**HR设计规格说明书**

（V1.0）

**XXXX软件技术有限公司**

2015年4月

**目 录**

[1 概述 4](#_Toc417564316)

[2 预期读者 4](#_Toc417564317)

[3 功能架构 4](#_Toc417564318)

[4 人员信息 5](#_Toc417564319)

[4.1 功能需求 5](#_Toc417564320)

[4.2 开发思路 5](#_Toc417564321)

[4.3 实现步骤 6](#_Toc417564322)

[4.3.1 前置条件 6](#_Toc417564323)

[4.3.2 创建工程 8](#_Toc417564324)

[4.3.3 创建主从表功能 9](#_Toc417564325)

[4.3.4 扩展开发 13](#_Toc417564326)

[4.3.5 测试运行 33](#_Toc417564327)

[4.4 注意事项 34](#_Toc417564328)

[5 考勤管理 35](#_Toc417564329)

[5.1 功能需求 35](#_Toc417564330)

[5.2 开发思路 35](#_Toc417564331)

[5.3 实现步骤 35](#_Toc417564332)

[5.3.1 前置条件 35](#_Toc417564333)

[5.3.2 创建单表操作功能 36](#_Toc417564334)

[5.3.3 扩展开发 37](#_Toc417564335)

[5.3.4 测试运行 46](#_Toc417564336)

[5.4 注意事项 48](#_Toc417564337)

[6 加班申请 49](#_Toc417564338)

[6.1 功能需求 49](#_Toc417564339)

[6.2 开发思路 49](#_Toc417564340)

[6.3 实现步骤 50](#_Toc417564341)

[6.3.1 前置条件 50](#_Toc417564342)

[6.3.2 创建单表操作功能模型 50](#_Toc417564343)

[6.3.3 扩展开发 52](#_Toc417564344)

[6.3.4 测试运行 62](#_Toc417564345)

[6.4 注意事项 66](#_Toc417564346)

[7 请假申请 66](#_Toc417564347)

[7.1 功能需求 66](#_Toc417564348)

[7.2 开发思路 67](#_Toc417564349)

[7.3 实现步骤 67](#_Toc417564350)

[7.3.1 前置条件 67](#_Toc417564351)

[7.3.2 创建单表操作模型 68](#_Toc417564352)

[7.3.3 扩展开发 69](#_Toc417564353)

[7.3.4 测试运行 79](#_Toc417564354)

[7.4 注意事项 82](#_Toc417564355)

[8 出差报销 83](#_Toc417564356)

[8.1 功能需求 83](#_Toc417564357)

[8.2 开发思路 83](#_Toc417564358)

[8.3 实现步骤 84](#_Toc417564359)

[8.3.1 前置条件 84](#_Toc417564360)

[8.3.2 创建主从表模型 85](#_Toc417564361)

[8.3.3 扩展开发 88](#_Toc417564362)

[8.3.4 测试运行 94](#_Toc417564363)

[8.4 注意事项 98](#_Toc417564364)

[9 薪资管理 98](#_Toc417564365)

[9.1 功能需求 98](#_Toc417564366)

[9.2 开发思路 99](#_Toc417564367)

[9.3 实现步骤 99](#_Toc417564368)

[9.3.1 前置条件 99](#_Toc417564369)

[9.3.2 创建单表操作模型 100](#_Toc417564370)

[9.3.3 扩展开发 102](#_Toc417564371)

[9.3.4 测试运行 115](#_Toc417564372)

[9.4 注意事项 118](#_Toc417564373)

# 概述

人力资源部是对企业中各类人员形成的资源进行管理的部门，通过实施人力资源管理，真正实现全面提升[企业核心竞争力](http://baike.baidu.com/view/1273948.htm)之目的，支撑和保障企业的可持续发展与永续经营，并实现最大化赢利。

AEAI HR人力资源管理系统主要用于对企业内部人员相关信息进行管理，包括人员信息、考勤管理、薪资管理、加班申请、请假申请、出差报销等。

# 预期读者

1. 数通畅联公司新员工。
2. 人力资源系统选型人员。

# 总体设计

## 技术架构

ｍｏｂｉｌｅ

**ＰＣ**

**ＡＥＡＩＨＲ人力资源管理系统**

**ＡＥＡＩ　ＤＰ**

**开发平台**

**人员信息**

**考勤管理**

**加班申请**

开发

**请假申请**

**出差报销**

**薪资管理**

## 功能架构

AEAI HR人力资源管理系统整体功能架构如下图所示：

AEAI HR系统业务功能包括6个模块，分别是人员信息（对公司员工信息的管理）、考勤管理（员工出勤率管理）、薪资管理（员工薪资发放、涨幅管理）、加班申请（员工加班记录，审核记录）、请假申请（员工请假记录，审核记录）、出差报销（员工出差后进行报销费用使用，审核记录）。

# 功能设计

## 人员信息

### 功能需求

1. 需要实现人力资源部相关人员对公司人员信息的录入、存储、编辑、删除。
2. 新员工信息录入hr\_employee中的同时添加到系统用户security\_user中，同时状态设置为无效，更新hr\_employee中的数据时需要同步到security\_user。
3. 添加人员的同时需要选择一个部门（如不填写，表单提交不成功）。
4. 除限制部门之外的人员需要得到权限才可以登录，
5. 没有权限的用户登录后只能看到自己的信息
6. 没有权限的用户登录后只能进行查看和复制操作。

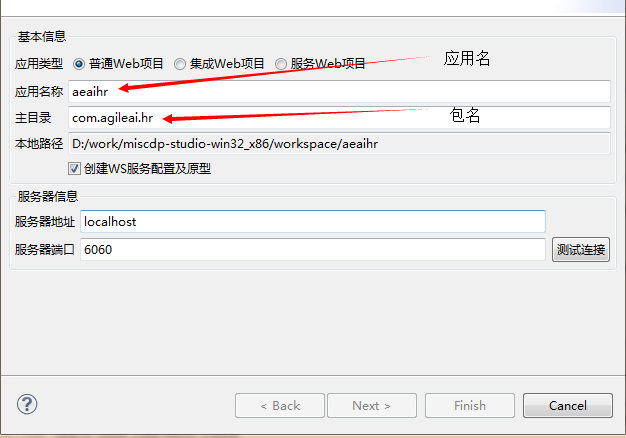
### 开发思路

1. 使用主从表管理功能模型，其中一个主表hr\_employee，三个从表hr\_education,hr\_experience,hr\_work\_performance,采用listDetailMode编辑模式，以及baseTabed功能模板。
2. 首先，在HrEmployee.xml文件中加入添加新用户到security\_user表中的sql语句，以及更新时同步的sql语句，然后通过在接口实现类HrEmployeeManageImpl.java中添加两个方法createMasterRecord和updateMasterRecord，用来添加/更新新用户到系统中。
3. 使用树形选择功能模型，使用数据库中的security\_user表，在HrEmployeeManageEdit.jsp页面调整部门的相应地方，添加js代码。
4. 管理员登陆在 功能管理->安全管理 中进行权限的设置，即可以看到全部人员信息的权限（一般为人力资源部的人员）。
5. 新建一个包com.agileai.hr.common，在包中新建一个类PrivilegeHelper.java,写一个用来判断登录用户是否属于得到权限的群组/角色，在HrEmployee.xml中加入添加用户到security\_user\_group\_rel的sql语句，在接口实现类HrEmployeeManageImpl.java中添加让hr\_employee中录入的新用户加入到security\_user\_group\_rel表中的方法，在HrEmployeeManageListHandler中添加一个方法用来判断是否为设权限的群组的用户，如果是不做约束，如果不是，获取到该用户的编号，只查询该编号的信息并显示。
6. 在前一步的HrEmployeeManageListHandler中的方法中加入语句，在登录的同时会进行一次判断，那么就需要在HrEmployeeManageList.jsp页面的新增、编辑、删除这三个按钮上判断条件。

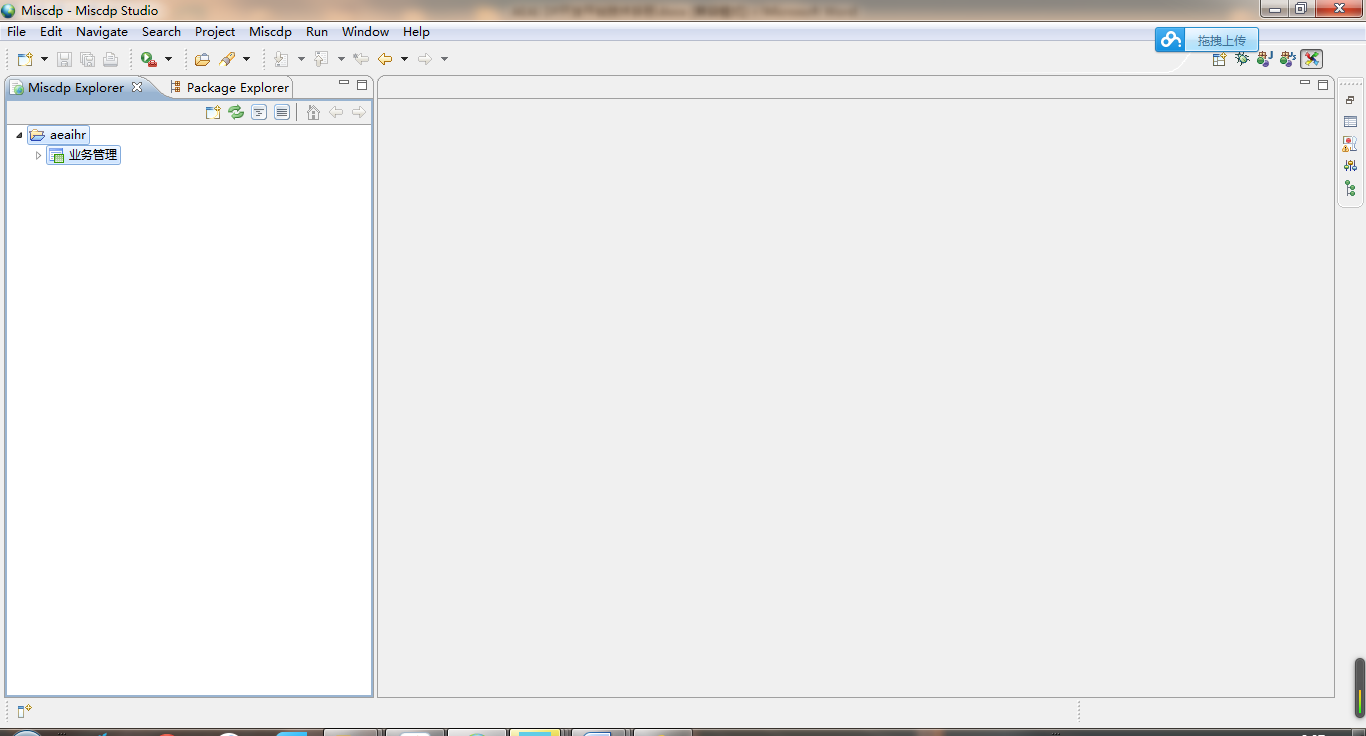
### 扩展设计

1. 创建工程

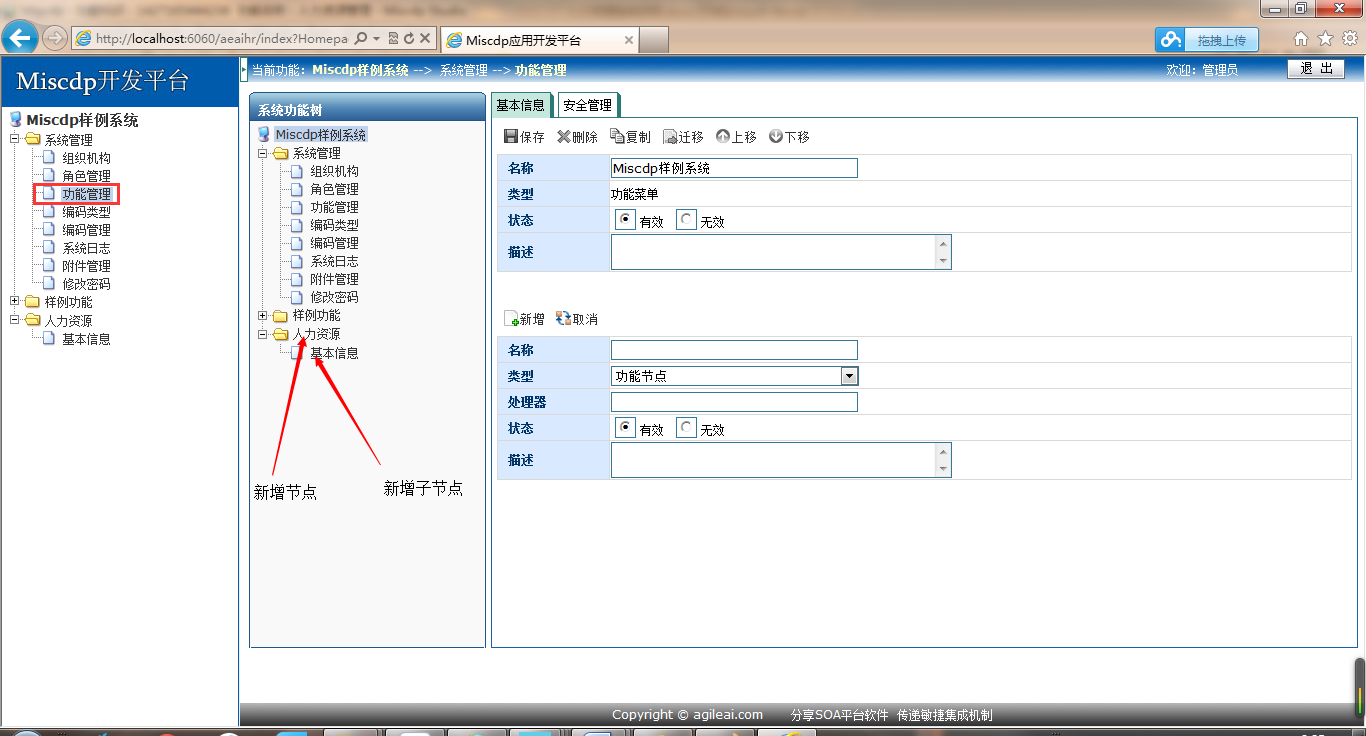
在Miscdp Exporler视图的右上角，点击，显示如下界面：



点击“测试连接”，测试通过后，Next按钮变为可用状态，修改DabaseName名为新建的数据库名aeaihr，“测试连接”成功后，进行“初始化数据”，初始化数据成功后完成工程创建。



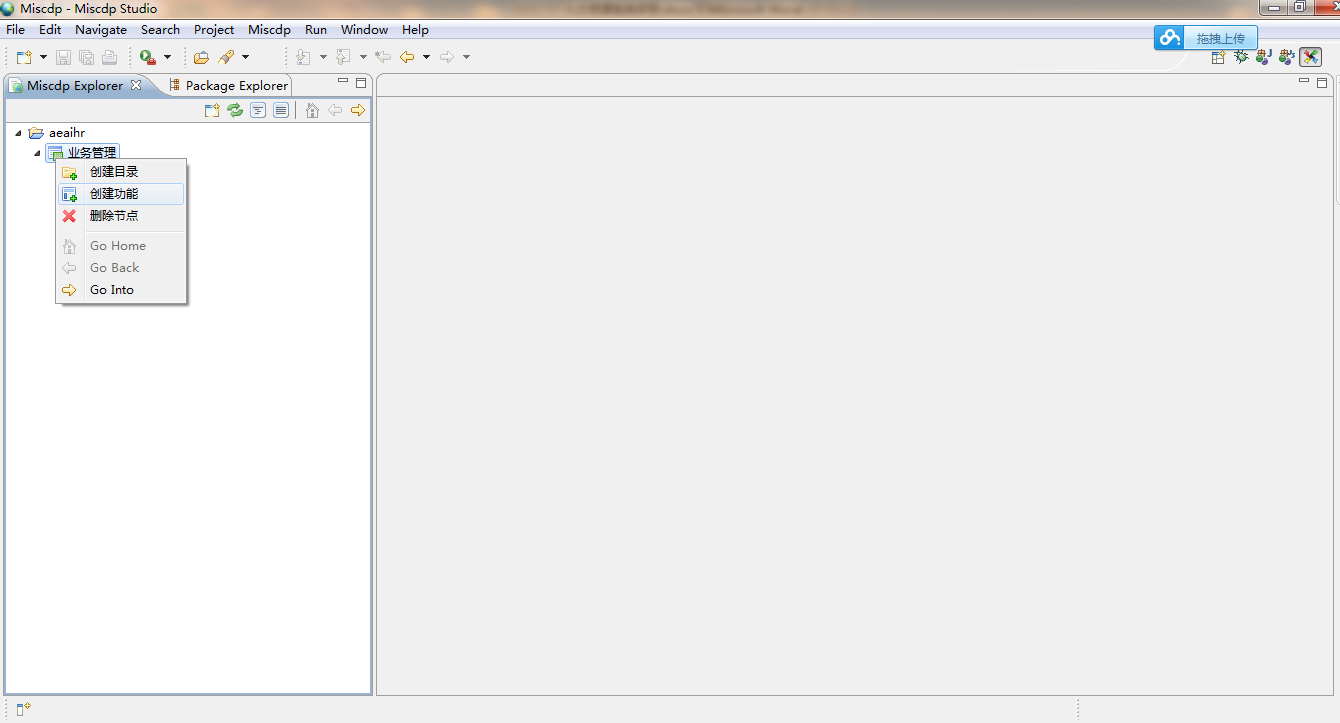
右键工程名“浏览运行”，用admin登录，在功能管理中新增节点“人力资源”以及子节点“基本信息”



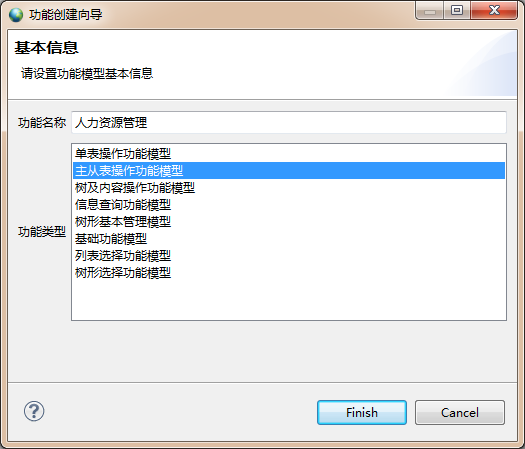
处理器可以再sqlmap的HandlerContext.xml文件中找到

1. 创建主从表功能

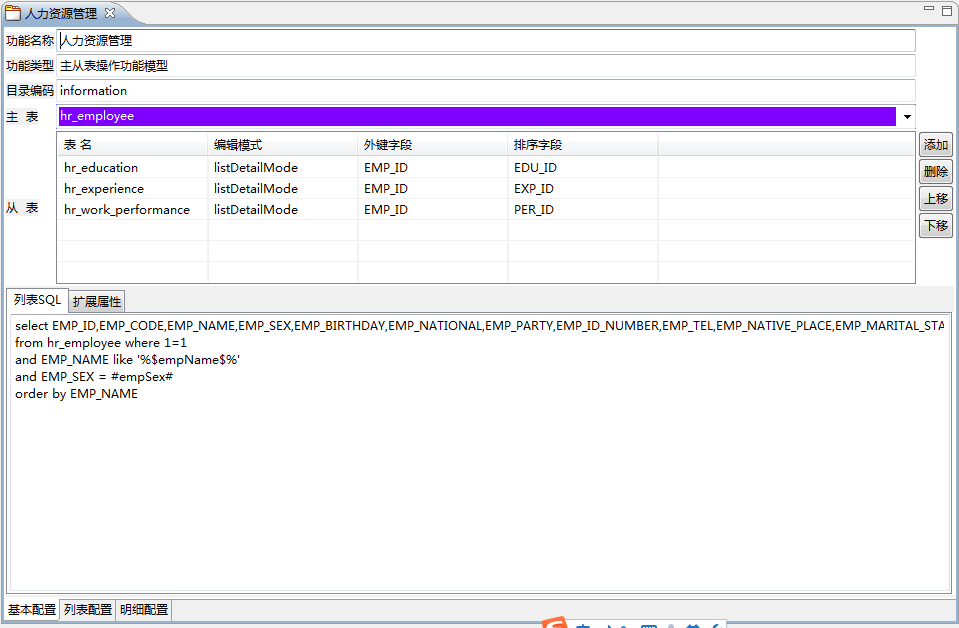
在业务管理上右键创建功能:



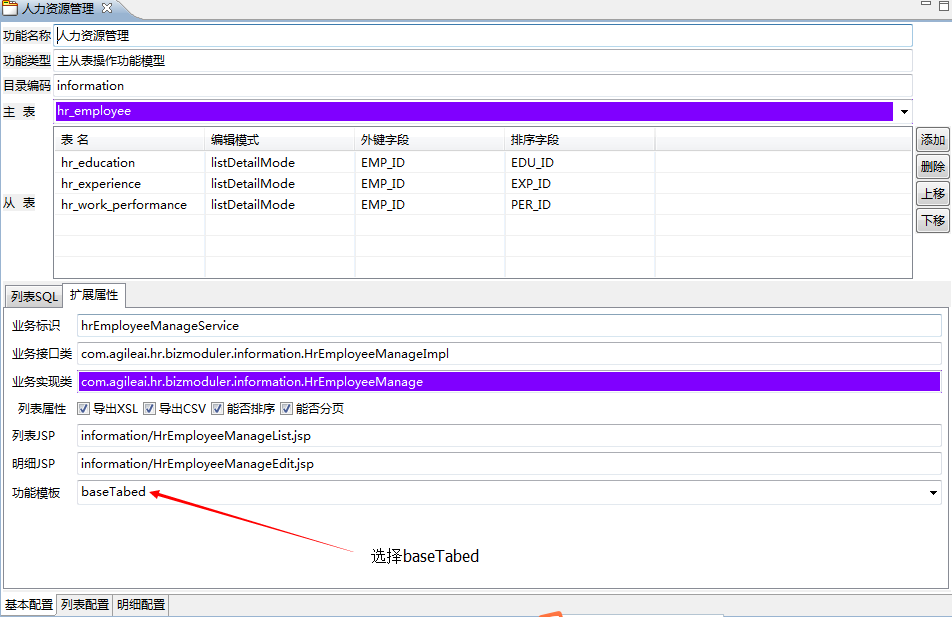
功能名称为人力资源管理，功能类型为主从表操作功能模型



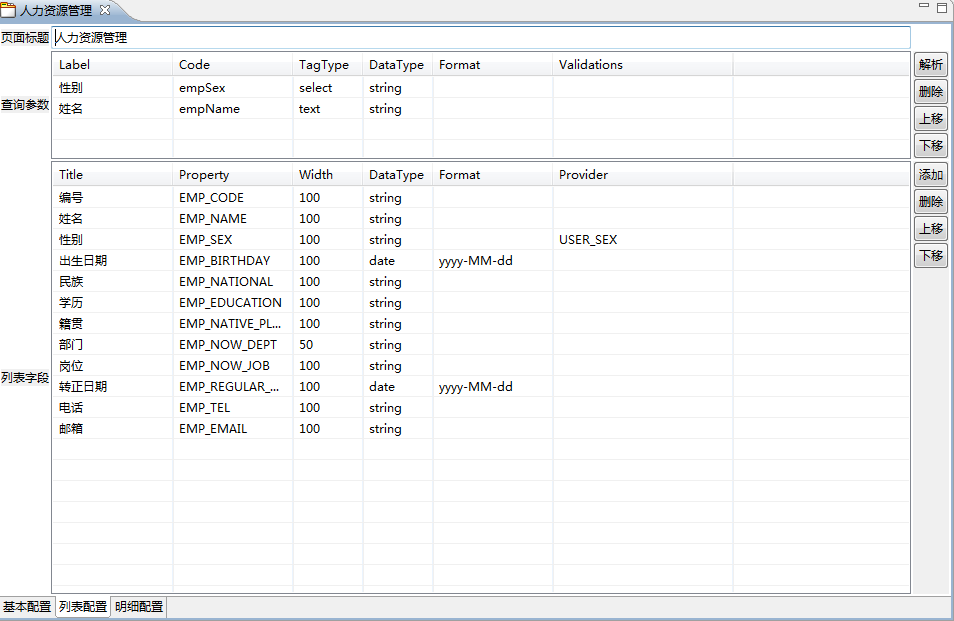
目录编码填写 information，主表采用hr\_employee，从表中的编辑模式采用listDetailMode，外键字段都选择EMP\_ID，排序字段使用各表中的主键字段，填写完后初始化，在列表sql中加入语句，如图



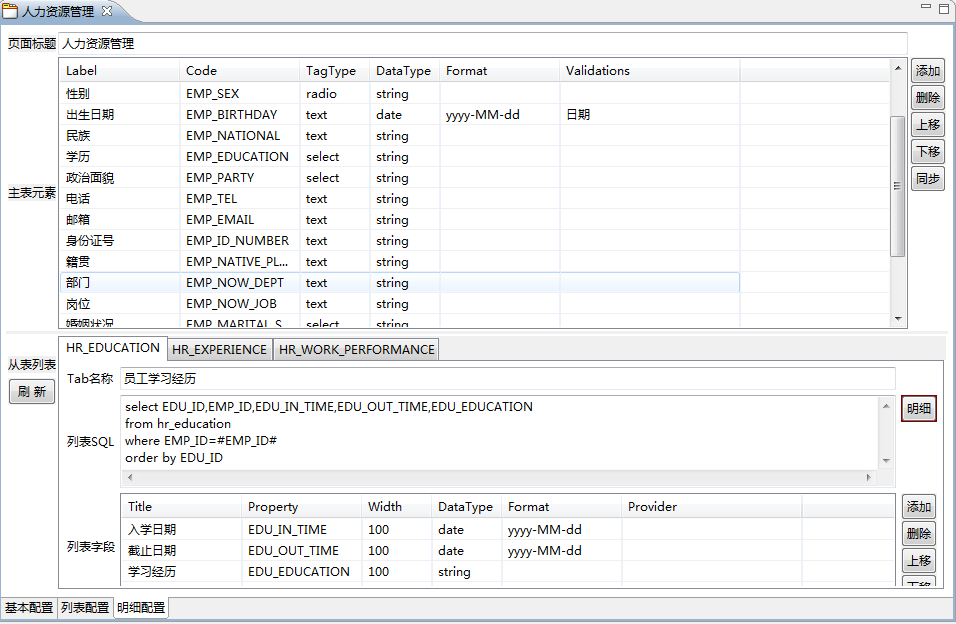
另外，在扩展属性中的功能模板选项一定要选择baseTabed



列表配置如下图：



明细配置如图所示：



在从表列表中的“明细”菜单中一定要记得同步。

用主从表模板只是用来创建一个大体的框架，有一些字段可能一开始的时候没有添加进去，需要在后面的操作中自己添加进去，减少对自动创建模板的依赖。

1. 扩展开发
2. 新增用户到EMP表中时自动存到USER表中，同时USER中的状态为无效。

需要在接口实现类HrEmployeeManageImpl中增加语句：

**public** **void** createMasterRecord(DataParam param) {

String statementId = sqlNameSpace+"."+"insertMasterRecord";

processDataType(param, tableName);

processPrimaryKeys(tableName,param);

**this**.daoHelper.insertRecord(statementId, param);

statementId = sqlNameSpace+"."+"insertSecurityUserRecord";

DataParam userParam = **new** DataParam();

String userId = KeyGenerator.*instance*().genKey();

userParam.put("USER\_ID",userId);

userParam.put("USER\_CODE",param.get("EMP\_CODE"));

userParam.put("USER\_NAME",param.get("EMP\_NAME"));

String encPassword = CryptionUtil.*md5Hex*(param.get("EMP\_CODE"));

userParam.put("USER\_PWD",encPassword);

userParam.put("USER\_SEX",param.get("EMP\_SEX"));

String strDesc = "普通用户";

userParam.put("USER\_DESC",strDesc);

String strState = "0";

userParam.put("USER\_STATE",strState);

String strSort = "3";

userParam.put("USER\_SORT",strSort);

userParam.put("USER\_MAIL",param.get("EMP\_EMAIL"));

userParam.put("USER\_PHONE",param.get("EMP\_TEL"));

**this**.daoHelper.insertRecord(statementId, userParam);

}

更新时需要同步到USER表中：

**public** **void** updateMasterRecord(DataParam param) {

String statementId = sqlNameSpace+"."+"updateMasterRecord";

processDataType(param, tableName);

**this**.daoHelper.updateRecord(statementId, param);

statementId = sqlNameSpace+"."+"updateSecurityUserRecord";

DataParam userParam = **new** DataParam();

userParam.put("USER\_CODE",param.get("EMP\_CODE"));

userParam.put("USER\_NAME",param.get("EMP\_NAME"));

String encPassword = CryptionUtil.*md5Hex*(param.get("EMP\_CODE"));

userParam.put("USER\_PWD",encPassword);

userParam.put("USER\_SEX",param.get("EMP\_SEX"));

String strDesc = "普通用户";

userParam.put("USER\_DESC",strDesc);

String strState = "0";

userParam.put("USER\_STATE",strState);

String strSort = "3";

userParam.put("USER\_SORT",strSort);

userParam.put("USER\_MAIL",param.get("EMP\_EMAIL"));

userParam.put("USER\_PHONE",param.get("EMP\_TEL"));

**this**.daoHelper.updateRecord(statementId, param);

}

1. 新增用户时选择的部门也需要同步到USER表中

在接口实现类HrEmployeeManageImpl中的createMasterRecord方法中增加语句：

statementId = sqlNameSpace+"."+"insertSecurityUserGroupRelRecord";

userParam.put("GRP\_ID", param.get("GRP\_ID"));

userParam.put("USER\_ID",param.get("USER\_ID"));

**this**.daoHelper.insertRecord(statementId, userParam);

1. 核准按钮的添加，在sql中新增语句，在handler中的语句添加，在接口实现类中的语句添加，在jsp页面的语句增加。

在sql中新增语句，用来执行核准操作：

<update id="approveRecord" parameterClass="com.agileai.domain.DataParam">

update hr\_employee

set

EMP\_CODE=#EMP\_CODE:VARCHAR#,EMP\_NAME=#EMP\_NAME:VARCHAR#,EMP\_SEX=#EMP\_SEX:CHAR#,EMP\_BIRTHDAY=#EMP\_BIRTHDAY:DATE#,EMP\_NATIONAL=#EMP\_NATIONAL:VARCHAR#,EMP\_ID\_NUMBER=#EMP\_ID\_NUMBER:VARCHAR#,EMP\_TEL=#EMP\_TEL:VARCHAR#,EMP\_EMAIL=#EMP\_EMAIL:VARCHAR#,EMP\_NATIVE\_PLACE=#EMP\_NATIVE\_PLACE:VARCHAR#,EMP\_PARTY=#EMP\_PARTY:CHAR#,EMP\_MARITAL\_STATUS=#EMP\_MARITAL\_STATUS:VARCHAR#,EMP\_EDUCATION=#EMP\_EDUCATION:VARCHAR#,EMP\_NOW\_DEPT=#EMP\_NOW\_DEPT:VARCHAR#,EMP\_NOW\_JOB=#EMP\_NOW\_JOB:VARCHAR#,EMP\_REFERENCE\_TIME=#EMP\_REFERENCE\_TIME:DATE#,EMP\_INDUCTION\_TIME=#EMP\_INDUCTION\_TIME:DATE#,EMP\_REGULAR\_TIME=#EMP\_REGULAR\_TIME:DATE#,EMP\_BASIC=#EMP\_BASIC:DECIMAL#,EMP\_PERFORMANCE=#EMP\_PERFORMANCE:DECIMAL#,EMP\_SUBSIDY=#EMP\_SUBSIDY:DECIMAL#,EMP\_STATE=#EMP\_STATE:VARCHAR#

where EMP\_ID=#EMP\_ID#

</update>

在接口实现类HrEmployeeManageImpl中增加语句：

**public** **void** approveRecord(DataParam param) {

String statementId = sqlNameSpace + "." + "approveRecord";

processDataType(param, tableName);

**this**.daoHelper.updateRecord(statementId, param);

}

在HrEmployeeManage中增加语句：

**public** **interface** HrEmployeeManage

**extends** MasterSubService {

**void** approveRecord(DataParam param);

}

在edit的jsp中增加一个核准按钮：

<td align=*"center"*class=*"bartdx"*onclick="stateApprove();" onmouseover="onMover(this);" onmouseout="onMout(this);"><input value=*"&nbsp;"*type=*"button"* class=*"approveImgBtn"* id=*"approveImgBtn"* title=*"核准"* />核准</td>

同时在js中增加语句

**function** stateApprove(){

doSubmit({actionType:'approve'});

}

在list的jsp中增加核准按钮：

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"R"* align=*"center"* onclick="doRequest('approveRequest')"><input id=*"approveImgBtn"* value=*"&nbsp;"* title=*"核准"* type=*"button"* class=*"approveImgBtn"* />核准</td>

在editHandler中增加语句：

**if** ("approve".equals(operaType)) {

**if** (!isReqRecordOperaType(operaType)) {

DataRow record = getService().getMasterRecord(param);

**this**.setAttributes(record);

}

}

在listhandler中增加方法：

**public** ViewRenderer doApproveRequestAction(DataParam param) {

storeParam(param);

**return** **new** DispatchRenderer(getHandlerURL(**this**.editHandlerClazz) + "&"

+ OperaType.*KEY* + "=approve&comeFrome=approve");

}

1. 登录时判断是否为人力负责人，同时对按钮进行控制。

首先新建一个包，包名为com.agileai.hr.common，在包中新建一个撞门用来判断权限的类PrivilegeHelper：

**public** **class** PrivilegeHelper {

**private** User user = **null**;

**public** PrivilegeHelper(User user){

**this**.user = user;

}

**public** **boolean** isHRMASTER(){

**boolean** result = **false**;

List<Role> roleList = user.getRoleList();

**for** (**int** i=0;i < roleList.size();i++){

Role role = roleList.get(i);

**if** ("HR\_MASTER".equals(role.getRoleCode())){

result = **true**;

**break**;

}

}

**return** result;

}

}

这个方法用来判断是否为人力负责人

在listhandler中的prepareDisplay方法中 增加一个判断:

**public** ViewRenderer prepareDisplay(DataParam param) {

User user = (User) **this**.getUser();

PrivilegeHelper privilegeHelper = **new** PrivilegeHelper(user);

**if** (!privilegeHelper.isHRMASTER()) {

param.put("currentUserCode", user.getUserCode());

setAttribute("hasRight", **true**);

} **else** {

param.put("currentUserCode", "");

setAttribute("hasRight", **false**);

}

mergeParam(param);

initParameters(param);

**this**.setAttributes(param);

List<DataRow> rsList = getService().findMasterRecords(param);

**this**.setRsList(rsList);

processPageAttributes(param);

**return** **new** LocalRenderer(getPage());

}

在list的jsp页面使用hasRight对按钮进行控制：

<%**if** (!pageBean.getBoolValue("hasRight")){%>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"A"* align=*"center"* onclick="doRequest('insertRequest')"><input id=*"createImgBtn"* value=*"&nbsp;"* title=*"新增"* type=*"button"* class=*"createImgBtn"* />新增</td>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"E"* align=*"center"* onclick="doRequest('updateRequest')"><input id=*"editImgBtn"* value=*"&nbsp;"* title=*"编辑"* type=*"button"* class=*"editImgBtn"* />编辑</td>

<%} %>

<%**if**(!pageBean.getBoolValue("hasRight")){ %>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"R"* align=*"center"* onclick="doRequest('approveRequest')"><input id=*"approveImgBtn"* value=*"&nbsp;"* title=*"核准"* type=*"button"* class=*"approveImgBtn"* />核准</td>

<%}%>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"V"* align=*"center"* onclick="doRequest('viewDetail')"><input id=*"detailImgBtn"* value=*"&nbsp;"* title=*"查看"* type=*"button"* class=*"detailImgBtn"* />查看</td>

<%**if** (!pageBean.getBoolValue("hasRight")){%>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"D"* align=*"center"* onclick="doDelete($('#'+rsIdTagId).val());"><input id=*"deleteImgBtn"* value=*"&nbsp;"* title=*"删除"* type=*"button"* class=*"delImgBtn"* />删除</td>

<%} %>

1. List列表页面的按钮控制

通过jsp页面的js代码控制：

<script type="text/javascript">

**function** controlUpdateBtn(stateResult){

**if**(stateResult =='drafe'){

<%**if** (!pageBean.getBoolValue("hasRight")){%>

enableButton("editImgBtn");

enableButton("approveImgBtn");

<%}%>

enableButton("detailImgBtn");

enableButton("deleteImgBtn");

}

**if**(stateResult =='approved'){

<%**if** (!pageBean.getBoolValue("hasRight")){%>

enableButton("editImgBtn");

disableButton("approveImgBtn");

<%}%>

enableButton("detailImgBtn");

disableButton("deleteImgBtn");

}

}

</script>

在下面的点击记录的控制jsp代码：

<ec:row styleClass="odd" ondblclick="clearSelection();doRequest('viewDetail')" oncontextmenu="selectRow(this,{EMP\_ID:'${row.EMP\_ID}'});controlUpdateBtn('${row.EMP\_STATE}');refreshConextmenu()" onclick="selectRow(this,{EMP\_ID:'${row.EMP\_ID}'});controlUpdateBtn('${row.EMP\_STATE}');">

1. List的查询条件中增加EMP\_STATE的筛选条件。

List的jsp页面进行增加语句：

&nbsp;状态<select id=*"EMP\_STATE"* label=*"状态"* name=*"EMP\_STATE"* class=*"select"* onchange=*"doQuery()"*><%=pageBean.selectValue("EMP\_STATE")%></select>

List的handler页面增加语句：

**protected** **void** initParameters(DataParam param) {

initParamItem(param, "EMP\_STATE", "");

initParamItem(param, "empSex", "");

initParamItem(param, "empName", "");

}

**protected** **void** processPageAttributes(DataParam param) {

setAttribute("empSex",

FormSelectFactory.*create*("USER\_SEX")

.addSelectedValue(param.get("empSex")));

initMappingItem("EMP\_SEX",

FormSelectFactory.*create*("USER\_SEX").getContent());

initMappingItem("EMP\_EDUCATION",

FormSelectFactory.*create*("EMP\_EDUCATION").getContent());

initMappingItem("EMP\_STATE",

FormSelectFactory.*create*("EMP\_STATE").getContent());

setAttribute("EMP\_STATE", FormSelectFactory.*create*("EMP\_STATE")

.addSelectedValue(param.get("EMP\_STATE")));

}

1. 在进行双击查看的动作时，通过权限和基本信息状态来对按钮进行控制。

在Edit的handler页面增加判断：

**if** ("detail".equals(operaType)) {

User user = (User) **this**.getUser();

PrivilegeHelper privilegeHelper = **new** PrivilegeHelper(user);

**if** (privilegeHelper.isHRMASTER()){

setAttribute("doDetail", **true**);

setAttribute("doApprove", **false**);

}**else**{

setAttribute("doApprove", **false**);

}

**if** (isReqRecordOperaType(operaType)) {

DataRow record = getService().getMasterRecord(param);

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

**if**(!privilegeHelper.isHRMASTER()&&record.get("EMP\_STATE").equals("drafe")){

setAttribute("doDetail", **true**);

setAttribute("doApprove",**false**);

}**else** **if**(privilegeHelper.isHRMASTER()){

setAttribute("doDetail", **true**);

}**else**{

setAttribute("doDetail", **false**);

setAttribute("doApprove",**false**);

}

}

**this**.setAttributes(record);

}

}

在edit的jsp页面做按钮控制：

<tr>

<%**if**(pageBean.getBoolValue("doDetail")){ %>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* align=*"center"* onclick="enableSave()" ><input value=*"&nbsp;"* type=*"button"* class=*"editImgBtn"* id=*"modifyImgBtn"* title=*"编辑"* />编辑</td>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* align=*"center"* onclick="saveMasterRecord()"><input value=*"&nbsp;"* type=*"button"* class=*"saveImgBtn"* id=*"saveImgBtn"* title=*"保存"* />保存</td>

<%} %>

<%**if**(pageBean.getBoolValue("doApprove")){ %>

<td align=*"center"*class=*"bartdx"*onclick="stateApprove();" onmouseover="onMover(this);" onmouseout="onMout(this);"><input value=*"&nbsp;"*type=*"button"* class=*"approveImgBtn"* id=*"approveImgBtn"* title=*"核准"* />核准</td>

<%} %>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* align=*"center"* onclick="goToBack();"><input value=*"&nbsp;"* type=*"button"* class=*"backImgBtn"* title=*"返回"* />返回</td>

</tr>

对人员学习经历、工作经历、工作表现的按钮也做控制：

<tr>

<%**if**(pageBean.getBoolValue("doDetail")){ %>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* align=*"center"* onclick="insertSubRecordRequest('人员学习经历','HrEducationEditBox')"><input value=*"&nbsp;"* title=*"新增"* type=*"button"* class=*"createImgBtn"* />新增</td>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* align=*"center"* onclick="copySubRecordRequest('人员学习经历','HrEducationEditBox','EDU\_ID')"><input value=*"&nbsp;"* title=*"复制"* type=*"button"* class=*"copyImgBtn"* />复制</td>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* align=*"center"* onclick="viewSubRecordRequest('update','人员学习经历','HrEducationEditBox','EDU\_ID')"><input value=*"&nbsp;"* title=*"编辑"* type=*"button"* class=*"editImgBtn"* />编辑</td>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* align=*"center"* onclick="deleteSubRecord()"><input value=*"&nbsp;"* title=*"删除"* type=*"button"* class=*"delImgBtn"* />删除</td>

<%} %>

</tr>

另外两个模块做相同处理就不赘述了。

1. 在list和edit页面上都添加反核准按钮，用来将已核准状态撤销为草稿状态，便于用户自己去修改。
   1. 在list列表页面上添加反核准按钮

List的jsp页面加按钮：

<td onmouseover="onMover(this);" onmouseout="onMout(this);" align=*"center"* class=*"bartdx"* onclick="revokeApproval()" ><input value=*"&nbsp;"*type=*"button"* class=*"revokeApproveImgBtn"* id=*"revokeApproval"* title=*"反核准"* />反核准</td>

在sqlmap中添加对应的sql：

<update id="revokeApprovalRecord" parameterClass="com.agileai.domain.DataParam">

update hr\_employee

set

EMP\_STATE=#EMP\_STATE:VARCHAR#

where EMP\_ID=#EMP\_ID#

</update>

在接口实现类中添加方法：

@Override

**public** **void** revokeApprovalRecords(String empId) {

DataParam param = **new** DataParam("EMP\_ID", empId,"EMP\_STATE","drafe");

String statementId = sqlNameSpace+"."+"revokeApprovalRecord";

processDataType(param, tableName);

**this**.daoHelper.updateRecord(statementId, param);

在HrEmployeeManage类中，添加方法：

**void** revokeApprovalRecords(String empId);

在listhandler中添加pageAction方法 ：

@PageAction

**public** ViewRenderer revokeApproval(DataParam param) {

String empId = param.get("EMP\_ID");

getService().revokeApprovalRecords(empId);

**return** prepareDisplay(param);

}

* 1. 在edit编辑页面上添加反核准按钮

在edit的jsp页面添加按钮：

<td onmouseover="onMover(this);" onmouseout="onMout(this);" align=*"center"* class=*"bartdx"* onclick="revokeApproval()" ><input value=*"&nbsp;"*type=*"button"* class=*"revokeApproveImgBtn"* id=*"revokeApproval"* title=*"反核准"* />反核准</td>

在上面的js中添加对应的方法：

**function** revokeApproval(){

doSubmit({actionType:'revokeApproval'});

}

在edithandler中添加对应的pageAction：

@PageAction

**public** ViewRenderer revokeApproval(DataParam param) {

String empId = param.get("EMP\_ID");

getService().revokeApprovalRecords(empId);

**return** **new** RedirectRenderer(getHandlerURL(listHandlerClass));

}

1. 新增用户时，编号等必填的五项可填写、选择，且编号不能重复，有权限人员编辑操作时，除编号只读以外都可编辑，无权限人员编辑时，必填项都为只读。

首先在editHandler中做保存的action方法中添加语句：

**public** ViewRenderer doSaveMasterRecordAction(DataParam param) {

String operateType = param.get(OperaType.*KEY*);

String responseText = "fail";

**if** (OperaType.*CREATE*.equals(operateType)) {

String empCode = param.get("EMP\_CODE");

DataParam repeatParam = **new** DataParam("EMP\_CODE", empCode);

DataRow record = getService().getMasterRecord(repeatParam);

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

responseText = "repeat";

} **else** {

getService().createMasterRecord(param);

responseText = param.get(baseTablePK);

}

} **else** **if** (OperaType.*UPDATE*.equals(operateType)) {

getService().updateMasterRecord(param);

saveSubRecords(param);

responseText = param.get(baseTablePK);

}

在getMasterRecord中增加一个筛选条件：

<isNotEmpty prepend=*"and "* property=*"EMP\_CODE"*>

EMP\_CODE=#EMP\_CODE#

</isNotEmpty>

对edit的jsp中的js代码进行扩展：

**function** saveMasterRecord(){

**if** (validate()){

**if** (ele("currentSubTableId")){

**var** subTableId = $("#currentSubTableId").val();

**if** (!checkEntryRecords(subTableId)){

return;

}

}

showSplash();

postRequest('form1',{actionType:'saveMasterRecord',onComplete:**function**(responseText){

**if** ("fail" != responseText){

**if**("repeat"==responseText){

hideSplash();

writeErrorMsg('编号重复，请检查！');

}**else**{

$('#operaType').val('update');

$('#EMP\_ID').val(responseText);

doSubmit({actionType:'prepareDisplay'});

}

}**else**{

hideSplash();

writeErrorMsg('保存操作出错啦！');

}

}});

}

}

添加一个页面控制，用来控制必填项的只读与否：

在editHandler中的preparedisplay添加语句：

**if** ("insert".equals(operaType)) {

setAttribute("doDetail", **true**);

setAttribute("doApprove", **false**);

setAttribute("onlyRead", "");

}

**if** ("update".equals(operaType)) {

setAttribute("doApprove", **false**);

setAttribute("doDetail", **true**);

setAttribute("onlyRead", "readonly");

**if** (isReqRecordOperaType(operaType)) {

DataRow record = getService().getMasterRecord(param);

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

User user = (User) **this**.getUser();

PrivilegeHelper privilegeHelper = **new** PrivilegeHelper(user);

**if**(privilegeHelper.isHRMASTER()){

setAttribute("onlyRead", "");

}**else**{

setAttribute("onlyRead", "readonly");

}

}

**this**.setAttributes(record);

}

}

**if** ("detail".equals(operaType)) {

setAttribute("onlyRead", "readonly");

User user = (User) **this**.getUser();

PrivilegeHelper privilegeHelper = **new** PrivilegeHelper(user);

**if** (privilegeHelper.isHRMASTER()) {

setAttribute("doDetail", **true**);

setAttribute("doApprove", **false**);

} **else** {

setAttribute("doApprove", **false**);

}

**if** (isReqRecordOperaType(operaType)) {

DataRow record = getService().getMasterRecord(param);

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

**if** (!privilegeHelper.isHRMASTER()) {

setAttribute("onlyRead", "readonly");

**if** (record.get("EMP\_STATE").equals("drafe")) {

setAttribute("doDetail", **true**);

setAttribute("doApprove", **false**);

}

} **else** **if** (privilegeHelper.isHRMASTER()

&& record.get("EMP\_STATE").equals("drafe")) {

setAttribute("onlyRead", "");

setAttribute("isappover","apporver");

setAttribute("doDetail", **true**);

setAttribute("doApprove", **true**);

} **else** **if** (privilegeHelper.isHRMASTER()

&& record.get("EMP\_STATE").equals("approved")) {

setAttribute("doDetail", **false**);

setAttribute("doApprove", **true**);

} **else** {

setAttribute("doDetail", **false**);

setAttribute("doApprove", **false**);

}

}

**this**.setAttributes(record);

}

}

在edit的jsp中的五个必填项添加添加控制：

<th width=*"100"* nowrap>编号</th>

<td><input id=*"EMP\_CODE"* label=*"编号"* name=*"EMP\_CODE"* type=*"text"* value=*"*<%=pageBean.inputValue("EMP\_CODE")%>*"* <%**if**("readonly".equals(pageBean.getAttribute("onlyRead"))){ %>readonly=*"readonly"*<%} %> <%**if**("codereadonly".equals(pageBean.getAttribute("codeReadonly"))){ %>readonly=*"readonly"*<%} %> size=*"24"* class=*"text"* />

</td>

<th width=*"100"* nowrap>姓名</th>

<td><input id=*"EMP\_NAME"* label=*"姓名"* name=*"EMP\_NAME"* type=*"text"* value=*"*<%=pageBean.inputValue("EMP\_NAME")%>*"* <%**if**("readonly".equals(pageBean.getAttribute("onlyRead"))){ %>readonly=*"readonly"*<%} %> size=*"24"* class=*"text"* />

</td>

<th width=*"100"* nowrap>性别</th>

<td>&nbsp;<%=pageBean.selectRadio("EMP\_SEX")%>

</td>

<th width=*"100"* nowrap>部门</th>

<td><input id=*"EMP\_NOW\_DEPT"* name=*"EMP\_NOW\_DEPT"* type=*"text"* value=*"*<%=pageBean.inputValue("EMP\_NOW\_DEPT")%>*"* size=*"24"* class=*"text"* readonly=*"readonly"* /><input type=*"hidden"* label=*"部门"* id=*"GRP\_ID"* name=*"GRP\_ID"* value=*"*<%=pageBean.inputValue("GRP\_ID")%>*"* readonly=*"*<%=pageBean.getAttribute("onlyRead") %>*"* />

<%**if**(!"readonly".equals(pageBean.getAttribute("onlyRead"))){ %>

<img id=*"grpIdSelectImage"* src=*"images/sta.gif"* width=*"16"* height=*"16"* onclick="openGrpIdBox()"/>

<%} %>

</td>

<th width=*"100"* nowrap>岗位</th>

<td><input id=*"EMP\_NOW\_JOB"* label=*"岗位"* name=*"EMP\_NOW\_JOB"* type=*"text"* value=*"*<%=pageBean.inputValue("EMP\_NOW\_JOB")%>*"* <%**if**("readonly".equals(pageBean.getAttribute("onlyRead"))){ %>readonly=*"readonly"*<%} %> size=*"24"* class=*"text"* />

</td>

在新增时编号可填写，在编辑以及查看时，编号只读：

在editHandler中再添加一个按钮控制：

**if** ("update".equals(operaType)) {

setAttribute("doApprove", **false**);

setAttribute("doDetail", **true**);

setAttribute("onlyRead", "readonly");

**if** (isReqRecordOperaType(operaType)) {

DataRow record = getService().getMasterRecord(param);

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

User user = (User) **this**.getUser();

PrivilegeHelper privilegeHelper = **new** PrivilegeHelper(user);

**if**(privilegeHelper.isHRMASTER()){

setAttribute("onlyRead", "");

setAttribute("codeReadonly", "codereadonly");

}**else**{

setAttribute("onlyRead", "readonly");

}

}

**this**.setAttributes(record);

}

}

在查看操作时，若登录人为业务核准人，且信息状态为草稿：

**if** (privilegeHelper.isHRMASTER()

&& record.get("EMP\_STATE").equals("drafe")) {

setAttribute("onlyRead", "");

setAttribute("codeReadonly", "codereadonly");

setAttribute("isappover","apporver");

setAttribute("doDetail", **true**);

setAttribute("doApprove", **true**);

}

在edit的jsp中添加控制：

<tr>

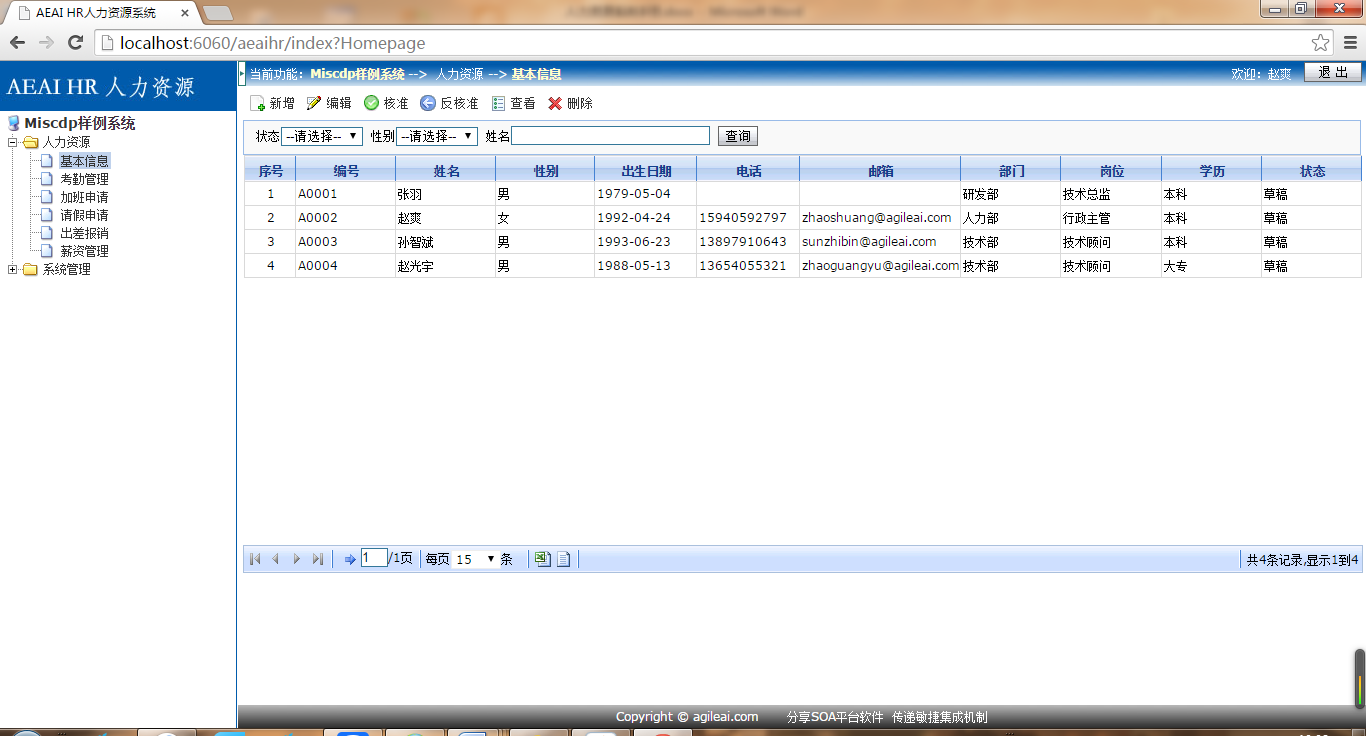
<th width=*"100"* nowrap>编号</th>

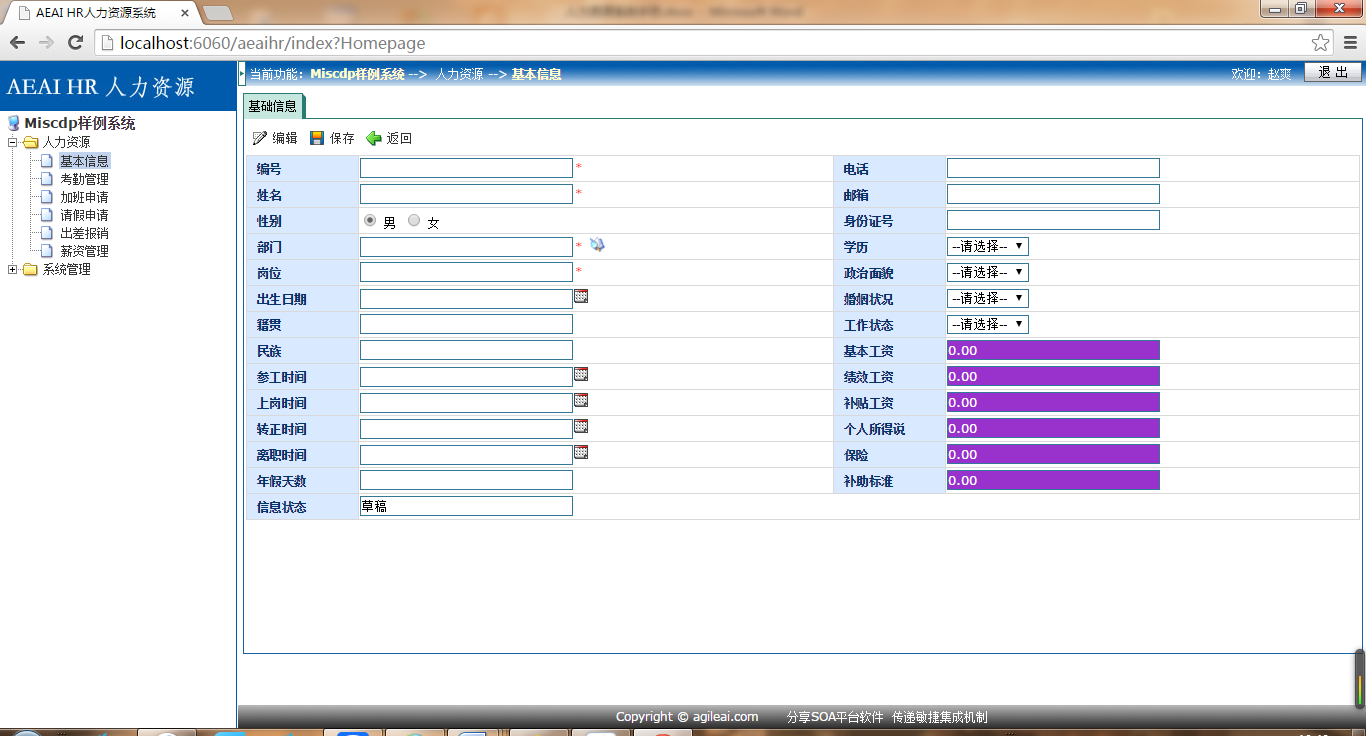
<td><input id=*"EMP\_CODE"* label=*"编号"* name=*"EMP\_CODE"* type=*"text"* value=*"*<%=pageBean.inputValue("EMP\_CODE")%>*"* <%**if**("readonly".equals(pageBean.getAttribute("onlyRead"))){ %>readonly=*"readonly"*<%} %> <%**if**("codereadonly".equals(pageBean.getAttribute("codeReadonly"))){ %>readonly=*"readonly"*<%} %> size=*"24"* class=*"text"* />

</td>

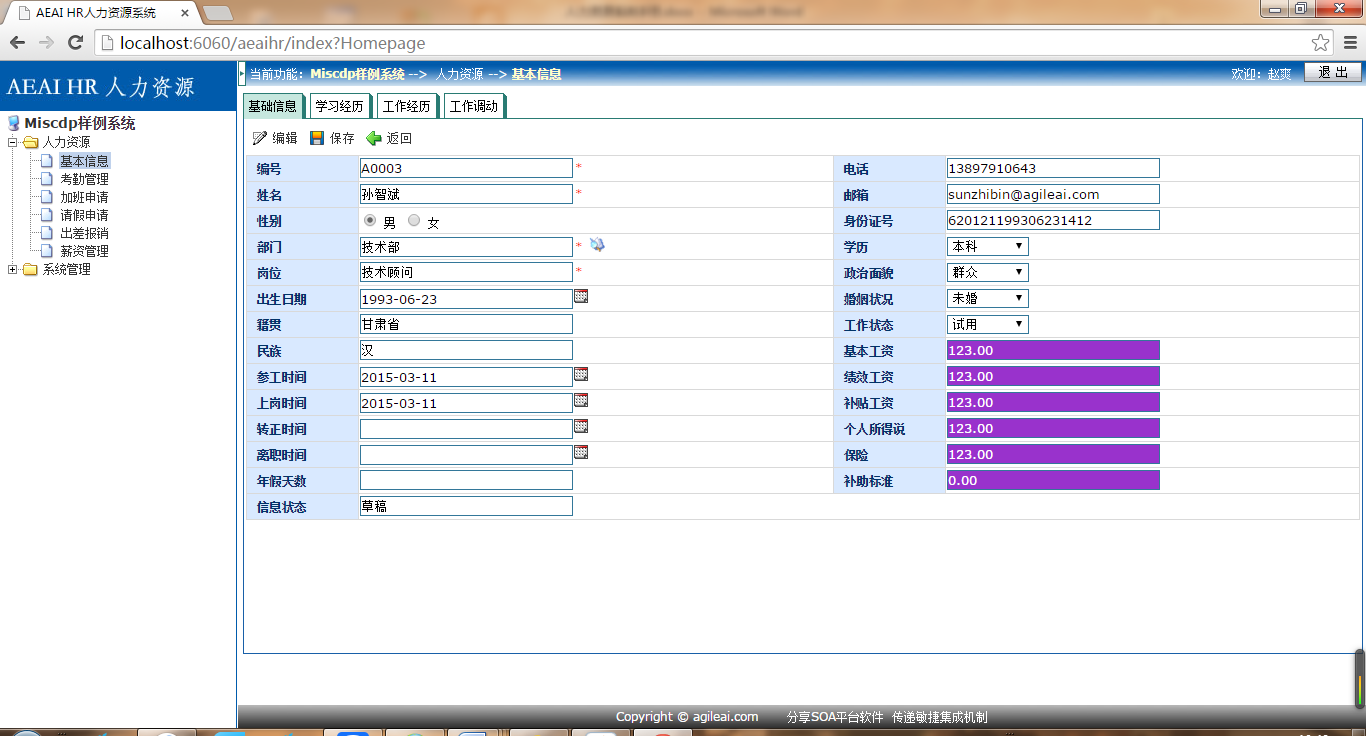
1. 测试运行

测试运行，效果如下图：

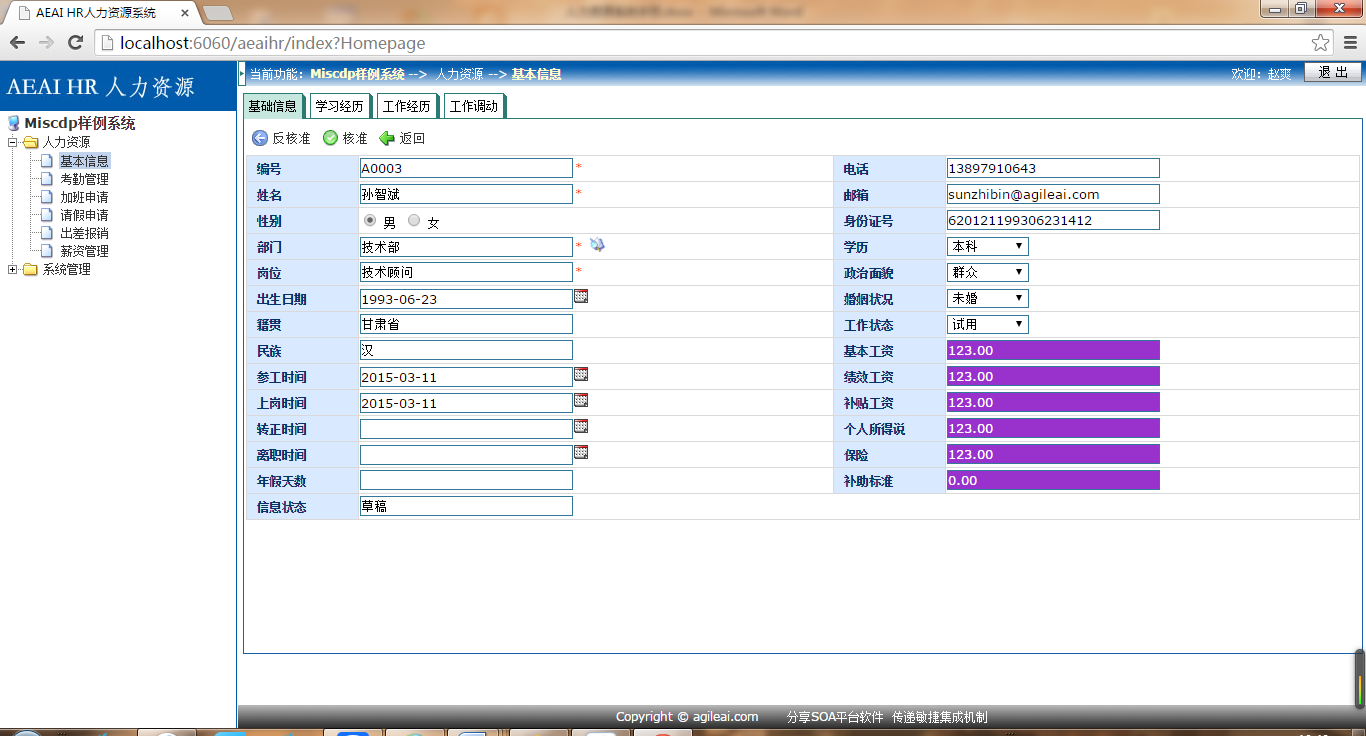




基础信息保存成功后，可以完善其他信息：







### 注意事项

1. 创建工程时一定要初始化数据表
2. 主从表的编辑模式要选择为listDetailMode，扩展属性中的功能模板选项一定要选择baseTabed
3. 在接口实现类HremployeeManageImpl.java中增加新用户的时候ID字段要选择自动添加的方法，密码需要使用md5加密，密码即用户的编号
4. com.agileai.hr.common包中的PrivilegeHelper.java中的方法可以根据需求的不同来改变
5. 草稿状态时，双击查看的时候，是可以做编辑操作的，核准后只有人力负责人才能对已核准的记录操作，没权限的用户只能查看。
6. 在edit页面点击反核准按钮的时候需要返回到列表页面。

## 考勤管理

### 功能需求

1. 需要实现公司员工每天的签到、签退功能，以及查询之前签到的信息。
2. 员工登录自动获取员工姓名到考勤表中。
3. 员工每天只能执行一次签到和签退。
4. 员工查询之前的签到记录时只能查看，签到、签退按钮为不可用。

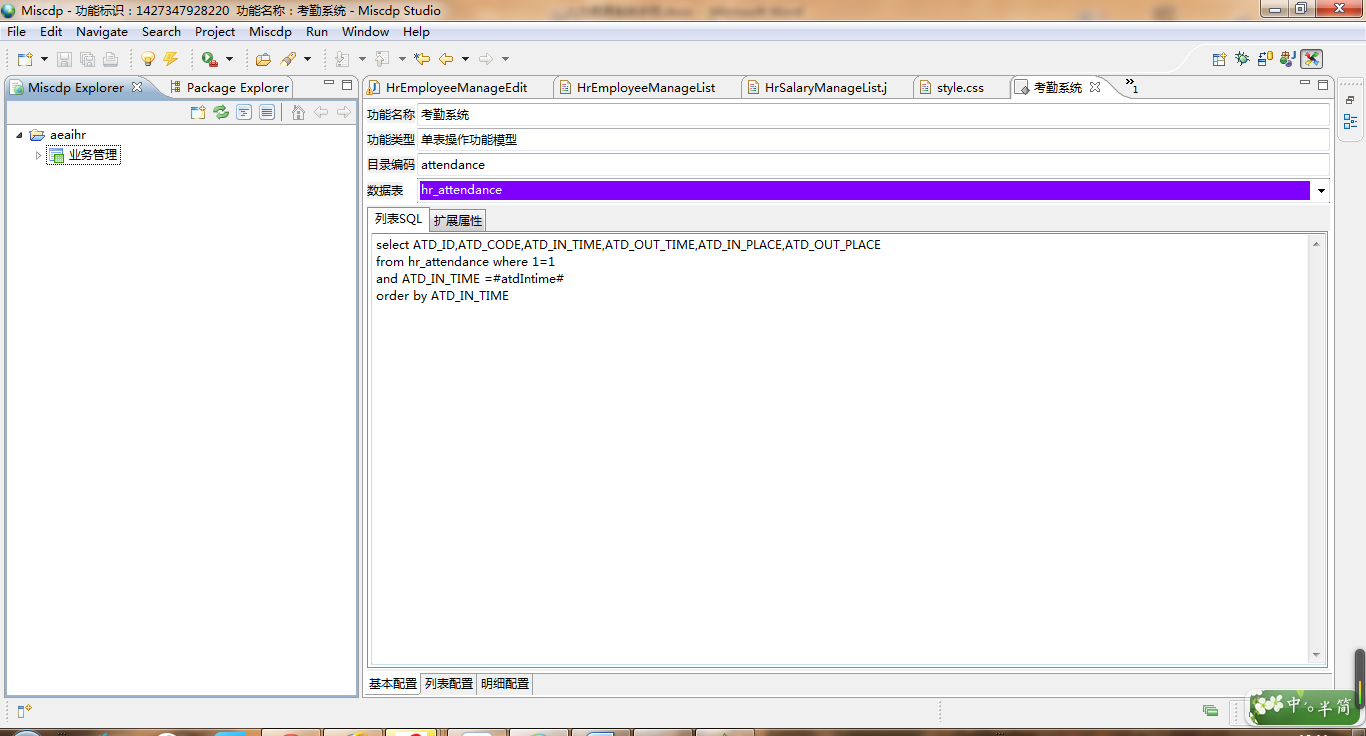
### 开发思路

1. 使用单表操作功能模型，其中有一个考勤表hr\_attendance
2. 员工登录时获取到员工的编号，通过编号在security\_user表中查询到员工的姓名，然后在考勤表的姓名中显示
3. 如果签到记录为空，可以进行签到，签到成功且只能签到一次，如果签退记录为空可以进行一次更新，用来添加签退记录，当记录不为空时，签到、签退按钮不可用。
4. 做一个判断，当员工查询以前的签到记录时，签到、签退按钮变为不可用状态。

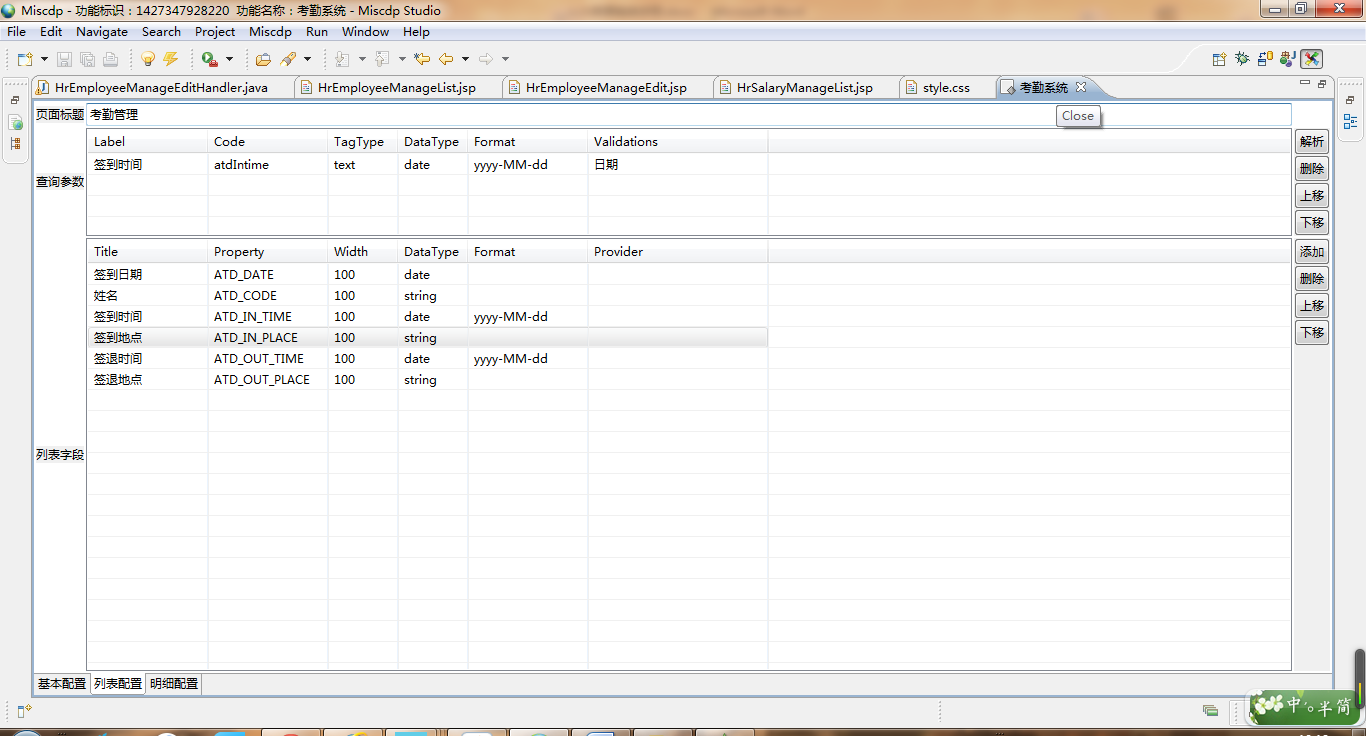
### 扩展设计

1. 创建单表操作功能

右键‘业务管理’🡪‘创建功能’🡪‘单表操作功能模型’🡪名称‘考勤管理’🡪目录编码填写‘attendance’数据表选择‘hr\_attendance’🡪初始化数据🡪填写列表sql，如图：



列表配置如图：



明细配置如图：



上述步骤做完后，生成代码。

1. 扩展开发
2. List列表页面只有签到、签退两个按钮，且一个用户一天只能签到一次，签退一次，进入考勤管理时签到按钮可用，签退按钮不可用；签到之后，签退按钮才可以使用；签退之后两个按钮都不可用。

在list的jsp页面做修改：

<tr>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"A"* align=*"center"* onclick="showEditBox('insertRequest')"><input id=*"singInImgBtn"* value=*"&nbsp;"* title=*"签到"* type=*"button"* class=*"createImgBtn"* />签到</td>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"E"* align=*"center"* onclick="showEditBox('updateRequest')"><input id=*"singOutImgBtn"* value=*"&nbsp;"* title=*"签退"* type=*"button"* class=*"editImgBtn"* />签退</td>

</tr>

对应的js也需要修改：

<%

**if**(!pageBean.getBoolValue("canSignIn")){

%>

disableButton("singInImgBtn");

<%}%>

<%

**if**(!pageBean.getBoolValue("canSignOut")){

%>

disableButton("singOutImgBtn");

<%}%>

在list的handler页面的preparedisplay方法中添加语句：

**public** ViewRenderer prepareDisplay(DataParam param) {

mergeParam(param);

initParameters(param);

String paramteDate = param.get("adtDate");

setAttribute("canSignIn", **true**);

setAttribute("canSignOut", **true**);

String currentDate = DateUtil.*getDateByType*(DateUtil.*YYMMDD\_HORIZONTAL*);

**if** (paramteDate.equals(currentDate)) {

User user = (User) **this**.getUser();

DataParam queryParam = **new** DataParam();

queryParam.put("currentDate", DateUtil.*getDateByType*(

DateUtil.*YYMMDD\_HORIZONTAL*, **new** Date()));

queryParam.put("currentUser", user.getUserId());

DataRow record = getService().getRecord(queryParam);

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

**if** (record.get("ATD\_IN\_TIME") != **null**) {

setAttribute("canSignIn", **false**);

}

**if** (record.get("ATD\_OUT\_TIME") != **null**) {

setAttribute("canSignOut", **false**);

}

} **else** {

setAttribute("canSignOut", **false**);

}

} **else** {

setAttribute("canSignIn", **false**);

setAttribute("canSignOut", **false**);

}

**this**.setAttributes(param);

List<DataRow> rsList = getService().findRecords(param);

**this**.setRsList(rsList);

processPageAttributes(param);

**return** **new** LocalRenderer(getPage());

}

1. 进入编辑页面时自动获取用户名，以及当前日期，当前时间，签到时只能看到需要签到的信息，签退时才能看到全部信息。

在edit的handler中的preparedisplay方法中添加语句：

**public** ViewRenderer prepareDisplay(DataParam param) {

String operaType = param.get(OperaType.*KEY*);

User user = (User) **this**.getUser();

param.put("currentDate",

DateUtil.*getDateByType*(DateUtil.*YYMMDD\_HORIZONTAL*, **new** Date()));

param.put("currentUser", user.getUserId());

**if** (isReqRecordOperaType(operaType)) {

DataRow record = getService().getRecord(param);

**this**.setAttributes(record);

}

**this**.setOperaType(operaType);

processPageAttributes(param);

**return** **new** LocalRenderer(getPage());

}

页面的显示方法processPageAttributes中添加语句用来显示用户名以及判断:

**protected** **void** processPageAttributes(DataParam param) {

User user = (User) **this**.getUser();

**this**.setAttribute("USER\_ID\_NAME",

**this**.getAttribute("USER\_ID\_NAME", user.getUserName()));

**this**.setAttribute("USER\_ID",

**this**.getAttribute("USER\_ID", user.getUserId()));

**if** (**this**.getAttribute("ATD\_IN\_TIME") == **null**){

**this**.setAttribute("ATD\_IN\_TIME", **new** Date());

setAttribute("doSignIn", **true**);

}**else**{

**this**.setAttribute("ATD\_OUT\_TIME", **new** Date());

setAttribute("doSignIn", **false**);

}

String adtDate = (String) **this**.getAttribute("ATD\_DATE",DateUtil.*getDateByType*(DateUtil.*YYMMDD\_HORIZONTAL*, **new** Date()));

**this**.setAttribute("ATD\_DATE", adtDate);

}

在sqlmap中的sql语句需要做关联查询：

<select id="findRecords" parameterClass="com.agileai.domain.DataParam"

resultClass="com.agileai.domain.DataRow">

select a.ATD\_ID,a.ATD\_DATE,a.USER\_ID,b.USER\_NAME as USER\_ID\_NAME,

a.ATD\_IN\_TIME,a.ATD\_OUT\_TIME,a.ATD\_IN\_PLACE,a.ATD\_OUT\_PLACE

from hr\_attendance a left join security\_user b on a.USER\_ID =

b.USER\_ID

where 1=1

<isNotEmpty prepend=*" "* property=*"adtDate"*>

and ATD\_DATE=#adtDate#

</isNotEmpty>

order by ATD\_IN\_TIME

</select>

<select id="getRecord" parameterClass="com.agileai.domain.DataParam"

resultClass="com.agileai.domain.DataRow">

select a.ATD\_ID,a.ATD\_DATE,a.USER\_ID,b.USER\_NAME as USER\_ID\_NAME,

a.ATD\_IN\_TIME,a.ATD\_OUT\_TIME,a.ATD\_IN\_PLACE,a.ATD\_OUT\_PLACE

from hr\_attendance a left join security\_user b on a.USER\_ID =

b.USER\_ID

where 1=1

<isNotEmpty prepend=*"and "* property=*"currentUser"*>

b.USER\_ID=#currentUser#

</isNotEmpty>

<isNotEmpty prepend=*"and "* property=*"currentDate"*>

ATD\_DATE=#currentDate#

</isNotEmpty>

</select>

Edit的jsp中需要修改姓名处的语句：

<tr>

<th width=*"100"* nowrap>姓名</th>

<td><input name="USER\_ID\_NAME" type="text" class="text" id="USER\_ID\_NAME"

value=*"*<%=pageBean.inputValue("USER\_ID\_NAME")%>*"* size=*"24"*

readonly=*"readonly"* label=*"姓名"* /> <input id=*"USER\_ID"* label=*"姓名"*

name=*"USER\_ID"* type=*"hidden"* value=*"*<%=pageBean.inputValue("USER\_ID")%>*"*

size=*"24"* class=*"text"* />

</td>

</tr>

在jsp中需要加判断，用来显示edit的显示：

<%

**if** (!pageBean.getBoolValue("doSignIn")) {

%>

<tr>

<th width=*"100"* nowrap>签退时间</th>

<td><input name=*"ATD\_OUT\_TIME"* type=*"text"* class=*"text"*

id=*"ATD\_OUT\_TIME"* value=*"*<%=pageBean.inputTime("ATD\_OUT\_TIME")%>*"*

size=*"24"* readonly=*"readonly"* label=*"签退时间"* /></td>

</tr>

<tr>

<th width=*"100"* nowrap>签退地点</th>

<td><input id="ATD\_OUT\_PLACE" label="签退地点" name="ATD\_OUT\_PLACE"

type=*"text"* value=*"*<%=pageBean.inputValue("ATD\_OUT\_PLACE")%>*"*

size=*"24"* class=*"text"* /></td>

</tr>

<%}%>

在下面的js中也需要做判断：

<%

**if** (!pageBean.getBoolValue("doSignIn")) {

%>

requiredValidator.add("ATD\_OUT\_PLACE");

<%}%>

1. 查询条件增加两个🡪上一天、下一天。

在list页面增加按钮：

&nbsp;<input type=*"submit"* name=*"button"* id=*"button"* value=*"上一天"* class=*"formbutton"* onclick="doSubmit({actionType:'beforeDay'})" />

&nbsp;<input type=*"submit"* name=*"button"* id=*"button"* value=*"下一天"* class=*"formbutton"* onclick="doSubmit({actionType:'nextDay'})" />

在list的handler中增加两个按钮的pageAction

@PageAction

**public** ViewRenderer beforeDay(DataParam param) {

String atdDate = param.get("adtDate");

Date selectDate = DateUtil.*getDate*(atdDate);

Date beforeDate = DateUtil.*getDateAdd*(selectDate, DateUtil.*DAY*, -1);

String targetDate = DateUtil.*getDateByType*(DateUtil.*YYMMDD\_HORIZONTAL*,

beforeDate);

param.put("adtDate", targetDate);

**return** prepareDisplay(param);

}

@PageAction

**public** ViewRenderer nextDay(DataParam param) {

String atdDate = param.get("adtDate");

Date selectDate = DateUtil.*getDate*(atdDate);

Date beforeDate = DateUtil.*getDateAdd*(selectDate, DateUtil.*DAY*, +1);

String targetDate = DateUtil.*getDateByType*(DateUtil.*YYMMDD\_HORIZONTAL*,

beforeDate);

param.put("adtDate", targetDate);

**return** prepareDisplay(param);

}

1. List列表页面中的记录直接显示，由于信息很少所以不需要双击查看。

在list的jsp页面进行修改：

<ec:rowstyle Class=*"odd"*

onclick="selectRow(this,{ATD\_ID:'${row.ATD\_ID}'})">

1. 点击签到、签退时，弹出框显示信息。

对list的jsp页面中的按钮进行修改：

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"A"* align=*"center"* onclick="showEditBox('insertRequest')"><input id=*"singInImgBtn"* value=*"&nbsp;"* title=*"新增"* type=*"button"* class=*"createImgBtn"* />签到</td>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"E"* align=*"center"* onclick="showEditBox('updateRequest')"><input id=*"singOutImgBtn"* value=*"&nbsp;"* title=*"编辑"* type=*"button"* class=*"editImgBtn"* />签退</td>

都是用showEditBox方法，在上面的js中加入该方法：

**var** editBox;

**function** showEditBox(operType){

clearSelection();

**if** (!editBox){

editBox = **new** PopupBox('editBox','详细信息',{size:'big',height:'400px',top:'30px'});

}

**var** url = "<%=pageBean.getHandlerURL()%>&actionType="+operType;

editBox.sendRequest(url);

}

这里需要对edit的jsp页面的按钮进行修改：

<table border=*"0"* cellpadding=*"0"* cellspacing=*"1"*>

<tr>

<td onmouseover="onMover(this);" onmouseout="onMout(this);"

class="bartdx" align="center"

onclick="doSubmit({actionType:'save',target:'\_parent'})"><input value=*"&nbsp;"*

type="button" class="saveImgBtn" id="saveImgBtn" title="确认" />确认</td>

<td onmouseover="onMover(this);" onmouseout="onMout(this);"

class=*"bartdx"* align=*"center"* onclick="parent.PopupBox.closeCurrent()">

<input value=*"&nbsp;"* type=*"button"* class=*"closeImgBtn"* title=*"关闭"* />关闭</td>

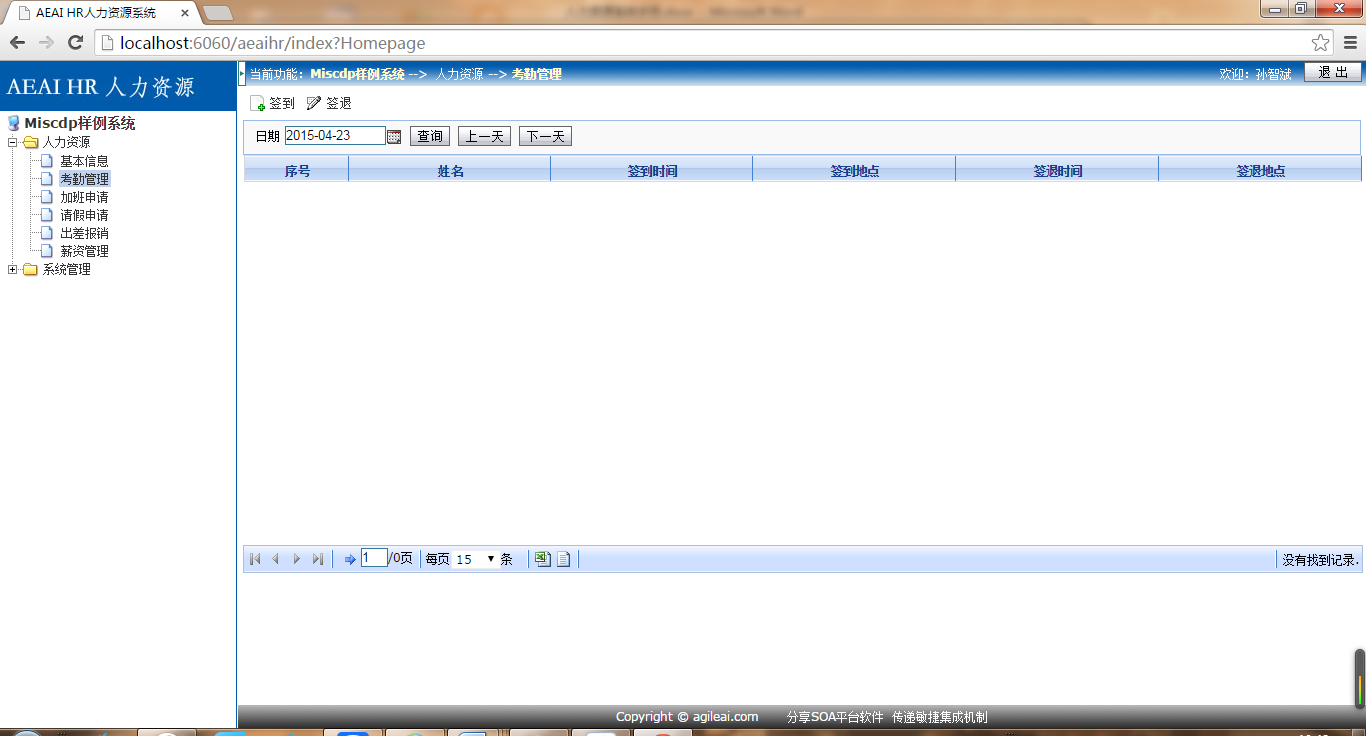
</tr>

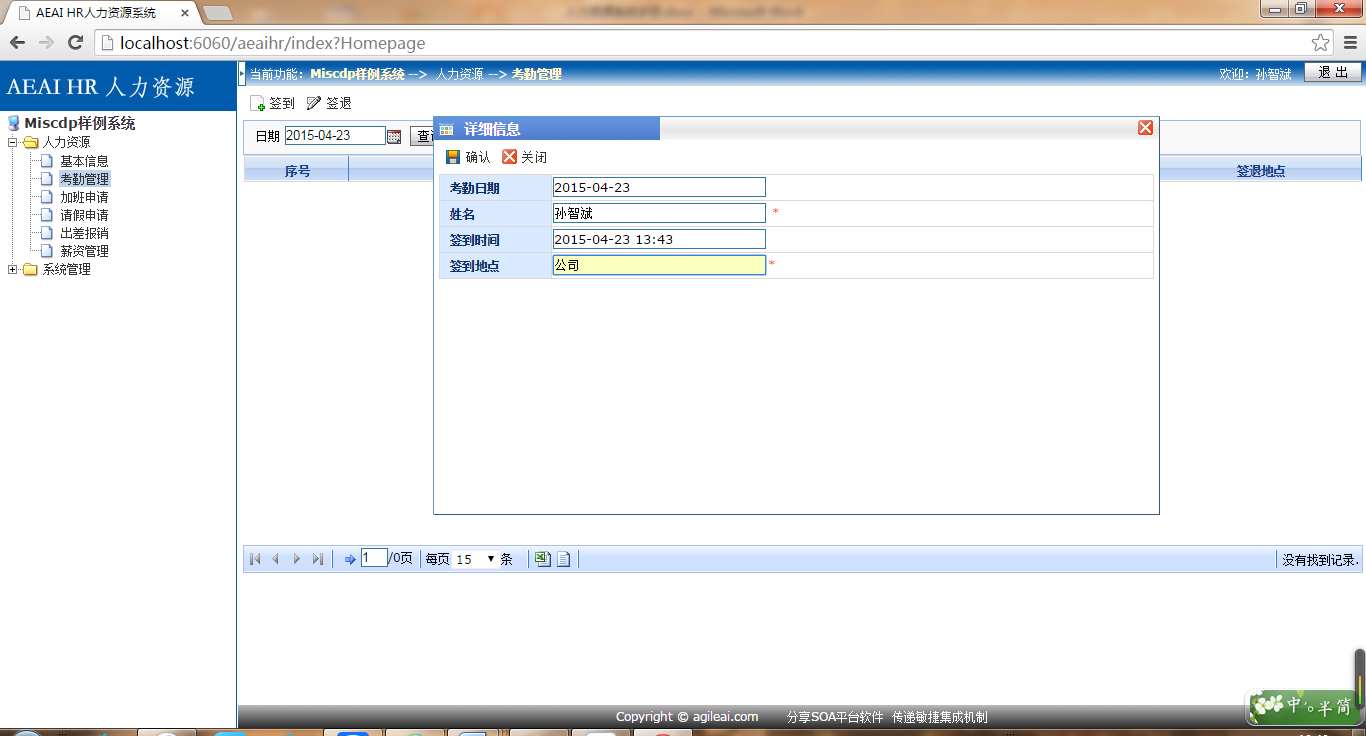
</table>

1. 测试运行

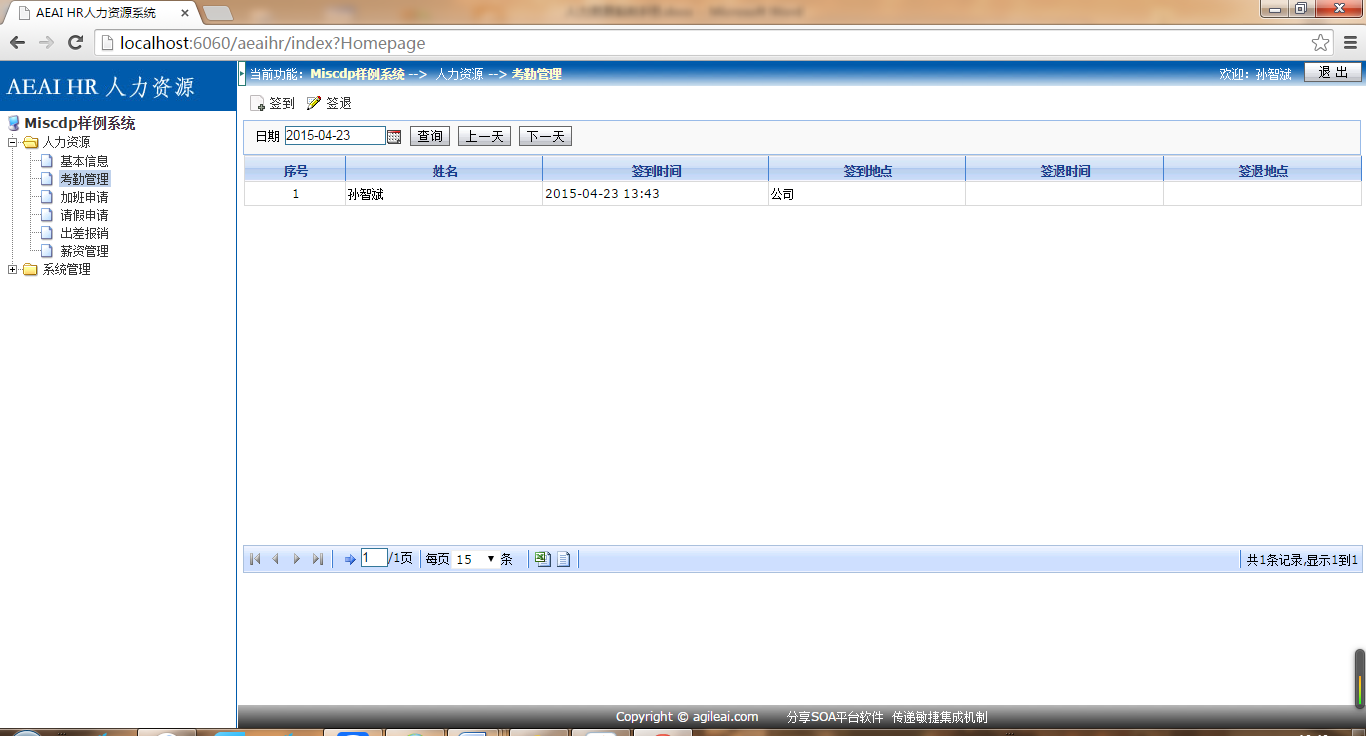
测试运行，效果如下图：

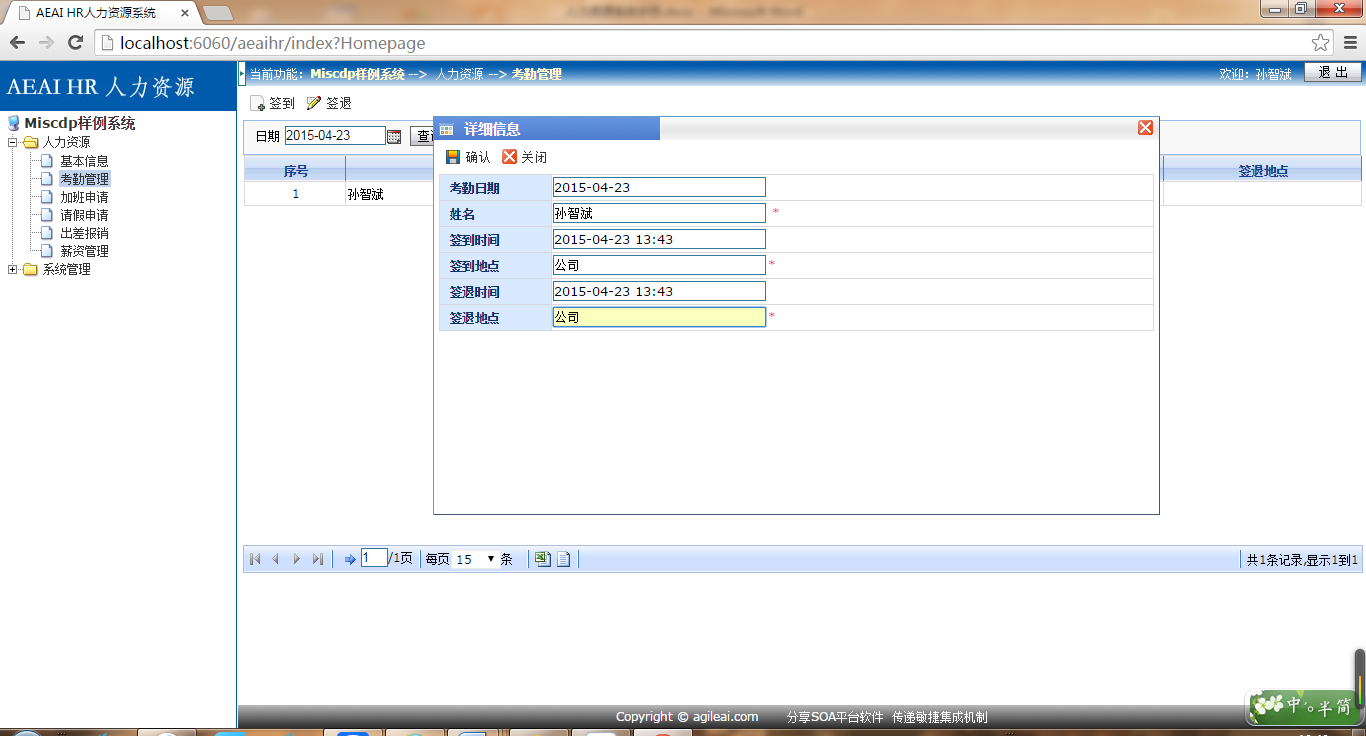
签到前：



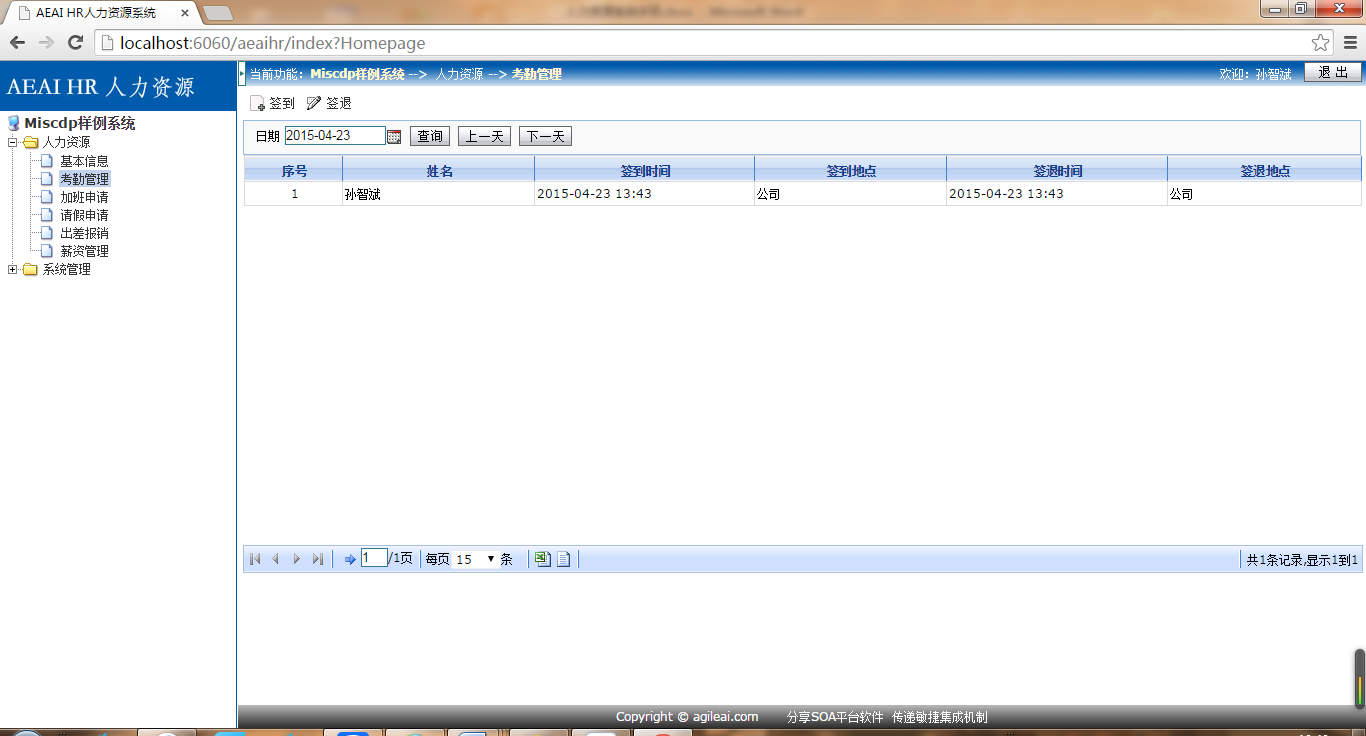


签到后：





签退后：



### 注意事项

1. 登录时获取到的USER\_ID存入隐藏域，通过USER\_ID查到的姓名存入USER\_ID\_NAME，要注意在list页面上的显示。
2. 签到，签退按钮控制不仅要在jsp中添加，还需要在js中添加。

## 加班申请

### 功能需求

1. 实现员工的加班的申请，需要自己填写加班地点、加班时长（0.5天/1天）（下拉框）
2. 在列表页面有申请、编辑/提交、核准、查看、删除五个按钮，查询条件为状态、时间段以及员工姓名
3. 在编辑页面有保存、提交、反提交、核准、返回四个按钮，点击申请和编辑按钮进入编辑页面时不会有反提交、核准按钮，点击核准进入编辑页面不会有保存和提交按钮，点击查看按钮时只有返回按钮
4. 若不是业务核准人那么核准按钮隐藏，点击申请按钮进入编辑页面后在加班申请人和申请时间上自动添加当前登录用户姓名和当前日期 ，同样，核准人核准信息时，自动添加核准人姓名和核准时间
5. 填写加班信息完成后，先保存,状态会自动存为草稿，只有通过提交按钮，状态才会变为已提交，草稿状态的记录可以进行编辑以及删除，而已提交和已核准状态的记录只能进行查看

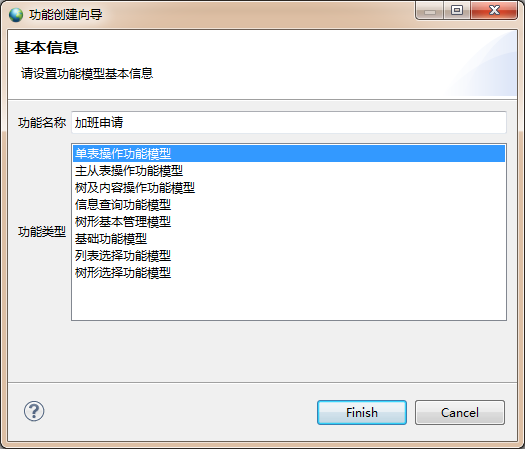
### 开发思路

1. 采用单表操作模型，使用数据表为hr\_workovertime
2. 通过jsp页面来对按钮进行修改以及添加对应的js代码，jsp页面状态查询时需要设为下拉框，选人查询的地方需要添加树形选择框
3. 在editHandler中进行判断通过在页面点击按钮执行动作的operaType的不同进行jsp页面的控制
4. 在用户登录的时候就做一次权限判断，如果登录用户不为核准人角色，那么登陆后列表页面的核准按钮不显示，点击申请按钮进入编辑页面时进行一次查询，若查询结果为空，添加当前用户姓名和日期到表单中，核准人登录时需要选择一条记录进行核准，核准完成后信息只能查看不能修改和删除
5. 申请进入编辑页面，填完表单信息后，点击保存按钮，会存入数据表中，通过在hr\_work\_overtime.xml中的sql语句进行修改，直接将状态存为草稿，只有用户点击了提交按钮后，状态才会通过editHandler中的pageAction将已提交状态存入状态的字段中

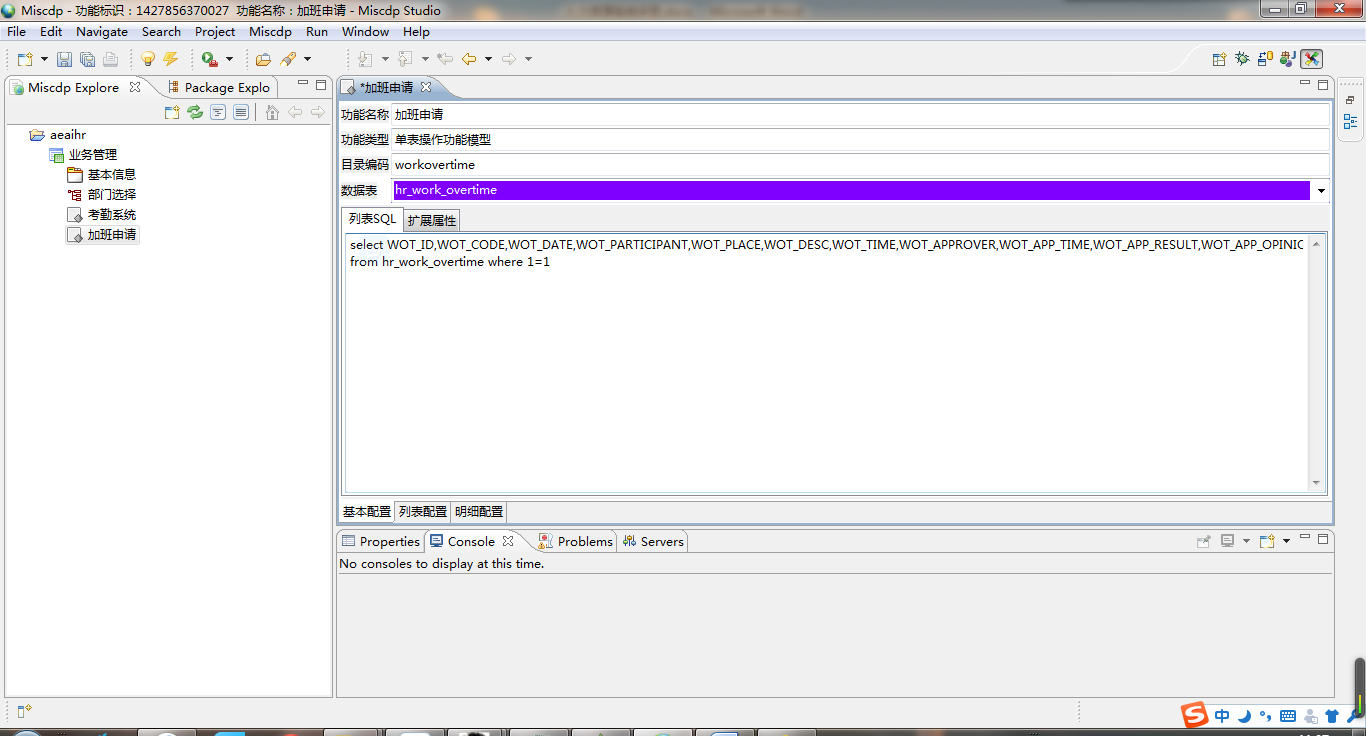
### 扩展设计

1. 创建单表操作功能模型

在业务管理功能上右键选择创建功能🡪创建加班申请模块，功能类型为单表操作管理功能：



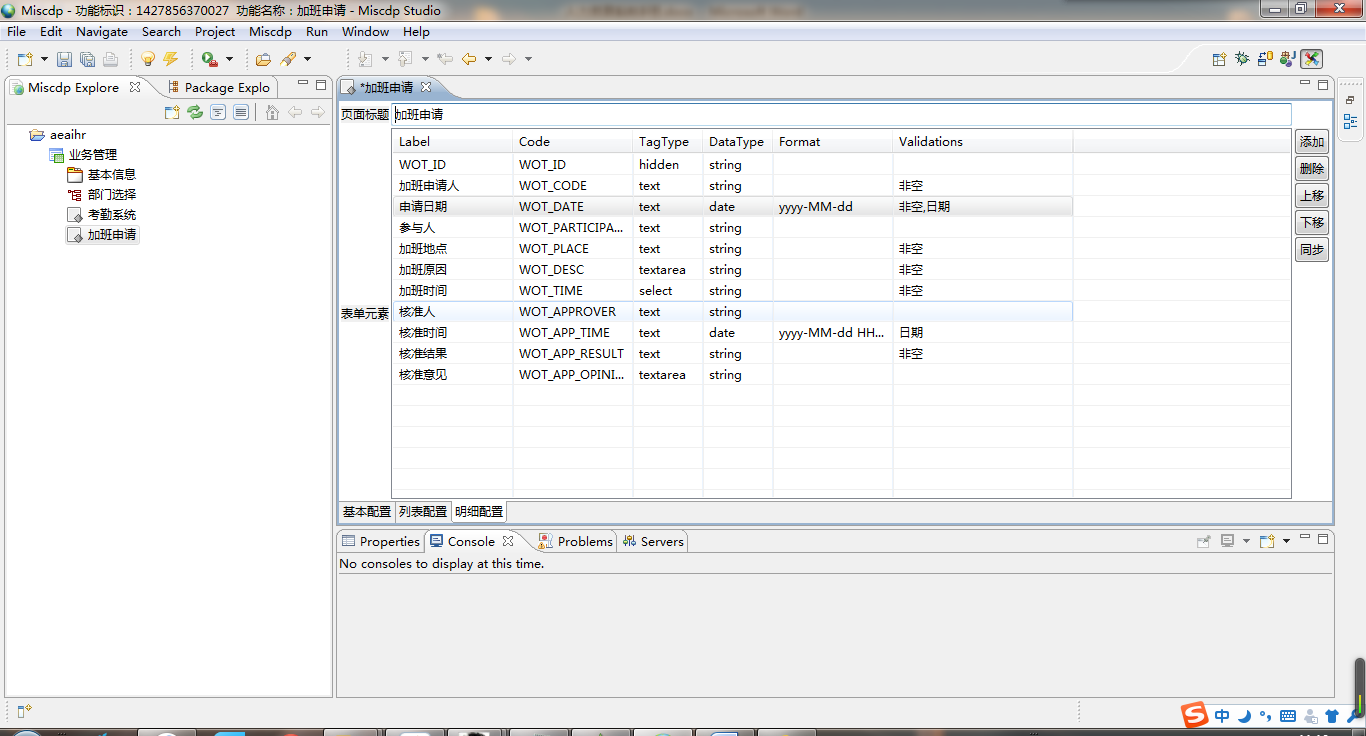
创建功能模型，目录编码为workovertime，数据表采用刚刚创建的hr\_work\_overtime表，点击初始化按钮自动生成列表sql：



对列表配置中进行修改，修改结果如下图：



明细配置修改如下图：



修改完成后，点击生成代码

1. 扩展开发
2. 在list的查询条件中增加状态筛选条件

在list的jsp页面增加下拉框选择：

&nbsp;状态<select id=*"STATE"* label=*"状态"* name=*"STATE"* class=*"select"* onchange=*"doQuery()"*><%=pageBean.selectValue("STATE")%></select>

在listHandler中的processPageAttribute方法中添加：

**protected** **void** processPageAttributes(DataParam param) {

initMappingItem("WOT\_TIME", FormSelectFactory.*create*("WOT\_TIME")

.getContent());

initMappingItem("APP\_RESULT",

FormSelectFactory.*create*("APP\_RESULT").getContent());

initMappingItem("STATE",

FormSelectFactory.*create*("STATE").getContent());

setAttribute("STATE", FormSelectFactory.*create*("STATE")

.addSelectedValue(param.get("STATE")));

}

在listHandler中的initParameters方法中添加语句：

**protected** **void** initParameters(DataParam param) {

initParamItem(param, "STATE", "");

initParamItem(param, "user\_Name", "");

}

对应的sqlmap中的sql语句需要有查询条件：

<isNotEmpty prepend=*" "* property=*"STATE"*>

and STATE = #STATE#

</isNotEmpty>

1. 在Edit中增加提交、反提交、核准按钮，对状态进行改变

首先在edit的jsp页面增加按钮：

<td align=*"center"*class=*"bartdx"*onclick="stateSubmit();" onmouseover="onMover(this);" onmouseout="onMout(this);"><input value=*"&nbsp;"*type=*"button"* class=*"submitImgBtn"* id=*"submitImgBtn"* title=*"提交"* />提交</td>

<td align=*"center"*class=*"bartdx"*onclick="stateDrafe();" onmouseover="onMover(this);" onmouseout="onMout(this);"><input value=*"&nbsp;"*type=*"button"* class=*"reSubmittedImgBtn"* id=*"drafeImgBtn"* title=*"反提交"* />反提交</td>

<td align=*"center"*class=*"bartdx"*onclick="stateApprove();" onmouseover="onMover(this);" onmouseout="onMout(this);"><input value=*"&nbsp;"*type=*"button"* class=*"approveImgBtn"* id=*"approveImgBtn"* title=*"核准"* />核准</td>

在上面的js中增加语句：

**function** stateSubmit(){

doSubmit({actionType:'submit'});

}

**function** stateApprove(){

doSubmit({actionType:'approve'});

}

**function** stateDrafe(){

doSubmit({actionType:'drafe'});

}

在editHandler中增加pageAction：

@PageAction

**public** ViewRenderer submit(DataParam param) {

param.put("STATE","submitted");

getService().updateRecord(param);

**return** **new** RedirectRenderer(getHandlerURL(listHandlerClass));

}

@PageAction

**public** ViewRenderer approve(DataParam param) {

param.put("STATE","approved");

getService().approveRecord(param);

**return** **new** RedirectRenderer(getHandlerURL(listHandlerClass));

}

@PageAction

**public** ViewRenderer drafe(DataParam param) {

param.put("STATE","drafe");

getService().updateRecord(param);

**return** **new** RedirectRenderer(getHandlerURL(listHandlerClass));

}

在接口类HrWorkOvertimeManage中增加方法：

DataRow getNowRecord(DataParam param);

**void** approveRecord(DataParam param);

同时，在接口实现类中增加方法：

**public** **void** updateRecord(DataParam param) {

String statementId = sqlNameSpace + "." + "updateRecord";

processDataType(param, tableName);

**this**.daoHelper.updateRecord(statementId, param);

}

**public** **void** approveRecord(DataParam param) {

String statementId = sqlNameSpace + "." + "approveRecord";

processDataType(param, tableName);

**this**.daoHelper.updateRecord(statementId, param);

}

在sql中增加语句：

<update id="updateRecord" parameterClass="com.agileai.domain.DataParam">

update hr\_work\_overtime

set USER\_ID=#USER\_ID:CHAR#,WOT\_DATE=#WOT\_DATE:VARCHAR#,WOT\_PARTICIPANT=#WOT\_PARTICIPANT:VARCHAR#,WOT\_PLACE=#WOT\_PLACE:VARCHAR#,WOT\_DESC=#WOT\_DESC:VARCHAR#,WOT\_OVERTIME\_DATE=#WOT\_OVERTIME\_DATE:DATE#,WOT\_TIME=#WOT\_TIME:VARCHAR#,STATE=#STATE:VARCHAR#

where WOT\_ID=#WOT\_ID#

</update>

<update id="approveRecord" parameterClass="com.agileai.domain.DataParam">

update hr\_work\_overtime

set WOT\_ID=#WOT\_ID#,WOT\_APPROVER=#WOT\_APPROVER:VARCHAR#,WOT\_APP\_TIME=#WOT\_APP\_TIME:TIMESTAMP#,APP\_RESULT=#APP\_RESULT:VARCHAR#,WOT\_APP\_OPINION=#WOT\_APP\_OPINION:VARCHAR#,STATE=#STATE:VARCHAR#

where WOT\_ID=#WOT\_ID#

</update>

在新增的时候，需要在页面直接显示当前状态：

Edit的jsp页面对下拉框状态进行修改：

<th width=*"100"* nowrap>状态</th>

<td>

<input id=*"STATE\_TEXT"* label=*"状态"* name=*"STATE\_TEXT"* type=*"text"* value=*"*<%=pageBean.selectedText("STATE")%>*"* size=*"24"* class=*"text"* readonly=*"readonly"*/>

<input id=*"STATE"* label=*"状态"* name=*"STATE"* type=*"hidden"* value=*"*<%=pageBean.selectedValue("STATE")%>*"* />

</td>

显示一个只读的文本框，后面有一个隐藏域用来存状态

1. List通过时间段进行查询

在list的jsp页面增加语句：

&nbsp;起止日期<input id=*"sdate"* label=*"开始时间"* name=*"sdate"* type=*"text"* value=*"*<%=pageBean.inputDate("sdate")%>*"* size=*"10"* class=*"text"* readonly=*"readonly"*/><img id=*"sdatePicker"* src=*"images/calendar.gif"* width=*"16"* height=*"16"* alt=*"日期/时间选择框"* />

-<input id=*"edate"* label=*"截止时间"* name=*"edate"* type=*"text"* value=*"*<%=pageBean.inputDate("edate")%>*"* size=*"10"* class=*"text"* readonly=*"readonly"*/><img id=*"edatePicker"* src=*"images/calendar.gif"* width=*"16"* height=*"16"* alt=*"日期/时间选择框"* />

在listHandler中的initParameters方法增加语句

initParamItem(

param,

"sdate",

DateUtil.getDateByType(DateUtil.YYMMDD\_HORIZONTAL,

DateUtil.*getBeginOfMonth*(**new** Date())));

initParamItem(

param,

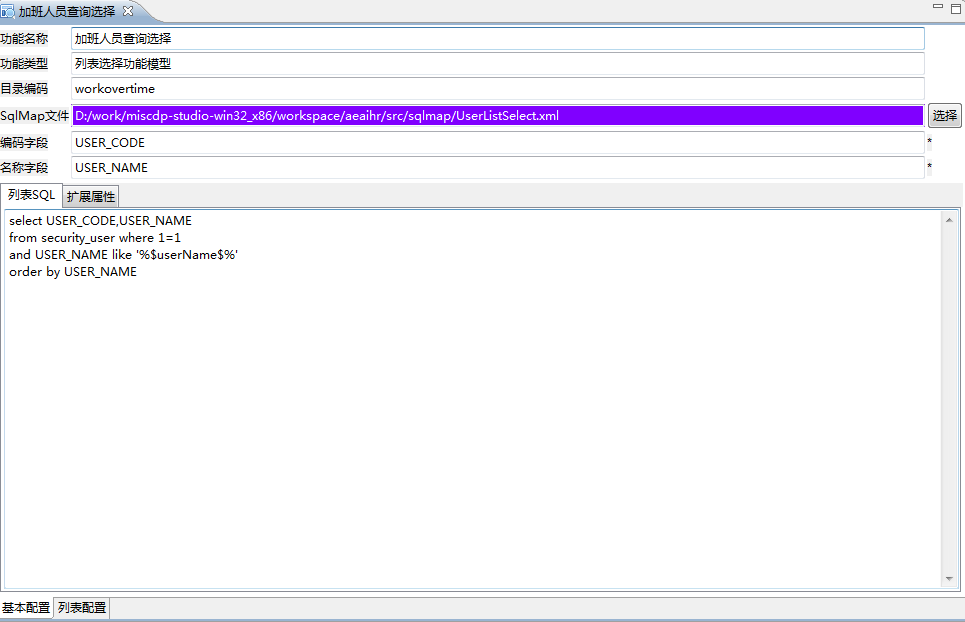
"edate",

DateUtil.getDateByType(DateUtil.YYMMDD\_HORIZONTAL,

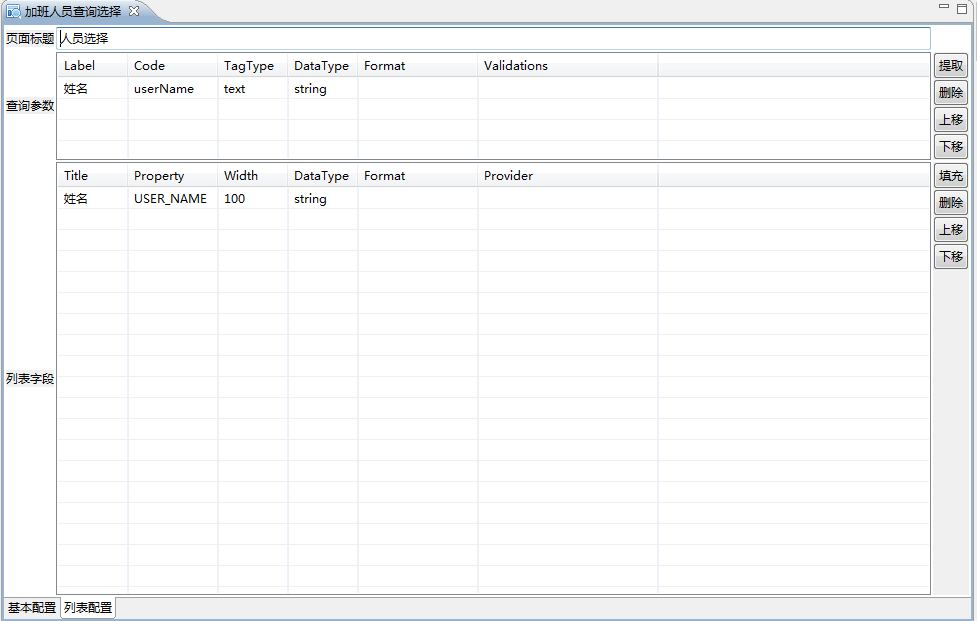
DateUtil.*getDateAdd*(**new** Date(), DateUtil.*DAY*, 1)));

1. 姓名查询通过树形选择框来选择

首先创建列表选择功能模型，如图：



列表配置如下图:



在list的jsp中增加语句：

&nbsp;姓名 <input id=*"user\_Name"* label=*"姓名 "* name=*"user\_Name"* type=*"text"* value=*"*<%=pageBean.inputDate("user\_Name")%>*"* size=*"24"* class=*"text"* /><input type=*"hidden"* label=*"人员选择"* id=*"USER\_CODE"* name=*"USER\_CODE"* value=*"*<%=pageBean.inputValue("USER\_CODE")%>*"* /><img id=*"userIdSelectImage"* src=*"images/sta.gif"* width=*"16"* height=*"16"* onclick="openUserIdBox()" />

在sql中的find语句需要添加筛选条件：

<isNotEmpty prepend=*" "* property=*"user\_Name"*>

and b.USER\_NAME = #user\_Name#

</isNotEmpty>

将UserListSelectListHandler中的initParameters方法修改：

**protected** **void** initParameters(DataParam param) {

initParamItem(param, "userName", "");

}

1. 登录时对权限进行判断，是否为业务核准人，list的显示不同

首先在listHandler中的preparedisplay方法中添加判断：

**public** ViewRenderer prepareDisplay(DataParam param) {

User user = (User) **this**.getUser();

PrivilegeHelper privilegeHelper = **new** PrivilegeHelper(user);

**if** (privilegeHelper.isApprove()) {

param.put("currentUserId","");

setAttribute("hasRight", **true**);

setAttribute("userId", user.getUserId());

} **else** {

param.put("currentUserId", user.getUserId());

setAttribute("hasRight", **false**);

}

mergeParam(param);

initParameters(param);

setAttribute("canSignIn", **true**);

**this**.setAttributes(param);

List<DataRow> rsList = getService().findRecords(param);

**this**.setRsList(rsList);

processPageAttributes(param);

**return** **new** LocalRenderer(getPage());

}

对于list的jsp页面的改动：

<tr>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"A"* align=*"center"* onclick="doRequest('insertRequest')"><input id=*"createImgBtn"* value=*"&nbsp;"* title=*"申请"* type=*"button"* class=*"createImgBtn"* />申请</td>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"E"* align=*"center"* onclick="doRequest('updateRequest')"><input id=*"editImgBtn"* value=*"&nbsp;"* title=*"编辑"* type=*"button"* class=*"editImgBtn"* />编辑</td>

<%**if**(pageBean.getBoolValue("hasRight")){ %>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"R"* align=*"center"* onclick="doRequest('approveRequest')"><input id=*"approveImgBtn"* value=*"&nbsp;"* title=*"核准"* type=*"button"* class=*"approveImgBtn"* />核准</td>

<%}%>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"V"* align=*"center"* onclick="doRequest('viewDetail')"><input id=*"detailImgBtn"* value=*"&nbsp;"* title=*"查看"* type=*"button"* class=*"detailImgBtn"* />查看</td>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"D"* align=*"center"* onclick="doDelete($('#'+rsIdTagId).val());"><input id=*"deleteImgBtn"* value=*"&nbsp;"* title=*"删除"* type=*"button"* class=*"delImgBtn"* />删除</td>

</tr>

1. 对于记录的不同状态，对list页面的按钮进行控制

在list的jsp页面上的js代码中进行判断：

**function** controlUpdateBtn(stateResult){

**if**(stateResult =='drafe'){

enableButton("editImgBtn");

<%**if**(pageBean.getBoolValue("hasRight")){ %>

disableButton("approveImgBtn");

<%}%>

enableButton("detailImgBtn");

enableButton("deleteImgBtn");

}else

**if**(stateResult =='submitted'){

disableButton("editImgBtn");

<%**if**(pageBean.getBoolValue("hasRight")){ %>

enableButton("approveImgBtn");

<%}%>

enableButton("detailImgBtn");

disableButton("deleteImgBtn");

}else

**if**(stateResult =='approved'){

disableButton("editImgBtn");

<%**if**(pageBean.getBoolValue("hasRight")){ %>

disableButton("approveImgBtn");

<%}%>

enableButton("detailImgBtn");

disableButton("deleteImgBtn");

}

}

**function** controlCanEdit(userId,state){

**if** (state == "drafe" && userId != "<%=pageBean.getStringValue("userId")%>" ){

disableButton("editImgBtn");

}

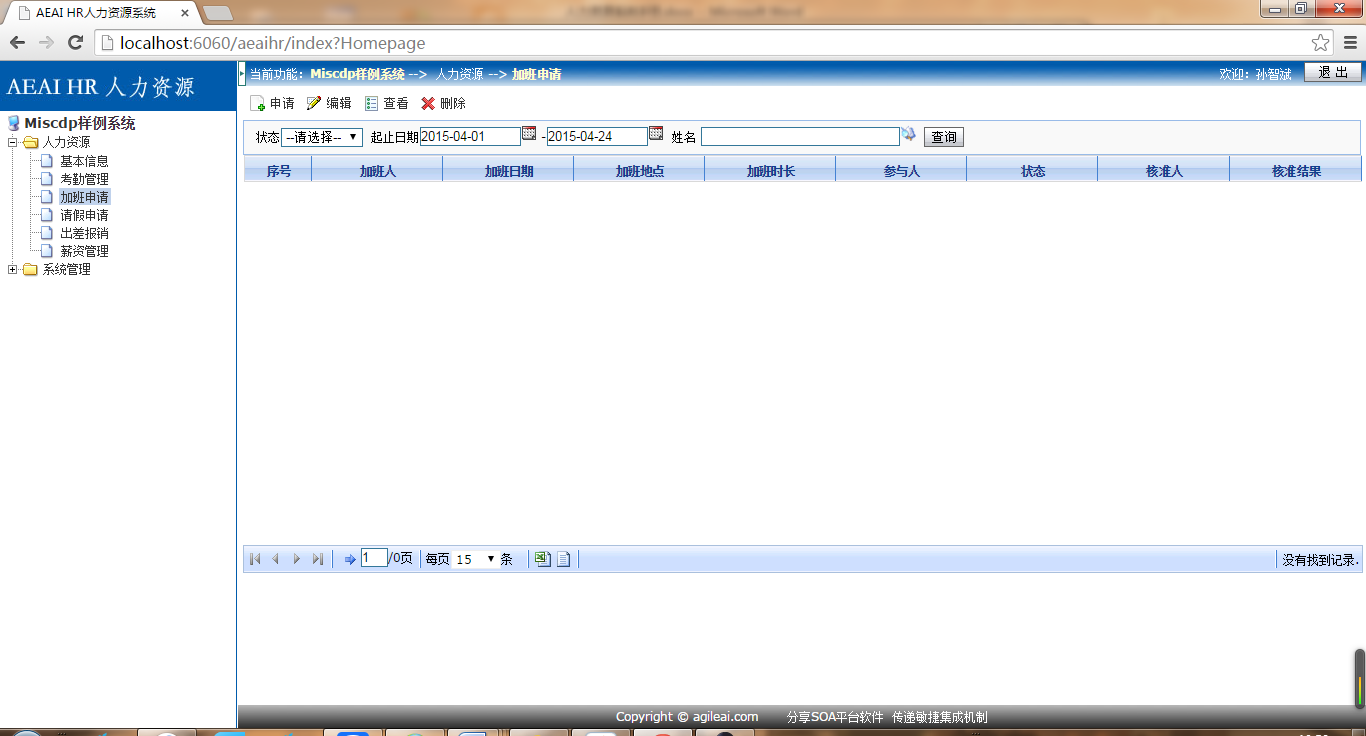
}

在下面的列表显示行上添加语句：

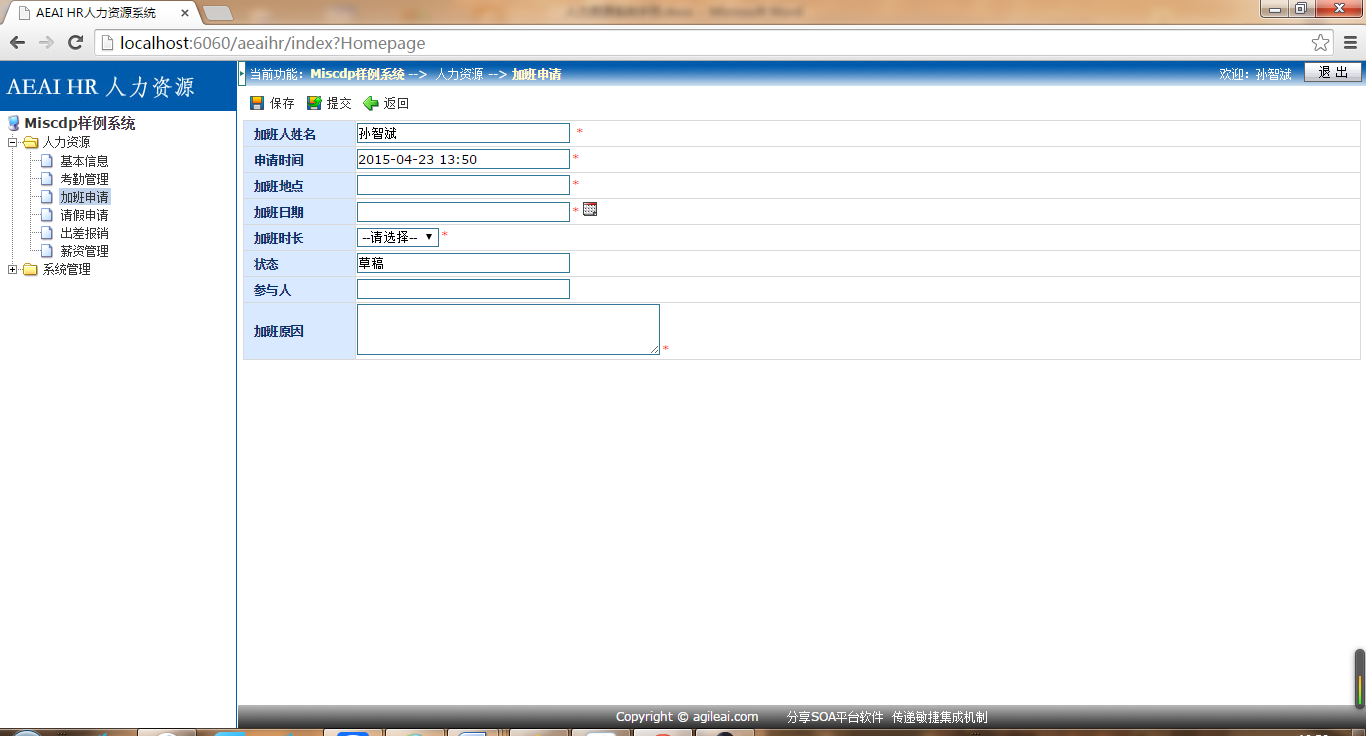
<ec:row styleClass="odd" ondblclick="clearSelection();doRequest('viewDetail')" oncontextmenu="selectRow(this,{WOT\_ID:'${row.WOT\_ID}'});controlUpdateBtn('${row.STATE}');controlCanEdit('${row.USER\_ID}','${row.STATE}');refreshConextmenu()" onclick="selectRow(this,{WOT\_ID:'${row.WOT\_ID}'});controlUpdateBtn('${row.STATE}');controlCanEdit('${row.USER\_ID}','${row.STATE}');">

1. 测试运行

测试运行，效果如下图:



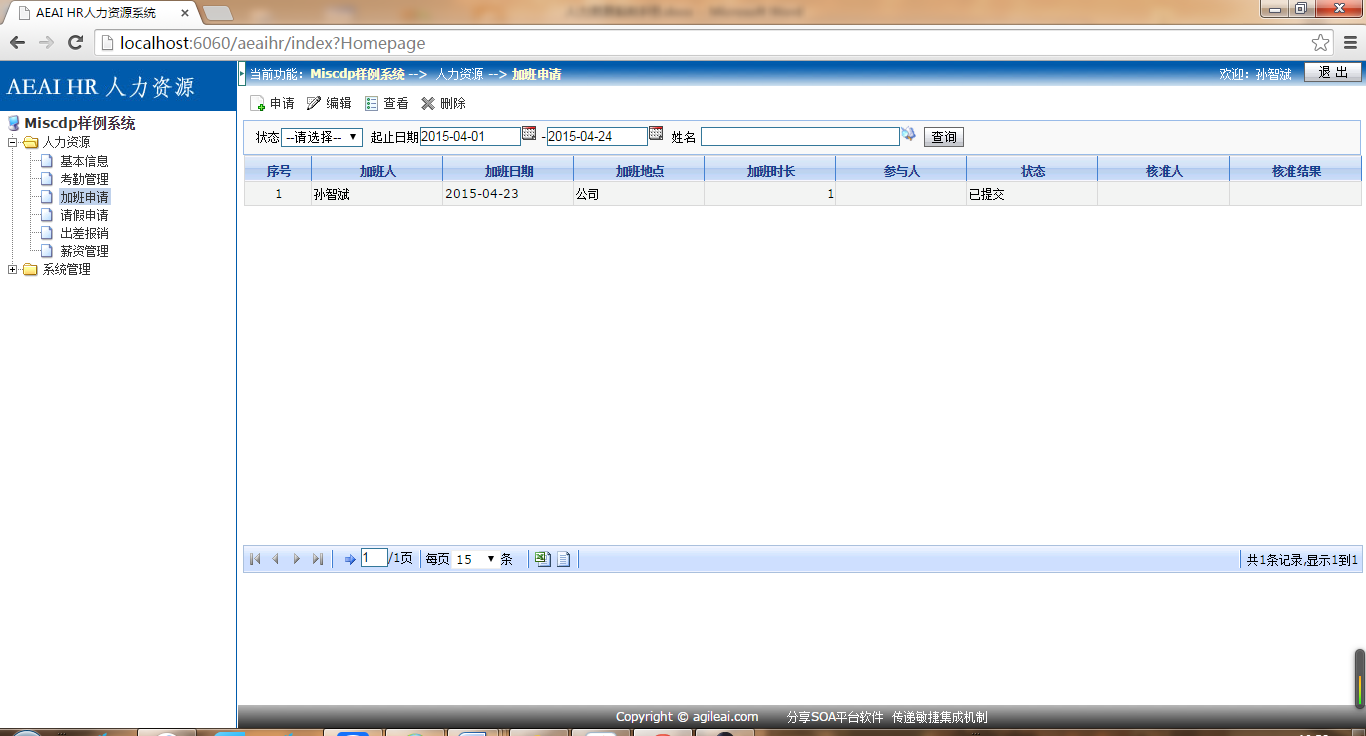
申请后进入编辑页面：



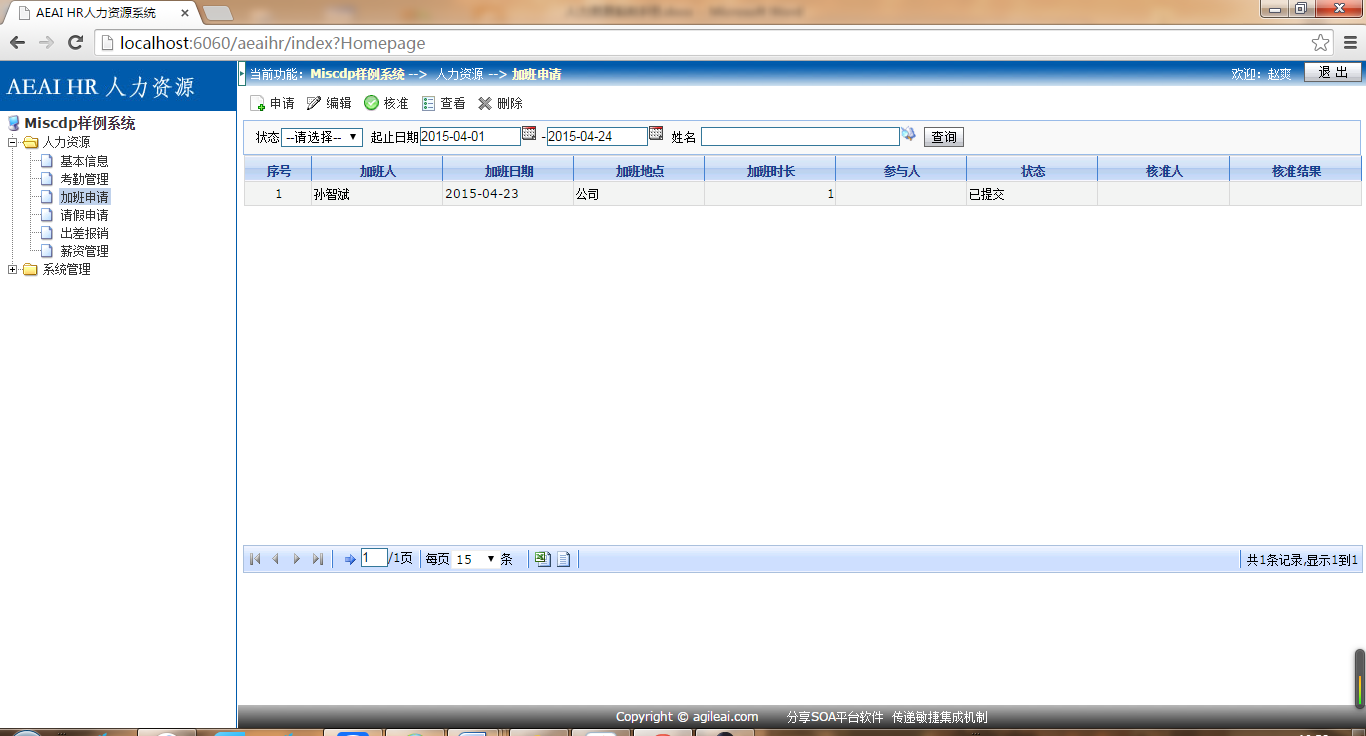
保存后：



提交后：

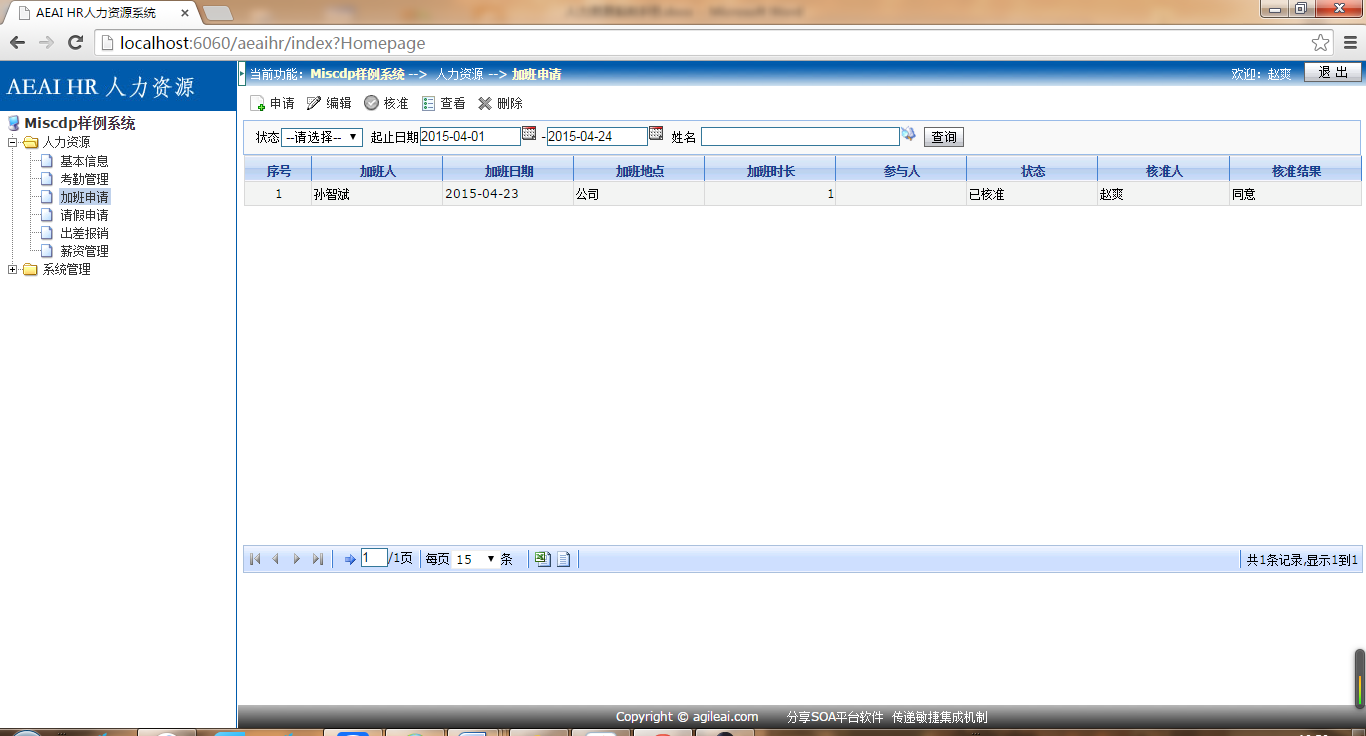


核准人进行核准：





核准以后：



### 注意事项

1. 在双击记录时，采用的方法还是查看，需要在editHandler中添加对应的判断，通过对登录人权限和状态的综合判断来对按钮进行控制。
2. 若记录状态为草稿时，还需要将记录申请人的ID和登录用户的ID进行一次比较，来控制编辑页面的按钮。

## 请假申请

### 功能需求

1. 请假申请模块作为员工请假时提交申请的一个模块
2. 员工选择请假申请按钮时，请假人为当前的用户，请假日期为当前日期
3. 当员工登录时进行一次判断查询员工是否为核准角色，若不是，请假核准按钮失效
4. 核准人选择请假核准按钮时，请假人和请假时间不变，核准人为当前用户，核准时间为当前时间，核准一旦生效，信息只能查看不能修改和删除
5. 列表页面有按时间段查询和通过树形选择框来选择员工查询两种方式

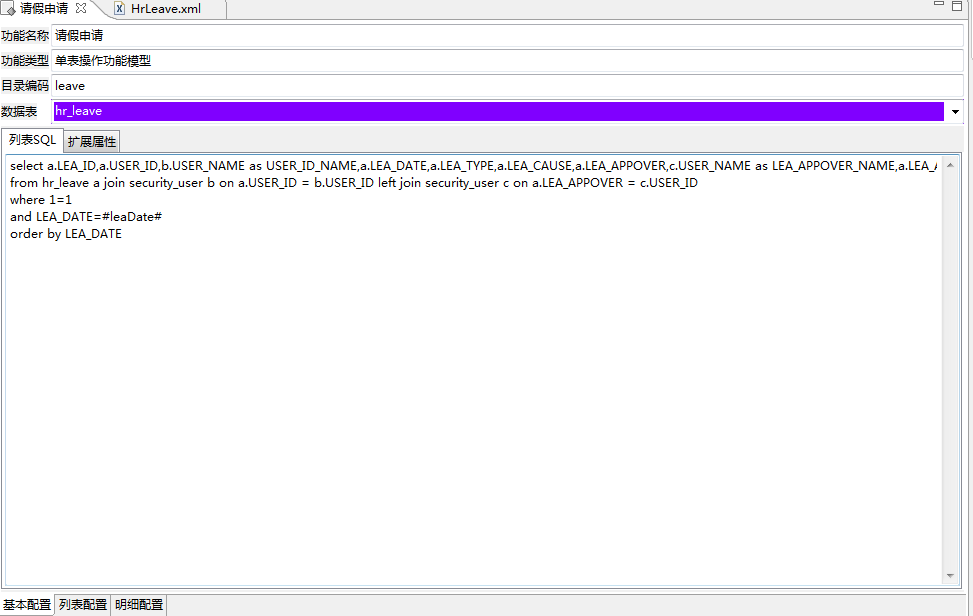
### 开发思路

1. 采用单表操作模型，使用数据库中的hr\_leave表
2. 登录人选择请假申请按钮时查询数据库关联查询找到登录用户的姓名和当前时间添加到表单中
3. 需要进行权限判断，判断是否为核准人（App），若不是请假核准按钮失效
4. 类似于考勤管理系统，请假按钮只能保存一次，保存后请假按钮失效，使用核准按钮时请假人和请假日期、原因、类型都为只读
5. 需要在listjsp页面上添加对应的按钮，还有员工树形选择框功能扩展，类似于部门选择

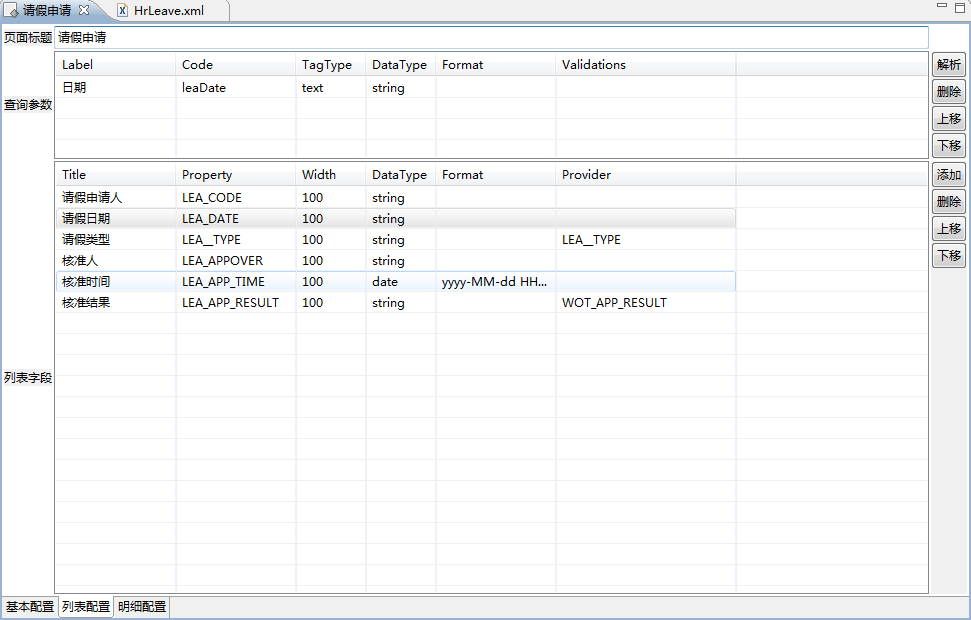
### 扩展设计

1. 创建单表操作模型

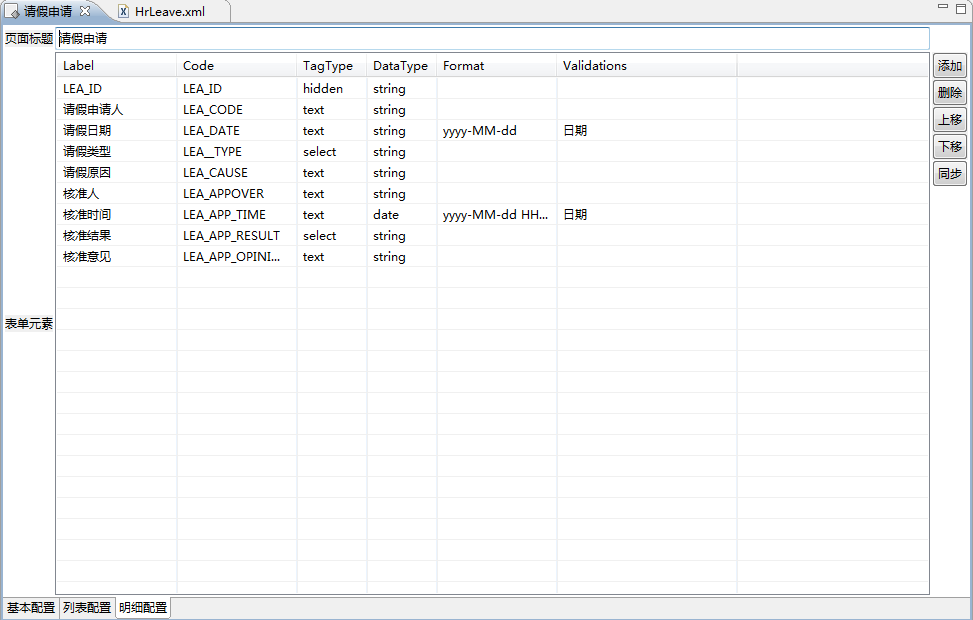
右键‘业务管理’🡪‘创建功能’🡪创建单表操作功能模型，名称为‘请假申请’🡪数据表采用hr\_leave的（截图为最初模型，一切以sql为准）：



切换到列表配置中修改为中文：



在明细配置中修改为中文名：



修改完成后，生成代码。

1. 扩展开发
2. 在list的查询条件中增加状态筛选条件

与加班申请中的一样, 在list的jsp页面增加下拉框选择：

&nbsp;状态<select id=*"STATE"* label=*"状态"* name=*"STATE"* class=*"select"* onchange=*"doQuery()"*><%=pageBean.selectValue("STATE")%></select>

在listHandler中的processPageAttribute方法中添加：

**protected** **void** processPageAttributes(DataParam param) {

initMappingItem("LEA\_TYPE",

FormSelectFactory.*create*("LEA\_TYPE").getContent());

initMappingItem("APP\_RESULT",

FormSelectFactory.*create*("APP\_RESULT").getContent());

initMappingItem("STATE",

FormSelectFactory.*create*("STATE").getContent());

setAttribute("STATE", FormSelectFactory.*create*("STATE")

.addSelectedValue(param.get("STATE")));

}

在listHandler中的initParameters方法中添加语句：

**protected** **void** initParameters(DataParam param) {

initParamItem(param, "STATE", "");

initParamItem(param, "user\_Name", "");

}

对应的sqlmap中的sql语句需要有查询条件：

<isNotEmpty prepend=*" "* property=*"STATE"*>

and STATE = #STATE#

</isNotEmpty>

1. 在Edit中增加提交按钮和核准按钮，同时对状态进行改变

首先在edit的jsp页面增加按钮：

<td align=*"center"*class=*"bartdx"*onclick="stateSubmit();" onmouseover="onMover(this);" onmouseout="onMout(this);"><input value=*"&nbsp;"*type=*"button"* class=*"submitImgBtn"* id=*"submitImgBtn"* title=*"提交"* />提交</td>

<td align=*"center"*class=*"bartdx"*onclick="stateDrafe();" onmouseover="onMover(this);" onmouseout="onMout(this);"><input value=*"&nbsp;"*type=*"button"* class=*"reSubmittedImgBtn"* id=*"drafeImgBtn"* title=*"反提交"* />反提交</td>

<td align=*"center"*class=*"bartdx"*onclick="stateApprove();" onmouseover="onMover(this);" onmouseout="onMout(this);"><input value=*"&nbsp;"*type=*"button"* class=*"approveImgBtn"* id=*"approveImgBtn"* title=*"核准"* />核准</td>

在上面的js中增加语句：

**function** stateSubmit(){

doSubmit({actionType:'submit'});

}

**function** stateApprove(){

doSubmit({actionType:'approve'});

}

**function** stateDrafe(){

doSubmit({actionType:'drafe'});

}

在editHandler中增加pageAction：

@PageAction

**public** ViewRenderer submit(DataParam param) {

param.put("STATE","submitted");

getService().updateRecord(param);

**return** **new** RedirectRenderer(getHandlerURL(listHandlerClass));

}

@PageAction

**public** ViewRenderer approve(DataParam param) {

param.put("STATE","approved");

getService().approveRecord(param);

**return** **new** RedirectRenderer(getHandlerURL(listHandlerClass));

}

@PageAction

**public** ViewRenderer drafe(DataParam param) {

param.put("STATE","drafe");

getService().updateRecord(param);

**return** **new** RedirectRenderer(getHandlerURL(listHandlerClass));

}

在接口类HrLeaveManage中增加方法：

DataRow getNowRecord(DataParam param);

**void** approveRecord(DataParam param);

同时，在接口实现类中增加方法：

**public** **void** updateRecord(DataParam param) {

String statementId = sqlNameSpace + "." + "updateRecord";

processDataType(param, tableName);

**this**.daoHelper.updateRecord(statementId, param);

}

**public** **void** approveRecord(DataParam param) {

String statementId = sqlNameSpace + "." + "approveRecord";

processDataType(param, tableName);

**this**.daoHelper.updateRecord(statementId, param);

}

在sql中修改增加语句：

<update id="updateRecord" parameterClass="com.agileai.domain.DataParam">

update hr\_leave

set USER\_ID=#USER\_ID:CHAR#,LEA\_DATE=#LEA\_DATE:VARCHAR#,LEA\_TYPE=#LEA\_TYPE:VARCHAR#,LEA\_SDATE=#LEA\_SDATE:TIMESTAMP#,LEA\_EDATE=#LEA\_EDATE:TIMESTAMP#,LEA\_DAYS=#LEA\_DAYS:VARCHAR#,LEA\_CAUSE=#LEA\_CAUSE:VARCHAR#,STATE=#STATE:VARCHAR#

where LEA\_ID=#LEA\_ID#

</update>

<update id="approveRecord" parameterClass="com.agileai.domain.DataParam">

update hr\_leave

set LEA\_APPOVER=#LEA\_APPOVER:VARCHAR#,LEA\_APP\_TIME=#LEA\_APP\_TIME:TIMESTAMP#,APP\_RESULT=#APP\_RESULT:VARCHAR#,LEA\_APP\_OPINION=#LEA\_APP\_OPINION:VARCHAR#,STATE=#STATE:VARCHAR#

where LEA\_ID=#LEA\_ID#

</update>

在新增的时候，需要在页面直接显示当前状态：

Edit的jsp页面对下拉框状态进行修改：

<th width=*"100"* nowrap>状态</th>

<td>

<input id=*"STATE\_TEXT"* label=*"状态"* name=*"STATE\_TEXT"* type=*"text"* value=*"*<%=pageBean.selectedText("STATE")%>*"* size=*"24"* class=*"text"* readonly=*"readonly"*/>

<input id=*"STATE"* label=*"状态"* name=*"STATE"* type=*"hidden"* value=*"*<%=pageBean.selectedValue("STATE")%>*"* />

</td>

显示一个只读的文本框，后面有一个隐藏域用来存状态

1. List通过时间段进行查询

在list的jsp页面增加语句：

&nbsp;起止日期<input id=*"sdate"* label=*"开始时间"* name=*"sdate"* type=*"text"* value=*"*<%=pageBean.inputDate("sdate")%>*"* size=*"10"* class=*"text"* readonly=*"readonly"*/><img id=*"sdatePicker"* src=*"images/calendar.gif"* width=*"16"* height=*"16"* alt=*"日期/时间选择框"* />

-<input id=*"edate"* label=*"截止时间"* name=*"edate"* type=*"text"* value=*"*<%=pageBean.inputDate("edate")%>*"* size=*"10"* class=*"text"* readonly=*"readonly"*/><img id=*"edatePicker"* src=*"images/calendar.gif"* width=*"16"* height=*"16"* alt=*"日期/时间选择框"* />

在listHandler中的initParameters方法增加语句

initParamItem(

param,

"sdate",

DateUtil.getDateByType(DateUtil.YYMMDD\_HORIZONTAL,

DateUtil.*getBeginOfMonth*(**new** Date())));

initParamItem(

param,

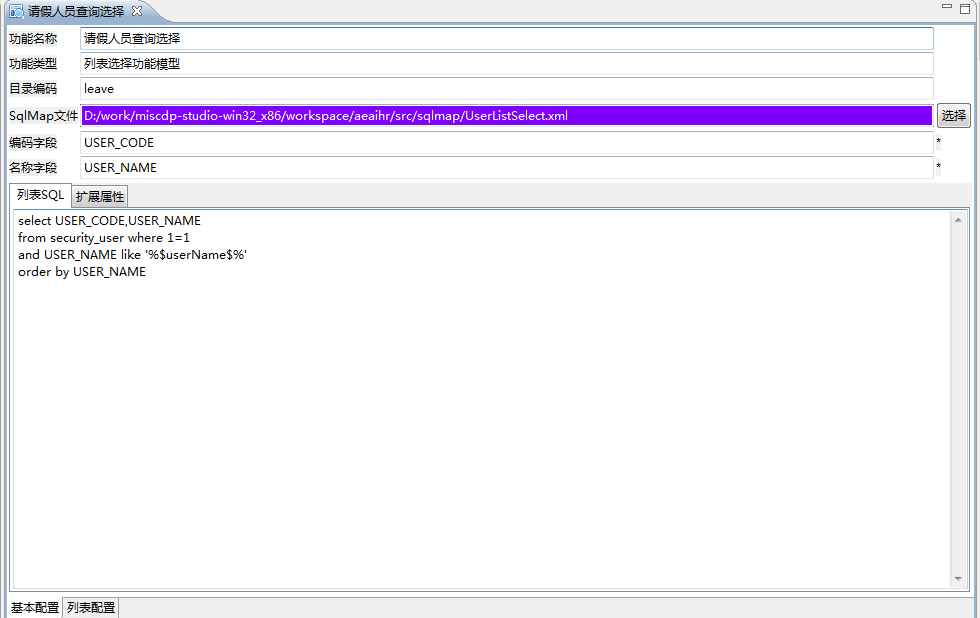
"edate",

DateUtil.getDateByType(DateUtil.YYMMDD\_HORIZONTAL,

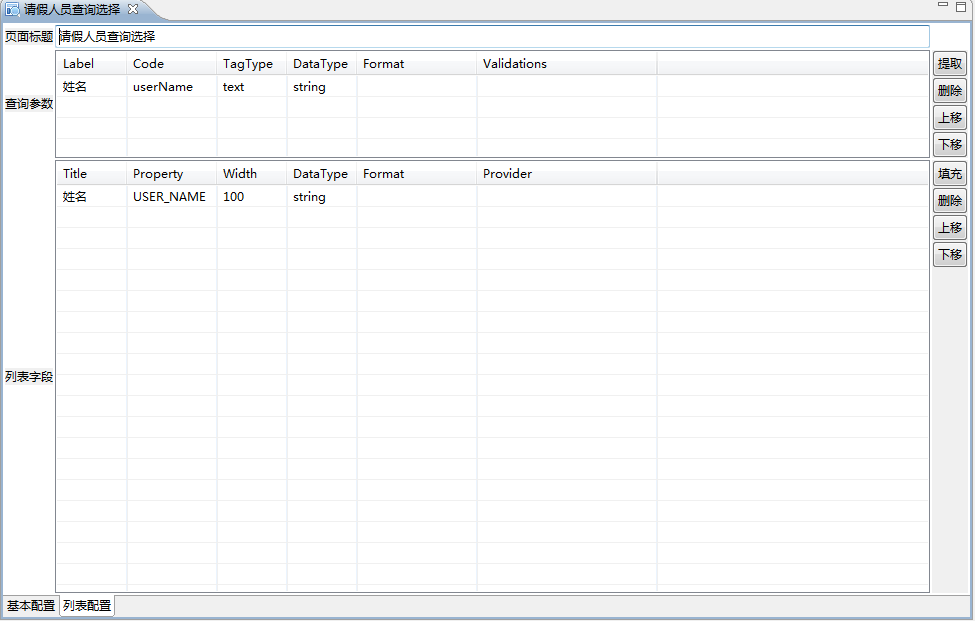
DateUtil.*getDateAdd*(**new** Date(), DateUtil.*DAY*, 1)));

1. 姓名查询通过树形选择框来选择

首先创建列表选择功能模型，如图：



列表配置如下图:



在list的jsp中增加语句：

&nbsp;姓名 <input id=*"user\_Name"* label=*"姓名 "* name=*"user\_Name"* type=*"text"* value=*"*<%=pageBean.inputDate("user\_Name")%>*"* size=*"24"* class=*"text"* /><input type=*"hidden"* label=*"人员选择"* id=*"USER\_CODE"* name=*"USER\_CODE"* value=*"*<%=pageBean.inputValue("USER\_CODE")%>*"* /><img id=*"userIdSelectImage"* src=*"images/sta.gif"* width=*"16"* height=*"16"* onclick="openUserIdBox()" />

在sql中的find语句需要添加筛选条件：

<isNotEmpty prepend=*" "* property=*"user\_Name"*>

and b.USER\_NAME = #user\_Name#

</isNotEmpty>

将UserListSelectListHandler中的initParameters方法修改：

**protected** **void** initParameters(DataParam param) {

initParamItem(param, "userName", "");

}

1. 登录时对权限进行判断，是否为业务核准人，list的显示不同

首先在listHandler中的preparedisplay方法中添加判断：

**public** ViewRenderer prepareDisplay(DataParam param){

User user = (User) **this**.getUser();

PrivilegeHelper privilegeHelper = **new** PrivilegeHelper(user);

**if** (privilegeHelper.isApprove()) {

param.put("currentUserId", "");

setAttribute("hasRight", **true**);

setAttribute("userId", user.getUserId());

} **else** {

param.put("currentUserId", user.getUserId());

setAttribute("hasRight", **false**);

}

mergeParam(param);

initParameters(param);

setAttribute("canSignIn", **true**);

DataParam queryParam = **new** DataParam();

queryParam.put("leaDate",

DateUtil.*getDateByType*(DateUtil.*YYMMDD\_HORIZONTAL*, **new** Date()));

queryParam.put("currentUser", user.getUserId());

**this**.setAttributes(param);

List<DataRow> rsList = getService().findRecords(param);

**this**.setRsList(rsList);

processPageAttributes(param);

**return** **new** LocalRenderer(getPage());

}

对list的jsp页面进行调整：

<tr>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"A"* align=*"center"* onclick="doRequest('insertRequest')"><input id=*"createImgBtn"* value=*"&nbsp;"* title=*"申请"* type=*"button"* class=*"createImgBtn"* />申请</td>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"E"* align=*"center"* onclick="doRequest('updateRequest')"><input id=*"editImgBtn"*value=*"&nbsp;"* title=*"编辑"* type=*"button"* class=*"editImgBtn"* />编辑</td>

<%**if**(pageBean.getBoolValue("hasRight")){ %>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"R"* align=*"center"* onclick="doRequest('approveRequest')"><input id=*"approveImgBtn"* value=*"&nbsp;"* title=*"核准"* type=*"button"* class=*"approveImgBtn"* />核准</td>

<%}%>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"V"* align=*"center"* onclick="doRequest('viewDetail')"><input id=*"detailImgBtn"*value=*"&nbsp;"* title=*"查看"* type=*"button"* class=*"detailImgBtn"* />查看</td>

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"D"* align=*"center"* onclick="doDelete($('#'+rsIdTagId).val());"><input id=*"deleteImgBtn"* value=*"&nbsp;"* title=*"删除"* type=*"button"* class=*"delImgBtn"* />删除</td>

</tr>

对list页面按钮的控制:

在list的jsp页面上的js代码中进行判断：

**function** controlUpdateBtn(stateResult){

**if**(stateResult =='drafe'){

enableButton("editImgBtn");

<%**if**(pageBean.getBoolValue("hasRight")){ %>

disableButton("approveImgBtn");

<%}%>

enableButton("detailImgBtn");

enableButton("deleteImgBtn");

}else

**if**(stateResult =='submitted'){

disableButton("editImgBtn");

<%**if**(pageBean.getBoolValue("hasRight")){ %>

enableButton("approveImgBtn");

<%}%>

enableButton("detailImgBtn");

disableButton("deleteImgBtn");

}else

**if**(stateResult =='approved'){

disableButton("editImgBtn");

<%**if**(pageBean.getBoolValue("hasRight")){ %>

disableButton("approveImgBtn");

<%}%>

enableButton("detailImgBtn");

disableButton("deleteImgBtn");

}

}

**function** controlCanEdit(userId,state){

**if** (state == "drafe" && userId != "<%=pageBean.getStringValue("userId")%>" ){

disableButton("editImgBtn");

}

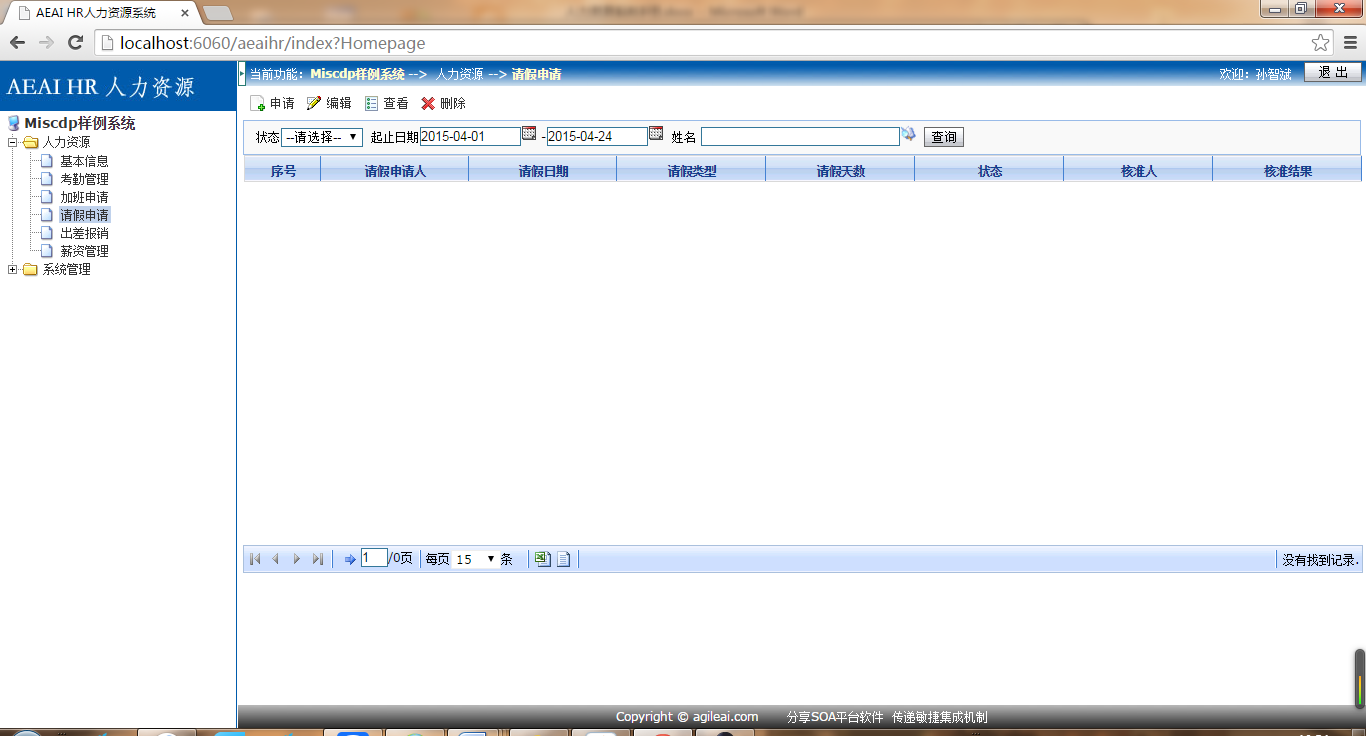
}

在下面的列表显示行上添加语句：

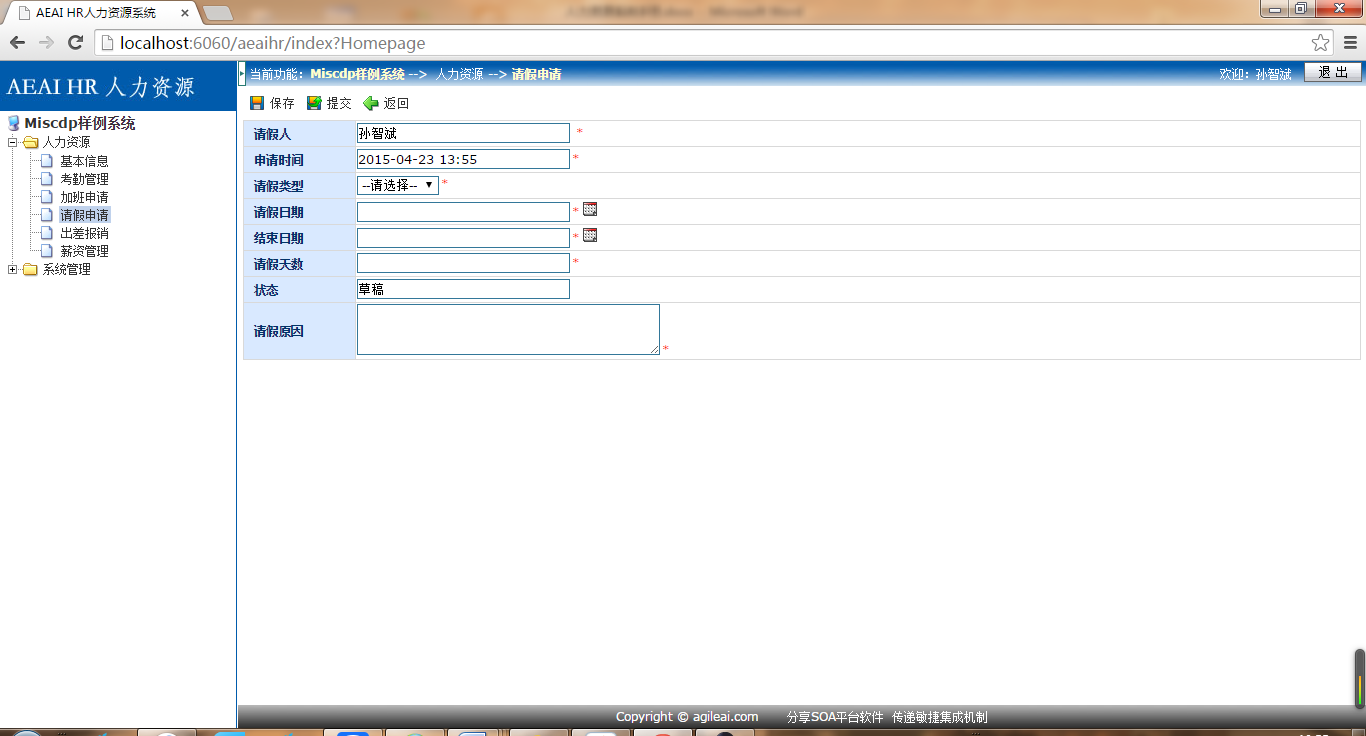
<ec:row styleClass="odd" ondblclick="clearSelection();doRequest('viewDetail')" oncontextmenu="selectRow(this,{LEA\_ID:'${row.LEA\_ID}'});controlUpdateBtn('${row.STATE}');controlCanEdit('${row.USER\_ID}','${row.STATE}');refreshConextmenu()" onclick="selectRow(this,{LEA\_ID:'${row.LEA\_ID}'});controlUpdateBtn('${row.STATE}');controlCanEdit('${row.USER\_ID}','${row.STATE}');">

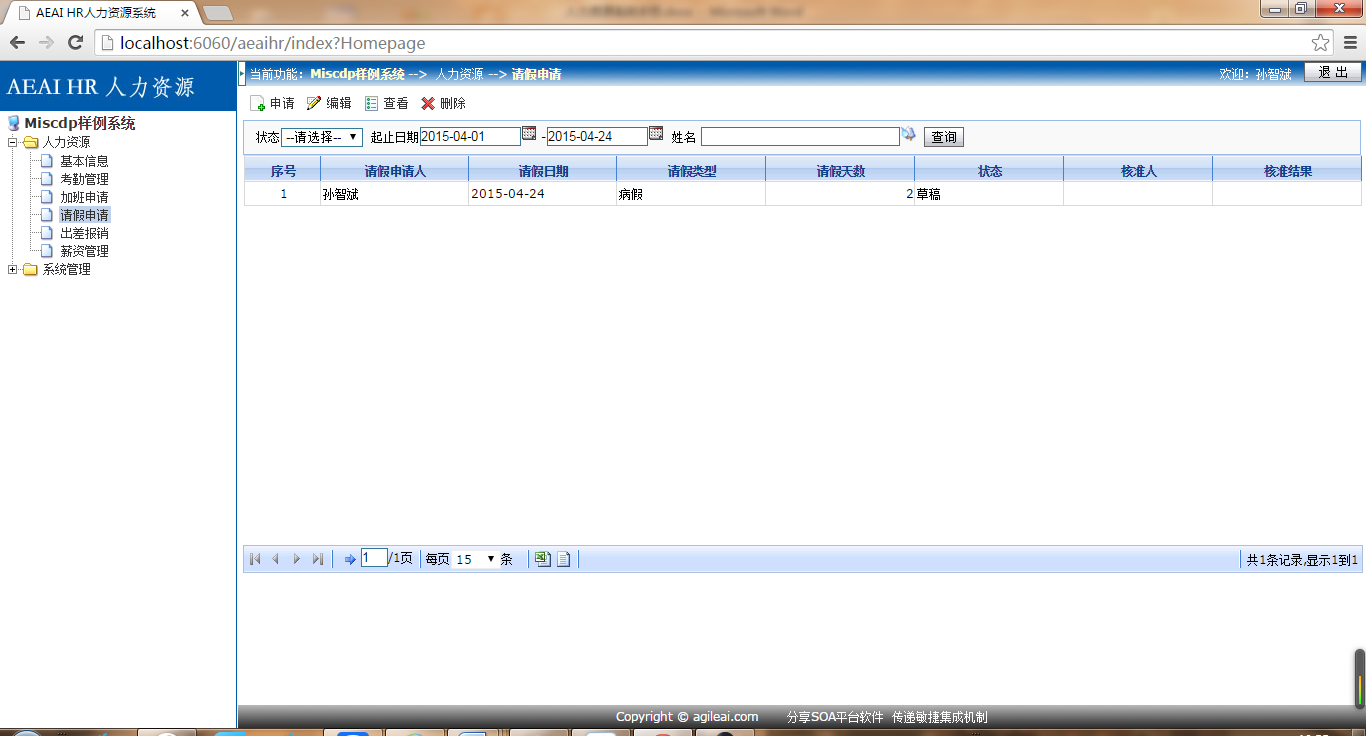
1. 测试运行

测试运行，效果如下图：

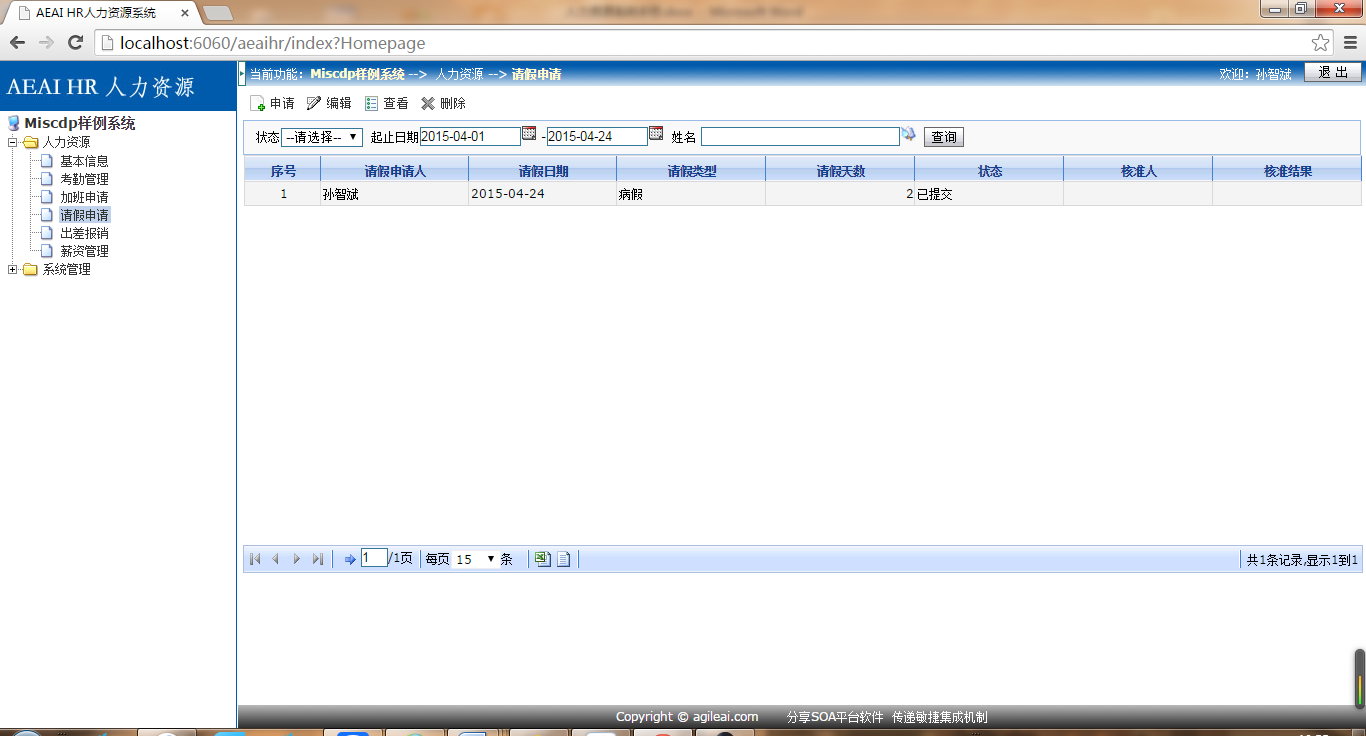


申请记录时：

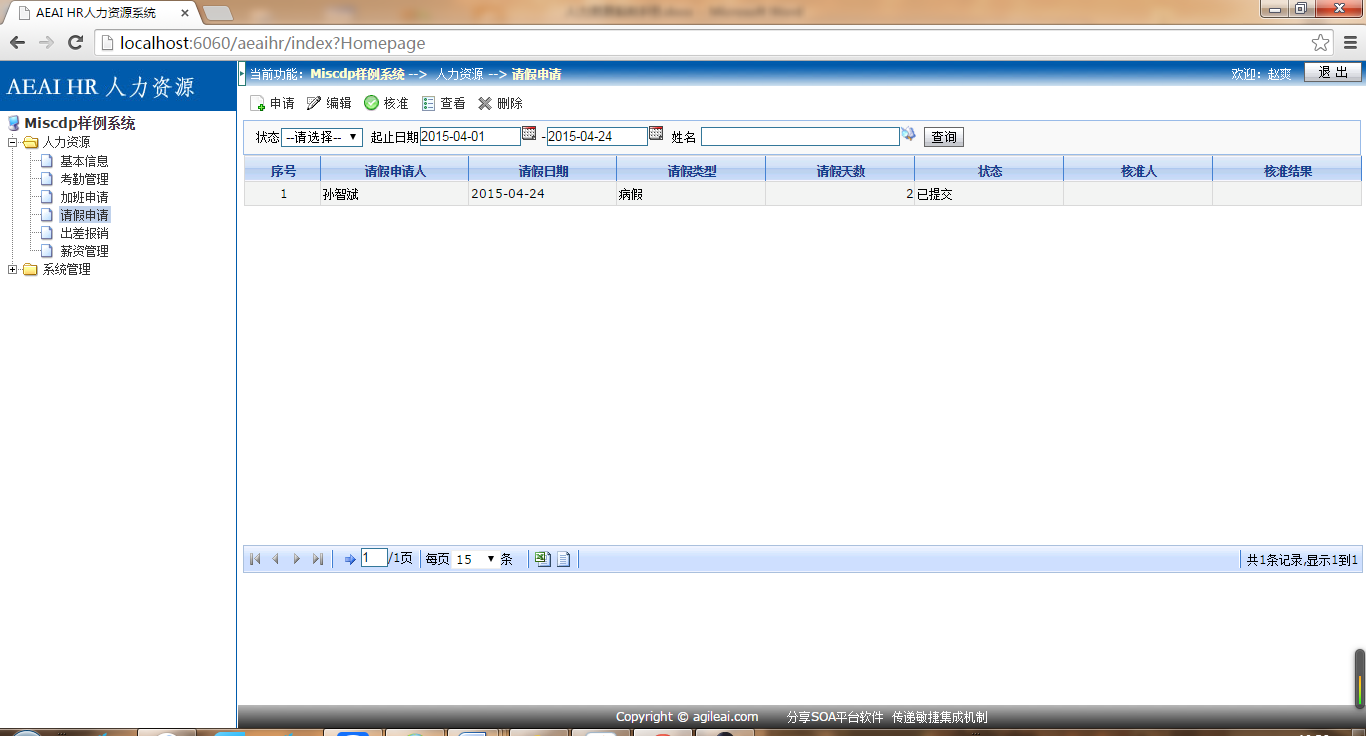


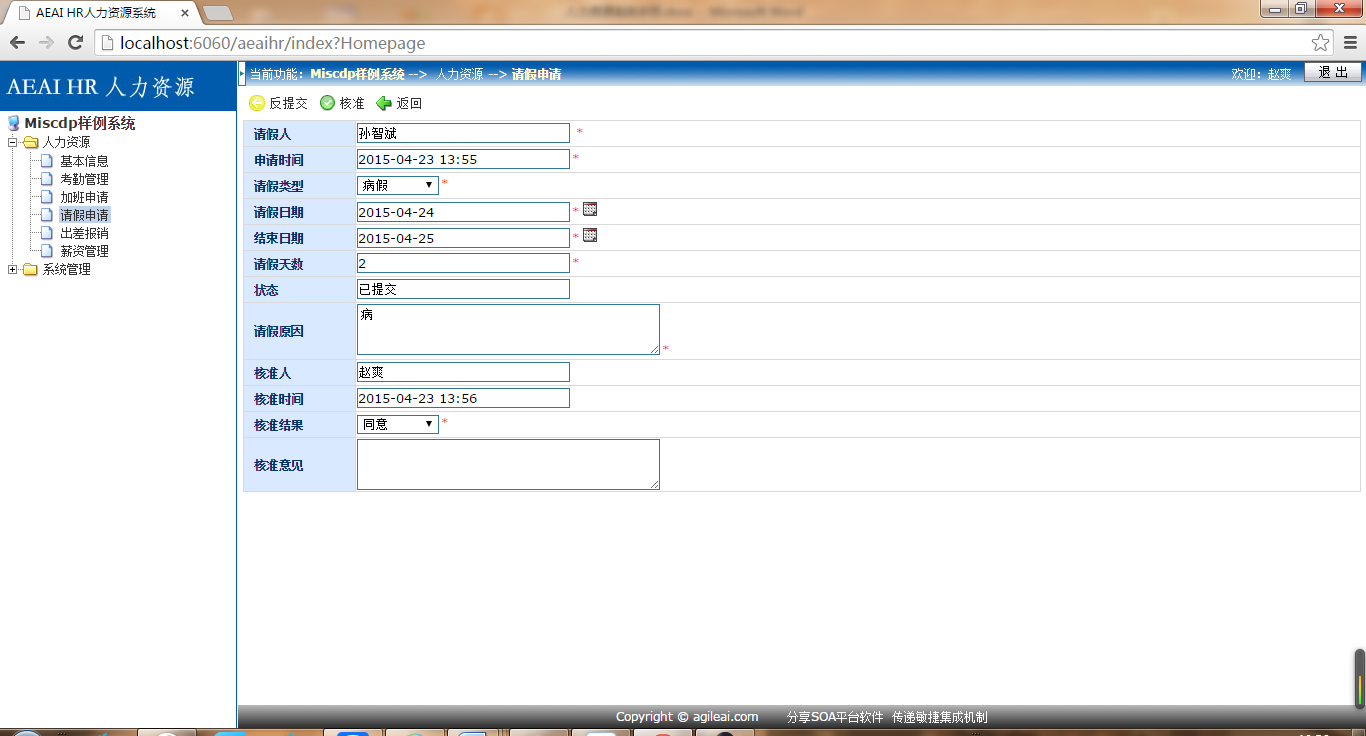


提交记录后：

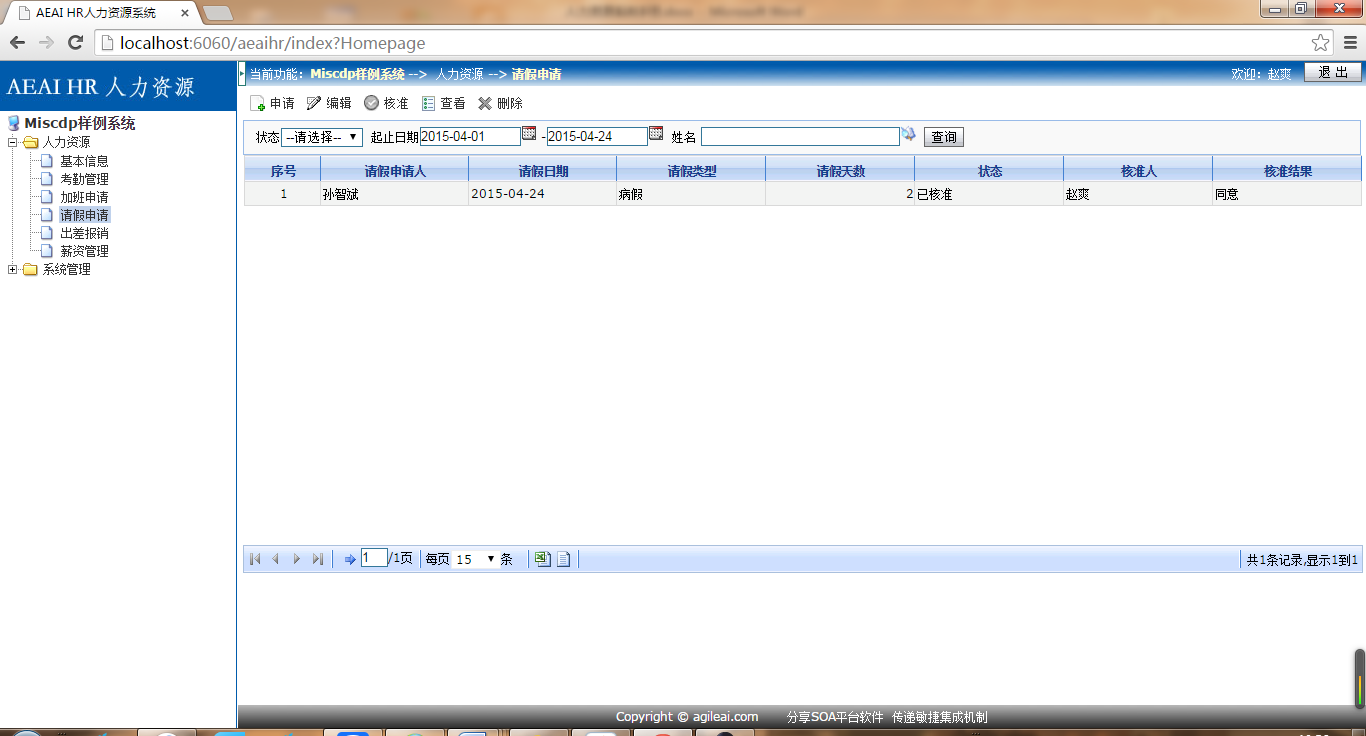


核准人登录进行审核：





核准以后：



### 注意事项

1. 在双击记录时，采用的方法还是查看，需要在editHandler中添加对应的判断，通过对登录人权限和状态的综合判断来对按钮进行控制。
2. 若记录状态为草稿时，还需要将记录申请人的ID和登录用户的ID进行一次比较，来控制编辑页面的按钮。

## 出差报销

### 功能需求

1. 员工出差报销模块，由出差员工和业务核准人共同使用。
2. 业务核准人在列表页面上可以看到所有员工的出差信息同时可以使用状态、时间段、姓名来进行筛选。
3. 业务核准人能对所有员工的信息状态为已提交的记录，进行核准操作及反提交操作，当其他员工状态为草稿时业务核准人只能查看。
4. 除业务核准人以外的其他员工在登陆后只能看到自己的信息，看不到其他人的信息，当信息状态为草稿时可以进行编辑修改，一旦提交后不能再对信息进行操作，只有核准人对信息反提交之后才能进行修改。
5. 在新增一条记录时，总费用不需要个人去填写，是自动计算出来的，保存后对费用清单进行填写，填写完的费用清单会自动计算费用加到总费用中。
6. 当信息被提交后，状态修改为已提交状态，这是只能由业务核准人进行核准或反提交操作，一旦信息被核准，核准后的信息不可以在进行操作，只能进行查看。

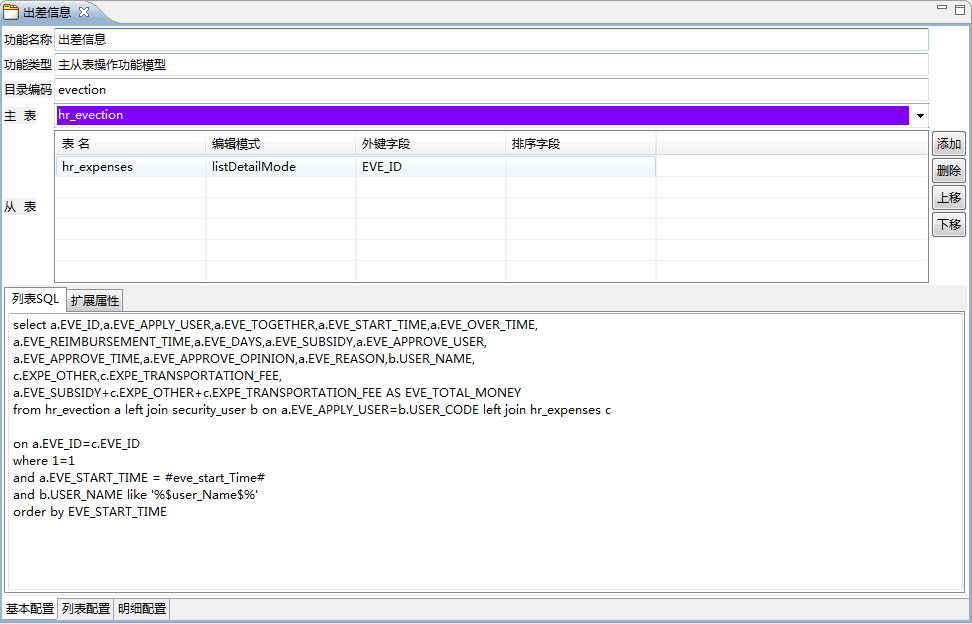
### 开发思路

1. 登录时就需要区分出来是那种角色使用，在listHandler中做判断。
2. 通过listHandler中做出判断是否为核准人，若是sql语句中的筛选条件就为空，若不是sql中的筛选条件为当前用户的USER\_ID。
3. 需要在EditHandler中通过operatype的不同来进行按钮的控制，在list和edit 的jsp页面增加按钮控制。
4. 在登录时判断权限，在通过权限来控制在list中显示个人信息还是显示所有人的信息。
5. 通过HR\_EXPENSES费用清单信息表里的交通费用，其他费用和HR\_EVETCTION表的里的补助金额，计算出汇总费用，需要在接口实现类中添加具体的方法实现，还需要在sql中进行修改。
6. 非业务核准人看不到核准和反提交按钮，在editHandler中进行判断，edit的jsp页面增加按钮控制，在list的jsp页面也需要增加按钮控制

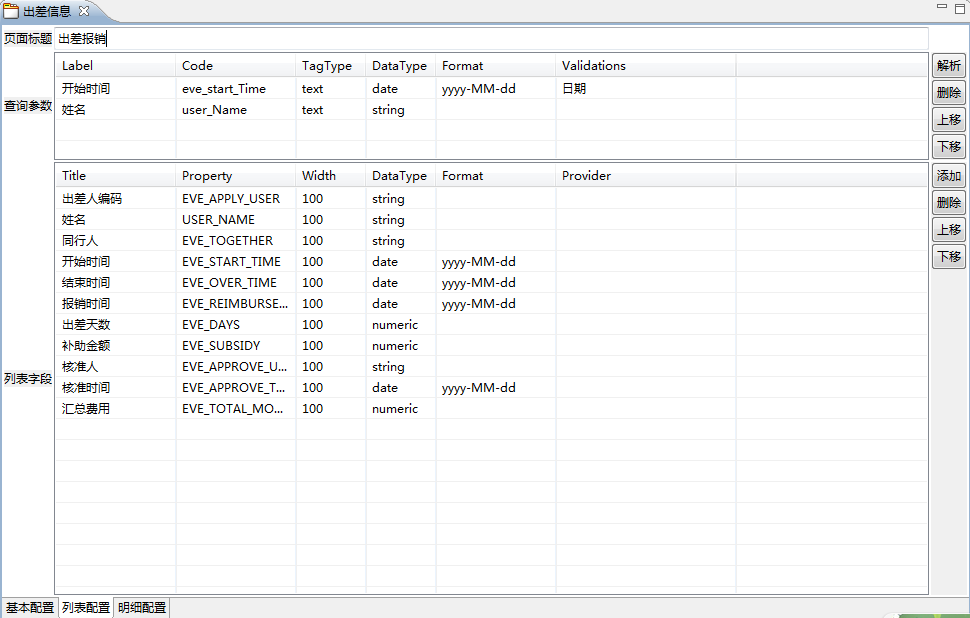
### 扩展设计

1. 创建主从表模型

右键‘业务管理’🡪‘创建功能’🡪‘主从表功能操作模型’，名称为‘出差报销’🡪目录编码evection主表hr\_evection🡪‘初始化’，其他信息如图:

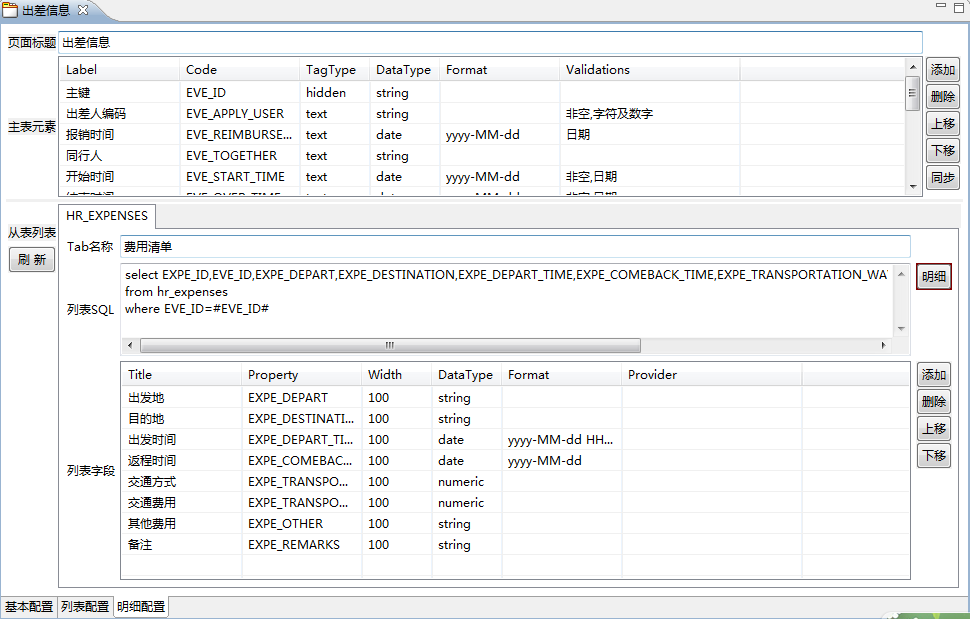


列表配置页面需要修改为中文:

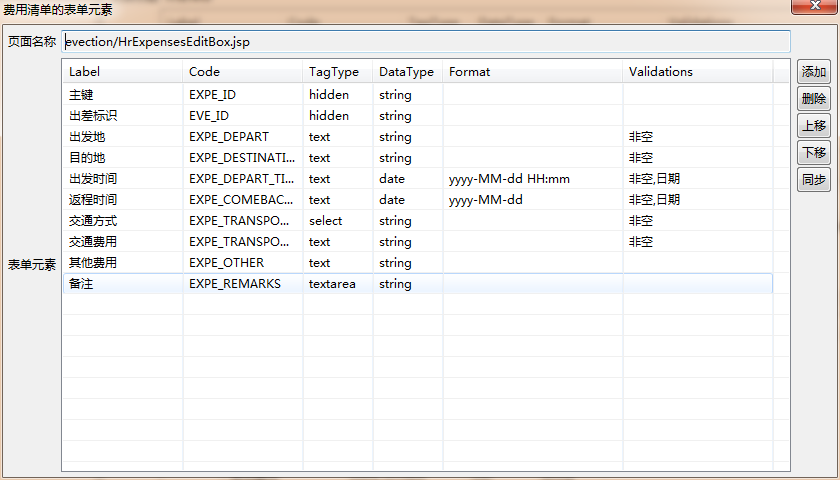


明细配置中也修改为中文：





点击‘明细’按钮，同样修改为中文，并对下拉框‘交通方式’的code的编码值和编码类型进行选择。

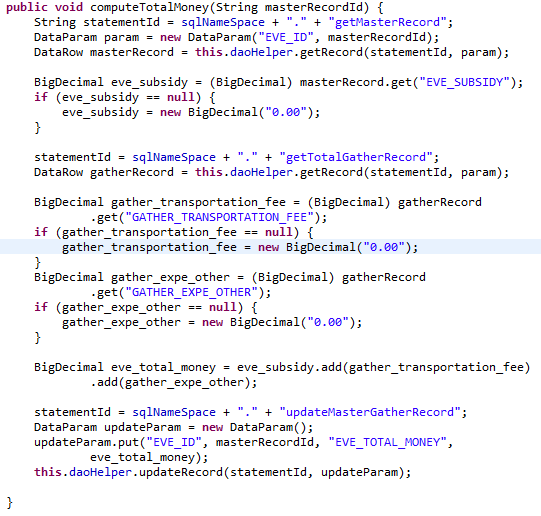


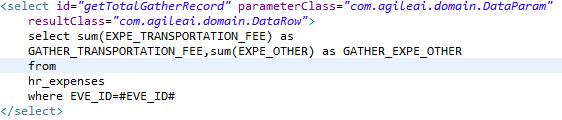
之后确认无误，点击生成代码。

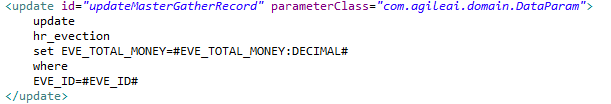
1. 扩展开发
2. 计算汇总费用

通过HR\_EXPENSES费用清单信息表里的交通费用，其他费用和HR\_EVETCTION表的里的补助金额，计算出汇总费用，实现步骤如下：

首先在HrEvectionManageImpl添加方法然后将三个字段值分别获取到，其中交通费用和其他费用的数量不止一个，所以需要定义两个新的字段，分别去接收交通费和其他费用各自的值。

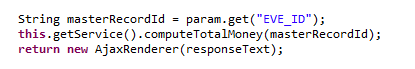






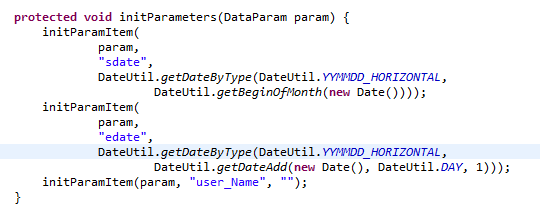
两个字段分别为gather\_transportation\_fee和gather\_expe\_other，然后需要到sqlmap里去修改SQL语句，首先我们需要创建个sql的查询语句，用sum查出相同ID值的交通费和其他费用的各自综合。然后在房里将三个值加到一起得出EVE\_TOTAL\_MONEY。我们还需要重新创建一个sql语句，然后将EVE\_TOTAL\_MONEY字段放到列表的查询sql语句中。这样总和就可以显示在列表页面上了。

还有一步，因为这个方法是保存时对数据更新，所以需要到EditHandler类里找到doSaveAction方法,如果没有进到父类里去找，然后拷贝到子类里，然后在方法里添加代码调用computeTotalMoney方法。代码如下；

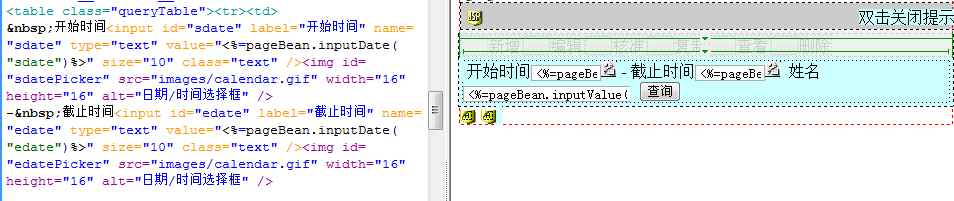


1. 设置查询条件为时间段查询

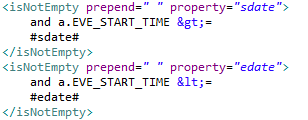
首先在列表的Handler里添加方法，sdate为开始时间的命名，edate为结束时间的命名，然后将开始时间设置为月初的第一天，结束时间设置为当天的第二天，具体代码如下:



下一步我们需要在JSP页面修改视图，将文本框添加上去id值分别为上面命名的sdate和edate。



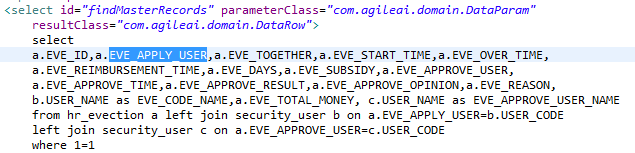
最后，修改sqlmap里的列表的查询语句，这样功能就可以实现了。将查询条件修改为：



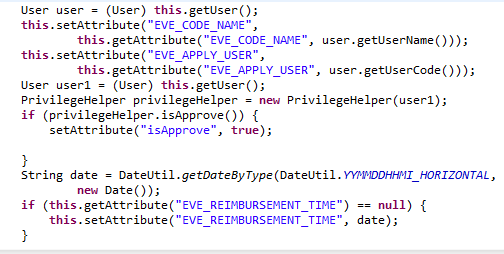
1. 默认获取当前的账户信息

在主表编辑页面里，申请时通过登录账号将申请人姓名和系统时间显示出来，核准时，将核准人的姓名和核准时的系统时间显示出来，具体步骤如下

首先将出差人编码设置为隐藏域，出差人编码设置hidden属性，新添加EVE\_CODE\_NAME字段，然后修改sql语句，和user表关联，将user表里的USER\_NAME as 到EVE\_CODE\_NAME。重点提示下：如果需要关联两个用户，就需要关联两次user表，先将b.USER\_NAME as 到EVE\_CODE\_NAME，然后再将c.USER\_NAME as 到EVE\_APPROVE\_USER\_NAME具体sql语句如下。

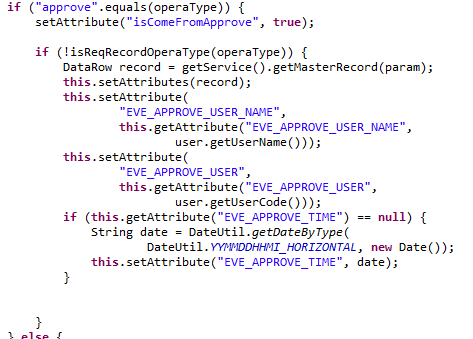


然后在编辑页面的Handler里面扩展方法，第一个为申请人的，将代码放到了processPageAttributes方法里。代码如下：



第二个为核准人的，将代码放入了prepareDisplay方法里，首先将jsp页面的核准按钮的id取出，在代码第一行进行判断，判断是否为点击核准按钮进入的编辑页面。如果是代码将继续执行下去，最后就会自动获取到当前核准人的姓名和系统时间。

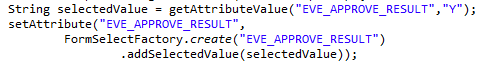




1. 手动添加下拉框列表框类型的字段

在主表里添加了核准结果字段，首先肯定是在数据库里添加EVE\_APPROVE\_RESULT核准结果字段，然后在sqlmap里找到对应的查询，插入和更新的sql语句加入EVE\_APPROVE\_RESULT字段。

然后在编辑页面的handler类里的processPageAttributes方法里添加代码，，如果需设置默认值，就将getAttributeValue里的默认值设置下



下一步我们需要将列表页面修改一下，找到processPageAttributes方法，我们需要将页面显示的结果为同意和不同意，而不是Y和N，具体代码如下：



最后我们只需要列表JSP页面，添加一列EVE\_APPROVE\_RESULT来接收和显示核准结果的，具体代码如下：

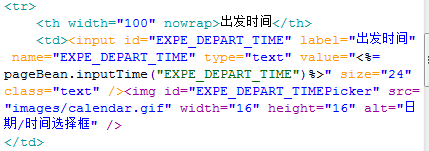


1. 手动设置时间精确到分钟数

将从表的出发时间，精确到分，首先也是将数据库对应字段的date类型更改为datatime类型。

然后需要修改jsp页面

第一步，找到从表的编辑页面，将pageBean的代码设置为inputTime。



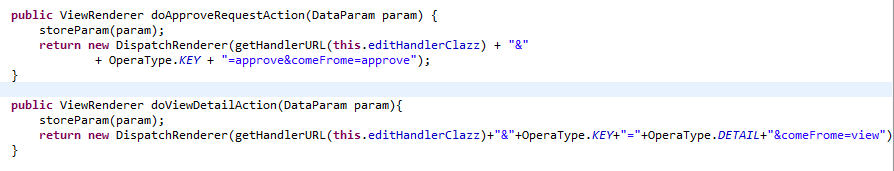
然后再将js代码更改为yyyy-MM-dd HH:mm格式，然后在页面上操作的时候就自动为系统当前的时间，也可以自行修改。



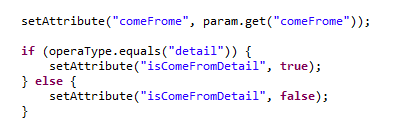
1. 核准权限和核准按钮

核准时候的OperaType是approve，一切显示正常，但是保存之后变为detail，因为需求查看的时候也需要将核准的四项字段显示出来，需要将approve和detail设置为一个值comeFrome，具体介绍如下：

首先在list集合里将两个按钮的OperaType取出放入comeFrome里，代码如下：



然后在编辑页面的handler里进行判断，判断点击按钮的OperaType的值。



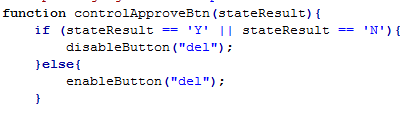
最后在jsp页面里判断将核准的四个字段，括起来，最后末端以<%}%>完了



1. List页面按钮失效

核准后的记录，只要有核准结果，一律不可以删除，编辑和再次进入核准界面，这个扩展功能，只需要在jsp页面修改即可，具体介绍如下。

首先需要在js代码里定义方法controlApproveBtn，填入按钮的id值。如下图：

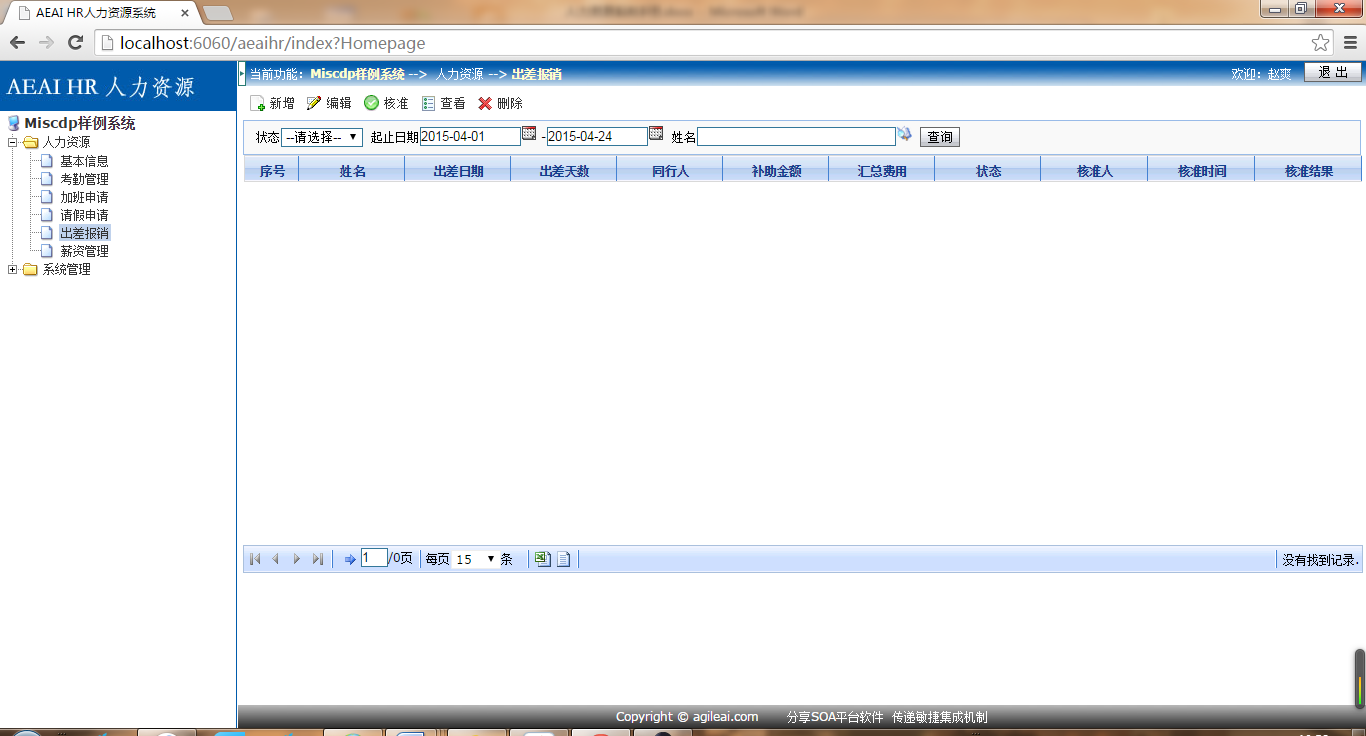


然后在单击选中记录的事件响应中添加方法，使其生效。

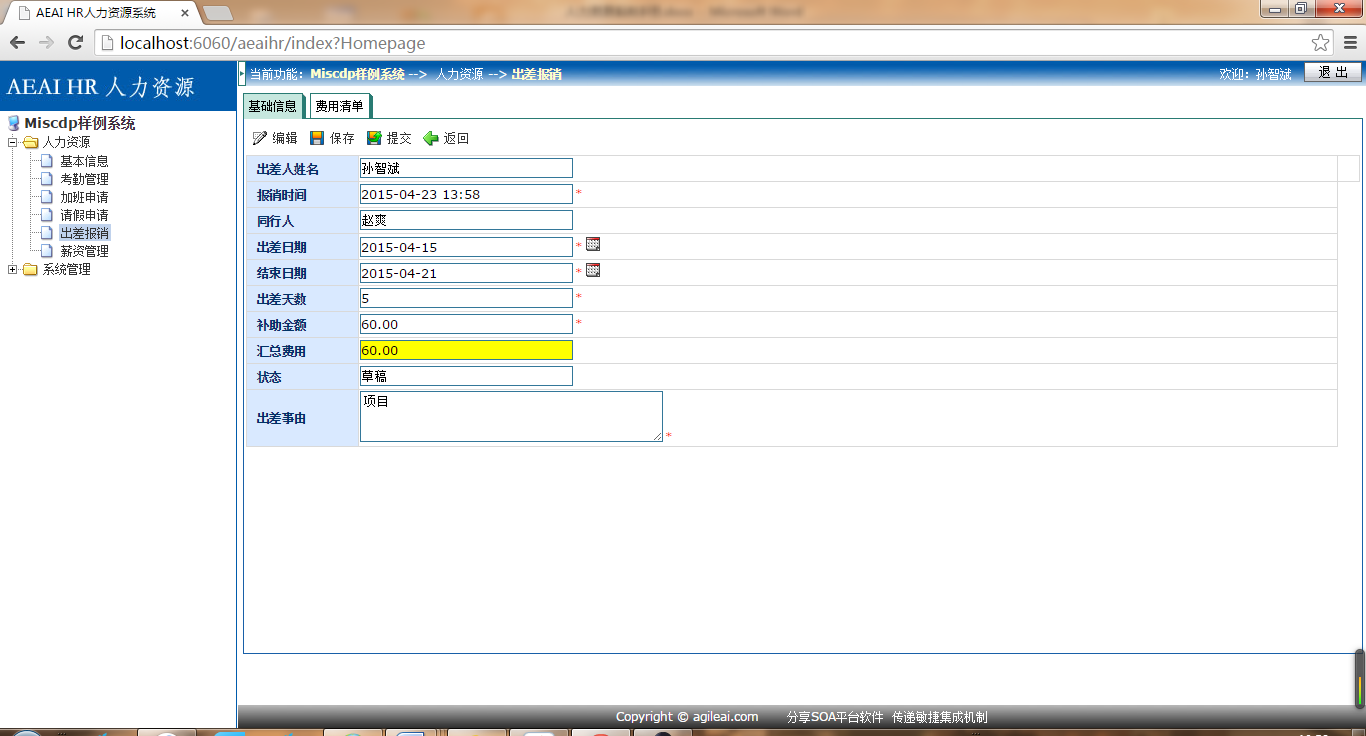


1. 测试运行

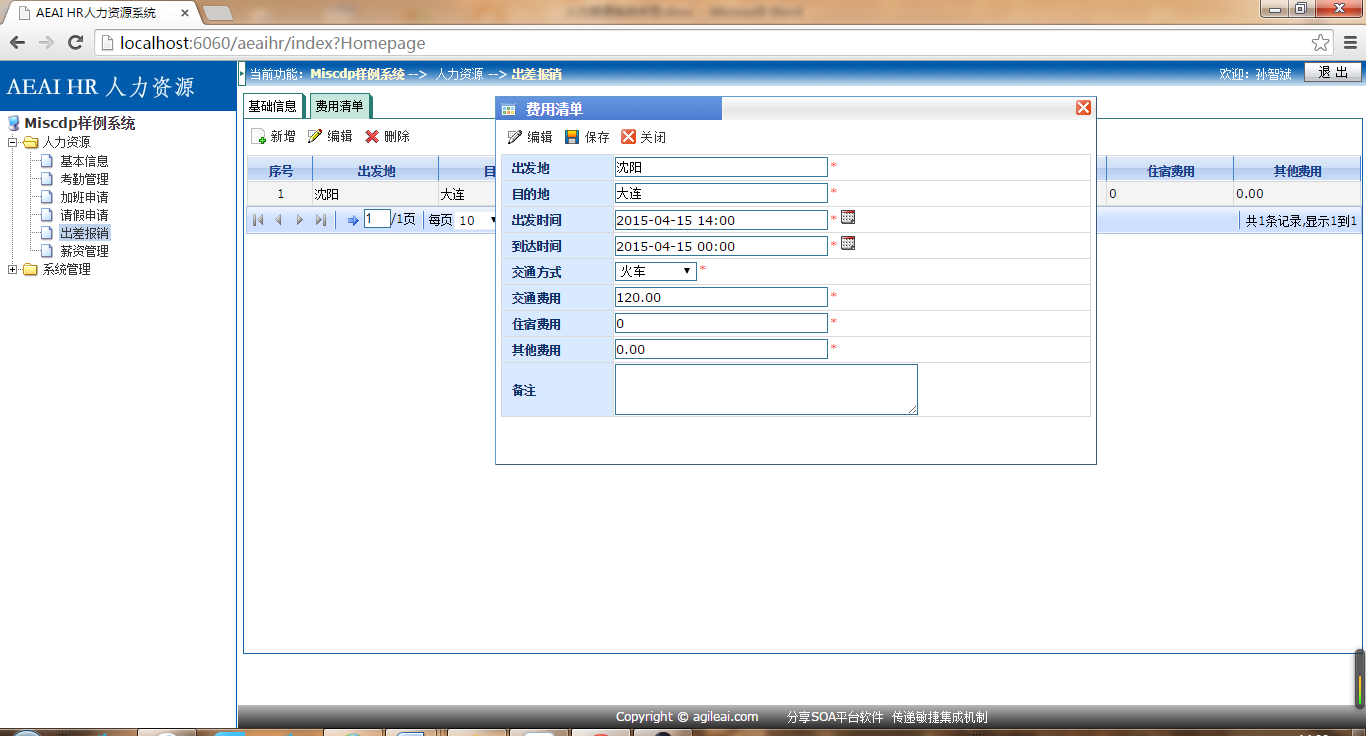
测试运行，效果如下图：

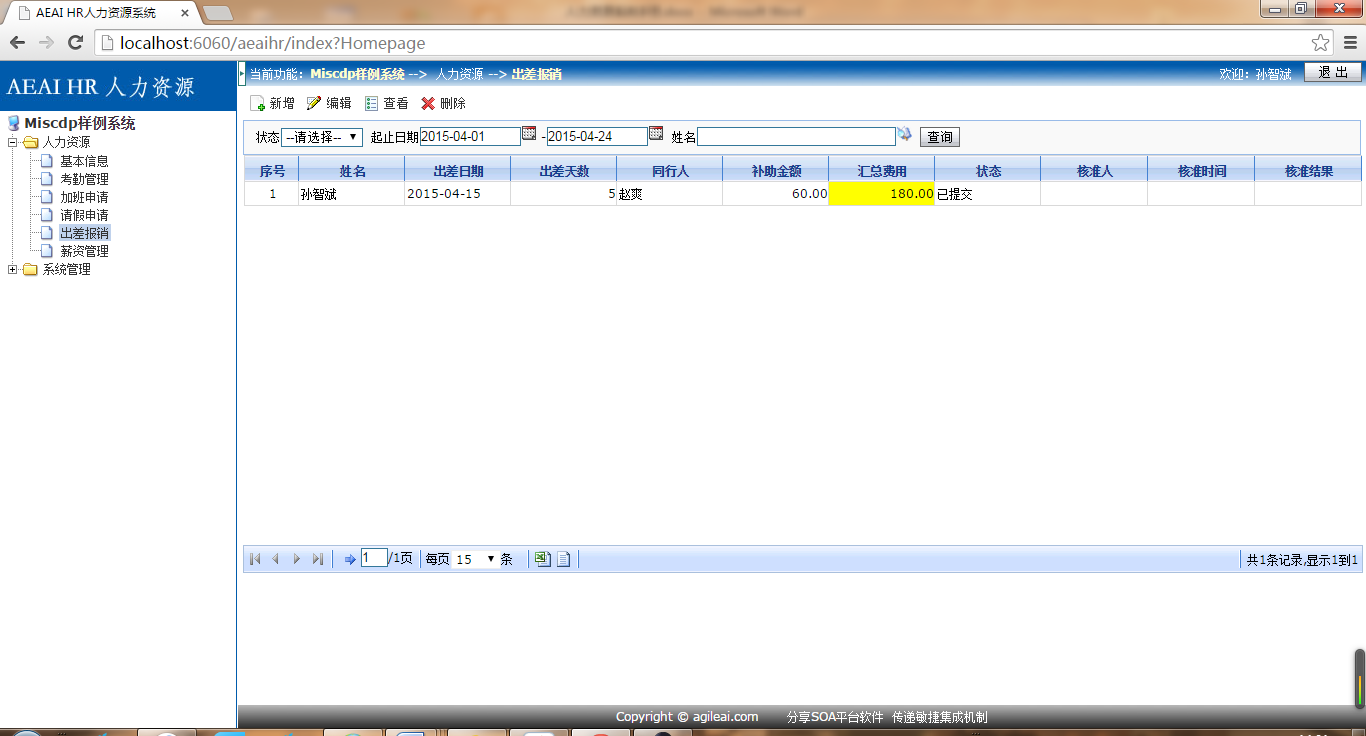


新增记录：

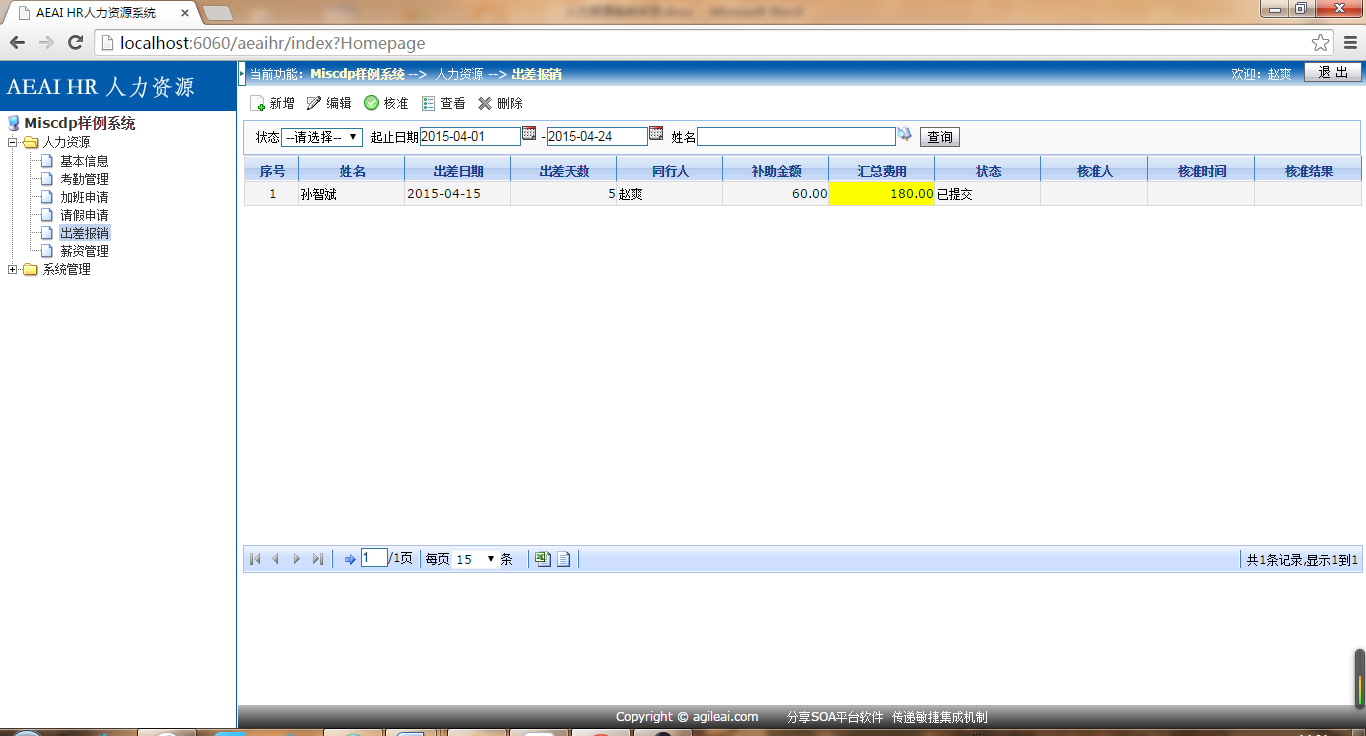


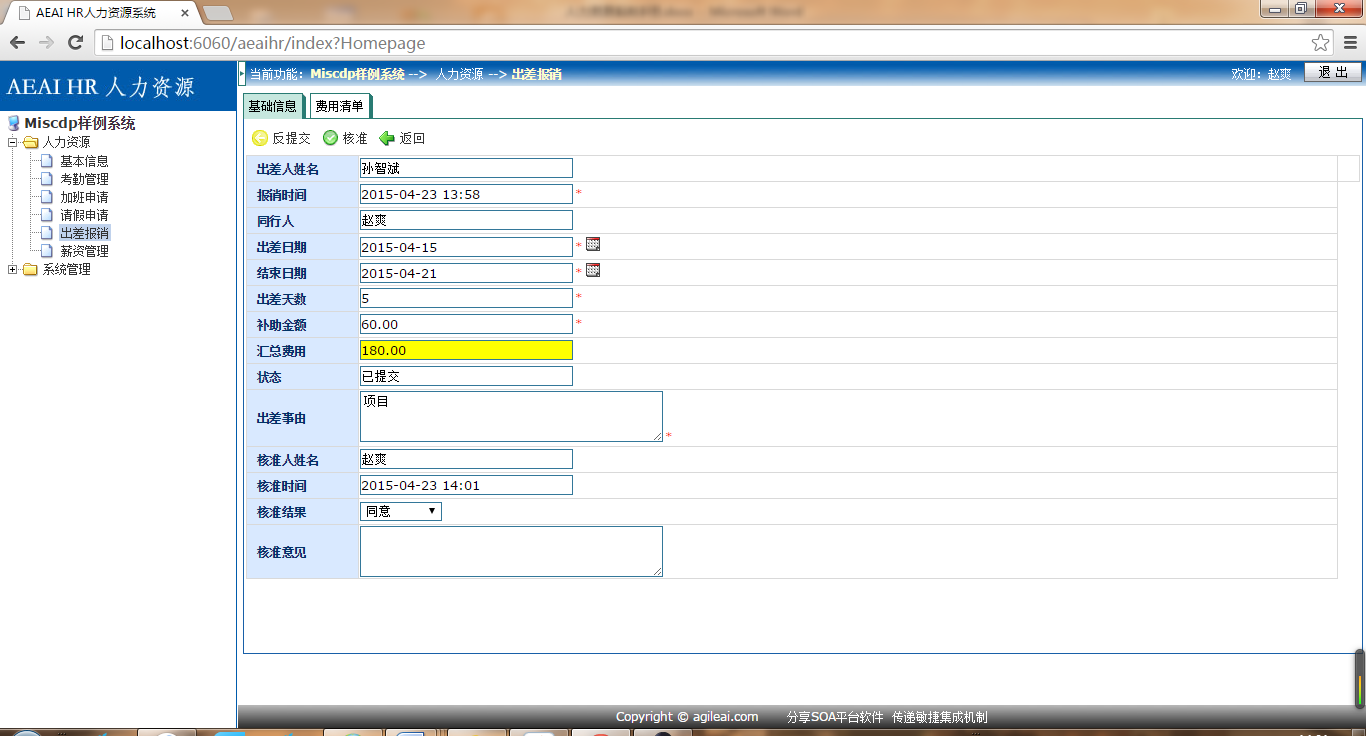
费用清单填写：

 提交后：

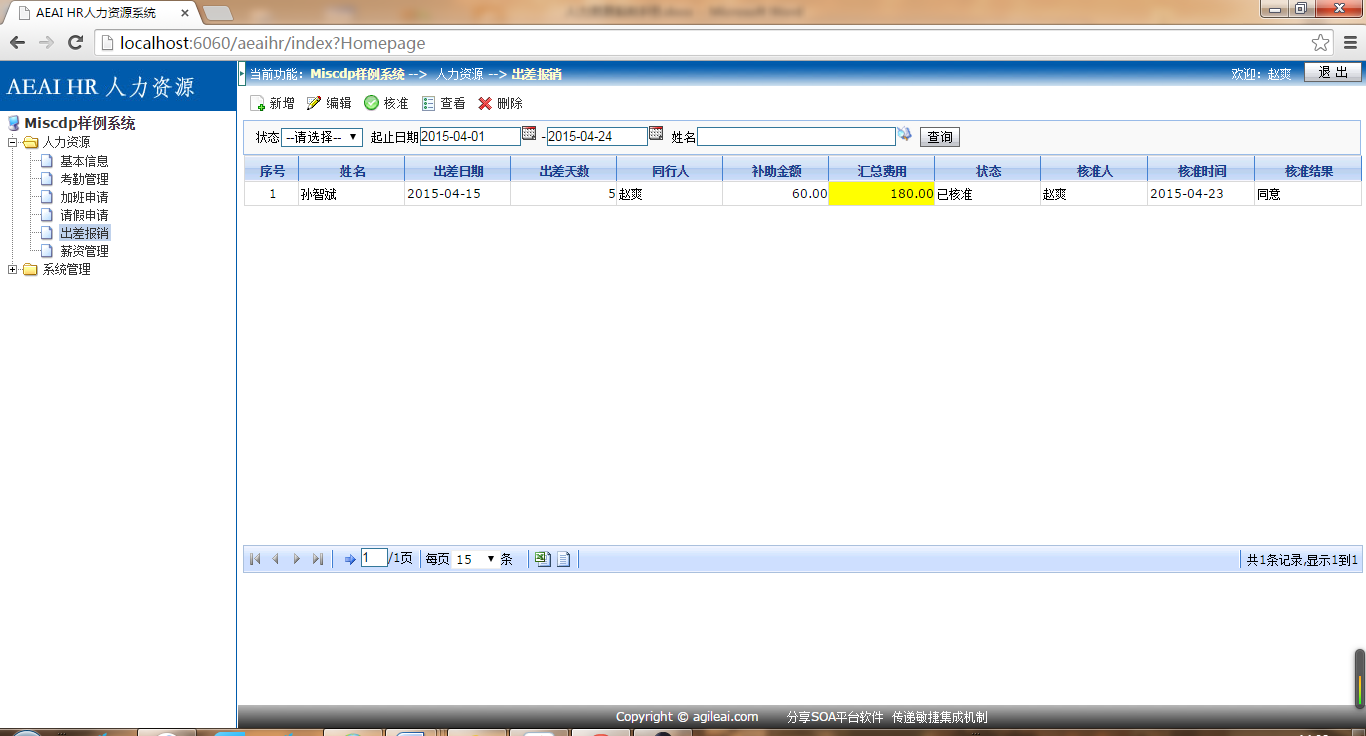


核准人登录核准：





核准后：



### 注意事项

和加班、请假不同，非权限用户可以看到其他用户的信息。

总费用是通过补助费用和费用清单中的费用相加的总和。

## 薪资管理

### 功能需求

1. 在列表页面显示出来员工的工作天数、出勤天数、加班天数、请假天数、以及工资保险等
2. 点击汇总按钮自动将上个月的所有员工的工资计算出来
3. 汇总完成后，可以编辑记录，填写奖金数值。
4. 一旦核准，再次汇总时不会改变记录，可以通过反核准按钮将薪资状态改变为初始化状态。
5. 只有薪资核准人才可以看到全部员工的薪资信息，普通职员登录只能看到自己的薪资信息，且只能查看。

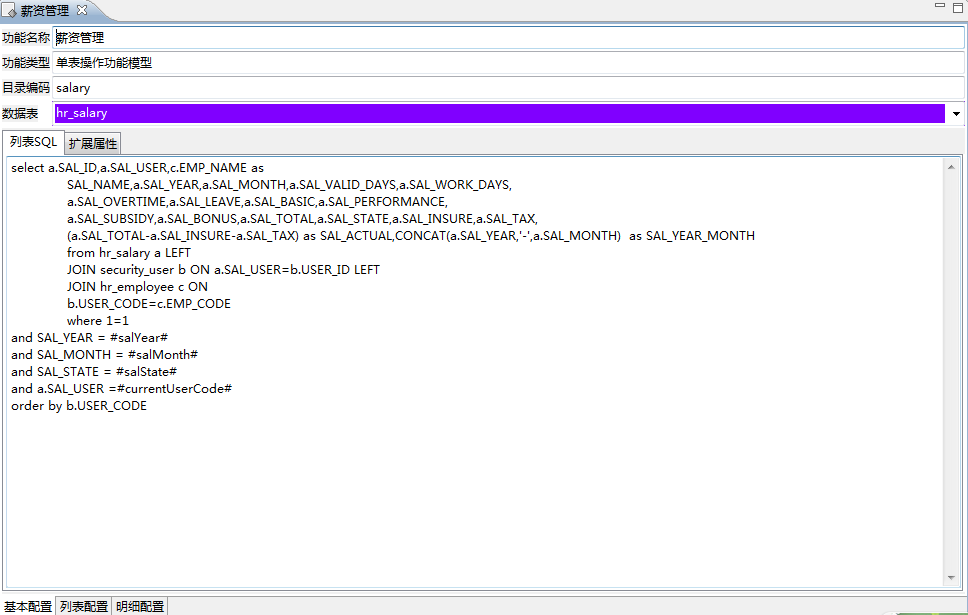
### 开发思路

1. 通过关联查询基本信息表、考勤表、加班表、请假表来找到基本工资、绩效工资、补贴、保险金、个人所得税、出勤天数、加班天数、请假天数等相关的字段。
2. 需要在jsp页面的汇总按钮修改action，在listhandler中添加对应的pageAction，也需要添加新的sql语句，这样就需要在HrSalaryManage类中添加方法，在接口实现类中将方法具体化。
3. 双击进入编辑页面时除了奖金，其他项都为只读。
4. 需要在sql语句中完成筛选，在jsp中添加反核准按钮，需要在sql中添加对应的语句，在接口实现类中实现具体的方法，在listHandler和EditHandler中添加pageAction。
5. 这个在登陆的时候进行判断，需要在listHandler中添加。

### 扩展设计

1. 创建单表操作模型

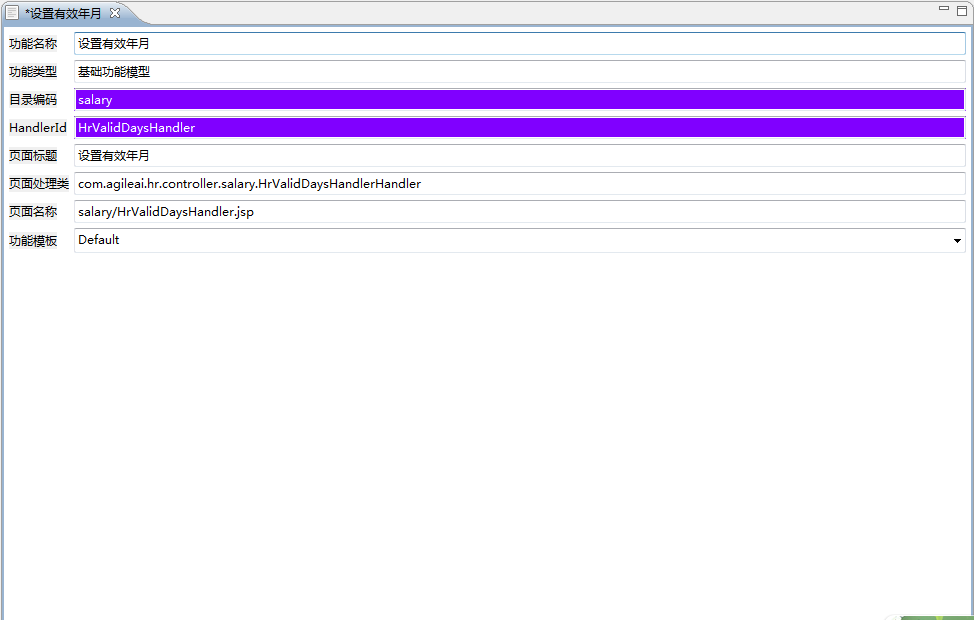
右击‘业务管理’🡪创建功能🡪单表操作模型🡪名称‘薪资管理’🡪目录编码salary数据表hr\_salary，列表sql如图：



由于列表sql中做了关联查询，列表配置，和明细配置在下文中会提到

右击‘业务管理’🡪创建功能🡪基础功能模型🡪名称‘设置有效年月’

其他信息如图：



1. 扩展开发
2. 手动设置弹出框并获取查询时间的默认值

首先需要在列表jsp页面编写js代码，定义方法setValidDays()，弹出有效天数的标记页面。然后获取默认值，将year的值截取查询时间(从第0位开始截取4位为year，第5位到7位为month)。

List的Jsp页面的代码：

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* align=*"center"* onclick="setValidDays()" ><input value=*"&nbsp;"* title=*"设置有效天数"* type=*"button"* class=*"editImgBtn"* />设置有效天数</td>

上面的js需要添加：

**var** setValidDaysBox;

**function** setValidDays(){

**var** title = '有效天数';

**if** (!setValidDaysBox){

setValidDaysBox = **new** PopupBox('setValidDaysBox',title,{size:'normal',height:'300px',top:'10px'});

}

**var** year = $("#salDate").val().substring(0,4);

**var** month = $("#salDate").val().substring(5,7);

**var** url = "index?HrValidDays&actionType=prepareDisplay&year="+year+"&month="+month;

setValidDaysBox.sendRequest(url);

}

在HrValidDaysHandler页面需要添加代码：

**protected** **void** processPageAttributes(DataParam param){

**this**.setAttribute("VALID\_YEAR",**this**.getAttribute("VALID\_YEAR",param.get("year")));

**this**.setAttribute("VALID\_MONTH",**this**.getAttribute("VALID\_MONTH",param.get("month")));

}

**public** ViewRenderer doSaveAction(DataParam param){

String responseText = *FAIL*;

**try** {

HrSalaryManage hrSalaryManage = **this**.lookupService(HrSalaryManage.**class**);

String operateType = param.get(OperaType.*KEY*);

**if** (OperaType.*CREATE*.equals(operateType)){

hrSalaryManage.createValidDayRecord(param);

}

**else** **if**(OperaType.*UPDATE*.equals(operateType)){

hrSalaryManage.updateValidDayRecord(param);

}

responseText = *SUCCESS*;

} **catch** (Exception e) {

log.error(e.getLocalizedMessage(), e);

}

**return** **new** AjaxRenderer(responseText);

}

}

在HrSalaryManageListHandler类中的preparedisplay中添加语句

**public** ViewRenderer prepareDisplay(DataParam param) {

User user = (User) **this**.getUser();

PrivilegeHelper privilegeHelper = **new** PrivilegeHelper(user);

**if** (!privilegeHelper.isSalMaster()) {

param.put("currentUserCode", user.getUserId());

setAttribute("hasRight", **true**);

} **else** {

param.put("currentUserCode", "");

setAttribute("hasRight", **false**);

}

mergeParam(param);

initParameters(param);

setAttribute("canGather", **true**);

String salDate = param.get("salDate");

String paramDate = salDate.substring(0, 7);

Date date = DateUtil.*getBeginOfMonth*(**new** Date());

Date beforeDate = DateUtil.*getDateAdd*(date, DateUtil.*MONTH*, -1);

String currentDate = DateUtil.*getDateByType*(DateUtil.*YYMMDD\_HORIZONTAL*,

beforeDate);

String currenDate = currentDate.substring(0, 7);

String year = salDate.substring(0, 4);

String month = salDate.substring(5, 7);

DataRow validDays = getService().retrieveValidDays(year, month);

**if**(validDays == **null**){

setAttribute("validDays", **false**);

}**else**{

setAttribute("validDays", **true**);

}

param.put("salYear", year, "salMonth", month);

**this**.setAttributes(param);

List<DataRow> rsList = getService().findRecords(param);

**this**.setRsList(rsList);

processPageAttributes(param);

**return** **new** LocalRenderer(getPage());

}

需要在HrSalaryManageImpl类添加代码了，首先定义retrieveValidDays方法，将有效天数的表里的字段查询出来。下面的是插入方法和更新方法。和添加对用的sqlmap

@Override

**public** DataRow retrieveValidDays(String year, String month) {

DataParam param = **new** DataParam("year", year, "month", month);

String statementId = sqlNameSpace + "." + "retrieveValidDays";

DataRow row = **this**.daoHelper.getRecord(statementId, param);

**return** row;

}

**public** **void** createValidDayRecord(DataParam param) {

String statementId = sqlNameSpace + "." + "insertValidDaysRecord";

processDataType(param, tableName);

processPrimaryKeys(param);

**this**.daoHelper.insertRecord(statementId, param);

}

**public** **void** updateValidDayRecord(DataParam param) {

String statementId = sqlNameSpace + "." + "updateValidDaysRecord";

processDataType(param, tableName);

**this**.daoHelper.updateRecord(statementId, param);

}

在sql中需要添加相应的sql语句：

<insert id="insertValidDaysRecord" parameterClass="com.agileai.domain.DataParam">

insert into

hr\_valid (VALID\_YEAR,VALID\_MONTH,VALID\_DAYS) values

(#VALID\_YEAR:VARCHAR#,#VALID\_MONTH:VARCHAR#,#VALID\_DAYS:DECIMAL#)

</insert>

<update id="updateValidDaysRecord" parameterClass="com.agileai.domain.DataParam">

update

hr\_valid

set

VALID\_YEAR=#VALID\_YEAR:VARCHAR#,VALID\_MONTH=#VALID\_MONTH:VARCHAR#,VALID\_DAYS=#VALID\_DAYS:DECIMAL#

where VALID\_YEAR=#VALID\_YEAR# and VALID\_MONTH=#VALID\_MONTH#

</update>

1. 汇总按钮的实现

在jsp中的汇总按钮的修改：

<td onmouseover="onMover(this);" onmouseout="onMout(this);" class=*"bartdx"* hotKey=*"A"* align=*"center"* onclick="gatherData()" ><input id=*"gatherImgBtn"* value=*"&nbsp;"* title=*"汇总"* type=*"button"* class=*"createImgBtn"* />汇总</td>

在上面的js中添加语句：

**function** gatherData(){

doSubmit({actionType:'gather'});

}

下一步进入列表页面的handler里，添加方法，截取年和月后将值传入到service实现类的gatherData方法里：

@PageAction

**public** ViewRenderer gather(DataParam param) {

String salDate = param.get("salDate");

String year = salDate.substring(0, 4);

String month = salDate.substring(5, 7);

getService().gatherData(year, month);

**return** **this**.prepareDisplay(param);

}

通过查询员工基本信息表的信息，获取到信息后将和hr\_salary表中相同的内容放入进去。

重点介绍下出勤天数，加班天数和请假天数的获取方法，首先通过sql查询到对应的USER\_ID和天数，然后将通过和EMP\_CODE比较对应的相同的CODE将天数传进去。

需要在接口实现类中添加语句：

@Override

**public** **void** gatherData(String year, String month) {

String statementId = sqlNameSpace + "." + "findMasterDaysRecords";

List<DataRow> basicRecords = **this**.daoHelper.queryRecords(statementId,

**new** DataParam("year",year,"month",month));

KeyGenerator keyGenerator = **new** KeyGenerator();

DataRow validDaysRow = **this**.retrieveValidDays(year, month);

String yearMonth = year + "-" + month;

statementId = sqlNameSpace + "." + "workDayRecords";

HashMap<String, DataRow> workDaysMap = **this**.daoHelper.queryRecords(

"USER\_ID", statementId, **new** DataParam("yearMonth", yearMonth));

statementId = sqlNameSpace + "." + "leaveDayRecords";

HashMap<String, DataRow> leaveDaysMap = **this**.daoHelper.queryRecords(

"USER\_ID", statementId, **new** DataParam("yearMonth", yearMonth));

statementId = sqlNameSpace + "." + "overTimeDayRecords";

HashMap<String, DataRow> overTimeDaysMap = **this**.daoHelper.queryRecords(

"USER\_ID", statementId, **new** DataParam("yearMonth", yearMonth));

statementId = sqlNameSpace + "." + "existsDataRecords";

HashMap<String, DataRow> existsDataMap = **this**.daoHelper.queryRecords(

"SAL\_USER", statementId, **new** DataParam("year",year,"month",month));

List<DataParam> insertParamList = **new** ArrayList<DataParam>();

List<DataParam> updateParamList = **new** ArrayList<DataParam>();

**for** (**int** i = 0; i < basicRecords.size(); i++) {

DataRow row = basicRecords.get(i);

DataParam dataParam = **new** DataParam();

dataParam.put("SAL\_ID", keyGenerator.genKey());

String userCode = row.getString("USER\_ID");

DataRow workDaysRow = workDaysMap.get(userCode);

**if**(workDaysRow == **null**){

dataParam.put("SAL\_WORK\_DAYS",**new** BigDecimal("0.0"));

}**else**{

dataParam.put("SAL\_WORK\_DAYS", workDaysRow.get("WORK\_DAYS"));

}

DataRow leaveDaysRow = leaveDaysMap.get(userCode);

**if**(leaveDaysRow == **null**){

dataParam.put("SAL\_LEAVE", **new** BigDecimal("0.0"));

}**else**{

dataParam.put("SAL\_LEAVE", leaveDaysRow.get("LEAVE\_DAYS"));

}

DataRow overTimeDaysRow = overTimeDaysMap.get(userCode);

**if**(overTimeDaysRow == **null**){

dataParam.put("SAL\_OVERTIME", **new** BigDecimal("0.0"));

}**else**{

dataParam.put("SAL\_OVERTIME", overTimeDaysRow.get("WOT\_DAYS"));

}

dataParam.put("SAL\_VALID\_DAYS", validDaysRow.get("VALID\_DAYS"));

dataParam.put("SAL\_USER", userCode);

dataParam.put("SAL\_YEAR", year);

dataParam.put("SAL\_MONTH", month);

dataParam.put("SAL\_BASIC", row.get("EMP\_BASIC"));

dataParam.put("SAL\_PERFORMANCE", row.get("EMP\_PERFORMANCE"));

dataParam.put("SAL\_SUBSIDY", row.get("EMP\_SUBSIDY"));

dataParam.put("SAL\_INSURE", row.get("EMP\_INSURE"));

dataParam.put("SAL\_TAX", row.get("EMP\_TAX"));

dataParam.put("SAL\_STATE", "0");

BigDecimal sal\_basic = (BigDecimal) row.get("EMP\_BASIC");

BigDecimal sal\_performance = (BigDecimal) row

.get("EMP\_PERFORMANCE");

BigDecimal sal\_subsidy = (BigDecimal) row.get("EMP\_SUBSIDY");

dataParam.put("SAL\_BONUS", "0.00");

BigDecimal sal\_total = sal\_basic.add(sal\_performance).add(

sal\_subsidy);

dataParam.put("SAL\_TOTAL", sal\_total);

**if** (existsDataMap.containsKey(userCode)){

updateParamList.add(dataParam);

}**else**{

insertParamList.add(dataParam);

}

}

同时需要在sql中添加对应的sql：

<select id="workDayRecords" parameterClass="com.agileai.domain.DataParam"

resultClass="com.agileai.domain.DataRow">

SELECT USER\_ID,COUNT(ATD\_DATE) as WORK\_DAYS

FROM hr\_attendance

WHERE

1=1

<isNotEmpty prepend=*" "* property=*"userCode"*>

and USER\_ID = #userCode#

</isNotEmpty>

<isNotEmpty prepend=*" "* property=*"yearMonth"*>

and LEFT(ATD\_DATE,7) =

#yearMonth#

</isNotEmpty>

GROUP BY USER\_ID

</select>

<select id="leaveDayRecords" parameterClass="com.agileai.domain.DataParam"

resultClass="com.agileai.domain.DataRow">

SELECT USER\_ID,SUM(LEA\_DAYS) as LEAVE\_DAYS

FROM hr\_leave

WHERE 1=1

<isNotEmpty prepend=*" "* property=*"userCode"*>

and USER\_ID = #userCode#

</isNotEmpty>

<isNotEmpty prepend=*" "* property=*"yearMonth"*>

and LEFT(LEA\_SDATE,7) =

#yearMonth#

</isNotEmpty>

GROUP BY USER\_ID

</select>

<select id="overTimeDayRecords" parameterClass="com.agileai.domain.DataParam"

resultClass="com.agileai.domain.DataRow">

SELECT USER\_ID,SUM(WOT\_TIME) as WOT\_DAYS

FROM hr\_work\_overtime

WHERE

1=1

<isNotEmpty prepend=*" "* property=*"userCode"*>

and USER\_ID = #userCode#

</isNotEmpty>

<isNotEmpty prepend=*" "* property=*"yearMonth"*>

and LEFT(WOT\_DATE,7) =

#yearMonth#

</isNotEmpty>

GROUP BY USER\_ID

</select>

<select id="existsDataRecords" parameterClass="com.agileai.domain.DataParam"

resultClass="com.agileai.domain.DataRow">

select

a.SAL\_ID,a.SAL\_USER,c.EMP\_NAME as

SAL\_NAME,a.SAL\_YEAR,a.SAL\_MONTH,a.SAL\_VALID\_DAYS,a.SAL\_WORK\_DAYS,a.SAL\_OVERTIME,a.SAL\_LEAVE,

a.SAL\_BASIC,a.SAL\_PERFORMANCE,a.SAL\_SUBSIDY,a.SAL\_BONUS,a.SAL\_TOTAL,a.SAL\_STATE,a.SAL\_INSURE,a.SAL\_TAX

from hr\_salary a LEFT JOIN security\_user b ON a.SAL\_USER=b.USER\_ID

LEFT

JOIN hr\_employee c ON b.USER\_CODE = c.EMP\_CODE

where 1=1

<isNotEmpty prepend=*" "* property=*"year"*>

and SAL\_YEAR = #year#

</isNotEmpty>

<isNotEmpty prepend=*" "* property=*"month"*>

and SAL\_MONTH = #month#

</isNotEmpty>

order by SAL\_YEAR,SAL\_MONTH,SAL\_USER

</select>

1. 时间的年和月查询

在listHandler中的preparedisplay方法中添加语句，在页面刷新时直接传入年月。

**public** ViewRenderer prepareDisplay(DataParam param) {

User user = (User) **this**.getUser();

PrivilegeHelper privilegeHelper = **new** PrivilegeHelper(user);

**if** (!privilegeHelper.isSalMaster()) {

param.put("currentUserCode", user.getUserId());

setAttribute("hasRight", **true**);

} **else** {

param.put("currentUserCode", "");

setAttribute("hasRight", **false**);

}

mergeParam(param);

initParameters(param);

setAttribute("canGather", **true**);

String salDate = param.get("salDate");

String paramDate = salDate.substring(0, 7);

Date date = DateUtil.*getBeginOfMonth*(**new** Date());

Date beforeDate = DateUtil.*getDateAdd*(date, DateUtil.*MONTH*, -1);

String currentDate = DateUtil.*getDateByType*(DateUtil.*YYMMDD\_HORIZONTAL*,

beforeDate);

String currenDate = currentDate.substring(0, 7);

**if** (paramDate.equals(currenDate)) {

setAttribute("canGather", **true**);

} **else** {

setAttribute("canGather", **false**);

}

String year = salDate.substring(0, 4);

String month = salDate.substring(5, 7);

DataRow validDays = getService().retrieveValidDays(year, month);

**if**(validDays == **null**){

setAttribute("validDays", **false**);

}**else**{

setAttribute("validDays", **true**);

}

param.put("salYear", year, "salMonth", month);

**this**.setAttributes(param);

List<DataRow> rsList = getService().findRecords(param);

**this**.setRsList(rsList);

processPageAttributes(param);

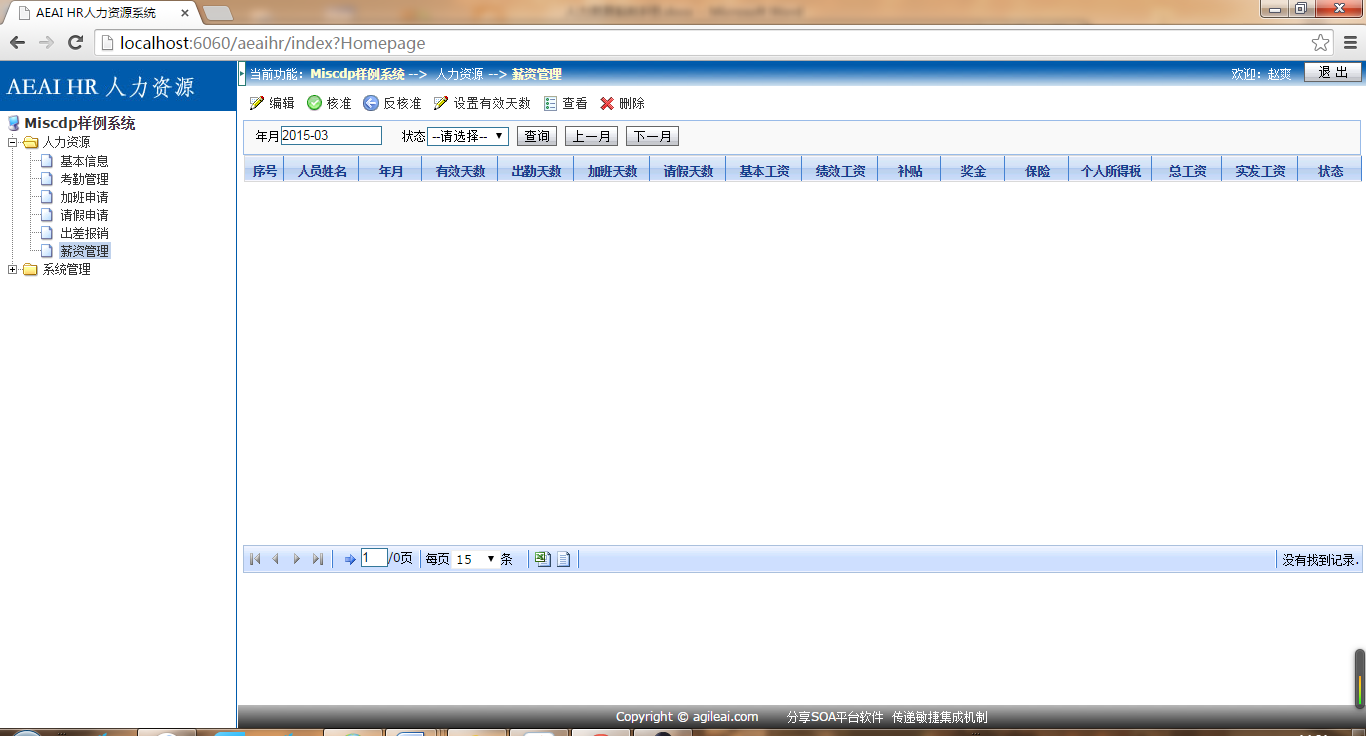
**return** **new** LocalRenderer(getPage());

}

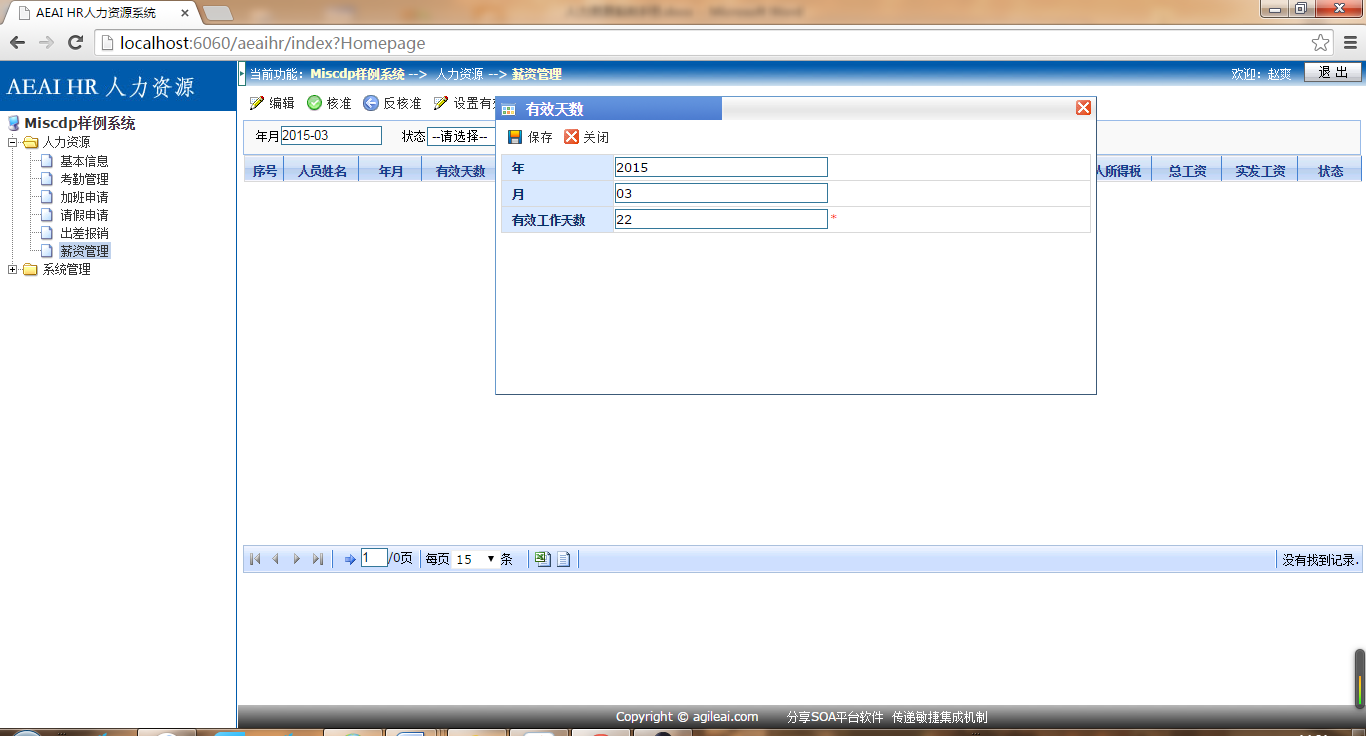
1. 测试运行

测试运行，效果如下图：

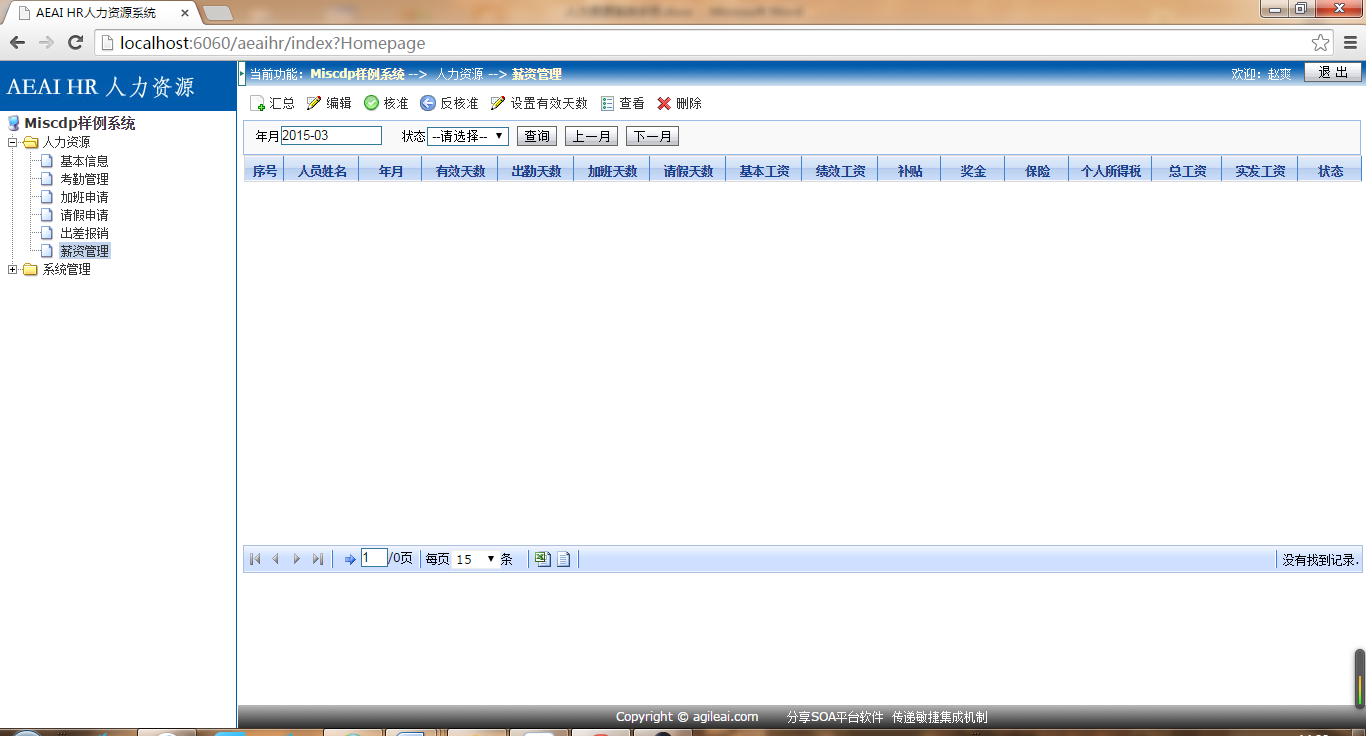
未设置有效天数：



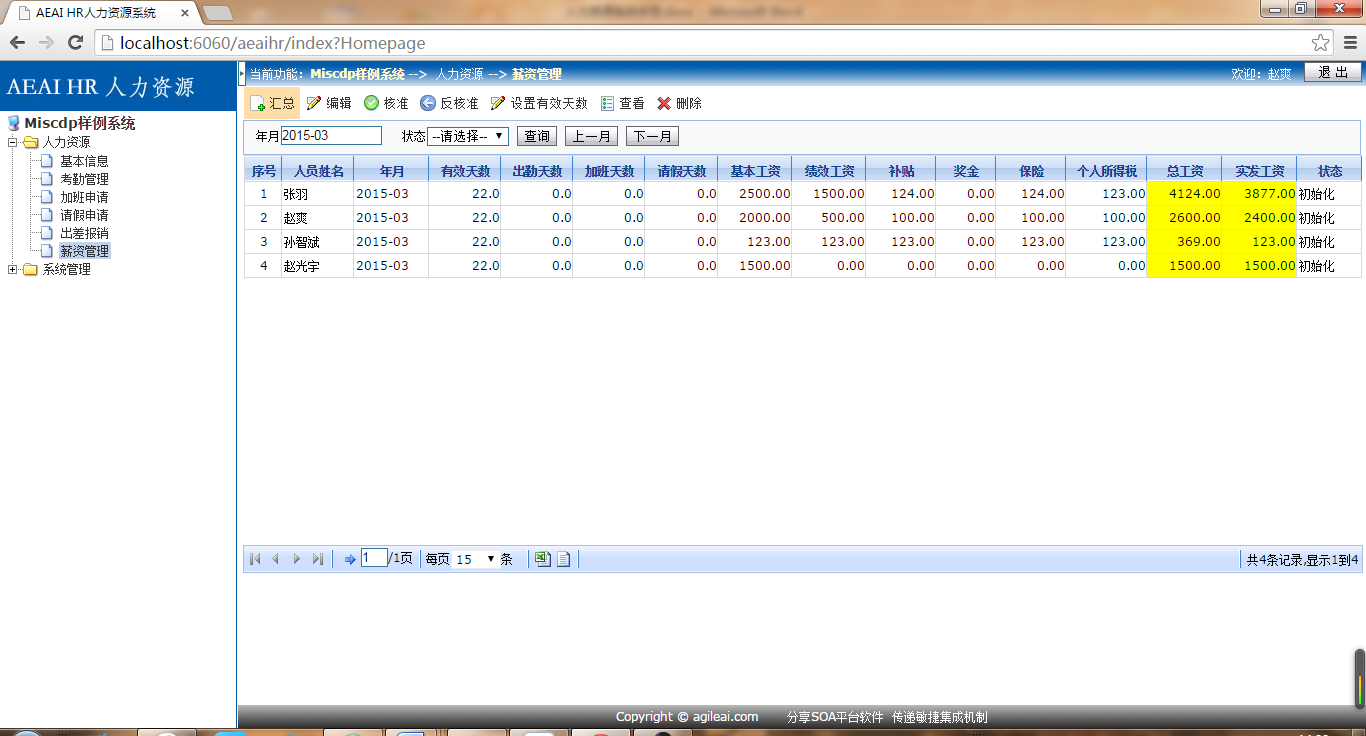
设置有效天数：



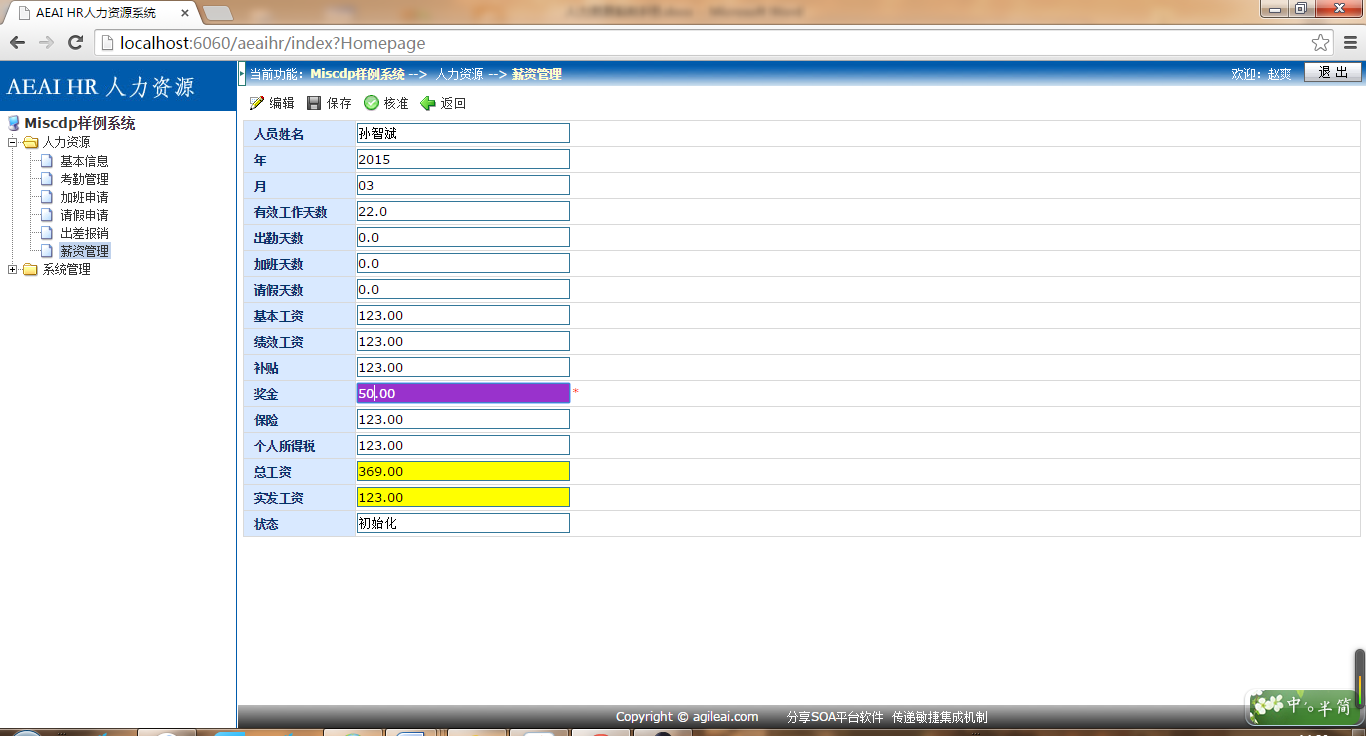
设置完成后：



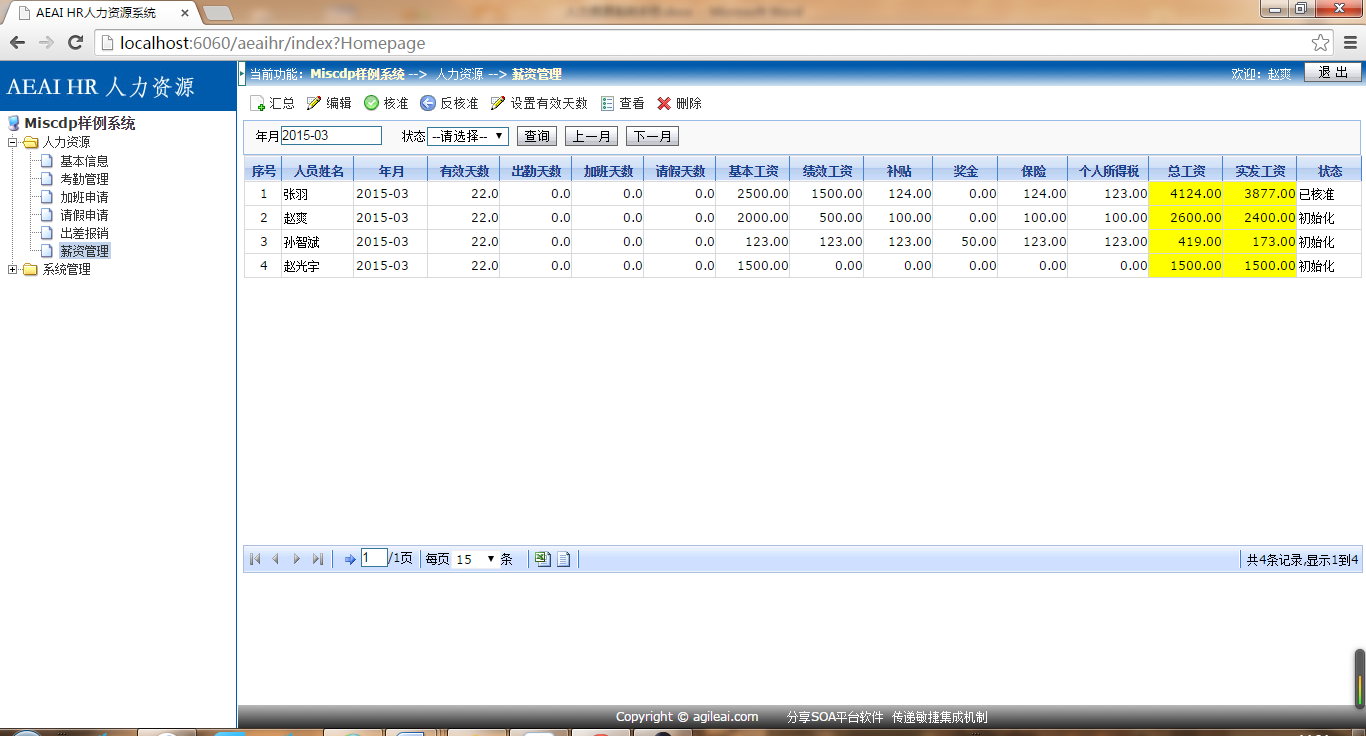
点击汇总后：



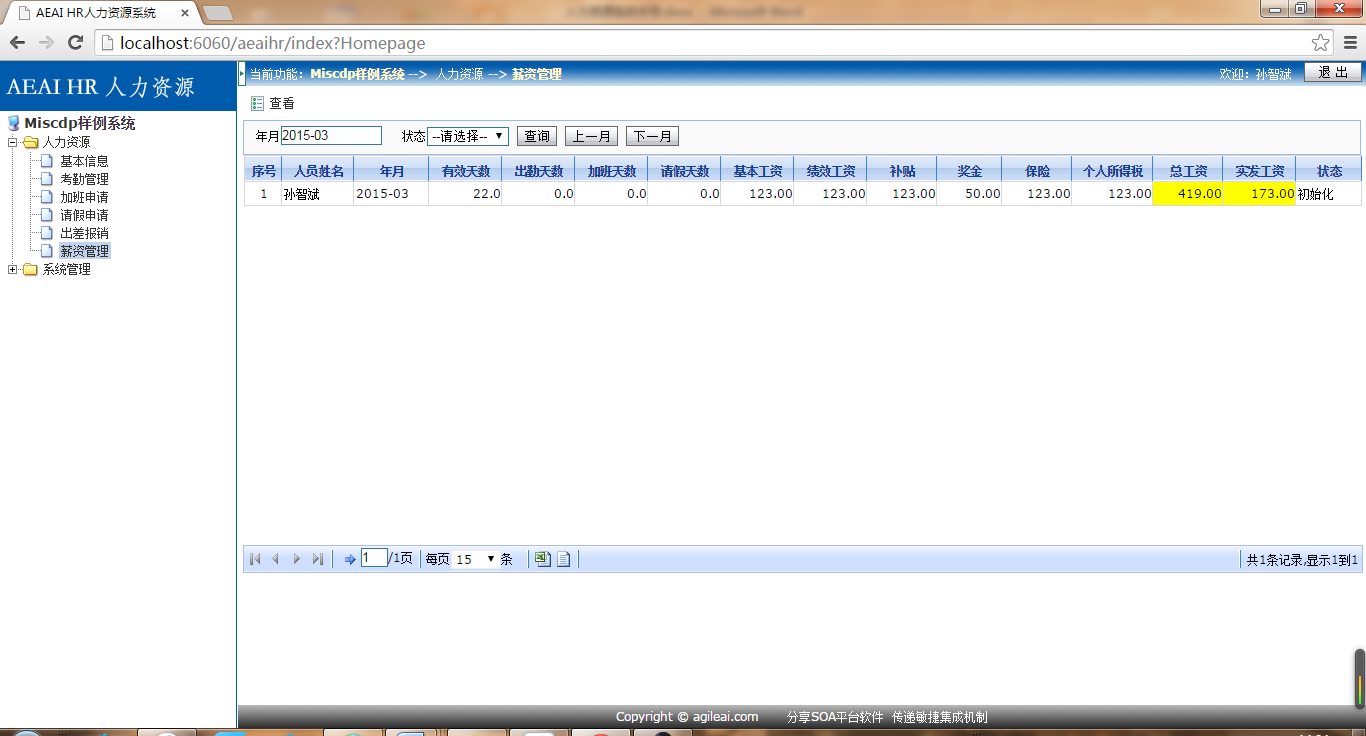
核准人可编辑记录中的奖金：



核准后，再次汇总，记录不变：



非核准人进入后只能查看：



### 注意事项

1. 汇总第一次做新增操作，第二次点击时做更新操作。
2. 在汇总之前需要先设置有效天数，若不设置天数，汇总按钮不显示。
3. 在查询其他月的薪资信息时，不显示汇总按钮。

# 数据库设计

数据库设计详细说明参见《数据库设计》文档。