Anyblok / Marshmallow Documentation

Release 2.3.0

Jean-Sébastien SUZANNE

Dec 03, 2019

Contents

Contents

CHAPTER 1

Front Matter

Information about the AnyBlok / Marshmallow project.

1.1 Project Homepage

AnyBlok is hosted on github - the main project page is at https://github.com/AnyBlok/AnyBlok_Marshmallow. Source code is tracked here using GIT.

Releases and project status are available on Pypi at http://pypi.python.org/pypi/anyblok_marshmallow.

The most recent published version of this documentation should be at http://doc.anyblok-marshmallow.anyblok.org.

1.2 Project Status

AnyBlok with Marshmallow is currently in beta status and is expected to be fairly stable. Users should take care to report bugs and missing features on an as-needed basis. It should be expected that the development version may be required for proper implementation of recently repaired issues in between releases;

1.3 Installation

Install released versions of AnyBlok from the Python package index with pip or a similar tool:

```
pip install anyblok_marshmallow
```

Installation via source distribution is via the `setup.py` script:

```
python setup.py install
```

Installation will add the `anyblok` commands to the environment.

1.4 Unit Test

Run the test with `pytest`:

```
pip install pytest pytest-cov
ANYBLOK_DRIVER_NAME=postgres ANYBLOK_DATABASE_NAME=mybase pytest anyblok_marshmallow/
↪tests
```

1.5 Dependencies

AnyBlok works with **Python 3.6** and later. The install process will ensure that AnyBlok, marshmallow >= 3.2.0 and marshmallow-sqlalchemy are installed, in addition to other dependencies. The latest version of them is strongly recommended.

1.6 Contributing (hackers needed!)

Anyblok / Marshmallow is at a very early stage, feel free to fork, talk with core dev, and spread the word!

1.7 Author

Jean-Sébastien Suzanne

1.8 Contributors

Anybox team:

- Jean-Sébastien Suzanne

Sensee team:

- Franck Bret

1.9 Bugs

Bugs and feature enhancements to AnyBlok should be reported on the Issue tracker.

Contents

        * *Add the **registry** by the class attribute*
        * *Add the **registry** during init*
        * *Add the **registry** by the context*
        * *Add the **registry** when the de(serialization or validator is called*
    - ***model** option*
    - ***only_primary_key** option*
    - ***required_fields** option*
    - *Use the field JsonCollection*

CHAPTER 2

Memento

2.1 Declare your AnyBlok model

```
from anyblok.column import Integer, String
from anyblok.relationship import Many2One, Many2Many
from anyblok import Declarations


@Declarations.register(Declarations.Model)
class City:

    id = Integer(primary_key=True)
    name = String(nullable=False)
    zipcode = String(nullable=False)

    def __repr__(self):
        return '<City(name={self.name!r})>'.format(self=self)


@Declarations.register(Declarations.Model)
class Tag:

    id = Integer(primary_key=True)
    name = String(nullable=False)

    def __repr__(self):
        return '<Tag(name={self.name!r})>'.format(self=self)


@Declarations.register(Declarations.Model)
class Customer:
    id = Integer(primary_key=True)
    name = String(nullable=False)
    tags = Many2Many(model=Declarations.Model.Tag)
```

(continues on next page)

(continued from previous page)

```
    def __repr__(self):
        return '<Customer(name={self.name!r}, '
               'tags={self.tags!r})>'.format(self=self)


@Declarations.register(Declarations.Model)
class Address:

    id = Integer(primary_key=True)
    street = String(nullable=False)
    city = Many2One(model=Declarations.Model.City, nullable=False)
    customer = Many2One(
        model=Declarations.Model.Customer, nullable=False,
        one2many="addresses")
```

Warning: The **AnyBlok model** must be declared in a blok

2.2 Declare your schema

```
from anyblok_marshmallow import SchemaWrapper, PostLoadSchema, Nested

class CitySchema(SchemaWrapper):
    model = 'Model.City'


class TagSchema(SchemaWrapper):
    model = 'Model.Tag'


class AddressSchema(SchemaWrapper):
    model = 'Model.Address'

    class Schema:
        # Add some marshmallow fields or behaviours

        # follow the relationship Many2One and One2One
        city = Nested(CitySchema)


class CustomerSchema(SchemaWrapper):
    model = 'Model.Customer'
    # optionally attach an AnyBlok registry
    # to use for serialization, desarialization and validation
    registry = registry

    class Schema(PostLoadSchema):
        # follow the relationship One2Many and Many2Many
        # - the many=True is required because it is *2Many
        # - exclude is used to forbid the recurse loop
        addresses = Nested(AddressSchema, many=True, exclude=('customer', ))
        tags = Nested(TagSchema, many=True)
```

(continues on next page)

(continued from previous page)

```
customer_schema = CustomerSchema()
```

Note: **New** in version **1.1.0** the Nested field must come from **anyblok_marshmallow**, because **marshmallow** cache the Nested field with the context. And the context is not propagated again if it changed

Note: **Ref** in version **1.4.0**, `post_load_return_instance` was replaced by the mixin class `PostLoadSchema`

Note: **Ref** in version **2.1.0**, `ModelSchema` was replaced by `SchemaWrapper`. This action break the compatibility with the previous version, but allow to follow the upgrade of **marshmallow**

2.3 (De)serialize your data and validate it

```
customer = registry.Customer.insert(name="JS Suzanne")
tag1 = registry.Tag.insert(name="tag 1")
customer.tags.append(tag1)
tag2 = registry.Tag.insert(name="tag 2")
customer.tags.append(tag2)
rouen = registry.City.insert(name="Rouen", zipcode="76000")
paris = registry.City.insert(name="Paris", zipcode="75000")
registry.Address.insert(customer=customer, street="Somewhere", city=rouen)
registry.Address.insert(customer=customer, street="Another place", city=paris)

dump_data = customer_schema.dump(customer).data
# {
#     'id': 1,
#     'name': 'JS Suzanne',
#     'tags': [
#         {
#             'id': 1,
#             'name': 'tag 1',
#         },
#         {
#             'id': 2,
#             'name': 'tag 2',
#         },
#     ],
#     'addresses': [
#         {
#             'id': 1
#             'street': 'Somewhere'
#             'city': {
#                 'id': 1,
#                 'name': 'Rouen',
#                 'zipcode': '76000',
#             },
```

(continues on next page)

(continued from previous page)

```
#         },
#         {
#             'id': 2
#             'street': 'Another place'
#             'city': {
#                 'id': 2,
#                 'name': 'Paris',
#                 'zipcode': '75000',
#             },
#         },
#     ],
# }

customer_schema.load(dump_data).data
# <Customer(name='JS Suzanne' tags=[<Tag(name='tag 1')>, <Tag (name='tag 2')>])>

errors = customer_schema.validate(dump_data)
# dict with all the validating errors
```

Note: We have an instance of the model cause of the mixin `PostLoadSchema`

2.4 Give the registry

The schema need to have the registry.

If no registry found when the de(serialization) or validation then the **RegistryNotFound** exception will be raised.

2.4.1 Add the registry by the class attribute

This is the solution given in the main exemple:

```
class CustomerSchema(SchemaWrapper):
    model = 'Model.Customer'
    registry = registry
```

2.4.2 Add the registry during init

This solution is use during the instanciation

```
customer_schema = CustomerSchema(registry=registry)
```

2.4.3 Add the registry by the context

This solution is use during the instanciation or after

```
customer_schema = CustomerSchema(context={'registry': registry})
```

or

```
customer_schema = CustomerSchema()
customer_schema.context['registry'] = registry
```

2.4.4 Add the registry when the de(serialization or validator is called

```
customer_schema.dumps(customer, registry=registry)
customer_schema.dump(customer, registry=registry)
customer_schema.loads(dump_data, registry=registry)
customer_schema.load(dump_data, registry=registry)
customer_schema.validate(dump_data, registry=registry)
```

2.5 model option

This option add in the model name. As the registry, this option can be passed by definition, initialization, context or during the call of the (de)serialization / validation

```
class AnySchema(SchemaWrapper):
    model = "Model.Customer"
```

or

```
any_schema = AnySchema(model="Model.customer")
```

or

```
any_schema.context['model'] = "Model.Customer"
```

or

```
any_schema.dumps(instance, model="Model.Customer")
any_schema.dump(instance, model="Model.Customer")
any_schema.loads(dump_data, model="Model.Customer")
any_schema.load(dump_data, model="Model.Customer")
any_schema.validate(dump_data, model="Model.Customer")
```

2.6 only_primary_key option

This option add in the only argument the primary keys of the model. As the registry, this option can be passed by definition, initialization, context or during the call of the (de)serialization / validation

```
class CustomerSchema(SchemaWrapper):
    model = "Model.Customer"
    only_primary_key = True
```

or

```
customer_schema = CustomerSchema(only_primary_key=True)
```

or

```
customer_schema.context['only_primary_key'] = True
```

or

```
customer_schema.dumps(instance, only_primary_key=True)
customer_schema.dump(instance, only_primary_key=True)
customer_schema.loads(dump_data, only_primary_key=True)
customer_schema.load(dump_data, only_primary_key=True)
customer_schema.validate(dump_data, only_primary_key=True)
```

2.7 required_fields option

This option force the generated fields, and only them to be requried.

```
class CustomerSchema(SchemaWrapper):
    model = "Model.Customer"
    required_fields = True
    # or required_fields = [ list of fieldname ]
```

or

```
customer_schema = CustomerSchema(required_fields=True)
```

or

```
customer_schema.context['required_fields'] = True
```

or

```
customer_schema.loads(dump_data, required_fields=True)
customer_schema.load(dump_data, required_fields=True)
customer_schema.validate(dump_data, required_fields=True)
```

Note: All the attributes can take **True** or the list of the fieldname to be required

2.8 Use the field JsonCollection

This field allow the schema to inspect an AnyBlok.fields.Json in an any specific instance to validate the value.

AnyBlok models:

```
@register(Model)
class Template:
    name = anyblok.column.String(primary_key=True)
    properties = anyblok.column.Json(defaumt={})

@register(Model)
class SaleOrder:
    id = anyblok.column.Integer(primary_key=True)
    number = anyblok.column.Integer(nullable=False)
    discount = anyblok.column.Integer()
```

AnyBlok / Marchmallow schema:

```
class SaleOrderSchema(SchemaWrapper):
    model = 'Model.SaleOrder'

    class Schema:
        discount = JsonCollection(
            fieldname='properties',
            keys=['allowed_discount'],
            cls_or_instance_type=marshmallow.fields.Integer(required=True)
        )
```

Use:

```
goodcustomer = registry.Template.insert(
    name='Good customer',
    properties={'allowed_discount': [10, 15, 30]
)
customer = registry.Template.insert(
    name='Customer',
    properties={'allowed_discount': [0, 5, 10]
)
badcustomer = registry.Template.insert(
    name='Bad customer',
    properties={'allowed_discount': [0]
)

schema = SaleOrderSchema(registry=registry)

---------------------------

data = schema.load(
    {
        number='SO0001',
        discount=10,
    },
    instances={'default': goodcustomer}
)

---------------------------

data = schema.load(
    {
        number='SO0001',
        discount=10,
    },
    instances={'default': customer}
)
==> error = {}

---------------------------

data = schema.load(
    {
        number='SO0001',
        discount=10,
    },
    instances={'default': badcustomer}
```

(continues on next page)

(continued from previous page)

```
)
==> error = {'discount': ['Not a valid choice']}
```

Contents

CHAPTER 3

Exceptions

3.1 RegistryNotFound

exception anyblok_marshmallow.exceptions.**RegistryNotFound**

Bases: Exception

Exception raised when no registry is found to build schema

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

CHAPTER 4

Fields

4.1 Nested

class anyblok_marshmallow.fields.**Nested**(*nested: Union[marshmallow.base.SchemaABC, type, str], *, default: Any = <marshmallow.missing>, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, unknown: str = None, **kwargs*)

Bases: marshmallow.fields.Nested

Inherit marshmallow fields.Nested

context
The context dictionary for the parent Schema.

deserialize(*value: Any*, *attr: str = None*, *data: Mapping[str, Any] = None*, ***kwargs*)
Deserialize value.

Parameters

- **value** – The value to deserialize.
- **attr** – The attribute/key in *data* to deserialize.
- **data** – The raw input data passed to *Schema.load*.
- **kwargs** – Field-specific keyword arguments.

Raises **ValidationError** – If an invalid value is passed or if a required value is missing.

fail(*key: str*, ***kwargs*)
Helper method that raises a *ValidationError* with an error message from self.error_messages.

Deprecated since version 3.0.0: Use *make_error* <*marshmallow.fields.Field.make_error*> instead.

get_value(*obj*, *attr*, *accessor=None*, *default=<marshmallow.missing>*)
Return the value for a given key from an object.

Parameters

- **obj** (*object*) – The object to get the value from.
- **attr** (*str*) – The attribute/key in *obj* to get the value from.
- **accessor** (*callable*) – A callable used to retrieve the value of *attr* from the object *obj*. Defaults to *marshmallow.utils.get_value*.

make_error(*key: str*, ***kwargs*) → marshmallow.exceptions.ValidationError
Helper method to make a *ValidationError* with an error message from `self.error_messages`.

root
Reference to the *Schema* that this field belongs to even if it is buried in a container field (e.g. *List*). Return *None* for unbound fields.

schema
Overload the super property to remove cache

it is the only way to propagate the context at each call

serialize(*attr: str, obj: Any, accessor: Callable[[Any, str, Any], Any] = None, **kwargs*)
Pulls the value for the given key from the object, applies the field's formatting and returns the result.

Parameters

- **attr** – The attribute/key to get from the object.
- **obj** – The object to access the attribute/key from.
- **accessor** – Function used to access values from `obj`.
- **kwargs** – Field-specific keyword arguments.

4.2 File

class anyblok_marshmallow.fields.**File**(**, default: Any = <marshmallow.missing>, missing: Any = <marshmallow.missing>, data_key: str = None, attribute: str = None, validate: Union[Callable[[Any], Any], Sequence[Callable[[Any], Any]], Generator[Callable[[Any], Any], None, None]] = None, required: bool = False, allow_none: bool = None, load_only: bool = False, dump_only: bool = False, error_messages: Dict[str, str] = None, **metadata*)

Bases: `marshmallow.fields.Field`

context
The context dictionary for the parent `Schema`.

deserialize(*value: Any*, *attr: str = None*, *data: Mapping[str, Any] = None*, ***kwargs*)
Deserialize `value`.

Parameters

- **value** – The value to deserialize.
- **attr** – The attribute/key in *data* to deserialize.
- **data** – The raw input data passed to *Schema.load*.
- **kwargs** – Field-specific keyword arguments.

Raises **ValidationError** – If an invalid value is passed or if a required value is missing.

fail(*key: str*, ***kwargs*)
Helper method that raises a *ValidationError* with an error message from `self.error_messages`.

Deprecated since version 3.0.0: Use *make_error* <*marshmallow.fields.Field.make_error*> instead.

get_value(*obj*, *attr*, *accessor=None*, *default=<marshmallow.missing>*)
Return the value for a given key from an object.

Parameters

- **obj** (*object*) – The object to get the value from.
- **attr** (*str*) – The attribute/key in *obj* to get the value from.
- **accessor** (*callable*) – A callable used to retrieve the value of *attr* from the object *obj*. Defaults to *marshmallow.utils.get_value*.

make_error(*key: str*, ***kwargs*) → marshmallow.exceptions.ValidationError
Helper method to make a *ValidationError* with an error message from `self.error_messages`.

root
Reference to the *Schema* that this field belongs to even if it is buried in a container field (e.g. *List*). Return *None* for unbound fields.

serialize(*attr: str, obj: Any, accessor: Callable[[Any, str, Any], Any] = None, **kwargs*)
Pulls the value for the given key from the object, applies the field's formatting and returns the result.

Parameters

- **attr** – The attribute/key to get from the object.
- **obj** – The object to access the attribute/key from.
- **accessor** – Function used to access values from `obj`.
- **kwargs** – Field-specific keyword arguments.

4.3 Text

class anyblok_marshmallow.fields.**Text**(**, default: Any = <marshmallow.missing>, missing: Any = <marshmallow.missing>, data_key: str = None, attribute: str = None, validate: Union[Callable[[Any], Any], Sequence[Callable[[Any], Any]], Generator[Callable[[Any], Any], None, None]] = None, required: bool = False, allow_none: bool = None, load_only: bool = False, dump_only: bool = False, error_messages: Dict[str, str] = None, **metadata*)
Bases: `marshmallow.fields.String`

Simple field use to distinct by the class String and Text

context
The context dictionary for the parent `Schema`.

deserialize(*value: Any*, *attr: str = None*, *data: Mapping[str, Any] = None*, ***kwargs*)
Deserialize `value`.

Parameters

- **value** – The value to deserialize.
- **attr** – The attribute/key in *data* to deserialize.
- **data** – The raw input data passed to *Schema.load*.
- **kwargs** – Field-specific keyword arguments.

Raises **ValidationError** – If an invalid value is passed or if a required value is missing.

fail(*key: str*, ***kwargs*)
Helper method that raises a *ValidationError* with an error message from `self.error_messages`.

Deprecated since version 3.0.0: Use *make_error* <*marshmallow.fields.Field.make_error*> instead.

get_value(*obj*, *attr*, *accessor=None*, *default=<marshmallow.missing>*)
Return the value for a given key from an object.

Parameters

- **obj** (*object*) – The object to get the value from.
- **attr** (*str*) – The attribute/key in *obj* to get the value from.
- **accessor** (*callable*) – A callable used to retrieve the value of *attr* from the object *obj*. Defaults to *marshmallow.utils.get_value*.

make_error(*key: str*, ***kwargs*) → marshmallow.exceptions.ValidationError
Helper method to make a *ValidationError* with an error message from `self.error_messages`.

root
Reference to the *Schema* that this field belongs to even if it is buried in a container field (e.g. *List*). Return *None* for unbound fields.

serialize(*attr: str, obj: Any, accessor: Callable[[Any, str, Any], Any] = None, **kwargs*)
Pulls the value for the given key from the object, applies the field's formatting and returns the result.

Parameters

- **attr** – The attribute/key to get from the object.
- **obj** – The object to access the attribute/key from.
- **accessor** – Function used to access values from `obj`.
- **kwargs** – Field-specific keyword arguments.

4.4 JsonCollection

class anyblok_marshmallow.fields.**JsonCollection**(*fieldname=None*, *keys=None*, *instance='default'*, *cls_or_instance_type=<class 'marshmallow.fields.String'>*, **args*, ***kwargs*)
Bases: `marshmallow.fields.Field`

context
The context dictionary for the parent `Schema`.

deserialize(*value: Any*, *attr: str = None*, *data: Mapping[str, Any] = None*, ***kwargs*)
Deserialize `value`.

Parameters

- **value** – The value to deserialize.
- **attr** – The attribute/key in *data* to deserialize.
- **data** – The raw input data passed to *Schema.load*.
- **kwargs** – Field-specific keyword arguments.

Raises **ValidationError** – If an invalid value is passed or if a required value is missing.

fail(*key: str*, ***kwargs*)
Helper method that raises a *ValidationError* with an error message from `self.error_messages`.

Deprecated since version 3.0.0: Use *make_error* <*marshmallow.fields.Field.make_error*> instead.

get_value(*obj*, *attr*, *accessor=None*, *default=<marshmallow.missing>*)
Return the value for a given key from an object.

Parameters

- **obj** (*object*) – The object to get the value from.
- **attr** (*str*) – The attribute/key in *obj* to get the value from.
- **accessor** (*callable*) – A callable used to retrieve the value of *attr* from the object *obj*. Defaults to *marshmallow.utils.get_value*.

make_error(*key: str*, ***kwargs*) → marshmallow.exceptions.ValidationError
Helper method to make a *ValidationError* with an error message from `self.error_messages`.

root
Reference to the *Schema* that this field belongs to even if it is buried in a container field (e.g. *List*). Return *None* for unbound fields.

serialize(*attr: str, obj: Any, accessor: Callable[[Any, str, Any], Any] = None, **kwargs*)
Pulls the value for the given key from the object, applies the field's formatting and returns the result.

Parameters

- **attr** – The attribute/key to get from the object.
- **obj** – The object to access the attribute/key from.
- **accessor** – Function used to access values from `obj`.
- **kwargs** – Field-specific keyword arguments.

4.5 PhoneNumer

4.6 Country

class anyblok_marshmallow.fields.**Country**(*mode=None*, *load_mode=<Modes.ALPHA_3: 'alpha_3'>*, *dump_mode=<Modes.ALPHA_3: 'alpha_3'>*, **args*, ***kwargs*)
Bases: `marshmallow.fields.String`

class Modes
Bases: `enum.Enum`

An enumeration.

context
The context dictionary for the parent `Schema`.

deserialize(*value: Any*, *attr: str = None*, *data: Mapping[str, Any] = None*, ***kwargs*)
Deserialize `value`.

Parameters

- **value** – The value to deserialize.
- **attr** – The attribute/key in *data* to deserialize.
- **data** – The raw input data passed to *Schema.load*.
- **kwargs** – Field-specific keyword arguments.

Raises **ValidationError** – If an invalid value is passed or if a required value is missing.

fail(*key: str*, ***kwargs*)
Helper method that raises a *ValidationError* with an error message from `self.error_messages`.

Deprecated since version 3.0.0: Use *make_error* <*marshmallow.fields.Field.make_error*> instead.

get_value(*obj*, *attr*, *accessor=None*, *default=<marshmallow.missing>*)
Return the value for a given key from an object.

Parameters

- **obj** (*object*) – The object to get the value from.
- **attr** (*str*) – The attribute/key in *obj* to get the value from.
- **accessor** (*callable*) – A callable used to retrieve the value of *attr* from the object *obj*. Defaults to *marshmallow.utils.get_value*.

make_error(*key: str*, ***kwargs*) → marshmallow.exceptions.ValidationError
Helper method to make a *ValidationError* with an error message from `self.error_messages`.

root
Reference to the *Schema* that this field belongs to even if it is buried in a container field (e.g. *List*). Return *None* for unbound fields.

serialize(*attr: str, obj: Any, accessor: Callable[[Any, str, Any], Any] = None, **kwargs*)
Pulls the value for the given key from the object, applies the field's formatting and returns the result.

Parameters

- **attr** – The attribute/key to get from the object.
- **obj** – The object to access the attribute/key from.
- **accessor** – Function used to access values from `obj`.
- **kwargs** – Field-specific keyword arguments.

4.7 InstanceField

class anyblok_marshmallow.fields.**Country**(*mode=None*, *load_mode=<Modes.ALPHA_3: 'alpha_3'>*, *dump_mode=<Modes.ALPHA_3: 'alpha_3'>*, **args*, ***kwargs*)
Bases: `marshmallow.fields.String`

class Modes
Bases: `enum.Enum`

An enumeration.

context
The context dictionary for the parent `Schema`.

deserialize(*value: Any*, *attr: str = None*, *data: Mapping[str, Any] = None*, ***kwargs*)
Deserialize `value`.

Parameters

- **value** – The value to deserialize.
- **attr** – The attribute/key in *data* to deserialize.
- **data** – The raw input data passed to *Schema.load*.
- **kwargs** – Field-specific keyword arguments.

Raises **ValidationError** – If an invalid value is passed or if a required value is missing.

fail(*key: str*, ***kwargs*)
Helper method that raises a *ValidationError* with an error message from `self.error_messages`.

Deprecated since version 3.0.0: Use *make_error* <*marshmallow.fields.Field.make_error*> instead.

get_value(*obj*, *attr*, *accessor=None*, *default=<marshmallow.missing>*)
Return the value for a given key from an object.

Parameters

- **obj** (*object*) – The object to get the value from.
- **attr** (*str*) – The attribute/key in *obj* to get the value from.
- **accessor** (*callable*) – A callable used to retrieve the value of *attr* from the object *obj*. Defaults to *marshmallow.utils.get_value*.

make_error(*key: str*, ***kwargs*) → marshmallow.exceptions.ValidationError
Helper method to make a *ValidationError* with an error message from `self.error_messages`.

root
Reference to the *Schema* that this field belongs to even if it is buried in a container field (e.g. *List*). Return *None* for unbound fields.

serialize(*attr: str, obj: Any, accessor: Callable[[Any, str, Any], Any] = None, **kwargs*)
Pulls the value for the given key from the object, applies the field's formatting and returns the result.

Parameters

- **attr** – The attribute/key to get from the object.
- **obj** – The object to access the attribute/key from.
- **accessor** – Function used to access values from `obj`.
- **kwargs** – Field-specific keyword arguments.

CHAPTER 5

Schema

5.1 update_from_kwargs

anyblok_marshmallow.schema.**update_from_kwargs**(**entries*)
decorator to get temporaly the value in kwargs and put it in schema

Params entries array ok entry name to take from the kwargs

5.2 format_field

anyblok_marshmallow.schema.**format_fields**(*x*)
remove the anyblok prefix form the field name

5.3 ModelConverter

class anyblok_marshmallow.schema.**ModelConverter**(*schema_cls=None*)
Bases: marshmallow_sqlalchemy.convert.ModelConverter

Overwrite the ModelConverter class of marshmallow-sqlalchemy

The goal if to fix the fieldname, because they are prefixed.

fields_for_model(*Model*, ***kwargs*)
Overwrite the method and remove prefix of the field name

5.4 TemplateSchema

class anyblok_marshmallow.schema.**TemplateSchema**
Bases: object

Base class of Schema generated by `SchemaWrapper`

OPTIONS_CLASS
alias of `marshmallow_sqlalchemy.schema.ModelSchemaOpts`

5.5 PostLoadSchema

class anyblok_marshmallow.schema.**PostLoadSchema**
Bases: `object`

Return the AnyBlok instance from marshmallow deserialize

5.6 SchemaWrapper

class anyblok_marshmallow.schema.**SchemaWrapper**(**args*, ***kwargs*)
Bases: `marshmallow.base.SchemaABC`

Schema Wrapper to generate marshmallow schema

```
class MySchema(SchemaWrapper):
    model = 'Model.Name'
```

the wrapper implement the **marshmallow.base.SchemaABC** abstract class. And call the methods of the marshmallow schema with the same parameter

Some class attributes can be added to improve the schema:

- **model**: str, the registry name of the AnyBlok model
- **required_fields**: list of the field which become required In this case the anyblok columns are not required but schema force them to be required
- registry: the anyblok registry, only if you know it
- only_primary_key: boolean, if True the marshmallow parameter only will be filled with the name of the primary keys.

Note: The model and registry are required to generate the schema. they can be defined by class attribute, parameter in the methods (load, loads, validate, dump, dumps) or in the context attribute.

---

generate_marsmallow_instance
Generate the real mashmallow-sqlalchemy schema

schema
property to get the real schema

Contents

CHAPTER 6

CHANGELOG

6.1 2.3.1 (Unreleased)

- Improved Country field to add parametrization. Allowed modes are alpha 3, alpha 2, numeric, name and official name.

6.2 2.3.0 (2019-10-31)

- Fix Marshmallow dependency to 3.2.1 release

6.3 2.2.4 (2019-09-09)

- Fix Marshmallow dependency to 3.0.3 release
- Drop support for python 3.4 and 3.5

6.4 2.2.3 (2019-05-06)

- Fix Marshmallow dependency to 3.0.0RC5 release
- Refactored unittest from nosetest to pytest

6.5 2.2.2 (2018-12-06)

- Fix Marshmallow dependency to 3.0.0RC1 release

6.6 2.2.1 (2018-10-26)

- Fix Marshmallow dependency to 3.0.0b19 release

6.7 2.2.0 (2018-10-17)

- Fixed the conversion of type between **AnyBlok.Column** and **marshmallow.Field**

6.8 2.1.0 (2018-09-26)

- Fixed the compatibility with **Marshmallow > 3.0.0b8**
- Removed `ModelSchema` class
- Added `SchemaWrapper`, this is the best way to defined a generated schema with the **marshmallow_sqlalchemy** library

Warning: This version break the compatibility with previous version, in the only goal to be adapted with the latest version of **marshmallow**

6.9 2.0.1 (2018-06-07)

- Fix required_field put allow_none to False

6.10 2.0.0 (2018-05-30)

- Add JsonCollection field, Allow to add a check in function of an collection stored in a AnyBlok.fields.Json
- Add Text field, to represent an `anyblok.column.Text`
- Migration of the code and unit test to marshmallow 3.0.0
- Add Email matching for `anyblok.column.Email`
- Add URL matching for `anyblok.column.URL`
- Add PhoneNumber matching for `anyblok.column.PhoneNumber`
- Add Country matching for `anyblok.column.Country`
- Add required_fields option
- Add InstanceField

6.11 1.4.0 (2018-04-07)

- Replace **post_load_return_instance** method by **PostLoadSchema** class

- In the case of the field **Selection**, the validator **OneOf** is applied with the available values come from the AnyBlok columns
- Replace **marshmallow_sqlalchemy.fields.Related** by **anyblok_marshmallow.fields.Nested**. The goal is to improve the consistent between all field in the schema

6.12 1.3.0 (2017-12-23)

- [ADD] unittest on some case
- [FIX] AnyBlok field.Function is return as MarshMallow fields.Raw
- [ADD] fields.File, type to encode and decode to/from base 64

6.13 1.2.0 (2017-11-30)

- [REF] decrease complexity
- [IMP] Add `validates_schema` on ModelSchema to automaticly check if the field exist on the model

6.14 1.1.0 (2017-11-02)

- Add option put only the primary keys
- Fix the Front page
- REF model option, can be given by another way than Meta
- Put RegistryNotFound in exceptions
- Add Nested field, this field is not and have not to be cached

6.15 1.0.2 (2017-10-25)

- Fix pypi documentation

6.16 1.0.0 (2017-10-24)

- Add marshmallow schema for AnyBlok for:
  - Serialization
  - Deserialization
  - Validation

CHAPTER 7

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

A