Что такое YAML? Синтаксис, примеры и применение

07.04.2024

Если вы разработчик, пишущий API, вы наверняка сталкивались с YAML, хотя вашим основным инструментом сериализации может быть JSON. YAML имеет свой собственный дружественный синтаксис, и это удобный язык, который стоит добавить в свой арсенал разработчиков. Давайте познакомимся с основами YAML.

Сериализация данных

Когда вы хотите отправить какую-то структуру данных или объект по компьютерным сетям, например по Интернету, вам необходимо преобразовать ее в специальный формат для чтения и хранения. Этот процесс известен как сериализация и имеет огромное значение в Интернете. Чаще всего сериализация используется при чтении данных из баз данных и передаче их через Интернет. Некоторые форматы сериализации включают JSON, YAML, XML. В этой статье я расскажу о YAML, и в конце статьи вы сможете работать с YAML и будете иметь четкое представление о нем.

Что такое YAML?

YAML — это формат сериализации данных, который расшифровывается как **YAML ain’t Markup language**. Основное преимущество использования YAML — это удобство чтения и записи. Если у вас есть конфигурационный файл, который должен быть проще для чтения человеком, лучше использовать YAML. YAML не является полной заменой JSON, поскольку JSON и XML тоже имеют свое место; тем не менее, изучать YAML полезно. Еще одно преимущество YAML — поддержка различных типов данных, таких как регистры, массивы, словари, списки и скаляры. Он хорошо поддерживает самые популярные языки, такие как JavaScript, Python, Ruby, Java и другие. В YAML поддерживаются

только пробелы, и он чувствителен к регистру, а также к пробелам. Табуляции не принимаются повсеместно. Файл YAML имеет расширение .yaml.

Основной синтаксис YAML

Каждый YAML начинается с символа ---, который обозначает начало YAML-файла. При создании API нас интересует функция, предоставляемая YAML, известная как отображение. В следующих примерах показан пример отображения в YAML.

```
---
name: James
boy: yes
GPA: 3.41
```

Синтаксис отображения – ключ: значение. (Обратите внимание на пробел, он очень важен в YAML, в отличие от JSON или XML. YAML также поддерживает такие типы данных, как символы, строки, целые числа, плавающие значения, а также коллекции, такие как массивы, списки, которые строятся из базовых типов данных.

Типы данных в YAML

Ниже приведен пример YAML:

```
---

MALE: FALSE

GPA: 3.61

ISSUES: NULL

NAME: “HARRY”

AGE: 16
```

Первый тип данных – булево, у которого может быть два значения: true или false. Значение GPA имеет плавающую точку.

YAML также поддерживает тип данных null, как и в случае с “*Issues”.* Значение “*Name*” – это строка, которую необходимо заключить в двойные или одинарные кавычки. YAML также поддерживает многострочную строку и строку из нескольких строк как одну для удобства чтения.

Многострочные и однострочные строки

```
---

About: >

 Hello this is Ryan

 From Alabama and I like to

 Play soccer.
```

Символ позволяет записывать однострочную строку в несколько строк. Предложение на самом деле является однострочным описанием, хотя у нас несколько строк. Мы также можем иметь несколько строк, если используем символ |, как это разрешено:

```
About: |

 This is a multiline string

 And will be printed line wise.
```

Список

Списки очень важны в YAML. Пример списка приведен ниже.

```
---

- apple

- banana

- mango
```

Ниже показано отображение скаляров на списки, что очень важно для большинства конфигурационных файлов.

```
Fruits:

 Apple

 Banana

 Guava
```

Вложенность необходима для отображения скаляра в список. Мы также можем иметь несколько вложенных списков, как показано в примере ниже.

```
Automobiles:

 Car:

     Hyundai

     Volkswagen

     Ford
```

Здесь автомобили вложены в автомобили, а Hyundai вложена в автомобили. Это пример множественной вложенности. Мы можем использовать множественную вложенность сколько угодно.

```
Subjects:

     Engineering:

       Mechanical engineering:

         Design and manufacture

         Automobile

         Control and Design
```

```
        Civil engineering:

          Structural engineering

          Hydropower

        Arts:

          Medieval

          Modern

          Painting
```

В YAML также предусмотрены символы & и * в качестве якорей и ссылок на якорь, чтобы избежать дублирования. Они необходимы в конфигурационных файлах таких фреймворков, как Ruby on Rails, чтобы сделать YAML-файл меньше. Смотрите пример ниже.

```
details: &details
    name: "John"
    age: 18

profession: engineer

<< : * details
```

что эквивалентно:

```
profession: engineer

name: "John"

age: 18
```

YAML в Python

Python поддерживает YAML, включая некоторые модули, такие как ruamel и pyyaml. Начните с установки pyyaml

```
pip install pyyaml
```

Для учебника создайте файл с именем details.yaml

```
name: "john"

age:18

gender: male
```

Создайте еще один файл с именем feed.yaml со следующим содержимым:

```
sports:

 football
 basketball
 cricket
 baseball

---
countries:
 Brazil
 Lithuania
 Australia
 USA
```

Давайте начнем с чтения файла details.yaml

```
import yaml

with open('details.yaml') as f:
    data = yaml.load(f, Loader=yaml.FullLoader)
    print(data)
```

Запустив файл details.py, мы получим следующий результат

```
 $ python details.py
{'name': "john", 'age': 18, 'gender': male}
```

```
import yaml

with open(r'feed.yaml') as file:
     # The FullLoader parameter handles the conversion from
YAML
```

```
    # scalar values to Python the dictionary format
    fruits_list = yaml.load(file, Loader=yaml.FullLoader)

    print(fruits_list)
```

Запись YAML в файлы в Python

```
import yaml

dict_file = [{'sports' : ['hockey', 'rugby', 'tennis', 'ping
pong', 'football', 'badminton']},
{'countries' : ['Jamaica', 'England', 'Nepal', 'Netherlands',
'South Africa', 'Bolivia', 'Portugal']}]

with open(r'E:data.yaml', 'w') as file: #create a new yaml
file
    data = yaml.dump(dict_file, file)
```

Реализация YAML в Node.js

Node.js – это язык обработки данных на стороне сервера, и сериализация данных имеет огромное значение в процессе разработки.

Для нашего учебника рассмотрим следующий файл example.yaml:

```
name:John

age:18

Hobbies:

 Hobby1:Football

 Hobby2:BasketBall

 Hobby3:Hockey

Job:
```

-System administrator

-Programmer

Для Node.js у нас есть библиотека npm под названием js-yaml. Давайте начнем с установки модуля с помощью

```
npm install js-yaml
```

Затем мы используем модуль js-yaml в нашем файле.

```
const yaml = require('js-yaml'); //initialize js-yaml
const fs   = require('fs'); //initialize filestream

try {
   const result = yaml.load(fs.readFileSync('example.yml', 'utf8'));
  console.log(result);
} catch (e) {
  console.log(e); //catch exception
}
```

Заключительные слова

В современных программах, фреймворках и приложениях, где хранятся или распределяются данные, YAML становится все более распространенным в конфигурационных файлах. YAML нацелен на многие из тех же коммуникационных приложений, что и Extensible Markup Language (XML), но имеет минимальный синтаксис, который намеренно отличается от XML. Файлы YAML можно создавать для фиксированных структур данных с помощью команд печати, которые записывают как данные, так и конкретное оформление YAML. Однако для дампа различных или сложных иерархических данных предпочтительнее использовать специальный эмиттер YAML. Аналогично, с помощью регулярных выражений можно легко разобрать базовые файлы YAML (например, пары ключ-значение).