****

**期末考核大作业报告**

作业成绩：

课 程：程序设计（4）（Java平台）

姓 名： 何庚坤

学 号： 2018214139

班 级： 13901801

[1 大作业2 3](#_Toc43034339)

[1.1 作业题目 3](#_Toc43034340)

[1.2 程序分析与设计 3](#_Toc43034341)

[1.3 程序关键代码及解释 5](#_Toc43034342)

[1.4 系统运行截图 11](#_Toc43034343)

[1.5 总结与心得 13](#_Toc43034344)

# 1 大作业2

## 1.1 作业题目

基于 MVC 架构的 XXX 信息查询系统

## 1.2 程序分析与设计

**1、程序设计思想**

本系统采用MVC的架构，采用JSP+Servlet+JavaBean实现，JSP负责页面数据展示即View层，Servlet负责HTTP请求的接收与转发即Controller层，JavaBean负责数据即Model层。

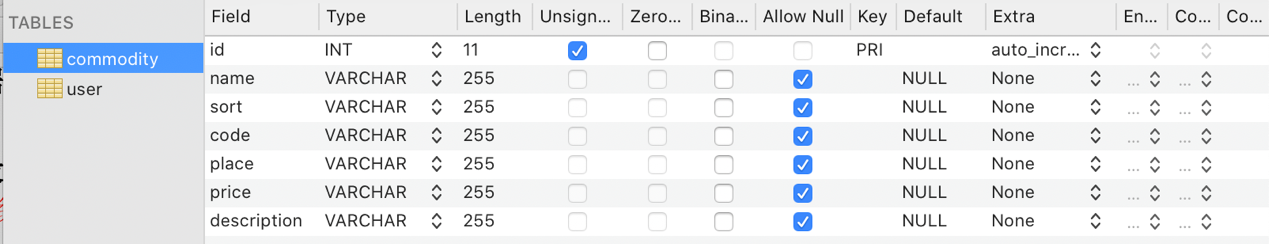
**2、程序功能介绍**

**登录功能：**项目启动后，首先进入登录页面login.jsp，用户名和密码经过数据库校验后可登陆成功，否则会重新转发到登录页。

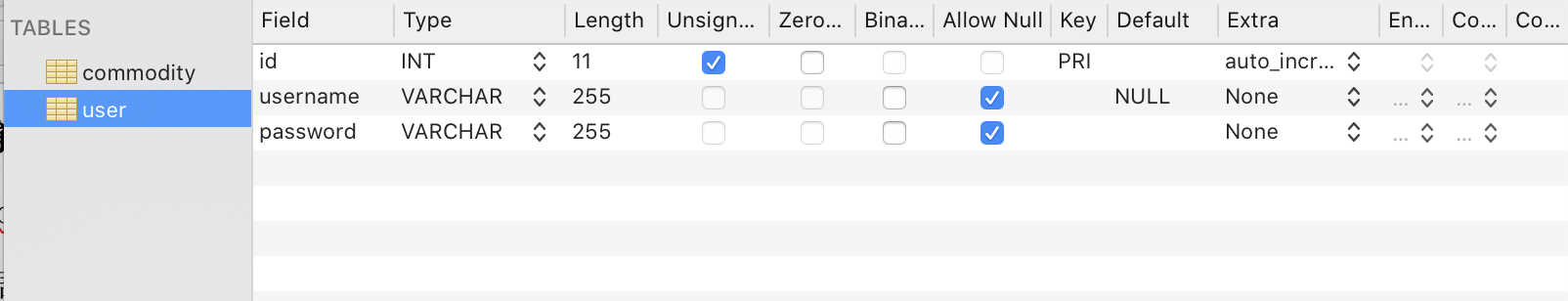
**展示功能：**登录进入系统后，commodities.jsp页面会展示数据库内所有商品的全部信息。

**查询功能：**在commodities.jsp中文本框内输入商品编码，点击提交后，请求经过GetCommoditiesServlet处理，在commodities.jsp展示该商品编码对应商品的所有信息。

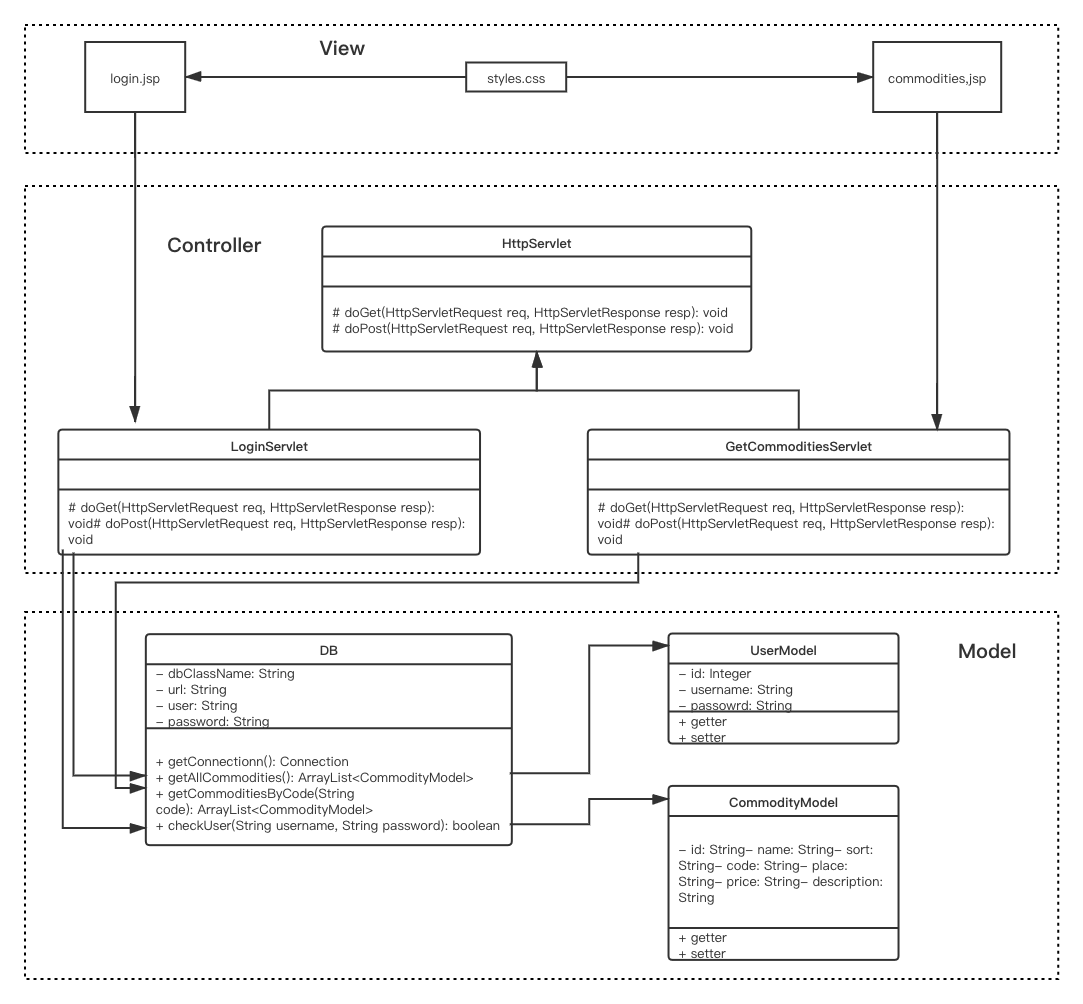
**3、数据库设计**



商品信息表



用户表

**4、类设计**

**LoginServlet:**继承HttpServlet，实现doGet方法，doPost的实现是调用doGet进行操作，用户通过login.jsp登录，通过form表单对LoginServlet发起post请求，在doPost中调用DB封装的checkUser方法，验证用户是否在mySQL数据库中，并验证密码是否正确，如果没问题就调用DB封装的getAllCommodities方法获取当前数据库中所有的商品信息，并存到session中，然后跳转到commodities.jsp页面进行显示

**GetCommoditiesServlet:**继承HttpServlet，实现doGet方法，doPost的实现是调用doGet进行操作，用户登录后进入到commodities.jsp页面中，通过输入商品编号进行查询，form表单发送get请求到GetCommoditiesServlet中的doGet进行处理，doGet调用DB封装的getComoditiesByCode方法获取商品编号相同的商品信息并返回，存入到session中通过commodities.jsp显示到页面上

**DB：**是对数据库操作Model层的封装，一个连接Model和Controller层的中介者，通过UserModel和CommodityModel实例操作数据库

**UserModel：**是数据库中User表的JavaBean

**CommodityModel：**是数据库中Commodity表的JavaBean

## 1.3 程序关键代码及解释

**GetCommoditiesServlet.class**

@WebServlet(name = "GetCommoditiesServlet", urlPatterns = "/GetCommoditiesServlet")

public class GetCommoditiesServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws IOException {

req.setCharacterEncoding(String.valueOf(StandardCharsets.UTF\_8));

resp.setContentType("text/html");

resp.setCharacterEncoding(String.valueOf(StandardCharsets.UTF\_8));

// **获取commodities.jsp页面form表单发来的code参数，用于查询商品**

String code = req.getParameter("code");

DB dbUtil = new DB();

try {

// **通过DB实例获取数据库中查询code相同的商品信息**

ArrayList<CommodityModel> commodities = dbUtil.getCommoditiesByCode(code);

HttpSession session = req.getSession();

session.setMaxInactiveInterval(3600);

// **通过session传给commodities.jsp页面**

session.setAttribute("commodities",commodities);

session.setAttribute("code",code);

resp.sendRedirect(req.getContextPath()+"/commodities.jsp");

} catch (SQLException | ClassNotFoundException e) {

e.printStackTrace();

}

}

@Override

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

doGet(req, resp);

}

}

**LoginServlet.class**

@WebServlet(name = "LoginServlet", urlPatterns = "/LoginServlet")

public class LoginServlet extends HttpServlet {

@Override

protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

req.setCharacterEncoding(String.valueOf(StandardCharsets.UTF\_8));

resp.setContentType("text/html");

resp.setCharacterEncoding(String.valueOf(StandardCharsets.UTF\_8));

DB dbUtil = new DB();

try {

// **通过DB实例检查login.jsp页面输入的username和password是否在user表中，并且password正确**

if (dbUtil.checkUser(req.getParameter("username"), req.getParameter("password"))) {

try {

// **如果正确则获取数据库中所有的商品信息**

ArrayList<CommodityModel> commodities = dbUtil.getAllCommodities();

System.out.println(commodities);

HttpSession session = req.getSession();

session.setMaxInactiveInterval(3600);

// **存session中给commodities.jsp页面**

session.setAttribute("commodities", commodities);

// **显示登录成功页面**

resp.getWriter().println("<!DOCTYPE html>" +

"<html lang=\"en\">" +

"<head>" +

"<meta charset=\"UTF-8\">" +

"<meta name=\"viewport\" content=\"width=device-width, initial-scale=1.0\">" +

"<meta http-equiv=\"X-UA-Compatible\" content=\"ie=edge\">" +

"<link href=\"./assets/styles.css\" rel=\"stylesheet\">" +

"<title>" +

"登录成功" +

"</title>" +

"</head>" +

"<body class=\"container\">" +

"<img class=\"login\_\_success\" src=\"./assets/images/success.png\" />" +

"登录成功" +

// **手动跳转commodities.jsp页面**

"<a href=\"commodities.jsp\" class=\"login\_\_form--link\">查询商品</a>" +

"</body>" +

"</html>");

} catch (SQLException | ClassNotFoundException e) {

e.printStackTrace();

}

} else {

// **错误则提示错误并跳转到login.jsp页面重新登录**

resp.getWriter().println("账号或密码错误");

req.getRequestDispatcher("src/view/login.jsp").forward(req, resp);

}

} catch (SQLException | ClassNotFoundException e) {

e.printStackTrace();

}

}

@Override

protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {

doGet(req, resp); // **调用doGet方法**

}

}

**DB.class**

public class DB {

private final String dbClassName = "com.mysql.jdbc.Driver";

private final String url = "jdbc:mysql://localhost:3306/MVCcurd?useSSL=false";

private final String user = "root";

private final String password = "Hgk20180921";

// **获取数据库的Connection实例**

public Connection getConnection() throws SQLException, ClassNotFoundException {

Class.forName(dbClassName);

return DriverManager.getConnection(url, user, password);

}

// **获取所有数据库中commodities信息**

public ArrayList<CommodityModel> getAllCommodities() throws SQLException, ClassNotFoundException {

Statement statement = getConnection().createStatement();

ResultSet resultSet = statement.executeQuery("select \* from commodity");

ArrayList<CommodityModel> list = new ArrayList<CommodityModel>();

while(resultSet.next()){

CommodityModel g = new CommodityModel();

g.setId(resultSet.getInt("id"));

g.setSort(resultSet.getString("sort"));

g.setCode(resultSet.getString("code"));

g.setName(resultSet.getString("name"));

g.setPlace(resultSet.getString("place"));

g.setPrice(resultSet.getString("price"));

g.setDescription(resultSet.getString("description"));

list.add(g);

}

return list;

}

// **通过code获取数据库中commodities信息**

public ArrayList<CommodityModel> getCommoditiesByCode(String code) throws SQLException, ClassNotFoundException {

Statement statement = getConnection().createStatement();

ResultSet resultSet = statement.executeQuery("select \* from commodity where code = '" + code + "'");

ArrayList<CommodityModel> list = new ArrayList<CommodityModel>();

while(resultSet.next()){

CommodityModel g = new CommodityModel();

g.setId(resultSet.getInt("id"));

g.setSort(resultSet.getString("sort"));

g.setCode(resultSet.getString("code"));

g.setName(resultSet.getString("name"));

g.setPlace(resultSet.getString("place"));

g.setPrice(resultSet.getString("price"));

g.setDescription(resultSet.getString("description"));

list.add(g);

}

return list;

}

// **检查user的username和password**

public boolean checkUser(String username, String password) throws SQLException, ClassNotFoundException {

Statement statement = getConnection().createStatement();

ResultSet resultSet = statement.executeQuery(" select username,password from user where username ='" + username + "'and password = '" + password + "' ");

UserModel g = null;

while(resultSet.next()){

g = new UserModel();

g.setUsername(resultSet.getString("username"));

g.setPassword(resultSet.getString("password"));

}

return g != null;

}

}

**UserModel.class**

public class UserModel { // **User表的JavaBean**

private Integer id;

private String username;

private String password;

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getUsername() {

return username;

}

public void setUsername(String username) {

this.username = username;

}

public String getPassword() {

return password;

}

public void setPassword(String password) {

this.password = password;

}

}

**CommodityModel.class**

public class CommodityModel { // **commodity表的JavaBean**

private Integer id;

private String name;

private String sort;

private String code;

private String place;

private String price;

private String description;

public void setId(Integer id) {

this.id = id;

}

public Integer getId() {

return id;

}

public String getName() {

return name;

}

public void setName(String name) {

this.name = name;

}

public String getSort() {

return sort;

}

public void setSort(String sort) {

this.sort = sort;

}

public String getCode() {

return code;

}

public void setCode(String code) {

this.code = code;

}

public String getPlace() {

return place;

}

public void setPlace(String place) {

this.place = place;

}

public String getPrice() {

return price;

}

public void setPrice(String price) {

this.price = price;

}

public void setDescription(String description) {

this.description = description;

}

public String getDescription() {

return description;

}

}

## 1.4 系统运行截图



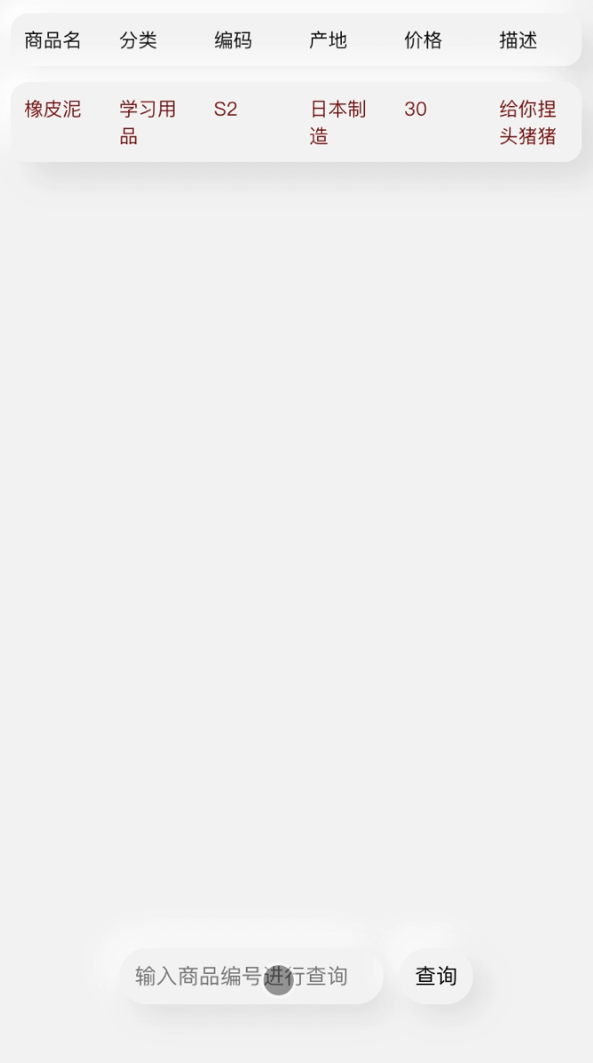
登录页面



登录成功页



查询全部商品页



商品编码查询商品页

## 1.5 总结与心得

这次Java大作业二确实让我学到了许多，也让我认识到了更多，让我有了许多技术态度方面的想法。之前的C++虽然足够底层，但是学校更多是让我们入门编程，获得编程的一些基本思想，基本不能接触到实际编写出工程的一些东西和思想，比如做出一个实际可以看到的网页，MVC的设计模式等。

虽然这学期的Java课教的Web部分对于技术上来说比较过时，但是通过学习这些历史上的东西，确实可以认识到以前人们做网页的方式。我一直认为如果学习一样东西需要从头开始，学习他以前的历史，只有学会之前的技术方案，才能理解现在使用的技术方案所出现的意义和理念的进步性。

在我真正写大作业二的JSP之前，我一直都是对于JSP这种开发方式不屑一顾，但是本着了解历史的心态，我还是完成了JSP的编写。而在完成后我才明白JSP仅仅是在技术上过时，但是在某些技术架构上仍然适用，比如多页面架构，现在的Web应用大多是单页面架构，虽然单页面交互性更强，但是单页面也有其不足之处，不利于SEO，首屏渲染缓慢等，对于交互性不需要那么强的网站门户页面等，多页面其实更适合，利于SEO，渲染快，这时候JSP就很合适，由于交互性的减少，导致后端请求接口少，业务逻辑不多，页面的编写和业务逻辑混合到一起实则加快了开发效率，省时省人力。这让我知道没有绝对的技术过时，只有技术架构上的适合与否。