json-enhanced

Daniel Diego Horcajuelo

Feb 27, 2022

TABLE OF CONTENTS:

Note: JSON Enhanced currently supports Python 3.8 and higher.

CHAPTER

ONE

INTRODUCTION

1.1 Installation

This library can be installed with pip:

We recommend a virtual environment (https://dev.to/bowmanjd/python-tools-for-managing-virtual-environments-3bko#howto)

1.2 Quickstart

To start, we import the module into our workspace:

There are different ways to load a JSON object in our interpreter:

- By directly entering the corresponding Python object in the constructor

```
json_object = js.JSONObject(
    {
        "data": [
            {
                "name": "Daniel",
                "age": 30,
                "hobbies": ["music", "reading", "football"]
            },
            {
                "name": "Gloria",
                "age": 25,
                "hobbies": ["tennis", "music", "programming"]
            }
        ]
    }
)
```

- By calling the method `open` of the `JSONObject` class:

```
json_object = js.JSONObject.open("<path_of_json_file>") # to open a local json file

json_object = js.JSONObject.open("<url_of_json_file>") # to open a remote url json file
```

Note: The method automatically detects if the argument is a local path or a remote URL. Remote URLs must contain the protocol (http/https).

Once the json data has been loaded as a `JSONNode` instance, we will be able to perform some useful things, such as browsing the nested object by attribute access with autocompletion features:

```
json_object.data._0.hobbies._1
# 'reading'

>> json_object.data._1.name
    'Gloria'
```

Or changing node elements as we want:

```
json_object.data._0.hobbies._1 = "SLEEPING"

>> json_object
    {
        "data": [
            {
                "name": "Daniel",
                "age": 30,
                "hobbies": ["music", "SLEEPING", "football"]
            },
            {
                "name": "Gloria",
                "age": 25,
                "hobbies": ["tennis", "music", "programming"]
            }
        ]
    }
```

But probably the most important feature is the ability to make queries, following a syntax similar to the one offered by Django's ORM. Let's see some examples:

```
>> json_object.query(hobbies__contains="football")
    <QuerySet [['music', 'SLEEPING', 'football']]>

>> json_object.query(age__lt=30, include_parent_=True).first() # retrieving the first␣
↪query result including parent object (the inner dict)
    {'name': 'Gloria', 'age': 25, 'hobbies': ['tennis', 'music', 'programming']}

>> json_object.query(name__regex=r"(?:Daniel|Gloria)")
    <QuerySet ['Daniel', 'Gloria']>

>> json_object.query(hobbies__contains="music").count() # counting the number of nodes␣
↪with 'music' as hobbie
    2

>> json_object.query(hobbies=js.All).update(None) # updating all hobbies nodes to null␣
↪values
```

(continues on next page)

(continued from previous page)

```
>> json_object
    {
        "data": [
            {
                "name": "Daniel",
                "age": 30,
                "hobbies": None
            },
            {
                "name": "Gloria",
                "age": 25,
                "hobbies": None
            }
        ]
    }
```

CHAPTER

TWO

COMPARISON WITH OTHER RELATED LIBRARIES

There are other libraries that implement similar functionality to that of json-enhanced, but all of them use their own syntax for querying data, whereas json-enhanced uses a purely Python syntax, based on Django's ORM.

Let's see a quick usage comparison between the different related packages.

2.1 jsonpath-ng

Perhaps the most used library to cover this type of needs. Following the example concerning authors and books, published on its web site:

Listing 1: with jsonpath-ng

```
data = {
    "store": {
        "book": [
            {
                "category": "reference",
                "author": "Nigel Rees",
                "title": "Sayings of the Century",
                "price": 8.95
            },
            {
                "category": "fiction",
                "author": "Evelyn Waugh",
                "title": "Sword of Honour",
                "price": 12.99
            },
            {
                "category": "fiction",
                "author": "Herman Melville",
                "title": "Moby Dick",
                "isbn": "0-553-21311-3",
                "price": 8.99
            },
            {
                "category": "fiction",
                "author": "J. R. R. Tolkien",
                "title": "The Lord of the Rings",
                "isbn": "0-395-19395-8",
```

(continues on next page)

(continued from previous page)

```
                "price": 22.99
            }
        ],
        "bicycle": {
            "color": "red",
            "price": 19.95
        }
    }
}

from jsonpath_ng.ext import parse

# retrieving the authors of all books in the store
query_all_book_authors = parse("$.store.book[*].author")

[match.value for match in query_all_book_authors.find(data)]
    ['Nigel Rees', 'Evelyn Waugh', 'Herman Melville', 'J. R. R. Tolkien']

# getting the json path of the first author
str(query_all_book_authors.find(data)[0].full_path)
    'store.book.[0].author'

# filter all books with isbn number
query_isbn_books = parse("$..book[?(@.isbn)]")

[match.value for match in query_isbn_books.find(data)]
    [{'category': 'fiction',
    'author': 'Herman Melville',
    'title': 'Moby Dick',
    'isbn': '0-553-21311-3',
    'price': 8.99},
    {'category': 'fiction',
    'author': 'J. R. R. Tolkien',
    'title': 'The Lord of the Rings',
    'isbn': '0-395-19395-8',
    'price': 22.99}]
```

Listing 2: with json-enhanced

```
import jsonutils as js

data = js.JSONObject(data) # load previous data as a JSONNode instance

# retrieving the authors of all books in the store
data.store.book.query(author=js.All) # just one statement
    <QuerySet ['Nigel Rees', 'Evelyn Waugh', 'Herman Melville', 'J. R. R. Tolkien']>

# getting the json path of the first author
data.store.book.query(author=js.All).first().jsonpath
    store/book/0/author/

# or if we want a python path expression
```

(continues on next page)

(continued from previous page)

```
data.store.book.query(author=js.All).first().jsonpath.expr
    '["store"]["book"][0]["author"]'

# filter all books with isbn number
data.store.book.query(isbn__isnull=False, include_parent_=True)
    <QuerySet [{'category': 'fiction', 'author': 'Herman Melville', 'title': 'Moby Dick',
↪ 'isbn': '0-553-21311-3', 'price': 8.99}, {'category': 'fiction', 'author': 'J. R. R.␣
↪Tolkien', 'title': 'The Lord of the Rings', 'isbn': '0-395-19395-8', 'price': 22.99}]>
```

2.2 objectpath

This library is currently out of maintenance. Its syntax is very similar to that of jsonpath-ng. Let's compare its functionality following the example of json above.

Listing 3: with objectpath

```
from objectpath import Tree

tree = Tree(data) # loading the data above

# retrieving books with price greater than 12
list(tree.execute("$.store.book[@.price > 12]"))
    [{'category': 'fiction',
    'author': 'Evelyn Waugh',
    'title': 'Sword of Honour',
    'price': 12.99},
    {'category': 'fiction',
    'author': 'J. R. R. Tolkien',
    'title': 'The Lord of the Rings',
    'isbn': '0-395-19395-8',
    'price': 22.99}]
```

Listing 4: with json-enhanced

```
import jsonutils as js

data = js.JSONObject(data)

# retrieving books with price greater than 12
data.store.book.query(price__gt=12) # without including parent nodes
    <QuerySet [12.99, 22.99]>

# getting the last element's parent
data.store.book.query(price__gt=12).last().parent
    {'category': 'fiction',
    'author': 'J. R. R. Tolkien',
    'title': 'The Lord of the Rings',
    'isbn': '0-395-19395-8',
    'price': 22.99}
```

2.3 pandas

The pandas library offers very user-friendly tools for querying structured data. The main problem is that it can only properly read data that has a defined structure (which can be converted to a dataframe). In the case we are concerned with, we could proceed as follows:

Listing 5: with pandas

```
import pandas as pd

df = pd.json_normalize(data, ["store", ["book"]])

df
        category            author                   title  price           isbn
    0  reference        Nigel Rees  Sayings of the Century   8.95            NaN
    1    fiction      Evelyn Waugh         Sword of Honour  12.99            NaN
    2    fiction   Herman Melville                Moby Dick   8.99  0-553-21311-3
    3    fiction  J. R. R. Tolkien   The Lord of the Rings  22.99  0-395-19395-8

# filter books with no isbn
df.query("isbn.isna()")
        category        author                   title  price isbn
    0  reference    Nigel Rees  Sayings of the Century   8.95  NaN
    1    fiction  Evelyn Waugh         Sword of Honour  12.99  NaN

# filter books whose title contains the string "the" and has a valid ISBN
df.query("title.str.contains('the') & isbn.isna() == False")
    category            author                 title  price           isbn
    3  fiction  J. R. R. Tolkien  The Lord of the Rings  22.99  0-395-19395-8
```

Listing 6: with json-enhanced

```
import jsonutils as js

data = js.JSONObject(data) # load the data into JSONNode instance

# filter books with no isbn
data.store.book.annotate(isbn=None).query(isbn__isnull=True, include_parent_=True)
    <QuerySet [{'category': 'reference', 'author': 'Nigel Rees', 'title': 'Sayings of␣
↪the Century', 'price': 8.95, 'isbn': None}, {'category': 'fiction', 'author': 'Evelyn␣
↪Waugh', 'title': 'Sword of Honour', 'price': 12.99, 'isbn': None}]>

# filter books whose title contains the string "the" and has a valid ISBN
data.store.book.query(title__contains="the", isbn__isnull=False)
    <QuerySet [{'category': 'fiction', 'author': 'J. R. R. Tolkien', 'title': 'The Lord␣
↪of the Rings', 'isbn': '0-395-19395-8', 'price': 22.99}]>
```

CHAPTER

THREE

INDICES AND TABLES

- genindex
- modindex
- search