XMLing with Python

ملفات ال xml من اهم الملفات اللى بنتعامل معاها بصورة شبه يومية وبايثون من انسب الحلول للتعامل معاها..
فى اكتر من باكيج للتعامل مع ال Markups
http://docs.python.org/lib/markup.html

على فرض عندنا ملف كالتالى

```
<?xml version="1.0"?>

<!DOCTYPE books SYSTEM "books.dtd">
<?xml-stylesheet type="text/xsl" href="books.xsl"?>


<books>
   <book id="1">
      <name>Introduction to Python</name>
      <author>Ahmed Youssef</author>
      <price>80</price>
   </book>
   <book id="2">
      <name>Introduction to Java</name>
      <author>Wael Muhammed</author>
      <price>130</price>
   </book>
   <book id="3">
      <name>Introduction to Ruby</name>
      <author>Ahmed Youssef</author>
      <price>70</price>
   </book>
   <book id="4">
         <name>Introduction to Linux Programming</name>
         <author>Ahmed Mostafa</author>
         <price>90</price>
   </book>
</books>
```

فى root وهى ال books tag
وليها ابناء كل واحد بإسم book
كل book tag ليه attributes ؟ ايوة كل book لية id معين
داخل كل book بيشمل name, author, price tags لإسم الكتاب والكاتب والسعر

من الملف دا عايزين نجيب اسم كل كتاب ومجموع السعر بتاعهم

minidom -1
فى implementation خفيفة ل DOM بإسم minidom هنستدعيها كالتالى
import xml.dom.minidom as md #(parse, parseString..)
فى عندنا دالتين مهمين وهم parse, parseString
parse للتعامل مع file
parseString للتعامل مع string
والإتنين هيدولك ريترن ب document object

ال node object
هو يعتبر الأب لكل العناصر ملف ال xml وليه ميثودز/صفات مهمة
nodeType
بتعبر عن النوع هل هى text node, element, comment,document,.. etc
parentNode
رفرنس للأب (ماعدا ال document root) ولل attrs هتكون ديما None
previousSibling
ال node السابقة لل node الحالية إلا اذا كانت هى الأولى
nextSibling
ال node التالية إلا اذا كانت ال node الحالية هى الأخيرة

childNodes
جميع ال nodes اللى داخل ال node الحالية

hasChildNodes()
هل فى nodes داخلها ؟
firstChild
اول ابن
lastChild
اخر ابن
hasAttributes()
هل فيها attributes ؟
appendChild(child)
إضافة ابن جديد
insertBefore(child, befored)
بتضيف child قبل ال befored وفى حال عدم وجوده بيتم إضافته فى النهاية
removeChild(child)
حذف ابن child

normalize()
ربط ال text nodes المتقاربة

ال document object بيعبر عن الملف وليه ميثودز/صفات مهمة زى
documentElement # used as a property
ودى بتعبر عن ال root element وفى مثالنا هنا هى books
getElementsByTagName(tagName) #tagName
بتدور على tagName معين فى كل الأبناء وابنائهم وهكذا وتديلك ريترن ب element object
createElement(tagName)

لإنشاء tag جديد

createComment(comment)

لإنشاء تعليق داخلى

createAttribute(attr)

لإنشاء صفة attribute

ملحوظة فى بعض الميثودز بنفس الإسم ولكن اخرها NS ودى لربطها مع namespace ما وبتاخد nsURI كمعامل ليها.

ال Element Object بيعبر عن عنصر معين فى الملف وليه ميثودز مهمة زى

tagName #used as a property

بتعيد الإسم المجرد لل element

getElementsByTagName*

مشابهه للموجودة بال document object

hasAttribute(attrName)

هل بيحتوى على attribute ؟

getAttribute(attrName)

بيعيدلك قيمة attribute معينة بإسم attrName

setAttribute(attrName, val)

بيربط attribute معينة attrName ليها قيمة val بالعنصر

removeAttribute(attrName)

لحذف attribute معينة attrName (مش بيرفع اى exception !)

ملحوظة فى بعض الميثودز بنفس الإسم ولكن اخرها NS ودى لربطها مع namespace ما وبتاخد nsURI كمعامل ليها.

مجموعة ال exceptions

http://docs.python.org/lib/dom-exceptions.html

طيب تمام

1- استدعى ال minidom

```
import xml.dom.minidom as md #(parse, parseString..)
```

2- انشئ ال document object سواء بإستخدام parse او parseString حسب تخزينك لملف ال xml

```
doc=md.parse("books.xml")
```

3- احصل على ال document root و اعرضه واحصل على كل tag قيمته book واطبعه

```
def inspectBooks():
    global doc
    print "Root Element: ", doc.documentElement.tagName
    books=doc.getElementsByTagName("book")
    for book in books:
        if book.hasAttribute("id"): #id and it should have one!
```

```
            print "ID: ",book.getAttribute("id")
        for child in book.childNodes:
            if child.nodeType==child.ELEMENT_NODE:
                if child.tagName=="name":
                child.normalize()
                print "Book: ",child.firstChild.data
```

تمام ال doc هنا global variable

```
global doc
```

الحصول على ال document root هنا جالنا ريترن ب Element object واحنا عايزين ال tagName

```
doc.documentElement.tagName
```

نحصل على كل العناصر اللى tagName بتاعها book

```
books=doc.getElementsByTagName("book")
```

نعمل loop على كل عنصر فيها

```
for book in books:
```

اذا كان فيه id attribute (لمجرد عرض المثال)

```
    if book.hasAttribute("id"): #id and it should have one!
        print "ID: ",book.getAttribute("id")
```

طيب ولطباعة اسم الكتاب؟ لاحظ انه متخزن فى ال name tag
بسيطة جدا نعمل loop على كل الأبناء فى لا book element ونشوف النوع اذا كان ELEMENT NODE
و ال tagName بتاعه هو name نطبعه

```
                if child.tagName=="name":
                child.normalize()
                print "Book: ",child.firstChild.data
```

ملحوظة لل nodes انواع كتير element, comment, text, .. etc

ال firstChild دا بيعبر عن ال text node اللى فى ال name tag وال data بتدى ريترن بال string اللى جواها

```
<name> text node ... </name
```

الحصول على الثمن الكلى

```
def getTotalSum():
    global doc
    thesum=0
    prices=doc.getElementsByTagName("price")
    for price in prices:
        price.normalize()
        thesum += int(price.firstChild.data) #TO int.
    return thesum
```

نحصل على كل ال price elements

```
<price>numeric_value</price>
```

ونحول القيمة ل int وبس ونضيفها على ال thesum وبعد مانخلص نعمل الريترن بيها

ناتج التنفيذ ل

```
inspectBooks()
print "Total Sum: ", getTotalSum()
```

```
Root Element:  books
ID:  1
Book:  Introduction to Python
ID:  2
Book:  Introduction to Java
ID:  3
Book:  Introduction to Ruby
ID:  4
Book:  Introduction to Linux Programming
Total Sum:  370
```

SAX -2

بيعتمد على ال events بمعنى انه بيديلك خبر كل مايبدأ عنصر او يبدأ ال content اللى داخله وهكذا يمكن تشوفه اعقد شوية لكن انا عن نفسى من محبى استخدامه

1- استدعى اللى هنستخدمه

```
from xml.sax import make_parser, parseString
from xml.sax.handler import ContentHandler
```

ال ContentHandler دا هو المفتاح السحرى بتاعنا فيه ميثودز event handlers بيتعمل ليها استدعاء عند حدوث حدث معين

startDocument()

بيتم استدعائها مرة واحدة عند بداية الملف

endDocument()

بيتم استدعائها مرة واحدة عند نهاية الملف

startElement(name, attrs)

بيتم استدعائها عند بداية قراءة كل عنصر el

<el [attr1=val1, attr2=val2, ... attrN=valN]>CONTENT</el>

characters(content)

بيتم استدعائها عن بداية قراءة محتوى العنصر

<el [attr1=val1, attr2=val2, ... attrN=valN]>**CONTENT**</el>

endElement(el)

بيتم استدعائها عند نهاية قراءة عنصر el

<el [attr1=val1, attr2=val2, ... attrN=valN]>CONTENT**</el>**

فى بعض الميثودز بتنتهى ب NS ودى فى حالة التعامل مع namespace

اللل Attributes ماهى الا mapping او dictionary مضاف ليها بعض الميثودز مثل

getLength()

للحصول على عددهم

getNames()

الحصول على اسم كل attribute

getType()

للحصول على النوع وهى عادة CDATA

getValue(attrName)

الحصول على القيمة المرافقة لل attribute المسماة attrName

نرجع للمثال

1- هنستدعى الميثودز/الكلاسز المستخدمة

```
from xml.sax import make_parser, parseString
from xml.sax.handler import ContentHandler
```

ال make_parser هتعيد لينا XML reader

parseString لقراءة ال xml من string

parse هى ميثود تبع ال XML reader object بتاخد مسار ملف (نفس parse,parseString اللى تحدثنا عنهم فى minidom)

2- ملف ال xml ك string مخزن

```
xmldoc="""<?xml version="1.0"?>


<books>
   <book id="1">
      <name>Introduction to Python</name>
      <author>Ahmed Youssef</author>
      <price>80</price>
   </book>
   <book id="2">
      <name>Introduction to Java</name>
      <author>Wael Muhammed</author>
      <price>130</price>
   </book>
   <book id="3">
      <name>Introduction to Ruby</name>
      <author>Ahmed Youssef</author>
      <price>70</price>
   </book>
   <book id="4">
         <name>Introduction to Linux Programming</name>
         <author>Ahmed Mostafa</author>
         <price>90</price>
   </book>
</books>
"""
```

3- انشئ كلاس جديد مشتق من ال ContentHandler

```
class BooksHandler(ContentHandler):
```

ملحوظة اى ميثود مش هتعملها override مش تكتبها فى الهاندلر..

```
    def __init__(self):

        self._total=0 #Sum of prices.
        self._curel=None
        self._curid=None
        self._booksInfo=[]
        self._authors=[]
```

ايه المتغيرات دى كلها ؟
self._total للتخزين المجموع الكلى للأسعار
self._curel لتخزين اسم العنصر اللى بيتم معالجته
self._curid لتخزين اخر id تم قرائته
self._booksInfo تخزين معلومات عن الكتاب مكونة من ال name, id
self._authors تخزين اسماء الكتاب

```
    def getTotal(self):
        return self._total

    def getBooksInfo(self):
        return self._booksInfo

    def getAuthors(self):
        return self._authors
```

عرفنا getters للوصول للمتغيرات الداخلية
ملحوظة يفضل تستخدم properties مع lambda

```
    booksinfo=property(fget=lambda self: self._booksInfo)
    authors=property(fget=lambda self: self._authors)
    total=property(fget=lambda self: self._total)

    def startDocument(self):
        #print "Starting Document."
            pass
```

لو حبيت تضيف اى رسالة او اى حاجة على هواك يتم تنفيذها عند بداية قراءة الملف

```
    def endDocument(self):
        #print "Ending Document."
            pass
```

نفس السابقة ولكن عند انتهاء القراءة

```
    def startElement(self, el, attrs):

        #print "Starting ", el
        self._curel=el
            if el=="book":
                    #get the id..
                    self._curid=attrs.getValue("id") #attrs["id"]
```

هنا هيتم الإستدعاء عند بداية قراءة كل عنصر el والصفات الخاصة بيه attrs
1- نخزن العنصر الحالى فى ال self._curel

```
self._curel=el
```

2- نختبر اذا كان العنصر الحالى هو book فليه attribute بإسم id فنحصل عليها ونخزنها كآخر id لآخر كتاب تم قرائته فى ال reader

```
if el=="book":
        #get the id..
        self._curid=attrs.getValue("id") #attrs["id"]
```

طبعا تقدر تحصل عليها كأنك بتتعامل مع dict مش بإستخدام .getValue ميثود

```
def characters(self, content):
    if content.strip():

        if self._curel=="price":
            #print "In Price.."
            try:
                self._total += int(content)
            except:
                pass
        elif self._curel=="name":
            self._booksInfo +=[(content, self._curid)]
        elif self._curel=="author":
            self._authors +=[content]
        else:
            pass
```

هنا هيتم استدعائها عن قراءة المحتوى للعنصر الحالى
وبناءا على العنصر الحالى هنتعامل سواء اذا كان ثمن او اسم الكتاب او الكاتب

4- ننشئ اوبجكت من الهاندلر الجديد

```
bh = BooksHandler()
```

ننشئ XML reader ونباصى ليه ال handler الجديد والفايل اللى هيتعالج او نستخدم parseString ونباصى ليها الهاندلر(bh)

```
p = make_parser( )
p.setContentHandler(bh)
p.parse(open("books2.xml"))
```

او نستخدم parseString نحدد ال string اللى هيتعالج وال هاندلر (bh)

```
parseString(xmldoc, bh)
```

للمزيد عن
http://docs.python.org/lib/module-xml.sax.html
ومش تنسى http://www.saxproject.org/

استخدم مين ؟
همم DOM بيتعمد على انشاء tree للملف ودا شاق جدا للملفات اللى حجمها كبير!
من ناحية اخرى SAX بيعتمد على ال events ودا اسلوب فعال جدا

3- Expat undercover

Expat مكتبة سى سريعة لمعالجة ملفات ال XML وتم عمل wrapper ليها فى بايثون

1- استدعى الموديلز اللازمة

```
import xml.parsers.expat as exp
```

2- انشئ كلاس جديد بنفس فكرة ال ContentHandler

```
class ParsingHandler(object):

    def __init__(self, xml):
        self._curel=None
        self._curattrs=None
        self._inbook=False
        self._books=[]
        self._thesum=0

        self._p=exp.ParserCreate()
        self._p.StartElementHandler=self.__startElement
        self._p.EndElementHandler=self.__endElement
        self._p.CharacterDataHandler=self.__charsDataHandler
        self._p.Parse(xml)
```

لاحظ عندنا متغيرات لمتابعة العنصر الحالى والصفات الحالية ليه وهل احنا داخل ال book tag او لأ واسماء الكتاب والمجموع الكلى

القسم التانى متعلق بال parser
1- انشئ XMLParserType object بإستخدام ParserCreate
2- اربط ال handlers المختصين ببداية كل عنصر ونهايته والمحتوى ب handlers انت هتجهزهم لاحقا
3- عالج ال xml بإستخدام ال Parse method

انشئ getters

```
    def getTotalSum(self):
        return self._thesum

    def getBooksInfo(self):
        return self._books

    def printBooksInfo(self):
        for book in self._books:
            print book
```

عرف ال handlers بتوعنا اللى اسندناهم للهاندلرز الأساسين لل self._p parser

```
    def __startElement(self, el, attrs):
        print "Starting: ",el, attrs
        if el=="book":
            self._inbook=True
        self._curel=el
        self._curattrs=attrs

    def __charsDataHandler(self, data):
        if data.strip():
            if self._inbook and self._curel=="name" :
                self._books += [data]
            elif self._curel=="price" :
                self._thesum += int(data)
            else:
                pass

    def __endElement(self, el):
        if el=="book":
            self._inbook=False
        self._curel, self._curattrs=None,None
```

الإستخدام

```
if __name__=="__main__":
    p=ParsingHandler(xmldoc)
    print "Total sum: ", p.getTotalSum()
    p.printBooksInfo()
```

لاحظ ان ال xmldoc هو string بيعبر عن ملف ال xml اللى هيتم معالجته

ناتج التنفيذ

```
Total sum:  370
Introduction to Python
Introduction to Java
Introduction to Ruby
Introduction to Linux Programming
```