



XML과 JSON

파이썬 프로그래밍

목차

1. XML의 이해
2. Lab: XML 파싱
3. JSON의 이해
4. Lab: JSON 데이터 분석





01

XML의 이해



01. XML의 이해

■ XML의 개념

- XML(eXtensible Markup Language)은 확장적인 마크업(markup) 언어라는 뜻으로, 데이터의 구조와 의미를 설명하는 태그를 사용하여 어떤 데이터의 속성과 값을 표현하는 언어이다. 즉, 시작 태그와 종료 태그 사이에 어떤 값이 있고, 그 값은 태그의 이름으로 만들어진 속성에 대한 값이 된다.



01. XML의 이해

■ XML 표현하기

- XML의 구조는 다음과 같이 간단하다

```
<?xml version="1.0"?>
<학생>
  <이름>한재일</이름>
  <학번>20105503</학번>
  <나이>26</나이>
  <학과>산업경영공학과</학과>
  <성별>남성</성별>
</학생>
```



01. XML의 이해

여기서 잠깐! 현재 XML 상황

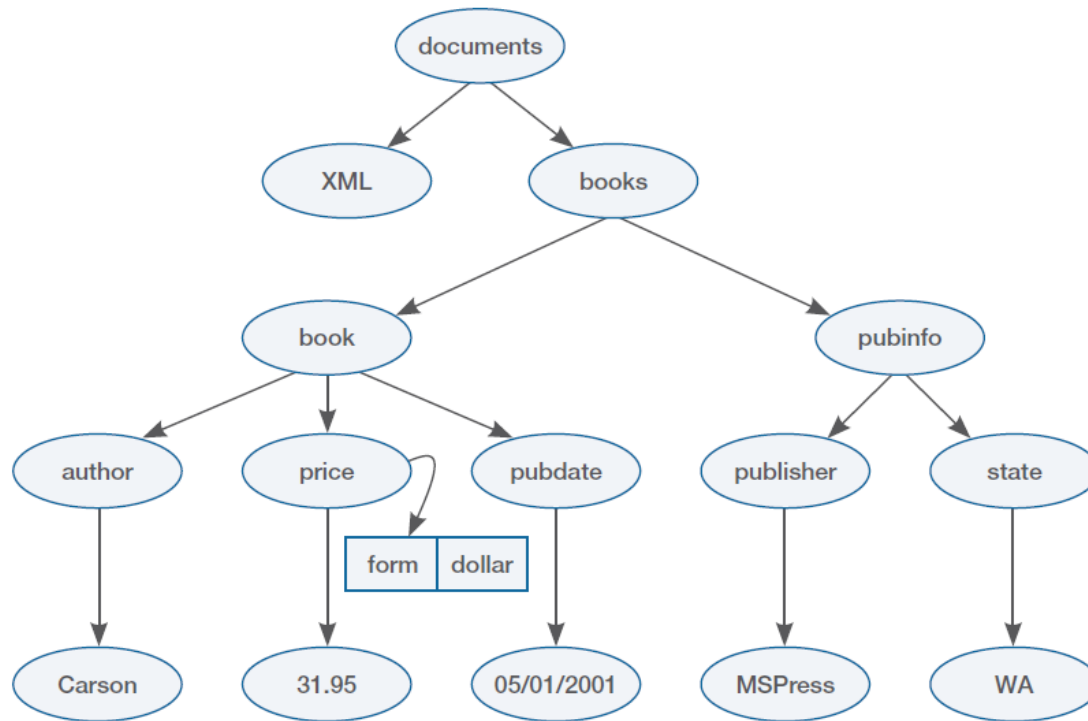
- 현재 XML은 10년 전보다 그 중요성이 매우 떨어졌다. 가장 큰 이유는 XML이 데이터베이스보다 좀 더 자유롭게 데이터를 저장하기 위해 만들어졌는데, 이 기능이 최근 많이 사용하는 JSON보다 훨씬 무겁기 때문이다. 그래도 여전히 기존의 오래된 시스템, 흔히 레거시(legacy)라고 하는 시스템에서는 기기 간 또는 컴퓨터와 스마트폰 등의 다른 기기 간의 정보를 XML로 주고받고 있다.



01. XML의 이해

■ XML 문서

- XML로 정보를 표현할 때 가장 기본적인 방법은 트리 형태로 표현하는 것이다. 이는 HTML과 완전히 같으며 모든 태그 기반의 언어가 지닌 공통적인 특징이다.



[트리 형태의 데이터 표현(XML)]



01. XML의 이해

■ XML 문서

- 그림의 구조적인 정보를 XML로 나타내면 다음과 같다.

```
<?xml version="1.0"?>
  <books>
    <book>
      <author>Carson</author>
      <price format="dollar">31.95</price>
      <pubdate>05/01/2001</pubdate>
    </book>
    <pubinfo>
      <publisher>MSPress</publisher>
      <state>WA</state>
    </pubinfo>
  </books>
```



01. XML의 이해

■ XML 문서

- 간단히 딕셔너리로 생각하면 다음과 같은 방식으로 표현할 수 있다.

```
{books : [{book : {author: carson, price: 31.95, pubdate: 05/01/2001} ] }}
```





02

Lab: XML 파싱



02. Lab: XML 파싱

■ BeautifulSoup 모듈 개요

- BeautifulSoup 모듈은 일종의 래퍼(wrapper)로, 기존 파싱 기능이 있는 다른 라이브러리를 쉽게 사용할 수 있도록 한다. 전통적인 파이썬 XML 파서(XML parser)에는 lxml과 html5lib 등이 있으며, BeautifulSoup 모듈은 이를 차용하여 데이터를 쉽고 빠르게 처리한다.

파서	파이썬 2.7		파이썬 3.2	
	속도(KB/s)	성공률(%)	속도(KB/s)	성공률(%)
BeautifulSoup 3.2(SGMLParser)	221	100	—	—
html5lib(BS3 treebuilder)	253	99	—	—
BeautifulSoup 4.0 + lxml	255	100	2140	96
html5lib(lxml treebuilder)	270	99	—	—
BeautifulSoup 4.0 + html5lib	271	98	—	—
BeautifulSoup 4.0 + HTMLParser	299	59	1705	57
html5lib(simpлетree treebuilder)	332	100	—	—
HTMLParser	5194	52	3918	57
lxml	179252	100	14258	96

[파서의 성능 비교]



02. Lab: XML 파싱

■ BeautifulSoup 모듈 설치

- [시작] – [Windows 시스템] – [명령 프롬프트] 관리자 권한으로 실행
- **pip install beautifulsoup4**
- * 3.4 이후 버전은 pip(파이썬 패키지 관리 시스템) 기본 포함
- 이후 모듈이 설치되는 모든 과정을 거친 후 파이썬 셸에서 다음을 실행하여 이상이 없다면, 정상적으로 설치가 완료된 것이다.

```
>>> from bs4 import BeautifulSoup
```



02. Lab: XML 파싱

■ BeautifulSoup 모듈 사용법

목적	코드	설명
객체 생성	<code>soup = BeautifulSoup(books_xml, "lxml")</code>	xml 문서를 분석하는 새로운 객체를 생성, books_xml은 문자열형, lxml은 파서의 이름
태그 검색	<code>soup.find_all("author")</code>	필요한 태그를 검색하여 여러 개를 반환하는 함수, 여기서는 'author'라는 이름의 태그를 검색하여 반환

[BeautifulSoup 모듈의 주요 코드]



02. Lab: XML 파싱

■ BeautifulSoup 모듈 사용법

- BeautifulSoup 모듈을 사용하는 코드를 만들기 위해 소스 파일에서 'books.xml'을 작업 폴더에 가져오고, [코드 15-1]을 작성해 보자.

코드 15-1 beautifulsoup1.py

```
1 from bs4 import BeautifulSoup
2
3 with open("books.xml", "r", encoding = "utf8") as books_file:
4     books_xml = books_file.read()           # 파일을 문자열로 읽어 오기
5
6 soup = BeautifulSoup(books_xml, "lxml")    # lxml 파서를 사용해 데이터 분석
7                                           pip install lxml
8 # author가 들어간 모든 요소의 값 추출
9 for book_info in soup.find_all("author"):
10     print(book_info)
11     print(book_info.get_text())           # 해당 요소에서 값 추출
```



02. Lab: XML 파싱

■ BeautifulSoup 모듈 사용법

```
<author>Carson</author>  
Carson  
<author>Sungchul</author>  
Sungchul
```

- ➔ 다운로드한 파일의 내용을 3~4행에서 읽어 와 문자열 파일로 변환한다. 그리고 6행에서 해당 문자열 정보를 받아오면서 lxml을 사용해 파싱한 BeautifulSoup의 객체를 생성한다. 9행의 find_all() 함수를 사용하여 author가 포함된 요소들을 받아 오고, 마지막으로 해당 요소에서 get_text() 함수를 사용해 해당 값의 결과를 출력한다.



02. Lab: XML 파싱

■ USPTO XML 데이터

- 미국 특허청(USPTO)의 특허 데이터는 XML로 제공된다. 이번 Lab에서는 특정 특허 XML 문서로부터 필요한 정보를 가져오겠다. 분석할 특허는 등록번호 '08621662'인 'Adjustable shoulder device for hard upper torso suit'이며, 다음 링크에 접속하면 자세히 볼 수 있다.

<https://patents.google.com/patent/US20120260387>


코드 15-2 beautifulsoup2.py

```
1 import urllib.request
2 from bs4 import BeautifulSoup
3
4 with open("US08621662-20140107.XML", "r", encoding="utf8") as patent_xml:
5     xml = patent_xml.read()           # 파일을 문자열로 읽어 오기
6
7 soup = BeautifulSoup(xml, "lxml")    # lxml 파서 호출
8
9 # invention-title 태그 찾기
```


02. Lab: XML 파싱

■ USPTO XML 데이터

```
10 invention_title_tag = soup.find("invention-title")
11 print(invention_title_tag.get_text())
```



Adjustable shoulder device for hard upper torso suit

- ➡ 5행에서 파일을 읽어 와 문자열로 변환한다. 7행에서 BeautifulSoup 객체를 생성하고, 10~11행에서 invention-title만 추출할 수 있도록 find() 함수를 사용하여 필요한 정보를 추출한다.



02. Lab: XML 파싱

■ USPTO XML 데이터

- 이 태그들은 각각 publication-reference와 application-reference의 하위 태그이며, 둘 모두 이름이 같다. 이 경우 어떻게 처리해야 할까?

```
<publication-reference>                                # 등록 관련 정보
  <document-id>
  <country>US</country>
  <doc-number>08621662</doc-number>                    # 등록번호
  <kind>B2</kind>                                       # 상태
  <date>20140107</date>                                # 등록일자
</document-id>
</publication-reference>
<application-referenceappl-type="utility">            # 출원 관련 정보
  <document-id>
  <country>US</country>
  <doc-number>13175987</doc-number>                    # 출원 번호
  <date>20110705</date>                                # 출원일
  </document-id>
</application-reference>
```



02. Lab: XML 파싱

■ USPTO XML 데이터

- 정규 표현식에서 배웠듯이 구조적으로 두 번 접근하는 방식을 사용한다. 다음과 같이 먼저 publication-reference에 접근하여 요솟값을 획득한 후, 해당 태그 정보에서 다시 필요한 정보를 추출할 수 있다. 다음 코드를 [코드 15-2] 뒷부분에 이어 입력하면 문제를 해결할 수 있다.

```
publication_reference_tag = soup.find("publication-reference")
p_document_id_tag = publication_reference_tag.find("document-id")
p_country = p_document_id_tag.find("country").get_text()
p_doc_number = p_document_id_tag.find("doc-number").get_text()
p_kind = p_document_id_tag.find("kind").get_text()
p_date = p_document_id_tag.find("date").get_text()

application_reference_tag = soup.find("application-reference")
a_document_id_tag = application_reference_tag.find("document-id")
a_country = a_document_id_tag.find("country").get_text()
a_doc_number = a_document_id_tag.find("doc-number").get_text()
a_date = a_document_id_tag.find("date").get_text()
```

03

JSON의 이해



03. JSON의 이해

■ JSON의 개념

- JSON은 XML보다 데이터 용량이 적고 코드로의 전환이 쉽다는 측면에서 XML의 대체재로 가장 많이 활용되고 있다.
- JSON은 파이썬의 딕셔너리형과 매우 비슷하여, 키-값의 쌍으로 구성되어 있다.

```
{  
  "dataTitle":"JSON Tutorial!",  
  "swiftVersion":2.1  
  
  "users":[  
    {  
      "name":"John",  
      "age":25  
    },  
    {  
      "name":"Mark",  
      "age":29  
    },  
  ],  
}
```



03. JSON의 이해

■ JSON의 개념

```
{  
  "name":"Sarah",  
  "age":22  
},  
}
```

- ➡ 'users'라는 키에는 값으로 리스트형이 있고, 그 안에 'name'과 'age'라는 2개의 키가 또 하나의 딕셔너리형으로 있는 것을 확인할 수 있다.



03. JSON의 이해

■ JSON과 XML

- XML과 비교할 때 JSON의 장점은 일단 코드가 간결하고, 코드의 전환이 쉽다는 점이다. 그리고 코드의 간결함 때문에 용량의 절약이라는 가장 큰 장점이 있다.

Json

```
{  
  "sibling": [  
    {"firstName": "Anna", "lastName": "Clayton"},  
    {"lastName": "Alex", "lastName": "Clayton"}  
  ]  
}
```

XML

```
<siblings>  
  <sibling>  
    <firstName>Anna</firstName>  
    <lastName>Clayton</lastName>  
  </sibling>  
  <sibling>  
    <firstName>Alex</firstName>  
    <lastName>Clayton</lastName>  
  </sibling>  
</siblings>
```

04

Lab: JSON 데이터 분석



04. Lab: JSON 데이터 분석

- 파이썬에서 JSON을 사용하기 위해서는 json 모듈을 이용한다. JSON 데이터 포맷은 데이터 저장 및 읽기가 딕셔너리형과 완벽히 상호 호환되어, 딕셔너리형에 익숙한 사용자가 매우 쉽게 사용할 수 있다는 장점이 있다.



04. Lab: JSON 데이터 분석

■ JSON 읽기

- JSON을 읽기 위해서는 JSON 파일의 구조를 확인한 후, json 모듈로 읽고 딕셔너리형처럼 처리한다.

```
{  
  "employees": [  
    {"firstName": "John", "lastName": "Doe"},  
    {"firstName": "Anna", "lastName": "Smith"},  
    {"firstName": "Peter", "lastName": "Jones"}  
  ]  
}
```



04. Lab: JSON 데이터 분석

■ JSON 읽기

- 앞 데이터에는 'employees' 아래에 3개의 데이터가 있다. 이를 파이썬으로 읽어 오기 위해 [코드 15-3]과 같이 입력한다.

코드 15-3 json1.py

```
1 import json
2
3 with open("json_example.json", "r", encoding="utf8") as f:
4     contents = f.read()                # 파일 내용 읽어 오기
5     json_data = json.loads(contents)    # json 파싱
6     print(json_data["employees"])      # 딕셔너리처럼 사용하기
```

```
[{'firstName': 'John', 'lastName': 'Doe'}, {'firstName': 'Anna', 'lastName': 'Smith'},
{'firstName': 'Peter', 'lastName': 'Jones'}]
```

04. Lab: JSON 데이터 분석

■ JSON 읽기

- ➔ 먼저 1행에서 json 모듈을 호출하고, 3행에서 open() 함수를 사용하여 파일 내용을 가져온다. 그리고 4행에서 문자열형으로 변환하여 처리한다. 5행에서는 loads() 함수를 사용하여 해당 문자열형을 딕셔너리형처럼 변환한다. 6행에서 딕셔너리처럼 json_data["employees"]를 print() 함수로 출력하면 결과값이 출력된다.



04. Lab: JSON 데이터 분석

■ JSON 쓰기

- 딕셔너리형으로 구성된 데이터를 json 형태의 파일로 변환하는 과정에 대해 알아보자.

코드 15-4 json2.py

```
1 import json
2
3 dict_data = {'Name': 'Zara', 'Age': 7, 'Class': 'First'}      # 딕셔너리 생성
4
5 with open("data.json", "w") as f:
6     json.dump(dict_data, f)
```

- ➡ json을 쓰기 위해서는 먼저 3행처럼 데이터를 저장한 딕셔너리형을 생성하고, 6행에서 json.dump() 함수를 사용하여 데이터를 저장한다. 이때 인수는 딕셔너리형과 파일 객체가 차례대로 들어가면 완성할 수 있다. 실행 결과, 작업 폴더에 'data.json' 파일이 생성된 것을 확인할 수 있다.





Thank You !

파이썬 프로그래밍