详细设计

网上书店系统

71107302 陈子涵
71107321 蒋泊淼
71107327 罗　皓

目录

1 系统设计

1. 项目概述

1) 背景基本介绍

通过为期三周左右的时间，从需求分析到详细设计的文档阶段，到最终开发和整合结果的代码阶段，来逐步设计和开发一个网上书店系统。

2) 人员任务分配

小组成员：罗皓 71115327，蒋泊淼 71115321，陈子涵 71115302

项目基本分工如下：

任务分配	时间	概要描述	负责人	备注
需求分析和系统设计	12.24	基本需求分析和整个三层 cs 风格的系统架构设计	罗皓，陈子涵	Hwk6.1
系统详细设计和数据库设计	12.25-12.31	对网上书店系统的具体设计和数据库的分析	罗皓，陈子涵，蒋泊淼	Hwk6.2
开发和提交	12.31-1.12	开发阶段的程序设计和提交最终的详细设计	罗皓，陈子涵，蒋泊淼	Hwk6.3

3) 技术路线

计划采用的技术路线如下：

- ✓ Html5 + Css3 前端开发手段
- ✓ JavaScript 交互开发手段
- ✓ JavaEE 底层开发手段
- ✓ SpringMVC 三层视图结构
- ✓ Maven/Gradle 依赖关系
- ✓ SSM 框架（spring, springMVC, Mybatis）
- ✓ Mysql 整合 Mybatis 数据库框架
- ✓ Tomcat 服务器配置
- ✓ 可以考虑的算法：
  - ✓ 搜索引擎推荐算法
  - ✓ 书籍喜好推荐算法

4) 开发运行环境

运行环境分为客户端和服务端。客户端作为用户的角度，所需要的运行环境；服务端作为开发者的角度，提供设计开发者的开发运行环境。

客户端：

- ✓ windows/linux 系统均可

- ✓ 浏览器 建议版本偏高，Chrome/firefox 最佳

服务端：开发规范统一如下

- ✓ windows 10 系统下开发
- ✓ intelliJ 2017.3 版本
- ✓ java9 jdk 版本
- ✓ Tomcat8.5 服务器版本
- ✓ Chrome 浏览器最新版

2. 实施计划

提供项目计划书，设定具体开发规程：

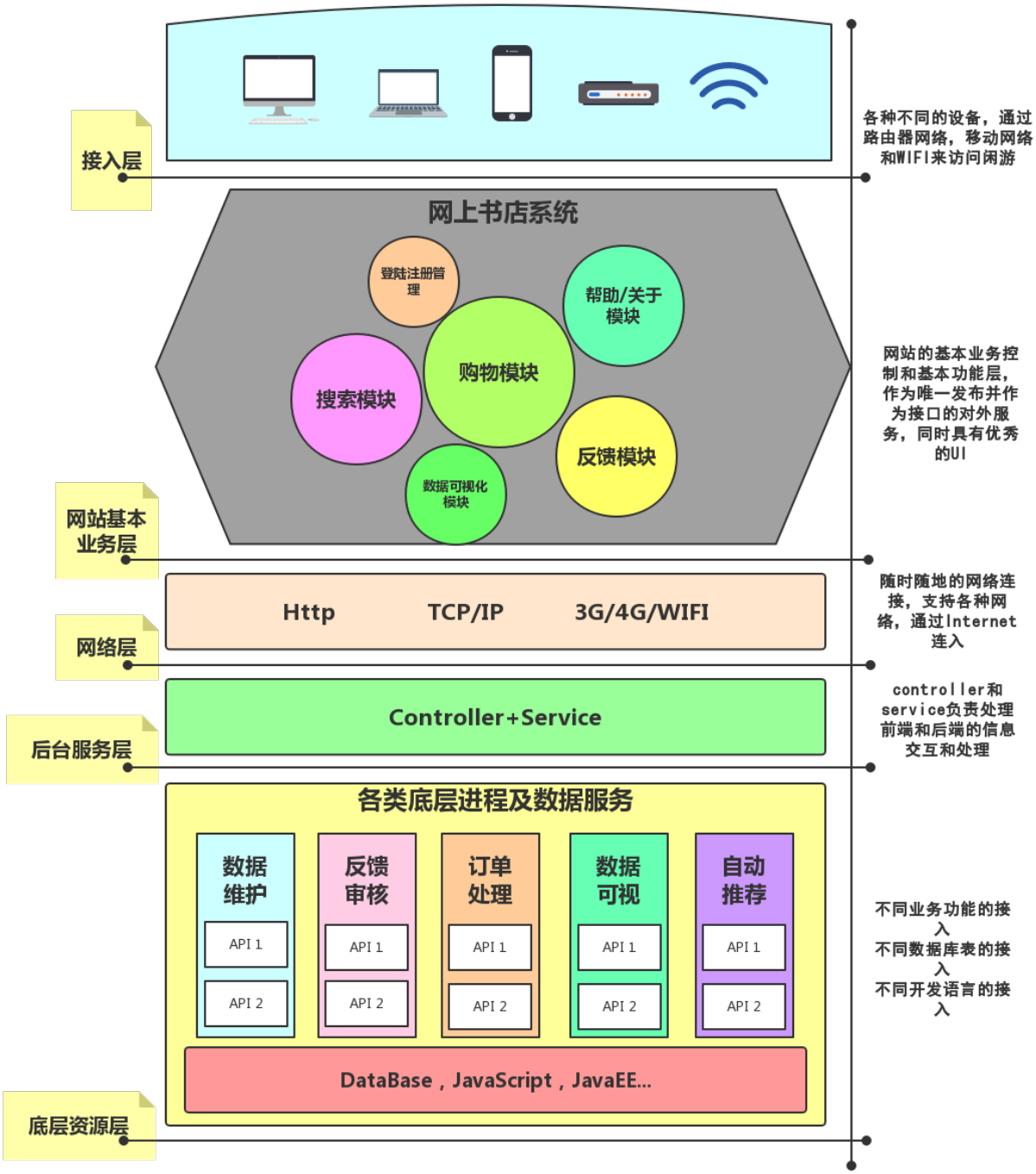
里程碑	deadline 时间	描述	负责人	备注
需求分析和系统设计	12.24	需求和系统	罗皓，陈子涵	文档
详细设计和数据库设计	12.31	功能需求详细设计和确定数据库的关系	罗皓，陈子涵	文档
初步系统架构和 mysql 数据库搭建	1.3	helloworld 和 mysql 数据库	罗皓	初步搭建系统架构，设定层次结构
算法初步设计	1.3	推荐算法	蒋泊淼	算法的简单设计

前端页面和交互设计	1.3	Axure RP 设计初步的简单交互页面	罗皓，陈子涵，蒋泊淼	建立基本的跳转逻辑
设计功能模块	1.9	功能模块之间的具体分工	罗皓，陈子涵，蒋泊淼	逐步完善系统的各个模块
UI 界面的优化	1.12	对 UI 的一些美化	罗皓，陈子涵，蒋泊淼	界面优化

3. 基本系统结构概述

1) 系统架构设计

整个系统采用 javaEE 的 SSM 框架来实现（SSM=Spring+SpringMVC+Mybatis），为系统提供了 MVC 的三层 CS 风格系统架构，我们的系统设计基本是基于这样一套成熟的 javaEE 风格体系上，Spring 和 SpringMVC 为整个网上书店系统提供了 Controll 层的设计和实现，Mybatis 则能够链接数据库 Mysql 通过依赖关系来生成 Model 即数据实体层 javaBean 类体，并且 SpringMVC 提供了 jsp 页面的跳转逻辑设计，利用 html5 设计的方式来展示整个系统的 View 层。



2) 系统优化

对整个系统的优化和改善，期望能够做到的几点如下：

i. 考虑数据可视化——提供数据上的展示

ii. **考虑轻量的设计——Maven/Gradle 包依赖**

iii. **考虑重用的设计——Util 公共模块的设计实现**

iv. **考虑交互的设计——确立一套人性化的交互逻辑**

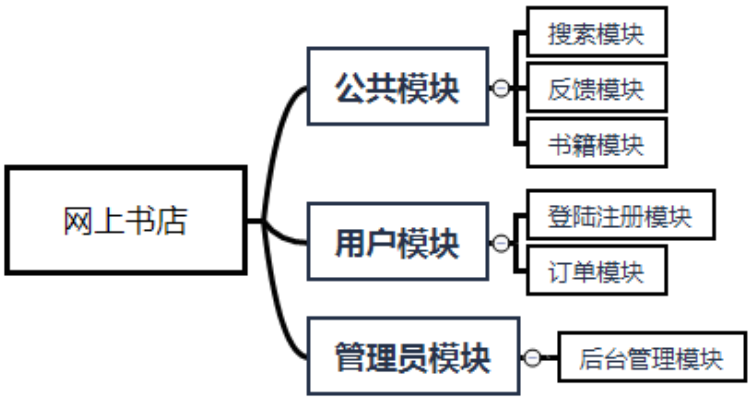
v. **考虑简便的代码逻辑结构——优化代码结构，采取轻便的写法**

4. 版本控制

版本控制：采取分布式服务器系统，将代码和文件托管到 Github 来进行版本控制，并且相关编辑器和 IDE 均有链接 Github 的直接代码提交插件。

2 需求详细设计

由 assignment 中的 Menubar 和 client Functionaliy 设计可知，设计时可以参考 Menubar 的设计来进行模块的划分。总体需求功能模块如下：



模块划分和功能具体需求分析概要如下：

1. 公共模块划分

搜索模块——Search Page

搜索页面：通过导航栏搜索框进入，用户可以通过书名/作者/ISBN 搜索所需图书，该界面将展示通过数据库查询匹配到的图书清单，若没有匹配的图书信息或非法输入，将会提出警告。

抽离功能如下：

- ✓ 提供用户和游客的公共的搜索书籍的功能
- ✓ 键入关键词（作者，标题，ISBN）并且搜索得出书籍结果

反馈模块——About Us/Help/Contact Page

查看帮助页面：用户点击导航栏中的帮助即可进入帮助信息页面，该页面简单介绍该网站并提供三个链接，分别是常见问题及回答、系统规则、联系我们。

查看常见问题及回答页面（F.A.Q.）：用户点击帮助信息页面的常见问题及回答链接进入该页面，该页面列举出了用户的常见问题及对问题的回答帮助用户更好地了解和使用该系统。

系统规则页面：用户点击帮助页面的系统规则的链接进入该页面，该页面列举出用户使用该系统需要遵守的规则。

联系我们页面：用户点击帮助页面的联系我们的链接进入该页面，该页面列出了联系系统员工的 Email 地址，用户可以通过 Email 与网站工作人员联系来解决问题。

抽离功能如下：

- ✓ 关于页面：提供开发人员相关信息
- ✓ 帮助页面：提供网站使用指南
- ✓ 反馈页面：反馈网站 bug 和交互建议给作者

书籍模块——Home/Book Info Page

主页：包括欢迎界面及网站主页，网站主页展示一些推荐图书，用户可以通过导航栏选择登陆/注册，搜索图书，查看图书信息，添加首页展示的图书至购物车等

书籍信息页面：通过点击书籍进入，用户可以在该页面看到特定书籍的 ISBN/书名/作者/价格/是否有存货/剩余本数/预计发货时间（基于剩余本数进行估算）

抽离功能如下：

- ✓ 展示书籍的信息，详细信息等内容
- ✓ 提供部分书籍预览
- ✓ 书籍价格和优惠
- ✓ 书籍数量信息
- ✓ *近期购买情况——数据可视化
- ✓ 加入购物车——用户登录
- ✓ 书籍信息报错——反馈

2. 用户模块划分

用户登录注册模块——Create Account/Login/Checkout

注册页面：用户点击导航栏中的登录/注册即可进入，用户设置的密码需符合系统要求，用户名也应不与他人重复，注册完成后系统将自动登录。

登陆页面：用户点击导航栏中的登录/注册即可进入，通过输入用户名密码登录后身份为会员并跳转至购物车页面，异常登录转至注册，忘记密码则转入找回密码。

找回密码页面：所有用户在忘记密码后可以通过登录页面转至本页面，用户输入用户名，将密码发送至其邮箱，并推荐其修改密码。

抽离功能如下：

- ✓ 用户注册，登录，注销
- ✓ 用户忘记密码，修改密码

订单模块——Shopping Basket

用户点击导航栏中的购物车即可进入购物车页面，展示用户目前还未结账但添加进购物车的购物清单，用户可以在此页面修改所选书籍的数量或删除书籍，更新页面获得最新购物车信息，最终用户将从购物车提交订单。

生成订单页面：已登陆用户提交订单后出现，需要用户填写姓名、收货地址并支付订单，要求用户选择确认或取消，确认则通知用户订单生成成功，更新数据库中订单信息，书籍信息中的数量相应减少；取消则取消生成订单

查看订单页面：已登陆用户可以在导航栏进入订单列表页面对自己的订单进行查看和删除，订单可以链接到相关书籍信息页面：订单编号，订单书籍书名，提交时间，目前状态（待处理已同意/被拒绝交易成功）

抽离功能如下：

- ✓ 加入购物车
- ✓ 购物车内书籍数量增删

3. 管理员模块划分

管理用户信息

抽离功能如下：

- ✓ 对用户信息的增删改查
- ✓ 处理用户提交的订单

管理订单信息

抽离功能如下：

- ✓ 对订单信息的增删改查

管理用户反馈信息

抽离功能如下：

- ✓ 对用户反馈信息的审核

管理书籍页面信息

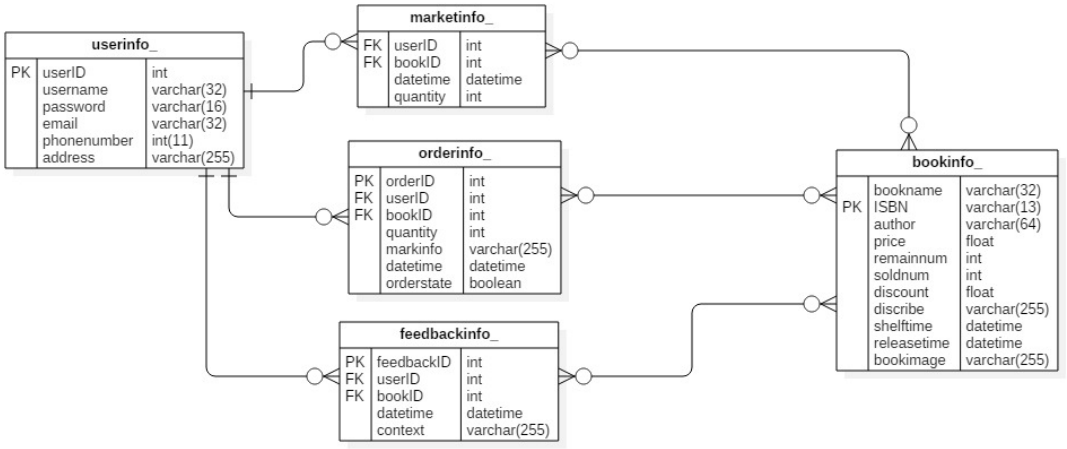
抽离功能如下：

- ✓ 对书籍信息的增删改查

3 数据库详细设计

本系统初步设计数据库为 5 个表，分别为**用户表，购物篮表，订单表，书籍信息表，反馈信息表**，同时因为每个表各自维护不同的信息，各个表之间存在着一对多和多对多的逻辑关系。

1. 数据库整体结构图



2. 数据库详细描述

用户信息表 userinfo_

字段名	类型	备注
UserID	Int	主键 用户 ID

Username	Varchar(32)	用户名
Password	Varchar(16)	密码
Email	Varchar(32)	邮箱
Phonenumber	Int(11)	电话
address	Varchar(255)	地址

购物篮表 marketinfo_

字段名	类型	备注
UserID	Int	用户 ID 外键
ISBN	Varchar(13)	书籍 ISBN 外键
Datetime	Datetime	加入购物车的时间
Quantity	Int	该书籍放入购物车的数量

订单表 orderinfo_

字段名	类型	备注
OrderID	Int	订单 ID 主键
UserID	Int	用户 ID 外键
ISBN	Varchar(13)	书籍 ISBN 外键
Quantity	Int	该书籍的下单的数量
Markinfo	Varchar(255)	订单备注

Datetime	Datetime	订单下单时间
Orderstate	Boolean	订单状态

反馈信息表 feedbackinfo_

字段名	类型	备注
FeedbackID	Int	反馈 ID 主键
UserID	Int	用户 ID 外键
BookID	Int	书籍 ID 外键
Datetime	Datetime	提交反馈时间
Context	Varchar(255)	反馈内容

书籍信息表 bookinfo_

字段名	类型	备注
Bookname	Varchar(32)	书名
ISBN	Varchar(13)	书籍 ISBN 号 主键
Author	Varchar(64)	书籍作者
Price	Float	书籍价格
Discount	Float	书籍折扣
Remainnum	Int	库存
Soldnum	Int	已卖出的书籍数量

Shelftime	**Datetime**	**上架时间**
Releasetime	**Datetime**	**出版时间**
Bookimage	**Varchar(255)**	**书籍图片**
Describe	**Varchar(255)**	**书籍描述**
Bookstate	**Boolean**	**是否上架是否销售**

3. 数据库实体设计

因为基于 SSM 框架设计的网上书店系统，本身 Mybatis 具有一个特性即数据持久化，可以提取 MySQL 等数据库中的数据库表，通过 Mybatis 的逆向工程可以直接转化成.java 的类和.XML 的 Mapper 依赖文件来生成数据实体，即作为 java 中的 JavaBean 体结构。

同时由于 Mybatis 逆向工程的缺陷性，生成的每个表都是独立的，因此在.XML 的 Mapper 依赖文件中添加 MySQL 的外键链接数据库语句，来形成表之间的一对多和多对多关系。

4. 备份计划

技术难点 1：考虑用 windows 实现 MySQL 服务器端对特定的数据库或特定的表定时进行数据库备份**。

5. 并发控制

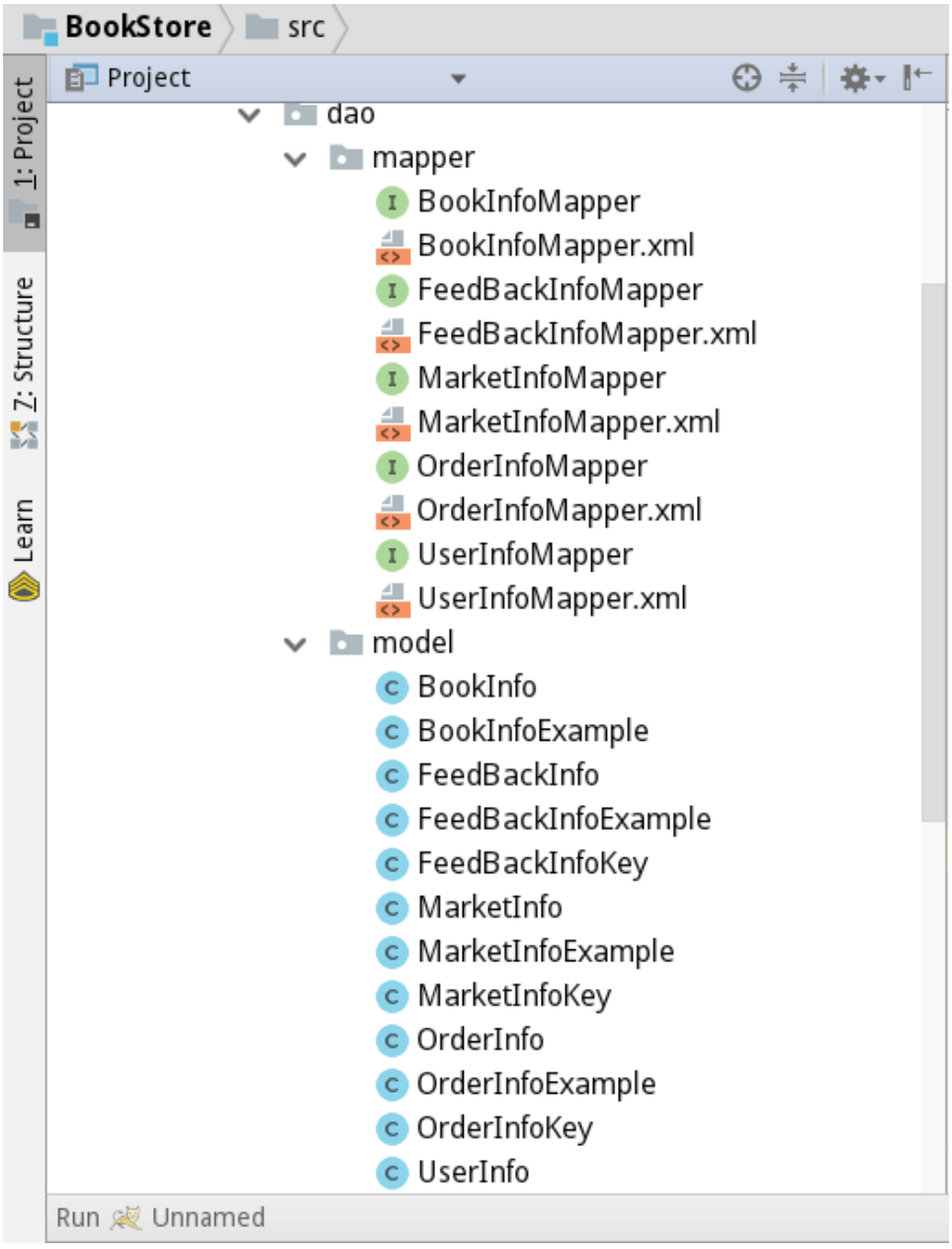
技术难点 2：并发控制是出于**保证数据一致性和多用户并发操作，现考虑利用事务回滚的手段，通过加锁的方式，来实现多用户进行增删改查的数据一致性。**

4 类逻辑设计

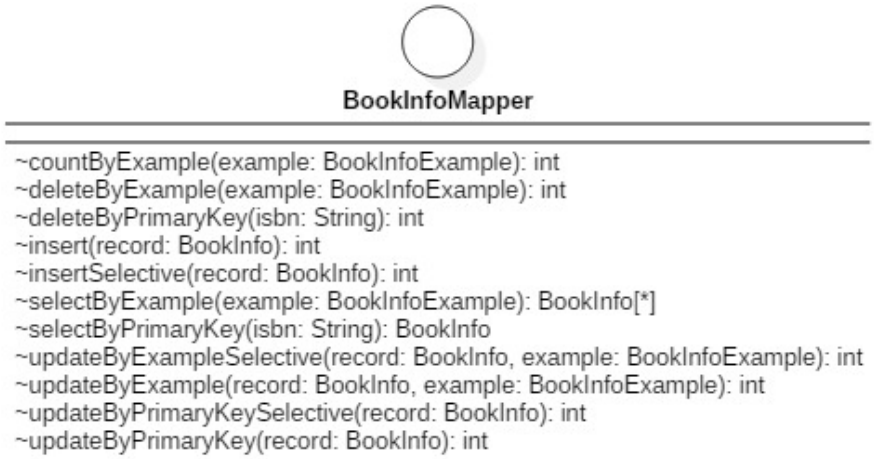
1. 数据访问层

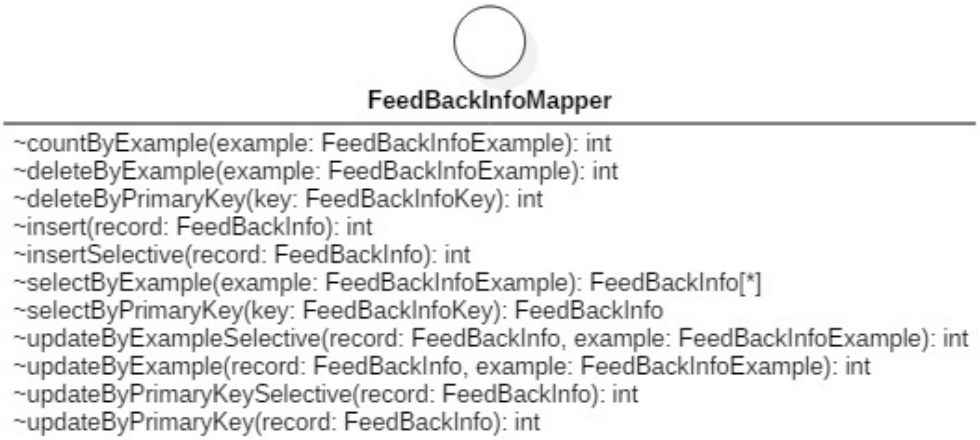
由 Mybatis-generator 逆向工程生成的数据库实体如下，生成的.java 文件如下，均为 JavaBean 体的实体类。

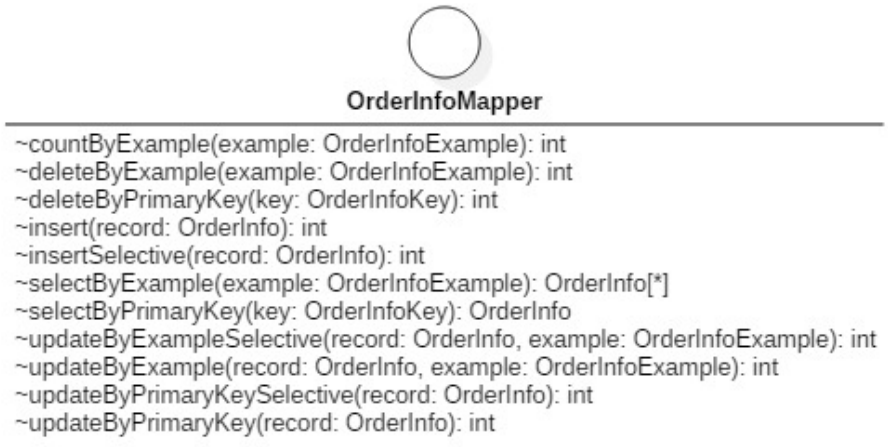
项目中数据访问层目录结构如下：

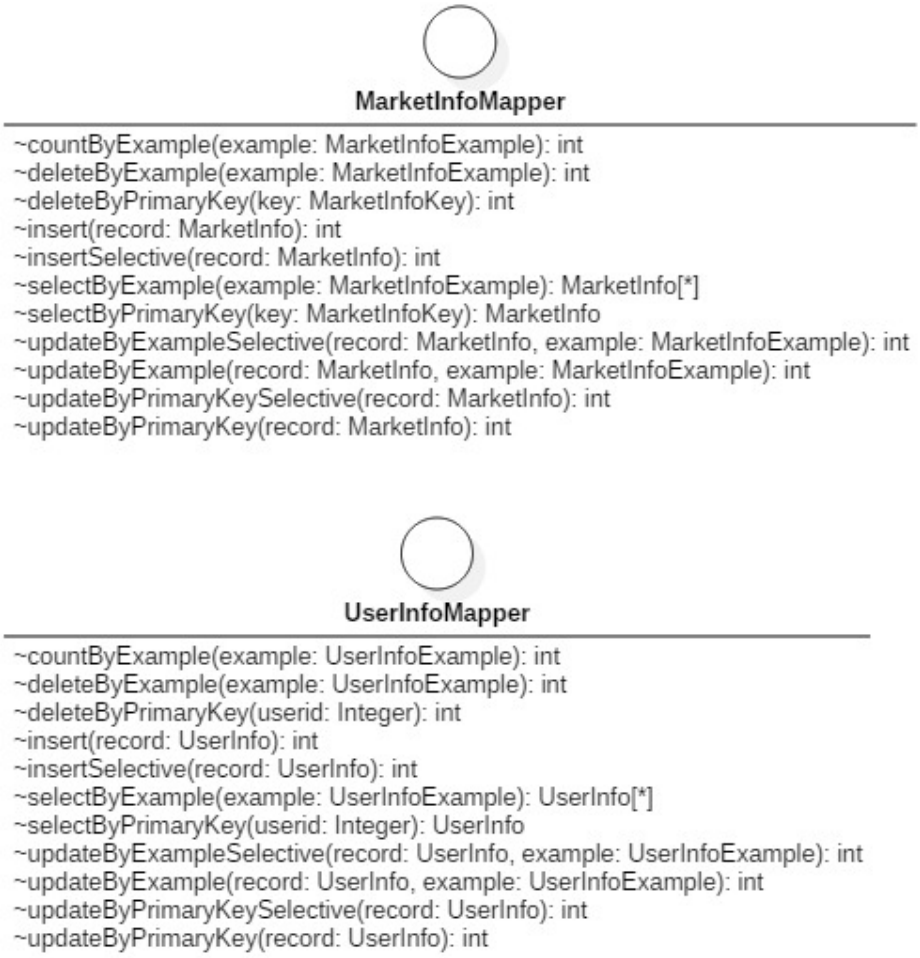


Mapper 层中：Mapper.java** 是作为 java 的接口，**.Mapper.xml** 则是提供了配置项，通过 xml 配置数据库语句来实现接口类的函数功能，提供了基于数据库的增删改查的所有操作函数接口，生成类图总览如下：

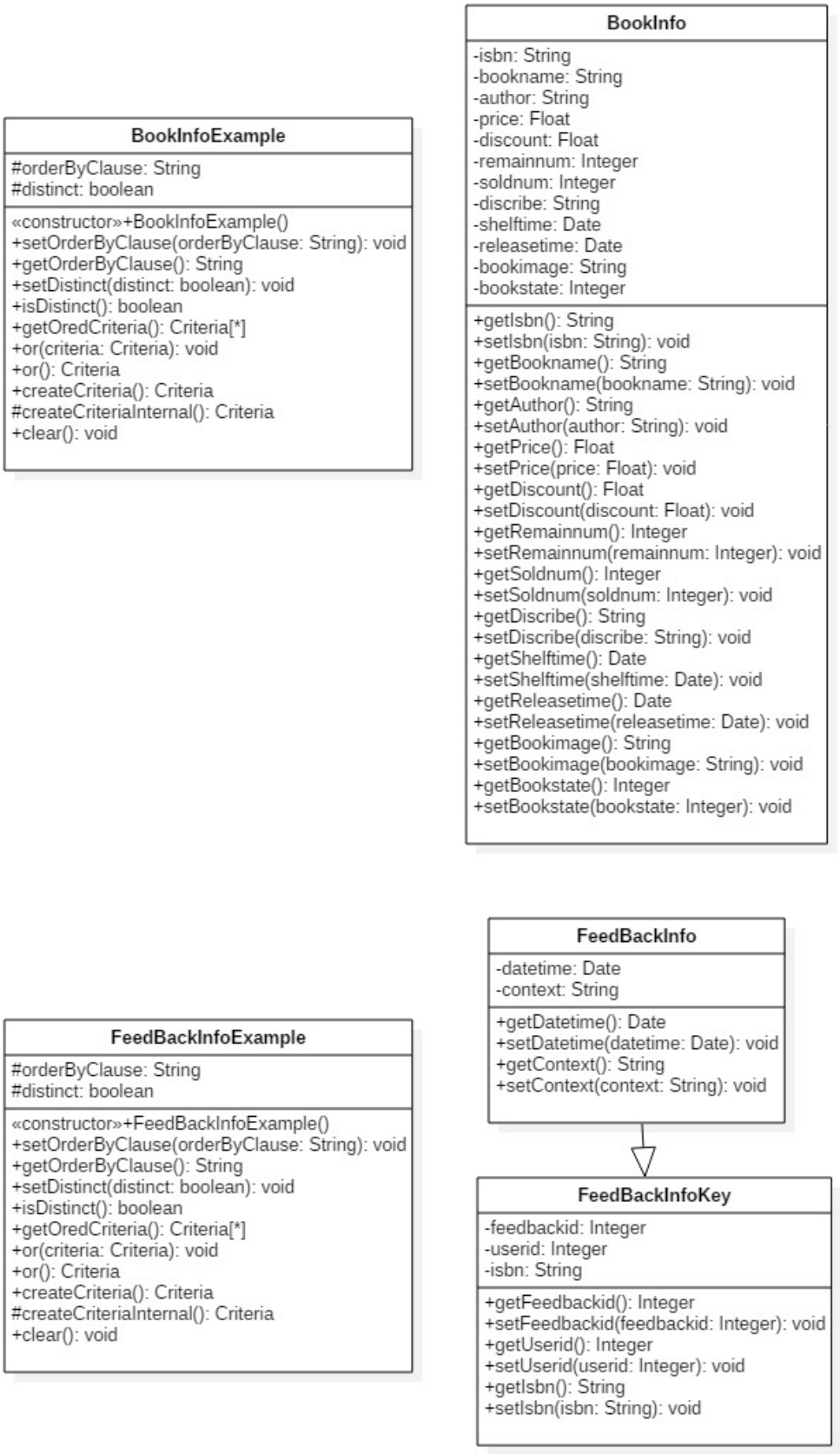


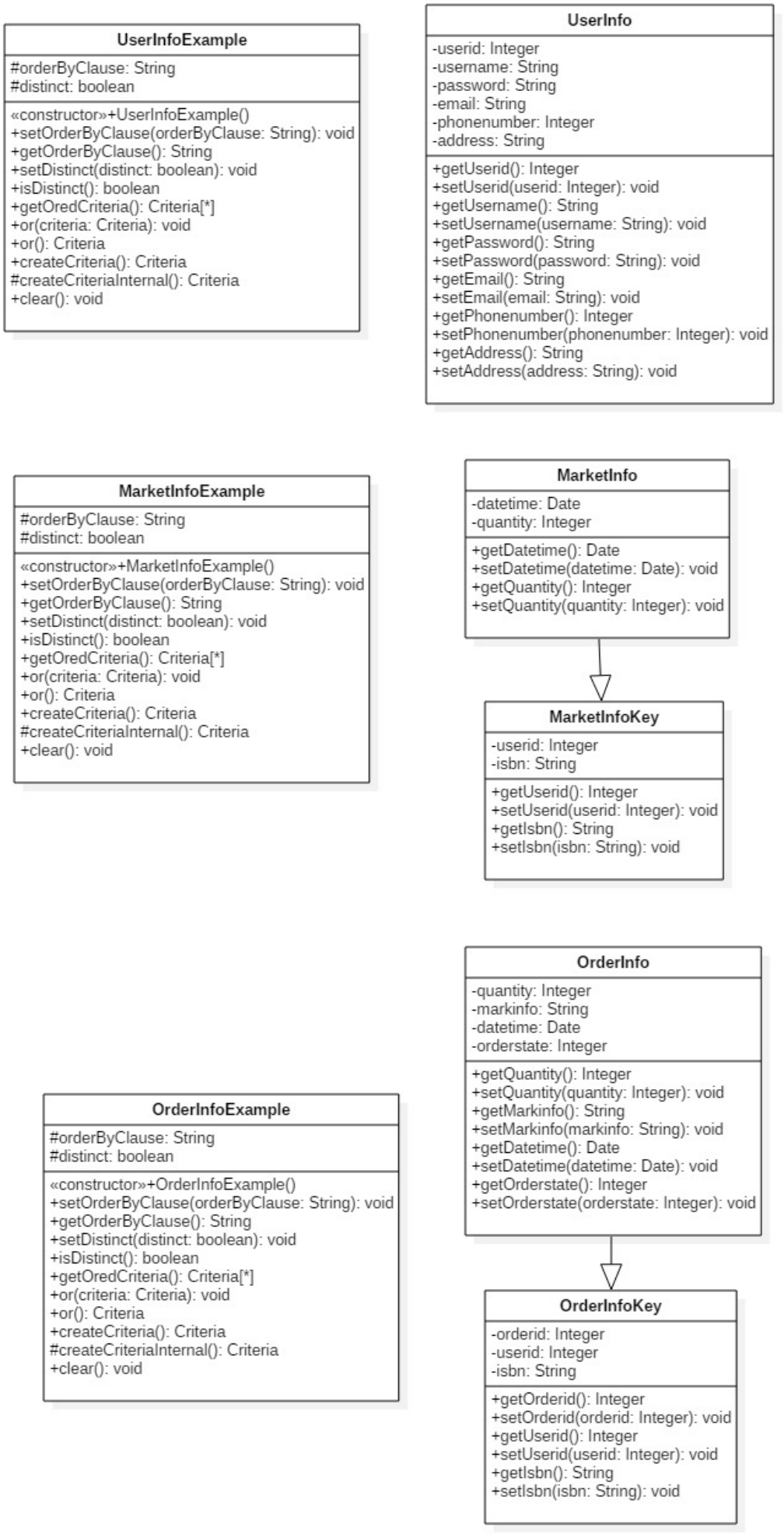






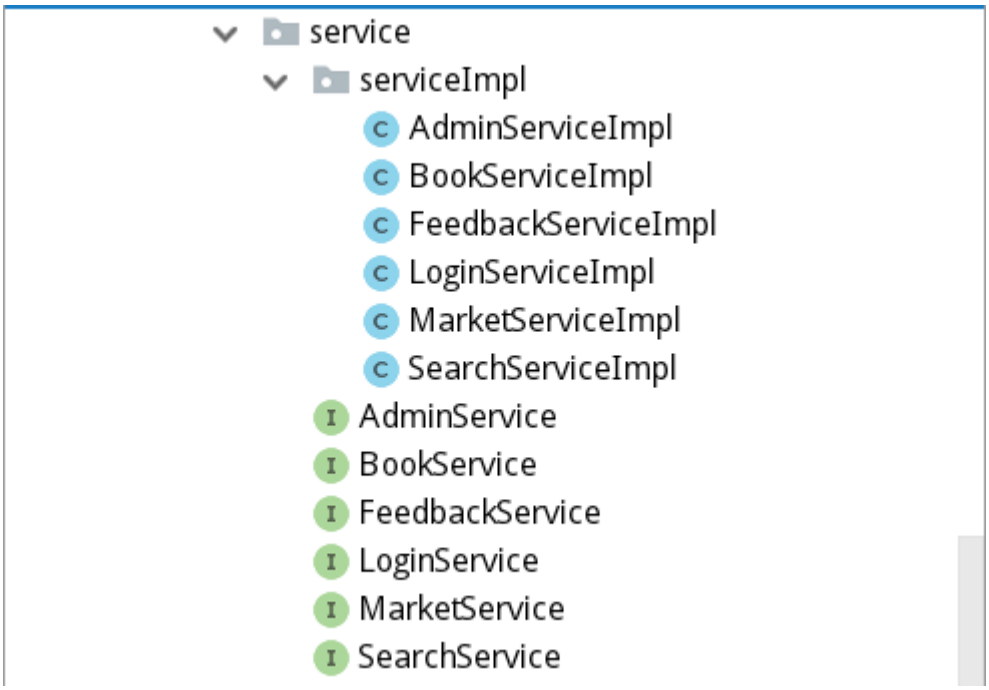
Model 层中：Info.java** 是对应数据库所封装的 JavaBean 实体类，**InfoExample.java 则是提供了查询数据库封装的实体的 JavaBean 的实体的查询示例，并且内部提供 Criteria 类作为查询条件，**InfoKey.java 则提供了 Example 查询主键的 Primary Key 有主键和外键的情况。具体类图如下：


BookInfoExample
#orderByClause: String
#distinct: boolean
«constructor»+BookInfoExample()
+setOrderByClause(orderByClause: String): void
+getOrderByClause(): String
+setDistinct(distinct: boolean): void
+isDistinct(): boolean
+getOredCriteria(): Criteria[*]
+or(criteria: Criteria): void
+or(): Criteria
+createCriteria(): Criteria
#createCriteriaInternal(): Criteria
+clear(): void
BookInfo
-isbn: String
-bookname: String
-author: String
-price: Float
-discount: Float
-remainnum: Integer
-soldnum: Integer
-discribe: String
-shelftime: Date
-releasetime: Date
-bookimage: String
-bookstate: Integer
+getIsbn(): String
+setIsbn(isbn: String): void
+getBookname(): String
+setBookname(bookname: String): void
+getAuthor(): String
+setAuthor(author: String): void
+getPrice(): Float
+setPrice(price: Float): void
+getDiscount(): Float
+setDiscount(discount: Float): void
+getRemainnum(): Integer
+setRemainnum(remainnum: Integer): void
+getSoldnum(): Integer
+setSoldnum(soldnum: Integer): void
+getDiscribe(): String
+setDiscribe(discribe: String): void
+getShelftime(): Date
+setShelftime(shelftime: Date): void
+getReleasetime(): Date
+setReleasetime(releasetime: Date): void
+getBookimage(): String
+setBookimage(bookimage: String): void
+getBookstate(): Integer
+setBookstate(bookstate: Integer): void
FeedBackInfoExample
#orderByClause: String
#distinct: boolean
«constructor»+FeedBackInfoExample()
+setOrderByClause(orderByClause: String): void
+getOrderByClause(): String
+setDistinct(distinct: boolean): void
+isDistinct(): boolean
+getOredCriteria(): Criteria[*]
+or(criteria: Criteria): void
+or(): Criteria
+createCriteria(): Criteria
#createCriteriaInternal(): Criteria
+clear(): void
FeedBackInfo
-datetime: Date
-context: String
+getDatetime(): Date
+setDatetime(datetime: Date): void
+getContext(): String
+setContext(context: String): void
FeedBackInfoKey
-feedbackid: Integer
-userid: Integer
-isbn: String
+getFeedbackid(): Integer
+setFeedbackid(feedbackid: Integer): void
+getUserid(): Integer
+setUserid(userid: Integer): void
+getIsbn(): String
+setIsbn(isbn: String): void


UserInfoExample
#orderByClause: String
#distinct: boolean
«constructor»+UserInfoExample()
+setOrderByClause(orderByClause: String): void
+getOrderByClause(): String
+setDistinct(distinct: boolean): void
+isDistinct(): boolean
+getOredCriteria(): Criteria[*]
+or(criteria: Criteria): void
+or(): Criteria
+createCriteria(): Criteria
#createCriteriaInternal(): Criteria
+clear(): void
UserInfo
-userid: Integer
-username: String
-password: String
-email: String
-phonenumber: Integer
-address: String
+getUserid(): Integer
+setUserid(userid: Integer): void
+getUsername(): String
+setUsername(username: String): void
+getPassword(): String
+setPassword(password: String): void
+getEmail(): String
+setEmail(email: String): void
+getPhonenumber(): Integer
+setPhonenumber(phonenumber: Integer): void
+getAddress(): String
+setAddress(address: String): void
MarketInfoExample
#orderByClause: String
#distinct: boolean
«constructor»+MarketInfoExample()
+setOrderByClause(orderByClause: String): void
+getOrderByClause(): String
+setDistinct(distinct: boolean): void
+isDistinct(): boolean
+getOredCriteria(): Criteria[*]
+or(criteria: Criteria): void
+or(): Criteria
+createCriteria(): Criteria
#createCriteriaInternal(): Criteria
+clear(): void
MarketInfo
-datetime: Date
-quantity: Integer
+getDatetime(): Date
+setDatetime(datetime: Date): void
+getQuantity(): Integer
+setQuantity(quantity: Integer): void
MarketInfoKey
-userid: Integer
-isbn: String
+getUserid(): Integer
+setUserid(userid: Integer): void
+getIsbn(): String
+setIsbn(isbn: String): void
OrderInfo
-quantity: Integer
-markinfo: String
-datetime: Date
-orderstate: Integer
+getQuantity(): Integer
+setQuantity(quantity: Integer): void
+getMarkinfo(): String
+setMarkinfo(markinfo: String): void
+getDatetime(): Date
+setDatetime(datetime: Date): void
+getOrderstate(): Integer
+setOrderstate(orderstate: Integer): void
OrderInfoExample
#orderByClause: String
#distinct: boolean
«constructor»+OrderInfoExample()
+setOrderByClause(orderByClause: String): void
+getOrderByClause(): String
+setDistinct(distinct: boolean): void
+isDistinct(): boolean
+getOredCriteria(): Criteria[*]
+or(criteria: Criteria): void
+or(): Criteria
+createCriteria(): Criteria
#createCriteriaInternal(): Criteria
+clear(): void
OrderInfoKey
-orderid: Integer
-userid: Integer
-isbn: String
+getOrderid(): Integer
+setOrderid(orderid: Integer): void
+getUserid(): Integer
+setUserid(userid: Integer): void
+getIsbn(): String
+setIsbn(isbn: String): void

2. 业务逻辑层

通过设计 Service 层来为整个网站提供中间业务逻辑层，实现数据之间的转接和基本业务功能的交接，按照模块分布设计相应接口和接口实现来提供业务逻辑层的功能。具体目录如下：



初步拟定为如下业务逻辑层的相关服务，提供以下功能：

业务名称	**业务功能**
AdminService	提供管理员的相关业务功能，包含管理员模块的所有功能
BookService	提供书籍方面的相关业务功能，包含增删改查书籍信息的功能
FeedbackService	提供反馈的相关业务功能，包含了反馈信息提交查询等功能
LoginService	提供与用户相关的业务功能，包含用

	户注册，登陆和注销等功能的基本设计实现
MarketService	提供购物篮的相关业务功能，包含加入购物篮和修改购物篮内信息等功能
SearchService	提供简单的搜索业务功能，在底层考虑算法的同时，也包含了数据的相关的搜索和查询功能

类的相关设计：

出于敏捷开发的需要和考虑，我们在编码之前，暂时不设计所有的增删改查的接口函数实现，而是在考虑具体实现过程中需要用到的功能，考虑不同的情景，再进行相关代码的设计和编码。

5 界面设计

暂无界面设计，具体见 final_report。