基于 springboot 与 layui 的数据展示设计与实现

杨国举 张桂花
四川大学锦城学院计算机与软件学院 四川 成都 611731

【摘 要】全靠人工去分析和处理大量数据是完全不可靠的，于是就设计了基于 springBoot 和 layui 结合的数据展示系统，前端页面采用了 Layui 框架，Echarts 框架 , 数据库使用到 MySQL, 后端开发使用到了 springBoot+myBatis 框架。本项目通过将有效的方法让大量数据简单化，直观化，使其更易于分析，可以从繁琐的数据里，一击命中目标，完美提高工作效率与分析正确率。

【关键词】springBoot; bigdata; 数据展示; layui

1 项目简要概述

对于 Layui 与 springBoot 相结合的业务开发来说，它相对于 Vue 与 springBoot 相结合的业务开发，更加简单易懂，适合初学者去入门尝试，去设计数据分析后的数据展示与实现。Layui 拥有一个现成的前端框架，相对于 vue 来说，layui 更像是一个基于许多功能于一身的工具集。Springboot 应用于后端开发，其实就是 Spring 开发技术的升级，引入了注解，简化了 Web 框架的配置和开发流程。Mytatis 是对象映射框架，内置 jdbc, 只关注 SQL 本身 [1]。对于此项目前端开发来说，前端是通过调用后端预留出的特定接口来进行数据的交互。前端主要文件存放路径是 resources 下的 static 和 templates 两个文件夹。static 存放静态文件，比如存放 css 文件，echarts.js，jquery.js 和图片文件等，templates 存放的是动态文件，基本就是页面相关的 html 文件。对其后端开发而言：为了简化操作，我们使用 mybatis 的增强版 mybatis-plus(MyBatis-Plus 是一个 MyBatis 的增强工具，在 MyBatis 的基础上做的增强，号称是”为简化开发、提高效率而生”[2]。) 来自动生成目标文件对应的映射文件，在 entity 层建议手写生成 sql 表的实体类，按照表的字段书写实体类的成员变量。在 mapper 层里面创建一个接口用于继承 BaseMapper 然后泛型就是 entity 的类名 public interface ProductMapper extends BaseMapper <product>，还需要一个 vo 层，因为实体类的数据类型与前端要求传入的数据类型存在差异，所以就要求根据实体类来封装一套跟前端需求数据相吻合的结构来实现数据传输。再装配 service 层，封装业务返回固定的数据格式，最后通过 controller 层接收 service 层实体类返回的数据，并且将返回的数据通过特定保留的接口传递给前端，实现接收数据。

2 项目技术简单分析

2.1 项目整体流程示意图

产生的大量数据通过 flume 与 kafka 连接充当生产者，被读取到的数据存储在 kafka 的 topic 里面，然后通过 kafka 与 sparkStreaming（Spark Streaming 是一种 spark 提供的，对于大数据，进行实时计算的一种框架 [3]。）连接充当消费者，从 kafka 的 topic 里面消费数据，将得到的流式数据通过 sparkStreaming 进行特定需要的处理。然后将处理好的数据依赖 mysql 连接池并且插入数据到数据库，接着再通过 springBoot 与 myBatis 框架读取数据库数据，将获取的数据格式化，然后传入前端实现数据交互，最后通过 layui 框架和 echarts 框架实现数据可视。项目流程图如图 1 所示：

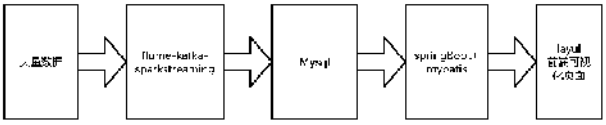

图 1 项目流程图

Figure 1 Project flow chart

2.2 后端模块解析

2.2.1 业务准备

要在 springBoot 应用里面完成真实的数据对接，就需要 springBoot 开放一个特定接口，将在 mysql 读取的数据真实的返回到开放接口，便可以完成数据输出，然后前端框架通过开放接口加载数据，实现数据对接，完成数据可视化。主要工作就是要完成数据的对接，首先应该对数据库里面需要展示的数据做一个映射，需要加一个 @ Data 的注解在 entity 层的新建立的 product 类上，再按照数据库表的字段去写类的成员变量，成员变量必须和数据库表成员字段一样才可以实现映射。

再在 mapper 层里面创建一个接口 ProductMapper 继承 BaseMapper<product>，泛型是 entity 里面 product 类型。接下来在 application.yml 文件写入数据库连接配置，再在启动类里面添加一个 MapperScan，就可以将接口的实例化对象扫入 IOC 里面，于是可以成功地调用查出数据库的数据。

接下来就是业务层的编写，因为前后端的对接，必须要有对应的接口并且需要数据规格相同。所以就要求按照前端数据传入要求，将 mysql 查出的数据封装给前端。因为前端要求数据是一个特殊格式的 json 数据，它不是直接就是一个数据的数组，它是由四个属性组成：code，msg，count 和 data，code 表示状态，Count 表示数据在数据库的条数，data 表示的是核心数据。为了完成数据封装，由于实体类的数据类型与前端要求的数据类型不一致，就需要根据实体类封装一套和前端所吻合的数据结构，就必须在 vo 层并且创建一个 DateVo 类。DataVo 成员变量就要与 json 数据格式的四个属性一一对应 code，msg，count 和 data，重点就是对 data 的封装。

2.2.2 数据封装

为了完成 data 数据的封装，还需要一个 Service 层将 json 数据返回，利用 mapper 进行数据的封装。由于 entity 层下实体类 product 的成员变量就是我们 data 所需要的数据，所以就可以直接获取。在业务层写一个接口 ProductService，在接口里面添加一个返回数据的方法 public DateVo<product> findData ( ) ; 然后添加一个 ProductServiceImpl 实现类来实现这个接口，实现接口的方法就是返回特定规则 json 数据的方法。DateVo dateVo1 = new DateVo() ; dateVo1 .setCode(0) ; dateVo1. setMsg (""); 但是对于数据里的 count 属性就需要根据数据库数据的条数动态改变，因为在 mapper 里面创建一个接口 ProductMapper 继承 Basemapper 并且泛型就是 entity 的类名，所以就可以使用 private ProductMapper productMapper ; dateVo1.setCount (productMapper1 . selectCount (null)) 来直接查询数据条数；最后就是 data 的封装，第一步查询所有的 data 数据，第二步将查询的数据转化为 vo 类型，第三步就是将遍历数据进行赋值后返回数据 for (category category1 : productList1) { category cat = new category () ; BeanUtils.copyProperties (category1 , cat) ; productVoList1.add (cat) ;}。通过 service 层的封装，就可以得到与 DateVo 相对应的 json 数据，数据封装如图 2 所示：

图 2 数据封装图

Figure 2 Data package diagram

然后再在 Controller 控制层将封装好的数据返回到预留接口里面：@ RequestMapping ("/list1") @ResponseBody public DateVo list1 ( ) { return productService . findData1 ( ) ; } 控制层数据返回如图 3 所示，然后用前端 html 用 url：直接调用 /list1 就可以实现将 mysql 数据穿入前端框架。

```
@RequestMapping("/list1")
@ResponseBody
public DateVo list1(){return productService.findData1();}
```

图 3 数据返回代码

Figure 3 Data return codes

2.2.3 注意事项

但是在这过程中会出现很多问题，主要会出现跨域问题，解决方法就是可以配置一个专有类，让它实现 WebMvcConfigurer 接口，并且完成里面需要的跨域操作， @Override public void addCorsMappings (CorsRegistry registry) { registry.addMapping ("/**") . allowedOrigins (" * ") . allowCredentials (true) . allowedMethods (" GET " , " POST " , " DELETE " , " PATCH ") . maxAge (3600) ; } 跨域问题代码如图 4 所示：

```
//解决跨域问题
@Configuration
public class CrossConfig implements WebMvcConfigurer {

    @Override
    public void addCorsMappings(CorsRegistry registry) {
        registry.addMapping( pathPattern: "/**")
                .allowedOrigins("*")
                .allowCredentials(true)
                .allowedMethods("GET","POST","DELETE","PATCH")
                .maxAge(3600);
    }
}
```

图 4 跨域代码

Figure 4 Cross-domain code

并且还需要配置一个视图解析器： thymeleaf : prefix : classpath : / templates / suffix : .html，并且在控制层写一个映射代码 @ GetMapping (" /{ url } ") // 通过后台去完成映射前端视图 public String redirect (@ PathVariable (" url ") String url) { return url ;}。映射如图 5 所示：

```
@GetMapping("/{url}")  //通过后台映射前端视图
public String redirect(@PathVariable("url") String url) { return url; }
```

图 5 映射代码

Figure 5 Mapping code

2.3 前端模块解析

2.3.1 前端数据格式

通过使用 layui(layui（谐音：类 UI) 是一款采用自身模块规范编写的前端 UI 框架，遵循原生 HTML/CSS/JS 的书写与组织形式，门槛极低，拿来即用[4]。) 后台布局框架为项目的主框架，然后再在这个后台布局框架里面进行嵌套。首先，后台布局框架基本可以分为两个部分：左侧导航区域主要是功能选项，要求展示数据处理的维度，右边框里就是类容主体部分，用于数据规则下的数据展示，将规则下的数据以数据表格展示出来，并且还可以用到了 echarts 框架，让数据表格里面的数据以扇形图，柱状图等更加直观形式展示出来。

首先是前端的后台布局框架，我们可以在官方网站上面找到简单的代码示例，最为主要就是需要修改 layui 框架文件的引入路径，基本上所有的 layui 框架都是有这两部分：一是 <link rel= ” stylesheet ” href = “ xxx/xxx/layui.css ” >，二是 < script src = ” xxx/xxx/layui.js ” charset = ” utf-8 ” />。也就是说这是正常使用框架的基本条件，但是也会用到 JavaScript 代码书写一些函数表达式，就需要引入 JavaScript 函数库，也就是引入 jquery.min.js 文件。

通过点击左部份的功能栏就会出现对应的需要选择的数据维度页面，这一部分的主要操作就会用到 JavaScript 的一些函数表达式，所以就要引入 jQuery.js 包。通过点击操作 onclick = ” change (‘xxx’) ” （xxx 是前端 html 文件的名字），类容的主体区域 < iframe src = ” ” frameborder = ” 0” id = ” xx” />，src 的值就表示类容区域会展示上面类容，所以还需要一个给 src 赋值的操作，这里就需要一个函数式：function change(title) { let url = null ; switch (title) { case 'xxx' : url = "/xxx " break ; case 'xxx' : url = "/xxx " break ; case ' xxx ' : url = "/xxx " break ; case 'xxx’ : url = "/xxx " break ; } $ ('iframe') . attr (' src ' , title) ; }。函数代码如图 6 所示：

图 6 函数代码

Figure 6 Function code

主框架里面需要点击后展示的子框架，同样也用 layui 的一个数据表格做示范。因为 layui 的数据表格对数据接收的格式有要求，必须是特定的 json 格式：{ "code": 0 , "msg" : "" , "count" :10 , "data": [{…}] }，为了实现与后端数据交互，后端就给 layui 框架预留了一个专门用作提供特定格式数据的接口，只需要在 layui 框架代码里面引用后端提供的同一个接口就可以实现动态数据获取，将前端 table.render 里面的 url 用接口名字代替，比如 url ： ” /list” 。需求的格式如图 7 所示，图 7 展示一条访问类型最多的数据：

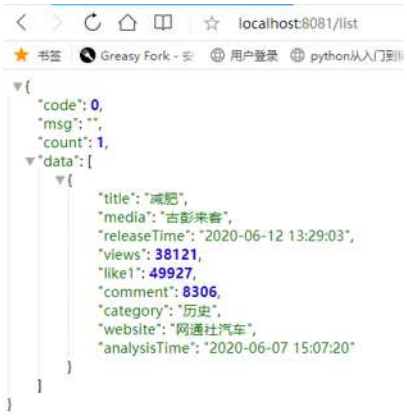

图 7 数据格式

Figure 7 Data format

与 echarts 框架不同的是，echarts 只需要两个值” name” 和” values” ，所以它的格式 { "name": ["z", "c","x"],"values":[2,1,4,]}，调用方式和上述 layui 相同。Echarts 需求数据如图 8 展示：

图 8 数据格式

Figure 8 Data format

2.3.2 前端页面展示

前端数据维度页面，通过选择数据维度为数据分析设立条件，前端设立了五个维度，第一个是通过任务编号的数据，第二个是用户名字，第三个是数据生成条数，第四个是数据的关键字，通过数据的关键来对需要分析的数据进行一个细致的筛选，第五个是数据所属的类型，

通过数据的类型选项，对大量数据小量化。

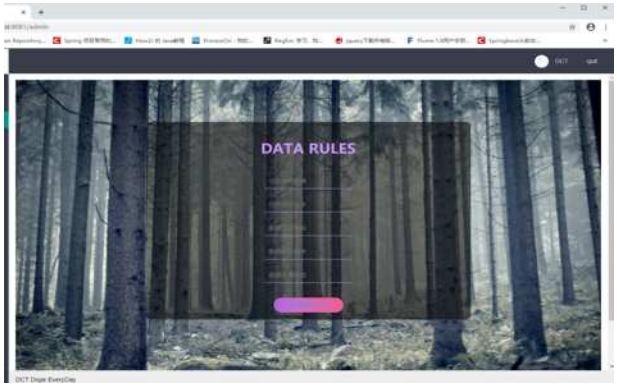

图 9 数据规则

Figure 9 Data format

Echarts 视图页面 (ECharts，一个使用 JavaScript 实现的开源可视化库，可以流畅地在 PC 和移动设备上运行，兼容当前绝大部分浏览器 [5])，通过对数据库类容做特定选择，查询特定要求的数据，以柱状图的方式展现出来：

@Select(“ select menu_name,menu_level from menu “),

图 10 柱状图

Figure 10 Histogram

Layui 数据表格展示页面：

图 11 数据表格图

Figure 11 Data table diagram

3 结束语

本项目的意义就在于利用前端现成框架 layui 并且集合后端 springBoot 与 myBatis 框架将大量经过处理后的数据以一个特殊的形式展示在用户的面前，大大地提升了我们对数据重要信息的获取速度。因为数据是前后端交互的，这样更方便数据分析者去通过数据剖析出更有价值的事实，相比其他前端框架来说，layui 就相当于是一个工具集，它的操作简单但易懂。

【参考文献】

[1] 陈倩怡 , 何军 .Vue+Springboot+MyBatis 技术应用解析 [J]. 电脑编程技巧与维护 ,2020(01):14–15+28.

[2] 王福强 .SpringBoot 揭秘 [M]. 北京 : 机械工业出版社 ,2016.

[3] 张友生 . 系统架构设计师考试全程指导 [M]. 北京 : 清华大学出版社 ,2014.

[4] 陈德春 .HTML5+CSS+JavaScript 深入学习实录 [M]. 北京 : 电子工业出版社 ,2013.