链滴

Java list 数据类型组成 tree

作者：mnizht
原文链接：https://ld246.com/article/1575446238378
来源网站：链滴
许可协议：署名-相同方式共享 4.0 国际 (CC BY-SA 4.0)

Java list数据类型组成tree

项目中企业组织架构类数据通常需要返回给前端树形结构的数据

目前项目中的企业机构层级关系有 sys_tenant_org表中的org_num字段确定。每个企业在创建时都创建一个同名的根机构，org_num分配为00，之后再创建的子机构或部门，org_num都会是以00开。例:

根机构(00)

- 子机构(0001)
  - 子机构(000101)
  - 子部门(000102)
    - 子部门(00010201)
- 子部门(0002)
  - 子部门(000201)

目前设计同一上级下的同级机构/部门最多99个(**如果是两位十进制数字，最大就是99，如需扩展，可选用更高的进制**)

通过一个org_num字段，可以找到一个机构/部门的上级，算出机构/部门深度。

所以我们现在以org_num字段为基础，设计对象使得可以将查询出的平级数据构造成树形数据对象。务数据可以自己定义对象并继承OrgNodeDTO.

```
@Getter
@Setter
@Accessors(chain = true)
public class OrgNodeDTO implements Serializable {
  private String orgNum; //机构编码
  private List<? super OrgNodeDTO> children; //子机构

  public OrgNodeDTO(String orgNum) {
    this.orgNum = orgNum;
    this.children = new ArrayList<>();
  }
}
```

list 转tree的utils

```
  /**
   * 节点转换为树形结构
   * 依据对象中orgNum字段分级
   *
   * @param parent    需要返回的父节点，通常为根节点，也可以是层级中某一层的节点
   * @param treeNodes 要转换的子节点,会将属于parent下的子节点组装到parent下
   * @return 返回一个包含了下属节点的节点
   */
```

```
 public static <T extends OrgNodeDTO> T findChildren(T parent, List<T> treeNodes) {
   for (T child : treeNodes) {
     if (parent.getOrgNum().equals(child.getOrgNum().substring(0, child.getOrgNum().length()
2))) {
       if (parent.getChildren() == null) {
         parent.setChildren(new ArrayList<>());
       }
       parent.getChildren().add(findChildren(child, treeNodes));
     }
   }
   return parent;
 }
```

此处的转换方法用了递归，还有一种双层for循环的方法

```
public static List<TreeNode> toTree01(List<TreeNode> treeNodes) {
   List<TreeNode> retList = new ArrayList<>();

   for (TreeNode parent : treeNodes) {
     if ("0".equals(parent.getParentId())) {
       retList.add(parent);
     }
     for (TreeNode child : treeNodes) {
       if (child.getParentId().equals(parent.getId())) {
         if (parent.getChildren() == null) {
           parent.setChildren(new ArrayList<>());
         }
         parent.getChildren().add(child);
       }
     }
   }
   return retList;
 }
```

关于两种方法的效率，虽然说递归会在重复调用函数时会增加耗时和对内存的占用，但考虑到这个功要处理的元素个数通常不会太多，这里做了10000个元素做测试；

测试中发现节点数相同时，最终结果的层级对耗时也是有影响的，可以看到，在数据总量不变的前提，leaf(叶子)越多则最终层级越少，大概在四层或五层时两种方法的执行效率达到最高。而且两种方跑起来占用的系统内存基本都在60-80MB之间波动（这里直接看的系统任务管理器，直接看的当前行的程序，这里面也包含了构造数据那部分代码占用的内存，实际只处理数据也用不了这么多内存）最终项目中选用了递归的方法；

（ps 本来想做满树来测试，但是满树会因为层级的变化导致元素总数不断增加，这样变量又增多了所以最终采用了固定元素个数的测试）

测试代码：https://github.com/mnizht/JDKTest/blob/master/src/cn/zhuht/jdk8test/util/List2Tre.java

结果对比：

特别感谢同事zsp的技术支持：https://gitee.com/fantasyzsp/AuthorityService.git

原文链接：Java list 数据类型组成 tree