

day32-综合案例

案例一:添加用户

一,案例需求

- 1. 点击添加用户跳转添加用户页面

显示所有用户

编号	姓名	性别	年龄	籍贯	QQ	邮箱	操作
1	张三	男	11	广东	766335435	766335435@qq.com	<div>修改删除</div>
2	李四	男	12	广东	243424242	243424242@qq.com	<div>修改删除</div>
3	王五	女	13	广东	474574574	474574574@qq.com	<div>修改删除</div>
4	赵六	女	18	广东	77777777	77777777@qq.com	<div>修改删除</div>
5	钱七	女	15	湖南	412132145	412132145@qq.com	<div>修改删除</div>
6	王八	男	25	广西	412132775	412132995@qq.com	<div>修改删除</div>
<div>添加用户</div>							

- 2. 在添加用户页面，点击提交按钮,把数据提交到服务器,保存到数据库

添加用户页面

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，可以查看到新建的用户信息

显示所有用户

编号	姓名	性别	年龄	籍贯	QQ	邮箱	操作
1	张三	男	11	广东	766335435	766335435@qq.com	<input type="button" value="修改"/> <input type="button" value="删除"/>
2	李四	男	12	广东	243424242	243424242@qq.com	<input type="button" value="修改"/> <input type="button" value="删除"/>
3	王五	女	13	广东	474574574	474574574@qq.com	<input type="button" value="修改"/> <input type="button" value="删除"/>
4	赵六	女	18	广东	77777777	77777777@qq.com	<input type="button" value="修改"/> <input type="button" value="删除"/>
5	钱七	女	15	湖南	412132145	412132145@qq.com	<input type="button" value="修改"/> <input type="button" value="删除"/>
6	王八	男	25	广西	412132775	412132995@qq.com	<input type="button" value="修改"/> <input type="button" value="删除"/>
<input type="button" value="添加用户"/>							

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.

- 传统方式

查询所有的用户 :<http://localhost:8080/day32/findAll>

添加用户: :<http://localhost:8080/day32/add>

删除用户 :<http://localhost:8080/day32/delete>

- 以模块为单位创建, 创建一个UserServlet

查询所有的用户 :<http://localhost:8080/day32/userServlet?method=findAll>

添加用户: :<http://localhost:8080/day32/userServlet?method=add>

删除用户 :<http://localhost:8080/day32/userServlet?method=delete>

```

class UserServlet extend HttpServlet{
    public void doGet(HttpServletRequest request, HttpServletResponse response){
        //1. 获得method请求参数的值
        String methodStr = request.getParameter("method");
        //2. 判断methodStr对应的是哪一个请求
        if("findAll".equals(methodStr)){
            //查询所有的用户
            findAll(request,response);

        }else if("add".equals(methodStr)){
            //添加用户
            add(request,response);
        }else if("delete".equals(methodStr)){
            //删除用户
            delete(request,response);
        }
    }

    //查询所有用户的方法
    public void findAll(HttpServletRequest request, HttpServletResponse response){
        //1. 获得请求参数
        //2. 调用业务
        //3.分发转向
    }

    //添加用户的方法
    public void add(HttpServletRequest request, HttpServletResponse response){
        //1. 获得请求参数
        //2. 调用业务
        //3.分发转向
    }

    //删除用户的方法
    public void delete(HttpServletRequest request, HttpServletResponse response){
        //1. 获得请求参数
        //2. 调用业务
        //3.分发转向
    }
}

```

三,思路分析

四代码实现

- UserServlet

```

@WebServlet("/userServlet")
public class UserServlet extends javax.servlet.http.HttpServlet {
    protected void doGet(javax.servlet.http.HttpServletRequest request,
        javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
        IOException {
        //0.处理post请求的乱码
        request.setCharacterEncoding("utf-8");
        //1. 获得method请求参数
        String methodStr = request.getParameter("method");
        //2. 判断methodStr对应的是哪一个请求
        if ("findAll".equals(methodStr)) {
            //查询所有的用户
            findAll(request, response);
        } else if ("add".equals(methodStr)) {
            //添加用户
            add(request, response);
        }
    }
}

/**
 * 添加用户
 * @param request
 * @param response
 */
public void add(HttpServletRequest request, HttpServletResponse response) {
    try {
        //1. 获得请求参数，封装成User
        Map<String, String[]> map = request.getParameterMap();
        User user = new User();
        BeanUtils.populate(user, map);

        //2. 调用业务
        UserService userService = new UserService();
        userService.addUser(user);
        //3. 再查询所有展示(用重定向)
        //findAll(request, response)
        response.sendRedirect("http://localhost:8080/web32a-user/userServlet?
method=findAll");

    } catch (Exception e) {
        e.printStackTrace();
    }
}
}

```

- UserService

```
public class UserService {
    public void addUser(User user)throws Exception {
        //调用Dao
        UserDao userDao = new UserDao();
        userDao.save(user);
    }
}
```

- UserDao

```
public class UserDao {

    public void save(User user) throws Exception{
        JdbcTemplate jdbcTemplate = new JdbcTemplate(C3P0Utils.getDataSource());
        String sql = "insert into user values(?,?,?, ?, ?, ?, ?)";
        Object[] params =
{null,user.getName(),user.getSex(),user.getAge(),user.getAddress(),user.getQq(),user.getEmail()};
        ;

        jdbcTemplate.update(sql,params);

    }
}
```

案例二:删除用户

一,案例需求

编号	姓名	性别	年龄	地区	电话	邮箱	操作
1	张三	男	11	广东	766335435	766335435@qq.com	修改 删除
2	李四	男	12	广东	243424242	243424242@qq.com	修改 删除
3	王五	女	13	广东	474574574	474574574@qq.com	修改 删除
4	赵六	女	18	广东	77777777	77777777@qq.com	修改 删除
5	钱七	女	15	湖南	412132145	412132145@qq.com	修改 删除
6	王八	男	25	广西	412132775	412132995@qq.com	修改 删除

添加用户

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

- UserServlet

```

@WebServlet("/userServlet")
public class UserServlet extends javax.servlet.http.HttpServlet {
    protected void doGet(javax.servlet.http.HttpServletRequest request,
        javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
        IOException {
        //0. 处理post请求的乱码
        request.setCharacterEncoding("utf-8");
        //1. 获得method请求参数
        String methodStr = request.getParameter("method");
        //2. 判断methodStr对应的是哪一个请求
        if ("findAll".equals(methodStr)) {
            //查询所有的用户
            findAll(request, response);
        } else if ("add".equals(methodStr)) {
            //添加用户
            add(request, response);
        } else if ("deleteById".equals(methodStr)) {
            //根据id删除用户
            deleteById(request, response);
        }
    }

    /**
     * 根据id删除用户
     * @param request
     * @param response
     */
    public void deleteById(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {
        try {
            //1. 获得请求参数(id)
            int id = Integer.parseInt(request.getParameter("id"));
            //2. 调用业务
            UserService userService = new UserService();
            userService.deleteById(id);
            //3. 再查询所有展示
            response.sendRedirect("http://localhost:8080/web32a-user/userServlet?method=findAll");
        } catch (Exception e) {
            e.printStackTrace();
            //向request存msg
            request.setAttribute("msg", "服务器异常...");
            //转发到msg.jsp
            request.getRequestDispatcher("/msg.jsp").forward(request, response);
        }
    }
}

```

- UserService

```
public class UserService {

    public void deleteById(int id) throws Exception {
        //调用Dao
        UserDao userDao = new UserDao();
        userDao.deleteById(id);
    }
}
```

- UserDao

```
public class UserDao {
    public void deleteById(int id) throws Exception {
        JdbcTemplate jdbcTemplate = new JdbcTemplate(C3P0Utils.getDataSource());
        String sql = "delete from user where id = ?";
        jdbcTemplate.update(sql,id);

    }
}
```

案例三:分页展示用户

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	<button>修改</button> <button>删除</button>
2	张三	男	20	广东	44444222	zs@qq.com	<button>修改</button> <button>删除</button>
3	张三	男	20	广东	44444222	zs@qq.com	<button>修改</button> <button>删除</button>
4	张三	男	20	广东	44444222	zs@qq.com	<button>修改</button> <button>删除</button>
5	张三	男	20	广东	44444222	zs@qq.com	<button>修改</button> <button>删除</button>

« 1 2 3 4 5 6 7 8 9 »

- 分页查询出用户信息

二,技术分析

1,数据库操作 limit

```
select * from product limit a,b;
```

a(从哪里开始查询) : $a = (\text{当前页码} - 1) * b$;

b(一页显示的数量): 是由开发者自定义(需求)的; $b = 2$

```
-----  
-- 分页查询用户; 一页显示5条 b=5;  
-- limit a ,b; a = (当前页码-1)*b;  
-- 第一页 a =0, b = 5  
SELECT * FROM `user` LIMIT 0,5  
-- 第二页 a =5, b = 5  
SELECT * FROM `user` LIMIT 5,5  
-- 第三页 a =10, b = 5  
SELECT * FROM `user` LIMIT 10,5
```

2. 分页需要的数据的封装

```
//封装分页的数据的  
public class PageBean{  
    //一页显示的用户的集合 List<User> list  
    //当前页码 int curPage;  
    //总页数 int sumPage;  
    //用户的总数 int count;  
    //一页显示的数量 int curSize;  
}  
  
//-----计算sumPage-----//  
  
一页显示2条数据, 总数量 8, 总页数4  
一页显示3条数据, 总数量 8, 总页数3  
一页显示4条数据, 总数量 8, 总页数2  
一页显示4条数据, 总数量 9, 总页数3  
  
if(count % curSize ==0){  
    sumPage = count/curSize;  
}else{  
    sumPage = count/curSize+1;  
}
```

三,思路分析

三,代码实现

- UserServicelet


```

@WebServlet("/userServlet")
public class UserServlet extends javax.servlet.http.HttpServlet {
    protected void doGet(javax.servlet.http.HttpServletRequest request,
        javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
        IOException {
        //0.处理post请求的乱码
        request.setCharacterEncoding("utf-8");
        //1. 获得method请求参数
        String methodStr = request.getParameter("method");
        //2. 判断methodStr对应的是哪一个请求
        if ("findAll".equals(methodStr)) {
            //查询所有的用户
            findAll(request, response);
        } else if ("add".equals(methodStr)) {
            //添加用户
            add(request, response);
        } else if ("deleteById".equals(methodStr)) {
            //根据id删除用户
            deleteById(request, response);
        } else if ("findByPage".equals(methodStr)) {
            //分页查询用户
            findByPage(request, response);
        }
    }

    /**
     * 分页查询用户
     * @param request
     * @param response
     */
    public void findByPage(HttpServletRequest request, HttpServletResponse response) throws
        ServletException, IOException {
        try {
            //1. 获得请求参数(当前查询的哪一页)
            int curPage = Integer.parseInt(request.getParameter("curPage"));
            //2. 调用业务
            UserService userService = new UserService();
            PageBean pageBean = userService.findByPage(curPage);

            //3.把pageBean存到request, 转发 list_page.jsp
            request.setAttribute("pageBean", pageBean);
            request.getRequestDispatcher("/list_page.jsp").forward(request, response);
        } catch (Exception e) {
            e.printStackTrace();
            //向request存msg
            request.setAttribute("msg", "服务器异常...");
            //转发到msg.jsp
            request.getRequestDispatcher("/msg.jsp").forward(request, response);
        }
    }
}

```

```
}
```

- UserService

```
public class UserService {

    /**
     * 分页查询数据
     * 调用Dao,封装pageBean
     * @param curPage
     * @return
     */
    public PageBean findByPage(int curPage) throws Exception {
        //0 创建Dao
        UserDao userDao = new UserDao();

        //一页显示的数量
        int curSize = Constants.USER_CUR_SIZE;
        //用户总数量(查询数据库的 select count(*) ...)
        int count = userDao.findCount();
        //总页码
        int sumPage = 0;
        if(count % curSize == 0){
            sumPage = count/curSize;
        }else{
            sumPage = count/curSize+1;
        }

        //一页显示用户的List (查询数据库的 select * from user limit a,b)
        int b = curSize;
        int a = (curPage -1)*b;
        List<User> list = userDao.findLimit(a,b);

        //1. 创建PageBean
        PageBean pageBean = new PageBean(list,curPage,sumPage,count,curSize);
        /* //a. 封装当前页码
        pageBean.setCurPage(curPage);
        //b.封装一页显示的数量
        pageBean.setCurSize(curSize);
        //c. 封装用户总数量(查询数据库的 select count(*) ...)
        pageBean.setCount(count);
        //d 封装总页码
        pageBean.setSumPage(sumPage);
        //e 封装一页显示用户的List (查询数据库的 select * from user limit a,b)
        pageBean.setList(list);*/

        return pageBean;

    }
}
```

- UserDao

```
public class UserDao {

    //统计用户总数量
    public int findCount() throws Exception {
        JdbcTemplate jdbcTemplate = new JdbcTemplate(C3P0Utils.getDataSource());
        String sql = "select count(*) from user";
        Long n = jdbcTemplate.queryForObject(sql, Long.class);
        return n.intValue();
    }

    //查询用户一页显示的List
    public List<User> findLimit(int a, int b) throws Exception {
        JdbcTemplate jdbcTemplate = new JdbcTemplate(C3P0Utils.getDataSource());
        String sql = "select * from user limit ?,?";
        List<User> list = jdbcTemplate.query(sql, new BeanPropertyRowMapper<>(User.class), a, b);
        return list;
    }
}
```

扩展

一, PageBean优化

```
/**
 * 分页的数据的封装类
 */
public class PageBean<T> {

    //一页显示的用户的集合 List<User> list
    //当前页码          int curPage;
    //总页数            int sumPage;
    //用户的总数        int count;
    //一页显示的数量    int curSize;

    private List<T> list;
    private int curPage;
    private int sumPage;
    private int count;
    private int curSize;

    public void setList(List<T> list) {
        this.list = list;
    }

    public List<T> getList() {
        return list;
    }

    public int getCurPage() {
        return curPage;
    }

    public void setCurPage(int curPage) {
        this.curPage = curPage;
    }

    public int getSumPage() {
        return sumPage;
    }

    public void setSumPage(int sumPage) {
        this.sumPage = sumPage;
    }

    public int getCount() {
        return count;
    }

    public void setCount(int count) {
        this.count = count;
    }

    public int getCurSize() {
```

```

        return curSize;
    }

    public void setCurSize(int curSize) {
        this.curSize = curSize;
    }
}

```

二, doGet()方法里面的优化

```

@WebServlet("/userServlet")
public class UserServlet extends javax.servlet.http.HttpServlet {
    //1. 判断是哪一个方法 2. 让这个方法执行
    protected void doGet(javax.servlet.http.HttpServletRequest request,
        javax.servlet.http.HttpServletResponse response) throws javax.servlet.ServletException,
        IOException {
        try {
            //0. 处理post请求的乱码
            request.setCharacterEncoding("utf-8");
            //1. 获得method请求参数(说白了就是方法名)
            String methodStr = request.getParameter("method");
            //2. 获得字节码对象
            Class clazz = this.getClass();
            //3. 反射获得methodStr(方法名)对应的Method对象
            Method method = clazz.getMethod(methodStr, HttpServletRequest.class,
                HttpServletResponse.class);
            //4. 让这个方法执行
            method.invoke(this, request, response);

        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}

```