https://blog.csdn.net/qq\_35624642/article/details/53931111

**Easyui实现combotree,输入文本及时筛选功能**

2016年12月29日 18:04:10 [QC班长](https://me.csdn.net/qq_35624642) 阅读数 5824

 版权声明：本文为博主原创文章，未经博主允许不得转载。 https://blog.csdn.net/qq\_35624642/article/details/53931111

easyui提供的ComboTree组件没有查找功能，如果结点多的话很不好找,想要的功能是在ComboTree中输入文字就显示出相应关键字的结点，有点像自动补全。

***$***(**"#so\_id"**).combotree({

**url**: **"${path}/sys/user\_tree.do"**,

**onLoadError**: *showError*,

onClick: **function** (node) {

**var** user = {**"refresh"**: **"1"**};

**if** (node.**id** != 0) user.**so\_id** = node.**attributes**.**so\_id**;

***$***(**"#table"**).datagrid(**"clearSelections"**).datagrid(**"load"**, user);

***$***(**"#parentName"**).val(user.**so\_id**);

},

**panelHeight**:400,**editable**: **true**,

onLoadSuccess: **function** (node, data) {

***$***(**this**).tree(**'collapseAll'**); *//关闭树节点*

**var** root = ***$***(**this**).tree(**"getRoot"**); *//展开根节点*

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

***$***(**this**).tree(**"expand"**, root.**target**);

**var** childrens = ***$***(**this**).tree(**"getChildren"**, root.**target**);

**for** (**var** i = 0; i < childrens.**length**; i++) {

**if** (childrens[i].**attributes**.**so\_id** == **'0'**) {

***$***(**this**).tree(**"expand"**, childrens[i].**target**);

}

}

**var** item = ***$***(**"#so\_id"**).tree(**"find"**, ***openid***); *//展开并定位到指定节点*

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

***$***(**this**).tree(**"expandTo"**, item.**target**);

***$***(**this**).tree(**"select"**, item.**target**);

}

**var** user = {**"refresh"**: **"1"**};

user.**so\_id** = ***openid***;

***$***(**"#table"**).datagrid(**"clearSelections"**).datagrid(**"load"**, user);

}

}

});

发现实际输入的input是easyui自动生成的，class样式为combo-text；是这个输入框我们就监听它的值改变事件代码如下:

*//扩展combotree搜索功能*

***$***(**".combo-text"**).bind(**"input propertychange"**, **function**() {

***$***(**'#so\_id'**).combotree(**'tree'**).tree(**"search"**,***$***(**this**).val());

**if**(***$***(**this**).val()==**""** || **null**==***$***(**this**).val()){

***$***(**'#so\_id'**).combotree(**'tree'**).tree(**"expandAll"**);

}

});

把上面的代码加入页面就可以实现树查找了，上面树查找用了一个网上找的树查找扩展代码如下：

(**function**($) {

*//扩展combotree搜索功能*

$.extend($.**fn**.tree.**methods**, {

search: **function**(jqTree, searchText) {

*// easyui tree的tree对象。可以通过tree.methodName(jqTree)方式调用easyui tree的方法*

**var** tree = **this**;

*// 获取所有的树节点*

**var** nodeList = *getAllNodes*(jqTree, tree);

*// 如果没有搜索条件，则展示所有树节点*

searchText = $.trim(searchText);

**if** (searchText == **""**) {

**for** (**var** i=0; i<nodeList.**length**; i++){

$(**".tree-node-targeted"**, nodeList[i].**target**).removeClass(**"tree-node-targeted"**);

$(nodeList[i].**target**).show();

}

*// 展开已选择的节点（如果之前选择了）*

**var** selectedNode = tree.getSelected(jqTree);

**if** (selectedNode) {

tree.expandTo(jqTree, selectedNode.**target**);

}

**return**;

}

*// 搜索匹配的节点并高亮显示*

**var** matchedNodeList = [];

**if** (nodeList && nodeList.**length**>0) {

**var** node = **null**;

**for** (**var** i=0; i<nodeList.**length**; i++){

node = nodeList[i];

**if** (*isMatch*(searchText, node.**text**)) {

matchedNodeList.push(node);

}

}

*// 隐藏所有节点*

**for** (**var** i=0; i<nodeList.**length**; i++){

$(**".tree-node-targeted"**, nodeList[i].**target**).removeClass(**"tree-node-targeted"**);

$(nodeList[i].**target**).hide();

}

*// 折叠所有节点*

tree.collapseAll(jqTree);

*// 展示所有匹配的节点以及父节点*

**for** (**var** i=0; i<matchedNodeList.**length**; i++){

*showMatchedNode*(jqTree, tree, matchedNodeList[i]);

}

}

},

showChildren: **function**(jqTree, node) {

*// easyui tree的tree对象。可以通过tree.methodName(jqTree)方式调用easyui tree的方法*

**var** tree = **this**;

*// 展示子节点*

**if** (!tree.isLeaf(jqTree, node.**target**)) {

**var** children = tree.getChildren(jqTree, node.**target**);

**if** (children && children.**length**>0) {

**for** (**var** i=0; i<nodeList.**length**; i++){

**if** ($(children[i].**target**).is(**":hidden"**)) {

$(children[i].**target**).show();

}

}

}

}

},

scrollTo:**function**(jqTree, param) {

*// easyui tree的tree对象。可以通过tree.methodName(jqTree)方式调用easyui tree的方法*

**var** tree = **this**;

*// 如果node为空，则获取当前选中的node*

**var** targetNode = param && param.targetNode ? param.targetNode : tree.getSelected(jqTree);

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

*// 判断节点是否在可视区域*

**var** root = tree.getRoot(jqTree);

**var** $targetNode = $(targetNode.**target**);

**var** container = param && param.treeContainer ? param.treeContainer : jqTree.parent();

**var** containerH = container.**height**();

**var** nodeOffsetHeight = $targetNode.offset().**top** - container.offset().**top**;

**if** (nodeOffsetHeight > (containerH - 30)) {

**var** scrollHeight = container.**scrollTop**() + nodeOffsetHeight - containerH + 30;

container.**scrollTop**(scrollHeight);

}

}

}

});

**function** *showMatchedNode*(jqTree, tree, node) {

*// 展示所有父节点*

$(node.**target**).show();

$(**".tree-title"**, node.**target**).addClass(**"tree-node-targeted"**);

**var** pNode = node;

**while** ((pNode = tree.getParent(jqTree, pNode.**target**))) {

$(pNode.**target**).show();

}

*// 展开到该节点*

tree.expandTo(jqTree, node.**target**);

*// 如果是非叶子节点，需折叠该节点的所有子节点*

**if** (!tree.isLeaf(jqTree, node.**target**)) {

tree.collapse(jqTree, node.**target**);

}

}

**function** *isMatch*(searchText, targetText) {

**return** $.trim(targetText)!=**""** && targetText.indexOf(searchText)!=-1;

}

**function** *getAllNodes*(jqTree, tree) {

**var** allNodeList = jqTree.**data**(**"allNodeList"**);

**if** (!allNodeList) {

**var** roots = tree.getRoots(jqTree);

allNodeList = *getChildNodeList*(jqTree, tree, roots);

jqTree.**data**(**"allNodeList"**, allNodeList);

}

**return** allNodeList;

}

**function** *getChildNodeList*(jqTree, tree, nodes) {

**var** childNodeList = [];

**if** (nodes && nodes.**length**>0) {

**var** node = **null**;

**for** (**var** i=0; i<nodes.**length**; i++){

node = nodes[i];

childNodeList.push(node);

**if** (!tree.isLeaf(jqTree, node.**target**)) {

**var** children = tree.getChildren(jqTree, node.**target**);

childNodeList = childNodeList.concat(*getChildNodeList*(jqTree, tree, children));

}

}

}

**return** childNodeList;

}

})(jQuery);