1. **Python的XML解析**

**第一部分**

**一、什么是xml？**(eXtensible Markup Language)

xml即可扩展标记语言,它可以用来标记数据、定义数据类型，是一种允许用户对自己的标记语言进行定义的源语言。<?xml version="1.0" encoding="utf-8"?>是其文档说明。

一个XML文档的例子：

<?xml version="1.0" encoding="UTF-8"?>

<students>

<student sid="1">

<id>1</id>

<name>陈丽水</name>

<sex>女</sex>

<address>北京昌平</address>

<age>18</age>

</student>

<student sid="2">

<id>2</id>

<name>方军</name>

<sex>男</sex>

<address>北京海淀</address>

<age>20</age>

</student>

<student sid="3">

<id>3</id>

<name>李华</name>

<sex>女</sex>

<address>北京通州</address>

<age>19</age>

</student>

<student sid="4">

<id>4</id>

<name>淮海</name>

<sex>男</sex>

<address>北京顺义</address>

<age>22</age>

</student>

<student sid="5">

<id>5</id>

<name>成功</name>

<sex>男</sex>

<address>北京海淀</address>

<age>18</age>

</student>

</students>

它很像我们常见的HTML超文本标记语言。但他们被设计的目的是不同的，超文本标记语言被设计用来显示数据，其焦点是数据的外观。它被设计用来传输和存储数据，其焦点是数据的内容。

那么它有如下特征：

首先，它是有标签对组成，<aa></aa>

标签可以有属性：<aa id='123'></aa>

标签对可以嵌入数据：<aa>abc</aa>

标签可以嵌入子标签（具有层级关系）：

**二、使用xml.dom解析xml**

文件对象模型（Document Object Model，简称DOM），是W3C组织推荐的处理可扩展置标语言的标准编程接口。一个 DOM 的解析器在解析一个 XML 文档时，一次性读取整个文档，把文档中所有元素

保存在内存中的一个树结构里，之后你可以利用DOM 提供的不同的函数来读取或修改文档的内容和结构，也可以把修改过的内容写入xml文件。python中用xml.dom.minidom来解析xml文件。

mxl.dom.minidom 模块被用来处理xml文件，所以要先引入。xml.dom.minidom.parse() 用于打开一个xml文件，并将这个文件对象xmldom变量。

xmldom.documentElement 用于得到dom对象的根节点，并把获得的对象给root，每一个结点都有它的nodeName，nodeValue，nodeType属性。nodeName为结点名字。nodeValue是结点的值，只对文本结点有效。nodeType是结点的类型，[nodeType的12种类型](http://www.cnblogs.com/gongshunkai/p/5839148.html)：

const unsigned short ELEMENT\_NODE = 1; 元素节点

const unsigned short ATTRIBUTE\_NODE = 2; 属性节点

const unsigned short TEXT\_NODE = 3; 文本节点

const unsigned short CDATA\_SECTION\_NODE = 4; CDATA 区段

const unsigned short ENTITY\_REFERENCE\_NODE = 5; 实体引用元素

const unsigned short ENTITY\_NODE = 6; 实体

const unsigned short PROCESSING\_INSTRUCTION\_NODE = 7; 表示处理指令

const unsigned short COMMENT\_NODE = 8; 注释节点

const unsigned short DOCUMENT\_NODE = 9; 最外层的Root element,包括所有其它节点

const unsigned short DOCUMENT\_TYPE\_NODE = 10; <!DOCTYPE………..>

const unsigned short DOCUMENT\_FRAGMENT\_NODE = 11; 文档碎片节点

const unsigned short NOTATION\_NODE = 12; DTD 中声明的符号节点

**实例1：获取要解析的文档对象和根节点**

#encoding=utf-8

import xml.dom.minidom

DomParser=xml.dom.minidom

#获取要解析的文档

xmldom=DomParser.parse("Students.xml")

#获取要解析的文档的根节点

root=xmldom.documentElement

#根节点的名字

print(root.nodeName)

#根节点的值

print(root.nodeValue)

#根节点的类型

print(root.nodeType)

#根节点下的所有的子节点，包括换行节点

print(root.childNodes)

print(len(root.childNodes))

#根节点下的所有的student子节点

print(root.getElementsByTagName("student").length)

执行结果：

students

None

1

[<DOM Text node "'\n'">, <DOM Element: student at 0x2fd1710>, <DOM Text node "'\n'">, <DOM Element: student at 0x2fd18f0>, <DOM Text node "'\n'">, <DOM Element: student at 0x2fd1ad0>, <DOM Text node "'\n'">, <DOM Element: student at 0x29d98a0>, <DOM Text node "'\n'">, <DOM Element: student at 0x2fd1df0>, <DOM Text node "'\n'">]

11

5

**第二部分**

**实例2、解析节点的属性值**

1、Element.hasAttribute(name)

Returns true if the element has an attribute named by name.

2、Element.getAttribute(name)

Return the value of the attribute named by name as a string. If no such attribute exists, an empty string is returned, as if the attribute had no value.

3、Element.setAttribute(name, value)

Set an attribute value from a string.

#encoding=utf-8

import xml.dom.minidom

DomParser=xml.dom.minidom

#获取要解析的文档

xmldom=DomParser.parse("Students.xml")

#获取要解析的文档的根节点

root=xmldom.documentElement

studentsTags=root.getElementsByTagName("student")

for s in studentsTags:

#获取节点sid的属性值

print(s.getAttribute("sid"))

执行的结果：

1

2

3

4

5

**实例3 解析所有的已知子节点名的子节点**

使用父节点.getElementsByTagName(子节点名)的方法可以获取父节点下所有的已知子节点名的子节点，该返回值是一个列表，可以通过下标值来访问。

纯文本是该子节点下的第一个节点，可以使用data属性获取该值。

#encoding=utf-8

import xml.dom.minidom

DomParser=xml.dom.minidom

#获取要解析的文档

xmldom=DomParser.parse("Students.xml")

#获取要解析的文档的根节点

root=xmldom.documentElement

studentsTags=root.getElementsByTagName("student")

for s in studentsTags:

#获取节点s的下所有子节点

print(s.getElementsByTagName("id")[0].firstChild.data)

print(s.getElementsByTagName("name")[0].firstChild.data)

print(s.getElementsByTagName("sex")[0].firstChild.data)

print(s.getElementsByTagName("address")[0].firstChild.data)

print(s.getElementsByTagName("age")[0].firstChild.data)

print("########")

执行结果：

1

陈丽水

女

北京昌平

18

########

2

方军

男

北京海淀

20

########

3

李华

女

北京通州

19

########

4

淮海

男

北京顺义

22

########

5

成功

男

北京海淀

18

########

**第三部分**

**实例4、解析未知子节点名的子节点**

使用if语句判断是元素节点即可， if nodechild.nodeType==1:

#encoding=utf-8

import xml.dom.minidom

DomParser=xml.dom.minidom

#获取要解析的文档

xmldom=DomParser.parse("Students.xml")

#获取要解析的文档的根节点

root=xmldom.documentElement

studentsTags=root.getElementsByTagName("student")

for s in studentsTags:

#解析未知子节点

for nodechild in s.childNodes:

#节点类型是元素节点

if nodechild.nodeType==1:

print("<%s>%s</%s> " %(nodechild.nodeName,nodechild.firstChild.data,nodechild.nodeName))

print()

执行结果：

<id>1</id>

<name>陈丽水</name>

<sex>女</sex>

<address>北京昌平</address>

<age>18</age>

<id>2</id>

<name>方军</name>

<sex>男</sex>

<address>北京海淀</address>

<age>20</age>

<id>3</id>

<name>李华</name>

<sex>女</sex>

<address>北京通州</address>

<age>19</age>

<id>4</id>

<name>淮海</name>

<sex>男</sex>

<address>北京顺义</address>

<age>22</age>

<id>5</id>

<name>成功</name>

<sex>男</sex>

<address>北京海淀</address>

<age>18</age>

**第四部分**

**实例五、在XML文档中添加一个文档，并保存**

1、Document.createElement(tagName)

Create and return a new element node. The element is not inserted into the document when it is created. You need to explicitly insert it with one of the other methods such as insertBefore() or appendChild()

2、Node.appendChild(newChild)

Add a new child node to this node at the end of the list of children, returning newChild. If the node was already in the tree, it is removed first.

3、Node.insertBefore(newChild, refChild)

Insert a new child node before an existing child. It must be the case that refChild is a child of this node; if not, [ValueError](https://docs.python.org/3/library/exceptions.html#ValueError) is raised.newChild is returned. If refChild is None, it inserts newChild at the end of the children’s list.

4、Node.removeChild(oldChild)

Remove a child node. oldChild must be a child of this node; if not, [ValueError](https://docs.python.org/3/library/exceptions.html#ValueError) is raised. oldChild is returned on success. If oldChild will not be used further, its unlink() method should be called.

5、Node.replaceChild(newChild, oldChild)

Replace an existing node with a new node. It must be the case that oldChild is a child of this node; if not, [ValueError](https://docs.python.org/3/library/exceptions.html#ValueError) is raised.

开发实例：

#encoding=utf-8

import xml.dom.minidom

DomParser=xml.dom.minidom

#获取要解析的文档

xmldom=DomParser.parse("Students.xml")

#获取要解析的文档的根节点

root=xmldom.documentElement

#添加一个子节点

childXML=xmldom.createElement("student")

#把这个节点追加到根节点下

root.appendChild(childXML)

#把id节点追加到student节点下

idNode=xmldom.createElement("id")

idNode.appendChild(xmldom.createTextNode("6"))

childXML.appendChild(idNode)

#把age节点追加到student节点下

ageNode=xmldom.createElement("age")

ageNode.appendChild(xmldom.createTextNode("16"))

childXML.appendChild(ageNode)

#把name节点追加到student节点下

nameNode=xmldom.createElement("name")

nameNode.appendChild(xmldom.createTextNode("花海"))

childXML.appendChild(nameNode)

#把sex节点追加到student节点下

sexNode=xmldom.createElement("sex")

sexNode.appendChild(xmldom.createTextNode("女"))

childXML.appendChild(sexNode)

#把address节点追加到student节点下

addressNode=xmldom.createElement("address")

addressNode.appendChild(xmldom.createTextNode("北京房山"))

childXML.appendChild(addressNode)

#把新内容写入文件

fileXML =open("student.xml","w")

#toprettyxml:toprettyxml()输出美化后的XML文本

fileXML.write(xmldom.toprettyxml())

studentsTags=root.getElementsByTagName("student")

for s in studentsTags:

#解析未知子节点

for nodechild in s.childNodes:

#节点类型是元素节点

if nodechild.nodeType==1:

print("<%s>%s</%s> " %(nodechild.nodeName,nodechild.firstChild.data,nodechild.nodeName))

print()

**第五部分**

**实例6、删除一个节点**

#encoding=utf-8

import xml.dom.minidom

DomParser=xml.dom.minidom

#获取要解析的文档

xmldom=DomParser.parse("Students.xml")

#获取要解析的文档的根节点

root=xmldom.documentElement

#删除一个sid=2的子节点

Childs=root.childNodes

for nodechild in Childs:

#节点类型是元素节点

if nodechild.nodeType==1:

#节点的id属性值为2

if nodechild.getAttribute("sid")=="2":

#删除该节点

root.removeChild(nodechild)

#把新内容写入文件

fileXML =open("student.xml","w")

#toprettyxml:toprettyxml()输出美化后的XML文本

fileXML.write(xmldom.toprettyxml())

**实例七、修改一个**子节点

#encoding=utf-8

import xml.dom.minidom

DomParser=xml.dom.minidom

#获取要解析的文档

xmldom=DomParser.parse("Students.xml")

#获取要解析的文档的根节点

root=xmldom.documentElement

#把一个sid=2的子节点下的name节点下的值改为陈城

Childs=root.childNodes

for nodechild in Childs:

#节点类型是元素节点

if nodechild.nodeType==1:

#节点的id属性值为2

if nodechild.getAttribute("sid")=="2":

for t in nodechild.childNodes:

if t.nodeName=="name":

print(t.firstChild.data)

t.firstChild.data='陈城'

break

#把新内容写入文件

fileXML =open("student.xml","w")

#toprettyxml:toprettyxml()输出美化后的XML文本

fileXML.write(xmldom.toprettyxml())