day28-综合练习

今日内容

- [ ] 能够完成显示所有联系人案例
- [ ] 能够完成添加联系人案例
- [ ] 能够完成删除联系人案例
- [ ] 能够理解PageBean分页数据的封装
- [ ] 掌握limit关键字的使用
- [ ] 能够完成分页展示联系人案例

案例一-显示所有联系人案例

一，案例需求

- 查询数据库里面所有的用户, 展示在页面


显示所有联系人

编号	姓名	性别	年龄	籍贯	QQ	邮箱	操作
1	张三	男	11	广州	766335435	766335435@qq.com	修改 删除
2	李四	男	12	上海	243424242	243424242@qq.com	修改 删除
3	王五	女	13	广州	474574574	474574574@qq.com	修改 删除
4	赵六	男	14	北京	987069697	987069697@qq.com	修改 删除
5	钱七	女	15	广州	412132145	412132145@qq.com	修改 删除

添加联系人

二,技术分析

传统方式的开发一个请求对应一个Servlet:这样的话会导致一个模块的Servlet过多,导致整个项目的Servlet都会很多. 能不能做一个处理?让一个**模块**用一个Servlet处理请求. 当前是**联系人模块**, 就创建一个==LinkManServlet==

- 传统方式

  查询所有的联系人 :http://localhost:8080/day28/findAll

  添加联系人: :http://localhost:8080/day28/add

  删除联系人 :http://localhost:8080/day28/delete

  ....

- 以模块为单位创建Servlet分析

  查询所有的联系人 :http://localhost:8080/day28/linkManServlet?method=findAll

  添加联系人: :http://localhost:8080/day28/linkManServlet?method=add

  删除联系人 :http://localhost:8080/day28/linkManServlet?method=delete

  ....

- 以模块为单位

```
package com.itheima.web;

import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;
import java.io.IOException;

/**
 * @Author: pengzhilin
 * @Date: 2021/5/10 8:46
 */
@WebServlet("/LinkManServlet")
public class LinkManServlet extends HttpServlet {
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // 1.处理乱码
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html;charset=utf-8");

        // 2.获得method请求参数
        String method = request.getParameter("method");

        // 3.判断method请求参数的值,做出对应的处理
        if ("findAll".equals(method)) {
            // 调用findAll方法
            findAll(request, response);

        } else if ("add".equals(method)) {
            // 调用add方法
            add(request, response);

        } else if ("delete".equals(method)) {
            // 调用delete方法
            delete(request, response);

        } else if ("findPage".equals(method)) {
            // 调用findPage方法
            findPage(request, response);

        } else if ("update".equals(method)) {
            // 调用update方法
            update(request, response);
        }
    }

    // 修改联系人
    private void update(HttpServletRequest request, HttpServletResponse
response) {

    }

    // 分页查询
```

```
    private void findPage(HttpServletRequest request, HttpServletResponse
response) {

    }

    // 删除联系人
    private void delete(HttpServletRequest request, HttpServletResponse
response) {

    }

    // 添加联系人
    private void add(HttpServletRequest request, HttpServletResponse response) {

    }

    // 查询所有联系人
    private void findAll(HttpServletRequest request, HttpServletResponse
response) {

    }

    protected void doGet(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        doPost(request, response);
    }
}
```

三, 思路分析

四, 代码实现

1.准备工作

- 数据库的创建

```
CREATE TABLE linkman (
  id int primary key auto_increment,
  name varchar(50),
  sex varchar(50),
  age int,
  address varchar(50),
```

```
  qq varchar(50),
  email varchar(50)
);

INSERT INTO `linkman`  (`id`, `name`, `sex`, `age`, `address`, `qq`, `email`)
VALUES
(null, '张三', '男', 11, '广东', '766335435', '766335435@qq.com'),
(null, '李四', '男', 12, '广东', '243424242', '243424242@qq.com'),
(null, '王五', '女', 13, '广东', '474574574', '474574574@qq.com'),
(null, '赵六', '女', 18, '广东', '77777777', '77777777@qq.com'),
(null, '钱七', '女', 15, '湖南', '412132145', '412132145@qq.com'),
(null, '王八', '男', 25, '广西', '412132775', '412132995@qq.com');
```

- JavaBean的创建

```
public class LinkMan implements Serializable{
    /**
     *   id int primary key auto_increment,
     name varchar(50),
     sex varchar(50),
     age int,
     address varchar(50),
     qq varchar(50),
     email varchar(50)
     */
    private int id;
    private String name;
    private String sex;
    private int age;
    private String address;
    private String qq;
    private String email;
}
```

- jar包
- 工具类
- 配置文件
- 页面

2.代码

- index.jsp页面

```
<%@ page contentType="text/html;charset=UTF-8" language="java" %>
<html>
  <head>
    <title>$Title$</title>
  </head>
  <body>
    <a href="linkMan?method=findAll">查询所有的联系人</a>
  </body>
</html>
```

- LinkManServlet

```
// 查询所有联系人
    private void findAll(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        try {
            //1. 调用service层,获得所有联系人List<LinkMan>
            List<LinkMan> list = service.findAll();

            //2. 把List<LinkMan>集合存储到域对象(request)
            request.setAttribute("list", list);

            //3. 响应页面
            if (list != null) {
                //成功: 转发到list.jsp

 request.getRequestDispatcher("list.jsp").forward(request,response);
            }else {
                //失败: 显示服务器异常
                response.getWriter().println("<h1>服务器异常</h1");
            }
        } catch (Exception e) {
            e.printStackTrace();
            response.getWriter().println("<h1>服务器异常</h1");
        }
    }
```

- LinkManService

```
public class LinkManService {
    // 创建LinkManDao对象
    private LinkManDao dao = new LinkManDao();

    // 查询所有联系人
    public List<LinkMan> findAll() {
        try {
            // 1.调用dao层,获得所有联系人信息
            List<LinkMan> list = dao.findAll();

            // 2.返回所有联系人信息
            return list;
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }
}
```

- LinkManDao

```

/**
 * @Author: pengzhilin
 * @Date: 2021/5/10 8:59
 */
public class LinkManDao {
    // 查询所有联系人
```

```
    public List<LinkMan> findAll() throws SQLException {
        QueryRunner qr = new QueryRunner(C3P0Utils.getDataSource());
        String sql = "select * from linkman";
        List<LinkMan> list = qr.query(sql, new BeanListHandler<LinkMan>
(LinkMan.class));
        return list;
    }
}
```

- list.jsp

```
 <%--每遍历一次 就是一个联系人 就对应一行; items:从域对象里面取出list; var:每遍历一次的赋值
变量--%>
<c:forEach items="${list}" var="lm">
            <tr>
                <td>${lm.id}</td>
                <td>${lm.name}</td>
                <td>${lm.sex}</td>
                <td>${lm.age}</td>
                <td>${lm.address}</td>
                <td>${lm.qq}</td>
                <td>${lm.email}</td>
                <td><a class="btn btn-default btn-sm" href="修改联系人.html">修改
</a> <a class="btn btn-default btn-sm" href="修改联系人.html">删除</a></td>
            </tr>
        </c:forEach>
```

五,小结

1. 浏览器 请求LinkManServlet, 携带method=findAll
2. 在LinkManServlet里面创建findAll()方法

```
//1.调用业务 获得所有的联系人List<LinkMan> list
//2.把list存到request里面, 转发list.jsp
```

3. 创建LinkManService

```
public List<LinkMan> findAll(){
    //1.调用Dao 获得List
    //2.返回List

}
```

4. 创建LinkManDao

```
public List<LinkMan> findAll(){
    //1.使用DBUtils查询所有的联系人(BeanListHandler)

}
```

案例二:添加联系人

一,案例需求

1. 点击添加联系人跳转添加联系人页面

显示所有联系人

编号	姓名	性别	年龄	籍贯	QQ	邮箱	操作
1	张三	男	20	广东	44444222	zs@qq.com	修改 删除
2	张三	男	20	广东	44444222	zs@qq.com	修改 删除
3	张三	男	20	广东	44444222	zs@qq.com	修改 删除
4	张三	男	20	广东	44444222	zs@qq.com	修改 删除
5	张三	男	20	广东	44444222	zs@qq.com	修改 删除

添加联系人

2. 在添加联系人页面，点击提交按钮,把数据提交到服务器,保存到数据库

添加联系人页面

姓名：

请输入姓名

性别：男 女

年龄：

请输入年龄

籍贯：

广东

QQ：

请输入QQ号码

Email：

请输入邮箱地址

提交 重置 返回

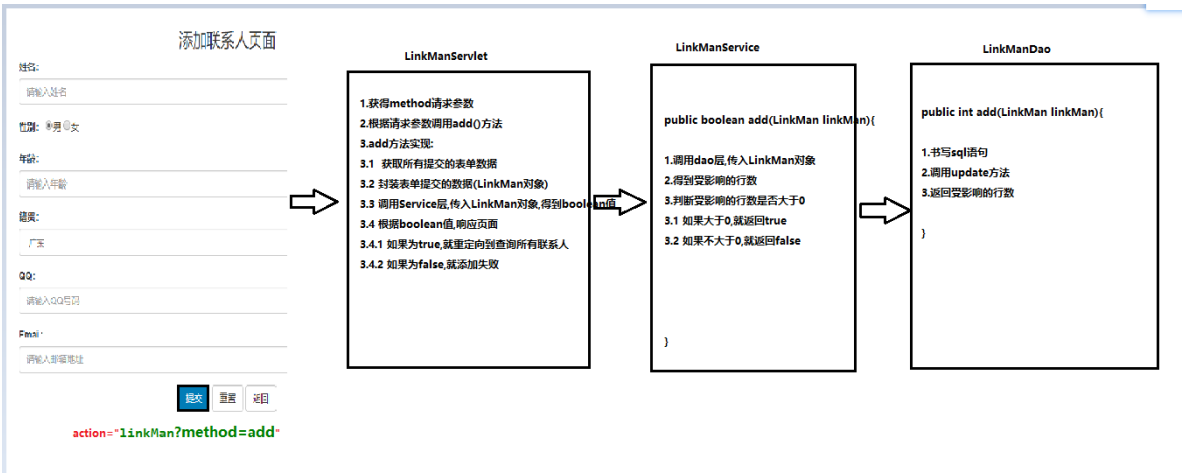
3. 在添加完成，可以查看到新建的联系人信息

显示所有联系人

编号	姓名	性别	年龄	籍贯	QQ	邮箱	操作
1	张三	男	20	广东	44444222	zs@qq.com	修改 删除
2	张三	男	20	广东	44444222	zs@qq.com	修改 删除
3	张三	男	20	广东	44444222	zs@qq.com	修改 删除
4	张三	男	20	广东	44444222	zs@qq.com	修改 删除
5	张三	男	20	广东	44444222	zs@qq.com	修改 删除

添加联系人

二,思路分析



三代码实现

- add.jsp 页面

```
1.设置form的action属性
<form action="linkMan?method=add" method="post">
```

- LinkManServlet

```
// 添加联系人
    private void add(HttpServletRequest request, HttpServletResponse response)
throws IOException {
        try {
            //1 获取所有提交的表单数据
            Map<String, String[]> map = request.getParameterMap();

            //2 封装表单提交的数据(LinkMan对象)
            LinkMan linkMan = new LinkMan();
            BeanUtils.populate(linkMan,map);

            //3 调用Service层,传入LinkMan对象,得到boolean值
            boolean flag = service.add(linkMan);

            //4 根据boolean值,响应页面
            if (flag) {
                //4.1 如果为true,就重定向到查询所有联系人
                response.sendRedirect("linkMan?method=findAll");

            }else {
                //4.2 如果为false,就添加失败
                response.getWriter().println("<h1>服务器异常,添加联系人失败</h1");
            }
        } catch (Exception e) {
            e.printStackTrace();
            response.getWriter().println("<h1>服务器异常</h1");
        }
    }
```

- LinkManService

```
public boolean add(LinkMan linkMan){
        try {
            // 1.调用dao层,传入LinkMan对象,得到受影响的行数
            int rows = dao.add(linkMan);

            // 2.判断受影响的行数是否大于0
            if (rows > 0) {
                // 2.1 如果大于0,就返回true
                return true;
            }else {
                // 2.2 如果不大于0,就返回false
                return false;
            }
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
```

- LinkManDao

```
public class LinkManDao {
    QueryRunner qr = new QueryRunner(C3P0Utils.getDataSource());

    // 查询所有联系人
    public List<LinkMan> findAll() throws SQLException {
        String sql = "select * from linkman";
        List<LinkMan> list = qr.query(sql, new BeanListHandler<LinkMan>
(LinkMan.class));
        return list;
    }

    // 增加联系人
    public int add(LinkMan linkMan) throws SQLException {
        // 1.书写sql语句
        String sql = "insert into linkman values(null,?,?,?,?,?,?)";
        Object[] args = {
                linkMan.getName(),
                linkMan.getSex(),
                linkMan.getAge(),
                linkMan.getAddress(),
                linkMan.getQq(),
                linkMan.getEmail()
        };
        // 2.调用update方法
        int rows = qr.update(sql, args);

        // 3.返回受影响的行数
        return rows;
    }

}
```

四,小结

1. 新增联系人说白了就是向数据库里面插入一条记录

2. 思路

   - 浏览器 点击提交 把数据提交到LinkManServlet(method=add, 联系人的基本信息)
   - 在LinkManServlet创建add()方法

```
//1.获得请求参数(联系人的基本信息)，封装成LinkMan对象
//2.调用业务新增联系人
//3.再查询所有展示【建议使用重定向】
```

   - 在LinkManService创建add()方法

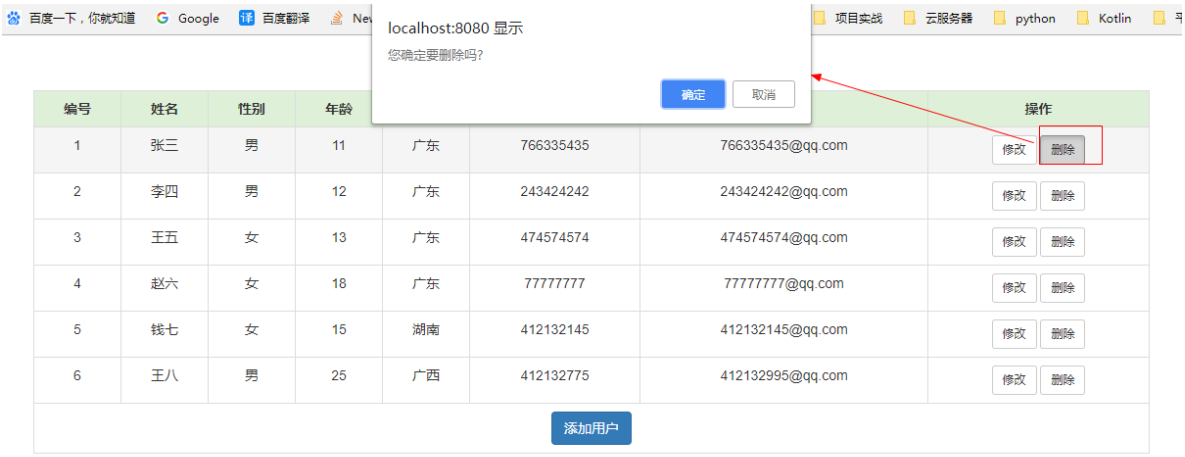
```
public boolean add(LinkMan linkMan){
    //1.调用Dao 保存
}
```

   - 在LinkManDao创建save()方法

```
public int add(LinkMan linkMan){
    //1.使用DBUtils保存
}
```

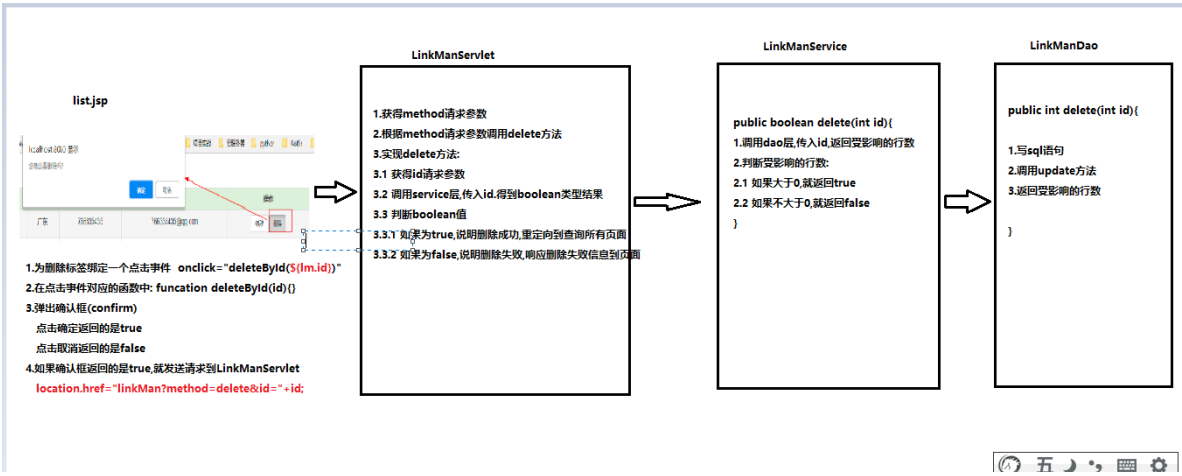
案例三:删除联系人

一,案例需求



点击确定删除之后, 再重新查询所有全部展示,

二,思路分析



三,代码实现

- list.jsp 前端页面

```
1.为删除按钮添加点击事件
<a class="btn btn-default btn-sm" onclick="deleteById(${lm.id})">删除</a>
2.实现点击事件函数
<script>
    function deleteById(id) {
        //1.为删除标签绑定一个点击事件
        //2.在点击事件对应的函数中:
        //3.弹出确认框(confirm)
        var flag = confirm("确定要删除id为" + id + "的信息吗?");
        // 点击确定返回的是true
        // 点击取消返回的是false
        // 4.如果确认框返回的是true,就发送请求到LinkManServlet
        if (flag) {
            location.href = "linkMan?method=delete&id=" + id;
        }
    }
</script>
```

- LinkManServlet

```
  // 删除联系人
    private void delete(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        try {
            //1 获得id请求参数
            int id = Integer.parseInt(request.getParameter("id"));

            //2 调用service层,传入id.得到boolean类型结果
            boolean flag = service.delete(id);

            //3 判断boolean值
            if (flag) {
                //3.1 如果为true,说明删除成功,重定向到查询所有页面
                response.sendRedirect("linkMan?method=findAll");
            }else {
                //3.2 如果为false,说明删除失败,响应删除失败信息到页面
                response.getWriter().println("<h1>服务器异常,删除联系人失败</h1>");
```

```
            }
        } catch (Exception e) {
            e.printStackTrace();
            response.getWriter().println("<h1>服务器异常</h1>");
        }
    }
```

- LinkManService

```
// 删除联系人
    public boolean delete(int id){
        try {
            // 1.调用dao层,传入id,返回受影响的行数
            int rows = dao.delete(id);

            // 2.判断受影响的行数:
            if (rows > 0) {
                // 2.1 如果大于0,就返回true
                return true;
            }else {
                // 2.2 如果不大于0,就返回false
                return false;
            }
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
```

- LinkManDao

```
// 删除联系人
    public int delete(int id) throws SQLException {
        // 1.写sql语句
        String sql = "delete from linkMan where id = ?";
        // 2.调用update方法
        int rows = qr.update(sql,id);
        // 3.返回受影响的行数
        return rows;
    }
```

四,小结

1. 点击了==确定之后==请求服务器删除, 用到location.href请求

```
Location.href="linkMan?method=delete&id="+id;
```

2. 思路
    - 浏览器点击了确定之后, 请求LinkManServlet, 携带method=delete,id
    - 在LinkManServlet里面创建delete()方法

```
//1.获得请求参数id
//2.调用业务 根据id删除
//3.再查询所有展示
```

- 在LinkManService里面创建deleteById()

```
public boolean deleteById(int id){
  //1.调用Dao 根据id删除
}
```

- 在LinkDao里面创建deleteById()

```
public int deleteById(int id){
  //1.使用DBUtils根据id删除
}
```

案例四:分页展示联系人

一,案例需求

显示所有联系人

编号	姓名	性别	年龄	籍贯	QQ	邮箱	操作
1	张三	男	20	广东	44444222	zs@qq.com	修改 删除
2	张三	男	20	广东	44444222	zs@qq.com	修改 删除
3	张三	男	20	广东	44444222	zs@qq.com	修改 删除
4	张三	男	20	广东	44444222	zs@qq.com	修改 删除
5	张三	男	20	广东	44444222	zs@qq.com	修改 删除

« 1 2 3 4 5 6 7 8 9 »

- 分页查询出联系人信息

二,技术分析

1,数据库分页操作 limit

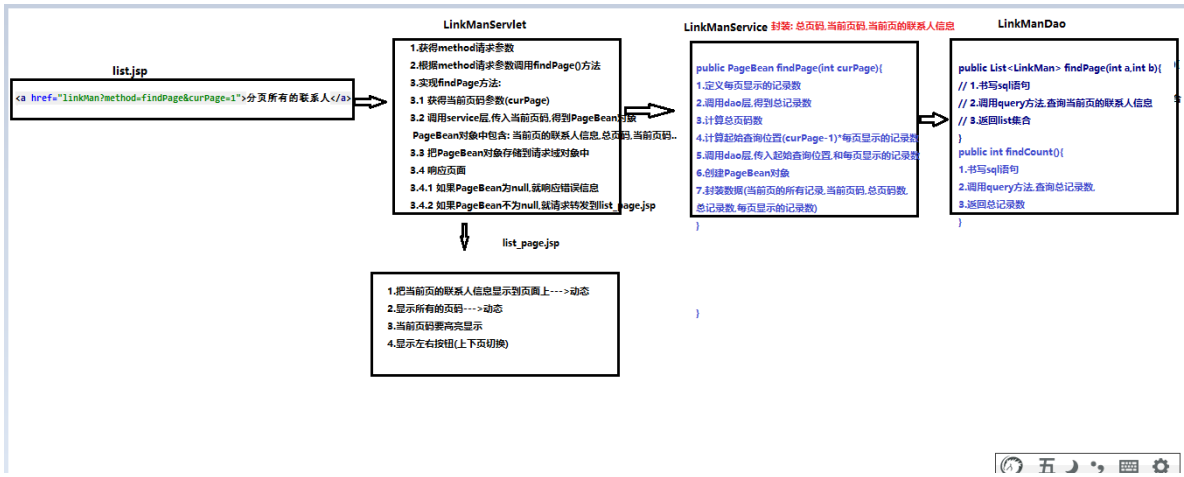
```
SELECT * FROM linkman LIMIT a,b;

-- 一, 解释limit
-- a: 从哪里开始查询            (页码-1)*每页显示条数
-- b: 每页条数(一页展示的数量) 【自定义的】

10条记录,每页显示3条
第一页: select * from linkman limit 0,3;
第二页: select * from linkman limit 3,3;
第三页: select * from linkman limit 6,3;
...
第n页: select * from linkman limit (n-1)*3,3;

-- 总记录数
-- 总页码数:
```

```
总记录数%每页显示条数==0? 总记录数/每页显示条数 : 总记录数/每页显示条数 + 1;


服务器端返回的数据:
    1.当前页所有联系人的信息
    2.当前页码
    3.总页码数
```

2. 分页需要的数据的封装

```
//封装分页的数据的
public class PageBean{
    // 前页所有联系人的信息
    private List<LinkMan> list;
    // 当前页码
    private int curPage;
    // 总页码
    private int countPage;
    // 每页显示条数
    private int pageCount;
    // 总记录数
    private int count;
    // ...
}
```

三,思路分析



四,代码实现

- index.jsp

```
    <a href="linkMan?method=findPage&curPage=1">分页所有的联系人</a><br/>
```

- LinkManServlet

```
// 分页查询
```

```
    private void findPage(HttpServletRequest request, HttpServletResponse
response) throws IOException {
        try {
            //1 获得当前页码参数(curPage)
            int curPage = Integer.parseInt(request.getParameter("curPage"));

            //2 调用service层,传入当前页码,得到PageBean对象
            PageBean pageBean = service.findPage(curPage);

            // PageBean对象中包含: 当前页的联系人信息,总页码,当前页码..
            //3 把PageBean对象存储到请求域对象中
            request.setAttribute("pageBean", pageBean);

            //4 响应页面
            if (pageBean == null) {
                //4.1 如果PageBean为null,就响应错误信息
                response.getWriter().println("<h1>服务器异常,分页查询联系人失败
</h1>");
            } else {
                //4.2 如果PageBean不为null,就请求转发到list_page.jsp

 request.getRequestDispatcher("list_page.jsp").forward(request,response);
            }
        } catch (ServletException e) {
            e.printStackTrace();
            response.getWriter().println("<h1>服务器异常</h1>");
        }
    }
```

- LinkManService

```
    // 分页查询
    public PageBean findPage(int curPage) {
        try {
            // 1.定义每页显示的记录数
            int pageCount = 2;

            // 2.调用dao层,得到总记录数
            int count = dao.findCount();

            // 3.计算总页码数:总记录数%每页显示条数==0? 总记录数/每页显示条数 : 总记录数/
每页显示条数 + 1;
            int countPage = count % pageCount == 0 ? count / pageCount : count /
pageCount + 1;

            // 4.计算起始查询位置(curPage-1)*每页显示的记录数
            int a = (curPage - 1) * pageCount;
            int b = pageCount;

            // 5.调用dao层,传入起始查询位置,和每页显示的记录数
            List<LinkMan> list = dao.findPage(a, b);

            // 6.创建PageBean对象
            PageBean pageBean = new PageBean();
            // 7.封装数据(当前页的所有记录,当前页码,总页码数,总记录数,每页显示的记录数)
```

```
            pageBean.setList(list);
            pageBean.setCurPage(curPage);
            pageBean.setCountPage(countPage);
            pageBean.setCount(count);
            pageBean.setPageCount(pageCount);

            return pageBean;
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }
```

- LinkManDao

```
// 查询总联系人数
    public int findCount() throws SQLException {
        // 1.书写sql语句
        String sql = "select count(*) from linkman";

        // 2.调用query方法,查询总记录数,
        Long count = (Long)qr.query(sql, new ScalarHandler());

        // 3.返回总记录数
        return count.intValue();
    }

    // 查询每页的联系人信息
    public List<LinkMan>  findPage(int a,int b) throws SQLException {
        // 1.书写sql语句
        String sql = "select * from linkman limit ?,?";
        // 2.调用query方法,查询当前页的数据,封装成集合
        List<LinkMan> list = qr.query(sql, new BeanListHandler<LinkMan>
(LinkMan.class), a, b);
        // 3.返回list集合
        return list;
    }
```

- list_page.jsp

```
1.引入标签库
2.显示数据
3.完成分页
 <%@ page contentType="text/html;charset=UTF-8" language="java" %>
<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core" %>
<!DOCTYPE html>
<!-- 网页使用的语言 -->
<html lang="zh-CN">
<head>
    <!-- 指定字符集 -->
    <meta charset="utf-8">
    <!-- 使用Edge最新的浏览器的渲染方式 -->
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <!-- viewport视口：网页可以根据设置的宽度自动进行适配，在浏览器的内部虚拟一个容器，
容器的宽度与设备的宽度相同。
```

```
    width: 默认宽度与设备的宽度相同
    initial-scale: 初始的缩放比，为1:1 -->
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- 上述3个meta标签*必须*放在最前面，任何其他内容都*必须*跟随其后！ -->
    <title>Bootstrap模板</title>

    <!-- 1. 导入CSS的全局样式 -->
    <link href="css/bootstrap.min.css" rel="stylesheet">
    <!-- 2. jQuery导入，建议使用1.9以上的版本 -->
    <script src="js/jquery-2.1.0.min.js"></script>
    <!-- 3. 导入bootstrap的js文件 -->
    <script src="js/bootstrap.min.js"></script>
    <style type="text/css">
        td, th {
            text-align: center;
        }
    </style>
</head>
<body>
<div class="container">
    <h3 style="text-align: center">显示所有联系人</h3>
    <table border="1" class="table table-bordered table-hover">
        <tr class="success">
            <th>编号</th>
            <th>姓名</th>
            <th>性别</th>
            <th>年龄</th>
            <th>籍贯</th>
            <th>QQ</th>
            <th>邮箱</th>
            <th>操作</th>
        </tr>

        <%--获取域对象中pageBean对象的list集合,然后遍历集合--%>
        <c:forEach items="${pageBean.list}" var="lm">
            <tr>
                <td>${lm.id}</td>
                <td>${lm.name}</td>
                <td>${lm.sex}</td>
                <td>${lm.age}</td>
                <td>${lm.address}</td>
                <td>${lm.qq}</td>
                <td>${lm.email}</td>
                <td><a class="btn btn-default btn-sm" href="修改联系人.html">
修改</a> <a class="btn btn-default btn-sm"
 onclick="deleteById(${lm.id})">删除</a></td>
            </tr>
        </c:forEach>

        <tr>
            <td colspan="8" align="center">
                <ul class="pagination success">

                    <%--如果当前页码为第1页,就不显示,否则就显示--%>
                    <c:if test="${pageBean.curPage > 1}">
                        <li><a href="linkMan?
method=findPage&curPage=${pageBean.curPage - 1}" aria-label="Previous"><span
aria-hidden="true">&laquo;</span></a></li>
```

```
                    </c:if>


                    <%--循环遍历,生成页码--%>
                   <c:forEach begin="1" end="${pageBean.countPage}" var="i">
                       <%-- class="active"  高亮显示--%>
                       <%--如果当前页面等于遍历的i的值,就高亮显示,否则就不高亮显示--
%>
                       <c:if test="${pageBean.curPage == i}">
                           <li class="active"><a href="#">${i}</a></li>
                       </c:if>

                       <c:if test="${pageBean.curPage != i}">
                           <li><a href="linkMan?
method=findPage&curPage=${i}">${i}</a></li>
                       </c:if>

                   </c:forEach>

                        <%--如果当前页码 < 总页码 就显示,否则就不显示--%>
                        <c:if test="${pageBean.curPage <
pageBean.countPage}">
                            <li><a href="linkMan?
method=findPage&curPage=${pageBean.curPage + 1}" aria-label="Previous"><span
aria-hidden="true">&raquo;</span></a></li>
                        </c:if>

                </ul>
            </td>
        </tr>
    </table>
</div>
</body>
<script>
    function deleteById(id) {
        //1.为删除标签绑定一个点击事件
        //2.在点击事件对应的函数中:
        //3.弹出确认框(confirm)
        var flag = confirm("确定要删除id为" + id + "的信息吗?");
        // 点击确定返回的是true
        // 点击取消返回的是false
        // 4.如果确认框返回的是true,就发送请求到LinkManServlet
        if (flag) {
            location.href = "linkMan?method=delete&id=" + id;
        }
    }
</script>
</html>
```

五,小结

1. limit关键字

```
limit a,b;
b: 一页展示的数量【自定义的】
a: 从哪里开始查询【a=(当前页码-1)*b】
```

2. 分页实体类 PageBean

3. 思路

- 浏览器 点击 `分页展示联系人` 请求LinkManServlet 携带(method=findPage, curPage=1)
- 在LinkManServlet里面创建findPage()方法

```
//1.获得请求参数 curPage
//2.调用业务 获得分页的数据 PageBean
//3.把pageBean存到request, 转发list_page.jsp页面展示
```

- 在LinkManService里面

```
public PageBean findPage(int curPage){
    //调用Dao 封装PageBean

}
```

- 在LinkManDao 两个方法
  - 查询联系人的列表 List
  - 统计联系人的总数量

扩展

1.PageBean优化

```java
public class PageBean<T> implements Serializable {
    // 前页所有联系人的信息
    private List<T> list;
    // 当前页码
    private int curPage;
    // 总页码
    private int countPage;
    // 每页显示条数
    private int pageCount;
    // 总记录数
    private int count;
    // ...
  }
```

2,模块Servlet优化

1. 问题:

```java
//2.判断
if ("findAll".equals(methodStr)) {
    //3.调用 findAll()
    findAll(request, response);
} else if ("add".equals(methodStr)) {
    //3.调用 add()
    add(request, response);
} else if ("delete".equals(methodStr)) {
    //3.调用 delete()
    delete(request, response);
} else if ("findPage".equals(methodStr)) {
    //3.调用 findPage()
    findPage(request, response);
}
```

   - doGet()方法里面的 if 太多, 可读性很差
   - doGet()方法里面, 每加一个请求, 都需要添加一个if, 维护麻烦

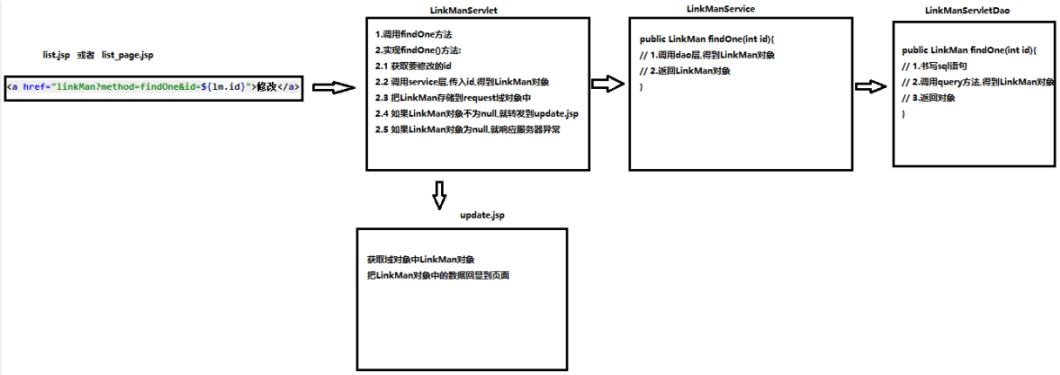
1. 解决: ==反射==
   - 一个if都不要
   - 优化之后, 代码就不变了, 不会随着请求的增加而增加if

```java
@WebServlet("/linkMan")
public class LinkManServlet extends HttpServlet {

    // 创建LinkManService对象
    private LinkManService service = new LinkManService();
```

```
    protected void doPost(HttpServletRequest request, HttpServletResponse
response) throws ServletException, IOException {
        // 1.处理乱码
        request.setCharacterEncoding("utf-8");
        response.setContentType("text/html;charset=utf-8");

        // 2.获得method请求参数
        String method = request.getParameter("method");

        // 3.判断method请求参数的值,做出对应的处理
       /* if ("findAll".equals(method)) {
            // 调用findAll方法
            findAll(request, response);

        } else if ("add".equals(method)) {
            // 调用add方法
            add(request, response);

        } else if ("delete".equals(method)) {
            // 调用delete方法
            delete(request, response);

        } else if ("findPage".equals(method)) {
            // 调用findPage方法
            findPage(request, response);

        } else if ("update".equals(method)) {
            // 调用update方法
            update(request, response);
        }*/
        try {
            // 通过反射执行
            // 1.通过反射获得要执行的方法对象
            Method m = this.getClass().getDeclaredMethod(method,
HttpServletRequest.class, HttpServletResponse.class);

            // 2.让该方法执行
            m.invoke(this, request, response);

        } catch (Exception e) {
            e.printStackTrace();
            response.getWriter().println("<h1>服务器异常</h1>");
        }

    }
}
```

扩展-修改联系人

- 需求1:update.jsp页码需要回显,要修改的联系人的信息
  - 思路分析



- 代码实现
  - list.jsp 或者 list_page.jsp

```
<a class="btn btn-default btn-sm" href="linkMan?
method=findOne&id=${lm.id}">修改</a>
```

  - LinkManServlet

```
// 查询要修改的联系人信息
    private void findOne(HttpServletRequest request,
HttpServletResponse response) throws IOException {
        //1.调用findOne方法
        //2.实现findOne()方法:
        try {
            //2.1 获取要修改的id
            int id = Integer.parseInt(request.getParameter("id"));

            //2.2 调用service层,传入id,得到LinkMan对象
            LinkMan linkMan = service.findOne(id);

            //2.3 把LinkMan存储到request域对象中
            request.setAttribute("linkMan",linkMan);

            if (linkMan != null) {
                //2.4 如果LinkMan对象不为null,就转发到update.jsp

 request.getRequestDispatcher("update.jsp").forward(request,respons
e);
            }else {
                //2.5 如果LinkMan对象为null,就响应服务器异常
                response.getWriter().println("<h1>服务器异常,查询要修改
的联系人失败</h1>");
            }
        } catch (Exception e) {
            e.printStackTrace();
            response.getWriter().println("<h1>服务器异常</h1");
        }
    }
```

  - LinkManService

```
// 查询要修改的那条记录
    public LinkMan findOne(int id) {
        try {
            // 1.调用dao层,得到LinkMan对象
            LinkMan linkMan = dao.findOne(id);
            // 2.返回LinkMan对象
            return linkMan;
        } catch (SQLException e) {
            e.printStackTrace();
            return null;
        }
    }
```

- LinkManDao

```
 // 查询要修改的联系人信息
    public LinkMan findOne(int id) throws SQLException {
        // 1.书写sql语句
        String sql = "select * from linkman where id = ?";
        // 2.调用query方法,得到LinkMan对象
        LinkMan linkMan = qr.query(sql, new BeanHandler<LinkMan>
(LinkMan.class), id);
        // 3.返回对象
        return linkMan;
    }
```

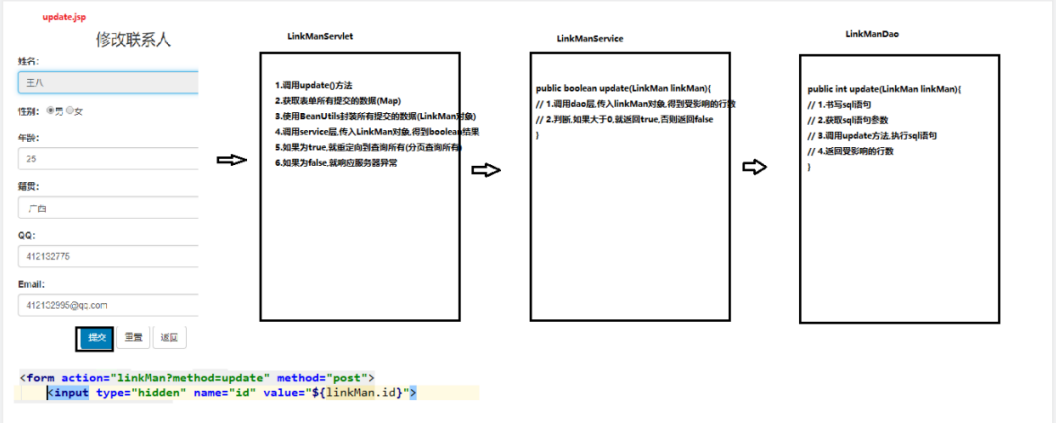
- update.jsp

```
<form action="linkMan?method=update" method="post">
            <input type="hidden" name="id" value="${linkMan.id}">
      <div class="form-group">
        <label for="name">姓名: </label>
        <input type="text" value="${linkMan.name}" class="form-
control" id="name" name="name"  readonly="readonly" placeholder="请
输入姓名" />
      </div>

      <div class="form-group">
        <label>性别: </label>
          <%--判断:如果是男,就勾选男单选框,如果是女,就勾选女单选框--%>
          <c:if test="${linkMan.sex == '男'}">
              <input type="radio" name="sex" value="男"
checked="checked" />男
              <input type="radio" name="sex" value="女"  />女
          </c:if>

          <c:if test="${linkMan.sex == '女'}">
              <input type="radio" name="sex" value="男"/>男
              <input type="radio" name="sex" value="女"
 checked="checked"  />女
          </c:if>

      </div>
```

```
      <div class="form-group">
        <label for="age">年龄：</label>
        <input type="text" value="${linkMan.age}" class="form-
control" id="age"  name="age" placeholder="请输入年龄" />
      </div>

      <div class="form-group">
        <label for="address">籍贯：</label>

          <c:if test="${linkMan.address == '广东'}">
              <select name="address" class="form-control" >
                  <option value="广东" selected="selected">广东
</option>
                  <option value="广西">广西</option>
                  <option value="湖南">湖南</option>
              </select>
          </c:if>

          <c:if test="${linkMan.address == '广西'}">
              <select name="address" class="form-control" >
                  <option value="广东">广东</option>
                  <option value="广西"  selected="selected">广西
</option>
                  <option value="湖南">湖南</option>
              </select>
          </c:if>

          <c:if test="${linkMan.address == '湖南'}">
              <select name="address" class="form-control" >
                  <option value="广东" >广东</option>
                  <option value="广西">广西</option>
                  <option value="湖南" selected="selected">湖南
</option>
              </select>
          </c:if>


      </div>

      <div class="form-group">
        <label for="qq">QQ：</label>
        <input type="text" value="${linkMan.qq}" class="form-
control" name="qq" placeholder="请输入QQ号码"/>
      </div>

      <div class="form-group">
        <label for="email">Email：</label>
        <input type="text" value="${linkMan.email}" class="form-
control" name="email" placeholder="请输入邮箱地址"/>
      </div>

         <div class="form-group" style="text-align: center">
            <input class="btn btn-primary" type="submit" value="提
交" />
            <input class="btn btn-default" type="reset" value="重置"
/>
```

```
            <input class="btn btn-default" type="button" value="返回"/>
        </div>
    </form>
```

- 需求2: 修改联系人信息
  - 思路分析



  - 代码实现
    - update.jsp

```
<form action="linkMan?method=update" method="post">
<input type="hidden" name="id" value="${linkMan.id}">
```

    - LinkManServlet

```
// 修改联系人
    private void update(HttpServletRequest request,
HttpServletResponse response) throws IOException {
        try {
            //1.调用update()方法
            //2.获取表单所有提交的数据(Map)
            Map<String, String[]> map = request.getParameterMap();

            //3.使用BeanUtils封装所有提交的数据(LinkMan对象)
            LinkMan linkMan = new LinkMan();
            BeanUtils.populate(linkMan,map);

            //4.调用service层,传入LinkMan对象,得到boolean结果
            boolean flag = service.update(linkMan);

            if (flag) {
                //5.如果为true,就重定向到查询所有(分页查询所有)
                response.sendRedirect("linkMan?method=findAll");
            }else {
                //6.如果为false,就响应服务器异常
                response.getWriter().println("<h1>服务器异常,修改联系人
失败</h1");
            }
        } catch (Exception e) {
            e.printStackTrace();
```

```
            response.getWriter().println("<h1>服务器异常</h1");
        }
    }
```

- LInkManService

```
// 修改联系人信息
    public boolean update(LinkMan linkMan) {
        try {
            // 1.调用dao层,传入linkMan对象,得到受影响的行数
            int rows = dao.update(linkMan);

            // 2.判断,如果大于0,就返回true,否则返回false
            if (rows > 0) {
                return true;
            } else {
                return false;
            }
        } catch (SQLException e) {
            e.printStackTrace();
            return false;
        }
    }
```

- LinkManDao

```
// 修改联系人信息
    public int update(LinkMan linkMan) throws SQLException {
        // 1.书写sql语句
        String sql = "update linkman set
name=?,sex=?,age=?,address=?,qq=?,email=? where id = ? ";
        // 2.获取sql语句参数
        Object[] args = {
                linkMan.getName(),
                linkMan.getSex(),
                linkMan.getAge(),
                linkMan.getAddress(),
                linkMan.getQq(),
                linkMan.getEmail(),
                linkMan.getId()
        };
        // 3.调用update方法,执行sql语句
        int rows = qr.update(sql, args);
        // 4.返回受影响的行数
        return rows;
    }
```