YAML 이란?

서로 다른 시스템 간에 데이터를 주고받을 필요가 있을 때 데이터 연동과 호환성을 위해 포맷에 대한 규칙이 필요하다. CSV, XML, JSON, Properties 등이 바로 이런 용도로 사용되는 파일들이다.

웹에서는 XML과 JSON 형식을 자바 애플리케이션에서는 properties 파일을 이용하여 필요한 값들을 정의하여 사용한다. XML은 사용하기 까다롭고 가독성도 좋지 않기 때문에 요즘 들어 JSON 포매팅 방식이 많이 이용되는 편이지만, JSON 역시 주석을 달수 없는 등 약간의 제한이 있고 인용부호, 중괄호 그리고 대괄호의 사용으로 코드 길이가 강제적으로 길어지게 된다는 단점이 있다.

2001년에 Clark Evans에 의해 최초로 제안된 새로운 포매팅 방식이 YAML / YML 이다.

YAML은 XML과 JSON 포맷과 같이 타 시스템 간에 데이터를 주고받을 때 약속된 포맷(규칙)이 정의되어 있는 또 하나의 파일 형식이다. 다만 좀더 인간 친화적으로 작성해 가독성을 높이는 쪽으로 무게를 두었으며 프로그래밍 언어에 친화적이다.

JSON과 달리 주석도 쓸 수 있으며, 위의 그림과 같이 간결한 문법으로 같은 데이터 양이라도 코드길이를 많이 줄일 수 있다. YAML은 주로 Doker Compose, Kubernetes, Flutter, Spring Boot 프로젝트에서 설정파일을 정의할 때 사용된다.

[YAML 이름 어원]

YAML의 뜻은 'Yaml Ain't Markup Language' 라는 뜻으로 'YAML은 마크업 언어가 아니다'라는 말장난 같은 이름을 가지고 있다. 사실 YAML은 Yet Another Markup Language로 또 다른 마크업 언어라는 이름을 가지고 있다가 마치 NoSQL이 Not Only SQL로 뜻이 변화했듯이, 핵심은 마크업이 아니라 데이터가 중심이라는 것을 보여주기 위해 Yaml Ain't Markup Language가 되었다.

[.yaml 과 .yml 차이점]
yaml과 yml 파일 확장자는 모두 해석 및 구문이 동일하다. 즉, 어떤 식으로 파일 확장자를 정하든 똑같은 것이다. 이렇게 분리된 이유는, 옛날 Windows에서 파일 확장자가 세 글자로 제한되는 특성이 있었기 때문이다. 예전에는 .html 대신 .htm 으로도 쓰였는데 이와 같은 원리이다. 하지만 요즘은 확장자에 3글자를 넣어야 하는 OS 시스템 수준의 규격이 없으니 .yaml 을 사용하면 된다.

```
- name: create users
  hosts: all
  tasks:
    - user:
      name: "{{ item.name }}"
      state: present
      groups: "{{ item.groups }}"
    with_items:
      - { name: 'linda', groups: 'wheel' }
      - { name: 'lisa', groups: 'root' }
```

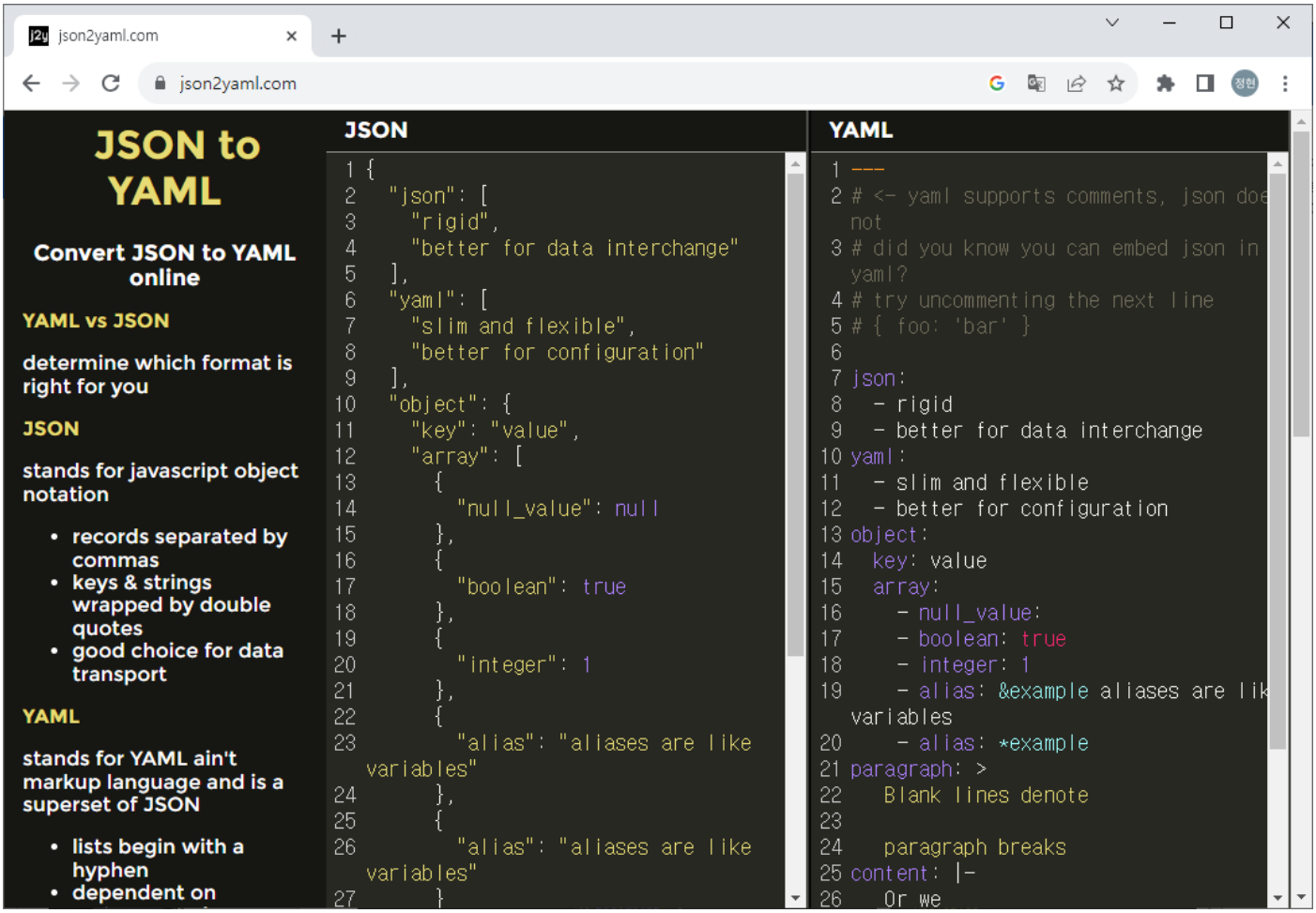
yaml 의 문법 생김새를 보면 무언가 파이썬스럽다는 것을 느낄 수 있다. JSON은 중괄호를 이용해 데이터 간의 구분을 표현하지만, 실제로 yaml은 띄어(들여)쓰기로 데이터를 구분하기 때문이다.

웹에서의 데이터 통신을 위해 대부분 JSON을 많이 사용하고 있고, reference를 정의하려는 경우 복잡한 객체(object) 구조를 표현하기 위해 YAML이 좀더 적합하다.

문법 생김새로 인해, 파이썬 커뮤니티에서는 파이썬 문법과 비슷한 indent로 구분하는 YAML을 더 선호하고, 자바스크립트 진영에서는 별도의 파서가 필요 없고 JavaScript Object와 구조가 유사한 JSON을 선호한다.

[YAML의 문법 정리]

다음 사이트에서는 JSON 소스를 입력하면 자동으로 YAML 소스로 변환되어, 해당 문법이 어떤 것을 의미하는지 쉽게 확인할 수 있다.



[데이터 정의]

Key: Value 표기

JSON 포맷과 같이 기본적으로 key와 value는 콜론(:)을 기준으로 구분한다.
주의할 점은 value를 작성할 때 반드시 콜론(:) 뒤에 띄어쓰기를 한 번 해줘야 한다는 점이다.
즉, 콜론을 쓰고 그 뒤에 공백문자까지 표기해 줘야 key와 value를 정의한 것이다.
만일 콜론(:) 뒤에 공백이 없으면 하나의 스트링으로 인식되어, 문법 오류가 나타나게 된다.

콤마 표기 안함

JSON과 달리 여러 개의 데이터가 있어도 콤마를 쓰지 않고 바로 개행 처리한다.

```
name: 둘리
age: 10
major: 컴퓨터공학
```

들여쓰기(indent)

중괄호로 계층 구조를 표현하는 JSON과 달리, YAML은 파이썬과 같이 들여쓰기로 계층 구조를 표현하기 때문에 들여쓰기 유무는 매우 중요하다. 들여쓰기는 기본적으로 2칸 혹은 4칸을 지원한다.
만일 student 속성안에 name, age, major 키를 가진 객체를 포함시키도록 표현하려면, 들여쓰기를 통해 포함 관계를 표시해야 한다.

```
student:
  name: 둘리
  age: 10
  major: 컴퓨터공학
```

"따옴표" 표현

JSON을 작성할 때는 문자열을 표현할 때 기본적으로 큰 따옴표를 써줘야 하지만, YAML을 작성할 때는 큰 따옴표, 작은 따옴표 혹은 안 써도 자동으로 숫자와 문자열로 인식된다.

그런데 콜론기호(:)가 들어간 문자열이 값으로 사용되는 경우엔 따옴표를 사용해야 제대로 인식된다.

```
name: 둘리
address1: 서울시
address2: "도봉구 쌍문동"
address3: '마이콜 옆집'
ratio1: "1:2"
ratio2: '1:2'
```

[작은 따옴표와 큰 따옴표 차이점]
YAML에서는 문자 데이터를 표현할 때 따옴표를 안 써도 되지만, 이스케이프 문자를 구분해야 할 때는 이 둘을 구분해서 사용해야 한다. 예를 들어 큰 따옴표는 escape sequence 처리를 해주고, 작은 따옴표는 주어진 그대로 문자열 처리한다. \n 문자를 큰 따옴표로 묶으면 개행 문자로 인식되고, 작은 따옴표로 묶으면 일반 문자로 \n을 인식한다.

```
string1: "\t tab \n NL"
string2: '\t tab \n NL'
```

배열 & 리스트

- 단순 배열
YAML에서의 배열을 표현할 때는 하이픈(-)으로 시작하는 하위 엘리먼트로 표현할 수 있다. 이때도 하이픈(-) 뒤에 띄어쓰기를 써주어야 리스트로 인식된다.

```
members:
  - 둘리
  - 또치
  - 도우너
```

또한 다음과 같이 하이픈(-) 대신, 일반적인 배열 표현법 같이 대괄호로 리스트를 한 줄로 표현하는 것도 가능하다.

```
members: [둘리, 또치, 도우너]
```

- 객체 배열

단순한 값 리스트의 나열이 아니라, 보다 복잡한 구조체의 리스트를 표현하고자 한다면 하이픈(-) 과 key: value 구조를 이용하여 객체를 포함한 배열 표현이 가능하다.

한 덩어리의 객체를 나타내기 위해 즉, 동일한 객체의 멤버이면 하이픈(-)을 쓰지 않고 그대로 나열한다.

```
students:
  - name: 둘리
    major: 국어
    age: 10
  - name: 또치
    major: 영어
    age: 11
  - name: 도우너
    major: 수학
    age: 12
```

[단순 원소 배열과 객체 원소 배열]

```
members: # 단순 원소 배열
  - 둘리
  - 또치
  - 도우너
```

```
members: # 객체 원소를 포함한 배열
  - class1: 둘리
  - class2: 또치
  - class3: 도우너
```

Boolean 표현

일반적인 프로그래밍 언어 또는 JSON과 달리 YAML의 불리언 표현 방법은 4가지이다. 데이터 부분에 yes, no, true, false 를 지정하면 boolean 값으로 인식된다. 대소문자는 구분하지 않는다.

```
booleans:
  - yes
  - Yes
  - YES
  - true
  - True
  - FALSE
  - "YES"
  - "NO"
```

주석 표현

JSON에서는 주석을 표기할 수 없지만, YAML은 파이썬과 같이 샵(#)기호로 주석을 표기할 수 있다.

```
# 학생 정보
name: 둘리
major: 수학
```

텍스트 Multi Line

개행이 있는 문자열을 JSON으로 표현한다면, 문자열 내에 \n 문자를 사이사이 넣어주어야 한다.
YAML에서는 일반적인 프로그래밍과는 다른 방법으로 지원한다. 개행된 문자열을 표현하는데 다음과 같이 두 가지 방법이 존재한다.

> 기호를 쓰는 방법(folded block scalar)

| 기호를 쓰는 방법(literal block scalar)

(1) folded block scalar

> 기호는 개행(엔터) 문자를 한 개의 공백 문자로 치환해주고, 빈 줄 하나가 단독으로 있는 경우만 개행문자로 치환한다. 맨 마지막은 자동으로 개행 처리를 해주지만, > 기호 옆에 - 를 붙여주면 맨 끝에는 개행 문자를 포함시키지 않는다.

```
paragraph1: >
  abc
  def
  ghi

  aab
  ccc
paragraph2: >- # 맨끝의 개행 문자를 포함시키지 않는다.
  abc
  def
  ghi

  aab
  ccc
```

```
paragraph1: "abc def ghi\naab ccc\n"
paragraph2": "abc def ghi\naab ccc"
```

(2) literal block scalar

| 기호는 개행(엔터) 문자가 인식될 때마다 개행문자로 처리한다.

| 기호 옆에 - 를 붙이면 맨 끝에 개행 문자를 포함시키지 않는다.

```
# 그대로 줄바꿈 + 맨 끝 개행문자 포함
```

```
paragraph3: |
abc
def
ghi

aab
ccc

# 그대로 줄바꿈 + 맨끝 개행문자 제외
paragraph4: |-
abc
def
ghi

aab
ccc
```

```
paragraph3: "abc\ndef\nghi\n\naab\nccc\n"
paragraph4": "abc\ndef\nghi\n\naab\nccc"
```

Multiple Documents

YAML 에서는 두 가지 파일의 데이터를 하나의 파일로 포함시켜 표현해주는 기능을 제공하므로
하나의 파일에서 모든 configuration을 관리할 수 있도록 지원한다.

스프링 프로젝트 환경이 dev 또는 product에 따라 적용되는 옵션 값들이 다른 설정 프로퍼티 파일들이 각각 존재하는 경우, YAML에서는 하이픈 3개를 이용하여 --- 이 두 프로퍼티 파일의 내용을 하나의 파일에 포함시켜 한꺼번에 정의해줄 수 있다.

```
spring:
  profiles: dev
  datasource:
    url: jdbc:mysql://
    hikari:
      minimum-idle: 1
```

```
        maximum-pool-size: 3
management:
  endpoints:
    web:
      base-path: /
---
spring:
  profiles: prod
  datasource:
    url: jdbc:mysql://
    hikari:
      minimum-idle: 5
      maximum-pool-size: 10
management:
  endpoints:
    web:
      base-path: /
```

[사용 분야]

일반적으로 설정파일로 사용하기에 더할 나위없이 좋은 형식이기 때문에 여러 프레임워크나 CI툴에서 설정파일로 쓰이고 있다.

GitLab CI: 루트에 위치한 .gitlab-ci.yml파일을 설정파일로 사용한다.
trevisCI: 루트에 위치한 .trevis.yml파일로 실행할 작업들을 지정한다.
GitHub Action: 설정 파일이 .github/workflows/<워크플로 이름>.yml로 YAML형식을 사용한다.
sourcehut: 루트에 위치한 .build.yml파일을 사용한다.
CircleCI: .circleci/config.yml 파일을 사용한다.
Docker: compose파일을 YAML로 작성하고 배포한다.
쿠버네티스: 어떤 프레임워크보다 YAML을 광범위하게 사용하는 프레임워크 중 하나이다.
기본적인 파드, 레플리카, 디플로이먼트 등 모든 내부 오브젝트를 YAML문서로 정의하며,

YAML 고유 기능 중 하나인 문서 스트림을 사용해 클러스터 전체의 설정을 파일 하나로 관리할 수도 있다. kubectl이나 GKE 클라우드 콘솔 등은 YAML 형식 출력을 제공하기도 한다.

Ansible: 자동화 툴 중 하나로, 설정 파일로 YAML을 사용한다.

prometheus: 설정파일로 prometheus.yml 파일을 사용한다. 파일 기반 SD(service discovery)의 경우에도 YAML 과 JSON 포맷을 지원한다.

Sublime Text: 설정 파일로 사용할 수 있다.

스프링 부트: YAML 방식의 설정 파일을 사용할 수 있다.
(application.properties -> application.yaml)

OpenAPI(전 Swagger): API문서를 YAML 또는 JSON으로 작성할 수 있다. doc-comment(annotaton)로 코드 내에 주석으로 삽입하는 방식의 경우 YAML 형식이 더욱 선호되는 편이다. 동일하게 반복되는 스키마를 참조하는 방법으로 YAML의 앵커 문법 대신 json-reference를 사용한다. 또한 swagger-ui의 경우 YAML파일을 서버로 직접 요청해 프런트에서 문서를 렌더링할 수 있다.