# 合同管理详细设计

## 设计思路

合同管理分为前端界面和后台实现。后台基于MVC模式，分别实现了合同新增、合同删除、合同修改、合同查询功能；前端基于HTML5+CSS3+Angularjs实现。用户通过进入合同新增页面，加载页面信息，录入合同基本信息，验证通过后，将合同信息添加到数据库中并提示新增信息成功，否则，提示相应的错误信息。

## 合同管理

合同管理（zenith.c*ont*ract）

包 名：zenith.c*ont*ract

数据处理：zenith.c*ont*ract/ c*ont*ract.model/dao

业务处理：zenith.c*ont*ract/ c*ont*ract.model/model.service

逻辑控制：zenith.c*ont*ract/ c*ont*ract.controller

### 包结构一览图

合同管理包结构如下图所示：

ZContractPartAUnitDao

ZContractPartAUnitService

ZContractDeptService

ZContractDept

ZContractPartAUnit

ZContractDeptDao

ZContractBaseService

ZContractBase

Zenith.c*ont*ract

C*ont*ract.model

Zenith.controller

Model.dbUtil

Model.dao

Model.entity

Model.service

ZContactBaseController

ZContactDeptController

ZContactPartAUnitController

IDbUtil

DbUtil

ZContractBaseDao

类

类

ZContractPartAUnitDao

ZContractPartAUnitService

ZContractDeptService

ZContractDept

ZContractPartAUnit

ZContractDeptDao

ZContractBaseService

ZContractBase

zenith\_c*ont*ract

C*ont*ract\_model

zenith\_controller

Model\_dbUtil

Model\_dao

Model\_entity

Model\_service

ZContactBaseController

ZContactDeptController

ZContactPartAUnitController

IDbUtil

DbUtil

ZContractBaseDao

类

类

说明： 合同管理（zenith\_contract）中包含包：Contract.model、Zenith.controller、Model.dbUtil、Model.dao、Model.entity、Model.service；类：DbUtil、ZContractBase、ZContractDept、ZContractPartAUnit、ZContractBaseDao、ZContractDeptDao、ZContractPartAUnitDao、ZContractBaseService、ZContractDeptService、ZContractPartAUnitService、ZContactBaseController、ZContactDeptController、ZContactPartAUnitController。其中，Model\_dbUtil包为连接数据库缩建的包；Model\_entity为实体类所建立的包；Model\_dao为实体类的增删改查所建立的包；Model\_service为处理业务逻辑缩建立的包；zenith\_controller包为提供接口所建立的包。

### 连接数据库

/\*\*

\* 摘要:

\* 连接Oracle数据库

\* 参数：无

\*

\*

\* 返回：无

\*

\*

\*\*/

方法：

public void zenith.contract.contract.model.DbUtil.getConn()｛

// 1.驱动

private static final String DIRVER = "oracle.jdbc.driver.OracleDriver";

// 2.连接地址

private static final String URL = "jdbc:oracle:thin:@192.168.16.224:1521:zcloud";

// 3.数据库名

private static final String USERNAME = "zsearch";

// 4.数据库密码

private static final String PASSWORD = "zenith";

java.sql.Connection nConnection = null;

try {

// 1.加载驱动

Class.forName(DIRVER);

// 2.创建数据库连接

nConnection = java.sql.DriverManager.getConnection(URL, USERNAME,PASSWORD);

} catch (ClassNotFoundException e) {

e.printStackTrace();

} catch (java.sql.SQLException e) {

e.printStackTrace();

}

return nConnection;

}

### 实体类的创建

/\*\*

\* 摘要:

\* ZcontractBase合同基本信息的实体类get、set方法。

\*

\*

\*\*/

public class ZContractBase {

根据数据库表结构的设计定义合同基本信息的字段。

设置字段的get、set方法。

}

### 实体类DAO的操作

在Model.dao包里实现实体类的增删改查操作。

#### 新增操作

/\*\*

\* 摘要:

\* 合同基本信息ZcontractBase的添加操作

\* 参数：

\* oiZcontractBase（合同基本信息）

\* 返回：

\* int count

\* 影响的行数

\*\*/

方法：

Public int add（ZcontractBase oiZcontractBase）｛

1.DbUtil dbConnection = new DbUtil();创建DbUtil对象

2.调用DbUtil对象中的添加功能。

2.1 调用DbUtil对象中的连接数据库的方法

dbConnection.getConn();

2.2定义SQL语句

String sql = “insert ZcontractBase…….”;

2.3根据SQL语句，调用DbUtil对象中的相应方法。

dbConnection. exeUpdate(sql)；

3.返回影响的行数。  
｝

其他类的操作省略。

#### 修改功能

#### 查询功能

#### 删除功能

### 业务服务实现

/\*\*

\* 摘要:

\* 合同基本信息ZcontractBase的新增业务逻辑操作

\* 参数：

\* oiZcontractBase（合同基本信息）

\* 返回：

\* int count

\* 影响的行数

\*\*/

方法：

public int add(ZcontractBase oiZcontractBase){

//创建nZcontractBaseDao对象

ZcontractBaseDao nZcontractBaseDao = new ZcontractBaseDao();

//调用DAO里的新增方法。

return gd.add(goods);

}  
｝

### 业务控制实现

/\*\*

\* 摘要:

\* 合同基本信息ZcontractBase的添加接口实现

\* 参数：

\* request，response

\* 返回：

\* 返回(request,response)请求

\*

\*\*/

方法：

public void add(HttpServletRequest request, HttpServletResponse response){

//1.接收页面传递的参数

String contractName= request.getParameter("contractName ");

………

ZcontractBase nZcontractBase= new ZcontractBase ();

//2.组装ZcontractBase对象

goods.setgName(gName);

nZcontractBase.seContractName(contractName);

………..

//3.定义ZcontractBaseService对象，调用添加方法

ZcontractBaseService nZcontractBaseService = new ZcontractBaseService ();

nZcontractBaseService.add(nZcontractBase);

返回(request,response);

}

### 页面功能的实现

* 页面初始化时：根据js，加载【承担部门】和【项目经理】下拉树的信息并绑定到对应下拉树中。
* 【合同规模】的类型为“大”“中”“小”，当【合同金额】失去焦点时，选择【合同规模】：

ContractApp.controller("contractCtl", function($scope) {

$scope.bigScale=false;

//合同金额失去焦点

$scope.ContractAmountBlur=function(){

if(“ContractAmount”>5000000){

选中“大”按钮

}else if(“ContractAmount”<=5000000 &&“ContractAmount”>1000000){

选中“中”按钮

}else{

选中“小”按钮

}

};

* 当【甲方单位】失去焦点时：从ZcontractPartAUnit表中查询出【联系人】【联系电话】。

/\*\*

\* 摘要:

\* 根据甲方单位查找联系人和联系电话

\* 参数：

\* PartAUnitName 甲方单位

\* 返回：

\* ZcontractPartAUnit

\*

\*\*/

Public ZcontractPartAUnit SelectPartAUnit（String PartAUnitName）｛

//1.创建ZcontractPartAUnit对象

ZcontractPartAUnit nZcontractPartAUnit = newZcontractPartAUnit();

nZcontractPartAUnit.setPartAUnitName(PartAUnitName);

//2.创建nZcontractPartAUnitService对象

调用nZcontractPartAUnitService里的SelectPartAUnit方法

//3.返回ZcontractPartAUnit对象

｝

* 点击保存:验证必填项非空

if(ContractNo==””|| ContractNo==null || ContractName==””|| ContractName==null……){

alert（“XX项不能为空”）；

}

* 【甲方单位】空时联系电话、联系人是否为空
* 【承担部门】为空时，【项目经理】选项的情况。等