Web开发经典丛书

Pro MERN Stack

MERN全栈开发

使用Mongo Express React和Node

[美] Vasan Subramanian 著
杜伟 柴晓伟 涂曙光 译

清华大学出版社

Web 开发经典丛书

MERN 全栈开发

使用 Mongo Express React 和 Node

[美] Vasan Subramanian 著

杜伟 柴晓伟 涂曙光 译

清华大学出版社

北 京

Vasan Subramanian
Pro MERN Stack
EISBN：978-1-4842-2652-0

北京市版权局著作权合同登记号　图字：01-2017-5754

图书在版编目(CIP)数据

MERN 全栈开发：使用 Mongo Express React 和 Node /(美)瓦萨尼•苏布拉玛尼安(Vasan Subramanian)著；杜伟，柴晓伟，涂曙光　译. —北京：清华大学出版社，2018
(Web 开发经典丛书)
书名原文：Pro MERN Stack
ISBN 978-7-302-49152-1

Ⅰ. ①M…　Ⅱ. ①瓦…　②杜…　③柴…　④涂…　Ⅲ. ①JAVA 语言－程序设计　Ⅳ. ①TP312.8

中国版本图书馆 CIP 数据核字(2017)第 322391 号

责任编辑：王　军　于　平
封面设计：孔祥峰
版式设计：思创景点
责任校对：牛艳敏
责任印制：沈　露

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：三河市君旺印务有限公司
经　　销：全国新华书店
开　　本：148mm×210mm　　印　　张：12.25　　字　　数：330 千字
版　　次：2018 年 1 月第 1 版　　印　　次：2018 年 1 月第 1 次印刷
印　　数：1～3000
定　　价：59.80 元

产品编号：076425-01

译者序

当我第一次接触 Web 开发领域时，当时最热门的 Web 网站开发技术是使用 C 或 Perl 语言在服务器上编写 CGI(Common Gateway Interface)程序，来处理客户端发起的 HTTP 请求。现在我虽然已经全然忘记了 Perl 语言的语法，但是仍然还依稀记得当年在使用这个强大而“变态”的语言时的感受。随着 Web 的迅速发展，ASP、PHP、JSP、ASP.NET 出现在我们的眼前，它们各具特色，为不同技术背景的开发人员进入 Web 开发领域提供了所需的武器。

当 Web 开发技术日渐成熟，各种最佳实践和模式逐渐被总结和沉淀下来，“技术栈”这个术语开始出现。技术栈通常指的是开发一个完整的 Web 应用程序时所需的特定工具、库、框架的组合。印象中，第一个最热门的 Web 开发技术栈是 LAMP(Linux、Apache、MySQL 和 PHP)，随着前端在 Web 开发领域中所占的比例越来越大，以及 Node.js 的流行，MEAN 技术栈(MongoDB、Express、AngularJS 和 Node.js)逐渐异军突起，影响力越来越大。由于 MEAN 技术栈的前后端全都使用了 JavaScript 语言，这也使得“全栈工程师”这个称呼开始流行起来。

本书所讲述的 MERN 技术栈，和 MEAN 只有一“字母”之差。MERN 将 MEAN 中的 AngularJS 更换成这两年明显更受欢迎、使用更加广泛的 React，使得自己相比 MEAN 更接地气，更适合有志成为“全栈工程师”的 Web 开发人员阅读。

在一个开放、开源的时代，如果一本讲述开发技术的书籍同时也能让你有机会了解和熟悉 GitHub，那显然是极好的。在阅读本书以及完成本书中所包含练习的过程中，你很自然地就会学习到如何寻找高质量的 GitHub 开源项目，并将它们应用到你的应用程序中。

祝您享受阅读本书的全过程，并能收获良多！

最后是译者的几则重要声明：

- 本书部分重要段落是由译者柴晓伟的女儿 Momo 完成的。谢谢 Momo。
- 译者杜伟的爱猫 Yuki 对本书任何章节都没有贡献。
- 译者涂曙光感谢 Phoebe 在翻译工作中所给予的所有支持。

译者(杜伟/柴晓伟/涂曙光)

作者简介

Vasan Subramanian 使用过各种各样的编程语言，从 8085 上手工编写 8 位机的汇编代码，一直到 AWS Lambda。他热衷于通过软件解决问题，更喜欢寻找合适的技术组合，帮助软件开发团队提高效率。他在 Corel、Wipro、Barracuda Networks 软件公司学习编程，从事程序员工作的同时，也在这些公司中担任团队负责人。

Vasan 就读于印度理工学院马德拉斯校区(IIT Madras)和印度管理学院班加罗尔分校(IIM Bangalore)。他目前在 Accel 公司担任 CTO，为创业公司提供各种技术指导。除了提供指导、编写程序(当然还有写书！)之外，Vasan 也是半程马拉松爱好者，还参加五人制足球比赛。你可以通过 vasan.promern@gmail.com 联系他，欢迎赞扬、批评，或是介于这两者之间的一切意见。

目　录

第 1 章

引 言

现今的 Web 应用程序开发和往日相比已然大不相同。今时今日可以选择的技术如此之多，使得刚入门的初学者经常会面对这么多的选择而感到无所适从。可供选择的不仅仅是整个技术栈(开发时所使用的技术与工具集)，就连开发过程中所使用的工具都面临同样的选择难题。本书提出了一种名为 MERN 的技术栈，并讲解如何使用这个优秀的技术栈开发一个完整的 Web 应用程序，读者也将通过本书了解 MERN 技术栈的方方面面。

在第 1 章，我将首先大致介绍一下组成 MERN 技术栈的各项技术。我不会在这一章就开始讲述各个技术的细节，或是展示任何示例，相反，我会从比较高的角度来纵览 MERN 所包含的各种概念。本章的重点就在于解释这些概念，以及它们为什么会让 MERN 成为你构建下一个 Web 应用程序的首选技术。

1.1 MERN 是什么

任何一个 Web 应用程序都是由多项技术构建而成。这些技术的组合就被称为一个“技术栈”。技术栈这个词最开始是因为 LAMP 技术栈

而开始流行，LAMP 技术栈指的是 Linux、Apache、MySQL 和 PHP 这几个开源组件的首字母缩写。随着 Web 开发方式的逐渐成熟，以及用户交互方式逐渐走向了前端，单页应用程序(Single Page Application，SPA)逐渐流行。SPA 是 Web 应用程序的一个类别，它避免了在需要显示新内容时刷新整个 Web 页面，当需要刷新页面时，SPA 会向服务器发出轻量级的请求，然后使用返回的数据或代码段更新 Web 页面。相比过去刷新整个页面的方式，SPA 的这种运作方式要好得多。由于 SPA 应用程序的大部分工作都是在客户端完成的，因此它的流行也导致了前端框架的崛起。虽说看起来似乎关系不大，但伴随着 SPA 的流行，NoSQL 数据库也逐渐开始流行。

MEAN(MongoDB、Express、AngularJS、Node.js)技术栈是专门针对 SPA 和 NoSQL 热潮而发展起来的早期开源技术栈之一。MEAN 技术栈中的 AngularJS 是一个基于模型-视图-控制器(Model-View-Controller，MVC)设计模式的前端框架。MongoDB 是一个流行的数据存储 NoSQL 数据库。Node.js 是一个服务器端 JavaScript 运行环境，而 Express 是一个构建于 Node.js 之上的 Web 服务器。MEAN 技术栈可以说是迄今为止最流行的 Web 应用程序技术栈。

React 是一个由 Facebook 创建的前端框架，虽说它的出现并非直接针对 AngularJS，但毫无疑问已经获得了巨大的关注，并成为 AngularJS 的一名接棒者。它将 MEAN 技术栈中的“A”替换成“R”，于是这就给了我们一个名为 MERN 的技术栈。之所以说 React 并非直接针对 AngularJS，是因为前者并未提供一个完整的 MVC 框架。React 是一个用来构建用户界面的 JavaScript 库，从这个角度看，它只涵盖了整个 MVC 中视图(View)的部分。

虽说我们已经逐个确定了用来构成这个技术栈的各项技术，但是仅仅靠它们还不足以创建一个完整的 Web 应用程序。在开发过程中，还需要使用其他的工具，React 也需要一些与之配合的其他 JavaScript 库。本书会讲述所有这些内容，包括如何基于 MERN 技术栈来构建一个完整的 Web 应用程序，以及如何使用其他辅助性的工具来简化开发流程。

1.2 本书的目标读者

开发过 Web 应用程序，但不熟悉 MERN 技术栈的开发人员和架构师，可以通过本书学习 MERN 技术栈。在阅读本书之前，读者需要对 Web 应用程序的运行原理有所了解。我们假设本书的读者已经了解了 HTML 和 CSS 的基础知识。如果你还熟悉版本控制工具 git 的使用，那就更好了；你可以通过复制包含了本书中所有源代码的 git 代码库，来尝试使用所有这些代码，并通过签出一个分支，亲手实践本书中所讲述的每一个开发步骤。

如果你已经确定了在新的应用程序中将要使用 MERN 技术栈，那么本书可以帮助你快速入门。即使你暂时还没有对技术选择做出决定，阅读本书也可以让你了解 MERN 技术栈，并帮助你在未来更好地做出技术决策。你将学到的最重要的事情就是，如何将多项技术组合在一起，以共同创建一个完整的、有用的 Web 应用程序。当合上本书的封底时，你将成长为一名合格的 MERN 全栈工程师或架构师。

1.3 本书组织结构

虽然本书的重点在于教会你如何构建一个完整的 Web 应用程序，但本书的大半部分都将围绕着 React 进行。这么做的原因是因为，对于大部分的现代 Web 应用程序而言，前端代码是它最主要的组成部分。在 MERN 技术栈中，React 就是负责前端的组件。

本书的写作风格比较偏向于教程。换言之，只有亲手实践编写代码并完成练习，才能从阅读本书中收获满满。本书中包含了许多代码清单(这些代码也同时位于一个 GitHub 代码库中：https://github.com/vasansr/pro-mern-stack)。也可扫描封底二维码获取代码清单。我希望，你不要仅仅复制粘贴这些代码然后运行，而是亲手在键盘上逐行编写这些代码。我发现在学习的过程中，亲自编写代码非常重要。代码中有一些细枝末节，例如所使用的引号的类型(单引号还是双引号)，有时

都会严重影响到代码的运行。当你亲自编写代码时，你对这些细枝末节的掌握程度会比仅仅阅读它们好得多。只有在你自己编写代码时发生卡壳的状况，想要将自己的代码与我的代码进行比较时，才需要将 GitHub 上的代码库复制到本地，由于代码库中的代码已经被测试过并确保它们的正确性，因此你可以将它们与你的代码进行对比，以定位出自己代码中的错误。不要从本书的电子版中复制代码再粘贴到开发 IDE 中直接使用。因为本书排版后的样式并不一定能保证电子版中代码的正确性，只有 GitHub 代码库中的代码才是唯一经过测试的版本。

每当对代码进行一次可以独立测试的修改时，我都会添加一个 checkpoint(本质上就是一个 git 分支)，这样你就可以在线查看两个 checkpoint 之间的 diff(差异)。在代码库的首页(README 文件)上列出了所有 checkpoint 和 checkpoint 之间 diff 的链接。你会发现，首页上的这些信息，要比直接查看整个源代码或是查看本书中的代码清单更有帮助。GitHub 上 diff 的详细程度要远远强过书中的文字描述。

相较于让每个章节都覆盖一个不同的主题或技术，本书将会采用一种更有实效的“提出问题-解决问题”讲述方式。在本书结尾，你会得到一个可工作的完整应用程序。但是在一开始，你会首先创建一个简单得像 Hello World 那样的小例子。然后就像一个真正的项目一样，你会不断地向应用程序添加更多功能。在你添加功能的过程中，会遇到一项又一项的开发任务。对于每一个任务，我都会介绍完成它所需要的概念和技能，然后再对这些概念和技能详加解释。因此，你会发现每个章节都不会只单纯地讲解一个主题或技术；相反，每个章节都会包含一组你在构建应用程序时所需要达成的目标。你将使用不同的技术和工具来完成每个章节的目标。

本书包含了许多练习，它们能够启发你思考，或是需要你去互联网上阅读一些相关的文档。这些练习能让你知道应该去哪里了解本书中没有涵盖的知识点(例如某些非常高深的主题或 API)。

通过本书的学习所最终构建完成的应用程序是一个问题跟踪应用程序。这是一个大多数开发人员都能理解的应用场景，而且类似于任何一个企业应用，它也将拥有许多属性和需求。对于这样一个应用程

序，通常都将它描述为一个“CRUD”应用程序(CRUD 的意思是指对数据库记录进行 Create/Read/Update/Delete 操作)。

1.4 格式约定

本书中所使用的许多格式都遵循了常见的范例，所以我不打算将它们一一列举出来。但是，由于代码的显示格式并没有一个默认的统一规范，所以我将列举出本书中使用的此类格式约定。

每一章都由多个小节所组成，每一小节都专注于构建和修改一组特定的代码，这组代码位于整个可工作的应用程序中，可以被单独测试。每个小节可能有多个代码清单，它们不一定可以被单独测试。每个小节都在 GitHub 代码库(www.apress.com)中有一个对应的入口，在那里可以看到小节结束时整个应用的完整源代码，以及当前小节和上一个小节源代码之间的差异。你会发现这个代码差异视图对于识别每个小节对应用所造成的影响是非常有用的。

所有对代码的修改都会显示在小节的代码清单中，但是它们的精确性并没有百分之百的保证。可信赖且可运行的代码位于 GitHub 代码库中。代码库中的有些代码甚至是在本书定稿之后才完成最后的修改，所以来不及反映在书中。所有的代码清单都有一个清单标题，里面包含被修改或创建的代码文件名称。

如果一段代码清单完整地包含了一个文件、类、函数或对象，那么它就是一段完整的代码清单。完整代码清单还可能会包含两个或多个类、函数、对象，但是不会包含多个文件。如果代码清单中的实体并非是连续定义的，我会使用省略号标识出未被修改的代码片段。

代码清单 1-1 演示了一个完整的代码清单，它包含整个文件的完整内容。

代码清单 1-1 server.js：Express Server

```
const express = require('express');
```

```
const app = express();
app.use(express.static('static'));

app.listen(3000, function () {
    console.log('App started on port 3000');
});
```

与之对应的，一段不完整代码清单则不会列出整个文件、函数或对象的完整代码。它会以省略号开始和结束，在中间部分也可能会使用省略号跳过未被修改的代码。只要有可能，就会突出显示对代码所进行的实际修改。被修改的代码会使用粗体进行标识，而未被修改的代码则显示成正常的字体。代码清单 1-2 演示了一个包含一些微小修改的不完整代码清单。

代码清单 1-2 package.json: 添加进行代码转换的脚本

```
...
  "scripts": {
    "compile": "babel src --presets react,es2015 --out-dir static",
    "watch": "babel src --presets react,es2015 --out-dir static --watch",
    "test": "echo \"Error: no test specified\" && exit 1"
  },
...
```

被删除的代码使用删除线进行标识，如代码清单 1-3 所示。

代码清单 1-3 index.html: 对 Script 名称与类型的修改

```
...
<script
    src="https://cdnjs.cloudflare.com/ajax/libs/babel-core/5.8.23/browser.
min.js"></script>
...
```

正文中也会用到一些代码块，以针对代码所做的修改进行讨论。这些代码块通常是从代码清单中摘出来的重复性内容。在这种情况下，就不会再专门去定义一个新的代码清单，而且通常这种代码块也只有一两行的长度。下面就是一个代码块的例子，它们来自于一个代码清单，其中有一个词被重点标识出来：

```
...
const contentNode = ...
...
```