[Tutorial] How to use The ElementTree XML API Python

时间 2014-05-22 00:55:24 Byte::Debugger();
原文 http://bytedebugger.wordpress.com/2014/05/21/tutorial-how-to-use-the-elementtree-xml-api-python/

Welcome ,

In this post I'm going to show you how to use xml.etree.ElementTree with Python. This module allow you to easily manipulate xml documents, reading and inserting data. In the final of this post, you can get a complete class to manipulate easily XML files with Python!!. Let's try it!!

In this post I'm using the following XML as example(" **downloads.xml"):

```
<?xml version="1.0"?>
<downloads>
  <file>
    <url>https://github.com/aron-bordin/Tyrant-Sql/archive/master.zip</url>
    <progress>23</progress>
    <filename>master.zip</filename>
  <path>/home/neo/Downloads/master.zip</path>
  <downloaded>1000</downloaded>
  <filesize>1231553</filesize>
  </file>
  <file>
    <url>https://github.com/aron-bordin/Haunted-Mind/archive/master.zip</url>
    <progress>11</progress>
    <filename>master.zip</filename>
  <path>/home/neo/Downloads/master2.zip</path>
  <downloaded>5335</downloaded>
  <filesize>23545363</filesize>
  </file>
</downloads>
```

I – Reading the XML

First, we need to read the XML from file, and after it, get the XML structure to manipulate it.

```
tree = ET.parse("downloads.xml")
root = tree.getroot()
```

Now, **root** has the XML object, so we are able to read this information with Python. First, let's list our downloads URLs. To do it, we need to read each child of <download> and the <url>. Let's do it:

```
for child in root:
    print(child[0].text) #we use 0 because url is the first child of <file>
```

output:

https://github.com/aron-bordin/Tyrant-Sql/archive/master.zip

https://github.com/aron-bordin/Haunted-Mind/archive/master.zip

Or, to make it easier, you can search for content. The **iter** method help iterate recursively over all the sub-tree below it (its children, their children, and so on). To list all URLs, you can do:

```
for url in root.iter("url"):
    print(url.text)
```

With this code we have the same output got with the previous code.

II – Modifying the XML file

II.i – Edit tag:

To manipulate the XML information, you need to use the same code above, but instead of get the **text** property, you need to set it.

For example, we need to set all download progress as 100%. To do it, we need to get all **progress** children and set the value as 100. Let's do it:

```
for progress in root.iter("progress"):
    progress.text = '100'

tree.write("downloads.xml")
```

It's done, all progress are updated to 100%.

II.ii – Remove tag:

If you need to remove a **<file>** from XML, you can use:

```
for file in root.iter("file"):
    root.remove(file)
tree.write("downloads.xml")
```

II.iii – Insert tag

Now, let's add a new **<file>** for our XML. Python provide us the command **Element** to create new elements, and **SubElement** to create children.

To save file data, we will need a file elements and some children. Let's do it!

```
file  = ET.Element('file')
progress = ET.SubElement(file, 'progress')
progress.text = "0" #if you want to set the prop, use it
ET.SubElement(file, 'url') #will create an empty tag
ET.SubElement(file, 'filename')
ET.SubElement(file, 'path')
ET.SubElement(file, 'downloaded')
ET.SubElement(file, 'filesize')

root.append(file) #add the tab file to our xml
tree.write("downloads.xml") #save the file
```

So, that's it! As you can see, Python works perfectly with XML. I used this commands in other project, so if you need to take a look, in the code bellow I posted the class that I used to manipulate my downloader.

III – Class XmlParser

With this class you can easily manipulate XML files. You can get this class and adapt for your project. Take a look:

```
import xml.etree.ElementTree as ET
import os.path

class XmlParser(object):

  def __init__(self):
    if os.path.isfile('downloads.xml') is False:
      self.createXML()
    self.tree = ET.parse("downloads.xml")
    self.root = self.tree.getroot()

  def createXML(self):
    f = open('downloads.xml', 'w')
    f.write('<downloads></downloads>')
    f.close()

  def getDownload(self, url):
    r = self.root
    for file in r:
      if(file.attrib['url'] == url):
        return file
    return False

  def getDownloads(self):
    r = self.root
    files = []
    for file in r:
      files.append(file)
    print(files)
```

```
def updateDownload(self, url, filename = None, path = None, progress = None, downloaded=None, filesize=None):
  d = self.getDownload(url)
  if d is False:
    return False
  if filename is not None:
    child = d.find('filename')
    child.text = filename
  if path is not None:
    child = d.find('path')
    child.text = path
  if progress is not None:
    child = d.find('progress')
    child.text = progress
  if downloaded is not None:
    child = d.find('downloaded')
    child.text = downloaded
  if filesize is not None:
    child = d.find('filesize')
    child.text = filesize
  self.tree.write('downloads.xml')

def removeDownload(self, url):
  d =self.getDownload(url)
  if d is False:
    return False
  self.root.remove(d)
  self.tree.write('downloads.xml')

def addFile(self, url, filename, path, progress = 0, downloaded=0, filesize=0):
  if self.checkIfExists(url):
    return "FileExists"
  file = ET.Element('file')
  file.attrib = {'url': url}
  child = ET.SubElement(file, 'url')
  child.text = str(url)
  child = ET.SubElement(file, 'filename')
  child.text = str(filename)
  child = ET.SubElement(file, 'path')
  child.text = str(path)
  child = ET.SubElement(file, 'downloaded')
  child.text = str(downloaded)
  child = ET.SubElement(file, 'filesize')
  child.text = 'filesize'
  child = ET.SubElement(file, 'progress')
  child.text = str(progress)
  self.root.append(file)
  self.tree.write('downloads.xml')

def checkIfExists(self, url):
  r = self.root
  for file in r.iter('url'):
    if url == file.text:
      return True
  return False
```

Thanks for reading, take a look in my blog for more tutorials.
Thx

- XML

- Python

热门文章

1. CFFI 1.0.1 released
2. PEP 492 vs. PEP 3152, new round
3. Building a realtime API with RethinkDB
4. Django Weblog: Security release issued: 1.8.2
5. Python Coroutines with Async and Await
6. Making the case for Jython

分享本文

×

用户登陆

邮箱

密码

登 陆

收藏到推刊

创建推刊

收 藏 取消

已收藏到推刊!

推刊名(必填) 请填写推刊名

推刊描述

描述不能大于100个字符!

权限设置： ◉ 公开 ○ 仅自己可见

创建 取消

网站相关

关于我们
移动应用
建议反馈

关注我们

推酷网

tuicool2012

QQ群:164644910

友情链接

人人都是产品经理 TMTForum 魔部网 PM256 品途网 移动信息化 行晓网 Code4App 智城外包网 LAMP人 安卓航班网 虎嗅 缘创派 IT耳朵 艾瑞网 创媒工场 硅谷网 经理人分享 市场部网 砍柴网 CocoaChina 北风网 云智慧 我赢职场 大数据时代 奇笛网 咕噜网 红联linux Win10之家 Cocos引擎中文官网 鸟哥笔记 爱游戏 投资潮 31会议网 极光推送 Teambition 更多链接>>