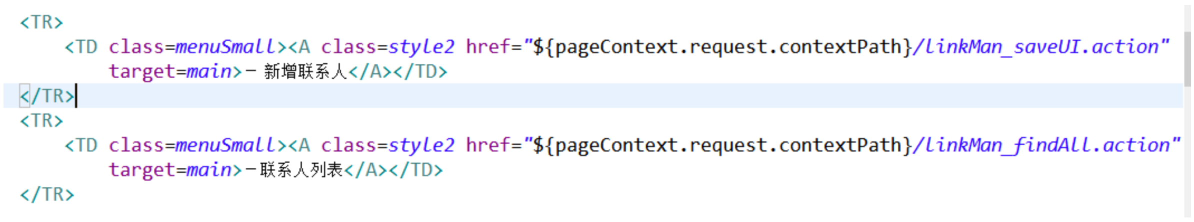
1.1 CRM综合练习_联系人管理

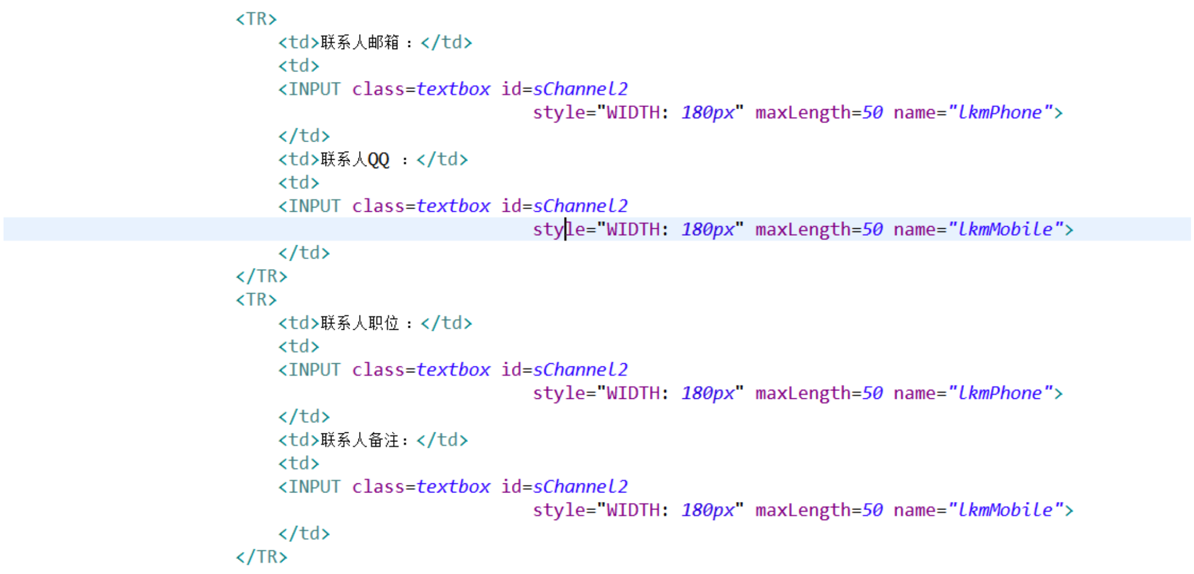
1.1.1 保存联系人

1.1.1.1 修改menu.jsp的链接



```
<TR>
    <TD class=menuSmall><A class=style2 href="${pageContext.request.contextPath}/linkMan_saveUI.action"
        target=main>－ 新增联系人</A></TD>
</TR>
<TR>
    <TD class=menuSmall><A class=style2 href="${pageContext.request.contextPath}/linkMan_findAll.action"
        target=main>－联系人列表</A></TD>
</TR>
```

- 在页面添加表单项



```
<TR>
    <td>联系人邮箱：</td>
    <td>
    <INPUT class=textbox id=sChannel2
                            style="WIDTH: 180px" maxLength=50 name="lkmPhone">
    </td>
    <td>联系人QQ ：</td>
    <td>
    <INPUT class=textbox id=sChannel2
                            style="WIDTH: 180px" maxLength=50 name="lkmMobile">
    </td>
</TR>
<TR>
    <td>联系人职位：</td>
    <td>
    <INPUT class=textbox id=sChannel2
                            style="WIDTH: 180px" maxLength=50 name="lkmPhone">
    </td>
    <td>联系人备注：</td>
    <td>
    <INPUT class=textbox id=sChannel2
                            style="WIDTH: 180px" maxLength=50 name="lkmMobile">
    </td>
</TR>
```

1.1.1.2 编写Action中的saveUI的方法

LinkManAction.java

```
package com.admiral.crm.web.action;

import org.hibernate.criterion.DetachedCriteria;

import com.admiral.crm.domain.LinkMan;
import com.admiral.crm.domain.PageBean;
import com.admiral.crm.service.LinkManService;
import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;
```

```
public class LinkManAction extends ActionSupport implements ModelDriven<LinkMan>
{

    //模型驱动使用的对象
    private LinkMan linkMan = new LinkMan();
    @Override
    public LinkMan getModel() {
        return linkMan;
    }

    //注入 Service
    private LinkManService linkManService;

    public void setLinkManService(LinkManService linkManService) {
        this.linkManService = linkManService;
    }

    private Integer currPage = 1;
    private Integer pageSize = 3;

    public void setCurrPage(Integer currPage) {
        if(currPage==null) {
            currPage = 1;
        }
        this.currPage = currPage;
    }

    public void setPageSize(Integer pageSize) {
        if (pageSize == null) {
            pageSize = 3;
        }
        this.pageSize = pageSize;
    }

    /**
     *
     * Title: findAll
     * Description: 待条件分页查询联系人
     * @return
     */
    public String findAll() {
        //创建离线条件查询
        DetachedCriteria detachedCriteria =
DetachedCriteria.forClass(LinkMan.class);
        //设置条件

        //调用业务层
        PageBean<LinkMan> pageBean =
linkManService.findAll(detachedCriteria,currPage,pageSize);
        //将 pageBean 压入值栈中
        ActionContext.getContext().getValueStack().push(pageBean);
        return "findAll";
    }

    /**
     *
     * Title: saveUI
     * Description: 联系人模块:跳转到添加页面的方法
     * @return
     */
    public String saveUI() {

        return "saveUI";
    }

}
```

- 配置跳转

```
        <!-- 配置联系人模块 -->
        <action name="linkMan_*" class="linkManAction" method="{1}">
            <result name="findAll">/jsp/linkman/list.jsp</result>
            <result name="saveUI">/jsp/linkman/add.jsp</result>
        </action>
```

1.1.1.3 编写Service

- 修改 saveUI 方法

```
    // 注入 CustomerService
    private CustomerService customerService;
    public void setCustomerService(CustomerService customerService) {
        this.customerService = customerService;
    }

    /**
     *
     * Title: saveUI
     * Description: 联系人模块:跳转到添加页面的方法
     * @return
     */
    public String saveUI() {
        // 查询所有客户
        List<Customer> list = customerService.findAll();
        // 将所有客户存入值栈
        ActionContext.getContext().getValueStack().set("list", list);
        return "saveUI";
    }
```

- 在 applicationContext.xml 中注入

```xml
    <!-- ===============  LinkMan 模块的相关配置  ====================== -->
    <bean id="linkManAction" class="com.admiral.crm.web.action.LinkManAction">
        <property name="linkManService" ref="linkManService" />
        <property name="customerService" ref="customerService" />
    </bean>
    <bean id="linkManService"
class="com.admiral.crm.service.impl.LinkManServiceImpl">
        <property name="linkManDao" ref="linkManDao" />
    </bean>
    <bean id="linkManDao" class="com.admiral.crm.dao.impl.LinkManDaoImpl">
        <property name="sessionFactory" ref="sessionFactory" />
    </bean>
```

- 编写 Service

CustomerServiceImpl.java

```java
package com.admiral.crm.service.impl;

import java.util.List;

import org.hibernate.criterion.DetachedCriteria;
import org.springframework.transaction.annotation.Transactional;

import com.admiral.crm.dao.CustomerDao;
import com.admiral.crm.domain.Customer;
import com.admiral.crm.domain.PageBean;
import com.admiral.crm.service.CustomerService;

/**
 * 客户管理的 Service 的实现类
 */
@Transactional
public class CustomerServiceImpl implements CustomerService {

    // 注入客户的 Dao
    private CustomerDao customerDao;

    public void setCustomerDao(CustomerDao customerDao) {
        this.customerDao = customerDao;
    }

    @Override
    public void save(Customer customer) {
        customerDao.save(customer);
    }

    @Override
    public PageBean<Customer> findByPage(DetachedCriteria detachedCriteria,
Integer currPage, Integer pageSize) {

        PageBean<Customer> pageBean = new PageBean<Customer>();
        //封装当前页
        pageBean.setCurrPage(currPage);

        //封装每页记录数
        pageBean.setPageSize(pageSize);

        //封装总记录数:查询
        //调用Dao
        Integer totalCount = customerDao.findCount(detachedCriteria);
        pageBean.setTotalCount(totalCount);

        //封装总页数:计算
        Double num = Math.ceil(totalCount.doubleValue()/pageSize);
        pageBean.setTotalPage(num.intValue());

        //封装每页显示的数据的集合
        Integer begin = (currPage - 1) * pageSize;
        List<Customer> customers  =
customerDao.findByPage(detachedCriteria,begin,pageSize);
        pageBean.setList(customers);
        return pageBean;
    }

    @Override
    public Customer findById(Long cust_id) {
        return customerDao.findById(cust_id);
    }

    @Override
    public void delete(Customer customer) {
        customerDao.delete(customer);
    }

    @Override
    public void update(Customer customer) {
        customerDao.update(customer);
    }

    @Override
    public List<Customer> findAll() {
        return customerDao.findAll();
    }

}
```

1.1.1.4 编写DAO

CustomerDaoImpl.java

```
package com.admiral.crm.dao.impl;

import java.util.List;
import org.hibernate.criterion.DetachedCriteria;
import org.hibernate.criterion.Projections;
import org.springframework.orm.hibernate5.support.HibernateDaoSupport;

import com.admiral.crm.dao.CustomerDao;
import com.admiral.crm.domain.Customer;

/**
 * 客户管理的Dao实现类
 */
public class CustomerDaoImpl extends HibernateDaoSupport implements CustomerDao
{

    @Override
    public void save(Customer customer) {
        this.getHibernateTemplate().save(customer);
    }

    @Override
    public Integer findCount(DetachedCriteria detachedCriteria) {
        //
        detachedCriteria.setProjection(Projections.rowCount());
        List<Long> list = (List<Long>)
this.getHibernateTemplate().findByCriteria(detachedCriteria);
        if(list!=null && list.size()>0) {
            return list.get(0).intValue();
        }
        return null;
    }

    @Override
    public List<Customer> findByPage(DetachedCriteria detachedCriteria, Integer
begin, Integer pageSize) {
        detachedCriteria.setProjection(null);
        return (List<Customer>)
this.getHibernateTemplate().findByCriteria(detachedCriteria, begin, pageSize);
    }

    @Override
    public Customer findById(Long cust_id) {
        return this.getHibernateTemplate().get(Customer.class, cust_id);
    }

    @Override
    public void delete(Customer customer) {
        this.getHibernateTemplate().delete(customer);
    }

    @Override
    public void update(Customer customer) {
        this.getHibernateTemplate().update(customer);
    }

    @Override
    public List<Customer> findAll() {
        return (List<Customer>) this.getHibernateTemplate().find("from
Customer");
```

```
    }

}
```

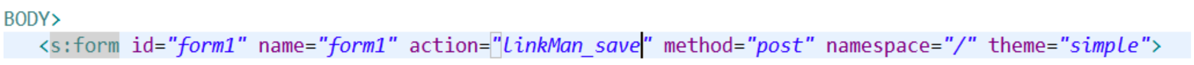
1.1.1.5 修改添加页面（改成struts2的标签）

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="/struts-tags" prefix="s"%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<TITLE>添加联系人</TITLE>
<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
<LINK href="${pageContext.request.contextPath }/css/Style.css" type=text/css
rel=stylesheet>
<LINK href="${pageContext.request.contextPath }/css/Manage.css" type=text/css
    rel=stylesheet>


<META content="MSHTML 6.00.2900.3492" name=GENERATOR>
</HEAD>
<BODY>
    <s:form id="form1" name="form1" action="" method="post" namespace="/"
theme="simple">

        <TABLE cellSpacing=0 cellPadding=0 width="98%" border=0>
            <TBODY>
                <TR>
                    <TD width=15><IMG src="${pageContext.request.contextPath
}/images/new_019.jpg"
                        border=0></TD>
                    <TD width="100%"
background="${pageContext.request.contextPath }/images/new_020.jpg"
                         height=20></TD>
                    <TD width=15><IMG src="${pageContext.request.contextPath
}/images/new_021.jpg"
                        border=0></TD>
                </TR>
            </TBODY>
        </TABLE>
        <TABLE cellSpacing=0 cellPadding=0 width="98%" border=0>
            <TBODY>
                <TR>
                    <TD width=15 background=${pageContext.request.contextPath
}/images/new_022.jpg><IMG
                        src="${pageContext.request.contextPath
}/images/new_022.jpg" border=0></TD>
                    <TD vAlign=top width="100%" bgColor=#ffffff>
                        <TABLE cellSpacing=0 cellPadding=5 width="100%"
border=0>
```

```
                                <TR>
                                    <TD class=manageHead>当前位置：联系人管理 &gt; 添加联
系人</TD>
                                </TR>
                                <TR>
                                    <TD height=2></TD>
                                </TR>
                            </TABLE>
                            <TABLE cellSpacing=0 cellPadding=5  border=0>
                                <tr>
                                    <td>所属客户：</td>
                                    <td colspan="3">
                                        <s:select list="list"
name="customer.cust_id" headerKey="" headerValue="-请选择-" listKey="cust_id"
listValue="cust_name"/>
                                    </td>
                                </tr>
                                <TR>
                                    <td>联系人名称：</td>
                                    <td>
                                        <s:textfield class="textbox" id="sChannel2"
cssStyle="WIDTH: 180px" maxLength="50" name="lkm_name"/>
                                    </td>
                                    <td>联系人性别：</td>
                                    <td>
                                        <s:radio list="#{'1':'男','2':'女'}"
name="lkm_gender"/>
                                    </td>
                                </TR>
                                <TR>
                                    <td>联系人办公电话 ：</td>
                                    <td>
                                        <s:textfield class="textbox" id="sChannel2"
cssStyle="WIDTH: 180px" maxLength="50" name="lkm_phone"/>
                                    </td>
                                    <td>联系人手机 ：</td>
                                    <td>
                                        <s:textfield class="textbox" id="sChannel2"
cssStyle="WIDTH: 180px" maxLength="50" name="lkm_mobile"/>
                                    </td>
                                </TR>
                                <TR>
                                    <td>联系人邮箱 ：</td>
                                    <td>
                                        <s:textfield class="textbox" id="sChannel2"
cssStyle="WIDTH: 180px" maxLength="50" name="lkm_email"/>
                                    </td>
                                    <td>联系人QQ ：</td>
                                    <td>
                                        <s:textfield class="textbox" id="sChannel2"
cssStyle="WIDTH: 180px" maxLength="50" name="lkm_qq"/>
                                    </td>
                                </TR>
                                <TR>
                                    <td>联系人职位 ：</td>
                                    <td>
                                        <s:textfield class="textbox" id="sChannel2"
cssStyle="WIDTH: 180px" maxLength="50" name="lkm_position"/>
```

```
                                </td>
                                <td>联系人备注：</td>
                                <td>
                                    <s:textarea name="lkm_memo" rows="3"
cols="26"/>
                                </td>
                            </TR>
                            <tr>
                                <td rowspan=2>
                                <INPUT class=button id=sButton2 type=submit
                                                        value="保存 "
name=sButton2>
                                </td>
                            </tr>
                        </TABLE>


                    </TD>
                    <TD width=15 background="${pageContext.request.contextPath
}/images/new_023.jpg">
                    <IMG src="${pageContext.request.contextPath
}/images/new_023.jpg" border=0></TD>
                </TR>
            </TBODY>
        </TABLE>
        <TABLE cellSpacing=0 cellPadding=0 width="98%" border=0>
            <TBODY>
                <TR>
                    <TD width=15><IMG src="${pageContext.request.contextPath
}/images/new_024.jpg"
                        border=0></TD>
                    <TD align=middle width="100%"
                        background="${pageContext.request.contextPath
}/images/new_025.jpg" height=15></TD>
                    <TD width=15><IMG src="${pageContext.request.contextPath
}/images/new_026.jpg"
                        border=0></TD>
                </TR>
            </TBODY>
        </TABLE>
    </s:form>
</BODY>
</HTML>
```

1.1.1.6 编写表单提交路径



```
BODY>
  <s:form id="form1" name="form1" action="linkMan_save" method="post" namespace="/" theme="simple">
```

1.1.1.7 编写Action的save方法

```
    /**
     *
     * Title: save
     * Description: 联系人模块:保存联系人的方法
     * @return
     */
    public String save() {
        linkManService.save(linkMan);
        return "saveSuccess";
    }
```

1.1.1.8 编写Service

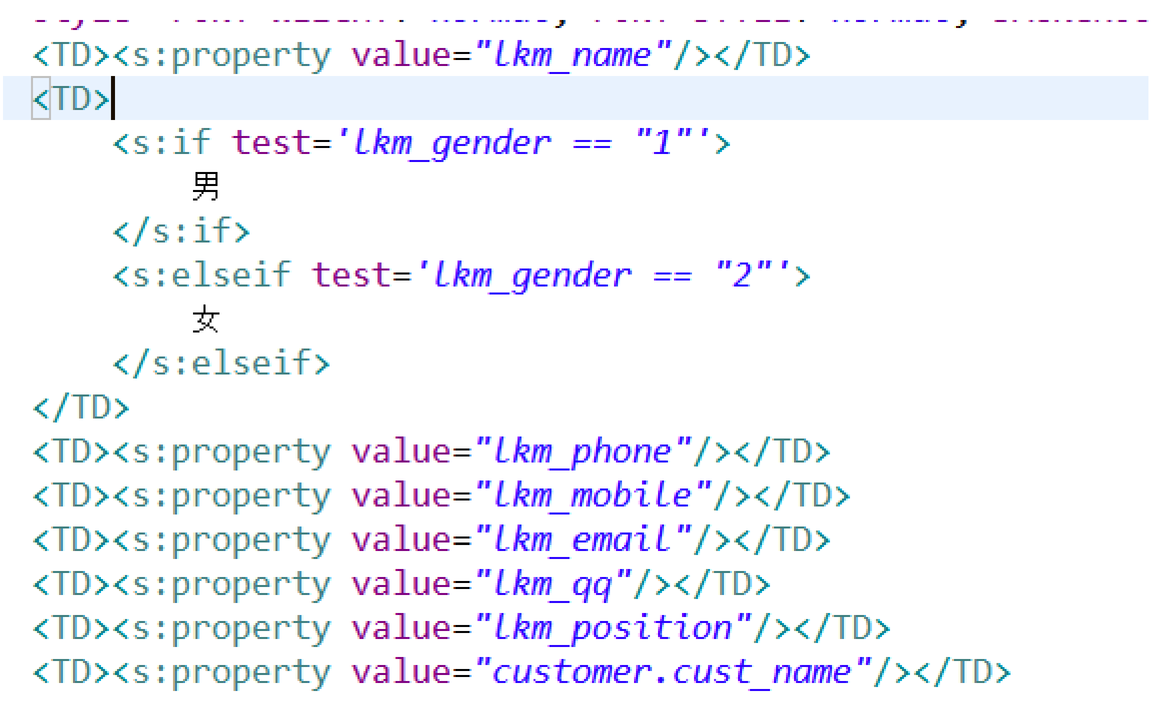
```
package com.admiral.crm.service.impl;

import java.util.List;

import org.hibernate.criterion.DetachedCriteria;
import org.springframework.transaction.annotation.Transactional;

import com.admiral.crm.dao.LinkManDao;
import com.admiral.crm.domain.LinkMan;
import com.admiral.crm.domain.PageBean;
import com.admiral.crm.service.LinkManService;

/**
 *
 * @Description: 联系人Service层的实现类
 * @author Admiral
 */
@Transactional
public class LinkManServiceImpl implements LinkManService {

    //注入 Dao
    private LinkManDao linkManDao;

    public void setLinkManDao(LinkManDao linkManDao) {
        this.linkManDao = linkManDao;
    }

    @Override
    public PageBean<LinkMan> findAll(DetachedCriteria detachedCriteria, Integer 
currPage, Integer pageSize) {
        PageBean<LinkMan> pageBean = new PageBean<LinkMan>();
        //设置当前页
        pageBean.setCurrPage(currPage);
        //设置每页的记录数
        pageBean.setPageSize(pageSize);

        //设置总记录数
        Integer totalCount = linkManDao.findCount(detachedCriteria);
        pageBean.setTotalCount(totalCount);

        //设置总页数
        Double num = Math.ceil(totalCount.doubleValue()/pageSize);
        pageBean.setTotalPage(num.intValue());

        //设置每页显示的数据
        Integer begin = (currPage-1) * pageSize;
        List<LinkMan> linkMans =
linkManDao.findByPage(detachedCriteria,begin,pageSize);
        pageBean.setList(linkMans);

        return pageBean;
    }

    @Override
    public void save(LinkMan linkMan) {
        linkManDao.save(linkMan);
    }

}
```

1.1.1.9 编写DAO

```
package com.admiral.crm.dao.impl;

import java.util.List;

import org.hibernate.criterion.DetachedCriteria;
import org.hibernate.criterion.Projections;
import org.springframework.orm.hibernate5.support.HibernateDaoSupport;

import com.admiral.crm.dao.LinkManDao;
import com.admiral.crm.domain.LinkMan;

/**
 *
 * @Description: 联系人Dao的实现类
 * @author Admiral
 */
public class LinkManDaoImpl extends HibernateDaoSupport implements LinkManDao {

    @Override
    public Integer findCount(DetachedCriteria detachedCriteria) {
        detachedCriteria.setProjection(Projections.rowCount());
        List<Long> list = (List<Long>)
this.getHibernateTemplate().findByCriteria(detachedCriteria);
        if(list!=null && list.size()>0) {
            return list.get(0).intValue();
        }
        return null;
    }
    @Override
    public List<LinkMan> findByPage(DetachedCriteria detachedCriteria, Integer
begin, Integer pageSize) {
        detachedCriteria.setProjection(null);
        return (List<LinkMan>)
this.getHibernateTemplate().findByCriteria(detachedCriteria, begin, pageSize);
    }

    @Override
    public void save(LinkMan linkMan) {
        this.getHibernateTemplate().save(linkMan);
    }

}
```

- 配置跳转

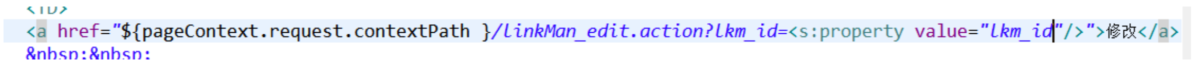
```
        <!-- 配置联系人模块 -->
        <action name="linkMan_*" class="linkManAction" method="{1}">
            <result name="findAll">/jsp/linkman/list.jsp</result>
            <result name="saveUI">/jsp/linkman/add.jsp</result>
            <result name="saveSuccess"
type="redirectAction">linkMan_findAll.action</result>
        </action>
```



```
<TD><s:property value="lkm_name"/></TD>
<TD>
    <s:if test='lkm_gender == "1"'>
        男
    </s:if>
    <s:elseif test='lkm_gender == "2"'>
        女
    </s:elseif>
</TD>
<TD><s:property value="lkm_phone"/></TD>
<TD><s:property value="lkm_mobile"/></TD>
<TD><s:property value="lkm_email"/></TD>
<TD><s:property value="lkm_qq"/></TD>
<TD><s:property value="lkm_position"/></TD>
<TD><s:property value="customer.cust_name"/></TD>
```

1.1.2 修改联系人

1.1.2.1 修改列表页面上链接



```
<a href="${pageContext.request.contextPath }/linkMan_edit.action?lkm_id=<s:property value="lkm_id"/>">修改</a>
&nbsp;&nbsp;
```

1.1.2.2 编写Action的edit方法

```
    /**
     * 
     * Title: edit
     * Description: 联系人模块:跳转到编辑页面的方法
     * @return
     */
    public String edit() {
        //查询所有客户
        List<Customer> list = customerService.findAll();
        //根据ID查询某个联系人
        linkMan = linkManService.findById(linkMan.getLkm_id());
        
        // 将所有客户和联系人带回到页面
        ActionContext.getContext().getValueStack().set("list", list);
        ActionContext.getContext().getValueStack().push(linkMan);
        
        return "editSuccess";
    }
```

1.1.2.3 编写Service

```
package com.admiral.crm.service.impl;

import java.util.List;

import org.hibernate.criterion.DetachedCriteria;
import org.springframework.transaction.annotation.Transactional;

import com.admiral.crm.dao.LinkManDao;
import com.admiral.crm.domain.LinkMan;
import com.admiral.crm.domain.PageBean;
import com.admiral.crm.service.LinkManService;

/**
 * 
 * @Description: 联系人Service层的实现类
 * @author Admiral
 */
@Transactional
public class LinkManServiceImpl implements LinkManService {

    //注入 Dao
    private LinkManDao linkManDao;

    public void setLinkManDao(LinkManDao linkManDao) {
        this.linkManDao = linkManDao;
    }

    @Override
    public PageBean<LinkMan> findAll(DetachedCriteria detachedCriteria, Integer
currPage, Integer pageSize) {
        PageBean<LinkMan> pageBean = new PageBean<LinkMan>();
        //设置当前页
        pageBean.setCurrPage(currPage);
        //设置每页的记录数
        pageBean.setPageSize(pageSize);

        //设置总记录数
        Integer totalCount = linkManDao.findCount(detachedCriteria);
        pageBean.setTotalCount(totalCount);

        //设置总页数
        Double num = Math.ceil(totalCount.doubleValue()/pageSize);
        pageBean.setTotalPage(num.intValue());

        //设置每页显示的数据
        Integer begin = (currPage-1) * pageSize;
        List<LinkMan> linkMans =
linkManDao.findByPage(detachedCriteria,begin,pageSize);
        pageBean.setList(linkMans);

        return pageBean;
    }

    @Override
    public void save(LinkMan linkMan) {
        linkManDao.save(linkMan);
    }

    @Override
    public LinkMan findById(Long lkm_id) {
        return linkManDao.findById(lkm_id);
    }

}
```

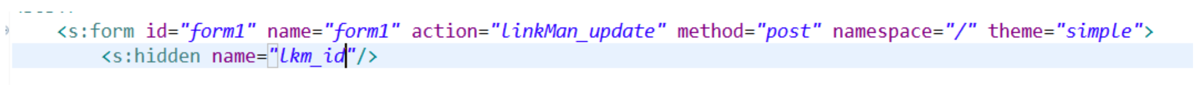
1.1.2.4 编写DAO

```
package com.admiral.crm.dao.impl;

import java.util.List;

import org.hibernate.criterion.DetachedCriteria;
import org.hibernate.criterion.Projections;
import org.springframework.orm.hibernate5.support.HibernateDaoSupport;
import com.admiral.crm.dao.LinkManDao;
import com.admiral.crm.domain.LinkMan;

/**
 *
 * @Description: 联系人Dao的实现类
 * @author Admiral
 */
public class LinkManDaoImpl extends HibernateDaoSupport implements LinkManDao {

    @Override
    public Integer findCount(DetachedCriteria detachedCriteria) {
        detachedCriteria.setProjection(Projections.rowCount());
        List<Long> list = (List<Long>)
this.getHibernateTemplate().findByCriteria(detachedCriteria);
        if(list!=null && list.size()>0) {
            return list.get(0).intValue();
        }
        return null;
    }

    @Override
    public List<LinkMan> findByPage(DetachedCriteria detachedCriteria, Integer
begin, Integer pageSize) {
        detachedCriteria.setProjection(null);
        return (List<LinkMan>)
this.getHibernateTemplate().findByCriteria(detachedCriteria, begin, pageSize);
    }

    @Override
    public void save(LinkMan linkMan) {
        this.getHibernateTemplate().save(linkMan);
    }

    @Override
    public LinkMan findById(Long lkm_id) {
        return this.getHibernateTemplate().get(LinkMan.class, lkm_id);
    }

}
```

- 配置跳转

```
        <!-- 配置联系人模块 -->
        <action name="linkMan_*" class="linkManAction" method="{1}">
            <result name="findAll">/jsp/linkman/list.jsp</result>
            <result name="saveUI">/jsp/linkman/add.jsp</result>
            <result name="saveSuccess"
type="redirectAction">linkMan_findAll.action</result>
            <result name="editSuccess">/jsp/linkman/edit.jsp</result>
        </action>
```

1.1.2.5 修改编辑页面提交的路径



```
    <s:form id="form1" name="form1" action="linkMan_update" method="post" namespace="/" theme="simple">
        <s:hidden name="lkm_id"/>
```

1.1.2.6 编写Action的update方法

```
    /**
     *
     * Title: update
     * Description: 联系人模块:修改联系人的方法
     * @return
     */
    public String update() {
        linkManService.update(linkMan);
        return "updateSuccess";
    }
```

1.1.2.7 编写Service

```
package com.admiral.crm.service.impl;

import java.util.List;

import org.hibernate.criterion.DetachedCriteria;
import org.springframework.transaction.annotation.Transactional;

import com.admiral.crm.dao.LinkManDao;
import com.admiral.crm.domain.LinkMan;
import com.admiral.crm.domain.PageBean;
import com.admiral.crm.service.LinkManService;

/**
 *
 * @Description: 联系人Service层的实现类
 * @author Admiral
 */
@Transactional
public class LinkManServiceImpl implements LinkManService {

    //注入 Dao
    private LinkManDao linkManDao;

    public void setLinkManDao(LinkManDao linkManDao) {
        this.linkManDao = linkManDao;
    }

    @Override
    public PageBean<LinkMan> findAll(DetachedCriteria detachedCriteria, Integer
currPage, Integer pageSize) {
        PageBean<LinkMan> pageBean = new PageBean<LinkMan>();
        //设置当前页
        pageBean.setCurrPage(currPage);
        //设置每页的记录数
        pageBean.setPageSize(pageSize);

        //设置总记录数
        Integer totalCount = linkManDao.findCount(detachedCriteria);
        pageBean.setTotalCount(totalCount);

        //设置总页数
        Double num = Math.ceil(totalCount.doubleValue()/pageSize);
        pageBean.setTotalPage(num.intValue());

        //设置每页显示的数据
        Integer begin = (currPage-1) * pageSize;
        List<LinkMan> linkMans =
linkManDao.findByPage(detachedCriteria,begin,pageSize);
        pageBean.setList(linkMans);

        return pageBean;
    }

    @Override
    public void save(LinkMan linkMan) {
        linkManDao.save(linkMan);
    }

    @Override
    public LinkMan findById(Long lkm_id) {
        return linkManDao.findById(lkm_id);
    }

    @Override
    public void update(LinkMan linkMan) {
        linkManDao.update(linkMan);
    }

}
```

1.1.2.8 编写DAO

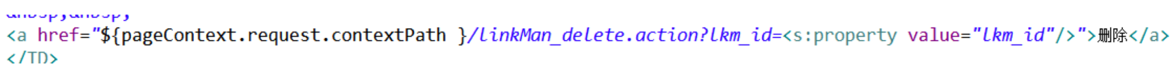
```
package com.admiral.crm.dao.impl;

import java.util.List;

import org.hibernate.criterion.DetachedCriteria;
import org.hibernate.criterion.Projections;
import org.springframework.orm.hibernate5.support.HibernateDaoSupport;
import com.admiral.crm.dao.LinkManDao;
import com.admiral.crm.domain.LinkMan;

/**
 *
 * @Description: 联系人Dao的实现类
 * @author Admiral
 */
public class LinkManDaoImpl extends HibernateDaoSupport implements LinkManDao {

    @Override
    public Integer findCount(DetachedCriteria detachedCriteria) {
        detachedCriteria.setProjection(Projections.rowCount());
        List<Long> list = (List<Long>)
this.getHibernateTemplate().findByCriteria(detachedCriteria);
        if(list!=null && list.size()>0) {
            return list.get(0).intValue();
        }
        return null;
    }

    @Override
    public List<LinkMan> findByPage(DetachedCriteria detachedCriteria, Integer
begin, Integer pageSize) {
        detachedCriteria.setProjection(null);
        return (List<LinkMan>)
this.getHibernateTemplate().findByCriteria(detachedCriteria, begin, pageSize);
    }

    @Override
    public void save(LinkMan linkMan) {
        this.getHibernateTemplate().save(linkMan);
    }

    @Override
    public LinkMan findById(Long lkm_id) {
        return this.getHibernateTemplate().get(LinkMan.class, lkm_id);
    }

    @Override
    public void update(LinkMan linkMan) {
        this.getHibernateTemplate().update(linkMan);
    }

}
```

- 配置跳转

```
        <!-- 配置联系人模块 -->
        <action name="linkMan_*" class="linkManAction" method="{1}">
            <result name="findAll">/jsp/linkman/list.jsp</result>
            <result name="saveUI">/jsp/linkman/add.jsp</result>
            <result name="saveSuccess"
type="redirectAction">linkMan_findAll.action</result>
            <result name="editSuccess">/jsp/linkman/edit.jsp</result>
            <result name="updateSuccess"
type="redirectAction">linkMan_findAll.action</result>
        </action>
```

1.1.3 删除联系人

1.1.3.1 修改链接地址



```
&nbsp;&nbsp;
<a href="${pageContext.request.contextPath }/linkMan_delete.action?lkm_id=<s:property value="lkm_id"/>">删除</a>
</TD>
```

1.1.3.2 编写Action中的delete方法

```
    /**
     *
     * Title: delete
     * Description: 联系人模块:删除联系人的方法
     * @return
     */
    public String delete() {
        //先查询再删除
        linkMan = linkManService.findById(linkMan.getLkm_id());
        //删除联系人
        linkManService.delete(linkMan);
        return "deleteSuccess";
    }
```

1.1.3.3 编写Service

```
package com.admiral.crm.service.impl;

import java.util.List;

import org.hibernate.criterion.DetachedCriteria;
import org.springframework.transaction.annotation.Transactional;

import com.admiral.crm.dao.LinkManDao;
import com.admiral.crm.domain.LinkMan;
import com.admiral.crm.domain.PageBean;
```

```java
import com.admiral.crm.service.LinkManService;

/**
 *
 * @Description: 联系人Service层的实现类
 * @author Admiral
 */
@Transactional
public class LinkManServiceImpl implements LinkManService {

    //注入 Dao
    private LinkManDao linkManDao;

    public void setLinkManDao(LinkManDao linkManDao) {
        this.linkManDao = linkManDao;
    }

    @Override
    public PageBean<LinkMan> findAll(DetachedCriteria detachedCriteria, Integer
currPage, Integer pageSize) {
        PageBean<LinkMan> pageBean = new PageBean<LinkMan>();
        //设置当前页
        pageBean.setCurrPage(currPage);
        //设置每页的记录数
        pageBean.setPageSize(pageSize);

        //设置总记录数
        Integer totalCount = linkManDao.findCount(detachedCriteria);
        pageBean.setTotalCount(totalCount);

        //设置总页数
        Double num = Math.ceil(totalCount.doubleValue()/pageSize);
        pageBean.setTotalPage(num.intValue());

        //设置每页显示的数据
        Integer begin = (currPage-1) * pageSize;
        List<LinkMan> linkMans =
linkManDao.findByPage(detachedCriteria,begin,pageSize);
        pageBean.setList(linkMans);

        return pageBean;
    }

    @Override
    public void save(LinkMan linkMan) {
        linkManDao.save(linkMan);
    }

    @Override
    public LinkMan findById(Long lkm_id) {
        return linkManDao.findById(lkm_id);
    }

    @Override
    public void update(LinkMan linkMan) {
        linkManDao.update(linkMan);
    }
    @Override
    public void delete(LinkMan linkMan) {
        linkManDao.delete(linkMan);
    }

}
```

1.1.3.4 编写DAO

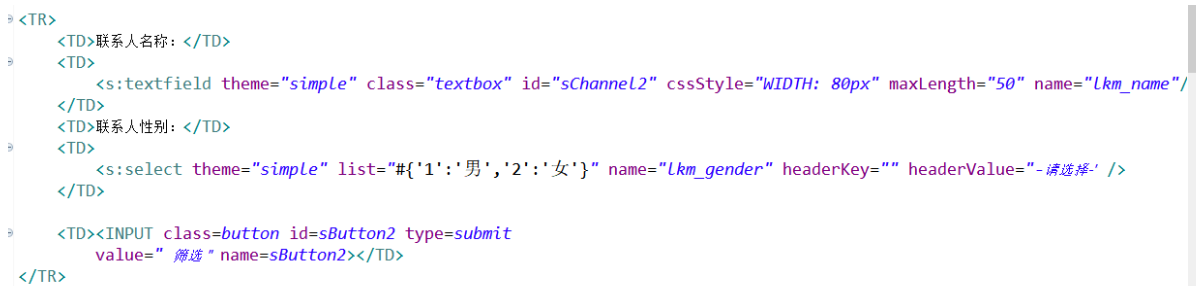
```
package com.admiral.crm.dao.impl;

import java.util.List;

import org.hibernate.criterion.DetachedCriteria;
import org.hibernate.criterion.Projections;
import org.springframework.orm.hibernate5.support.HibernateDaoSupport;

import com.admiral.crm.dao.LinkManDao;
import com.admiral.crm.domain.LinkMan;

/**
 *
 * @Description: 联系人Dao的实现类
 * @author Admiral
 */
public class LinkManDaoImpl extends HibernateDaoSupport implements LinkManDao {

    @Override
    public Integer findCount(DetachedCriteria detachedCriteria) {
        detachedCriteria.setProjection(Projections.rowCount());
        List<Long> list = (List<Long>)
this.getHibernateTemplate().findByCriteria(detachedCriteria);
        if(list!=null && list.size()>0) {
            return list.get(0).intValue();
        }
        return null;
    }

    @Override
    public List<LinkMan> findByPage(DetachedCriteria detachedCriteria, Integer
begin, Integer pageSize) {
        detachedCriteria.setProjection(null);
        return (List<LinkMan>)
this.getHibernateTemplate().findByCriteria(detachedCriteria, begin, pageSize);
    }

    @Override
    public void save(LinkMan linkMan) {
        this.getHibernateTemplate().save(linkMan);
    }

    @Override
    public LinkMan findById(Long lkm_id) {
        return this.getHibernateTemplate().get(LinkMan.class, lkm_id);
    }

    @Override
    public void update(LinkMan linkMan) {
        this.getHibernateTemplate().update(linkMan);
    }

    @Override
    public void delete(LinkMan linkMan) {
        this.getHibernateTemplate().delete(linkMan);
    }

}
```

- 配置跳转

```
        <!-- 配置联系人模块 -->
        <action name="linkMan_*" class="linkManAction" method="{1}">
            <result name="findAll">/jsp/linkman/list.jsp</result>
            <result name="saveUI">/jsp/linkman/add.jsp</result>
            <result name="saveSuccess"
type="redirectAction">linkMan_findAll.action</result>
            <result name="editSuccess">/jsp/linkman/edit.jsp</result>
            <result name="updateSuccess"
type="redirectAction">linkMan_findAll.action</result>
            <result name="deleteSuccess"
type="redirectAction">linkMan_findAll.action</result>
        </action>
```

1.1.4 条件查询联系人

1.1.4.1 修改列表页面：



```
<TR>
    <TD>联系人名称：</TD>
    <TD>
        <s:textfield theme="simple" class="textbox" id="sChannel2" cssStyle="WIDTH: 80px" maxLength="50" name="lkm_name"/
    </TD>
    <TD>联系人性别：</TD>
    <TD>
        <s:select theme="simple" list="#{'1':'男','2':'女'}" name="lkm_gender" headerKey="" headerValue="-请选择-' />
    </TD>

    <TD><INPUT class=button id=sButton2 type=submit
        value=" 筛选 " name=sButton2></TD>
</TR>
```

1.1.4.2 修改Action中findAll的方法

```
    /**
     *
     * Title: findAll
     * Description: 待条件分页查询联系人
     * @return
     */
    public String findAll() {
        //创建离线条件查询
        DetachedCriteria detachedCriteria =
DetachedCriteria.forClass(LinkMan.class);
        //设置条件
        if(linkMan.getLkm_name()!=null) {
            //设置按名称查询的条件
            detachedCriteria.add(Restrictions.like("lkm_name", "%" +
linkMan.getLkm_name()+ "%"));
        }
        if(linkMan.getLkm_gender()!=null && !"".equals(linkMan.getLkm_gender()))
{
            //设置按性别查询的条件
            detachedCriteria.add(Restrictions.eq("lkm_gender",
linkMan.getLkm_gender()));
        }

        //调用业务层
        PageBean<LinkMan> pageBean =
linkManService.findAll(detachedCriteria,currPage,pageSize);
        //将 pageBean 压入值栈中
        ActionContext.getContext().getValueStack().push(pageBean);
        return "findAll";
    }
```

1.1.5 解决与客户之间问题

1.1.5.1 级联删除的问题

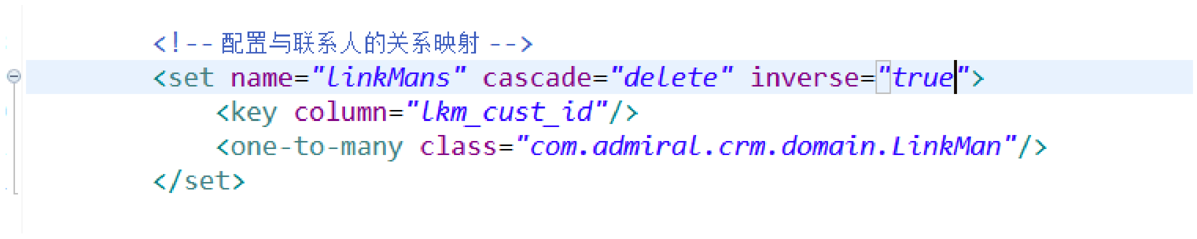
- 级联删除，在客户删除的时候，先查询再删除
- 在客户的映射删进行配置

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE hibernate-mapping PUBLIC
    "-//Hibernate/Hibernate Mapping DTD 3.0//EN"
    "http://www.hibernate.org/dtd/hibernate-mapping-3.0.dtd">

<hibernate-mapping>
    <class name="com.admiral.crm.domain.Customer" table="cst_customer">
```

```
        <!--建立类中的属性与表中的主键的映射关系 -->
        <id name="cust_id" column="cust_id">
            <!--主键的生成策略 -->
            <generator class="native"></generator>
        </id>

        <!-- 建立类中的普通属性与表中的字段的映射 -->
        <property name="cust_name" column="cust_name" />
        <!-- <property name="cust_source" column="cust_source" />
        <property name="cust_industry" column="cust_industry" />
        <property name="cust_level" column="cust_level" /> -->
        <property name="cust_phone" column="cust_phone" />
        <property name="cust_mobile" column="cust_mobile" />
        <property name="cust_image" column="cust_image" />

        <!-- 配置多对一关系 -->
        <many-to-one name="baseDictSource"
class="com.admiral.crm.domain.BaseDict" column="cust_source" />
        <many-to-one name="baseDictIndustry"
class="com.admiral.crm.domain.BaseDict" column="cust_industry" />
        <many-to-one name="baseDictLevel"
class="com.admiral.crm.domain.BaseDict" column="cust_level" />

        <!-- 配置与联系人的关系映射 -->
        <set name="linkMans" cascade="delete">
            <key column="lkm_cust_id"/>
            <one-to-many class="com.admiral.crm.domain.LinkMan"/>
        </set>

    </class>
</hibernate-mapping>
```

1.1.5.2 修改客户的时候，联系人的客户的信息就丢失了

- 因为在修改客户的时候，没有查询联系人的集合，当点击修改，修改客户（修改客户所关联联系人）因为联系人的集合是空，所以将外键置为null。



```
        <!-- 配置与联系人的关系映射 -->
        <set name="linkMans" cascade="delete" inverse="true">
            <key column="lkm_cust_id"/>
            <one-to-many class="com.admiral.crm.domain.LinkMan"/>
        </set>
```

1.2 CRM综合练习_抽取通用的DAO

1.2.1 通用的DAO的抽取

1.2.1.1 抽取通用的增删改的操作

- 定义 BaseDao 接口

```java
package com.admiral.crm.dao;

/**
 *
 * @Description: 通用的 Dao 接口
 * @author Admiral
 */
public interface BaseDao<T> {

    public void save(T t);
    public void update(T t);
    public void delete(T t);
}
```

- 定义 BaseDao 实现类

```java
package com.admiral.crm.dao.impl;

import org.springframework.orm.hibernate5.support.HibernateDaoSupport;

import com.admiral.crm.dao.BaseDao;

public class BaseDaoImpl<T> extends HibernateDaoSupport implements BaseDao<T>{

    @Override
    public void save(T t) {
        this.getHibernateTemplate().save(t);
    }

    @Override
    public void update(T t) {
        this.getHibernateTemplate().update(t);
    }

    @Override
    public void delete(T t) {
        this.getHibernateTemplate().delete(t);
    }

}
```

- 修改 CustomerDao

```java
package com.admiral.crm.dao;

import java.util.List;

import org.hibernate.criterion.DetachedCriteria;

import com.admiral.crm.domain.Customer;

/**
 * 客户管理的Dao接口
 */
public interface CustomerDao extends BaseDao<Customer>{

    /**
     *
     * Title: findCount
     * Description: 客户管理:Dao层带条件查询记录数
     * @param detachedCriteria
     * @return
     */
    public Integer findCount(DetachedCriteria detachedCriteria);

    /**
     *
     * Title: findByPage
     * Description: 客户管理:Dao层待条件分页查询
     * @param detachedCriteria :分页查询的条件
     * @param begin :分页查询的开始索引
     * @param pageSize :分页查询每页显示的记录数
     * @return
     */
    public List<Customer> findByPage(DetachedCriteria detachedCriteria, Integer
begin, Integer pageSize);

    /**
     *
     * Title: findById
     * Description: 客户管理:Dao层根据ID查询客户信息
     * @param cust_id : 要查询的客户的ID
     * @return : 查询到的客户信息
     */
    public Customer findById(Long cust_id);


    /**
     *
     * Title: findAll
     * Description: 客户管理:Dao层查询所有客户
     * @return
     */
    public List<Customer> findAll();

}
```

- 修改 CustomerDaoImpl

```java
package com.admiral.crm.dao.impl;

import java.util.List;

import org.hibernate.criterion.DetachedCriteria;
import org.hibernate.criterion.Projections;
import org.springframework.orm.hibernate5.support.HibernateDaoSupport;

import com.admiral.crm.dao.CustomerDao;
import com.admiral.crm.domain.Customer;

/**
 * 客户管理的Dao实现类
 */
public class CustomerDaoImpl extends BaseDaoImpl<Customer> implements
CustomerDao {


    @Override
    public Integer findCount(DetachedCriteria detachedCriteria) {
        //
        detachedCriteria.setProjection(Projections.rowCount());
        List<Long> list = (List<Long>)
this.getHibernateTemplate().findByCriteria(detachedCriteria);
        if(list!=null && list.size()>0) {
            return list.get(0).intValue();
        }
        return null;
    }

    @Override
    public List<Customer> findByPage(DetachedCriteria detachedCriteria, Integer
begin, Integer pageSize) {
        detachedCriteria.setProjection(null);
        return (List<Customer>)
this.getHibernateTemplate().findByCriteria(detachedCriteria, begin, pageSize);
    }

    @Override
    public Customer findById(Long cust_id) {
        return this.getHibernateTemplate().get(Customer.class, cust_id);
    }

    @Override
    public List<Customer> findAll() {
        return (List<Customer>) this.getHibernateTemplate().find("from
Customer");
    }

}
```

- 修改 LinkManDao

```java
package com.admiral.crm.dao;

import java.util.List;

import org.hibernate.criterion.DetachedCriteria;

import com.admiral.crm.domain.LinkMan;

/**
 *
 * @Description: 联系人Dao的接口
 * @author Admiral
 */
public interface LinkManDao extends BaseDao<LinkMan>{

    /**
     *
     * Title: findCount
     * Description: 联系人模块:Dao层根据添加查询记录数
     * @param detachedCriteria
     * @return
     */
    public Integer findCount(DetachedCriteria detachedCriteria);

    /**
     *
     * Title: findByPage
     * Description: 联系人模块:Dao层根据添加分页查询
     * @param detachedCriteria:离线的条件查询
     * @param begin : 开始索引
     * @param pageSize : 每页的记录数
     * @return
     */
    public List<LinkMan> findByPage(DetachedCriteria detachedCriteria, Integer
begin, Integer pageSize);


    /**
     *
     * Title: findById
     * Description: 联系人模块:Dao层根据ID查询联系人信息
     * @param lkm_id
     * @return
     */
    public LinkMan findById(Long lkm_id);

}
```

- 修改 LinkManDaoImpl

```java
package com.admiral.crm.dao.impl;

import java.util.List;

import org.hibernate.criterion.DetachedCriteria;
import org.hibernate.criterion.Projections;
import org.springframework.orm.hibernate5.support.HibernateDaoSupport;

import com.admiral.crm.dao.LinkManDao;
import com.admiral.crm.domain.LinkMan;

/**
 *
 * @Description: 联系人Dao的实现类
 * @author Admiral
 */
public class LinkManDaoImpl extends BaseDaoImpl<LinkMan> implements LinkManDao {

    @Override
    public Integer findCount(DetachedCriteria detachedCriteria) {
        detachedCriteria.setProjection(Projections.rowCount());
        List<Long> list = (List<Long>)
this.getHibernateTemplate().findByCriteria(detachedCriteria);
        if(list!=null && list.size()>0) {
            return list.get(0).intValue();
        }
        return null;
    }

    @Override
    public List<LinkMan> findByPage(DetachedCriteria detachedCriteria, Integer
begin, Integer pageSize) {
        detachedCriteria.setProjection(null);
        return (List<LinkMan>)
this.getHibernateTemplate().findByCriteria(detachedCriteria, begin, pageSize);
    }


    @Override
    public LinkMan findById(Long lkm_id) {
        return this.getHibernateTemplate().get(LinkMan.class, lkm_id);
    }

}
```

1.2.1.2 抽取一个查询一个的方法

- BaseDao

```
package com.admiral.crm.dao;

import java.io.Serializable;

/**
 *
 * @Description: 通用的 Dao 接口
 * @author Admiral
 */
public interface BaseDao<T> {

    public void save(T t);
    public void update(T t);
    public void delete(T t);

    //根据 ID 查询单个的方法
    public T findById(Serializable id);
}
```

1.2.1.3 解决方案一：在实现类的构造方法中传入一个Class

- BaseDaoImpl

```
package com.admiral.crm.dao.impl;

import java.io.Serializable;

import org.springframework.orm.hibernate5.support.HibernateDaoSupport;

import com.admiral.crm.dao.BaseDao;

public class BaseDaoImpl<T> extends HibernateDaoSupport implements BaseDao<T>{

    private Class clazz;

    public BaseDaoImpl(Class clazz) {
        this.clazz = clazz;
    }

    @Override
    public void save(T t) {
        this.getHibernateTemplate().save(t);
    }

    @Override
    public void update(T t) {
        this.getHibernateTemplate().update(t);
    }
    @Override
    public void delete(T t) {
        this.getHibernateTemplate().delete(t);
    }

    @Override
    public T findById(Serializable id) {
        return (T) this.getHibernateTemplate().get(clazz, id);
    }

}
```

- 完善 BaseDao 接口

```
package com.admiral.crm.dao;

import java.io.Serializable;
import java.util.List;

import org.hibernate.criterion.DetachedCriteria;

/**
 *
 * @Description: 通用的 Dao 接口
 * @author Admiral
 */
public interface BaseDao<T> {

    public void save(T t);
    public void update(T t);
    public void delete(T t);

    //根据 ID 查询单个的方法
    public T findById(Serializable id);

    //查询所有
    public List<T> findAll();

    //统计查询
    public Integer findCount(DetachedCriteria detachedCriteria);

    //根据条件分页查询
    public List<T> findByPage(DetachedCriteria detachedCriteria,Integer
begin,Integer pageSize);
}
```

- BaseDaoImpl

```java
package com.admiral.crm.dao.impl;

import java.io.Serializable;
import java.lang.reflect.ParameterizedType;
import java.lang.reflect.Type;
import java.util.List;

import org.hibernate.criterion.DetachedCriteria;
import org.hibernate.criterion.Projections;
import org.springframework.orm.hibernate5.support.HibernateDaoSupport;

import com.admiral.crm.dao.BaseDao;

public class BaseDaoImpl<T> extends HibernateDaoSupport implements BaseDao<T>{

    private Class clazz;

    public BaseDaoImpl() {
        //反射:第一步,需要获取到 Class
        Class clazz = this.getClass();//正在被调用的那个类的 class(子类)

        //查看 JKD 的 API
        Type type = clazz.getGenericSuperclass();//获取参数化类型
        System.out.println(type);

        //得到的这个 type 就是一个参数化类型,将 type 强转为参数化类型
        ParameterizedType pType = (ParameterizedType) type;

        //通过参数化类型获得实际类型参数:得到一个实例类型参数的数组?Map<String,Integer>
        Type[] types = pType.getActualTypeArguments();

        //只获得第一个实际类型参数即可
        this.clazz = (Class) types[0];
    }

    @Override
    public void save(T t) {
        this.getHibernateTemplate().save(t);
    }

    @Override
    public void update(T t) {
        this.getHibernateTemplate().update(t);
    }

    @Override
    public void delete(T t) {
        this.getHibernateTemplate().delete(t);
    }

    @Override
    public T findById(Serializable id) {
        return (T) this.getHibernateTemplate().get(clazz, id);
    }

    @Override
    public List<T> findAll() {
        return (List<T>) this.getHibernateTemplate().find("from " +
clazz.getSimpleName());
    }

    @Override
    public Integer findCount(DetachedCriteria detachedCriteria) {
        detachedCriteria.setProjection(Projections.rowCount());
        List<Long> list = (List<Long>)
this.getHibernateTemplate().findByCriteria(detachedCriteria);
        if(list!=null && list.size()>0) {
            return list.get(0).intValue();
        }
        return null;
    }

    @Override
    public List<T> findByPage(DetachedCriteria detachedCriteria, Integer begin,
Integer pageSize) {
        detachedCriteria.setProjection(null);
        return (List<T>)
this.getHibernateTemplate().findByCriteria(detachedCriteria, begin, pageSize);
    }

}
```

- 修改各种 Dao

- CustomerDao

```
package com.admiral.crm.dao;

import com.admiral.crm.domain.Customer;

/**
 * 客户管理的Dao接口
 */
public interface CustomerDao extends BaseDao<Customer>{



}
```

- CustomerDaoImpl

```
package com.admiral.crm.dao.impl;

import com.admiral.crm.dao.CustomerDao;
import com.admiral.crm.domain.Customer;

/**
 * 客户管理的Dao实现类
 */
```

```java
public class CustomerDaoImpl extends BaseDaoImpl<Customer> implements
CustomerDao {

    public CustomerDaoImpl() {
        super(Customer.class);
    }

}
```

- LinkManDao

```java
package com.admiral.crm.dao;

import com.admiral.crm.domain.LinkMan;

/**
 *
 * @Description: 联系人Dao的接口
 * @author Admiral
 */
public interface LinkManDao extends BaseDao<LinkMan>{

}
```

- LinkManDaoImpl

```java
package com.admiral.crm.dao.impl;

import java.util.List;

import org.hibernate.criterion.DetachedCriteria;
import org.hibernate.criterion.Projections;
import org.springframework.orm.hibernate5.support.HibernateDaoSupport;

import com.admiral.crm.dao.LinkManDao;
import com.admiral.crm.domain.LinkMan;

/**
 *
 * @Description: 联系人Dao的实现类
 * @author Admiral
 */
public class LinkManDaoImpl extends BaseDaoImpl<LinkMan> implements LinkManDao {

    public LinkManDaoImpl() {
        super(LinkMan.class);
    }
}
```

- BaseDictDao

```
package com.admiral.crm.dao;

import java.util.List;

import com.admiral.crm.domain.BaseDict;

/**
 *
 * @Description: 字典Dao的接口
 * @author Admiral
 */
public interface BaseDictDao extends BaseDao<BaseDict>{

    /**
     *
     * Title: findByTypeCode
     * Description: Dao 层根据类型名称查询字典
     * @param dict_type_code
     * @return
     */
    public List<BaseDict> findByTypeCode(String dict_type_code);

}
```

- BaseDictDaoImpl

```
package com.admiral.crm.dao.impl;

import java.util.List;

import com.admiral.crm.dao.BaseDictDao;
import com.admiral.crm.domain.BaseDict;
/**
 *
 * @Description: 字典Dao的实现类
 * @author Admiral
 */
public class BaseDictDaoImpl extends BaseDaoImpl<BaseDict> implements
BaseDictDao {

    public BaseDictDaoImpl() {
        super(BaseDict.class);
    }

    @Override
    public List<BaseDict> findByTypeCode(String dict_type_code) {
        return (List<BaseDict>) this.getHibernateTemplate().find("from BaseDict
where dict_type_code = ?", dict_type_code);
    }

}
```

- UserDao

```
package com.admiral.crm.dao;

import com.admiral.crm.domain.User;

/**
 * 用户管理的 Dao 接口
 */
public interface UserDao extends BaseDao<User>{


    /**
     *
     * Title: login
     * Description: Dao 层用户登录的方法
     * @param user
     * @return
     */
    public User login(User user);

}
```

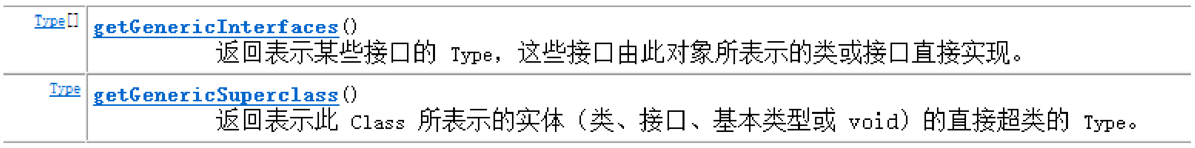
- UserDaoImpl

```
package com.admiral.crm.dao.impl;

import java.util.List;

import com.admiral.crm.dao.UserDao;
import com.admiral.crm.domain.User;

public class UserDaoImpl extends BaseDaoImpl<User> implements UserDao {

    public UserDaoImpl() {
        super(User.class);
    }

    @Override
    public User login(User user) {
        List<User> list = (List<User>) this.getHibernateTemplate().find("from
User where user_code = ? and user_password = ?", user.getUser_code(),
                user.getUser_password());
        if(list!=null && list.size() > 0) {
            return list.get(0);
        }
        return null;
    }

}
```

1.2.1.4 解决方案二：通过泛型的反射抽取通用的DAO

- 如果现在将DAO中的构造方法去掉，将父类的通用的DAO中提供无参数的构造即可，但是需要在无参数的构造中需要获得具体类型的Class才可以-----涉及到泛型的反射了。
- 泛型：
  - 泛型　：通用的类型。
  - <>　　：念为 typeof
  - List ：E称为类型参数变量
  - ArrayList　：Integer称为是实际类型参数
  - ArrayList　：ArrayList称为参数化类型

- 需要做的时候在父类的构造方法中获得子类继承父类上的参数化类型中的实际类型参数
- 泛型反射的步骤：
  - 第一步：获得代表子类对象的Class
  - 第二步：查看API



Type[]	getGenericInterfaces() 返回表示某些接口的 Type，这些接口由此对象所表示的类或接口直接实现。
Type	getGenericSuperclass() 返回表示此 Class 所表示的实体（类、接口、基本类型或 void）的直接超类的 Type。

-
  - Type[] getGenericInterfaces();　:获得带有泛型的接口，可以实现多个接口。
  - Type getGenericSuperclass();　　:获得带有泛型的父类，继承一个类。

- 第三步：获得带有泛型的父类
- 第四步：将带有泛型的父类的类型转成具体参数化的类型
- 第五步：通过参数化类型的方法获得实际类型参数
- 代码实现