《数据库系统原理》大作业

系统实现报告

题目名称：TopLaptop 笔记本销售商城

学号及姓名：20231111 李圣千

20377133 赵静晗

20231035 杨瑗珺

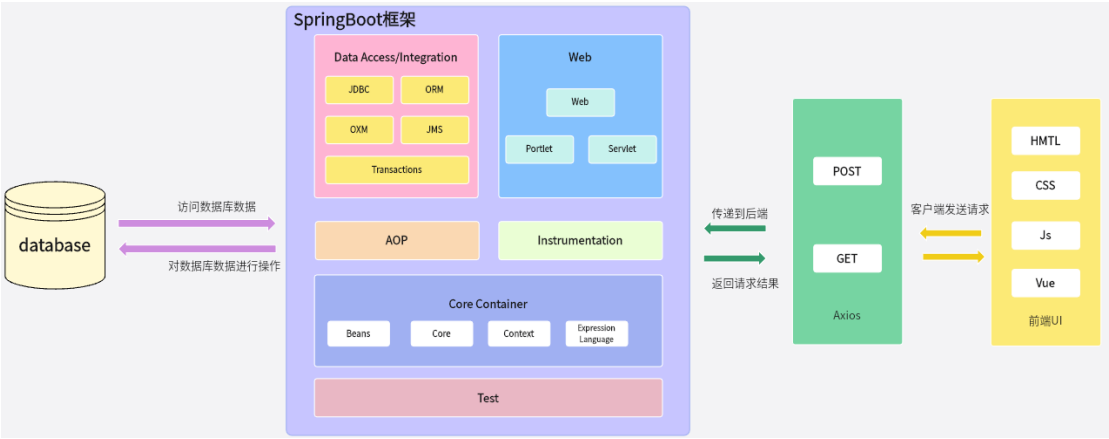
2022 年 10 月 15 日

一. 系统结构设计

1.1 体系结构

本系统采用前后端分离的体系结构。数据库对数据进行存储和管理；后端连接数据库并将数据传送给前端，同时接收前端请求并传递给数据库；前端与用户进行交互，显示数据库数据并接收用户需求。

本系统的体系结构如图所示 ：



1.2 环境与框架介绍

1. 前端使用 HTML+CSS+JavaScipt 的技术

(1)HTML：

HTML 的全称为超文本标记语言，是一种标记语言。它包括一系列标签．通过这些标签可以将网络上的文档格式统一，使分散的 Internet 资源连接为一个逻辑整体。HTML 文本是由 HTML 命令组成的描述性文本，HTML 命令可以说明文字，图形、动画、声音、表格、链接等。它是一种组织信息的方式，它通过超级链接方法将文本中的文字、图表与其他信息媒体相关联。这些相互关联的信息媒体可能在同一文本中，也可能是其他文件，或是地理位置相距遥远的某台计算机上的文件。这种组织信息方式将分布在不同位置的信息资源用随机方式进行连接，为人们查找，检索信息提供方便。

(2)CSS：

层叠样式表是一种用来表现 HTML（标准通用标记语言的一个应用）或 XML（标准通用标记语言的一个子集）等文件样式的计算机语言。CSS 不仅可以静态地修饰网页，还可以配合各种脚本语言动态地对网页各元素进行格式化。CSS 能够对网页中元素位置的排版进行像素级精确控制，支持几乎所有的字体字号样式，拥有对网页对象和模型样式编辑的能力。

(3)JavaScript:

JavaScript 是一种具有函数优先的轻量级，解释型或即时编译型的编程语言。虽然它是作为开发 Web 页面的脚本语言而出名，但是它也被用到了很多非浏览器环境中，JavaScript 基于原型编程、多范式的动态脚本语言，并且支持面向对象、命令式、声明式、函数式编程范式。JavaScript 的标准是 ECMAScript 。截至 2012 年，所有浏览器都完整的支持 ECMAScript 5.1，旧版本的浏览器至少支持 ECMAScript 3 标准。2015 年 6 月 17 日，ECMA 国际组织发布了 ECMAScript 的第六版，该版本正式名称为 ECMAScript 2015，但通常被称为 ECMAScript 6 或者 ES2015。

(4)Vue:

Vue 是一套用于构建用户界面的渐进式 JavaScript 框架。与其它大型框架不同的是，Vue 被设计为可以自底向上逐层应用。Vue 的核心库只关注视图层，不仅易于上手，还便于与第三方库或既有项目整合。另一方面，当与现代化的工具链以及各种支持类库结合使用时，Vue 也完全能够为复杂的单页应用（SPA）提供驱动。

2. 前后端交互采用 Axios

Axios 是一个基于 promise 的 HTTP 库，可以发送 get、post 请求两种方式向后端发出数据请求和传递参数。

3. 后端

(1)使用 Java 作为开发语言：

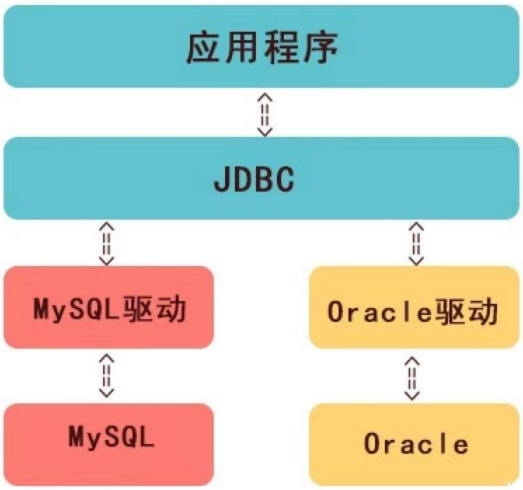
Java 是一门计算机编程语言，应用广泛，从我们日常用的安卓手机 APP 到大部分网站或管理信息系统的应用服务器程序都是用 Java 这中语言来写的。Java 是纯面向对象开发，功能强大，分支众多，没有 Java 不能做的软件；Java 通过 JDBC 来访问数据库，通过不同的数据库厂商提供的数据库驱动方便地访问数据库。访问数据库的接口比较统一

(2)使用 SpringBoot 框架进行后端开发:

SpringBoot 是由 Pivotal 团队提供的全新框架，其设计目的是用来简化新 Spring 应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。

4. 后端与数据库的交互

采用 JDBC 与数据库进行交互，在 Mapper 层操纵 sql 语句完成 crud 操作：JDBC 的全称是 Java 数据库连接(Java Database connect)，它是一套用于执行 SQL 语句的 Java API。应用程序可通过这套 API 连接到关系数据库，并使用 SQL 语句来完成对数据库中数据的查询、更新和删除等操作。应用程序使用 JDBC 访问数据库的方式如下图所示：



5. 数据库：采用 MySQL 数据库

MySQL 在过去由于性能高、成本低、可靠性好，已经成为最流行的开源数据库，因此被广泛地应用在 Internet 上的中小型网站中。随着 MySQL 的不断成熟，它也逐渐用于更多大规模网站和应用，比如维基百科、Google 和 Facebook 等网站。MySQL 是一种关系型数据库管理系统，关系数据库将数据保存在不同的表中，而不是将所有数据放在一个大仓库内，这样就增加了速度并提高了灵活性。MySQL 所使用的 SQL 语言是用于访问数据库的最常用标准化语言。

6. web server :tomcat

Tomcat 是一个免费的 Web 应用服务器，服务器其实就是代码编写的一个可以根据用户请求实时的调用执行对应的逻辑代码的一个容器。

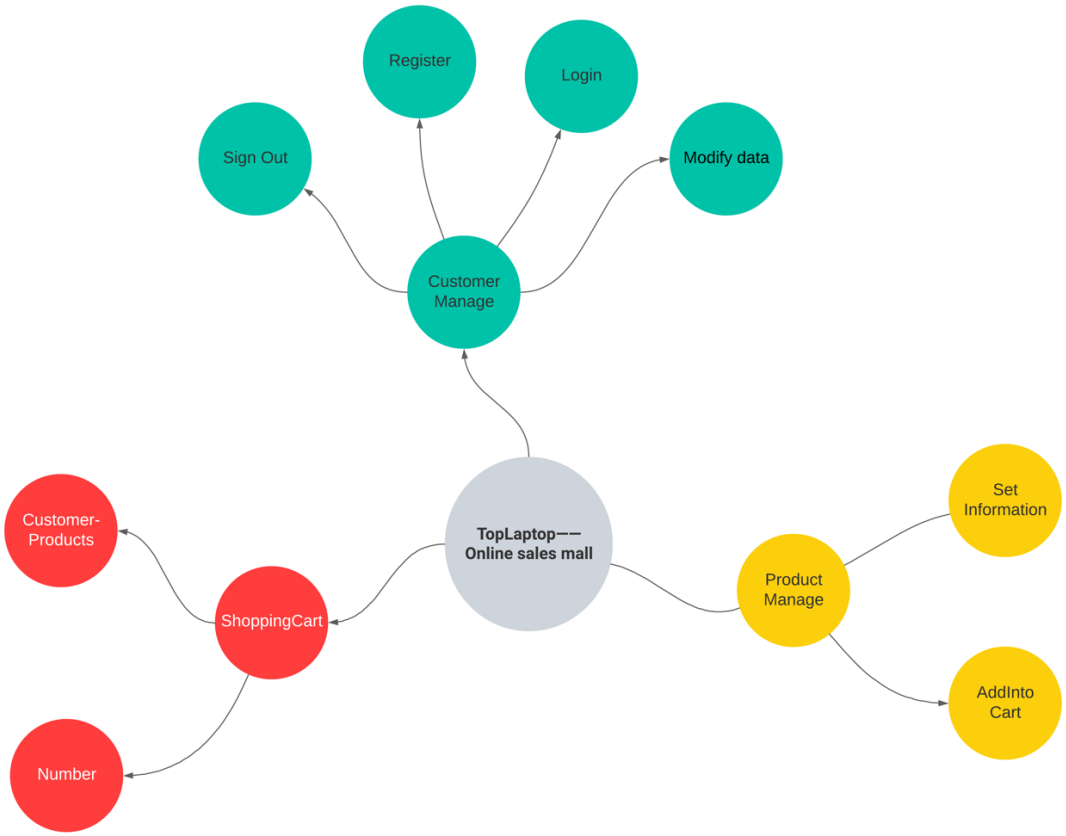
版本：

依赖	版本
axios	0.27.2
Element-plus	2.2.17
vue	3.2.13

tomcat	9.0.41
java	1.8
mybaits	2.2.2
mysql	8.0.22
springboot	2.3.7
redis	3.2.100

1.3 功能结构

系统的功能结构图如下：



二、数据库基本表的定义

1. 用户表

序号	名称	数据类型	大小	是否必填	是否主键
1	customer_id	Int	10	是	是
2	customer_name	varchar	100	是	否
3	login_pwd	varchar	100	是	否
4	phone	varchar	100	是	否
5	address	varchar	100	是	否
6	birthday	date	-	是	否
7	Email	varchar	100	是	否
8	picUrl	varchar	100	否	否

2. 产品表

序号	名称	数据类型	大小	是否必填	是否主键
1	product_id	Int	10	是	是
2	product_name	varchar	100	是	否
3	price	double	50	是	否
4	size	double	50	是	否
5	picture	varchar	100	是	否
6	SSD	int	10	是	否

7	memory	int	10	是	否
8	sales	int	10	是	否
9	brand	varchar	100	是	否

3. 购物车

序号	名称	数据类型	大小	是否必填	是否主键
1	customer_id	int	10	是	否
2	customer_name	int	10	是	否
3	isSelect	tinyint	1	是	否
4	Num	int	10	是	否

三、系统重要功能实现方法

(一)系统功能概述

“TopLaptop”——以售卖笔记本电脑为主题的线上商城！随着现代世界计算机科学技术的发展与普及，笔记本电脑凭借其便携性与出色的性能，已经成为了人们生活中必不可缺的伙伴，在日常的学习、工作、娱乐中扮演着重要角色；同时，随着线上网购的兴起，能通过网络选购自己心仪的电子产品也成为人们特别是年轻人群体的需求之一。 因此，我们通过开发设计 TopLaptop 笔记销售商场，以简洁而具有科技感的界面与海量的优质产品，满足各类顾客的不同需求，给予用户最佳的购物体验。

(二)客户端功能实现

1. 页面跳转

使用 Vue.js 搭建的实际上是单页面应用，但其所实现的效果为多页面程序，我们主要使用了两种页面切换方式:

(1) 使用路由跳转实现页面切换效果

Vue Router 是 Vue.js 官方的路由管理器，它和 Vue.js 的核心深度集成，可以通过将组件 (components) 映射到路由 (routes)，然后告诉 Vue Router 在哪里渲染它们来实现页面跳转的效果，如:

```
const routes = [
  { path: '/', redirect: '/home'}, // 根目录/ 重定向到 /home
  { path: '/home', component: Home},
  {
    path: '/login',
    name: 'Login',
    component: Login,
  },
  { path: '/test', component: Test},
  { path: '/register', component: Register},
  { path: '/shop/mShopPage', component: MShopPage},
  {
    path: '/userPage', // 设置这一级的路由展示页面
    redirect: '/userPage/overview', // 自动重定向到overview
    meta: { authRequired: true },
    children: [
      { path: 'overview', component: OverView},
      { path: 'details', component: Details},
      { path: 'payway', component: PayWay},
      { path: 'security', component: Security},
      { path: 'shopcart', component: ShopCart},
      { path: 'history', component: History},
    ]
  },
]
```

(2) 通过控制元素显示实现页面切换效果 ，Vue.js 中的 v-if 指令用于条件性地渲染一块内容，这块内容只会在指令的表达式返回 true 值的时候被渲染，可以通过在同一个位置设置两个(或多个) 无法同时显示的元素实现页面切换的效果，并通过 @click 或相应方法设置触发条件，如:

```
<el-dropdown-item @click="goLogin" v-if="!haveToken">
  <el-icon class="iconfont icon-yonghu_user"></el-icon>
  登陆账户
</el-dropdown-item>
<el-dropdown-item @click="goRegister" v-if="!haveToken">
  <el-icon class="iconfont icon-tianjia_add"></el-icon>
  注册账户
</el-dropdown-item>
<el-dropdown-item @click="goUserPage" v-if="haveToken">
  <el-icon class="iconfont icon-yinliu_drainage"></el-icon>
  个人页面
</el-dropdown-item>
<el-dropdown-item v-if="haveToken">
  <el-icon class="iconfont icon-tuichu_exit"></el-icon>
  退出登陆
</el-dropdown-item>
```

2. 格式验证

为了提高用户账号的安全性，我们在账号注册页面中加入了格式验证功能，对应数据的要求如下：

```
// 这是表单的验证规则对象
const registerFormRef = ref<FormInstance>()
const validatePass1 = (rule: any, value: string, callback: any) => {
  if (value === '') {
    callback(new Error('请输入密码'))
  } else if (value.length < 6 || value.length > 20){
    callback(new Error('长度在 6 到 20 个字符之间'))
  } else{
    if (registerForm.password_again !== '') {
      if (!registerFormRef.value) return
      registerFormRef.value.validateField('password_again', () => null)
    }
    callback()
  }
}
```

```
const validatePass2 = (rule: any, value: string, callback: any) => {
  if (value === '') {
    callback(new Error('请再次输入密码'))
  } else if (value.length < 6 || value.length > 20) {
    callback(new Error('长度在 6 到 20 个字符之间'))
  } else if (value !== registerForm.password) {
    callback(new Error("两次输入密码不一致"))
  } else {
    callback()
  }
}
```

(三)客户端与服务器端的交互

首先是前后端的连接问题。

通过设置跨域访问进行，当前端与后端处于一个局域网中时(例如连接同一个 wifi)，通过 IP 地址可以实现跨域访问。 前端依赖 axios 模块，通过设置 ip 地址连接同一个局域网内的后端。之后封装好所需要的 Http 请求传递给后端。在此过程中需要解决跨域问题。

在本项目中使用 spring 解决跨域的方式，即 Cross-Origin Resource Sharing Registry(CrosRegistry，跨域资源共享登记)。

首先在 pom.xml 中引入 jar 包：

```
<plugin>
    <groupId>org.springframework.boot</groupId>
    <artifactId>spring-boot-maven-plugin</artifactId>
    <version>2.3.7.RELEASE</version>
    <configuration...>
    <executions...>
</plugin>
```

之后在配置类中引入@Configuration 注解，并实现 WebMvcConfigurer 接口中的 addCorsMapping 方法。

```java
package com.example.demo.config;

import org.springframework.context.annotation.Configuration;
import org.springframework.web.servlet.config.annotation.CorsRegistry;
import org.springframework.web.servlet.config.annotation.WebMvcConfigurer;

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

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

使用@RestController 和@RequestMapping 注解驱动服务器端的 controller 层。通过@ResponseBody、@RequestBody、@Param 注解接收前端请求传递的参数并将前端发送的 JSON 字符串转成方法形参的对象。 例如

```java
    @PostMapping("/buyAll")
    public boolean buyAll(@Param("customerId")int customerId){
        List<Cart> carts = cartMapper.getAllSelect(customerId);
        String orderAddress = customerMapper.getAddress(customerId);
        for(Cart cart:carts){
            double pay = productsMapper.getPrice(cart.getProductId());
            Date orderTime = new Date();
            orderMapper.insert(customerId, orderAddress, orderTime,pay);
        }
        return cartMapper.deleteAllSelect(customerId) == carts.size();
    }
```

(四)服务器端功能实现

1. 服务器与数据库的连接

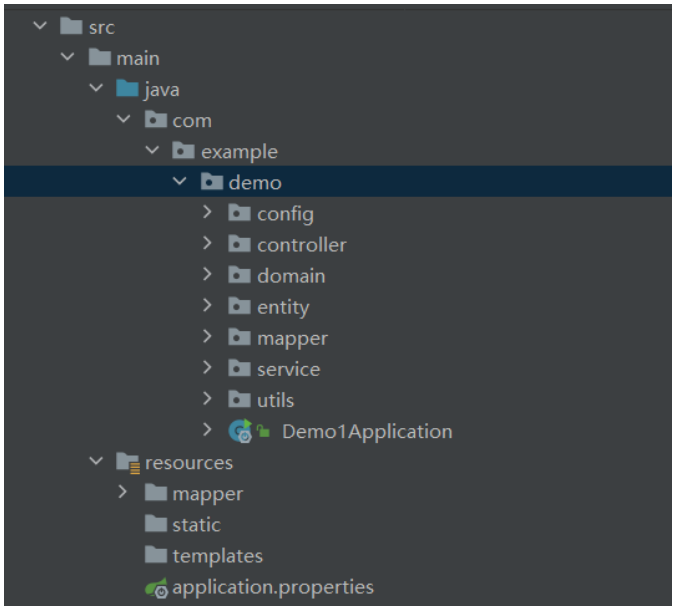
通过 springboot 框架内的 JDBC 接口连接到数据库，将数据库端口及密码等写入 application.properties 配置文件中，springboot 中的 org.springframework.boot.autoconfigure.jdbc 包根据配置文件进行数据源

的自动配置与连接。数据库连接相关的配置如下：

```
spring.datasource.driver-class-name=com.mysql.cj.jdbc.Driver
spring.datasource.name=defaultDataSource
spring.datasource.url=jdbc:mysql://localhost:3306/sale_system
spring.datasource.username=root
spring.datasource.password=20030816zjh
```

2. 系统业务逻辑实现

Springboot 中的业务逻辑主要分为三个模块，即控制层（controller）、实体层（entity）、映射层（mapper）：



（1） 控制层连接前端请求与映射层，调用映射层方法并将结果返回给前端进行展示。

```
@PostMapping("/register")
public boolean register(@RequestBody Customer customer){
    return customerMapper.insert(customer.getCustomerName(), customer.getLoginPwd())==1;
}
```

（2） 实体层提供了访问数据的对象，屏蔽了数据存储最终介质的不同与具体的实现细节，降低了业务逻辑层的复杂度。

```java
package com.example.demo.entity;

import lombok.Data;

import java.sql.Date;

14 usages
@Data
public class Customer {
    private int customerId;
    private String customerName;
    private String loginPwd;
    private String phone;
    private String address;
    private Date birthday;
    private String email;
    private String picUrl;
}
```

3. 安全性

我们为用户的每次登录生成一个一个独一无二的 token，并对需要鉴权的路由函数，从用户的请求头中取出对应的 token 进行验证。

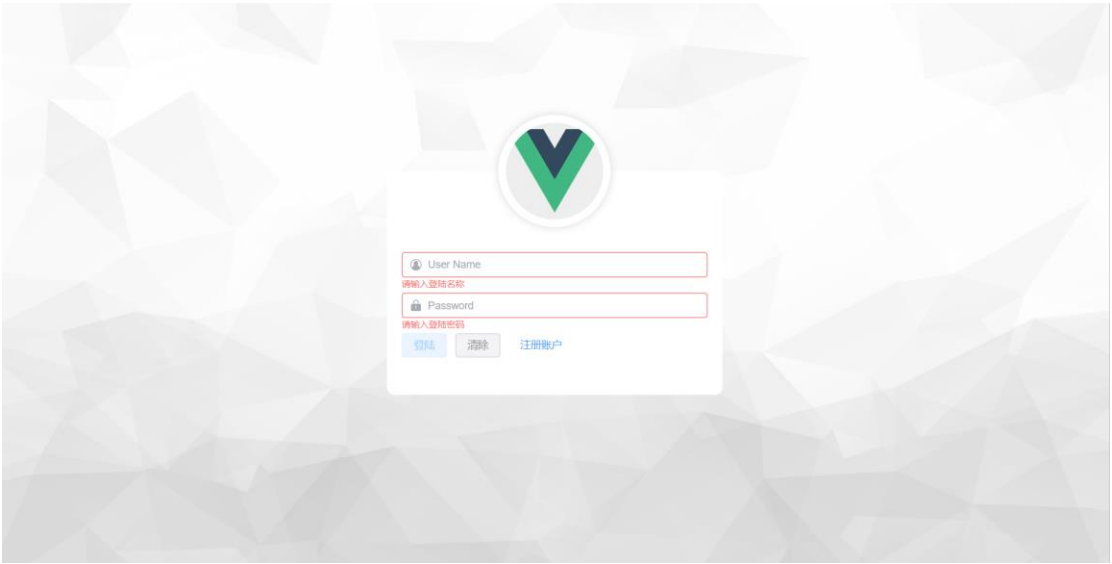
```java
@Override
public boolean preHandle(HttpServletRequest request, HttpServletResponse response,Object handler)throws  Exception{
    /** 使用HttpServletRequest获取客户端的请求参数 */
    if (request.getMethod().equals("OPTIONS")){
        response.setStatus(HttpServletResponse.SC_OK); //status=200
        return true;
    }
    response.setCharacterEncoding("utf-8");
    System.out.println(request.getHeader( s: "token"));
    String token = request.getHeader( s: "token"); //获取单个请求头name对应的value值，即获取token的字符串
    if(token!=null){
        boolean result = TokenUtil.verify(token);
        if(result){
            System.out.println("通过拦截器");
            return true;
        }
    }
    response.setCharacterEncoding("UTF-8");
    response.setContentType("application/json; charset=utf-8");
    PrintWriter out = null;
    //以json格式返回失败信息
    try {...}catch (Exception e){...}
    return false;

}
```

四、系统实现结果

（一）主页面

（二）登陆页面



（三）注册页面


User Name
请输入用户名
Password
请输入密码
Password Again
注册
已有账户?
登陆

（四）商城首页


TopLaptop
最新热卖
轻薄本
游戏本
全能本
电脑配件
我的账户
电脑1
戴尔
¥5000
加入购物车
电脑2
联想
¥4000
加入购物车
电脑3
apple
¥6000
加入购物车
电脑4
华硕
¥7000
加入购物车

（五）个人页面

（六）个人信息修改页面

（七）购物车页面

五、总结

本次数据库课设的任务 1，本组同学完成了一次基础的数据库系统开发。我们团队的分工模式为前后端分离开发，三人分别负责前端、中端与后端，从零开始学习搭建系统平台。在实现过程中，我们自学了使用 HTML、CSS、JavaScript 和 Vue 进行页面前端的开发，了解了网页开发前后端的相关知识；同时，学会使用 JDBC 接口连接数据库并对数据库进行操作的原理和流程，在框架搭建过程中更深入地理解了 MVC 模式，同时了解了前后端交互的基本方法与 webservice 的部署；掌握了 MySQL 数据库的操作方法，并对增删查改方法进行优化，以及如何使用爬虫将互联网上的海量信息爬取存入数据库中；我们对于课上学到的数据库的设计理论知识进行了实践，在实际工程中进一步加深了对于数据库理论的理解。

回看这个过程，我们从最初拿到任务时的懵懵懂懂不知如何下手，到在互联网上寻找资料进行自学，再到一次次讨论逐渐清楚明确思路，最终收获成功实现了一个功能较完整的线上售卖商城系统的喜悦。在这个过程中，我们曾因经验不足走过不少弯路，也在交流讨论中有过意见不一致的争吵，在实践学习

中一次次地优化和改进自己的代码——所有这些的经历，让我们学会了如何与团队进行沟通与合作，如何通过各种渠道探索学习新的知识。我们将携着在本次作业的收获，继续学习与改进，争取在第二次作业中完成更加优秀的成果！