目录

简介

treeTable实现了layui数据表格的大部分功能，并且在用法上与几乎数据表格一致， 支持懒加载、复选框联动(半选)、拖拽列宽、固定表头等功能。

演示地址：https://whvse.gitee.io/treetable-lay/

更新日志

- 2020-04-18 (v3.0)
  - 支持拖拽列宽、表头工具栏toolbar
  - 支持隐藏显示列、打印、导出
  - 支持url、method、where、headers等url方式加载参数，同时兼容之前reqData写法
  - cols使用二维数组，支持同layui表格一样配置复杂表头
  - 复选框半选状态不再受form.render影响
  - 解决ful-xxx高度计算与layui表格有偏差问题
  - 解决minWidth参数无效果问题
  - 解决图标列溢出不显示省略号的问题
  - 对于children形式数据不会用到id属性
- 2019-12-27
  - 增加单元格溢出省略，点击悬浮展开全部(2019-12-27)
  - 解决空数据时刷新后空提示不移除的bug(2019-12-21)
  - 优化固定表头及固定宽度(2019-12-21)
  - 获取选中数据增加可获取非半选的选中数据(2019-12-21)
- 2019-11-18 (v2.0)
  - 重构treeTable，不再基于数据表格table模块
  - 支持懒加载(异步加载)、支持数据渲染
  - 同时支持pid形式数据和children形式数据
  - 无需指定最顶级pid，自动查找
  - 支持复选框联动，支持半选状态
  - 支持折叠状态记忆
  - 支持只刷新某个节点下数据
  - 支持自定义树形图标
  - 支持设置节点勾选、获取勾选节点
  - 支持行单击、双击、单元格单击、双击事件
  - 支持单元格编辑，并且支持校验格式
  - 支持固定表头，支持ful-xxx的写法
  - 支持自定义复杂表头
  - 优化搜索功能，提供更好的搜索体验
- 2018-07-22 (v1.0)
  - 基于数据表格table模板实现树形结构
  - 实现折叠/展开功能

导入模块

最新版只需要一个 `treeTable.js` 即可：

```
layui.config({
    base: '/'
}).use(['treeTable'], function () {
    var treeTable = layui.treeTable;

});
```

如果不会引用先到layui官网查看模块规范介绍。

效果展示

	ID	name	创建时间	状态	操作
1	1	xxx	2019/11/18 10:44:00	正常	修改 删除
2	1_1	xxx	2019/11/18 10:44:00	正常	修改 删除
6	1_2	xxx	2019/11/18 10:44:00	正常	修改 删除
7	1_3	xxx	2019/11/18 10:44:00	正常	修改 删除
8	2	xxx	2019/11/18 10:44:00	正常	修改 删除
12	3	xxx	2019/11/18 10:44:00	正常	修改 删除
16	4	xxx	2019/11/18 10:44:00	正常	修改 删除
17	5	xxx	2019/11/18 10:44:00	正常	修改 删除

快速使用

url方式渲染：

```
<table id="demoTreeTb"></table>

<script>
   layui.use(['treeTable'], function () {
       var $ = layui.jquery;
       var treeTable = layui.treeTable;

       // 渲染树形表格
       var insTb = treeTable.render({
           elem: '#demoTreeTb',
           url: 'json/menus.json',
           tree: {
               iconIndex: 2,           // 折叠图标显示在第几列
               isPidData: true,        // 是否是id、pid形式数据
               idName: 'authorityId',  // id字段名称
               pidName: 'parentId'     // pid字段名称
           },
           cols: [[
               {type: 'numbers'},
               {type: 'checkbox'},
               {field: 'authorityName', title: '菜单名称'},
               {field: 'menuUrl', title: '菜单地址'},
               {field: 'authority', title: '权限标识'}
           ]]
       });

   });
</script>
```

接口json数据格式：

```
{
 "code": 0,
 "msg": "",
 "data": [
   {
    "authorityId": 1,
    "authorityName": "系统管理",
    "menuUrl": null,
    "authority": null,
    "parentId": 0,
    "open": true
   },
   {
    "authorityId": 2,
    "authorityName": "用户管理",
    "menuUrl": "system/user",
    "authority": null,
    "parentId": 1,
    "open": true
   },
   {
    "authorityId": 3,
    "authorityName": "查询用户",
    "menuUrl": "",
    "authority": "user:view",
    "parentId": 2
   },
   {
    "authorityId": 4,
    "authorityName": "添加用户",
    "menuUrl": null,
```

```
      "authority": "user:add",
      "parentId": 2
    }
  ]
}
```

code为0是成功，可以通过parseData参数自定义。

data方式渲染

```
var data = [{
    id: '1',
    name: 'xxx',
    createTime: '2019/11/18 10:44:00',
}, {
    pid: '1',
    id: '1_1',
    name: 'xxx',
    createTime: '2019/11/18 10:44:00'
},{
    id: '2',
    name: 'xxx',
    createTime: '2019/11/18 10:44:00',
},{
    pid: '2',
    id: '2_1',
    name: 'xxx',
    createTime: '2019/11/18 10:44:00'
}];

// 渲染表格
var insTb = treeTable.render({
    elem: '#demoTb1',
    data: data,  // 数据
    tree: {
        iconIndex: 1,    // 折叠图标显示在第几列
        isPidData: true  // 是否是pid形式数据
    },
    cols: [[
        {type: 'numbers'},
        {field: 'id', title: 'ID'},
        {field: 'name', title: 'name'},
        {field: 'createTime', title: '创建时间'}
    ]]
});
```

懒加载方式渲染

```
var insTb = treeTable.render({
    elem: '#demoTb1',
    url: 'json/menus.json',
    request: {pidName: 'pid'}        // 懒加载请求携带的参数名称
    tree: {
        iconIndex: 1,                // 折叠图标显示在第几列
        idName: 'authorityId',       // 自定义id字段的名称
        pidName: 'parentId',         // 自定义标识是否还有子节点的字段名称
        haveChildName: 'haveChild',  // 自定义标识是否还有子节点的字段名称
        isPidData: true              // 是否是pid形式数据
    },
    cols: [[
        {type: 'numbers'},
        {field: 'authorityName', title: '菜单名称'},
        {field: 'menuUrl', title: '菜单地址'},
```

```
        {field: 'authority', title: '权限标识'}
    ]]
});
```

reqData方式渲染

```
var insTb = treeTable.render({
    elem: '#demoTreeTb',
    tree: {
        iconIndex: 2,           // 折叠图标显示在第几列
        isPidData: true,        // 是否是id、pid形式数据
        idName: 'authorityId',  // id字段名称
        pidName: 'parentId'     // pid字段名称
    },
    cols: [[
        {type: 'numbers'},
        {type: 'checkbox'},
        {field: 'authorityName', title: '菜单名称'},
        {field: 'menuUrl', title: '菜单地址'},
        {field: 'authority', title: '权限标识'}
    ]],
    reqData: function(data, callback) {
        // 在这里写ajax请求，通过callback方法回调数据
        $.get('json/menus.json', function (res) {
            if(res.code==0) callback(res.data);
            else callback(res.msg);
        });
    }
});
```

全部参数

基础参数一览表

参数	类型	说明	示例值
elem	String/DOM	指定原始 table 容器的选择器或 DOM	'#demo'
cols	Array	设置表头。值是二维数组	详见表头参数
data	Array	直接赋值数据	[{}, {}, {}, {}, …]
width	Number	设定容器宽度	350
height	Number/String	设定容器高度	300 / 'full-150'
cellMinWidth	Number	定义所有单元格的最小宽度	100
text	Object	自定义文本，如空数据提示等	详见自定义文本
skin	String	设定表格风格	line行边框、row列边框、nob无边框
even	Boolean	开启隔行背景	true/false
size	String	设定表格尺寸	sm 小尺寸、lg 大尺寸
tree	Object	设定树相关参数	详见树相关参数
style	String	设定容器的样式	'margin-top: 0;'
url	String	url方式加载数据	
useAdmin	Boolean	url方式加载数据是否使用admin.ajax方法	默认true
method	String	url方式加载数据的请求方式	默认GET
where	Object	url方式加载数据的请求参数	
contentType	String	url方式加载数据的参数类型	contentType: 'application/json'
headers	Object	url方式加载数据的请求头	
parseData	Function	url方式加载数据自定义格式化数据	
request	Object	url方式加载数据自定义请求参数名称	
reqData	Function	自定义加载数据方式	
toolbar	String	自定义头部工具栏	默认toolbar: 'default'
defaultToolbar	Array	自定义头部工具栏右侧按钮	默认['filter', 'exports', 'print']
done	Function	表格渲染完回调	
title	自定义table大标题，文件导出会用到		

表头参数一览表cols

参数	类型	说明	示例值
field	String	设定字段名	'username'
title	String	设定标题名称	用户名
width	Number	设定列宽，若不填写，则自动分配	150、20%(数字和百分比)
minWidth	Number	单元格的最小宽度	100(数字)
type	String	设定列类型	checkbox复选框、radio单选框、numbers序号列、space空列
edit	String	单元格编辑类型	text（输入框）、number(数字输入框)
style	String	自定义单元格样式	color: red;

class	String	自定义单元格class	class: 'pd-tb-0';
align	String	单元格排列方式	center居中、right居右
templet	String	自定义列模板	详见自定义列模板
toolbar	String	绑定工具条模板	详见行工具事件
singleLine	Boolean	是否单行显示，溢出悬浮展开	true/false
hide	Boolean	是否初始隐藏列	true/false
unresize	Boolean	是否禁止拖拽列宽	true/false
colspan	Number	单元格所占列数，一般用于多级表头	
rowspan	Number	单元格所占行数	

> templet和toolbar用法与数据表格table模块一致

树相关参数一览表tree

参数	类型	说明	示例值
iconIndex	Number	图标列的索引	默认0
onlyIconControl	Boolean	仅允许点击图标折叠	默认false
arrowType	String	箭头类型	可选'arrow2'
getIcon	Function	自定义树形图标	详见自定义树形图标
isPidData	Boolean	是否是pid形式的数据，懒加载方式不需要	默认false
idName	String	设定id的字段名	默认'id'
pidName	String	设定pid的字段名，children形式数据不需要	默认'pid'
childName	String	设定children的字段名，pid形式数据不需要	默认'children'
haveChildName	String	设定是否有子集的字段名，用于懒加载	默认'haveChild'
openName	String	设定是否默认展开的字段名	默认'open'

自定义配置

自定义文本text

目前只支持自定义空数据提示：

```
treeTable.render({
  text: {
    none: '<div style="padding: 18px 0;">哎呀，一条数据都没有~</div>'
  }
});
```

自定义表头

treeTable实现复杂表头的做法与table模块不同，方法如下：

```
treeTable.render({
  getThead: function() {
    return '<tr><td colspan="6">我是表头</td></tr>';
  }
});
```

就是这么简单粗暴，直接返回表头的html即可，不用在cols里面纠结半天，所以treeTable的cols是一维数组。

自定义树形图标

内置了两种风格图标，你也可以很灵活的自己扩展风格：

```
treeTable.render({
  tree: {
    arrowType: 'arrow2',  // 自定义箭头风格
    getIcon: function(d) {  // 自定义图标
      // d是当前行的数据
      if (d.haveChild) {  // 判断是否有子集
        return '<i class="ew-tree-icon ew-tree-icon-folder"></i>';
      } else {
        return '<i class="ew-tree-icon ew-tree-icon-file"></i>';
      }
    }
  }
});
```

ew-tree-icon-folder(文件夹)和ew-tree-icon-file(文件)这两个class是treeTable内置的， 你可以换成其他clss自己加样式，ew-tree-icon这个class是必须的。

> 判断是否有子集d.haveChild这个写法根据你的实际情况写，如果是children形式数据可以写(d.children&&d.children.length>0)

事件监听

监听工具条点击事件：

```
treeTable.on('tool(test)', function(obj){
    var data = obj.data;  // 获得当前行数据
    var event = obj.event; // 获得lay-event对应的值
    obj.del(); // 删除对应行，并更新缓存
    // 同步更新缓存对应的值
    obj.update({
        username: '123',
        title: 'xxx'
    });
});
```

监听复选框选择：

```
treeTable.on('checkbox(test)', function(obj){
    console.log(obj.checked);  // 当前是否选中状态
    console.log(obj.data);  // 选中行的相关数据
    console.log(obj.type);  // 如果触发的是全选，则为：all，如果触发的是单选，则为：one
});
```

监听单元格编辑：

```
treeTable.on('edit(test)', function(obj){
    console.log(obj.value); //得到修改后的值
    console.log(obj.field); //当前编辑的字段名
    console.log(obj.data); //所在行的所有相关数据
});
```

监听行单双击事件：

```
// 监听行单击事件
treeTable.on('row(test)', function(obj){
    console.log(obj.tr) //得到当前行元素对象
    console.log(obj.data) //得到当前行数据
    // obj.del(); // 删除当前行
    // obj.update(fields) // 修改当前行数据
});

// 监听行双击事件
treeTable.on('rowDouble(test)', function(obj){
    // obj 同上
});
```

监听单元格单双击事件：

```
// 监听行单击事件
treeTable.on('cell(test)', function(obj){
    console.log(obj.field); //当前单元格的字段名
    console.log(obj.data) // 得到当前行数据
});

// 监听行双击事件
treeTable.on('cellDouble(test)', function(obj){
    // obj 同上
});
```

懒加载、默认展开

懒加载数据

实现reqData参数即可实现懒加载数据：

```
treeTable.render({
    elem: '#demoTable',
    reqData: function(data, callback) {
        // data是当前行的数据，通过callback回调子集数据
        callback([]);
    },
    tree: {
        iconIndex: 2,  // 折叠图标显示在第几列
        idName: 'id',  // 自定义id字段的名称
        haveChildName: 'haveChild'  // 自定义标识是否还有子节点的字段名称
    }
});
```

你可以在reqData里面写ajax请求：

```
treeTable.render({
    reqData: function(data, callback) {
        var pid = data?data.id:'';
        $.get('list.json?pid='+pid, function (res) {
            callback(res.data);
        });
    }
});
```

json数据参考格式：

```
{"code": 200, "data": [{"id": "1", "name": "xxx", "haveChild": true}]}
```

通过haveChild标识是否还有子节点，id也是必须的字段，这两个字段的名字可以在tree参数里面自定义。

> reqData这个方法里面也可以一次性把所有数据都返回，也可以懒加载。

默认展开

通过在数据里面增加open字段来控制是否默认展开：

```
var data = [{
    id: '1',
    name: 'xxx',
    open: true,  // 默认展开
    children: [ {
        id: '1_1',
        name: 'xxx',
        createTime: '2019/11/18 10:44:00'
    }]
}, {
    id: '2',
    name: 'xxx',
    open: true,  // 默认展开
    children: [{
        id: '2_1',
        name: 'xxx',
        state: 0,
        createTime: '2019/11/18 10:44:00',
        children: []
    }]
}];
```

字段名字也可以自定义：

```
treeTable.render({
    tree: {
        openName: 'open',  // 自定义默认展开的字段名
    }
});
```

实例方法

```
var insTb = treeTable.render({ });

// 刷新
insTb.reload(options);  // 重载表格
insTb.refresh();  // 刷新(异步模式)
insTb.refresh(data);  // 刷新(数据模式)
insTb.refresh(id);  // 刷新指定节点下的数据(异步模式)
insTb.refresh(id, data);  // 刷新指定节点下的数据(数据模式)

// 复选框
insTb.checkStatus();  // 获取选中数据(是否是半选会有一个isIndeterminate字段标识)
insTb.checkStatus(false);  // 获取选中数据，不要半选状态
insTb.setChecked(['1','2']);  // 设置选中数据
insTb.removeAllChecked();  // 移除全部选中

// 折叠/展开
insTb.expand(id);  // 展开指定节点
insTb.fold(id);  // 折叠指定节点
insTb.expandAll();  // 展开全部
insTb.foldAll();  // 折叠全部

// 搜索
insTb.filterData('keywords');  // 根据关键字模糊搜索
insTb.filterData(['1','2']);  // 根据id搜索
insTb.clearFilter();  // 清除搜索

// 更新数据(只更新数据，不更新界面)
insTb.del(id);  // 根据id删除
insTb.update(id, fields);  // 根据id更新
```

其他方法

```
// pid形式数据转children形式数据
treeTable.pidToChildren(data, idName, pidName, childName);
```

常见问题

1、单元格内下拉框下拉后显示不出来：

```
treeTable.render({
    cols: [
        {templet: '#demoTreeTableSel', title: '部门', singleLine: false, class: 'pd-tb-0'},
    ]
});
```

列参数增加 `singleLine:false` 关闭单行显示超出省略功能即可。

2、单元格内连续的字符串或数字不能自动换行：

```
treeTable.render({
    cols: [
        {field: 'name', title: 'name', singleLine: false, class: 'break-all'}
    ]
});
```

如果你设置了 `singleLine:false` 关闭单行显示，像自动换行，然而当单元格内容为连续的数字和字符串时，却又无法自动换行， 只需要添加 `class: 'break-all'` 参数即可，这个参数会给单元格增加一个 word-break: break-all 的样式。

3.没有子级、各种报错：
`检查id、pid这些字段是否与默认一致，如果不一致需要配置`

```
treeTable.render({
    tree: {
        idName: 'id',
        pidName: 'pid',
        childName: 'children'
    },
});
```