

06 户主信息管理

笔记本： 课堂笔记-房屋租赁系统

创建时间： 2020/9/8 21:51

更新时间： 2020/9/19 11:41

作者： yishuihan1104@163.com

1、创建户主信息前端页面

1.1 更改url连接



1.2 创建对应的静态页面(户主首页)

复制拷贝table.html即可 需要更改名称信息

2、户主查询功能实现

2.1 创建model对象类

```
package com.yanzhen.model;

import com.baomidou.mybatisplus.annotation.IdType;
import java.util.Date;
import com.baomidou.mybatisplus.annotation.TableId;
import java.io.Serializable;
```

```
import io.swagger.annotations.ApiModel;
import io.swagger.annotations.ApiModelProperty;
import lombok.Data;
import lombok.EqualsAndHashCode;
import lombok.experimental.Accessors;

/**
 * <p>
 * 房东信息表
 * </p>
 *
 * @author maqh
 * @since 2020-08-21
 */
@Data
@EqualsAndHashCode(callSuper = false)
@Accessors(chain = true)
@ApiModel(value="Owner对象", description="房东信息表")
public class Owner implements Serializable {

    private static final long serialVersionUID = 1L;

    @TableId(value = "id", type = IdType.AUTO)
    private Integer id;

    private String identity;

    private String custname;

    @ApiModelProperty(value = "0 代表女 1 代表男")
    private String sex;

    private String address;

    private String phone;

    private String career;

    private String remarks;

    private Date createTime;

    private String djr;
```

```
}
```

2.2 dao层接口和实现

```
package com.yanzhen.dao;

import com.baomidou.mybatisplus.core.mapper.BaseMapper;
import com.yanzhen.model.Owner;
import org.springframework.stereotype.Component;

import java.util.List;

@Component("ownerDao")
public interface OwnerMapper extends BaseMapper<Owner> {

    /**
     * 查询所有的户主信息
     */
    List<Owner> queryOwnerAll(Owner owner);
}
```

实现:

```
<?xml version="1.0" encoding="UTF-8" ?>
<!DOCTYPE mapper
    PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="com.yanzhen.dao.OwnerMapper">

    <select resultType="com.yanzhen.model.Owner"
        parameterType="com.yanzhen.model.Owner">
        select * from owner
        <where>
            <if test="identity!=null and identity!=''">
                and identity like '%${identity}%'
            </if>
            <if test="custname!=null and custname!=''">
                and custname like '%${custname}%'
            </if>
        </where>
    </select>
</mapper>
```

```
        </where>
    </select>

</mapper>
```

2.2 service的接口和实现

```
package com.yanzhen.service;

import com.github.pagehelper.PageInfo;
import com.yanzhen.model.Owner;

public interface OwnerService {

    /**
     * 分页查询户主信息
     */
    PageInfo<Owner> findOwnerAll(int page, int limit, Owner owner);
}
```

实现

```
package com.yanzhen.service;

import com.github.pagehelper.PageHelper;
import com.github.pagehelper.PageInfo;
import com.yanzhen.dao.OwnerMapper;
import com.yanzhen.model.Owner;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Service;

import java.util.List;

@Service("ownerService")
public class OwnerServiceImpl implements OwnerService{

    @Autowired
    private OwnerMapper ownerDao;

    @Override
    public PageInfo<Owner> findOwnerAll(int page, int limit, Owner owner) {
        //分页
        PageHelper.startPage(page, limit);
```

```

        //查询所有的户主信息
        List<Owner> list=ownerDao.queryOwnerAll(owner);
        PageInfo<Owner> pageinfo=new PageInfo<>(list);
        return pageinfo;
    }
}

```

2.3 Controller层进行编写

```

@Autowired
private OwnerService ownerService;

/**
 * 按条件查询房主信息
 */
@RequestMapping("/queryOwnerAll")
@ResponseBody
public Object queryOwnerAll(@RequestParam(defaultValue = "1") Integer
page, @RequestParam(defaultValue = "10")Integer limit, Owner owner){
    JsonObject object=new JsonObject();
    //分页查询所有的记录信息
    PageInfo<Owner> pageInfo=
ownerService.findOwnerAll(page, limit, owner);
    object.setCode(0);
    object.setCount(pageInfo.getTotal());
    object.setData(pageInfo.getList());
    object.setMsg("ok");
    return object;
}

```

2.4 前端页面获取房主列表信息

```

table.render({
    elem: '#currentTableId',
    url: 'http://localhost:8888/owner/queryOwnerAll',
    toolbar: '#toolbarDemo',
    defaultToolbar: ['filter', 'exports', 'print', {
        title: '提示',

```

```

        layEvent: 'LAYTABLE_TIPS',
        icon: 'layui-icon-tips'
    }],
    cols: [[
        {type: "checkbox", width: 50},
        {field: 'id', width: 80, title: 'ID', sort: true},
        {field: 'custname', width: 80, title: '姓名'},
        {field: 'sex', width: 80, title: '性别'},
        {field: 'address', width: 80, title: '地址'},
        {field: 'phone', title: '电话', minWidth: 150},
        {field: 'career', width: 80, title: '职业'},
        {title: '操作', minWidth: 150, toolbar: '#currentTableBar',
align: "center"}
    ]],
    limits: [10, 15, 20, 25, 50, 100],
    limit: 15,
    page: true,
    skin: 'line'
});

```

2.5 高级查询

```

<div >
    姓名:
    <div >
        <input name="custname" autocomplete="off">
    </div>
    身份证号码:
    <div >
        <input name="identity" autocomplete="off">
    </div>
    <button data-type="reload">高级查询</button>
</div>

```

js相关事件

```

var $ = layui.$, active = {
    reload: function() {
        var custname = $('#custname');
        var identity = $('#identity');
        //执行重载
    }
};

```

```

        table.reload('testReload', {
            page: {
                curr: 1 //重新从第 1 页开始
            }
            ,where: {
                identity:identity.val(),
                custname:custname.val()
            }
        }, 'data');
    }
};

$('.demoTable .layui-btn').on('click', function() {
    var type = $(this).data('type');
    active[type] ? active[type].call(this) : '';
});

```

注意点:

table.reload('testReload', {

testReload: 来自于table中对应的id

2.6 效果图



3、添加户主信息

3.1 监听按钮打开页面

```

table.on('toolbar(currentTableFilter)', function (obj) {
    if (obj.event === 'add') { // 监听添加操作
        var content = miniPage.getHrefContent('page/owner/add.html');
        var openWH = miniPage.getOpenWidthHeight();

        var index = layer.open({
            title: '添加用户信息',
            type: 1,
            shade: 0.2,
            maxmin: true,
            shadeClose: true,

```

3.2 设计add.html页面

```

<div >

    <div >
        <div >
            <label >身份证号</label>
            <div >
                <input type="text" name="identity" lay-verify="required" lay-reqtext="身份证号不能为空" placeholder="请输入用户名" value="" >
                <tip>填写自己管理账号的名称。</tip>
            </div>
        </div>
    </div>
    <div >
        <label >房主姓名</label>
        <div >
            <input type="text" name="custname" lay-verify="required" lay-reqtext="姓名不能为空" placeholder="请输入用户名" value="" >
        </div>
    </div>

    <div >
        <label >性别</label>
        <div >
            <input type="radio" name="sex" value="男" title="男">
            <input type="radio" name="sex" value="女" title="女">
        </div>
    </div>
    <div >
        <label >手机</label>

```



```

        <div >
            <input type="number" name="phone" lay-verify="required" lay-reqtext="手机不能为空" placeholder="请输入手机" value="" >
        </div>
    </div>
    <div >
        <label >联系地址</label>
        <div >
            <input type="text" name="address" >
        </div>
    </div>
    <div >
        <label >职业</label>
        <div >
            <input type="email" name="career" >
        </div>
    </div>

    <div >
        <label >备注信息</label>
        <div >
            <textarea name="remark" placeholder="请输入备注信息">
        </textarea>
    </div>
</div>

    <div >
        <div >
            <button lay-submit lay-filter="saveBtn">确认保存</button>
        </div>
    </div>
</div>
<script>
    layui.use(['form', 'table'], function () {
        var form = layui.form,
            layer = layui.layer,
            // table = layui.table,
            $ = layui.$;
        form.render();
        //监听提交
        form.on('submit(saveBtn)', function (data) {
            console.log(data.field);
            //向后台发送数据并进行添加操作

```

```

$.ajax({
    url:"http://localhost:8888/owner/addOwner",
    type:"POST",
    contentType:"application/json",
    data:JSON.stringify(data.field),
    success:function(result){
        //把json对象转string
        // result=JSON.parse(result);
        console.log(result)
        if(result.code==200){
            layer.msg("添加成功",{
                icon:6,
                time:500
            },function(){
                parent.window.location.reload();//重新页面
                var iframeIndex =
parent.layer.getFrameIndex(window.name);
                parent.layer.close(iframeIndex);
            });
        }else{
            layer.msg("添加失败");
        }
    }
});
return false;
});
});
</script>

```

3.3 后台添加功能实现

3.3.1 service接口和实现

```

/**
 * 添加功能
 */
int add(Owner owner);

/**
 * 删除房主信息
 */

```

```
int delete(Long id);

/**
 * 修改
 */
int updateData(Owner owner);
```

实现

```
@Service("ownerService")
public class OwnerServiceImpl extends ServiceImpl<OwnerMapper, Owner>
implements OwnerService{
    @Autowired
    private OwnerMapper ownerDao;
    @Override
    public PageInfo<Owner> findOwnerAll(int page, int limit, Owner owner) {
        //分页
        PageHelper.startPage(page, limit);
        //查询所有的户主信息
        List<Owner> list=ownerDao.queryOwnerAll(owner);
        PageInfo<Owner> pageinfo=new PageInfo<>(list);
        return pageinfo;
    }

    @Override
    public int add(Owner owner) {
        return baseMapper.insert(owner);
    }

    @Override
    public int delete(Long id) {
        return baseMapper.deleteById(id);
    }

    @Override
    public int updateData(Owner owner) {
        return baseMapper.updateById(owner);
    }
}
```

3.3.2 controller实现

```
/**
 * 添加房主信息
 */
@RequestMapping("/addOwner")
@ResponseBody
public R add(@RequestBody Owner owner) {
    int num=ownerService.add(owner);
    if (num>0) {
        return R.ok();
    }
    return R.fail(400,"添加失败");
}
```

3.4 前端请求后台处理

```
form.on('submit(saveBtn)', function (data) {
    console.log(data.field);
    //向后台发送数据并进行添加操作
    $.ajax({
        url:"http://localhost:8888/owner/addOwner",
        type:"POST",
        contentType:"application/json",
        data:JSON.stringify(data.field),
        success:function(result) {
            //把json对象转string
            // result=JSON.parse(result);
            console.log(result)
            if (result.code==200) {
                layer.msg("添加成功", {
                    icon:6,
                    time:500
                },function() {
                    parent.window.location.reload();//重新页面
                    var iframeIndex =
parent.layer.getFrameIndex(window.name);
                    parent.layer.close(iframeIndex);
                });
            }
        }
    });
});
```

```

        }else{
            layer.msg("添加失败");
        }
    }
})
return false;
});

```

3.5 功能演示

姓名: 身份证号码: 高级查询

添加 删除 列表 新增 打印 帮助

<input type="checkbox"/>	ID	姓名	身份...	性别	地址	电话	职业	操作
<input type="checkbox"/>	1	kappy	233232	女	西安	110	程序员	删除
<input type="checkbox"/>	13	kaven	4129...	女	11111	13211111111	chen...	删除
<input type="checkbox"/>	14	王五	4212...	男	2121...	1321839333	ui	删除
<input type="checkbox"/>	16	2222..	2222...	男	2222...	222222222	2222...	删除

< 1 > 到第 1 页 确定 共 4 条 15 条/页

4、房主信息的删除

4.1 后端功能的处理

```

/**
 * 删除功能
 */
@RequestMapping("/deleteOwner")
@ResponseBody
public R delete(String ids) {
    //获取前端传过来的ids集合 "1,2,3,4,5"
    List<String> list=Arrays.asList(ids.split(","));
    //遍历list集合进行删除操作
    for(String id:list){
        Long idLong=Long.parseLong(id);
        ownerService.delete(idLong);
    }
    return R.ok();
}

```

4.2 删除前端处理

4.2.1 监听删除事件

```
/**
 * 获取批量删除选中的id集合
 */
function getCheckId(data) {
    var arr=new Array();
    for(var i=0;i<data.length;i++){
        arr.push(data[i].id);
    }
    return arr.join(",");
};

/**
 * 删除功能的实现
 */
function deleteByIds(ids, index) {
    $.ajax({
        url:"http://localhost:8888/owner/deleteOwner",
        type:"POST",
        // contentType:"application/json",
        // data:JSON.stringify(data.field),
        data:{"ids":ids},
        success:function(result) {
            if(result.code==200) {
                layer.msg("删除成功", {
                    icon:6,
                    time:500
                },function() {
                    parent.window.location.reload();//重新页面
                    // var iframeIndex =
parent.layer.getFrameIndex(window.name);
                    // parent.layer.close(iframeIndex);
                });
            }else{
                layer.msg("删除失败");
            }
        }
    })
}
```

```
    return false;
}
```

监听调用

```
    } else if (obj.event === 'delete') { // 监听删除操作
        var checkStatus = table.checkStatus(obj.config.id)
        , data = checkStatus.data;
        /** 首先判断是否选择信息，如果没有提示 ...*/
        if(data.length==0){
            layer.msg("请选要删除的记录信息");
        }else{
            //获取删除id的集合
            var ids=getCheckId(data);
            layer.confirm("你真的要删除了吗?",function(index){
                deleteByIds(ids,index);
            })
        }
        // layer.alert(JSON.stringify(data));
    }
});
```

5、单个删除功能

```
    } else if (obj.event === 'delete') {
        layer.confirm('真的删除行么', function (index) {
            //调用删除功能即可
            deleteByIds(data.id,index);
            layer.close(index);
        });
    }
}
```

姓名: 身份证号码:

<input type="checkbox"/>	ID	姓名	身份...	性别	地址	电话	职业	操作
<input type="checkbox"/>	1	kappy	233232	女	西安	110	程序员	<input type="button" value="删除"/>

< 1 > 到第 1 页 共 1 条 15 条/页

6、房主信息的修改功能

6.1 controller层功能实现

```
/**
 * 添加房主信息
 */
@RequestMapping("/updateOwner")
@ResponseBody
public R update(@RequestBody Owner owner) {
    int num=ownerService.updateData(owner);
    if (num>0) {
        return R.ok();
    }
    return R.fail(400,"添加失败");
}
```

6.2 监听事件

```
<script type="text/html" id="currentTableBar">
    <a class="layui-btn layui-btn-normal layui-btn-xs data-count-edit" lay-event="edit">编辑</a>
    <a class="layui-btn layui-btn-xs layui-btn-danger data-count-delete" lay-event="delete">删除</a>
</script>
```

```
var data = obj.data;
if (obj.event === 'edit') {
    var content = miniPage.getHrefContent('page/owner/edit.html');
    var openWH = miniPage.getOpenWidthHeight();

    var index = layer.open({
        title: '编辑用户',
        type: 1,
        shade: 0.2,
        maxmin: true,
        shadeClose: true,
        area: [openWH[0] + 'px', openWH[1] + 'px'],
        offset: [openWH[2] + 'px', openWH[3] + 'px'],
        content: content,
    });
    setFormValue(obj, data);
}
```

6.3 设计修改页面

可以复制拷贝添加功能页面


```
<div >

    <div lay-filter="updateSubmit">
        <input type="hidden" name="id" >
        <div >
            <label >房主姓名</label>
            <div >
                <input type="text" name="custname" lay-verify="required" >
            </div>
        </div>

        <div >
            <label >性别</label>
            <div >
                <input type="radio" name="sex" value="男" title="男"
checked="">
                <input type="radio" name="sex" value="女" title="女">
            </div>
        </div>

        <div >
            <label >身份证号</label>
            <div >
                <input type="text" name="identity" lay-verify="required" lay-
reqtext="身份证号不能为空" placeholder="请输入用户名" value="" >
                <tip>填写自己管理账号的名称。</tip>
            </div>
        </div>

        <div >
            <label >手机</label>
            <div >
                <input type="number" name="phone" lay-verify="required" lay-
reqtext="手机不能为空" placeholder="请输入手机" value="" >
            </div>
        </div>

        <div >
            <label >联系地址</label>
            <div >
                <input type="text" name="address" >
            </div>
        </div>

        <div >
            <label >职业</label>
```

```

        <div >
            <input type="text" name="career" >
        </div>
    </div>

    <div >
        <label >备注信息</label>
        <div >
            <textarea name="remarks" placeholder="请输入备注信息">
        </textarea>
        </div>
    </div>

    <div >
        <div >
            <button lay-submit lay-filter="updateSubmit">确认修改
        </button>
        </div>
    </div>
</div>

```

6.4 页面信息的渲染

```

/**
 * 设置form中的值信息
 *
 */
function setFormValue(data) {

    form.val("updateSubmit", {
        "id":data.id,
        "identity":data.identity,
        "custname":data.custname,
        "sex":data.sex,
        "address":data.address,
        "phone":data.phone,
        "career":data.career,
        "remarks":data.remarks
    })
};

```

```
var data = obj.data;
if (obj.event === 'edit') {
    var content = miniPage.getHrefContent('page/owner/edit.html');
    var openWH = miniPage.getOpenWidthHeight();

    var index = layer.open({
        title: '编辑用户',
        type: 1,
        shade: 0.2,
        maxmin: true,
        area: [openWH[0] + 'px', openWH[1] + 'px'],
        offset: [openWH[2] + 'px', openWH[3] + 'px'],
        content: content,
    });
    setFormValue(data);
}
```

6.5 渲染后的效果图

编辑用户

房主姓名 *

kappy1

性别 *

☐ 男 ☒ 女

身份证号 *

233232

填写自己管理账号的名称。

手机 *

110

联系地址 *

西安

职业

程序员

备注信息

飒飒

6.6 修改信息进行提交工作

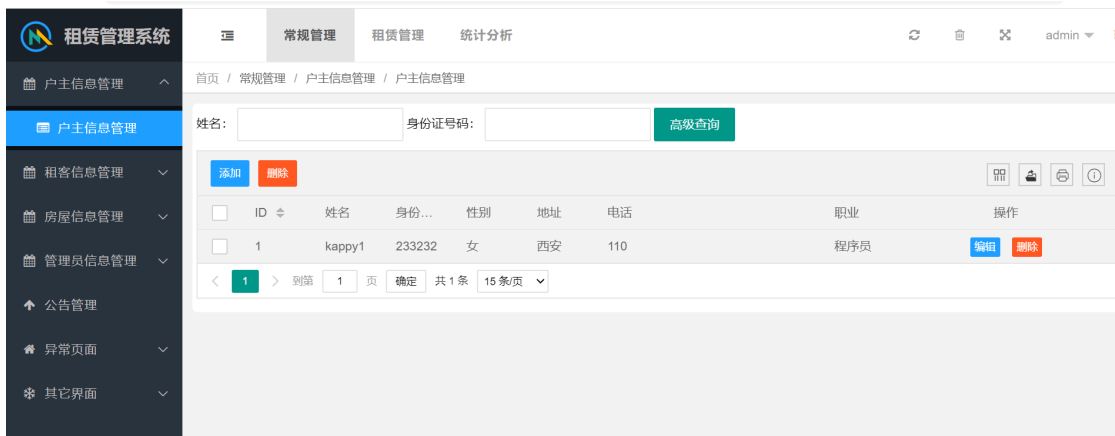
```
//监听提交
form.on('submit(updateSubmit)', function (data) {
    console.log(data.field);
    //向后台发送数据并进行添加操作
    $.ajax({
        url: "http://localhost:8888/owner/updateOwner",
        type: "POST",
        contentType: "application/json",
        data: JSON.stringify(data.field),
    });
});
```

```

        success:function(result){
            //把json对象转string
            // result=JSON.parse(result);
            console.log(result)
            if(result.code==200){
                layer.msg("修改成功",{
                    icon:6,
                    time:500
                },function(){
                    parent.window.location.reload();//重新页面
                    var iframeIndex =
parent.layer.getFrameIndex(window.name);
                    parent.layer.close(iframeIndex);
                });
            }else{
                layer.msg("修改失败");
            }
        }
    })
    return false;
});

```

7、总结



The screenshot displays a web application for a '租赁管理系统' (Leasing Management System). The interface is divided into a sidebar and a main content area. The sidebar contains navigation links for '户主信息管理' (Landlord Information Management), '租客信息管理' (Tenant Information Management), '房屋信息管理' (Property Information Management), '管理信息管理系统' (Management Information Management System), '公告管理' (Announcement Management), '异常页面' (Exception Pages), and '其它界面' (Other Interfaces). The main content area shows a table with columns for 'ID', '姓名' (Name), '身份...' (Identity...), '性别' (Gender), '地址' (Address), '电话' (Phone), and '职业' (Occupation). A single record is visible for 'kappy1' with ID '1' and occupation '程序员' (Programmer). The table also includes a '操作' (Action) column with buttons for '编辑' (Edit) and '删除' (Delete). The interface also features search filters for '姓名' (Name) and '身份证号码' (ID Number), a '高级查询' (Advanced Search) button, and pagination controls showing '1' of '15' pages.