# 前后端解耦模式及开发①

吴 贺

(中国科学院计算机网络信息中心, 北京 100190)

**摘 要**: 随着互联网技术的快速发展和移动终端设备不断普及, 应用系统需要适应不断变化的移动终端设备, 需要以用户为中心不断提高系统友好性. 因此, 本文研究前后端解耦模式的开发. 基于 JSON 数据格式, 前端独立开发不断提升用户体验并自适应各类终端的布局.

**关键词**: 前端开发; 前后端解耦; AngularJS; JSON

## Frontend and Backend Decoupling Model and Development

WU He

(Computer Network Information Center, Chinese Academy of Sciences, Beijing 100190, China)

**Abstract**: With the rapid development of internet technology and the popularity of mobile devices, the application system needs to adapt to the changing mobile terminal devices, and needs to be user centric and improve the system friendly. Therefore, this paper studies the development of the frontend and backend decoupling model. Using the data format of JSON, the frontend developments should independently improve user experience and adapt the layout of diffident mobile terminal devices.

**Key words**: frontend development; frontend and backend decoupling; AngularJS; JSON

## 1 引言

B/S 架构(Browser/Server, 浏览器/服务器模式), 作为当今最为流行的计算机软件开发模式, 受到了用户的青睐, 广泛应用于各类应用软件系统的开发中, 有很多优点, 发挥了很好的作用. 例如, 客户端不需要安装, 主要业务逻辑在服务器端处理; 升级维护更加方便, 可以降低成本等.

随着互联网应用新技术的发展和用户体验水平的不断提升, 传统 B/S 开发模式的弊端也日益显露, 对新技术的适应性和新需求的支持性还是有些不足. 其中, 比较突出的问题是前端和后端的关系问题. 具体体现在, 从项目组织或人的角度, 要求开发人员从数据库设计、前台页面展示, 到后台业务逻辑, 甚至是 UI 设计都要负责, 很难有这样的人才. 前端多屏互动的要求, PC 端和移动端都很精通的高手, 也很难得. 前后端关联过强, 对开发人员、开发组织职责安排带来很大的问题, 项目组织比较困难. 前后端的依赖性, 造成系统的灵活性差. 如果需求有变化, 前后端都需要改变, 前端的改变可能会影响后端, 后端的改变也可能影响前端, 不能很好地适应千变万化的需求. 在日常工作的系统开发、应用推进过程中, 也遇到了类似的问题. 因此考虑研究前后端解耦的模式.

## 2 研究现状

新型 B/S 架构开发模式的发展, 离不开相关技术的出现和逐渐成熟. 后端开发相关技术相对比较稳定, 前端相关技术, 如: 前端开发语言 Bootstrap 和 AngularJS, 前端自动化管理工具 NodeJS、NPM、gulp, 数据传输语言 Json 等, 促进了这一开发模式的发展.

AngularJS 诞生于 2009 年, 由 Misko Hevery 等人创建, 是一款来自 Google 的前端 JS 框架, 该框架已经被应用到了 Google 的多款产品中[1]. AngularJS 是完全使用 JavaScript 编写的客户端技术[2], 是为了克服 HTML 在构建应用上的不足而设计的. HTML 是一门

① 收稿时间:2016-04-30;收到修改稿时间:2016-07-19 [doi: 10.15888/j.cnki.csa.005638]

很好的为静态文本展示设计的声明式语言, 但要构建 WEB 应用的话它就显得乏力了. 通常, 开发人员通过类库、框架技术来解决静态网页技术在构建动态应用上的不足. AngularJS 尝试去补足 HTML 本身在构建应用方面的缺陷, 通过使用标识符(directives)的结构, 让浏览器能够识别新的语法. AngularJS 的开发团队将其描述为一种构建动态 WEB 应用的结构化框架. 这款框架最核心特性有: MVC、模块化、自动化双向数据绑定、语义化标签、依赖注入等.

Bootstrap 是 Twitter 推出的一个用于前端开发的开源工具包. 它由 Twitter 的设计师 Mark Otto 和 Jacob Thornton 合作开发, 是一个 CSS/HTML 框架. 它提供了丰富的元素和组件, 并使得形式和功能结合得很好. Bootstrap 的强大之处在于它对常见的 CSS 布局小组件和 JavaScript 插件都进行了完整且完善的封装, 即使代码不是很理解, 也轻松使用. Bootstrap 主要具有网格系统、响应式设计、扁平化和整洁的 UI、基于 LESS 等优点[3].

Node 是一个 javascript 运行环境. 它的实质是对 google V8 引擎进行了二次封装, 并且优化了一些特殊用例, 提供了替代的 API, 使得 V8 引擎在非浏览器环境得以顺畅运行. NodeJS 只是 javascript 运行环境. NodeJS 采用非阻塞的模式, 因此相对于其他传统的语言, 在某些领域它发挥得更出色. 比如使用 Java 开发 JSP 程序, 每一个请求都会分配一定的内存区, 这就导致内存成为性能瓶颈; 而 NodeJS 则完全不会, 它使用事件驱动, 单进程单线程(现在的版本也支持多进程), 因此内存不会成为它的性能瓶颈, 因此它更适合作为中间件, 进行事务处理. 目前, 大型互联网公司中都有专门的技术人员进行研究 NodeJS, 以融入进公司的产品中去. 同时, 它本身的发展还在继续中, 几乎每一个星期都会发布一个新的版本号以支持更多的功能和提供高稳定性[4].

JSON 是(JavaScript Object Notation, Java-Script 对象符号)一种轻量级的数据交换格式. JSON 采用与编程语言无关的文本格式, 但是也使用了类 C 语言(包括 C, C++, C#, Java, JavaScript, Perl, Python 等)的习惯, 这些特性使 JSON 成为理想的数据交换格式. 虽然, JSON 是一种用来存储数据的 JavaScript 语言符号, 但是更严格的说, JSON 不是一种编程语言, 其实是一种直接定义部分文档对象模型的方法. 像 JavaScript 语言的其他部分一样, JSON 可以直接用来编写 WEB 页面的某些部分. JSON 文件就是普通的 ASCII 文本文件, 使用普通的文字编辑器就可以创建. 与机器代码不同的是, JSON 永远不需要编译和执行[5].

随着互联网的高速发展, 对 WEB 前端的要求是越来越专业化, WEB 前端本身所包含的技术难度已经不亚于任何一个服务端语言开发难度, 因此 WEB 前端开发需要更高的专业化, 而不希望 WEB 前端工程师被服务端技术束缚的更多而限制了自身能力的发展, 新一代 B/S 架构下的开发模式逐渐产生, 实现前端与后端开发工作解耦.

## 3 前后端解耦模式及开发的研究

### 3.1 总体思路

用户使用各种终端访问系统, 项目中提供 IOS 和 android 的 APP, 前端技术采用 Bootstrap 可自适应各终端的布局, 通过 AngularJS 的路由管理各种用户请求和页面跳转, 用 JSON 数据直接传送给后端处理. 首先是由 struts2 的核心控制器 FilterDispatcher 截获前端请求, 将其转交给 struts2 的拦截器进行权限检查及初步的数据校验, struts2 将收到的 JSON 数据和请求交给 Action, Action 解析请求, 并根据解析后的用户请求调度相应业务规则. 业务(Service)将从 SessionFactory(会话工厂)中获取 Session, 通过接口调用封装的数据持久化方法, 根据业务路由规则和数据源配置信息, 自动匹配业务所在数据库, 如果是查操作, 直接从读库将数据读取出来; 如果是删除操作, 直接将读写库中对应数据删除; 如果是插入和更新操作, 项目自动将数据存储到读写数据库中(对象的持久化由 Hibernate 负责). 成功后, 关闭 Session. 如图 1 所示.

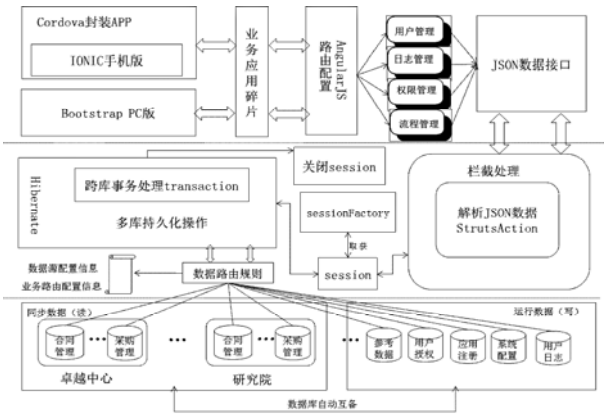


图 1 前后端数据处理流程

基于 JSON 数据格式, 前后端通过接口调用的方式实现后端数据库数据和业务处理与 WEB 的数据交互. 前端基于 NODEJS+NPM 技术对开源依赖包及工具管理, 通过 GULP 工具编写角本对前端开发语法检查、动态发布管理、静态模板管理、开发库依赖管理、单元测试管理、自动构件管理和监听自动化管理. 后端基于 maven 对 JAVA 外部引用包、外部工具包进行自动下载、自动管理、自动编译等, 通过 checkStyle 自动检测 JAVA 代码的语方格式, 用 Junit 进行单位测试, 完成与前端接口的结合. 如图 2 所示.

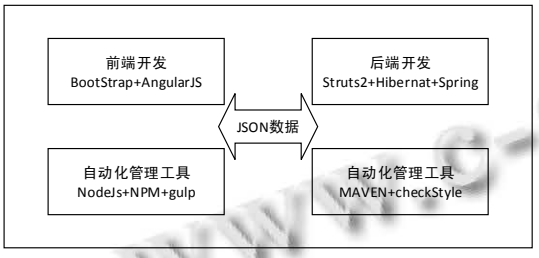


图 2 前后端接口方式

### 3.2 前端开发方法研究

前后端分离的开发模式, 业务上物理解耦、逻辑关联. 物理上前端采用 Bootstrap+AngularJS 开发技术, NODEJS+NPM+GULP 自动化管理工具.

前端代码结构如下图所示, 包括 api-接口层、comp-组件、desk-桌面业务前端、mobile-手机业务前端. Api-接口层、comp-组件是可以复用的, 供桌面业务前端、手机业务前端调动. 其中, api-接口提供基于 JSON 数据格式的接口, 供应用前端调用. 前端源代码结构如图 3 所示.

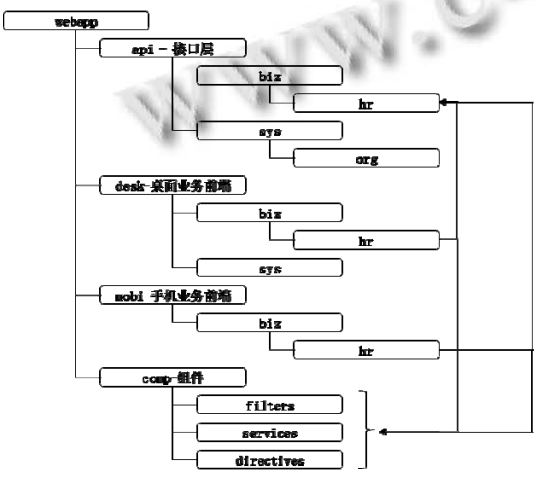


图 3 前端源代码结构

通过 GULP 工具编写的角本, 自动将源代码(JS、CSS、HTML)合并、压缩编译执行代码. 程序员对执行代码调测, 修改后直接自动编译, 自动发布到指定的工程目录下, 和后端进行无缝接入. 如图 4 所示.

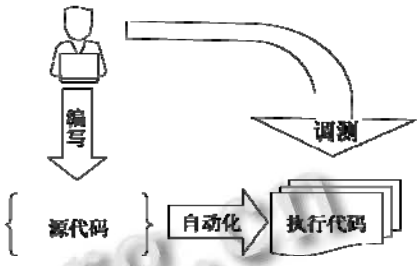


图 4 自动化工作链

### 3.3 后端开发实现

后端服务采用传统的 Struts2+Spring+Hibernate 三层架构设计. Struts2 主要体现在控制层, 它负责控制业务逻辑层与前端的交互, 截获前端请求, 并调用业务逻辑层, 最后将业务逻辑层处理后的数据返回给表现层. Spring 的工作主要体现在业务逻辑层, 在这一层次中, Spring 再次将逻辑控制和逻辑业务分离, 细分为以下两个层次: Service 层和 DAO 层, DAO 层负责与持久化对象交互, 封装了增, 删, 改, 查等操作的标准方法(提供增删改查的接口), Service 层以 DAO 层为基础, 通过标准接口来实现业务逻辑. Hibernate 则通过实体关系映射工具将关系型数据库的数据映射成对象, 方便实现以面向对象的方式来操作数据库, 是目前最流行的 ORM 框架.

三层体系结构将业务规则, 数据的访问, 逻辑校验等工作放在业务逻辑层处理, 客户端不直接与数据库交互, 而是通过 JSON 数据接口的方式, 通过调用控制层建立连接, 在通过业务逻辑层与数据库交互.

该架构的优势是使得层次之间相对独立, 在各层上的组件能单独更新、替换、增加或删除. 因此, 系统维护更加方便. 通过将业务逻辑集中到中间的业务逻辑层, 系统获得了对业务逻辑的独立处理能力. 当用户的需求改变时, 开发人员可以迅速的在中间层(业务逻辑层)上更新业务规则, 而客户端无需任何改动. 后端源代码结构如图 5 所示.

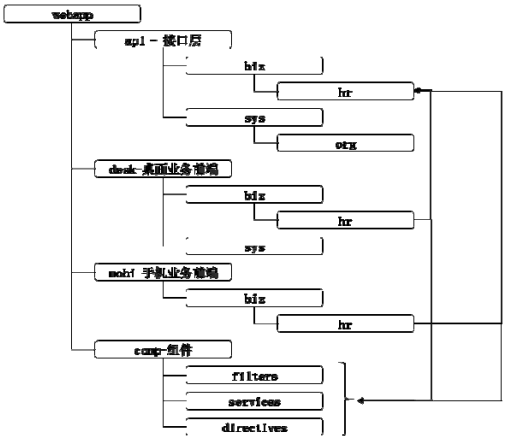


图 5　后端源代码结构

## 4　系统结构设计

### 4.1 系统概述

学秘系统面向中科院科研人员和硕博士研究生, 汇聚学术活动、项目指南、情报服务、专利信息等学术相关内容, 提供学生论文修改找导师、项目合作找专家、科技成果产业化找企业等服务.

### 4.2 系统设计

系统设计主要采用原型法. 首先进行需求分析, 调研科研人员的相关需求, 分析需求并绘制设计草图进行内部讨论. 在界面草图的基础上, 使用 Axure RP 工具绘制界面原型, 并和用户进行沟通. UI 设计师依据原型系统进行 UI 设计, 并进行切图. 与此同时, 前端开发工程师开始规划业务路由, 后端开发工程师开始进行数据库设计. 设计完成了前后端开发工程师开发分别进行前端开发测试、后端开发测试. 最终进行整体的测试和打包工作.

系统功能主要包括: 首页、消息管理、合作者管理、我的管理. 如图 6 所示.

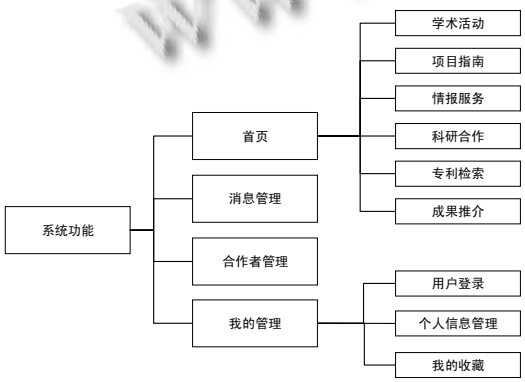


图 6　学秘系统功能设计

### 4.3 系统开发要点

#### 4.3.1 前端业务路由

规划业务路由, 以首页中的学术活动页面为例, 规划业务路由如下:

```
export default angular.module
  ('arp.ram.index.home', ['ionic'])
      .config(['$stateProvider',
        function($stateProvider) {
          $stateProvider
            .state('ram.home.academic', {
 url: '/academic',
 views: {
 "@": {
templateUrl: 'mobi/ram/biz/academic/list.html',
controller:
 'AcademicController as bizCtrl'
        }
        },
 resolve: {
    editingEntry: () => ({})
          }
 });
}])
                          .controller('AcademicController',
AcademicController);
```

#### 4.3.2 前后端数据接口

基于 JSON 数据格式, 前后端通过接口调用的方式实现后端数据库数据和业务处理与 WEB 的数据交互.

1) 前端接收后端传递的 rA_INFO 实体

```
export var readUri = '/service/rA_INFO!
queryCaList';
export var entryUri = '/service/rA_INFO!
queryEntity';
```

2) 读取后端传递的实体信息

```
return this.post(readUri, this.create
StreamParams(params, operatorType))
    .then(function(resp) {
     // 系统层次, 请求成功
      if (resp.data.success) {
        // 业务层次, 业务完成
```

```
          return resp.data.obj.obj;
        } else {
          // 业务层次, 业务失败
          self.errorMessage = resp.data.
      message;
          return [];
        }
  }, function() {
       // 系统层次, 请求失败
       self.errorMessage = '系统失败, 请联系管理员';
      });
```

3) 将读取的内容注入到控制器中

```
resolve: {
//resolve 中的内容需要注入到控制器里面去
editingEntry: ['$http', '$stateParams', 'configure',
function($http, $state
Params,configure) {
   console.info('$stateParams',$stateParams);
   console.debug("Id:" + $stateParams.
ID);
   return $http.post(configure.baseUrl + entryUri, {
      ID: $stateParams.ID
      }).then((resp) => resp.data.obj,
      () => ({}));
  }]
}
```

4) 在页面中解析显示后端读取的信息

```
<div ="card-list">
      <a class="reply-ul" ng-repeat="item in bizCtrl.data"
ng-click="bizCtrl.more
(item.ID)">
      <div class="favorite-list">
      <div class="favorite-head">
      <span class="reply-name">
      {{item.RELEASE_UNITNAME}}</span>
      <span class="reply-date">
      {{item.RELEASE_TIME|limitTo:10}}</span>
      </div>
      <font class="favorite-content">
      <span>{{item.TITLE}}</span>
      </font>
      </div>
      </a>
</div>
```

## 5 结语

前后端解耦的开发模式, 前后端工程师专注于本领域的技术进行研究, 分工更加明细, 提高开发质量. 前后端程序交互少, 前后端工程师各司其职, 减少很多不必要的沟通协调, 专业的人做专业的事.

未来前端技术不断发展, 前端开发将更加专业化, 系统将更加关注与用户交互的友好型, 以人为本, 不断完善前端用户体验.

## 参考文献


1 Green B, Seshadri S.大漠孤秋译.用 AngularJS 开发下一代 Web 应用.北京:工业电子出版社,2013:1–9.

2 Lerner A.赵望野,徐飞,何鹏飞译. AngularJS 权威教程.北京:人民邮电出版社,2014:1–3.

3 Spurlock J.李松峰译.BootStrap 用户手册:设计响应式网站.北京:人民邮电出版社,2013:1–10.

4 https://nodejs.org/en/.

5 Rischpater R. JavaScript JSON Cookbook. Packt Publishing, 2015: 1–30.