package com.bizlogic.cz.action;

import java.util.Date;

import java.util.List;

import org.apache.commons.lang.StringUtils;

import com.bizlogic.base.CertificateType;

import com.bizlogic.base.Country;

import com.bizlogic.base.Political;

import com.bizlogic.base.User;

import com.bizlogic.base.front.action.BaseUserEditAction;

import com.bizlogic.common.Element;

import com.bizlogic.cz.bean.DomicileType;

import com.bizlogic.cz.bean.Homeowner;

import com.bizlogic.cz.bean.Rental;

public class HomeownerEditAction extends BaseUserEditAction {

//出租房屋编号

private String rentalId;

//证件类型编号

private String certificateTypeId;

// 证件类型

private List certificateTypeList;

public List getCertificateTypeList() {

return super.getDefaultManager().findAll("from CertificateType a where 1=1 order by a.id");

}

//国家编号

private String countryId;

//国籍

private List countryList;

public List getCountryList() {

return super.getDefaultManager().findAll("from Country a where 1=1 order by a.id");

}

//户籍地类型编号

private String domicileTypeId;

//户籍地类型

private List domicileTypeList;

public List getDomicileTypeList() {

return super.getDefaultManager().findAll("from DomicileType a where 1=1 order by a.id");

}

//政治面貌编号

private String politicalId;

//政治面貌列表

private List politicalList;

public List getPoliticalList() {

return super.getDefaultManager().findAll("from Political a where 1=1 order by a.id");

}

protected Element createElement() {

Homeowner hromeowner = new Homeowner();

hromeowner.setBirthday(new Date());

return hromeowner;

}

protected void prepareShow() {

Homeowner hromeowner = (Homeowner) element;

//将rentalId设置session

rentalId = super.getParameterValue("rentalId");

if(rentalId != null){

super.getSession().setAttribute("rentalId",rentalId);

}else{

rentalId = (String)super.getSession().getAttribute("rentalId");

}

if(hromeowner.getId() != null){

this.certificateTypeId = hromeowner.getCertificateType() == null ? null : hromeowner.getCertificateType().getId();

this.countryId = hromeowner.getCountry() == null ? null : hromeowner.getCountry().getId();

this.domicileTypeId = hromeowner.getDomicileType() == null ? null : hromeowner.getDomicileType().getId();

this.politicalId = hromeowner.getPolitical() == null ? null : hromeowner.getPolitical().getId();

}else{

countryId = (super.getDefaultManager().get(Country.class, "CN")).getId();

}

}

protected boolean prepareSave(Element element) {

Homeowner hromeowner = (Homeowner) element;

//设置出租房屋

hromeowner.setRental(super.getDefaultManager().get(Rental.class, rentalId));

//当新增时，如果房主为个人则添加个人相关信息

if(hromeowner.getId() == null && hromeowner.getHomeownerType() == 0){

if (StringUtils.isNotBlank(certificateTypeId)) {

hromeowner.setCertificateType((CertificateType) super.getDefaultManager().get(CertificateType.class,certificateTypeId));

} else {

hromeowner.setCertificateType(null);

}

if (StringUtils.isNotBlank(countryId)) {

hromeowner.setCountry((Country) super.getDefaultManager().get(Country.class,countryId));

} else {

hromeowner.setCountry(null);

}

if (StringUtils.isNotBlank(domicileTypeId)) {

hromeowner.setDomicileType((DomicileType) super.getDefaultManager().get(DomicileType.class,domicileTypeId));

} else {

hromeowner.setDomicileType(null);

}

if (StringUtils.isNotBlank(politicalId)) {

hromeowner.setPolitical((Political) super.getDefaultManager().get(Political.class,politicalId));

} else {

hromeowner.setPolitical(null);

}

//当修改房主时，如果房主为个人，则清空单位房主信息

}else if(hromeowner.getId() != null && hromeowner.getHomeownerType() == 0){

//清空单位房主信息

hromeowner.setCompanyName(null);

hromeowner.setCompanyAddress(null);

hromeowner.setCompanyUser(null);

hromeowner.setCompanyUserPhone(null);

if (StringUtils.isNotBlank(certificateTypeId)) {

hromeowner.setCertificateType((CertificateType) super.getDefaultManager().get(CertificateType.class,certificateTypeId));

} else {

hromeowner.setCertificateType(null);

}

if (StringUtils.isNotBlank(countryId)) {

hromeowner.setCountry((Country) super.getDefaultManager().get(Country.class,countryId));

} else {

hromeowner.setCountry(null);

}

if (StringUtils.isNotBlank(domicileTypeId)) {

hromeowner.setDomicileType((DomicileType) super.getDefaultManager().get(DomicileType.class,domicileTypeId));

} else {

hromeowner.setDomicileType(null);

}

if (StringUtils.isNotBlank(politicalId)) {

hromeowner.setPolitical((Political) super.getDefaultManager().get(Political.class,politicalId));

} else {

hromeowner.setPolitical(null);

}

//当修改房主信息时，如果房主为单位，则清空个人房主信息

}else if(hromeowner.getId() != null && hromeowner.getHomeownerType() == 1){

hromeowner.setName(null);

hromeowner.setGender(0);

hromeowner.setBirthday(null);

hromeowner.setHometown(null);

hromeowner.setAccountLocation(null);

hromeowner.setUserPhone(null);

hromeowner.setCertificateType(null);

hromeowner.setIdentityCardTypeNumber(null);

hromeowner.setCountry(null);

hromeowner.setDomicileType(null);

hromeowner.setPolitical(null);

}

hromeowner.setCreateUser((User)getVisit());

hromeowner.setCreateDate(new Date());

hromeowner.setStatus(1);

return super.prepareSave(hromeowner);

}

public String save() {

Homeowner hromeowner = (Homeowner) element;

super.save();

this.setId(element.getId());

//将出租房之前房主信息改为历史信息

super.getDefaultManager().execute("update Homeowner t set t.status=2 where t.status=1 and t.rental.id=? and t.id != ?", rentalId, hromeowner.getId());

return SUCCESS;

}

protected Class getElementClass() {

return Homeowner.class;

}

public String getCertificateTypeId() {

return certificateTypeId;

}

public void setCertificateTypeId(String certificateTypeId) {

this.certificateTypeId = certificateTypeId;

}

public String getCountryId() {

return countryId;

}

public void setCountryId(String countryId) {

this.countryId = countryId;

}

public String getRentalId() {

return rentalId;

}

public void setRentalId(String rentalId) {

this.rentalId = rentalId;

}

public String getPoliticalId() {

return politicalId;

}

public void setPoliticalId(String politicalId) {

this.politicalId = politicalId;

}

public String getDomicileTypeId() {

return domicileTypeId;

}

public void setDomicileTypeId(String domicileTypeId) {

this.domicileTypeId = domicileTypeId;

}

}

package com.bizlogic.cz.action;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.List;

import net.sf.json.JSONObject;

import org.apache.commons.lang.StringUtils;

import com.bizlogic.base.front.action.BaseUserListAction;

import com.bizlogic.common.action.helper.ListInfo;

import com.bizlogic.cz.bean.Homeowner;

public class HomeownerListAction extends BaseUserListAction{

//出租房屋编号

private String rentalId;

@Override

protected long doGetCount(ListInfo listInfo) {

//将rentalId设置session

rentalId = super.getParameterValue("rentalId");

if(rentalId != null){

super.getSession().setAttribute("rentalId",rentalId);

}else{

rentalId = (String)super.getSession().getAttribute("rentalId");

}

sqlCondition = this.getConditions(rentalId);

return super.getDefaultManager().getCount("from Homeowner a where 1=1 "+sqlCondition);

}

@Override

protected List doSearch(ListInfo listInfo) {

List list = super.getDefaultManager().find("from Homeowner a where 1=1 "+sqlCondition + " order by a.createDate desc",

listInfo.getFirstNo(), listInfo.getMaxNo());

return list;

}

/\*\*

\* 删除函数(非物理删除)

\*/

@Override

protected void deleteObjectById(String homeownerId) {

super.getDefaultManager().execute("update Homeowner t set t.status=0 where t.id=?", homeownerId);

}

/\*\*

\* 获得查询条件

\*/

protected String getConditions(String rentalIdFlag)

{

//查询状态为1的有效数据和状态为2的历史数据

String conditionsStr = " and (a.status=1 or a.status=2)";

if(rentalIdFlag!=null && rentalIdFlag.compareTo("")!=0){

conditionsStr += " and a.rental.id='" + rentalIdFlag + "'";

}

return conditionsStr;

}

protected Class getImplementClass()

{

return Homeowner.class;

}

/\*\*

\* ajax请求

\* @throws IOException

\*/

public void ajaxMethod() throws IOException {

JSONObject res = new JSONObject();

//Response编码设置

super.getResponse().setCharacterEncoding("utf-8");

//获得old变更记录

if(StringUtils.equals(act, "get\_group\_user")) {

//返回html字符串

String returnHtml = "";

res.put("groupUserStr", returnHtml);

}

//结果输出

PrintWriter out = this.getResponse().getWriter();

out.println(res.toString());

out.flush();

out.close();

}

public String getRentalId() {

return rentalId;

}

public void setRentalId(String rentalId) {

this.rentalId = rentalId;

}

}

package com.bizlogic.cz.action;

import com.bizlogic.base.front.action.BaseUserEditAction;

import com.bizlogic.cz.bean.Homeowner;

public class HomeownerShowAction extends BaseUserEditAction{

//出租房屋编号

private String rentalId;

protected void prepareShow() {

Homeowner hromeowner = (Homeowner) element;

//将rentalId设置session

rentalId = super.getParameterValue("rentalId");

if(rentalId != null){

super.getSession().setAttribute("rentalId",rentalId);

}else{

rentalId = (String)super.getSession().getAttribute("rentalId");

}

}

protected Class getElementClass() {

return Homeowner.class;

}

public String getRentalId() {

return rentalId;

}

public void setRentalId(String rentalId) {

this.rentalId = rentalId;

}

}

package com.bizlogic.cz.action;

import java.util.Date;

import org.apache.commons.lang.StringUtils;

import com.bizlogic.base.User;

import com.bizlogic.base.front.action.BaseUserEditAction;

import com.bizlogic.common.Element;

import com.bizlogic.cz.bean.Household;

import com.bizlogic.cz.bean.Rental;

public class HouseholdEditAction extends BaseUserEditAction {

//房屋编号

private String rentalId;

protected void prepareShow() {

Household household = (Household) element;

if(household.getId()!=null){

rentalId=household.getRental().getId();

}

}

protected boolean prepareSave(Element element) {

Household household = (Household) element;

if (StringUtils.isNotBlank(rentalId)) {

household.setRental((Rental) super.getDefaultManager().get(Rental.class,rentalId));

} else {

household.setRental(null);

}

household.setStatus(1);

household.setCreateDate(new Date());

household.setUser((User)getVisit());

return super.prepareSave(household);

}

protected Class getElementClass() {

return Household.class;

}

public String getRentalId() {

return rentalId;

}

public void setRentalId(String rentalId) {

this.rentalId = rentalId;

}

}

package com.bizlogic.cz.action;

import java.util.List;

import com.bizlogic.base.front.action.BaseUserListAction;

import com.bizlogic.common.action.helper.ListInfo;

import com.bizlogic.cz.bean.Household;

import com.bizlogic.cz.bean.Tenant;

public class HouseholdListAction extends BaseUserListAction{

//房屋id

private String rentalId;

//地址

private String searchAddress;

@Override

protected long doGetCount(ListInfo listInfo) {

//将rentalId设置session

if(rentalId != null){

super.getSession().setAttribute("rentalId",rentalId);

}else{

rentalId = (String)super.getSession().getAttribute("rentalId");

}

sqlCondition = this.getConditions(1,searchAddress,rentalId);

return super.getDefaultManager().getCount("from Household a where 1=1 "+sqlCondition);

}

@Override

protected List doSearch(ListInfo listInfo) {

List list = super.getDefaultManager().find("from Household a where 1=1 "+sqlCondition+" order by a.id",

listInfo.getFirstNo(), listInfo.getMaxNo());

//list不为空时遍历

if(list!=null){

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

String strName="";

Household household=(Household)list.get(i);

//根据租户查询租户人员

List<Tenant> tenantList=super.getDefaultManager().findAll("from Tenant t where t.household=? and t.status=?", household,1);

//租户人员list不为空时遍历

if(tenantList!=null){

for(Tenant tenant:tenantList){

strName+=tenant.getName()+",";

}

household.setTenant(strName);

}

}

}

return list;

}

/\*\*

\* 删除函数(非物理删除)

\*/

protected void deleteObjectById(String householdId) {

super.getDefaultManager().execute("update Household a set a.status=0 where a.id = ? ", householdId);

}

/\*\*

\* 获得查询条件

\*/

protected String getConditions(int status,String searchAddress,String rentalId)

{

String conditionsStr = " and a.status="+status;

if(searchAddress!=null && searchAddress.compareTo("")!=0){

conditionsStr += " and a.rental.address like '%"+ searchAddress.trim() +"%'";

}

if(rentalId!=null && rentalId.compareTo("")!=0){

conditionsStr += " and a.rental.id = '"+ rentalId +"'";

}

return conditionsStr;

}

protected Class getImplementClass()

{

return Household.class;

}

public void setSearchAddress(String searchAddress) {

this.searchAddress = searchAddress;

}

public String getRentalId() {

return rentalId;

}

public void setRentalId(String rentalId) {

this.rentalId = rentalId;

}

}

package com.bizlogic.cz.action;

import java.util.List;

import com.bizlogic.base.front.action.BaseUserEditAction;

import com.bizlogic.cz.bean.Household;

import com.bizlogic.cz.bean.Tenant;

public class HouseholdShowAction extends BaseUserEditAction {

//租户人员

private List<Tenant> tenantList;

protected void prepareShow() {

Household household = (Household) element;

tenantList = super.getDefaultManager().findAll("from Tenant t where t.household=? and t.status=?",household,1);

}

protected Class getElementClass() {

return Household.class;

}

public List<Tenant> getTenantList() {

return tenantList;

}

}

package com.bizlogic.cz.action;

import java.io.ByteArrayInputStream;

import java.io.ByteArrayOutputStream;

import java.io.IOException;

import java.io.InputStream;

import java.io.UnsupportedEncodingException;

import java.util.List;

import org.apache.poi.hssf.usermodel.HSSFCell;

import org.apache.poi.hssf.usermodel.HSSFCellStyle;

import org.apache.poi.hssf.usermodel.HSSFRow;

import org.apache.poi.hssf.usermodel.HSSFSheet;

import org.apache.poi.hssf.usermodel.HSSFWorkbook;

import org.apache.poi.ss.usermodel.Font;

import org.apache.poi.ss.util.CellRangeAddress;

import com.bizlogic.base.Degree;

import com.bizlogic.common.action.BaseAction;

import com.bizlogic.cz.bean.Tenant;

import com.bizlogic.cz.bean.WorkType;

public class JobExportAction extends BaseAction{

public String execute() throws Exception

{

return super.execute();

}

public String getFileName() {

try {

return new String("亚运村流动人口就业情况.xls".getBytes("gbk"), "ISO8859\_1");

} catch (UnsupportedEncodingException e) {

return "亚运村流动人口就业情况.xls";

}

}

public InputStream getExcelFile() {

//创建excel工作簿

HSSFWorkbook workbook = new HSSFWorkbook();

String str = "";

str = "亚运村流动人口就业情况";

//创建一页

HSSFSheet sheet = workbook.createSheet(str);

//设置列宽

sheet.setColumnWidth(0, 2000);

sheet.setColumnWidth(1, 7000);

sheet.setColumnWidth(2, 5000);

HSSFCellStyle style0 = workbook.createCellStyle();

//议题样式

HSSFCellStyle style1 = workbook.createCellStyle();

//时间和地点样式

HSSFCellStyle style2 = workbook.createCellStyle();

//字段标题

HSSFCellStyle style3 = workbook.createCellStyle();

//强制换行

//标题样式

HSSFCellStyle style4 = workbook.createCellStyle();

//style0设置边框

style0.setBorderTop(HSSFCellStyle.BORDER\_THIN);

style0.setBorderLeft(HSSFCellStyle.BORDER\_THIN);

style0.setBorderRight(HSSFCellStyle.BORDER\_THIN);

style0.setBorderBottom(HSSFCellStyle.BORDER\_THIN);

//style1设置边框

style1.setBorderTop(HSSFCellStyle.BORDER\_THIN);

style1.setBorderLeft(HSSFCellStyle.BORDER\_THIN);

style1.setBorderRight(HSSFCellStyle.BORDER\_THIN);

style1.setBorderBottom(HSSFCellStyle.BORDER\_THIN);

//style3设置边框

style3.setBorderTop(HSSFCellStyle.BORDER\_THIN);

style3.setBorderLeft(HSSFCellStyle.BORDER\_THIN);

style3.setBorderRight(HSSFCellStyle.BORDER\_THIN);

style3.setBorderBottom(HSSFCellStyle.BORDER\_THIN);

//边框颜色

style0.setRightBorderColor((short) 00000);

style1.setRightBorderColor((short) 00000);

style3.setRightBorderColor((short) 00000);

//内容在单元格中间显示

style0.setVerticalAlignment(HSSFCellStyle.VERTICAL\_CENTER);

style1.setVerticalAlignment(HSSFCellStyle.ALIGN\_LEFT);

style3.setVerticalAlignment(HSSFCellStyle.VERTICAL\_CENTER);

style4.setVerticalAlignment(HSSFCellStyle.VERTICAL\_CENTER);

//内容居中显示

style0.setAlignment(HSSFCellStyle.ALIGN\_CENTER);

style1.setAlignment(HSSFCellStyle.ALIGN\_LEFT);

style3.setAlignment(HSSFCellStyle.ALIGN\_CENTER);

style4.setAlignment(HSSFCellStyle.ALIGN\_CENTER);

Font font = workbook.createFont();

font.setBoldweight((short) 10);//设置粗体

font.setColor((short) 00000);//字体颜色

font.setFontHeightInPoints((short) 12); //字体大小

font.setFontName("宋体");//设置字体

Font font2 = workbook.createFont();

font2.setBoldweight((short) 10);//设置粗体

font2.setColor((short) 00000);//字体颜色

font2.setFontHeightInPoints((short) 16);//字体大小

font2.setFontName("黑体");//设置字体

Font font3 = workbook.createFont();

font3.setBoldweight((short) 10);//设置粗体

font3.setColor((short) 00000);//字体颜色

font3.setFontHeightInPoints((short) 12);//字体大小

font3.setFontName("黑体");

style0.setFont(font);

style1.setFont(font);

style2.setFont(font);

style3.setFont(font3);

style4.setFont(font2);

List<WorkType> list = super.getDefaultManager().findAll("from WorkType t order by t.id");

//设置合并单元格（0行到0行，0列到2列进行合并）

sheet.addMergedRegion(new CellRangeAddress(0,0,0,2));

//第0行

HSSFRow row0 = sheet.createRow(0);

//0行0列值

HSSFCell cell0 = row0.createCell(0);

cell0.setCellValue("亚运村流动人口就业情况");

cell0.setCellStyle(style4);

//第二行

HSSFRow row2 = sheet.createRow(1);

HSSFCell cell2 = row2.createCell(0);

cell2.setCellValue("序号");

cell2.setCellStyle(style3);

cell2 = row2.createCell(1);

cell2.setCellValue("就业类别");

cell2.setCellStyle(style3);

cell2 = row2.createCell(2);

cell2.setCellValue("流动人口(人数)");

cell2.setCellStyle(style3);

int flag = 0;

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

WorkType workType= list.get(i);

//流动人口数量

List countList = super.getDefaultManager().findSQL("select t.tenant\_id from T\_CZ\_TENANT\_WORK\_INFO t where t.status=1 and t.work\_type\_id=?",workType.getId());

//获取行

HSSFRow row = sheet.getRow(i+2+flag);

//如果没有是空，进行创建

if(row == null )

row = sheet.createRow(i+2+flag);

//编号

HSSFCell cell = row.createCell(0);

cell.setCellStyle(style0);

cell.setCellValue(i+1+flag);

//社区

cell = row.createCell(1);

cell.setCellStyle(style0);

cell.setCellValue(workType.getName());

//流动人口数量

cell = row.createCell(2);

cell.setCellStyle(style0);

cell.setCellValue(countList.size());

}

ByteArrayOutputStream baos = new ByteArrayOutputStream();

try {

workbook.write(baos);

} catch (IOException e) {

e.printStackTrace();

}

byte[] ba = baos.toByteArray();

ByteArrayInputStream bais = new ByteArrayInputStream(ba);

return bais;

}

}

package com.bizlogic.cz.action;

import java.util.List;

import org.apache.commons.lang.StringUtils;

import com.bizlogic.base.Company;

import com.bizlogic.base.front.action.BaseUserEditAction;

import com.bizlogic.common.Element;

import com.bizlogic.cz.bean.PoliceCommunity;

public class PoliceCommunityEditAction extends BaseUserEditAction{

//社区编号

private String companyId;

//社区列表

private List<Company> companyList;

public List<Company> getCompanyList() {

return super.getDefaultManager().findAll(

"from Company t");

}

protected Element createElement() {

PoliceCommunity policeCommunity = new PoliceCommunity();

policeCommunity.setStatus(1);

return policeCommunity;

}

protected void prepareShow() {

PoliceCommunity policeCommunity = (PoliceCommunity) element;

if(policeCommunity.getId() != null && policeCommunity.getCompany() != null){

companyId = policeCommunity.getCompany().getId();

}

}

protected boolean prepareSave(Element element) {

PoliceCommunity policeCommunity = (PoliceCommunity) element;

if (StringUtils.isNotBlank(companyId)) {

policeCommunity.setCompany((Company) super.getDefaultManager().get(Company.class,companyId));

} else {

policeCommunity.setCompany(null);

}

return super.prepareSave(element);

}

protected Class getElementClass() {

return PoliceCommunity.class;

}

public String getCompanyId() {

return companyId;

}

public void setCompanyId(String companyId) {

this.companyId = companyId;

}

}

package com.bizlogic.cz.action;

import java.util.List;

import com.bizlogic.base.front.action.BaseUserListAction;

import com.bizlogic.common.action.helper.ListInfo;

import com.bizlogic.cz.bean.PoliceCommunity;

public class PoliceCommunityListAction extends BaseUserListAction{

@Override

protected long doGetCount(ListInfo listInfo) {

return 0;

}

@Override

protected List doSearch(ListInfo listInfo) {

return super.getDefaultManager().findAll("from PoliceCommunity a where a.status=1 order by a.id");

}

/\*\*

\* 删除函数(非物理删除)

\*/

protected void deleteObjectById(String policeCommunityId) {

super.getDefaultManager().execute("update PoliceCommunity a set a.status=0 where a.id=?", policeCommunityId);

}

protected Class getImplementClass()

{

return PoliceCommunity.class;

}

}

package com.bizlogic.cz.action;

import java.io.File;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.List;

import com.bizlogic.base.AttachmentValue;

import com.bizlogic.base.User;

import com.bizlogic.base.front.action.BaseUserEditAction;

import com.bizlogic.base.helper.AttachmentHelper;

import com.bizlogic.common.Element;

import com.bizlogic.cz.bean.Rental;

import com.bizlogic.cz.bean.RentalCheck;

import com.bizlogic.oa.bean.Task;

public class RentalCheckEditAction extends BaseUserEditAction{

//附件上传----start

private AttachmentHelper attachmentHelper;

private File[] fileAttachment;

private String[] fileAttachmentContentType;

private String[] fileAttachmentFileName;

private long serial = 0;

//附件列表

private List<AttachmentValue> attachmentList;

//出租房屋编号

private String rentalId;

//入户日期

private String checkDate;

protected void prepareShow() {

RentalCheck rentalCheck = (RentalCheck) element;

//将rentalId设置session

rentalId = super.getParameterValue("rentalId");

if(rentalId != null){

super.getSession().setAttribute("rentalId",rentalId);

}else{

rentalId = (String)super.getSession().getAttribute("rentalId");

}

if(rentalCheck.getId() == null){

checkDate = DateToString(new Date());

}else{

checkDate = DateToString(rentalCheck.getCheckDate());

}

if(rentalCheck.getId()!=null){

attachmentList = super.getDefaultManager().findAll("from AttachmentValue a where a.objectId=? order by a.createTime", rentalCheck.getId());

}

}

protected boolean prepareSave(Element element) {

RentalCheck rentalCheck = (RentalCheck) element;

rentalCheck.setRental(super.getDefaultManager().get(Rental.class, rentalId));

rentalCheck.setCheckDate(new java.sql.Date(StingToDate(checkDate).getTime()));

rentalCheck.setCreateUser((User)getVisit());

rentalCheck.setCreateDate(new java.sql.Date((new Date()).getTime()));

rentalCheck.setStatus(1);

return super.prepareSave(element);

}

public String save()

{

RentalCheck rentalCheck = (RentalCheck)element;

super.save();

this.setId(element.getId());

// 添加附件

if (fileAttachment != null)

this.addAttachment();

return SUCCESS;

}

private static String DateToString(Date param) {

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

return sdf.format(param);

}

private static Date StingToDate(String param) {

Date date = null;

SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd");

try {

date = sdf.parse(param);

} catch (ParseException ex) {

}

return date;

}

//添加附件

private String addAttachment()

{

//传入DAO

attachmentHelper.init(super.getDefaultManager());

for(int i=0;i<fileAttachment.length;i++)

{

AttachmentValue value = new AttachmentValue();

value.setContentType(fileAttachmentContentType[i]);

value.setFileName(fileAttachmentFileName[i]);

value.setType("RENTAL\_CHECK\_ATTACHMENT");

value.setName(fileAttachmentFileName[i].substring(0,fileAttachmentFileName[i].lastIndexOf(".")));

value.setSize(fileAttachment[i].length());

value.setUploadFile(fileAttachment[i]);

value.setObjectId(id);

value.setRealName(System.currentTimeMillis() + (serial++) + "." + attachmentHelper.getFileExtName(value.getFileName()) );

value.setCreateTime(new java.util.Date());

value.setCreateUser((User)getVisit());

attachmentHelper.saveAttachment(value);

}

return SUCCESS;

}

protected Class getElementClass() {

return RentalCheck.class;

}

public String getCheckDate() {

return checkDate;

}

public void setCheckDate(String checkDate) {

this.checkDate = checkDate;

}

public String getRentalId() {

return rentalId;

}

public void setRentalId(String rentalId) {

this.rentalId = rentalId;

}

public AttachmentHelper getAttachmentHelper() {

return attachmentHelper;

}

public void setAttachmentHelper(AttachmentHelper attachmentHelper) {

this.attachmentHelper = attachmentHelper;

}

public File[] getFileAttachment() {

return fileAttachment;

}

public void setFileAttachment(File[] fileAttachment) {

this.fileAttachment = fileAttachment;

}

public String[] getFileAttachmentContentType() {

return fileAttachmentContentType;

}

public void setFileAttachmentContentType(String[] fileAttachmentContentType) {

this.fileAttachmentContentType = fileAttachmentContentType;

}

public String[] getFileAttachmentFileName() {

return fileAttachmentFileName;

}

public void setFileAttachmentFileName(String[] fileAttachmentFileName) {

this.fileAttachmentFileName = fileAttachmentFileName;

}

public long getSerial() {

return serial;

}

public void setSerial(long serial) {

this.serial = serial;

}

public List<AttachmentValue> getAttachmentList() {

return attachmentList;

}

public void setAttachmentList(List<AttachmentValue> attachmentList) {

this.attachmentList = attachmentList;

}

}

package com.bizlogic.cz.action;

import java.util.List;

import com.bizlogic.base.AttachmentValue;

import com.bizlogic.base.front.action.BaseUserListAction;

import com.bizlogic.common.action.helper.ListInfo;

import com.bizlogic.cz.bean.RentalCheck;

public class RentalCheckListAction extends BaseUserListAction{

//房屋id

private String rentalId;

@Override

protected long doGetCount(ListInfo listInfo) {

//将rentalId设置session

rentalId = super.getParameterValue("rentalId");

if(rentalId != null){

super.getSession().setAttribute("rentalId",rentalId);

}else{

rentalId = (String)super.getSession().getAttribute("rentalId");

}

sqlCondition = this.getConditions(1,rentalId);

return super.getDefaultManager().getCount("from RentalCheck a where 1=1 "+sqlCondition);

}

@Override

protected List doSearch(ListInfo listInfo) {

List<RentalCheck> list = super.getDefaultManager().find("from RentalCheck a where 1=1 "+sqlCondition+" order by a.checkDate asc",

listInfo.getFirstNo(), listInfo.getMaxNo());

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

//图片附件列表

List<AttachmentValue> photoAttachmentList = super.getDefaultManager().findAll("from AttachmentValue a " +

" where " +

" a.objectId=? " +

" and (a.contentType='image/jpeg' " +

" or a.contentType='image/bmp' " +

" or a.contentType='image/png' " +

" or a.contentType='image/jpeg' " +

" or a.contentType='image/gif') order by a.createTime,a.id", list.get(i).getId());

//文件附件列表

List<AttachmentValue> fileAttachmentList = super.getDefaultManager().findAll("from AttachmentValue a " +

" where " +

" a.objectId=? " +

" and a.contentType!='image/jpeg' " +

" and a.contentType!='image/bmp' " +

" and a.contentType!='image/png' " +

" and a.contentType!='image/jpeg' " +

" and a.contentType!='image/gif' order by a.createTime,a.id", list.get(i).getId());

list.get(i).setPhotoAttachmentList(photoAttachmentList);

list.get(i).setFileAttachmentList(fileAttachmentList);

}

return list;

}

/\*\*

\* 获得查询条件

\*/

protected String getConditions(int status,String rId)

{

String conditionsStr = " and a.status="+status;

if(rId!=null && rId.compareTo("")!=0){

conditionsStr += " and a.rental.id = '"+ rId +"'";

}

return conditionsStr;

}

/\*\*

\* 删除函数(非物理删除)

\*/

@Override

protected void deleteObjectById(String rentalCheckId) {

super.getDefaultManager().execute("update RentalCheck t set t.status=0 where t.id=?", rentalCheckId);

}

protected Class getImplementClass()

{

return RentalCheck.class;

}

public String getRentalId() {

return rentalId;

}

public void setRentalId(String rentalId) {

this.rentalId = rentalId;

}

}

package com.bizlogic.cz.action;

import java.util.List;

import com.bizlogic.base.AttachmentValue;

import com.bizlogic.base.front.action.BaseUserListAction;

import com.bizlogic.common.action.helper.ListInfo;

import com.bizlogic.cz.bean.RentalCheck;

public class RentalCheckShowListAction extends BaseUserListAction{

//房屋id

private String rentalId;

@Override

protected long doGetCount(ListInfo listInfo) {

//将rentalId设置session

rentalId = super.getParameterValue("rentalId");

if(rentalId != null){

super.getSession().setAttribute("rentalId",rentalId);

}else{

rentalId = (String)super.getSession().getAttribute("rentalId");

}

sqlCondition = this.getConditions(1,rentalId);

return super.getDefaultManager().getCount("from RentalCheck a where 1=1 "+sqlCondition);

}

@Override

protected List doSearch(ListInfo listInfo) {

List<RentalCheck> list = super.getDefaultManager().find("from RentalCheck a where 1=1 "+sqlCondition+" order by a.checkDate desc",

listInfo.getFirstNo(), listInfo.getMaxNo());

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

//图片附件列表

List<AttachmentValue> photoAttachmentList = super.getDefaultManager().findAll("from AttachmentValue a " +

" where " +

" a.objectId=? " +

" and (a.contentType='image/jpeg' " +

" or a.contentType='image/bmp' " +

" or a.contentType='image/png' " +

" or a.contentType='image/jpeg' " +

" or a.contentType='image/gif') order by a.createTime,a.id", list.get(i).getId());

//文件附件列表

List<AttachmentValue> fileAttachmentList = super.getDefaultManager().findAll("from AttachmentValue a " +

" where " +

" a.objectId=? " +

" and a.contentType!='image/jpeg' " +

" and a.contentType!='image/bmp' " +

" and a.contentType!='image/png' " +

" and a.contentType!='image/jpeg' " +

" and a.contentType!='image/gif' order by a.createTime,a.id", list.get(i).getId());

list.get(i).setPhotoAttachmentList(photoAttachmentList);

list.get(i).setFileAttachmentList(fileAttachmentList);

}

return list;

}

/\*\*

\* 获得查询条件

\*/

protected String getConditions(int status,String rId)

{

String conditionsStr = " and a.status="+status;

if(rId!=null && rId.compareTo("")!=0){

conditionsStr += " and a.rental.id = '"+ rId +"'";

}

return conditionsStr;

}

/\*\*

\* 删除函数(非物理删除)

\*/

@Override

protected void deleteObjectById(String rentalCheckId) {

super.getDefaultManager().execute("update RentalCheck t set t.status=0 where t.id=?", rentalCheckId);

}

protected Class getImplementClass()

{

return RentalCheck.class;

}

public String getRentalId() {

return rentalId;

}

public void setRentalId(String rentalId) {

this.rentalId = rentalId;

}

}

package com.bizlogic.cz.action;

import java.util.Date;

import java.util.List;

import org.apache.commons.lang.StringUtils;

import com.bizlogic.base.Company;

import com.bizlogic.base.User;

import com.bizlogic.base.front.action.BaseUserEditAction;

import com.bizlogic.common.Element;

import com.bizlogic.cz.bean.BuildingUse;

import com.bizlogic.cz.bean.PoliceCommunity;

import com.bizlogic.cz.bean.Rental;

import com.bizlogic.qms.bean.BuildingType;

import com.bizlogic.qms.bean.HolderType;

import com.bizlogic.qms.bean.Room;

public class RentalEditAction extends BaseUserEditAction {

//建设性质id

private String buildingUseId;

//社区id

private String companyId;

//建筑类型id

private String buildingTypeId;

//所有权类型id

private String holderTypeId;

// 建设性质

private List buildingUseList;

public List getbuildingUseList() {

return super.getDefaultManager().findAll("from BuildingUse a order by a.id");

}

// 所有权类型

private List holderTypeList;

public List getholderTypeList() {

return super.getDefaultManager().findAll("from HolderType a order by a.id");

}

// 社区

private List companyList;

public List getcompanyList() {

return super.getDefaultManager().findAll("from Company a where a.parent!=null order by a.id");

}

// 建筑类型

private List buildingTypeList;

public List getbuildingTypeList() {

return super.getDefaultManager().findAll("from BuildingType a order by a.id");

}

//楼户编号

private String roomId;

//对应派出所编号

private String policeCommunityId;

protected Element createElement() {

Rental rental = new Rental();

return rental;

}

protected void prepareShow() {

Rental rental = (Rental) element;

//编辑时

if(rental.getId()!=null){

companyId=rental.getCompany().getId();

buildingTypeId=rental.getBuildingType().getId();

holderTypeId=rental.getHolderType().getId();

buildingUseId=rental.getBuildingUse().getId();

}

}

protected boolean prepareSave(Element element) {

Rental rental = (Rental) element;

if (StringUtils.isNotBlank(companyId)) {

rental.setCompany((Company) super.getDefaultManager().get(Company.class,companyId));

} else {

rental.setCompany(null);

}

if (StringUtils.isNotBlank(buildingTypeId)) {

rental.setBuildingType((BuildingType) super.getDefaultManager().get(BuildingType.class,buildingTypeId));

} else {

rental.setBuildingType(null);

}

if (StringUtils.isNotBlank(holderTypeId)) {

rental.setHolderType((HolderType) super.getDefaultManager().get(HolderType.class,holderTypeId));

} else {

rental.setHolderType(null);

}

if (StringUtils.isNotBlank(buildingUseId)) {

rental.setBuildingUse((BuildingUse) super.getDefaultManager().get(BuildingUse.class,buildingUseId));

} else {

rental.setBuildingUse(null);

}

rental.setPoliceCommunity(super.getDefaultManager().get(PoliceCommunity.class, policeCommunityId));

rental.setCreateDate(new Date());

rental.setUser((User)getVisit());

rental.setStatus(1);

//设置楼户编号

rental.setRoom(super.getDefaultManager().get(Room.class, roomId));

return super.prepareSave(rental);

}

protected Class getElementClass() {

return Rental.class;

}

public String getCompanyId() {

return companyId;

}

public void setCompanyId(String companyId) {

this.companyId = companyId;

}

public String getBuildingTypeId() {

return buildingTypeId;

}

public void setBuildingTypeId(String buildingTypeId) {

this.buildingTypeId = buildingTypeId;

}

public String getHolderTypeId() {

return holderTypeId;

}

public void setHolderTypeId(String holderTypeId) {

this.holderTypeId = holderTypeId;

}

public String getBuildingUseId() {

return buildingUseId;

}

public void setBuildingUseId(String buildingUseId) {

this.buildingUseId = buildingUseId;

}

public String getRoomId() {

return roomId;

}

public void setRoomId(String roomId) {

this.roomId = roomId;

}

public String getPoliceCommunityId() {

return policeCommunityId;

}

public void setPoliceCommunityId(String policeCommunityId) {

this.policeCommunityId = policeCommunityId;

}

}

package com.bizlogic.cz.action;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.Date;

import java.util.List;

import org.apache.commons.lang.StringUtils;

import com.bizlogic.base.CertificateType;

import com.bizlogic.base.User;

import com.bizlogic.base.front.action.BaseUserEditAction;

import com.bizlogic.common.Element;

import com.bizlogic.cz.bean.LiabilityLetter;

import com.bizlogic.cz.bean.LiabilityLetterType;

import com.bizlogic.cz.bean.RentCategory;

import com.bizlogic.cz.bean.RentType;

import com.bizlogic.cz.bean.RentUse;

import com.bizlogic.cz.bean.Rental;

import com.bizlogic.cz.bean.RentalInfo;

public class RentalInfoEditAction extends BaseUserEditAction {

//责任书类型

private String liabilityLetterTypeStr;

//出租房id

private String rentalId;

//出租方式id

private String rentTypeId;

//出租用途id

private String rentUseId;

//出租类别id

private String rentCategoryId;

//责任书类型id

private String[] liabilityLetterTypeIds;

//出租起始日期

private String rentStart;

//出租结束日期

private String rentEnd;

//证件类型id

private String certificateTypeId;

// 出租方式

private List rentTypeList;

public List getrentTypeList() {

return super.getDefaultManager().findAll("from RentType a order by a.id");

}

// 出租用途

private List rentUseList;

public List getrentUseList() {

return super.getDefaultManager().findAll("from RentUse a order by a.id");

}

// 出租类别

private List rentCategoryList;

public List getrentCategoryList() {

return super.getDefaultManager().findAll("from RentCategory a order by a.id");

}

// 证件类型

private List certificateTypeList;

public List getcertificateTypeList() {

return super.getDefaultManager().findAll("from CertificateType a order by a.id");

}

protected void prepareShow() {

liabilityLetterTypeStr="";

RentalInfo rentalInfo = (RentalInfo) element;

//将rentalId设置session

rentalId = super.getParameterValue("rentalId");

if(rentalId != null){

super.getSession().setAttribute("rentalId",rentalId);

}else{

rentalId = (String)super.getSession().getAttribute("rentalId");

}

//新增时

if(rentalInfo.getId()==null){

rentStart=new SimpleDateFormat("yyyy-MM-dd").format(new Date());

rentEnd=new SimpleDateFormat("yyyy-MM-dd").format(new Date());

List<LiabilityLetterType> liabilityLetterTypeList=super.getDefaultManager().findAll("from LiabilityLetterType a order by a.id");

for(LiabilityLetterType liabilityLetterType:liabilityLetterTypeList){

liabilityLetterTypeStr +="<input type='checkbox' value='"+liabilityLetterType.getId()+"' name='liabilityLetterTypeIds'>"+liabilityLetterType.getName()+" ";

}

//编辑时

}else{

rentTypeId=rentalInfo.getRentType().getId();

rentUseId=rentalInfo.getRentUse().getId();

rentCategoryId=rentalInfo.getRentCategory().getId();

rentStart=new SimpleDateFormat("yyyy-MM-dd").format(rentalInfo.getRentStart());

if(rentalInfo.getRentEnd()!=null){

rentEnd=new SimpleDateFormat("yyyy-MM-dd").format(rentalInfo.getRentEnd());

}else{

rentEnd="";

}

if(rentalInfo.getCertificateType()!=null){

certificateTypeId=rentalInfo.getCertificateType().getId();

}else{

certificateTypeId="";

}

List<LiabilityLetterType> liabilityLetterTypeList=super.getDefaultManager().findAll("from LiabilityLetterType a order by a.id");

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

LiabilityLetterType liabilityLetterType = liabilityLetterTypeList.get(i);

List list = super.getDefaultManager().findAll("from LiabilityLetter t where t.liabilityLetterType=? and t.rentalInfo=?", liabilityLetterTypeList.get(i), rentalInfo);

if(list.size()>0){

liabilityLetterTypeStr +="<input type='checkbox' value='"+liabilityLetterType.getId()+"' name='liabilityLetterTypeIds' checked='true'>"+liabilityLetterType.getName()+" ";

}else{

liabilityLetterTypeStr +="<input type='checkbox' value='"+liabilityLetterType.getId()+"' name='liabilityLetterTypeIds'>"+liabilityLetterType.getName()+" ";

}

}

}

}

protected boolean prepareSave(Element element) {

RentalInfo rentalInfo = (RentalInfo) element;

if (StringUtils.isNotBlank(rentTypeId)) {

String[] rentTypeIdS=rentTypeId.split(",");

rentalInfo.setRentType((RentType) super.getDefaultManager().get(RentType.class,rentTypeIdS[1]));

} else {

rentalInfo.setRentType(null);

}

if (StringUtils.isNotBlank(rentUseId)) {

rentalInfo.setRentUse((RentUse) super.getDefaultManager().get(RentUse.class,rentUseId));

} else {

rentalInfo.setRentUse(null);

}

if (StringUtils.isNotBlank(rentCategoryId)) {

rentalInfo.setRentCategory((RentCategory) super.getDefaultManager().get(RentCategory.class,rentCategoryId));

} else {

rentalInfo.setRentCategory(null);

}

if (StringUtils.isNotBlank(certificateTypeId)) {

rentalInfo.setCertificateType((CertificateType) super.getDefaultManager().get(CertificateType.class,certificateTypeId));

} else {

rentalInfo.setCertificateType(null);

}

if (StringUtils.isNotBlank(rentStart)) {

try {

rentalInfo.setRentStart(new SimpleDateFormat("yyyy-MM-dd").parse(rentStart));

} catch (ParseException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

} else {

rentalInfo.setRentStart(null);

}

if (StringUtils.isNotBlank(rentEnd)) {

try {

rentalInfo.setRentEnd(new SimpleDateFormat("yyyy-MM-dd").parse(rentEnd));

} catch (ParseException e) {

// TODO Auto-generated catch block

e.printStackTrace();

}

} else {

rentalInfo.setRentEnd(null);

}

rentalInfo.setCreateDate(new Date());

rentalInfo.setUser((User)getVisit());

rentalInfo.setRental((Rental) super.getDefaultManager().get(Rental.class,rentalId));

rentalInfo.setStatus(1);

return super.prepareSave(rentalInfo);

}

public String save()

{

RentalInfo rentalInfo = (RentalInfo) element;

super.save();

//将之前出租信息改为历史信息

super.getDefaultManager().execute("update RentalInfo t set t.status=2 where t.rental.id=? and t.id != ?", rentalId, rentalInfo.getId());

if(liabilityLetterTypeIds!=null){

if(liabilityLetterTypeIds.length>0){

for(String strId:liabilityLetterTypeIds){

LiabilityLetter liabilityLetter=new LiabilityLetter();

liabilityLetter.setRentalInfo(rentalInfo);

liabilityLetter.setLiabilityLetterType((LiabilityLetterType) super.getDefaultManager().get(LiabilityLetterType.class,strId));

liabilityLetter.setStatus(1);

super.getDefaultManager().save(liabilityLetter);

}

}

}

return SUCCESS;

}

protected Class getElementClass() {

return RentalInfo.class;

}

public String getRentTypeId() {

return rentTypeId;

}

public void setRentTypeId(String rentTypeId) {

this.rentTypeId = rentTypeId;

}

public String getRentUseId() {

return rentUseId;

}

public void setRentUseId(String rentUseId) {

this.rentUseId = rentUseId;

}

public String getRentCategoryId() {

return rentCategoryId;

}

public void setRentCategoryId(String rentCategoryId) {

this.rentCategoryId = rentCategoryId;

}

public String getCertificateTypeId() {

return certificateTypeId;

}

public void setCertificateTypeId(String certificateTypeId) {

this.certificateTypeId = certificateTypeId;

}

public String getRentStart() {

return rentStart;

}

public void setRentStart(String rentStart) {

this.rentStart = rentStart;

}

public String getRentEnd() {

return rentEnd;

}

public void setRentEnd(String rentEnd) {

this.rentEnd = rentEnd;

}

public String getRentalId() {

return rentalId;

}

public void setRentalId(String rentalId) {

this.rentalId = rentalId;

}

public String getLiabilityLetterTypeStr() {

return liabilityLetterTypeStr;

}

public void setLiabilityLetterTypeStr(String liabilityLetterTypeStr) {

this.liabilityLetterTypeStr = liabilityLetterTypeStr;

}

public String[] getLiabilityLetterTypeIds() {

return liabilityLetterTypeIds;

}

public void setLiabilityLetterTypeIds(String[] liabilityLetterTypeIds) {

this.liabilityLetterTypeIds = liabilityLetterTypeIds;

}

}

package com.bizlogic.cz.action;

import java.util.List;

import com.bizlogic.base.front.action.BaseUserListAction;

import com.bizlogic.common.action.helper.ListInfo;

import com.bizlogic.cz.bean.RentalInfo;

public class RentalInfoListAction extends BaseUserListAction{

//出租房id

private String rentalId;

@Override

protected long doGetCount(ListInfo listInfo) {

//当出租房屋id不为空时放入session

if(rentalId != null){

super.getSession().setAttribute("rentalId",rentalId);

}else{

//当出租房屋id为空时从session中拿出来

rentalId = (String)super.getSession().getAttribute("rentalId");

}

sqlCondition = this.getConditions(0,rentalId,searchName);

return super.getDefaultManager().getCount("from RentalInfo a where 1=1 "+sqlCondition);

}

@Override

protected List doSearch(ListInfo listInfo) {

List list = super.getDefaultManager().find("from RentalInfo a where 1=1 "+sqlCondition+" order by a.createDate desc",

listInfo.getFirstNo(), listInfo.getMaxNo());

return list;

}

/\*\*

\* 删除函数(非物理删除)

\*/

@Override

protected void deleteObjectById(String rentInfoId) {

super.getDefaultManager().execute("update RentalInfo t set t.status=0 where t.id=?", rentInfoId);

}

/\*\*

\* 获得查询条件

\*/

protected String getConditions(int status,String rentalId,String searchName)

{

String conditionsStr = " and a.status!="+status;

if(searchName!=null && searchName.compareTo("")!=0){

conditionsStr += " and a.user.name like '%"+ searchName.trim()+"%'";

}

if(rentalId!=null && rentalId.compareTo("")!=0){

conditionsStr += " and a.rental.id= '"+ rentalId+"'";

}

return conditionsStr;

}

protected Class getImplementClass()

{

return RentalInfo.class;

}

public String getRentalId() {

return rentalId;

}

public void setRentalId(String rentalId) {

this.rentalId = rentalId;

}

}

package com.bizlogic.cz.action;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.List;

import net.sf.json.JSONObject;

import org.apache.commons.lang.StringUtils;

import com.bizlogic.base.front.action.BaseUserListAction;

import com.bizlogic.common.action.helper.ListInfo;

import com.bizlogic.cz.bean.Homeowner;

import com.bizlogic.cz.bean.PoliceCommunity;

import com.bizlogic.cz.bean.Rental;

import com.bizlogic.cz.bean.RentalInfo;

import com.bizlogic.cz.bean.RentalSafe;

import com.bizlogic.cz.bean.WaterHeaterType;

import com.bizlogic.qms.bean.Building;

import com.bizlogic.qms.bean.Floor;

import com.bizlogic.qms.bean.Room;

public class RentalListAction extends BaseUserListAction{

//热水器类型列表

private List<WaterHeaterType> waterHeaterTypeList;

public List<WaterHeaterType> getWaterHeaterTypeList() {

return super.getDefaultManager().findAll("from WaterHeaterType");

}

//地址

private String searchAddress;

//居住类型

private List liveTypeList;

public List getLiveTypeList() {

return super.getDefaultManager().findAll("from LiveType a where a.status=1 order by a.id ");

}

private String searchLiveTypeId;

//热水器类型

private String searchBathroomFacilities;

//房屋结构 :false为未改动，true为私设隔断

private String searchHouseStructure;

@Override

protected long doGetCount(ListInfo listInfo) {

long l = 0;

//根据安全信息进行查询

if((searchLiveTypeId != null && !"".equals(searchLiveTypeId))

|| (searchBathroomFacilities != null && !"".equals(searchBathroomFacilities))

|| "true".equals(searchHouseStructure)

)

{

sqlCondition = this.getConditionsBySafe(this.getVisit().getId(), searchLiveTypeId, searchBathroomFacilities, searchHouseStructure);

l = super.getDefaultManager().getCount("from RentalSafe a where 1=1 "+sqlCondition);

}else{

sqlCondition = this.getConditions(1,searchAddress,this.getVisit().getId());

l = super.getDefaultManager().getCount("from Rental a where 1=1 "+sqlCondition);

}

return l;

}

@Override

protected List doSearch(ListInfo listInfo) {

List list = new ArrayList();

if((searchLiveTypeId != null && !"".equals(searchLiveTypeId))

|| (searchBathroomFacilities != null && !"".equals(searchBathroomFacilities))

|| "true".equals(searchHouseStructure)){

list = super.getDefaultManager().find("select a.rental from RentalSafe a where 1=1 "+sqlCondition,

listInfo.getFirstNo(), listInfo.getMaxNo());

}else{

list = super.getDefaultManager().find("from Rental a where 1=1 "+sqlCondition,

listInfo.getFirstNo(), listInfo.getMaxNo());

}

//加载安全信息

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

Rental rental = (Rental)list.get(i);

//获得安全信息

String[] str = {"rental.id","status"};

Object[] pro = {rental.getId(),1};

RentalSafe rentalSafe = (RentalSafe)super.getDefaultManager().getByPropertys(RentalSafe.class, str,pro);

if(rentalSafe != null){

rental.setRentalSafe(rentalSafe);

}

}

return list;

}

/\*\*

\* 删除函数(非物理删除)

\*/

@Override

protected void deleteObjectById(String rentId) {

super.getDefaultManager().execute("update Rental t set t.status=0 where t.id=?", rentId);

}

/\*\*

\* 获得查询条件

\*/

protected String getConditions(int status, String address,String userId)

{

String conditionsStr = " and a.status="+status;

if(address!=null && address.compareTo("")!=0){

conditionsStr += " and a.address like '%"+ address.trim() +"%'";

}

if(userId!=null && userId.compareTo("")!=0){

conditionsStr += " and a.user.id = '"+ userId.trim() +"'";

}

return conditionsStr;

}

/\*\*

\* 获得查询条件

\*/

protected String getConditionsBySafe(String userId,String liveTypeId ,String bathroomFacilities ,String houseStructure )

{

String conditionsStr = "";

if(userId!=null && userId.compareTo("")!=0){

conditionsStr += " and a.rental.user.id = '"+ userId.trim() +"'";

}

if(liveTypeId!=null && liveTypeId.compareTo("")!=0){

conditionsStr += " and a.liveType.id = '"+ liveTypeId.trim() +"'";

}

if(bathroomFacilities!=null && bathroomFacilities.compareTo("")!=0){

conditionsStr += " and a.waterHeaterType.id = '"+ bathroomFacilities.trim() +"'";

}

if(houseStructure!=null && "true".equals(houseStructure.trim())){

conditionsStr += " and a.houseStructure = '1'";

}

return conditionsStr;

}

/\*\*

\* ajax请求

\* @throws IOException

\*/

public void ajaxMethod() throws IOException {

JSONObject res = new JSONObject();

//Response编码设置

super.getResponse().setCharacterEncoding("utf-8");

//获取房主编号

if(StringUtils.equals(act, "get\_home\_owner\_id")) {

String rentalId = super.getParameterValue("rentalId");

//获取房主信息

Homeowner homeowner = (Homeowner)super.getDefaultManager().getObject("from Homeowner t where t.status=1 and t.rental.id=?", rentalId);

//如果有房主则返回房主编号，否则返回空字符串

if(homeowner != null){

res.put("homeownerId", homeowner.getId());

}else{

res.put("homeownerId", "");

}

//返回房屋编号

res.put("rentalId", rentalId);

}

//获取出租房安全信息编号

if(StringUtils.equals(act, "get\_rental\_safe\_id")) {

String rentalId = super.getParameterValue("rentalId");

//获取安全信息

RentalSafe rentalSafe = (RentalSafe)super.getDefaultManager().getObject("from RentalSafe t where t.status=1 and t.rental.id=?", rentalId);

//如果有安全信息不为空则返回安全信息编号，否则返回空字符串

if(rentalSafe != null){

res.put("rentalSafeId", rentalSafe.getId());

}else{

res.put("rentalSafeId", "");

}

//返回出租房编号

res.put("rentalId", rentalId);

}

//获取出租房出租信息编号

if(StringUtils.equals(act, "get\_rentalInfo\_id")) {

String rentalId = super.getParameterValue("rentalId");

//获取出租信息

RentalInfo rentalInfo = (RentalInfo)super.getDefaultManager().getObject("from RentalInfo t where t.status=1 and t.rental.id=?", rentalId);

//如果有出租信息则返回出租信息编号，否则返回空字符串

if(rentalInfo != null){

res.put("rentalInfoId", rentalInfo.getId());

}else{

res.put("rentalInfoId", "");

}

//返回房屋编号

res.put("rentalId", rentalId);

}

//根据社区编号获取社区对应的楼

if(StringUtils.equals(act, "get\_building\_list")) {

//社区编号

String companyId = super.getParameterValue("companyId");

List<Building> buildingList = super.getDefaultManager().findAll("from Building t " +

" where " +

" t.company.id=? " +

" and t.bigClassUse.id=? " +

" order by to\_number(t.lineNumber)", companyId,"110201001");

StringBuffer str = new StringBuffer();

str.append("<div class='selectBuilding\_bg'>");

str.append("<div class='selectBuilding\_dh'>选择楼<input type='button' class='closed' value='关闭' onclick='javascript:closeDiv2();return false;'></div>");

str.append("<div style='width:100%;height:90%;overflow:auto;'>");

str.append("<table cellpadding='0' cellspacing='0' class='list\_tb'>");

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

Building building = (Building)buildingList.get(i);

if(i == 0 || i%6 == 0){

str.append("<tr>");

}

str.append("<td width='16%' style='text-align:center'><a href='javascript:selectRoomNum("+'"'+building.getId()+'"'+")'>"+building.getBuildingAddress()+"</a></td>");

if(i != 0 && (i+1)%6 == 0){

str.append("</tr>");

}

if(i==buildingList.size()-1 && (i+1)%6 > 0){

for(int x=0;x<6-(i+1)%6;x++){

str.append("<td width='16%' style='text-align:center'>&nbsp;</td>");

if(x==5-(i+1)%6){

str.append("</tr>");

}

}

}

}

str.append("</table>");

str.append("</div>");

str.append("</div>");

res.put("message", str.toString());

}

//根据楼编号获取该楼中所有单元

if(StringUtils.equals(act, "get\_unit\_list")) {

//楼编号

String buildingId = super.getParameterValue("buildingId");

//获取楼

Building building = super.getDefaultManager().get(Building.class, buildingId);

//获得单元数

int unitCount = building.getUnitQuantity();

StringBuffer str = new StringBuffer();

str.append("<div class='selectBuilding\_bg'>");

str.append("<div class='selectBuilding\_dh'>选择单元(<font color='blue'>"+building.getBuildingAddress()+"</font>)<input type='button' class='closed' value='关闭' onclick='javascript:closeDiv2();return false;'></div>");

str.append("<div style='width:100%;height:85%;overflow:auto;'>");

str.append("<table cellpadding='0' cellspacing='0' class='list\_tb'>");

for(int i=0;i<unitCount;i++){

//单元数

int unit = i+1;

if(i == 0 || i%6 == 0){

str.append("<tr>");

}

str.append("<td width='16%' style='text-align:center'><a href='javascript:selectRoom("+'"'+buildingId+'"'+","+'"'+unit+'"'+")'>"+unit+"单元</a></td>");

if(i != 0 && (i+1)%6 == 0){

str.append("</tr>");

}

if(i==unitCount && (i+1)%6 > 0){

for(int x=0;x<6-(i+1)%6;x++){

str.append("<td width='16%' style='text-align:center'>&nbsp;</td>");

if(x==5-(i+1)%6){

str.append("</tr>");

}

}

}

}

str.append("</table>");

str.append("</div>");

str.append("<input type='button' class='pop\_btn' value='返回楼列表' onclick='javascript:backToBuilding();return false;'>");

str.append("</div>");

res.put("message", str.toString());

}

//根据楼层号及单元号获取楼层和楼户

if(StringUtils.equals(act, "get\_room\_list")) {

//楼编号

String buildingId = super.getParameterValue("buildingId");

//单元号

String unit = super.getParameterValue("unit");

//获取楼

Building building = super.getDefaultManager().get(Building.class, buildingId);

//获取所有楼层

List<Floor> floorList = super.getDefaultManager().findAll("from Floor t where t.building.id=? order by to\_number(t.atFloorNum)", buildingId);

//记录最大空td数

int maxtds = 0;

//记录每行td数

int tds = 0;

//循环楼层，找出x单元x层楼户最多数

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

Floor floor = (Floor)floorList.get(i);

List<Room> roomList = super.getDefaultManager().findAll("from Room t where t.floor.id=? and t.atUnit=? order by t.roomNum,t.roomAddress", floor.getId(),new Integer(unit));

tds = roomList.size();

if(tds > maxtds){

maxtds = tds;

}

tds = 0;

}

StringBuffer str = new StringBuffer();

str.append("<div class='selectBuilding\_bg'>");

str.append("<div class='selectBuilding\_dh'>选择房间(<font color='blue'>"+building.getBuildingAddress()+" "+unit+"单元</font>)<input type='button' class='closed' value='关闭' onclick='javascript:closeDiv2();return false;'></div>");

str.append("<div style='width:100%;height:85%;overflow:auto;'>");

str.append("<table cellpadding='0' cellspacing='0' class='list\_tb'>");

//记录每行的td数，如果小于最大td数(maxtds)，则添加空td

int tds2 = 0;

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

Floor floor = (Floor)floorList.get(i);

//根据楼层编号和单元号查找x单元x层所有楼户

List<Room> roomList = super.getDefaultManager().findAll("from Room t where t.floor.id=? and t.atUnit=? order by t.roomNum,t.roomAddress", floor.getId(),new Integer(unit));

str.append("<tr>");

for(int z=0;z<roomList.size();z++){

tds2++;

Room room = (Room)roomList.get(z);

//建筑类型编号

String buildingTypeId = "";

if(room.getFloor().getBuilding().getBuildingType() != null){

buildingTypeId = room.getFloor().getBuilding().getBuildingType().getId();

}

//所有权类型

String holderTypeId = "";

if(room.getHolderType() != null){

holderTypeId = room.getHolderType().getId();

}

String address = room.getRoomAddress();

if(address.indexOf("地下")<0){

str.append("<td style='text-align:center'><a href='javascript:setAddress("+'"'+room.getId()+'"'+","+'"'+room.getRoomAddress()+'"'+","+'"'+buildingTypeId+'"'+","+'"'+holderTypeId+'"'+")'>"+room.getRoomNum()+"</a></td>");

}else{

str.append("<td style='text-align:center'><a href='javascript:setAddress("+'"'+room.getId()+'"'+","+'"'+room.getRoomAddress()+'"'+","+'"'+buildingTypeId+'"'+","+'"'+holderTypeId+'"'+")'>地下"+room.getRoomNum()+"</a></td>");

}

//末尾添加空td

if(z==roomList.size()-1 && tds2 < maxtds){

for(int x=0;x<maxtds-tds2;x++){

str.append("<td style='text-align:center'>&nbsp;</td>");

}

}

//当楼层循环完毕后，上个td数标记为0

if(z==roomList.size()-1){

tds2 = 0;

}

}

str.append("</tr>");

}

str.append("</table>");

str.append("</div>");

str.append("<input type='button' class='pop\_btn' value='返回楼列表' onclick='javascript:backToBuilding();return false;'>&nbsp;&nbsp;");

str.append("<input type='button' class='pop\_btn' value='返回单元列' onclick='javascript:backToRoomNum("+'"'+buildingId+'"'+");return false;'>");

str.append("</div>");

res.put("message", str.toString());

}

//根据社区，获得派出所和民警姓名

if(StringUtils.equals(act, "get\_policeStationName\_and\_policeName")) {

String companyId = super.getParameterValue("companyId");

PoliceCommunity policeCommunity = (PoliceCommunity)super.getDefaultManager().getObject("from PoliceCommunity t " +

" where " +

" t.status=1 " +

" and t.company.id=?", companyId);

if(policeCommunity != null){

res.put("policeCommunityId", policeCommunity.getId());

res.put("policeStationName", policeCommunity.getPoliceStationName());

res.put("policeName", policeCommunity.getPoliceName());

}else{

res.put("policeStationName", "");

res.put("policeName", "");

}

}

//结果输出

PrintWriter out = this.getResponse().getWriter();

out.println(res.toString());

out.flush();

out.close();

}

protected Class getImplementClass()

{

return Rental.class;

}

public String getSearchAddress() {

return searchAddress;

}

public void setSearchAddress(String searchAddress) {

this.searchAddress = searchAddress;

}

public String getSearchLiveTypeId() {

return searchLiveTypeId;

}

public void setSearchLiveTypeId(String searchLiveTypeId) {

this.searchLiveTypeId = searchLiveTypeId;

}

public String getSearchBathroomFacilities() {

return searchBathroomFacilities;

}

public void setSearchBathroomFacilities(String searchBathroomFacilities) {

this.searchBathroomFacilities = searchBathroomFacilities;

}

public String getSearchHouseStructure() {

return searchHouseStructure;

}

public void setSearchHouseStructure(String searchHouseStructure) {

this.searchHouseStructure = searchHouseStructure;

}

}

package com.bizlogic.cz.action;

import com.bizlogic.base.front.action.BaseUserEditAction;

import com.bizlogic.common.Element;

import com.bizlogic.cz.bean.RentalManager;

public class RentalManagerEditAction extends BaseUserEditAction{

protected Element createElement() {

RentalManager rentalManager = new RentalManager();

//初始化页面时，设置聘用类型为专职

rentalManager.setUseType(1);

return rentalManager;

}

protected void prepareShow() {

RentalManager rentalManager = (RentalManager)element;

}

protected boolean prepareSave(Element element) {

RentalManager rentalManager = (RentalManager)element;

rentalManager.setStatus(1);

return super.prepareSave(element);

}

protected Class getElementClass() {

return RentalManager.class;

}

}

package com.bizlogic.cz.action;

import java.util.List;

import com.bizlogic.base.front.action.BaseUserListAction;

import com.bizlogic.common.action.helper.ListInfo;

import com.bizlogic.cz.bean.RentalManager;

public class RentalManagerListAction extends BaseUserListAction{

@Override

protected long doGetCount(ListInfo listInfo) {

return super.getDefaultManager().getCount("from RentalManager a where a.status=1");

}

@Override

protected List doSearch(ListInfo listInfo) {

return super.getDefaultManager().findAll("from RentalManager a where a.status=1 order by a.user.company.id");

}

/\*\*

\* 删除函数(非物理删除)

\*/

protected void deleteObjectById(String rentalManagerId) {

super.getDefaultManager().execute("update RentalManager a set a.status=0 where a.id = ? ", rentalManagerId);

}

protected Class getImplementClass()

{

return RentalManager.class;

}

}

package com.bizlogic.cz.action;

import java.util.Date;

import com.bizlogic.base.front.action.BaseUserEditAction;

import com.bizlogic.common.Element;

import com.bizlogic.cz.bean.RentalManager;

import com.bizlogic.cz.bean.RentalManagerWorkLog;

public class RentalManagerWorkLogEditAction extends BaseUserEditAction{

protected Element createElement() {

RentalManagerWorkLog rentalManagerWorkLog = new RentalManagerWorkLog();

rentalManagerWorkLog.setRentalManager((RentalManager)super.getDefaultManager().getObject("from RentalManager t where t.status=1 and t.user.id=?", getVisit().getId()));

rentalManagerWorkLog.setCreateTime(new Date());

rentalManagerWorkLog.setAddress("安苑北里26号");

rentalManagerWorkLog.setCoordinateX(Math.round(Math.random()\*89+10)+"."+Math.round(Math.random()\*890000000+1000000000)+"N");

rentalManagerWorkLog.setCoordinateY(Math.round(Math.random()\*899+100)+"."+Math.round(Math.random()\*890000000+1000000000)+"E");

rentalManagerWorkLog.setStatus(1);

return rentalManagerWorkLog;

}

protected void prepareShow() {

RentalManagerWorkLog rentalManagerWorkLog = (RentalManagerWorkLog)element;

}

protected boolean prepareSave(Element element) {

RentalManagerWorkLog rentalManagerWorkLog = (RentalManagerWorkLog)element;

return super.prepareSave(rentalManagerWorkLog);

}

protected Class getElementClass() {

return RentalManagerWorkLog.class;

}

}

package com.bizlogic.cz.action;

import java.util.List;

import net.sf.json.JSONObject;

import com.bizlogic.base.front.action.BaseUserListAction;

import com.bizlogic.common.action.helper.ListInfo;

import com.bizlogic.cz.bean.RentalManagerWorkLog;

public class RentalManagerWorkLogListAction extends BaseUserListAction{

//流管员编号

private String rentalManagerId;

//根据签到时间查询

private String searchCreateTime;

@Override

protected long doGetCount(ListInfo listInfo) {

//将rentalId设置session

rentalManagerId = super.getParameterValue("rentalManagerId");

if(rentalManagerId != null){

super.getSession().setAttribute("rentalManagerId",rentalManagerId);

}else{

rentalManagerId = (String)super.getSession().getAttribute("rentalManagerId");

}

sqlCondition = this.getConditions(rentalManagerId,searchCreateTime);

return super.getDefaultManager().getCount("from RentalManagerWorkLog a where 1=1 "+sqlCondition);

}

@Override

protected List doSearch(ListInfo listInfo) {

List list = super.getDefaultManager().find("from RentalManagerWorkLog a where 1=1 "+sqlCondition +" order by a.createTime desc",

listInfo.getFirstNo(), listInfo.getMaxNo());

return list;

}

/\*\*

\* 获得查询条件

\*/

protected String getConditions(String managerId,String createTime)

{

String conditionsStr = " and a.status=1 ";

if(managerId!=null && managerId.compareTo("")!=0){

conditionsStr += " and a.rentalManager.id= '"+managerId + "' ";

}

if(createTime!=null && createTime.compareTo("")!=0){

conditionsStr += " and to\_char(a.createTime,'yyyy-mm-dd')='"+createTime+"'";

}

return conditionsStr;

}

protected Class getImplementClass()

{

return RentalManagerWorkLog.class;

}

public String getSearchCreateTime() {

return searchCreateTime;

}

public void setSearchCreateTime(String searchCreateTime) {

this.searchCreateTime = searchCreateTime;

}

public String getRentalManagerId() {

return rentalManagerId;

}

public void setRentalManagerId(String rentalManagerId) {

this.rentalManagerId = rentalManagerId;

}

}

package com.bizlogic.cz.action;

import java.util.List;

import com.bizlogic.base.AttachmentValue;

import com.bizlogic.base.front.action.BaseUserEditAction;

import com.bizlogic.cz.bean.Homeowner;

import com.bizlogic.cz.bean.Household;

import com.bizlogic.cz.bean.LiabilityLetter;

import com.bizlogic.cz.bean.Rental;

import com.bizlogic.cz.bean.RentalInfo;

import com.bizlogic.cz.bean.RentalSafe;

import com.bizlogic.cz.bean.Tenant;

public class RentalRelatedInfoShowAction extends BaseUserEditAction {

//租户

private List<Household> householdList;

//附件列表

private List<AttachmentValue> attachmentList;

//房屋安全

private RentalSafe rentalSafe;

//房屋id

private String rentalId;

//责任书类型id

private String liabilityLetterTypeName;

//出租信息

private RentalInfo rentalInfo;

//房主

private Homeowner homeowner;

protected void prepareShow() {

Rental rental = (Rental) element;

liabilityLetterTypeName="";

rentalId=rental.getId();

rentalSafe=(RentalSafe)super.getDefaultManager().getObject("from RentalSafe s where s.status=1 and s.rental=?", rental);

//房屋安全不为空时

if(rentalSafe!=null){

//房屋安全id不为空时

if(rentalSafe.getId() != null){

attachmentList = super.getDefaultManager().findAll("from AttachmentValue a where a.objectId=? order by a.id", rentalSafe.getId());

}

}

householdList = super.getDefaultManager().findAll("from Household h where h.rental=? and h.status=?",rental,1);

//租户集合不为空时遍历

if(householdList!=null){

for(Household household:householdList){

List<Tenant> tenantList = super.getDefaultManager().findAll("from Tenant t where t.household=? and t.status=?",household,1);

household.setTenantList(tenantList);

}

}

homeowner=(Homeowner)super.getDefaultManager().getObject("from Homeowner h where h.status=1 and h.rental=?", rental);

rentalInfo=(RentalInfo)super.getDefaultManager().getObject("from RentalInfo r where r.status=1 and r.rental=? and r.status=?", rental,1);

//出租信息不为空时

if(rentalInfo!=null){

List<LiabilityLetter> liabilityLetterList=super.getDefaultManager().findAll("from LiabilityLetter r where r.rentalInfo=?", rentalInfo);

if(liabilityLetterList!=null){

for(LiabilityLetter liabilityLetter:liabilityLetterList){

if(liabilityLetter.getLiabilityLetterType()!=null){

liabilityLetterTypeName += liabilityLetter.getLiabilityLetterType().getName()+";";

}

}

}

}

}

public List<Household> getHouseholdList() {

return householdList;

}

protected Class getElementClass() {

return Rental.class;

}

public Homeowner getHomeowner() {

return homeowner;

}

public void setHomeowner(Homeowner homeowner) {

this.homeowner = homeowner;

}

public RentalInfo getRentalInfo() {

return rentalInfo;

}

public void setRentalInfo(RentalInfo rentalInfo) {

this.rentalInfo = rentalInfo;

}

public String getLiabilityLetterTypeName() {

return liabilityLetterTypeName;

}

public void setLiabilityLetterTypeName(String liabilityLetterTypeName) {

this.liabilityLetterTypeName = liabilityLetterTypeName;

}

public String getRentalId() {

return rentalId;

}

public void setRentalId(String rentalId) {

this.rentalId = rentalId;

}

public RentalSafe getRentalSafe() {

return rentalSafe;

}

public void setRentalSafe(RentalSafe rentalSafe) {

this.rentalSafe = rentalSafe;

}

public List<AttachmentValue> getAttachmentList() {

return attachmentList;

}

public void setAttachmentList(List<AttachmentValue> attachmentList) {

this.attachmentList = attachmentList;

}

}

package com.bizlogic.cz.action;

import java.io.File;

import java.util.Date;

import java.util.List;

import org.apache.commons.lang.StringUtils;

import com.bizlogic.base.AttachmentValue;

import com.bizlogic.base.User;

import com.bizlogic.base.front.action.BaseUserEditAction;

import com.bizlogic.base.helper.AttachmentHelper;

import com.bizlogic.common.Element;

import com.bizlogic.cz.bean.LiveType;

import com.bizlogic.cz.bean.Rental;

import com.bizlogic.cz.bean.RentalSafe;

import com.bizlogic.cz.bean.WaterHeaterType;

public class RentalSafeEditAction extends BaseUserEditAction{

//附件上传----start

private AttachmentHelper attachmentHelper;

//洗浴设施附件

private File[] fileAttachmentBathroom;

private String[] fileAttachmentBathroomContentType;

private String[] fileAttachmentBathroomFileName;

//用电情况附件

private File[] fileAttachmentElectric;

private String[] fileAttachmentElectricContentType;

private String[] fileAttachmentElectricFileName;

//房屋结构附件

private File[] fileAttachmentStructure;

private String[] fileAttachmentStructureContentType;

private String[] fileAttachmentStructureFileName;

private long serial = 0;

//附件列表

private List<AttachmentValue> attachmentList;

//出租房屋编号

private String rentalId;

//热水器类型编号

private String waterHeaterTypeId;

//居住类型编号

private String liveTypeId;

//居住类型列表

private List<LiveType> liveTypeList;

//热水器类型列表

private List<WaterHeaterType> waterHeaterTypeList;

public List<WaterHeaterType> getWaterHeaterTypeList() {

return super.getDefaultManager().findAll("from WaterHeaterType");

}

public List<LiveType> getLiveTypeList() {

return super.getDefaultManager().findAll("from LiveType t where t.status=1");

}

protected Element createElement() {

RentalSafe rentalSafe = new RentalSafe();

return rentalSafe;

}

protected void prepareShow() {

RentalSafe rentalSafe = (RentalSafe) element;

//将rentalId设置session

rentalId = super.getParameterValue("rentalId");

if(rentalId != null){

super.getSession().setAttribute("rentalId",rentalId);

}else{

rentalId = (String)super.getSession().getAttribute("rentalId");

}

this.liveTypeId = rentalSafe.getLiveType() == null ? null : rentalSafe.getLiveType().getId();

this.waterHeaterTypeId=rentalSafe.getWaterHeaterType()==null?null:rentalSafe.getWaterHeaterType().getId();

if(rentalSafe.getId() != null){

attachmentList = super.getDefaultManager().findAll("from AttachmentValue a where a.objectId=? order by a.id", rentalSafe.getId());

}

}

protected boolean prepareSave(Element element) {

RentalSafe rentalSafe = (RentalSafe) element;

//设置出租房屋

rentalSafe.setRental(super.getDefaultManager().get(Rental.class, rentalId));

if (StringUtils.isNotBlank(liveTypeId)) {

rentalSafe.setLiveType((LiveType) super.getDefaultManager().get(LiveType.class,liveTypeId));

} else {

rentalSafe.setLiveType(null);

}

if (StringUtils.isNotBlank(waterHeaterTypeId)) {

rentalSafe.setWaterHeaterType((WaterHeaterType) super.getDefaultManager().get(WaterHeaterType.class,waterHeaterTypeId));

} else {

rentalSafe.setWaterHeaterType(null);

}

rentalSafe.setCreateUser((User)getVisit());

rentalSafe.setCreateDate(new Date());

rentalSafe.setStatus(1);

return super.prepareSave(element);

}

public String save() {

RentalSafe rentalSafe = (RentalSafe) element;

super.save();

this.setId(element.getId());

//将之前安全信息改为历史信息

super.getDefaultManager().execute("update RentalSafe t set t.status=2 where t.status=1 and t.rental.id=? and t.id != ?", rentalId, rentalSafe.getId());

// 添加附件

if (fileAttachmentBathroom != null)

this.addAttachmentBathroom();

if (fileAttachmentElectric != null)

this.addAttachmentElectric();

if (fileAttachmentStructure != null)

this.addAttachmentStructure();

return SUCCESS;

}

//添加附件

private String addAttachmentBathroom()

{

//传入DAO

attachmentHelper.init(super.getDefaultManager());

//洗浴设施附件

for(int i=0;i<fileAttachmentBathroom.length;i++)

{

AttachmentValue value = new AttachmentValue();

value.setContentType(fileAttachmentBathroomContentType[i]);

value.setFileName(fileAttachmentBathroomFileName[i]);

value.setType("RENTAL\_SAFE\_BATHROOM");

value.setName(fileAttachmentBathroomFileName[i].substring(0,fileAttachmentBathroomFileName[i].lastIndexOf(".")));

value.setSize(fileAttachmentBathroom[i].length());

value.setUploadFile(fileAttachmentBathroom[i]);

value.setObjectId(id);

value.setRealName(System.currentTimeMillis() + (serial++) + "." + attachmentHelper.getFileExtName(value.getFileName()) );

value.setCreateTime(new java.util.Date());

value.setCreateUser((User)getVisit());

attachmentHelper.saveAttachment(value);

}

return SUCCESS;

}

//添加附件

private String addAttachmentElectric()

{

//传入DAO

attachmentHelper.init(super.getDefaultManager());

//用电情况附件

for(int i=0;i<fileAttachmentElectric.length;i++)

{

AttachmentValue value = new AttachmentValue();

value.setContentType(fileAttachmentElectricContentType[i]);

value.setFileName(fileAttachmentElectricFileName[i]);

value.setType("RENTAL\_SAFE\_ELECTRIC");

value.setName(fileAttachmentElectricFileName[i].substring(0,fileAttachmentElectricFileName[i].lastIndexOf(".")));

value.setSize(fileAttachmentElectric[i].length());

value.setUploadFile(fileAttachmentElectric[i]);

value.setObjectId(id);

value.setRealName(System.currentTimeMillis() + (serial++) + "." + attachmentHelper.getFileExtName(value.getFileName()) );

value.setCreateTime(new java.util.Date());

value.setCreateUser((User)getVisit());

attachmentHelper.saveAttachment(value);

}

return SUCCESS;

}

//添加附件

private String addAttachmentStructure()

{

//传入DAO

attachmentHelper.init(super.getDefaultManager());

//房屋结构附件

for(int i=0;i<fileAttachmentStructure.length;i++)

{

AttachmentValue value = new AttachmentValue();

value.setContentType(fileAttachmentStructureContentType[i]);

value.setFileName(fileAttachmentStructureFileName[i]);

value.setType("RENTAL\_SAFE\_STRUCTURE");

value.setName(fileAttachmentStructureFileName[i].substring(0,fileAttachmentStructureFileName[i].lastIndexOf(".")));

value.setSize(fileAttachmentStructure[i].length());

value.setUploadFile(fileAttachmentStructure[i]);

value.setObjectId(id);

value.setRealName(System.currentTimeMillis() + (serial++) + "." + attachmentHelper.getFileExtName(value.getFileName()) );

value.setCreateTime(new java.util.Date());

value.setCreateUser((User)getVisit());

attachmentHelper.saveAttachment(value);

}

return SUCCESS;

}

protected Class getElementClass() {

return RentalSafe.class;

}

public String getLiveTypeId() {

return liveTypeId;

}

public void setLiveTypeId(String liveTypeId) {

this.liveTypeId = liveTypeId;

}

public String getRentalId() {

return rentalId;

}

public void setRentalId(String rentalId) {

this.rentalId = rentalId;

}

public AttachmentHelper getAttachmentHelper() {

return attachmentHelper;

}

public void setAttachmentHelper(AttachmentHelper attachmentHelper) {

this.attachmentHelper = attachmentHelper;

}

public File[] getFileAttachmentBathroom() {

return fileAttachmentBathroom;

}

public void setFileAttachmentBathroom(File[] fileAttachmentBathroom) {

this.fileAttachmentBathroom = fileAttachmentBathroom;

}

public File[] getFileAttachmentElectric() {

return fileAttachmentElectric;

}

public void setFileAttachmentElectric(File[] fileAttachmentElectric) {

this.fileAttachmentElectric = fileAttachmentElectric;

}

public File[] getFileAttachmentStructure() {

return fileAttachmentStructure;

}

public void setFileAttachmentStructure(File[] fileAttachmentStructure) {

this.fileAttachmentStructure = fileAttachmentStructure;

}

public String[] getFileAttachmentBathroomContentType() {

return fileAttachmentBathroomContentType;

}

public String[] getFileAttachmentElectricContentType() {

return fileAttachmentElectricContentType;

}

public String[] getFileAttachmentStructureContentType() {

return fileAttachmentStructureContentType;

}

public void setFileAttachmentBathroomContentType(

String[] fileAttachmentBathroomContentType) {

this.fileAttachmentBathroomContentType = fileAttachmentBathroomContentType;

}

public void setFileAttachmentElectricContentType(

String[] fileAttachmentElectricContentType) {

this.fileAttachmentElectricContentType = fileAttachmentElectricContentType;

}

public void setFileAttachmentStructureContentType(

String[] fileAttachmentStructureContentType) {

this.fileAttachmentStructureContentType = fileAttachmentStructureContentType;

}

public String[] getFileAttachmentBathroomFileName() {

return fileAttachmentBathroomFileName;

}

public String[] getFileAttachmentElectricFileName() {

return fileAttachmentElectricFileName;

}

public String[] getFileAttachmentStructureFileName() {

return fileAttachmentStructureFileName;

}

public void setFileAttachmentBathroomFileName(

String[] fileAttachmentBathroomFileName) {

this.fileAttachmentBathroomFileName = fileAttachmentBathroomFileName;

}

public void setFileAttachmentElectricFileName(

String[] fileAttachmentElectricFileName) {

this.fileAttachmentElectricFileName = fileAttachmentElectricFileName;

}

public void setFileAttachmentStructureFileName(

String[] fileAttachmentStructureFileName) {

this.fileAttachmentStructureFileName = fileAttachmentStructureFileName;

}

public List<AttachmentValue> getAttachmentList() {

return attachmentList;

}

public void setAttachmentList(List<AttachmentValue> attachmentList) {

this.attachmentList = attachmentList;

}

public String getWaterHeaterTypeId() {

return waterHeaterTypeId;

}

public void setWaterHeaterTypeId(String waterHeaterTypeId) {

this.waterHeaterTypeId = waterHeaterTypeId;

}

}

package com.bizlogic.cz.action;

import java.util.List;

import com.bizlogic.base.front.action.BaseUserListAction;

import com.bizlogic.common.action.helper.ListInfo;

import com.bizlogic.cz.bean.RentalSafe;

public class RentalSafeListAction extends BaseUserListAction{

//出租房屋编号

private String rentalId;

@Override

protected long doGetCount(ListInfo listInfo) {

//将rentalId设置session

rentalId = super.getParameterValue("rentalId");

if(rentalId != null){

super.getSession().setAttribute("rentalId",rentalId);

}else{

rentalId = (String)super.getSession().getAttribute("rentalId");

}

sqlCondition = this.getConditions(rentalId);

return super.getDefaultManager().getCount("from RentalSafe a where 1=1 "+sqlCondition);

}

@Override

protected List doSearch(ListInfo listInfo) {

return super.getDefaultManager().find("from RentalSafe a where 1=1 "+sqlCondition + " order by a.createDate desc",

listInfo.getFirstNo(), listInfo.getMaxNo());

}

/\*\*

\* 获得查询条件

\*/

protected String getConditions(String rentalIdFlag)

{

//查询状态为1的有效数据和状态为2的历史数据

String conditionsStr = " and (a.status=1 or a.status=2)";

if(rentalIdFlag!=null && rentalIdFlag.compareTo("")!=0){

conditionsStr += " and a.rental.id='" + rentalIdFlag + "'";

}

return conditionsStr;

}

/\*\*

\* 删除函数(非物理删除)

\*/

@Override

protected void deleteObjectById(String rentalSafeId) {

super.getDefaultManager().execute("update RentalSafe t set t.status=0 where t.id=?", rentalSafeId);

}

protected Class getImplementClass()

{

return RentalSafe.class;

}

}

package com.bizlogic.cz.action;

import java.util.List;

import com.bizlogic.base.AttachmentValue;

import com.bizlogic.base.front.action.BaseUserEditAction;

import com.bizlogic.cz.bean.RentalSafe;

public class RentalSafeShowAction extends BaseUserEditAction{

//附件列表

private List<AttachmentValue> attachmentList;

//出租房屋编号

private String rentalId;

protected void prepareShow() {

RentalSafe rentalSafe = (RentalSafe) element;

//将rentalId设置session

rentalId = super.getParameterValue("rentalId");

if(rentalId != null){

super.getSession().setAttribute("rentalId",rentalId);

}else{

rentalId = (String)super.getSession().getAttribute("rentalId");

}

if(rentalSafe.getId() != null){

attachmentList = super.getDefaultManager().findAll("from AttachmentValue a where a.objectId=? order by a.id", rentalSafe.getId());

}

}

protected Class getElementClass() {

return RentalSafe.class;

}

public String getRentalId() {

return rentalId;

}

public void setRentalId(String rentalId) {

this.rentalId = rentalId;

}

public List<AttachmentValue> getAttachmentList() {

return attachmentList;

}

public void setAttachmentList(List<AttachmentValue> attachmentList) {

this.attachmentList = attachmentList;

}

}

package com.bizlogic.cz.action;

import java.io.IOException;

import java.io.PrintWriter;

import java.util.ArrayList;

import java.util.List;

import net.sf.json.JSONObject;

import org.apache.commons.lang.StringUtils;

import com.bizlogic.base.front.action.BaseUserListAction;

import com.bizlogic.common.action.helper.ListInfo;

import com.bizlogic.cz.bean.Homeowner;

import com.bizlogic.cz.bean.Rental;

import com.bizlogic.cz.bean.RentalInfo;

import com.bizlogic.cz.bean.RentalSafe;

import com.bizlogic.cz.bean.WaterHeaterType;

import com.bizlogic.qms.bean.Building;

import com.bizlogic.qms.bean.Room;

public class RentalSearchListAction extends BaseUserListAction{

//热水器类型列表

private List<WaterHeaterType> waterHeaterTypeList;

public List<WaterHeaterType> getWaterHeaterTypeList() {

return super.getDefaultManager().findAll("from WaterHeaterType");

}

//地址

private String searchAddress;

//居住类型

private List liveTypeList;

public List getLiveTypeList() {

return super.getDefaultManager().findAll("from LiveType a where a.status=1 order by a.id ");

}

private String searchLiveTypeId;

//热水器类型

private String searchBathroomFacilities;

//房屋结构 :0为未改动，1为私设隔断

private String searchHouseStructure;

@Override

protected long doGetCount(ListInfo listInfo) {

long l = 0;

//根据安全信息进行查询

if((searchLiveTypeId != null && !"".equals(searchLiveTypeId))

|| (searchBathroomFacilities != null && !"".equals(searchBathroomFacilities))

|| "true".equals(searchHouseStructure)

)

{

sqlCondition = this.getConditionsBySafe( searchLiveTypeId, searchBathroomFacilities, searchHouseStructure);

l = super.getDefaultManager().getCount("from RentalSafe a where 1=1 "+sqlCondition);

}else{

sqlCondition = this.getConditions(1,searchAddress);

l = super.getDefaultManager().getCount("from Rental a where 1=1 "+sqlCondition);

}

return l;

}

@Override

protected List doSearch(ListInfo listInfo) {

List list = new ArrayList();

if((searchLiveTypeId != null && !"".equals(searchLiveTypeId))

|| (searchBathroomFacilities != null && !"".equals(searchBathroomFacilities))

|| "true".equals(searchHouseStructure)

)

{

list = super.getDefaultManager().find("select a.rental from RentalSafe a where 1=1 "+sqlCondition,

listInfo.getFirstNo(), listInfo.getMaxNo());

}else{

list = super.getDefaultManager().find("from Rental a where 1=1 "+sqlCondition,

listInfo.getFirstNo(), listInfo.getMaxNo());

}

//加载安全信息

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

Rental rental = (Rental)list.get(i);

//获得安全信息

String[] str = {"rental.id","status"};

Object[] pro = {rental.getId(),1};

RentalSafe rentalSafe = (RentalSafe)super.getDefaultManager().getByPropertys(RentalSafe.class, str,pro);

if(rentalSafe != null){

rental.setRentalSafe(rentalSafe);

}

}

return list;

}

/\*\*

\* 删除函数(非物理删除)

\*/

@Override

protected void deleteObjectById(String rentId) {

super.getDefaultManager().execute("update Rental t set t.status=0 where t.id=?", rentId);

}

/\*\*

\* 获得查询条件

\*/

protected String getConditions(int status, String address)

{

String conditionsStr = " and a.status="+status;

if(address!=null && address.compareTo("")!=0){

conditionsStr += " and a.address like '%"+ address.trim() +"%'";

}

return conditionsStr;

}

/\*\*

\* 获得查询条件

\*/

protected String getConditionsBySafe(String liveTypeId ,String bathroomFacilities ,String houseStructure )

{

String conditionsStr = "";

if(liveTypeId!=null && liveTypeId.compareTo("")!=0){

conditionsStr += " and a.liveType.id = '"+ liveTypeId.trim() +"'";

}

if(bathroomFacilities!=null && bathroomFacilities.compareTo("")!=0){

conditionsStr += " and a.waterHeaterType.id = '"+ bathroomFacilities.trim() +"'";

}

if(houseStructure!=null && "true".equals(houseStructure.trim())){

conditionsStr += " and a.houseStructure = '1'";

}

return conditionsStr;

}

/\*\*

\* ajax请求

\* @throws IOException

\*/

public void ajaxMethod() throws IOException {

JSONObject res = new JSONObject();

//Response编码设置

super.getResponse().setCharacterEncoding("utf-8");

//获取房主编号

if(StringUtils.equals(act, "get\_home\_owner\_id")) {

String rentalId = super.getParameterValue("rentalId");

//获取房主信息

Homeowner homeowner = (Homeowner)super.getDefaultManager().getObject("from Homeowner t where t.status=1 and t.rental.id=?", rentalId);

//如果有房主则返回房主编号，否则返回空字符串

if(homeowner != null){

res.put("homeownerId", homeowner.getId());

}else{

res.put("homeownerId", "");

}

//返回房屋编号

res.put("rentalId", rentalId);

}

//获取出租房安全信息编号

if(StringUtils.equals(act, "get\_rental\_safe\_id")) {

String rentalId = super.getParameterValue("rentalId");

//获取安全信息

RentalSafe rentalSafe = (RentalSafe)super.getDefaultManager().getObject("from RentalSafe t where t.status=1 and t.rental.id=?", rentalId);

//如果有安全信息不为空则返回安全信息编号，否则返回空字符串

if(rentalSafe != null){

res.put("rentalSafeId", rentalSafe.getId());

}else{

res.put("rentalSafeId", "");

}

//返回出租房编号

res.put("rentalId", rentalId);

}

//获取出租房出租信息编号

if(StringUtils.equals(act, "get\_rentalInfo\_id")) {

String rentalId = super.getParameterValue("rentalId");

//获取出租信息

RentalInfo rentalInfo = (RentalInfo)super.getDefaultManager().getObject("from RentalInfo t where t.status=1 and t.rental.id=?", rentalId);

//如果有出租信息则返回出租信息编号，否则返回空字符串

if(rentalInfo != null){

res.put("rentalInfoId", rentalInfo.getId());

}else{

res.put("rentalInfoId", "");

}

//返回房屋编号

res.put("rentalId", rentalId);

}

//根据社区编号获取社区对应的楼

if(StringUtils.equals(act, "get\_building\_list")) {

//社区编号

String companyId = super.getParameterValue("companyId");

List<Building> buildingList = super.getDefaultManager().findAll("from Building t where t.company.id=?", companyId);

StringBuffer str = new StringBuffer();

str.append("<div class='selectBuilding\_bg'>");

str.append("<div class='selectBuilding\_dh'>选择楼<input type='button' class='closed' value='关闭' onclick='javascript:closeDiv2();return false;'></div>");

str.append("<div style='width:100%;height:90%;overflow:auto;'>");

str.append("<table cellpadding='0' cellspacing='0' class='list\_tb'>");

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

Building building = (Building)buildingList.get(i);

if(i == 0 || i%6 == 0){

str.append("<tr>");

}

str.append("<td width='16%' style='text-align:center'><a href='javascript:selectRoom("+'"'+building.getId()+'"'+")'>"+building.getBuildingAddress()+"</a></td>");

if(i != 0 && (i+1)%6 == 0){

str.append("</tr>");

}

if(i==buildingList.size()-1 && (i+1)%6 > 0){

for(int x=0;x<6-(i+1)%6;x++){

str.append("<td width='16%' style='text-align:center'>&nbsp;</td>");

if(x==5-(i+1)%6){

str.append("</tr>");

}

}

}

}

str.append("</table>");

str.append("</div>");

str.append("</div>");

res.put("message", str.toString());

}

//根据楼层号获取房间列表

if(StringUtils.equals(act, "get\_room\_list")) {

//楼编号

String buildingId = super.getParameterValue("buildingId");

List<Room> roomList = super.getDefaultManager().findAll("from Room t where t.building.id=?", buildingId);

StringBuffer str = new StringBuffer();

str.append("<div class='selectBuilding\_bg'>");

str.append("<div class='selectBuilding\_dh'>选择房间<input type='button' class='closed' value='关闭' onclick='javascript:closeDiv2();return false;'></div>");

str.append("<div style='width:100%;height:85%;overflow:auto;'>");

str.append("<table cellpadding='0' cellspacing='0' class='list\_tb'>");

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

Room room = (Room)roomList.get(i);

if(i == 0 || i%6 == 0){

str.append("<tr>");

}

str.append("<td width='16%' style='text-align:center'><a href='javascript:setAddress("+'"'+room.getRoomAddress()+'"'+")'>"+room.getRoomNum()+"("+room.getRoomAddress()+")"+"</a></td>");

if(i != 0 && (i+1)%6 == 0){

str.append("</tr>");

}

if(i==roomList.size()-1 && (i+1)%6 > 0){

for(int x=0;x<6-(i+1)%6;x++){

str.append("<td width='16%' style='text-align:center'>&nbsp;</td>");

if(x==5-(i+1)%6){

str.append("</tr>");

}

}

}

}

str.append("</table>");

str.append("</div>");

str.append("<input type='button' class='pop\_btn' value='返回楼列表' onclick='javascript:backToBuilding();return false;'>");

str.append("</div>");

res.put("message", str.toString());

}

//结果输出

PrintWriter out = this.getResponse().getWriter();

out.println(res.toString());

out.flush();

out.close();

}

protected Class getImplementClass()

{

return Rental.class;

}

public String getSearchAddress() {

return searchAddress;

}

public void setSearchAddress(String searchAddress) {

this.searchAddress = searchAddress;

}

public String getSearchLiveTypeId() {

return searchLiveTypeId;

}

public void setSearchLiveTypeId(String searchLiveTypeId) {

this.searchLiveTypeId = searchLiveTypeId;

}

public String getSearchBathroomFacilities() {

return searchBathroomFacilities;

}

public void setSearchBathroomFacilities(String searchBathroomFacilities) {

this.searchBathroomFacilities = searchBathroomFacilities;

}

public String getSearchHouseStructure() {

return searchHouseStructure;

}

public void setSearchHouseStructure(String searchHouseStructure) {

this.searchHouseStructure = searchHouseStructure;

}

}

package com.bizlogic.cz.action;

import java.util.List;

import com.bizlogic.base.AttachmentValue;

import com.bizlogic.base.front.action.BaseUserEditAction;

import com.bizlogic.cz.bean.Homeowner;

import com.bizlogic.cz.bean.Household;

import com.bizlogic.cz.bean.LiabilityLetter;

import com.bizlogic.cz.bean.Rental;

import com.bizlogic.cz.bean.RentalInfo;

import com.bizlogic.cz.bean.RentalSafe;

import com.bizlogic.cz.bean.Tenant;

public class RentalSearchShowAction extends BaseUserEditAction {

//租户

private List<Household> householdList;

//附件列表

private List<AttachmentValue> attachmentList;

//房屋安全

private RentalSafe rentalSafe;

//房屋id

private String rentalId;

//责任书类型id

private String liabilityLetterTypeName;

//出租信息

private RentalInfo rentalInfo;

//房主

private Homeowner homeowner;

protected void prepareShow() {

Rental rental = (Rental) element;

rentalId=rental.getId();

rentalSafe=(RentalSafe)super.getDefaultManager().getObject("from RentalSafe s where s.status=1 and s.rental=?", rental);

//房屋安全不为空时

if(rentalSafe!=null){

//房屋安全id不为空时

if(rentalSafe.getId() != null){

attachmentList = super.getDefaultManager().findAll("from AttachmentValue a where a.objectId=? order by a.id", rentalSafe.getId());

}

}

householdList = super.getDefaultManager().findAll("from Household h where h.rental=? and h.status=?",rental,1);

//租户集合不为空时遍历

if(householdList!=null){

for(Household household:householdList){

List<Tenant> tenantList = super.getDefaultManager().findAll("from Tenant t where t.household=? and t.status=?",household,1);

household.setTenantList(tenantList);

}

}

homeowner=(Homeowner)super.getDefaultManager().getObject("from Homeowner h where h.status=1 and h.rental=?", rental);

rentalInfo=(RentalInfo)super.getDefaultManager().getObject("from RentalInfo r where r.status=1 and r.rental=? and r.status=?", rental,1);

//出租信息不为空时

if(rentalInfo!=null){

LiabilityLetter liabilityLetter=(LiabilityLetter)super.getDefaultManager().getObject("from LiabilityLetter r where r.rentalInfo=?", rentalInfo);

if(liabilityLetter!=null)

liabilityLetterTypeName=liabilityLetter.getLiabilityLetterType().getName();

}

}

public List<Household> getHouseholdList() {

return householdList;

}

protected Class getElementClass() {

return Rental.class;

}

public Homeowner getHomeowner() {

return homeowner;

}

public void setHomeowner(Homeowner homeowner) {

this.homeowner = homeowner;

}

public RentalInfo getRentalInfo() {

return rentalInfo;

}

public void setRentalInfo(RentalInfo rentalInfo) {

this.rentalInfo = rentalInfo;

}

public String getLiabilityLetterTypeName() {

return liabilityLetterTypeName;

}

public void setLiabilityLetterTypeName(String liabilityLetterTypeName) {

this.liabilityLetterTypeName = liabilityLetterTypeName;

}

public String getRentalId() {

return rentalId;

}

public void setRentalId(String rentalId) {

this.rentalId = rentalId;

}

public RentalSafe getRentalSafe() {

return rentalSafe;

}

public void setRentalSafe(RentalSafe rentalSafe) {

this.rentalSafe = rentalSafe;

}

public List<AttachmentValue> getAttachmentList() {

return attachmentList;

}

public void setAttachmentList(List<AttachmentValue> attachmentList) {

this.attachmentList = attachmentList;

}

}

package com.bizlogic.cz.action;

import java.io.UnsupportedEncodingException;

import java.util.List;

import com.bizlogic.base.front.action.BaseUserListAction;

import com.bizlogic.common.action.helper.ListInfo;

import com.bizlogic.cz.bean.RentalManager;

import org.apache.commons.lang.StringUtils;

import com.bizlogic.base.CertificateType;

import com.bizlogic.base.Country;

import com.bizlogic.common.Element;

import com.bizlogic.cz.bean.DomicileType;

import com.bizlogic.cz.bean.Homeowner;

import com.bizlogic.cz.bean.Rental;

public class RentalUserJsonAction extends BaseUserListAction{

//返回值

private List allowanceList;

//总页数

private int pageCount = 0;

//每页显示行数(10)

protected int pageRow =10 ;

//struts2返回json对象

public String getJson()

{

String sPage = super.getRequest().getParameter("page"); //插件内部自动封装了 page 参数，使用者不需要操作 page 参数，插件自动传参

//按照名称查询

String searchName =super.getRequest().getParameter("searchName"); //param 参数中带的 arg2

if(searchName!=null){

try {

searchName = java.net.URLDecoder.decode(searchName, "UTF-8");

} catch (UnsupportedEncodingException e) {}

}

if(searchName!=null && !"".equals(searchName)){

sqlCondition = " and a.name like '%"+searchName.trim()+"%'";

}

int totalCount = 0;

if (totalCount>0) {

pageCount=totalCount/pageRow;

//求余数

int a = totalCount%pageRow;

if (a>0) {

pageCount++;

}

}

//页数减1

int startPage =(Integer.parseInt(sPage)-1)\*pageRow ;

int endPage = startPage+pageRow;

allowanceList = super.getDefaultManager().findSQL(

"select a.user\_id,"

+"a.name as userName,"

+"a.mobile,b.name as companyName"

+" from t\_bpm\_user a,t\_bpm\_company b"

+" where a.company\_id = b.company\_id(+) "

+" and a.user\_id not in (select c.user\_id from t\_cz\_rental\_manager c where c.status=1)"

+ sqlCondition, startPage, endPage);

return SUCCESS;

}

@Override

protected long doGetCount(ListInfo listInfo) {

return super.getDefaultManager().getCount("from RentalManager a where 1=1 "+ sqlCondition);

}

@Override

protected List doSearch(ListInfo listInfo) {

List list = super.getDefaultManager().find("from RentalManager a where 1=1 "+ sqlCondition,

listInfo.getFirstNo(), listInfo.getMaxNo());

return list;

}

protected Class getImplementClass()

{

return RentalManager.class;

}

public List getAllowanceList() {

return allowanceList;

}

public void setAllowanceList(List allowanceList) {

this.allowanceList = allowanceList;

}

public int getPageCount() {

return pageCount;

}

public void setPageCount(int pageCount) {

this.pageCount = pageCount;

}

--------END------------