Python XML Processing

XML adalah bahasa open source portable yang memungkinkan pemrogram mengemangkan aplikasi yang dapat dibaca oleh aplikasi lain, terlepas dari sistem operasi dan bahasa pengembangnya.

Apa itu XML?

Extensible Markup Languange (XML) adalah bahasa markup seperti HTML atau SGML. Ini direkomendasikan oleh World Wide Web Consortium dan tersedia sebagai standar terbuka.

XML sangat berguna untuk mencatat data berukuran kecil dan menengah tanpa memerlukan tulang punggung berbasis SQL.

2.1 Arsitektur Parsing XML dan API

Perpustakaan standar Python menyediakan seperangkat antarmuka minimal tapi berguna untuk bekerja dengan XML.

Dua API yang paling dasar dan umum digunakan untuk data XML adalah antarmuka SAX dan DOM.

API sederhana untuk XML (SAX): mendaftarkan panggilan kemali untuk acara yang diminati dan kemudian membiarkan parser berjalan melalui dokumen. Ini berguna bila dokumen berukuran besar atau memiliki keterbatasan memori, ini memparsing file tidak pernah tersimpan dalam memori.

API Document Objek Model (DOM): ini adalah rekomendasi World Wide Web Consortium dimana keseluruhan file dibaca ke memori dan disimpan dalam bentuk hierarkies (tree-based) untuk mewakili semua fitur dokumen XML.

SAX jelas tidak bisa memproses informasi secepat DOM saat bisa bekerjadengan file besar. Di sisi lain, menggunakan DOM secara eklusifenar-benar dapat membunuh sumber daya, terutama jika digunakan pada banyak file kecil.

SAX hanya bisa dibaca sementara DOM mengizinkan perubahan pada file XML. Kedua API yang berbeda ini saling melengkapi satu sama lain, tidak ada alasan mengapa tidak dapat menggunakannya untuk proyek besar.

Contoh:

```
<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>
```

2.2 Parsing XML dengan API SAX

SAX adalah antarmuka standar untuk parsing XML berbasis event. Parsing XML dengan SAX umumnya mengharuskan untuk membuat C*ontrolHandler* dengan subclassing xml.sax *controlhandler*.

ControlHandler menangani tag dan atribut tertentu dari XML. Objek *ControlHandler* menyediakan metode untuk menangani berbagai aktivitas parsing. Parsing memanggil metode *ControlHandler* saat memparsing file XML.

Metode *startDocument* dan *endDocument* disebut awal dan akhir setiap elemen. Jika parsing tidak dalam mode namespace, metode *startElement* (tag attribute) dan *endElement* (tag) dipanggil. Jika tidak, metode yang sesuai *startElemenNS* dan *endElemenNS* dipanggil. Disini, tah adalah tag elemen dan atriut adalah atribut.

Berikut ini metode penting untuk memahami sebelum melanjutkan ke materi berikutnya :

1) Metode *make _parser*

   Metode berikut membuat objek parsing baru dan mengembalikannya. Objek parsing diuat akan menjadi tipe parsing pertama yang ditemukan sistem.

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

   Berikut adalah detail parameternya :

   Parser _list : pilihan argumen yang terdiri dari daftar parsing untuk digunakan yang semuanya harus menerapkan metode *make _parse*

2) Metode *parser*

   Metode berikut membuat parsing SAX dan menggunakannya untuk mengurai dokumen

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

   Berikut adalah detail dari parameternya:

   - *Xmlfile*
     Ini adalah nama file XML yang bisa dibaca.
   - *ContentHandler*
     Ini harus menjadi objek *ContenHandler*
   - *ErrorHandler*
     Jika ditentukan, e*rrorhandler* harus menjadi objek *ErrorHandler* SAX
   - Metode *parseString*

   Membuat parsing SAX dan mengurai string XML yang ditentukan :

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

   Brikut ini adalah detail nama dar parameter :

3) *XMLstring*

   Nama dari string yang bisa dibaca

4) *ContentHandler*

   Menjadi objek ContenHandler

5) *ErrorHandler*

   Menjadi objek ErorHandler SAX

   Contoh :

   #!/usr/bin/python

```
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 = ""

   # Call when an element starts
   def startElement(self, tag, attributes):
      self.CurrentData = tag
      if tag == "movie":
         print "*****Movie*****"
         title = attributes["title"]
         print "Title:", title

   # Call when an elements ends
   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 = ""

   # Call when a character is read
```

```
    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 _ _"):

   # create an XMLReader
   parser = xml.sax.make _parser()
   # turn off namepsaces
   parser.setFeature(xml.sax.handler.feature _namespaces, 0)

   # override the default ContextHandler
   Handler = MovieHandler()
   parser.setContentHandler( Handler )

   parser.parse("movies.xml")
```

Ini akan menghasilkan hasil sebagai berikut:

```
*****Movie******
*****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

2.3 Parsing XML dengan API DOM

Document Ovject Model (DOM) adalah API lintas bahasa dari World Wide Web Consortium (W3C) untuk mengakses dan memodifikasi dokumen XML.

DOM sangat berguna untuk aplikasi akses acak. SAX hanya memungkinkan melihat satu bit dokumen sekaligus. Jika melihat satu elemen SAX, tidak memiliki akses ke yang lain.

Berikut adalah cara termudah untuk memuat dokumen XML dengan cepat dan membuat objek minidom menggunakan modul xml.dom. Objek minidom menyediakan metode parsing sederhana yang dengan cepat memuat pohon DOM dari file XML.

Contoh frase memanggil fungsi parsing (file [,parsing]) dari objek minidokumen untuk mengurai file XML yang ditunjuk oleh file ke objek pohon DOM.

```
#!/usr/bin/python

from xml.dom.minidom import parse
import xml.dom.minidom

# Open XML document using minidom parser
DOMTree = xml.dom.minidom.parse("movies.xml")
collection = DOMTree.documentElement
if collection.hasAttribute("shelf"):
  print "Root element : %s" % collection.getAttribute("shelf")

# Get all the movies in the collection
movies = collection.getElementsByTagName("movie")

# Print detail of each movie.
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
```

Ini akan menghasilkan hasil sebagai berikut :

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

2.4 Membangun Parsing Document XML menggunakan Python

Python mendukung untuk bekerja dengan berbagai bentuk markup data terstruktur. Selain mengurai xml.etree. *ElementTree* mendukung pembuatan dokumen XML yang terbentuk dengan baik dari objek elemen yang dibangun dalam aplikasi. Kelas elemen digunakakan saat sebuah dokumen diurai untuk mengetahui bagaimana menghasilkan bentuk serial dari isinya kemudian dapat ditulis ke sebuah file.

Untuk membuat instance elemeb gunakan fungsi elemen contructor dan *SubElemen()* pabrik.

Import xml.etree.ElementTree as xml

```
filename = ”/home/abc/Desktop/test _xml.xml ”
toot = xml.Element( ”Users ”)
userelement = xml.Element( ”user ”)
root.append(userelement)
```

Bila menjalankan ini, akan menghasilkan sebagai berikut :

```
<Users>
        <user>
        <user>
</Users>
```

Tambahkan anak-anak pegguna

```
Uid = xml.SubElement(userelement, ”uid ”)
Uid.text = ”1 ”

FirstName = xml.SubElement(userelement, ”FirstName ”)
FirstName.text = ”testuser ”

LastName = xml.SubElement(userelement, ”LastName ”
LastName.text = ”testuser ”

Email = xml.SubElement(userelement, ”Email ”)
Email.text = testuser@test.com

state = xml.SubElement(userelemet, ”state ”)
state.text = ”xyz ”

location = xml.SubElement(userelement, ”location)
location.text = abc

tree = xml.ElementTree(root)
with open(filename, ”w ”) as fh:
tree.write(fh)
```

Pertama buat elemen root dengan mengunakan fungsi *ElementTree*. Kemudian membuat elemen pegguna dan menambahkannya ke root. Selanjutnya membuat *SubElement* dengan melewatkan elemen pengguna (userelement) ke *SubElemen* beserta namanya seperto ”FirstName ”. Kemudian untuk setiap *SubElement* tetapkan

properti teks untuk memberi nilai. Di akhir, membuat *ElementTree* dan menggunakannya untuk menulis XML ke file.

Jika menjalankan ini akan menjadi sebagai berikut :

```
<users>
    <user>
        <uid>1</uid>
        <FirstName>testuser</FirstName>
        <LastName>testuser</LastName>
        <Email>testuser@test.com</Email >
        <state>xyz</state>
        <location>abc</location>
    </user>
</Users>
```

Parsing XML Documen :

```
import xml.etree.ElementTree as ET
tree = ET.parse('Your _XML _file _path')
root = tree.getroot()
```

Disini *getroot()* akan mengembalikan elemen dari dokumen XML

```
<Users version= ”1.0 ” languange= ”SPA ”>
        <user>
                <uid>1</uid>
                <FirstName>testuser</FirstName>
                <LastName>testuser</LastName>
                <Email>testuser@tes.com/Email>
                <state>xyz</state>
                <location>abc</location>
        </user>
</Users>
```