Join the discussion @ p2p.wrox.com
Wrox Programmer to Programmer™
Web开发经典丛书

Reliable JavaScript: How to code safely in the world's most dangerous language

编写可靠的JavaScript代码

测试驱动开发JavaScript商业软件

[美] Lawrence D. Spencer 著
Seth H. Richards
王肖峰 译

清华大学出版社

Web 开发经典丛书

编写可靠的 JavaScript 代码

测试驱动开发 JavaScript 商业软件

[美] Lawrence D. Spencer 著
Seth H. Richards
王肖峰 译

清华大学出版社
北 京

1467031

Lawrence D. Spencer, Seth H. Richards
Reliable JavaScript: How to code safely in the world's most dangerous language
EISBN: 978-1-119-02872-7
Copyright © 2015 by John Wiley & Sons, Inc., Indianapolis, Indiana
All Rights Reserved. This translation published under license.
Trademarks: Wiley, the Wiley logo, Wrox, the Wrox logo, Programmer to Programmer, and related trade dress are trademarks or registered trademarks of John Wiley & Sons, Inc. and/or its affiliates, in the United States and other countries, and may not be used without written permission. All other trademarks are the property of their respective owners. John Wiley & Sons, Inc., is not associated with any product or vendor mentioned in this book.

本书中文简体字版由 Wiley Publishing, Inc. 授权清华大学出版社出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

北京市版权局著作权合同登记号 图字：01-2015-7460

Copies of this book sold without a Wiley sticker on the cover are unauthorized and illegal.

本书封面贴有 Wiley 公司防伪标签，无标签者不得销售。
版权所有，侵权必究。侵权举报电话：010-62782989　13701121933

图书在版编目(CIP)数据

编写可靠的 JavaScript 代码　测试驱动开发 JavaScript 商业软件/(美)劳伦斯·D. 斯潘塞(Lawrence D.Spencer) 等著；王肖峰 译. —北京：清华大学出版社，2016
(Web 开发经典丛书)
书名原文：Reliable JavaScript: How to code safely in the world's most dangerous language
ISBN 978-7-302-44397-1

Ⅰ. ①编…　Ⅱ. ①劳… ②王…　Ⅲ. ①JAVA 语言—程序设计　Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字(2016)第 165262 号

责任编辑： 王　军　于　平
装帧设计： 孔祥峰
责任校对： 成凤进
责任印制： 何　芊

出版发行： 清华大学出版社
网　　址：http://www.tup.com.cn，http://www.wqbook.com
地　　址：北京清华大学学研大厦 A 座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质 量 反 馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者： 清华大学印刷厂
经　　销： 全国新华书店
开　　本： 185mm×260mm　　**印　　张：** 30.75　　**字　　数：** 729 千字
版　　次： 2016 年 8 月第 1 版　　**印　　次：** 2016 年 8 月第 1 次印刷
印　　数： 1~3000
定　　价： 69.80 元

产品编号：066972-01

译 者 序

JavaScript 作为 Web 开发中的一门重要技术，经历了长足的发展。虽然在初期人们仅仅将它看作一门普通的脚本语言，但随着 Web 2.0 和 HTML 5 的兴起以及 Node.js、jQuery、AngularJS 等重要程序库的出现，JavaScript 变成了一门广泛应用在各种设备上的开发语言。而与此同时，它独特的语言特性和不可靠的行为则让人望而生畏，如何使用它编写可靠的企业级应用程序成为一个难题，而本书试图解决这个问题。

本书首先介绍了软件工程的重要概念，例如 SOLID 和 DRY 原则；然后演示了如何使用 JavaScript 实现各种代码模式，例如回调、单例等；接着讲解了如何使用 JavaScript 的高级特性，例如混合、方法借用等；最后讲解了 DOM 测试和常见的代码检查工具。与此同时，测试驱动开发的概念将贯穿本书的所有内容。因此，本书不仅讲解了如何解决 JavaScript 中容易混淆的问题，也演示了如何开发正确的代码，软件工程和测试驱动开发概念的采用将成为开发可靠 JavaScript 代码的重要保障。

我非常高兴清华大学出版社引进了本书，也非常高兴自己能够负责本书的翻译工作。通过这个过程，不仅可以为读者带来一本真正的 JavaScript 开发秘籍，帮助大家了解如何在 JavaScript 开发中应用软件工程的概念，也让自己加深了对 JavaScript 开发的理解。

最后，感谢清华大学出版社的编辑们为本书付出的心血。同样感谢妻子对我翻译工作的支持和鼓励。没有你们的支持和鼓励，本书就不可能顺利出版。

对于这本经典之作，译者对本书进行了详细的阅读，对其中一些具有争议的地方也进行了反复的考证，但个人精力有限，难免有疏漏之处，敬请各位读者谅解。如有任何意见或建议，请不吝指正。本书全部章节由王肖峰翻译，参与本次翻译的还有杜欣、高国一、孙其淳、孙绍辰、徐保科、尤大鹏、张立红、邓伟、王蕊、王小红、马宁宁。

最后，希望各位读者能够早日熟练掌握各种 JavaScript 开发技能，坚持使用测试驱动开发创建可靠的企业级应用程序。

译者

作 者 简 介

Lawrence Spencer是ScerIS的应用开发副总裁，ScerIS是马萨诸塞州萨德伯里的一家软件和服务公司。他和他的团队使用AngularJS创建了基于浏览器的应用程序，服务器后端使用的是C#/Web API/SQL Server。Lawrence 35年职业生涯中使用过的语言包括COBOL、C、C++、C#，甚至还有大型机汇编语言，但是他说JavaScript是最有趣的。他经常在Code Camps和其他聚会上演讲，与开发社区分享他对软件的热爱。可以在http://FascinatedWithSoftware.com找到他的博客。

Lawrence 的其他兴趣包括哲学、象棋和古典吉他。他居住在马萨诸塞州马尔伯勒。

Seth Richards 从 2002 年开始一直在专业开发软件。从为酒吧和夜总会的嵌入式设备编程开始，他在 2007 年开始转做 Web 应用开发。他已经经历过众多基于 Web 的应用程序，从系统导向的企业级地理信息物理资产管理系统到产品发现和推荐的社交网络。

Seth 毕业于新罕布什尔州的普利茅斯州立大学，他在这里学习了计算机科学和数学。目前，他正在攻读乔治亚理工学院计算机科学的理科硕士学位。Seth 的博客网址为 http://blog.shrichards.com。

技术编辑简介

Keith Pepin 已经在网站和网络应用的开发上经历了超过 17 年的时间。在他早期的职业生涯中，他喜欢上了 JavaScript 并从此热衷于构建动态用户体验。目前，他是 Meltwater 的高级软件工程师，使用 HTML5、CSS3、JavaScript、AngularJS、Node.js 和 MongoDB 构建他们下一代的在线营销智能平台。在业余时间，他喜欢与其他极客一样的追求，包括各种游戏、漫画书、绘画和素描。

John Peloquin 是一位软件工程师，他拥有 10 年以上各种规模应用程序的 JavaScript 开发经验。John 在加州大学伯克利分校的数学系得到了学士学位，目前是 Spreemo 的首席工程师，Spreemo 是纽约新兴的医疗技术公司。在编辑本书之前，他编辑过 *Professional Website Performance*(Peter Smith 编著，Wiley 2012)和 *Professional JavaScript for Web Developers*, 3rd ed(Nicholas Zakas 编著，Wiley 2012)。在业余时间，他偶尔会在即兴表演会上表演脱口秀。

试读结束：需要全本请在线购买： www.ertongbook.com

致　谢

感谢我的妻子 Bethany，感谢她在我编写本书时对我的爱和支持，以及忍受我为了如期交稿而无法陪伴在她身边的许多夜晚和周末。

—Seth Richards

感谢我的家人，感谢他们鼓励我追求自己的梦想。

—Lawrence Spencer

如果其他人不愿意与我们分享他们的知识和经验，或者如果没有社区大量的图书、博客和源代码，本书的出版就无法成为现实。总之，我们要感谢：

- Douglas Crockford，感谢他揭示了 JavaScript 的优点以及他在 jsLint 上所做的工作。
- Nicolas Zakas，感谢他编写的众多书籍和博客文章(可以作为经历 JavaScript 危险水域时的指南)，以及他对 ESLint 的维护和共享。
- Stoyan Stefanov，感谢他指导我们如何在 JavaScript 中应用基于模式的开发。
- Robert C. Martin，感谢他逐渐向我们灌输编写干净代码的意愿。
- Fredrik Appelberg，感谢他创建了 AOP.js 面向切面编程框架，并感谢 Dave Clayton 为该框架做出的贡献。
- Mike Bostock，感谢他使用 SVG 图形的 D3 库激励我们。
- Pivotal Labs 的成员们，感谢他们创建了开源 JavaScript 框架 Jasmine，并感谢社区成员对该框架做出的贡献。
- AngularJS 团队，感谢他们向世界展示了一种构建单页面应用程序的伟大方式。
- 感谢像 Stack Overflow 和 GitHub 这样的网站上大量的和不断增长的由慷慨的人们构建的网络。没有你们，我们可能仍然在翻阅手册。

我们还希望表达对项目编辑 Chris Haviland 的感激，他灵活地控制了从头到尾的编写过程。与其他任何人相比，我们的文字编辑 Nancy Rapoport 更加认真并多次阅读了我们的书稿。我们衷心感谢她的奉献和建议。我们还希望向技术编辑 Keith Pepin 和 John Peloquin 表达最真诚的感谢。他们超凡的 JavaScript 技术水平帮助我们避免了不少技术错误。如果仍然有任何错误存在，那么可能是因为我们没有遵守他们的建议。向你们致敬，先生们。

最后，我们要感谢 Carol Long，Wiley 的执行策划编辑，她给予了我们编写本书的机会。Carol 就在我们完成本书之前宣布了她要从出版业退休。谢谢 Carol，我们希望你在退休生活中享受阳光和玛格丽塔酒。

—Lawrence 和 Seth

前　言

当我们与开发者同仁分享本书的书名《编写可靠的 JavaScript 代码》时，我们收到了如下反馈：

“‘可靠的’和‘JavaScript 代码’这两个词现在能放在一块？”

“这一定是一本非常薄的书。”

“我是否会在书店的小说区域 John Grisham 最近的惊悚小说附近找到它？”

不，本书不是小说。

我们收到的关于本书书名的反馈表明了一些具有经典的、编译语言经验的开发者对于 JavaScript 的普遍看法：JavaScript 被用于创建华丽的作品集网站或者简单的应用程序，它与任务关键的企业应用程序没有关系。

在过去这种观点是事实，但现在已经改变了。

JavaScript 作为一流语言崛起

JavaScript 作为野孩子的声誉是应得的，在接下来的两个小节中我们将揭示它的一些漏洞。不过，就像被宠坏的继承者一样，在继承了家族事业之后，通过提出挑战让所有人都感到震惊。但最近他变得严肃和负责，显示出真正伟大的能力。

JavaScript 早期仅仅被托付一些短“脚本”任务。任务非常简单：如果所需的字段未被填充，就将它标记为红色；如果按钮被单击了，就应该将另一个页面带入视图。尽管它的责任有限，但易于相处。到今天为止，大多数程序员使用 JavaScript 的经验都是此类。

接着，在重新定义 JavaScript 的变革中，世界转向了 Web。这一直是 JavaScript 游乐场，当 The Old Boys Club 的成员在服务器上完成真正的工作时，他就在 Web 中娱乐自己。

在 20 世纪 90 年代末期，当微软引入 iframes 和 XMLHTTP 时，变革开始被阻碍。而当 Google 在 2004 年和 2005 年分别使 Ajax 成为 Gmail 应用程序和 Google Maps 的一部分时，变革开始崩溃。世界突然注意到当浏览器被赋予更多任务时(不只是显示服务器分配的内容)，Web 体验可以如此丰富。

所以 JavaScript 被赋予了更多的责任，超出了任何人的计划。JavaScript 需要得到帮助。

此时帮助也应运而来，以工具包和框架的形式存在，例如 jQuery、 Ext JS、 Ember.js、Knockout、Backbone 和 AngularJS。这些有价值的顾问完成了所有他们能做的事情，为 JavaScript 带来了准则和结构。

使用 JavaScript 编写真正糟糕的代码是非常简单的

问题的部分原因是 JavaScript 作为页面脚本语言存在多年。在那个有限的范围内，使用全局变量或全局函数并没有问题。如果变量拼写错误，影响非常有限、也易于追踪(另外，影响可能是创建了另一个全局变量)。如果架构非常草率……那么有多少架构甚至可以只存在于一个 Web 页面上呢？

编译器的缺少隐藏了潜在的错误。使用 C#或 Java 编写的服务器端程序在运行之前至少可以保证语法上是正确的。而 JavaScript 则必须启动，然后就只能预祝好运了。拼写错误的变量或者调用不存在的函数都可能潜伏在代码中数月之久，直到遵循了特定的执行路径。

接着就是 JavaScript 的怪癖，一些可爱的、疯狂的怪癖。

这个列表的顶部一定就是==(使用类型强制转换的相等)和===(不使用类型强制转换的相等)。这是一个好想法，但是对于主要使用其他语言的程序员来说是如此难以适应！

JavaScript 拥有“真”和“假”的概念，它们让很多人感到糊涂。0 是假值，因此多亏了类型强制转换(type coercion)，表达式：

```
false == '0'
```

的结果是真。值 false 被强制转换成一个数字，也就是 0(真被转换为 1)。接下来，字符串‘0’也被强制转换成数字，也是 0，所以表达式的结果是真。

不过，false == 'false'的结果是假，因为左侧的 false 再次被强制转换成 0，与右侧字符串‘false’相比，也被强制转换成数字。因为‘false’根本不是数字，所以第二个转换将产生 NaN(Not a Number)，相等性检测失败。

如果声明了下面的函数：

```
function letsHaveFun(me, you) {
  // Fun things happening between me and you
}
```

然后调用它，因此：

```
letsHaveFun(me);
```

JavaScript 将在变量 *you* 未定义的情况下执行调用，只是为了开心地看着你与某个不存在的人玩耍。

这个代码清单可以不断扩展下去。还有令人吃惊的作用域规则、独特的“原型”继承机制、自动的和有时不正确的分号插入、一个对象从另一个完全无关的对象借用函数的功能等 。

伴随着全局变量的不请自来、几乎完全缺少的架构设计、与真理的可疑关系以及比动漫大会上发现的更多的怪癖，JavaScript 能做到如此好的地步真是一个奇迹。

无论你相信与否，它在变好之前会变得更糟糕。即使你是正确的，它也可能会非常容易出现问题。

JavaScript 代码容易被无意间破坏

JavaScript 有一种病态的幽默感。在一门稳重的编译语言中，如果有一行完全正确的调试代码在生产环境中运行得非常完美，例如：

```
myVariable = myObject.myProperty;
```

这时如果不小心碰到了键盘上的 x 键，那么它就变成了：

```
myVariable = myObject.myPropxerty;
```

编译器就会立即发出一条严厉的消息，警告下次一定要更加小心。而 JavaScript 则会高兴地运行该代码，并将 undefined 的值赋给 myVariable。

当你希望改变属性的名称时，JavaScript 喜欢玩捉迷藏。你可能会认为在整个源目录中搜索.myProperty 就可以找到所有需要改动的地方。“不，不，不！”JavaScript 笑着说。“你忘了搜索['myProperty']。”

实际上，应该使用正则表达式进行搜索，允许在括号和引号之间含有空格。你曾经这样做过吗？我们都没有。

你还应该搜索下面这样的结构：

```
var prop = 'myProperty';
// . . .
myObject[prop] = something;
```

即使是如此小的重构也很难完成，因此可以想象错误的产生是多么容易。不适合重构的代码几乎是单词“脆弱”的定义。

如何才能避免这些问题呢？如果有一个我们希望在本书进行宣扬并实践的概念，那么就是测试驱动开发。在不存在编译器的情况下，测试是针对错误的最佳防御。

JavaScript 也更加适用于按照软件工程的规则进行开发。实际上，由于 JavaScript 具有创造性，因此比其他语言更需要这些规则。

我们见过许多开发人员都对这个信息抱着开放的态度，并且希望学习更多相关知识。我们希望你是其中一员。

本书面向的读者

因为本书不是一本 JavaScript 入门书籍，所以我们假设你已经有一些 JavaScript 经验。下面勾勒了本书理想读者的特点。

从其他语言转向 JavaScript 的开发人员

我们的职业生涯都不是从 JavaScript 开发人员开始的，可能也不是当 JavaScript 遇到大规模应用开发时才进入程序开发领域的。

JavaScript 与我们曾经使用过的任何语言都大不相同。我们来自于编译的、静态类型语言 C#的舒适世界。

当我们在保持 C#程序员的架构和准则，并拥抱 JavaScript 的动态特性时，JavaScript 就会容易掌握。

如果你具有使用 JavaScript 之外的语言(例如 C#或 Java)进行思考和编程的背景的话，本书是适合你的。数据结构和架构方面的知识将为你掌握 JavaScript 的大规模开发提供牢固的基础。

许多小节都演示了 C#和 Java 中的语言特性(例如继承和接口)是如何对应于 JavaScript 中的功能的。我们还强调了 JavaScript 和其他语言之间的许多主要区别，例如作用域规则和类型强制转换相等性比较。掌握 JavaScript 功能和特性的相关知识将改善你用 JavaScript 思考的能力。

本书的另一个关注点是如何将 C#和 Java 开发中更常见的软件工程概念和实践应用到 JavaScript 中，例如设计模式、单元测试和测试驱动开发。合理的软件工程可以缓和 JavaScript 的本性，创建可靠的和可维护的代码。

具有小规模 JavaScript 经验的开发人员

在我们努力为团队增加具有 JavaScript 经验的开发人员时，我们遇到了许多候选人，都觉得具有小规模 JavaScript 开发经验(例如输入域验证和 jQuery 元素转换)就值得将“JavaScript”列在简历的重要位置。

在面试中，判断这样的候选人不需要花费太多时间：可能在 ASP.NET Web Forms 应用程序中，他们可以轻松地使用按钮处理程序，但是创建一个 JavaScript 模块，并在其中使用防止被外部操作的变量就比较困难了。

随着机构对 JavaScript 使用的进一步深化，我们对拥有 JavaScript 经验的定义也随之严格。数年之前，如果开发人员有点 jQuery 经验，我们就会对他的 JavaScript 经验非常满意了。

不过，现在我们需要更多的经验。使用 JavaScript 编写整个应用程序已经不再是凤毛麟角了。在所谓的单页面应用(Single-Page Application，SPA)中，JavaScript 代码将组成整个应用程序，与过去瞬间的单击处理程序相比，现在它承担着更多的责任。为了参与大规模 JavaScript 应用程序的开发，开发人员必须知道如何以结构化和规定的方式使用该语言，并同时使用它的许多独特功能和特性。

通过本书的样例，我们希望你——小规模 JavaScript 开发人员——能够参与大规模 JavaScript 应用程序的开发。

负责为新项目挑选编程语言的开发人员

可能你已经听过这个谚语："没有人会因为购买 IBM 产品而被解雇"。这个谚语反映了一种感觉:在为 IT 项目选择技术合作伙伴时，选择稳定的、有信誉的公司(例如 IBM)是不会受到质疑的。即使项目费用超支、逾期或完全失败，选择 IBM 也是无可指责的。

如果你正处于为新应用选择开发语言的位置，那么就处于与 IT 管理者选择技术合作伙伴一样的位置。有许多具有悠久历史并经过众人尝试和验证的语言，例如 C#和 Java，它们都是由大型、稳定的技术公司所支持的，已经被用于构建 Web 和桌面应用超过十年。没有人会因为选择 C#被解雇。

从新程序设计项目的安全选择来看，尤其是企业级的项目，JavaScript 明显与 C#不同。JavaScript 不是一门成熟、稳定、正式的编程语言。

与 C#和 Java 这样的语言相比，它没有大规模软件项目的、长期的成功记录。这不是说使用 C#和 Java 的项目就一定成功。不过，如果使用了其中一种语言的项目不成功，语言的选择可能不会被包含在失败的原因中。

如我们在之前小节中所提到的，使用 JavaScript 非常容易编写出灾难性的代码。这为它带来了一点糟糕的名声，降低了选择它的可能性。

不应该因为 JavaScript 的声誉而自动将它排除在项目的考虑之外，因为我们可以从它的强大功能中获益。Node.js——一个服务器端的 JavaScript 引擎——是轻量级的并且高度可伸缩的；对于实时和数据敏感应用程序来说是非常完美的选择。JavaScript 可以被用于在浏览器中创建富用户界面。客户端框架(例如 Ember 和 AngularJS)可以被用于构建完整的、基于浏览器的应用程序，可以通过将展示逻辑转移到客户端的方式减轻 Web 服务器的负担。

尽管我们不能保证它会成功，但是接下来的章节将展示如何通过应用我们在自己项目中所学到的教训，在下一个项目选择 JavaScript 时降低风险。

成功不是偶然的，尤其是在使用 JavaScript 时。它要求牢固掌握软件工程原则，这是第 1 章的主题。

本书的组织结构

我们将本书划分为 5 个部分。

第 I 部分"奠定坚实的基础"涵盖了软件工程的关键概念，例如 SOLID 和 DRY 原则，还讨论了单元测试和测试驱动开发的优点。第 I 部分还介绍了本书使用的工具和 JavaScript 库。最后，讨论了 JavaScript 中的对象和它们的可测试性。

在第 II 部分"测试基于模式的代码"中，我们描述并使用测试驱动开发创建了几个有用的代码模式。其中一些模式(例如单例)可能与其他你所熟悉的语言中的模式类似。其他(例如承诺)主要是与 JavaScript 相关的。

第III部分"测试和编写高级 JavaScript 特性"描述了如何使用和测试 JavaScript 语言更高级的特性。它还涵盖了使用了高级编程架构的应用的创建和测试，例如中介者和观察者

试读结束：需要全本请在线购买：　www.ertongbook.com

模式。

第Ⅳ部分“测试中的特殊主题”提供了测试 DOM 操作的样例，还演示了用于增强代码标准的静态分析工具的使用。

第Ⅴ部分“总结”回顾了测试驱动开发的概念，还展示了一些本书用到的 JavaScript 习语。

使用本书所需的工具

为了运行本书的样例，需要使用下列工具：

- 文本编辑器
- Web 浏览器

样例的源代码可以从 Wrox 网站下载：

```
www.wrox.com/go/reliablejavascript
```

基于本书的开源软件可以在 GitHub 上找到：

```
www.github.com/reliablejavascript
```

源代码

当你浏览本书的样例时，可能会选择手动输入所有代码，或者使用随书提供的源代码。本书使用的所有源代码都可以从 www.wrox.com 和 www.tupwk.com.cn/downpage 下载。对于本书来说，代码下载文件在下面网址的 Download Code 选项卡中：

```
www.wrox.com/go/reliablejavascript
```

也可以在 www.wrox.com 中通过 ISBN 搜索本书找到代码。所有当前 Wrox 图书的代码下载的完整文件列表在 www.wrox.com/dynamic/books/download.aspx 中。

www.wrox.com 的大多数代码都采用.ZIP、.RAR 或者适用于特定平台的类似归档模式进行了压缩。下载了代码之后，只需要使用适当的压缩工具解压即可。

注意：

因为许多书都有相似的书名，所以你可能发现使用 ISBN 搜索是最容易的；本书英文版的 ISBN 是 978-1-119-02872-7。

勘误表

尽管我们竭尽所能来确保在正文和代码中没有错误，但人无完人，错误难免会发生。如果你在 Wrox 出版的书中发现了错误(例如拼写错误或代码错误)，我们将非常感谢你的反

馈。发送勘误表将节省其他读者的时间，同时也会帮助我们提供更高质量的信息。

要找到本书的勘误页面，可以进入 www.wrox.com，使用 Search 搜索框或书名列表找到本书，然后在本书的详细信息页面上单击 Book Errata 链接。在这个页面上可以查看为本书提交的、Wrox 编辑粘贴上去的所有错误。完整的书名列表(包括每本书的勘误表)也可以从 www.wrox.com/misc-pages/booklist.shtml 上获得。

如果在本书的勘误页面上没有看到你发现的错误，则可以到 www.wrox.com/contact/techsupport.shtml 上填写表单，把你发现的错误发给我们。我们会检查这些信息，如果属实，就把它添加到本书的勘误页面上，并在本书随后的版本中更正错误。

p2p.wrox.com

如果想和作者或同行进行讨论，请加入 p2p.wrox.com 上的 P2P 论坛。该论坛是一个基于 Web 的系统，你可以发布有关 Wrox 图书及相关技术的消息，与其他读者或技术人员交流。该论坛提供了订阅功能，当你感兴趣的主题有新帖子发布时，系统会邮件通知。Wrox 的作者、编辑、其他业界专家和像你一样的读者都会出现在这些论坛中。

在 p2p.wrox.com 网站上，你会找到很多不同的论坛，它们不但有助于你阅读本书，还有助于你开发自己的应用程序。加入论坛的步骤如下：

(1) 进入 p2p.wrox.com，单击 Register 链接。

(2) 阅读使用条款，然后单击 Agree 按钮。

(3) 填写加入该论坛必需的信息和其他你愿意提供的信息，单击 Submit 按钮。

(4) 你将收到一封电子邮件，描述如何验证你的账户和完成加入过程。

注意：不加入 P2P 也可以阅读论坛里的消息。但是如果要发布自己的消息，就必须加入。

加入之后，就可以发布新的消息和回复其他用户发布的消息。可以随时在 Web 上阅读论坛里的消息。如果想让某个论坛的新消息以电子邮件的方式发给你，可以单击论坛列表中论坛名称旁边的 Subscribe to this Forum 图标。

要了解如何使用 Wrox P2P 的更多信息，请阅读 P2P FAQ，其中回答了论坛软件如何使用的问题，以及许多与 P2P 和 Wrox 图书相关的问题。要阅读 FAQ，单击任何 P2P 页面上的 FAQ 链接即可。

目　录

试读结束：需要全本请在线购买： www.ertongbook.com