```
第31章-用lxml解析XML
Python 101
导航
                       在第一部分中,我们研究了Python的一些内置XML解析器。在本章中,我们将介绍有趣
                       的第三方软件包, lxml 从codespeak。它使用elementtree API和其他功能。lxml包支持
Introduction
                       xpath和xslt,包括SAX的API和与C/Pyrex模块兼容的C级API。以下是我们将介绍的内
Part I - Learning the
Basics
Chapter 1 - IDLE
                           • 如何用lxml解析XML
Programming
                                                                                      (i) X
Chapter 2 - All About
                                                           CISSP® Certification
Strings
                                                           广告 Higher pass rates than all
Chapter 3 - Lists, Tuples
                                                           official CISSP® Courses. 7-Day...
and Dictionaries
                                                            infosecinstitute.com
Chapter 4 - Conditional
                                                             Learn more
Statements
                           • 重构示例
Chapter 5 - Loops
                           • 如何使用lxml.objectify分析XML
Chapter 6 - Python
                           • 如何使用lxml.objectify创建XML
Comprehensions
Chapter 7 - Exception
                       对于本章,我们将使用微型计算机解析示例,并了解如何使用lxml解析这些示例。下面
Handling
                       是一个XML示例,来自一个为跟踪约会而编写的程序:
Chapter 8 - Working with
                          <?xml version="1.0" ?>
Files
                          <zAppointments reminder="15">
Chapter 9 - Importing
                              <appointment>
Chapter 10 - Functions
                                 <br/><begin>1181251680</begin>
Chapter 11 - Classes
                                 <uid><uid>040000008200E000</uid>
                                 <alarmTime>1181572063</alarmTime>
Part II - Learning from
                                 <state></state>
the Library
                                 <location></location>
Chapter 12 - Introspection
                                 <duration>1800</duration>
                                 <subject>Bring pizza home</subject>
Chapter 13 - The csv
                              </appointment>
Module
                              <appointment>
Chapter 14 - configparser
                                  <begin>1234360800</pegin>
                                 <duration>1800</duration>
Chapter 15 - Logging
                                 <subject>Check MS Office website for updates
Chapter 16 - The os
                                 <location></location>
Module
                                 <uid><uid>604f4792-eb89-478b-a14f-dd34d3cc6c21-1234360800</uid></ui>
Chapter 17 - The email /
                                  <state>dismissed</state>
                            </appointment>
smtplib Module
                          </zAppointments>
Chapter 18 - The sqlite
Module
                       让我们学习如何用lxml解析它!
Chapter 19 - The
subprocess Module
                       用lxml解析XML
Chapter 20 - The sys
Module
                       上面的XML显示了两个约会。开始时间以秒为单位;根据开始时间和键的散列值生成
Chapter 21 - The
                       uid;报警时间是自开始时间起的秒数,但应小于开始时间;状态是约会是否已暂停、取
threading module
                       消或不是。XML的其余部分很容易解释。现在我们来看看如何解析它。
Chapter 22 - Working
with Dates and Time
                          from lxml import etree
Chapter 23 - The xml
                          def parseXML(xmlFile):
module
Part III - Intermediate
                              Parse the xml
Odds and Ends
                              with open(xmlFile) as fobj:
Chapter 24 - The Python
                                 xml = fobj.read()
Debugger
Chapter 25 - Decorators
                              root = etree.fromstring(xml)
Chapter 26 - The lambda
                              for appt in root.getchildren():
Chapter 27 - Code
                                 for elem in appt.getchildren():
Profiling
                                     if not elem.text:
Chapter 28 - An Intro to
                                         text = "None"
                                     else:
Testing
                                         text = elem.text
Part IV - Tips, Tricks and
                                     print(elem.tag + " => " + text)
Tutorials
                          if __name__ == "__main__":
Chapter 29 - Installing
                              parseXML("example.xml")
Packages
Chapter 30 - ConfigObj
                       首先,我们导入所需的模块,即 埃特里 LXML包中的模块和 斯特林吉奥 内置功能 斯特
Chapter 31 - Parsing XML
                       林吉奥 模块。我们的 分析XML 函数接受一个参数:相关XML文件的路径。我们打开文
with lxml
                       件,读取并关闭它。有趣的部分来了!我们使用etree的parse函数来解析从stringio模块
Chapter 32 - Python Code
                       返回的XML代码。由于我不完全理解的原因,parse函数需要一个类似文件的对象。
Analysis
                       无论如何,接下来我们将遍历上下文(即 lxml.etree.iterparse对象 )并提取标记元
Chapter 33 - The requests
                       素。我们添加了条件if语句,用单词"none"替换空字段,使输出更加清晰。就这样。
package
Chapter 34 - SQLAlchemy
                       解析书籍示例
Chapter 35 - virtualenv
Part V - Packaging and
Distribution
                       这个例子的结果有点无聊。大多数情况下,您希望保存提取的数据并对其进行处理,而
Chapter 36 - Creating
                       不仅仅是将其打印到stdout。因此,对于下一个示例,我们将创建一个包含结果的数据结
Modules and Packages
                       构。本例的数据结构是一个dict列表。我们将再次使用前面一章中的msdn-book示例。将
Chapter 37 - How to Add
                       以下XML另存为 example.xml
Your Code to PyPI
                          <?xml version="1.0"?>
Chapter 38 - The Python
                          <catalog>
                             <book id="bk101">
Chapter 39 - Python
                                <author>Gambardella, Matthew</author>
                                <title>XML Developer's Guide</title>
wheels
                                <genre>Computer
Chapter 40 - py2exe
                                <price>44.95</price>
Chapter 41 - bbfreeze
                                <publish_date>2000-10-01
                                <description>An in-depth look at creating applications
Chapter 42 - cx_Freeze
                               with XML.</description>
Chapter 43 - PyInstaller
                             </book>
Chapter 44 - Creating an
                             <book id="bk102">
                                <author>Ralls, Kim</author>
Installer
                                <title>Midnight Rain</title>
                                <genre>Fantasy
快速搜索
                                <price>5.95</price>
                                <publish_date>2000-12-16</publish_date>
                转向
                                <description>A former architect battles corporate zombies,
                               an evil sorceress, and her own childhood to become queen
                               of the world.</description>
                             </book>
                             <book id="bk103">
                                <author>Corets, Eva</author>
                                <title>Maeve Ascendant</title>
                                <genre>Fantasy
                                <price>5.95</price>
                                <publish_date>2000-11-17</publish_date>
                                <description>After the collapse of a nanotechnology
                                society in England, the young survivors lay the
                               foundation for a new society.</description>
                             </book>
                          </catalog>
                       现在让我们分析这个XML并将其放入我们的数据结构中!
                          from lxml import etree
                          def parseBookXML(xmlFile):
                              with open(xmlFile) as fobj:
                                 xml = fobj.read()
                              root = etree.fromstring(xml)
                              book_dict = {}
                              books = []
                              for book in root.getchildren():
                                 for elem in book.getchildren():
                                     if not elem.text:
                                         text = "None"
                                     else:
                                         text = elem.text
                                     print(elem.tag + " => " + text)
                                     book_dict[elem.tag] = text
                                 if book.tag == "book":
                                     books.append(book_dict)
                                     book_dict = {}
                              return books
                          if __name__ == "__main__":
                              parseBookXML("books.xml")
                       这个例子与上一个非常相似,所以我们只关注这里的差异。在开始遍历上下文之前,我
                       们创建一个空字典对象和一个空列表。然后在循环内部,我们创建如下字典:
                          book_dict[elem.tag] = text
                       文本可以是 elem.text 或 None.最后,如果标签恰好是 book,然后我们在一个图书区
                       的末尾,需要将dict添加到列表中,并为下一本书重置dict。如你所见,这正是我们所做
                       的。一个更现实的例子是将提取的数据放入 Book 类。我以前用JSON提要完成过后者。
                       现在我们准备学习如何用 lxml.objectify 你说什么?
                       使用lxml.objectify分析XML
                       lxml模块有一个名为 客观化 它可以将XML文档转换为python对象。我发现"对象化"的
                       XML文档非常容易使用,希望您也能使用。你可能需要通过一两个环来安装它 pip 无法
                       在Windows上使用LXML。一定要转到python包索引并查找为您的python版本创建的版
                       本。另外请注意,最新的lxml预构建安装程序只支持python 3.2(在编写时),因此,如
                       果您有更新版本的python,则可能难以为您的版本安装lxml。
                       不管怎样,一旦您安装了它,我们就可以再次开始介绍这段美妙的XML:
                          <?xml version="1.0" ?>
                          <zAppointments reminder="15">
                              <appointment>
                                 <br/><begin>1181251680</begin>
                                 <uid><uid>040000008200E000</uid>
                                 <alarmTime>1181572063</alarmTime>
                                 <state></state>
                                 <location></location>
                                 <duration>1800</duration>
                                 <subject>Bring pizza home</subject>
                              </appointment>
                              <appointment>
                                 <begin>1234360800</pegin>
                                 <duration>1800</duration>
                                 <subject>Check MS Office website for updates
                                 <location></location>
                                 <uid><uid><604f4792-eb89-478b-a14f-dd34d3cc6c21-1234360800</uid></ui>
                                 <state>dismissed</state>
                            </appointment>
                          </zAppointments>
                       现在我们需要编写一些代码来解析和修改XML。让我们看一下这个小演示,它展示了对
                       象化提供的一系列整洁的能力。
                          from lxml import etree, objectify
                          def parseXML(xmlFile):
                              """Parse the XML file"""
                              with open(xmlFile) as f:
                                 xml = f.read()
                              root = objectify.fromstring(xml)
                              # returns attributes in element node as dict
                              attrib = root.attrib
                              # how to extract element data
                              begin = root.appointment.begin
                              uid = root.appointment.uid
                              # loop over elements and print their tags and text
                              for appt in root.getchildren():
                                 for e in appt.getchildren():
                                     print("%s => %s" % (e.tag, e.text))
                                 print()
                              # how to change an element's text
                              root.appointment.begin = "something else"
                              print(root.appointment.begin)
                              # how to add a new element
                              root.appointment.new_element = "new data"
                              # remove the py:pytype stuff
                              objectify.deannotate(root)
                              etree.cleanup_namespaces(root)
                              obj_xml = etree.tostring(root, pretty_print=True)
                              print(obj_xml)
                              # save your xml
                              with open("new.xml", "w") as f:
                                 f.write(obj_xml)
                          if __name__ == "__main__":
                             f = r'path\to\sample.xml'
                              parseXML(f)
                       代码注释得很好,但我们还是要花点时间来研究它。首先,我们将示例XML文件传递给
                       它,然后客观化它。如果要访问标记的属性,请使用阿特里布属性。它将返回标记属
                       性的字典。要获得子标记元素,只需使用点表示法。如你所见,去 开始 标记的值,我们
                       可以这样做:
                          begin = root.appointment.begin
                       需要注意的一点是,如果值恰好有前导零,则返回的值可能会截断它们。如果这对您很
                       重要,那么您应该使用以下语法:
                          begin = root.appointment.begin.text
                       如果需要迭代子元素,可以使用爱尔兰人方法。您可能需要使用嵌套的for循环结构来获
                       取所有内容。改变一个元素的值就像赋予它一个新的值一样简单。
                          root.appointment.new_element = "new data"
                       现在我们准备学习如何使用 lxml.objectify.
                       使用lxml.objectify创建XML
                       objectify子包对于解析和创建XML非常方便。在本节中,我们将展示如何使用
                       lxml.objectify模块创建XML。我们将从一些简单的XML开始,然后尝试复制它。我们开
                       始吧!
                       我们将继续使用以下XML作为示例:
                          <?xml version="1.0" ?>
                          <zAppointments reminder="15">
                              <appointment>
                                 <br/><begin>1181251680</begin>
                                 <uid><uid>040000008200E000</uid>
                                 <alarmTime>1181572063</alarmTime>
                                 <state></state>
                                 <location></location>
                                 <duration>1800</duration>
                                 <subject>Bring pizza home</subject>
                              </appointment>
                              <appointment>
                                 <br/><begin>1234360800</begin>
                                 <duration>1800</duration>
                                 <subject>Check MS Office website for updates
                                 <location></location>
                                 <uid><uid>604f4792-eb89-478b-a14f-dd34d3cc6c21-1234360800</uid></ui>
                                 <state>dismissed</state>
                            </appointment>
                          </zAppointments>
                       让我们看看如何使用lxml.objectify重新创建此XML:
                          from lxml import etree, objectify
                          def create_appt(data):
                              Create an appointment XML element
                              appt = objectify.Element("appointment")
                              appt.begin = data["begin"]
                              appt.uid = data["uid"]
                              appt.alarmTime = data["alarmTime"]
                              appt.state = data["state"]
                              appt.location = data["location"]
                              appt.duration = data["duration"]
                              appt.subject = data["subject"]
                              return appt
                          def create_xml():
                              Create an XML file
                              xml = '''<?xml version="1.0" encoding="UTF-8"?>
                              <zAppointments>
                              </zAppointments>
                              root = objectify.fromstring(xml)
                              root.set("reminder", "15")
                              appt = create_appt({"begin":1181251680,
                                                "uid":"040000008200E000",
                                                "alarmTime":1181572063,
                                                "state":"",
                                                "location":"",
                                                "duration":1800,
                                                "subject":"Bring pizza home"}
                              root.append(appt)
                              uid = "604f4792-eb89-478b-a14f-dd34d3cc6c21-1234360800"
                              appt = create_appt({"begin":1234360800,
                                                "uid":uid,
                                                "alarmTime":1181572063,
                                                "state":"dismissed",
                                                "location":"",
                                                "duration":1800,
                                                "subject": "Check MS Office website for updates
                              root.append(appt)
                              # remove lxml annotation
                              objectify.deannotate(root)
                              etree.cleanup_namespaces(root)
                              # create the xml string
                              obj_xml = etree.tostring(root,
                                                    pretty_print=True,
                                                    xml_declaration=True)
                              try:
                                 with open("example.xml", "wb") as xml_writer:
                                     xml_writer.write(obj_xml)
                              except IOError:
                                  pass
                          if __name__ == "__main__":
                              create_xml()
                       让我们把这个分解一下。我们将从 create_xml 功能。在其中,我们使用Objectify模块
                       的 从字符串 功能。根对象将包含 灾难 作为它的标签。我们设置根的 提醒 属性,然后我
                       们调用 create_appt 函数的参数使用字典。在 create_appt 函数,我们创建一个元素
                       的实例(从技术上讲,它是 ObjectifiedElemen t)我们分配给我们的 appt 变量。这
                       里我们用 dot-notatio n创建此元素的标记。最后我们返回 appt 元素返回并将其附加到
                       root对象。我们对第二个约会实例重复此过程。
                        下一部分 create_xml 函数将删除lxml注释。如果不这样做, XML最终会如下所示:
                          <?xml version="1.0" ?>
                          <zAppointments py:pytype="TREE" reminder="15">
                              <appointment py:pytype="TREE">
                                 <br/><begin py:pytype="int">1181251680</begin>
                                 <uid py:pytype="str">040000008200E000</uid>
                                 <alarmTime py:pytype="int">1181572063</alarmTime>
                                 <state py:pytype="str"/>
                                 <location py:pytype="str"/>
                                 <duration py:pytype="int">1800</duration>
                                 <subject py:pytype="str">Bring pizza home</subject>
                              </appointment><appointment py:pytype="TREE">
                                 <br/><begin py:pytype="int">1234360800</begin>
                                 <uid py:pytype="str">604f4792-eb89-478b-a14f-dd34d3cc6c21-1234
                                 <alarmTime py:pytype="int">1181572063</alarmTime>
                                 <state py:pytype="str">dismissed</state>
                                 <location py:pytype="str"/>
                                 <duration py:pytype="int">1800</duration>
                                 <subject py:pytype="str">Check MS Office website for updates/
                              </appointment>
                          </zAppointments>
                        要删除所有不需要的注释,我们调用以下两个函数:
                          objectify.deannotate(root)
                          etree.cleanup_namespaces(root)
                       最后一个难题是让LXML生成XML本身。这里我们使用lxml 埃特里 做艰苦工作的模块:
```

©2019, python. | Powered by <u>Sphinx 2.1.2</u> & <u>Alabaster 0.7.12</u>

pretty\_print=True,

ToString函数将返回一个很好的XML字符串,如果设置 pretty\_print 如果是这样,它

现在您知道了如何使用lxml的etree和objectify模块来解析XML。您还知道如何使用

Objectify创建XML。了解如何使用多个模块来完成相同的任务对于从不同的角度处理相

通常也会以一种好的格式返回XML。这个 xml\_declaration 关键字参数告诉etree模块

xml\_declaration=True)

obj\_xml = etree.tostring(root,

总结

是否包括第一个声明行(即 <?xml version="1.0"?>.

同的问题是很有价值的。它还将帮助您选择最适合的工具。

站长统计