****

黑马商城的设计与实现

**设计题目： 黑马商城的设计与实现**

**系（部）： 信息科学学院**

**专 业： 软件工程**

**小 组: 李步官组**

**小组成员： 李步官17053007**

**李 赟 17053173**

**劳健峰17161059**

**张东源 17053069**

目录

**[绪 论](#_Toc20535_WPSOffice_Level1)** **[3](#_Toc20535_WPSOffice_Level1)**

**[一、 可行性研究阶段：](#_Toc26954_WPSOffice_Level1)** **[4](#_Toc26954_WPSOffice_Level1)**

[1.技术可行性](#_Toc26954_WPSOffice_Level2) [4](#_Toc26954_WPSOffice_Level2)

**[二、需求分析：](#_Toc2710_WPSOffice_Level1)** **[6](#_Toc2710_WPSOffice_Level1)**

[1.用户模块](#_Toc2710_WPSOffice_Level2) [6](#_Toc2710_WPSOffice_Level2)

[2.商品模块](#_Toc22266_WPSOffice_Level2) [6](#_Toc22266_WPSOffice_Level2)

**[三、设计阶段](#_Toc22266_WPSOffice_Level1)** **[6](#_Toc22266_WPSOffice_Level1)**

[1.概要设计——模块结构图](#_Toc24297_WPSOffice_Level2) [6](#_Toc24297_WPSOffice_Level2)

[2.数据库设计——ER图、数据表](#_Toc17911_WPSOffice_Level2) [7](#_Toc17911_WPSOffice_Level2)

[3.过程设计——各个功能模块对应的程序流程图](#_Toc423_WPSOffice_Level2) [10](#_Toc423_WPSOffice_Level2)

**[四、实现阶段](#_Toc24297_WPSOffice_Level1)** **[12](#_Toc24297_WPSOffice_Level1)**

[（1）用户注册页](#_Toc29626_WPSOffice_Level2) [12](#_Toc29626_WPSOffice_Level2)

[（2）用户登录页](#_Toc1805_WPSOffice_Level2) [15](#_Toc1805_WPSOffice_Level2)

[（3）用户退出](#_Toc17826_WPSOffice_Level2) [17](#_Toc17826_WPSOffice_Level2)

[（4）显示热门和最新商品](#_Toc16554_WPSOffice_Level2) [18](#_Toc16554_WPSOffice_Level2)

[（5）根据类别，分页显示所有商品](#_Toc2538_WPSOffice_Level2) [20](#_Toc2538_WPSOffice_Level2)

[（6）根据编号，显示商品详细信息](#_Toc8976_WPSOffice_Level2) [23](#_Toc8976_WPSOffice_Level2)

**[五、测试阶段：用的测试环境、测试方法和测试用例](#_Toc17911_WPSOffice_Level1)** **[25](#_Toc17911_WPSOffice_Level1)**

[1.测试环境](#_Toc19738_WPSOffice_Level2) [25](#_Toc19738_WPSOffice_Level2)

[2.测试方法](#_Toc12798_WPSOffice_Level2) [25](#_Toc12798_WPSOffice_Level2)

[3.测试用例](#_Toc20475_WPSOffice_Level2) [25](#_Toc20475_WPSOffice_Level2)

绪 论

黑马商城是一个综合性的线上购物平台，提供数码产品、服装饰品等多种类的商品，还支持按热度及按上架时间排序等诸多功能。在这次实训设计中，使用MVC模式，采用MySQL5.7数据库和基于JavaEE的java web技术，开发了界面友好、操作简单易懂的黑马商城网站。

在设计过程中，本文讲述了基于JavaEE的黑马商城网站的设计与实现，主要包括系统运用的关键技术，数据库设计，对各个功能模块的详细设计以及实现，本次设计主要实现黑马商城网站中以下几个功能：用户注册，用户登录，用户退出，显示热门和最新商品，根据商品类别，分页显示所有商品，根据商品编号，分页显示所有商品的功能。

一、可行性研究阶段：

1.技术可行性

黑马商城网站的开发主要包括前台和后台两个方面，不管是用户还是系统管理员都是通过浏览器（如谷歌，火狐，IE等）进入该系统。对于系统前台，要求系统的界面友好、美观大方、操作方便等要求，实现美观的界面；对于系统后台则要求信息更新及时、数据存储安全可靠、数据一致性高及易维护等要求，要实现用户注册，用户登录，用户退出，显示热门和最新商品，根据商品类别，分页显示所有商品，根据商品编号，分页显示所有商品的功能。

本系统采用java web及其三层架构技术以及HTML&CSS&JavaScript，JDBC，Druid连接池，JdbcTemplate，Servlet，JSP&JSTL&EL，使用Intellij IDEA集成开发环境是一套完整的开发工具集，可以用于生成基于JSP的应用程序，使用Navicat数据库管理工具对MySQL数据库各种操作，这些大大提高了编程的效率和代码质量，而现在开发所需语言技术都是在大学期间学过的，以及网上相关技术教程，从而在技术上为本系统的设计和开发提供了可能。

（1）三层架构

三层分别为：Dao层、service层和servlet层。

DAO(Data Access Object) 数据访问对象是一个面向对象的数据库接口，用于连接数据库及进行数据库的操作如：查询、删除、更改等。

Servlet（Server Applet）是Java Servlet的简称，用于验证数据、实例化JavaBean、调用DAO连接数据库、控制页面跳转。

service层叫服务层，被称为服务，肯定是相比之下比较高层次的一层结构，相当于将几种操作封装起来。

（2）HTML&CSS&JavaScript

HTML，用于对网页的前端进行设计，建立能与后端交互的通道。

CSS，层叠样式表(Cascading Style Sheets)是一种用来表现HTML等文件样式的计算机语言。常用于规定字体的格式，设置元素的位置。

JavaScript，一种直译式脚本语言，是一种动态类型、弱类型、基于原型的语言，内置支持类型，能用来给HTML网页增加动态功能，例如构造函数。

（3）JDBC

JDBC（Java DataBase Connectivity,java数据库连接）是一种用于执行SQL语句的Java API，可以为多种关系数据库提供统一访问，它由一组用Java语言编写的类和接口组成。

（4）Druid

Druid是阿里巴巴的数据库连接池，能提高访问数据库的速度。

（5）JdbcTemplate

Spring对数据库的操作在jdbc上面做了深层次的封装，使用Spring的注入功能，可以把DataSource（数据库连接池）注册到jdbcTemplate中。暴力映射。（6）Servlet

Servlet（Server Applet）是Java Servlet的简称，称为小服务程序或服务连接器，用Java编写的服务器端程序，具有独立于平台和协议的特性，主要功能在于交互式地浏览和生成数据，生成动态Web内容。

狭义的Servlet是指Java语言实现的一个接口，广义的Servlet是指任何实现了这个Servlet接口的类，一般情况下，人们将Servlet理解为后者。Servlet运行于支持Java的应用服务器中。从原理上讲，Servlet可以响应任何类型的请求，但绝大多数情况下Servlet只用来扩展基于HTTP协议的Web服务器。

（7）JSP&JSTL&EL

JSP，全名为Java Server Pages，中文名叫java服务器页面，其根本是一个简化的Servlet设计， JSP技术使用Java编程语言编写类XML的tags和scriptlets，来封装产生动态网页的处理逻辑。

JSTL，（JavaServer Pages Standard Tag Library)是一个不断完善的开放源代码的JSP标签库，是由apache的jakarta小组来维护的。

EL，（Expression Language） 是为了使JSP写起来更加简单。表达式语言的灵感来自于 ECMAScript 和 XPath 表达式语言，它提供了在 JSP 中简化表达式的方法，让Jsp的代码更加简化。

二、需求分析：

1.用户模块

（1）用户注册

如果用户为非注册用户，通过注册获得一个登陆身份。

（2）用户登录

如果用户已经是注册会员，可以进行登录，登录后可进行其他操作。

（3）用户退出

登录用户在网站的右上角进击退出按钮即可退出。

2.商品模块

（1）显示热门和最新商品

在网站首页的热门商品模块和最新商品模块中显示热门和最新商品

（2）根据商品类别，分页显示所有商品

在网站的商品列表页中，根据商品类别，分页显示所有商品

（3）根据商品编号，显示商品详细信息

在网站的商品列表页中，根据商品编号，显示商品详细信息

三、设计阶段

1.概要设计——模块结构图

（1）模块结构图



图3-1 模块结构图

2.数据库设计——ER图、数据表



图3-2 ER图

（2）数据表

表3-1 用户表（tab\_user 表）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 默认值 | 是否主键 | 是否为空 | 说明 |
| uid | int(11) | 自增 | Yes | No | 用户编号 |
| username | varchar(20) |  | No | No | 用户名 |
| password | varchar(32) |  | No | No | 密码 |
| name | varchar(20 |  | No | No | 用户姓名 |
| email | varchar(30) |  | No | No | 邮箱 |
| telephone | varchar(20) |  | No | No | 电话号码 |
| birthday | date |  | No | No | 出生日期 |
| sex | varchar(10) |  | No | No | 性别 |

表3-2 商品类别表（tab\_category 表）

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 默认值 | 是否主键 | 是否为空 | 说明 |
| cid | int(11) | 自增 | Yes | No | 商品类别号 |
| cname | varchar(20) |  | No | No | 商品类别名 |

表3-3 商品详情表（tab\_product 表）

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 字段名称 | 类型 | 默认值 | 是否主键 | 是否为外键 | 是否为空 | 说明 |
| pid | int(11) | 自增 | Yes | No | No | 商品编号 |
| pname | varChar(50) |  | No | No | No | 商品名 |
| market\_price | double |  | No | No | No | 市场价格 |
| shop\_price | double |  | No | No | No | 商城售价 |
| pimage | varchar(200) |  | No | No | No | 商品图片 |
| pdate | date |  | No | No | No | 上架时间 |
| is\_hot | int(11) |  | No | No | No | 是否热门 |
| pdesc | varchar(255) |  | No | No | No | 商品详情 |
| pflag | int(11) |  | No | No | No | 是否线上销售 |
| cid | int(11) | 自增 | No | Yes | No | 商品类别号 |

3.过程设计——各个功能模块对应的程序流程图

（1）用户注册



图3-3 用户注册流程图

（2）用户登录



图3-4 用户登录流程图

（3）用户退出



图3-5 用户退出流程图

四、实现阶段

（1）用户注册页

UserServlet.java

/\*\*  
 \* 用户注册功能  
 \*/  
 private void register(javax.servlet.http.HttpServletRequest request, javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException, IOException {  
 try {  
 //01 一次性获取请求数据(表单中提交的全部数据)  
 Map<String, String[]> map = request.getParameterMap();  
  
 //02 创建User对象，用来封装  
 User user = new User();  
  
 //03 使用工具类BeanUtil将Map的值一一封装到User对象中去  
 BeanUtils.populate(user,map);  
  
 //04 调用业务层对象的方法  
// UserService userService = new UserService();  
 userService.register(user);  
  
 //05 重定向到register\_ok.jsp页面  
 response.sendRedirect(request.getContextPath() + "/register\_ok.jsp");  
  
  
 } catch (NameExistsException e) {  
 e.printStackTrace();  
 request.setAttribute("MESSAGE", "用户名存在，请更换用户名");  
 request.getRequestDispatcher("WEB-INF/message.jsp").forward(request,response);  
 }catch (Exception e) {  
 e.printStackTrace();  
 System.out.println("系统错误，请联系管理员");  
 request.setAttribute("MESSAGE", "系统错误，请联系管理员");  
 request.getRequestDispatcher("WEB-INF/message.jsp").forward(request,response);  
 }  
 }

//UserService.java  
/\*\*  
 \* service层的用户注册方法  
 \* 在这里进行业务操作  
 \* 如：对用户名是否存在进行判断  
 \*/  
public void register(User user) throws Exception {  
  
 //02 获取用户名  
 String username = user.getUsername();  
  
 //03 根据用户名检查该用户是否存在，存在则返回true，否则返回false  
 boolean flag = userDao.exists(username);  
  
 //04 判断  
 if (flag){  
 System.out.println("该用户名已存在，请更换用户名");  
  
 throw new NameExistsException();  
 }else {  
 //获取明文密码  
 String password = user.getPassword();  
 //将明文密码进行加密  
 String passwordMd5 = Md5Util.encodeByMd5(password);  
 //将加密后的密码再次封装到User对象中  
 user.setPassword(passwordMd5);  
 //调用Dao层中的注册方法对数据库进行访问  
 userDao.register(user);  
 }  
}

//UserDao.java

public class UserDao {  
 //获取数据库连接池  
 JdbcTemplate jt = new JdbcTemplate(DataSourceUtil.getDataSource());  
  
   
 /\*\*  
 \* Dao层的用户注册方法  
 \*/  
 public void register(User user){  
 String sql = "insert into tab\_user values(null,?,?,?,?,?,?,?)";  
 Object[] params = {  
 user.getUsername(),  
 user.getPassword(),  
 user.getName(),  
 user.getEmail(),  
 user.getTelephone(),  
 user.getBirthday(),  
 user.getSex()  
 };  
 jt.update(sql, params);  
 }

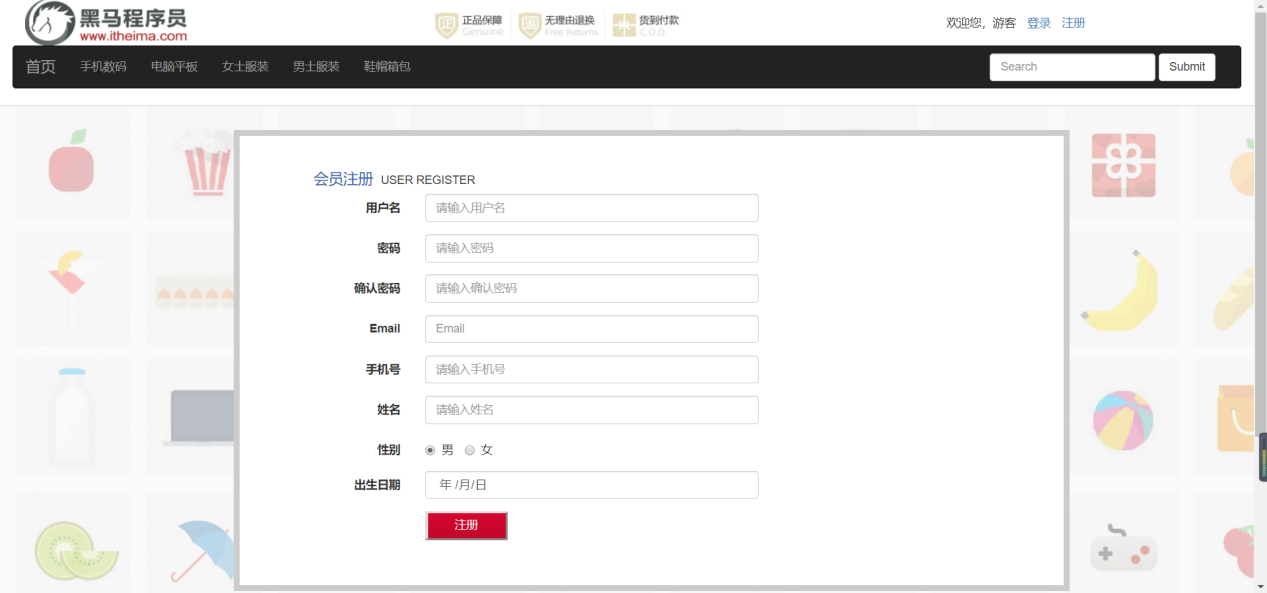


图4-1 用户注册页

（2）用户登录页

//UserServlet.java

/\*\*  
 \* 用户登录功能  
 \*/  
private void login(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 String username = request.getParameter("username");  
 String password = request.getParameter("password");  
  
  
 try {  
  
  
 //根据结果跳转到不同的页面  
 User user = userService.login(username, password);  
 if (user != null){  
 //登录成功情况  
 //将user对象放到HttpSession对象中  
 request.getSession().setAttribute("USER", user);  
  
 //重定向到default.jsp页面中  
 response.sendRedirect(request.getContextPath() + "/default.jsp");  
 }else{  
 //登录失败情况  
 request.setAttribute("MESSAGE", "帐号或密码错误");  
 request.getRequestDispatcher("WEB-INF/message.jsp").forward(request,response);  
 }  
  
  
 } catch (Exception e) {  
 e.printStackTrace();  
 request.setAttribute("MESSAGE", "系统出错，请联系管理员");  
 request.getRequestDispatcher("WEB-INF/message.jsp").forward(request,response);  
 }  
}

//UserService.java

/\*\*  
 \* service层的用户登录方法  
 \* 在这里进行业务操作  
 \* 对明文密码进行加密，再传到Dao层  
 \*/  
 public User login(String username, String password) throws Exception {  
   
 //将明文密码加密  
 String passwordMd5 = Md5Util.encodeByMd5(password);  
  
 //调用UserDao对象的方法  
 return userDao.login(username, passwordMd5);  
 }

//UserDao.java

/\*\*  
 \* Dao层的登录方法  
 \* @param username 用户名  
 \* @param passwordMd5 密文密码  
 \*/  
public User login(String username, String passwordMd5) {  
 try {  
 //查询用户名和密码  
 String sql = "select \* from tab\_user where username=? and password=?";  
 Object[] params = {  
 username,  
 passwordMd5  
 };  
  
 return jt.queryForObject(sql, params, new BeanPropertyRowMapper<User>(User.class));  
 } catch (DataAccessException e) {  
 e.printStackTrace();  
 return null;  
 }  
}

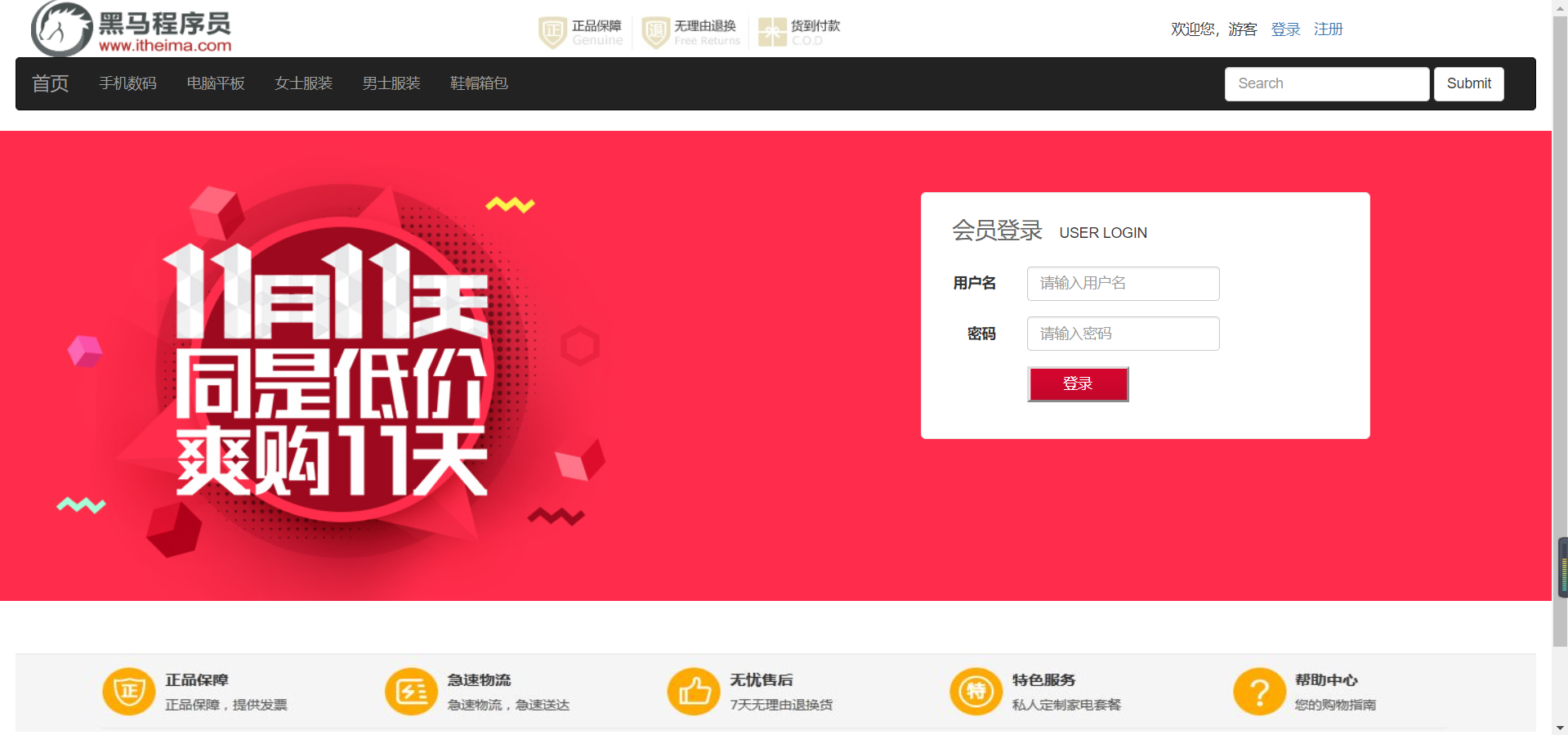


图4-2 用户登录页

（3）用户退出

//UserServlet.java

/\*\*  
 \* 用户退出功能  
 \*/  
private void logout(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 //01 获取HttpSession对象  
 HttpSession session = request.getSession();  
  
 //02 销毁HttpSession对象  
 session.invalidate();  
  
 //03 重定向到default.jsp对象  
 response.sendRedirect(request.getContextPath() + "/default.jsp");  
}

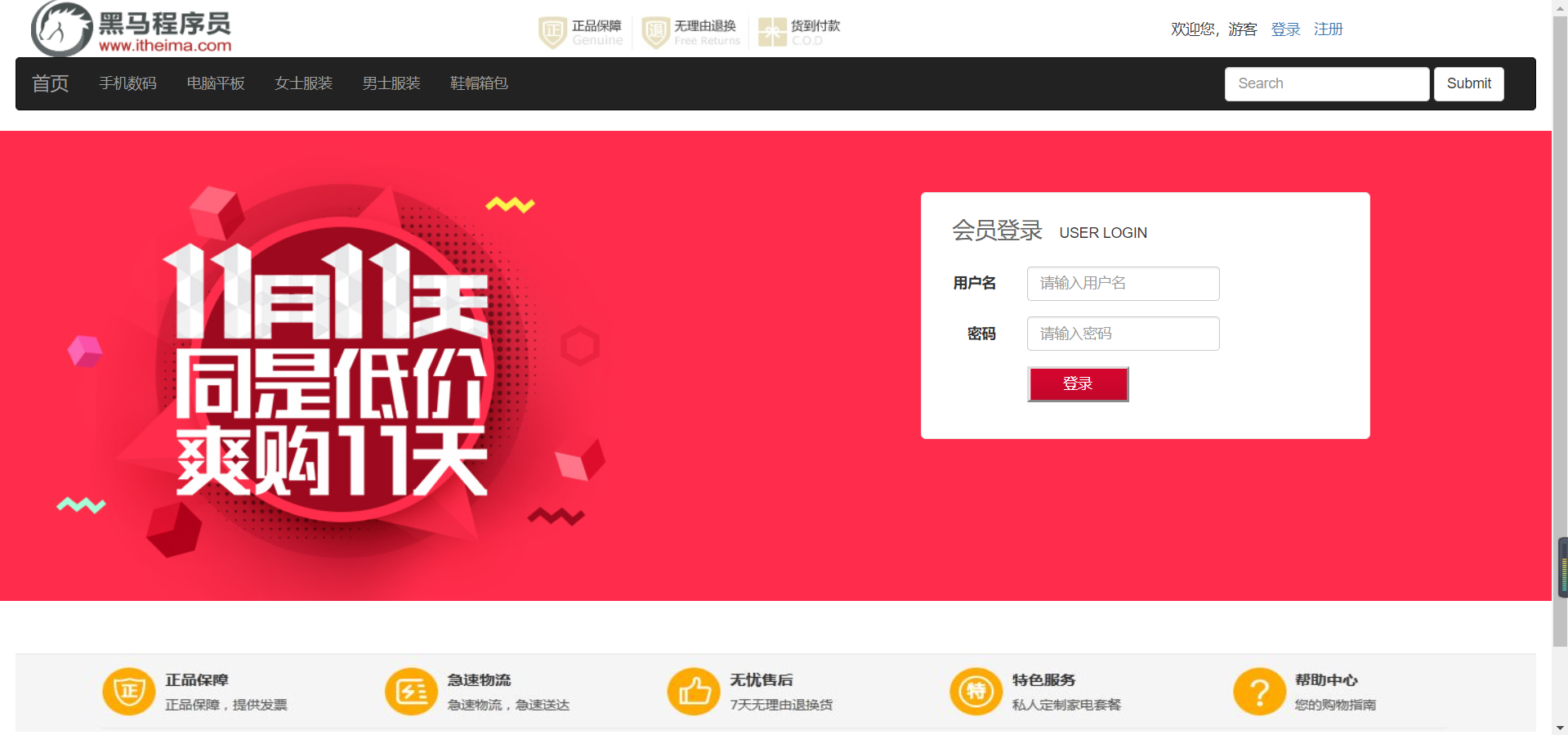


图4-3 用户退出页

（4）显示热门和最新商品

//ProductServlet.java  
/\*\*  
 \*查询最热和最新商品  
 \*/  
private void findHotNewProduct(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 List<Product> hotProductList = productService.findHotProduct();  
 List<Product> newProductList = productService.findNewProduct();  
  
 //02 将2个集合放入到request  
 request.setAttribute("HOT\_PRODUCT\_LIST",hotProductList);  
 request.setAttribute("NEW\_PRODUCT\_LIST",newProductList);  
  
 //03 转发到index.jsp页面中  
 request.getRequestDispatcher("index.jsp").forward(request,response);  
  
}

//ProductServlet.java

/\*\*  
 \* 查询最热商品  
 \*/  
public List<Product> findHotProduct(){  
 return productDao.findHotProduct();  
}  
  
/\*\*  
 \* 查询最新商品  
 \*/  
public List<Product> findNewProduct(){  
  
 return productDao.findNewProduct();  
}

//ProductDao.java

/\*\*  
 \* 查询最热商品且上架的商品  
 \*/  
public List<Product> findHotProduct(){  
 String sql = "select \* from tab\_product where is\_hot=1 and pflag=1 limit 0,12";  
 return jt.query(sql, new BeanPropertyRowMapper<>(Product.class));  
}  
  
  
/\*\*  
 \* 查询最新商品  
 \*/  
public List<Product> findNewProduct(){  
 String sql = "select \* from tab\_product where pflag=1 order by pdate desc limit 0,12";  
 return jt.query(sql, new BeanPropertyRowMapper<>(Product.class));  
}



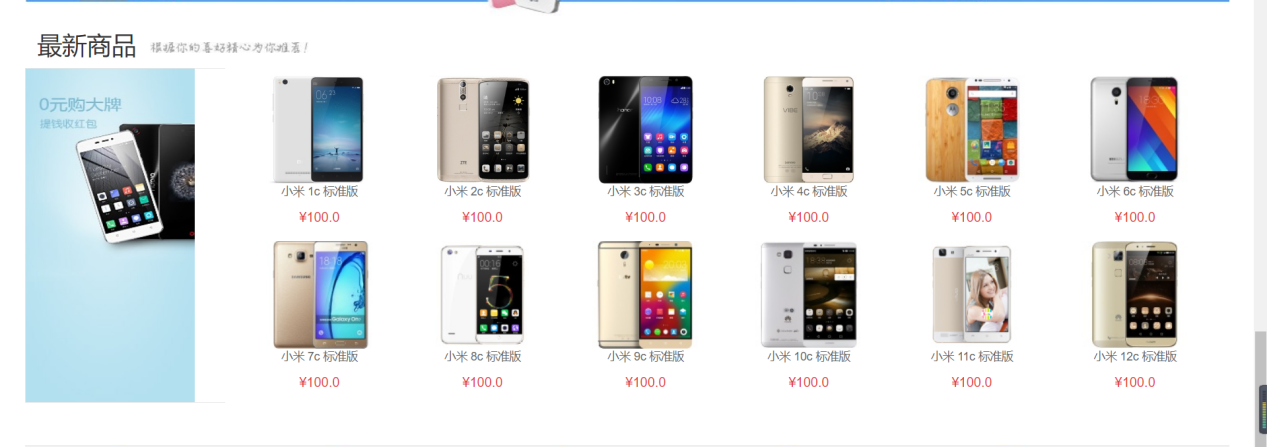


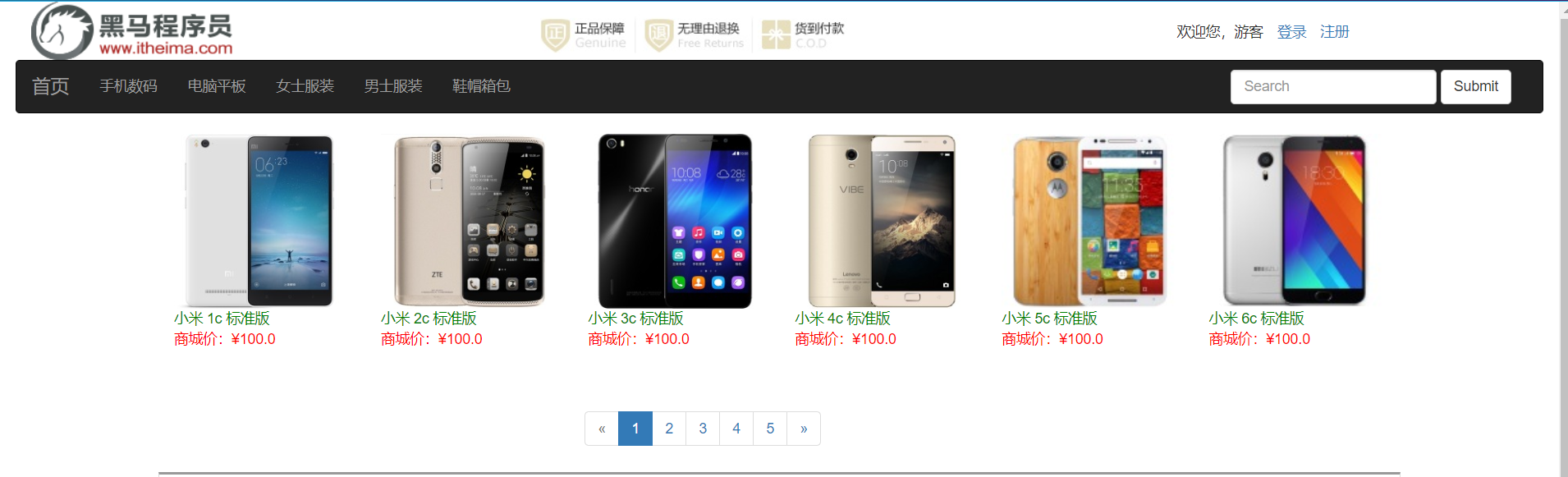
图4-4 显示热门和最新商品页

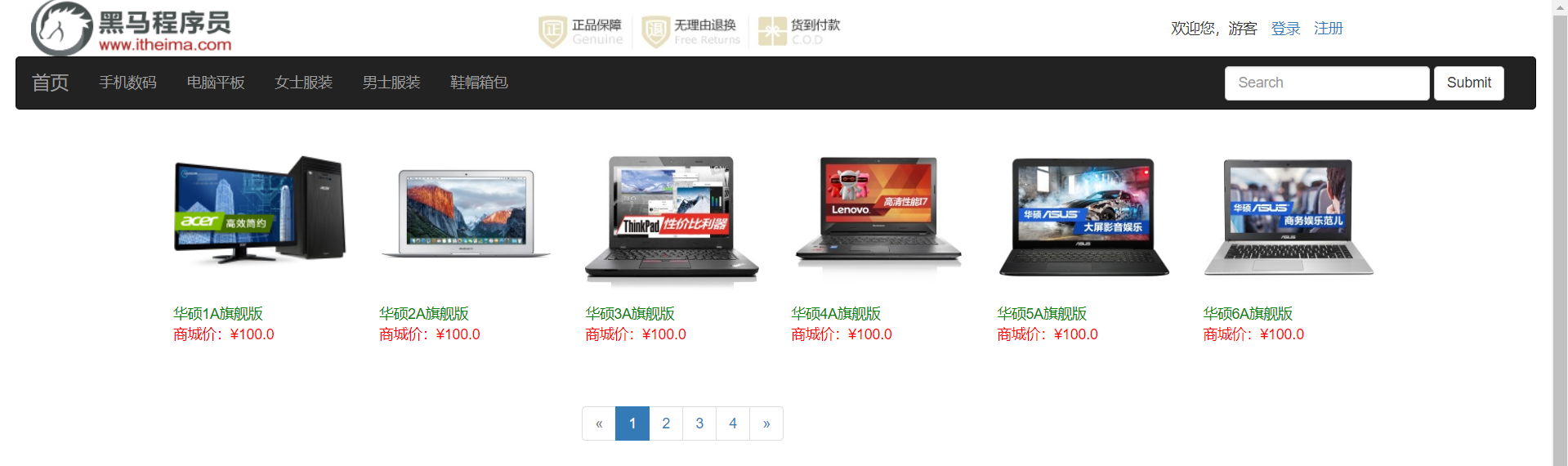
（5）根据类别，分页显示所有商品

//ProductServlet.java  
/\*\*  
 \* 根据类别编号，查询所有商品  
 \*/  
private void findAllProductByCid(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 //01 获取类别编号  
 String strCid = request.getParameter("cid");  
  
 //02 获取当前页号  
 String strCurrentPageNo = request.getParameter("currentPageNo");  
  
 //03 如果没有接收到当前页号，就默认当前页号为1  
 if (strCurrentPageNo == null){  
 strCurrentPageNo = "1";  
 }  
  
 //04 调用Service层  
 PageBean<Product> productPageBean = productService.findAllProductByCid(Integer.parseInt(strCid), Integer.parseInt(strCurrentPageNo));  
  
 //05 将PageBean对象放入到request对象中  
 request.setAttribute("PAGEBEAN",productPageBean);  
  
 //06 转发到product\_list.jsp中  
 request.getRequestDispatcher("/product\_list.jsp").forward(request, response);  
  
}

//ProductService.java  
/\*\*  
 \* 根据类别编号查询所有商品  
 \* @param currentCid 类别编号  
 \* @param currentPageNo 当前页号  
 \* @return  
 \*/  
public PageBean<Product> findAllProductByCid(int currentCid, int currentPageNo) {  
 PageBean<Product> pageBean = new PageBean<Product>();  
  
 //封装当前页号  
 pageBean.setCurrentPageNo(currentPageNo);  
  
 //封装页面数量  
 pageBean.setPageNums(6);  
  
 //封装总记录数  
 int allRecord = productDao.getAllRecord(currentCid);  
 pageBean.setAllRecord(allRecord);  
  
 //封装总页号  
 int allPage = 0;  
 if (pageBean.getAllRecord() % pageBean.getPageNums() == 0){ // 总记录数 % 总页数->30 % 6 = 0  
 allPage = pageBean.getAllRecord() / pageBean.getPageNums();  
 }else {  
 //如果不能整除，则页面总页数要加1  
 allPage = pageBean.getAllRecord() / pageBean.getPageNums() + 1;  
 }  
 pageBean.setAllPage(allPage);  
  
  
 //封装页面内容:计算方法：(start = (页号-1) \* size)  
 int size = pageBean.getPageNums();  
 int start = (pageBean.getCurrentPageNo() - 1) \* size;  
 List<Product> pageContent = productDao.findAllProductByCid(currentCid, start, size);  
 pageBean.setPageContent(pageContent);  
  
 return pageBean;  
  
}

//ProductDao.java  
/\*\*  
 \* 根据类别，分页查询所有商品  
 \* @param cid 类别编号  
 \* @param start 从第几条记录开始查询  
 \* @param size 要查询多少条记录  
 \* @return 这页显示的内容  
 \*/  
public List<Product> findAllProductByCid(int cid, int start, int size) {  
 String sql = "select \* from tab\_product where cid=? limit ?,?";  
 Object[] params = {cid, start, size};  
  
 return jt.query(sql,params, new BeanPropertyRowMapper<>(Product.class));  
}





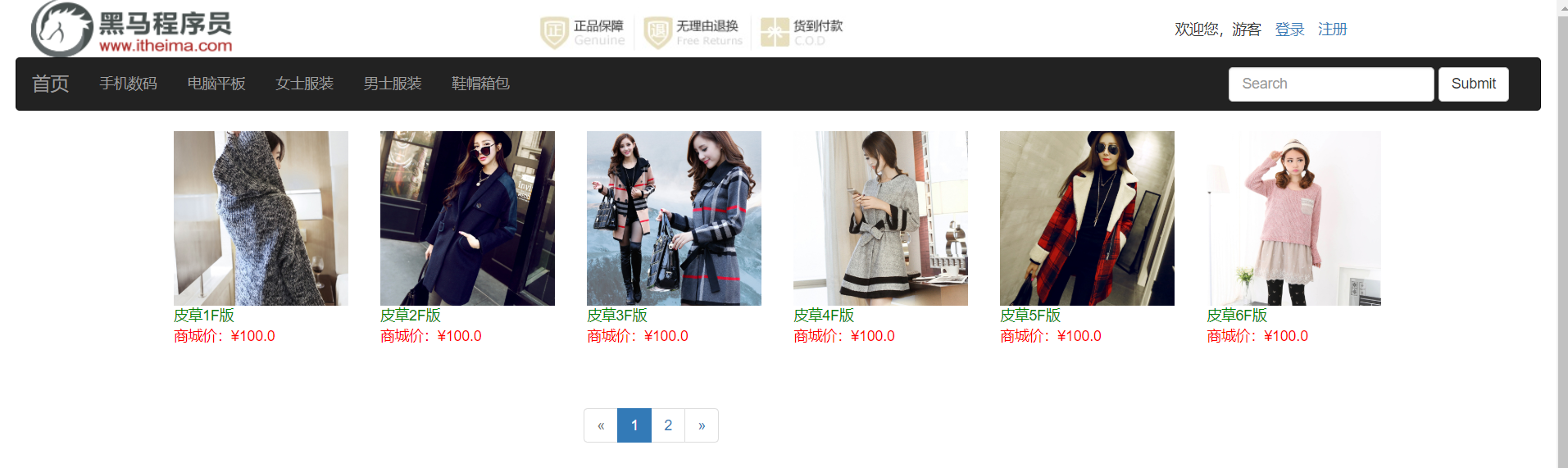


图4-5 分页显示所有商品评论页

（6）根据编号，显示商品详细信息

//ProductServlet.java  
/\*\*  
 \* 根据商品编号，查询所有商品  
 \*/  
private void findProductByPid(HttpServletRequest request, HttpServletResponse response) throws ServletException, IOException {  
 System.out.println("findProductByPid");  
 try {  
 //01 获取商品编号  
 String strPid = request.getParameter("pid");  
  
 //02 创建并调用业务层对象的方法  
 ProductService productService = new ProductService();  
 Product product = productService.findProductByPid(Integer.parseInt(strPid));  
  
 //03 将Product对象放入request域对象  
 request.setAttribute("PRODUCT",product);  
  
 //04 转发到product\_info.jsp中  
 request.getRequestDispatcher("/product\_info.jsp").forward(request, response);  
 } catch (Exception e) {  
 e.printStackTrace();  
 request.setAttribute("MESSAGE","系统出错，请联系管理员");  
 request.getRequestDispatcher("/WEB-INF/message.jsp").forward(request,response);  
 }  
  
}

//ProductService.java  
/\*\*  
 \* 根据商品编号，查询商品的详细信息，包括类别编号和名称  
 \*/  
public Product findProductByPid(int pid) throws Exception {  
 Product product = new Product();  
 Category category = new Category();  
  
 Map<String, Object> map = productDao.findProductByPid(pid);  
  
 //封装Product信息,但cid和cname没有封装  
 BeanUtils.populate(product, map);  
  
 //封装Category信息，cid和cname有封装  
 BeanUtils.populate(category, map);  
  
  
 //让Product封装Category  
 product.setCategory(category);  
  
 return product;  
}

//ProductDao.java  
/\*\*  
 \* 根据商品编号，查询商品的详细信息（含类别编号和名称）  
 \*/  
public Map<String, Object> findProductByPid(int pid) {  
 String sql = "select \* from tab\_category c,tab\_product p where c.cid = p.pid and pid=?";  
 Object[] params = {pid};  
 return jt.queryForMap(sql, params);  
  
}



图4-6 显示商品详细信息信息页

五、测试阶段：用的测试环境、测试方法和测试用例

1.测试环境

黑马商城网站测试的硬件和软件环境如下：

CPU：Intel Core i5-7200U 2.50GHz

内存：8GB

硬盘：500GB

操作系统：Windows 10

应用软件：MySQL 5.7，navicat(数据库管理工具)

2.测试方法

黑盒测试：等价类划分法

3.测试用例

（1）登陆模块测试用例设计

测试登陆模块时，主要看账户是否能够录入表5-1给出了登陆账户输入的等价类划分，表5-2是在此基础上的测试用例设计表。

表5-1 登陆账户输入等价类划分

|  |  |  |
| --- | --- | --- |
| **输入等价类** | **合理等价类** | **不合理等价类** |
| 登陆账户账号密码的输入内容 | 1.账户：admin，密码：123 | 2.与账户：admin，密码：123不匹配的内容 |

表5-2 账户登陆输入测试用例设计

|  |  |  |
| --- | --- | --- |
| **测试数据** | **期望结果** | **覆盖范围** |
| 账户：admin，密码：123 | 输入有效 | 1 |
| 账户：123，密码：123 | 输入无效 | 2 |

表5-3 账户登陆输入测试用例设计

|  |  |
| --- | --- |
| **测试操作** | **期望结果** |
| 登入时输入登陆的账户密码  账户：admin，密码：123 | 登陆成功，跳转到程序主页面index.jsp |
| 登入时输入登陆的账户密码  账户：123，密码：123 | 登录失败 |

（2）注册模块测试用例设计

测试注册模块时，主要看数据库是否和已录入表5-4给出了已注册账户

表5-4 已注册用户的等价类划分

|  |  |  |
| --- | --- | --- |
| **输入等价类** | **合理等价类** | **不合理等价类** |
| 注册账号用户名 | 1.账户：与账户：admin用户名不重复的字符 | 2.与账户：admin用户名重复的字符 |

表5-5 注册 - 等价类划分表

|  |  |  |
| --- | --- | --- |
| **输入条件** | **输入有效** | **输入无效** |
| 用户名 | （1）非空字符 | （9）空字符  （10）已存在的用户名admin |
| 密码 | （2）非空字符 | （11）空字符 |
| 确认密码 | （3）字符等于设置的密码 | （12）空字符  （13）与密码的字符不一 |
| email | （4）非空字符 | （14）空字符 |
| 手机号 | （5）非空字符 | （15）空字符 |
| 姓名 | （6）非空字符 | （16）空字符 |
| 性别 | （7）男/女 | （17）空字符 |
| 出生日期 | （8）非空值 | （18）空字符 |

表5-6 注册测试用例设计

|  |  |  |
| --- | --- | --- |
| **测试数据** | **测试范围** | **期望结果** |
| （1）  用户名：admin  密码：123  确认密码：123  email：zhaojun@abc.com  手机号：13512345678  姓名：赵君  性别：女  出生日期：1997-10-10 | 等价类（10） | 无效 |
| （2）  用户名：abc  密码：123  确认密码：123  email：zhaojun@abc.com  手机号：13512345678  姓名：赵君  性别：女  出生日期：1997-10-10 | 等价类（1） | 有效 |
| （3）用户名：abc  密码：  确认密码：  email：zhaojun@abc.com  手机号：13512345678  姓名：赵君  性别：女  出生日期：1997-10-10 | 等价类（11）（12） | 无效 |
| （4）  用户名：abc  密码：123  确认密码：234  email：zhaojun@abc.com  手机号：13512345678  姓名：赵君  性别：女  出生日期：1997-10-10 | 等价类（13） | 无效 |
| （5）  用户名：abc  密码：123  确认密码：123  email：  手机号：  姓名：赵君  性别：女  出生日期：1997-10-10 | 等价类（14）（15） | 无效 |
| （6）  用户名：abc  密码：123  确认密码：123  email：zhaojun@abc.com  手机号：13512345678  姓名：  性别：  出生日期： | 等价类（16）（17）（18） | 无效 |