广东药科大学

医药信息工程学院

数字媒体技术系

实验报告

课程名称 前端设计与开发实验

使用学期2023-2024学年第 一 学期

班级 21数媒（1）班

学号 2120505110 2120505145

姓名 冯培豪 吴佶优

日期 2023.12.08

## 目录

### 一、需求分析

**1. 功能需求**

## 图书列表页面：展示系统中所有可供借阅的图书，包括图书的基本信息和封面图片等，同时加入教材接口和视听教程。

## 图书分类管理：允许用户根据分类或标签浏览图书，以便更快地找到感兴趣的图书。

## 图书详细页面：显示特定图书的详细信息，包括作者、出版社、简介、评分等。

## 搜索功能：提供关键字搜索功能，允许用户根据图书名称、作者、关键词等进行搜索，以找到所需图书。

## 购物车管理：显示用户已添加到购物车的图书清单，允许用户修改和结算支付等。

**2. 模块设计/项目划分**

## （1）图书管理模块：

## 图书列表功能：展示系统中所有可供借阅的图书，包括图书的基本信息和封面图片等。

## 图书分类管理功能：允许用户根据分类或标签浏览图书，以便更快地找到感兴趣的图书。

## 图书详细页面功能：显示特定图书的详细信息，包括作者、出版社、简介、评分等。

## 搜索功能：提供关键字搜索功能，允许用户根据图书名称、作者、关键词等进行搜索，以找到所需图书。

## （2）教材接口模块：

## 教材接口功能：整合教材供应商接口，提供教材订购、查询和管理等功能。

## 教材列表功能：展示系统中的教材资源，包括教材的基本信息和封面图片等。

## 教材详细页面功能：显示特定教材的详细信息，包括作者、出版社、简介、评分等。

## （3）视听教程模块：

## 视听教程功能：整合视听教程资源，提供在线观看、搜索和管理等功能。

## 视听教程列表功能：展示系统中的视听教程资源，包括基本信息和封面图片等。

## 视听教程详细页面功能：显示特定视听教程的详细信息，包括作者、出版社、简介、评分等。

## （4）购物车管理模块：

## 购物车功能：显示用户已添加到购物车的图书和教材清单，允许用户修改和结算支付等。

## 购物车结算功能：计算购物车中图书和教材的总金额，提供支付接口以完成结算过程。

### 二、系统概要设计

**1. 系统功能设计**

## (1)图书分类管理功能：

## 分类列表：展示系统中的图书分类或标签列表，用户可以根据分类或标签进行图书的筛选和浏览。

## 分类详情：点击特定分类或标签后，展示该分类下的图书列表。

## (2)图书详细页面功能：

## 图书详情：显示特定图书的详细信息，包括作者、出版社、简介、评分等。

## 封面图片：展示图书的封面图片。

## 添加到购物车：用户可以将感兴趣的图书添加到购物车中。

## (3)搜索功能：

## 关键字搜索：用户可以根据图书名称、作者、关键词等进行搜索，以找到所需图书。

## (4)购物车管理功能：

## 购物车列表：显示用户已添加到购物车的图书清单，包括图书的基本信息、数量和总金额。

## 修改购物车：用户可以修改购物车中图书的数量或删除不需要的图书。

## 结算支付：用户可以选择购物车中的图书，并进行支付操作。

**2. 功能页面设计**

## 图书列表页面：展示系统中所有可供借阅的图书，包括图书的基本信息和封面图片等。每本图书可以包含一个链接，点击后可以跳转到图书详细页面。

## 图书分类页面：展示图书分类或标签的列表，用户可以根据分类或标签进行图书的筛选和浏览。

## 图书详细页面：显示特定图书的详细信息，包括作者、出版社、简介、评分等。页面上可以显示图书的封面图片，并提供添加到购物车的操作。

## 搜索页面：提供关键字搜索功能，用户可以输入图书名称、作者、关键词等进行搜索，搜索结果将展示在页面上。

## 购物车页面：显示用户已添加到购物车的图书清单，包括图书的基本信息、数量和总金额。用户可以修改购物车中图书的数量或删除不需要的图书，并进行支付操作。

### 三、系统详细设计与实现

**1. 首页**

## 设计一个用户友好的首页，展示系统的主要功能和特色图书、教材等推荐内容。

## 包含导航栏或菜单，方便用户导航到其他页面。

## 可以包括搜索框，让用户快速搜索图书或教材

**2. 商品管理**

## 提供管理员对商品的管理功能，包括新增商品、编辑商品、删除商品等操作。

## 设计一个商品列表页面，展示系统中所有的商品信息，包括图书和教材。

## 在商品列表页面，管理员可以进行筛选、排序等操作，以便更好地管理商品。

**3. 购物车和结算**

## 设计一个购物车页面，显示用户已添加到购物车的图书清单，包括图书的基本信息、数量和总金额。

## 用户可以在购物车页面修改图书的数量或删除不需要的图书。

## 提供结算和支付功能，让用户能够选择购物车中的图书，并进行支付操作。

**4. 用户管理**

## 提供管理员对用户的管理功能，包括新增用户、编辑用户、删除用户等操作。

## 设计一个用户列表页面，展示系统中所有的用户信息，包括用户名、联系方式等。

## 在用户列表页面，管理员可以进行搜索、筛选等操作，以便更好地管理用户。

**5. 路由配置及其他组件**

## 配置系统的路由，定义各个页面的路径和对应的组件。

## 设计通用的组件，如导航栏、页脚等，以提高系统的可复用性和一致性。

**6. 调试与发布**

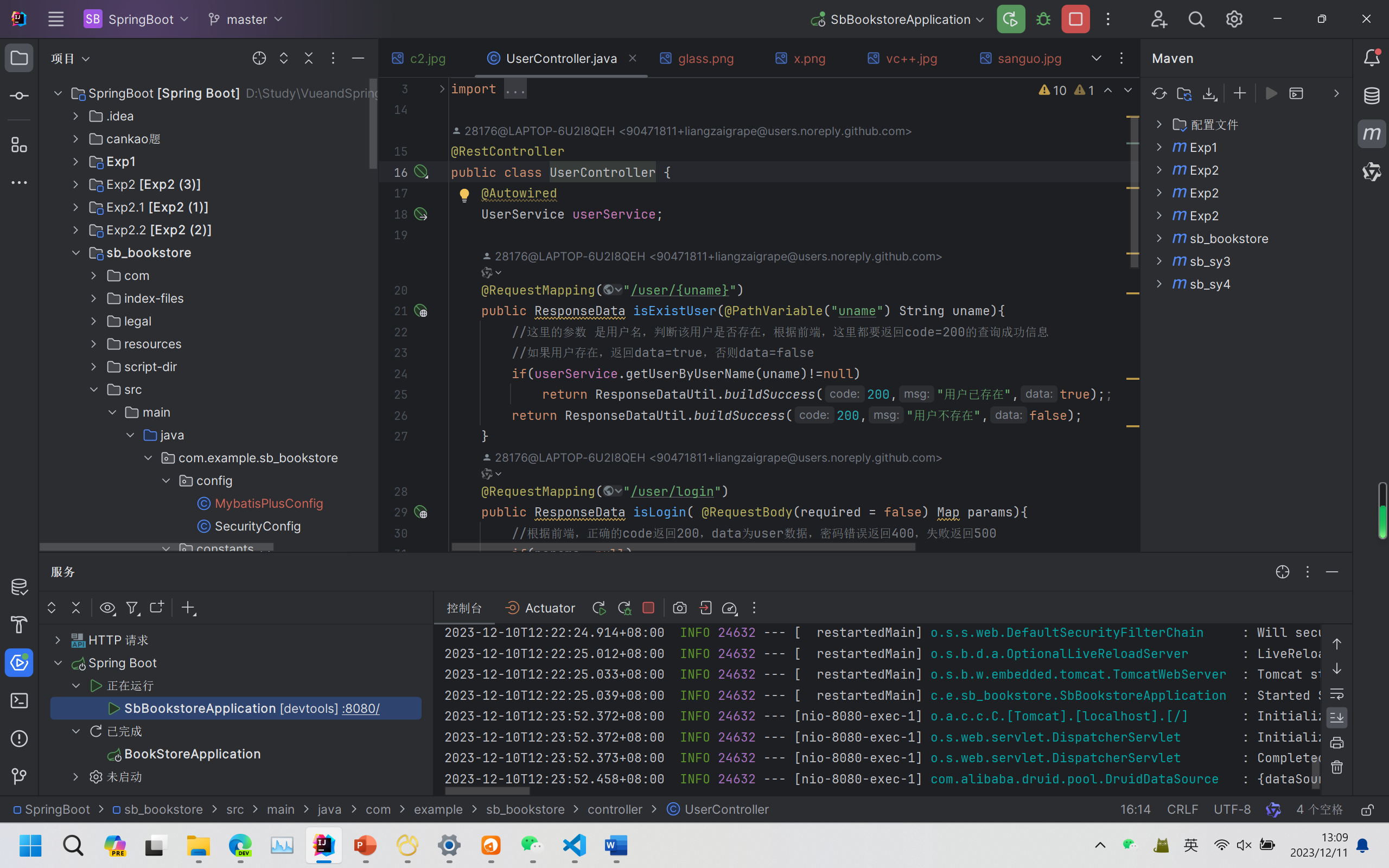
## 进行系统的调试和测试，确保系统的各项功能正常运行。

## 部署系统到服务器或云平台，以便用户可以通过网络访问系统。

## 配置系统的权限和安全性，确保用户数据和系统信息的安全性

### 四、测试

测试结果



### 五、附录

1. 项目实施效果

图形用户界面, 应用程序

描述已自动生成

2. 项目全部代码

package com.example.sb\_bookstore.config;  
  
import com.baomidou.mybatisplus.annotation.DbType;  
import com.baomidou.mybatisplus.extension.plugins.MybatisPlusInterceptor;  
import com.baomidou.mybatisplus.extension.plugins.inner.PaginationInnerInterceptor;  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
  
@Configuration  
public class MybatisPlusConfig {  
  
 @Bean  
 public MybatisPlusInterceptor mybatisPlusInterceptor() {  
 MybatisPlusInterceptor interceptor = new MybatisPlusInterceptor();  
 PaginationInnerInterceptor paginationInnerInterceptor = new PaginationInnerInterceptor(DbType.*MYSQL*);  
 // 设置请求的页面大于最大页后操作， true调回到首页，false 继续请求 默认false  
 paginationInnerInterceptor.setOverflow(false);  
 // 设置最大单页限制数量，默认 500 条，-1 不受限制  
 paginationInnerInterceptor.setMaxLimit(100L);  
 interceptor.addInnerInterceptor(paginationInnerInterceptor);  
 return interceptor;  
 }  
  
}

package com.example.sb\_bookstore.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.Customizer;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.web.SecurityFilterChain;  
  
  
@Configuration  
@EnableWebSecurity  
public class SecurityConfig {  
  
 @Bean  
  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http  
  
 .csrf().disable()  
 //关闭spring security的csrf功能  
 .authorizeHttpRequests(authorize -> authorize  
 //放行/test的访问，不用验证  
 .requestMatchers("/test").permitAll()  
 .requestMatchers("/register").permitAll()  
 // “/\*\*”放行/ 根路径以及其子路径，相当于整个系统不用验证了  
 //可以使用“/\*” 表示只放行根目录下页面和请求  
 .requestMatchers("/\*\*").permitAll()  
 //除了以上规则，默认拦截所有URL请求，要求认证后才能访问（该语句必须要放在末尾）  
 .anyRequest().authenticated()  
 )  
 .formLogin(formLogin -> formLogin  
 /// 定义登陆页面，不用验证  
 .loginPage("/login")  
 .permitAll()  
 )  
 //开始记住我,保持登录状态  
 .rememberMe(Customizer.*withDefaults*());  
  
 return http.build();  
 }  
}

package com.example.sb\_bookstore.config;  
  
import org.springframework.context.annotation.Bean;  
import org.springframework.context.annotation.Configuration;  
import org.springframework.security.config.Customizer;  
import org.springframework.security.config.annotation.web.builders.HttpSecurity;  
import org.springframework.security.config.annotation.web.configuration.EnableWebSecurity;  
import org.springframework.security.web.SecurityFilterChain;  
  
  
@Configuration  
@EnableWebSecurity  
public class SecurityConfig {  
  
 @Bean  
  
 public SecurityFilterChain filterChain(HttpSecurity http) throws Exception {  
 http  
  
 .csrf().disable()  
 //关闭spring security的csrf功能  
 .authorizeHttpRequests(authorize -> authorize  
 //放行/test的访问，不用验证  
 .requestMatchers("/test").permitAll()  
 .requestMatchers("/register").permitAll()  
 // “/\*\*”放行/ 根路径以及其子路径，相当于整个系统不用验证了  
 //可以使用“/\*” 表示只放行根目录下页面和请求  
 .requestMatchers("/\*\*").permitAll()  
 //除了以上规则，默认拦截所有URL请求，要求认证后才能访问（该语句必须要放在末尾）  
 .anyRequest().authenticated()  
 )  
 .formLogin(formLogin -> formLogin  
 /// 定义登陆页面，不用验证  
 .loginPage("/login")  
 .permitAll()  
 )  
 //开始记住我,保持登录状态  
 .rememberMe(Customizer.*withDefaults*());  
  
 return http.build();  
 }  
}

package com.example.sb\_bookstore.constants;  
  
import lombok.Getter;  
@Getter  
public enum ResultEnums {  
 *DEFAULT*(1,"默认消息"),  
 *SUCCESS*(200,"请求成功"),  
 *ERROR*(400,"请求失败");  
 private int code;//结果代码  
 private String msg;//结果信息  
  
 ResultEnums(int code, String msg){  
 this.code=code;  
 this.msg=msg;  
 }  
  
}

package com.example.sb\_bookstore.controller;  
  
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;  
import com.example.sb\_bookstore.domain.Book;  
  
import com.example.sb\_bookstore.domain.BookCategory;  
import com.example.sb\_bookstore.domain.Comment;  
import com.example.sb\_bookstore.response.ResponseData2;  
import com.example.sb\_bookstore.service.BookCategoryService;  
import com.example.sb\_bookstore.service.BookService;  
import com.example.sb\_bookstore.service.CommentService;  
import com.example.sb\_bookstore.utils.RandomUtil;  
import com.example.sb\_bookstore.utils.ResponseData2Util;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.PathVariable;  
import org.springframework.web.bind.annotation.RequestMapping;  
import org.springframework.web.bind.annotation.RequestParam;  
import org.springframework.web.bind.annotation.RestController;  
  
import java.util.List;  
  
@RestController  
public class RouterController {  
 @Autowired  
 BookService bookService;  
 @Autowired  
 BookCategoryService bookCategoryService;  
 @Autowired  
 CommentService commentService;  
 @RequestMapping("/books")  
 public List<Book> getAllBooks() {  
 return bookService.getAllBooks();  
 }  
  
 @RequestMapping("/book/new")  
 public List<Book> getNewBooks() {  
 return bookService.getNewBooks();  
 }  
  
 @RequestMapping("/book/hot")  
 public List<Book> getHotBooks() {  
 return bookService.getHotBooks();  
 }  
  
 @RequestMapping("/book/teach")  
 public List<Book> getTeachBooks() {  
 return bookService.getTeachBooks();  
 }  
  
  
 @RequestMapping("/book/{id}")  
 public Book getBookById(@PathVariable("id") Integer id) {  
 if (id != null)  
 return bookService.getBookById(id);  
 return null;  
 }  
// \*\*\*此处修改代码，增加代码，为List，返回书籍list  
 @RequestMapping("/book/category/{cid}")  
 public List<Book> getBooksByCategoryId(@PathVariable("cid") Integer cid){  
 return bookService.getBooksByCategoryId(cid);  
 }  
 // 分页查询  
 @RequestMapping("/book/category/{cid}/page")  
 public ResponseData2 getBooksByCategoryId(@PathVariable("cid") Integer cid,  
 @RequestParam(defaultValue = "1") Integer pageNum,  
 @RequestParam(defaultValue = "5") Integer pageSize  
  
 ){  
 Page<Book> books = bookService.selectPageByCategory(new Page<>(pageNum, pageSize), cid);  
 return ResponseData2Util.*buildSuccess*(books.getRecords(),books.getTotal());  
 }  
  
 //不分页查询  
 @RequestMapping("/search")  
 public List<Book> searchBooks(@RequestParam(value = "keyword",required = false) String keyword ){  
 //\*\* 修改 wd=> keyword  
 //因为前端用了keyword作为传递参数的命名，所以这里用keyword，保持对应关系,  
 // required=false表示可以有keyword，也可以没有keyword  
 if(keyword!=null &&!"".equals(keyword) )  
 {  
 return bookService.searchBooks(keyword);  
 }  
 //不加条件则空  
 return null;  
 }  
 //分页查询  
 @RequestMapping("/search/page")  
 //获得搜索结果的总数  
 public ResponseData2 countBySearch(@RequestParam(value = "keyword",required = false) String keyword,  
 @RequestParam(defaultValue = "1") Integer pageNum,  
 @RequestParam(defaultValue = "5") Integer pageSize  
 ){  
 // 默认分页大小为5，当前页为1  
 if(keyword!=null &&!"".equals(keyword) )  
 {  
 Page<Book> books = bookService.selectPageBySearch(new Page<>(pageNum, pageSize), keyword);  
 return ResponseData2Util.*buildSuccess*(books.getRecords(),books.getTotal());  
 }  
 //不加条件则空  
 return ResponseData2Util.*buildError*();  
 }  
  
 @RequestMapping("/categories")  
 public List<BookCategory> getAllCategories(){  
 //0 表示根，就是查出所有  
 return bookCategoryService.listCategories(0);  
 }  
  
 @RequestMapping("/category/{cid}")  
 public BookCategory getBookCategoryByCategoryId(@PathVariable("cid") Integer cid){  
 return bookCategoryService.getBookCategoryByCategoryId(cid);  
 }  
  
  
 @RequestMapping("/comments")  
 public List<Comment> getAllComments(){return commentService.getAllComments();}  
  
 @RequestMapping("/comment/{cid}")  
 public Comment getCommentByCommentId(@PathVariable("cid") Integer cid){  
 return commentService.getCommentByCommentId(cid);  
 }  
 @RequestMapping("/book/{bid}/comment")  
 public List<Comment> getCommentsByBookId(@PathVariable("bid") Integer bid){  
 return commentService.getCommentsByBookId(bid);  
 }  
 @RequestMapping("/user/{uid}/comment")  
 public List<Comment> getCommentsByUserId(@PathVariable("uid") Integer uid){  
  
 return commentService.getCommentsByUserId(uid);  
 }  
 @RequestMapping("/test")  
 public int test(){  
 return RandomUtil.*getRandom*();  
 }  
  
  
}

package com.example.sb\_bookstore.controller;  
  
import com.example.sb\_bookstore.domain.User;  
import com.example.sb\_bookstore.response.ResponseData;  
import com.example.sb\_bookstore.service.UserService;  
import com.example.sb\_bookstore.utils.RandomUtil;  
import com.example.sb\_bookstore.utils.ResponseDataUtil;  
import net.sf.jsqlparser.expression.DateTimeLiteralExpression;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.web.bind.annotation.\*;  
  
import java.util.Date;  
import java.util.Map;  
  
@RestController  
public class UserController {  
 @Autowired  
 UserService userService;  
  
 @RequestMapping("/user/{uname}")  
 public ResponseData isExistUser(@PathVariable("uname") String uname){  
 //这里的参数 是用户名，判断该用户是否存在，根据前端，这里都要返回code=200的查询成功信息  
 //如果用户存在，返回data=true，否则data=false  
 if(userService.getUserByUserName(uname)!=null)  
 return ResponseDataUtil.*buildSuccess*(200,"用户已存在",true);;  
 return ResponseDataUtil.*buildSuccess*(200,"用户不存在",false);  
 }  
 @RequestMapping("/user/login")  
 public ResponseData isLogin( @RequestBody(required = false) Map params){  
 //根据前端，正确的code返回200，data为user数据，密码错误返回400，失败返回500  
 if(params==null)  
 return ResponseDataUtil.*buildError*(400,"参数为空，请检查传递数据方式，不可直接用form");  
 String uname=params.get("username").toString();  
 String password=params.get("password").toString();  
 if(uname==null||"".equals(uname)||"".equals(password)||password==null)  
 return ResponseDataUtil.*buildError*(400,"请求失败");  
 User user=userService.getUserByUserNameAndPassword(uname,password);  
 if(user!=null){  
 return ResponseDataUtil.*buildSuccess*(200,"登录成功",user);  
 } else if (userService.getUserByUserName(uname)!=null) {  
 return ResponseDataUtil.*buildError*(400,"密码错误");  
 }  
 return ResponseDataUtil.*buildError*(500,"登录失败");  
 }  
 @RequestMapping("/user/register")  
 public ResponseData register(@RequestBody Map params){  
 //前端传递了3个值过来  
 if(params==null)  
 return ResponseDataUtil.*buildError*(400,"参数为空，请检查传递数据方式，不可直接用form");  
 String uname=params.get("username").toString();  
 String password=params.get("password").toString();  
 String mobile=params.get("mobile").toString();  
 if(uname==null||"".equals(uname)||"".equals(password)||password==null||mobile==null||"".equals(mobile))  
 return ResponseDataUtil.*buildError*(400,"用户名，密码和手机号码数据均不能为空");  
 if(userService.getUserByUserName(uname)!=null) return ResponseDataUtil.*buildError*(400,"用户名已经存在");  
 User user=new User(uname,"example@example.com",password, new Date(), RandomUtil.*getRandom*(),mobile,"user");  
  
 int id=userService.addUserReturnId(user);  
 //这个时候，正常的话，id会给回到user  
 if(id!=0)  
 return ResponseDataUtil.*buildSuccess*(200,"注册成功",user);  
 return ResponseDataUtil.*buildError*(500,"注册失败");  
 }  
}

package com.example.sb\_bookstore.domain;  
import com.baomidou.mybatisplus.annotation.IdType;  
import com.baomidou.mybatisplus.annotation.TableField;  
import com.baomidou.mybatisplus.annotation.TableId;  
import com.baomidou.mybatisplus.annotation.TableName;  
import com.fasterxml.jackson.annotation.JsonFormat;  
import lombok.Getter;  
import lombok.Setter;  
  
import java.util.Date;  
@Setter  
@Getter  
@TableName(value = "book")  
public class Book {  
 //处理Mysql自增的主键id  
 @TableId(value = "id", type = IdType.*AUTO*)  
 int id;  
 */\*\*  
 \*指定列名,若一致可以不用指定  
 \* 这里使用了驼峰写法，而数据库字段名为book\_id,  
 \* 也可以不用指定列名，可以关联的，默认驼峰和下划线可以自动识别  
 \*/* @TableField(value = "book\_id")  
 int bookId;  
 String title;  
 String author;  
 float price;  
 float discount;  
 String image;  
  
 String bookConcern;  
 @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss",locale="zh",timezone="GMT+8")  
 Date publishDate;  
 String brief;  
 int categoryId;  
//全参构造函数  
 public Book(int id, int bookId, String title, String author, float price, float discount, String image, String bookConcern, Date publishDate, String brief, int categoryId) {  
 this.id = id;  
 this.bookId = bookId;  
 this.title = title;  
 this.author = author;  
 this.price = price;  
 this.discount = discount;  
 this.image = image;  
 this.bookConcern = bookConcern;  
 this.publishDate = publishDate;  
 this.brief = brief;  
 this.categoryId = categoryId;  
 }  
 //无参构造函数  
 public Book(){}  
 //不带id的构造参数  
 public Book(int bookId, String title, String author, float price, float discount, String image, String bookConcern, Date publishDate, String brief, int categoryId) {  
 this.bookId = bookId;  
 this.title = title;  
 this.author = author;  
 this.price = price;  
 this.discount = discount;  
 this.image = image;  
 this.bookConcern = bookConcern;  
 this.publishDate = publishDate;  
 this.brief = brief;  
 this.categoryId = categoryId;  
 }  
  
}

package com.example.sb\_bookstore.domain;  
  
import com.baomidou.mybatisplus.annotation.IdType;  
import com.baomidou.mybatisplus.annotation.TableField;  
import com.baomidou.mybatisplus.annotation.TableId;  
import com.baomidou.mybatisplus.annotation.TableName;  
  
import lombok.Getter;  
import lombok.Setter;  
  
import java.util.List;  
  
@Getter  
@Setter  
@TableName("bookcategory")  
public class BookCategory {  
 //处理Mysql自增的主键id  
 @TableId(value = "id", type = IdType.*AUTO*)  
 int id;  
 @TableField(value = "category\_id")  
 int categoryId;  
  
 @TableField(value = "parent\_id")  
 int parentId;  
 @TableField(value = "category\_name")  
 String categoryName;  
 //这里用来存放子类别，不是表bookcategory的字段  
 @TableField(exist = false)  
 private List<BookCategory> children;  
  
 public BookCategory(int categoryId,String categoryName,int parentId){  
  
 this.parentId=parentId;  
 this.categoryId=categoryId;  
 this.categoryName=categoryName;  
 }  
 public BookCategory(int id,int categoryId,String categoryName,int parentId){  
 this.id=id;  
 this.parentId=parentId;  
 this.categoryId=categoryId;  
 this.categoryName=categoryName;  
 }  
}

package com.example.sb\_bookstore.domain;  
  
import com.baomidou.mybatisplus.annotation.IdType;  
import com.baomidou.mybatisplus.annotation.TableField;  
import com.baomidou.mybatisplus.annotation.TableId;  
import com.baomidou.mybatisplus.annotation.TableName;  
import com.fasterxml.jackson.annotation.JsonFormat;  
import lombok.AllArgsConstructor;  
import lombok.Getter;  
import lombok.Setter;  
  
  
import java.util.Date;  
@Getter  
@Setter  
@TableName("comment")  
@AllArgsConstructor  
public class Comment {  
 //处理Mysql自增的主键id  
 @TableId(value = "id", type = IdType.*AUTO*)  
 int id;  
 int commentId;  
 String content;  
 @TableField(value = "comment\_date")  
 @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss",locale="zh",timezone="GMT+8")  
 Date commentDate;  
 @TableField(value = "user\_id")  
 int userId;  
 @TableField(value = "book\_id")  
 int bookId;  
  
}

package com.example.sb\_bookstore.domain;  
  
import com.baomidou.mybatisplus.annotation.IdType;  
import com.baomidou.mybatisplus.annotation.TableField;  
import com.baomidou.mybatisplus.annotation.TableId;  
import com.baomidou.mybatisplus.annotation.TableName;  
import com.fasterxml.jackson.annotation.JsonFormat;  
import lombok.Getter;  
import lombok.Setter;  
  
import java.util.Date;  
  
@TableName("user")  
@Setter  
@Getter  
public class User {  
 String userName;  
 String email;  
 String password;  
 @TableField(value = "create\_time")  
 @JsonFormat(pattern="yyyy-MM-dd HH:mm:ss",locale="zh",timezone="GMT+8")  
 Date createTime;  
 int userId;  
 String mobile;  
 //处理Mysql自增的主键id  
 @TableId(value = "id", type = IdType.*AUTO*)  
 int id;  
 String role;  
  
 public User(String userName, String email, String password, Date createTime, int userId, String mobile, int id, String role) {  
 this.userName = userName;  
 this.email = email;  
 this.password = password;  
 this.createTime = createTime;  
 this.userId = userId;  
 this.mobile = mobile;  
 this.id = id;  
 this.role = role;  
 }  
  
 public User(String userName, String email, String password, Date createTime, int userId, String mobile, String role) {  
 this.userName = userName;  
 this.email = email;  
 this.password = password;  
 this.createTime = createTime;  
 this.userId = userId;  
 this.mobile = mobile;  
 this.role = role;  
 }  
}

package com.example.sb\_bookstore.exception;  
  
  
import lombok.Getter;  
import lombok.Setter;  
  
  
@Getter  
@Setter  
public class CustomException extends Exception{  
 private int errorCode=0;//异常错误代码  
 private String errorMsg="";//异常错误信息  
 public CustomException(int errorCode,String errorMsg){  
 super();  
 //因为异常Exception错误代码code是用的String类型，这里做个转换  
 this.errorCode=errorCode;  
 this.errorMsg=errorMsg;  
 }  
 public CustomException(String errorMsg){  
 super();  
 this.errorMsg=errorMsg;  
 }  
  
}

package com.example.sb\_bookstore.exception;  
  
import jakarta.servlet.http.HttpServletResponse;  
import com.example.sb\_bookstore.response.ResponseData;  
import com.example.sb\_bookstore.utils.ResponseDataUtil;  
import org.springframework.web.bind.annotation.ControllerAdvice;  
import org.springframework.web.bind.annotation.ExceptionHandler;  
import org.springframework.web.bind.annotation.ResponseBody;  
import org.springframework.web.multipart.MaxUploadSizeExceededException;  
  
import java.util.Arrays;  
  
@ControllerAdvice  
public class CustomExceptionHandler {  
  
 @ResponseBody  
 @ExceptionHandler(CustomException.class)  
 public ResponseData myCustomException(CustomException e, HttpServletResponse resp) {  
 //这里使用工具类ResponseDataUtil返回Response数据  
 return ResponseDataUtil.*buildError*(e.getErrorCode(),e.getErrorMsg(),"出现异常了，请检查您的数据或操作！");  
 }  
 @ResponseBody  
 @ExceptionHandler(Exception.class)  
 public ResponseData handleAllException(Exception e,HttpServletResponse resp){  
 return ResponseDataUtil.*buildError*(e.getMessage()+ Arrays.*stream*(e.getStackTrace()).toList());  
 }  
  
}

package com.example.sb\_bookstore.mapper;  
  
import com.baomidou.mybatisplus.core.mapper.BaseMapper;  
import com.example.sb\_bookstore.domain.BookCategory;  
import org.apache.ibatis.annotations.\*;  
  
import java.util.List;  
@Mapper  
public interface BookCategoryMapper extends BaseMapper<BookCategory> {  
 //插入一个类别  
 public int addCategoryReturnId(BookCategory bookCategory);  
  
 @Select("select \* from bookcategory where category\_id=#{cid}")  
 public BookCategory getCategoryByCategoryId(Integer cid);  
  
 //便利cid的子孙类别  
 @Select("select \* from bookcategory where parent\_id =#{cid}")  
 @Results({  
// 多表查询，这里结果为1对n，所以是many= @Many  
 @Result(column = "category\_id",property = "categoryId"),  
 @Result(column="category\_id",property = "children",  
 many=@Many(select = "com.example.sb\_bookstore.mapper.BookCategoryMapper.listCategories")  
 )  
 })  
 public List<BookCategory> listCategories(Integer cid);  
}

<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  
<mapper namespace="com.example.sb\_bookstore.mapper.BookCategoryMapper">  
 <insert id="addCategoryReturnId" useGeneratedKeys="true" keyProperty="id" keyColumn="id"  
 parameterType="com.example.sb\_bookstore.domain.BookCategory">  
 insert into bookcategory(category\_id,parent\_id,category\_name)  
 values (#{categoryId},#{parentId},#{categoryName})  
 </insert>  
  
</mapper>

package com.example.sb\_bookstore.mapper;  
  
import com.baomidou.mybatisplus.core.mapper.BaseMapper;  
import com.example.sb\_bookstore.domain.Book;  
import org.apache.ibatis.annotations.Delete;  
import org.apache.ibatis.annotations.Mapper;  
import org.apache.ibatis.annotations.Select;  
  
import java.util.List;  
  
@Mapper  
public interface BookMapper extends BaseMapper<Book> {  
  
  
  
  
 @Select("select \* from book where id=#{id}")  
 public Book getBookById(int id);  
  
  
 public int addBookReturnBookId(Book book);  
  
 public List<Book> getNewBooks();  
 public List<Book> getHotBooks();  
 public List<Book> getTeachBooks();  
 //增加代码  
 @Select("select \* from book where category\_id=#{cid}")  
 public List<Book> getBooksByCategoryId(int cid);  
  
}

<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  
<mapper namespace="com.example.sb\_bookstore.mapper.BookMapper">  
 <insert id="addBookReturnBookId" useGeneratedKeys="true" keyProperty="id" keyColumn="id"  
 parameterType="com.example.sb\_bookstore.domain.Book">  
 insert into book(book\_id,title,author,price,discount,image,book\_concern,publish\_date,brief,category\_id)  
 values (#{bookId},#{title},#{author},#{price},#{discount},  
 #{image},#{bookConcern},#{publishDate},#{brief},#{categoryId})  
 </insert>  
 <select id="getNewBooks" >  
 select book.\* from book ,newbook where book.book\_id=newbook.book\_id  
 </select>  
 <select id="getHotBooks" >  
 select book.\* from book,hotbook where book.book\_id=hotbook.book\_id  
 </select>  
 <select id="getTeachBooks" >  
 select book.\* from book,teachbook where book.book\_id=teachbook.book\_id  
 </select>  
</mapper>

package com.example.sb\_bookstore.mapper;  
  
import com.baomidou.mybatisplus.core.mapper.BaseMapper;  
import com.example.sb\_bookstore.domain.Comment;  
import org.apache.ibatis.annotations.Mapper;  
import org.apache.ibatis.annotations.Select;  
  
import java.util.List;  
@Mapper  
public interface CommentMapper extends BaseMapper<Comment> {  
 @Select("select \* from comment where comment\_id=#{cid}")  
 public Comment getCommentByCommentId(Integer cid);  
  
 @Select("select \* from comment where book\_id=#{bid}")  
 public List<Comment> getCommentsByBookId(Integer bid);  
  
  
 @Select("select \* from comment where user\_id=#{uid}")  
 public List<Comment> getCommentsByUserId(Integer uid);  
  
  
}

<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  
<mapper namespace="com.example.sb\_bookstore.mapper.CommentMapper">  
 <insert id="addCommentReturnId" useGeneratedKeys="true" keyProperty="id" keyColumn="id"  
 parameterType="com.example.sb\_bookstore.domain.Comment">  
 insert into comment(user\_id,book\_id,content,comment\_date,comment\_id)  
 values (#{userId},#{bookId},#{content},#{commentDate},#{commentId})  
 </insert>  
  
</mapper>

package com.example.sb\_bookstore.mapper;  
  
import com.baomidou.mybatisplus.core.mapper.BaseMapper;  
import com.example.sb\_bookstore.domain.User;  
import lombok.Setter;  
import org.apache.ibatis.annotations.Mapper;  
import org.apache.ibatis.annotations.Select;  
@Mapper  
public interface UserMapper extends BaseMapper<User> {  
  
 @Select("select \* from user where user\_id=#{uid}")  
 public User getUserByUserId(Integer uid);  
  
 @Select("select \* from user where user\_name=#{uname}")  
 public User getUserByUserName(String uname);  
  
 //根据用户名和密码找用户，即用来判断登录  
 @Select("select \* from user where user\_name=#{uname} and password=#{password}")  
 public User getUserByUserNameAndPassword(String uname,String password);  
  
 //插入一个用户，即注册功能  
 public int addUserReturnId(User user);  
}

<?xml version="1.0" encoding="UTF-8"?>  
<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"  
 "http://mybatis.org/dtd/mybatis-3-mapper.dtd">  
<mapper namespace="com.example.sb\_bookstore.mapper.UserMapper">  
 <insert id="addUserReturnId" useGeneratedKeys="true" keyProperty="id" keyColumn="id"  
 parameterType="com.example.sb\_bookstore.domain.User">  
 insert into user(user\_name,email,password,create\_time,user\_id,mobile,role)  
 values (#{userName},#{email},#{password},#{createTime},#{userId},#{mobile},#{role})  
 </insert>  
  
</mapper>

package com.example.sb\_bookstore.response;  
  
import com.example.sb\_bookstore.constants.ResultEnums;  
import lombok.Getter;  
import lombok.Setter;  
  
import java.io.Serializable;  
  
@Getter  
@Setter  
public class ResponseData <T> implements Serializable {  
 private int code;//响应代码  
 private String msg;//响应的消息  
 private T data;//响应传递的数据  
 //这里使用类型T，<T>作为泛化，代表以后使用的类型  
 public ResponseData(int code,String msg, T data){  
 //带数据  
 this.code=code;  
 this.msg=msg;  
 this.data=data;  
 }  
 public ResponseData(int code,String msg){  
 //不带数据  
 this.code=code;  
 this.msg=msg;  
 }  
 public ResponseData(ResultEnums resultEnums,T data){  
 //带数据  
 this.code=resultEnums.getCode();  
 this.msg=resultEnums.getMsg();  
 this.data=data;  
 }  
 public ResponseData(ResultEnums resultEnums){  
 //不带数据  
 this.code= resultEnums.getCode();;  
 this.msg=resultEnums.getMsg();  
 }  
  
}

package com.example.sb\_bookstore.response;  
  
import com.example.sb\_bookstore.constants.ResultEnums;  
import lombok.Getter;  
import lombok.Setter;  
  
import java.io.Serializable;  
  
@Getter  
@Setter  
public class ResponseData2 <T> implements Serializable {  
 private int code;//响应代码  
 private String msg;//响应的消息  
 private T data;//响应传递的数据  
 //这里使用类型T，<T>作为泛化，代表以后使用的类型  
 //total用于返回分页数据的总数  
 private long total;  
 public ResponseData2(int code,String msg, T data,long total){  
 //带数据  
 this.code=code;  
 this.msg=msg;  
 this.data=data;  
 this.total=total;  
 }  
 public ResponseData2(int code,String msg){  
 //不带数据  
 this.code=code;  
 this.msg=msg;  
 }  
 public ResponseData2(ResultEnums resultEnums,T data,long total){  
 //带数据  
 this.code=resultEnums.getCode();  
 this.msg=resultEnums.getMsg();  
 this.data=data;  
 this.total=total;  
 }  
 public ResponseData2(ResultEnums resultEnums){  
 //不带数据  
 this.code= resultEnums.getCode();;  
 this.msg=resultEnums.getMsg();  
 }  
  
}

package com.example.sb\_bookstore.service;  
  
import com.example.sb\_bookstore.domain.BookCategory;  
import com.example.sb\_bookstore.mapper.BookCategoryMapper;  
import org.apache.ibatis.annotations.Select;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
@Service  
public class BookCategoryService {  
 @Autowired  
 BookCategoryMapper bookCategoryMapper;  
  
 //根据categoryid查查category  
  
 public BookCategory getBookCategoryByCategoryId(Integer cid){  
 return bookCategoryMapper.getCategoryByCategoryId(cid);  
 };  
  
  
 //遍历cid子孙具有的bookcategory  
 public List<BookCategory> listCategories(Integer cid){  
 return bookCategoryMapper.listCategories(cid);  
 }  
  
}

package com.example.sb\_bookstore.service;  
  
import com.baomidou.mybatisplus.core.conditions.query.QueryWrapper;  
import com.baomidou.mybatisplus.extension.plugins.pagination.Page;  
import com.example.sb\_bookstore.domain.Book;  
import com.example.sb\_bookstore.mapper.BookMapper;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class BookService {  
  
 @Autowired  
 private BookMapper bookMapper;  
 public int addBook(Book book){  
 //注意的是，这里返回的是插入成功的记录条数  
 //return bookMapper.insert(book);  
 if( bookMapper.addBookReturnBookId(book)!=0)  
 return book.getBookId();//返回图书的book\_id  
 return 0;  
 }  
 public int updateBook(Book book){  
 return bookMapper.updateById(book);  
 }  
  
 public List<Book> getAllBooks(){  
 return bookMapper.selectList(null);  
 }  
 public Book getBookById(Integer id){return bookMapper.getBookById(id);}  
 public List<Book> getNewBooks(){return bookMapper.getNewBooks();}  
 public List<Book> getHotBooks(){return bookMapper.getHotBooks();}  
 public List<Book> getTeachBooks(){return bookMapper.getTeachBooks();}  
 public List<Book> searchBooks(String keyword){  
 //select \* from book where title like XXX or author like XXX or brief like XXX or book\_concern like XXX  
 QueryWrapper<Book> queryWrapper=new QueryWrapper<>();  
 queryWrapper.like("title",keyword);  
 queryWrapper.or();  
 queryWrapper.like("author",keyword);  
 queryWrapper.or();  
 queryWrapper.like("brief",keyword);  
 queryWrapper.or();  
 queryWrapper.like("book\_concern",keyword);  
 return bookMapper.selectList(queryWrapper);  
 }  
 //修改增加代码  
 public List<Book> getBooksByCategoryId(Integer cid){  
 return bookMapper.getBooksByCategoryId(cid);  
 }  
 public Page<Book> selectPageBySearch(Page<Book> page, String keyword){  
 QueryWrapper<Book> queryWrapper=new QueryWrapper<>();  
 queryWrapper.like("title",keyword);  
 queryWrapper.or();  
 queryWrapper.like("author",keyword);  
 queryWrapper.or();  
 queryWrapper.like("brief",keyword);  
 queryWrapper.or();  
 queryWrapper.like("book\_concern",keyword);  
 return bookMapper.selectPage(page,queryWrapper);  
 }  
 public Page<Book> selectPageByCategory(Page<Book> page,Integer cid){  
 QueryWrapper<Book> queryWrapper=new QueryWrapper<>();  
 queryWrapper.eq("category\_id",cid);  
 return bookMapper.selectPage(page,queryWrapper);  
 }  
  
}

package com.example.sb\_bookstore.service;  
  
import com.example.sb\_bookstore.domain.Comment;  
import com.example.sb\_bookstore.mapper.CommentMapper;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
import java.util.List;  
  
@Service  
public class CommentService {  
 @Autowired  
 CommentMapper commentMapper;  
 public List<Comment> getAllComments(){return commentMapper.selectList(null);}  
 public Comment getCommentByCommentId(Integer cid){return commentMapper.getCommentByCommentId(cid);}  
 public List<Comment> getCommentsByBookId(Integer bid){  
 return commentMapper.getCommentsByBookId(bid);  
 }  
 public List<Comment> getCommentsByUserId(Integer uid){  
 return commentMapper.getCommentsByUserId(uid);  
 }  
  
}

package com.example.sb\_bookstore.service;  
  
import com.example.sb\_bookstore.domain.User;  
import com.example.sb\_bookstore.mapper.UserMapper;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
  
@Service  
public class UserService {  
 @Autowired  
 UserMapper userMapper;  
 public User getUserByUserId(Integer uid){  
 return userMapper.getUserByUserId(uid);  
 }  
 public User getUserByUserName(String uname){  
 return userMapper.getUserByUserName(uname);  
 }  
 public User getUserByUserNameAndPassword(String uname,String password){  
 return userMapper.getUserByUserNameAndPassword(uname,password);  
 }  
 public int addUserReturnId(User user){  
 return userMapper.addUserReturnId(user);  
 }  
}

package com.example.sb\_bookstore;  
  
import org.mybatis.spring.annotation.MapperScan;  
import org.springframework.boot.SpringApplication;  
import org.springframework.boot.autoconfigure.SpringBootApplication;  
@MapperScan("com.example.sb\_bookstore.mapper")  
@SpringBootApplication  
public class SbBookstoreApplication {  
  
 public static void main(String[] args) {  
 SpringApplication.*run*(SbBookstoreApplication.class, args);  
 }  
  
}

package com.example.sb\_bookstore.utils;  
  
import java.util.UUID;  
  
public class RandomUtil {  
 public static int getRandom(){  
 //位数9，int最大的位数，使用UUID产生随机整数  
 int bits=9;  
 // 生成UUID  
 int temp=UUID.*randomUUID*().toString().replace("-","").hashCode();  
 temp=temp<0?-temp:temp;  
 String tempStr=Integer.*toString*(temp);  
 while(tempStr.length()<bits){  
 tempStr=tempStr+(int)(Math.*random*()\*10);  
 }  
 if(tempStr.length()>bits){  
 tempStr=tempStr.substring(0,bits);  
 }  
 return Integer.*parseInt*(tempStr);  
 }  
  
}

package com.example.sb\_bookstore.utils;  
  
import com.example.sb\_bookstore.constants.ResultEnums;  
import com.example.sb\_bookstore.response.ResponseData;  
import com.example.sb\_bookstore.response.ResponseData2;  
  
@SuppressWarnings("rawtypes")//去除使用原型的不安全警告  
public class ResponseData2Util {  
 //工具类一般都是静态方法，可以直接调用，不用实例化  
 //正确的响应情况返回  
 public static <T> ResponseData2 buildSuccess(int code , String msg, T data,long total){  
 //带数据  
 return new ResponseData2<T>(code, msg, data,total);  
 }  
 public static ResponseData2 buildSuccess(int code,String msg){  
 //不带数据  
 return new ResponseData2(code,msg);  
 }  
  
 public static <T> ResponseData2 buildSuccess(T data,long total){  
 //只带数据，如果不给出成功代码，默认使用Success的代码，即code=200，msg=“请求成功”  
 return new ResponseData2<>(ResultEnums.*SUCCESS*.getCode(),ResultEnums.*SUCCESS*.getMsg(),data,total);  
 }  
 public static ResponseData2 buildSuccess(ResultEnums resultEnums){  
 return new ResponseData2(resultEnums.getCode(),resultEnums.getMsg());  
 }  
 public static ResponseData2 buildSuccess(){  
 //不带参数，默认是用SUCCESS  
 return new ResponseData2(ResultEnums.*SUCCESS*);  
 }  
  
 //错误的情况返回数据  
 public static <T> ResponseData2 buildError(int code ,String msg,T data,long total){  
 //带数据  
 return new ResponseData2<T>(code, msg, data,total);  
 }  
 public static ResponseData2 buildError(int code,String msg){  
 //不带数据  
 return new ResponseData2(code,msg);  
 }  
 public static <T> ResponseData2 buildError(T data,long total){  
 //只带数据，如果不给出成功代码，默认使用ERROR的代码，即code=400，msg=“请求失败”  
 return new ResponseData2<>(ResultEnums.*ERROR*.getCode(),ResultEnums.*ERROR*.getMsg(),data,total);  
 }  
 public static ResponseData2 buildError(ResultEnums resultEnums){  
 return new ResponseData2(resultEnums.getCode(),resultEnums.getMsg());  
 }  
 public static ResponseData2 buildError(){  
 //不带参数，默认是用ERROR  
 return new ResponseData2(ResultEnums.*ERROR*);  
 }  
  
}

package com.example.sb\_bookstore.utils;  
  
import com.example.sb\_bookstore.constants.ResultEnums;  
import com.example.sb\_bookstore.response.ResponseData;  
@SuppressWarnings("rawtypes")//去除使用原型的不安全警告  
public class ResponseDataUtil {  
 //工具类一般都是静态方法，可以直接调用，不用实例化  
 //正确的响应情况返回  
 public static <T> ResponseData buildSuccess(int code ,String msg,T data){  
 //带数据  
 return new ResponseData<T>(code, msg, data);  
 }  
 public static ResponseData buildSuccess(int code,String msg){  
 //不带数据  
 return new ResponseData(code,msg);  
 }  
  
 public static <T> ResponseData buildSuccess(T data){  
 //只带数据，如果不给出成功代码，默认使用Success的代码，即code=200，msg=“请求成功”  
 return new ResponseData<>(ResultEnums.*SUCCESS*.getCode(),ResultEnums.*SUCCESS*.getMsg(),data);  
 }  
 public static ResponseData buildSuccess(ResultEnums resultEnums){  
 return new ResponseData(resultEnums.getCode(),resultEnums.getMsg());  
 }  
 public static ResponseData buildSuccess(){  
 //不带参数，默认是用SUCCESS  
 return new ResponseData(ResultEnums.*SUCCESS*);  
 }  
  
 //错误的情况返回数据  
 public static <T> ResponseData buildError(int code ,String msg,T data){  
 //带数据  
 return new ResponseData<T>(code, msg, data);  
 }  
 public static ResponseData buildError(int code,String msg){  
 //不带数据  
 return new ResponseData(code,msg);  
 }  
 public static <T> ResponseData buildError(T data){  
 //只带数据，如果不给出成功代码，默认使用ERROR的代码，即code=400，msg=“请求失败”  
 return new ResponseData<>(ResultEnums.*ERROR*.getCode(),ResultEnums.*ERROR*.getMsg(),data);  
 }  
 public static ResponseData buildError(ResultEnums resultEnums){  
 return new ResponseData(resultEnums.getCode(),resultEnums.getMsg());  
 }  
 public static ResponseData buildError(){  
 //不带参数，默认是用ERROR  
 return new ResponseData(ResultEnums.*ERROR*);  
 }  
  
}

spring.datasource.type=com.alibaba.druid.pool.DruidDataSource  
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver  
#??????bookstore????myuser???mypass  
spring.datasource.url=jdbc:mysql://localhost:3306/bookstore  
spring.datasource.username=myuser  
spring.datasource.password=mypass  
  
#??classpath\*:/mapper/\*\*/\*.xml,???????????????  
mybatis-plus.mapper-locations=classpath\*:/mapper/\*.xml  
spring.web.resources.static-locations=classpath:/META-INF/resources/,classpath:/resources/,classpath:/static/,classpath:/public/,classpath:/templates/

完整代码具体见开源网址：