**电子书在线阅读应用**

**设计文档**

Github地址：https://github.com/xuzihan125/JavaEEProject1

目录

[1.系统功能设计 3](#_Toc32425)

[1.1 功能需求 3](#_Toc4708)

[1.2系统流程分析 3](#_Toc6049)

[1.3 系统架构分析 4](#_Toc17738)

[2.系统数据库设计 5](#_Toc18462)

[3 系统核心模块说明 6](#_Toc4197)

[3.1 Controller模块 6](#_Toc10426)

[3.2 Service模块 6](#_Toc8687)

[3.2.1 BookListService 6](#_Toc28326)

[3.2.2 BookViewService 6](#_Toc8875)

[3.2.3 FileService 6](#_Toc11917)

[3.2.4 LoginService 7](#_Toc28617)

[3.2.5 SwitchService 7](#_Toc10080)

[3.3 Templates模块 7](#_Toc20714)

[3.4 Mybatis模块 7](#_Toc23757)

[3.5 Authorization和Authentication模块 8](#_Toc29520)

[3.6 异常处理模块 8](#_Toc17970)

[3.7 系统日志 8](#_Toc31611)

[3.8 文件传输模块 8](#_Toc15937)

[3.9 文件处理模块 8](#_Toc18627)

[3.10 Swagger框架 9](#_Toc179)

[3.11 RESTful接口 9](#_Toc8879)

[3.12 JWT权限加密 10](#_Toc11972)

[3.13 跨越 10](#_Toc1562)

[4 项目截图 10](#_Toc22600)

[5 项目测试 11](#_Toc14435)

[5.1 单元测试 12](#_Toc13969)

[5.1.1 BooksMapperTest 12](#_Toc14652)

[5.1.2 LogMapperTest 12](#_Toc28557)

[5.1.3 SysUserMapperTest 12](#_Toc12244)

[5.1.4 UserRoleMapperTest 13](#_Toc10941)

[5.1.5 RoleMapperExtTest 14](#_Toc5746)

[5.2 集成测试 14](#_Toc20214)

[5,2,1 AdminControllerTest 14](#_Toc16309)

[5.2.2 BooklistControllerTest 15](#_Toc14085)

[5.2.3 BookViewControllerTest 15](#_Toc27419)

[5.2.4 FileControllerTest 15](#_Toc23734)

[5.2.5 LogControllerTest 15](#_Toc2687)

[5.2.6 LoginControllerTest 16](#_Toc13514)

[5.2.7 SwitchControllerTest 16](#_Toc10402)

# 1.系统功能设计

## 1.1 功能需求

E-BookShop网站的设计与开发主要包括后台数据库的建立以及前端网站页面的简单设计开发两个方面。在此初次开发中，我们对于数据库的功能需求是数据一致性和完整性强，对于前端网页设计的功能需求是在可操作所有后端功能与展示完整数据信息的前提下，追求简洁易操作。本次开发中，主体以MVC框架+Mybatis+Thymeleaf实现客户与管理员区别访问、管理员上架书籍完整信息与查看日志、客户搜索与浏览书籍信息等功能，并使用了Spring Security实现授权管理与登录拦截功能，使用HttpSession实现邮箱账户的cookie存取功能。

## 1.2系统流程分析

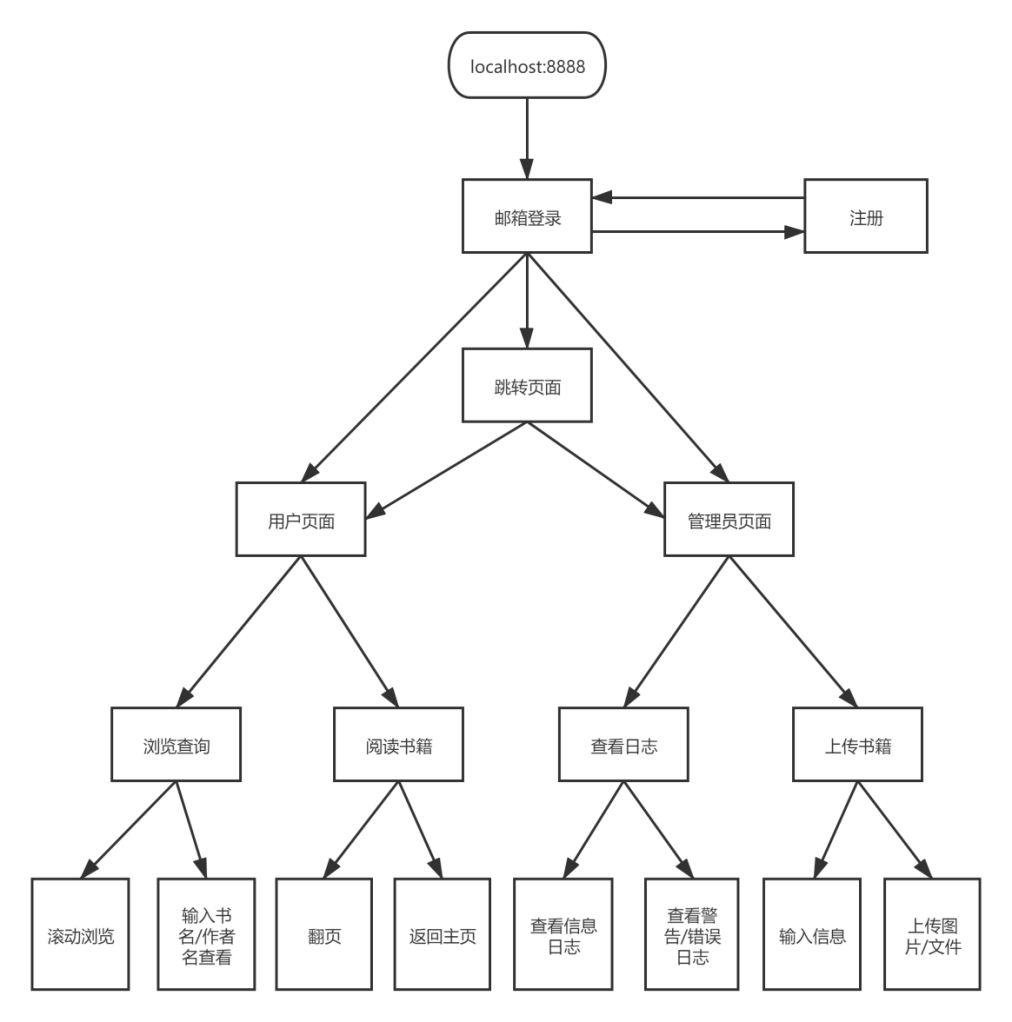


图1：系统功能流程图

本系统的操作流程主要分为用户操作和管理员操作两部分，用户操作主要是对用户的帐号和用户密码进行身份验证，通过验证的进入书籍查找浏览界面，继而可进行浏览查询与阅读书籍的操作，而身份验证失败时，则给出错误提示。管理员操作是通过管理员验证授权后，可以查看系统日志与上传书籍。

## 1.3 系统架构分析

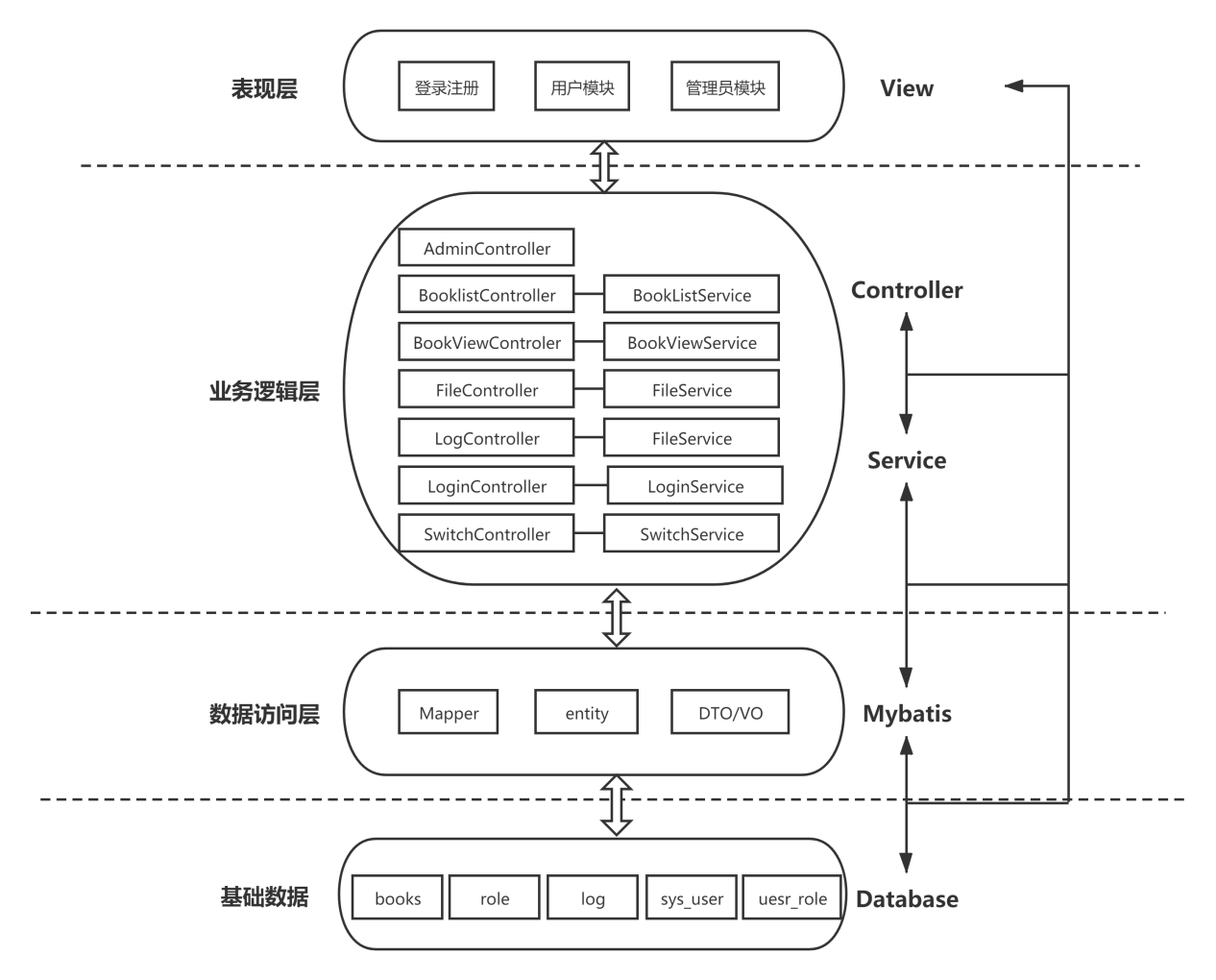


图2：系统总体架构图

# 2.系统数据库设计

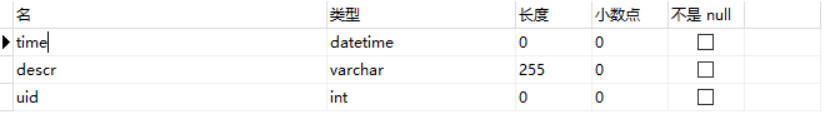
在本系统的初次开发中，主要将系统分为登陆注册、查看、增添三大模块，使用者分为用户、管理员两大群体。

在E-BookShop数据库中各个表的设计结果如下所示：

书籍信息表(ebook.books)



登录记录日志表(ebook.log)



系统角色表(ebook.role)



系统使用用户表(ebook.sys\_user)



系统用户角色对应表(ebook.user\_role)



# 3 系统核心模块说明

本项目github地址为：https://github.com/xuzihan125/JavaEEProject1

## 3.1 Controller模块

系统共有七大Controller类，类中并没有实际的处理操作，而是将系统的View与Service分隔开来，而又联系起来，Controller中对应的类收到相应的请求时，调用相应的Service中的模型方法进行处理，而处理结果通过Controller类返回确认视图，在保证前后端顺利完成任务同时，实现前后端分离。

## 3.2 Service模块

### 3.2.1 BookListService

ModelAndView getBookList(BookListVO bookListVO)方法实现用户浏览查询书籍功能，其中使用BooksExample对查询书籍进行搜索定位，进行完相应的Mapper处理后将处理结果以ModelAndView对象返回。

### 3.2.2 BookViewService

ModelAndView getBookView(String uid,Integer page)方法实现用户阅读书籍功能。以BooksExample对书籍进行定位，用Mapper操作确定其书籍信息与书籍内容，处理完成后以ModelAndView对象返回。

### 3.2.3 FileService

void getPDF(String file, HttpServletResponse response)方法实现以绝对路径获取书籍PDF资源的功能。以response进行函数响应

ModelAndView getLog(String type)方法实现管理员查看日志的功能，自定义系统内日志的生成。日志会显示在控制台内，同时输出到指定文件中。程序通过绝对路径+日志相对路径获取运行日志文件，并在日志页面中进行展示。

ModelAndView addBook(BookVO bookVO, HttpServletRequest request)方法实现管理员上架书籍功能。管理员首先输入书籍信息，系统以booksMapper将书插入，自动生成书籍ID。而后根据绝对路径+相对路径进行书籍封面与书籍内容设置，其中使用multipart进行文件传输，可同时上传pdf和图片附件，实现系统的图片回显。设置时发生异常则删除刚插入的书籍信息，成功则将封面与内容匹配插入，页面清空，显示上传成功。

### 3.2.4 LoginService

ModelAndView checkLogin(@Valid UserVO userVO, HttpSession session)方法实现检查登录功能。系统以SysUserExample与sysUserMapper对用户输入的邮箱与密码进行校验，校验成功后，以RoleMapperExt对账户进行角色审查，进入跳转至Switch服务。

ModelAndView logout(HttpSession session)方法实现退出功能，退出回login页面，并将缓存邮箱账号删除。

ModelAndView regist(@Valid UserVO userVO)方法实现账户注册功能。首先对邮箱进行sysUserMapper校验，确保此为未注册邮箱，进而再以sysUserMapper将注册成功的用户插入到数据库中，且初次注册时时钟为普通用户角色。

### 3.2.5 SwitchService

ModelAndView switchView(HttpServletRequest request)方法实现用户登录后的权限校验。根据账户的rid匹配进入不同页面。

## 3.3 Templates模块

完成每一个前端展示页面的组件部署与页面布局，并对通过Controller与后端程序完成对接。

## 3.4 Mybatis模块

持久层框架，免除了几乎所有的 JDBC 代码以及设置参数和获取结果集的工作。MyBatis 可以通过简单的 XML 或注解来进行配置和高级映射。其中Dao包中的Mapper类反向注入，获得单表查询（反向注入配置为common/config/generator，resource/config/），完成书籍信息、日志信息、用户信息、用户角色匹配映射，直接建立数据库与Service之间的连接部署；其中entity包中皆为一种表达业务概念的完整实体，系统中给出表示完整书籍、日志、系统用户、系统角色的example，在Service中对它们进行调用，以处理一些匹配映射、校验等操作；vo包中主要对一些数据类型的错误形式进行注解；

## 3.5 Authorization和Authentication模块

我们使用拦截器和session来完成系统的权限授予和验证。

对于权限授予，我们在用户登录时，系统将在session内写入对应的权限，从而实现权限授予。当用户退出时，session内的权限会被删除。

我们通过拦截器来实现权限的验证。我们对需要权限进行访问的页面配置了拦截器进行拦截。当用户访问需要特定权限才能访问的页面时，拦截器会对访问进行拦截，验证其中的权限信息。如果该用户不具有访问权限，则将该用户重定向至登录界面。

## 3.6 异常处理模块

我们使用异常处理器来对全局产生的异常进行自定义处理。当系统因为某些位置异常出现时，系统中会将该异常送入异常处理器，拦截器内通过判断该异常的类型对异常进行处理。经过处理后的异常会被返回至自定义的异常显示界面。

## 3.7 系统日志

我们通过对springboot框架内的日志系统进行配置，将所有的日志在显示与控制台的同时，在文件内进行存储，供管理员直接调用。便于后期的维护及测试。除此之外，并在各个controller、service子模块内进行日志信息提示，对危险信息生成警告信息，对系统内产生的错误生成错误信息。

## 3.8 文件传输模块

我们使用multipart来处理系统的文件传输。我们对系统的信息传输进行设置，基于目前的系统情况，对文件传输的大小和数量不做限制。

## 3.9 文件处理模块

为了便于查看，系统内会对所有的pdf文件进行图片转化，将pdf拆分成图片组，并以此实现电子书内容的展示。

## 3.10 Swagger框架

Swagger 是一个规范和完整的框架，用于生成、描述、调用和可视化RESTful风格的 Web 服务。总体目标是使客户端和文件系统作为服务器以同样的速度来更新。文件的方法，参数和模型紧密集成到服务器端的代码，允许API来始终保持同步。Swagger让部署管理和使用功能强大的API变得非常简单。

使用Swagger框架，就是把相关的信息存储在它定义的描述文件里面（yml或json格式），再通过维护这个描述文件可以去更新接口文档，以及生成各端代码。而Springfox-swagger，则可以通过扫描代码去生成这个描述文件，连描述文件都不需要再去维护了。所有的信息，都在代码里面了。代码即接口文档，接口文档即代码。

Swagger采用Open API规范，Open API规范这类API定义语言能够帮助你更简单、快速的表述API，尤其是在API的设计阶段作用特别突出。一旦编写完成，API文档可以作为：

· 需求和系统特性描述的根据。

· 前后台查询、讨论、自测的基础。

· 部分或者全部代码自动生成的根据。

· 其他重要的作用，比如开放平台开发者的手册。

## 3.11 RESTful接口

REST（英文：Representational State Transfer，简称REST）描述了一个架构样式的网络系统，比如 web 应用程序。它首次出现在 2000 年 Roy Fielding 的博士论文中，Roy Fielding是 HTTP 规范的主要编写者之一。在目前主流的三种Web服务交互方案中，REST相比于SOAP（Simple Object Access protocol，简单对象访问协议）以及XML-RPC更加简单明了，无论是对URL的处理还是对Payload的编码，REST都倾向于用更加简单轻量的方法设计和实现。值得注意的是REST并没有一个明确的标准，而更像是一种设计的风格。

REST 指的是一组架构约束条件和原则。满足这些约束条件和原则的应用程序或设计就是 RESTful。

Web 应用程序最重要的 REST 原则是，客户端和服务器之间的交互在请求之间是无状态的。从客户端到服务器的每个请求都必须包含理解请求所必需的信息。如果服务器在请求之间的任何时间点重启，客户端不会得到通知。此外，无状态请求可以由任何可用服务器回答，这十分适合云计算之类的环境。客户端可以缓存数据以改进性能。

在服务器端，应用程序状态和功能可以分为各种资源。资源是一个有趣的概念实体，它向客户端公开。资源的例子有：应用程序对象、数据库记录、算法等等。每个资源都使用 URI (Universal Resource Identifier) 得到一个唯一的地址。所有资源都共享统一的接口，以便在客户端和服务器之间传输状态。使用的是标准的 HTTP 方法，比如 GET、PUT、POST 和 DELETE。Hypermedia 是应用程序状态的引擎，资源表示通过超链接互联。

## 3.12 JWT权限加密

Json web token (JWT), 是为了在网络应用环境间传递声明而执行的一种基于JSON的开放标准（(RFC 7519).该token被设计为紧凑且安全的，特别适用于分布式站点的单点登录（SSO）场景。JWT的声明一般被用来在身份提供者和服务提供者间传递被认证的用户身份信息，以便于从资源服务器获取资源，也可以增加一些额外的其它业务逻辑所必须的声明信息，该token也可直接被用于认证，也可被加密。

## 3.13 跨越

跨域指的是浏览器不能执行其他网站的脚本。它是由浏览器的同源策略造成的，是浏览器对javascript施加的安全限制。

例如：a页面想获取b页面资源，如果a、b页面的协议、域名、端口、子域名不同，所进行的访问行动都是跨域的，而浏览器为了安全问题一般都限制了跨域访问，也就是不允许跨域请求资源。注意：跨域限制访问，其实是浏览器的限制。理解这一点很重要。因为本项目此阶段已经实现了前后端分离，因此需要在Spring Boot中设置对应的跨域Config来保证前后端数据对接正常。

## 3.14 微服务

得益于以Docker为代表的容器化技术的成熟以及DevOps文化的兴起，服务化的思想进一步演化，演变为今天我们所熟知的微服务。微服务相比于服务化有以下4个特点：

1.服务拆分粒度更细。微服务可以说是更细维度的服务化，小到一个子模块，只要该模块依赖的资源与其他模块都没有关系，那么就可以拆分为一个微服务。

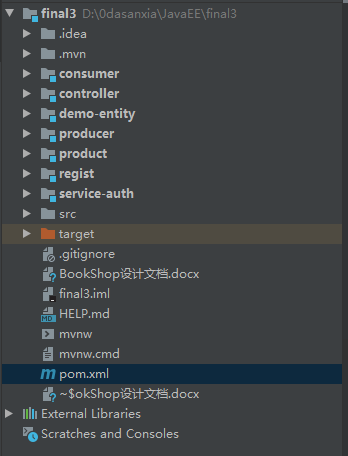
2.服务独立部署。每个微服务都严格遵循独立打包部署的准则，互不影响。比如一台物理机上可以部署多个Docker实例，每个Docker实例可以部署一个微服务的代码。

3.服务独立维护。每个微服务都可以交由一个小团队甚至个人来开发、测试、发布和运维，并对整个生命周期负责。

4.服务治理能力要求高。因为拆分为微服务之后，服务的数量变多，因此需要有统一的服务治理平台，来对各个服务进行管理。

继续以前面举的微博系统为例，可以进一步对内容模块的功能进行拆分，比如内容模块又包含了feed模块、评论模块和个人页模块。通过微服务化，将这三个模块变成三个独立的服务，每个服务依赖各自的资源，并独立部署在不同的服务池中，可以由不同的开发人员进行维护。当评论服务需求变更时，只需要修改评论业务相关的代码，并独立上线发布；而feed服务和个人页服务不需要变更，也不会受到发布可能带来的变更影响。

在我们的项目中，我们几个不同的模块划分成了不同的模块，最终实现了对应的微服务，具体的项目如下图所示：



由此可见，微服务化给服务的发布和部署，以及服务的保障带来了诸多好处。

## 3.15 Kafka

Kafka目前主要作为一个分布式的发布订阅式的消息系统使用，下面简单介绍一下kafka的基本机制。

Producer即生产者，向Kafka集群发送消息，在发送消息之前，会对消息进行分类，即Topic，上图展示了两个producer发送了分类为topic1的消息，另外一个发送了topic2的消息。

Topic即主题，通过对消息指定主题可以将消息分类，消费者可以只关注自己需要的Topic中的消息

Consumer即消费者，消费者通过与kafka集群建立长连接的方式，不断地从集群中拉取消息，然后可以对这些消息进行处理。

从上图中就可以看出同一个Topic下的消费者和生产者的数量并不是对应的。

谈到kafka的存储，就不得不提到分区，即partitions，创建一个topic时，同时可以指定分区数目，分区数越多，其吞吐量也越大，但是需要的资源也越多，同时也会导致更高的不可用性，kafka在接收到生产者发送的消息之后，会根据均衡策略将消息存储到不同的分区中。

在每个分区中，消息以顺序存储，最晚接收的的消息会最后被消费。

生产者在向kafka集群发送消息的时候，可以通过指定分区来发送到指定的分区中。

也可以通过指定均衡策略来将消息发送到不同的分区中。

如果不指定，就会采用默认的随机均衡策略，将消息随机的存储到不同的分区中

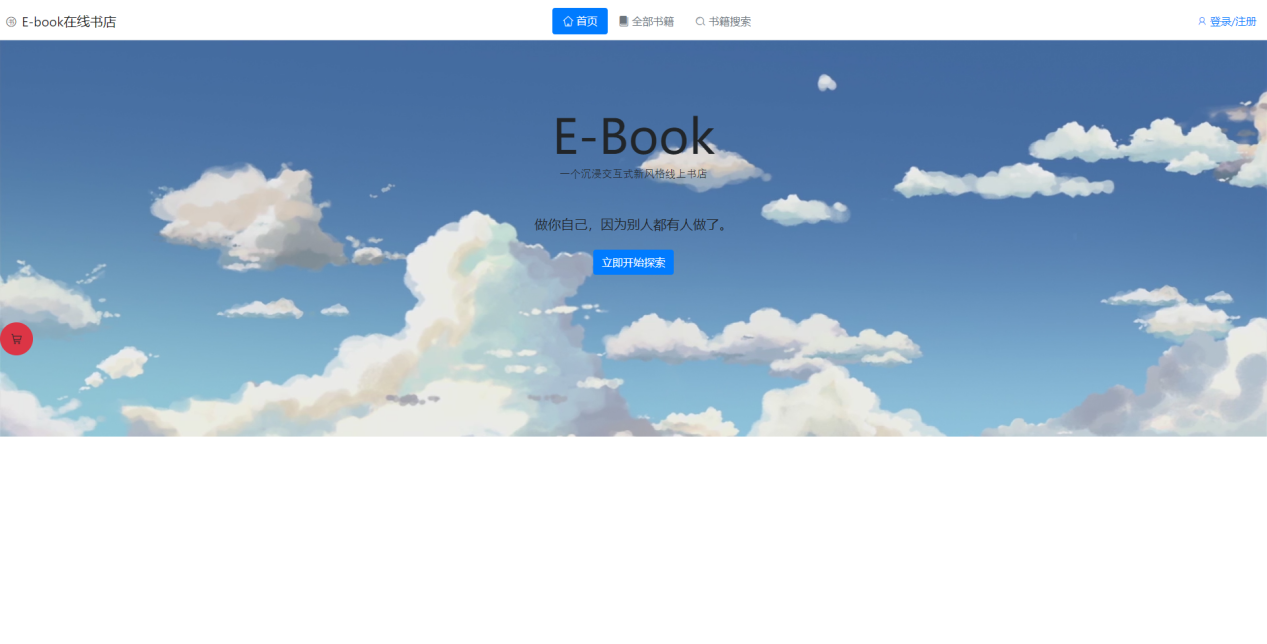
在消费者消费消息时，kafka使用offset来记录当前消费的位置。

在kafka的设计中，可以有多个不同的group来同时消费同一个topic下的消息，如图，我们有两个不同的group同时消费，他们的的消费的记录位置offset各不项目，不互相干扰。

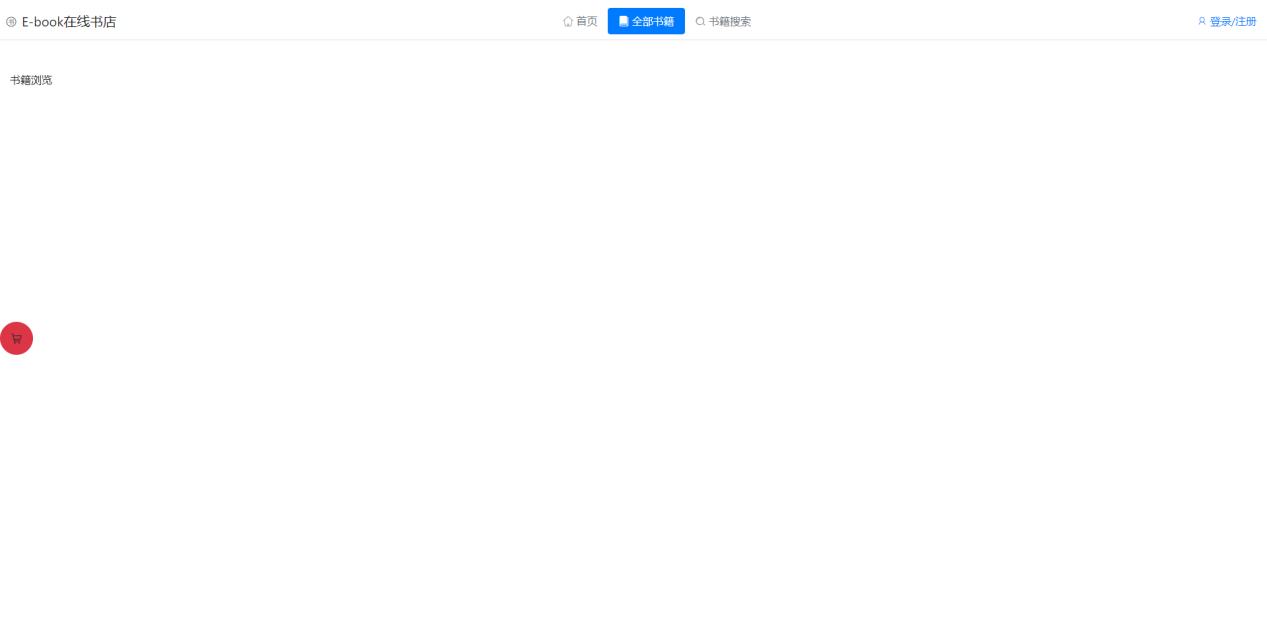
对于一个group而言，消费者的数量不应该多余分区的数量，因为在一个group中，每个分区至多只能绑定到一个消费者上，即一个消费者可以消费多个分区，一个分区只能给一个消费者消费

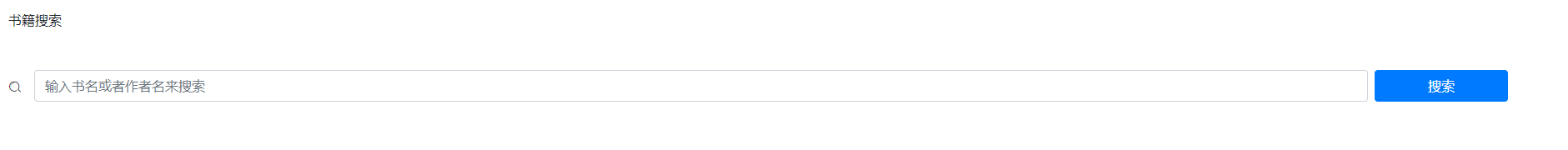
因此，若一个group中的消费者数量大于分区数量的话，多余的消费者将不会收到任何消息。

# 4 项目截图









# 5 项目测试

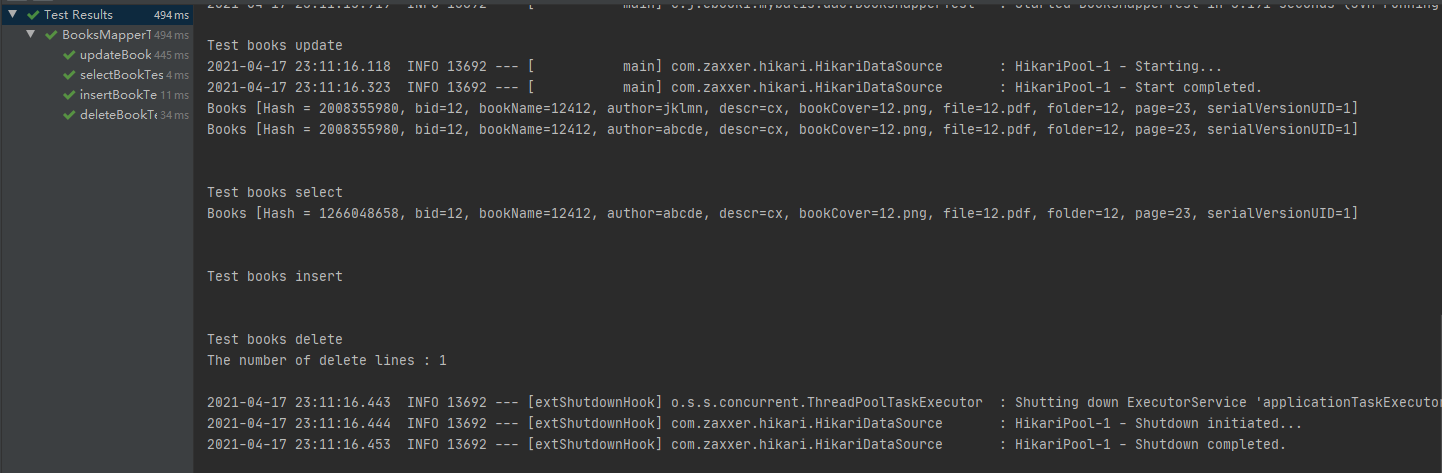
本项目中的测试分为单元测试和集成测试，基于MyBatis的自动化生成Mapper，我们可以直接使用Mapper中对应的生成方法。在集成测试中，则使用Mockmvc或直接使用controller的方式进行测试。

测试整体代码位于github仓库中，这里仅演示测试结果。

## 5.1 单元测试

### 5.1.1 BooksMapperTest

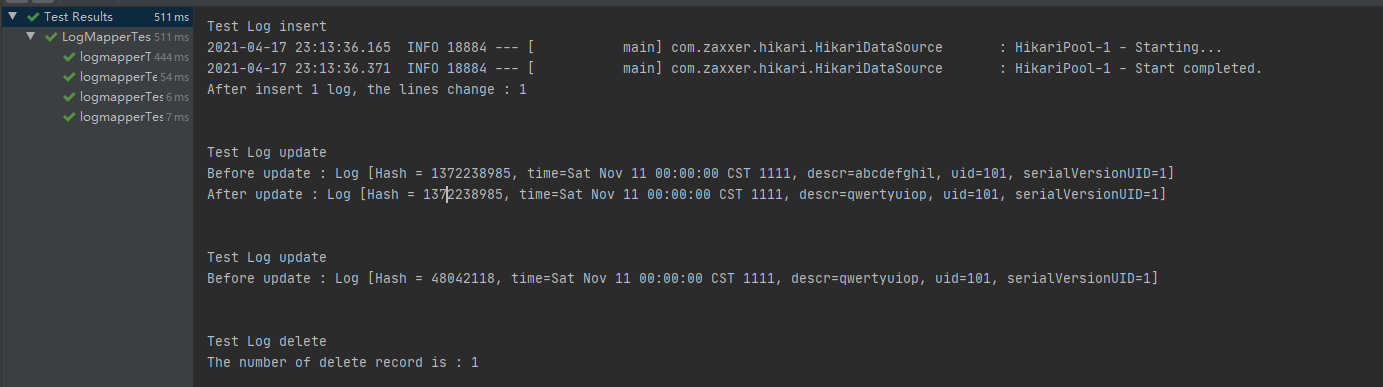
BooksMapper中包含书名，作者名，描述等内容，需要进行增删改查四种操作。最终测试结果如下图所示：



update操作成功修改了Books下的author，选择语句则选择了对应bid的书籍，插入操作则成功插入，并在之后的delete操作中被删除。四条测试语句通过，BooksMapperTest成功。

### 5.1.2 LogMapperTest

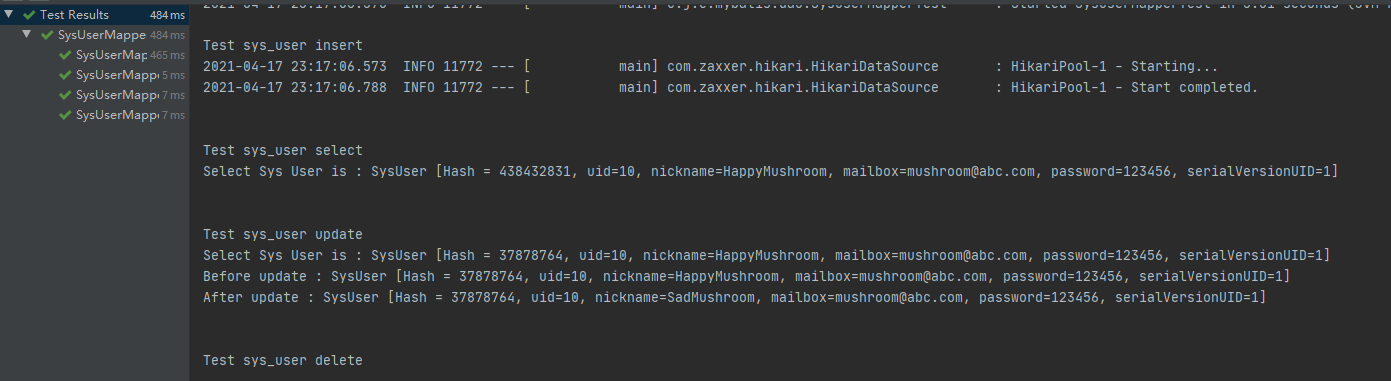
LogMapper中包含了描述，日期和对应编号，需要进行增删改查四种操作。最终测试结果如下图所示：



可以看到，在LogMapper中的insert操作成功，因此修改的行数为1,；update操作成功修改了log的描述，选择语句则成功选择了对应的内容，最后delete成功，数据库中无变动，且修改语句数为1.

### 5.1.3 SysUserMapperTest

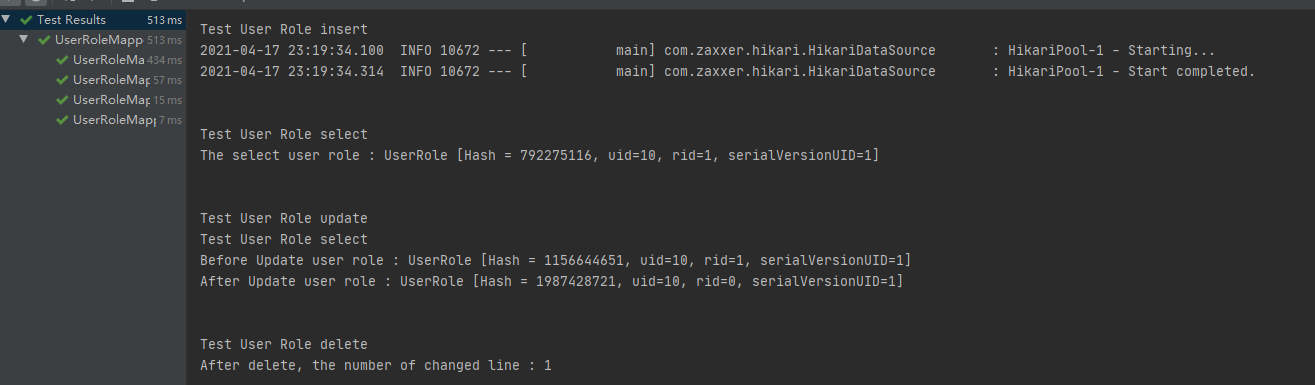
SysUserMapper中包含了用户的序号，昵称，邮件和密码，需要进行增删改查四种操作。因此最终测试结果如下图所示：



在insert中的断言成功，即判断修改行数为1，符合预期，成功插入；选择语句则成功根据select条件选出对应内容；update在select的基础上修改了对应的昵称；而delete操作则将之前插入的内容删除，最终断言修改行数为1，成功删除。

### 5.1.4 UserRoleMapperTest

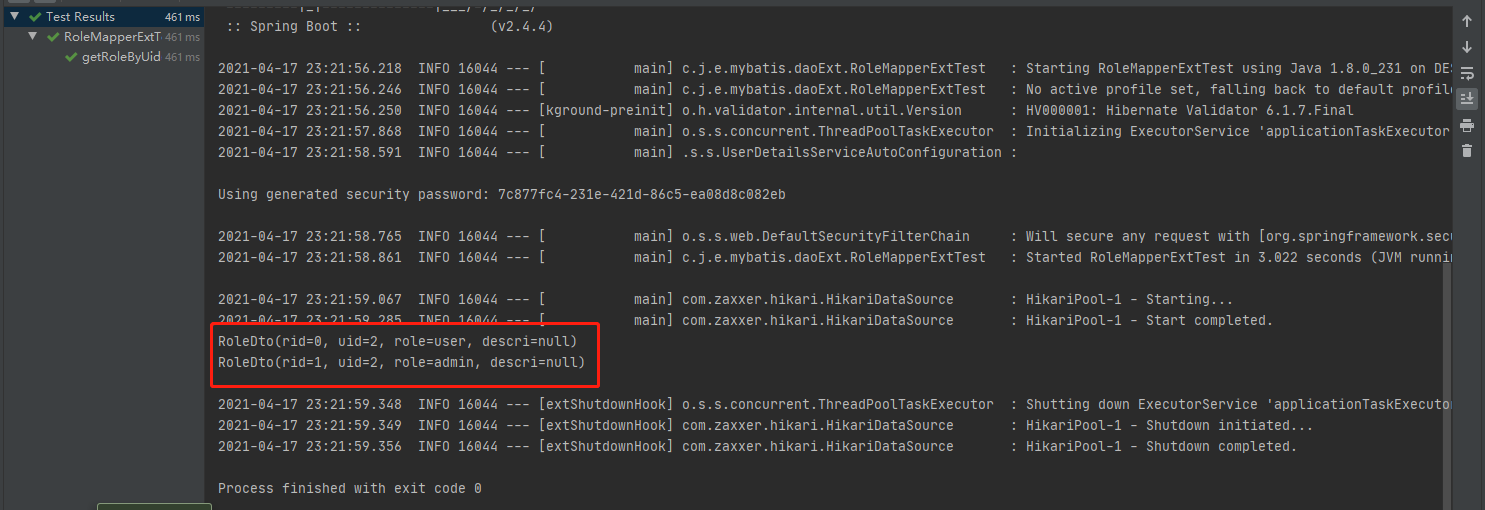
UserRoleMapperTest中包含了uid和rid，即用户编号和角色编号，需要进行增删改查四种操作。最终测试结果如下：

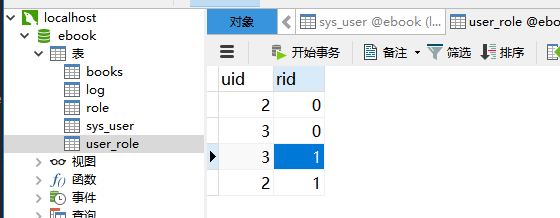


其中插入成功进行，最终断言结果为真，选择语句则成功选择出uid为10的系统角色，update则将uid为10的用户的rid从1修改为0，最后成功将其删除，修改的行数为1.

### 5.1.5 RoleMapperExtTest

RoleMapperExtTest中只提供了一条查询语句，即根据uid判断用户所拥有的身份数量，最终判断出uid为2的用户同时包含了user和admin两种身份，与数据库中的内容相符合，测试成功。

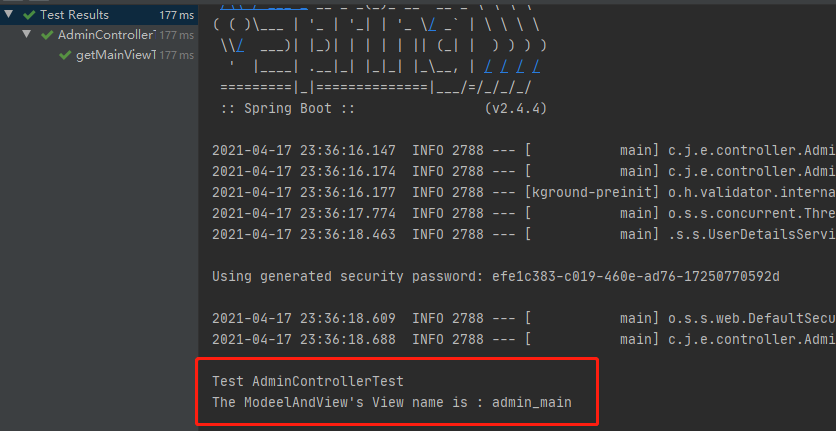




## 5.2 集成测试

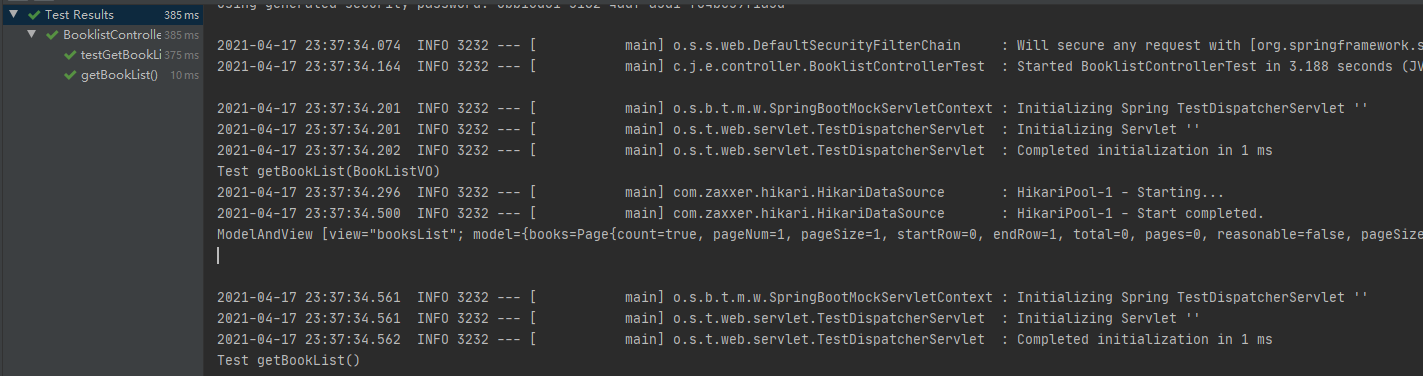
### 5,2,1 AdminControllerTest

AdminControllerTest主要进行判断是否能登录管理页面，这里通过对比对应的ViewName进行比较是否成功验证，最终成功显示。



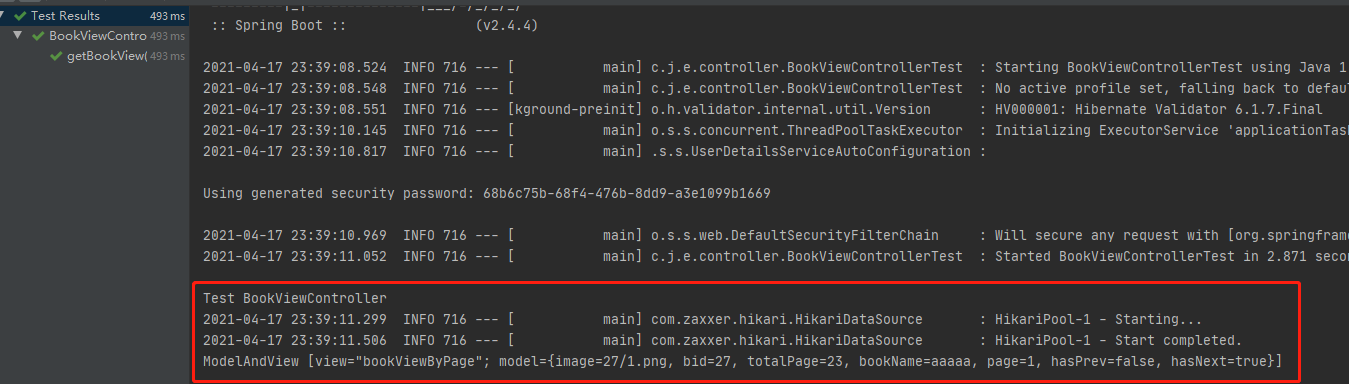
### 5.2.2 BooklistControllerTest

这里使用MockBean进行验证，使用MockMvc进行MVC访问控制，在两个GetBookList方法中分别进行测试，最终成功实现BookList的返回和测试。



### 5.2.3 BookViewControllerTest

在BookViewControllerTest中，需要对书籍进行对应的检查，通过调用对应Controller方法并获取ModelAndView进行检查后，成功获取到对应书籍的信息和内容。



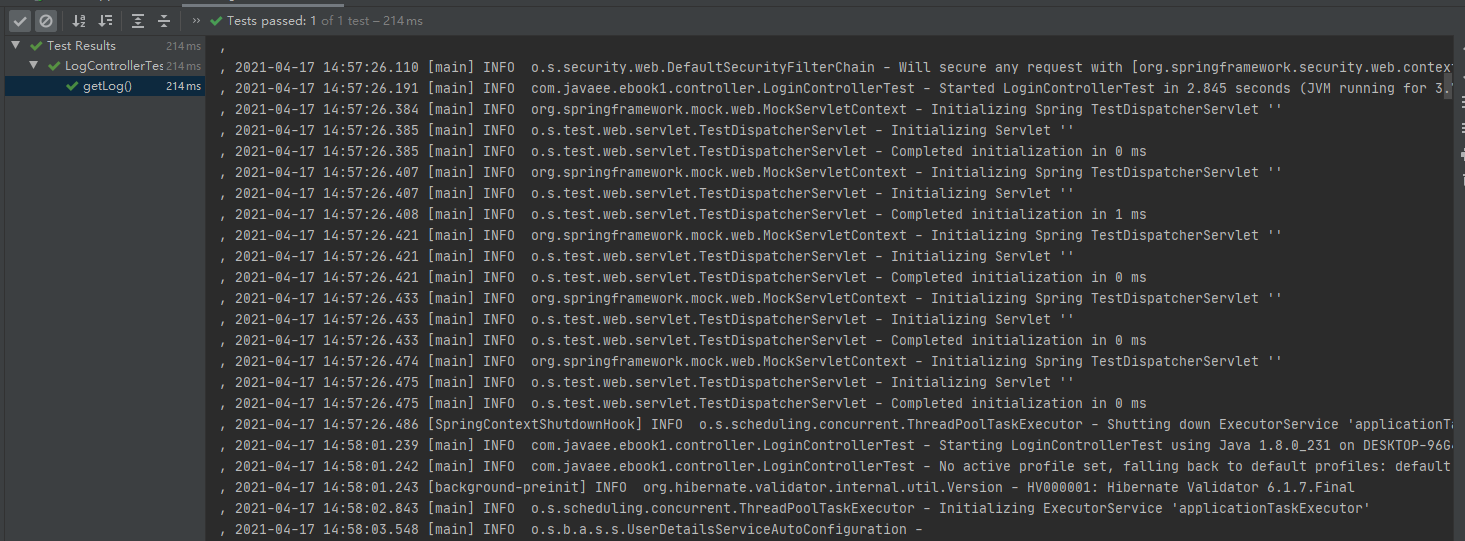
### 5.2.4 FileControllerTest

FileControllerTest中主要涉及到pdf文件和电子书信息的传输，这里因为涉及到http传输的一些问题，将在之后进行完善和补充。

### 5.2.5 LogControllerTest

LogControllerTest主要用于实现获取Log日志文件，通过输入对应的日志类型（包括error,info和warn三个类型）获取对应的日志信息。

在本测试中通过调用controller方法获取对应的日志信息，最终测试结果如下图所示：

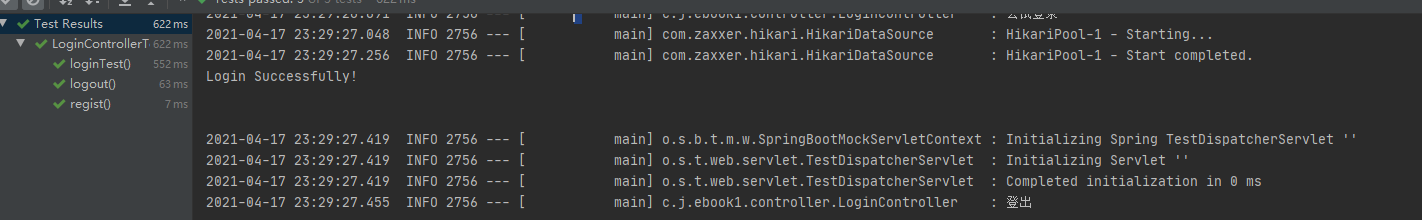


如图所示即为时间范围内的日志信息，返回成功。

### 5.2.6 LoginControllerTest

LoginController中包含三个方法，即需要测试login，regist和logout。

login方法通过本项目已有账号进行测试，最终成功的路；logout则通过消除session中的属性进行退出，因为拦截器的设置，最终会通过重定向返回login界面，而regist则成功于数据库中创建对应账号。LoginControllerTest测试成功。





### 5.2.7 SwitchControllerTest

因为SwitchController本身只涉及界面切换，即url重定向的选择，因此项目中的状态码为302而非200，根据本项目中的拦截设定，最终返回到login下，本测试成功。

