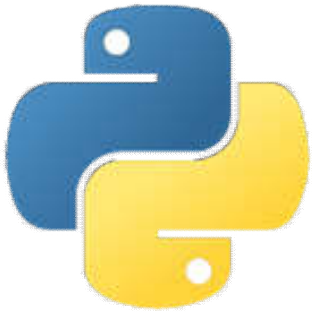


XML trong Python

Nội dung

I. Giới thiệu về XML
II. Cấu trúc XML
III. XML DOM/SAX
IV. Xử lý XML trong Python
V. Bài tập thực hành

Tổng quan XML

- Markup Language – ngôn ngữ đánh dấu: định nghĩa tập hợp các qui tắc để mã hóa các tài liệu trong một định dạng có thể đọc được bởi con người và máy tính.
- XML là viết tắt của *Extensible Markup Language* (Ngôn ngữ đánh dấu mở rộng) và là một ngôn ngữ đánh dấu *(không phải là ngôn ngữ lập trình)* dựa trên text được kế thừa từ Standard Generalized Markup Language (SGML).
- Các thẻ trong XML thường được tạo theo quy ước của người dùng.
- XML giới thiệu các tính năng mới bằng việc dựa trên các ưu điểm của HTML.

Đặc điểm

- **XML dễ dàng mở rộng**: XML cho phép tạo các thẻ theo qui ước của riêng mình để phù hợp với nghiệp vụ.
- **XML mang dữ liệu mà không hiển thị**: XML cho phép lưu giữ dữ liệu và không quan tâm đến cái cách nó sẽ được hiển thị.
- **XML là một chuẩn chung**: XML được phát triển bởi tổ chức World Wide Web Consortium (W3C) và là một chuẩn mở.

Sử dụng XML

- XML để đơn giản hóa việc tạo các tài liệu HTML cho các Website lớn.
- XML có thể được sử dụng để trao đổi thông tin giữa các thành phần hoặc giữa các hệ thống khác nhau.
- XML có thể được sử dụng để Offload và Reload các cơ sở dữ liệu.
- XML có thể được sử dụng để lưu trữ, sắp xếp và tinh chế dữ liệu cần xử lý.
- XML có thể dễ dàng để được hợp nhất với Style Sheet (CSS) để tạo ra hầu như bất kỳ kết quả mong muốn nào hay có thể nói bất kỳ kiểu dữ liệu nào có thể được biểu diễn như là một tài liệu XML.

Cú pháp

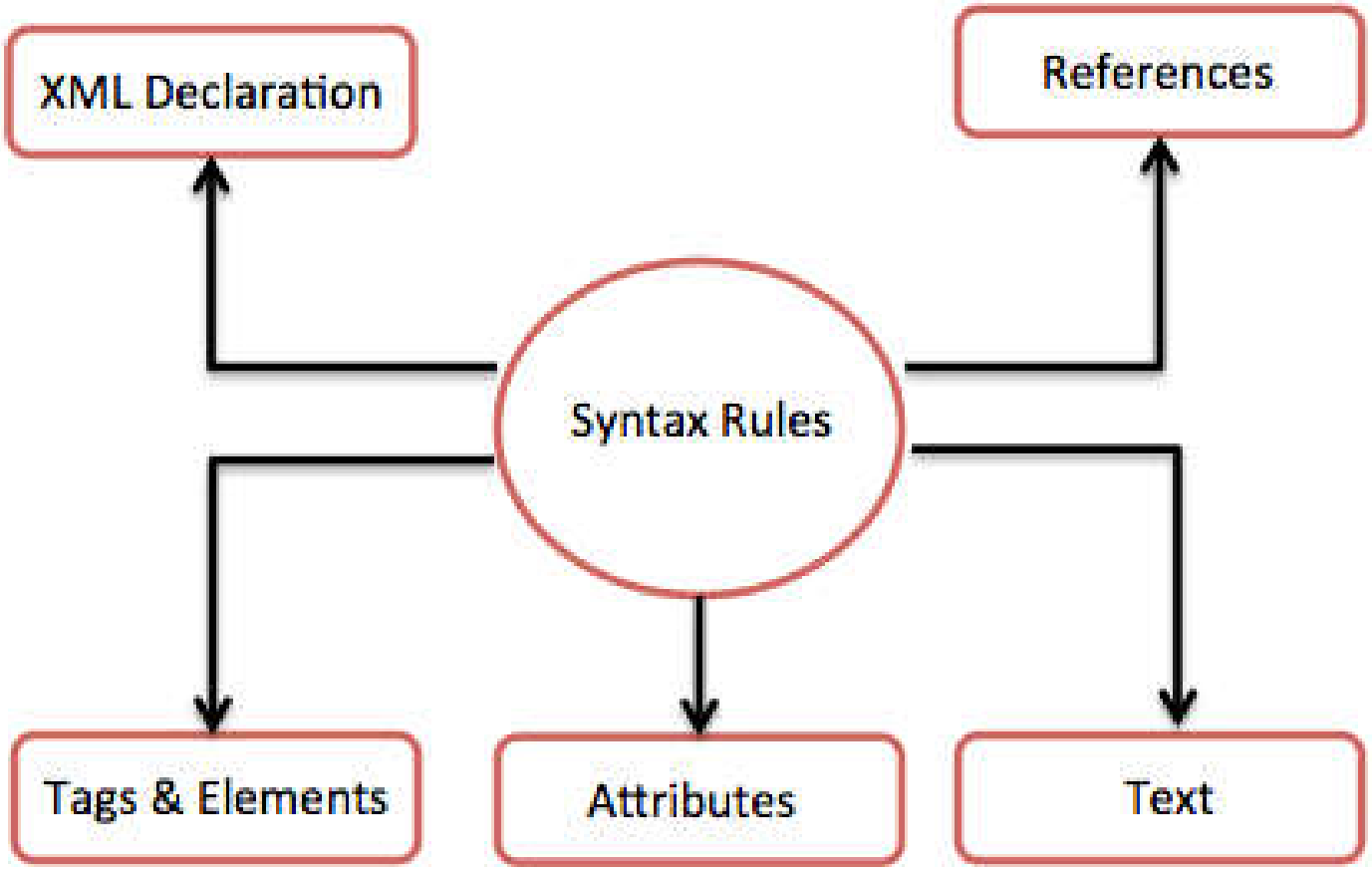
- Cú pháp một tài liệu xml:

```
<?xml version="1.0"?>
<contact-info>
    <name>Nguyen Van A</name>
    <company>Bravo</company>
    <phone>(84) 123-4567</phone>
</contact-info>
```

- Tài liệu xml cung cấp 2 loạithông tin:
  - Thông tin đánh dấu, như <contact-info> và
  - Text hay dữ liệu là ký tự, như *Nguyen Van A* và *(84) 123-4567.*

Cấu trúc cây XML

```
<?xml version="1.0" encoding="UTF-8"?>
<ds-sinhvien>
    <sinhvien>
        <tensv>[Data]</tensv>
        <namsinh>[Data]</namsinh>
        <gioitinh>[Data]</gioitinh>
    </sinhvien>
</ds-sinhvien>
```


XML Declaration
References
Syntax Rules
Tags & Elements
Attributes
Text

Khai báo XML (XML Declaration)

- Thẻ khai báo xml

```
<?xml version="1.0" encoding="UTF-8"?>
```

- Trong đó:
  - *version* là phiên bản XML
  - *encoding* xác định mã hóa ký tự được sử dụng trong tài liệu

Tags và Elements

- XML file được cấu thành bởi nhiều phần tử XML (Element), còn được gọi là XML-node hay XML-tags. Tên các phần tử XML được bao trong các dấu < > như sau:

```
<element>....</element>
```

- Một phần tử XML có thể chứa nhiều phần tử XML khác như là con của nó, nhưng các phần tử con này phải không đè lên nhau.

SAI

```
<?xml version="1.0"?>
<contact-info>
<company> Bravo
</contact-info>
</company>
```

ĐÚNG

```
<?xml version="1.0"?>
<contact-info>
    <company>Bravo</company>
</contact-info>
```

- Một tài liệu XML chỉ có một phần tử gốc (Root Element).

```
<x>...</x>
<y>...</y>
```

SAI (bởi vì cả hai phần tử x và y xuất hiện ở cấp cao nhất mà không phải là một phần tử gốc)

ĐÚNG

```
<root>
   <x>...</x>
   <y>...</y>
</root>
```

- Tên của các phần tử XML là phân biệt kiểu chữ. Nghĩa là tên của thẻ mở và thẻ đóng phải cùng kiểu. Ví dụ, **<name>** là khác với **<Name>.**

Thuộc tính (Attributes)

- Sử dụng một cặp tên/giá trị *(name = "value")* để xác định thuộc tính cho phần tử. Một phần tử XML có thể có một hoặc nhiều thuộc tính. Ví dụ, ở đây *href* là tên thuộc tính và *https://www.w3schools.com/* là giá trị thuộc tính.

```
<a href="https://www.w3schools.com/">w3cshool</a>
```

- Chú ý:
  - Khác với HTML tên thuộc tính trong XML là phân biệt kiểu chữ: HREF và href là hai thuộc tính khác nhau trong XML.
  - Cùng một thuộc tính không thể có hai giá trị trong một cú pháp. Ví dụ cú pháp sai:

```
<a b="x" c="y" b="z">....</a>
```

  - Tên thuộc tính được định nghĩa không có sự trích dẫn, trong khi giá trị thuộc tính phải luôn luôn trong các dấu trích dẫn trong dấu "". Ví dụ cú pháp sai:

```
<a b=x>....</a>
```

Tham chiếu trong XML (References)

- Tham chiếu cho phép thêm hoặc bao phần text hay phần đánh dấu bổ sung trong một tài liệu XML. Các tham chiếu luôn luôn bắt đầu với biểu tượng "&" , đây là ký tự dành riêng và kết thúc với ký tự ";". XML có hai kiểu tham chiếu:
  - **Tham chiếu thực thể (Entity Reference):** Một tham chiếu thực thể chứa một tên giữa dấu tách mở và dấu tách đóng. **&** có *amp* là *tên*.
  - **Tham chiếu ký tự (Character Reference):** Chứa các tham chiếu, ví dụ **A**, chứa một dấu băm (#) được theo sau bởi một số. Số này luôn luôn tham chiếu tới mã hóa Unicode của ký tự. 65 tham chiếu tới chữ cái "A".

Text trong XML

- Tên của phần tử XML và thuộc tính XML là phân biệt kiểu chữ, nghĩa là tên của phần tử mở <> và phần tử đóng </> phải ở được viết cùng kiểu.
- Để tránh các vấn đề về mã hóa ký tự, tất cả XML file nên được lưu ở dạng Unicode UTF-8 hoặc UTF-16.
- Các ký tự whitespace như khoảng trắng, tab và ngắt dòng giữa các phần tử XML và giữa các thuộc tính XML sẽ bị bỏ qua.

Ký tự thay thế trong XML

Ký tự không được phép sử dụng	Thực thể thay thế trong XML	Mô tả
<	<	Nhỏ hơn
>	>	Lớn hơn
&	&	Và
'	'	Dấu nháy đơn
"	"	Trích dẫn kép

CDATA

- Tài liệu XML được xây dựng bởi các thẻ XML do lập trình viên tự định nghĩa ra, và đương nó phải tuân thủ các quy tắc của XML.
- Tuy nhiên có một số trường hợp dữ liệu chứa trong các thẻ XML lại có những ký tự đặc biệt dẫn đến tài liệu XML bị đảo lộn. Trong trường hợp này sử dụng CDATA – Character Data (Dữ liệu ký tự) trong XML.

Nội dung bên trong thẻ <Code> là một đoạn mã XML khác, có nghĩa đây là trường hợp giá trị của thẻ XML lại là một đoạn mã XML khác. Nếu chạy lên sẽ bị lỗi.

```
<?xml version="1.0" encoding="UTF-8"?>
<Tutorials>
    <Post>
        <Title>
            CDATA trong XML
        </Title>
        <Code>
            <?xml version="1.0" encoding="UTF-8"?>
            <Persons>
                <Person>
                    Nguyen Van A
                </Person>
            </Persons>
        </Code>
    </Post>
</Tutorials>
```

Đoạn mã đúng sử dụng CDATA

```
<?xml version="1.0" encoding="UTF-8"?>
<Tutorials>
    <Post>
        <Title>
            CDATA trong XML
        </Title>
        <Code>
            <![CDATA[
            <?xml version="1.0" encoding="UTF-8"?>
            <Persons>
                <Person>
                    Nguyen Van A
                </Person>
            </Persons>
            ]]>
        </Code>
    </Post>
</Tutorials>
```

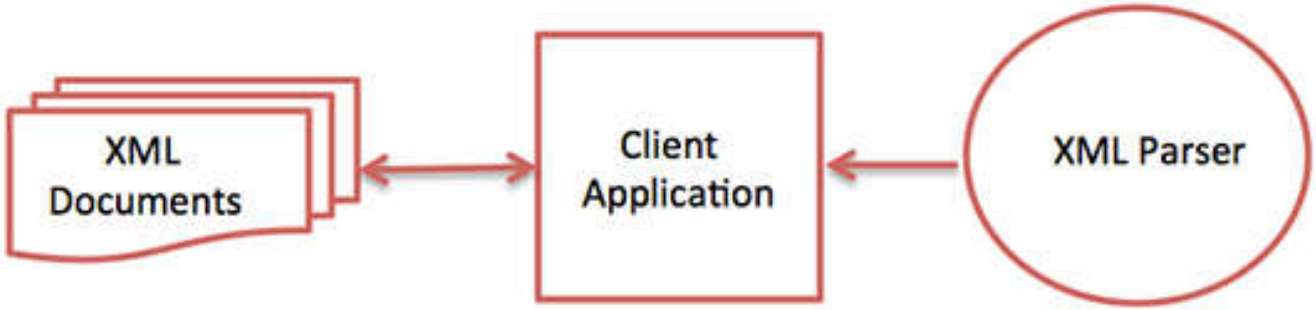
- Đoạn mã chạy đúng khi sử dụng chuyển đổi ký tự đặc biệt
- Cả hai cách trên đều dùng để xử lý nội dung bên trong thẻ XML, tuy nhiên chúng ta vẫn hay sử dụng CDATA bởi vì nó đơn giản nhưng lại rất hiệu quả.

```
<?xml version="1.0" encoding="UTF-8"?>
<Tutorials>
    <Post>
        <Title>
            CDATA trong XML
        </Title>
        <Code>
            &lt;?xml version="1.0" encoding="UTF-8"?&gt;
            &lt;Persons&gt;
                &lt;Person&gt;
                    Nguyen Van A
                &lt;/Person&gt;
            &lt;/Persons&gt;
        </Code>
    </Post>
</Tutorials>
```

XML parser

- XML Parser - trình phân tích XML là một thư việc phần mềm hoặc là một package mà cung cấp API cho các ứng dụng trên Client để làm việc với các tài liệu XML.
- Nhiệm vụ của XML Parser là đọc, kiểm tra định dạng thích hợp của tài liệu XML và xác nhận (validate) các tài liệu XML.
- Hiện nay, các trình duyệt hiện đại có XML Parser được xây dựng sẵn.

- Mục tiêu của một Parser là để biến đổi XML thành một code có thể đọc.



- Một số Parser được sử dụng phổ biến hiện nay:
  - **MSXML (Microsoft Core XML Services) :** Đây là một tập hợp các XML Tool chuẩn từ Microsoft, bao gồm một Parser.
  - **System.Xml.XmlDocument :** Lớp này là một phần của thư viện .NET, chứa một số lớp có liên quan trong khi làm việc với XML.
  - **Thư viện Java có Parser riêng:** Thư viện này được thiết kế để mà bạn có thể thay thế một Parser đã có sẵn với một triển khai ngoại vi như Xerces từ Apache hoặc Saxon.
  - **Saxon :** Saxon cung cấp các tool để phân tích, biến đổi, và truy vấn XML.
  - **Xerces :** Xerces được triển khai trong Java và được phát triển bởi mã nguồn mở nổi tiếng là Apache Software Foundation.

Phương pháp đọc & phân tích XML

- Có 2 phương pháp để đọc và phân tích XML bằng cách sử dụng:
  - DOM (Document Object Model)
  - SAX (Simple API XML).

XML DOM

- DOM (Document Object Model) là nền tảng của XML. Các tài liệu XML có một cấu trúc thứ bậc của các đơn vị thông tin được gọi là node;
- DOM là một cách để mô tả các node của nó và mối quan hệ giữa chúng với nhau. Hay nói cách khác DOM được xem như là *Tree Based*.
- Với một tài liệu XML, mô hình DOM sẽ duyệt và chuyển nó thành một mô hình cây của các Object.
- Nói cách khác, DOM là một API mà cho phép một lập trình viên có thể thêm, chỉnh sửa, di chuyển, hoặc xóa các node trong cây XML tại bất kỳ điểm nào để tạo một ứng dụng.

XML DOM trong Python

- Sử dụng module xml.dom trong python.
- Đối tượng minidom cung cấp một phương thức parser đơn giản mà tạo một DOM tree một cách nhanh chóng từ XML file.
- Hàm parse(file [,parser]) của đối tượng minidom để phân tích cú pháp XML file.

Tạo file staff.xml

```xml
<?xml version="1.0"?>
<company>
    <name>Mkyong Enterprise</name>
    <staff id="1001">
        <nickname>mkyong</nickname>
        <salary>100,000</salary>
    </staff>
    <staff id="1002">
        <nickname>yflow</nickname>
        <salary>200,000</salary>
    </staff>
    <staff id="1003">
        <nickname>alex</nickname>
        <salary>20,000</salary>
    </staff>
</company>
```

dom-staff.py

```python
from xml.dom.minidom import parse
import xml.dom.minidom

doc = xml.dom.minidom.parse("staff.xml")

# doc.getElementsByTagName returns NodeList
name = doc.getElementsByTagName("name")[0]
print(name.firstChild.data)

staffs = doc.getElementsByTagName("staff")
for staff in staffs:
    sid = staff.getAttribute("id")
    nickname = staff.getElementsByTagName("nickname")[0]
    salary = staff.getElementsByTagName("salary")[0]
    print("id:%s, nickname:%s, salary:%s" %
          (sid, nickname.firstChild.data,
salary.firstChild.data))
```

movies.xml

```
<collection shelf="Most Popular">
<movie title="Mission Impossible">
   <type>Action, Crime</type>
   <format>DVD</format>
   <year>2003</year>
   <rating>PG</rating>
   <stars>10</stars>
   <description>An action movie by Tom Cruise</description>
</movie>
<movie title="Rush Hour">
   <type>Action, Comedy</type>
   <format>DVD</format>
   <year>1989</year>
   <rating>R</rating>
   <stars>8</stars>
   <description>A comedy movie by Jackie Chan</description>
</movie>
   <movie title="Skyfall">
   <type>Action, Crime</type>
   <format>DVD</format>
   <episodes>4</episodes>
   <rating>PG</rating>
   <stars>10</stars>
   <description>An action movie by James Bond</description>
</movie>
</collection>
```

dom-movies.py

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

# Mo mot tai lieu XML document boi su dung minidom parser
DOMTree = xml.dom.minidom.parse("movies.xml")
collection = DOMTree.documentElement
if collection.hasAttribute("shelf"):
    print("Root element : ", collection.getAttribute("shelf"))
# Lay tat ca phim trong bo suu tap
movies = collection.getElementsByTagName("movie")
# in chi tiet ve moi phim.
for movie in movies:
    print("*****", movie, "*****")
    if movie.hasAttribute("title"):
        print("Ten Phim: ", movie.getAttribute("title"))

    type = movie.getElementsByTagName('type')[0]
    print("The loai: ", type.childNodes[0].data)
    format = movie.getElementsByTagName('format')[0]
    print("Dinh dang: ", format.childNodes[0].data)
    rating = movie.getElementsByTagName('rating')[0]
    print("Rating: ", rating.childNodes[0].data)
    description = movie.getElementsByTagName('description')[0]
    print("Gioi thieu: ", description.childNodes[0].data)
```

XML SAX

- SAX (Simple API for XML) là một API thuộc JAXP, nó cho phép xây dựng các ứng dụng truy xuất, phân tích các tài liệu XML theo cơ chế tuần tự và hướng sự kiện.
- Không như DOM, SAX phân tích dựa vào khái niệm sự kiện và phản hồi sự kiện, SAX không tạo ra một hình thức đại diện của tài liệu XML trong bộ nhớ, và cũng không sử dụng đối tượng để mô tả các thành phần trong file XML như DOM, vì thế nó hoạt động nhanh hơn, ít tốn bộ nhớ hơn.

- Xử lý dữ liệu ngay khi quá trình phân tích chưa xử lý xong toàn bộ tài liệu XML, dữ liệu sẽ được gửi đi thông qua các hàm Notification Method và sẽ bị xóa ngay sau đó, do đó SAX không thích hợp với các ứng dụng truy xuất ngẫu nhiên và thực hiện chỉnh sửa phức tạp.
- Khi SAX parser đã đi qua 1 element thì không thể quay trở lại element đó được nữa nên nếu cần lấy thông tin từ 1 element nhiều lần thì vẫn cần lưu thông tin đó trong bộ nhớ.

XML SAX trong Python

- Cần tạo riêng cho một ContentHandler là lớp con của xml.sax.ContentHandler.
- ContentHandler của sẽ xử lý các tag cụ thể và các thuộc tính của XML.
- Một đối tượng ContentHandler cung cấp các phương thức để xử lý các sự kiện parsing khác nhau.

Phương thức make_parser

- Phương thức make_parser tạo một đối tượng parser mới và trả về nó. Đối tượng parser đã được tạo này sẽ là kiểu parser đầu tiên mà hệ thống tìm thấy.

xml.sax.make_parser([parser_list])

- Tham số parser_list là tùy ý, bao gồm một danh sách các parser để sử dụng, tất cả phải triển khai phương thức make_parser.

Phương thức parse

- Phương thức này tạo một SAX parser và sử dụng nó để phân tích cú pháp một tài liệu.

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

- Chi tiết tham số:
  - xmlfile: Đây là tên của XML file để đọc từ đó.
  - contenthandler: Đây phải là một đối tượng ContentHandler.
  - errorhandler: Nếu được xác định, thì nó phải là một đối tượng SAX ErrorHandler.

Phương thức parseString

- Phương thức này cũng dùng để tạo một SAX parser và để phân tích cú pháp XML string đã cho.

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

- Chi tiết về tham số:
  - xmlstring: Là tên của XML string để đọc từ đó.
  - contenthandler: Phải là một đối tượng ContentHandler.
  - errorhandler: Nếu được xác định, thì nó phải là một đối tượng SAX ErrorHandler.

sax-movies.py

```
# Python XML Processing - Example Program
import xml.sax
class Saw(xml.sax.ContentHandler):
    def __init__(self):
        self.CurrentData = ""
        self.type = ""
        self.format = ""
        self.year = ""
        self.rating = ""
        self.stars = ""
        self.description = ""
    def startElement(self, tag, attributes):
        self.CurrentData = tag
        if tag == "movie":
            print("*****Movie*****")
            title = attributes["title"]
            print("Title:", title)
    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 = ""
```

sax-movies.py

```
    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__"):
    parser = xml.sax.make_parser()
    parser.setFeature(xml.sax.handler.feature_namespaces, 0)
    Handler = Saw()
    parser.setContentHandler(Handler)
    parser.parse("movies.xml")
```

Khác biệt giữa DOM và SAX parser

- DOM phân tích XML thông qua các Node trong Tree trong khi SAX thì dựa vào sự kiện và phản hồi sự kiện.
- DOM lưu và parse toàn bộ dữ liệu XML trong bộ nhớ trước khi parse trong khi SAX chỉ lưu một phần nhỏ, parse cho đến khi bạn dừng và xóa ngay sau khi gửi đi.
- DOM dễ sử dụng, không hạn chế về số lần thao tác trên một phần tử của DOM tree(Dễ dàng chuyển hướng), SAX thì phức tạp hơn và chỉ duyệt qua 1 lần các phần tử.
- SAX thích hợp hơn cho việc parse các file XML có kích thước lớn, vì không yêu cầu về bộ nhớ quá cao.
- DOM có thể đọc và ghi trên file XML trong khi SAX chỉ có thể đọc.
- DOM run time chậm hơn trong khi SAX nhanh hơn.

DOM hay SAX

- DOM thích hợp với những tập tin XML nhỏ hay cần nhiều các thao tác dữ liệu, cho phép bạn truy cập vào bất kỳ phần nào của tài liệu liên tục và cho phép bạn chỉnh sửa các cây DOM.
- Với SAX, lại thích hợp với dữ liệu đầu vào lớn, hơn thế nữa, nó chạy nhanh hơn và dễ dàng hơn để tìm hiểu hơn DOM vì API của nó là thực sự đơn giản. Nhưng từ quan điểm chức năng, nó cung cấp ít các chức năng hơn, người dùng phải tự "chăm sóc" nhiều hơn, chẳng hạn như tạo ra các cấu trúc dữ liệu của họ.

Bài tập thực hành

- a) Viết lớp initial.py trong đó có hàm create thực hiện tạo file input.xml bao gồm 3 bản ghi thông tin của một person, mỗi bản ghi bao gồm các thông tin sau: tên, email, mobile.
- b) Viết lớp analysic.py trong đó có hàm reading thực hiện đọc file input.xml vừa tạo và in ra màn hình dữ liệu của 3 bản ghi đó.
- c) Viết lớp main.py khởi tạo 2 đối tượng ở câu a và b. Sau đó gọi lần lượt đến hàm create, reading để in ra các kết quả.

❖THANKS FOR ATTENDING!