XML Parsing

Markup Language

- 개요
  - 원래 마크업(markup)이란 신문사나 잡지사의 교정 기자들이 쓰는 특수 목적의 표기법으로, 문서의 논리적 구조와 배치 양식에 대한 정보를 표현하는 언어를 말한다.
  - 그 파일이 프린터로 출력되거나 화면에서 어떻게 보여야 할 것인지를 나타내기 위해 또는 그 문서의 논리적인 구조를 묘사하기 위해서, 텍스트나 워드프로세싱 파일의 특정위치에 삽입되는 일련의 문자들이나 기호들을 말한다.
  - 마크업에 사용되는 표지를 흔히 '태그'(Tag)라고 부른다.
  - 예를 들어, HTML, XML

XML

개요

- XML eXtensible Markup Languagee 은 1996년 W3C가 제안한 웹 문서 표준 형식으로 SGML의 하위셋
- 마크업 언어인 SGML Standard Generalized Markup Language 은 1969년 미국 IBM이 만든 것으로 1986년 ISO 표준으로 제정
- HTML과 달리 웹 상에서 구조화된 문서를 전송 가능하도록 설계됨
- 확장자: .xml
- 데이터에 의미를 부여하는 메타데이터를 기술할 수 있음

  [예] “CPU 2.83GHz”:

  - HTML: 데이터 명과 실제 데이터 구분 표시가 불가능
  - XML: <dataname>CPU</dataname>과 <datavalue>2.83</datavalue>로 구분 가능

XML 구성

- 용어
  - 태그tag
    - 시작 태그: (예) <section>
    - 종료 태그: (예) </section>
    - 빈 요소empty-element 태그: (예) <line-break/>
  - 요소element
    - 시작 태그start tag <태그명> 과 종료 태그end tag </태그명> 및 두 태그 사이의 내용
    - 빈 요소 태그
    - 시작 태그와 종료 태그 사이의 내용은 다른 자식 요소를 포함할 수 있다.
  - 속성attribute
    - 이름/값 짝으로 이루어진 마크업 구조로 시작 태그 또는 빈 엘리먼트 태그 속에 위치한다.
    - (예) 에서 img 태그는 src와 alt의 두 속성을 갖는다.

XML 예제

```
<?xml version="1.0"?>
<data>
    <country name="Liechtenstein">
        <rank>1</rank>
        <year>2008</year>
        <gdppc>141100</gdppc>
        <neighbor name="Austria" direction="E"/>
        <neighbor name="Switzerland" direction="W"/>
    </country>
    <country name="Singapore">
        <rank>4</rank>
        <year>2011</year>
        <gdppc>59900</gdppc>
        <neighbor name="Malaysia" direction="N"/>
    </country>
    <country name="Panama">
        <rank>68</rank>
        <year>2011</year>
        <gdppc>13600</gdppc>
        <neighbor name="Costa Rica" direction="W"/>
        <neighbor name="Colombia" direction="E"/>
    </country>
</data>
```

XML Parser 선택문제

- **XML 파싱 기법**
  - XML 파싱 기법 선택 -> 구현 용이, 성능
  - DOM, SAX, ElementTree 등
- **표준 및 구현**
  - Standard : DOM의 경우 W3C
  - Implementation -> Library 선택 (Paper, Essay등 연구결과)
- **XML Parsing Library**
  - XML parsing 라이브러리는 아주 많음
  - xml.etree.ElementTree module, lxml 등

XML 파싱 기법

- **DOM 방식**
  - reads the XML into memory and converts it to objects
  - 메모리 사용량이 크지만, 속도는 빠른 편
  - 자바스크립트 등 웹에서 주로 사용

- **SAX 방식**
  - a stream parser, with an event-driven API
  - 자료를 하나 하나 읽어가며 파싱하여 메모리 사용량 적지만, 속도는 느린 편
  - 주로 스마트폰 등의 응용프로그램에서 사용
  - DOM, SAX 등 비교 : https://wiki.python.org/moin/PythonXml

- **ElementTree 방식 선택**
  - a simple-to-use and very fast XML tree library
  - Elementtree가 DOM이나 SAX보다 좋은 이유

  -Essay: https://eli.thegreenplace.net/2012/03/15/processing-xml-in-python-with-elementtree/

XML 표준 및 구현

- **Standard**
  - Dom 방식을 선택한 경우, W3C 표준에 따라 개발
  - ElementTree API 에 따라 개발

    http://omz-software.com/pythonista/docs/library/xml.etree.elementtree.html
  - 예) find(path) : Finds the first matching subelement, by tag name or path

- 구현 선택
  - XML parsing 라이브러리

```
import elementtree.ElementTree as ET
import cElementTree as ET
import lxml.etree as ET
import xml.etree.ElementTree as ET # Python 2.5
```

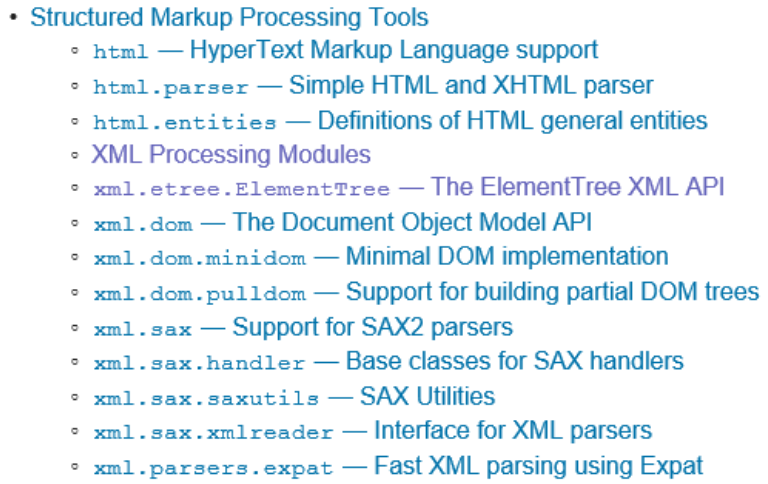
  - v2.5이후 파이썬 표준라이브러리에 구현된 xml.etree.ElementTree 선택

XML Parsing Library

- **lxml**
  - lxml은 XML을 빠르고 유연하게 처리하는 외부 라이브러리

    - lxml project site : https://lxml.de/
  - Xpath XML Path Language와 XSLT Extensible Stylesheet Language Transformation 를 지원하며, 많이 쓰이는 ElementTree API도 구현

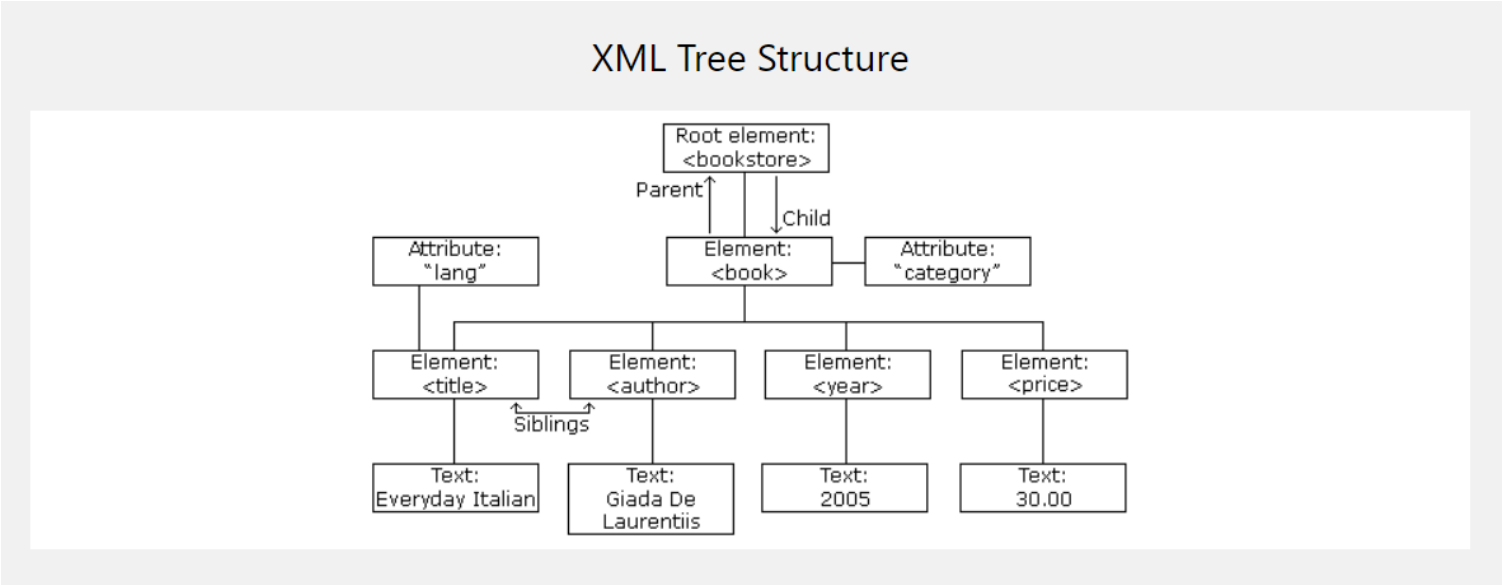
- **Python Standard Library**
  - xml.etree.ElementTree 모듈
  - xml.dom 모듈 (Document Object Model)
  - xml.sax 모듈 (Simple API for XML)



- Structured Markup Processing Tools
  - `html` — HyperText Markup Language support
  - `html.parser` — Simple HTML and XHTML parser
  - `html.entities` — Definitions of HTML general entities
  - XML Processing Modules
  - `xml.etree.ElementTree` — The ElementTree XML API
  - `xml.dom` — The Document Object Model API
  - `xml.dom.minidom` — Minimal DOM implementation
  - `xml.dom.pulldom` — Support for building partial DOM trees
  - `xml.sax` — Support for SAX2 parsers
  - `xml.sax.handler` — Base classes for SAX handlers
  - `xml.sax.saxutils` — SAX Utilities
  - `xml.sax.xmlreader` — Interface for XML parsers
  - `xml.parsers.expat` — Fast XML parsing using Expat

XML Tree

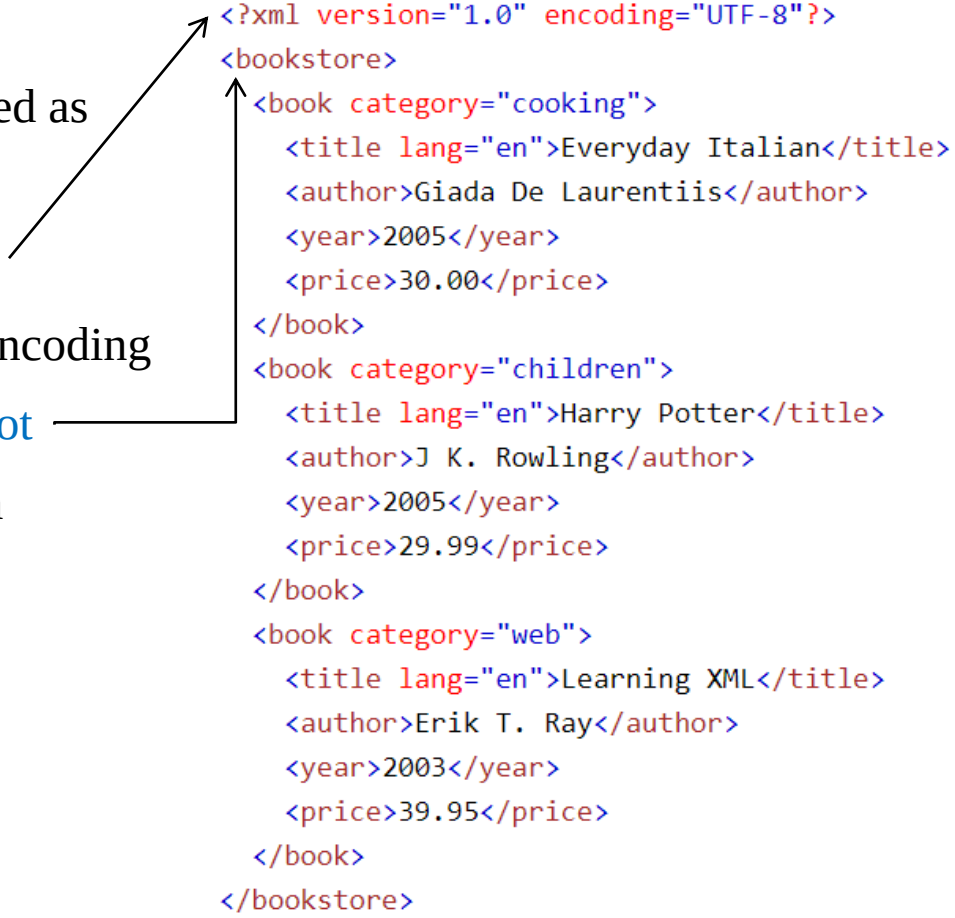
- **XML Tree**
  - XML documents form a tree structure that starts at "the root" and branches to "the leaves". cf. 자료구조 Tree



https://www.w3schools.com/XML/xml_tree.asp

XML Document

- vs. XML element tree
  - XML documents are formed as element trees.
  - A prolog defines the XML version and the character encoding
  - An XML tree starts at a root element and branches from the root to child elements.
  - All elements can have sub elements (child elements)



```
<?xml version="1.0" encoding="UTF-8"?>
<bookstore>
  <book category="cooking">
    <title lang="en">Everyday Italian</title>
    <author>Giada De Laurentiis</author>
    <year>2005</year>
    <price>30.00</price>
  </book>
  <book category="children">
    <title lang="en">Harry Potter</title>
    <author>J K. Rowling</author>
    <year>2005</year>
    <price>29.99</price>
  </book>
  <book category="web">
    <title lang="en">Learning XML</title>
    <author>Erik T. Ray</author>
    <year>2003</year>
    <price>39.95</price>
  </book>
</bookstore>
```

XML Element Tree

- XML element tree
  - The terms parent, child, and sibling are used to describe the relationships between elements.
  - Parents have children. Children have parents. Siblings are children on the same level (brothers and sisters)
  - All elements can have text content (Harry Potter) and attributes (category="children") -> XML의 2종류 데이터 저장공간
  - The <book> elements have 4 child elements: <title>, <author>, <year>, <price>

```
<book category="children">
  <title lang="en">Harry Potter</title>
  <author>J K. Rowling</author>
  <year>2005</year>
  <price>29.99</price>
</book>
```

실험 데이터 세트

- 교재 파일 [1] ref. chap3
  - github : https://github.com/jackiekazil/data-wrangling -> data-text.xml

- 실험 데이터 Experiment Data
  - data-text.xml (github) : 국가별 기대수명 데이터

XML Parsing

- **XML Parsing**

```
#xml.etree.ElementTree 모듈 사용
import xml.etree.ElementTree as ET

# parse xml file
tree = ET.parse('data-text.xml')   #tree : 전체 XML 객체

# get root node
root = tree.getroot()   # root는 xml문서의 최상단 루트 태그를 가리킴
```

- 유용한 도구
  - **dir( )** : 해당 데이터 유형의 메소드와 속성 확인

Tag 속성

- **ElementTree의 XML Tag의 속성**
  - **.tag** : 해당 태그의 이름
    - root의 tag는 “GHO“

      root.tag #tag속성
  - **.text** : 해당 태그의 내용
    - Disclaimer/Display태그의 text는 'The information in this database is … to this section.‘
  - **.attrib** : 해당 노드의 attribute 맵 (key, value)
    - 'Data/Observation/Dim' 태그는 다음의 attrib 맵을 가짐

      {'Category': 'PUBLISHSTATE', 'Code': 'PUBLISHED'}

Tag find()메소드

- **ElementTree의 특정 태그 찾기**

```
# root 하위의 Disclaimer의 자식 중에 “Display"와 일치하는 첫번째 태그를 찾아서 리턴   ( 없으면 None을 리턴 )
root.find('Disclaimer/Display')

# root 하위의 Data/Observation의 자식 중에 “Dim“과 일치하는 모든 태그를 리스트로 리턴
list_dim = root.findall('Data/Observation/Dim')

# root 하위의 Disclaimer의 자식 중에 “Display"와 일치하는 첫번째 태그를 찾아서 해당 태그의 text를 리턴
root.findtext('Disclaimer/Display‘)

# findtext는 find().text와 동일하다
root.find('Disclaimer/Display').text
```

  - 위 **find** 함수들은 **root**의 자식인 **Disclaimer**등에 대해서만 탐색이 가능하다.
  - 손자인 **Display**의 경우, **'Disclaimer/Display**로 지정하면 탐색 가능

Tag iter()메소드

- **iter( ) 함수**
  - **find**는 명령을 수행하는 태그의 바로 자식만 탐색이 가능하지만, **iter** 함수는 모든 자식, 손자 등에 대해 탐색이 가능하다.
  - 태그의 모든 자식, 손자 등을 순회하고자 할 때는 **iter** 함수를 **tag=None**으로 사용한다.

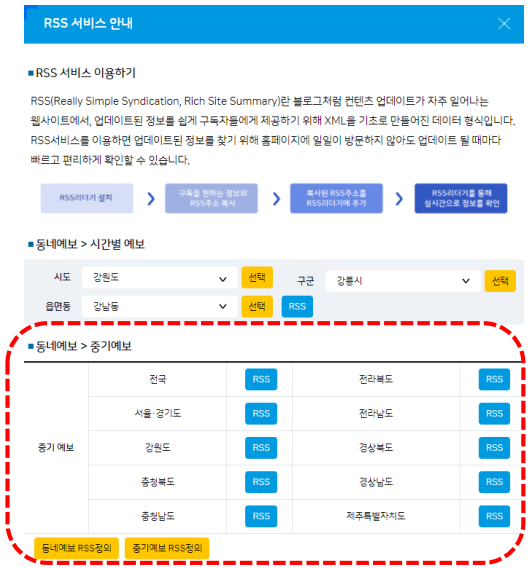
```
# root 태그에서도 iter('Dim')은 모두 순회해서 'Dim' 탐색 가능
for dim in root.iter('Dim'):
    print(dim.attrib)

# root 이하 모든 자식, 손자 등 모두 순회가 가능
for child in root.iter():
    print(child.tag)
```

RSS

- **really simple syndication (RSS)**
  - 데이트 업데이트가 빈번한 웹사이트의 정보를 사용자에게 보다 쉽게 제공하기 위하여 만들어진 XML 기반의 콘텐츠 배급 포맷
  - RSS는 넷스케이프(Netscape)에서 신문기사를 손쉽게 제공하기 위하여 시작
  - 설치형과 웹 기반형이 있는데, 간단한 계정 등록으로 어디에서든 이용할 수 있는 웹 기반형이 더 많이 이용
  - 사이트에서 제공하는 주소를 RSS 리더(RSS reader) 프로그램에 등록하면, PC나 휴대폰 등을 통하여 자동으로 전송된 콘텐츠를 이용할 수 있음

기상청 RSS 데이터

- **XML 데이터 받기**
  - https://www.weather.go.kr/w/pop/rss-guide.do
  - **RSS 단추 눌러서 XML 파일 복사 후 저장하기 (Chrome사용)**
    (확장자 .rss를 .xml로 변경)



참고도서

- **reference**

[1] 파이썬을 활용한 데이터길들이기

- 프로그래밍인사이트

[2] 모두의 데이터분석

- 길벗

[3] 파이썬머신러닝 판다스데이터분석

- 정보문화사

End