1.前台用户操作

1.1登录与注销（LoginController控制类）

1.1.1登录

1.1.1.1前端html实现

用户可以在login.html中登录，利用jQuery Validate前端验证输入框表单中用户输入的用户名和密码是否符合规范。点击提交，使用ajax技术向后端发送信息。利用serialize()方法将表单内容序列化为json数据后通过post方法提交至/login。若成功，接受后端的Bean。bean信息包括用来指示请求是否正确执行的code属性和存储信息的msg属性。若code属性的值为1，则代表后端正确执行，在这里提示用户登录成功，并跳转至网上书店首页；若code属性的值不为1，则代表未正确执行，此时浏览器将错误信息提示给用户。

1.1.1.2后端Java实现（login方法）

接收前端通过post方法提交至/login的用户名和密码，再将用户名和密码传递至Service层中loginService中的login方法。在Service层中，首先判断用户名或密码是否为空，不为空则将用户名传递至Dao层，通过与对数据库的查询获得用户信息。若无法在数据库中查找到相关信息，则提示用户名不存在。接着对获得的信息进行判断，判断密码是否有误及数据库中is\_locked字段是否为1（账户被锁定），抛出相应的异常。

同时，利用cookie简单记录用户密码错误次数。若用户输入密码错误，则将cookie中的次数+1，当超过3时，调用Service层的lock方法自动锁定用户。

Service层中的判断没有错误，则将用户名存放到session对象及cookie中，为后续操作进行铺垫。同时清空cookie中的密码重试次数信息。

1.1.2注册

1.1.2.1前端html实现

用户在register.html中提交用户名、密码和邮箱进行注册。当用户名输入框失去焦点时触发check方法，将用户名通过get方法提交至后台判断是否已经存在。同时，利用jQuery Validate验证用户填写的是否合法。点击立即注册，数据验证无误后将表单中的值序列化后通过post方法提交至/register。若正确执行，跳转至登录界面

1.1.2.2后端Java实现（register方法及check方法）

注册：接收前端通过post方法提交至/register的表单项并自动创建用户User对象。将此对象传入Service层的register方法，判断用户对象是否为空，及两次密码输入是否一致，通过后将用户对象传至Dao层，在数据库中创建一条新的用户记录。

检查用户名：接受前端通过get方法提交至/check的用户名字符串。将此对象传入Service层的userIsExist方法，判断是否已经存在以此名称命名的用户。

1.1.3注销

1.1.3.1前端html实现（logout方法）（JavaScript）

用户在任意界面点击注销按钮后触发logout方法，将get请求提交至/logout，通过后端进行注销操作。若正确执行，提示用户注销成功并跳转至登录页面；若code为-1，则表示用户未登录，提示用户“请先登录”并同样跳转至登录页面。若未正确执行，则通知用户。

1.1.3.2后端Java实现（logout方法）

获取登录后在session对象中存放的用户名，将其传入Service层中的userIsLogin方法，通过对字符串的合法性判断来判断用户是否已正确登录。若正确登录，则删除session中存放的用户名，并清空cookie中的存储；若未登录，则将bean的code属性设置为-1，供前端判断。

1.2图书展示（BookController控制类）

1.2.1分页展示图书（前台面向用户展示）

PageBean类用来存放当前的分页信息，包括当前页、总页数、每页显示的记录数（默认为4）、总记录数及查询到的分页数据（pageInfos）。pageInfos利用泛型可以存储任意类型的数据。

1.2.1.1前端html实现

用户可以在index.html查看所有图书的详细信息，包括书名、价格、库存、图片。页面加载后默认从第一页加载，将当前页码和前后台标识通过get方法发送至后端/books/page，得到当前页的图书信息（PageBean）并动态显示。

1.2.1.2后端Java实现（findByPage方法）

后端接收通过通过get方式发送至/books/page/{当前页}的数据，包括当前页码和前后台标识，将其传入Service层findByPage方法。在findByPage方法中，首先新建一个空的PageBean对象，将当前页记录在其中，再通过Dao层获取图书总数并记录。接着利用图书总数和每页的记录数计算总页数。最后将由当前页和每页的记录数组成的Map以及前后台标识传入Dao层，利用”WHERE state=0“子句限制只查询未删除的图书，再用LIMIT关键字（MySQL限定）从数据库中获取当前页的图书信息存放在List<Book>对象中。若读取成功，则将其存放在pageInfos中。最终，将PageBean对象存放在JsonBean中发送至前端。

1.3购物车（CartController控制类）

本系统的购物车信息存储在本地cookie中。

1.3.1添加购物车

1.3.1.1前端html实现

在index.html页面中选中图书后点击添加购物车，选中的图书信息会序列化后通过post方法发送至/carts。执行成功，跳转至我的订单。

1.3.1.2后端Java实现（addCart方法）

根据RESTful风格，addCart方法用来向购物车添加数据，因此使用post方法接收前端请求。在这里，接收选中的图书id数组，将其和cookie中已有的购物车信息传入Service层的addCart方法。在addCart方法中，首先判断用户是否选中图书，接着过滤掉cookie中已有的图书信息，并返回新的cookie。Control层将新的cookie覆盖。

1.3.2展示购物车

1.3.2.1前端html实现

在shoppng.html可以查看当前的购物车。进入该页面后通过get方法向后端/carts请求购物车中的图书列表并动态显示在当前页面，若购物车为空，则自动跳转至图书展示页面。利用jQuery，当某一图书的数量改变时，触发change事件，判断图书库存是否充足、输入的数量是否是正整数。

1.3.2.2后端Java实现（carts方法）

根据RESTful风格，addCart方法用来查询购物车数据，因此使用get方法接收前端请求。将cookie中的购物车信息传入Service层中的findCartInfo方法获取购物车中的图书列表。在findCartInfo方法中，读取cookie，并将其格式化后存入List<String>中，再传入Dao层，获取这些图书id对应的图书信息列表List<Book>并返回。Control层将获取到的图书列表存放在JsonBean中发送至前端。

1.3.3更新购物车（删除条目）

1.3.3.1前端html实现

在shopping.html中，点击购物车的某条记录旁的删除按钮可以将之删除。此时系统获取购物车中的所有图书id，并删除选中的记录的图书id。完成后，将这些id拼接成json格式并通过put方法发送至后端/carts。完成请求后重新载入购物车。http默认不支持put请求，需要修改web.xml使其支持。修改后，设置发送的数据\_method属性为相应的发送方法，则前端在向后端发送数据时会使用相应的方法。【【【图】】】

1.3.3.2后端Java实现（updateCarts方法）

根据RESTful风格，updateCarts方法用来更新购物车数据，因此使用put方法接收前端请求。将接收到的图书id数组发送至Service层的updateCart方法，使其转换为cookie格式。将Service层的返回值直接覆盖购物车的cookie值，以更新购物车的方式完成购物车条目的删除。

1.4我的订单（OrderController控制类）

在订单实体类Order和订单详情实体类OrderItem中，将相应的对象作为私有属性以表示外键关系。例如，在OrderItem中，私有属性order和book这两个对象用来表示本条订单详情所对应的订单对象和图书对象。

1.4.1添加订单

1.4.1.1前端html实现

在shopping.html中，确定购买项目后点击提交按钮即可提交订单。点击按钮后前台首先读取cookie判断用户是否登录，然后将购物车中的图书信息和总金额以post方法提交给后端/orders。若操作成功，则跳转至订单详情页面；若用户未登录，则提示用户登录后添加订单，并跳转至登录页面。

1.4.1.2后端Java实现（addOrder方法）

后端通过post方法接收前台发送的图书id数组、对应的图书数量数组和总价。首先后去session中存放的用户名，将用户名传入Service层的userIsLogin方法判断用户是否登录。若用户已登录，则将图书id数组、对应的图书数量数组、总价、用户名传入Service层的addOrderInfo方法。

在addOrderInfo方法中，首先将传入的总价和当前时间加入新创建的Order对象中，同时生成唯一的UUID作为订单号加入对象中，并将订单状态设置为0（未发货）。将用户名传入Dao层获得当前用户对象，并将其存入订单对象中以表示外键关联。接着，将订单对象传入Dao层，插入数据库的订单表。接着，调用Service层中的addOrderItems方法，将订单中的每一条信息循环插入数据库的订单详情表。在插入图书数量的同时判断库存是否充足。xml文件中，插入订单详情的同时更新图书表的库存数量。【【【图】】】

添加成功后返回Control层，用空的数据覆盖cookie中的购物车，即完成清空购物车的操作。

1.4.2分页展示订单

PageBean类用来存放当前的分页信息，包括当前页、总页数、每页显示的记录数（默认为4）、总记录数及查询到的分页数据（pageInfos）。pageInfos利用泛型可以存储任意类型的数据。

1.4.2.1前端html实现

用户可以在orderlist.html查看所有订单的详细信息，包括订单编号、用户名、总价、状态（待发货、待收货、已完成、退货中、已退货），以及订单中每个商品的书名、价格、数量、图片。页面加载后默认从第一页加载，将当前页码和前后台标识通过get方法发送至后端/books/page，得到当前页的图书信息（PageBean）动态展示。

1.4.2.2后端Java实现（findOrderInfo方法）

后端接收通过通过get方式发送至/orders/page/{当前页}的数据，包括当前页码和前后台标识，将这些数据和session中存储的当前用户的用户名传入Service层findItemByIndex方法。在findItemByIndex方法中，首先新建一个空PageBean对象，将当前页记录在其中，再通过Dao层获取当前用户的订单总数并记录。接着利用订单总数和每页的记录数计算总页数。最后将由当前页、每页的记录数和用户名组成的Map对象传入Dao层，利用INNER JION关键字将订单详情表、订单表、图书表和用户表进行内连接，并用LIMIT关键字（MySQL限定）从数据库中获取当前页的全部订单信息存放在List<OrderItem>对象中。【【【图】】】若读取成功，则将其存放在pageInfos中。最终，将PageBean对象存放在JsonBean中发送至前端。

1.4.3改变订单状态（前台用户操作）

1.4.3.1前端html实现

在orderlist.html中，用户可以操作状态为1（待收货）的订单进行确认收货，以及状态为2（已完成）的订单进行申请退货。点击按钮，网页通过put方法向后端/orders发送一个包含当前订的单编号和状态请求。若操作成功，则重新读取当前页码的订单数据并动态展示。

1.4.3.1后端Java实现（updateState方法）

后端通过put方法接收前端传来的订单对象。将订单对象直接传入Service层的changeState方法，通过Dao层更新数据库中的订单状态。

2.后台管理员操作

2.1用户管理（UserAdminController控制类）

2.1.1展示用户

2.1.2查找用户

2.1.3.用户锁定与解锁

2.1.4删除用户

2.2图书管理（BookAdminController控制类）

2.2.1分页展示图书（后台面向管理员展示）

PageBean类用来存放当前的分页信息，包括当前页、总页数、每页显示的记录数（默认为4）、总记录数及查询到的分页数据（pageInfos）。pageInfos利用泛型可以存储任意类型的数据。

1.2.1.1前端html实现

管理员在books.html可以查看所有图书（包括已删除的图书）的详细信息，包括书名、价格、库存、图片。页面加载后默认从第一页加载，将当前页码和前后台标识通过get方法发送至后端/books/page，得到当前页的图书信息（PageBean）并动态显示。

1.2.1.2后端Java实现（findByPage方法）

后端接收通过通过get方式发送至/books/page/{当前页}的数据，包括当前页码和前后台标识，将其传入Service层findByPage方法。在findByPage方法中，首先新建一个空的PageBean对象，将当前页记录在其中，再通过Dao层获取图书总数并记录。接着利用图书总数和每页的记录数计算总页数。最后将由当前页和每页的记录数组成的Map以及前后台标识传入Dao层，利用LIMIT关键字（MySQL限定）从数据库中获取当前页的图书信息存放在List<Book>对象中。若读取成功，则将其存放在pageInfos中。最终，将PageBean对象存放在JsonBean中发送至前端。

2.2.2.查找图书

2.2.3.添加图书

通过IDE对Tomca的配置，将工程文件夹中的upload文件夹映射为/image，实现了对新上传的图书图片的存储。由于存储上传图片的文件夹在工程文件夹内，在上传图片的时候直接通过相对路径对绝对路径的映射获取本机的upload文件夹的绝对位置，将文件上传后的位置设定为映射得到的绝对路径，避免了改变项目环境后绝对路径的不匹配问题。

2.2.3.1前端html实现

管理员在add\_book.html中可以通过提交书名、价格、库存及图片完成图书的添加。利用jQuery Validate判断表单项的合法性，利用jQuery Form异步上传图书图片。在点击添加图书后调用ajaxSubmit方法向后端/ upload.do发送一个post方法的请求，以及序列化后的表单数据和MultipartFile格式的图片文件。若操作成功，页面进行跳转。

2.2.3.2后端Java实现（add方法）

后端接收由post方法传来的MultipartFile类图片文件和图书对象。首先获得利用上传文件夹的绝对路径和图片的文件名创建一个新的File对象，利用Spring MVC中MultipartFile类的transferTo方法将图片文件传输至指定的文件夹（upload文件夹），完成后将图书对象中的img属性设置为图片的相对位置。最后将完整的图书对象传入Service层的add方法添加至数据库。

2.2.4.修改图书

2.2.4.1前端html实现

管理员在update\_book.html中修改相应的图书信息。与上述添加图书大致相同不再赘述。

2.2.4.2后端Java实现（update方法）

后端接收参数时，图片文件是不必要的，即前端可以不发送新的图片从而不修改原有的图片信息。后端的其他处理方式与上述添加图书部分相同，不再赘述。

2.2.5.删除图书（假删除）

2.3订单管理（OrderAdminController控制类）

2.3.1.分页展示订单

2.3.2.查找订单

2.3.3.改变订单状态（后台管理员操作）

2.3.4.删除订单