Python+Xml +操作+增删改查

python Xml 增删改查

由于小编的系统需要进程间通信，想通过对Xml文件操作，来进行信息交互，于是写了一组相关的类。

xml文件：

<flags>

<flag name="execute" value="false" />

<flag name="other" value="value" />

flags>

在写增删改差前，先封装了一组Xml方法

#!usr/bin/env python

#coding:utf-8

\_\_author\_\_='fangtao'

from xml.etree.ElementTree import ElementTree,Element

class XmlDao():

@staticmethod

def openXml(filename):

tree = ElementTree()

tree.parse(filename)

return tree

@staticmethod

def saveAs(tree,outfile):

tree.write(outfile, encoding="utf-8",xml\_declaration=True)

@staticmethod

def add\_child\_node(nodelist, element):

'''给一个节点添加子节点

nodelist: 节点列表

element: 子节点'''

print len(nodelist)

print element

for node in nodelist:

node.append(element)

@staticmethod

def del\_node\_by\_tagkeyvalue(nodelist, tag, kv\_map):

'''同过属性及属性值定位一个节点，并删除之

nodelist: 父节点列表

tag:子节点标签

kv\_map: 属性及属性值列表'''

for parent\_node in nodelist:

children = parent\_node.getchildren()

for child in children:

if child.tag == tag and XmlDao.if\_match(child, kv\_map):

parent\_node.remove(child)

@staticmethod

def create\_node(tag, property\_map, content=''):

'''新造一个节点

tag:节点标签

property\_map:属性及属性值map

content: 节点闭合标签里的文本内容

return 新节点'''

element = Element(tag, property\_map)

element.text = content

return element

@staticmethod

def change\_node\_text(nodelist, text, is\_add=False, is\_delete=False):

'''改变/增加/删除一个节点的文本

nodelist:节点列表

text : 更新后的文本'''

for node in nodelist:

if is\_add:

node.text += text

elif is\_delete:

node.text = ""

else:

node.text = text

@staticmethod

def change\_node\_properties(nodelist, kv\_map, is\_delete=False):

'''修改/增加 /删除 节点的属性及属性值

nodelist: 节点列表

kv\_map:属性及属性值map'''

for node in nodelist:

for key in kv\_map:

if is\_delete:

if key in node.attrib:

del node.attrib[key]

else:

node.set(key, kv\_map.get(key))

@staticmethod

def get\_node\_by\_keyvalue(nodelist, kv\_map):

'''根据属性及属性值定位符合的节点，返回节点

nodelist: 节点列表

kv\_map: 匹配属性及属性值map'''

result\_nodes = []

for node in nodelist:

if XmlDao.if\_match(node, kv\_map):

result\_nodes.append(node)

return result\_nodes

@staticmethod

def find\_nodes(tree, path):

'''查找某个路径匹配的所有节点

tree: xml树

path: 节点路径'''

return tree.findall(path)

@staticmethod

def if\_match(node, kv\_map):

'''判断某个节点是否包含所有传入参数属性

node: 节点

kv\_map: 属性及属性值组成的map'''

for key in kv\_map:

if node.get(key) != kv\_map.get(key):

return False

return True

如果有自己的需要，也可以自己拿着类库直接使用，然后，写了一组API才做上面的Xml文件。

#!usr/bin/env python

#coding:utf-8

import XmlDao

class FlagDao():

def \_\_init\_\_(self,filename=None):

if filename is None:

self.\_\_filename = '../grapdata/flag.xml'

else:

self.\_\_filename = filename

#获取节点属性

def getValueByName(self,name):

tree = XmlDao.openXml(self.\_\_filename)

print 'tree',tree,self.\_\_filename

if tree is None:

return None

nodes = XmlDao.find\_nodes(tree, 'flag')

nodes = XmlDao.get\_node\_by\_keyvalue(nodes, {'name':name})

if len(nodes) > 0:

return nodes[0].attrib['value']

return None

#设置节点

def setValueByName(self,name,value):

tree = XmlDao.openXml(self.\_\_filename)

if tree is None:

return None

nodes = XmlDao.find\_nodes(tree, 'flag')

nodes = XmlDao.get\_node\_by\_keyvalue(nodes, {'name':name})

if len(nodes) > 0:

nodes[0].attrib['value'] = value

XmlDao.saveAs(tree, self.\_\_filename)

#添加节点

def addTag(self,name,value):

tree = XmlDao.openXml(self.\_\_filename)

XmlDao.add\_child\_node([tree.getroot()],XmlDao.create\_node('flag', {'name':name,'value':value}))

XmlDao.saveAs(tree, self.\_\_filename)

#删除节点

def deleteTagByName(self,name):

tree = XmlDao.openXml(self.\_\_filename)

XmlDao.del\_node\_by\_tagkeyvalue([tree.getroot()], 'flag', {'name':name})

XmlDao.saveAs(tree, self.\_\_filename)

然后写了一组案列，

import FlagDao

flagDao = FlagDao()

#查找节点

print flagDao.getValueByName('execute')

#修改节点

flagDao.setValueByName('execute', 'false')

#打印修改的节点

print flagDao.getValueByName('execute')

#添加节点

flagDao.addTag('zhangcan', 'bendan')

#删除节点

flagDao.deleteTagByName('other')

运行之后，结果为

<flags>

<flag name="execute" value="false" />

<flag name="zhangcan" value="bendan" />flags>