# 网上商城实战篇

## 今日内容介绍

- 后台商品管理
- 后台订单管理

## 今日内容学习目标

- JavaWeb 知识巩固



# 第1章 后台订单管理

## 1.1 查询订单

### 1.1.1 准备工作

- 步骤 1：编写 Servlet 实现类

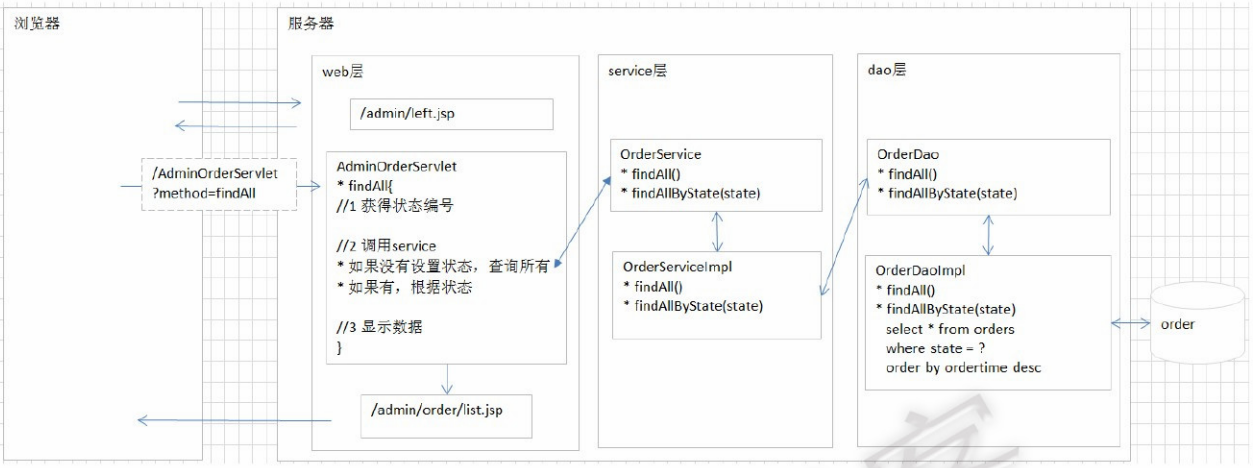
```
public class AdminOrderServlet extends BaseServlet {
    private static final long serialVersionUID = 1L;


}
```

- 步骤 2：web.xml 配置 servlet

```
<servlet>
    <servlet-name>AdminOrderServlet</servlet-name>
    <servlet-class>cn.itcast.store.web.servlet.AdminOrderServlet</servlet-class>
</servlet>
<servlet-mapping>
    <servlet-name>AdminOrderServlet</servlet-name>
    <url-pattern>/AdminOrderServlet</url-pattern>
</servlet-mapping>
```

## 1.1.2 分析




## 1.1.3 代码实现

- 步骤 1：修改 list.jsp 页面，进行查询所有订单的操作
  - /store_v1.0/WebContent/admin/left.jsp

```
d.add('010501','0105','订单管理','${pageContext.request.contextPath}/AdminOrderServlet?method=findAll','','mainFrame'
d.add('010502','0105','未付款的订单','${pageContext.request.contextPath}/AdminOrderServlet?method=findAll&state=1',''
d.add('010503','0105','已付款订单','${pageContext.request.contextPath}/AdminOrderServlet?method=findAll&state=2','',''
d.add('010504','0105','已发货的订单','${pageContext.request.contextPath}/AdminOrderServlet?method=findAll&state=3',''
d.add('010505','0105','已完成的订单','${pageContext.request.contextPath}/AdminOrderServlet?method=findAll&state=4',''
```

- 步骤 2：修改 AdminOrderServlet，添加 findAll 方法

```
//查询所有
public String findAll(HttpServletRequest request, HttpServletResponse response)
        throws Exception {

    //1 获得状态编号，从而确定查询的数据
    String value = request.getParameter("state");

    //2 业务操作
    OrderService orderService = new OrderServiceImpl();
    List<Order> list = null;
    if(value == null){
        // 查询所有
        list = orderService.findAll();
    }else{
        // 查询某个状态的订单
        int state = Integer.parseInt(value);
```

```
            list = orderService.findByState(state);
        }

        //3 显示数据
        request.setAttribute("list", list);
        return "/admin/order/list.jsp";
    }
```

- 步骤 3：修改 OrderService，添加 findAll 和 findByState 方法

```
    //接口
    //查询所有订单
    List<Order> findAll()throws Exception;
    //根据状态查询所有订单
    List<Order> findByState(int state) throws Exception;
```

```
    //实现类
    @Override
    public List<Order> findAll()throws Exception {
        return orderDao.findAll();
    }

    @Override
    public List<Order> findByState(int state) throws Exception {
        return orderDao.findByState(state);
    }
```

- 步骤 4：修改 OrderDao，添加 findAll 和 findByState 方法

```
    //接口
    //查询所有
    List<Order> findAll() throws Exception;
    //根据状态查询所有
    List<Order> findByState(int state) throws Exception;
```

```
    //实现类
    @Override
    public List<Order> findAll() throws Exception {
        QueryRunner queryRunner = new QueryRunner(JDBCUtils.getDataSource());
        String sql = "select * from orders order by ordertime desc";
        List<Order>         list         =         queryRunner.query(sql,         new
BeanListHandler<Order>(Order.class));
        return list;
    }

    @Override
```

```
    public List<Order> findByState(int state) throws Exception {
        QueryRunner queryRunner = new QueryRunner(JDBCUtils.getDataSource());
        String sql = "select * from orders where state = ? order by ordertime desc";
        List<Order>         list         =         queryRunner.query(sql,         new
BeanListHandler<Order>(Order.class),state);
        return list;
    }
```

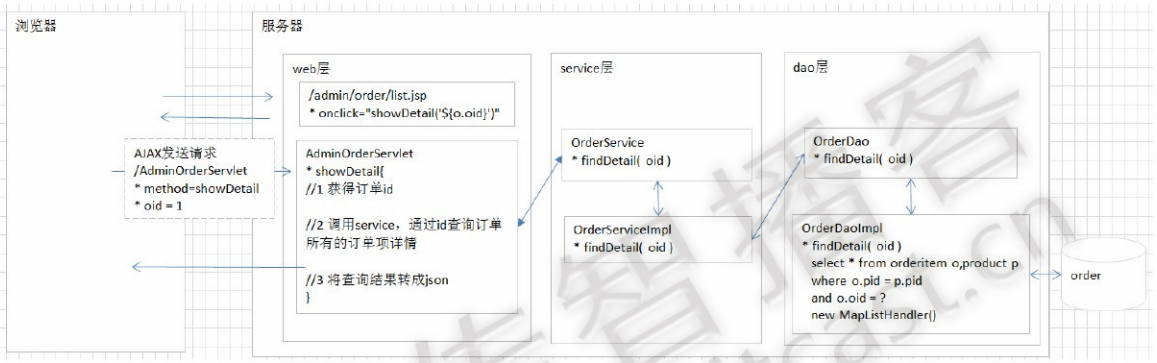
- 步骤 5：修改 list.jsp 页面
  - /store_v1.0/WebContent/admin/order/list.jsp

```
    <c:forEach var="o" items="${ list }" varStatus="status">
        <tr onmouseover="this.style.backgroundColor = 'white'"
            onmouseout="this.style.backgroundColor = '#F5FAFE';">
            <td style="CURSOR: hand; HEIGHT: 22px" align="center"
                width="18%">
                ${ status.count }
            </td>
            <td style="CURSOR: hand; HEIGHT: 22px" align="center"
                width="17%">
                ${ o.oid }
            </td>
            <td style="CURSOR: hand; HEIGHT: 22px" align="center"
                width="17%">
                ${ o.total }
            </td>
            <td style="CURSOR: hand; HEIGHT: 22px" align="center"
                width="17%">
                ${ o.name }
            </td>
            <td style="CURSOR: hand; HEIGHT: 22px" align="center"
                width="17%">
                    <c:if test="${ o.state == 1 }">
                        未付款
                    </c:if>
                    <c:if test="${ o.state == 2 }">
                        <a href="${ pageContext.request.contextPath }/#">发货</a>
                    </c:if>
                    <c:if test="${ o.state == 3 }">
                        未确认收费
                    </c:if>
                    <c:if test="${ o.state == 4 }">
                        订单结束
                    </c:if>
            </td>
```

```
            <td align="center" style="HEIGHT: 22px">
                <input type="button" value="订单详情" />
            </td>
        </tr>
    </c:forEach>
```

# 1.2 异步加载显示详情

## 1.2.1 分析




## 1.2.2 代码实现

- 步骤 1：修改 list.jsp，确定“订单详情”显示区域
  - /store_v1.0/WebContent/admin/order/list.jsp

```
    <%-- 触发 ajax 请求 --%>
    <input type="button" value="订单详情" id="but${o.oid}" onclick="showDetail('${o.oid}')"/>
    <table id="tab${ o.oid }">
        <%--显示详情区域 --%>
    </table>
```

- 步骤 2：编写 showDetail() 函数，发送 ajax

```
    <script type="text/javascript">
        function showDetail(oid){
            var $val = $("#but"+oid).val();
            if($val == "订单详情"){
                var url = "${pageContext.request.contextPath}/AdminOrderServlet";
                var params = {"method":"showDetail","oid":oid};
                $.post(url,params,function(data){
```

```
                    $(data).each(function(i,n){
                        // ajax 显示图片,名称,数量,小计
                        $("#tab"+oid).append("<tr><td><img
src='${    pageContext.request.contextPath    }/"+n.product.pimage+"'    width='30'
height='30'></td><td>"+n.product.pname+"</td><td>"+n.count+"</td><td>"+n.subtotal+"
</td></tr>");
                    });
                },"json");

                $("#but"+oid).val("关闭");
            }else{
                $("#tab"+oid).html("");
                $("#but"+oid).val("订单详情");
            }
        }
    </script>
```

- 步骤 3：修改 AdminOrderServlet，添加 showDetail 方法

```
    //ajax 显示订单详情
    public String showDetail(HttpServletRequest request, HttpServletResponse response)
            throws Exception {

        // 接收 oid:
        String oid = request.getParameter("oid");
        // 查询根据 oid:
        OrderService orderService =  new OrderServiceImpl();
        List<OrderItem> list = orderService.findDetail(oid);
        // 响应给浏览器 json 数据
        JSONArray jsonArray = JSONArray.fromObject(list);
        response.setContentType("text/html;charset=UTF-8");
        response.getWriter().println(jsonArray.toString());
        return null;
    }
```

- 步骤 4：修改 OrderService，添加 findDetail 方法

```
    //接口
    //通过 id 查询详情
    List<OrderItem> findDetail(String oid) throws Exception;
```

```
    //实现类
    public List<OrderItem> findDetail(String oid) throws Exception {
        return orderDao.findDetail(oid);
    }
```

- 步骤 5：修改 OrderDao，添加 findDetail 方法

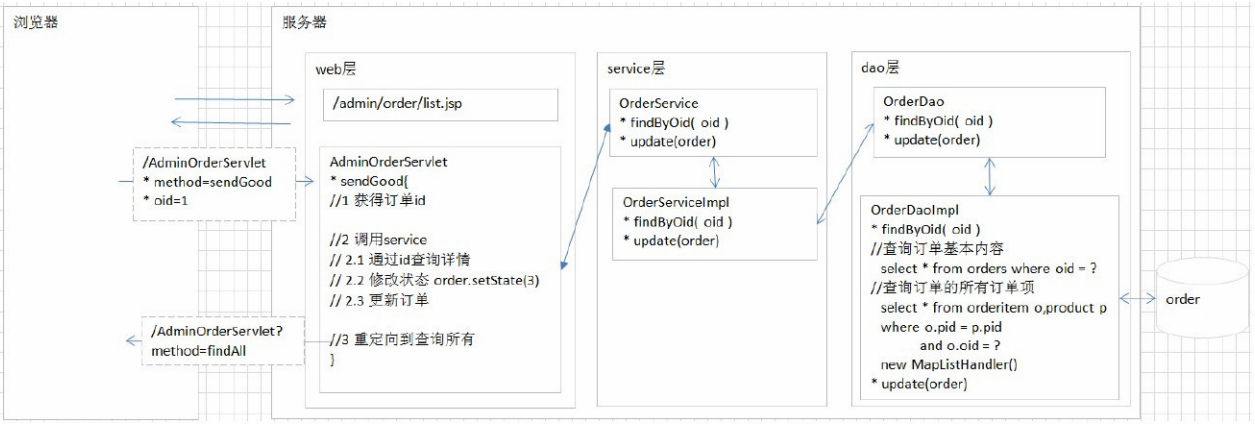
```
    //接口
    //通过 id 查询详情
    List<OrderItem> findDetail(String oid) throws Exception;
```

```
    //实现类
    @Override
    public List<OrderItem> findDetail(String oid) throws Exception {
        QueryRunner queryRunner = new QueryRunner(JDBCUtils.getDataSource());
        String sql = "select * from orderitem o,product p where o.pid = p.pid and o.oid
= ?";
        List<Map<String,Object>> oList = queryRunner.query(sql, new MapListHandler(),
oid);
        List<OrderItem> list = new ArrayList<OrderItem>();
        for (Map<String,Object> map : oList) {
            OrderItem orderItem = new OrderItem();
            BeanUtils.populate(orderItem, map);
            Product product = new Product();
            BeanUtils.populate(product, map);
            orderItem.setProduct(product);

            list.add(orderItem);
        }
        return list;
    }
```

# 1.3 修改订单状态：待发货

## 1.3.1 分析



### 1.3.2 代码实现

- 步骤 1：修改 list.jsp，进行“待发货”操作
  - /store_v1.0/WebContent/admin/order/list.jsp

```
    <c:if test="${ o.state == 2 }">
        <a
href="${ pageContext.request.contextPath }/AdminOrderServlet?method=sendGood&oid=${
o.id}">待发货</a>
    </c:if>
```

- 步骤 2：修改 AdminOrderServlet，编写 sendGood 方法，先查询，然后修改状态，最后更新。

```
    //发货
    public String sendGood(HttpServletRequest request, HttpServletResponse response)
            throws Exception {

        // 接收 oid:
        String oid = request.getParameter("oid");
        // 查询根据 oid:
        OrderService orderService = new OrderServiceImpl();
        Order order = orderService.findByOid(oid);
        order.setState(3);//可以提供常量

        orderService.update(order);

        response.sendRedirect(request.getContextPath()                              +
"/AdminOrderServlet?method=findAll");
        return null;
    }
```

# 第2章 部署

## 2.1 需要

将项目部署到 linux 系统下

传智播客
www.itcast.cn