应用技术

利用 Node.js 开发前后端分离的系统——以图书馆地方文献系统为例

刘红卫
（泰达图书馆档案馆　天津 300457）

摘　要：Web 应用开发模式的发展趋势是前后端分离。以图书馆地方文献系统的开发为例，介绍了前后端分离模式的系统开发实践，主要包括前后端分离的设计、应用 RESTful 风格制定 API 接口规范、前后端的实现过程以及系统的部署方法，最后总结了前后端分离开发模式的优势并列举了今后应用探索的着眼点。

关键词：Node.js　前后端分离　Web 应用　开发模式
中图分类号：TP311.52　　**文献标志码**：A　　**文章编号**：1006-8945(2018)07-0067-04

Using Node.js to Develop the Front and Back Ends Separation System: Taking the Library's Local Document System as an Example

LIU Hongwei
（TEDA Library & Archives，Tianjin 300457，China）

Abstract：The trend in Web application development mode is the separation of the front and back ends. The development of library local document system adopts the front and back ends separation mode. The system development practice includes the design of front and back end separation，the application of RESTful style to make API interface specification，the implementation process of the front and back ends，and the system deployment method. In the end，the advantages of the development mode of the front and back ends separation were summarized and furture orientation was forecasted.

Key words：Node. js；the front and back ends separation；Web application；development mode

0　引　言

如今移动互联网高速发展，多种终端访问对 Web 应用开发提出了更高的要求。用户需求不断变化，Web 应用能够自适应不同类型终端，不再单纯满足传统浏览器端的体验。用户需求的变化提高了 Web 开发难度，而针对不同终端开发不同版本应用的解决方法，并不是有效的开发方式，不仅需要额外的开发人员，还会带来成本增加、开发时间拖长等影响[1]。前后端分离是多终端 Web 应用开发的可行方案，将后端保持不变，前端根据用户的需求和体验要求进行多终端定制开发，这样只需要改变前端代码，后端保持不变或微调，即可满足用户多终端需求，节省开发时间，降低开发成本，提高开发效率。

综上所述，Web 应用开发的前后端分离模式是大势所趋。

1　图书馆地方文献系统概况

传承历史文明，服务现代化的物质文明和精神文明建设，搜集、整理和保存地方文献，历来都是公共图书馆的基本任务之一[2]。地方文献具有鲜明的区域性和实用性特点，在公共图书馆评估工作中，对地方文献的馆藏建设、开发利用都提出了相关要求，并占有较高分值。

笔者所在的图书馆在几年前建立了地方文献系统，并将拥有版权许可的文献进行数字化，读者可以在线检索、阅读，获得地方文献排架信息，文献类型包括书、报、刊、照片、图片以及音视频等。该系统是由第三方开发，运行在 Windows Server 系统下采用 Asp.net 开发，已运行多年，现在经常出现系统死机状态，需要重启服务器才能恢复服务，读者意见很大。由于该系统是早期产品，厂商不再提供支持，当时的

收稿日期：2018-06-04

开发人员已经跳槽，现在无人能解决这个问题。

为了彻底解决系统中存在的上述问题，我们仔细研究了软件目录和结构，确定地方文献的对象数据文件存于一定规则的目录中，并且保存完整，元数据保存在 KBase 数据库中，可以导出文本文件。于是，决定自行开发一套新的地方文献系统，既能实现旧系统的所有功能，又能实现响应式设计在多终端上使用。考虑到用户将来对多终端界面需求的变化以及开发人员少的现状，新系统的开发模式决定采用前后端分离模式。

2　前后端分离系统开发实践

2.1　前后端分离系统设计

2.1.1　软件开发平台的选择

Node.js 基于 Chrome V8 引擎的 JavaScript 运行，使用高效、轻量级的事件驱动、非阻塞 I/O 模型，它的包生态系统 npm 是目前世界上最大的开源库生态系统[3]。Node.js 使 JavaScript 不仅可以应用在浏览器端，也可以运行在服务器端。目前，Node.js 凭借其优秀的性能受到全球各大公司的重视，如 eBay、Microsoft、PayPal、Uber、Yahoo 等，国内阿里巴巴、百度、腾讯等也在很多项目中应用，可见 Node. js 的发展已经成熟，它能快速创建大规模的网络应用，处理高吞吐量的实时连接。Node.js 有 Windows、Linux、macOS、SunOS、AIX 等系统平台版本，具有良好的跨平台可移植性，可以在 Windows 上开发，然后部署到 Linux 等其他系统上。

图书馆能承担系统开发的人员很少，平日开发工作基本局限于页面程序的改写和功能提升上，大家都对 JavaScript 脚本语言比较熟悉。针对此情况，结合 Node.js 可用于全栈开发及诸多特性和优点，在前后端分离系统的开发管理方面也比较方便，为此选择了 Node.js 为软件开发平台。

2.1.2　前后端分离系统设计

如图 1 所示，在地方文献系统前后端分离架构设计中，前后端分别运行在不同的 Node.js 创建的 Web Server 上，前端负责接收用户的请求、相关验证和数据展示效果，后端负责按照约定好的 API 请求处理业务逻辑、访问数据库、处理数据，并按约定的数据格式向前端返回相关数据（JSON 格式），包括请求 API 的时候进行相关验证。前后端之间通过 HTTP 请求进行交互，前端获取数据后，对页面进行组装和渲染，并将最终生成的页面返回浏览器。

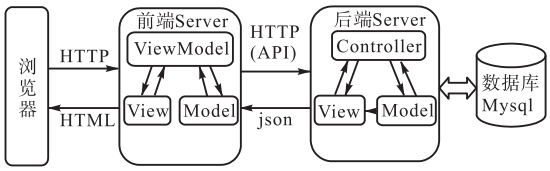

图 1　图书馆地方文献系统前后端分离架构图

Fig.1　Back-and-rear separation architecture of the local library documentation system

其前端采用 MVVM（Model-View-View Model）模式，它是在 MVC（Model-View-Controller）模式基础上改进的开发模式，它与 MVC 模式相同的是把视图和数据分离，不同的是引入 ViewModel 替换掉 Controller 来完成视图和数据的双向绑定，通过自动的方式完成大部分数据处理工作，从而降低了前端开发的复杂度。

2.2　API接口规范定义

在进行前后端分别开发之前，需要商定好前后端交互所需的 API，包括通讯接口、验证策略以及数据格式等，形成文档，确定文档版本号。前后端开发人员要共同遵守此文档，需要修改时要共同确认，变更文档和文档版本号，然后按照文档中的接口规范进行前后端的并行开发。

地方文献系统的前后端通讯是通过 HTTP 请求作为 API 接口，这些 API 当然可以随意制定，只要前后端开发人员达成一致即可。但随意的 API 形式缺乏合理性，可读性差存在较多隐患。因此，我们采用目前较为流行的 RESTful 制定了 API 接口。

REST（Representational State Transfer）描述了一个架构样式的网络系统，比如 Web 应用程序。在目前主流的 3 种 Web 服务交互方案中，REST 相比于 SOAP 和 XML-RPC 更加简单明了，无论是对 URL 的处理还是对 Payload 的编码，REST 都倾向于用更加简单轻量的方法设计和实现。值得注意的是 REST 并没有一个明确的标准，而更像是一种设计风格。每个资源都使用 URI（Universal Resource Identifier）得到一个唯一的地址，所有资源都共享统一的接口，以便在客户端和服务器之间传输状态，使用的是标准的 HTTP 方法，比如 GET、PUT、POST 和 DELETE[4]。符合这种风格的 Web 应用则称为 RESTful。使用 RESTful 制定的 API 很容易理解，可避免产生歧义。

2.3　前后端分离的实现

2.3.1　前端的实现

前端页面采用目前流行框架 Vue.js，它是基于 MVVM 模式的轻量级响应式框架，它能有效简化 Web 前端开发流程。Vue. js 与其他重量级框架不同

之处是，它是一套构建用户界面的渐进式框架，采用自底向上增量开发的设计方式，是更加灵活、开放的解决方案，架构更加简单，适合开发人员快速掌握其全部特性并投入使用，还便于与第三方库或既有项目整合[5]。

Vue.js 体量轻盈、性能优异、路由功能强大，而且易用、灵活、高效，在 npmjs. com 上有 vue.js 脚手架工具 vue-cli，通过 vue-cli 能直接生成 Node.js 环境下的 Vue 应用程序框架，减少构建时间。另外 vue.js 现在应用广泛，网络交流社区很多，使用中如遇到任何问题都可以利用这些资源及时找到解决办法，保证开发工作的顺利进行。

为了适应多终端用户浏览器环境，页面采用响应式布局设计，根据终端环境的不同，自动适应屏幕大小，优化页面显示。响应式布局框架有很多，比较流行的是 bootstrip.js，图书馆的开发人员在以前的项目中也曾使用过，使用 npm 可以轻松地将 bootstrap.js 加入到开发环境中，然后用 import 语句将 bootstrap 及相关 css 文件引入到程序文件中，bootstrap 和 vue 整合使用效果非常好。

前端的 Web Server 是由 vue-cli 工具生成 vue 开发框架中所包含的，它实际是一个 express 框架。前端向后端请求 HTTP API 异步操作时会出现跨越问题，可以通过在前端 Web server 程序中引入 http-proxy-middleware 中间件来解决。

前端程序通过 mock. js 按照约定的接口文档生成模拟数据进行单独测试，不用等后端 API 接口开发完毕，这样能够在前后端的集成测试前发现大部分问题，从而节省集成测试的时间，缩短开发周期。

2.3.2　后端的实现

后端 Web Server 采用 Koa2。Koa 是由 Express 原班人马打造的，致力于成为一个更小、更富有表现力、更健壮的 Web 框架。使用 Koa 编写 web 应用，通过组合不同的 generator，可以免除重复繁琐的回调函数嵌套，并极大地提升错误处理的效率。Koa 不在内核方法中绑定任何中间件，它仅仅提供了一个轻量优雅的函数库，使得编写 Web 应用变得得心应手[6]。Koa2 是一个简单的 MVC 架构，结构非常清晰，地方文献系统的数据库采用 mysql，通过 npm 将 mysql 访问包文件加入到后台开发环境中，将数据处理程序如 mysql 数据库的操作、本地文件访问等程序放到 models 目录，将业务逻辑处理程序放到 controllers 目录，另外将 API 接口定义在 router 目录中 api. js 文件内，该文件要引入 koa-router 包，在主程序中通过语句 app.use(require('./server/routes/ api.js').routes())启动 API 接口路由。

Koa2 中 Koa Context 将 node 的 request 和 response 对象封装到单个对象中，为编写 Web 应用程序和 API 提供了许多有用的方法。比如用 ctx 标识符创建一个 Context，使用语句 ctx.body = result_set，这里的 result_set 是指 API 接口处理程序的结果数据集，这样前端就获得了后端返回的数据。

应用 Koa2 开发是非常简单、快捷、清晰的，要实现 Koa 功能扩展既可以自己编写中间件程序，也可在 npmjs. com 网站上找那些优秀的支持 Koa2 的中间件直接使用，提高开发效率。但是，网上的中间件良莠不齐，需要仔细甄别。

2.4　部署

前后端分离的地方文献系统集成测试完毕后，可以部署到一台或多台服务器，前后端分开以及数据库独立，可根据业务运行的实际情况按需部署，应用非常灵活。

目前，图书馆的地方文献系统前后端和数据库暂时部署在一台 Windows Server 上，从运行的效果看已经满足读者的访问要求。这得益于部署时使用了 Node.js 的高级生产进程管理工具 PM2。

Node.js 以单线程的方式运行，多核心处理器系统不能发挥最大的性能。Node.js 提供了 cluster 模块，可以生成多个工作线程，只需要将代码封装到 cluster 的处理逻辑中，再增加额外的代码用于解决一个线程挂掉的问题。PM2 内置 cluster 模式包含了所有上述的处理逻辑，不必修改任何代码，轻松地实现负载均衡，并能实时进行扩展。

PM2 简化了很多 Node.js 应用管理中的繁琐任务，不论什么情况都能保持生产进程一直运行，生产环境实现零停机更新。它还提供了性能监控、自动重启、控制台检测、远程控制接口 API 等功能，目前 PM2 支持在 Linux、macOS、Windows 等多平台运行。

3　前后端分离的优势

通过 Node. js 进行前后端分离的地方文献系统开发的优势为：首先，前后端开发人员的职责完全分清，能够同时并行开发和测试，互不干扰，提高了开发效率，缩短了开发时间，降低了项目成本；其次，前端界面变化后端无需修改，可以轻松应对今后多终端需求的改变，降低了维护成本；最后，提高了代码的可复用性和可维护性，架构明确，代码清晰。

4　结　语

利用Node.js开发前后端分离的图书馆地方文献系统的实践，不仅仅是前后端开发的分工，也是开发环境、代码、部署的完全分离，与传统的Web应用开发模式相比，提高了开发效率，增强了代码的可维护性，提高了系统的可用性、伸缩性、扩展性。今后还应在接口服务化、代码模块化、功能组件化等方面进一步探索，以便应对越来越复杂的Web应用开发挑战。■

参考文献

［1］ 淘宝UED. 前后端分离的思考与实践[EB/OL]. (2014-04-18)［2018-04-10］. http://ued.taobao.org/blog/2014104/full-stack-development-with-nodejs.

［2］ 谭发军. 谈新时期公共图书馆地方文献征集工作[J]. 科学技术创新，2011(27)：47.

［3］ Nodejs官网. Node.js中文[EB/OL]. (2010-01-02)［2018-05-1］. https://nodejs. org/zh-cn/.

［4］ 百度百科. RESTful[EB/OL]. (2011-09-12)[2018-05-02]. https://baike. baidu. com/item/RESTful/4406165.

［5］ 朱二华. 基于Vue.js的Web前端应用研究[J]. 科技与创新，2017(20)：119-121.

［6］ Koa官网. Koa介绍[EB/OL]. (2017-08-09)[2018-05-02]. https://koa. bootcss. com/#introduction.

上接第66页

4　总　结

风电叶片的灌注生产要求：灌注环境温度15～32 ℃；打胶出胶温度(25 ± 3)℃，脱完泡温度≤32 ℃，若树脂经过单组分脱泡，出胶温度≤32 ℃即可，出胶后可直接灌注。实际生产中，在高温季节，如生产订单饱满的情况，在大型厂房中加装空调是可行的，单支叶片的平均能耗成本会较低，同时采取辅助设施会大幅降低生产成本。在寒冷季节，如生产量少，在大型厂房中加装空调是高成本、高耗能的，如交货量少甚至可以短暂停产，少量生产则可采取辅助设施，虽然生产周期会加长，但生产成本增幅不高。根据生产任务量可以采用的方法是加热模具表面至环境可达到的最高温度(如25 ℃)，这样既能保持灌注工艺的稳定性，也能通过降低树脂粘度提高灌注速度。高温季节灌注过程中，当树脂温度＞35 ℃时，必须每5 mim内检测一次树脂温度，一旦温度达到40 ℃，必须更换树脂，同时10 min内向管道补充新的树脂，防止先进入管道内的树脂固化。■

参考文献

［1］ 李奎. 风电叶片真空灌注成型工艺质量问题研究[J]. 现代工业经济和信息化，2017(13)：20-21.

［2］ 郝志勇. 风电叶片真空灌注成型工艺质量问题分析[J]. 天津科技，2016(7)：77-79.

［3］ 刘宾宾，路超，杨朋飞. 风电机组风轮叶片重量的影响因素[J]. 煤炭与化工，2015(9)：148-149.