TURING 图灵程序设计丛书

MANNING

JavaScript Web 应用开发

[阿根廷] Nicolas Bevacqua 著
安道 译

JavaScript Application Design
A Build First Approach

中国工信出版集团
人民邮电出版社
POSTS & TELECOM PRESS

TURING 图灵程序设计丛书

JavaScript Web 应用开发

[阿根廷] Nicolas Bevacqua 著
安道 译

JavaScript
Application Design
A Build First Approach

人民邮电出版社
北 京

图书在版编目（CIP）数据

JavaScript Web应用开发 / (阿根廷) 比瓦卡(Bevacqua,N.) 著 ; 安道译. -- 北京 : 人民邮电出版社, 2015.9
（图灵程序设计丛书）
ISBN 978-7-115-40210-3

Ⅰ. ①J… Ⅱ. ①比… ②安… Ⅲ. ①JAVA语言－程序设计 Ⅳ. ①TP312

中国版本图书馆CIP数据核字(2015)第201195号

内 容 提 要

本书是面向一线开发人员的一本实用教学，对最新的 Web 开发技术与程序进行了全面的梳理和总结，为 JavaScript 开发人员提供了改进 Web 开发质量和开发效率的最新技术。本书主要分两大块，首先是以构建为目标实现 JavaScript 驱动开发，其次介绍如何管理应用设计过程中的复杂度，包括模块化、MVC、异步代码流、测试以及 API 设计原则。

本书适合各层次 Web 开发人员阅读。

◆ 著 [阿根廷] Nicolas Bevacqua
译 安 道
责任编辑 岳新欣
执行编辑 张 曼
责任印制 杨林杰

◆ 人民邮电出版社出版发行 北京市丰台区成寿寺路11号
邮编 100164 电子邮件 315@ptpress.com.cn
网址 http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：800×1000 1/16
印张：18
字数：435千字 2015年9月第1版
印数：1－3 500册 2015年9月北京第1次印刷
著作权合同登记号 图字：01-2015-5405号

定价：59.00元

读者服务热线：(010)51095186转600 印装质量热线：(010)81055316
反盗版热线：(010)81055315
广告经营许可证：京崇工商广字第 0021 号

序

近几年，开发强大的JavaScript Web应用经历了一场轰轰烈烈的复兴。人们对JavaScript寄予厚望，越来越多的人使用这门语言开发应用和接口，这本书的出版恰逢其时。在这本书中，Nico Bevacqua通过简洁的示例、这个领域沉淀下来的经验教训，以及可伸缩性开发的关键概念，向我们展示了如何改进应用的设计和流程。

这本书还能帮助你打造一个能节省时间的构建过程。时间是保持效率的关键因素，而作为Web应用开发者，我们希望能充分利用我们的时间。“构建优先”原则能帮助我们从开发伊始就注重应用的结构，以便开发出简洁、可测试的应用。学会操作流程，以及如何管理复杂性，是现代化JavaScript应用开发的基石。从长远来看，如果能处理好这两方面，结果就会很不一样。

《JavaScript Web应用开发》这本书会告诉你如何在前端开发中使用自动化技术，涵盖你所需要知道的一切，比如说如何避免重复的任务，如何使用简洁的工具监控生产版本，减少人为错误造成的损失。在这个过程中，自动化是关键。如果时至今日你还没有在工作流程中使用自动化技术，你活得就太辛苦了。如果一系列日常任务能使用一个命令完成的话，请听从Nico的建议，使用自动化技术，把节省下来的时间用在提升应用的代码质量上。

模块化至关重要，能协助我们构建可伸缩且可维护的应用。模块化不仅能确保应用的各个部分都能轻易地加以测试，容易编写文档，而且还能鼓励我们重用代码，并把精力集中在提高代码质量上。在这本书中，Nico熟练地示范了如何编写模块化的JavaScript组件，如何正确处理异步流，还介绍了足够你用来构建应用的客户端MVC知识。

系好安全带，调整好命令行，享受这段改进开发流程的旅程吧。

Addy Osmani

谷歌高级工程师，对开发者使用的工具充满激情

前　言

像这个领域中的大多数人一样，我一直着迷于解决问题。虽然寻找解决方案时痛苦不堪，但找到后却无比欢欣——有什么能比得上这样的过程呢！年轻时我特别喜欢玩策略游戏，例如国际象棋，我从孩童起就开始玩了。《星际争霸》这个实时策略游戏，我已经玩了10年。还有万智牌，一种集换式卡片游戏，可理解为介于扑克和国际象棋之间的游戏。这些游戏为我提供了很多解决问题的机会。

上小学时，我学会了Pascal和基本的Flash编程。我兴奋坏了，又接着学习了Visual Basic、PHP和C语言，并利用我对<marquee>和<blink>标签的充分掌握以及对MySQL的粗浅理解，开始开发网站。没有什么能阻挡我，而且对解决问题的渴望没有就此结束，我又开始玩游戏了。

《网络创世纪》（简称UO）是一款大型多人在线角色扮演游戏（简称MMORPG），和其他游戏一样，我也沉迷于这个游戏很多年。后来，我发现了一个UO服务器的开源实现，叫RunUO[①]，完全使用C#开发。我所在的RunUO服务器的管理员没有编程经验，他们逐渐开始信任我，让我修正一些小缺陷，我们通过邮件来来回回地发送源码。我着迷了。C#是一门美妙而富有表现力的语言，而且用来开发UO服务器的开源软件友好且诱人，甚至不需要使用IDE（也不用知道IDE是什么），因为服务器能动态编译脚本文件。基本上，只需要在一个文件中写10~15行代码，继承Dragon类，就能在龙头上添加一个吓人的泡状文本框；或者覆盖一个方法，就能让龙吐出更多火球。这门语言和它的句法一点都不难学，在玩玩乐乐中就能学会。

后来，一个朋友告诉我，我可以靠编写C#代码为生。他说："知道吗，真的有人愿意付费让你做这件事。"随后我又开始开发网站了，不过这一次我不是为了找乐子，也没有仅仅使用FrontPage和一堆<marquee>标签。可是，对我来说，仍像是在玩游戏。

几年前，我读了《程序员修炼之道》[②]这本书，受到一些触动。这本书给出了很多可靠的建议，我强烈推荐你也读读。书中有个观点对我影响比较深：作者鼓励我们走出自己的安乐窝，尝试一些我们计划去做但还没有做的事。那时，我的安乐窝是C#和ASP.NET，所以我决定尝试Node.js。对于在服务器端做JavaScript开发，这是一个真真切切的类Unix平台。就那时我围绕微软的开发经验来说，这无疑是个突破。

① RunUO的网站地址是http://runuo.com，不过这个项目已经停止维护了。

② Andrew Hunt和David Thomas合著的这本书是永恒的经典之作，你一定要认真读一下。

我从这次尝试中学到了大量知识，还搭建了一个博客[①]，记录我在这个过程中学到的各种知识。大概半年之后，我决定把我在C#设计上多年积累的经验写成一本关于JavaScript的书。我联系了Manning出版社，他们欣然接受了我的请求，并帮助我做头脑风暴，把初步想法变得明确从容、简单明了。

我花了很多时间和精力写这本书，表明了我对Web的热爱。这本书中包含一些关于应用设计和过程自动化的实用建议和最佳实践，能帮助你提升Web项目的质量。

① 我的博客名为“Pony Foo”，地址是http://ponyfoo.com。我写的文章涉及Web、性能、渐进增强和JavaScript。

关于本书

Web开发的增长速度异乎寻常，现在很难想象没有Web的世界会是什么样子。Web以其容错性而著称。在传统编程技术中，缺少一个分号、忘记关闭标签或者声明无效的属性都会导致严重的后果，但Web中却有所不同。在Web中可以犯错，但错误的生存空间越来越少。之所以出现这种二元现象，是因为现代的Web应用和以前相比，要复杂一个数量级。在Web发展初期，我们可能会使用JavaScript适当地小幅度修改网页，但在现在的Web中，整个网站都使用JavaScript驱动，在单个页面中渲染。

这是一本指南书，会告诉你如何在现代的环境中使用更好的方式做Web开发，就像使用其他语言做开发一样，编写出可维护的JavaScript应用。你将学习如何利用自动化技术取代容易出错的繁复过程，如何设计易于测试的模块化应用，以及如何测试应用。

过程自动化是整个开发过程中节省时间的关键所在。在开发环境中使用自动化技术能帮助我们把精力集中在思考问题、编写代码和调试上。自动化技术有助于确保每次存入版本控制系统中的代码能正常运行。准备把应用部署到生产环境时，使用自动化技术能节省时间，自动化技术能打包、简化资源文件、创建子图集表单，还能执行其他性能优化措施。部署时，自动化技术还能减少风险，自动完成复杂且容易出错的操作。很多书讨论的都是后端语言使用的过程和自动化技术，很难找到针对JavaScript应用的资料。

本书主要想表达的观点是要注重质量。使用自动化技术能搭建一个更好的应用构建环境，但光有自动化技术还不够，应用本身也要有质量意识。为此，本书涵盖了应用设计的指导方针，先介绍语言相关的注意事项，然后告诉你模块化的强大作用，再帮你厘清异步代码，教你开发客户端MVC应用，最后为JavaScript代码编写单元测试。

本书和其他讲解Web技术的书一样，依赖于特定版本的工具和框架，不过本书把代码库相关的问题和所需掌握的理论区分开了。这是种妥协的做法，因为Web开发领域使用的工具频繁变化，但工具的设计理念和操作过程的变化节奏要慢得多。我把这两方面分开了，希望这本书在未来几年仍有价值。

本书结构

本书包含两部分和四篇附录。第一部分专门介绍构建优先原则，告诉你这个原则是什么，以及如何辅助你的日常工作。这一部分详细说明过程自动化，涵盖日常开发和自动部署，还有持续

集成和持续部署包，共含4章。

- 第1章说明构建优先原则的核心法则，以及可以建立的不同过程和流程。然后介绍贯穿全书的应用设计指导方针，这些方针是后续内容的基础。
- 第2章介绍Grunt，以及如何使用Grunt制定构建流程。然后介绍几个可以使用Grunt轻易完成的构建任务。
- 第3章专门介绍环境和部署流程。你会发现不是所有环境都是一样的，应该学习在开发环境中如何权衡调试便利性和生产力。
- 第4章示范发布流程，还会讨论部署相关的话题。你会学到几个针对性能优化的构建任务，并探索如何自动部署。你还会学习把应用部署到生产环境后如何连接持续集成服务，以及如何监控应用。

第一部分主要介绍如何使用Grunt构建应用，附录C会教你如何选择最符合任务需求的构建工具。读完第一部分后，该读本书第二部分了。第二部分专门介绍如何管理应用设计过程中的复杂度。模块、MVC、异步代码流、测试和设计良好的API在现代的应用中都扮演着重要角色，这些话题在下面几章中讨论。

- 第5章主要介绍如何开发模块化的JavaScript应用。这一章首先说明模块的构成，以及如何设计模块化的应用，还会列出这么做的好处。随后，简要说明JavaScript语言的词法作用域和怪异的地方。然后，概览实现模块化的主要方式：RequireJS、CommonJS和即将到来的ES6模块系统。最后，介绍几个包管理方案，例如Bower和npm。
- 第6章介绍异步代码流。如果你曾陷入到回调之坑中，这一章可能会为你提供摆脱这一困境的方法。这一章讨论了处理异步代码流中复杂度的多种方式，分别为回调、Promise对象、事件和ES6的生成器。你还会学到如何在这些范式中正确处理错误。
- 第7章首先介绍MVC架构，然后将其应用到Web中。你会学习如何借助MVC分离关注点，使用Backbone开发富客户端应用。随后，你会学习Rendr，使用它在服务器端渲染Backbone视图，优化应用的性能和可访问性。
- 现在你的应用已经模块化，外观精美，而且易于维护，接下来在第8章自然就该使用不同的方式测试应用了。为此，我会介绍各种JavaScript测试工具，并传授使用这些工具测试小型组件的实践经验。然后，我们要为第7章使用MVC架构开发的应用编写测试。我们不仅要做单元测试，还会学习持续集成、外观测试和性能评估。
- 第9章是本书最后一章，专门介绍REST API设计。API供客户端与服务器交互，而且为我们在应用中所做的一切奠定基础。如果API复杂得令人费解，那么整个应用有可能也是如此。REST为API的设计给出了明确的指导方针，能确保API简单明了。最后，我们会介绍如何使用传统方式在客户端使用API。

你可以在读完正文后再阅读附录，不过，在你遇到问题时就及时阅读更能发挥附录的作用，因为附录可能会为你的疑问提供解答。在正文中，如果某处需要使用附录的内容补充，我会指出来。

- 附录A简要介绍Node.js和其使用的模块系统CommonJS。这个附录能帮你解决安装Node.js的问题，还会解答一些关于CommonJS工作方式的疑问。

- 附录B详细介绍Grunt。第一部分中的几章只说明了使用Grunt必备的知识，而这个附录则详细说明了Grunt的内部工作机制。如果你真想使用Grunt开发一个成熟的构建过程，这个附录能为你提供一些帮助。
- 附录C明确表明了本书和Grunt没有任何“联姻”，给出了两个替代工具——Gulp和npm run。这个附录分析了这三个工具各自的优缺点，让你自己决定哪个最符合你的需求。
- 附录D是一个JavaScript代码质量指南，列出了大量最佳实践，你可以选择该遵守哪些。我的目的不是强制你遵守这些指导方针，而是要让你明白，在开发团队中保持代码基的一致性是件好事。

代码约定和下载

所有源码都使用等宽字体表示，例如`fixed-size width font`，而且有时源码会放在一个有名称的代码清单中。很多代码清单中都有注解，用于体现重要的概念。本书的配套源码是开源的，公开托管在GitHub中，如果想下载，请访问github.com/buildfirst/buildfirst。这个在线仓库中的源码始终都是最新版。虽然书中给出的代码有限，但在仓库中都有很好的注释，如果遇到问题，我建议你看一下带注释的代码。

代码还可以从出版社的网站中下载，地址是www.manning.com/JavaScriptApplicationDesign。

作者在线

购买本书英文版的读者可以免费访问由Manning出版社维护的在线论坛，在这个论坛中你可以对本书发表评论、询问技术问题、从作者和其他用户那里得到帮助。要访问并订阅该论坛，请访问www.manning.com/JavaScriptApplicationDesign。这个页面介绍了注册后如何访问论坛、可以得到什么帮助以及在论坛中的行为准则。

Manning致力于为读者提供一个场所，让读者之间、读者和作者之间能进行有意义的对话。但我们并不强制作者参与，他们在论坛上的贡献是自愿而且不收费的。我们建议你尽量问作者一些有挑战性的问题，免得他失去参与的兴趣！

只要本书英文版仍然在售，读者就能从出版社的网站上访问作者在线论坛和之前讨论话题的存档。

关于封面

本书封面中的画像题为“1760年堪察加的冬季习俗”。堪察加半岛位于俄罗斯最东边，东临太平洋，西接鄂霍次克海。这幅画像出自1757年至1772年在伦敦出版的《古代和现代不同国家的服饰图集》，作者为托马斯·杰弗里斯。这本图集的扉页指出，这些图像都是手工上色的铜板雕刻，并使用阿拉伯树胶提色。托马斯·杰弗里斯（1719—1771年）被称为“地理学界的乔治三世国王”。他是一名英国制图师，在他那个年代是主要的地图供应商。他为政府和官方机构雕刻并印刷地图，还生产了各种各样的商业地图和地图册，尤其是北美洲的地图。作为一名制图师，他对在所测绘地方生活的本地居民的服饰产生了兴趣，他在这本四卷图集中出色地把这些服饰展示了出来。

迷恋遥远的国度，为了消遣而旅行，这在18世纪是相对较新的现象，因此这本图集十分受欢迎，它向游客和憧憬旅行的人介绍了其他国家的居民。杰弗里斯这几卷图集中的绘画充满多样性，在几个世纪以前就生动表现出了世界各国人民的独特个性。现在，人们的着装规范已经改变，地区和国家之间的多样性曾经是多么丰富，如今则在慢慢消逝。现在甚至很难区分不同大陆的居民。或许，从乐观的一面来看，虽然我们丢失了文化和视觉的多样性，但换来了更多样化的个人生活——或者说是更多样化且更充满智能和技术的乐趣生活。

今时今日，计算机图书层出不穷，Manning就以两个半世纪以前杰弗里斯这套书中多样性的民族服饰，来表达对计算机行业日新月异的发明与创造的赞美。

致　谢

如果在写作过程中没有大家的支持和忍耐，你的手中就不可能捧着这本书了。我只希望最值得感谢的人，也就是我的朋友和家人已经知道，我对你们的爱、理解和不断的安慰充满感激，这份感激之情无法用言语表明。

还有很多人直接或间接地为本书贡献了大量知识和想法。

JavaScript开源社区的成员见识不凡，相互鼓励，始终在作无私的贡献。他们让我见识到了更好的软件开发方式，这种方式不仅使协作成为可能，而且还积极鼓励协作。这些人中的大多数都通过传播Web知识、维护博客、分享经验和资源或教我知识，间接为社区作了贡献。有些人则开发了本书讨论的工具，直接作出贡献，这些人包括Addy Osmani、Chris Coyier、Guillermo Rauch、Harry Roberts、Ilya Grigorik、James Halliday、John-David Dalton、Mathias Bynens、Max Ogden、Mikeal Rogers、Paul Irish、Sindre Sorhus和T.J. Holowaychuk。

还有一些书籍和文章的作者影响了我，让我变成了更合适的教育工作者。这些人撰写的文章和分享的知识对我帮助巨大，使我确定了自己的职业发展方向。他们是Adam Wiggins、Alan Cooper、Andrew Hunt、Axel Rauschmayer、Brad Frost、Christian Heilmann、David Thomas、Donald Norman、Frederic Cambus、Frederick Brooks、Jeff Atwood、Jeremy Keith、Jon Bentley、Nicholas C. Zakas、Peter Cooper、Richard Feynmann、Steve Krug、Steve McConnell和Vitaly Friedman。

特别感谢Manning出版社的开发编辑Susan Conant。她让我充分发挥了最佳水平写作这本书，如果没有她，这本书会逊色很多。这是我的第一本书，是她一直领着我走完整个细致入微的写作过程。她以严格而温和的指导，帮我把众多想法写成了这本不会羞于出版的书。得益于她的帮助，我的写作水平大有长进，我特别感谢她。

在这方面帮助我的人不止她一个。Manning出版社的所有人都希望这本书能做到最好。出版人Marjan Bace，连同所有编辑，都应得到感谢。Valentin Crettaz和Deepak Vohra两位技术校对不仅帮我确保了代码示例是一致且有用的，还给我提供了很好的反馈。

还有一大帮不知道姓名的人愿意通读书稿，说出他们的想法，帮助改进这本书。感谢MEAP的读者们，感谢你们在“作者在线”论坛中发布勘误和评论。还要感谢在本书出版的各个阶段阅读本书的各位审稿人员：Alberto Chiesa、Carl Mosca、Dominic Pettifer、Gavin Whyte、Hans Donner、Ilias Ioannou、Jonas Bandi、Joseph White、Keith Webster、Matthew Merkes、Richard Harriman、Sandeep Kumar Patel、Stephen Wakely、Torsten Dinkheller和Trevor Saunders。

特别感谢为本书作序的Addy Osmani，以及其他每个参与本书出版的人。有些人可能没有直接按键输入内容，但他们在本书出版的过程中也扮演了重要角色，加快了这本书的面世进程。

目　录

第一部分　构建过程

第二部分 管理复杂度

Part 1

第一部分 构建过程

本书第一部分专门介绍构建过程，还会通过实例介绍Grunt。这一部分既有理论也有实践，目的是告诉你什么是构建过程，为什么以及如何使用构建过程。

第1章说明构建优先原则包含的两层意思：构建过程和应用复杂度管理。然后开始编写第一个构建任务：使用lint程序检查代码，避免有句法错误。

第2章专门介绍构建任务。你会了解组成一次构建的各项任务，如何配置任务，以及如何自己编写任务。针对每种情况，我们都会先讲理论，然后再使用Grunt编写实例。

第3章介绍如何配置应用的环境，而且要安全存储敏感信息。我们会说明搭建开发环境的流程，以及如何自动完成这些构建步骤。

第4章再介绍一些需要在发布应用时执行的任务，例如优化静态资源和管理文档。你会学到如何使用持续集成服务检查代码的质量。我们还会把应用部署到线上环境，让你实际体验一把。

第1章 构建优先

本章内容

- 现代应用设计面临的问题
- 什么是构建优先原则
- 构建过程
- 管理应用中的复杂度

使用正确的方式开发应用可能很难，我们要合理规划。我曾只用一个周末就开发出了应用，但应用设计得可能并不好。创建随时会扔掉的原型可以即兴发挥，但是开发一个可维护的应用则需要规划，要知道怎么把脑海中设想的功能组织在一起，甚至还要考虑到不久之后可能会添加的功能。我曾付出无数努力，但应用的前端还是差强人意。

后来我发现，后端服务通常都有专门的架构，专门用于规划、设计和概览这些服务，而且往往还不止一个架构，而是一整套。可是前端开发的情况却完全不同，前端开发者会先开发出一个可以运行的应用原型，然后运行这个原型，希望在生产环境中依然正常。前端开发同样需要规划架构，像后端开发一样去设计应用。

以前，我们会从网上复制一些代码片段，然后粘贴到页面中，就这样收工了。可是这样的日子早已过去，先把JavaScript代码搅和在一起，事后再做修改，不符合现代标准了。如今，JavaScript是开发的焦点，有很多框架和库可以选择，这些框架和库能帮助我们组织代码，我们不会再编写一整个庞大的应用了，更多的是编写小型组件。可维护性不是随意就能实现的，我们从开发应用伊始就要考虑可维护性，并在这个原则的指导下设计应用。设计应用时如果不考虑可维护性，随着功能的不断增加，应用就会像叠叠乐搭出的积木塔一样慢慢倾斜。

如果不考虑可维护性，最后根本无法再往这个塔上放任何积木。应用的代码会变得错综复杂，缺陷越来越难追查。重构就要中断产品开发，业务可经不起这样折腾。而且还要保持原有的发布周期，根本不能让积木塔倒下，所以我们只能妥协。

1.1 问题出现了

你可能想把一个新功能部署到生产环境，而且想自己动手部署。你要用多少步完成这次部

署？八步还是五步？为什么你要在部署这样的日常工作中冒险呢？部署应该和在本地开发应用一样，只需一步就行。

可惜事实并非如此。我以前会手动执行部署过程中的很多步骤，你是不是也是这样？当然，你一步就能编译好应用，或者可能会使用服务器端解释型语言，根本不用事先编译。如果以后需要把数据库更新到最新版本，你甚至可能会编写一个脚本执行升级操作，但还是要登入数据库服务器，上传这个脚本文件，然后自己动手更新数据库模式。

做得不错，数据库已经更新了，可是有地方出错了，应用抛出了错误。你看了下时间，应用已经下线超过10分钟了。这只是一次简单的升级啊，怎么会出错呢？你查看日志，发现原来是忘记把新变量添加到配置文件里了，真是太傻了。你立即加上了新变量，抱怨着这次与代码基的斗争。你忘记在部署前修改配置文件，在部署到生产环境前忘了更新配置。这种情况是不是听起来很熟悉？不要害怕，这种情况很常见，在很多应用中都存在。我们来看看下面这个危险的案例。

1.1.1 45 分钟内每秒损失 17 万美元

我敢肯定，一个严重问题导致损失几乎五亿美元的案例会让你打起精神。在骑士资本公司就发生过这样的事。[①]他们开发了一个新功能，让股票交易员参与一个叫“零售流动性计划”（Retail Liquidity Program，简称RLP）的项目中。RLP的目的是取代已经停用九年的“权力限定”（Power Peg，简称PP）功能。RLP的代码中重用了一个用来激活PP功能的标志，添加RLP时，他们把PP移除了，所以一切都正常运行着，至少他们认为是正常的。但是，当他们打开这个标志时，问题出现了。

他们在部署时没有采用正式的过程，而且只由一个技术人员手动执行。这个人忘记把代码改动部署到八个服务器中的某一个，因此，在这个服务器中，这个标志控制的是PP功能，而不是RLP功能。直到一星期后他们打开这个标志时才发现问题：他们在七个服务器中激活了RLP，却在最后一个服务器上激活了停用九年的PP功能。

这台服务器上处理的订单触发执行的是PP代码，而不是RLP。这样一来，发送到交易中心的订单类型是错误的。他们试图补救，但情况进一步恶化了，因为他们从已经部署了RLP的服务器中把RLP删除了。长话短说，他们在不到一小时的时间内损失了差不多4亿6千万美元。他们只要使用更正式的构建过程，就能避免公司的衰败。想到这一点，就会发现这整件事都是那么不可思议，不负责任，其实又应该是很容易避免的。当然，这是个极端案例，但明确表明了我的观点：自动化的过程能尽量避免人为错误，至少也能更早发现问题。

1.1.2 构建优先

我写这本书的目的是教你使用构建优先原则，在还未编写任何代码之前就做好设计，让应用的结构清晰，易于测试。你会学习过程自动化的知识，减少人为出错的可能性，避免重蹈骑士资本的覆辙。构建优先原则是设计结构清晰、易于测试的应用之基础，使用这一原则开发出来的应

① 关于骑士资本公司这次事件的详情，请访问http://bevacqua.io/bf/knight。