nadia Documentation

Release 0.1.0

Konrad Jałowiecki

Jun 25, 2018

User documentation

nadia is a small and lightweight library for creating marshmallow schemas for objects defined in OpeanAPI 3 specifications.

CHAPTER 1

Usage example

nadia 's usage is as simple as it gets. Suppose you have standard OpenAPI petstore example yaml saved in your current directory. Below short snippet will create Schema for the Pet object and use it to validate two example objects.

```
import yaml
import nadia.api

with open('petstore.yaml') as petstore:
    data = yaml.load(petstore)

builder = nadia.api.SchemaBuilder.create()
schema = builder.build(data['components']['schemas']['Pet'])

valid_pet = {'id': 100, 'name': 'Doggo', 'tag': 'sometag'}
invalid_pet = {'id': 'foo', 'name': 'Lessie', 'tag': ['tag1', 'tag2']}

print('Validation errors for Doggo: {}'.format(schema.validate({'content': valid_pet}
↪)))
print('Validation errors for Lessie: {}'.format(schema.validate({'content': invalid_
↪pet})))
```

CHAPTER 2

What can **nadia** do?

- construct Schema for your data type provided as a dict read from your spec's yml or json file.
- skip you the hassle of redesigning your Schema definitions everytime your specification changes. Just load your specs and let **nadia** do the job for you.

CHAPTER 3

What can't **nadia** do?

- use json references in your yaml/json file - you have to resolve them yourself.
- validate your OpenAPI specification - but there are tools that can do that for you.
- generate webserver using some xyz framework.

3.1 nadia Tutorial

3.1.1 Basic concepts

The basic workflow with **nadia** consists of the following steps:

1. Load your yaml or json specification file.
2. Construct SchemaBuilder - **nadia** defines convenient method for creating the reasonable one via factory *nadia.api.SchemaBuilder.create()*
3. Extract object definitions for which you want to create schema from the loaded dictionary.
4. Construct schema for the extracted object. Note that the schema will contain top level *content* field.
5. Use your schema to validate or serialize/deserialize objects.

3.1.1.1 Why do all schemas created by nadia have a top level *content* field?

The reason why nadia hides your real object in the *content* field is that not all OpenAPI schemas can be expressed as `marshmallow.Schema`. For instance, consider the following schema:

```
type: array
items:
  type: number
```

Such a data type can be modelled as `marshmallow.field.List` but not as a *Schema*. However, if you wrap it inside a larger object under *content* field, than a Schema for such object can be constructed. This step is done automatically by **nadia** - even if it is unneccesary, to make behaviour consistent for all objects.

3.1.2 Creating Schemabuilders

You can create a *nadia.api.SchemaBuilder* instance in one of two ways:

1. You can explicitly initialize it by passing a object providings component builders (see *nadia.builder_provider*.
2. You can use *nadia.api.SchemaBuilder.create* staticmethod to construct default builder using **nadia's** default builder providers.

3.2 nadia package

3.2.1 Submodules

3.2.1.1 nadia.api module

The nadia public api.

class nadia.api.**SchemaBuilder**(*builder_provider*)
Bases: object

Class used for building schemas from given specification dict.

Parameters **builder_provider** – an object providing builders via get_builder method. Typically an instance of *nadia.builder_provider.BuilderProvider*.

build(*spec*)
Build schema from given specification dict.

Parameters **spec** (*dict*) – a dictionary containing specification of the object for which Schema should be build.

Returns Schema corresponding to the object defined by spec

Return type marshmallow.Schema

static create()
Create SchemaBuilder.

Return type *nadia.api.SchemaBuilder*

Note: This method is designed to be further extended as a factory method. Later it should be able to accept some parameters governing creation of the SchemaBuilder. Currently it creates *nadia.api.SchemaBuilder* by passing default instance of nadia.builder_provider.BulderProvider to initializer.

3.2.1.2 nadia.array module

Functionalities related to building schemas for array type.

class nadia.array.**ArrayBuilder**(*builder_provider*)
Bases: *nadia.common.Builder*

Schema builder for array datatype.

build_schema(*spec*, ***kwargs*)
Build Marshmallow schema for array datatype.

Parameters **spec** (*dict*) – specification of array for which Schema should be build.

Returns a List field constructed such that its properties correspond to the ones defined in *spec*.

Return type marshmallow.fields.List

3.2.1.3 nadia.builder_provider module

Implementation of class responsible for obtaining schema builders for OpenAPI types.

class nadia.builder_provider.**BuilderProvider**(*builders*)
Bases: object

Class for providing builders for various data types.

Parameters **builders** (*dict*) – mapping between OpenAPI types and classes implementing builder interface for them.

Note: The purpose of this class is to act as a mapping between types and corresponding builders, while elegantly handling unknown types and lazy creation of the builders. It is injected as a dependency in *nadia.api.SchemaBuilder*.

---

get_builder
Get builder instance for given OpenAPI type.

Parameters **typename** (*str*) – OpenAPI type for which to get a builder.

Returns a builder corresponding to given type.

Return type a subclass of *nadia.common.Builder*

Raises *nadia.exceptions.UnknownTypeException* if typename does not correspond to known OpenAPI type.

static get_default()
Construct BuilderProvider with default builders.

Return type *nadia.builder_provider.BuilderProvider*

logger
Logger used by this BuilderProvider.

3.2.1.4 nadia.common module

Functionalities related to all builders.

class nadia.common.**Builder**(*builder_provider*)
Bases: abc.ABC

Base class for all field builders.

Parameters **builder_provider** – a provider used for obtaining builders for other OpenAPI types.

build_schema(*spec*, ***kwargs*)
Build marshmallow schema from definition extracted from OpenAPI specs.

Parameters **spec** – an object definition extracted from OpenAPI specification.

Returns Schema or a Field constructed from the spec dictionary.

Return type `marshmallow.Schema` or :py:class:marshmallow.Field

Note: The return type as well as the nature of the object constructed depends on the concrete implementation of Builder. Usually the builder designed for handling object types will return Schema object, while builders designed for handling other data types will return Field instances.

key = None

marshmallow_class = None

static translate_args(*spec*, ***kwargs*)
Translate arguments given in OpenAPI spec to keyword arguments for marshmallow classes.

Parameters **spec** (*dict*) – a dictionary extracted from OpenAPI spec containing definition of some object.

Returns A mapping containing keyword arguments used for constructing marshmallow objects.

Return type dict

3.2.1.5 nadia.exceptions module

Common Exceptions.

exception nadia.exceptions.**UnknownTypeException**(*typename*)
Bases: `Exception`

Exception raised when we encounter unknown datatype in schema.

Note that circumstances under which this Exception is raised should not happen when the schema dict passed was created from valid OpenAPI specification.

Parameters **typename** (*str*) – name of type that resulted in raising exception.

3.2.1.6 nadia.object module

Implementation of schema builder for object type.

class nadia.object.**ObjectBuilder**(*builder_provider*)
Bases: *nadia.common.Builder*

Schema builder for object datatype.

build_schema(*spec*, ***kwargs*)
Build Schema from a definition of OpenAPI object.

Parameters **spec** (*dict*) – a mapping containing object definition extracted from OpenAPI spec.

Returns Schema constructed from *spec*.

Return type `marshmallow.Schema`

construct_attributes_schemas(*spec*)
Construct Schemas corresponding to object definition.

This method is used to construct attributes of the object schema being constructed.

Parameters **spec** (*dict*) – an OpenAPI object's definition.

Returns a mapping property-name -> Schema or Field.

Return type dict

static create_schema_type(*attrs*)
Create schema type from given dictionary of attributes.

Parameters **attrs** (*dict*) – mapping of attributes of the newly created Python type.

Returns newly created type with randomly chosen name and single base class - `nadia.NadiaSchema`.

Return type type

3.2.1.7 nadia.primitives module

Schema generators for primitive types.

class nadia.primitives.**FloatBuilder**(*builder_provider*)
Bases: *nadia.primitives.PrimitiveBuilder*

Float schema builder.

This builder is designed for constructing schemas for OpenAPI *number* type.

key = 'number'

marshmallow_class
alias of `marshmallow.fields.Float`

class nadia.primitives.**IntegerBuilder**(*builder_provider*)
Bases: *nadia.primitives.PrimitiveBuilder*

Integer schema builder.

This builder is designed for constructing schemas for OpenAPI 'integer' type.

key = 'integer'

marshmallow_class
alias of `marshmallow.fields.Integer`

class nadia.primitives.**PrimitiveBuilder**(*builder_provider*)
Bases: *nadia.common.Builder*

Base class for primitive fields builder.

classmethod build_schema(*spec*, ***kwargs*)
Build a Field for a primitive object.

Note: Conforming to the base class documentation, this method returns instances of `marshmallow.Field`.

---

key = None

marshmallow_class = None

class nadia.primitives.**StringBuilder**(*builder_provider*)
Bases: *nadia.primitives.PrimitiveBuilder*

Str schema builder.

This builder is designed for constructing schemas for OpenAPI 'string' type.

key = 'string'

marshmallow_class
alias of marshmallow.fields.String

3.2.1.8 nadia.schema module

Base schema used by Nadia.

class nadia.schema.**NadiaSchema**(*only=None*, *exclude=()*, *prefix=''*, *many=False*, *context=None*, *load_only=()*, *dump_only=()*, *partial=False*)
Bases: marshmallow.schema.Schema

Base marshmallow schema used by Nadia.

check_unknown_fields(*_data*, *original_data*)
Validator raising ValidationError if extra keys are provided.

This validator is taken from the the official recipe located at: http://marshmallow.readthedocs.io/en/latest/extending.html#validating-original-input-data

opts = <marshmallow.schema.SchemaOpts object>

Python Module Index

n

Index

T

U