Anyblok / Marshmallow Documentation

Release 1.0.2

Jean-Sébastien SUZANNE

Oct 25, 2017

Contents

Contents

CHAPTER 1

Front Matter

Information about the AnyBlok / Marshmallow project.

Project Homepage

AnyBlok is hosted on github - the main project page is at https://githusb.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.

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;

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.

Unit Test

Run the test with `nose`:

```
pip install nose
nosetests anyblok_marshmallow/tests
```

Dependencies

AnyBlok works with **Python 3.3** and later. The install process will ensure that AnyBlok, marshmallow and marshmallow-sqlalchemy are installed, in addition to other dependencies. The latest version of them is strongly recommended.

Contributing (hackers needed!)

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

Author

Jean-Sébastien Suzanne

Contributors

Anybox team:

- Jean-Sébastien Suzanne

Sensee team:

- Franck Bret

Bugs

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

Contents

CHAPTER 2

Memento

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)
```

```
    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

Declare your schema

```
from anyblok_marshmallow import ModelSchema
from marshmallow import fields

class CitySchema(ModelSchema):

    class Meta:
        model = 'Model.City'


class TagSchema(ModelSchema):

    class Meta:
        model = 'Model.Tag'


class AddressSchema(ModelSchema):

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

    class Meta:
        model = 'Model.Address'


class CustomerSchema(ModelSchema):

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

    class Meta:
```

```
        model = 'Model.Customer'
        # optionally attach an AnyBlok registry
        # to use for serialization, desarialization and validation
        registry = registry
        # optionally return an AnyBlok model instance
        post_load_return_instance = True


customer_schema = CustomerSchema()
```

(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',
#             },
#         },
#         {
#             '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: By default: the deserialization return a dict with deserialized data, here we get an instance of the model because the `CustomerSchema` add `post_load_return_instance = True` in their Meta

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.

Add the registry by the Meta

This is the solution given in the main exemple:

```
class CustomerSchema(ModelSchema):

    class Meta:
        model = 'Model.Customer'
        registry = registry
```

Add the registry during init

This solution is use during the instanciation

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

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
```

Add the registry when the de(serialization or validatoris called

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

post_load_return_instance option

As the registry this option can be passed by initialization of the schema, by the context or during the call of methods

The value of this options can be:

- False: **default**, the output is a dict
- True: the output is an instance of the model. The primary keys must be in value
- array of string: the output is an instance of the model, each str entry must be an existing column

Warning: If the option is not False, and the instance can no be found, then the **instance** error will be added in the errors dict of the method

Warning: The post load is only for load method!!!

Overriding Generated Fields

```
from anyblok_marshmallow import ModelSchema
from marshmallow import fields

class Customer(ModelSchema):

    date_created = field_for(Author, 'date_created', dump_only=True)

    class Meta:
        model = 'Model.Customer'
```

Contents

- *ModelSchema*

CHAPTER 3

Code

Exceptions

RegistryNotFound

exception anyblok_marshmallow.schema.**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.

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

format_field

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

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

ModelSchemaOpts

class anyblok_marshmallow.schema.**ModelSchemaOpts**(*meta*, **args*, ***kwargs*)
Bases: marshmallow.schema.SchemaOpts

Model schema option for Model schema

Add get option from the Meta:

- model: name of an AnyBlok model **required**
- registry: an AnyBlok registry
- post_load_return_instance: return an instance object

ModelSchema

class anyblok_marshmallow.schema.**ModelSchema**(**args*, ***kwargs*)
Bases: marshmallow.schema.Schema

A marshmallow schema based on the AnyBlok Model

Wrap the real schema, because at the instanciation the registry is not available

class Meta
Bases: object

Options object for a Schema.

Example usage:

```
class Meta:
    fields = ("id", "email", "date_created")
    exclude = ("password", "secret_attribute")
```

Available options:

- `fields`: Tuple or list of fields to include in the serialized result.
- **`additional`: Tuple or list of fields to include *in addition* to the** explicitly declared fields. `additional` and `fields` are mutually-exclusive options.
- **`include`: Dictionary of additional fields to include in the schema. It is** usually better to define fields as class variables, but you may need to use this option, e.g., if your fields are Python keywords. May be an *OrderedDict*.
- **`exclude`: Tuple or list of fields to exclude in the serialized result.** Nested fields can be represented with dot delimiters.
- **`dateformat`: Date format for all DateTime fields that do not have their** date format explicitly specified.
- **`strict`: If *True*, raise errors during marshalling rather than** storing them.

- **json_module: JSON module to use for *loads* and *dumps*.** Defaults to the `json` module in the stdlib.
- **ordered: If *True*, order serialization output according to the** order in which fields were declared. Output of *Schema.dump* will be a *collections.OrderedDict*.
- **index_errors: If *True*, errors dictionaries will include the index** of invalid items in a collection.
- `load_only`: Tuple or list of fields to exclude from serialized results.
- `dump_only`: Tuple or list of fields to exclude from deserialization

`ModelSchema.`**`OPTIONS_CLASS`**
alias of `ModelSchemaOpts`

`ModelSchema.`**`accessor`**(*func*)
Decorator that registers a function for pulling values from an object to serialize. The function receives the `Schema` instance, the `key` of the value to get, the `obj` to serialize, and an optional `default` value.

Deprecated since version 2.0.0: Set the `error_handler` class Meta option instead.

`ModelSchema.`**`dumps`**(*obj*, *many=None*, *update_fields=True*, **args*, ***kwargs*)
Same as `dump()`, except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.
- **many** (*bool*) – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **update_fields** (*bool*) – Whether to update the schema's field classes. Typically set to *True*, but may be *False* when serializing a homogenous collection. This parameter is used by *fields.Nested* to avoid multiple updates.

Returns A tuple of the form (`data`, `errors`)

Return type *MarshalResult*, a *collections.namedtuple*

New in version 1.0.0.

`ModelSchema.`**`error_handler`**(*func*)
Decorator that registers an error handler function for the schema. The function receives the `Schema` instance, a dictionary of errors, and the serialized object (if serializing data) or data dictionary (if deserializing data) as arguments.

Example:

```
class UserSchema(Schema):
    email = fields.Email()

@UserSchema.error_handler
def handle_errors(schema, errors, obj):
    raise ValueError('An error occurred while marshalling {}'.format(obj))

user = User(email='invalid')
UserSchema().dump(user)  # => raises ValueError
UserSchema().load({'email': 'bademail'})  # raises ValueError
```

New in version 0.7.0.

Deprecated since version 2.0.0: Set the `error_handler` class Meta option instead.

ModelSchema.**generate_marsmallow_instance**(*registry*, *post_load_return_instance*)
Generate the real mashmallow-sqlalchemy schema

ModelSchema.**get_attribute**(*attr*, *obj*, *default*)
Defines how to pull values from an object to serialize.

New in version 2.0.0.

ModelSchema.**handle_error**(*error*, *data*)
Custom error handler function for the schema.

Parameters

- **error** (*ValidationError*) – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.

New in version 2.0.0.

ModelSchema.**loads**(*json_data*, *many=None*, **args*, ***kwargs*)
Same as `load()`, except it takes a JSON string as input.

Parameters

- **json_data** (*str*) – A JSON string of the data to deserialize.
- **many** (*bool*) – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** (*bool | tuple*) – Whether to ignore missing fields. If *None*, the value for *self.partial* is used. If its value is an iterable, only missing fields listed in that iterable will be ignored.

Returns A tuple of the form (`data`, `errors`)

Return type *UnmarshalResult*, a *collections.namedtuple*

New in version 1.0.0.

ModelSchema.**on_bind_field**(*field_name*, *field_obj*)
Hook to modify a field when it is bound to the *Schema*. No-op by default.

ModelSchema.**schema**
property to get the real schema

Contents

CHAPTER 4

CHANGELOG

1.0.2 (2017-10-25)

- Fix pypi documentation

1.0.0 (2017-10-24)

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

Contents

- *Exhibit A - Source Code Form License Notice*
- *Exhibit B - "Incompatible With Secondary Licenses" Notice*

CHAPTER 5

Mozilla Public License Version 2.0

1. Definitions

1.1. “Contributor”

Means each individual or legal entity that creates, contributes to the creation of, or owns Covered Software.

1.2. “Contributor Version”

Means the combination of the Contributions of others (if any) used by a Contributor and that particular Contributor’s Contribution.

1.3. “Contribution”

Means Covered Software of a particular Contributor.

1.4. “Covered Software”

Means Source Code Form to which the initial Contributor has attached the notice in Exhibit A, the Executable Form of such Source Code Form, and Modifications of such Source Code Form, in each case including portions thereof.

1.5. “Incompatible With Secondary Licenses”

Means:

- **That the initial Contributor has attached the notice described in Exhibit B** to the Covered Software; or
- **That the Covered Software was made available under the terms of version 1.1** or earlier of the License, but not also under the terms of a Secondary License.

1.6. “Executable Form”

Means any form of the work other than Source Code Form.

1.7. “Larger Work”

Means a work that combines Covered Software with other material, in a separate file or files, that is not Covered Software.

1.8. “License”

Means this document.

1.9. “Licensable”

Means having the right to grant, to the maximum extent possible, whether at the time of the initial grant or subsequently, any and all of the rights conveyed by this License.

1.10. “Modifications”

Means any of the following:

- **Any file in Source Code Form that results from an addition to, deletion from,** or modification of the contents of Covered Software; or
- Any new file in Source Code Form that contains any Covered Software.

1.11. “Patent Claims” of a Contributor

Means any patent claim(s), including without limitation, method, process, and apparatus claims, in any patent Licensable by such Contributor that would be infringed, but for the grant of the License, by the making, using, selling, offering for sale, having made, import, or transfer of either its Contributions or its Contributor Version.

1.12. “Secondary License”

Means either the GNU General Public License, Version 2.0, the GNU Lesser General Public License, Version 2.1, the GNU Affero General Public License, Version 3.0, or any later versions of those licenses.

1.13. “Source Code Form”

Means the form of the work preferred for making modifications.

1.14. “You” (or “Your”)

Means an individual or a legal entity exercising rights under this License. For legal entities, “You” includes any entity that controls, is controlled by, or is under common control with You. For purposes of this definition, “control” means (a) the power, direct or indirect, to cause the direction or management of such entity, whether by contract or otherwise, or (b) ownership of more than fifty percent (50%) of the outstanding shares or beneficial ownership of such entity.

2. License Grants and Conditions

2.1. Grants

Each Contributor hereby grants You a world-wide, royalty-free, non-exclusive license:

- **Under intellectual property rights (other than patent or trademark)** Licensable by such Contributor to use, reproduce, make available, modify, display, perform, distribute, and otherwise exploit its Contributions, either on an unmodified basis, with Modifications, or as part of a Larger Work; and
- **Under Patent Claims of such Contributor to make, use, sell, offer for sale,** have made, import, and otherwise transfer either its Contributions or its Contributor Version.

2.2. Effective Date

The licenses granted in Section 2.1 with respect to any Contribution become effective for each Contribution on the date the Contributor first distributes such Contribution.

2.3. Limitations on Grant Scope

The licenses granted in this Section 2 are the only rights granted under this License. No additional rights or licenses will be implied from the distribution or licensing of Covered Software under this License. Notwithstanding Section 2.1(b) above, no patent license is granted by a Contributor:

- For any code that a Contributor has removed from Covered Software; or
- **For infringements caused by: (i) Your and any other third party's** modifications of Covered Software, or (ii) the combination of its Contributions with other software (except as part of its Contributor Version); or
- **Under Patent Claims infringed by Covered Software in the absence of its** Contributions.

This License does not grant any rights in the trademarks, service marks, or logos of any Contributor (except as may be necessary to comply with the notice requirements in Section 3.4).

2.4. Subsequent Licenses

No Contributor makes additional grants as a result of Your choice to distribute the Covered Software under a subsequent version of this License (see Section 10.2) or under the terms of a Secondary License (if permitted under the terms of Section 3.3).

2.5. Representation

Each Contributor represents that the Contributor believes its Contributions are its original creation(s) or it has sufficient rights to grant the rights to its Contributions conveyed by this License.

2.6. Fair Use

This License is not intended to limit any rights You have under applicable copyright doctrines of fair use, fair dealing, or other equivalents.

2.7. Conditions

Sections 3.1, 3.2, 3.3, and 3.4 are conditions of the licenses granted in Section 2.1.

3. Responsibilities

3.1. Distribution of Source Form

All distribution of Covered Software in Source Code Form, including any Modifications that You create or to which You contribute, must be under the terms of this License. You must inform recipients that the Source Code Form of the Covered Software is governed by the terms of this License, and how they can obtain a copy of this License. You may not attempt to alter or restrict the recipients' rights in the Source Code Form.

3.2. Distribution of Executable Form

If You distribute Covered Software in Executable Form then:

- **Such Covered Software must also be made available in Source Code Form, as** described in Section 3.1, and You must inform recipients of the Executable Form how they can obtain a copy of such Source Code Form by reasonable means in a timely manner, at a charge no more than the cost of distribution to the recipient; and
- **You may distribute such Executable Form under the terms of this License, or** sublicense it under different terms, provided that the license for the Executable Form does not attempt to limit or alter the recipients' rights in the Source Code Form under this License.

3.3. Distribution of a Larger Work

You may create and distribute a Larger Work under terms of Your choice, provided that You also comply with the requirements of this License for the Covered Software. If the Larger Work is a combination of Covered Software with a work governed by one or more Secondary Licenses, and the Covered Software is not Incompatible With Secondary Licenses, this License permits You to additionally distribute such Covered Software under the terms of such Secondary License(s), so that the recipient of the Larger Work may, at their option, further distribute the Covered Software under the terms of either this License or such Secondary License(s).

3.4. Notices

You may not remove or alter the substance of any license notices (including copyright notices, patent notices, disclaimers of warranty, or limitations of liability) contained within the Source Code Form of the Covered Software, except that You may alter any license notices to the extent required to remedy known factual inaccuracies.

3.5. Application of Additional Terms

You may choose to offer, and to charge a fee for, warranty, support, indemnity or liability obligations to one or more recipients of Covered Software. However, You may do so only on Your own behalf, and not on behalf of any Contributor. You must make it absolutely clear that any such warranty, support, indemnity, or liability obligation is offered by You alone, and You hereby agree to indemnify every Contributor for any liability incurred by such Contributor as a result of warranty, support, indemnity or liability terms You offer. You may include additional disclaimers of warranty and limitations of liability specific to any jurisdiction.

4. Inability to Comply Due to Statute or Regulation

If it is impossible for You to comply with any of the terms of this License with respect to some or all of the Covered Software due to statute, judicial order, or regulation then You must: (a) comply with the terms of this License to the maximum extent possible; and (b) describe the limitations and the code they affect. Such description must be placed in a text file included with all distributions of the Covered Software under this License. Except to the extent prohibited by statute or regulation, such description must be sufficiently detailed for a recipient of ordinary skill to be able to understand it.

5. Termination

5.1.

The rights granted under this License will terminate automatically if You fail to comply with any of its terms. However, if You become compliant, then the rights granted under this License from a particular Contributor are reinstated (a) provisionally, unless and until such Contributor explicitly and finally terminates Your grants, and (b) on an ongoing basis, if such Contributor fails to notify You of the non-compliance by some reasonable means prior to 60 days after You have come back into compliance. Moreover, Your grants from a particular Contributor are reinstated on an ongoing basis if such Contributor notifies You of the non-compliance by some reasonable means, this is the first time You have received notice of non-compliance with this License from such Contributor, and You become compliant prior to 30 days after Your receipt of the notice.

5.2.

If You initiate litigation against any entity by asserting a patent infringement claim (excluding declaratory judgment actions, counter-claims, and cross-claims) alleging that a Contributor Version directly or indirectly infringes any patent, then the rights granted to You by any and all Contributors for the Covered Software under Section 2.1 of this License shall terminate.

5.3.

In the event of termination under Sections 5.1 or 5.2 above, all end user license agreements (excluding distributors and resellers) which have been validly granted by You or Your distributors under this License prior to termination shall survive termination.

6. Disclaimer of Warranty

Warning: Covered Software is provided under this License on an "as is" basis, without warranty of any kind, either expressed, implied, or statutory, including, without limitation, warranties that the Covered Software is free of defects, merchantable, fit for a particular purpose or non-infringing. The entire risk as to the quality and performance of the Covered Software is with You. Should any Covered Software prove defective in any respect, You (not any Contributor) assume the cost of any necessary servicing, repair, or correction. This disclaimer of warranty constitutes an essential part of this License. No use of any Covered Software is authorized under this License except under this disclaimer.

7. Limitation of Liability

Warning: Under no circumstances and under no legal theory, whether tort (including negligence), contract, or otherwise, shall any Contributor, or anyone who distributes Covered Software as permitted above, be liable to You for any direct, indirect, special, incidental, or consequential damages of any character including, without limitation, damages for lost profits, loss of goodwill, work stoppage, computer failure or malfunction, or any and all other commercial damages or losses, even if such party shall have been informed of the possibility of such damages. This limitation of liability shall not apply to liability for death or personal injury resulting from such party's negligence to the extent applicable law prohibits such limitation. Some jurisdictions do not allow the exclusion or limitation of incidental or consequential damages, so this exclusion and limitation may not apply to You.

8. Litigation

Any litigation relating to this License may be brought only in the courts of a jurisdiction where the defendant maintains its principal place of business and such litigation shall be governed by laws of that jurisdiction, without reference to its conflict-of-law provisions. Nothing in this Section shall prevent a party's ability to bring cross-claims or counter-claims.

9. Miscellaneous

This License represents the complete agreement concerning the subject matter hereof. If any provision of this License is held to be unenforceable, such provision shall be reformed only to the extent necessary to make it enforceable. Any law or regulation which provides that the language of a contract shall be construed against the drafter shall not be used to construe this License against a Contributor.

10. Versions of the License

10.1. New Versions

Mozilla Foundation is the license steward. Except as provided in Section 10.3, no one other than the license steward has the right to modify or publish new versions of this License. Each version will be given a distinguishing version number.

10.2. Effect of New Versions

You may distribute the Covered Software under the terms of the version of the License under which You originally received the Covered Software, or under the terms of any subsequent version published by the license steward.

10.3. Modified Versions

If you create software not governed by this License, and you want to create a new license for such software, you may create and use a modified version of this License if you rename the license and remove any references to the name of the license steward (except to note that such modified license differs from this License).

10.4. Distributing Source Code Form that is Incompatible With Secondary Licenses

If You choose to distribute Source Code Form that is Incompatible With Secondary Licenses under the terms of this version of the License, the notice described in Exhibit B of this License must be attached.

Exhibit A - Source Code Form License Notice

```
This Source Code Form is subject to the terms of the Mozilla Public
License, v. 2.0. If a copy of the MPL was not distributed with this file,
You can obtain one at http://mozilla.org/MPL/2.0/.
```

If it is not possible or desirable to put the notice in a particular file, then You may include the notice in a location (such as a LICENSE file in a relevant directory) where a recipient would be likely to look for such a notice.

Note: You may add additional accurate notices of copyright ownership.

Exhibit B - “Incompatible With Secondary Licenses” Notice

```
This Source Code Form is “Incompatible With Secondary Licenses”, as defined
by the Mozilla Public License, v. 2.0.
```

CHAPTER 6

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

A