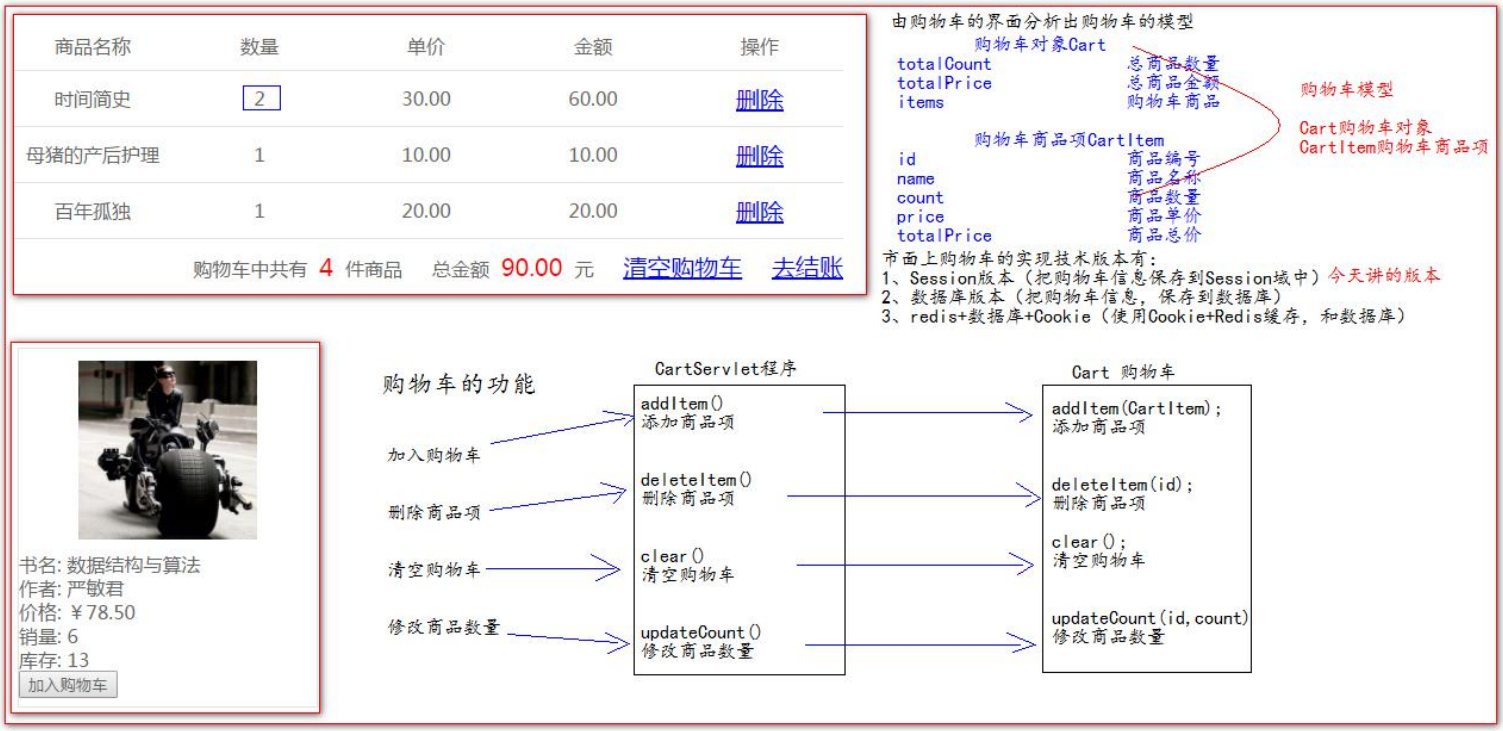
14-书城项目第六、七阶段

讲师：王振国

今日任务

1、项目第六阶段：购物车

1.1、购物车模块分析



1.2、购物车模型编写

1.2.1、购物车模型：

```
/**
 * 购物车的商品项
 */
public class CartItem {
    private Integer id;
```

```
    private String name;
    private Integer count;
    private BigDecimal price;
    private BigDecimal totalPrice;

/**
 * 购物车对象
 */
public class Cart {
//    private Integer totalCount;
//    private BigDecimal totalPrice;

    /**
     * key 是商品编号,
     * value, 是商品信息
     */
    private Map<Integer,CartItem> items = new LinkedHashMap<Integer,CartItem>();

    /**
     * 添加商品项
     *
     * @param cartItem
     */
    public void addItem(CartItem cartItem) {
        // 先查看购物车中是否已经添加过此商品，如果已添加，则数量累加，总金额更新，如果没有添加过，直接放到集合中即可
        CartItem item = items.get(cartItem.getId());

        if (item == null) {
            // 之前没添加过此商品
            items.put(cartItem.getId(), cartItem);
        } else {
            // 已经 添加过的情况
            item.setCount( item.getCount() + 1 ); // 数量 累加
            item.setTotalPrice( item.getPrice().multiply(new BigDecimal( item.getCount() )) ); // 更新总金额
        }

    }

    /**
     * 删除商品项
     */
    public void deleteItem(Integer id) {
        items.remove(id);
    }


    /**
```

```
    * 清空购物车
    */
    public void clear() {
        items.clear();
    }

    /**
    * 修改商品数量
    */
    public void updateCount(Integer id,Integer count) {
        // 先查看购物车中是否有此商品。如果有，修改商品数量，更新总金额
        CartItem cartItem = items.get(id);
        if (cartItem != null) {
            cartItem.setCount(count);// 修改商品数量
            cartItem.setTotalPrice( cartItem.getPrice().multiply(new
BigDecimal( cartItem.getCount() )) ); // 更新总金额
        }
    }


    public Integer getTotalCount() {
        Integer totalCount = 0;

        for (Map.Entry<Integer,CartItem>entry : items.entrySet()) {
            totalCount += entry.getValue().getCount();
        }

        return totalCount;
    }


    public BigDecimal getTotalPrice() {
        BigDecimal totalPrice = new BigDecimal(0);

        for (Map.Entry<Integer,CartItem>entry : items.entrySet()) {
            totalPrice = totalPrice.add(entry.getValue().getTotalPrice());
        }

        return totalPrice;
    }


    public Map<Integer, CartItem> getItems() {
        return items;
    }

    public void setItems(Map<Integer, CartItem> items) {
        this.items = items;
    }
```

```
    @Override
    public String toString() {
        return "Cart{" +
                "totalCount=" + getTotalCount() +
                ", totalPrice=" + getTotalPrice() +
                ", items=" + items +
                '}';
    }
}
```

1.2.2、购物车的测试：

```
public class CartTest {

    @Test
    public void addItem() {
        Cart cart = new Cart();

        cart.addItem(new CartItem(1, "java从入门到精通", 1, new BigDecimal(1000),new BigDecimal(1000)));
        cart.addItem(new CartItem(1, "java从入门到精通", 1, new BigDecimal(1000),new BigDecimal(1000)));
        cart.addItem(new CartItem(2, "数据结构与算法", 1, new BigDecimal(100),new BigDecimal(100)));

        System.out.println(cart);

    }

    @Test
    public void deleteItem() {

        Cart cart = new Cart();

        cart.addItem(new CartItem(1, "java从入门到精通", 1, new BigDecimal(1000),new BigDecimal(1000)));
        cart.addItem(new CartItem(1, "java从入门到精通", 1, new BigDecimal(1000),new BigDecimal(1000)));
        cart.addItem(new CartItem(2, "数据结构与算法", 1, new BigDecimal(100),new BigDecimal(100)));

        cart.deleteItem(1);

        System.out.println(cart);

    }

    @Test
    public void clear() {
        Cart cart = new Cart();
```

```
        cart.addItem(new CartItem(1, "java从入门到精通", 1, new BigDecimal(1000),new BigDecimal(1000)));
        cart.addItem(new CartItem(1, "java从入门到精通", 1, new BigDecimal(1000),new BigDecimal(1000)));
        cart.addItem(new CartItem(2, "数据结构与算法", 1, new BigDecimal(100),new BigDecimal(100)));

        cart.deleteItem(1);

        cart.clear();

        System.out.println(cart);
    }

    @Test
    public void updateCount() {

        Cart cart = new Cart();

        cart.addItem(new CartItem(1, "java从入门到精通", 1, new BigDecimal(1000),new BigDecimal(1000)));
        cart.addItem(new CartItem(1, "java从入门到精通", 1, new BigDecimal(1000),new BigDecimal(1000)));
        cart.addItem(new CartItem(2, "数据结构与算法", 1, new BigDecimal(100),new BigDecimal(100)));

        cart.deleteItem(1);

        cart.clear();

        cart.addItem(new CartItem(1, "java从入门到精通", 1, new BigDecimal(1000),new BigDecimal(1000)));

        cart.updateCount(1, 10);


        System.out.println(cart);

    }
}
```

1.3、加入购物车功能的实现

CartServlet 程序中的代码：

```
/**
 * 加入购物车
 * @param req
 * @param resp
 * @throws ServletException
 * @throws IOException
 */
protected void addItem(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException {
```

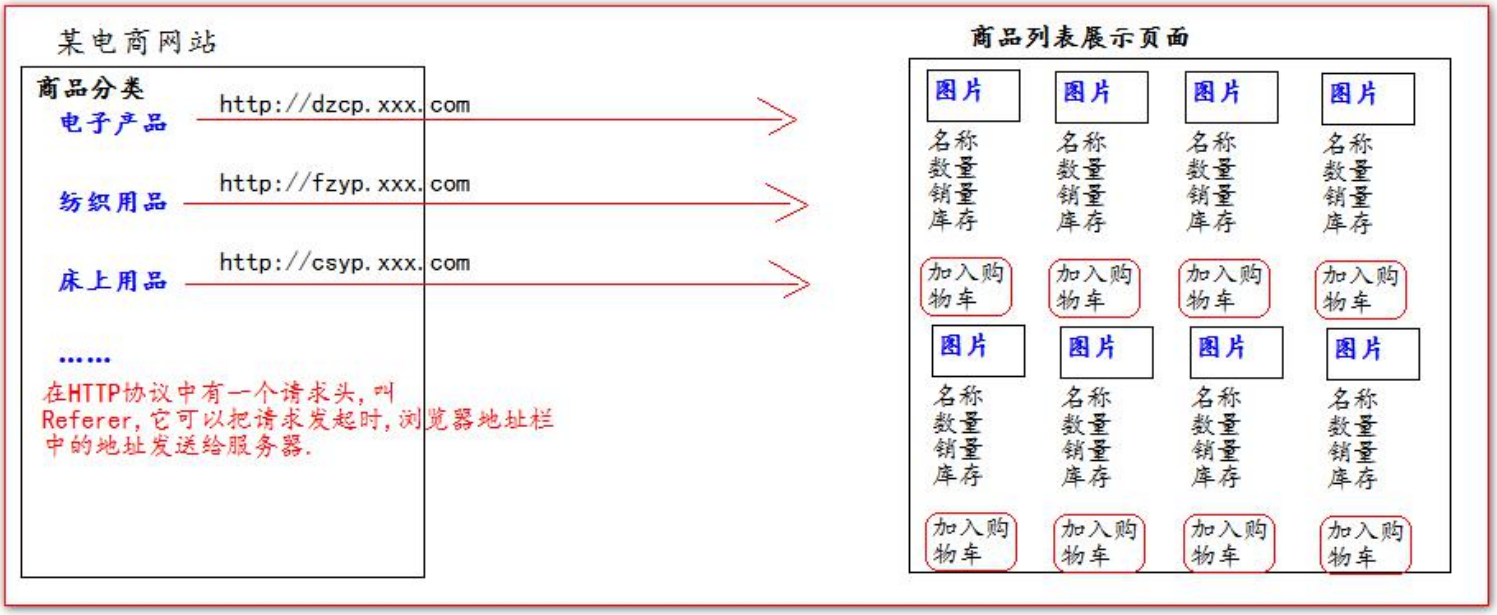
```
    // 获取请求的参数 商品编号
    int id = WebUtils.parseInt(req.getParameter("id"), 0);
    // 调用 bookService.queryBookById(id):Book 得到图书的信息
    Book book = bookService.queryBookById(id);
    // 把图书信息，转换成为 CartItem 商品项
    CartItem cartItem = new CartItem(book.getId(),book.getName(),1,book.getPrice(),book.getPrice());
    // 调用 Cart.addItem(CartItem);添加商品项
    Cart cart = (Cart) req.getSession().getAttribute("cart");
    if (cart == null) {
        cart = new Cart();
        req.getSession().setAttribute("cart",cart);
    }
    cart.addItem(cartItem);

    System.out.println(cart);

    System.out.println("请求头 Referer 的值： " + req.getHeader("Referer"));

    // 重定向回原来商品所在的地址页面
    resp.sendRedirect(req.getHeader("Referer"));
}
```

index.jsp 页面 js 的代码：

```
            <span class="sp2">${book.stock}</span>
        </div>
        <div class="book_add">
            <button bookId="${book.id}" class="addToCart">加入购物车</button>
        </div>
    </div>
div>
```

```
<Script type="text/javascript">
  $(function () {
    // 给加入购物车按钮绑定单击事件
    $("button.addToCart").click(function () {
      /**
       * 在事件响应的function 函数 中，有一个this 对象，这个this 对象，是当前正在响应事件的dom 对象
       * @type {jQuery}
       */
      var bookId = $(this).attr("bookId");
      location.href = "http://localhost:8080/book/cartServlet?action=addItem&id=" + bookId;

    });
  });
</Script>
```

图解说明，如何跳回添加商品的页面：



1.4、购物车的展示

```
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title>购物车</title>

   <%-- 静态包含 base 标签、css 样式、jQuery 文件 --%>
   <%@ include file="/pages/common/head.jsp"%>

</head>
<body>
   <div id="header">
         <img class="logo_img" alt="" src="static/img/logo.gif" >
         <span class="wel_word">购物车</span>

      <%--静态包含，登录 成功之后的菜单 --%>
      <%@ include file="/pages/common/login_success_menu.jsp"%>

   </div>

   <div id="main">

      <table>
         <tr>
            <td>商品名称</td>
            <td>数量</td>
```

```
            <td>单价</td>
            <td>金额</td>
            <td>操作</td>
        </tr>
        <c:if test="${empty sessionScope.cart.items}">
            <%--如果购物车空的情况--%>
            <tr>
                <td colspan="5"><a href="index.jsp">亲，当前购物车为空！快跟小伙伴们去浏览商品吧！！！</a>
</td>
            </tr>
        </c:if>
        <c:if test="${not empty sessionScope.cart.items}">
            <%--如果购物车非空的情况--%>
            <c:forEach items="${sessionScope.cart.items}" var="entry">
                <tr>
                    <td>${entry.value.name}</td>
                    <td>${entry.value.count}</td>
                    <td>${entry.value.price}</td>
                    <td>${entry.value.totalPrice}</td>
                    <td><a href="#">删除</a></td>
                </tr>
            </c:forEach>
        </c:if>
    </table>
    <%--如果购物车非空才输出页面的内容--%>
    <c:if test="${not empty sessionScope.cart.items}">
        <div class="cart_info">
            <span class="cart_span">购物车中共有<span
class="b_count">${sessionScope.cart.totalCount}</span>件商品</span>
            <span class="cart_span">总金额<span
class="b_price">${sessionScope.cart.totalPrice}</span>元</span>
            <span class="cart_span"><a href="#">清空购物车</a></span>
            <span class="cart_span"><a href="pages/cart/checkout.jsp">去结账</a></span>
        </div>
    </c:if>

  </div>
  <%--静态包含页脚内容--%>
  <%@include file="/pages/common/footer.jsp"%>
</body>
</html>
```

1.5、删除购物车商品项

CartServlet 程序：

```
/**
 * 删除商品项
 * @param req
 * @param resp
 * @throws ServletException
 * @throws IOException
 */
protected void deleteItem(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException{
    // 获取商品编号
    int id = WebUtils.parseInt(req.getParameter("id"), 0);
    // 获取购物车对象
    Cart cart = (Cart) req.getSession().getAttribute("cart");

    if (cart != null) {
        // 删除 了购物车商品项
        cart.deleteItem(id);
        // 重定向回原来购物车展示页面
        resp.sendRedirect(req.getHeader("Referer"));
    }

}
```

购物车/pages/cart/cart.jsp 页面的代码：

删除的请求地址：

```
    <td>${entry.value.totalPrice}</td>
    <td><a class="deleteItem" href="cartServlet?action=deleteItem&id=${entry.value.id}">删除</a></td>
</tr>
```

删除的确认提示操作：

```
<script type="text/javascript">
   $(function () {
      // 给 【删除】绑定单击事件
      $("a.deleteItem").click(function () {
         return confirm("你确定要删除【" + $(this).parent().parent().find("td:first").text() +"】吗?")
      });
   });
</script>
```

1.6、清空购物车

CartServlet 程序

```
/**
 * 清空购物车
 * @param req
 * @param resp
```

```
 * @throws ServletException
 * @throws IOException
 */
protected void clear(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException{
    // 1 获取购物车对象
    Cart cart = (Cart) req.getSession().getAttribute("cart");
    if (cart != null) {
        // 清空购物车
        cart.clear();
        // 重定向回原来购物车展示页面
        resp.sendRedirect(req.getHeader("Referer"));
    }
}
```

cart.jsp 页面的内容

给清空购物车添加请求地址，和添加 id 属性：

```
"cart_span">购物车中共有<span class="b_count">${sessionScope.cart.totalCount}</span>件商品</
"cart_span">总金额<span class="b_price">${sessionScope.cart.totalPrice}</span>元</span>
"cart_span"><a id="clearCart" href="cartServlet?action=clear">清空购物车</a></span>
"cart_span"><a href="pages/cart/checkout.jsp">去结账</a></span>
```

清空的确认提示操作：

```
// 给清空购物车绑定单击事件
$("#clearCart").click(function () {
   return confirm("你确定要清空购物车吗?");
})
```

1.7、修改购物车商品数量

CartServlet 程序

```
/**
 * 修改商品数量
 * @param req
 * @param resp
 * @throws ServletException
 * @throws IOException
 */
protected void updateCount(HttpServletRequest req, HttpServletResponse resp) throws ServletException,
IOException{
    // 获取请求的参数 商品编号 、商品数量
    int id = WebUtils.parseInt(req.getParameter("id"),0);
    int count = WebUtils.parseInt(req.getParameter("count"), 1);
```

```
    // 获取 Cart 购物车对象
    Cart cart = (Cart) req.getSession().getAttribute("cart");

    if (cart != null) {
        // 修改商品数量
        cart.updateCount(id,count);
        // 重定向回原来购物车展示页面
        resp.sendRedirect(req.getHeader("Referer"));
    }
}
```

修改 pages/cart/cart.jsp 购物车页面：

```
<c:forEach items="${sessionScope.cart.items}" var="entry">
    <tr>
        <td>${entry.value.name}</td>
        <td>
            <input class="updateCount" style="width: 80px;"
                   bookId="${entry.value.id}"
                   type="text" value="${entry.value.count}">
        </td>
        <td>${entry.value.price}</td>
```

修改商品数量 js 代码：

```
// 给输入框绑定 onchange 内容发生改变事件
$(".updateCount").change(function () {
    // 获取商品名称
    var name = $(this).parent().parent().find("td:first").text();
    var id = $(this).attr('bookId');
    // 获取商品数量
    var count = this.value;
    if ( confirm("你确定要将【" + name + "】商品修改数量为：" + count + " 吗?") ) {
        //发起请求。给服务器保存修改
        location.href =
"http://localhost:8080/book/cartServlet?action=updateCount&count="+count+"&id="+id;
    } else {
        // defaultValue 属性是表单项 Dom 对象的属性。它表示默认的 value 属性值。
        this.value = this.defaultValue;
    }
});
```

1.8、首页，购物车数据回显

在添加商品到购物车的时候，保存最后一个添加的商品名称：

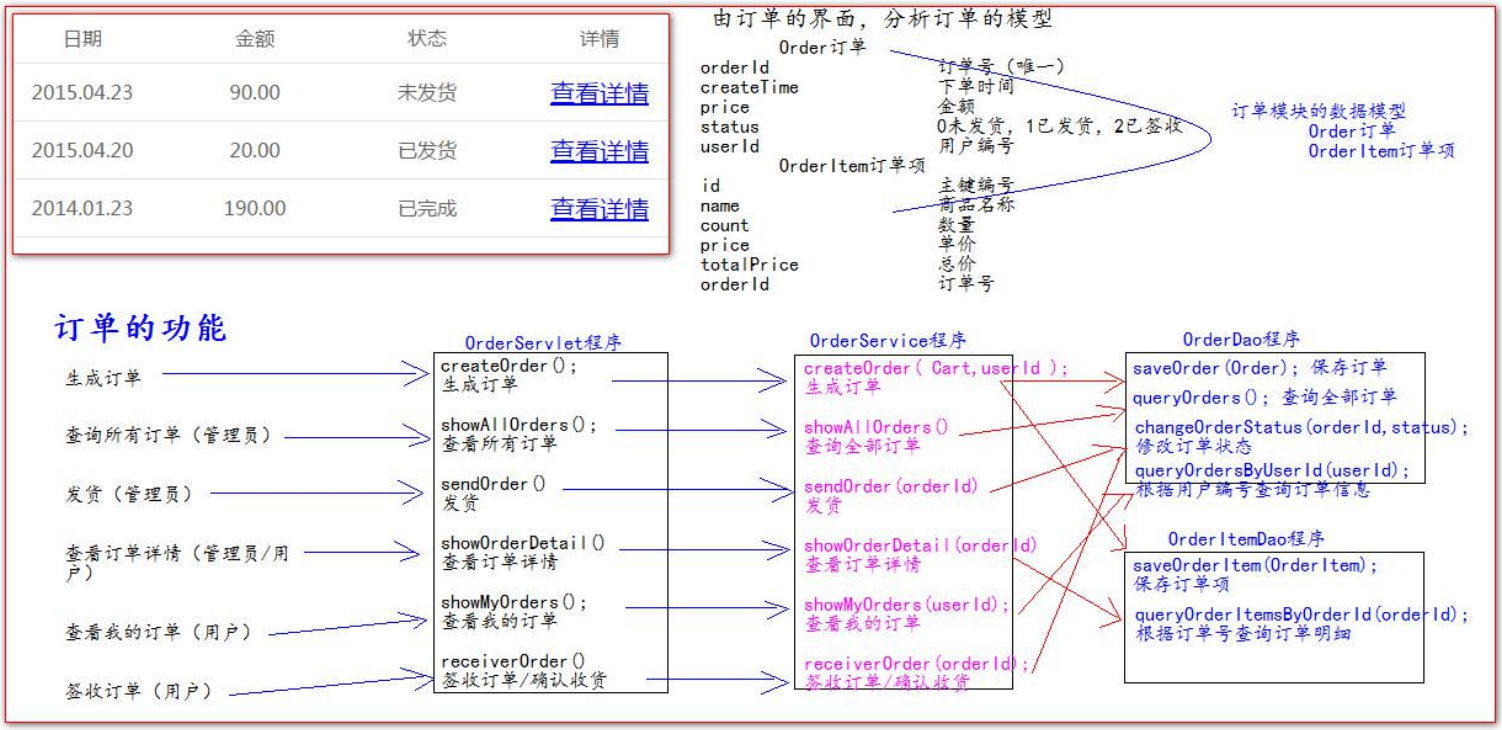
```
cart.addItem(cartItem);

System.out.println(cart);
System.out.println("请求头Referer的值：" + req.getHeader( s: "Referer"));
// 最后一个添加的商品名称
req.getSession().setAttribute( s: "lastName", cartItem.getName());

// 重定向回原来商品所在的地址页面
```

在 pages/client/index.jsp 页面中输出购物车信息：

```
<div style="text-align: center">
  <c:if test="${empty sessionScope.cart.items}">
    <%--购物车为空的输出--%>
    <span> </span>
    <div>
      <span style="color: red">当前购物车为空</span>
    </div>
  </c:if>
  <c:if test="${not empty sessionScope.cart.items}">
    <%--购物车非空的输出--%>
    <span>您的购物车中有 ${sessionScope.cart.totalCount} 件商品</span>
    <div>
      您刚刚将<span style="color: red">${sessionScope.lastName}</span>加入到了购物车中
    </div>
  </c:if>
</div>
```

2、项目第七阶段：订单

2.1、订单模块的分析：



2.2：订单模块的实现

2.2.1、创建订单模块的数据库表

```
use book;

create table t_order(
   `order_id` varchar(50) primary key,
   `create_time` datetime,
   `price` decimal(11,2),
   `status` int,
   `user_id` int,
   foreign key(`user_id`) references t_user(`id`)
);

create table t_order_item(
   `id` int primary key auto_increment,
   `name` varchar(100),
   `count` int,
   `price` decimal(11,2),
   `total_price` decimal(11,2),
```

```
  `order_id` varchar(50),
  foreign key(`order_id`) references t_order(`order_id`)
);
```

2.2.2、创建订单模块的数据模型

```
/**
 * 订单
 */
public class Order {
    private String orderId;
    private Date createTime;
    private BigDecimal price;
    // 0 未发货，1 已发货，2 表示已签收
    private Integer status = 0;
    private Integer userId;


/**
 * 订单项
 */
public class OrderItem {
    private Integer id;
    private String name;
    private Integer count;
    private BigDecimal price;
    private BigDecimal totalPrice;
    private String orderId;
```

2.2.3、编写订单模块的 Dao 程序和测试

OrderDao 接口

```
public interface OrderDao {
    public int saveOrder(Order order);
}
```

OrderDao 实现

```
public class OrderDaoImpl extends BaseDao implements OrderDao {
    @Override
    public int saveOrder(Order order) {
        String sql = "insert into t_order(`order_id`,`create_time`,`price`,`status`,`user_id`)
values(?,?,?,?,?)";

        return
update(sql,order.getOrderId(),order.getCreateTime(),order.getPrice(),order.getStatus(),order.getUs
erId());
    }
}
```

OrderItemDao 接口

```
public interface OrderItemDao {
    public int saveOrderItem(OrderItem orderItem);
}
```

OrderItemDao 实现

```
public class OrderItemDaoImpl extends BaseDao implements OrderItemDao {
    @Override
    public int saveOrderItem(OrderItem orderItem) {
        String sql = "insert into t_order_item(`name`,`count`,`price`,`total_price`,`order_id`)
values(?,?,?,?,?)";
        return
update(sql,orderItem.getName(),orderItem.getCount(),orderItem.getPrice(),orderItem.getTotalPrice(),
orderItem.getOrderId());
    }
}
```

测试

```
public class OrderDaoTest {

    @Test
    public void saveOrder() {

        OrderDao orderDao = new OrderDaoImpl();

        orderDao.saveOrder(new Order("1234567891",new Date(),new BigDecimal(100),0, 1));

    }
}


public class OrderItemDaoTest {

    @Test
    public void saveOrderItem() {
        OrderItemDao orderItemDao = new OrderItemDaoImpl();

        orderItemDao.saveOrderItem(new OrderItem(null,"java 从入门到精通", 1,new BigDecimal(100),new
BigDecimal(100),"1234567890"));
        orderItemDao.saveOrderItem(new OrderItem(null,"javaScript 从入门到精通", 2,new
BigDecimal(100),new BigDecimal(200),"1234567890"));
        orderItemDao.saveOrderItem(new OrderItem(null,"Netty 入门", 1,new BigDecimal(100),new
BigDecimal(100),"1234567890"));

    }
}
```

2.2.4、编写订单模块的 Service 和测试

OrderService 接口

```
public interface OrderService {
    public String createOrder(Cart cart,Integer userId);
}
```

OrderService 实现类

```
public class OrderServiceImpl implements OrderService {

    private OrderDao orderDao = new OrderDaoImpl();
    private OrderItemDao orderItemDao = new OrderItemDaoImpl();

    @Override
    public String createOrder(Cart cart, Integer userId) {
        // 订单号===唯一性
        String orderId = System.currentTimeMillis()+""+userId;
        // 创建一个订单对象
        Order order = new Order(orderId,new Date(),cart.getTotalPrice(), 0,userId);
        // 保存订单
        orderDao.saveOrder(order);

        // 遍历购物车中每一个商品项转换成为订单项保存到数据库
        for (Map.Entry<Integer, CartItem>entry : cart.getItems().entrySet()){
            // 获取每一个购物车中的商品项
            CartItem cartItem = entry.getValue();
            // 转换为每一个订单项
            OrderItem orderItem = new
OrderItem(null,cartItem.getName(),cartItem.getCount(),cartItem.getPrice(),cartItem.getTotalPrice(),
orderId);
            // 保存订单项到数据库
            orderItemDao.saveOrderItem(orderItem);

        }
        // 清空购物车
        cart.clear();

        return orderId;
    }
}
```

测试

```
public class OrderServiceTest {

    @Test
    public void createOrder() {
```

```
        Cart cart = new Cart();

        cart.addItem(new CartItem(1, "java从入门到精通", 1, new BigDecimal(1000),new BigDecimal(1000)));
        cart.addItem(new CartItem(1, "java从入门到精通", 1, new BigDecimal(1000),new BigDecimal(1000)));
        cart.addItem(new CartItem(2, "数据结构与算法", 1, new BigDecimal(100),new BigDecimal(100)));

        OrderService orderService = new OrderServiceImpl();

        System.out.println( "订单号是：" + orderService.createOrder(cart, 1) );

    }
}
```

2.2.5、编写订单模块的 web 层和页面联调

修改 OrderService 程序：

```
public class OrderServiceImpl implements OrderService {

    private OrderDao orderDao = new OrderDaoImpl();
    private OrderItemDao orderItemDao = new OrderItemDaoImpl();
    private BookDao bookDao = new BookDaoImpl();

    @Override
    public String createOrder(Cart cart, Integer userId) {
        // 订单号===唯一性
        String orderId = System.currentTimeMillis()+""+userId;
        // 创建一个订单对象
        Order order = new Order(orderId,new Date(),cart.getTotalPrice(), 0,userId);
        // 保存订单
        orderDao.saveOrder(order);

        // 遍历购物车中每一个商品项转换成为订单项保存到数据库
        for (Map.Entry<Integer, CartItem>entry : cart.getItems().entrySet()){
            // 获取每一个购物车中的商品项
            CartItem cartItem = entry.getValue();
            // 转换为每一个订单项
            OrderItem orderItem = new
OrderItem(null,cartItem.getName(),cartItem.getCount(),cartItem.getPrice(),cartItem.getTotalPrice(),
orderId);
            // 保存订单项到数据库
            orderItemDao.saveOrderItem(orderItem);

            // 更新库存和销量
            Book book = bookDao.queryBookById(cartItem.getId());
            book.setSales( book.getSales() + cartItem.getCount() );
            book.setStock( book.getStock() - cartItem.getCount() );
```

```
            bookDao.updateBook(book);

        }
        // 清空购物车
        cart.clear();

        return orderId;
    }
}
```

OrderServlet 程序：

```
public class OrderServlet extends BaseServlet {

    private OrderService orderService = new OrderServiceImpl();

    /**
     * 生成订单
     *
     * @param req
     * @param resp
     * @throws ServletException
     * @throws IOException
     */
    protected void createOrder(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        // 先获取 Cart 购物车对象
        Cart cart = (Cart) req.getSession().getAttribute("cart");
        // 获取 Userid
        User loginUser = (User) req.getSession().getAttribute("user");

        if (loginUser == null) {
            req.getRequestDispatcher("/pages/user/login.jsp").forward(req,resp);
            return;
        }

        Integer userId = loginUser.getId();
//        调用 orderService.createOrder(Cart,Userid);生成订单
        String orderId = orderService.createOrder(cart, userId);

//        req.setAttribute("orderId", orderId);
        // 请求转发到/pages/cart/checkout.jsp
//        req.getRequestDispatcher("/pages/cart/checkout.jsp").forward(req, resp);

        req.getSession().setAttribute("orderId",orderId);

        resp.sendRedirect(req.getContextPath()+"/pages/cart/checkout.jsp");
    }
```

```
}
```

修改 pages/cart/cart.jsp 页面，结账的请求地址：

```
    <span class="cart_span">总金额<span class="b_price">${sessionScope.cart.totalPrice}</span>元<
    <span class="cart_span"><a id="clearCart" href="cartServlet?action=clear">清空购物车</a></spa
    <span class="cart_span"><a href="orderServlet?action=createOrder">去结账</a></span>
</div>
```

修改 pages/cart/checkout.jsp 页面，输出订单号：

```
<div id="main">

    <h1>你的订单已结算，订单号为: ${sessionScope.orderId}</h1>
```