1.1 CRM综合练习

1.1.1 搭建开发环境

1.1.1.1 创建 Web 项目,引入相关 jar 包

1.1.1.2 引入相关配置文件

- Struts2
  - web.xml
  - struts.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID"
version="2.5">
  <display-name>crm</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>

  <!-- 配置 Struts2 核心过滤器 -->
  <filter>
    <filter-name>struts2</filter-name>
    <filter-
class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter<
/filter-class>
  </filter>

  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

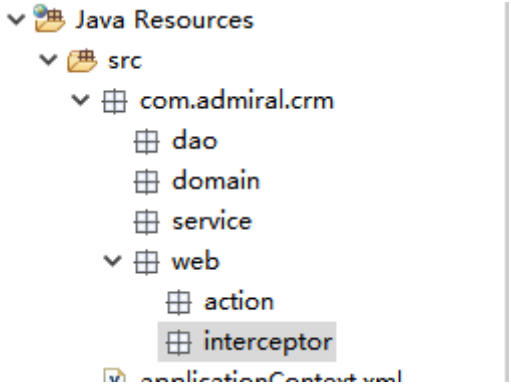
- Spring
  - applicaitonContext.xml
  - jdbc.properties
  - log4j.properties
  - web.xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

```
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID"
version="2.5">
  <display-name>crm</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>

  <!-- 配置 Spring 的核心监听器 -->
  <listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:accpliactionContext.xml</param-value>

  </context-param>

  <!-- 配置 Struts2 核心过滤器 -->
  <filter>
    <filter-name>struts2</filter-name>
    <filter-
class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter<
/filter-class>
  </filter>

  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

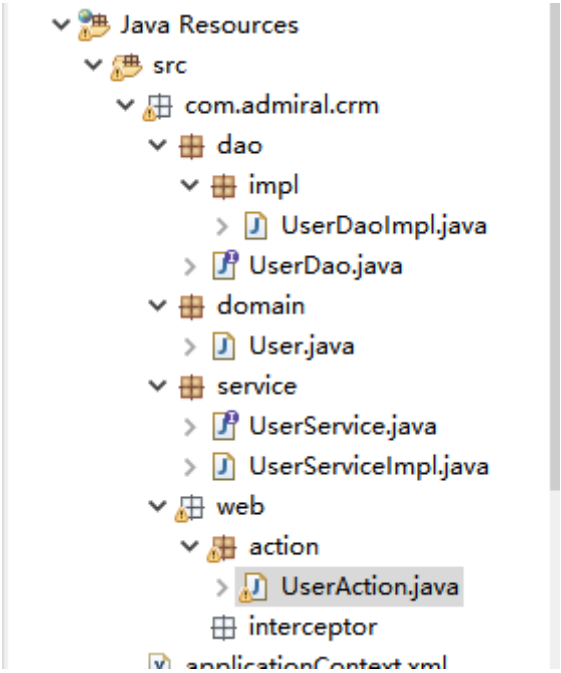
- Hibernate
  - hibernate.cfg.xml --交给 Spring 管理

1.1.1.3 创建相关包结构



1.1.1.4 引入相关页面

1.1.1.5 创建相关的类



1.1.1.6 将相关类交给 Spring 管理

applicationContext.xml

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:context="http://www.springframework.org/schema/context"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:tx="http://www.springframework.org/schema/tx"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
    http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context.xsd
    http://www.springframework.org/schema/aop
    http://www.springframework.org/schema/aop/spring-aop.xsd
    http://www.springframework.org/schema/tx
    http://www.springframework.org/schema/tx/spring-tx.xsd">


    <!-- ==============   Spring 整合 Hibernate   ============= -->

    <!-- 引入外部属性文件 -->
    <context:property-placeholder location="classpath:jdbc.properties"/>

    <!-- 配置 C3P0 连接池 -->
    <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource">
        <property name="driverClass" value="${jdbc.driverClass}" />
        <property name="jdbcUrl" value="${jdbc.url}" />
        <property name="user" value="${jdbc.username}" />
        <property name="password" value="${jdbc.password}" />
    </bean>
```

```
    <!-- 配置 Action -->
    <bean id="userAction" class="com.admiral.crm.web.action.UserAction">

    </bean>
    <!-- 配置 Service -->
    <bean id="userService" class="com.admiral.crm.service.impl.UserServiceImpl">

    </bean>
    <!-- 配置 Dao -->
    <bean id="userDao" class="com.admiral.crm.dao.impl.UserDaoImpl">

    </bean>

</beans>
```

1.1.2 用户模块--用户注册功能实现

1.1.2.0 添加注册页面

1.1.2.1 跳转到注册页面

- 在 login.jsp 页面添加注册按钮和事件

```
      <TD><INPUT id=btn
        style="BORDER-TOP-WIDTH: 0px; BORDER-LEFT-WIDTH: 0px; BORDER-BOTTOM-WIDTH:
        onclick='javascript:WebForm_DoPostBackWithOptions(new WebForm_PostBackOpti
        type=image src="images/login_button.gif" name=btn>
        <input type="button" value="注册" onclick="registUI()">
    </TD></TR></TBODY></TABLE></TD></TR></TBODY></TABLE></TD></TR>

<IMG src="images/login_3.jpg"
```

- 添加注册跳转事件

```
}
</STYLE>

<script type="text/javascript">
    function registUI() {
        window.location = "${ pageContext.request.contextPath }/regist.jsp"
    }
</script>
```

1.1.2.2 注册代码实现

1.1.2.2.1 提供一个注册页面

1.1.2.2.2. 创建表

```
CREATE TABLE `sys_user` (
  `user_id` bigint(32) NOT NULL AUTO_INCREMENT COMMENT '用户id',
  `user_code` varchar(32) NOT NULL COMMENT '用户账号',
  `user_name` varchar(64) NOT NULL COMMENT '用户名称',
  `user_password` varchar(32) NOT NULL COMMENT '用户密码',
  `user_state` char(1) NOT NULL COMMENT '1:正常,0:暂停',
  PRIMARY KEY (`user_id`)
) ENGINE=InnoDB AUTO_INCREMENT=9 DEFAULT CHARSET=utf8;
```

1.1.2.2.3 创建实体和映射

User.java

```
package com.admiral.crm.domain;

/**
 * 用户实体
 *
 */
public class User {
    private Long user_id;
    private String user_code;
    private String user_name;
    private String user_password;
    private String user_state;

    public Long getUser_id() {
        return user_id;
    }

    public void setUser_id(Long user_id) {
        this.user_id = user_id;
    }

    public String getUser_code() {
        return user_code;
    }

    public void setUser_code(String user_code) {
        this.user_code = user_code;
    }

    public String getUser_name() {
        return user_name;
```

```
    }

    public void setUser_name(String user_name) {
        this.user_name = user_name;
    }

    public String getUser_password() {
        return user_password;
    }

    public void setUser_password(String user_password) {
        this.user_password = user_password;
    }

    public String getUser_state() {
        return user_state;
    }

    public void setUser_state(String user_state) {
        this.user_state = user_state;
    }

    @Override
    public String toString() {
        return "User [user_id=" + user_id + ", user_code=" + user_code + ",
user_name=" + user_name + ", user_password="
                + user_password + ", user_state=" + user_state + "]";
    }

}
```

User.hbm.xml

```
<?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.User" table="sys_user">
        <id name="user_id" column="user_id">
            <generator class="natevie" />
        </id>

        <property name="user_code" column="user_code" />
        <property name="user_password" column="user_password" />
        <property name="user_name" column="user_name" />
        <property name="user_state" column="user_state" />
    </class>
</hibernate-mapping>
```

1.1.2.2.4 编写 Action

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

import com.admiral.crm.domain.User;
import com.admiral.crm.service.UserService;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

public class UserAction extends ActionSupport implements ModelDriven<User> {

    private User user = new User();

    @Override
    public User getModel() {
        return user;
    }

    // 注入 UserService
    private UserService userService;

    public void setUserService(UserService userService) {
        this.userService = userService;
    }

    /**
     * 注册的方法:regist
     */
    public String regist() {
        userService.regist(user);
        return NONE;
    }

}
```

1.1.2.2.5 编写 Service 并注入

UserServieImpl.java

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

import com.admiral.crm.dao.UserDao;
import com.admiral.crm.domain.User;
import com.admiral.crm.utils.MD5Utils;

public class UserServiceImpl implements UserService {

    // 注入 UserDao
    private UserDao userDao;
```

```
    public void setUserDao(UserDao userDao) {
        this.userDao = userDao;
    }

    @Override
    public void regist(User user) {

        //对密码进行加密处理
        user.setUser_password(MD5Utils.md5(user.getUser_password()));
        //设置用户状态
        user.setUser_state("1");
        userDao.save(user);
    }

}
```

```
    <!-- 配置 Action -->
    <bean id="userAction" class="com.admiral.crm.web.action.UserAction"
scope="prototype">
        <property name="userService" ref="userService" />
    </bean>
```

1.1.2.2.6 编写 Dao 并注入

- 整合 Hibernate 配置文件

applicationContext.xml

```
    <!-- 配置 Hibernate SessionFactory 相关配置 -->
    <bean id="sessionFactory"
class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">
        <!-- 注入连接池 -->
        <property name="dataSource" ref="dataSource" />
        <!-- 注入Hibernate 相关配置 -->
        <property name="hibernateProperties">
            <props>
                <prop
key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
                <prop key="hibernate.show_sql">true</prop>
                <prop key="hibernate.format_sql">true</prop>
                <prop key="hibernate.hbm2ddl.auto">update</prop>
            </props>
        </property>
        <!-- 引入映射文件 -->
        <property name="mappingResources">
            <list>
                <value>com/admiral/crm/domain/User.hbm.xml</value>
            </list>
        </property>
```

```
    </bean>
```

- 改写 Dao 继承 HibernateDaoSupport

- 在 Dao 中注入 SessionFactory

```
    <!-- 配置 Dao -->
    <bean id="userDao" class="com.admiral.crm.dao.impl.UserDaoImpl">
        <property name="sessionFactory" ref="sessionFactory" />
    </bean>
```

UserDaoImpl.java

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

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

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

public class UserDaoImpl extends HibernateDaoSupport implements UserDao {


    @Override
    public void save(User user) {
        this.getHibernateTemplate().save(user);
    }

}
```

- 在 Service 中注入 Dao

```
    <!-- 配置 Service -->
    <bean id="userService" class="com.admiral.crm.service.impl.UserServiceImpl">
        <property name="userDao" ref="userDao" />
    </bean>
```

1.1.2.2.7 配置 Spring 事务管理

- 配置平台事务管理器
- 开启注解事务

```
    <!-- 配置平台事务管理器 -->
    <bean id="transactionManager"
class="org.springframework.orm.hibernate5.HibernateTransactionManager">
        <property name="sessionFactorye" ref="sessionFactory" />
    </bean>

    <!-- 开启注解事务 -->
    <tx:annotation-driven transaction-manager="transactionManager"/>
```

- 在 Service 层使用注解事务

```
@Transactional
public class UserServiceImpl implements UserService {}
```

1.1.2.2.8 配置跳转

- 改写 UserAction

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

import com.admiral.crm.domain.User;
import com.admiral.crm.service.UserService;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

public class UserAction extends ActionSupport implements ModelDriven<User> {

    private User user = new User();

    @Override
    public User getModel() {
        return user;
    }

    // 注入 UserService
    private UserService userService;

    public void setUserService(UserService userService) {
        this.userService = userService;
    }

    /**
     * 注册的方法:regist
     */
    public String regist() {
        userService.regist(user);
        return LOGIN;
    }
```

```
}
```

- 配置 Struts2

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
    "http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>
    <package name="crm" extends="struts-default" namespace="/">

        <action name="user_*" class="userAction" method="{1}">
            <result name="login">/login.jsp</result>
        </action>

    </package>
</struts>
```

1.1.2.2.9 测试

1.2 CRM 综合练习:用户模块登陆功能

1.2.1用户模块：登录功能代码实现

1.2.1.1 修改登录页面

```
<FORM id=form1 name=form1 action="${ pageContext.request.contextPath }/user_login.action" method=post target="_parent">


          <TR>
            <TD style="HEIGHT: 28px" width=80>登 录 名：</TD>
            <TD style="HEIGHT: 28px" width=150><INPUT id=txtName
              style="WIDTH: 130px" name="user_code"></TD>
            <TD style="HEIGHT: 28px" width=370><SPAN
              id=RequiredFieldValidator3
              style="FONT-WEIGHT: bold; VISIBILITY: hidden; COLOR: white">请输入登录名</SPAN></TD></TR>
          <TR>
            <TD style="HEIGHT: 28px">登录密码：</TD>
            <TD style="HEIGHT: 28px"><INPUT id=txtPwd style="WIDTH: 130px"
              type=password name="user_password"></TD>
            <TD style="HEIGHT: 28px"><SPAN id=RequiredFieldValidator4
              style="FONT-WEIGHT: bold; VISIBILITY: hidden; COLOR: white">请输入密码</SPAN></TD></TR>
          <TR>
            <TD style="HEIGHT: 28px">验证码：</TD>
            <TD style="HEIGHT: 28px"><INPUT id=txtcode
              style="WIDTH: 130px" name=txtcode></TD>
            <TD style="HEIGHT: 28px"> </TD></TR>
          <TR>
            <TD style="HEIGHT: 18px"></TD>
            <TD style="HEIGHT: 18px"></TD>
```

1.2.1.2 编写 Action 中 login 方法

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

import org.apache.struts2.ServletActionContext;

import com.admiral.crm.domain.User;
import com.admiral.crm.service.UserService;
import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

public class UserAction extends ActionSupport implements ModelDriven<User> {

    private User user = new User();

    @Override
    public User getModel() {
        return user;
    }

    // 注入 UserService
    private UserService userService;

    public void setUserService(UserService userService) {
        this.userService = userService;
    }

    /**
     * 注册的方法:regist
     */
    public String regist() {
        userService.regist(user);
        return LOGIN;
    }

    /**
     * 用户登录的方法
     * Title: login
     * Description:
     * @return
     */
    public String login() {

        User existUser = userService.login(user);
        if (existUser != null) {
            //登录成功
            //将用户信息保存到 session 中
//
ServletActionContext.getRequest().getSession().setAttribute("existUser",
existUser);
            ActionContext.getContext().getSession().put("existUser", existUser);
            return SUCCESS;
        }else {
            //登录失败
            //保存错误信息
            this.addActionError("用户名或密码错!");
```

```
            return LOGIN;
        }
    }

}
```

1.2.1.3 编写Service

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

import org.springframework.transaction.annotation.Transactional;

import com.admiral.crm.dao.UserDao;
import com.admiral.crm.domain.User;
import com.admiral.crm.service.UserService;
import com.admiral.crm.utils.MD5Utils;

@Transactional
public class UserServiceImpl implements UserService {

    // 注入 UserDao
    private UserDao userDao;

    public void setUserDao(UserDao userDao) {
        this.userDao = userDao;
    }

    @Override
    public void regist(User user) {

        //对密码进行加密处理
        user.setUser_password(MD5Utils.md5(user.getUser_password()));
        //设置用户状态
        user.setUser_state("1");
        userDao.save(user);
    }

    @Override
    public User login(User user) {
        //对密码进行加密
        user.setUser_password(MD5Utils.md5(user.getUser_password()));

        return userDao.login(user);
    }

}
```

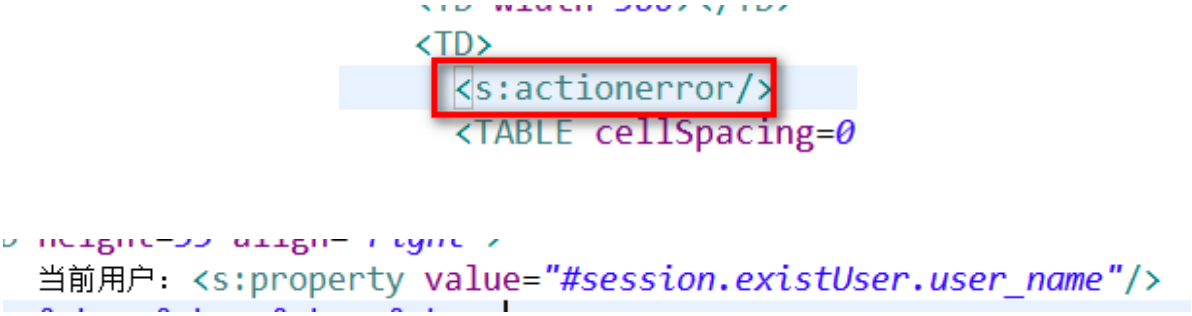
1.2.1.4 编写DAO

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

import java.util.List;

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

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

public class UserDaoImpl extends HibernateDaoSupport implements UserDao {

    @Override
    public void save(User user) {
        this.getHibernateTemplate().save(user);
    }

    @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;
    }

}
```

1.2.1.5 配置页面的跳转

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
    "http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>
    <package name="crm" extends="struts-default" namespace="/">

        <action name="user_*" class="userAction" method="{1}">
            <result name="login">/login.jsp</result>
            <result name="success" type="redirect">/index.jsp</result>
        </action>

    </package>
</struts>
```

1.2.1.6 在页面中显示数据



1.3 CRM 综合练习:客户管理模块保存客户

1.3.1 客户管理：准备工作

1.3.1.1 创建表

```
CREATE TABLE `cst_customer` (
  `cust_id` bigint(32) NOT NULL AUTO_INCREMENT COMMENT '客户编号(主键)',
  `cust_name` varchar(32) NOT NULL COMMENT '客户名称(公司名称)',
  `cust_source` varchar(32) DEFAULT NULL COMMENT '客户信息来源',
  `cust_industry` varchar(32) DEFAULT NULL COMMENT '客户所属行业',
  `cust_level` varchar(32) DEFAULT NULL COMMENT '客户级别',
  `cust_phone` varchar(64) DEFAULT NULL COMMENT '固定电话',
  `cust_mobile` varchar(16) DEFAULT NULL COMMENT '移动电话',
  PRIMARY KEY (`cust_id`)
) ENGINE=InnoDB AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;
```

1.3.1.2 创建实体和映射

```
package com.admiral.crm.domain;

public class Customer {

    private Long cust_id;
    private String cust_name;
    private String cust_source;
    private String cust_industry;
    private String cust_level;
    private String cust_phone;
    private String cust_mobile;
    public Long getCust_id() {
        return cust_id;
    }
    public void setCust_id(Long cust_id) {
        this.cust_id = cust_id;
```

```
    }
    public String getCust_name() {
        return cust_name;
    }
    public void setCust_name(String cust_name) {
        this.cust_name = cust_name;
    }
    public String getCust_source() {
        return cust_source;
    }
    public void setCust_source(String cust_source) {
        this.cust_source = cust_source;
    }
    public String getCust_industry() {
        return cust_industry;
    }
    public void setCust_industry(String cust_industry) {
        this.cust_industry = cust_industry;
    }
    public String getCust_level() {
        return cust_level;
    }
    public void setCust_level(String cust_level) {
        this.cust_level = cust_level;
    }
    public String getCust_phone() {
        return cust_phone;
    }
    public void setCust_phone(String cust_phone) {
        this.cust_phone = cust_phone;
    }
    public String getCust_mobile() {
        return cust_mobile;
    }
    public void setCust_mobile(String cust_mobile) {
        this.cust_mobile = cust_mobile;
    }

    @Override
    public String toString() {
        return "Customer [cust_id=" + cust_id + ", cust_name=" + cust_name + ",
cust_source=" + cust_source
                + ", cust_industry=" + cust_industry + ", cust_level=" +
cust_level + ", cust_phone=" + cust_phone
                + ", cust_mobile=" + cust_mobile + "]";
    }
}
```

```
<?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">
```

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

1.3.1.3 创建Action

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

import com.admiral.crm.domain.Customer;
import com.admiral.crm.service.CustomerService;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

public class CustomerAction extends ActionSupport implements
ModelDriven<Customer> {

    // 模型驱动需要用到的对象
    private Customer customer = new Customer();

    @Override
    public Customer getModel() {
        return customer;
    }

    // 注入 Service
    private CustomerService customerService;

    public void setCustomerService(CustomerService customerService) {
        this.customerService = customerService;
    }

}
```

1.3.1.4 创建Service

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

import com.admiral.crm.dao.CustomerDao;
import com.admiral.crm.service.CustomerService;

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

    // 注入客户的 Dao
    private CustomerDao customerDao;

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

}
```

1.3.1.5 创建DAO

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

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

import com.admiral.crm.dao.CustomerDao;

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

}
```

1.3.1.6 配置Action、Service、DAO

```
<!-- ===============  Customer 模块的相关配置  ====================== -->
<bean id="customerDao" class="com.admiral.crm.dao.impl.CustomerDaoImpl">
    <property name="sessionFactory" ref="sessionFactory" />
</bean>
<bean id="customerService"
class="com.admiral.crm.service.impl.CustomerServiceImpl">
    <property name="customerDao" ref="customerDao" />
</bean>
<bean id="customerAction" class="com.admiral.crm.web.action.CustomerAction"
scope="prototype">
    <property name="customerService" ref="customerService" />
</bean>
```

1.3.2 跳转到添加页面

1.3.2.1 修改左侧菜单页面

```
<TR>
    <TD class=menuSmall><A class=style2 href="${pageContext.request.contextPath }/customer_saveUI.action"
        target=main>- 新增客户</A></TD>
</TR>
```

1.3.2.2编写Action中的saveUI的方法

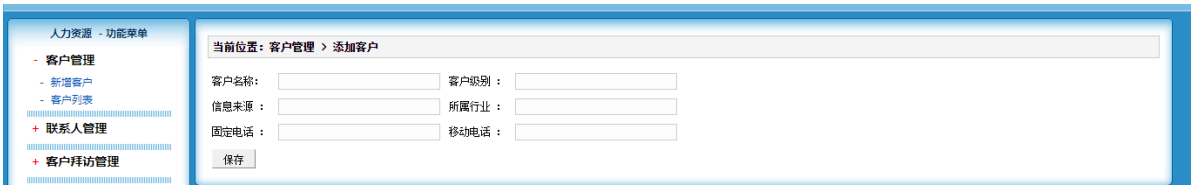
```
package com.admiral.crm.web.action;

import com.admiral.crm.domain.Customer;
import com.admiral.crm.service.CustomerService;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

public class CustomerAction extends ActionSupport implements
ModelDriven<Customer> {

    // 模型驱动需要用到的对象
    private Customer customer = new Customer();

    @Override
    public Customer getModel() {
        return customer;
    }

    // 注入 Service
    private CustomerService customerService;

    public void setCustomerService(CustomerService customerService) {
        this.customerService = customerService;
    }


    /**
     *
```

```
     * Title: saveUI
     * Description: 客户管理:跳转到添加页面:saveUI
     * @return
     */
    public String saveUI() {
        return "saveUI";
    }

}
```

1.3.2.3 配置Action的跳转

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
    "http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>
    <package name="crm" extends="struts-default" namespace="/">
        <!-- 配置用户模块 -->
        <action name="user_*" class="userAction" method="{1}">
            <result name="login">/login.jsp</result>
            <result name="success" type="redirect">/index.jsp</result>
        </action>

        <!-- 配置客户模块 -->
        <action name="customer_*" class="customerAction" method="{1}">
            <result name="saveUI">/jsp/customer/add.jsp</result>
        </action>
    </package>
</struts>
```

1.3.2.4 测试跳转



1.3.3 引入数据字典

1.3.3.1 什么是数据字典

数据字典用来规范某些地方具体值和数据

1.3.3.2 创建数据字典表

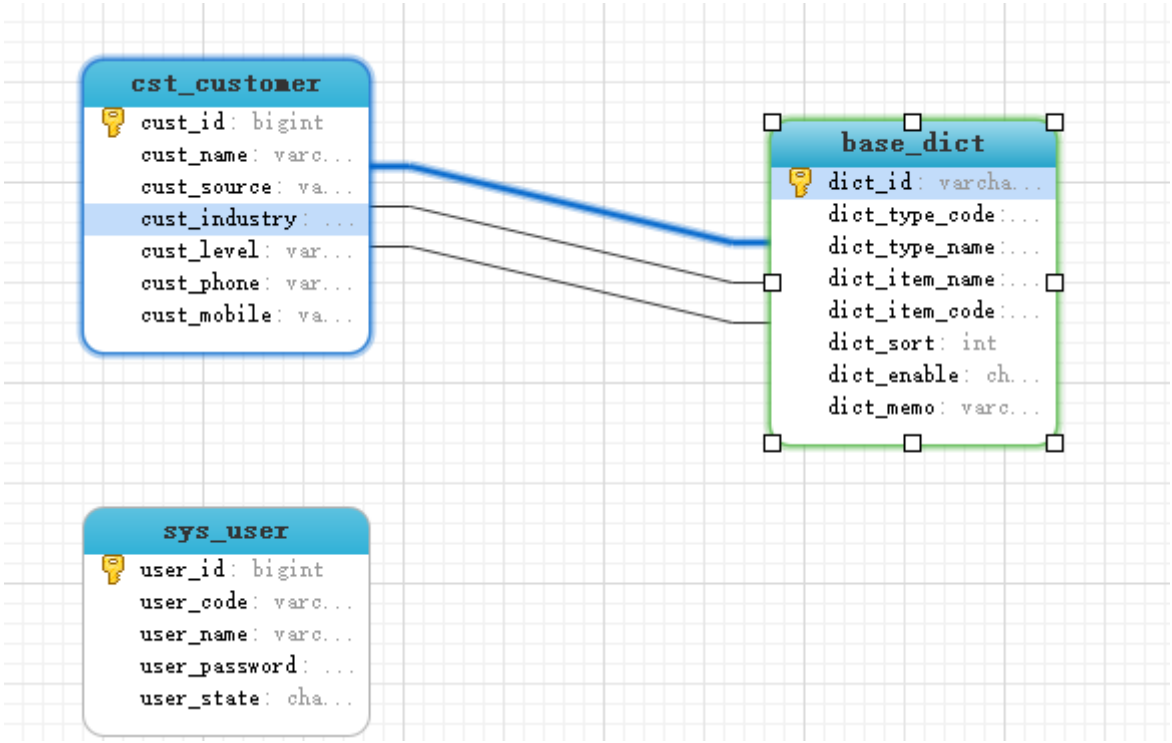
```
CREATE TABLE `base_dict` (
  `dict_id` varchar(32) NOT NULL COMMENT '数据字典id(主键)',
  `dict_type_code` varchar(10) NOT NULL COMMENT '数据字典类别代码',
  `dict_type_name` varchar(64) NOT NULL COMMENT '数据字典类别名称',
  `dict_item_name` varchar(64) NOT NULL COMMENT '数据字典项目名称',
  `dict_item_code` varchar(10) DEFAULT NULL COMMENT '数据字典项目(可为空)',
  `dict_sort` int(10) DEFAULT NULL COMMENT '排序字段',
  `dict_enable` char(1) NOT NULL COMMENT '1:使用 0:停用',
  `dict_memo` varchar(64) DEFAULT NULL COMMENT '备注',
  PRIMARY KEY (`dict_id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

```
insert  into
`base_dict`(`dict_id`,`dict_type_code`,`dict_type_name`,`dict_item_name`,`dict_i
tem_code`,`dict_sort`,`dict_enable`,`dict_memo`) values ('1','001','客户行业','教育
培训 ',NULL,1,'1',NULL),('10','003','公司性质','民企',NULL,3,'1',NULL),
('12','004','年营业额','1-10万',NULL,1,'1',NULL),('13','004','年营业额','10-20
万',NULL,2,'1',NULL),('14','004','年营业额','20-50万',NULL,3,'1',NULL),
('15','004','年营业额','50-100万',NULL,4,'1',NULL),('16','004','年营业额','100-500
万',NULL,5,'1',NULL),('17','004','年营业额','500-1000万',NULL,6,'1',NULL),
('18','005','客户状态','基础客户',NULL,1,'1',NULL),('19','005','客户状态','潜在客
户',NULL,2,'1',NULL),('2','001','客户行业','电子商务',NULL,2,'1',NULL),
('20','005','客户状态','成功客户',NULL,3,'1',NULL),('21','005','客户状态','无效客
户',NULL,4,'1',NULL),('22','006','客户级别','普通客户',NULL,1,'1',NULL),
('23','006','客户级别','VIP客户',NULL,2,'1',NULL),('24','007','商机状态','意向客
户',NULL,1,'1',NULL),('25','007','商机状态','初步沟通',NULL,2,'1',NULL),
('26','007','商机状态','深度沟通',NULL,3,'1',NULL),('27','007','商机状态','签订合
同',NULL,4,'1',NULL),('3','001','客户行业','对外贸易',NULL,3,'1',NULL),
('30','008','商机类型','新业务',NULL,1,'1',NULL),('31','008','商机类型','现有业
务',NULL,2,'1',NULL),('32','009','商机来源','电话营销',NULL,1,'1',NULL),
('33','009','商机来源','网络营销',NULL,2,'1',NULL),('34','009','商机来源','推广活
动',NULL,3,'1',NULL),('4','001','客户行业','酒店旅游',NULL,4,'1',NULL),
('5','001','客户行业','房地产',NULL,5,'1',NULL),('6','002','客户信息来源','电话营
销',NULL,1,'1',NULL),('7','002','客户信息来源','网络营销',NULL,2,'1',NULL),
('8','003','公司性质','合资',NULL,1,'1',NULL),('9','003','公司性质','国
企',NULL,2,'1',NULL);
```

1.3.3.3 客户表和字典表的关系分析

栏位 | 索引 | 外键 | 触发器 | 选项 | 注释 | SQL 预览

名	栏位	参考数据库	被参考表	参考栏位	删除时	更新时
FK_cst_customer_0001	cust_source	crm	base_dict	dict_id	NO ACTION	NO ACTION
FK_cst_customer_0002	cust_level	crm	base_dict	dict_id	NO ACTION	NO ACTION
FK_cst_customer_0003	cust_industry	crm	base_dict	dict_id	NO ACTION	NO ACTION



1.3.3.4 创建字典的实体和映射

- 创建实体

```
package com.admiral.crm.domain;

/**
 * 数据字典的实体
 */
public class BaseDict {

    private String dict_id;
    private String dict_type_code;
    private String dict_type_name;
    private String dict_item_name;
    private String dict_item_code;
    private Integer dict_sort;
    private String dict_enable;
    private String dict_memo;

    public String getDict_id() {
```

```
        return dict_id;
    }

    public void setDict_id(String dict_id) {
        this.dict_id = dict_id;
    }

    public String getDict_type_code() {
        return dict_type_code;
    }

    public void setDict_type_code(String dict_type_code) {
        this.dict_type_code = dict_type_code;
    }

    public String getDict_type_name() {
        return dict_type_name;
    }

    public void setDict_type_name(String dict_type_name) {
        this.dict_type_name = dict_type_name;
    }

    public String getDict_item_name() {
        return dict_item_name;
    }

    public void setDict_item_name(String dict_item_name) {
        this.dict_item_name = dict_item_name;
    }

    public String getDict_item_code() {
        return dict_item_code;
    }

    public void setDict_item_code(String dict_item_code) {
        this.dict_item_code = dict_item_code;
    }

    public Integer getDict_sort() {
        return dict_sort;
    }

    public void setDict_sort(Integer dict_sort) {
        this.dict_sort = dict_sort;
    }

    public String getDict_enable() {
        return dict_enable;
    }

    public void setDict_enable(String dict_enable) {
        this.dict_enable = dict_enable;
    }

    public String getDict_memo() {
        return dict_memo;
    }
```

```
    public void setDict_memo(String dict_memo) {
        this.dict_memo = dict_memo;
    }


}
```

- 创建映射

```
<?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.BaseDict" table="base_dict">
        <id name="dict_id" column="dict_id">
            <generator class="uuid" />
        </id>

        <property name="dict_type_code" column="dict_type_code" />
        <property name="dict_type_name" column="dict_type_name" />
        <property name="dict_item_name" column="dict_item_name" />
        <property name="dict_item_code" column="dict_item_code" />
        <property name="dict_sort" column="dict_sort" />
        <property name="dict_enable" column="dict_enable" />
        <property name="dict_memo" column="dict_memo" />
    </class>
</hibernate-mapping>
```

1.3.3.5 修改字典和客户的关系映射

- 修改客户的实体类

```
package com.admiral.crm.domain;

public class Customer {

    private Long cust_id;
    private String cust_name;
//  private String cust_source;
//  private String cust_industry;
//  private String cust_level;
    private String cust_phone;
    private String cust_mobile;
```

```java
    /*
     * 客户和字典是多对一关系,需要在多的一方放置一的一方的对象
     */
    private BaseDict baseDictSource;
    private BaseDict baseDictIndustry;
    private BaseDict baseDictLevel;

    public BaseDict getBaseDictSource() {
        return baseDictSource;
    }

    public void setBaseDictSource(BaseDict baseDictSource) {
        this.baseDictSource = baseDictSource;
    }

    public BaseDict getBaseDictIndustry() {
        return baseDictIndustry;
    }

    public void setBaseDictIndustry(BaseDict baseDictIndustry) {
        this.baseDictIndustry = baseDictIndustry;
    }

    public BaseDict getBaseDictLevel() {
        return baseDictLevel;
    }

    public void setBaseDictLevel(BaseDict baseDictLevel) {
        this.baseDictLevel = baseDictLevel;
    }

    public Long getCust_id() {
        return cust_id;
    }

    public void setCust_id(Long cust_id) {
        this.cust_id = cust_id;
    }

    public String getCust_name() {
        return cust_name;
    }

    public void setCust_name(String cust_name) {
        this.cust_name = cust_name;
    }

    public String getCust_phone() {
        return cust_phone;
    }

    public void setCust_phone(String cust_phone) {
        this.cust_phone = cust_phone;
    }

    public String getCust_mobile() {
        return cust_mobile;
    }
```

```
    public void setCust_mobile(String cust_mobile) {
        this.cust_mobile = cust_mobile;
    }

}
```

- 修改客户的映射

```
<?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" />

        <!-- 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" />

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

1.3.3.6 将映射文件交给Spring

```
    <!-- 配置 Hibernate SessionFactory 相关配置 -->
    <bean id="sessionFactory"
class="org.springframework.orm.hibernate5.LocalSessionFactoryBean">
        <!-- 注入连接池 -->
        <property name="dataSource" ref="dataSource" />
        <!-- 注入Hibernate 相关配置 -->
        <property name="hibernateProperties">
            <props>
```

```
                <prop
key="hibernate.dialect">org.hibernate.dialect.MySQLDialect</prop>
                <prop key="hibernate.show_sql">true</prop>
                <prop key="hibernate.format_sql">true</prop>
                <prop key="hibernate.hbm2ddl.auto">update</prop>
            </props>
        </property>
        <!-- 引入映射文件 -->
        <property name="mappingResources">
            <list>
                <value>com/admiral/crm/domain/User.hbm.xml</value>
                <value>com/admiral/crm/domain/Customer.hbm.xml</value>
                <value>com/admiral/crm/domain/BaseDict.hbm.xml</value>
            </list>
        </property>
    </bean>
```

1.3.4 在添加页面上异步加载字典数据

1.3.4.1 创建字典的Action、Service、DAO

BaseDictDaoImpl.java

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

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

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

}
```

BaseDictServiceImpl.java

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

import com.admiral.crm.dao.BaseDictDao;
import com.admiral.crm.service.BaseDictService;

/**
 *
```

```
 * @Description: 字典业务层的实现类
 * @author Admiral
 */
public class BaseDictServiceImpl implements BaseDictService {

    // 注入 Dao
    private BaseDictDao baseDictDao;

    public void setBaseDictDao(BaseDictDao baseDictDao) {
        this.baseDictDao = baseDictDao;
    }

}
```

BaseDictAction.java

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

import com.admiral.crm.domain.BaseDict;
import com.admiral.crm.service.BaseDictService;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

public class BaseDictAction extends ActionSupport implements
ModelDriven<BaseDict> {

    // 模型驱动使用的对象
    private BaseDict baseDict = new BaseDict();

    @Override
    public BaseDict getModel() {
        return baseDict;
    }

    // 注入 Service
    private BaseDictService baseDictService;

    public void setBaseDictService(BaseDictService baseDictService) {
        this.baseDictService = baseDictService;
    }

}
```

1.3.4.2 将字典相关类交给Spring

```
    <!-- ================  Customer 模块的相关配置  ====================== -->
    <bean id="baseDictAction" class="com.admiral.crm.web.action.BaseDictAction"
scope="prototype">
        <property name="baseDictService" ref="baseDictService" />
    </bean>

    <bean id="baseDictService"
class="com.admiral.crm.service.impl.BaseDictServiceImpl">
        <property name="baseDictDao" ref="baseDictDao" />
    </bean>

    <bean id="baseDictDao" class="com.admiral.crm.dao.impl.BaseDictDaoImpl">
        <property name="sessionFactory" ref="sessionFactory" />
    </bean>
```

1.3.4.3 引入jquery的js（在添加页面上）

```
<script type="text/javascript" src="${pageContext.request.contextPath }/js/jquery-1.11.3.min.js"></script>
```

1.3.4.4 编写异步加载的方法

```
<script type="text/javascript">
    $(function(){
        // 页面加载函数就会执行：
        // 页面加载，异步查询字典数据：
        // 加载客户来源
        $.post("${pageContext.request.contextPath
}/baseDict_findByTypeCode.action",{"dict_type_code":"002"},function(data){
            // 遍历json的数据：
            $(data).each(function(i,n){
                $("#cust_source").append("<option
value='"+n.dict_id+"'>"+n.dict_item_name+"</option>");
            });
        },"json");
    });
</script>
```

1.3.4.5 编写Action

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

import java.io.IOException;
import java.util.List;

import org.apache.struts2.ServletActionContext;
```

```
import com.admiral.crm.domain.BaseDict;
import com.admiral.crm.service.BaseDictService;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

import net.sf.json.JSONArray;
import net.sf.json.JsonConfig;

public class BaseDictAction extends ActionSupport implements
ModelDriven<BaseDict> {

    // 模型驱动使用的对象
    private BaseDict baseDict = new BaseDict();

    @Override
    public BaseDict getModel() {
        return baseDict;
    }

    // 注入 Service
    private BaseDictService baseDictService;

    public void setBaseDictService(BaseDictService baseDictService) {
        this.baseDictService = baseDictService;
    }

    /**
     *
     * Title: findByTypeCode
     * Description: 根据类型名称查询字典的方法
     * @return
     * @throws IOException
     */
    public String findByTypeCode() throws IOException {
        System.out.println("根据类型名称查询字典的方法执行了....");
        //调用业务层查询
        List<BaseDict> list =
 baseDictService.findByTypeCode(baseDict.getDict_type_code());

        //将 list 转换为 json    jsonlib  fastjson

        /**
         * JSONConfig:转json的配置对象
         * JSONArray:将数组和List转换为json
         * JSONObject:将对象和Map转换为json
         */
        JsonConfig jsonConfig = new JsonConfig();
        jsonConfig.setExcludes(new String[]
{"dict_sort","dict_enable","dict_memo"});

        JSONArray jsonArray = JSONArray.fromObject(list, jsonConfig);
        System.out.println(jsonArray);

        //将json数据打印到页面

ServletActionContext.getResponse().setContentType("text/html;charset=utf-8");
```

```
ServletActionContext.getResponse().getWriter().write(jsonArray.toString());

        return NONE;
    }

}
```

1.3.4.6 编写Service

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

import java.util.List;

import com.admiral.crm.dao.BaseDictDao;
import com.admiral.crm.domain.BaseDict;
import com.admiral.crm.service.BaseDictService;

/**
 *
 * @Description: 字典业务层的实现类
 * @author Admiral
 */
public class BaseDictServiceImpl implements BaseDictService {

    // 注入 Dao
    private BaseDictDao baseDictDao;

    public void setBaseDictDao(BaseDictDao baseDictDao) {
        this.baseDictDao = baseDictDao;
    }

    @Override
    public List<BaseDict> findByTypeCode(String dict_type_code) {
        return baseDictDao.findByTypeCode(dict_type_code);
    }

}
```

1.3.4.7 编写DAO

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

import java.util.List;
```

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

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 HibernateDaoSupport implements BaseDictDao
{

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

}
```

- 修改添加页面

```
<td>信息来源：</td>
<td>
<select id="cust_source">
    <option value="">--请选择--</option>
</select>
</td>
<td>所属行业：</td>
```

1.3.4.8 加载其他字典项数据

```
<script type="text/javascript">
    $(function(){
        // 页面加载函数就会执行：
        // 页面加载，异步查询字典数据：
        // 加载客户来源
        $.post("${pageContext.request.contextPath
}/baseDict_findByTypeCode.action",{"dict_type_code":"002"},function(data){
            // 遍历json的数据：
            $(data).each(function(i,n){
                $("#cust_source").append("<option
value='"+n.dict_id+"'>"+n.dict_item_name+"</option>");
            });
        },"json");
        $.post("${pageContext.request.contextPath
}/baseDict_findByTypeCode.action",{"dict_type_code":"006"},function(data){
            // 遍历json的数据：
            $(data).each(function(i,n){
```

```
                $("#cust_level").append("<option
value='"+n.dict_id+"'>"+n.dict_item_name+"</option>");
            });
        },"json");
        $.post("${pageContext.request.contextPath
}/baseDict_findByTypeCode.action",{"dict_type_code":"001"},function(data){
            // 遍历json的数据:
            $(data).each(function(i,n){
                $("#cust_industry").append("<option
value='"+n.dict_id+"'>"+n.dict_item_name+"</option>");
            });
        },"json");
    });
</script>
```

```
            <td>客户级别：</td>
            <td>
                <select id="cust_level">
                    <option value="">-请选择-</option>
                </select>
            </td>
        </TR>

        <TR>

            <td>信息来源：</td>
            <td>
            <select id="cust_source">
                <option value="">-请选择-</option>
            </select>
            </td>
            <td>所属行业：</td>
            <td>
                <select id="cust_industry">
                    <option value="">-请选择-</option>
                </select>
            </td>
```

1.3.5 保存数据到数据库中

1.3.5.1 修改添加页面

```
ODY>
  <FORM id=form1 name=form1
      action="${pageContext.request.contextPath }/customer_save.action"
      method=post>
```

- 修改表单项名称

```
        </td>
        <td>客户级别：</td>
        <td>
            <select id="cust_level" name="baseDictLevel.dict_id">
                <option value="">-请选择-</option>
            </select>
        </td>
    </TR>

    <TR>

        <td>信息来源：</td>
        <td>
        <select id="cust_source" name="baseDictSource.dict_id">
            <option value="">-请选择-</option>
        </select>
        </td>
        <td>所属行业：</td>
        <td>
            <select id="cust_industry" name="baseDictIndustry.dict_id">
                <option value="">-请选择-</option>
            </select>
        </td>
    </TR>
```

1.3.5.2 编写Action

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

import com.admiral.crm.domain.Customer;
import com.admiral.crm.service.CustomerService;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

public class CustomerAction extends ActionSupport implements
ModelDriven<Customer> {

    // 模型驱动需要用到的对象
    private Customer customer = new Customer();

    @Override
    public Customer getModel() {
        return customer;
    }
```

```
    // 注入 Service
    private CustomerService customerService;

    public void setCustomerService(CustomerService customerService) {
        this.customerService = customerService;
    }


    /**
     *
     * Title: saveUI
     * Description: 客户管理:跳转到添加页面:saveUI
     * @return
     */
    public String saveUI() {
        return "saveUI";
    }

    /**
     *
     * Title: save
     * Description: 客户管理:保存客户
     * @return
     */
    public String save() {
        customerService.save(customer);
        return NONE;
    }

}
```

1.3.5.3 编写Service

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

import org.springframework.transaction.annotation.Transactional;

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

}
```

1.3.5.4 编写DAO

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

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);
    }

}
```

1.3.5.5 添加事务：

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

1.4 CRM 综合练习:客户管理分页查询客户

1.4.1 查询客户（分页）

1.4.1.1 修改menu.jsp

```
<TR>
    <TD class=menuSmall><A class=style2 href="${pageContext.request.contextPath }/customer_findAll.action"
        target=main>- 客户列表</A></TD>
</TR>
```

1.4.1.2 编写Action中findAll的方法

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

import org.hibernate.criterion.DetachedCriteria;

import com.admiral.crm.domain.Customer;
import com.admiral.crm.domain.PageBean;
import com.admiral.crm.service.CustomerService;
import com.opensymphony.xwork2.ActionContext;
import com.opensymphony.xwork2.ActionSupport;
import com.opensymphony.xwork2.ModelDriven;

public class CustomerAction extends ActionSupport implements
ModelDriven<Customer> {

    // 模型驱动需要用到的对象
    private Customer customer = new Customer();

    @Override
    public Customer getModel() {
        return customer;
    }

    // 注入 Service
    private CustomerService customerService;

    public void setCustomerService(CustomerService customerService) {
        this.customerService = customerService;
    }

    // 注入当前页
    private Integer currPage = 1;

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

    // 注入每页记录数
    private Integer pageSize = 3;
```

```
    public void setPageSize(Integer pageSize) {
        if (pageSize == null) {
            pageSize = 1;
        }
        this.pageSize = pageSize;
    }


    /**
     *
     * Title: findAll
     * Description: 客户管理:分页查询
     * @return
     */
    public String findAll() {
        // 接收分页参数
        //
        DetachedCriteria detachedCriteria =
DetachedCriteria.forClass(Customer.class);
        System.out.println(pageSize);
        //调用业务层查询
        PageBean<Customer> pageBean =
customerService.findByPage(detachedCriteria,currPage,pageSize);

        //将分页对象保存到值栈
        ActionContext.getContext().getValueStack().push(pageBean);
        return "findAll";
    }


    /**
     *
     * Title: saveUI
     * Description: 客户管理:跳转到添加页面:saveUI
     * @return
     */
    public String saveUI() {
        return "saveUI";
    }

    /**
     *
     * Title: save
     * Description: 客户管理:保存客户
     * @return
     */
    public String save() {
        customerService.save(customer);
        return NONE;
    }

}
```

PageBean.java

```
package com.admiral.crm.domain;

import java.util.List;

/**
 *
 * @Description: 分页查询实体对象
 * @author Admiral
 */
public class PageBean<T> {

    // 当前页
    private Integer currPage;
    // 每页记录数
    private Integer pageSize;
    // 总页数
    private Integer totalPage;
    // 总记录数
    private Integer totalCount;
    // 每页显示的数据
    private List<T> list;

    public Integer getCurrPage() {
        return currPage;
    }

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

    public Integer getPageSize() {
        return pageSize;
    }

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

    public Integer getTotalPage() {
        return totalPage;
    }

    public void setTotalPage(Integer totalPage) {
        this.totalPage = totalPage;
    }

    public Integer getTotalCount() {
        return totalCount;
    }

    public void setTotalCount(Integer totalCount) {
        this.totalCount = totalCount;
    }

    public List<T> getList() {
```

```
        return list;
    }

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

}
```

1.4.1.3 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.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;
    }

}
```

1.4.1.4 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.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);
    }

}
```

1.4.1.5 配置页面跳转

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE struts PUBLIC
    "-//Apache Software Foundation//DTD Struts Configuration 2.3//EN"
    "http://struts.apache.org/dtds/struts-2.3.dtd">

<struts>
    <package name="crm" extends="struts-default" namespace="/">

        <action name="user_*" class="userAction" method="{1}">
            <result name="login">/login.jsp</result>
            <result name="success" type="redirect">/index.jsp</result>
        </action>

        <!-- 配置客户模块 -->
        <action name="customer_*" class="customerAction" method="{1}">
            <result name="saveUI">/jsp/customer/add.jsp</result>
            <result name="findAll">/jsp/customer/list.jsp</result>
        </action>

        <!-- 配置字典 -->
        <action name="baseDict_*" class="baseDictAction" method="{1}">

        </action>
    </package>
</struts>
```

1.4.1.6 在list.jsp中显示数据

```
    style="FONT-WEIGHT: normal; FONT-STYLE: normal; BACKGROUND-COLOR: wh
    <TD><s:property value="cust_name"/></TD>
    <TD><s:property value="baseDictLevel.dict_item_name"/></TD>
    <TD><s:property value="baseDictSource.dict_item_name"/></TD>
    <TD><s:property value="baseDictIndustry.dict_item_name"/></TD>
    <TD><s:property value="cust_phone"/></TD>
    <TD><s:property value="cust_mobile"/></TD>
    <TD>
```

- 解决延迟加载问题

```
<?xml version="1.0" encoding="UTF-8"?>
<web-app xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://java.sun.com/xml/ns/javaee"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
http://java.sun.com/xml/ns/javaee/web-app_2_5.xsd" id="WebApp_ID" version="2.5">
  <display-name>crm</display-name>
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.htm</welcome-file>
    <welcome-file>index.jsp</welcome-file>
    <welcome-file>default.html</welcome-file>
    <welcome-file>default.htm</welcome-file>
    <welcome-file>default.jsp</welcome-file>
  </welcome-file-list>

  <!-- 配置 Spring 的核心监听器 -->
  <listener>
    <listener-
class>org.springframework.web.context.ContextLoaderListener</listener-class>
  </listener>
  <context-param>
    <param-name>contextConfigLocation</param-name>
    <param-value>classpath:applicationContext.xml</param-value>

  </context-param>

  <!-- 解决延迟加载问题 -->
  <filter>
    <filter-name>openSessionInViewFilter</filter-name>
    <filter-
class>org.springframework.orm.hibernate5.support.OpenSessionInViewFilter</filter
-class>
  </filter>
  <filter-mapping>
    <filter-name>openSessionInViewFilter</filter-name>
    <url-pattern>*.action</url-pattern>
  </filter-mapping>


  <!-- 配置 Struts2 核心过滤器 -->
  <filter>
    <filter-name>struts2</filter-name>
```

```
    <filter-
class>org.apache.struts2.dispatcher.ng.filter.StrutsPrepareAndExecuteFilter</fil
ter-class>
  </filter>

  <filter-mapping>
    <filter-name>struts2</filter-name>
    <url-pattern>/*</url-pattern>
  </filter-mapping>
</web-app>
```

list.jsp

```
<%@ page language="java" contentType="text/html; charset=UTF-8"
    pageEncoding="UTF-8"%>
<%@ taglib uri="/struts-tags" prefix="s" %>
<%-- <%@ taglib uri="http://java.sun.com/jsp/jstl/core" prefix="c"%> --%>
<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN"
"http://www.w3.org/TR/html4/loose.dtd">
<html>
<head>
<TITLE>客户列表</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>
<script type="text/javascript" src="${pageContext.request.contextPath
}/js/jquery-1.4.4.min.js"></script>
<SCRIPT language=javascript>
    function to_page(page){
        if(page){
            $("#page").val(page);
        }
        document.customerForm.submit();

    }
</SCRIPT>

<META content="MSHTML 6.00.2900.3492" name=GENERATOR>
</HEAD>
<BODY>
    <FORM id="customerForm" name="customerForm"
        action="${pageContext.request.contextPath }/customer_findAll.action"
        method=post>

        <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：客户管理 &gt; 客户列表
</TD>
                            </TR>
                            <TR>
                                <TD height=2></TD>
                            </TR>
                        </TABLE>
                        <TABLE borderColor=#cccccc cellSpacing=0 cellPadding=0
                            width="100%" align=center border=0>
                            <TBODY>
                                <TR>
                                    <TD height=25>
                                        <TABLE cellSpacing=0 cellPadding=2
border=0>
                                            <TBODY>
                                                <TR>
                                                    <TD>客户名称：</TD>
                                                    <TD><INPUT class=textbox
id=sChannel2
                                                        style="WIDTH: 80px"
maxLength=50 name="custName"></TD>

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

                                <TR>
                                    <TD>
                                        <TABLE id=grid
                                            style="BORDER-TOP-WIDTH: 0px; FONT-
WEIGHT: normal; BORDER-LEFT-WIDTH: 0px; BORDER-LEFT-COLOR: #cccccc; BORDER-
BOTTOM-WIDTH: 0px; BORDER-BOTTOM-COLOR: #cccccc; WIDTH: 100%; BORDER-TOP-COLOR:
#cccccc; FONT-STYLE: normal; BACKGROUND-COLOR: #cccccc; BORDER-RIGHT-WIDTH: 0px;
TEXT-DECORATION: none; BORDER-RIGHT-COLOR: #cccccc"
```

```
                                        cellSpacing=1 cellPadding=2
rules=all border=0>
                                        <TBODY>
                                            <TR
                                                style="FONT-WEIGHT: bold;
FONT-STYLE: normal; BACKGROUND-COLOR: #eeeeee; TEXT-DECORATION: none">
                                                <TD>客户名称</TD>
                                                <TD>客户级别</TD>
                                                <TD>客户来源</TD>
                                                <TD>所属行业</TD>
                                                <TD>电话</TD>
                                                <TD>手机</TD>
                                                <TD>操作</TD>
                                            </TR>
                                            <s:iterator value="list">

                                            <TR
                                                style="FONT-WEIGHT: normal;
FONT-STYLE: normal; BACKGROUND-COLOR: white; TEXT-DECORATION: none">
                                                <TD><s:property
value="cust_name"/></TD>
                                                <TD><s:property
value="baseDictLevel.dict_item_name"/></TD>
                                                <TD><s:property
value="baseDictSource.dict_item_name"/></TD>
                                                <TD><s:property
value="baseDictIndustry.dict_item_name"/></TD>
                                                <TD><s:property
value="cust_phone"/></TD>
                                                <TD><s:property
value="cust_mobile"/></TD>
                                                <TD>
                                                <a
href="${pageContext.request.contextPath }/customerServlet?
method=edit&custId=${customer.cust_id}">修改</a>
                                                  
                                                <a
href="${pageContext.request.contextPath }/customerServlet?
method=delete&custId=${customer.cust_id}">删除</a>
                                                </TD>
                                            </TR>
                                            </s:iterator>

                                        </TBODY>
                                    </TABLE>
                                </TD>
                            </TR>

                            <TR>
                                <TD><SPAN id=pagelink>
                                        <DIV
                                            style="LINE-HEIGHT: 20px;
HEIGHT: 20px; TEXT-ALIGN: right">
                                            共[<B><s:property
value="totalCount"/></B>]条记录,[<B><s:property value="totalPage"/></B>]页
                                            ,每页显示
```

```
                                                <select name="pageSize"
onchange="to_page()">
                                                    <option value="3" <s:if
test="pageSize == 3">selected</s:if>>3</option>
                                                    <option value="5" <s:if
test="pageSize == 5">selected</s:if>>5</option>
                                                    <option value="10" <s:if
test="pageSize == 10">selected</s:if>>10</option>
                                                </select>
                                                条
                                                <s:if test="currPage != 1">
                                                    [<A
href="javascript:to_page(<s:property value="1" />)">首页</A>]
                                                    [<A
href="javascript:to_page(<s:property value="currPage-1" />)">前一页</A>]
                                                </s:if>  
                                                <B>

                                                    <s:iterator var="i"
begin="1" end="totalPage">
                                                        <s:if test="#i ==
currPage">
                                                            <s:property
value="#i"/>
                                                        </s:if>
                                                        <s:else>
                                                            <a
href="javascript:to_page(<s:property value="#i" />)"><s:property value="#i"/>
</a>
                                                        </s:else>
                                                    </s:iterator>

                                                </B>  
                                                <s:if test="currPage !=
totalPage">
                                                    [<A
href="javascript:to_page(<s:property value="currPage+1" />)">后一页</A>]
                                                    [<A
href="javascript:to_page(<s:property value="totalPage" />)">尾页</A>]
                                                </s:if>
                                                到
                                                <input type="text" size="3"
id="page" name="currPage" />
                                                页

                                                <input type="button" value="Go"
onclick="to_page()"/>
                                            </DIV>
                                        </SPAN></TD>
                                    </TR>
                                </TBODY>
                            </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>
    </FORM>
</BODY>
</HTML>
```