MANNING

# 反应式 Web应用开发

# Reactive Web Applications



[奥地利] 曼努埃尔·伯恩哈特（Manuel Bernhardt） 著
张卫滨 译

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

MANNING

# 反应式 Web 应用开发

# Reactive Web Applications


[奥地利] 曼努埃尔·伯恩哈特（Manuel Bernhardt） 著
张卫滨 译

人民邮电出版社
北京

**图书在版编目（CIP）数据**

反应式Web应用开发 /（奥）曼努埃尔·伯恩哈特（Manuel Bernhardt）著；张卫滨译. -- 北京：人民邮电出版社，2018.11
ISBN 978-7-115-48954-8

Ⅰ. ①反… Ⅱ. ①曼… ②张… Ⅲ. ①网页制作工具—程序设计 Ⅳ. ①TP393.092.2

中国版本图书馆CIP数据核字(2018)第221532号

## 版权声明



◆ 著　[奥地利] 曼努埃尔·伯恩哈特（Manuel Bernhardt）
译　张卫滨
责任编辑　吴晋瑜
责任印制　焦志炜

◆ 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
固安县铭成印刷有限公司印刷

◆ 开本：800×1000　1/16
印张：18.5
字数：393 千字　2018 年 11 月第 1 版
印数：1－2 400 册　2018 年 11 月河北第 1 次印刷

著作权合同登记号　图字：01-2016-9528 号

定价：69.00 元

**读者服务热线：(010)81055410　印装质量热线：(010)81055316**
**反盗版热线：(010)81055315**
**广告经营许可证：京东工商广登字 20170147 号**

# 内容提要

本书以 Play 框架为例阐述了反应式编程的理念以及在实际的编码中实践这些理念的方法，以实现更加灵活和高性能的 Web 应用程序。

本书共 11 章，分成三大部分。第一部分（第 1 章到第 4 章）主要介绍了反应式编程的基础理念，并讲解了函数式编程和 Play 框架的基础知识。第二部分（第 5 章到第 8 章）介绍了反应式 Web 编程的核心概念，如 Future 和 Actor，还讲解了将反应式的理念应用到用户界面层的方法。第三部分（第 9 章到第 11 章）介绍了反应式 Web 编程的高级主题，涵盖反应式流以及应用程序的部署和测试等内容。

本书适合 Java Web 程序开发人员和架构师阅读，尤其适合希望借助反应式技术提升系统性能的开发人员参考，还可以作为 Java 编程人员学习函数式编程理念的进阶读物。

# 推荐序

几年前，我编写的每个主要的 Web 应用都是分层且可信赖的执行模型，它们采用的是“一个请求对应一个线程”的方式。在几个小型应用中，我曾经用过某些形式的基于事件的 I/O，如果那时有人说采用这种模式进行通用的 Web 开发，那么我一定会付之一笑。在那个时代，行业中基本上还没有人听说过“反应式”这个词。

对 Web 本身而言，切换至反应式应用是巨大的架构变化，它以迅雷不及掩耳之势席卷了整个行业。我数年前认为不靠谱的技术，现在却每天都在使用，目前我是 Play 框架的领导开发者，这个框架就用了反应式技术。一个理念在短短的时间内就从模糊不清发展成主流的最佳实践，无数的 Web 开发人员都在问“什么是反应式？”也就不足为奇了。在这一点上，《反应式 Web 应用开发》一书很好地填充了这个空白。

Manuel 首先回答了“我们为什么需要反应式编程”的问题，其次他将反应式开发的理念用到了 Web 应用程序中，而此过程是基于 Play 框架、Akka 和反应式流构建的。读者将会看到很多具体的样例和练习，通过这样的学习能够对反应式 Web 应用如何架构、开发、测试和部署有深入的理解，然后读者就可以自行尝试了。

反应式应用的发展过程也是我们不断学习的过程。**反应式宣言**（Reactive Manifesto）本身从我的同事起草之后，也在短时间内经历了多次修订。我和 Manuel 一起参加过很多会议，在私下以及开源软件事务方面，我们都会经常交流，讨论在 Web 应用中如何实现反应式开发。我非常开心 Manuel 能够系统地总结出这么多 Web 应用开发的前沿最佳实践。如果读者需要构建应对高负载访问的软件，那么我相信本书针对 Web 应用开发所给出的实践经验能够让读者立于不败之地。

James Roper
Play 框架的领导开发者

# 序

我萌生撰写本书的念头始于 2014 年 4 月，当时我刚刚用 4 个月的时间协助一家客户重构了整个应用，在此过程中便用到了 Scala、Play 框架和 Akka，在此之前，这三项技术我已经用了好几年了。

已有的应用因面临两项主要的挑战而需要重构：一方面，整个应用的数据分散到两个独立的数据库系统、多个缓存以及一些外部云服务（如 Amazon EC2、YouTube、SoundCloud 和 Mixcloud）之中，这样几乎不可能保证数据实时和同步。另一方面，随着用户数量的不断增长，每次举办新活动时，请求的洪流总会把系统搞垮。更有意思的是，站点的重新发布不仅涉及迁移、重新整合、更新上百万用户的数据和上千万的数据条目，还要求在一个周末完成。

这个项目很好地代表了新一代 Web 应用的特点，在过去的几年中，这种类型的应用变得越发重要。反应式 Web 应用开发需要应对多种类型的请求，这种请求的数量有可能非常庞大，不仅需要管理并提供对大数据集的访问，还要与多个云服务进行实时通信。更复杂的是，这些任务需要克服各种不可避免的故障，尤其在网络环境中，它的复杂度会随之增加。将所有数据放到同一台计算机上或同一个数据中心的时代一去不返了，在这种场景下通常会隐藏真相和计算机网络复杂的本质。虽然反应式 Web 应用开发严重依赖于多样化和分布式的服务，但与此形成鲜明对比的是我们越来越无法草率地将错误展现给用户。如今，用户对错误的容忍度已经接近零。每个人都已经习惯可靠性巨头（如 Google 或 Facebook）所提供的服务，却完全不会关心工程构建和运维这些系统所面临的巨大技术挑战。

构建反应式 Web 应用并不是小菜一碟，它与最近几年的技术发展息息相关。反应式技术不仅将异步编程变为可能，还把故障处理作为首要处理的问题。Play 框架和 Akka 并发工具集这两项技术为构建反应式 Web 应用提供了坚实的基础。它们都使用了 Scala 编程语言所提供的强大的函数式编程理念，实现了异步和反应式编程。

本书旨在成为 Play 框架、Akka 和其他几项强大技术的指南，通过组合使用它们来

构建反应式 Web 应用。从某种程度上来说，我真希望几年前就写这样一本关于该技术栈的图书。希望它对你有用，也希望你在阅读本书时发现其中的乐趣！

曼努埃尔·伯恩哈特

# 前言

本书将带领读者学习通过 Scala 编程语言、Play 框架和 Akka 并发工具集构建反应式 Web 应用的方法。虽然 Play 目前是 JVM 上非常流行的 Web 框架，但是很少有项目能够充分发挥它的优势并采取必要的步骤将应用变成反应式的。这是因为这些步骤并非能那么轻而易举地实现，而且也不是开发人员的强项。与此类似，虽然很多开发人员都听说过 Akka 技术，但是并不知道如何将其运用到项目中。本书旨在改善这种现状，为读者展示如何在实际中使用这些技术并将它们组合起来。本书会介绍异步编程的理念、基于 Future 和 Actor 进行反应式编程的方法，还会阐述在实际项目中配置应用和集成其他的技术的方法。

## 读者对象

为了更加充分地理解本书的内容，读者应具备一定的编程经验，并至少熟练掌握一门现代编程语言，如 Java 或 C#。另外，读者应该掌握有关 Scala 的语法和核心理念，这样才能阅读本书的样例并完成练习。虽然关于函数式编程的知识并不是必需的，但是如果能够掌握，这将会是一项优势。附录 B 给出了一个参考资料列表，旨在帮助读者通过它们快速掌握 Scala 和函数式编程。

因为本书主要介绍构建 Web 应用的相关内容，所以我们假设读者已经掌握了 HTML 和 JavaScript 的基础知识，并且熟悉大多数现代 Web 应用框架所使用的模型-视图-控制器（Model-View- Controller，MVC）范式。

## 本书内容结构

本书的第一部分将阐述函数式编程的基本原理。基于此，我们就能构建异步应用。这一部分还会介绍 Play 框架的基础知识。

第 1 章不仅介绍了为什么需要反应式 Web 应用，还讨论了 Web 应用的架构是如何演化的，以及反应式 Web 应用的理念是如何形成的。

第 2 章会深入学习反应式 Web 应用的开发，不仅会设置必要的工具开始构建第一个反应式 Play 项目，还会为 Twitter 过滤器 API 构建异步流处理管道，从而使无数的 Tweet 通过 WebSocket 连接展现在浏览器上。学完本章的内容，你应该对如何编写反应式 Web 应用有更好的理解。

第 3 章介绍了函数式编程的基本理念。本章介绍了不可变性、函数和高阶函数，展示了使用这些概念操作不可变集合的方法，这种方式与以后操作异步值非常类似。

第 4 章快速且完整地介绍了 Play 框架。我们会构建一个基本的 Play 应用，手动创建每个文件以便让读者熟悉它的结构和配置。在此过程中，我们还会深入内部，看看 Play 的请求处理是如何真正实现反应式特性的。

本书的第二部分阐述了反应式 Web 应用的核心概念。

第 5 章介绍了 Future，它是操作和连接短期存活的异步计算过程的核心概念。首先，我们会看一下 Future 背后的理论；其次会了解如何使用 Future 设计业务逻辑，使应用实现异步功能并且能够容忍故障情况的出现。

第 6 章介绍了 Actor，它是为长期存活的异步计算过程建模的核心概念。我们将会看到 Akka 如何实现 Actor 模型并提供监管和恢复功能，以及如何用它来构建能够应对故障和流量突然飙升的应用。

第 7 章介绍了如何使用 Future 和 Actor 模型来处理无状态应用中的状态。在这种应用中，每个节点都可能在任意的时间消失或重新加入进来。它探索了如何使用 Play 框架集成传统的关系型数据库，同时还要保留反应式范式的优势。最后，本章还介绍了组合使用命令查询职责分离（Command and Query Responsibility Segregation，CQRS）和事件溯源（Event Sourcing），这是一种在反应式应用中进行大规模数据处理的模式。

第 8 章介绍了如何使用 Scala.js 构建响应性的用户界面。借助这个库，我们能够用 Scala 编写在浏览器中运行的代码。它展示了如何使用已有的 JavaScript 库（例如 AngularJS）以及如何将 Scala.js 尚未集成的任意库集成进来。本章还探讨了在客户端采取什么样的预防措施，从而保证应用的反应式特性不遭到破坏。

第三部分阐述了反应式 Web 应用的高级话题。

第 9 章介绍了**反应式流**（reactive stream），这是在 JVM 上进行异步流操作并且能够容忍故障状况出现的新标准。你将会看到如何通过使用这项标准的 Akka 从而实现（即 Akka stream）访问、切割和过滤 Twitter 的流式 API。

第 10 章介绍了采用不同方式进行反应式 Play 应用的部署。我们不仅会看到如何使用 Jenkins 持续集成服务器和 Docker 进行本地环境的部署，还会看到如何使用托管的部署服务 Clever Cloud 来部署简单的反应式应用。

第 11 章介绍了反应式应用要测试哪些方面的内容。对于测试反应式应用与非反应式应用的差异，我们将关注点放到了负载处理和故障处理方面。在本章中，你将看到如

何用 Clever Cloud 的自动扩展功能来处理不断增长的服务负载。

## 本书约定

本书代码清单的源码可以通过 GitHub 获取。大多数章节都包含一个可执行的应用，应用的最终源码（包括相关章节所有练习的解决方案）可以在一个单独的目录中找到，它随时可以执行（读者需要自行配置 Twitter API 凭证和数据库）。

除了完整的应用，每章中的代码清单都可以在 GitHub 相应的 listings 目录中找到。

## 作者在线

购买本书后，读者就可以免费访问 Manning 出版社提供的在线论坛，通过它你不仅可以给本书写评论，还可以问一些技术问题并能得到作者和其他用户的帮助。

Manning 承诺为读者提供一个交流平台，以便读者之间以及读者和作者之间可以进行有意义的交流。

只要本书在销，读者就可以访问作者在线论坛和以前讨论的存档资料。

## 作者简介

Manuel Bernhardt 是一位很有热情的工程师、作者、演讲者和咨询师，他对构建和运维网络应用方面的科学有着强烈的兴趣。从 2008 年开始，他指导并培训企业团队将应用转移到分布式计算架构。最近几年里，他着重关注包含反应式应用架构的生产型系统，而在这个过程中，主要用的是 Scala、Play 框架和 Akka。

Manuel 喜欢旅行，经常会在国际会议上演讲。他住在维也纳，是维也纳 Scala 用户组的联合发起者。除了思考、谈论和摆弄计算机，他喜欢将时光用在陪伴家人上，和他们一起跑步、潜水和阅读。读者可以在领英官网上了解 Manuel 的最新动态。

## 封面插图简介

《反应式 Web 应用开发》的封面图片是“Chamanne Bratsquienne”。这幅图片出自 Jacques Grasset de Saint-Sauveur（1757—1810）编写的各国着装风格合辑，其书名为“Costumes de Différents Pays”，于 1797 年在法国发行。其中的每一幅插图都绘制精美且手工着色。合辑中丰富的图画生动描绘了 200 年前世界上各个城镇和地区的独特魅力。由于地域差异，不同地区的人说着不同的语言或方言。在街区或乡村中，根据人们的着装能够判断他们住在哪里、他们从事何种职业或地位是什么样的。

但从那以后，服装风格便发生了改变，颇具地方特色的多样性开始淡化。现在，有时很难说一个州的居民和其他州的居民有什么不同，更不用说不同的城镇、地区或国家

了。抑或说我们以牺牲原来文化和视觉上的多样性为代价换来了个人风格的多变性，当然，我们得到了更为多样化和快节奏的科技生活。

现在，我们很难识别两本计算机图书之间的差异，Manning 创造性地将两个世纪前地区生活的多样性转移到了计算机业务的图书封面上，通过 Grasset de Saint-Sauveur 的图片将我们带回到那个时代。

# 致谢

如果没有众人的支持、灼见、灵感、鼓励和反馈，就不会有本书的面世。

首先感谢我的朋友 Peter Brachwitz，围绕本书所描述的技术，我们进行了多次有趣的讨论，他还分享了他的故事。这些会谈不断给我灵感，并为本书的样例提供了原材料。我们这样的交往在未来也应该持续下去！

还要感谢 Rafael Cordones 于 2013 年在维也纳开创了 Scala 社区，同时感谢维也纳 Scala 用户组所有成员举办的令人愉快的聚会。

本书样例用到了很多技术和库，如果没有这些技术的开发人员的协助，那么我对它们的使用和阐述将会大打折扣。来自 Akka 团队的 Konrad Malawski 帮助我提升了本书的质量，他指出了其中的错误并让我见识到来自 Akka 团队的最佳实践。Lukas Eder 是 jOOQ 库的发明者，他不但对我提出的问题快速答复，而且对使用数据库相关的事宜提供了有价值的反馈。Sébastien Doeraene 是 Scala.js 的发明者，他始终热心解答我关于技术的问题并提供了优雅的解决方案。还要感谢 sbt-play-scalajs 库的作者 Vincent Munier 以及 scalajs-angulate 库的作者 Johannes Kastner。感谢 play-angular-require-seed 库的作者 Marius Soutier，感谢他的反馈，他还提供了在 Play 中配置 JavaScript 优化过程的真知灼见。同时感谢 Clément Delafargue 回答了我所有关于 Clever Cloud 的问题，并允许我在书中访问当时尚未发布的 API。最后，特别感谢 James Roper（Play 框架的领导开发者），他不仅耐心地回答我所有的问题，帮助我分析影响本书相关技术的演化，还为本书撰写推荐序并为之背书。

感谢 Manning 出版社所有帮助过我的工作人员：Karen Miller 是我的开发编辑，她耐心地逐章审阅本书，并且不介意我在编写时采用了一种创新式的英语表述；Bert Bates 教会我如何组织自己的想法，使其编写出来更加对读者有所帮助；策划编辑 Mike Stephens，他建议我将本书的范围扩展至反应式 Web 应用开发，还有 Candace Gillhoolley，将这些文字印刷出来并不断宣传本书。最后，我还要扩大感谢将本书生产出来的每个人，他们是文字编辑 Andy Carroll 和 Benjamin Berg、校对 Katie Tennant、执行编辑 David

Novak、制作者 Janet Vail 以及幕后所有其他的人。

如果没有审校者花费时间阅读本书早期版本的章节，并提供何处可以改善的反馈，那么本书不可能达到如今的质量水准。在此感谢 Antonio Magnaghi、Arsen Kudla、Changgeng Li、Christian Papauschek、Cole Davisson、David Pardo、David Torrubia、Erim Erturk、Jeff Smith、Jim Amrhein、Kevin Liao、Narayanan Jayaratchagan、Nhu Nguyen、Pat Wanjau、Ronald Cranston、Sergio Martinez、Sietse de Kaper、Steve Chaloner、Thomas Peklak、Unnikrishnan Kumar、Vladimir Kuptsov、Wil Moore III、William E. Wheeler 和 Yuri Kushch。还要感谢早期访问项目（Early Access Program）版本的所有读者，他们在 Manning Author Forum 上给出评论，指出源码中的错误并告诉我何处无法运行：如果没有你们，那么让这些样例顺利运行起来要困难得多。

最后，感谢 Veronika，我的朋友、伙伴和妻子，感谢你的支持、耐心、理解和爱。如果没有你的帮助，那么本书以及我的很多其他项目都是无法实现的。

# 资源与支持

本书由异步社区出品，社区（https://www.epubit.com/）为您提供相关资源和后续服务。

## 配套资源

本书提供如下资源：

- 本书部分源代码；

要获得以上配套资源，请在异步社区本书页面中单击 配套资源，跳转到下载界面，按提示进行操作即可。注意：为保证购书读者的权益，该操作会给出相关提示，要求输入提取码进行验证。

如果您是教师，希望获得教学配套资源，请在社区本书页面中直接联系本书的责任编辑。

## 提交勘误

作者和编辑尽最大努力来确保书中内容的准确性，但难免会存在疏漏。欢迎您将发现的问题反馈给我们，帮助我们提升图书的质量。

当您发现错误时，请登录异步社区，按书名搜索，进入本书页面，单击“提交勘误”，输入勘误信息，然后单击“提交”按钮即可。本书的作者和编辑会对您所提交的勘误进行审核，确认并接受后，将赠予您异步社区的 100 积分。积分可用于在异步社区兑换优惠券、样书或奖品。

## 扫码关注本书

扫描下方二维码，您将会在异步社区微信服务号中看到本书信息及相关的服务提示。

## 与我们联系

我们的联系邮箱是 contact@epubit.com.cn。

如果您对本书有任何疑问或建议，请您发邮件给我们，并请在邮件标题中注明本书书名，以便我们更高效地做出反馈。

如果您有兴趣出版图书、录制教学视频，或者参与图书翻译、技术审校等工作，可以发邮件给我们；有意出版图书的作者也可以到异步社区在线提交投稿（直接访问 www.epubit.com/selfpublish/submission 即可）。

如果您是学校、培训机构或企业，想批量购买本书或异步社区出版的其他图书，也可以发邮件给我们。

如果您在网上发现有针对异步社区出品图书的各种形式的盗版行为，包括对图书全部或部分内容的非授权传播，请您将怀疑有侵权行为的链接发邮件给我们。您的这一举动是对作者权益的保护，也是我们持续为您提供有价值的内容的动力之源。

## 关于异步社区和异步图书

**异步社区**是人民邮电出版社旗下 IT 专业图书社区，致力于出版精品 IT 技术图书和相关学习产品，为作译者提供优质出版服务。异步社区创办于 2015 年 8 月，提供大量精品 IT 技术图书和电子书，以及高品质技术文章和视频课程。更多详情请访问异步社区官网 https://www.epubit.com。

**异步图书**是由异步社区编辑团队策划出版的精品 IT 专业图书的品牌，依托于人民邮电出版社近 30 年的计算机图书出版积累和专业编辑团队，相关图书在封面上印有异步图书的 LOGO。异步图书的出版领域包括软件开发、大数据、AI、测试、前端、网络技术等。

异步社区

微信服务号

# 目录

## 第一部分　反应式 Web 应用起步

# 第二部分 核心概念