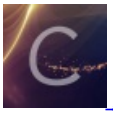
使用ztree通过ajax进行数据的层级获取

原创

封神尘 于 2020-02-18 14:06:56 发布 370 收藏
分类专栏： ztree

本文链接：https://blog.csdn.net/qq_30306501/article/details/104373965


ztree 专栏收录该内容
1 篇文章 0 订阅
订阅专栏

本例使用的是 ztree的 简单数据类型

在使用ztree时，**需要自定义返回数据的类型**，如果需要的话可以直接按本demo的实体进行封装

前台页面：

```
<div>
    <button type="button" onclick="addSuperNode()"  class="layui-btn layui-btn-primary"/> 新增顶级节点</butto
</div>
<div>
    <ul id="treeDemo" class="ztree"></ul>
</div>
```

js文件 推荐直接使用js写函数，而不是进行封包

setting配置可以使用以下配置

```
var zNodes ;       //定义数据节点信息
var setting = {
    async: {
        enable: true,     //开启异步加载功能
        contentType: "application/x-www-form-urlencoded", //传输数据类型
        url: "../xtjoCandidateSetting/selectByPid",        //异步加载获取数据的地址
        autoParam: ["id"], //autoParamFunction, //["Id","name"],
            //点击父节点时，获取子节点时需要提交的参数，可以是多个参数
        dataFilter: function filter(treeId, parentNode, childNodes) {
            //ajax返回数据的预处理 --可以不需要操作
            console.log(treeId);
            console.log(parentNode);
            console.log(childNodes);
            if (!childNodes) return null;
            return childNodes;
        }
    },
    data:{
        key:{
            name:"name"         //使用返回数据的哪一个字段作为树中节点的显示名称
        },
        simpleData: {
            enable: true,
                //开启简单数据类型，下面参数都需要写，同时必须要满足对应的层级关系，
```

```
                //后台返回类型 参考后台代码，要与后台实体对应起来
            idKey: "id",        //当前节点的id
            pIdKey: "pid",      //父节点id
            rootPId: 0          //根节点id
        }
    },
    edit: {
        // enable: true,
                //是否开启 ztree自定义的编辑工具 开启后  renameTitle  removeTitle 才生效
        // renameTitle: setRenameTitle,   //设置 修改名称 图标的提示文字信息 方法
        // removeTitle: setRemoveTitle    //设置 修改删除 图标的提示文字信息 方法
    },
    view: {
        // addDiyDom: addDiyDom,  //添加用户自定义图标方法
        expandSpeed:"normal",     //正常速度的动画
        addHoverDom: addHoverDom, //添加鼠标浮动的自定义图标方法
        removeHoverDom: removeHoverDom,   //移除鼠标浮动的自定义图标方法
        selectedMulti: false              //是否允许选中多个节点，推荐1false
    },
    callback:{ //这个回调参考官方文档，里面很详细
        // onClick:zTreeOnClick        //当用户点击节点时调用的方法
    }
};
$(function(){
    getNodes();   //获取后台数据信息
    $.fn.zTree.init($("#treeDemo"), setting,zNodes);
    //树的初始化  该方法为根方法
    //三个参数的含义 树的dom元素，定义的setting配置，初始时数据的节点信息/为null时表示异步加载
    layui.use('element', function(){
        var element = layui.element;
    });
});

//从后台获取节点数据
function getNodes() {
    $.ajax({
        type: "GET",
        cache:false,
        url: "../xtjoCandidateSetting/selectByPid?Id=0",
        dataType: "json",
        async:false,
        success: function(data){     //获取根节点信息
            console.log(data);
            //[{"id":"1","pId":"0","name":"各市分公司","isParent":true,"open":false},
            // {"id":"2","pId":"0","name":"省公司各部门","isParent":true,"open":false}]
            zNodes = data;
        }
    });
};
// 异步传递参数时调用该方法，可以将当前节点的信息查看
function autoParamFunction(treeId,treeNode){
    console.log(treeNode);
    var re = {
        "Id":treeNode.Id
        // ,"name":treeNode.name
    };
    console.log(re);
    return re;
}
```

```
//浮动添加自定义元素
function addHoverDom(treeId, treeNode) {
    //获得name元素
    var sObj = $("#" + treeNode.tId + "_span");
    if (treeNode.editNameFlag || $("#addBtn_"+treeNode.tId).length>0) return;
    //添加 新增按钮
    var addStr = "<span class='button add' id='addBtn_" + treeNode.tId + "' title='增加子节点' onfocus='this
    sObj.after(addStr);
    var addbtn = $("#addBtn_"+treeNode.tId);
    //给新增按钮添加操作
    if (addbtn) addbtn.bind("click", function(){});
    //添加 编辑按钮
    var editStr = "<span class='button edit' id='editBtn_" + treeNode.tId + "'title='编辑当前节点' ></span>";
    addbtn.after(editStr);
    var editBtn = $("#editBtn_"+treeNode.tId);
    //给编辑按钮添加操作
    if (editBtn) editBtn.bind("click", function(){});
    //添加 删除按钮
    var delStr = "<span class='button remove' id='delBtn_" + treeNode.tId + "' title='删除当前节点' ></span>'
    editBtn.after(delStr);
    var delBtn = $("#delBtn_"+treeNode.tId);
    //给删除按钮添加操作
    if (delBtn) delBtn.bind("click", function () {});
};
//浮动删除自定义元素
function removeHoverDom(treeId, treeNode) {
    $("#addBtn_"+treeNode.tId).unbind().remove();
    $("#editBtn_"+treeNode.tId).unbind().remove();
    $("#delBtn_"+treeNode.tId).unbind().remove();
};
```

在获取到数据库数据后，如果需要转换为前端使用的格式，那么最好将数据进行重新封装，

后台返回数据格式，该实体将与前台的实体进行对应，类名：simpleData

```
/**
*  ztree中的节点实体类
*  @author jizhen
*  @since 2020-02-14 11:24:35
*/
public class SimZtreeNode {

    //当前节点id
    private String id;

    //父节点id
    private String pId;

    //当前节点名称
    private String name;

    //当前节点是否为父节点
    private Boolean isParent;

    //节点的 展开 / 折叠 状态
    private Boolean open;

    ....get/set/toString

}
```

例如我自己的实体类转换

```
    @RequestMapping(value = "/selectByPid",method = RequestMethod.GET)
    @ResponseBody
    public String queryInfoByPidToGet(Model model, HttpServletRequest request,String id) {
        LOGGER.info("--------Get查询，id："+id);
        long s = System.currentTimeMillis();
        List<XtjoCandidateSetting> returnObject = this.xtjoCandidateSettingServiceImpl.queryInfoByPid(id);
        List<SimZtreeNode> returnZtree = setInfoToSimTree(returnObject);
        LOGGER.info("--------通过父id："+id+"查询数据，并转化为数结构，耗时："+(System.currentTimeMillis() - s)+
        return  gson.toJson(returnZtree);
    }
```

最后通过 谷歌的gson转换，返回json类型的数据到前台