# 目录

使用说明 ..... 2
方法 1：初始化 luckysheet 前在 options 中设置 ..... 2
方法 2 初始化 luckysheet 完毕后使用 ..... 2
源码更改范围 ..... 2
Lucksheet ..... 2
src\controllers\menuButton.js ..... 2
src\controllers\updateCell.js ..... 3
src\controllers\cellTreePickerCtrl.js ..... 3
src\global\api.js ..... 8
src\controllers\constant.js ..... 9
src\global\createdom.js ..... 9
MultipleTreeSelect ..... 10
src\lib\tree\js\MultipleTreeSelect.js ..... 10

# 使用说明

## 方法 1：初始化 luckysheet 前在 options 中设置

1. 将 cell.ct.t 的值设置为'trRa'或者'trCh'，trRa 为单选，trCh 为多选，cell.ct.fa 的值请置为空字符串
2. 将 cell.data 的值设置为下拉框的值，JSON 数组,主键 key 为 id，父级 key 为 pId，显示内容 key 为 name
3. 示例："celldata": [{"r":0,"c":0,"v":{"v":'1',"m":"1","ct":{"fa":"","t":"trRa"},"data":[ { id: 1, pId: 0, name: "火之国", open: true }, { id: 11, pId: 1, name: "木叶忍者", open: true }]}}]

## 方法 2 初始化 luckysheet 完毕后使用

1. 手动引入 css/ ztreecss 下 awesomeStyle、metroStyle、zTreeStyle 其中一个文件夹下的 css，必须引入其中一个，不然会导致下拉树显示异常。
2. 初始化 luckysheet。
3. 调用 luckysheet.setTreeDropDownData(sheetIndex,row ,col, type,data)，此方法为异步方法。
   sheetIndex:sheet 页数，从 0 开始，传-1 或者不可用自动取当前页
   Row:单元格所在的行，从 0 开始
   col：单元格所在的列，从 0 开始
   type ： 下拉树类型，有'trRa'、'trCh'两种可选，trRa 为单选，trCh 为多选
   data：下拉树显示的内容，JSON 数组,主键 key 为 id，父级 key 为 pId，显示内容 key 为 name
4. 示例：luckysheet.setTreeDropDownData(0,0, 1, 'trRa', [{ id: 1, pId: 0, name: "火之国", open: true }, { id: 11, pId: 1, name: "木叶忍者", open: true }])。
5. 多选分割符为英文分号。

# 源码更改范围

## Lucksheet

### src\controllers\menuButton.js

menuButton.updateFormatCell
2982 行

```
else if (foucsStatus == 'trRa' || foucsStatus == 'trCh') {
        type = foucsStatus;
}
```

## src\controllers\updateCell.js

luckysheetupdateCell
7 行

```
import cellTreePickerCtrl from './cellTreePickerCtrl';
```

252 行

```
cellTreePickerCtrl.destory()
```

254 行

```
 //日期
    if (d[row_index1][col_index1] && d[row_index1][col_index1].ct) {
        let type = d[row_index1][col_index1].ct.t
        switch (type) {
            case 'd':
                cellDatePickerCtrl.cellFocus(row_index1, col_index1, d[row_index1][col_ind
ex1]);
                break
            case 'trRa':
            case 'trCh':
                cellTreePickerCtrl.cellFocus(row_index1, col_index1, d[row_index1][col_ind
ex1], dataVerificationCtrl);
                break
        }

    }
```

## src\controllers\cellTreePickerCtrl.js

```
import menuButton from './menuButton';
import Store from '../store';
import dataVerificationCtrl from './dataVerificationCtrl'
import '../lib/tree/js/jquery.ztree.all'
import '../lib/tree/js/MultipleTreeSelect'

import { update, datenum_local } from '../global/format';
import { setCellValue, setCellFormat } from '../global/api';
const defaults = {
    textLabel: "jasontext",
    zNodes: undefined,
    height: '100%',
    width: 'auto',
    callback: {
        onCheck: undefined
    },
    data: {
        key: {
```

```
            name: "name",
            isParent: "isParent"
        },
        simpleData: {
            enable: true,
            idKey: "id",
            pIdKey: "pId",
            rootPId: 0
        }
    }
}
var treeobj = null;
const split = ';'
function GetValueToString(treeSelectObj, keyName) {
    let array = GetValue(treeSelectObj, keyName)
    return array.join(split)
}
function GetValue(treeSelectObj, keyName) {
    let array = []
    try {
        if (!treeSelectObj) {
            return array
        }
        let length = treeSelectObj.length
        for (let i = 0; i < length; i++) {
            let item = treeSelectObj[i]
            if (!item[keyName]) {
                continue
            }
            array.push(item[keyName])
        }
        return array
    } catch (e) {
        console.log(e)
    }
    return array
}
function GetItemByName(searchList, keyName, key, returnKey) {
    let array = []
    if (!searchList || !key) {
        return array
    }
    let length = searchList.length
    for (let i = 0; i < length; i++) {
        let item = searchList[i]
        if (item[keyName] == key) {
            if (returnKey) {
```

```
                array.push(item[returnKey])
            } else {
                array.push(obj)
            }

        }
    }
    return array
}
function GetItemByNameMultpleString(searchList, keyName, key, returnKey) {
    let array = GetItemByNameMultple(searchList, keyName, key, returnKey)
    return array.join(split)
}
function GetItemByNameMultple(searchList, keyName, key, returnKey) {
    let array = []
    if (!searchList || !key) {
        return array
    }
    let keyList = key.toString().split(split);
    let length = keyList.length
    for (let i = 0; i < length; i++) {
        let item = keyList[i]
        let obj = GetItemByName(searchList, keyName, item, returnKey)
        if (obj && obj.length > 0) {
            obj.forEach((e) => {
                array.push(e)
            })
        }
    }
    return array
}
let getSheetIndex = function (index) {
    for (let i = 0; i < Store.luckysheetfile.length; i++) {
        if (Store.luckysheetfile[i]["index"] == index) {
            return i;
        }
    }

    return null;
}
const cellTreePickerCtrl = {
    setDropDownData: function (sheetIndex, r, c, type, data) {
        let sheetData = Store.flowdata
        let timeOut = 3000
        let index = setInterval(() => {
            if (!sheetIndex || sheetIndex < 0) {
                sheetIndex = getSheetIndex(Store.currentSheetIndex);
```

```
            }
            let file = Store.luckysheetfile[sheetIndex];
            sheetData = file["data"];
            console.log(sheetData)
            if (sheetData && sheetData.length > 0) {
                clearInterval(index)

                if (!sheetData[r]) {
                    sheetData[r] = []
                }
                if (!sheetData[r][c]) {
                    sheetData[r][c] = {}
                }
                if (!sheetData[r][c]["ct"]) {
                    sheetData[r][c]["ct"] = {}
                }
                sheetData[r][c]["ct"]["fa"] = ''
                sheetData[r][c]["ct"]["t"] = type
                sheetData[r][c].data = data;
            }
        }, 500)

    },
    cellFocus: function (r, c, cell, dataVerificationCtrl) {
        let status = false
        let row = Store.visibledatarow[r],
            row_pre = r == 0 ? 0 : Store.visibledatarow[r - 1];
        let col = Store.visibledatacolumn[c],
            col_pre = c == 0 ? 0 : Store.visibledatacolumn[c - 1];
        let margeset = menuButton.mergeborer(Store.flowdata, r, c);
        let data;
        let temp;
        if (cell.data) {
            data = cell.data
        }
        else if (dataVerificationCtrl && dataVerificationCtrl.dataVerification && dataVeri
ficationCtrl.dataVerification[r + '_' + c]) {
            temp = dataVerificationCtrl.dataVerification[r + '_' + c]
            temp = dataVerificationCtrl.getDropdownList(temp.value1);
            data = []
            temp.forEach((e) => {
                data.push({
                    id: e,
                    pId: 0,
                    name: e
                })
            })
```

```
        } else {
            data = []
        }
        let type = cell.ct.t;
        let value = cell.v
        if (cell.tree && cell.tree.v) {
            value = cell.tree.v
        }
        let option = defaults
        option.zNodes = data
        option.callback.onCheck = function (treeSelectObj, treeNode) {
            if (!status) {
                return
            }
            let ct = cell.ct
            //ct = 'ra'
            let currentValShow = '';
            let currentVal = '';
            if (treeSelectObj.length == 0) {

            } else {
                currentValShow = GetValueToString(treeSelectObj, option.data.key.name);
                currentVal = GetValueToString(treeSelectObj, option.data.simpleData.idKey)
;
            }
            $("#luckysheet-rich-text-editor").html(currentValShow);
            setCellValue(r, c, currentValShow, { isRefresh: false })
            setCellFormat(r, c, 'ct', ct)
            if (!cell.tree) {
                cell.tree = {}
            }
            cell.tree.v = currentVal
        }
        switch (type) {
            case 'trRa':
                option.chkStyle = "radio"
                break
            case 'trCh':
                option.chkStyle = "checkbox"
                break
        }


        if (!!margeset) {
            row = margeset.row[1];
            row_pre = margeset.row[0];
```

```
            col = margeset.column[1];
            col_pre = margeset.column[0];
        }
        let classname = '.menuContent';
        let id = '#luckysheet-input-box';
        var inputObj = $('.luckysheet-cell-input');
        var idOffset = inputObj.offset();
        if ($(classname)[0] && treeobj) {
            treeobj.destory()
            treeobj = null;
        } else {

        }
        treeobj = $(id).drawMultipleTree(option);
        if (value) {
            value = GetItemByNameMultple(data, option.data.key.name, value, option.data.si
mpleData.idKey)
            treeobj.val(value);
        }
        $(id).click();

        $(".menuContent").css({ left: idOffset.left + "px", top: idOffset.top + inputObj.o
uterHeight() + "px" }).slideDown("fast");
        status = true
    }, destory() {
        let classname = '.menuContent';
        if ($(classname)[0] && treeobj) {
            treeobj.destory()
            treeobj = null;
        }
    }
}
export default cellTreePickerCtrl;
```

## src\global\api.js

```
import { common_extend } from '../utils/util';
import {initialWorkBook} from '../core'
import luckysheetConfigsetting from "../controllers/luckysheetConfigsetting";
```

5532 行

```
export function refreshCellRightClickConfig(options = {}) {
    let extendsetting = common_extend(luckysheetConfigsetting.cellRightClickConfig, option
s);
```

```
    luckysheetConfigsetting.cellRightClickConfig = extendsetting
    createRightclickDom();
    initialWorkBook()
    let {
        success
    } = { ...options }

    if (success && typeof success === 'function') {
        success();
    }
}
```

## src\controllers\constant.js

620 行

```
if ($('#' + id)[0]) {
        $('#' + id).empty()
        $('#' + id).append(rightclickContainer);
    } else {
        const rightdom = '<div id="' + id + '" class="luckysheet-cols-menu luckysheet-
rightgclick-menu luckysheet-mousedown-cancel">' +
            rightclickContainer + '</div>';

        $("body").append(rightdom);
    }

    return rightclickContainer;
```

## src\global\createdom.js

22 行

```
export function createRightclickDom() {
    rightclickHTML()
}
```

112 行

```
createRightclickDom();
```

## src\lib\tree\js\MultipleTreeSelect.js

256 行

```
this.all_container = this.$target_element.wrap('<div class="mts-container" style="inline-block;position: relative;"/>').parent();
```

替换为

```
 this.all_container = this.$target_element.wrap('<div class="mts-container" >').parent();
```

133 行添加

```
destory: function () {
            if (this.all_container) {
                this.$target_element.removeData('drawMultipleTree');
                this.$target_element.val('');
                this.$target_element.attr("checks", "")
                this.$target_element.attr("title", "")
                this.all_container.replaceWith(this.$target_element);
                this.all_container = null;
                this.$zTreeObj.destroy();
                if (this.$searchZTreeObj) {
                    this.$searchZTreeObj.destroy();
                }

            }
        },
        val: function (ids) {
            try {
                var nodes = this.$zTreeObj.getCheckedNodes(true);
                var _this = this;

                if (ids == undefined) {
                    ids = [];
                    $(nodes).each(function (index, node) {
                        ids.push(node[_this.options.data.simpleData.idKey]);
                    });
                    return ids;
                } else {
                    /*赋值*/
                    this.options.checks = ids;
                    $(nodes).each(function (index, node) {
                        node.checked = false;
                        _this.$zTreeObj.updateNode(node);
                    });
                    $(ids).each(function (index, id) {
                        var node = _this.$zTreeObj.getNodeByParam("id", id);
```

```
                    node.checked = true;
                    _this.$zTreeObj.updateNode(node);
                });
                _this.m2v();
                return this;
            }
        } catch (e) {
            console.log(e)
        }
        return this;


    }, m2v: function () {
        var nodes = this.$zTreeObj.getCheckedNodes(true);
        var text = "";
        var checks = "";
        if (this.options.renderText) {
            text = this.options.renderText(nodes);
            for (var i = 0, l = nodes.length; i < l; i++) {
                checks += nodes[i][this.options.data.simpleData.idKey] + ",";
            }
        } else {
            for (var i = 0, l = nodes.length; i < l; i++) {
                var texts = [nodes[i][this.options.data.key.name]];
                this.getNodeName(nodes[i], texts);
                text += texts.join("/") + ",";
                checks += nodes[i][this.options.data.simpleData.idKey] + ",";
            }
            if (text.length > 0) {
                text = text.substring(0, text.length - 1);
            }
        }
        if (checks.length > 0) {
            checks = checks.substring(0, checks.length - 1);
        }
        this.$target_element.attr("checks", checks);
        this.$target_element.val(text);
        this.$target_element.attr("title", "编码:" + checks + ";名称:" + text);
    }, getNodeName: function (node, texts) {
        if (this.options.textModel == 'simple') {
            return texts;
        }
        var parent = node.getParentNode();
        if (parent) {
            texts.unshift(parent[this.options.data.key.name]);
            return this.getNodeName(parent, texts);
        } else {
```

```
                return texts;
            }
        }, reloadNode: function (nodes) {
            /*纯前台加载*/
            if (nodes) {
                this.options.zNodes = nodes;
            }
            this.init_ztree = init_ztree.call(this)
        },
```

377 行更改 drawMultipleTree 函数返回值

```
$.fn.drawMultipleTree = function (options) {
        var option = arguments[0], value
        args = arguments;
        var resultArr = [];
        this.each(function () {
            var $this = $(this);
            data = $this.data('drawMultipleTree')
            if (!data) {
                treeSelectObj = new DrawMultipleTree(this, options);
                $.data(this, 'drawMultipleTree', treeSelectObj);
                resultArr.push(treeSelectObj);
            }
            if (typeof option === 'string') {
                value = data[option](args[1]);
            }
        });
        return resultArr.length == 1 ? resultArr[0] : resultArr;
        //return typeof value !== 'undefined' ? value : this;
    };
```