服装购物网站

**大数据3班第五组:**

组长:

闫冰洁（Yan Bingjie）

组员:

牛聪（NiuCong）

滕畅（TengChang）

王楠（WangNan）

2019.12.18

**证书**

这份服装购物网站项目报告展示了大数据3班第五组4名同学根据niit的对于项目的相关要求，运用所学知识独立完成的原创项目。

协助者：Rupali、Abhijeet、Manmeet、刘德才

**致谢**

1.是什么激发我们写该项目

首先，就技术而言，类似线上服装销售网站的代码有许多，如果我们的制作遇到了瓶颈，还会有大量的资料可供我们参考；然后，就普及度而言，线上服装销售网站更容易被普及，这样能有效提高我们在有限时间内可获得的用户反馈数量和质量，为我们对该项目的维护和升级提供了更大的机率。

2.谁帮助了我们

在艰难的项目制作的过程中，我们获得过许多人的帮助，刘德才老师、Rupali老师、Abhijeet老师以及所有为我们的项目提过想法、出过主意、提供帮助的同学和亲朋都给我们在技术和情感上提供了极大的帮助。

3.表示感谢

我们对所有曾帮助我们攻克难关、陪伴我们一起度过这段枯燥又有趣的日子的所有人表示诚挚的感谢。

4.简单介绍开发本项目的过程

以我们的组内讨论为轴来看我们对本项目的开发，我们共进行了四次重大的讨论。第一次讨论，我们确定了分工，组长按每一项任务的权重安排不同数量的组员去完成任务，给我们之后大致努力的方向指明了方向；第二次，距离第一次讨论过去了将近三周，我们完成并合并了我们所有的项目内容，并在组内进行评价和改进建议，然后我们进行了下一轮的整改；第三次，距离第二次讨论过去了一周，我们再次将所有内容进行合并和讨论，确定了项目的完成；第四次讨论，我们确定了ppt的内容和展示。

**使用技术说明**

操作系统：Windows、Linux

数据库：Mysql、HBase

服务器：Tomcat

数据库连接驱动：Java web 集成开发环境

数据分析：Hadoop MapReduce

**系统分析(需求)**

1. **功能需求：**

|  |
| --- |
| **用户注册、登录** |
| **查看商品主分类** |
| **搜索所需商品** |
| **加入购物车** |
| **购买商品** |
| **分析数据** |

1.1功能块描述

(1)注册功能。

顾客首先要注册为网上商城的用户。注册时只要填写登录用户名、密码2项信息即可。注册后，用户可继续如实填写详细个人信息及收货人信息，同时可修改密码、查询及修改订单。

(2)选择产品功能。

顾客浏览网上商城，将自己需求的产品放入到购物车中,可连续添加商品。

(3)管理购物车。

顾客选择完商品后可进入购物车页面，查看自己要购买的商品，可修改某一商品数

量、取消购买某商品和清空整个购物车。  
(4)订单功能。

顾客确定购物车中的商品后提交订单，如顾客已填写收货人信息，则页面显示该信息并由顾客确认。如尚未填写则显示相应表单请其填写，系统记录顾客提交的收货人信息以便其下次购物时使用。顾客提交订单后可在网上商城查询该订单，并可对尚未处理的订单进行取消、修改等操作。

1. **对自己的日志进行分析**

2.1 最受欢迎的商品Top5

商品的点击量可以反映一个商品的受欢迎程度，因此我们对最受欢迎的商品的定义是在所有商品中点击量最多的商品Top5，利用Map统计出所有被点击过的商品，利用Reduce统计出所有商品的点击次数，最后在cleanup阶段统计出Top5，输出到结果文件。

2.2 最畅销的商品Top5

商品的销售量可以反映一个商品的畅销程度，利用Map统计出所有被购买过的商品，利用Reduce统计出所有商品的购买次数，最后在cleanup阶段统计出Top5，输出到结果文件。

2.3 购买率最高的商品Top3

商品的购买率可以反映用户群体对商品的忠诚度和对商品描述的接受程度，利用Map统计出所有商品，在Reduce中统计出每种商品的点击数和购买数，通过公式购买率=购买数/(点击数+购买数)来进行记录，在cleanup阶段统计出Top3。

2.4 最多购买组合Top5

通过对购买组合的分析可以对商品进行捆绑销售或者特殊促销的手段来提高销量，利用Map统计所有用户各自的购买商品情况，在Reduce阶段统计所有商品组合，最后在cleanup阶段统计出Top5

1. **网络日志分析需求**

2.1 最受欢迎的商品Top5

商品的点击量可以反映一个商品的受欢迎程度，因此我们对最受欢迎的商品的定义是在所有商品中点击量最多的商品Top5，利用Map统计出所有被点击过的商品，利用Reduce统计出所有商品的点击次数，最后在cleanup阶段统计出Top5，输出到结果文件。

2.2 最畅销的商品Top5

商品的销售量可以反映一个商品的畅销程度，利用Map统计出所有被购买过的商品，利用Reduce统计出所有商品的购买次数，最后在cleanup阶段统计出Top5，输出到结果文件。

2.3 购买率最高的商品Top3

商品的购买率可以反映用户群体对商品的忠诚度和对商品描述的接受程度，利用Map统计出所有商品，在Reduce中统计出每种商品的点击数和购买数，通过公式购买率=购买数/(点击数+购买数)来进行记录，在cleanup阶段统计出Top3。

2.4 最多购买组合Top5

通过对购买组合的分析可以对商品进行捆绑销售或者特殊促销的手段来提高销量，利用Map统计所有用户各自的购买商品情况，在Reduce阶段统计所有商品组合，最后在cleanup阶段统计出Top5

**详细介绍**

1. 软件定位

本网站的定位在于为用户提供了更加方便、快速的购买衣服的方式。即在空间上突破了地域上的局限性，使用户在衣服种类的挑选上拥有更多选择的权力；在时间上，省去了不断去不同的商场、不同的门店挑选衣服、试衣服的时间，为人们的生活增添更多的乐趣。

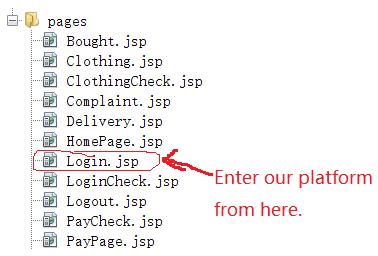
1. 软件的基本功能

本网站是面向用户的平台，包括:顾客在线注册、在线登录、浏览商品页面、购物、提交订单、付款、查看订单、投诉等操作。

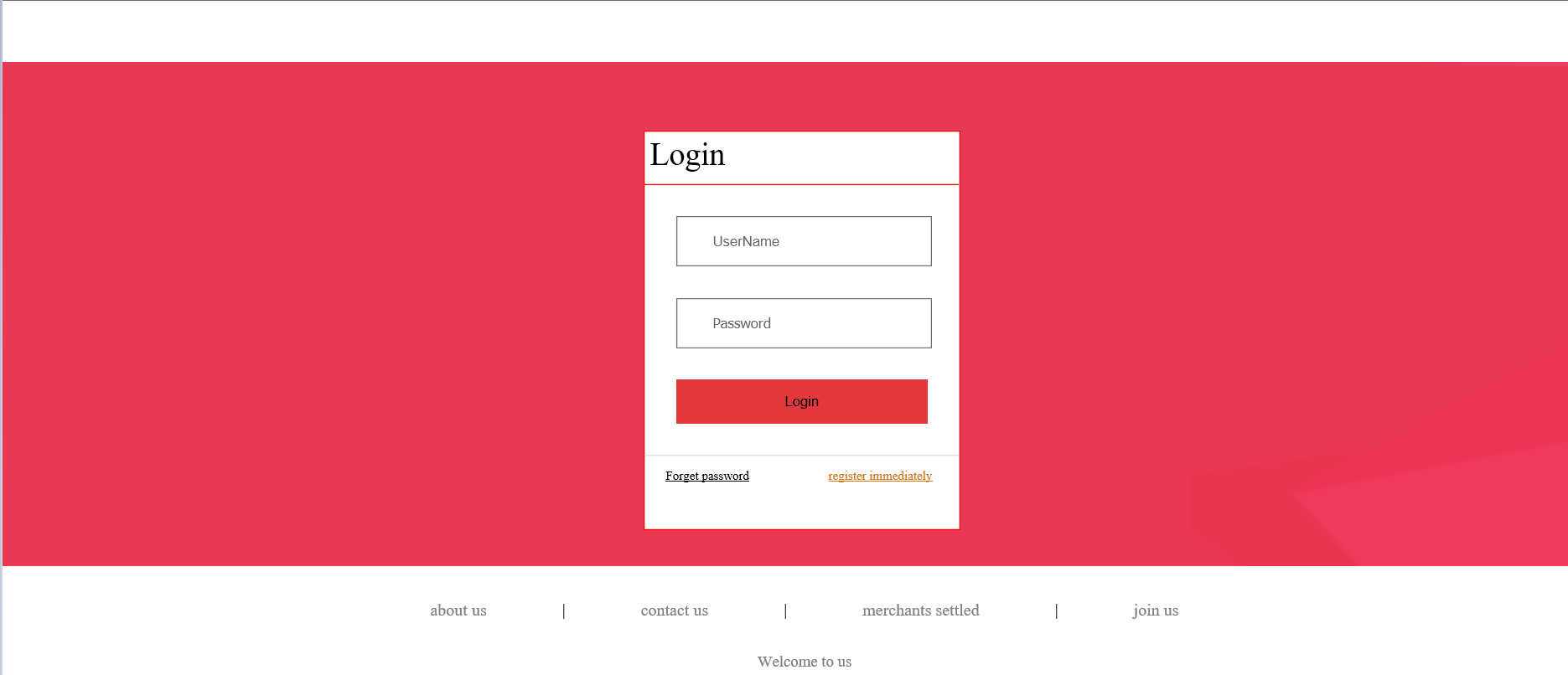
1. 运行网站的方法

使用netbeans IDE软件运行程序即可。

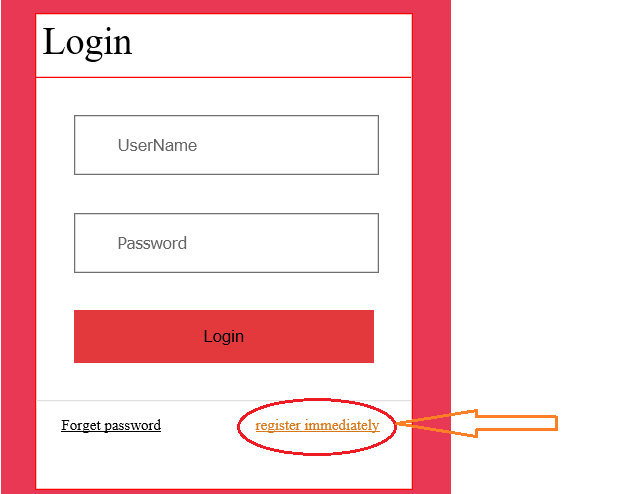
1. 使用说明
2. 从/Project/webpage/first/pages中获得Login.jsp，右键运行该.jsp文件



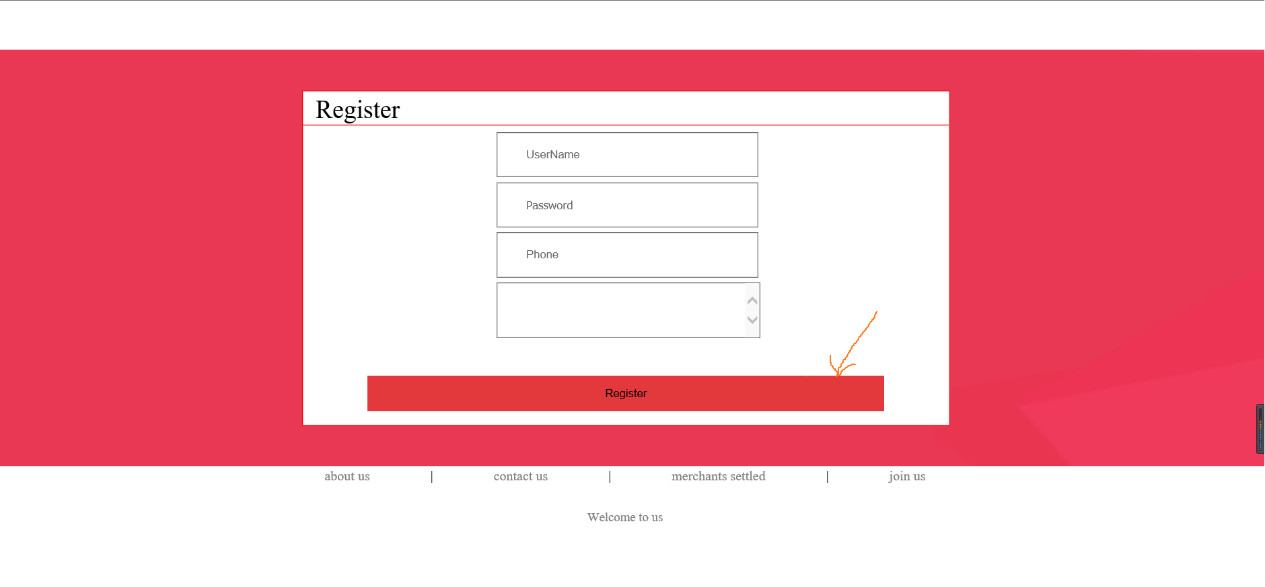
1. 运行以后出现下列界面



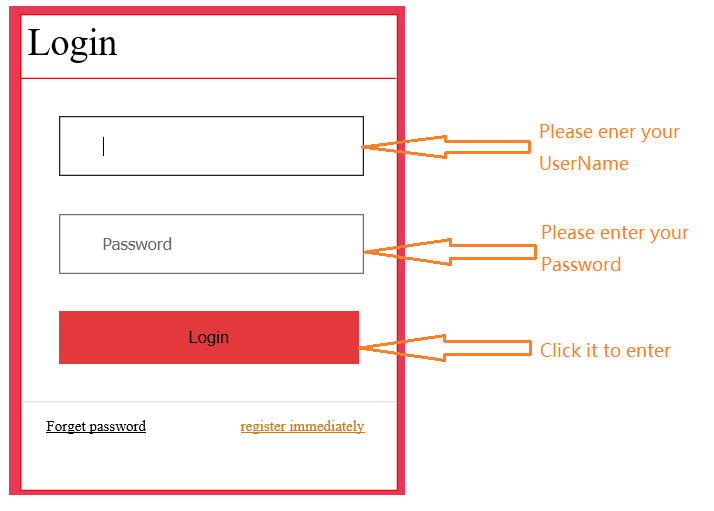
1. 如果您是第一次登录我们的网站，请点击右下角“Register immediately”进行用户注册（若已经注册则忽略此步）



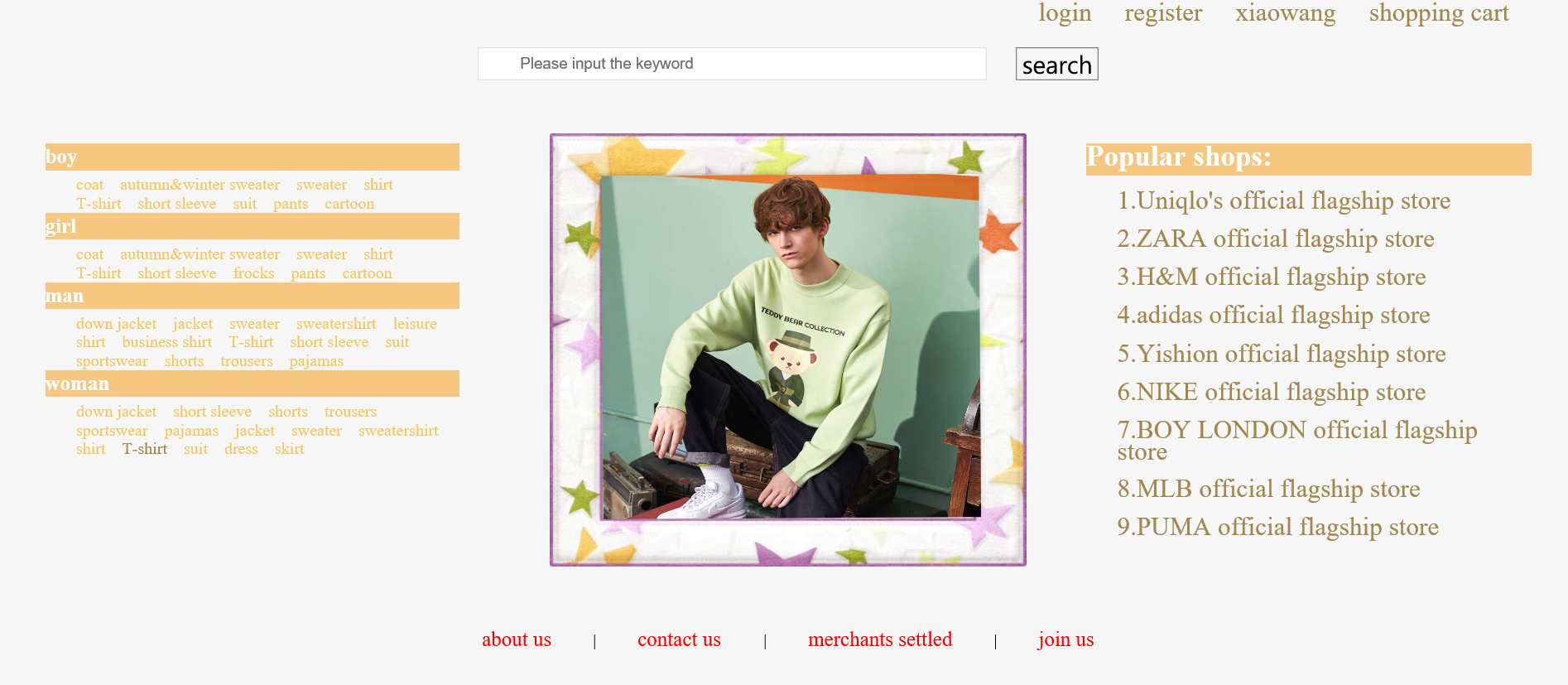
1. 填写用户注册界面后点击“Register”按钮回到Login界面进行登录（若已经注册则忽略此步）



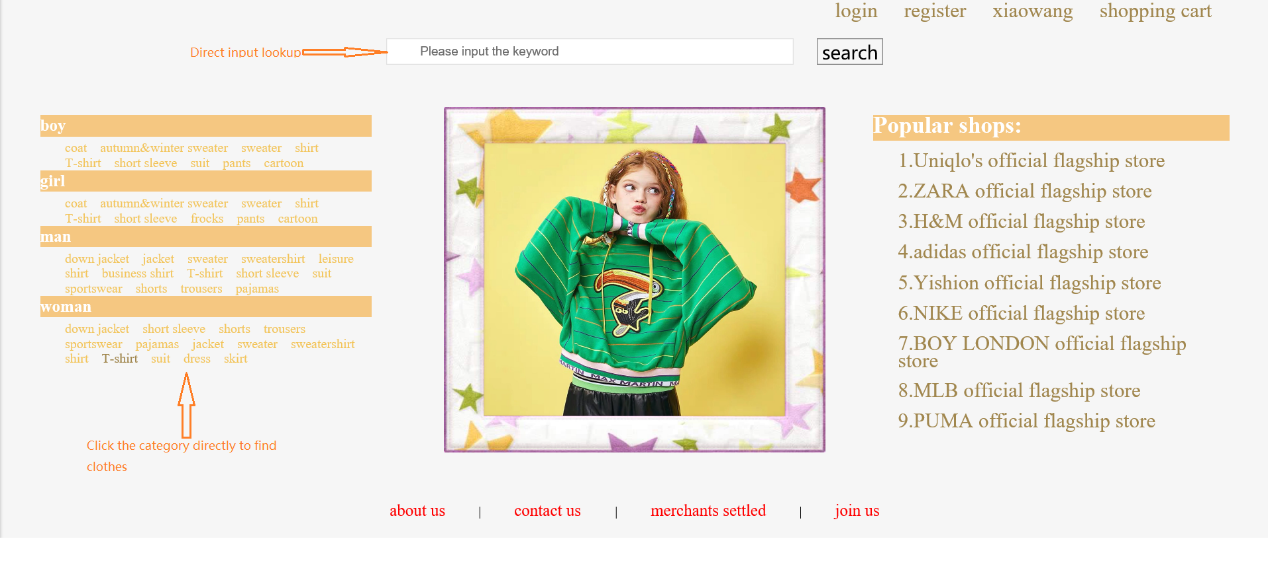
1. 输入账户和密码，然后点击“Login”按钮进入本网站。



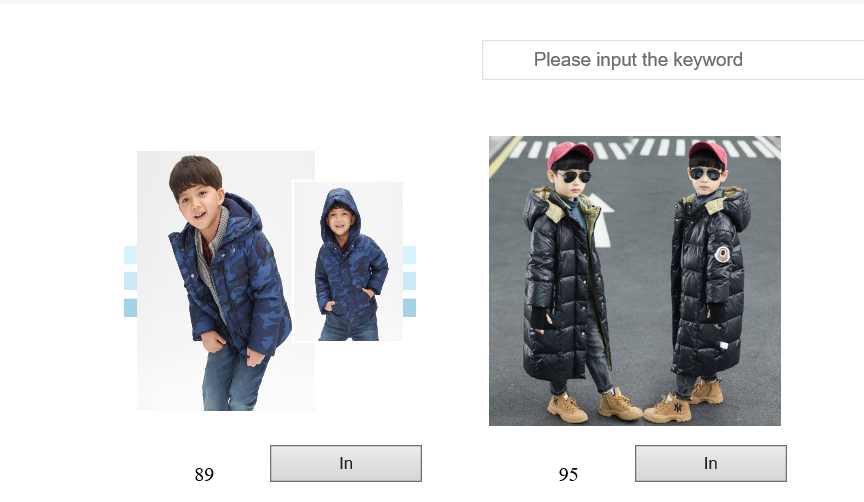
1. 进入本网站后出现下列页面：



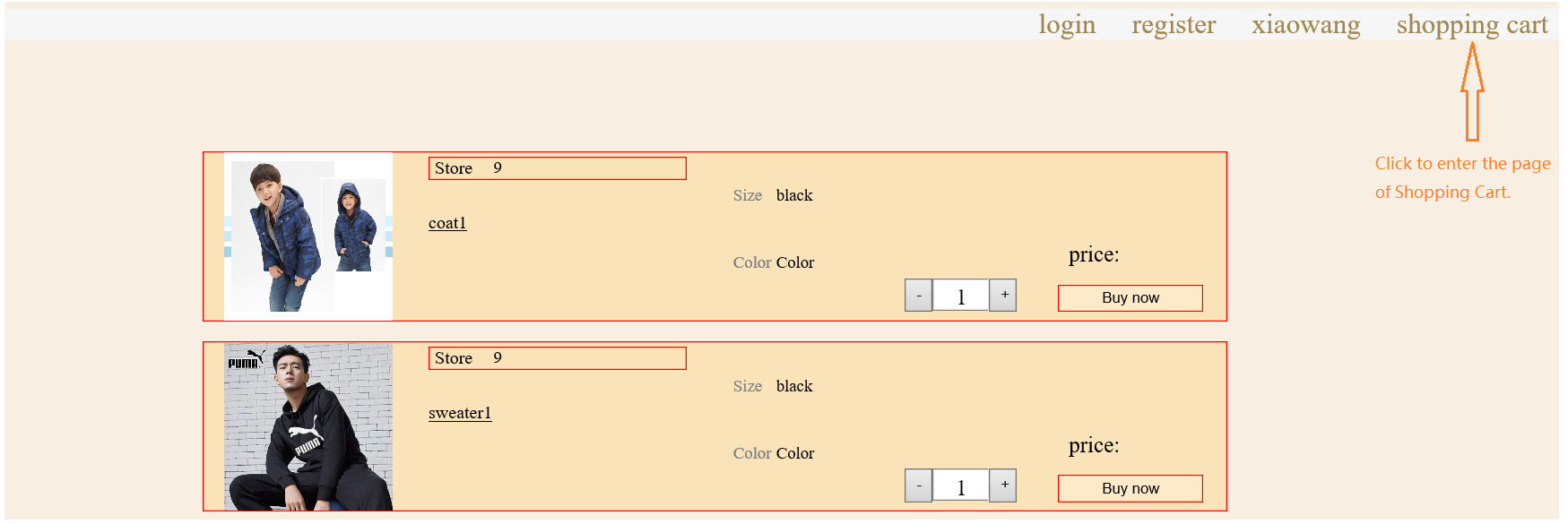
1. 本网站中查找商品的方式有两种方式：



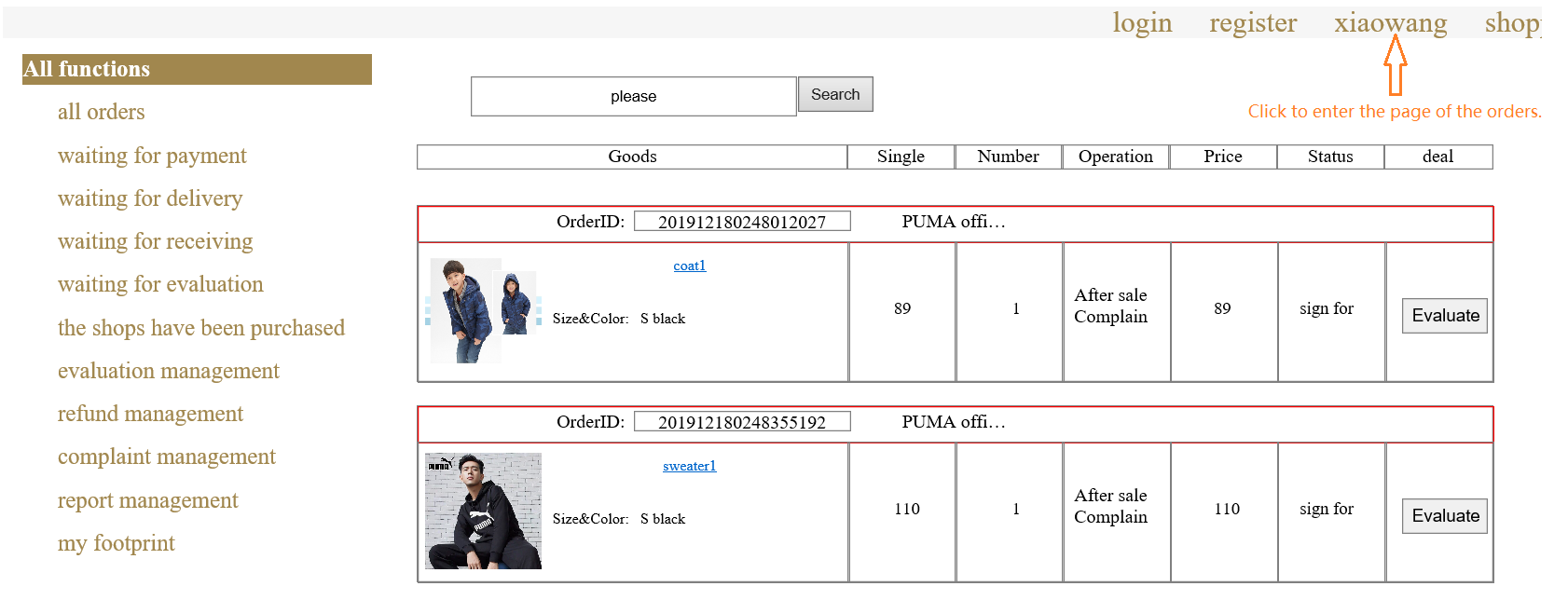
1. 进入商品界面浏览商品



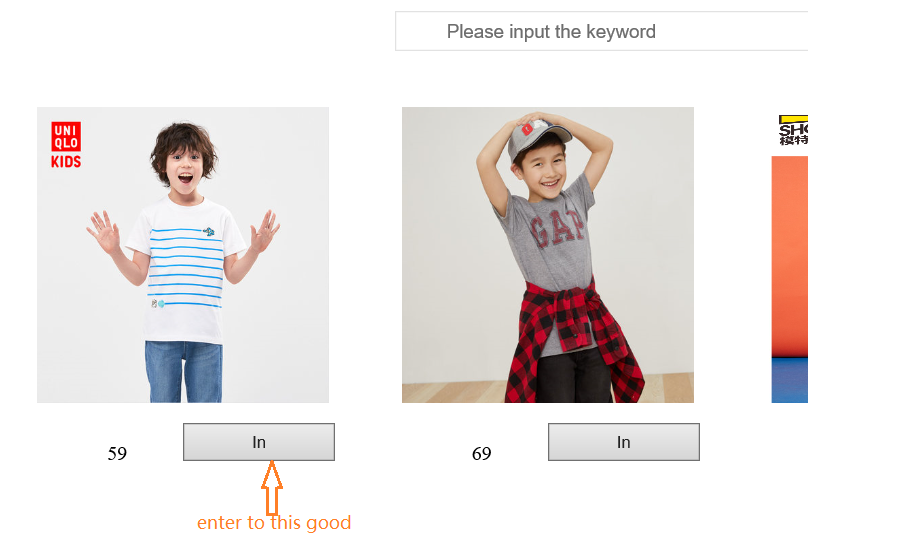
1. 进入购物车查看收藏到购物车的商品



1. 查看已经购买的商品（未收货和已收货）



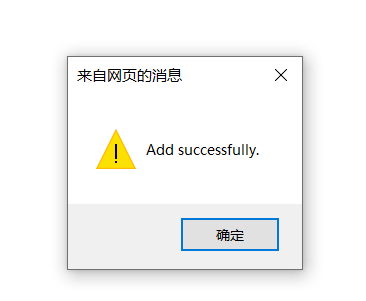
1. 从浏览商品页面进入商品页面



1. 点击加入购物车



1. 成功加入购物车，点击”确定”返回



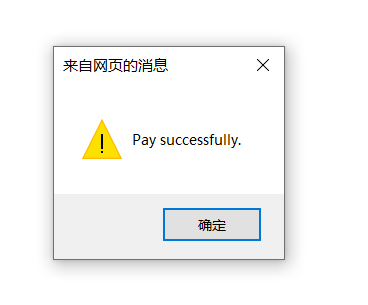
1. 点击购买



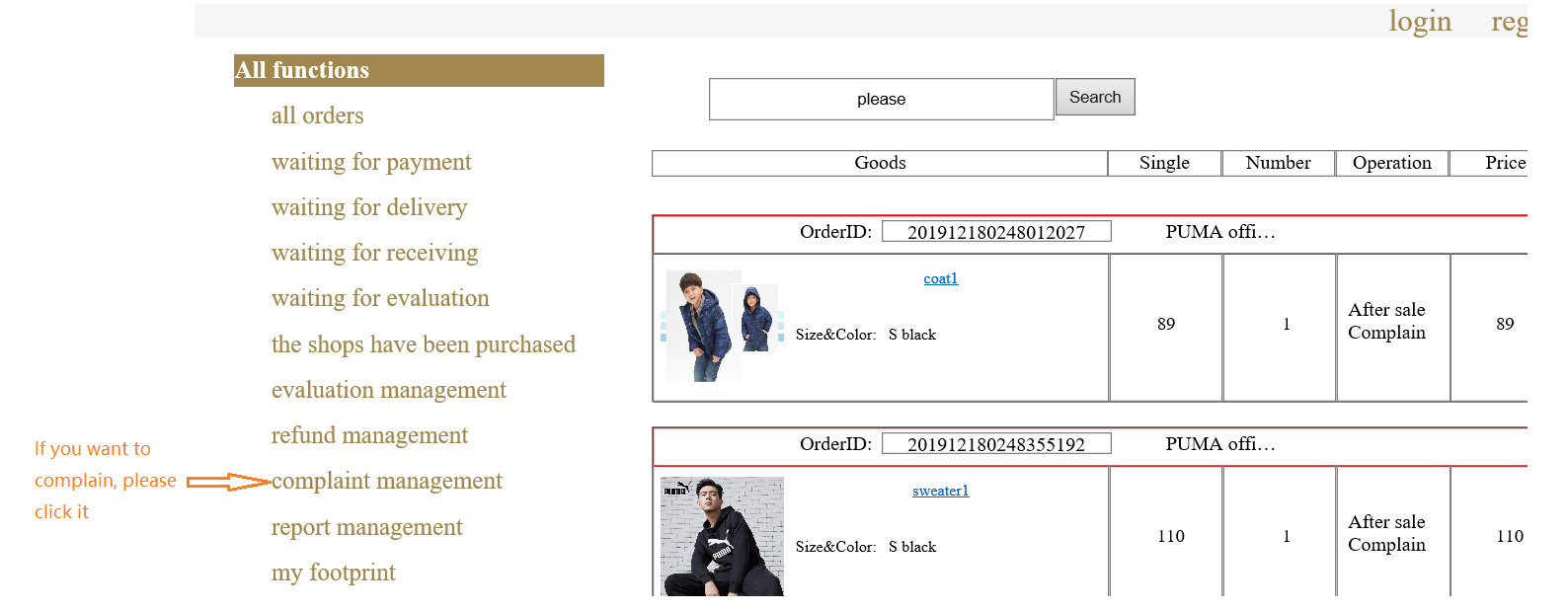
1. 付钱



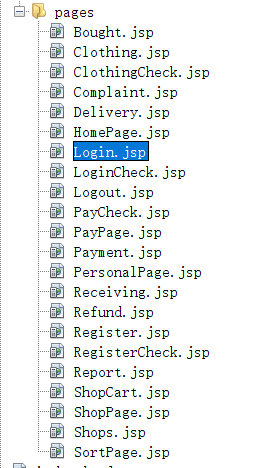
1. 购买成功，点击“确定”返回



1. 如果您要投诉，请点击页面中的按钮

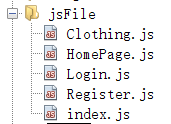


1. 代码目录结构说明
2. 用于显示页面的目录



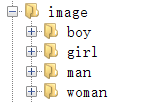
上图为用于显示页面的目录，其中，Bought.jsp为订单界面，详细地显示了用户购买的每一个订单的信息；Clothing.jsp为商品界面，显示了某一商品的详细信息；ClothingCheck.jsp为Clothing.jsp做承接，通过日志记录了用户对该商品的操作，比如加入购物车、直接购买等；Complaint.jsp显示了用户对某一产品或对本网站的一些历史意见或建议；Delivery.jsp显示了用户的订单状况；HomePage.jsp作为用户的首页；Login.jsp为登录界面；LoginCheck.jsp为Login.jsp做承接，通过日志记录了用户登录的状况，包括用户登录成功、登陆失败的详细信息；PayCheck.jsp为支付的确认界面，其中通过日志记录了用户支付成功和支付失败的信息；PayPage.jsp为支付密码的输出界面，其中显示了支付的金额等必要的信息；Payment.jsp显示了我们每一单支出的情况；Receiving.jsp为物流信息页面，为用户显示了商品的运送进度；Refound.jsp为商品退还界面，为用户显示了退换商品的详细的历史记录；Register.jsp为注册界面，用户必须输入用户名、密码等必要信息才可以登录成功；RegisterCheck.jsp为Register.jsp做承接，通过日志记录了用户注册成功的信息；Report.jsp为用户反馈界面，显示了用户对本网站的每一次反馈；ShopCart.jsp为购物车界面，详细地显示了用户加入购物车的商品的信息；ShopPage.jsp为商铺页面，显示了商铺的所有宝贝；SortPage.jsp为浏览商品页面，通过日志记录用户对每一个商品的浏览情况。

1. 用于为显示页面做事件修饰的目录



上图为用于为显示页面做事件修饰的目录，其中，Clothing.js为Clothing.jsp的点击事件集合，其中包括商品数量的增减、尺码和颜色的选择，以及加入购物车和直接购买的操作；HomePage.js为HomePage.jsp的点击事件的集合；Login.js为Login.jsp的点击事件的集合，其中包括立即注册、忘记密码和成功登录进入下一页面的操作；Register.js为Register.jsp的点击事件的集合，其中包括对用户名、密码的判空、手机号码的判错、注册成功跳转登陆页面等操作；index.js为首页图片轮播的操作的集合。

1. 用于保存商品图片的目录



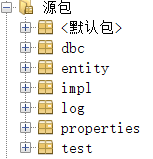
上图为商品图片的集合，均集中在image包下。

1. 用于为显示页面美化做修饰的目录



用于为所有页面做整体布局和美化。

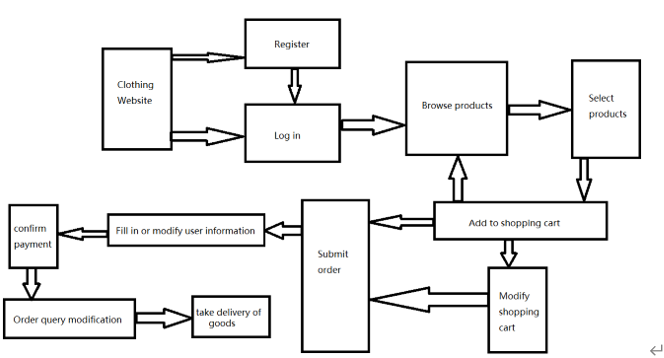
1. 用于实现连接数据库、调用数据库数据显示在网站上的类的目录



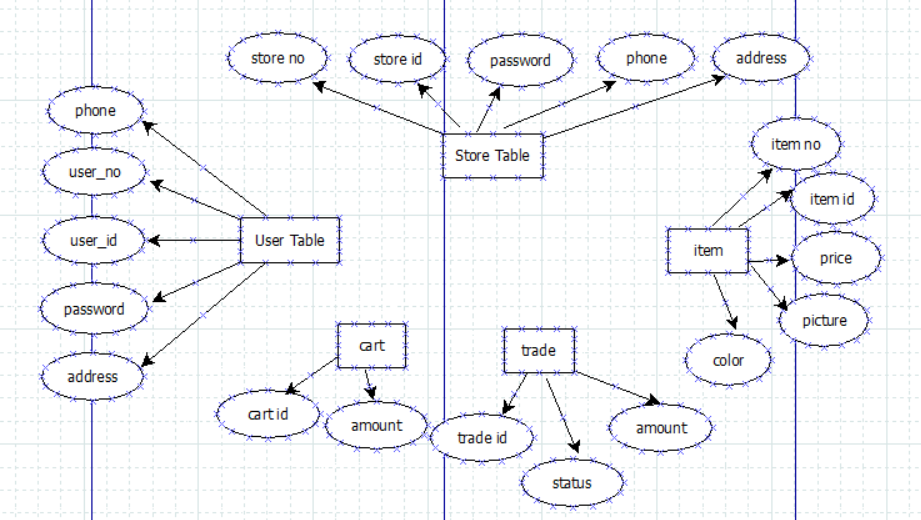
上图为用于实现连接数据库、调用数据库数据显示在网站上的类包集合的目录，其中，dbc包为连接数据库的类包；entity用于对单个商品、订单、购物车、商铺进行数据的增删改查；impl包用于对整体的商品、订单、购物车、商铺进行查找和整体数据的输入；log包用于保存log文件信息；properties包用于保存数据库用户、密码等信息；test包中存放用来测试网站的代码。

**流程图**

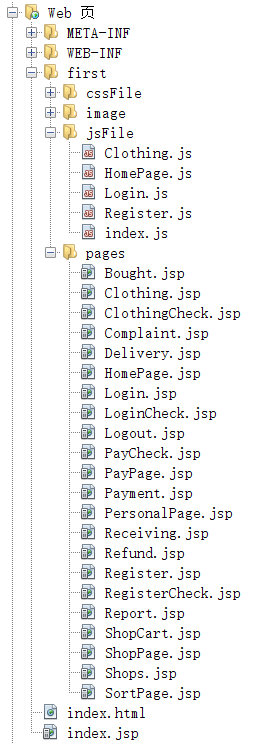
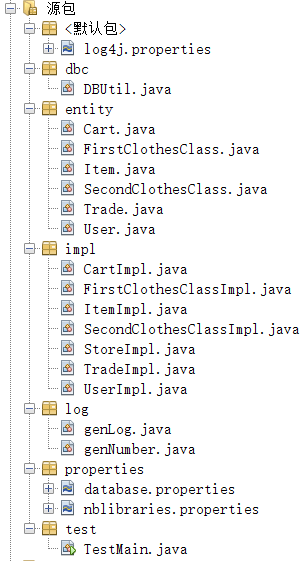
**1.客户端**



**2.为数据库设计的ER流程图**



**类图**



**任务分配**

|  |  |  |  |
| --- | --- | --- | --- |
| **任务名称** | **开始时间** | **完成时间** | **负责人** |
| 需求分析 | 2019/11/02 | 2019/11/04 | 滕畅 王楠 |
| 寻找网络日志 | 2019/11/06 | 2019/11/08 | 闫冰洁 |
| 网页设计 | 2019/11/06 | 2019/11/25 | 滕畅 王楠 |
| 数据库 | 2019/11/10 | 2019/11/25 | 牛聪 |
| 数据库连接配置 | 2019/11/12 | 2019/11/25 | 闫冰洁 |
| 日志分析 | 2019/12/01 | 2019/12/17 | 闫冰洁 |
| 项目报告 | 2019/12/17 | 2019/12/18 | 滕畅 王楠 |
| 展示PPT | 2019/12/18 | 2019/12/18 | 牛聪 |

**重要代码**

/\*Log file\*/

public static void Log(String user\_id, String ip\_address, String action\_id, String category\_id1, String category\_id2,

String item\_id, String seller\_id) { // timestamp, user\_id, ip\_address, action\_id, first\_category\_id, second\_category\_id, item\_id, seller\_id

Logger log = Logger.getLogger(genLog.class.getName());

SimpleDateFormat df = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

String time = df.format(new Date());

log.info(time + "," + user\_id + "," + ip\_address + "," + action\_id + "," + category\_id1 + "," + category\_id2 + "," + item\_id + "," + seller\_id);

}

/\*User operation file\*/

public class UserImpl {

public static int add(User user) throws Exception {

String sql = "insert into user(user\_no,user\_id,user\_passwd,user\_phone, user\_addr) values(?,?,?,?,?)";

if(user.getUser\_no() == null || user.getUser\_no().length() == 0) {

user.setUser\_no(genNumber.getno());

}

try (Connection conn = getConnection();) {

return exceuteUpdate(conn, sql, new Object[]{user.getUser\_no(), user.getUser\_id(), user.getUser\_passwd(), user.getUser\_phone(), user.getUser\_addr()});

}

}

public static User findIdUser(String name) throws Exception {

String sql = "select \* from user where user\_id=?";

User use = null;

try (Connection conn = getConnection();

PreparedStatement pst = conn.prepareStatement(sql)) {

pst.setString(1, name);

try (ResultSet rs = pst.executeQuery()) {

if (rs.next()) {

use = new User(rs.getString("user\_no"), rs.getString("user\_id"), rs.getString("user\_passwd"), rs.getString("user\_phone"), rs.getString("user\_addr"));

}

return use;

}

}

}

public static User findNoUser(String name) throws Exception {

String sql = "select \* from user where user\_no=?";

User use = null;

try (Connection conn = getConnection();

PreparedStatement pst = conn.prepareStatement(sql)) {

pst.setString(1, name);

try (ResultSet rs = pst.executeQuery()) {

if (rs.next()) {

use = new User(rs.getString("user\_no"), rs.getString("user\_id"), rs.getString("user\_passwd"), rs.getString("user\_phone"), rs.getString("user\_addr"));

}

return use;

}

}

}

}

/\*PayCheck file\*/

<body>

<%

String pass = request.getParameter("pass");

User u = (User) session.getAttribute("user");

genLog log = new genLog();

// genIp ip = new genIp();

String item\_id = (String)session.getAttribute("item\_id");

String store = StoreImpl.findById(item\_id);

Cookie[] co = request.getCookies();

if (!u.getUser\_passwd().equals(pass)) {

log.Log(u.getUser\_id(), InetAddress.getLocalHost().getHostAddress(), "08", co[0].getName(), co[1].getName(), item\_id, store);

out.println("<script>alert(\"Password is incorrect.\")</script>");

out.println("<script> window.history.back(-1); </script>");

} else {

log.Log(u.getUser\_id(), InetAddress.getLocalHost().getHostAddress(), "09", co[0].getName(), co[1].getName(), item\_id, store);

Item item = ItemImpl.findIdItem(item\_id);

String item\_no = item.getItem\_no();

List<Cart> c = CartImpl.findCartUser(u.getUser\_no()); //通过item\_no找到该cart

int i = 0;

while (i < c.size()) {

if (c.get(i).getItem\_no().equals(item\_no)) {

break;

} else {

i++;

}

}

Trade trade = new Trade(null, c.get(i).getItem\_amount(), item\_no, c.get(i).getUser\_no(), c.get(i).getItem\_size(), c.get(i).getItem\_color());

int t = TradeImpl.add(trade);

if(c.get(i).getCart\_no()!=null){

CartImpl.deleteCartId(c.get(i).getCart\_no());

}

out.println("<script>alert(\"Pay successfully.\")</script>");

out.println("<script> window.location.href=\"HomePage.jsp\"; </script>");

}

%>

</body>

/\*connect database\*/

public static void init() throws IOException {

Properties params = new Properties();

String configFile = "properties/database.properties";

InputStream is = DBUtil.class.getClassLoader().getResourceAsStream(configFile);

params.load(is);

driver = params.getProperty("driver");

url = params.getProperty("url");

username = params.getProperty("username");

password = params.getProperty("password");

}

/\*Update database\*/

public static int exceuteUpdate(Connection conn, String preparedSql, Object[] param) throws SQLException {

int num;

try (Connection connection = conn;

PreparedStatement pstmt = connection.prepareStatement(preparedSql);) {

if (param != null) {

for (int i = 0; i < param.length; i++) {

pstmt.setObject(i + 1, param[i]);

}

}

num = pstmt.executeUpdate();

return num;

}

}

/\*BestSellMapReduce\*/

static class MapDistinctTask extends Mapper<LongWritable, Text, Text, Text> {

@Override

protected void map(LongWritable key, Text value, Context context)

throws IOException, InterruptedException {

String[] ss = value.toString().split(",");

// 3 action\_id 6 item\_id 7 store\_id

if("1".equals(ss[3])) { // buy

context.write(new Text("1"), new Text(ss[6]));

}

}

}

static class ReduceDistinctTask extends Reducer<Text, Text, Text, Text> {

TreeMap<Integer, String> tree = new TreeMap<Integer, String>();

@Override

protected void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

int sum = 0;

tree.clear();

List<String> s = new ArrayList();

for (Text val : values) {

String ss = val.toString();

s.add(ss);

}

Collections.sort(s);

for (int i = 0; i < s.size(); i++) {

int j = i;

for (; j < s.size() && s.get(j).equals(s.get(i)); j++) ;

tree.put(j - i + 1, s.get(i));

if (tree.size() > 5) {

tree.remove(tree.firstKey());

} i=j-1;

}

Iterator<Integer> it1 = tree.keySet().iterator();

while (it1.hasNext()) {

int kk = it1.next();

String vv = tree.get(kk);

context.write(new Text(vv), new Text(Integer.toString(kk)));

}

}

}

/\*BuyRateMapReduce\*/

static class MapDistinctTask extends Mapper<LongWritable, Text, Text, Text> {

@Override

protected void map(LongWritable key, Text value, Context context)

throws IOException, InterruptedException {

String[] ss = value.toString().split(",");

// 4 action\_id 6 item\_id 7 store\_id

context.write(new Text(ss[6]), new Text(ss[3]));

}

}

static class ReduceDistinctTask extends Reducer<Text, Text, Text, Text> {

TreeMap<Double, String> tree = new TreeMap<Double, String>();

@Override

protected void cleanup(Context context) throws IOException, InterruptedException{

Iterator<Double> it1 = tree.keySet().iterator();

while (it1.hasNext()) {

Double kk = it1.next();

String vv = tree.get(kk);

context.write(new Text(vv), new Text(kk.toString()));

}

}

@Override

protected void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

int sum0 = 0, sum1 = 0;

for(Text t: values) {

if("1".equals(t.toString())) sum1 ++;

else sum0 ++;

}

tree.put(1.0\*sum1/(sum1+sum0), key.toString());

}

}

/\*CombineSoldTop5\*/

static class MapDistinctTask extends Mapper<LongWritable, Text, Text, Text> {

@Override

protected void map(LongWritable key, Text value, Context context)

throws IOException, InterruptedException {

String[] s = value.toString().split(",");

if ("1".equals(s[3])) { // buy

context.write(new Text(s[1]), new Text(s[6]));

}

}

}

static class ReduceDistinctTask extends Reducer<Text, Text, Text, Text> {

TreeMap<Integer, String> tree = new TreeMap<Integer, String>();

@Override

protected void cleanup(Context context) throws IOException, InterruptedException{

Iterator<Integer> it1 = tree.keySet().iterator();

while (it1.hasNext()) {

Integer kk = it1.next();

String vv = tree.get(kk);

context.write(new Text(vv), new Text(kk.toString()));

}

}

@Override

protected void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

List<String> ls = new ArrayList();

List<String> res = new ArrayList();

// tree.clear();

for (Text t : values) {

ls.add(t.toString());

}

int tot = ls.size();

for (int i = 0; i < tot; i++) {

for (int j = i + 1; j < tot; j++) {

String ss = ls.get(i) + "+" + ls.get(j);

if (ls.get(i).compareTo(ls.get(j)) > 0) {

ss = ls.get(j) + "+" + ls.get(i);

}

res.add(ss);

}

}

Collections.sort(ls);

for (int i = 0; i < res.size(); i++) {

int j;

for (j = i; j < res.size() && res.get(i).equals(res.get(j)); j++) ;

tree.put(j - i + 1, res.get(i));

if (tree.size() > 5) {

tree.remove(tree.firstKey());

} i=j-1;

}

}

}

/\*MostPopularItem\*/

static class MapDistinctTask extends Mapper<LongWritable, Text, Text, Text> {

@Override

protected void map(LongWritable key, Text value, Context context)

throws IOException, InterruptedException {

String[] ss = value.toString().split(",");

// 4 action\_id 6 item\_id 7 store\_id

if("0".equals(ss[3])) { // click

context.write(new Text("1"), new Text(ss[6]));

}

}

}

static class ReduceDistinctTask extends Reducer<Text, Text, Text, Text> {

TreeMap<Integer, String> tree = new TreeMap<Integer, String>();

@Override

protected void reduce(Text key, Iterable<Text> values, Context context)

throws IOException, InterruptedException {

int sum = 0;

tree.clear();

List<String> s = new ArrayList();

for (Text val : values) {

String ss = val.toString();

s.add(ss);

}

Collections.sort(s);

for (int i = 0; i < s.size(); i++) {

int j = i;

for (; j < s.size() && s.get(j).equals(s.get(i)); j++) ;

tree.put(j - i + 1, s.get(i));

if (tree.size() > 5) {

tree.remove(tree.firstKey());

} i=j-1;

}

Iterator<Integer> it1 = tree.keySet().iterator();

while (it1.hasNext()) {

int kk = it1.next();

String vv = tree.get(kk);

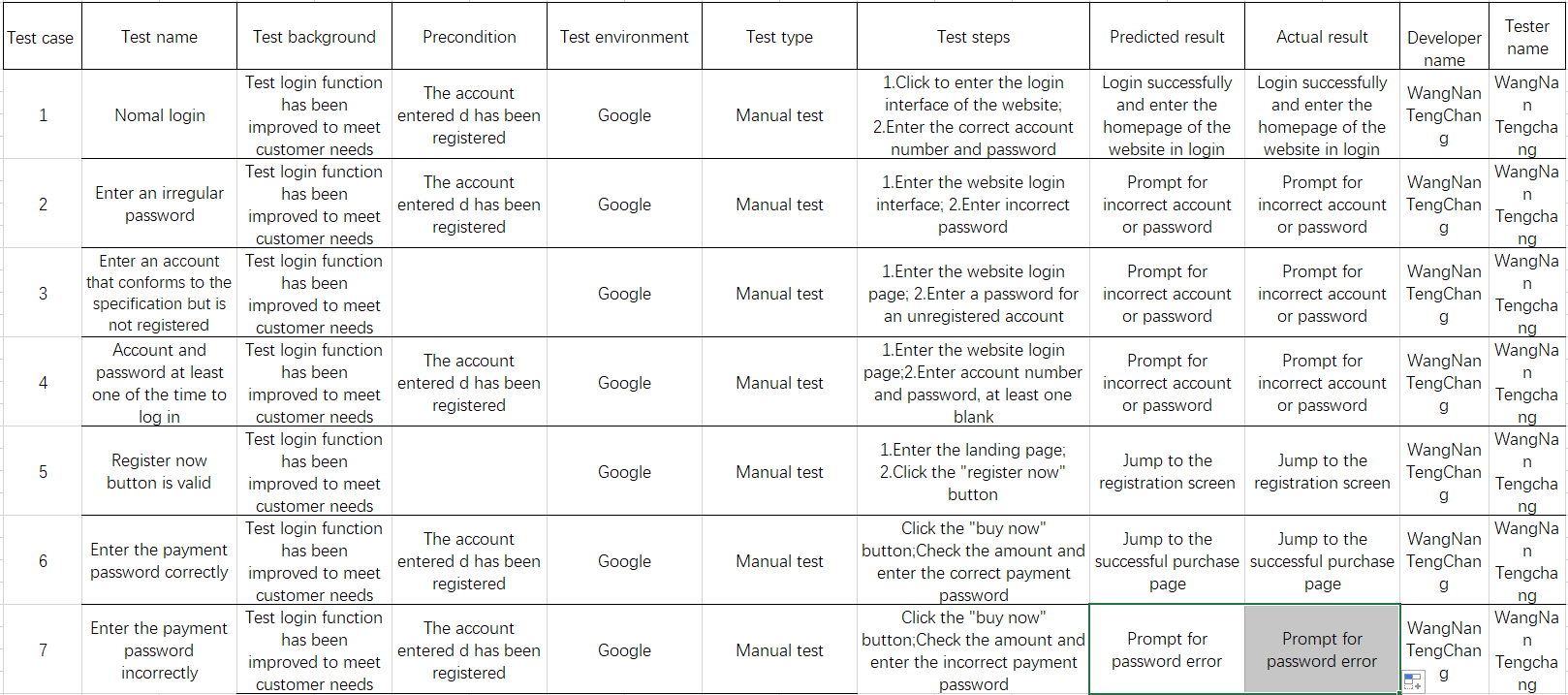
context.write(new Text(vv), new Text(Integer.toString(kk)));

}

}

}

**测试**

****

**潜在改进**

1. 安全性

在目前的系统中，即使用户使用合适的用户名和密码进入银行内网站点，也不是完全安全的，我们可以使用session来防止这个问题。

1. 数据验证

在当前系统中，所有添加页面的数字和日期字段都有数据验证，但更新页面没有。如果用户在更新过程中犯了错误，应该在所有更新页面中添加数据验。

1. 有限历史记忆

用户有时会错误地注销自己的账户，然而被错误注销的账户中存放一些用户来不及移出的数据，此时应该有一个注销的期限，在一段时间内进行的账户注销可以恢复。

1. 页面指数

页面指数函数如果表视图页中有页索引函数，则从下拉菜单进行搜索的行为将不适当。例如，当用户查看所有产品并转到第二页时，用户不能从下拉框中选择任何其他选项。有了这个问题，页面索引函数就从当前系统中删除了。