CS548 2013-Summer Term Project

My First Online Shop Website

Ying Zhang, SID 10167

Northwestern Polytechnic University

Index

Part I: Project introduction

This is my first online shop website, where a new customer could register himself and then log in to browse all available products. The customer could add products into his shopping cart and place the order at any time later. One order could contain multiple products. The tax is calculated according to the state of the customer registered.
Payment is not handled in this project.

Part II: Project details

Main Threads between the pages:

- Login Page → ProdList Page (if login successfully: cusname match the DB records)
- Login Page → Registration Page (if cusname not found in DB records)
- ProdList Page → ProdDetails Page (by selecting specific product) → AddItem Page (enter the quantities to order) → ProdList Page (by selecting "continue shopping") or Place Order Page (by selecting "Placing Order")

Part III: Project Source Code

Domain Classes

CUSTOMER.JAVA

```
package edu.npu.shop.domain;

import javax.validation.constraints.Size;

import org.hibernate.validator.constraints.Email;
import org.hibernate.validator.constraints.NotEmpty;

public class Customer {
     private int id;
     @NotEmpty
```

```
@Size(min=2,max=8)
private String name;
private String state;
//private String pwd;
@NotEmpty
private String addr;
@Email
private String email;
private String phone;

public Customer(){

}
public Customer(String name){
     this.name = name;
}
public Customer(String name, String state){
     this.name = name;
     this.state = state;
}
public int getId() {
     return id;
}

public String getState(){
     return state;
}
public String getName(){
     return name;
}
public String getAddr(){
     return addr;
}
public String getEmail(){
     return email;
```

```
}
public String getPhone(){
     return phone;
}
public void setId(int id){
     this.id = id;
}
public void setName(String name){
     this.name = name;
}
public void setAddr(String addr){
     this.addr = addr;
}
public void setEmail(String email) {
     this.email = email;
}
public void setPhone(String phone) {
     this.phone = phone;
}
public void setState(String state) {
     this.state = state;
}
public String getCustState(){
     return state;
}

public String toString(){
     return "Name: " + name
               + "\n\tAddress: " + addr + "\tState: " + state
               + "\n\tEmail: " + email + "\tPhone: " + phone;
}

public boolean equals(Object obj) {
     if(obj instanceof Customer) {
          Customer cust = (Customer) obj;
```

```
                if(this.id == cust.id)
                    return true;
            }
            return false;
        }
}
```

PRODUCT.JAVA

```
package edu.npu.shop.domain;

public class Product {
    private int id;
    private String brand;
    private int invtQuantity;
    private int totalOrders;
    public int getTotalOrders() {
        return totalOrders;
    }
    public void setTotalOrders(int totalOrders) {
        this.totalOrders = totalOrders;
    }

    private String name;
    private double price;

    public void setId(int id) { this.id = id;}
    public void setName(String name) { this.name = name;}
    public void setBrand(String brand) { this.brand = brand;}
    public void setPrice(double price) { this.price = price;}
    public void setInvtQuantity(int invtQuantity) { this.invtQuantity  = invtQuantity;}

    public int getId() {return id;}
```

```
	public String getName() {return name;}
	public String getBrand() {return brand;}
	public double getPrice() {return price;}
	public int getInvtQuantity() {return invtQuantity;}

	public Product(){}
	public Product(String name, double price){
		this.name = name;
		this.price = price;
	}

	public String toString(){
		return name + "\tunit price: " + price;
	}

	public boolean equals(Object obj){
		if (obj instanceof Product){
			Product otherProd = (Product) obj;
			if(name.equals(otherProd.name) && price==otherProd.price)
				return true;
		}
		return false;
	}
}
```

ORDERITEM.JAVA

```
package edu.npu.shop.domain;

public class OrderItem {
	private int orderItemId;
	private int orderId;
	private int numOfProdOrdered;
	private int prodId;
```

```
public OrderItem(){}
public OrderItem(int prodId, int num){
     this.numOfProdOrdered = num;
     this.prodId = prodId;
}
public void setOrderItemId(int id) {this.orderItemId = id;}
public void setOrderId(int id) {this.orderId = id;}
public void setNumOfProdOrdered(int num) { this.numOfProdOrdered = num; }
public void setProdId(int prodId) { this.prodId = prodId; }

public int getOrderItemId() {return this.orderItemId;}
public int getOrderId() {return this.orderId;}
public int getNumOfProdOrdered() { return this.numOfProdOrdered; }
public int getProdId() { return this.prodId; }

public String toString(){
     return "\t" + prodId + "\tx " + numOfProdOrdered + "\n";
}

public boolean matches(Object obj){
     if (obj instanceof OrderItem){
          OrderItem otherOrderItem = (OrderItem) obj;
          if(this.prodId == otherOrderItem.prodId)
               return true;
     }
     return false;
}
public boolean matches(Product prod){
     if(this.prodId == prod.getId()){
          return true;
     } else{
          return false;
     }
}
```

```
}
```

ORDER.JAVA

```
package edu.npu.shop.domain;

import java.util.*;

public class Order {
     private int id;
     private Date date;
     private String cusname;
     private List<OrderItem> itemsOrdered;

     private double subtotal;
     private double tax;
     private double total;

     public void setId(int id){this.id = id;}
     public void setDate(Date date) {this.date = date;}
     public void setCusname(String cusname) {this.cusname = cusname;}
     public void setSubtotal(double subtotal) {this.subtotal = subtotal;}
     public void setTax(double tax) {this.tax = tax;}
     public void setTotal(double total) {this.total = total;}
     public void setItemsOrdered(List<OrderItem> itemsOrdered){
          this.itemsOrdered = itemsOrdered;
     }
     public void calcOrderTotal() {total = subtotal + tax;}

     public int getId(){return id;}
     public Date getDate(){return date;}
     public String getCusname(){return cusname;}
     public double getSubtotal(){return subtotal;}
     public double getTax(){return tax;}
     public double getTotal(){return total;}
```

```
public List<OrderItem> getItemsOrdered() { return itemsOrdered;}

public Order() {
     itemsOrdered = new ArrayList<OrderItem>();
}
public Order(int id){
     this.id = id;
     itemsOrdered = new ArrayList<OrderItem>();
}
public Order(String cusname) {
     this.cusname = cusname;
     itemsOrdered = new ArrayList<OrderItem>();
}
public Order(String cusname, int prodId, int num){
     this.cusname = cusname;
     itemsOrdered = new ArrayList<OrderItem>();

     OrderItem orderItem = new OrderItem(prodId, num);
     itemsOrdered.add(orderItem);
}

public void addItem(OrderItem item){
     //If the Product is the same as an already existing Product in the Order,
     //do not add the OrderItem. Instead, merge it into the existing OrderItem
     int numOfProd = itemsOrdered.size();
     int i;
     OrderItem eachItem;
     for(i=0;i<numOfProd;i++){
          eachItem = itemsOrdered.get(i);
          if(eachItem.matches(item)){
               int newNum = item.getNumOfProdOrdered() + eachItem.getNumOfProdOrdered();
               eachItem.setNumOfProdOrdered(newNum);
               break;
          }
     }
```

```
        //new product into the order
        if(i== numOfProd){
            itemsOrdered.add(item);
        }
    }

    public void updateProduct(int prodId, int newNum) {
        int i;
        int numOfProd = itemsOrdered.size();
        OrderItem eachItem;
        for(i=0;i<numOfProd;i++){
            eachItem = itemsOrdered.get(i);
            if(eachItem.getProdId() == prodId){
                eachItem.setNumOfProdOrdered(newNum);
                break;
            }
        }
        //to-be-updated product not found
        //if(i== numOfProd){
        //   throw new Exception("Exception: to-be-updated Product not found");
        //}
    }

    public OrderItem getOrderItemAt(int index) {
        OrderItem orderItem = itemsOrdered.get(index);
        return orderItem;
    }

    public int getNumOfOneProd(int prodId){
        int i;
        int numOfProd = itemsOrdered.size();
        OrderItem eachItem;
        for(i=0;i<numOfProd;i++){
            eachItem = itemsOrdered.get(i);
            if(eachItem.getProdId() == prodId){
```

```
                    return eachItem.getNumOfProdOrdered();
               }
          }
          return 0;
     }

     public int getNumProd(){
          return itemsOrdered.size();
     }

     public String toString(){
          return "++++++Order#: " + id
                    + "\t Customer: " + cusname
                    + "\n" + itemsOrdered
                    + "\n\tSubtotal: \t" + subtotal
                    + "\n\tTax: \t\t" + tax
                    + "\n\tTotal: \t\t" + total
                    + "\nEnd of Order Information+++++++++";
     }


}
```

DAO interface

CUSTOMERDAO.JAVA

```
package edu.npu.shop.dao;

import edu.npu.shop.domain.Customer;

public interface CustomerDAO {
     public int findCustCount();
```

```
	public Customer findCustByName(String name);
	public void insertCust(Customer cust);
	public int updateCustAddr(int id, String addr);
	public int updateCustEmail(int id, String email);
	public int updateCustPhone(int id, String phone);
	public int deleteCust(int id);

}
```

PRODUCTDAO.JAVA

```
package edu.npu.shop.dao;

import java.util.List;
import edu.npu.shop.domain.Product;

public interface ProductDAO {
	public int findProductCount();
	public int findInvtQuantityByName(String prodName);
	public int findInvtQuantityById(int id);
	public String findProdNameById(int id);
	public void insertProduct(Product prod);
	public Product findProductByName(String name);
	public Product findProductById(int id);
	public List<Product> findProductsByKeyword(String name);
	public List<Product> findAllProducts();
	public List<Product> findAllProductsWithInvt();
	public int updateProdInvt(int id, int invt);
	public int updateProdPrice(int id, double price);
}
```

ORDERITEMDAO.JAVA

```
package edu.npu.shop.dao;
```

```
import java.util.List;

import edu.npu.shop.domain.OrderItem;

public interface OrderItemDAO {
     public void insertOrderItem(OrderItem orderItem);
     public List<OrderItem> findOrderItemsByOrderId(int orderId);
}
```

ORDERDAO.JAVA

```
package edu.npu.shop.dao;
import java.util.List;

import edu.npu.shop.domain.Order;
import edu.npu.shop.domain.Customer;

public interface OrderDAO {
     public int findOrderCount();
     public List<Order> findAllOrders();
     public Order findOrderById(int id);
     public List<Order> findOrderByCust(Customer cust);
     public void insertOrder(Order order);
     //public List<Order> findOrdersByCustAndAmtGtrThan(Order order);
     public int deleteOrder(int id);
}
```

DAO implementation

CUSTROWMAPPER.JAVA

```
package edu.npu.shop.dao.jdbc;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

import edu.npu.shop.domain.Customer;


public class CustRowMapper implements RowMapper<Customer>{
     public Customer mapRow(ResultSet resultSet, int row) throws SQLException {
          Customer cust;

          cust = new Customer();
          cust.setId(resultSet.getInt("id"));
          cust.setName(resultSet.getString("name"));
          cust.setAddr(resultSet.getString("addr"));
          cust.setEmail(resultSet.getString("email"));
          cust.setPhone(resultSet.getString("phone"));
          cust.setState(resultSet.getString("state"));

          return cust;
     }
}
```

CUSTOMERDAOJDBCIMPL.JAVA

```
package edu.npu.shop.dao.jdbc;

import javax.annotation.PostConstruct;
import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
```

```
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.namedparam.BeanPropertySqlParameterSource;
//import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.jdbc.core.namedparam.SqlParameterSource;
import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
import org.springframework.stereotype.Repository;

import edu.npu.shop.dao.CustomerDAO;
import edu.npu.shop.domain.Customer;

@Repository("customerDaoJdbc")
public class CustomerDaoJdbcImpl implements CustomerDAO {
     @Autowired
     @Qualifier("dataSource")
     private DataSource dataSource;

     private JdbcTemplate jdbcTemplate;
     //private NamedParameterJdbcTemplate dbTemplate;
     private SimpleJdbcInsert jdbcInsert;

     private CustRowMapper custRowMapper;

     @PostConstruct
     public void setup() {
          jdbcTemplate = new JdbcTemplate(dataSource);
          //dbTemplate = new NamedParameterJdbcTemplate(dataSource);

          custRowMapper = new CustRowMapper();

          jdbcInsert = new SimpleJdbcInsert(dataSource)
                         .withTableName("Customer")
                         .usingGeneratedKeyColumns("id")
                         .usingColumns("name", "state", "addr","email","phone");
     }
```

```
public int findCustCount() {
     String sql = "select count(*) from Customer";
     return jdbcTemplate.queryForInt(sql);
}

public Customer findCustByName(String name) {
     String sql = "select * from Customer where name=?";
     try{
     Customer cust = jdbcTemplate.queryForObject(sql,custRowMapper,name);
     return cust;
     } catch (Exception ex) {
          Customer cust = null;
          return cust;
     }
}

public void insertCust(Customer cust) {
     SqlParameterSource params = new BeanPropertySqlParameterSource(cust);
    Number newId = jdbcInsert.executeAndReturnKey(params);
    cust.setId(newId.intValue());
}

public int updateCustAddr(int id, String addr) {
     String sql = "update Customer set addr= ? where id=?";
     int rowsChanged = jdbcTemplate.update(sql,addr,id);
     return rowsChanged;
}
public int updateCustEmail(int id, String email) {
     String sql = "update Customer set email= ? where id=?";
     int rowsChanged = jdbcTemplate.update(sql,email,id);
     return rowsChanged;
}
public int updateCustPhone(int id, String phone) {
     String sql = "update Customer set phone= ? where id=?";
```

```
            int rowsChanged = jdbcTemplate.update(sql,phone,id);
            return rowsChanged;
      }

      public int deleteCust(int id) {
            String sql = "delete from Customer where id= ?";
            int rowsRemoved = jdbcTemplate.update(sql,id);
            return rowsRemoved;
      }

}
```

PRODROWMAPPER.JAVA

```
package edu.npu.shop.dao.jdbc;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

import edu.npu.shop.domain.Product;

public class ProdRowMapper implements RowMapper<Product> {
      public Product mapRow(ResultSet resultSet, int row) throws SQLException {
            Product prod;

            String name = resultSet.getString("name");
            float price = resultSet.getFloat("price");
            prod = new Product(name, price);

            prod.setId(resultSet.getInt("id"));
            prod.setBrand(resultSet.getString("brand"));
            prod.setInvtQuantity(resultSet.getInt("invtquantity"));
```

```
        return prod;

    }
}
```

PRODUCTDAOJDBCIMPL.JAVA

```
package edu.npu.shop.dao.jdbc;

import edu.npu.shop.dao.ProductDAO;
import edu.npu.shop.domain.Product;

import java.util.List;

import javax.annotation.PostConstruct;
import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.namedparam.BeanPropertySqlParameterSource;
import org.springframework.jdbc.core.namedparam.SqlParameterSource;
import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
import org.springframework.stereotype.Repository;


@Repository("productDaoJdbc")
public class ProductDaoJdbcImpl implements ProductDAO {
    @Autowired
    @Qualifier("dataSource")
    private DataSource dataSource;
    private ProdRowMapper prodRowMapper;
```

```
private JdbcTemplate jdbcTemplate;
private SimpleJdbcInsert jdbcInsert;

@PostConstruct
public void setup() {
     jdbcTemplate = new JdbcTemplate(dataSource);
     prodRowMapper = new ProdRowMapper();
     jdbcInsert = new SimpleJdbcInsert(dataSource)
   .withTableName("Product")
   .usingGeneratedKeyColumns("id")
   .usingColumns("name", "brand", "price","invtquantity");
}

public int findProductCount() {
     String sql = "select count(*) from Product";
     return jdbcTemplate.queryForInt(sql);
}

public int findInvtQuantityByName(String prodName) {
     String sql = "select invtquantity from Product where name=?";
     return jdbcTemplate.queryForInt(sql,prodName);
}
public int findInvtQuantityById(int id) {
     String sql = "select invtquantity from Product where id=?";
     return jdbcTemplate.queryForInt(sql,id);
}

public String findProdNameById(int id) {
     String sql = "select name from Product where id=?";
     return jdbcTemplate.queryForObject(sql, String.class, id);
}

public void insertProduct(Product prod) {
     SqlParameterSource params = new BeanPropertySqlParameterSource(prod);
```

```
    Number newId = jdbcInsert.executeAndReturnKey(params);
    prod.setId(newId.intValue());
}

public int updateProdInvt(int id, int invt) {
     String sql = "update Product set invtquantity= ? where id=?";
     int rowsChanged = jdbcTemplate.update(sql,invt,id);
     return rowsChanged;
}
public int updateProdPrice(int id, double price) {
     String sql = "update Product set price= ? where id=?";
     int rowsChanged = jdbcTemplate.update(sql,price,id);
     return rowsChanged;
}

public Product findProductByName(String name) {
     String sql = "select * from Product where name=?";
     Product prod = jdbcTemplate.queryForObject(sql,prodRowMapper,name);
     return prod;
}
public Product findProductById(int id) {
     String sql = "select * from Product where id=?";
     Product prod = jdbcTemplate.queryForObject(sql,prodRowMapper,id);
     return prod;
}

public List<Product> findProductsByKeyword(String name) {
     String sql = "select * from Product where name like '%" + name + "%'";
     List<Product> prodList = jdbcTemplate.query(sql, prodRowMapper);
     return prodList;
}

public List<Product> findAllProducts() {
     String sql = "select * from Product";
     List<Product> prodList = jdbcTemplate.query(sql, prodRowMapper);
```

```
        return prodList;
    }

    public List<Product> findAllProductsWithInvt() {
        String sql = "select * from Product where invtquantity > 0";
        List<Product> prodList = jdbcTemplate.query(sql, prodRowMapper);
        return prodList;
    }
}
```

ORDERITEMROWMAPPER.JAVA

```
package edu.npu.shop.dao.jdbc;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

import edu.npu.shop.domain.OrderItem;

public class OrderItemRowMapper implements RowMapper<OrderItem> {
    public OrderItem mapRow(ResultSet resultSet, int row) throws SQLException {
        OrderItem orderItem = new OrderItem();

        orderItem.setOrderItemId(resultSet.getInt("orderItemId"));
        orderItem.setOrderId(resultSet.getInt("orderId"));
        orderItem.setNumOfProdOrdered(resultSet.getInt("numOfProdOrdered"));
        orderItem.setProdId(resultSet.getInt("prodId"));

        return orderItem;
    }
}
```

ORDERITEMDAOJDBCIMPL.JAVA

```
package edu.npu.shop.dao.jdbc;

import java.util.List;

import javax.annotation.PostConstruct;
import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
//import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.namedparam.BeanPropertySqlParameterSource;
import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.jdbc.core.namedparam.SqlParameterSource;
import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
import org.springframework.stereotype.Repository;

import edu.npu.shop.dao.OrderItemDAO;
import edu.npu.shop.domain.OrderItem;

@Repository("orderItemDaoJdbc")
public class OrderItemDaoJdbcImpl implements OrderItemDAO {
     @Autowired
     @Qualifier("dataSource")
     private DataSource dataSource;

     //private JdbcTemplate jdbcTemplate;
     private NamedParameterJdbcTemplate dbTemplate;
     private SimpleJdbcInsert jdbcInsert;
     private OrderItemRowMapper orderItemRowMapper;
```

```
    @PostConstruct
    public void setup() {
        //jdbcTemplate = new JdbcTemplate(dataSource);
        dbTemplate = new NamedParameterJdbcTemplate(dataSource);
        orderItemRowMapper = new OrderItemRowMapper();
        jdbcInsert = new SimpleJdbcInsert(dataSource)
                        .withTableName("orderitem")
                        .usingGeneratedKeyColumns("orderItemId")
                        .usingColumns("orderId","prodId","numOfProdOrdered");
    }

    public void insertOrderItem(OrderItem orderItem) {
        SqlParameterSource params = new BeanPropertySqlParameterSource(orderItem);
       Number newId = jdbcInsert.executeAndReturnKey(params);

       orderItem.setOrderItemId(newId.intValue());
    }

    public List<OrderItem> findOrderItemsByOrderId(int orderId) {
        String sql =
            "SELECT * FROM orderitem WHERE orderId=:id";
             MapSqlParameterSource params = new MapSqlParameterSource("id", orderId);
             List<OrderItem>          orderItemList          =          dbTemplate.query(sql,
params,orderItemRowMapper);
        return orderItemList;
    }

}
```

ORDERROWMAPPER.JAVA

```
package edu.npu.shop.dao.jdbc;
```

```
import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

import edu.npu.shop.domain.Order;

public class OrderRowMapper implements RowMapper<Order> {

     public Order mapRow(ResultSet resultSet, int row) throws SQLException {
          int id = resultSet.getInt("id");
          Order order = new Order(id);
          //ArrayList<OrderItem> itemsOrdered = new ArrayList<OrderItem>();

          order.setDate(resultSet.getDate("date"));
          order.setCusname(resultSet.getString("cusname"));
          order.setSubtotal(resultSet.getDouble("subtotal"));
          order.setTax(resultSet.getDouble("tax"));
          order.setTotal(resultSet.getDouble("total"));
          //order.setItemsOrdered(itemsOrdered);

          return order;
     }
}
```

ORDERDAOJDBCIMPL.JAVA

```
package edu.npu.shop.dao.jdbc;

import java.util.List;

import javax.annotation.PostConstruct;
import javax.sql.DataSource;
```

```
import edu.npu.shop.dao.OrderDAO;
import edu.npu.shop.domain.Customer;
import edu.npu.shop.domain.Order;


import org.springframework.beans.factory.annotation.*;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.namedparam.BeanPropertySqlParameterSource;
import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.jdbc.core.namedparam.SqlParameterSource;
import org.springframework.jdbc.core.simple.SimpleJdbcInsert;
import org.springframework.stereotype.Repository;

import edu.npu.shop.dao.jdbc.OrderRowMapper;

@Repository("orderDaoJdbc")
public class OrderDaoJdbcImpl implements OrderDAO {
     @Autowired
     @Qualifier("dataSource")
     private DataSource dataSource;

     private JdbcTemplate jdbcTemplate;
     private NamedParameterJdbcTemplate dbTemplate;
     private SimpleJdbcInsert jdbcInsert;
     private OrderRowMapper orderRowMapper;

     @PostConstruct
     public void setup() {
          jdbcTemplate = new JdbcTemplate(dataSource);
          dbTemplate = new NamedParameterJdbcTemplate(dataSource);
          orderRowMapper = new OrderRowMapper();
          jdbcInsert = new SimpleJdbcInsert(dataSource)
                         .withTableName("CusOrder")
                         .usingGeneratedKeyColumns("id")
```

```
                    .usingColumns("cusname", "date","subtotal","tax","total");
    }


    public int findOrderCount() {
         String sql = "select count(*) FROM CusOrder";
         return jdbcTemplate.queryForInt(sql);
    }

    public List<Order> findAllOrders() {
         String sql = "SELECT * FROM CusOrder";
         List<Order> orderList = jdbcTemplate.query(sql, orderRowMapper);
         return orderList;
    }

    public Order findOrderById(int id) {
         String sql = "SELECT * FROM CusOrder WHERE id = :id";
         MapSqlParameterSource params = new MapSqlParameterSource("id", id);
         Order order = dbTemplate.queryForObject(sql, params, orderRowMapper);
         return order;
    }

    public List<Order> findOrderByCust(Customer cust) {
         String sql =
             "SELECT * FROM CusOrder WHERE cusname=:name";
              BeanPropertySqlParameterSource              params              =              new
BeanPropertySqlParameterSource(cust);
              List<Order> orderList = dbTemplate.query(sql, params, orderRowMapper);
         return orderList;
    }

    public void insertOrder(Order order) {
         SqlParameterSource params = new BeanPropertySqlParameterSource(order);
        Number newId = jdbcInsert.executeAndReturnKey(params);
```

```
            order.setId(newId.intValue());
        }

        public int deleteOrder(int id) {
             String sql = "delete from cusorder where id= ?";
             int rowsRemoved = jdbcTemplate.update(sql,id);
             return rowsRemoved;
        }

}
```

Service interface

CUSTOMERSERVICE.JAVA

```
package edu.npu.shop.services;

import edu.npu.shop.domain.Customer;

public interface CustomerService {
     public boolean isCustomerNameExisted(String name);
     public boolean registerNewCust(Customer customer);
}
```

PRODUCTSERVICE.JAVA

```
package edu.npu.shop.services;

import java.util.List;
```

```
import edu.npu.shop.domain.Product;


public interface ProductService {
     public void productSale(int prodId, int num) throws Exception;
     public List<Product> findAllProd() throws Exception;
     public List<Product> findAllProdWithInvt() throws Exception;
     public Product findProdById(int id) throws Exception;
}
```

ORDERITEMSERVICE.JAVA

```
package edu.npu.shop.services;

import edu.npu.shop.domain.OrderItem;

public interface OrderItemService {
     public void purchaseOrderItem(OrderItem orderItem);
}
```

ORDERSERVICE.JAVA

```
package edu.npu.shop.services;
import edu.npu.shop.domain.*;

public interface OrderService {
     public Order createNewCustOrder(String cusNum);
     public Order createNewCustOrder(String cusname, int prodId, int num);
     public void addProdIntoOrder(Order newOrder, int prodId, int num);
     public void updateProdInOrder(Order newOrder, int prodId, int newNum);
     //public void updateOrderSubtotal(Order newOrder);
     public String getOrderState(Order newOrder);
```

```
    public int getTotalOrders();
    public Order placeOrder(Order newOrder);
    public Order findOrderById(int id);
}
```

TAXSERVICE.JAVA

```
package edu.npu.shop.services;
import edu.npu.shop.domain.*;

public interface TaxService {
    public void printTaxRates();
    public double computeTax(Order order, String state);
}
```

PURCHASESERVICE.JAVA

```
package edu.npu.shop.services;

import edu.npu.shop.domain.Order;


public interface PurchaseService {
    public Order processCustomerPurchase(Order newOrder) throws Exception;
}
```

Service implement

CUSTOMERSERVICEIMPL.JAVA

```
package edu.npu.shop.services;
```

```
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.npu.shop.dao.CustomerDAO;
import edu.npu.shop.domain.Customer;



@Service("customerService")
@Transactional(readOnly=true)
public class CustomerServiceImpl implements CustomerService {
     @Autowired
     private CustomerDAO customerDao;

     public boolean isCustomerNameExisted(String name) {
          Customer customerFound = customerDao.findCustByName(name);
          if(customerFound == null) {
               return false;
          }
          return true;
     }

     @Transactional(readOnly=false, rollbackFor=java.lang.Exception.class)
     public boolean registerNewCust(Customer customer) {
          try{
               customerDao.insertCust(customer);
          } catch (Exception ex) {
               return false;
          }
          return true;
     }
}
```

PRODUCTSERVICEIMPL.JAVA

```
package edu.npu.shop.services;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.npu.shop.dao.ProductDAO;
import edu.npu.shop.domain.Product;


@Service("productService")
@Transactional(readOnly=true)
public class ProductServiceImpl implements ProductService {
     @Autowired
     private ProductDAO prodDao;

     @Transactional(readOnly=false, rollbackFor=java.lang.Exception.class)
     public void productSale(int prodId, int num) throws Exception {
          int origNum = prodDao.findInvtQuantityById(prodId);
          if(origNum < num) {
               throw new Exception("ProductDao Exception: NOT enough inventory for purchase");
          }
          int newNum = origNum - num;
          int rowsUpdated = prodDao.updateProdInvt(prodId, newNum);
          if (rowsUpdated != 1) {
               throw new Exception("ProductDao Exception: Unable to update order total");
          }
     }
```

```
@Transactional(readOnly=false)
public void insertProduct(Product prod) {
     prodDao.insertProduct(prod);
}

@Transactional(rollbackFor=java.lang.Exception.class)
public Product findProdByName(String prodName) throws Exception {
     Product prod;

     prod = prodDao.findProductByName(prodName);
     if (prod == null) {
          throw new Exception("ProductDao Exception: " + prodName + " Not Found");
     }
     return prod;
}

@Transactional(rollbackFor=java.lang.Exception.class)
public List<Product> findAllProd() throws Exception {
     List<Product> prodList;

     prodList = prodDao.findAllProducts();
     if (prodList == null) {
          throw new Exception("ProductDao Exception: No products");
     }
     return prodList;
}

@Transactional(rollbackFor=java.lang.Exception.class)
public List<Product> findAllProdWithInvt() throws Exception {
     List<Product> prodList;

     prodList = prodDao.findAllProductsWithInvt();
     if (prodList == null) {
          throw new Exception("ProductDao Exception: No products");
     }
```

```
        return prodList;
    }

    @Transactional(rollbackFor=java.lang.Exception.class)
    public Product findProdById(int id) throws Exception {
        Product product;

        product = prodDao.findProductById(id);
        if (product == null) {
            throw new Exception("ProductDao Exception: No products");
        }
        return product;
    }
}
```

ORDERITEMSERVICEIMPL.JAVA

```
package edu.npu.shop.services;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Service;

import edu.npu.shop.dao.OrderItemDAO;
import edu.npu.shop.domain.OrderItem;

@Service("orderItemService")
public class OrderItemServiceImpl implements OrderItemService {
    @Autowired
    @Qualifier("orderItemDaoJdbc")
    private OrderItemDAO orderItemDao;

    public void purchaseOrderItem(OrderItem orderItem) {
```

```
            orderItemDao.insertOrderItem(orderItem);
        }
}
```

ORDERSERVICEIMPL.JAVA

```
package edu.npu.shop.services;
import edu.npu.shop.domain.*;
import edu.npu.shop.dao.*;

import java.util.Date;
import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.beans.factory.annotation.Qualifier;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;


@Service("orderService")
public class OrderServiceImpl implements OrderService {
    @Autowired
    @Qualifier("orderDaoJdbc")
    private OrderDAO orderDao;
    @Autowired
    @Qualifier("productDaoJdbc")
    private ProductDAO productDao;
    @Autowired
    @Qualifier("customerDaoJdbc")
    private CustomerDAO customerDao;
    @Autowired
    @Qualifier("orderItemDaoJdbc")
    private OrderItemDAO orderItemDao;
```

```
@Autowired
private TaxService taxService;

public int getTotalOrders() {
     return orderDao.findOrderCount();
}

public Order findOrderById(int id) {
     Order order = orderDao.findOrderById(id);
     List<OrderItem> orderItems = orderItemDao.findOrderItemsByOrderId(id);
     order.setItemsOrdered(orderItems);
     return order;
}

public Order createNewCustOrder(String cusname) {
     Order newOrder= new Order(cusname);
     return newOrder;
}

public Order createNewCustOrder(String cusname, int prodId, int num) {
     Order newOrder = new Order(cusname, prodId, num );
     return newOrder;
}

public void addProdIntoOrder(Order newOrder, int prodId, int num) {
     OrderItem orderItem = new OrderItem(prodId, num);
     newOrder.addItem(orderItem);
     updateOrderSubtotal(newOrder, prodId, num);
}
public void updateProdInOrder(Order newOrder, int prodId, int newNum) {
     newOrder.updateProduct(prodId, newNum);
     updateOrderSubtotal(newOrder, prodId, newNum);
}
public void updateOrderSubtotal(Order newOrder, int prodId, int num) {
     double subtotal=newOrder.getSubtotal();
```

```
            Product prod = productDao.findProductById(prodId);
            double newAmt = num * prod.getPrice();
            subtotal += newAmt;

            newOrder.setSubtotal(subtotal);
      }
      public String getOrderState(Order newOrder) {
            String cusname = newOrder.getCusname();
            Customer cust = customerDao.findCustByName(cusname);
            String state = cust.getCustState();
            return state;
      }

      @Transactional(readOnly=false)
      public Order placeOrder(Order newOrder) {
            Date todaysDate = new Date();
            newOrder.setDate(todaysDate);

            //set the total amount for the order: call the taxservice here
            String state = getOrderState(newOrder);
            System.out.println("++++++++++the state of customer is:" + state);
            double tax = taxService.computeTax(newOrder, state);
            newOrder.setTax(tax);
            newOrder.calcOrderTotal();

            orderDao.insertOrder(newOrder);
            return newOrder;
      }
}
```

TAXSERVICEIMPL.JAVA

```
package edu.npu.shop.services;
```

```
import edu.npu.shop.domain.*;

import java.util.*;

import org.springframework.stereotype.Service;

//@Service("taxServiceForSales")
public class TaxServiceForSales implements TaxService {
     private Map<String, String> stateTaxRates;

     public void setStateTaxRates(Map<String, String> stateTaxRates){
          this.stateTaxRates = stateTaxRates;
     }
     public Map<String, String> getStateTaxRates(){
          return stateTaxRates;
     }
     public void printTaxRates(){
          Set<String> keys = stateTaxRates.keySet();
          Iterator<String> iter = keys.iterator();
          while(iter.hasNext()){
               String state = iter.next().toString();
               String rate = stateTaxRates.get(state);
               System.out.println("Tax Rate for " + state + " is: " + rate);
          }
     }
     public double getTaxRateForState(String state) throws Exception {
          String rateStr = stateTaxRates.get(state);
          System.out.println("++++++++++++string rate= " + rateStr);
          if(rateStr != null){
               try{
               double rate = Double.parseDouble(rateStr); //assume no error
               return rate;
               } catch (NumberFormatException ex){
                    throw new Exception("Exception: invalid input for state rates");
```

```
            }
        } else {
            return 0;
        }
    }

    public double computeTax(Order order, String state){
        double orderSubtotal = order.getSubtotal();
        double rate;

        try{
        rate= getTaxRateForState(state);
        } catch (Exception ex){
            System.out.println("ex msg:" + ex.getMessage());
            rate = 0;
        }
        //System.out.println("sub:  "  +  orderSubtotal  +  "rate:  "  +  rate  +  "For  ST:  "  +
state);
        double tax = orderSubtotal * rate;
        return tax;
    }
}
```

PURCHASESERVICEIMPL.JAVA

```
package edu.npu.shop.services;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;
import org.springframework.transaction.annotation.Transactional;

import edu.npu.shop.domain.Order;
import edu.npu.shop.domain.OrderItem;
```

```
@Transactional
@Service("purchaseService")
public class PurchaseServiceImpl implements PurchaseService {
     @Autowired
     OrderService orderService;
     @Autowired
     ProductService productService;
     @Autowired
     OrderItemService orderItemService;

     @Transactional(rollbackFor=java.lang.Exception.class)
     public Order processCustomerPurchase(Order newOrder) throws Exception {
          //place the order to get the orderId created from database
          newOrder = orderService.placeOrder(newOrder);

          //For each product ordered, check whether enough inventory, roll back at any
exceptions
          int numProducts = newOrder.getNumProd();
          int orderId = newOrder.getId();
          OrderItem eachItem;
          for(int i=0;i<numProducts;i++) {
               eachItem = newOrder.getOrderItemAt(i);
               eachItem.setOrderId(orderId);
               int eachProdId = eachItem.getProdId();
               int eachProdNum = eachItem.getNumOfProdOrdered();
               productService.productSale(eachProdId, eachProdNum);

               orderItemService.purchaseOrderItem(eachItem);
          }
          return newOrder;
     }
}
```

Controller

CUSTOMERCONTROLLER.JAVA

```
package edu.npu.shop.controllers;

import java.util.List;

import javax.servlet.http.HttpSession;
import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;


import edu.npu.shop.domain.Customer;
import edu.npu.shop.domain.Order;
import edu.npu.shop.domain.Product;
import edu.npu.shop.services.CustomerService;
import edu.npu.shop.services.OrderService;
import edu.npu.shop.services.ProductService;

@Controller
public class CustomerController {
     @Autowired
     ProductService prodService;
     @Autowired
     CustomerService custService;
     @Autowired
     OrderService orderService;
```

```
@RequestMapping(value = {"/", "login"}, method = RequestMethod.GET)
public ModelAndView presentCustLoginForm() {
     ModelAndView modelView = null;

     modelView = new ModelAndView("custLogin");
     modelView.addObject("customer", new Customer());
     return modelView;
}
@RequestMapping(value = {"/", "login"}, method = RequestMethod.POST)
public ModelAndView processCustLoginForm(ModelAndView modelView,
          @Valid Customer customer, BindingResult result,HttpSession session) {

     if(result.hasErrors()) {
          //re-presenting with errors
          modelView.setViewName("custLogin");
          modelView.addObject(customer);
          return modelView;
     }
     String cusname = customer.getName();
     boolean isCustExisted = custService.isCustomerNameExisted(cusname);
     if(!isCustExisted) {
          //re-presenting with errors
          modelView.setViewName("redirect:unknownCustLogin");
          modelView.addObject(customer);
          return modelView;
     }

     //login successful, show the product lists
     String cusNum = customer.getName();
     Order order = orderService.createNewCustOrder(cusNum);
     session.setAttribute("customer", customer);
     session.setAttribute("order", order);

     modelView.setViewName("redirect:product/prodList");
```

```
        return modelView;

    }

    /*
    @RequestMapping(value = {"/", "login"}, method = RequestMethod.POST)
    public    String    processCustLoginForm(@Valid    Customer    customer,    BindingResult
result,HttpSession session) {

        if(result.hasErrors()) {
            //re-presenting with errors
            return "redirect:login";
        }
        String cusname = customer.getName();
        boolean isCustExisted = custService.isCustomerNameExisted(cusname);
        if(!isCustExisted) {
            //re-presenting with errors
            return "redirect:unknownCustLogin";
        }

        //login successful, show the product lists
        String cusNum = customer.getName();
        Order order = orderService.createNewCustOrder(cusNum);
        session.setAttribute("customer", customer);
        session.setAttribute("order", order);

        return "redirect:product/prodList";
    }
    */

    @RequestMapping(value = {"unknownCustLogin"}, method = RequestMethod.GET)
    public ModelAndView unknownCustLogin() {
        ModelAndView modelView = null;
        modelView = new ModelAndView("unknownCustLogin");
        return modelView;
```

```
    }

    @RequestMapping(value = {"register"}, method = RequestMethod.GET)
    public ModelAndView presentCustRegForm(@Valid Customer customer, BindingResult result) {
        ModelAndView modelView = null;
        modelView = new ModelAndView("custRegistration");
        modelView.addObject("customer", new Customer());
        return modelView;
    }

    /*
    @RequestMapping(value = {"register"}, method = RequestMethod.POST)
    public String processCustRegForm(@Valid Customer customer, BindingResult result,
HttpSession session) {
        //ModelAndView modelView = null;

        if(result.hasErrors()) {
            //re-presenting with errors: validation errors
            //modelView = new ModelAndView("custRegistration","customer",customer);
            //return modelView;
            return "redirect:register";
        }
        String cusname = customer.getName();
        boolean isCustExisted = custService.isCustomerNameExisted(cusname);
        if(isCustExisted) {
            //re-presenting with errors: cusname already existed
            //modelView = new ModelAndView("custRegistration","customer",customer);
            //return modelView;
            return "redirect:register";
        }

        //add new customer into customer table
        boolean isCustCreated = custService.registerNewCust(customer);
        if(!isCustCreated){
            //re-presenting with errors: new customer insertion failed
```

```
            //modelView = new ModelAndView("custRegistration","customer",customer);
            //return modelView;
            return "redirect:register";
        }

        //login successful, show the product lists
        String cusNum = customer.getName();
        Order order = orderService.createNewCustOrder(cusNum);
        session.setAttribute("customer", customer);
        session.setAttribute("order", order);

        return "redirect:product/prodList";

    }*/
    @RequestMapping(value = {"register"}, method = RequestMethod.POST)
    public ModelAndView processCustRegForm(ModelAndView modelView,
            @Valid Customer customer, BindingResult result, HttpSession session) {

        if(result.hasErrors()) {
            //re-presenting with errors: validation errors
            modelView.setViewName("custRegistration");
            modelView.addObject(customer);
            return modelView;
        }
        String cusname = customer.getName();
        boolean isCustExisted = custService.isCustomerNameExisted(cusname);
        if(isCustExisted) {
            //re-presenting with errors: cusname already existed
            modelView.setViewName("custRegistration");
            modelView.addObject("errorMessage", "The customer ID already existed, please
try something else");
            modelView.addObject(customer);
            return modelView;
        }
```

```
            //add new customer into customer table
            boolean isCustCreated = custService.registerNewCust(customer);
            if(!isCustCreated){
                 //re-presenting with errors: new customer insertion failed
                 modelView.setViewName("custRegistration");
                 modelView.addObject("errorMessage",  "The  customer  registration  is  failed,
please re-try");
                 modelView.addObject(customer);
                 return modelView;
            }

            //login successful, show the product lists
            String cusNum = customer.getName();
            Order order = orderService.createNewCustOrder(cusNum);
            session.setAttribute("customer", customer);
            session.setAttribute("order", order);

            modelView.setViewName("redirect:product/prodList");
            return modelView;
       }

}
```

PRODUCTCONTROLER.JAVA

```
package edu.npu.shop.controllers;

import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
```

```
import org.springframework.web.bind.annotation.ModelAttribute;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;


import edu.npu.shop.domain.Product;
import edu.npu.shop.services.ProductService;

@Controller
@RequestMapping("/product")
public class ProductController {
     @Autowired
     ProductService prodService;

     @RequestMapping(value = "/prodList", method = RequestMethod.GET)
     public ModelAndView listAllProducts() {
          List<Product> prodList;
          ModelAndView modelView = null;

          try{
          prodList = prodService.findAllProdWithInvt();
          modelView = new ModelAndView("productsList");
          modelView.addObject("prodList", prodList);

          } catch (Exception ex) {
               // exception in listing all the products
               modelView = new ModelAndView("error");
               modelView.addObject("errorMessage",  "Errors  while  retrieving  all  products
information, please retry");
               return modelView;
          }

          return modelView;
```

```
    }

    @RequestMapping(value = "/prodDetails/{prodId}")
    public ModelAndView productDetails(@PathVariable int prodId, HttpServletRequest request,
           HttpServletResponse response) {
        Product product;
        ModelAndView modelView = null;

        try{
            product = prodService.findProdById(prodId);
            modelView = new ModelAndView("productDetails");
            modelView.addObject("product", product);
        } catch (Exception ex) {
            // exception in listing all the products
            modelView = new ModelAndView("error");
            modelView.addObject("errorMessage", "Errors while retrieving product details,
please retry");
            return modelView;
        }

        return modelView;
    }
}
```

ORDERCONTROLLER.JAVA

```
package edu.npu.shop.controllers;

import java.util.ArrayList;
import java.util.List;

import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
```

```
import javax.servlet.http.HttpSession;
import javax.validation.Valid;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Controller;
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.PathVariable;
import org.springframework.web.bind.annotation.RequestMapping;
import org.springframework.web.bind.annotation.RequestMethod;
import org.springframework.web.servlet.ModelAndView;


import edu.npu.shop.domain.Customer;
import edu.npu.shop.domain.Order;
import edu.npu.shop.domain.OrderItem;
import edu.npu.shop.domain.Product;
import edu.npu.shop.services.OrderService;
import edu.npu.shop.services.ProductService;
import edu.npu.shop.services.PurchaseService;

@Controller
@RequestMapping("/order")
public class OrderController {
     @Autowired
     ProductService prodService;
     @Autowired
     OrderService orderService;
     @Autowired
     PurchaseService purchaseService;

     @RequestMapping(value = "/addItem/{prodId}")
     public ModelAndView processAddItem(@PathVariable int prodId, HttpServletRequest request,
            HttpServletResponse response, HttpSession session) {

          ModelAndView modelView = null;
```

```
          String numStr = request.getParameter("numOfProdOrdered");
          int numToBuy = Integer.parseInt(numStr);

          try{
               Product product = prodService.findProdById(prodId);
               int prodInvt = product.getInvtQuantity();
               if(numToBuy > prodInvt) {
                    modelView = new ModelAndView("error");
                    modelView.addObject("errorMessage", "Not enough inventory, please reduce
your order quantities or come later");
                    return modelView;
               }

          } catch(Exception ex) {
               modelView = new ModelAndView("error");
               modelView.addObject("errorMessage", "Errors while adding the items into your
shopping cart, please retry");
               return modelView;
          }
          //save the orderItem into order obj. for this session
          Order order = (Order) session.getAttribute("order");
          orderService.addProdIntoOrder(order, prodId, numToBuy);
          session.setAttribute("order", order);

          modelView = new ModelAndView("addItemDone");

          return modelView;
     }

     //@RequestMapping(value = "/placeOrder", method = RequestMethod.POST)
     @RequestMapping(value = "/placeOrder")
     public ModelAndView processPlaceOrder(HttpSession session) {

          ModelAndView modelView = null;
          Order order = (Order) session.getAttribute("order");
```

```
        List<OrderItem> orderItems = order.getItemsOrdered();
        int numOrderItems = orderItems.size();

        if(numOrderItems == 0){
             modelView = new ModelAndView("error");
             modelView.addObject("errorMessage", "No items in your shopping cart, please buy
something before placing your order");
             return modelView;
        }

        try{
             order = purchaseService.processCustomerPurchase(order);
        } catch (Exception ex){
             modelView = new ModelAndView("error");
             modelView.addObject("errorMessage", "Errors while placing your order, please
retry");
             return modelView;
        }

        List<Product> prodList = new ArrayList<Product>();
        try{
             for(int i=0;i<numOrderItems;i++){
                  OrderItem eachItem = orderItems.get(i);
                  int eachProdId = eachItem.getProdId();
                  Product eachProd = prodService.findProdById(eachProdId);
                  int totalOrders = eachItem.getNumOfProdOrdered();
                  eachProd.setTotalOrders(totalOrders);
                  prodList.add(eachProd);
             }
             modelView = new ModelAndView("orderSuccess");
             modelView.addObject("prodList", prodList);
             modelView.addObject("order", order);

        } catch (Exception ex) {
             // exception in listing all the products
```

```
                modelView = new ModelAndView("error");
                modelView.addObject("errorMessage", "Errors while listing your order, please
send mail to supports");
                return modelView;
           }

           return modelView;
      }

      @RequestMapping(value = "/shoppingcart")
      public ModelAndView presentShoppingCart(HttpSession session) {

           ModelAndView modelView = null;
           Order order = (Order) session.getAttribute("order");

           List<OrderItem> orderItems = order.getItemsOrdered();
           int numOrderItems = orderItems.size();
           List<Product> prodList = new ArrayList<Product>();
           try{
                for(int i=0;i<numOrderItems;i++){
                     OrderItem eachItem = orderItems.get(i);
                     int eachProdId = eachItem.getProdId();
                     Product eachProd = prodService.findProdById(eachProdId);
                     int totalOrders = eachItem.getNumOfProdOrdered();
                     eachProd.setTotalOrders(totalOrders);
                     prodList.add(eachProd);
                }
                modelView = new ModelAndView("shoppingCart");
                modelView.addObject("prodList", prodList);

           } catch (Exception ex) {
                // exception in listing all the products
                modelView = new ModelAndView("error");
                modelView.addObject("errorMessage", "Errors while retrieving your shopping
cart, please retry");
```

```
                return modelView;
            }

            return modelView;
        }
}
```

View jsp

1) CUSTLOGIN.JSP

```
<%@ include file="./include.jsp"%>
<html>
<head>
     <title>Customer Login</title>
     <style>
          .error {
              font-size: 0.8em;
               color: #ff0000;
          }
     </style>
</head>
<body>
<h3 align="right"><a href="${context}/register" style="margin-right:30px">Register</a></h3>

<h1><fmt:message key="custLoginForm.title" /></h1>

  <form:form action="./login" method="POST" commandName="customer">
  <table>
```

```
     <tr>
     <td><fmt:message key="name" /></td>
        <td><form:input path="name" />
            <form:errors path="name" cssClass="error"/>
        </td>
    </tr>
    <tr>
     <td><fmt:message key="addr" /></td>
        <td><form:input path="addr" />
            <form:errors path="addr" cssClass="error"/>
        </td>
    </tr>

     <tr>
    <td><input type="submit" value="<fmt:message key="loginBtn" />"> </td>
    </tr>
    </table>
  </form:form>

</body>
</html>
```

2) CUSTREGISTRATION.JSP

```
<%@ include file="./include.jsp"%>
<html>
<head>
     <title>Customer Registration</title>
     <style>
```

```
        .error {
            font-size: 0.8em;
             color: #ff0000;
        }
    </style>
</head>
<body>
<h3 align="right"><a href="${context}" style="margin-right:30px">LogIn</a>
</h3>

<h1><fmt:message key="custRegForm.title" /></h1>
<br>
  <form:form action="./register" method="POST" commandName="customer">
<c:if test="${not empty errorMessage}">
  <p style="font-size: 0.8em;color:#ff0000">${errorMessage}</p>
</c:if>

  <table>
     <tr>
     <td><fmt:message key="name" /></td>
        <td><form:input path="name" />
            <form:errors path="name" cssClass="error"/>
        </td>
    </tr>
    <tr>
     <td><fmt:message key="email" /></td>
        <td><form:input path="email" />
            <form:errors path="email" cssClass="error"/>
        </td>
```

```
    </tr>
    <tr>
     <td><fmt:message key="phone" /></td>
        <td><form:input path="phone" />
            <form:errors path="phone" cssClass="error"/>
        </td>
    </tr>
    <tr>
     <td><fmt:message key="addr" /></td>
        <td><form:input path="addr" />
            <form:errors path="addr" cssClass="error"/>
        </td>
    </tr>
     <tr>
     <td><fmt:message key="state" /></td>
        <td><form:input path="state" />
            <form:errors path="state" cssClass="error"/>
        </td>
    </tr>
     <tr>
    <td>
    <br><input type="submit" value="<fmt:message key="registerBtn" />"> </td>
    </tr>
    </table>
  </form:form>

</body>
</html>
```

3) PRODUCTLIST.JSP

```
<%@ include file="./include.jsp"%>

<html>
<head>
   <title>ALL Products Available</title>
</head>

<body>
<h3 align="right">
<a href="${context}/order/shoppingcart">Shopping Cart</a>
<a href="${context}" style="margin-left:9px;margin-right:30px">Logout</a>
<br></h3>

  <h1>ALL Products Available For Now:<br></h1>
  <table BORDER="3" bgcolor="#FAEBD7" cellspacing="10" cellpadding="15">
<tr>
     <th>Product Id</th>
     <th>Name</th>
     <th>Brand</th>
     <th>Price</th>
     <th>Inventory</th>
</tr>
     <c:forEach var="curProd" items="${prodList}">
          <tr>
             <td>${curProd.id}</td>
             <td>${curProd.name}</td>
```

```
            <td>${curProd.brand}</td>
            <td>${curProd.price}</td>
            <td>${curProd.invtQuantity}</td>
            <td><a href="${context}/product/prodDetails/${curProd.id}">Details</a></td>
        </tr>
    </c:forEach>
</table>


<br><br>
</body>
  </html>
```

4) PRODUCTDETAILS.JSP

```
<%@ include file="./include.jsp"%>

<html>
<head>
   <title>Product Details</title>
</head>

<body>
<h3 align="right">
<a href="${context}/product/prodList">Home</a>
<a href="${context}/order/shoppingcart" style="margin-left:9px">Shopping Cart</a>
<a href="${context}" style="margin-left:9px;margin-right:30px">Logout</a>
<br></h3>
```

```
  <h1>ALL about this product ...<br></h1>
  <table BORDER="3" bgcolor="#FAEBD7" cellspacing="10" cellpadding="15">
<tr>
     <th>Product Id</th>
     <th>Name</th>
     <th>Brand</th>
     <th>Price</th>
     <th>Inventory</th>
</tr>
          <tr>
             <td>${product.id}</td>
             <td>${product.name}</td>
             <td>${product.brand}</td>
             <td>${product.price}</td>
             <td>${product.invtQuantity}</td>
          </tr>

</table>

<br>
  <form:form action="${context}/order/addItem/${product.id}">
  <table BORDER="2" bgcolor="#FAEBD7" cellspacing="10" cellpadding="15">
     <tr>
     <td>Quantity to Order:</td>
        <td><input name="numOfProdOrdered" type="text"/>
        </td>
    </tr>
 <br><br>
     <tr>
```

```
    <td><input type="submit" value="Add to Cart" /></td>
    </tr>
    </table>
  </form:form>

</body>
   </html>
```

5) SHOPPINGCART.JSP

```
<%@ include file="./include.jsp"%>
<html>
<head>
     <title>Item Added</title>
     <style>
          .error {
              font-size: 0.8em;
               color: #ff0000;
          }
     </style>
</head>
<body>
<h3 align="right">
<a href="${context}/product/prodList">Home</a>
<a href="${context}" style="margin-left:9px;margin-right:30px">Logout</a>
<br></h3>

<h1>Items in your shopping cart:<br></h1>
```

```
  <table BORDER="3" bgcolor="#FAEBD7" cellspacing="10" cellpadding="15">
<tr>
     <th>Name</th>
     <th>Brand</th>
     <th>Price</th>
     <th>Quantities Ordered</th>
</tr>
     <c:forEach var="curProd" items="${prodList}">
          <tr>
             <td>${curProd.name}</td>
             <td>${curProd.brand}</td>
             <td>${curProd.price}</td>
             <td>${curProd.totalOrders}</td>
          </tr>
     </c:forEach>
</table>

<br><br>
<h2>
<a href="${context}/product/prodList">click here</a> to continue shopping
<br><br>
<a href="${context}/order/placeOrder">click here</a> to place your order
</h2>
<br><br>
</body>
</html>
```

6) ADDITEMDONE.JSP

```
<%@ include file="./include.jsp"%>
<html>
<head>
     <title>Item Added</title>
     <style>
          .error {
              font-size: 0.8em;
               color: #ff0000;
          }
     </style>
</head>
<body>

<h1>Product added into your shopping cart... <br><br></h1>
<h2>
<a href="${context}/product/prodList">click here</a> to continue shopping
<br><br>
<a href="${context}/order/placeOrder">click here</a> to place your order
</h2>

</body>
</html>
```

7) ORDERSUCCESS.JSP

```
<%@ include file="./include.jsp"%>
<html>
<head>
```

```
    <title>Item Added</title>
    <style>
        .error {
            font-size: 0.8em;
             color: #ff0000;
        }
    </style>
</head>
<body>

<h1>Your order placed successfully, details as below: </h1>
<h2>Order Number: ${order.id} <br></h2>

  <table BORDER="3" bgcolor="#FAEBD7" cellspacing="10" cellpadding="15">
<tr>
    <th>Name</th>
    <th>Brand</th>
    <th>Price</th>
    <th>Quantities Ordered</th>
</tr>
    <c:forEach var="curProd" items="${prodList}">
        <tr>
           <td>${curProd.name}</td>
           <td>${curProd.brand}</td>
           <td>${curProd.price}</td>
           <td>${curProd.totalOrders}</td>
        </tr>
    </c:forEach>
```

```
<tr>
     <td>Subtotal:</td>
     <td>${order.subtotal}</td>
</tr>
<tr>
     <td>Tax:</td>
     <td>${order.tax}</td>
</tr>
<tr>
     <td>Total:</td>
     <td>${order.total}</td>
</tr>
</table>

<br>
<h2><a href="${context}/">Login again</a> to place another order</h2>
<br>
</body>
</html>
```

8) UNKNOWNCUSTLOGIN.JSP

```
<%@ include file="./include.jsp"%>
<html>
<head>
     <title>Unknown Customer</title>
     <style>
          .error {
              font-size: 0.8em;
```

```
                color: #ff0000;
            }
        </style>
</head>
<body>

<h1>Unknown Customer, Please <a href="${context}/register">click here</a> to register first
<br><br>Or retry <a href="${context}">LogIn</a>
</h1>

</body>
</html>
```

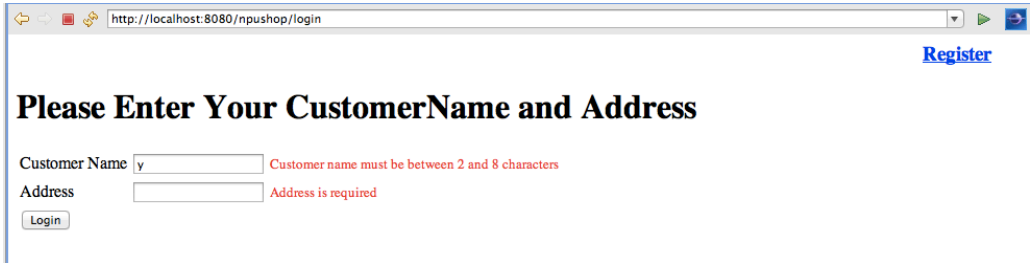
9) ERROR.JSP

```
<%@ include file="./include.jsp"%>

<html>
<head>
   <title>Error</title>
</head>
<body>
<h3  align="right">
<a href="${context}//product/prodList">Home</a>
<a href="${context}" style="margin-left:9px;margin-right:30px">Logout</a>
</h3>

  <h1>Oops ... <br></h1>
```

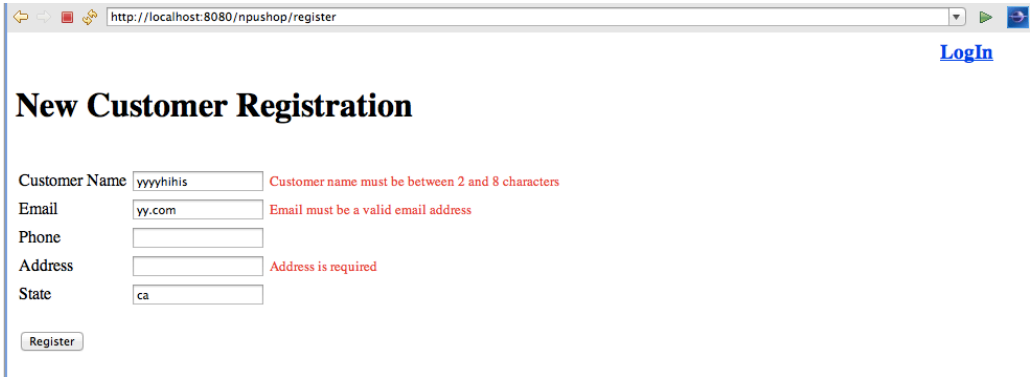
```
<c:if test="${not empty errorMessage}">
  <p style="font-size: 0.8em;color:#ff0000">${errorMessage}</p>
</c:if>

<br><br>
</body>
   </html>
```

Part IV: Test and results

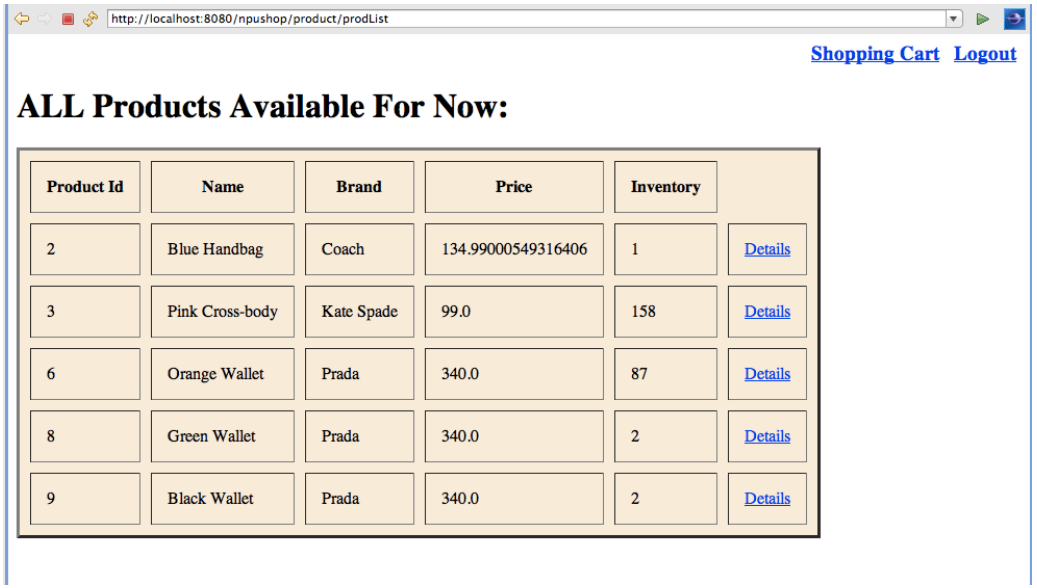
1) Login Form validation:



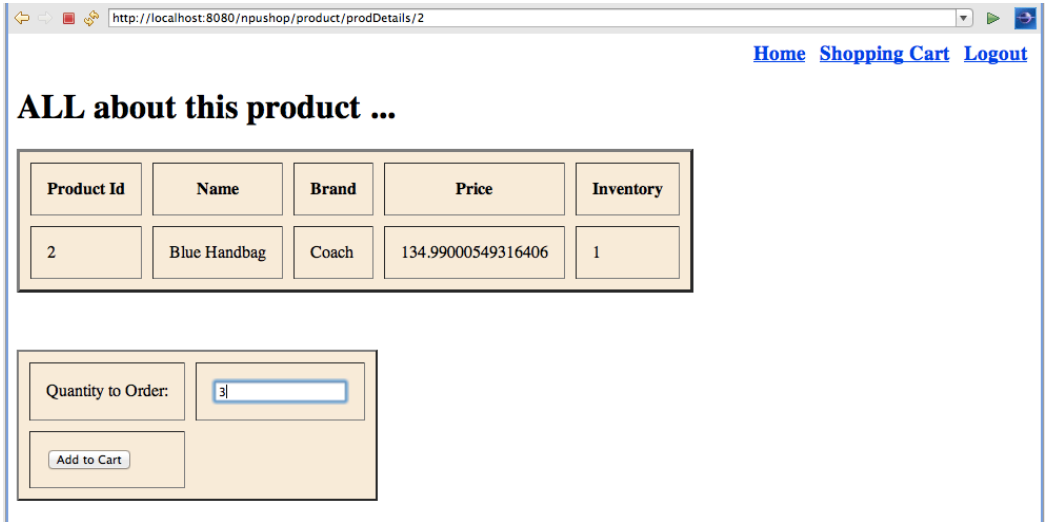
2) Registration Form validation:



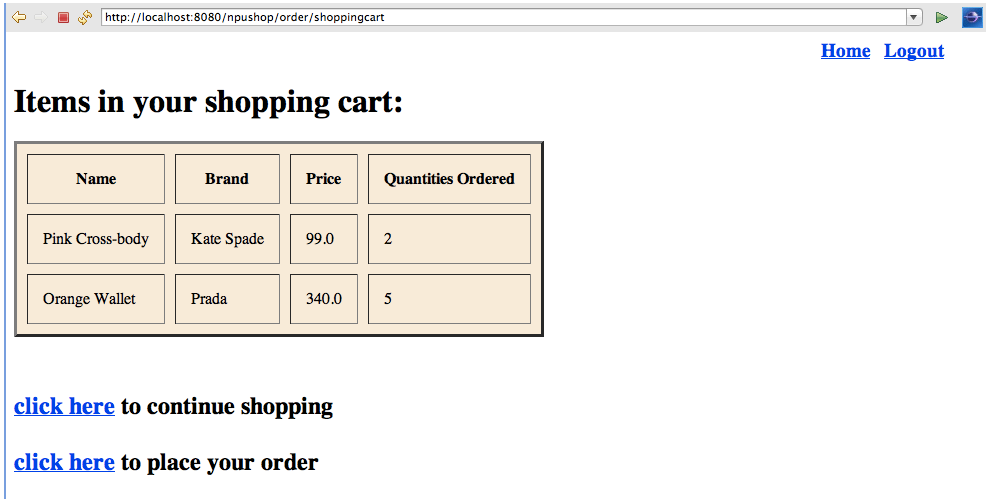
3) Product List View:



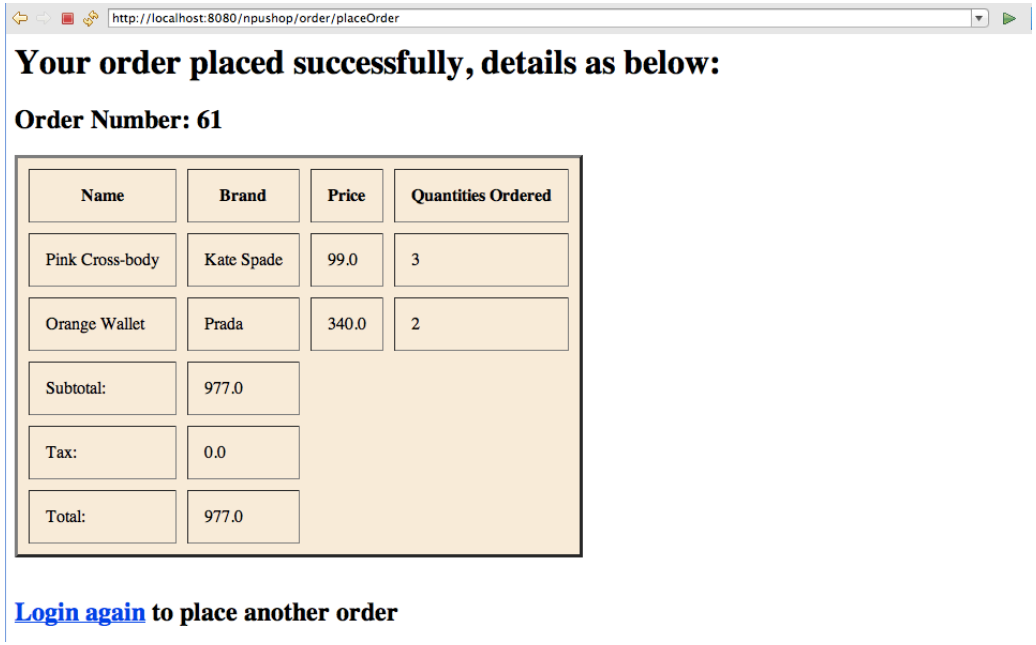
4) Product Details View:



5) Shopping Cart View:



6) Placing Order successfully:



7) Error Page: