网上商城实战

今日内容介绍

- 分类管理：查询所有分类
- 商品管理

今日内容学习目标

- JavaWeb知识巩固

1.查询所有分类

1.1 分析

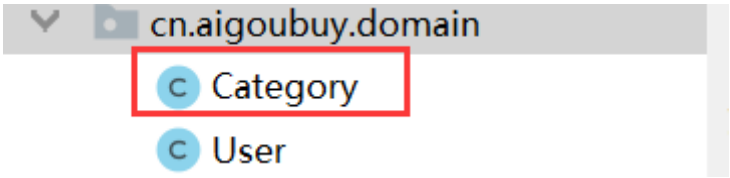
1.2 准备工作

创建数据库、JavaBean、Dao接口和实现类，Service接口和实现类

- 步骤1：创建分类表

```
-- 2.1 创建分类表
CREATE TABLE `category` (
  `cid` varchar(32) NOT NULL,
  `cname` varchar(20) DEFAULT NULL, #分类名称
  PRIMARY KEY (`cid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- 2.2 初始化分类默认数据
INSERT INTO `category` VALUES ('1','手机数码'),
('172934bd636d485c98fd2d3d9cccd409','运动户外'),('2','电脑办公'),('3','家具家居'),('4','鞋靴箱包'),('5','图书音像'),
('59f56ba6ccb84cb591c66298766b83b5','aaaa'),('6','母婴孕婴'),
('afdba41a139b4320a74904485bdb7719','汽车用品');
```

- 步骤2：创建JavaBean之 `Category`




```
public class Category {
    private String cid;
    private String cname;
}
```

- 步骤3：编写dao接口，及实现类

```
/**
```

```
 * 分类模块 dao 层接口
 * @author azzhu
 * @create 2020-07-22 21:13:24
 */
public interface CategoryDao {
}

/**
 * 分类模块 dao 层实现类
 * @author azzhu
 * @create 2020-07-22 21:13:49
 */
public class CategoryDaoImpl implements CategoryDao {
}
```

- 步骤4：编写service接口，及实现类

```
package cn.aigoubuy.service;

/**
 *分类模块 service 层接口
 * @author azzhu
 * @create 2020-07-22 21:14:35
 */
public interface CategoryService {
}


package cn.aigoubuy.service.impl;

import cn.aigoubuy.service.CategoryService;

/**
 * 分类模块 service 层实现类
 * @author azzhu
 * @create 2020-07-22 21:15:10
 */
public class CategoryServiceImpl implements CategoryService {
}
```

1.3 代码实现

- 步骤1：完善 `IndexServlet` ，显示 `/jsp/index.jsp` 前查询分类

```
@WebServlet("/IndexServlet")
public class IndexServlet extends BaseServlet {
    @Override
    public String execute(HttpServletRequest request, HttpServletResponse response) throws Exception {
        // 查询所有的分类
        CategoryService categoryService = new CategoryServiceImpl();
        List<Category> allCategory = categoryService.findAll();

        // 将查询结果存放到request域中
        request.setAttribute("allCategory",allCategory);
        //转发到真实的首页
```

```
            return "/jsp/index.jsp";
        }
    }
```

- 步骤2：完善 `CategoryService` ,添加 `findAll()` 方法

```
// 接口
public interface CategoryService {
    /**
     * 查询所有
     * @return
     */
    List<Category> findAll() throws SQLException;
}

// 实现类
public class CategoryServiceImpl implements CategoryService {
    private CategoryDao categoryDao = new CategoryDaoImpl();
    @Override
    public List<Category> findAll() throws SQLException{
        return categoryDao.findAll();
    }
}
```

- 步骤3：完善 `CategoryDao` ，添加 `findAll()` 方法

```
// 接口
public interface CategoryDao {
    /**
     * 查询所有
     * @return
     * @throws SQLException
     */
    List<Category> findAll() throws SQLException;
}

// 实现类
public class CategoryDaoImpl implements CategoryDao {
    @Override
    public List<Category> findAll() throws SQLException {
        QueryRunner queryRunner = new
QueryRunner(JDBCUtils.getDataSource());
        String sql = "select * from category";
        return queryRunner.query(sql,new BeanListHandler<>(Category.class));
    }
}
```

- 步骤4： `header.jsp` 遍历数据

```
    <li class="active"><a href="${pageContext.request.contextPath}/jsp/product_list.jsp">
    <li><a href="#">电脑办公</a></li>
    <li><a href="#">电脑办公</a></li>
    <li><a href="#">电脑办公</a></li>
```

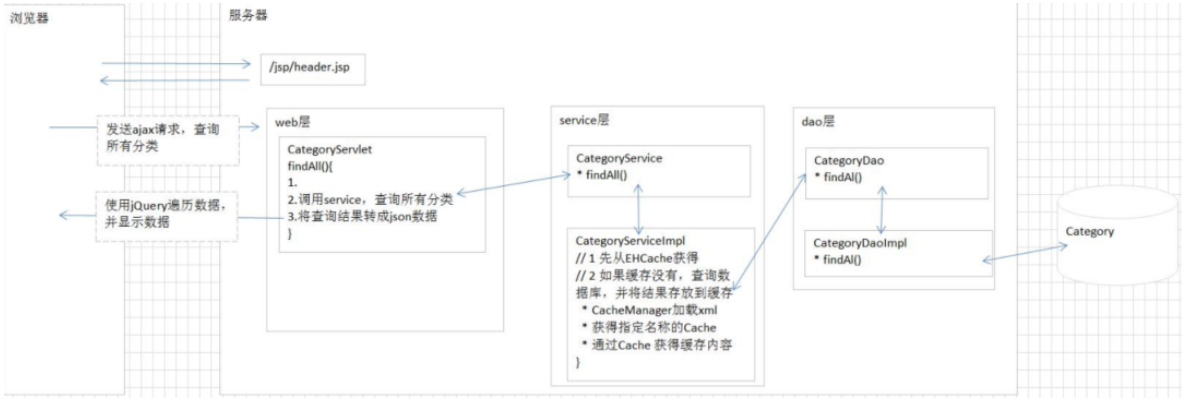
```
<ul class="nav navbar-nav">
    <%--                    <li class="active">
    <a href="${pageContext.request.contextPath}/jsp/product_list.jsp">手机数码
<span class="sr-only">(current)</span></a>
    </li>
<li><a href="#">电脑办公</a></li>
<li><a href="#">电脑办公</a></li>
<li><a href="#">电脑办公</a></li>--%>
<c:forEach items="${allCategory}" var="category">
    <li value="${category.cid}">
        <a href="#">${category.cname}</a>
    </li>
</c:forEach>
</ul>
```

- 步骤5：观察效果

1.4 优化：Ajax异步加载

1.4.1 分析

当访问首页时可以显示分类导航条，但访问其他模块时无法访问显示分类。通过比较程序我们发现，显示首页前我们查询了所有分类，显示登录等其他模块时我们没有查询分类。为了所有模块都可以显示分类，我们需要发送Ajax单独查询分类。



1.4.2 代码实现

引入 `FastJson` 的依赖

```
<dependency>
    <groupId>com.alibaba</groupId>
    <artifactId>fastjson</artifactId>
    <version>1.2.68</version>
</dependency>
```

- 步骤1：修改 `IndexServlet` ，将查询分类代码注释掉

```java
@WebServlet("/IndexServlet")
public class IndexServlet extends BaseServlet {
    @Override
    public String execute(HttpServletRequest request, HttpServletResponse response) thr
        // 查询所有的分类
//      CategoryService categoryService = new CategoryServiceImpl();
//      List<Category> allCategory = categoryService.findAll();
//
//      // 将查询结果存放到request域中
//      request.setAttribute("allCategory",allCategory);
        //转发到真实的首页
        return "/jsp/index.jsp";
    }
}
```

- 步骤2：修改 `header.jsp` 给 `<ul>` 添加id，并注释掉查询所有的遍历内容

```jsp
                <ul class="nav navbar-nav" id="menu">
<%--                    <li class="active">
                        <a href="${pageContext.request.contextPath}/jsp/product_list.jsp">手机数
                    </li>
                    <li><a href="#">电脑办公</a></li>
                    <li><a href="#">电脑办公</a></li>
                    <li><a href="#">电脑办公</a></li>--%>
                   <%-- <c:forEach items="${allCategory}" var="category">
                        <li value="${category.cid}">
                            <a href="#">${category.cname}</a>
                        </li>
                    </c:forEach>--%>
                </ul>
```

- 步骤3：修改 `header.jsp` 添加js函数，页面加载发送ajax查询所有分类

```jsp
<script>
    $(function () {
        var url = "${pageContext.request.contextPath}/categoryServlet";
        $.post(url,{"method":"findAll"},function (data) {
            $.each(data,function (i,n) {
                $("#menu").append("<li value='"+n.cid+"'><a
href='#'>"+n.cname+"</a></li>")
            })
        })
    });
</script>
```

- 步骤4：编写 `CategoryServlet` ，提供 `findAll()` 方法

```java
@WebServlet("/categoryServlet")
public class CategoryServlet extends BaseServlet {
    public String findAll(HttpServletRequest req, HttpServletResponse resp)
throws Exception {
        // 1.1 查询所有的分类
        CategoryService categoryService = new CategoryServiceImpl();
        List<Category> allCategory = categoryService.findAll();

        // 1.2 将查询结果转为json
        String jsonStr = JSONObject.toJSONString(allCategory);

        // 1.3 将结果响应给浏览器
        resp.setContentType("application/json;charset=UTF-8");
        resp.getWriter().println(jsonStr);
        return null;
    }
}
```

1.5 增强：缓存技术

1.5.1 分析

当我们在不同模块之间切换时，发现菜单栏显示的分类数据都是一样的。浏览器每发送一次请求，服务器端都会查询一次数据库，从而对数据库服务器造成不必要的访问。实际开发中，我们采用缓存技术来解决此问题。

1.5.2 相关技术

- 缓存（cache）：通常指的就是内存中的一块空间，介于应用程序和永久性数据存储源（如硬盘上的文件或者数据库）之间，其作用是降低应用程序直接读写永久性数据存储源的频率，从而提高应用的运行性能。
- 常见的缓存技术：
  - redis：是一个key-value存储系统。和Memcached类似，它支持存储的value类型相对更多。
  - Memcached：是一个高性能的分布式内存对象缓存系统，用于动态Web应用以减轻数据库负载。
  - Ehcache：是一个纯Java的进程内缓存框架，具有快速、精干等特点。
- 导入redis依赖：

```
<dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
    <version>3.2.0</version>
</dependency>
```

1.5.3 代码实现

如果缓存中已经有，将直接从缓存获得，如果没有将从数据库获取。修改`CategoryService`代码，给当前查询所有添加业务缓存。

```
    /**
     * 查询分类通过缓存服务器
     * @return
     * @throws Exception
     */
    public String findAllByAjax() throws Exception {
        Jedis j = null;
        String value = null;
        try {
            // 从redis获取分类信息
            //1.获取连接
            j = JedisUtils.getJedis();

            // 2.获取数据，判断数据是否为空
            value = j.get("category_list");

            // 2.1 若不为空，直接返回数据
            if(value != null) {
                System.out.println("缓存中有数据....");
                return value;
            }

            // 2.2 若为空，从mysql数据库中获取，并放入redis中
```

```
            List<Category> clist = findAll();
            // 将clist转为json返回且放入redis中即可
            value = JSONObject.toJSONString(clist);

            // 将value放入redis中
            j.set("category_list",value);
            return value;
        } finally {
            // 释放jedis
            JedisUtils.closeJedis(j);
        }
    }
```

1.5.4 Redis工具类

```
public class JedisUtils {
    //创建连接池
    private static JedisPoolConfig config;
    private static JedisPool pool;

    static{
        config=new JedisPoolConfig();
        config.setMaxTotal(30);
        config.setMaxIdle(2);

        pool=new JedisPool(config, "localhost", 6379);
    }


    //获取连接的方法
    public static Jedis getJedis(){
        return pool.getResource();
    }


    //释放连接
    public static void closeJedis(Jedis j){
        if(j != null) {
            j.close();
        }
    }
}
```

2.前台商品管理

2.1 准备工作

- 步骤1：创建表并完善数据

```
-- 3.1 创建商品表
CREATE TABLE `product` (
  `pid` varchar(32) NOT NULL,
  `pname` varchar(50) DEFAULT NULL,      #商品名称
  `market_price` double DEFAULT NULL,    #市场价
```

```
  `shop_price` double DEFAULT NULL,      #商城价
  `pimage` varchar(200) DEFAULT NULL,    #商品图片路径
  `pdate` date DEFAULT NULL,             #上架时间
  `is_hot` int(11) DEFAULT NULL,         #是否热门：0=不热门,1=热门
  `pdesc` varchar(255) DEFAULT NULL,     #商品描述
  `pflag` int(11) DEFAULT 0,             #商品标记：0=未下架(默认值),1=已经下架
  `cid` varchar(32) DEFAULT NULL,        #分类id
  PRIMARY KEY (`pid`),
  KEY `product_fk_0001` (`cid`),
  CONSTRAINT `product_fk_0001` FOREIGN KEY (`cid`) REFERENCES `category` (`cid`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
-- 3.2 初始化商品默认数据
INSERT INTO `product` VALUES ('1','适用小米note m4小米4c小米3手机屏幕总成寄修维修单独换外屏触摸',1399,1299,'products/1/c_0001.jpg','2015-11-02',1,'小米 4c 标准版 全网通 白色 移动联通电信4G手机 双卡双待',0,'1')
```

参考：《istore_v1.0.sql》

- 步骤2：编写 `Product`

```
public class Product {
    private String pid; //商品编号
    private String pname; //商品名称
    private double market_price;//商品市场价格
    private double shop_price;//商品商场价格
    private String pimage;//商品图片路径
    private Date pdate;//商品上架日期
    private int is_hot;//商品是否热门
    private String pdesc;//商品描述
    private int pflag;//商品是否在货架上  0:在货架上 1:下架
    private String cid;//商品所在分类id   或者给Category，以面向对象的方式描述商品与分类之间的关系
}
```

- 步骤3：编写dao接口，及实现类

```
package cn.aigoubuy.dao;

/**
 * 商品 dao 接口
 * @author azzhu
 * @create 2020-07-22 23:00:15
 */
public interface ProductDao {
}

package cn.aigoubuy.dao.impl;

import cn.aigoubuy.dao.ProductDao;

/**
 * 商品dao实现类
 * @author azzhu
 * @create 2020-07-22 23:00:40
 */
public class ProductDaoImpl implements ProductDao {
}
```

- 步骤4：编写service接口，及实现类

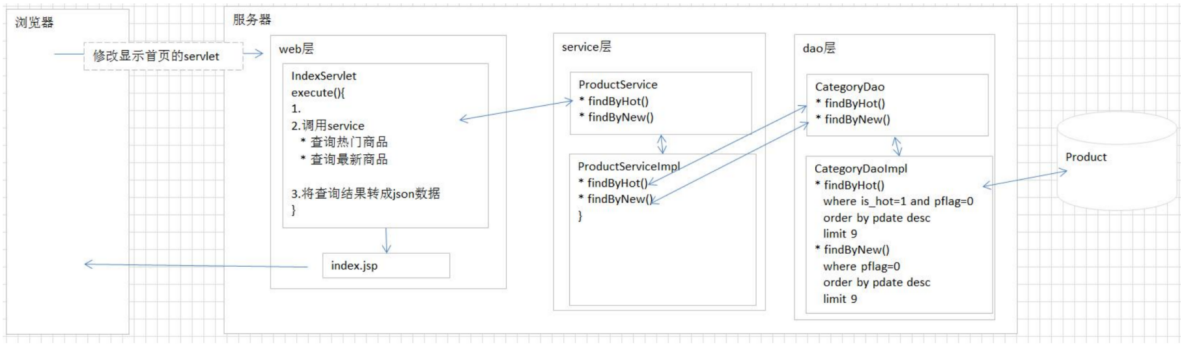
```
package cn.aigoubuy.service;

/**
 * 商品Service接口
 * @author azzhu
 * @create 2020-07-22 23:01:22
 */
public interface ProductService {
}

package cn.aigoubuy.service.impl;

import cn.aigoubuy.service.ProductService;

/**
 * 商品Service实现类
 * @author azzhu
 * @create 2020-07-22 23:01:47
 */
public class ProductServiceImpl implements ProductService {
}
```

- 步骤5：编写Servlet

```
/**
 * @author azzhu
 * @create 2020-07-22 23:02:42
 */
@WebServlet("/productServlet")
public class ProductServlet extends BaseServlet {
}
```

2.2 首页热门商品和最新商品的显示

2.2.1 分析



2.2.2 代码实现

- 步骤1：修改 `IndexServlet` ，添加查询热门和最新商品的查询

```
    @Override
```

```
    public String execute(HttpServletRequest request, HttpServletResponse response) throws Exception {
        ProductService productService = new ProductServiceImpl();
        // 1.1 查询热门商品
        List<Product> hotList = productService.findByHot();

        // 1.2 查询最新商品
        List<Product> newList = productService.findByNew();

        // 2.将查询结果存放
        request.setAttribute("hotList",hotList);
        request.setAttribute("newList",newList);
        return "/jsp/index.jsp";
    }
```

- 步骤 2：修改 service，提供 findByHot()和 findByNew()方法

```
//接口
public interface ProductService {
    /**
     * 热门商品
     * @return
     */
    List<Product> findByHot() throws SQLException;
    /**
     * 最新商品
     * @return
     */
    List<Product> findByNew() throws SQLException;
}

//实现类
public class ProductServiceImpl implements ProductService {
    private ProductDao productdao = new ProductDaoImpl();
    @Override
    public List<Product> findByHot() throws SQLException {
        return productdao.findByHot();
    }

    @Override
    public List<Product> findByNew() throws SQLException{
        return productdao.findByNew();
    }
}
```

- 步骤 3：修改 dao，提供 findByHot()和 findByNew()方法

```
// 接口
public interface ProductDao {
    /**
     * 热门商品
     * @return
     * @throws SQLException
     */
    List<Product> findByHot() throws SQLException;
    /**
     * 最新商品
     * @return
```

```
     * @throws SQLException
     */
    List<Product> findByNew() throws SQLException;
}

//实现类
public class ProductDaoImpl implements ProductDao {
    @Override
    public List<Product> findByHot() throws SQLException {
        QueryRunner queryRunner = new QueryRunner(JDBCUtils.getDataSource());
        String sql = "select * from product where is_hot=? and pflag=? order by pdate desc limit ?";
        return queryRunner.query(sql,new BeanListHandler<>(Product.class),1,0,9);
    }

    @Override
    public List<Product> findByNew() throws SQLException {
        QueryRunner queryRunner = new QueryRunner(JDBCUtils.getDataSource());
        String sql = "select * from product where pflag=? order by pdate desc limit ?";
        return queryRunner.query(sql,new BeanListHandler<>(Product.class),0,9);
    }
}
```

- 步骤 4：修改 `/jsp/index.jsp` ，页面显示
  - 热门商品

```
<div class="container-fluid">
    <div class="col-md-12">
        <h2>热门商品  <img src="${pageContext.request.contextPath}/img/title2.jpg"/></h2>
    </div>
    <div class="col-md-2" style="...">
```

```
<c:forEach var="p" items="${hotList}">
    <div class="col-md-2" style="text-align:center;height:200px;padding:10px 0px;">
        <a href="product_info.htm">
            <img src="${pageContext.request.contextPath}/${p.pimage}" width="130" height="130" style="display: inline-block;">
        </a>
        <p><a href="#" style='color:#666'>${p.pname}</a></p>
        <p><font color="#E4393C" style="font-size:16px">&yen;${p.shop_price}</font></p>
    </div>
</c:forEach>
```

```
            <a href="product_info.htm">
                <img src="products/hao/middle01.jpg" width="516px" height="200px" style="...">
            </a>
        </div>
        <c:forEach var="p" items="${hotList}">
            <div class="col-md-2" style="...">
                <a href="product_info.htm">
                    <img src="${pageContext.request.contextPath}/${p.pimage}" width="130" height="130"
                </a>
                <p><a href="#" style='...'>${p.pname}</a></p>
                <p><font color="#E4393C" style="...">&yen;${p.shop_price}</font></p>
            </div>
        </c:forEach>
    </div>
```

- 最新商品

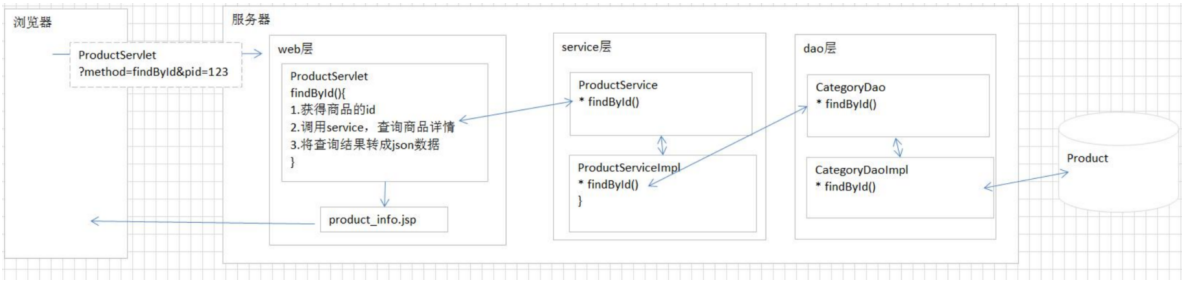
```
        <div class="container-fluid">
            <div class="col-md-12">
                <h2>最新商品  <img src="${pageContext.request.contextPath}/img/title2.jpg"/></h2>
            </div>
            <div class="col-md-2" style="...">
                <img src="products/hao/big01.jpg" width="205" height="404" style="..."/>
            </div>
            <div class="col-md-10">
                <div class="col-md-6" style="...">
                    <a href="product_info.htm">
                        <img src="products/hao/middle01.jpg" width="516px" height="200px" style="...">
                    </a>
                </div>

                <div class="col-md-2" style="...">
                    <a href="product_info.htm">
                        <img src="products/hao/small03.jpg" width="130" height="130" style="...">
                    </a>
```

```
<c:forEach var="p" items="${newList}">
    <div class="col-md-2" style="text-align:center;height:200px;padding:10px 0px;">
        <a href="#">
            <img src="${pageContext.request.contextPath}/${p.pimage}" width="130" height="130" style="display: inline-block;">
        </a>
        <p><a href="#" style='color:#666'>${p.pname}</a></p>
        <p><font color="#E4393C" style="font-size:16px">&yen;${p.shop_price}</font></p>
    </div>
</c:forEach>
```

```
<div class="col-md-10">
    <div class="col-md-6" style="...">
        <a href="product_info.htm">
            <img src="products/hao/middle01.jpg" width="516px" height="200px" style="...">
        </a>
    </div>

    <c:forEach var="p" items="${newList}">
        <div class="col-md-2" style="...">
            <a href="#">
                <img src="${pageContext.request.contextPath}/${p.pimage}" width="130" height="130" style="
            </a>
            <p><a href="#" style='...'>${p.pname}</a></p>
            <p><font color="#E4393C" style="...">&yen;${p.shop_price}</font></p>
        </div>
    </c:forEach>
```

2.3 商品详情

2.3.1 分析



2.3.2 代码实现

- 步骤 1：修改 `/jsp/index.jsp` ，点击图片或标题可以查询商品详情

```
84          <div class="col-md-2" style="text-align:center;height:200px;padding:10px 0px;">
85              <a href="${pageContext.request.contextPath}/productServlet?method=findById&pid=${p.pid}">
86                  <img src="${pageContext.request.contextPath}/${p.pimage}" width="130" height="130" style="...">
87              </a>
88              <p><a href="${pageContext.request.contextPath}/productServlet?method=findById&pid=${p.pid}" style='
89              <p><font color="#E4393C" style="...">&yen;${p.shop_price}</font></p>
```

- 步骤 2：修改 ProductServlet，添加 findById 方法

```
@WebServlet("/productServlet")
public class ProductServlet extends BaseServlet {
    // 通过id查询详情
    public String findById(HttpServletRequest request, HttpServletResponse response) throws Exception {
        // 1.接收参数
        String pid = request.getParameter("pid");

        // 2.调用service查询
        ProductService productService = new ProductServiceImpl();
        Product product = productService.findById(pid);

        // 3.1 将查询结果放到request域中
        request.setAttribute("product",product);

        // 3.2 页面跳转
        return "/jsp/product_info.jsp";
    }
}
```

- 步骤 3：修改 ProductService，添加 findById()方法

```
// 接口
    /**
     * 通过 id 查询详情
     * @param pid
     * @return
     */
    Product findById(String pid) throws SQLException;

//实现类
    @Override
    public Product findById(String pid) throws SQLException {
        return productdao.findById(pid);
    }
```

- 步骤 4：修改 ProductDao，添加 findById()方法

```
// 接口
```

```
    /**
     * 通过 id 查询详情
     * @param pid
     * @return
     */
    Product findById(String pid) throws SQLException;

// 实现类
    @Override
    public Product findById(String pid) throws SQLException {
        QueryRunner queryRunner = new
QueryRunner(JDBCUtils.getDataSource());
        String sql = "select * from product where pid = ?";
        return queryRunner.query(sql,new BeanHandler<>(Product.class),pid);
    }
```

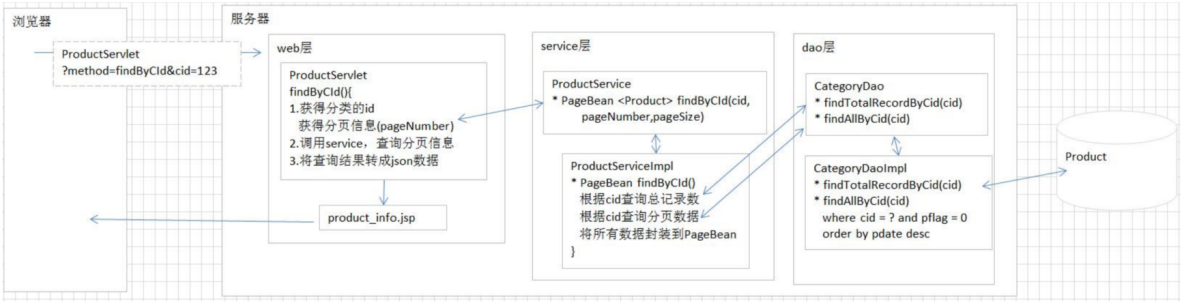
- 步骤 5：修改 `product_info.jsp` ，显示信息

```
product_info.jsp
<div style="margin:0 auto;width:950px;">
    <div class="col-md-6">
        <img style="opacity: 1;width:400px;height:350px;" title="" class="medium" src="${pageContext.request.contextPath}/${product.pimage}">
    </div>

    <div class="col-md-6">
        <div><strong>${ product.pname }</strong></div>
        <div style="border-bottom: 1px dotted #dddddd;width:350px;margin:10px 0 10px 0;">
            <div>编号：${ product.pid }</div>
        </div>

        <div style="margin:10px 0 10px 0;">亿家价: <strong style="color:#ef0101;">￥: ${ product.shop_price }</strong> 参 考 价： <del>￥${ product.market_price }</del>
        </div>

        <div style="margin:10px 0 10px 0;">促销: <a target="_blank" title="限时抢购(2014-07-30 ~ 2015-01-01)" style="background-color: #f07373;">限时抢购</a> </div>

        <div style="padding:10px;border:1px solid #e7dbb1;width:330px;margin:15px 0 10px 0;;background-color: #fffee6;">
            <div style="margin:5px 0 10px 0;">白色</div>

            <div style="border-bottom: 1px solid #faeac7;margin-top:20px;padding-left: 10px;">购买数量:
                <input id="quantity" name="quantity" value="1" maxlength="4" size="10" type="text"> </div>

            <div style="margin:20px 0 10px 0;;text-align: center;">
                <%--加入到购物车 --%>
                <a href="${pageContext.request.contextPath}/jsp/cart.jsp">
                    <input style="background: url('${pageContext.request.contextPath}/img/product.gif') no-repeat scroll 0 -600px rgba(0, 0, 0, 0);height:36px;width:1
                </a>  收藏商品</div>
        </div>
    </div>
</div>
<div class="clear"></div>
<div style="width:950px;margin:0 auto;">
    <div style="background-color:#d3d3d3;width:930px;padding:10px 10px;margin:10px 0 10px 0;">
        <strong>商品介绍</strong>
    </div>

    <div>
        ${ product.pdesc }
    </div>
```

```
    <div class="clear"></div>
    <div style="width:950px;margin:0 auto;">
        <div style="...">
            <strong>商品介绍</strong>
            <h3>${product.pdesc}</h3>
        </div>
    </div>

</div>
div>

iv style="...">
   <img src="${pageContext.request.contextPath}/img/footer.jpg" width="100%" height="78" alt="我们的优势" tit
```

2.4 查询分类商品(含分页)

2.4.1 分析



2.4.2 代码实现

- 步骤 1：修改 `/jsp/header.jsp` 显示分类列表的 js 代码

```
$("#menu").append("<li value='"+n.cid+"'><a
href='${pageContext.request.contextPath}/productServlet?
method=findByCid&cid="+n.cid+"'>"+n.cname+"</a></li>")
```

- 步骤 2：修改 ProductServlet，添加 findByCid 方法

```
    /**
     * 通过分类 id 查询所有商品
     *
     * @param request
     * @param response
     * @return
     * @throws Exception
     */
    public String findByCid(HttpServletRequest request, HttpServletResponse
response) throws Exception {
        // 1 接收参数
        // 1.1 分类 id
        String cid = request.getParameter("cid");
        // 1.2 当前页
        int pageNumber = 1;
        String pageNumberStr = request.getParameter("num");

        try {
            if(pageNumberStr != null) {
                //没有传递参数，或参数错误，使用默认值 1
                pageNumber = Integer.parseInt(pageNumberStr);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        // 1.3 每页显示个数
        int pageSize = 12; //固定值，可以通过请求参数获得
        // 2 调用业务层：
        ProductService productService = new ProductServiceImpl();
        PageBean<Product> pageBean =
productService.findByCid(cid,pageNumber,pageSize);

        // 3 将数据存放 request 作用域，并请求转发到指定的 jsp
        request.setAttribute("page", pageBean);
        return "/jsp/product_list.jsp";
    }
```

- 步骤 3：编写 PageBean 对象

```
package cn.aigoubuy.domain;

import java.util.List;

/**
 * @author azzhu
 * @create 2020-07-22 23:48:57
 */
public class PageBean<T> {
```

```java
    //基本属性
    private int currentPageNum;//当前页数，由用户指定                *
    private int pageSize = 5 ;//每页显示的条数，固定的                 *
    private int totalRecords;//总记录条数，数据库查出来的                    *
    private int totalPageNum;//总页数，计算出来的                       *
    private int startIndex;//每页开始记录的索引，计算出来的                      *
    private int prePageNum;//上一页                                  *
    private int nextPageNum;//下一页                                 *

    private List<T> list;//已经分好页的结果集,该list中只有10条记录

    //扩展属性
    //一共每页显示9个页码按钮
    private int startPage;//开始页码
    private int endPage;//结束页码

    //完善属性
    private String url;

    //要想使用我的分页，必须给我两个参数。一个是要看哪一页，另一个是总记录条数
    public PageBean(int currentPageNum,int totalRecords,int pageSize){
        this.currentPageNum = currentPageNum;
        this.totalRecords = totalRecords;
        this.pageSize=pageSize;

        //计算查询记录的开始索引
        startIndex = (currentPageNum-1)*pageSize;
        //计算总页数
        totalPageNum = totalRecords%pageSize==0?(totalRecords/pageSize):
(totalRecords/pageSize+1);


        startPage = currentPageNum - 4; //5
        endPage = currentPageNum + 4;  //13
        //看看总页数够不够9页
        if(totalPageNum>9){
            //超过了9页
            if(startPage < 1){
                startPage = 1;
                endPage = startPage+8;
            }
            if(endPage>totalPageNum){
                endPage = totalPageNum;
                startPage = endPage-8;
            }
        }else{
            //不够9页
            startPage = 1;
            endPage = totalPageNum;
        }
    }

    public String getUrl() {
        return url;
    }

    public void setUrl(String url) {
        this.url = url;
```

```
67        }
68
69        public int getStartPage() {
70            return startPage;
71        }
72
73        public void setStartPage(int startPage) {
74            this.startPage = startPage;
75        }
76
77        public int getEndPage() {
78            return endPage;
79        }
80
81        public void setEndPage(int endPage) {
82            this.endPage = endPage;
83        }
84
85        public int getPrePageNum() {
86            prePageNum = currentPageNum-1;
87            if(prePageNum<1){
88                prePageNum = 1;
89            }
90            return prePageNum;
91        }
92
93        public int getNextPageNum() {
94            nextPageNum = currentPageNum+1;
95            if(nextPageNum>totalPageNum){
96                nextPageNum = totalPageNum;
97            }
98            return nextPageNum;
99        }
100
101       public int getCurrentPageNum() {
102           return currentPageNum;
103       }
104
105
106       public void setCurrentPageNum(int currentPageNum) {
107           this.currentPageNum = currentPageNum;
108       }
109
110
111       public int getPageSize() {
112           return pageSize;
113       }
114
115
116       public void setPageSize(int pageSize) {
117           this.pageSize = pageSize;
118       }
119
120
121       public int getTotalRecords() {
122           return totalRecords;
123       }
124
```

```
125
126         public void setTotalRecords(int totalRecords) {
127             this.totalRecords = totalRecords;
128         }
129
130
131         public int getTotalPageNum() {
132             return totalPageNum;
133         }
134
135
136         public void setTotalPageNum(int totalPageNum) {
137             this.totalPageNum = totalPageNum;
138         }
139
140
141         public int getStartIndex() {
142             return startIndex;
143         }
144
145
146         public void setStartIndex(int startIndex) {
147             this.startIndex = startIndex;
148         }
149
150         public void setPrePageNum(int prePageNum) {
151             this.prePageNum = prePageNum;
152         }
153
154         public void setNextPageNum(int nextPageNum) {
155             this.nextPageNum = nextPageNum;
156         }
157
158         public List getList() {
159             return list;
160         }
161
162         public void setList(List list) {
163             this.list = list;
164         }
165
166         @Override
167         public String toString() {
168             return "PageBean{" +
169                     "currentPageNum=" + currentPageNum +
170                     ", pageSize=" + pageSize +
171                     ", totalRecords=" + totalRecords +
172                     ", totalPageNum=" + totalPageNum +
173                     ", startIndex=" + startIndex +
174                     ", prePageNum=" + prePageNum +
175                     ", nextPageNum=" + nextPageNum +
176                     ", list=" + list +
177                     ", startPage=" + startPage +
178                     ", endPage=" + endPage +
179                     ", url='" + url + '\'' +
180                     '}';
181         }
182     }
```

- 步骤 4：修改 ProductService，添加 findByCid()方法

```
// 接口
   /**
     * 分页查询所有商品
     * @param cid
     * @param pageNumber
     * @param pageSize
     * @return
     * @throws SQLException
     */
    PageBean<Product> findByCid(String cid, int pageNumber, int pageSize)
            throws SQLException;

//实现类
    @Override
    public PageBean<Product> findByCid(String cid, int pageNumber, int
pageSize) throws SQLException {
        // 1.获得总记录数
        int totalRecord = productdao.findTotalRecordByCid(cid);

        // 2.封装数据
        PageBean<Product> pageBean = new PageBean<>
(pageNumber,totalRecord,pageSize);
       //设置url
        pageBean.setUrl("productServlet?method=findByCid&cid="+cid);

        // 3.分页数据
        List<Product> data =
productdao.findAllByCid(cid,pageBean.getStartIndex(),pageBean.getPageSize())
;
        pageBean.setList(data);
        return pageBean;
    }
```

- 步骤 5：修改 ProductDao，提供 findTotalRecordByCid() 和 findAllByCid()

```
//接口
    /**
     *  查询总记录（含分页）
     * @param cid
     * @return
     * @throws SQLException
     */
    int findTotalRecordByCid(String cid) throws SQLException;

    /**
     * 查询指定分类的所有书籍（含分页）
     * @param cid
     * @param startIndex
     * @param pageSize
     * @return
     * @throws SQLException
     */
    List<Product> findAllByCid(String cid, int startIndex, int pageSize)
throws SQLException;

```

```
// 实现类
    @Override
    public int findTotalRecordByCid(String cid) throws SQLException {
        QueryRunner queryRunner = new
QueryRunner(JDBCUtils.getDataSource());
        String sql = "select count(*) from product where cid = ? and
pflag=?";
        Long count = (Long)queryRunner.query(sql,new ScalarHandler<>
(),cid,0);
        return count.intValue();
    }

    @Override
    public List<Product> findAllByCid(String cid, int startIndex, int
pageSize) throws SQLException {
        QueryRunner queryRunner = new
QueryRunner(JDBCUtils.getDataSource());
        String sql = "select * from product where cid = ? and pflag = ?
order by pdate desc limit ?,?";
        return queryRunner.query(sql,new BeanListHandler<>
(Product.class),cid,0,startIndex,pageSize);
    }
```

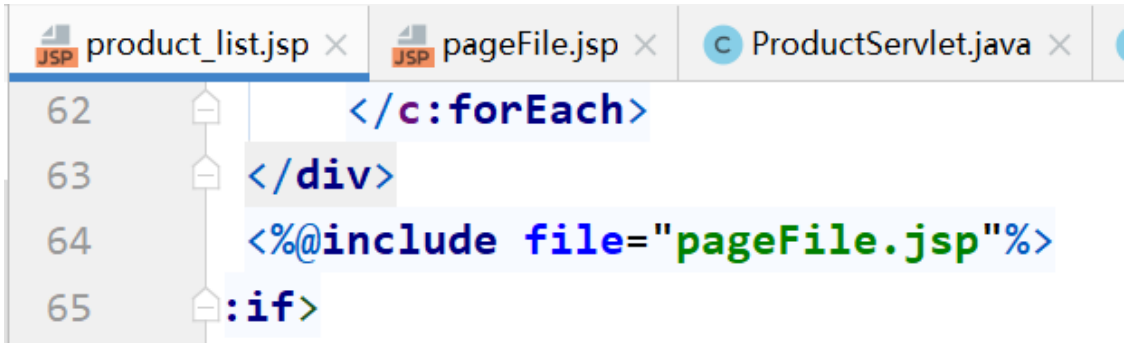
- 步骤 6：修改/jsp/product_list.jsp 页面，显示数据

```
    <c:forEach items="${page.list}" var="product">
        <div class="col-md-2">
            <a href="${pageContext.request.contextPath}/productServlet?method=findById&pi
                <img src="${pageContext.request.contextPath}/${product.pimage}" width="17
            </a>
            <p><a href="${pageContext.request.contextPath}/productServlet?method=findById
                <c:if test="${fn:length(product.pname) lt 27}">${product.pname}</c:if>
                <c:if test="${fn:length(product.pname) gt 27}">${fn:substring(product.pna
            </a></p>
            <p><font color="#FF0000">商城价：&yen;${product.shop_price}</font></p>
        </div>
    </c:forEach>
```

```
<c:if test="${empty page.list}">
    <div class="row" style="width:1210px;margin:0 auto;">
        <div class="col-md-12">
            <h1>暂无商品信息</h1>
        </div>
    </div>
</c:if>

<c:if test="${not empty page.list}">
    <div class="row" style="width:1210px;margin:0 auto;">
        <div class="col-md-12">
            <ol class="breadcrumb">
                <li><a href="#">首页</a></li>
            </ol>
        </div>


        <c:forEach items="${page.list}" var="product">
            <div class="col-md-2">
                <a href="${pageContext.request.contextPath}/productServlet?
method=findById&pid=${product.pid}">
```

```
21                      <img
    src="${pageContext.request.contextPath}/${product.pimage}" width="170"
    height="170" style="display: inline-block;">
22                  </a>
23                  <p><a
    href="${pageContext.request.contextPath}/productServlet?
    method=findById&pid=${product.pid}" style='color:green'>
24                      <c:if test="${fn:length(product.pname) lt
    27}">${product.pname}</c:if>
25                      <c:if test="${fn:length(product.pname) gt
    27}">${fn:substring(product.pname, 0, 27)}</c:if>
26                      </a></p>
27                  <p><font color="#FF0000">商城价：&yen;${product.shop_price}
    </font></p>
28              </div>
29          </c:forEach>
30
31      </div>
32  </c:if>
```

- 步骤 7：引入分页的jsp `pageFile.jsp`




```
1   <%@ page language="java" import="java.util.*" pageEncoding="UTF-8"%>
2   <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%>
3   <%--分页显示的开始 --%>
4   <div style="text-align:center">
5       共${page.totalPageNum}页/第${page.currentPageNum}页
6
7
8       <a href="${pageContext.request.contextPath}/${page.url}&num=1">首页</a>
9       <a
    href="${pageContext.request.contextPath}/${page.url}&num=${page.prePageNum}"
    >上一页</a>
10      <%--显示的页码，使用forEach遍历显示的页面 --%>
11      <c:forEach begin="${page.startPage}" end="${page.endPage}"
    var="pagenum">
12          <a
    href="${pageContext.request.contextPath}/${page.url}&num=${pagenum}">${pagen
    um}</a>
13      </c:forEach>
14
15      <a
    href="${pageContext.request.contextPath}/${page.url}&num=${page.nextPageNum}
    ">下一页</a>
16      <a
    href="${pageContext.request.contextPath}/${page.url}&num=${page.totalPageNum
    }">末页</a>
```

```
17        <input type="text" id="pagenum" name="pagenum" size="1"/><input type="button" value="前往" onclick="jump()" />
18        <script type="text/javascript">
19            function jump(){
20                var totalpage = ${page.totalPageNum};
21                var pagenum = document.getElementById("pagenum").value;
22                //判断输入的是一个数字
23                var reg =/^[1-9][0-9]{0,1}$/;
24                if(!reg.test(pagenum)){
25                    //不是一个有效数字
26                    alert("请输入符合规定的数字");
27                    return ;
28                }
29                //判断输入的数字不能大于总页数
30                if(parseInt(pagenum)>parseInt(totalpage)){
31                    //超过了总页数
32                    alert("不能大于总页数");
33                    return;
34                }
35                //转向分页显示的Servlet
36
 window.location.href="${pageContext.request.contextPath}/${page.url}&num="+pagenum;
37            }
38        </script>
39    </div>
40    <%--分页显示的结束--%>
```