

# 业务

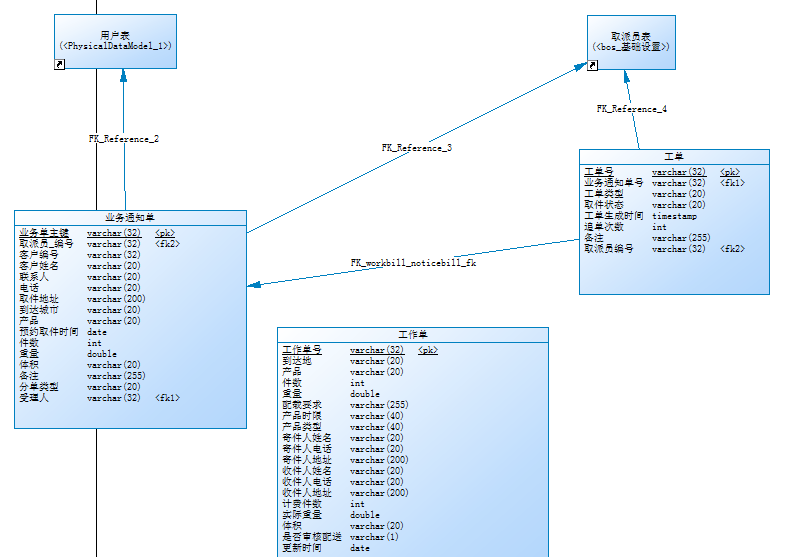
1.、使用powerDesigner设计表 再把表转成sql语句 导入到数据库中

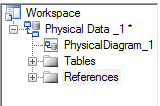
3、实现业务受理自动分单

* + 在CRM服务端扩展方法根据手机号查询客户信息
  + 在CRM服务端扩展方法根据取件地址查询定区id
  + 调整业务受理页面回显客户信息
  + 实现自动分单

4、数据表格编辑功能使用方法

# 创建业务受理环节的数据表

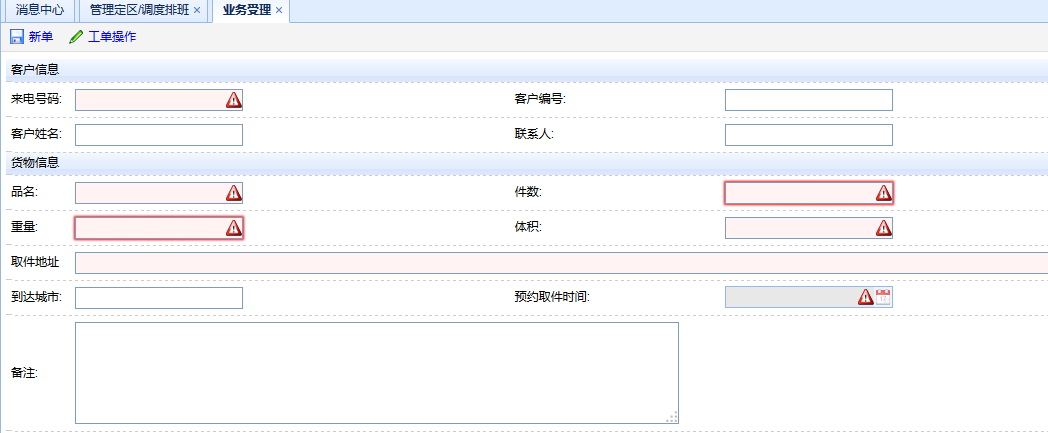


**这是目录：工作目录下，表 视图外键都有 可以保存为.Pdm文件 也可以生成sql语句**

**注意事项：设置外键的时候需要注意方法和字段 如果字段弄错了 可以点击中间的线双击进入窗口 join进行更改**

# 业务受理自动分单

页面：WEB-INF/pages/qupai/noticebill\_add.jsp

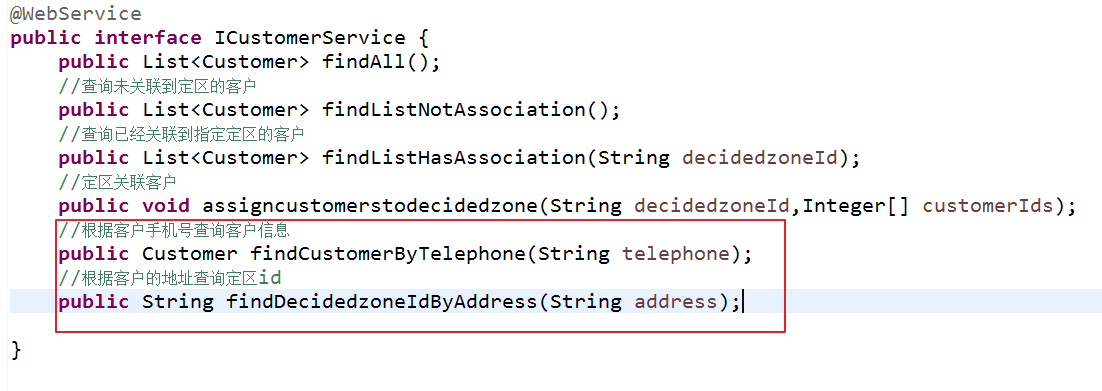


## 在crm服务中扩展方法

**根据电话查询customer 根据地址查询定区**

* 根据客户的手机号查询客户信息
* 根据客户的取件地址查询定区id

接口：



实现类：

//根据手机号查询客户信息

**public** Customer findCustomerByTelephone(String telephone) {

String sql = "select \* from t\_customer where telephone = ?";

//**使用spring 整合dao层 （不使用别的什么工具或是框架）**

**List<Customer> list = jdbcTemplate.query(sql, new RowMapper<Customer>(){**

**public Customer mapRow(ResultSet rs, int arg1) throws SQLException {**

**int id = rs.getInt("id");//根据字段名称从结果集中获取对应的值**

**String name = rs.getString("name");**

**String station = rs.getString("station");**

**String telephone = rs.getString("telephone");**

**String address = rs.getString("address");**

**String decidedzone\_id = rs.getString("decidedzone\_id");**

**return new Customer(id, name, station, telephone, address, decidedzone\_id);**

**}**

**},telephone); //返回一个对象的集合**

**if**(list != **null** && list.size() > 0){

**return** list.get(0);

}

**return** **null**;

}

**public** String findDecidedzoneIdByAddress(String address) {

String sql = "select decidedzone\_id from t\_customer where address = ?";

**String decidedzoneId = jdbcTemplate.queryForObject(sql, String.class, address);**

**//返回某个字段**

**return** decidedzoneId;

}

## 页面调整

第一步：为手机号输入框绑定离焦事件

<td>来电号码:</td>

<td><input type=*"text"* class=*"easyui-validatebox"* name=*"telephone"* required=*"true"* />

<script type=*"text/javascript"*>

$(function(){

//页面加载完成后，为手机号输入框绑定离焦事件

**$("input[name=telephone]").blur(function(){**

//获取页面输入的手机号

var telephone = this.value;

//发送ajax请求，请求Action，在Action中远程掉调用crm服务，获取客户信息，用于页面回显

$.post('noticebillAction\_findCustomerByTelephone.action',{"telephone":telephone},function(data){

if(data != null){

//查询到了客户信息，可以进行页面回显

var customerId = data.id;

var customerName = data.name;

var address = data.address;

$("input[name=customerId]").val(customerId);

$("input[name=customerName]").val(customerName);

$("input[name=delegater]").val(customerName);

$("input[name=pickaddress]").val(address);

}else{

//没有查询到客户信息，不能进行页面回显

$("input[name=customerId]").val("");

$("input[name=customerName]").val("");

$("input[name=delegater]").val("");

$("input[name=pickaddress]").val("");

}

});

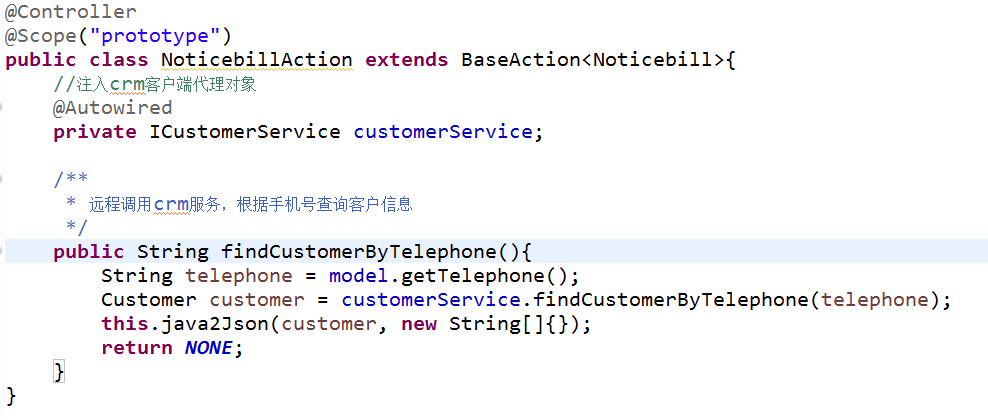
});

});

</script>

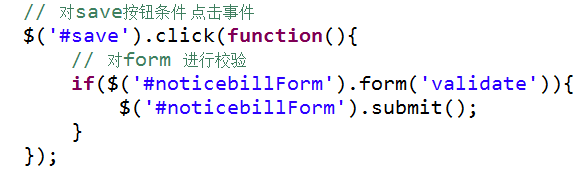
</td>

第二步：创建NoticebillAction，注入crm代理对象，提供方法根据手机号查询客户信息，返回json



注意：配置struts.xml

第三步：为页面中“新单”按钮绑定事件



## 在NoticebillAction中提供方法实现业务受理自动分单



Service代码：

@Service

@Transactional

**public** **class** NoticebillServiceImpl **implements** INoticebillService {

@Autowired

**private** INoticebillDao noticebillDao;

@Autowired

**private** IDecidedzoneDao decidedzoneDao;

@Autowired

**private** IWorkbillDao workbillDao;

@Autowired

**private** ICustomerService customerService;

/\*\*

\* **保存业务通知单，还有尝试自动分单**

\*/

**public** **void** save(Noticebill model) { //封装业务通知单 受理人 取派员

User user = BOSUtils.*getLoginUser*();

model.setUser(user);//设置当前登录用户

noticebillDao.save(model);

//获取客户的取件地址

String pickaddress = model.getPickaddress();

**//远程调用crm服务，根据取件地址查询定区id**

String decidedzoneId = customerService.findDecidedzoneIdByAddress(pickaddress);

**if**(decidedzoneId != **null**){

//查询到了定区id，可以完成自动分单

Decidedzone decidedzone = decidedzoneDao.findById(decidedzoneId);

Staff staff = decidedzone.getStaff();

model.setStaff(staff);//业务通知单关联取派员对象

//设置分单类型为：自动分单

model.setOrdertype(Noticebill.***ORDERTYPE\_AUTO***);

//为取派员产生一个工单

Workbill workbill = **new** Workbill();

workbill.setAttachbilltimes(0);//追单次数

workbill.setBuildtime(**new** Timestamp(System.*currentTimeMillis*()));//创建时间，当前系统时间

workbill.setNoticebill(model);//工单关联页面通知单

workbill.setPickstate(Workbill.***PICKSTATE\_NO***);//取件状态

workbill.setRemark(model.getRemark());//备注信息

workbill.setStaff(staff);//工单关联取派员

workbill.setType(Workbill.***TYPE\_1***);//工单类型

workbillDao.save(workbill);

//调用短信平台，发送短信

}**else**{

//没有查询到定区id，不能完成自动分单

model.setOrdertype(Noticebill.***ORDERTYPE\_MAN***);

}

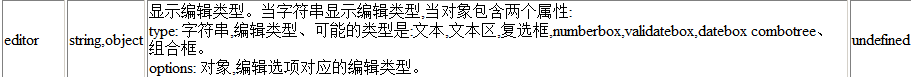
}

}

# datagrid编辑功能使用方式

**数据表格编辑功能是以列为单位。**

* 通过数据表格中的列属性定区具体那一列具有编辑功能：



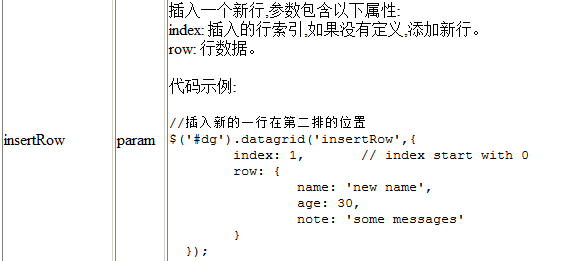
* 开始编辑：



结束编辑：



* 插入一行数据：



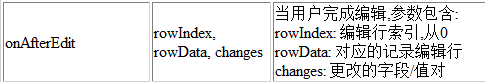
* 删除一行：



* 获得指定行对象的索引



* 数据表格提供的用于监听结束编辑事件



<table id=*"mytable"*></table>

<!-- 方式三：使用easyUI提供的API创建datagrid -->

<script type=*"text/javascript"*>

$(**function**(){

**var myIndex = -1;//全局变量，值为正在编辑行的索引**

//页面加载完成后，创建数据表格datagrid

$("#mytable").datagrid({

//定义标题行所有的列 **在这里设置如果打开编辑器会以什么形式编辑**

columns:[[

{title:'编号',field:'id',checkbox:**true**},

{width:150,title:'姓名',field:'name',**editor:{**

**type:'validatebox',//输入**

**options:{}//设置项**

**}**},

{width:150,title:'年龄',field:'age',**editor:{**

**type:'numberbox',//只能输入数字**

**options:{}**

**}**},

{width:150,title:'日期',field:'address',**editor:{**

**type:'datebox',//只能输入日期 会有一个控件**

**options:{}**

**}**}

]],

//指定数据表格发送ajax请求的地址

url:'${pageContext.request.contextPath }/json/datagrid\_data.json',

rownumbers:**true**,

singleSelect:**true**,

//定义工具栏

toolbar:[

{text:'添加',iconCls:'icon-add',

//为按钮绑定单击事件

handler:**function**(){

**$("#mytable").datagrid("insertRow",{// 插入数据**

**index:0,//在第一行插入数据 在哪插入**

**row:{}//空行**

**});**

**$("#mytable").datagrid("beginEdit",0);//行号 索引号**

**myIndex = 0;//用于表示正编辑的行的索引**

}

},

{text:'删除',iconCls:'icon-remove',handler:**function**(){

//获得选中的行对象

**var rows = $("#mytable").datagrid("getSelections");//获得选中的行（得到的是json对象的数组）**

**if**(rows.length == 1){

**var** row = rows[0];

**//获得指定行对象的索引**

**myIndex = $("#mytable").datagrid("getRowIndex",row);//获得行索引**

}

**$("#mytable").datagrid("deleteRow",myIndex);**

//$.post();

}},

{text:'修改',iconCls:'icon-edit',handler:**function**(){

//获得选中的行对象

**var** rows = $("#mytable").datagrid("getSelections");

**if**(rows.length == 1){

**var** row = rows[0];

//获得指定行对象的索引

myIndex = $("#mytable").datagrid("getRowIndex",row);

}

**$("#mytable").datagrid("beginEdit",myIndex);**

}},

{text:'保存',iconCls:'icon-save',handler:**function**(){

**$("#mytable").datagrid("endEdit",myIndex);**

}}

],

//显示分页条

pagination:**true**,

pageList:[3,5,7,10],

//数据表格提供的用于监听结束编辑事件

onAfterEdit:**function**(index,data,changes){

console.info(data);

$.post();

}

});

});

</script>

# 总结：

1．**easyUI中都有的特点：写属性设置的时候一般都是写成json键值对的形式（属性：属性值） 属性之间用，分开 如果有多个属性中还有子属性就用{属性：属性值，属性：属性值}**

**2.数据表格编辑功能是以列为单位。但是具体的操作（增删改都是以列为单位）**

**添加（插入一行）**

**3.无论是增删改最后都需要点击选中行保存**

**4. myIndex用于表示编辑行的索引**

**获得方式：**

**var rows = $("#mytable").datagrid("getSelections");//获得选中的行（得到的是json对象的数组）**

**if**(rows.length == 1){

**var** row = rows[0];

**//获得指定行对象的索引**

**myIndex = $("#mytable").datagrid("getRowIndex",row);//获得行索引**

}

**插入数据**

**$("#mytable").datagrid("insertRow",{// 插入数据**

**index:0,//在第一行插入数据 在哪插入**

**row:{}//空行**

**});**

**保存行**

**$("#mytable").datagrid("endEdit",myIndex);//保存**

**开始编辑行**

**$("#mytable").datagrid("beginEdit",myIndex);//修改**

删除行

**$("#mytable").datagrid("deleteRow",myIndex); //删除**

设置可编辑的列

**editor:{**

**type:'numberbox',//只能输入数字**

**options:{**

**required: true;**

**}//基本的设置**

**}**

//**使用spring 整合dao层 （不使用别的什么工具或是框架）**

**List<Customer> list = jdbcTemplate.query(sql, new RowMapper<Customer>(){**

**public Customer mapRow(ResultSet rs, int arg1) throws SQLException {**

**int id = rs.getInt("id");//根据字段名称从结果集中获取对应的值**

**String name = rs.getString("name");**

**String station = rs.getString("station");**

**String telephone = rs.getString("telephone");**

**String address = rs.getString("address");**

**String decidedzone\_id = rs.getString("decidedzone\_id");**

**return new Customer(id, name, station, telephone, address, decidedzone\_id);**

**}**

**},telephone); //返回一个对象的集合**

**String decidedzoneId = jdbcTemplate.queryForObject(sql, String.class, address);**

**//返回某个字段**

移开就会触发的事件

**$("input[name=telephone]").blur(function(){**