链滴

用 Java 组装树形 List 数据

作者：luomuren

原文链接：https://ld246.com/article/1647999819745

来源网站：链滴

1. 节点 model 类

```
package com.huiyou.yzf.tree;

import java.util.List;

public class TreeNode {

    private String id;  //编号（不一定是主键）

    private String parentId;  //父级编号

    private String text;  //显示名称

    private String state;//combotree 设置为closed 则默认此节点不展开

    private List<TreeNode> children;

    public TreeNode(String id, String text, String parentId) {
        this.id = id;
        this.parentId = parentId;
        this.text = text;
    }
    public TreeNode(String id, String text, TreeNode parent) {
        this.id = id;
        this.parentId = parent.getId();
        this.text = text;
    }
    public TreeNode(String id, String text, String parentId,String state) {
        this.id = id;
        this.parentId = parentId;
        this.text = text;
        this.state = state;
    }

    public String getParentId() {
        return parentId;
    }

    public void setParentId(String parentId) {
        this.parentId = parentId;
    }

    public String getText() {
        return text;
    }

    public void setText(String text) {
        this.text = text;
    }

    public String getId() {
        return id;
```

```
    }

    public void setId(String id) {
        this.id = id;
    }

    public List<TreeNode> getChildren() {
        return children;
    }

    public void setChildren(List<TreeNode> children) {
        this.children = children;
    }

    public String getState() {
        return state;
    }
    public void setState(String state) {
        this.state = state;
    }
    @Override
    public String toString() {
        return "TreeNode{" +
                "id='" + id + '\'' +
                ", parentId='" + parentId + '\'' +
                ", text='" + text + '\'' +
                ", state='" + state + '\''+
                ", children=" + children +
                '}';
    }
}
```

2.树形List构造器（两种方式：1 循环 2 递归）---配有测试main方法

```
package com.huiyou.yzf.tree;

import java.util.ArrayList;
import java.util.List;

public class TreeBuider {

    /**
     * 两层循环实现建树
     * @param treeNodes 传入的树节点列表
     * @return
     */
    public static List<TreeNode> bulid(List<TreeNode> treeNodes) {

        List<TreeNode> trees = new ArrayList<TreeNode>();

        for (TreeNode treeNode : treeNodes) {

            if ("0".equals(treeNode.getParentId())) {
                trees.add(treeNode);
```

```
        }

        for (TreeNode it : treeNodes) {
            if (it.getParentId().equals(treeNode.getId())) {
                if (treeNode.getChildren() == null) {
                    treeNode.setChildren(new ArrayList<TreeNode>());
                }
                treeNode.getChildren().add(it);
            }
        }
    }
    return trees;
}

/**
 * 使用递归方法建树
 * @param treeNodes
 * @return
 */
public static List<TreeNode> buildByRecursive(List<TreeNode> treeNodes) {
    List<TreeNode> trees = new ArrayList<TreeNode>();
    for (TreeNode treeNode : treeNodes) {
        if ("0".equals(treeNode.getParentId())) {
            trees.add(findChildren(treeNode,treeNodes));
        }
    }
    return trees;
}

/**
 * 递归查找子节点
 * @param treeNodes
 * @return
 */
public static TreeNode findChildren(TreeNode treeNode,List<TreeNode> treeNodes) {
    for (TreeNode it : treeNodes) {
        if(treeNode.getId().equals(it.getParentId())) {
            if (treeNode.getChildren() == null) {
                treeNode.setChildren(new ArrayList<TreeNode>());
            }
            treeNode.getChildren().add(findChildren(it,treeNodes));
        }
    }
    return treeNode;
}



public static void main(String[] args) {

    TreeNode treeNode1 = new TreeNode("1","广州","0");
    TreeNode treeNode2 = new TreeNode("2","深圳","0");

    TreeNode treeNode3 = new TreeNode("3","天河区",treeNode1);
```

```
        TreeNode treeNode4 = new TreeNode("4","越秀区",treeNode1);
        TreeNode treeNode5 = new TreeNode("5","黄埔区",treeNode1);
        TreeNode treeNode6 = new TreeNode("6","石牌",treeNode3);
        TreeNode treeNode7 = new TreeNode("7","百脑汇",treeNode6);

        TreeNode treeNode8 = new TreeNode("8","南山区",treeNode2);
        TreeNode treeNode9 = new TreeNode("9","宝安区",treeNode2);
        TreeNode treeNode10 = new TreeNode("10","科技园",treeNode8);


        List<TreeNode> list = new ArrayList<TreeNode>();

        list.add(treeNode1);
        list.add(treeNode2);
        list.add(treeNode3);
        list.add(treeNode4);
        list.add(treeNode5);
        list.add(treeNode6);
        list.add(treeNode7);
        list.add(treeNode8);
        list.add(treeNode9);
        list.add(treeNode10);

        List<TreeNode> trees = bulid(list);
        System.out.println(trees);
        List<TreeNode> trees_ = buildByRecursive(list);
        System.out.println(trees_);

    }
}
```

节点model类中属性 可根据实际情况进行改变，当list存在多级关系时，常会在Java组装成树形Json据返回，将数据与节点model对应，用上述构造器 中任一方法都可实现 多级list数据组装，然后进后续处理。

以前借鉴某位博主的，地址已找不到，最近又用到 ，在此记录一下，主要供日后使用。

原文链接：https://blog.csdn.net/ZaberyJava/article/details/81914857