4-9. 여러가지 파일 포맷 다루기 (2) XML 파일

XML

XML은 Extensible Markup Language의 약자로, 다목적 마크업 언어입니다.

기존에 HTML을 접한 적이 있다면 마크업 언어가 익숙하실 수 있을텐데요. 인터넷 웹상에서 문서 즉, 내용을 교환할 때 이러한 마크업 언어를 이용합니다. 마크업 언어를 아주 간단하게 말한다면, 태그라고 불리는 꺽쇠 모양의 괄호(`<>`)로 구분된 언어라고 할 수 있습니다. API에서 데이터를 요청하고 저장할 때 `XML` 이나 `JSON` 형식을 이용해 데이터를 교환한답니다. 간단한 XML 파일을 한번 살펴 볼까요?

```
<Person>
    <Name>이펠</Name>
    <Age>28</Age>
    <Place>강남</Place>
</Person>

#- 꺽쇠 괄호 안에 태그 이름을 정의하고, 태그 사이에 데이터를 기록하는 형식입니다.
#- 참고로, 태그 이름은 사용자가 마음대로 지정할 수 있습니다.
```

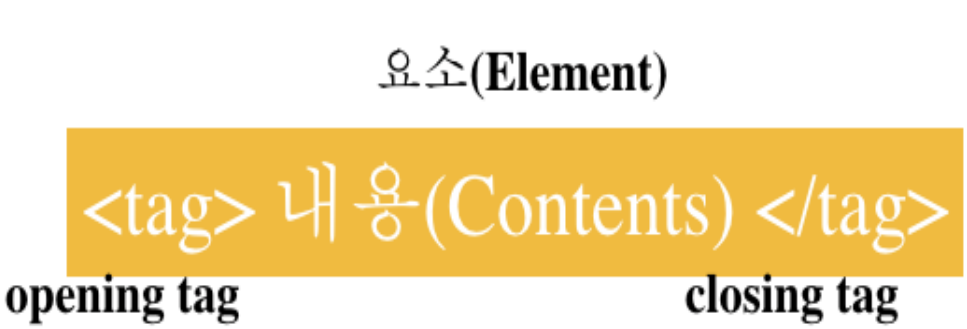
WHO athena API GHO

아래 예시는 세계보건기구(WHO)에서 제공하는 데이터 API 정보의 XML 파일 일부를 추출한 것입니다. XML 파일이 어떤 식으로 구성이 되어 있는지 가볍게 살펴본 후 넘어가 보도록 하겠습니다. 전체 파일은 여기에서 다운로드 하실 수 있습니다. 단, 웹사이트에서 다운로드 할 경우 데이터의 내용이 업데이트 될 수 있으므로 이 점 유의해주세요.

```
<?xml version="1.0" encoding="utf-8"?>
#- XML의 버전과 인코딩을 명시하는 태그입니다. *필수!
<Workbook xmlns="urn:schemas-microsoft-com:office:spreadsheet"
```

```
        xmlns:o="urn:schemas-microsoft-com:office"
        xmlns:x="urn:schemas-microsoft-com:excel"
        xmlns:ss="urn:schemas-microsoft-com:spreadsheet"
        xmlns:html="http://www.w3.org/TR/REC-html40">
<DocumentProperties/>
<Styles>
   <Style ss:ID="Hyperlink" ss:Name="Hyperlink">
      <Font ss:Color="#0000FF" ss:Underline="Single"/>
   </Style>
   <Style ss:ID="header">
      <Alignment ss:Vertical="Bottom" ss:WrapText="1"/>
      <NumberFormat ss:Format="@"/>
   </Style>
</Styles>
<Worksheet ss:Name="notice">
   <Table>
      <Row>
         <Cell>
            <Data ss:Type="String">Notice</Data>
         </Cell>
      </Row>
      <Row/>
      <Row>
         <Cell>
            <Data ss:Type="String">Date generated</Data>
         </Cell>
      </Row>
```

XML 파일을 훑어보니 꺾쇠(**<글자>**)가 정말 많이 보이죠? 이 부분을 **태그**(tag)라고 부르는데요. 위 파일에는 **<Workbook>**, **<Data>** 등의 태그들로 구성되어 있음을 확인할 수 있습니다. 태그는 기본적으로 **<태그> 내용 </태그>** 형태로 구성되어 있지만 아래의 오른쪽 그림처럼 태그에 **속성**(attribute) 값이 포함될 수도 있습니다.



자, 하나만 더 짚고 넘어 가볼게요. 상위 태그 안에 하위 태그가 속해 있을 수 있는데요. 이런 관계를 부모-자식 관계라고 합니다. 위에서 살펴 보았던 예시에서 그 유형을 찾아 보겠습니다. 아래를 보시면 `<Table>` 태그는 `<Row>` , `<Cell>` , 그리고 `<Data>` 태그를 자식 태그로 가지고 있네요.

```
<Table>
      <Row>
         <Cell>
            <Data ss:Type="String">Notice</Data>
         </Cell>
      </Row>
```

자, 지금까지 배운 내용들을 간략히 정리해 보겠습니다.

1. XML은 다목적 마크업 언어(Extensible Markup Language)이다.
2. 마크업 언어는 태그(tag)로 이루어진 언어를 말하며, 상위(부모)태그 - 하위(자식)태그의 계층적 구조를 가지고 있다.
3. XML은 요소(element)들로 이루어져 있다.
4. 요소는 `<열린태그> 내용 </닫힌태그>` 가 기본적인 구조이며, 속성(attribute)값을 가질 수도 있다.

XML 파일 만들기

자, 그럼 지금부터 간단한 XML 파일을 만들어 보겠습니다.

ElementTree

파이썬 표준 라이브러리인 ElementTree는 XML 관련 기능을 다음과 같이 제공합니다.

- `Element()` : 태그 생성
- `SubElement()` : 자식 태그 생성
- `tag` : 태그 이름
- `text` : 텍스트 내용 생성
- `attrib` : 속성 생성

dump()

생성된 XML 요소 구조를 시스템(sys.stdout)에 사용합니다.

- `write()` : XML 파일로 저장
- 리스트(list)와 유사한 메소드를 제공
    - append, insert, remove, pop

아래는 맨 처음에 사용한 person XML을 만드는 코드입니다.

> **[Input]**
>
> ```
> import xml.etree.ElementTree as ET
>
> person = ET.Element("Person")
> name = ET.Element("name")
> name.text = "이펠"
> person.append(name)
> ```

```
age = ET.Element("age")
age.text = "28"
person.append(age)

ET.SubElement(person, 'place').text = '강남'

ET.dump(person)
```

실행 ▶

[Output]

다음과 같은 XML 파일이 만들어 집니다.

```
<Person>
<name>이펠</name>
<age>28</age>
<place>강남</place>
</Person>
```

속성값은 **`attrib`** 란 메소드를, name 태그명은 **`tag`** 메소드를 이용해 변경할 수 있습니다.

[Input]

```
person.attrib["id"] = "0x0001"
name.tag = "firstname"
ET.dump(person)
```

실행 ▶

[Output]

```
<Person id="0x0001">
<firstname>이펠</firstname>
<age>28</age>
<place>강남</place>
</Person>
```

이번에는 새롭게 lastname이라는 태그를 firstname 태그 다음으로 삽입하고 속성에 date를 넣어 보겠습니다.

[Input]

```
lastname = ET.Element('lastname', date='2020-03-20')
lastname.text = '아'
person.insert(0,lastname)
ET.dump(person)
```

실행

[Output]

```
<Person id="0x0001">
<firstname>이펠</firstname>
<lastname date='2020-03-20'>아</lastname>
```

속성은 태그를 생성하는 함수인 **`Element`** 의 인자로 넣어 주어도 됩니다. firstname 태그의 다음이니까 인덱스 번호는 1번이 되겠네요. **`insert`** 를 이용해 인덱스 번호와 태그 이름을 삽입했습니다. 리스트의 **`insert`** 와 형태가 동일하지요?

```
<age>28</age>
<place>강남</place>
</Person>
```

삭제는 `remove` 혹은 `pop` 을 이용하면 됩니다. age 태그를 지우고 싶다면 다음과 같이 하면 되겠습니다.

[Input]

```
person.remove(age)
```

실행

[Output]

자, 이제 마지막으로 XML 파일로 저장을 해 보겠습니다.

[Input]

```
ET.ElementTree(person).write('person.xml')
```

실행

[Output]

`person.xml` 파일이 잘 저장되었나요?

XML 파싱하기

그럼 이제 본격적으로 XML 파일을 파싱해 보도록 하겠습니다.

파이썬에서는 그 방법으로 크게 2가지를 제공하는데요. 위에서 살펴 본 ElementTree가 첫 번째이고, 다른 하나가 바로 BeautifulSoup 라이브러리입니다. 지금부터 이 BeautifulSoup을 이용해 파싱을 시도해 보도록 하겠습니다.

- 예제 파일 : books.xml

(1) 설치

BeautifulSoup, lxml은 표준 라이브러리가 아니기 때문에 pip install로 설치합니다.

```
pip install beautifulsoup4
pip install lxml
```

(2) books.xml 파일의 "title" 태그의 내용만 가져오기

방금 다운로드 받은 books.xml 파일을 저장한 후, 해당 파일의 title 태그에 속한 내용만 가져오는 코드를 작성해 보겠습니다.

[Input]

```
from bs4 import BeautifulSoup
with open("books.xml", "r", encoding='utf8') as f:
    booksxml = f.read()
    #- 파일을 문자열로 읽기

soup = BeautifulSoup(booksxml,'lxml')
#- BeautifulSoup 객체 생성 : lxml parser를 이용해 데이터 분석

for title in soup.find_all('title'):
#-  태그를 찾는 find_all 함수 이용
    print(title.get_text())
```

실행

[Output]

BeautifulSoup의 더욱 다양한 메소드들을 알고 싶다면 링크를 확인해 보세요!