**Python之xml 模块详解**

xml 模块

Python 有三种方法解析 XML，SAX，DOM，以及 ElementTree:

### 1.SAX (simple API for XML )

Python 标准库包含 SAX 解析器，SAX 用事件驱动模型，通过在解析XML的过程中触发一个个的事件并调用用户定义的回调函数来处理XML文件。

### 2.DOM(Document Object Model)

将 XML 数据在内存中解析成一个树，通过对树的操作来操作XML。

### 3.ElementTree(元素树)

ElementTree就像一个轻量级的DOM，具有方便友好的API。代码可用性好，速度快，消耗内存少。

**注：**因DOM需要将XML数据映射到内存中的树，一是比较慢，二是比较耗内存，而SAX流式读取XML文件，比较快，占用内存少，但需要用户实现回调函数（handler）。

## A.ElementTree解析

**一、引入**

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。\*\*从结构上，很像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Python是如何处理XML语言文件的呢?下面一起来看看Python常用内置模块之xml模块吧。

**xml模块介绍**

xml是实现不同语言或程序之间进行数据交换的协议，功能跟json差不多，但json使用起来更简单，不过，古时候，在json还没诞生的黑暗年代，大家只能选择用xml呀，至今很多传统公司如金融行业的很多系统的接口还主要是xml。

**二、xml文档模板**

xml的格式如下，就是通过<>节点来区别数据结构的：

<?xml version="1.0"?>

<data> *#此处data就是一个根节点，所有的country都是data的子节点，country下的标签又是country的子节点*

<country name="Liechtenstein"> *#此处country就是标签名（tag），name是属性名，Liechtenstein是值*

<rank updated="yes">2</rank>

<year>2008</year>

<gdppc>141100</gdppc>

<neighbor name="Austria" direction="E"/>

<neighbor name="Switzerland" direction="W"/>

</country>

<country name="Singapore">

<rank updated="yes">5</rank>

<year>2011</year>

<gdppc>59900</gdppc>

<neighbor name="Malaysia" direction="N"/>

</country>

<country name="Panama">

<rank updated="yes">69</rank>

<year>2011</year>

<gdppc>13600</gdppc>

<neighbor name="Costa Rica" direction="W"/>

<neighbor name="Colombia" direction="E"/>

</country>

</data>

XML所用的标签语言又分为自闭和标签和非自闭和标签，自闭和标签只有在开头有tag，非自闭合标签在开头和结尾都有tag。  
这是一个自闭和标签， 2</ rank>这是一个非自闭合标签。

**三、xml模块的用法**

xml协议在各个语言里的都 是支持的，在python中可以用ElementTree操作xml：ElementTree是python的XML处理模块，它提供了一个轻量级的对象模型。在使用ElementTree模块时，需要import xml.etree.ElementTree的操作。ElementTree表示整个XML节点树，而Element表示节点数中的一个单独的节点。

**Python内XML树的增删改查**

**1、查看**

"""

# 注意: 返回值都是对应的标签节点对象.

print(root.iter('year')) # 全文搜索

print(root.find('country')) # 在root的子节点找，只找一个

print(root.findall('country')) # 在root的子节点找，找所有(类始于subprocess中的找到所有sections)

"""

import xml.etree.ElementTree as ET *# 这样导入的好处就是xml和etree包中的功能都能直接使用. 同时三者可以结合使用*

*# 打开文件, 读出xml文件对象*

tree = ET.parse('db.xml') *# 如上xml模板存入db.xml文件中*

print(tree) *# <xml.etree.ElementTree.ElementTree object at 0x0000023703B24070>*

*# 读出顶级节点对象date*

root = tree.getroot()

print(root) *# <Element 'data' at 0x0000023703C7E720>*

*# 查找三种方式*

*# 1. 全文搜索: root.iter('year')*

res = root.iter("year")

print(res) *# <\_elementtree.\_element\_iterator object at 0x0000023EE649DE50>*

for year in res:

print(''.center(50, '-'))

print(year.tag) *# 获取year节点对象的标签名*

print(year.attrib) *# 获取year节点对象的属性. 以key:value对的形式输出. key代指属性名, value代指属性值*

print(year.text) *# 获取year节点对象中的文本内容.*

"""

--------------------------------------------------

year

{'update': 'no'}

2018

--------------------------------------------------

year

{'update': 'no'}

2021

--------------------------------------------------

year

{'update': 'no'}

2021

"""

*# 2. 在root的子节点找，只找一个: root.find('country')*

res = root.find('country')

print(res.tag) *# country*

print(res.attrib) *# {'name': 'Liechtenstein'}*

print(res.text) *# 文本内容为空*

*# 递归查找country下的year. 并获取其标签名, 属性, 文本内容*

res = root.find('country').find('year') *# 等同于接着上面的继续, res.find('year')*

print(res) *# <Element 'year' at 0x000002A64D5BD590>*

print(res.tag) *# year*

print(res.attrib) *# {'update': 'no'}*

print(res.text) *# 2018*

*# 3. 在root的子节点找所有: root.findall("country")*

res = root.findall("country")

print(res) *# [<Element 'country' at 0x00000253D3A9E770>, <Element 'country' at 0x00000253D3ACD810>, <Element 'country' at 0x00000253D3ACDAE0>]*

for country in res:

print(''.center(50, '-'))

res = country.find('year')

print(res.tag)

print(res.attrib)

print(res.text)

'''

--------------------------------------------------

year

{'update': 'no'}

2018

--------------------------------------------------

year

{'update': 'no'}

2021

--------------------------------------------------

year

{'update': 'no'}

2021

'''

**2、修改**

import xml.etree.ElementTree as ET

tree = ET.parse('db.xml')

root = tree.getroot()

*# 需求: 把"db.xml"文件中的country所有year标签属性名改为no, 标签文本加10*

for year in root.iter('year'):

print(year)

year.text = str(int(year.text) + 10) *# 注意: "db.xml"使用year.text读出, 默认是字符串, 我们要使用int转换成整型才能进行数字运算.*

year.attrib = {'update': 'no'}

tree.write('db.xml')

*# 需求: 把"db.xml"文件中的country下所有gdppc标签文本加10000*

for gdppc in root.iter('gdppc'):

print(gdppc)

gdppc.text = str(int(gdppc.text) + 10000)

tree.write('db.xml')

**3、增加**

import xml.etree.ElementTree as ET

tree = ET.parse('db.xml')

root = tree.getroot()

for country in root.iter('country'):

year = country.find("year")

if int(year.text) > 2010:

*# 1. 调用ET.Element()方法增加标签, 属性, 文本*

flag = ET.Element('egon') *# 2. 添加标签*

flag.attrib = {'DSB': 'yes'} *# 3. 为添加的flag标签对象添加属性*

flag.text = '大帅逼1' *# 4. 为添加的flag标签对象添加文本内容*

country.append(flag) *# 5. 把添加的flag标签对象追加到country标签中, 作为country的子节点标签对象.(往country节点下添加子节点)*

tree.write("db.xml")

**4、删除**

import xml.etree.ElementTree as ET

tree = ET.parse('db.xml')

root = tree.getroot()

*# 需求: 在所有的country标签节点对象下的rank如果它的文本内容大于50, 那么就删除这个country*

for country in root.findall('country'):

rank = int(country.find('rank').text)

if rank > 50:

root.remove(country)

tree.write('db.xml')

**构建xml文件**

import xml.etree.ElementTree as ET

new\_xml = ET.Element("country") *# 创建标签country节点, 返回new\_xml节点对象*

name = ET.SubElement(new\_xml, "name", attrib={"update": "yes"}) *# 在new\_xml节点对象下创建标签名为"name"的子节点对象, 并指定属性*

name.text = 'egon' *# 为"name"字节的点对象添加文本内容*

age = ET.SubElement(new\_xml, 'year', attrib={'update': 'no'})

age.text = '18'

sex = ET.SubElement(new\_xml, 'sex')

sex.text = 'male'

et = ET.ElementTree(new\_xml) *# 生成文档对象*

et.write('text.xml', encoding='utf-8', xml\_declaration=True) *# 创建文件, 将该文档对象"et"写入.*

**获取根**

* xml树示例

<data>*#此处data就是一个根节点，所有的country都是data的子节点，country下的标签又是country的子节点*

<country name = "Liechtenstein">*#此处country就是标签名（tag），name是属性名，Liechtenstein是值*

<rank updated = "yes">2</rank>

<year>2008</year>

<gdppc>141100</gdppc>

<neighbor name ="Austria" direction ="E"/>

<neighbor name ="Switzerland" direction = "W"/>

</country>

<country name ="Singapore">

<rank updated="yes">5</rank>

<year>2011</year>

<gdppc>59900</gdppc>

<neighbor name ="Malaysia" direction="N"/>

</country>

</data>

**以上面的xml树获取根**

import xml.etree.cElementTree as ET*#import...as 是一种简写方式*

tree = ET.parse("a.xml")*#此处的a*

root = tree.getroot()

print(root)*#会打印一个内存地址，相当于一个对象*

**tag、attrib、text方法**

import xml.etree.cElementTree as ET

tree = ET.parse("a.xml")

root = tree.getroot()

for i in root:

print(i.tag)*#tag方法会得到标签内容*

print(i.attrib)*#attrib方法会得到属性名和属性值，以字典的形式显示*

for i in root:

for j in i:

print(j.tag)

print(j.attrib)

for i in root:

for j in i :

print(j.text)*#text方法会获得所有的数字值即两个属性名中间存的值，例 <year>2011</year>*

**.iter方法**

* .iter方法可以实现只遍历其中的某一个节点
* tree = ET.parse("a.xml")
* root = tree.getroot()
* for node in root.iter("year"):*#可以获得所有的year节点的内容*
* print(node.tag,node.attrib,node.text)

本章节使用到的 XML 实例文件 movies.xml 内容如下：

## movies.xml

<collection shelf="New Arrivals"> <movie title="Enemy Behind"> <type>War, Thriller</type> <format>DVD</format> <year>2003</year> <rating>PG</rating> <stars>10</stars> <description>Talk about a US-Japan war</description> </movie> <movie title="Transformers"> <type>Anime, Science Fiction</type> <format>DVD</format> <year>1989</year> <rating>R</rating> <stars>8</stars> <description>A schientific fiction</description> </movie> <movie title="Trigun"> <type>Anime, Action</type> <format>DVD</format> <episodes>4</episodes> <rating>PG</rating> <stars>10</stars> <description>Vash the Stampede!</description> </movie> <movie title="Ishtar"> <type>Comedy</type> <format>VHS</format> <rating>PG</rating> <stars>2</stars> <description>Viewable boredom</description> </movie> </collection>

## B、python使用SAX解析xml

SAX是一种基于事件驱动的 API。

利用SAX解析XML文档牵涉到两个部分: **解析器**和**事件处理器**。

解析器负责读取XML文档，并向事件处理器发送事件，如元素开始跟元素结束事件。

而事件处理器则负责对事件作出响应，对传递的XML数据进行处理。

<psax适于处理下面的问题：< p="" style="color: rgb(51, 51, 51); font-family: "Helvetica Neue", Helvetica, "PingFang SC", "Hiragino Sans GB", "Microsoft YaHei", "Noto Sans CJK SC", "WenQuanYi Micro Hei", Arial, sans-serif; font-size: 12px; font-style: normal; font-variant-ligatures: normal; font-variant-caps: normal; font-weight: 400; letter-spacing: normal; orphans: 2; text-align: start; text-indent: 0px; text-transform: none; white-space: normal; widows: 2; word-spacing: 0px; -webkit-text-stroke-width: 0px; background-color: rgb(255, 255, 255); text-decoration-style: initial; text-decoration-color: initial;">

* 1、对大型文件进行处理；
* 2、只需要文件的部分内容，或者只需从文件中得到特定信息。
* 3、想建立自己的对象模型的时候。

在python中使用sax方式处理xml要先引入xml.sax中的parse函数，还有xml.sax.handler中的ContentHandler。

### ContentHandler类方法介绍

**characters(content)方法**

调用时机：

从行开始，遇到标签之前，存在字符，content 的值为这些字符串。

从一个标签，遇到下一个标签之前， 存在字符，content 的值为这些字符串。

从一个标签，遇到行结束符之前，存在字符，content 的值为这些字符串。

标签可以是开始标签，也可以是结束标签。

**startDocument() 方法**

文档启动的时候调用。

**endDocument() 方法**

解析器到达文档结尾时调用。

**startElement(name, attrs)方法**

遇到XML开始标签时调用，name是标签的名字，attrs是标签的属性值字典。

**endElement(name) 方法**

遇到XML结束标签时调用。

## make\_parser方法

以下方法创建一个新的解析器对象并返回。

xml.sax.make\_parser( [parser\_list] )

参数说明:

* **parser\_list** - 可选参数，解析器列表

## parser方法

以下方法创建一个 SAX 解析器并解析xml文档：

xml.sax.parse( xmlfile, contenthandler[, errorhandler])

参数说明:

* **xmlfile** - xml文件名
* **contenthandler** - 必须是一个ContentHandler的对象
* **errorhandler** - 如果指定该参数，errorhandler必须是一个SAX ErrorHandler对象

## parseString方法

parseString方法创建一个XML解析器并解析xml字符串：

xml.sax.parseString(xmlstring, contenthandler[, errorhandler])

参数说明:

* **xmlstring** - xml字符串
* **contenthandler** - 必须是一个ContentHandler的对象
* **errorhandler** - 如果指定该参数，errorhandler必须是一个SAX ErrorHandler对象

## Python 解析XML实例

## 实例

#!/usr/bin/python # -\*- coding: UTF-8 -\*- import xml.sax class MovieHandler( xml.sax.ContentHandler ): def \_\_init\_\_(self): self.CurrentData = "" self.type = "" self.format = "" self.year = "" self.rating = "" self.stars = "" self.description = "" # 元素开始事件处理 def startElement(self, tag, attributes): self.CurrentData = tag if tag == "movie": print "\*\*\*\*\*Movie\*\*\*\*\*" title = attributes["title"] print "Title:", title # 元素结束事件处理 def endElement(self, tag): if self.CurrentData == "type": print "Type:", self.type elif self.CurrentData == "format": print "Format:", self.format elif self.CurrentData == "year": print "Year:", self.year elif self.CurrentData == "rating": print "Rating:", self.rating elif self.CurrentData == "stars": print "Stars:", self.stars elif self.CurrentData == "description": print "Description:", self.description self.CurrentData = "" # 内容事件处理 def characters(self, content): if self.CurrentData == "type": self.type = content elif self.CurrentData == "format": self.format = content elif self.CurrentData == "year": self.year = content elif self.CurrentData == "rating": self.rating = content elif self.CurrentData == "stars": self.stars = content elif self.CurrentData == "description": self.description = content if ( \_\_name\_\_ == "\_\_main\_\_"): # 创建一个 XMLReader parser = xml.sax.make\_parser() # turn off namepsaces parser.setFeature(xml.sax.handler.feature\_namespaces, 0) # 重写 ContextHandler Handler = MovieHandler() parser.setContentHandler( Handler ) parser.parse("movies.xml")

以上代码执行结果如下：

\*\*\*\*\*Movie\*\*\*\*\*

Title: Enemy Behind

Type: War, Thriller

Format: DVD

Year: 2003

Rating: PG

Stars: 10

Description: Talk about a US-Japan war

\*\*\*\*\*Movie\*\*\*\*\*

Title: Transformers

Type: Anime, Science Fiction

Format: DVD

Year: 1989

Rating: R

Stars: 8

Description: A schientific fiction

\*\*\*\*\*Movie\*\*\*\*\*

Title: Trigun

Type: Anime, Action

Format: DVD

Rating: PG

Stars: 10

Description: Vash the Stampede!

\*\*\*\*\*Movie\*\*\*\*\*

Title: Ishtar

Type: Comedy

Format: VHS

Rating: PG

Stars: 2

Description: Viewable boredom

完整的 SAX API 文档请查阅[Python SAX APIs](http://docs.python.org/library/xml.sax.html)

## C、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，实例如下：

## 实例

#!/usr/bin/python # -\*- coding: UTF-8 -\*- from xml.dom.minidom import parse import xml.dom.minidom # 使用minidom解析器打开 XML 文档 DOMTree = xml.dom.minidom.parse("movies.xml") collection = DOMTree.documentElement if collection.hasAttribute("shelf"): print "Root element : %s" % collection.getAttribute("shelf") # 在集合中获取所有电影 movies = collection.getElementsByTagName("movie") # 打印每部电影的详细信息 for movie in movies: print "\*\*\*\*\*Movie\*\*\*\*\*" if movie.hasAttribute("title"): print "Title: %s" % movie.getAttribute("title") type = movie.getElementsByTagName('type')[0] print "Type: %s" % type.childNodes[0].data format = movie.getElementsByTagName('format')[0] print "Format: %s" % format.childNodes[0].data rating = movie.getElementsByTagName('rating')[0] print "Rating: %s" % rating.childNodes[0].data description = movie.getElementsByTagName('description')[0] print "Description: %s" % description.childNodes[0].data

以上程序执行结果如下：

Root element : New Arrivals

\*\*\*\*\*Movie\*\*\*\*\*

Title: Enemy Behind

Type: War, Thriller

Format: DVD

Rating: PG

Description: Talk about a US-Japan war

\*\*\*\*\*Movie\*\*\*\*\*

Title: Transformers

Type: Anime, Science Fiction

Format: DVD

Rating: R

Description: A schientific fiction

\*\*\*\*\*Movie\*\*\*\*\*

Title: Trigun

Type: Anime, Action

Format: DVD

Rating: PG

Description: Vash the Stampede!

\*\*\*\*\*Movie\*\*\*\*\*

Title: Ishtar

Type: Comedy

Format: VHS

Rating: PG

Description: Viewable boredom

</psax适于处理下面的问题：<>

**参考资料**

* <https://www.w3school.com.cn/xml/xml_intro.asp>
* <https://blog.csdn.net/weixin_43690603/article/details/90765299>