Schema Tools Documentation

Release 0.0.19

Christophe VG

Mar 05, 2021

CONTENTS

Collection of tools to parse, query, map,... Json/Yaml schemas

CHAPTER

ONE

FEATURES AT A GLANCE

- support for (loading of) Json and Yaml schemas with additional access to line and column information as an Abstract Syntax Tree (AST)

```
>>> from schema_tools import json
>>> j = json.loads('''{
...   "hello" : "world",
...   "count" : [
...     1,
...     2,
...     3
...   ]
... }''')
>>> j
ObjectNode(len=2, line=1, column=1)
>>> j()
{'hello': 'world', 'count': [1, 2, 3]}

>>> from schema_tools import yaml
>>> y = yaml.loads('''
... hello : world
... count :
... - 1
... - 2
... - 3
... ''')
>>> y
ObjectNode(len=2, line=1, column=1)
>>> y()
{'hello': 'world', 'count': [1, 2, 3]}

>>> for k, v in y:
...   print(k, v)
...
hello ValueNode(value=world, line=2, column=9)
count ListNode(len=3, line=3, column=1)

>>> from schema_tools.utils import node_location
>>> l = node_location(j.count)
>>> l.line
3
>>> l.column
13
>>> node_location(y.count)
NodeLocation(3, 0)
```

(continues on next page)

(continued from previous page)

```
>>> node_location(j["count"])
NodeLocation(3, 13)
>>> node_location(y["count"])
NodeLocation(3, 0)
```

- a schema oriented object model

```
>>> from schema_tools.schema import loads
>>> json_src = '''
... {
...   "$schema": "http://json-schema.org/draft-07/schema#",
...   "$id": "test",
...   "title": "a title",
...   "description": "a description",
...   "version": "123",
...   "type": "object",
...   "properties" : {
...     "home" : {
...       "anyOf" : [
...         { "$ref" : "#/definitions/address" },
...         { "$ref" : "#/definitions/id"      }
...       ]
...     }
...   },
...   "definitions" : {
...     "id" : {
...       "type" : "string"
...     },
...     "address" : {
...       "type" : "object",
...       "properties" : {
...         "url": {
...           "type": "string",
...           "format": "uri-reference"
...         }
...       },
...       "additionalProperties" : false,
...       "required" : [
...         "url"
...       ]
...     }
...   }
... }'''
>>> schema = loads(json_src)
>>> schema
ObjectSchema($schema=http://json-schema.org/draft-07/schema#, $id=test, title=a title,
↪ description=a description, version=123, type=object, properties=1, definitions=2,␣
↪location=NodeLocation(line=2, column=1))
>>> schema.properties[0]
Property(name=home, definition=AnyOf(options=2, location=NodeLocation(line=10,␣
↪column=14)), location=None)
>>> schema.properties[0].definition.options[1]
Reference(ref=#/definitions/id)
>>> schema.properties[0].definition.options[1].resolve()
StringSchema(type=string, location=NodeLocation(line=18, column=12))
>>> schema.properties[0].definition.options[1].resolve().parent.name
'id'
```

- (re) generate dict/textual representation

```
>>> import json
>>> from schema_tools.schema import load
>>> schema = load("tests/schemas/json-schema-draft-07.json")
>>> d = schema.to_dict()
>>> s = json.dumps(d, indent=2, sort_keys=True)
>>> print(s[:150], "{}\n...\n{}".format(s[:147], s[-83:]))
{
  "$id": "http://json-schema.org/draft-07/schema#",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "default": true,
  "definitions": {
   {
  "$id": "http://json-schema.org/draft-07/schema#",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "default": true,
  "definitions": {
...
  "title": "Core schema meta-schema",
  "type": [
    "object",
    "boolean"
  ]
}
```

- use property selectors - across schema boundaries ;-)

```
>>> from schema_tools.schema import load
>>> schema = load("tests/schemas/invoice.json")
>>> amount = schema.select("lines.price.amount")
>>> amount.parent.parent
ObjectSchema($schema=http://json-schema.org/draft-07/schema#, $id=file:tests/schemas/
↪money.json, type=object, version=1, additionalProperties=False, required=['amount',
↪'currency'], properties=2, definitions=0, location=NodeLocation(line=1, column=1))
```

- define mapping between schemas with mapping validation

```
>>> from schema_tools.schema import load
>>> source = load("tests/schemas/product.json").select("cost.amount")
>>> target = load("tests/schemas/invoice.json").select("lines.price.amount")
>>> from schema_tools.mapping import Mapping
>>> m = Mapping(source, target)
>>> m.is_valid
True
>>> target = load("tests/schemas/invoice.json").select("lines.price.currency")
>>> m = Mapping(source, target)
>>> m.is_valid
False
>>> False
False
>>> print(m.status)
source:IntegerSchema(type=integer, location=NodeLocation(line=7, column=15))
target:StringSchema(type=string, location=NodeLocation(line=10, column=17))
source type 'IntegerSchema' doesn't match target type 'StringSchema'
```

CHAPTER

TWO

CONTENTS

2.1 What's in The Box?

TODO

2.2 Getting Started

Schema Tools is hosted on PyPi, so...

```
$ pip install schema-tools
```

2.3 Contributing

This is Open Source software, so given enough eyeballs, all bugs are shallow. Your contributions are more than welcome.

This project is hosted on GitHub and these (common) rules apply:

- Do use the issues tracker.
- Let's discuss any proposed change or fix in an issue, so your work is not done in vain - I hate to reject pull requests...
- Create pull requests against `next` branch.
- Try to keep pull requests "atomic", and if possible related to an issue.