天津工业大学计算机科学与技术学院

实训总结报告



　项目名称: 高数-您的高级数据信息服务平台

　合作培养企业：天津市融创软通科技股份有限公司

　专业方向：大数据方向

　项目负责人：何晶

　专业班级: 软件Z1802班

姓名：廖嘉伟

　实训起止日期：2020.6.15-2020.7.15

## 1 引言

#### 1.1项目概述

###### 1.1.1项目目的

大数据时代，信息海量且源源不绝，面对庞大的数据，如果运用合适的技术手段，就能提取出有价值的信息，这也就是本项目的目的。

本项目正是基于此目的，先使用基于Python的**Scrapy网络爬虫框架**爬取特定网站的信息数据，包括**豆瓣电影、音乐、图书以及当当图书、B站动漫评论**，并将数据保存到远端的**MariaDB数据库**中，然后使用前后端分离的开发模式，后端使用**SpringBoot**开发，前端则采用基于NodeJs的**Vue框架**，并通过**Mybatis**实现数据查询及数据分析功能，并通过**eCharts**将数据的分析结果展示到前端，最终完成数据、前端与后端相连接，形成完整的项目，使信息可以更方便轻松地被用户提取运用。

整个过程中，小组团队运用了代码托管平台Gitee以及代码版本控制器git。，、Python、Java、Ajax、JSON等在现代项目开发条环境下的主要技术。

###### 1.1.2项目背景

当下正处于信息爆炸的时代，互联网几乎是人们工作、生活、学习都无法脱离的平台，每天都会产生海量数据，音乐、电影、书籍、评论等种种数据种类繁杂且数量庞大因此现在也被称为大数据时代。几乎可以说，谁掌握了数据，谁就拥有了先机。

但我们也知道，数据越庞大，我们处理起来也会越棘手，只有通过合适的技术手段才能更有效地提取出更有价值的信息。于是我们采用了一系列的技术手段，加以正确的组合连接，设计了本项目。

###### 1.1.3项目意义

拿项目中豆瓣图书一块来举例，在数据查询与信息展示功能中，用户可以自定义地输入自己想要查询的条件，比如书名、作者名的关键词查询，也可以加入多个书籍类型以筛选自己想要的书籍类型，此外还可以通过筛选豆瓣的评分人数以及豆瓣的评分区间来找到当下的热书、好书，更方便用户找到想要的书。

而在数据分析功能中，用户则可以看到在庞大的数据下，书籍信息有着什么样的规律，比如什么类型的书更多，说明了该类型的书的需求更大；亦或者什么类型的书平均分高，而什么样的书平均评分低，可以得到什么类型的书更受欢迎的信息；通过分析书籍的出版时间，则可以了解人们从什么时候开始有读书热潮，书籍出版数量井喷式增长。这些分析信息都可以为用户的各类活动带来参考价值。

###### 1.1.4参考资料

企业授课内容：

Python3，idea2019，git等工具包；

python基础及相关ppt参考资料；

scrapy框架及讲解录屏；

spring boot框架及讲解录屏；

mybatis框架（数据接口）及讲解录屏；

layui基础及讲解录屏；

echarts讲解录屏

提供demo：

豆瓣电影top250scrapy框架python码参考；

数据接口java码参考；

Layui和echarts码参考；

服务：

远端MariaDB数据库；

Gitee远端代码仓库；

#### 1.2 需求分析

随着社会发展，人们的生活质量越来越高，伴随着各种娱乐产业的新增，每天都有海量数据生成。单单只看其中一些数据，是毫无意义的，担当将当量数据集中金雄分析，就能从中取得许多信息，这也是大数据的意义所在。

该项目将爬取豆瓣电影、音乐、图书和当当图书、B站动漫评论的数据，然后所获得的信息进行分析，得到诸如最畅销的书籍、最受欢迎的导演等等数据，之后以表格的形式将分析直观的展示出来，以此为客户挑选电影、音乐、动漫、书籍时提供便捷。

#### 1.3 运行环境

###### 1.3.1硬件环境

1.笔记本1台

2.内存：4GB以上。

3.局域网畅通

###### 1.3.2软件环境

1. 操作系统：Windows 10。

2. 文档编写工具：Microsoft office 2016（word excel powerpoint visio）。

3. 开发工具：IDEA2019，PyCharm2019，MySQL5.7.x，WebStorm2019、Navicat、ECharts、Postman。

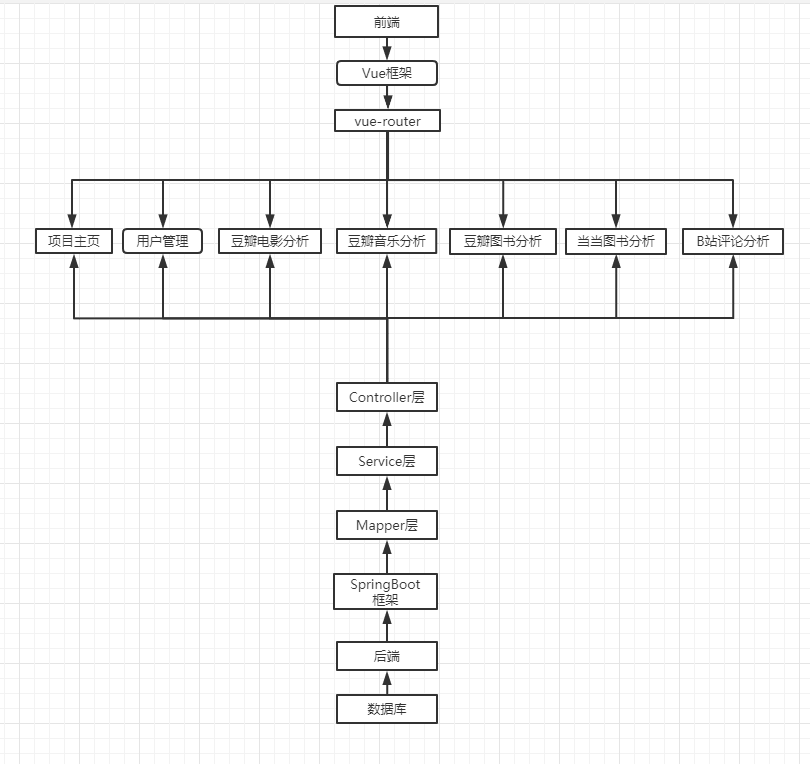
4. 版本控制工具：Git/码云。

5. 任意支持HTML5的浏览器

## 2 项目设计

#### 2.1 设计思路

###### 2.1.1总体设计思路



**系统大致可以分为三层：Vue前端、SSM后端、以及MariaDB数据库三层。MarialDB的数据来自于Scrapy爬虫爬取到的数据，用以进行后端分析以及前端展示；SSM后端主要有三层：**

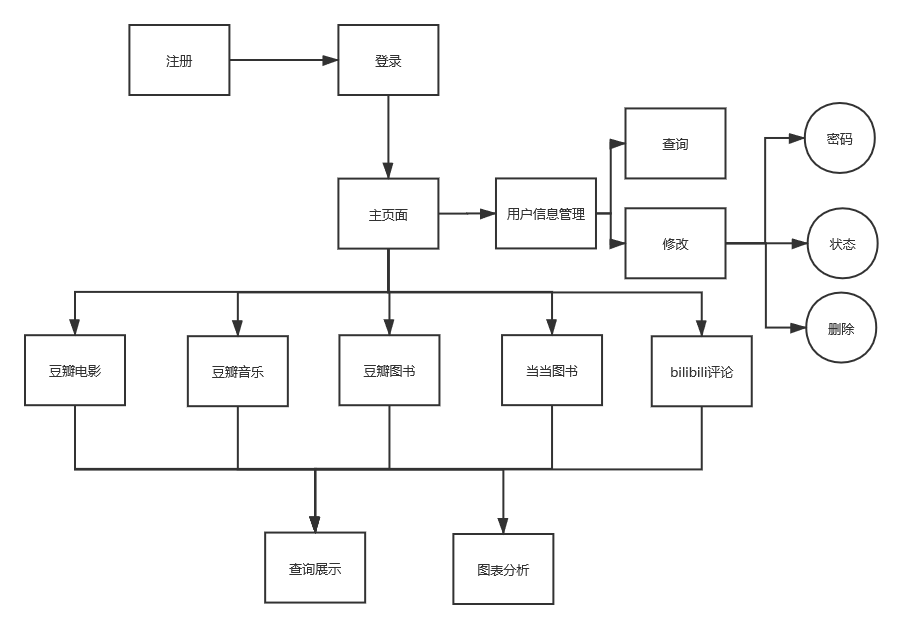
**·Controller层**：控制层，负责具体模块的业务流程控制，需要调用service逻辑设计层的接口来控制业务流程。

·Service层：业务层，包括service接口类以及serviceImp实现类，为给controller层的类提供接口进行调用。

·Mapper层：对数据库进行数据持久化操作，他的方法语句是直接针对数据库操作的，主要实现增删改查以及复杂查询等操作。

通过上述框架结构，向前端提供接口，完成交互，最后Vue前端通过接口获得相应数据，完成展示

###### 2.1.2总体程序流程图



#### 2.2 模块功能介绍

**模块一：系统基础管理模块**

1) 平台主界面设计及开发。

2) 管理首页设计与开发，图形化当前系统情况，包括已注册用户数量、已贮存影片（书籍、评论、音乐）数量、已分析的评论数、各分类的检索前十名汇总等信息；

**模块二：当当图书模块**

1) 通过不同关键字迅速返回符合条件的数据目录

2) 将会所获得数据进行分析，结果以图表方式展出

**模块三：B站评论模块**

1) 通过不同关键字迅速返回符合条件的数据目录

2) 将会所获得数据进行分析，结果以图表方式展出

**模块四：豆瓣音乐模块**

1) 通过不同关键字迅速返回符合条件的数据目录

2) 将会所获得数据进行分析，结果以图表方式展出

**模块五：豆瓣电影模块**

1) 通过不同关键字迅速返回符合条件的数据目录

2) 将会所获得数据进行分析，结果以图表方式展出

#### 2.3 模块结构图

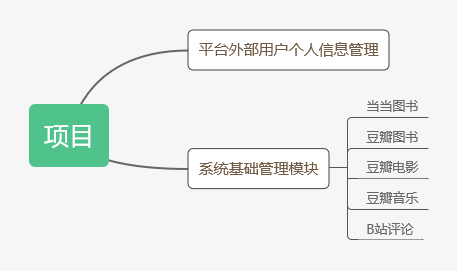
****

Figure 2模块结构图

项目大致可分为七个模块，包含两个主模块，五个小模块。主模块包括平台外部用户个人信息管理，主要负责用户注册，修改信息；系统基础管理模块负责平台页面框架的搭建，其中包含着当当图书、豆瓣图书、豆瓣电影、豆瓣音乐、B站评论五个模块的内容，各个模块内有自己的功能。

#### 2.4 程序流程图

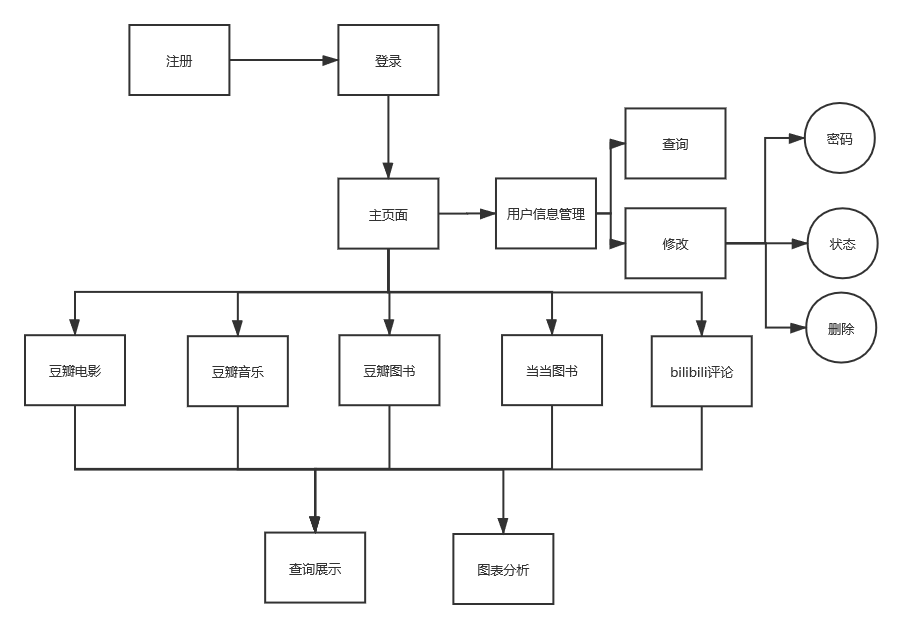


Figure 3程序流程图

#### 2.5 功能设计分工

**五大模块（个人独立完成所负责模块的全部内容）：**

豆瓣电影：段成源

豆瓣音乐：刘博

豆瓣图书：廖嘉伟

当当图书：薛彦涵

B站评论：李嘉威

**其余部分前端框架：**

主界面搭建：李嘉威

登录：薛彦涵

注册：李嘉威

用户管理：李嘉威

**其余部分服务器端：**

登录：李嘉威

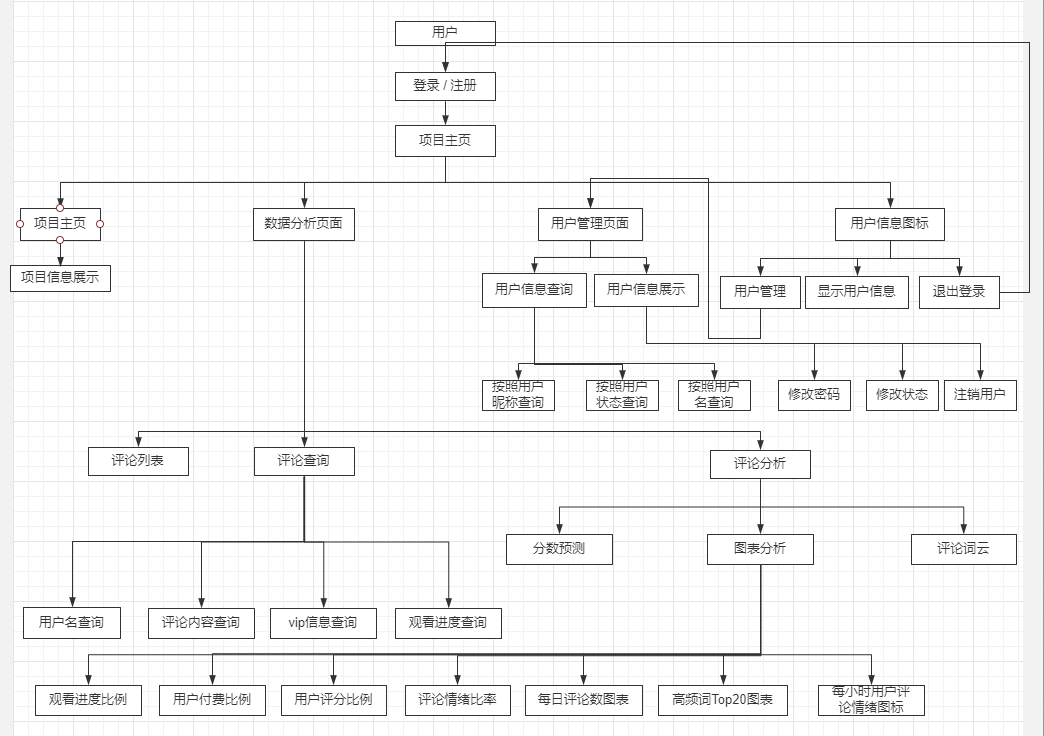
注册：刘博，薛彦涵

**3 详细设计**

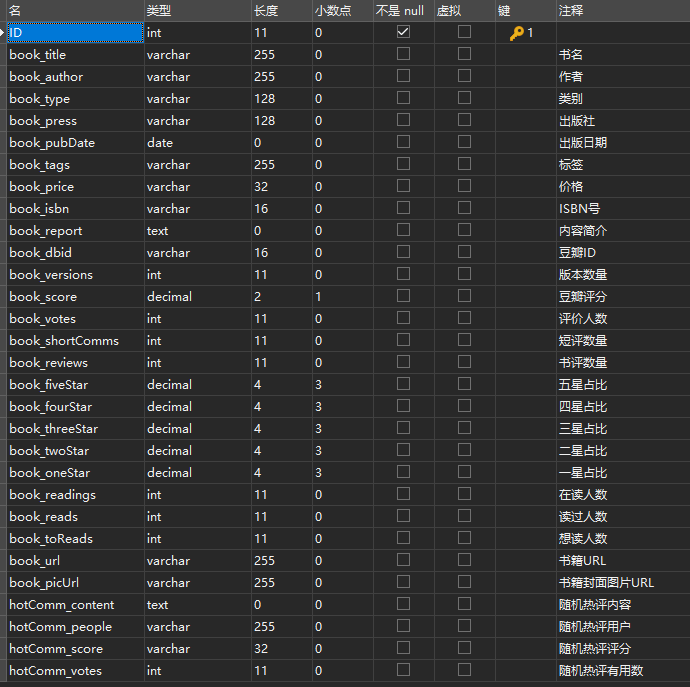
#### 3.1系统设计

整体上采用了前后端分离的开发方式，前段采用Vue框架进行开发，前端使用UI组件库为Ant-Design-Vue，前端使用Axios向后端发送请求，后端使用SpringBoot框架开发，通过scrapy爬虫机制针对豆瓣电影、音乐、图书和当当图书、B站动漫评论进行数据收集并分析，结果保存与远程mariaDB数据库中。提供简洁明了的数据查询服务，便于查询所需条件的数据，也提供关于数据内容的分析服务，并将其以图表的方式简洁直观的展示出来，方便用户观看。用户可根据分析所获结果，快捷的寻找满意的电影、音乐、书籍等资源，此外团队远程开发利用代码托管平台gitee和代码版本控制器git。

**具体设计如下：**

**系统设计：**

**远端Mariadb数据库**：



**scrapy爬虫，包括：**

Scheduler(调度器)：它负责接受引擎发送过来的Request请求，并按照一定的方式进行整理排列，入队，当引擎需要时，交还给引擎。

Downloader（下载器）：负责下载Scrapy Engine(引擎)发送的所有Requests请求，并将其获取到的Responses交还给Scrapy Engine(引擎)，由引擎交给Spider来处理。

Spider（爬虫）：它负责处理所有Responses,从中分析提取数据，获取Item字段需要的数据，并将需要跟进的URL提交给引擎，再次进入Scheduler(调度器)。

Item Pipeline(管道)：它负责处理Spider中获取到的Item，并进行后期处理（详细分析、过滤、存储等）的地方。

Downloader Middlewares（下载中间件）：一个可以自定义扩展下载功能的组件。

Spider Middlewares（Spider中间件）：一个可以自定扩展和操作引擎和Spider中间通信的功能组件。

**SSM后端：**

Controller层（控制层，负责具体模块的业务流程控制，需要调用service逻辑设计层的接口来控制业务流程）

Mapper层（对数据库进行数据持久化操作，他的方法语句是直接针对数据库操作的，主要实现一些增删改查操作）

Service层（业务service层，给controller层的类提供接口进行调用。）

主要包括：用户管理所需接口，包括登录、注册、修改用户信息；豆瓣音乐信息，包括带有关键字（音乐名、歌手、时间、标签、流派）的查询，可视化数据分析

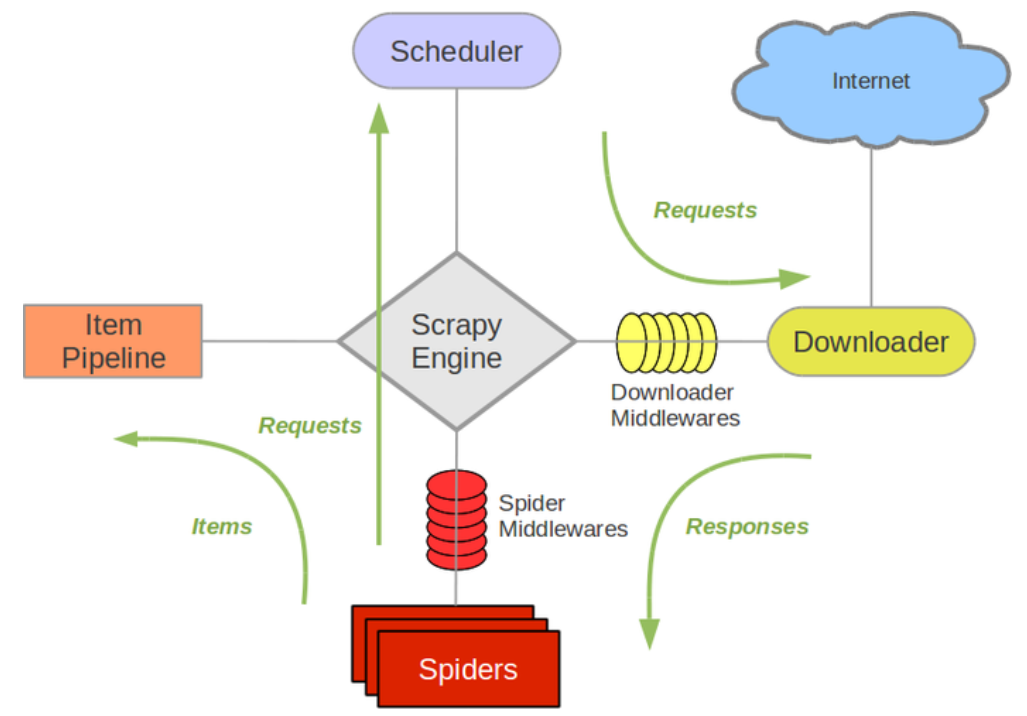
**VUE前端：**

前端框架，主要负责豆瓣音乐关键字查询、可视化图表分析界面

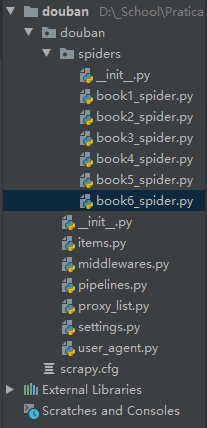
**4 实现方案**

#### 4.1爬取豆瓣读书数据功能实现

###### 4.1.1 scrapy爬取框架



###### 4.1.2 项目部署目录结构



六个爬虫对应着豆瓣读书的六个图书大类：

文学、流行、文化、生活、经管、科技

###### 4.1.3 核心代码展示（以爬取文学类书籍信息为例）

1. item['book\_url'] = response.url
2. item['book\_picUrl'] = response.xpath('//div[@id="mainpic"]/a/@href').get()
4. item['book\_title'] = response.xpath('//div[@id="wrapper"]/h1/span/text()').get()
5. item['book\_author'] = get\_author(response)
6. item['book\_type'] = '文学'
7. item['book\_press'] = response.xpath(u'//span[contains(./text(), "出版社:")]/following::text()[1]').get()
8. item['book\_pubDate'] = get\_pubdate(response)
9. item['book\_tags'] = get\_tags(response)
10. item['book\_price'] = fix\_field(response.xpath(u'//span[contains(./text(), "定价:")]/following::text()[1]').get())
11. item['book\_isbn'] = response.xpath(u'//span[contains(./text(), "ISBN:")]/following::text()[1]').get()
12. item['book\_report'] = get\_report(response)
13. item['book\_dbid'] = item['book\_url'].split('/')[-2]
14. item['book\_versions'] = get\_versions(response)

以上部分为爬取书籍url以及书籍基本信息，包括书名、作者、分类、出版社等的核心代码，其中有几项比较复杂，于是写了专门的函数来爬取。

1. item['book\_score'] = get\_score(response)
2. item['book\_votes'] = convert\_field(response.xpath(
3. '//a[@class="rating\_people"]/span/text()').get()) **if** response.xpath(
4. '//a[@class="rating\_people"]') **else** 0
5. item['book\_shortComms'] = convert\_field(response.xpath(
6. '//div[@class="mod-hd"]/h2/span[2]/a/text()').get()).split(' ')[1] **if** response.xpath(
7. '//div[@class="mod-hd"]/h2/span[2]') **else** 0
8. item['book\_reviews'] = convert\_field(response.xpath(
9. '//section[@id="reviews-wrapper"]/header/h2/span/a/text()').get()).split(' ')[1]
10. item['book\_fiveStar'] = float(convert\_field(response.css(
11. 'span.rating\_per')[0].css('::text').get()).split('%')[0]) / 100 **if** response.css(
12. 'span.rating\_per') **else** 0
13. item['book\_fourStar'] = float(convert\_field(response.css(
14. 'span.rating\_per')[1].css('::text').get()).split('%')[0]) / 100 **if** response.css(
15. 'span.rating\_per') **else** 0
16. item['book\_threeStar'] = float(convert\_field(response.css(
17. 'span.rating\_per')[2].css('::text').get()).split('%')[0]) / 100 **if** response.css(
18. 'span.rating\_per') **else** 0
19. item['book\_twoStar'] = float(convert\_field(response.css(
20. 'span.rating\_per')[3].css('::text').get()).split('%')[0]) / 100 **if** response.css(
21. 'span.rating\_per') **else** 0
22. item['book\_oneStar'] = float(convert\_field(response.css(
23. 'span.rating\_per')[4].css('::text').get()).split('%')[0]) / 100 **if** response.css(
24. 'span.rating\_per') **else** 0
26. **if** response.xpath('//div[@id="collector"]/p[3]'):
27. item['book\_readings'] = int(convert\_field(response.xpath(
28. '//div[@id="collector"]/p[1]/a/text()').get()).split('人在读')[0])
29. item['book\_reads'] = int(convert\_field(response.xpath(
30. '//div[@id="collector"]/p[2]/a/text()').get()).split('人读过')[0])
31. item['book\_toReads'] = int(convert\_field(response.xpath(
32. '//div[@id="collector"]/p[3]/a/text()').get()).split('人想读')[0])
34. **elif** response.xpath('//div[@id="collector"]/p[3]') **is** None **and** response.xpath('//div[@id="collector"]/p[2]'):
35. item['book\_readings'] = 0
36. item['book\_reads'] = int(convert\_field(response.xpath(
37. '//div[@id="collector"]/p[1]/a/text()').get()).split('人想读')[0])
38. item['book\_toReads'] = int(convert\_field(response.xpath(
39. '//div[@id="collector"]/p[2]/a/text()').get()).split('人想读')[0])
41. **elif** response.xpath('//div[@id="collector"]/p[2]') **is** None **and** response.xpath('//div[@id="collector"]/p[1]'):
42. item['book\_readings'] = 0
43. item['book\_reads'] = 0
44. item['book\_toReads'] = int(convert\_field(response.xpath(
45. '//div[@id="collector"]/p[1]/a/text()').get()).split('人想读')[0])
47. **else**:
48. item['book\_readings'] = 0
49. item['book\_reads'] = 0
50. item['book\_toReads'] = 0

以上部分为爬取涉及豆瓣用户给的反馈信息部分的核心代码，如爬取豆瓣评分、评价总人数、五星分别占比、书评数和短评数等，其中有几项比较复杂，于是写了专门的函数来爬取。

1. item['hotComm\_content'] = get\_hotcommcontent(response)
2. item['hotComm\_people'] = fix\_field(response.xpath(
3. '//div[@class="comment-list hot show"]/ul/li[1]/div/h3/span[2]/a/text()').get())
4. item['hotComm\_score'] = fix\_field(response.xpath(
5. '//div[@class="comment-list hot show"]/ul/li[1]/div/h3/span[2]/span[@class="user-stars allstar50 rating"]/@title').get())
6. item['hotComm\_votes'] = convert\_field(response.xpath(
7. '//div[@class="comment-list hot show"]/ul/li[1]/div/h3/span[1]/span/text()').get())

该部分为爬取了书籍的一条热短评，包括短评的内容、发布者、所给出的评分以及为该短评点赞的人数。

1. **def** convert\_field(field):
2. **return** field.strip() **if** field **else** '0'

5. **def** fix\_field(field):
6. **return** field.strip() **if** field **else** ''

9. **def** get\_author(response):
10. # 有些不同页面的author html会有所不同
11. author = response.css('#info > a:nth-child(2)::text').get()
12. **if** **not** author:
13. author = response.css('#info > span > a::text').get()
14. # 部分书籍如一千零一夜，没有作者，使用get，若无则直接返回None
15. **return** author.replace('\n            ', '').strip() **if** author **else** '未摘录/佚名'

18. **def** get\_pubdate(response):
19. date = response.xpath(u'//span[contains(./text(), "出版年:")]/following::text()[1]').get()
20. **if** date:
21. **return** date.strip() **if** len(date.split('-')) > 2 **else** date.strip() + '-15'
22. **else**:
23. **return** None

26. **def** get\_tags(response):
27. # 有问题，不知道怎么用xpath爬取tags
28. # tags\_list = response.xpath('//\*[@id="db-tags-section"]/div')
29. # tags = ''
30. # for tag in tags\_list:
31. #     tags += '#'
32. #     tags += tag.xpath('/a/text()').get().strip()
33. #     tags += ' '
34. # return tags
35. tags\_list = response.css('a.tag')
36. tags = ''
37. **for** tag **in** tags\_list:
38. **if** tag:
39. tags += '#'
40. tags += tag.css('::text').get().strip()
41. tags += ' '
42. **return** tags **if** tags != '' **else** '无'

45. **def** get\_report(response):
46. report = ''
47. **if** response.css('#link-report > div:nth-child(1) > div'):
48. report\_list = response.css('#link-report > div:nth-child(1) > div > p::text').extract()
49. **for** s **in** report\_list:
50. report += s
51. **else**:
52. report\_list = response.css('div.intro')[1].css('p::text').extract()
53. **for** s **in** report\_list:
54. report += s
55. **return** report

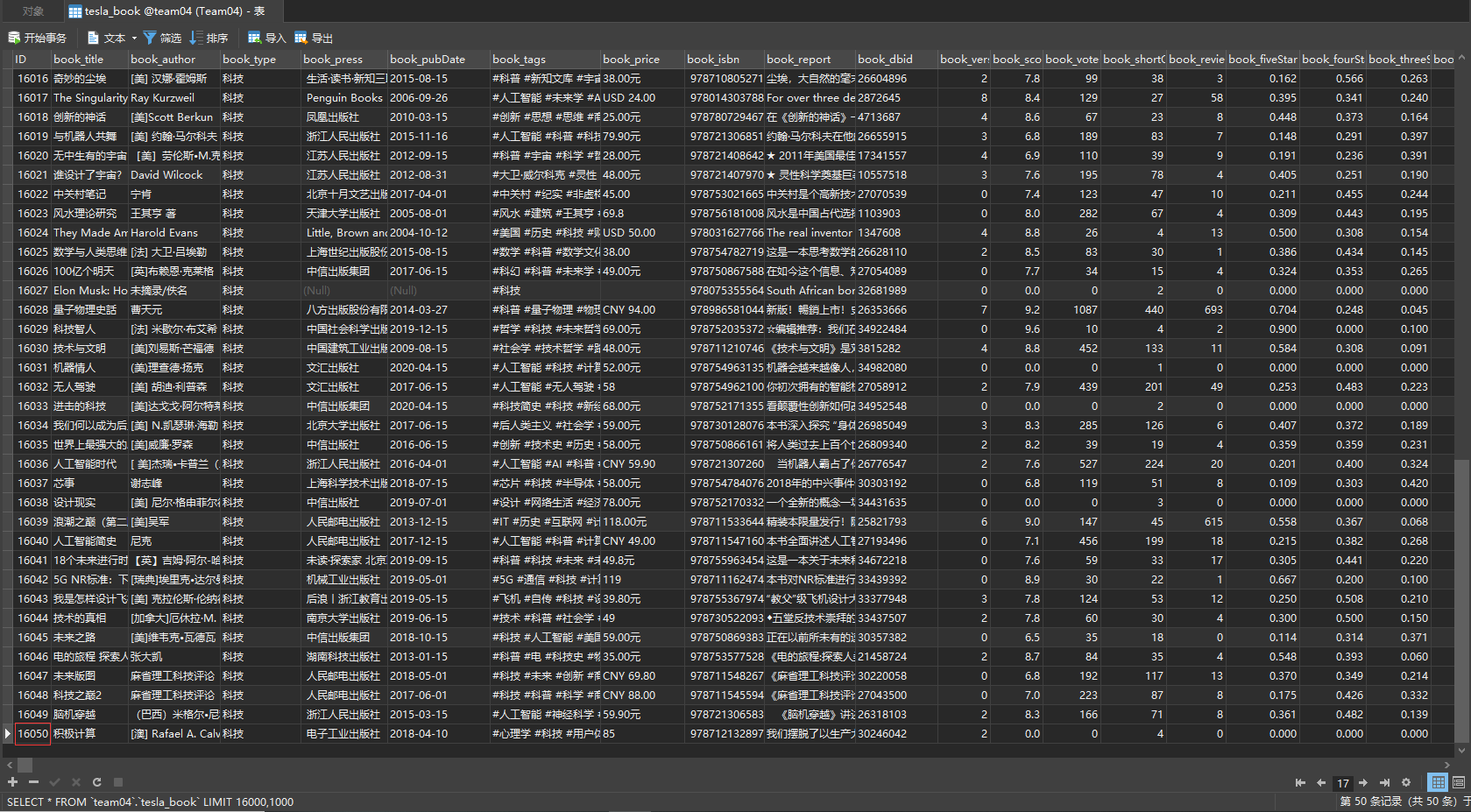
58. **def** get\_versions(response):
59. versions = response.css('div.version\_works > h2 > span:nth-child(2) > a::text').get()
60. **return** int(versions.split('全部')[1]) **if** versions **else** 0

63. **def** get\_score(response):
64. score = response.xpath('//div[@class="rating\_self clearfix"]/strong/text()').get().strip()
65. **return** float(score) **if** len(score) > 1 **else** 0.0

68. **def** get\_hotcommcontent(response):
69. content\_list = response.xpath(
70. '//div[@'
71. 'class="comment-list hot show"]/ul/li[1]/div/p/span/text()').get()
72. content = ''
73. **for** s **in** content\_list:
74. content += s
75. **return** content

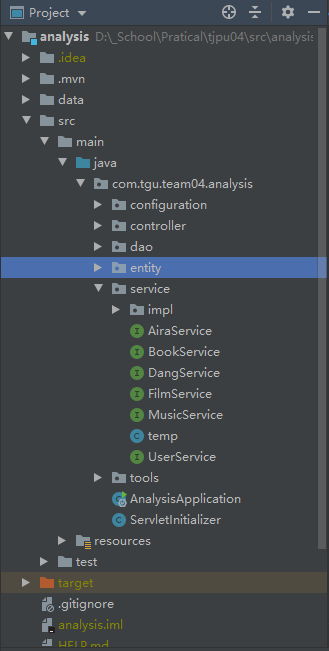
以上部分为所用到的爬取函数，这些都是书籍信息里较复杂的项目。

###### 4.1.4 成果展示



#### 4.2豆瓣读书模块后端数据接口功能实现

###### 4.2.1 项目部署目录结构



###### 4.2.2 核心代码

Entity层：TableDatar.class 用于包装服务器端返回给前端的信息，包括返回代码，返回是否成功，如果成功返回合适数据等

1. **package** com.tgu.team04.analysis.entity;
3. **import** lombok.Data;
5. **import** java.util.List;
7. @Data
8. **public** **class** TableData {
10. **private** **int** code;
11. **private** String msg;
12. **private** **int** count;
13. **private** List<? **extends** Object> data;
14. }

Entity层：Book\_SearchDatar.class 用于储存查询选项数据并传入/search接口

1. **package** com.tgu.team04.analysis.entity;
3. **import** com.fasterxml.jackson.annotation.JsonIgnoreProperties;
4. **import** lombok.Data;
6. @Data
7. //@JsonIgnoreProperties(value = { "hibernateLazyInitializer", "handler" })
8. **public** **class** Book\_SearchData {
10. **private** String title;       // 书名
11. **private** String author;      // 作者
12. **private** String press;       // 出版社
13. **private** Integer numOfType;  // 类别个数
14. **private** String[] type;      // 类别
15. **private** String isbn;        // ISBN号
16. **private** Integer minVotes;   // 最小评价数
17. **private** Float minScore;     // 最小豆瓣评分
18. **private** Float MaxScore;     // 最大豆瓣评分
20. **private** Integer page;           // 页码
21. **private** Integer limit;          // 每页数量
23. // 通过前端发送type0字符串构建type字符串数组，方便后续查询
24. **public** **void** setType(String type0) {
25. **this**.type = type0.split(" ");
26. }
27. }

Entity层：Book\_ResultDatar.class 用于储存查询的结果，封装成需要展示的数据，传给前端展示给用户

1. **package** com.tgu.team04.analysis.entity;
3. **import** com.fasterxml.jackson.annotation.JsonIgnoreProperties;
4. **import** lombok.Data;
6. @Data
7. //@JsonIgnoreProperties(value = { "hibernateLazyInitializer", "handler" })
8. **public** **class** Book\_ResultData {
9. // 两表格通用
10. **private** Integer id;
11. **private** String title;
13. // 表格一数据：书籍信息
14. **private** String author;
15. **private** String press;
16. **private** String type;
17. **private** Float score;
18. **private** Integer votes;
20. // 表格二数据：热评信息
21. **private** String hotCommContent;
22. **private** String hotCommPeople;
23. **private** String hotCommScore;
24. **private** Integer hotCommVotes;
25. }

Entity层：Book\_DataScorer.class 用于向前端返回**六类别书籍的平均评分分析**，该项分析是本模块分析页面中分析数据里相对最复杂的一个，也是本模块的亮点之一

1. **package** com.tgu.team04.analysis.entity;
3. **import** lombok.Data;
5. **import** java.util.List;
7. @Data
8. **public** **class** Book\_DataScore {
9. **private** Float score;
10. **private** Float fiveStar;
11. **private** Float fourStar;
12. **private** Float threeStar;
13. **private** Float twoStar;
14. **private** Float oneStar;
16. **private** String type;
18. //    public void setAnalysisData(Book\_DataScore0 data0) {
19. //        this.analysisData.add(data0.getScore());
20. //        this.analysisData.add(data0.getFiveStar());
21. //        this.analysisData.add(data0.getFourStar());
22. //        this.analysisData.add(data0.getThreeStar());
23. //        this.analysisData.add(data0.getTwoStar());
24. //        this.analysisData.add(data0.getOneStar());
25. //
26. //        this.type = data0.getType();
27. //    }
28. }

Entity层：Book\_DataHotr.class 用于向前端返回**热度TOP10作者的数据分析**，该项分析是本模块分析的亮点之一

1. **package** com.tgu.team04.analysis.entity;
3. **import** lombok.Data;
5. **import** java.util.List;
7. @Data
8. **public** **class** Book\_DataHot {
9. **private** Integer hot;
10. **private** Float votes;
11. **private** Float score;
13. **private** String author;
14. }

Entity层：Book\_DataReview.class 用于向前端返回**各出版社出版书籍的平均书评数与短评数分析**

1. **package** com.tgu.team04.analysis.entity;
3. **import** lombok.Data;
5. @Data
6. **public** **class** Book\_DataReview {
8. **private** String press;
10. **private** Float reviews;
11. **private** Float shortComms;
13. }

Controller层：BookController.class

1. **package** com.tgu.team04.analysis.controller;
3. **import** com.tgu.team04.analysis.entity.\*;
4. **import** com.tgu.team04.analysis.service.BookService;
5. **import** org.springframework.beans.factory.annotation.Autowired;
6. **import** org.springframework.stereotype.Controller;
7. **import** org.springframework.web.bind.annotation.RequestBody;
8. **import** org.springframework.web.bind.annotation.RequestMapping;
9. **import** org.springframework.web.bind.annotation.ResponseBody;

12. **import** java.util.List;
14. @Controller
15. @RequestMapping("/doubanBook")
16. **public** **class** BookController {
18. @Autowired
19. **private** BookService service;
21. @RequestMapping("/search")
22. @ResponseBody
23. **public** TableData search(String title, String author, String press,
24. **int** numOfType, String type, String isbn,
25. **int** minVotes, **float** minScore, **float** maxScore,
26. **int** page, **int** limit) {
27. Book\_SearchData searchData = **new** Book\_SearchData();
28. searchData.setTitle(title);
29. searchData.setAuthor(author);
30. searchData.setPress(press);
31. searchData.setNumOfType(numOfType);
32. searchData.setType(type);   // 用split函数来完成将前端发送的字符串型type转换为字符串数组，方便后面查询
33. searchData.setIsbn(isbn);
34. searchData.setMinVotes(minVotes);
35. searchData.setMinScore(minScore);
36. searchData.setMaxScore(maxScore);
38. searchData.setPage(page);
39. searchData.setLimit(limit);
41. System.out.println(searchData);
42. System.out.println("Page: " + page + ", limit: " + limit);
44. TableData tableData = **new** TableData();
45. List<Book\_ResultData> resultDataList = service.searchByOptions(searchData);
46. **if** (resultDataList != **null** && resultDataList.size() != 0) {
47. tableData.setCode(1000);
48. tableData.setMsg("查询成功");
49. tableData.setCount(service.searchCount(searchData));
50. tableData.setData(resultDataList);
51. } **else** {
52. tableData.setCode(2000);
53. tableData.setMsg("无结果");
54. tableData.setCount(0);
55. tableData.setData(**null**);
56. }
57. **return** tableData;
58. }
60. @RequestMapping("/analysis")
61. @ResponseBody
62. **public** TableData analysisData(String type){
63. TableData analysisData = **new** TableData();
65. **switch** (type) {
66. **case** "typeScore":
67. List<Book\_DataScore> resultDataList\_typeScore = service.analysisTypeScore();
68. //                System.out.println(resultDataList\_typeScore);
69. **if** (resultDataList\_typeScore != **null**){
70. analysisData.setCode(1000);
71. analysisData.setMsg("分析成功");
72. analysisData.setData(resultDataList\_typeScore);
73. }**else** {
74. analysisData.setCode(2000);
75. analysisData.setMsg("分析失败");
76. analysisData.setData(**null**);
77. }
78. **break**;
79. **case** "hotAuthor":
80. List<Book\_DataHot> resultDataList\_hotAuthor = service.analysisHotAuthor();
81. System.out.println(resultDataList\_hotAuthor);
82. **if** (resultDataList\_hotAuthor != **null**){
83. analysisData.setCode(1000);
84. analysisData.setMsg("分析成功");
85. analysisData.setData(resultDataList\_hotAuthor);
86. }**else** {
87. analysisData.setCode(2000);
88. analysisData.setMsg("分析失败");
89. analysisData.setData(**null**);
90. }
91. **break**;
92. **case** "reviewPress":
93. List<Book\_DataReview> resultDataList\_reviewPress = service.analysisReviewPress();
94. System.out.println(resultDataList\_reviewPress);
95. **if** (resultDataList\_reviewPress != **null**){
96. analysisData.setCode(1000);
97. analysisData.setMsg("分析成功");
98. analysisData.setData(resultDataList\_reviewPress);
99. }**else** {
100. analysisData.setCode(2000);
101. analysisData.setMsg("分析失败");
102. analysisData.setData(**null**);
103. }
104. **break**;
105. **default**:
106. List<Book\_Data> resultDataList = service.analysisBook(type);
107. **if** (resultDataList != **null**){
108. analysisData.setCode(1000);
109. analysisData.setMsg("分析成功");
110. analysisData.setData(resultDataList);
111. }**else** {
112. analysisData.setCode(2000);
113. analysisData.setMsg("分析失败");
114. analysisData.setData(**null**);
115. }
116. }
118. **return** analysisData;
119. }
120. }

Controller层中的分析接口采用了接收type用于switch判断的方式，可以节省写多个接口的工作量

Service层：BookService.class 提供各类接口

1. **package** com.tgu.team04.analysis.service;
3. **import** com.tgu.team04.analysis.entity.\*;
5. **import** java.util.List;
7. **public** **interface** BookService {
9. //    List<Book> search(int page, int limit);
11. List<Book\_ResultData> searchByOptions(Book\_SearchData searchData);
13. **int** searchCount(Book\_SearchData searchData);
15. List<Book\_Data> analysisBook(String type);
17. List<Book\_DataScore> analysisTypeScore();
19. List<Book\_DataHot> analysisHotAuthor();
21. List<Book\_DataReview> analysisReviewPress();
22. }

Service层：BookServiceImpl.class 实现BookService.class

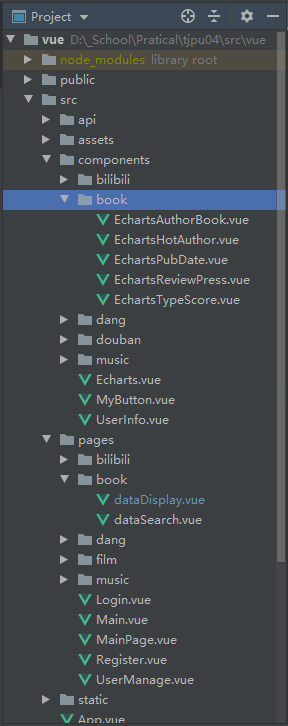
1. **package** com.tgu.team04.analysis.service.impl;
3. **import** com.tgu.team04.analysis.dao.BookMapper;
4. **import** com.tgu.team04.analysis.entity.\*;
5. **import** com.tgu.team04.analysis.service.BookService;
6. **import** org.springframework.beans.factory.annotation.Autowired;
7. **import** org.springframework.stereotype.Service;
9. **import** java.util.List;
11. @Service
12. **public** **class** BookServiceImpl **implements** BookService {
14. @Autowired
15. **private** BookMapper mapper;
17. @Override
18. **public** List<Book\_ResultData> searchByOptions(Book\_SearchData searchData) {
19. // 对四个String类型的查询属性做模糊查询预处理
20. **if** (searchData.getTitle() != **null** && !"".equals(searchData.getTitle().trim())) {
21. searchData.setTitle("%" + searchData.getTitle() + "%");
22. }
23. **if** (searchData.getAuthor() != **null** && !"".equals(searchData.getAuthor().trim())) {
24. searchData.setAuthor("%" + searchData.getAuthor() + "%");
25. }
26. **if** (searchData.getPress() != **null** && !"".equals(searchData.getPress().trim())) {
27. searchData.setPress("%" + searchData.getPress() + "%");
28. }
29. **if** (searchData.getIsbn() != **null** && !"".equals(searchData.getIsbn().trim())) {
30. searchData.setIsbn("%" + searchData.getIsbn() + "%");
31. }
32. System.out.println("============");
33. System.out.println(searchData);
34. System.out.println("============");
35. // 处理查询条数 为分页准备
36. **if** (searchData.getPage() > 0 && searchData.getLimit() > 0){
37. **return** mapper.selectByWhere(searchData, (searchData.getPage() - 1)\*searchData.getLimit(), searchData.getLimit());
38. }
39. **return** mapper.selectByWhere(searchData, **null**, **null**);
40. }
42. @Override
43. **public** **int** searchCount(Book\_SearchData searchData) {
44. **return** mapper.countSelectByWhere(searchData);
45. }
47. @Override
48. **public** List<Book\_Data> analysisBook(String type) {
49. **if** (type == **null**) **return** **null**;
51. **switch** (type) {
52. **case** "author":
53. **return** mapper.analysisAuthor();
54. **case** "pubDate":
55. **return** mapper.analysisPubDate();
56. //            case "AvgClass":
57. //                return mapper.avgclass();
58. //            case "commentpr":
59. //                return mapper.commentpr();
60. //            case "yearBook":
61. //                return mapper.yearBook();
63. }
64. **return** **null**;
65. }
67. @Override
68. **public** List<Book\_DataScore> analysisTypeScore() {
69. **return** mapper.analysisTypeScore();
70. }
72. @Override
73. **public** List<Book\_DataHot> analysisHotAuthor() {
74. **return** mapper.analysisHotAuthor();
75. }
77. @Override
78. **public** List<Book\_DataReview> analysisReviewPress() {
79. **return** mapper.analysisReviewPress();
80. }
81. }

Mapper层：BookMapper.class 用于连接Mybatis，通过接口传来的数据使用Mybatis查询或分析想要的数据并返回

1. **package** com.tgu.team04.analysis.dao;
3. **import** com.tgu.team04.analysis.entity.\*;
4. **import** org.apache.ibatis.annotations.Mapper;
5. **import** org.apache.ibatis.annotations.Param;
6. **import** org.apache.ibatis.annotations.Select;
7. **import** org.springframework.stereotype.Repository;
9. **import** java.util.List;
11. @Repository
12. @Mapper
13. **public** **interface** BookMapper {
15. @Select("<script>" +
16. "select ID, book\_title as title, book\_author as author, " +
17. "book\_press as press, book\_type as type, book\_score as score, book\_votes as votes, "+
18. "hotComm\_content, hotComm\_people, hotComm\_score, hotComm\_votes" +
19. "   from tesla\_book" +
20. // 【条件】
21. "<where>" +
22. // 书名
23. "   <if test='searchData.title != null and searchData.title.length > 0'>" +
24. "       AND book\_title like #{searchData.title}" +
25. "   </if>" +
26. // 作者
27. "   <if test='searchData.author != null and searchData.author.length > 0'>" +
28. "       AND book\_author like #{searchData.author}" +
29. "   </if>" +
30. // 出版社
31. "   <if test='searchData.press != null and searchData.press.length > 0'>" +
32. "       AND book\_press like #{searchData.press}" +
33. "   </if>" +
34. // 类型
35. "   <if test='searchData.numOfType > 0'>" +
36. "       AND book\_type IN" +
37. "       <foreach collection='searchData.type' item='type' index='index' open='(' close = ')' separator=', '>" +
38. "           #{type}" +
39. "       </foreach>" +
40. "   </if>" +
41. // ISBN号
42. "   <if test='searchData.isbn != null and searchData.isbn.length > 0'>" +
43. "       AND book\_isbn like #{searchData.isbn}" +
44. "   </if>" +
45. // 评价人数
46. "       AND book\_votes >= #{searchData.minVotes}" +
47. // 评分区间
48. "       AND book\_score BETWEEN #{searchData.minScore} AND #{searchData.MaxScore}" +
49. "</where>" +
50. // 页数处理
51. "<if test='start != null and limit != null'>" +
52. "   limit #{start}, #{limit}" +
53. "</if>" +
54. "</script>")
55. List<Book\_ResultData> selectByWhere(@Param("searchData") Book\_SearchData searchData, @Param("start") Integer start, @Param("limit") Integer limit);
57. @Select("<script>" +
58. "select count(1)" +
59. "   from tesla\_book" +
60. // 【条件】
61. "<where>" +
62. // 书名
63. "   <if test='searchData.title != null and searchData.title.length > 0'>" +
64. "       AND book\_title like #{searchData.title}" +
65. "   </if>" +
66. // 作者
67. "   <if test='searchData.author != null and searchData.author.length > 0'>" +
68. "       AND book\_author like #{searchData.author}" +
69. "   </if>" +
70. // 出版社
71. "   <if test='searchData.press != null and searchData.press.length > 0'>" +
72. "       AND book\_author like #{searchData.press}" +
73. "   </if>" +
74. // 类型
75. "   <if test='searchData.numOfType > 0'>" +
76. "       AND book\_type IN" +
77. "       <foreach collection='searchData.type' item='type' index='index' open='(' close = ')' separator=', '>" +
78. "           #{type}" +
79. "       </foreach>" +
80. "   </if>" +
81. // ISBN号
82. "   <if test='searchData.isbn != null and searchData.press.length > 0'>" +
83. "       AND book\_isbn like #{searchData.press}" +
84. "   </if>" +
85. // 评价人数
86. "       AND book\_votes >= #{searchData.minVotes}" +
87. // 评分区间
88. "       AND book\_score BETWEEN #{searchData.minScore} AND #{searchData.MaxScore}" +
90. "</where>" +
91. "</script>")
92. **int** countSelectByWhere(@Param("searchData")Book\_SearchData searchData);
94. @Select("select book\_author as analysisData, count(1) as count " +
95. "from tesla\_book where book\_author <> '无'" +
96. "group by book\_author order by count desc limit 0, 25")
97. List<Book\_Data> analysisAuthor();
99. @Select("select DATE\_FORMAT(book\_pubDate,'%Y-%m') as analysisData ,count(1) as count \n" +
100. "from tesla\_book WHERE book\_pubDate\n" +
101. "group by DATE\_FORMAT(book\_pubDate,'%Y-%m') \n" +
102. "order by DATE\_FORMAT(book\_pubDate,'%Y-%m')")
103. List<Book\_Data> analysisPubDate();
105. @Select("SELECT AVG(book\_score) AS score, AVG(book\_fiveStar) AS fiveStar,\n" +
106. "\t\tAVG(book\_fourStar) AS fourStar, AVG(book\_threeStar) AS threeStar,\n" +
107. "\t\tAVG(book\_twoStar) AS twoStar, AVG(book\_oneStar) AS oneStar,\n" +
108. "\t\tbook\_type AS type FROM tesla\_book GROUP BY book\_type;")
109. List<Book\_DataScore> analysisTypeScore();
111. @Select("SELECT SUM(book\_readings + book\_reads + book\_toReads) AS hot,\n" +
112. "\tAVG(book\_votes) AS votes, AVG(book\_score) AS score, book\_author \n" +
113. "\tAS author FROM tesla\_book GROUP BY book\_author\n" +
114. "ORDER BY hot DESC LIMIT 0, 10\n;")
115. List<Book\_DataHot> analysisHotAuthor();
117. @Select("SELECT book\_press AS press, AVG(book\_reviews) AS reviews, \n" +
118. "\tAVG(book\_shortComms) AS shortComms FROM tesla\_book \n" +
119. "\tWHERE book\_press NOT LIKE '%      %'  \n" +
120. "\tGROUP BY press HAVING COUNT(\*) > 5\n" +
121. "\tORDER BY reviews DESC")
122. List<Book\_DataReview> analysisReviewPress();
123. }

#### 4.3前端界面功能实现及成果展示

###### 4.3.1 项目部署目录结构



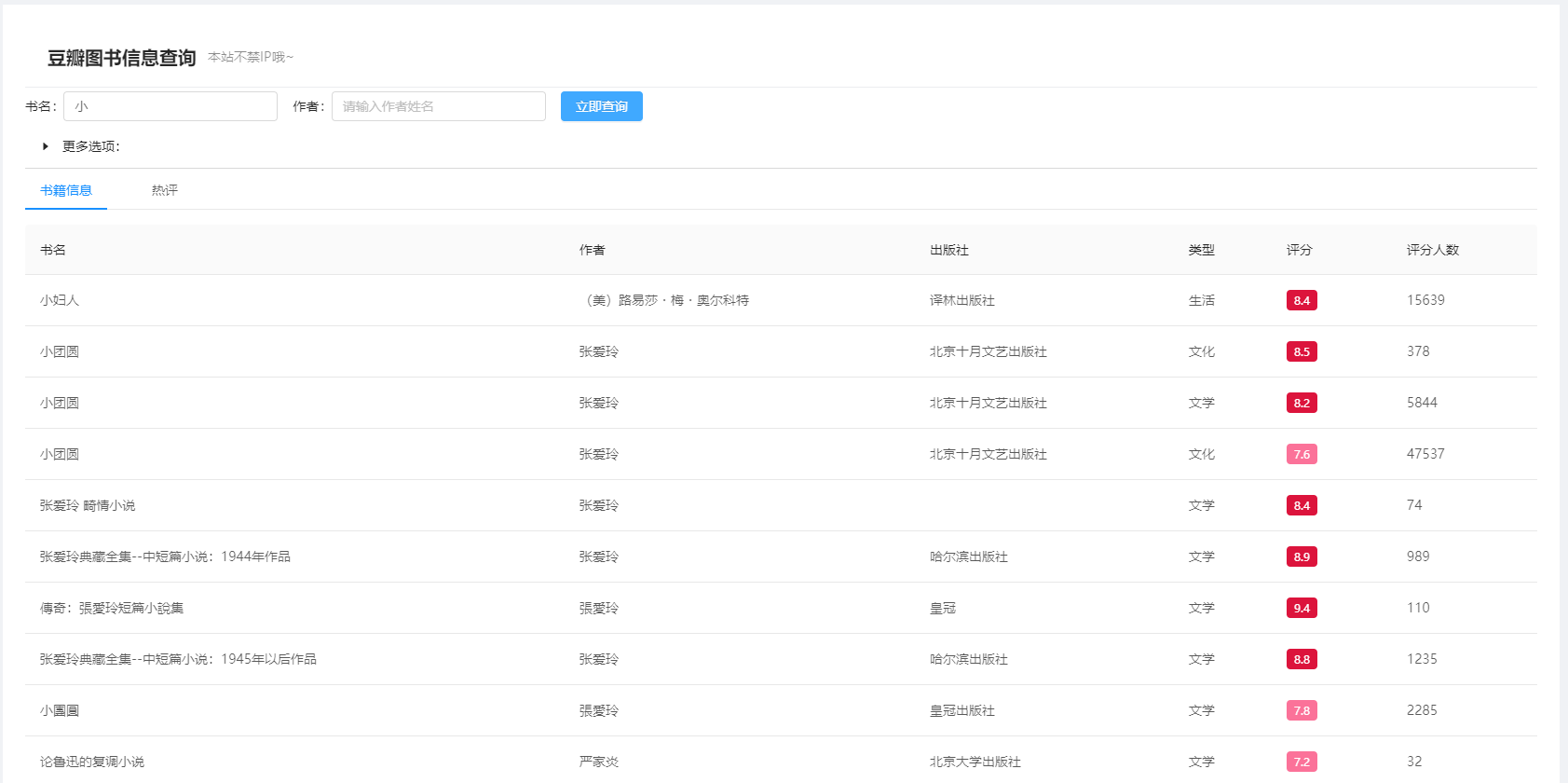
**由于前端文件代码过长，仅放入本模块亮点部分代码**

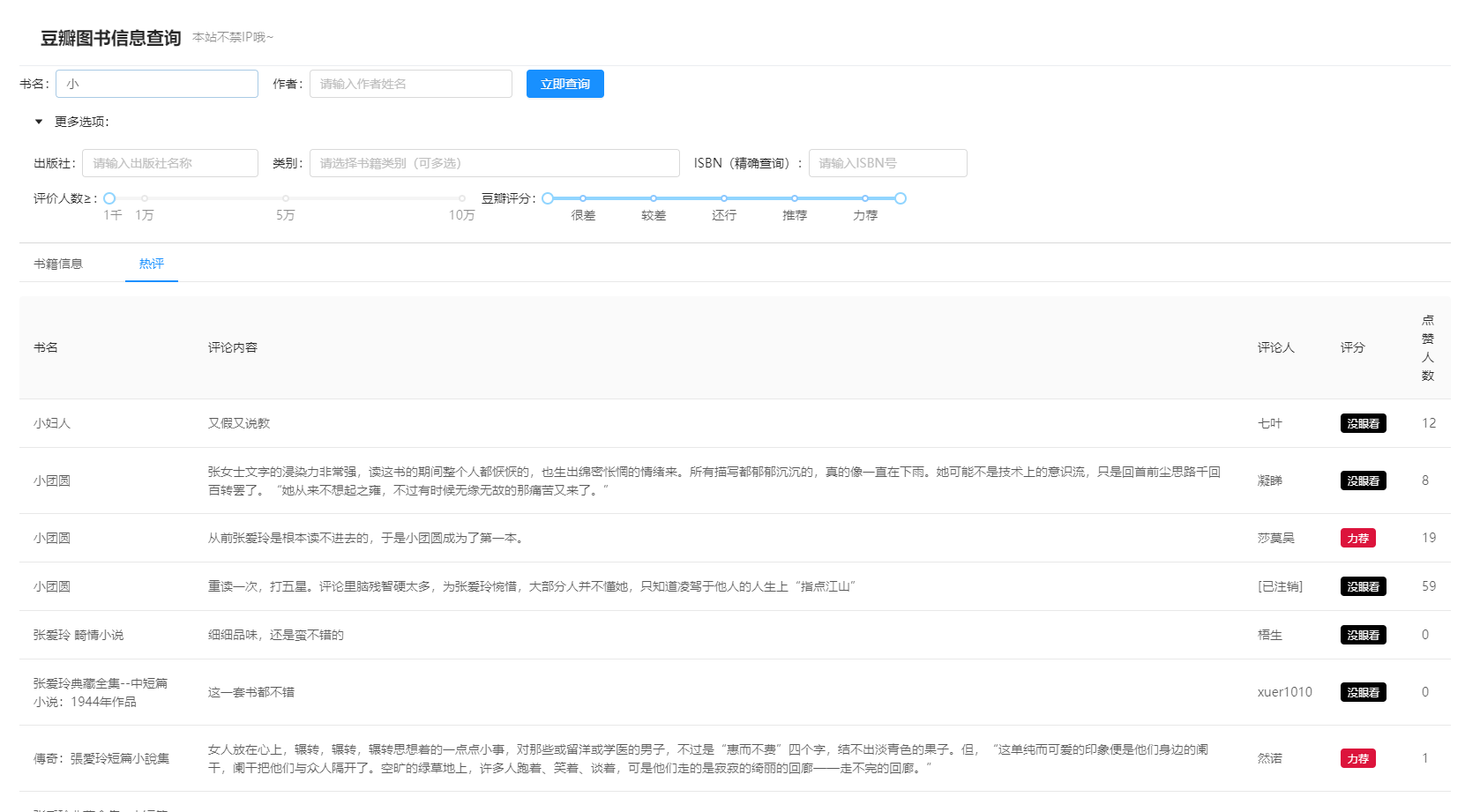
###### 4.3.2 数据查询界面-核心代码与成果展示

Page：查询页面，提供的更多高级选项查询，其中包括出版社模糊查询，**在已有分类中，不选（默认全选查询）或者多选查询**，ISBN号精确查询，**滑动条查询评论数大于一定数量的书籍以及在评分在一定区间内的书籍**

1. **<a-collapse** :bordered="false" :style="style\_collapse"**>**
2. **<template** #expandIcon="props"**>**
3. **<a-icon** type="caret-right" :rotate="props.isActive ? 90 : 0" **/>**
4. **</template>**
5. <!--            取消箭头展示-->
6. <!--            :show-arrow="false"-->
7. **<a-collapse-panel**
8. header="更多选项："
10. **>**
11. **<a-form** layout="inline"**>**
12. **<a-form-item**
13. label="出版社"
14. :label-col="formItemLayout.labelCol"
15. :wrapper-col="formItemLayout.wrapperCol"
16. **>**
17. **<a-input** placeholder="请输入出版社名称" style="width: 200px" v-model="press"**/>**
18. **</a-form-item>**
19. **<a-form-item**
20. label="类别"
21. :label-col="formItemLayout.labelCol"
22. :wrapper-col="formItemLayout.wrapperCol"
23. **>**
24. <!--                @change中是当option改变时执行的函数
25. 可以自行打印参数（应该有默认参数）看看获得的是什么
26. 如果我记得没错 会获得 每个选项的 value 和 内容
27. 例如：
28. {key： -1, content: 全部查询}
29. --**>**
30. **<a-select** mode="multiple"
31. style="width: 420px"
32. placeholder="请选择书籍类别（可多选）"
33. @change="onChangeType"
34. **>**
35. **<a-select-option** v-for='type in options\_type' :key="type"**>**
36. {{type}}
37. **</a-select-option>**
38. **</a-select>**
39. **</a-form-item>**
40. **<a-form-item**
41. label="ISBN（精确查询）"
42. :label-col="formItemLayout.labelCol"
43. :wrapper-col="formItemLayout.wrapperCol"
44. **>**
45. **<a-input** placeholder="请输入ISBN号" style="width: 180px" v-model="isbn"**/>**
46. **</a-form-item>**
47. **</a-form>**
49. **<a-form** layout="inline"**>**
50. **<a-form-item**
51. label="评价人数≥"
52. :label-col="formItemLayout.labelCol"
53. :wrapper-col="formItemLayout.wrapperCol"
54. **>**
55. **<a-slider**
56. v-decorator="['slider']"
57. :marks="{1000:'1千', 10000: '1万',50000: '5万', 100000: '10万'}"
58. style="width: 400px"
59. :default-value="0"
60. :max="100000"
61. :step="1000"
62. @change="onChangeVotes"
63. **/>**
64. **</a-form-item>**
66. **<a-form-item**
67. label="豆瓣评分"
68. :label-col="formItemLayout.labelCol"
69. :wrapper-col="formItemLayout.wrapperCol"
70. **>**
71. **<a-slider**
72. range
73. :marks="{1:'很差', 3:'较差', 5:'还行', 7:'推荐', 9:'力荐'}"
74. style="width: 400px"
75. :max="10"
76. :step="0.1"
77. :default-value="[0, 10]"
78. @change="onChangeScore"
79. **/>**
81. **</a-form-item>**
83. **</a-form>**
84. **</a-collapse-panel>**
85. **</a-collapse>**

打开 豆瓣图书-数据查询 页面，得到默认全部查询的查询结果，输入书名“小”，然后点击立即查询，得到结果如下：





点击更多选项，然后输入作者：“鲁”，出版社：“社”，类别选择：“文学”，将评价人数滑动为大于等于1千，将豆瓣评分区间滑动为5到10之间，得到如下查询结果：





###### 4.3.3 数据分析界面-核心代码及成果展示

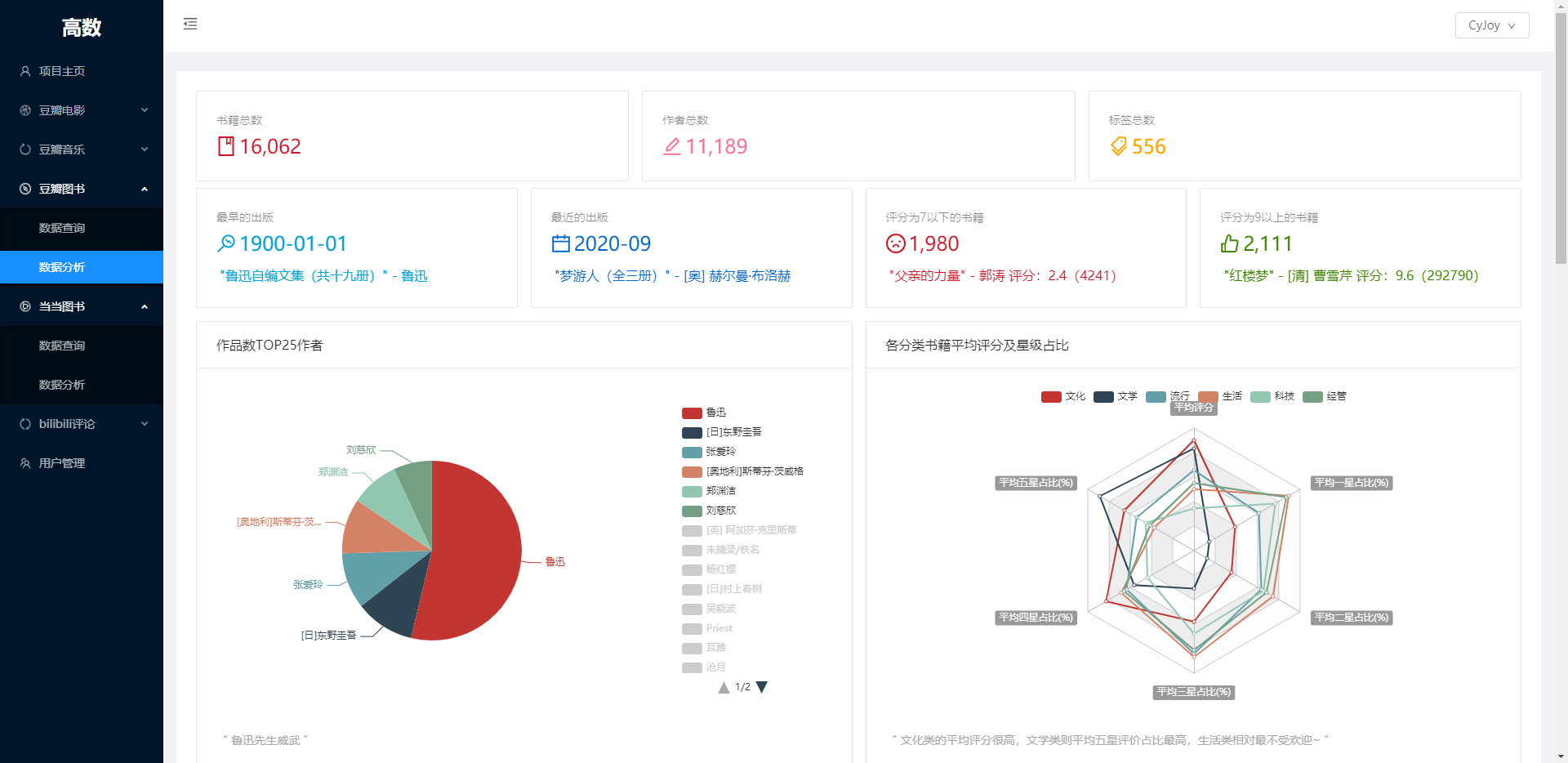
Page：数据分析页面，利用了尽可能多的数据以及功能较为多样化的eCharts样式，并做出合理分析以及分析结构点评，更直观地展示分析结果。

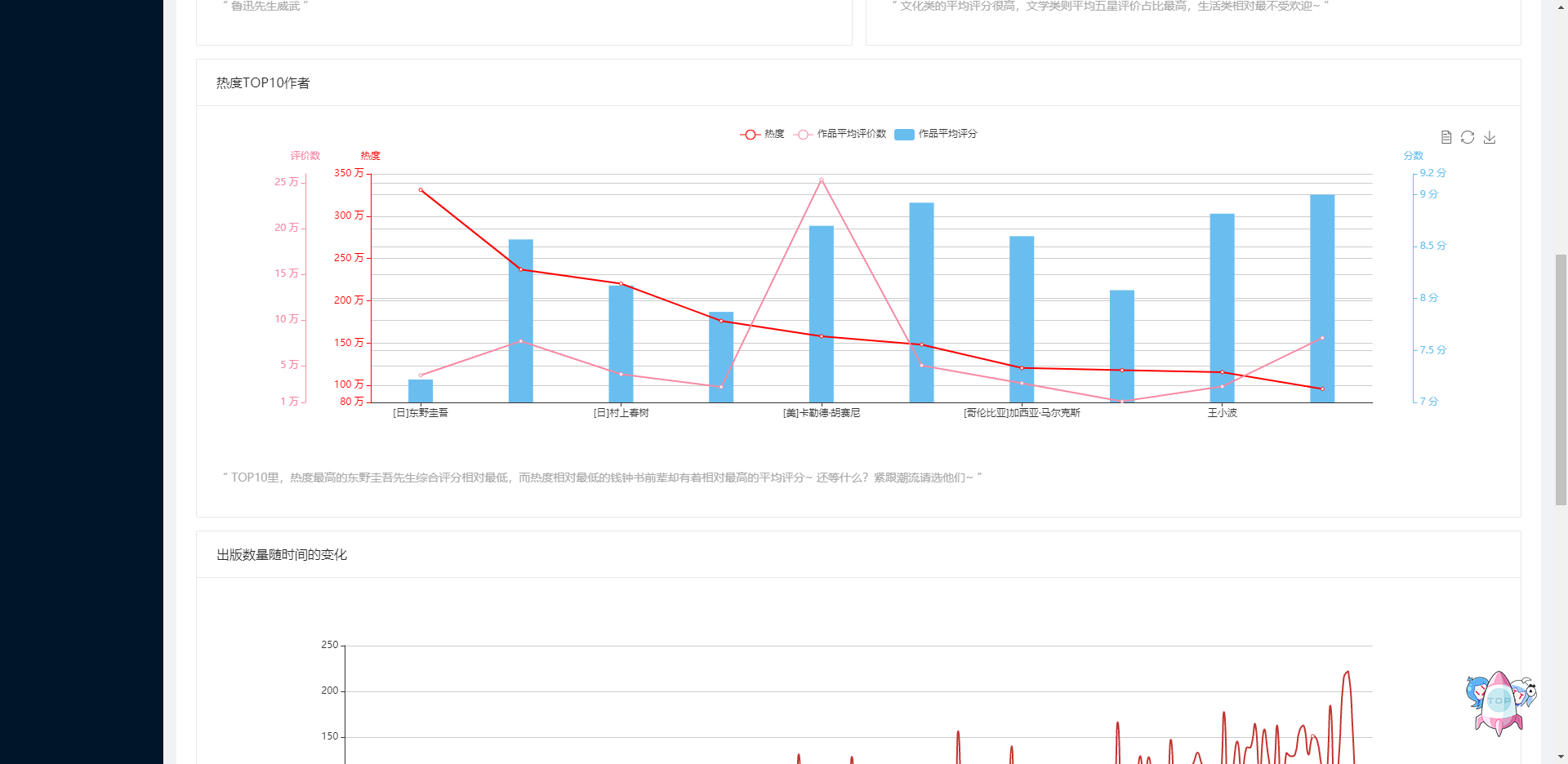
**Page代码：dataDisplay.vue：**

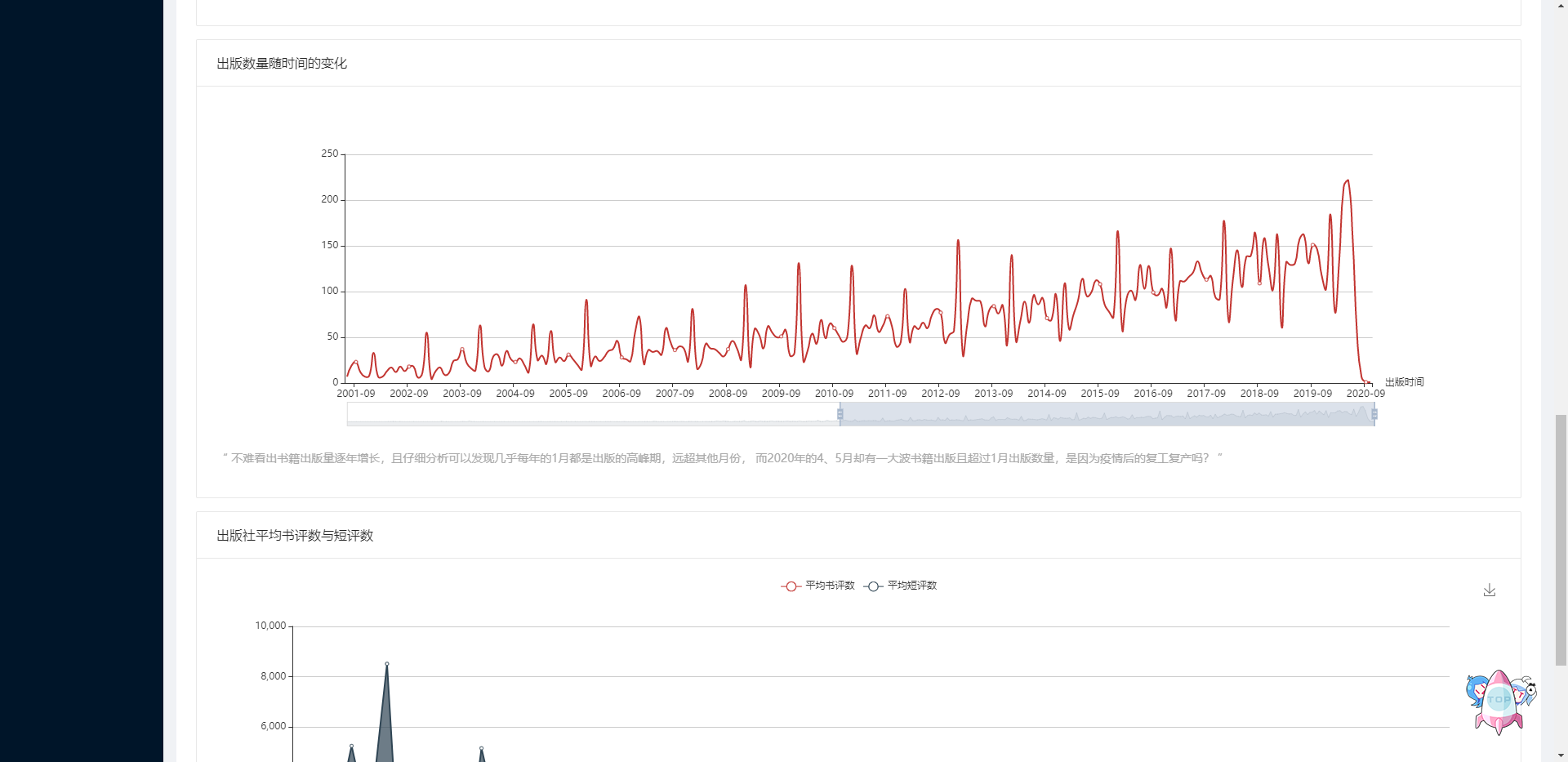
1. **<template** **>**
2. **<div** **>**
3. <!--        第一排-->
4. **<a-row** :gutter="16"**>**
5. **<a-col** :span="8"**>**
6. **<a-card** hoverable**>**
7. **<a-statistic**
8. title="书籍总数"
9. :value="16062"
10. :value-style="{ color: '#cf1322' }"
11. style="margin-right: 50px"
12. **>**
13. **<template** #prefix**>**
14. **<a-icon** type="book" **/>**
15. **</template>**
16. **</a-statistic>**
17. **</a-card>**
18. **</a-col>**
19. **<a-col** :span="8"**>**
20. **<a-card** hoverable**>**
21. **<a-statistic**
22. title="作者总数"
23. :value="11189"
24. class="demo-class"
25. :value-style="{ color: '#fb7299' }"
26. **>**
27. **<template** #prefix**>**
28. **<a-icon** type="edit" **/>**
29. **</template>**
30. **</a-statistic>**
31. **</a-card>**
32. **</a-col>**
33. **<a-col** :span="8"**>**
34. **<a-card** hoverable**>**
35. **<a-statistic**
36. title="标签总数"
37. :value="556"
38. class="demo-class"
39. :value-style="{ color: '#FFA500' }"
40. **>**
41. **<template** #prefix**>**
42. **<a-icon** type="tags" **/>**
43. **</template>**
45. **</a-statistic>**
46. **</a-card>**
47. **</a-col>**
48. **</a-row>**
50. <!--        第二排-->
51. **<a-row** :gutter="[16,16]"**>**
52. **<a-col** :span="6"**>**
53. **<a-card>**
54. **<a-statistic**
55. title="最早的出版"
56. value="1900-01-01"
57. :value-style="{ color: '#00a1d6' }"
58. style="margin-right: 50px"
59. **>**
60. **<template** #prefix**>**
61. **<a-icon** type="monitor" **/>**
62. **</template>**
63. **<template** #suffix**>**
64. **<span>**"鲁迅自编文集（共十九册）" - 鲁迅**</span>**
65. **</template>**
66. **</a-statistic>**
67. **</a-card>**
68. **</a-col>**
69. **<a-col** :span="6"**>**
70. **<a-card>**
71. **<a-statistic**
72. title="最近的出版"
73. value="2020-09"
74. class="demo-class"
75. :value-style="{ color: '#0066CC' }"
76. **>**
77. **<template** #prefix**>**
78. **<a-icon** type="calendar" **/>**
79. **</template>**
80. **<template** #suffix**>**
81. **<span>**"梦游人（全三册）" - [奥] 赫尔曼·布洛赫**</span>**
82. **</template>**
83. **</a-statistic>**
84. **</a-card>**
85. **</a-col>**
86. **<a-col** :span="6"**>**
87. **<a-card>**
88. **<a-statistic**
89. title="评分为7以下的书籍"
90. :value="1980"
91. class="demo-class"
92. :value-style="{ color: '#cf1322' }"
93. **>**
94. **<template** #prefix**>**
95. **<a-icon** type="frown"**/>**
96. **</template>**
97. **<template** #suffix**>**
98. **<span>**"父亲的力量" - 郭涛 评分：2.4（4241）**</span>**
99. **</template>**
100. **</a-statistic>**
101. **</a-card>**
102. **</a-col>**
103. **<a-col** :span="6"**>**
104. **<a-card>**
105. **<a-statistic**
106. title="评分为9以上的书籍"
107. :value="2111"
108. class="demo-class"
109. :value-style="{ color: '#3F8600' }"
110. **>**
111. **<template** #prefix**>**
112. **<a-icon** type="like" **/>**
113. **</template>**
114. **<template** #suffix**>**
115. **<span>**"红楼梦" - [清] 曹雪芹 评分：9.6（292790）**</span>**
116. **</template>**
117. **</a-statistic>**
118. **</a-card>**
119. **</a-col>**
120. **</a-row>**
122. <!--        第三排 作品数TOP25作者 各分类书籍平均评分-->
123. **<a-row** :gutter="[16,16]"**>**
124. **<a-col** :span="12" style="width: 50%"**>**
125. **<a-card** hoverable body-style="center" title="作品数TOP25作者"**>**
126. **<EchartsAuthorBook** align="center"**/>**
127. **<br/>**
128. **<p** style="color: darkgray"**>** “ 鲁迅先生威武 ” **</p>**
129. **</a-card>**
130. **</a-col>**
131. **<a-col** :span="12" style="width: 50%"**>**
132. **<a-card** hoverable body-style="center" title="各分类书籍平均评分及星级占比" **>**
133. **<EchartsTypeScore** align="center"**/>**
134. **<br/>**
135. **<p** style="color: darkgray"**>** “ 文化类的平均评分很高，文学类则平均五星评价占比最高，生活类相对最不受欢迎~ ” **</p>**
136. **</a-card>**
137. **</a-col>**
138. **</a-row>**
140. <!--        第四排 热度TOP10作者-->
141. **<a-row** :gutter="[16,16]"**>**
142. **<a-col** :span="12" style="width: 100%"**>**
143. **<a-card** hoverable body-style="center" title="热度TOP10作者"**>**
144. **<EchartsHotAuthor** align="center"**/>**
145. **<br/>**
146. **<p** style="color: darkgray"**>** “ TOP10里，热度最高的东野圭吾先生综合评分相对最低，而热度相对最低的钱钟书前辈却有着相对最高的平均评分~ 还等什么？紧跟潮流请选他们~ ” **</p>**
147. **</a-card>**
148. **</a-col>**
149. **</a-row>**
151. <!--        第五排 出版数量随时间的变化-->
152. **<a-row** :gutter="[16,16]"**>**
153. **<a-col** :span="12" style="width: 100%"**>**
154. **<a-card** hoverable body-style="center" title="出版数量随时间的变化"**>**
155. **<EchartsPubDate** align="center"**/>**
156. **<br/>**
157. **<p** style="color: darkgray"**>** “ 不难看出书籍出版量逐年增长，且仔细分析可以发现几乎每年的1月都是出版的高峰期，远超其他月份，
158. 而2020年的4、5月却有一大波书籍出版且超过1月出版数量，是因为疫情后的复工复产吗？ ” **</p>**
159. **</a-card>**
160. **</a-col>**
161. **</a-row>**
163. <!--        第六排 出版社平均书评数与短评数-->
164. **<a-row** :gutter="[16,16]"**>**
165. **<a-col** :span="12" style="width: 100%"**>**
166. **<a-card** hoverable body-style="center" title="出版社平均书评数与短评数"**>**
167. **<EchartsReviewPress** align="center"**/>**
168. **<br/>**
169. **<p** style="color: darkgray"**>** “ 北京十月文艺出版社、上海译文出版社、南海出版公司、武汉大学出版社都是读者们的宠儿~ ” **</p>**
170. **</a-card>**
171. **</a-col>**
172. **</a-row>**
173. **</div>**
174. **</template>**

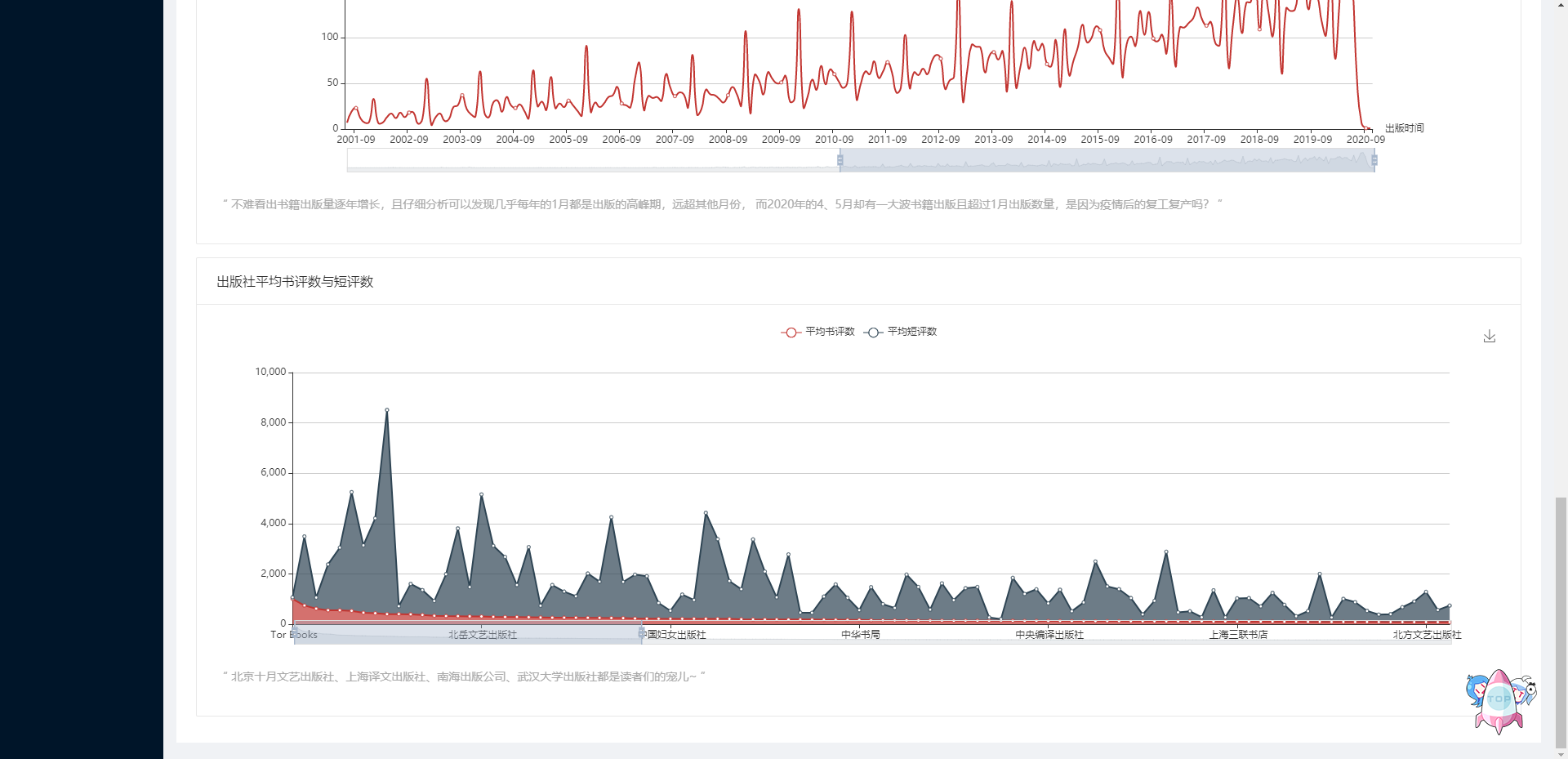
177. **<script>**
178. import EchartsAuthorBook from "../../components/book/EchartsAuthorBook";
179. import EchartsPubDate from "../../components/book/EchartsPubDate";
180. import EchartsTypeScore from "../../components/book/EchartsTypeScore";
181. import EchartsHotAuthor from "../../components/book/EchartsHotAuthor";
182. import EchartsReviewPress from "../../components/book/EchartsReviewPress";
184. export default {
185. name: "dataDisplay",
186. components: {EchartsHotAuthor, EchartsTypeScore, EchartsAuthorBook, EchartsPubDate, EchartsReviewPress}
187. }
188. **</script>**
190. **<style** scoped**>**
192. **</style>**

**点击 豆瓣图书-数据分析 进入数据分析页面，可以看到所有分析结果：**







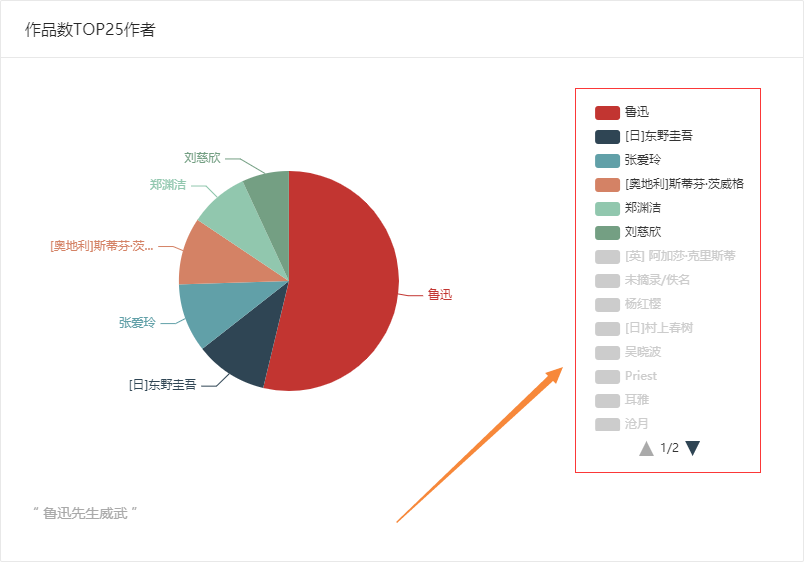


**eCharts代码：EchartsAuthorBook.vue 分析了作品数TOP25的作家**

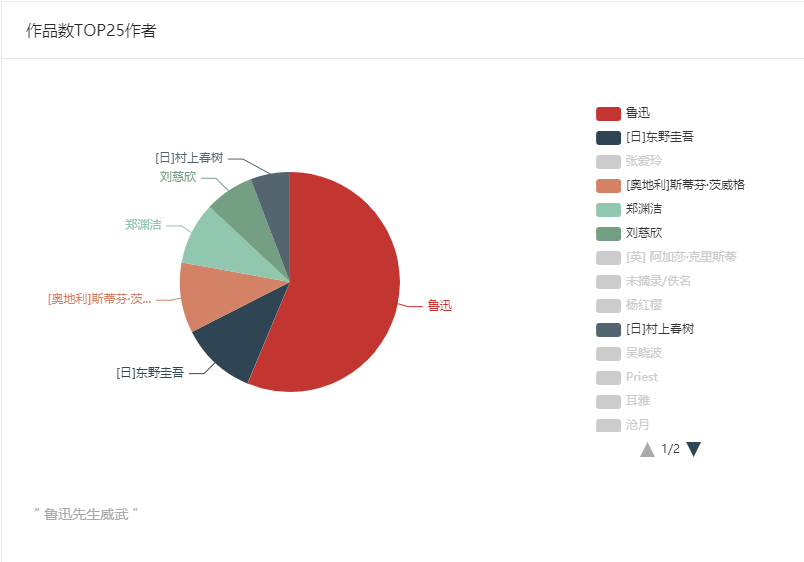
1. **<template>**
2. **<div** class="Echarts"**>**
3. **<div** id="authorBook" style="width:100%; height:400px;" **></div>**
4. **</div>**
5. **</template>**
7. **<script>**
8. import axios from "axios";
9. import qs from "qs";
10. export default {
11. name: "EchartsAuthorBook",
12. data(){
13. return{
14. analysisData\_dataList:[],
15. analysisData\_authors:[],
16. limit:[]
17. }
18. },
19. methods:{
20. myCharts(){
21. var myChart = this.$echarts.init(document.getElementById('authorBook'));
23. var option = {
24. tooltip: {
25. trigger: 'item',
26. formatter: '{a} **<br/>**{b} : {c} ({d}%)'
27. },
28. legend: {
29. type: 'scroll',
30. orient: 'vertical',
31. right: 30,
32. top: 20,
33. bottom: 20,
34. data: this.analysisData\_authors,
36. selected: this.limit
37. },
38. series: [
39. {
40. name: '作者名称:',
41. type: 'pie',
42. radius: '55%',
43. center: ['35%', '50%'],
44. data: this.analysisData\_dataList,
45. emphasis: {
46. itemStyle: {
47. shadowBlur: 10,
48. shadowOffsetX: 0,
49. shadowColor: 'rgba(0, 0, 0, 0.5)'
50. }
51. }
52. }
53. ]
54. };
56. myChart.setOption(option);
58. }

61. },
62. mounted() {
63. let result\_dataList = [];
64. let result\_authors = [];
65. let limit = {};
66. axios.post("http://localhost:8080/doubanBook/analysis", qs.stringify({
67. type: 'author'
68. }, {headers:{'Content-Type':'application/x-www-form-urlencoded'}}))
69. .then(response =**>**{
70. const resultDataList = response.data.data;
71. console.log(111);
72. console.log(resultDataList);
73. let i = 0;
74. for (let index in resultDataList){
75. let resultData = resultDataList[index];
76. result\_dataListresult\_dataList = result\_dataList.concat({
77. name:resultData.analysisData,
78. value:resultData.count
79. })
80. result\_authorsresult\_authors = result\_authors.concat(resultData.analysisData)
81. limit[resultData.analysisData] = i **<** **6**;
82. i++;
83. }
84. this.analysisData\_dataList = result\_dataList;
85. this.analysisData\_authors = result\_authors;
86. this.limit = limit;
87. console.log(111);
88. this.myCharts();
89. }).catch(error=**>**{
90. console.log(error)
91. })
92. }
93. }
94. **</script>**
96. <!--<style scoped>-->
98. <!--</style>-->

**本分析结果包括展示了TOP25的作者列表，如图：**



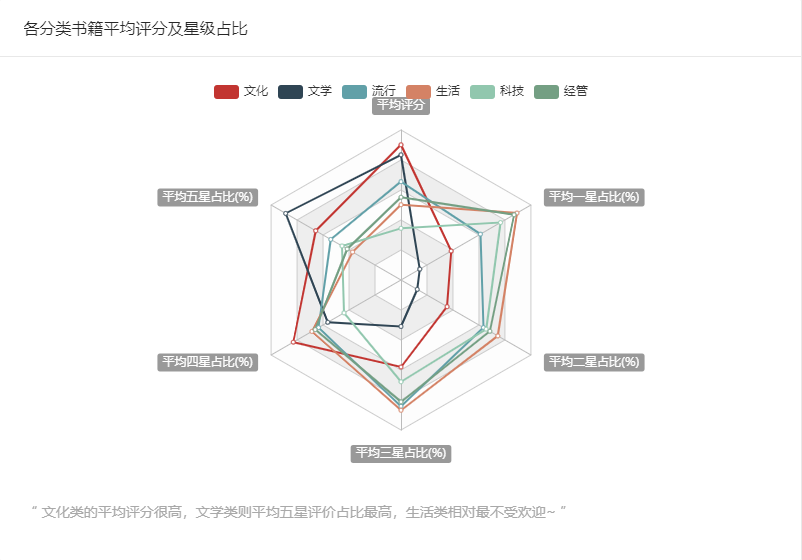
**同时可以通过在列表点击不同的作者，来使其在饼图中显示或消去，如点击张爱玲取消展示，点击村上春树进行展示，结果如图：**



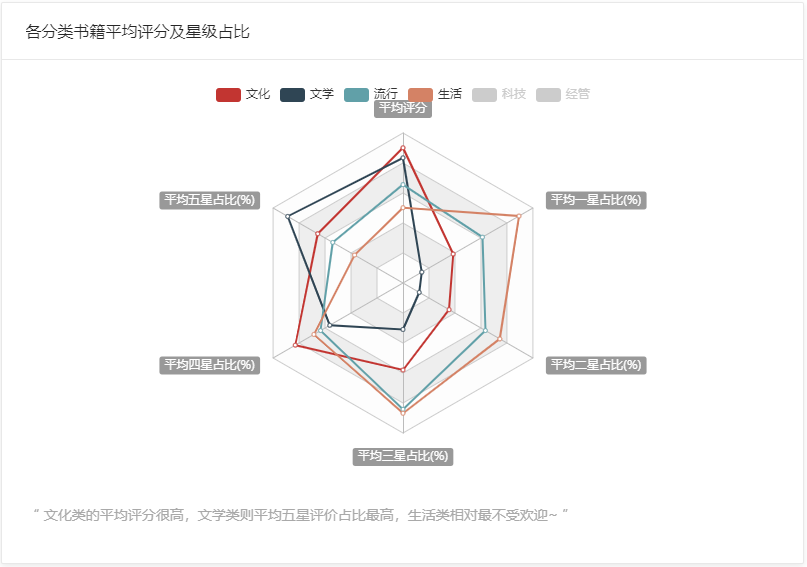
**eCharts代码：EchartsTypeScore.vue**

1. **<template>**
2. **<div** class="Echarts"**>**
3. **<div** id="typeScore" style="width:100%; height:400px;"**></div>**
4. **</div>**
5. **</template>**
7. **<script>**
8. import axios from "axios";
9. import qs from "qs";
11. export default {
12. name: "EchartsTypeScore",
13. data(){
14. return{
15. analysis\_dataList:[],
16. analysis\_type:[]
17. }
18. },
19. methods: {
20. myEcharts() {
21. // 基于准备好的dom，初始化echarts实例
22. var myChart = this.$echarts.init(document.getElementById('typeScore'));
24. var option = {
25. tooltip: {},
26. legend: {
27. data: this.analysis\_type
28. },
29. radar: {
30. // shape: 'circle',
31. name: {
32. textStyle: {
33. color: '#fff',
34. backgroundColor: '#999',
35. borderRadius: 3,
36. padding: [3, 5]
37. }
38. },
39. indicator: [
40. {name: '平均评分', min: 7, max: 8.2},
41. {name: '平均五星占比(%)', min: 28, max: 48},
42. {name: '平均四星占比(%)', min: 27.5, max: 40},
43. {name: '平均三星占比(%)', min: 7, max: 21},
44. {name: '平均二星占比(%)', min: 0.75, max: 4},
45. {name: '平均一星占比(%)', min: 0.3, max: 1.5},
46. // {name: '平均评分', min: 7.2, max: 8.2},
47. // {name: '平均五星占比(%)', min: 32, max: 48},
48. // {name: '平均四星占比(%)', min: 32, max: 40},
49. // {name: '平均三星占比(%)', min: 9.5, max: 22},
50. // {name: '平均二星占比(%)', min: 0.95, max: 4},
51. // {name: '平均一星占比(%)', min: 0.4, max: 1.5},
52. ]
53. },
54. series: [{
55. name: '六个核桃（x）',
56. type: 'radar',
57. // areaStyle: {normal: {}},
58. data: this.analysis\_dataList
59. }]
60. };
61. myChart.setOption(option);
62. }
63. },
64. mounted() {
65. let result\_dataList = [];
66. let result\_type = [];
67. axios.post("http://localhost:8080/doubanBook/analysis", qs.stringify({
68. type: 'typeScore'
69. }, {headers:{'Content-Type':'application/x-www-form-urlencoded'}}))
70. .then(response =**>** {
71. const resultDataList = response.data.data;
72. console.log(222);
73. console.log(resultDataList)
74. for(let index in resultDataList){
75. let resultData = resultDataList[index];
76. let t = [];
77. tt = t.concat(resultData.score);
78. tt = t.concat(resultData.fiveStar \* 100);
79. tt = t.concat(resultData.fourStar \* 100);
80. tt = t.concat(resultData.threeStar \* 100);
81. tt = t.concat(resultData.twoStar \* 100);
82. tt = t.concat(resultData.oneStar \* 100);
84. result\_typeresult\_type = result\_type.concat(resultData.type);
85. result\_dataListresult\_dataList = result\_dataList.concat({value:t, name:resultData.type});
86. }
87. this.analysis\_type = result\_type;
88. this.analysis\_dataList = result\_dataList;
89. console.log(this.analysis\_type);
90. console.log(this.analysis\_dataList);
91. console.log(222);
92. this.myEcharts();
93. }).catch(error =**>** {
94. console.log(error)
95. })
97. }
98. }
100. **</script>**
102. **<style** scoped**>**
104. **</style>**

**本模块亮点之一，综合分析了所有类别下平均豆瓣评分、以及各星级的平均占比，结果如图：**



**同样的，可以通过点击上面的标签来使某一类别的数据取消展示或者恢复展示，点击经管和科技，结果如图：**

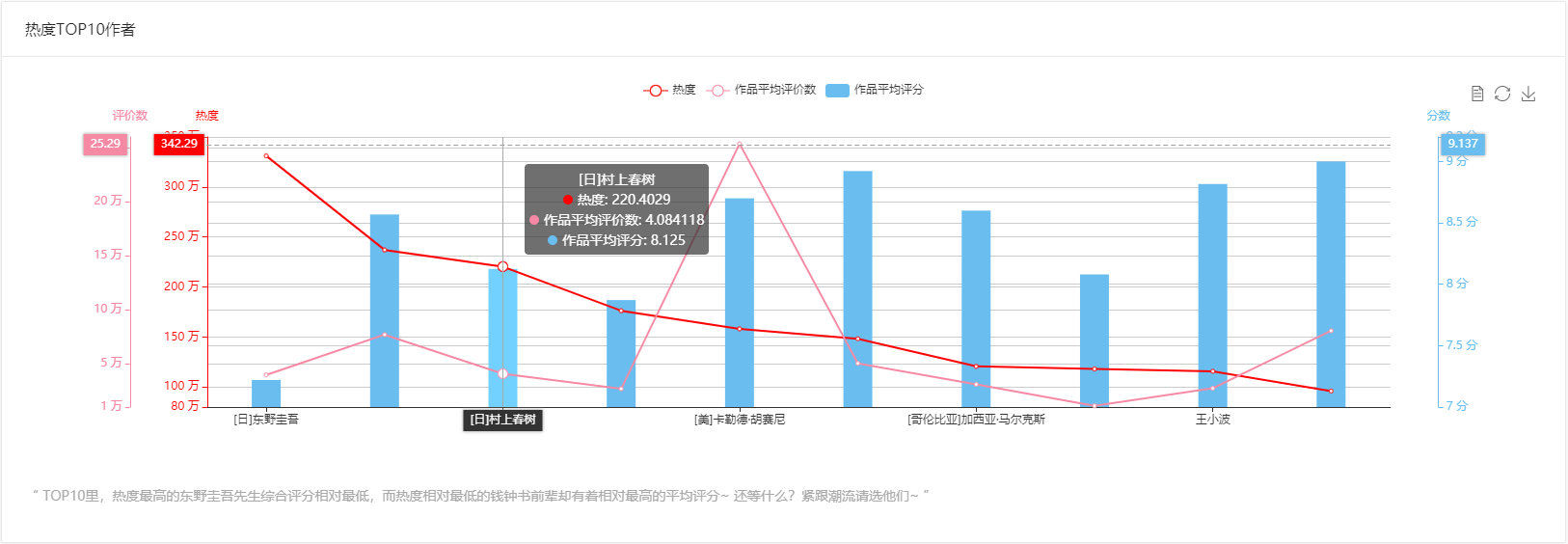


**通过本分析，得出了文化和文学类书籍综合评分最高最受欢迎，而生活类的书籍相对最不受欢迎的结论**

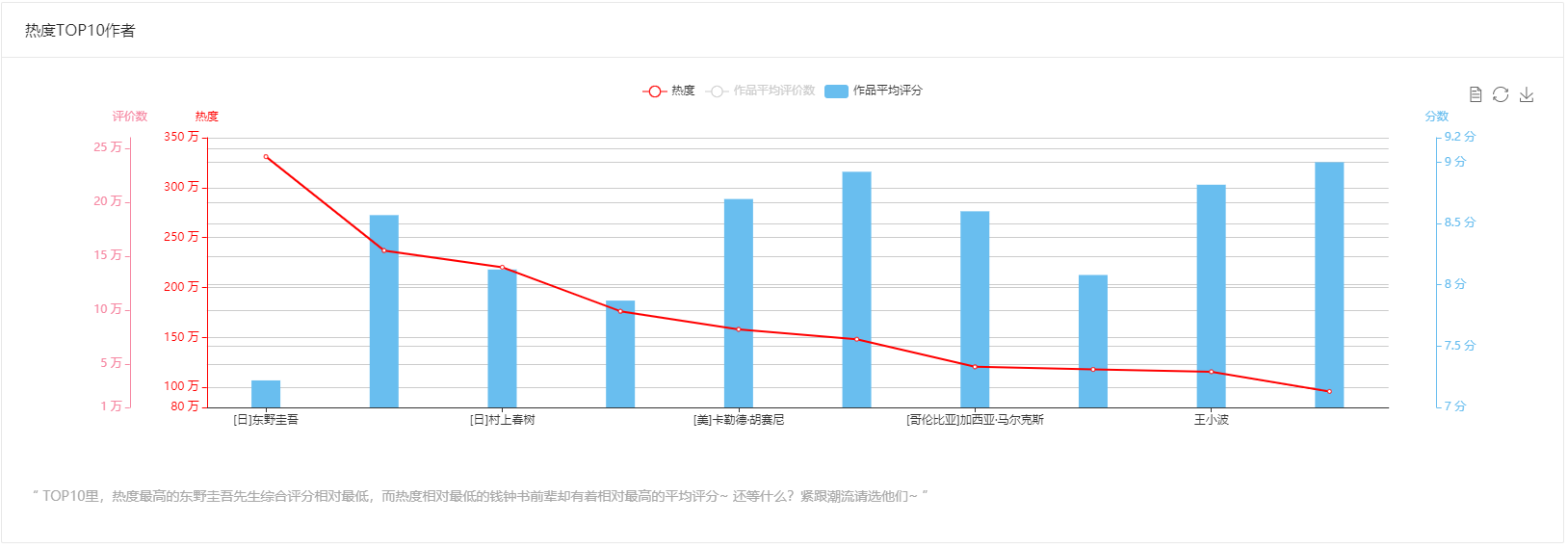
**eCharts代码：EchartsHotAuthor.vue**

1. **<template>**
2. **<div** class="Echarts"**>**
3. **<div** id="hotAuthor" style="width:100%; height:400px;"**></div>**
4. **</div>**
5. **</template>**
7. **<script>**
8. import axios from "axios";
9. import qs from "qs";
11. export default {
12. name: "EchartsHotAuthor",
13. data(){
14. return{
15. analysis\_dataList: [],
16. analysis\_authorList: [],
17. }
18. },
19. methods: {
20. myEcharts() {
21. // 基于准备好的dom，初始化echarts实例
22. var myChart = this.$echarts.init(document.getElementById('hotAuthor'));
24. var colors = ['#ff0000', '#f689a3', '#69beef'];
25. var option = {
26. color: colors,
28. tooltip: {
29. trigger: 'axis',
30. axisPointer: {
31. type: 'cross'
32. }
33. },
34. grid: {
35. right: '10%',
36. left: '12%'
37. },
38. toolbox: {
39. feature: {
40. dataView: {show: true, readOnly: false},
41. restore: {show: true},
42. saveAsImage: {show: true}
43. }
44. },
45. legend: {
46. data: ['热度', '作品平均评价数', '作品平均评分']
47. },
48. xAxis: [
49. {
50. type: 'category',
51. axisTick: {
52. alignWithLabel: true
53. },
54. data: this.analysis\_authorList
55. }
56. ],
57. yAxis: [
58. {
59. type: 'value',
60. name: '热度',
61. min: 80,
62. max: 350,
63. position: 'left',
64. axisLine: {
65. lineStyle: {
66. color: colors[0]
67. }
68. },
69. axisLabel: {
70. formatter: '{value} 万'
71. }
72. },
73. {
74. type: 'value',
75. name: '评价数',
76. min: 1,
77. max: 26,
78. position: 'left',
79. offset: 80,
80. axisLine: {
81. lineStyle: {
82. color: colors[1]
83. }
84. },
85. axisLabel: {
86. formatter: '{value} 万'
87. }
88. },
89. {
90. type: 'value',
91. name: '分数',
92. min: 7,
93. max: 9.2,
94. position: 'right',
95. offset: 50,
96. axisLine: {
97. lineStyle: {
98. color: colors[2]
99. }
100. },
101. axisLabel: {
102. formatter: '{value} 分'
103. }
104. }
105. ],
106. series: this.analysis\_dataList
107. // [
108. //     {
109. //         name: '蒸发量',
110. //         type: 'bar',
111. //         data: [2.0, 4.9, 7.0, 23.2, 25.6, 76.7, 135.6, 162.2, 32.6, 20.0, 6.4, 3.3]
112. //     },
113. //     {
114. //         name: '降水量',
115. //         type: 'bar',
116. //         yAxisIndex: 1,
117. //         data: [2.6, 5.9, 9.0, 26.4, 28.7, 70.7, 175.6, 182.2, 48.7, 18.8, 6.0, 2.3]
118. //     },
119. //     {
120. //         name: '平均温度',
121. //         type: 'line',
122. //         yAxisIndex: 2,
123. //         data: [2.0, 2.2, 3.3, 4.5, 6.3, 10.2, 20.3, 23.4, 23.0, 16.5, 12.0, 6.2]
124. //     }
125. // ]
126. };
127. myChart.setOption(option);
128. }
129. },
130. mounted() {
131. let result\_hot = [];
132. let result\_votes = [];
133. let result\_score = [];
134. let result\_author = [];
135. axios.post("http://localhost:8080/doubanBook/analysis", qs.stringify({
136. type: 'hotAuthor'
137. }, {headers:{'Content-Type':'application/x-www-form-urlencoded'}}))
138. .then(response =**>** {
139. const resultDataList = response.data.data;
140. console.log(333);
141. console.log(resultDataList)
143. for(let index in resultDataList){
144. let resultData = resultDataList[index];
145. result\_hotresult\_hot = result\_hot.concat(resultData.hot / 10000);
146. result\_votesresult\_votes = result\_votes.concat(resultData.votes / 10000);
147. result\_scoreresult\_score = result\_score.concat(resultData.score);
149. result\_authorresult\_author = result\_author.concat(resultData.author);
150. }
151. this.analysis\_dataList = [
152. {
153. name: '热度',
154. type: 'line',
155. data: result\_hot
156. },
157. {
158. name: '作品平均评价数',
159. type: 'line',
160. yAxisIndex: 1,
161. data: result\_votes
162. },
163. {
164. name: '作品平均评分',
165. type: 'bar',
166. barWidth: 30,
167. yAxisIndex: 2,
168. data: result\_score
169. }
170. ];
171. this.analysis\_authorList = result\_author;
172. console.log(this.analysis\_authorList);
173. console.log(this.analysis\_dataList);
174. console.log(333);
175. this.myEcharts();
176. }).catch(error =**>** {
177. console.log(error)
178. })
180. }
181. }
183. **</script>**
185. **<style** scoped**>**
187. **</style>**

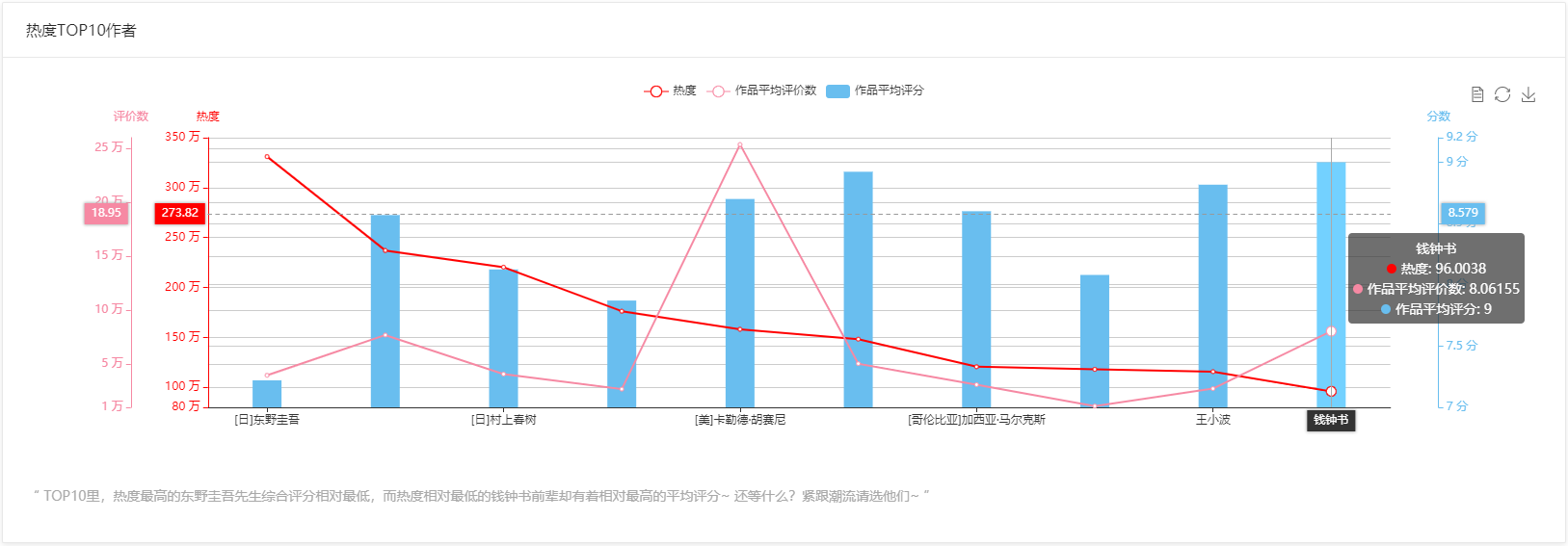
**本模块的亮点之一 经过精心调参将数据展示效果做到最优，并且数据也十分实用，结论非常直观，结果如图：**



**该分析给出了当下豆瓣读书中热度最高的10个作者，这个热度是通过加上作家所有作品中“在读”、“已读”和“读过”数据得出来的，同样的，通过点击数据描述，可以隐藏数据展示或者恢复展示，点击作品平均评价数，结果如图：**



**本分析结果显示了东野圭吾尽管平均评分不是特别高，但却是热度最高的作家，有着高达330w的“读”者，而作为第十位的钱钟书，虽然仅排在第十位，但却有着高达9.0的最高豆瓣平均评分：**

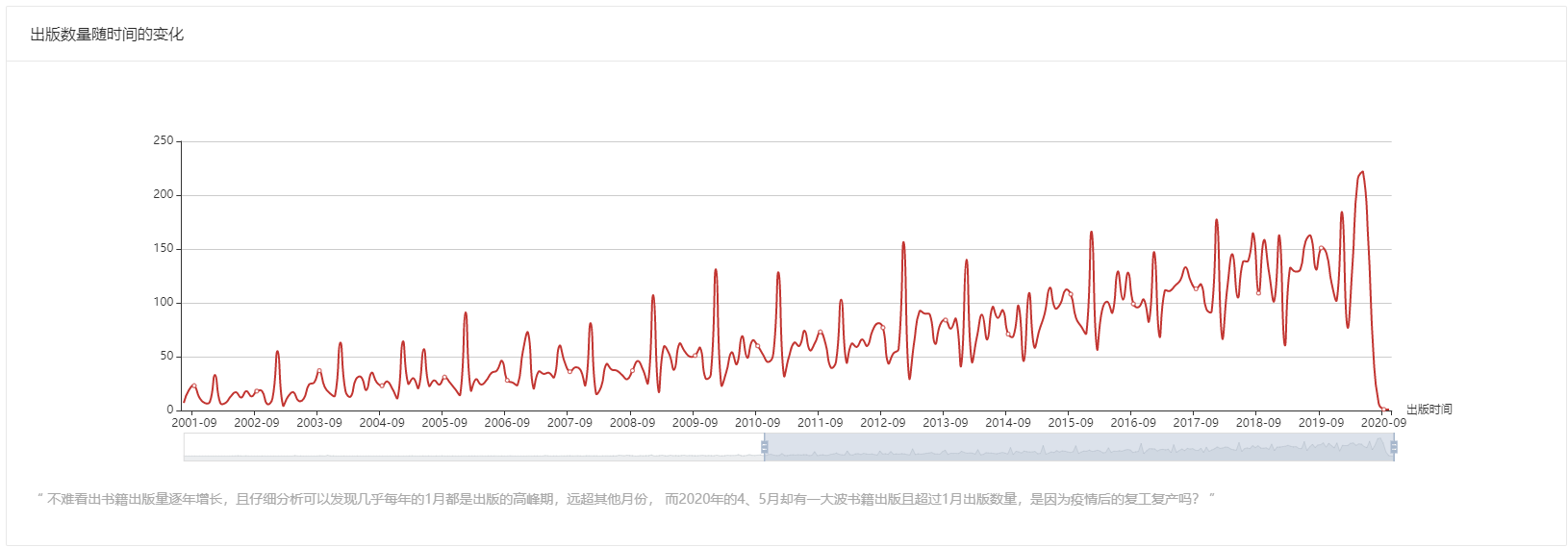


**不管怎样，该分析筛选出来的这10位作家的受欢迎程度和豆瓣评分都不低，所以如果遇到书荒，别犹豫，请选购他们的书籍就好了~**

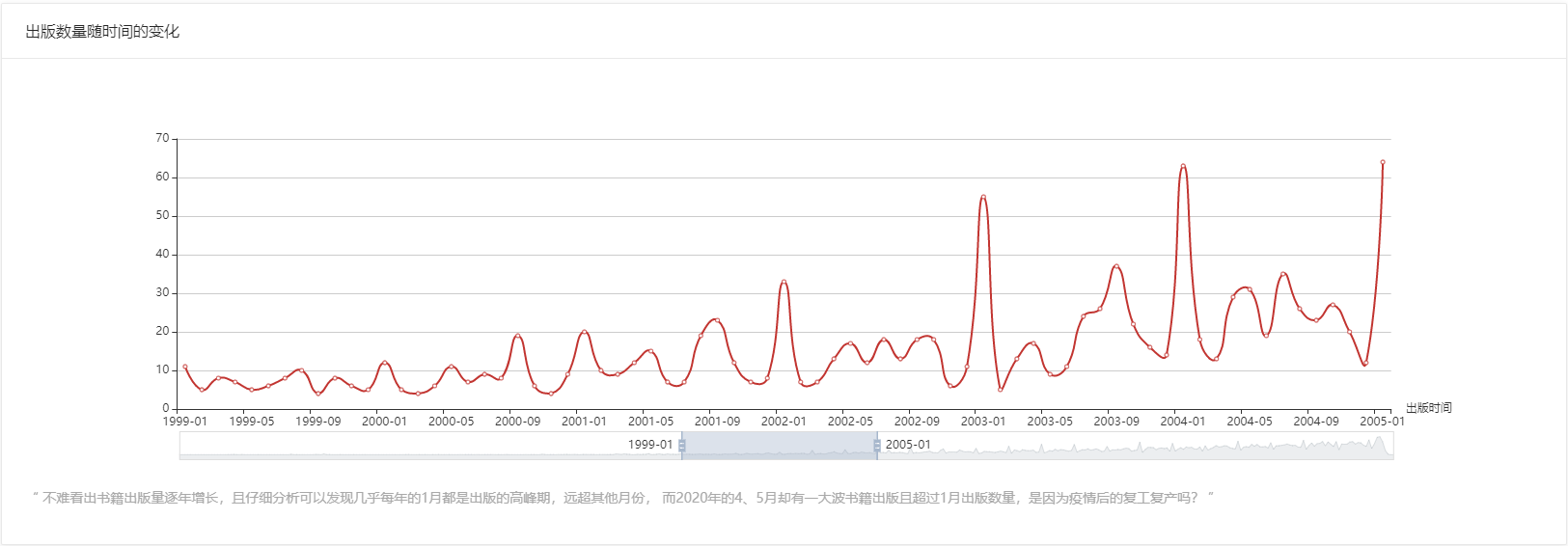
**eCharts代码：EchartsPubDate.vue**

1. **<template>**
2. **<div** class="Echarts"**>**
3. **<div** id="pubDate" style="width:100%;height:400px;"**></div>**
4. **</div>**
5. **</template>**
7. **<script>**
8. import axios from "axios";
9. import qs from "qs";
11. export default {
12. name: "EchartsPubDate",
13. data(){
14. return{
15. analysisData\_dateList:[],
16. analysisData\_countList:[]
17. }
18. },
19. methods:{
20. myEcharts(){
21. // 基于准备好的dom，初始化echarts实例
22. var myChart = this.$echarts.init(document.getElementById('pubDate'));
24. var option = {
26. dataZoom: [
27. {
28. realtime: true,
29. start: 48,
30. end: 100,
31. zoomOnMouseWheel: true,
32. xAxisIndex: 0
33. },
34. {
35. type: 'inside',
36. show: true,
37. start: 0,
38. end: 100,
39. left: "center",
40. // zoomOnMouseWheel:false,
41. yAxisIndex: 0
42. }
43. ],
44. tooltip: {
45. trigger: 'axis',
46. axisPointer: {
47. type: 'shadow'
48. }
49. },
50. xAxis: {
51. type: 'category',
52. name: '出版时间',
53. data: this.analysisData\_dateList
54. },
55. yAxis: {
56. type: 'value'
57. },
58. series: [{
59. radius: '55%',
60. center: ['40%', '50%'],
61. type: 'line',
62. smooth: true,
63. data: this.analysisData\_countList,
64. emphasis: {
65. itemStyle: {
66. shadowBlur: 10,
67. shadowOffsetX: 0,
68. shadowColor: 'rgba(0, 0, 0, 0.5)'
69. }
70. }
71. }]
72. };
74. // 使用刚指定的配置项和数据显示图表。
75. myChart.setOption(option);
76. }
77. },
78. mounted() {
79. let result\_dateList = []; // 存储日期列表
80. let result\_dateCount = []; // 存储数据列表
81. axios.post("http://localhost:8080/doubanBook/analysis", qs.stringify({
82. type: 'pubDate'
83. }, {headers:{'Content-Type':'application/x-www-form-urlencoded'}}))
84. .then(response =**>** {
85. const resultDataList = response.data.data
86. console.log(444);
87. console.log(resultDataList)
88. for(let index in resultDataList) {
89. let resultData = resultDataList[index];
90. result\_dateListresult\_dateList = result\_dateList.concat(resultData.analysisData);
91. result\_dateCountresult\_dateCount = result\_dateCount.concat(resultData.count);
92. }
93. this.analysisData\_dateList = result\_dateList;
94. this.analysisData\_countList = result\_dateCount;
95. console.log(this.analysisData\_dateList);
96. console.log(this.analysisData\_countList);
97. console.log(444);
98. this.myEcharts();
99. }).catch(error =**>** {
100. console.log(error)
101. })
103. }
104. }
105. **</script>**
107. **<style** scoped**>**
109. **</style>**

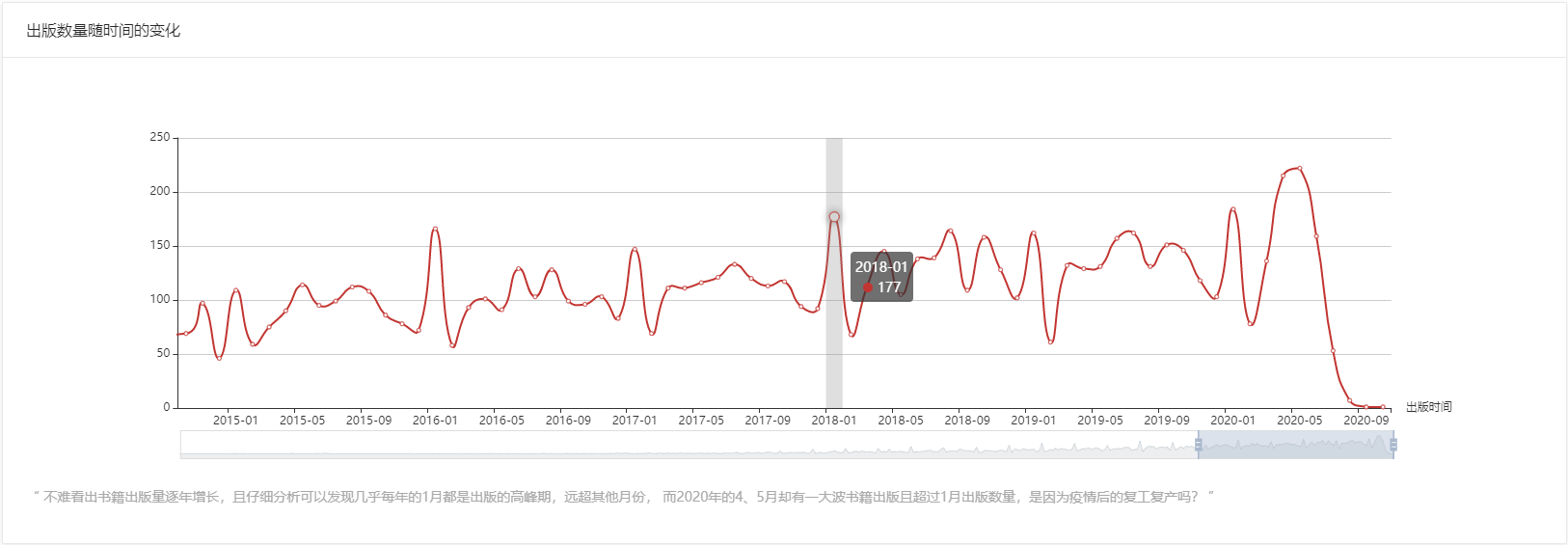
**该部分分析了本项目豆瓣数据集中最早的出版时间到最晚的出版时间中每个月出版的书籍数量变化，日期为从1990年1月，到2020年9月，结果如图：**

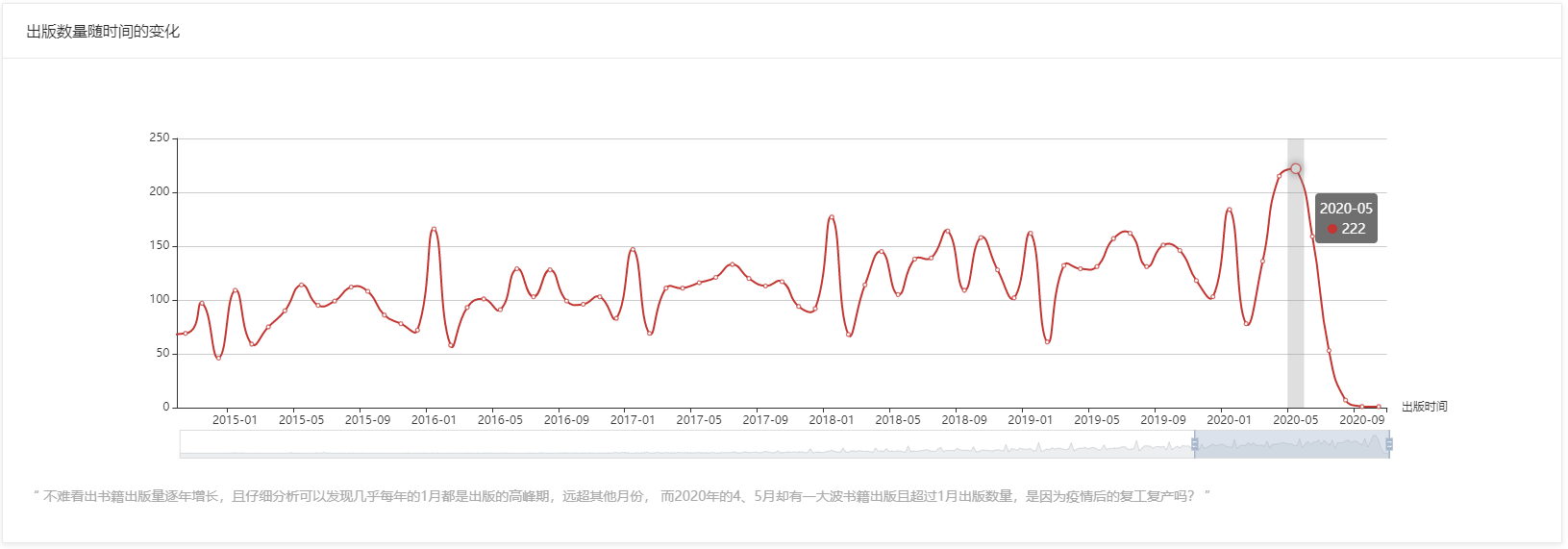


**由于数据特性，该图标的x轴十分长，于是通过使用组件和调参，加入了调整x轴区间以及移动区间和缩放区间的功能，使得图标可以动态地展示任意区间内的数据，且y轴数据会随着当前区间的最大最小值而自动缩放，如将区间设置为1990年1月至2005年1月，结果如图：**



**本分析整个出版数量随时间的变化是递增的，而在每年之内，1月是出版高峰期，而在2020年由于新冠疫情出现了例外——4、5月的书籍因复工复产出版量超过了1月：**

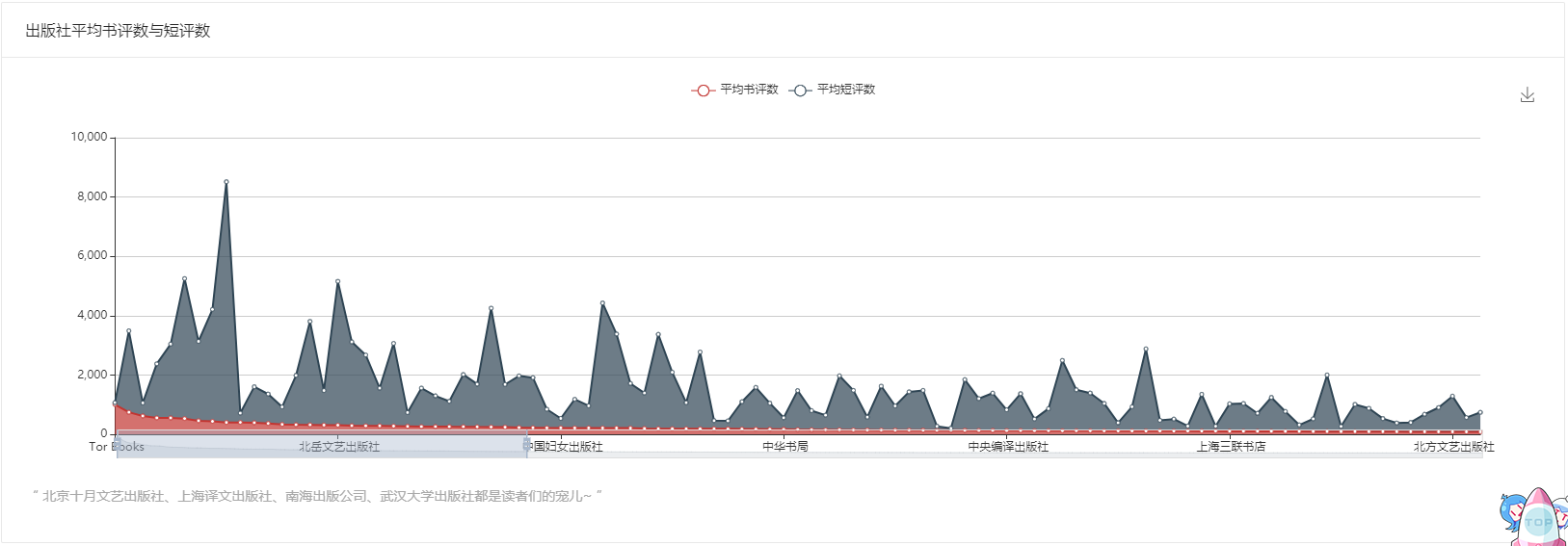




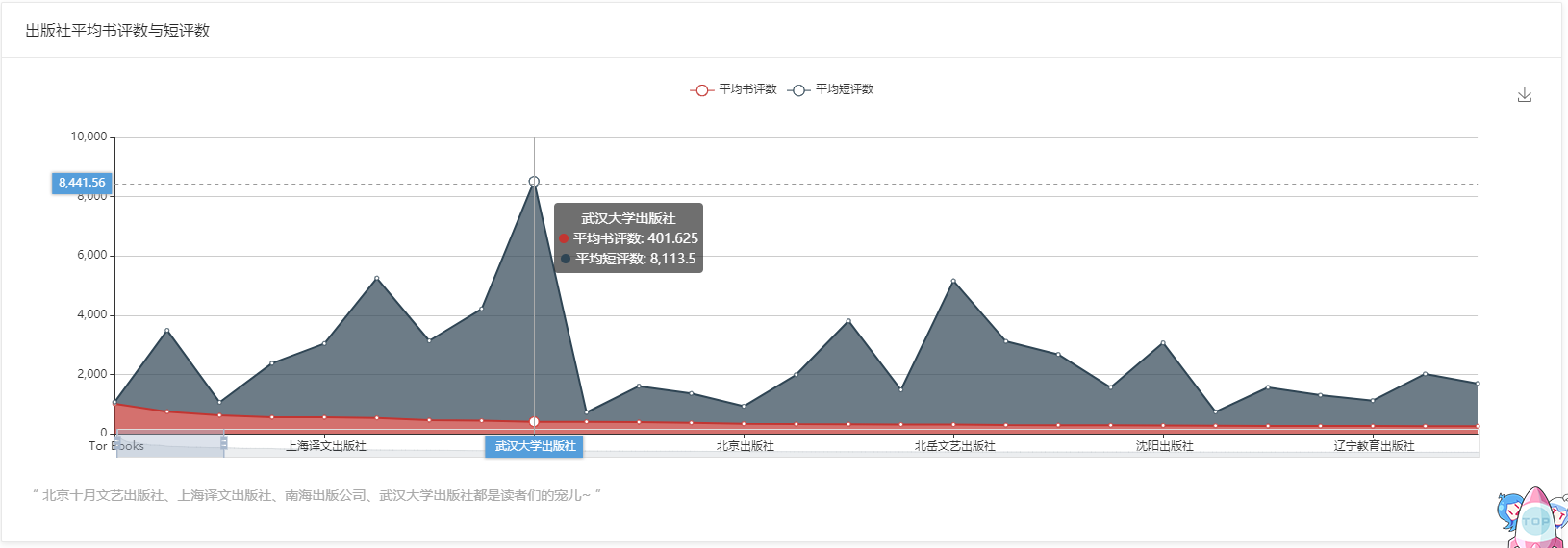
**eCharts代码：EchartsReviewPress.vue**

1. **<template>**
2. **<div** class="Echarts"**>**
3. **<div** id="reviewPress" style="width:100%; height:400px;"**></div>**
4. **</div>**
5. **</template>**
7. **<script>**
8. import axios from "axios";
9. import qs from "qs";
11. export default {
12. name: "EchartsReviewPress",
13. data(){
14. return{
15. analysis\_pressList: [],
16. analysis\_reviews: [],
17. analysis\_shortComms: [],
18. }
19. },
20. methods: {
21. myEcharts() {
22. // 基于准备好的dom，初始化echarts实例
23. var myChart = this.$echarts.init(document.getElementById('reviewPress'));
25. var option = {
26. dataZoom: [
27. {
28. realtime: true,
29. start: 0,
30. end: 30,
31. zoomOnMouseWheel: true,
32. xAxisIndex: 0
33. },
34. {
35. type: 'inside',
36. show: true,
37. start: 0,
38. end: 100,
39. left: "center",
40. // zoomOnMouseWheel:false,
41. yAxisIndex: 0
42. }
43. ],
44. tooltip: {
45. trigger: 'axis',
46. axisPointer: {
47. type: 'cross',
48. label: {
49. backgroundColor: '#559edb'
50. }
51. }
52. },
53. legend: {
54. data: ['平均书评数', '平均短评数']
55. },
56. toolbox: {
57. feature: {
58. saveAsImage: {}
59. }
60. },
61. grid: {
62. left: '3%',
63. right: '4%',
64. bottom: '3%',
65. containLabel: true
66. },
67. xAxis: [
68. {
69. type: 'category',
70. boundaryGap: false,
71. data: this.analysis\_pressList
72. }
73. ],
74. yAxis: [
75. {
76. type: 'value'
77. }
78. ],
79. series: [
80. {
81. name: '平均书评数',
82. type: 'line',
83. stack: '总量',
84. areaStyle: {},
85. data: this.analysis\_reviews
86. },
87. {
88. name: '平均短评数',
89. type: 'line',
90. stack: '总量',
91. areaStyle: {},
92. data: this.analysis\_shortComms
93. },
95. ]
96. };
97. myChart.setOption(option);
98. }
99. },
100. mounted() {
101. let result\_pressList = [];
102. let result\_reviews = [];
103. let result\_shortComms = [];
104. axios.post("http://localhost:8080/doubanBook/analysis", qs.stringify({
105. type: 'reviewPress'
106. }, {headers:{'Content-Type':'application/x-www-form-urlencoded'}}))
107. .then(response =**>** {
108. const resultDataList = response.data.data;
109. console.log(555);
110. console.log(resultDataList)
111. for(let index in resultDataList){
112. let resultData = resultDataList[index];
114. result\_pressListresult\_pressList = result\_pressList.concat(resultData.press);
115. result\_reviewsresult\_reviews = result\_reviews.concat(resultData.reviews);
116. result\_shortCommsresult\_shortComms = result\_shortComms.concat(resultData.shortComms);
117. }
118. this.analysis\_pressList = result\_pressList;
119. this.analysis\_reviews = result\_reviews;
120. this.analysis\_shortComms = result\_shortComms;
121. console.log(this.analysis\_pressList);
122. console.log(this.analysis\_reviews);
123. console.log(this.analysis\_shortComms);
124. console.log(555);
125. this.myEcharts();
126. }).catch(error =**>** {
127. console.log(error)
128. })
130. }
131. }
133. **</script>**
135. **<style** scoped**>**
137. **</style>**

**该部分分析了在出版书籍大于5的所有出版社中，所有的平均书评数和短评数，结果如图：**



**同样的，尽管该数据没有时间特性，但依然有者数据量庞大的特点，所以也引入了坐标轴缩放的功能，使其可以更清晰地被查阅：**



**此外，本数据按照出版社的平均书评数进行降序排序，得到了北京十月文艺出版社、上海译文出版社、南海出版公司、武汉大学出版社是最受点评读者青睐的出版社的结论**

**5 系统测试**

爬虫：通过Navicat查看远端MariaDB，然后在scrapy爬虫代码中编写测错报告。

服务器端：使用了Postman软件来测试服务器端的数据传入与返回，检测是否能返回想要的结果，同时结合Idea自带的报错系统以及System.out.println()函数来输出关键数据来查看数据是否正常。

前端：页面部分直接通过浏览器显示页面来查看，而数据则通过console.log()函数来返回关键数据。