址为 http://wordpress.org（中文站点为 http://cn.wordpress.org）。

② UIE 网页应用开发峰会（UIE Web App Summit）是 UIE 公司举办的年度盛会，包括为期各 2 天的密集讲座和专家演讲，范围涵盖 Ajax、RIA、表单设计、草图设计、可用性、设计模式和 Web 标准等。网站是 http://webappsummit.com/。

讲座，有人提出了一个有趣的问题：框架体系是否会让交互设计产生同一化，最后让交互设计师们丢掉饭碗？

答案当然是不会。

恰恰相反，建立框架体系的目标之一就是为了让设计师们能避免把时间都花在单调、重复的界面和架构设计上，从而能有更充裕的时间来做设计师们应该做的事情：致力于为企业和客户提升用户体验。

讲座的另一位听众问到，这些框架累加到最后，是否会出现整个界面变成臃肿的科学怪人[①]那样的局面，并询问如何避免。我们的回答是：

和人体系统一样，框架体系相互间的结合可以说是完好无缺。它们彼此之间有很多重合的部分，而且联系非常紧密，因此相互之间的界限几乎根本看不到，能够实现完全无缝的交互。在很多情况下，一个模式很快就能变身为多个框架的构成部分，而且为用户扮演多种角色。其中一个例子就是名为 Register Now（马上注册）的按钮，它可以用于鼓动用户为某次会议登记，也可以作为后台账户管理系统的入口，方便会议组织者在会议之前与登记者取得联系。

一个单独的框架体系本身并没有太多价值。即使是 Google 的搜索框架也不例外，它是该公司核心任务流程的主导框架，但仍然需要与其他元素协同合作，例如为访问其网站的用户提供导向，鼓励他们使用更多的 Google 服务，以及登录并获得之前储存的信息。只有把各个框架体系编织在一起，才能构成一个完整的网站或应用——形成一次完整的体验。

框架面对面

接下来的四章将为大家极为详细地介绍了几个重要的框架：目录、搜索、注册以及关于我们。在这几章中，我们将讨论是哪些人类行为导致这些框架成为了标准，以及与之相关的调查和可用性研究的结果，还有当你在自己的项目中使用它们时需要考虑的问题。

在第 7 章，我们将展示的框架体系面向的是一个专门的市场——电影工业。这么做并不是因为我们相信所有的读者都和电影工业有关（当然或许你正是其中之一），而是希望借此阐明对于这种不太常见的框架，我们应该如何标识出它所包含的元素。事实上，几乎每一种领域都有它自身的一套标准化方案，例如

① “科学怪人”的典故来自于 1818 年出版的哥特小说 Frankenstein（弗兰肯斯坦）（又名 The Modern Prometheus，现代普罗米修斯）。故事中的疯狂医生从坟场挖出尸体，挑选其中还能使用的部分，胡乱拼凑成臃肿巨大的人型并赋予其生命。

高等教育网站都会有申请表，摄影图库网站都会有为摄影师提供的管理工具，金融网站都会有投资组合管理系统，等等。而我们将利用这一章来说明应该如何了解这些特定行业的具体情况，以及如何让它们与其他框架共同构建出一个完整的网站。

再次强调，当你完成了这些的时候，请访问 http://webanatomy.rhjr.net，这里有一个活生生的框架资源库，它能帮你记录你自己的框架，并筹备你自己的资源库。

让我们去近距离地看看框架吧！