WEB CRUD & MVC 思想

目标

- [ ] 理解并掌握 WEB CRUD 的组件交互图 `*****`
- [ ] 能独立画出 WEB CRUD 每个操作的组件交互图 `*****`
- [ ] 理解并掌握 WEB CRUD 的代码实现 `*****`
- [ ] 理解并掌握请求分发器的作用和设计 `*****`
- [ ] 理解 MVC 思想 `*****`

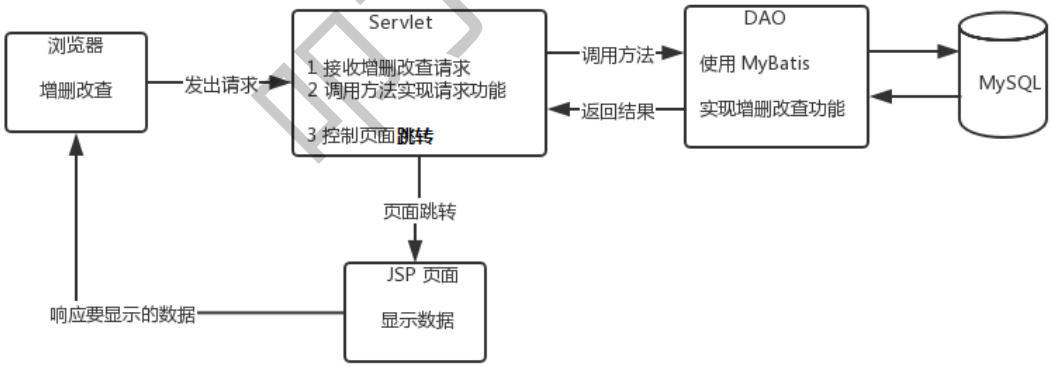
第一章 WEB CRUD 实现

需求：在浏览器上实现对 product 表的增删改查操作

技术使用

- 页面显示：JSP
- 接受用户请求：Servlet
- 和数据库交互：MyBatis

技术交互思路图：

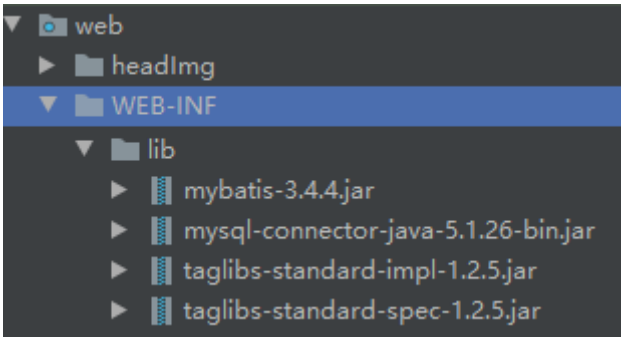


1.1 复习后台 CRUD

需求 : 完成商品表中数据的 CRUD(web)，让用户能够通过浏览器的相关操作,完成商品信息的增删改查

1.1.1 项目准备工作

1. 创建 web 项目，导入需要依赖的 jar 包,放入 web/WEB-INF/lib目录中.
2. mybatis,jdbc 驱动包,[log4j],[lombok]



3. 创建 product 表

栏位 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

名	类型	长度	小数点	不是 null	
id	bigint	11	0	☑	🔑1
productName	varchar	50	0	☐	
dir_id	bigint	11	0	☐	
salePrice	decimal	10	2	☐	
supplier	varchar	50	0	☐	
brand	varchar	50	0	☐	
cutoff	decimal	2	2	☐	
costPrice	decimal	10	2	☐	

4. 根据表创建 domain 包以及类

```
@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
@ToString
public class Product {
    private Long id;
    private String productName;
    private Long dir_id;
    private BigDecimal salePrice;
    private String supplier;
    private String brand;
    private Double cutoff;
    private BigDecimal costPrice;
}
```

5. 根据 domain 类创建 dao 包以及接口

cn.wolfcode.pmis.dao --> IProductDAO
cn.wolfcode.pmis.dao.impl --> ProductDAOImpl

6. 生成测试类（测试先行原则）
7. 完成一个功能，测试一个功能，并测试通过

1.1.2 MyBatis 配置文件实现

1. mybatis-config.xml 配置
2. ProductMapper.xml 创建和配置
3. MyBatisUtil 工具类抽取实现

1.1.3 后台 CRUD 实现

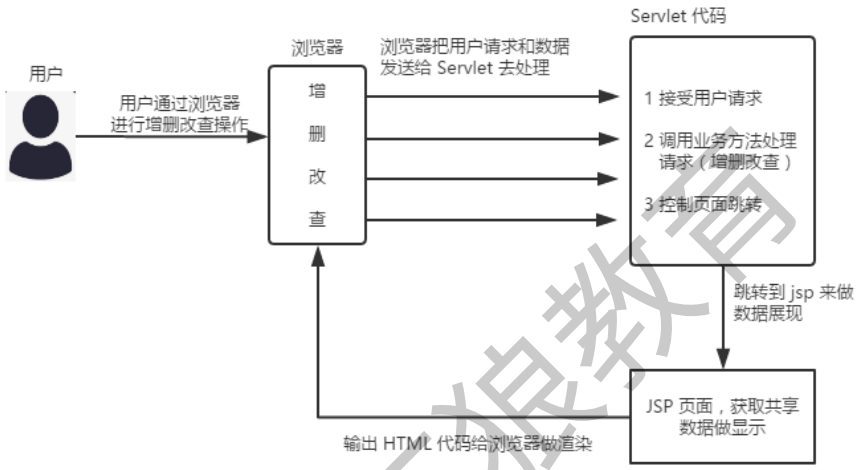
1. 列表查询实现
2. 添加功能实现
3. 修改功能实现
4. 删除功能实现

1.2 前台 CRUD 实现

前台 CRUD 指的是和用户打交道涉及到的相关代码组件书写。

JSP： 用户查看和操作的位置。

Servlet : 对用户使用浏览器发起的请求进行处理和给用户控制数据显示。



1.2.1 请求分发器设计和实现

用户所发起的增删改查请求，按目前的 Servlet 操作，只是在 service 中直接处理，如下

- 查询商品列表： 需要一个 Servlet 来处理，ProductListServlet
- 删除指定的商品：需要一个 Servlet 来处理，ProductDeleteServlet
- 点击编辑和添加都是进入到可编辑的界面 : 需要一个 Servlet 来处理，ProductInputServlet
- 在编辑界面,点击保存 : 需要一个 Servlet 来处理，ProductSaveOrUpdateServlet

以上设计问题:

1. 每个操作都需要一个 Servlet 来处理请求，那么100 张表的 CRUD ,需要 400 个 Servlet 来处理。
2. 类文件会爆炸式增长，不利后期类文件的归类管理和维护

解决方案：使用一个 Servlet 类来处理一张表的所有请求操作。

思考： 如何在一个Servlet 中来区分不同的操作（增删改查）

答案：增加一个参数，用参数值来区分

使用 cmd 参数来区分是哪种请求,不同的值,表示不同的请求。

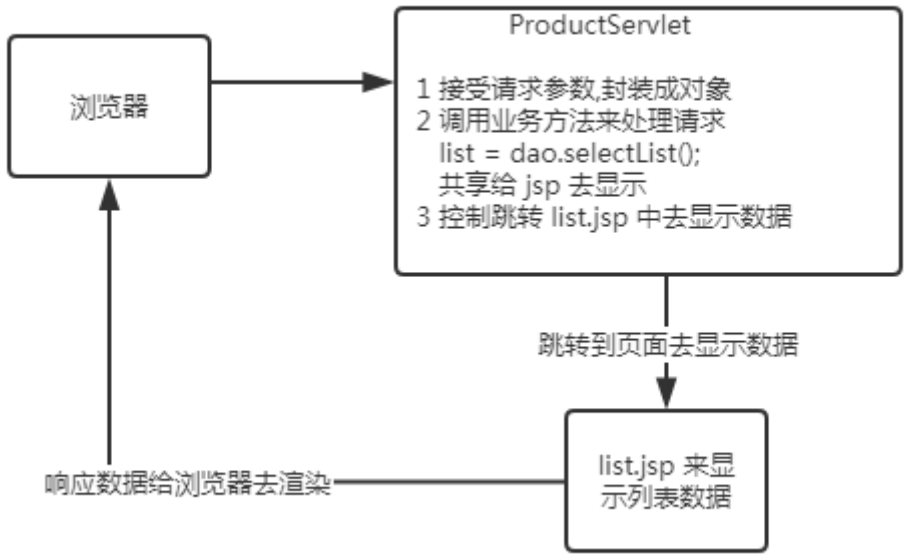
- cmd=delete : 删除操作
- cmd=input : 点击添加或者编辑进入到相同的输入界面
- cmd=saveOrUpdate : 在输入界面点击提交按钮的操作
- cmd=list 或者没写cmd：默认就是列表查询操作

代码实现:

```
protected void service(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        // 使用 cmd 参数值来区分不同的请求操作
        String cmd = req.getParameter("cmd");
        if("delete".equals(cmd)){
            delete(req,resp);
        }else if("input".equals(cmd)){
            input(req,resp);
        }else if("saveOrUpdate".equals(cmd)){
            saveOrUpdate(req,resp);
        }else{
            list(req,resp);
        }
    }

    // 查询列表操作
    protected void list(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        System.out.println("ProductServlet.list");
    }
    // 删除操作
    protected void delete(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        System.out.println("ProductServlet.delete");
    }
    // 跳转到录入数据页面
    protected void input(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
        System.out.println("ProductServlet.input");
    }
    // 保存或修改操作
    protected void saveOrUpdate(HttpServletRequest req, HttpServletResponse
resp) throws ServletException, IOException {
        System.out.println("ProductServlet.saveOrUpdate");
    }
```


1.2.2 查询列表实现

流程分析:



代码:

list 方法

```
// 1 接受请求参数封装成对象
// 2 调用业务方法去处理请求
List<Product> list = productDAO.selectList();
System.out.println(list);
// 把数据共享给 list.jsp 去显示
req.setAttribute("list",list);
// 控制跳转到 list.jsp 页面
req.getRequestDispatcher("/WEB-INF/views/product/list.jsp").forward(req,resp);
```

list.jsp

```
<a href="#">添加</a>
<table border="1" cellspacing="0" cellpadding="0" width="80%">
    <tr>
        <th>编号</th>
        <th>货品名</th>
        <th>分类编号</th>
        <th>零售价</th>
        <th>供应商</th>
        <th>品牌</th>
        <th>折扣</th>
        <th>进货价</th>
        <th>操作</th>
    </tr>
    <c:forEach var="product" items="${list}" varStatus="status">
        <tr>
            <td>${status.count}</td>
            <td>${product.productName}</td>
            <td>${product.dir_id}</td>
            <td>${product.salePrice}</td>
            <td>${product.supplier}</td>
            <td>${product.brand}</td>
            <td>${product.cutoff}</td>
```

```
            <td>${product.costPrice}</td>
            <td>
                <a href="#">删除</a>
                <a href="#">编辑</a>
            </td>
        </tr>
    </c:forEach>
</table>
```

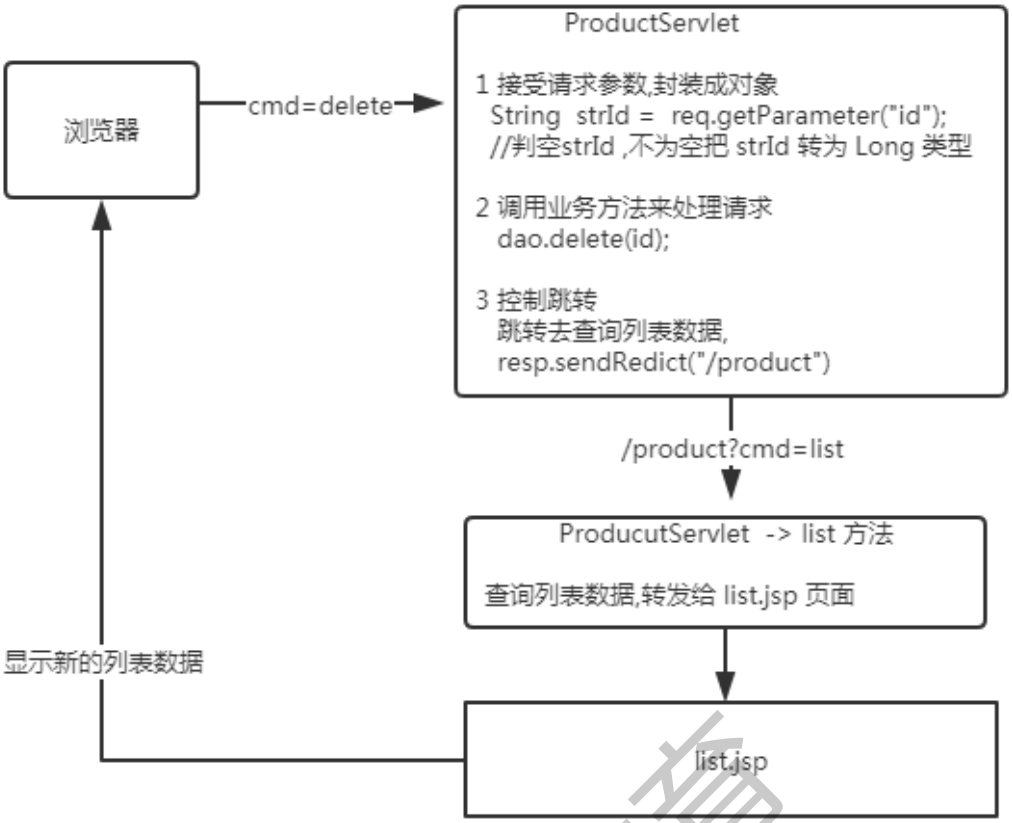
JS 鼠标悬停效果： 鼠标移入某行数据，行背景变灰色，移出恢复原样。

```
<script type="text/javascript">
    window.onload = function () {
    var trClzs = document.getElementsByClassName("trClassName");
    for(var i=0; i < trClzs.length ;i++){
        trClzs[i].onmouseover = function () {
            this.style.backgroundColor = "gray";
        }
        trClzs[i].onmouseout = function () {
            this.style.backgroundColor = "";
        }
    }
}
</script>
```

```
<!-- 给 tr 加 class 可通过 js 获取元素对象再绑定事件 -->
<tr class="trClassName">
    <td>${status.count}</td>
    <td>${product.productName}</td>
    <td>${product.dir_id}</td>
    <td>${product.salePrice}</td>
    <td>${product.supplier}</td>
    <td>${product.brand}</td>
    <td>${product.cutoff}</td>
    <td>${product.costPrice}</td>
    <td>
        <a href="#">删除</a>
        <a href="#">编辑</a>
    </td>
</tr>
```


1.2.3 删除实现

流程分析:



代码:

表格代码

```
<a href="/product?cmd=delete&id=${product.id}">删除</a>
```

delete方法

```
protected void delete(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
        // 1 接受请求参数
        String strId = req.getParameter("id");
        // 判空
        if(StringUtil.hasLength(strId)){
            Long id = Long.valueOf(strId);
            // 2 调用业务方法来处理请求
            productDAO.delete(id);
        }
        // 3 控制跳转
        resp.sendRedirect("/product");
    }
```

StringUtil

```
// 判断字符串是否有长度
public static boolean hasLength(String val){
    return val != null && !"".equals(val.trim());
}
```

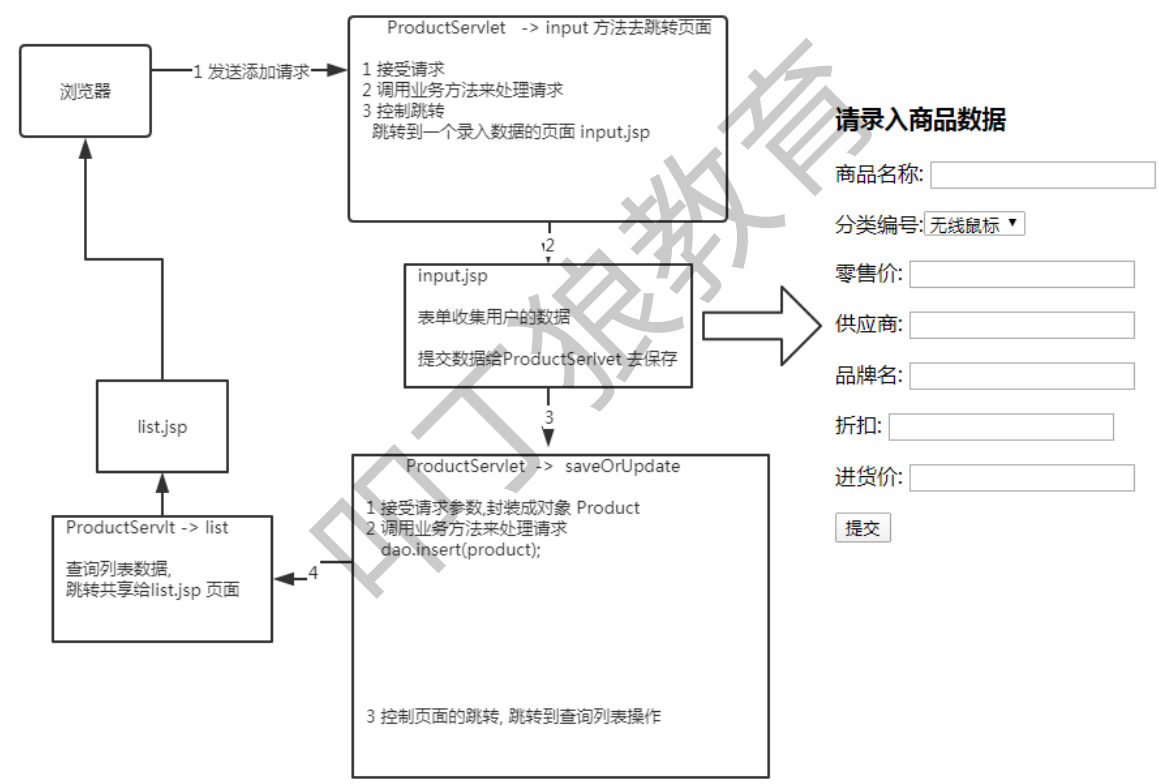
JS 删除确认:

```
<a href="#" onclick="deleteTr(${product.id})">删除</a>
```

```
function deleteTr(id) {
    // 弹出确认框
    var flag = window.confirm("亲,确定删除吗?");
    if(flag){
        // 使用js 去发送请求
        window.location.href="/product?cmd=delete&id=" + id;
    }
}
```

1.2.4 保存实现

流程分析:



添加按钮:

```
<a href="/product?cmd=input">添加</a>
```

input 方法:

```
protected void input(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
    // 控制跳转
    req.getRequestDispatcher("/WEB-
INF/views/product/input.jsp").forward(req,resp);
}
```

编辑页面 input.jsp：

```
h3>请录入商品数据</h3>
<form action="/product?cmd=saveOrUpdate" method="post">
    <p>商品名称: <input type="text" name="productName"></p>
    <p>分类编号:<select name="dirId">
        <option value="2">无线鼠标</option>
        <option value="3">有线鼠标</option>
        <option value="4">游戏鼠标</option>
        </select>
    </p>
    <p>零售价: <input type="number" name="salePrice"></p>
    <p>供应商: <input type="text" name="supplier"></p>
    <p>品牌名: <input type="text" name="brand"></p>
    <p>折扣: <input type="number" name="cutoff" step="0.1"></p>
    <p>进货价: <input type="number" name="costPrice"></p>
    <input type="submit" value="提交">
</form>
```

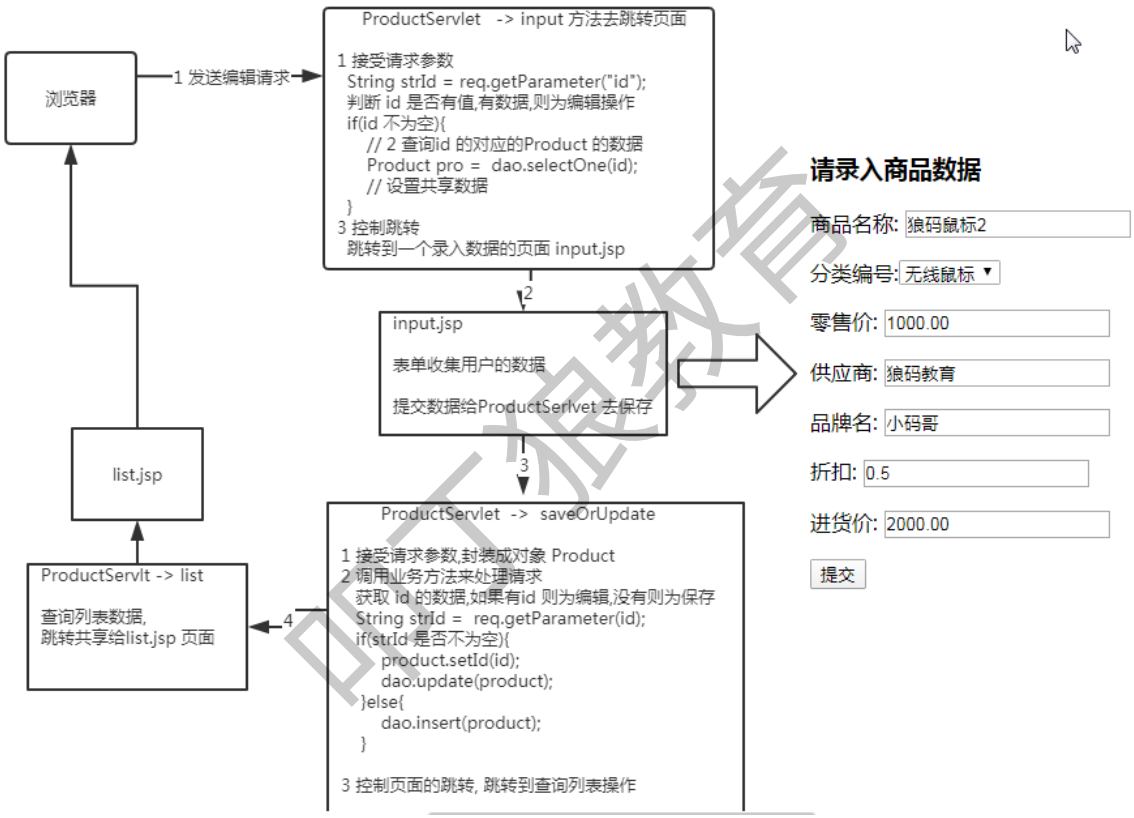
saveOrUpdate 方法

```
protected void saveOrUpdate(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
    // 1 接受请求参数,封装成对象
    Product product = new Product();
    // 获取请求中的参数,封装到product对象中
    req2roduct(req,product);
    System.out.println(product);
    // 2 调用业务方法来处理请求
    productDAO.insert(product);
    // 3 控制页面跳转
    resp.sendRedirect("/product");
}

// 获取请求中的参数,封装到product对象中
private void req2roduct(HttpServletRequest req, Product product) {
    String productName = req.getParameter("productName");
    product.setProductName(productName);

    String strDirId = req.getParameter("dirId");
    if(StringUtil.hasLength(strDirId)){
        product.setDir_id(Long.valueOf(strDirId));
    }
    String strSalePrice = req.getParameter("salePrice");
    if(StringUtil.hasLength(strSalePrice)){
        product.setSalePrice(new BigDecimal(strSalePrice));
    }

    String supplier = req.getParameter("supplier");
    product.setSupplier(supplier);
    String brand = req.getParameter("brand");
    product.setBrand(brand);
```

```
        String strCutoff = req.getParameter("cutoff");
        if(StringUtil.hasLength(strCutoff)){
            product.setCutoff(Double.valueOf(strCutoff));
        }

        String strCostPrice = req.getParameter("costPrice");
        if(StringUtil.hasLength(strCostPrice)){
            product.setCostPrice(new BigDecimal(strCostPrice));
        }
    }
```

1.2.5 编辑实现

流程分析:



代码实现:

编辑按钮

```
<a href="/product?cmd=input&id=${product.id}">编辑</a>
```

input方法;

```
protected void input(HttpServletRequest req, HttpServletResponse resp) throws
ServletException, IOException {
    String strId = req.getParameter("id");
    // 判断id是否有数据,有则为编辑,没有则为保存
    if(StringUtil.hasLength(strId)){
        // 获取id对象的Product 对象的数据
        Product product = productDAO.selectOne(Long.valueOf(strId));
        System.out.println(product);
        // 共享数据给 input.jsp 做回显
        req.setAttribute("product",product);
    }
    // 控制跳转
    req.getRequestDispatcher("/WEB-
INF/views/product/input.jsp").forward(req,resp);
}
```

编辑界面:

```
<h3>请录入商品数据</h3>
<form action="/product?cmd=saveOrUpdate" method="post">
     <%-- id 使用来判断是添加还是编辑的操作,所以必须带上--%>
     <input type="hidden" name="id" value="${product.id}">
     <%--value 是用来做回显的--%>
     <p>商品名称: <input type="text" name="productName"
value="${product.productName}"></p>
     <p>分类编号:
        <select name="dirId">
           <option value="2" ${product.dir_id == 2 ? 'selected' : ''}>无线鼠标
</option>
           <option value="3" ${product.dir_id == 3 ? 'selected' : ''}>有线鼠标
</option>
           <option value="4" ${product.dir_id == 4 ? 'selected' : ''}>游戏鼠标
</option>
        </select>
     </p>
     <p>零售价: <input type="number" name="salePrice"
value="${product.salePrice}"></p>
     <p>供应商: <input type="text" name="supplier" value="${product.supplier}">
</p>
     <p>品牌名: <input type="text" name="brand" value="${product.brand}"></p>
     <p>折扣: <input type="number" name="cutoff" step="0.1"
value="${product.cutoff}"></p>
     <p>进货价: <input type="number" name="costPrice"
value="${product.costPrice}"></p>
     <input type="submit" value="提交">
</form>
```

saveOrUpdate方法:

```
protected void saveOrUpdate(HttpServletRequest req, HttpServletResponse resp)
throws ServletException, IOException {
    // 1 接受请求参数,封装成对象
```

```
        Product product = new Product();
        // 获取请求中的参数,封装到product对象中
        req2roduct(req,product);
        System.out.println(product);
        // 获取id的数据,判断是否是编辑还是保存
        String strId = req.getParameter("id");
        if(StringUtil.hasLength(strId)){
            // 给product 设置要修改的数据的id
            product.setId(Long.valueOf(strId));
            productDAO.update(product);
        }else {
            // 2 调用业务方法来处理请求
            productDAO.insert(product);
        }
        // 3 控制页面跳转
        resp.sendRedirect("/product");
    }
```

1.3 如何找 bug

1. 看错误信息描述，定位错误位置
2. 如果没有错误信息，只是数据不对
    1. 打开浏览器 F12 查看请求数据是否正确
    2. 在 servlet 的 service方法中打断点，查看数据的变化，判断是封装问题还是业务方法问题
3. 定位到错误思考原因，解决问题
4. 收集异常，写明原因和解决方案

常见的错误:

1. NumberFormatException ：没有给字符判断是否为空,直接转为 number 类型
2. 做编辑操作变成了保存操作：
    1. input.jsp 没有带id,
    2. 要么 saveOrUpdate 方法没有根据id来做业务处理
    3. 字符串判空有问题
3. PropertyNotFoundException: 类型[cn.wolfcode.pmis.domain.Product]上找不到属性[dirId]
    1. 获取的属性和类型中的属性名不一致

注意事项: 以后不需要数据共享的跳转统统使用重定向即可

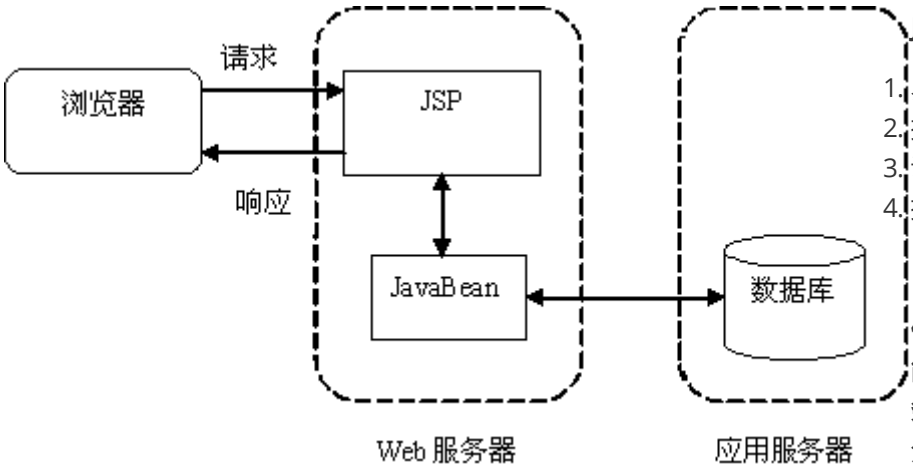
第二章 MVC 思想

软件(web应用)开发的模式：Model1（模型一），Model2(模型二)，MVC

2.1 Model1 模型一

以 JSP 为中心

核心技术： JSP + JavaBean



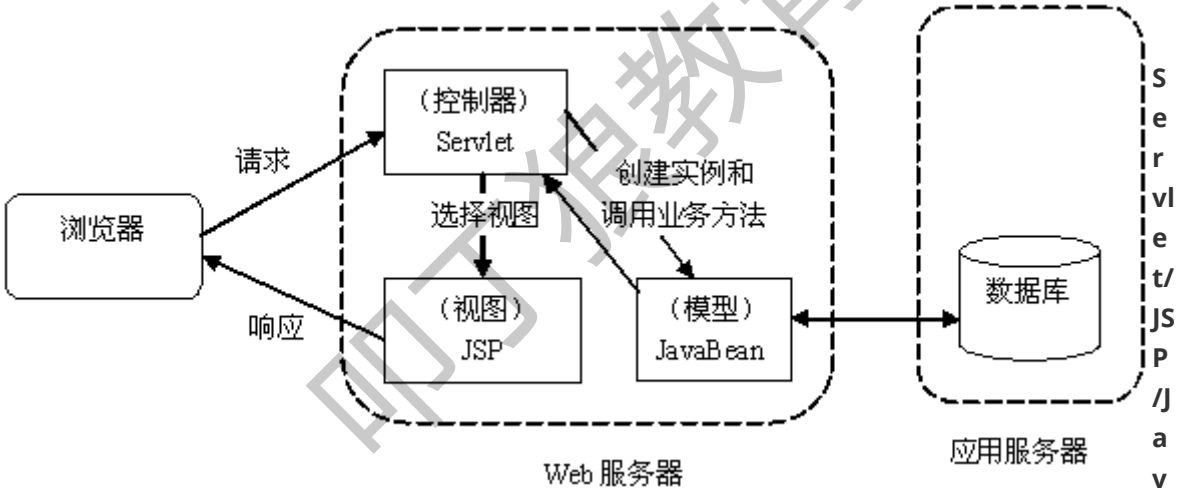
JSP的职责:

1.界面输出（页面渲染）.
2.接受请求参数
3.调用业务方法,处理请求
4.控制界面跳转.

优势: 适用于简单的功能,快速开发.
劣势: 没有体现出责任分离原则.

2.2 Model2 模型二

解决了Model1 存在的责任不分离的问题，以 Servlet 为中心 (所有请求都发送给 Servlet)
核心技术： JSP + Servlet + JavaBean



aBean 的职责:

- JSP： 界面输出(页面渲染)
- Servlet
  - 1:接受请求参数,封装成对象
  - 2:调用业务方法,处理请求
  - 3:控制跳转.
- JavaBean : 体现封装，封装数据，封装业务操作API,可重复使用.

优势: 体现出责任分离原则，提高代码可读性和维护度
劣势: 实现相对 Model1 复杂一点.

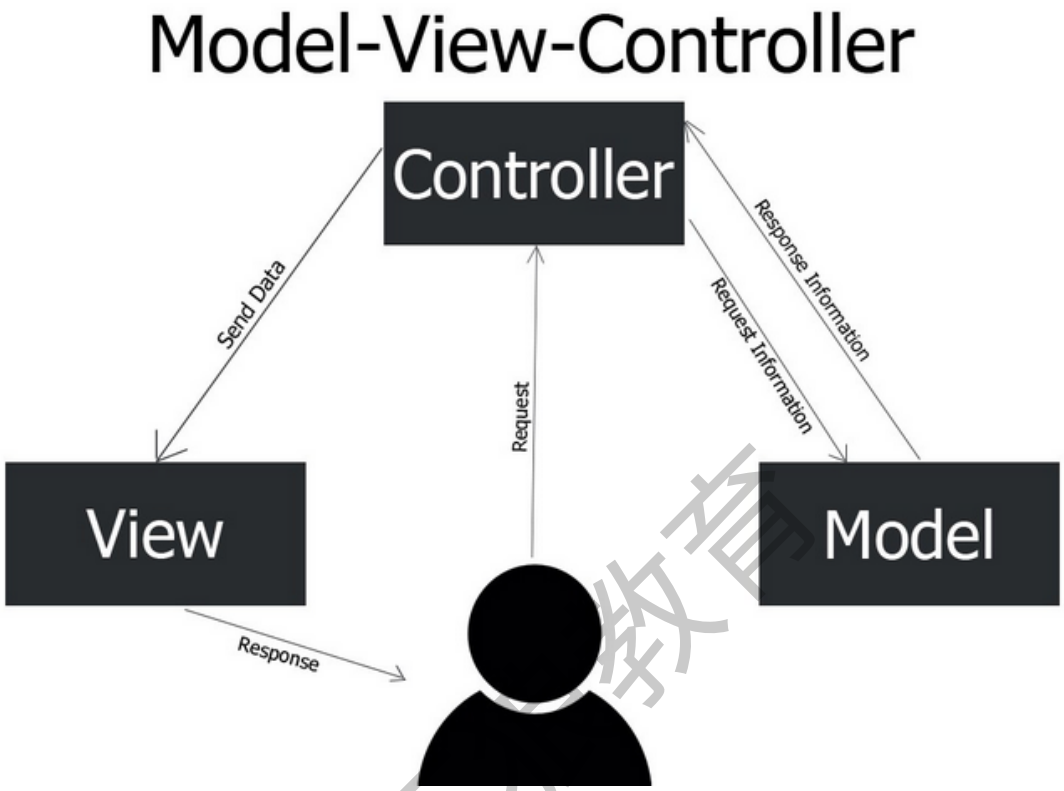
2.3 MVC 设计思想（面试题）

目的: 责任分离,把业务代码从视图中剥离出来，早期运用于 CS 领域(桌面程序).

核心掌握点:

M : Model，模型对象 (封装业务操作,算法,可重复使用,JavaBean).DAO,Domain
V： View，视图 (界面) JSP,HTML
C： Controller，控制器 (接受请求，控制跳转.) Servlet

MVC 实现图:



结论:

1. Model2 就是一个小型的 MVC 架构
2. 目前咱们就是使用 MVC 架构 JSP + Servlet + Model（Domain,DAO）
3. 跟着老师走，潜移默化中会学到很多好的设计思想，代码规范

小结

1. 理解理清 WEB CRUD 的执行流程(画图)
2. 掌握 WEB CRUD 的代码实现 (完成一个模型的 WEB CRUD : 直接拷贝后台 crud 代码,重点练习前台CRUD 实现.
3. 完成 product(其他表) 表的 WEB CRUD (从零开始写,复习后台的crud,巩固前台的crud操作)
4. 鼠标悬停效果,删除确认效果(拓展)
5. 理解 MVC 思想 (M: V: C:)
6. 写总结,收集异常
7. 今晚把 前台crud 练习代码提交
8. 明天晚上20:00 之前提交当天的练习